To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 22 November 2005
Subject: 05-603r0 SAM-4 LUN representation format

Revision history
Revision 0 (22 November 2005) First revision

Related documents
sam4r04 - SCSI Architecture Model - 4 revision 4
IETF RFC 4173 - iSCSI Boot strapping

Overview
Various OSes and management APIs are handling the 8-byte logical unit number (LUN) differently. Some treat it as an array of 8 bytes (e.g., CSMI/SDI, linux), while others (e.g. EFI, FC-HBA, SM-HBA) treat it as a single 8-byte value. If the system is big-endian, this difference doesn't matter. If the system is little-endian, however, those treating it as a single 8-byte value end up swapping all the bytes.

A LUN with byte 0, 1, ... 7 equal to 00 01 00 00 00 00 00 00 (peripheral addressing type LUN #1) is thus stored as either:

a) 00 01 00 00 00 00 00 00 in big-endian systems or little endian treating it as an array of bytes
b) 00 00 00 00 00 00 01 00 in little-endian systems treating it as an 8-byte value.

Internal representations don't really matter; software knows what system it's in and can swap the bytes if needed to exchange the LUN with other software (or send it out on a wire over a SCSI protocol). However, it is unclear how the LUN should be presented (e.g., displayed) to a user. The user might see either order in a graphical user interface, log file, configuration file, etc.:

a) 00 01 00 00 00 00 00 00 or
b) 00 00 00 00 00 00 01 00

Different systems will use minor differences in formatting conventions regarding spaces, “0x” prefixes, “h” suffixes, etc. When displaying 00 01 00 00 00 00 00 00, the LUN could appear as:

a) 00 01 00 00 00 00 00 00
b) 0001000000000000
c) 0001000000000000h
d) 0x0001000000000000
e) 0001 0000 0000 0000
f) 0001-0000-0000-0000 (iSCSI Bootstrapping format)
g) 1-0-0-0 (iSCSI Bootstrapping format)
h) 1 (iSCSI Bootstrapping format)

If a user sees a LUN displayed in the opposite order 00 00 00 00 00 00 01 00, and then tries to use that value literally in a configuration file used by an application that expects to see 00 01 00 00 00 00 00 00, the LUN would not be valid.

SAM-4 should recommend one approach. Displaying it like a single-value seems the most common and straightforward. Internally, representing it as an array of bytes is probably best.

Array of 8 bytes approach:
a) SCSI Architecture Model (SAM-4) and the SCSI transport protocol standards
b) SCSI Driver Interface (SDI) revision 0
c) Linux: Although it doesn't support 8-byte LUNs yet, there are several currently proposed patches that treat the LUN as an 8-byte array.

Single 8-byte value approach:

a) T11.5 SM-HBA and FC-HBA
b) Extensible Firmware Interface (EFI)
c) Microsoft Windows iSCSI driver (the general SCSI stack does not support 8-byte LUNs yet)

d) iSCSI Bootstrapping (RFC 4173): The DHCP path string uses a case-insensitive format like 4752-3A4F-6b7e-2F99, allowing up to 3 leading 0s in each group to be omitted and allowing trailing groups with all zeros to be completely eliminated (so 4186-0009-0000-0000 is equivalent to 4189-9)

Unclear:

a) T13 BIOS Enhanced Disk Drive Services (EDD-3): The SCSI, FIBRE, and SAS device paths are unclear. They define a QWord Logical Unit Number. Other QWords are also unclear, like the 64-bit SAS Address, 64-bit Worldwide Identifier, and 64-bit Extended Unique Identifier.

Suggested changes to SAM-4

4.9 Logical unit numbers

4.9.1 Introduction

~~See~~Subclause 4.9 ~~for a definition~~defines of the construction of logical unit numbers to be used by SCSI target devices. Application clients should use only those logical unit numbers returned by a REPORT LUNS command. The task router shall respond to logical unit numbers other than those returned by a REPORT LUNS command (i.e., incorrect logical unit numbers) as specified in 5.8.4 and 7.10.

The 64-bit quantity called a LUN is the logical unit number object defined by this standard. The fields containing the acronym LUN that compose the logical unit number object are historical nomenclature anomalies, not logical unit number objects. Logical unit number objects having different values represent different logical units, regardless of any implications to the contrary in 4.9 (e.g., LUN 00000000 00000000h is a different logical unit from LUN 40000000 00000000h and LUN 00FF0000 00000000h is a different logical unit from LUN 40FF0000 00000000h).

4.9.2 Logical unit numbers overview

All logical unit number formats described in this standard are hierarchical in structure even when only a single level in that hierarchy is used. The HISUP bit shall be set to one in the standard INQUIRY data (see SPC-3) when any logical unit number format described in this standard is used. Non-hierarchical formats are outside the scope of this standard.

A logical unit number shall contain 64 bits or 16 bits, with the size being defined by the SCSI transport protocol. For SCSI transport protocols that define 16-bit logical unit numbers, the two bytes shall be formatted as described for the FIRST LEVEL ADDRESSING field (see table 6 in 4.9.5).

4.9.x Logical unit representation format

When an application client displays or otherwise makes a 64-bit LUN value visible to a user, it should display it in hexadecimal with byte 0 first (i.e., on the left) and byte 7 last (i.e., on the right), regardless of the internal representation of the LUN value (e.g., a single level LUN with an ADDRESS METHOD field set to 01b (i.e., flat space addressing) and a FLAT SPACE LUN field set to 0001h should be displayed as 40 01 00 00 00 00 00 00h, not 00 00 00 00 00 00 01 40h). Spaces or dashes may be included between each byte, each two bytes (e.g., 4001-0000-0000-0000h), or each four bytes (e.g., 40010000 00000000h).

When displaying a single level 64-bit LUN value, an application client may display it as a single 2-byte value representing only the first level LUN (e.g., 40 01h). A space or dash may be included between each byte.

When displaying a 16-bit LUN value, an application client should display it as a single 2-byte value (e.g., 40 01h). A space or dash may be included between each byte.