To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 27 February 2004
Subject: 03-364r1 MSC Report Bridge Mapping command

**Revision history**
Revision 0 (23 October 2003) First revision
Revision 1 (27 February 2004) Incorporated comments from November CAP WG and 19 November conference call - support 1 to n mappings in one data structure, added more examples, support initiator port-specific maps.

**Related documents**
03-354 SPC-3 Specify initiator ports in EXTENDED COPY target descriptors (Rob Elliott, HP)
03-344 SPC-3 Report all initiator port and target port identifiers (Rob Elliott, HP)
03-242 MSC presentation (George Penokie, IBM)
02-037 MSC Management commands proposal (Bob Griswold, Crossroads)
01-151 BCC project proposal issues (Rob Elliott, HP)
01-095 Project proposal for Management Server Commands (MSC) (Rob Elliott, HP)

**Overview**
Today, SCSI transport protocol bridges (a.k.a. gateways or routers)(e.g. iSCSI to Fibre Channel) are invisible. This generally works fine for basic read and write traffic, but often fails to support more sophisticated SCSI commands. Thse bridges are limited to supporting only commands they know about, and often do not support complicated commands like EXTENDED COPY.

Bridges should to be visible to SCSI applications. This reduces or eliminates the need for bridges to intercept and change data in commands ranging from INQUIRY to EXTENDED COPY.

Although this information might be available via out-of-band management structures such as CIM, it’s difficult to guarantee that the SCSI application has access to that data. If the the application can generate SCSI commands that go through the bridge, it is very likely that it can generate SCSI commands to the bridge itself. Thus, a SCSI command set to talk to bridges, MSC, is proposed.

Figure 1 shows an example iSCSI to FCP bridge, showing initiator port, target port, and logical unit objects. SCSI device objects are not shown (they are only evident in the INQUIRY command VPD data and are not used for routing in the domains).

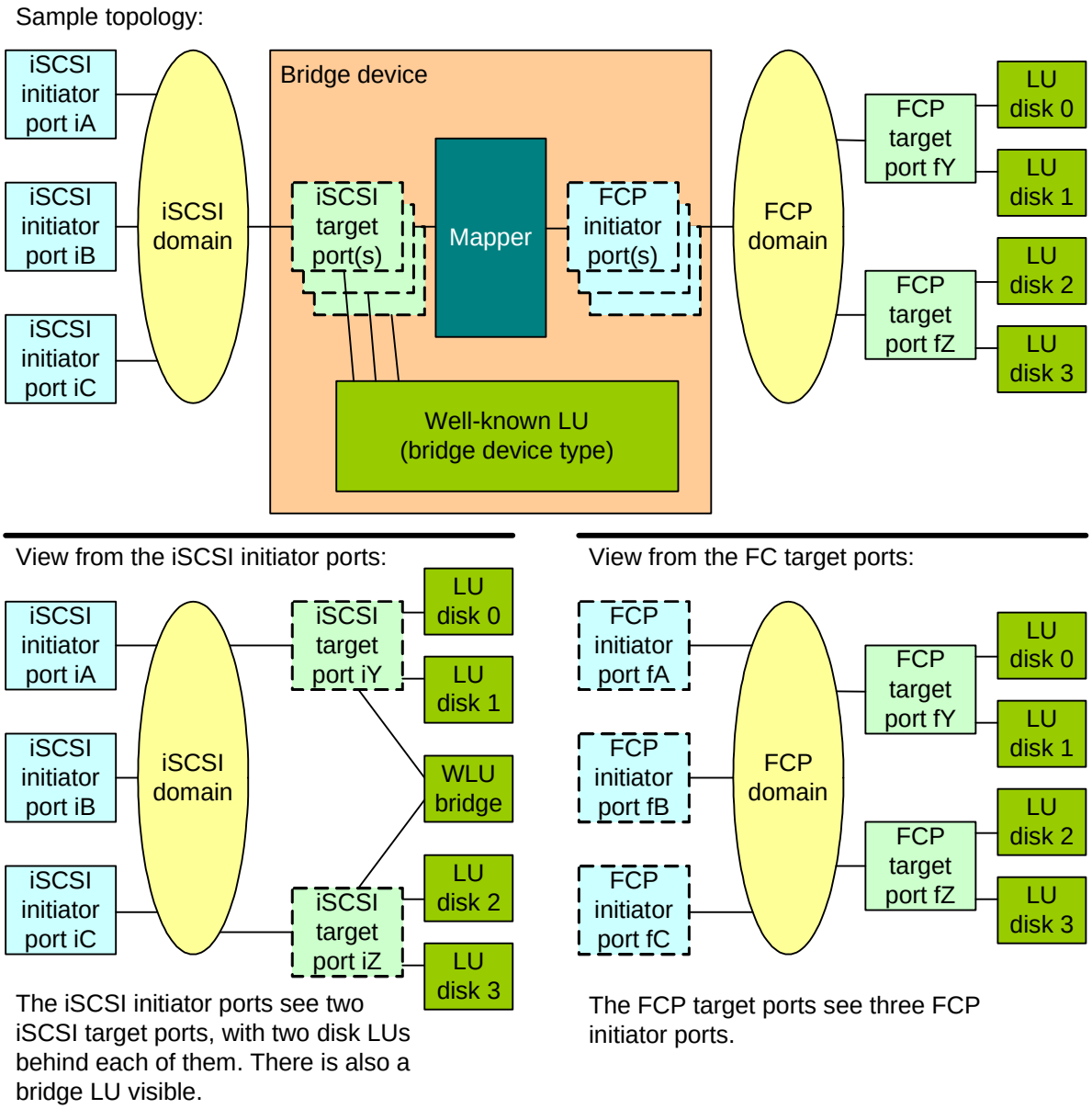



**Figure 1 — iSCSI to FCP bridge**

To enable communication between the iSCSI initiator ports and the FCP target ports, the bridge maps:

a) the FCP target ports into iSCSI target ports in the iSCSI domain (target mapping); and
b) the iSCSI initiator ports into FCP initiator ports in the FCP domain (initiator mapping).

An FCP target port thinks that an FCP initiator port is communicating with it; really, it's the bridge acting on behalf of an iSCSI initiator port. The iSCSI initiator port thinks that it is communicating with an iSCSI target port; really, it's the bridge acting on behalf of an FCP target port.

The bridge interprets the frames (transmitting command, task management function, data, or status) on one protocol and recreates them on the other protocol (with modifications based on the mapper, e.g. changing the initiator port identifier, target port identifier, logical unit number, and task tag).

Initiator ports and target ports can be on either side of the bridge. Figure 2 shows an example.

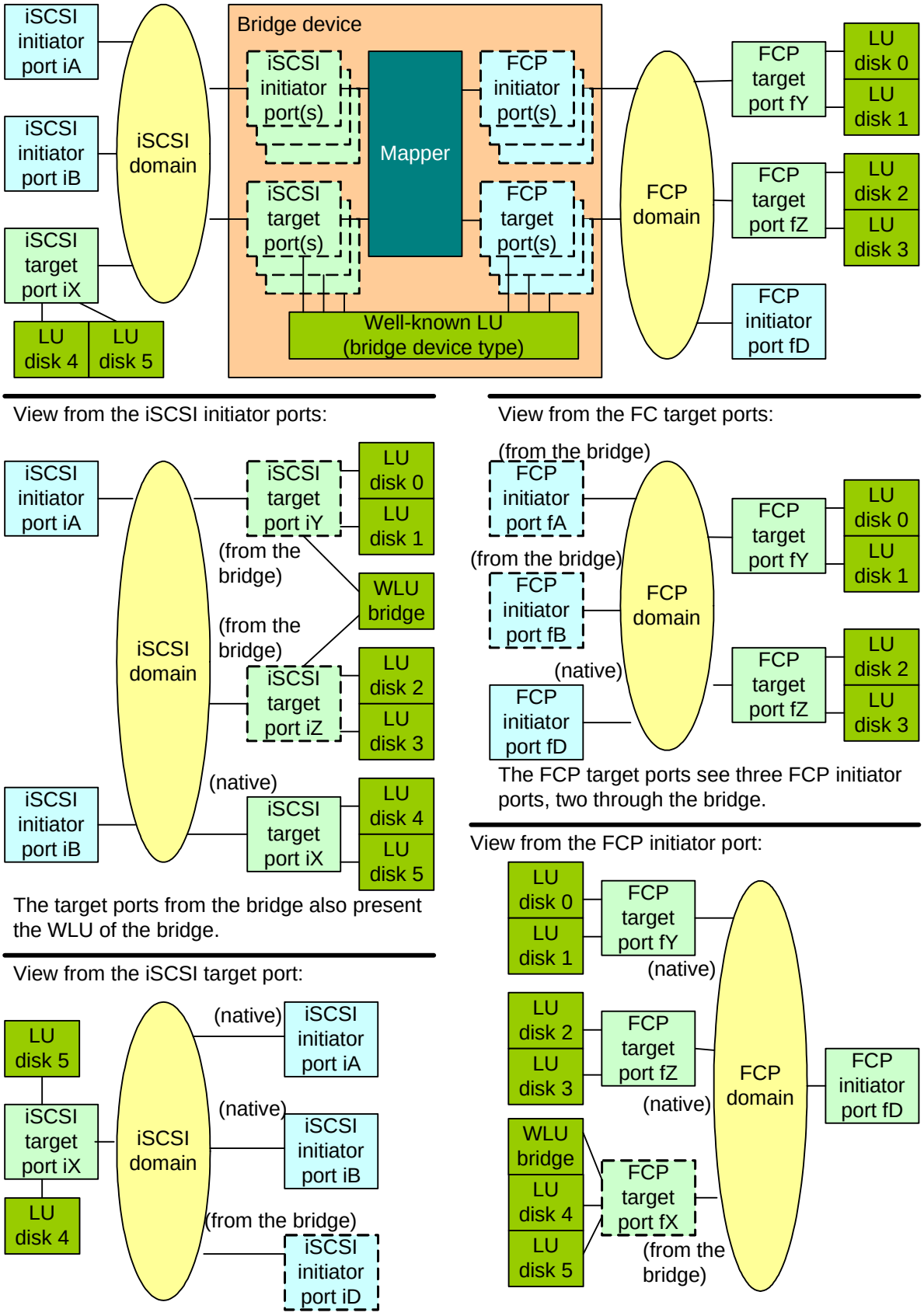



**Figure 2 — iSCSI to FCP bridge with initiator ports and target ports on both sides**

The bridge probably does not employ lots of physical ports to serve these distinct roles; it shares a small number of physical ports (the number depends on bandwidth requirements). The techniques to do this differ for each protocol:

a) In Fibre Channel, the bridge can use virtual N_Ports in Fibre Channel let the bridge appear to be multiple ports at once, or look like a switch with multiple ports behind it;
b) In iSCSI, ports are logical concepts and can share an IP address;
c) In InfiniBand SRP, ports are logical concepts behind a channel adapter;
d) In SAS, the bridge has to appear as an expander with multiple ports behind it (a SAS physical link to an end device is not allowed to support more than one SAS address);
e) in parallel SCSI, the bridge can consume multiple SCSI IDs (although there are a limited number of choices available).

**Mapping logical units**

Bridges vary in complexity and capability. For mapping of logical units behind target ports on the far side of a bridge to the near side (near the initiator port), there are two major approaches:

a) **Target port mapping**. Each target port on the far side of the bridge is mapped to a unique target port on the near side of the bridge. Logical unit numbers accessible via each target port are unchanged, even though the protocol of the target port apparently changes. This is the cleanest model, but may be difficult to implement on some protocols because the bridge has to pretend to have virtual target ports even though it might only have one physical port.
b) **Logical unit mapping**. The bridge presents a single target port on the near side of the bridge. Select logical units from select target ports on the far side of the bridge are mapped to LUNs behind the near side target port. As far as the initiator port is concerned, the bridge serves as the SCSI target device containing those logical units; the fact that there are really in separate SCSI target devices on the far side is hidden.

   Due to the likely LUN collisions (many of the back-side LUNs are probably LUN 0), the bridge must remap all the LUNs. Since combining them into one target device and renumbering them changes the target port/logical unit relationship, it introduces several problems:

   A) REPORT LUNS data from any logical unit is wrong. A logical unit only knows about logical units in the same physical SCSI target device. It doesn't know about other logical units that the bridge has mapped as its peers. The logical unit doesn't even know its own LUN as seen through the bridge. The bridge must intercept REPORT LUNS data and make it reflect the target device LUN inventory that the bridge has created.
   B) TARGET RESET behaves incorrectly. The near side initiator port expects to reset all the LUs visible in the near side target device (the one created by the bridge). If the bridge forwards the TARGET RESET to each far side target port containing a mapped logical unit, it resets all the LUs in one of the far side target devices, some of which might not be mapped through the bridge. It's not friendly to reset logical units the near side initiator knows nothing about. The bridge needs to convert TARGET RESET into a LOGICAL UNIT RESET for each of the LUs that are mapped to restrict their scope. Older target devices might not support LOGICAL UNIT RESET, however.

   NOTE 1 TARGET RESET is obsolete in SAM-3. Nevertheless, implementations will continue for a while.

   C) Commands using relative target port identifiers reference the wrong values. The far side target device's relative target port identifiers are different than the near side target device's relative target port identifiers. In particular, the asymmetric logical unit access commands (REPORT/SET TARGET PORT GROUP) will not work correctly. It would be very difficult for a bridge to intercept these commands and handle them in an intelligent fashion.
   D) Well-known logical unit numbers conflict with each other. If the far side target device has any well-known logical units, it is impossible for the bridge to map them and preserve their well-known LUNs. They can overlap with each other and with the bridge's own well-known logical units (especially with nested bridges).

**Mapping initiator ports**

The bridge should map each initiator port to a unique far side initiator port. For many SCSI commands, the target port needs to realize there are separate initiator ports communicating with it. This is critical for Access Controls and Persistent Reservations, for example.

Some bridges just ignore this issue and present one initiator port on the far side. Commands cognizant of initiator port identity like Access Controls and Persistent Reservations do not operate correctly unless the bridge intercepts them and implements them itself (meaning, at least for those commands, that it becomes a full-fledged SCSI target port and device server and hides the far side like a RAID controller). These bridges should at least be identified so applications can determine when not to send certain commands through them. More advanced bridges of this type that do implement some of the commands themselves should be able to identify which commands they support.

Thus there are three levels:

1) one initiator port;
2) one initiator port with special command support; and
3) many initiator ports.

**Bridge device type**

Current protocol bridges use the SCSI Controller Commands (SCC) device type 0Ch for their vendor-unique configuration/management logical units. So do RAID controllers, the type of devices for which that device type was designed. Neither bridges nor RAID controllers, however, actually comply with the SCC command set and implement the commands defined in SCC (e.g., VOLUME SET IN/OUT, REDUNDANCY GROUP IN/OUT, etc.).

Rather than continue to misuse the SCC device type, a new SCSI device type is proposed for bridge devices ( 13h is currently the next available code). A SCSI target device with bridge functionality shall have one well-known logical unit using the bridge device type.

The bridge well-known logical unit shall be accessible through every target port in the bridge device. For logical unit mappign bridges, there are a fixed set of target ports. For target port mapping bridges, most target ports are virtual target ports created as maps of far-side target ports. The bridge probably also has at least one target port of its own, so communication can occur before any mapping is enabled.

The bridge logical unit shall support these commands:

a) **INQUIRY**: Returns PERIPHERAL DEVICE TYPE field set to Bridge.
b) **REPORT LUNS**: Reports the LUNs accessible via the target port that is being used to access this well-known logical unit. If the target port is one of the bridge's own, this only reports native bridge logical units. If the target port is a mapped target port, it reports logical units owned by that target port.
c) **TEST UNIT READY**: Indicates if the bridge mapping table is available. This command probably always returns GOOD status since there is no media to access. If bridge mapping tables ever get large and are stored on media, however, this could be used to indicate the mapping table is ready. It could also be used to delay access while the table is being configured (by vendor-specific means).
d) **REPORT BRIDGE MAPPING**: Returns a data structure describing all the mapping that a bridge performs (as viewed by the specified initiator port).
e) **WAIT FOR BRIDGE MAPPING CHANGE**: Since the SCSI architecture no longer defines asynchronous event notification, this serves as a camp-on command that completes and returns status whenever a change in mapping occurs. If willing to consume one of its own queue slot resources, software can leave this command outstanding forever and get notified of a change.

The logical unit shall generate a unit attention whenever its mapping changes.

**Configuring bridges**

At some point in the future, a **SET BRIDGE MAPPING TABLE** command could be defined. This is not proposed for MSC-1. This proposal only tries to detect bridges that are present and figure out what they are doing. Controls may be better handled via SMI-S (Storage Management Initiative Specification - see

http://www.snia.org) - and other techniques. SCSI application using a bridge do not necessarily need access to control the bridge - they are already working under zoning and other restrictions outside their control..

**Multiple bridges**

A well known logical unit works well for one bridge. However, if the first W-LU claims ownership of a W-LUN, the W-LUs of a bridge behind the bridge are unreachable.

Figure 3 shows an example of two bridges where accessing both of their W-LUs becomes a problem..

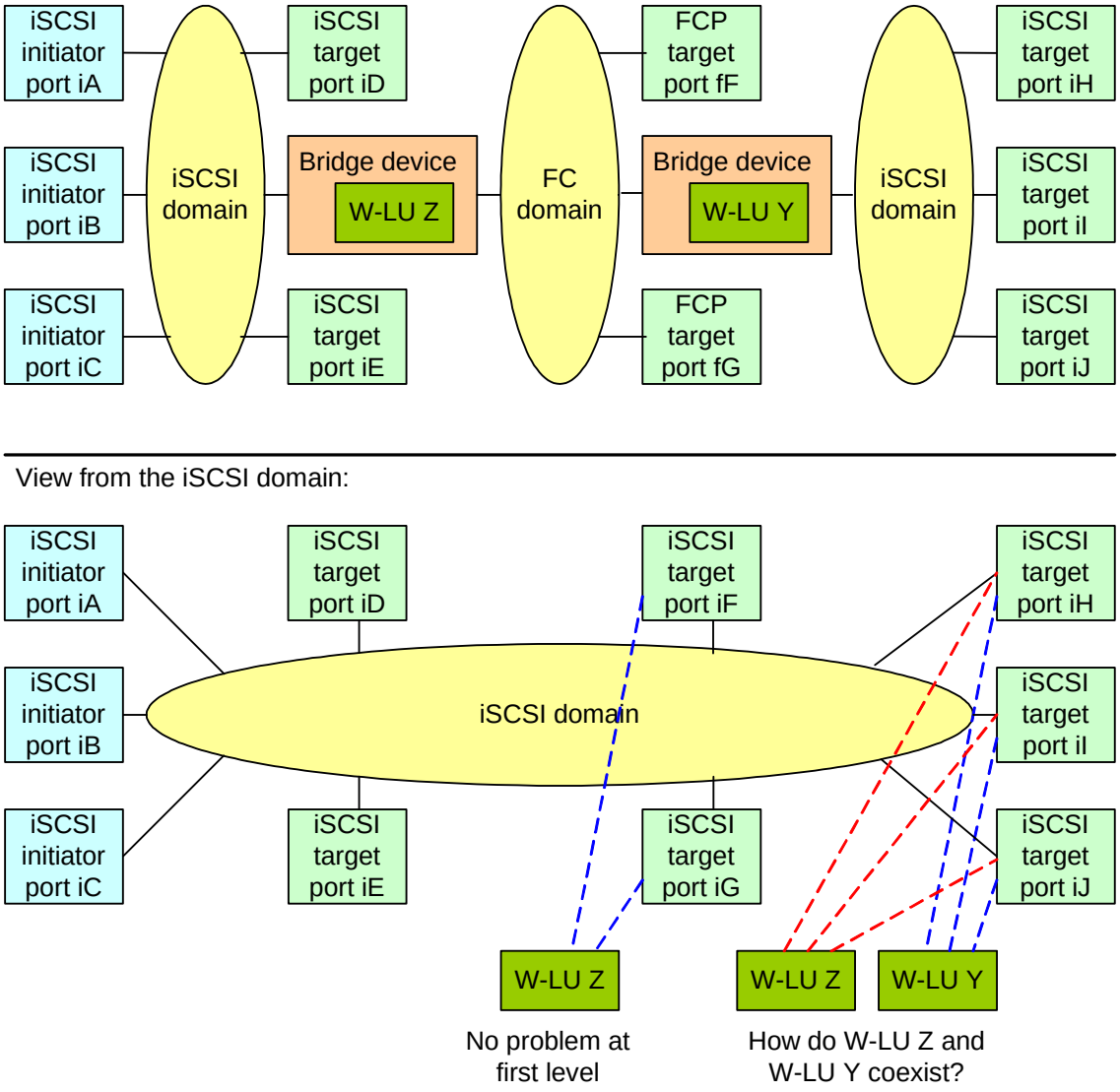



**Figure 3 — Nested bridges**

The hierarchical addressing format from SAM-3 and SCC-2 can be modified to facilitate this (for 4 levels of bridges).

**SAM-3 hierarchical addressing overview**

The 8-byte logical unit number field is divided into four 2 byte structures. Each 2 byte field assumes one of the following formats:

a) peripheral device addressing;
b) flat space addressing;
c) logical unit addressing; or

d) extended logical unit addressing.

Table 1 shows the peripheral device addressing method defined in SAM-3. This is the most widely used method. It provides access for 256 LUNs in byte 1; the BUS IDENTIFIER field is typically set to 00000b.

**Table 1 — Peripheral device addressing**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 00b | | BUS IDENTIFIER | | | | | |
| 1 | TARGET/LUN | | | | | | | |

Table 2 shows the flat space addressing method defined in SAM-3. This is the next most widely used method. It provides access for 16K LUNs

**Table 2 — Flat space addressing**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 01b | | | | | | | |
| 1 | LUN | | | | | | | |

Table 3 shows the logical unit addressing method defined in SAM-3. It does not seem to be used much, with small BUS NUMBER, TARGET, and LUN fields. SCC-2 envisioned a pre-configured parallel SCSI environment behind a RAID controller; this could be used to address the physical disk drives.

**Table 3 — Logical unit addressing**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 10b | | TARGET | | | | | |
| 1 | BUS NUMBER | | | LUN | | | | |

Table 4 shows the extended logical unit addressing method defined in SAM-3. This is the method used to address well-known logical units.

**Table 4 — Extended logical unit addressing**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 11b | | LENGTH | | EXTENDED ADDRESS METHOD | | | |
| 1 | EXTENDED ADDRESS METHOD SPECIFIC | | | | | | | |

To address a well-known LUN behind a bridge (not the bridge's own W-LUN, the W-LUN of some other bridge or some other target device), the logical unit addressing format 10b is replaced with a new format that specifies the outgoing relative initiator port identifier instead of unusably small TARGET, BUS NUMBER, and LUN fields. Table 5 shows the proposed type 10b for bridges.

**Table 5 — Bridge addressing method**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 10b | | (MSB) | | | | | |
| 1 | RELATIVE INITIATOR PORT | | | | | | | (LSB) |

This does not conflict with the SCC-2 use of type 10b. If a bridge receives this LUN, it interprets it per table 5. If an SCC-2 target received this LUN, it would interpret it per table 3.

Each bridge interprets the first level (first 2 bytes) of the LUN to decide where to send the frame. When it outputs the frame, it shifts off the first level (first 2 bytes) of the LUN and fills the last two bytes with zeros.

With this, to access W-LU Z in the figure 3, iSCSI initiator port iA, iB, or iC sends a command to the LUN shown in table 6 with target port identifier iF, iG, iH, iI, or iJ. It cannot use iD or iE, since they are not behind the first bridge. This is the first level LUN; nothing special.

**Table 6 — Accessing W-LU Z from iA, iB, or iC - initial (and only) LUN**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 11b | | LENGTH | | EXTENDED ADDRESS METHOD (1h) | | | |
| 1 | EXTENDED ADDRESS METHOD SPECIFIC (W-LUN Z) | | | | | | | (LSB) |
| 2 | Unused (000000h) | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |

With this, to access W-LU Y in the second bridge, iSCSI initiator port iA, iB, or iC sends a command to the LUN shown in table 7 with target port identifier iH, iI, or iJ. It cannot use iD, iE, iF, or iG, since they are not behind the second bridge. This is a two-level LUN. It instructs the first bridge which output port to use to send the rest of the LUN (shifted into a first-level LUN).

**Table 7 — Accessing W-LU Y from iA, iB, or iC - initial LUN**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 10b | | (MSB) | | | | | |
| 1 | RELATIVE INITIATOR PORT (in first bridge) | | | | | | | (LSB) |
| 2 | 11b | | LENGTH | | EXTENDED ADDRESS METHOD (1h) | | | |
| 3 | EXTENDED ADDRESS METHOD SPECIFIC (W-LUN Y) | | | | | | | (LSB) |
| 4 | Unused (0000h) | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |

As the first bridge forwards this LUN, it shifts off the first two bytes and sets the last two bytes to zero, sending the command to the LUN shown in table 8 with the target port identifier changed to the fH, fI, or fJ based on the first bridge's mapping efforts..

**Table 8 — Accessing W-LU Y from iA, iB, or iC- LUN as output by first bridge**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 11b | | LENGTH | | EXTENDED ADDRESS METHOD (1h) | | | |
| 1 | EXTENDED ADDRESS METHOD SPECIFIC (W-LUN Y) | | | | | | | (LSB) |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |

Unused (000000h)

If the first bridge does not have a map yet for fH, fI, or fJ, this communication cannot happen. Most bridges will have a target port of their own available for management; if that target port is mapped, communication can occur.

Figure 4 shows the target port identifiers and LUN values described in table 6, table 7, and table 8 as they are used to access the bridges in figure 3.

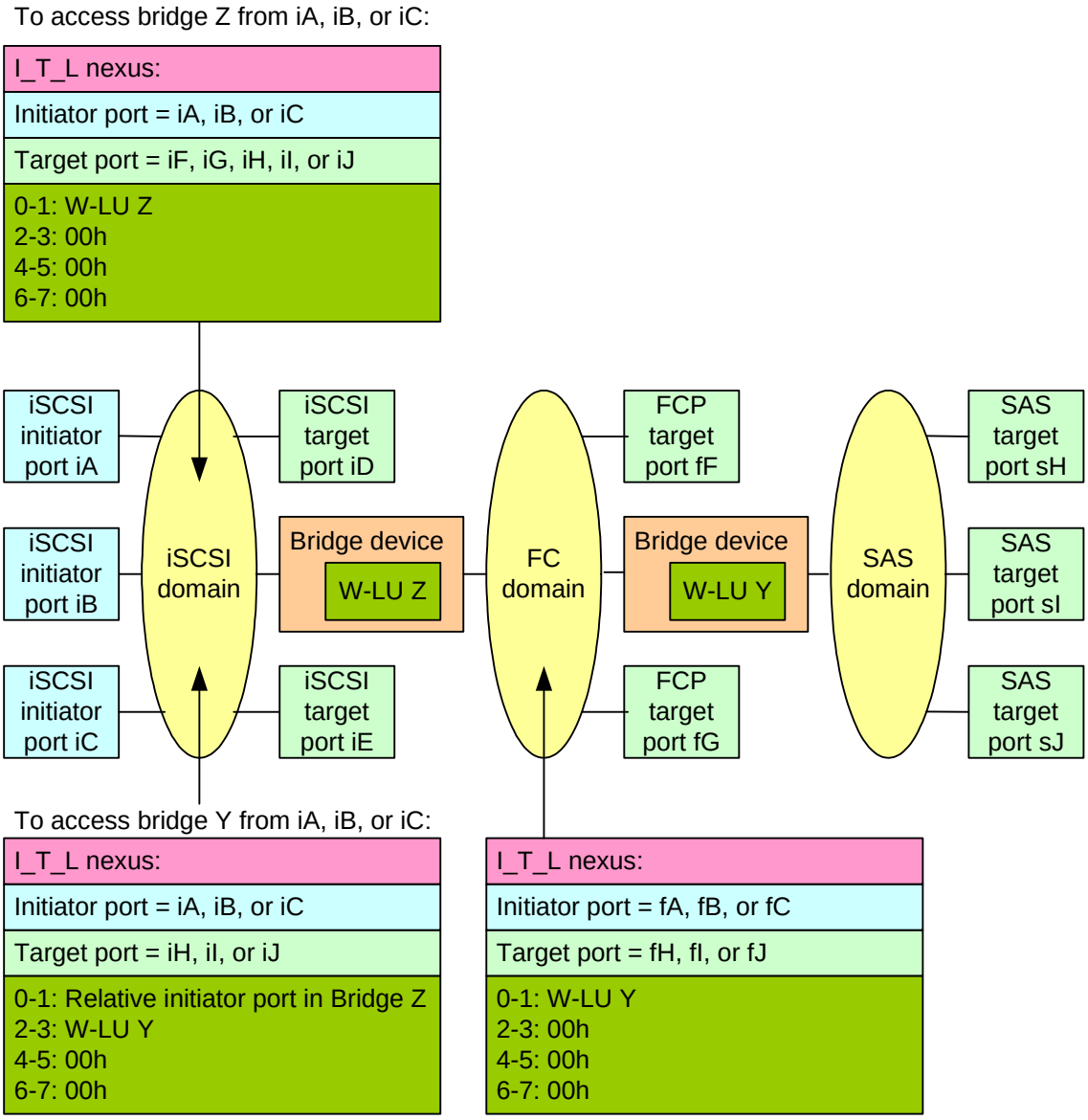



**Figure 4 — Accessing W-LU Z and W-LU Y from iA, iB, or iC**

If a third bridge were present with W-LU W in the SAS domain, an iSCSI initiator port would send a command to the LUN shown in table 9 through some target ports behind that third bridge (or owned by that third bridge itself). Traffic to target ports iD, iE, iF, iG, iH, iI, and iJ does not go through the third bridge, so the second bridge would not be able to parse the LUN field.

**Table 9 — Accessing W-LU W from iA, iB, or iC - initial LUN**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 10b | | (MSB) | | | | | |
| 1 | RELATIVE INITIATOR PORT (in first bridge) | | | | | | | (LSB) |
| 2 | 10b | | (MSB) | | | | | |
| 3 | RELATIVE INITIATOR PORT (in second bridge) | | | | | | | (LSB) |
| 4 | 11b | | LENGTH | | EXTENDED ADDRESS METHOD (1h) | | | |
| 5 | EXTENDED ADDRESS METHOD SPECIFIC (W-LUN W) | | | | | | | (LSB) |
| 6 | Unused (00h) | | | | | | | |
| 7 | | | | | | | | |

Overall, the LUN format supports four levels of bridges as shown in table 10. Larger topologies are not accessible through this limited 8-byte LUN field and will remain beyond the scope of this standard.

**Table 10 — Maximum bridge addressing LUN**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 10b | | (MSB) | | | | | |
| 1 | RELATIVE INITIATOR PORT (in first bridge) | | | | | | | (LSB) |
| 2 | 10b | | (MSB) | | | | | |
| 3 | RELATIVE INITIATOR PORT (in second bridge) | | | | | | | (LSB) |
| 4 | 10b | | (MSB) | | | | | |
| 5 | RELATIVE INITIATOR PORT (in third bridge) | | | | | | | (LSB) |
| 6 | 11b | | LENGTH | | EXTENDED ADDRESS METHOD (1h) | | | |
| 7 | EXTENDED ADDRESS METHOD SPECIFIC (W-LUN of fourth level bridge) | | | | | | | (LSB) |

Figure 5 shows an example with nesting and peer bridges.

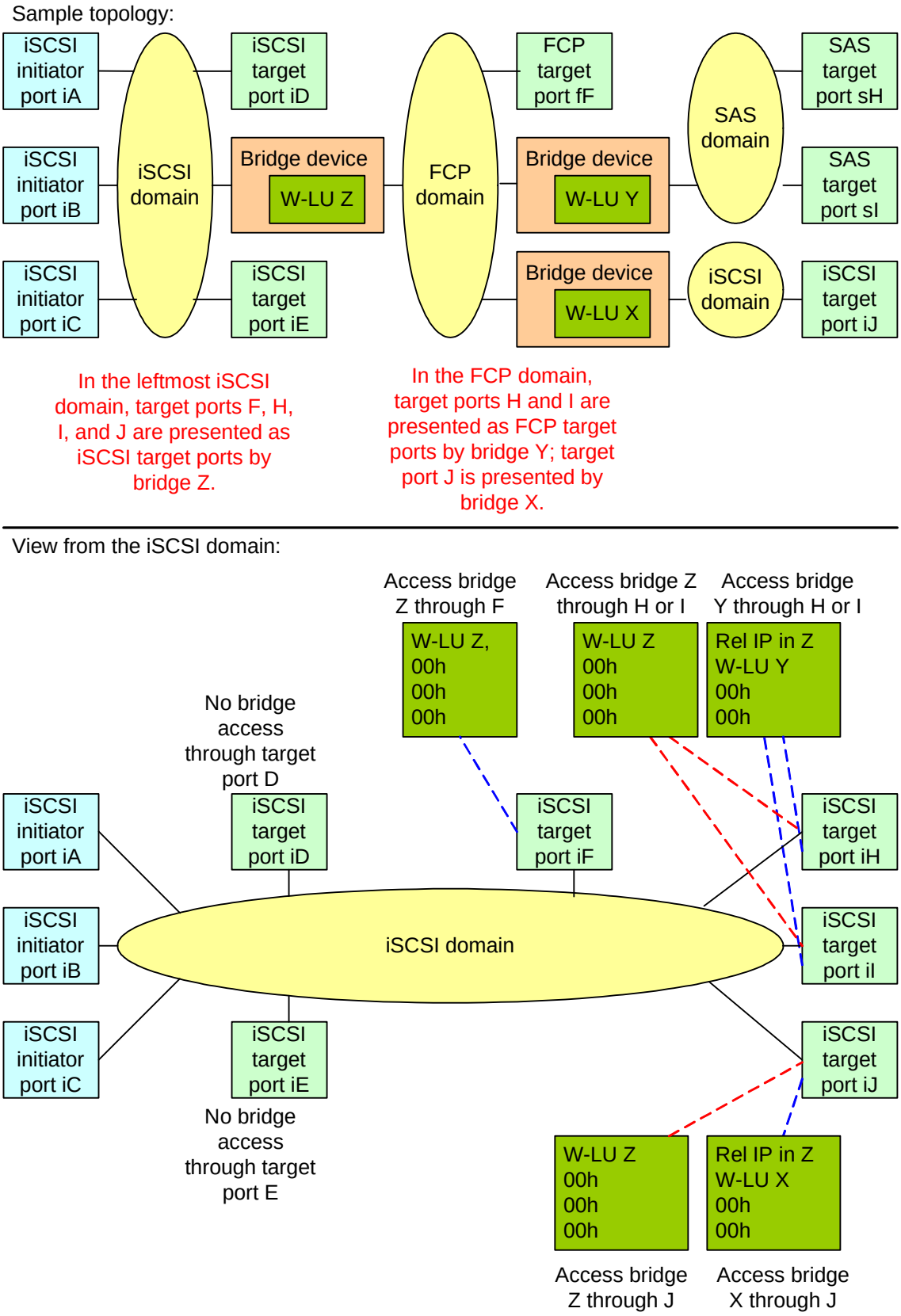



**Figure 5 — Nesting and peer bridges**

**Different initiator port views**

Bridges might implement features like zoning, LUN mapping, LUN masking that present different views of the the domain to different initiator ports. To accommodate this, the READ BRIDGE MAPPING command is proposed as a bidirectional command, with an optional parameter list (write data) specifying which initiator port's view to return.

If the bridge does not employ such features, it returns the same view for any initiator port.

One example is when a backup application needs to obtain the mapping for a copy manager on the far side of a bridge. The application can start by requesting its own initiator port mapping information, learning the true identity of the target ports it is accessing. It then queries the bridges for the mappings for the initiator port(s) used by the copy manager. Combining the two, it can determine which target port names/identifiers to use in copy manager target descriptors to describe the correct source and destination.

Figure 6 shows an example of a backup application and a copy manager.

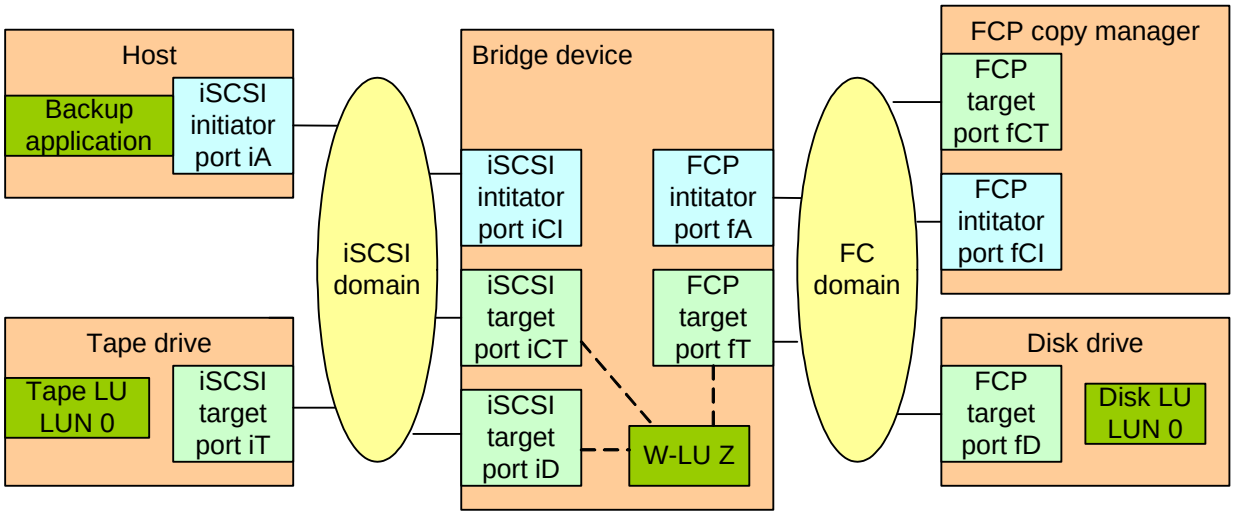



**Figure 6 — Copy manager example**

Table 11 shows the addresses used in that figure 6.

**Table 11 — Copy manager addressing**

| Object | Native address | iSCSI initiator port view | FCP copy manager initiator port view |
|---|---|---|---|
| iSCSI initiator port | (iSCSI name) iA | iA (its own name) | fA (through bridge) |
| FCP copy manager target port | (FC WWPN) fCT | iCT (through bridge) | fCT |
| FCP copy manager initiator port | (FC WWPN) fCI | N/A | fCI (its own name) |
| FC disk drive | (FC WWPN) fD | iD (through bridge) | fD |
| iSCSI tape drive | (iSCSI name) iT | iT | fT (through bridge) |

Table 12 shows the mapping table entries retrieved for iA through iCT or iD.

**Table 12 — Mapping table for iA**

| Object | Near side address | Far side address |
|---|---|---|
| FCP copy manager target port | iCT | fCT |
| FCP disk drive | iD | fD |

The backup application discovers that fCT is in a target/initiator device by retrieving the SCSI Ports VPD page (see 03-344) and noticing its initiator ports. It finds the TransportID for fCI in that page and sends a REPORT MAPPING TABLE command specifying fCI as the initiator port whose mapping is requested.

Table 13 shows the mapping table entries retrieved for fCI through iCT or iD.

**Table 13 — Mapping table for fCT**

| Object | Near side address | Far side address |
|---|---|---|
| iSCSI tape drive | fT | iT |

When it builds a target descriptor for fCI to copy from disk to tape, known to itself as iD and iT, it uses the identifiers that fCI uses for those devices as shown in table 14.

**Table 14 — Target descriptors**

| Object | Value |
|---|---|
| Source (FCP tape drive) target port identifier | fD |
| Destination (iSCSI tape drive) target port identifier | fT |
| LUN | unchanged |

# 1 Interceptable commands

## 1.1 Interceptable commands overview

This clause lists commands that present problems for bridges. When the bridge and application software both support MSC, many of these issues can be avoided; the bridge can just pass through the data to/from the logical unit unchanged and the application can figure out how to interpret it correctly (for read data) and set it correctly (for write data). If the application does not fully support MSC (especially the mapping table queries), the bridge may intercept select commands and make them behave properly.

## 1.2 INQUIRY command

The Device Identification VPD page (83h) identification descriptors with an ASSOCIATION field set to 1h (i.e., target port) are based on the far side target port. A transparent bridge has to replace these descriptors with ones representing the near side target port.

Identification descriptors with an IDENTIFIER TYPE field set to 4h (relative target port) are based on the far side target port. A transparent bridge preserves this mapping. A LUN mapping bridge changes this mapping; new relative target port numbers might need to be assigned. This affects commands that use the relative target port identifier (see REPORT/SET TARGET PORT GROUPS and PERSISTENT RESERVE IN/OUT).

Identification descriptors with an ASSOCIATION field set to 2h (i.e., target device) are based on the far side target device. A transparent bridge has to replace these descriptors with ones representing the bridge device itself.

## 1.3 Alias lists (CHANGE ALIASES and REPORT ALIASES commands)

Alias entry designators are all protocol specific and need to be translated by a transparent bridge.

## 1.4 Extended copy (EXTENDED COPY and REPORT COPY RESULTS commands)

Most target descriptors are protocol specific and need to be translated by a transparent bridge.

## 1.5 Persistent reservations (PERSISTENT RESERVE IN and PERSISTENT RESERVE OUT commands)

The TransportIDs used by the Specify Initiator Ports feature are protocol specific and need to be translated by a transparent bridge.

Relative target port identifiers returned by the PERSISTENT RESERVE IN command READ FULL STATUS service action (proposed by 03-342) need to be translated by a transparent bridge.

## 1.6 Access controls (ACCESS CONTROLS IN and ACCESS CONTROLS OUT commands)

TransportIDs are protocol specific and need to be translated by a transparent bridge.

AccessIDs, on the other hand, should flow through without problems.

## 1.7 Asymmetric logical unit access (REPORT TARGET PORT GROUPS and SET TARGET PORT GROUPS commands)

Relative target port identifiers returned by the PERSISTENT RESERVE IN command READ FULL STATUS service action (proposed by 03-342) need to be translated by a transparent bridge.

## 1.8 Log pages (LOG SELECT and LOG SENSE commands)

The Protocol-Specific log page need to be handled by the transparent bridge (Note: this page includes relative target port identifiers). It is not a good idea to pass through accesses to this page, since the near and far side meanings could be completely different.

## 1.9 Mode pages (MODE SELECT and MODE SENSE commands)

The Protocol-Specific Port mode page and Logical Unit mode page need to be handled by a transparent bridge.

It is not a good idea to pass through accesses to these pages, since the near and far side meanings could be completely different.

# 2 Commands

## 2.1 REPORT BRIDGE MAPPING open issues

a) Identify any limitations on # initiators that can be mapped (e.g. a bridge to parallel SCSI might be limited to 2 initiators). If the limit is 1, then initiator-cognizant commands shall not be used through the bridge.

## 2.2 REPORT BRIDGE MAPPING command

The REPORT BRIDGE MAPPING command (see table 15) requests information on task management functions (see SAM-3) the addressed logical unit supports. The REPORT BRIDGE MAPPING command is optionally bidirectional:

a) the optional write parameter list lets the application client specify the initiator port and bridge target port combination for which the bridge shall return mapping table information; and
b) the read parameter data contains the requested mapping table information.

The REPORT BRIDGE MAPPING command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit equal to one in their standard INQUIRY data (see SPC-3).

**Table 15 — REPORT BRIDGE MAPPING command**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (A3h) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION (nnh) | | | | |
| 2 | (MSB) | PARAMETER LIST LENGTH | | | | | | |
| 5 | | | | | | | | (LSB) |
| 6 | (MSB) | ALLOCATION LENGTH | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | Reserved | | | | | | | |
| 11 | CONTROL | | | | | | | |

The OPERATION CODE field is set to A3h.

The SERVICE ACTION field is set to nnh.

The PARAMETER LIST LENGTH field specifies the number of bytes of the write parameter list available in the data-out buffer, if any. If no parameter data is provided, the bridge shall return mapping information for the initiator port running the REPORT BRIDGE MAPPING command.

The ALLOCATION LENGTH field specifies the number of bytes that have been allocated for the read parameter data in the data-in buffer. The allocation length shall be at least four. If the allocation length is less than for the command shall be terminated with a CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

The CONTROL field is defined in SAM-3.

The format of the parameter list optionally provided in the data-out buffer is shown in table 17.

**Table 16 — REPORT BRIDGE MAPPING parameter list (data-out)**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | RELATIVE TARGET PORT | | | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | (MSB) | INITIATOR PORT TRANSPORTID LENGTH (n - 3) | | | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | | INITIATOR PORT TRANSPORTID | | | | | | |
| n | | | | | | | | |

The RELATIVE TARGET PORT field specifies the target port through which the specified initiator port applies. A RELATIVE TARGET PORT field set to zero specifes that the target port through which the REPORT BRIDGE MAPPING command was received shall be used.

The INITIATOR PORT TRANSPORTID LENGTH field specifies the length of the INITIATOR PORT TRANSPORTID LENGTH field.

The INITIATOR PORT TRANSPORTID field specifies the TransportID of the initiator port for which the bridge shall return mapping information.

The format of the parameter data returned in the data-in buffer is shown in table 17.

**Table 17 — REPORT BRIDGE MAPPING parameter data (data-in)**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | EXTENDED COPY | ACCESS CONTROL | PERSISTENT RESERVE | TARGET PORT GROUPS | ALIAS | MODE | LOG | INQUIRY |
| 1 | Reserved | | | | | | | |
| 3 | | | | | | | | |
| 4 | (MSB) | MAPPING TABLES LENGTH (m - 7) | | | | | | |
| 7 | | | | | | | | (LSB) |
| | Mapping table entries | | | | | | | |
| 8 | Mapping table entry (first) | | | | | | | |
| | | | | | | | | |
| | ... | | | | | | | |
| | Mapping table entry (first) | | | | | | | |
| m | | | | | | | | |

An EXTENDED COPY bit set to one means the bridge intercepts the EXTENDED COPY command (see 1.4). An EXTENDED COPY bit set to one means the bridge does not intercept the EXTENDED COPY command.

An ACCESS CONTROL bit set to one means the bridge intercepts the ACCESS CONTROL IN and ACCESS CONTROL OUT commands (see 1.6). An ACCESS CONTROL bit set to one means the bridge does not intercept the ACCESS CONTROL IN and ACCESS CONTROL OUT commands.

A PERSISTENT RESERVE bit set to one means the bridge intercepts the PERSISTENT RESERVE IN and PERSISTENT RESERVE OUT commands (see 1.5). A PERSISTENT RESERVE bit set to one means the bridge does not intercept the PERSISTENT RESERVE IN and PERSISTENT RESERVE OUT commands.

A TARGET PORT GROUPS bit set to one means the bridge intercepts the REPORT TARGET PORT GROUPS and SET TARGET PORT GROUPS commands (see 1.7). A TARGET PORT GROUPS bit set to one means the bridge does not intercept the REPORT TARGET PORT GROUPS and SET TARGET PORT GROUPS commands.

An ALIAS bit set to one means the bridge intercepts the CHANGE ALIASES and REPORT ALIASES commands (see 1.3). An ALIAS bit set to one means the bridge does not intercept the CHANGE ALIASES and REPORT ALIASES commands.

A MODE bit set to one means the bridge intercepts the MODE SENSE and MODE SELECT commands (see 1.9). A MODE bit set to one means the bridge does not intercept the MODE SENSE and MODE SELECT commands.

A LOG bit set to one means the bridge intercepts the LOG SENSE and LOG SELECT commands (see 1.8). A LOG bit set to one means the bridge does not intercept the LOG SENSE and LOG SELECT commands.

An INQUIRY bit set to one means the bridge intercepts the INQUIRY command (see 1.2). An INQUIRY bit set to one means the bridge does not intercept the INQUIRY command.

The mapping table indicates:

a) how the bridge is mapping far side target ports and logical units into near side target ports and logical units as seen by the initiator port specified in the write parameter list; and
b) how the bridge is mapping the specified initiator port into a far side initiator port as seen by each of those far side target ports and logical units.

The MAPPING TABLES LENGTH field indicates the length of all the mapping table entries that follow. The bridge shall return all mapping table entries applicable to the specified initiator port. If the allocation length is not large enough to return all the mapping table entries, this field shall not be changed.

The mapping table entry format is defined in table 18.

**Table 18 — Mapping table entry**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | ENTRY LENGTH (n - 1) | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | (MSB) | | | NEAR SIDE RELATIVE TARGET PORT | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | | | | NEAR SIDE LOGICAL UNIT NUMBER | | | | |
| 11 | | | | | | | | |
| 12 | (MSB) | | | FAR SIDE RELATIVE INITIATOR PORT | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | (MSB) | | | FAR SIDE TARGET DESCRIPTOR LENGTH (n - 15) | | | | |
| 15 | | | | | | | | (LSB) |
| 16 | | | | FAR SIDE TARGET DESCRIPTOR | | | | |
| n | | | | | | | | |

The ENTRY LENGTH field indicates the number of bytes that that follow in the mapping table entry.

The NEAR SIDE RELATIVE TARGET PORT field contains the relative target port identifier of the bridge device capable of mapping to one or more logical units.

The NEAR SIDE LOGICAL UNIT NUMBER field shall be set to FFFFFFFF FFFFFFFFh if all logical units behind the target port are being mapped. Otherwise, it contains the logical unit number accessible through the target port identified by the NEAR SIDE RELATIVE TARGET PORT field/

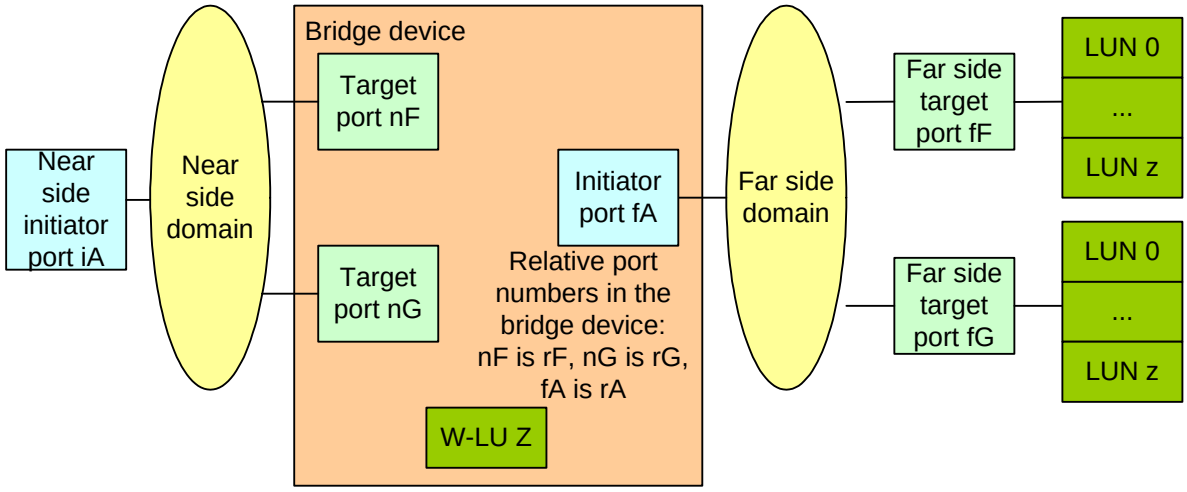
The BACK SIDE INITIATOR TRANSPORTID LENGTH field indicates the number of bytes in the BACK SIDE INITIATOR TRANSPORTID field.

The FAR SIDE RELATIVE INITIATOR PORT field contains the relative port identifier of the bridge device's initiator port used to access the specified far side target port and logical unit(s).

The FAR SIDE TARGET DESCRIPTOR LENGTH field indicates the number of bytes in the BACK SIDE TARGET DESCRIPTOR field.

The FAR SIDE TARGET DESCRIPTOR field contains an EXTENDED COPY target descriptor (see SPC-3) identifying a target port and one or more logical units that a near side initiator port accesses through the target port indicated by the NEAR SIDE RELATIVE TARGET PORT field. The bridge forwards these accesses with the initiator port indicated by the FAR SIDE RELATIVE INITIATOR PORT field. The LU IDENTIFIER field of the far side target descriptor shall be set to FFFFFFFF FFFFFFFFh if all logical units behind the target port are being mapped.

Figure 7 shows an example of a mapping table.




When iA queries W-LU Z through iF, the mapping table returns:
a) near side relative target port rF;
b) near side LUN=FFFFFFFF FFFFFFFFh;
c) far side relative initiator port rA; and
d) far side target descriptor describing target port fF with LUN=FFFFFFFF FFFFFFFFh

When iA queries W-LU Z through iG, the mapping table returns:
a) near side relative target port rG;
b) near side LUN=FFFFFFFF FFFFFFFFh;
c) far side relative initiator port rA; and
d) far side target descriptor describing target port fF with LUN=FFFFFFFF FFFFFFFFh

**Figure 7 — Mapping table example**

## 2.3 WAIT FOR BRIDGE MAPPING CHANGE command

The WAIT FOR BRIDGE MAPPING command (see table 19) waits for a bridge mapping to change before completing. After this command completes, an application client should send a REPORT BRIDGE MAPPING command to determine what changed.

The WAIT FOR BRIDGE MAPPING CHANGE command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit equal to one in their standard INQUIRY data (see SPC-3).

**Table 19 — WAIT FOR BRIDGE MAPPING TABLE command**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (A3h) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION (nnh) | | | | |
| 2 | Reserved | | | | | | | |
| 9 | | | | | | | | |
| 10 | Reserved | | | | | | | |
| 11 | CONTROL | | | | | | | |

The OPERATION CODE field is set to A3h.

The SERVICE ACTION field is set to nnh.

The CONTROL field is defined in SAM-3.