SAN Management & Mode Pages

Roger Cummings

January 17, 2001
Contents

- Introduction & Background
- Problem #1 Definition
- Need & Proposal
- Problem #2 Definition
- Need & Proposal
- Putting all this together - Phases
- Summary
Introduction

• Two related problems to present in the area of SCSI Control & Status
  – Solving will require work in both T10 & T11
• Problem definition based on experience with developing management applications for a wide range of current SANs
  – And extrapolating trends with respect to the HUGE SANs of the future
• Aim is to define one approach now that will work for all sizes of SANs, regardless of the transport type (SCSI, FC, TCP/IP etc.)
Background

• All the high-end storage devices today have an Ethernet port in addition to SCSI/FC ports
• And the devices are reachable over the LAN from the servers connected to the SAN
What this means?

- Management apps can obtain information from the SAN via 4 methods:
  - SNMP (over Ethernet)
  - FC Fabric Services
  - SCSI Mode and Log Pages
  - SCSI Inquiry Data
- Information is not always consistent
- Trend is clearly towards more and more information being available via “out-of-SAN” access:
  - More secure (if only by obscurity)
  - Often “almost but not quite” same information as in Inquiry and Mode Pages
And further…..

- iSCSI has additional method to get information - Text Command and Text Response
  - Multiple key:value pairs separated by ASCII NUL
  - Used for addressing, URLs, enabling Request To Transfer etc.
    - Equivalent to Process Login
  - Target processes each key separately
    - If not recognized, not echoed in Text Response
- Easily add vendor-unique keys:
  - Prepend with reversed domain name (e.g. com.veritas.enablemode1:Yes)
Why was this done?

• Religious issue – REAL Networks don’t do management and control with binary data
• There is actually some sense behind the religion:
  – Binary data is:
    • Not self-describing (its just a bunch of bits)
    • Not human-readable
    • Not much use for offline analysis and change detection
• Lets be honest, network guys have more experience in managing LARGE numbers of interconnected equipments than we do:
  – We can learn from their approaches
And…. 

• Existing SCSI Mode Page scheme is not very user-friendly
  – Each page has a different layout
  – Pages can be truncated – need to get exact transfer length (and networks don’t always provide this)
  – No way to mask information
  – Have to do a read before write
  – “Return All Pages” is of almost no use

• Vendor unique extensions to SCSI are difficult & require completely new pages
But...

- SNMP et al is good at reading data, but less good at real time control
- CIM is gaining in popularity, but a complete storage subsystem model in CIM is a daunting task:
  - It will happen, just not anytime soon
- Meanwhile methods of status retrieval and control proliferate…
  - Added piecemeal with new transport support (as in iSCSI)
Need #1

• Need one storage status & control scheme that works over ALL existing transports and new ones:
• Need something more easily extended than current SCSI page-based scheme
• Need single field “namespace” (if only to prevent confusion in multiple transport case)
• Eventually need to secure access to mode pages and some other SCSI functions:
  – Authentication of access
  – Protect against changed configuration by application that should only be accessing data
  – Transaction basis (send, check, execute)
Proposal #1

• T10 defines a standard translation of SCSI Mode Pages and Inquiry Data to XML

• Why XML?
  – Self-describing, human-readable plain-text
  – Transport-neutral (works over LANs, also FC)
  – Can represent complex hierarchies
  – Each field can be read and set separately
  – Supports variable field length
  – Better field contents typing
  – Can indicate allowable ranges
  – Can use vendor-unique naming like iSCSI
  – Its key to the future of the Internet (i.e. there will be LOTS of tools)
Why XML (Contd)?

- Standards exist for:
  - Signing and encrypting XML docs
  - Displaying XML in browsers (style sheets)
  - Searching and transforming etc.
- “Get All” will finally be useful
  - All information can be retrieved in one operation and processed offline
  - Standard format for analyzer output?
- Format can infinitely extendable without page and bit constraints
  - But it’s not about minimizing interface bandwidth!
  - See the tutorial
Tutorial

- XML is like HTML, but with user-defined tag values
  - It’s also simplified SGML
  - Much less tolerance for sloppy formatting than HTML
- Tags don’t define display properties:
  - That’s what style sheets (XSL or CSS) are for!
- Definition of tags used in an associated Document Type Definition (DTD) file
  - Allows XML document to be “parsed” for correct structures
  - Extension being proposed for values as well
Simplified Example

```xml
<?xml version="1.0" ?>
<SCSIMLTransaction id="14567">
  <TransactionType>Response</TransactionType>
  <SCSIClass>Mode_Page</SCSIClass>
  <Epoch>987</Epoch>
  <Page>
    <PageName>Disconnect-reconnect</PageName>
    <Field>
      <Name>Data_Transfer_Disconnect_Control</Name>
      <SizeBits>3</SizeBits>
      <Value Type="bin">011</Value>
      <Attr>PPI,1,NVPC</Attr>
      <SPCPageCode>02</SPCPageCode>
      <SPCName>DTDC</SPCName>
      <SPCStart>12/2</SPCStart>
      <SPCEnd>12/0</SPCEnd>
    </Field>
  </Page>
</SCSIMLTransaction>
```
Proposal

• Start with an XML representation of Inquiry & Mode Page information in SPC-2

• Possible to add new optional types of information for each field, e.g.:
  – Per Port or Per Initiator all Ports or Per Port per Initiator
  – Per LUN or 1 value all LUNs
  – Volatile or Non-Volatile across Resets or Non-Volatile across power cycles
Proposal

• Add an Epoch ID from Target
  – Number incremented by 1 each time a configuration change is made in a device
  – Quick check that SOMETHING has changed without having to check every bit

• Need a Transaction ID from Initiator
  – Tag to link response to a specific management request
  – Will eventually allow a full set of changes to be received, checked, “complied” and then activated
Problem #2

• As SANs grow:
  – The number of Initiators seen by each device grows
  – Management becomes more specialized, and more separate from normal access

• Storage status and control as in traditional SCSI is based on two assumptions, each becoming less true:
  – The Initiator is a portal to BOTH the data access application and the management application
  – All storage status changes are associated with access

• What if management is separate?
• Do you really need to tell the next accessing application that a fan just died in the RAID cabinet?
Need #2

• A method of storage status & control that is separate from data access:
  – Doesn’t have to be on the same system as the application accessing data
    • Doesn’t have to use the same driver stack:
    • Doesn’t have to deal in the same levels of abstraction
  – Doesn’t have to have read or write access to the storage
  – Perhaps doesn’t even have to be able to access the storage device directly
Proposal #2

- Create a “Device Service” in the SAN
  - Accessed via a well-known address (like a Name Server) or equivalent methods over TCP/IP
  - Repository of SCSI Mode Sense and Inquiry Data for all attached storage devices
  - Note - many fabrics already poll for storage devices today to add their addresses to Name Server

- Management apps talk to the service, not the devices
  - Walled off from normal access paths
  - May even be able to provide information across Zones with this approach
Proposal

• The Device Service provides and accepts data in the XML format defined earlier
• Can retrieve all parameters for one device (or multiple devices) in a single transaction
• Supports a “subscribe” model to receive information about changes (similar to state change notification in FC today)
  – Support a LARGE number of devices without impacting the storage devices themselves
Putting all of this together

• Multiple year project with at least four phases
• T11 & T10 parts, also get SNIA & IETF IP Storage Working Group involved
• Important to get the overall XML structure defined quickly
• Migration strategy incorporated
• Will this ever get incorporated in a single SCSI disk?
  – Maybe, but that’s not a key reason for doing this
## Anticipated Phases

<table>
<thead>
<tr>
<th>Phase</th>
<th>Features</th>
</tr>
</thead>
</table>
| 1     | a) Agree XML Format  
b) Create standard DTD, XSL |
| 2     | a) Define FC Device Service (read)  
b) Define TCP/IP access (read) |
| 3     | a) Add set access definition  
b) Add security (authentication) |
| 4     | a) Add new fields (not in SPC-2)  
b) Define In/Out Service Actions for XML command |
Phase 1

- Agree the basic XML structure and the field names (in T10 TR?)
- Define the DTD and XSL
- Work with W3C to get a namespace established for storage devices
  - Allow multiple organizations to define fields and preserve unique names
- Ensure enough flexibility for future developments
Phase 2

• In parallel with Phase 1 work to get a “Device Service” definition which uses the XML representations (for read access only)
  – Defined in FC-GS-4
  – Defined for access across TCP/IP (LDAP, SOAP, Browser access etc.)

• Work with IETF IP Storage WG on establishing methods of storage management access over TCP/IP
Phase 3

- Extend the XML definition to support set capability
  - Possibly also add non-mode functionality that needs restricted access (e.g. format, prevent/allow medium removal)

- Incorporate security features:
  - Authentication using Public Key Infrastructure
  - Encryption
  - Can be made optional or mandatory
Phase 4

- Define new status and control parameters
  - Cannot be accessed via Inquiry, Mode Pages etc.
- Define new SCSI command to transport XML directly to & from the device
  - Could even just be service actions of an existing command
One SCSI Command

• One command, 2 Service Actions only required:
  – XML In
  – XML Out

• Allocation Length required only – everything else should be in XML

• Infinitely extendable

• Simple to add new fields (as long as the names do not conflict)

• Richer set of value types

• Clean way of supporting vendor unique information
Summary

• Existing status & control doesn’t support management approach need for a large SAN
  – Separate scheme for each transport adds complexity
  – Always dealing with inconsistent information

• XML-based schemes can
  – Cleanly map the existing information
  – Provide significant flexibility for the future
  – Work with all transport types now & in the future

• One more chance to get it right
  – Must support all future evolutions of SANs – even the ones we cannot anticipate
  – Leverage Internet trends
Feedback Please!

Even if it is only “this XML stuff is CRAZY”