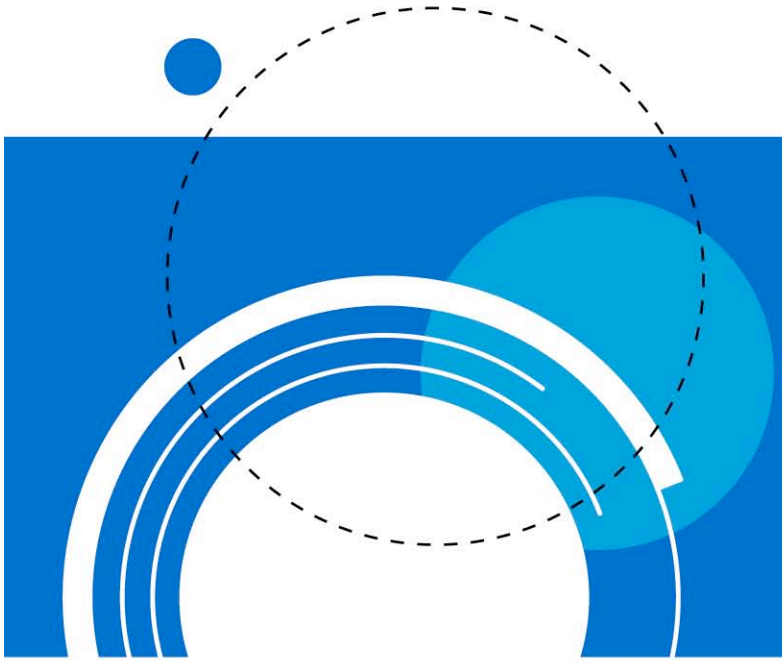


OSD-2 & XAM

Erik Riedel
Seagate Technology
May 2007

OSD-1 Commands

OSD-1 r10, as ratified in September 2004

Basic Protocol

- READ
- WRITE

} very basic

- CREATE
- REMOVE

} space mgmt

- GET ATTR
- SET ATTR

} attributes

- timestamps
- vendor-specific
  - opaque
  - shared

Specialized

- FORMAT OSD
- APPEND – write w/o offset
- CREATE & WRITE – save msg
- FLUSH – force to media
- FLUSH OSD – device-wide
- LIST – recovery of objects

Security

- Authorization – each request
- Integrity – for args & data
- SET KEY
- SET MASTER KEY

} shared secrets

Groups

- CREATE COLLECTION
- REMOVE COLLECTION
- LIST COLLECTION
- FLUSH COLLECTION

Management

- CREATE PARTITION
- REMOVE PARTITION
- FLUSH PARTITION
- PERFORM SCSI COMMAND
- PERFORM TASK MGMT

Status of OSD-2 Standard

Standard OSD-1 r10 for Project T10/1355-D (v1) ratified by ANSI in September 2004 after five years of SNIA effort

Draft OSD-2 r1 is out as of January 2007

- Currently being reviewed (take a look, send comments!)

SNIA TWG working on v2 features

- Richer collections – multi-object operations [in OSD-2 r1]
- Snapshots – managed on-device [proposal]
- Extended exception handling and recovery [proposal]
- Additional security support [proposal]
- Additional features (reservations, CLEAR, PUNCH) [proposal]
- Mapping of XAM onto OSD [ongoing w/ FCAS TWG]
- Quality of Service attributes [discussion]
- Device-to-device communication [discussion]

Expect OSD-2 r2 in August/September 2007

OSD-2 Richer Collections

Current (OSD-1) definition is preserved

- Two extensions are backward compatible

Stronger LIST and LIST COLLECTION Commands

- Return some member object attributes along with each OID
- Similar to READDIRPLUS in NFSv3

Multi-object operations

- Execute a single command on multiple objects
  - Simplicity and performance
- Use of “cloned” collections for status of multi-object operations
- QUERY – search attributes, returns list of matching objects
- REMOVE MEMBER OBJECTS – bulk remove
- SET MEMBER ATTRIBUTES – bulk attribute update
- GET MEMBER ATTRIBUTES – bulk attribute retrieval

OSD-2 Exception Management

Media error handling – fencing, error maps, recovery

- Report object-level errors, enable hosts or controllers to recover

Atomicity & Isolation – reported by devices

- Atomicity – complete an entire write or do not commit any data
  - Atomicity can be guaranteed on data, attributes or both
  - Reported using atomicity sizes: A, D, and C- limits
  - (limits may be zero)
- Isolation – not interleaving overlapping reads/writes
  - strong (per command); weak (per phase); none

File system check (OSD_FSCK) – externally-directed recovery

Boot epoch – updated on device reboot

- Prevents clients from performing commands on corrupted data
- Device-level attribute encoded into each capability, can be set by file manager and may be incremented on device reboot

OSD-2 Snapshots

Snapshots are point-in-time copies of partitions

Requires two credentials: one for source, one for destination

SNAPSHOT CREATE

- Creates a copy of the partition object and copies the content of the source partition to the newly created partition (snapshot)
- Either full copy (byte-by-byte) or COW are possible

DIFF READ

- Compares two objects

Snapshot chains handled via newly defined attributes

- Forward/backward pointers
- Destination pointers

COPY/CLONE can be applied to individual objects

- Used to create cloned collections for multi-object operations

Additional Changes in OSD-2

Extended Collections

Exception Management

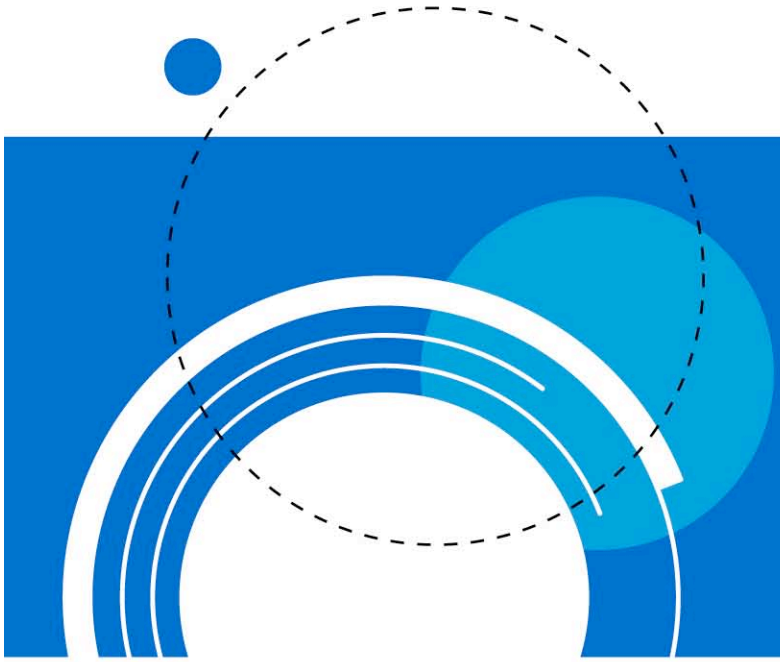
Snapshots

Security

- Support extended collections and snapshots, including multi-object capabilities

Miscellaneous proposals

- FC & SAS issue resolution with transfer of unused bytes
- Alignment issue with Attribute Lists
- 64-bit CDB Alignment issue
- Read-past-end-of-object semantics
- Setting attribute without data buffer
- Reservations
- Range-based FLUSH
- CLEAR & PUNCH commands



XAM over OSD

Seagate

eXtensible Access Method (XAM) Goals

See SNIA tutorial “Green Eggs and XAM”,
April 2007 for additional details

Compliance
- Integrated record retention and disposition metadata

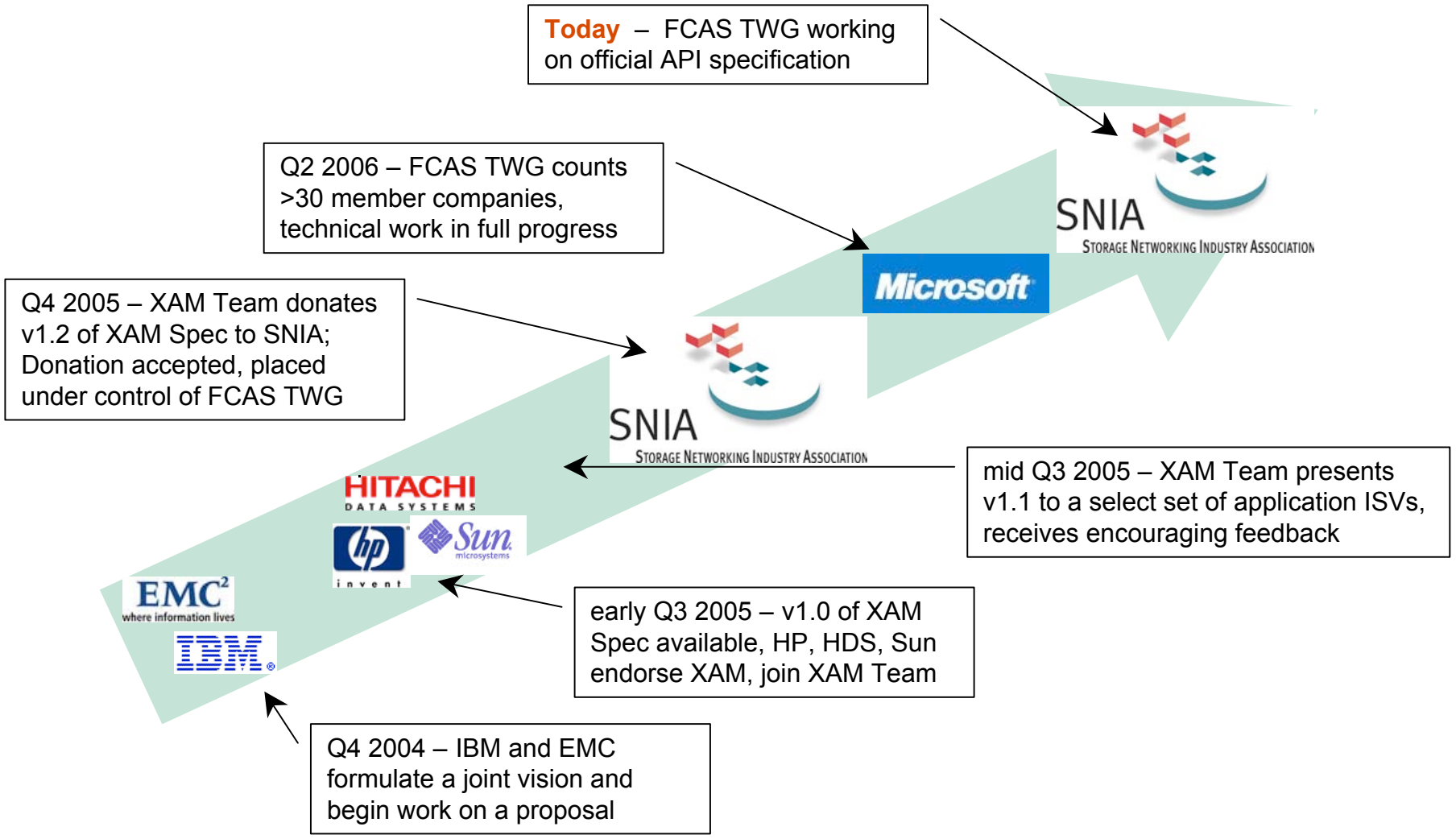
Standardized ILM (Information Lifecycle Management)
- Extensible metadata allows for external data classification and annotation
- Standardized ILM policies and ILM practices, managed by systems

Universal access to *reference data*
- Application independent and long-term storage and retrieval
- Application independent query interface

Standards-based interoperability
- XAM-compliant apps work with any XAM storage systems from any vendor
- Rich metadata allows multiple applications to share information
- Information can be easily migrated among XAM systems

➢ ***Driven by application vendors, analysts, ecosystem providers, and customers***

XAM History


Today – FCAS TWG working on official API specification
Q2 2006 – FCAS TWG counts >30 member companies, technical work in full progress
Q4 2005 – XAM Team donates v1.2 of XAM Spec to SNIA; Donation accepted, placed under control of FCAS TWG
mid Q3 2005 – XAM Team presents v1.1 to a select set of application ISVs, receives encouraging feedback
early Q3 2005 – v1.0 of XAM Spec available, HP, HDS, Sun endorse XAM, join XAM Team
Q4 2004 – IBM and EMC formulate a joint vision and begin work on a proposal
SNIA
Storage Networking Industry Association
Microsoft
HITACHI
DATA SYSTEMS
hp
invent
Sun
microsystems
EMC²
where information lives
IBM

XAM Status

FCAS Technical Work Group (TWG)

- V0.6 XAM Specification – just released (11 May)
  - open to feedback from SNIA members; available to public RSN
- V0.9 – targeted for September 2007
- V1.0 – targeted for December 2007 (previously October 2007)

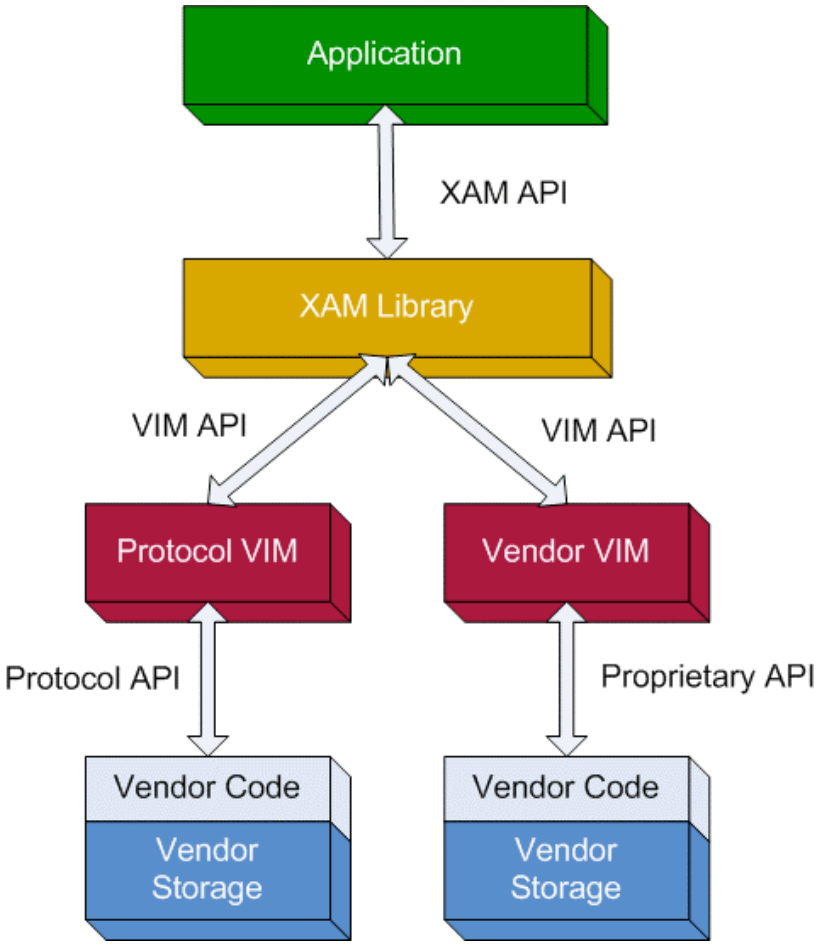
XAM SDK Technical Work Group (TWG)

- Technical group recently formed to develop XAM reference software
- Currently 18 members (6 companies/institutions)
- Free to join, requires companies to sign the new SNIA IP Policy
- Current work items are: XAM Library and Reference VIM (on a FS)
- OSD VIM is proposed as a work item and currently being voted on

XAM Initiative

- A set of companies promoting development of XAM software
- Determines the priorities of the XAM SDK TWG
- Fee-based membership to support development activities

XAM Architecture (1)



An application uses the libxam.dll to 'connect' to a specified XSystem.

- A single application may connect to multiple XSystems simultaneously
- Multiple applications may connect to a single XSystem simultaneously

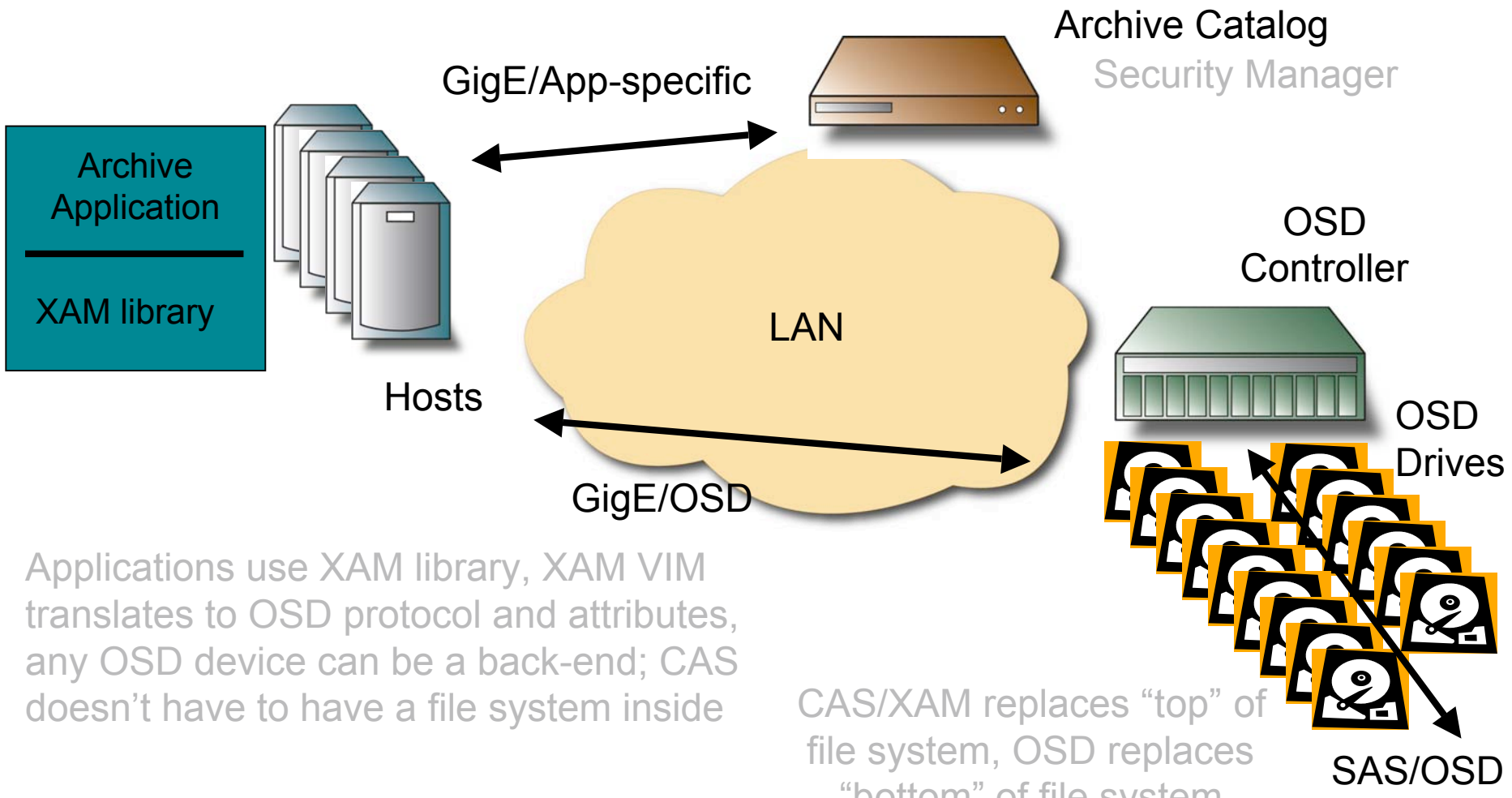
An XSystem is not identical to a vendor's "storage box", but a logical abstraction which should be viewed as 'bag of storage'.

The application may be required to authenticate at the time the connection to an XSystem is established.

The application uses libxam.dll to store/retrieve "content objects" to/from the XSystem.

These "content objects" are bundles of data and metadata, and are called XSets.

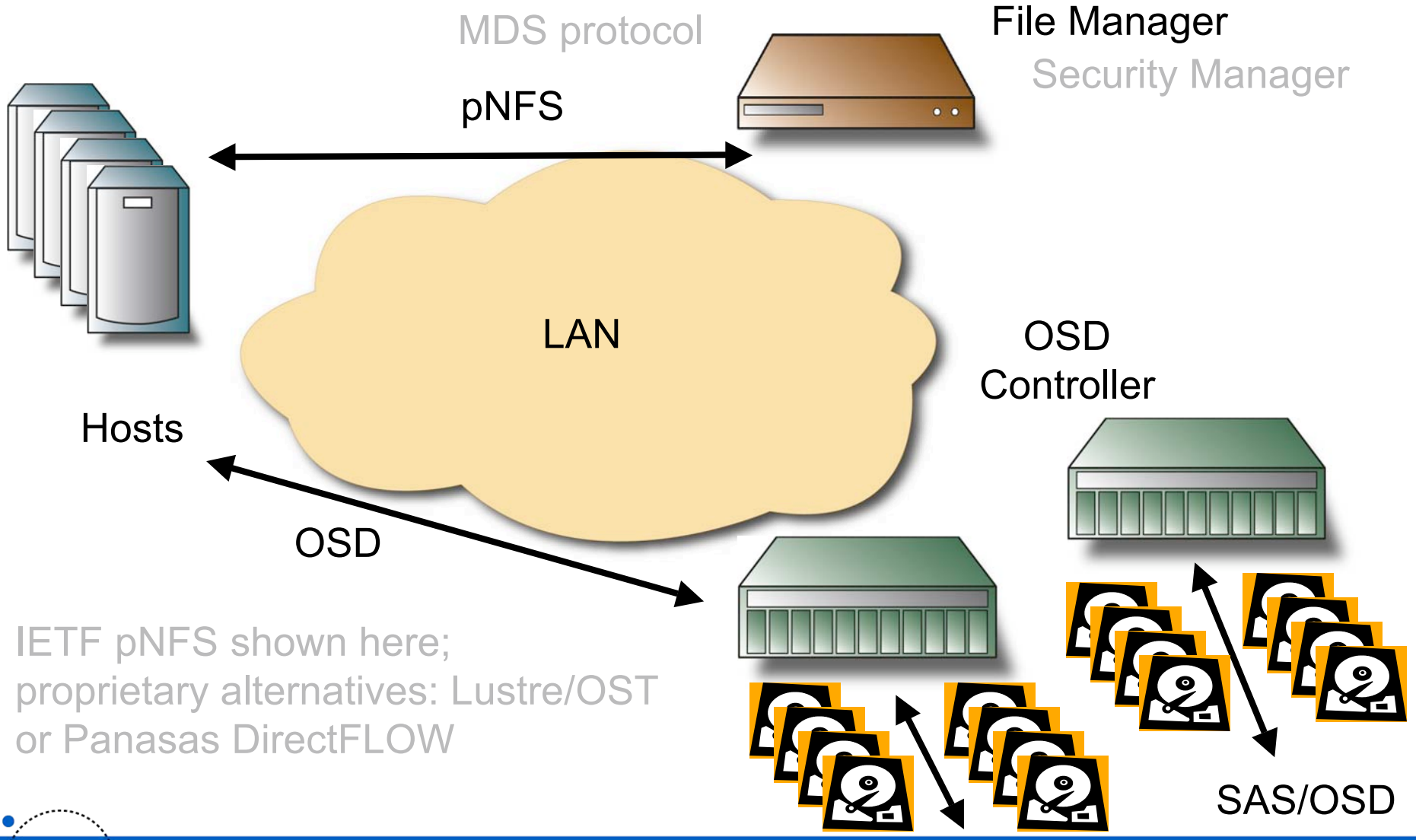
CAS with OSD



Applications use XAM library, XAM VIM translates to OSD protocol and attributes, any OSD device can be a back-end; CAS doesn't have to have a file system inside

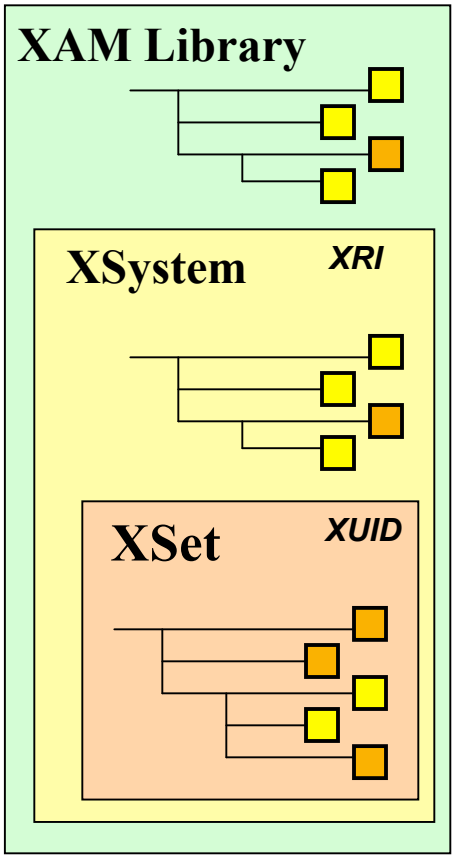
CAS/XAM replaces "top" of file system, OSD replaces "bottom" of file system

Scalable NAS with OSD



IETF pNFS shown here;
proprietary alternatives: Lustre/OST
or Panasas DirectFLOW

XAM Architecture (2)

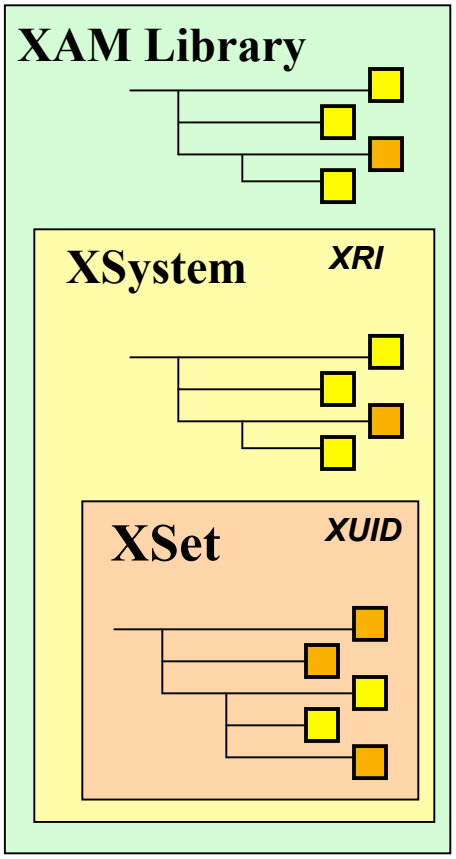


3-levels of objects (hierarchy)

- XAM Library: top level object for the XAM API library
  - Contains methods to get and set fields describing the configuration of the XAM system
  - Acts as a factory of XSystems
- XSystem: object that abstracts the connection between application and storage systems
  - Encapsulates any resource management associated with the connection
  - Contains methods used to authenticate operations
  - Acts as a virtual storage system, partitioning the content
- XSet: object that contains application/user data and metadata
  - Has a globally unique identifier, called XUID (80 bytes)

Each level of XAM abstraction (XAM Library, XSystem, XSet) contains “fields” (of type “property” or “xstream”)

XAM Architecture (3)



Two types of Fields:

- Properties
  - **"Simple" Types (Boolean, Uint64, Float64, String, DateTime, XUID)**
  - **Type checked/enforced by Storage System**
  - **Manipulated via "Property Get/Set" Methods**
- Streams
  - **Bytestreams, bound in Length**
  - **Type assumed to be a valid MIME-type, but not checked/enforced by Storage System**
  - **Manipulated via Posix-style I/O Methods (e.g. open, read, write, close)**

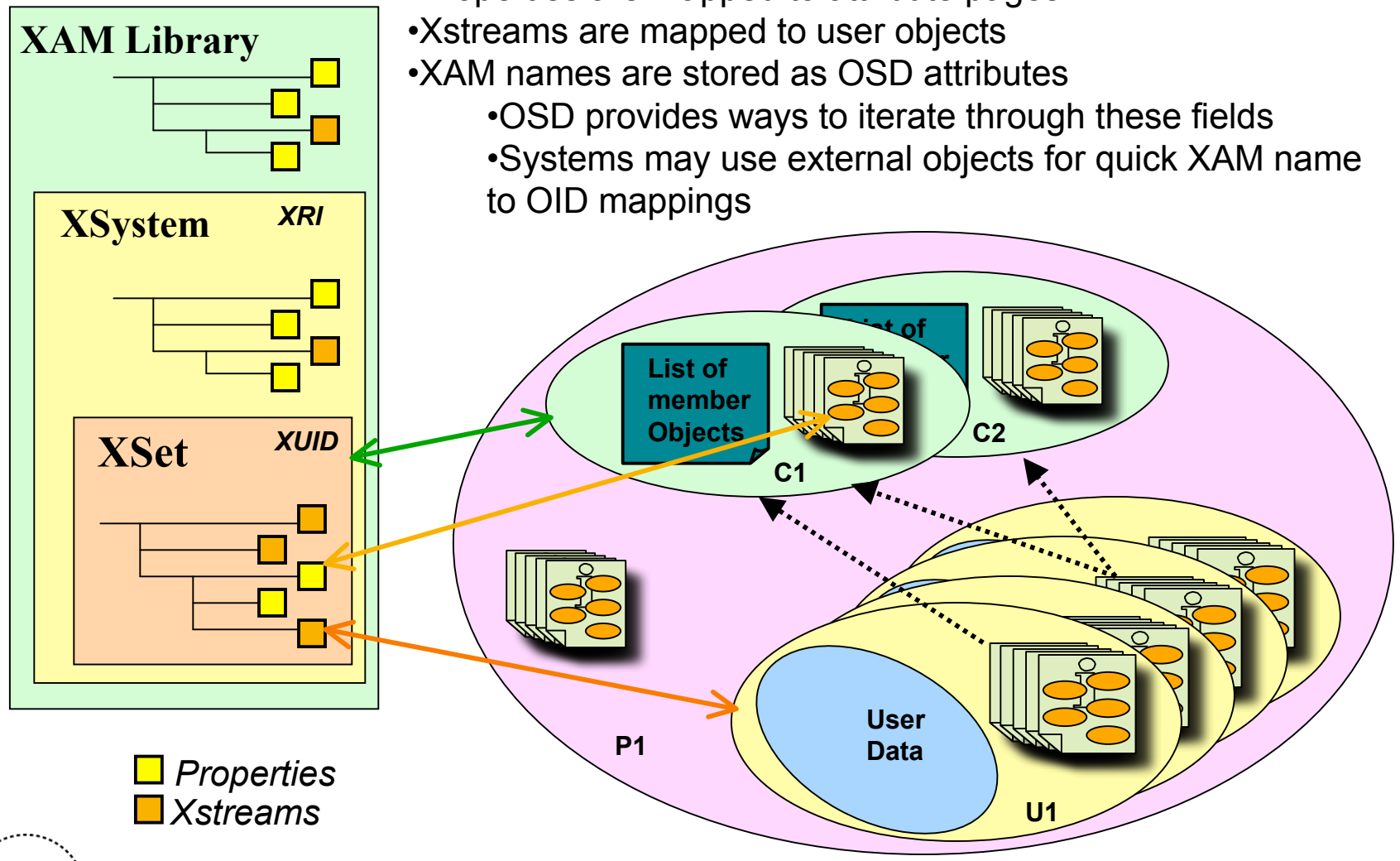
Each Field Has Four Basic Attributes:

- **Type – stype for Properties, any other MIME-type for streams**
- **Length – The actual size of the field's value**
- **Readonly – Flag indicating whether field is modifiable by applications**
- **Fixed – Flag indicating whether field is Fixed/Variable content**
- **Manipulated via "Attribute Get/Set" methods**

XAM to OSD Recommended Mapping

Option 1

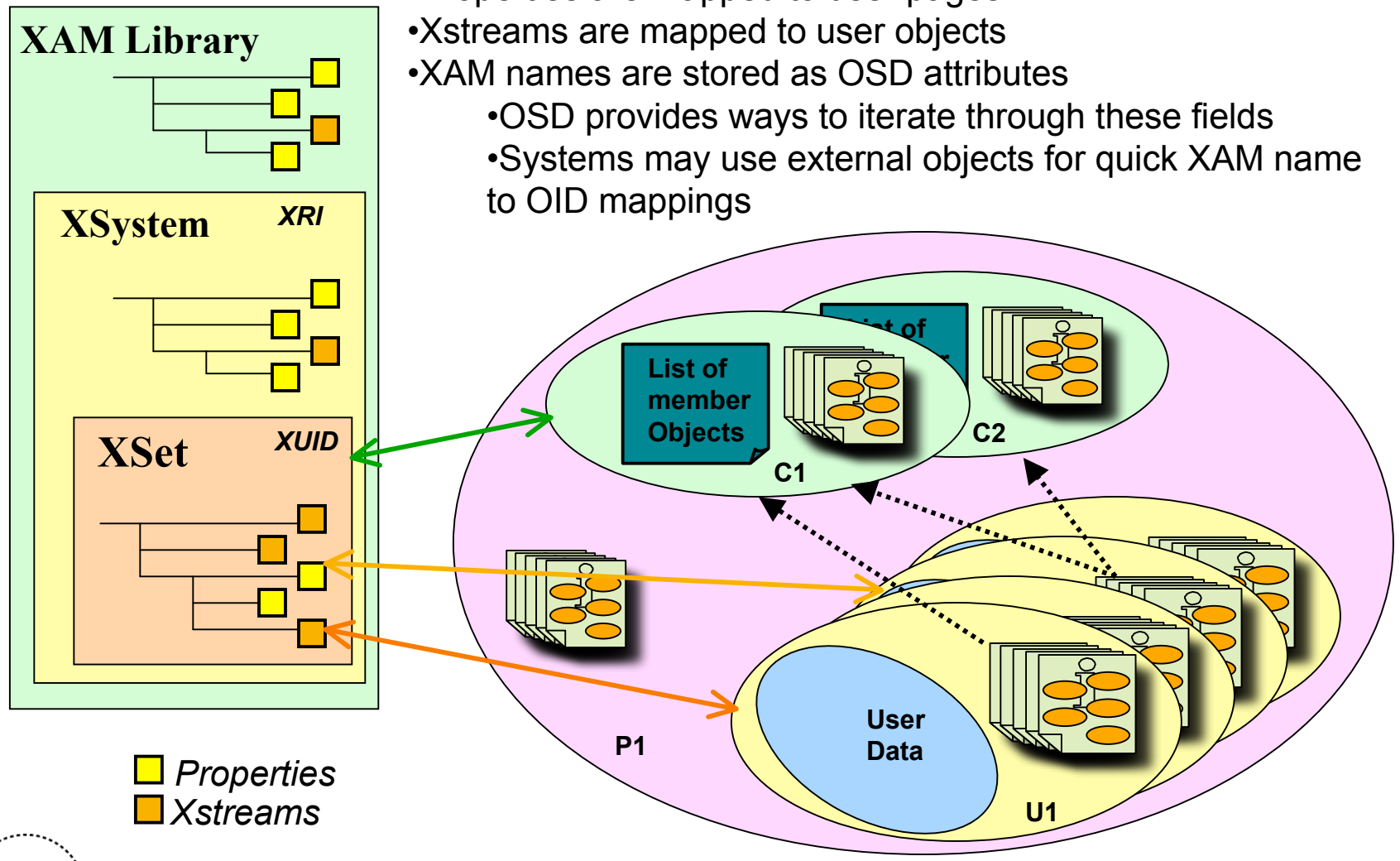
- XSETs are mapped to collection objects
- Properties are mapped to attribute pages
- Xstreams are mapped to user objects
- XAM names are stored as OSD attributes
  - OSD provides ways to iterate through these fields
  - Systems may use external objects for quick XAM name to OID mappings



XAM to OSD Recommended Mapping

Option 2

- XSETs are mapped to collection objects
- Properties are mapped to user pages
- Xstreams are mapped to user objects
- XAM names are stored as OSD attributes
  - OSD provides ways to iterate through these fields
  - Systems may use external objects for quick XAM name to OID mappings



Ongoing Work

XAM Field Attributes

- How are they mapped to OSD attributes
  - Probably define new OSD attributes
  - (user-defined vs. standards-defined)

Mapping of XAM methods to OSD commands

Mapping of default fields (Xstreams, properties) to OSD

Management Policies

- Retention, deletion, storage

Jobs

- Submit and halt
- Query processing
  - Possible overlap with OSD-2 QUERY command for level 1 queries

Roadmap

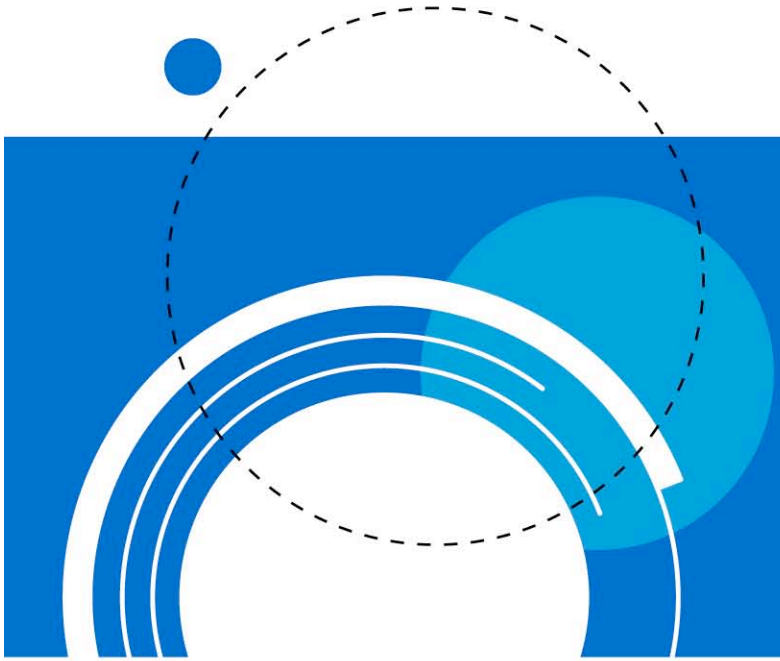
XAM to OSD Mapping – ongoing work

- Basic object mapping is done, but a lot more details to go…

Plan to have a complete document in Summer 2007

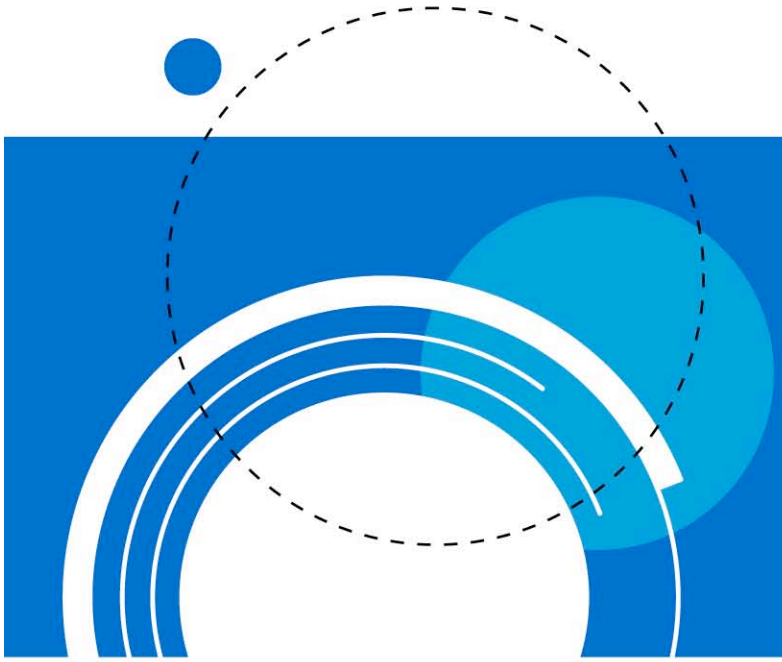
- Available to the general public in the form of a White Paper and Best Practices document
- Joint FCAS and OSD group work

Demonstrate a prototype implementation among group of OSD partners once XAM reference is available



Backup Slides

Seagate



Strong LIST (COLLECTION) and Multi-Object Operations in OSD-2

Seagate

Strong LIST and LIST COLLECTION

–Idea: Return some member object attributes along with each OID

–Mechanism:

- A 1-bit field called “LIST ATTR” is added to CDB
- When this bit is set, clients can request *member object* attributes via the “Get and set attributes parameters” field
  - Just like they request regular object attributes
  - OSD uses attribute page numbers to differentiate between container object and member object attributes
- OSD returns requested member object attributes in the “command data or parameter data segment” of the data-in buffer alongside the OIDs

Multi-Object Operations (1)

–Idea: Execute a single command on multiple objects

- Simplicity
- Performance

–Mechanism:

- Can only be issued to “cloned” collection objects
  - Exception for REMOVE MEMBER OBJECT command
- Operations can be done one at a time or in parallel
  - No order assumed
- As objects are operated on, they are removed from the collection
  - At any point, the collection contains only those objects that have not been operated on, yet.
  - A new attribute in Collection Information Attributes Page will be defined to store the number of user objects in the collection to help track progress
- Command returns when the whole operation is completed or an error is detected
  - Might be a long time …

Multi-object Operations (2)

–Error Recovery: if a MO operation fails in the middle for any reason, OSD will

- Issue no more sub-commands as part of the MO command,
- Complete any sub-commands that are currently in-flight,
- Fence any objects that have been detected as damaged (possibly multiple objects),
- Return an error code for the **first** damaged object

–A client can re-issue the same command after damaged objects have been fixed

- Operation will resume and only those objects that have not been operated before will be operated on

–If an ABORT TASK is received during the MO operation, OSD will ensure objects are left at a stable state or *fenced*

Multi-object Operations (3)

–Multi-object operations defined:

- QUERY
- REMOVE MEMBER OBJECTS
- SET MEMBER ATTRIBUTES
- GET MEMBER ATTRIBUTES

Multi-object Operations (4)

QUERY Command

–Idea: provide a search mechanism for OSD based on attributes

- Upon receipt of a QUERY command, OSD returns the list of all objects whose attributes match the specified criteria
- E.g., List all objects that were created within a certain time range

–Mechanism:

- Similar to LIST COLLECTION command
- Requested attribute values are specified in the modified “Get and set attributes parameters” field as follows:
  - Attribute page number
  - Attribute number
  - Minimum value desired
  - Maximum value desired

Multi-object Operations (5)
REMOVE MEMBER OBJECTS Command

–Idea: multi-object version of the REMOVE command

–Member objects are removed, but not the collection object

–Unlike other MO operations, can be issued to a regular collection object

Multi-object Operations (6)
SET MEMBER ATTRIBUTES Command

–Idea: multi-object version of SET ATTRIBUTES command

–Same attribute values are stored on all the member objects

–Attributes are specified in the “Get and set attributes parameters” field

- OSD uses attribute page numbers to differentiate between container object and member object attributes

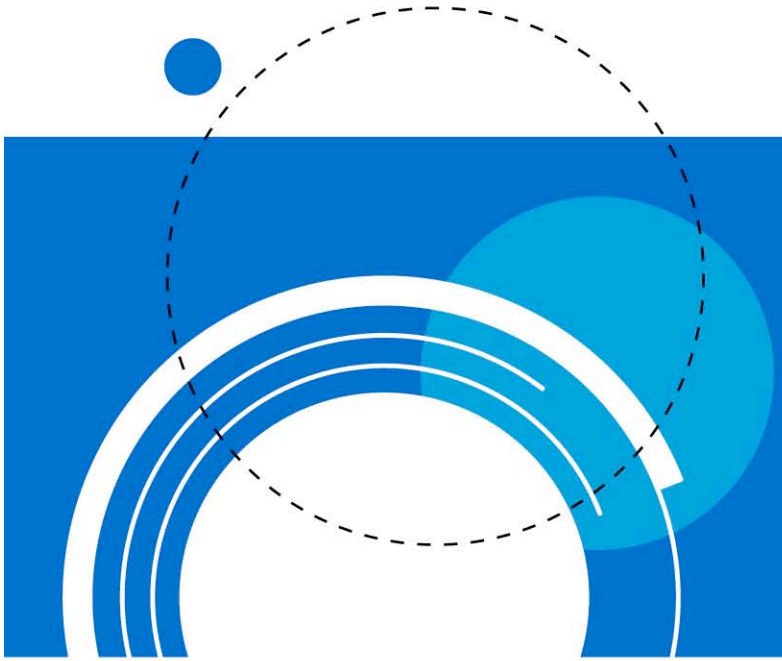
Multi-object Operations (7)
GET MEMBER ATTRIBUTES Command

–Idea: multi-object version of GET ATTRIBUTES command

–Similar to strong LIST but provides total randomness

–Attributes are specified in the “Get and set attributes parameters” field

- OSD uses attribute page numbers to differentiate between container object and member object attributes



Miscellaneous Proposals Approved for OSD-2

Seagate

Alignment Issue with Attribute Lists

Set/Retrieve attribute list alignment

- Alignment not considered in version 1 OSD
  - > Just a stream of bytes
- Some cpu's require alignment
  - > SPARC, MIPS, IA-64, ARM
  - > Must explicitly copy bytes to access fields
- Some cpu's just slow down when data unaligned
  - > X86, x86-64, PPC
- Need to change list such that it can be aligned

Courtesy of **Todd Pisek**

New set/retrieve list proposal

- Extend list header length from 16 bits to 32 bits
  - > Prepend 2 bytes of padding for alignment
  - > Not all 32 bits need to be defined as length
- Change list entry format
  - > Add 6 bytes after attribute length
  - > Aligns attribute value to 64-bits
- Require list entries to be 64-bit aligned
  - > Padding after attribute value undefined

C example

- typedef struct {
  uint8_t list_type;
  uint8_t reserved[3];
  uint32_t list_length;
  } new_list_header_t;
- typedef struct {
  uint32_t page;
  uint32_t num;
  uint16_t len;
  uint8_t reserved[6];
  } new_entry_t;

64-bit CDB Alignment Issue

–There are several 64-bit fields in the CDB that are not aligned at 8 byte boundaries. On a 64-bit Sun SPARC, this is very inconvenient, since attempting to access these fields as 64 bit values will cause an address fault. They have to pull the fields out 32 bits at a time.

–The CDB could easily be rearranged (by moving a reserved field) so that all 64 bit values fall on 8 byte boundaries.

APPEND:
- LENGTH field (offset: 36)

CREATE AND WRITE:
- LENGTH field (offset: 36)
- STARTING BYTE ADDRESS field (offset: 44)

FORMAT OSD:
- FORMATTED CAPACITY field (offset: 36)

LIST:
- ALLOCATION LENGTH field (offset: 36)
- INITIAL OBJECT_ID field (offset: 44)

LIST COLLECTION:
- ALLOCATION LENGTH field (offset: 36)
- INITIAL OBJECT_ID field (offset: 44)

PERFORM TASK MANAGEMENT FUNCTION:
- TASK TAG field (offset: 44) (variable length?)

READ:
- LENGTH field (offset: 36)
- STARTING BYTE ADDRESS field (offset: 44)

WRITE:
- LENGTH field (offset: 36)
- STARTING BYTE ADDRESS field (offset: 44)

FC & SAS Issues Resolved

–Several requirements in the SCSI OSD standard conflict with the FCP and SAS transport standards.

- Fill bytes issue: With the transfer length of buffer segments in bytes, fill bytes may be needed at the end of each segment transfer.
  - In both FCP and SAS, fill bytes are only allowed on the last frame (highest offset) in each direction per command.
- Buffer gaps issue: Unused bytes are not transferred.
  - In the SCSI architecture, modify data pointer is required to support any out of order transfers.
  - FCP requires all bytes in the Data-Out and Data-In buffers be transferred.
    - Modify data pointers are optional but not widely supported.
    - There is currently a proposal to remove modify data pointers from the standard.
  - SAS requires the offset of each frame be the sum of the data length and the data offset of the previous frame.
    - Modify data pointers are not supported.

–Solution: allow unused bytes to be transferred.

Range-based FLUSH

–Purpose:

- Update the current FLUSH command to enable clients to specify a range of bytes they wish to flush to permanent storage (useful for large objects).

–Mechanism:

- Modify the CDB to include the following fields:
  - Use bytes 32-39 for LENGTH
  - Use bytes 40-47 for STARTING BYTE ADDRESS
- Define a new value for FLUSH SCOPE field (Table 58 in OSD-2 r1)
  - 00b: User object data and attributes
  - 01b: User object attributes only
  - 10b: User object data range and attributes
  - 11b: Reserved

CLEAR Command

–Clear is a specialized write operation in which a range in the object content needs to become all ‘0’s.

- Purpose: to efficiently make a ‘0’-filled hole in an object (sparse objects)
- OSD should support this command, now that it manages the block allocation, to support file clear effectively.

–Mechanism:

- Define a new CDB for CLEAR that is very similar to the WRITE CDB (new service action code and no user data in the data-out buffer)
- Can probably use WRITE permission bit, no need to define a new one.

PUNCH Command

–PUNCH is a specialized write operation similar to CLEAR only that it “zips up” the object to eliminate the hole completely.

- For an object of 1024 bytes, if 256 bytes at offset 256 are punched, the new object will have 768 bytes and the bytes formerly at offset 512-1023 are now at 256-767.
- This is a logical companion to clear.
- Also called “CUT”.
- Purpose: to efficiently remove a section of an object

–Mechanism:

- Define a new CDB for PUNCH that is very similar to the WRITE CDB (new service action code and no user data in the data-out buffer)
- Can probably use WRITE permission bit, no need to define a new one.

Set One Attribute without Data Buffer

Setting Attribute Without Data Buffer

- Currently, setting one attribute required Data-out
- Allocating, pinning, and mapping buffers is expensive
  - > A lot of expense for a 4 or 8 byte value
- Need a simple way to set one attribute
- Proposal is to place attribute value in the CDB

New Short Set Attribute Proposal

- Exploit reserved CDB format codes for get/set attributes
  - > Version 1 spec, table 42 section 5.2.2.1
- New format code 01b
  - > Bytes 52-55 Set Attribute Page
  - > Bytes 56-59 Set Attribute Number
  - > Bytes 60-63 Set Attribute Length
    - > Legitimate values are 0 through 16
  - > Bytes 64-79 Set Attribute value (1-16 bytes)

Read Past End of Object

- Section 6.17 Read (Version 1 Spec)
- Currently, there are 2 different errors for reading past the end of object:
  1. If the STARTING BYTE ADDRESS field specifies a byte that is beyond the user object logical length attribute value in User Object Information attributes page (see 7.1.2.11), then: a) No bytes shall be transferred; and b) the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB
  2. If the values in the LENGTH field and STARTING BYTE ADDRESS field result an attempt to read a byte that is beyond the user object logical length attribute value in the User Object Information attributes page, then: a) the bytes between the starting byte address and the user object logical length shall be transferred; b) the command shall be terminated with a CHECK CONDITION status, with the sense key shall be set to RECOVERED ERROR and the additional sense code set to READ PAST END OF USER OBJECT; c) the command-specific information sense data descriptor (see SPC-3) shall be included in the sense data; and d) the COMMAND-SPECIFIC INFORMATION field shall contain the number of bytes transferred.

- Proposal to return 2. for both cases
- If the STARTING BYTE ADDRESS field specifies a byte that is beyond the user object logical length, then no bytes will be transferred and the COMMAND-SPECIFIC INFORMATION field shall contain zero.

Reservations

Problem

- Guarantee that space will be available on the OSD when a client flushes the dirty page created by the write().
- E.g.
  - application does delayed write() on a client.
  - write() returns.
  - client flushes the dirty page.
  - client gets ENOSPC.

Advantages

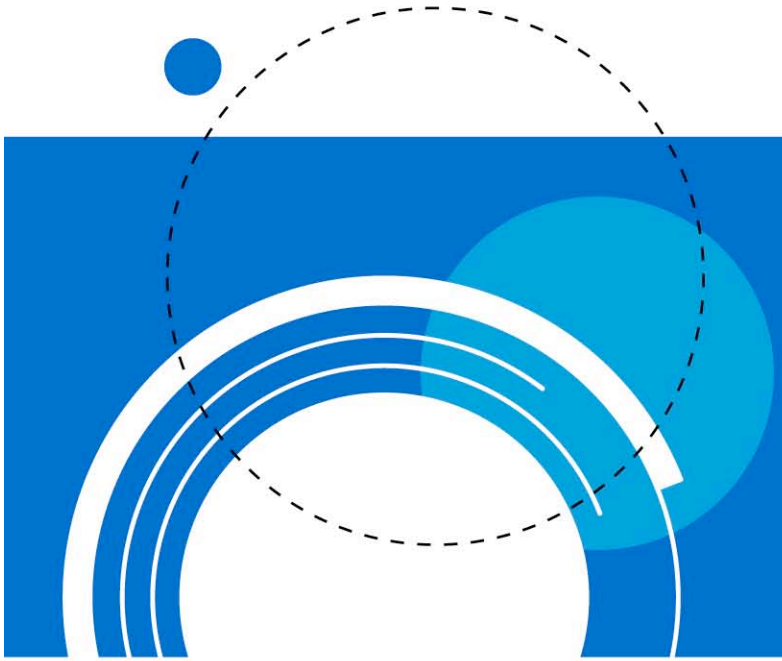
- Very few messages to guarantee space availability. Clients can aggressively reserve space before the write and release the reservation after the write is over.

Issues

- Snapshots - should allocation done for copy on writes be accounted in the objects space reservation ?.

Proposal

- Add an actual_data_space attribute to the object.
- The actual_data_space shows the amount of space used to store data for the object. The actual_data_space does not include the space used to store attributes or other metadata of the object.
- Add a reserved_data_space attribute to the object.
- Setting the reserved_space attribute to a non zero value results in the OSD reserving so much of space for the object.
- The reserved_data_space includes the amount of space used to store the data for the object.
- Setting the reserved_data_space to a non zero value ensures that allocating (reserved_data_space - actual_data_space) bytes will not fail due to ENOSPC errors.
- As data is written to the object, the actual_data_space attribute is incremented by the OSD to show the amount of space used to store data in the object.
- Client file systems can look at the difference between reserved_data_space and actual_data_space to find out the extra space reservation for the object.
- The reserved_data_space and actual_data_space attributes exist for a partition also.
- The actual_data_space attribute for the partition shows the amount of space used to store data for objects in the partition.
- The reserved_data_space attribute for the partition can be used to reserve space for objects in the partition.



Security Related Changes in OSD-2

Seagate

Boot Epoch

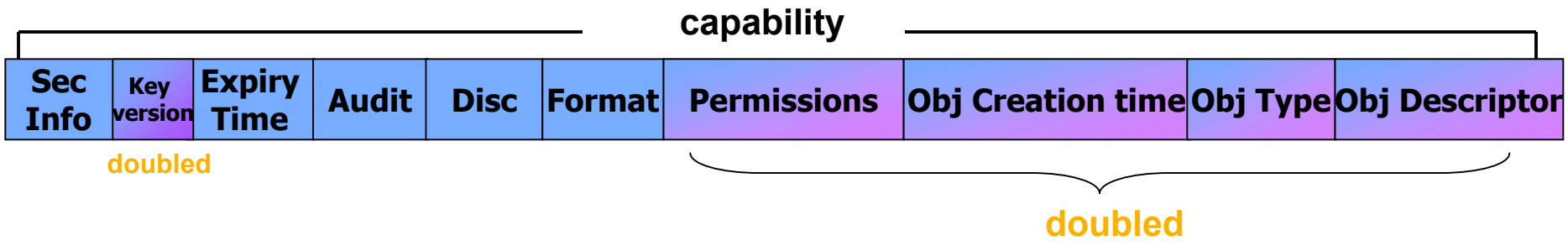
–What is boot epoch?
- Associated with the OSD
- Settable by security admin to an arbitrary value
- May be incremented by target on reboot
  - A cyclic value
- Role: lock out client actions on OSD

–2-byte integer, included in capability
- Must match value of attribute in root object

–Root security/policy attribute
- Settable

–Enforcement
- Capability boot epoch must equal root attribute value
- Zero value implies ‘bypass’

Changes to Capability



Capability Format – more formats

Key version x 2

Object created time x 2

Object type x 2

[Object descriptor, Object descriptor type] x 2

Permissions

- Duplicate bit vector
- More bits for new operations

Integrity value calculation, extended to K1 and K2

Extends by 32 bytes

Extending integrity value calculation

Integrity value calculation extended to two keys

K1 – key of source partition

K2 – key of destination partition

Goal: Capability key depending on K1 AND K2

temp_key = HMAC(K1, capability)

capability_key = HMAC(K2, temp_key)

command_integrity_value = HMAC(capability_key, CDB)

Range Capabilities

Objective

- Restrict capability to a given range within the object

Two additional fields (8 bytes each)

- STARTING_BYTE_OFFSET
- LENGTH

Applicable for user-object and the following commands:

- Create_and_write, Write, Read, Append

Enforcement

- CDB range must be within the capability range
- Appended data should not exceed capability range

Capabilities for Specific Attributes

Support finer grain protection over attributes

- Specify in capability which attributes it protects

Assumption:

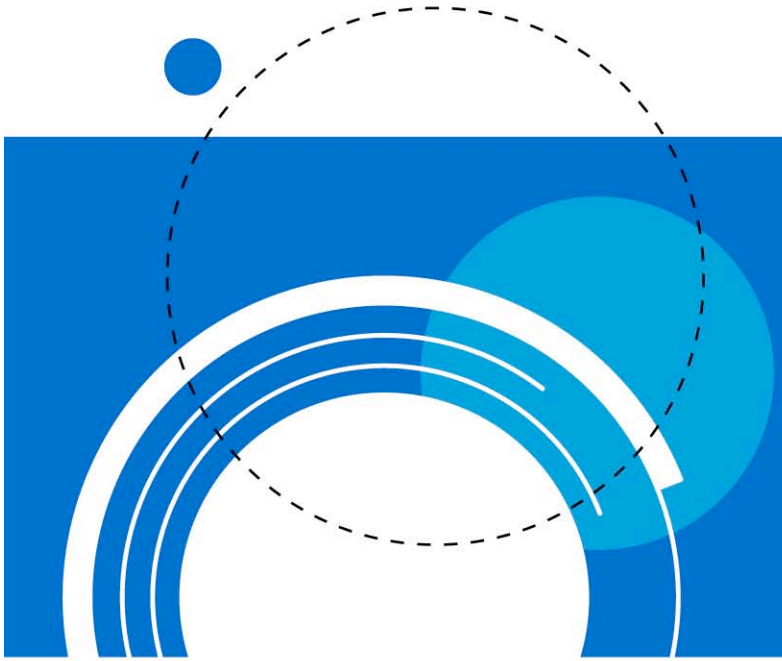
- Small # of subsets of attributes to protect
- Subsets are relatively static

Mechanism

- Define a user-defined page P
- Specify the page number P in a new capability field
  - ALLOWED_ATTR_PAGE
- P is a list of [attr_page, attr_number]
  - Allow syntax for [attr_page, *]

Enforcement

- All attributes accessed by command must be listed in the page



Exception Management

Seagate

Exception Management
Atomicity

–Atomicity is roughly the guarantee that all of a command's effects are either completely committed w/in the OSD or none of a commands effects are seen within the OSD. In other words;
- either all data is written or no data is written
- either all attributes are updated or no attributes are updated

–Atomicity can occur on data, attributes or both
- Controlled by atomicity size
  - D-limit: maximum amount of data that is guaranteed to be written atomically
  - A-limit: maximum amount of user-settable attributes that are guaranteed to be written atomically
  - C-limit: maximum amount of data PLUS user-settable attributes that are guaranteed to be written atomically
- OSD MUST implement D-, A- and C-limits, but those limits may be ZERO

–What about OSD maintained (i.e. system-settable only) attributes?
- These MUST be maintained atomically by the OSD and are not considered in the D-, A-, or C- limits
- e.g., timestamps, capacity_used,

–What about single user-settable attributes?
- Is it possible for a setattr on a single attribute to be non-atomic?
- Yes, it is possible for it to be non-atomic *with respect to* system-settable attributes if the A-limit = 0.

Exception Management
Isolation

- Changes made by one operation are not visible to other simultaneous operations on the system until its completion
  - Avoids data from two writes becoming intermingled
  - Avoids attributes from two setattr becoming intermingled
- Possible solutions
  1) Strong isolation – Isolate between commands (only needs to be done on a per-object basis)
  2) Weak isolation – Isolate between command phases (e.g., no two commands in the data-phase can modify an object at the same time)
     - Can be relaxed by implementation if data regions are non overlapping
  3) No isolation

Exception Management

Misc.

–New Command: OSD FSCK

- To fix FS issues that cannot be fixed by outside world

–New Command: READ MAP

- Returns a map of the object indicating DATA, DAMAGED, and HOLE sections of the object.

–Media Error Handling

- Damage discovery, handling, and reporting