Introducing the new Cloud Data Management Interface

Standardizing the Cloud for Interoperability

Mark A. Carlson
Sun and SNIA Technical Council
Chair, SNIA Cloud Storage TWG

Abstract

- The use of the term cloud in describing new models for storage and computing arose from architecture drawings that typically used a cloud as the dominant networking icon. The cloud conceptually represented any to any connectivity in a network, but also an abstraction of concerns such the actual connectivity and the services running in the network that accomplish that connectivity with little manual intervention.

Agenda

- Some background on cloud storage
- The Cloud Storage Reference Model
- CDMI – the interface
  - Data Objects
  - Containers
  - Accounts
  - Capabilities
  - Queues
  - Data System Metadata

SNIA Cloud Storage TWG

- Launched April 2009
  - 140 Technical Work Group members (50 active)
  - Google group for broader community (276 members): http://groups.google.com/group/snia-cloud
- Published first documents June 2009
  - Use Cases/Requirements, Reference Model
  - Public web page http://snia.org/cloud
- Draft of Cloud Data Management Interface (CDMI)
  - Targeted at ANSI and ISO certification
- Working on a CDMI Reference Implementation
  - Portable, works on any filesystem

SNIA Cloud Storage Initiative

- Launching at Fall SNW 2009
  - Planning press release listing charter members
  - Cloud Pavilion on show floor
- Supporting the development and adoption of CDMI, Cloud Storage
- Marketing, Outreach, Education on Cloud Storage
- Requirements gathering
- Premier

SNIA Cloud Storage Initiative

Some Customer Scenarios

- Escalating Amount of Data, becoming more expensive to manage
  - Is that old data valuable still?
- Meet governance needs
  - Who is in charge of deleting old data, finding it again?
- New projects with a desire not to have isolated local storage
  - Do they already go around existing storage practices?
- Storage of important data
  - How do you make sure it's requirements are being met?
- Storage for new Cloud Computing projects
  - Where does the data live long term?

Some use cases

- Offerings in the Data Storage as a Service space are increasing in capabilities
  - Additional service levels beyond "Best Effort" storage
  - Local appliances that "wrap" cloud storage for legacy applications
  - Cloud Storage offerings that layer onto best effort services
- These offerings go beyond merely storing the data, and start to account for data with differing requirements
  - There is a danger that the resulting complexity may cause the simplicity of cloud storage to be lost
  - There is an increasing "exit cost" to move from one vendor to another

Overview

- When discussing cloud storage and standards, it is important to distinguish the various ***resources*** that are being offered as services.
- These resources are exposed to clients as Functional interfaces (Data Path) and are managed by Management interfaces (Control Path).
- We explore the various types of interfaces that are part of offerings today and show how they are related.
- We propose a model for the interfaces that can be mapped to the various offerings as well as form the basis for rich cloud storage interfaces into the future.

What is Cloud Storage?

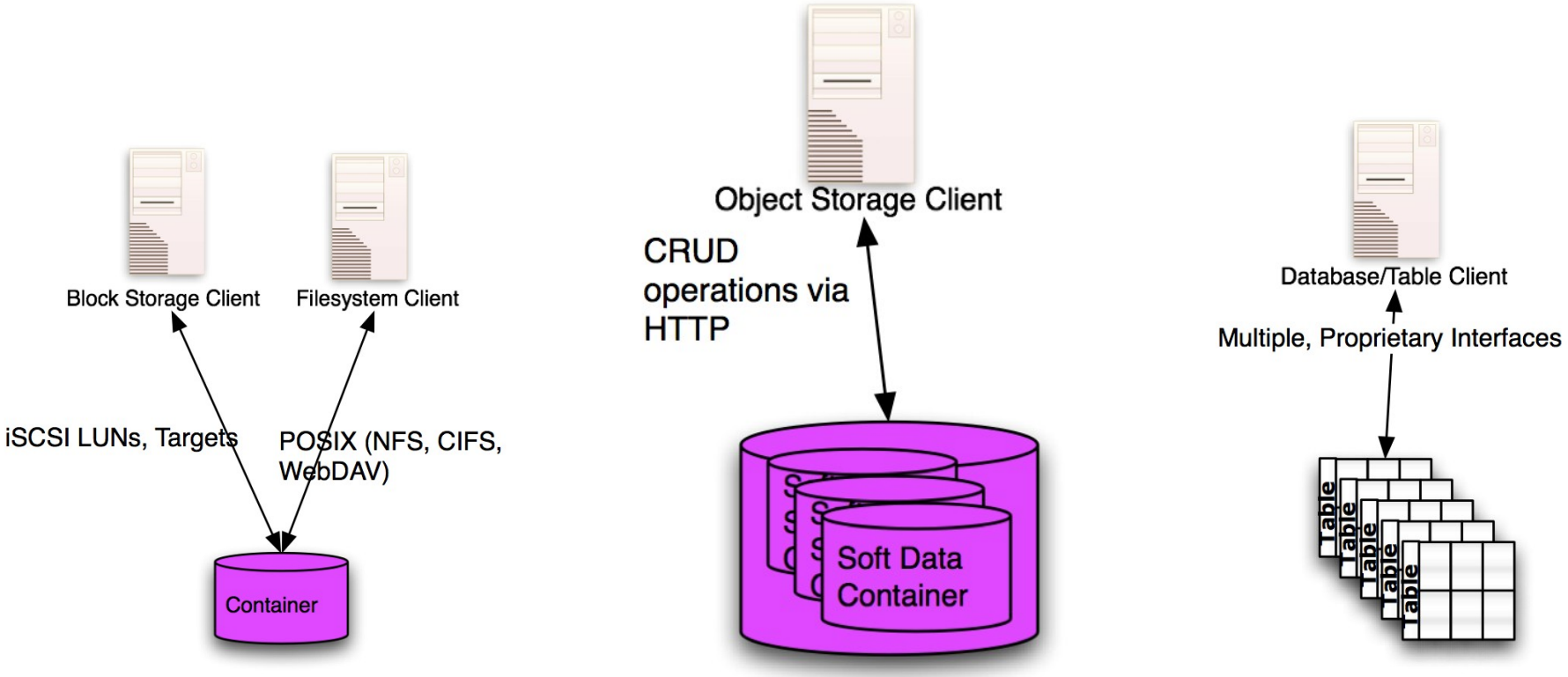
- The use of the term ***cloud*** in describing these new models arose from architecture drawings that typically used a cloud as the dominant networking icon.
- The cloud conceptually represented any to any connectivity in a network, but also an abstraction of concerns such the actual connectivity and the services running in the network that accomplish that connectivity with little manual intervention.

Cloud Storage Defined

- This abstraction of complexity and promotion of simplicity is what primarily constitutes a cloud of resources, regardless of type.
  - An important part of the cloud model in general is the concept of a pool of resources that is drawn from upon demand in small increments (smaller than what you would typically purchase by buying equipment).
  - The recent innovation that has made this possible is virtualization.
- Thus cloud storage is simply the delivery of virtualized storage on demand. The formal term we proposed for this is *Data Storage as a Service* (**DaaS**).
- ***Data Storage as a Service***
  - *Delivery over a network of appropriately configured virtual storage and related data services, based on a request for a given service level..*

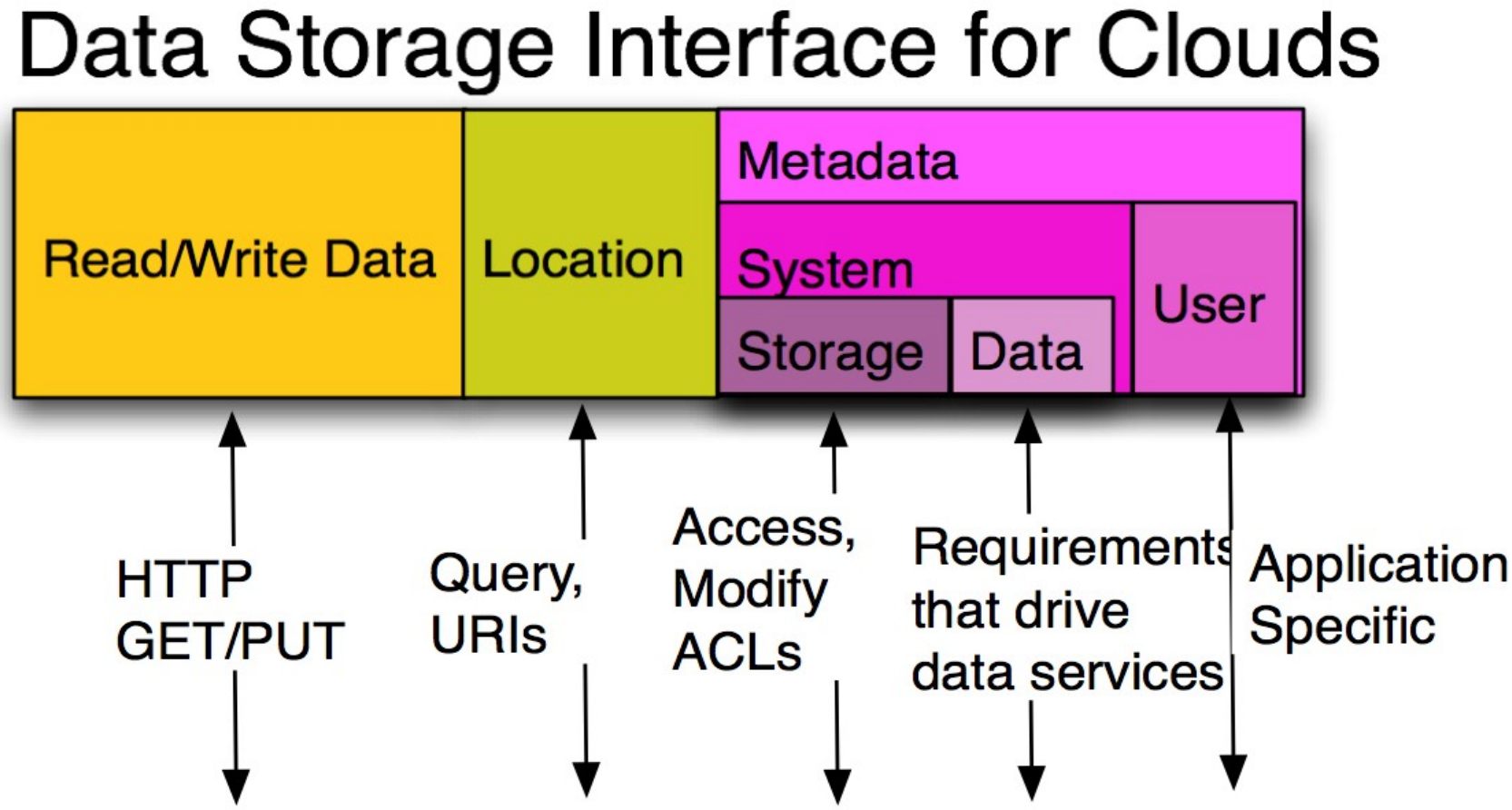
A look at some existing Cloud APIs

- What are some of the offerings and their Data Storage Interfaces?



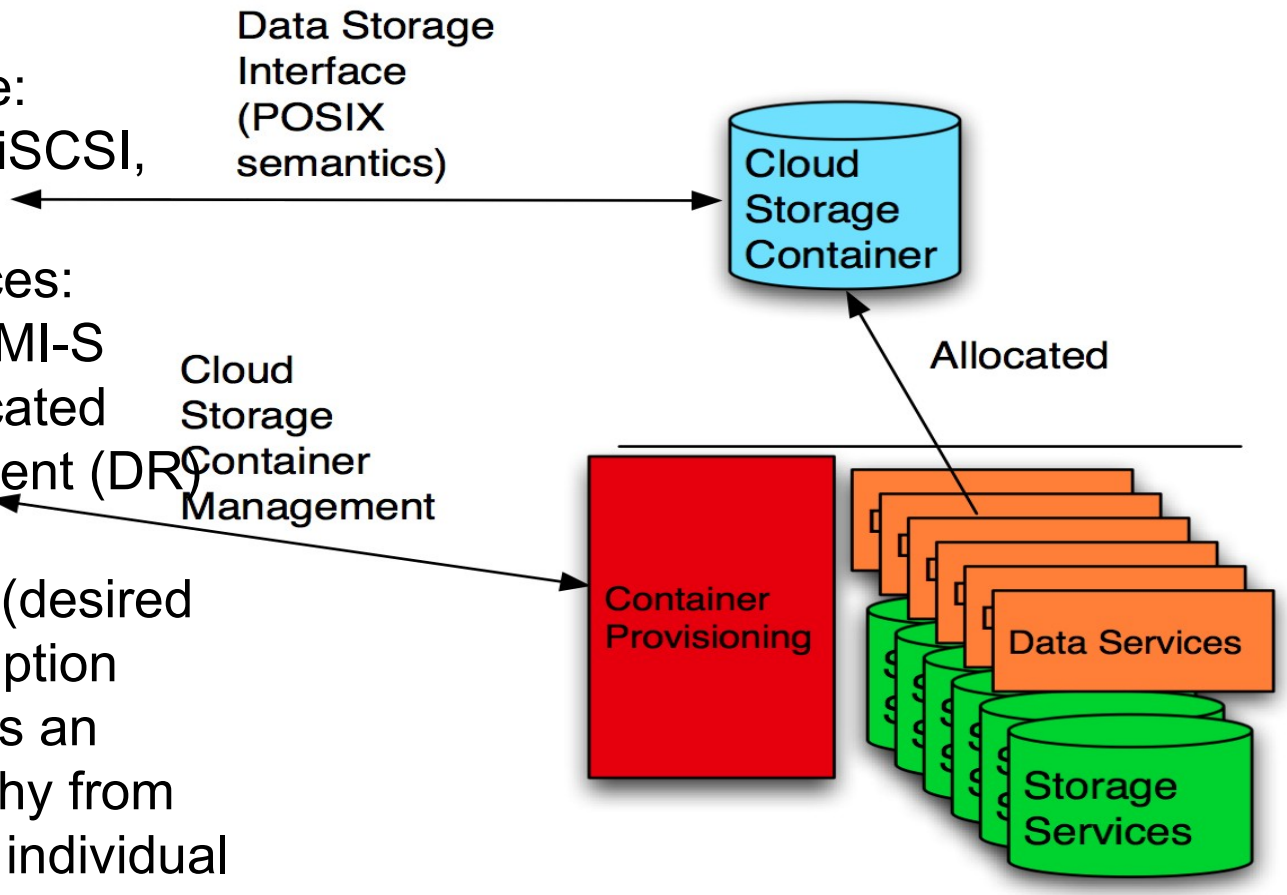
Leveraging the Storage Industry Resource Domain Model

All of these interfaces support some or all of this model. The key to retaining the simplicity of the cloud, however, is in the use of metadata to drive the underlying services so that users need not manage the services themselves.



Cloud Storage Container

- Cloud Storage is used as a volume/filesystem
- DSI Protocols include: WebDAV, NFS, CIFS, iSCSI, OSD
- Management interfaces: proprietary, Web UI, SMI-S
- Billing based on allocated space, Data Requirement (DR) parameters
- Resource guarantee (desired and required), consumption
- Configuration of DR is an object oriented hierarchy from containers on down to individual data elements



Container DSI

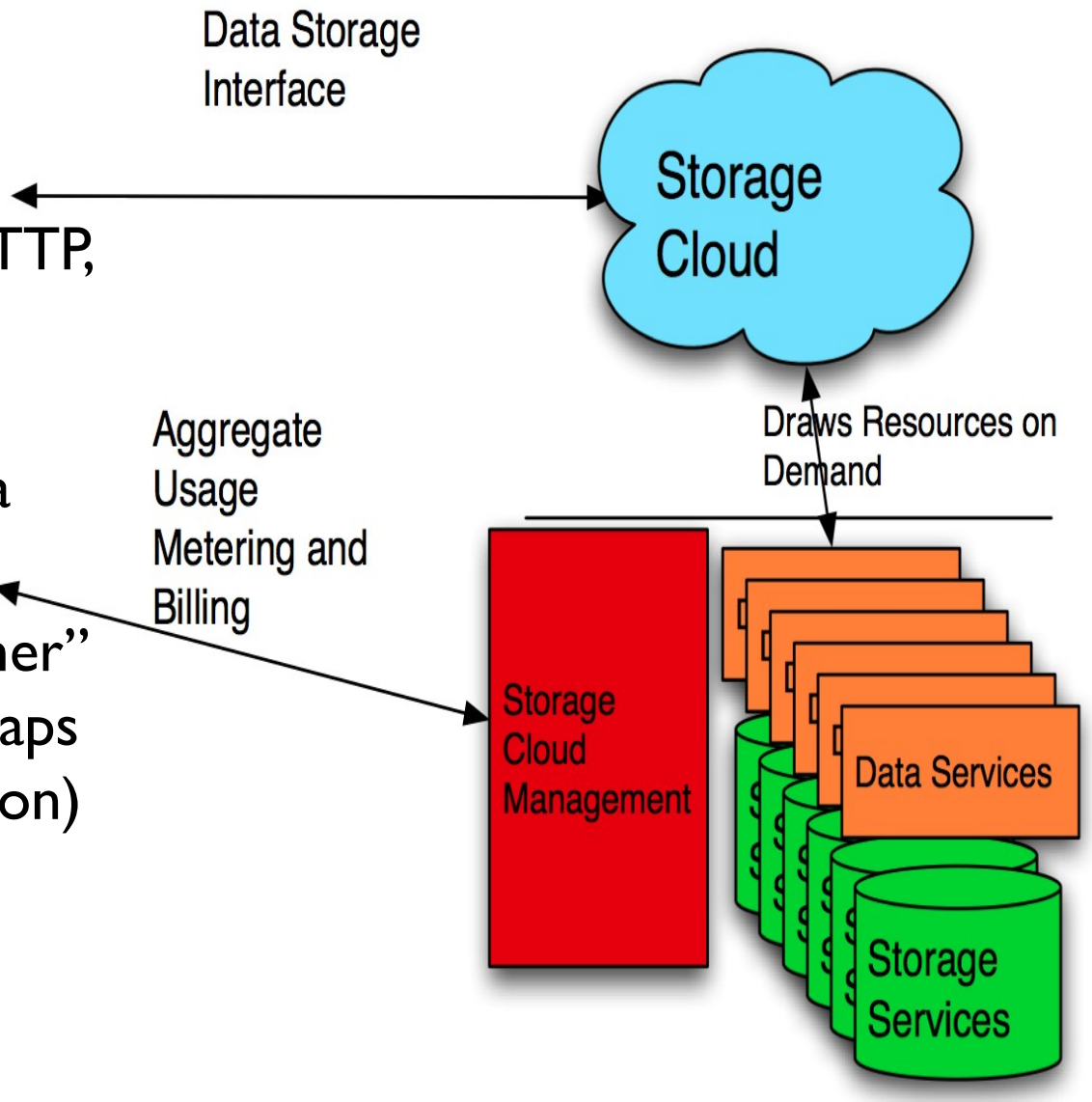
- Typically already standardized
- Needs support for metadata
- Need for standardization:
  - Standard Data System Metadata to be interpreted by the cloud as Data Requirements
- Broad categories of Data Requirements
  - Retention
  - Initial Security, Performance and Availability Requirements
    - Lifecycle – Defined Epochs with Requirements for each, Defined state transitions
    - Security Classification
    - Requirement may be expressed as minimum and desired
  - Location
    - Geography (political boundaries) – specific local requirements
    - Network Topology (bandwidth, latency to specific clients - Affinity to specific resources)
  - Budget
- History and Versioning metadata

Container Management

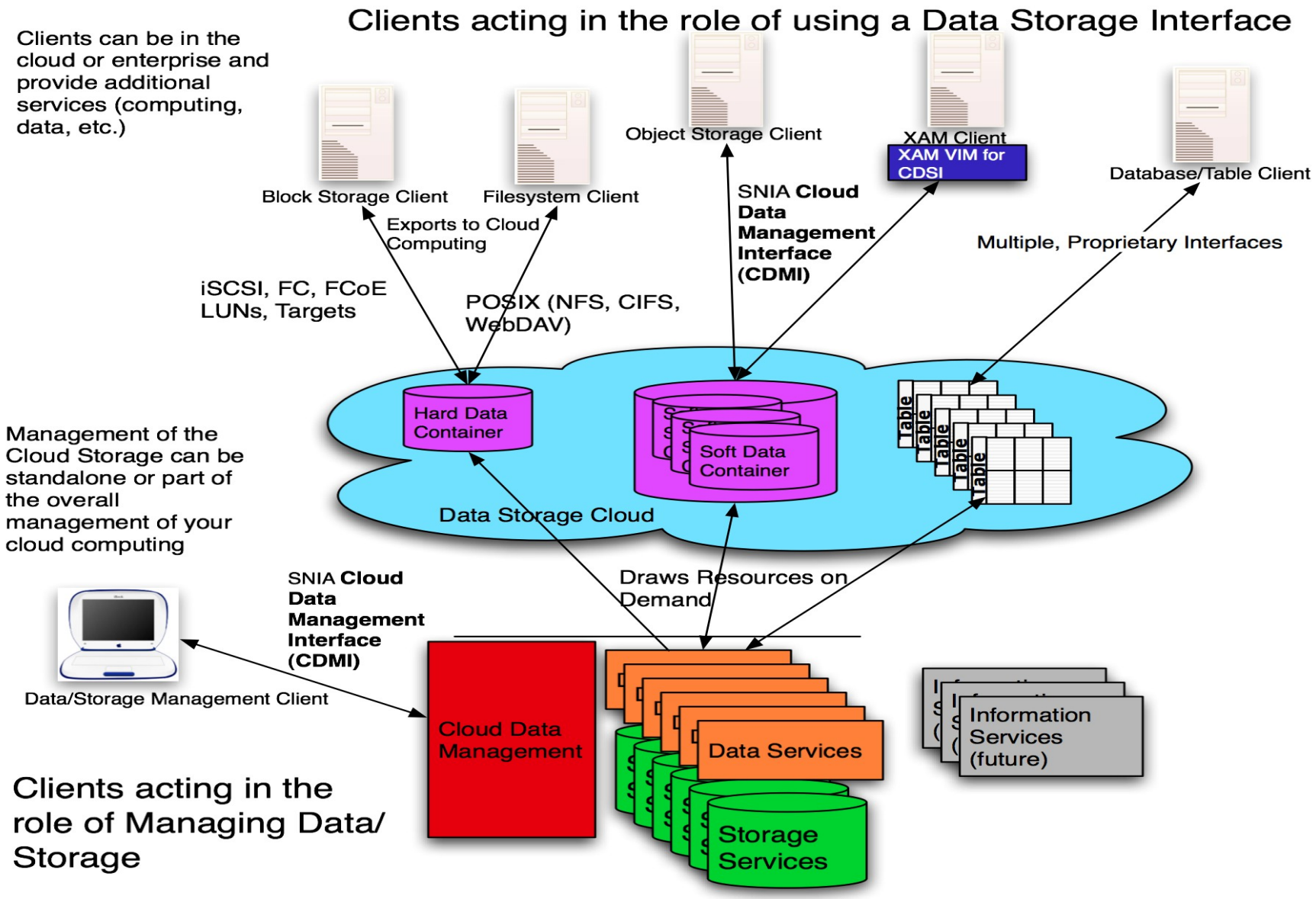
- Management Aspects
  - For iSCSI (for example) the "Container" could be equivalent to a Target, the LUNs would inherit (support a hierarchy of Containers) the Target's requirements, but could be overridden
- Allocation of Storage
  - Size (Initial. May be thin provisioned. May be grow-able)
  - Default Data Requirements (DR) expressed in the same schema as the file metadata
    - ACLs for the container as a whole as well
  - Requirements may be drawn from an existing set of templates
  - Price per GB (hint) of capacity allocated at this requirement
  - Price per GB (hint) of access (R vs. W) at this requirement
  - Timing aspects of the allocation – perhaps a "job" submittal
  - Characteristics and capabilities on each requirement (knob "stops")
    - Maximums (i.e. number of objects)
- Monitoring of Storage
  - Aggregate Consumed
  - Bandwidth Used – per Type
  - Reporting of "performance" against requirements
  - Based on connections to other elements
  - Container capability ranges for each requirement
  - Aggregate across containers

Storage Cloud

- Cloud storage is un-provisioned
- DSI interface: RESTful, HTTP, S3, XAM
- Management interfaces: proprietary, Web UI, Data System Metadata
- Soft definition of "container" and sub-containers (perhaps through a query expression)



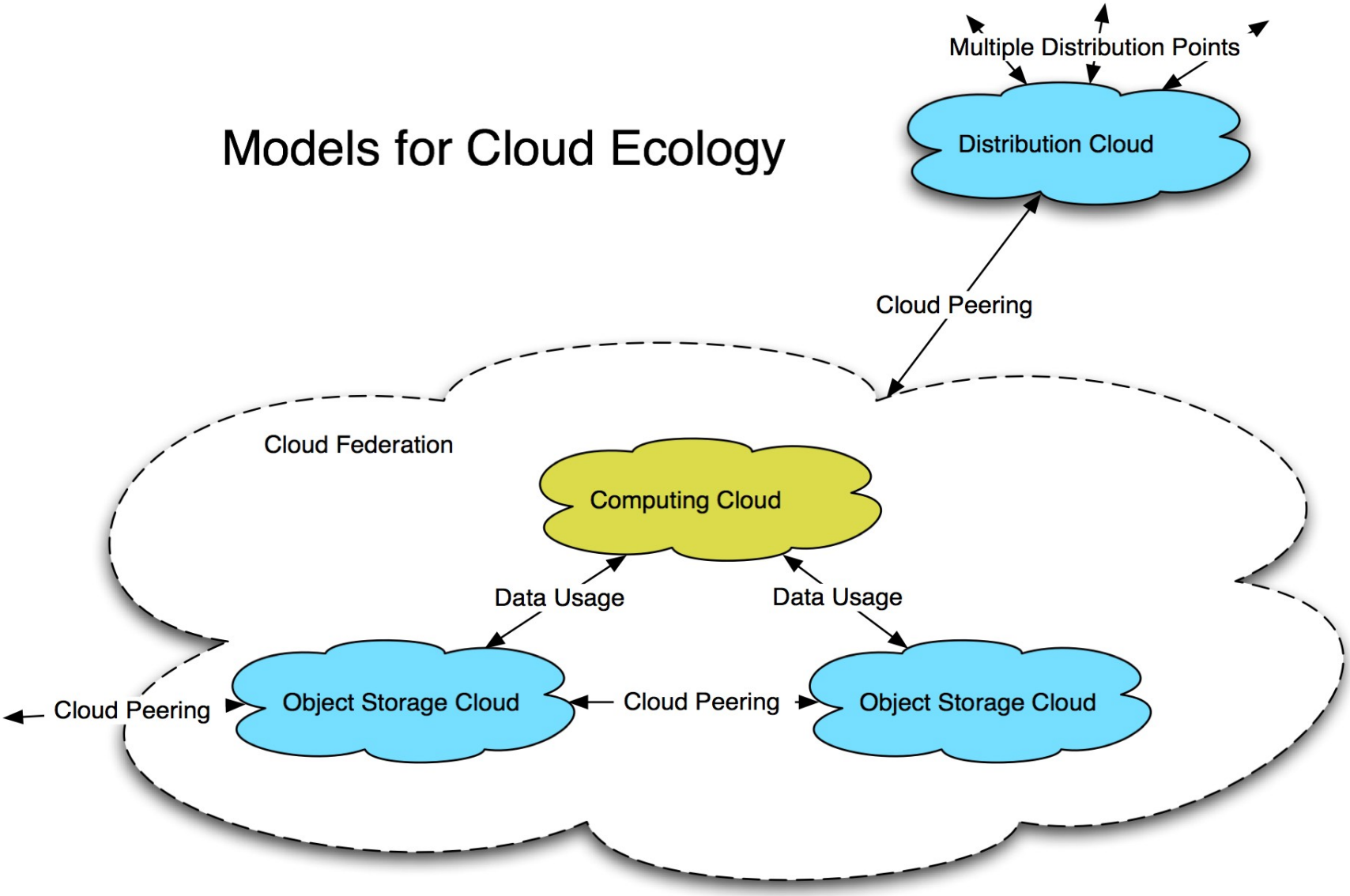
The Complete Picture



Cloud Data Management Interface

- Applicable to three types of Cloud Storage:
  - Cloud Storage for Cloud Computing
    - Whitepaper at snia.org/cloud – the management interface for the lifecycle of storage in a compute cloud
  - Public Storage Cloud
    - Both a Data Path for the Cloud and a Management Path for the Cloud Data
  - Private Cloud Storage
    - As well as hybrid clouds
    - An API for Storage Vendors selling into Cloud based solutions
- Semantics
  - Simple Containers and Data Objects with tagged Metadata
  - Data System Metadata expresses the data requirements
- Protocol
  - RESTful HTTP as "core" interface style
  - JSON (JavaScript Object Notation)– format of the representations are extensible

Cloud Peering

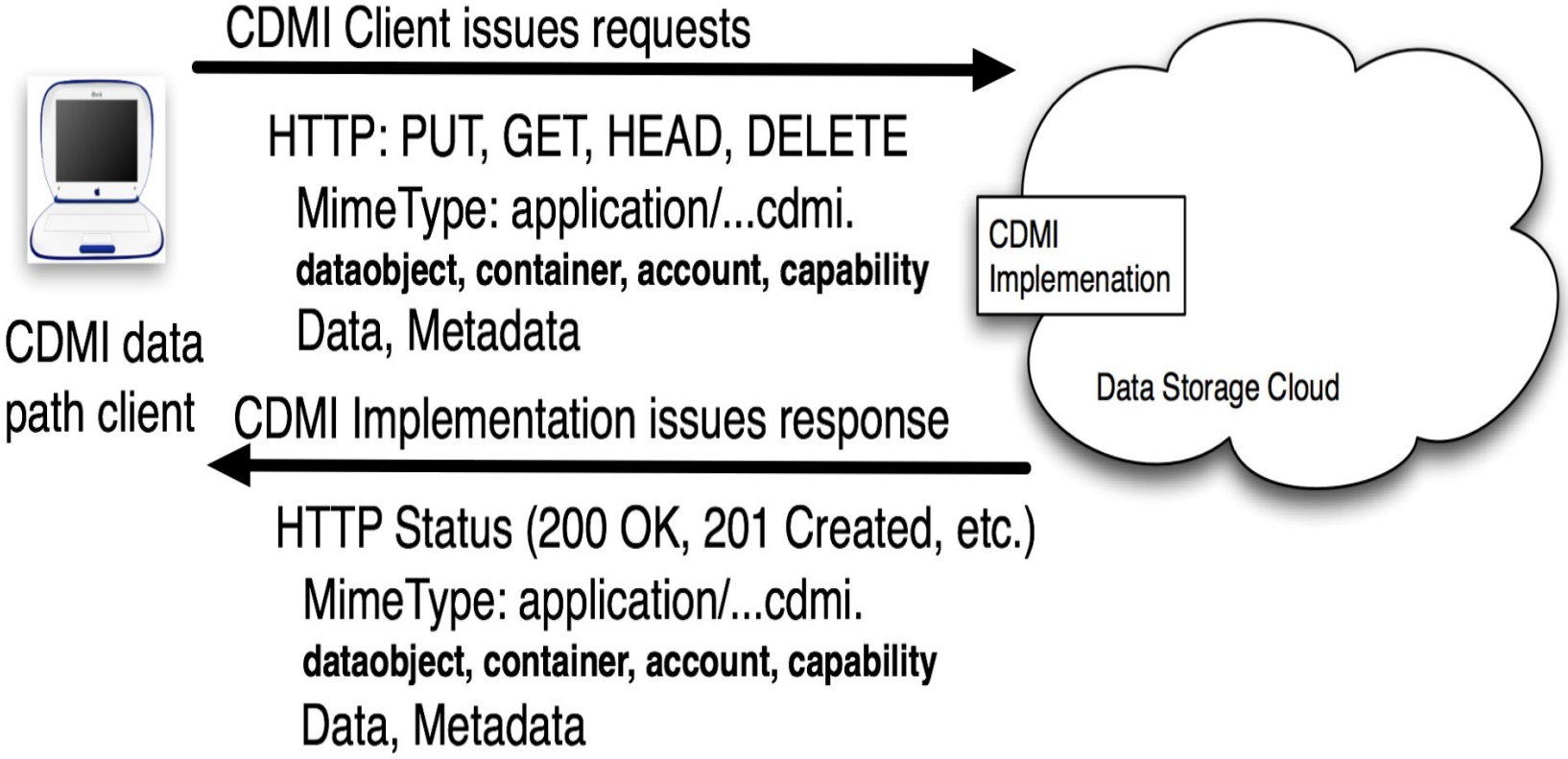


CDMI Specification

- First public draft posted today:
  - http://snia.org/cloud/CDMIspec.pdf
  - Version 0.8 – work in progress for public comment
- Entire specification is 90 pages
  - Intent is simplicity!
- Cloud Storage TWG is working towards a 1.0 release next year
  - Join us…

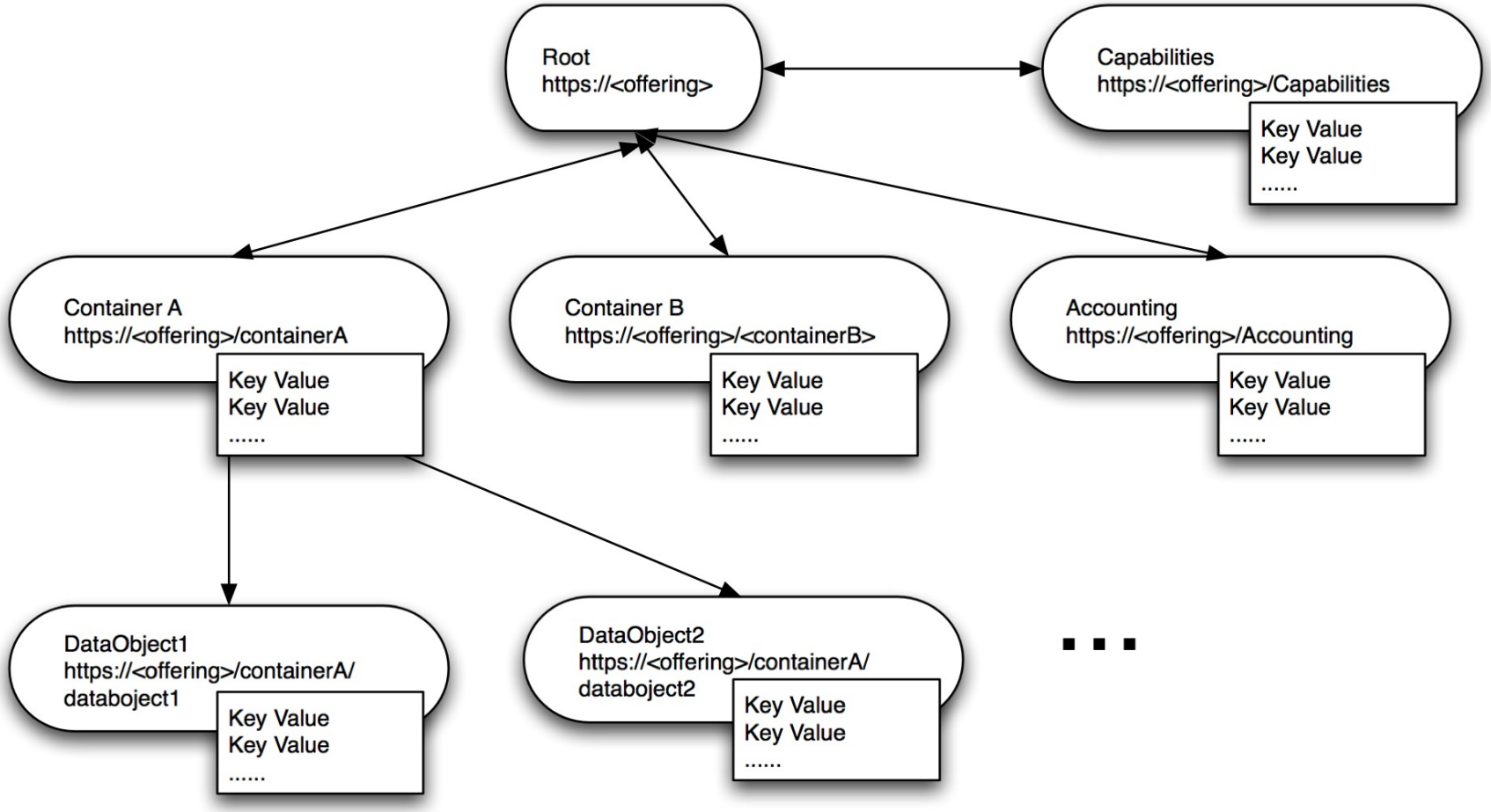
CDMI Overview

- CDMI Basic flow:



Model for the Interface

The resources which are accessed through the RESTful interface



CDMI Overview

- Chapter 5 – executive overview, based on earlier reference model
- Chapter 6 – some quick examples to get you started on coding
- Chapter 7 – the actual interface
  - 7.3 – Data Objects, 7.4 – Containers, 7.5 – Accounts, 7.6 – Queues, 7.7 – Capabilities, 7.8 – Import/Export Serialization
- Chapter 8 – the metadata
- Appendices - Implementation Compliance, Clients

CDMI Data Objects

- CDMI provides a RESTful HTTP API to allow management access for a wide range of stored data in cloud environments, including:
  - Files
    - http://cloud.example.com/files/notes.txt
  - Block Devices
    - http://cloud.example.com/luns/iscsi/bootimage
  - Object Stores
    - http://cloud.example.com/0x239F0930D3294FA8
  - Database Tables
    - http://cloud.example.com/database/table

CDMI Data Objects

- Stored data can be accessed using native protocols:
  - HTTP, CIFS, NFS, iSCSI, SQL, etc.
- Stored data can also be accessed using CDMI as a Data Path in a standardized manner. This facilitates:
  - Cloud-to-cloud migration
  - Cloud federation
  - Cloud backup
  - Cloud virus scanning
  - Cloud search
  - And more.
- Desired cloud storage characteristics can be associated with stored data:
  - Replication, Compression, Placement, Retention, QoS, etc.

Data Object Example

PUT to the container URI the data object name and contents

```
PUT: /MyContainer/MyDataObject.txt
Host: cloud.example.com
Accept: application/vnd.org.snia.cdmi.dataobject+json
Content-Type: application/vnd.org.snia.cdmi.dataobject+json
X-CDMI-Specification-Version: 1.0
{
   "mimemype" : "application/txt",
   "metadata" : [ ],
   "value" : "This is the Contents",
 }
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/vnd.org.snia.cdmi.dataobject+json
X-CDMI-Specification-Version: 1.0
{
   "objectURI" : "/MyContainer/MyDataObject",
   "objectID" : "AAAAFAAo7EFMb3JlbSBpcHN1bSBkb2xvciBzaXQgYW1ldCBhbWV0Lg==",
   "parentURI" : "/MyContainer",
   "accountURI" : "/cdmi_accounts/MyAccount",
```

Get the Data

GET from the data object URI

```
GET: /MyContainer/MyDataObject.txt
Host: cloud.example.com
```

The response looks like:

```
200 OK
Content-Type: application/txt
This is the Contents
```

Getting the Metadata

HEAD from the data object URI

```
HEAD: /MyContainer/MyDataObject.txt
Host: cloud.example.com
Accept: application/vnd.org.snia.cdmi.dataobject+json
Content-Type: application/vnd.org.snia.cdmi.dataobject+json
X-CDMI-Specification-Version: 1.0
```

The response looks like:

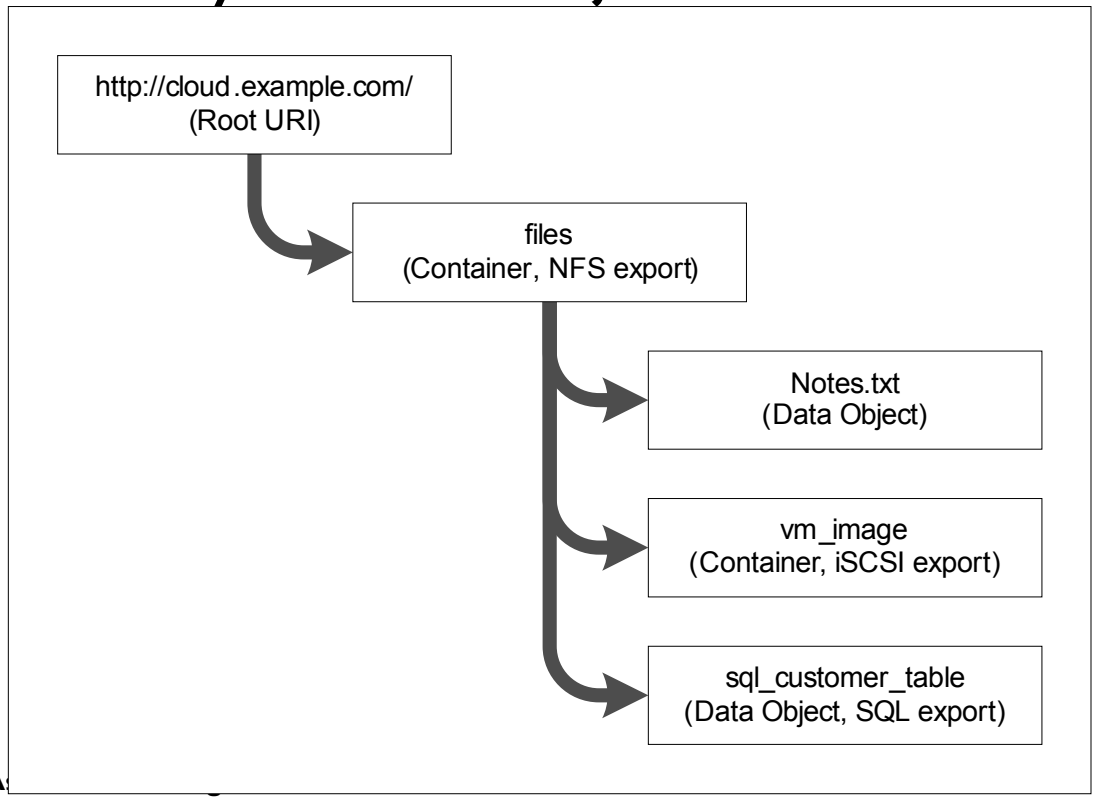
```
HTTP/1.1 200 OK
Content-Type: application/vnd.org.snia.cdmi.dataobject+json
X-CDMI-Specification-Version: 1.0
{
   "objectURI" : "/MyContainer/MyDataObject.txt",
   "objectID" : "AAAAFAAo7EFMb3JlbSBpcHN1bSBkb2xvciBzaXQgYW1dCBhbWV0Lg=="
   "parentURI" : "/MyContainer",
   "accountURI" : "/cdmi_accounts/MyAccount",
   "capabilitiesURI" : "/cdmi_capabilities/dataobject",
   "percentageComplete" : "Complete",
   "mimetype" : "application/txt",
   "metadata" : [ ],
}
```

CDMI Containers

- Containers are an abstract place to put "data"
- A Container can contain Data Objects and/or can be "exported" as a block based volume or filesystem
- Containers have data system metadata that specifies the requirements for the data contained in them (rather than a explicit "configuration")
- Data system metadata in CDMI is inherited from parent containers to child containers and data objects.
- Containers can also be serialized and stored into a data object.

CDMI Containers

- CDMI uses hierarchies to enable simplified management
  - Data System Metadata can be specified on a container, and inherited by all child objects.



Create a Container

o the URI the container name and other metadata

```
UT: /MyContainer
ost: cloud.example.com
ccept: application/vnd.org.snia.cdmi.container+json
ontent-Type: application/vnd.org.snia.cdmi.container+json
-CDMI-Specification-Version: 1.0

  "metadata" : [ ],
```

esponse looks like:

```
TTP/1.1 201 Created
ontent-Type: application/vnd.org.snia.cdmi.container+json
-CDMI-Specification-Version: 1.0

  "objectURI" : "/MyContainer",
bjectID" : "AAAAFAAo7EFMb3JlbSBpcHN1bSBkb2xvciBzaXQgYW1dCBhbWV0Lg==",
arentURI" : "/",
  "accountURI" : "/cdmi_accounts/MyAccount",
  "capabilitiesURI" : "/cdmi_capabilities/container",
  "percentageComplete" : "Complete",
  "metadata" : [ ],
  "children" : [ ],
```

Exporting Containers

- The export of a container, via data path protocols other than CDMI, is done by creating or updating a container and supplying one or more export structures, one for each such protocol.
  - The elements of the export structure include:
  - The protocol being used
  - The identify of the container as standardized by the protocol
  - The list of who can access that container via that protocol, identified as standardized by that protocol (may leverage the CDMI accounting for this)
- CDMI standardizes several export structures for various protocols. Export structures can also be defined for proprietary and vendor extensions of protocols.

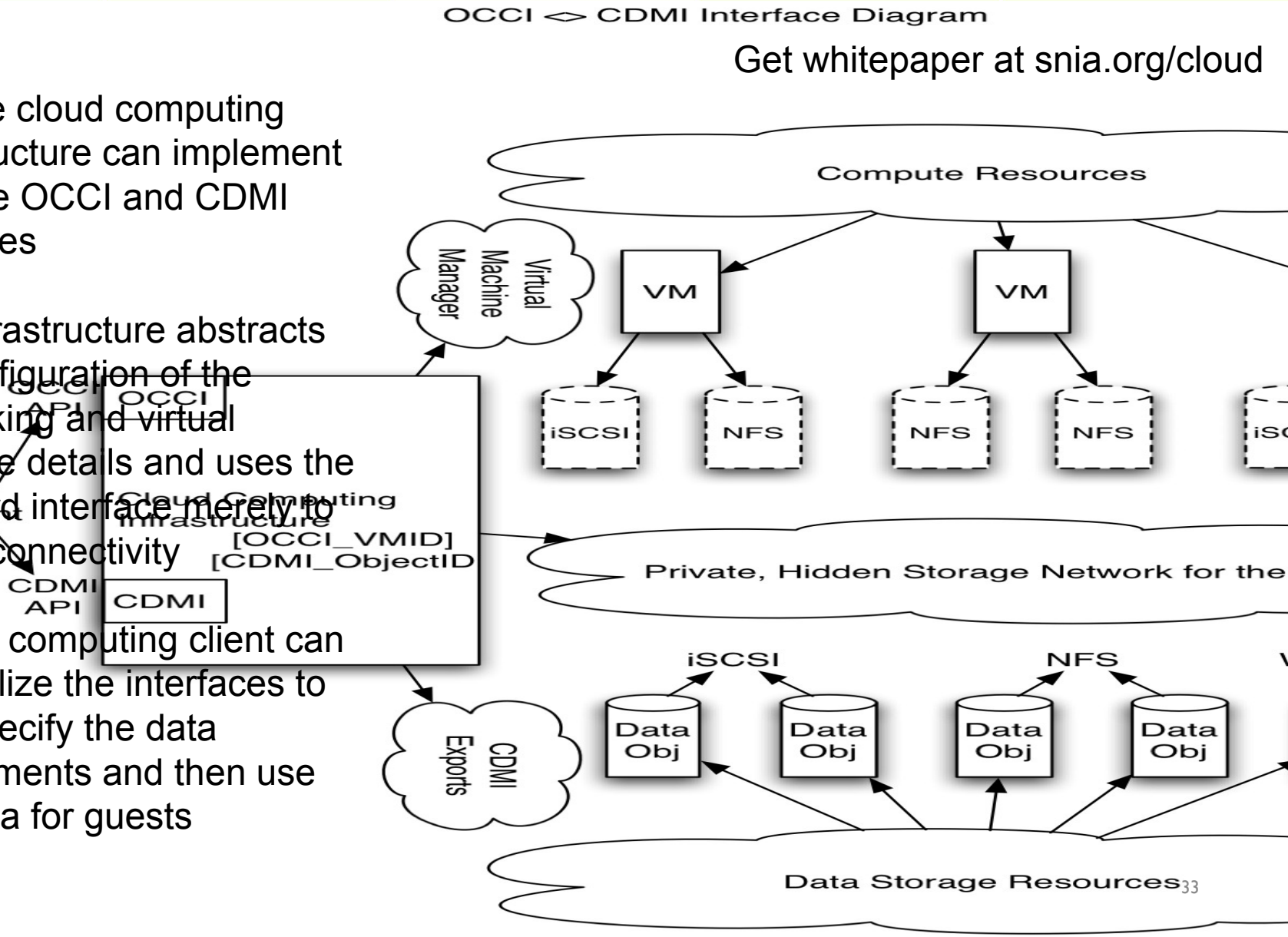
CDMI in Cloud Computing

Get whitepaper at snia.org/cloud

A single cloud computing infrastructure can implement both the OCCI and CDMI interfaces

The infrastructure abstracts the configuration of the networking and virtual machine details and uses the standard interface merely to define connectivity

A cloud computing client can then utilize the interfaces to both specify the data requirements and then use that data for guests



Exporting OCCI Containers

- CDMI provides a type of export that contains information obtained via the OCCI interface. A client of both interfaces would perform the following operations as an example:
  - The client creates a CDMI container through the CDMI interface and exports it as an OCCI export type. The CDMI container objectID is returned as a result.
  - The client then creates a virtual machine through the OCCI interface and attaches a storage volume of type CDMI using the objectID. The OCCI virtual machine ID is returned as a result.
  - The client then updates the CDMI container object export information with the OCCI virtual machine ID to allow the virtual machine access to the container.
  - The client then starts the virtual machine through the OCCI interface.

CDMI Queues

- Queues are a special class of data object that have first-in-first-out style access when storing and retrieving data. If supported by the cloud storage system, cloud clients create the queue objects by using the same mechanism used to create data objects.
- Every queue object has a parent object, from which the queue object inherits data system metadata. For example, the "receipts.queue" queue object stored at "http://cloud.example.com/finance/recipts.queue" would inherit data system metadata from its parent container, "finance".

CDMI Accounts

- Accounts provide a place to manage who can access cloud content, discover what operations have been performed on cloud content, and obtain billing and reporting information related to cloud content.
- Account Summaries provide summary information about the account
- Account Membership provides access to credential mapping and permissions
- Account Notifications provide details on events occurring in the account
- Account Query provides mechanisms for discovering content within an account

CDMI Accounts - Membership

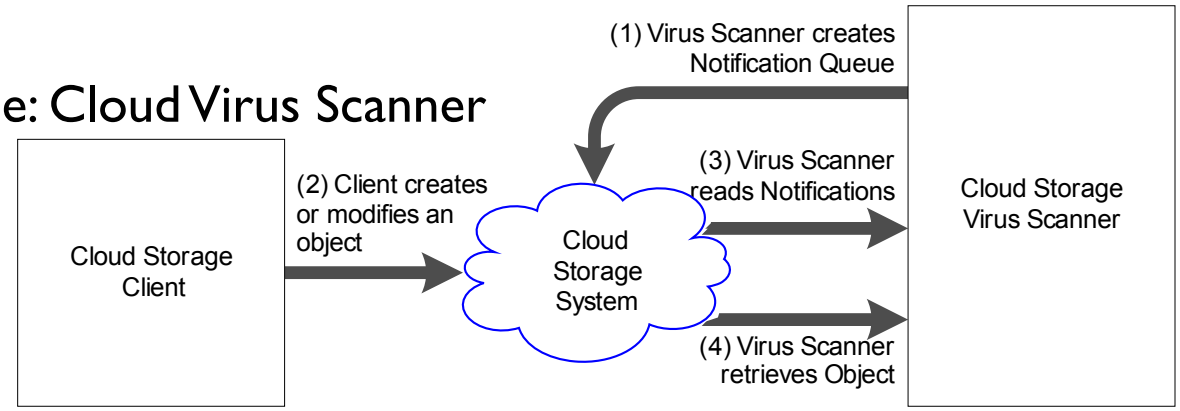
- Account Membership allows access to user/account/privilege information for the account
  - Add, disable and remove users (enrollment)
  - Specify user credentials
  - Allow users to change their credentials
  - Define delegation of credentials (e.g., external LDAP, AD, etc.)

- Allows users to define multi-party sharing relationships
  - Essential for peering/federation relationships
  - User can create an account for a third-party virus scanner, for example, and grant it only the privileges needed for this role

CDMI Accounts - Notification

- Notification queues allow a cloud storage system to tell a client about changes of state
  - Object created, modified, deleted, etc…
  - Each change results in an event that is enqueued into a queue
  - Clients can create queues, and specify what events they are interested in, and for each event, what information should be included

- Example: Cloud Virus Scanner



CDMI Capabilities

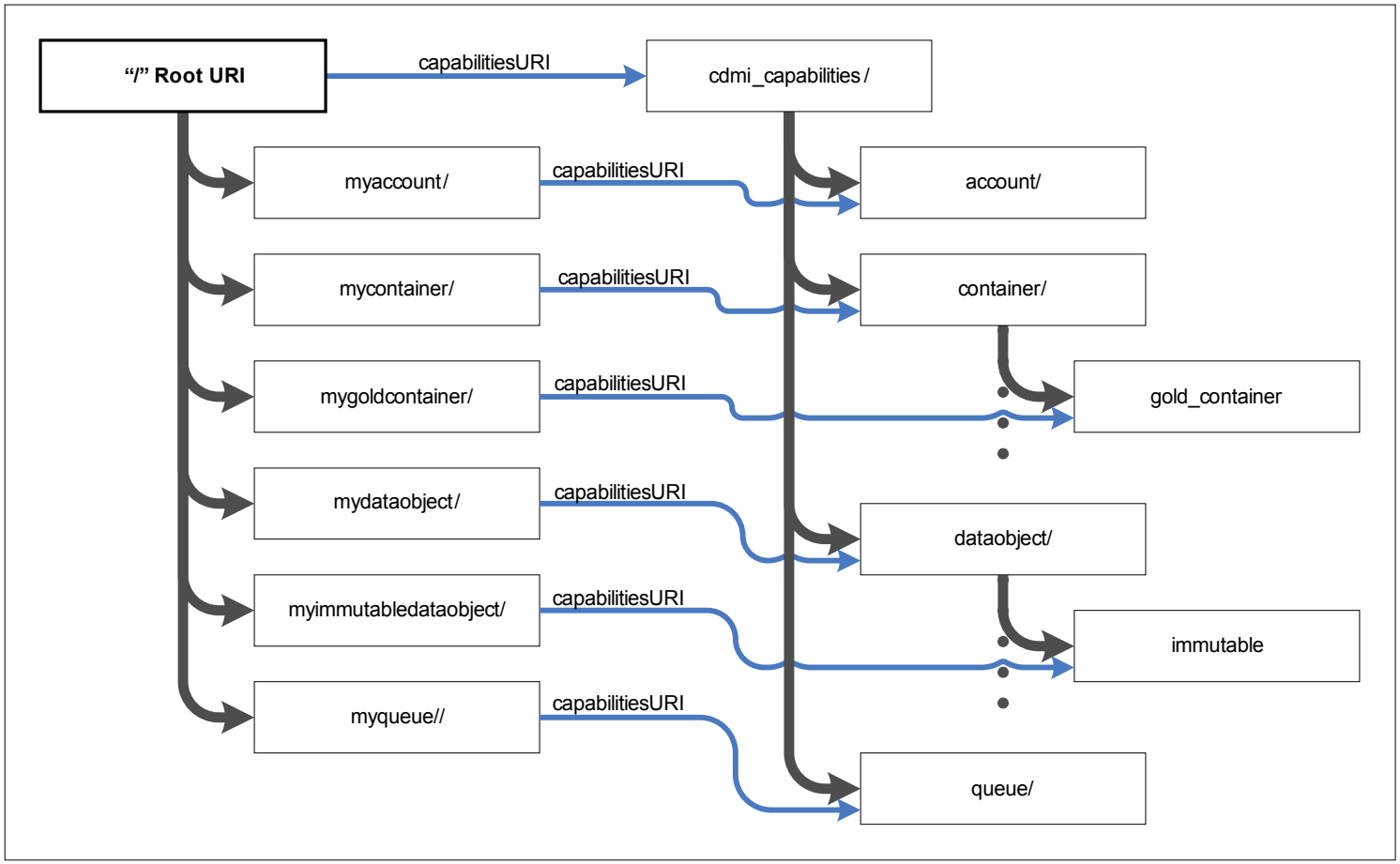
- Capabilities define what storage operations a CDMI provider is capable of providing.
- Contrast with permissions, which define what storage operations a CDMI provider will permit a user to perform.
- Capabilities are static for a given cloud storage system, but different sets of capabilities may be present for different URIs.

CDMI Capabilities

- CDMI Clients typically use capabilities for the following purposes:
  - Initial introspection of the capabilities of a cloud storage provider
  - Introspection of the capabilities of a CDMI object at a given URI
- In order to facilitate the first use case, every CDMI system has a tree structure of capabilities objects that is referenced from the root.
- In order to facilitate the second use case, every CDMI object has a capabilitiesURI field that points to the capabilities object corresponding to the capabilities of the specific CDMI object.

CDMI Capabilities

- The below diagram illustrates how CDMI objects relate to capabilities:



CDMI Capabilities

- The following aspects of capabilities are up to the cloud storage implementer:
  - The layout of the capabilities tree
  - The capabilities in each capabilities object
  - The mapping between created CDMI objects and capabilities objects
- The following aspects of capabilities are mandatory:
  - A root capabilities object that describes system-wide capabilities
  - A capabilities URI link from every object to a corresponding capabilities object
  - A capabilities URI link from the root URI to the root capabilities object

CDMI Data System Metadata

Metadata Name	Description	Source	Permissions
cdmi_size	The number of bytes stored in the data object	Server	Read-Only
cdmi_actualsize	The number of bytes consumed by storing the data item. This number may be less or more than the size of the data stored by the client due to overhead, compression, de-duplication, etc.	Server	Read-Only
cdmi_ctime	The time when the data object was created in ISO 8601 format	Server	Read-Only
cdmi_atime	The time when the data object was last accessed in ISO 8601 format	Server	Read-Only
cdmi_mtime	The time when the data object was last modified in ISO 8601 format	Server	Read-Only
cdmi_acount	The number of times that the object has been accessed since it was originally created. Accesses include all reads, writes, and lists.	Server	Read-Only
cdmi_mcount	The number of times that the object has been modified since it was originally created. Modifications include all value and metadata changes. Modifications to metadata resulting from reads (such as updates to atime) do not count as a modification.	Server	Read-Only
ACL metadata	Standard ACL metadata. (Based on NFSv4?) If not specified when the object is created, this metadata will be filled in by the system.	Client	Read/Write

Query

- Query queues allow CDMI clients to efficiently discover what content matches a given set of metadata query criteria or full-content search criteria. Clients PUT a query specification into the queue, including a URI to a query results data object to be created. The cloud storage query engine will then initiate a query, creating a new query results object in the designated location. As query results are found, they are added to the query results queue object, and when the query is complete, the state of the query results queue object is changed to indicate that the query has completed.
- External query engines can be easily integrated into a cloud system by using CDMI to read the contents of the query queue, then performing the queries on behalf of the user who posted the query.

Questions

- Thank you!