Using SMI-S with the Cloud Data Management Interface

Scott Baker
September 21th, 2010
Scope

- CDMI is an interoperable & vendor neutral interface for cloud offerings consisting of:
  - Data path
  - Management / control path
Scope - SMI-S

- An interoperable and vendor-neutral interface for management of SAN systems

- SMI-S is an ideal solution for providing the control path implementation for CDMI implementations that are used to export legacy protocols such as iSCSI.
Agenda

- CDMI overview
- SMI-S overview
- A simple example CDMI hierarchy to SMI-S model mapping
- CDMI Reference Implementation overview
- Reference Implementation adapted with SMI-S middle layer
- Code
- Demonstration
CDMI Overview

- Developed by the SNIA Cloud TWG

- CDMI v1.0 an approved SNIA Technical Position
CDMI Overview

- REST based interface.
- Stands for Representational State Transfer
- Attributes:
  - Resources are URIs
  - Operations constrained to a few HTTP basic verbs (GET, PUT, DELETE)
    - State is in message body representation
- Provides for Create, Delete, Update, & Delete of data (CRUD)
- Model is a hierarchy of containers with data objects as leaves.
- Data objects are files, pictures, documents, etc.
- Capabilities (QOS) modeled as a hierarchy of pseudo-containers
SMI-S - Model

- Web-based Enterprise Management (WBEM)
- Domain model expressed using the Distributed Management Task Force (DMTF)'s Common Information Model
- Management interface provides for access to model and invocation of services using the DMTF's CIM-XML Operations over HTTP
  - Get class instances
  - Traverse (walk) associations
SMI-S – Block Storage Model

- Storage is modeled as a hierarchy of Storage Pools
- Block-based exported logical units are modeled as Storage Volumes
- Capabilities are modeled as Capabilities or Settings
- Operations modeled as services
SMI-S – Pool Model

CIM_StorageVolume

CIM_AllocatedFromStoragePool

CIM_StoragePool

CIM_StorageCapabilities
  DataRedundancy = 2

CIM_AllocatedFromStoragePool

CIM_StoragePool

CIM_StorageCapabilities
CDMI to SMI-S Mapping

- Container → StoragePool
- Dataobject → StorageVolume
- Capability → StorageSetting or StorageCapability

- Simple example – not exhaustive by any means
CDMI to SMI-S mapping

root/pool_container1/pool_container2/volume_dataobject1: CIM_StorageVolume

CIM_AllocatedFromStoragePool

root/pool_container1/pool_container2: CIM_StoragePool

CIM_AllocatedFromStoragePool

root/pool_container1: CIM_StoragePool

root/cdmi_capabilities/dataobject: CIM_StorageSetting

root/cdmi_capabilities/container: CIM_StorageCapabilities

DataRedundancy = 2

root/cdmi_capabilities/container/default: CIM_StorageCapabilities
Design provides for different implementations
Currently implemented using a local filesystem
Alternative for a control path implementation for this presentation is an SMI interface.
Reference Implementation Architecture

CDMI Reference Implementation Architectural Diagram
Green: SNIA Developed Code
Blue: 3rd Party Code

Date: July 20, 2010
RI Architecture

* File System Naming:
  Container Objects = Folders named with the container name
  Data Objects = Files named with the object name, if one was given, else the Object ID
  Metadata = Files named with the same name as the corresponding object with an additional "." in front

Examples:
  Container: /mnt/cdmi server/MyContainer
  Container Metadata: /mnt/cdmi server/.MyContainer

  Data Object's Data: /mnt/cdmi server/MyContainer/MyDataObject.txt
  Data Object's Metadata: /mnt/cdmi server/MyContainer/./MyDataObject.txt

  Data Object's Data: /mnt/cdmi server/MyContainer/0000706D0010B84FAD185C425D8B537E
  Data Object's Metadata: /mnt/cdmi server/MyContainer/./0000706D0010B84FAD185C425D8B537E
Reference Implementation Design

- Uses the JAX-RS API Standard
- Annotations on Classes and Methods provides for:
  - URI Matching and Routing to methods
  - Parameter injection (URI segments)
  - Content-type handling (e.g.) container vs dataobject)
- Java
- Uses Apache CXF framework implementation of JAX-RS
Framework support (web.xml). Links to Apache CXF framework servlet and specifies url pattern to route to CXF servlet.

- `<servlet>`
  - `<servlet_name>`

- `<servlet-mapping>`
  - `<url-pattern>`
RI Design

- CXF Framework support -ApplicationContext.xml
  - Identification and location of REST resource service classes including point of entry
  - Identification of injection points and current injection values
Design – Build Environment

- Maven
- Standardized directory structure
- pom.xml file specifies internet repositories for satisfying library dependencies
- Netbeans used by CDMI RI developers
Design – Module Breakdown

- Model classes
- REST resource classes
- DAO ( Data Access Object ) pattern
  - Interfaces
  - Implementation classes
Design – Model Classes

- Contains serialization/deserialization <> JSON
  - Container
  - Data Object
  - Capability
  - Domain
Design – Resource Classes

- PathResource class
  - cdmi_server/{mycontainer}
- ObjectIdResource class
  - cdmi_server/cdmi_objectid/{object id of mycontainer}
- CapabilityResource class
  - cdmi_server/cdmi_capabilities
Design – DAO interfaces & impls

- ContainerDAO
- DataObjectDAO
- CapabilityDAO
- DomainDAO
SMI-S layer adaptation

- Replaces Filesystem DAO Impl classes with a WBEM Client layer
- WBEM Services implementation of JSR48
SMI-S layer adaptation

- Container(s) under cdmi_root are SMI “Primordial Pools”
  - CIM_StoragePool.Primordial=true

- Containers in hierarchy beyond primordial accessed by traversing “AllocatedFromStoragePool” association class instances

- DataObjects/Volumes accessed from non-primordial pools by “AllocatedFromStoragePool”
Demonstration

- OSX HttpClient
- CDMI Reference Implementation
- Local CIMOM with MOF instances simulating SMI-S Block Server.
- Get container/pools, dataobjects/volumes
Demonstration –
Get root container
Questions?

sjbaker@mindspring.com