SNIA™
Cloud Storage Initiative

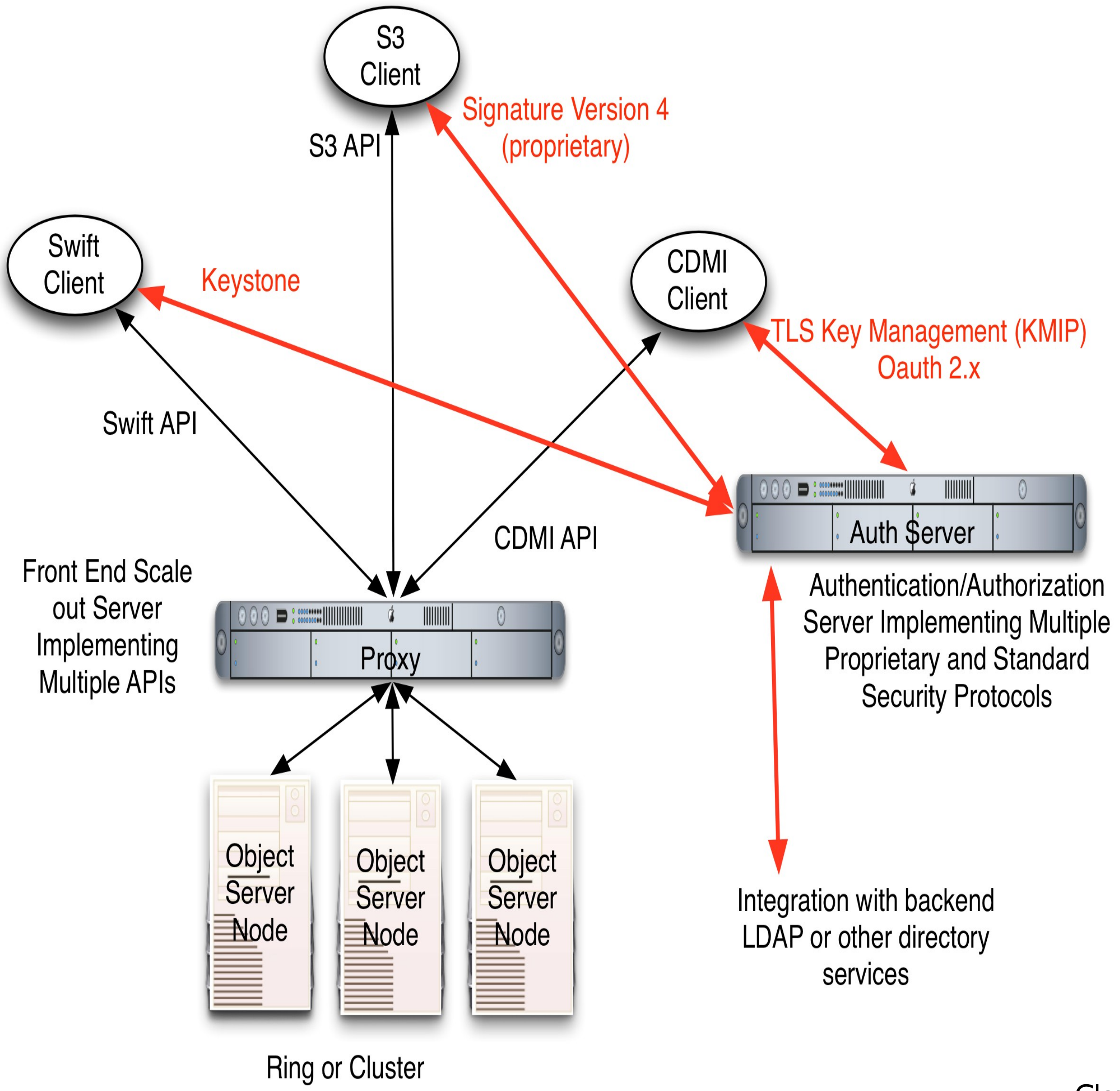
Implementing a Standard API

Mark Carlson, Toshiba
co-Chair SNIA Cloud Storage TWG

What experience have you had?

- When customers ask for support of a given API, can a vendor survive if they ignore these requests?
- A strategy many vendors are taking is to support multiple APIs with a single implementation.
    - Besides the Swift API, many support the S3 defacto and CDMI standard APIs in their implementation.
- What is needed for these APIs to co-exist in an implementation?
- Basic operations are nearly identical between APIs, but what about semantics that have multiple different expressions such as metadata?
- Best practices and tips to implementing multiple protocols in your cloud storage solution

What does this look like?


S3 Client
S3 API
Signature Version 4 (proprietary)
Swift Client
Keystone
CDMI Client
TLS Key Management (KMIP) Oauth 2.x
Swift API
CDMI API
Auth Server
Authentication/Authorization Server Implementing Multiple Proprietary and Standard Security Protocols
Front End Scale out Server Implementing Multiple APIs
Proxy
Object Server Node
Object Server Node
Object Server Node
Integration with backend LDAP or other directory services
Ring or Cluster

Breakdown

Cloud Storage Initiative

- Storage Operations

•CRUD – All pretty much determined by HTTP standard (common code)

•Headers are API unique however (handle in API specific modules)

- Security Operations

•Client communication with Auth Server (API unique)

•Multiple separate services running in Auth Server

What resources are available

- Comparison of S3/Swift functions

•https://wiki.openstack.org/wiki/Swift/APIFeatureComparis

  - Recently updated

- Implementation of CDMI filter driver for Swift

•https://github.com/osaddon/cdmi

  - More slides later – being updated by TTU
  - Implementation of S3 filter driver for Swift
  - https://github.com/fujita/swift3
  - Also stagnant

Support

- Latest version of CDMI - **http://www.snia.org/sites/default/files/CDMI_Spec_v1.1.1.pdf**

•Baseline operations (mostly governed by RFC 2616) now documented in Clause 6 (pgs. 29-36)

•CDMI now uses content type to indicate CDMI-style operations (as opposed to X-CDMI-Specification-Version)

•Spec text that explicitly forbid (in 1.0) functionality required for S3/Swift integration has been removed from the spec.

•HTTP Basic/Digest authentication is no longer mandatory. CDMI implementations can now use S3 or Swift authentication exclusively, if desired.

Discovery of Security Protocol Implementations

- CDMI 1.1 now includes a standard means of discovering what methods are available:
  - •cdmi_authentication_methods (Data System Metadata) ***12.1.3***
  - •If present, this capability contains a list of server-supported authentication methods that are supported by a domain. The following values for authentication method strings are defined:

• "anonymous"-Absence of authentication supported

• "basic"-HTTP basic authentication supported (RFC2617)

• "digest"-HTTP digest authentication supported (RFC2617)

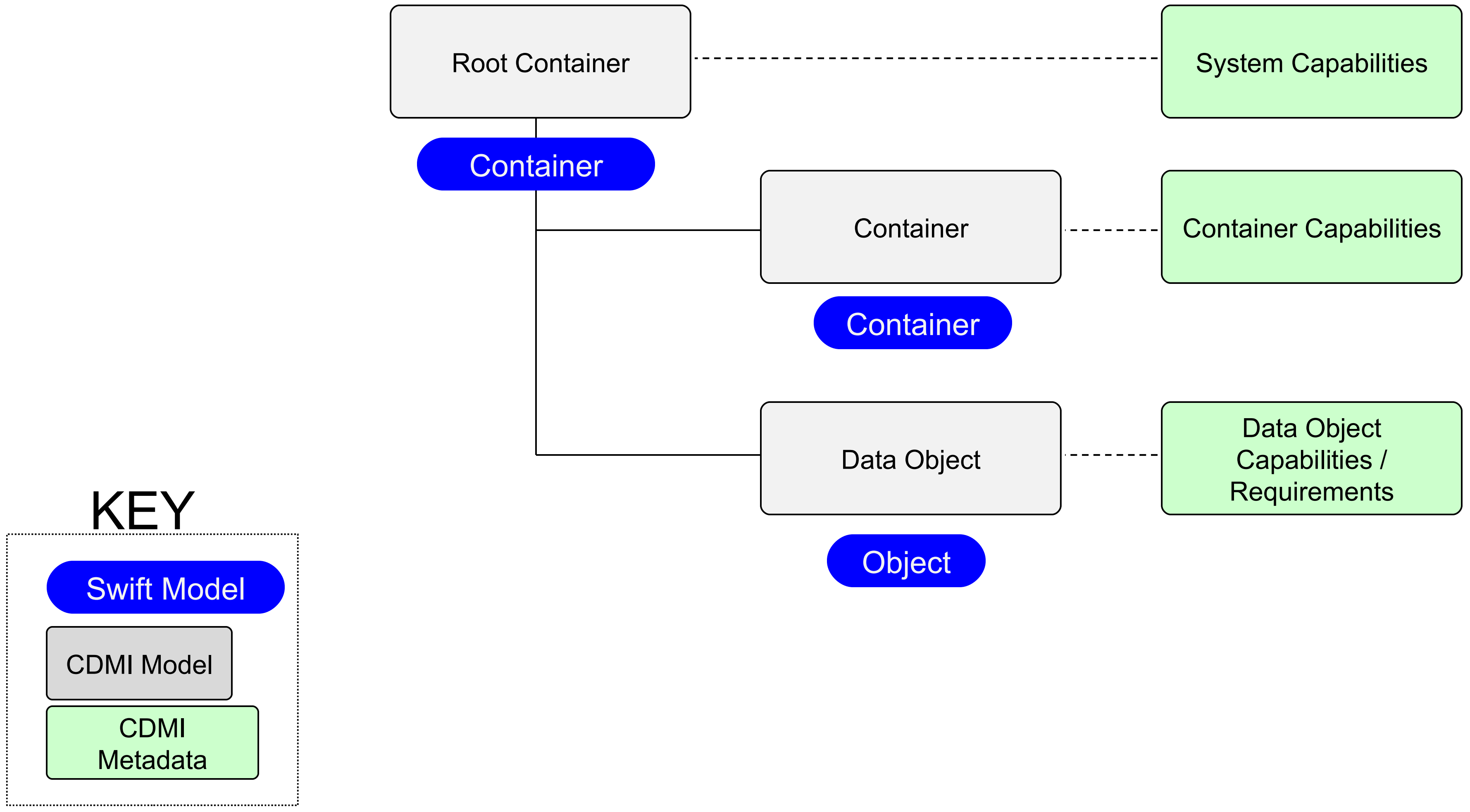
• "krb5"-Kerberos authentication supported, using the Kerberos Domain specified in the CDMI domain (RFC 4559)

• "x509"-certificate-based authentication via TLS (RFC5246)

Extending the standard security types

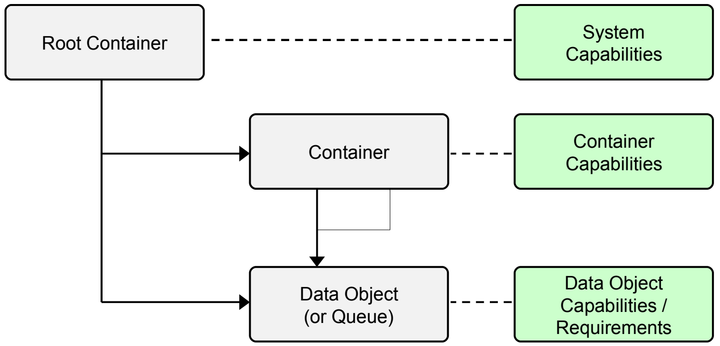
- The following values are examples of other widely used authentication methods that may be supported by a CDMI server:
  - "s3"-S3 API signed header authentication supported
  - "openstack"-OpenStack Identity API header authentication supported
  - Interoperability with these authentication methods are not defined by this international standard.
  - Servers may include other authentication methods not included in the above list. In these cases, it is up to the CDMI client and CDMI server (implementations themselves) to ensure interoperability.
  - When present, the cdmi_authentication_methods data system metadata shall be supported for all domains.

Swift and CDMI Models

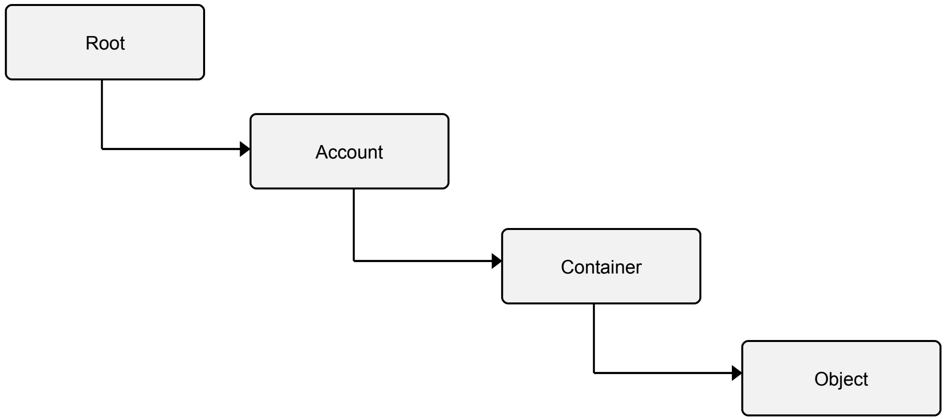


Swift and CDMI Models

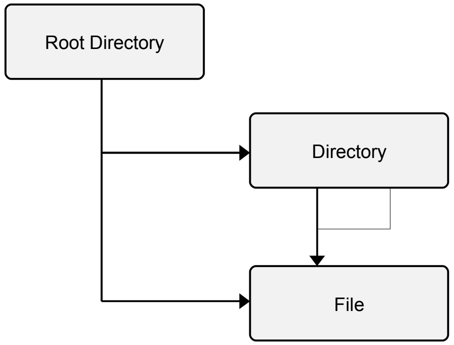
CDMI Object Model



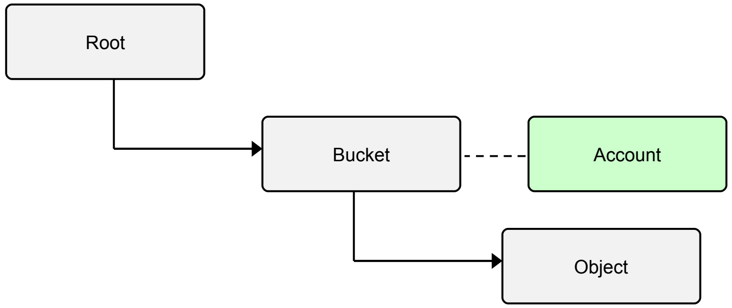
Swift Object Model



File System Model

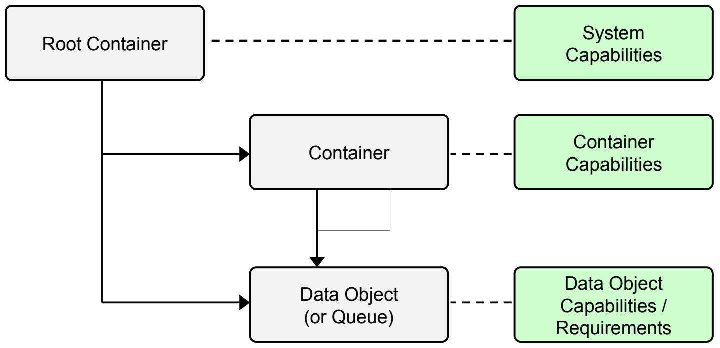


S3 Object Model

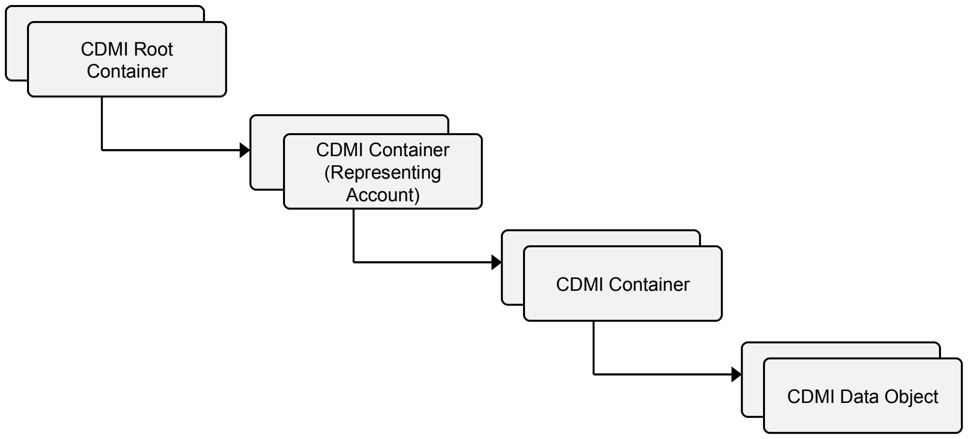


Swift and CDMI Models

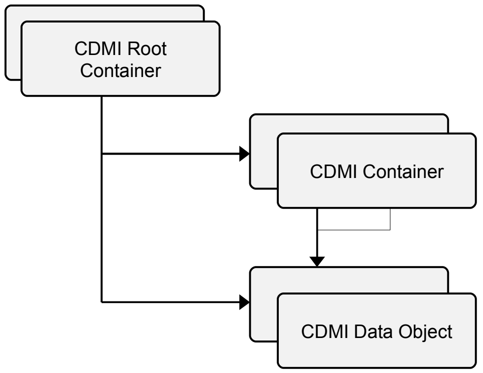
CDMI Object Model



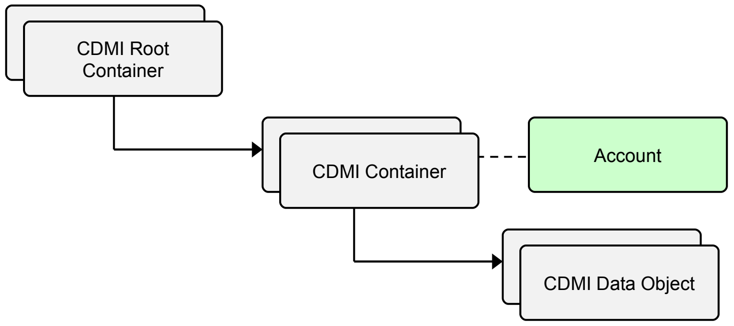
Overlay for Swift Object Model



Overlay for File System Model

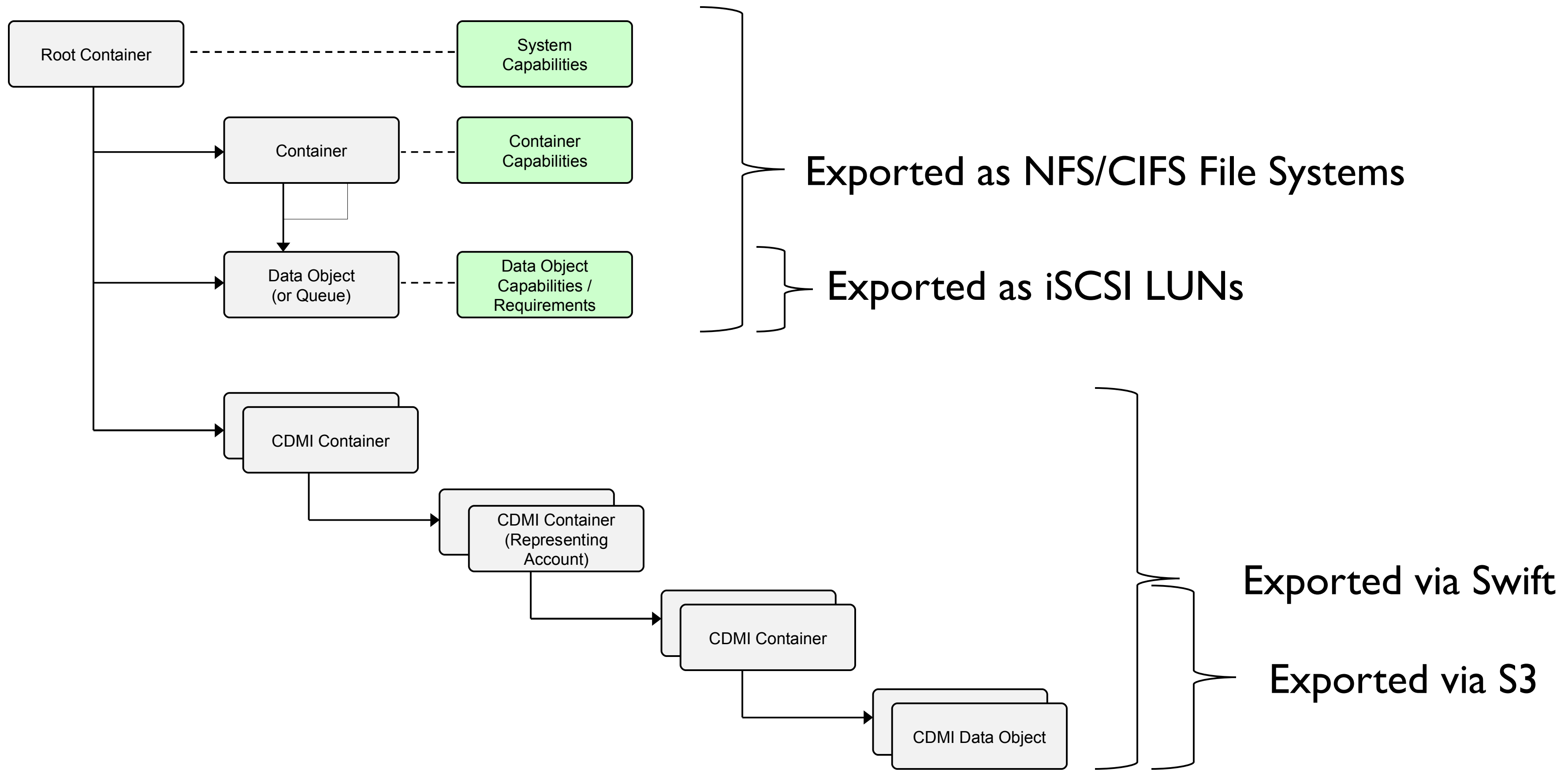


Overlay for S3 Object Model



Swift and CDMI Models

CDMI Cross-namespace Management for Multiple Protocol



* File-system-like hierarchies can be emulated on top of S3/Swift, but lack much of the operational expressiveness

CDMI Object Metadata

- 2 major classes – System and User
- SYSTEM Metadata
  - Storage System: Timestamps, Traditional ACLs, Counts etc.
  - Data System: requirements of the object – eg. Retention, Backup, Replication, Performance
- USER Metadata
  - Application and data specific
  - E.g. EXIF data on photos
  - Location data of objects
  - Relationship data
  - SEARCHABLE!

CDMI Capability Metadata

- Describes the capability of a service participating in CDMI cloud environment
  - Performance
  - Retention capability
  - Location information
  - Storage features (Compression, Encryption, Hashes etc.)
- Shrink to fit development and consumption model
  - Advertising capability means that developers only need to implement the standard partially and only advertise what is implemented

CDMI and Swift

SNIA
Cloud Storage Initiative

- CDMI works alongside Swift and S3 models, not replacing them
- Any of the APIs can be used to access the same data
- CDMI has been implemented for Swift as a filter which allows leverage of the Swift authentication filter
- If Swift and CDMI disagree, then CDMI "faults"
- Swift and CDMI use hierarchical containers but are slightly different in the implementation and language used
  - Swift Folder = CDMI Container
- CDMI Metadata can be stored in Swift metadata storage
  - This means the size limitation is implied for CDMI metadata currently

Modules and What they do

- cdmi.py – entry point
- cdmiconrollers.py – if a CDMI request
- noncdmicontrollers.py – non CDMI request
- cdmibase.py – base class to extended
- cdmiutils.py – utility classes and methods
- test_cdmi_container.py – testing of CDMI container functions
- test_cdmi_object.py – testing of CDMI object functions

Module Structure

- The implementation includes total of 8 modules.
  - Six modules are the implementation and two modules are the test cases.
- The first module is named cdmi.py which serves as a bootstrap or SWGI server.
  - This module also defines the filter_factory method which is to make the entire module an OpenStack filter.
  - inspect each request and dispatch the request to container or data object controllers in other two modules

Module Structure

- cdmicommoncontroller.py is created to handle common operations among container and data object such as read, delete, and capabilities
  - Responds to the capability (a GET) request for both container and data object
- cdmicontrollers.py and noncdmicontrollers.py are created to handle cdmi content type and non-cdmi content type request operations such as container creation and update

Module Structure

- cdmibase.py defines a base controller class so that the other controllers can inherit from it
- cdmiutils.py a few utility methods so the entire implementation can be a little cleaner
- test_cdmi_container.py tests containers and capabilities
- test_cdmi_object.py tests objects
- 70 test cases created to verify various use cases

Installation and Testing

- swift/common/middleware
  - cdmi.py
- swift/common/middleware/cdmiapps
  - cdmiconrollers.py – if a CDMI request
  - noncdmicontrollers.py – non CDMI request
  - cdmibase.py – base class to extended
  - cdmiutils.py – utility classes and methods
- swift/test/functional/cdmi
  - test_cdmi_container.py – testing of CDMI container functions
  - test_cdmi_object.py – testing of CDMI object functions

Setup

1. In /etc/swift/proxy-server.conf, add cdmi filter before proxy-server

```
[pipeline:main]
pipeline = healthcheck cache tempauth *cdmi* proxy-server

[filter:cdmi]
use = egg:swift#cdmi
```

2. In <swiftroot>/swift.egg-info/entry-points.txt, add the following line at the end of the section [paste.filter_factory]

```
cdmi = swift.common.middleware.cdmi:filter_factory
```

3. Once the above two steps are done, restart swift.

Optional Configuration

Optionally, this implementation can be configured to use different access root.

By default, the access root looks like "http://hostname:port/cdmi", you can configure the implementation to use different root like the following

```
http://hostname:port/another/root
```

Simply change /etc/swift/proxy-server.conf file by adding the following line in the [default] section:

```
cdmi_root = another/root
```

The cdmi capability prefix can also be configured by adding the following line.

```
cdmi_capabilities = cap.
```

When it is configured this way, the cdmi capability URL may look like this http://host:port/cdmi/cap/some/thing/else

Testing

CDMI test cases were developed as functional tests, it will access a running Swift system with CDMI filter enabled. Before you run the test cases, make sure CDMI filter configuration is correct by checking the proxy-server.conf file of your installation.

Once OpenStack Swift is up and running, switch to the following directory

```
    <SwiftInstallRoot>/test/functional/cdmi
```

Run the following two commands to test container and objects operations.

  To test container behaviors, issue "python test_cdmi_container.py" command

  To test object behaviors, issue "python test_cdmi_object.py" command

Example - #1

Use http://hostname:port/cdmi to login, the host name and port should be the same host name and port number Swift uses, normally it should be 8080 in a Swift All In One environment. The login should be a GET request with header like the following:

```
X-Storage-User: test:tester
X-Storage-Pass: testing
```

Example - #2

Once you logged in, you should get something like the following in the response header:

```
X-Auth-Token: AUTH_tk629536dff4ca4915b55227ab88363370
X-Storage-Token: AUTH_tk629536dff4ca4915b55227ab88363370
X-Storage-Url: http://192.168.1.63:8080/cdmi/AUTH_test
```

Example - #3

Use the X-Auth-Token and the X-Storage-Url to retrieve all containers of an account. Use the storage url in the login response header to send a GET request with the auth token, you should receive the following.

Vanilla GET would return:

```
HTTP/1.1 200 OK
X-Account-Object-Count: 7
X-Account-Bytes-Used: 130
X-Account-Container-Count: 3
Accept-Ranges: bytes
Content-Length: 69
Content-Type: text/plain; charset=utf-8
Date: Tue, 24 Jan 2012 14:21:46 GMT

cdmi_test_container_11327335466
cdmi_test_container_11327335467
pics
```

Example - #4

A CDMI GET would return:

```
HTTP/1.1 200 OK
X-Account-Object-Count: 7
X-Account-Bytes-Used: 130
X-Account-Container-Count: 3
Accept-Ranges: bytes
Content-Type: application/json; charset=utf-8
Content-Length: 340
Date: Tue, 24 Jan 2012 14:19:51 GMT

{ "mimetype": "application/cdmi-container",
  "objectID": "41608b843cd98c2f598648fd2bd72c1fae0119be",
  "objectName": "AUTH_test",
  "parentURI": "/cdmi/",
  "objectType": "application/cdmi-container",
  "children": [
    "cdmi_test_container_11327335466",
    "cdmi_test_container_11327335467",
    "pics"
  ],
  "metadata": {}
}
```

Where is the capability URI?

Prepare your laptop

- Got network connectivity?
- First install Netbeans from the Network or USB stick
  - -> show on laptop
- Next copy over the .zip file and expand
  - -> show on laptop
- Now build and run the CDMI cloud
  - -> show on laptop
  - It should look like this…
- Install your favorite REST Client

Getting Started with CDMI RI

- The CDMI Cloud runs at <your URL>
  - Containers are created below this “root”
  - You can treat this root as a container and GET it
    - List of containers already created.
  - You can also discovery the Cloud capabilities

System-wide Capabilities

- cdmi_domains
  - If present and "true", indicates that the cloud storage system supports domains. If not present, the domainURI field shall not be present in response bodies and the "cdmi_domains" URI shall not be present.
- cdmi_export_cifs
  - If present and "true", indicates that the cloud storage system supports CIFS exports.
- cdmi_export_iscsi
  - If present and "true", indicates that the cloud storage system supports FC exports.
- cdmi_export_nfs
  - If present and "true", indicates that the cloud storage system supports NFS protocol exports.

System-wide Capabilities

- cdmi_export_occi_iscsi
  - If present and "true", indicates that the cloud storage system supports OCCI/iSCSI exports.
- cdmi_export_webdav
  - If present and "true", indicates that the cloud storage system supports WebDAV exports.
- cdmi_metadata_maxitems
  - If present, this capability specifies the maximum number of user-defined metadata items supported by the cloud storage system. If absent, there is no limit placed on the number of user-defined metadata items.
- cdmi_metadata_maxsize
  - If present, this capability specifies the maximum size in bytes of each user-defined metadata item supported by the cloud storage system. If absent, there is no limit placed on the size of user-defined metadata items

System-wide Capabilities

- cdmi_notification
  - If present and "true", indicates that the cloud storage system supports notification queues.
- cdmi_logging
  - If present and "true", indicates that the cloud storage system supports logging queues.
- cdmi_query
  - If present and "true", indicates that the cloud storage system supports query queues.
- cdmi_query_regex
  - If present and "true", indicates that the cloud storage system supports query with regular expressions.
- cdmi_query_contains
  - If present and "true", indicates that the cloud storage system supports query with the “contains” expressions.

System-wide Capabilities

- cdmi_query_tags
  - If present and "true", indicates that the cloud storage system supports query with tag-matching expressions.
- cdmi_query_value
  - If present and "true", indicates that the cloud storage system supports query of value fields.
- cdmi_queues
  - If present and "true", indicates that the cloud storage system supports queue objects.
- cdmi_security_access_control
  - If present and "true", indicates that the cloud storage system supports ACLs. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.

System-wide Capabilities

- cdmi_security_audit
  - If present and "true", indicates that the cloud storage system supports audit logging. See Section 17.3, "Security Logging" for additional information.
- cdmi_security_data_integrity
  - If present and "true", indicates that the cloud storage system supports data integrity/authenticity. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.
- cdmi_security_encryption
  - If present and "true", indicates that the cloud storage system supports data at-rest encryption. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.

System-wide Capabilities

- cdmi_security_immutability
  - If present and "true", indicates that the cloud storage system supports data immutability/retentions. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.
- cdmi_security_sanitization
  - If present and "true", indicates that the cloud storage system supports data/media sanitization. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.
- cdmi_serialization_json
  - If present and "true", indicates that the cloud storage system supports json as a serialization format.

System-wide Capabilities

- cdmi_snapshots
  - If present and “true”, indicates that the cloud storage system supports snapshots.
- cdmi_xmlrepresentation
  - This capability is reserved for future use, as the xml representation is not defined in this version of CDMI. This capability shall never be present for CDMI 1.0.x.
- cdmi_references
  - If present and “true”, indicates that the cloud storage system supports references.

Storage System Metadata Capabilities

- cdmi_acl
  - If present and "true", indicates that the cloud storage system supports ACLs. When a CDMI implementation supports ACLs for the purpose of access control, the system-wide capability of “cdmi_security_access_control” specified in Table 10 of Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no support for access control.
- cdmi_size
  - If present and "true", indicates that the cloud storage system shall generate a "size" storage system metadata for each stored object.
- cdmi_ctime
  - If present and "true", indicates that the cloud storage system shall generate a "ctime" storage system metadata for each stored object.

Storage System Metadata Capabilities

- cdmi_atime
  - If present and "true", indicates that the cloud storage system shall generate an "atime" storage system metadata for each stored object.
- cdmi_mtime
  - If present and "true", indicates that the cloud storage system shall generate an "mtime" storage system metadata for each stored object.
- cdmi_acount
  - If present and "true", indicates that the cloud storage system shall generate an "acount" storage system metadata for each stored object.
- cdmi_mcount
  - If present and "true", indicates that the cloud storage system shall generate an "mcount" storage system metadata for each stored object.
- ACLs.

Storage System Metadata Capabilities

- cdmi_hash
  - If present, indicates that the cloud storage system shall generate a "cdmi_hash" storage system metadata for each stored object using the algorithm specified in the value of the cdmi_value_hash data system metadata.
- cdmi_acl
  - If present and "true", indicates that the cloud storage system shall support ACLs.

Data System Metadata Capabilities

- cdmi_data_redundancy
  - If present, this capability specifies the maximum number of redundancy copies that may be specified. If absent, redundancy copies specified shall be ignored.
- cdmi_infrastructure_redundancy
  - If present, this capability specifies the maximum number of infrastructure redundancy copies that may be specified. If absent, infrastructure redundancy copies specified shall be ignored.
- cdmi_data_dispersion
  - If present and “true”, indicates that the cloud storage system shall disperse data. If absent, redundancy copies specified shall be ignored.
- cdmi_data_retention
  - If present and "true", indicates that the cloud storage system shall support retention.

Data System Metadata Capabilities

- cdmi_data_autodelete
  - If present and “true”, indicates that the cloud storage system shall support the autodeletion of objects when retention ends.
- cdmi_data_holds
  - If present and "true", indicates that the cloud storage system shall support placing holds on objects. When a CDMI implementation supports holds for the purpose of making data immutable, the system-wide capability of “cdmi_security_immutability” specified in Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no support for data immutability.

Data System Metadata Capabilities

- cdmi_data_holds
  - If present and "true", indicates that the cloud storage system shall support placing holds on objects. When a CDMI implementation supports holds for the purpose of making data immutable, the system-wide capability of “cdmi_security_immutability” specified in 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no support for data immutability.
- cdmi_encryption
  - If present, this capability lists the encryption algorithms and key lengths supported. If absent, objects shall not be encrypted. When a CDMI implementation supports at-rest encryption, the system-wide capability of “cdmi_security_encryption” specified in "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no support for at-rest encryption.

Data System Metadata Capabilities

- cdmi_value_hash
  - This metadata is used to indicate if the object data is to be hashed and indicates the desired hash algorithm and length. Supported algorithm/length values are provided by the cdmi_value_hash capability.
- cdmi_latency
  - If present and "true", indicates that the cloud storage system shall tier data based on desired latency. If absent, the max latency specified shall be ignored.
- cdmi_throughput
  - If present and "true", indicates that the cloud storage system shall tier data based on desired throughput. If absent, the max throughput specified shall be ignored.

Data System Metadata Capabilities

- cdmi_sanitization_method
  - If present, this capability lists the sanitization methods supported. When a CDMI implementation supports sanitization, the system-wide capability of “cdmi_security_sanitization” specified in Table 10 of Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no sanitization support.
- cdmi_RPO
  - If present and "true", indicates that the cloud storage system shall manage data to achieve a specified RPO. If absent, the RPO specified shall be ignored.
- cdmi_RTO
  - If present and "true", indicates that the cloud storage system shall manage data to achieve a specified RTO. If absent, the RTO specified shall be ignored.

Data Object Capabilities

SNIA Cloud Storage Initiative

- cdmi_read_value
  - If present and "true", indicates that the object's value may be read.
- cdmi_read_value_range
  - If present and "true", indicates that the object's value may be read with byte ranges.
- cdmi_read_metadata
  - If present and "true", indicates that the object's metadata may be read.
- cdmi_modify_value
  - If present and "true", indicates that the object's value may be modified.
- cdmi_modify_value_range
  - If present and "true", indicates that the object's value may be modified with byte ranges.

Data Object Capabilities

- cdmi_modify_metadata
  - If present and "true", indicates that the object’s metadata may be modified.
- cdmi_serialize_dataobject
  - If present and "true", indicates that the object may be serialized.
- cdmi_delete_dataobject
  - If present and "true", indicates that the object may be deleted.

Container Capabilities

- cdmi_list_children
  - If present and "true", indicates that the container's children may be listed.
- cdmi_list_children_range
  - If present and "true", indicates that the container's children may be listed with ranges.
- cdmi_read_metadata
  - If present and "true", indicates that the container's metadata may be read.
- cdmi_modify_metadata
  - If present and "true", indicates that the container's metadata may be modified.

Container Capabilities

- cdmi_snapshot
  - If present and "true", indicates that the container allows a new snapshot to be created.
- cdmi_serialize_container
  - If present and "true", indicates that the container and all children's contents may be serialized.
- cdmi_deserialize_container
  - If present and "true", indicates that the container permits the deserialization of serialized containers and associated serialized children into the container
- cdmi_deserialize_queue
  - If present and "true", indicates that the container permits the deserialization of serialized queues into the container.

Container Capabilities

SNIA
Cloud Storage Initiative

- cdmi_deserialize_dataobject
  - If present and "true", indicates that the container permits the deserialization of serialized data objects into the container.
- cdmi_create_dataobject
  - If present and "true", indicates that the container allows a new object to be added.
- cdmi_post_dataobject
  - If present and "true", indicates that the container allows a new object to be added via POST.
- cdmi_post_queue
  - If present and "true", indicates that the container allows a new queue to be added via POST.
- cdmi_create_container
  - If present and "true", indicates that the container allows a new container may be added.

Container Capabilities

- cdmi_create_queue
  - If present and "true", indicates that the container allows queues to be created.
- cdmi_create_reference
  - If present and "true", indicates that the container allows a new child reference may be added.
- cdmi_delete_container
  - If present and "true", indicates that the container may be deleted.
- cdmi_move_container
  - If present and "true", indicates that the container may be moved (via PUT) to another URI.
- cdmi_copy_container
  - If present and "true", indicates that the container may be copied (via PUT) to another URI.

Exercise #1 - capabilities

SNIA™ Cloud Storage Initiative

- Format a GET request to /cdmi_capabilities
  - What do you see?
  - -> show on laptop
  - Format GET request to /cdmi_capabilities/container
  - What do you see?
  - -> show on laptop
  - Format GET request to /cdmi_capabilities/container/default
  - Format GET request to /cdmi_capabilities/dataobject

CDMI Containers

- CDMI Containers are logical place to put objects
  - To organize them
  - To give them the same "treatment" i.e. Data System Metadata
- Containers can be accessed by their URL or ObjectID
  - http://cloud.example.com/container/
  - http://cloud.example.com/cdmi_objectid/0000706D0010B84FAD185C425D8B537
- Containers can have child objects or child containers
- Container children can be discovered directly
  - http://cloud.example.com/container?children
  - http://cloud.example.com/container?children:0-2

CDMI Containers

- Fields in a CREATE operation include
  - *Metadata*
    - Metadata for the container.
  - • If this field is included when deserializing, serializing, copying, or moving a container, the value provided in this field shall replace the metadata from the source URI.
  - • If this field is not included when deserializing, serializing, copying, or moving a container, the metadata from the source URI shall be used.
  - • If this field is included when creating a new container by specifying a value, the value provided in this field shall be used as the metadata.
  - • If this field is not included when creating a new container by specifying a value, an empty JSON object ("{}") shall be assigned as the field value.
  - • This field shall not be included when referencing a container.

CDMI Container Fields

- *domainURI*
  - URI of the owning domain.
  - If different from the parent domain, the user shall have the "cross_domain" privilege.
  - If not specified, the parent domain shall be used.
- *Exports*
  - A structure for each protocol enabled for this container. This field shall not be included when referencing a data object.
- *Deserialize*
  - URI of a serialized CDMI data object that shall be deserialized to create the new container, including all child objects inside the source serialized data object.
  - When deserializing a container, any exported protocols from the original serialized container are not applied to the newly created container(s).

CDMI Container Fields

Cloud Storage Initiative

- *Copy*
  - URI of a CDMI container that shall be copied into the new container, including all child objects under the source container. When copying a container, exported protocols are not preserved across the copy.
- *Move*
  - URI of a CDMI container that shall be copied into the new container, including all child objects under the source container., then removing the container at the source URI upon the successful completion of the copy.
- *Reference*
  - URI of a CDMI data object that shall be pointed to by a reference. If other fields from this table are supplied when creating a reference, the server shall respond with a 400 Bad Request error response.
- Deserializevalue
  - A container object serialized as specified in Chapter 15, "Serialization/Deserialization”.

Exercise #2 - Containers

- Format PUT command to create /MyContainer
  - -> show on laptop
- Format GET request to see metadata on MyContainer

CDMIObjects

SNIA™
Cloud Storage Initiative

- CDMI Objects are where data is stored
- Objects have a “value” where the bytes are stored
- Objects also have “metadata” and fields
- Object fields include:
- *Mimetype*
  - MIME type of the data contained within the value field of the data object.
  - This field shall be kept as part of the metadata and shall be included when deserializing, serializing, copying, moving, or referencing a data object.
  - • If this field is not specified, the value of "text/plain" shall be assigned as the field value.
  - • This field shall not be included when referencing a data object

CDMI Object Fields

- *metadata*
  - JSON Object
  - Metadata for the data object.
  - • If this field is included when deserializing, serializing, copying, or moving a data object, the value provided in this field shall replace the metadata from the source URI.
  - • If this field is not included when deserializing, serializing, copying, or moving a data object, the metadata from the source URI shall be used.
  - • If this field is included when creating a new data object by specifying a value, the value provided in this field shall be used as the metadata.
  - • If this field is not included when creating a new data object by specifying a value, an empty JSON object ("{}") shall be assigned as the field value.
  - • This field shall not be included when referencing a data object.

CDMIObjects

SNIA™ Cloud Storage Initiative

- *domainURI*
  - URI of the owning domain. If different from the parent domain, the user shall have the "cross_domain" privilege. If not specified, the parent domain shall be used.
- *Deserialize*
  - URI of a serialized CDMI data object that shall be deserialized to create the new data object.
- *Serialize*
  - URI of a CDMI object that shall be serialized into the new data object.
- *Copy*
  - URI of a CDMI data object or queue that shall be copied into the new data object.
- *Move*
  - URI of a CDMI data object or queue that shall be copied into the new data object, then removing the data object or queue value at the source URI upon the successful completion of the copy.

CDMIObjects

- *Reference*
  - URI of a CDMI data object that shall be pointed to by a reference. If other fields from this table are supplied when creating a reference, the server shall respond with a 400 Bad Request error response.
- Deserializevalue
  - A data object serialized as specified in Chapter 15, "Serialization/Deserialization".
- *Value*
  - JSON-encoded data.
  - • If this field is not included, an empty JSON String ("") shall be assigned as the field value.
  - • Binary data shall be escaped as per the JSON escaping rules described in [RFC4627].

Exercise #3 – Data Objects

- Format PUT request to create dataobject to MyContainer/MyObject.txt
  - Using the text "This is my object"
- Format GET request from MyContainer/MyObject.txt
- Browser fetch of MyContainer/MyObject.txt
- Update MyContainer/MyObject.txt with new metadata and modify text
- Browser fetch of MyContainer/MyObject.txt

Domains

- Domains represent the concept of administrative ownership
- Each Container or Object has a DomainURI to the Domain that “owns” it
- Domain summaries are available to summarize the cloud usage
- Sub-container of Domain called cdmi_domain_summary
  - cumulative – data from time zero
  - daily, monthly yearly

Domain Summary Fields

- *cdmi_domainURI*
- Domain name corresponding to the domain that is summarized.
- *cdmi_summary_start*
  - An [ISO-8601] time indicating the start of the time range that the summary information is presenting.
- *cdmi_summary_end*
  - An [ISO-8601] time indicating the end of the time range that the summary information is presenting.
- *cdmi_summary_objecthours*
  - The sum of the time each object belonging to the domain existed during the summary time period.
- *cdmi_summary_objectsmin*
  - The minimum number of objects belonging to the domain during the summary time period.

Domain Summary Fields

- *cdmi_summary_objectsmax*
  - The maximum number of objects belonging to the domain during the summary time period.
- *cdmi_summary_objectsaverage*
  - The average number of objects belonging to the domain during the summary time period.
- *cdmi_summary_puts*
  - The number of objects written to the domain.
- *cdmi_summary_gets*
  - The number of objects read from the domain.
- *cdmi_summary_bytehours*
  - The sum of the time each byte belonging to the domain existed during the summary time period.

Domain Summary Fields

- *cdmi_summary_bytesmin*
  - The minimum number of bytes belonging to the domain during the summary time period.
- *cdmi_summary_bytesmax*
  - The maximum number of bytes belonging to the domain during the summary time period.
- *cdmi_summary_bytesaverage*
  - The average number of bytes belonging to the domain .during the summary time period
- *cdmi_summary_writes*
  - The number of bytes written to the domain.
- *cdmi_summary_reads*
  - The number of bytes read from the domain.

Domain Summary Fields

- *cdmi_summary_charge*
  - A free-form monetary measurement of the charge for the use of the service to the user of the domain.
- *cdmi_summary_kwhours*
  - The sum of power consumed by the domain during the summary time period.
- *cdmi_summary_kwmin*
  - The minimum power consumed by the domain during the summary time period.
- *cdmi_summary_kwmax*
  - The maximum power consumed by the domain during the summary time period.
- *cdmi_summary_kwaverage*
  - The average power consumed by the domain during the summary time period.

Domain Members

- Each Domain can have members (users)
  - Each with their own credentials
- Domain member fields include:
- *cdmi_member_enabled*
  - Indicates if the member is enabled.
- *cdmi_member_type*
  - The type of member record. Values include "user" and "delegation".
- *cdmi_member_name*
  - This field contains the user name as presented by the client.
- *cdmi_member_credentials*
  - This field contains credentials to be matched against the credentials as presented by the client. If this field is not present, one or more delegations must be present and shall be used to resolve user credentials.
-

Domain Member Fields

SNIA
Cloud Storage Initiative

- *cdmi_member_principal*
  - This field indicates to which principal name (used in ACLs) the user is mapped. If this field is not present, one or more delegations must be present and shall be used to resolve the user principal.
- *cdmi_member_privileges*
  - JSON Array of JSON Strings
  - This field contains a JSON list of special privileges associated with the user.
  - The following privileges are defined:
  - • "administrator". All ACL access checks are always successful.
  - • "backup_operator". All read ACL access checks are always successful.
  - • "cross_domain". Operations specifying a domain other than the domain of the parent object are permitted.

Domain Members

- *cdmi_member_groups*
  - JSON Array of JSON Strings
  - This field contains a JSON array of group names to which the user belongs.

Domain Fields/Operations

- *Metadata*
  - Metadata for the domain. If this field is included when copying a domain object, the value provided shall be replace the metadata from the source URI. If this field is not specified, an empty JSON object ("{}") shall be assigned as the field value.
- *Copy*
  - URI of a CDMI domain that shall be copied into the new domain, including all child domains and membership from the source domain.
- Deserialize
  - URI of a serialized CDMI data object that shall be deserialized to create the new domain, including all child objects inside the source serialized data object.
- Deserializevalue
  - A domain object serialized as specified in Chapter 15, "Serialization/Deserialization".

Queues

- Queues are a special type of object with first-in/first-out semantics
- Queue writer PUTs objects into a queue
- Queue reader GETs objects from a queue
- Example:
  - GET to the queue object URI to read all fields of the queue object
  - GET /MyContainer/MyQueue HTTP/1.1
  - Host: cloud.example.com
  - Accept: application/cdmi-queue
  - X-CDMI-Specification-Version: 1.0

Get from Queue Response

- The response looks like:
  - HTTP/1.1 200 OK
  - Content-Type: application/cdmi-queue
  - X-CDMI-Specification-Version: 1.0

```
-   {
-     "objectURI" : "/MyContainer/MyQueue",
-     "objectID" : "0000706D00101ADEBC119D1BFE98672A",
-     "objectName" : "MyQueue",
-     "parentURI" : "/MyContainer/",
-     "domainURI" : "/cdmi_domains/MyDomain/",
-     "capabilitiesURI" : "/cdmi_capabilities/Queue/",
-     "completionStatus" : "Complete",
-     "metadata" : {
-     },
-     "queueValues" : "1-2",
-     "mimetype" : [
-         "text/plain"
-     ],
-     "valuerange" : [
-         "0-19"
-     ],
-     "value" : [
-         "First Enqueued Value"
-      ]
-   }
```

Exported Protocols

- Containers can be “exported” with legacy protocols
- Some Solaris code exists to export via NFS
  - See if you can find it

- How would you export via WebDAV?
  - Hint: google “Milton”

- How would you export via iSCSI?

Data System Metadata

- *cdmi_data_redundancy*
  - Contains the desired number of complete copies of the data item to be maintained. This number determines the minimum number of primary copies of the data that the cloud shall maintain. Additional primary copies may be made to satisfy demand for the value.
- *cdmi_immediate_redundancy*
  - If present and set to the value "true", indicates that at least a cdmi_data_redundancy number of copies shall contain the newly written value before the operation completes. This metadata is used to make sure that multiple copies of the data are written to permanent storage to prevent possible data loss.
- cdmi_assignedsize (only valid for a container.)
  - Contains the number of bytes that are reported via exported protocols (and may be thin provisioned by the system). This number may limit cdmi_size for the container. This metadata is the size that shall be shown through any number of data path protocols that are used to export a container. If the container is thin provisioned, this may be greater than the actual storage consumed.

Data System Metadata

- *cdmi_infrastructure_redundancy*
  - Contains the number of desired independent storage infrastructures supporting the data. This metadata is used to convey that, of the primary copies specified in cdmi_data_redundancy, these copies shall be stored on this many separate infrastructures. Any two infrastructures may not share common elements, such as a network or power source.
- *cdmi_data_dispersion*
  - Contains the desired distance (km) between the infrastructures supporting the multiple copies of data. This metadata is used to separate the (cdmi_infrastructure_redundancy number of) infrastructures by a minimum geographic distance to prevent data loss due to site disasters.

Data System Metadata

- *cdmi_geographic_placement*
  - JSON Array of JSON Strings
  - Contains a list of geopolitical identifiers, each specifying a region where the object is permitted or not permitted to be stored. Geopolitical boundaries are a list of ISO-3166 country codes. A "!" in front of a country code excludes that country from the previous list of geopolitical boundaries. This metadata limits where the data may be placed physically and constrains all cloud movement of the data within the cloud. It does not apply to data once it leaves the cloud. This metadata takes precedent over other metadata, such as cdmi_data_dispersion.
- *cdmi_retention_id*
  - Contains a user-specified retention identifier. This metadata is a user-specified text field that is used to tag a given object as being managed by a specific retention policy. It is not required to place an object under retention but is useful when needing to be able to perform a query to find all objects under a specific retention policy.

Data System Metadata

- *cdmi_retention_period*
  - Contains an [ISO-8601] time interval (as described in Section 5.14) during which the object is under retention. Only the end-date may be altered when updated. If an object is under retention, the object may not be deleted and its value may not be altered. After the retention date has passed, the object may be deleted.
- *cdmi_retention_autodelete*
  - This metadata is used to indicate if the object is to be automatically deleted when retention expires. The value of this metadata item shall be "true" when set.
- *cdmi_hold_id*
  - JSON Array of JSON Strings
  - This metadata is used to indicate if the object is to be placed under a retention hold. If the array is not empty, the object is under a hold, with each string in the array containing a user-specified hold identifier. If an object is under one or more holds, the object is completely immutable.

Data System Metadata

- *cdmi_encryption*
  - This metadata is used to indicate if the object is to be encrypted and indicates the desired encryption algorithm, the mode of operation, and the key size. This metadata is the desired encryption support that the client is requesting of the cloud. All data related to the data item/container shall be encrypted when this value is set, including metadata. This metadata is the desired encryption support that the client is requesting of the cloud. Using the template, ALGORITHM_MODE_KEYLENGTH, the client is able to specify the encryption where:
    - "ALGORITHM" is the encryption algorithm (e.g., "AES" or "3DES").
    - "MODE" is the mode of operation (e.g., "XTS", "CBC", or "CTR").
    - "KEYLENGTH" is the key size (e.g., "128", "192", "256").

Data System Metadata

- *cdmi_encryption (continued)*
- To improve interoperability between CDMI implementations, the following designators should be used for the more common encryption combinations:
  - • "3DES_ECB_168" for the three-key Triple DES algorithm, the Electronic Code Book (ECB) mode of operation, and a key size of 168 bits.
  - • "3DES_CBC_168" for the three-key Triple DES algorithm, the Cipher Block Chaining (CBC) mode of operation, and a key size of 168 bits.
  - • "AES_CBC_128" for the AES algorithm, the CBC mode of operation, and a key size of 128 bits.
  - • "AES_CBC_256" for the AES algorithm, the CBC mode of operation, and a key size of 256 bits.
  - • "AES_XTS_128" for the AES algorithm, the XTS mode of operation, and a key size of 128 bits.
  - "AES_XTS_256" for the AES algorithm, the XTS mode of operation, and a key size of 256 bits.

Data System Metadata

- *cdmi_value_hash*
  - If present, this capability lists the hash algorithm/lengths supported. If absent, objects shall not present a hash value as system metadata. Values are in the form of "Algorithm Length", for example, "SHA256". When a CDMI implementation supports hashing, the system-wide capability of "cdmi_security_data_integrity" specified in Table 10 of Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to "true". Otherwise, it shall not be present, indicating that there is no hashing support.
- *cdmi_latency*
  - Contains the desired maximum time to first byte, in milliseconds. This metadata is the desired latency (in milliseconds) to the first byte of data in a primary copy, as measured from the edge of the cloud and factoring out any propagation latency between the client and the cloud. For example, this metadata may be used to determine, in an interoperable way, from what type of storage medium the primary copy(s) of the data may be served.

Data System Metadata

- *cdmi_throughput*
  - Contains the desired maximum data rate on retrieve, in bytes per second. This metadata is the desired bandwidth (in Mbits/sec) to the primary copy of data, as measured from the edge of the cloud and factoring out any bandwidth capability between the client and the cloud. This metadata is used to stage the primary data copies in locations where there is sufficient bandwidth to accommodate a maximum usage.
- *cdmi_sanitization_method*
  - If present, this metadata specifies the sanitization method selected from the list in the cdmi_sanitization_method capability list. If absent, objects shall not be securely sanitized.

Data System Metadata

- *cdmi_RPO*
  - Contains the largest acceptable duration in time between an update and when the update may be recovered, specified in seconds. This metadata is used to indicate the desired backup frequency from the primary copy(s) of the data to the secondary copy(s). It is the maximum acceptable duration between a write to the primary copy and the backup to the secondary copy during which a failure of the primary copy(s) shall result in data loss.
- *cdmi_RTO*
  - Contains the largest acceptable duration in time to restore data, specified in seconds. This metadata is used to indicate the desired maximum acceptable duration to restore the primary copy(s) of the data from a secondary backup copy(s).

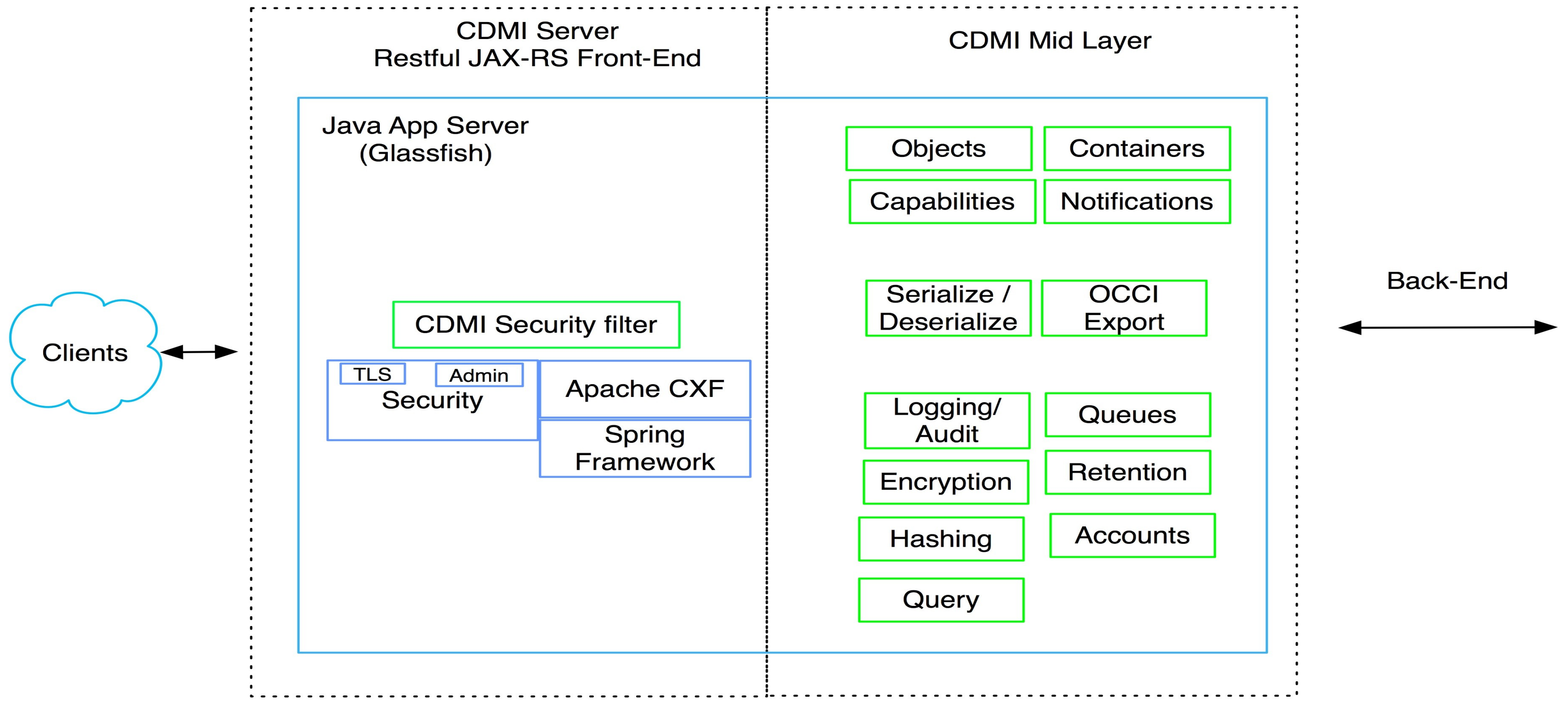
Extra Exercises

- Try out some other CDMI commands and see if they are implemented
- How would you do a Move Container?
- How would you snapshot a Container?
- What about Serialize/Deserialize?

For More information

- One Web Site to Remember: ***http://snia.org/cloud***
- Large Cloud Storage Community
  - http://groups.google.com/group/snia-cloud
  - http://twitter.com/SNIAcloud (@SNIAcloud)
  - http://www.google.com/profiles/SNIAcloud

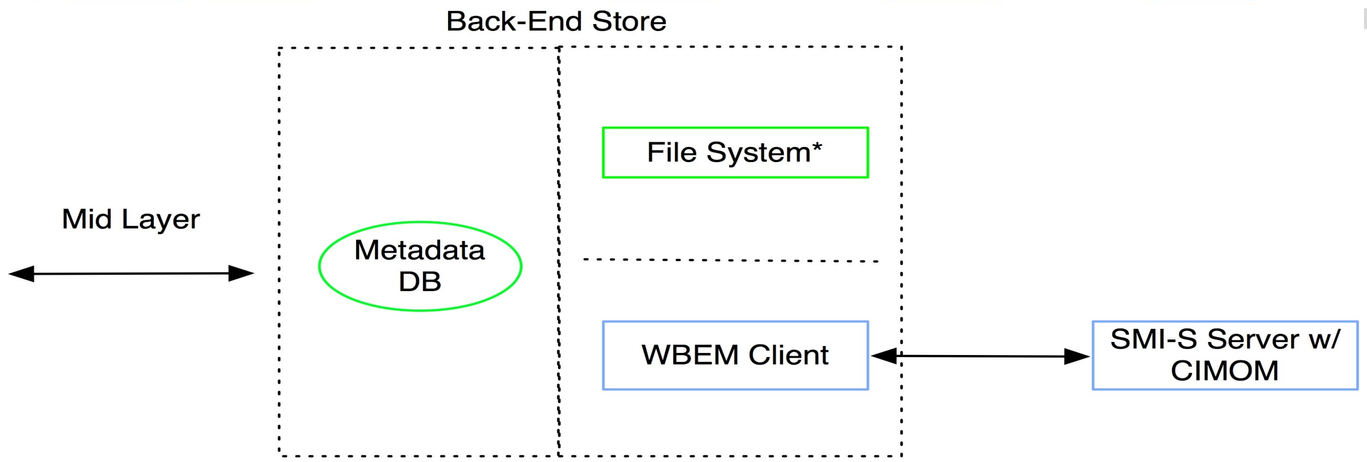
Reference Implementation Architecture



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

➢RI Architecture

- Mapping of CDMI Model to Filesystem:
  - Containers to Directories
  - Data Objects to Files
  - Capabilities to hidden files
  - Domains not yet implemented

RI Architecture

- Infrastructure information is stored in "."files in parent directory
  - CDMI "Fields"
  - CDMI "Metadata"
- Stored directly in JSON format to leverage serialize/deserialize functionality of Model classes
- Capabilities hierarchy in a editable "." file

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)

RI Design

- Java
- Uses Apache CXF framework implementation of JAX-RS

RI Design

- 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

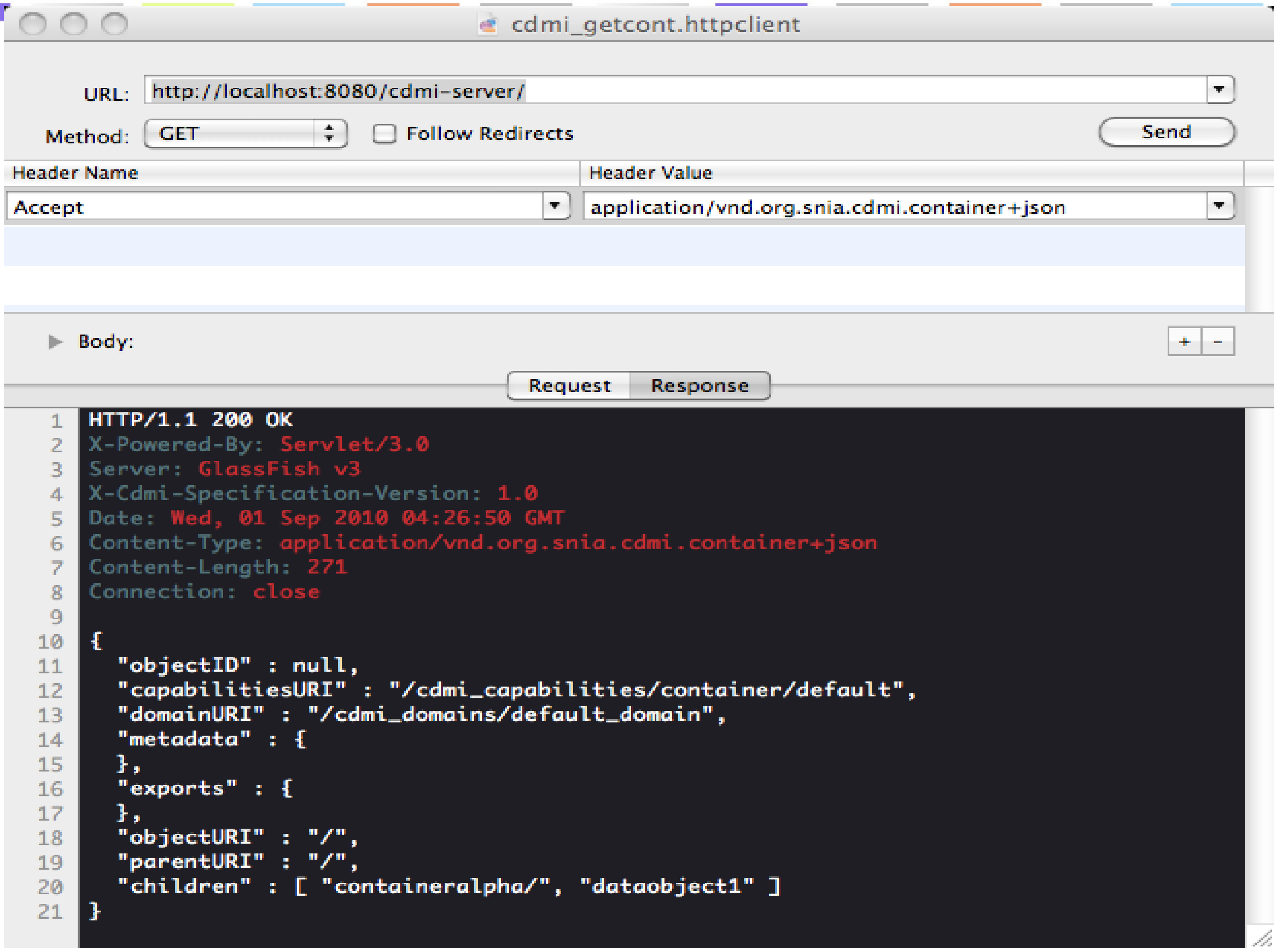
Design – DAO interfaces & impls

- ContainerDAO
- DataObjectDAO
- CapabilityDAO
- DomainDAO

- Uses java.io and java.io.File to access local filesystem

Demonstration

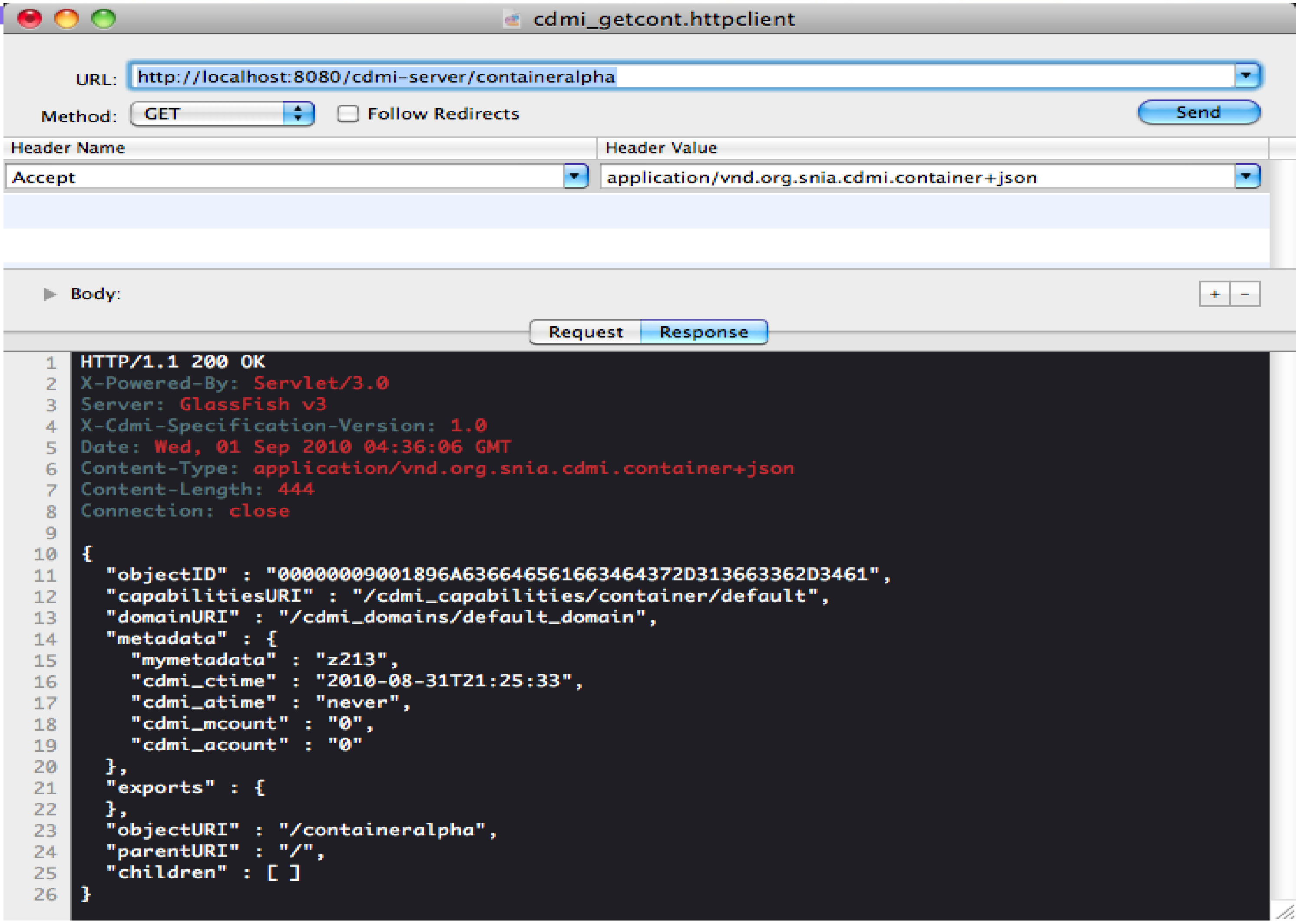
- OSX HttpClient
- Get CDMI containers, dataobjects, capabilities
- Create container and dataobject
- Inspect changes to underlying local filesystem



```
HTTP/1.1 200 OK
X-Powered-By: Servlet/3.0
Server: GlassFish v3
X-Cdmi-Specification-Version: 1.0
Date: Wed, 01 Sep 2010 04:26:50 GMT
Content-Type: application/vnd.org.snia.cdmi.container+json
Content-Length: 271
Connection: close

{
  "objectID" : null,
  "capabilitiesURI" : "/cdmi_capabilities/container/default",
  "domainURI" : "/cdmi_domains/default_domain",
  "metadata" : {
  },
  "exports" : {
  },
  "objectURI" : "/",
  "parentURI" : "/",
  "children" : [ "containeralpha/", "dataobject1" ]
}
```

Demonstration – Get container



URL: http://localhost:8080/cdmi-server/containeralpha

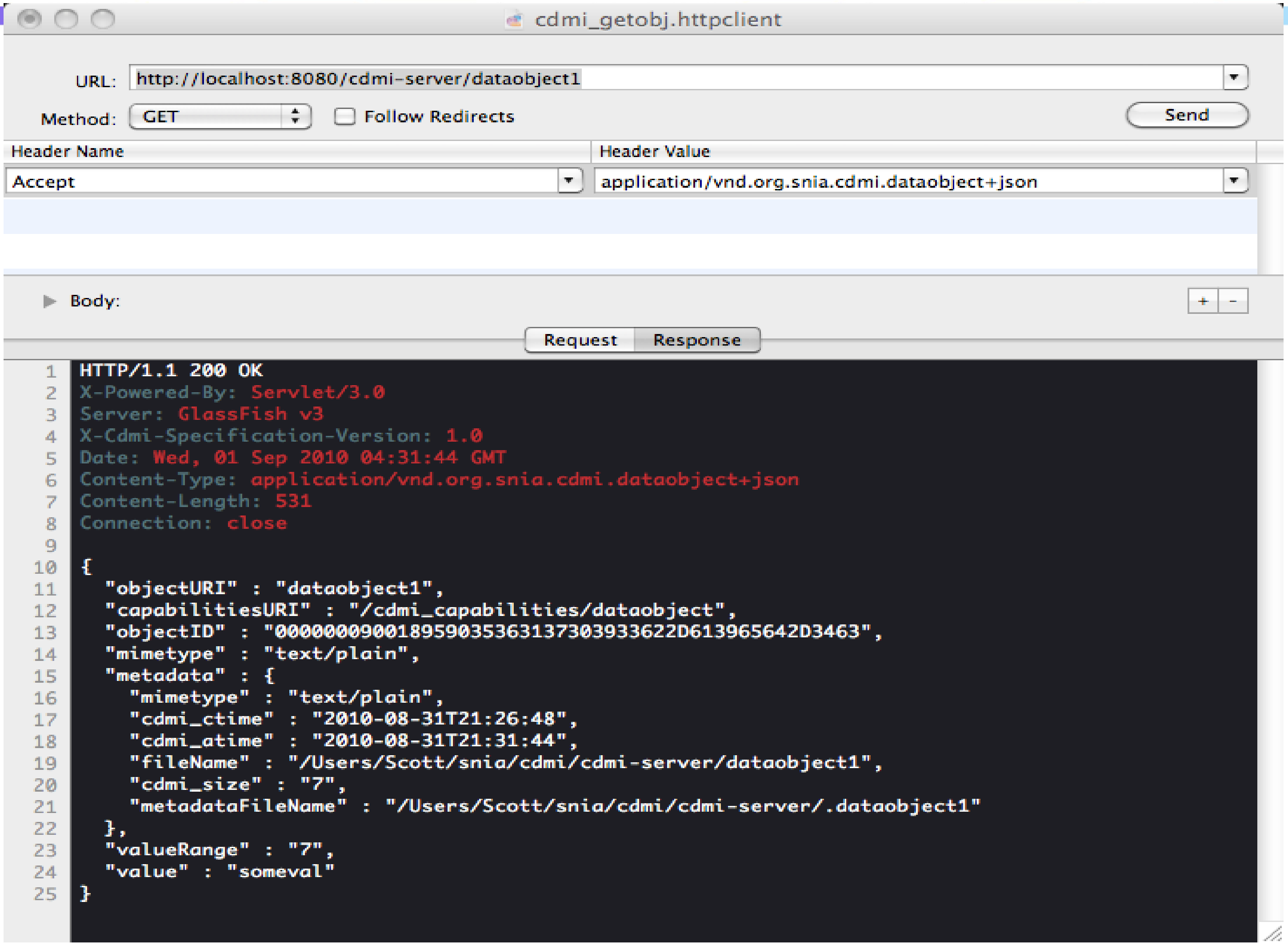
Method: GET ☐ Follow Redirects

Header Name	Header Value
Accept	application/vnd.org.snia.cdmi.container+json

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/3.0
Server: GlassFish v3
X-Cdmi-Specification-Version: 1.0
Date: Wed, 01 Sep 2010 04:36:06 GMT
Content-Type: application/vnd.org.snia.cdmi.container+json
Content-Length: 444
Connection: close

{
  "objectID" : "00000009001896A6366465616634643720313663362D3461",
  "capabilitiesURI" : "/cdmi_capabilities/container/default",
  "domainURI" : "/cdmi_domains/default_domain",
  "metadata" : {
    "mymetadata" : "z213",
    "cdmi_ctime" : "2010-08-31T21:25:33",
    "cdmi_atime" : "never",
    "cdmi_mcount" : "0",
    "cdmi_acount" : "0"
  },
  "exports" : {
  },
  "objectURI" : "/containeralpha",
  "parentURI" : "/",
  "children" : [ ]
}
```

Demonstration – Get object



URL: http://localhost:8080/cdmi-server/dataobject1

Method: GET

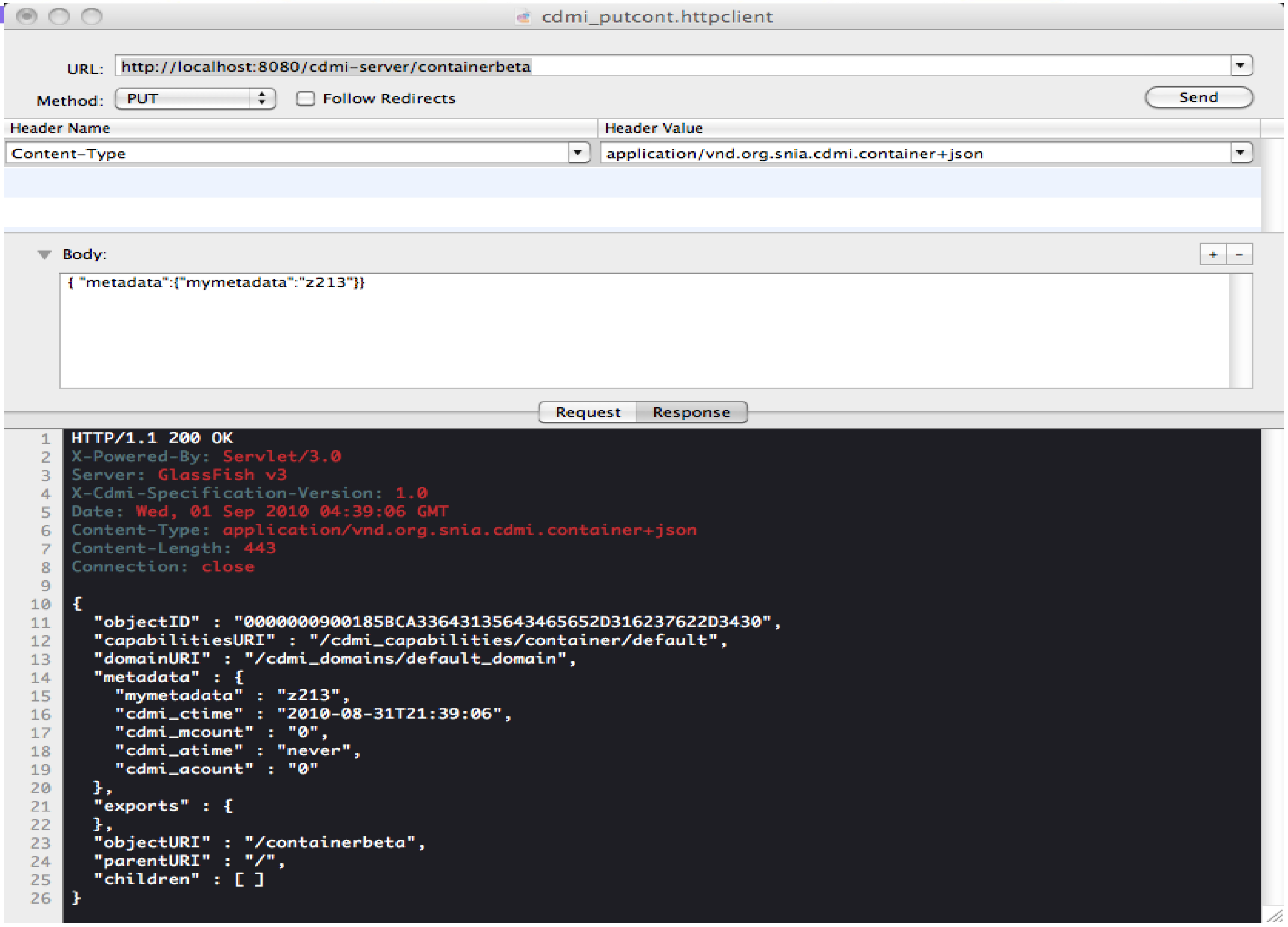
Follow Redirects

Header Name	Header Value
Accept	application/vnd.org.snia.cdmi.dataobject+json

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/3.0
Server: GlassFish v3
X-Cdmi-Specification-Version: 1.0
Date: Wed, 01 Sep 2010 04:31:44 GMT
Content-Type: application/vnd.org.snia.cdmi.dataobject+json
Content-Length: 531
Connection: close

{
  "objectURI" : "dataobject1",
  "capabilitiesURI" : "/cdmi_capabilities/dataobject",
  "objectID" : "00000009001895903536313730393362 2D613965642D3463",
  "mimetype" : "text/plain",
  "metadata" : {
    "mimetype" : "text/plain",
    "cdmi_ctime" : "2010-08-31T21:26:48",
    "cdmi_atime" : "2010-08-31T21:31:44",
    "fileName" : "/Users/Scott/snia/cdmi/cdmi-server/dataobject1",
    "cdmi_size" : "7",
    "metadataFileName" : "/Users/Scott/snia/cdmi/cdmi-server/.dataobject1"
  },
  "valueRange" : "7",
  "value" : "someval"
}
```

Demonstration – Create Container



URL: http://localhost:8080/cdmi-server/containerbeta

Method: PUT

Follow Redirects

Header Name	Header Value
Content-Type	application/vnd.org.snia.cdmi.container+json

Body:

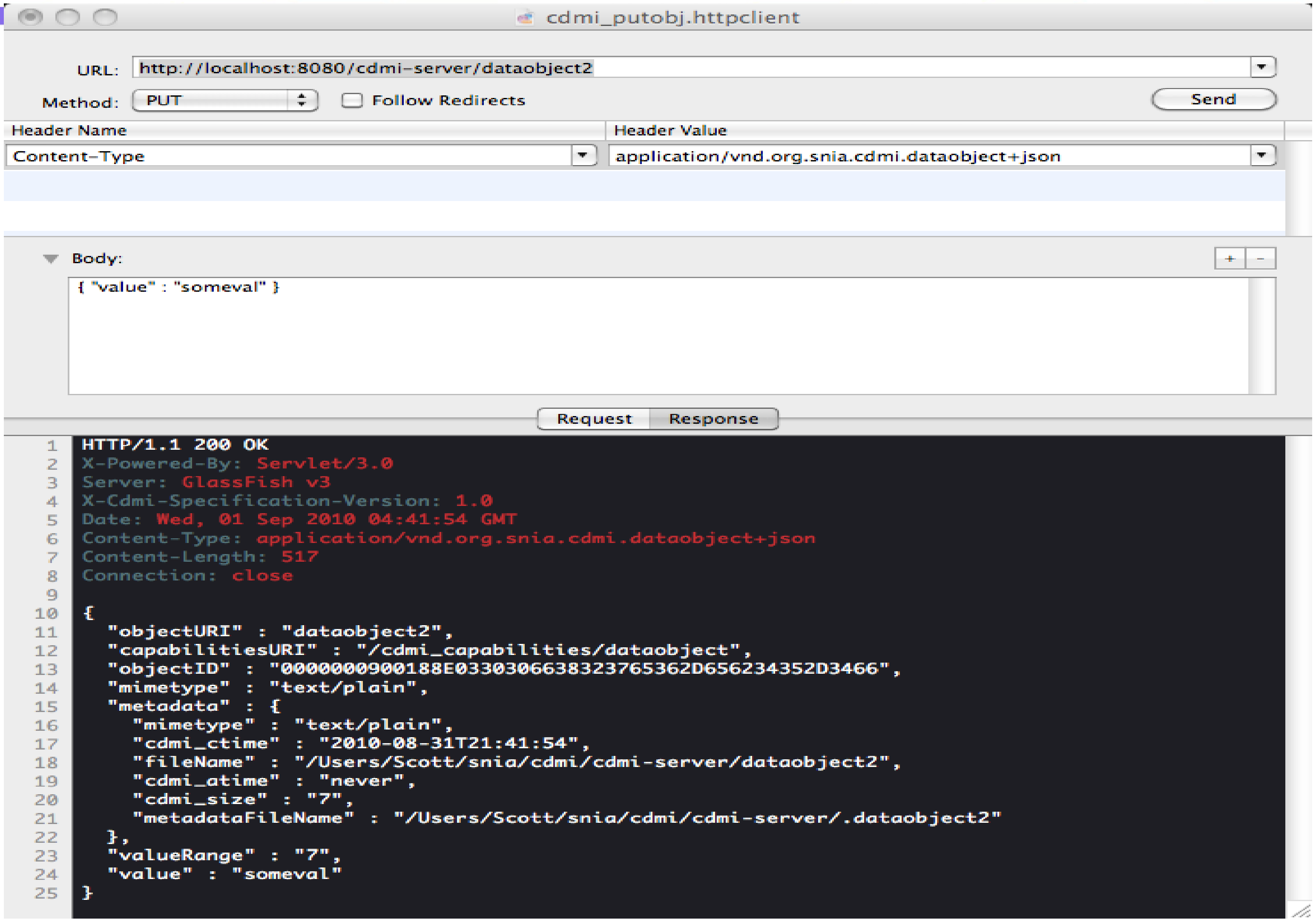
```
{ "metadata":{"mymetadata":"z213"}}
```

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/3.0
Server: GlassFish v3
X-Cdmi-Specification-Version: 1.0
Date: Wed, 01 Sep 2010 04:39:06 GMT
Content-Type: application/vnd.org.snia.cdmi.container+json
Content-Length: 443
Connection: close

{
  "objectID" : "00000009000185BCA33643135643465652D316237622D3430",
  "capabilitiesURI" : "/cdmi_capabilities/container/default",
  "domainURI" : "/cdmi_domains/default_domain",
  "metadata" : {
    "mymetadata" : "z213",
    "cdmi_ctime" : "2010-08-31T21:39:06",
    "cdmi_mcount" : "0",
    "cdmi_atime" : "never",
    "cdmi_acount" : "0"
  },
  "exports" : {
  },
  "objectURI" : "/containerbeta",
  "parentURI" : "/",
  "children" : [ ]
}
```

Demonstration – Create Object

SNIA Cloud Storage Initiative



URL: http://localhost:8080/cdmi-server/dataobject2

Method: PUT

☐ Follow Redirects

Header Name	Header Value
Content-Type	application/vnd.org.snia.cdmi.dataobject+json

Body:

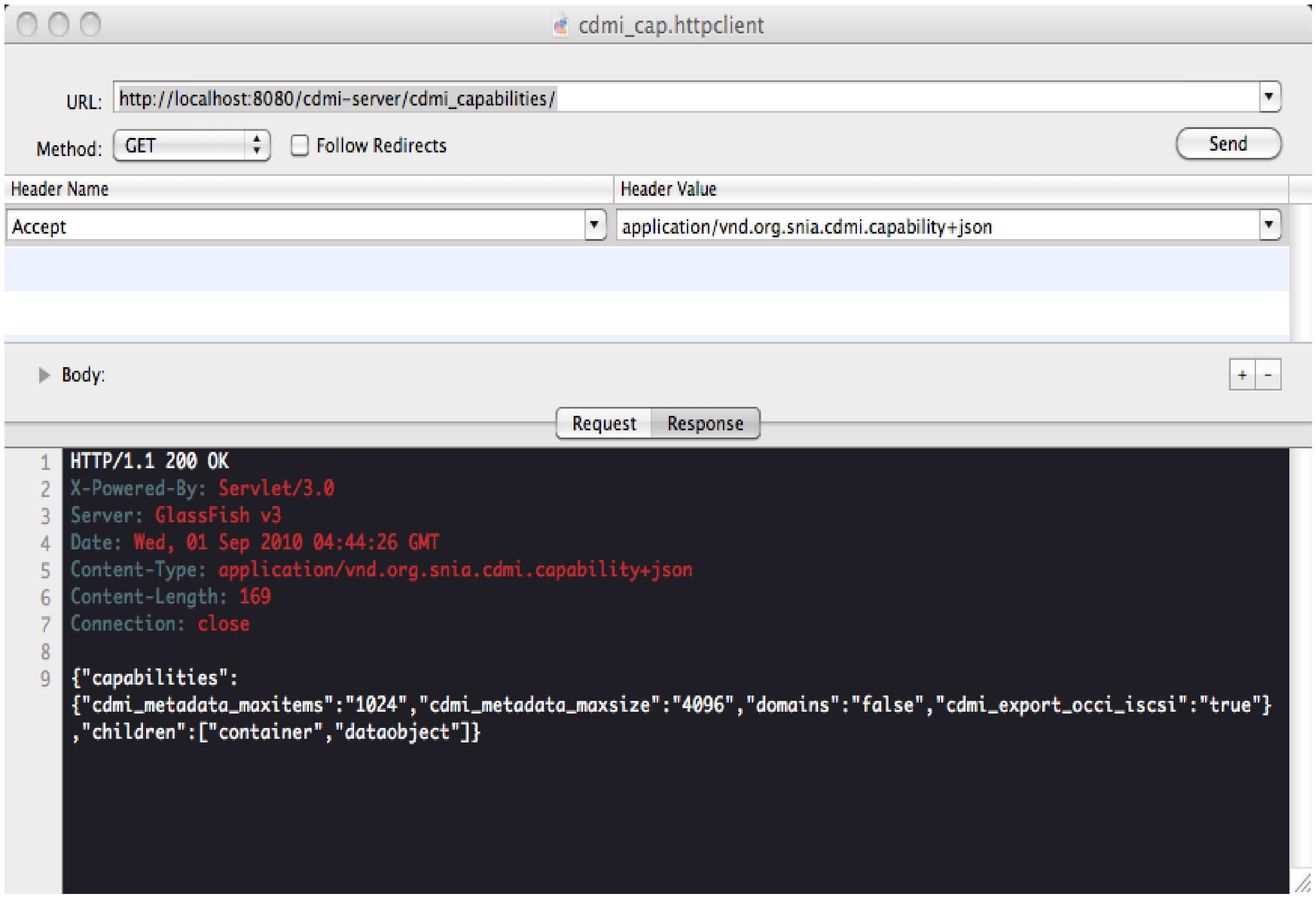
```
{ "value" : "someval" }
```

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/3.0
Server: GlassFish v3
X-Cdmi-Specification-Version: 1.0
Date: Wed, 01 Sep 2010 04:41:54 GMT
Content-Type: application/vnd.org.snia.cdmi.dataobject+json
Content-Length: 517
Connection: close

{
  "objectURI" : "dataobject2",
  "capabilitiesURI" : "/cdmi_capabilities/dataobject",
  "objectID" : "0000000900188E03303066383237653626D656234352D3466",
  "mimetype" : "text/plain",
  "metadata" : {
    "mimetype" : "text/plain",
    "cdmi_ctime" : "2010-08-31T21:41:54",
    "fileName" : "/Users/Scott/snia/cdmi/cdmi-server/dataobject2",
    "cdmi_atime" : "never",
    "cdmi_size" : "7",
    "metadataFileName" : "/Users/Scott/snia/cdmi/cdmi-server/.dataobject2"
  },
  "valueRange" : "7",
  "value" : "someval"
}
```

Demonstration – Capabilities

SNIA™
Cloud Storage Initiative



URL: http://localhost:8080/cdmi-server/cdmi_capabilities/

Method: GET ☐ Follow Redirects

Header Name	Header Value
Accept	application/vnd.org.snia.cdmi.capability+json

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/3.0
Server: GlassFish v3
Date: Wed, 01 Sep 2010 04:44:26 GMT
Content-Type: application/vnd.org.snia.cdmi.capability+json
Content-Length: 169
Connection: close

{"capabilities":
{"cdmi_metadata_maxitems":"1024","cdmi_metadata_maxsize":"4096","domains":"false","cdmi_export_occi_iscsi":"true"}
,"children":["container","dataobject"]}
```

Testing

- Test Client using JUnit

License

Cloud Storage Initiative

- BSD 3 Clause
- Oracle & SNIA
- Right to modify & redistribute provided entire license/copyright is included

Call for involvement

- We are looking for volunteers to help complete the Reference Implementation!
  - Access by ObjectID
  - Container Move/Copy
  - Domains
  - Queues
  - .....

SNIA™
Cloud Storage Initiative

Questions?

mark@carlson.net