



CDMI Reference Implementation – Developer’s Guide

Version 1.0c

“Publication of this Working Draft for review and comment has been approved by the Cloud Storage TWG. This draft represents a “best effort” attempt by the Cloud Storage TWG to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a “work in progress.” Suggestion for revision should be directed to the SNIA Technical Council Managing Director at tcmd@snia.org.”

WORKING DRAFT

May 13, 2011

Revision History

Version	Date	Originator	Comments
1.0a	5/18/10	M. McMinn	1st draft
1.0b	6/8/10	M. Carlson	Added content to Ch 2-4.
1.0c	05/13/11	M. McMinn	Added references; updated capabilities table (formatting only). Added content from Tom Weissinger for Chapter 5 - Reference Implementation Design.

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

- Any text, diagram, chart, table or definition reproduced shall be reproduced in its entirety with no alteration, and,
- Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced shall acknowledge the SNIA copyright on that material, and shall credit the SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, sell any excerpt or this entire document, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by e-mailing tcmd@snia.org please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

Copyright © 2011 Storage Networking Industry Association.

Contents

1 About this Guide	1
Purpose and Audience	1
Contents	1
References	1
Additional Information	2
Conventions	2
SNIA Welcomes Your Comments	2
2 Introduction	3
Terms of Use	3
Supported Operating Systems	3
Software Requirements	4
3 CDMI Architecture	5
Architecture Overview	5
File System Naming	6
Theory of Operations	6
4 CDMI Capabilities	9
Cloud Storage System-Wide Capabilities	9
Storage System Metadata Capabilities	10
Data System Metadata Capabilities	10
Data Object Capabilities	11
Container Capabilities	12
Domain Capabilities	12
Queue Object Capabilities	12
5 CDMI Reference Implementation Design	13
Introduction	13
Where Can I Find the Code?	13
Technologies Used	13
Using the SPI to Implement Your Own Storage Adapter	14
Creating a New Storage Adapter	14
Responding to a Request Sequence	15
Logging	15
Building and Deploying	15
Testing	15
6 CDMI Error Code Mapping	16

Chapter 1: About this Guide

Purpose and Audience

The *CDMI Reference Implementation – Developer’s Guide* is written for programmers and application developers who create custom applications for Cloud Storage Systems. This document, along with the *Cloud Data Management Interface Version 1.0*, provides the information that developers need to create custom applications for a Cloud Storage System.

Contents

The contents of this document are described as follows:

- Chapter 1, “About this Guide” describes the audience and purpose, contents of this guide, additional references and web sites, and typographical conventions.
- Chapter 2, “Introduction” lists the terms of use, software requirements, and supported operating systems.
- Chapter 3, “CDMI Architecture” describes the components of the reference implementation and the theory of operations.
- Chapter 4, “CDMI Capabilities” describes the capabilities of the reference implementation.
- Chapter 5, “CDMI Reference Implementation Design” describes how to perform typical tests and how to add your own tests.
- Chapter 6, “CDMI Error Code Mapping” maps the error codes in the reference implementation to JavaExceptions.

References

The following documents may be helpful when applying the concepts and/or following the steps in this developer’s guide.

- [REST] - “Representational State Transfer” - http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- [RESTful Web] - Richardson, Leonard and Sam Ruby, *RESTful Web Services*, O'Reilly, 2007.

- [HTTP-RESPONSE] Hypertext Transfer Protocol – HTTP/1.1, Chapter 10 Status Code Definitions - <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>
- [URL-Encoding] URL Safe character encoding <http://www.w3.org/TR/html40/appendix/notes.html#non-ascii-chars>. This mechanism is available for Java in the classes URLEncoder and URLDecoder.
- Related third-party web site references. This document references third-party URLs that provide additional, related information.

Note: SNIA is not responsible for the availability of third-party web sites mentioned in this document. SNIA does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. SNIA will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Additional Information

The SNIA web site provides additional information about the SNIA Cloud Storage Initiative at <http://www.snia.org/cloud>.

Conventions

Table 1 describes the typographic conventions that are used in this document.

Table 1 – Typographic Conventions

Convention	Description
Fixed-width text	The names of commands, files and directories, and on-screen computer output
Bold, fixed-width text	What you type, contrasted with on-screen computer output
<i>Italicized text</i>	Variables, field names, and book titles
Note:	Additional or useful informative text.
WARNING:	Indicates that you should pay careful attention to the probable action, so that you may avoid system failure or harm.

SNIA Welcomes Your Comments

SNIA is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by sending an e-mail to tcmd@snia.org.

Chapter 2: Introduction

This chapter provides an introduction to the SNIA *CDMI Reference Implementation - Developer's Guide*.

The following topics are discussed:

- Terms of Use
- Supported Operating Systems
- Software Requirements

Terms of Use

This developer's guide is released to you under the following copyright notice:

"Copyright © 2010 Storage Networking Industry Association. Use is subject to license terms."

Supported Operating Systems

The reference implementation is written in Java and should work on any platform that supports Java version 1.6 and above.

Software Requirements

Installing Dependencies

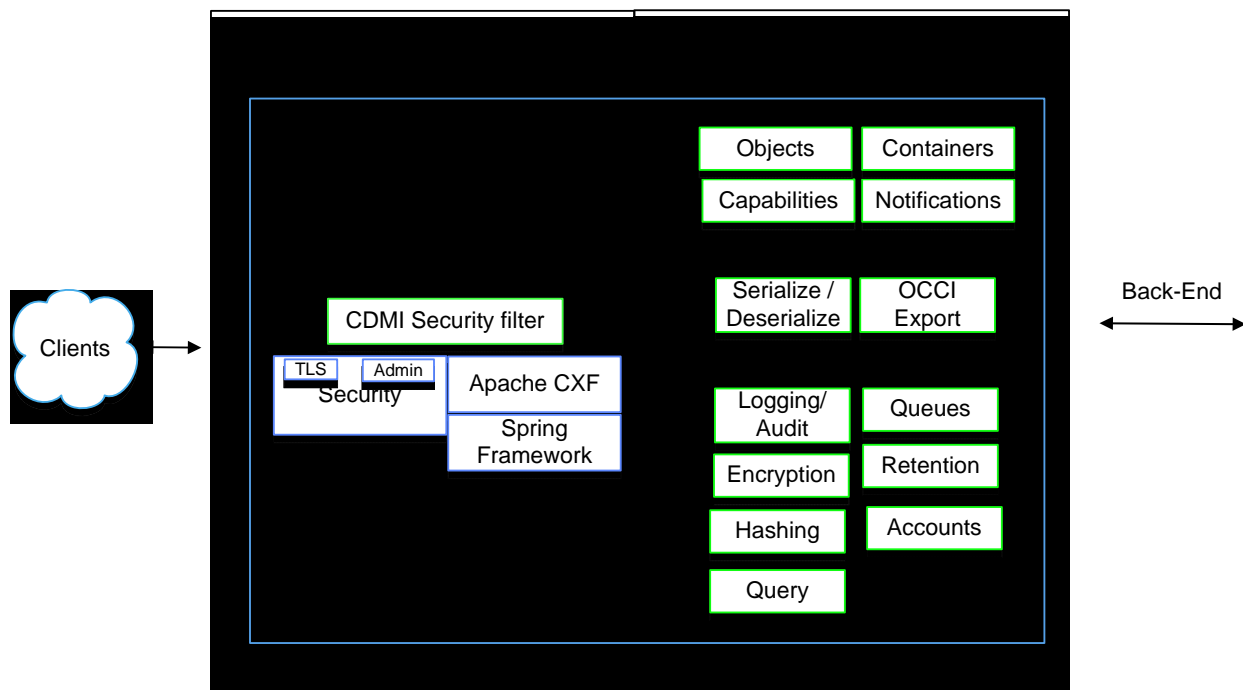
- 1** Install version 1.6 of Java.
- 2** Download and install version 2.0.9 of Maven. See <http://maven.apache.org/> for details.
- 3** Install an IDE such as NetBeans or Eclipse if desired.

Chapter 3: CDMI Architecture

This chapter provides information about the architecture of a Cloud storage system.

Architecture Overview

Figure 1 and Figure 2 show the relationships of the front-end server, the mid-layer, and the back-end store of the CDMI architecture reference implementation.



Green: SNIA-developed code
 Blue: 3rd-party code

Figure 1 – CDMI Architecture: Front-End Server and Mid-Layer

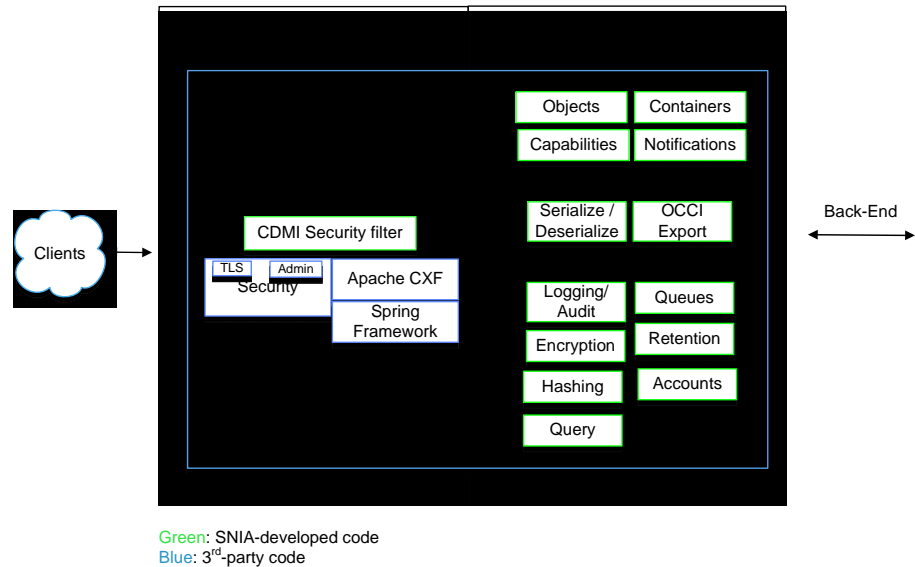


Figure 2 – CDMI Architecture: Mid-Layer, Back-End Store, and Other Clouds

File System Naming

The file system for the CDMI architecture contains the following naming conventions:

- 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

Examples of file names for containers, container metadata, and data object data and metadata include the following:

- 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 objects data: `/mnt/cdmi server/MyContainer/0000706D0010B84FAD185C425D8B537E`
- Data object's metadata: `/mnt/cdmi server/MyContainer/.0000706D0010B84FAD185C425D8B537E`

Theory of Operations

This reference implementation is based on the Apache CXF implementation of JAX-RS, which gets along well with Spring, so it also uses Spring dependency injection design patterns for all of the application pieces. The various packages contain the following:

(1) `org.snia.cdmiserver.dao`

The DAO interfaces for the various CDMI data types.

(2) `org.snia.cdmiserver.dao.filesystem`

The implementations of the DAO interfaces, using an ordinary filesystem base directory to store things in, and modelling containers as directories and data objects as files. For each resource implementation, there is a corresponding bean in the Spring configuration file: (`applicationContext.xml`).

(3) `org.snia.cdmiserver.exception`

RuntimeException implementations that a JAX-RS resource method is allowed to throw, which are converted (by an exception mapper; see below) into an appropriate HTTP response with the right status code. For example, if a resource method throws `NotFoundException`, a 404 response will be generated.

If you add any additional exception classes here, you will also need to add a new exception mapper class into `org.snia.cdmiserver.provider` (and add it to the Spring configuration).

(4) `org.snia.cdmiserver.filter`

Originally contains two Jersey-specific filters, to enforce validation of the "X-CDMI-Specification-Version" header matching a version supported by this server, and a security filter to do authentication and authorization. You'll likely want to find CXF-compatible replacements for these two functions, since they need to apply to basically all of the resource methods.

(5) `org.snia.cdmiserver.model`

Simple JavaBean classes for the various CDMI data structures. For each one, you'll need a provider class to support conversion to/from JSON, following the exact rules in the spec. Only the `Capability`, `Container`, and `DataObject` classes have been fleshed out so far.

(6) `org.snia.cdmiserver.provider`

JAX-RS exception mappers for all the exception classes that you want to allow a resource method to throw (and then get converted into an appropriate HTTP status) and providers for converting CDMI data structures to/from JSON. As you add new exception mapper or provider implementations, don't forget to add configuration data in the Spring configuration file (`applicationContext.xml`).

(7) `org.snia.cdmiserver.resource`

JAX-RS resource classes for the container and data object related calls. Because of the open-ended way the spec defines mapping URIs to methods,

you'll end up (in PathResource) sometimes having to deal with the entire path (which is passed in as a method parameter).

Note: Resource classes should be declared with `scope="prototype"`, meaning you'll get a new one for each request. Otherwise, the logic in these classes won't be threadsafe.

(8) `org.snia.cdmiserver.util`

A spot for generic utility classes.

Chapter 4: CDMI Capabilities

This chapter lists the following capabilities of the CDMI Reference Implementation:

- Cloud Storage System-Wide Capabilities
- Storage System Metadata Capabilities
- Data System Metadata Capabilities
- Data Object Capabilities
- Container Capabilities
- Domain Capabilities
- Queue Object Capabilities

Cloud Storage System-Wide Capabilities

Table 2 lists the system-wide capabilities that are supported in this reference implementation of a cloud storage system. These capabilities, which are found in the capabilities object, are referred to by the root URI (root capabilities).

Table 2 – Cloud Storage System-Wide Capabilities

Capability	Implemented?
cdmi_domains	No
cdmi_export_occi_iscsi	No
cdmi_metadata_maxitems	Yes
cdmi_metadata_maxsize	Yes
cdmi_notification	No
cdmi_query	No
cdmi_queues	No

Table 2 – Cloud Storage System-Wide Capabilities

Capability	Implemented?
cdmi_security_audit	No
cdmi_security_data_integrity	No
cdmi_security_encryption	No
cdmi_security_https_transport	No
cdmi_security_immutability	No
cdmi_security_sanitization	No
cdmi_serialization_json	No

Storage System Metadata Capabilities

Table 3 lists the storage system metadata capabilities that are supported in this reference implementation of a cloud storage system. These capabilities are found in the capabilities objects for domains, data objects, containers, and queues.

Table 3 – Storage System Metadata Capabilities

Capability	Implemented?
cdmi_acl	No
cdmi_acount	No
cdmi_billingsize	No
cdmi_ctime	Yes - From filesystem
cdmi_hash	No
cdmi_mcount	No
cdmi_mtime	Yes - From filesystem
cdmi_size	Yes - From filesystem

Data System Metadata Capabilities

Table 4 lists the capabilities for data system metadata that are supported in this reference implementation of a cloud storage system. These capabilities are found in the capabilities objects for domains, data objects, containers, and queues.

Table 4 – Data System Metadata Capabilities

Capability	Implemented?
cdmi_data_autodelete	No
cdmi_data_dispersion	No - Unset. Will not be implemented.
cdmi_data_holds	No

Table 4 – Data System Metadata Capabilities

Capability	Implemented?
cdmi_data_redundancy	Yes - Always set to 1
cdmi_data_retention	No
cdmi_encryption	No
cdmi_infrastructure_redundancy	Yes - Always set to 1
cdmi_max_latency	No
cdmi_max_throughput	No
cdmi_RPO	No - Unset. Will not be implemented.
cdmi_RTO	No - Unset. Will not be implemented.
cdmi_sanitation_method	No
cdmi_value_hash	No

Data Object Capabilities

Table 5 lists the capabilities for data objects that are supported in this reference implementation of a cloud storage system.

Table 5 – Data Object Capabilities

Capability	Implemented?
cdmi_delete_dataobject	Yes - Always true
cdmi_deserialize_dataobject	No
cdmi_modify_metadata	Yes - Always true
cdmi_modify_value	Yes - Always true
cdmi_modify_value_range	No
cdmi_read_metadata	Yes - Always true
cdmi_read_value	Yes - Always true
cdmi_read_value_range	No
cdmi_serialize_dataobject	No

Container Capabilities

Table 6 lists the capabilities for containers that are supported in this reference implementation of a cloud storage system.

Table 6 – Container Capabilities

Capability	Implemented?
cdmi_copy_container	No
cdmi_create_container	Yes - Always true
cdmi_create_dataobject	Yes - Always true
cdmi_create_queue	No
cdmi_create_reference	No
cdmi_delete_container	Yes - Always true
cdmi_list_children	Yes - Always true
cdmi_list_children_range	No
cdmi_modify_metadata	Yes - Always true
cdmi_move_container	No
cdmi_post_dataobject	Yes - Always true
cdmi_read_metadata	Yes - Always true
cdmi_serialize_container	No
cdmi_snapshot	No

Domain Capabilities

Not implemented in this release

Queue Object Capabilities

Not implemented in this release

Chapter 5: CDMI Reference Implementation Design

Introduction

The CDMI reference implementation is an example implementation of the CDMI specification.

As of this writing, not every feature is implemented. Currently, only the following features are implemented:

- Create/delete container
- List container objects
- Create/delete object
- Retrieve object
- Update data object by range
- Retrieve data object by range
- Retrieve object by ID

Where Can I Find the Code?

You can find the reference implementation in subversion under `snia-cloud/trunk/java-ref-impl/snia-cdmi`.

Technologies Used

- Java
- CXF for REST library
- Spring for dependency management
- Maven for build management
- SLF4J for logging
- Jackson for JSON parsing

Using the SPI to Implement Your Own Storage Adapter

The reference implementation ships with a filesystem storage adapter. The storage adapter architecture is such that other implementers of the CDMI specification may use the Service Provider Interface (SPI) to implement their own storage adapter.

Figure 3 shows the class hierarchy.

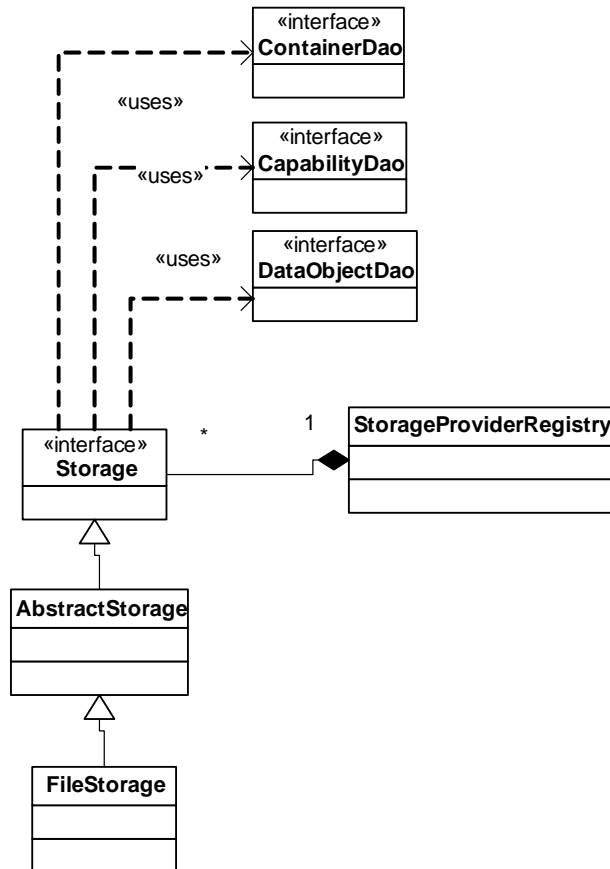


Figure 3 – Class Hierarchy

Creating a New Storage Adapter

To create a new storage adapter, do the following:

- 1 Create a new Maven project for the new storage adapter.
- 2 Ensure the POM has the appropriate dependencies.
- 3 Create a new storage-derived class.
- 4 Create new implementations for the following interfaces:
 - `CapabilityDao`
 - `ContainerDao`
 - `DataObjectDao`
 - `ObjectIdMapperDao`

- 5 Create a new Spring config file (i.e., `applicationContext.xml`) for your storage adapter.
- 6 Select a prefix in `Storage.getStorageType()`.
- 7 Use the prefix you selected for each of the relevant bean names in `applicationContext.xml`.
- 8 Add a dependency in the `snia-cdmi-webapp pom.xml` for your storage adapter

Responding to a Request Sequence

When requests come in from customers, a likely architecture would have the customer certificate queried in an external data store to determine from which customer the request came.

In the pluggable architecture, this customer information may include a “storage type.” For the reference implementation, a simpler route was taken. Rather than query a certificate, the client is expected to provide a header “X-Storage” which indicates the storage type to use. Then requests are made using that storage provider.

The example sequence diagram in Figure 4 shows how a container operation may be performed.

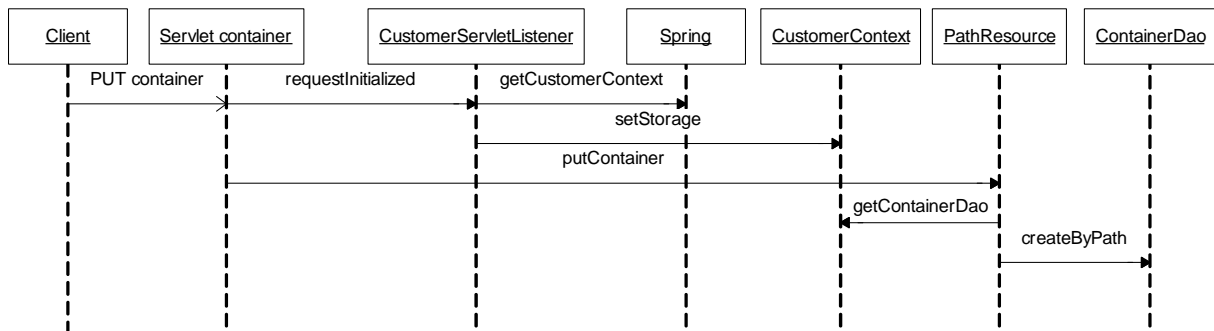


Figure 4 – Example Sequence Diagram

Logging

SLF4J is used as the reference implementation’s logging framework because it is logging subsystem agnostic.

Building and Deploying

Simply build with Maven from the top level, and then copy the resulting WAR file into the web resource directory of a servlet container (such as Tomcat).

The filesystem storage adapter example expects a directory named `/data` to be present and writable.

Testing

The class “CDMltest” has a simple test against basic working operations.

Chapter 6: CDMI Error Code Mapping

The following Java exceptions should be mapped to HTTP status codes (see Table 7).

Table 7 – Error Code Mapping

Java Exception	HTTP Status Code
BadRequestException	400 - Bad Request
ConflictException	409 - Conflict
ForbiddenException	403 - Forbidden
NotFoundException	404 - Not Found
UnauthorizedException	401 - Unauthorized