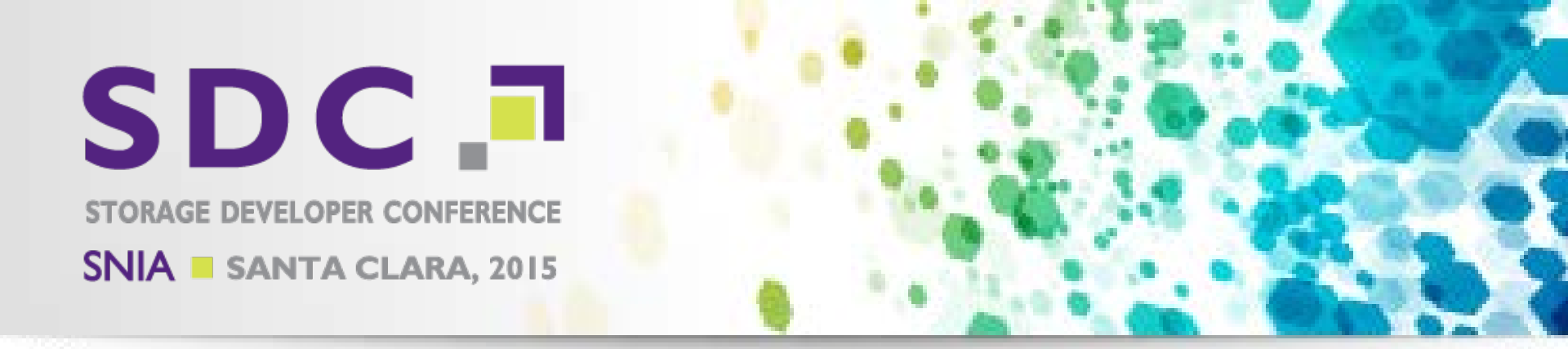



# How to test CDMI extension feature like LTFS, Data Deduplication, and OVF, partial – value copy functionality: Challenges, Solutions and Best Practice?”

**Sachin Goswami**
**TATA Consultancy Services**

# Abstract

The Cloud Storage space has been outperforming the industry expectations as is evident in several Industry report. SNIA provided Cloud Data Management Interface (CDMI) specification is increasingly being adopted as a standard across the cloud.

The popularity of the CDMI specification can be judged by the present cloud storage market being flooded with CDMI server based products offered by many big and small cloud storage vendors. SNIA is constantly upgrading the specification and has included extensions like Data Deduplication, OVF, LTFS for cloud storage and others.

The SNIA Cloud Storage Technical Workgroup has been unceasingly working to address all storage challenges that exist in the storage domain. It is striving to provide support and solution for Data Deduplication, Open virtualization format (OVF), Partial Upload, server side partial value copy and LTFS as a primary cloud storage, managing Latency as well as backup and archival solution.

TCS is focusing on maturing the Conformance Test Suite and SNIA CDMI Conformance Test Specification by adding more enhancements.

In this proposal we will share the approach TCS will adopt to overcome the challenges in testing of LTFS integration with CDMI, Data Deduplication, partial upload on Server and Open Vitalization format (OVF) of CDMI and Non-CDMI based scenarios of the cloud products.

Additionally, we will also be sharing challenges faced / learnings gathered from testing of CDMI Products for conformance. These learnings will help serve as a ready reference for organizations developing LTFS, Data Deduplication, OVF and partial upload in CDMI, Non-CDMI based product suite.

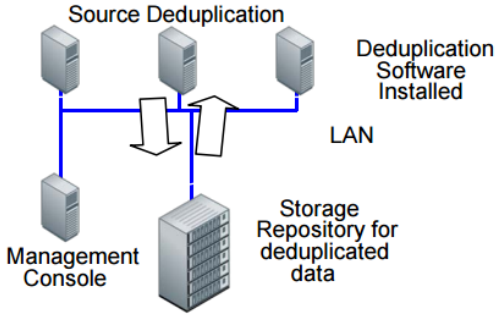
# Agenda

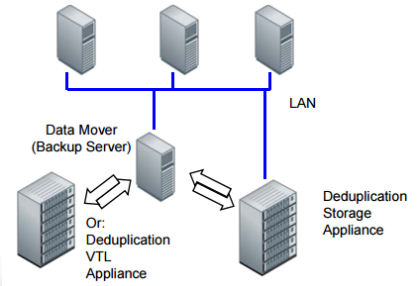
- How Data deduplication works in CDMI
  - Challenges & Best Practices
  - Use Cases
  - Test Case Scenarios
- How LTFS works in CDMI
  - Challenges & Best Practices
  - Use Cases
  - Test Case Scenarios
- How OVF works in CDMI
  - Challenges & Best Practices
  - Use Cases
  - Test Case Scenarios
- Questions
- Contact Details

# Data Deduplication

- Data deduplication Known as “De-dup” .
- Removal of redundant data.
- **Industry Segregation**
  - **Source – based De duplication or Client De duplication**

    Client software or agent installed at the de duplication point.
    - Process to verify new or changed data/file
    - No change , same reference
    - Unique data sent to central repository
  - **Target-based de duplication**
    - Stand-alone central repository for data
    - De-Dup data as internal process as in-line , post-process or both

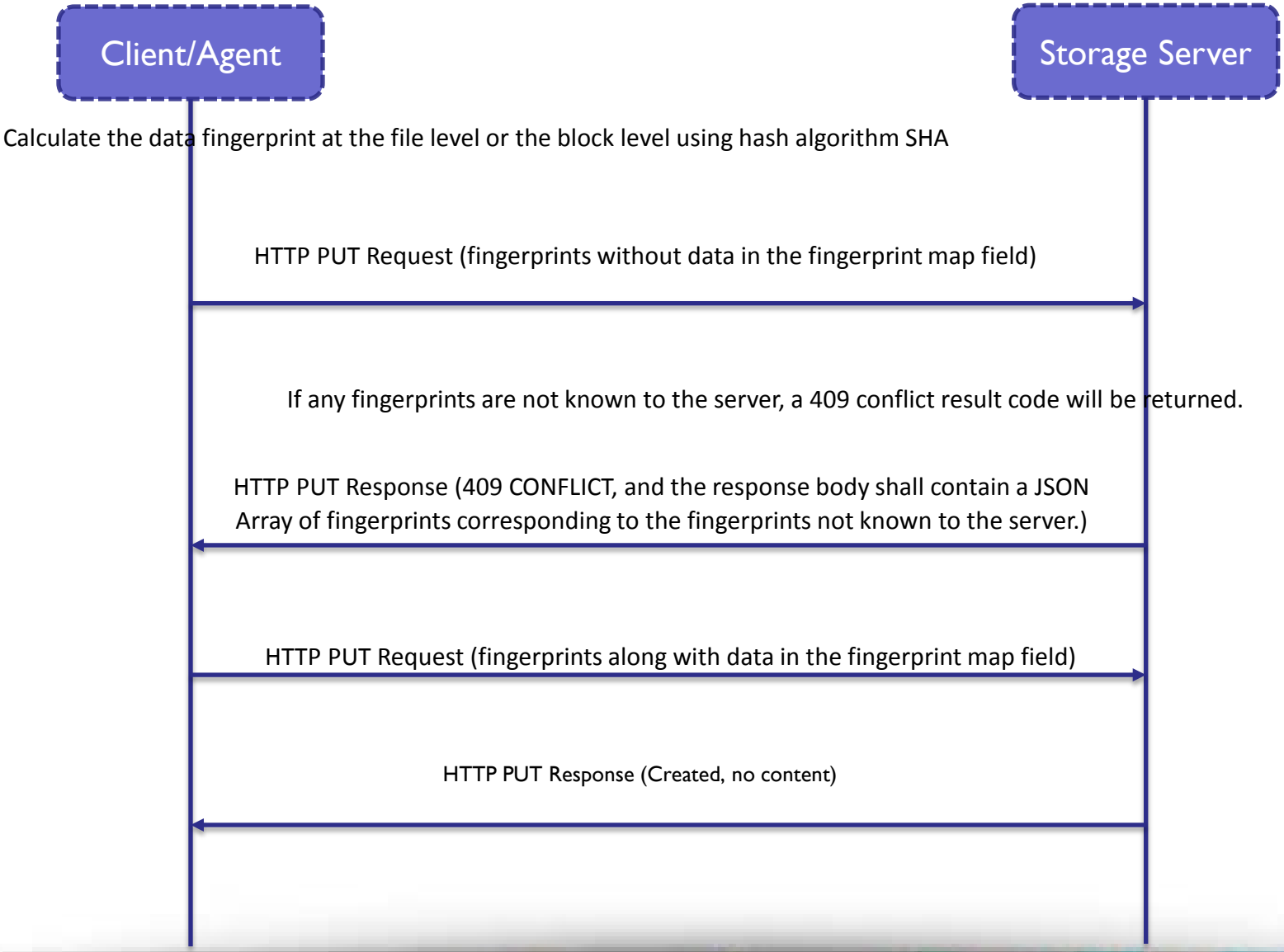


# CDMI De-duplication

- Is a capacity optimization technologies that is used to drastically improved storage efficiency.
- Is the process of eliminating redundant copies of data.
- Any algorithm that searches for duplicate data objects ,e.g. blocks ,chunks and files and store a single copy of those objects.

# CDMI Deduplication - Data Flow






Reference : (CDMI Extension )

# CDMI De–Dup Challenges & Best Practice

**Challenges**

- Figure out best algorithm- based on Hashing .
- Single or multiple hashing algorithm supports.
- De – Duplication Supported CDMI Server availability.

**Best Practice**

- **Cost:** Reduced the cost of what your system will need in order to store the data on storage/tape.
- **Tape:** If you are migrating all of your data to tape on a monthly basis or lower frequency, then the need for de-duplication.
- **Bandwidth :** As per CDMI De-dup specification , user first check availability of finger print of data on the CDMI server.

# CDMI De-Dup Test Cases Scenario's

| | |
|---|---|
| **Create** | 1. Create finger print on data (data object, chunks, file, blocks) which is unknown to the CDMI server. **-> fingerprints on data created on client side.**<br>2. Create finger print on data (data object, chunks, file, blocks) which is known to the CDMI server partially fingerprint **- > 409 conflict with JSON Array and unknown fingerprints as response code.**<br>3. Create finger print on data (data object, chunks, file, blocks) which is known to the CDMI server complete finger print- **> zero-length value fingerprints, nothing returned.**<br>4. Create Object and it should be stored on CDMI server if non zero length value finger prints with response code 409 conflict**->zero length value fingerprints with 409 conflict returned.** |
| **Read** | 1. Read object with the help of finger prints available on CDMI Client. |
| **Delete** | 1. Delete object with known finger print.<br>2. Delete Object wrong finger print which is not available on CDMI Server. |

# Test Execute Step-*CDMI_DataObject_Create_DD_2C_1*

| | |
|---|---|
| **CDMI Specification Reference** | **Reference:** Data Deduplication Metadata Extension |
| **Test Specification Id** | DD_2C_1 |
| **Test Case Name** | dataObjectCreate_DD_2C_1 |
| **Test Case Description** | To create a data object using fingerprints of possible previously stored data |
| **Pre-Test Dependencies** | System-wide capability: cdmi_dataobjects<br><**Reference:** CDMI Specification v1.0.2 – Section: **12.1.1**><br>Capability in parent container: cdmi_create_dataobject<br><**Reference:** CDMI Specification v1.0.2 – Section: **8.2.3**> |

## Description

<Test Case : Start>

1. Set valid login credentials. <**Reference:** Test Spec Id: **1C_L_1**>
2. Create a container named "TestContainer1" <**Reference:** Test Spec Id: **1C_1**>
3. Request URL: <**Reference:** Data Deduplication Metadata Extension>
   Request Headers: <**Reference:** Data Deduplication Metadata Extension>
   Request Body:<**Reference:** Data Deduplication Metadata Extension>

   Request Sample with Body:

```
PUT <root URI>/TestContainer1/data_object1 HTTP/1.1
Host: cloud.example.com

Content-Type: text/plain;charset=utf-8
Content-Length: 37
Transfer-Encoding: chunked

0;fingerprint=SHA256:86e1de74820a9b252ba33b2eed445b0cd02c445b5f4b800720
5af f1762d7301a
0;fingerprint=SHA256:30e70dda3fb3acd5aafd3e6426613247f2c88b2384ad048ad7
18f 5520f7b2460
```

# Test Execute Step-*CDMI_DataObject_Create_DD_2C_1 ……*

4. Verify response code, if its 409:
   a. Extract unknown fingerprints from response.
   b. PUT to create object and register unknown fingerprints

```
PUT <root URI>/TestContainer1/data_object1 HTTP/1.1
Host: cloud.example.com
Content-Type: text/plain;charset=utf-8
Content-Length: 37
Transfer-Encoding: chunked

4;fingerprint=SHA256:86e1de74820a9b252ba33b2eed445b0cd02c445b5f4b
8007205af f1762d7301a
This
33;fingerprint=SHA256:30e70dda3fb3acd5aafd3e6426613247f2c88b2384a
d048ad718 f5520f7b2460
 is the Value of this Data Object
```

5. Verify that all mandatory fields exist: <**Reference:** CDMI Specification v1.0.2 – Section: **8.2.7**>
   List of mandatory fields for "data_object1":
   - objectType
   - objectID
   - objectName
   - parentURI
   - domainURI (**Conditional:** If system-wide capability "cdmi_domains" is present and true)
   - parentID
   - capabilitiesURI
   - metadata
   - completionStatus
   - mimetype
6. Check for HTTP status code: **201 Created** returned.
   <**Reference:** CDMI Specification v1.0.2 – Section: **8.2.8**>
7. Expected Result: Data object "data_object1" should be created.
8. Container cleanup process: <**Reference:** 6.5.1 Container Cleanup>

<Test Case : End>

# Linear Tape file system - LTFS

- Vendor Neutral - Removing dependencies From proprietary software and tape formats.
- Method of shaping data on tape .
- Empowers tape as a file-based storage medium, without attaching metadata or modifying your files.
- Simplifies /quick file access .
- Enable data partitioning for faster access
- Easily integrate with existing application
- Reduces storage cost

**How it works**

- LTFS partitions LTO-6/LTO-5 tapes into two segments called partitions. Partition 0 holds directory structures and pointers that let the tape drive quickly seek specific data from the tape.
- The data itself is stored in Partition -1 applying a file system to a tape allows users to organize and search the contents of tape as they would on hard disk, improving access time for data stored on tape. LTFS makes it possible to drag and drop files to tape in the same way that files might be dragged and dropped to disk.

# CDMI LTFS

- Defines an interoperable way to store and exchange files and directories .
- Allow files to be stored on one or more LTFS volumes.
- Provides a single namespace view for random size files and file collections.
- Associating CDMI Containers and LTFS Volume
- Help to manage Latency in LTFS export
- Saving Data object on LTFS
- Saving Container object on LTFS

# CDMI LTFS Challenges & Best Practice

- As a cloud provider challenges
  - Protect against data lost.
  - Bulk data transfer between/into/out of cloud.
- Best Practice
  - Require many PB's of archival data, LTFS works.
  - Require to physically move data around , LTFS works.
  - Require to persist data , LTFS works.

# LTFS CDMI Testing Scenarios

| | |
|---|---|
| **Create** | 1. Create a CDMI Container and associate with LTFS volume.<br>2. Create a CDMI container and associate with set of LTFS volumes.<br>3. Create an object on the container mapped with LTFS volume. |
| **Update** | 1. Move a object from Container mapped LTFS volume to different container. |
| **Delete** | 1. Delete an associated container with LTFS volume. |

# Test Execute Step-*CDMI_Container_Create_EP_1C_1*

| **CDMI Specification Reference** | **Reference:** LTFS Export Extension |
|---|---|
| Test Specification Id | EP_1C_1 |
| Test Case Name | containerCreate_EP_1C_1 |
| Test Case Description | To create a container and associate with LTFS volumes |
| Pre-Test Dependencies | Capability in parent container: cdmi_create_container<br><Reference: CDMI Specification v1.0.2 – Section: 9.2.3> |

## Description

<Test Case : Start>

1. Set valid login credentials. <**Reference:** Test Spec Id: **1C_L_1**>
2. Request URL: <**Reference:** LTFS Export Extension >
   Request Headers: <**Reference:** LTFS Export Extension >
   Request Body:<**Reference:** LTFS Export Extension >

   Request Sample with Body:

```
PUT <root URI>/TestContainer1/ HTTP/1.1
Host: cloud.example.com

X-CDMI-Specification-Version: 1.0.2
Content-Type: application/cdmi-container
Accept: application/cdmi-container
```

# Test Execute Step-*CDMI_Container_Create_EP_1C_1……*

```
{
 "exports" : {
   "ltfs" : {
     "ltfs_volume_uuids" : [
       "1F912610-3F48-43F3-A53A-D0761B0238DE",
       "0A198010-740E-433B-920F-0CC95CDD0C7F",
       "5573B072-FFF7-408A-A599-3FC383E72DDC",
       "6DECAAD5-9507-4052-876A-F45B1CE1F2AA"
     ],
     "usermap" : {
        {"myuser", "<-", "*"}
     },
     "groupmap" : {
        {"mygroup", "<-", "*"}
     },
     "domainmap" : {
        {"/cdmi_domains/mydomain/", "<-",
"http://sourcecloud.example.com/cdmi_domains/source/"}

    }
   }
  }
}
```

3. Check for HTTP status code: **201 Created** returned.
   <**Reference:** CDMI Specification v1.0.2 – Section: **9.2.8**>
4. Expected Result: Container " TestContainer1/" should be created successfully and all LTFS files and objects stored on the specified volumes will be accessible through the exported container
5. Container cleanup process: <**Reference:** 6.5.1 Container Cleanup>

<Test Case : End>

# Open Virtualization Format (OVF)

- DMTF Specification describes an open, secure, efficient and extensible format for the packaging and distribution of software to be run in virtual systems.
- OVF package enables the authoring of portable virtual systems and the transport of virtual systems between virtualization platforms.
- Open Virtualization Format (OVF) standard for packaging and delivering virtual machines (VMs).

# CDMI OVF

- DMTF Open Virtualization Format (OVF) standard allows a virtual appliance to be serialized from one platform and deserialized onto another platform.
- OVF package can be created as a CDMI object by using the TAR file as the value of the object.

# OVF Benefits

- Optimized for distribution
- Optimized for a simple, automated user experience
- Portable and packaging
- Vendor and platform independent
- Extensible
- Localizable

# OVF CDMI Testing Scenarios

- Create OVF Format CDMI Objects  in CDMI Container.
- Serialize a OVF format CDMI Object in CDMI Container.
- De-serialize a OVF format CDMI object in CDMI Container.

# Questions ?

Mail us @

sachin.goswami@tcs.com

Special Thanks to Gajanan Kamble & Udayan Singh for making this presentation possible

# References


- http://www.snia.org/sites/default/education/tutorials/2008/fall/dataprot_mng/MattBrisse-GideonSenderov-Data_Deduplication_DDSR-SIG.pdf
- LTFS References:http://www.snia.org/ltfs
- http://www.snia.org/sites/default/orig/SDC2013/presentations/Cloud/DavidSlick_LTFS_and_CDMI-R3.pdf.
- http://www.storage-switzerland.com/Blog/Entries/2013/2/19_Use_Cases_for_Enterprise_Dedupe.html