Extended Child Listing CDMI Extension

Version 2.0

ABSTRACT: This CDMI Extension is intended for developers who are considering a standardized way to add functionality to CDMI. When multiple compatible implementations are demonstrated and approved by the Technical Working Group, this extension will be incorporated into the CDMI standard.

This document has been released and approved by the SNIA. The SNIA believes that the ideas, methodologies, and technologies described in this document accurately represent the SNIA goals and are appropriate for widespread distribution. Suggestion for revision should be directed to http://www.snia.org/feedback/.

SNIA Working Draft

March 12, 2021

USAGE

DISCLAIMER

The information contained in this publication is subject to change without notice. The SNIA makes no warranty of any kind with regard to this specification, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this specification.

Suggestions for revisions should be directed to https://www.snia.org/feedback/.

Contents

Clause 1

Extended Child Listing CDMI Extension

1.1 Overview

CDMI clients often list the contents of a container for display or analysis purposes (e.g. computing the total space consumed). These operations often require metadata about each object within a container, thus the CDMI client must obtain this metadata for each child object. CDMI currently offers two ways to do this: a) perform a GET for each CDMI Object within the container, and b) use CDMI queries to perform a query for all objects within the container, and specifying which metadata should be returned for each result of the query.

There are two downsides for these approaches: Performing a GET for each object introduces high aggegate latencies and consumes significant resources, and support for CDMI queries is a high development effort and is not implemented by all CDMI servers. Introducing an extension to CDMI Child Listing addresses the resource inefficiencies and is much simpler to implement.

This extension adds the following functionality:

- A new query parameter format is added that allows a CDMI client to specify which fields and metadata they would like to be returned for each child object. These requested fields are returned in the children array in the response body.
- A new query paramater is added to indicate if child listing is recursive. Recursive listings are returned in the children array in the response body.

1.2 Instructions to the Editor

To merge this extension into the CDMI 2.0.0 specification, make the following changes:

1. Add an entry after the “cdmi_list_children_range” entry in the table starting on line 668 of cdmi_advanced/cdmi_capability_object.txt, as follows:

Table 1: Capabilities for container objects

Capability name	Type	Definition
`cdmi_list_children_extended`	JSON string	If present and “`true`”, indicates that the cloud storage system supports requesting specific fields and metadata for children.
`cdmi_list_children_recursive`	JSON string	If present and “`true`”, indicates that the cloud storage system supports requesting children be listed recursively.

2. Add entries to the synopsis section starting on line 6 of cdmi_core/cdmi_container_object/read.txt, as follows:
   - `GET <root URI>/<ContainerName>/<TheContainerName>/?children=!&...`
   - `GET <root URI>/cdmi_objectid/<ContainerObjectID>/?children=!&...`
   - `GET <root URI>/<ContainerName>/<TheContainerName>/?children=!<range>&...`
   - `GET <root URI>/cdmi_objectid/<ContainerObjectID>/?children=!<range>&...`

- GET <root URI>/<ContainerName>/<TheContainerName>/?children=[<fieldname>,<fieldname>,...]&...
- GET <root URI>/cdmi_objectid/<ContainerObjectID>/?children=[<fieldname>,<fieldname>,...]&...
- GET <root URI>/<ContainerName>/<TheContainerName>/?children=![<fieldname>,<fieldname>,...]&...
- GET <root URI>/cdmi_objectid/<ContainerObjectID>/?children=![<fieldname>,<fieldname>,...]&...

Where:

- "!" is a indicator that recursive listing shall be performed.
- <fieldname> can contain "/" characters, which indicate a sub-field within a JSON Object. For example "metadata/cdmi_size" indicates the "cdmi_size" sub-field within the "metadata" field. If a field name includes a "/" character, it must be escaped with a "".

3. Add an entry after the "cdmi_list_children_range" entry in the table starting on line 35 of cdmi_core/cdmi_container_object/read.txt, as follows:

Table 2: Capabilities - Read a CDMI Container Object using CDMI

Capability	Location	Description
cdmi_list_children_extended	Container object	Ability to list requested specific fields and metadata for children
cdmi_list_children_recursive	Container object	Ability to list the children of an existing container object recursively

3. Update "children" entry in the table starting on line 128 of cdmi_core/cdmi_container_object/read.txt, as follows:

Table 3: Response message body - Read a container object using CDMI

Field Name	Type	Description	Requirement
children	JSON array of JSON strings or JSON array of JSON strings and JSON Arrays or JSON array of JSON arrays of JSON Strings or JSON array of JSON arrays of JSON Strings and JSON Arrays	Names of the children objects in the container object. When a client uses a child name in a request URI or a header URI, the client shall escape reserved characters according to RFC 3986 [rfc3986], e.g., a "%" character in a child name shall be replaced with "%25". • Children that are container objects shall have "/" appended to the child name. • Children that are references shall have "?" appended to the child name. When extended children listing is requested by specifying an array of field names to return for each child, a JSON array shall be returned for each child, with the contents of the array being JSON Strings for each requested field name, in the order requested. When recursive children listing is requested by specifying an "!", a JSON array shall be returned after each container, with the contents of the array being JSON Strings and JSON arrays for each child of each container.	Mandatory

4. Add the following examples to the end of the examples section in cdmi_core/cdmi_container_object/read.txt, as

follows:

EXAMPLE 5: GET to the container object URI to read the name, size, and creation date of each child:

```
--> GET /cdmi/2.0.0/MyContainer/?parentURI&children=[objectName,metadata/cdmi_size,
↪metadata/cdmi_ctime] HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-container

<-- HTTP/1.1 200 OK
<-- Content-Type: application/cdmi-container
<--
<-- {
<--   "parentURI" : "/",
<--   "children" : [
<--     ["red", "7823683", "2021-02-02T12:42:31.237849Z"],
<--     ["green", "23834", "2021-02-02T12:42:32.185734Z"],
<--     ["yellow", "15", "2021-02-02T12:42:33.178433Z"],
<--     ["orange/", null, "2021-02-02T12:42:35.746234Z"],
<--     ["purple/", "null, "2021-02-02T12:42:35.927473Z"]
<--   ]
<-- }
```

EXAMPLE 6: GET to the container object URI to read the parentURI and recursively read the children of the container object:

```
--> GET /cdmi/2.0.0/MyContainer/?parentURI&children=! HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-container

<-- HTTP/1.1 200 OK
<-- Content-Type: application/cdmi-container
<--
<-- {
<--   "parentURI" : "/",
<--   "children" : [
<--     "red",
<--     "green",
<--     "yellow",
<--     "orange/",
<--     [
<--       "purple",
<--       "brown"
<--     ],
<--     "purple/",
<--     [
<--       "blue"
<--     ]
<--   ]
<-- }
```

EXAMPLE 7: GET to the container object URI to read the parentURI and recursively read the name, size, and creation date of each child:

```
--> GET /cdmi/2.0.0/MyContainer/?parentURI&children=![objectName,metadata/cdmi_size,
↪metadata/cdmi_ctime] HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-container

<-- HTTP/1.1 200 OK
<-- Content-Type: application/cdmi-container
<--
<-- {
<--   "parentURI" : "/",
<--   "children" : [
<--     ["red", "7823683", "2021-02-02T12:42:31.237849Z"],
<--     ["green", "23834", "2021-02-02T12:42:32.185734Z"],
<--     ["yellow", "15", "2021-02-02T12:42:33.178433Z"],
<--     ["orange/", null, "2021-02-02T12:42:35.746234Z"],
<--     [
```

(continues on next page)

(continued from previous page)

```
<--       ["purple", "9237394", "2021-02-02T12:42:36.847563Z"],
<--       ["brown", "1253", "2021-02-02T12:42:37.827643Z"]
<--     ],
<--     ["purple/", null, "2021-02-02T12:42:35.927473Z"],
<--     [
<--       ["blue", "48733", "2021-02-02T12:42:36.783632Z"]
<--     ]
<--   ]
<-- }
```