Validators CDMI Extension

Version 2.0a

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.

SNIA Working Draft

September 1, 2023

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

Validators CDMI Extension

1.1 Overview

Some CDMI systems allow validation to be performed against CDMI objects. In such a system, multiple validations may be performed simultaneously against the same or multiple objects. In these systems, the client receives an indication of validation failure on object creation or modification, with validation success either resulting in normal HTTP responses being returned, or an indication of validation failure annotated as object metadata.

This extension proposes a new type of data object to define validations on object creation and modification. The validation data object (extended in a similar manner as a query queue object) may be used to define validations independently from the objects on which the validator is acting.

Validating existing objects is performed using CDMI jobs.

1.2 Instructions to the Editor

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

1.2.1 terms.txt

Insert into preamble/terms.txt, as follows:

x.1
validator
: **|br|** a data object with specific metadata that defines and manages validation operations performed against matching newly created and updated CDMI objects (validation targets) **|br|**

x.2
validation operation
: **|br|** the process of evaluating a validation schema against a validation target **|br|**

x.3
validation schema
: **|br|** metadata that describes the organization and format of CDMI objects **|br|**

x.4
validation scope
: **|br|** metadata that defines which validation targets validation operations are performed against. **|br|**

x.5
validation targets
: **|br|** the set of CDMI objects against which validation operations are performed **|br|**

1.2.2 cdmi_capability_object.txt

Add an entry to the end of the table starting on line 135 of cdmi_advanced/cdmi_capability_object.txt, as follows:

Table 1.1: System-wide capabilities

Capability name	Type	Definition
`cdmi_validators`	JSON string	If present and "`true`", indicates that the CDMI server supports validation data objects.
`cdmi_validators_global_container`	JSON string	If present, contains the URI for the container for all validator data objects in the CDMI server.

Add an entry to the end of the table starting on line 612 of cdmi_advanced/cdmi_capability_object.txt, as follows:

Table 1.2: Capabilities for data objects

Capability name	Type	Definition
`cdmi_validator_schema_formats`	JSON strings	If present, contains a list of schema formats that may be specified a validator data objects. Schema formats are media types as specified in RFC 6838. Currently defined schema formats include: `application/schema+json`

Add an entry to the end of the table starting on line 662 of cdmi_advanced/cdmi_capability_object.txt, as follows:

Table 1.3: Capabilities for container objects

Capability name	Type	Definition
`cdmi_create_validator_container`	JSON strings	If present and "`true`", indicates that the container allows the creation of validator container objects.
`cdmi_create_validator_dataobject`	JSON strings	If present and "`true`", indicates that the container allows the creation of validator data objects.

1.2.3 cdmi_metadata.txt

Add an entry to the end of the table starting on line 533 of cdmi_advanced/cdmi_metadata.txt, as follows:

Table 1.4: Data system metadata

Metadata name	Type	Description	Requirement
`cdmi_validation_schema_provided`	JSON array of JSON strings	When an object is validated by one or more validator object, contains the URI(s) for each matching validator object. When CDMI data object versions are supported, the URI to the version of the validator data object used for validation shall be provided.	Optional
`cdmi_validation_result_provided`	JSON array of JSON strings	When an object is validated by one or more validator object, contains the validation result for each matching validator object. Supported values are `passed` and `failed`: • `passed` - The object was validated against the schema and validation succeeded • `failed` - The object was validated against the schema and validation failed	Optional

1.2.4 validators.txt

Create new clause, “cdmi_validators.txt.txt” after existing clause 25 “Data Object Versions”, as follows.

Clause 2

Object Validators

2.1 Overview

A cloud storage system may optionally implement object validation functionality. Validator implementation is indicated by the presence of the cloud storage system-wide capabilities for validators.

Validators allow the evaluation of schemas on object creation and modification. In addition, multiple validators may perform validation actions against a single CDMI object. By creating a well-defined “validator” object, clients may define validators, specify the schema to be used to perform the validation, and specify which objects the validation is to be performed against.

Validators may be stored in container objects or may exist as standalone data objects with no parent container.

Cloud storage systems should consider implementing support for validator data objects when the system supports the following types of client-controlled activities:

- Data format consistency: If the user requires CDMI objects to conform to a given schema in order to ensure data consistency, the user may define a validator to prevent non-conformant objects. For example, this allows the user to specify that created data objects shall have a value that validates against a given schema.
- Metadata presence and values: If the user requires CDMI objects’ metadata to conform to a given schema in order to specify metadata constraints, the user may define a validator to prevent non-conformant objects. For example, this allows the user to specify that created data objects shall have a metadata value greater than one for the `cdmi_data_redundency` data system metadata.
- Limiting object types: If a user requires the limitation of what types of objects can be created, the user may define a validator to prevent the creation of non-conformant objects. For example, this allows the user to specify that created data objects shall have a mimetype that validates against a given schema.
- Limiting use of CDMI features: If a user requires the limitation of which CDMI creation and modification features are to be exposed, the user may define a validator to prevent the specification of non-desired CDMI features. For example, this allows the user to specify that created data objects cannot specify deserialization sources.

2.2 Validator Creation

Validators are CDMI objects with the following properties:

- A `cdmi_validation_scope` data system metadata item, indicating which objects the validation is performed against, and,
- A schema in the value of the data object

When a client wishes to create a validator data object, it should first check the following:

1. Check if the system is capable of providing validation functionality by checking for the presence of the `cdmi_validator` capability in the root container capabilities. If this capability is not present, creating a validator data object shall be successful, but no validation operations shall be performed.
2. Check if the system supports the schema format to be used by the validator by checking the contents of the `cdmi_validator_schema_formats` data object capability.

3. If the data object is being created in a container, check if the container is capable of providing validation functionality by checking for the presence of the `cdmi_create_validator_dataobject` for the container.

If these conditions are not met, creating a validator data object shall be successful, but no validation operations shall be performed.

Validators are created as CDMI data objects with additional metadata:

- `cdmi_validation_scope` - Indicates which objects are to be validated.
- `cdmi_validation_mark` - Indicates that all validated objects are to be marked with the result of the validation process. (optional)
- `cdmi_validation_deny` - Indicates that object creation and update requests that fail validation shall be denied. (optional)

CDMI clients may create validators through a variety of methods:

1. A client may create a validator data object without specifying the location by performing a POST operation. In this case, the system may create the validator in a validator container and return an HTTP response code of `202 Accepted`. The URI for the newly created validator data object shall be returned in an HTTP `Location` response header.
2. A client may create a validator data object at a specific location by performing a PUT operation. Only containers with a cdmi_create_validator_container capability shall allow validator data objects to be created. The semantics for creating this object are the same as for other data objects.

A client may view and access validators created by internal system processes through a CDMI container containing validator data objects. To get a list of system-created validators, clients may list the children of the container.

2.3 Validator Object Value

A validator object shall contain the schema used to perform validation. If the schema format for the data in the object value does not match against one of the schema formats listed in the `cdmi_validator_schema_formats` data object capability, the validator shall not be used.

The value of a validator object may be changed. If valid, the updated value shall be used for all subsequent validations.

2.4 Validator Object Metadata

When a client creates a validator data object, the presence of the metadata item `cdmi_validation_scope` indicates that the data object represents a validator.

The metadata of a validator object may be changed. If valid, the updated metadata shall be used for all subsequent validations.

If the `cdmi_validation_scope` metadata is removed from a validator object, the data object shall no longer be considered a validator object.

If the `cdmi_validation_scope` metadata is added to a data object, the data object shall be considered to be a validator object.

Metadata items for a validator data object are shown in Table 6:

Table 2.5: Validator object metadata

Metadata name	Type	Description	Requirement
`cdmi_validation_scope`	JSON Object	If this data system metadata item is present, it indicates which objects validations should be performed against. The format of this object is described in `%s`.	Optional

continues on next page

Table 2.5 – continued from previous page

Metadata name	Type	Description	Requirement
`cdmi_validation_mark`	JSON string	If this data system metadata item is present and "`true`", it indicates that all objects validated shall have `cdmi_validation_schema_provided` and `cdmi_validation_result_provided` data system metadata added to each object when a validation is performed.	Optional
`cdmi_validation_deny`	JSON string	If this data system metadata item is present and "`true`", it indicates that object creation or update that matches against the validation scope and fails validation shall result in a `400 Bad Reqeust` HTTP status code response.	Optional

2.5 Examples

EXAMPLE 1: Allow/Deny - Create a validator that denies the creation or update of objects in the “sandbox” container if they do not validate against the specified JSON schema:

```
--> PUT /validators/myValidator.json HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-object
--> Content-Type: application/cdmi-object
-->
--> {
-->   "valuetransferencoding" : "json",
-->   "mimetype" : "application/schema+json",
-->   "metadata" : {
-->     "cdmi_validation_scope" : [
-->       {
-->         "parentURI": "starts /sandbox/"
-->       }
-->     ],
-->     "cdmi_validation_deny" : "true"
-->   },
-->   "value" : {
-->     "$schema": "https://json-schema.org/draft/2019-09/schema",
-->     "type": "object",
-->     "required": [ "value" ],
-->     "properties": {
-->       "value": {
-->         "type": "object",
-->         "required": [ "name" ],
-->         "properties": {
-->           "name": {
-->             "type": "string"
-->           }
-->         },
-->         "additionalProperties": false
-->       }
-->     }
-->   }
--> }

<-- HTTP/1.1 201 Created
<-- Content-Type: application/cdmi-object
<--
<-- {
<--   "objectType" : "application/cdmi-object",
<--   "objectID" : "00007ED90010D891022876A8DE0BC0FD",
<--   "objectName" : "myValidator.json",
<--   "parentURI" : "/validators/",
<--   "parentID" : "00007E7F00102E230ED82694DAA975D2",
<--   "domainURI" : "/cdmi_domains/MyDomain/",
<--   "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
<--   "completionStatus" : "Complete",
<--   "mimetype" : "application/schema+json",
<--   "metadata" : {
<--     "cdmi_size" : "314",
<--     ...
<--   }
<-- }
```

EXAMPLE 2: Allow/Deny - Create a object in the sandbox container that successfully validates against the schema:

```
--> PUT /sandbox/test1.json HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-object
--> Content-Type: application/cdmi-object
-->
--> {
-->   "valuetransferencoding" : "json",
-->   "mimetype" : "application/json",
-->   "value" : {
-->     "name": "John Smith"
-->   }
--> }

<-- HTTP/1.1 201 Created
<-- Content-Type: application/cdmi-object
<--
<-- {
<--   "objectType" : "application/cdmi-object",
<--   "objectID" : "00007ED90010D891022876A8DE0BC0FD",
<--   "objectName" : "test1.json",
<--   "parentURI" : "/sandbox/",
<--   "parentID" : "00007E7F00102E230ED82694DAA975D2",
<--   "domainURI" : "/cdmi_domains/MyDomain/",
<--   "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
<--   "completionStatus" : "Complete",
<--   "mimetype" : "application/json",
<--   "metadata" : {
<--     "cdmi_size" : "24",
<--     ...
<--   }
<-- }
```

EXAMPLE 3: Allow/Deny - Create a object in the sandbox container that does not successfully validate against the schema:

```
--> PUT /sandbox/test1.json HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-object
--> Content-Type: application/cdmi-object
-->
--> {
-->   "valuetransferencoding" : "json",
-->   "mimetype" : "application/json",
-->   "value" : {
-->     "firstName": "John"
-->   }
--> }

<-- HTTP/1.1 400 Bad Reqeust
```

EXAMPLE 4: Mark - Create a validator that marks newly created or updated objects in the “sandbox” container with the results of a validation against the specified JSON schema:

```
--> PUT /validators/myValidator.json HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-object
--> Content-Type: application/cdmi-object
-->
--> {
-->   "valuetransferencoding" : "json",
-->   "mimetype" : "application/schema+json",
-->   "metadata" : {
-->     "cdmi_validation_scope" : [
-->       {
-->         "parentURI": "starts /sandbox/"
-->       }
-->     ],
-->     "cdmi_validation_mark" : "true"
-->   },
-->   "value" : {
-->     "$schema": "https://json-schema.org/draft/2019-09/schema",
-->     "type": "object",
-->     "required": [ "value" ],
-->     "properties": {
-->       "value": {
-->         "type": "object",
-->         "required": [ "name" ],
-->         "properties": {
-->           "name": {
-->             "type": "string"
-->           }
-->         },
-->         "additionalProperties": false
-->       }
-->     }
-->   }
--> }

<-- HTTP/1.1 201 Created
<-- Content-Type: application/cdmi-object
<--
<-- {
<--   "objectType" : "application/cdmi-object",
<--   "objectID" : "00007ED90010D891022876A8DE0BC0FD",
<--   "objectName" : "myValidator.json",
<--   "parentURI" : "/validators/",
<--   "parentID" : "00007E7F00102E230ED82694DAA975D2",
<--   "domainURI" : "/cdmi_domains/MyDomain/",
<--   "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
<--   "completionStatus" : "Complete",
<--   "mimetype" : "application/schema+json",
<--   "metadata" : {
<--     "cdmi_size" : "314",
<--     ...
<--   }
<-- }
```

EXAMPLE 5: Mark - Create a object in the sandbox container that successfully validates against the schema:

```
--> PUT /sandbox/test1.json HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-object
--> Content-Type: application/cdmi-object
-->
--> {
-->   "valuetransferencoding" : "json",
-->   "mimetype" : "application/json",
-->   "value" : {
-->     "name": "John Smith"
-->   }
--> }

<-- HTTP/1.1 201 Created
<-- Content-Type: application/cdmi-object
<--
<-- {
<--   "objectType" : "application/cdmi-object",
<--   "objectID" : "00007ED90010D891022876A8DE0BC0FD",
<--   "objectName" : "test1.json",
<--   "parentURI" : "/sandbox/",
<--   "parentID" : "00007E7F00102E230ED82694DAA975D2",
<--   "domainURI" : "/cdmi_domains/MyDomain/",
<--   "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
<--   "completionStatus" : "Complete",
<--   "mimetype" : "application/json",
<--   "metadata" : {
<--     "cdmi_size" : "24",
<--     "cdmi_validation_schema_provided" : [
<--       "/cdmi_objectid/00007ED90010D891022876A8DE0BC0FD"
<--     ],
<--     "cdmi_validation_result_provided" : [
<--       "passed"
<--     ],
<--     ...
<--   }
<-- }
```

EXAMPLE 6: Mark - Create a object in the sandbox container that does not successfully validate against the schema:

```
--> PUT /sandbox/test1.json HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-object
--> Content-Type: application/cdmi-object
-->
--> {
-->   "valuetransferencoding" : "json",
-->   "mimetype" : "application/json",
-->   "value" : {
-->     "firstName": "John"
-->   }
--> }

<-- HTTP/1.1 201 Created
<-- Content-Type: application/cdmi-object
<--
<-- {
<--   "objectType" : "application/cdmi-object",
<--   "objectID" : "00007ED90010D891022876A8DE0BC0FD",
<--   "objectName" : "test1.json",
<--   "parentURI" : "/sandbox/",
<--   "parentID" : "00007E7F00102E230ED82694DAA975D2",
<--   "domainURI" : "/cdmi_domains/MyDomain/",
<--   "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
<--   "completionStatus" : "Complete",
<--   "mimetype" : "application/json",
<--   "metadata" : {
<--     "cdmi_size" : "24",
<--     "cdmi_validation_schema_provided" : [
<--       "/cdmi_objectid/00007ED90010D891022876A8DE0BC0FD"
<--     ],
<--     "cdmi_validation_result_provided" : [
<--       "failed"
<--     ],
<--     ...
<--   }
<-- }
```