Cloud Data Management Interface Extension: Versioning

Version 1.0g

"Publication of this Working Draft for review and comment has been approved by the Cloud Storage Technical Working Group. This draft represents a "best effort" attempt by the Cloud Storage Technical Working Group 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 http:/snia.org/feedback."

Working Draft

Revision History

Date	Version	By	Comments
2011-12-05	1.0a	David Slik, NetApp, Inc.	Updates to include standard SNIA front matter, minor edits to proposed extension as discussed at last face-to-face.
2012-01-12	1.0b	Marie McMinn	Updates to include standard SNIA front matter and technical edit
2012-01-13	1.0c	David Slik, NetApp, Inc.	Added serialization example, updates based on SNIA TWG ballot feedback.
2012-01-23	1.0d	David Slik, NetApp, Inc.	Incorporated comments and feedback from January face-to-face review, including re-written introduction, updated diagrams and added the X-CDMI-Version-Inhibit header and the ability to specify if only content updates create versions.
2012-01-24	1.0e	David Slik, NetApp, Inc.	Fixed data system metadata lists to be JSON Arrays of JSON Strings instead of comma-delimited lists.
2012-01-24	1.0f	David Slik, NetAPp, Inc.	Updates to reflect TWG discussions, removed X-CDMI-Version-Inhibit.
2012-01-26	1.0g	Marie McMinn	Updates include minor edits.

Object Versioning CDMI Extension

Overview

This CDMI extension adds the ability to request that data objects be versioned and defines how versions are accessed and managed. Version-enabled data objects provide access to and retention of historical versions of a data object and can provide compliance functionality and revision history. Version-enabled data objects also help applications handle multiple concurrent writers in disconnected distributed environments.

Versioning is based around the snapshot concept introduced in CDMI 1.0 and follows the same architectural pattern. It should be reviewed in this context.

Important note for reviewers: Please start reading at section 23 on Page 10.

Modifications to the CDMI 1.0.1 spec:

1) Insert into Clause 3 - "Terms"

3.x
current data object version
the most recent version of a version-enabled data object

3.x
data object version
either the current data object version or a historical data object version

3.x
historical data object version
a non-current state of a version-enabled data object

3.x
version-enabled data object
a CDMI data object with versioning enabled

2) Insert into Clause 8.4.8 - "Examples", at the end of the clause:

EXAMPLE 5 GET to the URI to read a newly-created data object with a current version:

```
GET /MyContainer/MyVersionedDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.0.1
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0.1

{
  "objectType" : "application/cdmi-object",
  "objectID" : "00007ED900100DA32EC94351F8970400",
  "objectName" : "MyVersionedDataObject.txt",
  "parentURI" : "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
```

```
  "domainURI" : "/cdmi_domains/MyDomain/",
  "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
  "completionStatus" : "Complete",
  "mimetype" : "text/plain",
  "metadata" : {
    "cdmi_size" : "33",
    "cdmi_versioning" : "user",
    "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
    "cdmi_version_current" : "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA",
    "cdmi_version_oldest" : [
      "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
    ]
  },
  "valuerange" : "0-32",
  "valuetransferencoding" : "utf-8",
  "value" : "First version of this Data Object"
}
```

EXAMPLE 6 GET to the URI to read a data object with two historical versions:

```
GET /MyContainer/MyVersionedDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.0.1
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0.1

{
  "objectType" : "application/cdmi-object",
  "objectID" : "00007ED900100DA32EC94351F8970400",
  "objectName" : "MyDataObject.txt",
  "parentURI" : "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI" : "/cdmi_domains/MyDomain/",
  "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
  "completionStatus" : "Complete",
  "mimetype" : "text/plain",
  "metadata" : {
    "cdmi_size" : "33",
    "cdmi_versioning" : "user",
    "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
    "cdmi_version_current" : "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC",
    "cdmi_version_oldest" : [
      "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
    ]
  },
  "valuerange" : "0-32",
  "valuetransferencoding" : "utf-8",
  "value" : "Third version of this Data Object"
}
```

EXAMPLE 7 GET to the URI of a data object version:

```
GET /cdmi_objectid/00007ED9001005192891EEBE599D94BB HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.0.1
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0.1

{
  "objectType" : "application/cdmi-object",
  "objectID" : "00007ED9001005192891EEBE599D94BB",
  "objectName" : "MyVersionedDataObject.txt",
  "parentURI" : "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI" : "/cdmi_domains/MyDomain/",
  "capabilitiesURI" : "/cdmi_capabilities/dataobject/dataobject_version/",
  "completionStatus" : "Complete",
  "mimetype" : "text/plain",
  "metadata" : {
    "cdmi_size" : "34",
    "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
    "cdmi_version_current" : "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC",
    "cdmi_version_oldest" : [
      "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
    ],
    "cdmi_version_parent" : "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA",
    "cdmi_version_children" : [
      "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC"
    ]
  },
  "valuerange" : "0-33",
  "valuetransferencoding" : "utf-8",
  "value" : "Second version of this Data Object"
}
```

3) Insert into Clause 12.1.3 - "Data System Metadata Capabilities", Table 104 - "Capabilities for Data System Metadata":

Capability Name	Type	Description
cdmi_versioning	JSON Array of JSON Strings	If present, this capability indicates that the cloud storage system shall support versioning of data objects and contains a list of which versioning behaviors are supported. The following values are defined: • "value" indicates that the system shall support the versioning of the object value. • "user" indicates that the system shall support the versioning of the object value and user metadata. • "all" indicates that the system shall support the versioning of all updates made to a data object. When present, the system shall support the following storage system metadata: "cdmi_version_object", "cdmi_version_current", "cdmi_version_oldest", "cdmi_version_parent", and "cdmi_version_children", as indicated by the corresponding storage system metadata capabilities.
cdmi_versions_count	JSON String	If present, this capability specifies the maximum number of historical versions that may be specified. If absent, restrictions on the number of historical versions specified shall be ignored.

Capability Name	Type	Description
cdmi_version_age	JSON String	If present, this capability specifies the maximum age of historical versions that may be specified. If absent, restrictions on the age of historical versions specified shall be ignored.
cdmi_versions_size	JSON String	If present, this capability specifies the maximum total size of historical versions that may be specified. If absent, restrictions on the size of historical versions specified shall be ignored.

4) Insert into Clause 16.3 - "Support for Storage System Metadata", Table 117 - "Storage System Metadata":

Metadata Name	Type	Description	Requirement
cdmi_version_object	JSON String	If present and "true", this capability indicates that the cloud storage system shall generate a "cdmi_version_object" storage system metadata for each version-enabled data object and data object version.	Conditional
cdmi_version_current	JSON String	If present and "true", this capability indicates that the cloud storage system shall generate a "cdmi_version_current" storage system metadata for each version-enabled data object and data object version.	Conditional
cdmi_version_oldest	JSON Array of JSON Strings	If present and "true", this capability indicates that the cloud storage system shall generate a "cdmi_version_oldest" storage system metadata for each version-enabled data object and data object version.	Conditional
cdmi_version_parent	JSON String	If present and "true", this capability indicates that the cloud storage system shall generate a "cdmi_version_parent" storage system metadata for each data object version that has a previous version.	Conditional
cdmi_version_children	JSON Array of JSON Strings	If present and "true", this capability indicates that the cloud storage system shall generate a "cdmi_version_children" storage system metadata for each data object version.	Conditional

5) Insert into Clause 16.4 - "Data System Metadata", Table 118 - "Data System Metadata":

Metadata Name	Type	Description	Requirement
cdmi_versioning	JSON String	If present, this metadata item indicates that versioning is requested to be enabled for the data object. • If set to the value "value", versions shall be created when the value is updated. • If set to the value "user", versions shall be created when the value and/or user metadata is updated. • If set to the value "all", versions shall be created when any update is performed against the version-enabled data object. This data system metadata item shall not be present in data object versions.	Optional
cdmi_versions_count	JSON String	This metadata item contains the maximum number of historical versions requested to be retained. • If "cdmi_versions_count" is not present, no limit should be placed on the number of versions that are retained. • If "cdmi_versions_count" is present and has a value of zero, only the current version should be retained. • If "cdmi_versions_count" is present and has a value greater than zero, up to the specified number of historical versions should be retained. • If the number of historical versions exceeds the value specified, historical versions should be deleted from the oldest to the newest until the number of historical versions equals the value contained in "cdmi_versions_count".	Optional

Metadata Name	Type	Description	Requirement
cdmi_versions_age	JSON String	This metadata item contains the maximum age of the oldest historical version requested to be retained, specified as the number of seconds before the current time. • If "cdmi_versions_age" is not present, no limit should be placed on the age of versions that are retained. • If "cdmi_versions_age" is present, historical versions should be retained until their age is greater than the value contained in "cdmi_versions_age". • If the age of a historical version exceeds the value specified, that historical version should be deleted.	Optional
cdmi_versions_size	JSON String	This metadata item contains the maximum amount of space requested to be used to retain historical versions, specified in bytes. • If "cdmi_versions_size" is not present, no limit should be placed on the size of versions that are retained. • If "cdmi_versions_size" is present, historic versions should be retained until the total storage consumption of the historical versions exceeds the value contained in "cdmi_versions_size". • If the total size consumed by historical versions exceeds the value specified, historical versions should be deleted from the oldest to the newest until the total storage consumption of historical versions is equal or less than the value contained in "cdmi_versions_count".	Optional

6) Insert into Clause 16.5 - "Support for Provided Data System Metadata", Table 119 - "Support for Provided Data System Metadata":

Capability Name	Type	Description	Requirement
cdmi_versioning_provided	JSON String	Contains the value "value", "user", or "all" if versioning is enabled for the data object.	Conditional
cdmi_versions_count_provided	JSON String	Contains the maximum number of historical versions that will be retained.	Optional

Capability Name	Type	Description	Requirement
cdmi_versions_age_provided	JSON String	Contains the oldest age of a historical version that will be retained, in seconds before the current time.	Optional
cdmi_versions_size_provided	JSON String	Contains the maximum amount of space that can be used to retain historical versions, in bytes.	Optional

7) Insert new Clause after 22 - "Query Queues":

23 Data Object Versions

23.1 Overview

Version-enabled data objects allow the previous state of a data object to be retained when an update is performed. In a non-version-enabled data object, each update changes the state of the object, and the previous state is lost. This state change is illustrated in Figure 10:

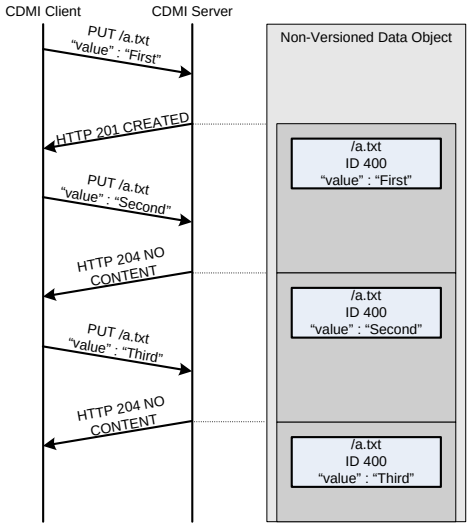

Figure 10 – Updates to a non-version-enabled data object

When a data object has versioning enabled, each update creates a new "current version" with the same contents of the version-enabled data object, and the previous current version becomes a historical version. All versions can be accessed via separate URIs and are immutable. The version-enabled data object continues to be mutable and has the same behaviors to clients as a non-version-enabled data object. This behavior is illustrated in Figure 11 from the perspective of a client.

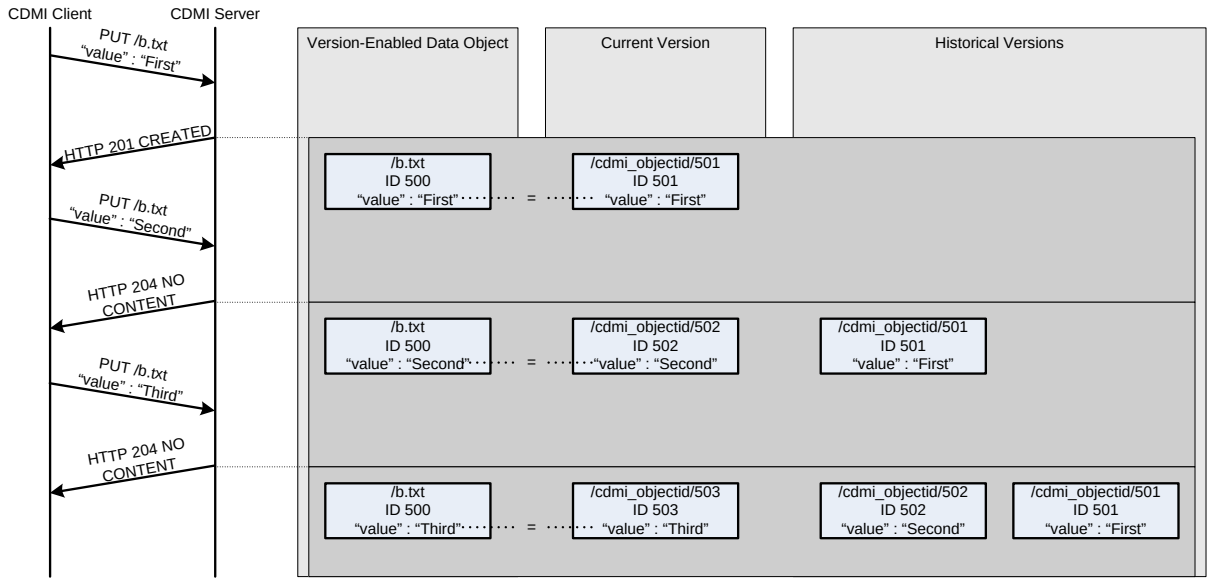

Figure 11 – Updates to a version-enabled data object

Using this approach, CDMI clients that are not aware of versioning can continue to access version-enabled data objects the same way as non-version-enabled data objects, while CDMI clients that are aware of versioning can access and manage the immutable versions associated with the version-enabled data object.

Versioning is enabled for a data object by adding a data system metadata item that indicates that versioning is desired.

Version-enabled data objects and all associated versions contain additional storage system metadata items. These metadata items allow a client to discover the versions that are associated with a version-enabled data object and to iterate through these versions.

The maximum number of versions to be retained, maximum age of versions to be retained, and the maximum space that can be consumed by versions is controlled by data system metadata.

When a data object is version enabled, it always contains at least one version, the "current version". The current version has the same contents as the version-enabled data object but has a different identifier (URI and Object Identifier) and is immutable. When a version-enabled data object is changed, a new current version is created, and the previous current version becomes a historical version.

Versioning has multiple client use cases:

- Clients that need to preserve all data written to a data object over time can use versions to retain all updates made to a data object.
- Clients can restore the contents of a historical version by copying it to the version-enabled data object.
- Clients that retrieve a large data object across multiple parallel or sequential transactions or that need to be able to resume a retrieval at a later time can retrieve the URI for the current version of the data object. Clients can then use that URI to retrieve the data object itself. As the current version is immutable and retains its identifier, even if an update occurs (where the current version becomes a historical version), the client will always receive the same results and will not receive a mixture of the older and newer data object contents.

- Clients can iterate through historical versions to detect where concurrent updates have occurred and can access any overwritten data.

Distributed CDMI implementations can also use versions to merge concurrent changes made on different, eventually consistent nodes without resulting in data loss.

23.2 Traversing Version-Enabled Data Objects

Version-enabled data objects have multiple metadata items that allow a client to traverse through the data object versions.

When a client enables versioning for a data object, the following metadata items shall be added to the version-enabled data object:

- a cdmi_version_object metadata item that contains the URI to the corresponding version-enabled data object. This metadata item allows a client to detect that a given object is a version-enabled data object and not a data object version.
- a cdmi_version_current field that contains the URI to the current version of the version-enabled data object.
- a cdmi_version_oldest field that contains the URI of one or more of the oldest versions. More than one version can exist in this metadata item, as explained in clause 23.3 Concurrent Updates and Version-Enabled Data objects.

Each data object version shall contain the above three fields, with the same values as found in the version-enabled data object. Each data object version shall also contain the following two fields:

- a cdmi_version_parent field that contains the URI of the previous version. If the data object version does not have a parent, this field is omitted.
- cdmi_version_children field that contains the URIs of the versions created by modifying this version. If the data object version does not have any children, this metadata item shall be empty.

To visualize how these fields allow a client to traverse data object versions, the linkages between the version-enabled data object and data object versions in the final state of Figure 11 is illustrated in Figure 12.

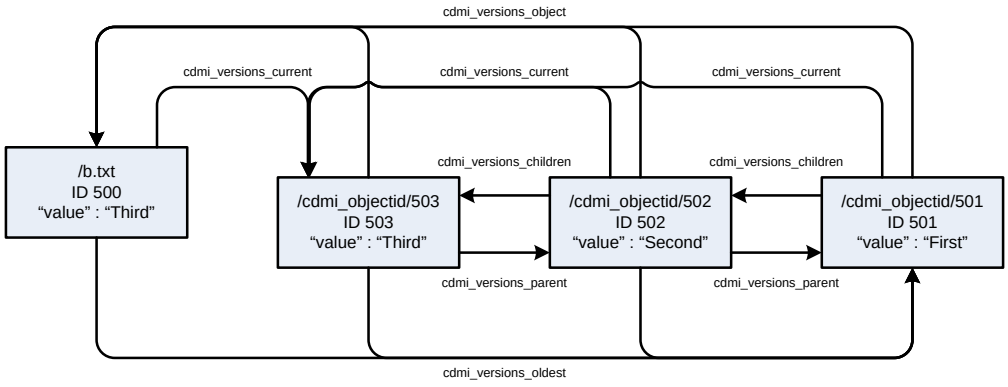

Figure 12 – Linkages between a version-enabled data object and data object versions

A client accessing the version-enabled data object (/b.txt) can traverse to the current version and to the oldest version.

A client accessing a data object version can traverse to the version-enabled data object, to the current version, to the parent version, to child versions, and to the oldest version.

23.3 Concurrent Updates and Version-Enabled Data Objects

When multiple concurrent updates are performed against a version-enabled data object, each update is performed against the state of the object at the time the update starts. The change to the state resulting from the update to the object becomes visible to clients at the time the update completes.

Two different types of concurrent updates can occur: overlapping updates and nested updates. Figures 13 and 14 show the update sequence and resulting version linkages for overlapping updates:

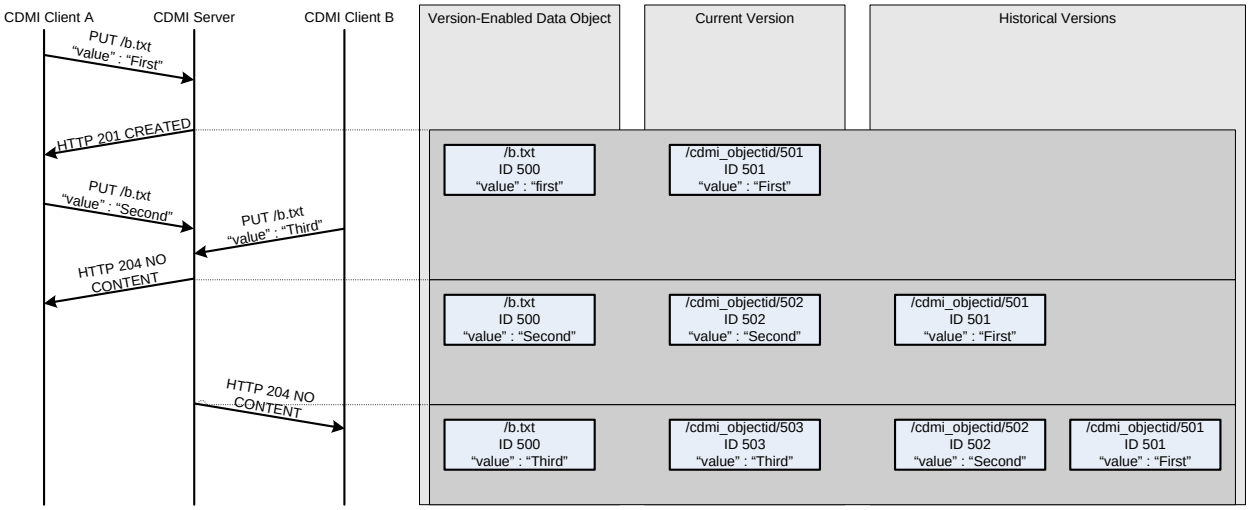

Figure 13 – Overlapping concurrent updates

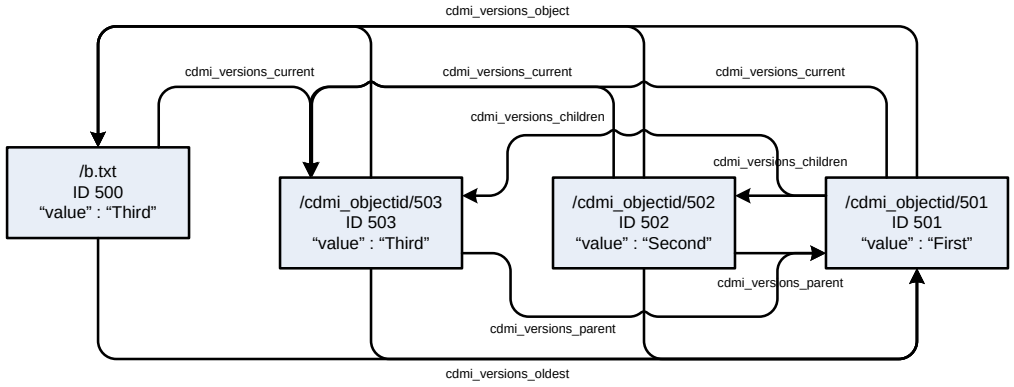

Figure 14 – Linkages for overlapping updates

In the sequence illustrated in Figure 13, both the "Second" and "Third" updates are performed against the "First" state. As the "Third" update completes last, it becomes the current version. In this example, historical version 501 would have two children, versions 502 and 503. Both versions 502 and 503 would have the same parent 501.

Figures 15 and 16 show the update sequence and resulting version linkages for nested updates:

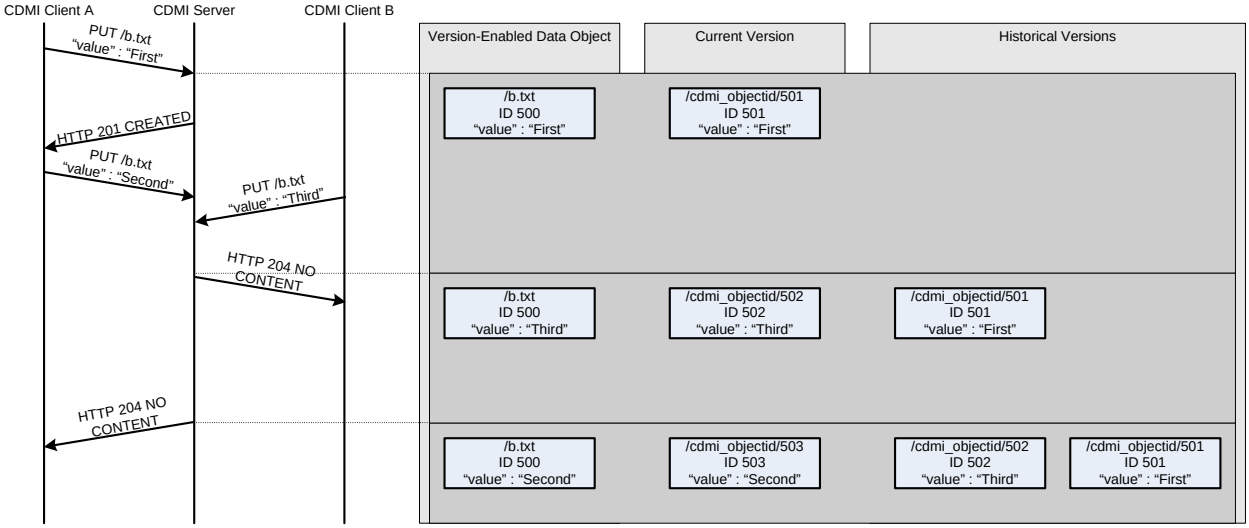

Figure 15 – Nested concurrent updates

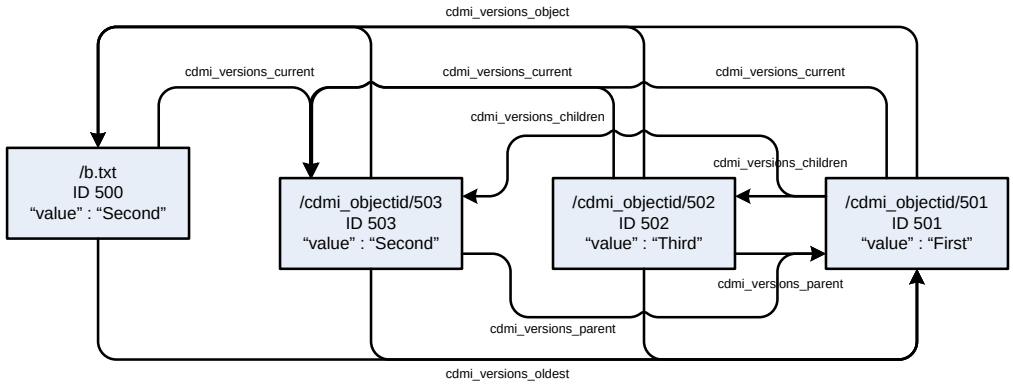

Figure 16 – Linkages for nested updates

In the sequence illustrated in Figure 16, both the "Second" and "Third" updates are performed against the "First" state. As the "Second" update completes last, it becomes the current version. In this example, historical version 501 would have two children, versions 502 and 503. Both versions 502 and 503 would have the same parent 501.

Both of these data structures are equivalent, with the only difference being which update completed last.

23.4 Capabilities for Version-Enabled Data Objects

The relationship between version-enabled data objects, data object versions, and capabilities is shown in Figure 17.

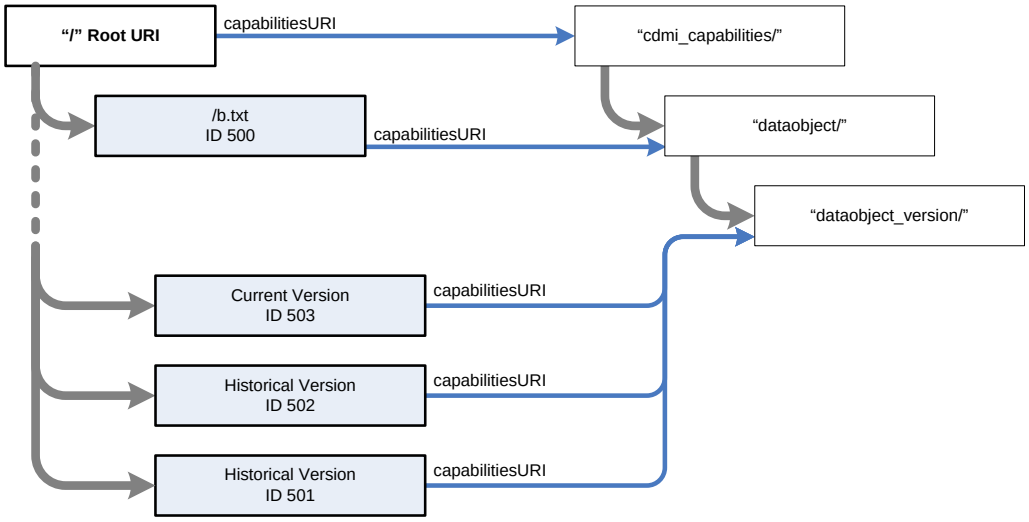

Figure 17 – Version to capabilityURI Relationships

Data object versions are immutable but may be deleted by a client or by the system, depending on the data system metadata specified.

23.5 Updates Triggering Version Creation

If versioning is enabled by setting the value of the cdmi_versions metadata item in the version-enabled data object to "value", the following updates will trigger the creation of a new version:

- changing the mimetype,
- changing the value, or
- changing the valuetransferencoding.

If versioning is enabled by setting the value of the cdmi_versions metadata item in the version-enabled data object to "user", the following updates will trigger the creation of a new version:

- changing the mimetype,
- changing the value,
- changing the valuetransferencoding, or
- adding, modifying, or removing user metadata.

If versioning is enabled by setting the value of the cdmi_versions metadata item in the version-enabled data object to "all", then all updates to the data object will trigger the creation of a new version.

The effective ACL, owner, and domain of the data object versions shall be the ACL, owner, and domain of the version-enabled data object.

Modifications performed with the X-CDMI-Partial header shall not trigger the creation of a new version until the completionStatus is changed from "Processing" to "Complete".

23.6 Operations against Version-Enabled Data Objects

Moving a version-enabled data object within a system is considered to be an update to the name and/or parentURI fields.

Moving a version-enabled data object between systems moves all data object versions associated with the version-enabled data object and preserves all identifiers. If the destination name and/or URI are different, the move is considered to be an update to the name and/or parentURI fields.

Copying a version-enabled data object shall only copy the version-enabled data object itself. Versions of the version-enabled data object are not copied.

Deleting a version-enabled data object shall also delete all versions associated with that version-enabled data object.

Disabling versioning for a version-enabled data object shall preserve all versions. Previously existing versioning metadata shall remain present while versioning is disabled. Re-enabling versioning for a data object that previously was version-enabled shall result in the creation of a new current version.

If a version-enabled data object is placed under retention or hold, the retention behaviors of the version-enabled data object shall be applied to the data object versions.

No additional log messages or notifications are defined for version-enabled data objects. When a version-enabled data object is updated, an additional creation log message and/or notification message shall be generated for the created data object version. Likewise, when a data object version is accessed or deleted, a log and/or notification message is generated.

If a limited number, size, or age for versions is requested and a change to a version-enabled data object results in a version being automatically deleted, then the system shall generate a corresponding deletion log and/or notification message for the deleted data object version.

23.7 Operations against Data Object Versions

A data object version is presented to the client as a standard CDMI data object.

Moving, copying over, deserializing over, and updating a data object version shall not be permitted and shall result in an HTTP status code of 403 Forbidden.

Copying a data object version is permitted. For example, to promote a version to become the current version of a version-enabled data object, the URI of the data object version is used in the copy field when performing an update to the URI of the version-enabled data object. Updates can also be performed as part of the copy operation.

Deleting a historical data object version shall be permitted if the client has ACL permissions to delete the version-enabled data object and the version-enabled data object.

Deleting the current version of a version-enabled data object shall revert the current version to the current version's parent. If there is no parent version, deleting the current version shall result in an HTTP status code of 403 Forbidden.

When an intermediate historical version is deleted, the parent and children metadata items of the parent and all child data object versions of the data object version being deleted must be updated.

EXAMPLE In a version chain "C" -> "B" -> "A", where "C" is the newest and "A" is the oldest, deleting version "B" shall produce the following results:

- The cdmi_version_parent metadata item of "C" is set to the URI contained in the cdmi_version_parent metadata item of "B".

- The URI of "B" in the cdmi_version_children metadata item of "A" is replaced with the URIs contained in the cdmi_version_children metadata item of "B".

In pseudocode, the above translates to:

```
C->cdmi_version_parent = B->cdmi_version_parent
A->cdmi_version_children[B] = B->cdmi_version_children
Delete B
```

If the oldest version of a version-enabled data object is deleted and there are two or more children of that version, both of the children of the deleted oldest version will become the new oldest version.

When accessing a data object version, the cdmi_acount and cdmi_atime of the data object version shall be updated if present.

When accessing a historical version of a version-enabled data object, the ACL, owner, and domainURI of the version-enabled data object shall be in effect.

Standard log and notification messages are sent when data object versions are accessed and deleted.

23.8 Query of Data Object Versions

As data object versions are regular CDMI objects, they will be included in query results unless explicitly excluded.

Querying for data object versions is performed by including the scope:

```
"metadata" :
{
  "cdmi_version_children" : "*"
}
```

Querying for version-enabled data objects (but not their versions) is performed by including the scope:

```
"metadata" :
{
  "cdmi_versioning" : "*"
}
```

Querying for non-versioned data objects with no versions is performed by including the scope:

```
"metadata" :
{
  "cdmi_version_current" : "!*"
}
```

Querying for non-versioned data objects with versions is performed by including the scope:

```
"metadata" :
{
  "cdmi_versioning" : "!*",
  "cdmi_version_current" : "*"
}
```

23.9 Version-Enabled Data Object Serialization

Serializing a version-enabled data object shall serialize the data object, the versioning-related metadata, the current version, and all historical versions. The current version and all historical versions shall be serialized as data objects contained within a JSON array. These data objects shall replace the contents of the value field of the serialized representation of the version-enabled data object.

EXAMPLE A version-enabled data object with three versions is serialized.


```
{
  "objectType" : "application/cdmi-object",
  "objectID" : "00007ED900100DA32EC94351F8970400",
  "objectName" : "MyVersionedDataObject.txt",
  "parentURI" : "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI" : "/cdmi_domains/MyDomain/",
  "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
  "completionStatus" : "Complete",
  "mimetype" : "text/plain",
  "metadata" : {
    "cdmi_size" : "33",
    "cdmi_versioning" : "user",
    "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
    "cdmi_version_current" : "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC",
    "cdmi_version_oldest" : [
      "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
    ]
  },
  "value" : [
    {
      "objectType" : "application/cdmi-object",
      "objectID" : "00007ED90010F077F4EB1C99C87524CC",
      "objectName" : "MyVersionedDataObject.txt",
      "parentURI" : "/MyContainer/",
      "parentID" : "00007E7F00102E230ED82694DAA975D2",
      "domainURI" : "/cdmi_domains/MyDomain/",
      "capabilitiesURI" : "/cdmi_capabilities/dataobject/dataobject_version/",
      "completionStatus" : "Complete",
      "mimetype" : "text/plain",
      "metadata" : {
        "cdmi_size" : "33",
        "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
        "cdmi_version_current" : "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC",
        "cdmi_version_oldest" : [
          "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
        ],
        "cdmi_version_parent" : "/cdmi_objectid/00007ED9001005192891EEBE599D94BB",
        "cdmi_version_children" : [
        ]
      },
      "valuerange" : "0-32",
      "valuetransferencoding" : "utf-8",
      "value" : "Third version of this Data Object"
    },
    {
      "objectType" : "application/cdmi-object",
      "objectID" : "00007ED9001005192891EEBE599D94BB",
      "objectName" : "MyVersionedDataObject.txt",
      "parentURI" : "/MyContainer/",
      "parentID" : "00007E7F00102E230ED82694DAA975D2",
      "domainURI" : "/cdmi_domains/MyDomain/",
```

```
      "capabilitiesURI" : "/cdmi_capabilities/dataobject/dataobject_version/",
      "completionStatus" : "Complete",
      "mimetype" : "text/plain",
      "metadata" : {
        "cdmi_size" : "34",
        "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
        "cdmi_version_current" : "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC",
        "cdmi_version_oldest" : [
          "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
        ],
        "cdmi_version_parent" : "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA",
        "cdmi_version_children" : [
          "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC"
        ]
      },
      "valuerange" : "0-33",
      "valuetransferencoding" : "utf-8",
      "value" : "Second version of this Data Object"
    },
    {
      "objectType" : "application/cdmi-object",
      "objectID" : "00007ED90010512EB55A9304EAC5D4AA",
      "objectName" : "MyVersionedDataObject.txt",
      "parentURI" : "/MyContainer/",
      "parentID" : "00007E7F00102E230ED82694DAA975D2",
      "domainURI" : "/cdmi_domains/MyDomain/",
      "capabilitiesURI" : "/cdmi_capabilities/dataobject/dataobject_version/",
      "completionStatus" : "Complete",
      "mimetype" : "text/plain",
      "metadata" : {
        "cdmi_size" : "33",
        "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
        "cdmi_version_current" : "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC",
        "cdmi_version_oldest" : [
          "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
        ],
        "cdmi_version_children" : [
          "/cdmi_objectid/00007ED9001005192891EEBE599D94BB"
        ]
      },
      "valuerange" : "0-32",
      "valuetransferencoding" : "utf-8",
      "value" : "First version of this Data Object"
    }
  ]
}
```


Serializing a non-version-enabled data object that has versions shall serialize the data object, the versioning-related metadata, and all historical versions. The contents of the value field of the data object, the current version, and all historical versions serialized as data objects shall be contained within a JSON array. These data objects shall replace the contents of the value field of the serialized representation of the version-enabled data object.

Deserializing either a version-enabled data object or a non-version-enabled data object with versions shall restore the data object and all serialized versions.

Serializing and deserializing a data object version shall not be permitted.

Attempting to deserialize a serialized version-enabled data object or non-version-enabled data object with versions onto a system that does not support versions shall result in an HTTP status code of 400 Bad Request. This error code results because a CDMI system that does not

support versions expects a JSON string for the value field of a serialized data object, not a JSON array.