# Content Management Interoperability Services Version 0.60

# Part II – ReSTful AtomPub Binding

## Committee Draft

## 14 March 2009

**Specification URIs:**
**This Version:**
http://docs.oasis-open.org/cmis/ [additional path/filename] .html
http://docs.oasis-open.org/cmis/ [additional path/filename] .doc
http://docs.oasis-open.org/cmis/ [additional path/filename] .pdf

**Previous Version:**
http://docs.oasis-open.org/cmis/ [additional path/filename] .html
http://docs.oasis-open.org/cmis/ [additional path/filename] .doc
http://docs.oasis-open.org/cmis/ [additional path/filename] .pdf

**Latest Version:**
http://docs.oasis-open.org/cmis/ [additional path/filename] .html
http://docs.oasis-open.org/cmis/ [additional path/filename] .doc
http://docs.oasis-open.org/cmis/ [additional path/filename] .pdf

**Technical Committee:**
OASIS Content Management Interoperability Services TC

**Chair(s):**
David Choy

**Editor(s):**
Al Brown
Ethan Gur-Esh

**Related work:**
This specification replaces or supercedes:

- CMIS Draft 0.50

This specification is related to:

- CMIS Part I Draft 0.52
- CMIS Part II – Web Services Draft 0.52

**Declared XML Namespace(s):**
http://docs.oasis-open.org/ns/cmis/core/200901
http://docs.oasis-open.org/ns/cmis/messaging/200901
http://docs.oasis-open.org/ns/cmis/ws/200901

**Abstract:**

This specification defines the ReSTful AtomPub based binding.

**Status:**

This document was last revised or approved by the CMIS TC on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/CMIS/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/CMIS/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/CMIS/.

## Notices

# Table of Contents

# 1 Introduction

The Content Management Interoperability Services (CMIS) ReSTful AtomPub binding specification defines a specification based on AtomPub that can be used by applications to work with one or more Content Management Repositories.

## 1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **Error! Reference source not found.**. (AL - Change to ISO)

## 1.2 Normative References

| | |
|---|---|
| **[RFC4287]** | M. Nottingham, *Atom Syndication Format*, http://www.ietf.org/rfc/rfc4287.txt, December 2005 |
| **[RFC5023]** | J. Gregorio, *Atom Publishing Protocol*, http://www.ietf.org/rfc/rfc5023.txt, October 2007 |
| **[RFC2616]** | R. Fielding, *Hypertext Transfer Protocol --HTTP/1.1*, http://www.w3.org/Protocols/rfc2616/rfc2616.html, June 1999 |
| **[RFC5005]** | M. Nottingham, *Feed Paging and Archiving*, http://www.ietf.org/rfc/rfc5005.txt, September 2007 |
| **[CMISDM]** | OASIS, Committee Draft 0.52, “Content Management Interoperability Services (CMIS) Domain Model”, March 2009 |

## 1.3 Non-Normative References

**NOTE: The proper format for a citation to an OASIS Technical Committee’s work (whether Normative or Non-Normative) is:**

OASIS

Stage (Committee Draft 01, Committee Draft 02, Committee Specification 01, etc. or Standard)

Title (italicized or in quotation marks)

Approval Date (Month YYYY)

URI of the actual Authoritative Specification (namespace is not acceptable as the content changes over time)

For example:

| | |
|---|---|
| **EDXL-HAVE** | OASIS Standard, “Emergency Data Exchange Language (EDXL) Hospital AVailability Exchange (HAVE) Version 1.0”, November 2008. http://docs.oasis-open.org/emergency/edxl-have/os/emergency_edxl_have-1.0-spec-os.doc |

# 2 Overview

The REST binding has one mode: Pure-Atom following naming conventions and additional information in Atom documents.

The client will request the service document at the URL provided by vendor.  The client will then choose a collection, and then start accessing the repository.

The URI for different items are found off of the atom feed document in the link tags for operations Atom or APP does not natively support.  The tags have special names that will be registered with IANA.

Optional parameters are passed in as HTTP arguments for non-default behavior.

Custom properties will be part of the CMIS namespace using generic property tags.

Special collections have been created that have semantic meaning beyond collection membership. These are:

- Unfiled – All documents added to this collection will be removed from all other collections
- Checkedout – All documents added to this collection will be checkedout

## 2.1 Authentication

Authentication will be handled by the transport protocol.  Support for HTTP Basic authentication is recommended.

## 2.2 Response Formats

The client can specify in HTTP the Accept header which specifies which formats are acceptable to the client.  With this mechanism the client can chose which response format the CMIS should respond with. CMIS compliant implementation MUST support this one response format:

- application/atom+xml

The CMIS repository will chose the response format based on the Accept header if specified.  If the Accept header is not specified, then the CMIS repository should

The CMIS repository is free to support other formats such as:

- application/x-javascript (or text/javascript) for JSON
- text/html for an HTML interface to the API

## 2.3 Optional Arguments

The binding supports adding optional parameters to CMIS resources to modify default behavior. These arguments would be appended to the URI specified.

CMIS implementations MUST support arguments being specified as HTTP arguments.

If optional arguments are specified by the client, the server must return a Content-Location header with the more specific URI of the resource if one exists.

## 2.4 Method Overrides

In case there is a proxy that is blocking PUT and DELETE, X-Method-Override header SHOULD be supported.

# 3 CMIS

## 3.1 Mime Types

CMIS introduces two new mime types for a CMIS Query document and a CMIS AllowableActions document.

In addition to those mime types specified by CMIS, CMIS also leverages these mimetypes:

- APP Service
- Atom Entry
- Atom Feed

### 3.1.1 Query

Type: Application/cmisquery+xml

Starting tag: query

This document contains the representation of a query to be executed in a CMIS repository.

Example:

```
<query xmlns=”http://www.cmis.org/CMIS/1.0”>
            <statement>SELECT * FROM document</statement>
            <searchAllVersions>false</searchAllVersions>
            <pageSize>0</pageSize>
            <skipCount>0</skipCount>
</query>
```

### 3.1.2 AllowableActions

Type: Application/cmisallowableactions+xml

Starting tag: allowableactions

This document contains the representation of the allowable actions the user may perform on the referenced object.

## 3.2 CMIS Link Types for ATOM

The table below outlines the different link types in CMIS. This is in addition to the link relations specified by Atom and Atom Publishing Protocol.

Links can have the following attributes in addition to the ones specified by Atom and Atom Publishing Protocol:

- (CMIS) id: Specifies the CMIS ID of the resource pointed at the link. It is recommended to include this attribute.

These are the links specified by CMIS:

### 3.2.1 parents

This is the collection of parents of the object, specifically the folders a document is filed into.

Mime/Type: Atom Feed

### 3.2.2 repository

This link points to the service document containing the object.  The current repository will use relationship self on the workspace element

Mime/Type: APP Service

### 3.2.3 children

This is the collection of children for a folder or type entry.

Mime/Type: Atom Feed

### 3.2.4 descendants

This is the collection of children including their children for a folder or type entry.

Mime/Type: Atom Feed

### 3.2.5 allowableactions

This link references the document containing allowable actions for the respective objects.

Mime/Type: Allowable Actions

### 3.2.6 allversions

This link references the document containing the version history (set of versions) for the document.

Mime/Type: Atom Feed

### 3.2.7 latestversion

This link references the document containing the latest version of the document

Mime/Type: Atom Entry

### 3.2.8 relationships

This link references the document containing the set of relationship of the current object.

Mime/Type: Atom Feed

### 3.2.9 type

This link references the document containing the type definition of the current object

Mime/Type: Atom Entry

### 3.2.10 source

This link references the source of the current object.  In a relationship document, this link references the source object.  In feeds, this link references the atom entry document of the folder.

Mime/Type: Atom Entry

### 3.2.11 target

This link references the target of the current object.  In a relationship document, this link references the source object.

Mime/Type: Atom Entry

### 3.2.12 stream

This link references the [content] stream of the current object.

Mime/Type: MimeType of the stream

| **CMIS Link** | **Description** |
|---|---|
| **stream** | This points to the file for the document |

## 3.3 Exceptions

### 3.3.1 Common Exceptions

The following table defines the HTTP status codes that repositories will return for the various common exceptions defined in Part I of the CMIS specification.

| **CMIS Services Exception** | **HTTP Status Code** |
|---|---|
| InvalidArgumentException | 400 |
| ConstraintViolationException | 409 |
| ObjectNotFoundException | 404 |
| PermissionDeniedException | 401 |
| OperationNotSupportedException | 405 |
| UpdateConflictException | 409 |
| RuntimeException | 500 |
| Successful DELETE TREE | 204 |

### 3.3.2 Other Exceptions

| **CMIS Services Exception** | **HTTP Status Code** |
|---|---|
| TypeNotFoundException | 404 |
| FilterNotValidException | 400 |
| StreamNotSupportedException | 403 |
| StorageException | 500 |
| OffsetException | 400 |
| NotInFolderException | 404 |

| | |
|---|---|
| ContentAlreadyExistsException | 409 |
| VersioningException | 409 |
| ContentStreamNotProvided | 400 |

## 3.4 Resources

### 3.4.1 Atom Entries and Feeds

All resources below that are represented as Entries and Feeds, must follow the guidance in this section.

#### 3.4.1.1 Feeds

Any feed must be a valid atom Feed document and conform to the guidelines below:

1. Updated will be the latest time the folder or its contents was updated. If unknown by the underlying repository, it should be the current time.
2. Author/name will be CMIS:createdBy
3. Title will be CMIS:name
4. App:edited will be CMIS:lastModifiedDate
5. Link without relation will be generated to return the uri of the feed

Paging of feeds will follow RFC 5005 <http://www.ietf.org/rfc/rfc5005.txt>.

#### 3.4.1.2 Entries

At any point where an Atom document of type Entry is sent or returned, it must be a valid Atom Entry document and conform to the guidelines below:

a. Atom:Title will be best efforts by the repository.  The repository should chose a property closest to Title.

b. App:edited will be CMIS:lastModifiedDate

c. Link with relation self will be the URI that returns the Atom Entry document

d. Published will be CMIS:createdDate

e. Atom:author will be CMIS:creator

f. For content tags

g. Documents with content

    1. Leverage the src attribute to point to the same link as cmis-stream

    2. The repository SHOULD populate the summary tag with text that at best efforts represents the documents.  For example, an HTML table containing the properties and their values for simple feed readers

        ii. Other (Content-less document, Folder, Relationship, Type, etc) – best efforts at generating HTML text that represents the object.  That text would normally go into the summary tag, but since there is no content, goes in the content tag.

    b. For summary tag,

h. If content src is specified, the summary SHOULD contain a text or html representation of the object.

h. Links will be used to provide URIs to CMIS functionality

        ii. CMIS Links may be omitted if the function is not allowed and that function would not show up on getAllowableActions.
        iii. Links may be omitted if the repository does not support that capability
    c. All CMIS properties will be exposed in CMIS properties tag even if they are duplicated in an atom element

When POSTing an Atom Document, the atom fields take precedence over the CMIS property field for writeable properties. For example, atom:title will overwrite cmis:name

#### 3.4.1.3 Categories

The categorization functionality of Atom will be repository specific. Specifically the repository may:

1. Internalize categories via some scheme and expose
2. Ignore the category tags

Category schema:

```
        <?xml version="1.0" ?>
        <app:categories
            xmlns:app="http://www.w3.org/2007/app"
            xmlns:atom="http://www.w3.org/2005/Atom"
            fixed="yes" scheme="http://example.com/cats/bag3">
          <atom:category term="animal" />
          <atom:category term="vegetable" />
          <atom:category term="mineral" />
        </app:categories>
```

Categories included inside the Atom entry document:

```
<atom:category scheme="http://example.com/cats/b”g3" term="mineral"/>
```

### 3.4.2 Repository

This is the repository logical object. It is represented by the atom service document. This is also exposed on entry as link. Each repository is mapped to a Workspace in the Service document.

Document Type: Service

CMIS Services exposed:

GET: getRepositories, getRepositoryInfo

PUT: Not Supported

POST: Not Supported

DELETE: Not Supported

How the client will get the ATOM (APP) service document is repository specific. Examples are via URI, or loading the service document from disk.

In the Atom (APP) Service Document, each workspace maps to a single repository. A workspace element for a CMIS repository will have a collection element for the following collections.

- Root folder: Root folder of the Repository
    - ‘root-children’ for the children collection of the root folder
    - ‘root-descendants’ for the descendants collection of the root folder
- Unfiled folder: Folder for posting documents to be unfiled; read can be disabled

  - ‘unfiled’
- Checkedout Folder: Folder containing all checked out documents user can see
  - ‘checkedout’
- Types Folder: Folder containing all the types in the repository
  - ‘types-children’ for the children collection
  - ‘types-descendants for the children collection
- Query collection: Collection for posting queries to be executed

The workspace element will have two CMIS attributes: id and repositoryRelationship. RepositoryRelationship specifies the relationship of the repository to others listed in the service document. The repository name will be exposed in the workspace element via the atom:title element.

This service document represents both getRepositories() and getRepositoryInfo(repid). Each repository will be a workspace in the service document.

#### 3.4.2.1 HTTP Methods

##### 3.4.2.1.1 GET

This retrieves the APP Service document for a specified repository. This exposes the capabilities defined in getRepositories and getRepositoryInfo in the Domain Model.

###### 3.4.2.1.1.1 repositoryId Argument

If specified, this argument specifies the repositoryId to be used to generate the APP Service document.

### 3.4.3 Service Collections

#### 3.4.3.1 Root Folder Collections

This is a collection described in the service document. Please see Folder Children or Folder Descendants.

Document Type: Feed

CMIS Services:

GET: getChildren or getDescendants

PUT: Not Supported

POST: addObjectToFolder

DELETE: Not Supported

#### 3.4.3.2 Query Collection

This is a collection of persisted Queries

Document Type: Feed

CMIS Services:

GET:If items are returned in the feed, they are persisted queries.

PUT: Not Supported

POST: Query

DELETE: Not Supported

##### 3.4.3.2.1 HTTP Methods

###### 3.4.3.2.1.1 POST

This collection must accept CMIS Query documents.  Upon submission of a query document, a response must be returned with a feed for that query or an exception must be thrown.

#### 3.4.3.3 Checked Out Collection

This is a collection described in the service document that contains all the checkedout documents

Document Type: Feed

CMIS Services:

GET: getCheckedOutDocuments

PUT: Not Supported

POST: checkout

DELETE: Not Supported

##### 3.4.3.3.1 HTTP Methods

###### 3.4.3.3.1.1 GET

###### 3.4.3.3.1.1.1 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

###### 3.4.3.3.1.1.2 folderId Argument

If specified, argument folderId limits the checked out documents returned to be in that folder. The repository may allow documents in subfolders to be returned.

Valid values: Opaque string

Default: Not set

###### 3.4.3.3.1.1.3 maxItems Argument

If specified, argument maxitems limits the types returned to be that size or less.  The repository may choose a lower maxItems than specified due to its configuration.

Valid values: Positive integer

Default: repository specific

###### 3.4.3.3.1.1.4 skipCount Argument

If specified, argument skipCount skips the specified number of types returned to be that size or less.  The repository may choose a lower skipCount than specified due to its configuration.

Valid values: Zero or Positive integer

Default: 0

###### 3.4.3.3.1.1.5 includeAllowableActions Argument

If specified, argument includeAllowableActions includes the allowable actions of the current user context in the atom entry.

Valid values: true, false

Default: false

##### 3.4.3.3.1.1.6 includeRelationships Argument

If specified, argument includeRelationships includes the allowable actions of the current user context in the atom entry.

Valid values: enumIncludeRelationships

Default: none

```
	<xs:simpleType name="enumIncludeRelationships">
		<xs:restriction base="xs:string">
			<xs:enumeration value="none" />
			<xs:enumeration value="source" />
			<xs:enumeration value="target" />
			<xs:enumeration value="both" />
		</xs:restriction>
	</xs:simpleType>
```

## 3.4.3.4 Unfiled Collection

This is a collection described in the service document that contains all the unfiled documents in the repository.

Document Type: Feed

CMIS Services:

GET:  getUnfiled

PUT: Not Supported

POST: removeDocumentFromFolder, removeDocumentFromAllFolders

DELETE: Not Supported

### 3.4.3.4.1 HTTP Methods

#### 3.4.3.4.1.1 GET

##### 3.4.3.4.1.1.1 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

##### 3.4.3.4.1.1.2 folderId Argument

If specified, argument folderId limits the checked out documents returned to be in that folder. The repository may allow documents in subfolders to be returned.

Valid values: Opaque string

Default: Not set

###### 3.4.3.4.1.1.3 maxItems Argument

If specified, argument maxitems limits the types returned to be that size or less.  The repository may choose a lower maxItems than specified due to its configuration.

Valid values: Positive integer

Default: repository specific

###### 3.4.3.4.1.1.4 skipCount Argument

If specified, argument skipCount skips the specified number of types returned to be that size or less.  The repository may choose a lower skipCount than specified due to its configuration.

Valid values: Zero or Positive integer

Default: 0

###### 3.4.3.4.1.1.5 includeAllowableActions Argument

If specified, argument includeAllowableActions includes the allowable actions of the current user context in the atom entry.

Valid values: true, false

Default: false

###### 3.4.3.4.1.1.6 includeRelationships Argument

If specified, argument includeRelationships includes the allowable actions of the current user context in the atom entry.

Valid values: enumIncludeRelationships

Default: none

```
	<xs:simpleType name="enumIncludeRelationships">
		<xs:restriction base="xs:string">
			<xs:enumeration value="none" />
			<xs:enumeration value="source" />
			<xs:enumeration value="target" />
			<xs:enumeration value="both" />
		</xs:restriction>
	</xs:simpleType>
```

#### 3.4.3.5 Types Children Collection

This is a collection described in the service document that contains the types in the repository under the specified parent type.  If no parent type is specified, then the base types are returned in the feed.  This feed does not include any nesting.

Document Type: Feed

CMIS Services:

GET:  getTypes

PUT: Not Supported

POST: Not Supported

DELETE: Not Supported

#### 3.4.3.5.1 HTTP Methods

##### 3.4.3.5.1.1 GET

###### 3.4.3.5.1.1.1 includePropertyDefinitions Argument

If specified, property definitions will be included in the document.

Valid values: true, false

Default: false

###### 3.4.3.5.1.1.2 maxItems Argument

If specified, argument maxitems limits the types returned to be that size or less.  The repository may choose a lower maxItems than specified due to its configuration.

Valid values: Positive integer

Default: repository specific

###### 3.4.3.5.1.1.3 skipCount Argument

If specified, argument skipCount skips the specified number of types returned to be that size or less.  The repository may choose a lower skipCount than specified due to its configuration.

Valid values: Zero or Positive integer

Default: 0

### 3.4.3.6 Types Descendants Collection

This is a collection described in the service document that contains all the types in the repository.

Document Type: Feed

CMIS Services:

GET:  getTypes

PUT: Not Supported

POST: Not Supported

DELETE: Not Supported

#### 3.4.3.6.1 HTTP Methods

##### 3.4.3.6.1.1 GET

###### 3.4.3.6.1.1.1 includePropertyDefinitions Argument

If specified, property definitions will be included in the document.

Valid values: true, false

Default: false

###### 3.4.3.6.1.1.2 depth Argument

If specified, argument depth limits the types returned to be within a certain depth of the specified type.  The repository may choose a lower depth than specified due to its configuration.

Valid values: Positive integer

Default: 1

### 3.4.4 Collections

#### 3.4.4.1 Relationships Collection

This is the set of relationships available (either source or target or both) from a specific item such as  a document, folder or policy.

Document Type: Atom Feed

CMIS Services:

GET:  getRelationships

PUT: Not Supported

POST: createRelationship

DELETE: Not Supported

##### 3.4.4.1.1 HTTP Methods

###### 3.4.4.1.1.1  GET

###### 3.4.4.1.1.1.1 relationshipType Argument

If specified, this will limit the relationships returned.

Valid values: typeId

Default: not set (all relationship types)

###### 3.4.4.1.1.1.2 includeSubRelationshipTypes Argument

If specified, this will limit the relationships to the type specified.  If no type specified, then the base relationship type.

Valid values: true, false

Default: true

###### 3.4.4.1.1.1.3 direction Argument

If specified, this will limit the direction of relationships to be retrieved.

Valid values: enumRelationshipDirection

Default: both

```
      <xs:simpleType name="enumRelationshipDirection">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="source" />
                  <xs:enumeration value="target" />
                  <xs:enumeration value="both" />
            </xs:restriction>
      </xs:simpleType>
```

###### 3.4.4.1.1.1.4 maxItems Argument

If specified, argument maxitems limits the types returned to be that size or less.  The repository may choose a lower maxItems than specified due to its configuration.

Valid values: Positive integer

Default: repository specific

###### 3.4.4.1.1.1.5 skipCount Argument

If specified, argument skipCount skips the specified number of types returned to be that size or less.  The repository may choose a lower skipCount than specified due to its configuration.

Valid values: Zero or Positive integer

Default: 0

###### 3.4.4.1.1.1.6 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

###### 3.4.4.1.1.1.7 includeAllowableActions Argument

If specified, argument includeAllowableActions includes the allowable actions of the current user context in the atom entry.

Valid values: true, false

Default: false

### 3.4.4.2 Parents Collection

This is the set of parents for a specific object.  This can also be the ancestor of folders if returnToRoot is selected.

Document Type: Atom Feed

CMIS Services:

GET:  getDocumentParents

PUT: Not Supported

POST: Not Supported

DELETE: Not Supported

#### 3.4.4.2.1 HTTP Methods

##### 3.4.4.2.1.1  GET

###### 3.4.4.2.1.1.1 returnToRoot Argument

If specified, the repository will return the entire set of parent(s) from the current object to the root folder.  The parents will be nested inside Atom Entries and not a flat list.

Valid values: true, false

Default: false

###### 3.4.4.2.1.1.2 maxItems Argument

If specified, argument maxitems limits the types returned to be that size or less.  The repository may choose a lower maxItems than specified due to its configuration.

Valid values: Positive integer

Default: repository specific

##### 3.4.4.2.1.1.3 skipCount Argument

If specified, argument skipCount skips the specified number of types returned to be that size or less.  The repository may choose a lower skipCount than specified due to its configuration.

Valid values: Zero or Positive integer

Default: 0

##### 3.4.4.2.1.1.4 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

##### 3.4.4.2.1.1.5 includeAllowableActions Argument

If specified, argument includeAllowableActions includes the allowable actions of the current user context in the atom entry.

Valid values: true, false

Default: false

##### 3.4.4.2.1.1.6 includeRelationships Argument

If specified, argument includeRelationships includes the allowable actions of the current user context in the atom entry.

Valid values: enumIncludeRelationships

Default: none

```
	<xs:simpleType name="enumIncludeRelationships">
		<xs:restriction base="xs:string">
			<xs:enumeration value="none" />
			<xs:enumeration value="source" />
			<xs:enumeration value="target" />
			<xs:enumeration value="both" />
		</xs:restriction>
	</xs:simpleType>
```

### 3.4.4.3 Folder Children Collection

This is a pseudo-object comprised of all the direct children of a particular folder.  This is exposed as 'children'.

Document Type: Atom Feed

CMIS Services:

GET:  getChildren

PUT: Not Supported

POST:

createDocument (POST to folder)

createFolder (POST to folder)

createPolicy

or moveObject

or  addDocument-ToFolder (Entry)

DELETE: Not Supported

### 3.4.4.3.1 HTTP Methods

#### 3.4.4.3.1.1 GET

##### 3.4.4.3.1.1.1 orderBy Argument

If argument supplied, the argument defines the order in which the results must be returned.

Valid values: must be a valid part of the ORDER BY clause in Query. E.g., ‘creationdate ASC, creator DESC’

Default: repository specific

##### 3.4.4.3.1.1.2 maxItems Argument

If specified, argument maxitems limits the types returned to be that size or less. The repository may choose a lower maxItems than specified due to its configuration.

Valid values: Positive integer

Default: repository specific

##### 3.4.4.3.1.1.3 skipCount Argument

If specified, argument skipCount skips the specified number of types returned to be that size or less. The repository may choose a lower skipCount than specified due to its configuration.

Valid values: Zero or Positive integer

Default: 0

##### 3.4.4.3.1.1.4 childTypes Argument

If specified, argument depth limits the types returned to be within a certain depth of the specified folder. The repository may choose a lower depth than specified due to its configuration.

Valid values: enumTypesOfFileableObjects

Default: 1

```
	<xs:simpleType name="enumTypesOfFileableObjects">
		<xs:restriction base="xs:string">
			<xs:enumeration value="documents" />
			<xs:enumeration value="folders" />
			<xs:enumeration value="policies" />
			<xs:enumeration value="any" />
		</xs:restriction>
	</xs:simpleType>
```

##### 3.4.4.3.1.1.5 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

#### 3.4.4.3.1.1.6 includeAllowableActions Argument

If specified, argument includeAllowableActions includes the allowable actions of the current user context in the atom entry.

Valid values: true, false

Default: false

#### 3.4.4.3.1.1.7 includeRelationships Argument

If specified, argument includeRelationships includes the allowable actions of the current user context in the atom entry.

Valid values: enumIncludeRelationships

Default: none

```
	<xs:simpleType name="enumIncludeRelationships">
		<xs:restriction base="xs:string">
			<xs:enumeration value="none" />
			<xs:enumeration value="source" />
			<xs:enumeration value="target" />
			<xs:enumeration value="both" />
		</xs:restriction>
	</xs:simpleType>
```

#### 3.4.4.3.1.2 POST

POST follows Atom Publishing Protocol for adding items to collections.

If the atom entry has an object id, the object will be added to the folder. If not, it will be created and then added to the folder

A content stream must be specified regardless of versioningState if required by the type definition. If not provided and it is required, ContentStreamNotProvided exception will be thrown.

The behavior is repository specific when a non-(atom/cmis)-entry document is posted to a folder URI. For example, the repository MAY auto-create a document with a specific type (document) the client could edit. The repository MAY also throw an exception.

Posting the content stream for a document is a separate operation, using the edit-media link in the response.

If an OID for an existing document is specified, then the document will be filed into the specified folder.

#### 3.4.4.3.1.2.1 versioningState Argument

If specified, argument versioningState controls on new document creation whether or not to checkin the document.

Valid values: enumIncludeRelationships

Default: major

```
      <xs:simpleType name="enumVersioningState">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="checkedout" />
                  <xs:enumeration value="minor" />
                  <xs:enumeration value="major" />
            </xs:restriction>
      </xs:simpleType>
```

###### 3.4.4.3.1.2.2 removeFrom Argument

If specified, remove the POST'ed entry from the folder specified.  If the Atom Entry POST'ed, does not have an ID, document does not exist, or the document is not in that folder, an exception is thrown.

Valid values: opaque id (string)

Default: not specified

###### 3.4.4.3.1.2.3 thisVersion Argument

If specified, the document version to be filed will be fixed at the versionPOST'ed to the folder or will be updated to reflect the latest version of the document.

Valid values: true, false

Default: false

#### 3.4.4.4 Folder Descendants Collection

This is a pseudo-object comprised of all the direct and indirect children of a particular folder.  This is exposed as 'descendants'

Document Type: Atom Feed

CMIS Services:

GET:  getDescendants

PUT: Not Supported

POST:

createDocument (POST to folder)

createFolder (POST to folder)

createPolicy

or moveObject

or  addDocument-ToFolder (Entry)

DELETE: deleteTree

##### 3.4.4.4.1 HTTP Methods

###### 3.4.4.4.1.1  GET

###### 3.4.4.4.1.1.1 orderBy Argument

If argument supplied, the argument defines the order in which the results must be returned.

Valid values:  must be a valid part of the ORDER BY clause in Query.  E.g., 'creationdate ASC, creator DESC'

Default: repository specific

#### 3.4.4.4.1.1.2 depth Argument

If specified, argument depth limits the types returned to be within a certain depth of the specified folder. The repository may choose a lower depth than specified due to its configuration.

Valid values: Positive integer

Default: 1

#### 3.4.4.4.1.1.3 childTypes Argument

If specified, argument depth limits the types returned to be within a certain depth of the specified folder. The repository may choose a lower depth than specified due to its configuration.

Valid values: enumTypesOfFileableObjects

Default: 1

```
	<xs:simpleType name="enumTypesOfFileableObjects">
		<xs:restriction base="xs:string">
			<xs:enumeration value="documents" />
			<xs:enumeration value="folders" />
			<xs:enumeration value="policies" />
			<xs:enumeration value="any" />
		</xs:restriction>
	</xs:simpleType>
```

#### 3.4.4.4.1.1.4 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

#### 3.4.4.4.1.1.5 includeAllowableActions Argument

If specified, argument includeAllowableActions includes the allowable actions of the current user context in the atom entry.

Valid values: true, false

Default: false

#### 3.4.4.4.1.1.6 includeRelationships Argument

If specified, argument includeRelationships includes the allowable actions of the current user context in the atom entry.

Valid values: enumIncludeRelationships

Default: none

```
	<xs:simpleType name="enumIncludeRelationships">
		<xs:restriction base="xs:string">
			<xs:enumeration value="none" />
			<xs:enumeration value="source" />
			<xs:enumeration value="target" />
			<xs:enumeration value="both" />
		</xs:restriction>
	</xs:simpleType>
```

##### 3.4.4.4.1.2 POST

POST follows Atom Publishing Protocol for adding items to collections.

If the atom entry has an object id, the object will be added to the folder.  If not, it will be created and then added to the folder

A content stream must be specified regardless of versioningState if required by the type definition.  If not provided and it is required, ContentStreamNotProvided exception will be thrown.

The behavior is repository specific when a non-(atom/cmis)-entry document is posted to a folder URI.  For example, the repository MAY auto-create a document with a specific type (document) the client could edit.  The repository MAY also throw an exception.

Posting the content stream for a document is a separate operation, using the edit-media link in the response.

If an OID for an existing document is specified, then the document will be filed into the specified folder.

###### 3.4.4.4.1.2.1 versioningState Argument

If specified, argument versioningState controls on new document creation whether or not to checkin the document.

Valid values: enumIncludeRelationships

Default: major

```
      <xs:simpleType name="enumVersioningState">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="checkedout" />
                  <xs:enumeration value="minor" />
                  <xs:enumeration value="major" />
            </xs:restriction>
      </xs:simpleType>
```

###### 3.4.4.4.1.2.2 removeFrom Argument

If specified, remove the POST'ed entry from the folder specified.  If the Atom Entry POST'ed, does not have an ID, document does not exist, or the document is not in that folder, an exception is thrown.

Valid values: opaque id (string)

Default: not specified

###### 3.4.4.4.1.2.3 thisVersion Argument

If specified, the document version to be filed will be fixed at the versionPOST'ed to the folder or will be updated to reflect the latest version of the document.

Valid values: true, false

Default: false

##### 3.4.4.4.1.3 DELETE

This removes the entire tree.

#### 3.4.4.4.1.3.1 continueOnFailure Argument

If specified true, the repository will continue deleting items in the folder when an error is encountered.

Valid values: true, false

Default: false

#### 3.4.4.4.1.3.2 unfileMultiFiledDocuments Argument

If specified, the argument controls how objects are removed from folders.

Valid values: enumUnfileNonfolderObjects

Default: deletesinglefiled

```
      <xs:simpleType name="enumUnfileNonfolderObjects">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="unfile" />
                  <xs:enumeration value="deletesinglefiled" />
                  <xs:enumeration value="delete" />
            </xs:restriction>
      </xs:simpleType>
```

### 3.4.4.5 AllVersions Collection

This is a pseudo-object comprised of all the direct and indirect children of a particular folder.  This is exposed as 'descendants'

Document Type: Atom Feed

CMIS Services:

GET:  getAllVersions

PUT: Not Supported

POST: Not Supported

DELETE: deleteAllVersions

### 3.4.4.5.1 HTTP Methods

#### 3.4.4.5.1.1  GET

#### 3.4.4.5.1.1.1 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

#### 3.4.4.5.1.2  DELETE

This removes the entire version history of the document.

## 3.4.5 Entries

### 3.4.5.1 Type Entry

This represents the type of an object.

This is enclosed as an atom entry

Document Type: Atom Entry

CMIS Services:

GET:  getTypeDefinition

PUT: Not Supported

POST: Not Supported

DELETE: Not Supported

#### 3.4.5.1.1 HTTP Methods

##### 3.4.5.1.1.1 GET

###### 3.4.5.1.1.1.1 filter Argument

If specified, argument filter limits the property definition returned.

Valid values: Comma separated list of property names

Default: Repository specific

### 3.4.5.2 Document Entry

This is a CMIS Document instance.  These are exposed in feeds or in an entry document.

This can also be a private working copy (checkedout)

Document Type: Atom Entry

CMIS Services:

GET:  getProperties

PUT: updateProperties

POST: Not Supported

DELETE: Delete [This specific version]

#### 3.4.5.2.1 HTTP Methods

##### 3.4.5.2.1.1 GET

###### 3.4.5.2.1.1.1 returnVersion Argument

If specified, the repository will return the version specified.

Valid values: enumReturnVersion

Default: this

```
	<xs:simpleType name="enumReturnVersion">
		<xs:restriction base="xs:string">
			<xs:enumeration value="this" />
			<xs:enumeration value="latest" />
			<xs:enumeration value="latestmajor" />
		</xs:restriction>
	</xs:simpleType>
```

###### 3.4.5.2.1.1.2 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

###### 3.4.5.2.1.2 PUT

This does a complete replacement of the atom entry with the atom entry document specified. If properties are not included, they will be not set.

###### 3.4.5.2.1.3 DELETE

This removes the document entry.

#### 3.4.5.3 Document Private Working Copy (PWC) Entry

This is a document in the private working copy of the document (checkedout version of document)

Document Type: Atom Entry

CMIS Services:

GET: getProperties

PUT: updateProperties or checkin

POST: Not Supported

DELETE: cancelCheckout

##### 3.4.5.3.1 HTTP Methods

###### 3.4.5.3.1.1 GET

###### 3.4.5.3.1.1.1 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

###### 3.4.5.3.1.2 PUT

This does a complete replacement of the atom entry with the atom entry document specified. If properties are not included, they will be not set.

###### 3.4.5.3.1.2.1 checkinComment Argument

If specified, this includes the checkinComment for the checkin operation. If specified and checkin is not set, an exception is thrown.

Valid values: String

Default: not Set

###### 3.4.5.3.1.2.2 major Argument

If specified, specifies whether the checked in version should be major or minor.

Valid values: true, false

Default: true

#### 3.4.5.3.1.2.3 checkin Argument

If specified, specifies whether this PUT should be treated as checkin

Valid values: true, false

Default: false

#### 3.4.5.3.1.3 DELETE

This removes the document entry, in this case, cancels the check out. The PWC will be removed.

### 3.4.5.4 Folder Entry

This is a CMIS Folder instance. This is exposed as a feed. The properties of a folder map onto the feed tag.

Document Type: Atom Entry

CMIS Services:

GET: getProperties

PUT: updateProperties

POST: Not Supported

DELETE: Delete

### 3.4.5.4.1 HTTP Methods

#### 3.4.5.4.1.1 GET

#### 3.4.5.4.1.1.1 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

#### 3.4.5.4.1.2 PUT

This does a complete replacement of the atom entry with the atom entry document specified. If properties are not included, they will be not set.

#### 3.4.5.4.1.3 DELETE

This removes the object from the repository.

### 3.4.5.5 Relationship Entry

This is a CMIS relationship instance. These objects are exposed via ‘relationships’ link type.

Document Type: Atom Entry

CMIS Services:

GET: getProperties

PUT: updateProperties

POST: Not Supported

DELETE: Delete

#### 3.4.5.5.1 HTTP Methods

##### 3.4.5.5.1.1 GET

###### 3.4.5.5.1.1.1 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

##### 3.4.5.5.1.2 PUT

This does a complete replacement of the atom entry with the atom entry document specified.  If properties are not included, they will be not set.

##### 3.4.5.5.1.3 DELETE

This removes the document entry.

### 3.4.5.6 Policy Entry

This is a CMIS policy instance.  This is exposed via 'policies' as a feed.

Document Type: Atom Entry

CMIS Services:

GET:  getProperties

PUT: updateProperties

POST: Not Supported

DELETE: Delete

#### 3.4.5.6.1 HTTP Methods

##### 3.4.5.6.1.1 GET

###### 3.4.5.6.1.1.1 filter Argument

If specified, argument filter limits the properties.

Valid values: Comma separated list of property names

Default: Repository specific

##### 3.4.5.6.1.2 PUT

This does a complete replacement of the atom entry with the atom entry document specified.  If properties are not included, they will be not set.

##### 3.4.5.6.1.3 DELETE

This removes the document entry.

### 3.4.5.7 Content Stream

This is the content stream portion of the document object.  This is exposed via 'stream' on the entry

Document Type: Mime/Type of Content Stream

CMIS Services:

GET:  getContentStream

PUT: setContentStream

POST: Not Supported

DELETE: deleteContentStream

### 3.4.5.7.1 HTTP Methods

#### 3.4.5.7.1.1  GET

#### 3.4.5.7.1.2  PUT

This does a replacement of the content stream.  If the document is not checked out, an exception will be thrown.

#### 3.4.5.7.1.3  DELETE

This removes the content stream.  If the document is not checked out, an exception will be thrown.

# 4 IANA Considerations

| Document Type | Description | MimeType | Namespace | Starting Tag |
|---|---|---|---|---|
| Query | This is a CMIS Query document | Application/cmisquery+xml | CMIS | query |
| AllowableActions | This is a CMIS AllowableActions document | Application/cmisallowableactions+xml | CMIS | allowableactions |

## 4.1 Document Types

### 4.1.1 Query

A CMIS Query Document, when serialized as XML 1.0, can be identified with the following media type:

MIME media type name: application

MIME subtype name: cmisquery +xml

Mandatory parameters: None.

Optional parameters:

"charset": This parameter has semantics identical to the charset parameter of the "application/xml" media type as specified in [RFC3023].

Encoding considerations:

Identical to those of "application/xml" as described in [RFC3023], Section 3.2.

Security considerations: As defined in this specification.

In addition, as this media type uses the "+xml" convention, it shares the same security considerations as described in [RFC3023], Section 10.

Interoperability considerations:

There are no known interoperability issues.

Published specification: This specification.

Applications that use this media type:

No known applications currently use this media type.

Additional information:

Magic number(s):

As specified for "application/xml" in [RFC3023], Section 3.2.

File extension: .cmisquery

Fragment identifiers:

As specified for "application/xml" in [RFC3023], Section 5.

Base URI:

As specified in [RFC3023], Section 6.

Macintosh File Type code: TEXT

Person and email address to contact for further information:

Al Brown <albertcbrown@us.ibm.com>

Intended usage: COMMON

Author/Change controller:  IESG

### 4.1.2 Allowable Actions

A CMIS Allowable Actions Document, when serialized as XML 1.0, can be identified with the following media type:

MIME media type name:  application

MIME subtype name:  cmisallowableactions +xml

Mandatory parameters:  None.

Optional parameters:

"charset":  This parameter has semantics identical to the charset parameter of the "application/xml" media type as specified in [RFC3023].

Encoding considerations:

Identical to those of "application/xml" as described in [RFC3023], Section 3.2.

Security considerations:  As defined in this specification.

In addition, as this media type uses the "+xml" convention, it shares the same security considerations as described in [RFC3023], Section 10.

Interoperability considerations:

There are no known interoperability issues.

Published specification:  This specification.

Applications that use this media type:

No known applications currently use this media type.

Additional information:

Magic number(s):

As specified for "application/xml" in [RFC3023], Section 3.2.

File extension:  .cmisquery

Fragment identifiers:

As specified for "application/xml" in [RFC3023], Section 5.

Base URI:

As specified in [RFC3023], Section 6.

Macintosh File Type code:  TEXT

Person and email address to contact for further information:

Al Brown <albertcbrown@us.ibm.com>

Intended usage:  COMMON

Author/Change controller:  IESG

## 4.2 Link Registration

### 4.2.1 parents

Attribute Value:  edit-media

Description:

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.2 parents

Description:

This relation can be used to specify the parents of the object including the folders the object is filed into.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.3 repository

Description:

This relation can be used to specify the repository of the object.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.4 children

Description:

This relation can be used to specify the children of the object.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.5 descendants

Description:

This relation can be used to specify the descendants of the object which includes children and their children.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.6 allowableactions

Description:

This relation can be used to specify the allowableactions of an object.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.7 allversions

Description:

This relation can be used to specify all versions of the object, typically in a feed.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.8 latestversion

Description:

This relation can be used to specify the latest version of the object.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.9 relationships

Description:

This relation can be used to specify the relationships of the object.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.10 type

Description:

This relation can be used to specify the type of the object.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.11 source

Description:

This relation can be used to specify the source of the object.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.12 target

Description:

This relation can be used to specify the target of the object.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

### 4.2.13 stream

Description:

This relation can be used to specify the stream associated with the object.

Expected display characteristics:

Undefined; this relation can be used for background processing or to provide extended functionality without displaying its value.

Security considerations:

Automated agents should take care when this relation crosses administrative domains (e.g., the URI has a different authority than the current document).

# # Conformance

The last numbered section in the specification must be the Conformance section. Conformance Statements/Clauses go here.

# A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

**Participants:**

[Participant Name, Affiliation | Individual Member]
[Participant Name, Affiliation | Individual Member]

# B. Non-Normative Text

## B.1 Atom Link Relations by Object Type

| ATOM Link | Type | Document | Folder | Relationship | Policy |
|---|---|---|---|---|---|
| self | X | X | X | X | X |
| alternate | | X | | | |
| edit | | X | X | X | X |
| edit-media | | X | | | |

## B.2 Atom and APP Extensions

The following extensions to Atom and APP have been leveraged in this specification:

1. Entries can contain Entries (for descendants)
2. Entries contain CMIS Type and CMIS Object information
3. Link relations have been extended
4. Links have been extended to add an cmis:id attribute
5. APP Service document has been extended
    a. CmisRepositoryInfo added to workspace
    b. Collections have cmis coltype

## B.3 Examples

Each of the following resources have an example as defined below:

| Resource (Root Tag) | Description | Example |
|---|---|---|
| Repository (Service) | This is the repository logical object. It is represented by the atom service document. This is also exposed on entry as link 'cmis-repository' | Please see Example-Service.xml |
| Root Folder Collection (Feed) | This is a collection described in the service document | Please see Example-FolderChildren.xml. |
| Checked Out Collection (Feed) | This is a collection described in the service document that contains all the checkedout documents | Please see Example-FolderChildren.xml. This will however, be a feed of private working copy document's only. |
| Unfiled Collection (Feed) | This is a collection of unfiled documents. | Please see Example-FolderChildren.xml. However there should be nothing returned. |
| Types Collection (Feed) | This is a collection described in the service document that contains all the types in the repository | |

| | | |
|---|---|---|
| Type<br>(Type) | This is the CMIS type of the object. This is exposed as link ‘cmis-type’, | Please see Example-Type.xml. |
| Document<br>(Entry) | This is a CMIS Document instance. These are exposed in feeds or in an entry document.<br>This can also be a private working copy (checkedout) | Please see Example-DocumentEntry.xml. |
| Document Private Working Copy<br>(Entry) | This is a document in the private working copy of the document (checkedout version of document) | Please see Example-DocumentPWCEntry.xml. |
| Folder<br>(Entry) | This is a CMIS Folder instance.  This is exposed as a feed.  The properties of a folder map onto the feed tag. | Please see Example-FolderEntry.xml. |
| Relationship<br>(Entry) | This is a CMIS relationship instance. These objects are exposed via ‘cmis-relationships’ | Please see Example-Relationship.xml. |
| Policy<br>(Entry) | This is a CMIS policy instance.  This is exposed via ‘cmis-policies’ as a feed | Please see Example-PolicyEntry.xml |
| Content Stream<br>(Non-XML) | This is the content stream portion of the document object.  This is exposed via ‘cmis-stream’ on the entry | No example.  This is the original document format. |
| Allowable Actions<br>(Allowable-Actions) | This is the set of actions available to the current user on the current object.  This is exposed as a link ‘cmis-allowableactions’ | Please see Example-AllowableActions.xml |
| Relationships Collection (for a specific item)<br>(Feed) | This is the set of relationships available (either source or target or both) from a specific item such as  a document, folder or policy | Please see FolderChildren.  However, the entries will be Relationships as in the RelationshipEntry example. |
| Parents Collection (for a specific document or policy object)<br>(Feed) | This is the set of parents for a specific document or policy object.  This can also be the ancestor of folders if returnToRoot is selected | Please see FolderChildren.  However, the entries will be Folder as in the FolderEntry Example |
| Children<br>(Feed) | This is a pseudo-object comprised of all the direct children of a particular folder.  This is exposed as ‘cmis-children’ | Please see FolderChildren. |
| Descendants | This is a pseudo-object comprised of all the direct and indirect children of | Please see FolderDescendants. |

| | | |
|---|---|---|
| (Feed) | a particular folder. This is exposed as ‘cmis-descendants’ | |
| AllVersions (Feed) | This is a pseudo-object made up of all document versions related to the current document version. This collection is also known as the version history | Please see FolderChildren. However, the entries will be Documents as in DocumentEntry and DocumentPWCEntry. |

## B.4 Expressing multiple content streams in REST

CMIS does not currently support multiple content streams in the specification due to complexity and lack of support across a set of repositories. However, exposing the multiple content streams that already exist in a repository can be done quite readily with REST.

Inside the ATOM entry section:

```
            <link rel=”stream” foo:streamnumber=3
             href="http://example.org/media/atom03">
```

The same can be done in the cmis:object tag as well. The foo:streamnumber attribute tag exposes the stream id. Non-aware applications will see many links of relationship cmis-stream. If they are aware, they can chose which stream they want to retrieve.

For setting multiple content streams, this must be done outside of CMIS today.

# C. Revision History

| Revision | Date | Editor | Changes Made |
|---|---|---|---|
| [Rev number] | [Rev Date] | [Modified By] | [Summary of Changes] |