1 Browser Binding

1.1 Overview

The CMIS Browser Binding is intended to make it simpler for browser-based applications to find, create, update and delete content stored in CMIS repositories. Also this binding is optimized for use in browser applications, it can also be useful as a simpler HTTP based binding in other application models.

1.2 Common Service Elements

1.2.1 Protocol

HTTP shall be used as the protocol for service requests. HTTP GET shall be used for reading content and HTTP POST shall be used for creating, updating and deleting content.

1.2.2 Data Representation

Browser applications are typically written in JavaScript and a popular lightweight data representation format amongst JavaScript developers is JavaScript Object Notation (JSON) as described in RFC 4627 (see http://www.ietf.org/rfc/rfc4627.txt). So in this binding JSON shall be used to represent CMIS repositories, folders, documents, relationships and policies.

1.2.2.1 Mapping Schema Elements to JSON

JSON only defines a few types, including Object, String, Number, Boolean, Null, and Arrays. Since not all the types used in the CMIS schema have direct JSON equivalents, some explanation of mapping is necessary.

CMIS	JSON
string	string
boolean	boolean
decimal	number
integer	number
datetime	string (xsd:dateTime format)
uri	string (unencoded url)
id	string
html	string

1.2.3 Referencing Resources by Path and by Id

Resources can be referenced using either the full path or by Id.

When path is used, URI used to address the resource shall be relative to the `rootFolderUri` returned from the getRepositoryInfo service.

In the case of Id, the request parameter `id` shall be used.

Examples

Referencing an object by path.

```
GET /cmis/repository/123/myFolder/myDocument
```

Referencing an object by id.

```
GET /cmis/repository/123?id=0192018282
```

1.2.4 Paging

Since the number of objects returned from the navigation services can be huge, a mechanism for paging is provided. The optional input parameters defined on the navigation services, `maxItems` and `skipCount`, shall be represented as HTTP request parameters of the same name.

A JSON object with the key "`cmis:pageinfo`" shall be present if the request parameter `maxItems` is present. This object shall have the following JSON key/value pairs.

boolean `hasMoreItems`

number `numItems`

See the description in Section 2.2.1.1 for the meaning of these parameters.

Example:

```
GET /cmis/repository/123/myFolder?maxItems=10 HTTP/1.1
   Host: www.example.com
   User-Agent: Mozilla/5.0

{cmis:pageinfo":
  {
  "hasMoreItems": true,
  "numItems: 127
  }
}
 . . .
```

1.2.5 Multipart Forms

Browser applications also typically use HTTP multipart forms as described in RFC 2388 (see http://tools.ietf.org/html/rfc2388) to create and update content. This is especially useful for updating file content with the addition of the FILE attribute in RFC 1867 (see http://tools.ietf.org/html/rfc1867). In this binding, HTTP POST of multipart/form-data shall be to update content streams, and perform multiple operations in a single POST.

1.2.6 Namespaces

JSON does not need to define namespaces since the scope of JSON object allows key/value pairs within it to be uniquely referenced, but when it is becomes necessary to uniquely JSON objects because they may otherwise conflict with repository-defined values, the prefix "`cmis:`" shall be used to identify objects

unique to this specification. For example, it is necessary to include the prefix "`cmis:`" in the JSON key name for "`cmis:name`", for otherwise this could conflict with a repository defined property with a key of "`name`" within the JSON object with a key of "`properties`".

But when not necessary to uniquely define a JSON object, the prefix shall be omitted so conserve space. For example, "`cmis:read`" can be expressed as "`read`" in a JSON array since the array element will only be referenced in the context of a specific CMIS property. As another example, it is not necessary to use the prefix of "`cmis:`" in the JSON object with a key of "`properties`" since that key is already unique within its containing object.

***QUESTION*:** Is it worth the bother to make this distinction or should we just use the "`cmis:`" prefix whenever we reference a CMIS schema element?

***COMMENT*:** We could get more savings by eliminating the prefix by using more JSON nesting, e.g. use a "`cmis:properties`" key for a object to hold all the properties defined by the specification and a sibling object with a key of "`properties`" to hold all the other properties.

1.2.7 Authentication

Authentication SHOULD be handled by the transport protocol.

1.2.8 Return Codes

HTTP Return Codes shall be used to indicate success or failure of an operation. Please see the HTTP specification for more information on the HTTP status codes. These are provided as guidance from the HTTP specification. If any conflict arises, the HTTP specification is authoritative.

CMIS Services Exception	HTTP Status Code
invalidArgument	400
objectNotFound	404
permissionDenied	403
notSupported	405
runtime	500
constraint	409
filterNotValid	400
streamNotSupported	403
storage	500
contentAlreadyExists	409
versioning	409
updateConflict	409
nameConstraintViolation	409

1.3 Services

1.3.1 Repository Services

1.3.1.1 getRepositories, getRepositoryInfo

An HTTP GET operation shall be used to obtain a list of CMIS repositories, and information about each.

***QUESTION*:** do we want to treat these into separate services so a client can get just a simple list of repositories without all the detailed properties?

1.3.1.1.1 Inputs

None

1.3.1.1.2 Outputs

A set of JSON objects representing repositories. Each repository is represented as a JSON object, with a key of `repositoryId` and key/value pairs representing each of the properties defined in the `cmisRepositoryInfoType` schema.

1.3.1.1.3 Exceptions Thrown & Conditions

See section “General Exceptions”

1.3.1.1.4 Example

```
GET /cmis/repositories HTTP/1.1
   Host: www.example.com
   User-Agent: Mozilla/5.0

{"cmis:repositories" :
  {
  "repository123" :
    {
    “repositoryID” : “repository123”,
    “repositoryName”, “Repository 123”,
    “repositoryDescription”, “This is a CMIS repository”,
    “vendorName” : “Foo Inc.”,
    “productName” : “Product Bar”,
    “productVersion” : “1.0”,
    “rootFolderId” : “123”,
    “rootFolderUri” : “http://www.example.com/cmis/repository/123”,
    “latestChangeLogToken” : “0123456789”,
    “thinClientURI” : “http://www.example.com/web/index.html”,
    “changesIncomplete” : true,
    “changesOnType” :
       [
       “cmis:document”,
       “cmis:folder”
       ]
    “supportedPermissons” : “basic”,
    “propagation” : “objectonly”,
    “permission” : [“read”, “write”, “all”],
    “permissionMapping” :
    [
      [“canMoveObject.Source”,[“write”,”read”]],
      [“canDelete.Object, [“write”]
    ]
    “principalAnonymous” : “anon”,
    “principalAnyone” : “any”
    }
  }
{
  "repository456" :
    {
    “repositoryID” : “repository456”,
    “repositoryName”, “Repository 456”,
    “repositoryDescription”, “This is another CMIS repository”,
    “vendorName” : “Bar Inc.”,
    “productName” : “Product Foo”,
    “productVersion” : “1.0”,
```

```
    “rootFolderId” : “456”,
    “rootFolderUri” : “http://www.example.com/cmis/repository/456”,
    “latestChangeLogToken” : “9876543210”,
    “thinClientURI” : “http://www.example.com/web/index.html”,
    “changesIncomplete” : true,
    “changesOnType” :
       [
       “cmis:document”,
       “cmis:folder”
       ]
    “supportedPermissons” : “basic”,
    “propagation” : “objectonly”,
    “permission” : [“read”, “write”, “all”],
    “permissionMapping” :
    [
      [“canMoveObject.Source”,[“write”,”read”]],
      [“canDelete.Object, [“write”]
    ]
    “principalAnonymous” : “anon”,
    “principalAnyone” : “any”
    }
  }
}
```

1.3.1.2 getTypeChildren

1.3.1.3 getTypeDescendants

1.3.1.4 getTypeDefinition

1.3.2 Navigation Services

1.3.2.1 getChildren

An HTTP GET operation shall be used to obtain a list of the child folders and documents of a folder.

1.3.2.1.1 Inputs

Required The resource on which the GET operation is performed shall be specified as an id, or as a path. See section 1.2.3 for a description of how the id and path shall be specified.

Optional See section 2.2.3.1.1 for a list of the optional parameters for this service. When used, each parameter shall be specified as a request parameter.

1.3.2.1.2 Outputs

A set of JSON objects representing folders and documents. Each child resource is represented as a JSON object, with a key of the value of the “`cmis:name`” property of the child resource, with key/value pairs in the object representing the properties of the child resource.

1.3.2.1.3 Exceptions Thrown & Conditions

See section “General Exceptions”

1.3.2.1.4 Example

```
GET /cmis/repository/123/myFolder?includePathSegment” HTTP/1.1
   Host: www.example.com
   User-Agent: Mozilla/5.0
```

```
{
"myChildFolder 1" :
  {
  “properties” :
    {
      “cmis:objected” : “f1234567890”,
      “cmis:baseTypeId” : “cmis:folder”,
      “cmis:objectTypeId” : “personnelRecords”,
      “cmis:createdBy” : “Gregory Melahn”,
      “cmis”creationDate” : “2010-07-16T03:10:32.088-05:00”,
      “cmis”lastModifiedBy” : “David Neuscheler”,
      “cmis”lastModificationDate” : “2010-07-22T05:18:52.018-05:00”,
      “cmis:changeToken” : ‘1234567890”,
      “cmis:parentId” : “f9876543210”,
      “cmis:path” : “/myChildFolder”,
      “cmis:allowedChildObjectTypeIds” :
      [
        “Type012”, “Type345”
      ]
      “cmis:pathSegment” : “myChildFolder”
    }
  }
}
{
“myChildDocument” :
  {
  “properties” :
    {
      “cmis:objected” : “d1234567890”,
      “cmis:baseTypeId” : “cmis:document”,
      “cmis:objectTypeId” : “personnelRecord”,
      “cmis:createdBy” : “Ryan McVeigh”,
      “cmis”creationDate” : “2010-07-16T03:10:32.088-05:00”,
      “cmis”lastModifiedBy” : “David Neuscheler”,
      “cmis”lastModificationDate” : “2010-07-22T05:18:52.018-05:00”,
      “cmis:changeToken” : ‘1234567890”,
      “cmis:parentId” : “f9876543210”,
      “cmis:path” : “/myFolder/MyChildDocument”,
      “cmis:isUmmutable” : false,
      “cmis:isLatestVersion” : true,
      “cmis:isMajorVersion” : false,
      “cmis:isLatestMajorVersion” : false,
      “cmis:versionLabel” : “1.1”,
      “cmis:versionSeriesId” : “v001”,
      “cmis:isVersionSeriesCheckedOut” : true,
      “cmis:versionSeriesCheckedOutBy” : “Gregory Melahn”,
      “cmis: versionSeriesCheckedOutId” : “v001abc”,
      “cmis:checkinComment” : “this is a  change”,
      “cmis:contentStreamLength” : 1002991,
      “cmis:contentStreamMimeType” : “application/pdf”,
      “cmis:contentStreamFileName” : “doc001.pdf”,
      “cmis:contentStreamUri” :
“http://www.example.com/cmis/repository/123/myFolder/doc001.pdf” ,
    }
  }
}
```