Content Management Interoperability Services (CMIS) Extension

CMIS Endpoints Document (Draft)

00 May 2015

Latest version:

http://…..

Technical Committee:

[OASIS Content Management Interoperability Services (CMIS) TC](http://www.oasis-open.org/committees/cmis/)

Author:

Florian Müller <florian.mueller02@sap.com>

Extends:

CMIS 1.0 and CMIS 1.1
This extension will be part of subsequent CMIS specification versions.

Abstract:

The CMIS Endpoints Document allows CMIS clients to discover endpoints of a CMIS server. It also contains information about the supported authentication methods, cookie requirements, and CSRF protection.

# Motivation

The CMIS specification assumes that a CMIS client knows the CMIS endpoint and knows how to authenticate a user. CMIS does not define any discovery mechanism. The CMIS Endpoints Document should allow CMIS clients to find all endpoints of a CMIS server, present them to a user or administrator, and simplify the configuration of the connection.

# Specification

It is not defined how a CMIS client gets the CMIS Endpoints Document. In most case, this is a manual configuration.

The content of the CMIS Endpoints Document is specified as follows:

* The CMIS Endpoints Document is a JSON document.
	+ The MIME type MUST be “application/json”.
	+ The encoding MUST be UTF-8.
	+ The document SHOULD be named “cmis-endpoints.json”.
	+ The JSON document MAY contain not specified, server specific name-value pairs anywhere in the document. The names of those name-value pairs SHOULD have a server/implementation/company prefix to prevent collisions with subsequent versions of this extension.
* The top-level element is a JSON object. It has one entry “endpoints”, which value is an array that contains the CMIS endpoints.
* A JSON object represents each endpoint with the following name-value pairs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Required** | **Values** | **Descriptions** |
| displayName | String | No |  | Human readable name of the endpoint. |
| cmisVersion | String | Yes | “1.0”, “1.1” | CMIS version of the endpoint. |
| binding | String | Yes | “webservices”, “atompub”, “browser” | CMIS binding of the endpoint. |
| url | String | Yes, for the AtomPub and Browser Binding | URL | URL of the endpoint.This is the WSDL URL for the Web Services binding and the service documents for the AtomPub and the Browser bindings. |
| repositoryServiceWdsl | String | No | WSDL URL | WSDL URL of the repository service. It MAY only be set for Web Service binding endpoints. It SHOULD only be set if the service WSDL URLs are different. If the “url” name-value pair is set, the “url” name-value pair takes precedence.  |
| navigationServiceWdsl | String | No | WSDL URL | WSDL URL of the navigation service. It MAY only be set for Web Service binding endpoints. It SHOULD only be set if the service WSDL URLs are different. If the “url” name-value pair is set, the “url” name-value pair takes precedence.  |
| objectServiceWdsl | String | No | WSDL URL | WSDL URL of the object service. It MAY only be set for Web Service binding endpoints. It SHOULD only be set if the service WSDL URLs are different. If the “url” name-value pair is set, the “url” name-value pair takes precedence.  |
| multifilingServiceWdsl | String | No | WSDL URL | WSDL URL of the mutlifiling service. It MAY only be set for Web Service binding endpoints. It SHOULD only be set if the service WSDL URLs are different. If the “url” name-value pair is set, the “url” name-value pair takes precedence.  |
| discoveryServiceWdsl | String | No | WSDL URL | WSDL URL of the discovery service. It MAY only be set for Web Service binding endpoints. It SHOULD only be set if the service WSDL URLs are different. If the “url” name-value pair is set, the “url” name-value pair takes precedence.  |
| versioningServiceWdsl | String | No | WSDL URL | WSDL URL of the versioning service. It MAY only be set for Web Service binding endpoints. It SHOULD only be set if the service WSDL URLs are different. If the “url” name-value pair is set, the “url” name-value pair takes precedence.  |
| relationshipServiceWdsl | String | No | WSDL URL | WSDL URL of the relationship service. It MAY only be set for Web Service binding endpoints. It SHOULD only be set if the service WSDL URLs are different. If the “url” name-value pair is set, the “url” name-value pair takes precedence.  |
| policyServiceWdsl | String | No | WSDL URL | WSDL URL of the policy service. It MAY only be set for Web Service binding endpoints. It SHOULD only be set if the service WSDL URLs are different. If the “url” name-value pair is set, the “url” name-value pair takes precedence.  |
| aclServiceWdsl | String | No | WSDL URL | WSDL URL of the ACL service. It MAY only be set for Web Service binding endpoints. It SHOULD only be set if the service WSDL URLs are different. If the “url” name-value pair is set, the “url” name-value pair takes precedence.  |
| soapVersion | String | No | “1.1” (default), “1.2” | SOAP version for Web Services binding endpoints. |
| cookies | String | No | “required”, “recommended”, “optional” (default) | Defines if the endpoint requires cookies, recommends the use of cookies (for example, for better performance), or doesn’t make use of cookies. (See section below for details.) |
| compression | String | No | “none” (default), “server”, “client”, “both” | Defines if the endpoint supports compression and in which direction. |
| csrfHeader | String | No | HTTP header name | The name of the CSRF header if CSRF protection is supported. If this header is set, it is assumed that cookies are supported and MUST be used. (See section below for details.) |
| csrfParameter | String | No | URL parameter name/form control name | The name of the CSRF URL parameter/form control name if CSRF protection is supported. (See section below for details.) |
| authentication | Array of objects | Yes | Array of authentication objects | A list of supported authentication methods. (See section below for details.)This list MAY be incomplete or empty if the authentication is not handled by the CMIS server but by a different component. |

* An authentication object consists of the following name-value pairs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Required** | **Values** | **Descriptions** |
| type | String | Yes | (see table below) | The authentication type. |
| displayName | String | No |  | Human readable name of the authentication type. |
| documentationUrl | String | No | A URL to the developer documentation, if available | A link to developer documentation, which explains the authentication method and provides required information for the authentication type. |
| preference | Integer | No | A positive integer, a lower number indicates a higher preference | A hint for the client, which endpoint and authentication type combination the server prefers. If two or more authentication objects have the same preference value, the server doesn’t prefer one or the other. Authentication objects without a preference value have the least priority. |

## Authentication Types

The following table contains the defined authentication methods. A server MAY provide additional, server specific types. The names of those types SHOULD have a server/implementation/company prefix.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Additional name-value pairs** | **Requires cookies**(If “Yes”, the “cookies” value is assumed as “required”.) | **Description** |
| none | - | No | No authentication. |
| basic | “charset”: (String, optional) user name and password encoding (default is “UTF-8”)  | No | HTTP basic authentication. |
| usernameToken | - | No | usernameToken authentication (Web Services only). |
| form | “loginUrl”: (String, optional) the URL of the login page | Yes | Form authentication. |
| certificate | - | No | SSL client certificate authentication. |
| saml | - | Yes | SAML 2.0 authentication. |
| oauth | - | No | OAuth 2.0 authentication. |
| oidc | - | No | OpenID Connect authentication. |
| ntlm | “version”: (String, optional) the NTML version (“NTLMv1” (default) or “NTMLv2”) | No | NTLM authentication. |
| kerberos | - | No | Kerberos authentication. |
| ltpa | - | Yes | LTPA authentication. |

The authentication type descriptions are deliberately vague because the implementation details and the additional data that is required to connect can be significantly different from one repository to another. Therefore, it is RECOMMENDED to provide a “documentationUrl” for each authentication type.

## Cookies

The CMIS specification doesn’t require a server implementation to manage any state. This stateless nature of CMIS allows building a very scalable server infrastructure and simplifies client development.

However, there are many good reasons for a CMIS server to manage state. That some authentication methods require state to work or to perform well is just one of them.

Cookies are a very common and well-supported mechanism for HTTP based protocols to enable state management. Therefore, the use of cookies as defined in RFC 6265 is RECOMMENDED if a server requires state management. Clients SHOULD also be able to handle cookies that follow earlier, now obsolete cookie specifications.

## CSRF Protection

The CMIS 1.1 specification describes only a method to prevent cross-site request forgery (a.k.a. CSRF or XSRF) attacks for the browser binding and only if the server supports tokens (section 5.2.9.2.1.).

CSRF protection for the other bindings and other authentication methods has not been defined and is repository specific. This CMIS specification extension defines a CSRF protection method that is generic and works for all bindings. It relies on an additional HTTP header and cookies.

If the repository supports this protection method, a “csrfHeader” name-value pair MUST exist in the endpoint JSON object, which defines the name of HTTP header that is used for transmitting a CSRF token.

Retrieving and using a CRSF token works as follows:

* Firstly, the client calls getRepositories() or getRepositoryInfo() and MUST send the CSRF header with the value “fetch”.
* The server adds a CSRF header that provides a token and a cookie to the response. The cookie SHOULD have the HttpOnly flag.
* Further requests from the client MUST provide the CSRF token and the cookie in the HTTP header. The server MUST respond to requests that lack the CSRF header or the cookie or provide wrong, invalid, or outdated values with a permissionDenied exception.
* The server MAY provide a CSRF header with a new token in any response. The client MUST use this new token for subsequent requests until the server sends a new token.

Providing an additional HTTP header can be difficult in web applications. For example, if the web applications wants to provide a CMIS download link (getContentStream()) it cannot set a header. For cases like this, the server MAY accept the CSRF token also as a URL parameter in HTTP GET requests and control values in HTTP POST requests. If the server supports this, it MUST provide a “csrfParameter” name-value pair in the endpoint JSON object. The value is the name of the URL parameter respectively the control name in an HTML form.

### Example:

The following sequence of requests and responses demonstrates the exchange of CSRF tokens and cookies. The example assumes that the “csrfHeader” is set to “X-CSRF-Token” and the “csrfParameter” is set to “x-token”.

#### 1. Call getRepositoryInfo() and request a token

Request:

***GET /docserv/browser***

**X-CSRF-Token: fetch**

Response:

***200 OK***

**X-CSRF-Token: 6FC66750-C999-11E4-ABE7-1E80DD9DA696**

**Set-Cookie: sessionid=23D8273D2BF7E304772F2902C93C7967; Path=/docserv;**

 **Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly**

#### 2. Call getObject()

Request:

***GET /docserv/browser/repo1/root?cmisSelector=object&objectId=D2082F62-C998-11E4-953C-1E80DD9DA696***

**X-CSRF-Token: 6FC66750-C999-11E4-ABE7-1E80DD9DA696**

**Cookie: sessionid=23D8273D2BF7E304772F2902C93C7967**

Response:

***200 OK***

#### 3. Call getChildren().

Request:

***GET /docserv/browser/repo1/root?cmisSelector=children&objectId=D2082F62-C998-11E4-953C-1E80DD9DA696***

**X-CSRF-Token: 6FC66750-C999-11E4-ABE7-1E80DD9DA696**

**Cookie: sessionid=23D8273D2BF7E304772F2902C93C7967**

Response:

**200 OK**

#### 4. Call getChildren() (server returns new token)

Request:

***GET /docserv/browser/repo1/root?cmisSelector=children&objectId=D2082F62-C998-11E4-953C-1E80DD9DA696***

**X-CSRF-Token: 6FC66750-C999-11E4-ABE7-1E80DD9DA696**

**Cookie: sessionid=23D8273D2BF7E304772F2902C93C7967**

Response:

***200 OK***

**X-CSRF-Token: 7CB7E010-C999-11E4-81F1-1E80DD9DA696**

#### 5. Call getObject() (with new token)

Request:

***GET /docserv/browser/repo1/root?cmisSelector=object&objectId=D2082F62-C998-11E4-953C-1E80DD9DA696***

**X-CSRF-Token: 7CB7E010-C999-11E4-81F1-1E80DD9DA696**

**Cookie: sessionid=23D8273D2BF7E304772F2902C93C7967**

Response:

***200 OK***

#### 6. Call getObject() (with old token)

Request:

***GET /docserv/browser/repo1/root?cmisSelector=object&objectId=D2082F62-C998-11E4-953C-1E80DD9DA696***

**X-CSRF-Token: 6FC66750-C999-11E4-ABE7-1E80DD9DA696**

**Cookie: sessionid=23D8273D2BF7E304772F2902C93C7967**

Response:

**403 Forbidden**

#### 7. Call getContent() (with token as URL parameter)

Request:

***GET /docserv/browser/repo1/root?cmisSelector=content&objectId=2E311184-CBEF-11E4-89CC-F18BDD9DA696&x-token=*7CB7E010-C999-11E4-81F1-1E80DD9DA696**

**Cookie: sessionid=23D8273D2BF7E304772F2902C93C7967**

Response:

***200 OK***

## Endpoints Document Example

**{**

 **"endpoints" : [**

 **{**

 **"displayName" : "DocServ CMIS 1.1 AtomPub Binding",**

 **"cmisVersion" : "1.1",**

 **"binding" : "atompub",**

 **"url" : "https://host:8080/cmis/atompub",**

 **"cookies" : "recommended",**

 **"compression" : "server",**

 **"csrfHeader" : "X-CSRF-Token",**

 **"csrfParameter" : "x-token",**

 **"authentication" :**

 **[**

 **{**

 **"type" : "basic",**

 **"displayName" : "HTTP basic authentication",**

 **"documentationUrl" : "http://www.example.com/docserv/cmis/basic",**

 **"preference" : 5**

 **},**

 **{**

 **"type" : "certificate",**

 **"displayName" : "SSL certificate authentication for employees",**

 **"documentationUrl" : "http://www.example.com/docserv/cmis/cert",**

 **"preference" : 2**

 **}**

 **]**

 **},**

 **{**

 **"displayName" : "DocServ CMIS 1.1 Browser Binding",**

 **"cmisVersion" : "1.1",**

 **"binding" : "browser",**

 **"url" : "https://host:8080/cmis/browser",**

 **"cookies" : "required",**

 **"compression" : "server",**

 **"csrfHeader" : "X-CSRF-Token",**

 **"csrfParameter" : "x-token",**

 **"authentication" :**

 **[**

 **{**

 **"type" : "basic",**

 **"displayName" : "HTTP basic authentication",**

 **"documentationUrl" : "http://www.example.com/docserv/cmis/basic",**

 **"preference" : 4**

 **},**

 **{**

 **"type" : "certificate",**

 **"displayName" : "SSL certificate authentication for employees",**

 **"documentationUrl" : "http://www.example.com/docserv/cmis/cert",**

 **"preference" : 1**

 **},**

 **{**

 **"type" : "form",**

 **"displayName" : "Form-based authentication for guests",**

 **"loginUrl" : "http://host:8080/login",**

 **"documentationUrl" : "http://www.example.com/docserv/cmis/form",**

 **"preference" : 3**

 **}**

 **]**

 **}**

 **]**

**}**

# Extension Definition

There is no feature extension entry in the repository info because then endpoint discovery takes place before repository info can be obtained.