Authentication with Tokens for Browser Clients

The authentication mechanism described in this section addresses the following scenario:

A web application is hosted on one domain; the CMIS browser binding interface is served from another domain. There is no proxy process on the server that hosts the web application. That is, all communication between the application and the repository has to happen in the web browser via JavaScript. The "same origin policy" enforced by the web browser prohibits a direct and secure two-way communication between the application and the repository.

To access the repository, a user has to authenticate and has to authorize the application (and only this application, not all scripts in the web browser) to make CMIS calls.

JSONP and Form Requests

Cross-domain requests should use JSONP and callbacks (see section 5.2.8) for GET requests and should use HTML forms for POST requests. A token SHOULD be added to each request to prevent cross-site request forgery (CSRF) attacks. For this purpose, a parameter “token” MUST be added to the parameters of a GET request and a control “token” MUST be added to the controls of a HTML form.

The repository SHOULD return a permissionDenied error if the client sends an invalid token.

If the client sends any other form of authentication (basic authentication, OAuth, etc.), the token MAY be omitted. It is RECOMMENDED for web applications always to provide a token, even if another form of authentication is in place.

Login and Tokens

Tokens are obtained from the repository in the following way.

The repository provides a JavaScript script that the web application includes into its HTML page via the HTML <script> tag.

This script provides four functions:

- cmisServiceURL()
  This function returns the Service URL. See section 5.3.1 Service URL.

- cmisLogin(callback)
  This function triggers the login process. The web application MUST call this function before it calls any other functions.
  How the login works is repository specific. A repository MAY replace the application page in the web browser with a login page and later return back to the application page.
The function takes a callback function. It is called when the login process has been completed. The callback function MUST accept a boolean parameter. The repository MUST provide \texttt{true} if the login process was successful and \texttt{false} if it was not successful.

- \texttt{cmisLogout(callback)}
  This function triggers the logout process. How the logout works is repository specific. After a successful logout, \texttt{cmisNextToken()} MUST NOT return a valid token. The application \textbf{MAY call} \texttt{cmisLogin()} again to trigger a new login.
  The function takes a callback function. It is called when the logout process has been completed. The callback function MUST accept a boolean parameter. The repository MUST provide \texttt{true} if the logout process was successful and \texttt{false} if it was not successful.

- \texttt{cmisNextToken(callback)}
  This function calls the provided callback function with a new token. How this token is generated and obtained is repository specific. Whether the repository returns unique tokens or the same token for a user or for all users is repository specific. The repository \textbf{SHOULD} signal an invalid state (e.g. no logged in user) with an empty token (=empty string).

The flow in a web application could looks like this:

```html
<script src="http://cmis.example.com/cmis.js"/>

function cmisLogin(function(success) {
  if (success) {
    displayRootFolder();
  } else {
    showLoginErrorMessage();
  }
});

function displayRootFolder() {
  cmisNextToken(function(token) {
    loadChildren('/', ..., token);
  });
}

...  

</script>
```