CloudMe API

Revision 1.0.4, 2015-11-05

2.5.3 Delete webshare (REST)

2.5.4 Get webshare (REST)

2.5.5 WebShare content (REST)

2.5.6 WebShare authentication (REST)

2.5.7 Following WebShares (REST)

GET-STARTED

To use CloudMe's server api you must be able to send SOAP and REST messages. If you have no previous knowledge of using them you might consider using the Java API. If you still wish to use the raw api, a few helper methods are provided for java. They require the Apache http components library.

```
public class Test {
   final private static String host = "www.cloudme.com";
   final private static String USER_AGENT_INFO = "Nothing special";
   final private static String SOAP_HEADER ="<SOAP-ENV:Envelope xmlns:SOAP-
ENV=\"http://schemas.xmlsoap.org/soap/envelope/\" SOAP-ENV:encodingStyle=\"\" xmlns
:xsi=\"http://www.w3.org/1999/XMLSchema-instance\" xmlns:xsd=\"http://www.w3.org/19
99/XMLSchema\"><SOAP-ENV:Body>";

   final private static String SOAP_FOOTER = "</SOAP-ENV:Body></SOAP-ENV:Envelope>";

   private static AuthScheme scheme = null;
   private static Credentials creds = null;

   private String username = "username";
   private String password = "password";

   public Test() {
      // We call the login call on CloudMe

      String action = "login";
      String body = "";



      String something = callServerStringResponse(action, body);

      System.out.println(something);
   }

   // Creates the connection to the server

   private DefaultHttpClient getConnection() {
      DefaultHttpClient httpClient = new DefaultHttpClient();
      HttpProtocolParams.setUseExpectContinue(httpClient.getParams(), false);
      HttpProtocolParams.setUserAgent(httpClient.getParams(), USER_AGENT_INFO);
      httpClient.getCredentialsProvider().setCredentials(
         new AuthScope(host, 80, "os@xcerion.com", "Digest"),
         new UsernamePasswordCredentials(username, password));    // Encrypts password
& username
```

```
    HttpRequestInterceptor preemptiveAuth = new HttpRequestInterceptor() {
      public void process(final HttpRequest request, final HttpContext context) thr
ows HttpException, IOException {
        AuthState authState = (AuthState) context.getAttribute(ClientContext.TARGET
_AUTH_STATE);

        if ( authState.getAuthScheme() == null ) {
          if ( creds != null && scheme != null ) {
            authState.setAuthScheme(scheme);
            authState.setCredentials(creds);
          } else {

            scheme = authState.getAuthScheme();
          }
        }
      }

    };

    httpClient.addRequestInterceptor(preemptiveAuth, 0);

    return httpClient;
  }

  // Converts the HTTPResponse into a huge string

  private String getResponse(HttpResponse response) {
    try {
      DataInputStream stream = new DataInputStream(response.getEntity().getContent(
));
      StringBuffer buf = new StringBuffer();
      String tmp;
      while ((tmp = stream.readLine()) != null) {
        buf.append(tmp);
      }
      stream.close();
      return buf.toString();
    } catch (IOException e) {
      return "IOException: " + e.getMessage();
    }
  }

  // Does the actuall SOAP-call to the server, returns the response as a string fo
r printing

  private synchronized String callServerStringResponse(String soapAction, String bod
y) {
    DefaultHttpClient httpClient = getConnection();
    byte[] b = null;

    HttpPost httpPost = new HttpPost("http://" + host + "/v1/");
    httpPost.setHeader("soapaction", soapAction);
    httpPost.setHeader("Content-Type", "text/xml; charset=utf-8");

    final StringBuffer soap = new StringBuffer();
    soap.append(SOAP_HEADER);
    soap.append("<" + soapAction + ">");
    soap.append(body);
    soap.append("</" + soapAction + ">");
    soap.append(SOAP_FOOTER);

    System.out.println("\n\n" + soap.toString() + "\n\n");  // Prints the call so
that the SOAP-call can be checked manually for errors


    try {
      HttpEntity entity = new StringEntity(soap.toString());
      httpPost.setEntity(entity);
      HttpResponse response = httpClient.execute(httpPost);

      // Call to getResponse needs to be done before httpClient is shut down

      // otherwise the response disappears and null is returned

      String stringResponse = getResponse(response);

      httpClient.getConnectionManager().shutdown();
      return stringResponse;
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
    httpClient.getConnectionManager().shutdown();
    return null;
  }

    /* All of these REST calls return string for readability, hence no parsing is
done */

  // Used for PUT REST calls

  private synchronized String putServerREST(String putCall, String xmlString, String
type) {
    DefaultHttpClient httpClient = getConnection();

    HttpPut httpPut = new HttpPut(putCall);

    try {
      StringEntity entity = new StringEntity(xmlString, "UTF-8");
      entity.setContentType("application/xml");

      httpPut.setEntity(entity);

      System.out.println("\n\n" + httpPut.getRequestLine().toString() + "\n\n");

      HttpResponse response = httpClient.execute(httpPut);

      // Call to getResponse needs to be done before httpClient is shut down

      // otherwise the response disapears and null is returned

      String stringResponse = getResponse(response);
      httpClient.getConnectionManager().shutdown();
      return stringResponse;
    } catch (Exception e) {
      System.out.println("excep");
      System.out.println(e.getMessage());
    }
    httpClient.getConnectionManager().shutdown();
    return null;
  }

  // Used for DELETE REST calls

  private synchronized String deleteServerREST(String deleteCall) {
    DefaultHttpClient httpClient = getConnection();

    HttpDelete httpDelete = new HttpDelete(deleteCall);

    try {
      System.out.println("\n\n" + httpDelete.getRequestLine().toString() + "\n\n");

      HttpResponse response = httpClient.execute(httpDelete);

      // Call to getResponse needs to be done before httpClient is shut down

      // otherwise the response disapears and null is returned

      String stringResponse = getResponse(response);
      httpClient.getConnectionManager().shutdown();
      return stringResponse;
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
    httpClient.getConnectionManager().shutdown();
    return null;
  }
  // Used for POST REST calls

  // Not complet since e.g. uploading a file would need the body to contain the fi
le, not an xml string

  private synchronized String postServerREST(String postCall, String xmlString) {
    DefaultHttpClient httpClient = getConnection();

    HttpPost httpPost = new HttpPost(postCall);

    try {
    //  StringEntity entity = new StringEntity(xmlString, "UTF-8");

    //  entity.setContentType("text/plain");

      FileBody file = new FileBody(new File("bird.jpg"));
      StringBody comment = new StringBody("A nice bird");
      MultipartEntity reqEntity = new MultipartEntity();
       reqEntity.addPart("bin", file);
       reqEntity.addPart("comment", comment);

       httpPost.setEntity(reqEntity);

      System.out.println("\n\n" + httpPost.getRequestLine().toString() + "\n\n");

      HttpResponse response = httpClient.execute(httpPost);

      // Call to getResponse needs to be done before httpClient is shut down

      // otherwise the response disapears and null is returned

      String stringResponse = getResponse(response);
      httpClient.getConnectionManager().shutdown();
      return stringResponse;
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
    httpClient.getConnectionManager().shutdown();
    return null;
  }

  private synchronized String postServerREST2(String postCall, String xmlString) {
    DefaultHttpClient httpClient = getConnection();

    HttpPost httpPost = new HttpPost(postCall);

    try {
      StringEntity entity = new StringEntity(xmlString, "UTF-8");
      entity.setContentType("text/plain");
    //  FileBody file = new FileBody(new File("bird.jpg"));

     //   StringBody comment = new StringBody("A nice bird");

    //  MultipartEntity reqEntity = new MultipartEntity();

    //   reqEntity.addPart("bin", file);

     //    reqEntity.addPart("comment", comment);


       httpPost.setEntity(entity);

      System.out.println("\n\n" + httpPost.getRequestLine().toString() + "\n\n");

      HttpResponse response = httpClient.execute(httpPost);

      // Call to getResponse needs to be done before httpClient is shut down

      // otherwise the response disapears and null is returned

      String stringResponse = getResponse(response);
      httpClient.getConnectionManager().shutdown();
      return stringResponse;
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
    httpClient.getConnectionManager().shutdown();
    return null;
  }


  // Used for GET REST calls

  private synchronized String getServerREST(String getCall) {
    DefaultHttpClient httpClient = getConnection();

    HttpGet httpGet = new HttpGet(getCall);

    try {
      System.out.println("\n\n" + httpGet.getRequestLine().toString() + "\n\n");

      HttpResponse response = httpClient.execute(httpGet);

      // Call to getResponse needs to be done before httpClient is shut down

      // otherwise the response disapears and null is returned

      String stringResponse = getResponse(response);
      httpClient.getConnectionManager().shutdown();
```

```
            return stringResponse;
        } catch (IOException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        httpClient.getConnectionManager().shutdown();
        return null;
    }
}
```

CloudMe account

login

Summary

login: []
Used to login a user. This is the first operation you have to make as it provides necessary data for further operations. As you might have noticed, login has no arguments, but like all other api calls you need to authenticate yourself.Read the getstarted page to see how.
The most important things provided in the return message are probably:
userid: Used for operations concerning the user, for example to add a listener to the user.
home: The home folder, you need a folder to start from to get information about other folders.
driveId: The id of a drive, used for operations on drives. You will probably get more than one **<drive>** listed on **<drives>** from the login message, however the drive with the name **xios** is the most important as its the drive with the users homefolder.

Example

In this example the user with the username hakelo is logged in. In the return message we can retrieve the users user id,homefolder id and drive id.You can get other information like the firstname and lastname of the user, this information is however not that important for other api calls.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle=""
 xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
 <SOAP-ENV:Body>
  <login>
  </login>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <xcr:loginResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
   <userid>12885514210</userid> // Id of user
   <username>hakelo</username> //Username of user
   <firstname>john</firstname> //Firstname of user
   <lastname>johnson</lastname> //Lastname of user
   <locale>en_US</locale> // Localisation setting for the account
   <lastlogin>2012-10-18T06:25:42Z</lastlogin> // Last time user logged in
   <logins>22</logins> // The total number of times the user has logged in
   <home>562958546675878</home> // Home folder of the user
   <library>562958546675879</library> //Location for the library?
   <in>562958546675880</in>
   <out>562958546675881</out>
   <contacts>17182308671</contacts>
   <settings></settings>
   <params></params>
   <session>660d319261fe66912519b36794e0e7be</session> //The session token for this l
ogin
   <mycloudme>http://my.cloudme.com/</mycloudme>
   <eula/>
   <drives>
    <drive>
     <driveId>30067187875</driveId> //Id of this drive
     <ownerId>12885514210</ownerId> // owner of this drive, as you can see its the sam
e as the userid in this case
     <folderId>562958546675878</folderId> //root folder of drive
     <sysname>xios</sysname>&nbsp; // name of drive in system
     <name></name> // if you have given your drive a name
     <system>home</system> // If the drive has a special purpose in the system
     <currentSize>2171312</currentSize> //Amount of data on drive
     <quotaLimit>3221225472</quotaLimit> //How much you can store at most, this differ
s depending on account type
     <maxFileSize>157286400</maxFileSize> //File size limit
     <readLocked>false</readLocked> //If the drive is locked for reading, true/false
     <writeLocked>false</writeLocked> //If the drive is locked for writing,true/false
     <created>2012-09-18T15:52:11Z</created> // date the drive was created
    </drive>  //There can be several drive
   </drives>
  <group id='17179869184' name='xios'/> //The id and name of the group the user is af
filiated with
 </xcr:loginResponse>
</SOAP-ENV:Body>  </SOAP-ENV:Envelope>
```

Get account information

Summary

Get Account Information:
GET https://www.cloudme.com/v1/users/https://www.cloudme.com/v1/users/"userId"/account
Retrieves account information about a user with id **userId**.
Account information consist of:
balance: The users balance.
type: Free or paid account?
state: State of the account
freeSize: Free space available

Example

In this example we fetch the account information for the user with id **32348765**

Outgoing message

```
GET https://www.cloudme.com/v1/users/"32348765"/account HTTP/1.1
```

Incoming answer

```
<account>
  <balance>0</balance>
  <type>FREE</type>
  <state>NORM</state>
  <freeSize>3221225472</freeSize>
</account>
```

Drives

getDriveInfo

Summary

getDriveInfo: ["driveId"]
Retreives information about a drive with the id **driveId**

Example

In this example we want information about the drive with id **30067187875**

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle=""
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
 <SOAP-ENV:Body>
  <getDriveInfo>
   <drive id ='30067187875'/> // the id if the drive
  </getDriveInfo>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
 <xcr:getDriveInfoResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
  <drive>
   <driveId>30067187875</driveId>
   <ownerId>12885514210</ownerId>
   <folderId>562958546675878</folderId>
   <sysname></sysname>
   <name></name>
   <system>home</system>
   <currentSize>2439768</currentSize>
   <quotaLimit>3221225472</quotaLimit>
   <maxFileSize>157286400</maxFileSize>
   <readLocked>false</readLocked>
   <writeLocked>false</writeLocked>
   <created>2012-09-18T15:52:11Z</created>
  </drive>
 </xcr:getDriveInfoResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Folders

Folders (REST)

Summary

Folders:
GET [https://www.cloudme.com/v1/folders/https://www.cloudme.com/v1/folders/ "folderid"]
Returns the subfolders of the folder with id **folderid**

Example

Returns the subfolders for the folder with id 456445

Outgoing message

```
GET https://www.cloudme.com/v1/folders/456445 HTTP/1.1
```

Incoming answer

```
<fs:folder id='456445' xmlns:xlink='http://www.w3.org/1999/xlink' xmlns:fs='http://x
cerion.com/folders.xsd'>
 <fs:folder id='562958547009387' name='Music'/>
 <fs:folder id='562958547009388' name='CloudMe'/>
 <fs:folder id='562958547009389' name='Pictures' listview='apps/system/listviews/docu
ments_thumbnails.xsl'/>
</fs:folder>
```

deleteFolder

Summary

deleteFolder: ["folderId", "childFolderId"]
Deletes a folder with id **folderId**, or its child if you supply **childFolderId**

Example

In this example we want to delete the folder with id **562958546913094** which lies in the folder with id **562958546675878**. The answer will tell you if the deletion was OK or not.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle=""
 xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">

  <SOAP-ENV:Body>
    <deleteFolder>
      <folder id='562958546675878'/>
      <childFolder id='562958546913094'/> //optional, if not given, the whole paren
t folder will be deleted
    </deleteFolder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:deleteFolderResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">OK</x
cr:deleteFolderResponse> // Ok, Not found, no permission are possible outputs
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

getFolderACL

Summary

getFolderACL: ["folderId", "childFolderId"]
Retrieve the acl(access control list) on a folder with id **folderId** or its child if you supply **childFolderId**

Example

In this example we want to get the acl of the folder with id **562958546886038** that lies in the folder with id **562958546675878**
In the answer we get the acl back, apparently the user with id **12885514210** is allowed to read,delete and change acl but not allowed to write.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle=""
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <getFolderACL><folder id='562958546675878'/> //id of folder
      <childFolder id='562958546886038'/> //(optional) id of child folder
    </getFolderACL>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:getFolderACLResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      <acl tag='Ov9axeQk3q/6DX4tgfN2Q7OHZ58'> //acl tag
        <user id='12885514210' read='1' write='0' delete='1' changeACL='1'/> // acl
properties
      </acl>
    </xcr:getFolderACLResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

getFolderXml

Summary

getFolderXML: ["folderId"]
Returns a tree structure with all subfolders of the folder with id **folderId**

Example

Outgoing message

In this example we want to retrieve the sub folder structure of the folder with id **562958546675878** which is the homefolder of this user.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <getFolderXML>
      <folder id='562958546675878'/> // id of parent folder
    </getFolderXML>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:getFolderXMLResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      <fs:folder id='562958546675878' xmlns:xlink='http://www.w3.org/1999/xlink' xm
lns:fs='http://xcerion.com/folders.xsd'> //parent folder
        <fs:folder id='562958546886038' name='cannotwrite'/>
        <fs:folder id='562958546675882' name='Desktop' system='desktop'>
          <fs:folder id='562958546885065' name='test'/>
        </fs:folder>
        <fs:folder id='562958546675883' name='Trashcan' system='trashcan'/>
        <fs:folder id='562958546675884' name='Documents' system='documents'>
          <fs:folder id='562958546675885' name='Music'/>
          <fs:folder id='562958546675886' name='CloudMe'/>
          <fs:folder id='562958546675887' name='Pictures' listview='apps/system/li
```

```xml
stviews/documents_thumbnails.xsl'>
            <fs:folder id='562958546675888' name='Nature Album' listview='apps/sys
tem/listviews/documents_thumbnails.xsl'/>
          </fs:folder>
        </fs:folder>
        <fs:folder id='562958546885898' name='name'/>
        <fs:folder id='562958546885903' name='testest'/>
        <fs:folder id='562958546886217' name='temp' type='folder'/>
        <fs:folder id='562958546886218' name='Applications' type='folder'>
          <fs:folder id='562958546886219' name='Data' type='folder'/>
        </fs:folder>
        <fs:folder id='562958546886220' name='Profile' type='folder'>
          <fs:folder id='562958546886221' name='Guestbook' type='folder'/>
          <fs:folder id='562958546886222' name='Files' type='folder'/>
          <fs:folder id='562958546886223' name='Photos' type='folder'/>
          <fs:folder id='562958546886224' name='Activity' type='folder' listview='
custom' custom='apps/social/profile/listviews/activities.xsl'/>
          <fs:folder id='562958546886225' name='Visitors' type='folder'/>
        </fs:folder>
      </fs:folder>
    </xcr:getFolderXMLResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

modifyFolder

Summary

modifyFolder: ["folderId", "childFolderId", "name"]
Sets the name of a folder with id **childFolderId** of parent folder with id **folderId** to a new **name**

Example

In this example we set the name of folder with id **562958546886038** inside the folder with id **562958546675878** to **hololo**. The answer will tell us if the operation was OK or not.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <modifyFolder>
      <folder id='562958546675878'/>  //Parent folder id
      <childFolder id='562958546886038'/> //Child folder id
      <name>hololo</name> // the new name
    </modifyFolder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:modifyFolderResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">OK</x
cr:modifyFolderResponse> // Succusfull or not
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

moveFolder

Summary

moveFolder: ["srcFolderId", "srcChildFolderId", "dstFolderId", "name"]
Moves a folder with id **srcChildFolderId** with parent folder with id **srcFolderId** to a new folder with id **dstFolderId** and gives it the new name **name**. If you dont supply a **name**, it will be given a random new one.

Example

In this example we move the folder with id **562958546886038** inside the folder **562958546675878** to the folder with id **562958546885898** and gives it the new name **hololo**. In the answer we can see whether the operation was **OK**or not.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <moveFolder>
      <srcFolder id='562958546675878'/> // parent folder id
      <srcChildFolder id='562958546886038'/>// id of folder to be moves
      <dstFolder id='562958546885898'/> //destination folder
      <name>hololo</name> //optional, if not used a random name will be given
    </moveFolder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:moveFolderResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">OK</xcr
:moveFolderResponse> // successfull or not
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

newFolder

Summary

newFolder: ["folderId", "childFolder", "acl"]
Creates a new folder with the name **childFolder** inside the folder with id **folderId**. You can optionally supply a non default **acl**(access control list) as well.

Example

In this example we create a new folder with the name **123** inside the folder with id **562958546675878**. We also supply a custom acl.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <newFolder>
      <folder id='562958546675878'/> // Parent folder
      <childFolder>123</childFolder> // New folder name
      <acl>
        <user id='12885514210' read='1' write='0' delete='1' changeACL='1'/> //opti
onal acl
      </acl>
    </newFolder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:newFolderResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      <newFolderId>562958546913094</newFolderId> // the new folder id
    </xcr:newFolderResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

setFolderACL

Summary

setFolderACL: ["folderId", "childFolderId", "acl"]
Sets the acl to a new **acl** in a folder with id **folderId** or its child with id **childFolderId**
Old acl entries are destroyed when this method is called so use with care.

Example

In this example we set a new acl to the folder with id **562958546886038** inside its parent folder with id **562958546675878**
In the answer back we get the tag of the new acl.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <setFolderACL>
      <folder id='562958546675878'/> //parent folder
      <childFolder id='562958546886038'/> //optional child folder
      <acl tag='kySmyfKl0RItPS2ktHzOtpNzlGc'>
        <user id='12885514210' read='1' write='1' delete='1' changeACL='1'/> //the
new acl that u want the folder to have
      </acl>
    </setFolderACL>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:setFolderACLResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      <acl tag='ZLmQicYMXCHPwujVnhk4U1bCwaQ'/> //If succesfull returns the new acl
tag
    </xcr:setFolderACLResponse>
  </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

queryFolder

Summary

queryFolder: ["folderId[]"(you can provide more than one), "query", "count", "order","ascending"]
folderId[] The ids of the folders to search or list
query(optional) The query to search for.If you don't use it, all files will be listed. You can read more about the query syntax herehere
count(optional) The maximum amount of results that will be returned, default 100
order(optional) The sort order. Can be one of published, updated, title, length, s1, s2, s3 and s4.Default setting is to sort by relevance according to the search criteria.
ascending(optional True or false, default is true

Can be used to search for files or list all files in given folders.

Example File listing

List all files for the folders with id 562958547009389 and 562958547009387

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
 <SOAP-ENV:Body>
  <queryFolder>
   <folder id='562958547009389'/>
   <folder id='562958547009387'/>
  </queryFolder>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<?xml version="1.0" encoding="utf-8"?><SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <xcr:queryFolderResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
  <atom:feed xmlns:atom='http://www.w3.org/2005/Atom' xmlns:os='http://a9.com/-/spec/
```

```
opensearch/1.1/' xmlns:dc='http://xcerion.com/directory.xsd' xmlns:ni='http://xcerio
n.com/noindex.xsd'>
 <os:totalResults>2</os:totalResults>
 <os:startIndex>0</os:startIndex>
 <os:itemsPerPage>100</os:itemsPerPage>
 <atom:entry>
  <atom:title>Halmstad.jpg</atom:title>
  <atom:published>2012-10-11T12:39:11Z</atom:published>
  <atom:updated>2012-10-24T09:04:54Z</atom:updated>
  <atom:link rel='alternate' type='image/jpeg' href='http://os.cloudme.com/v1/docume
nts/562958547009389/4398895499/1' length='449732'/>
  <atom:id>mid:10631d18b@xios.xcerion.com</atom:id>
  <dc:folder>562958547009389</dc:folder>
  <dc:document>4398895499</dc:document>
  <dc:width>1280</dc:width>
  <dc:height>960</dc:height>
  <dc:thumbnail>32769</dc:thumbnail>
  <dc:date>2007-08-04T11:02:15</dc:date>
  <dc:orientation>top-left</dc:orientation>
  <dc:camera-make>CASIO COMPUTER CO.,LTD</dc:camera-make>
  <dc:camera-model>EX-Z3</dc:camera-model>
  <dc:flash>No, compulsory</dc:flash>
  <atom:rights>Copyright Daniel Arthursson 2008, All rights reserved. May only be us
ed as demo picture on Xcerions iCloud.</atom:rights>
  <dc:dpi>72x72</dc:dpi>
  <dc:focal>5.8 mm</dc:focal>
  <dc:aperture>F4.3</dc:aperture>
  <dc:exposure>1/400 s</dc:exposure>
 </atom:entry>
 <atom:entry>
  <atom:title>Berg.jpg</atom:title>
  <atom:published>2012-10-11T12:39:11Z</atom:published>
  <atom:updated>2012-10-24T09:04:54Z</atom:updated>
  <atom:link rel='alternate' type='image/jpeg' href='http://os.cloudme.com/v1/docume
nts/562958547009389/4398895498/1' length='489431'/>
  <atom:id>mid:10631d18a@xios.xcerion.com</atom:id>
  <dc:folder>562958547009389</dc:folder>
  <dc:document>4398895498</dc:document>
  <dc:width>1280</dc:width>
  <dc:height>720</dc:height>
  <dc:thumbnail>32769</dc:thumbnail>
  <dc:date>2008-05-10T21:37:18</dc:date>
  <dc:orientation>top-left</dc:orientation>
  <dc:camera-make>SONY</dc:camera-make>
  <dc:camera-model>HDR-CX7EK</dc:camera-model>
  <dc:flash>No, compulsory</dc:flash>
  <atom:rights>Copyright Daniel Arthursson 2008, All rights reserved. May only be us
```

```
ed as demo picture on Xcerions iCloud.</atom:rights>
   <dc:dpi>72x72</dc:dpi>
   <dc:focal>5.4 mm</dc:focal>
   <dc:aperture>F4.8</dc:aperture>
   <dc:exposure>1/150 s</dc:exposure>
  </atom:entry>
 </atom:feed>
</xcr:queryFolderResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example Query search

Searches the folder with id 562958546675878 with the query **'test.xml' 'dog'**

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
 <SOAP-ENV:Body>
  <queryFolder>
   <folder id='562958546675878'/>
   <query>'test.xml' 'dog' </query>
   <count>5</count>
   <order>title</order>
   <ascending>false</ascending>
  </queryFolder>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<?xml version="1.0" encoding="utf-8"?><SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <xcr:queryFolderResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
   <atom:feed xmlns:atom='http://www.w3.org/2005/Atom' xmlns:os='http://a9.com/-/spec
/opensearch/1.1/' xmlns:dc='http://xcerion.com/directory.xsd' xmlns:ni='http://xceri
on.com/noindex.xsd'>
    <os:totalResults>2</os:totalResults>
    <os:startIndex>0</os:startIndex>
    <os:itemsPerPage>5</os:itemsPerPage>
    <atom:entry>
     <atom:title>test.xml</atom:title>
    <atom:published>2012-10-18T12:29:01Z</atom:published>
    <atom:updated>2013-03-06T21:09:17Z</atom:updated>
    <atom:link rel='alternate' type='text/xml' href='http://os.cloudme.com/v1/documen
ts/562958546675878/4405949618/1' length='553'/>
    <atom:id>mid:1069d74b2@xios.xcerion.com</atom:id>
    <dc:folder>562958546675878</dc:folder>
    <dc:document>4405949618</dc:document>
    <dc:root>data</dc:root>
   </atom:entry>
   <atom:entry>
    <atom:title>dog</atom:title>
    <atom:published>2012-11-07T08:32:16Z</atom:published>
    <atom:updated>2012-11-10T10:46:04Z</atom:updated>
    <atom:link rel='alternate' type='image/jpeg' href='http://os.cloudme.com/v1/docum
ents/562958546675878/4399505706/1' length='0'/>
    <atom:id>mid:1063b212a@xios.xcerion.com</atom:id>
    <dc:folder>562958546675878</dc:folder>
    <dc:document>4399505706</dc:document>
   </atom:entry>
  </atom:feed>
 </xcr:queryFolderResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Query syntax

Terms

A term is a string you want to search for, or sometimes exclude from the search result. For the sake of simplicity you should always use **'** or **"** around the terms.
You can put a **+** before a term force it to be included in the search query.
You can put a **-** before a term to force it to excluded in the search query.

Examples

```
"cat" "dog" "horse"
```

This would search for files that include **(cat OR dog OR horse)**

```
+"cat" +"dog" "horse"
```

This would search for files that include **(cat AND dog) AND MAYBE (horse)**

```
-"cat" +"dog" "horse"
```

This would search for files that include **(dog) AND NOT (cat) AND MAYBE (horse)**

Prefixed Terms

It is possible to add prefixes on terms to search certain metadata.It should be noted that prefixed terms have an AND relation to terms with another prefix and OR relation to terms with the same prefix.
The syntax for Prefixed terms looks like this: **prefix**:**term**.

Examples

```
-title:"animated" +title:"dog" type:"image/gif"
```

This would search for all gif files that contain dog but doesn't contain animated in the title, **(title:dog AND NOT title:animated) AND (type:image/gif)**

```
title:"animated" title:"dog" -type:"image/gif"
```

This would search for all files that contain animated or dog in the title, except for the gif files, **(title:animated OR title:dog) AND NOT (type:image/gif)**

List of prefixes

Query prefix	Atom entry field	Description

author	atom:entry	The author's name, email or URI.
country	dc:country	A two-letter ISO country code (same as used in domain names).
(no prefix)	atom:content	Full-text index
document	dc:document	The document ID as a decimal number.
	dc:enddate	This element is not indexed as-is. See below.
flag	dc:flag	A general-purpose flag that can be used by applications.
folder	dc:folder	The ID of the folder the document is located in (a decimal number).
	dc:atom:id	This element is not indexed.
keyword	dc:keyword	Keywords that describe the document.
lang	dc:lang	The language of the document as a two-character ISO code.
month		This prefix is calculated from the `startdate` and `enddate` elements.
namespace	dc:namespace	The document's XML namespace.
	atom:published	This element is not indexed as-is. See below.
pubmonth		This prefix is calculated from the `atom:published` element.
pubweek		This prefix is calculated from the `atom:published` element.
pubyear		This prefix is calculated from the `atom:published` element.
recipient	dc:recipient	The recipient's name, email or URI.
rights	atom:rights	A copyright string.
s1	dc:s1, ni:s1	A general purpose prefix that is also sortable
s1	dc:s2, ni:s2	A general purpose prefix that is also sortable
s2	dc:s3, ni:s3	A general purpose prefix that is also sortable
s3	dc:s4, ni:s4	A general purpose prefix that is also sortable
schema	atom:category type="schema" scheme="..."	The document's W3C schema.
	dc:startdate	This element is not indexed as-is. See below.
summary	atom:summary	A summary of the document's content.

`tag`	`dc:tag`	A user-specified tag (like "family", "London", "My wedding" etc).
`title`	`atom:title`	A documents title or rather simply the file name.
`type`	`atom:link rel="alternate" type="..."`	A documents MIME type.
`week`		This prefix is calculated from the `startdate` and `enddate` elements.
`year`		This prefix is calculated from the `startdate` and `enddate` elements.

Documents

Download file

Summary

Download file:
GET https://www.cloudme.com/v1/documents/https://www.cloudme.com/v1/documents/_"folder_id"/"document_id"/1

Its also possible to use the optional parameter **dl=filename** to set a name for the downloaded file e.g https://www.cloudme.com/v1/documents/562958548450050/4400924056/1/Halmstad.jpg?dl=Halmstad.jpg

Example

Downloads a text file with id **786876**, that lies in the folder with id **456546**

Outgoing message

```
GET https://www.cloudme.com/v1/documents/456546/786876/1 HTTP/1.1
```

Incoming answer

```
This is a textfile,sdfsdfs
```

Upload a file

Summary

Upload File:
POST https://www.cloudme.com/v1/documents/https://www.cloudme.com/v1/documents/ "folderId"
Uploads a file to the folder with id **folderId**
You need to attach the file to the message body. For example in java with the apache http client you can attach a as follow

```
FileBody file = new FileBody(new File("bird.jpg"));
   StringBody comment = new StringBody("A nice bird");
   MultipartEntity reqEntity = new MultipartEntity();
    reqEntity.addPart("bin", file);
    reqEntity.addPart("comment", comment);

    httpPost.setEntity(reqEntity);
```

Example

In this example we upload a file to the folder with id **45458454**.
In the reply from the server you get useful information about the file you uploaded, like its document id.

Outgoing message

```
POST https://www.cloudme.com/v1/documents/45458454 HTTP/1.1
```

Incoming answer

```
<atom:entry xmlns:atom='http://www.w3.org/2005/Atom' xmlns:os='http://a9.com/-/spec/
opensearch/1.1/' xmlns:dc='http://xcerion.com/directory.xsd' xmlns:ni='http://xcerio
n.com/noindex.xsd'>
   <atom:title>bird.jpg</atom:title> // title of the file
   <atom:published>2012-10-21T09:15:10Z</atom:published>
   <atom:updated>2012-10-21T09:15:10Z</atom:updated>
   <atom:link rel='alternate' type='image/jpeg' href='https://www.cloudme.com/v1/docu
ments/562958546675878/4398763170/1' length='63057'/> // direct link to file
   <atom:id>mid:1062fcca2@xios.xcerion.com</atom:id>
   <dc:folder>562958546675878</dc:folder> // folder where file exists
   <dc:document>4398763170</dc:document> //document/file id
</atom:entry>
```

copyDocument

Summary

copyDocument: ["srcFolderId", "srcDocumentId", "srcDocument", "dstFolderId", "dstDocument"]
Used to copy a file.Copies a document with id **srcDocumentId** or name **srcDocument** in the folder with id **srcFolderId** to another folder with id **dstFolderId** and gives it the new name **dstDocument**.

Example

In this example we copy a file with the name Sunset.jpg in a folder with id 562958546675878 to the same folder and give the copy the name sunset_copy.jpg. The answer contains various metadata, the most important are probably **<dc:document>** as its the document id of the new copy of the file.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <copyDocument>
      <srcFolder id='562958546675878'/> // id of the folder the file exist in
      <srcDocument>Sunset.jpg</srcDocument> // you can use document id aswell i.e <
srcDocument id='36546465654'/>
      <dstFolder id='562958546675878'/> // destination folder
      <dstDocument>sunset_copy.jpg</dstDocument> // name of the copy
    </copyDocument>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:copyDocumentResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      <atom:entry xmlns:atom='http://www.w3.org/2005/Atom' xmlns:os='http://a9.com/
-/spec/opensearch/1.1/' xmlns:dc='http://xcerion.com/directory.xsd' xmlns:ni='http:/
/xcerion.com/noindex.xsd'>
```

```
          <atom:title>sunset_copy.jpg</atom:title> // name of the file
          <atom:published>2012-10-17T16:29:25Z</atom:published> // date of creation
          <atom:updated>2012-10-19T07:52:53Z</atom:updated> // date of last update
          <atom:link rel='alternate' type='image/jpeg' href='https://www.cloudme.com/
v1/documents/562958546675878/4398684297/1' length='318203'/> // direct link to file
          <atom:id>mid:1062e9889@xios.xcerion.com</atom:id>
          <dc:folder>562958546675878</dc:folder> // folder the file is located in
          <dc:document>4398684297</dc:document> //document id
          <dc:width>800</dc:width> // width of jpg picture
          <dc:height>600</dc:height> //height of jpg picture
          <dc:thumbnail>32769</dc:thumbnail> //
          <dc:dpi>96x96</dc:dpi> //dpi of picture
          <dc:businesscompanyname xmlns:dc="http://xcerion.com/directory.xsd"/>
          <atom:author xmlns:atom="http://www.w3.org/2005/Atom">
            <atom:name/>
          </atom:author>
        </atom:entry>
      </xcr:copyDocumentResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

createDocument

Summary

createDocument: ["folderId", "acl", "overwrite", "entry"]
Creates a document in a folder with the id **folderId**.The entry specifies the properties of document.The most basic entry would only specify the name, like in the example below. **overwrite** determines if you should overwrite if a document with the same already exists.The default value for overwrite is false. **acl** allows you to give the document a specified access control list, as shown in the example below.acl is optional.

Example

In this example we create a new document in the folder with id 562958546675878. We give it a custom acl and set overwrite to true.We also provide a simple entry with only the title of the new document. In the answer we get an entry which contains information about the create document, the most important being **dc:document** which is the document id.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <createDocument>
      <folder id='562958546675878'/> //folder id where you want to place the file
      <acl>
        <user id='12885514210' read='1' write='1' delete='1' changeACL='1'/> // opt
ional acl
      </acl>
      <overwrite>true</overwrite> //optional overwrite
      <atom:entry xmlns:atom='http://www.w3.org/2005/Atom'>
        <atom:title>my_new_text_document </atom:title> // name of the file
      </atom:entry>
    </createDocument>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
```

```
    <xcr:createDocumentResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      <atom:entry xmlns:atom='http://www.w3.org/2005/Atom' xmlns:os='http://a9.com/
-/spec/opensearch/1.1/' xmlns:dc='http://xcerion.com/directory.xsd' xmlns:ni='http:/
/xcerion.com/noindex.xsd'>
        <atom:title>my_new_text_document </atom:title> // the name of the file
        <atom:published>2012-10-19T07:24:34Z</atom:published> // when it was create
d
        <atom:updated>2012-10-19T07:24:35Z</atom:updated> // time of last update
        <atom:link rel='alternate' type='text/plain' href='https://www.cloudme.com/
v1/documents/562958546675878/4398682309/1' length='0'/> // direct link to the file
        <atom:id>mid:1062e90c5@xios.xcerion.com</atom:id>
        <dc:folder>562958546675878</dc:folder> // id of the folder the document exi
st in
        <dc:document>4398682309</dc:document> //document id
      </atom:entry>
    </xcr:createDocumentResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

deleteDocument

Summary

deleteDocument: ["folderId", "documentId", "document"]
Deletes a document with id **documentId** or name **document** in the folder with id **folderId**.The answer from the server will only respond with OK if the document were deleted as intended.

Example

In the example we delete the file hello.txt in the folder 562958546675878. In the answer from the server we can see that the operation was OK.

Outgoing output

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <deleteDocument>
      <folder id='562958546675878'/> // the id of the folder the file exist in
      <document>hello.txt</document> // name of the file to be deleted, you can use
document id aswell i.e <document id='36546465654'
    </deleteDocument>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:deleteDocumentResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      OK //successfull or not
    </xcr:deleteDocumentResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

getDocumentACL

Summary

getDocumentACL: ["folderId", "documentId", "document"]
Gets the access control list of a document with id **documentId** or name **document** in the folder with id **folderId**.
In the answer from the server you get the access control list with its defining tag.

Example

In this example we want to get the acl of the file Sunset.jpg in the folder with id 562958546675878.
From the server we get the answer which contains an acl with the tag BfiU/Vkz980Y8xsetQYAYFMvR5s.
Apparently the user with id 12885514210 have full control over the file.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <getDocumentACL>
      <folder id='562958546675878'/> //id of the folder where the file exist
      <document>Sunset.jpg</document> // you can use document id aswell i.e <documen
t id='36546465654'/>
    </getDocumentACL>
  </SOAP-ENV:Body>
/SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
     <xcr:getDocumentACLResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
       <acl tag='BfiU/Vkz980Y8xsetQYAYFMvR5s'>
         <user id='12885514210' read='1' write='1' delete='1' changeACL='1'/> // acl
of the file
       </acl>
     </xcr:getDocumentACLResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

loadMetadata

Summary

loadMetadata: ["folderId", "documentId", "document"]
Retrieves metadata(describing information) about a document with the id **documentId** or name **document** in the folder with id **folderId**
Metadata between different files and file types will differ, a text file wont have the attribute height like a jpg would for example.

Example

In this example we want to load the metadata of the file bird.jpg in the folder with id 562958546675878.
In the answer we get an entry with the files metadata.

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <loadMetadata>
      <folder id='562958546675878'/> // folder where the file exist
      <document>bird.jpg</document> // you can use document aswell i.e <document id='4654646'/>
    </loadMetadata>
  </SOAP-ENV:Body>
/SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
```

```
    <xcr:loadMetadataResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      <atom:entry xmlns:atom='http://www.w3.org/2005/Atom' xmlns:os='http://a9.com/
-/spec/opensearch/1.1/' xmlns:dc='http://xcerion.com/directory.xsd' xmlns:ni='http:/
/xcerion.com/noindex.xsd'>
         <atom:title>bird.jpg</atom:title>
         <atom:published>2012-10-21T09:15:10Z</atom:published>
         <atom:updated>2012-10-23T07:49:34Z</atom:updated>
         <atom:link rel='alternate' type='image/jpeg' href='https://www.cloudme.com/
v1/documents/562958546675878/4398763170/1' length='262341'/>
         <atom:id>mid:1062fcca2@xios.xcerion.com</atom:id>
         <dc:folder>562958546675878</dc:folder>
         <dc:document>4398763170</dc:document>
         <dc:width xmlns:dc="http://xcerion.com/directory.xsd">600</dc:width>
         <dc:height xmlns:dc="http://xcerion.com/directory.xsd">500</dc:height>
         <dc:thumbnail xmlns:dc="http://xcerion.com/directory.xsd">32769</dc:thumbna
il>
         <atom:summary>Ducky&lt;</atom:summary>
         <dc:dpi xmlns:dc="http://xcerion.com/directory.xsd">100x100</dc:dpi>
         <dc:businesscompanyname xmlns:dc="http://xcerion.com/directory.xsd"/>
         <atom:author xmlns:atom="http://www.w3.org/2005/Atom">
           <atom:name/>
         </atom:author>
      </atom:entry>
    </xcr:loadMetadataResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

moveDocument

Summary

moveDocument: ["srcFolderId", "srcDocumentId", "srcDocument", "dstFolderId", "dstDocument"]
Moves a document with id **srcDocumentId** or name **srcDocument** in the folder with id **srcFolderId** to another folder with id **dstFolderId** and gives it the new name **dstDocument**.

Example

In this example we move the document **Red.jpg** that resides in the folder with id **562958546675878** to the same folder and give it the new name **moved_file.jpg**

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <moveDocument>
      <srcFolder id='562958546675878'/> //folder where file exist
      <srcDocument>RED.jpg</srcDocument> // name of file,you can use document id as
well i.e <document id='36546465654'/>
      <dstFolder id='562958546675878'/> //the folder where you want to put the file
      <dstDocument>moved_file.jpg</dstDocument> //Name of the file at its new locat
ion
    </moveDocument>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:moveDocumentResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      <atom:entry xmlns:atom='http://www.w3.org/2005/Atom' xmlns:os='http://a9.com/
-/spec/opensearch/1.1/' xmlns:dc='http://xcerion.com/directory.xsd' xmlns:ni='http:/
/xcerion.com/noindex.xsd'>
```

```
         <atom:title>moved_file.jpg</atom:title>
         <atom:published>2012-10-17T16:29:25Z</atom:published>
         <atom:updated>2012-10-17T16:29:52Z</atom:updated>
         <atom:link rel='alternate' type='image/jpeg' href='https://www.cloudme.com/
v1/documents/562958546675878/4398620818/1' length='318203'/> // direct link to file
         <atom:id>mid:1062da092@xios.xcerion.com</atom:id>
         <dc:folder>562958546675878</dc:folder>
         <dc:document>4398620818</dc:document>
         <dc:width>800</dc:width> //this information will differ depening on filetyp
e
         <dc:height>600</dc:height>
         <dc:thumbnail>32769</dc:thumbnail>
         <dc:dpi>96x96</dc:dpi>
         <dc:businesscompanyname xmlns:dc="http://xcerion.com/directory.xsd"/>
         <atom:author xmlns:atom="http://www.w3.org/2005/Atom">
           <atom:name/>
         </atom:author>
       </atom:entry>
     </xcr:moveDocumentResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

renameDocument

Summary

renameDocument: ["folderId", "documentId", "document","newName"]
Renames a document with id **documentId** or name **document** in the folder with id **folderId** to the new name **newName**

Example

In this example we rename to file **Sunset.jpg in the folder with id *562958546675878** to the new name **RED.jpg**

Outgoing message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <renameDocument>
      <folder id='562958546675878'/> // folder id where file is located
      <document>Sunset.jpg</document> //you can use document id aswell i.e <documen
t id='36546465654'/>
      <newName>RED.jpg</newName> //the new name of the file
    </renameDocument>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <xcr:renameDocumentResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
      <atom:entry xmlns:atom='http://www.w3.org/2005/Atom' xmlns:os='http://a9.com/
-/spec/opensearch/1.1/' xmlns:dc='http://xcerion.com/directory.xsd' xmlns:ni='http:/
/xcerion.com/noindex.xsd'>
        <atom:title>RED.jpg</atom:title>
        <atom:published>2012-10-17T16:29:25Z</atom:published>
        <atom:updated>2012-10-17T16:29:52Z</atom:updated>
        <atom:link rel='alternate' type='image/jpeg' href='https://www.cloudme.com/
v1/documents/562958546675878/4398620818/1' length='318203'/> //direct link to file
        <atom:id>mid:1062da092@xios.xcerion.com</atom:id>
        <dc:folder>562958546675878</dc:folder>
        <dc:document>4398620818</dc:document>
        <dc:width>800</dc:width>
        <dc:height>600</dc:height>
        <dc:thumbnail>32769</dc:thumbnail>
        <dc:dpi>96x96</dc:dpi>
        <dc:businesscompanyname xmlns:dc="http://xcerion.com/directory.xsd"/>
        <atom:author xmlns:atom="http://www.w3.org/2005/Atom">
          <atom:name/>
        </atom:author>
      </atom:entry>
    </xcr:renameDocumentResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

setDocumentACL

Summary

setDocumentACL: ["folderId", "documentId", "acl", "document"]
Sets the ACL of a document with id **documentId** or name **document** in the folder with id **folderId** to the ACL **acl**
This call should be used with care, as old ACL entries will be removed when you set a new one.

Example

In this example we set the acl of the document **sunset.jpg** to a new one, which gives the user with id **12885514210** the right to write,delete and change acl of the file, but disallows reading of the file

Outgoing message

```
 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns
:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
     <setDocumentACL>
       <folder id='562958546675878'/> //folder where file exist
       <acl tag='BfiU/Vkz980Y8xsetQYAYFMvR5s'>
         <user id='12885514210' read='0' write='1' delete='1' changeACL='1'/> // the
new acl
       </acl>
       <document>sunset.jpg</document> //name of the file,you can use document id as
well i.e <document id='36546465654'/>
     </setDocumentACL>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Incoming answer

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
     <xcr:setDocumentACLResponse xmlns:xcr="http://xcerion.com/xcRepository.xsd">
       <acl tag='KDPzIsz0II/uaUn5eyTc4DonsaU'/> // tag of the new acl
     </xcr:setDocumentACLResponse>
```

```
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

WebShares

Create webshare

Summary

Create Webshare:
POST https://www.cloudme.com/v1/users/https://www.cloudme.com/v1/users/ "userId"/webshares
Creates a new webshare for the user with id **userId**

Example

We create a new webshare for the user with id **15151241**

Outgoing message

```
POST https://www.cloudme.com/v1/users/15151241/webshares HTTP/1.1
```

And attach the webshare message in the body, an example is

```
<webshare name='webshareName' password='password' description='description'><folder
id='562958546675878' name='hej'/></webshare>"
```

Incoming answer

```
<webshare id='646307' userId='12885514210' name='webshareName' description='descript
ion' visibility='public' password='password' type='' created='2012-10-22T10:08:34Z'
updated='2012-10-22T10:08:34Z'><folder name='hej' id='562958546675878'/></webshare>
```

Summary

Change Webshare:
POST https://www.cloudme.com/v1/users/https://www.cloudme.com/v1/users/"userId"/webshares/"webshareId"
Changes a webshare with the id **webshareId** owned by the user with id **userId**

Example

In this example we change the webshare with id **648548** owned by the user with id **12885514210**

Outgoing message

```
POST https://www.cloudme.com/v1/users/12885514210/webshares/648548 id" HTTP/1.1
 //where userid is the owner of the share and webshare id is the id of the webshare
```

You attach the changes u want to make to the message body, for example

```
<webshare name='mina filer' password='rmf' description='my files' visibilty='public'
><folder id='562958546675878' name='hej'/></webshare>
```

Delete webshare

Summary

Delete Webshare:
DELETE https://www.cloudme.com/v1/users/https://www.cloudme.com/v1/users/"userId"/webshares/"webshareId"
Deletes a webshare with the id **webshareId** owned by user with the id **userId**

Example

In this example we deleted the webshare with id **45456456** owned by the user with id **45161616**

Outgoing message

```
DELETE https://www.cloudme.com/v1/users/45161616/webshares/45456456 id" HTTP/1.1
```

Get webshare

Summary

Get Webshares: GET http://os.cloudme.com/v1/users/http://os.cloudme.com/v1/users/ "userId"/webshares?

List the webshares from the user with id **usedId**

Example

In this example we retrieve the webshares from the user with id **2734564228**

Outgoing message

```
GET http://os.cloudme.com/v1/users/2734564228/webshares? HTTP/1.1
```

Paging

- offset: >= 0 (Default: 0)

count: 0 <= 1000 (Default: 100)

resources: true/false (includes all resources for webshare) (Default: false)

Example:
/v1/users/12884901890/webshares/?offset=0&count=100

Sorting

- order: created | name | visibility | updated (Default: created)

desc: true | false (Default: false)

Example:
/v1/users/12884901890/webshares/?order=name&desc=true

Filters

- name: String

visibility: String (public | friends | private)

Example:
/v1/users/12884901890/webshares/?name=moonshine

Incoming answer

```
<webshares xmlns:os='http://a9.com/-/spec/opensearch/1.1/'>

   <os:totalResults>1</os:totalResults> // number of webshares found
   <os:startIndex>0</os:startIndex>
   <os:itemsPerPage>100</os:itemsPerPage>
   <webshare id='646450' userId='12885514210' name='mina filer' description='my files' vis
ibility='public' password='rmf' type='' created='2012-10-24T08:09:15Z' updated='2012-10-
24T08:09:15Z'></webshare>
</webshares>
```

WebShare content

/v1/webshares/{user_id}/{webshare_name}

user_id is the numeric user-ID. *webshare_name* is the name of the webshare.

GET

Returns an Atom feed with entries for each folder and document in the webshare. This resource (and all decendants) may require a password to access.

Pagination is available using two query parameters: *offset* (which defaults to 0) and *count* (which defaults to 100 and may not exceed 1000).

POST

Creates a folder in the specified folder with folderId.
This is used for inbound sharing.

Request

A request should look like this.

```
<newFolder folderId=\"{folder_id}\" childName=\"{NameOfNewFolder}\" />
```

Response

```
<newFolderId>1234</newFolderId>
```

/v1/webshares/{user_id}/{webshare_name}/{path}

user_id is the numeric user-ID. *webshare_name* is the name of the webshare. *path* is a slash-separated sequence of folder names, possibly ending in a document name.

GET

Returns either an Atom feed with entries for each folder and document in the folder specified by *path*, or the contents of the document if *path* points to such a resource.

Pagination is available for folder listings using two query parameters: *offset* (which defaults to 0) and *count* (which defaults to 100 and may not exceed 1000).

For document resources, the *s* query parameter may be used to specify what document stream to return. Default is 1.

WebShare authentication

/v1/wsauth/{user_id|:username}/{webshare_id|:webshare_name}

POST

Verify the password for a WebShare. Requires the username and password properly sent in digest. Username is not verified (can be any).

- The username should be prepended with a colon (':') to distinguish between user-id and usernames
  The webshare-name should be prepended with a colon (':') to distinguish between webshare-id and webshare-names

Body

<auth/>

Response

- 204 No Content if successful
  401 Unathorized if no digest was submitted

  403 Forbidden if the password is wrong

Following WebShares

A favorite (webshare favorite) is a webshare added to the user account.

Structure of a (webShare) favorite:

- id
  userId
  sharingUserId
  webShareId
  created
  name
  password

REST resources

/v1/users/{user_id}/favorites/

user_id is the numeric user-ID.

GET

Returns a paginated list of favorites

Response Options

- extended: true/false (Default: false) (Includes extra information including inbound option and (if items == 1) root item)

Paging

- offset: >= 0 (Default: 0)
  count: 0 <= 1000 (Default: 1000)

  Example:
  /v1/users/12884901890/favorites/?offset=0&count=100

Sorting

- order: created | favoritename | sharinguserid | sharingusername (Default: created)
  desc: true | false (Default: false)

  Example:
  /v1/users/12884901890/favorites/?order=favoritename&desc=true

Filters

- webshareid: Integer
  favoritename: String

  sharinguserid: Integer

sharingusername: String

Example:
/v1/users/12884901890/favorites/?webshareid=4&extended=true

Response

/v1/users/12884901890/favorites/?desc=true&order=favoritename&offset=0&count=20

```
<?xml version="1.0"?>
<favorites xmlns:os="http://a9.com/-/spec/opensearch/1.1/">
 <os:totalResults>2</os:totalResults>
 <os:startIndex>0</os:startIndex>
 <os:itemsPerPage>20</os:itemsPerPage>
 <favorite id="1" userId="12884901890" sharingUserId="12884901892" sharingUserName="test3
" webShareId="4" created="2012-01-16T19:29:25Z" name="sunshine" password="" acces
s="update" folder_id="1234" document_id="0"/>
 <favorite id="3" userId="12884901890" sharingUserId="12884901889" sharingUserName="test"
webShareId="1" created="2012-01-19T19:18:09Z" name="moonshine" password="" folder_i
d="1234" document_id="4567"/>
 <favorite id="4" userId="12884901890" sharingUserId="12884901889" sharingUserName="test2
" webShareId="2" created="2012-01-19T19:18:09Z" name="moonshine" password=""/>
</favorites>
```

POST

Create a favorite.

Postdata

Requires (sharingUserId or sharingUsername) and (webShareId or webShareName)

Examples:

```
<favorite name="Vacation 2012" sharingUserId="12884901892" webShareId="4" password="test"
/>
<favorite name="Vacation 2012" sharingUsername="test4" webShareName="Vacation 2012" passw
ord="test"/>
```

Response

```
<favorite id="1" userId="12884901890" sharingUserId="12884901892" sharingUserName="test3"
webShareId="4" created="2012-01-16T19:29:25Z" name="Vacation 2012" password="test"/>
```

Errors

Error	**Cause**
You must provide either 'sharingUsername' or 'sharingUserId'	*Missing sharingUsername or sharingUserId in postdata*
You must provide either 'webShareId' or	*Missing webShareName or webShareId in postdata*

'webShareName'	
Wrong password	*The password provided for the webshare is incorrect*
The WebShare is already added as a favorite	*The webshare has been added previously to this user's favorites*

/v1/users/{user_id}/favorites/{favorite_id}

user_id is the numeric user-ID.
favorite_id is the numeric user-ID.

GET

Returns one favorite.

Request

/v1/users/12884901890/favorites/13

Response

```
<?xml version="1.0"?>
<favorite id="13" userId="12884901890" sharingUserId="12884901889" sharingUserName="test"
webShareId="13" created="2012-02-13T15:13:06Z" name="Birds" password="bird">
 <webshare id="13" userId="12884901889" name="Birds" description="Bird pictures" visibili
ty="public" password="bird" type="" created="2012-02-13T15:08:16Z" updated="2012-02-
13T15:08:16Z">
  <folder name="Pictures" id="8589934600"/>
 </webshare>
</favorite>
```

If the password in the favorite does no longer match with the password in the webshare, the response will look as follows:

```
<?xml version="1.0"?>
<favorite id="13" userId="12884901890" sharingUserId="12884901889" sharingUserName="test"
webShareId="13" created="2012-02-13T15:13:06Z" name="Birds" password="bird">
 <error cause="Wrong password"/>
</favorite>
```

PUT

Update a favorite.

Request

/v1/users/12884901890/favorites/3

Putdata

@name *or* @password is required.

```
<favorite name="sunshine2"/>  <!-- Updates only name -->
<favorite password="testing"/>  <!-- Updates only password -->
<favorite name="sunshine2" password="testing"/>  <!-- Updates both -->
```

Response

No content

DELETE

Delete a favorite.

Request

/v1/users/12884901890/favorites/3

Response

No content

POST

Operate on multiple webshare favorite objects. Currently only copy item to own user-folder is supported.

Postdata

```
<copy destinationFolder="4747">
  <documents>
    <document document="1919" folder="2020" />
    <document />
    <document />
  </documents>
  <folders>
    <folder childFolder="3030" />
    <folder />
  </folders>
</copy>
```

Response

No content

Errors

Error	Cause

/v1/users/{user_id}/favorites/{favorite_id}/{folder_id}

user_id is the numeric user-ID.
favorite_id is the numeric favorite-ID.
folder_id is the numeric id of the folder which the file should be uploaded to.

Use header to get the correct href tag, the header should look like this:
relative_path = folder/in/favorite[/]
NOTE: *relative_path* header is ONLY to get the correct favorite href tag, has nothing else with the upload to do.

POST

Uploads a document to the folder with *folder_id* through a favorite bound to a inbound webshare.
Returns an atom entry with information about the uploaded document,
This is used for inbound sharing through a favorite.

```
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:os="http://a9.com/-/spec/opens
earch/1.1/" xmlns:dc="http://xcerion.com/directory.xsd" xmlns:ni="http://xcerion.com/noin
dex.xsd">
   <atom:title>inboundtest3.jpg</atom:title>
   <atom:published>2012-12-06T12:20:13Z</atom:published>
   <atom:updated>2012-12-06T12:20:16Z</atom:updated>
   <atom:link rel="enclosure" type="image/jpeg" href="http://localhost/v1/users/1288490189
0/favorites/1/webshare/folder/in/favorite/inboundtest3.jpg" length="1200699"/>
   <atom:id>mid:10000015c@xios.xcerion.com</atom:id>
   <dc:folder>8589934599</dc:folder>
   <dc:document>4294967644</dc:document>
   <dc:width>2048</dc:width>
   <dc:height>1152</dc:height>
   <dc:thumbnail>32769</dc:thumbnail>
   <dc:date>2012-10-08T10:15:28</dc:date>
   <dc:orientation>right-top</dc:orientation>
   <dc:camera-make>samsung</dc:camera-make>
   <dc:camera-model>GT-I9300</dc:camera-model>
   <dc:dpi>96x96</dc:dpi>
</atom:entry>
```

/v1/users/{user_id}/favorites/{favorite_id}/{folder_id}/folders/{child_folder_id}

user_id is the numeric user-ID.
favorite_id is the numeric favorite-ID.
folder_id is the numeric id of a parent folder.
child_folder_id is the numeric id of a child folder.

PUT

Renames folder with *child_folder_id* in folder with *folder_id* in favorite with *favorite_id*.
This is used for inbound sharing through a favorite.

REQUEST

```
<renameFolder newName={newName} />
```

DELETE

Deletes folder with *child_folder_id* in folder with *folder_id* in favorite with *favorite_id*.
This is used for inbound sharing through a favorite.

/v1/users/{user_id}/favorites/{favorite_id}/{folder_id}/documents/{document_id}

user_id is the numeric user-ID.
favorite_id is the numeric favorite-ID.
folder_id is the numeric id of a folder.
document_id is the numeric id of a document.

PUT

Renames document with *document_id* in folder with *folder_id* in favorite with *favorite_id*.
This is used for inbound sharing through a favorite.

REQUEST

```
<renameDocument newName={newName} />
```

DELETE

Deletes document with *document_id* in folder with *folder_id* in favorite with *favorite_id*.
This is used for inbound sharing through a favorite.

/v1/users/{user_id}/favorites/{favorite_id}/webshare[/{path}]

user_id is the numeric user-ID.
favorite_id is the numeric favorite-ID.
path is the optional path to a specific folder in the webshare.

GET

Returns the webshare pointed to by the favorite as an atom:feed.
If the webshare contains exactly one folder, the content of this folder is returned.

Paging

- offset: >= 0 (Default: 0)

count: 0 <= 1000 (Default: 100)

order: title|size|created (Default: title)

Request

/v1/users/12884901890/favorites/13/webshare

Response

```
<?xml version="1.0"?>
<atom:feed xmlns:atom="http://www.w3.org/2005/Atom" xmlns:os="http://a9.com/-/spec/opense
arch/1.1/" xmlns:dc="http://xcerion.com/directory.xsd" xmlns:ni="http://xcerion.com/noind
ex.xsd">
 <os:totalResults>3</os:totalResults>
 <os:startIndex>0</os:startIndex>
 <os:itemsPerPage>100</os:itemsPerPage>
 <atom:link rel="self" href="http://localhost/v1/users/12884901890/favorites/13/webshare/
?offset=0&amp;count=100"/>
 <atom:author>
  <atom:name>John Doe</atom:name>
  <atom:uri>test2</atom:uri>
 </atom:author>
 <atom:title>Birds</atom:title>
 <atom:updated>2012-02-13T15:08:16Z</atom:updated>
 <atom:published>2012-02-13T15:08:16Z</atom:published>
 <atom:summary>Bird pictures</atom:summary>
 <atom:entry>
  <atom:title>Nature Album</atom:title>
  <atom:published>2012-02-13T15:08:16Z</atom:published>
  <atom:updated>2012-02-13T15:08:16Z</atom:updated>
  <atom:id>urn:x-i-o-s:folder:8589934601</atom:id>
  <dc:folder>8589934600</dc:folder>
  <dc:childFolder>8589934601</dc:childFolder>
  <atom:link rel="alternate" type="text/html" href="test2/ws/13/Nature%20Album/"/>
  <atom:link rel="enclosure" type="application/atom+xml" href="http://localhost/v1/users/
12884901890/favorites/13/webshare/Nature%20Album/"/>
 </atom:entry>
 <atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:dc="http://xcerion.com/direct
ory.xsd">
  <atom:title>Berg.jpg</atom:title>
  <atom:published>2012-01-26T20:15:18Z</atom:published>
  <atom:updated>2012-01-31T12:54:27Z</atom:updated>
  <atom:link rel="alternate" type="text/html" href="test2/ws/13/Berg.jpg" length="489431"
/>
  <atom:id>mid:100000001@xios.xcerion.com</atom:id>
  <dc:folder>8589934600</dc:folder>
```

```
   <dc:document>4294967297</dc:document>
   <dc:width>1280</dc:width>
   <dc:height>720</dc:height>
   <dc:thumbnail>32769</dc:thumbnail>
   <dc:date>2008-05-10T21:37:18</dc:date>
   <dc:orientation>top-left</dc:orientation>
   <dc:camera-make>SONY</dc:camera-make>
   <dc:camera-model>HDR-CX7EK</dc:camera-model>
   <dc:flash>No, compulsory</dc:flash>
   <atom:rights>Copyright Daniel Arthursson 2008, All rights reserved. May only be used as
demo picture on Xcerions iCloud.</atom:rights>
   <dc:dpi>72x72</dc:dpi>
   <dc:focal>5.4 mm</dc:focal>
   <dc:aperture>F4.8</dc:aperture>
   <dc:exposure>1/150 s</dc:exposure>
   <atom:link rel="enclosure" type="image/jpeg" href="http://localhost/v1/users/1288490189
0/favorites/13/webshare/Berg.jpg"/>
  </atom:entry>
  <atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:dc="http://xcerion.com/direct
ory.xsd">
   <atom:title>Halmstad.jpg</atom:title>
   <atom:published>2012-01-26T20:15:18Z</atom:published>
   <atom:updated>2012-01-31T12:54:28Z</atom:updated>
   <atom:link rel="alternate" type="text/html" href="test2/ws/13/Halmstad.jpg" length="449
732"/>
   <atom:id>mid:100000002@xios.xcerion.com</atom:id>
   <dc:folder>8589934600</dc:folder>
   <dc:document>4294967298</dc:document>
   <dc:width>1280</dc:width>
   <dc:height>960</dc:height>
   <dc:thumbnail>32769</dc:thumbnail>
   <dc:date>2007-08-04T11:02:15</dc:date>
   <dc:orientation>top-left</dc:orientation>
   <dc:camera-make>CASIO COMPUTER CO.,LTD</dc:camera-make>
   <dc:camera-model>EX-Z3</dc:camera-model>
   <dc:flash>No, compulsory</dc:flash>
   <atom:rights>Copyright Daniel Arthursson 2008, All rights reserved. May only be used as
demo picture on Xcerions iCloud.</atom:rights>
   <dc:dpi>72x72</dc:dpi>
   <dc:focal>5.8 mm</dc:focal>
   <dc:aperture>F4.3</dc:aperture>
   <dc:exposure>1/400 s</dc:exposure>
   <atom:link rel="enclosure" type="image/jpeg" href="http://localhost/v1/users/1288490189
0/favorites/13/webshare/Halmstad.jpg"/>
  </atom:entry>
</atom:feed>
```

Errors

Error	Cause
Forbidden	*Username and password does not match the provided userId*
You need to provide a valid password to access this webshare (Forbidden 403)	*The password provided for the webshare is incorrect*

POST

Creates a folder in the specified folder with folderId.
This is used for inbound sharing through a favorite.
path is ignored.

Request

A request should look like this.

```
<newFolder folderId=\"{folder_id}\" childName=\"{NameOfNewFolder}\" />
```

Response

```
<newFolderId>1234</newFolderId>
```