MashUp

# General description

MashUp is a web service, which lets the user register their cloud storage accounts with a web-service, which becomes a single, unified cloud storage account. When users upload files to MashUp, MashUp disassembles the files into parts and stores them in different cloud accounts. Then, when the file is to be downloaded, MashUp will assemble it back and present it to the user.

Structure of the service:

* /register – registration resource
* /login – authentication resource
* /clouds – container resource for different cloud providers
* /files – root of the user’s MashUp file system

# Account services:

Provides the possibility to securely register and log in to the application

## Register:

### Description:

Creates a new, empty account for MashUp

### HTTP Request:

Method: POST  
Path: /register

Headers: Content-Type: application/x-www-form-urlencoded  
Body: username=<username> and password=<password> URL encoded

### Response:

Status:

* 200 OK if the request was accepted
* 400 Bad Request if:
  + The username contains characters besides [a-Z][0-9] and \_
  + Username is already taken
  + Username is shorter than 3 or longer than 60 characters
  + Username is a reserved name

### Example:

**Request:**

Method: POST  
Path: /register  
Headers: Content-Type: application/x-www-form-urlencoded  
Body: username=john\_doe&password=secret

**Response:**

Status: 200 OK

## Login:

### Description:

Checks if the provided user and password are correct, and creates a session for the user if they are. Session id is returned

### HTTP Request:

Method: POST  
Path: /login

Headers: Content-Type: application/json  
Body: JSON dictionary containing username=<username> and password=<password>

### Response:

* 200 OK if the login succeeded  
  **Body:** JSON containing the session\_id:

{ “session\_id”: <session\_id> }

* 400 Bad Request if the request is not a valid login request
* 403 Unauthorized if authentication failed

### Example:

**Request:**

Method: POST  
Path: /login  
Headers: Content-Type: application/json  
Body: { “username”=”john\_doe” “password”=”secret” }

**Response:**

Status: 200 OK

Body: { “session\_id”:”Ava442...PLl2” }

## Log out:

### Description:

Terminates the current session

### HTTP Request:

Method: POST  
Path: /logout

Headers: X\_SESSION\_ID=<session\_id>

### Response:

* 200 OK if the logout succeeded
* 403 Unauthorized if authentication failed

### Example:

**Request:**

Method: POST  
Path: /logout

Headers: X\_SESSION\_ID:ApgD3...gga1

**Response:**

Status: 200 OK

Session ApgD3...gga1 has been terminated

# Cloud management:

Provides the ability to view and add new clouds to the MashUp account. All calls require the X\_SESSION\_ID header for authentication:

X\_SESSION\_ID=<session\_id>

Unauthenticated calls will return:

403 Forbidden

## Create a cloud authorization URI:

### Description:

Generates a URI that redirects the user to the specified clouds’ registration page. The URI will be mapped to the session\_id, so the user needs to remain logged in for the duration of authorization.

### HTTP Request:

Method: POST  
Path: /clouds/<cloud\_name>/auth\_info

<cloud\_name> can be either “dropbox” or “onedrive”

### Response:

* 200 OK if the request was accepted  
  Body: JSON: { “authorize\_uri”:<uri\_to\_authorize\_mashup> }

### Example:

**Request:**

Method: POST  
Path: /clouds/dropbox/auth\_info

**Response:**

Status: 200 OK

Body: { “authorize\_uri”:”https://api.dropbox.com/...” }

## List clouds:

### Description:

Lists all the clouds belonging to the user

### HTTP Request:

Method: GET  
Path: /clouds/list

### Response:

* 200 OK if the request was legal  
  Body: JSON containing data for all clouds belonging to the user  
  [{ “id”:<universal\_cloud\_id>, “name”:<cloud\_name>, “provider”:<cloud\_provider>, “quota”:<maxmimum\_space\_mashup\_can\_use\_in\_bytes”, “taken”:<space\_already\_used\_by\_mashup\_in\_bytes”}, ... ]

### Example:

**Request:**

Method: GET  
Path: /clouds/list

**Response:**

Status: 200 OK

Body: [{ “id”:1, “name”:”dropbox\_1”, “provider”:”dropbox”, “quota”:684161441, “taken”:12591591 }, { “id”:2, “name”:”onedrive\_1”, “provider”:”onedrive”, “quota”:967911436, “taken”:76814610}]

# File system:

Provides the interface for storing and retrieving data from MashUp, as well as basic file system capabilities – creating directories, listing directories, traversing the directory tree, removing files and directories.

All calls require the X\_SESSION\_ID header for authentication:

X\_SESSION\_ID=<session\_id>

Unauthenticated calls will return:

403 Forbidden

## Retrieve file:

### Description:

Retrieves data from a file stored with MashUp

### HTTP Request:

Method: GET  
Path: /files/path/to/your/file   
Query: start=<first byte in the file to be downloaded> (optional)

length=<number of bytes to be downloaded> (optional)

If start is not provided, 0 is assumed

If length is not provided, it is assumed that all the bytes after start should be returned

### Response:

* Status: 200 OK if the request was accepted  
  Body: Binary representation of the specified range of the file’s contents
* Status: 404 Not Found if the file does not exist
* Status: 400 Bad Request if the path is not a valid path to a resource (it contains a file as part of the path)

### Example:

**Requested file contents:**Hello world!

**Request:**

Method: GET  
Path: /files/folder/example.txt   
Query: start=2&length=9

**Response:**

Status: 200 OK

Body: llo world

## Store file:

### Description:

Stores a file in MashUp. If the catalog for the file does not exist, it is created. Supports uploading the file in separate requests

### Upload session:

Upload sessions let the users upload the files in multiple requests. Once an upload session is started for a given file, subsequent upload calls will append to that file. Once a request to end the session is sent, the file upload is finalized.

### HTTP Request:

Method: PUT  
Path: /files/path/to/your/file   
Query: item\_type=file&session=<add or end or cancel>

Body: binary data to be put in the selected file

If session=add data from body will be added to an ongoing upload session for that file. If no session is ready for that file, it will be created before adding data.

If session=end data from body will be added to an ongoing upload session for that file. After that the session will be closed and upload will be finalized. If no session is ready for that file, it will be created before adding data.

If session=cancel the upload operation for that file is cancelled and resources are freed.

### Response:

* Status: 200 OK if the request was accepted
* Status: 400 Bad Request if the path is not a valid path to a resource (it contains a file as part of the path)

### Example:

**Request:**

Method: PUT  
Path: /files/folder/example.txt   
Query: item\_type=file&session=add

Body: Hel

**Response:**

Status: 200 OK

**Request:**

Method: PUT  
Path: /files/folder/example.txt   
Query: item\_type=file&session=end

Body: lo world!

**Response:**

Status: 200 OK

**Request:**

Method: GET  
Path: /files/folder/example.txt

**Response:**

Status: 200 OK

Body: Hello world!

## List folder:

### Description:

Lists the contents of the specified folder

### HTTP Request:

Method: GET  
Path: /files/path/to/your/folder

### Response:

* Status: 200 OK if the request was accepted  
  Body: JSON containing the children of the folder  
  { “children”:[{“name”:<item\_name>, “type”:<file/folder>}, ...] }
* Status: 404 Not Found if the folder does not exist
* Status: 400 Bad Request if the path is not a valid path to a resource (it contains a file as part of the path)

### Example:

**Requested folder contents:**images/  
videos/  
readme.md  
file.txt

**Request:**

Method: GET  
Path: /files/folder/

**Response:**

Status: 200 OK

Body: { “children”:[{“name”:”images”, “type”:”folder”}, {“name”:”videos”, “type”:”folder”}, {“name”:”readme.md”, “type”:”file”}, {“name”:”file.txt”, “type”:”file”}]

## Remove:

### Description:

Removes a file or a folder and all its content

### HTTP Request:

Method: DELETE  
Path: /files/path/to/your/item

### Response:

* Status: 200 OK if the item is deleted
* Status: 404 Not Found if the item does not exist
* Status: 400 Bad Request if the path is not a valid path to a resource (it contains a file as part of the path)

### Example:

**/folder/ contents:**images/  
videos/  
readme.md  
file.txt

**Request:**

Method: DELETE  
Path: /files/folder/videos

**Response:**

Status: 200 OK

**/folder/ contents afterwards:**

images/  
readme.md  
file.txt

## Make directory:

### Description:

Creates the requested catalog, and all the catalogs leading to that catalog in the path

### HTTP Request:

Method: PUT  
Path: /files/path/to/your/catalog  
Query: item\_type=folder

### Response:

* Status: 200 OK if the catalog was created
* Status: 400 Bad Request if the path is not a valid path to a resource (it contains a file as part of the path)

### Example:

**/ content before:**movies/  
readme.md

**Request:**

Method: PUT  
Path: /files/folder/folder2   
Query: item\_type=folder

**Response:**Status: 200 OK

**/ content after:**folder/movies/  
readme.md

**/folder/ content before:**folder2/

## Move item:

### Description:

Moves an item into a different path

### HTTP Request:

Method: PUT  
Path: /files/new/path/to/your/item  
Query: item\_type=other\_item

Body: path to the item to be moved (as a string)

### Response:

* Status: 200 OK if the item was moved
* Status: 400 Bad Request if the path is not a valid path to a resource or if the path is occupied

### Example:

**Cloud tree before:**movies/  
 movies/video2.mp4  
pictures/  
 pictures/video.mp4

**Request:**

Method: PUT  
Path: /files/movies/moved\_video.mp4   
Query: item\_type=other\_item

**Response:**

Status: 200 OK

**Cloud tree after:**movies/  
 movies/video2.mp4  
 movies/moved\_video.mp4  
pictures/