The current draft includes service descriptions for the requirements marked with green in the requirements document.

Introduction

The API is divided into sections, one for each URI identifying a part of the service.
Each service description starts with an URI (URL). Any part of the URL which is written in uppercase is used by the service to identify some resource. An example */accounts/USERNAME* URI could be:

/accounts/seah which identifies the account having the username “seah”

After the URL, a short description of the resource is stated, followed by any headers expected by the service. For most of the requests, these are a content-accept header “Accept: application/json”, which states that the response should be in JSON format, and the custom “Token: <token>” header, where <token> is an actual token retrieved by the /auth service.

After the “Headers” section, a section follows which states which types of API clients may use this service in one way or another. The “accessible by” section may state any user type of the system, i.e. Unauthenticated (normal user), Customer (authenticated user), Admin, or Content Provider. An API client is considered to be one of the user types based on the username they have authenticated as.
When a user type is listed, it does not mean that it may perform any type of request on the resource (the service offered by the URL), just that it may perform some of the listed operations on the resource.

Next, each HTTP method supported by the resource is listed, in the following order: GET, PUT, POST, DELETE. Any query parameters to the right of the method, denotes the parameters the method expects (i.e. requires) to be passed.

For example: **GET ?user=USERNAME&password=PASSWORD**
The above denotes a GET request which requires two parameters to be passed along, namely “user” and “password”. The parts in big are in the description of the HTTP method used as references to the values.

The query parameters beside a HTTP method may also be enclosed in square brackets.

For example: **GET [?types=ACP&info=username&include_banned=false]**
Query parameters enclosed in bracket define the default values for each parameter, if they are left out in the request.

For each HTTP method, a description is given on the what the method does, what input is

expected, and what responses as result may be produced. Input JSON objects sent in the message body of a request along with HTTP headers are marked in bold. JSON objects being returned in the message body of a response is also marked in bold, together with the produced HTTP status code.

JSON objects used to describe requests or responses contain all names of the required properties, along with example data. The order of the properties is not specified by this API. If no data is available for a specific property, the property will be left out. This applies to both requests and responses.
The accepted request message bodies and produced response messages might differ depending of the account type the client has authenticated as. This is described in the method descriptions as well.

As the last part of any HTTP method description for a resource, the possible failure responses are described. Each failure response is described by its HTTP status code and the reason why the service may return this code.
The most typical failure responses are **400 Bad Request** for requests which do not pass parameters as required and **403 Forbidden** for requests where the API client does not have permission to carry out the request. This is either because the client is not authenticated as the required user type, the token has expired, was not passed, or has become corrupted. Another reason may be that the the user, which the client authenticated as, after authentication has become banned.

If you have any questions related to this document, feel more than free to ask as many questions you like :)

REST API

/auth

Description: Authentication service.

Headers:

Accept: application /json

Accessible by:

Unauthenticated
Customers
Content Providers
Admins

GET ?user=USERNAME&password=PASSWORD

Retrieves an authentication token which is valid for X hours.
The "token" value of the response should be passed along with future requests that require the client to authenticate themselves.
When the token is passed, it should be so in a custom "Token" HTTP header.
The "expires" value of the response says when the token will expire and is in the format "yyyy-MM-dd HH:mm:ss zzz".
When the token expires it is no longer possible to use it for authentication purposes, and a new token must be retrieved.

The PASSWORD query parameter must match the password associated with the account identified by the USERNAME query parameter.

Successful call response example:
Status: 200 OK
{
token:
"MDAzNjEyMzQ3OCB8IDIwMTMtMDMtMjEgMTk6NDU6MjkgKzAxOjAwIHwgUGhpbGlw",
expires: "2013-03-21 19:45:29 +01:00"
}

Failure responses:

401 Unauthorized
Reason: The passed username and password credentials did not match any account.

403 Forbidden
Reason: The credentials matched an account, but the account is banned.

/accounts

Description: Collection of all accounts.

Headers:

Accept: application/json
Token: <token>

Accessible by:

Admins

GET [?types=ACP&info=username&include_banned=false]

Retrieves an overview of all accounts matching some criteria.
By default only the usernames of every of unbanned account of any type are returned:
Status: 200 OK
["username1", "username2", "username3", ...]

To include banned accounts as well, set the "include_banned" parameter to "true".

Filtering by account type may be applied by specifying the "types" parameter explicitly.
A = include admin accounts
C = include customer accounts
P = include content provider accounts

Example: /accounts?types=AP will return admin and content provider accounts, but not customer accounts.

More information may be returned by setting the "info" parameter to "more":
Status: 200 OK
[{
"user": "username1",
"email": "some.email@address.com",
"type": "Admin",
"banned": false // Whether the account has been banned by an admin
}, {
"user": "username2",
"email": "some.other@email.com",
"type": "Customer",
"banned": false
}, {
"user": "username3",
"email": "some.great@gmail.com",

“type”: "Content Provider",
“banned”: “false”
},
...]

Even more information may be returned by setting the "info" parameter to "detailed”:
Status: 200 OK
[{
“user”: "username1",
“email”: "some.email@address.com",
“type”: "Admin",
“banned”: “false”, // Whether the account has been banned by an admin
“created”: "2013-03-21 19:45:29 +01:00", // when account was created
“authenticated”: "2013-03-21 19:45:29 +01:00" // when the user last authenticated
}, {
“user”: "username2",
“email”: "some.other@email.com",
“type”: "Customer",
“banned”: false,
“name”: “Lynette Seah”,
“address”: “125 Bukit Merah Lane 1\n#01-170”,
“postal”: “150125”,
“country”: “Singapore”,
“birth”: “1991-06-18”,
“about”: "the about me field used by SMU",
“credits”: “0”,
“created”: "2013-03-21 19:45:29 +01:00", // when account was created
“authenticated”: "2013-03-21 19:45:29 +01:00" // when the user last authenticated
}, {
“user”: "username3",
“email”: "some.great@gmail.com",
“type”: "Content Provider",
“banned”: “false”,
“name”: “Some Company Inc.”,
“address”: “Nørrebrogade 54D st. tv\nKøbenhavn N”,
“postal”: “2200”,
“country”: “Denmark”,
“created”: "2013-03-21 19:45:29 +01:00", // when account was created
“authenticated”: "2013-03-21 19:45:29 +01:00" // when the user last authenticated
},
...]

Failure responses:

400 Bad Request

Reason: The "types" parameter was empty or contained other letters than the ones supported, the "info" parameter was neither "username", "more", or “detailed”, or the "include_banned" parameter was neither "true" or "false".

403 Forbidden

Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The user account authenticated as may have become banned since authentication.

/accounts/USERNAME

Description: Information associated with a specific account. USERNAME is case insensitive, i.e. “seah” is the same as “SEAH”, “Seah”, and “SeAh”.

Headers:

Accept: application/json
Token: <token>

Accessible by:

Admins	
Customers	(GET, PUT, only own account)
Content Providers	(GET, PUT, only own account)
Unauthenticated	(POST)

GET

Retrieves the information associated with the account having username USERNAME.
The following JSON values are returned for all successful responses:

Status: 200 OK
{
“email”: "the.email@address.com",
“type”: "Customer" // or "Admin" or "Content Provider"
}

Accounts of type "Customer" also return the following JSON values:
{
“name”: "Full Name",
“address”: “125 Bukit Merah Lane 1\n#01-170”,
“postal”: “150125”,
“country”: “Singapore”,
“credits”: “0”,
“birth”: “1991-06-18”,
“about”: "the about me field used by SMU"
}

Accounts of type “Content Provider” also return the following JSON values:
{
“name”: "Full Name",
“address”: “Nørrebrogade 54D st. tv\nKøbenhavn N”,
“postal”: “2200”,

“country”: “Denmark”
}

When an admin requests this resource, the following JSON values are also returned:
{
“banned”: “false”, // or true, if the account has been banned
“created”: "2013-03-21 19:45:29 +01:00", // when account was created
“authenticated”: "2013-03-21 19:45:29 +01:00" // when the user last authenticated
}

Admins may retrieve information about any account, but a customer or content provider may only access the account of the user they have authenticated as.

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The user account authenticated as may have become banned since authentication.

404 Not Found
Reason: No account with the associated username USERNAME exists.

PUT

Updates the information associated with the account having the username USERNAME.

Clients authenticated as customers or content providers are only allowed to update information for the account they have authenticated as. The values to update are each part of the JSON object which is sent in the message body of the request. Any of the values described below may be left out, if no new value for the item is wanted.

An information update to a customer account may include any of the following values, if the client is authenticated as customer:
{
“email”: "some.other@email.com",
“name”: “Lynette Seah”,
“address”: “125 Bukit Merah Lane 1\n#01-170”,
“postal”: “150125”,
“country”: “Singapore”, // Must be one of the values returned by /countries
“birth”: “1991-06-18”,
“about”: "the about me field used by SMU",
“password”: “my new password”
}
NB: the value of country must be one of the values returned by */countries*

If the client is authenticated as admin, the JSON object may also include the following values:
{
“credits”: “0”, // Must be a natural number (non-negative integer)
“banned”: “true”
}

An information update to a content provider may include any of the following values, if the client is authenticated as content provider:
{
“email”: "some.dude@greatness.com",
“name”: “Greatness Inc.”,
“address”: “Nørrebrogade 54D st. tv\nKøbenhavn N”,
“postal”: “2200”,
“country”: “Denmark”, // Must be one of the values returned by /countries
“password”: “my new password”
}
NB: the value of country must be one of the values returned by */countries*

If the client is authenticated as admin, the JSON object may also include the following values:
{
“banned”: “true” // or “false”
}

An information update to an admin may include any of the following values, if the client is authenticated as admin:
{
“email”: "the.email@adress.com",
“banned”: “true”, // or “false”
“password”: “the new password”
}

In any case, **204 No Content** will be returned as status code if the operation was successful.

Failure responses:

400 Bad Request
Reason: The message body JSON was malformed, the values included had illegal values, and/or properties that was not understood was included.

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The user account authenticated as may have become banned since authentication.

404 Not Found
Reason: No account with the associated username USERNAME exists.

413 Request Entity Too Large
Reason: The server denied to fulfill the request, because one or more of the JSON properties had values which were too large.

POST

Creates an account with username USERNAME.
The following JSON must be sent in the message body to create a customer account:

```
{
“password”: “password for the account”,
“email”: "the.email@address.com"
}
```

When creating a customer account, the JSON may also include:

```
{
“name”: "Full Name",
“address”: “125 Bukit Merah Lane 1\n#01-170”,
“postal”: 150125,
“country”: “Singapore”, // Must be one of the values returned by /countries
“birth”: “1991-06-18”,
“about”: "the about me field used by SMU",
“type”: “Customer”
}
```

NB: the value of country must be one of the values returned by */countries*

A client authenticated as admin may also create new admin and content provider accounts.
To create an admin account, send the following JSON in the message body:

```
{
“password”: “password for the account”,
“email”: "the.email@address.com",
“type”: “Admin”
}
```

To create a content provider account, send the following JSON in the message body:

```
{
“password”: “password for the account”,
“email”: "the.email@address.com",
“type”: “Content Provider”
}
```

The content provider account JSON message body may also contain:
{
“name”: "Company Name",
“address”: “Nørrebrogade 54D st. tv\nKøbenhavn N”,
“postal”: “2200”,
“country”: “Denmark” // Must be one of the values returned by /countries
}
NB: the value of country must be one of the values returned by */countries*

When a client successfully performs this request, the server will respond with:
201 Created
Location: /accounts/USERNAME

Failure responses:

400 Bad Request
Reason: The message body JSON was malformed and/or did not contain all values needed to complete the request. This code may also be returned, if one or more passed values are invalid (The username was too long, for instance).

403 Forbidden
Reason: If another account type than “Customer” was tried to be created, and the passed token (if any) was expired, malformed, or did not grant permission to access this resource. The user account authenticated as may have become banned since authentication.

409 Conflict
Reason: Another account already uses this username.

413 Request Entity Too Large
Reason: The server denied to fulfill the request, because one or more of the JSON properties had values which were too large.

/countries

Description: Collection of all countries.

Headers:

Accept: application/json

Accessible by:

Admins
Customers
Content Providers
Unauthenticated

GET

Retrieves an overview of all countries accepted as value for the country property of an account address. When one assign a country to an account, it must be a country from this list.

Any type of client may perform this request.

If successful, the server will respond with a list of all names of accepted countries, for example:

Status: 200 OK
["Denmark", "Singapore", ...]

/products

Description: Collection of all products
Headers:
Accept: application/json
Token: <token>

Accessible by:
Admins
Content Providers
Customers
Unauthenticated

GET [search=&type=&info=id&unpublished=false]

Retrieves all products matching some criteria.
By default the ID of every published product is returned:
Status: 200 OK
[1, 3, 7, ...]

To include unpublished products as well, set the "unpublished" parameter to "true".
Only clients authenticated as admins may view the unpublished products of the store.

By default the "type" parameter has no value, meaning that no filtering based on type is done. To only return products whose types are part of a set of defined types, set the value of the parameter to the name of each accepted type separated by a pipe ("|").

Example: */products?type=music|ebook* will return every product whose type either is "music" or "ebook". Types are case insensitive, meaning that the previous query will return the same results as */products?type=MUSIC|eBoOk*
If a type is passed which is not defined by the system or no products of the given type exists, the type is simply ignored.

More filtering is possible by (also) setting the "search" parameter. By default the parameter has no value meaning that the filter is not applied. By setting it to one or more search words separated by a plus ("+"), only products whose title, description, and meta data in combination contains all search words are returned. If the "type" parameter is set the matched products are also filtered by product type. A piece of meta information matches a search word if either the type of the meta data or its value contains the search word. Search words must match whole words to match: The search word "car" does not match the text "card", but it does match "Car" (search word are matched case insensitively).

Example: */products?search=2010* could return a list of the following products, if they existed in the system:

- The film with the title “2010” (http://www.imdb.com/title/tt0086837/), which has been matched based on product title.
- The music album “2010 Grammy Nominees” (http://www.amazon.com/2010-Grammy-Nominees-Various-Artists/dp/B002ZFEQ80), which also has been matched based on product title.
- $The film “Inception” released in 2010 (http://www.imdb.com/title/tt1375666/), which has been matched on the meta data “released: 2010”.
- The film “District 9” released in 2009 (http://www.imdb.com/title/tt1136608/), which has been matched by its description (“... story takes place in 2010 …”).

More information about products may also be returned by setting the “info” parameter to “more”:

```
Status: 200 OK
[{
“title”: “Head First REST Architecture”,
“description”: “This is the description of the product”,
“type”: “ebook”, // film, music, ...
“price”: {
        buy: 20         // Prices are in credits
},
“owner”: “username”         // The username of the content provider owning this product
}, {
“title”: “Inception”,
“description”: “This is the description of the product”,
“type”: “film”, // ebook, music, ...
“price”: {
        “b y”: “30”,    // Prices are in credits
        “rent”: “15”   // “rent” only included if the product is rentable
},
“owner”: “username”         // The username of the content provider owning this product
},
...]
```

All data about all products is returned by setting the “info” parameter to “detailed”:

```
Status: 200 OK
[{
“title”: “Head First REST Architecture”,
“description”: “This is the description of the product”,
“type”: “ebook”, // film, music, ...
“price”: {
```

“buy”: “20” // Prices are in credits
},
“rating”: {
“score”: 0, // The average rating of the product
“count”: 0 // Not rated yet
},
“owner”: “username”, // The username of the content provider owning this product
“meta”: [
{“name”: “meta1”, “value”: “some text”},
{“name”: “meta2”, “value”: “1337”},
...
]
}, {
“title”: “Inception”,
“description”: “This is the description of the product”,
“type”: “film”, // ebook, music, ...
“price”: {
“buy”: “30”, // Prices are in credits
“rent”: “15” // “rent” only included if the product is rentable
},
“rating”: {
“score”: “-5”, // The average rating of the product (from -5 to 5)
“count”: “1” // How many customers have rated the product
},
“owner”: “username”, // The username of the content provider owning this product
“meta”: [
{“name”: “meta2”, “value”: “1337”},
{“name”: “meta5”, “value”: “Products do not necessary have the same meta data”},
...
]
},
...]

If the client is authenticated as admin, the JSON returned when “info” is set to “more” or “detailed” also includes:

{
“published”: “true”, // or “false”, if the product is not made visible/available in the store
}

Failure responses:

400 Bad Request
Reason: The "info" parameter was neither "id", "more", or “detailed”, or the "unpublished" parameter was neither "true" or "false".

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource in the requested way. The user account authenticated as may have become banned since authentication.

/products/types

Description: Collection of all product types supported by the service.

Headers:

Accept: application/json

Accessible by:

Admins
Customers
Content Providers
Unauthenticated

GET

Retrieves an overview of all product types accepted as type for products.
When one creates a new product, its type must be one of the types returned as response to this request.

Any type of client may perform this request.

If successful, the server will respond with a list of all product types supported by the service, for example:

Status: 200 OK
["ebook", "music", ...]

/products/ID

Description: Information associated with a specific product.
Headers:

Accept:	application/json
Token:	<token>
Content-Type:	MIME for the product media to upload
Content-Length:	Byte length of uploaded product media

Accessible by:

Admins	
Content Providers	
Customers	(GET only)
Unauthenticated	(GET only)

GET

Retrieves the information associated with the product having the id ID.
Only clients authenticated as admins or as the content provider owning the product may view it, if it currently is unpublished. Customers and Unauthenticated users may only view it, if it is published.

Example response:

Status: 200 OK

```
{
“title”: “Inception”,
“description”: “This is the description of the product”,
“type”: “film”,
“price”: {
        “buy”: “30”,  // Prices are in credits
        “rent”: “15”   // “rent” only included if the product is rentable
},
“rating”: {
        “score”: “4.6”, // The average rating of the product
        “count”: “3”      // How many customers have rated the product
},
“owner”: “username”, // The username of the content provider owning this product
“meta”:        [
        {“name”: “meta2”, “value”: “1337”},
        {“name”: “meta5”, “value”: “Products do not necessary have the same meta data”},
```

...
]
}

If the client is authenticated as admin or the content provider which owns the product, the following JSON properties are also included in the response:

{
“published”: “true” // or “false”, if it is not published
}

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No product with the given ID exists.

PUT

Updates the information associated with the product having the id ID.
Clients authenticated as content providers are only allowed to update information for products they own. Admins may update the information for any product,The values to update are each part of the JSON object which is sent in the message body of the request. Any of the values described below may be left out, if no new value for the item is wanted.

An information update to a product may include any of the following values:

{
“title”: “Inception”, // Must not be empty
“description”: “This is the description of the product”,
“price”: {
“buy”: “30”, // Prices are in credits, must be non-negative without decimals
“rent”: “15” // “rent” is only allowed if the product is rentable
},
“meta”: [
{“name”: “meta2”, “value”: “1337”},
{“name”: “meta5”, “value”: “Products do not necessary have the same meta data”},
...
],
“published”: “false” // Unpublishes the product
// If the product currently is unpublished, it may only be published if a

media file has been uploaded for it.
}

If the operation was successful, **204 No Content** will be returned as status code.

Failure responses:

400 Bad Request
Reason: The product was attempted to be given a “rent” price, but the product is not rentable. May also be returned if the JSON was malformed or contained unallowed properties.

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No product with the given ID exists.

409 Conflict
Reason: The product was attempted to be published, but no media file have been uploaded for it yet.

413 Request Entity Too Large
Reason: The server denied to fulfill the request, because one or more of the JSON properties had values which were too large.

POST

Uploads a new media file for the product having the id ID.

The message body of the request is the binary data of the media file to upload. The Content-Length header must be set to the length of the file in bytes. The Content-Type header must be set to the MIME type for the file.

The kind of file (described by its MIME type) is restricted by the type of the product:

Product Type	**Accepted MIME Types (Content-Type Header)**
ebook	*application/pdf*
audio	*audio/**, ** = ogg, mpeg, mp4, mid, wav, x-wav, x-aiff,* or *x-ms-wma*
music	*music/**, ** = ogg, mpeg, mp4, mid, wav, x-wav, x-aiff,* or *x-ms-wma*

film	*video/**, * = ogg, mp4, webm, H264,* or *x-ms-wmv*
series	See “film” type

The above for instance means that you cannot upload a PDF as the media file for a product with the type “music”.

Only clients which are authenticated as the content provider owning the particular product may perform this request.

If the operation was successful, **204 No Content** will be returned as status code.

Failure responses:

400 Bad Request
Reason: Headers required to perform the file upload were not given correctly, or the “Content-Type” was not accepted for the type of the product.

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No product with the given ID exists.

413 Request Entity Too Large
Reason: The server denied to fulfill the request, because the file which was attempted to be uploaded was too large.

DELETE

Deletes the product having the id ID.
Only clients which are authenticated as admin or as the content provider owning the particular product, may perform this request.

The product will be removed from the store and from the content provider’s account, but it will still remain in the purchase histories of customer (i.e. customers who have bought the product is still able to access it).

If the operation was successful, **204 No Content** will be returned as status code.

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found

Reason: No product with the given ID exists.

/products/ID/rating

Description: The rating of the product having id ID
Headers:

Accept: application/json
Token: <token>

Accessible by:

Admins	(GET only)
Content Providers	(GET only)
Customers	
Unauthenticated	(GET only)

GET

Retrieves the average rating of the product having id ID.The rating of a product is defined by the following scale, going from -5 to 5, both inclusive:

Score	Description
-5 to -1	Disliked. Customers in general do not like this product
0	Neutral. Unrated products have this rating.
1 to 5	Liked. Customers in general like this product.

When a client retrieves the average rating of a product it also retrieves the number of customers which have rated it. For example:

Status: 200 OK
{
score: 3.2, // 0 for unrated products
count: 24 // number of ratings
}

The product must be published in order for non admin, non content provider clients to access this resource.

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No product with the given ID exists.

PUT

Rate the product with id ID as the customer identified by the passed token.
The JSON passed in the message body should contain a single property named “rating”:

{
rating: 3 // -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, or 5
}

If the customer has rated the product earlier, the rating by the customer is updated to the new rating.
If the customer has not rated the product earlier, a new rating is registered for the customer and the count of ratings for the product is increased by 1.

Only customers may rate products, and only if they are published.

If the operation was successful, **204 No Content** will be returned as status code.

Failure responses:

400 Bad Request
Reason: The JSON request did not contain a “rating” property or its value was not an integer in the interval [-5 ; 5]

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No product with the given ID exists.

/products/ID/thumbnail

Description: The thumbnail image for the product with the id ID
Headers:

Accept: image/png, image/jpeg, image/gif
Content-Type: image/png, image/jpeg, image/gif
Token: <token>

Accessible by:

Admins
Content Providers
Customers (GET only)
Unauthenticated (GET only)

GET

Downloads the image thumbnail which has been uploaded for the product with the id ID. If no image has been uploaded yet, **404 Not Found** is returned.

A successful response will be accompanied with:
Status: 200 OK.
Content-Length: XXX (Length of image in bytes)
Content-Type: MIME (MIME-type of the image)

Clients not authenticated as admins or as the content provider which owns the particular product, may only perform this request if the product is published.

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No product with the given ID exists, or the product has been given no thumbnail yet.

PUT

Uploads a new thumbail image for the product having the id ID.

The message body of the request is the binary data of the image file to upload. The Content-Length header must be set to the length of the file in bytes. The Content-Type header must be set to the MIME type for the file, which must be one of the below:

Accepted Image Type	Accepted MIME Types (Content-Type Header)
PNG	image/png
GIF	image/gif
JPEG/JPG	image/jpeg

Only clients which are authenticated as admins or as the content provider owning the particular product may perform this request.

If the operation was successful, **204 No Content** will be returned as status code.

Failure responses:

400 Bad Request
Reason: Headers required to perform the file upload were not given correctly, or the "Content-Type" was not one of the accepted MIME types.

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No product with the given ID exists.

413 Request Entity Too Large
Reason: The server denied to fulfill the request, because the file which was attempted to be uploaded was too large.

/credits

Description: Handles purchases of credits
Headers:
Accept: application/json
Token: <token>

Accessible by:
Customers

POST

Purchases credits for the account associated with the authenticated client.
Credits are used to pay for product purchases and product rentals.

Only customers may perform this request.

The following JSON must be sent in the message body to buy credits:
{
“credits”: “30” // Any integer value > 0
}

If successful, the number of credits will inserted into the account associated with the client, and **204 No Content** will be returned as status code.

Failure responses:

400 Bad Request
Reason: The JSON request did contain any “credits” property, or its value was negative, zero, or not an integer.

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

413 Request Entity Too Large
Reason: The request could not be completed, because the account cannot store any more credits.

/accounts/CUSTOMER/purchases

Description: Collection of all products purchased by a customer
Headers:

Accept: application/json
Token: <token>

Accessible by:

Admins
Customers

GET [purchases=BR&info=id]

Retrieves all purchases made by the customer with the username CUSTOMER.
Only admins and the customer which have made the purchases may perform this request.

By default the ID of every purchase is returned:
Status: 200 OK
[1, 3, 7, ...]

The returned ids may be used like: **/accounts/CUSTOMER/products/RETURNED_ID_HERE**

To filter the returned purchases, the “purchases” query parameter may be set.
If the value includes “B”, purchases of **B**ought products will be included.
If the value includes “R”, purchases of **R**entals will be included.
If the value is left out, the value default to “BR”.

More information about the purchases may also be returned by setting the “info” parameter to “more”:
Status: 200 OK
[{
purchased: "2013-03-21 19:45:29 +01:00",// When the purchase was made
paid: 20, // How much was paid for the purchase
type: “R”, // if the purchase was a rental, “B” if the purchase was a buy
expires: "2013-03-23 19:45:29 +01:00", // The date and time a rental expires and no longer may be used. Left out for bought product purchases.
id: 12, // The id of the purchase: *//accounts/CUSTOMER/purchases/12*
product: 34 // The id of the purchased product
}, {
purchased: "2013-03-21 19:45:29 +01:00",// When the purchase was made
paid: 40, // How much was paid for the purchase

type: “B”, // if the purchase was a buy, “R” if the purchase was a rental
id: 10, // The id of the purchase: /accounts/CUSTOMER/purchases/10
product: 28 // The id of the purchased product
},
...]

Failure responses:

400 Bad Request
Reason: The "info" parameter was neither "id" or "more" or the "purchases" parameter contained characters other than “B” and “R”.

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource in the requested way. The user account authenticated as may have become banned since authentication.

404 Not Found
Reason: No customer account with the username CUSTOMER exists.

POST

Makes a combined purchase for the customer with the username CUSTOMER of each of the given products.
The customer must have enough credits on his account to purchase all the products at once. If the purchases would make the balance go below 0, the request is refused.

Only customers may perform this request, and only for their own account.

The following JSON must be sent in the message body to make purchases:
[{
“product”: “23”, // Id of the product to buy/rent
“purchased”: “B” // “B” for buy, “R” for rent
},
{
“product”: “24”, // Id of the product to buy/rent
“purchased”: “R” // “B” for buy, “R” for rent
},
...]

If successful, the products will be purchased, and the following response will be returned:
Status: 201 Created
[12, 13, 14, 15, ...] // List of the ids of the newly made purchases

To download a purchased product, perform a request to:
/accounts/CUSTOMER/purchases/ID/media

Failure responses:

400 Bad Request
Reason: The request was malformed. This could be because the list was empty, the “Ppr” property of any of the JSON objects was neither “B” or “R”, at least one of the listed products to be rented/bought was not rentable/buyable..

402 Payment Required
Reason: The customer did not have sufficient credits on his account to complete the purchase process. No purchase was made.

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No customer account with the username CUSTOMER exists.

409 Conflict
Reason: The “product” property of any of the JSON objects identified a non-existent or unpublished product.

/accounts/CUSTOMER/purchases/ID

Description: A specific purchase made by a customer
Headers:

Accept: application/json
Token: <token>

Accessible by:

Admins
Customers

GET

Retrieves the information about a specific product purchase.

Example response:
Status: 200 OK
[{
purchased: "2013-03-21 19:45:29 +01:00",// When the purchase was made
paid: 20, // How much was paid for the purchase
type: “R”, // if the purchase was a rental, “B” if the purchase was a buy
expires: "2013-03-23 19:45:29 +01:00", // The date and time a rental expires and no longer may be used. Left out for bought product purchases.
product: 34 // The id of the purchased product
}

Only clients authenticated as admin or as the customer which have made the purchase may access this resource.

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No customer account with the username CUSTOMER exists, or the customer account exists but has made no purchase with the id ID.

/accounts/CUSTOMER/purchases/ID/media

Description: Download a purchased product
Headers:
Accept: MIME-type of purchased product
Token: <token>

Accessible by:
Customers

GET

Downloads the media associated with the product of the purchase identified by ID.

Only clients authenticated as the customer which have made the purchase may access this resource. (username of authenticated client = CUSTOMER)

A successful response will be accompied with:
Status: 200 OK (For responses to “normal” requests)
Content-Length: XXX (Length of media in bytes)
Content-Type: MIME (MIME-type of the media)
OR
Status: 206 Partial Content (For responses to content-range requests)
Content-Length: XXX
Content-Type: MIME

The message body of the response contains the binary data of the requested media.

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

404 Not Found
Reason: No customer account with the username CUSTOMER exists, or the customer account exists but has made no purchase with the id ID.

410 Gone
Reason: The product was rented and the rental period has expired.

/accounts/PROVIDER/products

Description: Collection of all products owned by a content provider
Headers:

Accept: application/json
Token: <token>
Location: <URL>

Accessible by:

Admins	(GET only)
Content Providers	
Customers	(GET only)
Unauthenticated	(GET only)

GET [search=&type=&info=id&unpublished=false]

Retrieves all products matching some criteria, which are owned by the content provider having the username PROVIDER.
By default the ID of every published product is returned:
Status: 200 OK
[1, 3, 7, ...]

To include unpublished products as well, set the "unpublished" parameter to "true".
Only clients authenticated as the content provider which owns the products or as admins may view the unpublished products of the content provider.

By default the "type" parameter has no value, meaning that no filtering based on type is done. To only return products whose types are part of a set of defined types, set the value of the parameter to the name of each accepted type separated by a pipe ("|").
}
Example: */accounts/PROVIDER/products?type=music|ebook* will return every product owned by PROVIDER whose type either is "music" or "ebook". Types are case insensitive, meaning that the previous query will return the same results as */accounts/PROVIDER/products?type=MUSIC|eBoOk*
If a type is passed which is not defined by the system or no products of the given type exists, the type is simply ignored.

More filtering is possible by (also) setting the "search" parameter. By default the parameter has no value meaning that the filter is not applied. By setting it to one or more search words separated by a plus ("+"), only products whose title, description, and meta data in combination contains all search words are returned. If the "type" parameter is set the matched products are

also filtered by product type.A piece of meta information matches a search word if either the type of the meta data or its value contains the search word. Search words must match whole words to match: The search word “car” does not match the text “card”, but it does match “Car” (search word are matched case insensitively).

Example: */products?search=2010* could return a list of the following products, if the content provider had uploaded all of them:

- The film with the title “2010” (http://www.imdb.com/title/tt0086837/), which has been matched based on product title.
- The music album “2010 Grammy Nominees” (http://www.amazon.com/2010-Grammy-Nominees-Various-Artists/dp/B002ZFEQ80), which also has been matched based on product title.
- The film “Inception” released in 2010 (http://www.imdb.com/title/tt1375666/), which has been matched on the meta data “released: 2010”.
- The film “District 9” released in 2009 (http://www.imdb.com/title/tt1136608/), which has been matched by its description (“... story takes place in 2010 …”).

More information about products may also be returned by setting the “info” parameter to “more”:

Status: 200 OK
[{
“title”: “Head First REST Architecture”,
“description”: “This is the description of the product”,
“type”: “ebook”, // film, music, ...
“price”: {
“buy”: “20” // Prices are in credits
},
}, {
“title”: “Inception”,
“description”: “This is the description of the product”,
“type”: “film”, // ebook, music, ...
“price”: {
“buy”: “30”, // Prices are in credits
“rent”: “15” // “rent” only included if the product is rentable
},
},
...]

All data about all products is returned by setting the “info” parameter to “detailed”:

Status: 200 OK
[{
“title”: “Head First REST Architecture”,
“description”: “This is the description of the product”,

“type”: “ebook”, // film, music, ...
“price”: {
“buy”: “20” // Prices are in credits
},
“rating”: {
“score”: “0”, // The average rating of the product
“count”: “0” // Not rated yet
},
“meta”: [
{“name”: “meta1”, “value”: “some text”},
{“name”: “meta2”, “value”: “1337”},
...
]
}, {
“title”: “Inception”,
“description”: “This is the description of the product”,
“type”: “film”, // ebook, music, ...
“price”: {
“buy”: “30”, // Prices are in credits
“rent”: “15” // “rent” only included if the product is rentable
},
“rating”: {
“score”: “-5”, // The average rating of the product (from -5 to 5)
“count”: “1” // How many customers have rated the product
},
“meta”: [
{“name”: “meta2”, “value”: “1337”},
{“name”: “meta5”, “value”: “Products do not necessary have the same meta data”},
...
]
},
...]

If the client is authenticated as admin or as the content provider which owns the products, the JSON returned when “info” is set to “more” or “detailed” also includes:

{
“published”: “true”, // or “false”, if the product is not made visible/available in the store
}

Failure responses:

400 Bad Request
Reason: The "info" parameter was neither "id", "more", or “detailed”, or the "unpublished" parameter was neither "true" or "false".

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource in the requested way. The user account authenticated as may have become banned since authentication.

404 Not Found
Reason: No content provider account with the username PROVIDER exists.

POST

Creates a new, unpublished product owned by the content provider with username PROVIDER. Only the content provider which will be the owner of the new product may perform this request.

The following JSON must be sent in the message body to create a new product:

```
{
“title”: “Inception”,                                    // Required, must not be empty
“description”: “This is the description of the product”,        // Optional
“type”: “film”,         // Must be one of the types returned by /products/types
“price”: {       // Prices are in credits. Must be non-negative, without decimals
        “buy”: “30”,  // “buy” only allowed if the product is buyable
        “rent”: “15”   // “rent” only allowed if the product is rentable
},
“meta”: [       // May be empty or left out completely
        {“name”: “meta2”, “value”: “1337”},
        {“name”: “meta5”, “value”: “Products do not necessary have the same meta data”},
        ...
]
}
```

The “type” property must be one of the types defined by the system, as returned by */products/types*.
Only rentable products may be assigned a “rent” price.
Only buyable products may be assigned a “buy” price.
A product without a rent price cannot be rented, even though it is rentable.
A product without a buyprice cannot be bought, even though it is buyable.
Setting a price to 0, makes the product free to buy/rent.

If the product successfully has been created, the service will respond with:

Status: 201 Created
Location: /accounts/PROVIDER/products/ID (ID = id of newly created product)
{
id: ID // The id of the newly created, unpublished product
}

After the product is created, the client should upload a media file for the product by performing a POST request to: */accounts/PROVIDER/products/ID*
Only then may the product be published to the store.

Failure responses:

400 Bad Request
Reason: The title was left out or empty, the type did not match one defined by the system, the product pricing was illegal (see example comments), the request was malformed (bad syntax, properties where of unexpected types, required data was left out).

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The user account authenticated as may have become banned since authentication.

404 Not Found
Reason: No content provider account with the username PROVIDER exists.

413 Request Entity Too Large
Reason: The server denied to fulfill the request, because one or more of the JSON properties had values which were too large.

/accounts/PROVIDER/products/ID

Description: Alias for */products/ID*, for any product owned by PROVIDER
Headers:

Accept: application/json
Token: <token>

Accessible by:

Admins	(GET, PUT, DELETE only)
Content Providers	
Customers	(GET only)
Unauthenticated	(GET only)

GET

See GET description for */products/ID*

Failure response clarifications:

404 Not Found
Reason: No product with the given ID is owned by a content provider with the username PROVIDER.

PUT

See PUT description for */products/ID*

Failure response clarifications:

404 Not Found
Reason: No product with the given ID is owned by a content provider with the username PROVIDER.

POST

See POST description for */products/ID*

Failure response clarifications:

404 Not Found
Reason: No product with the given ID is owned by a content provider with the username PROVIDER.

DELETE

See DELETE description for */products/ID*

Failure response clarifications:

404 Not Found

Reason: No product with the given ID is owned by a content provider with the username PROVIDER.

/accounts/PROVIDER/products/ID/thumbnail

Description: Alias for */products/ID/thumbnail*, for any product owned by PROVIDER
Headers:

Accept: image/png, image/jpeg, image/gif
Content-Type: image/png, image/jpeg, image/gif
Token: <token>

Accessible by:

Admins
Content Providers
Customers (GET only)
Unauthenticated (GET only)

GET

See GET description for */products/ID/thumbnail*

Failure response clarifications:

404 Not Found
Reason: No product with the given ID is owned by a content provider with the username PROVIDER.

PUT

See PUT description for */products/ID/thumbnail*

Failure response clarifications:

404 Not Found
Reason: No product with the given ID is owned by a content provider with the username PROVIDER.

TEMPLATE

Description: Description
Headers:

Accept: application/json
Token: <token>

Accessible by:

Admins
Content Providers
Customers
Unauthenticated

GET

Some GET description

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

PUT

Some PUT description

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.

POST

Some POST description

Failure responses:

403 Forbidden
Reason: The passed token was expired, malformed, or did not grant permission to access this resource. The account may have become banned since authentication.