api-spec

Bergar Simonsen

May 14, 2013

Contents

1   api spec

1.1   M6, Movie Distribution Web Service API Specs

The web service uses RESTful concepts to provide a standardized and stateless programming interface, for interacting with the system.

Standard response
All responses consists of the following data:

Table 1: Response message objects

Message	Description
error_number	0 on success, otherwise a number indicating the relevant error. NOT TO BE CONFUSED WITH HTTP-Response-code! This is assuming the HTTP-Response-code is 200.
error_message	A message in English, describing the eventual error.

Security and encoding
All requests should be made with a valid API-key, and values hashed into a "Checksum". (TODO: Describe hashing details). Requests should be accompanied by a UTC timestamp and a nonce. All authenticated requests should contain an access token.
So every request should look something like:
`Resource]?auth=AUTH-STRING]&parama=xxx&paramb=yyy`

Where [AUTH-STRING] is something similar to: { "api-key" : "AB14" , "hmac" : "XZ45" , "time" : "213" , "nonce" : "XC98" ; "token" = "DE95" }

The only optional part of this auth object, is the token, which is only used after the user has successfully logged in.

Objects
Every area of the application has some certain objects with a well defined structure. These are described initially in the relevant section, and utilized throughout the API. This allows for client to receive and process these objects in a uniform manner.

Parameters
Are GET or POST arguments, depending on the request-type. The parameters documentation consists of a argument name along with a description of the parameter

Response
Is an associative JSON object, with fields corresponding to the left side column of the table. The right side of the column describes the data in the field of the returned response. All responses are wrapped in `Response<TYPE>`

objects. For example when calling the `GET user` method, it returns `Response<User>`.

1.2 Users (users, tokens, user data & user groups)

The users are at the heart of the system. All actions are performed by users, and their identities are associated with both actions and data entities within the system. Users are very simple, but can be extended to application specific needs. This is done by creating new user data types (unique text indexes, that can be used to store data for each user), and assigning data to the users. Tokens are used for accessing the application as a specific user, and is given upon submission of a valid email and password combination.

Table 2: User objects

Field	Description
id	The users id
email	E-mail of the user. False if currently logged in user is not permitted to read.
user_data	All data that exists for this user as an associative structure

Table 3: Token objects

Field	Description
token	A token string, to be used in further queries
issued	Date and time for issuance
expires	Date and time for expiration

Get user

Url structure:	/user/{ID}
Description:	Who is the user with the id {ID}?
Method:	GET
Parameters:	**id** Id of the user
Response:	**Response<User>**
	Response object containing the user object.

Get currently logged in user

Url structure:	/user/me
Description:	Who is the currently logged in user?
Method:	GET
Parameters:	None
Authorization:	Token
Response:	**Response<User>**
	Response message containing the user object.

Get all users (with parameters)

Url structure:	/user?group_id={group_id}&emailFilter={emailFilter}&limit={limit} &page={page}&order_by={order_by}&order={order}
Description:	Who are the users that match the given parameters
Method:	GET
Parameters:	**group_id** Id of a user group.
	Only show users who are members of this/these group(s)
	emailFilter Filter by email
	limit How many users to return?
	page Should there be an offset? Default = 1 means no offset.
	order_by Order by what column? Default = e-mail.
	order Order which way? Default = ASC.
Authorization:	Token
Response:	**Response<User>**
	Response object containing an array of user objects along with page number and count.

Post user access token

Url structure:	/user/token/{email}/{password}
Description:	Can i have an access-token with these credentials?
Method:	POST
Parameters:	**email** The users e-mail
	password An SHA-1 hash of the users password.
Response:	**Response<Token>**
	Response object containing the token.

Renew user access token

Url structure:	/user/token/renew
Description:	Can I renew this token?
Method:	POST
Parameters:	None
Response:	**Response<Token>**
	Response object containing the renewed token.

Create new user

Url structure:	/user
Description:	Create a new user with this data
Method:	POST
Content-Type	**application/json**
	User User object.
Response:	**Response<User>**
	Response object containing the new user.

Delete user

Url structure:	/user/{ID}
Description:	Delete the user with this id
Method:	DELETE
Parameters:	**ID** Id of the user to delete.
Response:	**Response<User>**
	Response object containing an empty user.

Update user

Url structure:	/user/{ID}/{oldPassword}
Description:	Update this user with this data
Method:	PUT
Parameters:	**ID** ID of the user to update.
	old-password The users current password.
	SHA-1 hashed.
Content-Type:	**application/json**

User User object with updated user information.

| Response: | **Response<User>** |
| | Response object containing the updated user. |

Get user data

Url structure:	/userData/{userId}
Description:	Get user data for the user with the given user id.
Method:	GET
Parameters:	**userId** Id of the user.
Response:	**Response<User_Data>**
	Response object containing the user data.

Create a new user data

Url structure:	/userData
Description:	Make a new user data.
Method:	POST
Parameters:	None.
Response:	**Response<User_Data>**
	Response object containing the new user data.

Update user data

Url structure:	/userData/{id}
Description:	Updates existing user data.
Method:	PUT
Parameters:	**id** Id of the user data to update.
Response:	**Response<User_Data>**
	Response object containing the updated user data.

Delete user data

Url structure:	/userData/{id}
Description:	Delete user data with this id
Method:	DELETE
Parameters:	**id** Id of the user data to delete.
Response:	**Response<User_Data>**
	Response object containing empty user data.

Get user group by id

Url structure:	/userGroup/{id}
Description:	Gets a user group based on it's id.
Method:	GET
Parameters:	**id** Id of the user group to get.
Response:	**Response<UserGroup>**
	Response object containing the user group.

Create a new user group

Url structure:	/userGroup
Description:	Creates a new user group.
Method:	POST
Parameters:	None.
Response:	**Response<UserGroup>**
	Response object containing the new user group.

Update a user group

Url structure:	/userGroup/{id}
Description:	Updates an existing user group based on it's id.
Method:	PUT
Parameters:	**id** Id of the user group to update.
Response:	**Response<UserGroup>**
	Response object containing the updated user group.

Delete a user group

Url structure:	/userGroup/{id}
Description:	Deletes a user group based on it's id.
Method:	DELETE
Parameters:	**id** Id of the user group to delete.
Response:	**Response<UserGroup>**
	Response object containing an empty user group object.

Get user's groups

Url structure:	/groupOfUser/{userId}
Description:	Gets all groups that a user is member of.
Method:	GET
Parameters:	**userId** Id of the user.
Response:	**Response<UserGroup>**
	Response object containing an array of the user groups.

Insert user into group

Url structure:	/userInGroup/{userId}/{groupId}
Description:	Inserts a user into a user group.
Method:	POST
Parameters:	**userId** Id of the user to insert
	groupId Id of the user group to insert into
Response:	**Response<UserGroup>**
	Response object containing the user group.

Remove user from group

Url structure:	/userInGroup/{userId}/{groupId}
Description:	Deletes a user from a user group.
Method:	DELETE
Parameters:	**userId** Id of the user to delete.
	groupId Id of the group to delete from.
Response:	**Response<UserGroup>**
	Response object containing an empty user group.

1.3   Actions (user actions & user group actions)

TITLE

Url structure:	URL
Description:	DESC
Method:	METHOD
Parameters:	PARAMS.
Response:	RESPONSE

Get an action

Url structure:	/action/{id}
Description:	Gets an action based on it's id
Method:	GET
Parameters:	**id** Id of the action to get.
Response:	**Response\<Action\>** Response object containing the action.

Get all actions

Url structure:	/action
Description:	Gets all actions.
Method:	GET
Parameters:	None.
Response:	Response object containing an array of action objects.

Get user's actions

Url structure:	/userActions/{userId}
Description:	Gets all actions available for a specific user.
Method:	GET
Parameters:	**userId** The id of the user.
Response:	Response message containing an array of action objects.

Give action to user

Url structure:	/userActions/{userId}/{actionId}/{contentId}/{allow}
Description:	Assign an action to a user.
Method:	POST
Parameters:	**userId** The id of the user
	actionId The id of the action.
	contentId Id of the content the action allows.
	allow True if action is allowed, false otherwise.
Response:	Response object containing the new action object.

Remove action from user

Url structure:	/userActions/{userId}/{actionId}

Description:	Removes a specific action from a specific user.
Method:	DELETE
Parameters:	**userId** Id of the user
	actionId Id of the action to remove.
Response:	Response object containing an empty action object.

Get user group actions

Url structure:	/groupActions/{groupId}
Description:	Get all actions available for a specific user group.
Method:	GET
Parameters:	**groupId** The id of the user group.
Response:	Response object containing an array of action objects.

Post group action

Url structure:	/groupActions/{groupId}/{actionId}/{contentId}/{allow}
Description:	Posts an action to a user group.
Method:	POST
Parameters:	**groupId** Id of the user group.
	actionId Id of the action to assign to the group.
	contentId Id of the content the action allows.
	allow True if the action is allowed, false otherwise.
Response:	Response object containing the new action object.

Remove action from group

Url structure:	/groupActions/{groupId}/{actionId}
Description:	Removes a specific action from a specific user group.
Method:	DELETE
Parameters:	**groupId** The id of the user group to remove from.
	actionId The id of the action to remove.
Response:	Response object containing an empty action object.

1.4 Media (Media & Media Category)

Table 4: Media objects

Field	Description
id	A unique id of the media
media_category	The id of the media's category
media_category_name	The name of the media's category
user	The id of the user who uploaded
file_location	The location of the connected file
title	The title of the media
description	The description of the media
media_length	The length of the media in minutes
format	The format of the file
tags	A list of tags connected to the media

MediaCategory

Field	Description
id	A unique id
name	The name of the media category

Get media with specific id

Url structure:	/media/{ID}
Description:	Get a specific media, based on it's id
Method:	GET
Parameters:	**id** Id of the media to get.
Response:	Response object containing the media object.

Get all medias (with parameters)

Url structure:	/media?tag={tag}&mediaCategoryFilter={mediaCategoryFilter}& nameFilter={nameFilter}&page={page}&limit={limit}
Description:	Get all media matching the giver criteria.
Method:	GET
Parameters:	**tag** Tag connected to the media.
	mediaCategoryFilter Media category that the media might be member of.

	nameFilter Filter media by name
	page Page offset. Default is 1 (no offset).
	limit How many results to return.
Response:	Response object containing an array of media objects along with meta information such as page offset and limit.

Create a new media

Url structure:	/media
Description:	Create a new media and get a path for your upload. This will only create an entry in the database with the meta data provided.
Method:	POST
Parameters:	None
Content-Type	**application/json**
	Media Media object containing all information.
Response:	Response object containing the newly inserted media object.

Upload a media file associated with a media

Url structure:	/mediaFiles/{ID}
Description:	Upload a media file. You give the ID connected to the posted meta data and the file you want to upload.
Method:	POST
Parameters:	**ID** The id of the media that belongs to the file.
Content-Type	**File Stream** The file to upload
Response:	Response message

Update media

Url structure:	/media/{ID}
Description:	Update the metadata of a media.
Method:	PUT
Parameters:	**ID** Id of the media to update.
Content-Type	**application/json**
	Media Media object containing the updated media information.
Response:	Response object containing the updated media object.

Delete media

Url structure:	/media/{ID}
Description:	Delete a media. This will also delete the file connected to the media.
Method:	DELETE
Parameters:	**ID** Id of the media to delete.
Response:	Response object containing an empty media object.

Get all media categories

Url structure:	/mediaCategory
Description:	Get a list of all media categories
Method:	GET
Parameters:	None
Response:	Response object containing a list of all media categories.

Get specific media category

Url structure:	/mediaCategory/{ID}
Description:	Get a media specific category
Method:	GET
Parameters:	**ID** Id of the media category to get.
Response:	Response object containing the media category object.

New media category

Url structure:	/mediaCategory
Description:	Creates a new media category
Method:	POST
Parameters:	None
Content-Type	**application/json**
	MediaCategory MediaCategory object containing all related information.
Response:	Response object containing the new media category object.

Update media category

Url structure:	/mediaCategory/{ID}
Description:	Update media category
Method:	PUT
Parameters:	**ID** Id of the media category to update.
Content-Type	**application/json**
	MediaCategory Media category object containing information of the updated n
Response:	Response object containing the newly updated media category object.

Delete media category

Url structure:	/mediaCategory/{ID}
Description:	Delete a media category
Method:	DELETE
Parameters:	**ID** Id of the media category to delete.
Response:	Response object containing a empty media category object.

1.5   Tags

Table 5: Tag objects

Field	Description
id	A unique id
name	The name of the tag
simple_name	The short version of the name
tag-group	Tag group

Table 6: Tag group objects

Field	Description
id	A unique id
name	The name of the tag group
description	The tag group description

Get all tags

Url structure:	/tags?tagGroupFilter={tagGroupFilter}& limit={limit}&page={page}
Description:	Get a list of all tags
Method:	GET
Parameters:	**tagGroupFilter** The id of the tag group you want to filter by
	limit Amount of tags per page. Default = 10
	page Page offset. Default = 1 (no offset)
Response:	Response object containing an array of tag objects.

Get single tag

Url structure:	/tags/{ID}
Description:	Get a single tag based on it's id.
Method:	GET
Parameters:	**ID** Id of the tag.
Response:	Response object containing the selected tag object.

New tag

Url structure:	/tags
Description:	Create a new tag
Method:	POST
Parameters:	None
Content-Type	**application/json**
	Tag Tag object with the related information.
Response:	Response object containing the new tag object.

Update tag

Url structure:	/tags/{ID}
Description:	Update a tag
Method:	PUT
Parameters:	**ID** Id of the tag to update.
Content-Type	**application/json**
	Tag Tag object containing the updated tag.
Response:	Response object containing the newly updated tag object.

Delete tag

Url structure:	/tags/{ID}
Description:	Delete a tag
Method:	DELETE
Parameters:	**ID** Id of the tag to delete.
Response:	Response object containing an empty tag object.

Get all tag groups (with parameters)

Url structure:	/tagGroups?limit={limit}&page={page}
Description:	Get a list of all tag groups.
Method:	GET
Parameters:	**limit** How many results to return (default = 10)
	page Page offset (default = 1, no offset)
Response:	Response object containing an array of tag group objects.

Get single tag group

Url structure:	/tagGroups/{ID}
Description:	Get a single tag group
Method:	GET
Parameters:	**ID** Id of the tag group to get.
Response:	Response object containing the tag group object.

New tag group

Url structure:	/tagGroups
Description:	Create a new tag group
Method:	POST
Parameters:	None
Content-Type	**application/json**
	TagGroup TagGroup object containing all relevant information.
Response:	Response object containing the new tag group object.

Update tag group

Url structure:	/tagGroups/{ID}

Description:	Update a tag group
Method:	PUT
Parameters:	**ID** Id of the tag group to edit.
Content-Type	**application/json**
	TagGroup TagGroup object with the new information.
Response:	Response object containing the newly updated tagGroup object.

Delete tag group

Url structure:	/tagGroups/{ID}
Description:	Delete a tag group (this will also delete tags connected to the tag group, or delete the connection)
Method:	DELETE
Parameters:	**ID** Id of the tag group to delete
Response:	Response object containing a empty tag group object.

Get tags belonging to a media

Url structure:	/tagsByMedia/{mediaId}
Description:	Gets all tags related to a specific media.
Method:	GET
Parameters:	**mediaId** The id of the media to filter by.
Response:	Response object containing a list of tags.

Post tag to media

Url structure:	/tagsByMedia/{mediaId}/{tagId}
Description:	Connects a tag to a media.
Method:	POST
Parameters:	**mediaId** The id of the media to connect to.
	tagId Id of the tag to connect.
Response:	Response object containing the media connected to the tag.

Get single user tags

Url structure:	/userTags/{id}
Description:	Gets all tags connected to a user.

Method:	GET
Parameters:	**id** Id of the user.
Response:	Response object containing a list of UserAccountTag objects.

Get all user tags

Url structure:	/userTags
Description:	Gets all user tags available in the system.
Method:	GET
Parameters:	None.
Response:	Response object containing a list of UserAccountTag objects.

Create new user tag

Url structure:	/userTags
Description:	Posts a new user tag.
Method:	POST
Parameters:	None.
Content-type:	**application/json**
	UserAccountTag UserAccountTag object containing all relevant information.
Response:	Response object containing the new UserAccountTag object.

Delete user tag

Url structure:	/userTags/{ID}
Description:	Deletes a specific user tag.
Method:	DELETE
Parameters:	**ID** Id of the user tag to delete..
Response:	Response object containing a empty UserAccountTag object.

Get media by user tag

Url structure:	/mediaByUserTag/{userId}/{userTagId}
Description:	Gets media based on user and usertag.
Method:	GET
Parameters:	**userId** The id of the user.
	userTagId Id of the user tag connected to the media
Response:	Response object containing a list of Media objects.

Get user tags by user

Url structure:	/userTagsByUser/{userId}
Description:	Returns all userTags that belong to a specific user.
Method:	GET
Parameters:	**userId** Id of the user.
Response:	Response object containing a list of all UserAccountTag objects.

Post user tag to media

Url structure:	/userAccountTag/{userId}/{mediaId}/{tagId}
Description:	Connects a user tag to a media.
Method:	POST
Parameters:	**userId** Id of the user that posts the tag.
	mediaId Id of the media.
	tagId Id of the user tag to connect to the media.
Response:	Response object containing the newly posted user tag.

Remove media from tag

Url structure:	/mediaByUserTag/{mediaId}/{tagId}
Description:	Removes a media from a user account tag.
Method:	DELETE
Parameters:	**mediaId** Id of the media to remove.
	tagId Id of the user tag to remove from.
Response:	Response object containing a empty user tag object.

1.6 Rating

Get all rating for specific media

Url structure:	/rating?media={media}&user={user}& limit={limit}&page={page}
Description:	Returns all the ratings / comments on a specific media.
Method:	GET
Parameters:	**media** Id of the media .
	user Id of the user that posted the rating.
	limit How many results to return (default = 10)
	page Page offset. (default = 1, no offset)
Response:	Response object containing a list of ratings.

New rating for media

Url structure:	/rating
Description:	Posts a new rating for a media
Method:	POST
Parameters:	None.
Content-Type:	**application/json**
	Rating Rating object containing all relevant information.
Response:	Response object containing the newly posted rating.

Edit rating

Url structure:	/rating{ID}
Description:	Edits an already existing comment.
Method:	PUT
Parameters:	**id** Id of the rating to edit
Content-Type:	**application/json**
	Rating Rating object containing the updated rating.
Response:	Response object containing the updated rating object.

Delete rating

Url structure:	/rating{ID}
Description:	Delete a rating
Method:	DELETE
Parameters:	**id** Id of the rating to delete
Response:	Response object containing an empty rating object.