

CINEMA DRIVE - API V1 DOCUMENTATION

General	3
Authorization Token	3
Response Status Codes	3
Successful Response Layout.....	3
Error Response Layout	4
Authentication - Login & Logout.....	4
[POST] /api/sessions – Login via Email & Password	4
[POST] /api/auth/facebook/callback – Login via Facebook	5
[DELETE] /api/sessions – Logout.....	5
Account Registration & Data	5
[POST] /api/registrations – Register a new user account with email & password	5
[GET] /api/registrations/current – Get data about the current logged in user account	6
[POST] /api/auth/facebook/callback – Register a new user account via Facebook	6
[POST] /api/registration_codes/verify – Check validity of a registration code.....	7
Password Reset	7
[POST] /api/password – Request password reset	7
[PUT] /api/password – Change password using reset token	7
Interactive Videos	8
[GET] /api/interactive_videos/ - get a list of available videos	8
[GET] /api/interactive_videos/:id – get a specific video by its ID	9
[POST] /api/interactive_videos/:id/done – indicate that the user finished watching a video	9
Surveys.....	9
[GET] /api/surveys/:id – get a specific survey by its ID	9
[POST] /api/surveys/:id/answers – post answers for a survey	10
Scores.....	11
[GET] /api/scores – get the score of the current user and its group	11
Score updates	12

Campaigns.....	12
[GET] /api/campaigns/current – get the current campaign	12
[GET] /api/campaigns/:id/click – indicate a click on the campaign with the given ID	12
Invites	12
[POST] /api/invites – invite another person to the site	13
Interactive Video Choices	13

General

Authorization Token

All API calls require an authorization HTTP header with a valid token (retrieved by authenticating with the API). The HTTP header should have the following format:

```
Authorization: Token token="<valid-access-token>"
```

Each use of a token, postpones its expiry by 1 additional hour.

Response Status Codes

The API responds with a different HTTP status code to indicate the status of an issued request.

A successful request may be responded with the following status codes:

- **200** - OK
- **201** - Record Created

A failed request may be responded with the following status codes:

- **400** - Bad Request
- **401** - Unauthorized
- **403** - Forbidden
- **404** - Not Found
- **422** - Unprocessable Entity
- **500** - Internal Server Error

To be sure, any request that is responded with a status code which is not 200 or 201 should be considered as failed.

Successful Response Layout

The body of all responses will be formatted as JSON.

A successful response has the following parts:

- **data** – a JSON object with the response specific to the issued request (exists only when not empty)
- **score** – a JSON object with information about a user's score update occurred by the last request (exists only when not empty)

Example:

```
{
  "data": {
    "access_token": "1234567890abcdefghijklmnopqrstuvwxyz"
  },
  "score": {
    "current": 3000,
    "updates": [
      {
        "action": "survey_answer",
        "points": 1000
      }
    ]
  }
}
```

```
    ]
  }
}
```

Note that below when writing that a certain request returns an empty response, that means that the "data" part is empty or missing, but the "score" part may still be present.

Error Response Layout

A failed response has the following parts:

- **error_code** - a string key identifying the error type
- **message** - a human readable text identifying the error type
- **data** - a JSON object with data about the specific error (exists only when not empty)

Example:

```
{
  "error_code": "missing_parameter",
  "message": "Missing Parameter",
  "data": {
    "email": "parameter is required"
  }
}
```

Possible error codes are:

- **not_authenticated** – a valid authorization token header is missing
- **not_found** – the requested URL was not found
- **access_denied** – access to the requested URL is not allowed to the current user
- **incomplete_registration** -user's registration is incomplete
- **unprocessable_entity** - may have specific details under the "data" object
- **missing_parameter** - will have specific details under the "data" object as in the example above

In addition, each API action may have its own specific error codes.

Authentication - Login & Logout

[POST] /api/sessions – Login via Email & Password

Parameters:

- **email** – a valid user's email
- **password** – the user's password

Response:

An access token that should be used with all further requests (see "General – Authorization Token" Authorization Token)

```
{
  "data": {
    "access_token": "1234567890abcdefghijklmnopqrstuvwxyz"
  }
}
```

```
}
```

Possible specific errors keys:

- **invalid_login_credentials**

[POST] /api/auth/facebook/callback – Login via Facebook

Parameters:

- **auth_response** – the "authResponse" object received as part of a successful login with Facebook login on the client side.

Response:

An access token that should be used with all further requests (see "General – Authorization Token" Authorization Token)

```
{
  "data": {
    "access_token": "1234567890abcdefghijklmnop"
  }
}
```

Possible specific errors keys:

- **invalid_facebook_authentication** – the given "auth_response" parameter is invalid
- **missing_facebook_email** – client side login to facebook must use the "email" scope to allow the server access to the user's email on facebook.

[DELETE] /api/sessions – Logout

Parameters:

None other than the authorization token in the header.

Response:

Empty data

Account Registration & Data

[POST] /api/registrations – Register a new user account with email & password

Parameters:

- **email** – user's email
- **password** – user's password
- **first_name** – user's first name
- **last_name** – user's last name
- **birth_date** – user's birth date
- **gender** – male / female
- **age** – user's age
- **reg_code** – a valid group registration code

- **tos_accepted** – "true" if user accepted the terms of service

Response:

The created user's details:Authorization Token

```
{
  data: {
    email: "test@test.com",
    first_name: "aaa",
    last_name: "bbb",
    birth_date: null,
    gender: "male",
    age: "16",
    tos_accepted: true,
    group_name: "My Group"
  }
}
```

Possible specific errors keys:

- **unprocessable_entity** – will have a detailed explanation of missing parameters, for example:

```
{
  error_code: "unprocessable_entity"
  message: "Unprocessable Entity"
  data: {
    first_name: [
      "can't be blank"
    ],
    last_name: [
      "can't be blank"
    ],
    tos_accepted: [
      "must be accepted"
    ],
    reg_code: [
      "is invalid"
    ],
    email: [
      "can't be blank"
    ],
    password: [
      "is too short (minimum is 8 characters)"
    ]
  }
}
```

[GET] /api/registrations/current – Get data about the current logged in user account

Response is the same after registration (see: "[POST] /api/registrations" above).

[POST] /api/auth/facebook/callback – Register a new user account via Facebook

Parameters:

- **auth_response** – the "authResponse" object received as part of a successful facebook login (on the client side)
- **reg_code** – a valid group registration code
- **tos_accepted** – should have a value of "true" to indicate that the user has accepted our terms of service

Response:

An access token that should be used with all further requests (see "General – Authorization Token" Authorization Token)

```
{
  "data": {
    "access_token": "1234567890abcdefghijklmnopqrstuvwxyz"
  }
}
```

Possible specific errors keys:

- **invalid_facebook_authentication** – the given "auth_response" parameter is invalid
- **missing_facebook_email** – client side login to facebook must use the "email" scope to allow the server access to the user's email on facebook.

[POST] /api/registration_codes/verify – Check validity of a registration code

Parameters:

- **reg_code** – a group registration code

Response:

An empty response (with status code 200) will be returned on success:

```
{}
```

Possible specific errors keys:

- **unprocessable_entity** – when the given code is not valid, for example:

```
{
  "error_code": "unprocessable_entity",
  "message": "Unprocessable Entity",
  "data": {
    "reg_code": "is invalid"
  }
}
```

Password Reset

[POST] /api/password – Request password reset

Parameters:

- **email** – the user's email

Response:

The response body will be empty, but an email with a password reset token will be sent to the user.

[PUT] /api/password – Change password using reset token

Parameters:

- **reset_password_token** – the token received by email
- **password** – the new password

Response:

Empty data

Interactive Videos

[GET] /api/interactive_videos/ - get a list of available videos

Parameters:

None

Response:

A list of videos, with their different attributes and pre/post surveys.

For example, a response with 2 videos, one that has a pre & post surveys and another that has none:

```
{
  "data": [
    {
      "id": 1,
      "name": "Interactive video 1",
      "description": "This is the first interactive video",
      "content": "<xml-content-for-flash-player>",
      "content_mobile": "<xml-content-for-mobile-player>",
      "surveys": {
        "pre_survey": {
          "id": 1,
          "url": "http://localhost:3000/api/surveys/1"
        },
        "post_survey": {
          "id": 2,
          "url": "http://localhost:3000/api/surveys/2"
        }
      },
      "enabled": true,
      "watched": true,
      "url": "http://localhost:3000/api/interactive_videos/1"
    },
    {
      "id": 3,
      "name": "Interactive video 3",
      "description": "",

```



```

        "enabled": false,
        "watched": false,
        "url": "http://localhost:3000/api/interactive_videos/3"
      }
    ]
  }
}

```

Note the following important attributes:

- **enabled** – indicates whether the video is currently enabled for the user.
- **watched** – indicates whether the video was previously watched by the user.
- **url** – for a video or survey, give the API URL for its details.
- **content / content_mobile** – will only be included if the video is currently enabled for the user.

[GET] /api/interactive_videos/:id – get a specific video by its ID

Parameters:

- **id** – part of the URL – the ID of the required interactive video

Response:

```

{
  "data": {
    "id": 2,
    "name": "Interactive video 2",
    "description": "This is the second video",
    "content": "<xml-content-for-flash-player>",
    "content_mobile": "<xml-content-for-mobile-player>",
  }
}

```

Possible specific errors keys:

- **missing_pre_survey** – a survey has to be answered before getting access to the interactive video.

[POST] /api/interactive_videos/:id/done – indicate that the user finished watching a video

Parameters:

- **id** – part of the URL – the ID of the required interactive video

Response:

Empty data

Possible specific errors keys:

- **missing_post_survey** – a survey has to be answered before setting the video as watched.

Surveys

[GET] /api/surveys/:id – get a specific survey by its ID

Parameters:

- **id** – part of the URL – the ID of the required survey

Response:

The required survey that includes a list of questions and their answers.

```
{
  "data": {
    "id": 1,
    "name": "A sample survey",
    "questions": [
      {
        "id": 6,
        "number": 1,
        "name": "What's up",
        "question": "What's up ?",
        "answers": [
          {
            "number": 1,
            "answer": "good"
          },
          {
            "number": 2,
            "answer": "bad"
          }
        ]
      },
      {
        "id": 3,
        "number": 2,
        "name": "Day of the week",
        "question": "What day is it today ?",
        "answers": [
          {
            "number": 1,
            "answer": "Sunday"
          },
          {
            "number": 2,
            "answer": "Monday"
          },
          {
            "number": 3,
            "answer": "Tuesday"
          }
        ]
      }
    ]
  }
}
```

Note the following important attributes:

- Each **question** has an **ID** that should be used when posting answers for the survey.
- Each **question** has a **number** that should be used to order the questions in the survey.
- Each **answer** has a **number** that should be used when posting answers for the survey.

[POST] /api/surveys/:id/answers – post answers for a survey

Parameters:

- **id** – part of the URL – the ID of the required survey
- **answers** – an array of answers, each with a question ID and answer number, for example:

```
{
  "answers": [
    {"question_id": "1", "answer_number": "2"},
    {"question_id": "2", "answer_number": "4"}
  ]
}
```

Response:

Empty data

Possible specific errors keys:

- **unprocessable_entity** – will have a detailed explanation of missing parameters, for example:

```
{
  "error_code": "unprocessable_entity",
  "message": "Unprocessable Entity",
  "data": [
    {
      "question_id": 55,
      "errors": [
        "Question is invalid"
      ]
    },
    {
      "question_id": 2,
      "errors": [
        "Answer number is invalid",
        "Question is invalid"
      ]
    },
    {
      "question_id": 1,
      "errors": [
        "Question is missing"
      ]
    }
  ]
}
```

The data object is an array, in which each objects includes a question_id parameter, and an "errors" array of errors regarding that question.

Scores

[GET] /api/scores – get the score of the current user and its group

Parameters:

None

Response:

The current score of the user and its group:

```
{
  "data": {
    "score": 2000,
    "group_score": 64887
  }
}
```

Possible specific errors keys:

None

Score updates

As already stated, some API requests give additional score points to the user. In that case, the returned response will include a "score" part that looks like this:

```
{
  "score": {
    "current": 3000,
    "updates": [
      {
        "action": "survey_answer",
        "points": 1000
      }
    ]
  }
}
```

The following attributes should be explained:

- **current** – the current updated score of the user
- **updates** – an array of actions that caused a gain of points, and the amount of points that was added for that action.

The above example shows that user has a score of 3000 and that in the last request, the user gained 1000 points for answering a survey.

Campaigns

[GET] /api/campaigns/current – get the current campaign

Not yet implemented

[GET] /api/campaigns/:id/click – indicate a click on the campaign with the given ID

Not yet implemented

Invites

[POST] /api/invites – invite another person to the site

Parameters:

- **email** – the email of the person to invite
- **message** – a message to send as the invitation's email body

Response:

Empty data, but an invitation email will be sent to the required person.

Interactive Video Choices

Not yet implemented