

# API Reference

---



2023, © Approval Studio, v. 1.23

Please find the most recent **API Reference** online [here](#).

## Table of contents

---

- [API Reference](#)
- [Table of contents](#)
- [General](#)
- [Approval Studio basic concepts](#)
  - [REST HTTP Codes](#)
- [Authorization/authentication](#)
- [Token management](#)
  - [POST /api/v1/token/login](#)
  - [GET /api/v1/token/validate](#)
- [Project management](#)

- [GET /api/v1/project](#)
- [POST /api/v1/project](#)
- [PUT /api/v1/project](#)
- [DELETE /api/v1/project](#)
- [GET /api/v1/project/proofreport](#)
- [PUT /api/v1/project/state](#)
- [GET /api/v1/project/meta](#)
- [POST /api/v1/project/meta](#)
- [PUT /api/v1/project/meta](#)
- [DELETE /api/v1/project/meta](#)
- [GET /api/v1/project/meta/projects](#)
- [POST /api/v1/kanban/column](#)
- [Asset management](#)
  - [GET /api/v1/asset](#)
  - [DELETE /api/v1/asset](#)
  - [POST /api/v1/asset/upload](#)
  - [POST /api/v1/asset/upload\\_url](#)
  - [GET /api/v1/asset/download](#)
  - [GET /api/v1/asset/proofreport](#)
- [Reference documents management](#)
  - [POST /api/v1/refdoc/upload](#)
  - [POST /api/v1/refdoc/upload\\_url](#)
  - [POST /api/v1/refdoc/add\\_url](#)
  - [GET /api/v1/refdoc/download](#)
  - [DELETE /api/v1/refdoc](#)
- [Task management](#)
  - [GET /api/v1/task/all](#)
  - [GET /api/v1/task](#)
  - [DELETE /api/v1/task](#)

- [POST /api/v1/task/asset\\_upload](#)
- [POST /api/v1/task/refdoc\\_upload](#)
- [POST /api/v1/task/review\\_asset](#)
- [POST /api/v1/task/review\\_asset\\_ext](#)
- [PUT /api/v1/task/complete](#)
- [Annotations management](#)
  - [GET /api/v1/annotation/all](#)
  - [GET /api/v1/annotation](#)
  - [DELETE /api/v1/annotation](#)
  - [PUT /api/v1/annotation/hide](#)
  - [PUT /api/v1/annotation/complete](#)
  - [PUT /api/v1/annotation/uncomplete](#)
- [Users management](#)
  - [GET /api/v1/users](#)
  - [GET /api/v1/user/loggedin](#)
- [Webhooks](#)
  - [Security](#)
  - [Retry logic](#)
  - [Event object](#)
- [Webhooks management](#)
  - [GET /api/v1/webhooks](#)
  - [POST /api/v1/webhook](#)
  - [DELETE /api/v1/webhook](#)
  - [PUT /api/v1/webhook/test](#)

## General

---

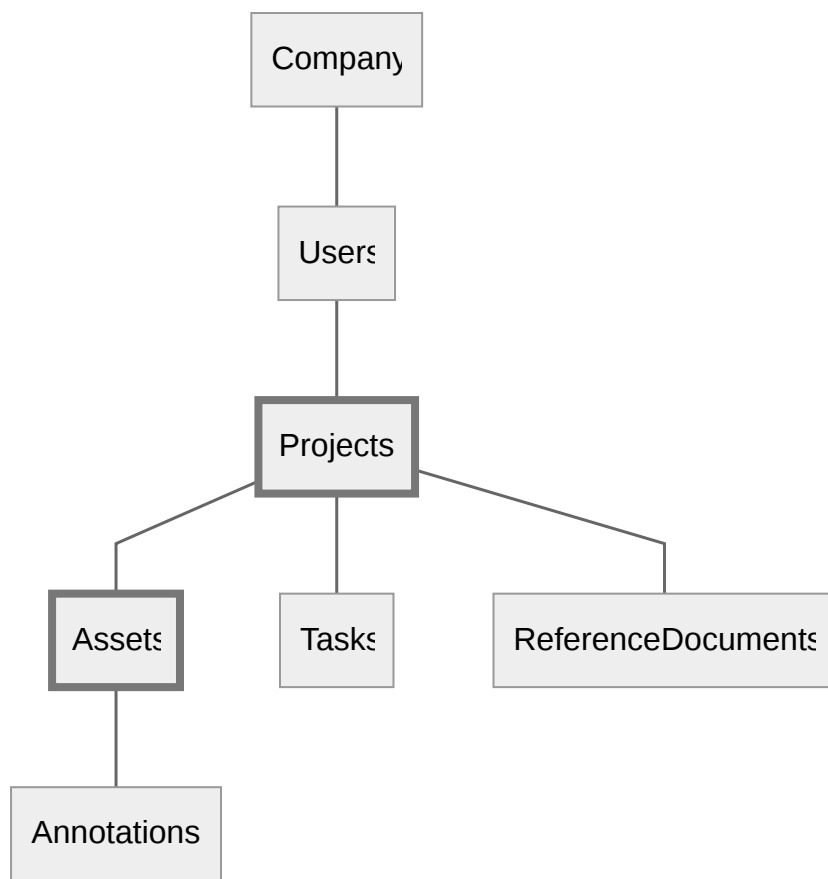
This is a REST API to the [Approval Studio](#), design review, and packaging approval SAAS. API utilizes its flow, authorization, storage, and so on.

## Approval Studio basic concepts

---

From the business point of view, Approval Studio is based on a multitenant concept when a single user may be a part of one or more tenants (called Companies). Please see <https://approvalstudio.freshdesk.com/> for detailed instructions on how to work with Approval Studio.

Currently the only point where you may choose a tenant is project creation, please see [POST /api/v1/project](#) / [Request](#).



## REST HTTP Codes

---

API may return one of the following HTTP codes:

HTTP Code	Response
200	<b>Success.</b> See the method description to get what and how the method returns.

**400**

**Validation error.** API validates input parameters and when finds an invalid parameter, throws HTTP code 400. Those errors are related to parameters' presence, format, emptiness, etc. Validating against a database, like looking for data by an ID is conducted separately and reflected in case of error, in HTTP codes 404, 406, 412, etc., please see the method's description for detailed explanations.

```
{
  "errors": {
    "parameterName": [
      "'parameterName' must not be empty."
    ]
  },
  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
  "title": "One or more validation errors occurred.",
  "status": 400,
  "traceId": "|a4a45224-4a7c03bbd5172369."
}
```

HTTP Code	Response
404, 406, 412	<b>Resource not found.</b> Generally, it means that requested resources are either not found or somehow restricted to proceed. For example, if a given project id is non-existent, the project is not found so gets HTTP code 404.
429	<b>Rate Limit Reached.</b> API host calculates the number of calls per sec, minute, and hour, and when the rate of requests reaches the limit, throws HTTP code 429, which means that the API host is overloaded and the client needs to wait before retrying.

If the request gets blocked then the client receives a text response like this:

Status Code: 429  
Retry-After: 58  
Content: API calls quota exceeded! maximum admitted 2 per 1m.

Retry-After header value is expressed in seconds. And X-Rate-Limit-XXX HTTP headers are injected in the response:

X-Rate-Limit-Limit: the rate limit period (eg. 1m, 12h, 1d)  
X-Rate-Limit-Remaining: number of requests remaining  
X-Rate-Limit-Reset: UTC date-time (ISO 8601) when the limits reset

HTTP Code	Response
500	<b>Infrastructure failure.</b> This means critical unrecoverable technical error generally related to database connections, external services availability, hardware failure, etc. Depending on the client application flow you can retry the calling method or halt processing and call our technical support.

## Authorization/authentication

The API is based on an **authorization token** which should be requested prior to using any available API methods.

The flow is the following: request auth token by calling **POST/api/v1/token/login** (see) providing Approval Studio's username/password. If ok, the method returns an authorization token that you should **provide with any other API call** as an **HTTP header** like this: Authorization: Bearer XXXXXXXXXXXX .

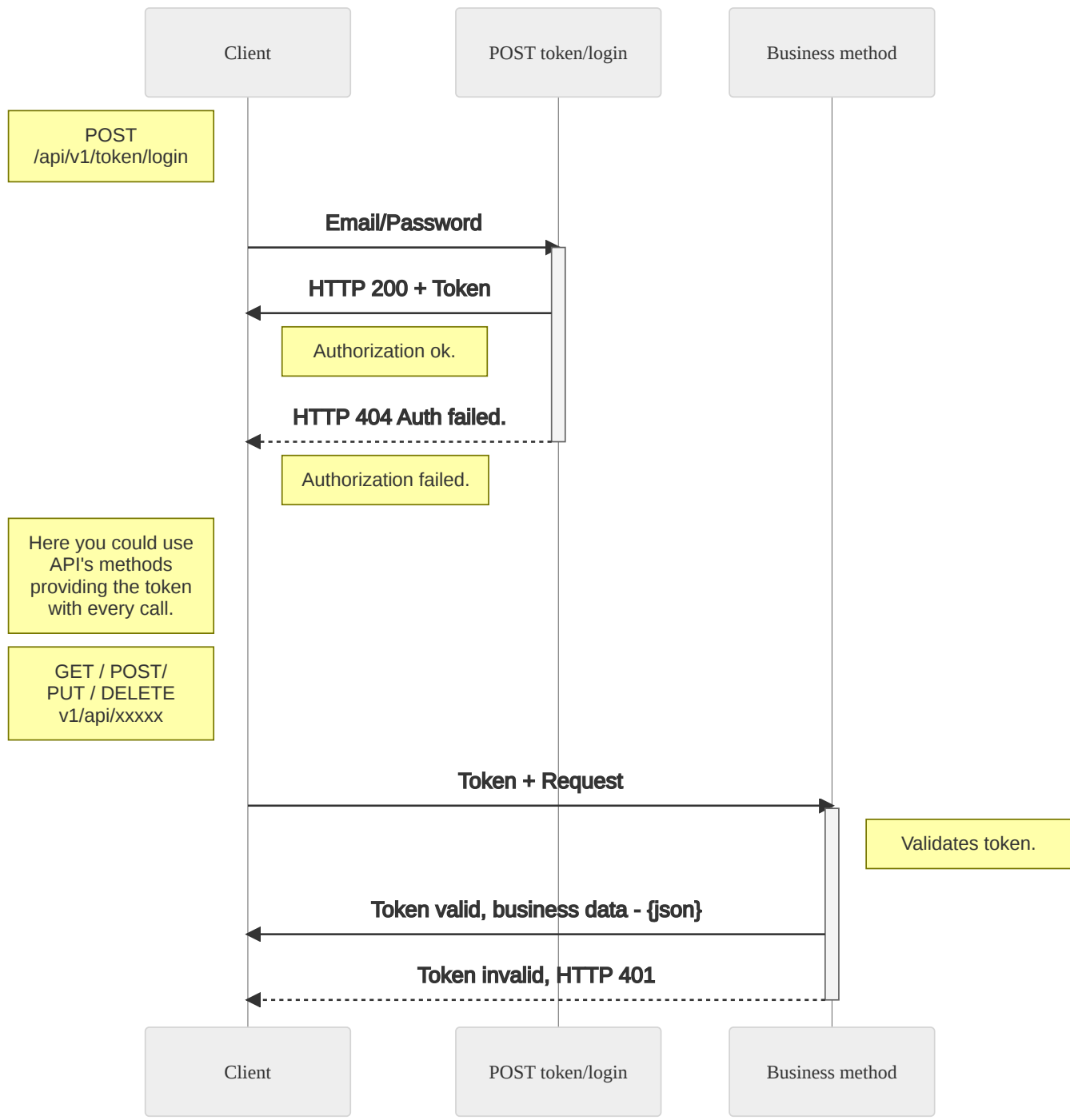
The token is valid for a limited amount of time, **600 minutes** by default. When the token is expired and still used you will have the response `HTTP code 401, Unauthorized` :

```
{
  "isError": true,
  "type": "https://httpstatuses.com/401",
  "title": "Unauthorized",
  "status": 401,
  "instance": "/api/v1/annotation"
}
```

If so, you need to **obtain a new token**.

If the token is invalid or not provided, it gets the same response **HTTP code 401**.





# Token management

---

## POST /api/v1/token/login

---

**Authorizes a user.** When successfully authorized it returns an **authorization token** which must be supplied to every API as a header, like `"Authorization: Bearer YYYYYYYYYYYYYYYYYY..."`.

### Request

```
{
  "userName": "john.smith@gmail.com",
  "password": "Q@fG%^18_A",
  "keepAliveTime": 12                // Optional, minutes.
}
```

`keepAliveTime` is an optional parameter that defines how much time the token will be valid. Response's `expirationDate` provides an exact time of the token expiration (see below).

**Note:** When `keepAliveTime` is zero, the default value is used (600 min/10 hours).

### Curl:

```
curl -X POST "https://api.approval.studio/api/v1/token/login" \
  -H "accept: text/plain" \
  -H "Content-Type: application/json-patch+json" \
  -d '{"userName":"joshn.smith@pepsico.com","password":"Q@fG%^18_A~", "keepAliv
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Login successful, AUTH token provided.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "POST Request successful.",
  "result": {
    "token": "eyJhbGciOiJIUzI1NiIsIjkiOiJ1eSIsImVudCI6IkpvaW50b3RlbnQifQ==", // This is the auth token.
    "status": "Success",
    "expirationDate": "2022-02-15T18:42:15.8398451Z", // A date/time when the token expires
    "user": {
      "userID": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
      "fullName": "John Smith",
      "email": "john.smith@gmail.com",
      "role": "RegularUser|Administrator"
    }
  }
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed. See <b>HTTP code 400 description</b> .
404	<b>Error:</b> User with given email and password not found. Either the provided credentials are wrong or the user is locked and not able to login anymore.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Authorization failed.",
  "result": {
    "status": "WrongCredentials"
  }
}
```

HTTP Code	Response
412	<b>Error:</b> The user uses <b>Lite</b> payment plan option. API is available only for <b>Pro</b> users. Please consult our support on this: <a href="https://approval.studio/contact/">https://approval.studio/contact/</a>

## GET /api/v1/token/validate

**Validates an auth token.** This method is an kind of a health checker, that validates an authentication token, if it's still valid. You may use it to ensure that the your authentication process completed successfully or that the token is still valid in case of a long-running scenarios like external workflow engines where tokens are saved for further use.

### Request

#### Curl:

```
curl -X 'GET' \
  'http://localhost:8000/api/v1/token/validate' \
```

```
-H "accept: text/plain" \  
-H "Content-Type: application/json-patch+json" \
```

## Responses

HTTP Code	Response
200	<b>Success.</b> authentication token provided in the <b>Auth</b> header is valid.

```
{  
  "version": "1.0",  
  "statusCode": 200,  
  "message": "POST Request successful.",  
  "result": {}  
}
```

HTTP Code	Response
401	<b>Error:</b> Parameters' validation failed.

## Project management

Project attributes:

Name		Explanation
projectUID	r/o	System-wide unique project identifier, GUID

projectState		Projects are in one of the following states: Active, OnHold, Completed, InTransit, Archived .
name		Free-form project name, string, max-length 50 chars, mandatory.
customer		Project's customer name, max-length 50 chars, optional.
project		Business project or product name this Approval Studio project related for, max-length 50 chars, optional.
design		Point to which type of design this is, max-length 50 chars, optional.
revision		Ideally, a sequential number or version number; generally a free form string max-length 50 chars, optional.
description		Any kind of additional information, descriptions, comment etc., max-length 200 chars, optional.
tags		Array of free-form single-word tags associated with the project, optional, max 20 elements of 20-chars tags.
dueDate		Optional date, UTC, that points to when the project is desired to be completed.
reviewStatus	r/o	pendingCount - count of non-completed review tasks assigned on the project
		approvedCount - count of approves made on the project's assets
		rejectedCount - count of rejected made on the project's assets
created	r/o	Refers to when the project was created, UTC.
templateUID		It is <b>a new concept</b> , introduced in API v.1.15. <b>Project template</b> is a set of some data, usually a JSON, that administrator can

provide when creating a new project. That could be a list of users, emails, etc any other business-specific attributes that could be used in building a business flow and should be specific for every project.

For example, you might need to create a review task just after the project creation and you use API to implement this. In that case, you may provide UID of a user to create a review task for. The template itself doesn't make any changes in the default application flow, it's just a customizable attribute.

Method/Path	Description
GET /api/v1/project	Gets a list of projects.
POST /api/v1/project	Creates new project.
PUT /api/v1/project	Edits existing project.
DELETE /api/v1/project	Deletes a project.
GET /api/v1/project/proofreport	Gets a link to a proof report for a project.
PUT /api/v1/project/state	Changes project's state.

## GET /api/v1/project

Returns one or more projects and projects' assets, tasks, and reference documents – depending on the parameters passed.

All the parameters are optional; the method always uses **AND** combination of all the parameters.

Parameter	Type & Explanation
-----------	--------------------

ProjectUID	<p><code>string</code> Unique project GUID. If provided, only one project will be retrieved.</p> <p><b>Note:</b> behavior is changed in v.1.15.</p> <p>Now if <code>ProjectUID</code> is provided and this project belongs to an appropriate client, you will get project data regardless of the owner list and whether you are an admin or not. In other words, if this project belongs to your client (tenant), you will get it.</p>
States	<p><code>string</code> One or more project states, comma-separated, see <i>ProjectStates</i> above. Default value is <code>Active,OnHold,Completed</code>.</p>
Query	<p><code>string</code> A free-form text to case-sensitive search in projects' attributes: <code>Project name</code>, <code>Customer</code>, <code>Project</code>, <code>Design</code>, <code>Revision</code>, <code>Description</code>, <code>Tags</code>.</p>
IsLoadAssets	<p><code>boolean</code>, <code>false</code> by default. If it is set, returns assets for every project as a child collection.</p>
IsLoadLastVerAssets	<p><code>boolean</code>, <code>true</code> by default. If set, only the most recent version of every asset be returned. Ignored, if <code>IsLoadAssets</code> is not set.</p>
IsLoadTasks	<p><code>boolean</code>, <code>false</code> by default. If set, a list of active tasks for every project would be returned as a child collection.</p>
IsLoadRefDocs	<p><code>boolean</code>, <code>false</code> by default. If set, a list of uploaded reference documents for every project would be returned as a child collection.</p>
IsLoadMetadata	<p><code>boolean</code>, <code>false</code> by default. If set, metadata for every project would be returned as a list of key-value entries.</p>

## Request URL



```
https://api.approval.studio/api/v1/project?ProjectUID=XXXXX&
States=Active%2COnHold%2CCompleted%2CArchived%2CInTransit&Query=Some%20Free%20Text&
IsLoadAssets=true&IsLoadLastVerAssets=true&IsLoadTasks=true&IsLoadRefDocs=true
```

## Curl:

```
curl -X GET "https://api.approval.studio/api/v1/project?
                ProjectUID=XXXXXXXXXXXXXXXXXXXX&States=Active%2COnHold%2CCompleted&
                Query=Some%20Free%20Text&IsLoadAssets=true&
                IsLoadLastVerAssets=true&IsLoadTasks=true&IsLoadRefDocs=true" \
-H "accept: text/plain" \
-H "Authorization: Bearer YYYYYYYYYYYYYY"
```

- Default `States` is `Active,OnHold,Completed`.
- If `ProjectUID` is provided and no project is found, an empty list returns an HTTP code 200.

## Response

```
[
  {
    "projectUID": "GUID",
    "projectState": "Active|OnHold|Completed|InTransit|Archived",
    "name": "string",
    "customer": "string",
    "project": "string",
    "design": "string",
    "revision": "string",
    "description": "string",
    "tags": [
      "tag 1", "tag 2"...
    ],
    "dueDate": "2020-11-22T23:00:53.425Z",
```

```

"reviewStatus": {                                     // Project-level proof review status
                                                    // (see proofing flow explanations).
    "pendingCount": 1,                             // The asset has 1 uncompleted review task
    "approvedCount": 2,                             // The asset has been approved 2 times
    "rejectedCount": 3                             // The asset has been rejected 3 times
},
"created": "2020-11-22T23:00:53.425Z",
"template": {
    "templateUID": "XXXXXXXXXXXXXXXX", // Template's unique ID.
    "name": "string",                  // Template name, presumable unique for a given te
    "data": {                          // Optional data, associated with the project.
        "key": "value",                // Json or free-form string.
        "key1": "value1"
    }
}
"assets": { // Optional,
    "asset_one.jpeg": // Unique asset name
    [
        // List of assets' versions, one or more
        {
            "assetUID": "GUID",
            "version": 0,
            "status": "Pending|Processed|Failed", // Or integer, 0,1,2
            "reviewStatus": {                  // Proof review status (see proofin
                "pendingCount": 1,            // The asset has 1 uncompleted review t
                "approvedCount": 2,           // The asset has been approved 2 times
                "rejectedCount": 3            // The asset has been rejected 3 times
            },
            "pagesCount": int, >=1,
            "created": "2020-11-22T23:00:53.425Z",
            "fileSize": int, bytes,
            "reviewUrl": string,              // URL to a prooftool to view/proof this as
            "thumbnailUrl": string            // URL to asset's thumbnail image (.jpg).
            "fullSizeBitmapUrl": string        // URL to converted asset's image (.png).
        }
    ],

```

```
"asset_two.pdf": [
  {
    "assetUID": "GUID",
    "version": 0,
    "status": "Pending|Processed|Failed",
    "reviewStatus": {
      "pendingCount": 1,
      "approvedCount": 2,
      "rejectedCount": 3
    },
    "pagesCount": 0,
    "created": "2020-11-22T23:00:53.425Z",
    "fileSize": 0,
    "reviewUrl": "https://app.approval.studio/xxx",
    "thumbnailUrl": "https://app.approval.studio/yyy"
    "fullSizeBitmapUrl": "https://app.approval.studio/zzz"
  }
],
...
],
},
"tasks": [
  {
    "taskUID": "GUID",
    "type": "UploadAssets|UploadRefDocs|ReviewAssets|ExternalReviewAssets|UploadChanged
    "status": "Pending|Closed|Approved|Rejected",
    "comment": "string",
    "dueDate": "2020-11-22T23:00:53.425Z", // Optional.
    "created": "2020-11-22T23:00:53.425Z",
    "closed": "2020-11-22T23:00:53.425Z",
    "user": {
      "userUID": "GUID",
      "fullName": "string",
      "email": "string"
    },
  },
]
```

```

        "assets": [
            "Asset GUID", "Asset GUID 2"...
        ],
        "reviewUrl": string                // URL to launch a prooftool for the given ta
                                           // Appears only for the ReviewAsset task.
    }
],
"refDocs": [
    {
        "refDocGUID": "GUID",
        "created": "2020-11-22T23:00:53.425Z",
        "name": "filename.ext",
        "fileSize": int, bytes.
    }
],
"metadata": [
    "orderNumber": "A19KQ64A",            // Key-Value pair(s) of metadata
    "contectEmail": "contact@mail.com"
],
"client": {
    "clientUID": "YYYYYYYYYYYYYY",       // Client's unique ID.
    "name": "Yourcompany Ltd"             // Client name.
},
"kanbanColumnUID": "j2ax"                // Optional kanban column UID for this project.
}
]
```

## POST /api/v1/project

---

**Creates a new project** taking mandatory project name and a list of owners and a set of optional attributes.

## Request

Field	Type & Explanation
clientUID	string[50] Optional client UID. See <a href="#">GET /api/v1/users</a> / <a href="#">Responses</a> . If no client ID is provided, the first client will be chosen by default.
projectName	string[200] Mandatory project name, free-form text.
customer	string[200] Optional customer name.
project	string[50] Optional (sub)project name.
design	string[50] Optional design type/name, like “package” or “banner” etc.
revision	string[50] Optional revision number, sequential or free-form.
description	string[1000] Optional project description, free-form text.
tags	string array Optional tag list, max 20 tags of max length of 25 chars each.
dueDate	ISO date Optional UTC date (or date-time) that point to a date when project supposed to be completed. The date affects the project’s status and sort order on the application dashboard.
projectOwnersUIDS	string array Mandatory list of the project owner(s)'s UIDs. At least one owner must be provided, max number of owners is 20.

```
{
  "clientUID": "string",          // Optional tenant(client) ID.
  "projectName": "string",       // Project name, mandatory.
  "customer": "string",
  "project": "string",
```

```

"design": "string",
"revision": "string",
"description": "string",
"tags": [
  "string", "string"
],
"dueDate": "2020-12-07T20:56:38.818Z",
"projectOwnersUIDS": [ // Project owner(s), at least one owner must be provided.
  "XXXXXXXX"
],
"templateUID": "string" // Optional project template UID. See client list to get an appro
}

```

```

curl -X POST "https://api.approval.studio/api/v1/project"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d '{"projectName\":\"string\",\"customer\":\"string\",\"project\":\"string\",\"d

```

## Responses

HTTP Code	Response
200	Project created and instance returned.

```

{
  "projectUID": "string",
  "projectState": 0,
  "name": "string",
  "customer": "string",
  "project": "string",
  "design": "string",

```

```
{
  "revision": "string",
  "description": "string",
  "tags": [
    "string"
  ],
  "dueDate": "2020-11-27T13:37:07.534Z",
  "created": "2020-11-27T13:37:07.534Z",
  "template": {
    "templateUID": "string",
    "name": "string",
    "data": "string or object"
  },
  "client": {
    "clientUID": "string",
    "name": "string"
  }
}
```

HTTP Code	Response
400	<b>Error:</b> Bad Request. One of the pre-requisites failed to validate

## PUT /api/v1/project

---

**Changes project's attribute(s)** including name, due date, and list of owners (edit project).

### Request

```
{
  "projectUID": "ProjectUID", // ID of project to edit
```

```

    "projectName": "string",
    "customer": "string",
    "project": "string",
    "design": "string",
    "revision": "string",
    "description": "string",
    "tags": [
        "string", "string", "string"
    ],
    "dueDate": "2020-12-02",
    "projectOwnersUIDS": [
        "UserID", "UserID" ...
    ]
}

```

Omit those properties you want to stay untouched; so if you provide a request like this below, only the project name will be updated:

```

{
    "projectUID": "XXXXXXXXXX", // Project ID to edit
    "projectName": "New name"
}

```

### Curl:

```

curl -X PUT "http://api.approval.studio/api/v1/project"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYY..."
-H "Content-Type: application/json"
-d "{\"projectUID\":\"XXXXXXXXXX\",\"projectName\":\"New Name\"}"

```

### Responses



HTTP Code	Response
<b>200</b>	<b>Success.</b> The project's attributes changed.

```
{
  "projectUID": "XXXXXXXXXX",
  "projectState": "Active|OnHold|Completed|InTransit|Archived",
  "name": "string",
  "customer": "string",
  "project": "string",
  "design": "string",
  "revision": "string",
  "description": "string",
  "tags": [
    "string", "string", ...
  ],
  "dueDate": "2020-11-30",
  "created": "2020-11-30T12:09:36.426Z"
}
```

HTTP Code	Response
<b>400</b>	<b>Error:</b> Parameters' validation failed.

See HTTP code 400 description.

HTTP Code	Response
<b>404</b>	<b>Error:</b> Project with the given ID not found or it has been already deleted.

```
{
  "version": "1.0",
  "statusCode": 404,
```

```
"message": "Project not found or already deleted."
}
```

HTTP Code	Response
406	<b>Error:</b> Project editing is possible only when the project is <code>Active</code> or <code>OnHold</code> . <code>Completed</code> , <code>Archived</code> or <code>InTransit</code> projects are not mutable. Please see <a href="#">PUT /api/v1/project/state</a> .

```
{
  "version": "1.0",
  "statusCode": 406,
  "message": "Can't edit completed or archived projects."
}
```

HTTP Code	Response
412	<b>Error:</b> Project must be in state <b>Completed</b> or <b>Archived</b> . If not, the code 412 returns.

```
{
  "version": "1.0",
  "statusCode": 412,
  "message": "Only archived or on-hold project can be deleted."
}
```

## DELETE /api/v1/project

**Deletes a project.**

This is undoable; once the project is deleted, it disappears from a list of projects; assets and uploaded reference documents are deleted as well.

The project should have the status **Completed** or **Archived** to be deleted; an error will be thrown elsewhere,

## Request

```
{
  "projectUID": "XXXXXXXXXXXX"
}
```

## Curl:

```
curl -X DELETE "https://api.approval.studio/api/v1/project" \
  -H "accept: text/plain" \
  -H "Authorization: Bearer YYYYYYYYYYYYYYYYYY..." \
  -H "Content-Type: application/json" \
  -d "{\"projectUID\":\"XXXXXXXXXXXX\"}"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Project deleted.

```
{
  "version": "1.0",
  "statusCode": 200,
```

```
"message": "Project deleted."
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.

See HTTP code 400 description.

HTTP Code	Response
404	<b>Error:</b> Project with the given ID not found, and, therefore project deletion failed.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project not found."
}
```

HTTP Code	Response
412	<b>Error:</b> Project must be either <b>Completed</b> or <b>Archived</b> to be deleted.

## GET /api/v1/project/proofreport

**Returns URL to** get a printable **proof report** for the given project. The method does not allow direct downloading.

Read `result/downloadURL` and navigate it to get the report body. The report is a plain single-layer PDF 1.4 file.

## Request

The request is a set of **URL GET** parameters separated by **&**:

```
curl -X GET "https://api.approval.studio/api/v1/project/proofreport?ProjectUID=XXXXXXXXXXXX" \
-H "accept: text/plain" \
-H "Authorization: Bearer YYYYYYY"
```

Parameter	Type & Explanation
ProjectUID	string[50] Mandatory project identifier.

## Responses

HTTP Code	Response
200	<b>Success.</b> Proof report URL generated.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "downloadURL": "https://approval.studio/ProofApi/GetProofReport/DDDDDDDDDDDDDDDDDDDDDDDD",
    "project": {
      "projectUID": "3AB3A5F58951467B975378198C7265D1",
      "projectState": "Complete",
      "name": "Project for Pepsi Co",
      "tags": ["tag1", "tag2"],
      "created": "2020-12-02T13:57:15.568992"
    }
  }
}
```

```
}  
}
```

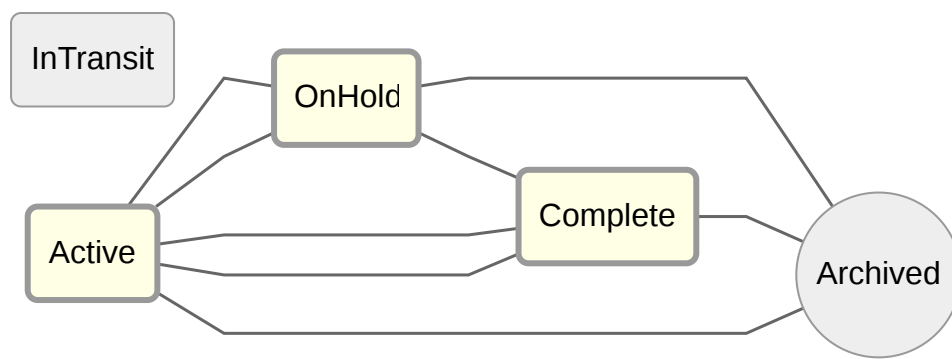
HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
-----	-----
404	<b>Error:</b> Project with the given ID not found and therefore proof report generation failed.

```
{  
  "version": "1.0",  
  "statusCode": 404,  
  "message": "Project not found."  
}
```

## PUT /api/v1/project/state

---

**Changes project's state** according to the state rules:



The diagram above explains the pre-requisite check for changing the project's state.

**Note:** The diagram above **is a recommended way** to change project's state, **it is not mandatory to follow it**, but highly recommended.

**Note:** Detailed description of the project management is out of scope of this document.

## States

Those are the states:

State	Explanation
<b>Active</b>	Work on a project is undergoing. This is where all the review work is going on.
<b>OnHold</b>	Due to some reason you want to postpone working on a project. Setting state to <b>OnHold</b> moves the project to a separate lane. Tasks are still intact, the only difference from Active is a position on a separate lane on the dashboard.
<b>Completed</b>	When setting the state to <b>Completed</b> all project tasks are deleted and the project itself moves from the dashboard to a separate screen where all completes are stored separately. All the assets and reference documents are preserved as well as history etc. The project goes read-only.

<b>Archived</b>	This is generally the same as <b>Completed</b> but assets and reference documents are moved to a remote, slow storage. The process of moving files to other storage could take a while, so when it is going on, the system marks a project as <b>InTransit</b> (see below).
<b>InTransit</b>	Project is locked to <b>InTransit</b> state by API itself when it is switching to <b>Archive</b> state or, vice versa, from <b>Archived</b> to <b>Active</b> . You can't set this state forcedly.

### Request

```
{
  "projectUID": "XXXXXXXXXX",
  "projectState": "Active|OnHold|Completed|Archived"
}
```

### Responses

HTTP Code	Response
<b>200</b>	<b>Success.</b> The project's state changed.

```
{
  "projectUID": "XXXXXXXXXX",
  "projectState": "Active|OnHold|Completed|Archived",
  "name": "string",
  "customer": "string",
  "project": "string",
  "design": "string",
  "revision": "string",
  "description": "string",
  "tags": [
```



```
"string", "string", ...
],
"dueDate": "2020-11-30",
"created": "2020-11-30T12:09:36.426Z"
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Project with the given ID not found or it has been already deleted.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project not found."
}
```

HTTP Code	Response
406	<b>Error:</b> The project with the given ID already has the requested state; no updating made.

```
{
  "version": "1.0",
  "statusCode": 412,
  "message": "Project already has state XXXX."
}
```

## GET /api/v1/project/meta

This methods **gets a key-value list of the metadata** for the given project.

The concept of project's metadata is the following: you can assign any kind of textual invisible for end-users data to a chosen project. Technically this is a list of key-value pairs.

Project's metadata might be used to provide data necessary for integration with other software, like order number, linked with a project, email of contact person etc.

**Note:** Apart from this method, you may use [GET /api/v1/project](#) with flag `IsLoadMetadata` set to `true`.

**Curl:**

```
curl -X 'GET' \
'https://api.approval.studio/api/v1/project/meta?ProjectUID=XXXXXXXXXXXXXXXXXX' \
-H 'accept: text/plain'
-H 'Authorization: Bearer ZZZZZZZZZZZZZZZZZZZZZZ'
```

Parameter	Type & Explanation
ProjectUID	<code>string</code> Unique project identifier.

## Responses

HTTP Code	Response
200	<b>Success.</b> List of metadata entries returned.

```
{
  "version": "1.0",
```

```
"statusCode": 200,
"message": "GET Request successful.",
"result": {
  "orderNumber": "A19KQ64A",
  "contactEmail": "contact@username.com"
}
}
```

// Metadata group  
// Key-Value pair(s) of metadata

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Project with the given ID not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project not found."
}
```

## POST /api/v1/project/meta

This methods **adds one or more metadata entry** to the given project.

### Request:

```
{
  "projectUID": "D37683CE23E24079951DC48C96114094",
```

```
"metadata": {
  "orderNumber": "A19KQ64A",
  "contactEmail": "contact@username.com"
}
```

## Curl:

```
curl -X 'POST' \
  'https://api.approval.studio/api/v1/project/meta' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer YYYYYYYYY' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
    "projectUID": "D37683CE23E24079951DC48C96114094",
    "metadata": {
      "orderNumber": "A19KQ64A",
      "contactEmail": "contact@username.com"
    }
  }'
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Metadata added successfully.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Metadata added."
}
```

HTTP Code	Response
<b>400</b>	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
<b>404</b>	<b>Error:</b> Project with the given ID not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project not found."
}
```

HTTP Code	Response
<b>406</b>	<b>Error:</b> Completed and archived projects can't be altered. You can add metadata to only project with statuses <code>Active</code> or <code>OnHold</code> .

## PUT /api/v1/project/meta

This methods **edits an existing metadata entry** of the given project.

**Request:**

```
{
  "projectUID": "XXXXXXXXXXXXXXXX",
  "key": "orderNumber",
```

```
"value": "A19KQ64A"
}
```

## Curl:

```
curl -X 'PUT' \
  'https://api.approval.studio/api/v1/project/meta' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer YYYYYYYY' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
    "projectUID": "XXXXXXXXXXXXXXXX",
    "key": "orderNumber",
    "value": "A19KQ64A"
  }'
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Metadata changed successfully.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Metadata edited."
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.

	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Project with the given ID not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project not found."
}
```

HTTP Code	Response
406	<b>Error:</b> Completed and archived projects can't be altered. You can add metadata to only project with statuses <code>Active</code> or <code>OnHold</code> .

## DELETE /api/v1/project/meta

This methods **deleted an existing metadata entry** of the given project.

### Request:

```
{
  "projectUID": "XXXXXXXXXXXXXXXX",
  "key": "orderNumber"
}
```

### Curl:

```
curl -X 'DELETE' \
  'https://api.approval.studio/api/v1/project/meta' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer YYYYYYYY' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
    "projectUID": "XXXXXXXXXXXXXXXX",
    "key": "orderNumber"
  }'
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Metadata entry deleted successfully.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Metadata deleted."
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.

See HTTP code 400 description.

HTTP Code	Response
404	<b>Error:</b> Project with the given ID not found.



```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project not found."
}
```

HTTP Code	Response
406	<b>Error:</b> Completed and archived projects can't be altered. You can add metadata to only project with statuses <code>Active</code> or <code>OnHold</code> .

## GET /api/v1/project/meta/projects

This methods gets one or more projects that contain metadata with the given key and optional value.

### Curl:

```
curl -X 'GET' \
  'https://api.approval.studio/api/v1/project/meta/projects?Key=orderNumber&Value=A19KQ64A&' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer YYYYYYYYY'
```

### Responses

HTTP Code	Response
200	<b>Success.</b> Search successful. Actually, zero or more projects found.

```

{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": [
    {
      "projectUID": "XXXXXXXXXXXX",
      "projectState": "Active",
      "name": "Integratiion Design",
      "tags": [
        "Design"
      ],
      "reviewStatus": {
        "pendingCount": 5,
        "approvedCount": 4,
        "rejectedCount": 0
      },
      "created": "2022-10-17T16:30:14.971521",
      "owners": [
        "DDDDDDDDDDDDDD"
      ]
      "client": {
        "clientUID": "CCCCCCCCCCCCCCCC",
        "name": "Enterprize Company"
      }
    },
    [...]
  ]
}

```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.

See HTTP code 400 description.

## POST /api/v1/kanban/column

This methods **moves a given project to a given kanban column** on the dashboard.

**Curl:**

```
curl -X 'POST' \
  'https://api.approval.studio/kanban/column' \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
    "projectUID": "XXXXXXXXXX", // ProjectUID as is throughout the system.
    "kanbanColumnUID": "asdFxQ" // A valid kanban column UID.
  }'
```

**Note:** You may retrieve a list of kanban column UIDs from the `GET /api/v1/users` method.

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
404	<b>Error:</b> Project with the given UID not found.
404	<b>Error:</b> Kanban column with the given UID not available for this project.
	Please read kanban column UIDs using the <code>GET /api/v1/users</code> method

406	<b>Error:</b> Can't edit completed or archived projects.
	You may move a project to a given kanban column only when this project is active or on hold.

## Asset management

Method/Path	Description
GET /api/v1/assets	Gets an asset instance.
DELETE /api/v1/assets	Deletes an asset.
POST /api/v1/assets/upload	Uploads an asset from file.
POST /api/v1/assets/upload_url	Uploads an asset from the Internet.
GET /api/v1/assets/download	Downloads an asset.
GET /api/v1/assets/proofreport	Gets a link to a proof report for an asset.

## GET /api/v1/asset

**Gets an asset** instance by a given asset id.

Parameter	Type & Explanation
-----------	--------------------

AssetUID

string Unique asset GUID.

## Curl:

```
curl -X GET "http://api.approval.studio/api/v1/asset?AssetUID=XXXXXXX" -H "accept: text/pla
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Asset instance returned.

```
"version": "1.0",
"statusCode": 200,
"message": "GET Request successful.",
"result": {
  "assetUID": "xxxxxxxxxx",
  "version": 1,
  "name": "drawing.pdf",
  "status": "Processed",
  "reviewStatus": {
    "pendingCount": 1,
    "approvedCount": 2,
    "rejectedCount": 3
  },
  "pagesCount": 3,
  "created": "2020-10-22T08:09:05.778512",
  "fileSize": 1063800,
  "reviewUrl": "https://app.approval.studio/xxx", // URL to launch prooftool to view th
  "thumbnailUrl": "https://app.approval.studio/yyy" // URL to asset's thumbnail image (.j
  "fullSizeBitmapUrl": "https://app.approval.studio/zzz/pageNum" // URL to converted asse
```

```
}  
}
```

**Note:** `fullSizeBitmapUrl` is provided in form `"https://baseurl/assettoken/0"` .

– “0” here is a zero-based page number, zero is used by default, but you can change it to whatever you need.

Please remember that some image asset formats are multipage (.pdf, .tiff, office formats etc.).

– You will get **http code 404** if provide wrong page number, bigger than asset’s page count or if you provide negative value.

HTTP Code	Response
400	<b>Error:</b> Parameters’ validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Asset with the given ID not found.

```
{  
  "version": "1.0",  
  "statusCode": 404,  
  "message": "Asset not found."  
}
```

## DELETE /api/v1/asset

---

**Deletes an asset** by a given asset id.

Asset deletion is an unrecoverable operation that leads to removing the asset from the project and deleting

a file from storage.

It is no way to restore an asset after it has been deleted.

## Request

```
{
  "AssetUID": "XXXXXXXXXXXX"
}
```

## Curl:

```
curl -X DELETE "https://api.approval.studio/api/v1/asset" \
  -H "accept: text/plain" \
  -H "Authorization: Bearer YYYYYYYYYY" \
  -H "Content-Type: application/json-patch+json" \
  -d "{\"assetUID\":\"XXXXXXXXXXXX\"}"
```

## Responses

HTTP Code	Response
200	Success. Asset deleted.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Asset deleted."
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Asset with the given ID not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Asset not found."
}
```

## POST /api/v1/asset/upload

---

**Uploads an asset** and **initiates asset processing**.

It validates initial input and **adds the asset to the asset processing queue**, which runs asynchronously.

After uploading you **need to pool asset status** to get to know when it is processed or failed to process.

You can **upload assets directly** to a selected project or **point an upload task** as an upload initiator.

- Time required to process assets is highly dependent on asset file size, dimensions, number of pages, and on how much processing node(s) are loaded.
- Asset processing may fail or be rejected due to a number of reasons, business and technical. Those could be asset type (file extension), resolution or physical size (in case of vector images - PDF/AI, etc), payment plan, and a number of other options. Please refer to the company's site or/and support to learn more.





HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
403	<b>Error:</b> You have reached the storage limit for your plan (XXX Gb) and therefore not allowed to upload new assets.
	This code means you have no free storage space to upload a new asset; asset is not uploaded and will not be processed any further. Please buy more storage or contact support to assist in dealing with this.
404	<b>Error:</b> Project UID is either invalid or points to a non-existing or inactive project.
404	<b>Error:</b> Task UID is either invalid or points to a non-existing or inactive task.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project UID provided is either invalid or points to a non-existing or inactive t
            "Task UID provided is either invalid or points to a non-existing or inactive t
}
```

HTTP Code	Response
412	<b>Error:</b> Only task types <b>UploadAssets</b> and <b>UploadChangedAsset</b> are allowed.
412	<b>Error:</b> In case when it is <b>UploadChangedAsset</b> task: the file to upload must be the same type as the original file.

Note: **UploadChangedAsset** means that the project owner(s) requested a new version of the asset to upload. It is mandatory that all asset versions must be the same type, i.e. all versions of the same asset are PDF or JPEG or PNG, etc. If you provide a different file type, you get this error.

```
{
  "version": "1.0",
  "statusCode": 412,
  "message": "Only task types UploadAssets and UploadChangedAsset are allowed." -or-
    "File to upload must be the same type as the original file (.pdf)."
}
```

## POST /api/v1/asset/upload\_url

---

This method works, generally, in the same way as `/api/v1/asset/upload` with the only difference that the file to upload is taken from the publicly accessible http(s) server.

A mandatory `URL` parameter points to the file to upload and process as a project's asset.

### Request

Parameter	Type & Explanation
<b>ProjectUID</b>	<code>string</code> Unique project ID, <b>mandatory</b> .
<b>FileName</b>	<code>form file</code> Asset file name, <b>optional</b> . Note: It's just a name, not a path. If not provided, the filename would be assigned automatically using filename taken from http header

	Content-Disposition . If this header is not provided, filename would be "asset.extension" where extension is taken from http content type.
<b>URL</b>	<code>string</code> URL to download a file from, <b>mandatory</b> .

### Curl:

```
curl -X POST "http://api.approval.studio/api/v1/asset/upload_url"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d '{
  "projectUID": "XXXXXXXXX",
  "fileName": "filename.jpeg",
  "url": "https://cdn.images.com/YYYYYYYYYYYY"
}'
```

### Responses

HTTP Code	Response
<b>200</b>	<b>Success.</b> "Asset uploaded successfully and is pending to process."

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Asset uploaded successfully and is pending to process. Track it's status to c
  "result": {
    "assetUID": "XXXXXXXXX"
  }
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
403	<b>Error:</b> You have reached the storage limit for your plan (XXX Gb) and therefore not allowed to upload new assets.
	This code means you have no free storage space to upload a new asset; asset is not uploaded and will not be processed any further. Please buy more storage or contact support to assist in dealing with this.
404	<b>Error:</b> Project UID is either invalid or points to a non-existing or inactive project.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project UID provided is either invalid or points to a non-existing or inactive project"
}
```

## GET /api/v1/asset/download

Returns URL to **download asset**.

### Request

Parameter	Type & Explanation
-----------	--------------------

**AssetUID**

string Unique project ID, **mandatory**.

## Curl:

```
curl -X GET "http://api.approval.studio/api/v1/asset/download?AssetUID=XXXXXXXXXXXXXX"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYY"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> URL generated.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "downloadURL": "https://approval.studio/ProofApi/DownloadAsset/ZZZZZZZZZZ[...]",
    "asset": {
      "assetUID": "XXXXXXXXXXXXXX",
      "version": 1,
      "name": "FileName.jpg",
      "status": "Processed",
      "pagesCount": 1,
      "created": "2020-12-03T23:54:53.40858",
      "fileSize": 20743
    }
  }
}
```

HTTP Code	Response
<b>400</b>	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
<b>404</b>	<b>Error:</b> Asset with the given ID not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Asset not found."
}
```

## GET /api/v1/asset/proofreport

---

Returns **URL** to download **asset's proof report**.

### Request

Parameter	Type & Explanation
<b>AssetUID</b>	<code>string</code> Unique project ID, <b>mandatory</b> .

**Curl:**

```
curl -X GET "http://api.approval.studio/api/v1/asset/proofreport?AssetUID=XXXXXXXXXXXXX"  
-H "accept: text/plain"  
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYY"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> URL generated.

```
{  
  "version": "1.0",  
  "statusCode": 200,  
  "message": "GET Request successful.",  
  "result": {  
    "downloadURL": "https://approval.studio/ProofApi/GetProofReport/ZZZZZZZZZZ[...]",  
    "asset": {  
      "assetUID": "XXXXXXXXXXXXX",  
      "version": 1,  
      "name": "FileName.jpg",  
      "status": "Processed",  
      "pagesCount": 1,  
      "created": "2020-12-03T23:54:53.40858",  
      "fileSize": 20743  
    }  
  }  
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.



	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Asset with the given ID not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Asset not found."
}
```

## Reference documents management

Method/Path	Description
POST /api/v1/refdoc/upload	Uploads a reference document and initiates document's processing.
POST /api/v1/refdoc/upload_url	Uploads an document taken from the publicly accessible URL and initiates its processing.
POST /api/v1/refdoc/add_url	Adds a URL as a reference document.
GET /api/v1/refdoc/download	Returns a URL to download the reference document.
DELETE/api/v1/refdoc	Deletes a reference document.

# POST /api/v1/refdoc/upload

Uploads a document as a local file and assigns it to the given project.  
generally, works like asset uploading,

## Request

Parameter	Type & Explanation
ProjectUID	<code>string</code> Unique project ID, <b>mandatory</b> .
FileName	<code>form file</code> Document file name, <b>mandatory</b> . Note: It's just a name, not a path.

## Curl:

```
curl -X POST "http://api.approval.studio/api/v1/refdoc/upload?ProjectUID=XXXXXXXXX&FileName
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYY"
-H "Content-Type: multipart/form-data"
-F "uploadedFile=refdocfilename.jpg;type=image/jpeg"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> "Asset uploaded successfully and is pending to process."

```
{
  "version": "1.0",
}
```

```

"statusCode": 200,
"message": "Document uploaded successfully and is pending to process.
           Track its status to catch when it is ready to use.",
"result": {
  "refDocUID": "AAAAAAAAAAAAAAAAAAAA" // newly generated document id.
}
}

```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Project UID is either invalid or points to a non-existing or inactive project.

```

{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project UID provided is either invalid or points to a non-existing or inactive project."
}

```

## POST /api/v1/refdoc/upload\_url

This method uploads reference document by taking it from the publicly accessible http(s) server.

A mandatory `URL` parameter points to the file to upload and process as a project's asset.

### Request

Parameter	Type & Explanation
<b>ProjectUID</b>	<code>string</code> Unique project ID, <b>mandatory</b> .
<b>FileName</b>	<code>form file</code> Document file name, <b>optional</b> . Note: It's just a name, not a path. If not provided, the filename would be assigned automatically using filename taken from http header <code>Content-Disposition</code> . If this header is not provided, filename would be <code>"asset.extension"</code> where extension is taken from http content type.
<b>URL</b>	<code>string</code> URL to download a file from.

#### Curl:

```
curl -X POST "http://api.approval.studio/api/v1/refdoc/upload_url"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d '{
  "projectUID": "XXXXXXXXXX",
  "fileName": "filename.jpeg",
  "url": "https://cdn.images.com/YYYYYYYYYYYY"
}'
```

#### Responses

HTTP Code	Response
<b>200</b>	<b>Success.</b> "Document uploaded successfully and is pending to process."

```
{
  "version": "1.0",
```

```
"statusCode": 200,  
"message": "Document uploaded successfully and is pending to process. Track it's status t  
"result": {  
  "refDocUID": "XXXXXXXXX"  
}  
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Project UID is either invalid or points to a non-existing or inactive project.

```
{  
  "version": "1.0",  
  "statusCode": 404,  
  "message": "Project UID provided is either invalid or points to a non-existing or inactiv  
}
```

## POST /api/v1/refdoc/add\_url

Plain text URL is a kind of a reference document in the Approval Studio. You can either upload a file or add a URL to the list of reference documents in a project.

This method adds a URL to the document list for a given project.

**Curl:**

```
curl -X POST "http://api.approval.studio/api/v1/refdoc/add_url"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d '{
  "projectUID": "XXXXXXXXX",
  "url": "https://cdn.images.com/YYYYYYYYYYYY"
}'
```

## Responses

HTTP Code	Response
200	<b>Success.</b> "Document uploaded successfully and is pending to process."

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Reference document's URL added to the list.",
  "result": {
    "refDocUID": "XXXXXXXXX"
  }
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
-----------	----------

**404****Error:** Project UID is either invalid or points to a non-existing or inactive project.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project UID provided is either invalid or points to a non-existing or inactive project."
}
```

## GET /api/v1/refdoc/download

This method returns URL to download a given reference document.

If you need to download the document itself, please use any available suitable technology to do this using provided URL.

### Curl:

```
curl -X POST "http://api.approval.studio/api/v1/refdoc/download?refDocUID=XXXXXXXXX"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYY"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> URL returned in the response JSON.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "downloadURL": "https://ZZZZZZZZZZ"
  }
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Reference document UID is either invalid or points to a non-existing or deleted document.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Reference document not found."
}
```

## DELETE/api/v1/refdoc

This method deletes a reference document or reference URL.



**Curl:**

```
curl -X DELETE "http://api.approval.studio/api/v1/refdoc"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYY"
```

**Responses**

HTTP Code	Response
200	<b>Success.</b> URL returned in the response JSON.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Document deleted."
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Reference document UID is either invalid or points to a non-existing or deleted document.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Reference document not found or it was already deleted."
}
```

## Task management

---

Method/Path	Description
GET /api/v1/task/all	Gets a list of tasks assigned to the user.
GET /api/v1/task	Gets a task instance.
DELETE /api/v1/task	Deletes an task.
POST /api/v1/task/asset_upload	Creates a new AssetUpload task.
POST /api/v1/task/refdoc_upload	Creates a new RefDocUpload task.
POST /api/v1/task/review_asset	Creates a new CreateReviewAsset task.
PUT /api/v1/task/complete	Completes a task.

### GET /api/v1/task/all

---

**Gets a list of tasks** optionally filtered by task types, assigned to a currently logged-in user.

This is what a user sees in Approval Studio web application in the list of tasks in the dashboard.

Parameter	Type & Explanation
Types	<code>string</code> One or more task types, comma-separated.
	Default value is <code>UploadAssets, UploadRefDocs, ReviewAssets, ExternalReviewAssets, UploadChangedAsset, UploadVideo, ExternalReviewVideo, ReviewVideo</code> .
	If no parameter is provided or it is empty, a default list of types is used (see above).

### Curl:

```
curl -X GET "http://api.approval.studio/api/v1/task/all?Types=UploadAssets"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYY"
```

### Responses

HTTP Code	Response
200	<b>Success.</b> Task instance returned.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": [
    {
```

```

    "taskUID": "XXXXXXXXXXXX1",
    "projectUID": "YYYYYYYYYYYY1",
    "projectName": "Project name, string",
    "type": "UploadAssets|UploadRefDocs|ReviewAssets|ExternalReviewAssets|UploadCha
    "status": "Pending|Closed|Approved|Rejected|ApprovedWithChanges",
    "created": "2020-11-27T15:39:59.864882",
    "closed": "2020-11-27T15:40:51.407384",
    "user": {
      "userUID": "92AFE33153124F0980E43EF80133FE9B",
      "fullName": "John Smith",
      "email": "john.smith@email.com"
    },
    "reviewUrl": string , // URL to launch prooftool for giv
                        // Appears only in ReviewAsset tas
    "requestedAssetName": "butterfly_poster.pdf" // Name of the asset, a task r
                        // Appears only in UploadChang
  },
  {
    "taskUID": "XXXXXXXXXXXX2",
    "projectUID": "YYYYYYYYYYYY2",
    ...
  },
  ...
]
}

```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.

# GET /api/v1/task

**Gets a task** instance for a given task id.

Parameter	Type & Explanation
TaskUID	string Unique task ID.

**Curl:**

```
curl -X GET "http://api.approval.studio/api/v1/task?TaskUID=XXXXXXXXXXXXXXXX"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYY"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Task instance returned.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "taskUID": "XXXXXXXXXX",
    "projectUID": "YYYYYYYYYYYY",
    "projectName": "string",
    "clientUID": "ZZZZZZZZZZ",
    "type": "UploadAssets|UploadRefDocs|ReviewAssets|ExternalReviewAssets|UploadChangedAsse
```

```

"status": "Pending|Closed|Approved|Rejected|ApprovedWithChanges",
"created": "2020-11-27T15:39:59.864882",
"closed": "2020-11-27T15:40:51.407384",
"user": {
  "userID": "92AFE33153124F0980E43EF80133FE9B",
  "fullName": "John Smith",
  "email": "john.smith@email.com"
},
"reviewUrl": string,           // URL to launch prooftool for given task.
                                // Appears only for ReviewAsset task.
"requestedAssetName": "filename.pdf" // Name of the asset, a task references to.
                                    // Appears only in UploadChangedAsset task
}
}

```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Task with the given ID not found.

```

{
  "version": "1.0",
  "statusCode": 404,
  "message": "Task not found"
}

```

## DELETE /api/v1/task

## Deletes a task.

### Request

Parameter	Type & Explanation
taskUID	string Unique task ID.

### Curl:

```
curl -X DELETE "https://api.approval.studio/api/v1/task"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d "{\"taskUID\": \"XXXXXXXXXX\"}"
```

### Responses

HTTP Code	Response
200	<b>Success.</b> Login successful, AUTH token provided.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Task deleted."
}
```

HTTP Code	Response
-----------	----------

<b>400</b>	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
<b>404</b>	<b>Error:</b> Task with the given ID not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Task not found"
}
```

HTTP Code	Response
<b>410</b>	<b>Error:</b> the task is already completed or deleted or approved/rejected. Only pending task can be deleted.

```
{
  "version": "1.0",
  "statusCode": 410,
  "message": "The task is not pending and cannot be deleted."
}
```

## POST /api/v1/task/asset\_upload

**Creates** a new **AssetUpload task** for a given user providing optional due date and comment.



## Request

```
{
  "projectUID": "XXXXXXXXXXXXXXX", // Project this task belongs to, mandatory.
  "userID": "ZZZZZZZZZZZZZZ",      // User this task assigned to, mandatory.
  "dueDate": "2020-12-04",          // Task due date, UTC, optional.
  "comment": "comment text"         // Comment text, optional.
}
```

## Curl:

```
curl -X POST "https://api.approval.studio/api/v1/task/asset_upload" \
  -H "accept: text/plain" \
  -H "Authorization: Bearer YYYYYYYYYYYYYYYY" \
  -H "Content-Type: application/json-patch+json" \
  -d '{"projectUID":"XXXXXXXXXXXXXXX",
      "userID":"ZZZZZZZZZZZZZZ",
      "dueDate":"2020-12-04T13:11:11.526Z",
      "comment":"comment text"}'
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Task created.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Task created."
  "result": {
```

```
"task": {
  "taskId": "XXXXXXXXXXXXXXXXXXXXXXX",
  "projectId": 981,
  "projectUID": "YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY",
  "type": "UploadAssets",
  "status": "Pending",
  "comment": "Free-form text comment to the task",
  "dueDate": "2020-12-22",
  "created": "2020-12-22",
  "userID": "ZZZZZZZZZZZZZZ",
  "userName": "John Smith",
  "userEmail": "john.smith@gmail.com"
}
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.

|

HTTP Code	Response
404	<b>Error:</b> Given project not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Given project not found."
}
```

HTTP Code	Response
412	<b>Error:</b> You cannot create a task for a project in state [Complete]. Only projects that are <code>Active</code> or <code>OnHold</code> can have tasks.
412	<b>Error:</b> User with UID [ZZZZZZZZZZZZZZ] not found.

```
{
  "version": "1.0",
  "statusCode": 412,
  "message": "You cannot create a task for a project in state [Complete]."
```

## POST /api/v1/task/refdoc\_upload

**Creates** a new **RefDocUpload task** for a given user providing optional due date and comment.

### Request

```
{
  "projectUID": "XXXXXXXXXXXXXXXX", // Project this task belongs to, mandatory.
  "userID": "ZZZZZZZZZZZZZZ",      // User this task assign to, mandatory.
  "dueDate": "2020-12-04",          // Task due date, UTC, optional.
  "comment": "comment text"         // Comment text, optional.
}
```

### Curl:

```
curl -X POST "https://api.approval.studio/api/v1/task/refdoc_upload" \
-H "accept: text/plain" \
-H "Authorization: Bearer YYYYYYYYYYYYYYYY" \
-H "Content-Type: application/json-patch+json" \
-d "{\"projectUID\":\"XXXXXXXXXXXXXXXX\",
    \"userID\":\"ZZZZZZZZZZZZ\",
    \"dueDate\":\"2020-12-04\",
    \"comment\":\"comment text\" [...] }"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Task created.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Task created."
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Given project not found.

```

{
  "version": "1.0",
  "statusCode": 404,
  "message": "Given project not found."
  "result": {
    "task": {
      "taskUID": "XXXXXXXXXXXXXXXXXXXXXXX",
      "projectId": 981,
      "projectUID": "YYYYYYYYYYYYYYYYYYYYYYYYYYYYYY",
      "type": "UploadRefDocs",
      "status": "Pending",
      "comment": "Free-form text comment to the task",
      "dueDate": "2020-12-22",
      "created": "2020-12-22",
      "userUID": "ZZZZZZZZZZZZZZ",
      "userName": "John Smith",
      "userEmail": "john.smith@gmail.com"
    }
  }
}

```

HTTP Code	Response
412	<b>Error:</b> You cannot create a task for a project in the state <code>Complete</code> . Only projects that are <code>Active</code> or <code>OnHold</code> can have tasks.
412	<b>Error:</b> User with UID <code>[ZZZZZZZZZZZZZZ]</code> not found.

```

{
  "version": "1.0",
  "statusCode": 412,

```

```
"message": "You cannot create a task for a project in the state [Complete]."  
}
```

## POST /api/v1/task/review\_asset

---

**Creates a new Review Asset Task** for a given user, project, and a list of assets providing optional due date and comment.

### Request

```
{  
  "projectUID": "XXXXXXXXXXXXXXXX", // Project this task belongs to, mandatory.  
  "userID": "ZZZZZZZZZZZZZZ",      // User this task assign to, mandatory.  
  "assetUIDs": [  
    "AAAAAAAAAAAAAAAAA1",          // One or more assets to review, mandatory.  
    "AAAAAAAAAAAAAAAAA2"  
  ],  
  "dueDate": "2020-12-04",          // Task due date, UTC, optional.  
  "comment": "comment text"         // Comment text, optional.  
}
```

### Curl:

```
curl -X POST "https://api.approval.studio/api/v1/task/review_asset" \  
  -H "accept: text/plain" \  
  -H "Authorization: Bearer YYYYYYYYYYYYYYYY" \  
  -H "Content-Type: application/json-patch+json" \  
  -d '{"projectUID":"XXXXXXXXXXXXXXXX",  
      "userID":"ZZZZZZZZZZZZZZ",  
      "assetUIDs":["AAAAAAAAA1", "AAAAAAAAA2"]},'
```

```
\ "dueDate\ ": \ "2020-12-04\ ",  
\ "comment\ ": \ "comment text\ " [...]}"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Task created.

```
{  
  "version": "1.0",  
  "statusCode": 200,  
  "message": "Task created.",  
  "result": {  
    "task": {  
      "taskUID": "XXXXXXXXXXXXXXXXXXXXXXX",  
      "projectId": 981,  
      "projectUID": "YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY",  
      "type": "ReviewAssets",  
      "status": "Pending",  
      "comment": "Free-form text comment to the task",  
      "dueDate": "2020-12-22",  
      "created": "2020-12-22",  
      "userID": "ZZZZZZZZZZZZZZZZ",  
      "userName": "John Smith",  
      "userEmail": "john.smith@gmail.com"  
    }  
  }  
}
```

HTTP Code	Response
-----------	----------

<b>400</b>	<b>Error:</b> Parameters' validation failed.
<b>HTTP Code</b>	See HTTP code 400 description. <b>Response</b>
<b>404</b>	<b>Error:</b> Given project not found.
<b>404</b>	<b>Error:</b> Asset [AAAAAAAAAAAAAAAAAAAA1] not found...
<b>404</b>	<b>Error:</b> Asset [AAAAAAAAAAAAAAAAAAAA1] does not belong to project [XXXXXXXXXXXXXXXXXXXX].

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Given project not found."
}
```

<b>HTTP Code</b>	<b>Response</b>
<b>412</b>	<b>Error:</b> You cannot create a task for a project in the state [Complete]. Only projects that are <b>Active</b> or <b>OnHold</b> can have tasks.
<b>412</b>	<b>Error:</b> User with UID [ZZZZZZZZZZZZZZ] not found.

```
{
  "version": "1.0",
  "statusCode": 412,
```



```
"message": "You cannot create a task for a project in the state [Complete]."  
}
```

## POST /api/v1/task/review\_asset\_ext

---

**Creates a new External Review Asset Task** for someone who doesn't have an account in Approval Studio.

The flow is the following:

1. When a task is created, an email is sent to an email address from the request.
2. Email contains a link (URL) to a proof report review session.
3. Approve or reject terminates the review task and makes the URL expire.

### Request

```
{  
  "projectUID": "XXXXXXXXXXXXXXXX", // Project this task belongs to, mandatory.  
  "email": "string",                // User's email, mandatory.  
  "assetUIDs": [  
    "AAAAAAAAAAAAAAAAA1",          // One or more assets to review, mandatory.  
    "AAAAAAAAAAAAAAAAA2"  
  ],  
  "dueDate": "2020-12-04",          // Task due date, UTC, optional.  
  "password": "string",              // Optional password. When provided, the reviewer is  
                                     // to enter it before the review session.  
  "emailSubject": "string",          // Custom email subject line; override template's one  
  "emailLanguage": "English",        // Email language, optional. English is default.  
                                     // English, German, French, Polish, Spanish, Hebrew,  
  "comment": "comment text"          // Comment text, optional.  
  "isAllowDownloadAssets": true,     // When true, allows user to download original file.  
  "isReadOnly": true                 // When true, makes a review session read-only: no an
```

```
                                // no comments, no approve/reject. Optional.  
}
```

**Curl:**

```
curl -X POST "https://api.approval.studio/api/v1/task/review_asset_ext" \  
  -H "accept: text/plain" \  
  -H "Authorization: Bearer YYYYYYYYYYYYYYYYYY" \  
  -H "Content-Type: application/json-patch+json" \  
  -d '{"projectUID":"XXXXXXXXXXXXXXXXX",  
      "email":"ZZZZZZZZZZZZZ",  
      "assetUIDs":["AAAAAAAAA1", "AAAAAAAAA2"],  
      "dueDate":"2020-12-04",  
      "comment":"comment text" [...]}'
```

**Responses**

HTTP Code	Response
200	<b>Success.</b> Task created.

```
{  
  "version": "1.0",  
  "statusCode": 200,  
  "message": "Task created.",  
  "result": {  
    "task": {  
      "taskUID": "XXXXXXXXXXXXXXXXXXXXX",  
      "projectId": 981,  
      "projectUID": "YYYYYYYYYYYYYYYYYYYYYYYYYYYYY",  
      "type": "ExternalReviewAssets",  
      "status": "Pending",  
    }  
  }  
}
```

```
    "comment": "Free-form text comment to the task",
    "dueDate": "2020-12-22",
    "created": "2020-12-22",
    "userID": "ZZZZZZZZZZZZZZ",
    "userName": "John Smith",
    "userEmail": "john.smith@gmail.com"
  }
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Given project not found.
404	<b>Error:</b> Asset [AAAAAAAAAAAAAAAAAAAA1] not found...
404	<b>Error:</b> Asset [AAAAAAAAAAAAAAAAAAAA1] does not belong to project [XXXXXXXXXXXXXXXXXXXX].

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Given project not found."
}
```

HTTP Code	Response
412	<b>Error:</b> You cannot create a task for a project in the state [Complete]. Only projects that are <code>Active</code> or <code>OnHold</code> can have tasks.

```
{
  "version": "1.0",
  "statusCode": 412,
  "message": "You cannot create a task for a project in the state [Complete]."
}
```

## PUT /api/v1/task/complete

**Completes** a given **task**.

### Request

```
{
  "taskUID": "XXXXXXXXXXXXXX" // Task id to complete.
}
```

### Curl:

```
curl -X POST "https://api.approval.studio/api/v1/task/complete" \
  -H "accept: text/plain" \
  -H "Authorization: Bearer YYYYYYYYYYYYYYYY" \
```

```
-H "Content-Type: application/json-patch+json" \  
-d "{\"taskUID\":\"XXXXXXXXXXXXXXXXX\"}"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Task marked as completed.

```
{  
  "version": "1.0",  
  "statusCode": 200,  
  "message": "Task completed."  
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	<b>Error:</b> Given task not found.

```
{  
  "version": "1.0",  
  "statusCode": 404,  
  "message": "Task not found."  
}
```

HTTP Code	Response
410	<b>Error:</b> the task is already completed or deleted or approved/rejected. Only pending tasks can be completed.

```
{
  "version": "1.0",
  "statusCode": 410,
  "message": "The task is not pending and cannot be completed."
}
```

HTTP Code	Response
412	<b>Error:</b> Only upload-related tasks might be marked as completed:
	Allowed task types: UploadAssets , UploadChangedAsset , UploadRefDocs , UploadVideo

```
{
  "version": "1.0",
  "statusCode": 412,
  "message": "This type of task cannot be manually completed"
}
```

## Annotations management

---

Method/Path	Description
GET /api/v1/annotation/all	Returns a list of annotations for a given asset .
GET /api/v1/annotation	Returns an annotation.
DELETE /api/v1/annotation	Deletes an annotation.
PUT /api/v1/annotation/hide	Hides an annotation.
PUT /api/v1/annotation/complete	Completes an annotation.
PUT /api/v1/annotation/uncomplete	Un-completes an annotation.

## GET /api/v1/annotation/all

Returns a **list of annotations** for a given asset and page.

### Request

Parameter	Type & Explanation
<b>AssetUID</b>	string Unique project ID, <b>mandatory</b> .
<b>PageNum</b>	int Page number, zero-based, <b>mandatory</b> .

```
curl -X GET "http://api.approval.studio/api/v1/annotation/all?AssetUID=XXXXXXXXXXXXXXXXXX&PageNum=1" \
-H "accept: text/plain" \
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYY"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Annotations returned and a hierarchy built.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    {
      "0": [ // Key is the page number
        {
          "commentId": 0,
          "commentUID": "CCCCCCCCCCCCCCC",
          "body": "Annotation body",
          "drawingCode": "XXXXXXXXXXXXX",
          "created": "2020-12-04T16:35:11.083Z",
          "pageNum": 0,
          "sequenceId": 0,
          "isCompleted": true,
          "replies": [
            { annotation instance 1},          // The same annotation inst
            { annotation instance 2} [...]      // tree-like annotation hie
          ],
          "user": {
            "userId": "ZZZZZZZZZZ",
            "fullName": "string",
            "email": "string"
          }
        }
      ]
    },
  ],
}
```



```
      "1" : [...]  
    [...]  
  }  
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Asset with the given ID not found.

```
{  
  "version": "1.0",  
  "statusCode": 404,  
  "message": "Asset not found."  
}
```

## GET /api/v1/annotation

**Returns an annotation** for a given annotation id.

### Request

Parameter	Type & Explanation
-----------	--------------------

## AnnotationUID

string Unique annotation ID, **mandatory**.

```
curl -X GET "http://api.approval.studio/api/v1/annotation?AnnotationUID=XXXXXXXXXXXXXXXXX" \
-H "accept: text/plain" \
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYY"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Annotations returned and a hierarchy built.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "commentUID": "XXXXXXXXXXXXXXXXX",
    "body": "Annotation body",
    "drawingCode": "DDDDDDDDDDDD",          // Simple json-based markup code.
    "created": "2020-12-04T16:35:11.083Z",
    "pageNum": 0,
    "sequenceId": 0,
    "isCompleted": true,
    "user": {
      "userID": "ZZZZZZZZZZ",
      "fullName": "string",
      "email": "string"
    }
  }
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Annotation with the given ID not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Annotation not found."
}
```

## DELETE /api/v1/annotation

---

**Deletes an annotation** for a given annotation id.

### Request

Parameter	Type & Explanation
taskUID	string Unique task ID.

**Curl:**

```
curl -X DELETE "https://api.approval.studio/api/v1/annotation"  
-H "accept: text/plain"  
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"  
-H "Content-Type: application/json-patch+json"  
-d "{\"annotationUID\":\"XXXXXXXXXX\"}"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Annotation deleted.

```
{  
  "version": "1.0",  
  "statusCode": 200,  
  "message": "Annotation deleted."  
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.  See HTTP code 400 description.
HTTP Code	Response
404	<b>Error:</b> Annotation with the given ID not found.

```
{  
  "version": "1.0",  
  "statusCode": 404,  
}
```

```
"message": "Annotation not found."
}
```

HTTP Code	Response
412	<b>Error:</b> Only the user that created the annotation or comment can delete it. You cannot delete other's users' annotations and this annotation does not belong to you. Deletion failed.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "You can delete only your annotation."
}
```

## PUT /api/v1/annotation/hide

**Hides an annotation** for a given annotation id.

### Request

Parameter	Type & Explanation
taskUID	string Unique annotation ID.

**Curl:**

```
curl -X PUT "https://api.approval.studio/api/v1/annotation/hide"  
-H "accept: text/plain"  
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"  
-H "Content-Type: application/json-patch+json"  
-d "{\"annotationUID\":\"XXXXXXXXXX\"}"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Annotation was hidden.

```
{  
  "version": "1.0",  
  "statusCode": 200,  
  "message": "Annotation was hidden."  
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
406	<b>Error:</b> A user can hide only their own annotation.
404	<b>Error:</b> Annotation with the given ID not found.

```
{  
  "version": "1.0",  
  "statusCode": 404,
```

```
"message": "Annotation not found."
}
```

## PUT /api/v1/annotation/complete

---

**Completes an annotation** for a given annotation id.

### Request

Parameter	Type & Explanation
taskUID	string Unique annotation ID.

### Curl:

```
curl -X PUT "https://api.approval.studio/api/v1/annotation/complete"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d "{\"annotationUID\":\"XXXXXXXXXX\"}"
```

### Responses

HTTP Code	Response
200	<b>Success.</b> Annotation was marked as completed.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Annotation was completed."
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
404	<b>Error:</b> Annotation with the given ID not found.
406	<b>Error:</b> Annotation is already completed.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Annotation not found."
}
```

## PUT /api/v1/annotation/uncomplete

**Un-completes an annotation** for a given annotation id.

### Request

Parameter	Type & Explanation
-----------	--------------------



taskUID

string Unique annotation ID.

## Curl:

```
curl -X PUT "https://api.approval.studio/api/v1/annotation/uncomplete"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d "{\"annotationUID\": \"XXXXXXXXXX\"}"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Annotation was marked as uncompleted.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Annotation was uncompleted."
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
404	<b>Error:</b> Annotation with the given ID not found.
406	<b>Error:</b> Annotation is already completed.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Annotation not found."
}
```

## Users management

---

Method/Path	Description
GET /api/v1/users	Returns all clients(tenants) with users.
GET /api/v1/user/loggedin	Returns currently logged in user user.

### GET /api/v1/users

---

This method returns a list of available active users per tenant (client) and a template list for every client. Please read more about the concept of project templates in the project methods' group.

#### Curl:

```
curl -X GET "https://api.approval.studio/api/v1/users"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"
```

#### Responses

HTTP Code	Response
200	<b>Success.</b> List of users with companies they belong to returned.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "clients": [
      {
        "clientUID": "XXXXXXXXXXXXXXXXXXXX",
        "name": "HiTech Service",
        "users": [
          {
            "userID": "UUUUUUUUUUUUUUUUUUUU1",
            "fullName": "John Smith",
            "email": "john.smith@gmail.com",
            "role": "Administrator|RegularUser"
          },
          [...]
        ],
        "templates": [
          {
            "templateUID": "TTTTTTTTTTTTTTTTTTTT",
            "name": "First template",
            "data": {
              "key": "value",
              "key1": "value1"
            }
          },
          [...]
        ],
        "kanbanColumns": [ // List of kanban columns configured for the given client.
```

```
    {
      "name": "Active Projects",
      "kanbanColumnUID": "p8i41"
    },
    {
      "name": "In Production",
      "kanbanColumnUID": "d4df5a"
    }
  ]
},
[...]
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.

## GET /api/v1/user/loggedin

This method returns a currently logged in user; this is the used that was used when creating an authorization token in method `POST /api/v1/token/login`.

**Curl:**

```
curl -X GET "https://api.approval.studio/api/v1/users"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> List of users with companies they belong to returned.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "user": {
      "userID": "XXXXXXXXXXXXXXXXXXXX",
      "fullName": "John Smith",
      "email": "john.smith@company.com",
      "role": "Administrator"
    }
  }
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.

## Webhooks

---

Approval Studio API uses webhooks to notify your application when an event happens. Webhooks are particularly useful for asynchronous events like when an asset is accepted or rejected or when a new task is created or someone made a comment or a new asset version is uploaded etc.

Begin using the webhooks in three steps:

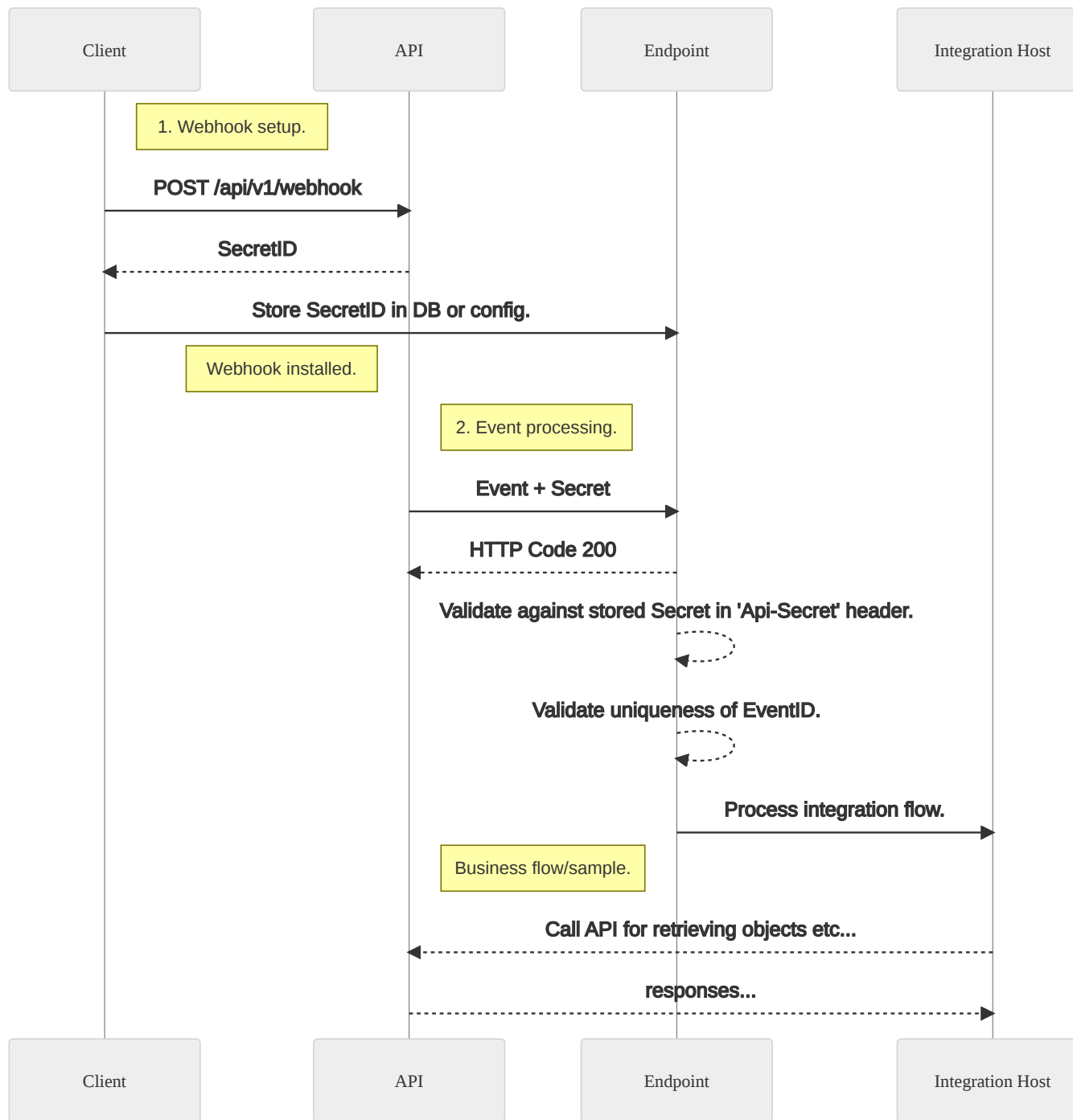
1. Create a webhook endpoint on your server.
2. Register the endpoint with Webhook management method **POST /api/v1/webhook**.
3. Test it to ensure that you can receive events using method **PUT /api/v1/webhook/test**.

Webhooks generally refer to a combination of elements that collectively create a notification and reaction system within a larger integration.

The webhook endpoint is code on your server, which could be written in C#, Java, Ruby, PHP, Node.js, or any other language/technology you prefer. The webhook endpoint has an associated URL (e.g., <https://example.com/webhook>).

**Note:** Explanations on how to build an endpoint are out of the scope of this document.

The Approval Studio notifications are Event objects. Those event objects contain all the relevant information about what just happened, including the type of event and the data associated with that event. The webhook endpoint uses the event details to take any required actions.



## Security

---

The API host provides a **Secret ID** with every event object posted on an endpoint. The endpoint needs to check against this ID for every incoming post to avoid spamming. Keep secret id safe.

## Retry logic

---

Webhook host implements a simple retry logic in order to try to deliver events in case of possible troubles on the endpoint side

If the endpoint does not return HTTP code 200, the webhooks host starts retrying posting the same event object up to ~30 minutes. After that time the event delivery silently fails.

## Event object

---

An event object is a JSON document that API posts on a selected endpoint. It generally has the following structure:

```
{
  "EventId": "DY3h1Pl040uIRvL74oKFlQ",      // Unique event id.
  "EventType": "annotation.added",           // Codename of the event, see below.
  "Created": "2021-02-18T14:08:08.615088Z",  // UTC datetime when the event was generated.
  "Data": {                                 // Data block, event-type dependent.
    "SomeData": "XXXXXXXXX"
```



```
}  
}
```

**EventID** is unique for every Event object. If for whatever reason, the API host makes more than one post on your endpoint with the same event object, you can identify it by reading EventID.

**Event types** are self-explanatory:

Event code	Description
project.created	Fires when a new project is created.
project.edited	Fires when a project's attribute changes, like name, description, due date, etc.
project.state	Fires when project state changes.
asset.uploaded	Fires when an asset or a new version of an existing asset is uploaded.
asset.deleted	Fires when asset deleted.
refdoc.uploaded	A new reference document uploaded.
refdoc.deleted	An existing reference document is deleted.
annotation.added	An annotation to asset is created.
annotation.edited	An annotation is edited.
annotation.deleted	An annotation is deleted.
task.created	A new task of any type is created.
task.completed	A task is marked as completed.
task.deleted	A task is deleted.

<code>task.approved</code>	An asset review task marked as approved.
<code>task.rejected</code>	An asset review task marked as rejected.
<code>webhook.test</code>	Dummy event object for the testing endpoint.

## **project.created**

Fires when a new project is created.

```
{
  "EventId": "XppRxqUyNEqYrZtY9lRBdA",
  "EventType": "project.created",
  "Created": "2021-02-18T23:22:33.2944308Z",
  "Data": {
    "ProjectUID": "B0CADA6D72824F85A38BE8471503D204", // Unique ID of a newly create
    "Name": "ProjectName plain text", // Project name.
    "Project": { // Here is a Project entity as
                  // in the GET /api/v1/project
    [...]
    "ProjectUID": "XXXXXXXXXXXX",
    "ProjectState": "OnHold",
    "Name": "string",
    "Customer": "string",
    "Project": "string",
    "Design": "string",
    "Revision": "string",
    "Description": "string",
    "Tags": [
      "string",
      "string"
    ],
  },
}
```

```

        "ReviewStatus": {
            "PendingCount": 1,
            "ApprovedCount": 2,
            "RejectedCount": 3
        },
        "Created": "2021-04-29T11:56:44.690266Z",
        "Owners": [
            "YYYYYYYYYYYYY0",
            "YYYYYYYYYYYYY1"
        ],
        "Client": {
            "ClientUID": "ZZZZZZZZZZZZ",
            "Name": "string"
        }
    }
}

```

## project.edited

Fires when one of the project's attributes is changed: `Name`, `Customer`, `Project`, `Design`, `Revision`, `Description`, `Tags`, `DueDate`, `ProjectOwners`.

```

{
    "EventId": "xDPuRZ1F_Ea03f0-WPZqnQ",
    "EventType": "project.edited",
    "Created": "2021-02-18T23:47:17.7572002Z",
    "Data": {
        "ProjectUID": "XXXXXXXXXXXXXXXXXXXX", // Unique ID of a project where t
        "Attribute": "Name", // Attribute name, see the list a
        "Value": "New project name", // New attribute value.
        "Project": { // Here is a Project entity as it
                        // in the GET /api/v1/project met
    }
}

```

```

        [...]
    }
}
}

```

## project.state

Fires when the project's state is changed, interactively or automatically by Approval Studio itself.

```

{
  "EventId": "JvuKXVIPDE-V7wDZ9wPt9A",
  "EventType": "project.state",
  "Created": "2021-02-18T23:55:28.00321Z",
  "Data": {
    "ProjectUID": "XXXXXXXXXXXXXXXXXXXX", // Unique ID of a project.
    "State": "Active|OnHold|Completed|InTransit|Archived", // New project's state.
    "Project" : { // Here is a Project entity
                  // in the GET /api/v1/project endpoint
    [...]
  }
}

```

## asset.uploaded

Fires when an asset (or an asset version) is uploaded and successfully processed. If asset processing fails due to any possible reason, an event is not sent.

```

{
  "EventId": "inuBFhnHn0CUZ4u400-8Ww",
  "EventType": "asset.uploaded",

```

```

    "Created": "2021-02-19T12:57:53.9413529Z",
    "Data": {
      "ProjectUID": "XXXXXXXXXXXXXXXXXXXX",
      "AssetUID": "YYYYYYYYYYYYYYYYYYYY",
      "Name": "filename.png",
      "Project" : {                                     // Here is a Project entity as it is
                                                         // in the GET /api/v1/project method.

        [...]

      }
    }
  }
}

```

## asset.deleted

Fires when an asset (or an asset version) is deleted.

**Note:** when an asset is deleted, all the related objects – tasks, comments, attachments – are deleted as well, but you will not get separate webhook calls for them.

```

{
  "EventId": "__Z6LC1FFkyxEPEVx-ytQ",
  "EventType": "asset.deleted",
  "Created": "2021-02-19T13:12:39.2669051Z",
  "Data": {
    "ProjectUID": "XXXXXXXXXXXXXXXXXXXX",
    "AssetUID": "YYYYYYYYYYYYYYYYYYYY",
    "Project" : {                                     // Here is a Project entity as it is
                                                         // in the GET /api/v1/project method.

      [...]

    }
  }
}

```

## refdoc.uploaded

Fires when a reference document is uploaded and processed successfully.

```
{
  "EventId": "q2bwNSmAoU00wXMVTyn7Cg",
  "EventType": "refdoc.uploaded",
  "Created": "2021-02-19T13:33:57.6370653Z",
  "Data": {
    "ProjectUID": "XXXXXXXXXXXXXXXXXXXX",
    "RefDocUID": "YYYYYYYYYYYYYYYYYYYY",
    "Name": "refdocument.docx",
    "Project" : {
      // Here is a Project entity as it is
      // in the GET /api/v1/project method.
    }
  }
}
```

## refdoc.deleted

Fires when a reference document is deleted.

```
{
  "EventId": "AT4ShAVLN0Ck7kVhMHYzUw",
  "EventType": "refdoc.deleted",
  "Created": "2021-02-19T13:37:59.3381091Z",
  "Data": {
    "ProjectUID": "XXXXXXXXXXXXXXXXXXXX",
    "RefDocUID": "YYYYYYYYYYYYYYYYYYYY",
    "Project" : {
      // Here is a Project entity as it is
      // in the GET /api/v1/project method.
    }
  }
}
```

```

        [...]
    }
}
}

```

## annotation.added

Fires when an annotation is added to an asset.

This can be a high-level annotation associated with the selected region on the image, or a comment (to an annotation).

```

{
  "EventId": "Vx5S8r4os0K5WlwYFsYRhg",
  "EventType": "annotation.added",
  "Created": "2021-02-19T13:45:24.7158494Z",
  "Data": {
    "ProjectUID": "XXXXXXXXXXXXXXXXXXXX",
    "AnnotationUID": "YYYYYYYYYYYYYYYYYYYY",
    "AssetUID": "AAAAAAAAAAAAAAAAAAAA",
    "Text": "Annotation text",
    "Project": {
      // Here is a Project entity as it is
      // in the GET /api/v1/project method.
      [...]
    }
  }
}

```

## annotation.deleted

Fires when an annotation or comment to an annotation is deleted.

```

{
  "EventId": "HdBv5m13CEaZPBAr4E0AIw",
  "EventType": "annotation.deleted",
  "Created": "2021-02-19T14:51:23.9392566Z",
  "Data": {
    "ProjectUID": "XXXXXXXXXXXXXXXXXXXX",
    "AnnotationUID": "YYYYYYYYYYYYYYYYYYYY",
    "AssetUID": "AAAAAAAAAAAAAAAAAAAA",
    "Project" : {                                // Here is a Project entity as it is
                                                // in the GET /api/v1/project method.

    [...]

  }
}

```

## task.created

Fires when a task is created. Only a task ID is passed to an Event object.

```

{
  "EventId": "QbLNa0r3UkuLPJ3KZpoDNg",
  "EventType": "task.created",
  "Created": "2021-02-19T15:10:19.7757392Z",
  "Data": {
    "TaskUID": "XXXXXXXXXXXXXXXXXXXX",
    "Project" : {                                // Here is a Project entity as it i
                                                // in the GET /api/v1/project method.

    [...]

  },
  "Task" : {                                    // Here is a Task entity as it is
                                                // in the GET /api/v1/task.

    "taskUID": "XXXXXXXXXXXX",
    "projectUID": "YYYYYYYYYYYY",

```



```

        "projectName": "string",
        "clientUID": "ZZZZZZZZZZZZ",
        "type": "UploadAssets|UploadRefDocs|ReviewAssets|ExternalReviewAssets|U
        "status": "Pending|Closed|Approved|Rejected|ApprovedWithChanges",
        "created": "2020-11-27T15:39:59.864882",
        "closed": "2020-11-27T15:40:51.407384",
        "user": {
            "userID": "92AFE33153124F0980E43EF80133FE9B",
            "fullName": "John Smith",
            "email": "john.smith@email.com"
        },
        "reviewUrl": string, // URL to launch prooftool for giv
                                // Appears only for ReviewAsset ta
        "requestedAssetName": "filename.pdf" // Name of the asset, a task refer
                                                // Appears only in UploadC
    }
}

```

## task.deleted

Fires when a task is deleted, interactively, or through API.

```

{
    "EventId": "0Jd-zSo0KU2_PoG6zSb5uQ",
    "EventType": "task.deleted",
    "Created": "2021-02-19T15:20:58.5833087Z",
    "Data": {
        "TaskUID": "XXXXXXXXXXXXXXXXXXXX",
        "Project": { // Here is a Project entity as it i
                    // in the GET /api/v1/project method.
        }
    }
}

```

```

    },
    "Task" : {
        // Here is a Task entity as it is
        // in the GET /api/v1/task.

        [...]

    }
}

```

## task.approved

Fires when an asset is approved in a proof tool or using API.

TaskUID is an ID of an asset review task associated with a proof tool session.

```

{
    "EventId": "0Jd-zSo0KU2_PoG6zSb5uQ",
    "EventType": "task.deleted",
    "Created": "2021-02-19T15:20:58.5833087Z",
    "Data": {
        "TaskUID": "XXXXXXXXXXXXXXXXXXXX",
        "Project" : {
            // Here is a Project entity as it i
            // in the GET /api/v1/project method.

            [...]

        },
        "Task" : {
            // Here is a Task entity as it is
            // in the GET /api/v1/task.

            [...]

        }
    }
}

```

## task.rejected

Fires when an asset is rejected in a proof tool or using API.

TaskUID is an ID of an asset review task associated with a proof tool session.

```
{
  "EventId": "0Jd-zSo0KU2_PoG6zSb5uQ",
  "EventType": "task.deleted",
  "Created": "2021-02-19T15:20:58.5833087Z",
  "Data": {
    "TaskUID": "XXXXXXXXXXXXXXXXXXXX",
    "Project" : {
      // Here is a Project entity as it is
      // in the GET /api/v1/project method.
      [...]
    },
    "Task" : {
      // Here is a Task entity as it is
      // in the GET /api/v1/task.
      [...]
    }
  }
}
```

## webhook.test

Fires when API method **PUT /api/v1/webhook/test** is invoked. It works like any other real event including a secret UID in an HTTP header `Api-Secret` .

```
{
  "EventId": "0Jd-zSo0KU2_PoG6zSb5uQ",
  "EventType": "webhook.test",
  "Created": "2021-02-19T15:20:58.5833087Z",
  "Data": {
    "DummyData": "random string"
  }
}
```

```
}  
}
```

# Webhooks management

---

Method/Path	Description
GET /api/v1/webhooks	Returns all webhooks for the current user's tenant.
POST /api/v1/webhook	Sets us a new webhook.
DEL /api/v1/webhook	Deletes a webhook.
PUT /api/v1/webhook/test	Test a webhook endpoint.

## GET /api/v1/webhooks

---

Returns all the installed webhooks.

**Curl:**

```
curl -X GET "https://api.approval.studio/api/v1/webhooks"  
  -H "accept: text/plain"  
  -H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> List of webhooks.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": [
    {
      "webhookUID": "XXXXXXXXXXXXXXXXXXXX01",
      "url": "http://xxxxx.com/part/of/url01",
      "secret": "YYYYYYYYYYYYYYYYYYYY01",          // This secret will be sent with every POST
      "created": "2021-02-09T09:17:27.904911"      // to ensure that this is API host is post
    },
    {
      "webhookUID": "XXXXXXXXXXXXXXXXXXXX02",
      "url": "http://xxxxx.com/part/of/url02",
      "secret": "YYYYYYYYYYYYYYYYYYYY02",
      "created": "2021-02-09T09:17:27.904911"
    }
  ]
}
```

## POST /api/v1/webhook

---

Sets up a new webhook.

### Request

```
{
  "url": "string",          // Valid URL string, http or https.
```

```
"clientId": "string", // Client UID, optional.  
"eventType": "string" // One of the event type names, optional.  
}
```

**clientId** is an optional client identifier. Events are client-wide, so in the case of multitenancy, when a user belongs to multiple clients, the events through webhooks are produced based on the client you provided when creating a webhook. You need to install multiple webhooks to get events from all tenants you belong to.

If no clientId is provided, the first client in the list is used.

**eventType** is an optional event type, please see [Event object](#) When provided, it limits the webhook to be invoked only when the given event type occurs. If it's not provided, all events will be sent to the webhook and in this case an API consumer is responsible for filtering events (when necessary).

**Note:** The API host might need up to ~1 minute to start processing a newly installed webhook.

**Curl:**

```
curl -X POST "https://api.approval.studio/api/v1/webhook"  
-H "accept: text/plain"  
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"  
-H "Content-Type: application/json-patch+json"  
-d "{\"url\":\"http://xxxx.com/part/of/url\", \"clientId\": \"XXXX\", \"eventType\":
```

## Responses

HTTP Code	Response
200	<b>Success.</b> A newly created webhook instance returned.

```
{  
  "version": "1.0",  
}
```

```
"statusCode": 200,
"message": "POST Request successful.",
"result": {
  "webhookUID": "XXXXXXXXXXXXXXXXXXXX01", // Newly assigned webhook ID.
  "url": "http://xxxx.com/part/of/url",   // Url of the webhook, as in the request.
  "secret": "YYYYYYYYYYYYYYYYYYYY01",    // This secret will be sent with every POST i
  "created": "2021-02-09T09:17:27.904911" // to ensure that this is API host is posting
                                           // see HTTP response header "Api-Secret".
}
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
412	<b>Error:</b> Webhook with the given URL is already registered for this tenant. API does not allow to setup duplicates.

## DELETE /api/v1/webhook

Deletes a webhook with the given ID.

Note that deletion of a webhook might need up to ~1 minute for API host to update.

**Curl:**

```
curl -X DELETE "https://api.approval.studio/api/v1/webhook"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"
```

```
-H "Content-Type: application/json-patch+json"
-d "{\"webHookUID\":\"XXXXXXXXXXXXXXXXXXXX\"}"
```

## Responses

HTTP Code	Response
200	<b>Success.</b> Webhook with the given ID deleted.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Webhook is successfully deleted."
}
```

HTTP Code	Response
400	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
404	<b>Error:</b> Webhook with the given ID not found or was already deleted.

## PUT /api/v1/webhook/test

Test a webhook's endpoint with dummy data in form of JSON:

```
{
  "EventId": "0Jd-zSo0KU2_PoG6zSb5uQ",
}
```



```
      "EventType": "webhook.test",
      "Created": "2021-02-19T15:20:58.5833087Z",
      "Data": {
        "DummyData": "random string"
      }
    }
  }
```

When posting test data, the API host uses the same retry logic as it is for real calls, i.e. tries to re-deliver it in case of failure, so multiple calls are expected if the endpoint fails to accept the call instantly.

#### Curl:

```
curl -X PUT "https://api.approval.studio/api/v1/webhook/test"
-H "accept: text/plain"
-H "Authorization: Bearer YYYYYYYYYYYYYYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d "{\"webHookUID\":\"XXXXXXXXXXXXXXXXXXXXX\"}"
```

#### Responses

HTTP Code	Response
200	<b>Success.</b> Webhook is tested, see your endpoint logs.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Webhook is tested, see your endpoint logs."
}
```

HTTP Code	Response
-----------	----------

<b>400</b>	<b>Error:</b> Parameters' validation failed.
	See HTTP code 400 description.
<b>404</b>	<b>Error:</b> Webhook with the given ID not found.