API Reference

Table of contents

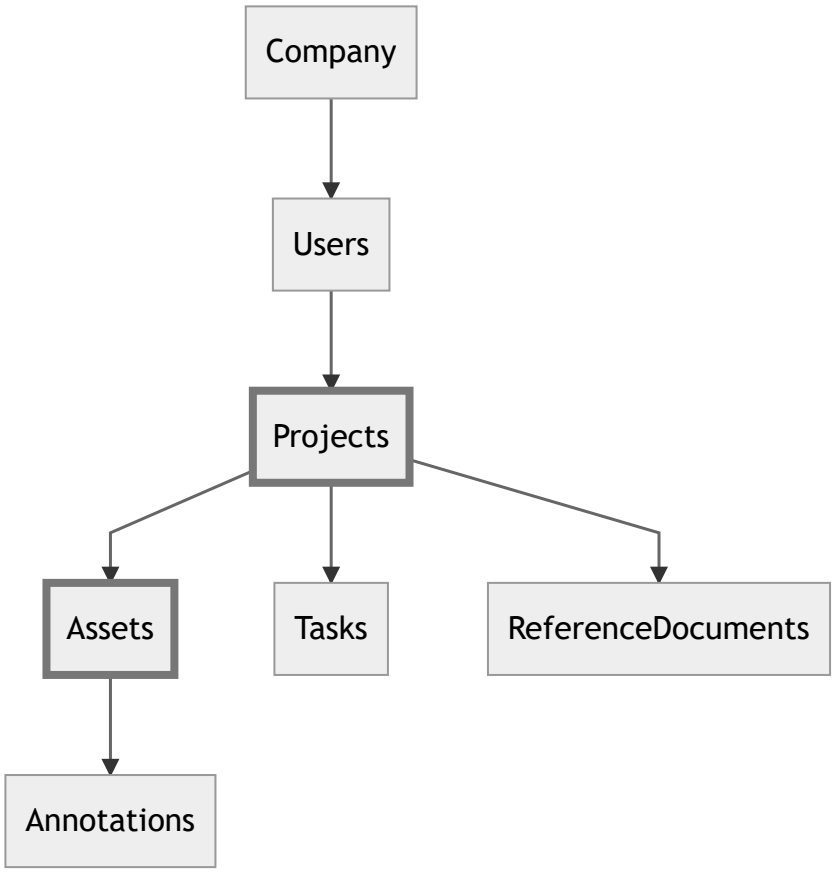
General

This is a REST API to the Approval Studio, a review and packaging SAAS.
API utilizes its flow, authorization , storage and so on.

Approval Studio basic concepts

From business point of view, Approval Studio is based on a multitenant concept when a single user may by 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	**Success**. See method description to get what and how method returns.
400	**Validation error**. API validates input parameters and when finds invalid parameter, throws HTTP code 400. Those errors are related to parameters' presence, format, emptiness etc. Validating against database, like looking for data by a ID is conducted separately and reflected in case of error, in HTTP codes 404, 406, 412 etc., please see methods 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**	**Resource not found**. Generally it means that requested resources either not found and somehow restricted to proceed. For example, a given project id is non-existent, the project not found so gets HTTP code 404.
429	**Rate Limit Reached**. API host calculates number of calls per sec, minute and hour, and when rate of requests reaches limit, throws HTTP code 429 that means that the API host is overloaded and 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 request remaining
X-Rate-Limit-Reset: UTC date time (ISO 8601) when the limits resets
```

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

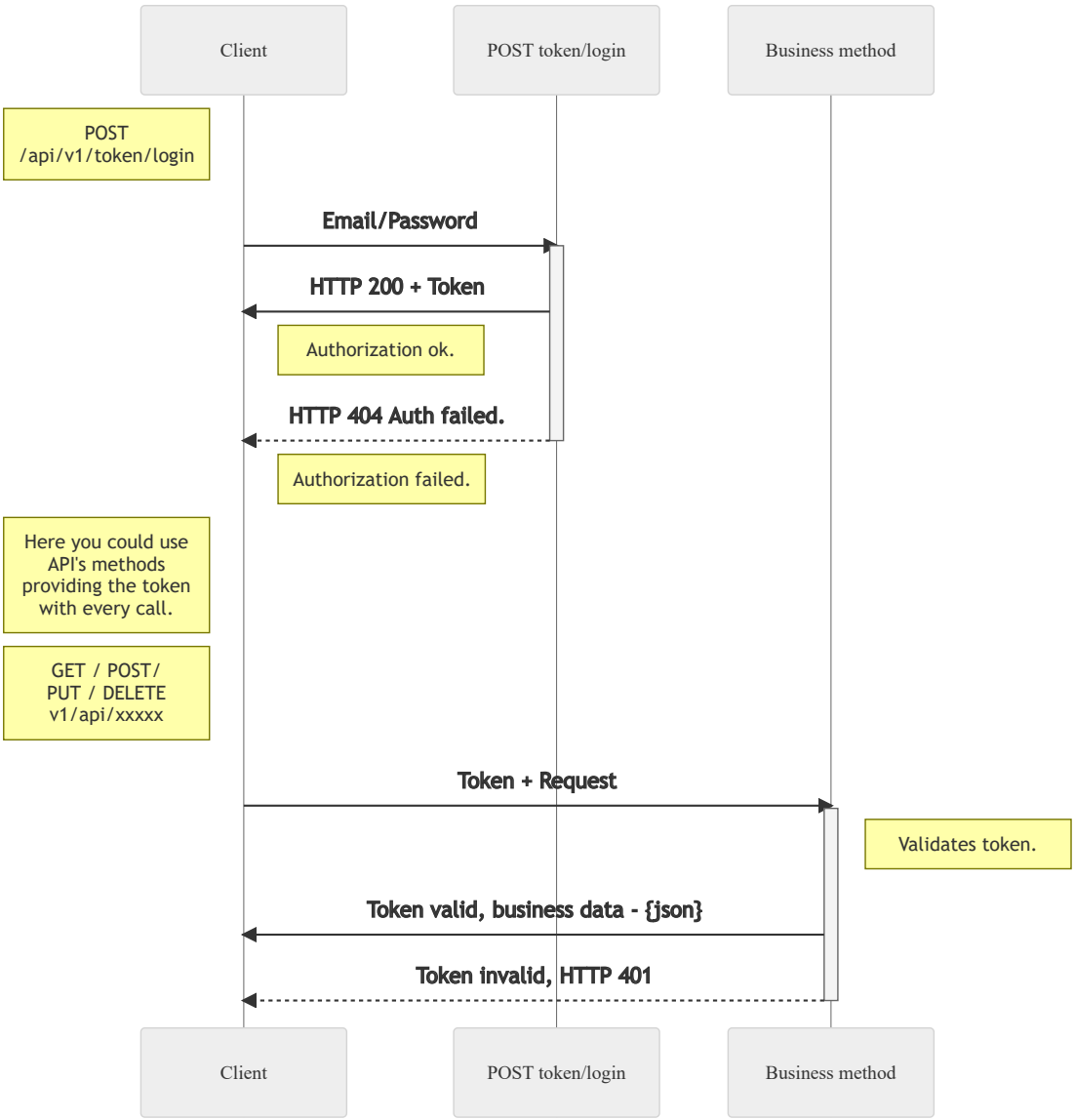
Authorization/authentication

The API is based on **authorization token** which should be requested prior to using any available API methods.
The flow is 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 a **HTTP header** like this: `Authorization: Bearer XXXXXXXXXX`.

The token is valid for a limited amount of time, **120 minutes** by default. When expired and still used you will have 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 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 returns an **authorization token** which must be supplied to every API as a header, like `"Authorization: Bearer YYYYYYYYYYYYYYYYY..."`.

Request

```
{
  "userName": "joshn.smith@gmail.com",
  "password": "Q@fG%^18_A"
}
```

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_
```

Responses

HTTP Code	Response
200	**Success**. Login successful, AUTH token provided.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "POST Request successful.",
  "result": {
    "token": "eyJhbGciOiJIUzI1NiIsI[...]", // This is the auth token.
    "status": "Success",
    "user": {
      "userUID": "XXXXXXXXXXXXXXXXXXXXXXXXXXX",
      "fullName": "John Smith",
      "email": "john.smith@gmail.com",
      "role": "RegularUser|Administrator"
    }
  }
}
```

HTTP Code	Response

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See **HTTP code 400 description**.

HTTP Code	Response
404	**Error**: User with given email and password not found. Either the provided credentials are wrong or a user is locked and is no able to login anymore.

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

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.

Name		Explanation
`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 project, optional, max 20 elements of 20-chars tags.
`dueDate`		Optional date, UTC, that points 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 when the project was created, UTC.

Method/Path	Description
`GET /api/v1/project`	Getting list of projects.
`POST /api/v1/project`	Create new project.
`PUT /api/v1/project`	Edit existing project.
`DELETE /api/v1/project`	Delete a project.
`GET /api/v1/project/proofreport`	Get a link to a proof report for a project.
`PUT /api/v1/project/state`	Change project's state.

GET /api/v1/project

Returns one or more projects and projects' assets, tasks, reference documents – depends on the parameters passed.
All the parameters are optional; method always uses **AND** combination of all the parameters.

Parameter	Type & Explanation
`ProjectUID`	`string` Unique project GUID. If provided, only one project will retrieved.
`States`	`string` One or more project states, comma-separated, see *ProjectStates* above. Default value is `Active,OnHold,Completed`.
`Query`	`string` A free-form text to case-sensitive search in projects' attributes: `Project name`, `Customer`, `Project`, `Design`, `Revision`, `Description`, `Tags`.
`IsLoadAssets`	`boolean`, `false` by default. if set, returns assets for every project as a child collection.
`IsLoadLastVerAssets`	`boolean`, `true` by default. If set, only the most recent version of every asset be returned. Ignored, if `IsLoadAssets` is not set.
`IsLoadTasks`	`boolean`, `false` by default. Is set, list of active tasks for every project would be returned as a child collection.
`IsLoadRefDocs`	`boolean`, `false` by default. Is set, list of uploaded reference documents for every project would be returned as a child collection.

Request URL

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

Curl:

```
curl -X GET "https://api.approval.studio/api/v1/project?
             ProjectUID=XXXXXXXXXXXXXXXXXXX&States=Active%2COnHold%2CComple
             Query=Some%20Free%20Text&IsLoadAssets=true&
```

```
            IsLoadLastVerAssets=true&IsLoadTasks=true&IsLoadRefDocs=true"
    -H  "accept: text/plain" \
    -H  "Authorization: Bearer YYYYYYYYYYYYY"
```

- Default `States` is `Active,OnHold,Completed`.
- If `ProjectUID` is provided and no project found, an empty list returns and 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 sta
                                         // (see proofing flow explanation
          "pendingCount": 1,             // The asset has 1 uncompleted re
          "approvedCount": 2,            // The asset has been approved 2
          "rejectedCount": 3             // The asset has been rejected 3
        },
    "created": "2020-11-22T23:00:53.425Z",
    "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
              "pendingCount": 1,               // The asset has 1 uncomple
              "approvedCount": 2,              // The asset has been appro
              "rejectedCount": 3               // The asset has been rejec
          },
          "pagesCount": int, >=1,
```

```
      "created": "2020-11-22T23:00:53.425Z",
      "fileSize": int, bytes,
      "reviewUrl": string,                  // URL to a prooftool to vi
      "thumbnailUrl": string                // URL to asset's thumnail
    }
  ],
  "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"
    }
  ],
  ...
  ]
},
"tasks": [
  {
    "taskUID": "GUID",
    "type": "UploadAssets|UploadRefDocs|ReviewAssets|ExternalReviewAsse
    "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 prooftool fo
                                         // Appears only for ReviewAss
  }
],
```

```
    "refDocs": [
      {
        "refDocGUID": "GUID",
        "created": "2020-11-22T23:00:53.425Z",
        "name": "filename.ext",
        "fileSize": int, bytes.
      }
    ]
  }
]
```

POST /api/v1/project

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

Request

Field	Type & Explanation
`clientUID`	`string[50]` Optional client UID. See GET /api/v1/users / Responses. If no client ID is provided, the first client will be choosen 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.

Field	Type & Explanation
`dueDate`	`ISO date` Optional UTC date (or date-time) that point to a date when project supposed to be completed. The date affects 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 p
    "XXXXXXXX"
  ]
}
```

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

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"
}
```

HTTP Code	Response
400	**Error**: 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": [
        "UserUID","UserUID" ...
      ]
    }
```

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

```
{
    "projectUID": "XXXXXXXXXX", // ID of project to edit
    "projectName": "New name"
}
```

Curl:

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

Responses

HTTP Code	Response
200	**Success**. 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
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: 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	**Error**: Project's editing is possible only when the project is `Active` or `OnHold`. `Completed`, `Archived` or `InTransit` projects are not mutable. Please see PUT /api/v1/project/state.

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

HTTP Code	Response
412	**Error**: Project must be in state **Completed** or **Archived**. 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 project is deleted disappears from a list of projects; assets and uploaded reference documents are deleted as well.

Project should have status **Completed** or **Archived** to be deleted; a 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 YYYYYYYYYYYYYYYYY..." \
     -H  "Content-Type: application/json" \
     -d "{\"projectUID\":\"XXXXXXXXXXXXX\"}"
```

Responses

HTTP Code	Response
200	**Success**. Project deleted.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response

HTTP Code	Response
404	**Error**: Project with the given ID not found and therefore project deteion failed.

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

HTTP Code	Response
412	**Error**: Project must be either **Completed** or **Archived** 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

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

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

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

Responses

HTTP Code	Response
200	**Success**. Proof report URL generated.

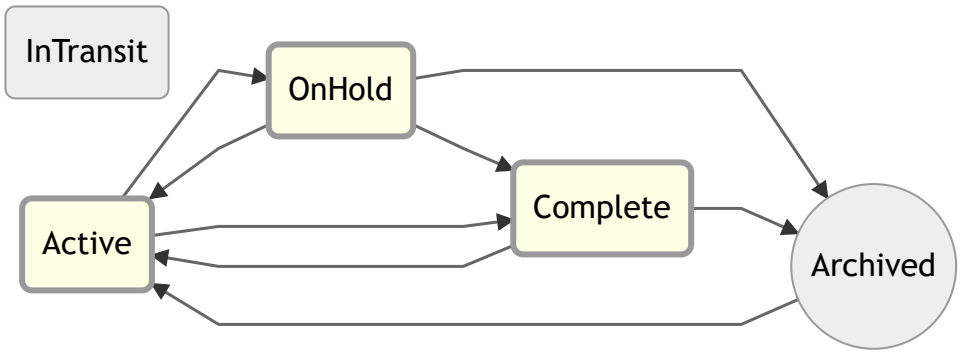

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "downloadURL": "https://approval.studio/ProofApi/GetProofReport/DDDDDDD
    "project": {
      "projectUID": "3AB3A5F58951467B975378198C7265D1",
      "projectState": "Complete",
      "name": "Project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


HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.
HTTP Code	Response
----------	----------
404	**Error**: 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 state rules:



The diagram above explains pre-requisite check for changing project's state.
Note: Detailed description of the project management is out of scope of this document.

States

Those are the states:

State	Explanation
Active	Work on a project is undergo. This is where all the review work is going on.
OnHold	Due some reason you want to postpone working on a project. Setting state to **OnHold** 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.
Completed	When setting state to **Completed** all projects task are deleted and the project itself moves from dashboard to a separate screen where all completes stored separately. All the assets, reference documents are preserved as well as history etc. The project goes read-only.
Archived	This is generally the same as **Completed** but assets and reference documents moved to a remote, slow storage. Process of moving files to other storage could take a while, so when it is going on, the system marks a project as **InTransit** (see below).
InTransit	Project is locked to **InTransit** state by API itself when it is switching to **Archive** state or, vice versa, from **Archived** to **Active**. You can't set this state forcedly.

Request

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

Responses

HTTP Code	Response
200	**Success**. Project's state changed.

```
{
  "projectUID": "XXXXXXXXXX",
  "projectState": "Active|OnHold|Complete|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	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: 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	**Error**: Project with the given ID is already have requested state; no updating made.


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


HTTP Code	Response
412	**Error**:Please see the diagram above.
	You cannot change state arbitrarily, for instance you can not change from **Active** to **Archived**. To move project to archive, you need to make it **Complete** first, which forces project cleaning, and then change the state to **Archive**.


```
{
  "version": "1.0",
  "statusCode": 412,
  "message": "You cannot directly change state from {XXXX} to {YYYY}. Pleas
}
```


Asset management

Method/Path	Description
`GET /api/v1/assets`	Gets an asset instance.

Method/Path	Description
`DELETE /api/v1/assets`	Deletes an asset.
`POST /api/v1/assets/upload`	Uploads an asset.
`GET /api/v1/assets/download`	Downloads an asset.
`GET /api/v1/assets/proofreport`	Get 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 "
```

Responses

HTTP Code	Response
200	**Success**. Asset instance returned.

```
"version": "1.0",
"statusCode": 200,
"message": "GET Request successful.",
"result": {
  "assetUID": "xxxxxxxxxx",
  "version": 1,
  "name": "drawing.pdf",
  "status": "Processed",
  "reviewStatus": {                                  // Proof review statu
     "pendingCount": 1,                              // The asset has 1 un
     "approvedCount": 2,                             // The asset has been
     "rejectedCount": 3                              // The asset has been
```

```
    },
    "pagesCount": 3,
    "created": "2020-10-22T08:09:05.778512",
    "fileSize": 1063800,
    "reviewUrl": "https://app.approval.studio/xxx",   // URL to launch proc
    "thumbnailUrl": "https://app.approval.studio/yyy" // URL to asset's thu
  }
}
```

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: 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 unrecoverable operation that leads to removing the asset from project and deleting a file from storage.
It is no way to restore asset after it's 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\":\"XXXXXXXXXXX\"}"
```

Responses

HTTP Code	Response
200	**Success**. Asset deleted.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.

HTTP Code	Response
404	**Error**: 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 asset processing queue**, which runs asynchronously. After uploading you **need to pool asset status** to get know when it is processed or failed to process.

You can **upload assets directly** to a selected project or **point an upload task** as a 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 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 number of other options. Please refer company's site or/and support to learn more.

Request

Parameter	Type & Explanation
ProjectUID	`string` Unique project ID, **mandatory**.
TaskUID	`string` Unique project ID, **optional**.
	Note: When provided, it points to a task this upload is made for. In other words, **UploadAsset** or **UploadChangedAsset** tasks are executed using this method.
FileName	`form file` Asset file name, **mandatory**. Note: It's just name, not a path.

Curl:

```
curl -X POST "http://api.approval.studio/api/v1/asset/upload?ProjectUID=XXX
     -H "accept: text/plain"
     -H "Authorization: Bearer YYYYYYYYYYYYYY"
     -H "Content-Type: multipart/form-data"
     -F "uploadedFile=assetfilename.jpg;type=image/jpeg"
```

Responses

HTTP Code	Response
200	**Success**. "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 catch when it'd ready to use.",
    "result": {
    "assetUID": "AAAAAAAAAAAAAAAA" // newly generated asset id.
  }
}
```

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Project UID is either invalid or points to an non-existing or inactive project.
404	**Error**: Task UID is either invalid or points to an non-existing or inactive task.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Project UID provided is either invalid or points to an non-ex
             "Task UID provided is either invalid or points to an non-exist
}
```

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

HTTP Code	Response
	Note: **UploadChangedAsset** means that the project owner(s) requested a new version of 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 allow
             "File to upload must the the same type as the original file (.
}
```

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=XXXX
     -H  "accept: text/plain"
     -H  "Authorization: Bearer YYYYYYYYYYYYYYYYYY"
```

Responses

HTTP Code	Response
200	**Success**. URL generated.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "downloadURL": "https://approval.studio/ProofApi/DownloadAsset/ZZZZZZZZ
    "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	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: 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

Parameter	Type & Explanation
AssetUID	`string` Unique project ID, **mandatory**.

Curl:

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

Responses

HTTP Code	Response
200	**Success**. URL generated.


```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "downloadURL": "https://approval.studio/ProofApi/GetProofReport/ZZZZZZZ
    "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	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response

HTTP Code	Response
404	**Error**: Asset with the given ID not found.

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

Task management

Method/Path	Description
`GET /api/v1/task/all`	Gets list of tasks assigned on 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 list of tasks optionally filtered by task types, assigned to a currently logged in user. This is what a user see in Approval Studio web application in the list of tasks in the dashboard.

Parameter	Type & Explanation
`Types`	`string` One or more task types, comma-separated.

Parameter	Type & Explanation
	Default value is `UploadAssets, UploadRefDocs, ReviewAssets, ExternalReviewAssets, UploadChangedAsset, UploadVideo, ExternalReviewVideo, ReviewVideo.`
	If no parameter 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	**Success**. Task instance returned.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": [
    {
        "taskUID": "XXXXXXXXXXX1",
        "projectUID": "YYYYYYYYYYYY1",
        "projectName": "Project name, string",
        "type": "UploadAssets|UploadRefDocs|ReviewAssets|ExternalReviewAsse
        "status": "Pending|Closed|Approved|Rejected",
        "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 proof
                                                       // Appears only in Rev
        "requestedAssetName": "butterfly_poster.pdf" // Name of the asset,
                                                       // Appears only in Upl
```

```
    },
    {
        "taskUID": "XXXXXXXXXX2",
        "projectUID": "YYYYYYYYYYY2",
        ...
    },
    ...
  ]
}
```

HTTP Code	Response
400	**Error**: 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=XXXXXXXXXXXXXXX
     -H "accept: text/plain"
     -H  "Authorization: Bearer YYYYYYYYYY"
```

Responses

HTTP Code	Response
200	**Success**. Task instance returned.

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

```
  "result": {
    "taskUID": "XXXXXXXXXXX",
    "projectUID": "YYYYYYYYYYYY",
    "projectName": "string",
    "type": "UploadAssets|UploadRefDocs|ReviewAssets|ExternalReviewAssets|U
    "status": "Pending|Closed|Approved|Rejected",
    "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 for ReviewAsset ta
    "requestedAssetName": "filename.pdf" // Name of the asset, a task refer
                                         // Appears only in UploadChangedAs
  }
}
```

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: 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 YYYYYYYYYYYYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d "{\"taskUID\":\"XXXXXXXXXXX\"}"
```

Responses

HTTP Code	Response
200	**Success**. Login successful, AUTH token provided.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Task with the given ID not found.

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

HTTP Code	Response

HTTP Code	Response
410	**Error**: 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 can not 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": "XXXXXXXXXXXXXX", // Project this task belongs to, mandator
  "userUID": "ZZZZZZZZZZZZZ",     // 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/asset_upload" \
     -H "accept: text/plain" \
     -H "Authorization: Bearer YYYYYYYYYYYYYYYY" \
     -H "Content-Type: application/json-patch+json" \
     -d "{\"projectUID\":\"XXXXXXXXXXXXXXX\",
          \"userUID\":\"ZZZZZZZZZZZZZ\",
          \"dueDate\":\"2020-12-04T13:11:11.526Z\",
          \"comment\":\"comment text\"}"
```

Responses

HTTP Code	Response
200	**Success**. Task created.


```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "Task created."
  "result": {
    "task": {
      "taskUID": "XXXXXXXXXXXXXXXXXXXXX",
      "projectId": 981,
      "projectUID": "YYYYYYYYYYYYYYYYYYYYYYYYYYYY",
      "type": "UploadAssets",
      "status": "Pending",
      "comment": "Free text comment to the task",
      "dueDate": "2020-12-22",
      "created": "2020-12-22",
      "userUID": "ZZZZZZZZZZZZZ",
      "userName": "John Smith",
      "userEmail": "john.smith@gmail.com"
    }
  }
}
```


HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Given project not found.


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

HTTP Code	Response
412	**Error**: You can not create a task for a project in state [Complete]. Only projects that are `Active` or `OnHold` can have tasks.
412	**Error**: User with UID [ZZZZZZZZZZZZZ] not found.

```
{
  "version": "1.0",
  "statusCode": 412,
  "message": "You can not 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": "XXXXXXXXXXXXXX", // Project this task belongs to, mandator
  "userUID": "ZZZZZZZZZZZZZ",     // 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\":\"XXXXXXXXXXXXXXX\",
          \"userUID\":\"ZZZZZZZZZZZZZ\",
          \"assetUIDs\":[\"AAAAAAAAAA1\", \"AAAAAAAAAA2\"],
          \"dueDate\":\"2020-12-04\",
          \"comment\":\"comment text\" [...] }"
```

Responses

HTTP Code	Response
200	**Success**. Task created.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Given project not found.

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

HTTP Code	Response
412	**Error**: You can not create a task for a project in state `Complete`. Only projects that are `Active` or `OnHold` can have tasks.
412	**Error**: User with UID [ZZZZZZZZZZZZZ] not found.

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

POST /api/v1/task/review_asset

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

Request

```
{
  "projectUID": "XXXXXXXXXXXXXX", // Project this task belongs to, mandator
  "userUID": "ZZZZZZZZZZZZZ",     // User this task assign to, mandatory.
  "assetUIDs": [
    "AAAAAAAAAAAAAAA1",           // One or more assets to review, namdator
    "AAAAAAAAAAAAAAA2"
  ],
  "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 YYYYYYYYYYYYYYY" \
     -H "Content-Type: application/json-patch+json" \
     -d "{\"projectUID\":\"XXXXXXXXXXXXXXX\",
          \"userUID\":\"ZZZZZZZZZZZZZ\",
```

```
\"assetUIDs\":[\"AAAAAAAAAA1\", \"AAAAAAAAAA2\"],
\"dueDate\":\"2020-12-04\",
\"comment\":\"comment text\" [...]}"
```

Responses

HTTP Code	Response
200	**Success**. Task created.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Given project not found.
404	**Error**: Asset [AAAAAAAAAAAAAAA1] not found...

HTTP Code	Response
404	**Error**: Asset [AAAAAAAAAAAAAAA1] does not belong to project [XXXXXXXXXXXXXXX].

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

HTTP Code	Response
412	**Error**: You can not create a task for a project in state [Complete]. Only projects that are `Active` or `OnHold` can have tasks.
412	**Error**: User with UID [ZZZZZZZZZZZZZ] not found.

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

POST /api/v1/task/review_asset_ext

Creates a new External Review Asset Task for someone who doesn't have account in Approval Studio.
The flow is the following:

1. When task is created, a 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 made the URL expired.

Request

```
{
      "projectUID": "XXXXXXXXXXXXXX", // Project this task belongs to, mand
      "email": "string",              // User's email, mandatory.
      "assetUIDs": [
        "AAAAAAAAAAAAAAA1",           // One or more assets to review, namd
        "AAAAAAAAAAAAAAA2"
      ],
      "dueDate": "2020-12-04",        // Task due date, UTC, optional.
      "password": "string",           // Optional password. When provided,
                                      // before review session.
      "emailSubject": "string",       // Custome email subject line; overri
      "emailLanguage": "English",     // Email language, optional. English
                                      // English, German, French, Polish, S
      "comment": "comment text"       // Comment text, optional.
      "isAllowDownloadAssets": true,  // When true, allows user to download
      "isReadOnly": true              // When true, make a review session r
                                      // no comments, no approve/reject. Op
}
```

Curl:

```
curl -X POST "https://api.approval.studio/api/v1/task/review_asset_ext" \
     -H "accept: text/plain" \
     -H "Authorization: Bearer YYYYYYYYYYYYYYYY" \
     -H "Content-Type: application/json-patch+json" \
     -d "{\"projectUID\":\"XXXXXXXXXXXXXXX\",
          \"email\":\"ZZZZZZZZZZZZ\",
          \"assetUIDs\":[\"AAAAAAAAAA1\", \"AAAAAAAAAA2\"],
          \"dueDate\":\"2020-12-04\",
          \"comment\":\"comment text\" [...]}"
```

Responses

HTTP Code	Response
200	**Success**. Task created.

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

```
      "taskUID": "XXXXXXXXXXXXXXXXXXXXX",
      "projectId": 981,
      "projectUID": "YYYYYYYYYYYYYYYYYYYYYYYYYYYY",
      "type": "ExternalReviewAssets",
      "status": "Pending",
      "comment": "Free text comment to the task",
      "dueDate": "2020-12-22",
      "created": "2020-12-22",
      "userUID": "ZZZZZZZZZZZZZ",
      "userName": "John Smith",
      "userEmail": "john.smith@gmail.com"
    }
  }
}
```


HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Given project not found.
404	**Error**: Asset [AAAAAAAAAAAAAAA1] not found...
404	**Error**: Asset [AAAAAAAAAAAAAAA1] does not belong to project [XXXXXXXXXXXXXXX].


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


HTTP Code	Response
412	**Error**: You can not create a task for a project in state [Complete]. Only projects that are `Active` or `OnHold` can have tasks.


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

```
  "statusCode": 412,
  "message": "You can not create a task for a project in 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\":\"XXXXXXXXXXXXXXXX\"}"
```

Responses

HTTP Code	Response
200	**Success**. Task marked as completed.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Given task not found.

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

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

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

HTTP Code	Response
412	**Error**: 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 can not 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 given asset and page.

Request

Parameter	Type & Explanation
AssetUID	`string` Unique project ID, **mandatory**.
PageNum	`int` Page number, zero-based, **mandatory**.

```
curl -X GET "http://api.approval.studio/api/v1/annotation/all?AssetUID=XXXX
     -H  "accept: text/plain" \
     -H "Authorization: Bearer YYYYYYYYYYYYYYY"
```

Responses

HTTP Code	Response
200	**Success**. Annotations returned and hierarchy built.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
        {
          "0": [ // Key is the page number
            {
              "commentId": 0,
              "commentUID": "CCCCCCCCCCCCCC",
              "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
                  { annotation instance 2} [...]     // tree-like annotation
              ],
              "user": {
                "userUID": "ZZZZZZZZZZ",
                "fullName": "string",
                "email": "string"
              }
            }
          ],
          "1" : [...]
    [...]
  }
}
```

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: 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=XXX
     -H  "accept: text/plain" \
     -H "Authorization: Bearer YYYYYYYYYYYYYYYY"
```

Responses

HTTP Code	Response
200	**Success**. Annotations returned and hierarchy built.

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

```
      }
  }
}
```

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Annotation with the given ID not found.

```
{
  "version": "1.0",
  "statusCode": 404,
  "message": "Annotationnot 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 YYYYYYYYYYYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d "{\"annotationUID\":\"XXXXXXXXXX\"}"
```

Responses

HTTP Code	Response
200	**Success**. Annotation deleted.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Annotation with the given ID not found.

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

HTTP Code	Response
412	**Error**: It's only that user that created annotation or comment can delete it. You can not delete other's users' annotation 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 YYYYYYYYYYYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d "{\"annotationUID\":\"XXXXXXXXXX\"}"
```

Responses

HTTP Code	Response
200	**Success**. Annotation was hidden.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.
406	**Error**: A user can hide only his/hers own annotation.
404	**Error**: 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 YYYYYYYYYYYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d "{\"annotationUID\":\"XXXXXXXXXX\"}"
```

Responses

HTTP Code	Response
200	**Success**. Annotation was marked as completed.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.

HTTP Code	Response
404	**Error**: Annotation with the given ID not found.
406	**Error**: 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 YYYYYYYYYYYYYYYYYYYYYYYYYY"
-H "Content-Type: application/json-patch+json"
-d "{\"annotationUID\":\"XXXXXXXXXXX\"}"
```

Responses

HTTP Code	Response
200	**Success**. Annotation was marked as uncompleted.


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

```
  "message": "Annotation was uncompleted."
}
```

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.
404	**Error**: Annotation with the given ID not found.
406	**Error**: 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/users

Curl:

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

Responses

HTTP Code	Response

HTTP Code	Response
200	**Success**. List of users with companies they belong to returned.

```
{
  "version": "1.0",
  "statusCode": 200,
  "message": "GET Request successful.",
  "result": {
    "clients": [
     {
        "clientUID": "XXXXXXXXXXXXXXXXXXX",
        "name": "HiTech Service",
        "users": [
          {
            "userUID": "UUUUUUUUUUUUUUUUUU1",
            "fullName": "John Smith",
            "email": "john.smith@gmail.com",
            "role": "Administrator|RegularUser"
          },
          [...]
        ],
    },
    [...]
  }
}
```

HTTP Code	Response
400	**Error**: 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 asset accepted or rejected or when a new task created or someone made a comment or a new asset version uploaded etc.

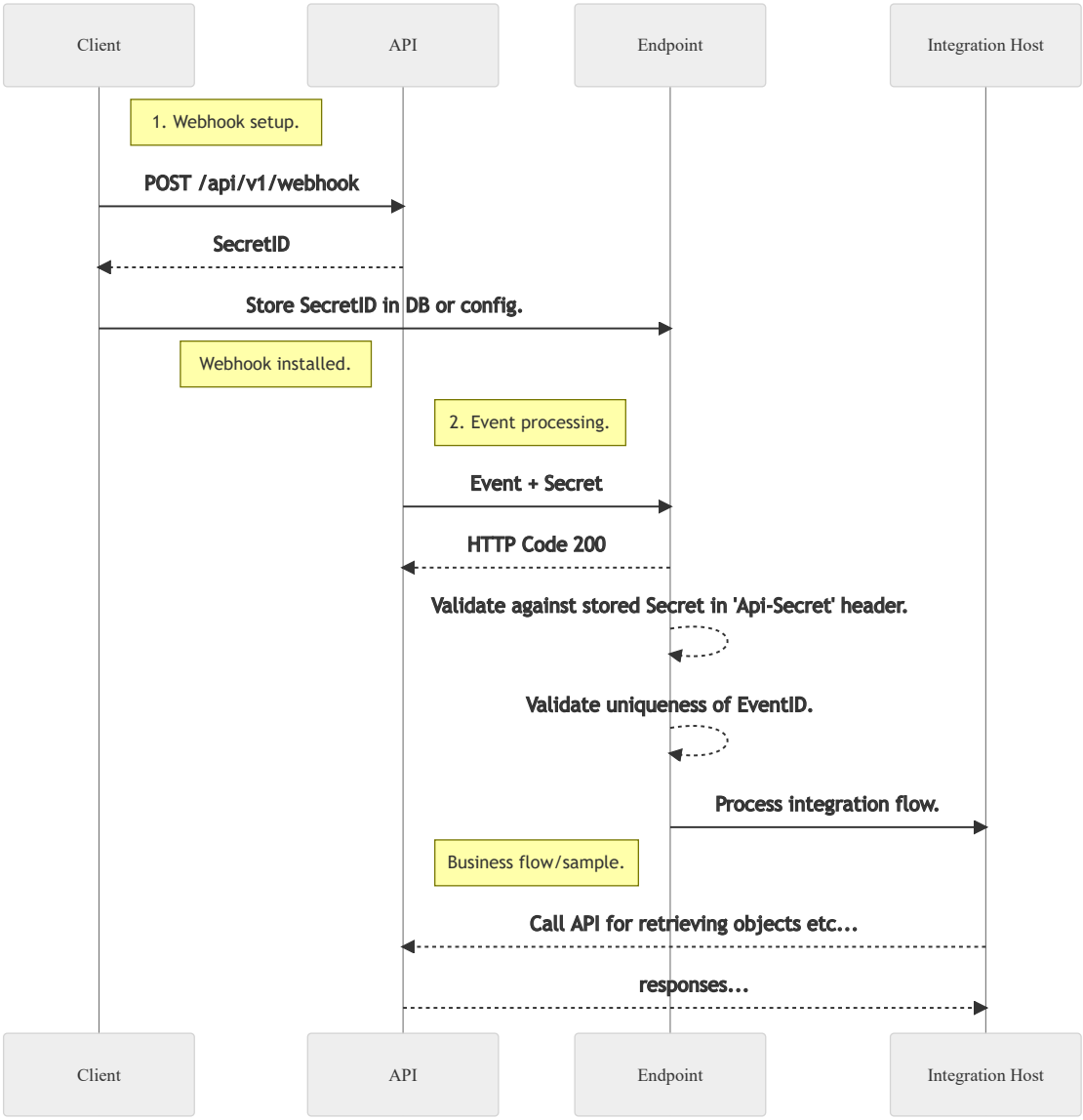
Begin using the webhooks in three steps:

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

Webhooks generally refers 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 how to build a endpoint is out of scope of this document.

The Approval Studio notifications are Event objects. This Event object contains 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.


Client
API
Endpoint
Integration Host
1. Webhook setup.
POST /api/v1/webhook
SecretID
Store SecretID in DB or config.
Webhook installed.
2. Event processing.
Event + Secret
HTTP Code 200
Validate against stored Secret in 'Api-Secret' header.
Validate uniqueness of EventID.
Process integration flow.
Business flow/sample.
Call API for retrieving objects etc...
responses...
Client
API
Endpoint
Integration Host

Security

The API host provides a **Secret ID** with every event object posted on an endpoint. 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 endpoint side
If 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

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
    "Created":"2021-02-18T14:08:08.615088Z", // UTC datetime when the event
    "Data": {                                // Data block, event-type depe
        "SomeData": "XXXXXXXX"
    }
}
```

EventID is unique for every Event object. If, for whatever reason, API host make more than one post on your endproint with the same event object, you can it by reading EventID.
Event codes are self-explanatory:

Event code	Description
`project.created`	Fires when a new project created.
`project.edited`	Fires when a project's attribute changed, like name, description, due date etc.

Event code	Description
`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.
`task.approved`	An asset review task marked as approved.
`task.rejected`	An asset review task marked as rejected.
`webhook.test`	Dummy event object for 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
        "Name":"ProjectName plain text"                  // Project name.
    }
}
```

project.edited

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

```
{
    "EventId":"xDPuRZ1F_Ea03fO-WPZqnQ",
    "EventType":"project.edited",
    "Created":"2021-02-18T23:47:17.7572002Z",
    "Data": {
        "ProjectUID":"XXXXXXXXXXXXXXXXXX",           // Unique ID of a pro
        "Attribute":"Name",                          // Attribute name, se
        "Value":"New project name"                   // New attribute valu
    }
}
```

project.state

Fires when 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":"XXXXXXXXXXXXXXXXXX",                  // Unique ID o
        "State":"Active|OnHold|Completed|InTransit|Archived" // New project
    }
}
```

asset.uploaded

Fires when an asset (or an asset version) is uploaded and successfully processed. If asset processing failed due any of possible reason, an event will not send.

```
{
    "EventId":"inuBFhnHn0CUZ4u4OO-8Ww",
    "EventType":"asset.uploaded",
    "Created":"2021-02-19T12:57:53.9413529Z",
    "Data": {
```

```
        "ProjectUID":"XXXXXXXXXXXXXXXXXX",
        "AssetUID":"YYYYYYYYYYYYYYYYYY",
        "Name":"filename.png"
    }
}
```

asset.deleted

Fires when an asset (or an asset version) is deleted.
Note: when an assets is deleted, all the related objects – tasks, comments, attachment are deleted as well, but you will not get separate webhook calls for them.

```
{
    "EventId":"__Z6LClFFkyxeEPeVx-ytQ",
    "EventType":"asset.deleted",
    "Created":"2021-02-19T13:12:39.2669051Z",
    "Data": {
        "ProjectUID":"XXXXXXXXXXXXXXXXXX",
        "AssetUID":"YYYYYYYYYYYYYYYYYY"
    }
}
```

refdoc.uploaded

Fires when a reference document is uploaded and processed successfully.

```
{
    "EventId":"q2bwNSmAoUO0wXMVTyn7Cg",
    "EventType":"refdoc.uploaded",
    "Created":"2021-02-19T13:33:57.6370653Z",
    "Data":{
        "ProjectUID":"XXXXXXXXXXXXXXXXXX",
        "RefDocUID":"YYYYYYYYYYYYYYYYYY",
        "Name":"refdocument.docx"
    }
}
```

refdoc.deleted

Fires when a reference document is deleted.

```
{
    "EventId":"AT4ShAVLN0Ck7kVhMHYzUw",
    "EventType":"refdoc.deleted",
    "Created":"2021-02-19T13:37:59.3381091Z",
    "Data":{
        "ProjectUID":"XXXXXXXXXXXXXXXXXX",
        "RefDocUID":"YYYYYYYYYYYYYYYYYY"
    }
}
```

annotation.added

Fires when an annotation added in a proof tool to an asset.
This can be a high-level annotation associated with 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":"XXXXXXXXXXXXXXXXXX",
        "AnnotationUID":"YYYYYYYYYYYYYYYYYY",
        "AssetUID":"AAAAAAAAAAAAAAAAAA",
        "Text":"Annotation text"
    }
}
```

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":"XXXXXXXXXXXXXXXXXX",
        "AnnotationUID":"YYYYYYYYYYYYYYYYYY",
        "AssetUID":"AAAAAAAAAAAAAAAAAA"
    }
}
```

task.created

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

```
{
    "EventId":"QbLNa0r3UkuLPJ3KZpoDNg",
    "EventType":"task.created",
    "Created":"2021-02-19T15:10:19.7757392Z",
    "Data": {
        "TaskUID":"XXXXXXXXXXXXXXXXXX"
    }
}
```

task.deleted

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

```
{
    "EventId":"0Jd-zSoOKU2_PoG6zSb5uQ",
    "EventType":"task.deleted",
    "Created":"2021-02-19T15:20:58.5833087Z",
    "Data": {
        "TaskUID":"XXXXXXXXXXXXXXXXXX"
    }
}
```

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-zSoOKU2_PoG6zSb5uQ",
    "EventType":"task.deleted",
    "Created":"2021-02-19T15:20:58.5833087Z",
    "Data": {
        "TaskUID":"XXXXXXXXXXXXXXXXXX"
    }
}
```

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-zSoOKU2_PoG6zSb5uQ",
    "EventType":"task.deleted",
    "Created":"2021-02-19T15:20:58.5833087Z",
    "Data": {
        "TaskUID":"XXXXXXXXXXXXXXXXXX"
    }
}
```

webhook.test

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

```
{
    "EventId":"0Jd-zSoOKU2_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 YYYYYYYYYYYYYYYYYYYYYYYYYY"
```

Responses

HTTP Code	Response
200	**Success**. 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 sen
      "created": "2021-02-09T09:17:27.904911"    // to ensure that this is
    },
    {
      "webhookUID": "XXXXXXXXXXXXXXXXXXXX02",
      "url": "http://xxxxx.com/part/of/url02",
      "secret": "YYYYYYYYYYYYYYYYYYYY02",
      "created": "2021-02-09T09:17:27.904911"
    }
  ]
}
```

POST /api/v1/webhook

Sets us a new webhook.

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 YYYYYYYYYYYYYYYYYYYYYYYYYY"
     -H  "Content-Type: application/json-patch+json"
     -d "{\"url\":\"http://xxxx.com/part/of/url\"}"
```

Responses

HTTP Code	Response
200	**Success**. List of webhooks.

```
{
  "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
      "secret": "YYYYYYYYYYYYYYYYYYYY01",     // This secret will be sent w
      "created": "2021-02-09T09:17:27.904911" // to ensure that this is API
                                              // see HTTP response header "
  }
}
```

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.
412	**Error**: 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 to API host to update.
Curl:

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

Responses

HTTP Code	Response
200	**Success**. Webhook with the given ID deleted.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.
404	**Error**: 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-zSoOKU2_PoG6zSb5uQ",
        "EventType":"webhook.test",
        "Created":"2021-02-19T15:20:58.5833087Z",
        "Data": {
            "DummyData":"random string"
```

```
        }
    }
```

When posting test data, 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 YYYYYYYYYYYYYYYYYYYYYYYYY"
     -H  "Content-Type: application/json-patch+json"
     -d "{\"webHookUID\":\"XXXXXXXXXXXXXXXXXXXXXX\"}"
```

Responses

HTTP Code	Response
200	**Success**. Webhook is tested, see your endpoint logs.

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

HTTP Code	Response
400	**Error**: Parameters' validation failed.
	See HTTP code 400 description.
404	**Error**: Webhook with the given ID not found.