# Action Items Marketplace + Partner Portal

## Labeling marketplace place

### Status / context

#### **Communication APIs with Labeling Partners**

***“Availability Validation API”*** [**GET**]: Alectio sends a signal to check how many annotators are theoretically available for a specific task type in the immediate future; receives number of annotators (to be push to jobs table)

#### Workflow

Alectio backend will have schedular to which will constantly be calling RESTful endpoint which is hosted by partner and responding in the following JSON format.

| {  "labelers": [{  "labeler\_id": "uuid",  "hours\_available": [{  "from": "ISO8601 time",  "to": "ISO8601 time"  }],  "task\_types": ["object\_detection", "image\_classification"]  }] } |
| --- |

Labelers availability will be continuously updated, and this information will help to calculate the availability of labeling partners. Alectio will provide smart partner selection choices to the end user.

### Authentication & Authorization

Alectio backend endpoints are protected with access token header, in order to access alectio marketplace endpoints partners need to provide the Authorization header with Bearer <Access Token>.

There are 2 types of tokens

1. Access Token: Short expiry usually after 24 hours
2. Refresh Token: Long expiry up to 6 months and use to generate new access token without presenting the email/password.

*Note: Generating new refresh token will replace the older one. So previous refresh token will no longer work*

#### Getting Access Token

There are 2 possible ways to generate access token,

### Method 1 (via email/password):

Requesting a token from the login API by calling `/user-services/v1/user/login` POST request and passing credentials as request payload.

| {  "email": "admin@partner.com",  "password": "super-secret-passcode" } |
| --- |

In success response:

| {  "id": "8155daf2-6a87-43bb-b77f-245cf2797255",  "first\_name": "Partner",  "last\_name": "Admin",  "email": "user@partner.com",  **"access\_token": "<access-token>",  "token\_expiry": "2021-10-09T06:52:40.878954662Z",**  "role": "Partner" } |
| --- |

Access token with short expiry, but can be used with the requests

### Method 2 (via refresh token)

Generate refresh key and use refresh key to generate new access token upon expiry.

Refresh token generation endpoint: POST `/user-services/v1/oauth/refresh-token`

Header: Authorization Bearer ACCESS\_TOKEN\_FROM\_LOGIN

**Example cURL**

| **curl --location --request POST '{{base\_url}}/user-services/v1/oauth/refresh-token' \ --header 'Authorization: Bearer ACCESS\_TOKEN\_FROM\_LOGIN\_RESPONSE'** |
| --- |

**Response**  
Accepted 202

| {  "token": "**de8cf4cbeea747ff8077024388a19812**", **// refresh token**  "token\_expiry": "2022-04-06T09:57:00.812462479Z",  "grant\_type": "refresh\_token",  "message": "this token invalidate the older one, keep it safe" } |
| --- |

**Note:** *One client can have 1 refresh token, so generating a new one will invalidate the older. New access key can not be regenerated using older token*

Now generate access key using refresh token (refresh token is alternative to email/password)

`POST /user-services/v1/oauth/access-token`

Request payload: (no authorization header is required for this one)

| {  "refresh\_token": "**de8cf4cbeea747ff8077024388a19812**",  "grant\_type": "access\_token" } |
| --- |

In response the client will get an access token which will be used in subsequent API calls.

| {  "token": "ACCESS\_TOKEN",  "token\_expiry": "2021-10-09T10:22:41.742334375Z",  "grant\_type": "access\_token" } |
| --- |

### Job Offer, Submission & Status Update

***“Request & Job API”*** [POST]: Alectio sends to the selected partner a summary of the task (task type, price, expected time) + instructions; the partner accepts or refuses the task within N minutes. Another request is sent to the next best candidate partner if the first request is refused.

Once a customer chooses the labeling partner from the marketplace the next step is to check if the selected partner is willing to perform the task. For that Alectio backend will call the partner RESTFul endpoint which is designed to accept following payload as POST request body:

Request payload:

| {  "job\_id": "uuid",  "task\_type": "object\_detection | image\_classification | etc.",  "instructions": "s3-download-link",  "dataset\_size": Number,  "estimated\_cost": Double,  "currency": "USD | CAD | GBP | EUR",  "time\_tag": "ISO Timestamp" } |
| --- |

Now the partner will either accept the labeling request or can reject it with any friendly reason. In response, the partner will call Alectio’s RESTful endpoint which is expecting the following response as POST request body using job\_id from previous request.

[POST] /marketplace/v1/jobs/{**job\_id**}/offer

[Header] Authorization: Bearer **ACCESS\_TOKEN**

| {  "status": "accepted | rejected",  "message": "<optional> any reason for rejection" } |
| --- |

After receiving a response from labeling partner Alectio backend will validate if partner chooses status as ***rejected*** then backend will record the response and relay the request to next customer’s choice.

If chosen status is ***accepted*** the in response API will respond with the following response.

| {  "job\_id": uuid,  "instructions": "partner\_workspace/job\_slug/instructions.pdf",  "submission\_bucket": "partner\_workspace/job\_slug/submission",  "dataset\_bucket": "partner\_workspace/job\_slug/dataset",  "annotations\_format": "coco | pascal"  } |
| --- |

Now Alectio have shared all the information and s3 buckets with IAM permissions etc.

***“Job Detail API”*** [GET]: Labeling partner at any point in time can request for job details by calling Alectio’s endpoint.

[GET] api.alectio.com/marketplace/v1/jobs/{**job\_id**}

[Header] Authorization: Bearer **ACCESS\_TOKEN**

API will response back with following:

| {  "job\_id": uuid,  "instructions": "s3-download-link",  "submission\_bucket": "s3\_path/bucket",  "dataset\_bucket": "s3\_path/bucket",  "annotations\_format": "coco | pascal" } |
| --- |

***“Job Status API”*** [POST]: Alectio main portal can check the status of a job by calling the following endpoint.

[GET] api.alectio.com/marketplace/v1/jobs/{**job\_id**}

[Header] Authorization: Bearer **ACCESS\_TOKEN**

API will response back with following:

| {  "job\_id": uuid,  "status": "sent | in\_progress | pending | completed" } |
| --- |

***“Results or Interim Updates Submission API”*** [POST]: The partner sends the results in a standard format of his choice, with a field specifically stating what the format is. The results are converted to Alectio format and pushed to ES. The performance is updated in the DB.

Once the partner has done with the labeling and this time to submit the results back at the given bucket address submission\_bucket retrieve from job detail API and call the following endpoint:

[POST] api.alectio.com/marketplace/v1/jobs/{**job\_id**}/status  
[Header] Authorization: Bearer **ACCESS\_TOKEN**

And pass the following payload as POST request body:

| {  "status": "in\_progress | completed | submitted",  "message": "<optional> String" } |
| --- |

#### 

#### **Tables**

* PARTNER\_INFO
* PARTNER\_SUPPORTED\_TASKS
* LABELING\_TASKS (PARENT)
* LABELING\_SUBTASKS

#### **Job Statuses**

Database table: rc\_job\_status

| **Job Status** | **Description** |
| --- | --- |
| AVAILABILITY INQUIRED | We have info about number of annotators available - or proxy |
| REQUEST SENT | We asked if they wanted the job - number recs, price, exp. time, instructions |
| REQUEST ACCEPTED | They agreed and wait for the job |
| REQUEST REFUSED | They refused and we move on to the next |
| REQUEST EXPIRED | They didn't reply |
| JOB SENT | They received the job |
| JOB COMPLETED | We received the results |

### To-do

* Availability API returns number of annotators; this needs to be pushed to the LABELING\_SUBTASKS database
* The recommendation system needs to be updated with num\_annotators with **time = num\_rec \* calc\_time\_per\_record / num\_annotators**
* Scheduler needs to manage how the requests are managed for the 3 choices; if job refused or expired, LABELING\_SUBTASKS table needs status to be updated
* The request API needs to pass **instructions**, total price, expected time, num records and task type
* Instructions need to be passed as pre-signed urls
* The results API needs to return results in standard format + field with format
* Alectio needs to convert results to Alectio standard format and push to ES
* The time per record needs to be updated in DB at time when the job comes back, the accuracy needs to be updated when the user resumes the job

## Labeling Partner Portal

## Authentication

TODO: Login in separate DB

### To-do

* Partners need to be invited to sign-in, or Alectio manually sign them up
* Check the accuracy of the computation on the portal
* Fix styling on portal