Crowd Guru - DataAPI

v1.13.75

Introduction

Crowd Guru provides mikrotasking capabilities for pretty much any scenario that can be solved via webforms. Those tasks get done by setting up “jobs” on our platform that our “Gurus” work on.

Our jobs use datasources. These datasources can be accessed with our API so you can seed your job units whenever you need to.

The incoming data e.g. will be used by our gurus to complete jobs.

The resulting datasets could be posted to a custom URL on completion or you pull the data whenever you want via the API.

Basics

Every job gets a unique identifier. So you might have one job to push data to us and another where you retrieve your completed units from because we might have a quality assurance job active for you. We will give you a short summary of the URLs during setup.

We have a standard format for our API. You cannot change or define this URL at your own, please contact it@crowdguru.de for your specific request-URL.

https://<username>:<password>@crowdmodul.crowdguru.de/api/<jobId>

We only offer SSL encrypted connections. Username, Password and jobId will be provided by us. **Don’t add slashes at the end of the URL.**

Data exchange format

Every data exchange has to be formatted in JSON. Please make sure you send valid JSON. Our JSON-Parser is strict (and will, for example, not accept single quotes). **The POST field "data" acts as a container for the JSON.**

You can mail us the example JSON you would like to use and we configure this within 10 min on our platform for you. After completing the job configuration we send you a draft of the JSON you would receive from us.

IDs

The "id"-Key is a reserved key for you to have a reference within your data. You should use it to reference new jobunits posted to us and can use it to pull selected jobunits again if they weren't transmitted properly.

Errors

Successful response contain the **200** Status-Code and application/json Content-Type. If you get a **404** Error without any content check your request-url. If you get the **412** Precondition Failed error code, your access is not yet available. Any other **4xx** error codes are explained in the section below. If you get a **5xx** Status-Code, please contact it@crowdguru.de.

Methods

Seed new entries

Send your new entry data in the POST[data] field with Content-Type application/x-www-form-urlencoded.

Please make sure your data structure matches exactly your example JSON, or you will get a 400 error. If your structure has changed, please contact us, you can't change our configuration remotely at this moment.

HTTP-Method: POST, URL: <Base-URL>
POST[**data**]= [**{},{}**] new entries in JSON array
Content-Type: application/x-www-form-urlencoded
e.g. https://username:password@crowdmodul.crowdguru.de/api/a525fdasedfg23

Possible errors:

- 412 Preconditions Failed
- 400 Bad Request; Content: POST[data] is empty
- 400 Bad Request; Content: JSON syntax errors
- 400 Bad Request; Content: JSON structure differs from config

Example-Response:
{ “1236”: 1, “1237”: 2 }

Fetch all completed units

HTTP-Method: GET, URL: <Base-URL>
Response has Content-Type: application/json
e.g. https://username:password@crowdmodul.crowdguru.de/api/a525fdasedfq23/new

Possible errors:
412 Preconditions Failed

Example-Response (no new completed units):
[]

Example-Response (2 entries):
[
{ “id”: 1234, “title”: “hello world” },
{ “id”: 1235, “title”: “hello world” }
]

Fetch single completed unit by your ID

HTTP-Method: GET, URL: <Base-URL>/id
e.g. https://username:password@crowdmodul.crowdguru.de/api/a525fdasedfq23/1

Possible errors:
412 Preconditions Failed
404 Not Found; Content: entry '<id>' not exists

Example-Response:
{ “id”: 1234, “title”: “hello world” }

ACK for the completed data you received

To verify that you received your completed units successfully we need you to acknowledge the received unitsIds to us. We expect your product ID's, the list must contain strings or integers.

HTTP-Method: POST, URL: <Base-URL>/jobId/received
POST[**units**]= [**1234,1235**]
Content-Type: application/x-www-form-urlencoded

Example-Response:
{1234 => true, 1235 => true}

The results include a list of id's with the flag true or false that shows if we could set the entries to exported or not.

eventdriven push from our side

HTTP-Method: POST | PUT | GET, URL: <YOUR-URL>

Example-Response:
[
{ "id": 1234, "title": "hello world" },
{ "id": 1235, "title": "hello world" }
]

If we get HTTP 200 back we will mark this data entry as delivered, otherwise we'll try to deliver it later.

Delete a jobunit

HTTP-Method: DELETE, URL: <Base-URL>/id
POST[**data**]= new entry data in JSON

Possible errors:
412 Preconditions Failed

Example-Response:
OK

Update a jobunit before work has begun on it

HTTP-Method: PUT, URL: <Base-URL>/id
POST[**data**]= new entry data in JSON

Possible errors:
412 Preconditions Failed
400 Bad Request; Content: POST[data] is empty
400 Bad Request; Content: JSON syntax errors
400 Bad Request; Content: JSON structure differs from config

Example-Response:
OK

Same behavior as POST request.

Send unit based feedback to CG

HTTP-Method: POST, URL: <Base-URL>/feedback
POST[**data**]= [**{id: ", text:",data:", rating:"}, ...**]
Content-Type: application/x-www-form-urlencoded
Possible errors:
412 Preconditions Failed
400 Bad Request; Content: POST[data] is empty
400 Bad Request; Content: JSON syntax errors

Example-Response:
{ "1236": true, "1237": false }

We should receive a feedback for every unit.

id
Unit-id to identify previously posted unit.

[id_field = "cg_id"**]**
"cg_id" or "client_id", while client_id means **id** will be interpreted as the id given by the client when unit was sent to us. "cg_id" means we lookup in the internal crowdguru id column we create and return when a unit was posted to us.

qa_user_id
id of the person checking on customer side.

text
Message e.g. "wrong categorisation".

data
e.g. {text : [true,false,true], image:false}
or {link:false,color:false}
we can use this data to validate your feedback.

rating
default: 0
A basic rating either wrong (0) or right (1)

Crowd Guru - CrowdProvider API

This part of the API is used by customers to take advantage of our crowd inside their own Platform. We provide the possibility to embed external sites as an iFrame (including custom encrypted attributes to the iFrame-URL, e.g. GuruID, age, gender, location.) The following endpoints are mainly for tracking purposes. But we are also receiving payment information and rejection messages.

Start unit

HTTP-Method: POST, URL: <Base-URL>/v2/unit/start
POST[**data**]={
"jobID": "a",
"unitID": "123",
"guruID": "xxx",
"jobName": "test"
}

Content-Type: application/x-www-form-urlencoded
Possible errors:
412 Preconditions Failed
412 Parameter missmatch
400 Bad Request; Content: POST[data] is empty
400 Bad Request; Content: JSON syntax errors

Example-Response:
{ "success": true }

Tracking: You tell us that a unit has been started.

Finish unit

HTTP-Method: POST, URL: <Base-URL>/v2/unit/finish
POST[**data**]={
"jobID": "a",
"unitID" : "123",
"guruID": "xxx",
"unitValue": 23 //euro cent
}

Content-Type: application/x-www-form-urlencoded
Possible errors:
412 Preconditions Failed
412 Parameter missmatch
400 Bad Request; Content: POST[data] is empty
400 Bad Request; Content: JSON syntax errors

Example-Response:
{ “success”: true }

Tracking: You tell us that a unit has been finished.

Approve unit

HTTP-Method: POST, URL: <Base-URL>/v2/unit/approve
POST[**data**]={
"jobID": "a",
"unitID" : "123"
}

Content-Type: application/x-www-form-urlencoded
Possible errors:
412 Preconditions Failed
412 Parameter missmatch
400 Bad Request; Content: POST[data] is empty
400 Bad Request; Content: JSON syntax errors

Example-Response:
{ “success”: true }

Feedback: you tell us that a unit has been approved

Reject unit

HTTP-Method: POST, URL: <Base-URL>/v2/unit/reject
POST[**data**]={
"jobID": "a",
"unitID" : "123",
"guruID": "xxx",
"mode" : "hardreject|softreject",
"message": "this is a plaintext msg that will be sent as an email, it can contain carriage returns "
}

Content-Type: application/x-www-form-urlencoded
Possible errors:
412 Preconditions Failed
412 Parameter missmatch
400 Bad Request; Content: POST[data] is empty
400 Bad Request; Content: JSON syntax errors

Example-Response:
{ “success”: true }

Feedback: You tell us that a unit has been rejected, you also provide the reason. A hardreject will block the Guru for this job.