



GreenLink Platform

API Documentation

Contents

Introduction	3
Who This Document is For	3
Assistance Using the API	3
Using the API	3
API URL	3
Authorization Token Request	3
Uploading A File	5
Without File Review	5
With File Review	5
File Requirements	5
Data Categories	5
Interpreting API Responses	5
Upload Example	7
Document Revision History	7

Introduction

The Application Programming Interface (API) serves to connect with the GreenLink backend architecture to make data submission to the platform without manually using the GreenLink Data Submission Portal. (Link for manual submissions, <https://greenlink.cremedatafoundry.com>)

To access the GreenLink backend, the user sends an API call (aka data transmission request) by executing the code provided within this document that sends the request and processes the response. This document describes how to programmatically execute an API call.

Who This Document is For

This guide is intended for implementation by a data engineer or IT professional authorized by your organization to transfer data outside your organization. Its use depends on familiarity with data storage and transfer through use of the python programming language.

Assistance Using the API

This document is meant to provide comprehensive information about using the GreenLink API; however, if you need assistance, please contact Creme Global at help@cremeglobal.com.

Using the API

Run a Python script to request an access token and upload your data to the GreenLink platform without manually using the upload tool on the platform. You can use the API Python script "API_Script.py" provided or cut and paste the Python code online in the GreenLink API Documentation from the Data Submission Portal: <https://greenlink.cremedatafoundry.com/support/articles/190>

API URL

The API can be accessed using the following link along with an access token:

<https://greenlink-api.cremedatafoundry.com/api/>

Authorization Token Request

Users of the GreenLink API will be given a combination of an email address and password to perform a preliminary authentication step to obtain a 24-hour token, which they will subsequently use when interacting with the API.

Each Authorization Token is assigned to a user. One of two sequences will be initiated depending on the user type:

1. Data is automatically approved once pushed (no approval step required).
2. Data requires approval and sits in the approval required state on the portal until approved.

Here is a screenshot of the script.

```

1  # -*- coding: utf-8 -*-
2  """
3  GreenLink API Script
4  Created on 8th September 2022
5  @authors: Badri O'Nair, Viorica Botea
6  """
7
8  #!/usr/bin/env python
9  import configparser
10 import json
11 import os
12 import requests
13 import sys
14
15 #-----
16 #   API upload
17 #-----
18
19 requests.packages.urllib3.disable_warnings()
20
21 ROOT_DIR = os.path.abspath(os.path.dirname(__file__))
22
23 """Derive all the configurations from the config.ini file"""
24 Config = configparser.ConfigParser()
25 Config.read("%s/api_config.ini" % ROOT_DIR)
26
27 """ Read all the values for the configurable parameters from the
28 api_config.ini file"""
29 portal_url = Config.get('DEFAULT', 'portal_url')
30 username = Config.get('DEFAULT', 'username')
31 password = Config.get('DEFAULT', 'password')
32
33 """ The Token URL and the data URL that is required for the API pull """
34 TKN_URL = "https://" + portal_url + "/api/login/"
35 UPLO_URL = "https://" + portal_url + "/api/upload/"
36
37 """ Payload for the authentication API call """
38 tkn_payload = {
39     'email': username,
40     'password': password,
41 }
42
43 try:
44     r = requests.post(
45         TKN_URL,
46         headers={"Content-Type": "application/json", "Accept": "application/
47 json"},
48         data=json.dumps(tkn_payload)
49     )
50 except:
51     print('## FATAL: API call failed to get the authentication token')
52     sys.exit()
53
54 token_dict = json.loads((r.content).decode("utf-8"))
55 token = token_dict['token']
56
57 """ Format the header component with the token embedded"""
58 upld_header = {
59     "Authorization": 'JWT %s' % token
60 }
61
62 def upload_to_dataplatform(file_type, file_path, needs_approval=False):
63     """ Function to call the API to upload the file over to the data platform
64     takes two arguments"""
65
66     post_data = {"metadata": json.dumps({"type": "%s", "needs_approval":
67 needs_approval}) % file_type}
68
69     upload_file = file_path
70     try:
71         response = requests.post(
72             UPLO_URL,
73             headers=upld_header,
74             files={'file': open(upload_file, 'rb')},
75             data=post_data
76         )
77         if response.status_code == 201:
78             print('## INFO : API call to upload the file has been successful:
79 %s' % (response.content).decode("utf-8"))
80         else:
81             print('## FATAL: API call failed to upload the file: %s' %
82 (response.content).decode("utf-8"))
83         except Exception as ex:
84             print('## FATAL: API call failed to upload the file: %s' % ex)
85             sys.exit()
86
87
88 def main():
89     # update path of the file to upload
90     # ROOT_DIR is the home directory of this script
91     file_path = '%s/upload_test_file.csv' % ROOT_DIR #
92     'file_to_upload_full_path'
93     file_type = "Water Sampling and Testing Results"
94
95     needs_approval = True # or False
96     upload_to_dataplatform(file_type, file_path,
97 needs_approval=needs_approval)
98
99 if __name__ == '__main__':
100     main()
101
102 """ The file type must be specific. There is a list of agreed file types that
103 can be passed on """
104 """ Example Function to call upload of each of the file to the platform
105
106 upload_to_dataplatform("Tissuesampling", "C:/Tissuesamplingdata.csv")
107 upload_to_dataplatform("watersampling", "C:/waterDataFile.csv",
108 needs_approval=True)
109
110 """

```

Uploading A File

When pushing data from your organization to GreenLink using the API, a bespoke user data control will be implemented. GreenLink has functionality that allows files to be submitted for review and requires approval by someone else in your organization before being integrated into to your organization's repository. This control feature is set up during your Onboarding Meeting.

By default, users are set up to submit data to GreenLink without File Review.

Without File Review

Files submitted will be sent to the processing queue and then merged into your data repository.

With File Review

Files submitted will be labeled For Review and stay in a holding folder until the designated approver reviews and marks the files either as approved or rejected. Approved files are integrated into your organization's repository. Rejected files do not get integrated into your organization's repository.

File Requirements

There is currently a 10MB size limit per upload. Only comma separated value (.csv) and Excel files (.xls and .xlsx) are accepted at this time. Other file types may be accepted in the future. Contact Western Growers if you are interested in submitting a file of a different file type extension.

Data Categories

GreenLink has the following pre-designated data categories. Note that the spelling, letter case, and punctuation must match, character-for-character, the text below inside the API script:

- Planting and Harvesting Information
- Pre-Harvest Field Inspection Information
- Pre Harvest Assessment
- Pre-Season Assessments
- Tissue Sampling and Testing Results
- Water Sampling and Testing Results

Since the file structures relative to each of those categories will differ by organization, the first upload for each of those categories will be used to configure automated data processing of future file uploads for your organization.

Use the API to upload a file by indicating the file parameter. If the needs_approval parameter is set to True, the status of the submission is designated as Ready to Submit, otherwise the submission is sent directly to the processing queue, waiting for the ETL scripts to process and transition it to your organization's repository.

Interpreting API Responses

When running the API script, responses will be generated based on successfully completed or failed actions. Below is a list of the responses and their descriptions:

Code	Response	Description
201	{submission_id: <submission_id>}	Request was successful. A submission with id <submission_id> has been created for the requesting user. It should be able to be seen on the data portal under the Submissions tab.
400	{“file”: [“No file was submitted.”], “status_code”: 400, “error_type”: “ValidationError”}	Request is missing the file parameter.
400	{“file”: [“The submitted data was not a file. Check the encoding type on the form.”], “status_code”: 400, “error_type”: “ValidationError”}	Data submitted under the file parameter is not a file.
400	{“detail”: “Submissions of type <file_type> aren’t supported”, “status_code”: 400, “error_type”: “UnsupportedMimeType”}	The type of file submitted for upload is not one of the accepted types. Currently only .csv, .xlsx, and .xls file types are accepted. This may change in the future.
400	{“detail”: “File with name ‘<file_name>’ already exists in this location.”, “status_code”: 400, “error_type”: “DuplicateNameError”}	A file with the same name was already uploaded by the user at some time in the past. Make sure your files have different names.
400	{“metadata”: [“This field is required.”], “status_code”: 400, “error_type”: “ValidationError”}	Metadata parameter was missing from the request.
400	{“detail”: [“Invalid JSON passed for metadata”], “status_code”: 400, “error_type”: “ValidationError”}	The data passed under the metadata parameter is not a valid JSON object.
400	{“detail”: [“type is required”], “status_code”: 400, “error_type”: “ValidationError”}	No type was provided as part of the metadata. Type needs to be specified to upload the file. Refer to Data Categories section of the API Documentation.
400	{“detail”: [“Inexistent upload type <type>”], “status_code”: 400, “error_type”: “ValidationError”}	The type provided was not one of the available upload types. See above for available types.
413	Entity too large	The file sent for upload was too big. Currently there is a size limit of 10MB for any uploaded file.
401	{“detail”: “Authentication credentials were not provided.”, “status_code”: 401, “error_type”: “NotAuthenticated”}	No Authorization header has been sent with the request.
401	{“detail”: “Invalid Authorization header. No credentials provided.”, “status_code”: 401, “error_type”: “AuthenticationFailed”}	The token sent with the Authorization header was not valid. It could have been a token that has expired (authorization tokens are only valid for 24h after they are generated).

Upload Example

Here is an example of uploading using your Command Line:

```
curl -i -X POST -F "file=@/path_to_file_to_upload.csv" -F 'metadata={"type":"Water Sampling and Testing Results", "needs_approval":"True", "property1":"value1", "property2":"value2"}' -H 'Authorization:-JWT <authorization_token>' https://greenlink-api.cremedatafoundry.com/api/upload/
```

Config file:

```
[DEFAULT]
portal_url : greenlink-api.cremedatafoundry.com
username : <user-name>
password : <password>
```

Document Revision History

Version	Contributor	Date	Description
1.0	Viorica Botea	07/30/2022	API Version 1.0
1.1	Viorica Botea	09/12/2022	API Version 1.1
1.2	Marlene Hanken	03/08/2023	API Version 1.2 – includes more process language