# COD Services API

# Application Integrations: Bulk P-Code geographic coordinates to spreadsheet.

## Contacts

Tom Haythornthwaite, United Nations, Office for the Coordination of Humanitarian Affairs, haythornthwaite@un.org

David Megginson, United Nations, Office for the Coordination of Humanitarian Affairs, david.megginson@humdata.org

Anita Russo, University of Georgia, Carl Vinson Institute of Government, Information Technology

Outreach Services, [arusso@uga.edu](mailto:arusso@uga.edu)

Implementation Specifications:

Internet connection

Microsoft Excel spreadsheet (.xls) of geographic coordinates in decimal degrees with latitude and longitude values in separate columns

Python 3.6+ (may work with earlier versions)

Python modules (requests, pandas)

## History

First draft: 16 September 2021

Contents

[COD Services API 1](#_Toc82685492)

[Application Integrations: Bulk P-Code geographic coordinates to spreadsheet. 1](#_Toc82685493)

[Contacts 1](#_Toc82685494)

[History 1](#_Toc82685495)

[Summary 2](#_Toc82685496)

[Instructions 2](#_Toc82685497)

## Summary

The COD Services API (<https://apps.itos.uga.edu/CODV2API>) provides various RESTful1 outputs. In addition to finding metadata for the CODs, download materials or gazetteer content, external applications have access to more functions from this resource. This document describes how the lookup function for P-Code output may be implemented in a bulk scheme compatible with spreadsheet output. This may be useful for Q/C or other visualization reporting where data may be joined at that admin unit level represented by the P-Codes. Note that a check on the https://data.humdata.org website is recommended for population data linkages or using the API itself to ascertain the provenance of the source of the P-Codes in terms of date and source for such a use case.

## Instructions

1. Ensure the proper dependencies are installed as noted in the Implemented Specification section of this document. Jupyter or other IDE’s may be used to run the python script. This set of instructions are to run the script on the command line in a Windows 10 Enterprise 32 Gig of Ram environment (Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz 3.70 GHz). Other environments may work as well.

Open a command line console. Note that several python instances may be installed on any given machine, so that running the command in a shell that has access to the right version of python with modules installed is important.

2. Download the script: [https://github.com/UGA-ITOSHumanitarianGIS/CODV2API/tree/main/Scripts pcode4latlong2excel.py](https://github.com/UGA-ITOSHumanitarianGIS/CODV2API/tree/main/Scripts%20pcode4latlong2excel.py) (the test input coordinates spreadsheet (Location\_Coordinates.xls) is at that location as well.

3. Update the test input spreadsheet with the coordinates that need to be P-Coded (only those CODs that are enhanced CODs that are coincident with the input coordinates will produce results).

4. Type into the command line console the command to run the script and generate an output:

python -m pcode4latlong2excel

5. After the script is complete, an output.xlsx file with the coordinates and their respective P-Codes are returned. Example output: Table, Excel

Description automatically generated

Note that level 2 is returned and that some countries support more detailed levels. Updating the script to replace the level number in the CODV2API api url will work to return another level of P-Code in the output. This has not been tested as of 16, September 2021 and the script is a beta release which may support parameters in subsequent releases to support level updates and file name location changes without having to update the code.

DRAFT

**Instructions for running Python script on PyCharm IDE:**

**Step 1: Creating a new project/ adding script to the existing project**

You could add the python script to your existing projects by pasting the file at the project location. If you are new to PyCharm IDE, then follow these simple steps

* Click on new project and set its location and name.
* Verify that the base interpreter is set to the python available in your system
* Click on the create button on the right bottom corner
* Now you can add the python script file to this project, or you can simply copy paste script’s content to the main.py file that IDE has created for you.

**Step 2: Installing required packages**

import pandas as pd  
import requests

If your code looks like the above snippet then it means you don’t have these packages installed in your system so to do that click on the terminal option available at the bottom of project window and type the following commands one by one and hit enter

* pip install pandas
* pip install requests
* pip install openpyxl

Once you are done installing these packages, PyCharm will remove the red underline from the import commands in the script.

**Step 3: Adding input file**

You have to provide the path of your input file to the variable named “filePath”. If your path has single \ then change it to \\ as you can see in this example

filePath = "C:\\Users\\bagod\\Desktop\\RA Work\\Location\_Coordinates.xlsx"

Make sure that your input file has two columns with name Longitude and Latitude.

**Step 4: Output**

Once you run the script, an output file of the name “output.xlsx” will get generated at the location your project is running from.

**Instructions for running Python script on Jupyter Notebook:**

**(Need Jupyter? Here is a resource:** [How to install Jupyter Notebook in Windows? - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-install-jupyter-notebook-in-windows/#:~:text=After%20updating%20the%20pip%20version%2C%20follow%20the%20instructions,Jupyter%20using%20command-line%3A%20jupyter%20notebook%20Attention%20geek%21%20). If you have already installed Anaconda, Jupyter is installed.)

**Step 1: (After launching jupyter)**

**Choose, File> Open> Python 3 from the menu in the browser window.** Here is what that might look like:

Graphical user interface, text, application, chat or text message

Description automatically generated

**Step 1: Installing packages**

Run this command

import sys  
!{sys.executable} -m pip install <package\_name>

replace package name with pandas and requests in our case.

**Step 2: At the prompt cell load the script: pcode4latlong2excel**

**Use the %Load command (using the correct path to the location of the .py script. For example:**

**%Load C:\Users\me\Documents\GitHub\CODV2API\Scripts\pcode4latlong2excel.py**

**This will load the script file in the execution window:**

Graphical user interface, text, email

Description automatically generated

**Step 3. Check and change file paths as needed.**

The locations to the ouput.xlsx file and the input Location\_Coordinates.xls files may need to be qualified with a full path so update the location as needed using double back slashes to valid locations**.** (Location\_Coordinates.xls may be updated with the latitude and longitude values that need coding. The spreadsheet contains sample coordinates).

**Step 2: Run the Script**

Ctrl + Enter will run the script showing in the execution cell/window.

When you run the script, an output file with the name “output.xlsx” will generate in your current working directory or specified path. If you try to open it through Jupyter, you might see this error “…….is not UTF-8 encoded, saving disabled”. This error simply means that Jupyter doesn’t know how to display this file. It doesn’t mean that your file isn’t saved, you could simply open it externally by navigating to that folder.

|  |  |
| --- | --- |
| A picture containing plate  Description automatically generated | http://alpha.itos.uga.edu/USAID.MapCODLocationsPreviewer/Content/Images/usaid.png |