# 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

Draft revision: 22 September 2021, to correct the Jupyter instructions and account for additional user conventions with logging and user confirm with potential output file overwrite.

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 Implementation Specification section of this document. Jupyter or other IDE’s may be used to run the python script.   
To run the script on the command line in a Windows environment, follow the steps below. 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. Typing run at the windows start input will usually show an option for “Python Command Window”, which is usually a good choice. Typing the word “Python” and enter in the command window will start up python and show the version in the window as well, and ensure it is compatible with the script. You can exit python now just type exit() and enter.

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. Ensure that the script and input coordinate spreadsheet are located in the current directory of the prompt.

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

python -m pcode4latlong2excel

your screen may look like this:

python -m pcode4latlong2excel

The input file should exist here: C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\Location\_Coordinates.xls, also the output.xlsx will be overwritten if it exists there. Continue? [Y/n]

6. The script will need your input to continue. If you are ok with any output.xlsx that is located in the current working directory being overwritten, then type y or yes. The script will continue. Otherwise, type n or no and hit enter. If you choose to continue, the script will continue and create a log file in

that directory and the output.xlsx file in that location. The message done will appear when the processing is finished.

5. The output.xlsx file with the coordinates and their respective P-Codes may show something similar as follows: 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 2 choices for the next step: A) copy the input coordinate spreadsheet named Location\_Coordinates.xls to the current working directory. Or B) You have to provide the path of your input file to the variable named “QF” and remove the text on the right side of the “=” and replace with the fully qualified and slash escape value of the path and input file. 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"

With either choice, make sure that your input file has two columns with name Longitude and Latitude.

**Step 4: Output**

Follow the prompts. Once you run the script, and confirm to continue, 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)**

**At the project level to start a new notebook, a “New” menu is located at the top right and choose the “Python 3” option.**

**If you already have a notebook running then 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.

The above works great if you are running as administrator. If you see issues, restart Jupyter running on elevated privileges, such as administrator. Note that the python executable may have a space in the path, and then quoting the executable with path is needed for instance:

Graphical user interface, text, application

Description automatically generated

Typing sys.executable will show the location of the python executable. You can then surround the executable and path with double quotes for example: Text

Description automatically generated

**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 location to the input Location\_Coordinates.xls file should exist in the current working directory and typing import os and shift enter with os.getcwd() and shift+enter or the run button will show exactly where the input script should be copied. The 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. A prompt will appear asking you to continue if an output.xlsx file already exists in the current working directory, it will be overwritten. You must choose y or yes at the prompt for the script to continue. The result will be a “done” message and an output file with the name “output.xlsx. 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. The script will also place a log file in that current working directory. The log file lists the coordinates and if there is a point that fails, that is the row where the data most likely were not pcoded after that point.

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