# Data Preparation for Integration Automation

*Updated April 10, 2016*

The following document describes the setup for the data integration automation data preparation. The output will be a zipped file geodatabase (containing the data integration layers) that is ready to be emailed or dropped into Google Drive. There is a folder in our shared Drive called “Consolidated Geodatabases”.

It is up to each agency whether they want to schedule the process to run in the background on a regular basis, or run it manually as needed.

## Schematic

Configuration \*.ini

(script tool)

Batch file and/or Scheduled task

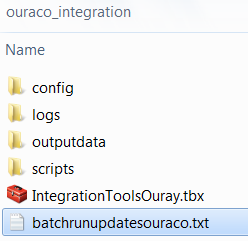
Summary  
Step 1: Set up environment  
Step 2: Configure using script tools  
Step 3: Create batch file to run script  
Step 4 (optional): Set up a task to run updates on a schedule  
Step 5: Monitoring success/failure

## Requirements

Computer to execute scripts/batch file which has Python 2.x and ArcGIS 10.1+ (for arcpy library)

## Step 1: Environment

**Unpack the zip file** into an easily findable and permanent location. It contains the following folder structure, which you can place under a root folder such as ouraco\_integration.



These locations can all be local or on your server, but must be accessible to the computer which will execute the script (i.e. has Python & ArcGIS installed). Best not to have any spaces in the paths if you can avoid it.

\*If you move/rename your source data, you will need to run the configuration script tools again with the new locations. If you move the integration scripts folder structure, you will also need to update the batch file paths (see step 3) and possibly your scheduled task with the path to the batch file (see step 4). It would be optimal to find a good home for the folders before running the configurations.

The config folder is where all your config files (\*.ini files) will be located.

The logs folder holds the log files.

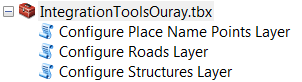
Outputdata is the location of the “blueprint” geodatabase I am providing.

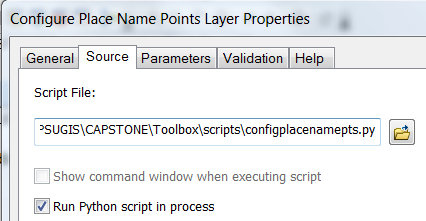
Scripts is the folder for the scripts.

The toolbox holds the script tools used for configuration.

## Step 2: Configuration

**Navigate to the location** of the Integration Toolbox using ArcCatalog. **Go to the properties** of each script tool and **set the script source** (right click/Properties/Source tab) to where the scripts are located. It may be correct already, but check.





configaddresspts.py – for access points and structure points: tool can be copy/pasted (in ArcCatalog) and then set with different data source defaults for the two layers, which share a schema. Or just run it again with a different name for the \*.ini file and different data sources/outputs.

configplacenamepts.py – for place name points

configroads.py – for roads

configesns.py – for ESNs, if applicable

*Optional:* You may also choose to set up script tool defaults in the parameters tab, such as your agency code/SOD.

**Run each script tool in the Integration Toolbox** to set up a different configuration file (\*.ini file) for each layer. Each layer has a different script tool because of the different field mappings. Right now there are tools for access points, structure points (same tool), place name points and roads.

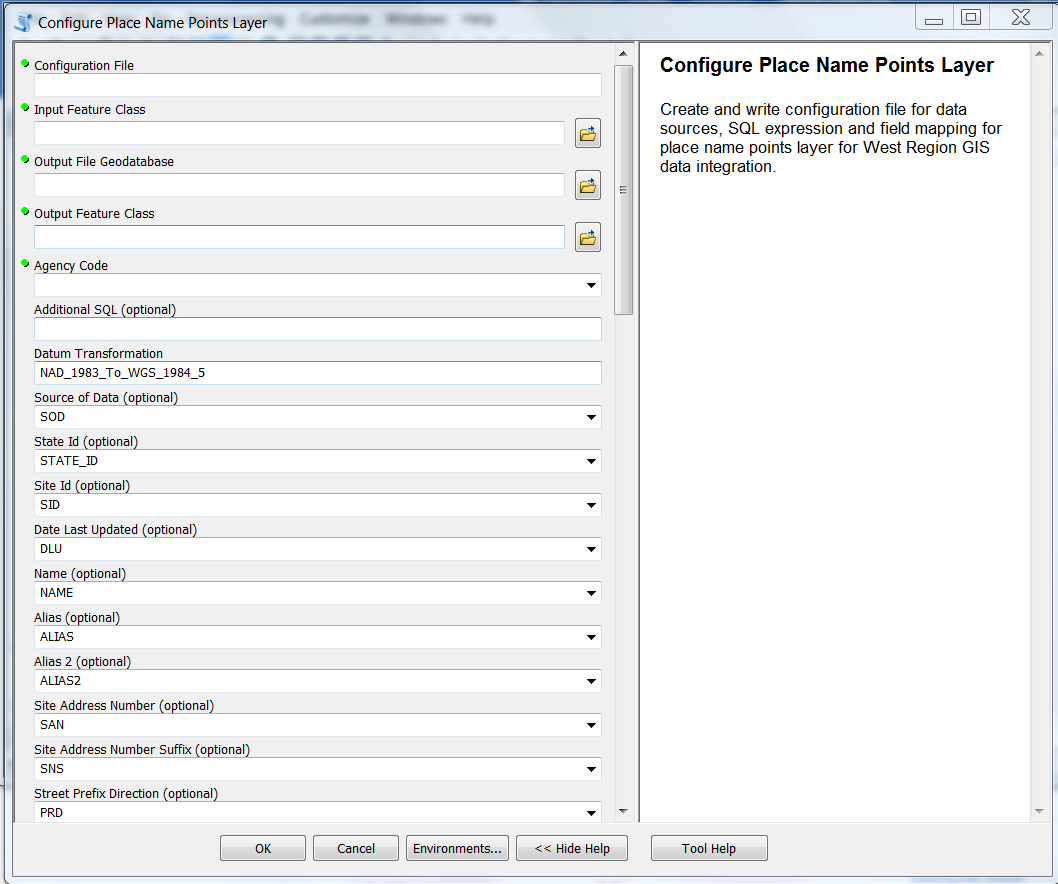
Use the tool help to see what the tool is looking for in each box.

In the top box, you will need to **paste in the path to the config folder and give the .ini file a name**, such as places.ini. For example, c:\mypath\ouraco\_integration\config\places.ini.

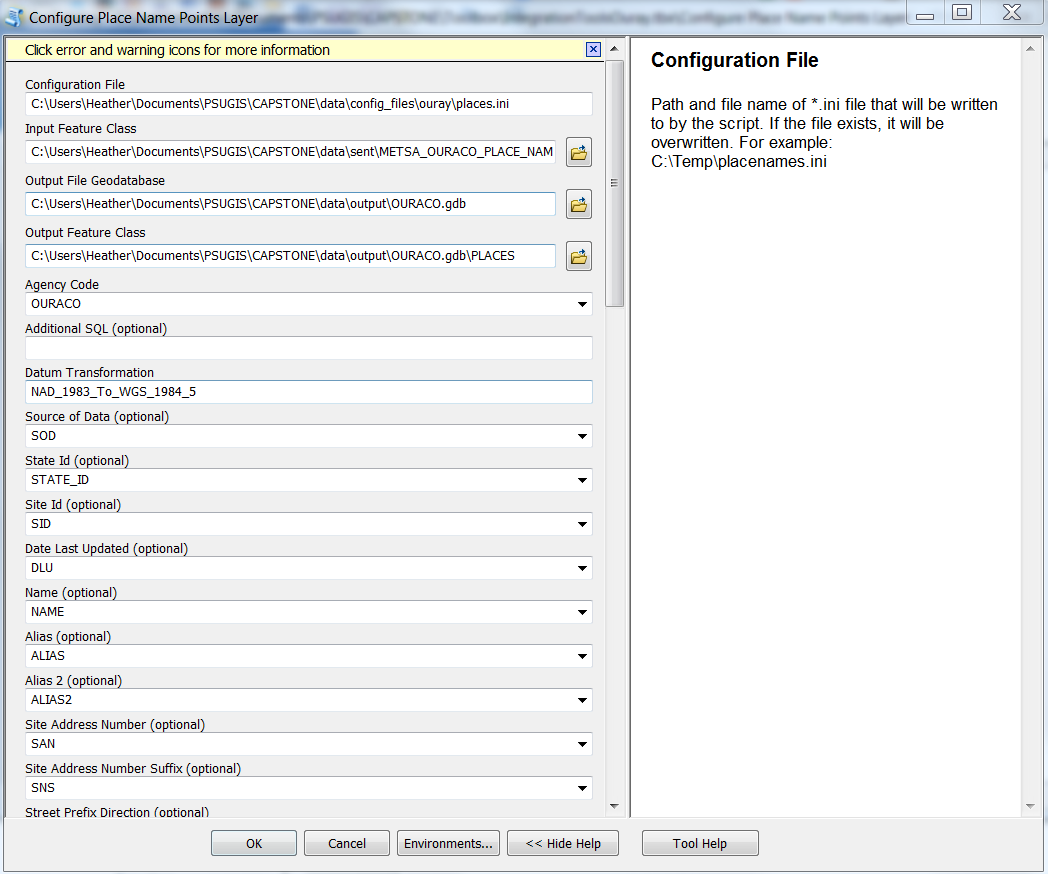
If you have additional data querying (“Additional SQL”) to do besides SOD=“YOURAGENCYCODE” (already coded in), I can help you with the formatting for this box, such as: AND COUNTY=”SAN MIGUEL”

For the field mapping section (below datum transformation), if there isn’t a field that matches the schema in your data, delete the text in the corresponding box. If your field has a different name, use the pull-down menu to select it. Below are screenshots of running the script tools.

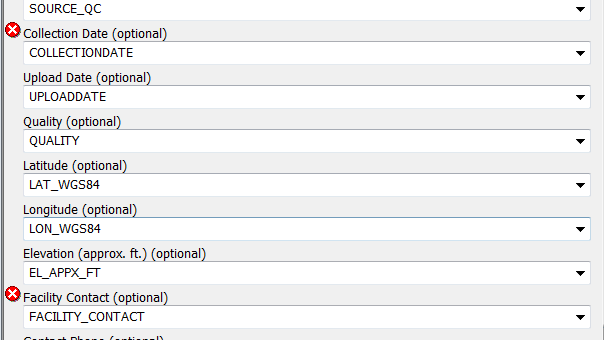
Place Names layer configuration - blank



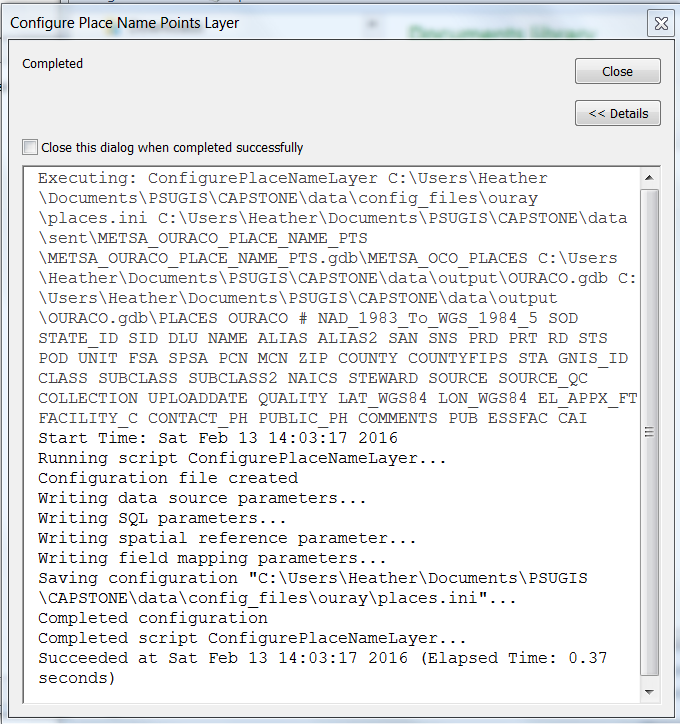
Place Name points configuration after choosing Input Feature Class - note warning



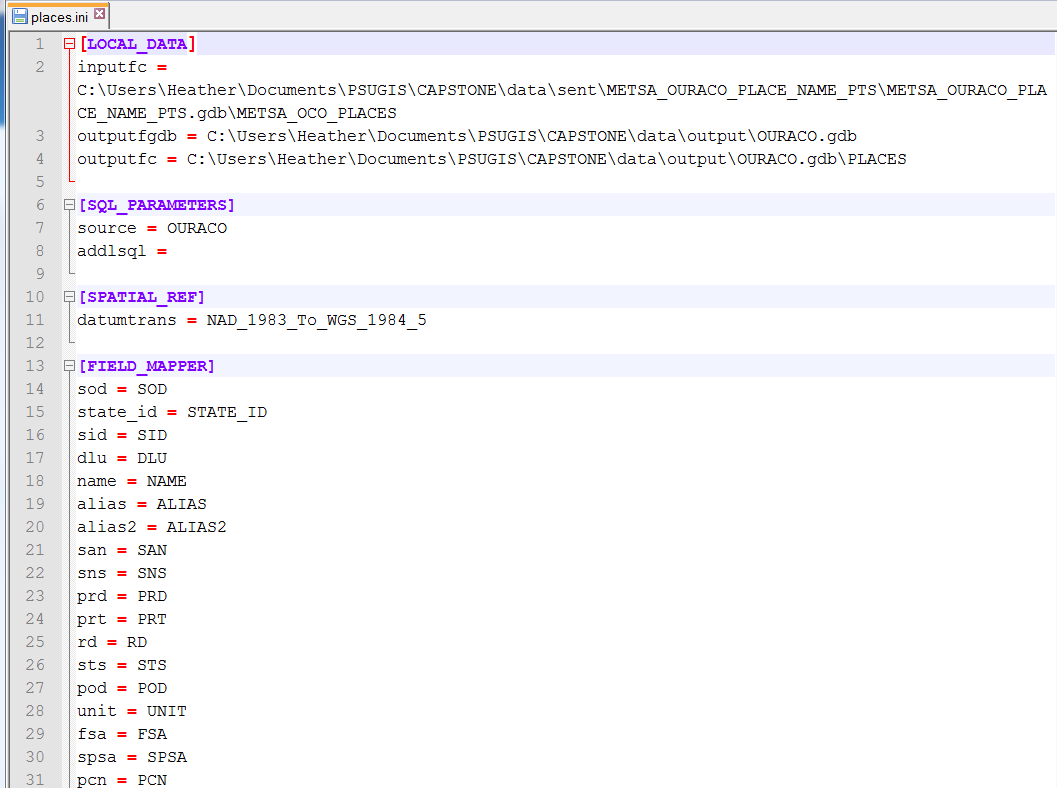
Place Name configuration errors - These two fields have different names in the input than the default. Simply choose the correct field name from the dropdown. **If there isn't a matching field, delete the text in the box.**



Takes a fraction of a second to complete



Place Names configuration (.ini) file written by the config script tool. Can be viewed in a text editor. If you need to make changes, simply run the config script tool again OR you can perform edits in the text editor if you’re careful.



## Step 3: Batch file

I have included a batch file template. **Change the extension of the batch file to “.bat” instead of “.txt”** and **open in a text editor** such as NotePad or NotePad++ (not MS Word).

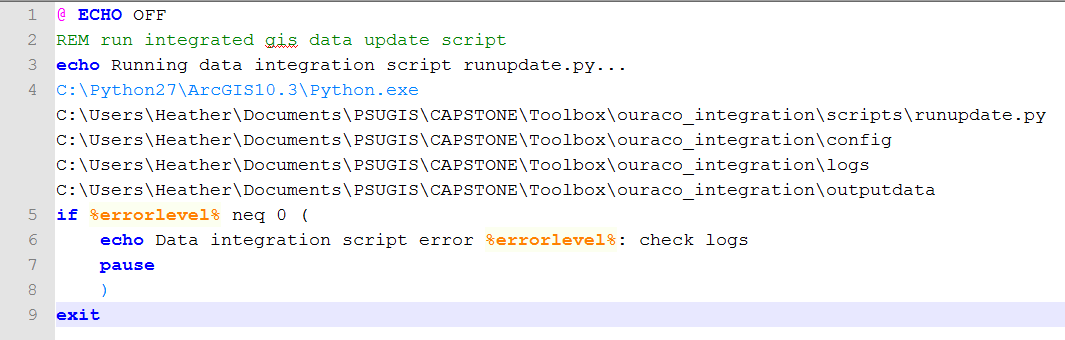
This is the only hard-coded part of these scripts. The file requires:

1. Path to your Python installation (depends on your ArcGIS version)
2. Path of the runupdates.py script
3. Path to config files folder
4. Path to logs folder
5. Path to data output folder

See example below. There is a single space between each parameter, not a carriage return. Note that the command is all on one line - line 4 in the screenshot below. If there are spaces in any of the paths, enclose each whole path in double quotes: “c:\my\path has\spaces”. The script and folders can be located anywhere as long as the computer running the batch file has Python and ArcGIS (for the arcpy library) installed.

I have written simple error checking in the batch file which will leave the black command window open if there is an error, so you know to check the logs.

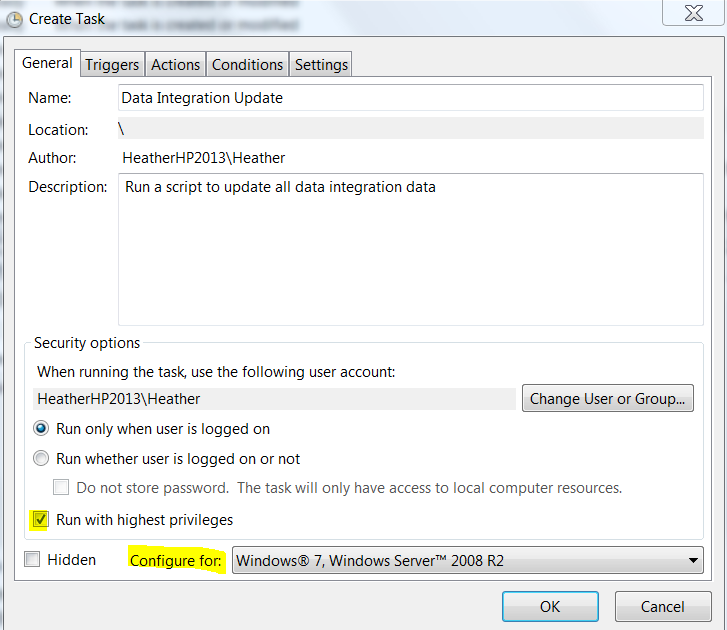
**If you don’t want to set up a scheduled task, you can double click the .bat file (in Windows File Explorer) to run it manually on a computer with Python & ArcGIS installed.**



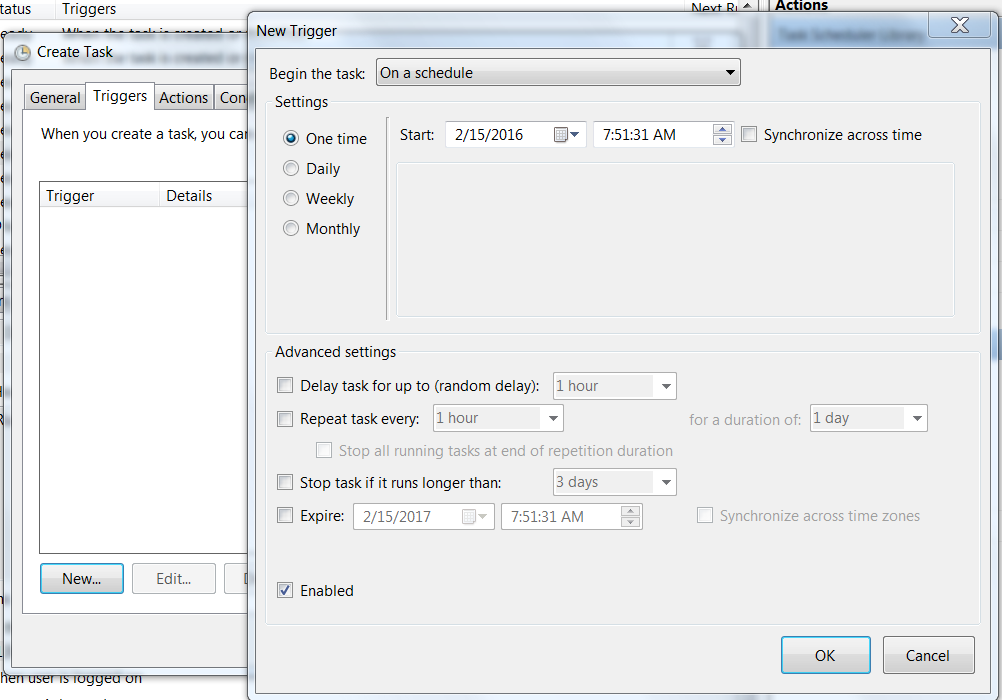
## Step 4: Scheduled task

At the Windows start menu search box, **type Task Scheduler, right click and Run as Administrator**.

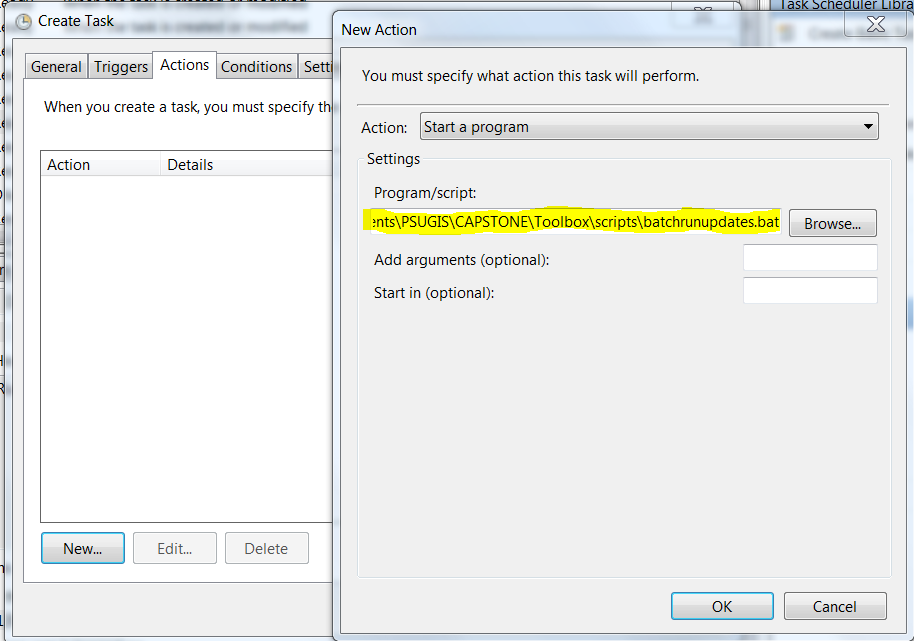
**Create a task** (not a basic task), choosing when to run, how often and what to run (your bat file).



Go to the Triggers tab and set up a schedule if desired. You can also *not* create a trigger, to run manually.



On the Actions tab, set the action to Start a Program, and then browse or paste in the path to your batch file. No arguments are necessary.

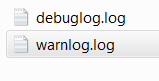


You can experiment with the other settings, but those are the ones that matter. The task may pop up a command window during execution, which you can ignore. If the window stays open, there has been an error.

## Step 5: Monitoring success/failure

I have configured the batch file to stay open with a warning message if the main script fails. At that point, you would go into the logs folder and review the logs. Also, I have created a function to send you a message via email if you have the ability to send SMTP email on your network. We would edit the sendmessage.py file with your SMTP server hostname, port, and email address.

There are two log files, which are created and updated automatically. The warn log holds only warnings and errors. The debug log has all the details. These files can be viewed in a text editor.



The following are samples from the two logs. They have the timestamp, the level of the logging message, the name of the module generating the entry, and the message. The warn log will be empty if there are no warnings or errors.

