**Species list processing tutorial**

This tutorial provides the steps needed to process new species lists generated through the search protocol and also any updates or changes to master taxonomy tables before updating the database. It is important to note that updating the species lists and the taxonomy tables follow an almost identical workflow with a few minor differences that will be outlined below.

Note: Updates to the database will occur bi-weekly providing new data has been added to the temporary data files in GitHub according the search protocol tutorial.

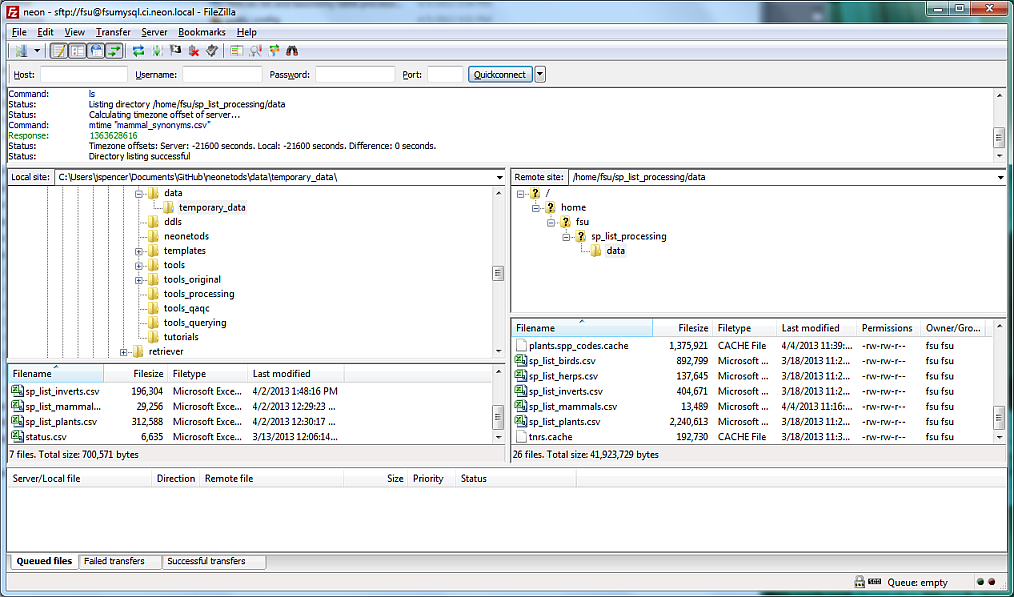
Note: Before using this tutorial you will need to execute the steps in the species list pre-processing tutorial.

The lists you will most likely be working with are in the temporary data folder in GitHub. If you need to make changes to the taxonomy tables these are simply in the data folder tagged as a taxonomy list.

1. The first thing you need to do is transfer the updated files over to the server using Filezilla. When you open the program make sure you are connected to the neon server. To do this just click the ‘Server’ drop down menu and hit reconnect. Remember in the window the left side is your local machine and the right side is the server.
   1. Navigate to the temporary\_data folder in your local GitHub repository and then to the data folder in the fsu directory on the server. The file path’s are shown in the image below:

Local: C:\Users\jspencer\Documents\GitHub\neonetods\data\temporary\_data\

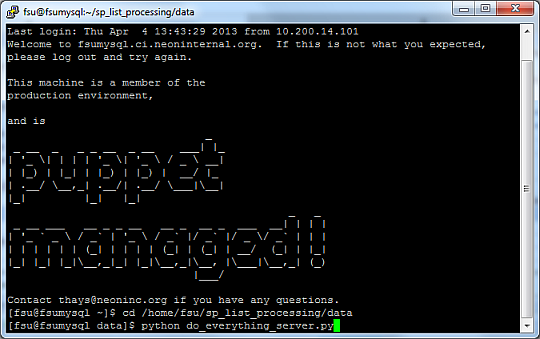
Server: /home/fsu/sp\_list\_processing/data



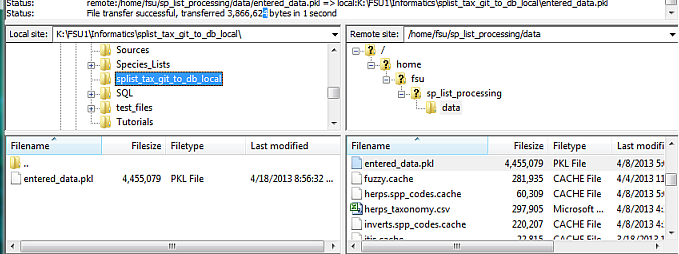
* 1. Drag and drop the species lists that you have updated from the local machine (left side of window) to the data folder in the server (right side of window). It will ask you if you want to replace the existing species list, click yes. Now you should have all the data you want to add to the database on the server.

NOTE: if you are adding changes to a taxonomy table you need to delete the spp\_codes cache that corresponds with the taxonomy table. These chaches hold the generated species ids from each taxonomy table. The file name will look like this: plants.spp\_codes.cache and will be a CACHE file.

1. Now that you have all of the data you wish to add to the dodobase on the server, open Putty login and navigate to the tools folder: cd /home/fsu/sp\_list\_processing/tools.
   1. Start python simply by typing python
   2. Before hitting enter tell putty to run the script do\_everything\_server.py. To do this simply type the file name in after the word python you typed in the previous step as in the screen below (this will actually start python and run the script in one command). Running the script can take a while so I would suggest doing something else for about 10-15 minutes.

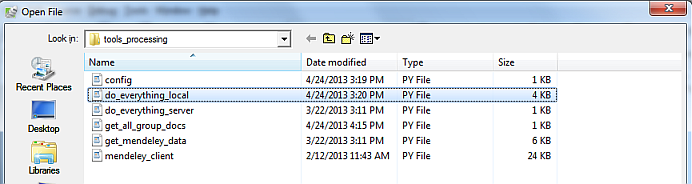


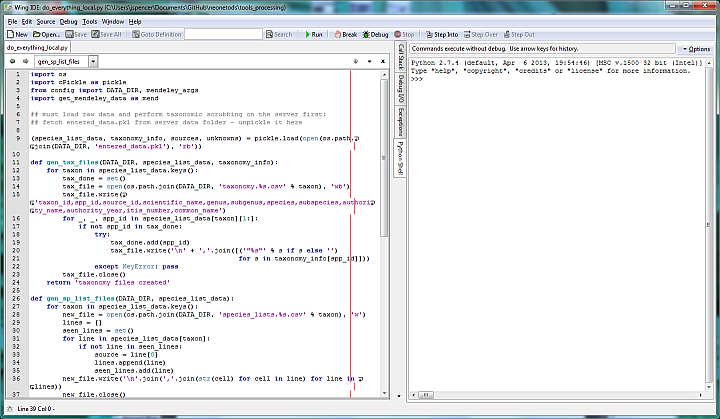
* 1. When the script is done running you will know because activity will have stopped in the putty shell.
  2. When the script is done running navigate to the data folder in the server /home/fsu/sp\_list\_processing/data. You should see a file “entered\_data.pkl” with today’s date. This is file that was generated by running the script on the server.
  3. Once you have located the .pkl file navigate on the left side of the window (your local machine) to the splist\_tax\_git\_to\_db file located in Informatics: K:\FSU1\Informatics\splist\_tax\_git\_to\_db\_local\. Next drag and drop the updated entered\_data.pkl file into this folder. Replace any older files with new one.



* 1. Now exit out of filezilla and navigate to the splist\_tax\_git\_to\_db folder on your machine to make sure the file transfer was successful.

1. Run the code on your local machine to generate the scrubbed species lists:
   1. Open WingIDE and click the open tab in the upper left corner. Navigate to your GitHub local repository and open the tools\_processing folder (C:\Users\jspencer\Documents\GitHub\neonetods\tools\_processing) and double click on the do\_everything\_local.py file. This will open the script in Wing.



When the script is open your screen should look like the one below:

* 1. Hit the green arrow at the top of your screen that says “Run” next to it. This will run the script in Python. Again this will take about 15 minutes.
  2. When the script is finished running look in splist\_tax\_git\_to\_db. If the script ran properly you should see the scrubbed species list and taxonomy CSV files.

1. Updating the database with the scrubbed data:
   1. If you have updated a taxonomy table make sure to update those tables first. The database is set up in such a way that a spp\_id cannot exist in a species list without it being in the associated taxonomy tables.
   2. The code and process for updating the species lists and the taxonomy tables perform the same function with a few differences (taxonomy code is included at the end of the tutorial). The instructions follow:
      1. For the list you are updating create a temporary table in the species list or taxonomy schema named something like birds\_update\_temp. This table’s structure should be identical to the original table you are updating.

DROP TABLE IF EXISTS species\_lists.birds\_update\_temp CASCADE;

CREATE TABLE species\_lists.birds\_update\_temp

(

source\_id varchar(1000) ,

site\_id varchar(4) ,

spp\_id varchar(255)

);

COMMIT;

* + 1. Import the scrubbed version of the list from the splist\_tax\_git\_to\_db folder into the temporary table using wbimport :

wbimport

-file=K:\FSU1\Informatics\ splist\_tax\_git\_to\_db \ species\_lists.birds.csv

-table=species\_lists.birds\_update\_temp

-delimiter=,

-header = TRUE;

COMMIT;

* + 1. Check in the temporary table to make sure it has been populated with the data.
    2. Next append the data from the temporary table into the new table ignoring (not adding) any duplicates. To do this you must create a rule telling the database to ignore any primary key violations as is works through the data. Append with the following code:

CREATE RULE birds\_spp\_list\_duplicate\_ignore AS ON INSERT TO species\_lists.birds

WHERE (EXISTS (SELECT 1 FROM species\_lists.birds

WHERE (source\_id, site\_id, spp\_id) = (NEW.source\_id, NEW.site\_id, NEW.spp\_id))) DO INSTEAD NOTHING;

COMMIT;

INSERT INTO species\_lists.birds SELECT \* FROM species\_lists.birds\_update\_temp;

DROP RULE birds\_spp\_list\_duplicate\_ignore ON species\_lists.birds;

COMMIT;

DROP TABLE species\_lists.birds\_update\_temp;

COMMIT;

The code above allows the database to add records while ignoring duplicates and primary key violations. Thus it only adds records that are not in the database already eliminating the insertion of duplicate records. While updating the table if there are any spp\_id’s not in the taxonomy table it will give you an error and you will have to update the taxonomy table with that species and species id. After the update has occurred be sure do delete your temporary tables.

Update Taxonomy Code:

DROP TABLE if exists taxonomy.plants\_update\_temp CASCADE;

CREATE TABLE taxonomy.plants\_update\_temp

(

taxon\_id varchar(255),

spp\_id varchar(255),

source\_id varchar(255),

scientific\_name varchar(255),

genus varchar(255),

subgenus varchar(255),

species varchar(255),

subspecies varchar(255),

authority\_name varchar(255),

authority\_year numeric,

itis\_number numeric,

common\_name varchar(255)

);

COMMIT;

wbimport

-file=K:\FSU1\Informatics\splist\_tax\_git\_to\_db\_local\taxonomy.plants.csv

-table=taxonomy.plants\_update\_temp

-delimiter=,

-header = TRUE;

COMMIT;

CREATE RULE plants\_tax\_duplicate\_ignore AS ON INSERT TO taxonomy.plants

WHERE (EXISTS (SELECT 1 FROM taxonomy.plants

WHERE (spp\_id) = (NEW.spp\_id))) DO INSTEAD NOTHING;

COMMIT;

INSERT INTO taxonomy.plants SELECT \* FROM taxonomy.plants\_update\_temp;

DROP RULE plants\_tax\_duplicate\_ignore ON taxonomy.plants;

COMMIT;

DROP TABLE taxonomy.plants\_update\_temp;

COMMIT;

**Update Sources (sources.sources**)

DROP TABLE IF EXISTS sources.sources\_update\_temp CASCADE;

CREATE TABLE sources.sources\_update\_temp

(

source\_id varchar(1000) NOT NULL,

info\_type varchar(255),

file\_type varchar(255),

notes text,

isbn varchar(255),

author varchar(255),

title varchar(1000),

journal varchar(255),

volume varchar(255),

issue varchar(255),

pages varchar(255),

year integer,

url varchar(255),

tags varchar(1000),

spat\_scale varchar(255),

spat\_extent varchar(255)

);

COMMIT;

wbimport

-file=K:\FSU1\Informatics\splist\_tax\_git\_to\_db\_local\sources.sources.csv

-table=sources.sources\_update\_temp

-delimiter=,

-header = TRUE;

COMMIT;

CREATE RULE sources\_duplicate\_ignore AS ON INSERT TO sources.sources

WHERE (EXISTS (SELECT 1 FROM sources.sources

WHERE (source\_id) = (NEW.source\_id))) DO INSTEAD NOTHING;

COMMIT;

INSERT INTO sources.sources SELECT \* FROM sources.sources\_update\_temp;

DROP RULE sources\_duplicate\_ignore ON sources.sources;

COMMIT;

DROP TABLE sources.sources\_update\_temp;

COMMIT;