This document describes a generalized process for downloading data files from a remote source on a regular basis and then moving those files through various directories and into a database staging table for import into the data warehouse.

At a base level, the application needs to:

1. Get a file from a remote source.
2. Convert that file into a CSV.
3. Move that file into a processing folder.
4. Import that file into a staging table.
5. Archive that file with a date time stamp.
6. Allow the file to be reprocessed.

Below are some other requirements in no particular order.

1. Be able to pull a file from any remote source using a URL as input.
2. URLs should be predefined in a database or a file of some sort.
3. Be able to convert that file into a CSV if necessary.
4. Be able to move that file from the landing folder to the processing folder.
5. From the processing folder the data should be moved to a staging table.
6. The file then should be moved to the archive with some sort of timestamp as part of the file name.
7. The process should be built in such a manner as it loops over every file in the directory.
8. The process should be built in such a manner that reprocessing files from the archive is just a matter of moving them out of the archive and back to the processing folder. This might prove challenging because of the time stamp requirement.
9. The process needs to be efficient enough to process large amounts of data.
10. The process needs to be able to send email alerts when it encounters various issues like an empty directory when a file is expected.
11. The process, of course, requires basic error handling.
12. Process should have the ability to discriminate the kinds of files it processes, xml, vs, csv, txt etc.
13. Process should use config files for settings.

Within the QEPM repo there are several python scripts that you can use as examples.

**LoadAssetPrices.py**

This is the prime example script. This is the script that currently imports end of day stock price data. It’s a good process but specialized to this particular task. It needs to be more generalized. In this script are several methods you’ll want to alter.

Main – this is the general controller that kicks things off and controls general execution. This is not the proper way to build a class in Python. It’s just stolen from Java.

Move\_file – moves the file from one directory to another. Notice how the method takes a source and a target so it’s generalized to move a file anywhere.

Load\_file – this loads the file into a staging table and will be the hardest to generalize. This is NOT the correct method to do this, however, it was written before I understood Pandas pretty well.

Send\_exception\_email – every ETL process needs some sort of notification. In this case, a notification goes out if there are no files in a directory.

**FinancialDataTest.py**

This is a script to test downloading data from Quandl using the Python API.

**DownloadFiles.py**

This is a good place to start with a design. In here are all the imports I’m currently doing. Notice that Yield Curve data doesn’t come from Quandl. This is a good example of a file that has to be converted to CSV first.

**ParseAndStoreYieldCurveData.py**

This took forever to write. It parses the Yield Curve XML file into a CSV. Parsing XML is a highly specialized task and that script will ONLY work with the YieldCurve data coming from that particular source.