



# Massachusetts Parcel Collector: townships to county accumulator

**Weeks of manual work transformed into 1 minute of FME**

Robert Raiz – Senior Manager at Yardi

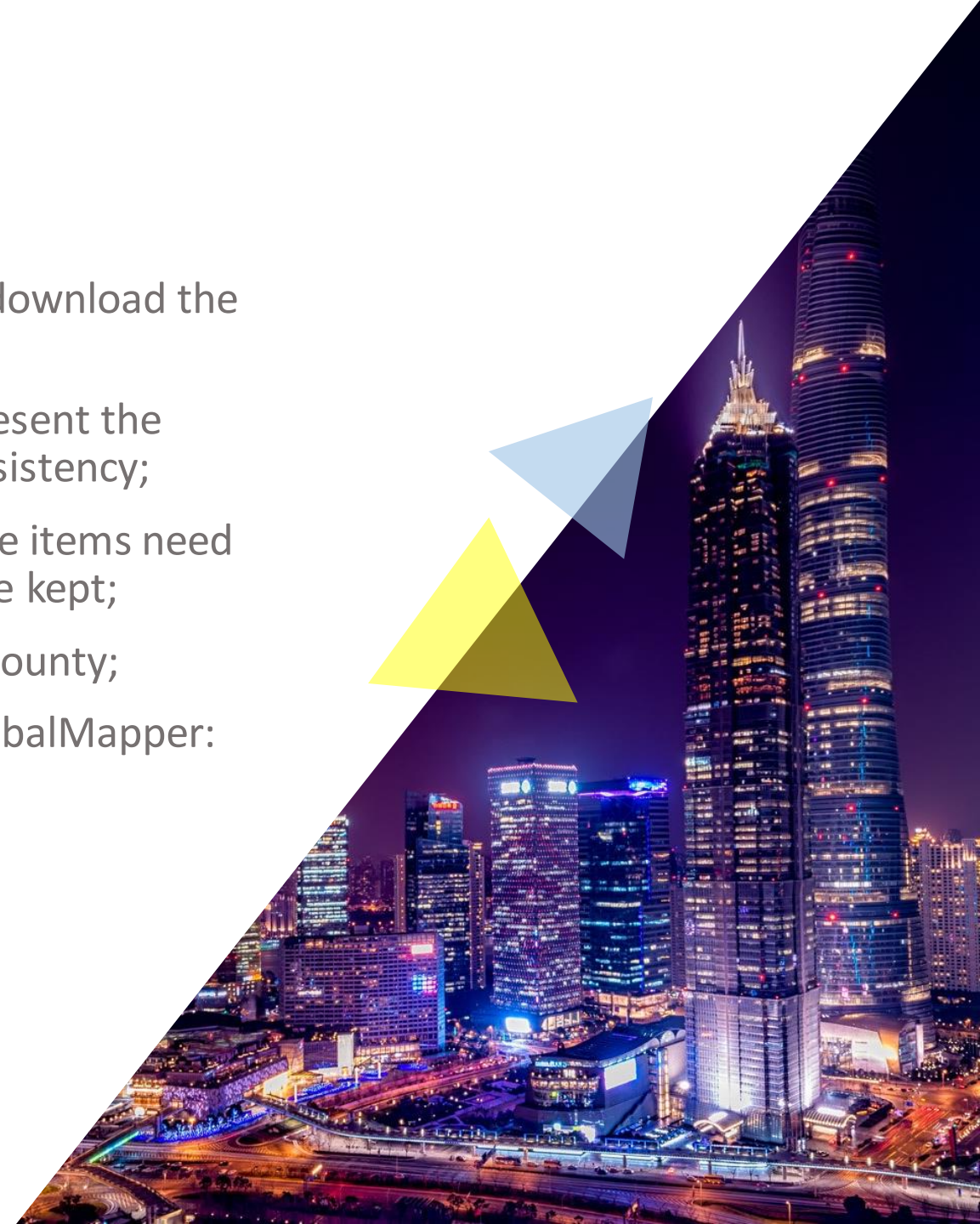
# A few key terms

- 01 State
- 02 County
- 03 Township
- 04 Parcel
- 05 Parcel ID/Township ID/Long PID



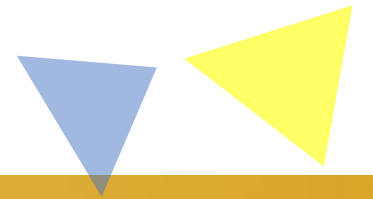
# The Problem

- 14 counties to complete: 350 townships to manually download the parcel data;
- All of the townships might use a different way to represent the parcel id: the data needs manual intervention for consistency;
- Many redundant information and extra columns: some items need to be removed and only useful attribute columns to be kept;
- It used to take a week to complete the maps for one county;
- We were using a combination of MapServer/QGis/GlobalMapper: too many tools;



## Solution

Use Python and FME to automate the workflow;  
One week/county was reduced to a few minutes;





# The Workflow - part 1: using the FME Python start-up script function

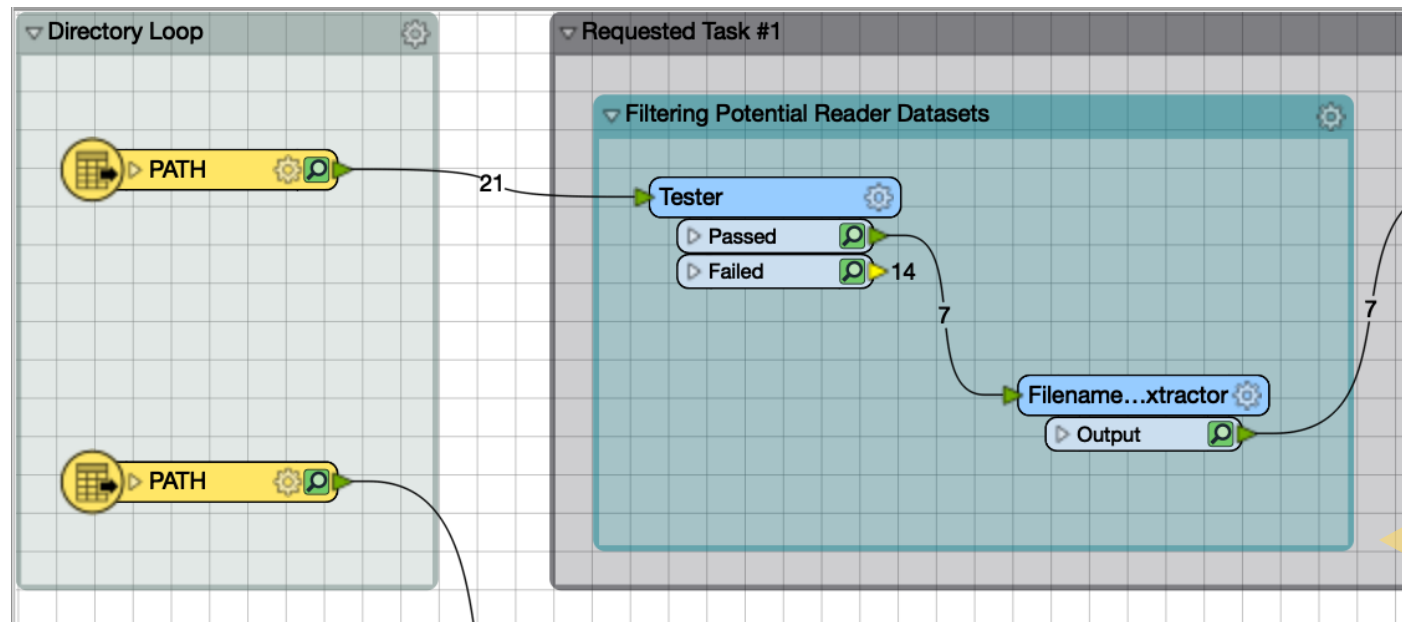
- ▶ Using a basic Python script (v3.5) we create the county\_name variable as an FME\_MacroValue to allow the user to select the county of interest;
- ▶ The script will access the county data by using the county to township dictionary and start downloading the files;
- ▶ as Boston has a different place of storing the parcels, we use that path separately in the script, if chosen by the user (Suffolk County);
- ▶ all the data is Open Source provided for free usage by the state of Massachusetts;

```
1 import urllib.request
2 import zipfile
3 import os
4
5 county_name = FME_MacroValues['MYCOUNTY']
6 dest_path = FME_MacroValues['SourceFolder']
7 path = "{}/{}".format(dest_path, county_name)
8
9 if not os.path.exists(path):
10     os.makedirs(path)
11
12 county = {
13     "Barnstable": ["M041_BREWSTER", "M036_BOURNE", "M020_BARNSTABLE", "M075_DENNIS", "M086_EASTHAM", "M055_CHATHAM",
14     "Berkshire": ["M004_ADAMS", "M006_ALFORD", "M022_BECKETT", "M070_DALTON", "M090_EGREMONT", "M058_CHESHIRE", "M063_
15     "Bristol": ["M016_ATTLEBORO", "M003_ACUSHNET", "M027_BERKLEY", "M072_DARTMOUTH", "M076_DIGHTON", "M088_EASTON", '
16     "Dukes": ["M104_AQUINNAH", "M089_EDGARTOWN", "M062_CHILMARK", "M109_GOSNOLD", "M221_OAKBLUFFS", "M296_TISBURY
17     "Essex": ["M007_AMESBURY", "M009_ANDOVER", "M030_BEVERLY", "M038_BOXFORD", "M071_DANVERS", "M092_ESSEX", "M105_GI
18     "Franklin": ["M013_ASHFIELD", "M029_BERNARDSTON", "M047_BUCKLAND", "M053_CHARLEMONT", "M068_CONWAY", "M074_DEEF
19     "Hampden": ["M005_AGAWAM", "M043_BRIMFIELD", "M033_BLANDFORD", "M085_EASTLONGMEADOW", "M059_CHESTER", "M061_I
20     "Hampshire": ["M008_AMHERST", "M024_BELCHERTOWN", "M069_CUMMINGTON", "M087_EASTHAMPTON", "M060_CHESTERFIELD",
21     "Middlesex": ["M010_ARLINGTON", "M012_ASHBY", "M014_ASHLAND", "M002_ACTON", "M031_BILLERICA", "M049_CAMBRIDGE", "I
22     "Nantucket": ["M197_NANTUCKET"],
23     "Norfolk": ["M018_AVON", "M040_BRAINTREE", "M046_BROOKLINE", "M050_CANTON", "M025_BELLINGHAM", "M073_DEDHAM", "M
24     "Plymouth": ["M001_ABINGTON", "M042_BRIDGEWATER", "M044_BROCKTON", "M052_CARVER", "M082_DUXBURY", "M083_EASTBR
25     "Suffolk": ["M057_CHELSEA", "M248_REVERE", "M346_WINTHROP"],
26     "Worcester": ["M011_ASHBURNHAM", "M015_ATHOL", "M017_AUBURN", "M028_BERLIN", "M032_BLACKSTONE", "M039_BOYLSTON
27 }
28
29 for i in county[county_name]:
30     link = 'http://[REDACTED]' + i + ".zip"
31     urllib.request.urlretrieve(link, path + i + r"_parcels.zip")
32     with zipfile.ZipFile(path + i + r"_parcels.zip", "r") as z:
33         z.extractall(path)
34
35 boston_path = "http://[REDACTED].zip"
36
37 if county_name == "Suffolk":
38     urllib.request.urlretrieve(boston_path, path + r"Boston_Parcels2017.zip")
39     with zipfile.ZipFile(path + r"Boston_Parcels2017.zip", "r") as b:
40         b.extractall(path)
```

## The Workflow - part 2: Input Data via directory loop, tester and filename extractor

the process is quite straightforward: we add a “Directory and File Pathnames” FME Reader and set a few parameters to pull only shapefiles;

we test the files via the “TESTER” feature as to pull only Parcel data;  
-finally, for this stage, we use the “FileNamePartExtractor” feature to save the city for which the parcels were pulled;

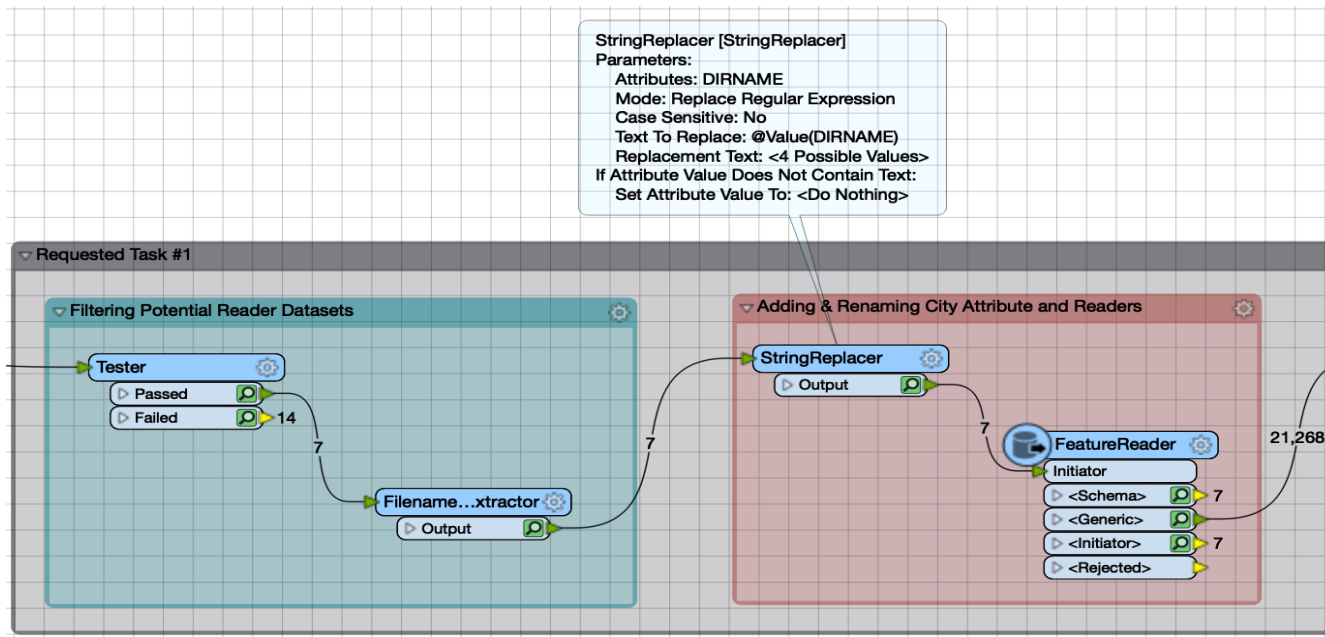


# The Workflow - part 3: using FME's "stringreplacer" feature via the regex capability

▶ the "FileExtractor" did its part but we need only the city name, not the different codes that come with it;

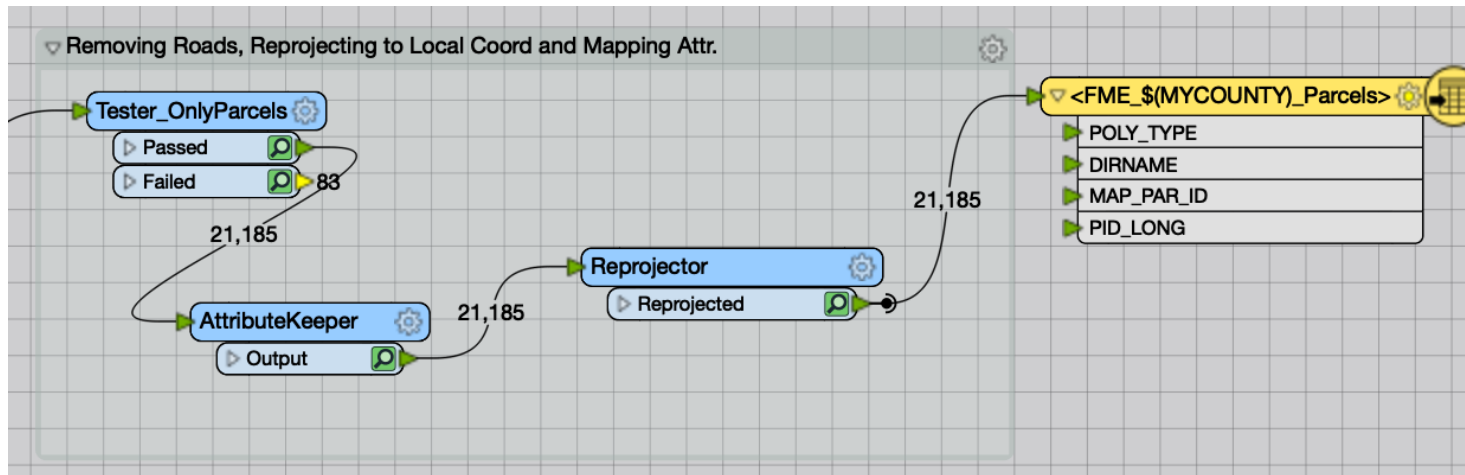
▶ for eg: "L3\_SHP\_M062\_Chilmark" would need to end up as "Chilmark"; this information will be stored as an attribute value for the city name;

▶ there are several ways of doing this in FME but we have chosen the Regular Expression option because we really like Regex;



## The Workflow - part 4: Removing Roads, Reprojecting to Local Coord and Mapping Attributes

- 01** the cities provide shapefiles for roads too, in the initial dataset so those need to be removed
- 02** reprojecting to local coordinates and mapping the attributes to their final position;
- 03** obtain the final county shapefile





# Bonus request for Suffolk: Boston Parcels impacted by airport noise

## Request

extract a list of addresses from Boston, in an Excel format, with all the single family properties that are impacted by a noise level that is at least 50dB;

## Available Files

- Suffolk county parcel data obtained from the previous steps;
- Boston Addresses (point data in a geoJson file);
- Boston Building Footprints (polygon data in a geoJson file);
- Airport Noise Level (digitized from PDF map, presented as polygon file in a geoJson format);
- Boston Zoning info (proprietary data ©PropertyShark.com);

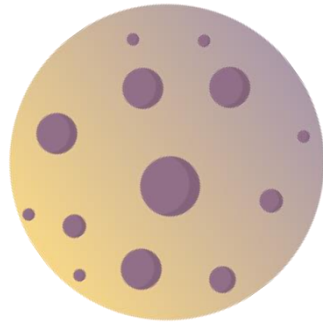
## Steps To Follow

Use FME to overlap the above datasets and pull the list of properties;





Live DEMO – FME at work



# Thank you for your attention

Robert Roland Raiz