ORMap Map Production Tools

2018-APR-02 Brian Wilson

To produce taxmap pdfs you need three things:

1. An ArcMap “Map Document” with the right stuff in it; henceforth called “the MXD”.
2. The data, currently consisting of a File Geodatabase (“the FGDB”), a few shapefiles, and one Excel spreadsheet.
3. A “Python Toolbox”, which I call “the Python Toolbox”.

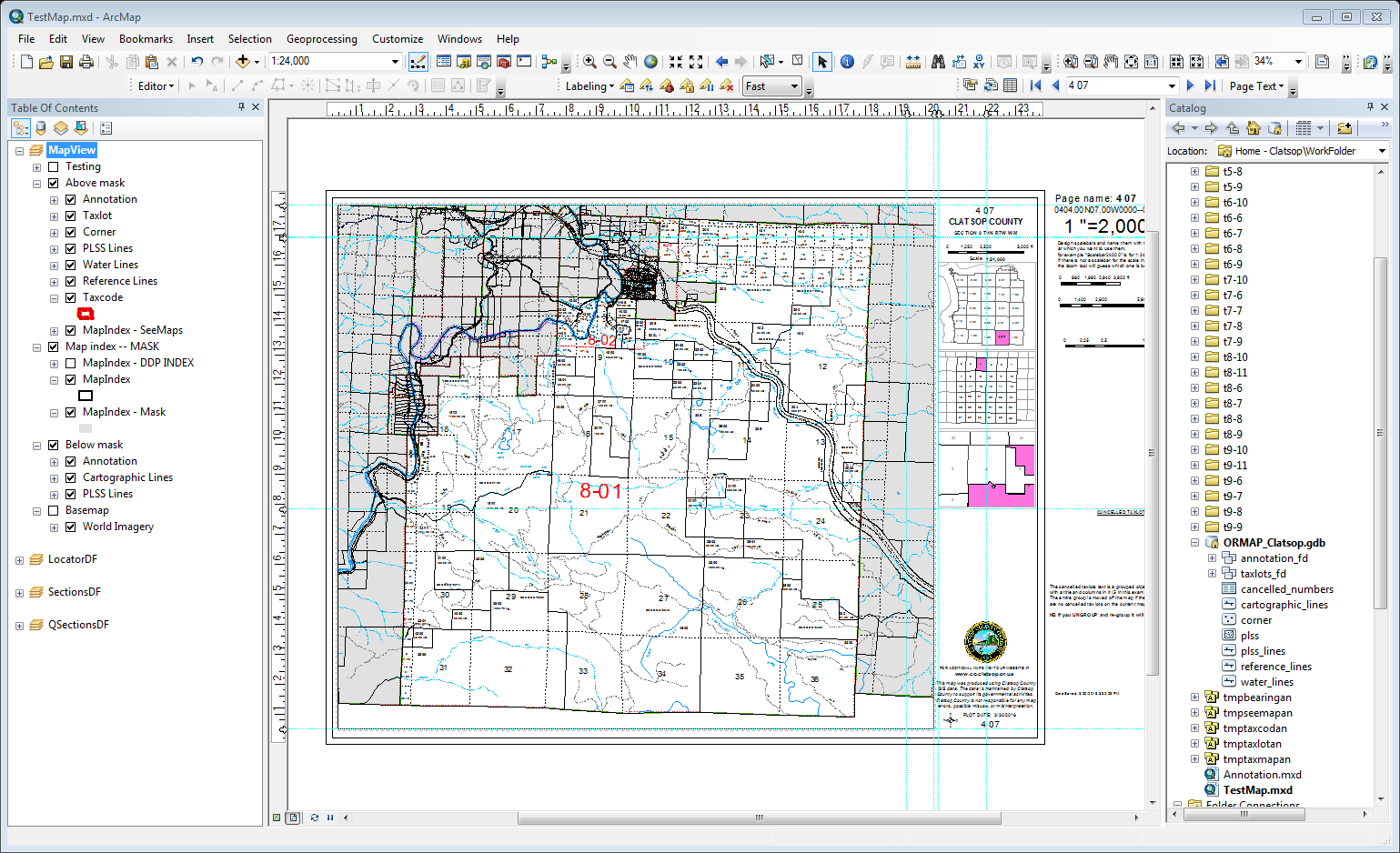
The “taxmap” I refer to here is a single page PDF (or JPEG) document defined by an entry in the “mapindex” feature class in the FGDB.

Most of the data can currently be found in the ORMAP\_Conversion folder on K: drive. There are a few shapefiles in K:\LISdata too. All the data will be moving to ArcGIS Server soon.

# Overview

## MXD

The only MXD I have set up right now is called “TestMap.mxd”. It’s intended to serve as a starting point for a production map MXD.



## Data

### What’s in the FGDB

The FGDB for this project contains data converted from coverages in K:\taxmaped\Clatsop\towned.

### What’s NOT in the FGDB

Currently I reference a couple shapefiles in the MXD to make the locator maps look good. The shapefiles used are PLSS polygons and OR/WA county lines.

The Python Toolbox reads the “cancelled taxlots” Excel spreadsheet stored in **K:\taxmaped\Clatsop\towned\cancelled.xlsx**

There is a basemap in the MXD which references ESRI’s “World Imagery”. It’s normally turned off. It’s there to help debug the map.

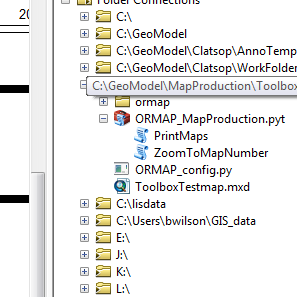
## Python To*o*lbox

This is a set of files in a folder called Toolbox. You can copy the folder and rename it as you wish, just keep the files inside it together. The file that is read from ArcMap is called ORMAP\_MapProduction and has an extension of “pyt”. ORMAP\_MapProduction references a configuration file called ORMAP\_config.py, and loads a bunch of python files from the ormap subdirectory.

The ORMAP\_config.py has settings that reference features in TextMap.MXD. If you base any MXD documents you create on TestMap.mxd you probably won’t have to touch it.

# Using the Python Toolbox

The tools in the Python Toolbox **only** work in ArcMap when a valid map is open; you cannot run it standalone from ArcCatalog. I normally have the catalog sidebar open in ArcMap and run it from there. It looks like this:



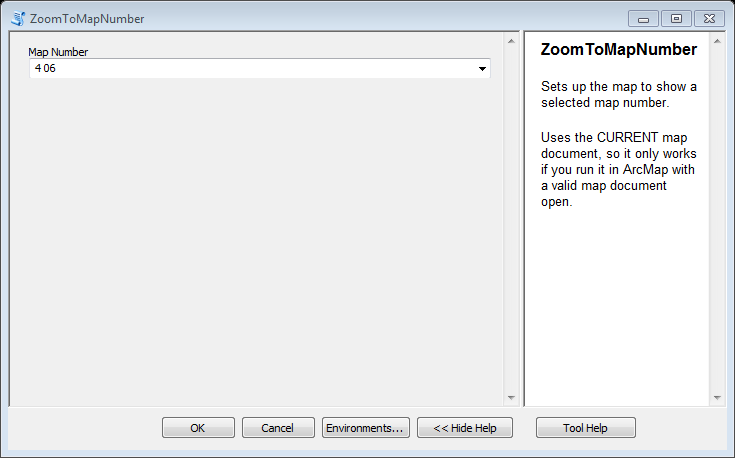
A “valid map” means one that has been set up with data driven pages and has fields in its layout that the zoom tool can fill in. I will describe that in more detail later. The map I created while building the tool, “TestMap.mxd” is the only current example.

You can copy the toolbox to your C: drive or wherever is convenient for you. Just make sure you keep the folder contents together; ORMAP\_MapProduction.pyt looks for a config file called **ORMAP\_config.py** and subdirectory full of supporting python files called **ormap.**

## The tools

### ZoomToMapNumber tool

Double click the ZoomToMap tool to launch it. It looks like this:



To use it, select (or type) a Map Number, then click OK

### What ZoomToMapNumber does

The tool uses the map number you selected to look up a page in the index layer defined in Data Driven Pages. Then it adjusts the main dataframe (MapView in this map) to show that page.

Most of the work is done by the Data Driven Page feature of ArcMap.

Changing the current Data Driven Page is that it will also change the any layers that have “Page Definition Queries” set up. That includes the “mask”, which covers over all the areas that are not inside the current map page polygon.

Data Driven Pages can pull attributes from the index layer and display them on the page layout. I added attributes to the Map Index layer to make this work.

These attributes include

MapNumber--

PageName – This is the string used to uniquely identify each taxmap, in addition to the MapNumber string it has a suffix if the map is a detail map, like “4 10 6BC D1”

MapScale

Angle

LongMapTitle

ShortMapTitle

There are others that might be useful including ORMAPNUM, Township, Range, Section, Quarter

So far you could get all these results just using built in Data Driven Pages. You can even use the control bar to change pages…

***But the zoom tool gives you more!***

The zoom tool also changes a bunch of settings in the sidebar on the map including

can\_group

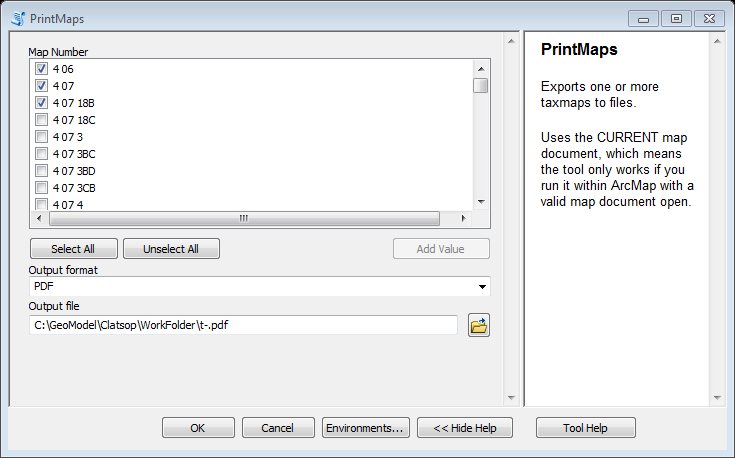
PlotDate

The scalebar

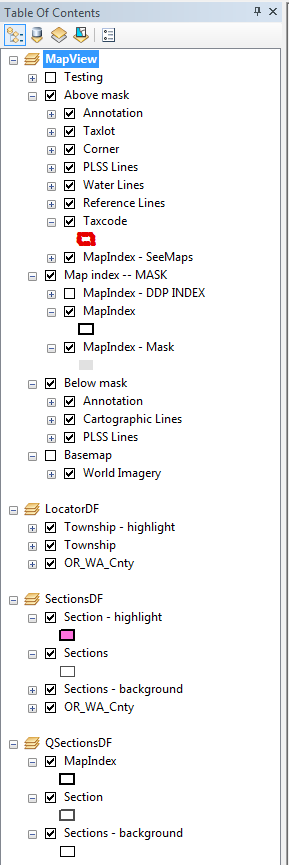
If you use the controls for Data Driven Pages directly (instead of using the zoom tool), you can see the difference. The main map will update, some of the text will update, but the locator maps and the scalebar will not.

Even though everything is not updated, I find it handy to use the DDP controls sometimes, to quickly page through the taxmaps.

### PrintMaps tool



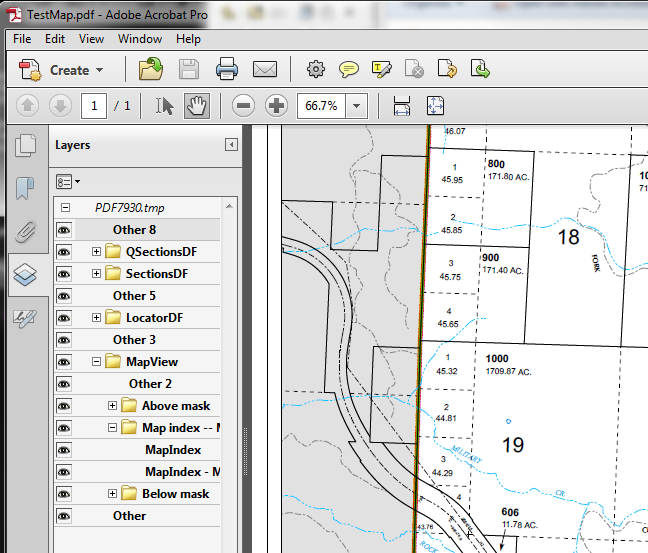
# Configuration details



### Note regarding the PDF files

When you have a PDF open in Adobe Acrobat, you can turn off the “mask” layer. I think this is kind of cool because it means you can still see all the annotation on adjacent maps if you want to, but when you print the map with the mask turned on, you still get a standard ORMAP map.

With mask turned on (normal) on a taxmap



Now click the “eye” next to the “Map index – Mask” group to turn the mask off. This reveals annotation and features outside this taxmap.

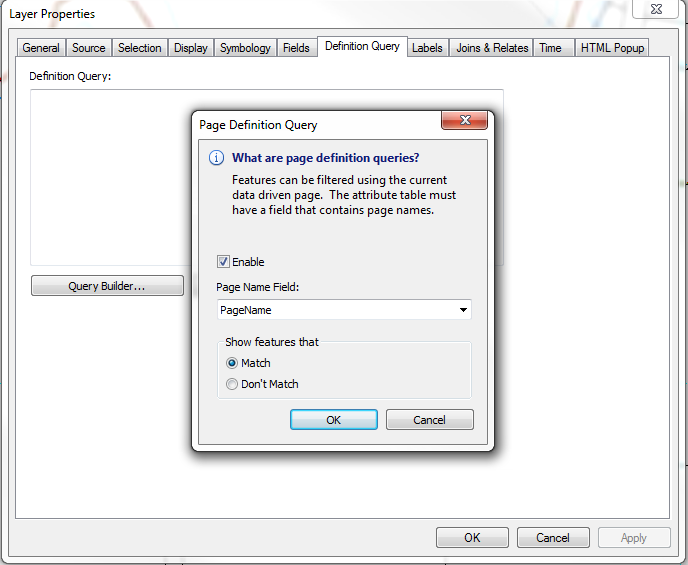
## 

## Data Driven Pages set up

The feature class in the FGDB called “mapindex” (inside the TaxlotsFD feature dataset) is used to control the “data driven pages” feature in the map.

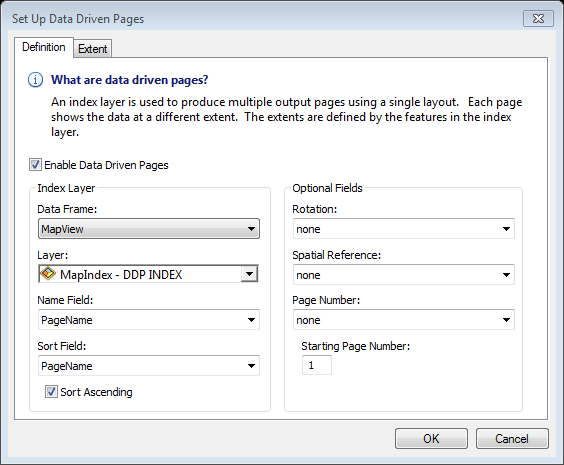
“Mapindex” is included in the map document in the “Map index – MASK” group, three times.

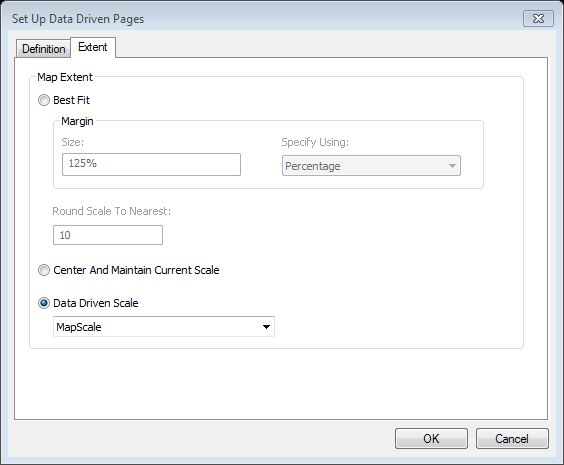
This is so it can be symbolized three different ways. The first layer “Mapindex – DDP Index” is not normally turned on. I use it as the layer in the “Set Up” for Data Driven Pages (see screenshots below).

The second layer “MapIndex” is used to draw a black line around the current map. Because it has a “Page Definition Query” defined, only the current map is outlined. This is what that looks like in the properties for the layer:

### The mask layer

The third layer “MapIndex – Mask”, also uses a Page Definition Query, but uses the “Don’t Match” option. This causes all of the polygons outside the currently selected one to turn on, creating the opaque grey mask that obscures all the data in the “Below mask” group.





## Python file “ORMAP\_config.py”

Here is the content of the current ORMAP\_config.py file. This file is used to tell the Python Tools a few things I have not figured out how to put into the MXD.

As long as you don’t change the names of the dataframes used to hold the locator maps (LocatorDF, SectionsDF, and QSectionsDF) or the names of the layers used to control the locators, you probably won’t have to change it.

You can change symbology, colors, locations, things like that, as long as you don’t change the dataframe and layer **names** you should not have any problem.

# ORMAP\_config.py

# Clatsop County

# 2018-03-30 -- folded in the two config files, layout and map

# 2017-12-11 -- Brian converted this file from MapProduction18x24.ini (then deleted 95% of it!)

##### Locator map data frame #####

LocatorDF = "LocatorDF"

LocatorScale = 800000

LocatorLayers=[

("Township - highlight",

'"TR=\'{0}{1}{2}{3}\'".format(int(orm.township), orm.township\_dir, int(orm.range), orm.range\_dir)'),

("Township",

'"TR<>\'{0}{1}{2}{3}\'".format(int(orm.township), orm.township\_dir, int(orm.range), orm.range\_dir)'),

]

LocatorExtentLayer = None # Don't pan this locator map. It always shows the whole county.

LocatorFeatureCount = None

##### Sections map data frame #####

SectionDF = "SectionsDF"

SectionScale = 180000

SectionLayers=[

("Section - highlight",

'"TOWN = \'{0}\' AND RANGE = \'{1}\' AND SECTION = \'{2}\'".format(orm.township, orm.range, orm.section)'),

("Sections",'"TOWN = \'{0}\' AND RANGE = \'{1}\'".format(orm.township, orm.range)'),

]

# Pan locator map to this selection

SectionExtentLayer = SectionLayers[1][0]

# If there are no features showing (due to query definition) then hide this locator map

SectionFeatureCount = SectionLayers[0][0]

##### Quarter sections map data frame #####

QSectionDF = "QSectionsDF"

QSectionScale = 50000

QSectionLayers=[

('Section',

'"TOWN = \'{0}\' AND RANGE = \'{1}\' AND SECTION = \'{2}\'".format(orm.township, orm.range, orm.section)'),

('Sections - background',

'"NOT (TOWN = \'{0}\' AND RANGE = \'{1}\' AND SECTION = \'{2}\')".format(orm.township, orm.range, orm.section)'),

]

# Pan locator map to this selection

QSectionExtentLayer = QSectionLayers[0][0]

# If there are no features showing (due to query definition) then hide this locator map

QSectionFeatureCount = QSectionLayers[0][0]

# ---------------------------------------------------------------------------

# Cancelled Taxlot Numbers appear in a table with the numbers sorted

# vertically. The elements in the table are a title and several columns.

#

# Note if you "ungroup" the elements in arcmap and then regroup them

# you will have to reset the name on the group.

CancelledNumbersTable = "K:/taxmaped/Clatsop/towned/cancelled.xlsx"

MaxCancelledRows = 15 # Go to 8 point font if # of rows exceeds this