Python Map Automation – Beyond the Basics of arcpy.mapping

Jeff Barrette
Jeff Moulds
Basic rules

• Reference an MXD using a path or “current” keyword
  - When using CURRENT
    - Always run in foreground, may need to refresh (e.g., `RefreshActiveView`)

• Uniquely name all the objects (or at least the ones you want to find)

• Pre-author MXDs with all possible elements
  - Can’t create new objects (e.g., north arrow, data frames)
  - Author the extra elements off the page
  - No "New Map" function, so keep an empty MXD available

• This is not a replacement for ArcObjects – we are trying to draw a line in the sand
Sample Applications
Jeff Barrette

http://esriurl.com/5907
**Cloning elements**

- You can clone text and graphic elements
- This allows you to automate things like dynamic tables

```python
vertl = arcpy.mapping.ListLayoutElements(
    mxd, "GRAPHIC_ELEMENT", "VerticalLine")[0]
vertl.elementPositionX = xPos;
vertl.elementPositionY = 4
vertl.elementHeight = 3
for line in range(1, numColumns+1):
    vert_clone = vertLine.clone("_clone")
    xPos = xPos + colWidth
    vert_clone.elementPositionX = xPos
```
arcpy.mapping group on ArcGIS Online

http://esriurl.com/5915
Performance tips

• Don’t keep calling list functions

   ```python
   import map as arcpy.mapping
   item1 = map.ListLayoutElements(mxd,wildcard="Item1")
   item2 = map.ListLayoutElements(mxd,wildcard="Item2")
   item3 = map.ListLayoutElements(mxd,wildcard="Item3")
   ```

• Call them once instead and iterate through the items

   ```python
   for elm in arcpy.mapping.ListLayoutElements(mxd):
       if elm.name =="Item1": item1 = elm
       if elm.name =="Item2": item2 = elm
       if elm.name =="Item3": item3 = elm
   ```
Performance tips (continued)

• Use dictionaries

```python
dict = {}
    for elm in arcpy.mapping.ListLayoutElements(mxd):
        dict[elm.name] = elm

    dict["Item1"].text = "Dictionaries"
    dict["Item2"].text = "are really"
    dict["Item3"].text = "COOL!!!"
```
Functions for web map printing and server publishing

- **ConvertWebMapToMapDocument()**
  - Use with the ArcGIS web APIs for advanced web map printing workflows

- **CreateMapSDDraft()**
  - Automate publishing map documents to map services
Server printing out-of-the-box

- ArcGIS Server and the ArcGIS web APIs support web map printing via print services.
  - Out-of-the-box print service and template maps ship with Server
  - Print services sample: http://esriurl.com/6465

Related Session: Enabling High-Quality Printing in Web Applications (Wednesday @ 12pm – 12:30pm General Theater 3 Exhibit Hall A)
Advanced server printing with arcpy.mapping

- Build web apps with customized versions of the out-of-the-box print service

  Web application → arcpy.mapping → High-quality output (e.g. PDF)

- arcpy.mapping method for converting Web Maps to Map Documents:

  \[
  \text{ConvertWebMapToMapDocument} \quad \text{\{webmap\_json, \{template\_mxd\}, \{notes\_gdb\}, \{extra\_conversion\_options\}\}}
  \]
Advanced server printing with arcpy.mapping

• Full capabilities of arcpy.mapping on the document
  - Swap out service layers for local vector data for **vector PDF output**
  - Export using advanced options
  - Export data driven pages
  - Export to PDF and insert additional pages (title page, reports, etc.)
  - Controlling the appearance of the legend
  - Etc.

• Return a printer-friendly output file (PDF, PNG, etc.)

• Online help and examples  [http://esriurl.com/4600](http://esriurl.com/4600)
Demo: Web app to export vector PDF using arcpy.mapping

- Output or print vector layers instead of “flat” image of service layers
  - Vector layers will be staged in template map document

Map service tiled cache (low dpi)

Vector data (or high dpi image)

Output PDF viewed in Adobe Reader
Demo: Web app to export vector PDF using arcpy.mapping

- Reference the custom arcpy.mapping based GP service
Demo: Web app to export vector PDF using arcpy.mapping

Python code used in custom GP service

```python
import arcpy, os, uuid

# Get web map JSON
Web_Map_as_JSON = arcpy.GetParameterAsText(0)

# Get template MXD
Layout_Template = arcpy.GetParameterAsText(1)

templatePath = '\\gilbert\Austin\Templates\'
templateMxd = os.path.join(templatePath, Layout_Template + '.mxd')

# Convert the WebMap to a map document
result = arcpy.mapping.ConverWebMapToMapDocument(Web_Map_as_JSON, templateMxd)
mxd = result.mapDocument

# Remove service layers
for lyr in arcpy.mapping.ListLayers(mxd, data_frame=df):
    if lyr.isServiceLayer:
        arcpy.mapping.RemoveLayer(df, lyr)

# Export the web map to PDF
output = '\\WebMap{0}.pdf'.format(uuid.uuid1())
Output_File = os.path.join(arcpy.env.scratchFolder, output)

arcpy.mapping.ExportToPDF(mxd, Output_File, georef_info=True)
```

Get web map JSON
Get template MXD
Create new MXD based on web map
Remove service layers
Export PDF
Output file of job
Web app to export vector PDF using arcpy.mapping

- Two tutorials in the help:
  - Basic vector web map printing: http://esriurl.com/4601
  - Advanced web map printing: http://esriurl.com/4602

```
# Reference the legend in the map document
legend = arcpy.mapping.ListLayoutElements(mxd, "LEGEND_ELEMENT")[0]

# Get a list of service layers that are on in the legend because the incoming
# JSON can specify which service layers/sublayers are on/off in the legend
legendServiceLayerNames = [l lyr.name for l lyr in legend.listLegendItemLayers()
   if l lyr.isServiceLayer and not l lyr.isGroupLayer]

legendServiceLayerNames = []
for l lyr in legend.listLegendItemLayers():
   if l lyr.isServiceLayer and not l lyr.isGroupLayer:
      legendServiceLayerNames.append(l lyr.name)
```

```
For example, assume we want to create a list of squares, like:

```
>>> squares = []
>>> for x in range(10):
...     squares.append(x**2)
... squares
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

We can obtain the same result with:

```
squares = [x**2 for x in range(10)]
```

```
squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```
Advanced Server Printing

• Modify arcpy.mapping scripts authored on Desktop and use them in geoprocessing and print services
Advanced Server Printing: new function at 10.3

- **Layer.UpdateLayerFromJSON(json_layer_definition)**
  - Used in web map printing applications that support changing the renderer (or other properties) of dynamic web service layers.
  - Will apply the renderer (or other layer properties) as specified in the webmap_json to the corresponding vector layers staged in the template map document.

```python
# Convert the web map to a map document
result = arcpy.mapping.ConvertWebMapToMapDocument(Web_Map_as_JSON, templateMxd)
mxd = result.mapDocument

# Reference the data frame that contains the web map
def = arcpy.mapping.ListDataframes(mxd, 'Webmap')[0]

# This is the layer that will get updated based on the layer definition in the JSON
lyr = arcpy.mapping.ListLayers(mxd, "U.S. States (Generalized)", df)[0]

# Read the JSON and extract the layer definition
json_data = json.loads(Web_Map_as_JSON)
layerDefinition = json_data["operationalLayers"][1]["layerDefinition"]

# Update the staged vector layer with the layer definition (e.g. renderer info) from the JSON
lyr.updateLayerFromJSON(layerDefinition)
```
Publishing map services with arcpy.mapping

- `arcpy.mapping.CreateMapSDDraft(map_document, out_sddraft, service_name, {server_type}, {connection_file_path}, {copy_data_to_server}, {folder_name}, {summary}, {tags})`

- Workflow to convert map document to map service.

- Use python scripts for:
  - Scheduled service updates. E.g. nightly.
  - Publishing automated analysis results.
  - Batch publishing.

- Create SDDraft file (and optionally edit XML)
  - `arcpy.mapping.CreateMapSDDraft()`

- Stage and Publish Map Service (arcpy server GP tools)
  - `arcpy.StageService_server()`
  - `arcpy.UploadServiceDefinition_server()`
Publishing map services with arcpy.mapping

Sample script: **CreateMapSDDraft**

```python
import arcpy

# define local variables
wrkspc = 'C:\Project/'
mapDoc = arcpy.mapping.MapDocument(wrkspc + 'counties.mxd')
con = 'GIS Servers\arcgis on MyServer_6080 (publisher).ags'
Service = 'Counties'
sddraft = wrkspc + service + '.sddraft'
sd = wrkspc + service + '.sd'
summary = 'Population Density by County'
tags = 'county, counties, population, density, census'

# create service definition draft
arcpy.mapping.CreateMapSDDraft(mapDoc, sddraft, service, 'ARCGIS_SERVER',
    con, True, None, summary, tags)

# analyze the service definition draft
analysis = arcpy.mapping.AnalyzeForSD(sddraft)

# stage and upload the service if the sddraft analysis did not contain errors
if analysis['errors'] == {}:
    # Execute StageService
    arcpy.StageService_server(sddraft, sd)
    # Execute UploadServiceDefinition
    arcpy.UploadServiceDefinition_server(sd, con)
else:
    # if the sddraft analysis contained errors, display them
    print analysis['errors']
```

Reference MXD

Server connection, service properties, etc.

Create and analyze sddraft for errors, warnings, etc.

Stage and publish Map Service

Don’t publish if errors exist

Online help and samples: [http://esriurl.com/4598](http://esriurl.com/4598)
Publishing other service types with python

- Create geoprocessing services
  - `arcpy.CreateGPSDDraft()`

- Create image services
  - `arcpy.CreateImageSDDraft()`

- Create geocoding services
  - `arcpy.CreateGeocodeSDDraft()`
Migrating to ArcGIS Pro

• Help Topic: Migrating arcpy.mapping from ArcMap to ArcGIS Pro
  - Python 3.4
  - ArcGIS project file (.aprx)
  - Stand-alone functions have moved to appropriate classes
    - mapFrame.exportToPDF()
    - map.addLayer(), map.insertLayer(), etc
  - Layer files have changed
  - DataFrame replaced by Map, MapFrame, and Camera
  - New Layout object
  - Application always refreshes when using CURRENT
Thank you…

Please fill out the session evaluation

**First Offering ID:** 1307  
**Second Offering ID:** 1405

Online – www.esri.com/ucsessionssurveys  
Paper – pick up and put in drop box