

DevSummit DC

February 11, 2015 | Washington, DC



Python for Map Automation

James Tedrick

What is arcpy.mapping?

- A map scripting environment introduced at 10.0
- Python mapping module that is part of the arcpy site-package

What can you do with arcpy.mapping?

- **Manage map documents, layer files, & their contents**
 - Find a layer with data source X and replace with Y
 - Update a layer's symbology in many MXDs
 - **Generate reports that lists document information**
 - data sources, broken layers, spatial reference info, etc.
- **Automate the exporting and printing of map documents**
- **Automate map production and create map books**
 - Extend Data Driven Pages capabilities

What **can't** you do with arcpy.mapping?

- Create new symbology from scratch
- Create new layout elements
- Create new map documents (can 'Save As')

- Not designed to be a complete replacement for ArcObjects
- Product team is willing to consider additional improvements – let us know what you're missing!

Referencing Map Documents (MXDs)

- Opening Map Documents (MXD) with `arcpy.mapping`
- Use the `arcpy.mapping.MapDocument` function
 - Takes a path to MXD file on disk or special keyword "CURRENT"
 - Reference map on disk
`mxd = arcpy.mapping.MapDocument(r"C:\some.mxd")`
 - Get map from current ArcMap session
`mxd = arcpy.mapping.MapDocument("CURRENT")`

Layer Symbology

- Update layer symbology from templates
- Reclassify the symbology (updated data)
- Provide custom break values
- Limitations:
 - Cannot change symbology type directly – use `arcpy.mapping.UpdateLayer`
 - Cannot change colors directly – use `arcpy.mapping.UpdateLayer`
 - Once break values manually set, classification set to manual

Limitations

- **Can't create new content; must have pre-built content**
 - Map Documents
 - Layers
 - Data Frames
 - Layouts (text elements*, graphic elements*, north arrow, scale bar)
- *** Text and Graphic elements can be cloned from existing elements**

arcpy.mapping group on ArcGIS Online

<http://esriurl.com/5915>

The screenshot shows a web browser window displaying the ArcGIS Online group page for "arcpy.mapping / Map Automation". The browser's address bar shows the URL: <http://www.arcgis.com/home/group.html?owner=MapAutomationTeam&title=arcpy.mapping%20%2F%20Map%20Automation>. The page header includes the ArcGIS logo and navigation links: FEATURES, PLANS, GALLERY, MAP, HELP. A "Sign In" button and a search bar are also present.

The main content area features a blue banner with the group name "arcpy.mapping / Map Automation" and a "SHARE" button. Below this, a description states: "This group is for those people interested in learning and working with the arcpy.mapping scripting environment. It will serve as a location to download and share sample applications, scripts and other map automation related materials." A "SHARE" button is located above this text.

The "Group Content" section displays a list of items with the following columns: Title, Owner, Rating, Views, and Date. The items listed are:

- DDPwithDynamicTablesAndGraphs_10.1_v1**: Use the new arcpy.mapping cloning capabilities to build dynamic tables in a layout. Code Sample by MapAutomationTeam. Last Modified: April 16, 2013. (1 rating, 4 comments, 931 downloads)
- MultipleElementLayoutManager_10.0_v1**: Rearrange layout elements (inset maps, text, north arrows) on different pages in a map series using a single map document. Code Sample by MapAutomationTeam. Last Modified: October 16, 2012. (0 ratings, 3 comments, 649 downloads)
- ModifyRasterSymbologyWithPythonAddin_10.1_v1**: Use a Python Add-In to modify the raster symbology in a map document. Code Sample by MapAutomationTeam. Last Modified: June 28, 2012. (0 ratings, 0 comments, 92 downloads)
- StripMapsWith2PanelsPerPage_10_v1**: Demonstrates how to display 2 strip map panels on a single page using Date Driven Pages and arcpy.mapping. Code Sample by MapAutomationTeam. Last Modified: June 26, 2012. (0 ratings, 0 comments, 2 downloads)
- MapBookWithGraphicTables_10_v1**: Data Driven Pages are used to control map extent and dynamic text but arcpy.mapping is used to populate dynamic tables.

On the right side of the page, there are social media links for Facebook and Twitter, and a "Group Details" section for "MapAutomationTeam". The details include: Status: public; Contributors: All members; Tags: arcpy, arcpy.mapping, mapping, python, scripting, map automation, automation, Data Driven Pages, DDP, map books, map series; and 3 Members: MapAutomationTeam, jbarrette, mgrossman.

Demo: Managing Layout Elements

Arcpy.mapping & ArcGIS Server

- **CreateMapSDDraft()** – Automate publishing map documents to services

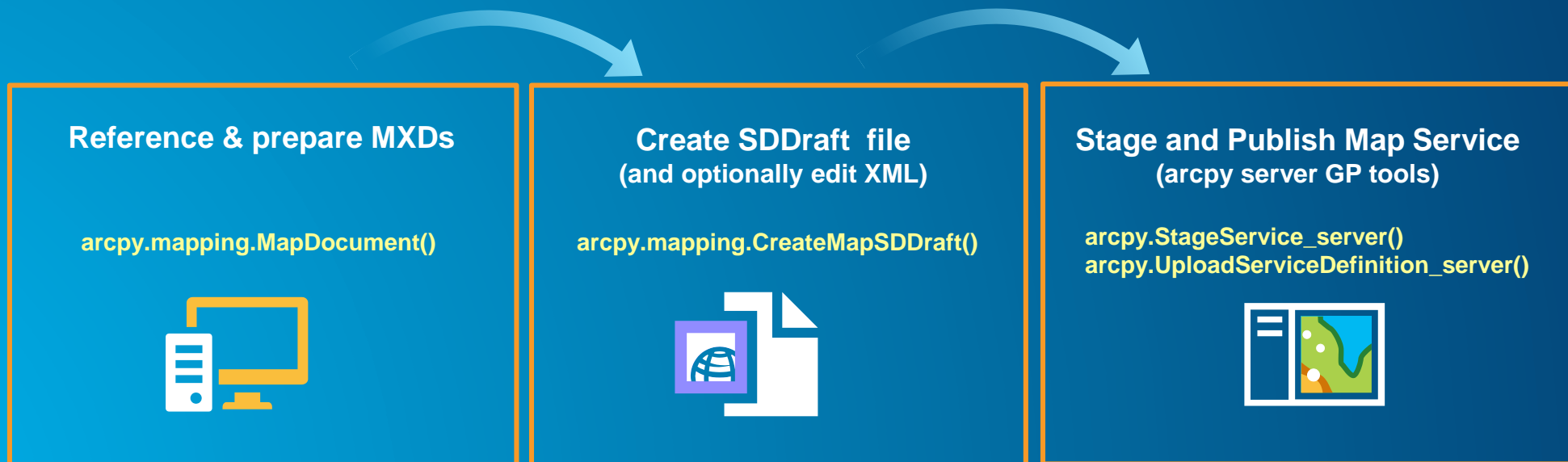


- **ConvertWebMaptoMapDocument()** – Load Web Map for advanced printing workflows



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.



Sample Script: Publishing Service

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

```
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']
```

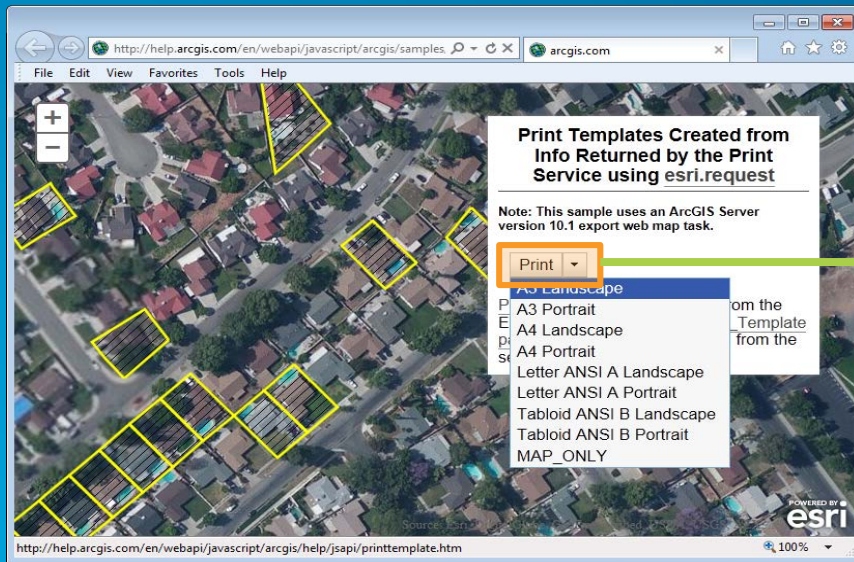
Online help and samples: <http://esriurl.com/4598>

Publishing other service types with python

- **Create geoprocessing services**
 - `arcpy.CreateGPSDDraft()`
- **Create image services**
 - `arcpy.CreateImageSDDraft()`
- **Create geocoding services**
 - `arcpy.CreateGeocodeSDDraft()`

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>



```
app.printer = new esri.dijit.Print({  
  "map": app.map,  
  "templates": templates,  
  url: app.printUrl,  
}, dojo.byId("print_button"));  
app.printer.startup();
```

```
app.printUrl = "http://gilbert:6080/arcgis/rest/services/Utilities/PrintingTools/GPServer/Export%20Web%20Map%20Task";
```

Advanced server printing with arcpy.mapping

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



- `ConvertWebMapToMapDocument (webmap_json, {template_mxd}, {notes_gdb}, {extra_conversion_options})`
- `<Layer>.UpdateLayerFromJSON(json_layer_definition)`

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>

Demo: Web app to export vector PDF using arcpy.mapping

Python code used in custom GP service

Get web map JSON

```
import arcpy, os, uuid

# Input WebMap json
Web_Map_as_JSON = arcpy.GetParameterAsText(0)
```

Get template MXD

```
# Input Layout template
Layout_Template = arcpy.GetParameterAsText(1)
```

Create new MXD based on web map

```
# The template location in the server registered folder
templatePath = '//gilbert/Austin/Templates'
templateMxd = os.path.join(templatePath, Layout_Template + '.mxd')
```

Remove service layers

```
# Convert the WebMap to a map document
result = arcpy.mapping.ConvertWebMapToMapDocument(Web_Map_as_JSON, templateMxd)
mxd = result.mapDocument
df = arcpy.mapping.ListDataFrames(mxd, 'Webmap')[0]
```

Export PDF

```
# Remove the service layer
# This will just leave the vector layers from the template
for lyr in arcpy.mapping.ListLayers(mxd, data_frame=df):
    if lyr.isServiceLayer:
        arcpy.mapping.RemoveLayer(df, lyr)
```

Output file of job

```
# Export the web map to PDF
output = 'WebMap_{}.pdf'.format(str(uuid.uuid1()))
Output_File = os.path.join(arcpy.env.scratchFolder, output)
arcpy.mapping.ExportToPDF(mxd, Output_File, georef_info=True)
```

```
# Set the output parameter to be the output file of the server job
arcpy.SetParameterAsText(3, Output_File)
```

```
var printUrl = "http://gilbert:6080/arcgis/rest/services/Austin
printTask = new esri.tasks.PrintTask(printUrl, {async: true});
params = new esri.tasks.PrintParameters();
params.map = app.map;
ptemplate.format = "PDF"
params.template = ptemplate;
printTask.execute(params, printComplete);
```

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.

```
# 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
df = arcpy.mapping.ListDataFrames(mxd, 'Webmap')[0]

# Reference the staged vector data that corresponds to the dynamic layer in the JSON
# 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]
```

Get JSON Layer
Defn. from web map

```
# Read the JSON and extract the layer definition
# In this case we have hardcoded it to second operational layer
data = json.loads(Web_Map_as_JSON)
layerDefinition = data["operationalLayers"][1]["layerDefinition"]
```

Update vector layer
from JSON

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

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

Resources

- **Examples:**
 - Top N highlighting: <https://github.com/tedrick/dcdev2015/tree/master/topN>
 - Publishing Services: <https://github.com/tedrick/dcdev2015/tree/master/atlasPublish>
- **Print Service Tutorials:** <http://server.arcgis.com/en/server/latest/create-web-apps/windows/printing-in-web-applications.htm>
- **ArcGIS Online arpy.mapping group:** <http://esriurl.com/5915>



Understanding our world.