

Esri Developer Summit

March 26–29, 2012 | Palm Springs, California

esri.com/events/devsummit



Python Scripting for Map Automation

Jeffrey Barrette

Michael Grossman

A decorative graphic at the bottom of the slide. It features a curved orange border at the top. Below it is a semi-circular area containing a satellite map of a landscape. Overlaid on the map is semi-transparent Python code. The code includes symbols like 'esri.symbol.SimpleLineSymbol', 'new dojo.Color([0,0,0,0.5])', and 'feature.setSymbol'.

```
esri.symbol.SimpleLineSymbol
new dojo.Color([0,0,0,0.5])
polySymbol = SimpleLineSymbol(
feature.setSymbol(polySymbol)
}
else if(f == 1) {
var polySymbolGreen = new esri.symbol.SimpleLineSymbol(
polySymbolGreen.setOutlineColor([0,0,0,0.5]), 1);
polySymbolGreen.setColor(new dojo.Color([0,0,0,0.5]), 1);
feature.setSymbol(polySymbolGreen)
}
else if(f == 2) {
var polyBlue = new esri.symbol.SimpleLineSymbol(
polyBlue.setOutlineColor([0,0,0,0.5]), 1);
polyBlue.setColor(new dojo.Color([0,0,0,0.5]), 1);
feature.setSymbol(polyBlue);
}
```

What is map scripting (arcpy.mapping)?

- **A mapping module that is part of the ArcPy site-package**
- **A python scripting API that allows you to:**
 - **Manage map documents, layer files, and the data within them**
 - **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 PDF map books**
 - **Extend Data Driven Pages**

Why was it built? Who is arcpy.mapping for?

- **An environment to use for basic map/layer management and map automation tasks**
- **An easy to use, productive scripting environment for the GIS Analyst**
 - **Courser grained object model**
 - **Not a complete replacement for ArcObjects**
- **A simple way to publish mapping tasks to the server environment**
 - **arcpy.mapping scripts can be easily published as geoprocessing tools**

Basic rules

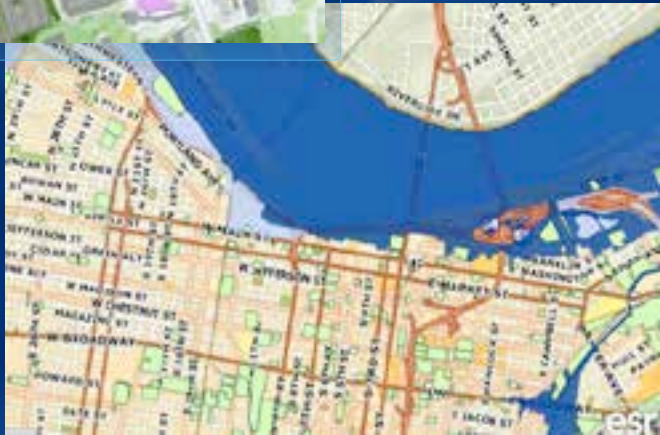
- Reference an MXD using a path or “**current**” keyword
 - When using CURRENT
 - Always run in foreground
 - May need to refresh (e.g., **RefreshActiveView\TOC**)
- Uniquely name all the objects you want to reference
- Pre-author MXDs with all possible elements
 - Can't create new objects (e.g., north 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





Existing Applications

Demonstration



DEMO1

DEMO2

What's new at 10.1

<http://esriurl.com/3880>

- What's new à Mapping à What's new for automating map workflows



The screenshot shows the ArcGIS 10.1 Web browser interface. The left sidebar contains a navigation menu with categories like 'What's new', 'Mapping', 'Data management', 'Geoprocessing', 'Services', 'Mobile', 'Editing', 'Geoprocessing', 'Boudata', 'Services', 'Developing', 'Extensions', 'Trade Tools', 'Mobile', 'ArcGIS Tutorials', 'Copyright information', 'License agreement', and 'ArcGIS ArcSDE products'. The main content area displays the article 'What's new for automating map workflows in ArcGIS 10.1'. The article title is 'Map automation with Python and arcpy.mapping'. The text describes significant additions to arcpy.mapping for the 10.1 release, including automated symbology properties, report generation, time-based analysis, legend updates, map service publishing, and printing support. A detailed list of new top-level arcpy.mapping functions is provided at the bottom.

What's new for automating map workflows in ArcGIS 10.1

Map automation with Python and arcpy.mapping

There were significant additions to arcpy.mapping for the 10.1 release, including:

- Symbology properties for the following renderers can now be automated: graduated colors, graduated symbols, unique values, and classified rasters.
- An **Export Report** function is available that allows you to automate the generation of reports.
- ArcGIS 10.1 provides access to a **layer's time** properties to perform analysis over time. You can also animate time on layers.
- The arcpy.mapping module now allows you to update the individual legend items in a **Legend Element** on a page layout by using the **UpdateItem** method. You can also remove legend items using the **RemoveItem** method.
- Two new functions exist to automate the map service publishing in a Python script. The **CreateMapService** function has been introduced to create draft service definition files. Secondly, the **AnalyzeService** function has been added to analyze service definition drafts for errors that might prevent publishing.
- ArcGIS provides support for printing WebMaps from the ArcGIS web APIs. The **ConvertWebMapToMapDocument** function will convert a WebMap that you intend to print or export to a map document. Once the document is converted, the full state of the WebMap exists in the map document.
- Text elements and graphic elements on a page layout can now be cloned.

A detailed list of all additions made at 10.1 is below.

New top-level arcpy.mapping functions:

- **AddTableView**—Provides the ability to add a table to a data frame within a map document (.mxd).
- **AnalyzeService**—Analyzes Service Definition Draft (.sdraft) files to determine suitability and sources of potential performance issues before converting a service definition draft file to a service definition (.sdf) file.
- **ConvertWebMapToMapDocument**—Converts a WebMap (in JSON format) that you intend to print or export to a map document. The map document can be further modified before finally being printed or exported.
- **CreateGISServerConnectionFile**—This function creates a connection file that can be used to connect to a GIS Server.
- **CreateMapSDraft**—Converts Map Document (.mxd) files to Service Definition Draft (.sdraft) files.

What's new in 10.1: cloning elements

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

```
vert1 = arcpy.mapping.ListLayoutElements(  
    mxd, "GRAPHIC_ELEMENT", "VerticalLine")[0]  
vert1.elementPositionX = xPos; vert1.elementPositionY = 4  
vert1.elementHeight = 3  
for line in range(1, numColumns+1):  
    vert_clone = vertLine.clone("_clone")  
    xPos = xPos + colWidth  
    vert_clone.elementPositionX = xPos
```

DEMO

What's new in 10.1: symbology classes

- **Layer.symbologyType** r/o : string
 - Returns:
 - GRADUATED_COLORS, GRADUATED_SYMBOLS, UNIQUE_VALUES
 - RASTER_CLASSIFIED, OTHER
- **Layer.symbology** r/o : Symbology Class

- **Example:**

```
if lyr.symbologyType == "GRADUATED_COLORS":  
    lyr.symbology.numClasses = 10  
    lyr.symbology.valueField = "POP2007"
```

- **General notes, can change:**

- Symbology class
 - Individual symbols
 - Classification methods
- } use **arcpy.mapping.UpdateLayer**

DEMO1
DEMO2

What's new in 10.1: legend items

- A referenced legend item in a .style file can then be used to update already existing legend items in a layout.
- Example:

```
mxd = arcpy.mapping.MapDocument("current")
legend = arcpy.mapping.ListLayoutElements(mxd,
    "LEGEND_ELEMENT")[0]
styleItem = arcpy.mapping.ListStyleItems(
    "USER_STYLE", "Legend Items", "MyNewStyle")[0]
for lyr in legend.listLegendItemLayers():
    legend.updateItem(lyr, styleItem)
```

DEMO

Publishing Map Services with arcpy.mapping

- Workflow from map document to map service
- Use Python for:
 - Publishing automated analysis results
 - Scheduled service upgrades
 - Batch migration from 10.0 to 10.1



Publishing Map Services with arcpy.mapping

Sample: CreateMapSDDraft (arcpy.mapping)

<http://esriurl.com/3934>

Open and
modify MXD

Create and
analyze
SDDraft,
optionally
modify XML

Stage and
publish Map
Service

```
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']
```

High Quality Server Printing with arcpy.mapping

- Build customized versions of the new high quality print services built into 10.1 server
- New arcpy.mapping method for converting Web Maps to MapDocuments: **ConvertWebMapToMapDocument ()**
- Online help and examples
 - <http://esriurl.com/3941>
- `ConvertWebMapToMapDocument (webmap_json, {template_mxd}, {notes_gdb}, {extra_conversion_options})`



High Quality Server Printing with arcpy.mapping

- **Convert the web map to a map document**
- **Full capabilities of arcpy.mapping on the document**
 - **Modify content**
 - **Add content**
 - **Export using custom options**
 - **Export data driven pages**
 - **Export to PDF and insert additional pages**
- **Return the PDF file or map book**

Related Session:

**Supporting High-Quality Printing in Web Applications
with ArcGIS 10.1 for Server**

High Quality Server Printing with arcpy.mapping

Sample: ConvertWebMapToMapDocument

<http://esriurl.com/3941>

Get webmap
JSON

Template MXD
name

Create new MXD
based on
webmap JSON

Export PDF and
append Title and
Contact Info
pages

```
import arcpy, os

# For this sample, we will read the JSON stored in a text file
webMapFile = open(r"C:\Project\WebMapWithMapNotes.txt")
json = webMapFile.read()

# Specify a template mxd
templateMxd = r'C:\Project\Landscapellx17.mxd'

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

# Export the WebMap to PDF
mapPdf = r'C:\Project\Output\map.pdf'
arcpy.mapping.ExportToPDF(mxd, mapPdf)

# Set the master pdf output file name and remove if it already exists
pdfPath = r"C:\Project\Output\EvacuationRoute.pdf"
if os.path.exists(pdfPath):
    os.remove(pdfPath)

# Create the file and append pages
pdfDoc = arcpy.mapping.PDFDocumentCreate(pdfPath)
pdfDoc.appendPages(r"C:\Project\Title.pdf")
pdfDoc.appendPages(mapPdf)
pdfDoc.appendPages(r"C:\Project>ContactInfo.pdf")

# Commit changes to PDF and delete variable reference
pdfDoc.saveAndClose()

del pdfDoc
```

Deployment - arcpy.mapping and Python Add-Ins

- **ArcGIS 10.1 – Python Add-Ins**
 - Add-in deployment of mapping tools
 - Mechanism for deploying ArcMap customizations
 - Online help:
<http://esriurl.com/3943>
- **Take mouse input, respond to app events**
 - FinishDrawing, DPP
PageChange, etc.

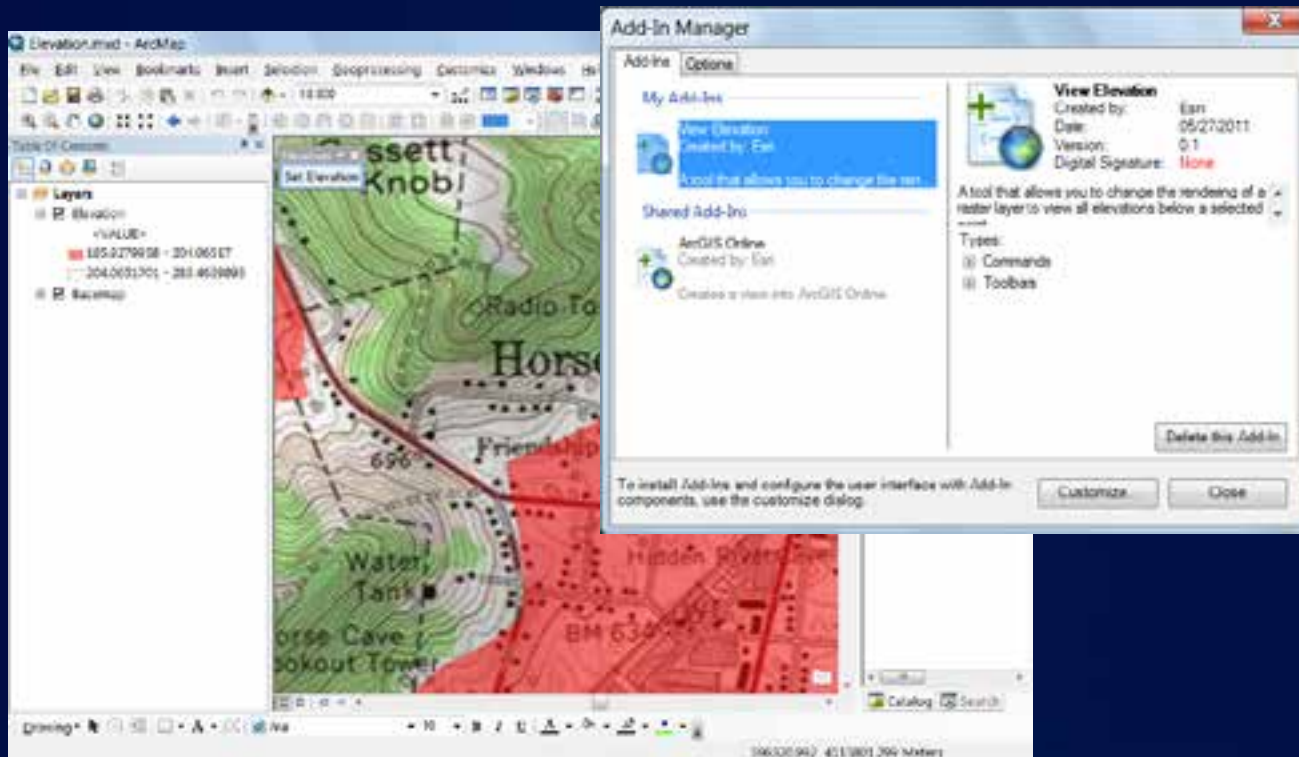


Python Add-in Wizard
Download from:
<http://esriurl.com/3942>

**Related Session:
Developing ArcGIS for Desktop Add-ins with Python**

Demonstration:

Modify class breaks of a layer – Elevation python add-in





esri