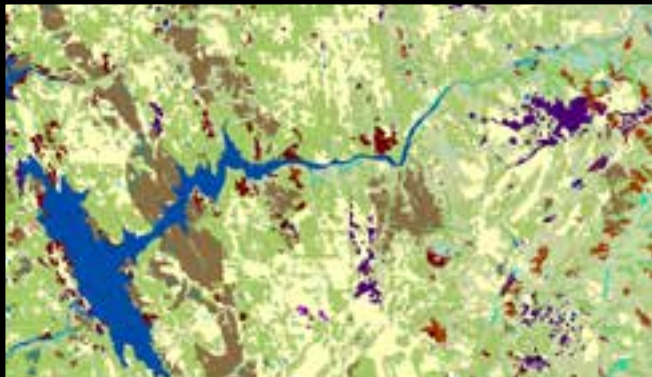


Using Arcpy.mapping and ArcGIS Server Admin API to Automate Services

2015 Esri User Conference

July 21st, 2015



Steve Goldman, GISP
GIS Manager / GIO
California Department of Fish and Wildlife
<http://www.wildlife.ca.gov>

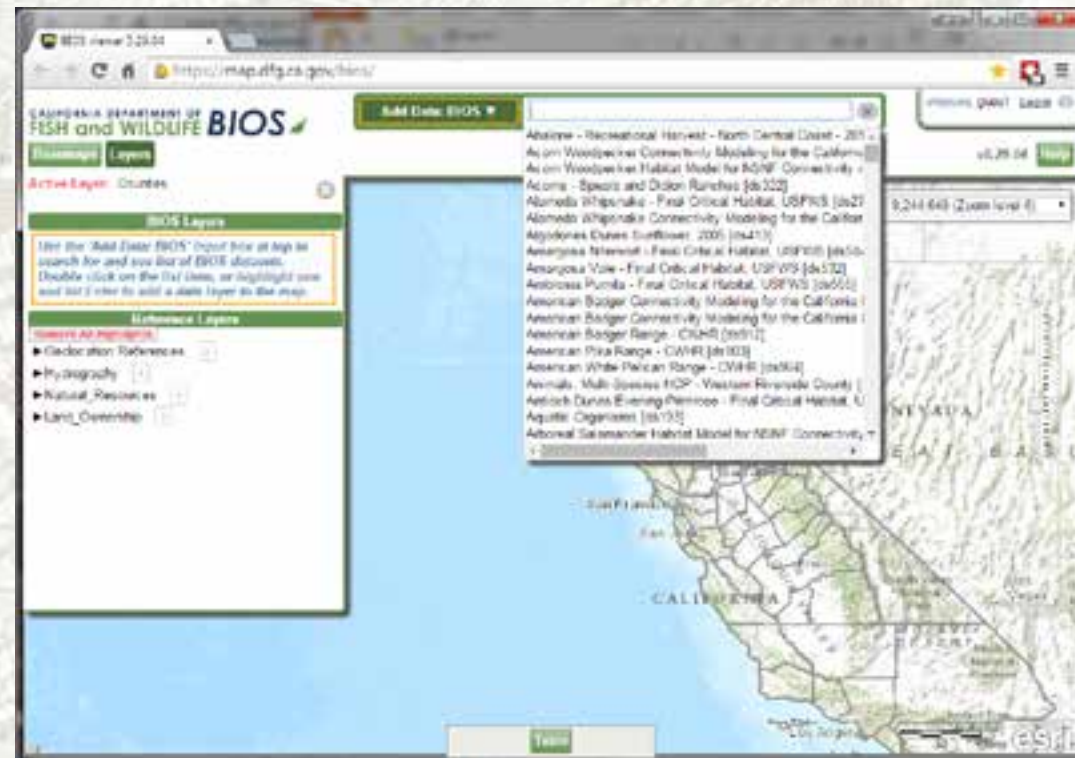


Problem

Complexity!

BIOS Web Mapping Application:

- 1,000+ natural resources datasets, weekly updates.
- 60 map services.
- Public/Secure content



Solution

Automate!

Scripting:

- Reduced staff time from several hours to minutes per dataset.
- Allowed us to perform operations that would've been restrictively time-consuming by hand.

```

C:\Windows\OS2\cmd.exe [C:\Windows\OS2\cmd.exe] [C:\Windows\OS2\cmd.exe]
File Edit Format Run Options Windows Help

def execute_LYNToSDE(enviro, egoDomain):
    global GC_log
    global GC_updateSDD
    global GC_caddlogovp
    global GC_caddlogovp
    setStat = True
    GC_log.write("C:\OS2\Function\execute_LYNToSDE.bat")
    try:
        #Read config for details and store in global constants.
        wd = memory_config(enviro)
        if wd == False:
            raise Exception, "Error in memory_config"
        #Get registry info. data GC_regInfo. Hold security groups, do session avail
        regData = get_registry_data(enviro)
        if regData == False:
            raise Exception, "Registry function error."
        #Get list of refreshed layers.
        refreshed = get_list_layers_refresh(enviro)
        if refreshed == False:
            raise Exception, "Error in get_list_layers_refresh"
        #Print: user for update passwords if both in both in refresh
        caddUpd = False
        for r in refreshed:
            if r.upper().find("CADD") > -1 or r.upper().find("CADD") > -1:
                caddUpd = True
        if caddUpd == True:
            cmd = True
            while cmd:
                GC_caddlogovp = raw_input("Enter new CADD_GOV password: ")
                GC_caddlogovp = raw_input("Enter new CADD_GOV password: ")
                ans = raw_input("Yes entered " + GC_caddlogovp + " for CADD_GOV ans ")
                if ans == "y":
                    cmd = False
            else:
                raise
        #Write the pw out to a file for SDEtoServices to read.
        spds = store_pw_for_cadd_standalone(enviro)
    
```



How

Scripting

LyrMayor (“Layer Mayor”)

Input = a .lyr file for a dataset

Output = dataset published in the correct map service.



Details

Architecture / Workflow

MS Access front-end "Registry" database (SQL Server)
(Dataset Number assignment, title and security)

Dataset loaded into SDE (DEV/TEST/PROD)

Individual Lyr file created and saved in "LyrFiles" folder

LyrMayor script is then run



Details

Script steps

- For every lyr file in LyrFiles folder (new or updated dataset)
- Spatially index features to statewide grid
 - Index attribute values for search
 - Build MXD for applicable Service (arcpy.mapping)
 - Export dataset HTML Metadata page (arcpy)
 - Publish MXD to SD, then to Service (arcpy.mapping)
 - Modify Service properties (AGS Admin API)



Details

MXD Building

Full MXDs built from scratch each time

Up to 100 datasets per MXD

Datasets are placed into MXDs based on:

- Security: Public vs Secure dataset/service
- Feature Type: Point/line/polygon/raster
- Internal Dataset Number (blocks of 100)



Details

MXD Building (arcpy.mapping)

#Handle layerfile

```
lyrObj = arcpy.mapping.Layer(<path to layer>)  
lyrObj.visible = False  
lyrObj.name = 'hungry causeway bats'
```

#Set datasource for enviro specified.

```
ws = lyrObj.workspacePath  
if enviro == 'dev':  
    lyrObj.findAndReplaceWorkspacePath(ws, <appropriate .sde file>, False)  
if enviro == 'test':  
    lyrObj.findAndReplaceWorkspacePath(ws, <appropriate .sde file>, False)  
if enviro == 'prod':  
    lyrObj.findAndReplaceWorkspacePath(ws, <appropriate .sde file>, False)  
lyrObj.save()
```

#Handle doc and add layer.

```
doc = arcpy.mapping.MapDocument(<path to appropriate mxd>)  
df = arcpy.mapping.ListDataFrames(doc)[0]  
arcpy.mapping.AddLayer(df, lyrObj, "BOTTOM")  
doc.save()
```



Details

Staging an SD file (arcpy and arcpy.mapping)

#Create the sdDraft

```
arcpy.mapping.CreateMapSDDraft(doc, sdDraft, service, serverType, conFile, \
    copyData, agsFolder, summary, flags)
```

#Analyze the sdDraft

```
analysis = arcpy.mapping.AnalyzeForSD(sdDraft)
```

```
if analysis['errors'] == {}:
```

```
    sys.stdout.write("Analysed W/O errors. \n")
```

```
else:
```

```
    sys.stdout.write("Analysed WITH errors. \n")
```

```
    raise SDDraftAnalysis, "SD could not be staged for " + mxd + ". \n"
```

#Stage the service to an SD file

```
arcpy.StageService_server(sdDraft, sd)
```



Details

Publish the SD file (arcpy and AGS Admin)

#Publish the SD to create the service

```
arcpy.UploadServiceDefinition_server(<path to sd>, conFile, service, "#", \  
    'EXISTING', agsFolder)
```

#Obtain current properties from the newly published service

```
url = "http://" + server + ":port/arcgis/admin/services/folder/service-type\  
params = urllib.urlencode({'token': GC_token, 'f': 'json'})  
headers = {"Content-type": "application/x-www-form-urlencoded", "Accept":  
    "text/plain"}  
httpConn = httplib.HTTPConnection(server, port)  
httpConn.request("POST", url, params, headers)
```

#Read http response

```
response = httpConn.getresponse()  
data = response.read()  
dataObj = json.loads(data)
```



Details

Edit service properties (AGS Admin)

#Make a python dictionary of your desired properties.

```
desiredProp[key] = x
```

#Alter the current properties given in response with desired values

```
dataObj['properties'][realkey] = desiredProp[key]
```

```
updatedSvcJson = json.dumps(dataObj)
```

#Call the service 'edit' endpoint

#Send the updated properties object through it to update the service

```
editSvcURL = "http://" + server + ":port/arcgis/admin/services/folder/service-  
type + "/edit"
```

```
params = urllib.urlencode({'token': GC_token, 'f': 'json', 'service':
```

```
updatedSvcJson})
```

```
httpConn.request("POST", editSvcURL, params, headers)
```



Questions?

<http://bios.dfg.ca.gov/>

Google "CDFW BIOS"

Steve Goldman, GISP

GIS Manager / GIO

Biogeographic Data Branch

California Department of Fish and Wildlife (CDFW)

steve.goldman@wildlife.ca.gov

916-445-9939

