CIP Automation

Standardized Python Scripts to Reduce Time and Errors for
The Army National Guard

Nate Bost
South Carolina Army National Guard
Common Installation Picture

• The Common Installation Picture (CIP) is a distinct, minimum set of geospatial data layers required to be maintained for all Army sites.

• Each year, every state must submit its geospatial data to National Guard Bureau.

• In order to be CIP compliant, the data must follow SDSFIE standards, and many fields and domains are required to be filled in.
Why Automate?

- Reduces/Removes human error.
- Removes need for tedious data entry.
- Saves significant time and money.
- Allows for changes in schema/standards.
Issues

• Geometry
  • Changes made, features added, need to be updated.

• Primary Keys
  • Needed for all features.
  • Can easily have duplicates, must be recalculated every time a change is made.

• Installation IDs
  • Must be calculated for every single feature in every single feature class.

• Domains
  • Many not necessary or unknown.
  • Require clicking, scrolling and choosing or looking up key and calculating fields.
Geometry

- Points
  - X, Y
- Lines
  - Length
  - From X, From Y
  - To X, To Y
  - Midpoint X, Y
- Polygons
  - Area
  - Perimeter
  - Centroid X, Y
...u_d

- Unit of measure domains
- perim_u_d, area_u_d, etc.
- Should be calculated same time as geometries.
  - Ensures accurate labels for measurements (ft, m, yds)
- Different for each feature class, requires override for some features in the script.
Geometry Script

- Assign Units for each field

```python
import arcpy

# Set Field Categories
UDict = {}
UDict['Feet'] = ['Linear_feet', 'cbl_len', 'linear_ft', 'fence_len', 'gate_len', 'pipe_lgth', 'trail_len', 'water
UDict['Acre'] = ['acres']
UDict['SqFt'] = ['sde_SDE_Rooms_Area', 'Square_Feet', 'area_size']
UDict['Miles'] = ['Miles']
UDict['Yards'] = ['3YD', 'Square_Yards', '3_Yards', 'square_yds']

# Set Domain Values
DomainDict = {}
DomainDict['length_u_d'] = 'TBD'
DomainDict['area_u_d'] = 'SFT'
DomainDict['perim_u_d'] = 'FT'
DomainDict['lgsarea_u_d'] = 'AC'
DomainDict['volume_u_d'] = 'GAL_US'
DomainDict['vol_u_d'] = 'CR'

ConVertDict = {'Feet': 3.28084, 'Acre': .000247105, 'SqFt': 10.764, 'Miles': .000621371, 'Yards': 1.19599}
```

- Get conversion from SDE default (Meters)

- Define ..u_d Domain values

- Set overrides for specific feature classes

```python
ODict = {'road_area': {'area_u_d': '5YD', 'perim_u_d': 'YD', 'area_size': 1.19599, 'perim': 1.09361}, 'airfi
```
Geometry Script cont..

- A list is created for every feature class requiring geometry.
  - This can include CIP and non-CIP Layers

```python
def FeatureNames():
    return ['compressed air tank point', 'stmswr stilling basin area', 'road area', 'hazmat stor']
```

- Using arcpy's Update Cursor, loop through each feature of each feature class and update fields where applicable.

```python
def edit_startEditing(True, True):
    edit.startOperation() 
    # Loop Through DOM and Update 
    print 'Updating'
    # with arcpy.da.UpdateCursor(FC, DOM) as Cursor:
        for row in Cursor:
            x = 2
            while x < len(DOM):
                try:
                    row[x] = row[ShapeDict[DOMDict(DOM[x])]] * GDict[FC.split('\')[-1].split('.')[][-1][DOM[x]]]
                except:
                    row[x] = row[ShapeDict[DOMDict(DOM[x])]] * ConvertDict[DOMDict(DOM[x])]
                x = x + 1
                Cursor.updateRow(row)
        # End Edit Session
        edit.stopOperation() 
        edit.startOperation() 
```

- Result: Every feature class has updated geometries and domains.
Primary Key

- Installation ID + feature name + Object ID
- Ex:
  45C70building142

- If feature is created, copied, split, merged or loaded, the Object ID changes, this creates nulls and duplicates.
  - Example:

```
<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>cblgrp_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>31249</td>
<td>45B60cblgrp31249</td>
</tr>
<tr>
<td>31650</td>
<td>45C70cblgrp31650</td>
</tr>
<tr>
<td>32859</td>
<td>&lt;Null&gt;</td>
</tr>
<tr>
<td>33260</td>
<td>45C00cblgrp742</td>
</tr>
</tbody>
</table>
```
Primary Key Script

- Features Calculated using the Primary_Key_Lookup table from the GKO products CIP_Rules database.
- One issue is duplicate attribute names for point/area features.
  - Script accounts for this by adding an “x” to the calculated output
  - Ex: 45C70envastx201
- Loops through table and using update cursor, creates unique IDs on the fly.
- Result: Every CIP feature has an updated Primary Key.
Installation and Address ID

- Must calculate for all features in each feature class, in each installation.
- 63 Feature classes (CIP layers), 50 installations = 3150 calculate fields.
- This causes even scripts to take hours to update every feature with the information. A way around this is to only calculate null and empty instln_id fields.
- Some features exist outside of the property boundaries and must be updated with the state abbreviation.
- address_id can be found in many feature classes, and is also a field in installation_area, so this can be automatically updated along with the instln_id where applicable.
Installation and Address ID

- Requires performing a select by location on each feature class for each location.
- Script loops through installation_area creates a list with the instln_id along with the address_id.
- The script then loops through each feature class and finds features with null or empty instln_id and address_id fields.
- Those feature’s centroids are then appended to an existing empty point shapefile containing fields for the features Object ID and the feature class path, as well as empty fields for the corresponding installation id and address id to be updated later.
Installation and Address ID

- For each installation in the created list, a temporary layer is created with just that site’s installation_area boundary.
- This is done using a dynamic SQL statement:

```python
for InstNum in Index:
    # Set Layers
    Area = 'Area'
    MLYr = 'MLYr'
    # Make Layers for Selection
    arcpy.MakeFeatureLayer_management(Inst_area, Area, 'instln_id = ' + '"" + InstNum + '""
    arcpy.MakeFeatureLayer_management(MissingShp, MLYr)
    # Select All Temp points within the installation
    arcpy.SelectLayerByLocation_management(MLYr, 'INTERSECT', Area)
    # Assign instln_id and address_id to each feature
    arcpy.CalculateField_management(MLYr, 'instln_id', '"" + InstNum + '"', 'PYTHON_9.3')
```

- A Select by Location is then performed with the installation area and the temporary shapefile with the null IDs.
- The instln_id and address_id are then calculated on the selected points.
- Any features not updated when done are given an installation id of the states abbreviation.
Installation and Address ID

- A filled out table can be seen below:

![](InstEmpty_DontDelete)

- The script then uses the finished table to calculate the correct installation and address IDs based on the recorded object id using an update cursor.

- Therefore, if only 3 feature classes had a null instlIn_id, the script would only run on those 3. Save a large amount of geoprocessing time.

- Result: Every Feature of every feature class has updated installation and address IDs
Domains

- The CIP data call requires many domain fields to be updated with information.
- Many times this information is unknown or not applicable to the feature, so they are often updated with ‘Unknown’ or ‘TBD’.
- This can become tedious when digitizing because fields must be updated by clicking on a drop down list or field calculated with a code that must be looked up.
- With this script, any field can be updated, not just one with domains including feat_name, feat_desc, etc.
Domains

- This script utilizes a simple .csv file, allowing for a simple code only 53 lines long.
- Uses Update Cursor to calculate fields.
- Only run on null fields to avoid overwriting work.
- Simply reads through the CSV file and uses the information to calculate fields with user defined values.
- Result: Every null domain/text field in spreadsheet is updated.
Summary

- **Automation:**
  - Reduces/Removes human error.
  - Removes need for tedious data entry.
  - Saves significant time and money.
  - Allows for changes in schema/standards.

- **Fields affected:**
  - All Geometry and UOM (length, area_size, x/y, area_u_d, etc.)
  - Primary Keys (envast_id, building_id, etc.)
  - Instln_id
  - Address_id
  - Any domain/text field desired (SCARNG: 1428/178)

- Does not fix 100% of the CIP requirements

- Each script takes a varying amount of time. (PK-5, Dmn-20, Inst-15)
Lessons learned

• Data access Module (arcpy.da.UpdateCursor and arcpy.da.Editor).

• Going through every feature in every feature class takes a long time, even for the computer. (Installation script).

• When working with personal gdb (.mdb) turn off background geoprocessing.

• Double check calculations (SqYds vs Yds).

• Run after Rec and Post!

• Adjoining Properties can cause issues with the Installation ID.
Questions?

Nate Bost
803-299-2434
bostn@tag.scmd.state.sc.us