Using Python to Gather Information about Data in SDE

Replication Requirements

David Watkins
Pacific Southwest Region GIS Coordinator
USDA Forest Service

Background

- The USDA Forest Service manages data in SDE and utilizes replication as a workflow to edit data

- This Presentation illustrates using Python scripts to determine whether datasets meet the requirements for participation in replication

Replication Requirements

- In Order to participate in replication a dataset must meet the following:
  - It must be Registered As Versioned
  - It must be stored with High Precision
  - Each Feature Class must have a GlobalID field
  - It must have ‘Public’ view access granted

Forest Service Scenario

- Data is stored in Oracle database
- ArcSDE 10.1 is installed
- All data has been converted to High Precision
- Each forest has a GIS Coordinator that uses a Proxy account to connect to the data
  - This account allows them to manage the data with elevated privileges

The Issue

- It can be time consuming to find out if all the data meets the requirements for replication if you have many feature datasets.
- The status changes frequently

The Solution

- Create a Python script to gather the necessary information for you
- All you need is a valid Proxy connection file to the database

The Script

- Two Parameters
  - SDE Workspace
  - Output CSV file
- Uses arcpy.da.walk module to walk through the datasets in the geodatabase
- Uses Describe function to get info on Precision and whether the dataset is registered as Versioned
- Checks field names for presence of ‘GlobalID’
- Uses SQL statement to determine ‘Public’ access
- Uses csvwriter to write the output file

Script Setup

```
# load required modules
import arceditor,sys,os,csv,arcpy
from arcpy import env

# Get SDE workspace
sdeConn = arcpy.GetParameter(0)
sdeConnText = arcpy.GetParameterAsText(0)

# Output file
outputFile = arcpy.GetParameterAsText(1)

# Get Workspace Information
desc = arcpy.Describe(sdeConn)
owner = desc.connectionProperties.user.replace("[","").replace("]","")
instance = desc.connectionProperties.instance.split(":")[3].split("$")[0]

# Set up SQL connection for fetching privilges from Oracle
sqlConn = arcpy.ArcSDESQLExecute(sdeConnText)
```

- Import required modules
- Get the parameters and set variables
- Owner is [S_R05] – need to drop []s
- Setup the SQL connection

Start the loop

```
with open(outputFile, 'wb') as csvfile:
    csvwriter = csv.writer(csvfile)
    arcpy.AddMessage('Dataset,Name,Data Type,versioned,Precision,Global,Public')
    csvwriter.writerow(['Dataset','Name','Data Type','versioned','Precision','Global','Public'])

    for dirpath, dirnames, filenames in arcpy.da.Walk(sdeConn,topdown=True,followlinks=True,datatype="Any",type="ALL"):
        dataset = os.path.split(dirpath)[1]
        if dataset.endswith(".sde"):
            dataset = "StandAlone"
        if (dataset.startswith(owner) or dataset == "StandAlone"):
            for filename in filenames:
                if filename.startswith(owner):
                    if arcpy.Exists(os.path.join(dirpath, filename)):
```

- Setup the CSV file for writing
- Set the da.walk parameters
- Check to see if data is in a feature dataset
- Check to see if data exists

Describe Object

```
# See if object is registered as versioned
odesc = arcpy.Describe(os.path.join(dirpath, filename))
datatype = odesc.datasetType
isSpatial = datatype not in ['Table','Topology','Text','RelationshipClass','RepresentationClass','GeometricNetwork','NetworkDataset']

if odesc.IsVersioned:
    versioned = 'True'
else:
    versioned = 'False'
#add code for Precision, Global, and Public

if isSpatial:
    sr = odesc.spatialReference
    if sr.ishighPrecision:
        isHigh = 'True'
    else:
        isHigh = 'False'
else:
    isHigh = 'NA'
```

- Create a Describe object
- Check the ‘IsVersioned’ property
- Get the Spatial Reference and check the ‘ishighPrecision’ property

GlobalID field

```
if datatype in ['FeatureClass','Table']:
    fieldlist = arcpy.ListFields(os.path.join(dirpath,filename))
    fieldnames = []
    for f in fieldlist:
        fieldnames.append(f.name)
    if 'GLOBALID' in fieldnames or 'GUID' in fieldnames:
        hasGUID = 'True'
    else:
        hasGUID = 'False'
else:
    hasGUID = 'NA'
```

- Get the field list using ListFields
- Check for presence of GlobalID field

Public Access

```
# Pass the SQL statement to the database and fetch privileges.
SQLStatement = "select grantee, privilege from all_tab_privs where
        TABLE_SCHEMA = UPPER('"+filename.split('.')[0]+"') and table_name = UPPER('"+filename.split('.')[1]+"')"
public = 'False'
try:
    sdeReturn = sqlConn.execute(SQLStatement)
    if not type(sdeReturn) == bool:
        for priv in sdeReturn:
            if 'PUBLIC' in priv:
                public = 'True'
                break
```

- Create SQL statement
  - Check all_tab_privs table
  - Check for ‘Public’ access granted

Wrapping Up

```
                    except Exception, ErrorDesc:
                            arcpy.AddWarning("No SYSDBA connection file found for this database. Privileges will not be listed")
                            arcpy.AddMessage(ErrorDesc)
                            arcpy.AddMessage("\n+++++++++")
                            sdeReturn = False
                            continue
                    arcpy.AddMessage(str(dataset)+','+str(filename)+','+datatype+','+versioned+','+isHigh+','+hasGUID+','+public)
                    csvWriter.writerow([str(dataset),str(filename),datatype,versioned,isHigh,hasGUID,public])

                else:
                    print(owner + " does not own " + filename)
            else:
                print(owner + " owns no objects in " + dataset)
arcpy.AddMessage("+++")
arcpy.AddMessage("+++ END: Listing of " + sdeConnText + " for owner " + owner + "\n")

arcpy.ClearWorkspaceCache_management()
```

- Write the CSV file
- Clear workspace cache

Questions?