



Creating Geoprocessing Services with Python Script Tools

Andrew Ortego

Creating Geoprocessing Services with Python Script Tools

1. Requirements
2. Preparing to Create a GP Service
3. Publishing Tips and Tricks
4. What's Next?

Creating Geoprocessing Services with Python Script Tools

1. Requirements

2. Preparing to Create a GP Service

3. Publishing Tips and Tricks

4. What's Next?

Requirements

- **ArcGIS Enterprise (includes Portal) or ArcGIS Server**
 - Must have **Admin** or **Publisher** permissions to publish a GP Service
 - 10.4+ Admins, set the **allowGPAndExtensionPublishingToPublishers** property equal to **True** for Publishers
- **ArcGIS Pro or ArcGIS for Desktop (i.e. ArcMap or ArcCatalog)**
 - Work flow has been available since 9.x, but stay up to date as often as possible
- **Optional: Federated Server**
 - Have the client-side do the heavy lifting prior to publishing
 - Keep data and services on the **Data Store** to let the tool run as fast as possible

Creating Geoprocessing Services with Python Script Tools

1. Requirements
2. Preparing to Create a GP Service
3. Publishing Tips and Tricks
4. What's Next?

Preparation -- Work Flows

- **Use Model Builder or Python to create a GP tool**
 - You can also publish any Esri tool since your custom tool can include those
- **Service won't publish if the tool won't run**
 - Check the **Results** window in **Pro** or **Desktop** for any errors
- **Opt for Asynchronous over Synchronous for medium to large tools**
 - This option is available in the **Publishing Wizard**
 - "Async" tools need to be queried, while "Sync" will run to completion

Preparation -- Data

- **Avoid "hard coding" all data paths and dependencies**
 - Favor input parameters which a user can customize
 - `my_param = arcpy.GetParameterAsText()` or "P" in Model Builder
- **If you have to use a "hard coded" data path, build it dynamically**
 - Utilize `os.path.join("root directory", "file_name")` so the tool works on any OS
 - Use `sys.path.append("server's path to the module")` if needed
- **Write intermediate/temporary output data to memory**
 - Keep everything running fast, and lighten the load on your Data Store
 - Use `os.path.join("in_memory", "your output name")` or Intermediate Data

Creating Geoprocessing Services with Python Script Tools

1. Requirements
2. Preparing to Create a GP Service
- 3. Publishing Tips and Tricks**
4. What's Next?

Publishing Tips and Tricks

- **Project Data is "consolidated" aka copied to the Data Store**
 - Data is found in any directory in the script/model and **Table of Contents**
 - Data referenced from the server (e.g. **Data Store**) is not consolidated
 - Python recursively searches directories for data to consolidate
- **Consider using a Geoprocessing Package to store Project Data**
 - Great for offline debugging and development
- **Tool Validation script will be published, and executed by SubmitJob()**
 - The validation occurs server-side, and **should** act the same, but test it!

Creating Geoprocessing Services with Python Script Tools

1. Requirements
2. Preparing to Create a GP Service
3. Publishing Tips and Tricks
4. What's Next?

What's Next?

- **GP Services can be added to Portal as a Web Tool**
 - You can access these in the Portal Analysis pane, or any Web App.
 - Doesn't work for ArcGIS Online, only ArcGIS Enterprise for Portal
- **Upon completion, integrate the service into Python or Model Builder**
 - Call the service using `arcpy.ImportToolbox("<URL to service>", "optional alias")`
 - REST API for automation and testing-- use `"requests"` and `"json"` modules
- ***What is a geoprocessing service?* documentation**
 - Great documentation for learning more about publishing GP Services



Thanks for joining us!

Andrew Ortego



esri

THE
SCIENCE
OF
WHERE