# How to create a geoprocessing service

Ashley DePottey

May 14, 2015

## Create the python script

(this is Outliers.py used in the Outliers GP service)

import arcpy

import os

from arcpy import env

from arcpy.sa import \*

try:

# Get parameters

features = arcpy.GetParameter(0)

# Do outlier

out\_features = arcpy.CreateUniqueName('results', arcpy.env.scratchWorkspace)

arcpy.AddMessage('Output directory: ' + out\_features)

arcpy.ClustersOutliers\_stats(Input\_Feature\_Class=features,

Input\_Field='VALUE',

Output\_Feature\_Class=out\_features)

arcpy.AddMessage('Outlier analysis complete')

# Write result

arcpy.SetParameter(1, out\_features)

except Exception as e:

arcpy.AddError(str(e))

finally:

arcpy.Delete\_management('in\_memory')

Some things to note:

* features = arcpy.GetParameter(0)

These are the input parameters that will be provided by the user when running this tool. They will be set up in the toolbox in the next step. All parameters (including out parameters) are accessible by index.

* arcpy.AddMessage('Output directory: ' + out\_features)
* arcpy.AddError(str(e))

Add messages and errors this way so they will show in the geoprocessing execution box in ArcMap and can be retrieved from the job when running as a geoprocessing service in ArcGIS Server.

* out\_features = arcpy.CreateUniqueName('results', arcpy.env.scratchWorkspace)

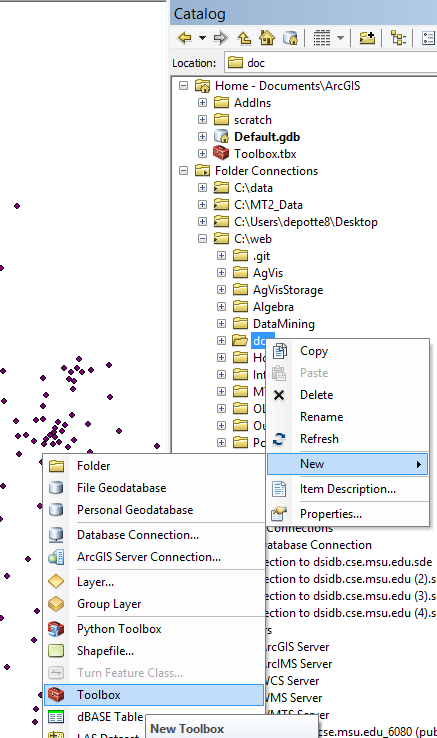
For more information on the scratch workspace, scratch folder, and scratch GDB, see <http://resources.arcgis.com/en/help/main/10.2/index.html#/Scratch_Workspace/001w00000003000000/>.

* arcpy.SetParameter(1, out\_features)

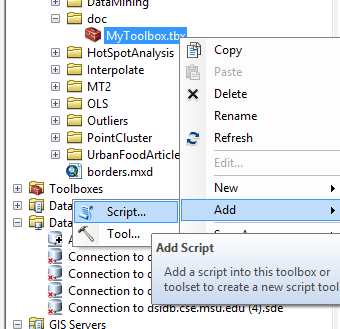
These output parameters need to be set up in the toolbox in the next step.

## Create a toolbox in ArcPy

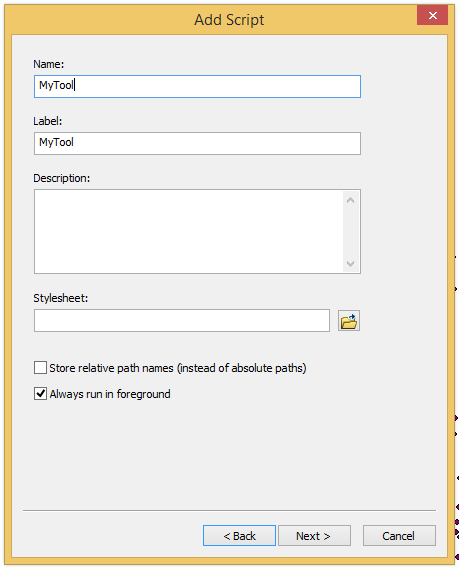
1. In your Catalog window in ArcPy, right-click on a folder > New > Toolbox.



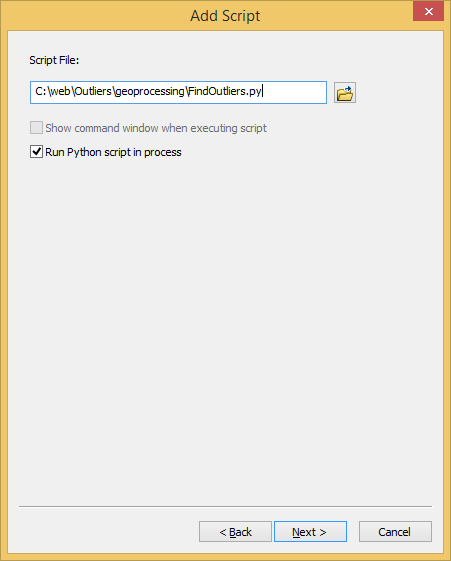
1. Right-click on the new toolbox > Add > Script



1. Fill in the name and label for the tool



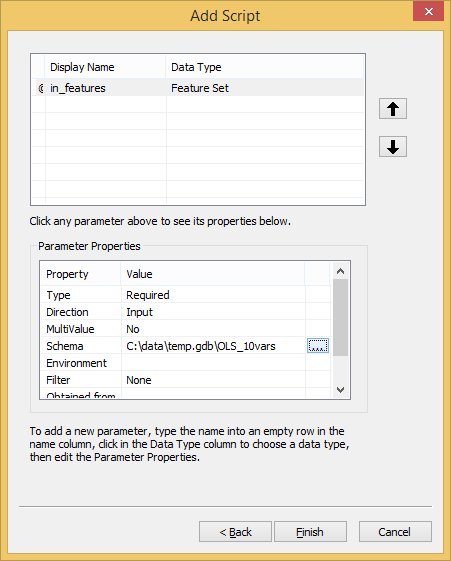
1. Navigate to your python script:



1. Enter the parameters that your tool uses, referring back to your python script to make sure they are in the correct order.

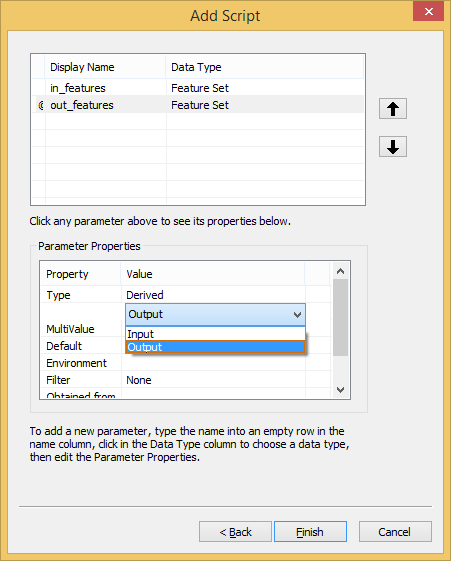
Some notes:

* + If your spatial input is a set of points, you will need to have a test dataset with the correct schema. Choose *Feature Set* as the Data Type and set a schema under Parameter Properties > Schema by clicking … and navigating to your test dataset with the correct schema. Although your tool will work fine in ArcMap without a defined schema, this is especially important because your geoprocessing service will not work, and you will not get any error messages that say you need to set a schema.



I am sure that there are other ways to input a set of points, possibly without needing to define a schema, but I was never able to get anything else to work.

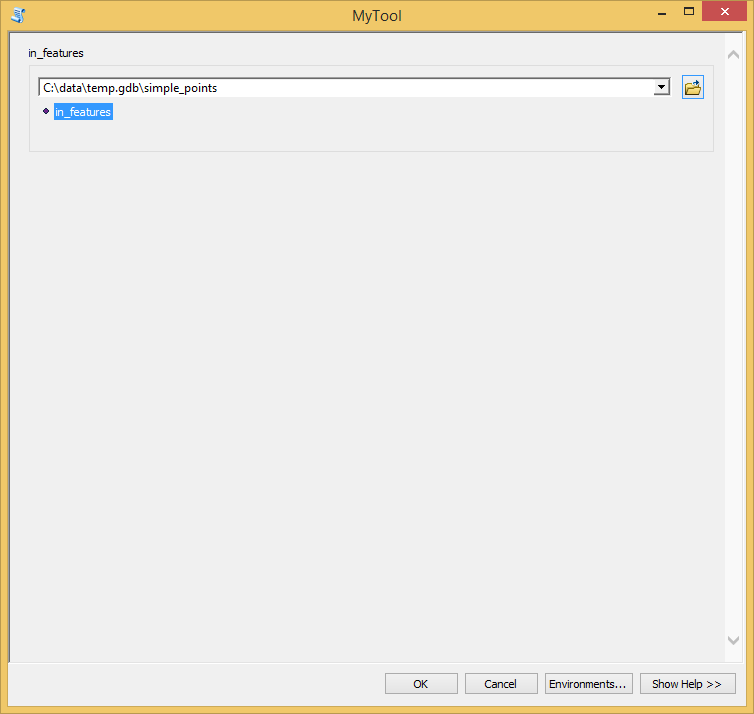
* + If your spatial input is a raster, set the Data Type to *Raster Dataset*.
  + Make sure to set your output parameters as well. Set their direction to Output under Parameter Properties > Direction. No schema needs to be set for output feature sets.

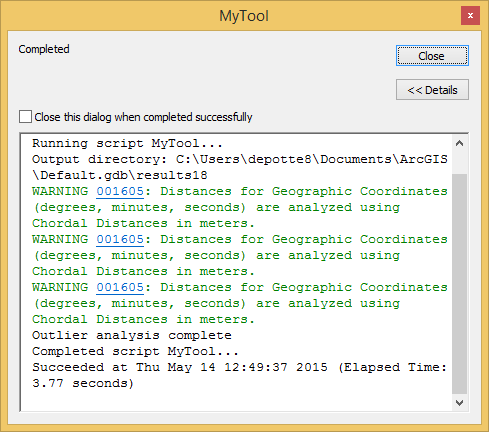


* + You can reorder parameters using the arrow buttons, which seems obvious, but it took me almost a whole year to realize it could be done.

## Run the tool and share as geoprocessing service

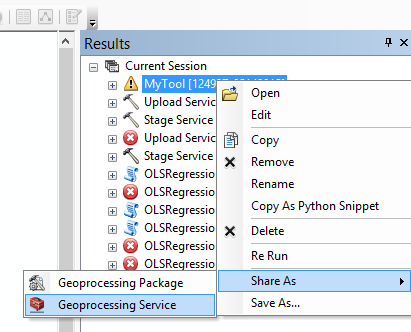
1. You can run your tool by double-clicking it in the Catalog window. Use your test dataset.



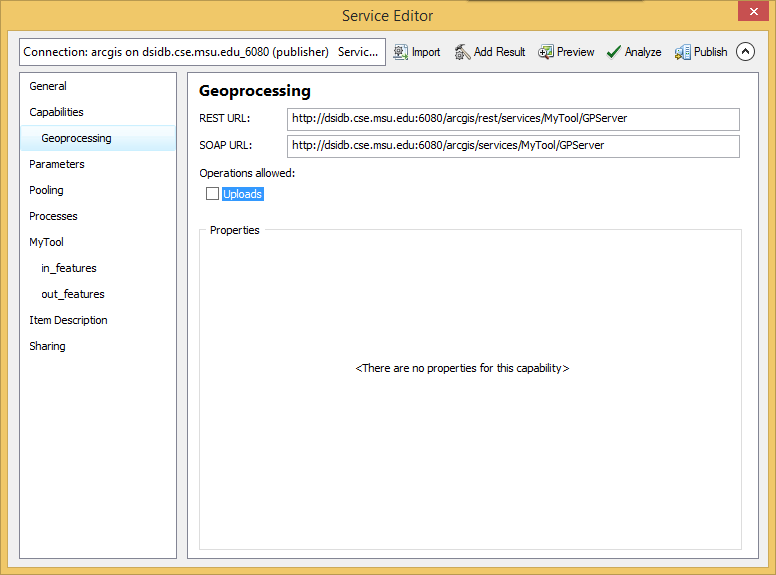


A note: I prefer to not to run geoprocessing tools in the background, as I have encountered bugs in the past with the background geoprocessing. To disable background geoprocessing, in your ArcMap menu, select Geoprocessing > Geoprocessing Options… and uncheck the Enable Background Geoprocessing box.

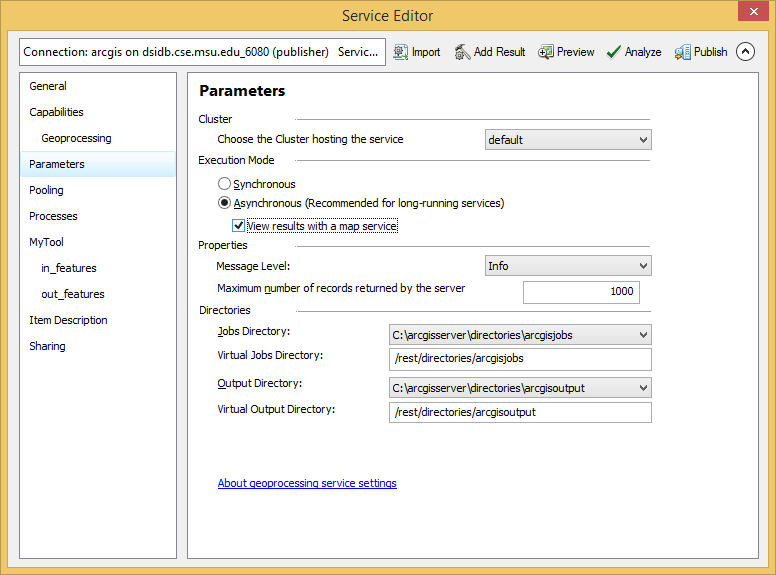
1. If your results panel is not already open, open it by selecting Geoprocessing > Results in the ArcMap menu.
2. You may have to troubleshoot your tool at this point. You can edit the python file directly. To re-run the tool with the same parameters, double-click the result in the Results panel that you wish to re-run.
3. After your tool has successfully completed, if your output data is a raster, edit the symbology on your result. This is the symbology that will be displayed in your application.
4. After your tool has successfully completed (and you have set the correct symbology), right-click on the successful result in the Results panel and choose Share as > Geoprocessing service.



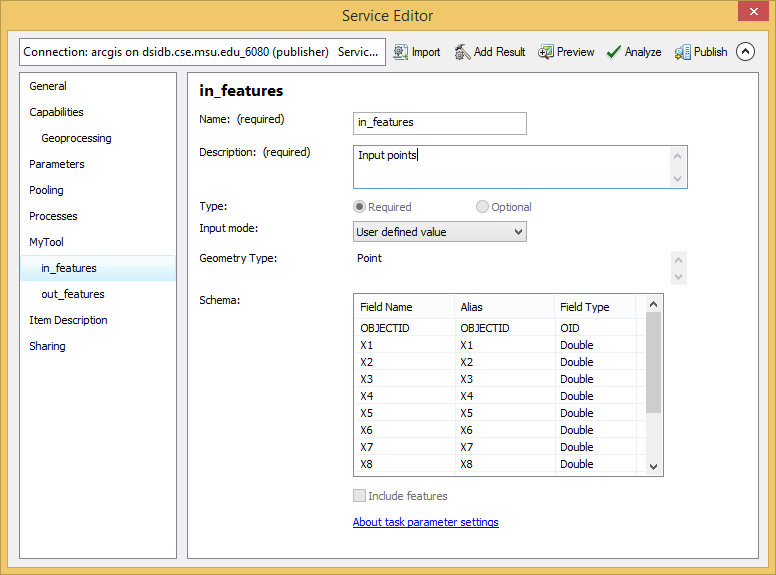
1. Select Publish a service, then Next. Choose a connection and a name for your geoprocessing service, then click Next.
2. If your tool input allows users to upload raster files, under Capabilities > Geoprocessing, check the Uploads box.



1. If your tool outputs a raster file, under Parameters > Execution Mode, check the View results with a map service box. This will create a map service to go along with the geoprocessing service that can be added to your map to view the output raster.
2. Update the message level under Parameters > Properties > Message Level. To see all messages printed by your script (using arcpy.AddMessage), choose “Info”.



1. Double-check that all your parameters are named appropriately and have a schema defined (if necessary). You will need to enter a description for each parameter.



1. Click “Analyze” and check to make sure that there are no errors. If not, click “Publish”.
2. Congratulations!! You’ve published a geoprocessing service. You can run the geoprocessing service through the REST endpoint (for example: [http://dsidb.cse.msu.edu:6080/arcgis/rest/services/Outlier/OutlierAnalysis /GPServer/OutlierAnalysis](http://dsidb.cse.msu.edu:6080/arcgis/rest/services/Outlier/OutlierAnalysis/GPServer/OutlierAnalysis/submitJob)).

## Using the geoprocessing service in JavaScript

For more information on all this, look at the Geoprocessor object in the API: <https://developers.arcgis.com/javascript/jsapi/geoprocessor-amd.html>

and the samples that use it: <https://developers.arcgis.com/javascript/jssamples/#search/Geoprocessor>

1. Create a parameters object. The field names on the parameters object need to exactly match the parameter names on the geoprocessing service (you can check this on the REST endpoint).
   * If your spatial input is a set of points, create graphics objects for each point, then create a FeatureSet object and set the features to your array of graphics:

var featureSet = new FeatureSet();

featureSet.features = graphics;

var params = {

in\_features: featureSet

};

(See the Outlier point application for an implementation.)

* + If your spatial input is a raster, first upload it using an esriRequest and get the itemId:

var itemid = "";

var uploadurl = "http://dsidb.cse.msu.edu:6080/arcgis/rest/services/Outlier/RasterOutlierAnalysis/GPServer/uploads/upload";

uploadRequest = esriRequest({

url: uploadurl,

form: dojo.byId(fileUploadId),

content: { "f": "pjson" },

load: function (response, io) {

var itemId = response["item"].itemID

callback(itemId);

},

error: function (error) {

console.log("error");

callback();

}

}, { usePost: true });

Then create an object using the itemId and set that as your parameter value:

var params = {

InputRaster: "{'itemID':" + itemid + "}"

};

(See the Outlier raster application for an implementation.)

1. Create a Geoprocessing object using the full endpoint to your geoprocessing service:

var gp = new Geoprocessor('http://dsidb.cse.msu.edu:6080/arcgis/rest/services/Outlier/OutlierAnalysis/GPServer/OutlierAnalysis');

1. Submit the job.

gp.submitJob(params, onComplete, onInfo, onFailure);

1. Once the job has completed, check to see if it has failed or not. If not request the output parameters, and do whatever the results. Again, the parameter name needs to exactly match the name in the geoprocessing service.

gp.getResultData(jobInfo.jobId, 'out\_features',

function (result) {

// do whatever here

});

1. If the geoprocessing service is returning a raster using a map service, add the map service to your map.

var mapServiceUrl = "http://dsidb.cse.msu.edu:6080/arcgis/rest/services/Outlier/RasterOutlierAnalysis/MapServer/jobs";

var resultLayer = new ArcGISDynamicMapServiceLayer(mapServiceUrl + "/" + jobInfo.jobId, {

"opacity": 1.0

});

map.addLayers([resultLayer]);

(See the Outlier raster application for an implementation.)

## Other notes & troubleshooting tips

* If you can’t get the geoprocessing tool to run in ArcMap, try turning off background geoprocessing.
* The symbology on your result in ArcMap before sharing will be the symbology that gets returned from the geoprocessing result.
* Check that the input schema is set and that your input data is following the correct schema.
* Check that you’re requesting the correct output parameter name.
* Check that you have the message level turned up so you can view the messages and errors in the server response.
* Sometimes, you can find more information by looking at the job directory on dsidb.

C:\arcgisserver\directories\arcgisjobs\outlier\outlieranalysis\_gpserver

* Try submitting directly through the REST endpoint instead of using JavaScript.
* Try to run the GP service using the exact same dataset used in ArcMap.