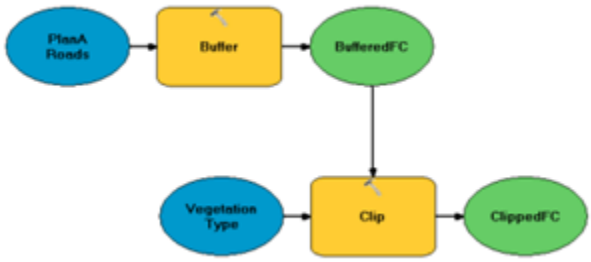
Chapter 18: Using the Geoprocessor

Geoprocessing refers to the automation and chaining of GIS operations in a logical fashion to accomplish some sort of GIS task. For example, you may want to buffer a stream layer and then clip a vegetation layer to this newly created buffer.

A model, like the one that you see below, can be built in ArcGIS Desktop using ModelBuilder. Models are run in an automated fashion from either a desktop environment or via a centralized server accessed through a web application.



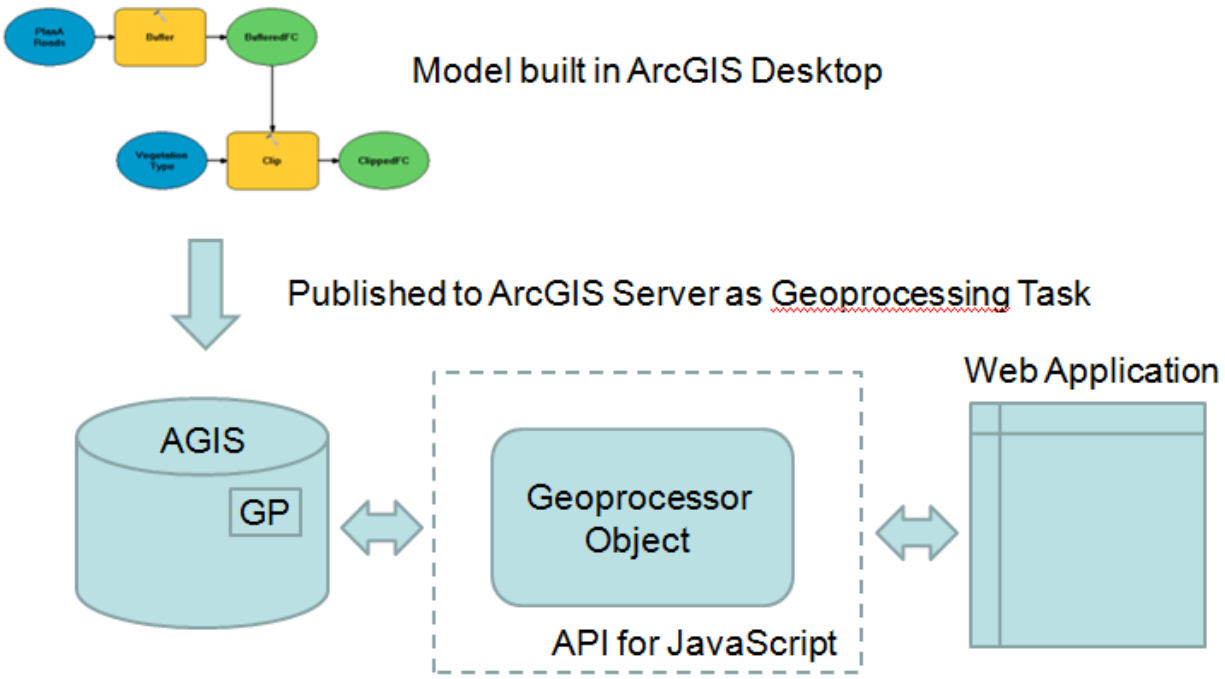
Any tool found in ArcToolbox whether that be a tool for your ArcGIS license level or a custom tool that you've built can be used in a model and chained together with other tools. Once constructed, these models can be run on a centralized server and accessed via web applications. In this section we will examine how you can access these geoprocessing tasks through the ArcGIS Server API for JavaScript.

What is a Geoprocessing Task?

Models are built in ArcGIS Desktop using ModelBuilder. Once built these models are published to ArcGIS Server as geoprocessing tasks. Web applications then use the Geoprocessor object found in the ArcGIS Server API for JavaScript to access these tasks and retrieve information. These models and tools are run on ArcGIS Server due to their computationally intensive nature and need for ArcGIS software.

Jobs are submitted to the server through a web application and the results are picked up after the service has completed. Submitting jobs and retrieving the results can be accomplished through the Geoprocessor object.

This process is illustrated in the figure below.



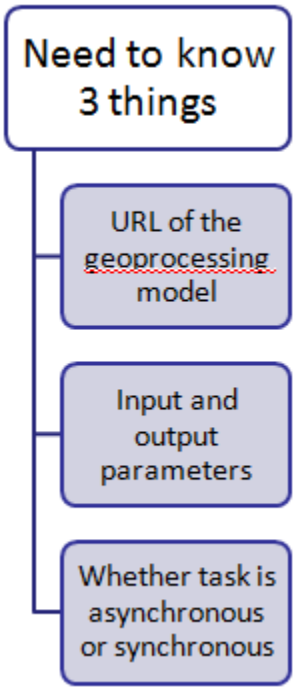
Obtaining Information about a Geoprocessing Task

There are three things that you need to know when using a geoprocessing service. First, you need to know the URL where the model or tool is located. An example URL is provided below:

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Population_World/GPServer/PopulationSummary

When you go to this link you can also find information about the input and output parameters, whether the task is asynchronous or synchronous and much more. Speaking of input and output parameters, you must know the datatypes associated with those parameters, and whether or not each of the parameters is required. Finally, you need to know whether the task is asynchronous

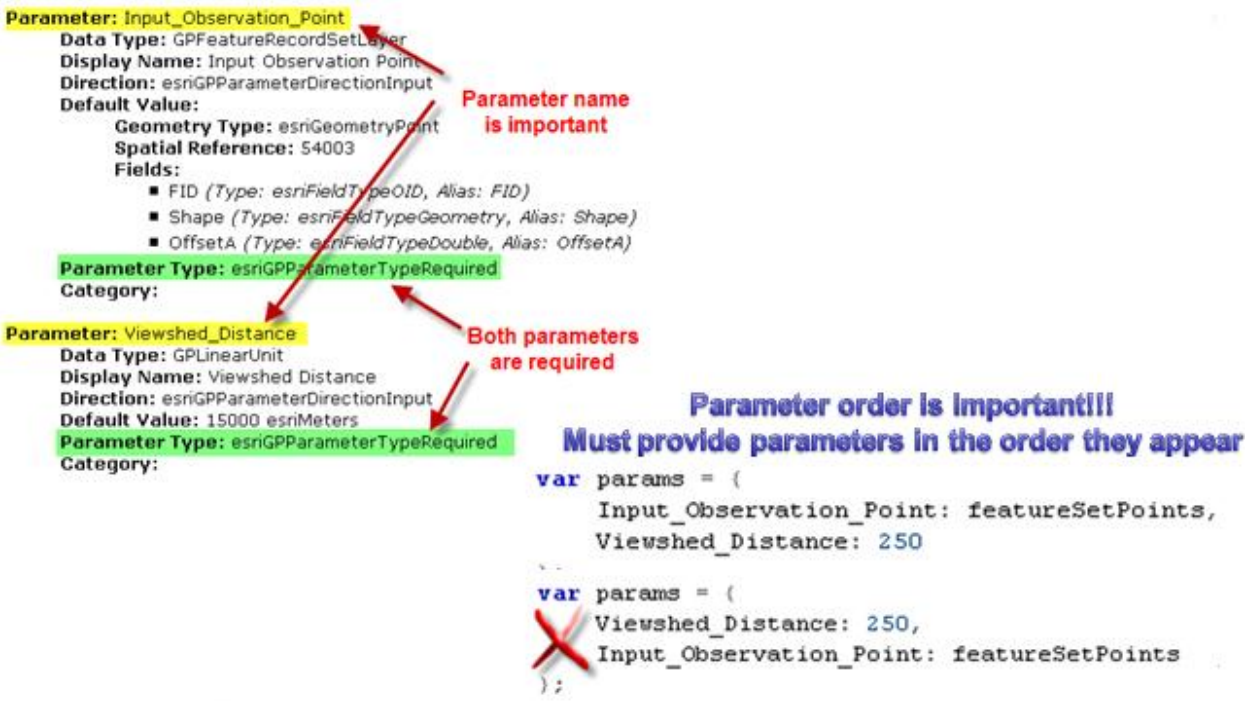
or synchronous and how your code should be configured based on that knowledge.



Input Parameters

There are a number of details that you must remember regarding input parameters that are submitted to the geoprocessing task. Almost all geoprocessing tasks will require one or more parameters. These parameters can be either required or optional and are created as JSON objects. When creating parameters as JSON objects you must remember to create them in the exact order that they appear on the services page. The parameter names must also be named exactly as they are on the services page.

Below you will see an illustration of the input parameters supplied to a geoprocessing task. When coding your JSON input parameters object it is critical that you provide the exact parameter name as given on the services page and that you provide the parameters in the order that they appear on the page. Notice in our code example that we are providing two parameters: Input_Observation_Point and Viewshed_Distance. Both parameters are required and we have named them exactly as they appear on the services page and they are in the correct order.



The Geoprocessor

The Geoprocessor class in the ArcGIS Server API for JavaScript represents a GP Task resource which is a single task in a geoprocessing service. Input parameters are passed into the Geoprocessor through a call to either Geoprocessor.execute() or Geoprocessor.submitJob. We'll discuss the difference between these two calls in more detail soon. After executing the geoprocessing task the results are returned to the Geoprocessor object where they are processed by a callback function.

Creating an instance of the Geoprocessor class simply requires that you pass in the url that points to the geoprocessing service exposed by ArcGIS Server. It does require that you import esri.tasks.gp.

Running the Task

Once you have an understanding of the geoprocessing models and tools available to you for an ArcGIS Server instance as well as the input and output parameters you can begin writing the code that will execute the task. Geoprocessing jobs are submitted to ArcGIS Server for either synchronous or asynchronous execution. A synchronous execution implies that the client calls for execution of the task and then waits for the result before continuing with the application code. In asynchronous execution a client submits a job, continues to run other functions, and checks back later for completion of the job. By default, the client checks back for completion every second until the job has finished. The Services page tells you how to submit your job for each geoprocessing task. Simply look for Execution Type on this page.

Synchronous tasks require that your application code submit a job and wait for a response before continuing. You will need to use the Geoprocessor.execute( ) method with the property input parameters and callback function supplied. The callback function is executed when the geoprocessing task returns the results of the job that was submitted. These results are stored in an array of ParameterValue.

Asynchronous tasks require that you submit a job, continue working on other functions while waiting for the process to finish, and then checking back in with ArcGIS Server on a periodic basis to retrieve the results after completion. The Geoprocessor.submitJob( ) method is used to submit a job to the geoprocessing task. You will need to supply input parameters, a callback function, and a status callback function. The status callback function executes each time your application checks back for the results. By default, the status is checked once per second. However, this interval can be changed using the Geoprocessor.setUpdateDelay( ) method. Each time the status is checked, a JobInfo object is returned and contains information indicating the status of the job. When JobInfo.jobStatus is set to STATUS_SUCCEEDED, the complete callback function is then called.

A visual diagram (seen below) of the process flow that occurs on asynchronous tasks can help to reinforce how these types of tasks operate. Input parameters are created and input to the geoprocessor object which uses these parameters to submit a geoprocessing job to ArcGIS Server. The Geoprocessor object then executes the statusCallback function at regular intervals. This function checks with the geoprocessing service to see if the job has finished. A JobInfo object is returned and contains a status indicator indicating it's completion status. This process continues until the job completes at which time a completeCallback function is called and passed the results of the job.