Esri Developer Summit in Europe

November 15 | Madrid

Building Rich GIS Capabilities into Web Applications

Andy Gup

Agenda

- What is Geoprocessing?
- How do you use Geoprocessing tasks?
- In-depth look at common coding patterns
- Synchronous patterns vs. Asynchronous
- Output as a Map Service
- Printing
- Editing

Assumptions

Knowledge of ArcGIS Desktop geoprocessing

Knowledge of how to create a feature service

Who am I?

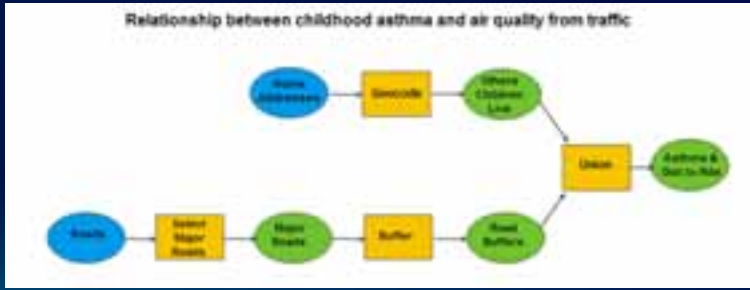
Andy Gup, Esri (U.S.)
Tech Lead for Web APIs and Android
Esri Developer Network

agup@esri.com
Twitter: @agup

What is Geoprocessing?

- Publish tools that can be used to automate the analysis of spatial data.
- Each tool, script and/or model in the toolbox becomes a task in 10.1.????

```
import arcpy
import os
import types

def convertPoints():
    arcpy.env.overwriteOutput = True

    # Input point FC
    # Output FC
    # Feature Field
    # Sort Field
    # Close Line or Leave Open
    inPts       = arcpy.GetParameterAsText(0)
    outFeatures = arcpy.GetParameterAsText(1)
    IDField     = arcpy.GetParameterAsText(2)
    sortField   = arcpy.GetParameterAsText(3)
    closeLine   = arcpy.GetParameterAsText(4)
```



What is a geoprocessing task?

- **Geoprocessing Task is a REST Endpoint to access server-side geoprocessing functionality.**
  - **REST Url : ArcGIS REST Services Directory/GPServer/<Task>**

ArcGIS Services Directory

Home > HomelandSecurity > Incident_Data_Extraction (GPServer) > Extract Data Task

Task: Extract Data Task

Display Name: Extract Data Task

Category:

Help URL: http://sampleserver4.arcgisonline.com/arcgisoutput/HomelandSecurity_Incident_Data_Extraction/ExtractDataTask.htm

Execution Type: esriExecutionTypeAsynchronous

Parameters:

Parameter: Layers_to_Clip
Data Type: GPMultiValue:GPString
Display Name: Layers to Clip
Direction: esriGPParameterDirectionInput
Default Value: Incident Points, Incident Lines, Incident Areas**Parameter Type:** esriGPParameterTypeRequired
Category:
Choice List:
- Incident Points
- Incident Lines
- Incident Areas

Parameter: Area_of_Interest
Data Type: GPFeatureRecordSetLayer
Display Name: Area of Interest

Demo: Geoprocessing Endpoint

Two Types of Task Operations

Execute Task

For faster processes (< 10 seconds)
Synchronous

```
gp.execute(params, completeCallback);
```

Submit Job

For long running processes (> 10 seconds)
Asynchronous/Cancellable

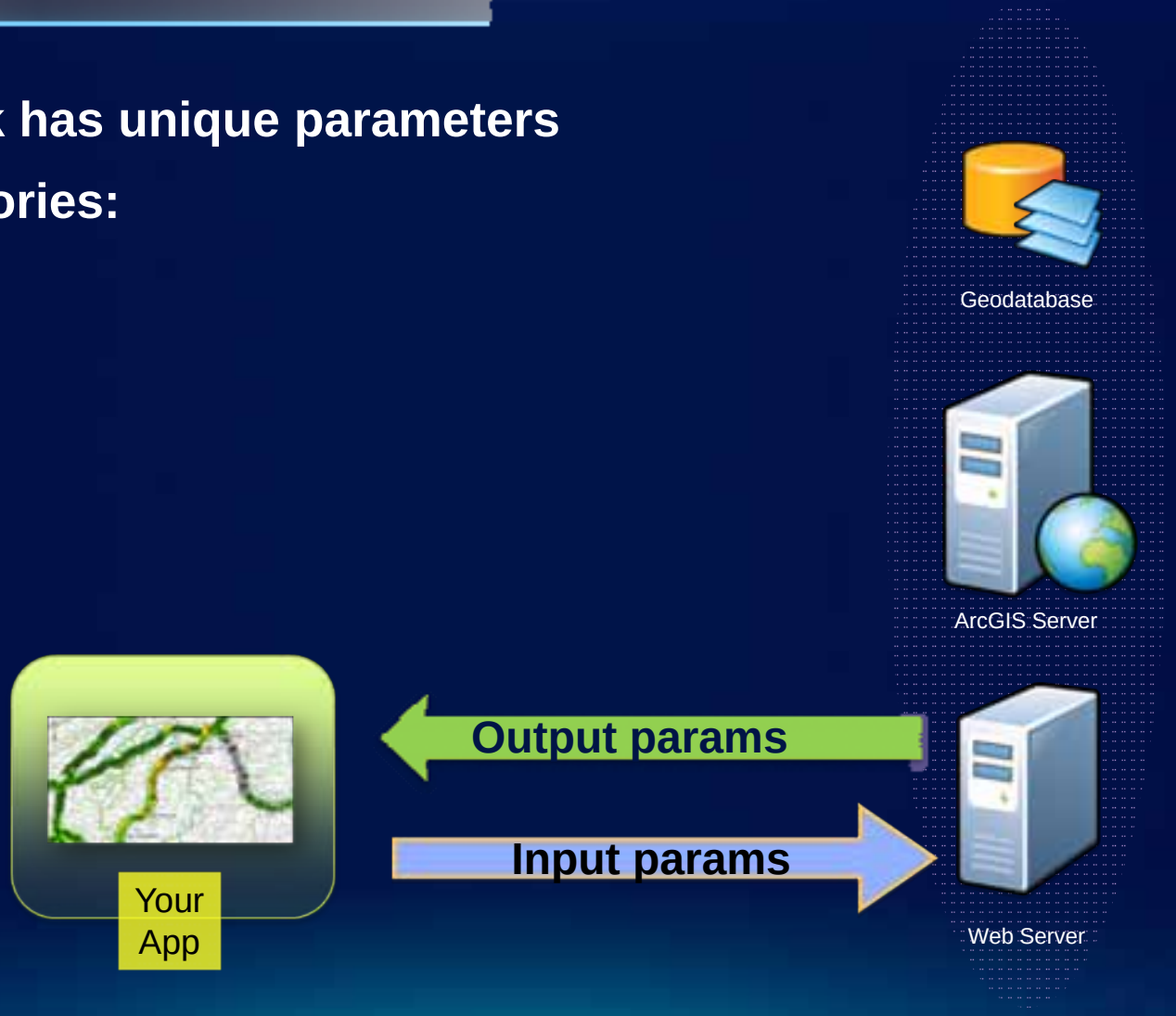
```
gp.submitJob(params, completeCallback , statusCallback);
```

Task Parameters

Every Task has unique parameters

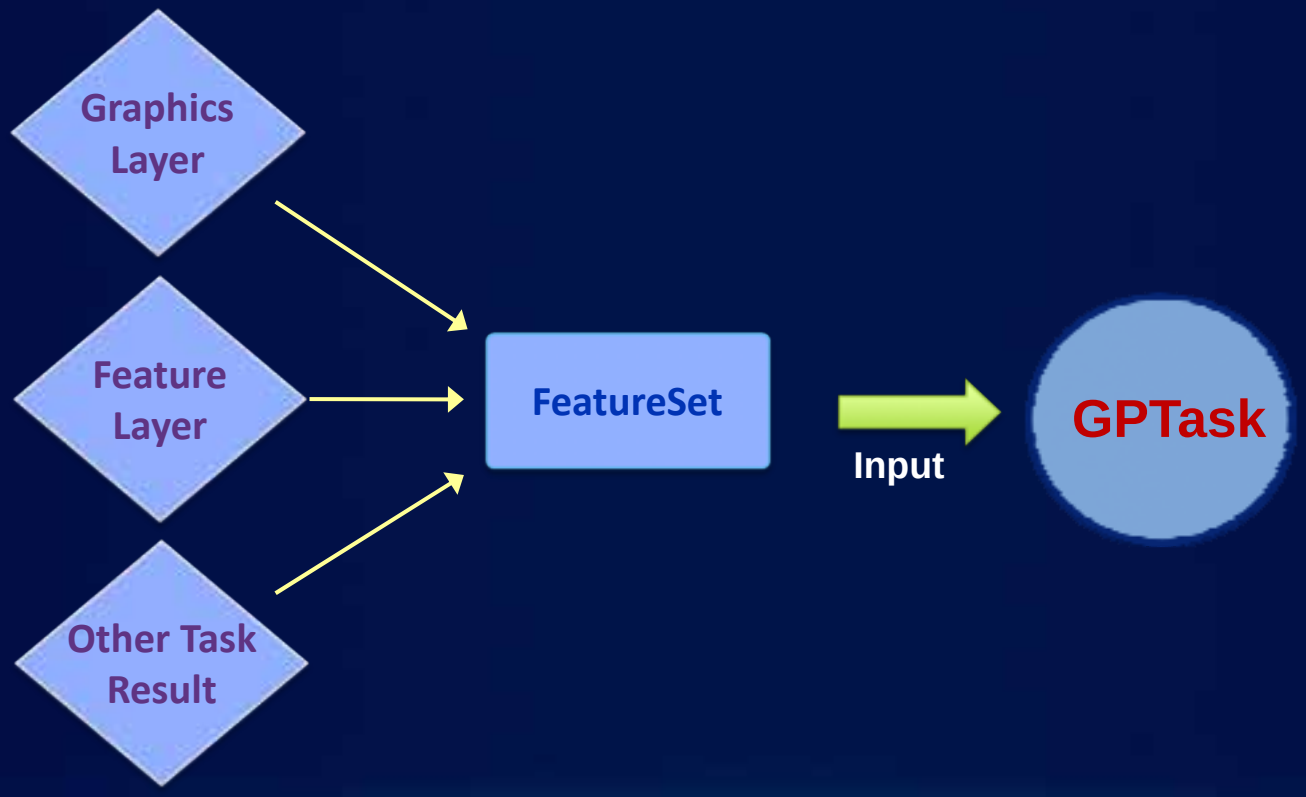
Two categories:

- Input
- Output



Task Parameters - Input

- Most common coding pattern



Input Task Parameters

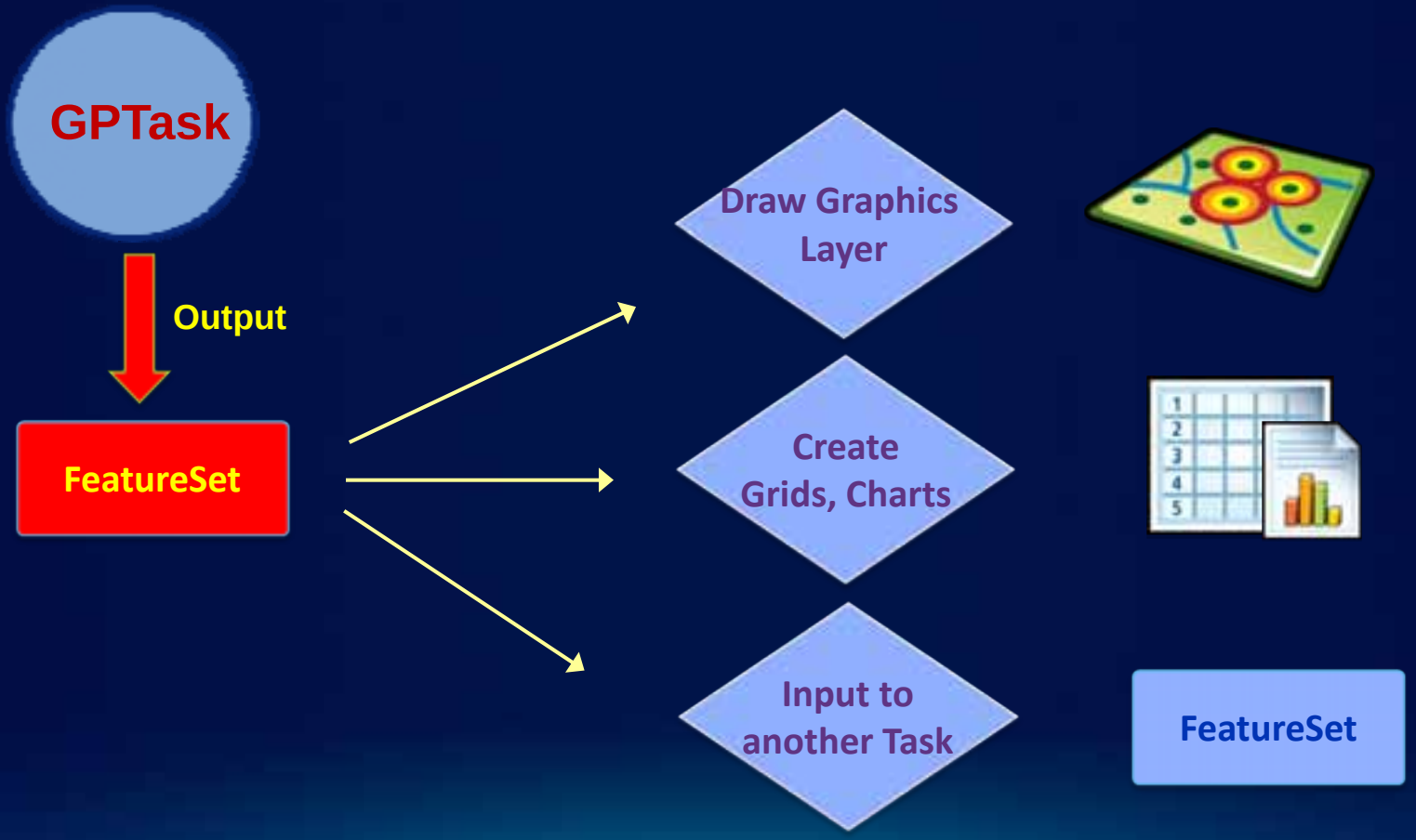
Parameter: Input_Observation_Point
- **Data Type:** GPFeatureRecordSetLayer
- **Display Name:** Input Observation Point
- **Direction:** esriGPParameterDirectionInput
- **Default Value:**
  - **Geometry Type:** esriGeometryPoint
  - **Spatial Reference:** 54003
  - **Fields:**
    - FID *(Type: esriFieldTypeOID, Alias: FID)*
    - Shape *(Type: esriFieldTypeGeometry, Alias: Shape)*
    - OffsetA *(Type: esriFieldTypeDouble, Alias: OffsetA)*
- **Parameter Type:** esriGPParameterTypeRequired
- **Category:**

Parameter: Viewshed_Distance
- **Data Type:** GPLinearUnit
- **Display Name:** Viewshed Distance
- **Direction:** esriGPParameterDirectionInput
- **Default Value:** 15000 esriMeters
- **Parameter Type:** esriGPParameterTypeRequired
- **Category:**

```
var params = {
    "Input_Observation_Point":featureSet,
    "Viewshed_Distance":vsDistance
};
gp.execute(params, drawViewshedComplete);
```

Task Parameters - Output

- Most common web application pattern



Output Task Parameters

Parameter: Viewshed_Result
Data Type: GPFeatureRecordSetLayer
Display Name: Viewshed Result
Direction: esriGPParameterDirectionOutput
Default Value:
Parameter Type: esriGPParameterTypeRequired
Category:

```
function drawViewshedComplete(results, messages) {
    var features = results[0].value.features;
    ...
    ...
    ...
}
```

Types of task parameters

FeatureSet

RecordSet (Tables)

Raster

Linear Unit (100 Km, 50 Miles…)

File (.xml, .gpx, .txt, .zip ….)

Integer, Long, Double

Date

Boolean

Example Output FeatureSet

- Output FeatureSets will also have Fields property.


```
{ "geometryType" : "esriGeometryPoint",
  "fields":[ {"name":"Id","type“: "esriFieldTypeOID","alias":"Id"},
             {"name":"Name","type":"esriFieldTypeString","alias":"Name“}],
  "spatialReference" : {"wkid" : 4326},
  "features" : [ { "geometry" : {"x" : -104.44, "y" : 34.83,
                   "spatialReference" : {"wkid" : 4326}},
                   "attributes" : {"Id" : 43,"Name" : "Feature 1"}},
                 { "geometry" : {"x" : -100.65, "y" : 33.69,
                   "spatialReference" : {"wkid" : 4326}},
                   "attributes" : {"Id" : 42,"Name" : "Feature 2"}} ]
}
```

Many features or complex geometries?

If request > ~2048 characters

Use a proxy

```
esriConfig.defaults.io.proxyUrl = "/proxy/proxy.ashx";
esriConfig.defaults.io.alwaysUseProxy = false;
```

Coding Pattern for Tasks

Geoprocessing tasks can be accessed through all ArcGIS Web APIs and Runtime SDKs

Four simple steps

1. Initialize task
2. Setup task parameters
3. Run the task
4. Get results from the task and process.

Step 1 – Initialize Geoprocessing Task

```
dojo.require("esri.tasks.gp");

var gp;

gp = new esri.tasks.Geoprocessor(
    “http://hostname/ArcGIS/rest/services/GPSServer/ServiceName");

gp.setOutputSpatialReference({wkid:102100});
```

Step 2 – Set up parameters

```
var graphic = new esri.Graphic(evt.mapPoint,pointSymbol);
var features= [];
features.push(graphic);
var featureSet = new esri.tasks.FeatureSet();
featureSet.features = features;
var vsDistance = new esri.tasks.LinearUnit();
vsDistance.distance = 5;
vsDistance.units = "esriMiles";
var params = {
    "Input_Observation_Point":featureSet,
    "Viewshed_Distance":vsDistance
};
```

Step 3 – Run the Task

```
gp.execute(params, completeCallback);
```

OR

```
gp.submitJob(params, completeCallback , statusCallback);
```

Step 4 – Process result set (Synchronous)

```
function drawViewshedComplete(results, messages) {
    var features = results[0].value.features;
    for (var f=0, fl=features.length; f<fl; f++) {
      var feature = features[f];
      feature.setSymbol(polySymbol);
      map.graphics.add(feature);
    }
}
```

Step 4 – Process result set (Asynchronous)

```
function doSomething(){
   gp.submitJob(params, completeCallback , statusCallback,function(error){
     alert(error);
   });
 }
 function completeCallback(jobInfo){
   if(jobInfo.jobStatus !== "esriJobFailed"){
     gp.getResultData(jobInfo.jobId,"Output_Zip_File", downloadFile);
   }
 }
 function statusCallback(jobInfo) {
   var status = jobInfo.jobStatus;
   if(status === "esriJobFailed"){
     alert(status);
   }
   else if (status === "esriJobSucceeded"){
     alert(“job finished”);
   }
 }
 function downloadFile(outputFile){
   map.graphics.clear();
   var theurl = outputFile.value.url;
   window.location = theurl;
 }
```

Looking at different coding patterns

Task Result as a Map Service

Printing

Task Result as Map Service

- How to Add Raster output to web map?
  - **Problem:** Raster cannot be drawn by Web APIs
  - **Solution:** Result Map Service

ArcGIS REST Services Directory

Home > services > 911CallsHotspot (GPServer)

JSON | SOAP | WPS

911CallsHotspot (GPServer)

Service Description: The tool filters 911 calls based
created using the Spatial Statistics Hot Spot Analysis

Tasks:

- 911 Calls Hotspot

Execution Type: esriExecutionTypeAsynchronous

Result Map Server: 911CallsHotspot

MaximumRecords: 1000

Child Resources: Info Uploads

Task Result as Map Service

Three good reasons to return task result as Map Service :

1. Output cannot be drawn by Web Clients
2. Output is too big to download
3. Better Cartographic Maps

Task Result as Map Service

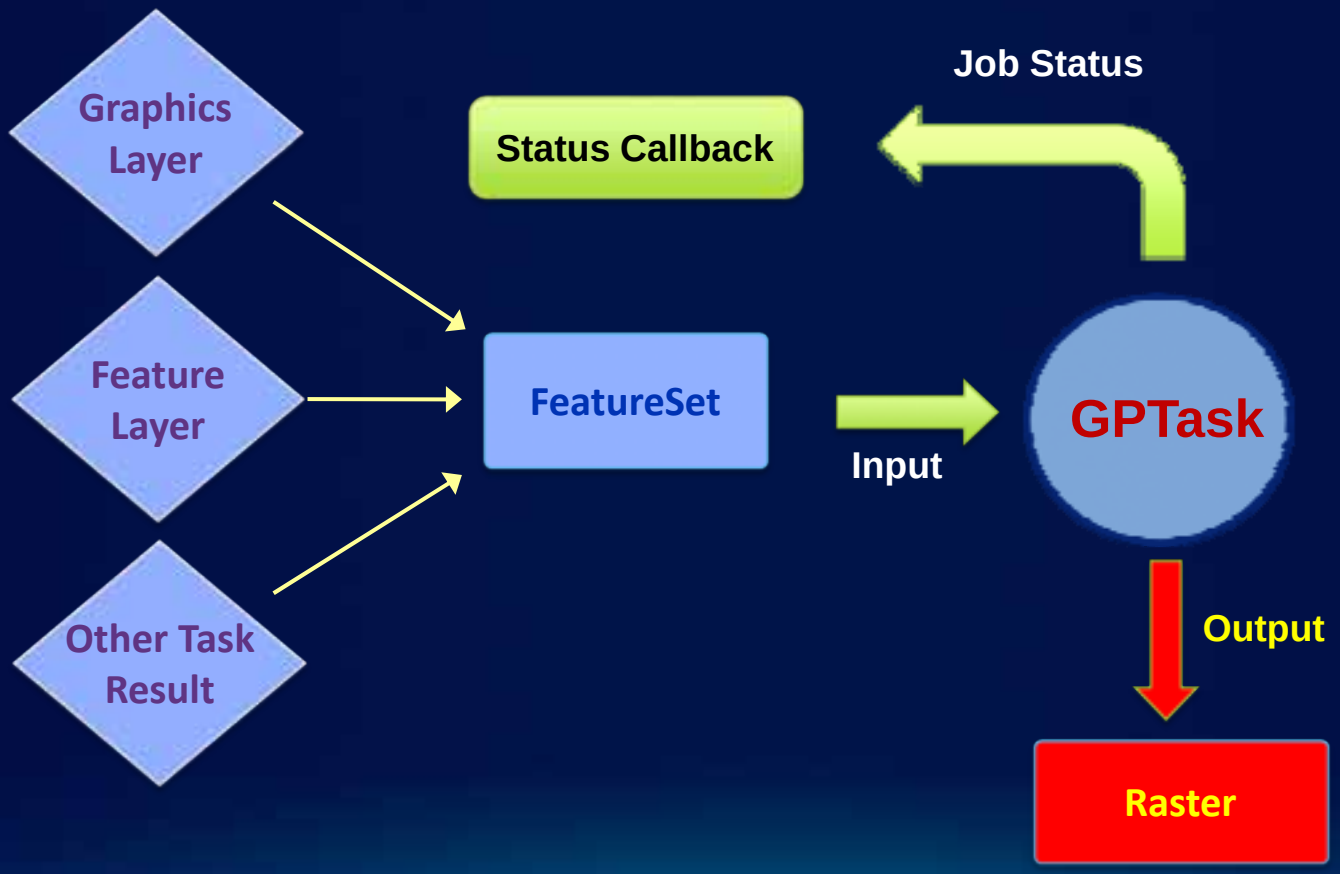
- Easy to access map service and its layers
- Download features based on query (Web API Feature Layer support)
- Dynamic Legends
- Time enabled layers
- Attachments and esriHTMLPopup support

Task Result as Map Service

Advantages

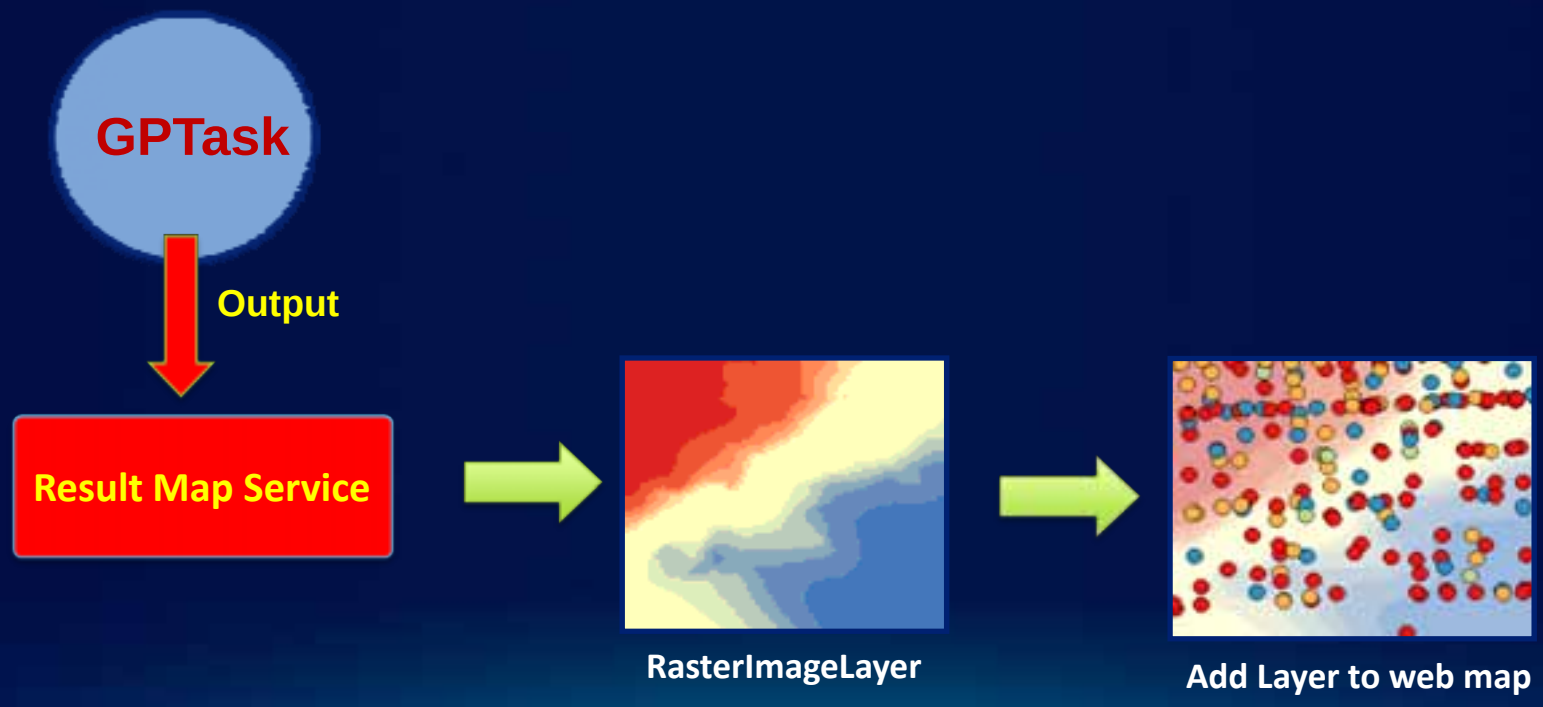
- Copyright protection
- Supports large dataset return values.
- Pre-cooked symbology.
- Raster outputs as layers.
- Support for advanced layers such as TIN, CAD etc.

Task Result as Map Service - Asynchronous



Task Result as Map Service

- Result Map service returns an Image Layer
- Add Layer to Web map to show result
- Asynchronous only!



Demo - Result Map Service

Asynchronous

Printing – Export Web Map Task

Problem: Need to print map including graphics and layers.

Solution: Export web map service

ArcGIS REST Services Directory

Home > services > Utilities > PrintingTools (GPServer) > Exp

JSON

Task: Export Web Map Task

Display Name: Export Web Map Task

Description: This Export Web Map Task takes the state of the
and returns either (a) a page layout or (b) a map without page
piece of text in JavaScript object notation (JSON) format descr
WebMap specification in the ArcGIS HelpThis tool is shipped wi
PrintingTools. The ArcGIS web APIs for JavaScript, Flex, and S
uses a model containing the Export Web Map tool.

Category:

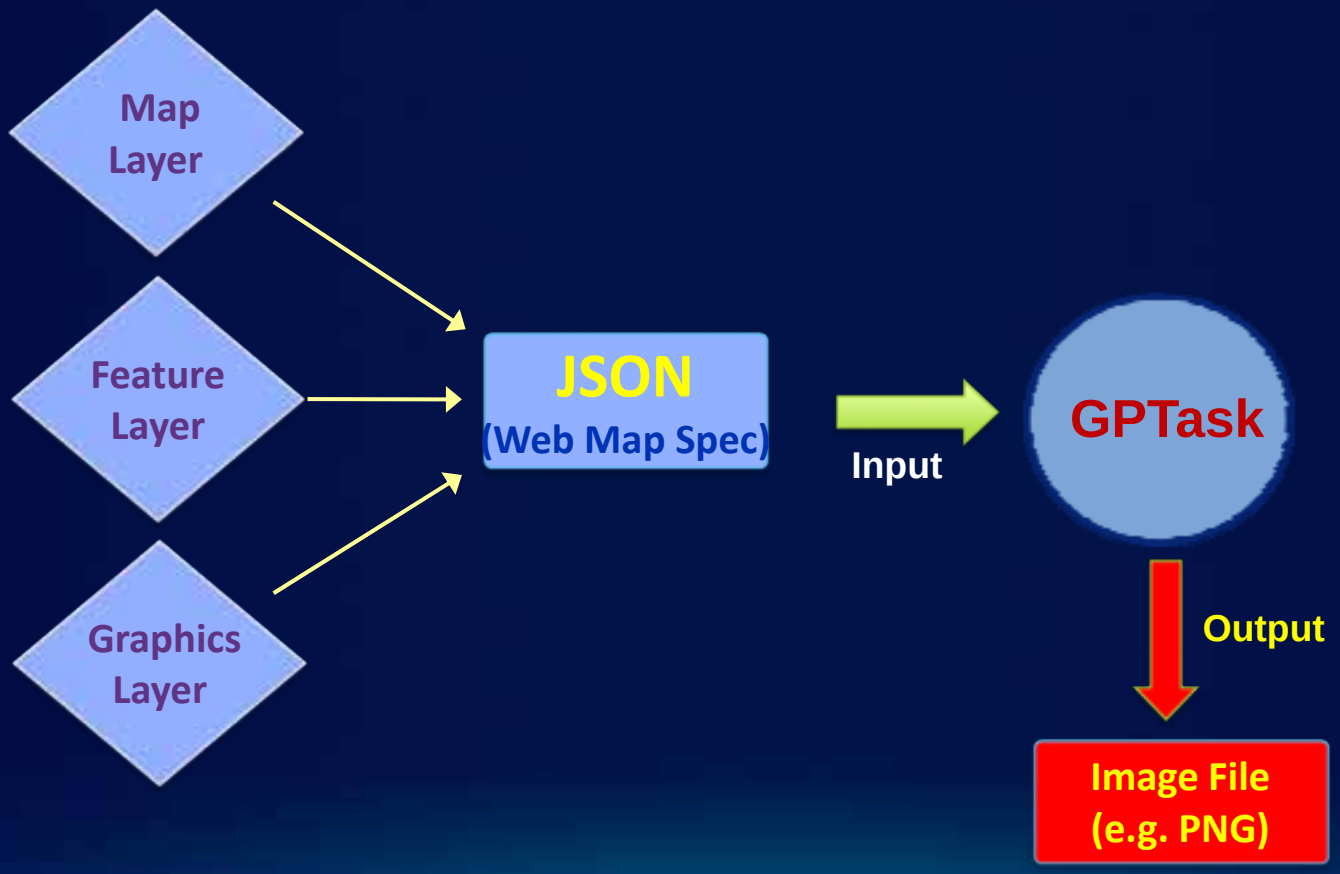
Help URL: http://sampleserver6.arcgisonline.com/arcgis/rest/

Execution Type: esriExecutionTypeSynchronous

Parameters:

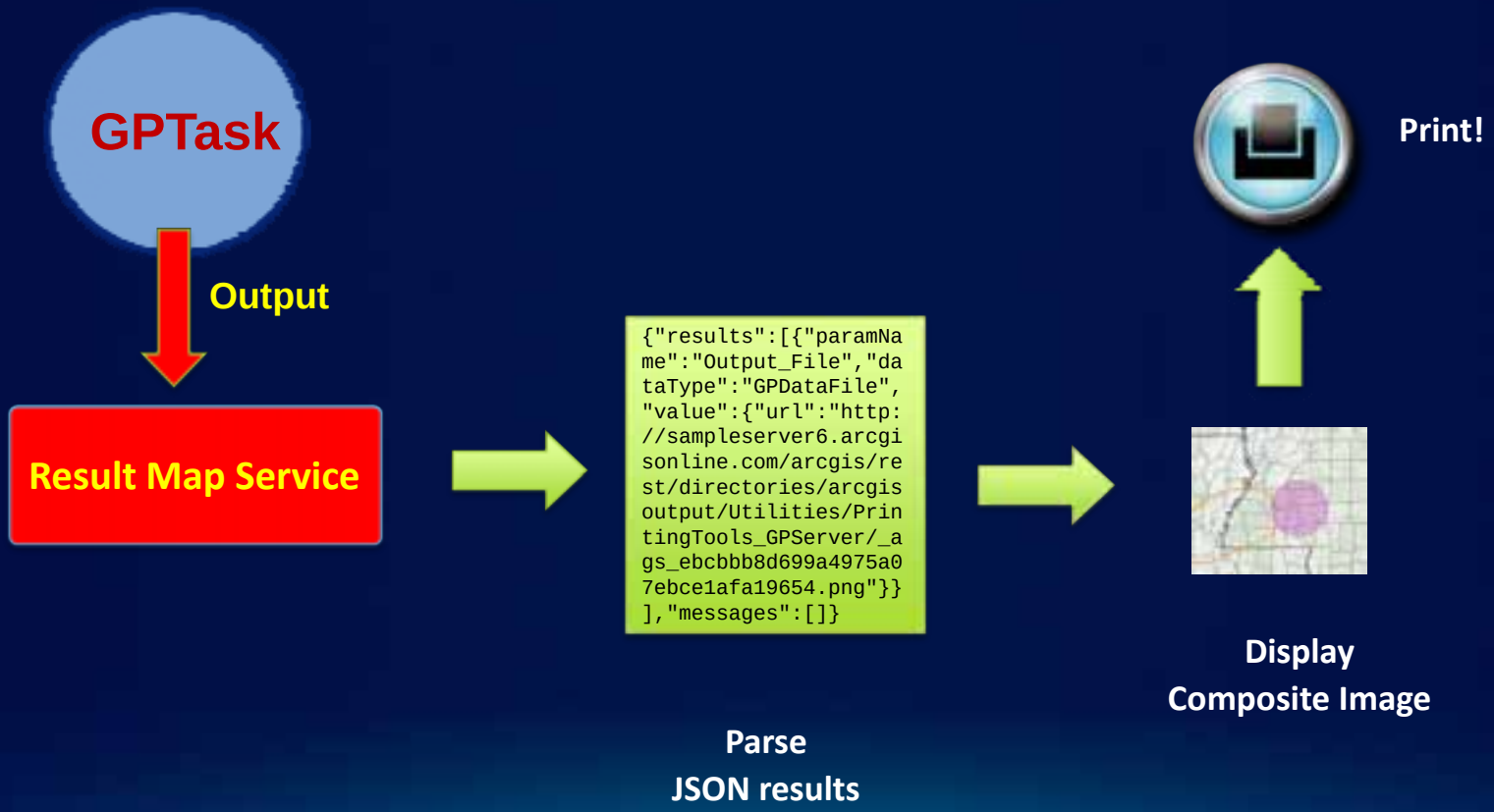
Parameter: Web_Map_as_JSON
Data Type: GPString
Display Name Web Map as JSON
Description: A JSON representation of the state o

Printing – Export Web Map Task



Printing – Export Web Map Task

- Export Web Map Task returns a URL to an image
- Provide a UI link to the URL



Demo - Export Web Map

Synchronous

Editing

Modify data over the web

- Add
- Delete
- Move
- Cut
- Union
- Edit Attributes
- Reshape

Requires a **Feature Service**

Requires an **ArcSDE Geodatabase**

Geometry Service (Optional)

Feature Template (Optional)

Editing (Pre-built components)

Editor Widget

Template Picker

Attribute Inspector

Attachment Inspector

Pro's:
- Ready out-of-the-box
- Configure and go

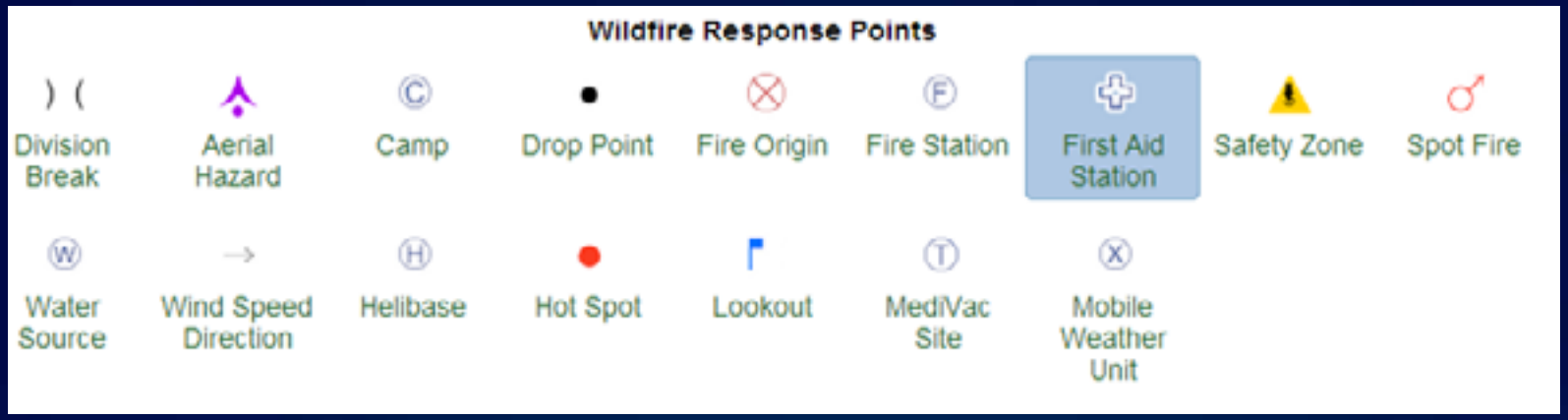
Cons:
- No access to widget source
- Limited configuration options

Template Picker

Uses pre-configured feature choices

Provides quality control

Can also be used a legend



Template Picker

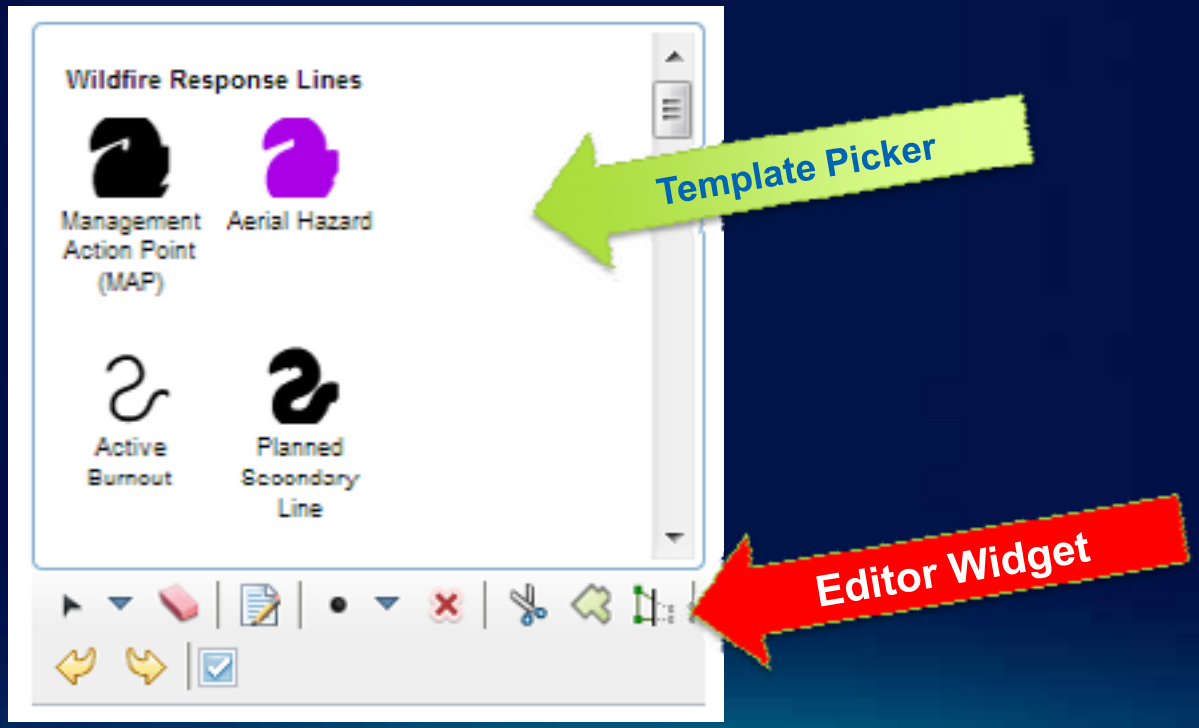
```
<div id="templatePickerDiv"
    style=".selectedItem{background-color:orange;}" >
</div>

dojo.require("esri.dijit.editing.TemplatePicker-all");

var widget = new esri.dijit.editing.TemplatePicker({
    featureLayers: layers,
    rows: "auto",
    columns:9,
    showTooltip:true,
    style: "height: 100%; width: 900px;"
  }, "templatePickerDiv“
);

widget.startup();
```

```
dojo.connect(widget, "onSelectionChange", function() {
    var selected = widget.getSelected();
    //TODO
});
```

Editor Widget

Provides all functionality needed

Customize functionality



Editor Widget

```
<div id="templatePickerDiv“ />
<div id="editorDiv" />

var params = {settings:{
    map: map,
    layerInfos: true,
    ...}
};

var editorWidget = new esri.dijit.editing.Editor(
    params,'editorDiv’
);

editorWidget.startup();
```

```
dojo.connect(templatePkr, "onSelectionChange", function() {
    if(templatePkr.getSelected()){
});
```

Demo – Editor Widget

Questions ?

Please fill session survey!