

Esri Developer Summit

March 26–29, 2012 | Palm Springs, California

esri.com/events/devsummit



Creating Geoprocessing Services

Kevin Hibma, Scott Murray

A decorative graphic at the bottom of the slide. It features a curved orange border at the top. Below it is a semi-transparent map of a landscape with green fields and blue water. Overlaid on the map is white text representing code snippets. The code includes symbols like 'esri.symbol.SimpleLineSymbol', 'new dojo.Color([0,0,0,0.5])', and 'feature.setSymbol'.

```
esri.symbol.SimpleLineSymbol
new dojo.Color([0,0,0,0.5])
polySymbol(feature.setSymbol)
} else if(f == 1) {
var polySymbolGreen =
polySymbolGreen.setOutline
symbol.SimpleLineSymbol(esri.symbol
Color([0,0,0,0.5]), 1));
polySymbolGreen.setSymbol(new dojo
feature.setSymbol(polySymbolGreen)
== 2) {
polyBlue = new esri.symbol.SimpleLineSymbol
setOutline(new
esri.symbol.SimpleLineSymbol
new dojo.Color([0,0,0,0.5]), 1));
```

Geoprocessing Services

- **The geoprocessing service allows you to publish custom tools to be used via ArcGIS Server**
- **Geoprocessing services can be used by many different client applications**
 - **ArcGIS Desktop**
 - **ArcGIS Engine**
 - **ArcGIS Explorer**
 - **WSDL**
 - **REST**
 - **JavaScript**
 - **FLEX**
 - **Silverlight**

Geoprocessing Services

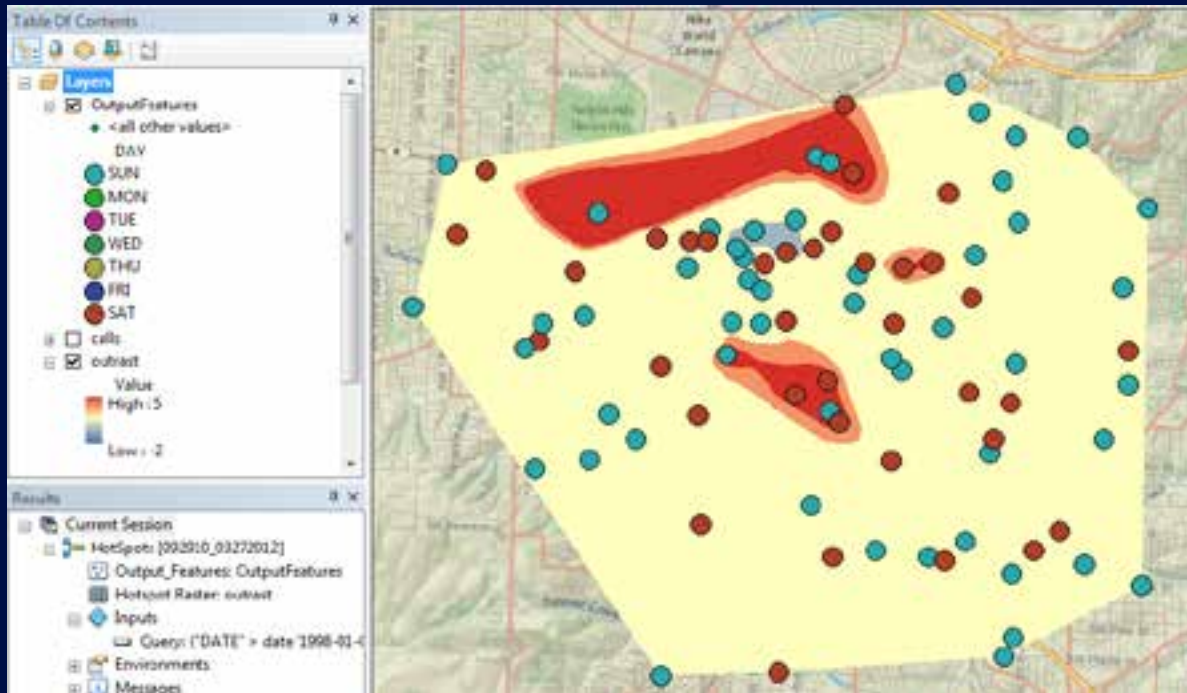
- **The service is composed of both the tools and the data needed by the tools**
- **Endless array of tasks can be created**
 - **Spatial analysis (vector, raster, network...)**
 - **Data Management (geodatabase, file based data)**
 - **Conversion (ETL and data loading)**
- **You need to be knowledgeable about using geoprocessing tools to create a good geoprocessing service**

Geoprocessing Service Behavior

- **Geoprocessing Services are very flexible and allow many different behaviors**
- **Before Authoring and Publishing, identify what you want your service to do and how you want it to behave with clients.**
 - **Input data from the client or select data on the server?**
 - **Draw results with map server or download and draw data on the client?**
 - **Save data on the server?**

How to create a service

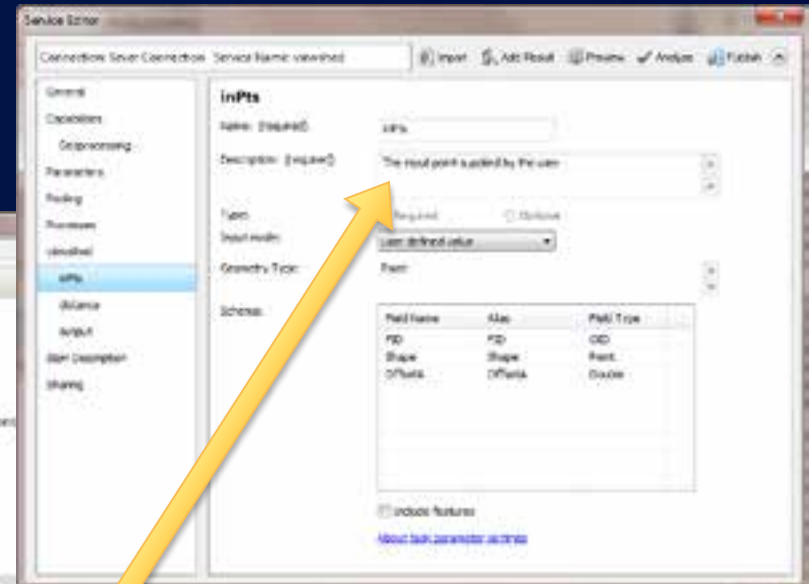
- All services start from a successful **result**
- The result acts as a template to build the service



Quick Tour of Publishing: <http://esriurl.com/gpSrvQuick>

Documenting your task

- All tasks must be documented
- Best practice to fill out the Item Description
- You can update metadata specific to the task you are publishing inside the Service Editor



Documenting your service:
<http://esriurl.com/gpSrvDoc>

Publishing Wizard

- **Manages parameter types**
- **Makes model or script portable**
 - **Fixes paths to data inside model and scripts**
- **Makes sure data is accessible to the service**
 - **What data is needed is packaged**

Demo – Surface Analysis

Creating Stack Profile and Viewsheds



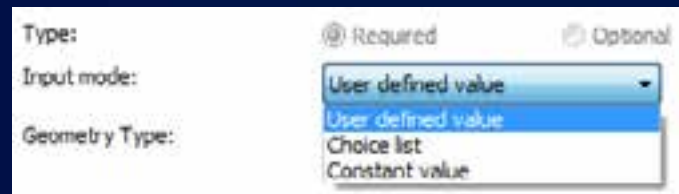
```
esri.symbol.SimpleLineSymbol({
  color: [0, 0, 255],
  width: 2,
  style: 'solid'
});
new dojo.Color([0, 0, 255]);
feature.setSymbol(new esri.symbol.SimpleLineSymbol({
  color: [0, 0, 255],
  width: 2,
  style: 'solid'
}));
feature.setColor(new dojo.Color([0, 0, 255]));
feature.setSymbol(new esri.symbol.SimpleLineSymbol({
  color: [0, 0, 255],
  width: 2,
  style: 'solid'
}));
feature.setColor(new dojo.Color([0, 0, 255]));
```


Surface Analysis Service Characteristics

- **Viewshed**
 - **Synchronous**
 - **Inputs : point feature, distance**
 - **Output: area features**
 - **Project Data: raster**
- **Stack Profile**
 - **Synchronous**
 - **Inputs : line feature**
 - **Output: table**
 - **Project Data: raster**

Parameter transformation

- Unsupported parameter types are handled through publishing
- You can update the Input Mode depending on the parameter type
- User Defined Value: allows the end user to interactively add features or enter text and number values
- Choice list: allows the end user to select from a list of layers already on the server
- Constant value: hard codes the parameter; the end user will not be able to provide input

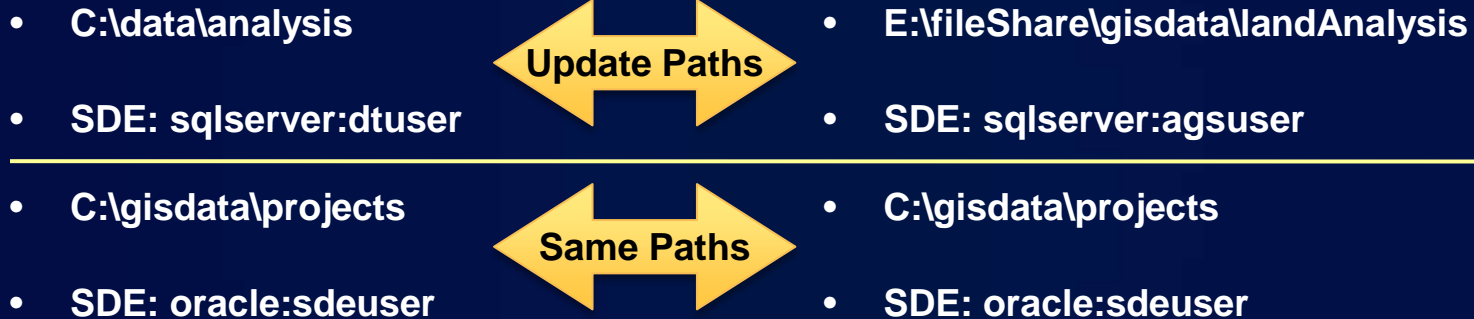


The screenshot shows a configuration window with the following elements:

- Type:** A label on the left.
- Input mode:** A dropdown menu with a blue header and a downward arrow. The menu is open, showing four options: "User defined value" (highlighted in blue), "User defined value", "Choice list", and "Constant value".
- Geometry Type:** A label on the left.
- Required/Optional:** Two radio buttons at the top right. The "Required" button is selected (indicated by a filled circle), and the "Optional" button is unselected (indicated by an empty circle).

Accessing your data

- Data Store tells ArcGIS Server about your data
- Without a Data Store entry, all required data is copied to the server
- Data Store acts as a lookup table



Data Store: <http://esriurl.com/datastore>

What happens during publishing?

- **Project data**
 - Copied if not in data store
 - Path updated if registered with the data store
- **Output and Intermediate paths**
 - Changed to scratchFolder and scratchGDB

Multiple Tasks

- **Use Add Result to create a service with multiple tasks**



- **Use Preview to see how the task would appear to a user consuming the service from ArcMap**

Sharing your service

- Make your service discoverable on ArcGIS.com
- Provide good metadata and search tags

Sharing

Share your service with:

My Content

Everyone (public)

Members of these groups:

<input checked="" type="checkbox"/>	 Esri Geoprocessing Services v 10.1
<input type="checkbox"/>	 MapSAR
<input type="checkbox"/>	 Python Test
<input type="checkbox"/>	 Washington DC Crime Analysis

Result Map Service

- **A result map service (RMS) provides an alternative way to get results from the Geoprocessing Service.**
- **An image is returned to the client.**
 - **The data can still be downloaded.**
- **Use a RMS when:**
 - **Want better cartography than the client can support**
 - **It is impractical to render a large dataset in a client.**
- **Execution must be Asynchronous when using a RMS**

Synchronous vs. Asynchronous

- **Execution mode defines how the client interacts with service while it executes**
 - **Synchronously: the client waits for the server to finish executing and then gets the result.**
 - **Asynchronously: client must ask the server if its finished then get the result. The client is free to do other work during this time.**
 - **Can only use a Result Map Service with Async.**
 - **Synchronous services are typically fast services**

Synchronous

- **Synchronous (Execute)**
 - Client always receives and draws data.
 - Desktop Client waits until job is completed and results are returned
 - Appropriate for faster processing jobs. (<10 seconds)

Asynchronous

- **Asynchronous (Submit Job)**
 - Results are saved on the server
 - Results can be drawn on the server
 - Results can also be downloaded if desired
 - Clients free to do other tasks
 - you can pan/zoom, run other tools while the job is running
 - Appropriate for longer processing jobs.

Demo – 911 Calls (with Result Map Service)

```
esri.symbol.SimpleLineSymbol({color:[0,0,0],width:2})
new dojo.Color([0,0,0]);
polySymbol(feature.setSymbol(polySymbolGreen));
} else if(f == 1) {
var polySymbolGreen = new esri.symbol.SimpleLineSymbol({color:[0,0,0],width:2});
polySymbolGreen.setOutlineColor([0,0,0,0.5]);
polySymbol(feature.setSymbol(polySymbolGreen));
} else if(f == 2) {
var polySymbolBlue = new esri.symbol.SimpleLineSymbol({color:[0,0,255],width:2});
polySymbolBlue.setOutlineColor([0,0,255,0.5]);
polySymbol(feature.setSymbol(polySymbolBlue));
}
```

911 Call Service Characteristics

- **Asynchronous**
- **Result Map Service**
- **Inputs : query (string)**
- **Output: raster, point features through RMS**
- **Project Data: point features**

Creating Script Tools

- **Paths and data handled the same as models**
- **Importing of modules**
 - **First looked in the same folder as source script**
 - **Second the PythonPath is searched**
- **Output and Intermediate paths**
 - **os.path.join(arcpy.env.scratchFolder, "out.shp")**
 - **os.path.join(arcpy.env.scratchGDB, "out")**
 - **In_memory\out**

Authoring GP tasks with PyScripts:
<http://esriurl.com/gpSrvPY>

Distributed Processing

- **Geoprocessing Server provides a framework for distributed processing**
- **One service to do the work (many instances)**
- **Another service to break up the task into pieces, submit to the worker service and re-assemble the pieces into a single output**

Demo – Distributed Geocoding

```
esri.symbol.SimpleLineSymbol({
  color: [0, 0, 255],
  width: 2
});
new dojo.Color([0, 0, 255]);
feature.setSymbol(new esri.symbol.SimpleLineSymbol({
  color: [0, 0, 255],
  width: 2
}));
feature.setSymbol(new dojo.Color([0, 0, 255]));
feature.setSymbol(new esri.symbol.SimpleLineSymbol({
  color: [0, 0, 255],
  width: 2
}));
feature.setSymbol(new dojo.Color([0, 0, 255]));
```

Distributed Geocode Service Characteristics

- **Asynchronous**
- **Inputs : file (csv)**
- **Output: point features**
- **Project Data: locator**

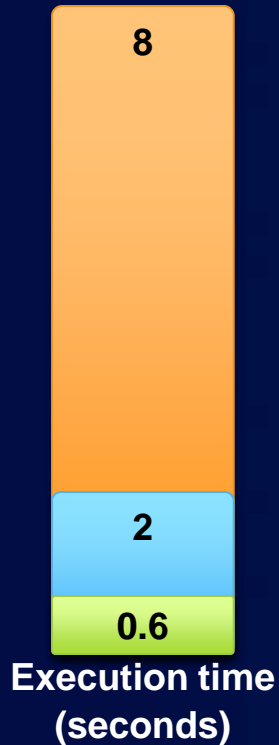
Distributed Processing

- **Submit job**
 - `arcpy.gp.AddToolbox(<service url>;<task name>)`
 - `result = arcpy.gp.<taskname>_<servicename>()`
- **Get Status**
 - `result.GetStatus()`
- **Get Result**
 - `result.GetOutput(<index>)`

Enhancements

- Easier publishing
- Native 64bit
- Dynamic legend in Result Map Service
- Local Jobs Directory
 - No longer have to set this. Its automatically used when server participates in multimachine cluster
- Grid path limit expanded
 - Can now use paths of up to 255 characters when working with grids
- Raster supported **in_memory** workspace
- Multivalue for all supported parameter types
- Better handling of feature sets in the WebAPIs

Performance Improvements (viewshed service)



- **9.3 = 8 seconds**
- **10 = 2 seconds**
 - Spatial Analyst read of geodatabase raster
- **10.1 = 0.6 seconds**
 - 25% faster because of 64bit
 - 25% faster because of layer
 - 25% faster because of in_memory output
 - REST handler faster

Geoprocessing Services in the Web APIs

- **Using Geoprocessing Services with ArcGIS Web Mapping APIs**
- **Wednesday March 28, 4:30 - 5:45**
- **Primerose A**

Publishing Custom GPFunctions

- Can publish custom GPFunctions: .net, c++, java
- Publishing DOES NOT package the dll
- You have to install the dll on the server machine before publishing
- Need to build 64 bit or Any compiler
- Need to register separately for server install
 - Desktop 32 bit
 - Server 64 bit

Questions

- Please fill out your survey
- Questions?



esri