

Please!
Turn **OFF** cell phones
and paging devices



Leveraging the Geoprocessing Framework in ArcGIS Engine in .NET (Best Practices)

Jason Pardy

Corey Tucker

Workshop Outline

- **What is Geoprocessing**
- **Accessing and Running Geoprocessing Tools**
 - Executing tools
 - Accessing licenses and extensions
- **Batch Processing**
- **Geoprocessing Messages**
- **Data properties and access**
 - Working with Geodatabases
- **Build new geoprocessing function tools**
- **SDK (Documentation and Samples)**

Who we think you are:

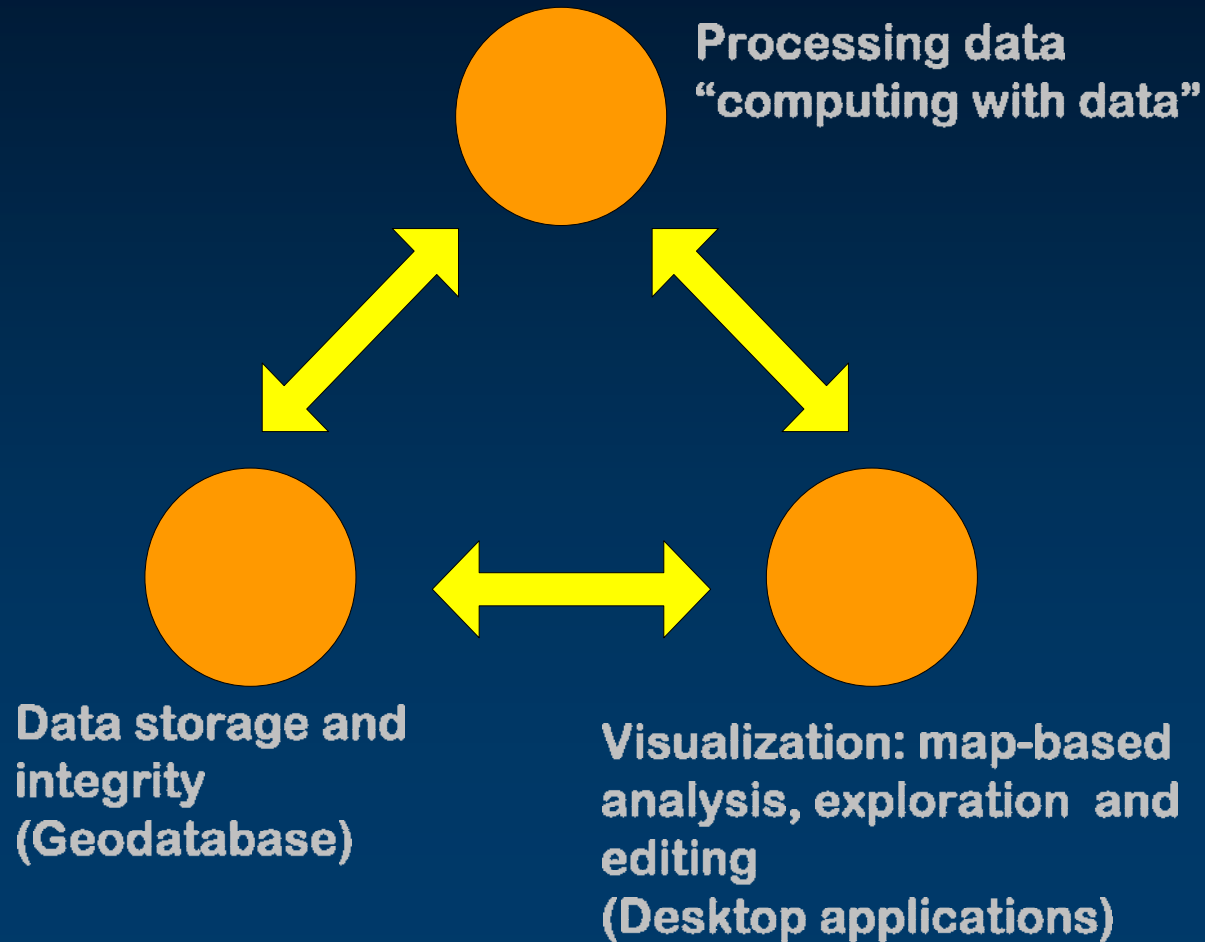
- **You add value to ArcGIS community**
 - Through your understanding of GIS and expertise in a particular domain (wastewater, urban planning, etc.)
- **You develop new tools for ArcGIS**
 - .NET
 - Extend behavior of the system (i.e., new feature class behavior)
 - Extend functionality of the system (new toolbars and tools)

...and why you should know about geoprocessing

- Provides the data analysis, management and conversion tools necessary for all GIS users.
- Empowers GIS professionals to implement workflows
 - Reduces barriers between GIS professionals and software developers
- A complete platform for delivering solutions
 - Universal capability that can be used and deployed by all GIS users to automate their work, build repeatable and well-defined methods and procedures, and to model important geographic processes.
- *Significant* reduction of your development cycle

Geoprocessing is...

...one of the 3 critical components of a GIS



Geoprocessing Tools

- **Take an input or set of inputs and generate one or more outputs (parameters).**
- **Tools are organized into Toolboxes.**
 - Analysis toolbox
 - Conversion toolbox
 - Data Management toolbox
 - Geocoding toolbox
 - Linear Referencing toolbox
 - Coverage toolbox
 - Spatial Statistics toolbox
 - Spatial Analyst toolbox
 - 3D Analyst toolbox

Geoprocessor

- **A Geoprocessing tool is executed by the Geoprocessor.**
- **The Geoprocessor is the main object that simplifies the task of executing geoprocessing tools.**
- **The Geoprocessor contains properties and methods which make it possible to:**
 - **execute tools**
 - **set global environment settings**
 - **examine the resulting messages**
- **It is the single access point for the execution of any geoprocessing tool in ArcGIS, including extensions.**

.NET

- **ArcGIS 9.2 includes a new .NET assembly called ESRI.ArcGIS.Geoprocessor containing a managed class called the *Geoprocessor*.**
 - **This assembly is built against the .NET Framework version 2.0.**
- **Each system geoprocessing toolbox is represented by a managed assembly.**

.NET cont.

- **In each toolbox assembly there are classes representing each geoprocessing tool in the standard ArcGIS geoprocessing toolboxes.**
- **You can use these classes to set up and run geoprocessing tools with the Geoprocessor class.**

.NET – Running a Tool

// Intialize the Geoprocessor

```
Geoprocessor GP = new Geoprocessor();
```

// Set workspace environment

```
GP.SetEnvironmentValue("workspace", @"C:\Newfoundland");
```

// Initialize the Buffer Tool

```
Buffer bufferTool = new Buffer();
```

```
bufferTool.in_features = "roads";
```

```
bufferTool.out_feature_class = "roads_500";
```

```
bufferTool.buffer_Distance_or_field = "500 METERS";
```

// Execute the buffer

```
GP.Execute(bufferTool, null)
```

.NET – Running a Tool By Name

- Can also execute a tool by name.
- No Toolbox Assembly required.

```
// Initialize the Geoprocessor
```

```
Geoprocessor GP = new Geoprocessor();
```

```
// Generate the array of parameters
```

```
IVariantArray parameters = new VarArrayClass();
```

```
parameters.Add(@"C:\newfoundland\roads.shp");
```

```
parameters.Add(@"C:\newfoundland\roads_500.shp");
```

```
parameters.Add("500 METERS");
```

```
// Execute the Model tool by name
```

```
GP.Execute("Buffer_analysis", parameters, null);
```

Running Custom Geoprocessing Tools

- It is also possible to execute custom tools such as model tools and script tools from custom toolboxes.

```
C#  
// Initialize the Geoprocessor  
GP = new Geoprocessor();  
  
// Load the BestPath toolbox to the Geoprocessor  
GP.AddToolbox(@"C:\SanDiego\BestPath.tbx");  
  
// Generate the array of parameters  
IVariantArray parameters = new VarArrayClass();  
parameters.Add(@"C:\SanDiego\source.shp");  
parameters.Add(@"C:\SanDiego\destination.shp");  
parameters.Add(@"C:\SanDiego\bestpath.shp");  
  
// Execute the Model tool by name  
GP.Execute("CalculateBestPath", parameters, null);
```

Running Custom Geoprocessing Tools

- Net users can use the IDE integration framework built in to Visual Studio .NET to generate a geoprocessing assembly to represent any custom toolbox.
- <http://edndoc.esri.com/arcobjects/9.2/NET/462f5942-7928-44d6-b85b-56dc1a0d2ac4.htm>



Working with Tool Names and Avoid Name Conflicts

- When using multiple toolboxes, it is possible that two or more toolboxes will contain a tool with the same name.
- All toolboxes have an Alias property. The alias is a short, alternative name for the toolbox.
- Use the **Execute** method in which you specify the tool name along with the toolbox alias.

```
gp.Execute("Buffer_analysis", parameters, null)
```

ArcObjects as Tool Input

- An ArcObject may be used instead of an ArcCatalog path when defining an input parameter.
 - IFeatureClass, IRasterDataset
- If you are accustomed to working with ArcObjects, you can continue with that object model when working with the Geoprocessor.
- New outputs must be defined by the ArcCatalog path.

```
IFeatureClass inputFC
inputFC = pInputName.Open

bufferTool.in_features = inputFC;
bufferTool.out_feature_class = @"C:\Nfld.gdb\roadsbuffer";
bufferTool.buffer_Distance_or_field = "500 METERS";
GP.Execute(bufferTool, null)
```

Geoprocessing Results

- All geoprocessing tools generate results.
- The Execute method returns an **IGeoProcessorResult** object which manages the results.
 - ESRI.ArcGIS.Geoprocessing namespace
- The result object will have the return value of a tool when executed.
- Return values are necessary when a tool has no output dataset, instead, it has an output scalar value, such as an integer or Boolean.
- The return value is an Object.
 - String, boolean, double, etc.

Geoprocessing Results

Example 1: Return path to output data

```
// The return value is an Object of type string
IGeoProcessorResult pResult = GP.Execute(bufferTool,
null);
object path = pResult.ReturnValue;
```

```
IFeatureClass pFc = GP.Open(path)
```

Example 2: Return the default grid size

```
// The return value is an Object of type double.
IGeoProcessorResult pResult = GP.Execute(calculateGridIndexTool,
null);
object defgrid = pResult.ReturnValue
```

Environment Settings

- **Environments are global parameters for tools.**
- **Script and program writers set the environment and tools use it**
 - **General settings: current workspace, output spatial reference, extent**
 - **Raster analysis settings: cell size, mask**
 - **Coverage settings: derived and new precision, project compare**
 - **More**

```
// Get the Cell Size environment value  
object env = GP.GetEnvironmentValue("cellsize");
```

```
// Set the Cell size environment  
GP.SetEnvironmentValue("CellSize", "50");
```

Licensing and Extensions

- **Whenever a tool is executed in a program, an ArcGIS license is required.**
- **Tools from ArcGIS extensions, such as ArcGIS Spatial Analyst, require an additional license for that extension.**
- **A program must explicitly use `AoInitialize` to check out an extension...**

Checking out an Extension

//Initialize the application

```
IAoInitialize m_AoInitialize = new AoInitializeClass();  
licenseStatus =  
m_AoInitialize.Initialize(esriLicenseProductCode.esriLicenseProductCodeArcE  
ngine);  
licenseStatus =  
m_AoInitialize.CheckOutExtension(esriLicenseExtensionCode.esriLicenseExte  
nsionCodeSpatialAnalyst);
```

// Initialize the Geoprocessor

```
Geoprocessor gp = new Geoprocessor();  
Slope tSlope = new Slope();  
tSlope.in_raster = @"E:\Data\demlatgrd";  
tSlope.out_raster = @"E:\Data\aspect03";  
gp.Execute(tSlope, null);  
licenseStatus =  
m_AoInitialize.CheckInExtension(esriLicenseExtensionCode.esriLicenseExtens  
ionCodeSpatialAnalyst);  
m_AoInitialize.Shutdown(); m_AoInitialize = null;
```

Batch Processing

- **Many geoprocessing tasks are repeated multiple times.**
 - **Example: executing a geoprocessing tool on each feature classes in a geodatabase**
- **Programs are typically used to support batch processing.**

Batch Processing

- **Geoprocessor provides a number of “list” functions:**
 - **ListFeatureClasses (string Wild Card, string Feature Type, string Dataset)**
 - **ListTables (string Wild Card, string Table Type)**
 - **ListDatasets (string Wild Card, string Dataset Type)**
 - **ListRasters (string Wild Card, string Raster Type)**
 - **ListWorkspaces (string Wild Card, string WorkspaceType)**
- **The workspace environment must be set.**
- **The return of each of these methods is an IGPEnumList.**

Batch Processing

```
// List all TIFF files in the workspace and build pyramids
GP.SetEnvironmentValue("workspace", @"C:\Ccm\results");

IGpEnumList rasters = GP.ListRasters("*", "TIFF");
string raster = rasters.Next();

// Intialize the BuildPyramids tool
BuildPyramids pyramids = new BuildPyramids();

while (raster != "") {
    // Set input raster dataset pyramids.
    pyramids.in_raster_dataset = raster;
    GP.Execute(pyramids, null);
    raster = rasters.Next();
}
```

Using Multiple Inputs

- Tools may accept a single input or many inputs, depending on the operation. i.e. Union
- In a program, inputs are passed to these tools as a multivalue string, which uses a semicolon to separate each input within the string.

```
input_features = "soils;landuse;forest"
```

Using Multiple Inputs

- **A value table is a flexible object that may be used as input for a multivalue parameter.**
- **The value table is used to organize the values into a table**
 - **Eliminates need for parsing strings.**
- **A value table can contain many columns. Each column corresponds to a value in the parameter being defined.**
 - **i.e. Union**
- **A value table can be populated with a multivalue string that has been passed to a program as an argument...**

Using Multiple Inputs (ValueTable)

```
IGpValueTableObject vobject = new  
GpValueTableObjectClass();
```

```
// Where input features is a multivalued input featureclasses  
vobject.LoadFromString(inputfeatures);
```

Geoprocessing Messages

- Executing a tool will produce messages. These can be:
 - Informative messages
 - Or warning messages
 - Or error messages
- Messages are retrieved from the Geoprocessing result object.
- The **GetMessages** method will return all messages for the specified severity (0=informative, 1=warning, 2=error)

```
object sev = 2;  
string messages = GPResult.GetMessages(ref sev);  
System.Console.WriteLine(messages);
```

Geoprocessing Messages

- Individual messages can be retrieved using the ***GetMessage*** method.

```
// Execute Union
```

```
IGeoProcessorResult GPRResult = GP.Execute(uniontool, null);
```

```
if (GPRResult.MessageCount > 0) {  
  for (int Count = 0; Count <= GPRResult.MessageCount - 1; Count++)  
  {  
    Console.WriteLine(GPRResult.GetMessage(Count));  
  }  
}
```

Describing Data

- The Geoprocessor's *GetDataElement* method can be used to describe data.
- Returns an IDataElement object with relevant properties based on type of data being described.
- Allow script or program to determine properties of data
 - Spatial reference
 - Extent of features
 - List of fields
 - ShapeType (point, polygon, etc)
 - Data type (coverage, shapefile, etc)
- Logic can be added to the program to branch (if statement) based on those properties.

Describing Data

```
// Describe the input featureclass
```

```
IDataElement de = GP.GetDataElement(@"C:\Portland.gdb\streets", ref dt)
```

```
// Open the featureclass dataelement and get the shapetype
```

```
IDFeatureClass defc = de as IDFeatureClass;
```

```
if (defc.ShapeType == esriGeometryType.esriGeometryPolyline)
```

```
    Console.WriteLine("ShapeType is polyline.");
```

ArcGIS 9.2 - Executing Geoprocessing Server Tools

- **With ArcGIS 9.2, you can now execute geoprocessing tools that have been published on an ArcGIS server.**
- **Server tools can be accessed and executed the same as custom tools.**
- **Toolboxes can be published on a local area network (LAN) or published as a web service on the internet.**
 - **To access the geoprocessing server tools you must add the toolbox.**

Executing Server Tools

- **Tools may be executed by reference.**
 - **The input parameters may reference layers in a map document**

```
Geoprocessor GP = new Geoprocessor();  
  
// Add the BestPath toolbox  
Gp.AddToolbox("http://flame7/arcgis/services;GP/Bestpathtoolbox");  
  
// Inputs reference layers in a map document on the server.  
IVariantArray parameters = new VarArrayClass();  
parameters.Add("source");  
parameters.Add("destination ");  
  
// Execute the server tool by reference  
IGeoProcessorResult result;  
result = GP.Execute("CalculateBestPath", parameters, null);
```

Executing Server Tools

- **Tools may be executed by value (enter inputs)**
 - The input parameters require you to enter the input parameter values by specifying the path to the data.

```
GP = new Geoprocessor();
```

```
// Add the BestPath toolbox
```

```
Gp.AddToolbox("http://flame7/arcgis/services;GP/Bestpathtoolbox");
```

```
// Input values are data on a local drive.
```

```
IVariantArray parameters = new VarArrayClass();
```

```
parameters.Add("C:\sandiego\source.shp");
```

```
parameters.Add("C:\sandiego\destination.shp");
```

```
// Execute the server tool by reference
```

```
IGeoProcessorResult result;
```

```
result = GP.Execute("CalculateBestPath", parameters, null);
```

Executing Geoprocessing Server Tools - Results

- **The GeoprocessorResult object is necessary for supporting geoprocessing with ArcGIS server.**
- **This result object will contain:**
 - **return value of a tool when executed**
 - **status of a job on the server**
 - **result id**
 - **geoprocessing messages**

Executing Geoprocessing Server Tools - Results

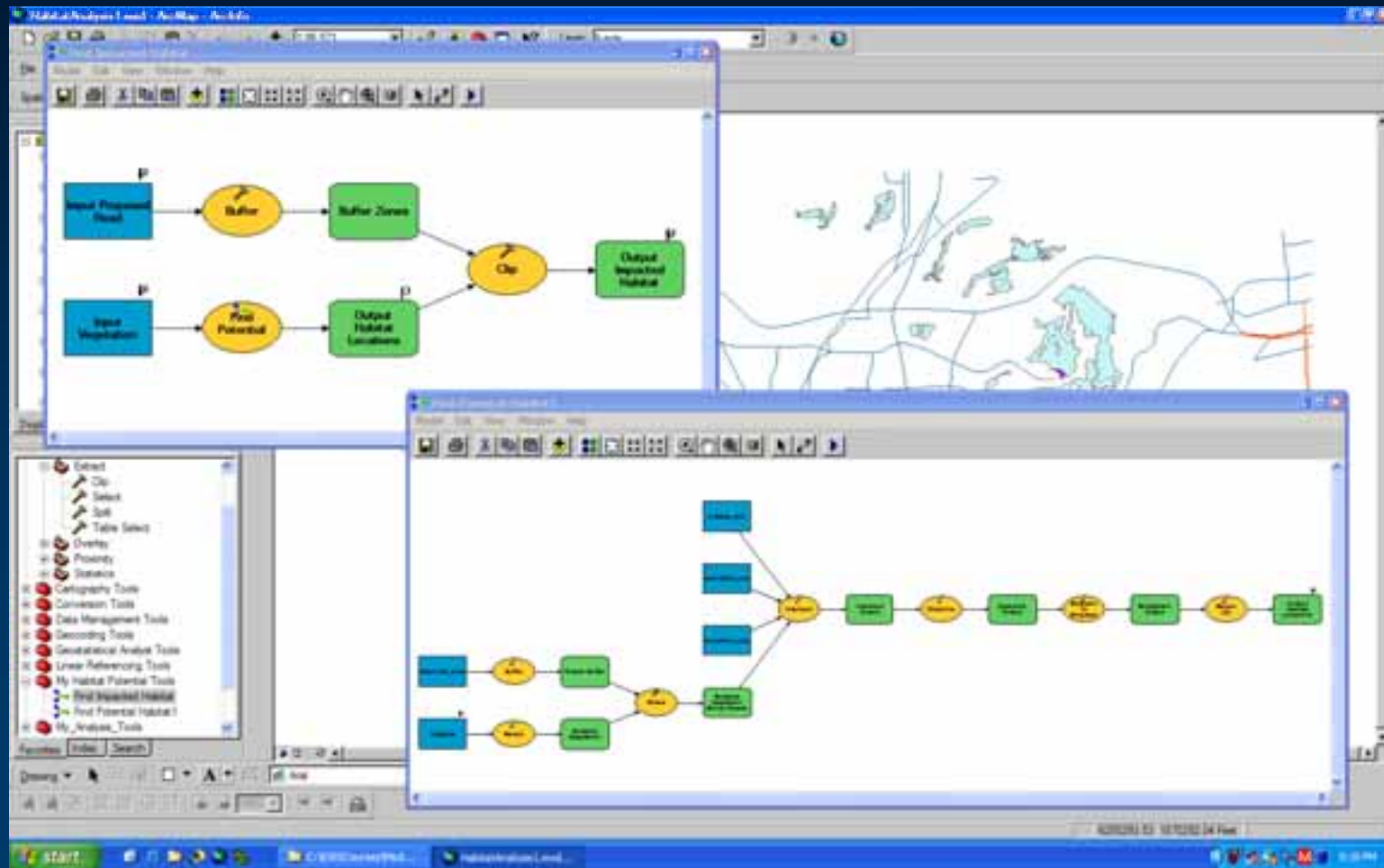
```
// Execute the server tool by reference  
IGeoProcessorResult result;  
result = GP.Execute("CalculateBestPath", parameters, null);  
  
while (result.Status != esriJobStatus.esriJobSucceeded)  
{  
  for (int Count = 0; Count <= result.MessageCount - 1; Count++)  
  {  
    Console.WriteLine(result.GetMessage(Count));  
  }  
}
```

Creating Tools

- **Model Tool**
 - FeatureClassToFeatureClass Tool
- **Script Tool**
 - FeatureClassToGeodatabase Tool
 - Spatial Statistics Tools
- **Function Tool (System Tool)**
 - Most Geoprocessing Tools are Function Tools
- **Custom Tool**
 - i.e. Data Interoperability Extension

Developing Models - ModelBuilder

Model Tools

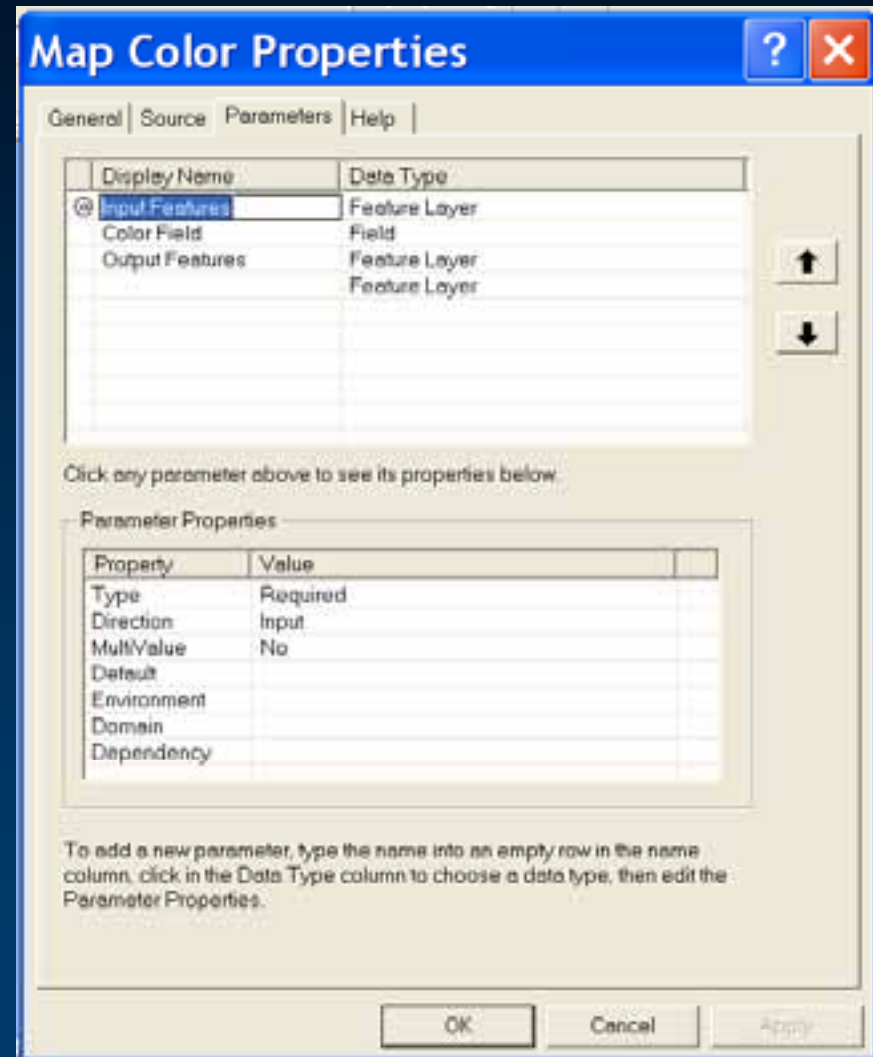


Developing Script Tools

- **Executables may be added to toolboxes as a script tool**
 - Program executables (exe) can be used as source to script tools
- **Once added as a tool, the program behaves like other tools, it can be run using a dialog, in ModelBuilder, at the command line, or in another script/program.**

Scripts and Executables as Tools

- **Script tool source is pathname to .exe file.**
- **Define input and output parameters**



Building Geoprocessing Function Tools

- Most ESRI Geoprocessing tools are implemented as COM function tools.
- Requires (a minimum) of implementing two interfaces:
 - *IGPFunction*
 - *IGPFunctionFactory*
- Many tools can be included in a single DLL.
- Technical Document:
 - Developer Help
 - <http://edndoc.esri.com/arcobjects/9.2/NET/e7d06ae9-a6d1-4248-a7a3-9d5f375f088c.htm>

Developer Help for ArcGIS 9.2

- **New Help section in the development kits – Getting started with Geoprocessing**
 - Using Geoprocessing Tools
 - Geoprocessing Tool Help
 - Geoprocessing Concepts
 - Samples
 - Building Geoprocessing Custom Tools
- http://edndoc.esri.com/arcobjects/9.2/NET/shared/geoprocessing/geoprocessing/what_is_geoprocessing_qst.htm

Summary

- **Full Geoprocessing support available with ArcGIS Engine 9.2.**
- **Geoprocessor is the single access point for the execution of any geoprocessing tool in ArcGIS.**
 - **Tool Execution is simple**
- **Run custom tools such as models and scripts developed by desktop.**
- **List methods to help easily do batch processing.**
- **Run Geoprocessing tools on an ArcGIS Server.**
- **Build new functions tools.**
- **Enhanced Geoprocessing Developer Help.**

Geoprocessing Resources

- General

- <http://support.esri.com/...>

- Geoprocessing
- Knowledgebase
- Downloads → Geoprocessing: Scripts, Models, Documentation
- User Forum → Geoprocessing ArcToolbox / ~ ModelBuilder / ~ Scripting

- <http://edn.esri.com/...>

- Search for “geoprocessing”

- <http://arcscripts.esri.com/...>

- Ex: Select “Python”, “ArcGIS Desktop”, type “geoprocessing”

NEW: ArcGIS 9.2 WebHelp

<http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=welcome>

- Geoprocessing Book
- Geoprocessing Tool Reference Book

Microsoft Keynote

Wednesday 6:00pm

Oasis 2

Eddie Amos

Senior Director, Developer & Platform Evangelism

**Thank-you. Fill out survey.
Questions?**