



## **A GP Service for Enterprise Printing**

Kevin Shows, Anadarko Petroleum  
Kirk Kuykendall, Idea Integration  
04/20/2011

# A GP Service for Enterprise Printing

## History

- Implemented ArcGIS Server / browser mapping in 2008
  - Web ADF - December, 2008
  - WPF - April, 2010
  - Internally branded “iMaps”
- Target audience: non-GIS users
  - Simple, fast, intuitive
  - No installation
  - “If we need user training we’ve done something wrong”

# A GP Service for Enterprise Printing

## History

- Immediate request: “how can I get a paper copy of the map”
  - User defined title box, scale and DPI
  - Legend with symbols
  - Support large size plotting

# A GP Service for Enterprise Printing

## Challenges

- Wide variety of data sources
  - Internal map services (cached and dynamic)
  - Internal image services (Image Extension)
  - External tiled services (Bing, ArcGIS Online)
  - External WMS (I3, weather, buoy information, etc)
  - User created annotation and user uploaded data

# A GP Service for Enterprise Printing

## Challenges

- Legends
  - May overrun page size
  - User needs to select which layers will appear
- Must support large plot size
  - 8.5x11 through 34x44, portrait and landscape
  - At up to 600 dpi
- Must be simple and WYSIWYG

# A GP Service for Enterprise Printing

## Solution: Web Services

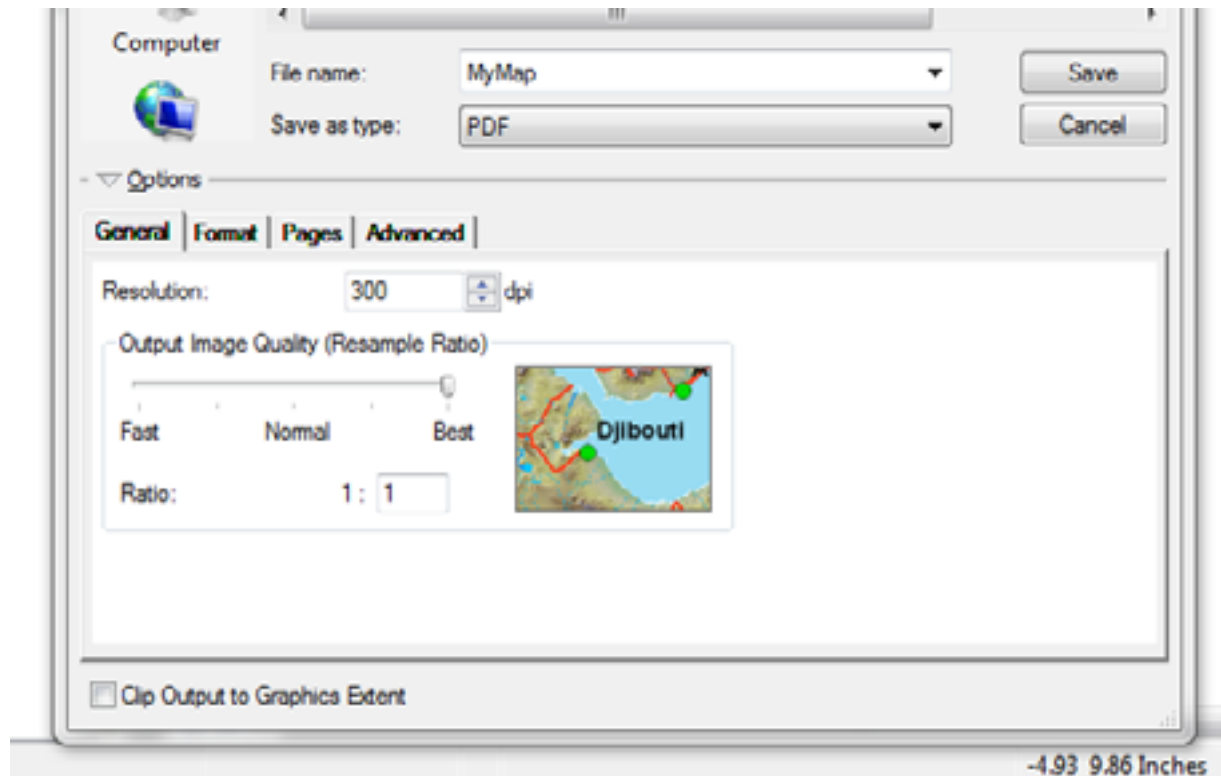
- GP Service that
  - dynamically creates MXD from ArcMap template
  - Populates TOC sources based on web map layers
  - Lyr files containing all the data source in the web browser
  - Use ArcMAP templates
    - ▶ Control paper size and object placement
    - ▶ Controls dynamic values (user name, date, title)
- Use ArcGIS Desktop export to PDF functionality
  - Upscale browser map data sources to appropriate PDF size

# A GP Service for Enterprise Printing

Simple, right?

# A GP Service for Enterprise Printing

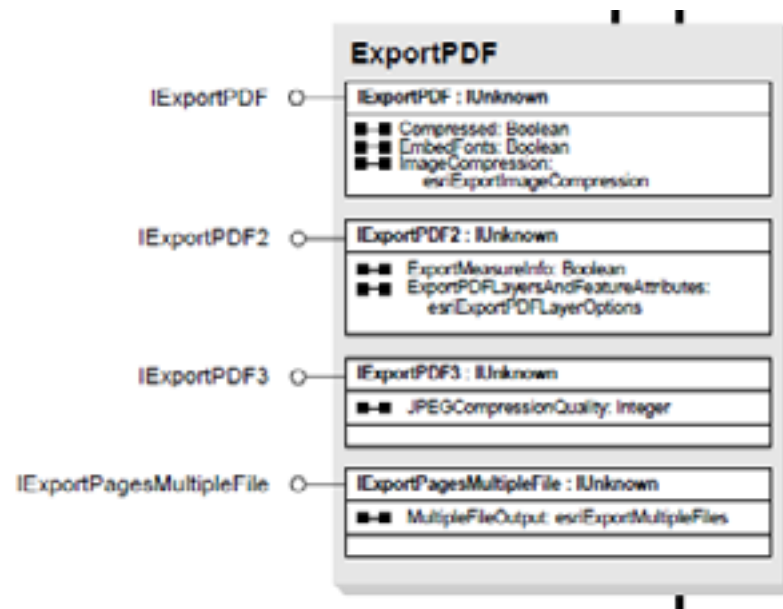
When we export with Arcmap ...





# A GP Service for Enterprise Printing

Behind the scenes ArcMap is using ArcObjects ...



# A GP Service for Enterprise Printing

This means we can write code to do what ArcMap does.

```
IExport exporter = null;
if(outfilePath.ToUpper().EndsWith("PDF"))
    exporter = new ExportPDFClass() as IExport;
else if (outfilePath.ToUpper().EndsWith("JPG"))
    exporter = new ExportJPEGClass() as IExport;

if (exporter is IExportPDF)
{
    ((IOutputRasterSettings)exporter).ResampleRatio = 1;
    SetPdfOptions(dtoExporter, exporter, messages);
}
```

# Pop Quiz

Q. how can we expose a long running ArcObjects function to web clients?

# A GP Service for Enterprise Printing

## Pop Quiz

Q. how can we expose a long running ArcObjects function to web clients?

A. as an asynchronous Geoprocessing (GP) Service

# A GP Service for Enterprise Printing

## Pop Quiz

Q. how can we expose a long running ArcObjects function to web clients?

A. as an asynchronous Geoprocessing (GP) Service

Which means we need to wrap our ArcObjects code with the appropriate GP interfaces...

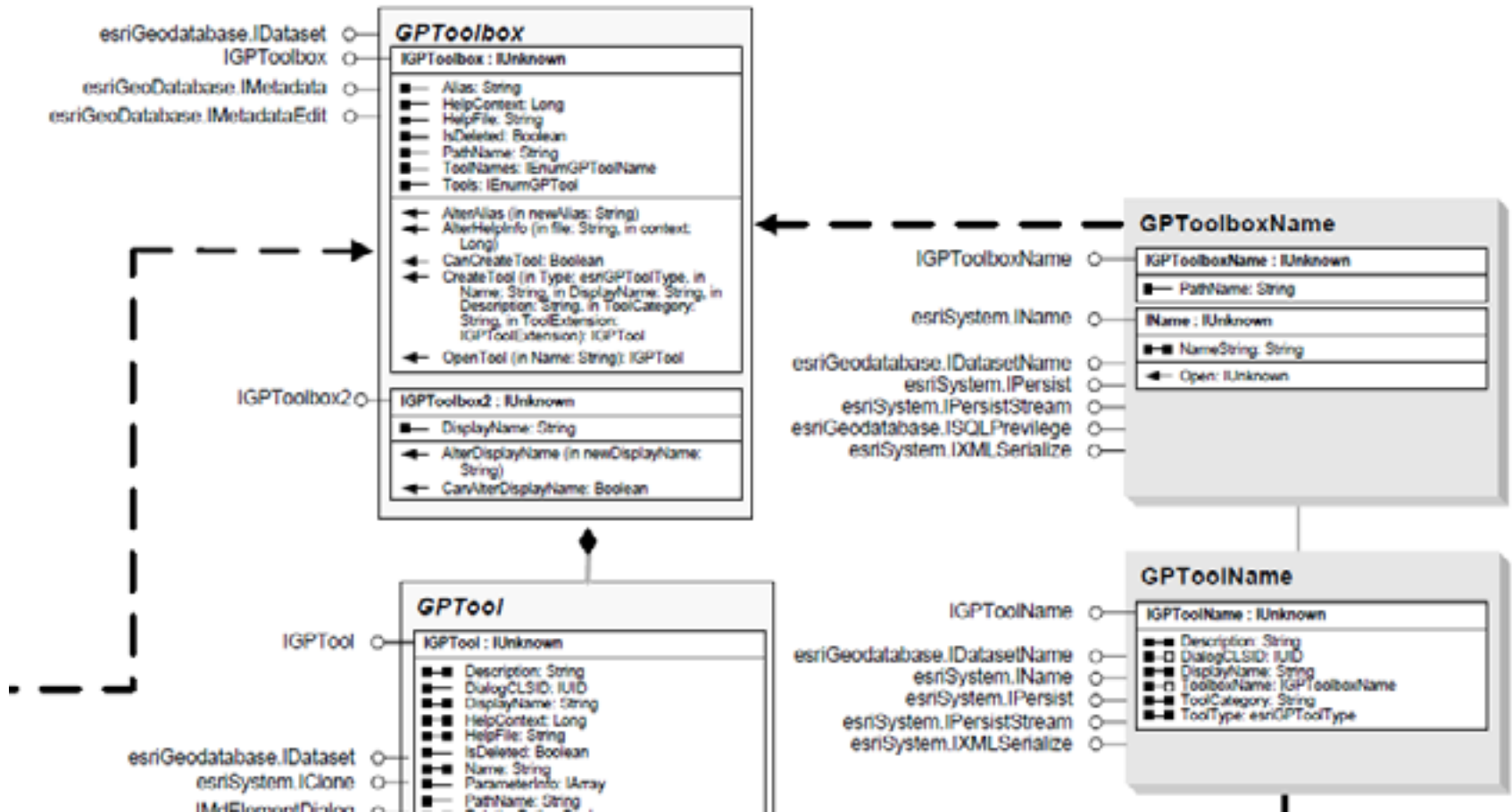
# A GP Service for Enterprise Printing

Using code provided by Esri as a starting point, Anadarko and Idea wrapped the ArcObjects export functions, by implementing:

- IGPFFunction2
- IGPFFunctionFactory

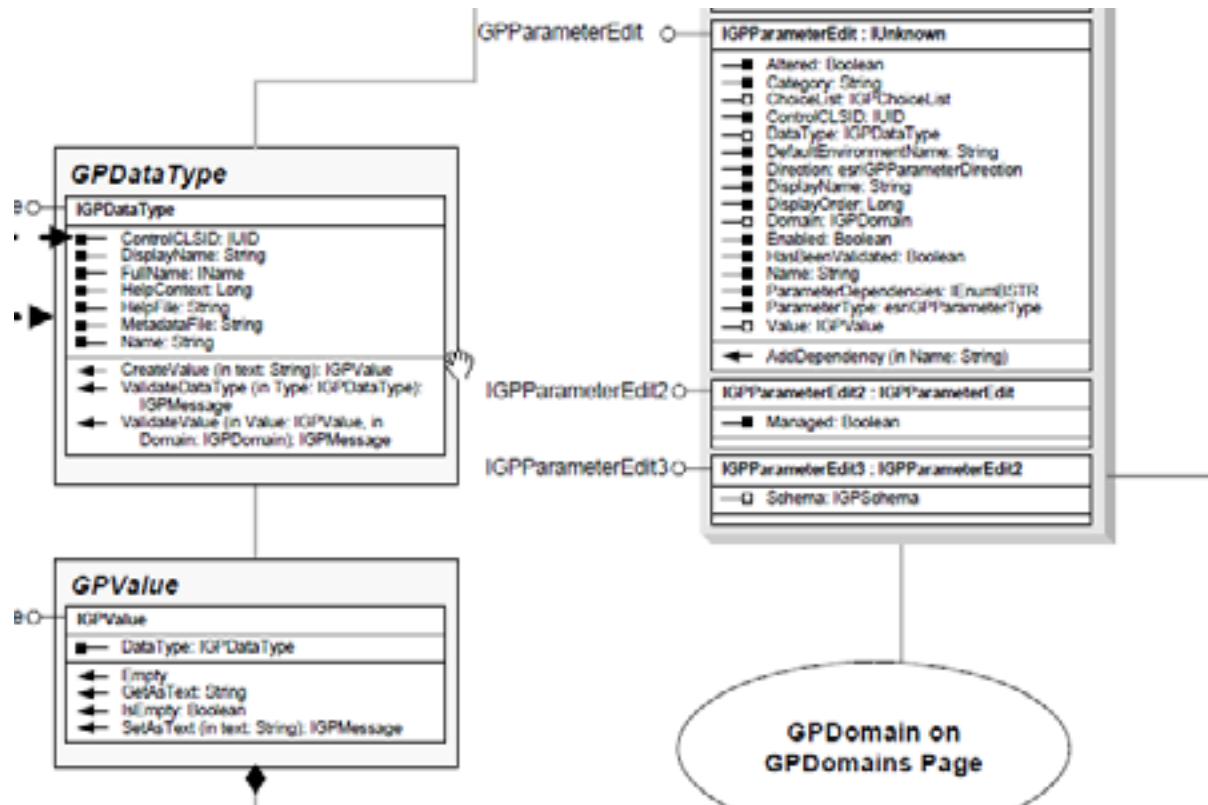
We re-factored along the way so we could always test it from a console test app.

# A GP Service for Enterprise Printing



# A GP Service for Enterprise Printing

Each argument to the GP tool needs code to handle it, which can turn into a maintenance headache as requirements evolve.





# A GP Service for Enterprise Printing

To overcome this we used the “Parameter Object” design pattern, and passed a single input string argument.



## Parameter Object

[DesignPattern](#)

**Name:** [ParameterObject](#)

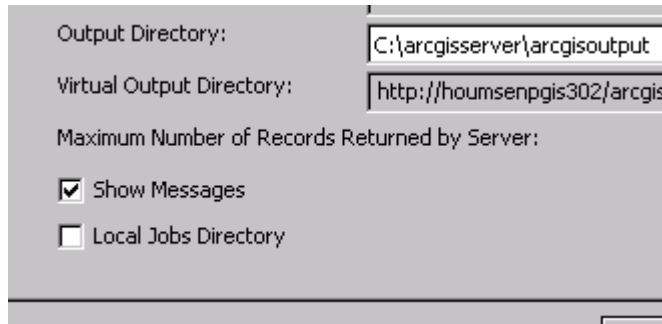
**Type:** Design

**Problem:** An API call gets huge, with [TooManyParameters](#) and/or the need to change the signature frequently during development. Most invocations use default arguments. Function overloading is unavailable or considered undesirable. Or, it is desired to expand an interface without breaking existing code. Or, you need to return multiple values and don't have reference parameters available.

## A GP Service for Enterprise Printing

To log to a file, we implement IGPMessages that contains the messages passed to IGPFfunctionExecute.

That way we can write to a log file, then also write to the contained IGPMessages object. The GP Service sends those messages back to the web client.



Output Directory: C:\arcgisserver\arcgisoutput

Virtual Output Directory: http://houmsenpgis302/arcgis

Maximum Number of Records Returned by Server:

Show Messages

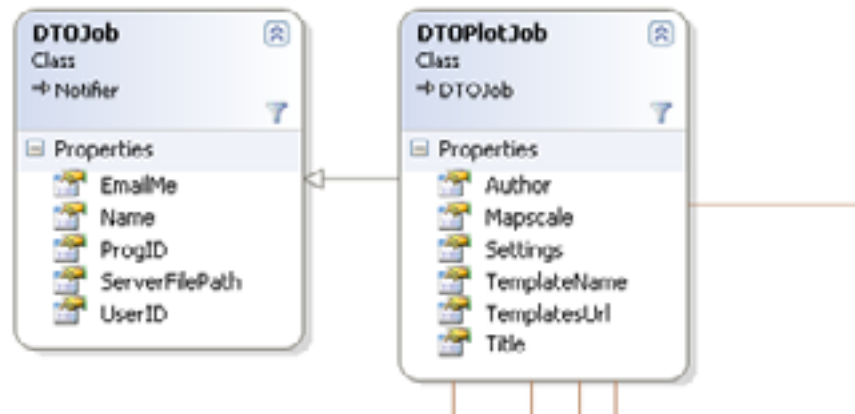
Local Jobs Directory

# A GP Service for Enterprise Printing

```
4/10/2011 2:25:04 PM inform: processID = 73092
4/10/2011 2:25:04 PM inform: MachineName = HOUMSENPGIS005
4/10/2011 2:25:04 PM inform: Version = 1.0.0.0
4/10/2011 2:25:04 PM inform: adding layer Street Map
4/10/2011 2:25:05 PM inform: adding layer Global APC Wells
4/10/2011 2:25:05 PM inform: adding layer Wells
4/10/2011 2:25:06 PM inform: adding layer Polygons
```

# A GP Service for Enterprise Printing

The Object passed to the GP Export is called a Job, serialized as a Json string.

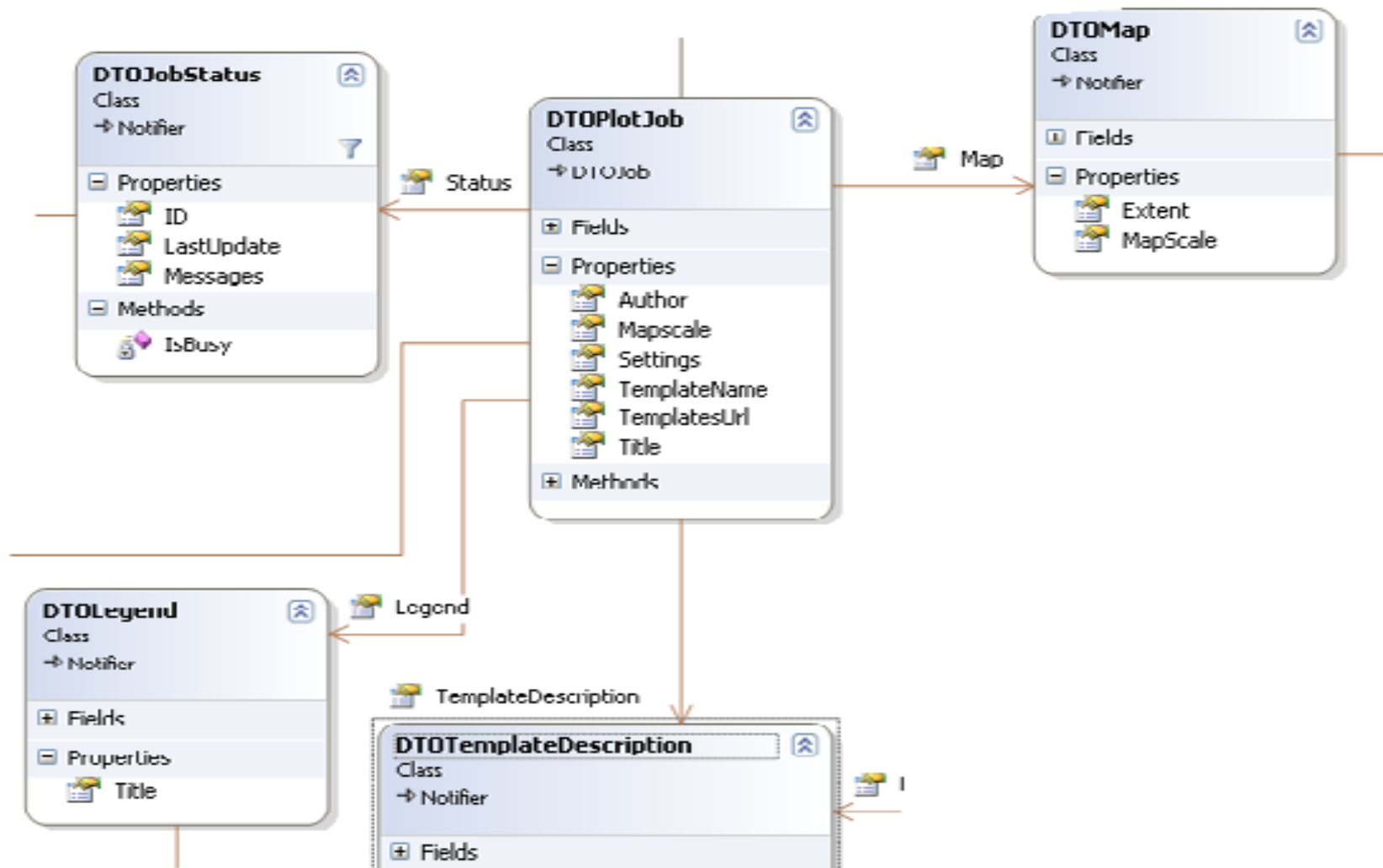


# A GP Service for Enterprise Printing

The Job Object needs to

- play nicely with the UI (INotifyPropertyChanged)
- Travel over http (Serializeable as json)

# A GP Service for Enterprise Printing



# A GP Service for Enterprise Printing

How do we handle Impatient Users?

- High Res jobs take a long time to complete
- If user logs out, Esri REST API provides no mechanism to find completed jobs from an earlier session.

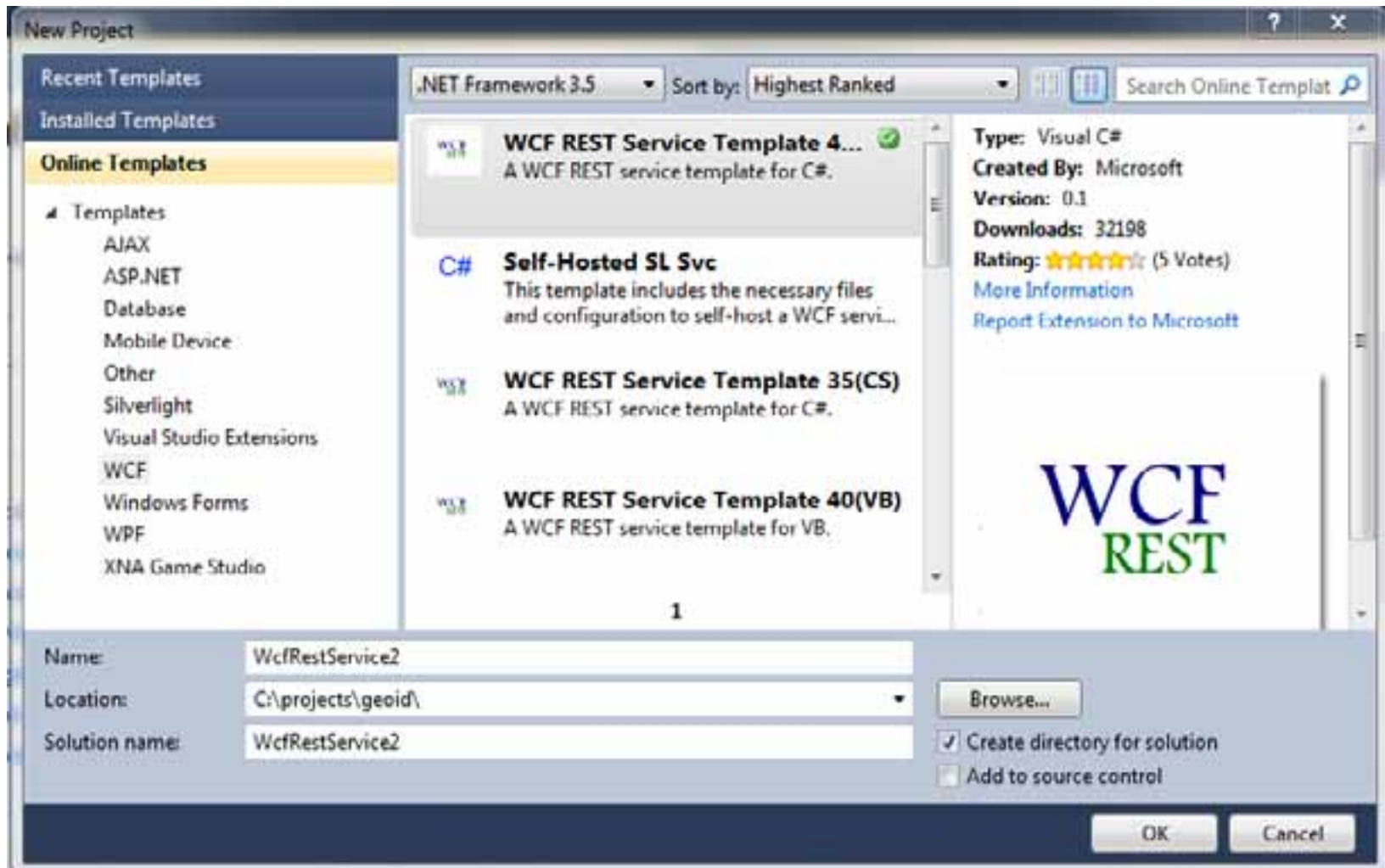
# A GP Service for Enterprise Printing

Provide a Job Manager Service WCF REST service

- Based on online Visual Studio REST project template
  - GET list of jobs for user
  - PUT a new job
  - DELETE a job
- Behind the scenes, jobs stored as text files (json)

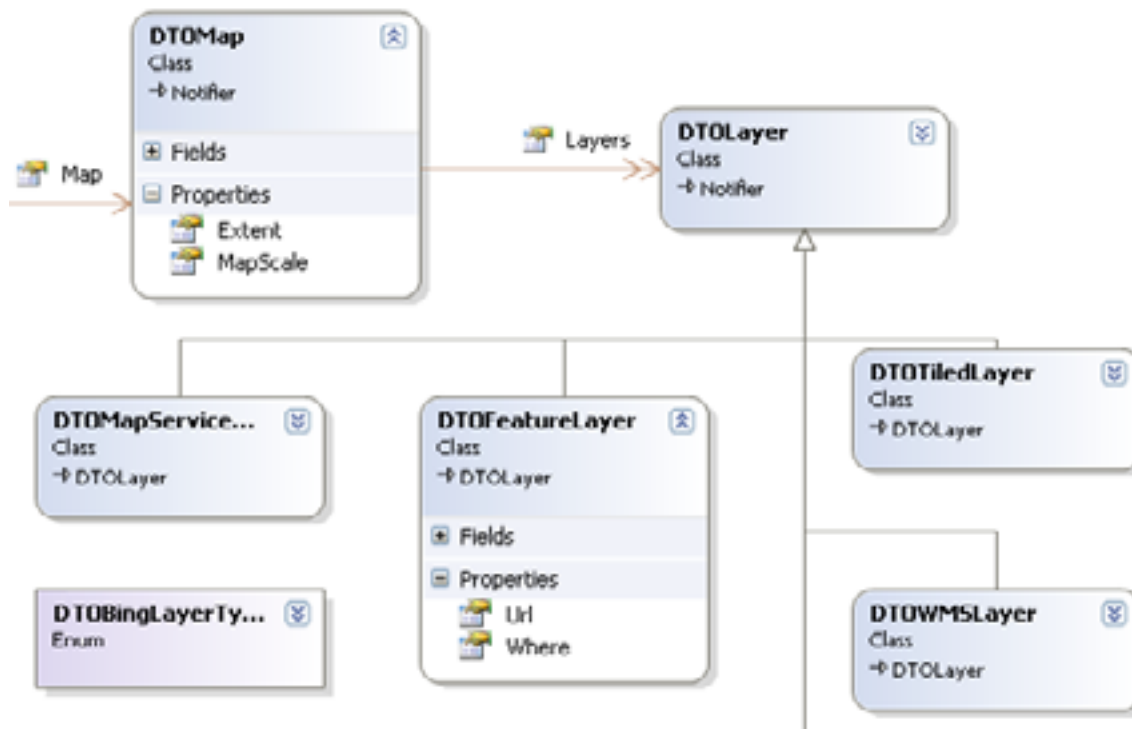


# A GP Service for Enterprise Printing



# A GP Service for Enterprise Printing

We couldn't find a way to serialize Abstract classes (like Layer) using Microsoft's serializer, Json.NET used instead.



# A GP Service for Enterprise Printing

## Life Cycle of a Job (Web App)

- The Web Job Tool
  - Creates a new Job in memory
  - Sets properties using a Wizard
  - Submits Job to Job Manager

# A GP Service for Enterprise Printing

## Life Cycle of a Job (GPTool)

- The GPTool
  - Deserializes json into Job object
  - Creates an Mxd
  - Exports the mxd

This breaking this into two separate steps, we keep open the possibility of changing the mxd, then re-exporting. (To keep it simple re-submittal of jobs was removed).

# A GP Service for Enterprise Printing

## Mxd Creation (the details)

- Deserialize Job from json
- report progress back to log file in job folder
- While this is happening, web client polls job, getting new logfile messages, and status.
- copy template Url to mxd
- add layers as described in Job
- add legend items
- Populate text box elements using name/value pairs
- Save mxd

# A GP Service for Enterprise Printing

## Export Mxd to Pdf (the details)

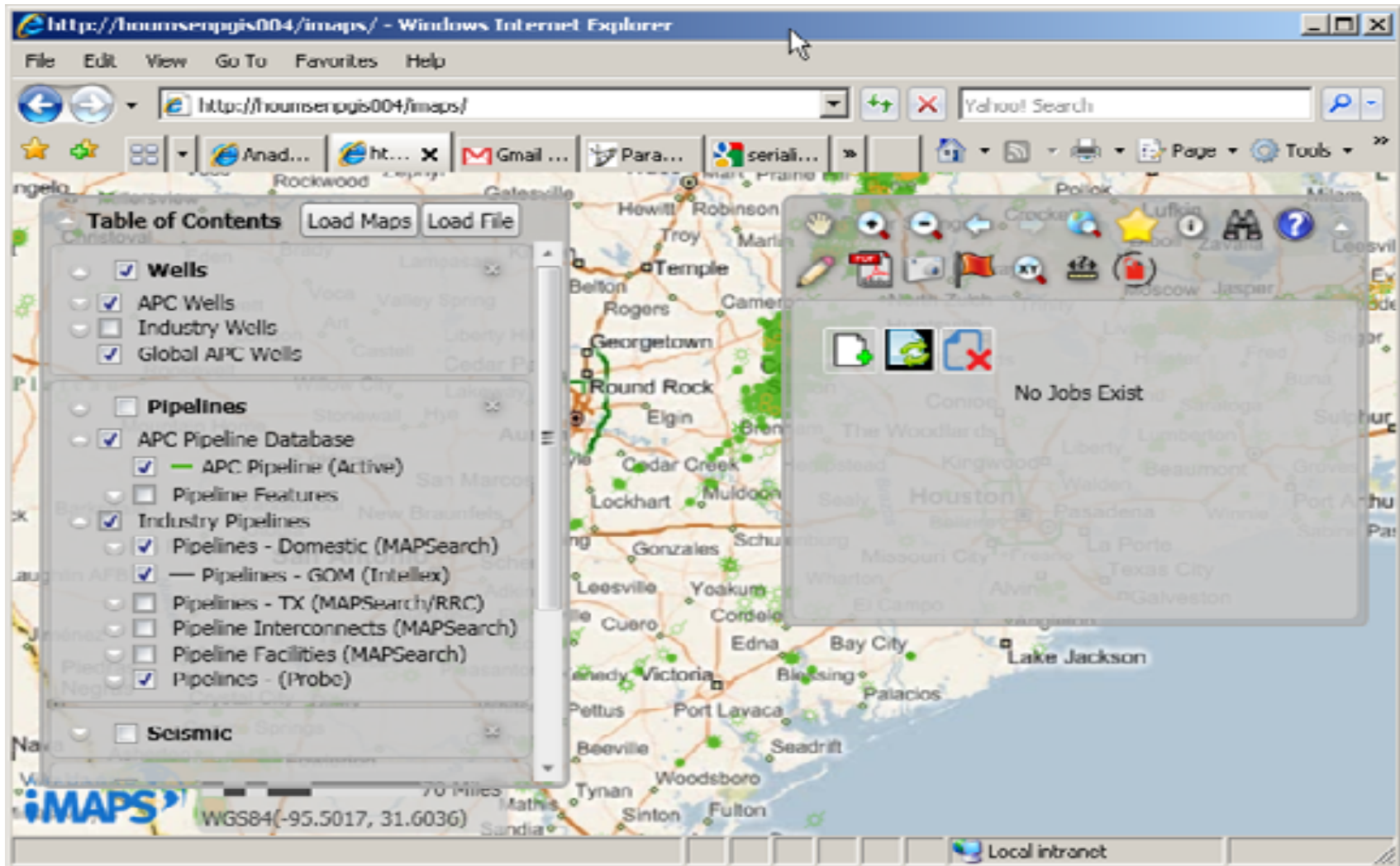
- Open mxd
- Cull legend items not visible (using SOE)
- Export layout to low res jpg
- Export layout to pdf at specified resolution
- Email user with UNC to pdf and jpeg

# A GP Service for Enterprise Printing

## Job Cleanup

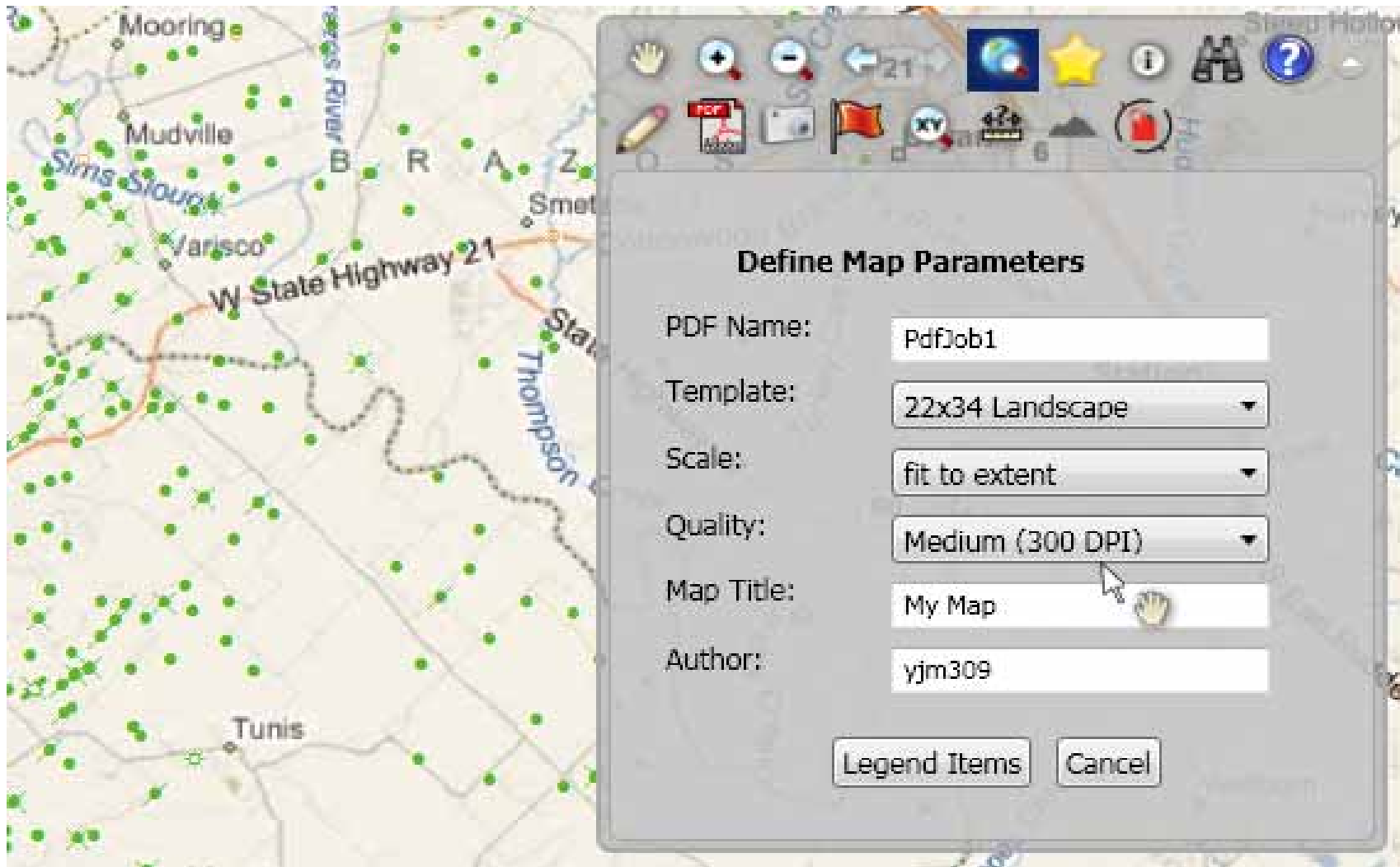
- Users can delete jobs from UI
- Windows service deletes jobs older than 60 days.
- Annoyance: Esri REST lacks method to cancel job (SOAP only).
- Users with Arcmap can open mxd in any of their job folders.

# A GP Service for Enterprise Printing





# A GP Service for Enterprise Printing



# A GP Service for Enterprise Printing

**Select Layers to Appear in Legend**

**Available Layers**

- ▲ Wells
  - ▶ APC Wells
  - ▲ Industry Wells
    - Wells - Domestic
    - Wells - Domestic (srfc)
    - Wells - Domestic (srvy)
    - Wells - International

>>

<<

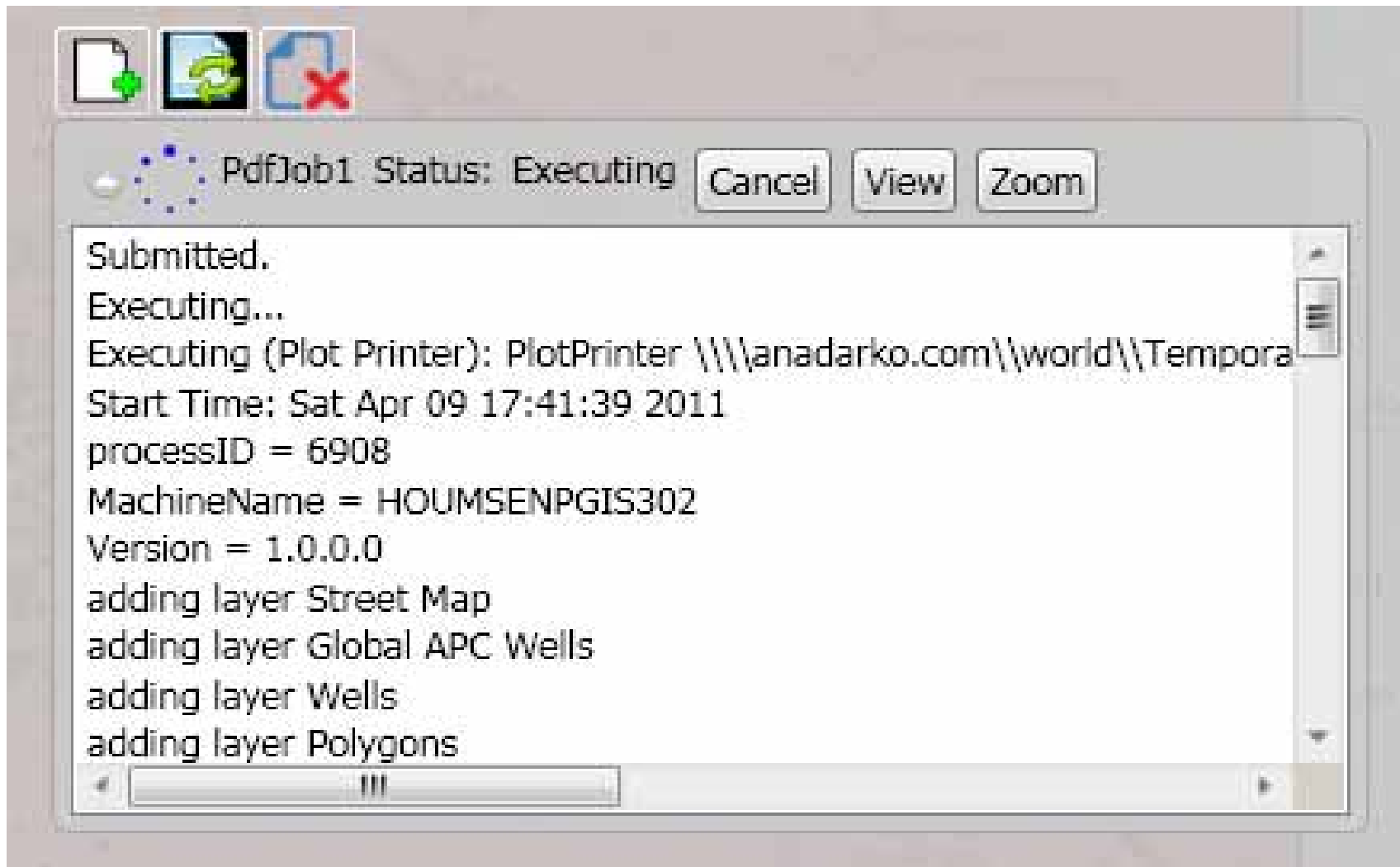
**Legend Items**

Name	Cull*
Wells - Domestic	<input checked="" type="checkbox"/>

*\*Culling lets you remove symbols from the legend that do not fall within the visible mapextent.*

Previous
Finish
Cancel

# A GP Service for Enterprise Printing



# A GP Service for Enterprise Printing

## Pdf Job PdfJob1 completed

● JobManager\_no\_reply@anadarko.com

**To:** Kuykendall, Kirk

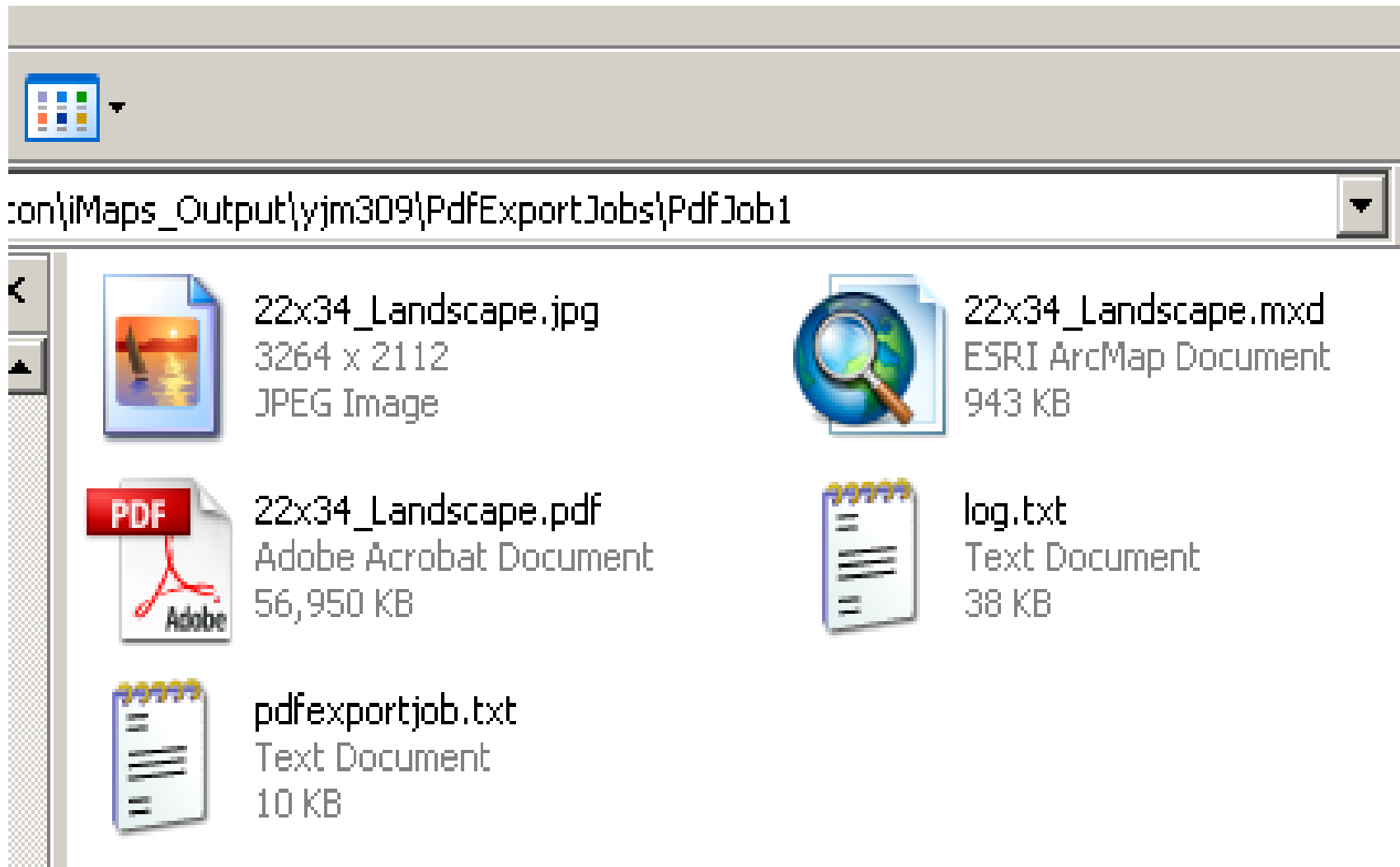
---

Here is your pdf: [file:\\anadarko.com\world\Temporary\(60Days\)\Houston\iMaps Output\yjm309\PdfExportJobs\PdfJob1\22x34 Landscape.pdf](file:\\anadarko.com\world\Temporary(60Days)\Houston\iMaps Output\yjm309\PdfExportJobs\PdfJob1\22x34 Landscape.pdf)

Jpeg preview: [file:\\anadarko.com\world\Temporary\(60Days\)\Houston\iMaps Output\yjm309\PdfExportJobs\PdfJob1\22x34 Landscape.jpg](file:\\anadarko.com\world\Temporary(60Days)\Houston\iMaps Output\yjm309\PdfExportJobs\PdfJob1\22x34 Landscape.jpg)

Please conserve paper by reviewing before printing.

# A GP Service for Enterprise Printing



# A GP Service for Enterprise Printing

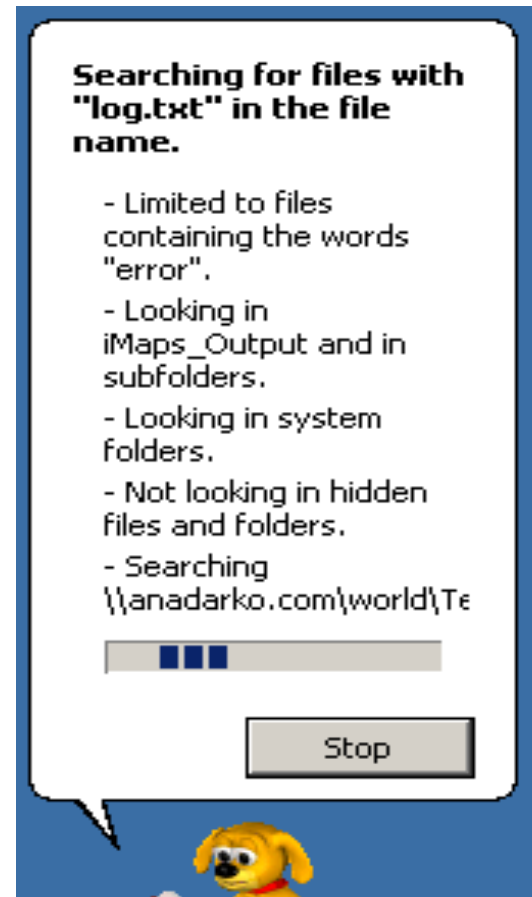
```

pdfexportjob.txt
1 {
2   "$type": "Anadarko.iMaps.DataTransfer.DTOPlotJob, Anadarko.iM
3   "Title": "My Map",
4   "TemplatesUrl": "http://houmsenpgis302/JobManager/JobService"
5   "TemplateDescription": {
6     "$type": "Anadarko.iMaps.DataTransfer.DTOTemplateDescriptio
7     "Filepath": "\\housenpgis302\\PdfExporterTemplates\\APC
8     "Width": 34.0,
9     "Height": 22.0,
10    "Legends": [
11      {
12        "$type": "Anadarko.iMaps.DataTransfer.DTOLegendDescript

```

## A GP Service for Enterprise Printing

The administrator can also search for jobs that had errors:



# A GP Service for Enterprise Printing

Q. How do we expose standard template mxd's to web clients?



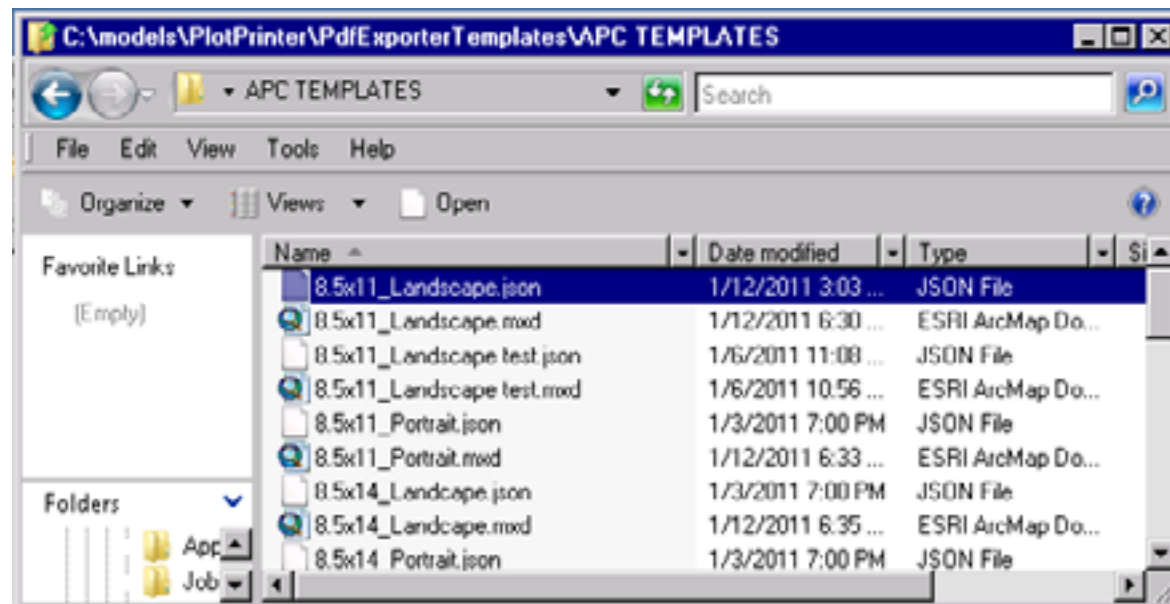
# A GP Service for Enterprise Printing

Q. How do we expose standard template mxd's to web clients?

A. As a REST Endpoint.

# A GP Service for Enterprise Printing

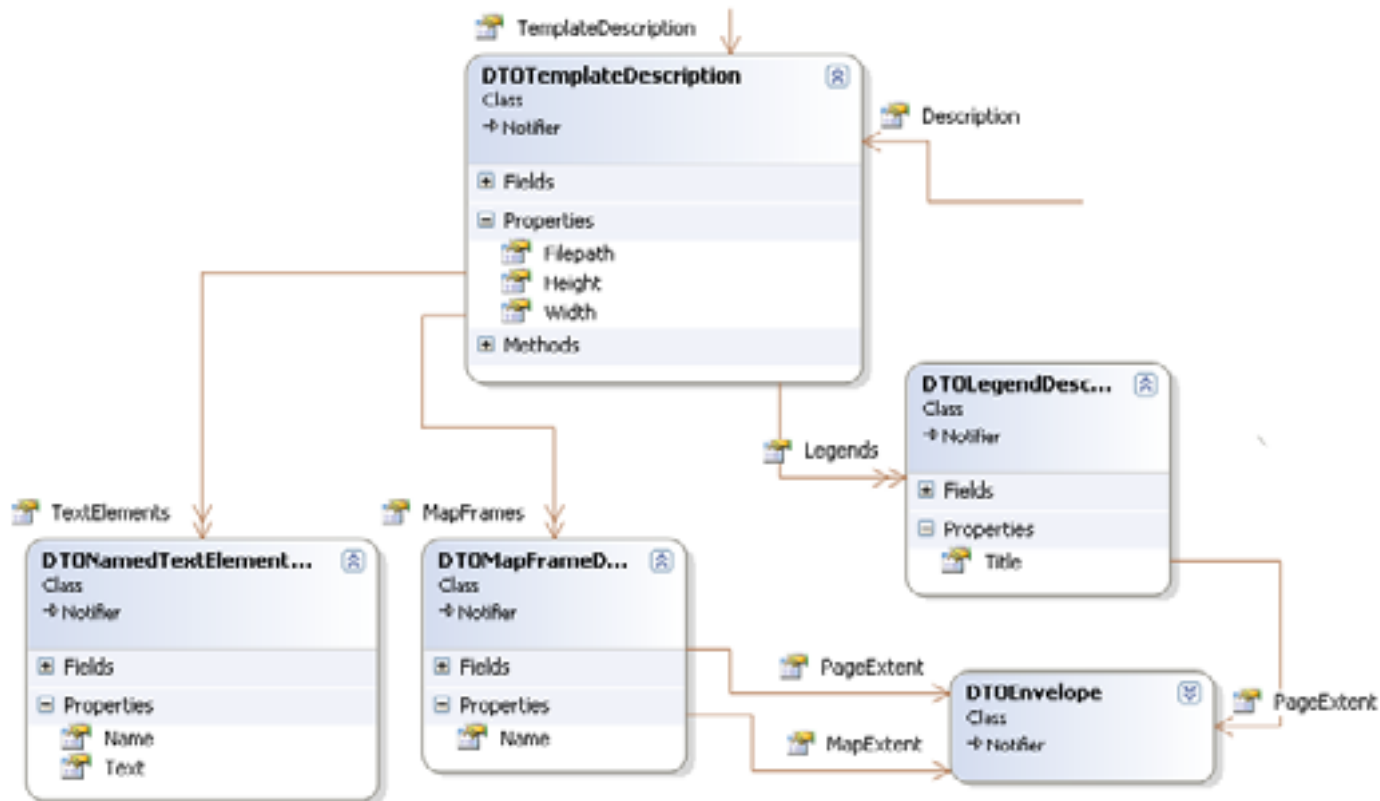
Also provide a process that dumps json representations of map templates mxd's as json.



# A GP Service for Enterprise Printing

The templates can then be retrieved using a url.

<http://gisserver/JobManager/JobService/MxdTemplates>



# A GP Service for Enterprise Printing

A *Layout Description* is derived from an mxd and contains:

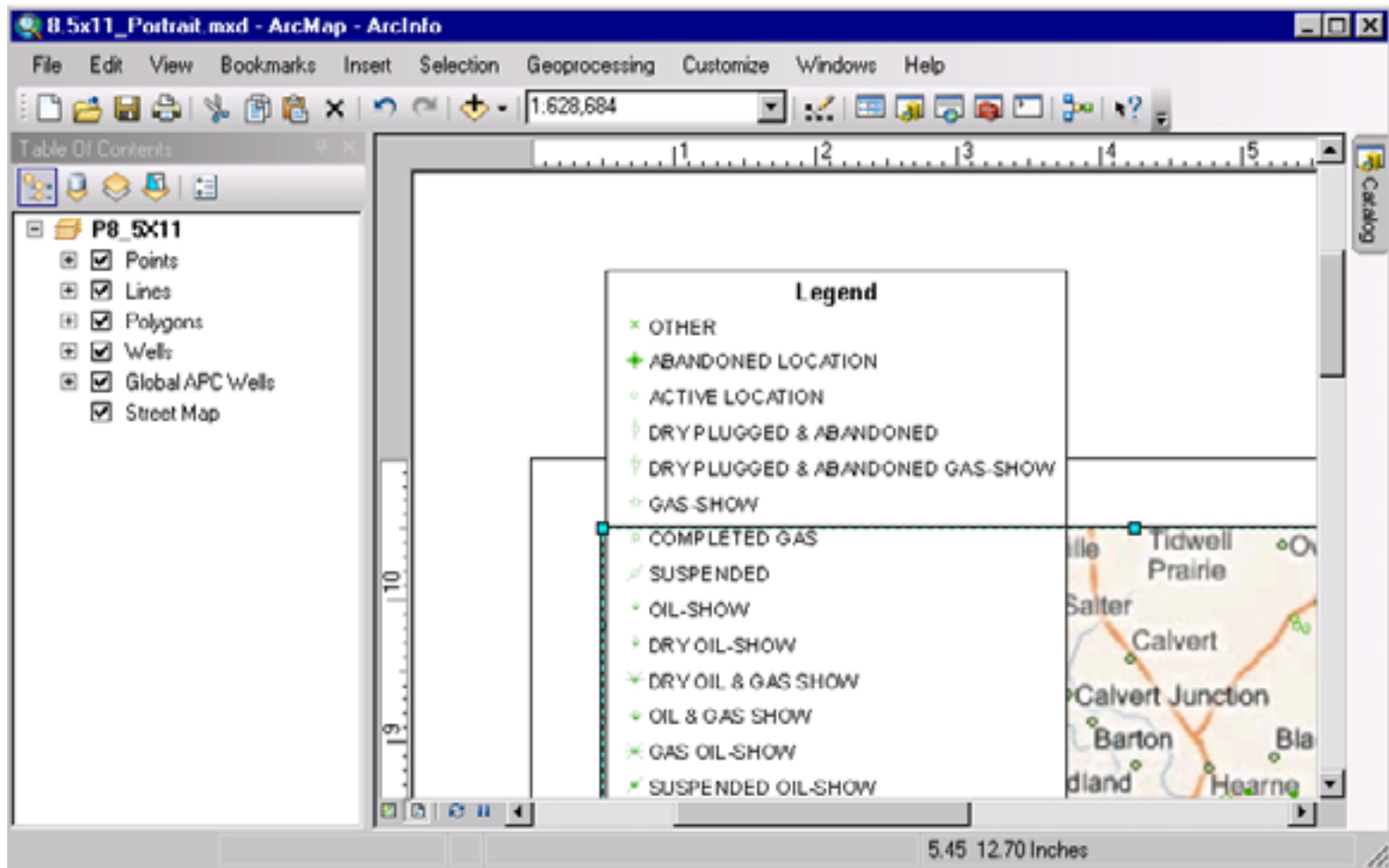
- Text element names (for dynamic text replacement)
- Legend size and position (if any)
- Size and position of Map Frame(s)

It is also:

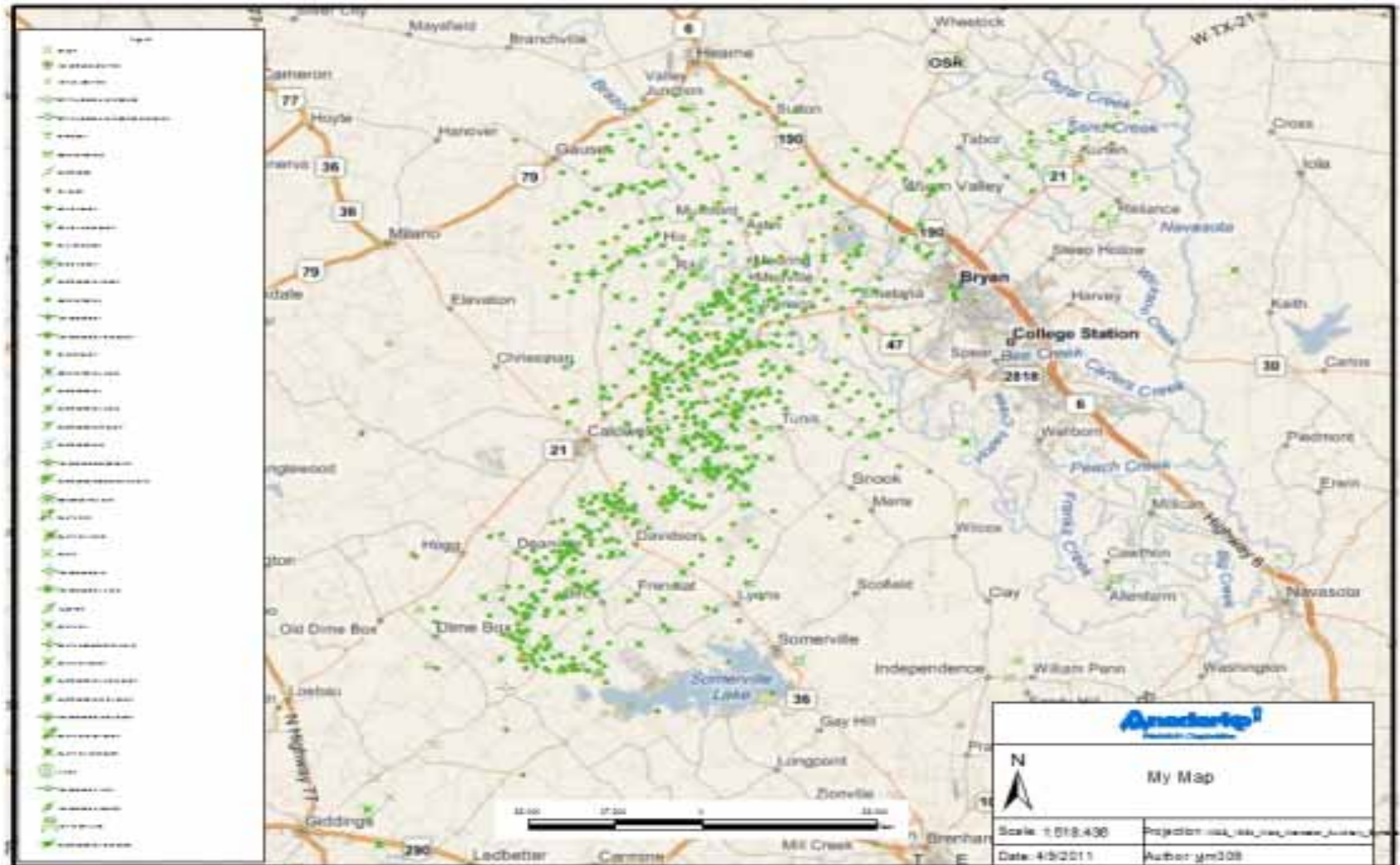
- Serializable (via Json.NET)
- UI Friendly (INotifyPropertyChanged)

# A GP Service for Enterprise Printing

Too many symbols causes Legend to run off page



# A GP Service for Enterprise Printing



# A GP Service for Enterprise Printing

Wells are accessible as a MapService layer - not as a FeatureLayer. So a Server Object Extension (SOE) was developed.

- Uses Esri REST API
- Request list of symbols that appear in extent
- Works only for layers that use a single field with a UniqueValueRenderer.

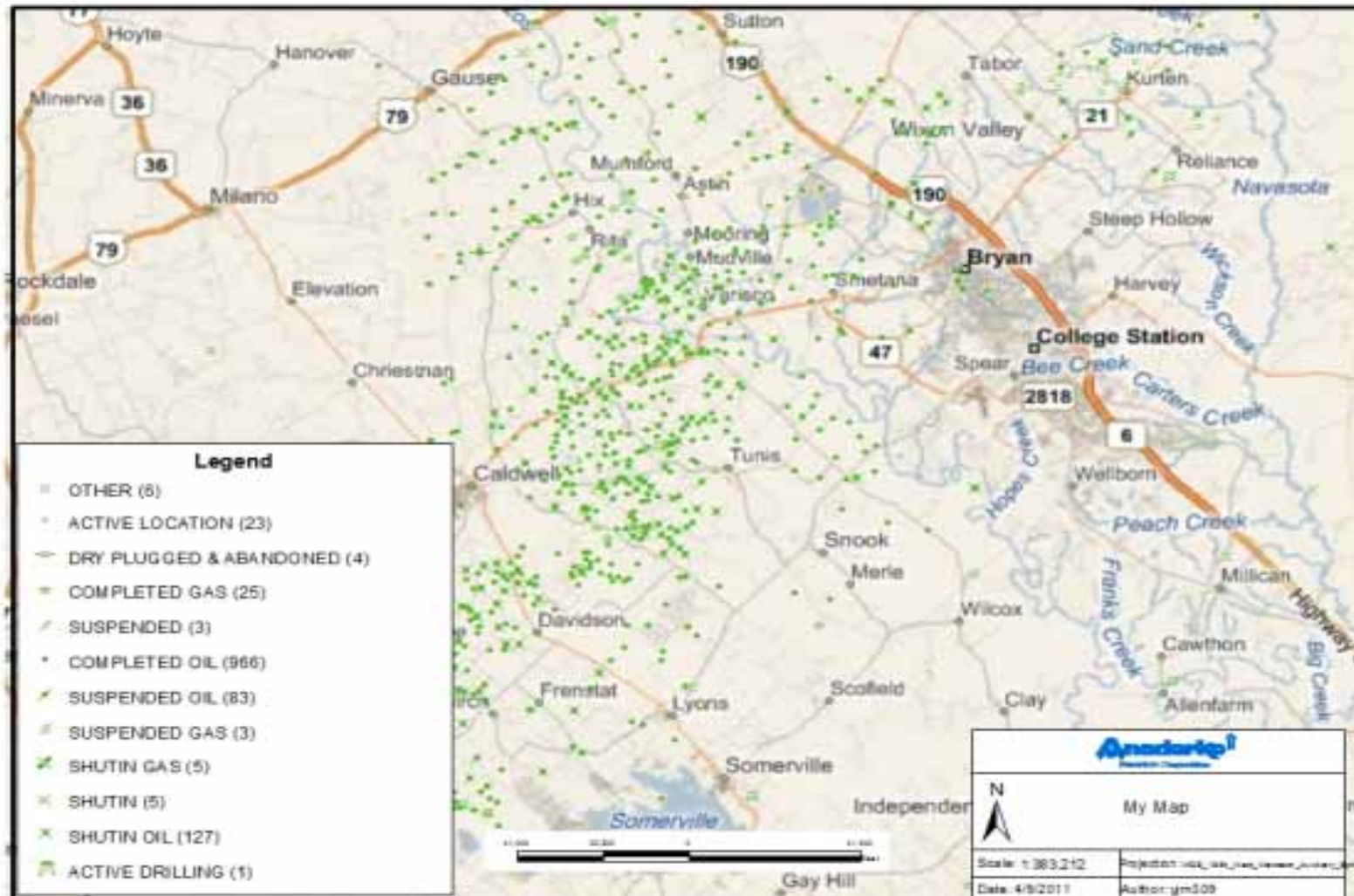
# A GP Service for Enterprise Printing

For legend layers that user has flagged for culling, the GP Export Tool:

- Calls SOE, passing map extent
- GETs a list of visible legend items.
- Removes legend items that are not in returned list
- Rescales Legend to fit on page.



# A GP Service for Enterprise Printing



## A GP Service for Enterprise Printing

Annoyance: when an mxd is opened, a Mapservice layer's legend items get repopulated.

While the pdf created for a job has a small legend, if the user opens the mxd from which the pdf was exported the legend gets repopulated with all symbols.

# A GP Service for Enterprise Printing

## Future Possibility: Generic Job Queue

Make Job Manager generic so that it handles Jobs for any asynchronous GP service that takes a long time to complete.

# A GP Service for Enterprise Printing

Questions?

Kevin Shows

[kevin.shows@anadarko.com](mailto:kevin.shows@anadarko.com)

Kirk Kuykendall

[kirk@ambergis.com](mailto:kirk@ambergis.com)