Publishing and Using Map Services with ArcGIS for Server

Craig Williams, Tanu Hoque, & Ty Fitzpatrick
Agenda

• Platform overview

• Publishing services
  - Demo: Publishing hosted feature service
  - Demo: Publishing in the enterprise
  - Demo: Publishing to the cloud with sync

• Map services and dynamic layers
  - Demo: Dynamic layers including data upload
  - Standardized queries at 10.2
  - Demo: standardized queries

• QA
ArcGIS Platform

- Online
- Desktop
- Server
- Mobile
- Developer
- Solutions
Publishing services
Information Sharing is Critical

... Sharing With a Group, Organization, or the Open Web

Users

Share

Search/Use

Transparency and easy information access are now expected...
Sharing as Services

Professional to Everyone

- Our goal: make it easier to share GIS resources
  - Unified sharing experience
  - Comprehensive Analysis
  - Sharing to servers in the enterprise, cloud, and to ArcGIS Online / Portal
Sharing as Services

ArcGIS Online

ArcGIS Server
Cloud

ArcGIS Server
On-Premises

ArcGIS Online or Portal

Desktop
Demo

Sharing services as hosted feature service
Hosted Mapping on ArcGIS Online

• **Sharing the easy way**
  - No Server to buy, install or maintain
  - Scales automatically
  - No Firewall or IT issues

• **Limitations**
  - Only Tiled Map and Feature Service supported
  - Data is private to each service
Service Definition (.sd)

- New file format for publishing starting at 10.1
  - Replaces .msd
  - For all services (Map, Geoprocessing, etc.)
- Contains everything required to create a service:
  - GIS-Resource (Map, Globe, etc.)
    - embeddable fonts (if needed)
  - Service Configuration
  - Data (if it needs to be copied to the server)
- Uploaded to the server when publishing
- Can be saved and published later
  - Using Catalog or Server Manager
Demo

Sharing services to an ArcGIS Server in the Enterprise
ArcGIS Server in the Enterprise

• Improvements that came at 10.1
  - Successful publishing to shared data
  - Comprehensive Analysis
  - Simple publishing with copying of data

• When to use
  - Need the full functionality of ArcGIS Server
  - Want full control over all hard and software
  - Want to publish services on live, shared data
ArcGIS Server Data Stores

- Server has a list of registered Data Stores
- Data Store is of type
  - Enterprise Geodatabase
  - Folder
    - Tip: register lowest folder level possible for maximal benefit
- Access to the Data Store is validated during registration
  - On all machines in the site
- Replicated data stores for separate data instances
- Tip: Read Help Topic “About registering your data with the server”

Data Stores are a key concept for publishing in 10.1
Demo

Publishing to the cloud with replication sync capabilities

ArcGIS Server in the Cloud
Map Services: Dynamic layers, statistics, queries
Review of map services since 10.1

• One unified map service
  - An updated optimized map service
  - Supports additional capabilities, data types, layers, renderers

• New extension capabilities:
  - Network Analysis

• New approach for Geoprocessing results
Dynamic Layers: The Concept

• Capability with the map service that allows for per-request changes to the map
  - Optional capability of map services

• Allows for:
  - Updating renderers and symbols
  - Removing and reordering layers
  - Changing layer data sources
  - Adding new layers from registered data sources
Dynamic Layers: Use Cases

- Simple updates to the map service
  - Remove layers or reorder layers
- Thematic mapping
  - Updates to renderers
- Adding content to the map service
  - Add data from registered workspaces
    - Including query layers
  - Add to the map on a per-request basis
    - WebAPIs handle this for you
Thematic mapping

• Special case of dynamic layers supported via
  - Ability to change renderer
  - Ability to change data sources – including joins
  - New map server API for constructing renderer classes

• Generate renderer operation
  - Supports class breaks and unique value class generation
  - Popular classification types from ArcGIS Desktop
Thematic Mapping

• When do I use dynamic layers instead of feature layers on the client for thematic mapping?
  - large number of features
  - complex geometries that cannot be generalized
  - when it provides a performance advantage

• Each approach has tradeoffs
  - e.g. Client side features scale better and provide more interactive behavior
  - Generate renderer can be used with both approaches
Dynamic Layers: More information

- See the topic “About dynamic layers” in the ArcGIS Server help
- REST API – See the “Dynamic Layer / Table” resource help
Query in map services

- Support for order by, output statistics, and group by statistics was added for both layers / tables
  - count | sum | min | max | avg | std dev | var

- Optional ability to return M and Z values for features
- Can query a specific geodatabase version
- Query response contains a flag when maxRecordCount was reached by the query
Demo

- Dynamic layers including data upload
Demo Overview:

- How to enable dynamicLayer?
- Change layer’s renderer
- Add new layer
- Use client side data with map service
Enabling dynamic layers
Demo #1: modify drawing info
http://esriurl.com/5978

dynamicLayers: generate renderer

REST request

http://.../MapServer/2/generateRenderer?
classificationDef = {
   "type": "classBreaksDef",
   "classificationField": "POP2007",
   "classificationMethod": "esriClassifyQuantile",
   "breakCount": 5,
   "colorRamp": {
      "type": "algorithmic",
      "fromColor": [255, 255, 255, 255],
      "toColor": [0, 0, 255, 255],
      "algorithm": "esriCIELabAlgorithm"
   }
}
**dynamicLayers: update renderer**

REST request

```
http://.../Census/MapServer/export?....&
dynamicLayers = [ {
    "source": {
        "type": "mapLayer",
        "mapLayerId": 0
    },
    "drawingInfo": {
        "renderer": {
            "type": "classBreaks",
            "field": "POP2007",
            "classBreakInfos": [ ... ]
        }
    }
} ]
```
var generateRenderer = new esri.tasks.GenerateRendererTask("http://serv../cens/MapServer/2");

var params = new esri.tasks.GenerateRendererParameters();
params.classificationDefinition = classDef;

generateRenderer.execute(params, applyRenderer, errorHandler)
dynamicLayers: update renderer

JavaScript API code

```javascript
generateRenderer.execute(params, applyRenderer, errorHandler);

function applyRenderer(renderer) {
    var drawingOptions = new
        esri.layers.LayerDrawingOptions();
    drawingOptions.renderer = renderer;
    optionsArray[2] = drawingOptions;
    usaLayer.setLayerDrawingOptions(optionsArray);
    usaLayer.show();
}
```
Demo #2: add new layer
**dynamicLayers: add layer from registered workspace**

REST request  

```
http://esriurl.com/5980
```

```
http://.../USA/MapServer/export?....&

dynamicLayers = [ {
    "id": 104,
    "source": {
        "type": "dataLayer",
        "dataSource": {
            "type": "table",
            "workspaceId": "MyDatabaseWorkspaceIDSSR2",
            "dataSourceName": "ss6.gdb.Lakes"
        }
    },
    "drawingInfo": {...}
} ]
```
dynamicLayers: add layer from registered workspace

JavaScript API code

```javascript
var dynamicLayerInfos = 
usaLayer.createDynamicLayerInfosFromLayerInfos();

var dataSource = new esri.layers.TableDataSource();
dataSource.workspaceId = "MyDatabaseWorkspaceIDSSR2";
dataSource.dataSourceName = "egdb.DBADMIN.USLakes";
var layerSource = new esri.layers.LayerDataSource();
layerSource.dataSource = dataSource;
layerSource.dataSource = dataSource;

var dynamicLayerInfo = new esri.layers.DynamicLayerInfo();
dynamicLayerInfo.source = layerSource;

dynamicLayerInfos.splice(0, 0, dynamicLayerInfo);
usaLayer.setDynamicLayerInfos(dynamicLayerInfos);
```
Demo #3: use client side data
Blog: http://esriurl.com/5993
http://esriurl.com/5982

dynamicLayers: add query layer

REST request

http://.../USA/MapServer/export?....&
dynamicLayers = [ {  
  "source": {  
    "type": "dataLayer",  
    "dataSource": {  
      "type": "queryTable",  
      "workspaceId": "MyDatabaseWorkspaceIDSSR2",  
      "query": "SELECT * FROM ss6.gdb.Lakes",  
      "oidFields": "objectid",  
      "geometryType": "esriGeometryPolygon",  
      "spatialReference": { "wkid": 4326 }  
    }  
  },  
  "drawingInfo": {...} } ]
var dynamicLayerInfos = usaLayer.createDynamicLayerInfosFromLayerInfos();  

var queryDataSource = new esri.layers.QueryDataSource();
queryDataSource.workspaceld = "MyDatabaseWorkspaceIDSSR2";
queryDataSource.query = "SELECT * FROM ss6.gdb.Lakes";
queryDataSource.oidFields = ["objectid"];
queryDataSource.geometryType = "polygon";
queryDataSource.spatialReference = new esri.SpatialReference({wkid:4326});

layerSource.dataSource = queryDataSource;
var dynamicLayerInfo = new esri.layers.DynamicLayerInfo();
dynamicLayerInfo.source = layerSource;

dynamicLayerInfos.splice(0, 0, dynamicLayerInfo);
usaLayer.setDynamicLayerInfos(dynamicLayerInfos);
Database agnostic way to query at 10.2

• One SQL syntax runs against all databases!
  - e.g. one date query syntax --- YEAH!
• Based on FileGDB query syntax and functions
  - Does not accept any DB vendor specific queries
• StandardizedQuery is turned on by default
  - Server level property
  - To switch back:
    http://<server>/arcgis/admin/system/properties/update
    {"standardizedQueries": "false"}

• Limitations at 10.2
  - Only a subset of functions available
  - Not supported for
    - Joined table/layer from multiple workspaces
    - OLE DB tables
Demo Overview:

- Advanced Analysis using Query Layer (!)
Hourly Rainfall Data and Relationship with other Layers
SQL to calculate total rainfall for each station
.... for January 2013

```
SELECT s.ESRI_OID, s.Site_No, s.shape, r.Rainfall
FROM mysde.map.usgs_rainfall_stations_il AS s
    INNER JOIN (
        SELECT site_no, SUM(rainfall_inch) AS rainfall
        FROM mysde.map.usgs_rainfall
        WHERE date_time
            BETWEEN '01/01/2013' AND '02/01/2013'
        GROUP BY site_no) AS r
    ON r.site_no = s.site_no
WHERE s.state_cd = 17
```
SELECT c.objectid, c.name, c.shape, fo.total_rainfall
FROM   mysde.map.usgs_counties_il AS c
       INNER JOIN (SELECT s.county_cd,
                        SUM(r.rainfall_inch) / Count(*) AS rainfall
                FROM   mysde.map.usgs_rainfall_stations_il AS s
                        INNER JOIN (SELECT site_no, SUM(rainfall_inch) AS rainfall_inch
                                      FROM   mysde.map.usgs_rainfall
                                      WHERE  date_time BETWEEN '1/1/2013' AND '2/1/2013'
                                      GROUP  BY site_no
                                    ) AS r
                        ON r.site_no = s.site_no
                WHERE  s.state_cd = 17
                GROUP  BY s.county_cd)  AS fo
       ON c.cnty_fips = fo.county_cd
SELECT c.objectid, c.name, c.shape, fo.rainfall
FROM usgs_counties_il AS c
INNER JOIN (SELECT t.fips,
 (SUM(r.rainfall_inch * t.area) / SUM(t.area)) AS rainfall
FROM usgs_counties_thsn_intsct_il AS t
INNER JOIN (SELECT site_no, Sum(rainfall_inch) AS rainfall_inch
FROM usgs_rainfall
WHERE date_time BETWEEN '1/1/2013' AND '2/1/2013'
GROUP BY site_no) AS r
ON r.site_no = t.site_no
GROUP BY t.FIPS) AS fo
ON c.fips = fo.fips
Session Survey

• Please fill out the survey for this session
  - http://esriurl.com/survey
  - Offering ID: 265

• Your comments help us with sessions for future conferences
Demo

- Standardized query
Questions?