Chapter 4: Working with Mapping and Rendering service

Displaying and Plotting Maps

Chapter Overview

- Working with layer, layer group, base layer group, and map
  - Access
  - Change the visibility
  - Layer definition file
  - Create new layers
  - Refresh and update map views
- Plotting maps
  - In DWF
  - In plain image

Overview of Layers

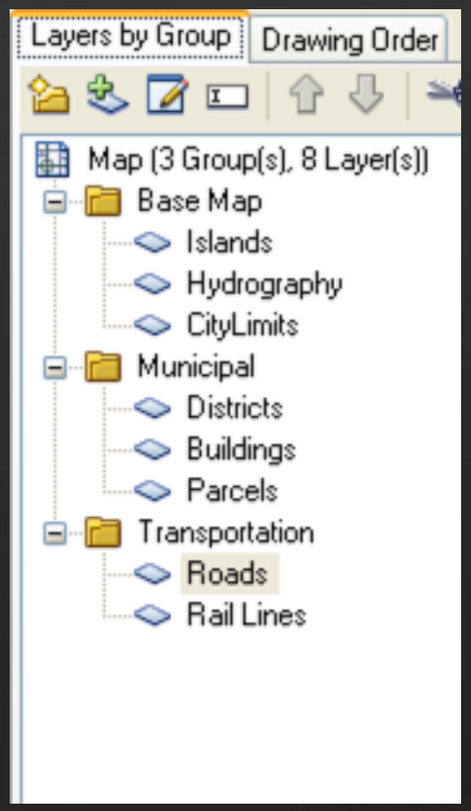
Map Layer:

- An overlay on a drawing composed of a logical group of data with the same geometric type.
- References a feature source or a drawing source.
- Represents vector data, raster data, or drawing data in a map.
- Contains styling and theming information, and a collection of scale ranges.

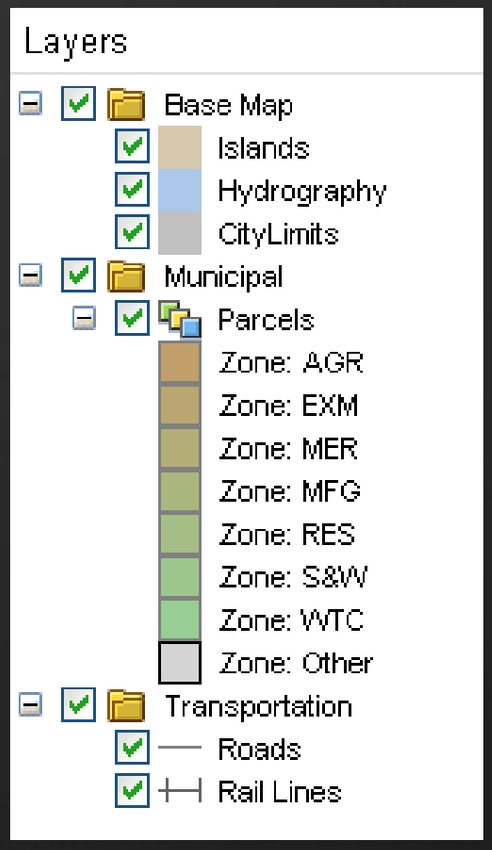
Layer Groups

- A collection of layers in a map sharing a common category
- Layer group's visibility overrides individual layer's setting.
- Layer groups can be nested.
- Each layer group in a map must have a unique name, even if it is nested within another group.
- A layer can only be in one group in a map at any time.

Studio



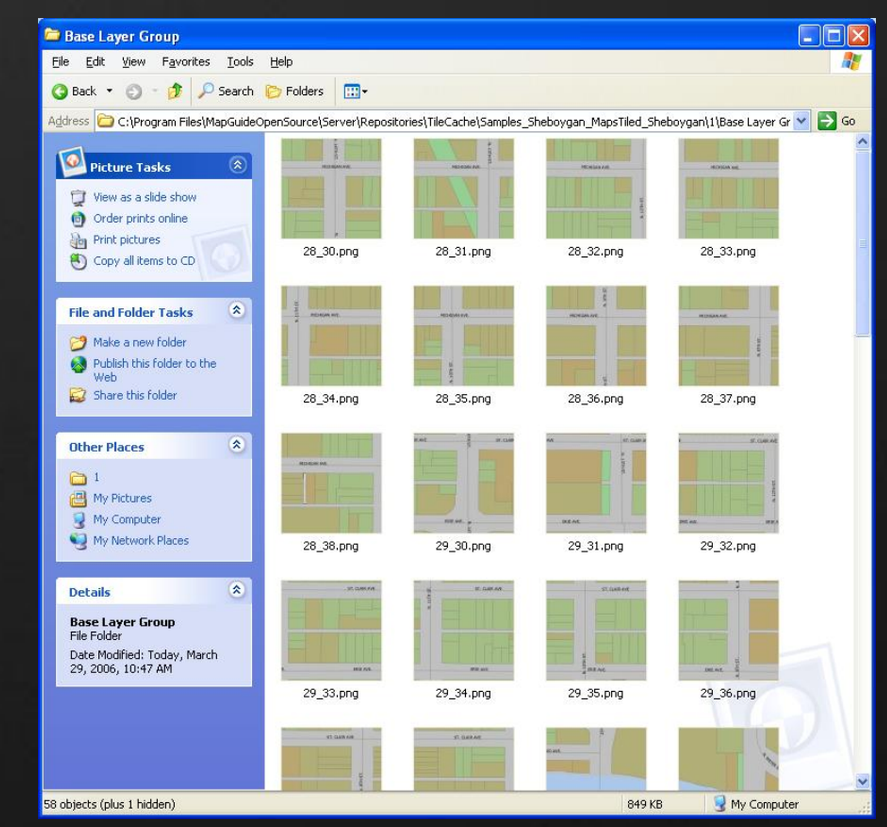
Map



Autodesk

Base Layer Groups

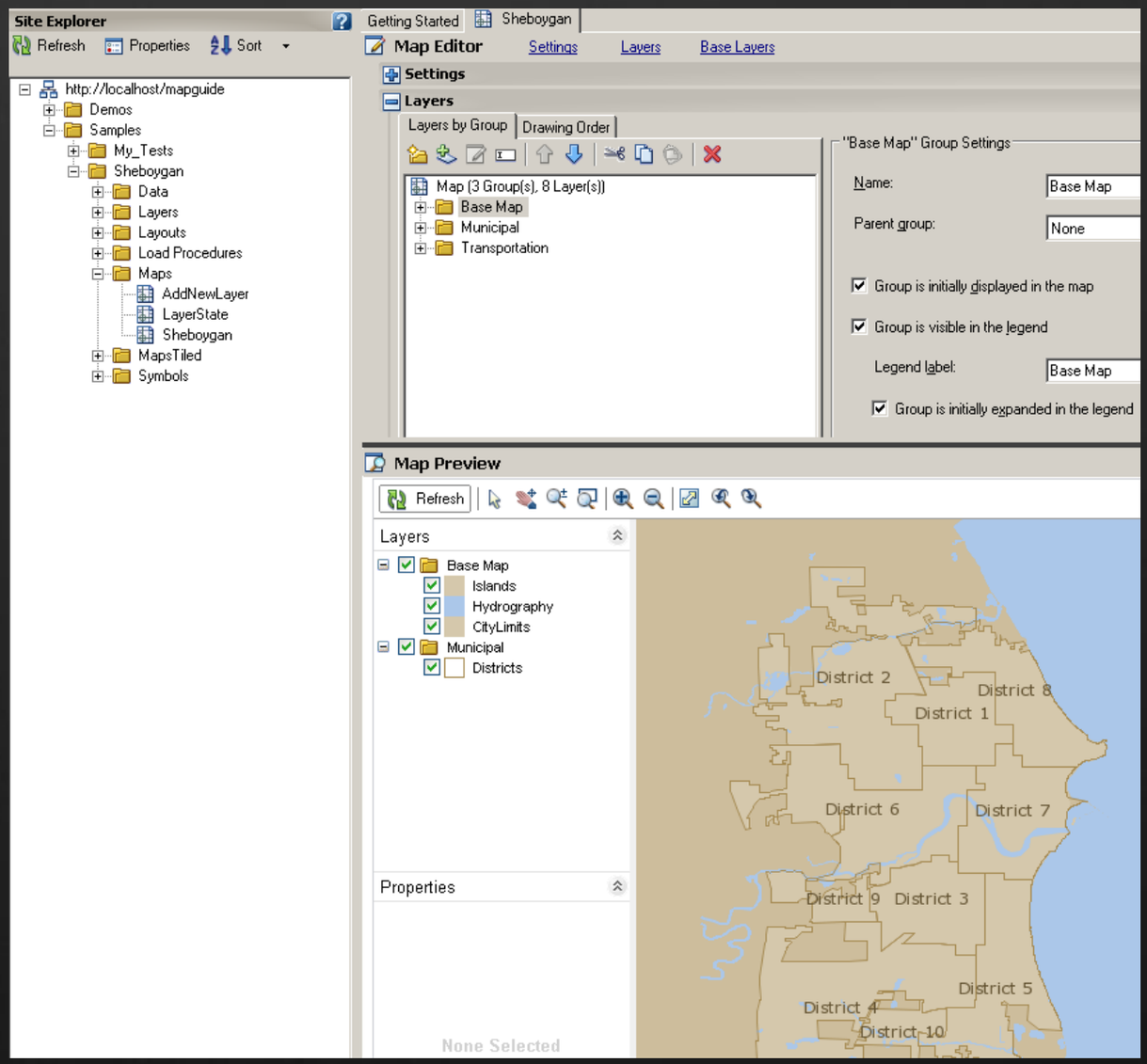
- Layers in a base layer group are rendered together to generate a single raster image.
- Individual layers cannot be turned on or off independently.
- Map images are tiled and cached.
- Has a list of scales. Only the scale in the list can be rendered.
- Base layer group is used by AJAX viewer.



Autodesk

Understanding Map

- Map is a resource that references a collection of layers and displays them within a consistent coordinate system and extents.



Autodesk

Accessing Layers

MgLayer object has member functions like – GetName(), GetSelectable(), GetVisibile(), GetGroup(), SetName(), SetVisible(), ForceRefresh() etc to get and set the properties of a Layer.

```
MgMap map = new MgMap();
map.Open(resService, "Sheboygan");
// Get layer Collection
MgLayerCollection layerCollection;
layerCollection = map.GetLayers();
MgLayer layer = null;

for (int i = 0; i < layerCollection.GetCount(); i++)
{
    layer = layerCollection.GetItem(i);
    Response.Write("<p>" + layer.GetName() + "</p>");
}
```

Using API to control Layer visibility

Whether a layer is visible in a given map depends on three criteria:

- The visibility setting for the layer
- The visibility settings for any groups that contain the layer
- The map view scale and the layer definition for that view scale

```
MgMap map = new MgMap();
map.Open(resService, "Sheboygan");
MgLayer tmpLayer =
    utility.getLayerByName(map, "Districts");
tmpLayer.SetVisible(!tmpLayer.IsVisible());
tmpLayer.ForceRefresh();
map.Save(resService);
```

Using API to control Layer Group visibility

The visibility for all layers in a group can be set at the group level. If the group visibility is turned off then none of the layers in the group will be visible, regardless of their individual visibility settings. If the group visibility is turned on, then individual layers within the group can be made visible or not visible separately.

```
MgMap map = new MgMap();
Map.Open(resourceService, mgMapName);

MgLayerGroupCollection
    layerGroups = map.GetLayerGroups();
// Get the Layer group Named 'Municipal'
MgLayerGroup municipal =
    layerGroups.GetItem("Municipal");
// Set the layer group visibility to 'OFF'
Municipal.SetVisible(False);
Map.Save(resourceService);
```

Add existing Layers to Map

- If the layer already exists in the resource repository, we can add it to the map by getting the map's layer collection and then adding the layer to that collection.

- Note: by default, newly added layers are added to the bottom of the drawing order, so they may be obscured by other layers. If you want to specify where the layer appears in the drawing order, use the LayerCollection.Insert() method.

```
// Get the layer collections
MgLayerCollection layerCollection = map.GetLayers();
layerCollection.Add(layer)
```

Create a New Layer

- Create a new layer with an existing layer definition file.
- Create a new layer with LayerDefinitionFactory.

Creating Layers from XML

1. Build a prototype layer through Studio UI
2. Use Studio to save the layer as an XML file
3. Load the XML file and change the necessary XML elements if necessary
4. Save the XML into the repository as Layer Definition
5. Create a new MgLayer object
6. Add the layer to map

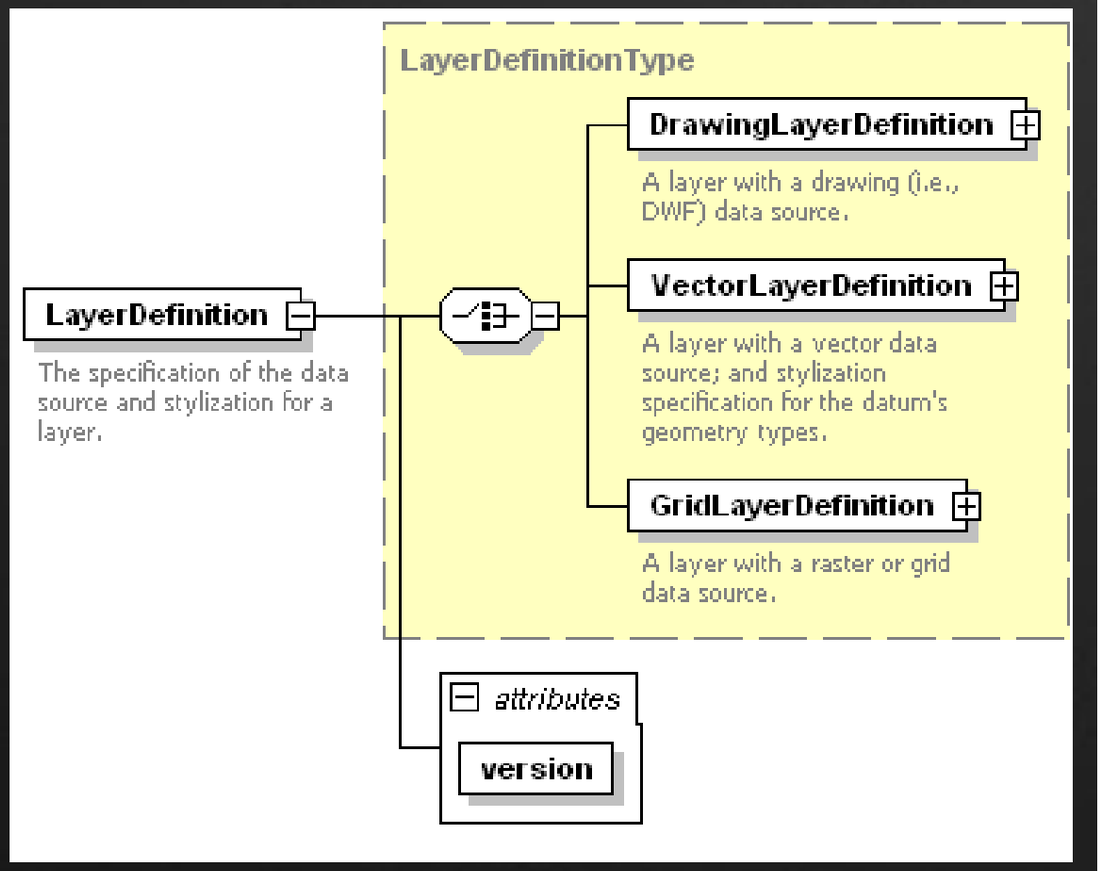
Creating Layers from XML

The following code snippet creates a layer by loading the XML file and add the layer to the map.

```
MgMap map = new MgMap();
map.Open(resourceService, "Sheboygan");
// load the XML file and create the layer definition
MgByteReader layerDefContent =
    BuildLayerDefinitionContent(@"C:\Temp\NewLayer.XML");
resourceService.SetResource(layerDefId, layerDefContent, null);
//Add the layer to the map, if it's not already in it
if (layer == null)
{
    layer = new MgLayer(layerDefId, resourceService);
    layer.SetName("Line1");
    layer.SetDisplayInLegend(true);
    layer.SetLegendLabel(legendName);
    layer.SetVisible(false);
    layers.Insert(0, layer);
}
```

LayerDefinition Structure

LayerDefinition – Stores the definition of a map layer, including a reference to the layer data (FeatureSource, DrawingSource, or FeatureSet) and the rules for how to present that data.



Elements	Complex types	Simple types
LayerDefinition	AreaRuleType	BackgroundStyleType
	AreaSymbolizationFillType	FeatureNameType
	AreaSymbolizationType	LengthUnitType
	AreaTypeStyleType	ShapeType
	BaseLayerDefinitionType	SizeContextType
	BlockSymbolType	
	ChannelBandType	
	DrawingLayerDefinitionType	
	FillType	
	FontSymbolType	
	GridColorBandsType	
	GridColorRuleType	
	GridColorStylizationType	
	GridColorType	
	GridLayerDefinitionType	
	GridScaleRangeType	
	GridSurfaceStylizationType	
	HillshadeType	
	ImageSymbolType	
	LayerDefinitionType	
	LineRuleType	
	LineTypeStyleType	
	MarkSymbolType	
	NameStringPairType	
	PointRuleType	
	PointSymbolization2DType	
	PointSymbolizationType	
	PointTypeStyleType	
	StrokeType	
	SymbolType	
	TextSymbolType	
	VectorLayerDefinitionType	
	VectorScaleRangeType	
	W2DSymbolType	

Autodesk

Creating A Layer by Layer Definition

The following code snippet demonstrates creating line-based rules to be used in Layer creation.

```
// Create a scale range.
string minScale = "0";
string maxScale = "1000000000000";
string pointScaleRange = factory.CreateScaleRange(
    minScale, maxScale, pointTypeStyle);

// Create the layer definiton.
string featureName = "//PointSchema:Points";
string geometry = "GEOM";
string layerDefinition = factory.CreateLayerDefinition(
    featureSourceName, featureName, geometry, pointScaleRange);
```

Accessing MgMap

The following code snippet demonstrates how to get the Map details.

```
MgMap map = new MgMap();
map.Open(resService, "Sheboygan");

MgEnvelope envelope = map.GetMapExtent();
MgCoordinate lowerLeft = envelope.GetLowerLeftCoordinate();
MgCoordinate upperRight = envelope.GetUpperRightCoordinate();
```

Forcing the Viewer to Refresh

In any server-side script where the Map is changed, the HTML returned must ask the Viewer to refresh itself. This is most easily accomplished with an DHTML **onLoad** event.

```
<script language="javascript">
function OnPageLoad()
{
    parent.MapFrame.Refresh();
}
</script>
<body onLoad="OnPageLoad()">
. . .
</body>
```

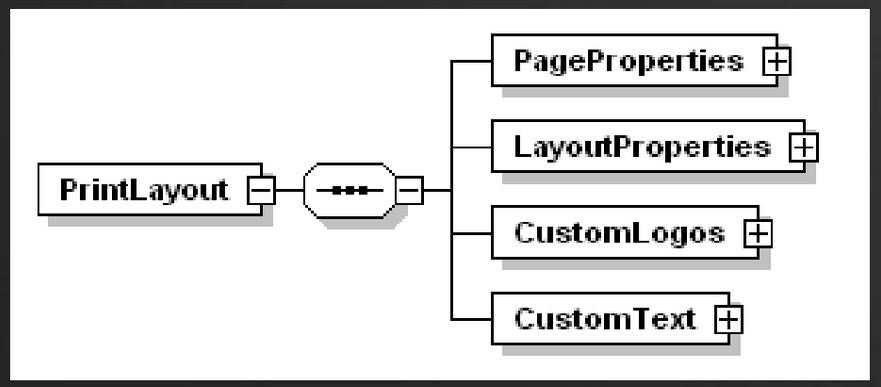
Plotting Maps in Plain Images

- Maps can be plotted to vector-based DWF format in high quality
- Map plotting is independent of map viewer

```
MgMap map = new MgMap();
map.Open(resourceService, "Sheboygan");

MgSelection selection = new MgSelection(map);
selection.Open(resourceService, "Sheboygan");

MgByteReader mapBytes =
    renderingService.RenderDynamicOverlay(map, selection, "PNG");

MgByteSink byteSink = new MgByteSink(byteReader);
string filePath = @"C:\Temp\" + "Map" + sessionId + ".PNG";
byteSink.ToFile(filePath);
```

PrintLayout Schema

Print Layout is an XML template for customizing the appearance of printed Maps.




```
<?xml version="1.0"  encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualifie
  <xs:element name="PrintLayout">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PageProperties">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="BackgroundColor" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Red" type="xs:integer" minOccurs="0"/>
                    <xs:element name="Blue" type="xs:integer" minOccurs="0"/>
                    <xs:element name="Green" type="xs:integer" minOccurs="0"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="LayoutProperties">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ShowTitle" type="xs:boolean" minOccurs="0"/>
              <xs:element name="ShowLegend" type="xs:boolean" minOccurs="0"/>
              <xs:element name="ShowScaleBar" type="xs:boolean" minOccurs="0"/>
              <xs:element name="ShowNorthArrow" type="xs:boolean" minOccurs="0"/>
              <xs:element name="ShowURL" type="xs:boolean" minOccurs="0"/>
              <xs:element name="ShowDateTime" type="xs:boolean" minOccurs="0"/>
              <xs:element name="ShowCustomLogos" type="xs:boolean" minOccurs="0"/>
              <xs:element name="ShowCustomText" type="xs:boolean" minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="CustomLogos">
```


Autodesk

Plotting Maps in DWF

The following code snippet shows how to plot maps in the format of DWF.

```
MgMap map = new MgMap();
map.Open(resourceService, "Sheboygan");
MgDwfVersion dwfVer = new MgDwfVersion("6.01", "1.2");
MgPlotSpecification plotSpec = new MgPlotSpecification(
    8.5f, 11.0f, MgPageUnitsType.Inches, 0f, 0f, 0f, 0f);
plotSpec.SetMargins(0.5f, 0.5f, 0.5f, 0.5f);

MgResourceIdentifier layoutRes = new MgResourceIdentifier(
    "Library://Test/SheboyganMap.PrintLayout");

MgLayout layout = new MgLayout(layoutRes,
    "City of Sheboygan", MgPageUnitsType.Inches);

MgByteReader byteReader = mappingService.GeneratePlot(
    map, plotSpec, layout, dwfVer);

MgByteSink byteSink = new MgByteSink(byteReader);
string filePath = @"C:\Temp\" + "Map" + sessionId + ".PNG";
byteSink.ToFile(filePath);
```

Questions

Questions ?

Autodesk

Exercise

- Examine the XML presentation of LayerDefinition / PrintLayout using MapAgent or Maestro