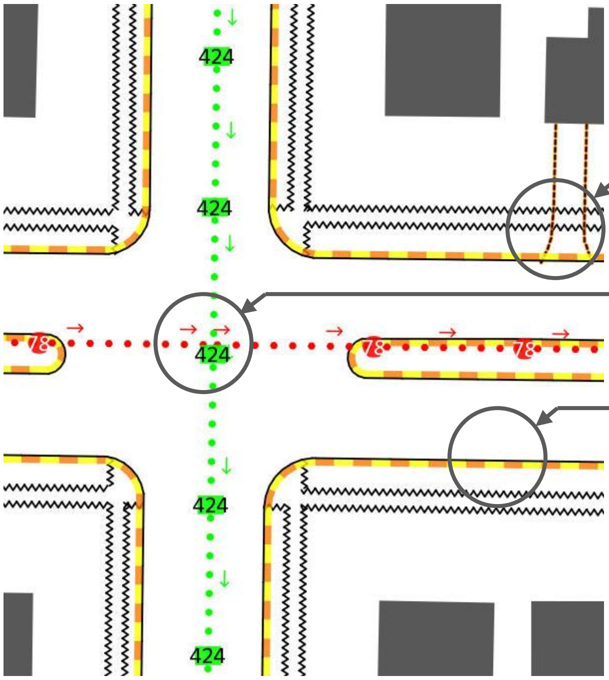
Building Symbol Libraries with Autodesk MapGuide® Enterprise

Robert Bray – Autodesk, Inc.

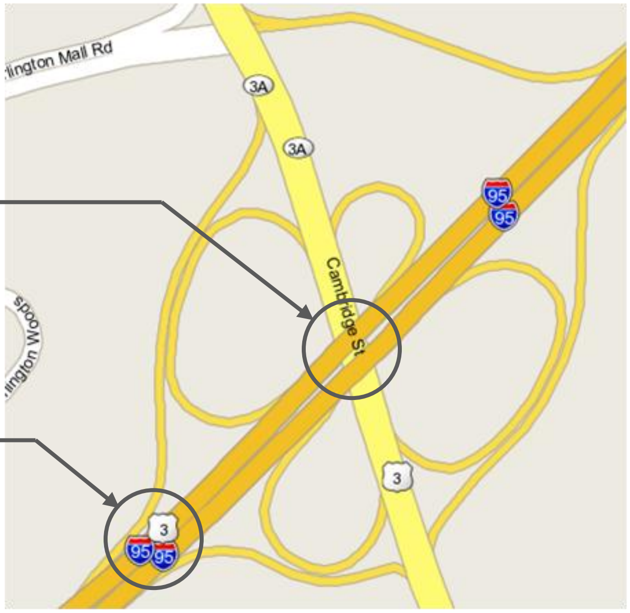
While not exposed to end users through Autodesk MapGuide Studio 2008, Autodesk MapGuide Enterprise 2008 includes new cartographic capabilities that can be used to build custom symbol libraries. This class will provide an introduction to MapGuide's new cartographic model, show how to create custom symbols and use them to symbolize points and lines, show how to create and use compound symbols, and how to create data driven symbols. The following examples illustrate some of the capabilities of MapGuide's new cartographic model.



Custom line styles and patterns.

Repeating line labels and directional arrows.

Ability to offset cartographic symbols from the feature geometry.

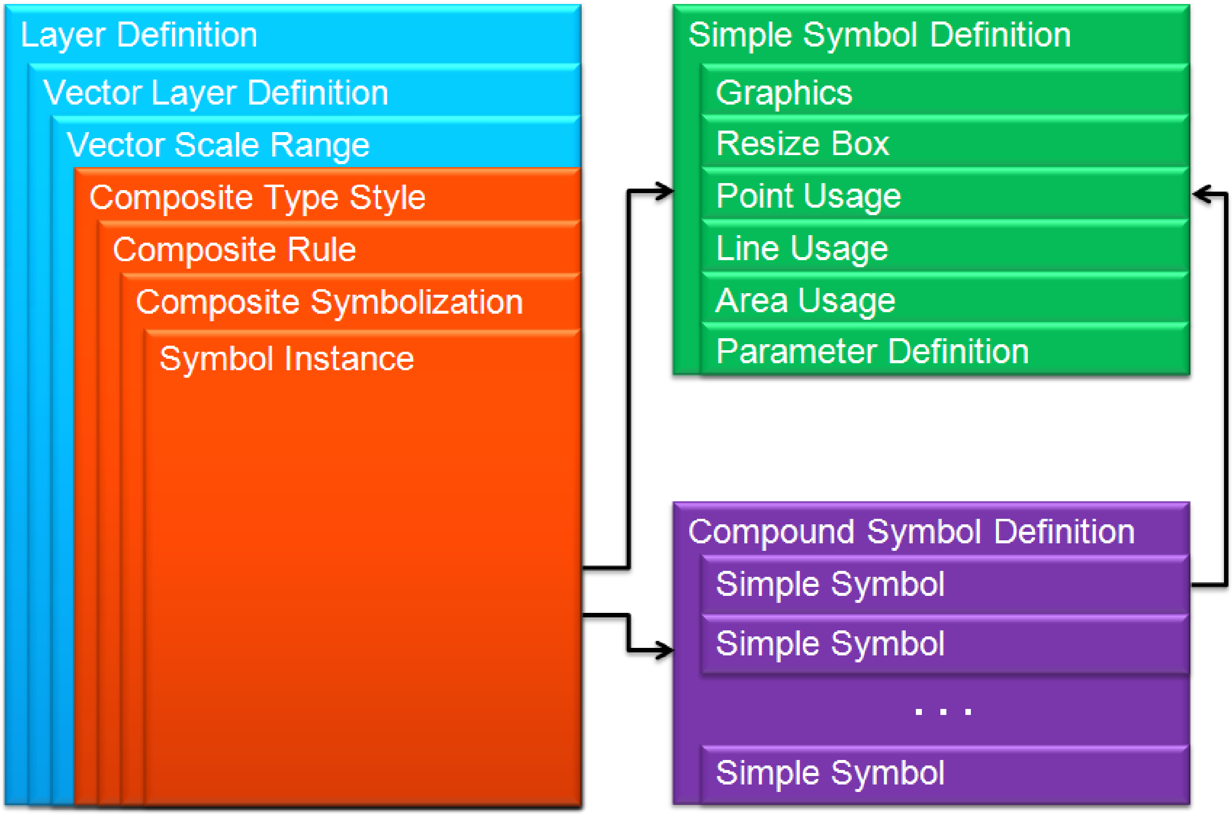


A more consistent and accurate way to define cased-line styles.

Use of symbols on lines for labeling, e.g. highway shields, utility symbols, etc.

An Overview of the New Cartographic Model

The new cartographic model is comprised of two new XML resource types Simple Symbol Definition and Compound Symbol Definition, as well as a related set of enhancements to the Layer Definition resource type as shown in the following diagram.



The Simple Symbol Definition resource type supports the definition of a symbol based on vector geometry, an image, or text. In addition a Simple Symbol Definition can specify how it is applied when used to stylize point, line, or area geometry. Finally the Parameter Definition specifies which values in the Simple Symbol Definition can be overridden by an FDO property value or expression during stylization.

The Compound Symbol Definition resource type supports combining multiple Simple Symbols to create a composite symbol. This is useful for creating cased lines or adding a repeating graphic at pre-defined intervals along a line (e.g. highway shields or directional arrows).

To support the new Symbol Definition resource types, the Layer Definition resource type has been modified to include a Composite Type Style element in addition to the Point Type Style, Line Type Style, and Area Type Style elements supported in the Autodesk MapGuide 2007 release. This new Composite Type Style element supports theming rules as before, but makes use of the new Symbol Definition resource types rather than a fixed set of stylization options per geometry type.

Simple and Compound Symbol Definition resources are standalone resource types that can be shared across any number of Layer Definitions. By creating folders of these resource types it is possible to create a symbol library that is usable across and organization.What's the Deal, Why no UI in Autodesk MapGuide Studio 2008?

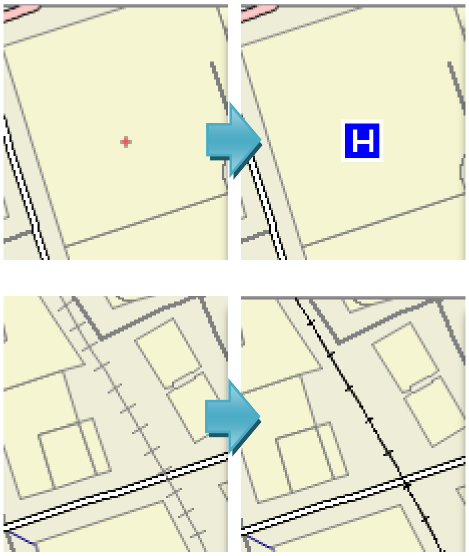
The implementation of the new cartographic model in Autodesk MapGuide 2008 is incomplete. The status of the implementation in Autodesk MapGuide 2008 and MapGuide Open Source 1.2.0 is as follows:

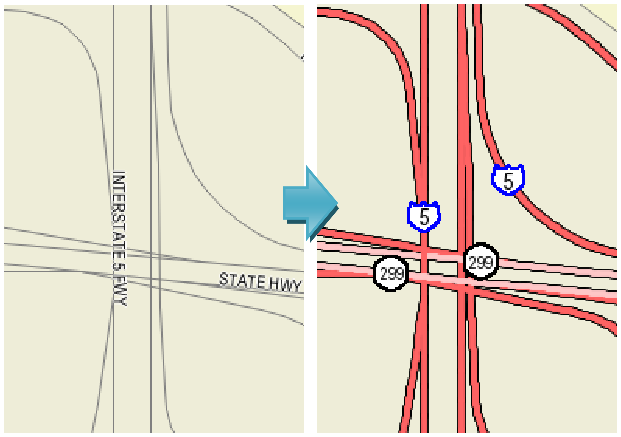
- Symbol definition usage with point features is fully functional.
- Symbol definition usage for highway shields and other linear annotation is fully functional.
- Complex line joins may or may not provide expected results, depending on the symbol definition.
- Symbol definition usage for area features is not supported.

The upcoming release of MapGuide Open Source 2.0 will have more complete support for using symbol definitions for line and area features, finer grained control over rendering passes, support for angular offsets relative to the geometry, and a bunch of bug fixes. Future releases of Autodesk MapGuide Enterprise will most likely incorporate these changes as well.

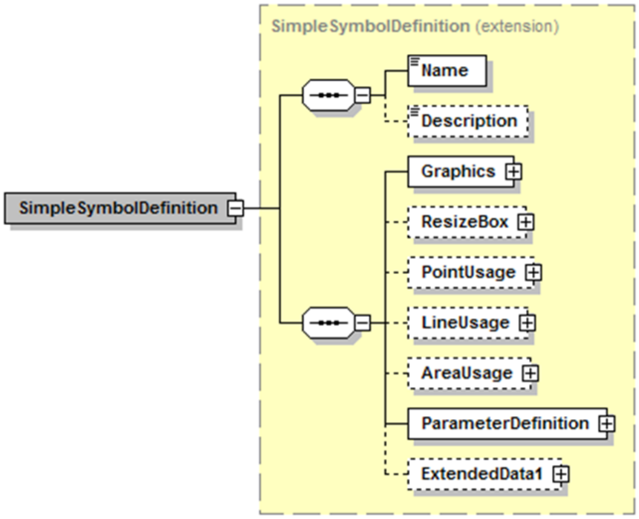
Samples Discussed in this Class

In this class we'll create a small symbol library and use it to update the cartography in the City of Redding sample data set. In particular we'll create an international hospital symbol, create a less gaudy railroad line style, and create new styles for the highways and roads (incorporating highway shields where appropriate). The images below show the map before and after:





Simple Symbols and Point Usage

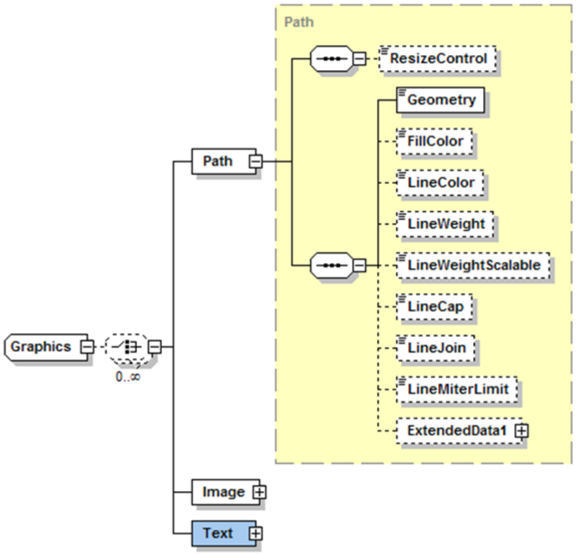


The structure of the Simple Symbol Definition is shown at right. The Name and Description elements are self explanatory, Name is mandatory while Description is optional, but highly recommended if you plan to build reusable libraries of symbols. The Graphics element defines the symbol in terms of vector geometry, an image, text, or some combination of these. The exact structure of the Graphics element will be covered in the next section. The Resize Box element specifies the center and size of the symbol, and whether the bounds of the symbol should grow based on the content of the Graphics element. More detail on this behavior will be provided a little later in the document. If the Resize Box element is not specified the size of the symbol will be derived from the content of the Graphics elements. The Point, Line, and Area Usage elements define how this symbol should be used for the various geometric primitives. All are optional and if not specified default behavior will be supported. Finally the Parameter Definition element defines a collection of parameters that can be substituted at runtime based on an FDO expression or property value. Both the Usage and Parameter Definition elements will be discussed in more detail a little later in this document.

Defining Graphics

All of the Graphics element types start with a Resize Control element that specifies how this graphic element interacts with the Resize Box of the symbol. This must evaluate to one of: ResizeNone (default), AddToResizeBox, or AdjustToResizeBox. AddToResizeBox means the element's graphical extent is added to the resize box, but the element is not resized or repositioned if the resize box grows. AdjustToResizeBox means the element is resized and repositioned relative to the resize box, but its extent is not added to the box. ResizeNone means the element does not interact with the resize box.

Graphics as a Path

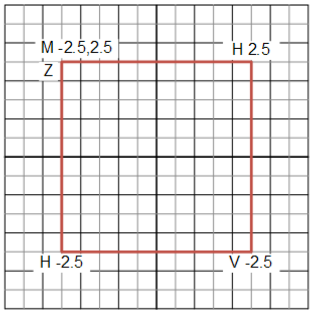


The structure of the Graphics element for path based vector geometry is shown at right. The format of the Geometry element is a sequence of segments, each represented by a letter indicating the segment type followed by one or more parameters. Uppercase letters denote absolute values and lowercase letters denote relative values. Segment types can be one of "M" (moveto), "L" (lineto), "H" (horizontal lineto), "V" (vertical lineto), "A" (arcto), or "Z" (close segment). All coordinates are specified in millimeters and defined in the Cartesian plane. The format and behavior of each of these segment types is described in the following table.

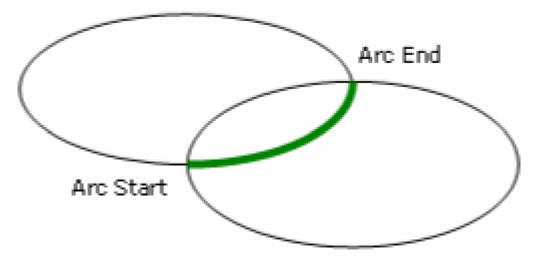
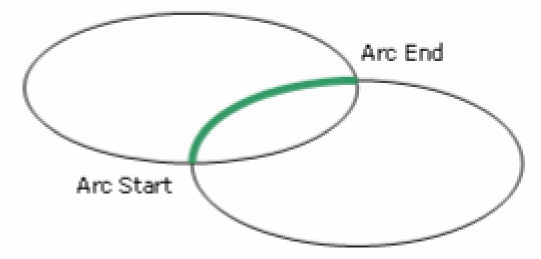
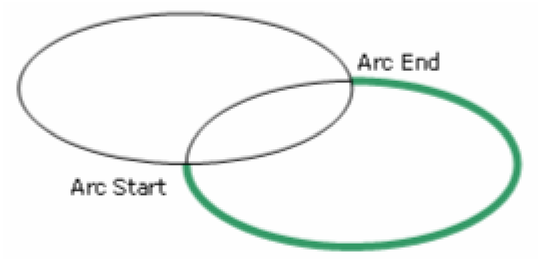
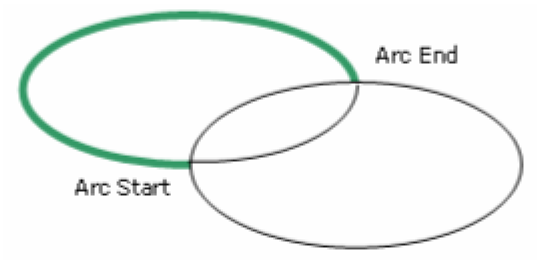
Command	Syntax	Description
Move	M x,y or m x,y	Establishes a new current endpoint. Every geometry may specify one or more figures. The first figure in a geometry must begin with a Move command.
Line	L x,y or l x,y	Draws a straight line from the current point to the specified point.
Horizontal Line	H x or h x	Draws a horizontal line from the current point to the x coordinate.
Vertical Line	V y or v y	Draws a vertical line from the current endpoint to the specified y coordinate.
Arc	A xr,yr rx fArc fSweep x,y or a xr,yr rx fArc fSweep x,y	Draws an elliptical arc from the current endpoint to the specified point (x,y). • xr defines the x radius • yr defines the y radius • rx defines the x-axis rotation in degrees • fArc identifies the arc segment to use, a value of 1 specifies to use the large sweep 180 degrees or greater, a value of 0 specifies to use the small sweep less than 180 degrees. • fSweep defines direction, if fSweep is 1, the arc is drawn clockwise. If fSweep is 0, the arc is drawn counter-clockwise.
Close	Z or z	Draws a straight line from the current endpoint to the first point of the current figure and then ends the figure.

For example to draw a 5 mm square box the Geometry element would be specified as follows:

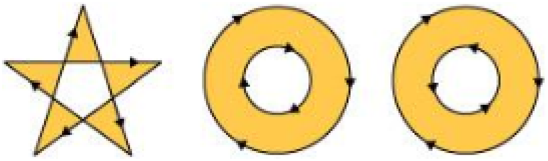
Geometry: M -2.5,2.5 H 2.5 V -2.5 H -2.5 Z



Arcs are somewhat more complex and the following table should help explain the fSweep and fArc parameters.

fArc = 0, fSweep = 0  	fArc = 0, fSweep = 1 
fArc = 1, fSweep = 0  	fArc = 1, fSweep = 1 

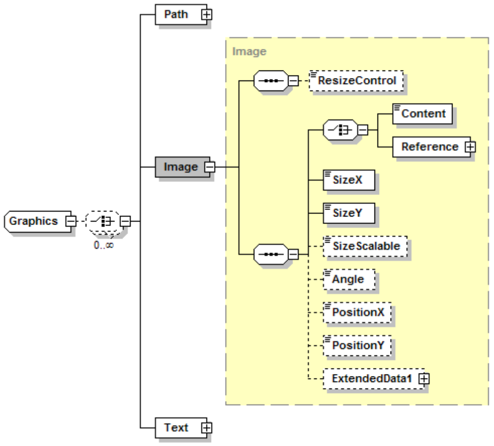
A concrete example of Arc usage will be shown a little later in this document. One final important point on the path element is how fills are handled. Which portions of a figure to fill is determined using an even-odd fill algorithm. This is illustrated as follows:



The other elements within the Path element are optional. Fill Color and Line Color are specified in ARGB hexadecimal format with an alpha transparency. Like all other values in symbol definitions Line Weight is specified in millimeters. Line Weight Scalable is a boolean value which specifies whether the line weight scales with the symbol. Line Cap specifies the cap type to use at the ends of each segment in the path outline which must evaluate to one of: None, Round (default), Triangle, or Square. Line Join specifies the join type to use at each vertex in the path outline which must evaluate to one of: None, Bevel, Round (default), or Miter. Finally the Line Miter Limit specifies the limit to use when drawing miter joins. A miter join is trimmed if the ratio of the miter length to line weight is greater than the miter limit. If specified this must be greater than zero; the default value is 5.

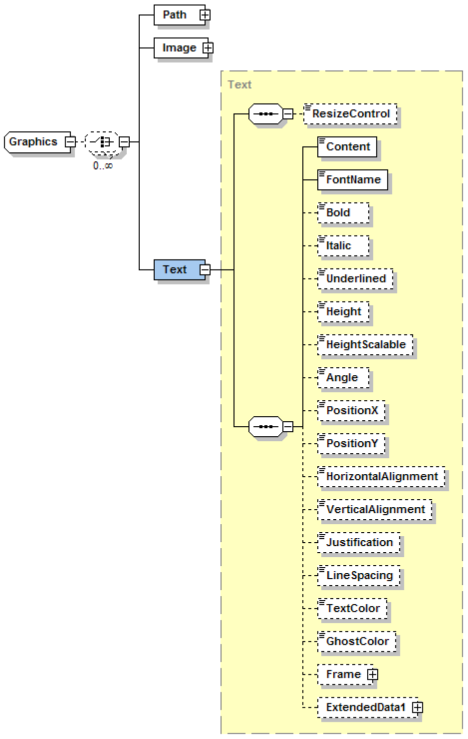
Graphics as an Image

The structure of the Image element for defining graphics as an image is shown at right. The image itself may either be specified in the Content element as base 64 encoded PNG or in the Reference element as a link to the image stored as Resource Data within the resource database. The Size X and Size Y elements specify the size of the symbol in millimeters. Size Scalable is a boolean value which specifies whether the image sizes scale with the symbol. The Angle element specifies the rotation angle of the image in symbol space, in degrees. The Position X and Position Y elements specify the image center within symbol space in millimeters.



Graphics as Text

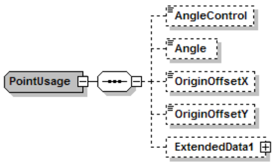
The structure of the Text element for defining graphics as text is shown at right. The Content element specifies the text to render for the symbol. The FontName, Bold, Italic, and Underlined elements specify the font and attributes used to render the text. The Height element specifies the text height in millimeters. The Height Scalable element is a boolean value which specifies whether the font height scales with the symbol. Angle specifies the rotation angle of the text in symbol space, in degrees. Position X and Position Yspecify the x and y coordinate of the text within symbol space in millimeters. The alignment values are relative to this coordinate. The Horizontal Alignment element specifies the horizontal alignment of the text box relative to its position, which must evaluate to one of: Left, Center (default), or Right. The Vertical Alignment element specifies the vertical alignment of the text box relative to its position, which must evaluate to one of: Bottom, Baseline, Halfline (default), Capline, or Top. The Justification element specifies the horizontal justification of individual lines of text in a multi-line text string, which must evaluate to one of: Left, Center, Right, Justified, or



FromAlignment (default). The Line Spacing element specifies the spacing between individual lines of text in a multi-line text string, as a multiple of the font height. The Text Color and Ghost Color elements specify the color of the text and ghosting effect in ARGB hexadecimal format. The Frame element provides options for drawing a rectangular frame around the text.

Point Usage

The Point Usage element provides control over how a symbol is used to symbolize point features. The structure of the Point Usage element is shown at right. The Angle Control element specifies how the symbol angle is defined, which must evaluate to one of: FromAngle (default) or FromGeometry. The Angle element specifies the symbol angle, in degrees and only applies if AngleControl evaluates to FromAngle. The Orgin Offset X and Y elements specifies the horizontal and vertical offset to apply to the symbol origin, in mm. This offset is applied before the symbol is scaled and rotated.



Defining the International Hospital Symbol

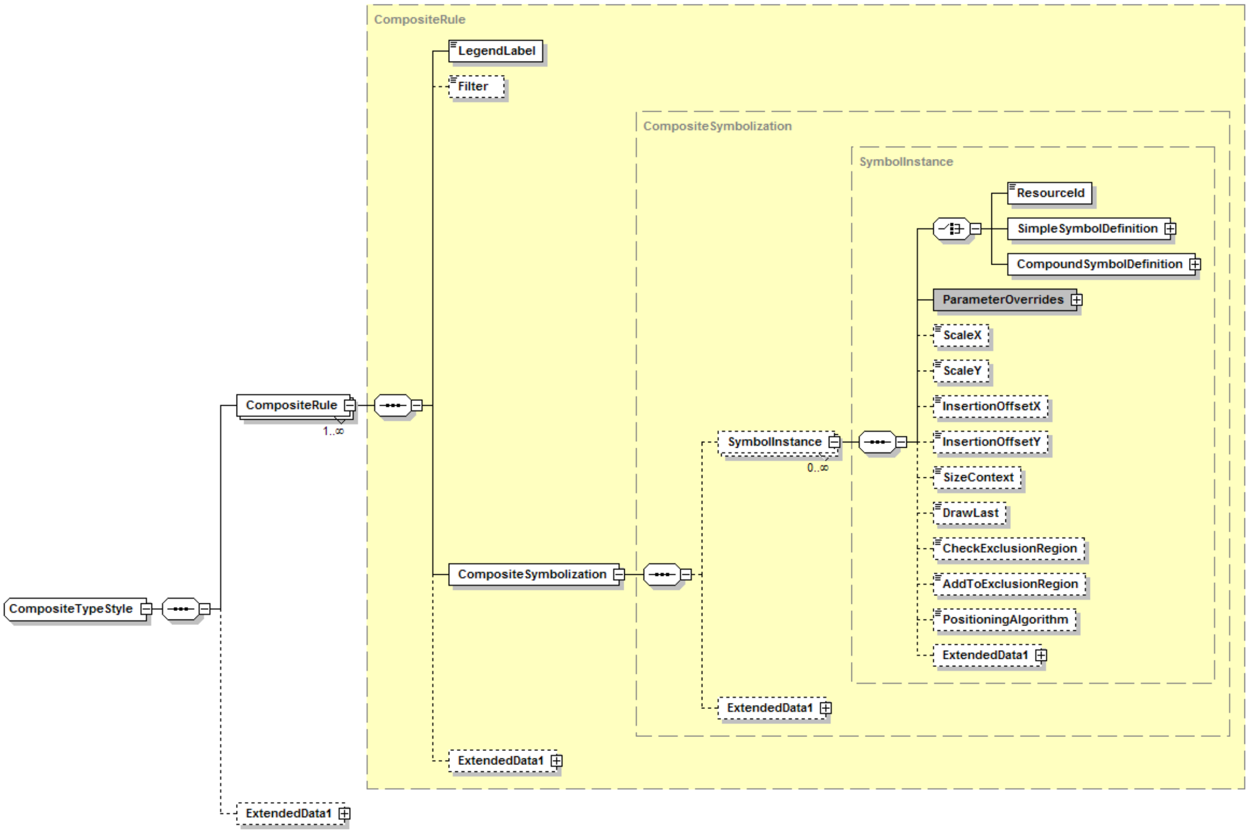
Now that the basics of symbol definition and point usage have been explained, let’s look at how to define an international hospital sign to replace the basic cross used on the Hospital layer.

```
<?xml version="1.0" encoding="UTF-8"?>
<SimpleSymbolDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SymbolDefinition-1.0.0.xsd" version="1.0.0">
  <Name>Hospital</Name>
  <Description>International Hospital Symbol</Description>
  <Graphics>
    <Path>
      <Geometry>M -2.5,2.5 H 2.5 V -2.5 H -2.5 Z</Geometry>
      <FillColor>FF0000FF</FillColor>
      <LineColor>FFFFFFFF</LineColor>
      <LineWeight>0.75</LineWeight>
      <LineWeightScalable>false</LineWeightScalable>
    </Path>
    <Path>
      <Geometry>M -1.0,1.25 V -1.25 Z M 1.0,1.25 V -1.25 Z M -1.0,0.0 H 1.0 Z</Geometry>
      <LineColor>FFFFFFFF</LineColor>
      <LineWeight>0.75</LineWeight>
      <LineWeightScalable>false</LineWeightScalable>
    </Path>
  </Graphics>
  <PointUsage/>
  <ParameterDefinition/>
</SimpleSymbolDefinition>
```

This symbol uses two Geometry elements, one to draw the blue box with a white border and one to draw the “H”. The Point Usage element is empty since we do not want the symbol rotated and we’ll use the center of the symbol as the insertion point.

Layer Definition Updates to Support Symbol Definitions

The structure of the new Composite Type Style element of Layer Definition is shown in the following diagram. In this document we’ll ignore Composite Type Style and Composite Rule as they are functionally equivalent to the Point, Line, and Area equivalents in previous versions of Layer Definition.



However the Symbol Instance element is new and interesting. Symbol Instance specifies the symbol(s) to use for a particular Composite Rule and provides control over how each symbol is rendered. First note that a Symbol Definition can either be referenced by the Symbol Instance via the Resource Id element or the definition of the symbol may be embedded within the Symbol Instance itself. The Parameter Overrides element will be covered later in the document, so we'll skip over that for now. Scale X and Y specify the additional amount to scale the symbol horizontally and vertically. Insertion Offset X and Y applies only to point symbols and specifies the additional amount to offset the symbol horizontally, in mm in device units, after scaling and rotation have been applied. Size Context specifies whether the symbol sizes are with respect to the map or the user's display device. Draw Last is a boolean value which specifies whether the symbol is drawn as part of a final rendering pass (e.g. for labeling). Check Exclusion Region is a boolean value which specifies whether to render this symbol only if its graphical extent does not overlap the exclusion region. If the positioning algorithm generates multiple candidate symbol positions and this setting is true, then only the first non-overlapping candidate is rendered. Similarly Add To Exclusion Region is a boolean value which specifies whether the graphical extent for this symbol instance should be added to the exclusion region (if it is rendered). Finally the Positioning Algorithm element specifies the algorithm used to generate candidate positions for the symbol. If specified this must evaluate to one of: Default or EightSurrounding. Default means generate one position using the feature geometry (used for normal rendering). EightSurrounding means generate eight positions surrounding the feature geometry (used when labeling point features).

The Updated Hospital Layer

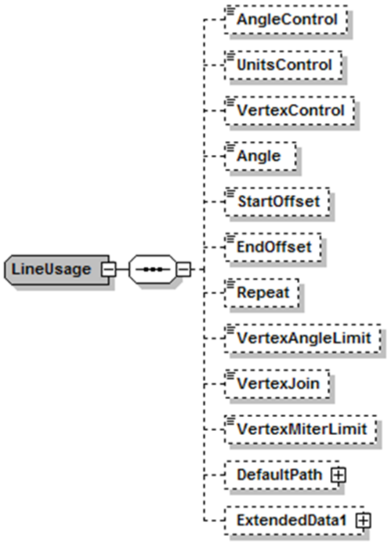
Now let’s take a look at our updated Hospital Layer Definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<LayerDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="LayerDefinition-1.1.0.xsd" version="1.1.0">
 <VectorLayerDefinition>
  <ResourceId>Library://MapsWithStyle/Redding/Basemap/Data/Hospital.FeatureSource</ResourceId>
  <FeatureName>Default:Hospital</FeatureName>
  <FeatureNameType>FeatureClass</FeatureNameType>
  <PropertyMapping>
   <Name>NAME</Name>
   <Value>NAME</Value>
  </PropertyMapping>
  <Geometry>Geometry</Geometry>
  <VectorScaleRange>
   <CompositeTypeStyle>
    <CompositeRule>
     <LegendLabel>Hospital</LegendLabel>
     <CompositeSymbolization>
      <SymbolInstance>
       <ResourceId>Library://MapsWithStyle/CartoSymbols/Hospital.SymbolDefinition</ResourceId>
       <ParameterOverrides>
       </ParameterOverrides>
      </SymbolInstance>
     </CompositeSymbolization>
    </CompositeRule>
   </CompositeTypeStyle>
  </VectorScaleRange>
 </VectorLayerDefinition>
</LayerDefinition>
```

Note that because there is no theming there is a single Composite Rule and within that a single Composite Symbolization and Symbol Instance referring to the Hospital Symbol Definition.

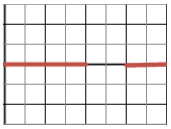
Simple Symbols and Line Usage

The Line Usage element of Simple Symbol Definition provides control over how a symbol is used to symbolize line features. The structure of the Line Usage element is shown at right. Within the Line Usage element the Angle Control element specifies how the symbol angle is defined, which must evaluate to one of: FromAngle or FromGeometry (default). The Units Control element specifies whether the distribution parameters are interpreted as Absolute (default) values (in mm) or Parametric values. The Vertex Control element specifies the symbol behavior at vertices and must evaluate to one of: OverlapNone (default), OverlapDirect, OverlapNoWrap, or OverlapWrap. The Angle element only applies if AngleControl evaluates to FromAngle and it specifies the symbol angle in degrees. The Start Offset and End Offset element specifies where the symbol distribution begins, relative to the start or end of the feature. The Repeat element allows you to repeat the rendering of a symbol along the line and specifies the separation between them in millimeters. The Vertex Angle Limit, Vertex Join, Vertex Miter Limit, and Default Path elements are not yet supported in Autodesk MapGuide 2008 and hence will not be discussed here.



Defining Line Patterns

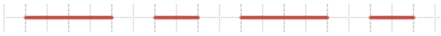
Dashed line patterns can be defined by specifying the geometry for a single complete sequence of dashing and then specifying a Repeat value within the Line Usage element. For example:



Geometry: M 0,0 H 4 Z M 6,0 H 2 Z

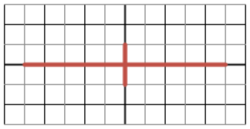
Repeat: 10

Results in:



Defining Lines with Decorations

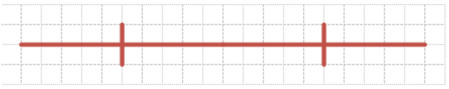
Crossing decorations can be added to line patterns by simply adding the additional geometry and repeating the pattern as before. For example:



Geometry: M -2.5,0.0 H 2.5 Z
M 0.0,0.5 V -0.5 Z

Repeat: 5

Results in:



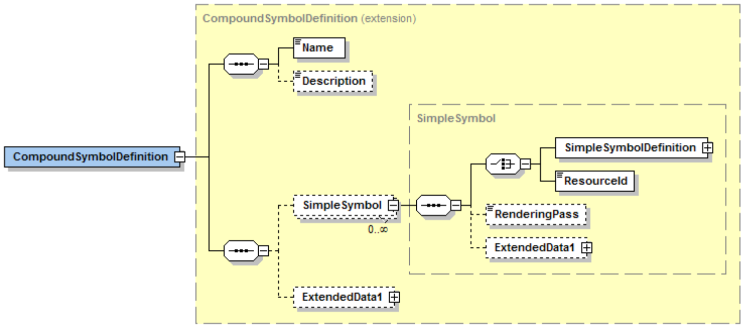
Defining the New Railroad Symbol

Now that the line usage has been explained, let's look at how to define the new railroad symbol to replace the MapGuide's standard Railroad line style.

```
<?xml version="1.0" encoding="UTF-8"?>
<SimpleSymbolDefinition version="1.0.0" xsi:noNamespaceSchemaLocation="SymbolDefinition-
1.0.0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Name>Railroad</Name>
 <Description> Railroad (single track)</Description>
 <Graphics>
  <Path>
   <Geometry>
    M -2.5,0.0 H 2.5 Z
    M 0.0,0.5 V -0.5 Z
   </Geometry>
   <LineColor>FF000000</LineColor>
   <LineWeight>0.125</LineWeight>
  </Path>
 </Graphics>
 <LineUsage>
  <VertexControl>'OverlapWrap'</VertexControl>
  <StartOffset>0</StartOffset>
  <Repeat>5.0</Repeat>
 </LineUsage>
 <ParameterDefinition/>
</SimpleSymbolDefinition>
```

Compound Symbols

Compound symbols can be used to create cased lines and to add decorative symbols at the start, end, or at repeating intervals along a line. The structure of the Compound Symbol Definition element is shown below.



Note that Compound Symbol Definitions simply allow you to aggregate multiple Simple Symbol Definitions to create a more complex symbol. The simple symbols may either be referenced by a resource identifier or may be embedded within the Compound Symbol Definition. For simplicity we'll use that approach throughout the remainder of this document, however if you are defining libraries of reusable symbols you may wish to define them independently and use references. The Rendering Pass element gives the symbol author fine grained control over the draw order of each Simple Symbol within a Compound Symbol Definition.

Defining a Freeway with a Compound Symbol

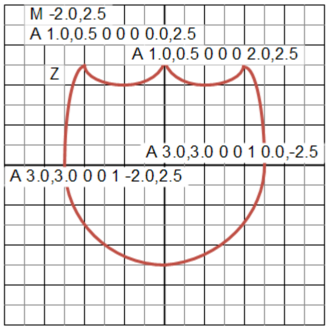
To define a cased freeway symbol we'll use a Compound Symbol Definition with two Simple Symbol Definitions embedded within it as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<CompoundSymbolDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SymbolDefinition-1.0.0.xsd" version="1.0.0">
  <Name>Freeway</Name>
  <SimpleSymbol>
    <SimpleSymbolDefinition>
      <Name>Outline</Name>
      <Graphics>
        <Path>
          <Geometry>M 0.0,0.0 H 5.0 Z</Geometry>
          <LineColor>FF000000</LineColor>
          <LineWeight>1.4</LineWeight>
        </Path>
      </Graphics>
      <LineUsage>
        <VertexControl>'OverlapWrap'</VertexControl>
        <StartOffset>0</StartOffset>
        <Repeat>5.0</Repeat>
      </LineUsage>
      <ParameterDefinition/>
    </SimpleSymbolDefinition>
    <RenderingPass>0</RenderingPass>
  </SimpleSymbol>
  <SimpleSymbol>
    <SimpleSymbolDefinition>
      <Name>Inner</Name>
      <Graphics>
        <Path>
          <Geometry>M 0.0,0.0 H 5.0 Z</Geometry>
          <LineColor>FFFF6464</LineColor>
          <LineWeight>0.8</LineWeight>
        </Path>
      </Graphics>
      <LineUsage>
        <VertexControl>'OverlapWrap'</VertexControl>
        <StartOffset>0</StartOffset>
        <Repeat>5.0</Repeat>
      </LineUsage>
      <ParameterDefinition/>
    </SimpleSymbolDefinition>
    <RenderingPass>1</RenderingPass>
  </SimpleSymbol>
</CompoundSymbolDefinition>
```

Note the RenderingPass elements clearly establish that the Outline is drawn first followed by the Inner line. This ensures that the Freeway geometry intersections render correctly.

Adding a Shield Symbol

To add a shield to our Freeway definition we'll add another Simple Symbol Definition to our Freeway Composite Symbol Definition. The shield will be composed of both a Path element to define the outline and a Text element to contain the route number. The Arc geometry for the Shield is shown to the right. The Simple Symbol XML fragment that defines our Shield is as follows:



```
<SimpleSymbol>
  <SimpleSymbolDefinition>
    <Name>Shield</Name>
    <Graphics>
      <Path>
        <ResizeControl>'AdjustToResizeBox'</ResizeControl>
        <Geometry>
          M -2.0,2.5
          A 1.0,0.5 0 0 0 0.0,2.5
          A 1.0,0.5 0 0 0 2.0,2.5
          A 3.0,3.0 0 0 1 0.0,-2.5
          A 3.0,3.0 0 0 1 -2.0,2.5
          Z
        </Geometry>
        <FillColor>FFFFFFFF</FillColor>
        <LineColor>FF0000FF</LineColor>
        <LineWeight>0.7</LineWeight>
        <LineWeightScalable>false</LineWeightScalable>
      </Path>
      <Text>
        <ResizeControl>'AddToResizeBox'</ResizeControl>
        <Content>'#'</Content>
        <FontName>'Arial'</FontName>
        <Height>4</Height>
        <PositionX>0</PositionX>
        <PositionY>0</PositionY>
        <HorizontalAlignment>'Center'</HorizontalAlignment>
        <VerticalAlignment>'Halfline'</VerticalAlignment>
      </Text>
    </Graphics>
    <ResizeBox>
      <SizeX>6</SizeX>
      <SizeY>6</SizeY>
      <PositionX>0</PositionX>
      <PositionY>0</PositionY>
      <GrowControl>'GrowInX'</GrowControl>
    </ResizeBox>
    <LineUsage>
      <AngleControl>'FromAngle'</AngleControl>
      <VertexControl>'OverlapNoWrap'</VertexControl>
      <Angle>0</Angle>
      <StartOffset>50</StartOffset>
      <EndOffset>50</EndOffset>
      <Repeat>200</Repeat>
    </LineUsage>
    <ParameterDefinition>
    </ParameterDefinition>
  </SimpleSymbolDefinition>
  <RenderingPass>2</RenderingPass>
</SimpleSymbol>
```

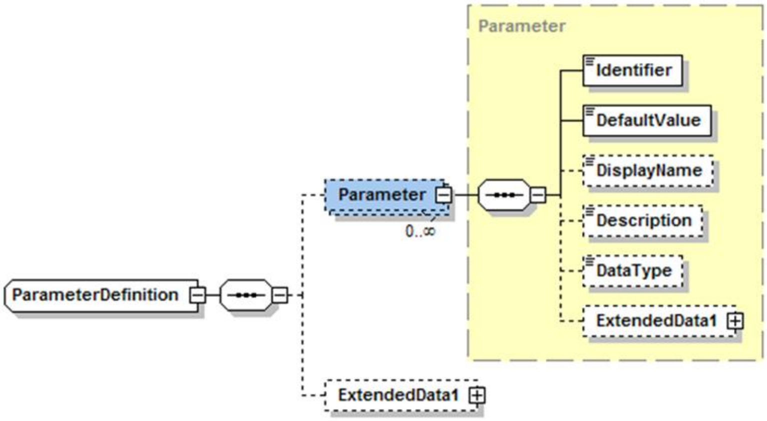
Within this fragment there are several noteworthy things. First is the use of the Resize Box and Resize Control elements. The Shields Path element sets the Resize Control value to AdjustToResizeBox while the Text element sets the Resize Control value to AddToResizeBox. Finally the Grow Control element in the ResizeBox is set to GrowInX. This allows the symbol to grow in width to adapt to longer route numbers. Second note the Start Offset, End Offset, and Repeat elements. Collectively these indicate that the shields should start 50 millimeters from the beginning of the feature geometry, repeat every 200 millimeters, and the final shield should appear 50 millimeters from the end of the feature geometry.

Data Driven Symbols

Every aspect of a Simple Symbol Definition whether standalone or as part of a Compound Symbol Definition can be driven by FDO expressions. Symbol Definitions advertize this via the

Parameter Definition element. Symbol Instances in the Layer Definition supply values for parameters via the Parameter Overrides element.

The structure of the Parameter Definition element is shown at right. Each Parameter Definition element can contain zero or more Parameter elements. Each Parameter element must supply an Identifier and a Default Value. Optionally values for Display Name, Description, and Data Type elements can be supplied. Again if you are building reusable symbol libraries this metadata can be extremely useful. The following XML fragment shows a completed Parameter Definition element:



```
<ParameterDefinition>
  <Parameter>

<Identifier>MY_PARAMETER</Identifie
r>
    <DefaultValue>0</DefaultValue>
    <DisplayName>Some
Parameter</DisplayName>
    <Description>A useful
description goes here.</Description>
    <DataType>Integer</DataType>
  </Parameter>
</ParameterDefinition>
```

Once the parameter is defined, replace the constant element values in the Symbol Definition that you want to be specified dynamically with %MY_PARAMETER% (e.g. the parameter identifier delimited by percent signs).

Going back to the Freeway symbol, we can now use a parameter to specify the route number displayed in the shield. The updated XML fragment defining the Shield symbol now looks as follows:

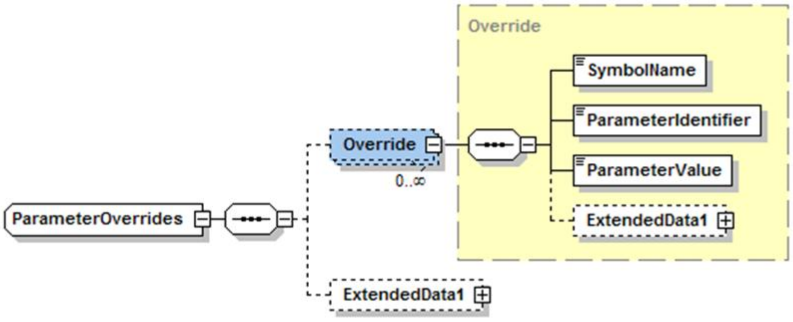
```
  <SimpleSymbol>
    <SimpleSymbolDefinition>
      <Name>Shield</Name>
      <Graphics>
        <Path>
          <ResizeControl>'AdjustToResizeBox'</ResizeControl>
          <Geometry>
            M -2.0,2.5
            A 1.0,0.5 0 0 0 0.0,2.5
            A 1.0,0.5 0 0 0 2.0,2.5
            A 3.0,3.0 0 0 1 0.0,-2.5
            A 3.0,3.0 0 0 1 -2.0,2.5
            Z
          </Geometry>
          <FillColor>FFFFFFFF</FillColor>
          <LineColor>FF0000FF</LineColor>
          <LineWeight>0.7</LineWeight>
          <LineWeightScalable>false</LineWeightScalable>
        </Path>
        <Text>
          <ResizeControl>'AddToResizeBox'</ResizeControl>
          <Content>%ROUTE_NUMBER%</Content>
          <FontName>'Arial'</FontName>
          <Height>4</Height>
          <PositionX>0</PositionX>
          <PositionY>0</PositionY>
          <HorizontalAlignment>'Center'</HorizontalAlignment>
          <VerticalAlignment>'Halfline'</VerticalAlignment>
        </Text>
      </Graphics>
      <ResizeBox>
        <SizeX>6</SizeX>
        <SizeY>6</SizeY>
        <PositionX>0</PositionX>
        <PositionY>0</PositionY>
        <GrowControl>'GrowInX'</GrowControl>
      </ResizeBox>
      <LineUsage>
        <AngleControl>'FromAngle'</AngleControl>
        <VertexControl>'OverlapNoWrap'</VertexControl>
        <Angle>0</Angle>
        <StartOffset>50</StartOffset>
        <EndOffset>50</EndOffset>
        <Repeat>200</Repeat>
      </LineUsage>
      <ParameterDefinition>
        <Parameter>
          <Identifier>ROUTE_NUMBER</Identifier>
          <DefaultValue>0</DefaultValue>
          <DisplayName>Route Number</DisplayName>
          <Description>The interstate number.</Description>
          <DataType>Integer</DataType>
        </Parameter>
      </ParameterDefinition>
    </SimpleSymbolDefinition>
    <RenderingPass>2</RenderingPass>
  </SimpleSymbol>
```

Note the ROUTE_NUMBER parameter is defined and the Content element of the Text Geometry is now set to %ROUTE_NUMBER%.

Now to set the value of the expression a Parameter Override needs to be specified within the Symbol Instance element in the Layer Definition. The structure of the Parameter Overrides element is shown at right. Note that the Parameter Overrides element contains zero or more Override element. Each Override element specifies an expression to use for a single parameter of a specific symbol. The following XML fragment shows a completed Parameter Overrides element:

```
<ParameterOverrides>
  <Override>
    <SymbolName>MySymbol</SymbolName>
    <ParameterIdentifier>
       MY_PARAMETER
    </ParameterIdentifier>

<ParameterValue>“MY_VALUE_FIELD"<
/ParameterValue>
  </Override>
</ParameterOverrides>
```



This fragment will replace the %MY_PARAMETER% values in a symbol called MySymbol with the value from the MY_VALUE_FIELD property of the layers FDO feature class. To make full use of the Freeway Symbol Definition, the completed Roads Layer Definition looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<LayerDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="LayerDefinition-1.1.0.xsd" version="1.1.0">
 <VectorLayerDefinition>
  <ResourceId>Library://MapsWithStyle/Redding/Basemap/Roads.FeatureSource</ResourceId>
  <FeatureName>Default:Roads</FeatureName>
  <FeatureNameType>FeatureClass</FeatureNameType>
  <Geometry>Geometry</Geometry>
  <VectorScaleRange>
   <MaxScale>15000</MaxScale>
   <CompositeTypeStyle>
    <CompositeRule>
     <LegendLabel>'ART'</LegendLabel>
     <Filter>"ST_TYPE" = 'ART' OR "ST_TYPE" = 'COL'</Filter>
     <CompositeSymbolization>
      <SymbolInstance>
<ResourceId>Library://MapsWithStyle/CartoSymbols/ArterialCollector.SymbolDefinition</ResourceId>
       <ParameterOverrides>
       </ParameterOverrides>
      </SymbolInstance>
     </CompositeSymbolization>
    </CompositeRule>
    <CompositeRule>
     <LegendLabel>'FWY'</LegendLabel>
     <Filter>"ST_TYPE" = 'FWY'</Filter>
     <CompositeSymbolization>
      <SymbolInstance>
       <ResourceId>Library://MapsWithStyle/CartoSymbols/Freeway.SymbolDefinition</ResourceId>
       <ParameterOverrides>
        <Override>
         <SymbolName>Shield</SymbolName>
         <ParameterIdentifier>ROUTE_NUMBER</ParameterIdentifier>
         <ParameterValue>"RTNUM"</ParameterValue>
        </Override>
       </ParameterOverrides>
      </SymbolInstance>
     </CompositeSymbolization>
    </CompositeRule>
    <CompositeRule>
     <LegendLabel>'HWY'</LegendLabel>
     <Filter>"ST_TYPE" = 'HWY'</Filter>
     <CompositeSymbolization>
      <SymbolInstance>
       <ResourceId>Library://MapsWithStyle/CartoSymbols/Highway.SymbolDefinition</ResourceId>
       <ParameterOverrides>
        <Override>
         <SymbolName>Shield</SymbolName>
         <ParameterIdentifier>ROUTE_NUMBER</ParameterIdentifier>
         <ParameterValue>"RTNUM"</ParameterValue>
        </Override>
       </ParameterOverrides>
      </SymbolInstance>
     </CompositeSymbolization>
    </CompositeRule>
    <CompositeRule>
     <LegendLabel></LegendLabel>
     <CompositeSymbolization>
      <SymbolInstance>
       <ResourceId>Library://MapsWithStyle/CartoSymbols/Street.SymbolDefinition</ResourceId>
       <ParameterOverrides>
       </ParameterOverrides>
      </SymbolInstance>
     </CompositeSymbolization>
    </CompositeRule>
   </CompositeTypeStyle>
  </VectorScaleRange>
 </VectorLayerDefinition>
```

```
</LayerDefinition>
```

The roads layer is themed based upon the ST_TYPE property of the Roads feature class. Note that both ST_TYPE = FWY and ST_TYPE = HWY use parameterized Symbol Definitions that display the route number.

Summary

This course has provided an overview of the new cartographic model in Autodesk MapGuide Enterprise 2008. While no support exists in the user interface of Autodesk MapGuide Studio 2008, you can use the techniques shown here to create a library of custom symbols and use those in your MapGuide applications.