Interpreting GeoSVG Output

1 Returned SVG Layout

```
<svg width="width" height="height" preserveAspectRatio="none" xmlns="http://www.w3.org/2000/svg" version="1.1">
  <metadata>
    Metadata explained below
  </metadata>
  <g transform="translate(x,y)">
    SVG elements
  </g>
</svg>
```

2 Metadata

The `<metadata>` tag contains information to help plot the SVG on a map. Either the four corners (`geoTL`, `geoBR`, `geoTR`, and `geoBL`) or `north`, `south`, `east`, and `west` latitudes and longitudes can be used for coordinates. Regardless of the symbol's shape the returned bounds will be a rectangle.

Element	type/format	Value
id	string	User defined from input
name	string	User defined from input
description	string	User defined from input
symbolID	string	20-30 digit code corresponding to a symbol
geoTL	number number	Top left coordinate point to place SVG
geoBR	number number	Bottom right coordinate point to place SVG
geoTR	number number	Top right coordinate point to place SVG
geoBL	number number	Bottom left coordinate point to place SVG
north	number	Latitude to place top edge of SVG
south	number	Latitude to place bottom edge of SVG
east	number	Longitude to place right edge of SVG
west	number	Longitude to place left edge of SVG
wasClipped	boolean	True if the rendered symbol fit inside the bounding box
width	number	Pixel width of image
height	number	Pixel height of image

3 Example SVG Output

```
WebRenderer.RenderSymbol2D("id", "Base Camp", "Control Measure / Command and Control Areas",
"11032500001205000000000000000000", "49.3,19.8 49.2,19.4 49.6,19.4 49.7,19.8", 1200, 900, "49.0,19.0,50.0,20.0",
new HashMap<>(), new HashMap<>(), WebRenderer.OUTPUT_FORMAT_GEOSVG);
```

```
<svg width="604.0px" height="364.0px" preserveAspectRatio="none" xmlns="http://www.w3.org/2000/svg" version="
1.1">
  <metadata>
    <id>id</id>
    <name>Base Camp</name>
    <description>Control Measure / Command and Control Areas</description>
    <symbolID>11032500001205000000000000000000</symbolID>
    <geoTL>49.19833333333333 19.802222222222223</geoTL>
    <geoBR>49.70166666666667 19.397777777777776</geoBR>
    <geoTR>49.70166666666667 19.802222222222223</geoTR>
    <geoBL>49.19833333333333 19.397777777777776</geoBL>
    <north>19.802222222222223</north>
    <south>19.397777777777776</south>
    <east>49.70166666666667</east>
    <west>49.19833333333333</west>
    <wasClipped>false</wasClipped>
    <width>604.0</width>
    <height>364.0</height>
  </metadata>
  <g transform="translate(-238.0,-178.0)">
    <path d="M360.0 180.0L240.0 540.0L720.0 540.0L840.0 180.0L360.0 180.0" stroke="#000000" stroke-width="3"
stroke-linecap="round" fill="none" />
    <text x="540.0" y="366.0" font-family="arial" font-size="12px" font-weight="bold" alignment-baseline="
middle" stroke-miterlimit="3" text-anchor="middle" stroke="#FFFFFF" stroke-width="3.0" fill="none">BC</text>
    <text x="540.0" y="366.0" font-family="arial" font-size="12px" font-weight="bold" alignment-baseline="
middle" stroke-miterlimit="3" text-anchor="middle" fill="#000000">BC</text>
  </g>
</svg>
```

4 Plotting SVG symbols in Cesium

The following function is an example of how to plot the SVG output with cesium

```
function renderMPSVG(svg: string, viewer: Cesium.Viewer) {
  if (svg.includes('"type":"error"')) {
    const errorJSON = JSON.parse(svg);
    console.log(errorJSON.error);
    return;
  } else if (svg == '<svg width="2px" height="2px" xmlns="http://www.w3.org/2000/svg" version="1.1"></svg>') {
    console.log("Received empty SVG");
    return;
  }

  let north = parseFloat(svg.substring(svg.indexOf("<north>") + "<north>".length, svg.indexOf("</north>")));
  let south = parseFloat(svg.substring(svg.indexOf("<south>") + "<south>".length, svg.indexOf("</south>")));
  let east = parseFloat(svg.substring(svg.indexOf("<east>") + "<east>".length, svg.indexOf("</east>")));
  let west = parseFloat(svg.substring(svg.indexOf("<west>") + "<west>".length, svg.indexOf("</west>")));

  // Safely encode SVG
  svg = "" + btoa(String.fromCodePoint(...new TextEncoder().encode(svg)))

  viewer.entities.add({
    rectangle: {
      coordinates: Cesium.Rectangle.fromDegrees(west, south, east, north),
      material: new Cesium.ImageMaterialProperty({ image: svg }),
    },
  });
}
```