EMSA

European Maritime Safety Agency

Appendix F
Sample Symbology Configuration Files

1 Technical Elements

The symbolyzer is based on the following architecture.

- Each symbolizer is mapped into a unique configuration file, which can be reused in many different contexts
- For some layers, it will be possible to specify multiple symbolizers (e.g. in thec ase of the default configuration for vessel tracks, it will be possible to configure IALA, Pirasat, etc.), so that the user can interactively choose the symbol set and switch into it automatically
- A symbolizer is composed by a name and by one or many geometry types (point, lines, polygons)
- Each geometry type is defined a ***style***
- The style is characterised by the definition of a number of parameters described hereafter
- Each parameter of the style can be defined in terms of:
    - A fixed value, syntax *parameter* : *value*
    - A attribute dependent value, syntax *parameter* : *"${ entity.property }"*
    - A value depending on a more complex function, called ***helper***, syntax *parameter* : *"${ ~functionname}"*
- All necessary ***helpers*** are defined in the configuration file, in terms of Javascript functions

Therefore the elements of the symbolyzer configuration file are.

Name	The name of the symbology
Geometry_symbolizer	It can be point/line/polygon

Table 1-1 Main elements of the symbolizer configuration file

The geometry_symbolizer in turn is composed by the following elements.

Symbols	An array of the following couple of key-value: • Type: 0 = no overlay; 1 = overlay • URL: patch to the symbol png
Helpers	A list of functions that are pointed to by the style definition
style	Style definition (see tables below)
legendHTML	A function that creates dynamically an HTML legend

Table 1-2 Description of the geometry_symbolizer

The **point symbolizer style** is defined by the following attributes.

symbol_idx	The symbol to be used as a basis for drawing (see e.g **Error! Reference source not found.**).
fill_color	The rule to be used for filling in the colour
Edge_color	The rule to be used for the edge color
Secondary_color	The rule to be used for the secondary color
Overlay_idx	Pointer to a symbol to be used for overlay
rotation	Rule for the symbol rotation
visible	Rule for defining the visibility

Table 1-3 Attributes of the point_symbolizer

The **line and polygons symbolizer styles** are defined by the following attributes.

line_color	The rule to be used for the line colour

Table 1-4 Attributes of the line_symbolizer

1.1. Supported helpers functions

Most of the complexity and richness of the symbolizer file relies in the helpers functions, that are used for generating dynamically the behaviour of the object to be drawn.

A few examples from the IALA definition will help understanding the capabilities of such approach. Here follows an extract from the configuration file to be used for the IALA.

The following helper is used for getting the symbol to be used.

```
getSymbol: function(geometry) {
        var idx = geometry.entity.position.heading !== undefined ? 0 : 1;
        return idx
},
```

Basically if the attribute heading of the position of the entity (in this case the entity is the vessel to be drawn) is defined it will use symbol 0 (isosceles triangle), else will use symbol 1 (equilateral triangle). Symbols are defined as in Table 1-2, e.g. an array pointed by the result of this function.

The following reports an example of how to set the fill colour in IALA. If the position has NPR enrichment (npre) and if it is alarmed, then the fill colour is red, else is black.

```
getFillColor: function(geometry) {
        var color;
        if (geometry.entity.position.npre & acs.imdate.Vessel.ALARMED) {
                color = [1.0, 0.0, 0.0]
        }
        else {
                color = [0.0, 0.0, 0.0]
        }
        return color;
}
```

Slightly more complex is the case of defining the overlay symbol, on the basis of some characteristics of the objects. The function below defines the logic on the basis of which the different overlay symbols are chosen. Symbols are defined as in Table 1-2, e.g. an array pointed by the result of this function.

```
getOverlay: function(geometry) {
        var entity = geometry.entity;
        var npre = entity.position.npre;
        var idx = -1;

        if (npre & acs.imdate.Vessel.ALARMED) {
```

```
                if (npre & acs.imdate.Vessel.HAZMAT) {
                        idx = 2;
                }
        }
        else {
                if (npre & acs.imdate.Vessel.BANNED) {
                        idx = 3;
                }
                else if (npre & acs.imdate.Vessel.HAZMAT) {
                        idx = 2;
                }
        }
        return idx;
},
```

Another interesting example is related to the helper for the definition of the fill colour for fishing vessels, as used in the BFT project. In this case, the color depends on the attribute CATCH_PERMIT and on its specific expiration date. Moreover it will depend on the ship type.

```
getFillColor: function(geometry) {
        var vessel = geometry.entity;
        var iccat_info = vessel.project_spec_info;
        var color = [0.0, 0.0, 0.0];

        if (iccat_info !== undefined) {
                if (iccat_info.CATCH_PERMIT == 'FALSE') {
                        color = [1.0, 0.0, 0.0];
                }
                else {
                        var pos_ts = Date.parseString(vessel.position.ts, 'yyyy-MM-
ddTHH:mm:ssZ').getTime();
                        var permit_from_ts = Date.parseString(iccat_info.PERMIT_FROM,
'yyyy-MM-dd').getTime();
                        var permit_to_ts = Date.parseString(iccat_info.PERMIT_TO, 'yyyy-
MM-dd').getTime();
                        if (permit_from_ts > pos_ts || permit_to_ts < pos_ts) {
                                color = [1.0, 0.0, 0.0];
                        }
                        else {
                                color = [0.0, 1.0, 0.0];
                        }
                }
        }
        else {
                var ovr_info = vessel.ovr_info;
                if ((ovr_info && ovr_info.ship_type == 30) || (vessel.position.ns == 7)) {
                        color = [1.0, 0.5, 1.0];
```

```
                }
        }

        return color;
},
```

1.1 Example of configuration file for the IALA symbolizer

```
{
        name: "IALA",
        point_symbolizer: {
                symbols: [
                        {
                                type: 0,
                                url: "symbolizers/images/iala_iso_triangle.png"
                        },
                        {
                                type: 0,
                                url: "symbolizers/images/iala_equi_triangle.png"
                        },
                        {
                                type: 1,
                                url: "symbolizers/images/iala_haz.png"
                        },
                        {
                                type: 1,
                                url: "symbolizers/images/iala_ban.png"
                        }
                ],
                helpers: {
                        getSymbol: function(geometry) {
                                var idx = geometry.entity.position.heading !== undefined ? 0 : 1;
                                return idx
                        },
                        getFillColor: function(geometry) {
                                var color;
                                if (geometry.entity.position.npre & acs.imdate.Vessel.ALARMED) {
                                        color = [1.0, 0.0, 0.0]
                                }
                                else {
                                        color = [0.0, 0.0, 0.0]
                                }
                                return color;
                        },
                        getEdgeColor: function(geometry) {

                                var color = [1.0, 1.0, 1.0];
                                var entity = geometry.entity;
                                var npre = entity.position.npre;

                                if (entity.display_props.highlighted) {
                                        color = [1.0, 1.0, 0.0]
                                }
                                else if (npre !== undefined){
                                        if (npre & acs.imdate.Vessel.ALARMED) {
                                                color = [0.0, 0.0, 0.0];
                                        }
                                        else if ((npre & acs.imdate.Vessel.SINGLE_HULL) && !(npre &
acs.imdate.Vessel.BANNED)) {
                                                color = [1.0, 0.0, 0.0];
                                        }
                                }
                                return color;
                        },
```

```
                                getOverlay: function(geometry) {
                                        var entity = geometry.entity;
                                        var npre = entity.position.npre;
                                        var idx = -1;

                                        if (npre & acs.imdate.Vessel.ALARMED) {
                                                if (npre & acs.imdate.Vessel.HAZMAT) {
                                                        idx = 2;
                                                }
                                        }
                                        else {
                                                if (npre & acs.imdate.Vessel.BANNED) {
                                                        idx = 3;
                                                }
                                                else if (npre & acs.imdate.Vessel.HAZMAT) {
                                                        idx = 2;
                                                }
                                        }

                                        return idx;
                                },
                                getSecondaryColor: function(geometry) {
                                        var entity = geometry.entity;
                                        var npre = entity.position.npre;

                                        var color = [0.0, 0.0, 0.0];
                                        if (npre & acs.imdate.Vessel.ALARMED || ((npre & acs.imdate.Vessel.SINGLE_HULL) && !(npre
& acs.imdate.Vessel.BANNED))) {
                                                color = [1.0, 0.0, 0.0]
                                        }

                                        return color;
                                }
                        },
                        style: {
                                symbol_idx: "${~getSymbol}",
                                fill_color: "${~getFillColor}",
                                edge_color: "${~getEdgeColor}",
                                secondary_color: "${~getSecondaryColor}",
                                overlay_idx: "${~getOverlay}",
                                rotation: "${entity.position.heading}",
                                visible: "${entity.display_props.visible}"
                        }
                },
                legendHTML: function(onReady) {

                        var content = "<div style=\"float:left\"><table class=\"legend_table\">";
                        content += "<tr><th colspan=\"2\" align=\"left\">Color (Source/Alarm)</th></tr>";
                        var sources = [
                                {color: [0.0, 0.0, 0.0], desc: "AIS Data"},
                                {color:      [0.5, 0.5, 0.5], desc: "LRIT Data"},
                                {color: [1.0, 1.0, 0.0], desc: "VMS Data"},
                                {color: [1.0, 0.0, 0.0], desc: "Alarmed"}
                        ];

                        for (src_id in sources) {
                                var src = sources[src_id];
                                var hex_color = acs.imdate.MapSymbolizer.GLToCSSColor(src.color)
                                var color_cell = "<td width=\"13px\" bgcolor=\""+ hex_color + "\" ></td>";
                                content += ("<tr>" + color_cell + "<td>" + src.desc + "</td></tr>");

                        }

                        content += "</table></div>";

                        content += "<div style=\"float:left\"><div><table class=\"legend_table\">";
                        content += "<tr><th colspan=\"2\" align=\"left\">Shape (Heading)</th></tr>";


                        var symbol_cell = "<td><img width='24' height='24' src='images/isosceles.png' /></td>";
                        content += ("<tr>" + symbol_cell + "<td>Position with heading</td></tr>");
```

```
                    symbol_cell = "<td><img width='24' height='24' src='images/triangle.png' /></td>";
                    content += ("<tr>" + symbol_cell + "<td>Position with no heading</td></tr>");

                    content += "</table></div>";

                    content += "<div><table class=\"legend_table\">";
                    content += "<tr><th colspan=\"2\" align=\"left\">Overlay (Risk Status)</th></tr>";

                    symbol_cell = "<td><img width='16' height='16' src='images/diamond.png' /></td>";
                    content += ("<tr>" + symbol_cell + "<td>Hazardous Materials</td></tr>");
                    symbol_cell = "<td><img width='24' height='24' src='images/red_border.png' /></td>";
                    content += ("<tr>" + symbol_cell + "<td>Single Hull</td></tr>");
                    symbol_cell = "<td><img width='24' height='24' src='images/red_cross.png' /></td>";
                    content += ("<tr>" + symbol_cell + "<td>Banned</td></tr>");

                    content += "</table></div></div>";

                    onReady(content);

            }
}
```