Theme Customization
(User’s Guide)

Table of Contents

Introduction

This document describes the various customizations options that are available to you, as a TrueVector user, to customize the look and feel of your product. TrueVector comes with a complete suite of recoloring, skinning, theming and branding options that can be as broad as altering every polygonal object displayed, or as detailed as specifying the border color on a single item.

Before reading this document you should:

- **Have a standard text editor (not MS Word) ready**
- **Be at least familiar with the basics of editing text files and XML.**
- **Be familiar with HTMLformatted hexadecimal representations of colors – The 6 character RRGGBB notation. ActionScript's RGB values are identical in every way to HTML's RGB values except for one difference – Instead of being preceeded by a # sign, they are preceeded by the characters 0x. That is, a number 0, not an upper case letter O. In this way, pure blue in HTML, which would be represented by `#0000FF` is represented in ActionScript by `0x0000FF`. Note: All colors used in the configuration files must be of this format, otherwise TrueVector may not render them correctly.**

All of TrueVector's theme customization options are stored in readily accessible XML files. These are called configuration files. All you need to edit these files is a standard text editor like notepad, although we recommend a more advanced editor like UltraEdit. Since some of the XML nodenames have uppercase letters in them or are numeric, a "standard" XML editor will probably reject the XML as noncompliant. This is true. TrueVector's XML is not designed to be standards compliant, since it is intended to be read only by TrueVector itself. Changing the way TrueVector looks and feels onscreen is as simple as changing the settings in the XML and reloading TrueVector in your web browser.

Please remember, editing these configuration files is entirely optional, and entirely at your own risk. At TrueVectorTech, we make every effort to ensure that TrueVector's initial default theme and colorization is attractive and functional before we ship this product to you. Always remember to make a backup of any file you are about to edit. That way, you can replace it easily, should the edits not work the way you expected.

TrueVector's theming and colorization can be broken down into three main categories: General Theme Customization, Dataset Symbolization and Feature Symbolization. At first glance, the sheer range of configuration and customization options available to you might seem daunting, but as you discover the flexibility of the system, changing a color will become intuitive.

General Theme Customization

TrueVector reads its general theme customization instructions from the file `theme.xml`, which in a default setup is placed inside the `config` folder.

In standard XML fashion, it is broken down into a number of nested sections, which are described below:

- mainBackground: This section contains configuration options for the background image.
- symbolizationData: This section contains display rules for each type of geometry.

mainBackground

mainBackground has 4 options, 1 of which is not currently supported.

- scaleBackground
  - Valid values: true, false
  - Function: Tells TrueVector whether or not to scale the background when zooming in or out. This option has no effect if there is no background image set
- bgColor
  - Valid values: Any valid ActionScript formatted hexadecimal RGB value
  - Function: The color of the background. The background is only shown if there is no background image set.
- bgImg
  - Valid values: The path, relative to the location of the TVFM .swf file of a jpg, png or gif raster image.
  - Function: The image to load in the background of TrueVector. For best results, it is recommended that this image be the same size as TrueVector itself.
- bgAlign
  - Valid values: None (not yet supported)
  - Function: This will control the alignment of the background image. Valid values will be the standard alignment ones – center, stretch, tile, etc.

symbolizationData

symbolizationData has 4 main sections, one for each type of object shown – Points, lines, text objects and polygons. Each type of object requires specific display instructions. The options here represent the default display instructions for each type of geometry at each zoom level. Each geometry type section is broken down into zoom level sections, and each zoom level section has its own display rules.

polygonSymbol

polygonSymbols contain display instructions for polygonal objects within TrueVector, such as the major clickable geographic regions like States or Counties. Their options are:

- fillColor: The color to display the filledin part of the polygon. This has a range from 1 to 6 because polygons can potentially have a color ID assigned to them from 16.
- highlightColor: The color that the polygon's fill changes to when the mouse is moved over it. It is not possible to disable the color changing functionality of TrueVector on a roll over.
- selectColor: The color that the polygon's fill changes to when it is selected. Objects that are selected do not respond to the mouse rolling over them.
- blurColor: This is the color that the polygon changes to when it is not in focus – That is, when we are at a zoom level other than the one in which this polygon was drawn.
- strokeColor: This is the color of the line around the polygonal object.
- blurAlpha: This is the opacity of the object when it is not in focus. 1100, 100 being fully opaque.
- focusAlpha: This is the opacity of the object when it is in focus. 1100,

100 being fully opaque.
- **selectAlpha: This is the opacity of the object when it is selected. 1100,**

100 being fully opaque.
- **highlightAlpha: This is the opacity of the object when it is highlighted**

lineSymbol
lineSymbols contain display instructions for line objects within TrueVector, such as roads. Their options are:
- **strokeColor: The color of the line**
- **strokeAlpha: The opacity of the line. 1100,**

100 being fully opaque.
- **strokeWidth: The width, in pixels of the line**

pointSymbol
pointSymbols contain display instructions for point objects within TrueVector, such as city dots or interstate
shields. Their options are:
- **fillColor: The color of the symbol, if this symbol is a geometric symbol such as a circle.**
- **fillAlpha: The opacity of the fill of the symbol. 1100,**

100 being fully opaque.
- **strokeWidth: Currently unsupported. The width of the line surrounding the symbol, in pixels**
- **rotation: Currently unsupported. The rotation, in degrees, of the symbol.**
- **symbolType: The type of symbol to draw. Currently supported types are circle, interstate, highway.**

Support for other geometric shapes such as square, star, etc will be supported in future versions of
TrueVector
- **strokeAlpha: Currently unsupported. The opacity of the line surrounding the geometric symbol. 1100,**

100 being fully opaque.

- **size: The size of the symbol. This has different meanings depending on the type of symbol.**
- **strokeColor. Currently unsupported. The color of the line surrounding the symbol**

textSymbols
textSymbols contain text formatting instructions for text objects within TrueVector such as labels. A City point in
TrueVector is comprised of both a point symbol and a text symbol, and takes options from both.
- **size: Size, in pts of the font to use**
- **bold: true or false. Whether or not to bold the text.**
- **font: Any valid Flash font.**
- **italic: true or false. Whether or not to italicize the text.**
- **color: The color of the text.**

Dataset Symbolization

In the GIS (Geographic Information Systems) world – the world of digital mapping and cartography, where
TrueVector was born – any object that is placed on a map is known as a "feature." This includes roads, ZIP

Codes, bodies of water, street names, anything. In TrueVector, these "features" are grouped into "layers" of a similar geometry type. These layers are what the General Theme Customization controls available to you in `theme.xml` allow you to control.

However, within TrueVector these "layers" are further subdivided by data type. For example, while both a city point and an interstate shield are of the geometric type "point" and as such, both are included in the point layer, they are of different data types and drawn from different datasets. They require different display instructions. To take the most basic example: Normally city points are circles, where as an interstate shield is most definitely not.

So with that in mind, TrueVector also gives you the ability to control the display instructions for geometry types by dataset only. This is a slightly more advanced means of customization and should only be attempted if you are familiar with XML.

The Dataset Symbolization XML, which is what we're about to edit is contained in the folder `symbolization_xml`. There is one XML file in that folder for each zoom level in your map. To change the look and feel of an dataset's features without changing others of the same geometric type, you first need to find which dataset to edit. The names of the datasets used in your project are listed in the `datasets.txt` document which should live in the same folder as this document, the `docs` folder.

Sample `datasets.txt` contents:

```
us_hwy_i_proj_polyline – Data for Interstates shown at State level
us_final_proj_region – Polygonal geometry for US States.
us_county_fake_region – Polygonal geometry for US Counties.
us_topstatecities_point – Cities shown at State level.
```

Therefore, to change the display options for a particular dataset, you need only find the correct dataset for the items you wish to change. You would then find the correct xml file in the `symbolization_xml` folder that refers to the zoom level where you wish the changes to appear. These files are numbered from zero, where zero is the initial starting zoom level of the map (The view showing the entire US). Note: If your map is a singlestate version of TrueVector, it is highly likely that it initially draws the entire US, and then pre zooms to your state before removing the preloading screen. In this case, your initial zoom level is 1, not

**0, and the initial file to edit would therefore be `sym_1.xml` not `sym_0.xml`. So the file that contains the datasetspecific
display information for the state level view (zoom level 1) would be `sym_1.xml`. A sample `sym_1.xml` is given on the next page:**

m .

Sample sym_1.xml:

```
<?xml version='1.0' encoding='UTF8'?>
<symbolization>
<dataset id="us_shld_st_i_proj_font_point">
<textSymbol>
<size>10</size>
<bold>false</bold>
<font>arial_embed</font>
<italic>true</italic>
<color>0xFFFFFF</color>
</textSymbol>
<pointSymbol>
<fillColor>0x000000</fillColor>
<fillAlpha>100</fillAlpha>
<lineWidth>0</lineWidth>
<rotation>0</rotation>
<symbolType>interstate</symbolType>
<lineAlpha>0</lineAlpha>
<size>1</size>
<lineColor>0x000000</lineColor>
</pointSymbol>
</dataset>
<dataset id="us_topstatecities_point">
<textSymbol>
<size>10</size>
<bold>false</bold>
<font>arial_embed</font>
<italic>true</italic>
<color>0x011A20</color>
</textSymbol>
<pointSymbol>
<fillColor>0x011A20</fillColor>
<fillAlpha>100</fillAlpha>
<lineWidth>0</lineWidth>
<rotation>0</rotation>
<symbolType>circle</symbolType>
<lineAlpha>0</lineAlpha>
<size>2</size>
<lineColor>0x000000</lineColor>
</pointSymbol>
</dataset>
```

```
<dataset id="us_hwy_i_proj_polyline">
<lineSymbol>
<strokeColor>0x105692</strokeColor>
<strokeAlpha>100</strokeAlpha>
<strokeWidth>2</strokeWidth>
</lineSymbol>
</dataset></symbolization>
```

As can be seen, each dataset to be modified is listed in turn, and inside each dataset is placed display information for each of the geometric types of object that that dataset contains. Remember, a label on a map contains both a text symbol object and a point symbol object, and so display instructions for both must be included when controlling the look and feel of labels.

The parameters inside each geometric section are identical to the parameters for the corresponding geometric type outlined earlier in this document. You can either edit the parameters that already exist in the relevant sym_x.xml file yourself, or you can create entirely new dataset sections if there are datasets that do not currently have any display information included in the file.

Feature Symbolization

Features, remember, are the individual items drawn on a map. TrueVector also gives you the opportunity to enter individual feature display instructions. This information will be stored in the same file as the dataset specific display instructions.

To change the look and feel of an individual feature without changing those of the same geometric type, or even those in the same dataset, you first need to find which feature to reference. Every single item drawn in your TrueVector interactive map has an individual and unqiue 6digit ID number. These numbers are held in the geometric XML, so to find the correct dataset, you need to understand a little about how TrueVector's geometric XML is structured.

The geometric XML is split up both by zoom level and by parent object. That is, the child regions (for example, counties) of the state "Virginia " have an XML file of their own and the child regions for Amelia

county in Virginia
(for example, ZIP Codes) also have a file of their own. Inside the file for Amelia County will be the XML
necessary to draw every single item that you see when zoomed into Amelia County.
A sample of this XML is shown below:


```
<feature oid="102928" state="51" county="005" name="Alleghany" ...
<feature oid="102929" state="51" county="007" name="Amelia" ...
<feature oid="102930" state="51" county="009" name="Amherst" ...
<feature oid="102931" state="51" county="011" name="Appomattox" ...
<feature oid="102932" state="51" county="013" name="Arlington" ...
<feature oid="102933" state="51" county="015" name="Augusta" ...
<feature oid="102934" state="51" county="017" name="Bath" ...
<feature oid="102935" state="51" county="019" name="Bedford" ...
```

This is an excerpt from the state file (i.e. the file containing the counties) for the state of Virginia. You can see
that each county has its own section, and each county has its own id. In TrueVector, we call it oid, for Object ID.
If you wanted to enter specific display instructions for Amelia County, for example, you would use the oid
102929. Since this is for the counties within the state of Virginia, we are obviously at zoom level 1. Therefore,
the file you would edit would be `sym_1.xml`.
A sample entry into `sym_1.xml` to change the look of Amelia County would look something like this:

```
<feature id="102929">
<polygonSymbol>
<fillColor>0xFF0000</fillColor>
<highlightColor>0x00FF00</highlightColor>
<selectColor>0x0000FF</selectColor>
<blurColor>0x000000</blurColor>
<strokeColor>0x000000</strokeColor>
<blurAlpha>50</blurAlpha>
<focusAlpha>99</focusAlpha>
<selectAlpha>99</selectAlpha>
<highlightAlpha>99</highlightAlpha>
</polygonSymbol>
</feature>
```

Once again, the parameters entered into the section are the exact same ones for the specific geometry type of
the object you want to alter, in this case, a polygon. However, not all parameters are necessary. TrueVector's
rendering rules work on a perparameter

basis, so you need only enter the parameters that you wish to be different from the default.

You may enter as many feature definitions as you wish into the symbolization xml files, simply list them one after the other like so:

```
<feature id="102929">
...
</feature>
<feature id="102929">
...
</feature>
<feature id="102929">
...
```

```
</feature>
```


Summary

In summary, then, TrueVector's customization and theming options are threetiered, each tier providing a level of control slightly higher than the last, and introducing a corresponding level of complexity.

- **General Theme Customization allows you to set the general look and feel of each type of object inside each zoom level.**
- **Dataset Symbolization allows for a higher degree of control, by giving you the ability to set the look and feel for each object within a specific dataset.**
- **Feature Symbolization is the most precise level of control available, allowing you to customize the look, feel and behaviour of individual objects.**

As you might expect each tier of customization overrides the previous tier, so if the General Theme customization decrees that all polygons are white, then all polygons will be white. However, if the specific Dataset Symbolization instructions read that states are blue, then all polygons will be white, except for states, which will be blue. Then finally, if the Feature Symbolization says that Virginia will be colored black, then all the polygons will be white, except for states, which will be blue, except for Virginia, which will be black.

Editing TrueVector's default theme and colorization scheme can potentially be a complex task. It is complex because it is flexible. However, it need only be as complex as you want it to be. We strive to ship TrueVector with an attractive and functional intial theme, usually echoing your website's own color scheme, or a color scheme previously agreed upon with you, but we do understand that there are times when that may

need to change – If you update your website, for example. It is because of those times that we have put a lot of effort into making the colors and theme of your TrueVector Flash Map as configurable and customizable as possible.

We hope that you will enjoy experimenting with the thematic configuration options in your TrueVector product and remember – Always make a back up or a copy. If you back up the files before you edit them, you can always return to a last known working state and try again.