# Track Paper Widget Library

This is a brief overview of how to implement the track paper widget.

## Overview

The track paper widget library provides a convenient wrapper to automatically build the SVG strings for an entire track system for the paper widget. The library utilizes the prebuilt SVG from the track system (post processed) in combination with acp trak status blocks to build a simple drop in widget to show the status of an acp trak system.

# Major Components

The TrakPaper widget consists of a few components.

1. TrakPaper Library
2. TrakDiag Task
3. SegmentStatus Custom Widget
4. MpPaper widget
5. ACOPOS Trak layout generated SVG

Each of these components works together to create the TrakMap widget and functionality.

## Usage

### Import the TrakMap Technology Solution

Place the provided technology solution in the Automation Studio technology solutions folder (BrAutomationInstallationFolder)/(AutomationStudioVersion)/AS/TechnologySolutions. In your project, select the root node in the logical view, in the toolbox select to add a “Technology Solution” package.

### Preparing the SVG

The first step to use the track map is to have the automatically generated SVG generated with a valid layout and the SVGData file device. Some simple post processing must be done on this SVG in order to function with the Trak Paper widget. The SVG must be generated from a system with MpMotion 5.13+. The post processing is done by executing the SvgPython.py script with the input file name of the pre-generated SVG. To run the script you will need to have Python (preferably 3.8+) installed

1. Place SvgPrep.py and generated SVG into the same directory
2. Open a command line in this directory
3. Type python SvgPrep.py -i preGeneratedSvg.svg -o OutputFileName.svg
   1. You can enter the command with or without the file extensions

Upon a success a new file named “OutputFileName.svg” will have been generated in the same directory. This is the new pre-processed SVG that you will use in the paper widget.

### Deploy to your Software Configuration

To your project add the track paper library and the “TrakDiag” task. The TrakDiag task is the task that automatically implements the TrakPaper Core function block and supporting function blocks. The TrackDiag task must be added to the cyclic 8 task class (or a large idle time task class)

### Populate the Segment List Variable

The TrakPaper library uses a list of segments to build the string. In order for this to work you must populate a segment list with the segment variable reference and the name of that segment. The variable is TrackDiag.Seg.SegList.

The easiest method is in your init of the TrakDiag task add your definitions for your segments and the total segment count:

TrackDiag.Seg.SegList[0].Name := 'gSeg\_A\_1'; TrackDiag.Seg.SegList[0].McType := gSeg\_A\_1;  
TrackDiag.Seg.SegCount := 1;

The above would make a system with a singular segment with a segment name of gSeg\_A\_1.

The last step in the init is to make sure to change your assembly name for the two Assembly status blocks.

### Setting up the Paper widget

The final step that must be done is to connect to the paper widget and custom status widget.

You’ll need to enable the following local variables in OPC-UA for the TrackDiag task

1. TrackDiag:clickedSegment
2. TrackDiag:clickId
3. TrackDiag:paperCore.SvgTransform
4. TrackDiag:paperCore.SvgContent

#### Paper Widget

To the paper widget, map the paperCore variable SvgTransform and SvgContent variables from the TrakDiag task.

The Trak paper widget uses the color List property of the paper widget for the segment status colors during operation. The default color values could be the following: #c9c9c9,#FF0000,#00FF00,#FFFF80

The important part is that the following indices are used for the styling of the segments:

|  |  |
| --- | --- |
| Style Description | Index |
| Segment Default Style | 0 |
| Segment is in an error state | 1 |
| Segment is okay style | 2 |
| Segment is in a warning state style | 3 |

Custom styling is possible by changing the color at the defined index.

In order to use the segment details, the “clickID” property must be set for the Paper Widget. This can be done by using the below code block in an event binding and setting the “contentRefId” and “widgetRefId” correctly

<EventBinding id="content\_trackmap.paperTrackMap.Click">  
 <Source xsi:type="widgets.brease.Paper.Event" contentRefId="trackmapContent" widgetRefId="paperTrackMap" event="Click" />  
 <EventHandler>  
 <Action>  
 <Target xsi:type="opcUa.NodeAction" refId="::TrackDiag:clickId">  
 <Method xsi:type="opcUa.NodeAction.SetValueString" value="= elementId" />  
 </Target>  
 </Action>  
 </EventHandler>  
 </EventBinding>

#### Custom Status Widget

To use the custom widget, add the custom library provided into the project. Add the SegmentStatus widget into your content wherever you desire. Connect the TrackDiag:clickedSegment variable to the widget. That is all that is needed for the custom widget

#### Theme Definitions

The custom widget utilizes some custom themes for the toggle buttons present. You can use your own themes should you like, or you can add the provided style file to your existing theme for styling.

#### Text Files

In the provided files there are the text files. These can be included to populate the error texts and segment statuses in the status widget. Add the provided text files to the text definition for the project.