DrawIO symbol definition and svg export plugin

# Overview

[DrawIO](https://www.diagrams.net/) (transitioning to diagrams.net) is an [open source](https://github.com/jgraph/drawio) technology for building diagramming applications. The application has been chosen to produce the symbols used within the DISC DEXPI project.

In addition to the functionality provided as-is by the product it was also necessary to build an additional plug-in to provide bulk export of svg files with support for layers and dimensioning of the symbol.

The plug-in has been developed by [pnb plants & bytes GmbH](https://www.plants-and-bytes.de/en/) for Equinor

This document describes:

* The structure of the symbol definition with reference to use of layers, special symbols and property values
* How to install the plug-in
* The configuration settings for the plug-in.

# DrawIO symbol definition

The symbol definition applied within the DISC DEXPI project provides the details needed to ensure that the CAD vendors can create each symbol using defined dimensions along with specified origin and connection point ordinates.

## Layer definition

Each symbol is drafted with the following layers. Layers can be turn on/off as required to view the information pertaining to that layer. The case sensitive layer name is used within the configuration of the svg export plug-in to define which layer information shall be exported within the given export file. (Ref: Configuration Options)

* Symbol - the ‘raw’ symbol that will be drafted in the 2D CAD tool
* Origo - red circle designating the origin point of the symbol
* Label - general label placement and lookup code for attributes within the label
* Grid - a grid pattern to be used during export to help the CAD drafters to recreate more complex symbols
* Dimensions - main dimension information related to the symbol
* Connection - green circles designating key connection points for the symbol
* Option *#*  - \*\* optional layer(s) e.g., Option1 & Option2 displaying the symbol for different attribute settings (example: PV003A has 3 variants based on open/close settings)

## ‘Special’ symbol definition

The DISC DEXPI symbol legend uses special coloured circle symbols to identify origin and connection points: the centre of the circle provides the exact x,y ordinates of the origin / connection point. The use of coloured circles allows for both human and machine readable interpretation of this important information.

**Origin (origo):** Identified by red (RGB: 255,0,0 Hex: #ff0000) circle

**Connection Points:** Identified by green (RGB: 0, 255,0 Hex: #00FF00) circle

## Property definition

### Dimension:

Each symbol is drawn with both a primary width and height dimension line. Properties of these lines are used within the svg export plugin to redimension the exported symbol to the correct size.

**Dimension:** W – indicates primary width dimension

H – indicates primary height dimension

**UoM:**  Unit of measure

**Value:**  Size of the primary dimension

### Connection Point:

Connection point symbols are defined with the following properties to aid with validation of the DEXPI file and for use in other applications where dynamic connection lines are used.

**Direction:** Comma delimited list of values in degrees indicating the valid direction at which a connecting line can intercept the symbol. E.g. Direction: 180 indicates a connecting line can only intersect with the symbol from the left side of the symbol along the x-axis plane. **0o start point is given along the positive x-axis as shown.**

**Chart, pie chart

Description automatically generated**

Figure : Connection line angle intercept value reference

**LabelConnector:** Indicates if the connection point allows label connection lines

**PipingConnector:** Indicates if the connection point allows piping connection lines

**SignalConnector:** Indicates if the connection point allows signal connection lines

# SVG Plug-in

## Install

Execute following steps to install and use the plugin:

1. Install draw.io (plugin is developed and tested with version 20.3.0)
2. Start draw.io with the command line argument --enable-plugins

* You can add it to the link of draw.io
* The argument is needed to add the plugin and also to use the plugin

1. Choose the plugin file in the open file dialog Extras|Plugins…|Add|External Plugins|Select File…
2. Draw.io adds the plugin file to the folder: C:\Users$(User)\AppData\Roaming\draw.io\plugins
   * Add the file “svg\_bulk\_export.config.json” to this folder
3. After restart of draw.io three new menu items should be added to the bottom of menu “Extras”
   * SVG bulk export – Single symbol
   * SVG bulk export – All symbols
   * SVG bulk export - Create zip file with folder structure' to menu 'Extras'

## Configuration Options

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| addDrawIOFilenameAsPrefix | Boolean | Prepends the filename of the opened drawio file followed by a hyphen. (Ref: SVG Filename) |
| openDevTools | Boolean | Opens the Developer Tools (which are also accessible in the menu Help->Open Developer Tools) for debug purpose at startup. |
| writeToFileSystem | Boolean | Writes svg files to the file system. Due to missing permission to create folders inside a drawio plugin the user has to create folders (defined by the property '*exportPath*' in *layerGroups*) manually in advance. |
| zipFilePath | String | Exports all svg files to a single zip file including folders as defined by the property '*exportPath*' in *layerGroups*. This bypasses the missing permission to create folders inside a drawio plugin (have a look at property *writeToFileSystem*). |
| cleanup | object | Performs some cleanup algorithms like resolve of transforms, element conversions, centering of origo layer, etc. |
| * enabled | Boolean | Enables the cleanup algorithms overall |
| * unitScaling | Object | Contains properties for the unit scaling algorithm |
| * + enabled | Boolean | Enables unit scaling. The scaling factor is defined by the  properties of the dimension layer (Dimension (W,H), UoM (mm), Value) and the width and height of the symbol layer. |
| layerGroups | Object | Contains one or several layerGroups. Each layerGroup can have its own *enabledLayers* such that svg files are generated with different combinations of enabled layers. In detail a layerGroup is an object itself and can have following properties: |
| * filenamePostfix | String | Overrides the last part of the filename. (Ref: SVG Filename) |
| * exportPath | String | By default svg files are exported to the folder containing the opened drawio file. This *exportPath* overrides or changes the default export path. It can be an **absolute** path which completely overrides the default path. Or it can be a path **relative** to the default path. |
| * exportPerOptionLayer | Boolean | If enabled a svg file is exported for each option layer. Option layers are defined by starting with Option. For each option layer a svg file will be exported by enabling the option layer and the enabledLayers. |
| * enabledLayers | Array | An array containing the layers which shall be exported for this layerGroup. |

## SVG Filename

The filename of an exported svg file has following format:

{drawio\_filename}-{symbol}\_{layer}

The {layer} is the name of the option layer if exportPerOptionLayer has been enabled or the name of the layerGroup otherwise. The property filenamePostfix completely overrides {layer}.

The filename can be influenced by following properties:

* addDrawIOFilenameAsPrefix
  + Prepends the filename of the opened drawio file followed by a hyphen: {drawio\_filename}-
* filenamePostfix
  + overrides the last part of the filename: \_{layer} (including the prefixed underscore)