

# Images and Symbols

FactoryStudio has great features to improve the productivity on creating graphical displays to your application. The drawing tools allows you to easily manage symbols, image files, import objects from a 5000 objects gallery, create your own symbols with dynamic properties, and map tags and templates to its default graphical user interface.

There are three main repositories for reusable components, the Images, the Symbol Factory library, and the Local Symbols Gallery. Let's explore each one.

**Images:** Used to insert an external image file from your computer into the project configuration, like an icon, wallpaper, object, or background image. You can manage the imported images at Edit > Displays > Resources. After the image is imported, you no longer need the original file. Images can be used as color brushes for any drawing object, and they can even be used as "color" or the ColorFill dynamic attributes.

**Symbol Factory:** This is an extensive library of elements, with more than 5000 symbols, all using vector graphics, that you can use for creating your displays or use as a template to create your own custom symbols. Those objects can be applied statically, just as an image or icon, or you can add Dynamics and save them to the Local Symbols library.

**Smart Symbols:** These are the most frequently used symbols. They include the symbols you create in the scope of a project and a library of more than 500 symbols designed specifically to FactoryStudio. The symbols in this library can be kept synchronized with the library, in this way, when you change a symbol in this library, the change will be automatically applied to all displays using that symbol. Local Symbols also can have dynamics properties that you can easily map to tags in displays; they can also be defined to be the default graphical representation of tags.

## In this section...

## On this page:

[Image Resources](#)  
[Symbol Factory](#)  
[Smart symbols](#)

## Image Resources

You can import any image file at the Edit > Displays > Resources tab. This creates a repository of images for the application that is stored inside the project database file, so you don't need the original files anymore. These images can be used to be the fill of an object, such as rectangles, ellipses, LabelBoxes, paths, page background, or target color in runtime dynamics using colors.

If you need to update an image with a new version, you can replace it, and it will be updated throughout the application. Be sure to keep the resource name the same.

You can use any image file to fill or "paint" an object when creating displays from Draw > Drawing.

You can import the following file types:

- .bmp
- .gif
- .ico
- .jpg
- .png
- .tiff
- .wdp

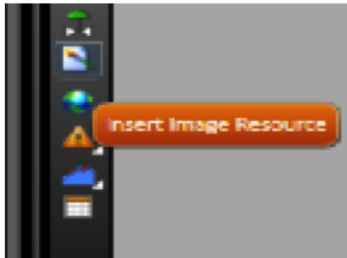
To manage resources:

- Go to Edit > Displays > Resources.
- Click Import Images.
- Navigate to where the images are that you want to import.
- Select the image files and click Open.
- In each row, enter or select information, as needed.

Column	Description
Name	Edit the default name.
Description	Enter a description of this image.

Folder	Specify the folder where the resource is
ResourceType	Specify the type of the resource
Dimension	Read-only. Show the dimension of the resource file
Size	Read-only. Show the size of the resource file

Adding an image to a display



- Go to Draw Environment, and click on the Insert Image Resource.
- A new window will open. Now Click on Import file button.
- The new image becomes the fill for the rectangle. Click on Fill to change the image settings.
- You can change the Stretch to None, Fill, Uniform and UniformToFill. Also you can see all image files that are in the project and change between then using the Select Resource button or just click on the image in the top toolbar .



Typically you should use UNIFORM or FILL for image stretching. The image (using the Appearance menu at the left, can be applied to any graphical shape at the display)

## Symbol Factory

The symbols from the Symbol Factory are efficient and well designed vector graphics images. If you need an object like a pump, valve or others in your applications, just open the library and browse for the element or search by name. The difference between the symbols from the Symbol Factory and Local Symbols is that symbols from the Symbol Factory, once included in a display, become part of that display, and have no further connection with the library. In contrast, Local

Symbols can keep an active link with the library, and can also have dynamic properties.

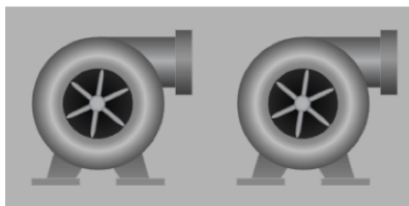
- Go to the Draw Environment then click on the Symbol Factory Icon on the vertical toolbar.
- After clicking on the icon the Symbol Factory will open, to insert a symbol just choose one category and symbol then double-click on the symbol and click in any place on the screen.

### *Adding color change animation to Symbol Factory elements*

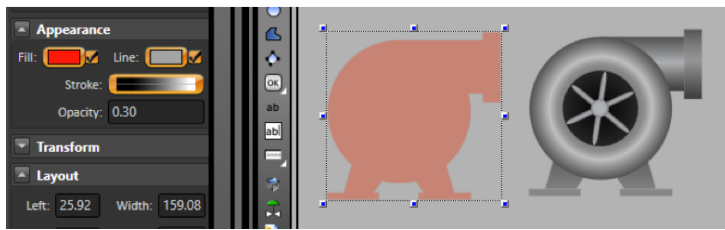
If you want to insert a color change animation, without changing anything on the symbol, you can copy the symbol and overlay a transparent image on top of your image.

For example, suppose we have two pumps. To insert a simple animation that shows when the pump is ON or OFF follow these steps:

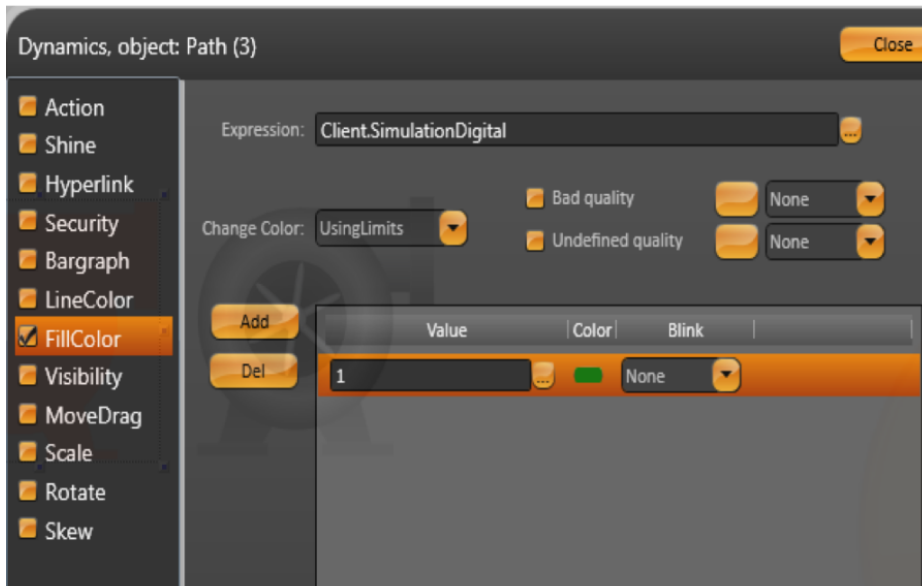
- First make a copy of the object, select the symbol and click Ctrl+D to duplicate it (you can also select the Symbol, hold the CTRL key and drag the object, or you can copy/paste it).



- Then select the second pump and click on the Union button on the horizontal toolbar.
- Now with this new pump, locate the OPACITY property at the left panel, select an Opacity from 0.2 to 0.5, according the desired visual effect, and select the color for your initial brush (red or green, for instance).



- Double click on the second pump and change the setting of the Fill Dynamic.

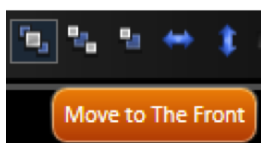


Use the Client.SimulationDigital to simulate a controller changing between 0 and 1.  
If you want to control the value yourself, use Client.DigitalValue or Client.NumericValue

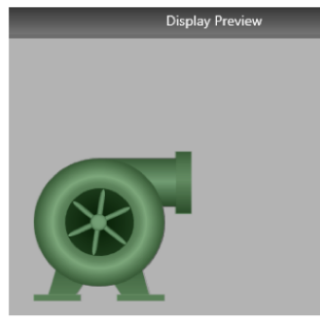
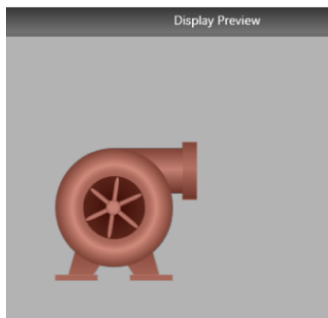
- Select both pumps and click on the Align Horizontal Center button and Align Vertical Center button.



- If the pump with the Opacity changes stays behind the other pump, click on the Move to The Front button to bring it to the front.



- You can now click on the Display Preview button  on the top toolbar to see the new pump's animation.



## Smart symbols

The Smart symbols library includes the most frequently used graphics. You can also create symbols and add them to the project local library. Symbols can just be a static image or they can have dynamic properties and tags links.

To view the symbols used by your project, do one of the following:

- To see only the symbols used in this project, go to Draw > Symbols. In that page you can export, import or delete symbols.
- To see all available symbols, go to Draw > Drawing, and click Open Symbol Library

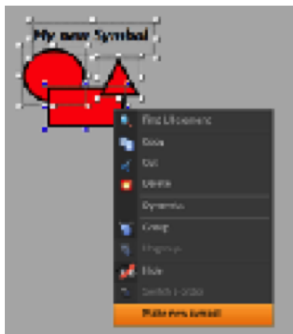
### *Creating new Symbols*

You can use the drawing tools and the COMBINE commands at the horizontal toolbar to draw your own symbols.

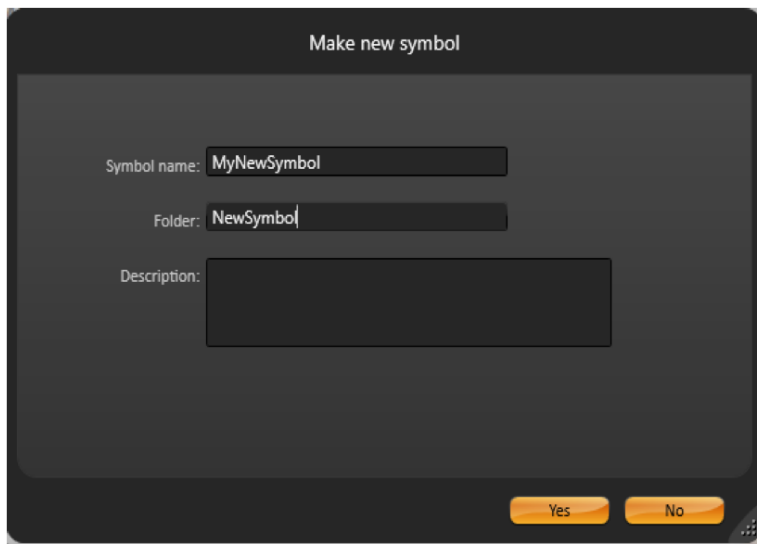
You can also bring symbols from Symbol Factory and click at the UNGROUP command at the horizontal toolbar to edit the imported symbols.

You can also use the "Direct Selection" cursor to edit internal elements without needing to ungroup.

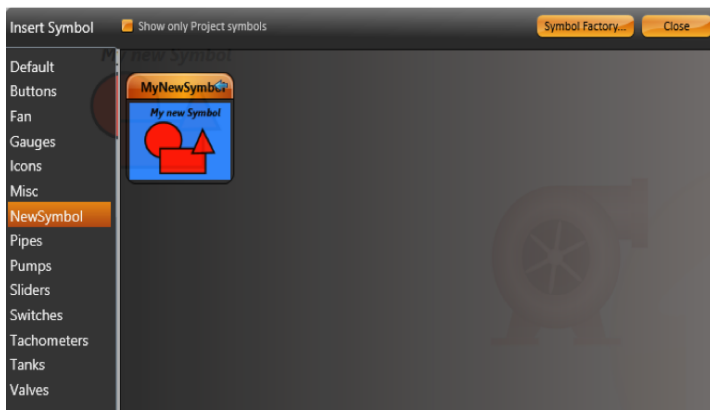
If you want to save the new symbol to the Local Symbols Library, just select the object or objects that you want to be part of the symbol, then right-click and select "Make a new symbol".



- Enter the Name, Folder and the description of the Symbol.



- Now you can open the Symbol Library and choose the symbol to use at your displays.



The components in the Local Symbol library may have built-in dynamic properties and an easy way to map them to realtime tags in your application. This is explained in the following section.

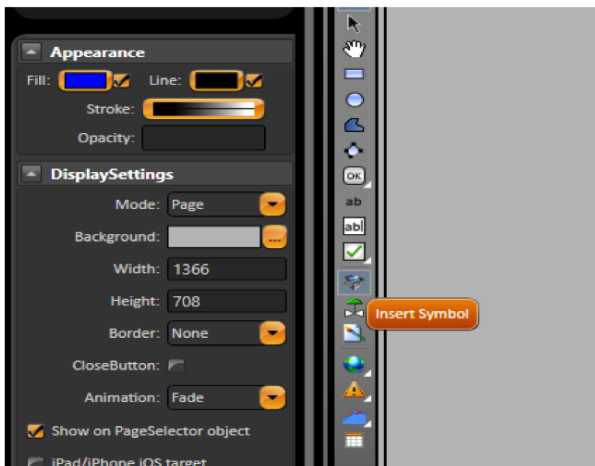
If you have dynamic properties and tags being used in that symbol, the system will automatically create the symbol label parameters. After making the new symbol, double click it to verify its configuration parameters.

In order to Edit a symbol that is already in the Local Symbol library, insert the symbol in a display, right click on the symbol, and select "Edit Symbol", that will change the drawing tool, locking the other elements in the page, so you change the symbol. When you finish, right-click at any position to select Save to Library, Save Only to Local display, or Cancel your editing.

The objects from "Local Symbols" are composed of some default symbols from the file SymbolLibrary.tproj, added with the product installation, and the symbols you created for your specific project. If you change, delete or insert a symbol in the SymbolLibrary.Tproj, it will be accessible to all projects in that computer. The symbols you are using in your project are saved inside the PROJECT file; therefore they are completely independent of any external file.

#### *Inserting a Local Symbol*

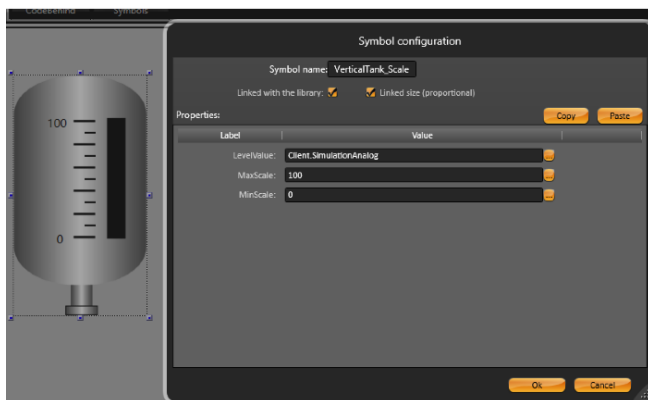
- Go to the Draw Environment, right-click any place of the screen and select "Insert Symbol". Alternately, you can click on the "Local Symbol icon" on the vertical toolbar.



- After selecting the category and symbol, click on the symbol and then click on the screen to position the symbol. You can also drag-drop symbols to the screen. If you double-click on a symbol, that will close the Library window, allowing you to insert the selected object.
- Double-click on the symbol to see its runtime properties and map it to tags in your application.



Some symbols do not have custom parameters. In this case, the double click will show the Dynamics dialog.

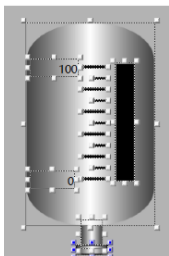


### Symbol Parameters

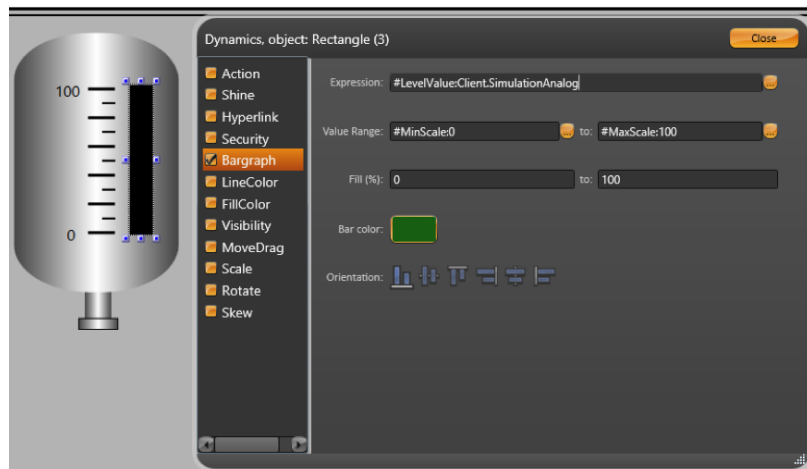
When you insert a symbol from the Symbol Library it may have some parameters, like in the example the VerticalTank symbol has the parameters LevelValue, MaxScale and MinScale. To change the linked tags to those parameters, just edit the new tag names in the dialog.

### Editing and modifying symbols

Right-click on the VerticalTank symbol and select Edit Symbol. Now you enter in the Edit mode, you can see all objects that are part of the Vertical Tank.



Click on the Black rectangle to see its Dynamics.



As you see we use `#<PropertyName>:TagName`

That syntax will create exposed label parameters for the symbol, making it easier to map the linked tags when using the symbol. In this examples, the symbol has the labels "Level", "MinScale" and "MaxScale". When you insert this symbol, you will be able to set values for those parameters. After the ":" character, you have the default value.



#### Example

"#LevelValue:Client.SimulationAnalog", we are creating a property called "LevelValue" which has "Client.SimulationAnalog" as the default value.  
You can use any name for the Label parameter.

When creating symbols, it is useful to initially map the properties to `Client.SimulationDigital`, `Client.SimulationAnalog` or `Client.SimulationDouble`, which are variables with values changing every second so you can see the result of your dynamic properties. You can use the internal variables `Client.DigitalValue`, `Client.NumericValue` and `Client.TextValue`, when you just want to put a placeholder value, with no built-in simulation.

In order to save the symbol, right-click on the symbol or on the display and press **Save to Symbol Library**, or click the **Save** icon on the top toolbar. Double-click on the symbol to verify its properties.

### Linking Tags and Symbols

You can configure a symbol to be the default representation of a Tag Type. In this way, every time you paste a tag into the Draw Environment, a symbol will be automatically created mapping to the inserted tag.

The tags of type Digital, Integer, Double, Decimal, Text, DateTime and TimeSpan have a default visualization that is the name of tag and an input /output text box. If you select multiple lines at **Edit>Tags>Objects** (click in one line, shift click in another line to mark a range), right-click to copy those lines to the clipboard, and paste into the displays, the system will automatically create one object to each of those tags.

If you want to change the properties of the objects created, for instance change the TextIO dynamic from "TwoWay" to "OutputOnly", select all the TextBox objects (clicking with the mouse on the display, and keeping it pressed, select the area with the objects), and double-click at any selected object to open the "Edit Combined Dynamics" dialog.



#### Hint

You can also copy the lines from **Edit > Tags > Objects** to Excel, add the columns **Left** and **Top** into the Excel table, then copy and paste that table, including the header column, back to the display; in this case the system will position the objects using the coordinates found in the table.

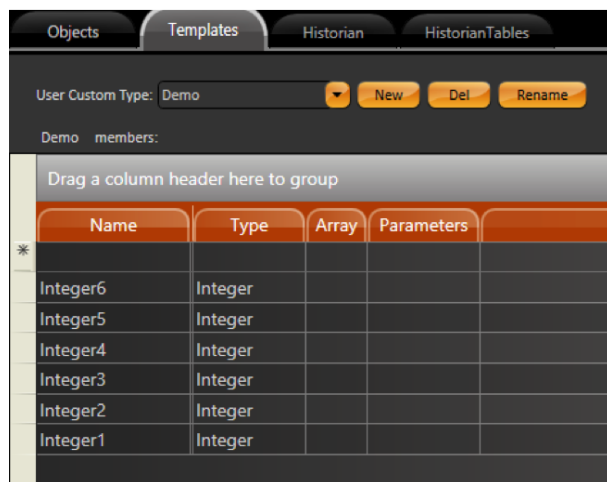


If you create a symbol with one dynamic parameter, and save it with the name "Integer", that will become the default visualization for the "Integer" tags. The same applies to any tag type.

### Mapping a Symbol to a Tag Template

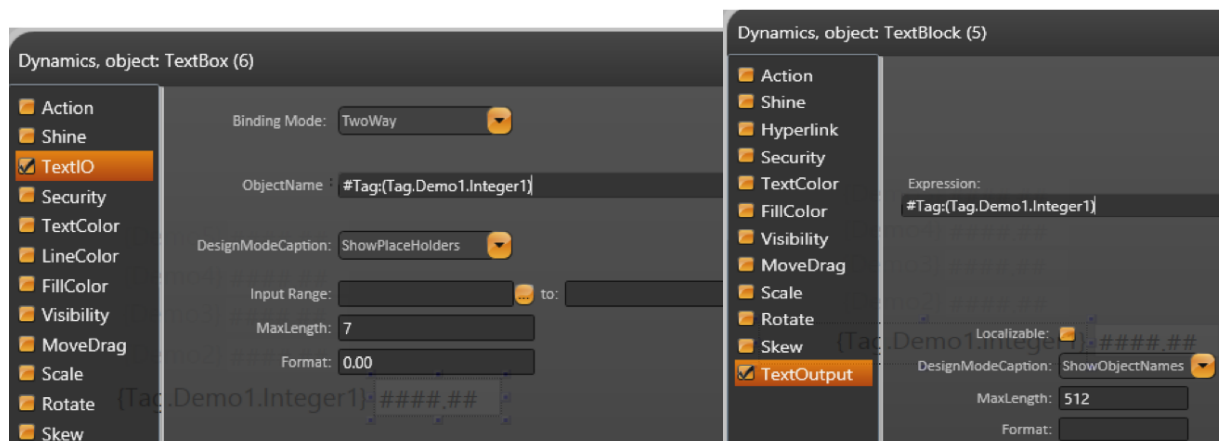
You can also define a symbol to be the default view for a tag template.

- Go to Edit > Tags > Templates to create a new template.

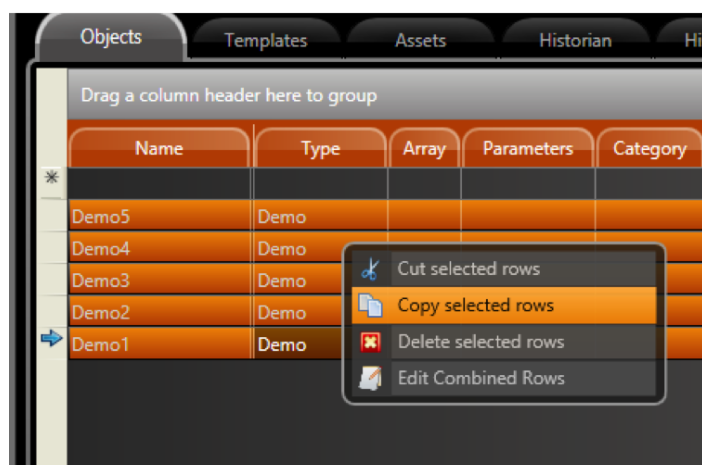


Go to Edit > Tags > Objects and create Tags with Type: Demo.

- Back at the Draw environment insert some objects that you want to be part of the symbol. In the example we use a TextBox and TextBox with the parameters #Tag: (Tag.Demo.Integer1) in their dynamic properties



- Now select the TextBox and the TextBox and right-click and make it a New Symbol with the Name and Folder called "Demo".
- The parentheses () in the expression of the TextBox and TextBox, defined the scope, on what should be replaced when you map the symbol. The hash tag syntax #Label: when you paste the Tags, it is the placeholder that will show where in the dynamic animations or expressions the name of tag should be replaced.
- Go to Edit>Tags and copy all tags with Type:Demo.



- Back to Draw environment and paste the tags.



When inserting tags of that template type, the system will look for a Symbol with the same name of Template. If a match is found, that symbol is created and linked to the Tag you were inserting at the display.

{Demo5}	####.##
{Demo4}	####.##
{Demo3}	####.##
{Demo2}	####.##
{Demo1}	####.##



If you create symbols using only Tag Elements, not expressions, you don't need to explicitly use the full syntax: #Label:(tag.Demo).Integer1 to define the label parameters. If you just use your tag at in the object dynamic properties and execute the Make Symbol command, the system will automatically search for tags in the symbol and create the related parameters. When using expressions, or when having multiple tags and templates in the same symbol, it is necessary to explicitly use the hash tag # and the parentheses in order to define the scope of your symbol's customizable parameters.

#### Linking Tags and Symbols using Categories

If you want to map a Symbol with a group of Tags, but you don't want to make this symbol map all Tags with the same Type. you can use the Category column to link which symbols should be used as the default interface to each group of tags.

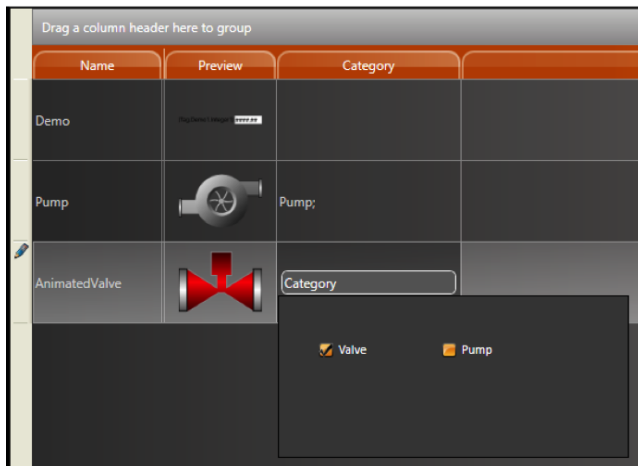
- Go to Run > Dictionaries > Categories and create new categories.

Localization	EnumerationSets	Categories
Drag a column header here to group		
Name	Title	
* Pump		
Valve		

- Back to Draw Environment and insert one symbol.
- Go to Draw > Symbols Tab and on the Category Column chose the respective category for the symbol.



If the Category column is not visible, right-click in any Column name and select the check columns you want to see.



Go to Edit > Tags and define the category for the tags.

Drag a column header here to group			
Name	Type	Category	Par
Integer4	Integer	Valve;	
Integer3	Integer	Valve;	
Integer2	Integer	Pump;	
Integer1	Integer	Pump;	
Demo5	Demo		
Demo4	Demo		
Demo3	Demo		
Demo2	Demo	Valve;	
Demo1	Demo	Valve;	

Select all Tags, copy then paste then into the Draw Environment. Each Tag will map to the respective symbol, according to the association defined in the Category column.