IBM ILOG JViews Diagrammer V8.6

Using the Symbol Editor

Table of contents

Getting started with the Symbol Editor

Presents the main features of the Symbol Editor using a Business Activity Monitoring (BAM) example.

In this section

Transforming symbols using parameters
Explains how to use parameters to convey some behavior to symbols.

Working in different modes
Describes the two modes available in the Symbol Editor.

Testing your symbol
Explains how to test a symbol in Preview mode.

Transforming symbols using conditions
Explains how to use conditions to apply some behavior to symbols.

Using link-in and link-out ports
Explains how to specify in and out ports on symbols for the connection of links.

Generating and using symbol reports
Explains how to generate and use reports on palette symbols.

The BAM Dashboard example
Shows how to create symbols that will be integrated into a Business Activity Monitoring dashboard.

About the Symbol Editor

Using the Symbol Editor you can refine the graphic representation of data by associating parameters and conditions with a symbol. You can also specify how symbol elements will behave in response to changes to data.

This section explains how to use the Symbol Editor for JViews Diagrammer to start developing IBM® ILOG® JViews symbols. Information provided in this section is based on an example Business Activity Monitoring application.

When you develop a graphical application, you first create your symbols using the Symbol Editor. You then incorporate the symbols into an application using the Designer for JViews Diagrammer, the Dashboard Editor sample application or directly in an IBM® ILOG® JViews application using the Java™ API. For more information on how to integrate a symbol into another application, see *Next steps after the Symbol Editor*.

Running the Symbol Editor

You can run the Symbol Editor under the Windows® or UNIX® operating systems. The startup conditions are the same in both cases.

To run the Symbol Editor under Windows:

♦ In the Start menu, select **All Programs>IBM ILOG>IBM ILOG JViews Diagrammer 8.6>Symbol Editor**.

  IBM® ILOG® is the default program group and may be different if you have installed JViews Diagrammer into a different program group.

-- or --

1. Go to the directory `<installdir>/jviews-diagrammer86/bin/symboleditor`.
2. Double-click `run.bat`.

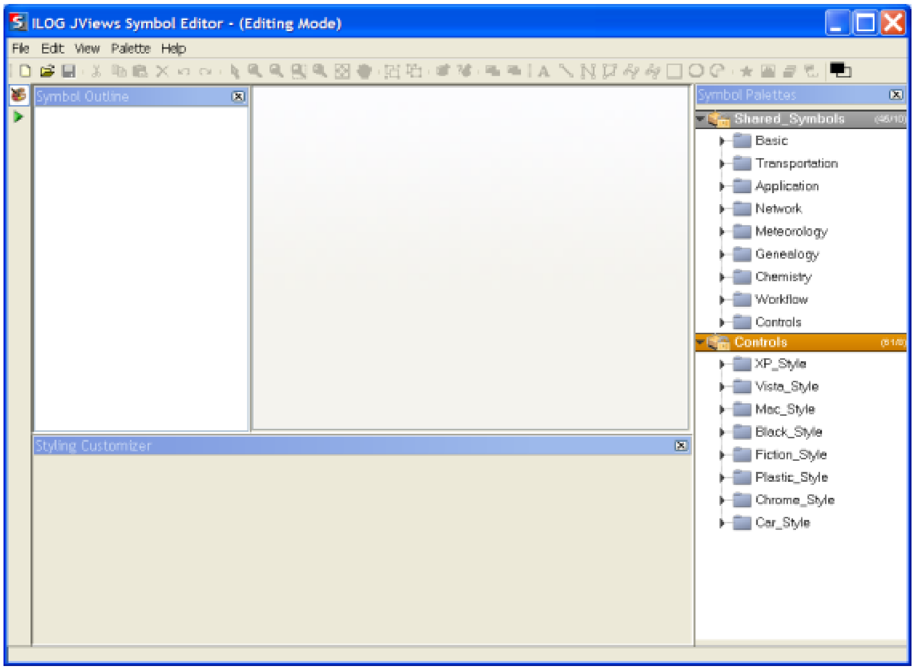
To run the Symbol Editor under UNIX:

1. Go to the directory `<installdir>/jviews-diagrammer86/bin/symboleditor`.

   Make sure your PATH environment variable is set correctly to find this directory.

2. Enter `run.sh`.

When you start the Symbol Editor, the interface is displayed as shown in the following figure.



The Symbol Editor window

The way the Symbol Editor operates depends on the mode it is in. You can see the current mode in the title bar, and you can change the mode using the buttons in the vertical toolbar. For a discussion of modes, see *Working in different modes*.

To close the Symbol Editor:

- Select **File>Exit** from the menu bar.

Note: The Symbol Editor stores the interface status when the application is closed.

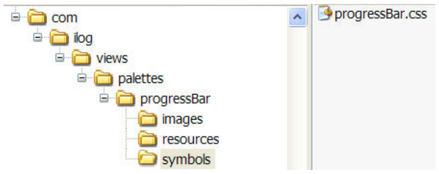
Default palettes of symbols

As shown in the figure *The Symbol Editor window*, the Shared Symbols, and Controls palettes are opened by default in the Symbol Palettes pane. These symbols are supplied by IBM® ILOG® to work as examples, templates and building blocks for your own symbols.

Understanding symbols

A symbol is a collection of graphic objects, parameters and conditions used to give a dynamic representation of changing data. It is made up of objects and other symbols. Once created, a symbol is manipulated either by being imported into other IBM® ILOG® JViews products such as the Designer, the Dashboard Editor, or directly using the IBM® ILOG® JViews Java™ API.

A CSS file is used to store the information necessary to represent a symbol, including the type and position of shapes, text, IVL files, images, and other symbols used to make up the symbol. Symbol data is stored inside the palette file structure. The physical structure of a palette is shown in the following figure.



Symbol structure in a palette

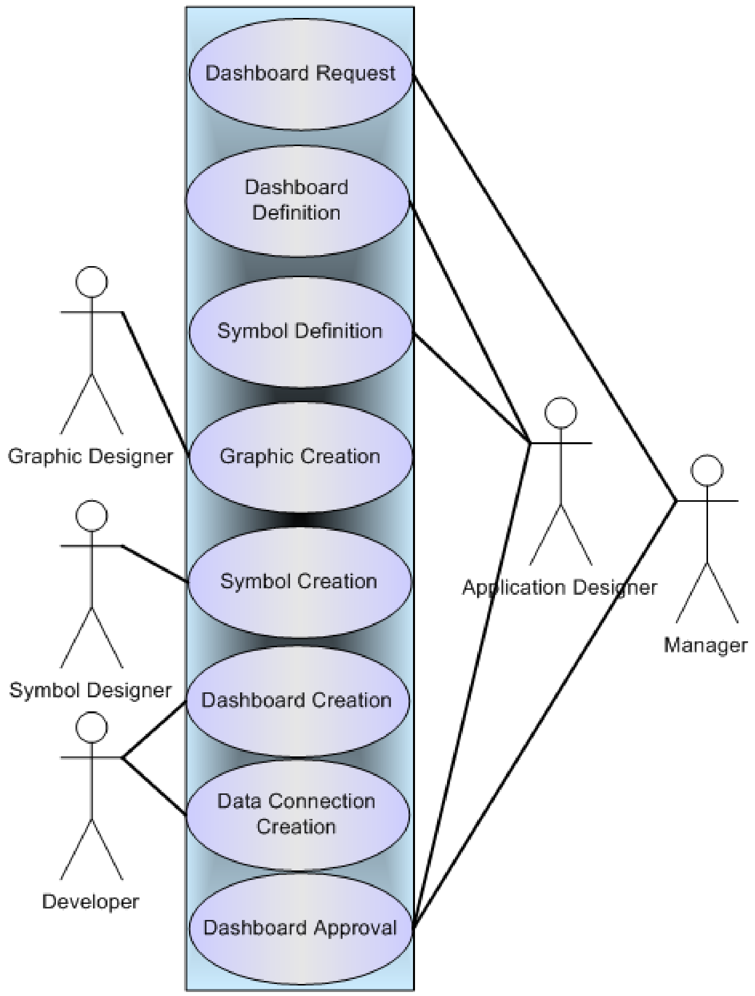
Symbols are based on simple graphic objects created using the Symbol Editor, or more complex images imported from standard graphics packages created by a graphics designer. The following graphics standards are supported:

- ♦ Portable Network Graphics (`.png`)
- ♦ Joint Photographic Experts Group (`.jpg`)
- ♦ Graphics Interchange Format (`.gif`)
- ♦ Scalable Vector Graphics (`.svg`, `.svgz`)
- ♦ IBM® ILOG® JViews Graphics (`.ivl`)

Planning a project for the Symbol Editor

Before creating a symbol, you must know what sort of application a symbol will be used for. For example, a BAM application will present only small amounts of specific data that is updated infrequently. An example of this data are the Key Performance Indicators. Symbols for such an application are graphics heavy and integrate many transformations to add intelligence. On the contrary, a SCADA application will use hundreds of symbols to present enormous amounts of data that is updated in real time. In this case, it is essential to create light symbols with as small a memory footprint as possible.

To ensure that you produce symbols that match your application requirements, the best practice is to follow the use case shown in the following figure.



Use case for creating symbols

1. A manager issues the initial dashboard request.
2. An application designer designs the application to be created and the symbols that will be needed.
3. An application designer defines the parameters, constraints, and look and feel for each symbol.
4. A graphic designer creates the graphics.
5. A symbol designer creates the symbols with the Symbol Editor.
6. A developer integrates the symbols into an application using the IBM® ILOG® JViews application.

The application is approved by the application designer and the manager.

Working with symbols

Shows you how to create a progress bar that will be used for a basic Business Activity Monitoring (BAM) application. It is used to monitor Key Performance Indicators for different regions.

In this section

Making a new symbol

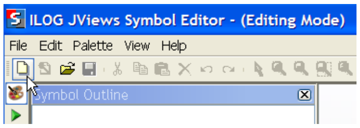
Before creating symbols, you need to know what you are creating them for. The use case for project planning is explained in *Planning a project for the Symbol Editor*. A full explanation of the BAM project is explained in *The BAM Dashboard example*.

To create a new symbol:

1. Select **File**>*New*.

   -- or --

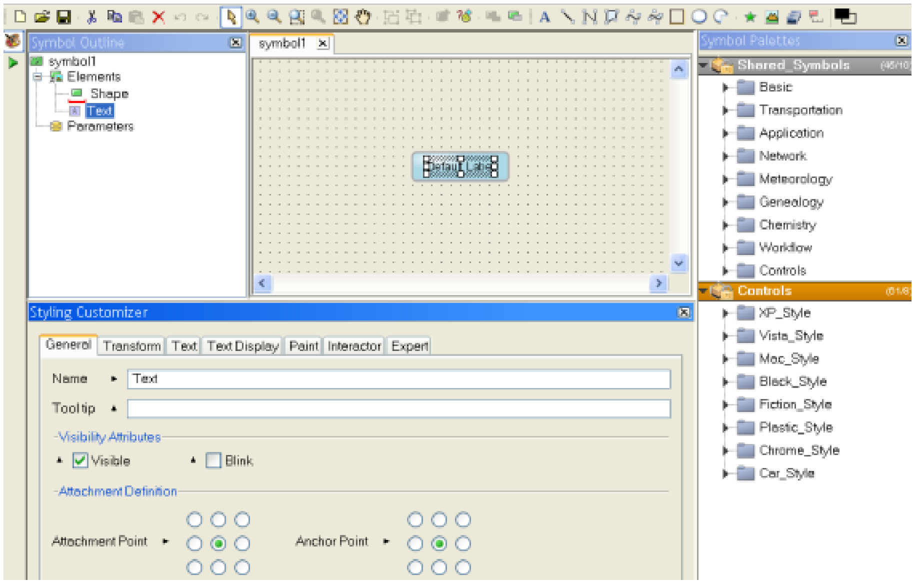
2. Click the New Symbol button at the top left side of the Symbol Editor window.


ILOG JViews Symbol Editor - (Editing Mode)
File Edit Palette View Help
Symbol Outline

The New Symbol button

3. Press CTRL+N on your keyboard.

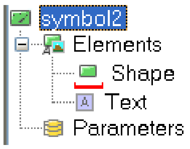
The default symbol appears in the drawing pane.



The default symbol

The default symbol shown in the figure *The default symbol* contains two objects: the Shape object and the Text object. The handles around the Text object indicate it is selected.

The position and coordinates of objects in a symbol are relative to the base object. The base object is the object on which the coordinates and position of all other objects in the symbol are calculated. The base object in a symbol is indicated in the Symbol Outline pane with a red underline.



The base object indicator

When a symbol is opened in the Symbol Editor, the actions available are indicated by the active buttons in the menu bar. The objects contained in the symbol are displayed in the Symbol Outline pane on the left of the Symbol Editor. Customization options for the selected object are displayed in the Styling Customizer pane at the bottom of the Symbol Editor.

Note: You can create a new symbol from an existing one by double-clicking the latter in the Symbol Palettes pane.

Understanding the Styling Customizer

The Styling Customizer is a user-friendly way to customize object properties and their values. The properties are subdivided into categories by tabs. The Styling Customizer for the current object selection is displayed in the lower pane of the Symbol Editor.

For each property of the currently selected object, the Styling Customizer displays one of the following, according to the type of value concerned:

- A checkbox to select or clear
- A list of valid values or items for you to choose from
- A step scale for numerical values
- A text field for you to enter text
- Buttons for you to add or remove values or lines of text
- An Ellipsis button for all properties that need to be edited using a dialog box. For example, when you change the paint and stroke.
- A choice of the following systems for color settings:
    - JViews color disk
    - RGB (Red, Green Blue)
    - HSB (Hue, Saturation, Brightness)
    - Swatches

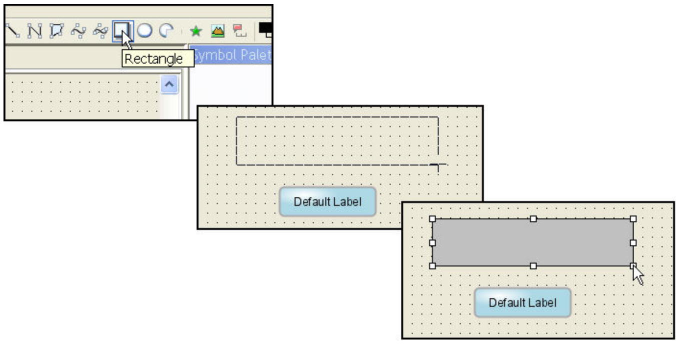
Adding an object

You will now create the first object for the basic progress bar. This progress bar will consist of a single rectangle object that changes width and color to show project advancement.

To add a rectangle object to your symbol:

1. Click the Rectangle button in the toolbar.
2. Hold down the left button of the mouse, drag the mouse over the area in the drawing pane where you want to place the object.

   The new rectangle appears in the drawing pane.



Adding a rectangle object

Selecting an object

You can select an object in a symbol in two different ways.

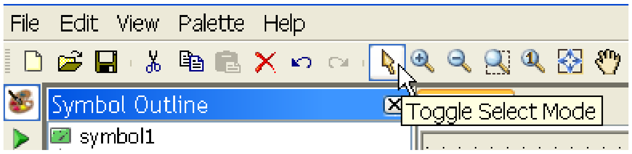
To select an object:

1. Select the object from the Symbol Outline pane.

   -- or --

2. In Select mode, click the object in the drawing pane.

By default, the Symbol Editor starts in select mode. If you have changed to another mode, such as zoom or pan, click the Toggle Select Mode icon as shown in the following figure to change to select mode.



Set Symbol Editor to select mode.

Tip: You can select multiple objects by holding down the CTRL or SHIFT keys while selecting the objects.

Performing basic object customization

As well as being added to the drawing pane, new objects are added to the object list in the Symbol Outline pane. By default, they are given the name of their object type. For example, the rectangle object created in *Adding an object* is called Rectangle.

To change the size and name of an object:

1. Select the Rectangle object.
2. Click the Transform tab in the Styling Customizer.
3. Type `120` in the Width field and press ENTER on your keyboard.

   The rectangle resizes to the new width.
4. Type `30` in the Height field and press ENTER.

   The rectangle resizes to the new height.
5. Click the General tab.
6. Type `progressbar` in the Name field and press ENTER.

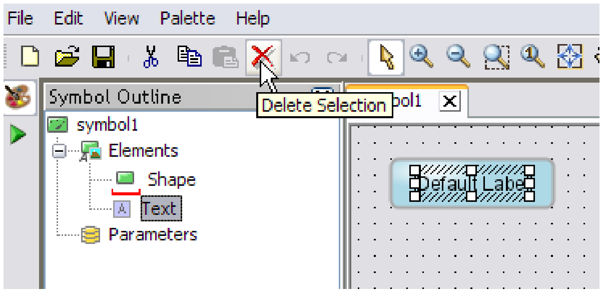
   You see the object name change in the Symbol Outline pane.

Note:
1. All object, palette and category names must be single words containing only alphanumeric characters.
2. In the Symbol Editor, all measurements are in pixels.

Deleting an object

To delete an object:

1. Select it.
2. Click the Delete Selection button in the toolbar.



The Delete Selection button

-- or --

♦ Right-click the object and select Delete from the shortcut menu.

-- or --

♦ Press the DELETE key.

The position and coordinates of an object in a symbol are relative to the base object. You cannot leave the drawing pane empty, there must be at least one base object present. The base object is marked in the Symbol Outline pane with a red underline.

If you delete the existing base object, a dialog box appears to warn you that you are about to change the symbol's base object. You will see this when you delete objects from the symbol you created in *Adding an object*, so `progressbar` becomes the base object.

To make the progressbar become the base object:

1. Select the Text object.
2. Press the DELETE key.
3. Right-click the Shape object and select Delete from the shortcut menu.

   The Confirm New Base dialog box opens.
4. Click Yes.

The progressbar is now the base object for this symbol.

Saving a symbol

Symbols are stored in palettes, and each symbol is assigned to a category in the palette.

To save the symbol created in the previous sections:

1. Choose **File>Save As**.

   The Save Symbol dialog box opens.

2. Select `bampalette` from the Choose a symbol palette list.
3. Select the `progressbars` category from the window.
4. Type `basicprogressbar` in the Name field.
5. Click Save.

Note: If a palette is not registered with the Symbol Editor, you will be prompted to create a new palette.

After a symbol has been saved once, select **File>Save** or press CTRL+S to save further changes to the symbol.

The Undo and Redo facilities

The Undo command can apply to the following:

- ♦ Property changes
- ♦ Cut or Delete operations
- ♦ Object moves
- ♦ Object creation
- ♦ Object deletion

To undo one of these actions, choose **Edit>Undo** or press CTRL+Z.

There are unlimited levels of Undo.

When you have undone an operation, you can redo it by choosing **Edit>Redo**.

Using palettes and categories

Explains how to work with palettes, categories of symbols, and symbols in palettes.

In this section

Creating a new palette

A palette is a Java™ archive file (`.jar`) containing one or more symbols organized in categories. Palettes are used for two purposes:

♦ To organize and store symbols for ease of use in the Symbol Editor.

♦ To export symbols to other IBM® ILOG® applications such as the JViews Dashboard Editor or the Designer for IBM® ILOG® JViews Diagrammer.

To create a new palette:

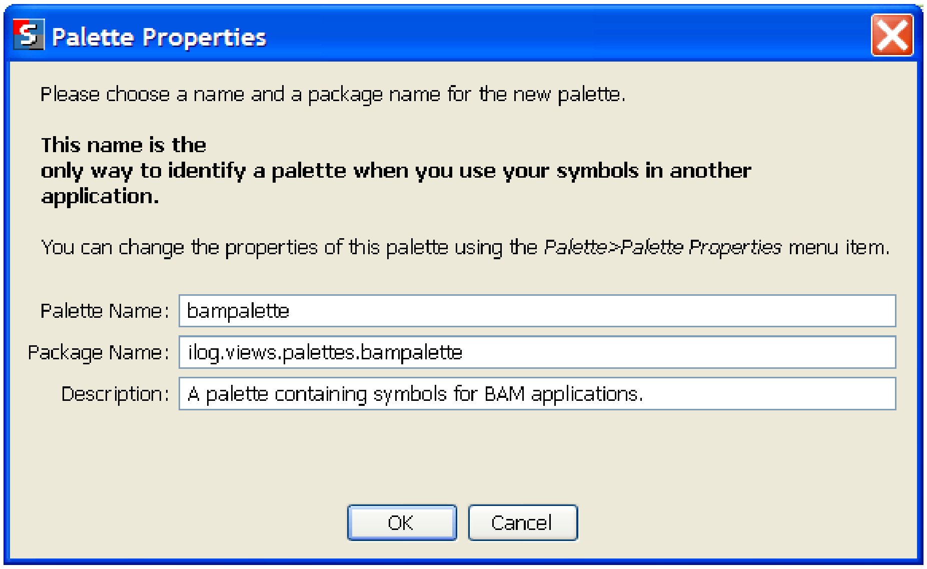
1. Select **Palette>New Palette** in the menu bar.

   The Palette Properties property sheet opens.

2. Type `bampalette` in the Name field.

3. Edit the name of the Java package used to store the palette in the Package Name field.

4. Type a description of the package in the Description field.



Palette Properties

Please choose a name and a package name for the new palette.

This name is the
only way to identify a palette when you use your symbols in another
application.

You can change the properties of this palette using the *Palette>Palette Properties* menu item.

Palette Name: bampalette
Package Name: ilog.views.palettes.bampalette
Description: A palette containing symbols for BAM applications.

The Palette Properties property sheet

5. Click OK.

   The Save Palette File dialog box opens.

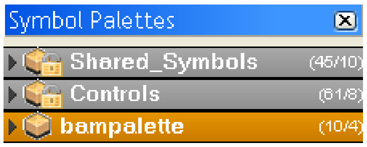
6. Navigate to the directory where you wish to save the palette.

   By default, palette files are saved in the current user's home directory.

7. Type `bampalette` in the File name field.

8. Click Save.

This palette is opened in the Symbol Palettes pane at the right of the Symbol Editor interface.



The Symbol Palettes pane

Tip: You can have multiple palettes open simultaneously in the Symbol Palettes pane.

Creating a new category

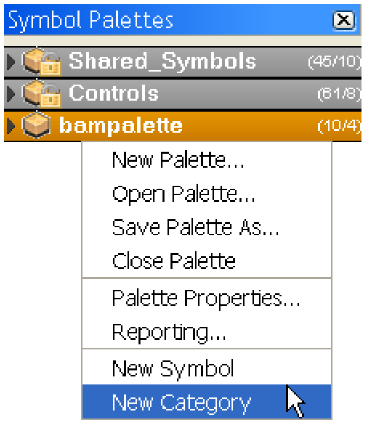
Palettes are subdivided into categories. You use categories to organize the symbols contained in a palette. All new categories are subcategories of the Symbols root category.

To create a new category:

1. Right-click the parent category in the Symbol Palettes pane.

   -- or --

2. To create a root category, right-click the header of the palette.

   Right-click the palette header to create a new root category

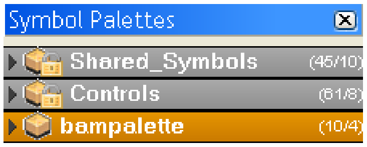
3. Select New Category in the shortcut menu.

   The Category Properties property sheet opens.

4. Type `progressbars` in the Name field.
5. Edit the Short Description and Long Description fields.
6. Click OK.

Understanding the Symbol Palettes pane

The Symbol Palettes pane displays the palettes that are open in the Symbol Editor. Each palette can be expanded as a tree. A palette is identified by a header which includes its name and a small icon.



Open palettes in the Symbol Palettes pane

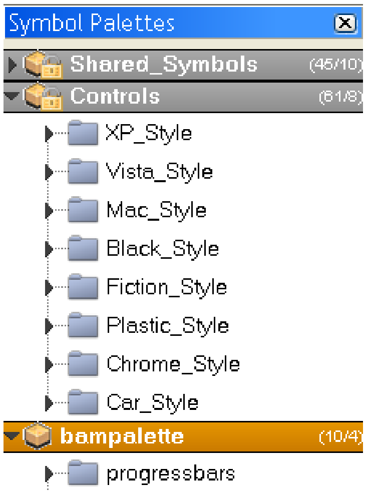
The icon represents a default palette (opened when the Symbol Editor is launched). It indicates a locked status, which means that the palette cannot be modified nor saved.

The icon represents a standard palette opened by the user.

The palette header also indicates the total number of symbols and categories in a palette. The first number is the symbol count and the second number is the category count. In the figure *Expanding the bampalette*, the Controls palette has a total of 61 symbols distributed in 8 categories.

Clicking a palette header selects the corresponding palette which becomes the new current palette. The header is highlighted (the color may vary depending on the Java™ and host platform look and feel).

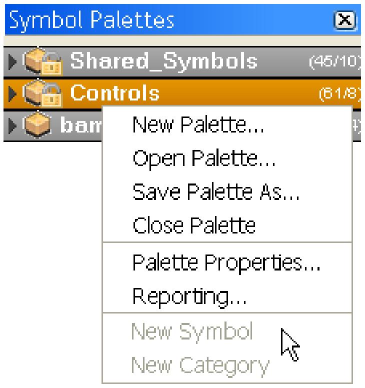
Clicking the arrow left of a palette header expands the corresponding palette tree. Note that this does not change the currently selected palette.



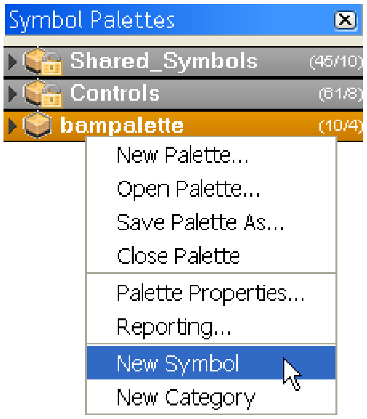
Expanding the bampalette

Select the expanded palette by clicking its header. The header is highlighted. You can further expand the categories of the selected palette by clicking their arrow.

Right-clicking a palette header gives access to a shortcut menu. This menu contains general actions that can be performed on palettes (New, Open...) as well as actions relative to the selected palette (Save As, Close, Properties, Reporting). New Symbol and New Category actions are disabled for default (locked) palettes, but not for standard palettes.



Shortcut menu for a default palette



Shortcut menu for a standard palette

Closing and opening symbols and palettes

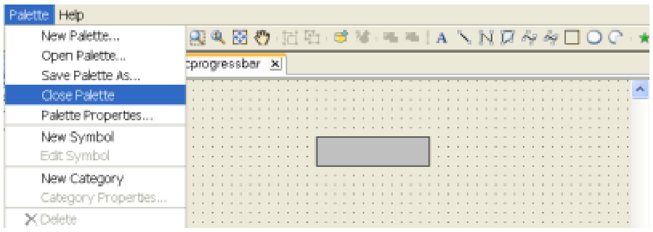
Palettes are used to store multiple symbols, and symbols are made up using objects and symbols from one or more palettes. You can open and close symbols and palettes independently.

To close a palette:

♦ Right-click the palette header in the Symbol Palettes pane, then select Close Palette from the shortcut menu.

-- or --

♦ Select the palette in the Symbol Palettes pane, then choose **Palette>Close Palette** from the menu bar.



Closing a palette

Note: Although the palette is closed, the opened symbols remain open.

To close a symbol:

♦ Select **File>Close**.

-- or --

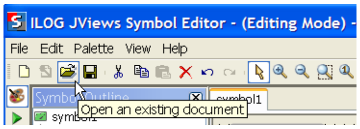
♦ Press CTRL+F4.

To open a symbol and palette:

♦ Select **File>Open**.

-- or --

♦ Click the Open an Existing Symbol button in the toolbar.


ILOG JViews Symbol Editor - (Editing Mode)
File
Edit
Palette
View
Help
Symbol Outline
symbol1
Open an existing document

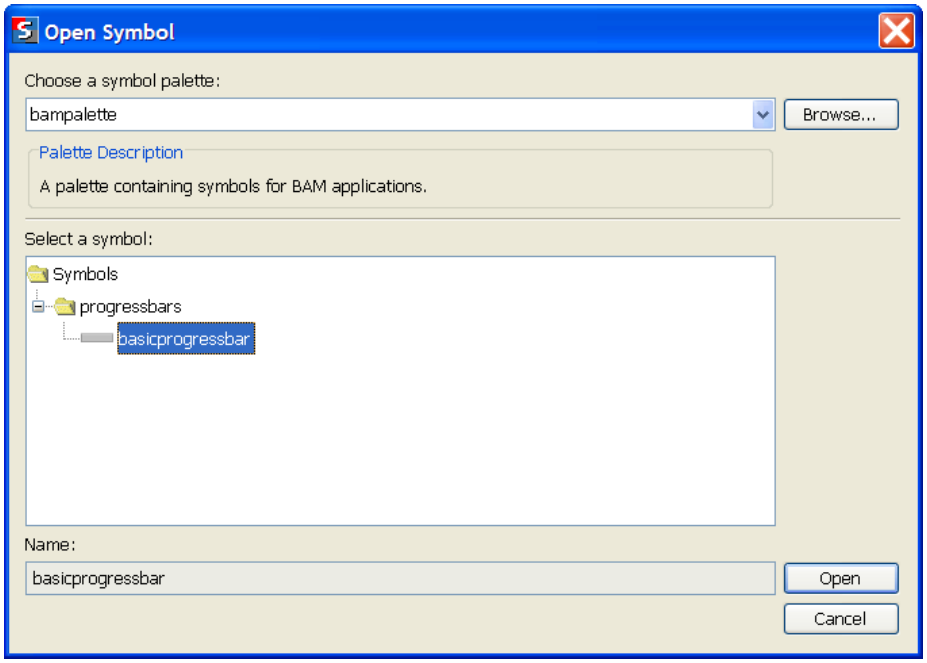
The Open an Existing Symbol button

-- or --

1. Press CTRL+O on the keyboard.

   The Open Symbol dialog box opens:

2. Click the Browse button.
3. Select `bampalette.jar`.
4. Click Open.
5. Select the `basicprogressbar` in the `progressbars` category.



The Open Symbol dialog box

6. Click Open.

The `basicprogressbar` symbol opens in a new tab in the Symbol Editor drawing pane and the `bampalette` opens in the Symbol Palettes pane.

Note: To open a symbol from a palette that is open in the Symbol Palettes pane, double-click the symbol.

Reorganizing palettes

In some situations, you may need to reorganize the content of one or more palettes by moving or copying symbols and categories within the same palette or to another one. The drag and drop feature allows you to do this.

The Symbol Palettes pane supports multiple drag and drop, which means that you can perform the drag and drop operation on several selected objects.

To select several objects:

- Press the CTRL key.

Please note that the multiple selection process may be platform-dependent.

If you press the CTRL key (depending on the platform) while dragging and dropping a selection, you will perform a copy operation: a copy of the selection is created in the destination, while the original objects remain in their initial location.

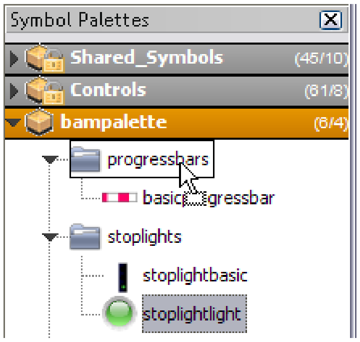
Note: You cannot drag and drop to or from a locked palette.

The examples described hereafter are applied to a single symbol, but you can do the same with a category or a set of objects (several symbols, several categories or several symbols and categories).

To move a symbol inside the same palette:

1. Select the symbol to be moved.
2. Drag it to the destination category.

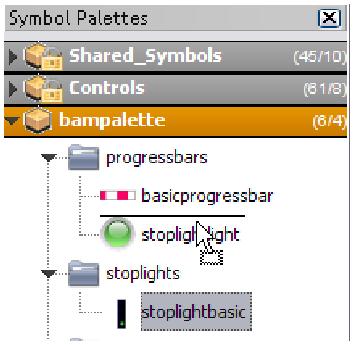
   The destination category appears surrounded by a selection rectangle.



Moving the spotlightlight symbol to the progressbars category

The symbol is added at the end of the destination category.

If the destination category is expanded, you can move the symbol at a specific location, represented by a horizontal line, inside the new category. See the following figure.



Inserting the spotlightlight symbol at a specific location

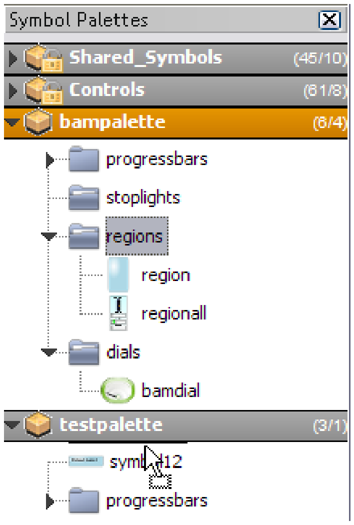
You can also reorganize symbols within their own category to match your needs.

Likewise, you can move a symbol or a category to another palette.

To move or copy a symbol or a category to a different palette:

- ♦ The destination palette is expanded.

  The process is the same as described above: you drop the selection to the destination.

  Moving the category regions to the palette testpalette

  If you drop the selection on the palette header, it will be moved or copied at the end of the palette.

-- or --

- ♦ The destination palette is collapsed.

  If you remain a couple of seconds over the palette header, the palette will expand after some short blinking. You can then proceed with the drag and drop.

Transforming symbols using parameters

Explains how to use parameters to convey some behavior to symbols.

In this section

Creating a new parameter

Parameters are used to control object transformation. That is, they control the position, size, color, or rotation of the object. You transform an object by binding one or more of its properties to parameters.

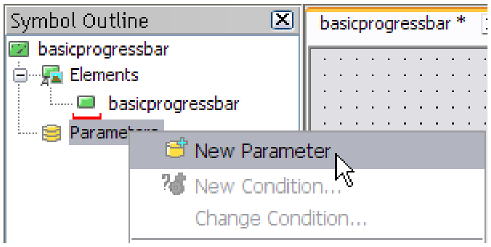
A symbol parameter can have one of the following types:

- String
- Integer
- Boolean
- Float
- Double
- Color
- Paint

Changes to the value of the parameter result in changes to the aspect of the object to which it is bound. For example, the width of the `basicprogressbar` will expand to the right as the parameter to which it is bound increases. In this way, parameters give intelligence to symbols.

To create a new parameter for the basicprogressbar symbol:

1. Open the `basicprogressbar` symbol.
2. Right-click the Parameters icon in the Symbol Outline pane.
3. Select New Parameter in the shortcut menu.



Adding a new parameter

4. Type `progress` in the Name field of the Styling Customizer
5. Press ENTER on your keyboard.

   The progress parameter is listed in the Symbol Outline pane.
6. Choose Integer from the Type list.
7. Leave the Default Value field as 0 (zero).

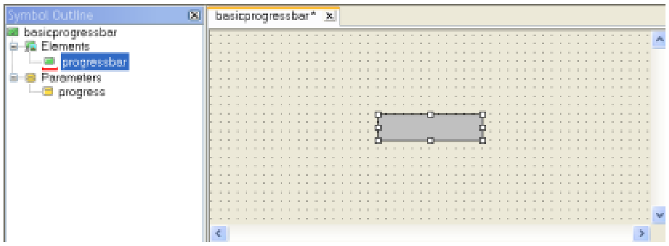
Transforming an object

Transformation occurs when an object changes aspect. This change is done by binding a parameter to one or more of the object properties. As the parameter changes value, so the object changes appearance. Parameters can be bound to one or more objects in a symbol. An object can have one or more parameters bound to the properties in the Position, Size or Rotation transformation property groups.

Note: A base element can only change its width and height properties.

To bind the progress parameter to the basicprogressbar symbol:

1. Open the `basicprogressbar` symbol.



The basicprogressbar symbol

2. Select the progressbar object in the Symbol Outline pane.
3. Click the Transform tab in the Styling Customizer.
4. Click the Bind button next to the Width field.

   The Transform Element Using Parameter dialog box opens.
5. Select progress from the Parameter list.
6. Select Compute the transformation according to the following min/max values.
7. Type `120` in Value Max.

   This is the width in pixels the progressbar object will attain when the progress parameter equals Parameter Max. That is, when progress equals 100, the progressbar will be 120 pixels wide, and when progress equals 50, the progressbar will be 60 pixels wide.
8. Click OK.

The width of the base rectangle is now bound to the progress parameter.

Note: The width of the rectangle is now set to 0, the default value of the progress parameter.

Adding flexibility using parameters

By default, an object uses static values to control its size. For example, the progressbar object created in *Working with symbols* has a minimum width of 0 and a maximum width of 120. The transformation added in *Transforming an object* still leaves the maximum at 100. To create reusable symbols, the best practice is to use parameters to control size. This way, the size of a symbol can be easily adjusted for inclusion in a symbol with different dimensions.

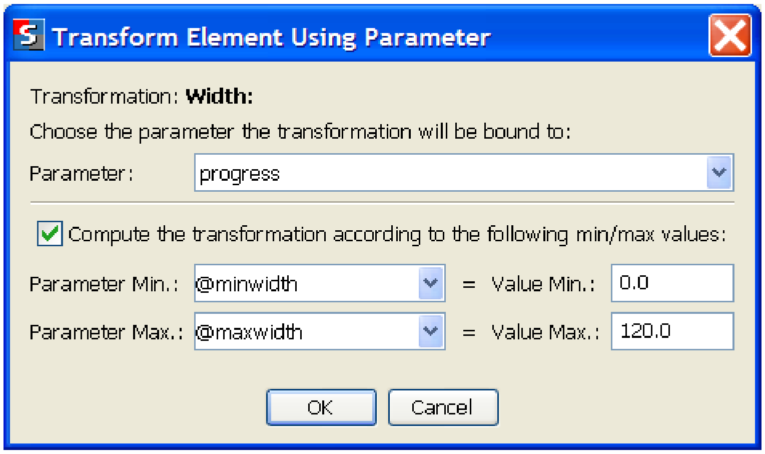
To add flexibility to the basicprogressbar symbol:

1. Open the `basicprogressbar` symbol.
2. Create a new Integer parameter, maxwidth, with a default value of 100.

   For more information, see *Creating a new parameter*.
3. Create a new Integer parameter, minwidth, with a default value of 0.

   For more information, see *Creating a new parameter*.
4. Select the progressbar object in the Symbol Outline pane.
5. Click the Transform tab in the Styling Customizer.
6. Click the Bind button next to the Width field.

   The Transform Element Using Parameter dialog box opens.
7. Select maxwidth in the Parameter Max list.
8. Select minwidth in the Parameter Min list.



Transformation: **Width:**

Choose the parameter the transformation will be bound to:

Parameter: progress

✓ Compute the transformation according to the following min/max values:

Parameter Min.: @minwidth = Value Min.: 0.0

Parameter Max.: @maxwidth = Value Max.: 120.0

Binding a transformation to parameters

9. Click OK.

Note: The width of a transformed object is calculated in function of the Parameter Max, Parameter Min, Value Max, and Value Min parameter values set in the Transform Element Using Parameter dialog box. If you take the value of Parameter Max to represent 100% of Value Max, the basicprogressbar symbol will expand to the width Value Max in function of the value of Parameter Max. That is, when the progress parameter has a value of 100, the basicprogressbar will be 120 pixels wide.

You can now set the maximum width for this object by changing either the default or the dynamic value of the maxwidth parameter.

Working in different modes

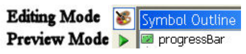
In the Symbol Editor you can work in two different modes:

- ♦ Editing mode to define your symbols
- ♦ Preview mode to test the behavior of a symbol as it would be in an application

By default, the Symbol Editor runs in Editing mode. In Editing mode, you add objects and parameters as you have been doing.

To change the Symbol Editor mode:

- ♦ Click the mode buttons in the vertical toolbar next to the Symbol Outline pane.



Mode buttons in the Symbol Editor

Testing your symbol

After having made some changes to your symbol in Editing mode, you want to test them in Preview mode.

To test the basicprogressbar symbol:

1. Click the Preview button in the vertical toolbar.
2. Change the value of the progress parameter displayed in the Symbol Parameters pane by either:
    - ♦ clicking the arrows on the right of the progress field, or
    - ♦ typing a value in the progress field.

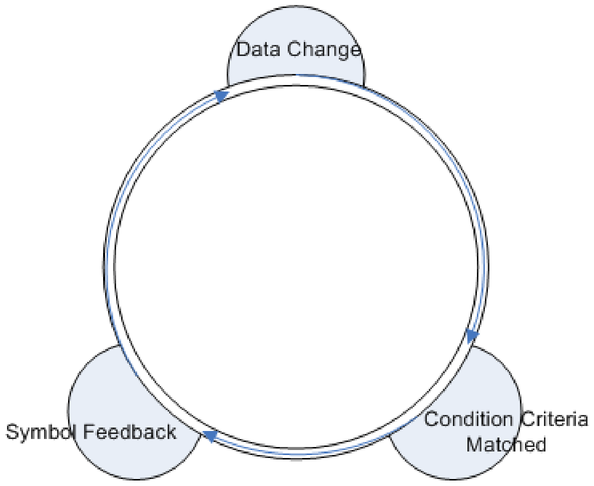
You see the progress bar extend to the right when the value of the parameter increases and retracts to the left as the value decreases.

Note: When you return to Editing mode, the transformed object is displayed with the last value given to its bound parameter while in Preview mode. For example, if you set progress to 50, the progressbar object will be 60 pixels wide when you return to Editing mode.

Transforming symbols using conditions

In the *Transforming symbols using parameters* section you were shown how to add intelligence to symbols with parameters that represent changes in a real world situation.

A condition triggers an event in a symbol whenever specific criteria are matched. The event provokes the symbol to change its representation to display the parameter change visually, for example, when the object is selected or when a parameter has a specific value. As with parameters, a condition can be bound to multiple objects in a symbol.



The condition workflow

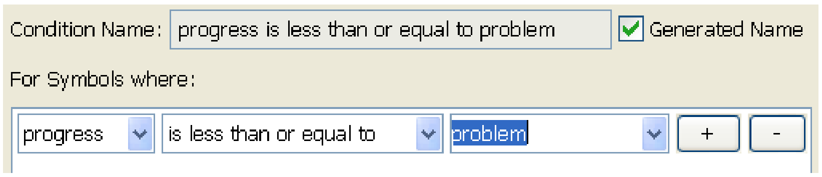
You will use conditions to show a state in the `basicprogressbar` symbol using a transformation. If progress is below a certain percentage, the progress bar will be displayed in red.

To transform the basicprogressbar symbol:

1. Open the `basicprogressbar` symbol.
2. Create a new Integer parameter, problem, with a default value of 35.

   For more information, see *Creating a new parameter*.
3. Right-click the progressbar object in the Styling Outline pane.
4. Select New Condition in the shortcut menu.

   The New Condition dialog box opens.
5. Use the lists in the For Symbols where area to set the condition to `progress is less than or equal to problem`.


Condition Name: progress is less than or equal to problem ✓ Generated Name

For Symbols where:

progress | is less than or equal to | problem

Setting a new condition

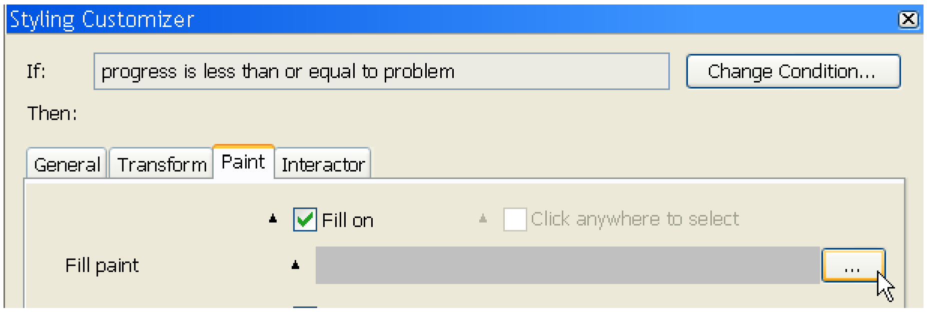
Note: A name for the condition is generated automatically. You can change this to a personalized name by deselecting Generated Name and typing a new name in the Condition Name field.

6. Click OK.

The new condition is added under the progressbar object in the Symbol Outline pane. The Styling Customizer displays the properties for this object when the condition is met.

7. Click the Paint tab.

8. Click the button to the right of the Fill paint field.



If: progress is less than or equal to problem

✓ Fill on

Click anywhere to select

The button to open the Paint Editor

The Paint Editor opens.

9. Select the RGB tab in the Color tab.

10. In the RGB tab, set the color to 255, 0, 51.

11. Click OK.

12. Click Apply in the Paint Editor.

The `basicprogressbar` symbol now changes color to show changes in progress.

13. Test your symbol using the procedure explained in *Testing your symbol* to see the transformations.

Using link-in and link-out ports

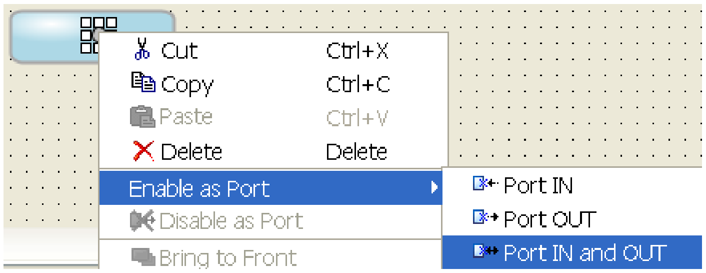
By default, as a symbol moves in a diagram, the link joining it to another symbol connects to the port closest to the other symbol. A symbol can be designed with special subcomponents that can be used as link-in or link-out ports. Links connected to link-in and link-out ports stay bound to these ports as the different symbols move. A link starts from an output port and ends on an input port. The elements are identified as ports thanks to their name: a prefix indicates the type of port.

- port_in_: link-in only
- port_out_: link-out only
- port_in_out_: both link-in and link-out ports

Note: When you create a port object, it is automatically renamed to contain the port_in, port_out or port_in_out_ prefix according to the port type. However, the prefix is visible in the CSS view only. It is not visible in the Name field of the Styling Customizer nor in the Symbol Outline.

To create a link port subcomponent:

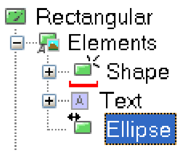
1. Create a new rectangular symbol by double clicking the Rectangular symbol in the Basic category of the Shared_Symbols palette.
2. Add a freehand ellipse to the center of the Shape object.
3. Right-click the ellipse and select **Enable as Port>Port IN and OUT** in the shortcut menu.



Create a new port subcomponent

Note: You can also create a port by right-clicking an object in the Symbol Outline pane.

The link port type of an object is indicated in the Symbol Outline pane.



A double headed arrow indicates a port_in_out_ object.

The following table shows the meaning of the different port indicators in the Symbol Outline pane.

Link port indicators

Icon	Meaning
←	link-in only
→	link-out only
↔	both link-in and link-out

For information on how to use link ports in the Dashboard Editor, see Link-in and link-out ports.

To change the link port type for an object in a symbol:

1. Right-click the object in the Symbol Outline pane.
2. Select the new link port type in the shortcut menu.

For example, **Enable as Port>Port IN**.

To remove the link port status of a symbol subcomponent:

1. Right-click the object in the Symbol Outline pane.
2. Select **Disable as Port** from the shortcut menu.

Generating and using symbol reports

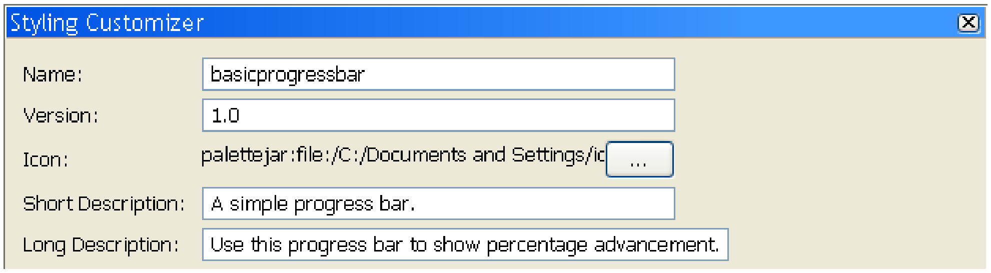
Explains how to generate and use reports on palette symbols.

In this section

Generating a report for all symbols

Symbols are designed to be used by a chain of different tools, the Symbol Editor, the Dashboard Editor, the Designer and a variety of custom applications. The Symbol Editor reporting feature allows you to generate documentation from the palettes currently open in the Symbol Editor. The symbol report is generated in HTML format; it can be viewed either in the Symbol Editor, or, when a report is exported, in a standard Web browser.

The symbol report is based on the descriptions you wrote about the symbols and their parameters when they were created.



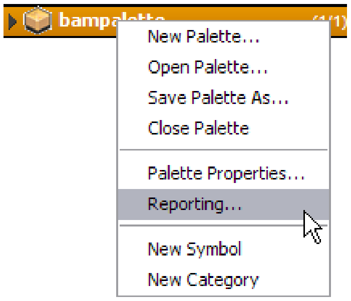
Name:	basicprogressbar
Version:	1.0
Icon:	palettejar:file:/C:/Documents and Settings/ic
Short Description:	A simple progress bar.
Long Description:	Use this progress bar to show percentage advancement.

A symbol description

To view a symbol report:

1. Open the palettes you want to document in the Symbol Editor.
2. In the Symbol Palettes pane, right-click the header of the palette you wish to document.

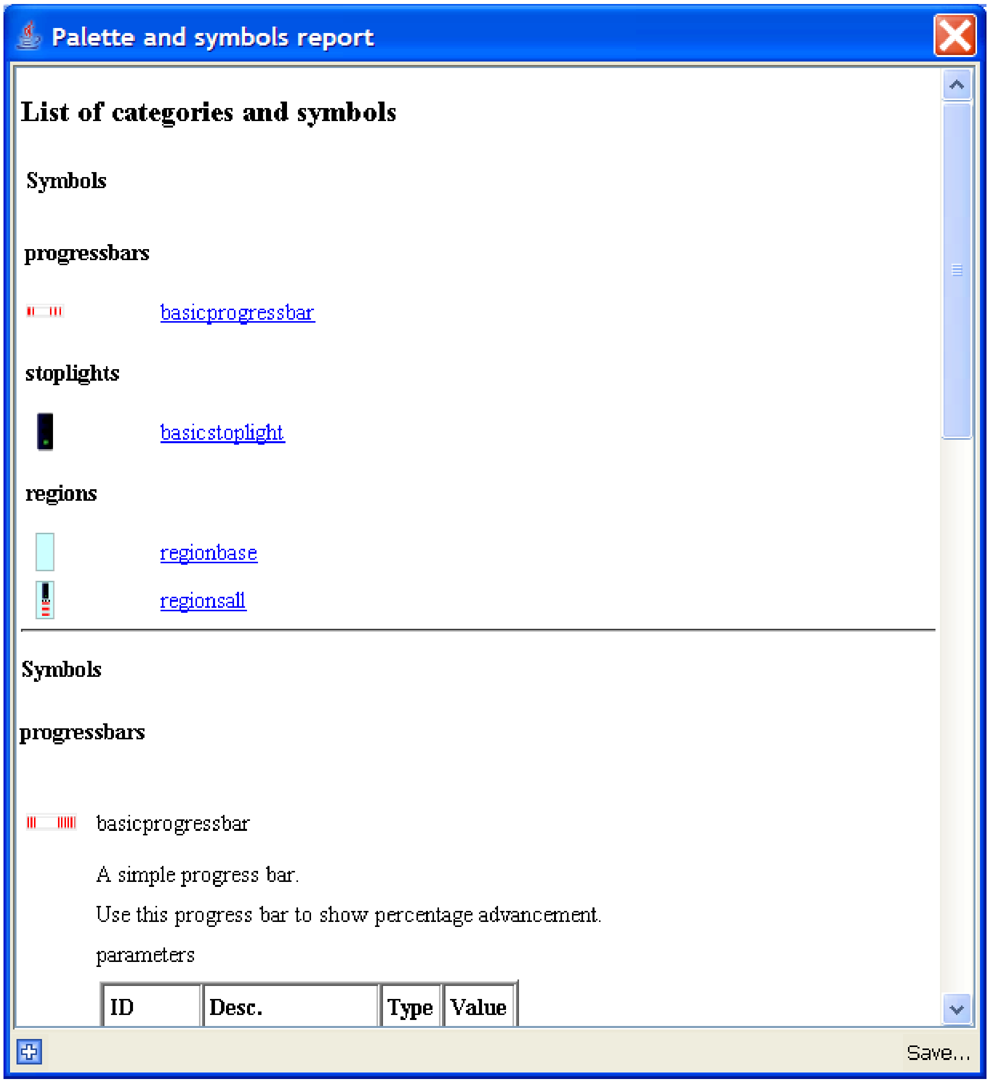
   A shortcut menu appears.

   The Reporting shortcut menu

3. Select Reporting in the shortcut menu.

   The Palette and symbols report window opens.



Palette and symbols report

List of categories and symbols

Symbols

progressbars

basicprogressbar

stoplights

basicstoplight

regions

regionbase

regionsall

Symbols

progressbars

basicprogressbar

A simple progress bar.

Use this progress bar to show percentage advancement.

parameters

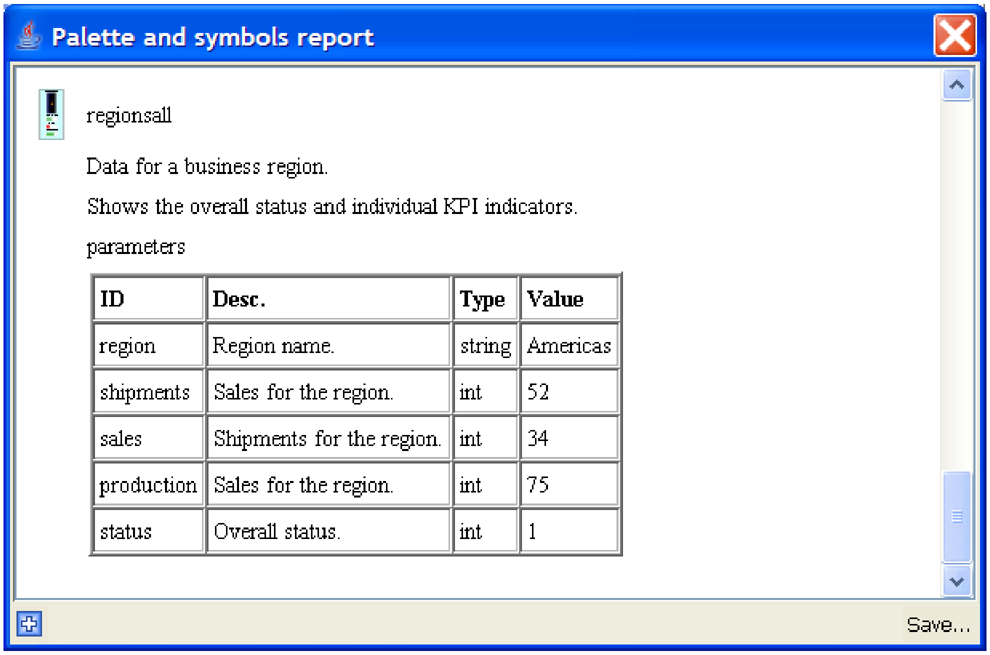
ID	Desc.	Type	Value

The Palette and symbols report window

As shown in the figure above, the page opens to show an overview of all the symbols contained in the palette.

To see detailed information for a symbol:

♦ Click the symbol name in the Palette and symbols report window.



regionsall

Data for a business region.

Shows the overall status and individual KPI indicators.

parameters

ID	Desc.	Type	Value
region	Region name.	string	Americas
shipments	Sales for the region.	int	52
sales	Shipments for the region.	int	34
production	Sales for the region.	int	75
status	Overall status.	int	1

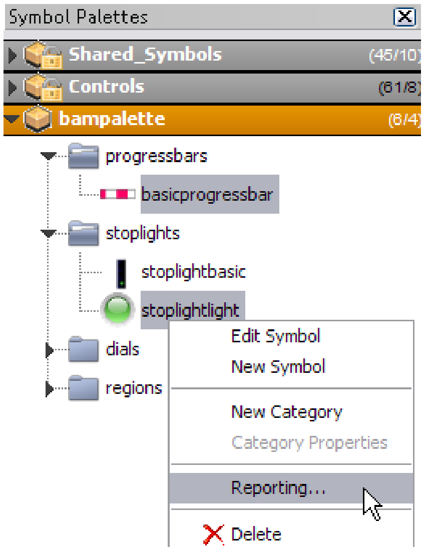
A detailed report for a single symbol

Filtering a report to show specific symbols

You can filter the report to show only the details of specific symbols.

To create a report for only a few symbols:

1. In the Symbol Palettes pane, select the symbols you wish to document.
2. Right-click one of the selected symbols in the Symbol Palettes pane. The shortcut menu opens.
3. Select Reporting in the shortcut menu.



The shortcut menu

The Palette and symbols report window opens with a report concerning only the selected symbols.

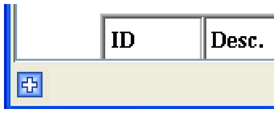
Navigating through different palettes in a report

To change the report to display information about the different palettes open in the Symbol Editor:

1. Open a standard palette report in the Palette and symbols report window.

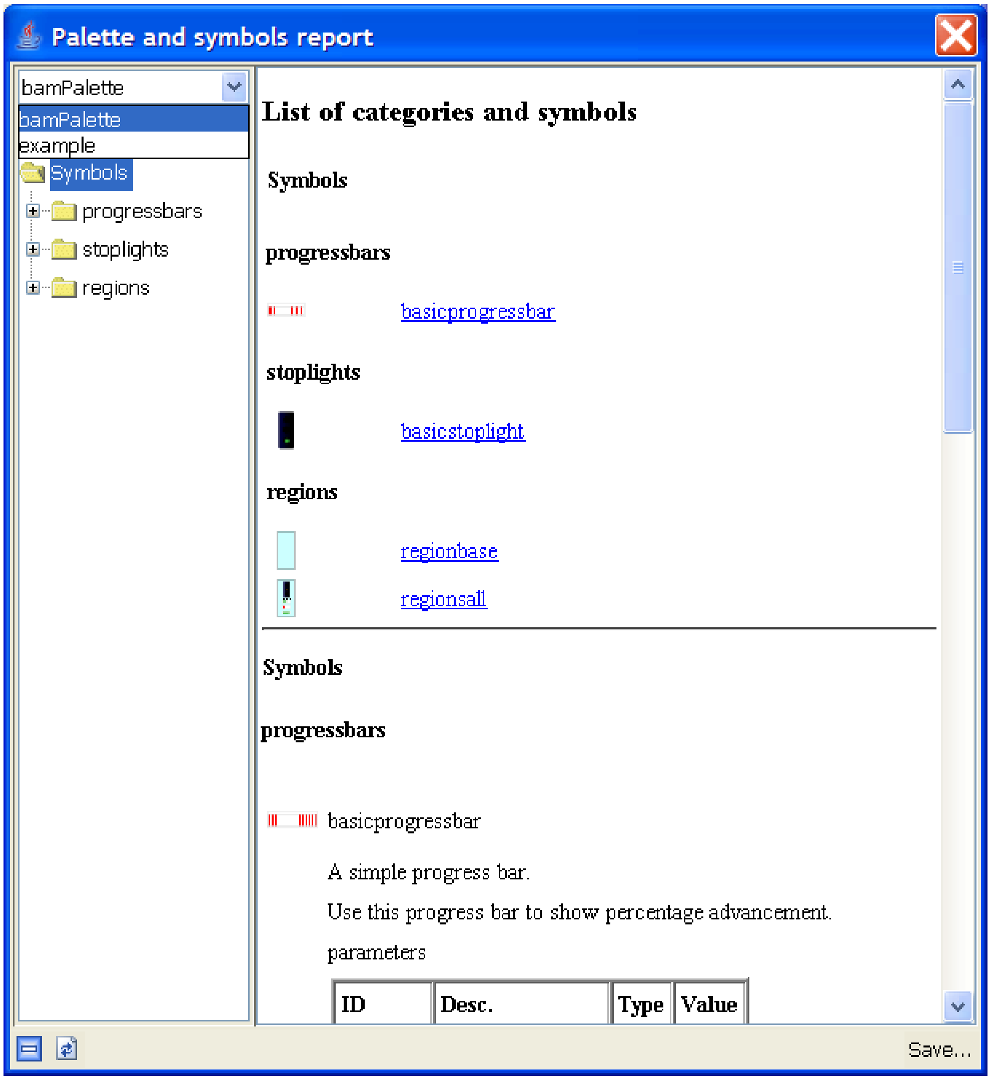
   For more information see *Generating a report for all symbols*.

2. Click the plus icon on the bottom left of the Palette and symbols report window.

   The Expand button

   The extended report pane opens.



List of categories and symbols

Symbols

progressbars

basicprogressbar

stoplights

basicstoplight

regions

regionbase

regionsall

Symbols

progressbars

basicprogressbar

A simple progress bar.

Use this progress bar to show percentage advancement.

parameters

ID	Desc.	Type	Value

The expanded Palette and symbols report window

As shown in figure above, the expanded pane in the Palette and symbols report window displays a list of the categories and symbols contained in the current palette.

To view the report for a different palette:

♦ Select the palette you want in the palette list.

To return to a normal report:

♦ Click the contract button (minus icon).

The palette report Contract button

Searching for a symbol in the open palettes

The expanded report panel provides a search facility. This facility searches for an occurrence of a string in all palettes open in the Symbol Editor.

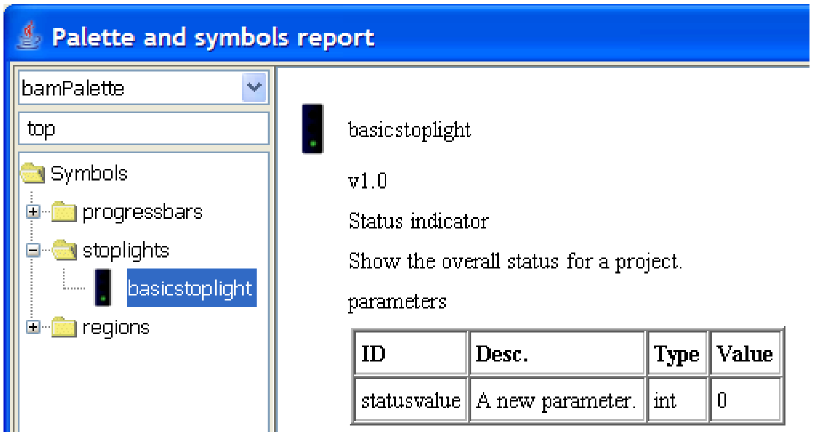
To search for a symbol in the palette list:

1. Open the expanded pane in the Palette and symbols report window.

   For more information see *Navigating through different palettes in a report*.

2. Type part of the name of the symbol you are looking for in the search field.

3. Press ENTER.



basicstoplight

v1.0

Status indicator

Show the overall status for a project.

parameters

ID	Desc.	Type	Value
statusvalue	A new parameter.	int	0

The palette report Search facility

Refreshing a palette report

To refresh the information in a palette report:

♦ Click the Refresh button in the Palette and symbols report window.

The palette report Refresh button

Saving a report

To save the report for the currently selected palette to HTML:

1. In the Palette and symbols report window, click Save.

   The palette report Save button

   The Save window opens.

2. Browse to the directory in which you want to save the report.
3. Type a name in the File name field.

   If you leave this field blank, the file will be called index.html.

4. Click Save.

The BAM Dashboard example

Shows how to create symbols that will be integrated into a Business Activity Monitoring dashboard.

In this section

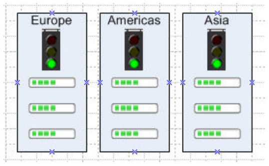
About Business Activity Monitoring

Business Activity Monitoring (BAM) covers a wide range of applications and services. The concept is to offer monitoring and supervision functions to track performance issues for a business. This is usually based on an integration layer and a reporting environment. The integration part gathers important technical and business data, aggregates and correlates information, and delivers Key Performance Indicators. The reporting part enables end users to select, present, declutter and analyze the data, based on importance and relevance.

BAM offers operational managers a way to understand how their business is running and to adjust resources dynamically in order to fulfill their objectives, minimize latency and increase revenue.

Definition of the dashboard

The management request is to create a dashboard to show Key Performance Indicators, namely Production, Supply and Sales. Each indicator will be displayed as a percentage of total capacity. The overall performance for each region will also be illustrated using a stoplight symbol. The following figure shows the dashboard definition.



A dashboard definition

Definition of the dashboard symbols

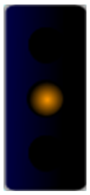
To create the BAM dashboard shown in figure *A dashboard definition*, you will need to create the following symbols:

- ♦ A progress bar symbol
- ♦ A stoplight symbol
- ♦ A region symbol

The progress bar will display the percentage of overall capacity being delivered. This percentage is displayed by a bar which changes width and color to show the percentage. The percentage is changed by altering the value of a single integer parameter, progress.

Progress bar states

The stoplight symbol indicates the overall state of key performance indicators. It can display three possible states: good, warning or problem. The state displayed represents the levels of key performance indicators. It is changed by altering the value of a single integer parameter, statusvalue.



The warning state in the stoplight symbol

The region symbol uses the progress and stoplight symbols to display key performance indicators and overall status for a business region. It contains the following parameters:

- ♦ region - a string containing the name of the region
- ♦ shipments - the percentage of products shipped
- ♦ sales - the percentage of the sales target achieved
- ♦ production - the percentage production quota achieved
- ♦ status - the overall status of the region in regard to shipments, sales and production

Creating the dashboard symbols

Shows how to create the three progress bar symbols and the stoplight symbol that compose a region symbol.

In this section

Customizing the progress bar

In this section, you will extend the progress bar made in *Working with symbols* to match the progress bar defined in *Definition of the dashboard*.

In *Transforming symbols using parameters* you were shown how to define the transformation necessary for this symbol to display percentage progress. You will now concentrate on updating object graphic styles to match the look and feel defined in the dashboard definition.

Graphic styles define the appearance of an object. They are customized using the Paint tab in the Styling Customizer. The appearance of an object is separated into two parts: the fill style and the stroke style. The fill style is the graphic rendering of anything inside an object, the stroke style defines the object border.

To extend the progress bar, you will proceed in four stages:

♦ *Changing the fill style*

♦ *Changing the stroke*

♦ *Adding shapes to a symbol*

♦ *Changing symbol layers*

Changing the fill style

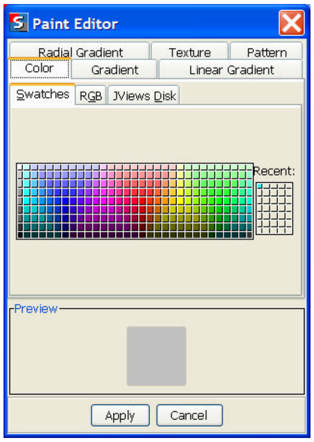
To change the fill style for the basicprogressbar, use the Paint Editor.

1. Open the basicprogressbar symbol from the bampalette.

   For more information, see *Closing and opening symbols and palettes*.

2. Select the progressbar object in the Symbol Outline pane.
3. Click the Paint tab in the Styling Customizer.
4. Click the button on the right of the Fill Paint field.

   The Paint Editor opens.



The Paint Editor

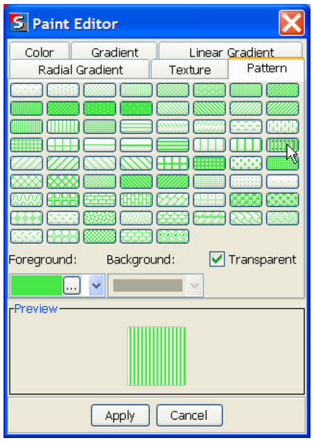
As shown in the figure above, the Paint Editor contains tabs for controlling the color, gradient, texture, and pattern of the object. The color can be set using one of the following options:

- Swatches
- RGB (Red, Green, Blue) codes
- JViews Disk

The gradient can be set using linear, radial or standard tools.

5. Click the Pattern tab.
6. Click the button in the Foreground field.

   The Color dialog box opens.
7. Click the RGB tab.
8. Type `71, 231, 71` in the Red, Green and Blue fields respectively.
9. Click OK.
10. Select the bars pattern as shown in the following figure.



The Paint Editor pattern chooser

11. Deselect Transparent.
12. Click the button in the Background field.

    The Color dialog box opens.
13. Select the color white in the top left of the Swatches color chooser.
14. Click OK.
15. Click Apply.

The color scheme of the progress bar now matches the dashboard definition in figure *A dashboard definition*. Before continuing, save the changes you have made to the symbol by pressing the CTRL+S keys.

Note: The graphic manipulation shown in this example is very basic. For information about more advanced graphics, see *Exploiting advanced graphics features*.

Changing the stroke

The stroke refers to the border surrounding an object.

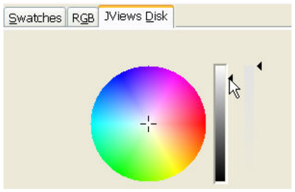
To change the stroke for the basicprogressbar symbol:

1. Open the basicprogressbar symbol.
2. Select the progressbar object in the Symbol Outline pane.

   The Styling Customizer opens on the last tab you used, in this case, the Paint tab.
3. Click the button next to the Stroke paint field.

   The Paint Editor opens.
4. Select the JViews Disk tab.

5. Slide the tone changer to a light gray tone.



The JViews Disk color chooser

6. Click Apply.

Adding shapes to a symbol

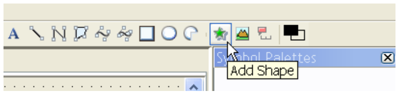
In the dashboard definition, the progress bar advances on a white background. To create this effect, you need to make a background object with the same width and length as the fully extended progress bar.

1. Open the basicprogressbar symbol from the bampalette.

   For more information, see *Closing and opening symbols and palettes*.

2. Click the Add Shape button in the toolbar.

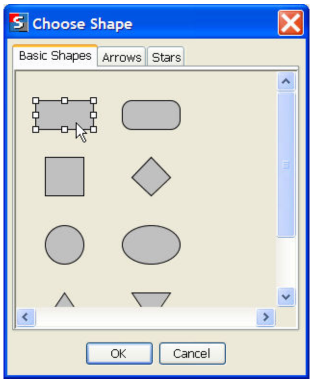
   The Choose Shape dialog box opens.



The Add Shape button

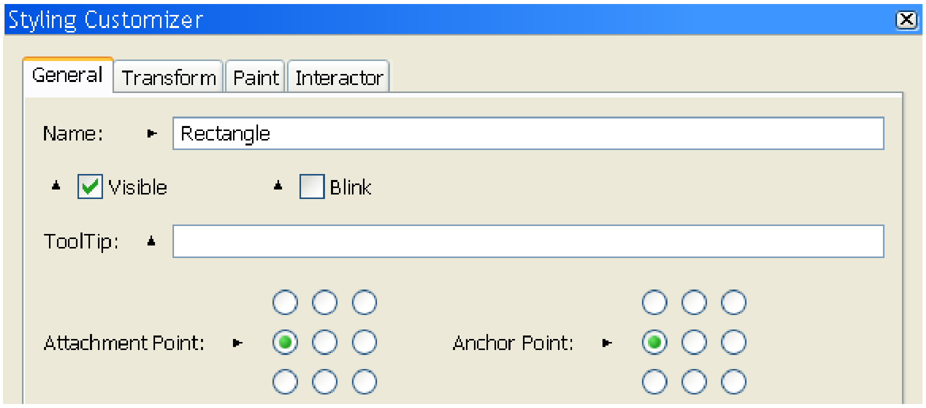
3. Select the Rectangle shape and click OK.

   The Rectangle object is added to your symbol, it is the current selected object.



The Choose Shape dialog box

4. In the General tab, change the Attachment and Anchor points to match the following figure.



Name: Rectangle

[✓ Visible] Blink

ToolTip:

Attachment Point: ● (middle left)

Anchor Point: ● (middle left)

Change the attachment and anchor points

The Rectangle shape is now aligned with the left of the basicprogressbar.

Note: The direction in which a transformation occurs is controlled by an object's attachment point. For example, if the object is attached to the base object from the left as in figure *Change the attachment and anchor points*, changes in width are reflected by the object expanding to the right of the attachment point.

5. In the Transform tab, type `30` in the Height field.
6. Type 120 in the Width field.
7. In the Paint tab, change the Fill paint color to white using the Paint Editor.

   For more information, see *Changing the fill style*.
8. Set the Stroke paint to gray.

   For more information, see *Changing the stroke*.

The basicprogressbar symbol now contains all the objects necessary to match the dashboard definition.

Changing symbol layers

The arrangement of the objects in a symbol is calculated in reference to the base object, in terms of position and layer. As shown in *Adding shapes to a symbol*, you can change the relative position of an object. Symbols are made up of objects stored in layers. The base object is the bottom layer, on which all other objects in the Symbol are stacked. Using the Symbol Editor, you can change the level of an object in the layer stack.

In the basicprogressbar symbol, the background image is currently in a layer higher than the progress bar. If you test the symbol, you will see that, as the value of the progress parameter increases, the background rectangle moves to the right.

To rectify this behavior, make the background rectangle the base symbol:

1. Right-click the Rectangle object in the Symbol Outline pane.
2. Click **Send to Back** in the shortcut menu.
3. Click OK in the Confirm New Base dialog box.
4. Save the symbol.

The basicprogressbar symbol is now complete. Test the symbol using the procedure described in *Testing your symbol*.

Creating the stoplight symbol

The dashboard description found in *Definition of the dashboard* requires a stoplight symbol to show the overall status for a region. To achieve this, you need to follow these steps:

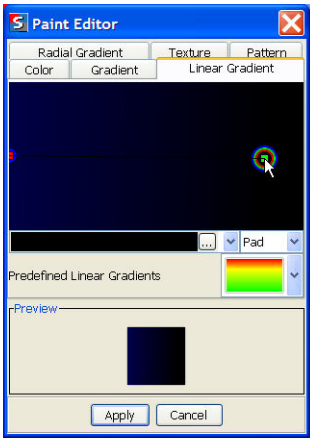
- *Creating a base object*
- *Using shapes to create a light template*
- *Aligning and grouping multiple objects*
- *Setting allowed values for a parameter*
- *Setting conditions on an object*

Creating a base object

In this section you will be shown how to use object gradients to create the stoplight symbol background.

1. Open the bampalette.
2. Create a new symbol by pressing CTRL+N.
3. Select and delete the Text object.

   For more information, see *Deleting an object*.
4. Right-click the Shape object.
5. Select **Flip or Rotate>Rotate Left** in the shortcut menu.
6. Select the Paint tab in the Styling Customizer.
7. Click the button in the Fill Paint field.
8. Change the setting in the Color>RGB tab to 0,0,70.
9. Select the Linear Gradient tab.
10. Move the right handle to the middle of the gradient field, as shown in the following figure.



The linear gradient tab

11. Click Apply.

    The stoplight base object is now complete.

12. Create a category `stoplights` in the bampalette.
13. Save this symbol as `stoplightbasic` in the stoplights category.

Using shapes to create a light template

The stoplightbasic symbol requires three lights to show the three states allowed. To do this, you create a single light object that is used as a template for the other two lights.

1. Open the stoplightbasic symbol from the bampalette.
2. Open the Choose Shape dialog box by clicking the Add Shape button as shown in figure *The Add Shape button*.
3. Select the circle shape.
4. Click OK.

   The circle is added to the symbol.

5. In the Transform tab, set the width and height of the circle to 20.
6. In the Paint tab, set the Fill Paint color of the circle to black.

   See *Changing the fill style* for more information.

7. Change the name of the circle object to `light`.

   See *Performing basic object customization* for more information.

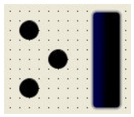
Aligning and grouping multiple objects

In the *Using shapes to create a light template* section, you created a single black light. You will now use this shape as the template for the other lights.

To create multiple lights:

1. Open the stoplightbasic symbol from the bampalette.
2. Right-click the light object.
3. Select Copy from the shortcut menu.
4. Press CTRL+V twice.

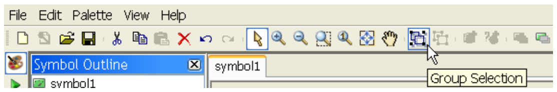
   You now have three light objects: light, light2, light3.
5. Move the light objects so they are aligned next to the base object, as shown in the following figure.

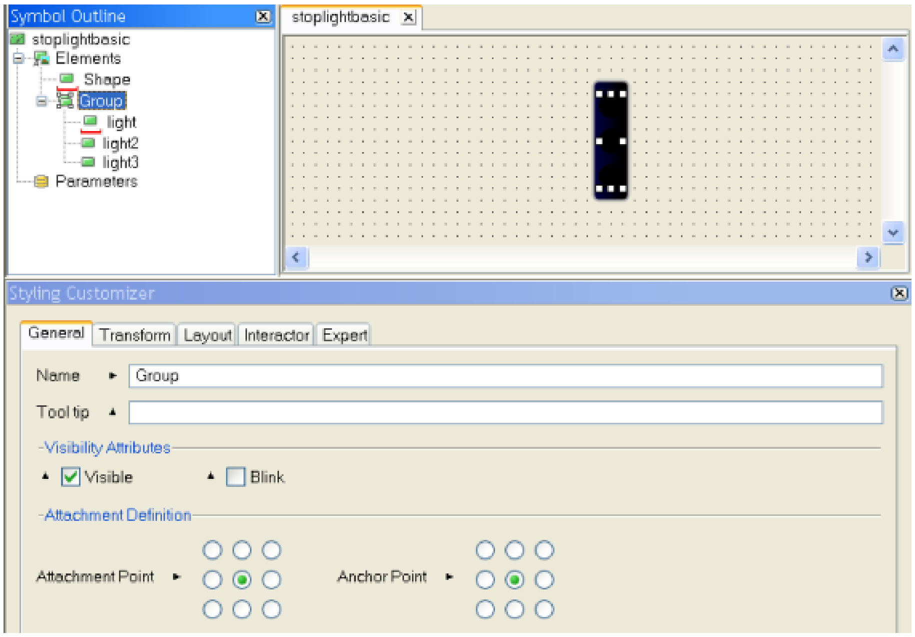
   Align objects
6. Select all three lights.

   For more information, see *Selecting an object*.
7. Right-click one of the lights.
8. Select **Align or Distribute>Align Horizontal Center** in the shortcut menu.
9. Select **Align or Distribute>Distribute Vertically** in the shortcut menu.

   The lights are now perfectly aligned and distributed.
10. Click the Group Selection button in the toolbar.

    The Group Selection button
11. In the General tab of the Styling Customizer, set the attachment and anchor points to the center point as shown in the following figure.



Name: Group

Visible [✓]

Attachment Point: centre

Anchor Point: centre

Aligning objects in the stoplight symbol

12. Save the stoplightbasic symbol.

Note: Each group of objects has a base object separate from the symbol base object.

Setting allowed values for a parameter

The stoplight symbol must always have one active light to display the status of the activity being monitored. This is done by setting conditions on the value of a parameter. A stoplight has three possible states:

- Problem
- Warning
- Good

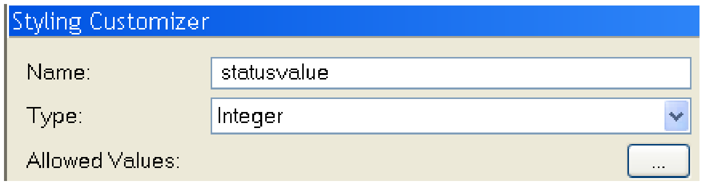
These states are created by limiting a parameter to values relating only to the states. As the parameter changes value, conditions set on individual objects change their appearance.

To create a parameter with only specific values allowed:

1. Open the stoplightbasic symbol from the bampalette.
2. Create a new Integer parameter, `statusvalue`.

   For more information, see *Creating a new parameter*.

3. Select the statusvalue parameter in the Symbol Outline pane.
4. Click the Allowed Values button in the Styling Customizer.



Name: statusvalue
Type: Integer
Allowed Values: ...

The Allowed Values button

The Value Set Editor opens.

5. Click Add.
6. Type `0` in the Value field.
7. Type `Good` in the Name field.
8. Click OK.
9. Add the values 1, Warning, and 2, Problem, by repeating steps 5 to 8.
10. Click Apply.
11. Set the Default Value for statusvalue to Good in the Styling Customizer.

You see the list of values in the Allowed Values field.

Setting conditions on an object

A Condition is used to transform the object or objects to which it is bound when:

♦ a parameter changes to a specific state

-- or --

♦ the object is selected

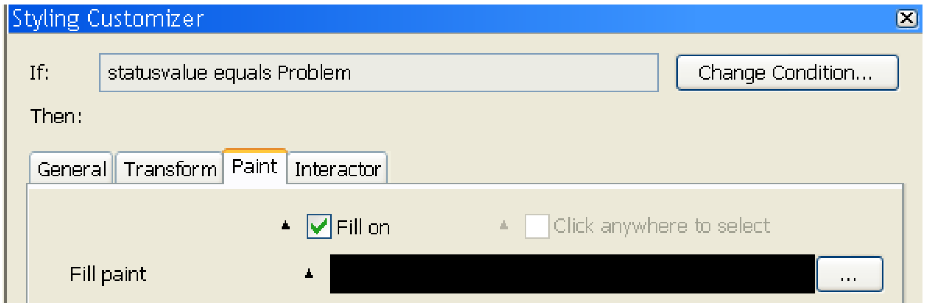
You will use conditions to change the colors of the stoplights as the value of the statusvalue parameter changes.

To set a condition on an object:

1. Open the stoplightbasic symbol.
2. Right-click the light object in the Symbol Outline pane.
3. Select New Condition from the shortcut menu.

   The New Condition dialog box opens.
4. Set the condition in the For Symbols pane to read, where `statusvalue, equals, Problem`.
5. Click OK.

The new condition is added under the light object in the Symbol Outline pane. The Styling Customizer displays the properties for this object when the status problem condition is met.



Properties for a condition

Note: The settings you apply in the Styling Customizer for a condition are applied only when the condition is fulfilled.

6. In the Paint tab, click the Ellipsis button next to the Fill paint field.

   The Paint Editor pops up.

7. Set the RGB settings in the Color tab to 255, 0, 0 and press ENTER.

   See *Changing the fill style* for more information.

8. Select the Radial Gradient tab.

9. Click Apply.

10. Repeat steps 2 to 8 for the light2 and light3 objects using the following values.

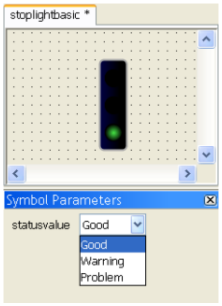
Conditions

Object	Condition name	Condition Setting	RGB Value
light2	status warning	statusvalue equals warning	255,153,0
light3	status good	statusvalue equals Good	71,231,71

The stoplightbasic symbol is now complete.

11. Run your symbol in Preview mode to test its functionality.

    For more information, see *Testing your symbol*.



The stoplightbasic symbol

Tip: The green status Good light is active by default when you test the stoplightbasic symbol. This is because the default value for the statusvalue parameter is Good. To change this, select the statusvalue parameter in the Symbol Outline pane and change the state of the Default Value list in the Styling Customizer.

Creating the region symbol

The Region symbol displays the Key Performance Indicators for a sales region. It is made up of the progress bar and stoplight symbols created in *Customizing the progress bar* and *Creating the stoplight symbol*. The next sections describe:

♦ *Creating a background object*

♦ *Adding symbols to a symbol*

♦ *Adding text to a symbol*

Creating a background object

1. Open the bampalette.
2. Click the New Symbol button in the toolbar.

   A new symbol is created.
3. Click the Rectangle button in the toolbar.
4. Drag the mouse in the drawing pane to create the rectangle.
5. In the Transform tab of the Styling Customizer, set the rectangle Width and Height fields to 130 and 300 respectively.

   See *Performing basic object customization* for more information.
6. In the Paint tab, set the Fill Paint style to RGB 204,255,255.

   See *Changing the fill style* for more information.
7. Delete the Shape and Text objects.

   For more information, see *Deleting an object*.
8. Create a category, `regions`, in the bampalette.

   For more information, see *Creating a new category*.
9. Save the symbol as regionbase in the regions category of the bampalette.

Adding symbols to a symbol

To match the dashboard in *Definition of the dashboard*, the Region symbol needs three progress bars to demonstrate shipments, sales and production. You will use the technique shown in *Aligning and grouping multiple objects* to make multiple progress bars.

1. Open the regionbase symbol.
2. Drag the basicprogressbar symbol from the Symbol Palettes pane onto the regionbase symbol.
3. Right-click the basicprogressbar object in the drawing pane.
4. Select Copy from the shortcut menu.
5. Press CTRL+V twice.

You now have three progressbar objects: basicprogressbar, basicprogressbar2, basicprogressbar3.

6. Rename the three progressBar symbols to the following:

   ♦ shipments

   ♦ sales

   ♦ production

   See *Performing basic object customization* for more information.

7. Align, group and distribute the shipments, sales and production objects on the region background to resemble figure *The region symbol*.

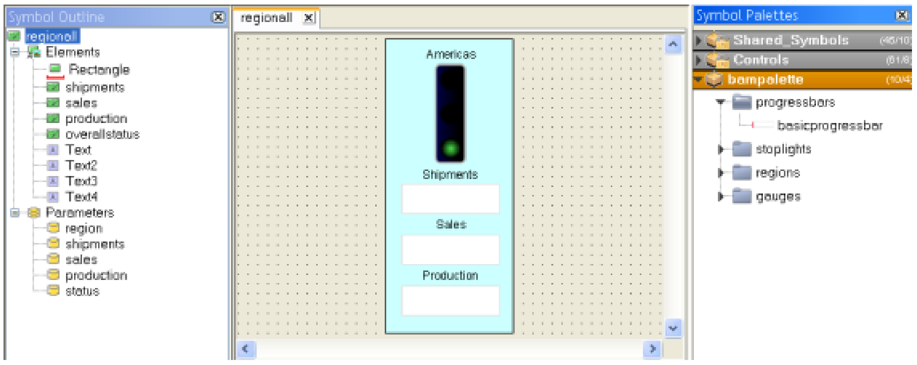
   For more information, see *Aligning and grouping multiple objects*.

8. Drag the stoplightbasic symbol onto the Rectangle object.

9. Rename stoplightbasic to overallstatus.

10. Align and group the objects in the symbol.

11. Save the symbol as regionall in the regions category of the bampalette.



The region symbol

12. Test the regionall symbol.

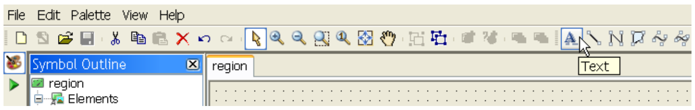
Adding text to a symbol

When you have multiple indicators in a symbol, use labels to show the data represented by the different symbols.

To display which Key Performance Indicator is being displayed:

1. Open the regionall symbol.

2. Click the Text button in the toolbar.



The Text button

3. Click above the shipments progress bar.

   A text area is created.

4. Type `shipments` in the text area.
5. Repeat steps 2 to 4 for the sales and production progress bars.
6. Create a new String parameter called region.

   For more information, see *Creating a new parameter*.

7. Set the region parameter to have the following allowed values:

   - Americas
   - Europe
   - Asia

     For more information, see *Setting allowed values for a parameter*.

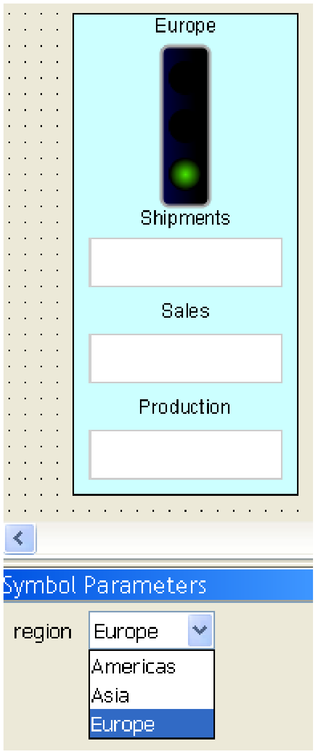
8. Create an empty text area above the overallstatus object.

   To do this, press the space bar after creating the text area.

9. Click the Text tab in the Styling Customizer.
10. Right-click the Label field and select Enter an Expression in the shortcut menu.
11. Press backspace until the list of values appears, then select `@region` from the list.
12. Press ENTER.
13. Set Alignment to Center.

    This text field now displays the default value of the region parameter.

14. Align all the symbols and objects in the regionall symbol.
15. Save and test the regionall symbol.



region: Europe
- Americas
- Asia
- Europe

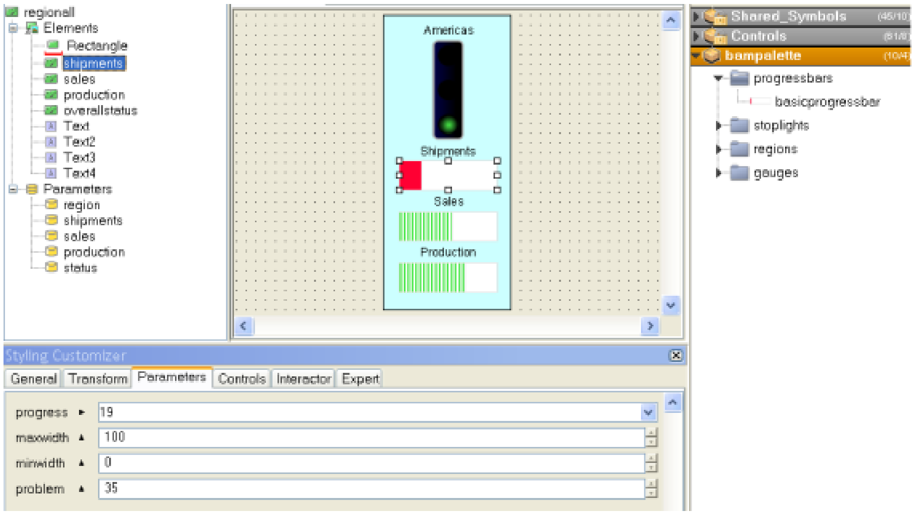
The improved regionall symbol

Connecting symbol parameters using expressions

The regionall symbol displayed in figure *The improved regionall symbol* is composed of three different symbols used to represent different forms of data; this type of symbol is referred to as a composite symbol. You can change the parameter values for symbol objects contained in the region symbol statically.

To change the value of the parameter in the design view:

1. Select the symbol whose parameter value you wish to change.
2. Open the Parameters tab in the Styling Customizer.
3. Change the value of the parameter.



progress 19
maxwidth 100
minwidth 0
problem 35

Setting a symbol parameter statically

The value remains the same when you run the symbol.

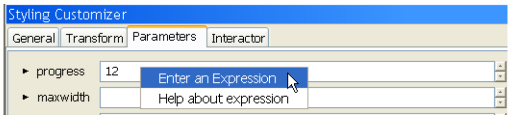
For a composite symbol to be fully effective, you need to change the parameter values for internal symbols dynamically. This is done be creating parameters for the region symbol and binding them to parameters contained in internal symbols.

To create parameters for the region symbol and bind them to parameters of the internal symbols:

1. In the region symbol, create four integer parameters:
   - ♦ shipments
   - ♦ sales
   - ♦ production
   - ♦ status

2. Select the shipments object.
3. Select the Parameters tab in the Styling customizer.
4. Right-click the progress parameter.

   A shortcut menu appears.
5. Select Enter an Expression in the shortcut menu.

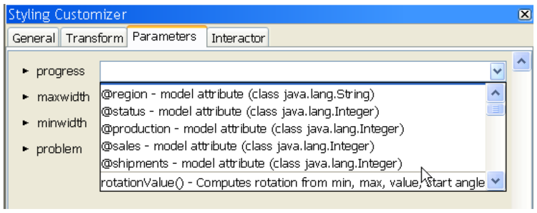


The expression shortcut menu

6. Delete the value in the progress field.

   The list of available parameters and expressions appears.
7. Select the @shipments parameter.

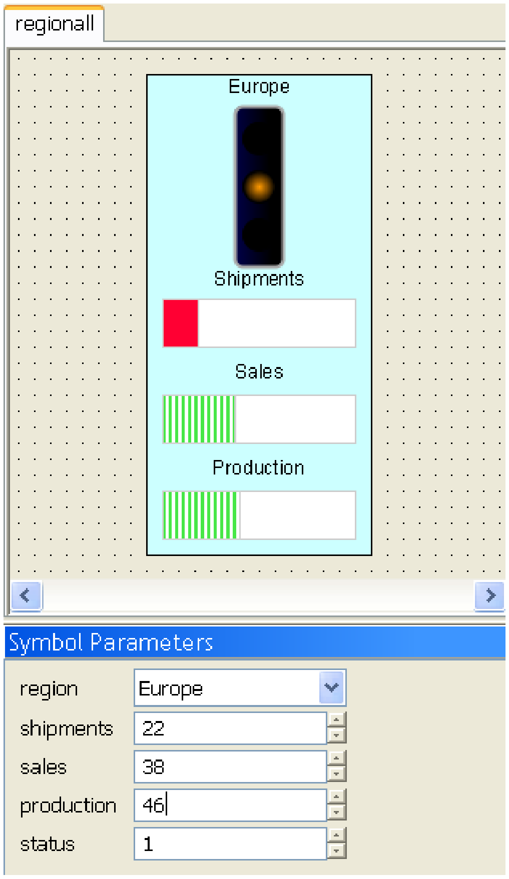
   The value of the progress parameter for the shipments symbol object is now bound to the shipments parameter of the regionall symbol.


Styling Customizer
General
Transform
Parameters
Interactor
progress
maxwidth
minwidth
problem
@region - model attribute (class java.lang.String)
@status - model attribute (class java.lang.Integer)
@production - model attribute (class java.lang.Integer)
@sales - model attribute (class java.lang.Integer)
@shipments - model attribute (class java.lang.Integer)
rotationValue() - Computes rotation from min, max, value, start angle

Select a parameter or expression

8. Bind the sales parameter to the progress parameter of the sales object.
9. Bind the production parameter to the progress parameter of the production object.
10. Bind the status parameter to the statusvalue parameter of the overallstatus object.

You can now alter the appearance of internal symbols by changing the value of top level symbol parameters.



region	Europe
shipments	22
sales	38
production	46
status	1

The complete region symbol

For more information about expressions, see *Using expressions*.

Creating the dashboard

You create the dashboard by importing the region symbol into either the Designer for JViews Diagrammer or the Dashboard Editor.

To create the dashboard:

- ♦ Import the region symbol into the Designer for JViews Diagrammer or the Dashboard Editor.

A symbol becomes dynamic when you connect the symbol parameters to real time data. For more information on how to do this, see *Next steps after the Symbol Editor*.

Using more features of the Symbol Editor

Describes the advanced features of the Symbol Editor and explains how to use them to create advanced symbols in general terms.

In this section

Exploiting advanced graphics features

Explains how to best use the capabilities of advanced graphics in symbols.

In this section

Supported graphic formats

In *Getting started with the Symbol Editor*, you learned how to manipulate graphic objects, parameters and conditions. In this section, you will see how to apply these techniques to symbols containing imported images.

The Symbol Editor can import the following file types:

♦ PNG

♦ JPG

♦ GIF

♦ SVG

♦ IVL

PNG, JPG, and GIF format files are static images. This means that their appearance cannot be edited directly using the Symbol Editor. However, static images have a smaller memory footprint. You use static images to create light symbols. For more information, see *Designing for performance*.

Both IVL and SVG files contain vector graphics. Using the Symbol Editor, you can edit and add conditions to each object contained in an imported vector graphic.

Importing a vector graphic

Vector graphics are added to a symbol in the same way as any other object.

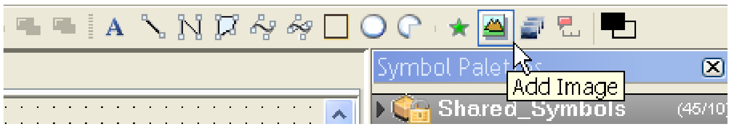
To import an SVG image:

1. In the Symbol Editor, create a new symbol.

   For more information, see *Making a new symbol*.

2. Click the Add Image button in the toolbar.

   The Choose Image File dialog box opens.

   The Add Image button

3. Navigate to the following directory:

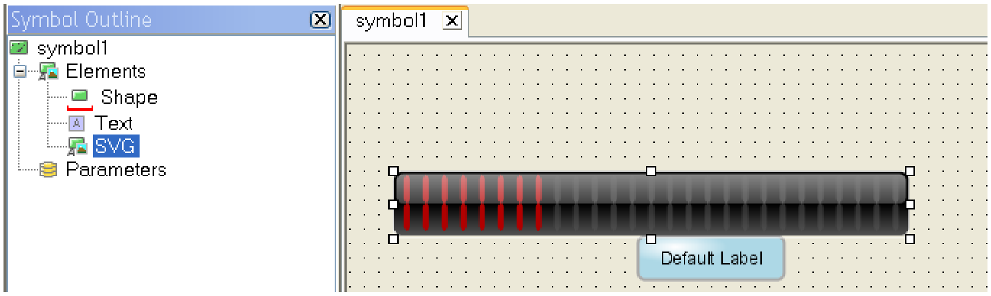
   `<installdir>/jviews-diagrammer86/bin/SymbolEditor/data/examples`.

4. Select Scalable Vector Graphics Files in the Files of Type list.
5. Select `progressbar.svg` and click Open.

   The Adding Vector Graphics dialog box opens.

6. Click OK.

   The progressbar image is added to your symbol.



A Symbol containing an SVG image

As shown in the figure above, the imported image is named SVG in the Symbol Outline pane. If more than one SVG image is added to the symbol, additional graphics are named SVG2, SVG3...SVGn automatically.

Note: When an SVG image is added to a symbol, the Image tab is added to the Styling Customizer. This tab is used to edit the width and height of the image manually. It also contains the URL to the imported image file.

An imported image is placed in the top layer of the symbol hierarchy. However, an image can be used as the base object for the symbol.

To use the imported image as the base object of the symbol:

1. In the symbol you have just created, change the name of the imported image from SVG to `progressbar`.

   For information on how to do this, see *Performing basic object customization*.

2. Delete the Shape and Text objects from your symbol.

   For more information, see *Deleting an object*. The progressbar SVG image is now the base object.

3. Open the bampalette created in *Using palettes and categories*.

   For more information, see *Closing and opening symbols and palettes*.

4. Save the new symbol as `svgprogressbar` in the progressbars category of the bampalette.

   See *Saving a symbol* for more information.

Converting SVG images to symbols

When you ungroup an SVG image, the Symbol Editor gives you access to all the objects contained in the graphic.

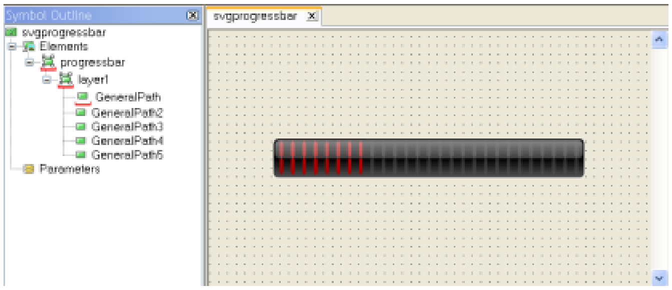
To ungroup the svgprogressbar symbol created in *Importing a vector graphic*:

1. Open the svgprogressbar symbol.

   For more information, see *Closing and opening symbols and palettes*.

2. Right-click the progressbar object.

3. Select Ungroup from the shortcut menu.

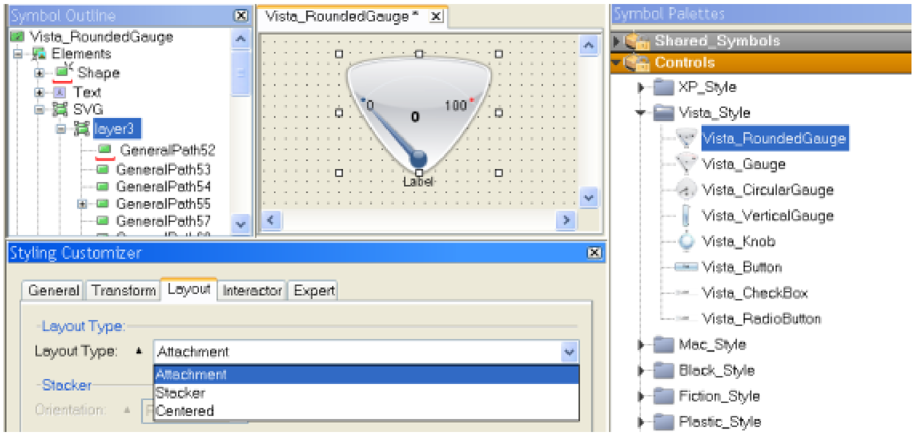


An ungrouped SVG image

As shown in the figure above, the Symbol Outline pane shows all the objects contained in progressbar. When an SVG image is ungrouped, the Symbol Editor gives each object a name automatically. An ungrouped image is no longer in SVG or IVL format, it is a standard IBM® ILOG® JViews symbol; the Image tab is no longer displayed in the Styling Customizer when the SVG root object is selected.

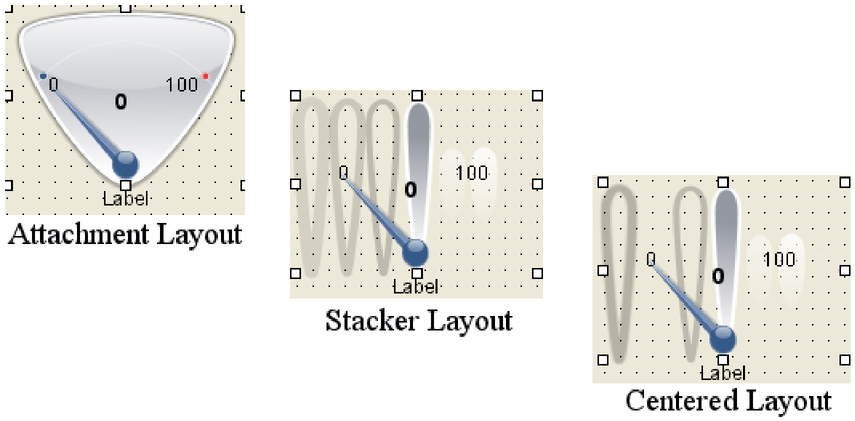
Control of group layout

As shown in figure *An ungrouped SVG image*, an ungrouped SVG file has multiple layers. Each layer is translated as a group of objects with a base object in the Symbol Editor. When a layer base object is selected, the Layout tab is displayed in the Styling Customizer.



The layout pane

You use the different layout options to control the distribution and arrangement of the objects inside the selected layer. The default layout type is an Attachment layout. The following figure shows how object distribution changes for different layout types.



Distribution differences according to layout types

Adding intelligence to an advanced graphic

The svgprogressbar symbol created in *Converting SVG images to symbols* is essentially the same as the basicprogressbar created in *Changing symbol layers*. That is, a collection of shapes where one object is expanded to display a change in real world events. The difference is that the svgprogressbar shown in figure *An ungrouped SVG image* is visually more pleasing than the basicprogressbar shown in figure *The region symbol*. As you will now see, it is much faster to add intelligence to a symbol made with advanced graphics.

To add intelligence to an advanced graphic symbol, you need to:

- ♦ Choose the object in the graphic to be transformed.
- ♦ Create the parameters necessary for transformation.
- ♦ Bind the parameters to the object.
- ♦ Add conditions if necessary.

The svgprogressbar symbol contains 5 different objects layered one on top of the other. The GeneralPath3 object, as shown in figure *An ungrouped SVG image*, is a red bar.

To transform this object:

1. In the Symbol Editor, select the GeneralPath3 object.
2. Create the following parameters:

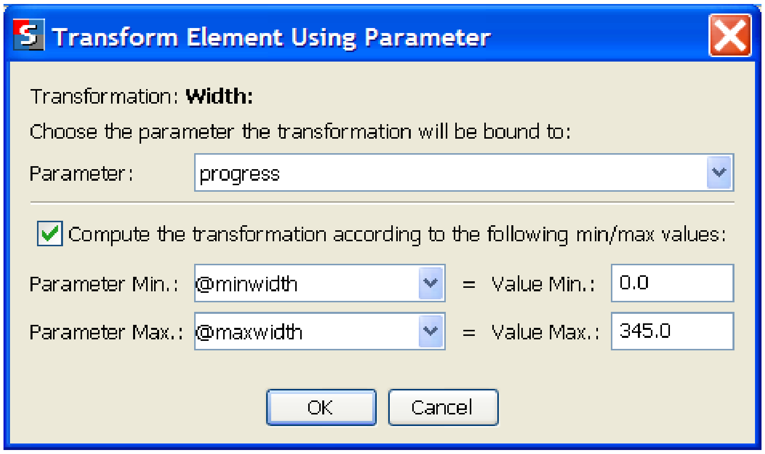
Parameters for the svgprogressbar symbol

Name	Type	Allowed Value	Allowed Value Name	Default Value
maxwidth	Integer	100	max	max
minwidth	Integer	0	min	min
problem	Integer	35	prob	prob
progress	float			0.1

   For more information, see *Adding flexibility using parameters*.

3. Bind the problem, minwidth and maxwidth parameters to the width attribute of the GeneralPath3 object.

   Pay attention to set the Value Max parameter to 345, that is, the maximum width of the bar in pixels.



Transform Element Using Parameter

Transformation: **Width:**

Choose the parameter the transformation will be bound to:

Parameter: progress

[✓] Compute the transformation according to the following min/max values:

Parameter Min.: @minwidth = Value Min.: 0.0

Parameter Max.: @maxwidth = Value Max.: 345.0

Transformation parameters

4. Set the same conditions on GeneralPath3 as shown in the *Transforming symbols using conditions* section.

Note: When you test GeneralPath3 and progress is set to 100, the bar width is 345 pixels.

Symbol graphics created by graphic artists using standard graphics packages have a finer look and feel. Importing advanced graphics into the Symbol Editor means symbols are created faster by the Symbol Editor. To add intelligence to a symbol based on SVG images, use exactly the same methods as you do for a basic symbol. See *Planning a project for the Symbol Editor* to see the recommended workflow for creating symbols.

Additional transformation techniques

Describes two more transformations that you can apply to symbols.

In this section

Transforming object position

The position of an object is set in function of the base object; the center of the base object is point 0,0. The base object is the central point of a symbol, it cannot change position. However, as shown in *Transforming an object*, you can apply a transformation on a base object to change its width or height. Objects in higher layers can change position either by setting the x and y objects manually, or by binding a parameter to these values.

To change the position of an object:

1. In the Symbol Editor, create a new default symbol.

   For more information, see *Making a new symbol*.

2. Select the Text object in the Symbol Outline pane.
3. Select the Transform tab in the Styling Customizer.

   The x and y values in the Transform tab are set to 0, that is, Text is in the exact center of Shape (Base).

4. Type `5` in the x field and press ENTER.

   You see Text moves to the right.

To change the position of an object using a parameter, use the method explained in *Transforming an object*.

Note: *The object in the lowest layer of a grouped object is designated as the group's base layer. As such, you can only customize its Width and Height objects, it cannot change position.*

Rotating an object

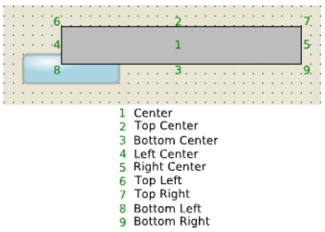
Rotation is when an object turns clockwise or anti-clockwise on an axis you define. Like changes to the size or position of an object, the degree of rotation is set in the following ways:

♦ Manually using the Transform tab in the Styling Customizer.

♦ Automatically by binding the object to a parameter.

Using rotation, you can create an alternative graphic representation for real time change.

The axis on which an object rotates is chosen by setting the Center field in the Styling Customizer. The following figure shows the different axis points that can be defined on an object.



Rotation axis points for a symbol object

Note: The axis is set on an object´s bounding box, that is, the smallest rectangle containing all parts of an object.

Setting the Angle field in the Styling Customizer to a positive value rotates the object clockwise, a negative value rotates anti-clockwise.

To create a rotating object in an imported SVG image:

1. In the Symbol Editor, create a new symbol.

   For more information, see *Making a new symbol*.

2. Navigate to the following directory:

   `<installdir>/jviews-diagrammer86/bin/symboleditor/data/examples`

3. Import `bamdial.svg`.

   The SVG image creates an object SVG. For more information, see *Importing a vector graphic*.

4. Delete the Text and Shape Objects.

   SVG is now the base object.

5. Create a float parameter called `progress`.

   For more information, see *Creating a new parameter*.

6. Ungroup SVG.

   For more information see *Converting SVG images to symbols*.

7. In the Symbol Outline pane, select the pointer object (General Path 6) and rename it `pointer`.

   See figure *The gauge symbol* to see the pointer object.

8. Select the Transform tab in the Styling Customizer.

9. Click the Bind button next to the Angle field.

10. Set the Parameter field to progress.

11. Enable `Compute the transformation according to the following min/max values`.

12. Set the following values:

    - Parameter Min: 0
    - Parameter Max: 100
    - Value Min: 0
    - Value Max 110.0. This is the maximum rotation for pointer.

13. Click OK.

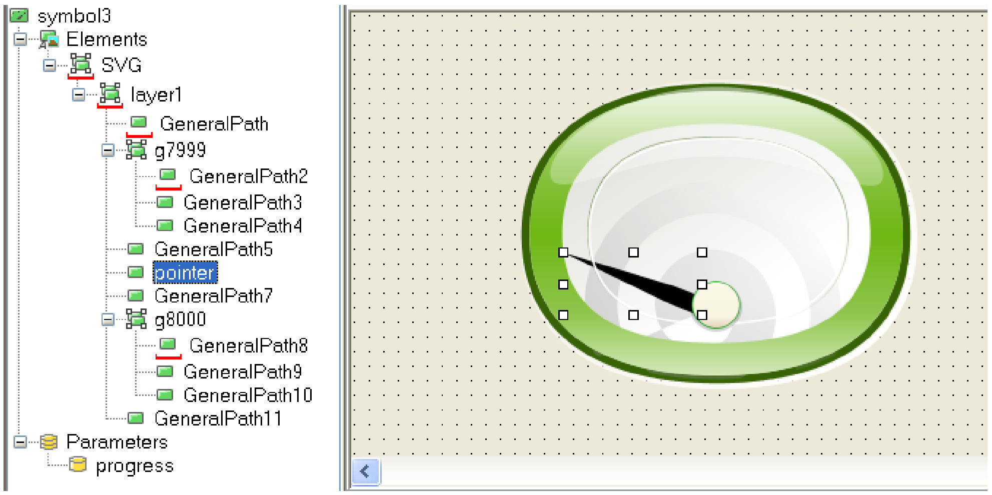
14. Set the Center field to BottomRight.

15. Save this symbol as `gauge` in a gauges category in the bampalette.

    For more information, see *Saving a symbol*.

16. Test your symbol.

    You will see that the pointer rotates in function of changes in the value of `progress`.



The gauge symbol

Customizing text style and contents

Text objects can be customized to have specific fonts and contents.

To customize a text object:

1. Open the gauge symbol created in *Rotating an object*.
2. Create an empty text area below the SVG symbol.

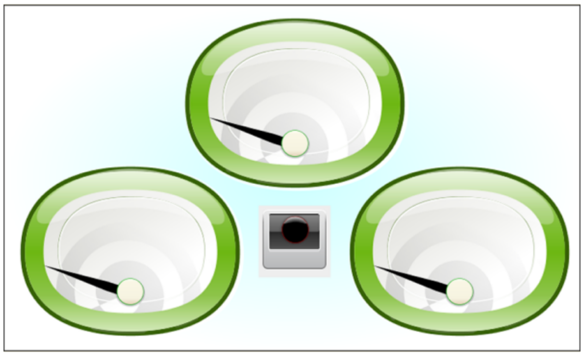
   For more information, see *Adding text to a symbol*.
3. Drag the Text object to the center of SVG (BASE).
4. Select the TextDisplay tab in the Styling Customizer.
5. Click the button next to the Font field.

   The Font Editor dialog box opens.
6. Select Arial, 14.0 and Bold.
7. Click Apply.
8. Select the Text tab in the Styling Customizer.
9. Type `@progress` in the Label field.

   The Text object now displays the value of the `progress` parameter.

Revisiting the BAM dashboard

You will use the gauge symbol created in *Rotating an object* and create a simple status indicator to present BAM data.



A graphics-intensive dashboard

Making a simple status indicator

To create a simple status indicator:

1. In the Symbol Editor, create a new symbol.

   For more information, see *Making a new symbol*.

2. Navigate to the following directory:

   `<installdir>/jviews-diagrammer86/bin/symboleditor/data/examples`

3. Import `light.svg`.

   The SVG image creates an object SVG. For more information, see *Importing a vector graphic*.

4. Delete the Text and Shape objects.

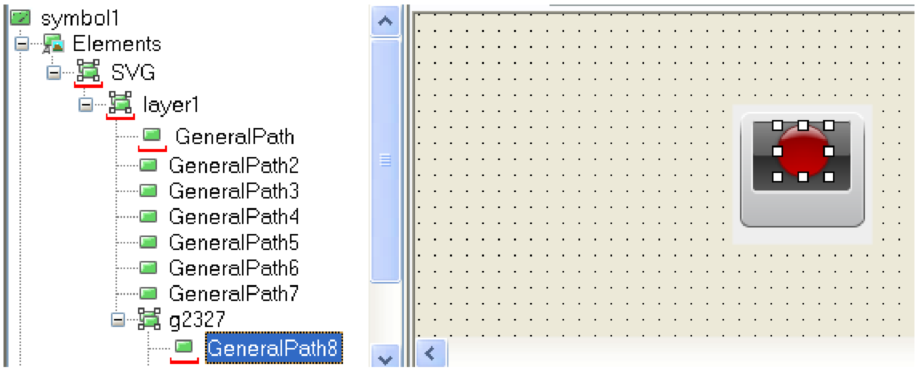
   SVG is now the base object.

5. Ungroup the SVG object.

6. Create an Integer parameter statusvalue, with three allowed values: good, warning and problem.

   For more details, see *Setting allowed values for a parameter*.

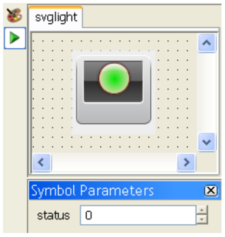
7. Select the GeneralPath8 object.



The GeneralPath8 object

8. Add the conditions good, warning and problem to GeneralPath8 so changes to the value of statusvalue are reflected by changes in color.

   For more information, see *Setting conditions on an object*.



The svglight symbol

9. Save this symbol as `svglight` in the stoplights category of the bampalette.

Creating the SVG region symbol

To create a new region symbol:

1. In the Symbol Editor, open the bampalette.
2. Double-click the regionbase symbol.

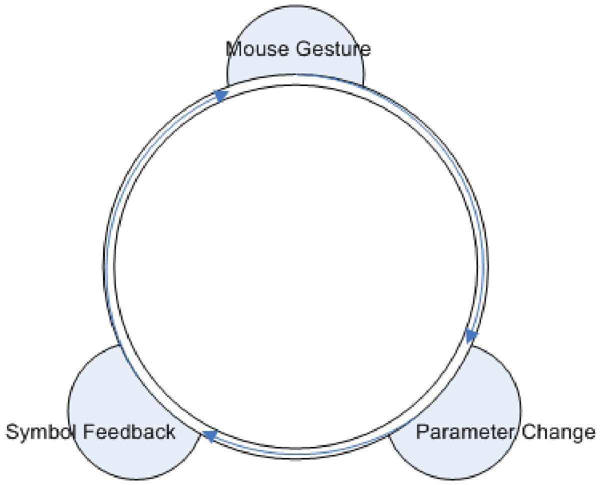
   The regionbase symbol opens in the Symbol Editor.
3. Resize regionbase to the following dimensions: Width:701, Height 421.
4. Add the svglight and gauge symbols so the dashboard resembles figure *A graphics-intensive dashboard*.
5. Create new parameters and link them to the composite symbol parameters using expressions.

   For more information, see *Connecting symbol parameters using expressions*.
6. Save this symbol as `svgregion` in the regions category of the bampalette.

Understanding interactors

You have seen that symbols have the ability to alter their visual aspect using parameters and conditions. This ability is useful for creating rich graphic objects such as dials or meters, for monitoring activities. Another useful aspect of GUI objects is to be able to respond to user actions using interactors. An interactor is used to create interactive objects by specifying how user actions can alter the value of a symbol parameter.

This mechanism works as a loop. A user interaction modifies the value of a parameter, this new value is processed by the symbol in order to give the appropriate visual feedback. Both actions in this loop are completely independent, that is, the interaction does not need to 'know' what the visual effect is.



The interactor workflow

A symbol interactor detects and processes input done by the user on a symbol. This is in the form of mouse events on the graphical representation of a symbol.

An interactor does one of the following:

♦ Uses the mouse event to change the value of a parameter in the symbol. Objects bound to the parameter are transformed according to this change in value.

♦ Passes the event to the object as a condition. This is used for Push and Discrete interactors.

The different kinds of interactions available are:

♦ Discrete: Increments or decrements the value of a parameter by a specific amount on a mouse click event. Parameter change is limited by maximum and minimum values.

♦ Push: Reacts to mouse BUTTON_UP, BUTTON_DOWN, and BUTTON_CLICKED events on a symbol. It sets the value of a string parameter in order to indicate the new state that the symbol could display.

- Horizontal: Changes the value of a parameter when an object is clicked and dragged horizontally. The change in parameter value depends on the distance the object is dragged. Parameter change is limited by maximum and minimum values.
- Vertical: Changes the value of a parameter when an object is clicked and dragged vertically. The change in parameter value depends on the distance the object is dragged. Parameter change is limited by maximum and minimum values.
- Rotation: Changes the value of a parameter when an object is clicked and rotated. The change in parameter value depends on the distance the object is dragged. Parameter change is limited by maximum and minimum values.

Discrete, Horizontal, Vertical, and Rotation interactors work in the same way. They respond to a mouse event by changing the value of a symbol parameter. This means that, although the interactor is applied to a specific object, the change occurs to any object bound to the parameter. Push interactors use conditions to transform the object clicked. When a condition has been created for a push interactor, it can be used to make simultaneous changes in multiple objects.

Working with interactors

Shows you how to create different transformations using interactors.

In this section

Creating a rotation interactor

A rotation interactor is used to revolve a symbol object when the mouse hovers over an object in a symbol. The rotation interactor changes the value of the parameter to which it is bound. This change in parameter value is used to make a rotation transformation on the object bound to the parameter. For example, if you rotate the mouse over a dial symbol, the dial knob rotates with the mouse.

To add a rotation interactor to a symbol:

1. In the Symbol Editor, create a new symbol.

   For more information, see *Making a new symbol*.

2. Import the `nobvista.svg` image from the `<installdir>/jviews-diagrammer86/bin/symboleditor/data/examples` directory.

   The SVG image creates an SVG object. For more information, see *Importing a vector graphic*.

3. Delete the Text and Shape objects.

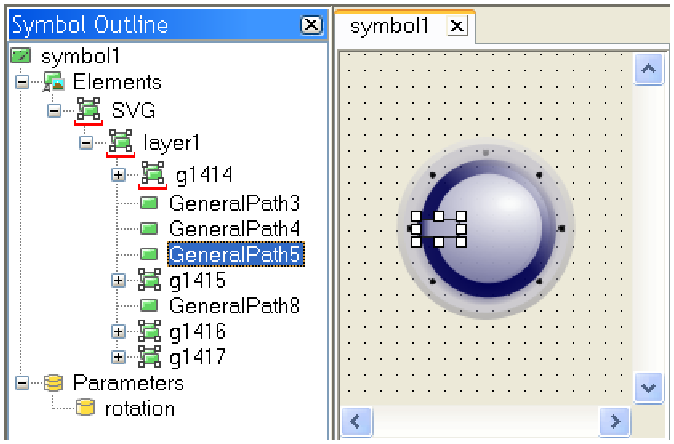
   SVG is now the base object.

4. Ungroup the SVG object.

5. Create a float parameter `rotation`.

   For more information, see *Creating a new parameter*.

6. Select the GeneralPath5 dial knob object.



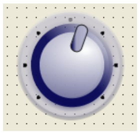
The GeneralPath5 object

7. In the Transform tab of the Styling Customizer, bind the Angle field to rotation.

   For more information, see *Transforming an object*.

8. Select Center Relative to Base in the Transform tab.

9. Select the SVG object.
10. Select the Interactor tab.
11. Set the following values:
    - Interactor: Rotation
    - Parameter: rotation
    - Maximum: 360
    - Start Angle: 180
    - Finish Angle: 360
    - Center: Center
    - Center Relative to Base Element.
12. Test this symbol by clicking the Preview button.



A rotation interactor

When you test this symbol, click on the symbol and drag the mouse in circles. The dial now follows the movement of the mouse.

Note: Objects with an associated interactor are indicated with a star SVG in the Symbol Outline pane.

Creating a push interactor

A push interactor is used to change the appearance of one or more objects when a mouse event occurs. By default, a Push interactor intercepts three mouse events: PRESSED, RELEASED, and CLICKED. When a mouse event occurs, the following values that represent the state of the object are passed to conditions

- ♦ BUTTON_UP - Sent when the mouse is released, or when it is dragged out of the symbol
- ♦ BUTTON_DOWN - Sent when the mouse is pressed and moves over the symbol
- ♦ BUTTON_CLICKED - Sent when the mouse is released over the symbol

These events fulfill conditions attached to one or more objects in the symbol. An example is a button that changes color when clicked.

To add a push interactor to a symbol:

1. In the Symbol Editor, create a new symbol.

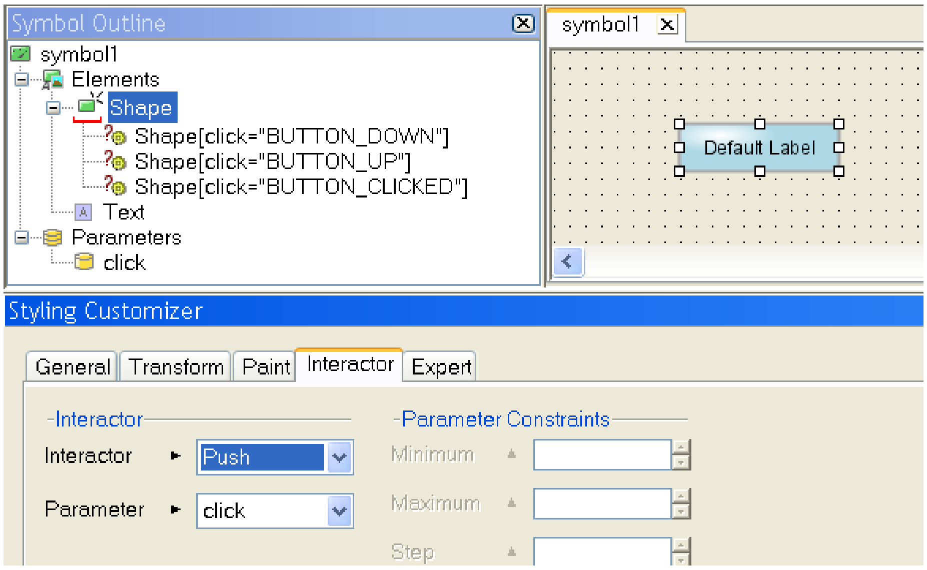
   For more information, see *Making a new symbol*.

2. Create a String parameter `click`.

   This parameter holds the string value of the mouse event.

3. Select the Shape object.
4. Select the Interactor tab in the Styling Customizer.
5. Select Push in the Interactor field.

   Three new conditions are created under the Shape object in the Symbol Outline pane.



Interactor: Push

Parameter: click

Automatic parameters for a push interactor

6. Select the BUTTON_DOWN condition in the Symbol Outline pane.
7. Select the Paint tab in the Styling Customizer.
8. Click the button at the right of the Fill Paint field.

   The Paint Editor opens.
9. In the Radial Gradient tab, change the gradient type from Pad to Reflect.
10. Click Apply.
11. Save this symbol as `basicbutton` in a new buttons category in the bampalette.

When you run this symbol in Preview mode, the Shape object changes color when it is clicked.

Note: You do not have to customize all three automatic push conditions. Delete the conditions you do not use.

Creating a horizontal interactor

A horizontal interactor is used to extend or contract a symbol object when the mouse moves horizontally over it. The horizontal interactor changes the value of the parameter to which it is bound. This change in parameter value is used to apply a transformation on the object bound to the parameter. For example, if you move the mouse over a progressbar symbol, the bar moves to the left or right with the mouse.

Note: Vertical interactors work in an identical way to horizontal interactors. The only difference is that the change in parameter value is caused by the mouse moving vertically and not horizontally.

To add a horizontal interactor to a symbol:

1. Open the basicprogressbar symbol you made in *Customizing the progress bar*.
2. Select the progressbar object.
3. Open the Interactor tab in the Styling Customizer.
4. Select Horizontal in the Interactor field.
5. Select progress in the Parameter field.
6. Set the Maximum field to 100.
7. Set the Step field to 1.
8. Test this symbol by clicking the Preview button.

You see that, as the mouse moves horizontally, the progressbar follows the mouse movement. As the value of maximum is set to 100, the progressbar never overruns the background image.

Note: Set the Horizontal Direction to Left in the Interactor tab of the Styling Customizer to make the progressbar move in the opposite direction to the mouse.

Applying interactors to multiple objects

Interactors change the value of a parameter or condition following mouse events on symbol objects. Use the change in parameter value to coordinate state changes for multiple symbol objects. For example, you can create a button whose background color and text change when the button is clicked, or a button with multiple rotating images.

Creating a push interactor for multiple objects

Push interactors in the Symbol Editor are used to fulfill a state in a condition. Once the condition has been fulfilled, the object to which the condition is bound changes representation. When a Push interactor is added to a symbol, the condition can be reused by other objects to coordinate symbol transformations.

1. In the Symbol Editor, open the symbol created in *Creating a push interactor*.
2. Right-click the Text object in the Symbol Outline pane.
3. Select New Condition in the shortcut menu.

   The New Condition dialog box opens.
4. Select [click=”BUTTON_DOWN”] in the `Other conditions defined for this symbol` pane.
5. Click OK.

   The condition is selected under the Text object in the Symbol Outline pane.
6. Select the Paint tab in the Styling Customizer.
7. Enable Stroke On.

When you run this symbol in Preview mode, the Text object is outlined when the mouse is pressed on the Shape object.

Creating a rotation interactor for multiple objects

A rotation interactor changes the value of a parameter. As the value of the parameter changes, the objects bound to it are transformed. By binding more than one object to a rotation parameter, you create synchronized movements in a symbol.

To rotate multiple objects:

1. In the Symbol Editor, create a new symbol.

   For more information, see *Making a new symbol*.
2. Add a Rotation interactor on the Text object bound to a float parameter.

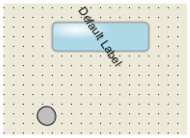
   Use the method explained in *Creating a rotation interactor*.
3. Create an ellipse object to the left of the Shape object.
4. With the new ellipse object selected, click the Transform tab in the Styling Customizer.
5. Bind the Angle field to the rotation parameter by setting the following values in the Transform Element Using Parameter dialog box:

   ♦ Parameter Min: 0

- ♦ Parameter Max: 100
- ♦ Value Min: 360
- ♦ Value Max: 0

6. Enable Center Relative to base.
7. Test this symbol by clicking the Preview button.
8. Save this symbol.

When you rotate the mouse over the Shape object, the ellipse rotates in the opposite direction to the text object.



Rotating multiple objects

Using expressions

Expressions are used to customize the format and value of a parameter. For example, you can use an expression to do the following to an object:

- ♦ Blink in two different colors
- ♦ Display the date in a specific format
- ♦ Join the value of two expressions
- ♦ Return the square root of a parameter

An expression can be set on any attribute contained in a symbol.

To use an expression on an object:

1. Open the symbol you made in *Creating a rotation interactor for multiple objects*.
2. Select the Text object.
3. Click the Text tab in the Styling Customizer.
4. Right-click the Label field.

   A shortcut menu appears.
5. Click Enter an Expression.
6. Delete the contents of the Label field.

   It is changed into a drop down list.
7. Click the down arrow on the right of the Label field.

   The list of expressions appears.
8. Select the `@rotationValue` expression.
9. Fill in the expression as follows: `rotationValue(0,360,@rotate,0,360)`.
10. Test this symbol by clicking the Preview button.

As you rotate the mouse over the symbol, the Text label is updated to show the angle of rotation.

Expressions can combine other expressions, model attributes and string literals. For example, the following expression prints the integer value of a parameter in a string.

```
concat("the rotation value is ", rotationValue(0,360,@rotate,0,360), ".")
```

Note: When creating expressions, always select values from the list, do not type the parameter name or expression by hand.

Possible expressions are listed in the following table.

Symbol Editor expressions

Name	Description	Example
`parameter`	Any parameter in the symbol	`@rotate`
`rotationValue`	Calculates the rotation angle	`rotationValue(@rotate)`
`int`	Casts the parameter to an `Integer`	`int(@rotate&100)`
`decimalFormat`	Reformats a parameter value. See *java.text.DecimalFormat* for more information.	`decimalFormat("#.#", pi) -> 3.1`
`childrenCount`	Returns the number of child objects below this object	`childrenCount()=0`
`simpleDateFormat`	Formats a `date`. See *java.text.SimpleDateFormat* for more information.	`simpleDateFormat("MM/dd/yyyy",@date)` This will give: 10/01/2001
`long`	Casts to a `long` value	`long(@val1+@val2)`
`double`	Casts to a `double` value	`double(@value*3)`
`messageFormat`	Formats different strings to a specific message format. See *java.text.MessageFormat* for more information.	`messageFormat('{0}:{1,number,#.##}','result',44.2874).`

Name	Description	Example
		This will give: 44.28"
`blinkingColor`	Specifies 2 different colors to alternate for a graphic object	`blinkingColor (green,yellow)`
`concat`	Concatenation for String parameters	`concat("result is:",@result)`
`float`	Casts parameter to a float value	`float(cos(@value))`
`abs`	Absolute value of the parameter	See *java.lang.Math*
`acos`	Returns the arc cosine of an angle, in the range of 0.0 through pi	See *java.lang.Math*
`asin`	Returns the arc sine of an angle, in the range of -pi/2 through pi/2	See *java.lang.Math*
`atan`	Returns the arc tangent of an angle, in the range of -pi/2 through pi/2	See *java.lang.Math*
`ceil`	Returns the smallest (closest to negative infinity) double value that is not less than the argument and is equal to a mathematical integer	See *java.lang.Math*
`tan`	Returns the trigonometric tangent of an angle	See *java.lang.Math*
`cos`	Returns the trigonometric cosine of an angle. Special cases:	See *java.lang.Math*
`exp`	Returns Euler's number e raised to the power of a double value	See *java.lang.Math*
`floor`	Returns the largest (closest to positive infinity) double value that is not greater than the argument and is equal to a mathematical integer	See *java.lang.Math*
`log`	Returns the natural logarithm (base e) of a double value	See *java.lang.Math*
`rint`	Returns the double value that is closest in value to the argument and is equal to a mathematical integer	See *java.lang.Math*
`sin`	Returns the trigonometric sine of an angle	See *java.lang.Math*
`sqrt`	Returns the correctly rounded positive square root of a double value	See *java.lang.Math*

Many of these expressions are CSS functions, see Using custom functions. If you want to register your own predefined CSS functions in the Symbol Editor, extract the file `ilog/views/util/css/cssfunctions.properties` from `jviews-framework-all.jar`. This file contains the settings for predefined CSS functions. Follow the instructions in this file to add your own CSS functions.

Designing for performance

Explains how to improve the performance of data-rich applications by using light symbols.

In this section

Performance guidelines

The applications you have been shown so far in this document are designed for non real-time systems, that is, systems that display a small amount of data which is refreshed infrequently. An IBM® ILOG® JViews Dashboard Editor or IBM® ILOG® JViews Designer application created for this purpose will contain only very few symbols.

Real-time systems deal with massive amounts of data that is updated continually. To handle the data quantity and update frequency, you must create light symbols. Light symbols use the minimum amount of objects, parameters, conditions and interactors necessary to display an understandable message.

Use the following guidelines to create symbols with the smallest memory footprint:

- ♦ Use as few objects as possible.
- ♦ Use a static background.
- ♦ Use light (PNG) images.
- ♦ Use the visibility option.
- ♦ Avoid transparency.
- ♦ Avoid complicated gradients.
- ♦ Avoid thick lines.
- ♦ Turn antialiasing off.
- ♦ Turn fraction metrics off.

Rendering operations take up a lot of CPU time. The fastest and lightest symbols are made by changing the visibility setting for a collection of PNG images.

Creating a light alarm symbol using the image list creation wizard

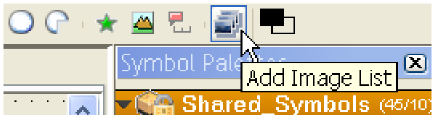
To create an alarm symbol that indicates status:

1. In the Symbol Editor, create a new symbol.

   For more information, see *Making a new symbol*.

2. Click the Add Image List button.

   The Image List Creation wizard opens.

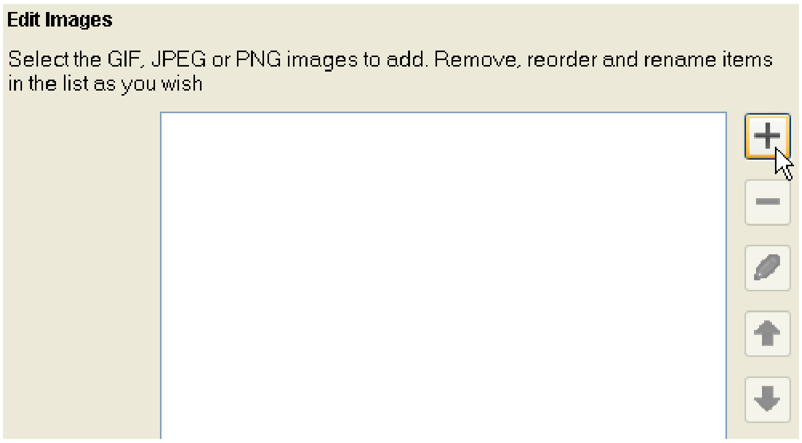


The Add Image List button

3. Click Next to pass the welcome page.

> **Note**: Clear the Do not show this page next time checkbox so the welcome page is not displayed the next time you use the Image List Creation Wizard.

4. In the main dialog, click the plus button to add images.



Edit Images

Select the GIF, JPEG or PNG images to add. Remove, reorder and rename items in the list as you wish

Adding images

5. In the Choose Image File browser, navigate to the `<installdir>/jviews-diagrammer86/bin/symboleditor/data/examples` directory.

6. Select the following image files:

   ♦ `led_circle_green.png`

- `led_circle_orange.png`
- `led_circle_red.png`

7. Click Open.

   The images are added to the list.

8. Select On order.
9. Create a new parameter by typing statusvalue in the Parameter field.

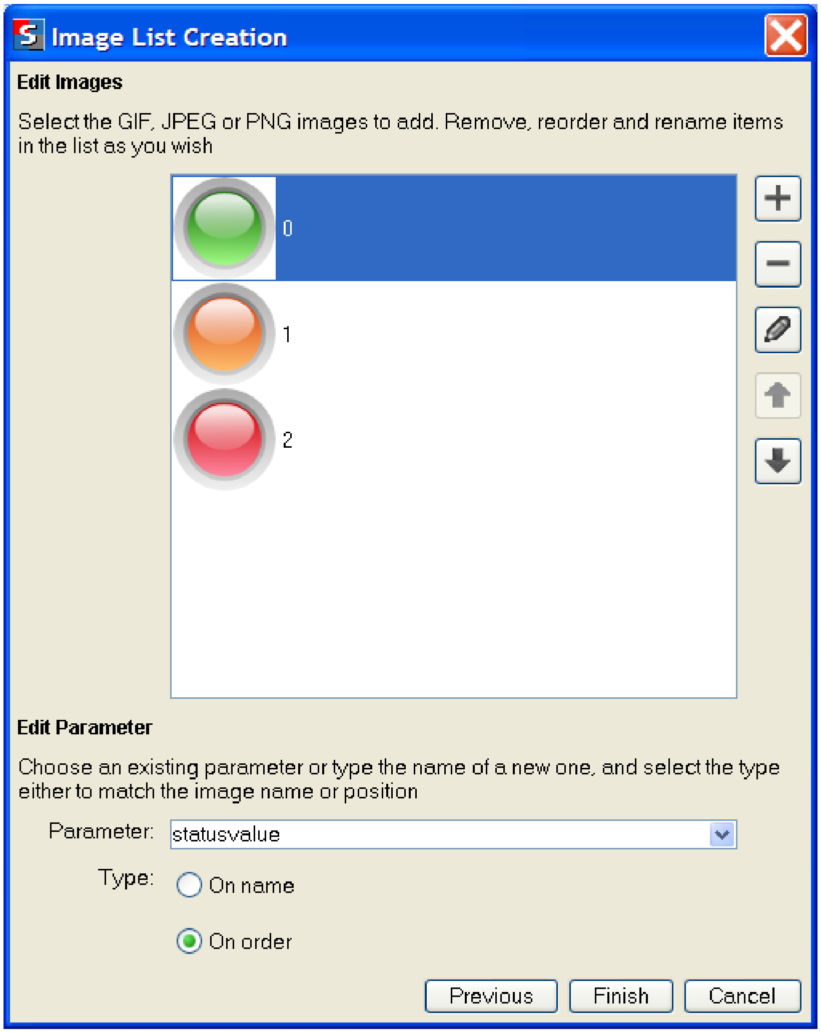


Image List Creation

Edit Images

Select the GIF, JPEG or PNG images to add. Remove, reorder and rename items in the list as you wish

Edit Parameter

Choose an existing parameter or type the name of a new one, and select the type either to match the image name or position

Parameter: statusvalue

Type: On name [✓ On order]

Adding a new parameter for the image list

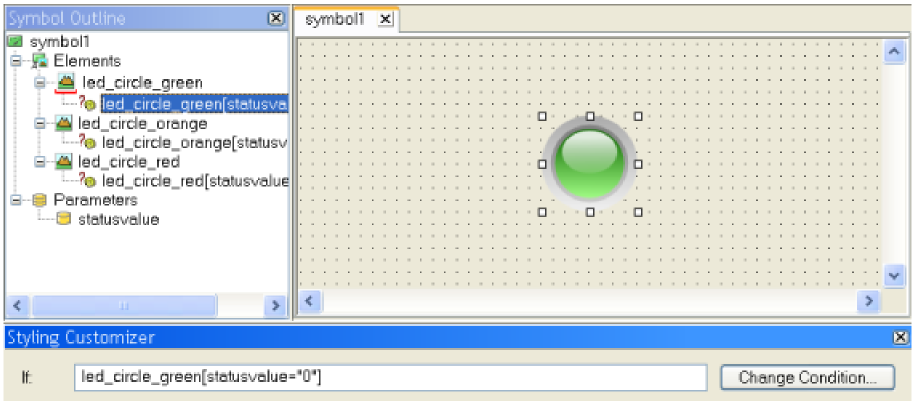
10. Click Finish.

The images, parameter and conditions are added to the symbol.

Note: The On name option is used to set the visible image using the image name. By default, the image name is the name of the file imported. You can rename each image using the rename button in the Image List Creation wizard.

11. Delete the Text and Shape Objects.

The led_circle_green object is now the base object.



If: led_circle_green[statusvalue="0"]

led_circle_green as base object

12. Create the following three allowed values on the `statusvalue` parameter.

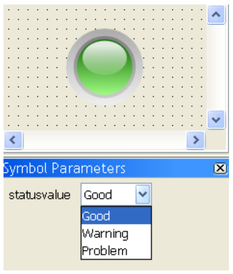
See *Setting allowed values for a parameter* for more information.

Allowed values for the statusvalue parameter

Allowed Value Name	Value
good	0
warning	1
problem	2

13. Set the default value of `statusvalue` to good.

14. Test this symbol by clicking the Preview button.



A light alarm symbol

As the value of `statusvalue` changes, a different PNG image becomes visible. For example, when statusvalue is equal to problem, the red image blinks to give an alarm.

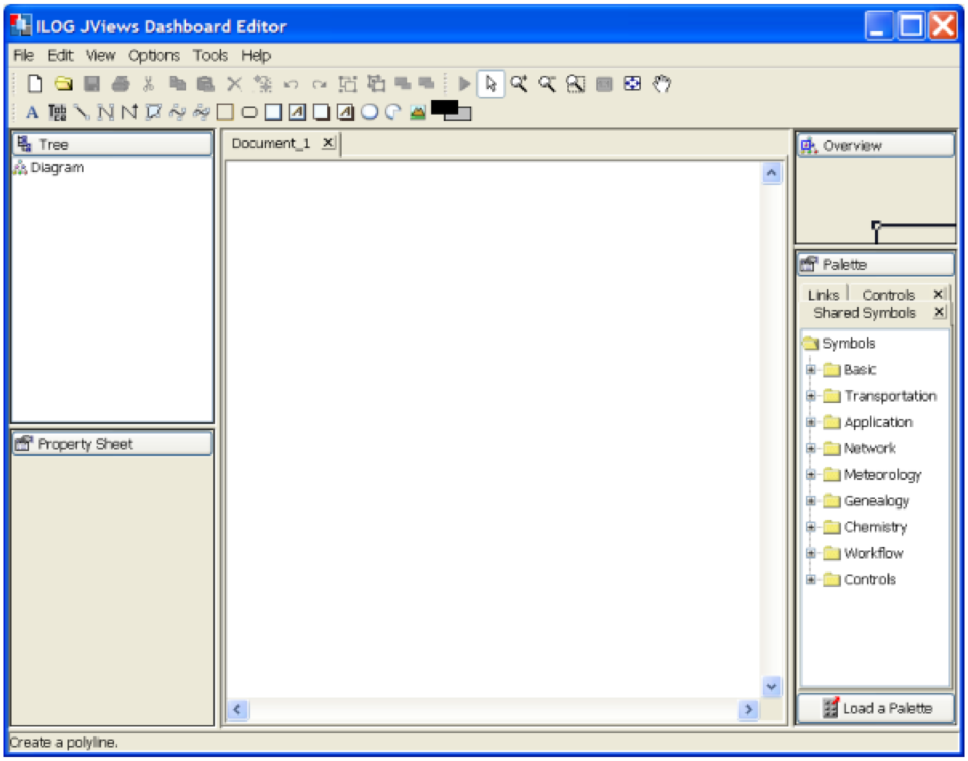
Next steps after the Symbol Editor

Indicates what you can do after configuring your symbol in the Symbol Editor.

In this section

Integrating a symbol into JViews Dashboard Editor

A Dashboard Diagram is a graphical component made up of IBM® ILOG® JViews symbols arranged on an editable background. It is used to graphically represent business or system critical information. The IBM® ILOG® JViews Dashboard Editor is a point-and-click editor that allows you to create and edit dashboards.



The JViews Diagrammer Dashboard Editor

Dashboard functionality is implemented in the `ilog.views.dashboard` package. It is used to:

♦ Load and animate a dashboard programmatically.

♦ Create your own graphical Dashboard Editor GUI application.

For more information on using the JViews Diagrammer Dashboard Editor and programming with the `ilog.views.dashboard` package, see Using the Dashboard Editor.

Integrating a symbol into Designer for JViews Diagrammer

Symbols are fully integrated into Designer. Using the Symbol menu in the Designer, you can do the following:

♦ Assign a symbol to a node in your Designer diagram.

♦ Edit symbols using the Symbol Editor.

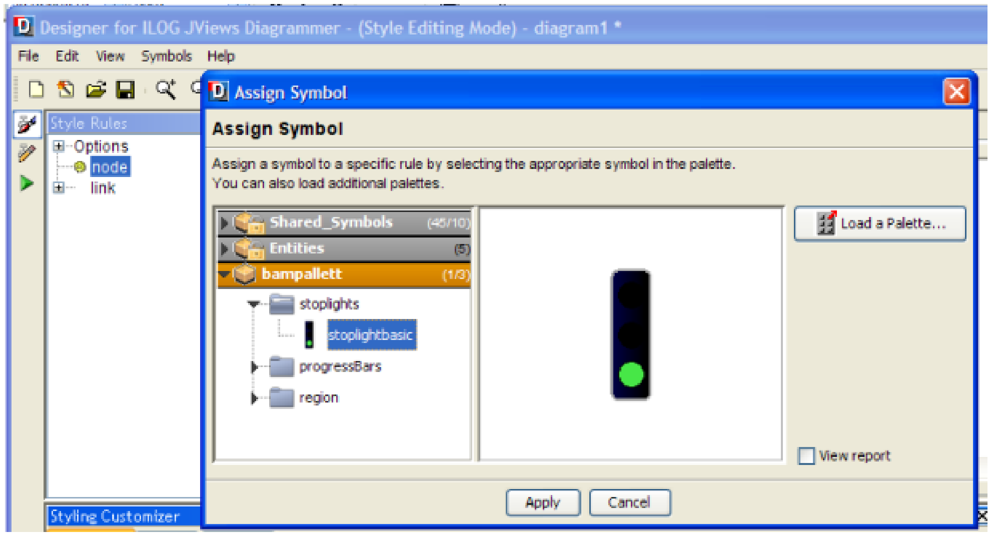
♦ Refresh the symbols in the Designer diagram from their palette.

♦ List the symbol palettes used in the Designer diagram.

The following figure shows the symbol you made in *Creating the stoplight symbol* integrated as a node image in a diagram.

To open the Assign Symbol dialog:

1. Choose **Symbols>Assign symbol**.
2. Select the stoplightbasic symbol.
3. Click Apply.



Assign a symbol to a specific rule by selecting the appropriate symbol in the palette. You can also load additional palettes.

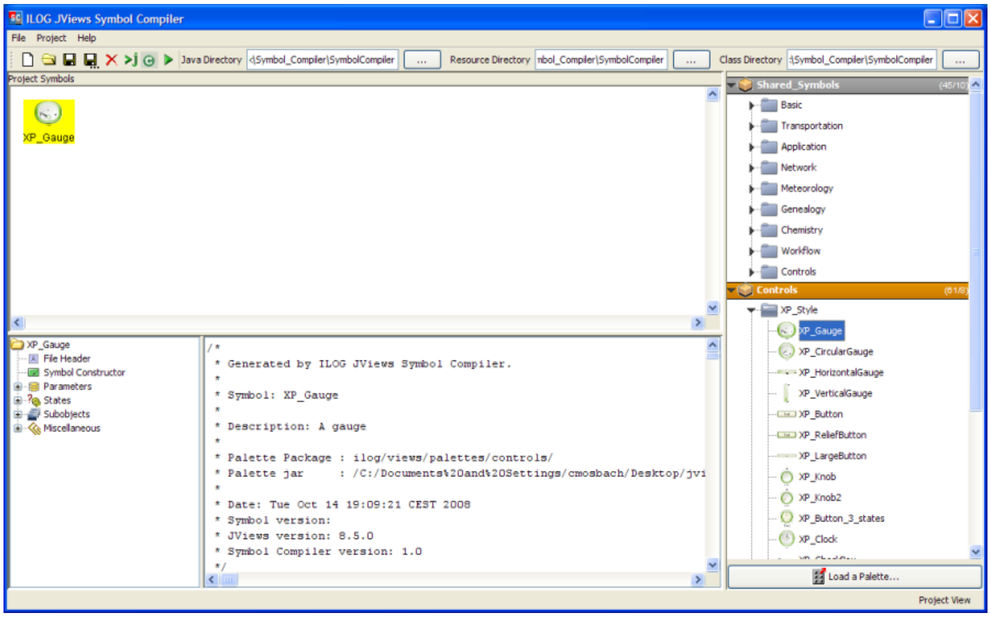
Importing a symbol into a Designer diagram

For more information on integrating symbols into the Designer, see Using the Designer.

Generating symbols as Java source code

Symbols created with the Symbol Editor can be used directly in JViews Diagrammer applications for dashboards and diagrams. Internally, the symbol definition is based on CSS directives that can be dynamically interpreted at run time.

In some cases, it is interesting to select a slightly different approach which is to generate Java™ source code from the symbols you have created with the Symbol Editor. This Java code uses simple graphics primitives managed by the JViews Framework APIs. Each symbol becomes a JavaBean™ that you can instantiate and manage from your JViews application. Every single parameter you defined in the Symbol Editor becomes a bean property, accessible through traditional set/get methods. The tool that allows you to produce this Java code is the Symbol Compiler.



```
/*
 * Generated by ILOG JViews Symbol Compiler.
 *
 * Symbol: XP_Gauge
 *
 * Description: A gauge
 *
 * Palette Package : ilog/views/palettes/controls/
 * Palette jar     : /C:/Documents%20and%20Settings/cmosbach/Desktop/jvi
 *
 * Date: Tue Oct 14 19:09:21 CEST 2008
 * Symbol version:
 * JViews version: 8.5.0
 * Symbol Compiler version: 1.0
 */
```

The IBM® ILOG® JViews Symbol Compiler

What are the benefits of using compiled symbols:

♦ Performance is usually better with compiled symbols. The time to create the symbol is slightly improved and the initial display time of a diagram or a dashboard is reduced. The animation time can be reduced too, and sometimes quite significantly. As a consequence, you may experience faster animation or save CPU for other important tasks. Lastly, memory footprint can be reduced which makes your application more responsive. Since performance improvements can vary depending on the kind of symbols you design, it is recommended to perform some dedicated tests to decide whether this approach is suitable for your particular case.

♦ You can generate and compile your symbols at the end of your development. For example, you can tell SDM to use compiled symbols, if they exist, rather than CSS-based symbols.

In the same way, you can dynamically load and animate dashboards using either interpreted or compiled symbols.

- When symbols are available as source code and usable as regular beans, you can specialize them by adding specific Java code. You can also create derived classes implementing domain-specific logic.

These cases are further discussed in Using the Symbol Compiler.

Index