

# 4D SVG

---

Component for  
Windows® and Mac OS®



---

## **4D SVG Component**

**Version 11.4 for Windows® and Mac OS®**

Copyright © 4D SAS/4D, Inc. 1985-2009

All rights reserved.

---

The Software described in this manual is governed by the grant of license in the 4D Product Line License Agreement provided with the Software in this package. The Software, this manual, and all documentation included with the Software are copyrighted and may not be reproduced in whole or in part except for in accordance with the 4D Product Line License Agreement.

4D, 4D Draw, 4D Write, 4D View, 4D Server as well as the 4D logo are registered trademarks of 4D, Inc.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Apple, Macintosh, Mac OS and QuickTime are trademarks or registered trademarks of Apple Computer, Inc.

Mac2Win Software Copyright © 1990-2009, is a product of Altura Software, Inc.

Spellchecker © Copyright SYNAPSE Développement, Toulouse, France, 1994-2009.

ACROBAT © Copyright 1987-2009, Secret Commercial Adobe Systems Inc. All rights reserved. ACROBAT is a registered trademark of Adobe Systems Inc.

All other referenced trade names are trademarks or registered trademarks of their respective holders.

# Contents

## 1. Introduction..... 7

4D SVG Component.....	9
Development tools.....	11
Syntax details.....	16

## 2. Attributes.....17

SVG_GET_ATTRIBUTES.....	19
SVG_Get_ID.....	20
SVG_SET_ATTRIBUTES.....	21
SVG_SET_ATTRIBUTES_BY_ARRAYS.....	22
SVG_SET_DIMENSIONS.....	23
SVG_SET_FILL_BRUSH.....	24
SVG_SET_FILTER.....	25
SVG_SET_ID.....	26
SVG_SET_MARKER.....	27
SVG_SET_OPACITY.....	30
SVG_SET_ROUNDING_RECT.....	31
SVG_SET_STROKE_BRUSH.....	32
SVG_SET_STROKE_LINECAP.....	33
SVG_SET_STROKE_LINEJOIN.....	34
SVG_SET_STROKE_WIDTH.....	35
SVG_SET_TRANSFORM_FLIP.....	36
SVG_SET_TRANSFORM_MATRIX.....	38
SVG_SET_TRANSFORM_ROTATE.....	40
SVG_SET_TRANSFORM_SCALE.....	41
SVG_SET_TRANSFORM_SKEW.....	42
SVG_SET_TRANSFORM_TRANSLATE.....	43
SVG_SET_VIEWBOX.....	44
SVG_SET_VIEWPORT_FILL.....	45
SVG_SET_VISIBILITY.....	46
SVG_SET_XY.....	47

### 3. Colors and Gradients..... 49

SVG Colors.....	51
SVG_Color_grey.....	52
SVG_Color_RGB_from_long.....	53
SVG_GET_DEFAULT_BRUSHES.....	54
SVG_SET_DEFAULT_BRUSHES.....	55

### 4. Documents..... 57

SVG_CLEAR.....	59
SVG_Copy.....	60
SVG_Export_to_picture.....	61
SVG_Export_to_XML.....	62
SVG_New.....	63
SVG_Open_file.....	65
SVG_Open_picture.....	66
SVG_SAVE_AS_PICTURE.....	67
SVG_SAVE_AS_TEXT.....	68
SVG_Validate_file.....	69

### 5. Drawing..... 71

SVG_ADD_POINT.....	73
SVG_New_arc.....	74
SVG_New_circle.....	76
SVG_New_ellipse.....	78
SVG_New_ellipse_bounded.....	80
SVG_New_embedded_image.....	82
SVG_New_image.....	84
SVG_New_line.....	86
SVG_New_path.....	87
SVG_New_polygon.....	91
SVG_New_polygon_by_arrays.....	92
SVG_New_polyline.....	94

SVG_New_polyline_by_arrays.....	96
SVG_New_rect.....	99
SVG_New_regular_polygon.....	101
SVG_PATH_ARC.....	103
SVG_PATH_CLOSE.....	105
SVG_PATH_CURVE.....	106
SVG_PATH_LINE_TO.....	107
SVG_PATH_MOVE_TO.....	108
SVG_PATH_QCURVE.....	109
SVG_Use.....	110

## 6. Filters..... 113

SVG Filters.....	115
SVG_Filter_Blend.....	116
SVG_Filter_Blur.....	117
SVG_Filter_Offset.....	118

## 7. Structure and Definitions..... 119

SVG_Define_filter.....	121
SVG_Define_linear_gradient.....	123
SVG_Define_marker.....	126
SVG_Define_radial_gradient.....	128
SVG_Define_shadow.....	130
SVG_Define_solidColor.....	132
SVG_Define_symbol.....	133
SVG_New_group.....	134
SVG_Set_description.....	136
SVG_Set_title.....	137

## 8. Text..... 139

SVG_New_text.....	141
SVG_New_textArea.....	144

SVG_New_tspan.....	147
SVG_New_vertical_text.....	149
SVG_SET_FONT_COLOR.....	151
SVG_SET_FONT_FAMILY.....	152
SVG_SET_FONT_SIZE.....	153
SVG_SET_FONT_STYLE.....	154
SVG_SET_TEXT_ANCHOR.....	155

## **9. Utilities..... 157**

SVGTool_SHOW_IN_VIEWER.....	159
SVG_Count_elements.....	160
SVG_ELEMENTS_TO_ARRAYS.....	161
SVG_Estimate_weight.....	162
SVG_Find_ID.....	163
SVG_Get_options.....	164
SVG_Get_version.....	166
SVG_Is_reference_valid.....	167
SVG_Read_element_type.....	168
SVG_Read_last_error.....	169
SVG_References_array.....	171
SVG_Set_error_handler.....	172
SVG_SET_OPTIONS.....	174

## **10. Appendixes..... 177**

Appendix A, Table of colors.....	179
Appendix B, External links.....	181

## **Command Index..... 183**

# 1

---

# Introduction





SVG (Scalable Vector Graphics) is a two-dimensional vector graphics file based on XML. 4D includes an integrated rendering engine that can be used to display SVG files.

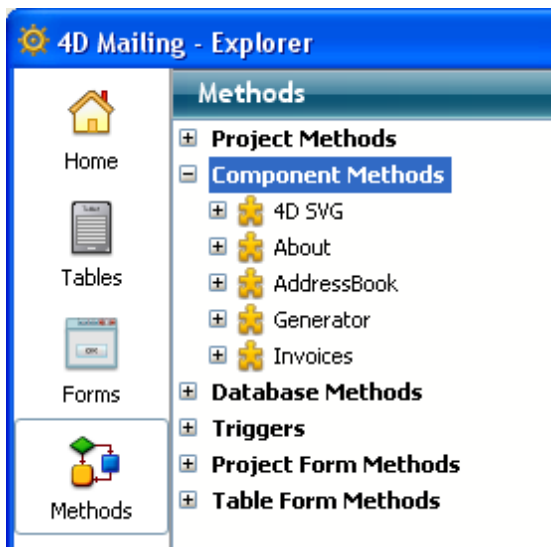
The XML language used for manipulating SVG pictures is particularly rich and extensive. In order to make getting started easier, 4D provides the SVG component, which includes numerous commands that can be used to create and manipulate common graphic objects. This library is not intended to be exhaustive but rather to meet the most frequent needs of 4D developers. Note that additional needs can be processed with the 4D XML commands.

### Installation and implementation

The 4D SVG Component must be installed in version 4D v11 SQL release 3 (version 11.3) or higher.

Like all 4D components, the 4D SVG Component is installed by copying the component folder (4D SVG.4dbase) into the Components folder of the database. The Components folder of the database must be located at the same level as the structure file. Since components are loaded on startup, the database must not be launched before completely copying all the elements.

If the component is correctly installed, the **4D SVG** element appears on the Methods page of the database, in the "Component Methods" section:



You can expand this element in order to view all the component commands. These commands can be used in the 4D Method editor just like 4D language or standard plug-in commands.

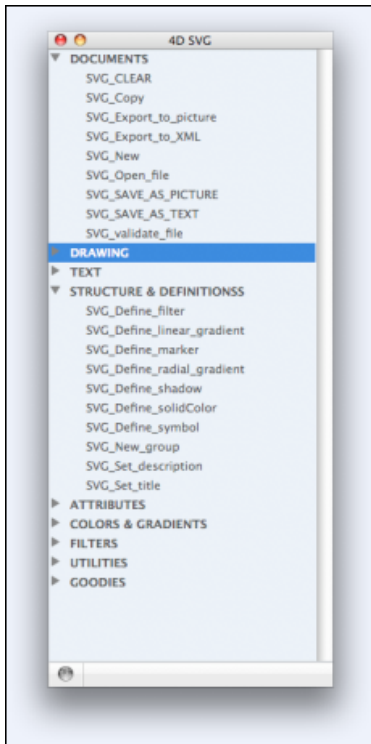
Note that the 4D SVG Component lets you benefit from additional windows for the selection of commands and for SVG code rendering. For more information, please refer to the Development Tools section.

The 4D SVG component provides a set of tools intended to facilitate the entry of code and to preview the SVG graphics:

- the syntax palette
- the color palette
- the SVG viewer.

### Syntax Palette

The syntax palette lists the 4D SVG component commands grouped by themes:



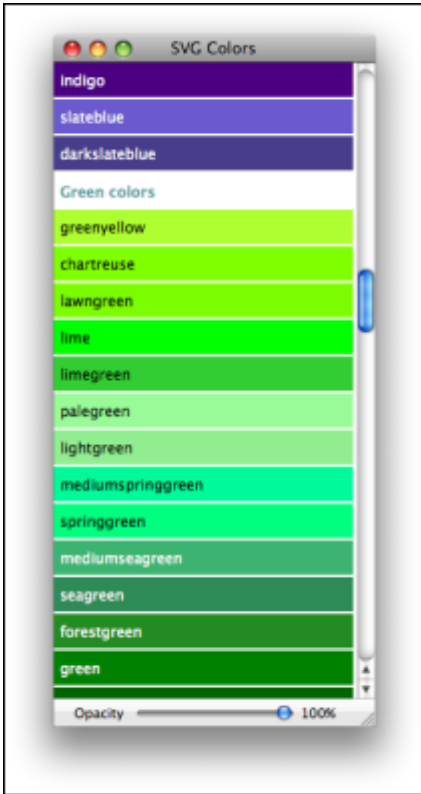
This palette can be used to insert the component commands into the Method editor by simple drag and drop. The command is then pasted in the method with its parameters. The optional parameters are prefixed by an underline.

To display the syntax palette, you can either:

- execute the SVGTool\_Display\_syntax method, or
- click on the **SVG** button and choose the **SVG Component syntax** command in the 4D Pop component palette if you are using it (see below).

## Color Palette

The color palette displays the name and a sample of each color specified in the SVG standard, as well as a slider that can be used to vary the rate of opacity:



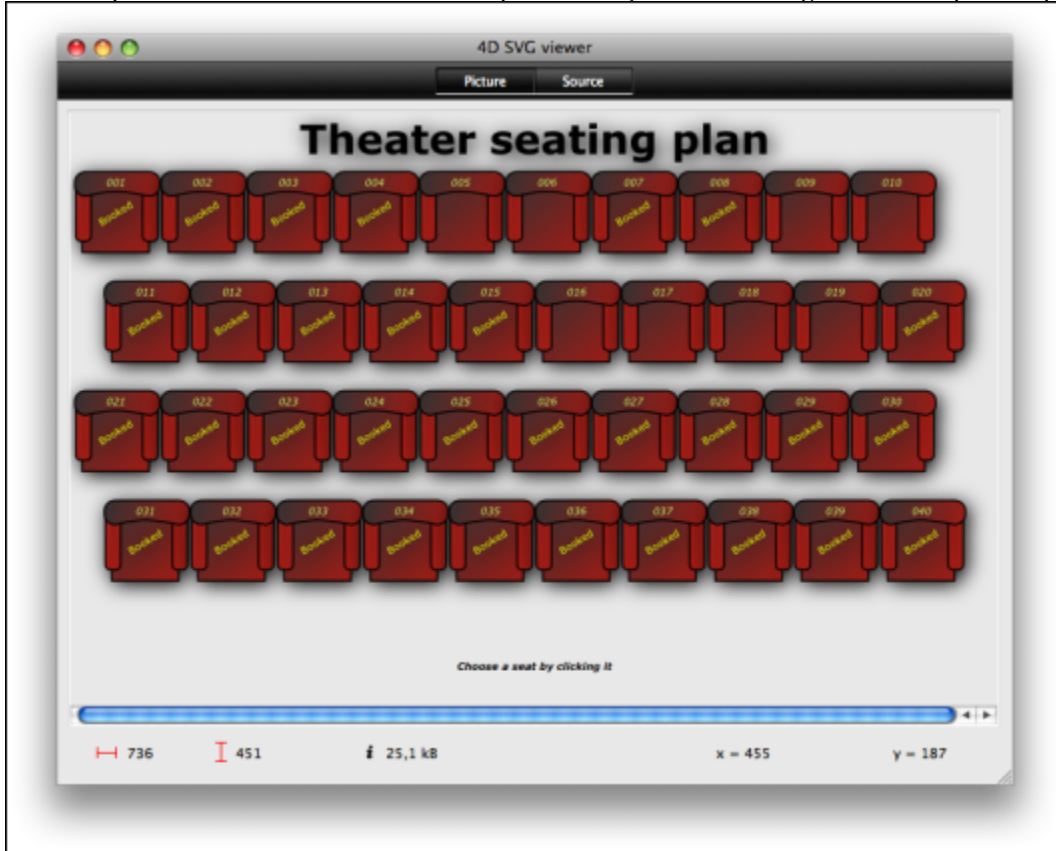
You can use this palette to insert an SVG color reference by drag and drop into the 4D Method editor. The color is inserted as a string that includes the rate of opacity, if any (for example "lavender:30" for the color lavender with an opacity of 30%). For more information about color references, please refer to the SVG Colors section.

You can also drag and drop a color into the 4D Form editor. This creates a square of color in the form of a static SVG picture.

To display the color palette, just execute the `SVGTool_Display_colors` method.

## SVG Viewer

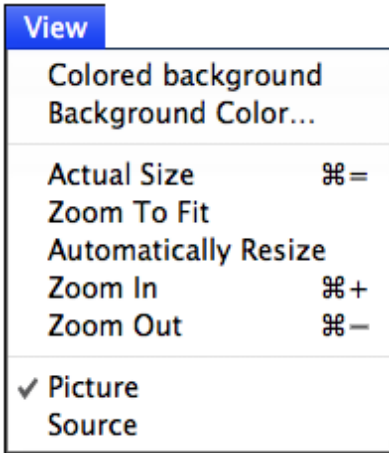
4D SVG provides an SVG viewer that is particularly useful during the development phase:



The viewer window has two pages which can be accessed via the **Picture** and **Source** buttons or the **View** menu:

- **Picture:** This page provides a display area into which you can drag and drop or open an SVG picture file (via the **File** menu). You can also display a valid SVG reference using the `SVGTool_SHOW_IN_VIEWER` command.
- **Source:** This page lets you view the XML code associated with the picture. You can select and copy the code, but you cannot modify it.

When the window is in the foreground, you can modify several display options and save the picture file on disk using the **View** menu:



**Note:** The "Picture" page has a standard contextual menu.

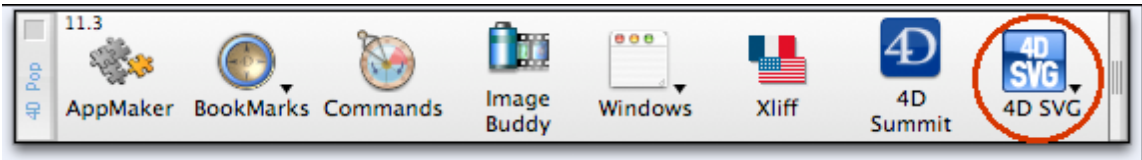
To display the viewer window, you can either:

- execute the `SVGTool_Display_viewer` method. In this case, the window appears empty.
- call the `SVGTool_SHOW_IN_VIEWER` method by passing a valid SVG reference in order to preview the picture reference (see the description of the `command`)
- click on the **SVG** button and choose the **SVG viewer** command in the 4D Pop component palette if you are using it (see below).

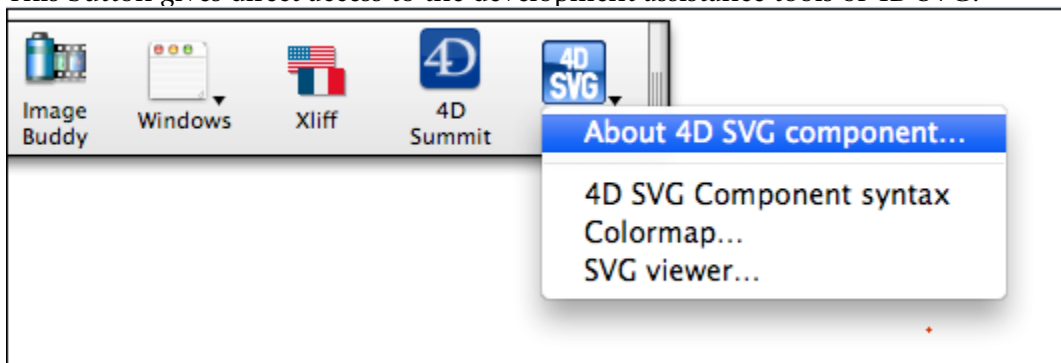
### Integration to 4D Pop

4D Pop is a set of components dedicated to developer productivity and grouped into a tool bar that can be integrated into the 4D development environment (see <http://www.4d.fr/products/4dpop.html>).

When you use 4D Pop and 4D SVG at the same time, a new button is added to the 4D Pop tool bar:



This button gives direct access to the development assistance tools of 4D SVG:



**SVG\_Ref**

Most of the 4D SVG component commands manipulate SVG structures via **SVG\_Ref** type references.

An **SVG\_Ref** is a 16-character string type expression. It uniquely identifies a SVG structure loaded in memory. This can be an SVG document loaded via the **SVG\_Copy**, **SVG\_New**, **SVG\_Open\_picture** or **SVG\_Open\_file** commands, or any SVG structure handled by programming (object, filter, path, etc.).

An **SVG\_Ref** is an XML reference. All **SVG\_Ref** references can be used as **elementRef** parameters for the 4D XML DOM commands.

Once you no longer need it, remember to call the **SVG\_CLEAR** command with the **SVG\_Ref** reference in order to free up memory.

**Optional parameters**

Unless otherwise mentioned, optional number arguments are ignored if their value is equal to -1 and text arguments are ignored if an empty string is passed.

**Coordinates**

Unless otherwise mentioned, the position (x, y) and size (width, height, radius) parameters are expected in the current user coordinate system.



# 2

---

## Attributes



SVG\_GET\_ATTRIBUTES (svgObject; namesArrayPointer; valuesArrayPointer)

Parameter	Type		Description
svgObject	SVG_Ref	→	SVG reference
namesArrayPointer	Pointer	→	Alpha array of attribute names
valuesArrayPointer	Pointer	→	Alpha array of attribute values

### Description

The SVG\_GET\_ATTRIBUTES command fills the arrays pointed to by namesArrayPointer and valuesArrayPointer respectively with the names and values of the attributes of the element whose reference is passed in the svgObject parameter. If svgObject is not valid or if this attribute does not exist, an error is generated.

### See Also

SVG\_SET\_ATTRIBUTES, SVG\_SET\_ATTRIBUTES\_BY\_ARRAYS.

SVG\_Get\_ID (svgObject) → String

Parameter	Type	Description
svgObject	SVG_Ref →	Reference of SVG element
Function result	String ←	Name of element

### Description

The SVG\_Get\_ID command returns the value of the 'id' attribute of the element whose reference is passed in the svgObject parameter. If svgObject is not valid or if this attribute does not exist, an error is generated.

### See Also

SVG\_Find\_ID, SVG\_SET\_ID.

SVG\_SET\_ATTRIBUTES (svgObject; attributeName; attributeValue{; attributeName2; attributeValue2; ...; attributeNameN; attributeValueN})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
attributeName	String	→ Name of attribute to set
attributeValue	String	→ Value of attribute

### Description

The SVG\_SET\_ATTRIBUTES command can be used to assign one or more custom attributes to an SVG object having the svgObject reference. If one or more of these attributes already exist, their values are replaced by those passed as parameters.

The attributes and their values are passed as paired parameters.

### Example

```
$svg:=SVG_New
$object:=SVG_New_rect ($svg; 10; 10; 200; 200; 0; 0; "black";"white"; 2)
SVG_SET_ATTRIBUTES ($object; "style"; "fill:red; stroke:blue; stroke-width:3")
```

### See Also

SVG\_GET\_ATTRIBUTES, SVG\_SET\_ATTRIBUTES\_BY\_ARRAYS.

---

SVG\_SET\_ATTRIBUTES\_BY\_ARRAYS (svgObject; namesArrayPointer; valuesArrayPointer)

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
namesArrayPointer	Pointer	→ Names of attributes
valuesArrayPointer	Pointer	→ Synchronized values of attributes

### Description

The SVG\_SET\_ATTRIBUTES\_BY\_ARRAYS command can be used to assign one or more custom attributes to an SVG object having the `svgObject` reference. If one or more of these attributes already exist, their values will be replaced by those passed as parameters.

The attributes and their values are passed using two arrays, to which `namesArrayPointer` and `valuesArrayPointer` point.

### Example

```

$svg:=SVG_New
$object:=SVG_New_rect ($svg; 10; 10; 200; 200; 0; 0; "black";"white"; 2)
ARRAY TEXT($attributes; 0)
ARRAY TEXT ($values; 0)
APPEND TO ARRAY($attributes; "fill")
APPEND TO ARRAY($values; "red")
APPEND TO ARRAY($attributes; "stroke")
APPEND TO ARRAY($values; "blue")
APPEND TO ARRAY($attributes; "stroke-width")
APPEND TO ARRAY($values; "3")
SVG_SET_ATTRIBUTES_BY_ARRAYS ($object; -> $attributes; -> $values)

```

### See Also

SVG\_GET\_ATTRIBUTES, SVG\_SET\_ATTRIBUTES.

SVG\_SET\_DIMENSIONS (svgObject; width{; height{; unit{}})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
width	Number	→ Dimension on the X axis
height	Number	→ Dimension on the Y axis
unit	String	→ Unit of measurement

**Description**

The SVG\_SET\_DIMENSIONS command can be used to set the dimensions for the SVG object having the svgObject reference. If these attributes already exist, their values are replaced by those passed as parameters.

If the unit parameter is passed, it will be used. The expected values are: *px, pt, pc, cm, mm, in, em, ex* or *%*. An incorrect unit value generates an error. If the parameter is omitted, the values of the width and height parameters are expected in the user coordinate system.

**Example**

```
$svg :=SVG_New ` Create a new document
$object:=SVG_New_rect ($svg; 10; 10; 200; 200; 0; 0; "black";"white"; 2)
SVG_SET_DIMENSIONS ($object; -1; 400)`New height
```

SVG\_SET\_FILL\_BRUSH (svgObject; color)

Parameter	Type	Description
svgObject	SVG_Ref →	Reference of SVG element
color	String →	Color expression

### Description

The **SVG\_SET\_FILL\_BRUSH** command can be used to set the fill color for the SVG object having the **svgObject** reference. If this attribute already exists, its value is replaced by the value passed in the parameter.

For more information about colors, please refer to the “SVG Colors” section.

### Example

```
$svg:=SVG_New  
$object:=SVG_New_rect ($svg; 10; 10; 200; 200; 0; 0; "black";"white"; 2)  
SVG_SET_FILL_BRUSH ($object; "blue")
```

### See Also

SVG Colors, SVG\_SET\_STROKE\_BRUSH.



SVG\_SET\_FILTER (svgObject; url)

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
url	String	→ Name of filter

### Description

The SVG\_SET\_FILTER command can be used to associate a filter with the object having the svgObject reference. If svgObject is not a valid reference, an error is generated. If the attribute already exists, its value is replaced.

The url parameter is the name of the filter to be used as specified by the SVG\_Define\_filter command. If this name does not exist, an error is generated.

### Example

See the SVG\_Define\_filter command.

### See Also

SVG Filters, SVG\_Define\_filter.

SVG\_SET\_ID (svgObject; id)

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
id	String	→ ID to assign to object

### Description

The SVG\_SET\_ID command can be used to set the 'ID' property of the SVG object having the svgObject reference. If this attribute already exists, its value is replaced by the value passed in the parameter.

The object id is used to reference an object. This reference will then be recovered using the SVG\_Get\_ID command. The id is also used by the 4D SVG Find element ID by coordinates command (see the 4D documentation).

### Example

```
$svg:=SVG_New  
$object:=SVG_New_rect ($svg; 10; 10; 200; 200; 0; 0; "black"; "white"; 2)  
SVG_SET_ID ($object; "border")
```

### See Also

SVG\_Find\_ID, SVG\_Get\_ID.

SVG\_SET\_MARKER (svgObject; url{; position})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
url	String	→ Name of marker
position	String	→ Position of marker

### Description

The SVG\_SET\_MARKER command can be used to associate a marker with the object having the svgObject reference. If svgObject is not the reference of a 'line', 'path', 'polyline' or 'polygon' element, an error is generated. If the attribute already exists, its value is replaced.

The url parameter is the name of the marker element to be used as specified by the SVG\_Define\_marker command. If this name does not exist, an error is generated.

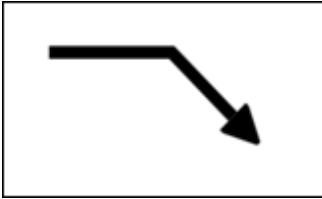
The optional position parameter can be used to set the position of the marker with respect to the object. It is possible to place different markers (if desired) at the beginning, end or any other peak of a path. The values may be as follows:

- start to place a marker at the beginning of the path
- end to place a marker at the end of the path
- middle to place a marker at each peak other than at the beginning and end.
- all to place markers at all peaks of the path.

If this parameter is omitted, the marker will be placed at the end of the path.

## Examples

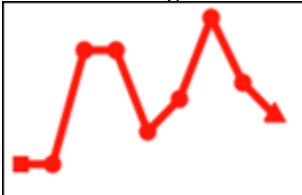
1. Draw an arrow:



```
$SVG:= SVG_New
  `Set the arrow
$arrow:=SVG_Define_marker ($SVG;"arrow";0;5;4;3;-1)
SVG_SET_VIEWBOX ($arrow;0;0;10;10)
$path:=SVG_New_path ($arrow;0;0)
SVG_SET_FILL_BRUSH ($path;"black")
SVG_PATH_LINE_TO ($path;10;5)
SVG_PATH_LINE_TO ($path;0;10)
SVG_PATH_CLOSE ($path)

$line:=SVG_New_path ($SVG;100;75)
SVG_SET_STROKE_WIDTH ($line;10)
SVG_PATH_LINE_TO ($line;200;75)
SVG_PATH_LINE_TO ($line;250;125)
  `Put an arrow at the end of a path
SVG_SET_MARKER ($line;" arrow ")
```

2. Draw a diagram with different markers at the beginning and end:



```
$SVG:= SVG_New
SVG_SET_DEFAULT_BRUSHES ("red";"red")

  `Set a circle to mark the points
$point:=SVG_Define_marker ($SVG;"pointMarker";2;2;3;3)
SVG_SET_VIEWBOX ($point;0;0;4;4)
SVG_New_circle ($point;2;2;1)
```

```

    `Set a square for the starting point
$start:=SVG_Define_marker ($SVG;"startMarker";1;1;2;2)
SVG_New_rect ($start;0;0;2;2)

    `Set a triangle for the end point
$end:=SVG_Define_marker ($SVG;"endMarker";5;5;3;3;60)
SVG_SET_VIEWBOX ($end;0;0;10;10)
SVG_New_regular_polygon ($end;10;3)

ARRAY LONGINT($tX;0)
ARRAY LONGINT($tY;0)
    `X axis
For ($Lon_i;0;200;20)
    APPEND TO ARRAY($tX;$Lon_i+10)
End for
    `Data
APPEND TO ARRAY($tY;100)
APPEND TO ARRAY($tY;100)
APPEND TO ARRAY($tY;30)
APPEND TO ARRAY($tY;30)
APPEND TO ARRAY($tY;80)
APPEND TO ARRAY($tY;60)
APPEND TO ARRAY($tY;10)
APPEND TO ARRAY($tY;40)
APPEND TO ARRAY($tY;50)
APPEND TO ARRAY($tY;70)
$line:=SVG_New_polyline_by_arrays ($SVG;->$tX;->$tY;"red";"none";5)
    `Arrange the markers:
SVG_SET_MARKER ($line;"startMarker";"start")
SVG_SET_MARKER ($line;"pointMarker";"middle")
SVG_SET_MARKER ($line;"endMarker";"end")

```

### See Also

SVG\_Define\_marker.

SVG\_SET\_OPACITY (svgObject; background{; line})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
background	Longint	→ Opacity (%)
line	Longint	→ Opacity (%)

### Description

The SVG\_SET\_OPACITY command can be used to set the opacity of the filling and the line of the object having the svgObject reference. If these attributes already exist, their values are replaced by those passed as parameters.

The values expected must be included between 0 and 100.

### Example

```
$svg :=SVG_New ` Create a new document
$object:=SVG_New_rect ($svg ;10;10;200;100;0;0;"red";"blue")
SVG_SET_OPACITY ($object; -1; 50) `Set the line opacity to 50%
```

### See Also

SVG Colors.

SVG\_SET\_ROUNDING\_RECT (svgObject; roundedX{; roundedY})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
roundedX	Number	→ Radius on X axis
roundedY	Number	→ Radius on Y axis

### Description

The SVG\_SET\_ROUNDING\_RECT command can be used to set the radii of the ellipse used to round the corners of a rectangle having the svgObject reference. If these attributes already exist, their values are replaced by those passed as parameters. If svgObject is not the reference of a rectangle, an error is generated.

The values are expected in the user coordinate system.

### Example

```
$svg :=SVG_New ` Create a new document  
$object:=SVG_New_rect ($svg ;10;10;200;100)  
SVG_SET_ROUNDING_RECT ($object; 20)`Round the corners
```

### See Also

SVG\_New\_rect, SVG\_SET\_STROKE\_LINEJOIN.

SVG\_SET\_STROKE\_BRUSH (svgObject; color)

Parameter	Type	Description
svgObject	SVG_Ref →	Reference of SVG element
color	String →	Color expression

### Description

The SVG\_SET\_STROKE\_BRUSH command can be used to set the color used for the lines of the SVG object having the svgObject reference. If this attribute already exists, its value is replaced by the value passed in the parameter.

For more information about colors, please refer to the “SVG Colors” section.

### Example

```
$svg:=SVG_New  
$object:=SVG_New_rect ($svg; 10; 10; 200; 200; 0; 0; "black";"white"; 2)  
SVG_SET_STROKE_BRUSH ($object; "red")
```

### See Also

SVG Colors, SVG\_GET\_DEFAULT\_BRUSHES, SVG\_SET\_FILL\_BRUSH.



SVG\_SET\_STROKE\_LINECAP (svgObject; mode)

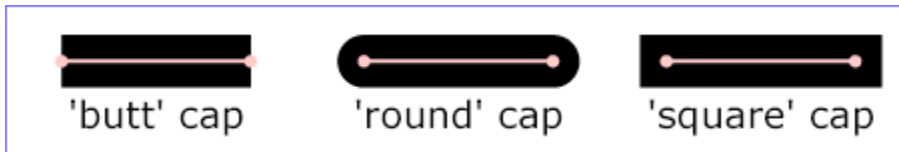
Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
mode	String	→ Rendering mode

### Description

The SVG\_SET\_STROKE\_LINECAP command can be used to specify the form of the path ends of the SVG object having the svgObject reference. If this attribute already exists, its value is replaced by the value passed as parameter.

The mode parameter must contain one of the following strings, handled by SVG:

- butt (default): standard
- round
- square
- inherit: inherited from parent object



If the mode parameter contains any other value, an error is generated.

### See Also

SVG\_SET\_STROKE\_LINEJOIN.

SVG\_SET\_STROKE\_LINEJOIN (svgObject; mode)

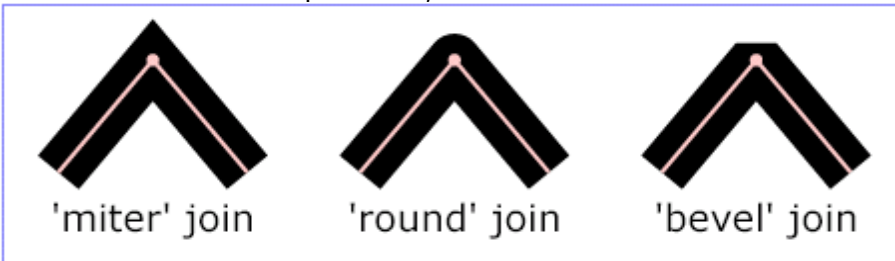
Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
mode	String	→ Rendering mode

### Description

The SVG\_SET\_STROKE\_LINEJOIN command can be used to specify the form of the path peaks of the SVG object having the svgObject reference. If this attribute already exists, its value is replaced by the value passed as parameter.

The mode parameter must contain one of the following values, managed by SVG:

- miter (default): standard
- round
- bevel
- inherit: inherited from parent object



If the mode parameter contains any other value, an error is generated.

### See Also

SVG\_SET\_ROUNDING\_RECT, SVG\_SET\_STROKE\_LINECAP.

SVG\_SET\_STROKE\_WIDTH (svgObject; strokeWidth{; unit})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
strokeWidth	Real	→ Line thickness
unit	String	→ Unit of measurement

### Description

The SVG\_SET\_STROKE\_WIDTH command can be used to set the thickness of lines for the SVG object having the svgObject reference. If this attribute already exists, its value is replaced by the value passed in the parameter.

Pass the value of the line thickness in strokeWidth. The optional unit parameter can be used to specify the unit to be used. You can pass one of the following values: *px*, *pt*, *pc*, *cm*, *mm*, *in*, *em*, *ex* or *%*. If the unit parameter is omitted, the strokeWidth parameter is expected in the user coordinate system

### Example

```
$svg :=SVG_New
SVG_SET_STROKE_WIDTH (SVG_New_rect ($svg; 10; 10; 200; 200; 0; 0; "black"; "white";
2); 10)
```

### See Also

SVG\_SET\_STROKE\_BRUSH.

SVG\_SET\_TRANSFORM\_FLIP (svgObject; horizontal{; vertical})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
horizontal	Boolean	→ Horizontal flip
vertical	Boolean	→ Vertical flip

**Description**

The SVG\_SET\_TRANSFORM\_FLIP command can be used to apply a horizontal and/or vertical flip to an SVG object having the svgObject reference.

If the horizontal parameter is set to True, a horizontal flip is applied.  
 If the vertical parameter is set to True, a vertical flip is applied.

**Example**

Flipping of a text object:



```
svgRef := SVG_New
SVG_SET_VIEWBOX (svgRef;0;0;400;200)
$tx:=SVG_New_text (svgRef;"4D";10;0;"";96)
SVG_SET_FONT_COLOR ($tx;"blue") `Change the color
```

Effect:  
\$tx:=SVG\_New\_text (svgRef;"4D";10;0;";96) `Take the same text  
SVG\_SET\_FONT\_COLOR (\$tx;"lightblue") `Change the color  
SVG\_SET\_TRANSFORM\_FLIP (\$tx;Vrai) `Apply a vertical flip  
SVG\_SET\_TRANSFORM\_SKEW (\$tx;-10) `Incline  
SVG\_SET\_TRANSFORM\_TRANSLATE (\$tx;-17;-193) `Reposition

**See Also**

SVG\_SET\_TRANSFORM\_SKEW.



SVG\_SET\_TRANSFORM\_MATRIX (objectRef; a; b{; c; d{; e; f})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
a	Number	→ Element a of transform matrix
b	Number	→ Element b of transform matrix
c	Number	→ Element c of transform matrix
d	Number	→ Element d of transform matrix
e	Number	→ Element e of transform matrix
f	Number	→ Element f of transform matrix

### Description

The SVG\_SET\_TRANSFORM\_MATRIX command applies a matrix transformation to the SVG object having the svgObject reference.

This type of transformation can be used to combine transformations like, for example, a rotation and a translation.

### Example

*Writing with SVG is easy*

**SVG\_SET\_TRANSFORM\_MATRIX (\$ID;0,707;-0,707;0,707;0,707;255,03;111,21)**

Is equivalent to applying the 3 following transformations:

**SVG\_SET\_TRANSFORM\_TRANSLATE (\$ID;50;90)**

**SVG\_SET\_TRANSFORM\_ROTATE (\$ID;-45)**

**SVG\_SET\_TRANSFORM\_TRANSLATE (\$ID;130;160)**

### See Also

SVG\_SET\_TRANSFORM\_ROTATE.

SVG\_SET\_TRANSFORM\_ROTATE (svgObject; angle{; x; y})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
angle	Number	→ Angle of rotation
x	Number	→ Coordinate on X axis of center of rotation
y	Number	→ Coordinate on Y axis of center of rotation

### Description

The SVG\_SET\_TRANSFORM\_ROTATE command applies a rotation of the value angle in degrees to the SVG object having the objectRef reference.

The angle of rotation is expected in degrees; the rotation is made clockwise.

If the optional x and y parameters are not passed, the rotation is carried out with respect to the origin of the current user coordinate system. If these parameters are provided, the rotation is carried out with respect to the coordinates passed (x, y).

### Example



```

svgRef := SVG_New
  `Draw a red rectangle with a blue border
$rec:=SVG_New_rect ($svg;150;50;200;400;0;0;"blue";"red";10)
  `Apply a rotation of 10° clockwise with respect to the center
SVG_SET_TRANSFORM_ROTATE ($rec;370;175;225)
  
```

### See Also

SVG\_SET\_TRANSFORM\_FLIP.



SVG\_SET\_TRANSFORM\_SCALE (svgObject; scaleX{; scaleY})

Parameter	Type	Description
svgObject	SVG_Ref →	Reference of SVG element
scaleX	Number →	Value on X axis
scaleY	Number →	Value on Y axis

### Description

The SVG\_SET\_TRANSFORM\_SCALE command applies a change of horizontal and/or vertical scale to an SVG object having the svgObject reference.

If the scaleX value is not null, the object is enlarged (value >1) or reduced (value < 1) horizontally for the number of units passed. The value 1 is equal to no change to the object scale.

If the scaleY parameter is provided, the object is enlarged (value >1) or reduced (value < 1) vertically for the number of units passed. The value 1 is equal to no change to the object scale. If this parameter is omitted, its value is supposed to be equal to scaleX.

### Example



```
svgRef := SVG_New
$Text:=SVG_New_text ($SVG;"Hello world!";5)
SVG_SET_TRANSFORM_SCALE ($Text;3;12) `Zoom x*3 y*12
```

### See Also

SVG\_SET\_TRANSFORM\_MATRIX.

SVG\_SET\_TRANSFORM\_SKEW (svgObject; horizontal{; vertical})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
horizontal	Number	→ Value of incline along X axis
vertical	Number	→ Value of incline along Y axis

### Description

The SVG\_SET\_TRANSFORM\_SKEW command specifies a horizontal and/or vertical incline for the SVG object having the svgObject reference.

If the value of the horizontal parameter is not null, the object will be inclined horizontally according to the number of units passed; otherwise, it is ignored.

If the value of the vertical parameter is not null, the object will be inclined vertically according to the number of units passed.

### Example



```
$svg := SVG_New
  `Draw a background
SVG_New_rect ($svg; 0; 0; 270; 160; 10; 10; "black"; "gray")
  `Place the text...
$tx:=SVG_New_text ($svg;"Hello world!"; 100; 5; ""; 48)
  `in white
SVG_SET_FONT_COLOR ($tx; "white")
  `Incline it
SVG_SET_TRANSFORM_SKEW ($tx; -50; 10) `Incline
```

### See Also

SVG\_SET\_TRANSFORM\_FLIP.

SVG\_SET\_TRANSFORM\_TRANSLATE (svgObject; x{; y})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
x	Number	→ Coordinate on X axis
y	Number	→ Coordinate on Y axis

### Description

The SVG\_SET\_TRANSFORM\_TRANSLATE command specifies a horizontal and/or vertical relocation of the SVG object having the svgObject reference.

If the x value is not null, the object will be moved horizontally for the number of units passed; otherwise, it will be ignored.

If the y parameter is provided, the object will be moved vertically for the number of units passed.

### Example



```
svgRef := SVG_New
  `Draw a red rectangle
$Object:=SVG_New_rect (svgRef;0;0;200;100;0;0;"black";"red")
  `Draw a square at 0,0
$Object:=SVG_New_rect (svgRef;0;0;20;20)
  `Move the square to 150,50
SVG_SET_TRANSFORM_TRANSLATE ($Object;150;50)
```

### See Also

SVG\_SET\_TRANSFORM\_ROTATE.

SVG\_SET\_VIEWBOX (svgObject; x; y; width; height{; mode})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
x	Longint	→ X position of viewBox
y	Longint	→ Y position of viewBox
width	Longint	→ Width of viewBox
height	Longint	→ Height of viewBox
mode	Text	→ Adjustment to viewBox

### Description

The SVG\_SET\_VIEWBOX command can be used to specify the viewBox of the SVG object having the svgObject reference. If this attribute already exists, its value is replaced by the value passed in the parameter.

The values are expected in the user coordinate system.

The optional mode parameter can be used to indicate if the graphic must be fitted, and how so, to the size of the viewBox. The value expected for mode must be one recognized by SVG: 'none', 'xMinYMin', 'xMidYMin', 'xMaxYMin', 'xMinYMid', 'xMidYMid', 'xMaxYMid', 'xMinYMax', 'xMidYMax', 'xMaxYMax' and 'true' (for xMidYMid).

### Example

```

`Create an SVG document of 4x8cm
$svg:=SVG_New
SVG_SET_DIMENSIONS($SVG;4;8;"cm")
`Declare the user coordinate system here as 1cm = 50 user points
SVG_SET_VIEWBOX ($svg; 0; 0; 1000; 2000; "true")

```

### See Also

SVG\_SET\_VIEWPORT\_FILL.

SVG\_SET\_VIEWPORT\_FILL (svgObject{; color{; opacity})

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
color	String	→ Fill color
opacity	Longint	→ Percentage of opacity

### Description

The SVG\_SET\_VIEWPORT\_FILL command can be used to set the background color of an SVG document having the svgObject reference.

If this attribute already exists, its value is replaced by the value passed as parameter. If svgObject is an SVG element that does not accept this attribute, an error is generated.

The optional color parameter indicates the color to be used for the picture background. If this parameter is omitted or contains an empty string, white will be used. For more information about colors, please refer to the SVG Colors section.

The optional opacity parameter can be used to specify the value of the percentage of opacity to be applied to this fill. If this parameter is omitted or if no opacity has been specified for the document, the value 100% is used.

### See Also

SVG Colors.

SVG\_SET\_VISIBILITY (svgObject{; hide))

Parameter	Type		Description
svgObject	SVG_Ref	→	Reference of SVG element
hide	Boolean	→	True = Show, False = Hide

### Description

The SVG\_SET\_VISIBILITY command hides or shows an SVG object having the objectRef reference. If objectRef is not the reference of an object that can be hidden, an error is generated.

If the optional hide parameter is set to True or omitted, the object will be shown. If it is False, the object will be hidden.

### Example

```
$svg :=SVG_New  
$object:=SVG_New_rect ($svg; 10; 10; 200; 200; 0; 0; "black";" white"; 2)  
SVG_SET_VISIBILITY ($object; False) `The object is described but will not be rendered.
```

SVG\_SET\_XY (svgObject; x{; y})

Parameter	Type		Description
svgObject	SVG_Ref	→	Reference of SVG element
x	Number	→	Coordinate on X axis
y	Number	→	Coordinate on Y axis

### Description

The SVG\_SET\_XY command can be used to set the coordinates of the top left corner of the rectangular area where the SVG object having the svgObject reference is placed. If these attributes already exist, their values are replaced by those passed as parameters. If svgObject is an SVG element that does not accept this attribute, an error is generated.

The values are expected in the user coordinate system.

### Example

```
$svg :=SVG_New `Create a new document  
$object:=SVG_New_image ($svg;"#Pictures/logo4D.png") `Place the logo  
SVG_SET_XY ($object; 10; 40)`Modify the position of the picture
```





# 3

---

## Colors and Gradients



**Definition of colors**

SVG recognizes all the alternative syntaxes for the colors defined in the CSS2 standard. The commands of the 4D SVG component support all these syntaxes.

A color can be expressed in one of the following forms:

- RGB format

<b>Format</b>	<b>Example</b>
#rgb	#f00
#rrggbb	#ff0000
rgb(r,g,b)	rgb(255, 0, 0)
	rgb(100%, 0%, 0%)

- "Color" keyword format

SVG accepts an extensive list of color name keywords, for example "red".

The list of keywords as well as their RGB correspondence is found in Appendix A, Table of colors. You can also view this list and insert the color values directly via the 4D SVG Color palette. For more information about this point, please refer to the Development Tools section.

**Opacity**

It is possible to specify the opacity in the color expressions of the component commands by using the syntax "color:opacity" where opacity is a number included between 0 (no color) and 100 (color completely opaque). So "red:50" will be interpreted as a red at 50% opacity.

**Gradients**

Gradients are progressive transitions of color along a vector. These gradients are set with the `SVG_Define_gradient`, `SVG_Define_linear_gradient` and `SVG_Define_radial_gradient` commands. Once set, the gradients are used by reference using the "url(#GradientName)" syntax.

Similarly, it is possible to set a custom color associated with an opacity using the `SVG_Define_solidColor` command.

SVG\_Color\_grey (percentage) → String

Parameter	Type		Description
percentage	Integer	→	Intensity of gray
Function result	String	←	Color string

### Description

The `SVG_Color_grey` command returns a string expressing a gray color having a percentage intensity. The string returned is in "RGB(red, green, blue)" form where the 3 values are equal, the syntax recognized by SVG rendering engines.

### Example

```
$txtColor:= SVG_Color_grey (60)
`$txtColor is "rgb(102,102,102)"
```

### See Also

`SVG_Color_RGB_from_long`.

SVG\_Color\_RGB\_from\_long (color{; format}) → String

Parameter	Type		Description
color	Longint	→	Value of color
format	Integer	→	Format of color
Function result	String	←	Color string

**Description**

The SVG\_Color\_RGB\_from\_long command returns a string expressing the color color passed as an argument. The string returned is in "RGB(red, green, blue)" form, the syntax recognized by SVG rendering engines.

The color parameter is a 4-byte longint whose format (0x00RRGGBB) is described below (the bytes are numbered from 0 to 3, from right to left):

Byte	Description
2	Red component of the color (0..255)
1	Green component of the color (0..255)
0	Blue component of the color (0..255)

The optional format parameter can be used to specify the desired format for the color string returned. The values are:

Value	Format
1 (default)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

**Example**

```
$txtColor:=SVG_Color_RGB_from_long ($color)
`$txtColor is "rgb(255,128,0)" if $color is 16744448 (orange)
```

**See Also**

SVG\_Color\_grey.

SVG\_GET\_DEFAULT\_BRUSHES (line{; background})

Parameter	Type	Description
line	Pointer	→ Alpha variable
background	Pointer	→ Alpha variable

### Description

The SVG\_GET\_DEFAULT\_BRUSHES command returns, in the variable pointed to by line, the current default color for drawing lines.

If the optional background parameter is passed, the variable pointed to by this parameter will receive the current default color used for backgrounds.

If they have not been modified, these colors are, respectively, black and white.

### Example

See the SVG\_SET\_DEFAULT\_BRUSHES command.

### See Also

SVG\_SET\_DEFAULT\_BRUSHES, SVG\_SET\_STROKE\_BRUSH.

SVG\_SET\_DEFAULT\_BRUSHES (line{; background})

Parameter	Type	Description
line	String	→ Color
background	String	→ Color

### Description

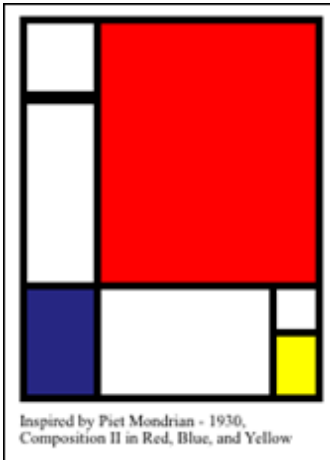
The SVG\_SET\_DEFAULT\_BRUSHES command can be used to set the default colors used by the component.

The line parameter contains the new color that will be used for lines. The optional background parameter contains the new color to be used for drawing backgrounds.

You can pass an empty string in either of these parameters in order to reset the default value of the component; in other words, black for the lines and white for the background.

### Example

Like Mondrian...



```

$svg:=SVG_New
  `Set the default colors
SVG_SET_DEFAULT_BRUSHES ("black";"white")
  `4-point thick lines
SVG_SET_STROKE_WIDTH ($svg;4)
$g:=SVG_New_group ($svg)
SVG_New_rect ($g;2;2;40;40)
SVG_New_rect ($g;2;45;40;100)
SVG_SET_FILL_BRUSH (SVG_New_rect ($g;2;144;40;60);"midnightblue")
SVG_SET_FILL_BRUSH (SVG_New_rect ($g;42;2;120;142);"red")
SVG_New_rect ($g;42;144;95;60)
SVG_New_rect ($g;137;144;25;25)
SVG_SET_FILL_BRUSH (SVG_New_rect ($g;137;169;25;35);"yellow")
SVG_SET_TRANSFORM_TRANSLATE ($g;10;10)
  `Caption
SVG_New_text ($svg;"Inspired by Piet Mondrian - 1930,\rComposition II in Red, Blue, and
                                                    Yellow";10;220;"";9)

```

### See Also

SVG\_GET\_DEFAULT\_BRUSHES, SVG\_SET\_STROKE\_BRUSH.



# 4

---

# Documents



SVG\_CLEAR {(svgObject)}

Parameter	Type	Description
svgObject	SVG_Ref →	SVG object reference

### Description

The SVG\_CLEAR command frees the memory taken up by the SVG object designated by svgObject.

If svgObject is not an SVG root object created with the SVG\_New, SVG\_Copy or SVG\_Open\_file commands, an error is generated.

If svgObject is not passed, the command frees all the SVG objects created using SVG\_New, SVG\_Copy or SVG\_Open\_file commands. This syntax is useful during the development phase during which an SVG reference can be created but the memory was not released because of an error that prevents the execution of the method from completing. In a final development, any SVG reference that is no longer used must be freed using the SVG\_CLEAR command.

### See Also

SVG\_Copy, SVG\_New, SVG\_Open\_file.

---

SVG\_Copy (svgObject) → SVG\_Ref

Parameter	Type		Description
svgObject	SVG_Ref	→	Reference of SVG objec to copy
Function result	SVG_Ref	←	Reference of new SVG object

### Description

The `SVG_Copy` command creates a new SVG document that is a copy of the document referenced by `svgObject`.

The command returns a 16-character string (`SVG_Ref`) that consists of the reference in memory of the document virtual structure. This reference must be used with the other commands of the component.

**Important:** Once you no longer need it, do not forget to call the `SVG_CLEAR` command with this reference in order to free up the memory.

### Example

```
svgRef:=SVG_New
...
svgRef Copy:=SVG_Copy (svgRef)
```

### See Also

`SVG_CLEAR`, `SVG_New`.

SVG\_Export\_to\_picture (svgObject{; exportType}) → Picture

Parameter	Type		Description
svgObject	SVG_Ref	→	SVG object reference
exportType	Longint	→	0 = Do not store data source 1 (default) = Copy data source 2 = Own data source
Function result	Picture	←	Picture rendered by SVG engine

**Description**

The SVG\_Export\_to\_picture command returns the picture described by the SVG structure referenced by svgObject.

The optional exportType parameter can be used to specify the way in which the XML data source must be handled by the command. For more information about this parameter, refer to the description of the 4D SVG EXPORT TO PICTURE command. If this parameter is omitted, the default value is 1, Copy XML Data Source.

**Example**

```

svgRef:=SVG_New(500;200; "Test component")
...
MyPicture:= SVG_Export_to_picture (svgRef ;0)

SVG_CLEAR (svgRef )
    
```

**See Also**

SVG\_Export\_to\_XML, SVG\_Open\_picture, SVG\_SAVE\_AS\_PICTURE.

SVG\_Export\_to\_XML (svgObject) → Text

Parameter	Type		Description
svgObject	SVG_Ref	→	SVG object reference
Function result	Text	←	XML text of SVG document

### Description

The SVG\_Export\_to\_XML command returns the XML text of the description of the SVG structure referenced by svgObject.

### Example

```
svgRef :=SVG_New (500; 200; " Test component ")
...
MyText:= SVG_Export_to_XML (svgRef )

SVG_CLEAR (svgRef )
```

### See Also

SVG\_Export\_to\_picture, SVG\_SAVE\_AS\_TEXT.

---

SVG\_New ({width}; height; title; description; rectangle; display}})) → SVG\_Ref

Parameter	Type		Description
width	Number	→	Document width
height	Number	→	Document height
title	String	→	Document title
description	String	→	Description
rectangle	Boolean	→	Set viewBox
display	Integer	→	Picture display format
Function result	SVG_Ref	←	SVG object reference

### Description

The SVG\_New command creates a new SVG document and returns its reference number.

The optional width and height parameters can be used to limit the space of the SVG document to the dimensions indicated. These 2 parameters are expected in user points ('px'); if you want to specify another unit, you must use the SVG\_SET\_DIMENSIONS command.

The optional title and description parameters can be used to give information about the contents.

If you pass True in the optional rectangle parameter, the viewBox ('viewBox' attribute) is automatically set to the size of the document created.

**Note:** It is possible to modify the coordinates of the graphic viewBox and to adjust the fitting of the picture to it more precisely using the SVG\_SET\_VIEWBOX command.

The optional display parameter can be used to indicate whether the graphic must be fitted to the size of the document. You can pass one of the following 4D picture display format constants as parameter: Scaled to fit prop centered or Scaled to Fit.

The command returns a 16-character string (SVG\_Ref) that consists of the reference in memory of the document virtual structure. This reference must be used with the other commands of the component.

**Important:** Once you no longer need it, do not forget to call the SVG\_CLEAR command with this reference in order to free up the memory.

### Example

```
svgRef:=SVG_New  
svgRef:=SVG_New(500;200)  
svgRef:= SVG_New (900; 700; "SVG component test"; "This is an example"; True;  
Scaled to Fit)
```

### See Also

SVG\_CLEAR, SVG\_Copy, SVG\_Open\_file.



SVG\_Open\_file (path) → SVG\_Ref

Parameter	Type		Description
path	String	→	Pathname of SVG document to open
Function result	SVG_Ref	←	Document reference

### Description

The `SVG_Open_file` command parses (and validates with the DTD) the SVG document found at the location designated by the `path` parameter and returns an SVG reference (16-character string) for this document.

**Important:** Once you no longer need it, do not forget to call the `SVG_CLEAR` command with this reference in order to free up the memory.

### See Also

`SVG_CLEAR`, `SVG_Copy`, `SVG_New`.

SVG\_Open\_picture (picture) → SVG\_Ref

Parameter	Type		Description
picture	Picture	→	4D picture field or variable
Function result	SVG_Ref	←	Reference of SVG document

### Description

The `SVG_Open_picture` command analyzes an SVG picture and returns an SVG reference for this picture. If picture does not contain an SVG picture, the command returns an empty string.

**Important:** Once you no longer need it, remember to call the `SVG_CLEAR` command with this reference in order to free up the memory.

### Example

```
READ PICTURE FILE("";$picture)
If(OK=1)
    $ref:=SVG_Open_picture ($picture)
    ...
    SVG_CLEAR ($ref)
End if
```

### See Also

`SVG_Export_to_picture`.

SVG\_SAVE\_AS\_PICTURE (svgObject; document{; codec})

Parameter	Type	Description
svgObject	SVG_Ref	→ SVG object reference
document	String	→ Document name or Full pathname of document
codec	String	→ Picture codec ID

### Description

The SVG\_SAVE\_AS\_PICTURE command writes the contents of the SVG object specified by svgObject into the picture file specified by document. If svgObject is not an SVG document, an error is generated.

In document, you can pass the full pathname of the file, or only the file name – in which case the file will be created next to the database structure file. If you pass an empty string ("" ) in document, the standard Save file dialog box appears so that the user can specify the name, location and format of the file to be created.

The optional codec parameter can be used to specify the format in which to save the picture. If this parameter is omitted, the picture is saved in png format.

### Example

```
svgRef:=SVG_New(500;200; " Sales statistics ")
...
SVG_SAVE_AS_PICTURE (svgRef ;"test.png") `Save
SVG_SAVE_AS_PICTURE (svgRef ;"test.gif" ;".gif")
SVG_CLEAR(svgRef )
```

### See Also

SVG\_Export\_to\_picture.

SVG\_SAVE\_AS\_TEXT (svgObject; document)

Parameter	Type	Description
svgObject	SVG_Ref	→ SVG object reference
document	String	→ Document name or Full pathname of document

### Description

The SVG\_SAVE\_AS\_TEXT command writes the content of the SVG object specified by svgObject into the disk file specified by document. If svgObject is not an SVG document, an error is generated.

In document, you can pass the full pathname of the file, or only the file name – in which case the file will be created next to the database structure file. If you pass an empty string ("") in document, the standard Save file dialog box appears so that the user can specify the name, location and format of the file to be created.

### Example

```
svgRef:=SVG_New(500;200; "Sales statistics")
```

```
...
```

```
SVG_SAVE_AS_TEXT (svgRef ;"test.svg") `The document is saved next to the structure  
SVG_CLEAR (svgRef)
```

### See Also

SVG\_Export\_to\_XML.

SVG\_Validate\_file (path) → Boolean

<b>Parameter</b>	<b>Type</b>		<b>Description</b>
path	String	→	Pathname of SVG document to validate
Function result	Boolean	←	True if the document corresponds to the DTD

**Description**

The `SVG_Validate_file` command attempts to validate the document specified in `path` on disk with the DTD (1.0). The command returns `True` if the document is well formed and `False` otherwise.



# 5

---

# Drawing





SVG\_ADD\_POINT (parentSVGObject; x; y{; x2; y2; ...; xN; yN})

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
x	Number	→	Coordinate on X axis of new point(s)
y	Number	→	Coordinate on Y axis of new point(s)

### Description

The SVG\_ADD\_POINT command adds one or more segments to the path referenced by parentSVGObject. The path may be of the 'path', 'polyline' or 'polygon' type. If parentSVGObject is not a path reference of this type, an error is generated.

If several pairs of coordinates are passed, the different points will be added successively. In this case, if the last pair of coordinates is incomplete (missing y), it will be ignored.

### Example

See the examples for the SVG\_New\_path command.

### See Also

SVG\_New\_path, SVG\_New\_polygon, SVG\_New\_polyline.

SVG\_New\_arc (parentSVGObject; x; y; radius; start; end{; foregroundColor{; backgroundColor{; strokeWidth}}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
x	Number	→	Coordinate on center X axis
y	Number	→	Coordinate on center Y axis
radius	Number	→	Radius of circle
start	Number	→	Value in degrees of start of arc
end	Number	→	Value in degrees of end of arc
foregroundColor	String	→	Color or gradient name
backgroundColor	String	→	Color or gradient name
strokeWidth	Real	→	Line thickness
Function result	SVG_Ref	←	Reference of arc

### Description

The SVG\_New\_arc command creates a new circle arc in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.

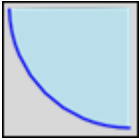
## Examples

1. Draw an arc from  $0^\circ$  to  $90^\circ$  (default fill and border color, default line thickness):



```
svgRef:= SVG_New  
objectRef:= SVG_New_arc (svgRef;100;100;90;90;180)
```

2. Draw the arc from  $90^\circ$  to  $180^\circ$  of a light blue circle with a blue edge and a 2-point link thickness:



```
svgRef:= SVG_New  
objectRef:= SVG_New_arc (svgRef;100;100;90;180;270;"blue";"lightblue";2)
```

## See Also

SVG\_PATH\_ARC.

SVG\_New\_circle (parentSVGObject; x; y; radius{; foregroundColor{; backgroundColor{; strokeWidth}}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
x	Number	→	Coordinate on center X axis
y	Number	→	Coordinate on center Y axis
radius	Number	→	Radius of circle
foregroundColor	String	→	Color or gradient name
backgroundColor	String	→	Color or gradient name
strokeWidth	Real	→	Line thickness
Function result	SVG_Ref	←	Reference of circle

### Description

The SVG\_New\_circle command creates a new circle in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

The circle is positioned and sized according to the center coordinates (x and y) and the radius passed as a parameter.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.

## Examples

1. Draw a circle (default fill and border color, default line thickness):



```
svgRef:= SVG_New  
objectRef:=SVG_New_circle (svgRef;100;100;90)
```

2. Draw a light blue circle with a blue edge and a 2-point link thickness:



```
svgRef:= SVG_New  
objectRef:= SVG_New_circle (svgRef;100;100;90;"blue";"lightblue";2)
```

## See Also

SVG\_New\_ellipse.

SVG\_New\_ellipse (parentSVGObject; x; y; xRadius; yRadius{; foregroundColor{; backgroundColor{; strokeWidth}}}) → SVG\_Ref

Parameter	Type	Description
parentSVGObject	SVG_Ref	→ Reference of parent element
x	Number	→ Coordinate on center X axis of ellipse
y	Number	→ Coordinate on center Y axis of ellipse
xRadius	Number	→ Radius on X axis
yRadius	Number	→ Radius on Y axis
foregroundColor	String	→ Color or gradient name
backgroundColor	String	→ Color or gradient name
strokeWidth	Real	→ Line thickness
Function result	SVG_Ref	← Reference of ellipse

### Description

The SVG\_New\_ellipse command creates a new ellipse in the SVG container designated by parentSVGObject. If parentSVGObject is not an SVG document, an error is generated.

The ellipse is positioned and sized according to the values of x, y, width and height.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.

## Examples

1. Draw an ellipse (default fill and border color, default line thickness):



```
svgRef:= SVG_New  
objectRef:=SVG_New_ellipse (svgRef;100;50;90;40)
```

2. Draw a light blue ellipse with a blue edge and a 2-point line thickness:



```
svgRef:= SVG_New  
objectRef:= SVG_New_ellipse (svgRef;100;50;90;40;"blue";"lightblue";2)
```

## See Also

SVG\_New\_circle, SVG\_New\_ellipse\_bounded.

SVG\_New\_ellipse\_bounded (parentSVGObject; x; y; width; height{; foregroundColor{; backgroundColor{; strokeWidth}}}) → SVG\_Ref

Parameter	Type	Description
parentSVGObject	SVG_Ref	→ Reference of parent element
x	Number	→ Coordinate on X axis of upper left corner
y	Number	→ Coordinate on Y axis of upper left corner
width	Number	→ Width of bounding rectangle
height	Number	→ Height of bounding rectangle
foregroundColor	String	→ Color or gradient name
backgroundColor	String	→ Color or gradient name
strokeWidth	Real	→ Line thickness
Function result	SVG_Ref	← Reference of ellipse

### Description

The SVG\_New\_ellipse\_bounded command creates a new ellipse in the SVG container designated by parentSVGObject. If parentSVGObject is not an SVG document, an error is generated.

The ellipse created fits into the rectangle set by x, y, width and height.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.



## Examples

1. Draw an ellipse (default fill and border color, default line thickness):



```
svgRef := SVG_New  
objectRef:= SVG_New_ellipse_bounded (svgRef;10;10;200;100)
```

2. Draw a light blue ellipse with a blue edge and a 2-point line thickness:



```
svgRef:= SVG_New  
objectRef:= SVG_New_ellipse_bounded (svgRef;100;100;200;100;"blue";"lightblue";2)
```

## See Also

SVG\_New\_ellipse.

SVG\_New\_embedded\_image (parentSVGObject; picture; x; y) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
picture	Picture	→	Picture to be embedded
x	Number	→	Coordinate on X axis of upper left corner
y	Number	→	Coordinate on Y axis of upper left corner
Function result	SVG_Ref	←	SVG object reference

### Description

The SVG\_New\_embedded\_image command can be used to embed the picture picture in the SVG container designated by parentSVGObject and to return its reference. If parentSVGObject is not an SVG document, an error is generated.

The picture will be encoded in base64 then embedded in the document.

The picture parameter is a 4D picture field or variable.

The optional x and y parameters can be used to specify the position of the upper left corner of the picture in the SVG containers (default value 0).

### Example

Embed the 'logo4D.png' picture located in the 'Resources' folder:



```
svgRef:= SVG_New  
$Path :=Resources folder+"logo4D.png")  
READ PICTURE FILE($Path; $Picture)  
If (OK=1)  
    objectRef:=SVG_New_embedded_image (svgRef; $Picture)  
End if
```

### **See Also**

SVG\_New\_image, SVG\_Open\_picture.

SVG\_New\_image (parentRef; url{; x; y{; width; height}) → SVG\_Ref

Parameter	Type	Description
parentSVGObject	SVG_Ref	→ Reference of parent element
url	String	→ Address of picture
x	Number	→ Coordinate on X axis of upper left corner
y	Number	→ Coordinate on Y axis of upper left corner
width	Number	→ Width of picture
height	Number	→ Height of picture
Function result	SVG_Ref	← SVG object reference

### Description

The `SVG_New_image` command can be used to reference a picture at the url address in the SVG container designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG document, an error is generated.

The `url` parameter specifies the location of the picture and can take several forms:

- A **local URL** (a pathname in the form: `file://...`): in this case, the picture will only be displayed if the file is actually accessible at the time the picture is rendered. This local URL can be relative (in the form: `"#Pictures/myPicture.png"`); in this case the command prefixes the pathname with that of the **Resources** folder of the host database. If the `width` and `height` parameters are omitted, they will be calculated by the command (to be avoided since this is slower than when the sizes are known). In the case of a relative path, if it is not valid, an error is generated.
- A **non-local URL** (`http://mySite.com/pictures/myPicture.jpeg`). In this case, no verification is carried out concerning the validity of the link and an error will be generated if the `width` and `height` parameters are omitted.

The optional `x` and `y` parameters can be used to specify the position of the upper left corner of the picture in the SVG containers (default value 0).

The `width` and `height` parameters specify the size of the rectangle in which the picture will be displayed and thus determine the size and aspect ratio of the picture. These parameter are only optional in the case of a picture referenced by a relative path in the **Resources** folder of the host database. If `width` and/or `height` equal 0 then the picture is not rendered.

## Examples

1. Place the 'logo4D.png' picture located in the 'Pictures' folder of the 'Resources' folder:



```
svgRef:= SVG_New  
objectRef:=SVG_New_image (svgRef;"#Pictures/logo4D.png")
```

2. Place the '4dlogo.gif' picture that can be accessed in the 'pictures' directory of the '4d.com' site:



```
svgRef:= SVG_New  
objectRef:=SVG_New_image (svgRef;"http://www.4d.com/pictures/4dlogo.gif";20;20;39;  
53)
```

## See Also

SVG\_New\_embedded\_image, SVG\_Open\_picture.

SVG\_New\_line (parentSVGObject; startX; startY; endX; endY{; color{; strokeWidth}) → SVG\_Ref

Parameter	Type	Description
parentSVGObject	SVG_Ref	→ Reference of parent element
startX	Number	→ Horizontal start position
startY	Number	→ Vertical start position
endX	Number	→ Horizontal end position
endY	Number	→ Vertical end position
color	String	→ Color or gradient name
strokeWidth	Real	→ Line thickness
Function result	SVG_Ref	← Reference of line

### Description

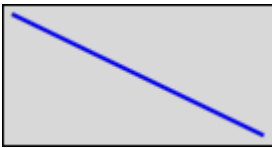
The SVG\_New\_line command creates a new line in the SVG container designated by parentSVGObject and returns its reference. The object is positioned according to the startX, startY, endX and endY coordinates. The SVG container can be the document root or any other reference to an SVG object that can contain this type of element.

The optional color parameter contains the name of the line color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the pen size expressed in pixels. Its default value is 1.

### Example

Draw a blue line that is 3 pixels thick:



```
svgRef:= SVG_New
objectRef:= SVG_New_line (svgRef;10;10;200;100;"blue";3)
```

### See Also

SVG\_New\_polyline, SVG\_PATH\_LINE\_TO.

SVG\_New\_path (parentSVGObject; x; y; foregroundColor{; backgroundColor{; strokeWidth}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
x	Number	→	Coordinate on X axis of start of path
y	Number	→	Coordinate on Y axis of start of path
foregroundColor	String	→	Color or gradient name
backgroundColor	String	→	Color or gradient name
strokeWidth	Real	→	Line thickness
Function result	SVG_Ref	←	SVG object reference

### Description

The `SVG_New_path` command starts a new path in the SVG container designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG document, an error is generated.

A path represents the outline of a shape. A path is depicted by calling upon the concept of a current point. By analogy with a drawing on paper, the current point can be assimilated to the position of the pen. This point can change and the outline of a shape (open or closed) can be traced by moving the pen along a straight or curved line.

Paths represent the geometry of the outline of an object, defined according to the statements of the following elements: `SVG_PATH_MOVE_TO` (establish a new current point), `SVG_PATH_LINE_TO` (draw a straight line), `SVG_PATH_CURVE` (draw a curve using a cubic Bezier curve), `SVG_PATH_ARC` (draw a circular or elliptical arc) and `SVG_PATH_CLOSE` (close the current form by drawing a line to the last beginning of the path). It is possible to have compound paths (in other words, a path with several subpaths) that can be used for effects such as a "doughnut hole" in objects.

The `x` and `y` parameters can be used to specify the start position of a path in the SVG container.

The optional `foregroundColor` and `backgroundColor` parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional `strokeWidth` parameter contains the size of the pen expressed in pixels. Its default value is 1.

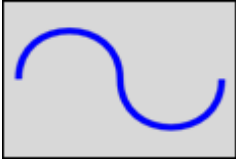
## Examples

1. Draw a closed broken line:



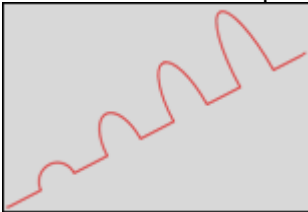
```
svgRef:= SVG_New
objectRef:=SVG_New_path (svgRef;20;20;"red";"none";5)
SVG_PATH_LINE_TO (objectRef;40)
SVG_PATH_LINE_TO (objectRef;40;40)
SVG_PATH_LINE_TO (objectRef;80;40;80;20;100;20;100;100;80;100;80;80;40;80;40;100;
20;100)
SVG_PATH_CLOSE (objectRef)
```

2. Draw a Bezier curve:



```
svgRef:= SVG_New
objectRef:=SVG_New_path (svgRef;100;200;"aquamarine";"none";10)
SVG_PATH_CURVE (objectRef;250;200;100;100;250;100)
SVG_PATH_CURVE (objectRef;400;200;400;300)
```

3. Arc commands in path data:



```
svgRef:= SVG_New
objectRef:=SVG_New_path (svgRef;20;300;"red";"none";2)
SVG_SET_OPTIONS (SVG_Get_options ?-4) `Change to relative coordinates
SVG_PATH_LINE_TO (objectRef;50;-25)
```



```

For ($Lon_i;1;4;1)
  SVG_PATH_ARC (objectRef; 25;25*$Lon_i; 50;-25;-30)
  SVG_PATH_LINE_TO (objectRef;50;-25)
End for

```

#### 4. Complex path (cubic Bezier curve):



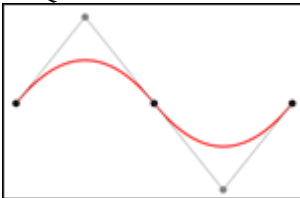
```

`Create a new SVG tree
$txt_svg:=SVG_New (174,96; 125,04; "4D Logo"; ""; True)

`Create a new path
$txt_path:=SVG_New_path ($txt_svg;150,665;13,021)
`Set colors
SVG_SET_STROKE_BRUSH ($txt_path;"rgb(33,42,111)")
SVG_SET_FILL_BRUSH ($txt_path;"rgb(33,42,111)")
...
SVG_PATH_CURVE ($txt_path;-9,683;-6,54;-20,842;-8,888;-33,06;-10,462)
SVG_PATH_CURVE ($txt_path;-7,042;-0,915;-14,587;-0,877;-22,087;-0,877)
SVG_PATH_CURVE ($txt_path;-1,725;0;-4,312;-0,405;-5,761;0,24)
SVG_PATH_CURVE ($txt_path;-1,762;0;-5,092;-0,382;-6,479;0,24)
...
SVG_PATH_CURVE ($txt_path;181,489;70,216;177,236;30,976;150,665;13,021)
SVG_PATH_MOVE_TO ($txt_path;146,03;98,078)
...
SVG_PATH_CURVE ($txt_path;153,11;78,668;151,407;89,558;146,03;98,078)

```

#### 5. Quadratic Bezier curve:



```

`Create a new SVG tree
$txt_svg:=SVG_New

```

```
`Reset stroke to black and set fill to none
SVG_SET_DEFAULT_BRUSHES ("","none")
```

```
`Draw a quadratic Bezier curve in red
$qCurve:=SVG_New_path ($svg;200;300)
SVG_SET_STROKE_BRUSH ($qCurve;"red")
SVG_SET_STROKE_WIDTH ($qCurve;5)
SVG_PATH_QCURVE ($qCurve;400;50;600;300)
SVG_PATH_QCURVE ($qCurve;1000;300)
```

```
`End points in black
$g:=SVG_New_group ($svg)
SVG_Set_description ($g;"End points")
SVG_SET_DEFAULT_BRUSHES ("black","black")
SVG_New_circle ($g;200;300;10)
SVG_New_circle ($g;600;300;10)
SVG_New_circle ($g;1000;300;10)
```

```
`Control points and lines from end points to control points in gray
$g:=SVG_New_group ($svg)
SVG_Set_description ($g;"Control points and lines from end points to control points")
SVG_SET_DEFAULT_BRUSHES (SVG_Color_grey (50);"none")
$path:=SVG_New_path ($svg;200;300)
SVG_SET_STROKE_WIDTH ($path;2)
SVG_PATH_LINE_TO ($path;400;50;600;300;800;550;1000;300)
$gray:= SVG_Color_grey (50)`grey 50%
SVG_SET_DEFAULT_BRUSHES ($gray;$gray)
SVG_New_circle ($g;400;50;10)
SVG_New_circle ($g;800;550;10)
```

### See Also

SVG\_PATH\_CLOSE, SVG\_PATH\_LINE\_TO.

SVG\_New\_polygon (parentSVGObject{; points{; foregroundColor{; backgroundColor{; strokeWidth}}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
points	String	→	Path
foregroundColor	String	→	Color or gradient name
backgroundColor	String	→	Color or gradient name
strokeWidth	Real	→	Line thickness
Function result	SVG_Ref	←	Reference of polygon

**Description**

The SVG\_New\_polygon command creates a new closed form in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not a valid reference, an error is generated.

The optional points parameter can be used to pass the path points of the polygon as expected by the SVG standard. If this parameter is omitted or empty, the points may be set using the SVG\_ADD\_POINT command.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the "Colors and gradients" section).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.

**See Also**

SVG\_New\_polygon\_by\_arrays, SVG\_New\_rect, SVG\_New\_regular\_polygon.

SVG\_New\_polygon\_by\_arrays (parentSVGObject; xArrayPointer; yArrayPointer{; foregroundColor{; backgroundColor{; strokeWidth}}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
xArrayPointer	Pointer	→	Coordinates on X axis of points
yArrayPointer	Pointer	→	Coordinates on Y axis of points
foregroundColor	String	→	Color or gradient name
backgroundColor	String	→	Color or gradient name
strokeWidth	Real	→	Line thickness
Function result	SVG_Ref	←	Reference of polygon

### Description

The SVG\_New\_polygon\_by\_arrays command draws a closed form consisting of a set of straight connected segments in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

All the coordinate values are in the user coordinate system.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.

## Example

Draw a star (default border color and line thickness):



```
ARRAY LONGINT($tX;0)
ARRAY LONGINT($tY;0)
```

```
APPEND TO ARRAY($tX;129)
APPEND TO ARRAY($tY;10)
APPEND TO ARRAY($tX;158)
APPEND TO ARRAY($tY;96)
APPEND TO ARRAY($tX;248)
APPEND TO ARRAY($tY;96)
APPEND TO ARRAY($tX;176)
APPEND TO ARRAY($tY;150)
APPEND TO ARRAY($tX;202)
APPEND TO ARRAY($tY;236)
APPEND TO ARRAY($tX;129)
APPEND TO ARRAY($tY;185)
APPEND TO ARRAY($tX;56)
APPEND TO ARRAY($tY;236)
APPEND TO ARRAY($tX;82)
APPEND TO ARRAY($tY;150)
APPEND TO ARRAY($tX;10)
APPEND TO ARRAY($tY;96)
APPEND TO ARRAY($tX;100)
APPEND TO ARRAY($tY;96)
```

```
objectRef:=SVG_New_polygon_by_arrays (svgRef;->$tX;->$tY)
```

## See Also

SVG\_New\_polygon, SVG\_New\_regular\_polygon.

SVG\_New\_polyline (parentSVGObject{; points{; foregroundColor{; backgroundColor{; strokeWidth}}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
points	String	→	Path
foregroundColor	String	→	Color or gradient name
backgroundColor	String	→	Color or gradient name
strokeWidth	Real	→	Line thickness
Function result	SVG_Ref	←	Reference of line

### Description

The SVG\_New\_polyline command creates a new open broken line in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not a valid reference, an error is generated.

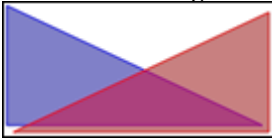
The optional points parameter can be used to pass the path points of the line as expected by the SVG standard. If this parameter is omitted or empty, the points may be set with the SVG\_ADD\_POINT command.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.

## Example

Draw two triangles:



```
$polyline:=SVG_New_polyline ($svg;"10,10 200,100 10,100 10,10";"blue";"blue:50")
$polyline:=SVG_New_polyline ($svg;"";"red";"red:50")
SVG_ADD_POINT ($polyline;205;15)
SVG_ADD_POINT ($polyline;15;105)
SVG_ADD_POINT ($polyline;205;105)
SVG_ADD_POINT ($polyline;205;15)
```

## See Also

SVG\_New\_polyline\_by\_arrays.

SVG\_New\_polyline\_by\_arrays (parentSVGObject; xArrayPointer; yArrayPointer; foregroundColor; backgroundColor; strokeWidth}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
xArrayPointer	Pointer	→	Coordinates on X axis of points
yArrayPointer	Pointer	→	Coordinates on Y axis of points
foregroundColor	String	→	Color or gradient name
backgroundColor	String	→	Color or gradient name
strokeWidth	Real	→	Line thickness
Function result	SVG_Ref	←	Reference of line

### Description

The SVG\_New\_polyline\_by\_arrays command draws a broken line composed of straight segments connected together in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

Usually, 'polyline' elements design open forms but they can be used for closed forms as well. In this case the last point must be set as equal to the first.

All the coordinate values are in the user coordinate system.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.



## Examples

1. Draw a triangle (default border color and line thickness):



```
ARRAY LONGINT($tX;0)  
ARRAY LONGINT($tY;0)
```

```
APPEND TO ARRAY($tX;10)  
APPEND TO ARRAY($tY;10)  
APPEND TO ARRAY($tX;200)  
APPEND TO ARRAY($tY;100)  
APPEND TO ARRAY($tX;10)  
APPEND TO ARRAY($tY;100)  
APPEND TO ARRAY($tX;10)  
APPEND TO ARRAY($tY;10)
```

```
svgRef:= SVG_New
```

```
objectRef:=SVG_New_polyline_by_arrays (svgRef;->$tX;->$tY)
```

2. Draw a line diagram:



```
ARRAY LONGINT($tX;0)  
ARRAY LONGINT($tY;0)  
  `X axis  
For ($Lon_i;0;200;20)  
  APPEND TO ARRAY($tX;$Lon_i)  
End for
```

Values  
APPEND TO ARRAY(\$tY;100)  
APPEND TO ARRAY(\$tY;100)  
APPEND TO ARRAY(\$tY;30)  
APPEND TO ARRAY(\$tY;30)  
APPEND TO ARRAY(\$tY;80)  
APPEND TO ARRAY(\$tY;60)  
APPEND TO ARRAY(\$tY;10)  
APPEND TO ARRAY(\$tY;40)  
APPEND TO ARRAY(\$tY;50)  
APPEND TO ARRAY(\$tY;70)

objectRef:= SVG\_New\_polyline\_by\_arrays (svgRef;->\$tX;->\$tY;"crimson";"none";5)

### See Also

SVG\_New\_polyline.

SVG\_New\_rect (parentSVGObject; x; y; width; height{; roundedX{; roundedY{; foregroundColor{; backgroundColor{; strokeWidth}}}})) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
x	Number	→	X of upper left corner
y	Number	→	Y of upper left corner
width	Number	→	Width of rectangle
height	Number	→	Height of rectangle
roundedX	Number	→	Horizontal curve
roundedY	Number	→	Vertical curve
foregroundColor	String	→	Color or gradient name
backgroundColor	String	→	Color or gradient name
strokeWidth	Real	→	Line thickness
Function result	SVG_Ref	←	Reference of rectangle

### Description

The SVG\_New\_rect command creates a new rectangle in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

The rectangle is positioned and sized according to the values of x, y, width and height.

The optional roundedX and roundedY parameters can be used to round off the angles according to the indicated values. If the roundedY parameter is omitted (or is -1), the curve will be regular. Pass -1 in these parameters if you want them to be ignored by the command.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.

## Examples

1. Draw a rectangle (default fill and border color, default line thickness):



```
svgRef:= SVG_New  
objectRef:=SVG_New_rect (svgRef;10;10;200;100)
```

2. Draw a blue rectangle with a 3-pixel red border:



```
svgRef:= SVG_New  
objectRef:=SVG_New_rect (svgRef;10;10;200;100;0;0;"red";"blue";3)
```

3. Draw a square with rounded edges (default fill and border color, default line thickness):



```
svgRef:= SVG_New  
objectRef:=SVG_New_rect (svgRef;10;10;100;100;20)
```

4. Draw a light blue rectangle with rounded ends and a blue edge (default line thickness):



```
svgRef:= SVG_New  
objectRef:=SVG_New_rect (svgRef;10;10;200;100;-1;50;"blue";"lightblue")
```

## See Also

SVG\_New\_polygon, SVG\_SET\_ROUNDING\_RECT.

SVG\_New\_regular\_polygon (parentSVGObject; width; number{; x{; y{; foregroundColor{; backgroundColor{; strokeWidth}}}})) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
width	Number	→	Diameter of surrounding circle
number	Number	→	Number of sides
x	Number	→	Coordinate on center X axis
y	Number	→	Coordinate on center Y axis
foregroundColor	String	→	Color or gradient name
backgroundColor	String	→	Color or gradient name
strokeWidth	Real	→	Line thickness
Function result	SVG_Ref	←	Reference of polygon

### Description

The SVG\_New\_regular\_polygon command draws a regular polygon with number of sides fit into a circle with a diameter of width in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

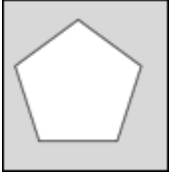
The optional x and y parameters can be used to specify the center of the circle. If they are omitted, the figure will be drawn in the upper left corner of the document.

The optional foregroundColor and backgroundColor parameters contain, respectively, the name of the line color and of the background color. (For more information about colors, please refer to the commands of the Colors and Gradients theme).

The optional strokeWidth parameter contains the size of the pen expressed in pixels. Its default value is 1.

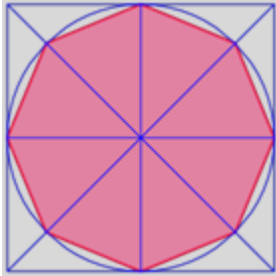
## Examples

1. Draw a pentagon (default fill and border color, default line thickness):



```
svgRef:= SVG_New  
objectRef:= SVG_New_regular_polygon (svgRef;100;5)
```

2. Draw an octagon, the circle containing it and the trace lines:



```
svgRef:= SVG_New  
$width:=200  
$sides:=8  
objectRef:= SVG_New_regular_polygon (svgRef;$width;$sides;0;0;"crimson";  
"palevioletred";2)  
  
$radius:=$width/2  
objectRef:=SVG_New_rect (svgRef;0;0;$width;$width;0;0;"blue";"none")  
objectRef:=SVG_New_line (svgRef;0;$radius;$width;$radius;"blue")  
objectRef:=SVG_New_line (svgRef;$radius;0;$radius;$width;"blue")  
objectRef:=SVG_New_line (svgRef;0;0;$width;$width;"blue")  
objectRef:=SVG_New_line (svgRef;$width;0;0;$width;"blue")  
objectRef:=SVG_New_circle (svgRef;$radius;$radius;$radius;"blue";"none")
```

## See Also

SVG\_New\_polygon.

SVG\_PATH\_ARC (parentSVGObject; xRadius; yRadius; x; y{; rotation{; arcpath}})

Parameter	Type	Description
parentSVGObject	SVG_Ref	→ Reference of path element
xRadius	Number	→ Radius of ellipse on X axis
yRadius	Number	→ Radius of ellipse on Y axis
x	Number	→ Coordinate on X axis of destination point
y	Number	→ Coordinate on Y axis of destination point
rotation	Number	→ Value of rotation
arcpath	Longint	→ Sets the way the arc will be drawn

### Description

The SVG\_PATH\_ARC command draws an elliptical arc, from the current point to the point (x, y), at the end of the path referenced by parentSVGObject. If parentSVGObject is not a path reference ('path' element), an error is generated.

The size and orientation of the ellipse are set by two radii (xRadius, yRadius) and a rotation value on the X axis that indicates the rotation of the ellipse as a whole with respect to the current coordinate system.

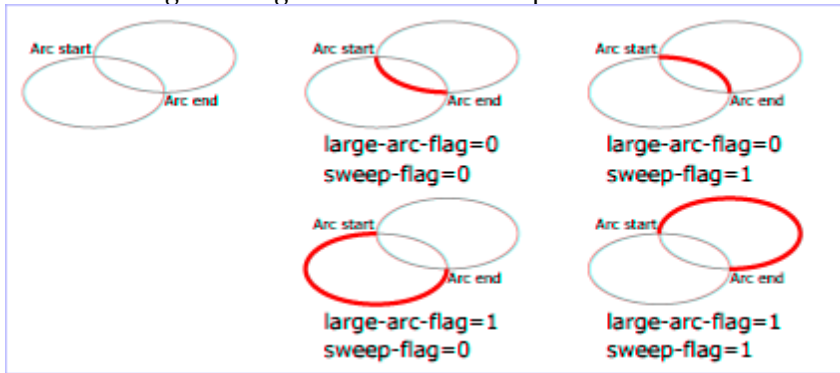
The optional arcpath parameter can be used to apply a combination of constraints which will determine how the arc will be drawn. The large-arc-flag constraint is used to choose (or not) the larger of the two possible arcs (greater than 180°) and the sweep-flag constraint chooses the direction it will be drawn (positive angle or negative angle).

The following values, representing the four possible combinations of the two constraints, can be passed:

- 0: large-arc-flag = 0, sweep-flag = 1
- 1: large-arc-flag = 1, sweep-flag = 0
- 2: large-arc-flag = 0, sweep-flag = 0
- 3: large-arc-flag = 1, sweep-flag = 1

When large-arc-flag is equal to 1, the larger arc is drawn (and the smaller when it is equal to 0). When sweep-flag is equal to 1, the arc is drawn at a positive angle (and at a negative angle when it is equal to 0).

The following drawing illustrates the four possible combinations:



By default, the value of `arcpath` is 0 (`large-arc-flag=0`, `sweep-flag=1`).

### Example

See the examples for the `SVG_New_path` command.

### See Also

`SVG_New_arc`.



SVG\_PATH\_CLOSE (parentSVGObject)

<b>Parameter</b>	<b>Type</b>	<b>Description</b>
parentSVGObject	SVG_Ref →	Reference of path

**Description**

The SVG\_PATH\_CLOSE command closes the current subpath referenced by parentSVGObject by drawing a straight line from the current point to the initial point. If parentSVGObject is not a path reference ('path' element), an error is generated.

**Example**

See the examples for the SVG\_New\_path command.

**See Also**

SVG\_New\_path.

SVG\_PATH\_CURVE (parentSVGObject{; controlStartX; controlStartY; }controlEndX; controlEndY; x; y)

Parameter	Type	Description
parentSVGObject	SVG_Ref	→ Reference of parent element
controlStartX	Number	→ Coordinate on X axis of control point
controlStartY	Number	→ Coordinate on Y axis of control point
controlEndX	Number	→ Coordinate on X axis of control point
controlEndY	Number	→ Coordinate on Y axis of control point
x	Number	→ Coordinate on X axis of destination point
y	Number	→ Coordinate on Y axis of destination point

### Description

The SVG\_PATH\_CURVE command adds a cubic Bezier curve to the path referenced by parentSVGObject starting from the current point to the point whose coordinates are passed (x, y). If parentSVGObject is not a path reference ('path' element), an error is generated.

The optional controlStartX and controlStartY parameters can be used to specify the position of the control point at the start of the curve. If they are omitted, the first control point is supposed to be the reflection of the second control point of the previous command with respect to the current point.

The controlEndX and controlEndY parameters are used to specify the position of the control point at the end of the curve.

### Example

See the examples for the SVG\_New\_path command.

### See Also

SVG\_New\_path, SVG\_PATH\_QCURVE.

SVG\_PATH\_LINE\_TO (parentSVGObject; x{; y}{; x2; y2; ...; xN; yN})

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
x	Number	→	Coordinate on X axis of new point(s)
y	Number	→	Coordinate on Y axis of new point(s)

### Description

The SVG\_PATH\_LINE\_TO command adds one or more straight segments to the path referred by parentSVGObject. If parentSVGObject is not a path reference ('path' element), an error is generated.

The x and y parameters can be used to specify the start position of the path in the SVG container.

- If only the x parameter is provided, the line will be drawn horizontally from the current point (xc, yc) to the point (x, yc).
- If both x and y are passed, a line will be drawn from the current point (xc, yc) to the point (x, y).
- If several pairs of coordinates are passed, the different points will be added successively. In this case, if the last pair of coordinates is incomplete (missing y), it will be ignored.

### Example

See the examples for the SVG\_New\_path command

### See Also

SVG\_New\_path.

SVG\_PATH\_MOVE\_TO (parentSVGObject; x; y)

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of path
x	Number	→	Coordinate on X axis
y	Number	→	Coordinate on Y axis

### Description

The SVG\_PATH\_MOVE\_TO command begins a new subpath at the point of the given coordinates (x, y) in the path referenced by parentSVGObject. If parentSVGObject is not a path reference ('path' element), an error is generated.

The effect produced is as if the "pen" were lifted and moved to a new location. The current point becomes the new starting point which will be taken into account by the SVG\_PATH\_CLOSE command.

### Example

See the examples for the SVG\_New\_path command.

### See Also

SVG\_New\_path.

---

SVG\_PATH\_QCURVE (parentSVGObject{; controlX; controlY; }x; y)

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
controlX	Number	→	Coordinate on X axis of control point
controlY	Number	→	Coordinate on Y axis of control point
x	Number	→	Coordinate on X axis of destination point
y	Number	→	Coordinate on Y axis of destination point

### Description

The SVG\_PATH\_CURVE command adds a quadratic Bezier curve from the current point to the point of the coordinates (x, y) to the line referenced by parentSVGObject. If parentSVGObject is not a path reference ('path' element), an error is generated.

The optional controlX and controlY parameters can be used to specify the position of the control point at the beginning of the curve. If they are omitted, the first control point is supposed to be a reflection of the second control point of the previous command with respect to the current point.

### Example

See the examples of the SVG\_New\_path command.

### See Also

SVG\_PATH\_CURVE.

SVG\_Use (parentRef; id; x; y; width; height; mode)) → SVG\_Ref

Parameter	Type		Description
parentSVGObjec	SVG_Ref	→	Reference of parent element
id	String	→	Name of symbol
x	Number	→	X position of viewBox
y	Number	→	Y position of viewBox
width	Number	→	Width of viewBox
height	Number	→	Height of viewBox
mode	String	→	Adjustment to viewBox
Function result	SVG_Ref	←	SVG object reference

### Description

The SVG\_Use command places an occurrence of the symbol in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document or if id is not the object name of an SVG document, an error is generated.

A symbol is used to specify graphic objects; it is never rendered directly but may be instantiated using the SVG\_Use command.

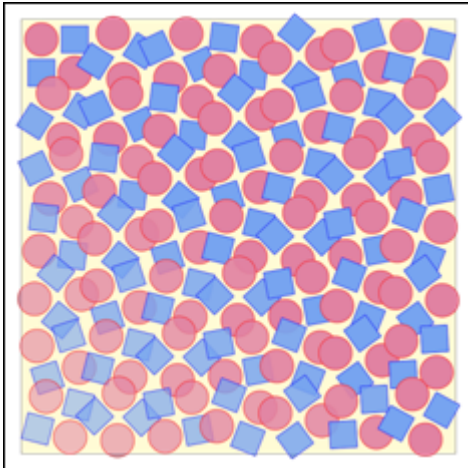
The id parameter specifies the name of the symbol.

The optional x, y, width and height parameters specify the viewBox rectangle ('viewBox' attribute).

The optional mode parameter can be used to indicate if the graphic must be fitted, and how so, to the size of the viewBox. (see the SVG\_New command).

### Example

Specify a graphic composed of two red circles and two blue squares. Then use this graphic in a loop to create 36 occurrences with varying positions, opacity and rotation of the original graphic.



```

$SVG:=SVG_New
  `Draw a background
SVG_New_rect ($SVG;20;20;650;650;0;0;"gray";"lemonchiffon")
  `Specify a symbol composed of 2 squares and 2 circles
$Symbol:=SVG_Define_symbol ($SVG;"MySymbol";0;0;110;110;"true")
SVG_New_circle ($Symbol;30;30;25;"red";"palevioletred")
SVG_New_rect ($Symbol;10;60;40;40;0;0;"blue";"cornflowerblue")
SVG_New_rect ($Symbol;60;10;40;40;0;0;"blue";"cornflowerblue")
SVG_New_circle ($Symbol;80;80;25;"red";"palevioletred")
  `In a group...
$g:=SVG_New_group ($SVG)
  `...positioned 20 units from the top left corner of the document...
SVG_SET_TRANSFORM_TRANSLATE ($g;20;20)
  `...place 36 patterns by varying the position, opacity and rotation
For ($x;0;540;90) `6 columns
  For ($y;0;540;90) `6 rows
    $use:=SVG_Use ($g;"MySymbol";$x;$y;110;110)
    SVG_SET_OPACITY ($use;100-($y/12)+($x/12))
    SVG_SET_TRANSFORM_ROTATE ($use;($x*(18/50))+($y*(18/50));($x+55);
      ($y+55))
  End for
End for

```

#### See Also

SVG\_Define\_symbol.





# 6

---

## Filters



The commands of the "Filters" theme can be used to set filter effects that are applicable to the SVG elements. A filter effect consists of a succession of graphic operations, applied to a source graphic, which produces a modified graphic.

The result of the filter effect is rendered on the target device in place of the original source graphic.

A filter is set using the `SVG_Define_filter` command, found in the "Structure and Definitions" theme, and is applied using the `SVG_SET_FILTER` command, found in the "Attributes" theme. The commands of the "Filters" theme are used to construct filtering operations or "filter primitives".

---

SVG\_Filter\_Blend (filterRef; picture; backgroundPict{; mode{; name{}}) → SVG\_Ref

Parameter	Type		Description
filterRef	SVG_Ref	→	Reference of filter
picture	String	→	Picture source
backgroundPict	String	→	Background picture source
mode	String	→	Mixing mode
name	String	→	Target of filter primitive
Function result	SVG_Ref	←	Reference of primitive

### Description

The SVG\_Filter\_Blend command sets a blend filter for the filterRef filter and returns its reference. If filterRef is not a filter reference, an error is generated.

This filter is made up of two sources, backgroundPict and picture, with the help of the mixing modes currently used by the imaging software.

The optional mode parameter can be used to set the combination mode of the pixels used for the blend (see the specification). Its value must be: "normal" (default value), "multiply", "screen", "darken" or "lighten".

The optional name parameter is the name, if any, assigned to the result of this filter primitive.

### See Also

SVG\_Filter\_Blur, SVG\_Filter\_Offset.

---

SVG\_Filter\_Blur (filterRef; deviation{; input{; name}) → SVG\_Ref

Parameter	Type		Description
filterRef	SVG_Ref	→	Reference of filter
deviation	Number	→	Standard deviation for blur operation
input	String	→	Source of filter primitive
name	String	→	Target of filter primitive
Function result	SVG_Ref	←	Reference of primitive

### Description

The SVG\_Filter\_Blur command sets a Gaussian blur for the filterRef filter and returns its reference. If filterRef is not a filter reference, an error is generated.

The deviation parameter can be used to set the standard deviation for the blur operation. If the number is an integer, the same deviation will be applied to the X and Y axes. If the number includes a decimal part, the integer part represents the deviation to be applied to the X axis and the decimal part represents the deviation to be applied to the Y axis.

The optional input parameter identifies the graphic source of the filter primitive.

The optional name parameter is the name, if any, assigned to the result of this filter primitive.

### See Also

SVG\_Filter\_Blend, SVG\_Filter\_Offset.

SVG\_Filter\_Offset (filterRef; dx{; dy{; input{; name}}}) → SVG\_Ref

Parameter	Type		Description
filterRef	SVG_Ref	→	Reference of filter
dx	Number	→	Offset on X axis
dy	Number	→	Offset on Y axis
input	String	→	Source of filter primitive
name	String	→	Target of filter primitive
Function result	SVG_Ref	←	Reference of primitive

### Description

The SVG\_Filter\_Offset command sets an offset for the filterRef filter and returns its reference. If filterRef is not a filter reference, an error is generated.

The dx parameter is the value of the horizontal offset.

The optional dy parameter is the value of the vertical offset.

The optional input parameter identifies the graphic source of the filter primitive.

The optional name is the name, if any, assigned to the result of this filter primitive.

### See Also

SVG\_Filter\_Blend, SVG\_Filter\_Blur.

# 7

---

## Structure and Definitions





SVG\_Define\_filter (parentRef; id{; frameX; frameY{; frameWidth; frameHeight{; frameUnit{; filterUnit}}})) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
id	String	→	Name of symbol
frameX	Number	→	Coordinate on X axis
frameY	Number	→	Coordinate on Y axis
frameWidth	Number	→	Width of target rectangle
frameHeight	Number	→	Height of target rectangle
frameUnit	String	→	Coordinate system of frame
filterUnit	String	→	Filter system of values
Function result	SVG_Ref	←	Reference of filter

### Description

The SVG\_Define\_filter command sets a new filter in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

A filter is a succession of graphic operations that will be applied to the target element. The filter element is never rendered directly; it will be applied to an object using the SVG\_SET\_FILTER command.

The id parameter specifies the name of the marker. The name will be used to associate a filter with an object. If an element with the same name exists, it will be replaced.

The optional frameX, frameY, frameWidth and frameHeight parameters set a rectangular region in the document to which this filter will be applied.

The optional frameUnit parameter sets the coordinate system for the 4 previous parameters. Expected values: "userSpaceOnUse" or "objectBoundingBox" (default value).

The optional `filterUnit` parameter sets the coordinate system for the lengths and the filter definition properties. Expected values: "userSpaceOnUse" (default value) or "objectBoundingBox".

**See Also**

SVG\_Filter\_Blend, SVG\_Filter\_Blur, SVG\_Filter\_Offset, SVG\_SET\_FILTER.

SVG\_Define\_linear\_gradient (parentSVGObject; id; startColor; endColor{; rotation}) → String

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
id	String	→	Name of gradient
startColor	String	→	Start color
endColor	String	→	End color
rotation	Integer	→	Rotation of gradient vector
Function result	String	←	Reference of gradient

### Description

The `SVG_Define_linear_gradient` command sets a new linear gradient in the SVG container designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG document, an error is generated.

A gradient consists in a continuous progressive color transition from one color to another along a vector. Once specified, gradients are called on a given graphic element, while indicating whether this element must be filled or edged with the gradient called.

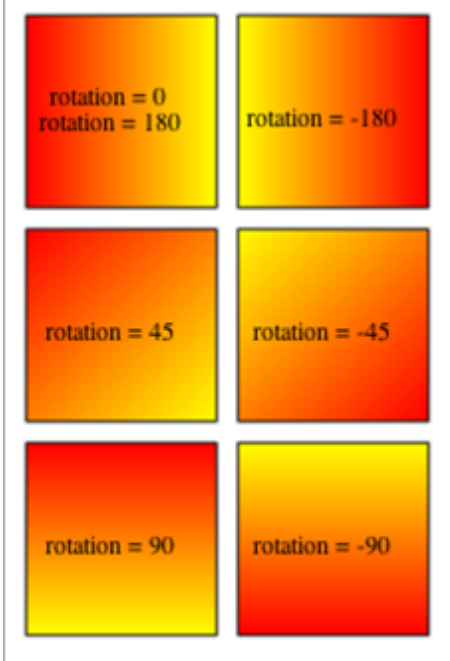
The `id` parameter specifies the name of the gradient. If an element with the same name exists, it will be replaced. This is the name that will be used to call the gradient each time a that a color expression is expected by using the syntax "url(#ID)".

The `startColor` and `endColor` parameters specify the colors used to begin and end the gradient.

The optional `rotation` parameter sets the position and direction of the gradient vector (see example).

## Example

Draw 6 solid squares where each uses a linear gradient paint server while varying the rotation and direction of the gradient vector:



```
$svg:=SVG_New
```

```
SVG_Define_linear_gradient ($svg;"demoGradient_1";"red";"yellow")  
SVG_New_rect ($svg;10;10;90;90;0;0;"black";"url(#demoGradient_1)")  
SVG_New_text ($svg;"rotation = 0\rrotation = 180";50;40;"";-1;-1;Center)
```

```
SVG_Define_linear_gradient ($svg;"demoGradient_2";"red";"yellow";180)  
SVG_New_rect ($svg;110;10;90;90;0;0;"black";"url(#demoGradient_2)")  
SVG_New_text ($svg;"rotation = -180";150;50;"";-1;-1; Center)
```

```
SVG_Define_linear_gradient ($svg;"demoGradient_3";"red";"yellow";45)  
SVG_New_rect ($svg;10;110;90;90;0;0;"black";"url(#demoGradient_3)")  
SVG_New_text ($svg;"rotation = 45";50;150;"";-1;-1; Center)
```

```
SVG_Define_linear_gradient ($svg;"demoGradient_4";"red";"yellow";-45)
SVG_New_rect ($svg;110;110;90;90;0;0;"black";"url(#demoGradient_4)")
SVG_New_text ($svg;"rotation = -45";150;150;"";-1;-1; Center)
```

```
SVG_Define_linear_gradient ($svg;"demoGradient_5";"red";"yellow";90)
SVG_New_rect ($svg;10;210;90;90;0;0;"black";"url(#demoGradient_5)")
SVG_New_text ($svg;"rotation = 90";50;250;"";-1;-1; Center)
```

```
SVG_Define_linear_gradient ($svg;"demoGradient_6";"red";"yellow";-90)
SVG_New_rect ($svg;110;210;90;90;0;0;"black";"url(#demoGradient_6)")
SVG_New_text ($svg;"rotation = -90";150;250;"";-1;-1; Center)
```

```
`Save document
SVG_SAVE_AS_TEXT ($svg;"test.svg")
`Free up memory
SVG_CLEAR ($svg)
```

### See Also

SVG Colors, SVG\_Define\_radial\_gradient.

SVG\_Define\_marker (parentRef; id{; x; y{; width; height{; orientation}})) → ObjectReference

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
id	String	→	Name of symbol
x	Number	→	Coordinate on X axis of reference point
y	Number	→	Coordinate on Y axis of reference point
width	Number	→	Width of marker
height	Number	→	Height of marker
orientation	Number	→	Orientation of marker
Function result	SVG_Ref	←	Reference of marker

### Description

The `SVG_Define_marker` command creates a marker in the SVG container designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG document, an error is generated.

A marker object is used to draw an arrow or multiple markers (the points of a curve for example). A marker will be attached to an SVG element with the `SVG_SET_MARKER` command.

The `id` parameter specifies the name of the marker. The name will be used to associate a marker with an object. If an element with the same name exists, it will be replaced.

The optional `x` and `y` parameters specify the coordinates of the reference point that must line up exactly with the marker position

The optional `width` and `height` parameters specify the width and height of the rendered rectangle.

The optional `orientation` parameter can be used to adjust the orientation of the marker. A value included between 0 and 360 represents the angle between the X axis of the marker and that of the user space. If this parameter is omitted or if its value does not fall in the 0 - 360 interval, the placement will be calculated automatically by the rendering engine according to the object.

**Example**

Refer to the example of the SVG\_SET\_MARKER command.

**See Also**

SVG\_SET\_MARKER.

SVG\_Define\_radial\_gradient (parentRef; id; startColor; endColor{; cx; cy; r{; fx; fy})) → String

Parameter	Type	Description
parentSVGObject	SVG_Ref	→ Reference of parent element
id	String	→ Name of gradient
startColor	String	→ Start color
endColor	String	→ End color
cx	Integer	→ Coordinate on X axis of center of endColor
cy	Integer	→ Coordinate on Y axis of center of endColor
r	Integer	→ Radius of endColor
fx	Integer	→ Coordinate on X axis of center of startColor
fy	Integer	→ Coordinate on Y axis of center of startColor
Function result	String	← Reference of gradient

### Description

The SVG\_Define\_radial\_gradient command sets a new radial gradient in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

A gradient consists in a continuous progressive color transition from one color to another along a vector. Once specified, gradients are called on a given graphic element, while indicating whether this element must be filled or edged with the gradient called.

The id parameter specifies the name of the gradient. If an element with the same name exists, it will be replaced. This is the name that will be used to call the gradient each time a that a color expression is expected by using the syntax "url(#ID)".

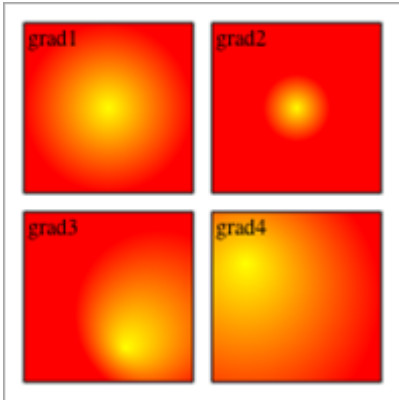
The startColor and endColor parameters specify the colors used to begin and end the gradient.

The optional cx, cy and r parameters specify, in percent, the external border circle of the endColor of the gradient. Their values must be included between 0 and 100.

The optional fx and fy parameters specify, in percent, the focus point of the gradient. The startColor begins at the point [fx,fy]. Their values must be included between 0 and 100. If these arguments are omitted, this point coincides with [cx,cy].



## Example



```
$svg:=SVG_New
```

```
SVG_Define_radial_gradient ($svg;"grad1";"yellow";"red")  
SVG_New_rect ($svg;10;10;90;90;0;0;"black";"url(#grad1)")  
SVG_New_text ($svg;"grad1";12;10)
```

```
SVG_Define_radial_gradient ($svg;"grad2";"yellow";"red";50;50;20;50;50)  
SVG_New_rect ($svg;110;10;90;90;0;0;"black";"url(#grad2)")  
SVG_New_text ($svg;"grad2";112;10)
```

```
SVG_Define_radial_gradient ($svg;"grad3";"yellow";"red";80;60;50;60;80)  
SVG_New_rect ($svg;10;110;90;90;0;0;"black";"url(#grad3)")  
SVG_New_text ($svg;"grad3";12;110)
```

```
SVG_Define_radial_gradient ($svg;"grad4";"yellow";"red";20;50;80;20;30)  
SVG_New_rect ($svg;110;110;90;90;0;0;"black";"url(#grad4)")  
SVG_New_text ($svg;"grad4";112;110)
```

```
  `Save document  
SVG_SAVE_AS_TEXT ($svg;"test.svg")  
  `Free up memory  
SVG_CLEAR ($svg)
```

## See Also

SVG Colors, SVG\_Define\_linear\_gradient.

---

SVG\_Define\_shadow (parentSVGObject; id{; deviation{; offsetX{; offsetY{}}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
id	String	→	Name of filter
deviation	Number	→	Value of shadow dispersion
offsetX	Number	→	Offset on X axis
offsetY	Number	→	Offset on Y axis
Function result	SVG_Ref	←	Reference of filter

### Description

The `SVG_Define_shadow` command sets a new shadow filter in the SVG container designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG document, an error is generated.

This filter will be applied to objects for which a shadow is desired using the `SVG_SET_FILTER` command.

The `id` parameter specifies the name of the filter. This name will be used to associate a filter with an object. If an element with the same name exists, it will be replaced.

The optional `deviation` parameter sets the intensity of the shadow dispersion. Default value: 4.

The optional `offsetX` and `offsetY` parameters specify, respectively, the horizontal and vertical offset of the shadow with respect to the object. Default value: 4.

## Example

Declaration of a filter that can be used to make a shadow beneath an object:



```
$svg:=SVG_New
```

```
$text:=SVG_New_text ($svg;"SVG";52;76-45;"Verdana";45)
```

```
SVG_SET_FONT_COLOR ($text;"red")
```

```
`Set filter
```

```
SVG_Define_shadow ($svg;"myShadow")
```

```
`and apply it to text
```

```
SVG_SET_FILTER ($text;"myShadow")
```

## See Also

SVG\_SET\_FILTER.

SVG\_Define\_solidColor (parentSVGObject; id; color{; opacity}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
id	String	→	Color name
color	String	→	Color expression
opacity	Longint	→	Opacity
Function result	SVG_Ref	←	Color reference

### Description

The `SVG_Define_solidColor` command sets a new custom color in the SVG container designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG document, an error is generated.

The `id` parameter specifies the color name. The name will be used to associate a color with an object. If an element with the same name exists, it will be replaced.

The `color` parameter is a color expression recognized by SVG (see [Colors and Gradients](#)).

The optional `opacity` parameter can be used to specify an opacity (from 0 to 100) for this color. If the parameter is omitted, the opacity will be 100%.

The color set in this way will be associated with the fill or stroke paint by passing the string `"url(#ID)"` as the value when a color expression is expected.

### Example

```

`Set blue to 50%
SVG_Define_solidColor($svg; "MyColor"; "blue"; 50)
...
SVG_New_rect ($svg; 0; 0; 20; 20; 0; 0; "url(#MyColor)" ; "url(#MyColor)")
...
$line:=SVG_New_line (10; 10 ;100 ;100)
SVG_SET_STROKE_BRUSH ($line; "url(#MyColor)")

```

### See Also

SVG Colors.

SVG\_Define\_symbol (parentSVGObject; id{; x{; y{; width{; height{; mode}}}})) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
id	String	→	Name of symbol
x	Number	→	X position of viewBox
y	Number	→	Y position of viewBox
width	Number	→	Width of viewBox
height	Number	→	Height of viewBox
mode	String	→	Adjustment to viewBox
Function result	SVG_Ref	←	Reference of symbol

### Description

The SVG\_Define\_symbol command creates a symbol in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

A symbol object is used to specify graphic objects that may be instantiated using the SVG\_Use command.

The id parameter specifies the name of the symbol.

The optional x, y, width and height parameters specify the viewBox rectangle ('viewBox' attribute).

The optional mode parameter can be used to indicate if the graphic must be fitted, and how so, to the size of the viewBox. For more information about this point, please refer to the description of the SVG\_New command.

### Example

Refer to the description of the SVG\_Use command.

### See Also

SVG\_Use.

---

SVG\_New\_group (parentSVGObject{; id{; url{; target{}}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
id	String	→	Name of group
url	String	→	External link
target	String	→	Target of link
Function result	SVG_Ref	←	Reference of group

### Description

The `SVG_New_group` command creates a group in the SVG container designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG document, an error is generated.

The group ('g' element) can be used to group together several linked graphic elements, which will inherit the properties of the group.

The optional `id` parameter can be used to assign a name to the group. Named groups are necessary for several purposes such as animation and reusable objects.

The optional `url` parameter can be used to associate an external link. Group objects are then clickable (similar to the 'a' element of HTML).

The optional `target` parameter specifies the name of the target where the document will open when the link is activated. The values expected are those of the HTML specification to which are added the 'new' value for opening a new window and the 'none' value which is equivalent to not processing this attribute.

**Note:** External links are ignored when the SVG is displayed in a picture object (variable or field) of a 4D form. Management of external references is handled by the rendering engine. Under these conditions, the result may depend on the platform and the viewing software.

## Examples

1. A group of lines, all the same color:



```
$SVG:=SVG_New
$group:=SVG_New_group ($SVG)
  `Assign a color to the group elements
SVG_SET_STROKE_BRUSH ($group; "firebrick")
SVG_New_line ($group; 100; 300; 300; 100; ""; 5)
SVG_New_line ($group; 300; 300; 500; 100; ""; 10)
SVG_New_line ($group; 500; 300; 700; 100; ""; 15)
SVG_New_line ($group; 700; 300; 900; 100; ""; 20)
SVG_New_line ($group; 900; 300; 1100; 100; ""; 25)
```

2. Clickable text:



```
$SVG:=SVG_New
$group:=SVG_New_group ($SVG;"w3Link";"http://www.w3.org";"new")
SVG_New_text ($group;"www.w3.org";10;10;"arial";12;Underline ;Align Left ;"blue")
```

---

SVG\_Set\_description (parentSVGObject; description) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
description	String	→	Text of comments
Function result	SVG_Ref	←	Reference of description

### Description

The `SVG_Set_description` command sets a text for the SVG element designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG element, an error is generated.

A description is often used to insert a comment or explanatory text in the SVG code.

### Example

```
$SVG:=SVG_New
$g:=SVG_group ($SVG)
SVG_Set_title($g;"Company sales per region")
SVG_Set_description($g;"Bar diagram of company sales per region.")
...
```

### See Also

`SVG_Set_title`.



SVG\_Set\_title (parentSVGObject; title) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
title	String	→	Text of title
Function result	SVG_Ref	←	Reference of title

### Description

The `SVG_Set_title` command specifies a title for the SVG element designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG element, an error is generated.

A title is text data that is not included in the rendered picture but is use for structuring complex documents. Certain SVG rendering engines use the text of this element to display a help tip when the mouse moves over the object.

### Example

```
$SVG:=SVG_New
$rec:=SVG_New_rect ($SVG;20;20;650;650;0;0;"gray";"lemonchiffon")
SVG_Set_title($rec;"Background rectangle")
$Symbol:=SVG_Define_symbol ($SVG;"MySymbol";0;0;110;110;"true")
SVG_Set_title($Symbol;" Set a symbol composed of 2 squares and 2 circles ")
...
```

### See Also

SVG\_Set\_description.



# 8

---

# Text



SVG\_New\_text (parentSVGObject; text{; x{; y{; font{; size{; style{; alignment{; color{; rotation{; lineSpacing{; stretching}}}}}}}}) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
text	Text	→	Text to insert
x	Number	→	Coordinate on X axis
y	Number	→	Coordinate on Y axis
font	String	→	Font name
size	Integer	→	Size of characters in points
style	Integer	→	Style of characters
alignment	Integer	→	Alignment
color	String	→	Text color
rotation	Number	→	Angle of rotation of text
lineSpacing	Number	→	Line spacing in points
stretching	Number	→	Horizontal stretch factor
Function result	SVG_Ref	←	Reference of SVG text object

### Description

The SVG\_New\_text command inserts the text in text in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

The optional x and y parameters can be used to specify the position on the X and Y axis of the upper corner of the first character of text. This point is situated differently according to the alignment value: to the left for a left alignment, to the right for a right alignment or in the center when the text is centered.

The optional font and size parameters can be used to specify the font and its size, in points, to be used. When these parameters are not passed, the text will be written in Times New Roman 12 pt.

The optional style parameter gives the character style used. In this parameter, you must pass one of the following styles, or a combination of several of these values:

- 0 = Plain
- 1 = Bold
- 2 = Italic
- 4 = Underline
- 8 = Strikethrough

The optional alignment parameter can be used to set the type of alignment applied to the text drawn. You can pass one of the following values:

- 2 = Align left
- 3 = Center
- 4 = Align right

The optional color parameter contains the name of the font color. (For more information about colors, please refer to the "Colors and gradients" section).

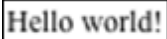
The optional rotation parameter can be used to specify the rotation to be applied to the text.

The optional lineSpacing parameter can be used to specify the value of the line spacing if the text has more than one line. Default value = 1.

The optional stretching parameter can be used to specify a horizontal stretching (value >1) or condensing (valeur <1) factor of the text.

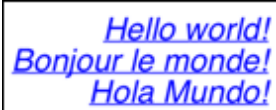
### Examples

1. Simple text using default text properties:



```
$SVG:=SVG_New  
$textID:=SVG_New_text ($SVG;"Hello world!")
```

2. Text that is blue, italic, underlined and aligned to the right:



```
$SVG:=SVG_New  
$text:="Hello world!\rBonjour le monde!\rHola Mundo!"  
$size:=48  
$font:="helvetica"  
$textID:=SVG_New_text ($SVG;$text;400;10;$font;$size;Italic +Underline ;Align Right ;  
"blue")
```

### 3. Vertical text:



```
$SVG:=SVG_New  
$textID:=SVG_New_text ($SVG;$text;-250;0;"";48;-1;-1;"red";-90)
```

### 4. Condensed or expanded text:



```
$SVG:=SVG_New  
$textID:=SVG_New_text ($SVG;"Hello world (condensed)";0;0;"";-1;-1;-1;"blue";0;1;0,8)  
$textID:=SVG_New_text ($SVG;"Hello world (normal)";0;24)  
$textID:=SVG_New_text ($SVG;"Hello world (stretched)";0;48;"";-1;-1;-1;"red";0;1;2)
```

### See Also

SVG\_New\_textArea, SVG\_New\_tspan, SVG\_New\_vertical\_text.

SVG\_New\_textArea (parentSVGObject; text{; x{; y{; width{; height{; font{; size{; style{; alignment}}}}}})) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
text	Text	→	Text to insert
x	Number	→	Coordinate on X axis
y	Number	→	Coordinate on Y axis
width	Number	→	Width of text area
height	Number	→	Height of text area
font	Alpha	→	Font name
size	Integer	→	Size of characters in points
style	Integer	→	Style of characters
alignment	Integer	→	Alignment
Function result	SVG_Ref	←	Reference of SVG text object

### Description

The SVG\_New\_textArea command inserts a text area in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

The "textArea" element is recommended by the SVG tiny 1.2 standard and implemented in 4D v11 SQL beginning with version 11.3 (see <http://www.w3.org/TR/SVGMobile12/text.html#TextAreaElement>). This element implements a text area that, unlike the "text" element, automatically handles the line feed when the text exceeds the width requested.

The optional x and y parameters can be used to specify the position on the X and Y axes of the top left corner of the area.

The optional width and height parameters specify the size of the area in the user coordinate space. If one or the other of these parameters is not provided, the text area will automatically be fitted to its contents.



The optional font and size parameters can be used to specify the font and size, in points, to be used. When these parameters are not passed, the text will be written in Times New Roman 12 pt.

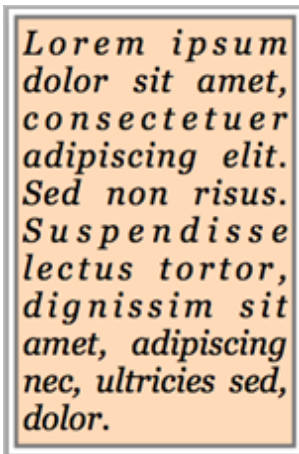
The optional style parameter gives information about the character style used. In the style parameter, you must pass one of the following values or a combination of several of them:

- 0 = Plain
- 1 = Bold
- 2 = Italic
- 4 = Underline
- 8 = Strikethrough

The optional alignment parameter can be used to set the type of alignment to be applied to the drawn text. You can pass one of the following values:

- 1 = Align default (left)
- 2 = Align left
- 3 = Center
- 4 = Align right
- 5 = Justify

#### Example



```
$svg:=SVG_New
```

```
  `Position a border rectangle
```

```
$rec:=SVG_New_rect ($svg;5;5;210;320;0;0;"#777";"peachpuff";3)
```

```
`The text
$txt:="Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse
      lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor."
$txtArea:=SVG_New_textArea ($svg,$txt;10;10;200;310;"Georgia";25;Italic ;5)
`Save document
SVG_SAVE_AS_TEXT ($svg,"test.svg")
```

### See Also

SVG\_New\_text, SVG\_New\_vertical\_text.

SVG\_New\_tspan (parentSVGObject; text{; x{; y{; font{; size{; style{; alignment{; color}}}}}}) → SVG\_Ref

Parameter	Type	Description
parentSVGObject	SVG_Ref	→ Reference of parent element
text	Text	→ Text to insert
x	Number	→ Coordinate on X axis
y	Number	→ Coordinate on Y axis
font	String	→ Font name
size	Integer	→ Size of characters in points
style	Integer	→ Style of characters
alignment	Integer	→ Alignment
color	String	→ Text color
Function result	SVG_Ref	← Reference of SVG text object

**Description**

The SVG\_New\_tspan command creates a new element in the 'text' or 'tspan' element designated by parentSVGObject and returns its reference. If parentSVGObject is not a reference to a 'text' or 'tspan' element, an error is generated.

The different optional parameters are described with the SVG\_New\_text command. If certain optional parameters are omitted, their values are inherited from parent element(s).

**Examples**

1. In a text, it is possible to create paragraphs that inherit the properties of a parent.



`$SVG:=SVG_New`

`Creates a new text that is in Arial, blue, and aligned to the left

`$textID:=SVG_New_text ($SVG;"";0;0;"arial";-1;-1;Align left ;"blue")`

`Nested paragraphs with indentation and changing of size and style

`$textID:=SVG_New_tspan ($textID;"TITLE 1"; 10; 10;""; 24; Bold +Underline)`

`$textID:=SVG_New_tspan ($textID;"Title 2"; 20; 42;""; 12; Bold)`

`$textID:=SVG_New_tspan ($textID;"Title 3"; 30; 60;""; 10; Bold +Italic)`

`$textID:=SVG_New_tspan ($textID;"Title 4"; 40; 78;""; 8; Italic)`

2.Changing a property while remaining in the same "text" element, here the size of the text.

Writing with **SVG is easy**

`$textID:=SVG_New_text ($SVG;"Writing ";10;10;"arial";12)`

`SVG_SET_FONT_SIZE (SVG_New_tspan ($textID;"with ");14)`

`SVG_SET_FONT_SIZE (SVG_New_tspan ($textID;"SVG ");18)`

`SVG_SET_FONT_SIZE (SVG_New_tspan ($textID;"is ");24)`

`SVG_SET_FONT_SIZE (SVG_New_tspan ($textID;"easy ");36)`

**See Also**

SVG\_New\_text.

SVG\_New\_vertical\_text (parentSVGObject; text{; x{; y{; font{; size{; style{; alignment{; color{; rotation}}}}}})) → SVG\_Ref

Parameter	Type		Description
parentSVGObject	SVG_Ref	→	Reference of parent element
text	Text	→	Text to insert
x	Number	→	Coordinate on X axis
y	Number	→	Coordinate on Y axis
font	String	→	Font name
size	Integer	→	Size of characters in points
style	Integer	→	Style of characters
alignment	Integer	→	Alignment
color	String	→	Text color
rotation	Number	→	Angle of rotation of text
Function result	SVG_Ref	←	Reference of SVG text object

### Description

The SVG\_New\_vertical\_text command inserts the text of text vertically in the SVG container designated by parentSVGObject and returns its reference. If parentSVGObject is not an SVG document, an error is generated.

The optional x and y parameters can be used to specify the position on the X and Y axis of the bottom left corner of the first character of text.

The optional font and size parameters can be used to specify the font and its size, in points, to be used. When these parameters are not passed, the text will be written in Times New Roman 12 pt.

The optional style parameter gives the character style used. In this parameter, you must pass one of the following styles, or a combination of several of these values:

- 0 = Plain
- 1 = Bold
- 2 = Italic
- 4 = Underline
- 8 = Strikethrough

The optional alignment parameter can be used to set the type of alignment applied to the text drawn. You can pass on eof the following values:

- 2 = Align left
- 3 = Center
- 4 = Align right

The optional color parameter contains the name of the font color. (For more information about colors, please refer to the "Colors and gradients" section).

The optional rotation parameter can be used to specify the rotation to be applied to the text.

### Example



```
$SVG:=SVG_New  
$textID:=SVG_New_text ($SVG;"Hello world";10;12)  
$textID:=SVG_New_vertical_text ($SVG;"Hello world";22;3;"";-1;-1;Center ;"blue")
```

### See Also

SVG\_New\_text, SVG\_New\_tspan.

SVG\_SET\_FONT\_COLOR (svgObject; color)

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
color	String	→ Text color

### Description

The SVG\_SET\_FONT\_COLOR command can be used to specify the font color for the for the SVG object having the svgObject reference. If svgObject does not reference a valid element, an error is generated.

The color parameter contains the name of the color to be used. (For more information about colors, please refer to the "SVG Colors" section).

### See Also

SVG Colors.

SVG\_SET\_FONT\_FAMILY (svgObject; font)

<b>Parameter</b>	<b>Type</b>	<b>Description</b>
svgObject	SVG_Ref	→ Reference of SVG element
font	String	→ Font name

**Description**

The SVG\_SET\_FONT\_FAMILY command can be used to specify the font for the SVG object having the svgObject reference. If svgObject does not reference a valid element, an error is generated.

The font parameter contains the name of the font to be used.



SVG\_SET\_FONT\_SIZE (svgObject; size)

Parameter	Type	Description
svgObject	SVG_Ref →	Reference of SVG element
size	Integer →	Size of characters in points

### Description

The SVG\_SET\_FONT\_SIZE command can be used to specify the font size for the SVG object having the svgObject reference. If svgObject does not reference a valid element, an error is generated.

The size parameter contains the font size in points.

### See Also

SVG\_SET\_FONT\_STYLE.

SVG\_SET\_FONT\_STYLE (svgObject; style)

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
style	Integer	→ Style of characters

### Description

The SVG\_SET\_FONT\_STYLE command can be used to specify the text style for the SVG object having the svgObject reference. If svgObject does not reference a valid element, an error is generated.

In the style parameter, you must pass one of the following values or a combination of several values:

- 0 = Plain
- 1 = Bold
- 2 = Italic
- 4 = Underline
- 8 = Strikethrough

### See Also

SVG\_SET\_FONT\_SIZE.

SVG\_SET\_TEXT\_ANCHOR (svgObject; alignment)

Parameter	Type	Description
svgObject	SVG_Ref	→ Reference of SVG element
alignment	Integer	→ Alignment

### Description

The SVG\_SET\_TEXT\_ANCHOR command can be used to modify the alignment of the SVG object having the svgObject reference. If svgObject does not reference a valid element, an error is generated.

In the alignment parameter, you must pass one of the following values:

- 1 = Align default (left)
- 2 = Align left
- 3 = Center
- 4 = Align right
- 5 = Justify (only for textArea object)

### See Also

SVG\_New\_textArea.



# 9

---

## Utilities

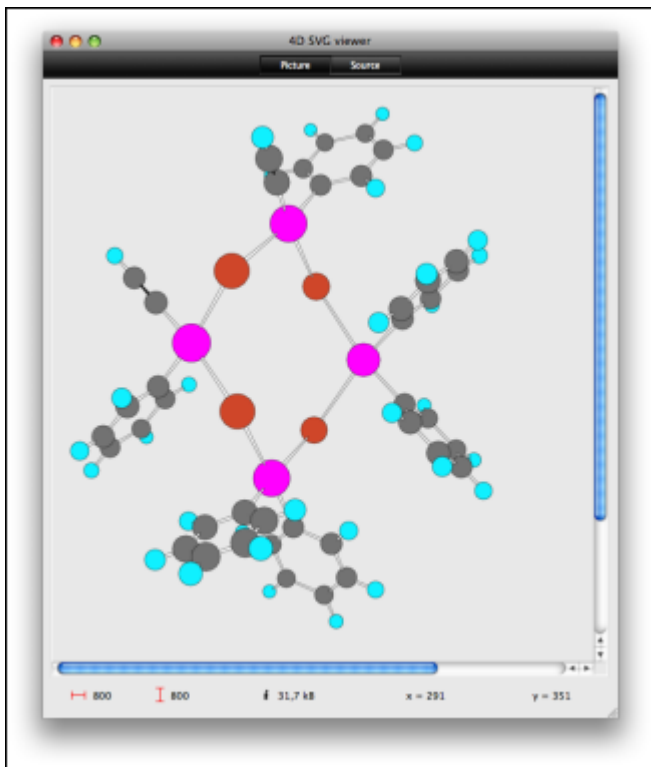


SVGTool\_SHOW\_IN\_VIEWER (svgObject)

Parameter	Type	Description
svgObject	SVG_Ref →	Reference of picture to be displayed

**Description**

The SVGTool\_SHOW\_IN\_VIEWER command displays the SVG picture specified by svgObject in an SVG Viewer window. This tool is provided with the SVG component:



For more information about the SVG Viewer, please refer to the Development Tools section.

**See Also**

Development tools.

SVG\_Count\_elements (svgObject) → Number

Parameter	Type		Description
svgObject	SVG_Ref	→	SVG reference
Function result	Number	←	Number of objects

### Description

The SVG\_Count\_elements command returns the number of graphic objects contains in the svgObject passed as parameter. A group counts as one object. To find out the number of graphic objects in a group, passed its reference to the command. If svgObject is not valid, an error is generated.

### See Also

SVG\_ELEMENTS\_TO\_ARRAYS, SVG\_New\_group.



---

SVG\_ELEMENTS\_TO\_ARRAYS (svgObject; refsArrayPointer{; typesArrayPointer{; namesArrayPointer{;}}

Parameter	Type	Description
svgObject	Alpha	→ SVG reference
refsArrayPointer	Pointer	→ String array of object references
typesArrayPointer	Pointer	→ String array of object types
namesArrayPointer	Pointer	→ String array of object IDs

### Description

The SVG\_ELEMENTS\_TO\_ARRAYS command fills the array pointed to by refsArrayPointer with the references of the graphic objects of the first level for the SVG reference passed in svgObject.

If the optional typesArrayPointer pointer is passed, the array will be filled with the object types.

If the optional namesArrayPointer pointer is passed, the array will be filled with object IDs.

A group counts as one object. To find out this information for the graphic objects in a group, passed its reference to the command.

If svgObject is not valid or if this attribute does not exist, an error is generated.

### See Also

SVG\_Count\_elements, SVG\_New\_group.

SVG\_Estimate\_weight (svgObject) → Number

Parameter	Type		Description
svgObject	SVG_Ref	→	Reference of SVG document
Function result	Number	←	Size in bytes of SVG document

### Description

The `SVG_Estimate_weight` command returns the size in bytes of the SVG tree whose reference is passed as in the `svgObject` parameter. If `svgObject` is not a valid reference, an error is generated.

### See Also

`SVG_Count_elements`.

SVG\_Find\_ID (name) → SVG\_Ref

Parameter	Type		Description
name	Alpha	→	ID of SVG element
Function result	SVG_Ref	←	Reference of element

### Description

The `SVG_Find_ID` command returns the reference of the element whose ID is passed in the name parameter.

If the element is not found, an error is generated.

### See Also

`SVG_Get_ID`, `SVG_SET_ID`.

SVG\_Get\_options → Longint

Parameter	Type	Description
-----------	------	-------------

This command does not require any parameters

Function result	Longint	← Options
-----------------	---------	-----------

### Description

The SVG\_Get\_options command returns a longint representing a 32-bit array where each bit can represent an option of the component. You can use the operators on the 4D bits to check the state of an option (??), and to enable (?+) or disable (?-) one of them.

The following options are currently available:

Bit	Option	Default
1	Assign an ID automatically when creating an element	0 (disabled)
2	Automatically close any objects that can be	0 (disabled)
3	Create objects with a background	1 (enabled)
4	Absolute coordinates for paths	1 (enabled)
5	Create more readable code	0 (disabled)
6	Beep when an error occurs	1 (enabled)
7	Do not display 4D errors	0 (disabled)
8	Transparent pictures	1 (enabled)

- *Assign an ID automatically when creating an element*

If this option is enabled, when the component creates a new element, it systematically adds and fills in an 'id' attribute for the object created, if this is not already specified.

- *Automatically close objects*

If this option is enabled, the objects created with the SVG\_New\_arc and SVG\_New\_polyline\_by\_arrays commands will be automatically closed.

- *Create objects with a background*

If this option is enabled, closed objects will be created with a background color; otherwise, the background will be transparent.

- *Absolute coordinates for paths*

When drawing paths with the SVG\_PATH\_MOVE\_TO, SVG\_PATH\_LINE\_TO, SVG\_PATH\_CURVE and SVG\_PATH\_ARC commands, the coordinates passed will be interpreted as absolute if this option is enabled; otherwise they will be considered as relative.

- *Create more readable code*

This option can be used to create indented and well-spaced code which is nevertheless unwieldy; its activation is particularly useful during the debugging phase.

- *Beep when an error occurs*

When an error occurs and no host database error-handling method has been installed with the SVG\_Set\_error\_handler command, a beep is emitted if this option is enabled.

- *Do not display 4D errors*

This option which is enabled by default blocks the display of 4D errors by installing an error-handling method peculiar to the component. You may prefer not to use this internal management and to let 4D display these messages. This can be useful during debugging for example.

- *Transparent pictures*

By default, SVG pictures created with the SVG\_New command are transparent. By disabling this option, the pictures will be on a white background.

### **Example**

See the SVG\_SET\_OPTIONS command.

### **See Also**

SVG\_SET\_OPTIONS.

SVG\_Get\_version → String

Parameter	Type	Description
-----------	------	-------------

This command does not require any parameters

Function result	String	← Version number
-----------------	--------	------------------

### Description

The `SVG_Get_version` command returns the version of the component in alphanumeric form. The string returned always includes the version and subversion number ("11.0" for the version 11 and "11.3" for the 3rd update of version 11 ). When it is a Beta version, the distribution number is specified, prefixed by the letter "B" ("11.3B1" for the Beta 1 of version 11.3)

SVG\_Is\_reference\_valid (svgObject) → Boolean

Parameter	Type		Description
svgObject	SVG_Ref	→	Reference of SVG element
Function result	Boolean	←	True if reference belongs to an SVG element

### Description

The `SVG_Is_reference_valid` command returns `True` if the reference passed in the `svgObject` parameter is that of an element of the SVG tree. If the element does not belong to an SVG tree, the command returns `False`. If `svgObject` is not a valid reference, an error is generated.

### See Also

`SVG_References_array`.

SVG\_Read\_element\_type (svgObject) → Type

Parameter	Type	Description
svgObject	SVG_Ref →	Reference of SVG element
Function result	Type ←	Type of element

### Description

The `SVG_Read_element_type` command returns the type of element whose reference is passed in the `svgObject` parameter.

If `svgObject` is not a valid reference or if this attribute does not exist, an error is generated.



SVG\_Read\_last\_error → Error

Parameter	Type	Description
This command does not require any parameters		
Function result	Error	← Number of last error

### Description

The `SVG_Read_last_error` command returns the number of the last error that occurred during the execution of a 4D SVG component command and resets this error.

The error number returned can be specific to a component command or may be an error generated by 4D. The following errors are generated by the component:

8850	Insufficient number of parameters
8851	Invalid parameter type
8852	Invalid reference
8853	Incorrect value for attribute
8854	The element does not accept this command
8855	Invalid ( ID not found in document) object name (symbol, marker, filter, etc.)
8856	DTD file not found
8857	Incorrect value for a parameter
8858	Unknown error

### Examples

1. Given the "SVG\_error\_mgmt" method of the example for the `SVG_Set_error_handler` command:

```

`Installation of error-handling method
$ Error_Method_Txt:=SVG_Set_error_handler ("SVG_error_mgmt")
`from now on it is the SVG_error_mgmt method that will be executed
`in the case of an error

`Creation of new SVG document
$SVG:=SVG_New (1200; 900; "SVG Component Test"; ""; True)
SVG_SET_VIEWBOX ($SVG; 0; 0; 1500; 1000)

```

```
If (SVG_Read_last_error =0)
```

```
...
```

```
Else
```

```
  `The SVG_error_mgmt method has been called and has received the error number
```

```
End if
```

```
  `Uninstalling of error-handling method
```

```
SVG_Set_error_handler
```

2. Given the following *SVG\_error\_mgmt* method:

```
C_LONGINT ($1)
```

```
C_TEXT ($2)
```

```
  `Keep the error and the context
```

```
errorNumber:=$1
```

```
commandName:=$2
```

```
  `Set the OK system variable to 0
```

```
OK := 0
```

```
  `Installation of error-handling method
```

```
$ Error_Method_Txt:=SVG_Set_error_handler ("SVG_error_mgmt")
```

```
  `Creation of new SVG document
```

```
$SVG:=SVG_New (1200; 900; " SVG Component Test "; ""; True)
```

```
SVG_SET_VIEWBOX ($SVG; 0; 0; 1500; 1000)
```

```
If (OK = 1)
```

```
...
```

```
Else
```

```
  ALERT("Error No." + String(errorNumber) + " during execution of the command \" +  
commandName + "\")
```

```
End if
```

```
  `Uninstalling of error-handling method
```

```
SVG_Set_error_handler
```

**See Also**

SVG\_Set\_error\_handler.

SVG\_References\_array (refsArrayPointer)

Parameter	Type	Description
refsArrayPointer	Pointer	→ Alpha array of document references

### Description

The `SVG_References_array` command returns the list of current references in the array pointed to by `refsArrayPointer`. This command is useful when debugging.

Each time an SVG document is created with the `SVG_New`, `SVG_Copy` or `SVG_Open_file` component commands, the component adds the reference returned by the command to an internal array. When an SVG document is released using the `SVG_CLEAR` command, the component removes its reference from the array.

### See Also

`SVG_Is_reference_valid`.

---

SVG\_Set\_error\_handler {(method)} → String

Parameter	Type	Description
method	String	→ Name of method to install
Function result	String	← Name of method previously installed

### Description

The `SVG_Set_error_handler` command can be used to install the host database method `method` that will be called in the case of an error and which returns the name of the previously installed method.

The commands of the component carry out a minimum of verifications when they are called: minimum number of parameters, validity of references, for the element to which a command is applied. The component thus handles errors in a structured manner and allows the host database to retrieve any errors.

When default functioning has not been modified, if an error occurs a beep is emitted and the command is interrupted.

The host database can retrieve the error number and the name of the faulty command in one of these methods. To do this, the host database must create a method that will receive the error number as the first parameter and the command name as the second parameter.

This method, if it is installed by the `SVG_Set_error_handler` command, will be called when an error occurs; in this case, no beep is generated by the code of the component.

If `method` is omitted or is an empty string, the method is uninstalled and the behavior changes back to the default behavior.

**Note:** The host database method that will be called by the component must have the "Shared by components and host database" property.

## Example

Installation of the *SVG\_error\_mgmt* method (host database method) as the error-handling method:

```
$error:=SVG_Set_error_handler("SVG_error_mgmt")
```

Method code:

```
` SVG_error_mgmt method
ALERT ("Error No." + String($1) +" during execution of the command \""+$2+"\"")
```

## See Also

*SVG\_Read\_last\_error*.

SVG\_SET\_OPTIONS {(options)}

Parameter	Type	Description
options	Longint	→ 4D SVG component options

### Description

The SVG\_SET\_OPTIONS command can be used to set the options of the 4D SVG component with the options longint. For more information about the contents of options, please refer to the description of the SVG\_Get\_options command.

Since all options will be set at once, this command must have been preceded with a call to the SVG\_Get\_options command, followed by the use of the Bitwise Operators of 4D.

If the options parameter is not passed, all the options are reset to their default value (see the SVG\_Get\_options command).

### Examples

1. Create readable code:
 

```

$Options := SVG_Get_options
$Options := $Options ?+ 5 `enable the option
SVG_SET_OPTIONS ($Options)
      
```

2. Draw a pie chart diagram:



`$svg:=SVG_New`

Enable automatic closing of objects  
**SVG\_SET\_OPTIONS** (**SVG\_Get\_options** ?+ 2)

```
SVG_New_arc ($svg;100;100;90;0;105;"gray";"lightcoral";1)  
SVG_New_arc ($svg;100;100;90;105;138;"gray";"lightskyblue";1)  
SVG_New_arc ($svg;100;100;90;138;230;"gray";"lightgreen";1)  
SVG_New_arc ($svg;100;100;90;230;270;"gray";"lightsteelblue";1)  
SVG_New_arc ($svg;100;100;90;270;360;"gray";"lightyellow";1)
```

### See Also

SVG\_Get\_options.





# 10

---

# Appendixes



Here is the list of colors recognized by SVG. For more information, please refer to the SVG Colors section.

	aliceblue	rgb(240, 248, 255)		darkslategrey	rgb(47, 79, 79)
	antiquewhite	rgb(250, 235, 215)		darkturquoise	rgb(0, 206, 209)
	aqua	rgb(0, 255, 255)		darkviolet	rgb(148, 0, 214)
	aquamarine	rgb(127, 255, 212)		deeppink	rgb(255, 20, 147)
	azure	rgb(240, 255, 255)		deepskyblue	rgb(0, 191, 255)
	beige	rgb(245, 245, 220)		dimgray	rgb(105, 105, 105)
	bisque	rgb(255, 228, 196)		dimgray	rgb(105, 105, 105)
	black	rgb(0, 0, 0)		dodgerblue	rgb(30, 144, 255)
	blanchedalmond	rgb(255, 235, 205)		firebrick	rgb(178, 34, 34)
	blue	rgb(0, 0, 255)		floralwhite	rgb(255, 250, 240)
	blueviolet	rgb(138, 43, 226)		forestgreen	rgb(34, 139, 34)
	brown	rgb(165, 42, 42)		fuchsia	rgb(255, 0, 255)
	burlywood	rgb(222, 184, 135)		gainsboro	rgb(220, 220, 220)
	cadetblue	rgb(95, 158, 160)		ghostwhite	rgb(248, 248, 255)
	chartreuse	rgb(127, 255, 0)		gold	rgb(255, 215, 0)
	chocolate	rgb(210, 105, 30)		goldenrod	rgb(218, 165, 32)
	coral	rgb(255, 127, 80)		gray	rgb(128, 128, 128)
	cornflowerblue	rgb(100, 149, 237)		grey	rgb(128, 128, 128)
	cornsilk	rgb(255, 248, 220)		green	rgb(0, 128, 0)
	crimson	rgb(220, 20, 60)		greenyellow	rgb(173, 255, 47)
	cyan	rgb(0, 255, 255)		honeydew	rgb(240, 255, 240)
	darkblue	rgb(0, 0, 139)		hotpink	rgb(255, 105, 180)
	darkcyan	rgb(0, 139, 139)		indianred	rgb(205, 92, 92)
	darkgoldenrod	rgb(184, 134, 11)		indigo	rgb(75, 0, 130)
	darkgray	rgb(169, 169, 169)		ivory	rgb(255, 255, 240)
	darkgreen	rgb(0, 100, 0)		khaki	rgb(240, 230, 140)
	darkgrey	rgb(169, 169, 169)		lavender	rgb(230, 230, 250)
	darkkhaki	rgb(189, 183, 107)		lavenderblush	rgb(255, 240, 245)
	darkmagenta	rgb(139, 0, 139)		lawngreen	rgb(124, 252, 0)
	darkolivegreen	rgb(85, 107, 47)		lemonchiffon	rgb(255, 250, 205)
	darkorange	rgb(255, 140, 0)		lightblue	rgb(173, 216, 230)
	darkorchid	rgb(153, 50, 204)		lightcoral	rgb(240, 128, 128)
	darkred	rgb(139, 0, 0)		lightcyan	rgb(224, 255, 255)
	darksalmon	rgb(233, 150, 122)		lightgoldenrodyellow	rgb(250, 250, 210)
	darkseagreen	rgb(143, 188, 143)		lightgray	rgb(211, 211, 211)
	darkslateblue	rgb(72, 61, 139)		lightgreen	rgb(144, 238, 144)
	darkslategrey	rgb(47, 79, 79)		lightgrey	rgb(211, 211, 211)

	lightpink	rgb (255, 182, 193)
	lightsalmon	rgb (255, 160, 122)
	lightseagreen	rgb (32, 178, 170)
	lightskyblue	rgb (135, 206, 250)
	lightslategray	rgb (119, 136, 153)
	lightslategray	rgb (119, 136, 153)
	lightsteelblue	rgb (176, 196, 222)
	lightyellow	rgb (255, 255, 224)
	lime	rgb (0, 255, 0)
	limegreen	rgb (50, 205, 50)
	linen	rgb (250, 240, 230)
	magenta	rgb (255, 0, 255)
	maroon	rgb (128, 0, 0)
	mediumaquamarine	rgb (102, 205, 170)
	mediumblue	rgb (0, 0, 205)
	mediumorchid	rgb (186, 85, 211)
	mediumpurple	rgb (147, 112, 219)
	mediumseagreen	rgb (60, 179, 153)
	mediumslateblue	rgb (123, 104, 238)
	mediumspringgreen	rgb (0, 250, 254)
	mediumturquoise	rgb (72, 209, 204)
	mediumvioletred	rgb (199, 21, 133)
	midnightblue	rgb (25, 25, 112)
	mintcream	rgb (245, 255, 250)
	mistyrose	rgb (255, 228, 225)
	moccasin	rgb (255, 228, 181)
	navajowhite	rgb (255, 222, 173)
	navy	rgb (0, 0, 128)
	oldlace	rgb (253, 245, 230)
	olive	rgb (128, 128, 0)
	olivedrab	rgb (107, 142, 35)
	orange	rgb (255, 165, 0)
	orangered	rgb (255, 69, 0)
	orchid	rgb (218, 112, 214)
	palegoldenrod	rgb (238, 232, 170)
	palegreen	rgb (152, 251, 152)
	paleturquoise	rgb (175, 238, 238)

	palevioletred	rgb (219, 112, 147)
	papayawhip	rgb (255, 239, 213)
	peachpuff	rgb (255, 218, 185)
	peru	rgb (205, 133, 63)
	pink	rgb (255, 192, 203)
	plum	rgb (221, 160, 221)
	powderblue	rgb (176, 224, 230)
	purple	rgb (128, 0, 128)
	red	rgb (255, 0, 0)
	rosybrown	rgb (188, 143, 143)
	royalblue	rgb (65, 105, 225)
	saddlebrown	rgb (139, 69, 19)
	salmon	rgb (250, 128, 114)
	sandybrown	rgb (244, 164, 96)
	seagreen	rgb (46, 139, 87)
	seashell	rgb (255, 245, 238)
	sienna	rgb (160, 82, 45)
	silver	rgb (192, 192, 192)
	skyblue	rgb (135, 206, 235)
	slateblue	rgb (106, 90, 205)
	slategray	rgb (112, 128, 144)
	slategray	rgb (112, 128, 144)
	snow	rgb (255, 250, 250)
	springgreen	rgb (0, 255, 127)
	steelblue	rgb (70, 130, 180)
	tan	rgb (210, 180, 140)
	teal	rgb (0, 128, 128)
	thistle	rgb (216, 194, 216)
	tomato	rgb (255, 99, 71)
	turquoise	rgb (64, 224, 208)
	violet	rgb (238, 130, 238)
	wheat	rgb (245, 222, 179)
	white	rgb (255, 255, 255)
	whitesmoke	rgb (245, 245, 245)
	yellow	rgb (255, 255, 0)
	yellowgreen	rgb (154, 205, 50)

Here is a list of links concerning the SVG standard:

### **Overview**

<http://www.w3.org/Graphics/SVG/>

### **Specification**

<http://www.w3.org/TR/SVG12/> (Status: Working Draft 13 April 2005)

<http://www.yoyodesign.org/doc/w3c/svg1/> (French translation (Version 1.0))

### **Tutorials**

<http://www.w3.org/Consortium/Offices/Presentations/SVG/1.svg>

### **Community**

<http://svg.org/>

<http://svgfr.org>

<http://www.openclipart.org/>

<http://www.gosvg.net/>



# Command Index

## A

SVG\_ADD\_POINT..... 73

## C

SVG\_CLEAR.....59  
SVG\_Color\_grey.....52  
SVG\_Color\_RGB\_from\_long.....53  
SVG\_Copy.....60  
SVG\_Count\_elements.....160

## D

SVG\_Define\_filter.....121  
SVG\_Define\_linear\_gradient.....123  
SVG\_Define\_marker.....126  
SVG\_Define\_radial\_gradient.....128  
SVG\_Define\_shadow.....130  
SVG\_Define\_solidColor.....132  
SVG\_Define\_symbol.....133

## E

SVG\_ELEMENTS\_TO\_ARRAYS.....161  
SVG\_Estimate\_weight.....162  
SVG\_Export\_to\_picture.....61  
SVG\_Export\_to\_XML.....62

## F

SVG\_Filter\_Blend.....116  
SVG\_Filter\_Blur.....117

SVG_Filter_Offset.....	118
SVG_Find_ID.....	163

## G

SVG_GET_ATTRIBUTES.....	19
SVG_GET_DEFAULT_BRUSHES.....	54
SVG_Get_ID.....	20
SVG_Get_options.....	164
SVG_Get_version.....	166

## I

SVG_Is_reference_valid.....	167
-----------------------------	-----

## N

SVG_New.....	63
SVG_New_arc.....	74
SVG_New_circle.....	76
SVG_New_ellipse.....	78
SVG_New_ellipse_bounded.....	80
SVG_New_embedded_image.....	82
SVG_New_group.....	134
SVG_New_image.....	84
SVG_New_line.....	86
SVG_New_path.....	87
SVG_New_polygon.....	91
SVG_New_polygon_by_arrays.....	92
SVG_New_polyline.....	94
SVG_New_polyline_by_arrays.....	96
SVG_New_rect.....	99
SVG_New_regular_polygon.....	101
SVG_New_text.....	141
SVG_New_textArea.....	144



SVG_New_tspan.....	147
SVG_New_vertical_text.....	149

## O

SVGTool_SHOW_IN_VIEWER.....	159
SVG_Open_file.....	65
SVG_Open_picture.....	66

## P

SVG_PATH_ARC.....	103
SVG_PATH_CLOSE.....	105
SVG_PATH_CURVE.....	106
SVG_PATH_LINE_TO.....	107
SVG_PATH_MOVE_TO.....	108
SVG_PATH_QCURVE.....	109

## R

SVG_Read_element_type.....	168
SVG_Read_last_error.....	169
SVG_References_array.....	171

## S

SVG_SAVE_AS_PICTURE.....	67
SVG_SAVE_AS_TEXT.....	68
SVG_SET_ATTRIBUTES.....	21
SVG_SET_ATTRIBUTES_BY_ARRAYS.....	22
SVG_SET_DEFAULT_BRUSHES.....	55
SVG_Set_description.....	136
SVG_SET_DIMENSIONS.....	23
SVG_Set_error_handler.....	172

SVG_SET_FILL_BRUSH.....	24
SVG_SET_FILTER.....	25
SVG_SET_FONT_COLOR.....	151
SVG_SET_FONT_FAMILY.....	152
SVG_SET_FONT_SIZE.....	153
SVG_SET_FONT_STYLE.....	154
SVG_SET_ID.....	26
SVG_SET_MARKER.....	27
SVG_SET_OPACITY.....	30
SVG_SET_OPTIONS.....	174
SVG_SET_ROUNDING_RECT.....	31
SVG_SET_STROKE_BRUSH.....	32
SVG_SET_STROKE_LINECAP.....	33
SVG_SET_STROKE_LINEJOIN.....	34
SVG_SET_STROKE_WIDTH.....	35
SVG_SET_TEXT_ANCHOR.....	155
SVG_Set_title.....	137
SVG_SET_TRANSFORM_FLIP.....	36
SVG_SET_TRANSFORM_MATRIX.....	38
SVG_SET_TRANSFORM_ROTATE.....	40
SVG_SET_TRANSFORM_SCALE.....	41
SVG_SET_TRANSFORM_SKEW.....	42
SVG_SET_TRANSFORM_TRANSLATE.....	43
SVG_SET_VIEWBOX.....	44
SVG_SET_VIEWPORT_FILL.....	45
SVG_SET_VISIBILITY.....	46
SVG_SET_XY.....	47

## U

SVG_Use.....	110
--------------	-----

## V

SVG_Validate_file.....	69
------------------------	----