

# 4D SVG Component

Introduction

Attributes

Colors and Gradients

Documents

Drawing

Filters

Structure and Definitions

Text

Utilities

4D SVG Constants

Appendices

What's new

# Introduction

---

[4D SVG Component](#)  
[Development tools](#)  
[Syntax details](#)

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.

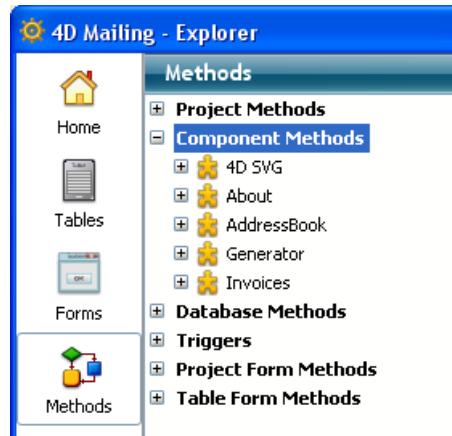
The XML language used for manipulating SVG pictures is particularly rich and extensive. In order to make getting started easier, 4D provides the SVG component, which includes numerous commands that can be used to create and manipulate common graphic objects. This library is not intended to be exhaustive 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

You can install the 4D SVG Component in 4D v11 SQL release 3 (version 11.3) or higher. The host database must be running in Unicode mode (you cannot use this component in databases running in ASCII Compatibility mode).

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.

## SVG Effects and Direct2D (Windows)

Starting with 4D v13, the Direct2D graphics rendering engine is used by default on Windows. Depending on your hardware and software configuration, using this engine can alter the rendering of some SVG effects such as shadows. In this case, you can disable Direct2D in your application with the **SET DATABASE PARAMETER** command.

# Development tools

---

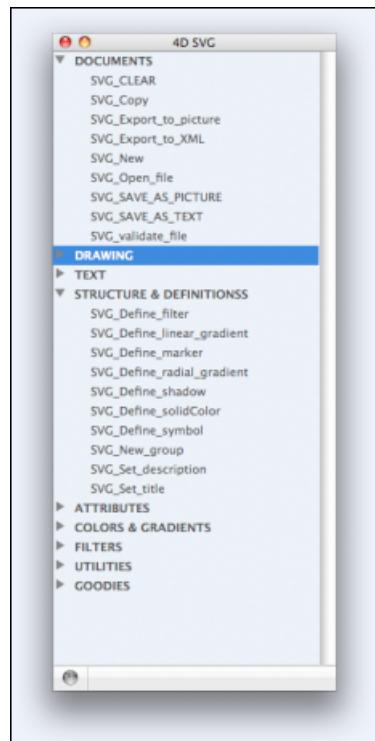
The 4D SVG component provides a set of tools intended to facilitate the entry of code and to preview the SVG graphics:

- 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:



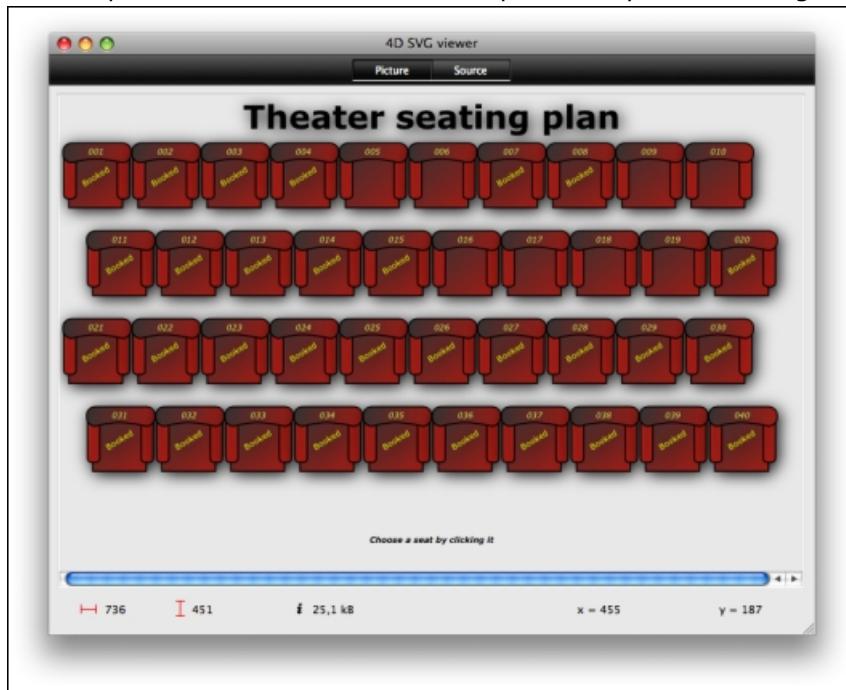
You can use this palette to insert an SVG color reference by drag and drop into the 4D Method editor. The color is inserted as a string that includes the rate of opacity, if any (for example "lavender:30" for the color lavender with an opacity of 30%). For more information about color references, please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

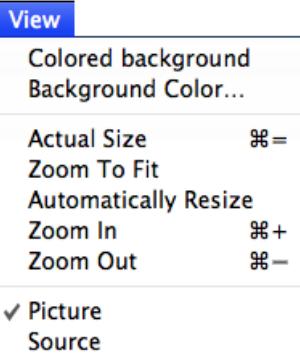
4D SVG provides an SVG viewer that is particularly useful during the development phase:



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 context 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).

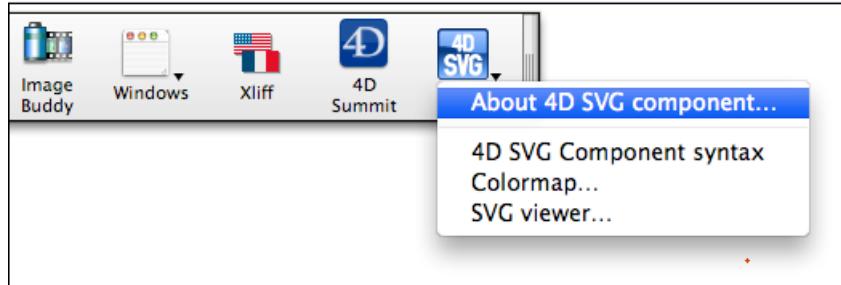
## Integration to 4D Pop

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.

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:



**Note:** 4DPop is included in the additional tools on the 4D full installer.

## 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.

# Attributes

---

SVG\_ADD\_NAMESPACE  
SVG\_GET\_ATTRIBUTES  
SVG\_Get\_class  
SVG\_Get\_fill\_brush  
SVG\_Get\_ID  
SVG\_SET\_ATTRIBUTES  
SVG\_SET\_ATTRIBUTES\_BY\_ARRAYS  
SVG\_SET\_CLASS  
SVG\_SET\_CLIP\_PATH  
SVG\_SET\_DIMENSIONS  
SVG\_SET\_FILL\_BRUSH  
SVG\_SET\_FILL\_RULE  
SVG\_SET\_FILTER  
SVG\_SET\_ID  
SVG\_SET\_MARKER  
SVG\_SET\_OPACITY  
SVG\_SET\_ROUNDING\_RECT  
SVG\_SET\_SHAPE\_RENDERING  
SVG\_SET\_STROKE\_BRUSH  
SVG\_SET\_STROKE\_DASHARRAY  
SVG\_SET\_STROKE\_LINECAP  
SVG\_SET\_STROKE\_LINEJOIN  
SVG\_SET\_STROKE\_MITERLIMIT  
SVG\_SET\_STROKE\_VECTOR\_EFFECT  
SVG\_SET\_STROKE\_WIDTH  
SVG\_SET\_TRANSFORM\_FLIP  
SVG\_SET\_TRANSFORM\_MATRIX  
SVG\_SET\_TRANSFORM\_ROTATE  
SVG\_SET\_TRANSFORM\_SCALE  
SVG\_SET\_TRANSFORM\_SKEW  
SVG\_SET\_TRANSFORM\_TRANSLATE  
SVG\_SET\_VIEWBOX  
SVG\_SET\_VIEWPORT\_FILL  
SVG\_SET\_VISIBILITY  
SVG\_SET\_XY

# SVG\_ADD\_NAMESPACE

SVG\_ADD\_NAMESPACE ( *svgObject* ; *prefix* {; *URI*} )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
<i>prefix</i>	Text	Prefix of namespace
<i>URI</i>	Text	URI of namespace

## Description

The **SVG\_ADD\_NAMESPACE** method adds an XML namespace attribute to the root of the DOM Tree for the SVG object designated by the *svgObject* parameter. You can use this method, more specifically, to add a namespace to an SVG code snippet.

In *prefix*, pass a string containing the prefix of the namespace attribute. You can use one of the following constants:

- "svgNS" for a standard SVG namespace ( <http://www.w3.org/2000/svg> )
- "xlinkNS" for a standard XLink namespace ( <http://www.w3.org/1999/xlink> )

In this case, the *URI* parameter is unnecessary.

You can also pass the prefix of a custom namespace in the *prefix* parameter and its *URI* in the corresponding parameter. In this case, the *URI* parameter is mandatory and if it is omitted, an error is generated.

## Example

The following code:

```
SVG_ADD_NAMESPACE( $svgRef; "svgNS" )
```

... adds the following code to the root of the SVG object:

```
<xmllns="http://www.w3.org/2000/svg">
```

# SVG\_GET\_ATTRIBUTES

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG reference
<i>namesArrayPointer</i>	Pointer	Alpha array of attribute names
<i>valuesArrayPointer</i>	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\_class

SVG\_Get\_class ( *svgObject* {; *classNames*} ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
<i>classNames</i>	Pointer	Pointer to array of class names
Function result	Text	Class name(s)

## Description

The **SVG\_Get\_class** command returns the class name(s) for an SVG image whose reference is passed in the *svgObject* parameter. Class name(s) are returned as a string, with each name separated by a space.

In the optional *classNames* parameter, you can pass a pointer to an array, whose elements will be filled with the class name(s).

## Example

```
// define 2 styles
$SVG_Define_style( $Dom_SVG;".colored {fill: yellow; fill-opacity: 0.6; stroke: red; stroke-width: 8; stroke-opacity: 0.6}")
$SVG_Define_style( $Dom_SVG;".blue {fill: blue}")

// create a group and set a default style
$Dom_g:= $SVG_New_group( $Dom_SVG)
$SVG_Set_Class( $Dom_g;"colored blue")

ARRAY TEXT( $tTxt_Classes;0)
$txt_buffer:= $SVG_Get_class( $Dom_g;-$> $tTxt_classes)

// $Txt_buffer = "colored blue"
// $tTxt_classes{1} = "colored"
// $tTxt_classes{2} = "blue"
```

## SVG\_Get\_fill\_brush

SVG\_Get\_fill\_brush ( *svgObject* ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
Function result	Text	Fill color

### Description

---

The **SVG\_Get\_fill\_brush** command returns the fill color of an SVG image whose reference is passed in the *svgObject* parameter. If *svgObject* does not have a "fill" attribute, the command returns an empty string.

# SVG\_Get\_ID

SVG\_Get\_ID ( *svgObject* ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>attributeName</i>	String	Name of attribute to set
<i>attributeValue</i>	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>namesArrayPointer</i>	Pointer	Names of attributes
<i>valuesArrayPointer</i>	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\_CLASS

SVG\_SET\_CLASS ( *svgObject* ; *class* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>class</i>	Text	Name of class

## Description

---

The *SVG\_SET\_CLASS* command sets the *class* for the object passed in *svgObject*. An error is generated if *svgObject* is not a valid reference.

**See Also:** <http://www.w3.org/TR/SVG/styling.html#ClassAttribute>

## Example

---

Refer to the example for the [SVG\\_Define\\_style](#) command.

# SVG\_SET\_CLIP\_PATH

SVG\_SET\_CLIP\_PATH ( *svgObject* ; *clipPathID* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>clipPathID</i>	Text	Name of clip path

## Description

The *SVG\_SET\_CLIP\_PATH* command sets the clip path named *clipPathID* for the object passed in *svgObject*. An error is generated if *svgObject* is not a valid reference or if the clip path is not defined.

See Also: <http://www.w3.org/TR/2001/REC-SVG-20010904/masking.html#EstablishingANewClippingPath>

## Example 1

Defining a circular clip path that will then be assigned to an image:



```
//Defining a circular clip path
$Dom_clipPath:= SVG_Define_clip_path( $Dom_SVG;"theClip")
$Dom_circle:= SVG_New_circle( $Dom_clipPath;150;100;100)

//Creating a group
$Dom_g:= SVG_New_group( $Dom_SVG)

//Inserting an image
$Txt_path:= Get 4D folder(6)+"logo.svg"
READ PICTURE FILE( $Txt_path; $Pic_buffer)
$Dom_picture:= SVG_New_embedded_image( $Dom_g; $Pic_buffer)
SVG_SET_ID( $Dom_picture;"MyPicture")

//Applying clip path to group
SVG_SET_CLIP_PATH( $Dom_g;"theClip")
```

## Example 2

The same image with a rectangular clip path with rounded corners:



```
//Defining a rectangular clip path
$Dom_clipPath:= SVG_Define_clip_path( $Dom_SVG;"theClip")
```

```
$Dom_rect:= SVG_New_rect( $Dom_clipPath;5;10;320;240;10;10)

//Creating a group
$Dom_g:= SVG_New_group( $Dom_SVG)

//Inserting an image
$txt_path:= Get 4D folder(6)+"logo.svg"
READ PICTURE FILE( $Txt_path; $Pic_buffer)
$Dom_picture:= SVG_New_embedded_image( $Dom_g; $Pic_buffer)
SVG_SET_ID( $Dom_picture;"MyPicture")

//Applying clip path to group
SVG_SET_CLIP_PATH( $Dom_g;"theClip")
```

## See also

[SVG\\_Define\\_clip\\_path](#)

# SVG\_SET\_DIMENSIONS

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>width</i>	Longint	Dimension on the X axis
<i>height</i>	Longint	Dimension on the Y axis
<i>unit</i>	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>color</i>	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\_FILL\_RULE

SVG\_SET\_FILL\_RULE ( *svgObject* ; *fillRule* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>fillRule</i>	Text	Mode for filling object

### Description

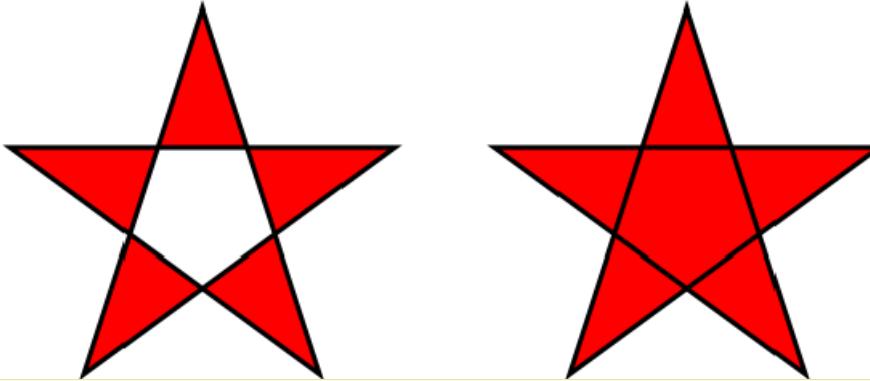
The *SVG\_SET\_FILL\_RULE* command is used to specify the fill rule for the SVG object designated by *svgObject*. An error is generated if *svgObject* is not a valid reference.

The *fillRule* parameter must contain one of the following values: "nonzero", "evenodd" or "inherit". Otherwise, an error is generated.

See Also: <http://www.w3.org/TR/SVG/painting.html#FillRuleProperty>

### Example

Illustration of filling modes:



```
//Creating a path with the 'evenodd' fill rule
$Dom_path:= SVG_New_path( $Dom_SVG;250;75)
SVG_PATH_LINE_TO( $Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE( $Dom_path)
SVG_SET_FILL_BRUSH( $Dom_path;"red")
SVG_SET_STROKE_WIDTH( $Dom_path;3)
SVG_SET_FILL_RULE( $Dom_path;"evenodd")

//Creating a similar object with the 'nonzero' fill rule
$Dom_path:= SVG_New_path( $Dom_SVG;250;75)
SVG_PATH_LINE_TO( $Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE( $Dom_path)
SVG_SET_FILL_BRUSH( $Dom_path;"red")
SVG_SET_STROKE_WIDTH( $Dom_path;3)
SVG_SET_FILL_RULE( $Dom_path;"nonzero")
//Horizontal movement
SVG_SET_TRANSFORM_TRANSLATE( $Dom_path;300)
```

## SVG\_SET\_FILTER

SVG\_SET\_FILTER ( *svgObject* ; *id* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>id</i>	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

SVG\_SET\_ID ( *svgObject* ; *id* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>id</i>	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

SVG\_SET\_MARKER ( *svgObject* ; *id* {; *position*} )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>id</i>	String	Name of marker
<i>position</i>	String	Position of marker

### Description

The **SVG\_SET\_MARKER** command can be used to associate a marker with the object having the *svgObject* reference or to remove an existing marker. 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 *id* 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.

In order to remove an existing marker, pass "none" or an empty string in the *id* parameter.

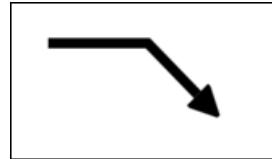
The optional *position* parameter can be used to set the position of the marker with respect to the object. It is possible to place different markers (if desired) at the beginning, end or any other peak of a path. The values may be as follows:

- *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.

### Example 1

Draw an arrow:

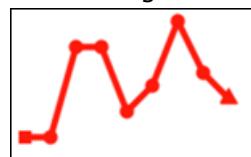


```
$SVG:= SVG_New
` Set the arrow
$arrow:= SVG_Define_marker( $SVG;"arrow";0;5;4;3;-1)
SVG_SET_VIEWBOX( $arrow;0;0;10;10)
$path:= SVG_New_path( $arrow;0;0)
SVG_SET_FILL_BRUSH( $path;"black")
SVG_PATH_LINE_TO( $path;10;5)
SVG_PATH_LINE_TO( $path;0;10)
SVG_PATH_CLOSE( $path)

$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 ")
```

### Example 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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>background</i>	Longint	Opacity (%)
<i>line</i>	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>roundedX</i>	Longint	Radius on X axis
<i>roundedY</i>	Longint	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\_SHAPE\_RENDERING

SVG\_SET\_SHAPE\_RENDERING ( *svgObject* ; *rendering* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>rendering</i>	Text	Type of rendering

### Description

---

The *SVG\_SET\_SHAPE\_RENDERING* command can be used to set which tradeoffs should be made regarding the rendering of graphic elements for the object designated by *svgObject*. If *svgObject* is not an SVG object, an error is generated.

The *rendering* parameter must contain one of the following values: "auto", "optimizeSpeed", "crispEdges", "geometricPrecision" or "inherit". Otherwise, an error is generated.

**See Also:** <http://www.w3.org/TR/2001/REC-SVG-20010904/painting.html#ShapeRenderingProperty>

# SVG\_SET\_STROKE\_BRUSH

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>color</i>	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\_DASHARRAY

SVG\_SET\_STROKE\_DASHARRAY ( *svgObject* ; *dash* {; *value*} {; *value2* ; ... ; *valueN*} )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>dash</i>	Real	Length of first dash
<i>value</i>	Longint	Length of spaces and dashes

## Description

The *SVG\_SET\_STROKE\_DASHARRAY* command is used to set the pattern of dashes and gaps used to outline the path of the SVG object passed in *svgObject*. If *svgObject* is not a valid SVG reference, an error is generated.

The whole value of the *dash* parameter indicates the length of the first dash of the dotted pattern. If the *value* parameters are omitted, the dotted line will consist of a series of dashes and gaps of the same length.

The decimal value of the *dash* parameter, if it is not null, indicates the distance into the pattern from which the dashes will start.

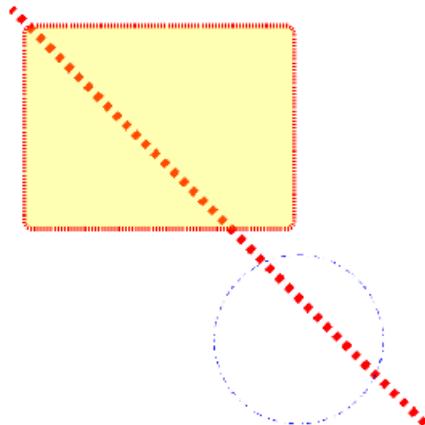
If *dash* is 0, the dotted pattern is removed.

The *value* parameters alternately specify the lengths of the gaps and dashes that follow the first dash. If an odd number of values is given (including the first dash), the list of values is repeated until it produces an even number of values.

**See Also:** <http://www.w3.org/TR/SVG/painting.html#StrokeProperties>

## Example

Illustrations of a dotted line path:



```
//Line
$Dom_line:= SVG_New_line( $Dom_SVG;10;10;500;500)
SVG_SET_STROKE_WIDTH( $Dom_line;10)
SVG_SET_STROKE_DASHARRAY( $Dom_line;8,099)
SVG_SET_STROKE_BRUSH( $Dom_line;"red")

//Rectangle
$Dom_rect:= SVG_New_rect( $Dom_SVG;25;30;320;240;10;10;"red";"yellow:30")
SVG_SET_STROKE_WIDTH( $Dom_rect;5)
SVG_SET_STROKE_DASHARRAY( $Dom_rect;2)

//Circle
$Dom_circle:= SVG_New_circle( $Dom_SVG;350;400;100;"blue";"none")
SVG_SET_STROKE_DASHARRAY( $Dom_circle;2;4;6;8)
```

## SVG\_SET\_STROKE\_LINECAP

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

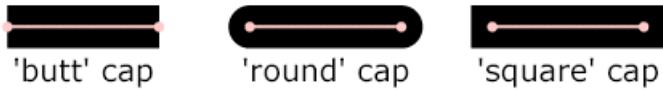
Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>mode</i>	String	Rendering mode

### 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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>mode</i>	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\_MITERLIMIT

SVG\_SET\_STROKE\_MITERLIMIT ( *svgObject* ; *join* )

Parameter	Type		Description
<i>svgObject</i>	SVG_Ref	→	Reference of SVG element
<i>join</i>	Longint	→	Value of join

### Description

---

The *SVG\_SET\_STROKE\_MITERLIMIT* command is used to set the limit for the length of the miter join between the path and the outline of the SVG object designated by *svgObject*. If *svgObject* is not a valid SVG reference, an error is generated.

If the *join* parameter is -1, the value will be the default value (4). If the *join* parameter is 0, then the definition of the attribute is removed. Any other value < 0 will cause an error.

**See Also:** <http://www.w3.org/TR/SVG/painting.html#StrokeProperties>

# SVG\_SET\_STROKE\_VECTOR\_EFFECT

SVG\_SET\_STROKE\_VECTOR\_EFFECT ( *svgObj* {; *vectorEffect*} )

Parameter	Type	Description
<i>svgObj</i>	SVG_Ref	Reference of SVG element
<i>vectorEffect</i>	String	"non-scaling-stroke", "none", "inherit"

## Description

The **SVG\_SET\_STROKE\_VECTOR\_EFFECT** command sets the vector-effect attribute for the lines of the *svgObj*. The vector-effect attribute specifies the behavior of an object's stroke if the object is transformed, for example by zooming in or out.

In the *svgObj* parameter, pass a reference to the SVG element whose vector-effect attribute will be set. If *svgObj* does not reference an SVG element, an error is generated. If the vector-effect attribute has already been defined, its value is replaced by the *vectorEffect* parameter.

The optional *vectorEffect* parameter indicates the value to be used for the attribute. The following values are available:

<b>vectorEffect</b>	<b>Description</b>
"non-scaling-stroke"	Stroke (line) width is not dependant on the transformations of the element
"none"	No vector effect is applied
"inherit"	The value of the element's parent is used

If *vectorEffect* is omitted, the vector-effect attribute is removed (equivalent to "none").

## Example

The following code creates a rectangle and resizes it:

```
C_TEXT( $Dom_rect; $Dom_svg)
$Dom_svg:= SVG_New
$Dom_rect:= SVG_New_rect( $Dom_svg;10;10;100;100;0;0;"blue";"red")
SVG_SET_STROKE_WIDTH( $Dom_rect;2;"px")

// Redimension
SVG_SET_TRANSFORM_SCALE( $Dom_rect;4;1)
```

By default, line thickness is modified:



To resize without changing line thickness, add **SVG\_SET\_STROKE\_VECTOR\_EFFECT**:

```
C_TEXT( $Dom_rect; $Dom_svg)
$Dom_svg:= SVG_New
$Dom_rect:= SVG_New_rect( $Dom_svg;10;10;100;100;0;0;"blue";"red")
SVG_SET_STROKE_WIDTH( $Dom_rect;2;"px")

// Redimension
SVG_SET_TRANSFORM_SCALE( $Dom_rect;4;1)

// Set non-scaling
SVG_SET_STROKE_VECTOR_EFFECT( $Dom_rect;"non-scaling-stroke")
```

The line thickness is not modified:



# SVG\_SET\_STROKE\_WIDTH

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>strokeWidth</i>	Real	Line thickness
<i>unit</i>	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
horizontal	Boolean	Horizontal flip
vertical	Boolean	Vertical flip

### Description

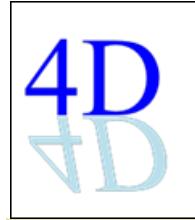
The *SVG\_SET\_TRANSFORM\_FLIP* command can be used to apply a horizontal and/or vertical flip to an SVG object having the *svgObject* reference.

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
& NBSP;
`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; True) `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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>a</i>	Longint	Element a of transform matrix
<i>b</i>	Longint	Element b of transform matrix
<i>c</i>	Longint	Element c of transform matrix
<i>d</i>	Longint	Element d of transform matrix
<i>e</i>	Longint	Element e of transform matrix
<i>f</i>	Longint	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>angle</i>	Longint	Angle of rotation
<i>x</i>	Longint	Coordinate on X axis of center of rotation
<i>y</i>	Longint	Coordinate on Y axis of center of rotation

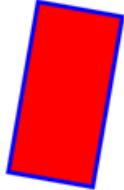
## Description

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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>scaleX</i>	Real	Value on X axis
<i>scaleY</i>	Real	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 (0 < 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 (0 < 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



```
$SVG:= 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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
horizontal	Longint	Value of incline along X axis
vertical	Longint	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



Hello world!

```
$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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>x</i>	Longint	Coordinate on X axis
<i>y</i>	Longint	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>x</i>	Real	X position of viewBox
<i>y</i>	Real	Y position of viewBox
<i>width</i>	Real	Width of viewBox
<i>height</i>	Real	Height of viewBox
<i>mode</i>	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 1 cm = 250 user points
SVG_SET_VIEWBOX( $svg;0;0;1000;2000;"true")
```

## See also

[SVG\\_SET\\_VIEWPORT\\_FILL](#)

## SVG\_SET\_VIEWPORT\_FILL

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>color</i>	String	Fill color
<i>opacity</i>	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>hide</i>	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>x</i>	Longint	Coordinate on X axis
<i>y</i>	Longint	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
```

# Colors and Gradients

---

SVG Colors  
SVG\_Color\_from\_index  
SVG\_Color\_grey  
SVG\_Color\_RGB\_from\_CMYK  
SVG\_Color\_RGB\_from\_HLS  
SVG\_Color\_RGB\_from\_long  
SVG\_FADE\_TO\_GREY\_SCALE  
SVG\_Filter\_ColorMatrix  
SVG\_GET\_COLORS\_ARRAY  
SVG\_GET\_DEFAULT\_BRUSHES  
SVG\_Get\_named\_color\_value  
SVG\_SET\_BRIGHTNESS  
SVG\_SET\_DEFAULT\_BRUSHES  
SVG\_SET\_HUE  
SVG\_SET\_SATURATION

## 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

Format	Example
#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 equivalents, can be found in the [4D SVG Constants theme: SVG Colors \(Names\)](#) and [SVG Colors \(RGB\)](#). You can also view this list and insert the color values directly via the 4D SVG Color palette. For more information, please refer to [Development tools](#).

## None

Pass the "**none**" keyword in the *backgroundColor* or *foregroundColor* parameters in order to not have a fill or stroke at all. The "none" keyword can be used with most SVG commands.

## 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\\_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.

## See also

[Development tools](#)  
[SVG\\_Define\\_linear\\_gradient](#)  
[SVG\\_Define\\_radial\\_gradient](#)  
[SVG\\_Define\\_solidColor](#)  
[SVG\\_SET\\_OPACITY](#)

# SVG\_Color\_from\_index

SVG\_Color\_from\_index ( index ) -> Function result

Parameter	Type		Description
index	Longint	→	Number of color
Function result	Text field	⇒	Color designated by index

## Description

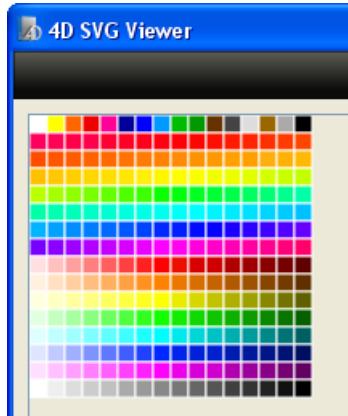
The **SVG\_Color\_from\_index** returns the SVG color matching the 4D color specified in the *index* parameter.

The *index* parameter designates a number in the 4D color palette, where colors are numbered from 1 to 256. For more information about this point, refer to the description of the 4D **OBJECT SET COLOR** command.

## Example

In this example, we recreate the 4D color palette:

```
$Dom_svg:= SVG_New
$Lon_line:=0
For( $Lon_ii;0;15;1)
  $Lon_column:=0
  For( $Lon_i;1;16;1)
    $Txt_color:= SVG_Color_from_index(( $Lon_ii*16)+ $Lon_i)
    $Dom_rect:= SVG_New_rect( $Dom_svg; $Lon_column;
      $Lon_line;11;11;0;0;"white"; $Txt_color)
    $Lon_column:= $Lon_column+11
  End for
  $Lon_line:= $Lon_line+11
End for
SVGTool_SHOW_IN_VIEWER( $Dom_svg)
```



## See also

[SVG\\_Color\\_RGB\\_from\\_long](#)

# SVG\_Color\_grey

SVG\_Color\_grey ( percentage ) -> Function result

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\_CMYK

SVG\_Color\_RGB\_from\_CMYK ( cyan ; magenta ; yellow ; black {; format} ) -> Function result

Parameter	Type	Description
cyan	Longint	Cyan value
magenta	Longint	Magenta value
yellow	Longint	Yellow value
black	Longint	Black value
format	Longint	Color format
Function result	Text	Color string

## Description

The *SVG\_Color\_RGB\_from\_CMYK* command returns a string expressing the color corresponding to the four color parameters, *cyan*, *magenta*, *yellow* and *black*, passed as arguments. The string returned is in the form "RGB(red,greeen,blue)" by default, the syntax recognized by SVG rendering engines.

*cyan*, *magenta*, *yellow* and *black* are longints included between 0 and 100%.

The optional *format* parameter is 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%)

# SVG\_Color\_RGB\_from\_HLS

SVG\_Color\_RGB\_from\_HLS ( hue ; luminosity ; saturation {; format} ) -> Function result

Parameter	Type		Description
hue	Longint	→	Hue value
luminosity	Longint	→	Luminosity value
saturation	Longint	→	Saturation value
format	Longint	→	Color format
Function result	Text	➡	Color string

## Description

---

The *SVG\_Color\_RGB\_from\_HLS* command returns a string expressing the color corresponding to the *hue*, *luminosity* and *saturation* parameters passed as arguments. The string returned is in the form "RGB(red,green,blue)" by default, the syntax recognized by SVG rendering engines.

*hue* is a longint included between 0 and 360°.

*luminosity* and *saturation* are longints included between 0 and 100%.

The optional *format* parameter is 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%)

# SVG\_Color\_RGB\_from\_long

SVG\_Color\_RGB\_from\_long ( color {; format} ) -> Function result

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\\_from\\_index](#)  
[SVG\\_Color\\_grey](#)

## SVG\_FADE\_TO\_GREY\_SCALE

SVG\_FADE\_TO\_GREY\_SCALE ( *svgObject* {; *value*} )

Parameter	Type	Description
svgObject	SVG_Ref	SVG object reference
value	Real	Grey value

### Description

---

The **SVG\_FADE\_TO\_GREY\_SCALE** command applies a filter to transform the gray scale for an SVG image whose reference is passed in the *svgObject* parameter.

You can pass the gray scale value to be applied in the optional *value* parameter. If you do not pass this parameter, the transformation is in accordance with the visual perception of the luminance (30% red, 59% green and 11% blue).

# SVG\_Filter\_ColorMatrix

SVG\_Filter\_ColorMatrix ( *svgObject* {; *in* ; *result*} {; *type* {; *values*} } ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
<i>in</i>	Text	Identifies input for the given filter primitive
<i>result</i>	Text	Provides a reference for the output result of a filter
<i>type</i>	Text	Indicates the type of matrix operation
<i>values</i>	Text	Numeric values for the transformation matrix
Function result	SVG_Ref	Reference for SVG object with new color values

## Description

The **SVG\_Filter\_ColorMatrix** command applies a color matrix transformation to each pixel in the source image passed in the *svgObject* parameter to produce a result with a new set of color values.

In the *in* parameter, you can either pass a string which matches a previous 'result' value or one of the six following keywords:

- *SourceGraphic*: the target element (image, shape, group etc.) that references the filter. This keyword represents the graphics elements that were the original input into the 'filter' element
- *SourceAlpha*: the canvas beneath the *SourceGraphic*. This keyword represents the graphics elements that were the original input into the 'filter' element.
- *BackgroundImage*: the canvas beneath the *SourceGraphic*. This keyword represents an image snapshot of the canvas under the filter region at the time that the 'filter' element was invoked.
- *BackgroundAlpha*: the alpha channel of the canvas beneath the *SourceGraphic*. Same as *BackgroundImage* except only the alpha channel is used
- *FillPaint*: a pseudo-graphic equal to the size of the filter region filled with the fill property of the target element. This keyword represents the value of the 'fill' property on the target element for the filter effect.
- *StrokePaint*: a pseudo-graphic equal to the size of the filter region filled with the stroke property of the target element. This keyword represents the value of the 'stroke' property on the target element for the filter effect.

If no value is passed and this is the first filter primitive, then the *SourceGraphic* is used as its input. If no value is provided and this is a subsequent filter primitive, then this filter primitive will use the result from the previous filter primitive as its input.

In the *result* parameter, you pass a reference for the output result of a filter which can be referenced by the *in* parameter on a subsequent use of this command within the same 'filter' element. If no value is provided, the output is only available for re-use as the implicit input for the next filter primitive if that filter primitive provides no value for its *in* parameter.

In the *type* parameter, you can specify the type of matrix operation by passing one of the following keywords:

- *saturate*: adjusts the saturation of all RGB color channels using a real number value of 0 to 1 that is passed in the *values* parameter.
- *hueRotate*: rotates the pixel hue of all RGB color channels by the angle specified (in degree) in the *values* parameter,
- *luminanceToAlpha*: converts the red, green and blue channels into a luminance value. The RGB channels are set to black (0,0,0).)
- *matrix*: sets the color using the list of values passed in the *values* parameter. Allows the value of each channel in the output to be specified from a combination of its existing color and alpha channels.

If you do not pass a *type* parameter, by default the effect is as if a value of *matrix* were specified.

In the *values* parameter, you pass numeric values based on the keyword passed in the *type* parameter:

- With the " *matrix*" keyword: you pass a list of 20 matrix values, separated by whitespace and/or a comma.
- With the " *saturate*" keyword: you pass a single real number value (0 to 1). The permitted value according to the specification is 0-1, but many browsers accept higher values >1 to allow over-saturation.
- With the " *hueRotate*" keyword: you pass a single real number value (to indicate degrees of rotation).
- With the " *luminanceToAlpha*" keyword: you do not pass a numeric value. The *values* parameter is not used with this type, which discards the alpha channel and replaces it with values equal to the input's luminance.

If you do not pass a *values* parameter, the default behavior depends on the keyword passed in the *type* parameter:

- With the "matrix" keyword: by default the values of the identity matrix are used
- With the "saturate" keyword: by default, the value is 1 (no change).
- With the "hueRotate" keyword: by default the value is 0 (no change).
- With the "luminanceToAlpha" keyword: by default, this parameter is not used.

**Note:** Under Windows, this command requires the prior disabling of Direct2D (see the [Direct2D disabled](#) constant in the description of the **SET DATABASE PARAMETER** command).

## Example



```
C_TEXT( $Dom_filter; $Dom_node; $Dom_rect; $Dom_svg; $Txt_matrix)

SVG_SET_OPTIONS( SVG_Get_options?+5)

$Dom_svg:= SVG_New

$Dom_filter:= SVG_Define_filter( $Dom_svg;"Matrix")
$txt_matrix:=\
".33 .33 .33 0 0 "\
+" .33 .33 .33 0 0 "\
+" .33 .33 .33 0 0 "\
+" .33 .33 .33 0 0"

$Dom_node:= SVG_Filter_ColorMatrix( $Dom_filter;"SourceGraphic";"";"matrix"; $Txt_matrix)

$Dom_filter:= SVG_Define_filter( $Dom_svg;"Saturate")
$Dom_node:= SVG_Filter_ColorMatrix( $Dom_filter;"SourceGraphic";"";"saturate";"1.5")
// another syntax for value
// $Dom_node:=SVG_Filter_ColorMatrix
($Dom_filter;"SourceGraphic";"";"saturate";String(1,5;"&xml"))

$Dom_filter:= SVG_Define_filter( $Dom_svg;"HueRotate90")
$Dom_node:= SVG_Filter_ColorMatrix( $Dom_filter;"SourceGraphic";"";"hueRotate";"90")

$Dom_filter:= SVG_Define_filter( $Dom_svg;"LuminanceToAlpha")
$Dom_node:= SVG_Filter_ColorMatrix( $Dom_filter;"SourceGraphic";"";"luminanceToAlpha")

$Dom_rect:= SVG_New_rect( $Dom_svg;2;0;797;100;0;0;"none";"coral")

$Dom_rect:= SVG_New_rect( $Dom_svg;2;100;797;100;0;0;"none";"coral")
SVG_SET_FILTER( $Dom_rect;"Matrix")

$Dom_rect:= SVG_New_rect( $Dom_svg;2;200;797;100;0;0;"none";"coral")
SVG_SET_FILTER( $Dom_rect;"Saturate")

$Dom_rect:= SVG_New_rect( $Dom_svg;2;300;797;100;0;0;"none";"coral")
SVG_SET_FILTER( $Dom_rect;"HueRotate90")

$Dom_rect:= SVG_New_rect( $Dom_svg;2;400;797;100;0;0;"none";"coral")
SVG_SET_FILTER( $Dom_rect;"LuminanceToAlpha")

SVG_New_text( $Dom_svg;"No filter";110;10;"Verdana";60; bold; -1;"black")
```

```
SVG_New_text( $Dom_svg;"Matrix";110;110;"Verdana";60; Bold; -1;"black")
SVG_New_text( $Dom_svg;"Saturate";110;210;"Verdana";60; Bold; -1;"black")
SVG_New_text( $Dom_svg;"HueRotate";110;310;"Verdana";60; Bold; -1;"black")
SVG_New_text( $Dom_svg;"Luminance";110;410;"Verdana";60; Bold; -1;"black")

//View the result
SVGTool_SHOW_IN_VIEWER( $Dom_svg)

//SVG_SAVE_AS_TEXT($Dom_svg;System folder/Desktop)+"export.svg")

//Don't forget to clear the memory
SVG_CLEAR( $Dom_svg)
```

## SVG\_GET\_COLORS\_ARRAY

SVG\_GET\_COLORS\_ARRAY ( *colorNamesArrayPointer* )

Parameter	Type	Description
<i>colorNamesArrayPointer</i>	Pointer	Pointer to array receiving color names

### Description

---

The *SVG\_GET\_COLORS\_ARRAY* command fills in the array pointed to by the *colorNamesArrayPointer* parameter with the names of colors recognized by SVG.

**See Also:** <http://www.w3.org/TR/SVG/types.html#ColorKeywords>

## SVG\_GET\_DEFAULT\_BRUSHES

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\_Get\_named\_color\_value

SVG\_Get\_named\_color\_value ( colorName {; rgbComponent} ) -> Function result

Parameter	Type	Description
colorName	Text	SVG color name
rgbComponent	Text	"R", "G" or "B" to indicate color component
Function result	Longint	Returns color value

### Description

---

The **SVG\_Get\_named\_color\_value** command returns the value of the SVG color whose name is specified in the *colorName* parameter.

In the optional *rgbComponent* parameter, you can pass either "R" (red), "G" (green) or "B" (blue) to indicate the specific color component for which you want to get the value. If you do not pass this parameter, the command returns the (complete) long color value.

## SVG\_SET\_BRIGHTNESS

SVG\_SET\_BRIGHTNESS ( *svgObject* ; *brightness* {; *brightness2* ; *brightness3*} )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
<i>brightness</i>	Real	Values between 0 and 1 to darken; > 1 to brighten, applied globally or only to red component
<i>brightness2</i>	Real	Brightness value for green component
<i>brightness3</i>	Real	Brightness value for blue component

### Description

---

The **SVG\_SET\_BRIGHTNESS** command sets the brightness for an SVG image or container whose reference is passed in the *svgObject* parameter.

In the *brightness* parameter, you pass either a value between 0 and 1 to darken the brightness, or a value greater than 1 to brighten it. When you pass a single *brightness* parameter, the brightness factor is applied globally to the object.

Alternatively, you can pass two additional (optional) *brightness2* and *brightness3* parameters, in which case each brightness value is applied to a separate part of the color component, i.e. *brightness* is applied to the "R" (red) part, *brightness2* is applied to the "G" (green) part and *brightness3* is applied to the "B" (blue) part.

# SVG\_SET\_DEFAULT\_BRUSHES

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

Parameter	Type	Description
line	String	Color
background	String	Color

## Description

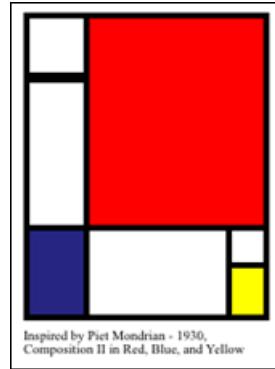
The `SVG_SET_DEFAULT_BRUSHES` command can be used to set the default colors used by the component.

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](#)

# SVG\_SET\_HUE

SVG\_SET\_HUE ( *svgObject* ; *hue* )

Parameter	Type		Description
<i>svgObject</i>	SVG_Ref	→	SVG object reference
<i>hue</i>	Longint	→	Hue value

## Description

---

The **SVG\_SET\_HUE** method sets a hue value for the SVG object designated by the *svgObject* parameter. *svgObject* must be an SVG container (svg, group, symbol, pattern, marker, etc.) or an image; otherwise, an error is returned. In the *hue* parameter, you pass a value between 0 and 360.

## See also

[SVG\\_SET\\_SATURATION](#)

# SVG\_SET\_SATURATION

SVG\_SET\_SATURATION ( *svgObject* ; *saturation* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
<i>saturation</i>	Longint	Saturation value

## Description

---

The **SVG\_SET\_SATURATION** method sets a saturation value for the SVG element designated by the *svgObject* parameter. *svgObject* must be an SVG container (svg, group, symbol, pattern, marker, etc.) or an image; otherwise, an error is returned.

In the *saturation* parameter, you pass a value between 0 and 100

## See also

[SVG\\_SET\\_HUE](#)

# Documents

---

SVG\_CLEAR  
SVG\_Copy  
SVG\_Export\_to\_picture  
SVG\_Export\_to\_XML  
SVG\_New  
SVG\_Open\_file  
SVG\_Open\_picture  
SVG\_SAVE\_AS\_PICTURE  
SVG\_SAVE\_AS\_TEXT  
SVG\_SET\_DOCUMENT\_VARIABLE  
SVG\_Validate\_file

# SVG\_CLEAR

SVG\_CLEAR {{ *svgObject* }}

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference

## Description

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

*svgObject* can be an SVG root object (created with the [SVG\\_New](#), [SVG\\_Copy](#) or [SVG\\_Open\\_file](#) commands) or any valid SVG object.

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

SVG\_Copy ( *svgObject* ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	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

SVG\_Export\_to\_picture ( *svgObject* {; *exportType*} ) -> Function result

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

SVG\_Export\_to\_XML ( *svgObject* ) -> Function result

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

SVG\_New {{ width ; height {; title {; description {; rectangle {; display}}}}} } ) -> Function result

Parameter	Type	Description
width	Longint	Document width
height	Longint	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

SVG\_Open\_file ( path ) -> Function result

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

SVG\_Open\_picture ( picture ) -> Function result

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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
<i>document</i>	String	Document name or Full pathname of document
<i>codec</i>	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.

The **SVG\_SAVE\_AS\_PICTURE** modifies the value of the variable (if any) designated by the **SVG\_SET\_DOCUMENT\_VARIABLE** command.

### 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\\_SET\\_DOCUMENT\\_VARIABLE](#)

## SVG\_SAVE\_AS\_TEXT

SVG\_SAVE\_AS\_TEXT ( *svgObject* {; *document*} )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
<i>document</i>	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* or omit this parameter, 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 **SVG\_SAVE\_AS\_TEXT** modifies the value of the variable (if any) designated by the **SVG\_SET\_DOCUMENT\_VARIABLE** command.

### 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\\_SET\\_DOCUMENT\\_VARIABLE](#)

## SVG\_SET\_DOCUMENT\_VARIABLE

SVG\_SET\_DOCUMENT\_VARIABLE ( pointer )

Parameter	Type	Description
pointer	Pointer	Pointer to variable to set

### Description

The **SVG\_SET\_DOCUMENT\_VARIABLE** method sets a pointer to the variable of the host database that is updated after each call to [SVG\\_SAVE\\_AS\\_PICTURE](#) or [SVG\\_SAVE\\_AS\\_TEXT](#). You must call this method only once per session (in an initialization method, for instance).

In the *pointer* parameter, you pass a pointer to the variable whose value you want to follow (usually the **Document** system variable).

You can pass a Nil pointer in the *pointer* parameter to remove the link.

### See also

[SVG\\_SAVE\\_AS\\_PICTURE](#)  
[SVG\\_SAVE\\_AS\\_TEXT](#)

## SVG\_Validate\_file

SVG\_Validate\_file ( path ) -> Function result

Parameter	Type	Description
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.

# Drawing

---

SVG\_Add\_object  
SVG\_ADD\_POINT  
SVG\_New\_arc  
SVG\_New\_circle  
SVG\_New\_ellipse  
SVG\_New\_ellipse\_bounded  
SVG\_New\_embedded\_image  
SVG\_New\_image  
SVG\_New\_line  
SVG\_New\_path  
SVG\_New\_polygon  
SVG\_New\_polygon\_by\_arrays  
SVG\_New\_polyline  
SVG\_New\_polyline\_by\_arrays  
SVG\_New\_rect  
SVG\_New\_regular\_polygon  
SVG\_PATH\_ARC  
SVG\_PATH\_CLOSE  
SVG\_PATH\_CURVE  
SVG\_PATH\_LINE\_TO  
SVG\_PATH\_MOVE\_TO  
SVG\_PATH\_QCURVE  
SVG\_Use

# SVG\_Add\_object

SVG\_Add\_object ( targetSVGObject ; sourceSVGObject ) -> Function result

Parameter	Type	Description
targetSVGObject	SVG_Ref	Reference of parent element
sourceSVGObject	SVG_Ref	Reference of object to be added
Function result	SVG_Ref	Reference of SVG object

## Description

---

The *SVG\_Add\_object* command is used to place an SVG object designated by *sourceSVGObject* in the SVG container designated by *targetSVGObject* and return its reference. The SVG container can be the root of the document or any other reference to an SVG object that is able to contain this type of element.

# SVG\_ADD\_POINT

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

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
x	Longint	Coordinate on X axis of new point(s)
y	Longint	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\\_polyline](#) command.

## See also

[SVG\\_New\\_path](#)  
[SVG\\_New\\_polygon](#)  
[SVG\\_New\\_polyline](#)

# SVG\_New\_arc

SVG\_New\_arc ( parentSVGObject ; x ; y ; radius ; start ; end {; foregroundColor {; backgroundColor {; strokeWidth}}})  
-> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
x	Longint	Coordinate on center X axis
y	Longint	Coordinate on center Y axis
radius	Longint	Radius of circle
start	Longint	Value in degrees of start of arc
end	Longint	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

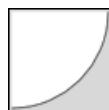
The `SVG_New_arc` command creates a new circle arc in the SVG container designated by `parentSVGObject` and returns its reference. If `parentSVGObject` is not an SVG document, an error is generated.

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.

## Example 1

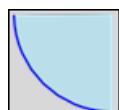
Draw an arc from 0° to 90° (default fill and border color, default line thickness):



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

## Example 2

Draw the arc from 90° to 180° 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

SVG\_New\_circle ( parentSVGObject ; x ; y ; radius {; foregroundColor {; backgroundColor {; strokeWidth}}} ) ->  
Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
x	Longint	Coordinate on center X axis
y	Longint	Coordinate on center Y axis
radius	Longint	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

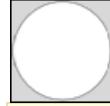
The *SVG\_New\_circle* command creates a new circle in the SVG container designated by *parentSVGObject* and returns its reference. If *parentSVGObject* is not an SVG document, an error is generated.

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.

## Example 1

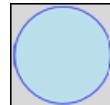
Draw a circle (default fill and border color, default line thickness):



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

## Example 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

SVG\_New\_ellipse ( parentSVGObject ; x ; y ; xRadius ; yRadius {; foregroundColor {; backgroundColor {; strokeWidth} } } ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
x	Longint	Coordinate on center X axis of ellipse
y	Longint	Coordinate on center Y axis of ellipse
xRadius	Longint	Radius on X axis
yRadius	Longint	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.

## Example 1

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



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

## Example 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

SVG\_New\_ellipse\_bounded ( parentSVGObject ; x ; y ; width ; height {; foregroundColor {; backgroundColor {; strokeWidth} } } ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
x	Longint	Coordinate on X axis of upper left corner
y	Longint	Coordinate on Y axis of upper left corner
width	Longint	Width of bounding rectangle
height	Longint	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.

## Example 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)
```

## Example 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

SVG\_New\_embedded\_image ( parentSVGObject ; picture {; x {; y}}{; codec} ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
picture	Picture	Picture to be embedded
x	Longint	Coordinate on X axis of upper left corner
y	Longint	Coordinate on Y axis of upper left corner
codec	Text	Codec to use
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).

The optional *codec* parameter sets the codec to use for the *picture*. By default, if this parameter is omitted, the codec is ".png".

## Example

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



```
svgRef:= SVG_New
$Path:= Get 4D folder( Current 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

SVG\_New\_image ( parentSVGObject ; url {; x ; y {; width ; height}} ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
url	String	Address of picture
x	Longint	Coordinate on X axis of upper left corner
y	Longint	Coordinate on Y axis of upper left corner
width	Longint	Width of picture
height	Longint	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.
- A **relative URL** ("..../picture.png"). This is particularly useful in client/server mode, when files are stored in the "Resources" folder. Relatives URLs can begin with:
  - "/" , to indicate the "~/Resources/SVG/" path
  - "./" , to indicate the "~/Resources/" path
  - "../" , to indicate the database folder.

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.

## Example 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" )
```

## Example 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)
```

## Example 3

Here are a few examples for accessing pictures using relative URLs:

```
SVG_New_image( $Dom_svg;"./images/picture.png";10;10)  
// base is the "Resources" folder  
// XML code will be xlink:href="../images/picture.png"
```

```
SVG_New_image( $Dom_svg;"../picture.png";70;180)  
// base is the database's folder  
// XML code will be xlink:href=".../..../picture.png"
```

```
SVG_New_image( $Dom_svg;"/sample pictures/picture.png";110;90;100;100)  
// base is the "SVG" folder in the "Resources" folder  
// XML code will be xlink:href="sample%20pictures/picture.gif"
```

## See also

[SVG\\_New\\_embedded\\_image](#)  
[SVG\\_Open\\_picture](#)

# SVG\_New\_line

SVG\_New\_line ( parentSVGObject ; startX ; startY ; endX ; endY {; color {; strokeWidth} } ) -> Function result

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

## Description

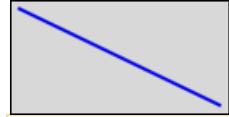
The `SVG_New_line` command creates a new line in the SVG container designated by `parentSVGObject` and returns its reference. The object is positioned according to the `startX`, `startY`, `endX` and `endY` coordinates. The SVG container can be the document root or any other reference to an SVG object 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

SVG\_New\_path ( parentSVGObject ; x ; y {; foregroundColor {; backgroundColor {; strokeWidth}}} ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
x	Longint	Coordinate on X axis of start of path
y	Longint	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.

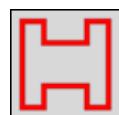
The *x* and *y* parameters can be used to specify the start position of a path in the SVG container.

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.

## Example 1

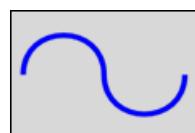
Draw a closed broken line:



```
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)
```

## Example 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)
```

## Example 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
```

## Example 4

Complex path (cubic Bezier curve):

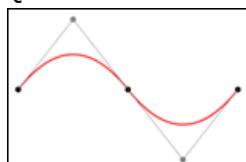


```
`Create a new SVG tree
$txt_svg:= SVG_New(174,96;125,04;"4D Logo";""; True)

`Create a new path
$txt_path:= SVG_New_path( $txt_svg;150,665;13,021)
`Set colors
SVG_SET_STROKE_BRUSH( $txt_path;"#212a6f")
SVG_SET_FILL_BRUSH( $txt_path;"#212a6f")
...
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)
```

## Example 5

Quadratic Bezier curve:



```
`Create a new SVG tree
$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

SVG\_New\_polygon ( parentSVGObject {; points {; foregroundColor {; backgroundColor {; strokeWidth}}} } ) ->

Function result

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

SVG\_New\_polygon\_by\_arrays ( parentSVGObject ; xArrayPointer ; yArrayPointer {; foregroundColor {; backgroundColor {; strokeWidth}}}) -> Function result

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.

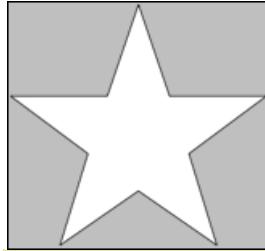
All the coordinate values are in the user coordinate system.

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

SVG\_New\_polyline ( parentSVGObject {; points {; foregroundColor {; backgroundColor {; strokeWidth}}} } ) ->  
Function result

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

SVG\_New\_polyline\_by\_arrays ( parentSVGObject ; xArrayPointer ; yArrayPointer {; foregroundColor {; backgroundColor {; strokeWidth}}}) -> Function result

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.

## Example 1

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



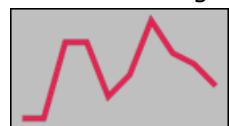
```
ARRAY LONGINT( $tx;0)
ARRAY LONGINT( $ty;0)

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)
```

## Example 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

SVG\_New\_rect ( parentSVGObject ; x ; y ; width ; height {; roundedX {; roundedY {; foregroundColor {; backgroundColor {; strokeWidth}}}} } ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
x	Longint	X of upper left corner
y	Longint	Y of upper left corner
width	Longint	Width of rectangle
height	Longint	Height of rectangle
roundedX	Longint	Horizontal curve
roundedY	Longint	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.

## Example 1

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



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

## Example 2

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



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

## Example 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)
```

## Example 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

SVG\_New\_regular\_polygon ( parentSVGObject ; width ; number {; x {; y {; foregroundColor {; backgroundColor {; strokeWidth}}} } } ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
width	Longint	Diameter of surrounding circle
number	Longint	Number of sides
x	Longint	Coordinate on center X axis
y	Longint	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.

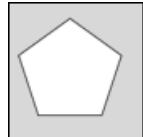
The optional *x* and *y* parameters can be used to specify the center of the circle. If they are omitted, the figure will be drawn in the upper left corner of the document.

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.

## Example 1

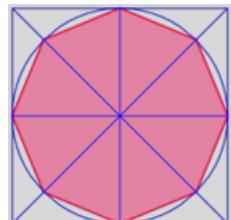
Draw a pentagon (default fill and border color, default line thickness):



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

## Example 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

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

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of path element
xRadius	Longint	Radius of ellipse on X axis
yRadius	Longint	Radius of ellipse on Y axis
x	Longint	Coordinate on X axis of destination point
y	Longint	Coordinate on Y axis of destination point
rotation	Longint	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.

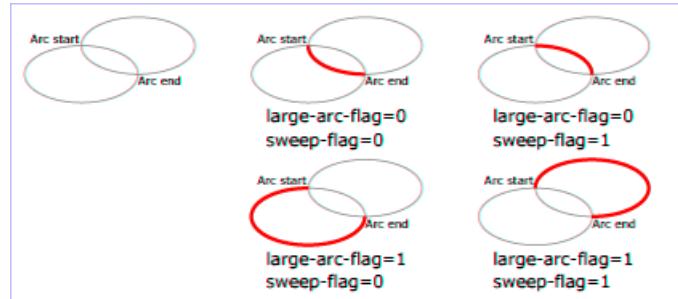
The optional *arcpath* parameter can be used to apply a combination of constraints which will determine how the arc will be drawn. The large-arc-flag constraint is used to choose (or not) the larger of the two possible arcs (greater than 180°) and the sweep-flag constraint chooses the direction it will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

SVG\_PATH\_CLOSE ( *parentSVGObject* )

Parameter	Type	Description
<i>parentSVGObject</i>	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

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

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

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

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

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

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of path
x	Longint	Coordinate on X axis
y	Longint	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

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

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

SVG\_Use ( parentSVGObjec ; id {; x ; y ; width ; height {; mode}} ) -> Function result

Parameter	Type	Description
parentSVGObjec	SVG_Ref	Reference of parent element
id	String	Name of symbol
x	Longint	X position of viewbox
y	Longint	Y position of viewbox
width	Longint	Width of viewbox
height	Longint	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.

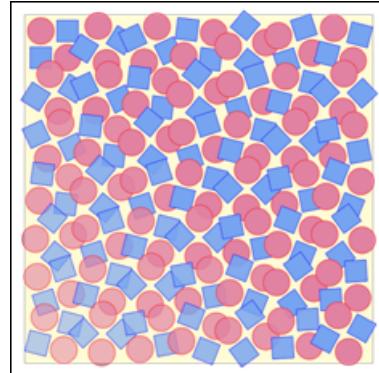
The *id* parameter specifies the name of the symbol.

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](#)

# Filters

---

SVG Filters  
SVG\_Filter\_Blend  
SVG\_Filter\_Blu  
SVG\_Filter\_Offset

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".

## See also

[SVG\\_Define\\_filter](#)  
[SVG\\_SET\\_FILTER](#)

# SVG\_Filter\_Bland

SVG\_Filter\_Bland ( filterRef ; picture ; backgroundPict {; mode {; name}} ) -> Function result

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\_Bland* 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", "darker" or "lighten".

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

**Note:** Starting with 4D v14 R5, this command works under Windows with Direct2D enabled in software mode (see the Direct2D software constant in the description of the **SET DATABASE PARAMETER** command).

## Example

In a form, we display two identical SVG pictures then we create a "blend" filter and assign to the one on the right. This filter is a combination of the "offset" and "blur" filters:

```
$root:= SVG_New(400;400;"filters test") //definition of first (left) picture
$rect:= SVG_New_rect( $root;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH( $root;"orange")
$textAreaRef:= SVG_New_textArea( $root;"Hello World!";10;10;380;100;"arial";60;Normal; Align_center)
<>pict1:= SVG_Export_to_picture( $root) //display first picture

$root2:= SVG_New(400;400;"filters test") //definition of identical (right) picture

//create filter
$filter:= SVG_Define_filter( $root2;"MyShadow")
$vGraph:= True //applied on graphic layer - pass False to apply to the alpha layer
If( $vGraph)
    $ref1:= SVG_Filter_Blu( $filter;2;"sourceGraphic";"blurResult") //"blurResult" will be used as the "input" of the offset filter
Else
    $ref1:= SVG_Filter_Blu( $filter;2;"sourceAlpha";"blurResult") //"blurResult" will be used as the "input" of the offset filter
End_if
//Adding offset filter
$ref2:= SVG_Filter_Offset( $filter;5;5;"blurResult";"alphaBlurOffset")
//Adding blend filter
$ref3:= SVG_Filter_Bland( $filter;"sourceGraphic";"alphaBlurOffset";"normal";"finalFilter");

$rect2:= SVG_New_rect( $root2;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH( $root2;"orange")
$textAreaRef2:= SVG_New_textArea( $root2;"Hello World!";10;10;380;100;"arial";60;Normal; Align_center)

SVG_SET_FILTER( $textAreaRef2;"MyShadow") //apply final filter
<>pict2:= SVG_Export_to_picture( $root2) //display second picture
```

Result (blur input filter = sourceGraphic):

pict1

pict2

Hello World!

Hello World!

Result (blur input filter = sourceAlpha):

pict1

pict2

Hello World!

Hello World!

## See also

[SVG\\_Filter Blur](#)  
[SVG\\_Filter Offset](#)

# SVG\_Filter Blur

SVG\_Filter Blur ( filterRef ; deviation {; input {; name}} ) -> Function result

Parameter	Type	Description
filterRef	SVG_Ref	Reference of filter
deviation	Real	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. You can pass:

- either "sourceGraphic", indicating that the graphic is the filter source (default),
- or "sourceAlpha", which indicates that the alpha channel is the filter source.

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

**Note:** Starting with 4D v14 R5, this command works under Windows with Direct2D enabled in software mode (see the Direct2D software constant in the description of the **SET DATABASE PARAMETER** command).

## Example

In a form, we display two identical SVG pictures then we create a "blur" filter and assign to the one on the right:

```
$root:= SVG_New(400;400;"filters test") //definition of first (left) picture
$rect:= SVG_New_rect( $root;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH( $root;"orange")
$textAreaRef:= SVG_New_textArea( $root;"Hello World!";10;10;380;100;"arial";60;Normal; Align center)
<>pict1:= SVG_Export_to_picture( $root) //display first picture

$root2:= SVG_New(400;400;"filters test") //definition of identical (right) picture

//create filter
$filter1:= SVG_Define_filter( $root2;"blur")
// filter definition
$vgGraph:= True //applied on graphic layer - pass False to apply to the alpha layer
>If( $vgGraph)
    SVG_Filter Blur( $filter1;Deviation{Deviation};"sourceGraphic")
Else
    SVG_Filter Blur( $filter1;Deviation{Deviation};"sourceAlpha")
End if

$rect2:= SVG_New_rect( $root2;10;10;380;100;0;0;"darkblue";"white";1) //definition of identical (right) picture
SVG_SET_FILL_BRUSH( $root2;"orange")
$textAreaRef2:= SVG_New_textArea( $root2;"Hello World!";10;10;380;100;"arial";60;Normal; Align center)

SVG_SET_FILTER( $textAreaRef2;"blur") //apply filter
<>pict2:= SVG_Export_to_picture( $root2) //display second picture
```

Result (input = sourceGraphic):

pict1

pict2

Hello World!

Hello World!

Result (input = sourceAlpha):

pict1

pict2

Hello World!

Hello World!

## See also

[SVG\\_Filter\\_Blend](#)  
[SVG\\_Filter\\_Offset](#)

## SVG\_Filter\_Offset

SVG\_Filter\_Offset ( filterRef ; dx {; dy {; input {; name}}} ) -> Function result

Parameter	Type	Description
filterRef	SVG_Ref	Reference of filter
dx	Longint	Offset on X axis
dy	Longint	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. You can pass:

- either "sourceGraphic", indicating that the graphic is the filter source (default),
- or "sourceAlpha", which indicates that the alpha channel is the filter source.

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

**Note:** Starting with 4D v14 R5, this command works under Windows with Direct2D enabled in software mode (see the [Direct2D software](#) constant in the description of the **SET DATABASE PARAMETER** command).

## Example

In a form, we display two identical SVG pictures then we create an "offset" filter and assign to the one on the right:

```
$root:= SVG_New(400;400;"filters test") //definition of first (left) picture
$rect:= SVG_New_rect( $root;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH( $root;"orange")
$TextAreaRef:= SVG_New_textArea( $root;"Hello World!";10;10;380;100;"arial";60;Normal; Align_center)
<>pict1:= SVG_Export_to_picture( $root) //display first picture

$root2:= SVG_New(400;400;"filters test") //definition of identical (right) picture
$rect2:= SVG_New_rect( $root2;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH( $root2;"orange")
$TextAreaRef2:= SVG_New_textArea( $root2;"Hello World!";10;10;380;100;"arial";60;Normal; Align_center)

$filter:= SVG_Define_filter( $root2;"Offset") //create filter
SVG_Filter_Offset( $filter;10;20)
SVG_SET_FILTER( $TextAreaRef2;"Offset") //apply filter

<>pict2:= SVG_Export_to_picture( $root2) //display second picture
```

Result:



## See also



# Structure and Definitions

---

SVG\_Define\_clip\_path  
SVG\_Define\_filter  
SVG\_Define\_linear\_gradient  
SVG\_Define\_marker  
SVG\_Define\_pattern  
SVG\_Define\_radial\_gradient  
SVG\_Define\_shadow  
SVG\_Define\_solidColor  
SVG\_Define\_style  
SVG\_DEFINE\_STYLE\_WITH\_ARRAYS  
SVG\_Define\_symbol  
SVG\_DELETE\_OBJECT  
SVG\_Get\_default\_encoding  
SVG\_New\_group  
SVG\_SET\_DEFAULT\_ENCODING  
SVG\_Set\_description  
SVG\_SET\_PATTERN\_CONTENT\_UNITS  
SVG\_SET\_PATTERN\_UNITS  
SVG\_Set\_title

## SVG\_Define\_clip\_path

SVG\_Define\_clip\_path ( parentSVGObject ; clipPathID ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
clipPathID	Text	Name of clip path
Function result	SVG_Ref	Reference of clip path

## Description

The *SVG\_Define\_clip\_path* command specifies a new clip path in the SVG container designated by *parentSVGObject* and returns its reference. If *parentSVGObject* is not (or does not belong to) an SVG document, an error is generated.

The *clipPathID* parameter designates the name of the clip path. This name will be used to associate a clip path with an object. If an element with the same name already exists in the document, an error is generated.

**See Also:** <http://www.w3.org/TR/2001/REC-SVG-20010904/masking.html#EstablishingANewClippingPath>

## Example

Refer to the example of the *SVG\_SET\_CLIP\_PATH* command.

## See also

[SVG\\_SET\\_CLIP\\_PATH](#)

## SVG\_Define\_filter

```
SVG_Define_filter( parentSVGObject ; id {; frameX ; frameY {; frameWidth ; frameHeight {; frameUnit ; filterUnit}}})  
)-> Function result
```

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
id	String	Name of symbol
frameX	Longint	Coordinate on X axis
frameY	Longint	Coordinate on Y axis
frameWidth	Longint	Width of target rectangle
frameHeight	Longint	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*".

## Example

In this example, we want to perform the following operations:

- create a rectangle with 50% blue background
- create a 4% blur filter and apply it to this rectangle
- save the result in an SVG file on disk.

```
$Dom_SVG:= SVG_New  
  
//creation of a rectangle with 50% blue background  
$Dom_rect:= SVG_New_rect( $Dom_SVG;50;50;50;50;0;0;"blue:50";"blue:50")  
  
//creation of 4% blur filter  
$Dom_filter:= SVG_Define_filter( $Dom_SVG;"blur")  
SVG_Filter_Blr( $Dom_filter;4)  
SVG_Filter_Offset( $Dom_filter;4)  
  
//application of this filter to the rectangle  
SVG_SET_FILTER( $Dom_rect;"blur")  
  
//saving result in an SVG file  
SVG_SAVE_AS_TEXT( $Dom_SVG; System folder( Desktop)+"test.svg")  
  
SVG_CLEAR( $Dom_SVG)
```

Result:



## See also

[SVG\\_Filter\\_Blend](#)

[SVG\\_Filter Blur](#)

[SVG\\_Filter\\_Offset](#)

[SVG\\_SET\\_FILTER](#)

# SVG\_Define\_linear\_gradient

SVG\_Define\_linear\_gradient ( parentSVGObject ; id ; startColor ; endColor {; rotation {; spreadMethod {; x1 ; y1 ; x2 ; y2} {; startColorOffset ; endColorOffset}} } ) -> Function result

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
spreadMethod	Text	Gradient spread method (pad, reflect or repeat)
x1	Real	x1 coordinate of gradient vector (If omitted = 0)
y1	Real	y1 coordinate of gradient vector (If omitted = 1)
x2	Real	x2 coordinate of gradient vector (If omitted = 0)
y2	Real	y2 coordinate of gradient vector (If omitted = 1)
startColorOffset	Real	Percentage value of offset for start color
endColorOffset	Real	Percentage value of offset for end color
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.

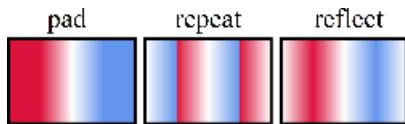
The *id* parameter specifies the name of the gradient. If an element with the same name exists, it will be replaced. This is the name that will be used to call the gradient each time a color expression is expected by using the syntax "url(#ID)".

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).

The optional *spreadMethod* parameter defines the filling to be used when the gradient begins or ends within the bounds of the *parentSVGObject*. You can pass one of the following strings in this parameter:

- "pad": use the terminal colors of the gradient to fill the remainder of the area.
- "reflect": reflect the gradient pattern start-to-end, end-to-start, start-to-end, etc. continuously until the object is filled.
- "repeat": repeat the gradient pattern start-to-end, start-to-end, start-to-end, etc. continuously until the object is filled.



If this parameter is omitted, the "pad" value effect is used.

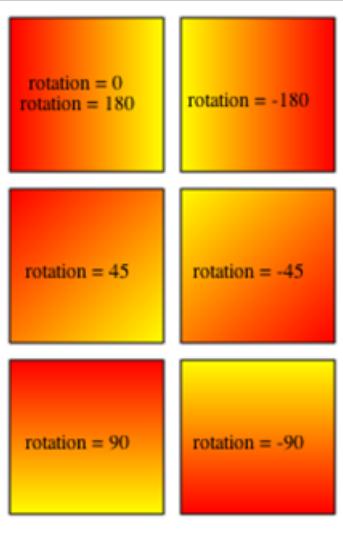
The optional *x1*, *y1*, *x2* and *y2* parameters define the gradient vector. This vector provides the starting and ending points used by the rendering engine. You can pass percentages expressed as ratios (0...1) in these parameters.

Starting with V14, you can pass the optional *startColorOffset* and *endColorOffset* parameters to define the percentage value, respectively, of the start color and end color offset. You can pass either a real value < 1, or a value between 0 and 100 to set the percentage, i.e. "0.1" and/or "10" are both interpreted as 10%.

Passing a negative value is interpreted as 0% for the *startColorOffset* parameter and as 100% for the *endColorOffset* parameter. If you pass a value > 100, it is interpreted as 100%.

## Example 1

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; Align 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; Align 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; Align 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; Align 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; Align 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; Align center)

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

## Example 2

Example using the `startColorOffset` and `endColorOffset` parameters (added in v14):

```
$Dom_node:= SVG_Define_linear_gradient( $Dom_svg;"clicked";"black:50";"black:20";-90;"reflect";0;80)
```

will give the definition:

```
<linearGradient id="clicked" spreadMethod="reflect" x1="0" x2="0" y1="1" y2="0"> <stop offset="0%" stop-color="black" stop-opacity="0.5"/> <stop offset="80%" stop-color="black" stop-opacity="0.2"/> </linearGradient>
```

## Example 3

This example illustrates the effect of the `startColorOffset` and `endColorOffset` parameters:

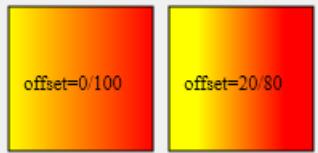
```

$svg := SVG_New

SVG_Define_linear_gradient( $svg;"demoGradient_1";"red";"yellow";-180;"reflect")
SVG_New_rect( $svg;10;10;90;90;0;0;"black";"url(#demoGradient_1)")
SVG_New_text( $svg;"offset=0/100";50;50;"";-1;-1; Align center)

SVG_Define_linear_gradient( $svg;"demoGradient_2";"red";"yellow";-180;"reflect";20;80)
SVG_New_rect( $svg;110;10;90;90;0;0;"black";"url(#demoGradient_2)")
SVG_New_text( $svg;"offset=20/80";150;50;"";-1;-1; Align center)
//Save document
SVG_SAVE_AS_TEXT( $svg;"test2.svg")
//Free up memory
SVG_CLEAR( $svg)

```



## See also

[SVG Colors](#)  
[SVG\\_Define\\_radial\\_gradient](#)

# SVG\_Define\_marker

SVG\_Define\_marker ( parentSVGObject ; id {; x ; y {; width ; height {; orientation}}} ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
id	String	Name of symbol
x	Longint	Coordinate on X axis of reference point
y	Longint	Coordinate on Y axis of reference point
width	Longint	Width of marker
height	Longint	Height of marker
orientation	Longint	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\_pattern

SVG\_Define\_pattern ( parentSVGObject ; patternID {; width {; height {; x {; y {; unit {; viewBox}}}}} } ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
patternID	Text	Name of pattern
width	Longint	Width of pattern
height	Longint	Height of pattern
x	Longint	Position X of pattern
y	Longint	Position Y of pattern
unit	Text	Unit of lengths and positions
viewBox	Text	Viewbox rectangle
Function result	SVG_Ref	Reference of pattern

## Description

The *SVG\_Define\_pattern* command is used to set a new custom pattern in the SVG container designated by *parentSVGObject* and returns its reference. If *parentSVGObject* is not (or does not belong to) an SVG document, an error is generated.

The *patternID* parameter specifies the name of the pattern. This name will be used to associate the pattern with an object. If an element with the same name already exists, an error is generated.

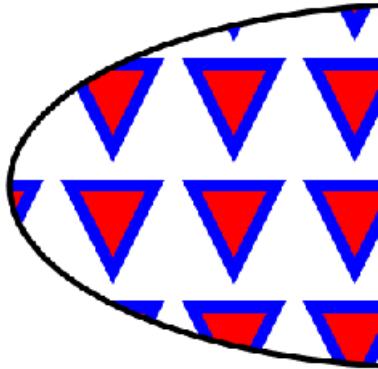
The optional *width*, *height*, *x*, *y*, *unit* and *viewBox* parameters define the reference rectangle of the pattern, in other words, the way the pattern tiles will be placed and spaced.

The pattern will be associated as fill or stroke paint by passing the "url(#id)" string as the value when a color expression is expected.

See Also: <http://www.w3.org/TR/SVG/patterns.html#Patterns>

## Example 1

Setting a pattern and using it to fill an ellipse:



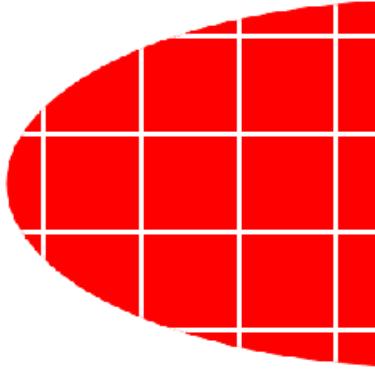
```
//Definition of pattern
$Dom_pattern:= SVG_Define_pattern( $Dom_SVG;"MyPattern";100;100;0;0;" ";"0 0 10 10")
$Dom_path:= SVG_New_path( $Dom_pattern;0;0)

SVG_PATH_MOVE_TO( $Dom_path;0;0)
SVG_PATH_LINE_TO( $Dom_path;7;0)
SVG_PATH_LINE_TO( $Dom_path;3,5;7)
SVG_PATH_CLOSE( $Dom_path)
SVG_SET_FILL_BRUSH( $Dom_path;"red")
SVG_SET_STROKE_BRUSH( $Dom_path;"blue")

//Drawing an ellipse filled with the pattern
$Dom_ellipse:= SVG_New_ellipse( $Dom_SVG;400;200;350;150;"black";"url(#MyPattern)";5)
```

## Example 2

Setting a pattern and using it to fill and stroke the outline of an ellipse:



```
//Definition of pattern
$Dom_pattern:= SVG_Define_pattern( $Dom_SVG;"MyPattern ";80;80;0;0;"";"0 0 20 20")
$Dom_rect:= SVG_New_rect( $Dom_pattern;0;0;20;20;0;0;"white";"red")

//Drawing an ellipse
$Dom_ellipse:= SVG_New_ellipse( $Dom_SVG;400;200;350;150)

//Using pattern for filling and outline
SVG_SET_FILL_BRUSH( $Dom_ellipse;"url(#MyPattern)")
SVG_SET_STROKE_BRUSH( $Dom_ellipse;"url(#MyPattern)")
```

# SVG\_Define\_radial\_gradient

SVG\_Define\_radial\_gradient ( parentSVGObject ; id ; startColor ; endColor {; cx ; cy ; r {; fx ; fy}} ) -> Function result

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
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.

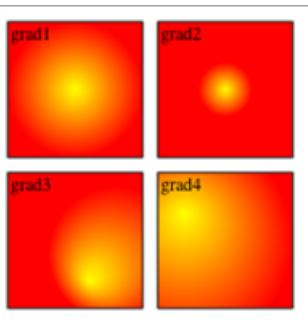
The *id* parameter specifies the name of the gradient. If an element with the same name exists, it will be replaced. This is the name that will be used to call the gradient each time a that a color expression is expected by using the syntax "url(#ID)".

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

SVG\_Define\_shadow ( parentSVGObject ; id {; deviation {; offsetX {; offsetY}}}) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
id	String	Name of filter
deviation	Longint	Value of shadow dispersion
offsetX	Longint	Offset on X axis
offsetY	Longint	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

SVG\_Define\_solidColor ( parentSVGObject ; id ; color {; opacity} ) -> Function result

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\_style

SVG\_Define\_style ( parentSVGObject ; style {; type {; media}} ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
style	Text	Definition of style OR Pathname of file to use
type	Text	Type of content
media	Text	Media descriptor
Function result	SVG_Ref	Reference of style

## Description

The *SVG\_Define\_style* command is used to set a new style sheet in the SVG container designated by *parentSVGObject* and returns its reference. If *parentSVGObject* is not (or does not belong to) an SVG document, an error is generated.

The *style* parameter is used to embed style sheets directly within SVG content:

- If the *style* parameter contains a valid pathname to a CSS file, the style definition is done using a mechanism referencing external style sheets. The path, if it begins with the # character or by the string "file:", expresses a relative path whose root is the "Resources" folder of the database.
- The *style* parameter can also be a URL of the "http://..." type; in this case, the style sheet will be referenced as an external resource.
- In addition, the *style* parameter can be a URL relative to the "Resources/SVG/" subfolder of the database. This is particularly useful in client/server mode, when files are stored in the "Resources" folder. Relatives URLs can begin with:
  - "/" , to indicate the "~/Resources/SVG/" path
  - "./" , to indicate the "~/Resources/" path
  - "../" , to indicate the database folder.

For examples of relative URLs, refer to the [SVG\\_New\\_image](#) command.

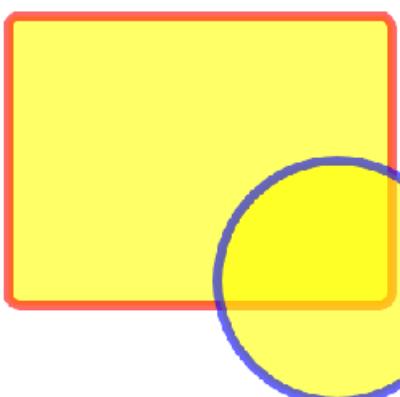
The optional *type* parameter specifies the language of the style sheet for the contents of the element. The default value is "text/css".

The optional *media* parameter specifies the intended destination media for the style information. If you omit this parameter, the default value used is "all". If the value is not included in the list of media types recognized by CSS2, an error is generated.

See Also: <http://www.w3.org/TR/SVG/styling.html#StyleElement>

## Example 1

Setting an embedded style and overlay of one of the elements:



```
//Definition of style
$txt_style:=".colored {fill: yellow; fill-opacity: 0.6; stroke: red;stroke-width:8; stroke-
opacity: 0.6}"
SVG_Define_style( $Dom_SVG; $txt_style)

//Creating a group and assigning a default style
$Dom_g:= SVG_New_group( $Dom_SVG)
SVG_Set_Class( $Dom_g;"colored")
```

```

//Drawing a rectangle
$Dom_rect:= SVG_New_rect( $Dom_g;25;30;320;240;10;10;"");"")
//Drawing a circle and style overlay with a custom outline color
$Dom_circle:= SVG_New_circle( $Dom_g;300;250;100;"");"")
SVG_Set_Stroke_Brush( $Dom_circle;"blue")

```

## Example 2

Referencing the "mystyle.css" file placed in the "dev" folder of the "Resources" folder:



```

//Definition of style
SVG_Define_style( $Dom_svg;"#dev/mystyle.css")

//Creating a group and assigning a default style
$Dom_g:= SVG_New_group( $Dom_SVG)
SVG_Set_Class( $Dom_g;"colored")

//Drawing a rectangle
$Dom_rect:= SVG_New_rect( $Dom_g;25;30;320;240;10;10;"");"")

```

mystyle.css file:

```
.colored {fill: red; fill-opacity: 0.6; stroke: blue; stroke-width:8; stroke-opacity: 0.6}
```

## SVG\_DEFINE\_STYLE\_WITH\_ARRAYS

```
SVG_DEFINE_STYLE_WITH_ARRAYS ( svgObject ; namesArrayPointer ; valuesArrayPointer {; className {; type {; media {; title}}}} )
```

Parameter	Type	Description
svgObject	SVG_Ref	SVG object reference
namesArrayPointer	Pointer	Pointer to array of style names
valuesArrayPointer	Pointer	Pointer to array of style values
className	Text	CSS class name
type	Text	Type of contents
media	Text	Media descriptor
title	Text	Style name

## Description

The **SVG\_DEFINE\_STYLE\_WITH\_ARRAYS** method defines styles (using arrays) for the SVG object designated in the *svgObject*. parameter.

- If the *svgObject* parameter designates the root element, styles are set as "style" elements included in the "defs" section (Internal Style Sheet). In this case, the *className* parameter is mandatory (if it is missing, an error is returned). You can then assign the *className* style sheet to the SVG objects by passing its name to the **SVG\_SET\_CLASS** method (see example 1).
- If the *svgObject* parameter designates an SVG element other than the root element, the style is set as a style attribute for this element (Inline Style Sheet) (see example 2).

The optional *type* parameter specifies the language of the style sheet for the contents of the element. The default value is "text/css".

The optional *media* parameter indicates the desired destination media for the style information. If you omit this parameter, the default value used is "all". If the value is not included in the list of media types recognized by CSS2, an error is generated.

The optional *title* parameter adds an attribute of the "title" type.

## Example 1

Example of definition of internal styles:

```
ARRAY TEXT( $arrnames;0)
ARRAY TEXT( $arrvalues;0)
APPEND TO ARRAY( $arrnames;"fill")
APPEND TO ARRAY( $arrvalues;"black")
APPEND TO ARRAY( $arrnames;"font-family")
APPEND TO ARRAY( $arrvalues;"'Lucida Grande' Verdana")
APPEND TO ARRAY( $arrnames;"font-size")
APPEND TO ARRAY( $arrvalues;"20px")
APPEND TO ARRAY( $arrnames;"text-align")
APPEND TO ARRAY( $arrvalues;"center")

$svg:= SVG_New
SVG_DEFINE_STYLE_WITH_ARRAYS( $svg;-> $arrnames;-> $arrvalues;"title")
$object:= SVG_New_textArea( $svg;"Hello World!";10;10;200;310)
SVG_SET_CLASS( $object;"title")
```

This method generates the following code:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?> <svg xmlns="http://www.w3.org/2000/svg">
<defs id="4D"> <style type="text/css">.title{fill:red;font-family:'Lucida Grande' Verdana;font-size:20px;text-align:center;}</style> </defs> <textArea class="title" height="310" width="200" x="10" y="10">Hello World!</textArea> </svg>
```

## Example 2

## Example of definition of inline styles:

```
ARRAY TEXT( $arrnames;0)
ARRAY TEXT( $arrvalues;0)
APPEND TO ARRAY( $arrnames;"fill")
APPEND TO ARRAY( $arrvalues;"black")
APPEND TO ARRAY( $arrnames;"font-family")
APPEND TO ARRAY( $arrvalues;"'Lucida Grande' Verdana")
APPEND TO ARRAY( $arrnames;"font-size")
APPEND TO ARRAY( $arrvalues;"20px")
APPEND TO ARRAY( $arrnames;"text-align")
APPEND TO ARRAY( $arrvalues;"center")

$svg:= SVG_New
$object:= SVG_New_textArea( $svg;"Hello World!";10;10;200;310)
SVG_DEFINE_STYLE_WITH_ARRAYS( $object;-> $arrnames;-> $arrvalues)
```

This method generates the following code:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?> <svg xmlns="http://www.w3.org/2000/svg">
<textArea height="310" style="fill:red;font-family:'Lucida Grande' Verdana;font-size:20px;text-align:center;" width="200" x="10" y="10">Hello World!</textArea> </svg>
```

# SVG\_Define\_symbol

SVG\_Define\_symbol ( parentSVGObject ; id {; x {; y {; width {; height {; mode}}}} } ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
id	String	Name of symbol
x	Longint	X position of viewbox
y	Longint	Y position of viewbox
width	Longint	Width of viewbox
height	Longint	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\_DELETE\_OBJECT

SVG\_DELETE\_OBJECT ( *svgObject* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element

### Description

---

The *SVG\_DELETE\_OBJECT* command deletes the SVG object designated by *svgObject* from the document to which it belongs. An error is generated when *svgObject* is not a valid reference.

# SVG\_Get\_default\_encoding

SVG\_Get\_default\_encoding -> Function result

Parameter	Type	Description
Function result	Text	 Character set

## Description

---

The *SVG\_Get\_default\_encoding* command returns the encoding used when a new document is created.

## See also

[SVG\\_SET\\_DEFAULT\\_ENCODING](#)

# SVG\_New\_group

SVG\_New\_group ( parentSVGObject {; id {; url {; target}}}) -> Function result

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 a valid SVG group or 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.

## Example 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")
$newobject:= SVG_New_line( $group;100;300;300;100;"";5)
$newobject:= SVG_New_line( $group;300;300;500;100;"";10)
$newobject:= SVG_New_line( $group;500;300;700;100;"";15)
$newobject:= SVG_New_line( $group;700;300;900;100;"";20)
$newobject:= SVG_New_line( $group;900;300;1100;100;"";25)
```

## Example 2

Clickable text:



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

## SVG\_SET\_DEFAULT\_ENCODING

SVG\_SET\_DEFAULT\_ENCODING {{ encoding }}

Parameter	Type	Description
encoding	String	Character set

### Description

The *SVG\_SET\_DEFAULT\_ENCODING* command is used to set the encoding that will be used when subsequent documents are created.

If the *encoding* parameter is omitted, the command reestablishes the default character set: "UTF-8".

### See also

[SVG\\_Get\\_default\\_encoding](#)

# SVG\_Set\_description

SVG\_Set\_description ( parentSVGObject ; description ) -> Function result

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\_PATTERN\_CONTENT\_UNITS

SVG\_SET\_PATTERN\_CONTENT\_UNITS ( patternObject ; sysCoord )

Parameter	Type	Description
patternObject	SVG_Ref	Reference of pattern to modify
sysCoord	Text	System of coordinates to be used

## Description

---

The `SVG_SET_PATTERN_CONTENT_UNITS` command is used to set the system of coordinates for the contents of the pattern designated by *patternObject*. If *patternObject* is not a pattern, an error is generated.

The *sysCoord* parameter specifies the name of the system to be used. It must be set to "userSpaceOnUse" or "objectBoundingBox", otherwise an error is generated.

**See Also:** <http://www.w3.org/TR/SVG/patterns.html#Patterns>

## See also

[SVG\\_SET\\_PATTERN\\_UNITS](#)

# SVG\_SET\_PATTERN\_UNITS

SVG\_SET\_PATTERN\_UNITS ( patternObject ; sysCoord )

Parameter	Type	Description
patternObject	SVG_Ref	Reference of pattern to modify
sysCoord	Text	System of coordinates to be used

## Description

---

The `SVG_SET_PATTERN_UNITS` command is used to set the system coordinates for the `x`, `y`, `width` and `height` attributes of the pattern designated by `patternObject`. If `patternObject` is not a pattern, an error is generated.

The `sysCoord` parameter specifies the name of the system to be used. It must be set to "userSpaceOnUse" or "objectBoundingBox", otherwise an error is generated.

**See Also:** <http://www.w3.org/TR/SVG/patterns.html#Patterns>

## See also

[SVG\\_SET\\_PATTERN\\_CONTENT\\_UNITS](#)

# SVG\_Set\_title

SVG\_Set\_title ( parentSVGObject ; title ) -> Function result

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 used 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](#)

# Text

---

```
SVG_APPEND_TEXT_TO_TEXTAREA  
SVG_Get_text  
SVG_New_text  
SVG_New_textArea  
SVG_New_tspan  
SVG_New_vertical_text  
SVG_SET_FONT_COLOR  
SVG_SET_FONT_FAMILY  
SVG_SET_FONT_SIZE  
SVG_SET_FONT_STYLE  
SVG_SET_TEXT_ANCHOR  
SVG_SET_TEXT_KERNING  
SVG_SET_TEXT_LETTER_SPACING  
SVG_SET_TEXT_RENDERING  
SVG_SET_TEXT_WRITING_MODE  
SVG_SET_TEXTAREA_TEXT
```

## SVG\_APPEND\_TEXT\_TO\_TEXTAREA

SVG\_APPEND\_TEXT\_TO\_TEXTAREA ( *svgObject* ; *addedText* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of text element
<i>addedText</i>	Text	Text to be added

### Description

The *SVG\_APPEND\_TEXT\_TO\_TEXTAREA* command is used to append text to the textual content of the text object designated by *svgObject*. If *svgObject* is not a "textArea" object, an error is generated.

Line return characters are automatically replaced by "<tbreak/>" elements.

### Example

Adding the following text:

It is today, 28/04/10

and it was exactly 17:59:55

```
//Display outlines using 'rect' element
$Dom_rect:= SVG_New_rect( $Dom_SVG;10;10;500;200;0;0;"blue:50";"none")

//Creating the text
$Dom_text:= SVG_New_textArea( $Dom_SVG;"It is today, ";10;30;500;200;"'Arial'";36;0;3)

//Adding the date and 2 CR
SVG_APPEND_TEXT_TO_TEXTAREA( $Dom_text; String( Current date)+"\r\r")

//Lastly, adding the current time
SVG_APPEND_TEXT_TO_TEXTAREA( $Dom_text;"and it was exactly "+ String( Current time))
```

### See also

[SVG\\_New\\_textArea](#)  
[SVG\\_SET\\_TEXTAREA\\_TEXT](#)

## SVG\_Get\_text

SVG\_Get\_text ( *svgObject* ) -> Function result

Parameter	Type		Description
<i>svgObject</i>	SVG_Ref	→	Reference of text element
Function result	Text	⤒	Text contents

### Description

---

The *SVG\_Get\_text* command returns the textual content of the element designated by *svgObject*. If *svgObject* is not a text object reference ('text', 'textArea' or 'tspan'), an error is generated.

In the case of a textArea object, <tbreak/> elements are converted to CRs.

## SVG\_New\_text

SVG\_New\_text ( parentSVGObject ; text {; x {; y {; font | styleDef {; size {; style {; alignment {; color {; rotation {; lineSpacing {; stretching}}}}}}}} } ) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
text	Text	Text to insert
x	Real	Coordinate on X axis
y	Real	Coordinate on Y axis
font   styleDef	Text	Font name or Style definition
size	Longint	Size of characters in points
style	Longint	Style of characters
alignment	Longint	Alignment
color	String	Text color
rotation	Real	Angle of rotation of text
lineSpacing	Real	Line spacing in points
stretching	Real	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.

**Note:** Starting with 4D v15, the **SVG\_New\_text** command supports simple Styled text (the text can contain different styles, but SPAN attributes must not be nested). See example 5.

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 **SVG\_New\_text** command accepts two different syntaxes for setting characters:

- You can pass various values in the *font*, *size*, *style* and *alignment* parameters: *font* and *size* can be used to specify the font and size, in points, to be used. When these parameters are omitted, the text will be written in **Times New Roman 12 pts**.

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 (or you can also use the corresponding 4D constants from the **Font Styles** theme):

- 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:

- 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 (value included between 0 and 1) factor of the text.

- Or you can pass a style definition in the *styleDef* parameter (instead of the *font* parameter) and then omit the following parameters. For example, you can pass:

```
SVG_New_textArea( $Dom_svg;"Hello World !";x;y;vWidth;vHeight;style_definition)
```

... where the *style\_definition* parameter contains a complete style definition. If you pass, for instance, "{font-size:48px;fill:red;}", this definition is added as a style attribute in the form:

```
style="font-size:48px;fill:red;"
```

In this case, any additional parameters are ignored.

## Example 1

Simple text using default text properties:

```
Hello world!
```

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

## Example 2

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

```
Hello world!  
Bonjour le monde!  
Hola Mundo!
```

```
$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")
```

## Example 3

Vertical text:

```
Hello world
```

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

## Example 4

Condensed or expanded text:

```
Hello world (condensed)  
Hello world (normal)  
Hello world (stretched)
```

```
$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)
```

## Example 5

## Display of multi-style text:

```
C_TEXT( $Dom_svg; $Dom_text; $Txt_buffer)
//definition of multi-style text
$Txt_buffer:="<SPAN STYLE=\"font-size:18pt\>Hello </SPAN>" + \
"<SPAN STYLE=\"font-size:24pt;font-weight:bold;color:#D81E05\>World</SPAN>" + \
"<SPAN STYLE=\"font-size:36pt\>!</SPAN><BR/>" + \
"<SPAN STYLE=\"font-size:19pt;font-style:italic\>It's </SPAN>" + \
"<SPAN STYLE=\"font-size:24pt\>Monday</SPAN>"
$Dom_svg:= SVG_New

//title
SVG_SET_FONT_COLOR( SVG_New_text( $Dom_svg; "______ svg_Newtext _____";10;30); "blue")
//text
$Dom_text:= SVG_New_text( $Dom_svg; $Txt_buffer;50;50)

SVGTool_SHOW_IN_VIEWER( $Dom_svg)
SVG_CLEAR( $Dom_svg)
```

□

## See also

[SVG\\_New\\_textArea](#)  
[SVG\\_New\\_tspan](#)  
[SVG\\_New\\_vertical\\_text](#)

## SVG\_New\_textArea

SVG\_New\_textArea ( parentSVGObject ; text {; x {; y {; textWidth {; textHeight {; font | styleDef {; size {; style {; alignment}}}}}}}}) -> Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
text	Text	Text to insert
x	Longint	Coordinate on X axis
y	Longint	Coordinate on Y axis
textWidth	Longint	Width of text area
textHeight	Longint	Height of text area
font   styleDef	Text	Font name or Style definition
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.

### Notes:

- In the "textArea" element, <tbreak/> elements replace line returns.
- Starting with 4D v15, the **SVG\_New\_textArea** command supports simple Styled text (the text can contain different styles, but SPAN attributes must not be nested). See example 2.

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 *textWidth* and *textHeight* 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 **SVG\_New\_textArea** command accepts two different syntaxes for setting characters:

- You can pass various values in the *font* , *size* , *style* and *alignment* parameters: *font* and *size* can be used to specify the font and size, in points, to be used. When these parameters are omitted, the text will be written in **Times New Roman 12 pts**.

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 (or you can also use the corresponding 4D constants from the **Font Styles** theme):

- 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

- Or you can pass a style definition in the `styleDef` parameter (instead of the `font` parameter) and then omit the following parameters. For example, you can pass:

```
SVG_New_textArea( $Dom_svg;"Hello World !";x;y;vWidth;vHeight;style_definition)
```

... where the `style_definition` parameter contains a complete style definition. If you pass, for instance, "`{font-size:48px;fill:red;}`", this definition is added as a style attribute in the form:

```
style="font-size:48px;fill:red;"
```

In this case, any additional parameters are ignored.

## Example 1

*Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor.*

```
$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")
```

## Example 2

Display of multi-style text:

```
C_TEXT( $Dom_svg; $Dom_text; $Txt_buffer)
//definition of multi-style text
$Txt_buffer:="<SPAN STYLE=\"font-size:18pt\">Hello </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt;font-weight:bold;color:#D81E05\">World</SPAN>"+\
"<SPAN STYLE=\"font-size:36pt\">!</SPAN><BR/>"+\
"<SPAN STYLE=\"font-size:19pt;font-style:italic\">It's </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt\">Monday</SPAN>"
$Dom_svg:= SVG_New

//title
SVG_SET_FONT_COLOR( SVG_New_text( $Dom_svg;"_____ SVG_New_textArea _____";10;30;"";-1);"blue")
//textArea
$Dom_text:= SVG_New_textArea( $Dom_svg; $Txt_buffer;50;50)

SVGTool_SHOW_IN_VIEWER( $Dom_svg)
SVG_CLEAR( $Dom_svg)
```

SVG\_APPEND\_TEXT\_TO\_TEXTAREA  
SVG\_New\_text  
SVG\_New\_vertical\_text  
SVG\_SET\_TEXTAREA\_TEXT

# SVG\_New\_tspan

SVG\_New\_tspan ( parentSVGObject ; text {; x {; y {; font | styleDef {; size {; style {; alignment {; color}}}}} } } ) ->  
Function result

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
text	Text	Text to insert
x	Longint	Coordinate on X axis
y	Longint	Coordinate on Y axis
font   styleDef	Text	Font name or Style definition
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', 'tspan' or 'textArea' element designated by *parentSVGObject* and returns its reference. If *parentSVGObject* is not a reference to a 'text', 'tspan' or 'textArea' 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).

## Example 1

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

TITLE 1  
TITLE 2  
TITLE 3  
TITLE 4

```
$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)
```

## Example 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

```
SVG_New_vertical_text ( parentSVGObject ; text {; x {; y {; font {; size {; style {; alignment {; color {; rotation}}}}}}}}) -> Function result
```

Parameter	Type	Description
parentSVGObject	SVG_Ref	Reference of parent element
text	Text	Text to insert
x	Longint	Coordinate on X axis
y	Longint	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	Longint	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

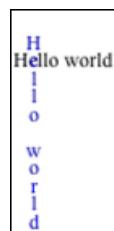
The optional `alignment` parameter can be used to set the type of alignment applied to the text drawn. You can pass on eof the following values:

- 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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>color</i>	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

SVG\_SET\_FONT\_FAMILY ( *svgObject* ; font {; font2 ; ... ; fontN} )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>font</i>	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. When several names are passed, the command automatically builds the list of font family names and/or generic family names.

### Example

Passing several font names:

```
SVG_SET_FONT_FAMILY(svgObject;"Lucida grande";"Sans-serif")
// will build the following list: " 'Lucida grande' 'Sans-serif'"
```

## SVG\_SET\_FONT\_SIZE

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>size</i>	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

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

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG element
<i>style</i>	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

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

Parameter	Type		Description
<i>svgObject</i>	SVG_Ref	→	Reference of SVG element
<i>alignment</i>	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](#)

# SVG\_SET\_TEXT\_KERNING

SVG\_SET\_TEXT\_KERNING ( *svgObject* ; *kerning* {; *unit*} )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of text element
<i>kerning</i>	Real	Letter spacing
<i>unit</i>	Text	Unit of spacing value

## Description

The [SVG\\_Get\\_text](#) command is used to modify the kerning for the text object designated by *svgObject*. If *svgObject* is not an SVG text object, an error is generated.

The optional *unit* parameter is used to specify the unit of the kerning value. The default value is "%".

If *kerning* is -1, the kerning value is set to 'auto'.

**Note:** Under Windows, the implementation is limited to text from left to right and top to bottom (disabled for right to left text) and to the 'text' and 'tspan' elements; under Mac OS, support is not limited.

**See Also:** <http://www.w3.org/TR/SVG/text.html#KerningProperty>

## Example

Examples of kerning variations:

Hello world !  
Hello world !

```
//Reference
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;40;"";36)
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_KERNING( $Dom_text;0,5)
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;120;"";36)
SVG_SET_TEXT_KERNING( $Dom_text;1)
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_KERNING( $Dom_text;1,5)
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_KERNING( $Dom_text;2)
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_KERNING( $Dom_text;1,5)
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;280;"";36)
SVG_SET_TEXT_KERNING( $Dom_text;1)
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_KERNING( $Dom_text;0,5)
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;360;"";36)
SVG_SET_TEXT_KERNING( $Dom_text;0)
```

## See also

[SVG\\_SET\\_TEXT\\_LETTER\\_SPACING](#)

# SVG SET TEXT LETTER SPACING

**SVG\_SET\_TEXT\_LETTER\_SPACING** ( *svgObject* ; *spacing* {; *unit*} )

Parameter	Type	Description
svgObject	SVG_Ref	Reference of text element
spacing	Real	Letter spacing
unit	Text	Unit of value

## Description

The `SVG_SET_TEXT_LETTER_SPACING` command is used to modify the letter spacing for the text object designated by `svgObject` in addition to the spacing due to the 'kerning' property. If `svgObject` is not an SVG text object, an error is generated.

The optional *unit* parameter is used to specify the unit of the spacing value. The default value is "%".

If *spacing* is -1, the spacing value is set to 'normal'.

**See Also:** <http://www.w3.org/TR/SVG/text.html#LetterSpacingProperty>

## Example

### Examples of spacing variations:

```
//Reference
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;40;"";36)

$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_LETTER_SPACING( $Dom_text;1)
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;120;"";36)
SVG_SET_TEXT_LETTER_SPACING( $Dom_text;1;"em")
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_LETTER_SPACING( $Dom_text;1;"px")
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_LETTER_SPACING( $Dom_text;1;"pt")
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_LETTER_SPACING( $Dom_text;1;"pc")
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;280;"";36)
SVG_SET_TEXT_LETTER_SPACING( $Dom_text;1;"mm")
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_LETTER_SPACING( $Dom_text;1;"cm")
$Dom_text:= SVG_New_text( $Dom_SVG;"Hello world !";20;360;"";36)
SVG_SET_TEXT_LETTER_SPACING( $Dom_text;1;"in")
```

#### See also

## SVG SET TEXT KERNING

## SVG\_SET\_TEXT\_RENDERING

SVG\_SET\_TEXT\_RENDERING ( *svgObject* ; *rendering* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of text element
<i>rendering</i>	Text	Value of rendering

### Description

---

The *SVG\_SET\_TEXT\_RENDERING* command is used to define the tradeoffs to make regarding the rendering of text for the text object designated by *svgObject*. If *svgObject* is not an SVG text object, an error is generated.

The *rendering* parameter can have one of the following values: "auto", "optimizeSpeed", "optimizeLegibility", "geometricPrecision" or "inherit". Otherwise, an error is generated.

**See Also:** <http://www.w3.org/TR/2001/REC-SVG-20010904/painting.html#TextRenderingProperty>

## SVG\_SET\_TEXT\_WRITING\_MODE

SVG\_SET\_TEXT\_WRITING\_MODE ( *svgObject* ; *writingMode* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of text element
<i>writingMode</i>	Text	Direction of writing

### Description

The *SVG\_SET\_TEXT\_WRITING\_MODE* command is used to set whether the writing direction for the text object designated by *svgObject* will be left to right, right to left or bottom to top. If *svgObject* is not an SVG text object, an error is generated.

The *writingMode* parameter can have one of the following values: "lr-tb", "rl-tb", "tb-rl", "lr", "rl", "tb" or "inherit ". Otherwise, an error is generated.

### Example

Writing from right to left:

الأدب العربي يشمل  
كافة الأعمال  
المكتوبة باللغة  
العربية ، ويشمل  
الأدب العربي الش رو  
الشعر المكتوبين  
بالعربية وكذلك  
يشمل الأدب

```
//Frame
$Dom_rect := SVG_New_rect( $Dom_SVG;5;5;210;310;0;0;"blue";"none")

//Text
$Dom_text := SVG_New_textArea( $Dom_SVG; $Txt_sample;10;10;200;300;"Baghdad 'Arial Unicode
MS'";25)
SVG_SET_TEXT_WRITING_MODE( $Dom_text;"rl")
```

## SVG\_SET\_TEXTAREA\_TEXT

SVG\_SET\_TEXTAREA\_TEXT ( *svgObject* ; *theText* )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of text element
<i>theText</i>	Text	Text to set

### Description

The *SVG\_SET\_TEXTAREA\_TEXT* command can be used to set/replace the textual content of the text object designated by *svgObject*. If *svgObject* is not a "textArea" object, an error is generated.

The line return characters are automatically replaced by "<tbreak/>" elements.

### See also

[SVG\\_APPEND\\_TEXT\\_TO\\_TEXTAREA](#)

[SVG\\_New\\_textArea](#)

# Utilities

---

SVG\_ABOUT  
SVG\_Count\_elements  
SVG\_ELEMENTS\_TO\_ARRAYS  
SVG\_Estimate\_weight  
SVG\_Find\_ID  
SVG\_Get\_options  
SVG\_Get\_root\_reference  
SVG\_Get\_version  
SVG\_Is\_reference\_valid  
SVG\_Post\_comment  
SVG\_Read\_element\_type  
SVG\_Read\_last\_error  
SVG\_References\_array  
SVG\_ROTATION\_CENTERED  
SVG\_SCALING\_CENTERED  
SVG\_Set\_error\_handler  
SVG\_SET\_OPTIONS  
SVGTool\_SET\_VIEWER\_CALLBACK  
SVGTool\_SHOW\_IN\_VIEWER

## SVG\_ABOUT

SVG\_ABOUT

Does not require any parameters

### Description

---

The *SVG\_ABOUT* command displays a dialog with the 4D SVG logo indicating the version number of the component:



# SVG\_Count\_elements

SVG\_Count\_elements ( *svgObject* ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG reference
Function result	Longint	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

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

Parameter	Type	Description
<i>svgObject</i>	Alpha	SVG reference
<i>refsArrayPointer</i>	Pointer	String array of object references
<i>typesArrayPointer</i>	Pointer	String array of object types
<i>namesArrayPointer</i>	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

SVG\_Estimate\_weight ( *svgObject* ) -> Function result

Parameter	Type	Description
svgObject	SVG_Ref	Reference of SVG document
Function result	Real	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

SVG\_Find\_ID ( *svgObject* ; *name* ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of SVG object
<i>name</i>	String	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, belonging to the SVG structure whose root element is passed in *svgObject*.

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

## See also

[SVG\\_Get\\_ID](#)  
[SVG\\_SET\\_ID](#)

# SVG\_Get\_options

SVG\_Get\_options -> Function result

Parameter	Type	Description
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)
9	Use trigonometric origin	0 (disabled)
10	Automatic Arial Substitution	1 (enabled)
11	Set shape-rendering='crispEdges' as default for a new canvas	0 (disabled)
12	Check parameters	1 (enabled)
13	Keep extra spaces	0 (disabled)
14	Centered rotation	0 (disabled)

- *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.

- *Use trigonometric origin*

By default, SVG places the origin at the top of the scale of degrees (midnight). This option lets you pass coordinates according to the usual trigonometric reference points (3h or 15mn). Conversion is performed on the fly.

- *Automatic Arial Substitution*

By default, 4D SVG replaces 'Arial' font with 'Arial Unicode MS', 'Arial' to improve compatibility with non-Roman characters (e.g. Japanese). In certain cases, you may want to disable this functioning. This option means you do not have to replace Arial fonts.

- *Set shape-rendering='crispEdges' as default for a new canvas*

The crispEdges attribute (see [SVG\\_SET\\_SHAPE\\_RENDERING](#)) can be forced by default using this option.

- *Checks parameters*

By default, 4D SVG checks the validity of parameters passed to commands. Once the development step is finished, you may want to disable this option to speed up code execution.

- *Keep extra spaces (New in v14)*

Allows multiple adjacent spaces to be displayed in text objects.

- *Centered rotation (New in v14)*

If this option is enabled, the [SVG\\_SET\\_TRANSFORM\\_ROTATE](#) command attempts to perform a centered rotation when the 3rd and 4th parameter are not passed. The center of rotation is calculated based on the x, y, width and height attributes of the object. If the object referenced does not have these attributes, the rotation is performed around the point (0,0).

## Example

---

See the [SVG\\_SET\\_OPTIONS](#) command.

## See also

[SVG\\_SET\\_OPTIONS](#)

## SVG\_Get\_root\_reference

SVG\_Get\_root\_reference ( *svgObject* ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
Function result	SVG_Ref	Root SVG element

### Description

---

The **SVG\_Get\_root\_reference** returns the reference of the root element for the SVG object whose reference was passed in the *svgObject* parameter.

# SVG\_Get\_version

SVG\_Get\_version -> Function result

Parameter	Type	Description
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

SVG\_Is\_reference\_valid ( *svgObject* ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	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\_Post\_comment

SVG\_Post\_comment ( *svgObject* ; *comment* ) -> Function result

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
<i>comment</i>	Text	Text to be added as a comment
Function result	SVG_Ref	Reference of comment

### Description

---

The **SVG\_Post\_comment** adds the text passed in *comment* as an XML comment to the SVG object designated by the *svgObject* parameter.

The method returns the SVG reference of the comment.

### Example

---

The following code:

```
[#code4D]C_TEXT($comment)
$comment:="Modified on "+String(Current date)
$ref:= SVG_Post_comment ($svg; $comment )
.. adds the following to the $svg SVG object:
<!--Modified on 12/12/2011-->[#/codeXML]
```

## SVG\_Read\_element\_type

SVG\_Read\_element\_type ( *svgObject* ) -> Function result

Parameter	Type	Description
svgObject	SVG_Ref	Reference of SVG element
Function result		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

SVG\_Read\_last\_error -> Function result

Parameter	Type	Description
Function result	Longint	 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

## Example 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
```

## Example 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
```

This method can be used as follows:

```
` 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

SVG\_References\_array ( refsArrayPointer ) -> Function result

Parameter	Type	Description
refsArrayPointer	Pointer	→ Alpha array of document references
Function result	Longint	↗ Number of references

### Description

The *SVG\_References\_array* command returns the list of current SVG document references in the array pointed to by *refsArrayPointer*. As the result, the command returns the number of references found.

*SVG\_References\_array* 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\_ROTATION\_CENTERED

SVG\_ROTATION\_CENTERED ( *svgObject* ; *angle* )

Parameter	Type		Description
<i>svgObject</i>	SVG_Ref	→	SVG object reference
<i>angle</i>	Real	→	Angle of rotation

### Description

---

The **SVG\_ROTATION\_CENTERED** command performs a centered rotation for the SVG object whose reference is passed in the *svgObject* parameter. This type of rotation can only be applied to objects having *x*, *y*, *width* and *height* attributes.

In the *angle* parameter, you pass the angle of rotation to be performed.

## SVG\_SCALING\_CENTERED

SVG\_SCALING\_CENTERED ( *svgObject* ; *scale* {; *x* ; *y*} )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	SVG object reference
<i>scale</i>	Real	Scale value
<i>x</i>	Real	X axis
<i>y</i>	Real	Y axis

### Description

---

The **SVG\_SCALING\_CENTERED** command performs a centered scaling of the SVG image whose reference is passed in the *svgObject* parameter.

In the *scale* parameter, you pass a positive scale value (>1). If you pass 1, the scale is set to 100%.

In the optional *x* and *y* parameters, you can pass, respectively, the x- and y-axis coordinates for the center point. If you do not pass these parameters, the center point is determined based on the "x", "y", "width" and "height" attributes of the object (if any). If this type of transformation is applied to an object that does not have these attributes, and the optional *x* and *y* parameters are omitted, an empty string is returned.

# SVG\_Set\_error\_handler

SVG\_Set\_error\_handler {{( method )} } -> Function result

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

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).

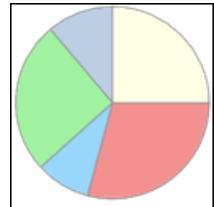
## Example 1

Create readable code:

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

## Example 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)
```

## Example 3

Displaying multiple spaces in text objects using the *Keep extra spaces* option (13) (added in v14):

```
$Txt_buffer:="abc      def"  
$Dom_text:= SVG_New_textArea( $Dom_svg; $Txt_buffer;50;50)
```

is displayed as "abc def"

```
SVG_SET_OPTIONS( SVG_Get_options?+13) // keep spaces in text objects  
$Txt_buffer:="abc      def"  
$Dom_text:= SVG_New_textArea( $Dom_svg; $Txt_buffer;50;50)
```

is displayed as "abc def"

## See also

[SVG\\_Get\\_options](#)

## SVGTool\_SET\_VIEWER\_CALLBACK

SVGTool\_SET\_VIEWER\_CALLBACK ( *methodName* )

Parameter	Type	Description
<i>methodName</i>	Text	Name of 4D project method

### Description

---

The *SVGTool\_SET\_VIEWER\_CALLBACK* command is used to install *methodName* as the project method that will be called when the [On Clicked](#) and [On Mouse Move](#) events occur on the image displayed by the [\*SVGTool\\_SHOW\\_IN\\_VIEWER\*](#) command.

This method receives a text parameter that is the ID of the element being clicked or over which the mouse is moving as provided by the 4D [SVG Find element ID by coordinates](#) command.

This parameter must be declared in the *methodName* project method of the host database by inserting it into the line C\_TEXT(\$1).

You must assign the "Shared by components and host database" property to the *methodName* method.

If the method does not exist or is not shared, the error -10508 is generated by 4D.

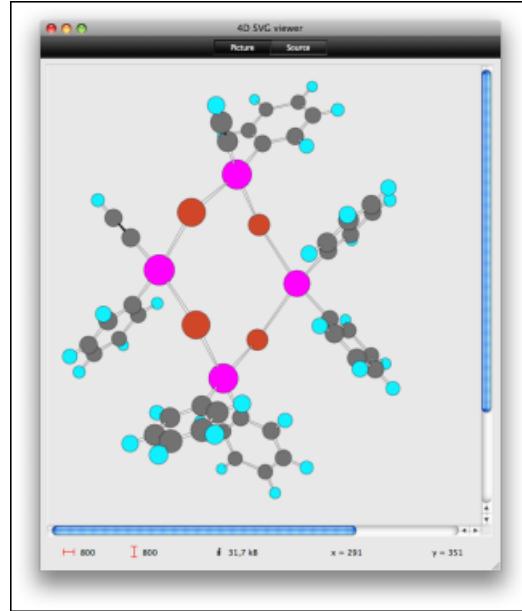
# SVGTool\_SHOW\_IN\_VIEWER

SVGTool\_SHOW\_IN\_VIEWER ( *svgObject* {; *sources*} )

Parameter	Type	Description
<i>svgObject</i>	SVG_Ref	Reference of picture to be displayed
<i>sources</i>	String	Opens viewer directly on the source page

## 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:



If you pass the optional *sources* parameter (added in v14), the viewer is opened directly on the source page. For more information about the SVG Viewer, please refer to the [Development tools](#) section.

## See also

[Development tools](#)

# 4D SVG Constants

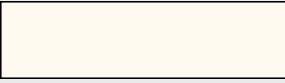
---

SVG Colors (Names)

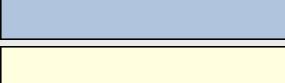
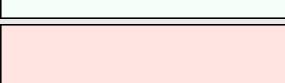
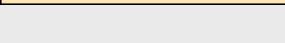
SVG Colors (RGB)

## SVG Colors (Names)

Constant	Type	Value	Comment
color SVG aliceblue	String	aliceblue	
color SVG antiquewhite	String	antiquewhite	
color SVG aqua	String	aqua	
color SVG aquamarine	String	aquamarine	
color SVG azure	String	azure	
color SVG beige	String	beige	
color SVG bisque	String	bisque	
color SVG black	String	black	
color SVG blanchedalmond	String	blanchedalmond	
color SVG blue	String	blue	
color SVG blueviolet	String	blueviolet	
color SVG brown	String	brown	
color SVG burlywood	String	burlywood	
color SVG cadetblue	String	cadetblue	
color SVG chartreuse	String	chartreuse	
color SVG chocolate	String	chocolate	
color SVG coral	String	coral	
color SVG cornflowerblue	String	cornflowerblue	
color SVG cornsilk	String	cornsilk	
color SVG crimson	String	crimson	
color SVG cyan	String	cyan	
color SVG darkblue	String	darkblue	
color SVG darkcyan	String	darkcyan	
color SVG darkgoldenrod	String	darkgoldenrod	

Constant	Type	Value	Comment
color SVG darkgray	String	darkgray	
color SVG darkgreen	String	darkgreen	
color SVG darkgrey	String	darkgrey	
color SVG darkkhaki	String	darkkhaki	
color SVG darkmagenta	String	darkmagenta	
color SVG darkolivegreen	String	darkolivegreen	
color SVG darkorange	String	darkorange	
color SVG darkorchid	String	darkorchid	
color SVG darkred	String	darkred	
color SVG darksalmon	String	darksalmon	
color SVG darkseagreen	String	darkseagreen	
color SVG darkslateblue	String	darkslateblue	
color SVG darkslategray	String	darkslategray	
color SVG darkslategrey	String	darksategrey	
color SVG darkturquoise	String	darkturquoise	
color SVG darkviolet	String	darkviolet	
color SVG deeppink	String	deeppink	
color SVG deepskyblue	String	deepskyblue	
color SVG dimgray	String	dimgray	
color SVG dimgrey	String	dimgrey	
color SVG dodgerblue	String	dodgerblue	
color SVG firebrick	String	firebrick	
color SVG floralwhite	String	floralwhite	
color SVG forestgreen	String	forestgreen	
color SVG fuchsia	String	fuchsia	

Constant	Type	Value	Comment
color SVG gainsboro	String	gainsboro	
color SVG ghostwhite	String	ghostwhite	
color SVG gold	String	gold	
color SVG goldenrod	String	goldenrod	
color SVG gray	String	gray	
color SVG green	String	green	
color SVG greenyellow	String	greenyellow	
color SVG grey	String	grey	
color SVG honeydew	String	honeydew	
color SVG hotpink	String	hotpink	
color SVG indianred	String	indianred	
color SVG indigo	String	indigo	
color SVG ivory	String	ivory	
color SVG khaki	String	khaki	
color SVG lavender	String	lavender	
color SVG lavenderblush	String	lavenderblush	
color SVG lawngreen	String	lawngreen	
color SVG lemonchiffon	String	lemonchiffon	
color SVG lightblue	String	lightblue	
color SVG lightcoral	String	lightcoral	
color SVG lightcyan	String	lightcyan	
color SVG lightgoldenrodyellow	String	lightgoldenrodyellow	
color SVG lightgray	String	lightgray	
color SVG lightgreen	String	lightgreen	
color SVG lightgrey	String	lightgrey	

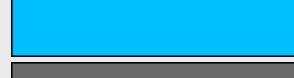
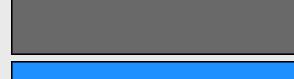
Constant	Type	Value	Comment
color SVG lightpink	String	lightpink	
color SVG lightsalmon	String	lightsalmon	
color SVG lightseagreen	String	lightseagreen	
color SVG lightskyblue	String	lightskyblue	
color SVG lightslategray	String	lightslategray	
color SVG lightslategrey	String	lightslategrey	
color SVG lightsteelblue	String	lightsteelblue	
color SVG lightyellow	String	lightyellow	
color SVG lime	String	lime	
color SVG limegreen	String	limegreen	
color SVG linen	String	linen	
color SVG magenta	String	magenta	
color SVG maroon	String	maroon	
color SVG mediumaquamarine	String	mediumaquamarine	
color SVG mediumblue	String	mediumblue	
color SVG mediumorchid	String	mediumorchid	
color SVG mediumpurple	String	mediumpurple	
color SVG mediumseagreen	String	mediumseagreen	
color SVG mediumslateblue	String	mediumslateblue	
color SVG mediumspringgreen	String	mediumspringgreen	
color SVG mediumturquoise	String	mediumturquoise	
color SVG mediumvioletred	String	mediumvioletred	
color SVG midnightblue	String	midnightblue	
color SVG mintcream	String	mintcream	
color SVG mistyrose	String	mistyrose	
color SVG moccasin	String	moccasin	

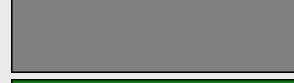
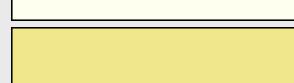
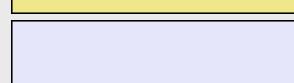
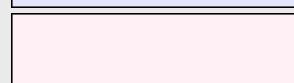
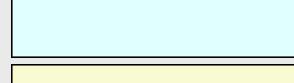
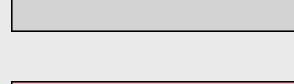
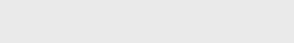
Constant	Type	Value	Comment
color SVG navajowhite	String	navajowhite	
color SVG navy	String	navy	
color SVG oldlace	String	oldlace	
color SVG olive	String	olive	
color SVG olivedrab	String	olivedrab	
color SVG orange	String	orange	
color SVG orangered	String	orangered	
color SVG orchid	String	orchid	
color SVG palegoldenrod	String	palegoldenrod	
color SVG palegreen	String	palegreen	
color SVG paleturquoise	String	paleturquoise	
color SVG palevioletred	String	palevioletred	
color SVG papyrus	String	papyrus	
color SVG peachpuff	String	peachpuff	
color SVG peru	String	peru	
color SVG pink	String	pink	
color SVG plum	String	plum	
color SVG powderblue	String	powderblue	
color SVG purple	String	purple	
color SVG red	String	red	
color SVG rosybrown	String	rosybrown	
color SVG royalblue	String	royalblue	
color SVG saddlebrown	String	saddlebrown	
color SVG salmon	String	salmon	
color SVG sandybrown	String	sandybrown	

Constant	Type	Value	Comment
color SVG seagreen	String	seagreen	
color SVG seashell	String	seashell	
color SVG sienna	String	sienna	
color SVG silver	String	silver	
color SVG skyblue	String	skyblue	
color SVG slateblue	String	slateblue	
color SVG slategray	String	slategray	
color SVG slategrey	String	slategrey	
color SVG snow	String	snow	
color SVG springgreen	String	springgreen	
color SVG steelblue	String	steelblue	
color SVG tan	String	tan	
color SVG teal	String	teal	
color SVG thistle	String	thistle	
color SVG tomato	String	tomato	
color SVG turquoise	String	turquoise	
color SVG violet	String	violet	
color SVG wheat	String	wheat	
color SVG white	String	white	
color SVG whitesmoke	String	whitesmoke	
color SVG yellow	String	yellow	
color SVG yellowgreen	String	yellowgreen	

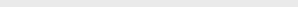
## SVG Colors (RGB)

Constant	Type	Value	Comment
color RGB aliceblue	Longint	15792383	
color RGB antiquewhite	Longint	16444375	
color RGB aqua	Longint	65535	
color RGB aquamarine	Longint	8388564	
color RGB azure	Longint	15794175	
color RGB beige	Longint	16119260	
color RGB bisque	Longint	16770244	
color RGB black	Longint	0	
color RGB blanchedalmond	Longint	16772045	
color RGB blue	Longint	255	
color RGB blueviolet	Longint	9055202	
color RGB brown	Longint	10824234	
color RGB burlywood	Longint	14596231	
color RGB cadetblue	Longint	6266528	
color RGB chartreuse	Longint	8388352	
color RGB chocolate	Longint	13789470	
color RGB coral	Longint	16744272	
color RGB cornflowerblue	Longint	6591981	
color RGB cornsilk	Longint	16775388	
color RGB crimson	Longint	14423100	
color RGB cyan	Longint	65535	
color RGB darkblue	Longint	139	
color RGB darkcyan	Longint	35723	
color RGB darkgoldenrod	Longint	12092939	

Constant	Type	Value	Comment
color RGB darkgray	Longint	11119017	
color RGB darkgreen	Longint	25600	
color RGB darkgrey	Longint	11119017	
color RGB darkkhaki	Longint	12433259	
color RGB darkmagenta	Longint	9109643	
color RGB darkolivegreen	Longint	5597999	
color RGB darkorange	Longint	16747520	
color RGB darkorchid	Longint	10040012	
color RGB darkred	Longint	9109504	
color RGB darksalmon	Longint	15308410	
color RGB darkseagreen	Longint	9419919	
color RGB darkslateblue	Longint	4734347	
color RGB darkslategray	Longint	3100495	
color RGB darkslategrey	Longint	3100495	
color RGB darkturquoise	Longint	52945	
color RGB darkviolet	Longint	9699539	
color RGB deeppink	Longint	16716947	
color RGB deepskyblue	Longint	49151	
color RGB dimgray	Longint	6908265	
color RGB dodgerblue	Longint	2003199	
color RGB firebrick	Longint	11673122	
color RGB forestgreen	Longint	2263842	
color RGB fuchsia	Longint	16711935	
color RGB gainsboro	Longint	14474460	
color RGB ghostwhite	Longint	16316671	
color RGB gold	Longint	16766720	

Constant	Type	Value	Comment
color RGB goldenrod	Longint	14329120	
color RGB gray	Longint	8421504	
color RGB green	Longint	32768	
color RGB greenyellow	Longint	11403055	
color RGB grey	Longint	8421504	
color RGB honeydew	Longint	15794160	
color RGB hotpink	Longint	16738740	
color RGB indianred	Longint	13458524	
color RGB indigo	Longint	4915330	
color RGB ivory	Longint	16777200	
color RGB khaki	Longint	15787660	
color RGB lavender	Longint	15132410	
color RGB lavenderblush	Longint	16773365	
color RGB lawngreen	Longint	8190976	
color RGB lemonchiffon	Longint	16775885	
color RGB lightblue	Longint	11393254	
color RGB lightcoral	Longint	1576136	
color RGB lightcyan	Longint	14745599	
color RGB lightgoldenrodyellow	Longint	16448210	
color RGB lightgray	Longint	13882323	
color RGB lightgreen	Longint	9498256	
color RGB lightgrey	Longint	13882323	
color RGB lightpink	Longint	16758465	
color RGB lightsalmon	Longint	16752762	
color RGB lightseagreen	Longint	2142890	

Constant	Type	Value	Comment
color RGB lightskyblue	Longint	8900346	
color RGB lightslategray	Longint	7833753	
color RGB lightslategrey	Longint	7833753	
color RGB lightsteelblue	Longint	11584734	
color RGB lightyellow	Longint	16777184	
color RGB lime	Longint	65280	
color RGB limegreen	Longint	3329330	
color RGB linen	Longint	16445670	
color RGB magenta	Longint	16711935	
color RGB maroon	Longint	8388608	
color RGB mediumaquamarine	Longint	6737322	
color RGB mediumblue	Longint	205	
color RGB mediumorchid	Longint	12211667	
color RGB mediumpurple	Longint	9662683	
color RGB mediumseagreen	Longint	3978097	
color RGB mediumslateblue	Longint	8087790	
color RGB mediumspringgreen	Longint	64154	
color RGB mediumturquoise	Longint	4772300	
color RGB mediumvioletred	Longint	13047173	
color RGB midnightblue	Longint	1644912	
color RGB mintcream	Longint	16121850	
color RGB mistyrose	Longint	16770273	
color RGB moccasin	Longint	16770273	
color RGB navajowhite	Longint	16768685	
color RGB navy	Longint	128	
color RGB oldlace	Longint	16643558	

Constant	Type	Value	Comment
color RGB olive	Longint	8421376	
color RGB olivedrab	Longint	7048739	
color RGB orange	Longint	16753920	
color RGB orangered	Longint	16729344	
color RGB orchid	Longint	14315734	
color RGB palegoldenrod	Longint	15657130	
color RGB palegreen	Longint	10025880	
color RGB paleturquoise	Longint	11529966	
color RGB palevioletred	Longint	14381203	
color RGB papyrus	Longint	16773077	
color RGB peachpuff	Longint	16767673	
color RGB peru	Longint	13468991	
color RGB pink	Longint	16761035	
color RGB plum	Longint	14524637	
color RGB powderblue	Longint	11591910	
color RGB purple	Longint	8388736	
color RGB red	Longint	16711680	
color RGB rosybrown	Longint	12357519	
color RGB royalblue	Longint	4286945	
color RGB saddlebrown	Longint	9127187	
color RGB salmon	Longint	16416882	
color RGB sandybrown	Longint	16032864	
color RGB seagreen	Longint	3050327	
color RGB seashell	Longint	16774638	
color RGB sienna	Longint	10506797	

Constant	Type	Value	Comment
color RGB silver	Longint	12632256	
color RGB skyblue	Longint	8900331	
color RGB slateblue	Longint	6970061	
color RGB slategray	Longint	7372944	
color RGB slategrey	Longint	7372944	
color RGB snow	Longint	16775930	
color RGB springgreen	Longint	65407	
color RGB steelblue	Longint	4620980	
color RGB tan	Longint	13808780	
color RGB teal	Longint	32896	
color RGB thistle	Longint	14204888	
color RGB tomato	Longint	16737095	
color RGB turquoise	Longint	4251856	
color RGB violet	Longint	15631086	
color RGB wheat	Longint	16113331	
color RGB white	Longint	16777215	
color RGB whitesmoke	Longint	16119285	
color RGB yellow	Longint	16776960	
color RGB yellowgreen	Longint	10145074	

# Appendices

---

External links

## External links

---

Here is a list of links concerning the SVG standard:

### Overview

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

### Specification

<http://www.w3.org/TR/SVG12/>

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

# 4D SVG Component - What's new

---

SVG\_SET\_STROKE\_VECTOR\_EFFECT

# SVG\_SET\_STROKE\_VECTOR\_EFFECT

SVG\_SET\_STROKE\_VECTOR\_EFFECT ( *svgObj* {; *vectorEffect*} )

Parameter	Type	Description
<i>svgObj</i>	SVG_Ref	Reference of SVG element
<i>vectorEffect</i>	String	"non-scaling-stroke", "none", "inherit"

## Description

The **SVG\_SET\_STROKE\_VECTOR\_EFFECT** command sets the vector-effect attribute for the lines of the *svgObj*. The vector-effect attribute specifies the behavior of an object's stroke if the object is transformed, for example by zooming in or out.

In the *svgObj* parameter, pass a reference to the SVG element whose vector-effect attribute will be set. If *svgObj* does not reference an SVG element, an error is generated. If the vector-effect attribute has already been defined, its value is replaced by the *vectorEffect* parameter.

The optional *vectorEffect* parameter indicates the value to be used for the attribute. The following values are available:

<b>vectorEffect</b>	<b>Description</b>
"non-scaling-stroke"	Stroke (line) width is not dependant on the transformations of the element
"none"	No vector effect is applied
"inherit"	The value of the element's parent is used

If *vectorEffect* is omitted, the vector-effect attribute is removed (equivalent to "none").

## Example

The following code creates a rectangle and resizes it:

```
C_TEXT( $Dom_rect; $Dom_svg)
$Dom_svg:= SVG_New
$Dom_rect:= SVG_New_rect( $Dom_svg;10;10;100;100;0;0;"blue";"red")
SVG_SET_STROKE_WIDTH( $Dom_rect;2;"px")

// Redimension
SVG_SET_TRANSFORM_SCALE( $Dom_rect;4;1)
```

By default, line thickness is modified:



To resize without changing line thickness, add **SVG\_SET\_STROKE\_VECTOR\_EFFECT**:

```
C_TEXT( $Dom_rect; $Dom_svg)
$Dom_svg:= SVG_New
$Dom_rect:= SVG_New_rect( $Dom_svg;10;10;100;100;0;0;"blue";"red")
SVG_SET_STROKE_WIDTH( $Dom_rect;2;"px")

// Redimension
SVG_SET_TRANSFORM_SCALE( $Dom_rect;4;1)

// Set non-scaling
SVG_SET_STROKE_VECTOR_EFFECT( $Dom_rect;"non-scaling-stroke")
```

The line thickness is not modified:

