Creating and Destroying Objects

createCharStyle(...)

```
createCharStyle(...)
```

Creates a character style. This function takes the following keyword parameters:

- "name" [required] -> name of the char style to create
- "font" [optional] -> name of the font to use
- fontsize [optional] -> font size to set (double)
- "features" [optional] -> nearer typographic details can be defined by a string that might contain the following phrases comma-seperated (without spaces!):
  - inherit
  - bold
  - italic
  - underline
  - underlinewords
  - strike
  - superscript
  - subscript
  - outline
  - shadowed
  - allcaps
  - smallcaps
- "fillcolor" [optional], "fillshade" [optional] -> specify fill options
- "strokecolor" [optional], "strokeshade" [optional] -> specify stroke options
- baselineoffset [optional] -> offset of the baseline
- shadowxoffset [optional], shadowyoffset [optional] -> offset of the shadow if used
- outlinewidth [optional] -> width of the outline if used
- underlineoffset [optional], underlinewidth [optional] -> underline options if used
- strikethruoffset [optional], strikethruwidth [optional] -> strikethru options if used
- scaleh [optional], scalev [optional] -> scale of the chars
- tracking [optional] -> tracking of the text
- "language" [optional] -> language code

createBezierLine(...)

```
createBezierLine(list, ["name"]) -> string
```

Creates a new bezier curve and returns its name. The points for the bezier curve are stored in the list "list" in the following order: [x1, y1, kx1, ky1, x2, y2, kx2, ky2...xn. yn, kxn. kyn] In the points list, x and y mean the x and y coordinates of the point and kx and ky meaning the control point for the curve. The coordinates are given in the current measurement units of the document (see UNIT constants). "name" should be a unique identifier for the object because you need this name for further access to that object. If "name" is not given Scribus will create one for you.

May raise NameExistsError if you explicitly pass a name that's already used. May raise ValueError if an insufficient number of points is passed or if the number of values passed don't group into points without leftovers.

createEllipse(...)

```
createEllipse(x, y, width, height, ["name"]) -> string
```

Creates a new ellipse on the current page and returns its name. The coordinates are given in the current measurement units of the document (see UNIT constants). "name" should be a unique identifier for the object because you need this name for further referencing of that object. If "name" is not given Scribus will create one for you.

May raise NameExistsError if you explicitly pass a name that's already used.

createImage(...)

```
createImage(x, y, width, height, ["name"]) -> string
```

Creates a new picture frame on the current page and returns its name. The coordinates are given in the current measurement units of the document. "name" should be a unique identifier for the object because you need this name for further access to that object. If "name" is not given Scribus will create one for you.

May raise NameExistsError if you explicitly pass a name that's already used.

createLine(...)

```
createLine(x1, y1, x2, y2, ["name"]) -> string
```

Creates a new line from the point(x1, y1) to the point(x2, y2) and returns its name. The coordinates are given in the current measurement unit of the document (see UNIT constants). "name" should be a unique identifier for the object because you need this name for further access to that object. If "name" is not given Scribus will create one for you.

May raise NameExistsError if you explicitly pass a name that's already used.

createParagraphStyle(...)

```
createParagraphStyle(...)
```

Creates a paragraph style. This function takes the following keyword parameters:

- "name" [required] -> specifies the name of the paragraphstyle to create
- linespacingmode [optional] -> specifies the linespacing mode; possible modes are:
    - fixed linespacing: 0
    - automatic linespacing: 1
    - baseline grid linespacing: 2
- linespacing [optional] -> specifies the linespacing if using fixed linespacing
- alignment [optional] -> specifies the alignment of the paragraph
    - left: 0
    - center: 1
    - right: 2
    - justify: 3
    - extend: 4
- leftmargin [optional], rightmargin [optional] -> specify the margin
- gapbefore [optional], gapafter [optional] -> specify the gaps to the heading and following paragraphs
- firstindent [optional] -> the indent of the first line
- hasdropcap [optional] -> specifies if there are caps (1 = yes, 0 = no)
- dropcaplines [optional] -> height (in lines) of the caps if used
- dropcapoffset [optional] -> offset of the caps if used
- "charstyle" [optional] -> char style to use

createPathText(...)

```
createPathText(x, y, "textbox", "beziercurve", ["name"]) -> string
```

Creates a new pathText by merging the two objects "textbox" and "beziercurve" and returns its name. The coordinates are given in the current measurement unit of the document (see UNIT constants). "name" should be a unique identifier for the object because you need this name for further access to that object. If "name" is not given Scribus will create one for you.

May raise NameExistsError if you explicitly pass a name that's already used. May raise NotFoundError if one or both of the named base object don't exist.

createPolyLine(...)

```
createPolyLine(list, ["name"]) -> string
```

Creates a new polyline and returns its name. The points for the polyline are stored in the list "list" in the following order: [x1, y1, x2, y2...xn. yn]. The coordinates are given in the current measurement units of the document (see UNIT constants). "name" should be a unique identifier for the object because you need this name for further access to that object. If "name" is not given Scribus will create one for you.

May raise NameExistsError if you explicitly pass a name that's already used. May raise ValueError if an insufficient number of points is passed or if the number of values passed don't group into points without leftovers.

createPolygon(...)

```
createPolygon(list, ["name"]) -> string
```

Creates a new polygon and returns its name. The points for the polygon are stored in the list "list" in the following order: [x1, y1, x2, y2...xn. yn]. At least three points are required. There is no need to repeat the first point to close the polygon. The polygon is automatically closed by connecting the first and the last point. The coordinates are given in the current measurement units of the document (see UNIT constants). "name" should be a unique identifier for the object because you need this name for further access to that object. If "name" is not given Scribus will create one for you.

May raise NameExistsError if you explicitly pass a name that's already used. May raise ValueError if an insufficient number of points is passed or if the number of values passed don't group into points without leftovers.

createRect(...)

```
createRect(x, y, width, height, ["name"]) -> string
```

Creates a new rectangle on the current page and returns its name. The coordinates are given in the current measurement units of the document (see UNIT constants). "name" should be a unique identifier for the object because you need this name to reference that object in future. If "name" is not given Scribus will create one for you.

May raise NameExistsError if you explicitly pass a name that's already used.

createText(...)

```
createText(x, y, width, height, ["name"]) -> string
```

Creates a new text frame on the actual page and returns its name. The coordinates are given in the actual measurement unit of the document (see UNIT constants). "name" should be a unique identifier for the object because you need this name for further referencing of that object. If "name" is not given Scribus will create one for you.

May raise NameExistsError if you explicitly pass a name that's already used.

deleteObject(...)

```
deleteObject(["name"])
```

Deletes the item with the name "name". If "name" is not given the currently selected item is deleted.

getAllStyles(...)

```
getAllStyles() -> list
```

Return a list of the names of all paragraph styles in the current document.

objectExists(...)

```
objectExists(["name"]) -> bool
```

Test if an object with specified name really exists in the document. The optional parameter is the object name. When no object name is given, returns True if there is something selected.