



Contents

- [1 Create the SVG files](#)
- [2 TIPS:](#)
- [3 Convert SVG files to Symbian mif file](#)
- [4 Practical experience](#)
- [5 Related Links](#)

Create the SVG files

There are at least three ways.

1. The easiest way is to buy Adobe Illustrator CS2 to draw icons. Illustrator CS2 can save to SVG-Tiny format. It costs some money but it will save you a lot of time. If you're not great at drawing - there's a feature that will convert a photo to an [SVG](#) line drawing with color fills. So, make a photo of an object with your phone and have illustrator convert it to a line drawing.
2. You can download a free editor named [Inkscape](#). [Inkscape](#) saves to [SVG](#) format but not SVG-Tiny. You'll have to use the SVG2SVGT tool to convert the icon (found under S60Tools folder in [SDK](#)). But there's a problem. If you use certain draw object types then [Inkscape](#) will produce [SVG](#) code that will not work. Start with a very simple image such as a box - test it - and then add some details. ([Inkscape](#) is open source C++ so anyone of us could fix it to directly generate SVG-T for [S60](#).)
3. You can type an [SVG](#) file into a text editor. It's [XML](#) syntax.

TIPS:

- An easy way to test your icon is to build the S60Ex\helloworldbasic example. Replace helloworldbasic\gfx\qgn_menu_helloworldbasic.svg with your [svg](#) file. Test on a phone.
- Make your icon on a 44x44 pixel canvas. This might not be necessary in some cases but it is not clear whether [S60](#) will scale your icon in all the places it can be used.
- Keep in mind that [S60](#) caches icons between runs of your application - so sometimes when you update an icon you still see the previous version. *You're not losing your mind* - it's caching the icon. **Try restarting the phone when updating icons.**

Convert SVG files to Symbian mif file

You would find a tool called "[mifconv](#)" in the epoc32/tools:

For e.g: Now from command line:

```
c:\Symbian\9.1\S60_3rd_MR\Epoc32\tools>mifconv MyNewApp.mif /c32 C:\MyProject\gfx\qgn_menu.svg
```

This will generate .mif into epoc32/tools folder.

How_to_create_application_icon(SVG)_in_S60_3rd_edition

Now open your .pkg file and find for path of .mif file. Then just give path of your newly created .mif file.
Like:

```
"C:\Symbian\9.1\S60_3rd_MR\Epoc32\tools\MyNewApp.mif" -"!:\resource\apps\YourProjectNamed_reg.mif"
```

You could either cut-paste your newly created .mif file into your folder and give its path in source path of your .pkg file. Like:

```
"C:\MyFolder\MyNewApp.mif" -"!:\resource\apps\YourProjectNamed_reg.mif"
```

Note: Here *qgn_menu.svg* has been redesigned in the [Inkscape](#) and replaced with the same name as *qgn_menu.svg*

Practical experience

Although I was quite enthusiastic about using Inkscape at first it turns out the gradient rendering differs a lot from what you see in Inkscape and what you get on the phone. So you end-up having to do a lot of guess work and tries to eventually get the desired result. I had designed with Inkscape an icon that was eventually working fine on 3rd edition devices but it failed to render properly on 5th edition. My guess is that there was some opacity problem. Maybe they have better opacity support on 5th edition SVG which broke the rendering of my icon.

I decided to redesign that icon using Adobe Illustrator CS4. AI is much easier to manipulate than Inkscape and the resulting SVG code much more compact. It also renders very much the same on the devices as it does on your PC. Of course you have to stick to the features supported by SVG Tiny 1.1+.

Adobe Illustrator CS4 offers an export function to SVG Tiny 1.1+ but that's not always working unfortunately. In many cases even though you specified to export as SVG Tiny 1.1+ it wants to rasterize your gradients. Even if you can manage to get the SVG Tiny 1.1+ export to work next time you open your SVGT file with AI it will complain and rasterize your gradient. One of the problem seems that there is no DTD for SVG Tiny 1.1+ and AI uses the DTD for SVG Tiny 1.1 which does not support gradients for instance.

I settled for exporting as SVG Basic (SVGB) and making sure not to use features not supported by SVG Tiny 1.1+. It seems the MIF compiler can process SVG files regardless of their headers (Tiny, Basic or full SVG) so what matters is that you use only the features supported by the phones you want to render the icon with.

Related Links

- See [SVG Profiles](#)
- [Issue with SVG icon display on 3rd edition](#)
- [KIS000398 - SVG rendering problems caused by missing viewBox attribute](#)
- [KIS000531 - Compatibility problem with binary-encoded SVG images](#)