

Symbian Scalable UI Framework

This article provides an overview of the Symbian Scalable UI framework, explaining why the framework is needed, and what developers need to do to make their apps work seamlessly on devices with different screen resolutions and orientations.

Introduction

Mobile devices are evolving to support higher resolutions and more accurate displays, which consumers demand and appreciate. Device segmentation, besides just price point, is getting broader with different requirements: for example, a music-optimized device can survive with a lower-resolution display compared to a device optimized for media consumption, such as Internet browsing. Bigger and more accurate displays also enable new use cases and richer content, such as maps, fancy games, photos, videos, and thereby also new business opportunities. In addition, screen orientation is becoming important - mobile TV and browsing, for example, utilize and fit better into a widescreen landscape display compared to the traditional portrait.

Display variance and evolution is needed and unavoidable. With that in mind, developers need to consider how to deal with that variance effectively, which is where the scalable UI concept comes into picture. Instead of providing multiple software platforms with almost the same APIs, one platform can support multiple UIs and screen resolutions by providing a scalable UI framework. Applications built on top of the [S60](#) platform either work automatically on the supported resolutions or can be made to support all those resolutions with the help of the APIs provided by the platform. The level to which extent the platform handles the scalability automatically, as opposed to being handled by the developer, varies significantly depending on the application UI implementation.

Scalable UI is part of the platform from [S60](#) 2nd Edition, Feature Pack 3. The supported screen resolutions, APIs and the scalable graphics format for icons and themes are described in the following sections.

Screen Resolutions

Applications implemented for [S60](#) 2nd Edition, Feature Pack 3 and later should scale to all of the supported resolutions. In addition to the 176 x 208 resolution currently in use, two new resolutions are now available. From this point on, [S60](#) devices may use one of the following screen resolutions:

- 176 x 208 – standard
- 240 x 320 – Quarter VGA (QVGA)
- 352 x 416 – double resolution
- 640 x 360 – nHD resolution

In addition to the traditional portrait layout, a landscape layout is supported in the double resolution (352 x 416) and QVGA (240 x 320) modes.

[Symbian Anna](#) (in particular the [Nokia E6](#)) adds support for the following screen resolutions:

- 640 x 480 – VGA at high pixel density (326 DPI),

New API's

The following new APIs have been introduced to allow the development of applications for scalable UIs:

- **UI Metrics API**

The API provides layout information about the UI component (size, position, etc.). The UI Metrics API is part of the Utilities API under the UI Framework in the SDK help. The term UI Metrics API does therefore not appear in the SDK help.

- **Orientation Mode API**

The API provides set and get methods for the orientation mode (portrait or landscape). The Orientation Mode API is part of the Application framework API under the UI Framework in the SDK help. The term Orientation Mode API does therefore not appear in the SDK help.

- **Scalable Icons API**

The API is used to create scalable icons. The Scalable Icons API is part of the Skins API under the UI Framework in the SDK help. The term Scalable Icons API does therefore not appear in the SDK help.

- **Logical Font API**

New logical fonts are defined to get suitable fonts regardless of the current screen resolution. The Logical Font API is part of the

Utilities API under the UI Framework in the SDK help. The term Logical Font API does therefore not appear in the SDK help.

Bitmaps and Icons

Bitmap graphics do not scale very well for different screen resolutions. Therefore a new C++ API has been introduced for loading old bitmap icons and scalable icons based on Scalable Vector Graphics – Tiny (SVG-T).

A new tool (MifConv.exe) is also available to produce multi-image files (MIFs) — similar to Multi-Bitmap (MBM) — from SVG-T files. A new application information file (AIF) framework has been introduced to support scalable icons.

Localization

Due to new resolutions and the change in the orientation mode the space available for text strings in the screen can change during runtime. Therefore a Symbian C++ application can use shorter and longer versions of the same text. However, it is wise to study the usability of each case. Texts in a localization file (.loc) can have alternative versions for different text spaces. When reading this kind of text from resources and passing it to Avkon components, the available space is checked for the text based on the current layout and orientation, and the correct alternative is selected. In addition, a utility function in **AknTextUtils** can be used directly, especially if the text is drawn directly to the screen.

Important Links

- [S60 Platform: Scalable Screen-Drawing How-to](#)
- [S60 Platform: Scalable Screen-Drawing Example](#)