

# Archived:DSA API not supported by compatibility mode in scalable UI platforms (Known Issue)



**Archived:** This article is [archived](#) because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template `{{ReviewForRemovalFromArchive|user=~~~~|write your reason here}}`.

The article is believed to be still valid for the original topic scope.

---

## Overview

The DSA API is not supported by the compatibility mode in scalable UI platforms.

---

## Description

The Direct Screen Access (DSA) API (`RDirectScreenAccess` and `CDirectScreenAccess` classes, defined in `w32std.h`) is meant for very low-level operations with display hardware. In platforms supporting the scalable UI (S60 2nd Edition FP3 and onwards) DSA API is not supported by the compatibility mode. For this and other breaks between different platforms, refer to the document [S60 Platform: Source And Binary Compatibility](#).

---

## How to reproduce

Applications working correctly in earlier S60 devices using the Direct Screen Access API may behave unexpectedly in Feature Pack 3 devices (with varying display characteristics), because, for example, the N90 has more resolution and a different color depth (18-bit colors) than previous S60 devices. Applications that use this API to access display buffer (memory) directly, may encounter different problems depending on how the applications have been implemented. Some applications render their screen only to 1/4th portion of the screen, whereas some may fill the screen with garbage, have incorrect colors, or may exit when they notice that the display attributes are not what they assume.

---

## Solution

Possible changes in resolutions between S60 devices must be taken into account when developing applications. You can use `CFbsScreenDevice::SizeInPixels()` to get the current display size.

