Upgrading Applications to Support New OS Platform Versions

Version: 2013.11.22

This section includes best practices that should be considered when planning for upgrading apps with support for new platform versions.

Checklist

- ❑ Plan ahead for periodic O/S upgrades
- ❑ Assume required changes to support any new O/S version
- ❑ Assume required changes to support any new platform or O/S
- ❑ Target beta periods to upgrade applications...as a continuous cycle
- ❑ Upgrade Native OS Vendor's SDK during Beta period
- ❑ Think twice about running in legacy mode
- ❑ Treat hybrid webview as different from standalone web browsers
- ❑ Upgrade application-level libraries used for new platform support
- ❑ On Android platform, consider using exclusion capabilities to protect against untested new device use

Discussion

Plan ahead for periodic O/S upgrades

When supporting an app on multiple mobile operating systems, because of the release frequency of OS upgrades it is important to have an ongoing plan for updating existing app's platform support on an periodic basis. Native OS api's can change the behavior of applications, and new OS versions are also typically when OS-supplied browsers are updated. Usually breaking changes are contained mainly to major version upgradees, but it's important to verify application regression testing through continuous testing as major.minor levels are released as well.

As part of a periodic application app upgrade plan, it is important to review the items within this document.

Assume required changes to support any new O/S version

Assume that there will always be changes needed to support a new version of an operating system already supported. Whether you are building native, hybrid or web apps, you should always assume that the new OS version upgrade will introduce some changes that will break your app--especially when adding a new major version of operating system support.

Adding support for a new major version of an operating system requires more effort than adding support for a new major.minor version, and less effort than adding support for a new operating system altogether.

Assume required changes to support any new platform or O/S

You should assume that there will always be changes needed to add support for a new operating system that is not already supported, even when using hybrid and web app architectures. Web and hybrid application architectures save costs over time because of fewer skills compared to native apps. However, web and hybrid apps are not write once, run on every device or operating system without any changes--they usually require some app development updates and testing to support additional devices or platforms, even when using mobile-first responsive design techniques.

Changes will always be needed to support new versions of operating systems. The costs of adding support for new devices/OS will generally be less than that of native projects, but there is still work involved. The browser engine on different mobile OS's may be webkit based; however, webkit is designed modularly and capabilities still vary greatly across versions of different operating systems (and sometimes within the same operating system).

Using mobile first responsive design techniques (ie. “flexible layouts and environment aware components”) can further reduce the costs of supporting new OS versions as compared to prescriptive designed applications (eg. “pixel perfect” designs); however, there are still maintenance costs involved with keeping media queries and stylesheets up to date with OS/browser changes.

Target beta periods to upgrade applications...as a continuous cycle

Your users will be buying new models the day they arrive and trying out your app on the very first day. The user expectation is that you’ll have done work already to ensure your app is ready to go for that new iGadget v128.0 on the day it comes out. This means you have to be done with development and testing prior to that GA date.

Whether you’re doing native, hybrid or web apps, plan for additional development work to add support for any new operating system to be completed in the beta period prior to the new operating system’s expected GA date.

The more apps you have that will need to support the new major version, the more testing/ development resources will be needed to upgrade them during the beta period.

This also means that your application’s dependencies, 3rd party vendor runtime libraries and tools your application uses above and beyond the device vendor’s SDK need to be certified for use with the new operating system version prior to the GA date of the new device’s operating system update. See additional guidance regarding library upgrades later in this document.

Upgrade Native OS Vendor’s SDK during Beta period

During the beta period, upgrade your prerequisite development tools to versions that correspond to the the new mobile OS version:

- XCode
- Android SDK
- Blackberry SDK
- Microsoft Windows Phone SDK

Think twice about running in legacy mode

Sometimes device operating system vendors allow running applications built on previous versions of tools on newer versions of an operating system in a legacy mode. Running in legacy mode can sometimes be leveraged to extend the time needed to migrate an application to a new version of the device operating system beyond the general availability date of the device operating system.

However, it is not a recommended practice to remain in legacy mode for an extended period of time and to use the time in legacy mode to upgrade the application with true

support for the new version of the operating system. The reason being is that the device operating system vendor could pull support for the previous operating system version's tools running on the new operating system version at their discretion.

Treat hybrid webview as different from standalone web browsers

You should treat hybrid webview environments as a different environment from that of the default standalone web browser on an operating system. Although usually the standalone web browser will use the same engine as the browser engine used by embedded web views (used by hybrid apps), this is sometimes not the case. Even if the same browser engine is used, different options will be enabled between webviews and standalone browser, such as JavaScript optimization levels that can provide different behaviors in the two environments.

Upgrade application-level libraries used for new platform support

3rd party Javascript libraries including any 3rd party plugins or extensions to these libraries from other vendors you may be using:

- jQuery
- Dojo
- EmberJS
- AngularJS
- Sencha
- etc.

3rd party Native (iOS, Android, Blackberry, Win8) client-side libraries

- Augmented Reality
- Barcode scanning, etc.

The libraries may be provided by a mobile platform provider (eg. IBM, Kony, Titanium, etc.) or provided by 3rd parties. It is important to review the versions of the native libraries being used to ensure that the libraries have been tested and are supported on the new version of the operating system the application will be supported on. If not, plan time prior to the OS update becoming available to upgrade these libraries.

Finally, plan appropriate time to regression test the application functionality on the new operating system prior to the general availability release of the new OS version if possible. Sometimes mobile operating system versions can change browser or webview dimensions or other native UI elements that require changes to an application these changes can occur in either native or JavaScript/HTML code and may require things like CSS styling and responsive design techniques (eg. media queries) to adjust UI layouts flexibly to the changes introduced by the mobile OS vendor.

On Android platform, consider using exclusion capabilities to protect against untested new device use

The Android beta program is typically limited to participation by device carriers, unlike Apple which as a beta period prior to GA that developers can get access to. Because of this, you will likely not be able to test your application prior to the GA release date for new versions of Android. Instead, as devices become available that can run the new version of Android (or if you can upgrade current device's at time of GA release, you can then begin testing).

When you already have an Android app in the Google Play app store, you can protect against new untested devices/OS versions being used with your app, until you have had time to test your app on the new version by using the Google Play exclusion list feature.

Reference:

- http://mobile.tutsplus.com/tutorials/android/android-essentials-publishing-to-specific-devices/

Trademarks