
pre market future Updating a pre-installed application

Posted by John Seghers - 2010/01/08 16:28

I had assumed that if an application was preloaded it could not be updated via the marketplace. My reasoning being:
1) A preloaded application is part of the OEM's ROM image. 2) The APK for the app would therefore be on the /system partition and thus be read-only 3) Therefore the only way to update such an application would be via a full update

=====

pre market future Updating a pre-installed application

Posted by Dianne Hackborn - 2010/01/08 16:28

I had assumed that if an application was preloaded it could not be updated via the marketplace. My reasoning being:
1) A preloaded application is part of the OEM's ROM image. 2) The APK for the app would therefore be on the /system partition and thus be read-only 3) Therefore the only way to update such an application would be via a full update

=====

pre market future Updating a pre-installed application

Posted by John Seghers - 2010/01/08 16:28

One more question about this. Our application is one that requires OEM integration to provide its full functionality and thus if someone downloaded it from the Market to a phone on which it is not normally pre-installed, it would not function properly for them. Is there a way to place the update on the marketplace such that it will only be available to phones on which it was originally preloaded?

=====

pre market future Updating a pre-installed application

Posted by Dianne Hackborn - 2010/01/08 16:28

One more question about this. Our application is one that requires OEM integration to provide its full functionality and thus if someone downloaded it from the Market to a phone on which it is not normally pre-installed, it would not function properly for them. Is there a way to place the update on the marketplace such that it will only be available to phones on which it was originally preloaded?

=====

pre market future Updating a pre-installed application

Posted by aika.n - 2010/01/08 16:28

One more question about this. Our application is one that requires OEM integration to provide its full functionality and thus if someone downloaded it from the Market to a phone on which it is not normally pre-installed, it would not function properly for them. Is there a way to place the update on the marketplace such that it will only be available to phones on which it was originally preloaded?

=====

pre market future Updating a pre-installed application

Posted by aika.n - 2010/01/08 16:28

Can somebody please confirm whether this functionality is available since 1.5 or only since 1.6? Also, if we're building an app for a device that will be launched with 1.5 OS, then if we preload a signed app, will the marketplace be able to notify for an upgrade once the OS is upgrade to 2.0? Happy holidays, everyone and I hope to get some help with this matter. thanks, aika

=====

pre market future Updating a pre-installed application

Actually, if your app needs some functionality from the device that is not part of the platform, the proper way to do this is to put it in a shared library like this that the application uses (and the OEM implements). This may just be what you want to do anyway. Our application doesn't actually need anything specific provided by the OEM for the app to run, but our app provides a service and content provider that the OEM then uses to obtain information. Since this integration is a key element of our product, it doesn't make sense to put it on a handset that doesn't have the OEM-level support. So I've created a project with a single class `com.cequint.platform.CequintConfig`. I've run `dx` on the resulting class files and packaged it in to a JAR file with a `MANIFEST.MF` similar to that in other .jar files I see in `/system/framework`. Indeed, I just copied the `MANIFEST.MF` from `monkey.jar`. So my jar file is named `com.cequint.platform.jar` and contains: `/META-INF/MANIFEST.MF /classes.dex`. This jar file layout is the same for `monkey.jar` and `com.google.android.maps.jar`. I then remounted `/system` on an OEM development phone and pushed `com.cequint.platform.jar` into the `/system/framework` directory. I added `<uses-library android:name=com.cequint.platform />` to a test application's `<application>` section. When I try to install the test application, I get an error from `adb install`: `Failure`. What am I missing here?

=====

pre market future Updating a pre-installed application

Actually, if your app needs some functionality from the device that is not part of the platform, the proper way to do this is to put it in a shared library like this that the application uses (and the OEM implements). This may just be what you want to do anyway. Our application doesn't actually need anything specific provided by the OEM for the app to run, but our app provides a service and content provider that the OEM then uses to obtain information. Since this integration is a key element of our product, it doesn't make sense to put it on a handset that doesn't have the OEM-level support. So I've created a project with a single class `com.cequint.platform.CequintConfig`. I've run `dx` on the resulting class files and packaged it in to a JAR file with a `MANIFEST.MF` similar to that in other .jar files I see in `/system/framework`. Indeed, I just copied the `MANIFEST.MF` from `monkey.jar`. So my jar file is named `com.cequint.platform.jar` and contains: `/META-INF/MANIFEST.MF /classes.dex`. This jar file layout is the same for `monkey.jar` and `com.google.android.maps.jar`. I then remounted `/system` on an OEM development phone and pushed `com.cequint.platform.jar` into the `/system/framework` directory. I added `<uses-library android:name=com.cequint.platform />` to a test application's `<application>` section. When I try to install the test application, I get an error from `adb install`: `Failure`. What am I missing here?

=====