# Making Your Application Manifest Compatible with TV

When developing Android applications for TV, it's important to set up the manifest to reflect your hardware feature requirements and permissions precisely. An incorrectly defined manifest might cause your application to not appear in Google Play on TV devices. This document shows how to avoid common mistakes when setting up your manifest file.

## The Basics

By default, Google Play assumes that an application requires hardware touchscreen support. If you want your application to appear in the Google Play Store for TV devices, you must specify in your manifest file (`AndroidManifest.xml`) that your application does not require touchscreen support. To do this, add the following element specification exactly as it appears:

```
<uses-feature android:name="android.hardware.touchscreen" android:required="false"/>
```

Your application must not have any requirements for multitouch in your manifest file, as described in the table below.

## Unsupported Hardware

The touch screen is not the only hardware feature that is often missing from TV devices. If your application wants to make use of this hardware on other devices, please use `android:required="false"`. You can check for features at runtime by calling `PackageManager.hasSystemFeature()`. The method takes a single argument, a string corresponding to the feature you want to check. For example, to test for a touchscreen, use `hasSystemFeature()` with the argument `FEATURE_TOUCHSCREEN`.

Here is the hardware to **not** rely on:

| Hardware | Feature descriptor[1] |
|---|---|
| Bluetooth | android.hardware.bluetooth |
| Camera | android.hardware.camera |
| | android.hardware.camera.autofocus |
| | android.hardware.camera.flash |
| | android.hardware.camera.front |

[1] The value of the android:name attribute in the <uses-feature> element.

| | |
|---|---|
| Location (GPS) | android.hardware.location.gps |
| Microphone | android.hardware.microphone |
| Near-Field Communications(NFC) | android.hardware.nfc |
| Sensors | android.hardware.sensor.accelerometer |
| | android.hardware.sensor.barometer |
| | android.hardware.sensor.compass |
| | android.hardware.sensor.gyroscope |
| | android.hardware.sensor.light |
| | android.hardware.sensor.proximity |
| Telephony | android.hardware.telephony |
| | android.hardware.telephony.cdma |
| | android.hardware.telephony.gsm |
| Touchscreen | android.hardware.faketouch |
| | android.hardware.touchscreen |
| | android.hardware.touchscreen.multitouch |
| | android.hardware.touchscreen.multitouch.distinct |
| | android.hardware.touchscreen.multitouch.jazzhand |

## Implied Features Based Upon Permissions

Your application's permission settings may imply that certain hardware components are required, even if these components are not explicitly declared in your manifest.  Google Play will kindly assume that the hardware will be required unless you tell it otherwise.

The reason for this is that some of the hardware feature constants listed in the table above were made available to applications after the corresponding API were introduced. For example, the `android.hardware.bluetooth` feature was added in Android 2.2 (API level 8), but the Bluetooth API that it refers to was added in Android 2.0 (API level 5). Because of this, some apps were able to use the API before they had the ability to declare that they require the API via the `<uses-feature>` system.

To prevent those apps from being made available unintentionally, Google Play assumes that certain hardware-related permissions indicate that the underlying hardware features are required by default. For instance, applications that use Bluetooth must request the BLUETOOTH permission in a `<uses-permission>` element — for legacy apps, Google Play assumes that the permission declaration means that the underlying android.hardware.bluetooth feature is required by the application and sets up filtering based on that feature.

The table below lists permissions that imply feature requirements equivalent to those declared in `<uses-feature>` elements. Note that `<uses-feature>` declarations, including any declared `android:required` attribute, always take precedence over features implied by the permissions below.

For any of the permissions below, you can disable filtering based on the implied feature by explicitly declaring the implied feature in a `<uses-feature>` element with an `android:required="false"` attribute. For example, to disable any filtering based on the CAMERA permission, you would add this `<uses-feature>` declaration to the manifest file:

```
<uses-feature android:name="android.hardware.camera" android:required="false"/>
```

| This Permission... | Implies This Feature Requirement |
|---|---|
| BLUETOOTH | android.hardware.bluetooth |
| BLUETOOTH_ADMIN | android.hardware.bluetooth |
| CAMERA | android.hardware.camera *and* android.hardware.camera.autofocus |
| ACCESS_MOCK_LOCATION | android.hardware.location |
| ACCESS_LOCATION_EXTRA_COMMANDS | android.hardware.location |
| INSTALL_LOCATION_PROVIDER | android.hardware.location |

| | |
|---|---|
| ACCESS_COARSE_LOCATION | android.hardware.location.network *and* android.hardware.location |
| ACCESS_FINE_LOCATION | android.hardware.location.gps *and* android.hardware.location |
| RECORD_AUDIO | android.hardware.microphone |
| CALL_PHONE | android.hardware.telephony |
| CALL_PRIVILEGED | android.hardware.telephony |
| MODIFY_PHONE_STATE | android.hardware.telephony |
| PROCESS_OUTGOING_CALLS | android.hardware.telephony |
| READ_SMS | android.hardware.telephony |
| RECEIVE_SMS | android.hardware.telephony |
| RECEIVE_MMS | android.hardware.telephony |
| RECEIVE_WAP_PUSH | android.hardware.telephony |
| SEND_SMS | android.hardware.telephony |
| WRITE_APN_SETTINGS | android.hardware.telephony |
| WRITE_SMS | android.hardware.telephony |
| ACCESS_WIFI_STATE | android.hardware.wifi |
| CHANGE_WIFI_STATE | android.hardware.wifi |
| CHANGE_WIFI_MULTICAST_STATE | android.hardware.wifi |

## Conclusion

Understanding implied features and TV device feature limitations will allow you to tailor your manifest to target the widest range of devices that your software can support.