

# Best practice tips for delivering apps for the Lumia 520

 **Deprecated:** This article is deprecated because its content has been incorporated into a new article [How to adapt to Lumia phones' hardware features](#) available in the [Nokia Lumia Developer's Library](#)

This article provides recommended best practices for preparing your Windows Phone apps for the [Nokia Lumia 520](#).

## Introduction



The [Nokia Lumia 520](#) does not have all of the hardware features found on more expensive devices, including: front camera, rear camera flash, gyroscope, compass and NFC. As a result there are some apps where it will be necessary to drop functionality, and others where the app use case can't be supported at all.

When a particular sensor is essential to the operation of the application you can opt-out (prevent installation of the app on the device) by specifying the sensor's hardware Id as a requirement in the app's manifest file. If the app can still function without the sensor then you can make run time checks to disable the functionality on less capable devices.

This article provides some best practice tips and techniques how to deal with the missing sensors to make sure that your apps will run smoothly across all the devices of the Windows Phone 8.0 portfolio.

## Lumia 520 sensor list

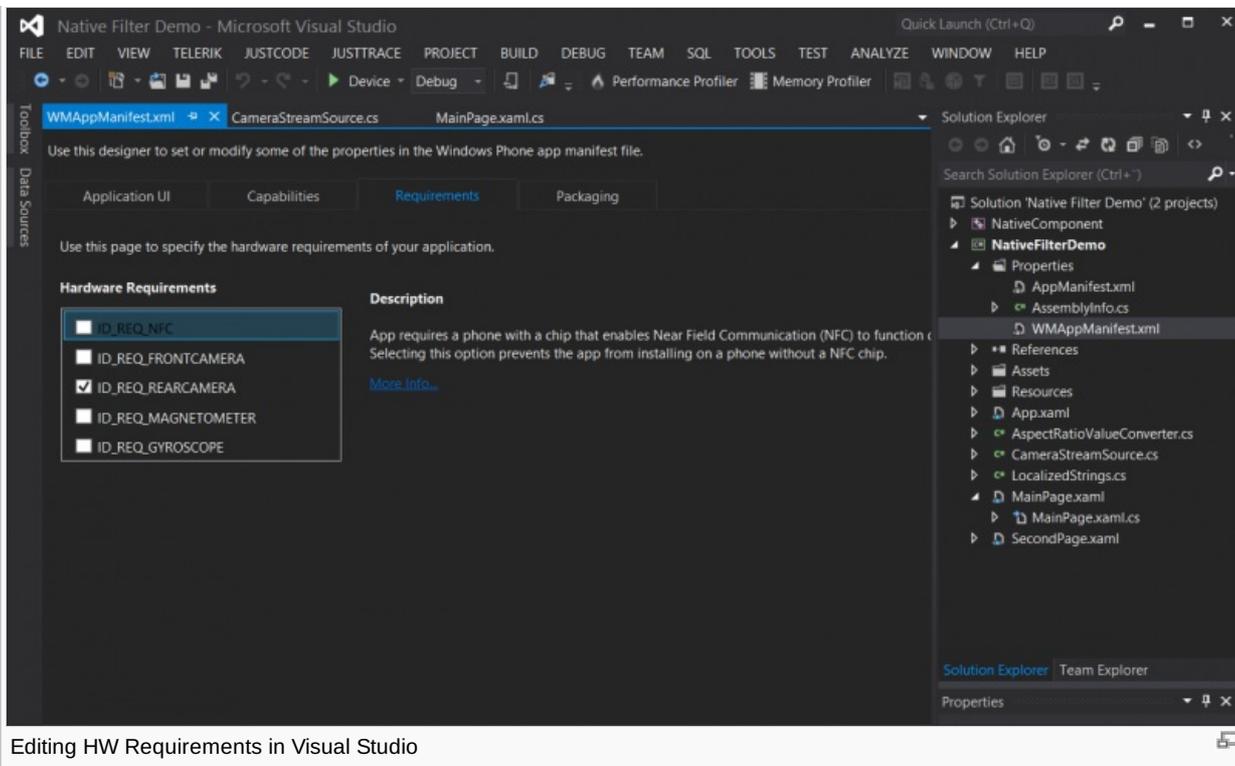
Sensor	Present on Lumia 520	HW Requirement
Accelerometer	Yes	
Back Camera	Yes	
Compass	No	ID_REQ_MAGNETOMETER
Flash for the Back Camera	No	
Front Camera	No	ID_REQ_FRONT_CAMERA
GPS	Yes	
Gyroscope	No	ID_REQ_GYROSCOPE
NFC	No	ID_REQ_NFC

## Preventing installation on unsupported devices ("opting out")

If a sensor is *essential* to the operation of your app then you can remove it as a Windows Store option for all devices without the sensor. Some examples of typical use cases that can't be supported by the Lumia 520 include:

- Video telephony
- Mirror application using the front camera
- Augmented reality application where the apps need to know the phone direction (North, South, East, West)
- Writing or reading content to/from an NFC tag
- Unlocking a game level, using an NFC tag

To ensure that the app will only be presented for installation to devices with the required hardware, specify the [HW Requirement](#) for the sensor in the application manifest file (**WMAppManifest.xml**). This can be done in Visual Studio as shown below.



Editing HW Requirements in Visual Studio

You can find more information about HW requirements in the Dev Center topic: [App capabilities and hardware requirements for Windows Phone](#).

## How to check for sensors at runtime

Applications where sensors are only used to *enhance* the user experience should check for sensor availability and disable any code/menu options which use the sensor. This will ensure that the application doesn't crash or propose an option that will do nothing due to missing sensors.

**Note:** If you are checking for sensors at runtime, remember not to also specify the HW Requirement in your manifest file

For example a game with NFC "Tap and share your score" functionality can still be a great game if this sharing method is unavailable, and in this case you would check for the presence of NFC and disable/enable the menu option as necessary. Real-world examples of such applications include [Burton](#) and [YouSendIt](#)

The code used to perform the run-time availability checks is sensor-dependent, as shown in the following sections.

### NFC / Tap to Share

```
var nfc = ProximityDevice.GetDefault();
if (nfc == null)
{
    // There is no HW support for tap and share (NFC)
}
```

### Front camera

```
var sensors = PhotoCaptureDevice.AvailableSensorLocations;
if (!sensors.Any(n => n == CameraSensorLocation.Front))
{
```



```
// This device doesn't have a front camera
```

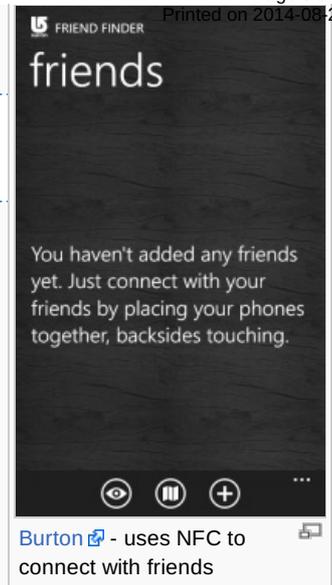
```
};
```

## Flash for the back camera

```
var flashModes =
```

```
PhotoCaptureDevice.GetSupportedPropertyValues(CameraSensorLocation.Back,
        KnownCameraPhotoProperties.FlashMode);

if (flashModes.Count == 1)
{
    // Count ==1, only FlashState.Off is supported -> Flash is not available on this
    device.
}
}
```



## Compass

```
_compass = Compass.GetDefault();
if (_compass == null)
{
    // The compass is not available on this device
}
}
```

## Gyroscope

```
_gyrometer = Gyrometer.GetDefault();
if (_gyrometer == null)
{
    // The gyroscope is not available on this device
}
}
```

## The Motion API

Windows Phone 8.0 provides the [Microsoft.Devices.Sensors.Motion](#) class, which aggregates data from several sensors (accelerometer, compass, gyroscope) and returns high level information about the phone's orientation and motion. The Motion API will not function on devices where the compass is absent.

Use the following code if you want to check the availability of the motion API at run-time:

```
if (!Microsoft.Devices.Sensors.Motion.IsSupported)
{
    // Motion class cannot be used on this device.
}
}
```

</div>