Table of content

1 Prologue .................................................................................................................................2
2 Introduction ............................................................................................................................3
3 Add XLabs to project (VS2013 - Update 2) .........................................................................4
   3.1 Initial installation ..................................................................................................................4
   3.2 Installing updates automatically .......................................................................................5
   3.3 Installing specific versions over the NuGet-Console ..........................................................6
4 Using XLabs ............................................................................................................................9
   4.1 Using a control ....................................................................................................................9
   4.2 Register a Service ..............................................................................................................10
      4.2.1 Android .......................................................................................................................10
      4.2.1.1 My MainActivity.cs before adding XLabs ..............................................................10
      4.2.1.2 Code to add ...............................................................................................................11
      4.2.1.3 My MainActivity.cs after adding XLabs .................................................................11
      4.2.2 iOS .............................................................................................................................12
      4.2.2.1 My AppDelegate.cs before adding XLabs ..............................................................12
      4.2.2.2 Code to add ...............................................................................................................12
      4.2.3 My AppDelegate.cs after adding XLabs .................................................................13
      4.2.4 Windows-Phone ........................................................................................................14
      4.2.5 My MainPage.xaml.cs before adding XLabs ............................................................14
      4.2.6 Code to add ...............................................................................................................15
      4.2.7 My MainPage.xaml.cs after adding XLabs ...............................................................15
   4.3 Using the services (example to geolocation- and device-service) ....................................16
      4.3.1 Usings .........................................................................................................................16
      4.3.2 Geolocation Service ..................................................................................................16
      4.3.3 Device-Service .........................................................................................................17
   4.4 Important notes ..................................................................................................................19
      4.4.1 Possible hurdles ..........................................................................................................19
      4.4.2 Windows Phone .........................................................................................................20
      4.4.3 Android .....................................................................................................................22

M:\Allgemeines_MATRIX\Internet_Strategie\Mobile_Entwicklung\Xamarin\XLabs\How_to_Install_and_Setup_XLabs.docx
1 Prologue

As the installation and use of XLabs is not self-explaining and there is no clear documentation available, I have wrote this document (parallel to implement some functions from XLabs) in the hope, this may help some others like me 😊.

Unfortunately, (at the time, I wrote this document) there are only small portions of information's available. On the XLabs-page there only are information’s to the “old” Forms.Labs (new: XLabs) implementation available (what can be misleading additionally).

I think further, that much of the interested users are not hard-core-developers, that have maximum experience with Xamarin and can follow the “snipped-description’s”. I think that much of the developers simply want to enhance standard-XF with functions of XLabs and can be overwhelmed by the hurdles to install and use it, what is very pity for the awesome work in this extension.

As I don’t use very much from XLabs, this is not a full description to XLabs, but you should be able to take the first hurdles (install and start) with it…

I work with VS (2013 - Update2) and my app is based on the template “Blank App (Xamarin.Forms Shared).”

So… the documentation is optimal for VS-developers that works with a shared-project (if you use another project-type, you may have to implement it differently).

My actual environment (when I have wrote this document):

VS2013 Update 2 / Xamarin-Forms 1.4.1-Pre-1, XLabs 2.0.5546.3567

This document was written at March 17, 2015. So the information’s may be old, if you read it 😊.

My native language is German I hope you understand this document nevertheless and… if you find some “English-bug’s” (or others), you can keep it for free… 😊😊

Finally, I hope this document helps you and... thanks for reading it…
2 Introduction
XLabs is an add-on to Xamarin.Forms. The target is to enhance missing functions in Xamarin-Forms. XLabs is free and powered thru some community-members, which all works for free.

So... I first want to thanks here the XLabs-team for their awesome work!

In XLabs, there are some controls and some services included.

My personal intent to use XLabs, was:

- **The PopUp-Control:**
  - In my app, I have heavy search-pages in a ScrollView.
  - Some search-items have to be selected over ListViews (e.g. city/zip-code with more then 4'000 items)
  - As ListView's can't be served from a ScrollView in Android since a few XF-Versions ago (and - it seems - generally should not be used directly in a ScrollView), I had to find a way to change my whole logic to my search-pages).
  - As I don't want to call separate pages for every ListView, I have searched another solution and found it in the **PopUp-Control** of XLabs.
  - With the PopUp-Control, I was able to change my code, so that I now can use my ListViews from a ScrollView without problems.
  - As there is not description to the PopUp-Control is available yet on the XLabs-Page, I have searched another .pdf with a description, especially to the PopUp-Control.

- **Some of the included services:**
  - Device (device information’s, access to media-files and camera)
  - Maybe Geolocation-Service

Of course, you may have other preferences - you can check-out the available controls / services and the latest information’s over the links below.

**XLabs-Links:**
- Wiki:  https://github.com/XLabs/Xamarin-Forms-Labs/wiki
- GitHub:  https://github.com/XLabs/Xamarin-Forms-Labs

**Important notes:**
- This documentation is based on XLabs 2.x and XF 1.4.1 with unified iOS.
- If you don't have migrated to unified API, I suggest you to do it first.
- I you already have migrated to unified, I recommends you, to control the references of your iOS-Project (they should be on: `lib/Xamarin.iOS10` (not on `lib/MonoTouch10`)
- I have lost full two days to sort the problems out...
  - *See also chapter “Important notes”.*
3 Add XLabs to project (VS2013 - Update 2)

3.1 Initial installation
My XF-project is based on the template "Blank App (Xamarin.Forms Shared)".
Note: I was on XF 1.3.4-pre-x, when I have installed XLabs the first time.

- Select your root-project in the solution-manager (1) and call the NuGet-Manager (2)
- Type "XLabs" in the search-bar (3) and select "Xlabs - Forms" (4) to install
  - Note: All depending packages then are installed automatically.

Note: There were some problems with dependencies between Xamarin.Forms and XLabs (the dependency of XLabs had to target exactly the installed XF-Version for WP-projects). This problem has been gone since XF-version 1.4 (as Xamarin has changed the WP-Implementation in XF)

- So... If you have installed a XF-Version >= 1.4, your don't should have problems 😊
  - Further possible problems (especially with iOS) see chapter “Important notes”
- And also all references in all sub-projects (iOS, Android and WP) are added automatically (5)

- Done… the XLabs-packages are now ready to use.
- The XLabs-controls (like the PopUp) can now be used by simple add a using on the page:
  ```csharp
  using XLabs.Forms.Controls;
  ```
3.2 Installing updates automatically

- If you start the NuGet-Manager, you can click the Tab “Aktualisierungen” (“updates”) and then can see **updates** to all **already** installed packages.
- In the ComboBox on top you can change between stable-versions and alpha/beta-versions:

  ![ComboBox](image)

  - In the example, you can see, that there is an update to XLabs available
  - If you select an Item, you can see the version-information’s on the right side:
  - To update the package, just click on “Aktualisieren” (Update)-Button
  - Note: If you select the XLabs - Forms package, all XLabs-Packages should be updated automatically, as there are dependencies set

  - This is the fastest and easiest way to update...
  - But... maybe you have to install a specific-version, that is not showed under updates...?
  - In this case, you have to install the package over the NuGet-Console (see next page)
3.3 Installing specific versions over the NuGet-Console

Note: This short-description can be used for every NuGet-Package (not only XLabs), e.g. also for Xamarin.Forms itself.

First, you should have a look, which versions are available and how to install them. Therefore you load the NuGet-Portal:

- Link: [https://www.nuget.org/](https://www.nuget.org/)

Search for **XLabs** returned 20 packages

Displaying results 1 - 20.  
Sorted by relevance

- **XLabs - Platform** by ravensorb sam1971 marinho
  XLabs is a open source project that aims to provide a powerful and cross platform set of controls tailored to work with Xamarin.Forms. This package contains the core cross-platform framework for all of XLabs.
  2,570 total downloads  | Tags: Xamarin.Forms XLabs Platform

- **XLabs - Core** by ravensorb sam1971 marinho
  XLabs is a open source project that aims to provide a powerful and cross platform set of controls tailored to work with Xamarin.Forms. This package contains the the core components of the XLabs Forms framework and controls.
  2,593 total downloads  | Tags: Xamarin.Forms XLabs

- **XLabs - Forms** by ravensorb sam1971 marinho
  XLabs is a open source project that aims to provide a powerful and cross platform set of controls tailored to work with Xamarin.Forms. This package contains the cross-platform XLabs Forms framework and controls.
  2,575 total downloads  | Tags: Xamarin.Forms XLabs

- **XLabs - Caching** by ravensorb sam1971 marinho
  XLabs is a open source project that aims to provide a powerful and cross platform set of controls tailored to work with Xamarin.Forms. This package contains the caching system of the core controls.

- Type in the name of the package, you want to install (1)
- In the result, click on a package (in the example, the XLabs - Forms - package (2) was selected)
Now, NuGet shows you all available version's to the package (3)

In the box on top, you can see the command (5), that has to be used to install the selected package (4) in the NuGet-Console (see next page)

If you change the selection to another version (6), also the command is changed automatically (7)
To install the package, call “Extras (Tools)” (8) - “NuGet-Paket-Manager” (8) - “Paket-Manager-Konsole” (10) from the VS-Menu:

Then, the Paket-Manager-Konsole ist loaded:

Notes:
- You have to select the target-project in the ComboBox (11)
- You have to install the package for every project (in the example for the .Android, the .iOS and the .WinPhone-project)

Now just paste the command from NuGet (see (5) / (7)) to the command prompt (12)

...and finally press Enter (13)
... done 😊
4 Using XLabs

4.1 Using a control
If you only want to use some XLabs-Controls (no services):
Just enhance your page (where the control should be used) with the using:

```csharp
using XLabs.Forms.Controls;
```

And start to implement it (e.g.):

```csharp
var oExLabel = new ExtendedLabel();
oExLabel.IsDropShadow = true;
oExLabel....
```
4.2 Register a Service
As soon as you want to use a service, you have to register it first for every platform.
As I use the "device-service" the following description is related to the device-service - further, I have added the GeoLocation-Service.
You should be able to add further services in the same way without problems.

4.2.1 Android
In your solution, select the Android-project (1) and open MainActivity.cs (2)

4.2.1.1 My MainActivity.cs before adding XLabs

```csharp
using System;
using Android.App;
using Android.Content;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Xamarin.Forms.Platform.Android;
using Android.Content.PM;

namespace MatrixGuide.Droid
{
    [Activity(Label = "MatrixGuide", ConfigurationChanges = ConfigChanges.Orientation | ConfigChanges.ScreenSize)]
    public class MainActivity :
        global::Xamarin.Forms.Platform.Android.FormsApplicationActivity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            Xamarin.Forms.Init(this, bundle);
            Xamarin.FormsMaps.Init(this, bundle);
            LoadApplication(new App());
        }
    }
}
```
4.2.1.2 Code to add

To register a service, you first have to add a SimpleContainer (included in Xabs). Then you have to register the services, you want to use.

Usings:

```csharp
using XLabs.Ioc; // For the X Labs simple container
using XLabs.Platform.Services.Geolocation; // For the GeoLocation-Service
using XLabs.Platform.Device; // For the Device-Service
```

Code:

```csharp
var container = new SimpleContainer(); // Create a SimpleContainer
container.Register<IGeolocator, Geolocator>(); // Register the Geolocator
container.Register<IDevice> (t => AndroidDevice.CurrentDevice); // Register the Device Service
Resolver.SetResolver(container.GetResolver()); // Resolve it
```

Maybe you want to full-qualify the objects. Example for Geolocator:

```csharp
```

4.2.1.3 My MainActivity.cs after adding X Labs

```csharp
using System;
using Android.App;
using Android.Content;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Xamarin.Forms.Platform.Android;
using Android.Content.PM;
// New X labs
using XLabs.Ioc; // Using for SimpleContainer
using XLabs.Platform.Device; // Using for Display
// End new X labs
namespace MatrixGuide.Droid
{
    [Activity(Label = "MatrixGuide", ConfigurationChanges = ConfigChanges.Orientation | ConfigChanges.ScreenSize)]
    public class MainActivity :
        global::Xamarin.Forms.Platform.Android.FormsApplicationActivity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            // New X labs
            var container = new SimpleContainer();
            container.Register<IDevice> (t => AndroidDevice.CurrentDevice);
            container.Register<IGeolocator, Geolocator>();
            Resolver.SetResolver(container.GetResolver()); // Resolving the services
            // End new X labs

            Xamarin.Forms.Forms.Init(this, bundle);
            Xamarin.FormsMaps.Init(this, bundle);
            LoadApplication(new App());
        }
    }
}
```
4.2.2 iOS
In your solution, select the iOS-project (1) and open **AppDelegate.cs** (2)

4.2.2.1 My AppDelegate.cs before adding XLabs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using Foundation;
using UIKit;
using Xamarin.Forms;

namespace MatrixGuide.iOS
{
    [Foundation.Register("AppDelegate")]
    public partial class AppDelegate :
        global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
    {
        // class-level declarations
        UIWindow window;
        //
        public override bool FinishedLaunching(
            UIApplication app,
            NSDictionary options)
        {
            Forms.Init();
            Xamarin.FormsMaps.Init();
            LoadApplication(new App());
            return base.FinishedLaunching(app, options);
        }
    }
}
```

4.2.2.2 Code to add

To register a service, you first have to add a SimpleContainer (included in Xabs)
Then you have to register the services, you want to use

**Usings:**

```csharp
using XLabs.Ioc;  // Using for SimpleContainer
using XLabs.Platform.Device;  // Using for Device
```

**Code:**

```csharp
var container = new XLabs.Ioc.SimpleContainer();  // Create SimpleContainer
container.Register<IDevice>(t => AppleDevice.CurrentDevice);  // Register Device
container.Register<IGeolocator, Geolocator>();  // Register Geolocator
Resolver.SetResolver(container.GetResolver());  // Resolving the services
```
4.2.3 My AppDelegate.cs after adding XLabs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using Foundation;
using UIKit;
using Xamarin.Forms;

// New XLabs
using XLabs.Ioc; // Using for SimpleContainer
using XLabs.Platform.Device; // Using for Device

// End new XLabs

namespace MatrixGuide.iOS
{
    [Foundation.Register("AppDelegate")]
    public partial class AppDelegate :
        global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
    {
        UIWindow window;

        public override bool FinishedLaunching(UIApplication app,
            NSDictionary options)
        {
            // New XLabs
            var container = new XLabs.Ioc.SimpleContainer();
            container.Register<IDevice>(t => AppleDevice.CurrentDevice);
            container.Register<IGeolocator, Geolocator>();
            Resolver.SetResolver(container.GetResolver());
            // End new XLabs

            Forms.Init();
            Xamarin.FormsMaps.Init();
            LoadApplication(new App());
            return base.FinishedLaunching(app, options);
        }
    }
}
```
4.2.4 Windows-Phone
In your solution, select the Windows-Phone-project (1) and open MainPage.xaml.cs (2)

4.2.5 My MainPage.xaml.cs before adding XLabs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using Xamarin.Forms;
namespace MatrixGuide.WinPhone
{
    public partial class MainPage : global::Xamarin.Forms.Platform.WinPhone.FormsApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();
            SupportedOrientations = SupportedPageOrientation.PoratraitOrLandscape;
            global::Xamarin.Forms.Forms.Init();
            Xamarin.FormsMaps.Init();
            LoadApplication(new MatrixGuide.App());
        }
    }
}
```
4.2.6 Code to add

Using's:

```csharp
using XLabs.Ioc; // Using for SimpleContainer
using XLabs.Platform.Device; // Using for Device
```

Code:

```csharp
var container = new SimpleContainer(); // Create SimpleContainer
container.Register<IDevice>(t => WindowsPhoneDevice.CurrentDevice); // Register Device
container.Register<IGeolocator, Geolocator>(); // Register Geolocator
Resolver.SetResolver(container.GetResolver()); // Resolving the services
```

4.2.7 My MainPage.xaml.cs after adding XLabs

```csharp
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using Xamarin.Forms;
// New XLabs
using XLabs.Ioc;
using XLabs.Platform.Device;
// End new XLabs

amespace MatrixGuide.WinPhone
{
    public partial class MainPage :
        global::Xamarin.Forms.Platform.WinPhone.FormsApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();
            SupportedOrientations = SupportedPageOrientation.PhotraitOrLandscape;
            // New XLabs
            var container = new SimpleContainer();
            container.Register<IDevice>(t => WindowsPhoneDevice.CurrentDevice);
            container.Register<IGeolocator, Geolocator>();
            Resolver.SetResolver(container.GetResolver());
            // End new XLabs

            global::Xamarin.Forms.Forms.Init();
            Xamarin.FormsMaps.Init();
            LoadApplication(new MatrixGuide.App());
        }
    }
}
```
4.3 Using the services (example to geolocation- and device-service)
Especially behind the device-service, there is a huge functionality with various “sub-services”.
In the following, I have documented, the functionality (by inspect the objects). The example-values were queried with an Android-tablet SM-T900, connected via WLAN.
Note: I suggest you, to encapsulate the code in a try catch and check the objects for Null, when you implement it. By my tests, I further have noted some problems (crash) with the debugger (not on device). But.. this seems to be a problem with the Xamarin-VS-extensions.

➢ Please read also the chapter “Important notes”

4.3.1 Usings
Usings:
using XLabs.Platform.Device;
using XLabs.Platform;
using XLabs.Ioc;

4.3.2 Geolocation Service

```csharp
var oGeolocator = Resolver.Resolve<IGeolocator>(); // Resolve the Geolocator over the resolver
//oGeolocator.DesiredAccuracy property that gets or sets the Accuracy
//oGeolocator.GetPositionAsync() method to get the current position
//oGeolocator.IsGeolocationAvailable property: geolocation implemented on device?
//oGeolocator.IsGeolocationEnabled property: geolocation enabled?
//oGeolocator.IsListening property: Listening is actived?
//oGeolocator.PositionChanged event
//oGeolocator.PositionError event
//oGeolocator.StartListening() Method that starts the Listening
//oGeolocator.StopListening Method that stops the Listening
//oGeolocator.SupportsHeading property
```
4.3.3 Device-Service

```csharp
var device = Resolver.Resolve<IDevice>(); // Resolve the device over the resolver
var oDisplay = device.Display; // create display-interface

//
var lBreite = oDisplay.Width; // Returns EG: 2560
var lHoehe = oDisplay.Height; // Returns EG: 1600
var lXdpi = oDisplay.Xdpi; // Returns EG: 248.182998657227
var lYdpi = oDisplay.Ydpi; // Returns EG: 247.804000854492

//
var HW = device.HardwareVersion; // Returns EG: "universal15420"
var FirmwareVersion = device.FirmwareVersion; // Returns EG: "4.4.2"
var Manufacturer = device.Manufacturer; // Returns EG: "samsung"
var ID = device.Id; // Returns EG: "5205ce9c4b88215b"
var Memory = device.TotalMemory; // Returns EG: 2910535680
var DeviceName = device.Name; // Returns EG: "SM-T900"
var cTimeZone = device.TimeZone; // Returns EG: "Europe/Zurich"
var cLanguageCode = device.LanguageCode; // Returns EG: "de"

//Further methods to Device
//device.HeightRequestInInches() // method, that convert inches to RunTimePixel for Height
//device.WidthRequestInInches() // method, that convert inches to RunTimePixel for Width
//device.IsInLandscape() // method that returns, if the device is in Landscape-mode
//device.IsInPortrait() // method that returns, if the device is in Portrait-mode

//
var oNetwork = device.Network; // Create Interface to Network-functions

TimeSpan TSTimeOut = new TimeSpan(1000);
var NetworkAvailable = oNetwork.IsReachable("172.2.13.33", TSTimeOut); // Returns EG: Id = 53, Status = Running

//Further properties / methods / events:
//oNetwork.IsReachableByWifi()
//oNetwork.ReachabilityChanged event

var oBlueTooth = device.BluetoothHub; // Create Interface to BluetoothHub

//Further properties / methods / events:
//oBlueTooth.Enabled property
//oBlueTooth.GetPairedDevices()
//oBlueTooth.GetType()
//oBlueTooth.OpenSettings()

var oMicroPhone = device.Microphone; // Create Interface to Microphone

//Further properties / methods / events:
//oMicroPhone.BitsPerSample property
//oMicroPhone.SampleRate property
//oMicroPhone.SupportedSampleRates property
//oMicroPhone.ChannelCount property
//oMicroPhone.OnBroadcast event
//oMicroPhone.Start() method to start recording
//oMicroPhone.Stop() method to stop recording

var oAccelometer = device.Accelometer; // Create Interface to Accelerometer

//Further properties / methods / events:
//oAccelometer.Interval property
//oAccelometer.LatestReading property
//oAccelometer.ReadingAvailable event
```
var oBattery = device.Battery; // Create Interface to Battery
// Further properties / methods / events:
// oBattery.Charging property
// oBattery.Level property
// oBattery.OnChargerStatusChanged event
// oBattery.OnLevelChange event

var oFileManager = device FileManager; // Create Interface to FileManager
// Further properties / methods / events:
// oFileManager.CreateDirectory() method
// oFileManager.DirectoryExists() method
// oFileManager.FileExists() method
// oFileManager.GetLastWriteTime() method
// oFileManager.OpenFile() method

var oGyroscope = device.Gyroscope;
// Further properties / methods / events:
// oGyroscope.Interval property
// oGyroscope.LatestReading property
// oGyroscope.ReadingAvailable event

var oMediaPicker = device.MediaPicker;
// Further properties / methods / events:
// oMediaPicker.IsCameraAvailable property
// oMediaPicker.IsPhotosSupported property
// oMediaPicker.OnError event Gets or sets the error
// oMediaPicker.OnMediaSelected event
// oMediaPicker.SelectPhotoAsync() // method, that select an image from library
// oMediaPicker.SelectVideoAsync() // method, that select a video from library
// oMediaPicker.TakePhotoAsync() // method, that takes a Photo
// oMediaPicker.TakeVideoAsync() // method, that takes a Video

var oPhoneServices = device.PhoneService;
// oPhoneServices.CanSendSMS property
// oPhoneServices.CellularProvider property
// oPhoneServices.DialNumber() method
// oPhoneServices.ICC property gets the ISO-Country-code
// oPhoneServices.IsCellularDataEnabled property
// oPhoneServices.IsCellularDataRoamingEnabled property
// oPhoneServices.IsNetworkAvailable property
// oPhoneServices.MCC property: Gets the Mobile country-code
// oPhoneServices.MNC property: gets the mobile network-code
// oPhoneServices.SendSMS() method to send a SMS
4.4 Important notes

4.4.1 Possible hurdles
By my implementation, I had some permission-problems with Android and with WP (see next chapter) and huge problems with the iOS-project (that have cost me full two days):

First, I had a problem, that my project was not updated properly to the new iOS-unified API by the Xamarin-“migration-tool”:
“Self-explaining” error-message (translated from German to English):
“Lambda Expression cannot be translated to system.type as it is no delegate-type”

Then I had problems with the add-in Xamarin.Mobile (that I use for Geolocation)

Error-message:
Cannot include both 'monotouch.dll' and 'Xamarin.iOS.dll' in the same Xamarin.iOS project - 'Xamarin.iOS.dll' is referenced explicitly, while 'monotouch.dll' is referenced by 'Xamarin.Mobile, Version=0.7.1.0, Culture=neutral, PublicKeyToken=null'.

The problem is, that you have to use Xamarin.Mobile 0.7.5, what is not available via NuGet.

If you also have such problems, you can have a look at my posting in the forum here http://forums.xamarin.com/discussion/35928/information-problems-update-xf-to-unified-appledevice-xamarin-mobile-xlabs#latest
4.4.2 Windows Phone
Depending on what services you use, you have to enhance the rights for your Application. When I have implemented / tested the device-service in my WP-Project, I had e.g. the following crash:

- The app has crashed, by trying to get the device.id (1) (and further by trying accessing the device-Capabilities (like `device.MediaPicker.IsCameraAvailable`)
- In WP, there is fortunately a meaningful message showed in debugger (2), so I have seen, that I had to set additional permissions.

For WP, select your Windows Phone project (3) and open the File “WMAppManifest.xml” (4):

![Screenshot of WMAppManifest.xml file in Visual Studio](image-url)
Select the tab “Functions” (5) and set the needed permissions (6)
4.4.3 Android
To add permissions to Android:

- Select the Android-project (7) - right-click and select “properties”:

- Select Tab “Android Manifest” (8) and set the needed permissions (9)

- The settings then are stored in the file “AndroidManifest.xml” (10)