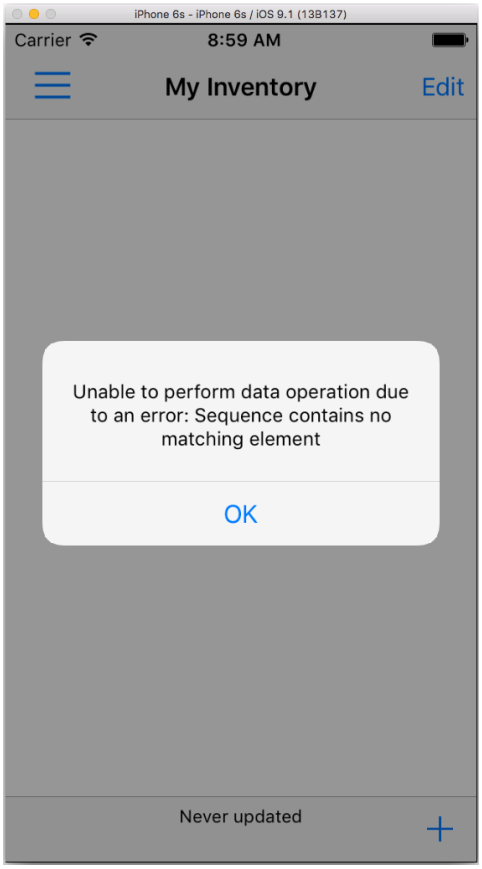
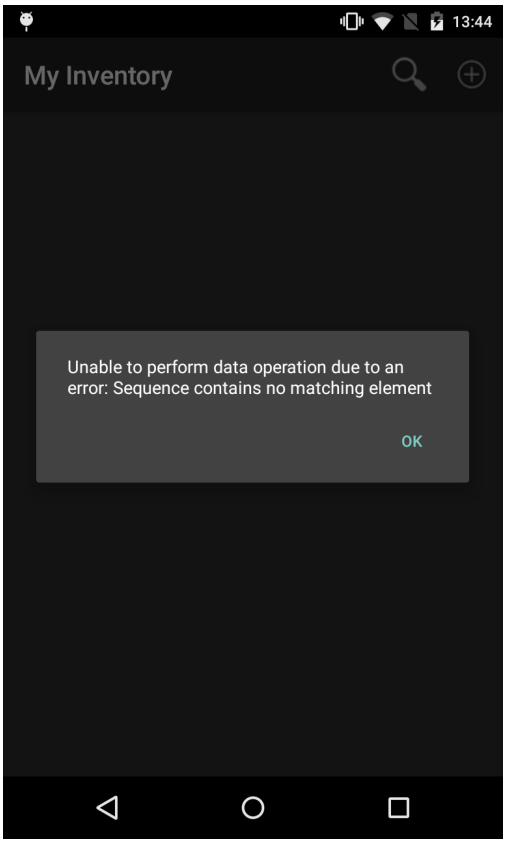
Configure Linker in Production-Ready Application Release Build

When building production in both iOS and Android, Xamarin method of deployment for iOS and Android to reduce size of the apps is the recommended way to optimized your application. However, when the linker is enabled, it may cause unexpected errors when the app tries methods which may not be directly referenced by the application, such as LINQ expressions.

If the LINQ is ignored, usually user will encounter this error: **Unable to perform data operation due to an error: Sequence contains no matching element**.



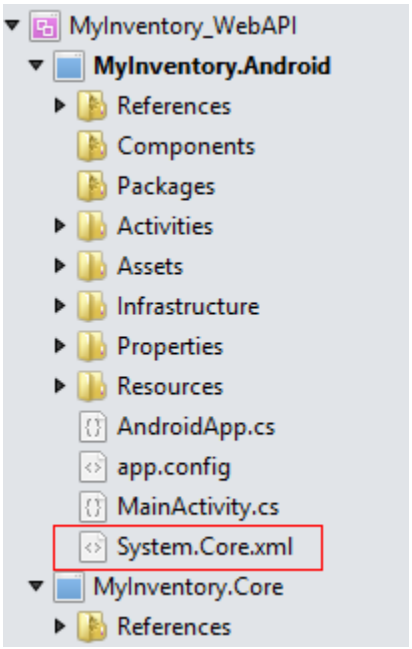


In order to use fully functional LINQ expressions, you will have to "tell" the Android/iOS linker to include the System.Xml.Linq assembly (do not exclude that assembly).

Here's how to enable the LINQ expressions in your production build:

Android

1. Download this file: System.Core.xml and put it directly in the root of your Android project so your project becomes:



Here's the content of System.Core.xml file.

System.Core.xml

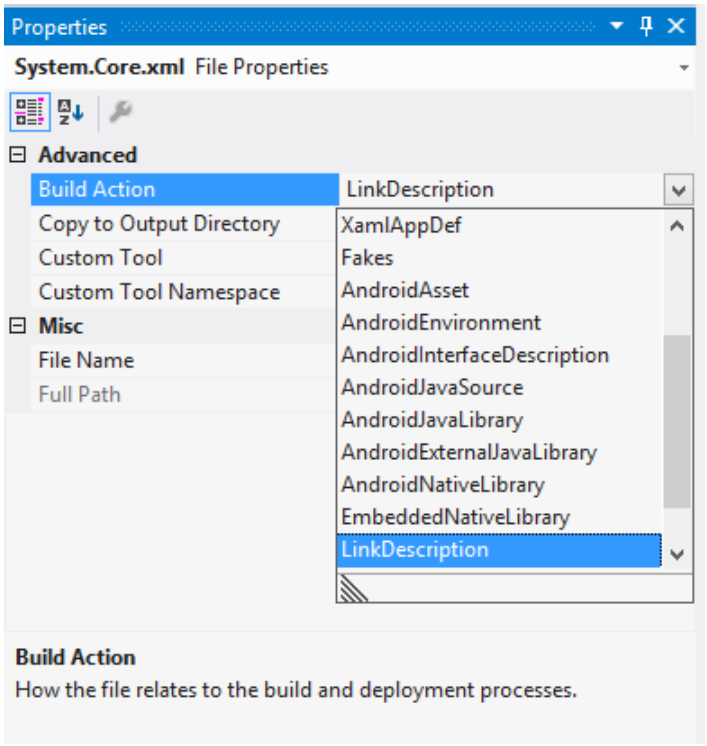
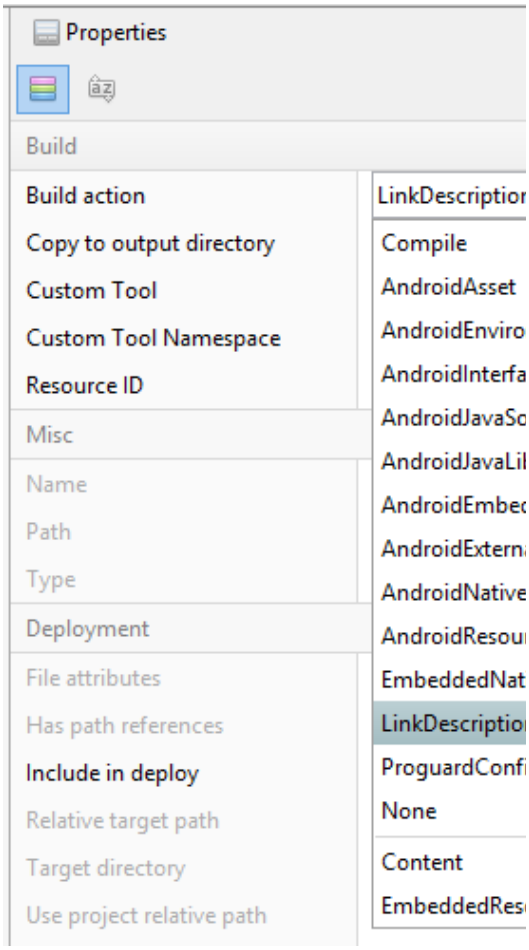
```
<?xml version="1.0" encoding="utf-8" ?>
<linker>
  <assembly fullname="System.Core">
    <type fullname="System.Linq.Expressions.Expression`1"></type>
    <type fullname="System.Linq.Queryable"></type>
  </assembly>
</linker>
```

In fact, you can name the .xml file as anything you want, for example: Assemblies.xml. Also, for Release build, you might need to include linkskip mscorlib as well.

System.Core.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<linker>
  <assembly fullname="System.Core">
    <type fullname="System.Linq.Expressions.Expression`1"></type>
    <type fullname="System.Linq.Queryable"></type>
  </assembly>
  <assembly fullname="mscorlib" />
</linker>
```

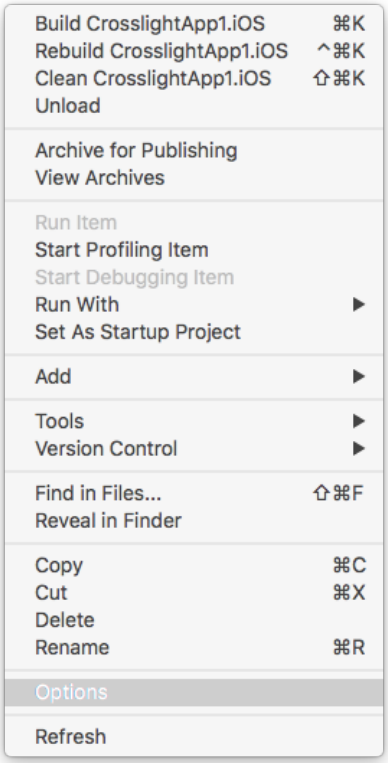
2. Change the Build Action of the .xml files into **LinkDescription** as shown in the following screenshots:

Visual Studio	Xamarin
	

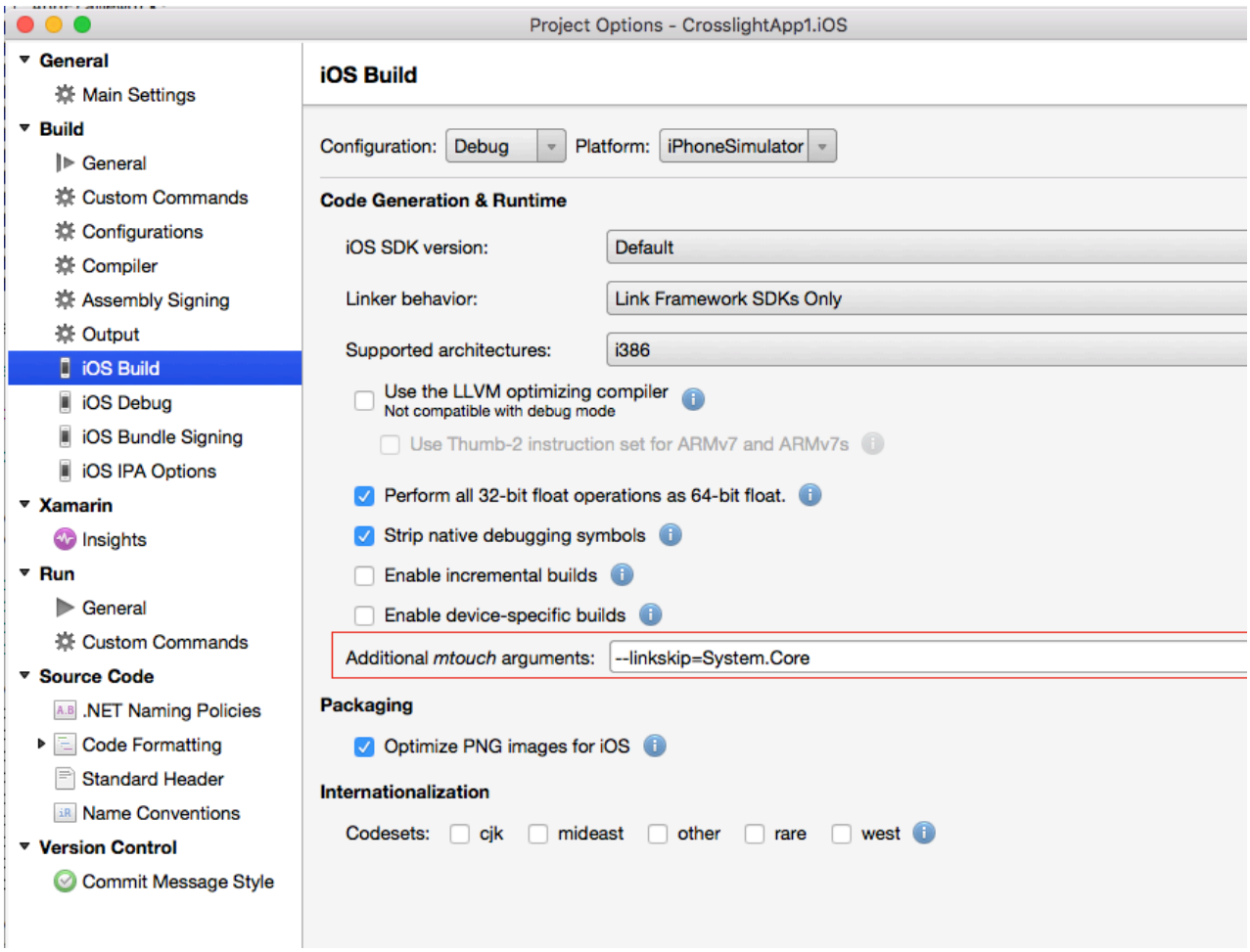
3. Afterwards, you can build your APK as usual and it will work as expected.

iOS

1. Right click your .iOS project and choose the Options



2. Navigate to the iOS Build Tab, change the linker behaviour to **"Link Framework SDKs Only" incase you do not yet**. Then type this inside the Additional mtouch arguments: **--linkskip=System.Core**


Project Options - CrosslightApp1.iOS
General
Main Settings
Build
General
Custom Commands
Configurations
Compiler
Assembly Signing
Output
iOS Build
iOS Debug
iOS Bundle Signing
iOS IPA Options
Xamarin
Insights
Run
General
Custom Commands
Source Code
.NET Naming Policies
Code Formatting
Standard Header
Name Conventions
Version Control
Commit Message Style
iOS Build
Configuration: Debug
Platform: iPhoneSimulator
Code Generation & Runtime
iOS SDK version: Default
Linker behavior: Link Framework SDKs Only
Supported architectures: i386
Use the LLVM optimizing compiler
Not compatible with debug mode
Use Thumb-2 instruction set for ARMv7 and ARMv7s
Perform all 32-bit float operations as 64-bit float.
Strip native debugging symbols
Enable incremental builds
Enable device-specific builds
Additional mtouch arguments: --linkskip=System.Core
Packaging
Optimize PNG images for iOS
Internationalization
Codesets: cjk mideast other rare west

3. Afterwards, you can build your application as usual and it will work as expected.
   If you wish you perform linkskip on multiple assemblies, use this:
   --linkskip=System.Core --linkskip=mscorlib

Customizing Your Own Linker

In case you want to create your own linker you can read this article by Xamarin:

https://developer.xamarin.com/guides/cross-platform/advanced/custom_linking/

Related Topics

- Configuring SQLite for Local Data Storage
- Create Crosslight Business Apps with Local Data Storage (SQLite)
- Displaying Simple List
- Using List as Navigation Interface
- Walkthrough: Getting Started with Crosslight Form Builder

39 related results