# iOS Native Extension Debugging Guide

Step through ActionScript and native code



© DiaDraw

### iOS Native Extension Debugging Guide, second edition by Radoslava Leseva

Find us on the web at http://easynativeextensions.com/

To report errors, please send a note to office@diadraw.com

Copyright © 2015 by DiaDraw

#### Project Editor: Hristo Lesev

Code Tester: Hristo Lesev Cover and Interior Design: Radoslava Leseva Proof Reader: Stephen Adams **Cover Photo:** Stephen Adams

#### Notice of Rights

All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author. For information on getting permission for reprints or excerpts, contact <u>office@diadraw.com</u>.

#### Notice of Liability

The information in this book is distributed on an "As Is" basis, without warranty. While every precaution has been taken in the preparation of the book, neither the author nor DiaDraw shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

#### Trademarks

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and DiaDraw was aware of a trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout this book are used in editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

# **Contents**

Introduction	2
What our readers say	3
What to expect	4
Got the code?	5
You will need the code for the extension	5
You will need a test app	5
Debugging the ActionScript side	6
Using Flash Builder 4.7?	6
Set up build scripts for your native extension	11
Set up build scripts for the test app	13
Set up a builder for the test app	14
Debugging the native side	17
Set up an application target in Xcode	18
Make the Xcode app target run your Flex Mobile app	21
Tell Xcode to run your app	22
Start debugging in Xcode	23
Diagnosing build problems	24
Misleading symptoms	24
Where to look instead	25
Compiler problems	25
Provisioning problems	25
Xcode's build log	26
Xcode's device console	27
If all else fails	28

### Acknowledgements and where to go from here

29

1

# Introduction

Being able to step through code is probably the most up-close and personal form of control you can achieve over the quality of your work.

No amount of automated testing, free lunches or health benefits for your testing team would give you the level of confidence that stepping through every line (or even instruction) of code and executing every fork in your logic can. Don't get me wrong: I am very much for testing any way you can. To solely rely on that to catch problems however, you have to anticipate them or apply a big dose of randomness. And when things get hairy, stepping through code provides a quick way to get to the source of pain. Compared to that most other alternatives are... well, like shooting in the dark.

If that was not enough to convince you, take it from **Steve Maguire**: although first published more than twenty years ago, his book <u>Writing Solid Code<sup>1</sup></u> has not dated one bit.

### What you need

- Mac OS X with Xode 6.0 and iOS SDK 8.0 or newer;
- Flash Builder 4.6 or 4.7 with Adobe AIR SDK 17.0 or newer;
   For instructions on how to overlay the AIR SDK on the Flex SDK see Adobe's article <u>Overlay AIR</u>
   <u>SDK on Flex SDK | Flash Builder</u>.
- the build scripts from the example project that comes with this book;
- your iOS device to be connected to your debugging machine via USB;
- for debugging ActionScript and if you are using Flash Builder 4.6: your debugger and your iOS device to be connected to the same network.

### Software editions used in the examples

The examples in this book have been tested with:

- Flash Builder 4.6 and 4.7
- AIR SDK 3.4 to 17.0
- Xcode 4.5 to 6.4, running on Mac OS X Lion, Mountain Lion and Yosemite
- iOS SDK 6.0 to 8.3

<sup>&</sup>lt;sup>1</sup> In the spirit of full disclosure: this is an affiliate link. This means that we may get a commission if you decide to purchase the book through the link above. This will not cost you extra and we only recommend books that we have read and live by, so we know you will be in good hands.

# What our readers say

"Well all I can say is that it was easily the best \$30 I've spent in a while. I was back up and running in under a day and as an added bonus Radoslava does a brilliant job of detailing how to wrap up the whole build, packaging, and deployment process into a single Ant script. Her build script was much better than the crummy one I remember cobbling together for my original ANE attempts. One of the things I hadn't really tackled previously was learning how to properly debug my ANEs. Thankfully a companion debugging guide came bundled with the package, which takes you through the steps required to write a native iOS project that will wrap around a test AIR app. With that you'll be able to add breakpoints to Xcode and inspect your ANE's native library." Read more...

#### **Christopher Caleb**

Author of Flash iOS Apps Cookbook

www.yeahbutisitflash.com

"Your ebook on iOS native extensions (at least the part of it that I've read so far) is the best, clearest communication about software development that I've ever read. It is rare and completely refreshing to find programmers who can speak just as fluently and cleverly in their 'natural' language as in code, and who can also use elegant graphics to clear-up abstractions. Bravo, and thanks!"

#### Craig Umanoff Moving Pictures

"<u>@DiaDrawCom</u> The book was great, very good tutorial and was easy to setup my custom ane. Thanks!"

#### @rudyvdblom

"Your eBook on ANE's is one of the best investments I have made. The excellent explanations and example code have saved me hours of trial-and-error. And I had planned to spend a couple of weeks developing custom email and dropbox extensions. Being able to download well-documented, ready-to-build source code gives me time to add extra features to my app. Very cool. Thanks."

#### Andrew Rapo

Quahog Entertainment

# What to expect

### At the end of this guide you will have:

- A set up for debugging your ActionScript native extension code.
- A set up for debugging your Objective-C native extension code.
- Confidence in your code.



24	FREObject ASSendMeAMessage(
25	<pre>FREContext ctx, void* funcData, uint32_t argc, FREObject argv[] )</pre>
20	FREDispatchStatusEventAsvnc( Thread 1: breakpoint 2.1
28	ctx,
29	<pre>( const uint8_t * ) "This is the message CODE",</pre>
30	( const uint8_t * ) "This is the message LEVEL" );
32	return NULL:
33	}
Ŧ	🗈 🔁 🛨 🛨 🚽 👘 AirMobileApp 👌 🥫 Thread 1 👌 🔽 0 ASSendMeAMessage

# Got the code?

### You will need the code for the extension

You are probably thinking 'Duh!'

This is just a reminder that if all you have is the extension ANE package that someone else developed, for example, but did n**o**t publish the code for, you will not be able to step through the code and debug it. You will need both the ActionScript library project and the Xcode library project and the source files that go with these.

Unless you are a reverse engineering whizz, of course. In this case drop me an email, I would like to buy you a cup of coffee and have a word: <u>radoslava.leseva@diadraw.com</u>.

### You will need a test app

An ANE is effectively a library. You can't run a library on its own and in order to be able to step through code, you need something that will run it. ANEs are written to be used by AIR applications and an application is definitely something you can run.

So you will need a test app. Two test apps, to be precise:

- to step through the ActionScript side of your ANE you will need to set up a Flex Mobile Project for iOS in Flash Builder. If you followed the tutorial in our Easy Native Extensions book, you will have already done that.
- to step through the **Objective-C** side you will need **Xcode** to be able to execute this test app and for that you have to set up an Xcode mobile app project that wraps around your Flex Mobile Project.

Sound complicated? Don't worry, you will get through it. One bite at a time.

# **Debugging the ActionScript side**

### Setting up a test app

If you followed the tutorial in our **Easy Native Extensions** book, you will have already set up a test app for your native extension. If you do not have a test app yet, no worries: **Step 4: Test the extension in an app**, in the <u>Easy Native Extensions book</u> will show you how to do it. It takes all of about 15 minutes.

### Using Flash Builder 4.7?

If your answer is 'yes', you can skip the rest of this chapter.

**Flash Builder 4.7** provides for **iOS** developers the conveniences that their **Android** brethren are used to and you don't have to do much in the way of gymnastics to start debugging over USB or over a network.

The Adobe Flash Builder 4.7 manual takes you through the process in detail: <u>Debug an application on an</u> <u>Apple iOS device</u>.

### Debugging with Flash Builder 4.6: the hard way

I should probably call this 'the two days of clicking saves two hours of reading' way, because it is not so much hard, as it is boring and prone to causing repetitive strain injuries.

You have probably been through it, experienced it and got sick of it, but here it comes. For completeness, you see...

With your Flex Mobile app project selected in Flash Builder's Package Explorer, click Run > Debug configurations... on the main menu or click the drop-down arrow next to the green bug on the tool bar and select Debug Configurations...



- In the **Debug Configurations** dialog select **Mobile Application** from the list on the right and click the **New** icon to add a new configuration.
- Under Project click Browse and select your test app project from the work space.
- Target platform should already be set to Apple iOS.
- Under Launch method select On device.
- Under Packaging method select Fast.
- Hit Debug.

$\bigcirc \bigcirc \bigcirc$	Debug Configurations
Create, manage, and run configur Debug a Mobile application.	ations Add a new mobile debug configuration
<ul> <li>type filter text</li> <li>Cesktop Application</li> <li>Video TestAppDesktop</li> <li>Java Applet</li> <li>Java Application</li> <li>Ju Junit</li> <li>Mobile Application</li> <li>AirMobileApp</li> <li>AirMobileApp Simulator</li> <li>Remote Java Application</li> <li>Ju Task Context Test</li> <li>Web Application</li> <li>AirMobileApp</li> <li>Video TestAppWeb</li> </ul>	Name:       AirMobileApp IOS         Image: Source       Common         Project:       AirMobileApp         Application file:       src/AirMobileApp.mxml         src/AirMobileApp.mxml       ‡         Target platform:       Apple iOS         Apple iOS       ‡         Launch method:       © On device:         Packaging method:       \$ Configure         On device:       Packaging takes several minutes, application performance is similar to a release build)         Image: Packaging takes several seconds, application runs significantly slower than a release build)         On Apple iOS, you will need to manually install and launch the application. [Learn more about deployment]         Configure packaging settings         Configure network debugging         Clear application data on each launch
Filter matched 19 of 23 items	Appy kevert
?	Close Debug

• You will be prompted to enter your Certificate Password:

00	Certificate Password				
Enter the certificate password for 'ios_developer_private_key.p12'. This password is required in order to package the application for Apple iOS.					
Password: Remember password for this session					
	Cancel OK				

• When Flash Builder finishes building and packaging your app, you should see this dialog giving you instructions. Follow these to install the app on your device:

10		
00	Waiting for Debugger Connection	
Packaging AirMo application, follo	obileApp for Apple iOS has completed successfully. To bo ow these steps:	egin debugging your
1. Mak 2. If th 3. Dray 4. Ensi 5. Syn 6. Lau	ke sure the device is connected via USB. he application was previously installed on your device, us ag and drop the .ipa package into iTunes library. ( <u>Reveal</u> sure the app is checked in the Apps tab for your device in ic the device using iTunes. anch the application on the device	ninstall it. <u>package in Finder</u> ) n iTunes.
Waiting for appli	ication to connect to debugger	
Learn more abou	ut Apple iOS deployment	Cancel

- Once the app is installed on your device, tap its icon to launch it.
- You have about **two minutes** to perform the steps above before the debugger loses its patience and shows you a **Launch Failed** alert:

0 0	O Launch Failed	
•	The Flash Builder debugger failed to connect to the running application.	
•	Ensure that:	
	1. For in-browser applications, you are running the debugger version of Flash Player.	
	<ol><li>For network debugging on a mobile device, you have a reliable network connection to the device, and port 7935 is open on your machine's firewall.</li></ol>	
	Device connection help	

Fear not! You can start the debugger again and now that your app has been built and installed on your device, all you have to do is run it. No need to repeat the installation steps again.

• When app and debugger have made contact, you can step through both the application and the ANE ActionScript code - just sprinkle breakpoints to taste.

Like this sample?

### Debugging with Flash Builder 4.6: the single-click way

The point of debugging your native extension is being able to step through changes you have just made or code you have just added. Being able to zero in on problems in the ANE too, of course.

### Waste of time and focus

Every time you make a change either to the ActionScript or to the native side of your extension, you will have to rebuild it, repackage the ANE and then rebuild and repackage the test app, so you can run it through the debugger and see the change in action. Having to do all of this and then follow the steps to install and run your app on a device for every change you made is, frankly, asking too much of a developer's time and concentration. I'm speaking from experience here. I have often found that by the time I had the debugger running I had forgotten what I last changed...

### The single-click way

If you could, would you rather click a button that would do all of the above for you? I know I would. Those of you who followed the tutorial in **Easy Native Extensions** (chapter **Making your life easier**) know where this is going: **Ant scripts**.

In fact, if you have set up your ANE and test app for a single-click build with the build scripts in that chapter, you have already done 99% of the work and you can jump to <u>Set up a builder for the test app</u>, where you will add the last little detail that you need to start debugging.

The rest of you, come with me. All you need is the build scripts in the example project that came with the book - I will show you how to use them for debugging.

#### Like this sample?

### Set up build scripts for your native extension

- Locate the **build.xml**, **build.properties** and **local.properties** files for the **Air Library** in the example project that came with the book (they should be in **NativeExtensionTutorial/Native Extension/AirLibrary/build scripts**) and copy them near your native extension code. What I have found to work best for me is putting the scripts in a subfolder in the Flash Builder library project of the extension, as that makes it easier to edit the files in Flash Builder.
- Open local.properties and set FLEX\_HOME to the path to where your Flex SDK is installed.

#### local.properties

```
FLEX_HOME=/Applications/Adobe Flash Builder 4.6/sdks/4.6.0
COMPC=${FLEX_HOME}/bin/compc
ADT=${FLEX_HOME}/bin/adt
```

- Open **build.properties** and set the following:
  - ane.name to the name of the ANE file you want to package.
  - **ane.destination** to the folder where you want the ANE file to be put. This folder is relative to the location of the **build.xml** script and doesn't have to exist yet: the build script will create it for you.
  - **iOS.library.name** to the name of your Xcode library and **iOS.library.root** to the path to your Xcode library project, relative to where the **build.xml** script is.
  - air.library.name to the name of your Flex Library project.
  - if you have a default implementation for your ANE, set **default.library.name** to the name of the Flex Library project for that and **default.library.root** to the path to that project folder, relative to where **build.xml** is.

Tip: See Adding support for the AIR Simulator chapter in the Easy Native Extensions book for what a default implementation is.

• If the file names and paths in your project are different from the defaults used in this tutorial, update these too.

#### build.properties

```
# Tips:
#
# 1. All paths set in this file, except iOS.library.builddir,
    are relative to where build.xml is.
# 2. You can override any of the values in this file by passing a new value
#
    on the command line, prefixed with -D, when you call ant.
#
    Example: ant -Dbuild.debug=false
# Set this to true for a Debug build and to false for a Release build
build.debug=true
# File name and folder for the packaged ANE:
ane.name=AirLibrary
ane.destination=../../ane
# XCode project paths:
iOS.library.name=NativeExtensionTemplateiOS
iOS.library.root=../../iOS/${iOS.library.name}
iOS.library=lib${iOS.library.name}.a
#Note that iOS.library.builddir is relative to the Xcode project folder:
iOS.library.builddir=${iOS.library.root}/build
# AIR Library paths:
air.library.name=AirLibrary
air.library.root=../
air.library.sourcedir=${air.library.root}/src
air.extension.descriptor=${air.library.name}-extension.xml
air.platform.descriptor=${air.library.name}-ios-platformoptions.xml
# Default AIR Library paths:
default.library.name=AirLibraryDefault
default.library.root=../../${default.library.name}
default.library.sourcedir=${default.library.root}/src
default.extension.descriptor=${default.library.name}-extension.xml
```

• You shouldn't have to customize anything in the build script, **build.xml**.

Now the good news: you won't have to run this build script yourself. In the next step you will set up a build script for the test app that will do this for you, so you can have the ANE rebuilt and repackaged automatically when you re-run your test app in the debugger.

**Tip:** If you would like to test this build script setup, which would be a sensible thing to do, see **Test the build script on the command line** in the <u>Easy Native Extensions book</u>.

# Set up build scripts for the test app

### Like this sample?

### Set up a builder for the test app

Setting up Flash Builder to use the build scripts to debug your app is essentially the same as the set up for running the app, described in the <u>Easy Native Extensions book</u>, **Making your life easier**, **Phase 3**, but with a couple of subtle differences.

- Select your application project in Flash Builder's Package Explorer and click Project > Properties.
- From the list on the left select **Builders** and on the right click New.
- On the dialog that appears next select Ant Builder and leave the text box empty.

	Open	Project	> Properties
\varTheta 🔿 🖸 🖉	Properties for AirMobi	ileApp	
(type filter text	Builders		⇔ • ⇔ • ▼
Resource	Configure the builders	for the project:	
Flex Applications ▶Flex Build Packaging Flex Build Path	<ul> <li>✓ → Flex</li> <li>✓ → AIR application</li> </ul>	.xml Builder	New
Flex Compiler			Import
Flex Server Flex Theme Project References			Edit
Run/ 😝 🔿 🕤 Choose co	onfiguration type		Remove
Task Valid	ol type to create:		Up
Wiki I			Down
V Program			
?		Cancel	ОК
? Cancel	ОК	Leave	e this empty

### Like this sample?

You will be asked for your Certificate Password and will again be presented with the **Waiting for Debugger Connection** dialog, but this time you don't have to follow the steps listed on it: the app should already be installed on your device and all you have to do is **tap to run it**.



# **Debugging the native side**

Acknowledgement: The steps in this section are based on the Rajorshi Ghosh's article <u>Debugging Native</u> <u>Extensions for AIR iOS</u>, published on his blog <u>Beautiful Code</u>.

I have tried to simplify it and remove the 'magic' component from the process. To those of you who like a sprinkle of magic in your debugging day I apologize. The rest, I assume that eliminating mysteries is why you picked up this book in the first place.

### The process explained

The Flex Mobile Project you set up for debugging the ActionScript side of your extension has pretty much everything you need for debugging the native code: a test app and a file with debug symbols (.dSYM).

What you will do in this step is help Xcode use these. For this purpose you will add an empty mobile application target to your ANE's Xcode project and set it up to use the IPA and the .dSYM produced by the Flash Builder tools (ADT).

Like this sample?

### Set up an application target in Xcode

• With your native extension project selected in **Xcode** click **File** > **New** > **Target** on the main menu:

Ś.	Xcode	File	Edit	View	Find	Navigate	Editor	Product	Debug	Source Co	ontrol V	/indo
	• •	Ne	w					•	Tab		жт	at
	Q A (	Ad	d Files	to "Nat	iveExte	nsionTempl	ateiOS"	\	Win	dow	仓策T	
▼ 🖻 Nat 5 ta	tiveExten. Irgets, iOS	Op Op	en en Rec	ent				¥0	File Play	 /ground	ለዤ ለዤኄፓ	
	NativeExte Supporting	Ор	en Qui	ckly				☆₩O	- Targ	jet	ት¥N	
	VativeExte	Clo	se Wir	ndow				жv	/ Wor	kspace	^%N	-
	AirMobileA	Clo	se Tab	)					Gro	un	77 9£ N	
	AirMobileA AirMobileA	Clo	se "Na se Pro	ativeExt ject	ension	emplateiOS	S.xcodepro	の第~ " [o て第9	Gro	up from Sel	ection	-
🔰 🕨 📄 F	Products								rchitectu	res		

- On the target template dialog select  ${\bf iOS} > {\bf Application} > {\bf Empty} \ {\bf Application}:$ 

Choose a template for your new target:							
iOS		$\bigcirc$					
Application		• • •	1	+			
Framework & Library							
Application Extension	Master-Detail	Page-Based	Single View	Tabbed			
Other	Application	Application	Application	Application			
OS X							
Application	2						
Framework & Library	Game						
Application Extension							
System Plug-in							
Other	Single View Applie	cation					
	This template provid a view controller to n	es a starting point for nanage the view, and a	an application that uses a storyboard or nib file ti	a single view. It provides hat contains the view.			
Cancel			Pr	evious Next			

• Set **Product Name** to the name of your Flex Mobile app. This should be the same name that's specified in the **<filename>** tag in your Flex project's app descriptor file: <filename>AirMobileApp</filename>

Choose options for your new target:		
Product Name	e: AirMobileApp	
Organization Name	e: DiaDraw	
Organization Identifie	r: com.diadraw	
Bundle Identifie	r: com.diadraw.AirMobileApp	
Language	e: Objective-C	<b>\$</b>
Devices	s: Universal	0
	Use Core Data	
Projec	t: 🕒 NativeExtensionTemplateiOS	٥
Cancel		Previous Finish

- Click Finish to create the target. There should now be two targets in your Xcode project.
- Select the target you just created, go to the **Build Settings** tab and make sure that **Product Name** is set to your Flex mobile app's name:

문 I < > I b NativeExtensionTen	plateiOS					< 🛆 >
General	Capabilities	Info	Build Settings	Build Phases	Build Rules	
PROJECT	Basic All	Combined	Levels +	Q~ pr	roduct name	8
TARGETS	▼ Packaging Se	tting			AirMobileApp	
NativeExtensionTemplateiOSTests	Pr	oduct Name			AirMobileApp	
AirMobileAppTests	Set Proc	cluet Nam	e to the no	me of you	r Flex mobile	aap

• Next, go to Build Phases and delete all phases except Target Dependencies:



 $\int$  Then add a **Run Script** build phase:

B												
	General Ca	apabilities	Info	Build Settings	Build Phases	Build Rules						
5	PROJECT NativeExtensionTempl TARGETS	+ New C New R	opy Files un Script	Phase Phase		λ Search						
	WativeExtensionTempl	New He New C	eaders Ph opy Bund	ase le Resources Phase	e							
	AirMobileApp	New Co New Li New Bo	ompile So nk Binary uild Carbo	ources Phase With Libraries Phase on Resources Phase	se e							

Like this sample?

# Make the Xcode app target run your Flex Mobile app

Like this sample?

### Tell Xcode to run your app

• From the main menu in Xcode select **Product** > **Scheme** > **Manage Schemes** and from the list select the scheme for your **native extension library target** (not for the app target), then click **Edit**:

Autocre	ate schemes	Autocreate Sc	Autocreate Schemes Now	
Show	Scheme	Container	Shared	
	NativeExtensionTemplateiOS	🚵 NativeExtensionTemplateiOS project 🗘		
$\checkmark$	AirMobileApp	📩 NativeExtensionTemplateiOS project 🗘		
+ - 8	<u>3</u>			
Edit			Close	

• Select the **Run** scheme on the left and on the right make sure that the app target is selected as the **Executable** on the **Info** tab:

3 targets	Info Argun	nents Options	Diagnostics
Run Debug	Build Configuration	Debug	\$
Jebug	Executable	AirMobileApp.ap	p 🗘
T Profile Release Analyze Debug	Debug Process As	<ul> <li>Debug executable</li> <li>Me (radoslava)</li> <li>root</li> </ul>	utable va)
P Archive Release	Launch	<ul> <li>Automatically</li> <li>Wait for executable</li> </ul>	le to be launched
Duplicate Scheme	Manage Schem	es Share	d Clos

www.diadraw.com

### Start debugging in Xcode

Finally!

- Make sure your iOS device is connected to your machine via USB.
- In Xcode's main window select your device in the **Scheme** drop-down on the top left and then hit the **Run** button:

A NativeExtensionTemplateiOS 👌 📕 Radoslava's iPhone 1. Select your iOS device 2. Hit the Run button to start debugging

• Did you put any breakpoints in your code?



Like this sample?

### Bonus chapter:

# **Diagnosing build problems**

All of this single-click build, package and run business is really good at providing convenience and saving you time. One downside of it however is that when things go wrong, you are slapped with a huge build log. In it, it seems, each of the tools in the chain couldn't wait to tell you what it's doing, done or got stuck on.

In this chapter we will go through some ways in which Xcode can be of help with narrowing down compiler, linker and packaging problems and diagnosing them quickly.

First, you will need to have set up your ANE and test app for building and debugging through Xcode, <u>as</u> <u>shown in the previous chapter</u>.

### Misleading symptoms

When Xcode finishes running your script, you will usually see a pop-up or a dialog of some sort that tells you if the build succeeded or failed. These can often be misleading in several ways, just to make your life more interesting.

For example, you might see "Build Succeeded" even if there were build problems, but Xcode managed to get hold of an old .ipa or .app to run:



Or you might see this error message, telling you "The file X couldn't be opened because you don't have permission to view it". 99% of the time this has nothing to do with permissions and could be an indication of an error in your ActionScript code, password mismatch in your provisioning, packaging issues or bad lunch.



Where to look instead

# Issues on the native side

Compiler problems Provisioning problems

Like this sample?

### **Errors on the AIR side**

Diagnosing problems that happened on the AIR side requires more digging in Xcode, but is rewarding in terms of finding out what the issues are without having to switch back and forth between IDEs.

Anything that happens on the AIR side up to installation will be reported in Xcode's build log. Issues that happen during installation however you will find in your device's console. Let us take a look at these in turn.

### Xcode's build log

You saw a screenshot of the build log above, but here it is again in its entire beauty:



Like this sample?

### Xcode's device console

Sometimes you only get to hear about a problem with your app package when you try to install it on a device. And you are often notified by a terse error message like this:

#### Application verification failed.

It is another one of those that can mean a range of things. This is where Xcode's device console comes in handy. To open it, plug your iOS device into your computer via USB, then go to **Window > Devices** and if you don't see a console at the bottom of the screen, click the little arrow button to expand it.

Name       Radcolava's (Phone Model         Redcolava's Phone SIMULATORS         Pad 2		DENGO	-	Device Information				
10:10.3 (14D136)         ■ Redoslavs's Phone         SIMULATORS         Pad 2         0:10.3 (14D136)         ■ Pad cslavs's Phone         SIMULATORS         Pad 2         0:10.3 (14D136)         Battery         100%		DEVICE	.o My Mac	Name	Radoslava's i	Phone		
Image: Structure is Processed in the second sec			10.10.3 (14D136)	Model	iPhone 45			
Battery       100%         IPad 2       5.1 (128411)         IPad Air       S.1 (128411)         IPad Air       S.1 (128411)         IPad Air       S.1 (128411)         IPad Air       Common State S			Radoslava's iPhone	Capacity	12.69 GB (52	9.5 MB available)		
SIMULATORS       I/32 4/11         IPad 2       I/128411         IPad Air       2114246503001122015d6b85763c9c266         IPad Air       21142411         IPad Rotha       0			0.1.3 (128400)	Battery	100%			
iPad 2         8.1 (12E411)         iPad Air         8.1 (12E411)         iPad Relina         8.1 (12E411)         iPad Relina         8.1 (12E411)         iPad Relina         8.1 (12E411)         iPhone 4s         8.1 (12E411)         iPhone 5         8.1 (12E411)         iPhone 5         8.1 (12E411)         iPhone 6         8.1 (12E411)         iPhone 7         8.1 (12E411)         iPhone 8         8.1 (12E411)         iPhone 8		SIMUL	ATORS	108	8.1.3 (128466	5)		
Pad Air B.1 (128411)         Pad Reitha B.1 (128411)         Phone 4s B.1 (128411)         B.1 (128411)         Phone 5s B.1 (128411)         Phone 6s B.1 (128411)         Phone			iPad 2 8.1 (12B411)	Identifier	2a144b5c300	01f22016d6b85763c9e266		
iPhone 4s 81 (128411)       installed Apps       on runtime problems like crashes.         iPhone 5 81 (128411)       iPhone 58 81 (128411)         iPhone 68 81 (128411)       iPhone 68 81 (128411)         iPhone 6 81 (128411)       iPhone 6 81 (128411)         iPhone 7 81 (128411)       iPhone 8 81 (128411)         iPhone 7 81 (128411)       iPhone 8 81 (128411)         iPhone 8 81 (128411)       iPhone 8 81 (128411)         iPhone 9 81 (128411)       iPhone 8 81 (128411)         iPhone 8 81 (128411)       iPhone 8 81 (128411)         iPhone 8 81 (128411)       iPhone 8 81 (128411)         iPhone 9 81 (128411)       iPhone 8 81 (128411)	~		iPad Air 8.1 (128411) iPad Retina 8.1 (128411)	View Device Logs Ta	we Screenshot	the Device logs for	r informatio	<i>i</i> n
IPhone 5         8.1 (12B411)         IPhone 5e         8.1 (12B411)         IPhone 6 Plus         8.1 (12B411)         IPhone 6 Plus         8.1 (12B411)         IPhone 6 Plus         8.1 (12B411)         IPhone 6         IPhone 6         8.1 (12B411)         IPhone 6         8.1 (12B411)         IPhone 8         8.1 (12B411)         IPhone 8         8.1 (12B411)         IPhone 8         8.1 (12B411)         IPhone 8         8.1 (12B411)         IPhone 9         IPhone 9         8.1 (12B411)         IPhone 9         IPhone 9         IPhone 9         8.1 (12B411)         IPhone 9         IPhone 9 <th></th> <th></th> <th>iPhone 4s 8.1 (12B411)</th> <th>Installed Apps</th> <th>on ru</th> <th>ntime problems like</th> <th>crashes.</th> <th></th>			iPhone 4s 8.1 (12B411)	Installed Apps	on ru	ntime problems like	crashes.	
<ul> <li>Phone 5s 8.1 (12B411)</li> <li>Phone 6 Plus 8.1 (12B411)</li> <li>Phone 6 8.1 (12B411)</li> <li>Resizable iPad 8.1 (12B411)</li> <li>Resizable iPhone 8.1 (12B411)<!--</td--><td></td><td></td><td>iPhone 5 8.1 (12B411)</td><td>AirMobileApp</td><td>1.1.0</td><td>AirMobilsApp</td><td></td><td></td></li></ul>			iPhone 5 8.1 (12B411)	AirMobileApp	1.1.0	AirMobilsApp		
<ul> <li>B.1 (12B411)</li> <li>Phone 6 Plus B.1 (12B411)</li> <li>Phone 6 B.1 (12B411)</li> <li>Resizable iPad B.1 (12B411)</li> <li>Resizable iPhone B.1 (12B411)</li> <li>The Device console can help you wead out installation issues.</li> <li>May 13 15:27:09 Radoslavas-iPhone kernel[0] <prore: could="" not="" option<br="" set="" socket="">S0_0PPORTUNISTIC: Invalid argument     <li>May 13 15:27:10 Radoslavas-iPhone lockdownd[26] <prore: could="" not="" option<br="" set="" socket="">S0_0PPORTUNISTIC: Invalid argument     <li>Aug 13 15:27:10 Radoslavas-iPhone kernel[0] <prore: could="" not="" option<br="" set="" socket="">S0_0PPORTUNISTIC: Invalid argument     <li>Aug 13 15:27:10 Radoslavas-iPhone kernel[0] <prore: could="" not="" option<br="" set="" socket="">S0_0PPORTUNISTIC: Invalid argument     <li>Aug 13 15:27:10 Radoslavas-iPhone kernel[0] <prore: could="" not="" option<br="" set="" socket="">S0_0PPORTUNISTIC: Invalid argument     </prore:></li> <li>The Device console can help you</li> </prore:></li></prore:></li></prore:></li></prore:></li></ul>			Phone 5s	CameraTutorialApp	1.0.0	CameraTutorialApp		
<ul> <li>Phone 6 Plus 6.1 (128411)</li> <li>Phone 6 8.1 (128411)</li> <li>Resizable iPad 8.1 (128411)</li> <li>Resizable iPhone 8.1 (1</li></ul>	1		8.1 (12B411)					
<ul> <li>Phone 6 8.1 (128411)</li> <li>Rosizable (Pad 8.1 (128411)</li> <li>Rosizable iPhone 8.1 (128411)</li> <li>Rosizable iPhone 8.</li></ul>	1		iPhone 6 Plus 8.1 (128411)	+ - 13	_ The I	Device console can l	nelp you	
Fosizable IPad 8.1 (128411)         Peeizable IPhone 8.1 (128411)         Pieizable IPhone 8.1 (128411)         Piezzable IPhone 8.1 (1284111)         Piezzable IPhone 8.1 (1284111) </td <td></td> <td></td> <td>iPhone 6 8.1 (128411)</td> <td></td> <td>weed</td> <td>out installation iss</td> <td>Ves.</td> <td></td>			iPhone 6 8.1 (128411)		weed	out installation iss	Ves.	
May 13 16:27:09 Radoslavas-iPhone lockdownd[26] <error>: Could not set socket option         S0_0PPORTUNISTIC: Invalid argument         May 13 16:27:10 Radoslavas-iPhone kernel[0] <notice>: flow_divert_token_set (0): Failed to get the key unit from the token: 22         May 13 16:27:10 Radoslavas-iPhone kernel[0] <notice>: flow_divert_token_set (0): Failed to get the key unit from the token: 22         May 13 16:27:10 Radoslavas-iPhone kernel[0] <notice>: flow_divert_token_set (0): Failed to get the key unit from the token: 22         May 13 16:27:10 Radoslavas-iPhone kernel[0] <notice>: flow_divert_token_set (0): Failed to get the key unit from the token: 22         May 13 16:27:10 Radoslavas-iPhone kernel[0] <notice>: flow_divert_token_set (0): Failed to get the key unit from the token: 22         May 13 16:27:10 Radoslavas-iPhone kernel[0] <notice>: flow_divert_token_set (0): Failed to get the key unit from the token: 22         May 13 16:27:10 Radoslavas-iPhone lockdownd[26] <error>: Could not set socket option         S0_0PPORTUNISTIC: Invalid argument         + 100         *</error></notice></notice></notice></notice></notice></notice></error>			Resizable IPad 8.1 (12B411)	May 13 15:27:09 Radoslava	s-iPhone kernel en: 22	<pre>(0) <notice>: flow_divert_token_set</notice></pre>	(0): Failed to get	
<ul> <li>Any 13 16:27:10 Radoslavas-iPhone kernel[0] <notice>: flow_divert_token_set (0): Failed to get the key unit from the token: 22 May 13 16:27:10 Radoslavas-iPhone lockdownd[26] <error>: Could not set socket option SO_OPPORTUNISTIC: Invalid argument May 13 16:27:10 Radoslavas-iPhone kernel[0] <notice>: flow_divert_token_set (0): Failed to get the key unit from the token: 22 May 13 16:27:10 Radoslavas-iPhone lockdownd[26] <error>: Could not set socket option SO_OPPORTUNISTIC: Invalid argument</error></notice></error></notice></li> <li>+ 183 </li> </ul>			Resizable iPhone	May 13 16:27:09 Radoslava 50 OPPORTUNISTIC: Invalid	s-iPhone lockdo argument	wnd[26] <error>: Could not set socke</error>	t option	
+ 🕸 💿			8.1 (128411)	May 13 16:27:10 Radoslava the key unit from the tak May 13 16:27:10 Radoslava SO_OPPORTUNISTIC: Invalid May 13 16:27:10 Radoslava the key unit from the tak May 13 16:27:10 Radoslava SO_OPPORTUNISTIC: Invalid	s-iPhone kernel en: 22 s-iPhone lockdo argument s-iPhone kernel en: 22 s-iPhone lockdo argument	<pre>(0) -Notice&gt;: flow_divert_token_set wnd(26) <error>: Could not set socke (0) -Notice&gt;: flow_divert_token_set wnd(26) <error>: Could not set socke</error></error></pre>	<ul><li>(0): Failed to get</li><li>(0): Failed to get</li><li>(c): Failed to get</li><li>(c): option</li></ul>	
		+ 🞲	Θ				tît	0

Like this sample?

### If all else fails...

Like this sample?

# Acknowledgements and where to go from here

As much as I would have loved to claim that I worked out how to debug an ANE in Xcode all by myself, I can't. I must give credit to **Rajorshi Gosh** and his spot-on blog post <u>Debugging Native Extensions for AIR</u> <u>iOS</u>.

### Useful online materials

- Free online tutorials on EasyNativeExtensions.com
- Debug an application on an Apple iOS device from the Flash Builder 4.6 Adobe Online Manual
- Debug an application on an apple iOS device from the Flash Builder 4.7 Adobe Online Manual
- <u>Debug and Tune Your App</u> from the **Xcode** User Guide
- <u>Understanding and Analyzing iOS Application Crash Reports, Technical Note TN2141 by Apple</u> for the times you get caught by surprise

### Want more?



For a library of conversion functions that will save you time passing data between Objective-C and ActionScript download our <u>iOS</u>
 <u>Data Types Guide: keeping your cool when jumping between languages</u>.

### Thank you!

Thank you for picking this Debugging Guide and investing the time and money in it! We hope it saved you hours and got you closer to the top-notch quality we know you always strive for.

If you have a few minutes, we would love to hear what you thoughts about it.

Leave a comment on our blog or drop us an e-mail at office@diadraw.com.

### Other ways to get in touch and share

@DiaDrawCom on Twitter

https://www.facebook.com/DiaDraw on Facebook

DiaDraw on LinkedIn

## Thanks again and may the source be with you!