it would be for Joe the Plumber to hook up the hot water to the hot water and the cold water to the cold water.

I was on a roll, all done in 30 minutes. A quick look at the clock: 11:30pm. I'll quickly package it all up, submit my update, and pick out a dark comedy on Netflix instant watch…

Little did I know that for the next two hours I will be beaten, cubed and ground into hamburger meat by Apple engineering's probably most ingenious attempt to one up Turing's theory of the "halting problem": **code signing**.

This was supposed to be easy: the certificates were correct and installed. I've successfully signed and posted this app before. I haven't changed any build settings. The certificates are still valid. I read the manual. I followed each step. I can make a wedding cake from scratch. I can even make sausage from scratch.

wedding cake and sausage made by me

Apparently Apple knows that code signing has issues. There are some suggestions on the program portal to fix code signing (as in delete all information, restart and retype all information- feels like reading a Windows 3.0 manual.) So why don't they fix it? Good question. Maybe they read this post, get angry and fix it…

I mean how hard can this be? Look up two field values and execute a script with those arguments? Geezus. In fact I have another question:

Why do developers have to bother with code signing at all? Why are the apps not automatically signed when uploaded to the store?

Now that I've got it all out let's look at some solutions after we reread the first step of **code signing anonymous**:

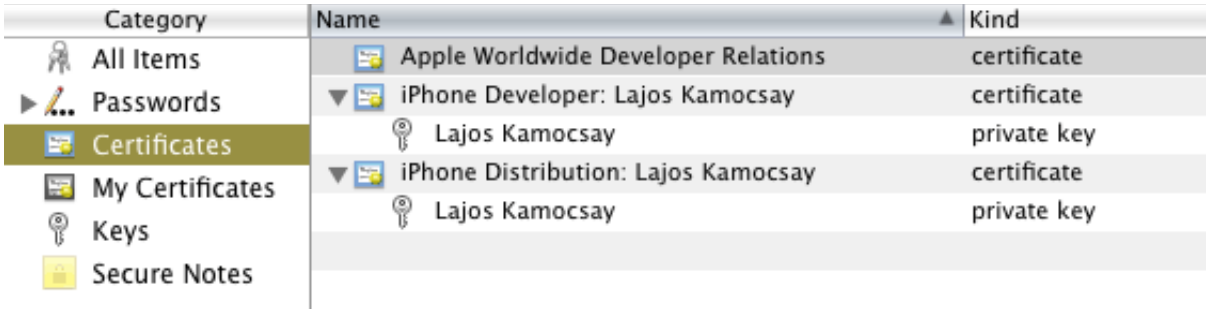
Step One: We admitted we were powerless over code signing - that our lives had become unmanageable.

</rant>

These solutions assume that you have valid certificates and the certificates are installed. Before we get too far into it, let's do some:

Sanity checks

1. Make sure your keychain has the Apple WDR certificate, your distribution certificate and under it the private key used to create the certificate. This is best checked by setting the **Category** to **Certificates**. Click the triangle next to the certificate to see your private key. You should see something like this:

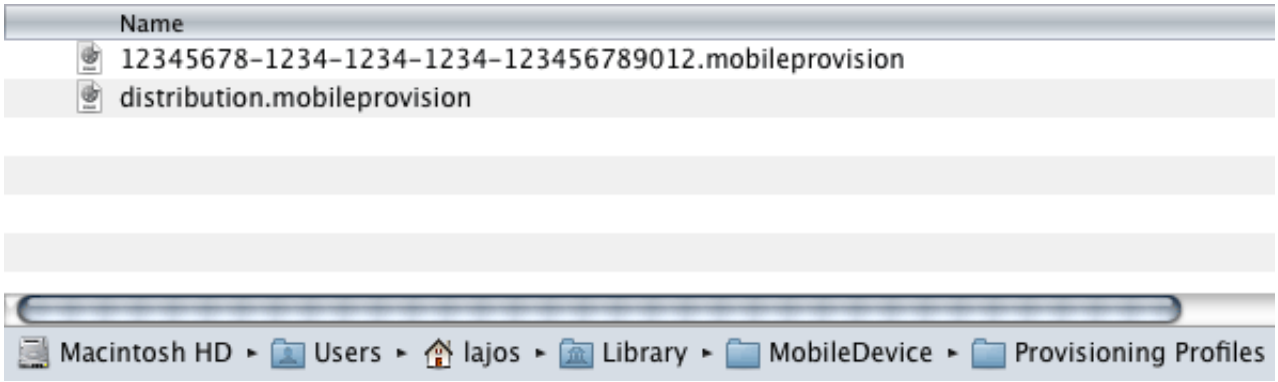


keychain certificate view

* If you don't have a private key (no triangle) because you got a new computer, reinstalled osx, or are using your boyfriend's computer and you have not exported your private keys (hint, hint, hint -> export your private keys now), you'll need to throw away all your certificates and provisioning profiles and create new ones on the program portal.

* I figured I could blur my name on the images, but then you could just scroll up a bit and see it anyway. So I decided to not go through the hassle.

2. Make sure that the provisioning profiles show up in the **{YOUR_HOME_DIRECTORY}/Library/Mobile Provisions** folder. The profiles might have the original name as you downloaded them from the program portal or they might have been renamed to their ids if you installed them by dropping on xcode. They will work either way.



provisioning profiles folder

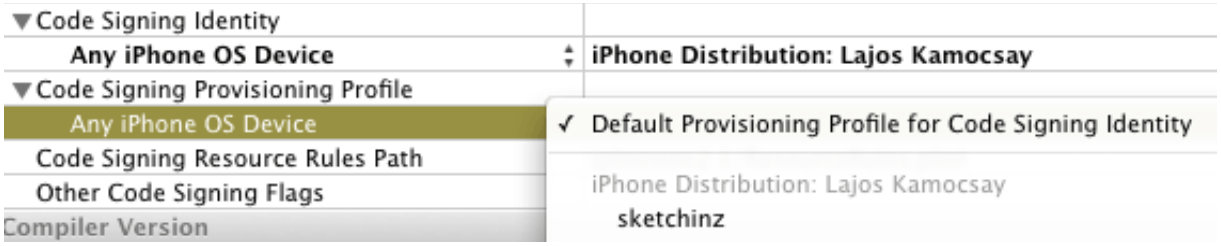
3. This is just a suggestion: I create wildcard app ids for all my apps so I don't have to worry about the **Identifier** setting in the Target Info / Properties tab. You can leave the default string: *"com.yourcompany.${PRODUCT_NAME:identifier}"*, or you can change it to whatever you want. To create a wildcard app id in the program portal, just put an asterix in the App ID field as below:

wildcard app id

Ok, let's look at the first evil.

"distribution profile not showing up"

The **Distribution** configuration is created by duplicating the **Release** configuration. At the target level enter **iPhone Distribution: Your Name** in the **Code Signing Entity** field. When you click on the **Code Signing Provisioning Profile** field value, you should see something like this (sketchinz profile showing):



distribution profile showing

Sometimes, however, you are presented with no provisioning profiles like this:

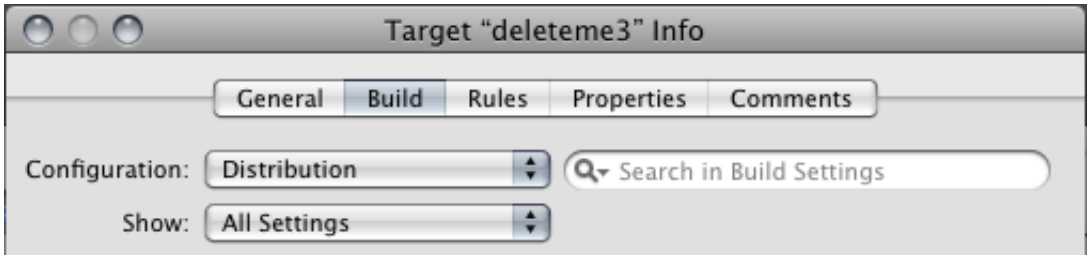
distribution profile not showing

Before we go much further, let's do some easy checks:

1. Make sure you have the **Distribution** configuration set active.

distribution configuration active

2. Make sure you are modifying the **Distribution** configuration.



modifying distribution config

2. Make sure there's no typo. The code signing identity field should match the name of your certificate in your keychain. It's case sensitive.

3. Make sure that there is one space between "Distribution:" and "Your Name".

4. There should be no space before “iPhone” and no space after “Your Name”.

Excercise #1

Leave your project open in xcode and create a new project. Create a **Distribution** configuration, enter **iPhone Distribution: Your Name** in the **Code Signing Identity** field and see if your profiles show up here. If they do, it’s your lucky day because most likely they will show up in your original project now.

I think this was originally suggested at v2ex.com in this post.

Excercise #2

1. Do a “Clean All” in your project.

2. Delete the value in the **Code Signing Identity** field.

3. Exit xcode.

4. Go to the **build** folder for your project and delete everything. (Especially the folder called {YOUR_PROJECT_NAME}.build.) The build folder is usually located right in your project’s folder, if it’s not there you can find the loction in the project’s info window under the **General** tab.

5. Restart xcode, fill in **Code Signing Identity** and see if your profile shows up. If it does, it’s your lucky day.

6. If it doesn’t you can try this again, but restart your computer before restarting xcode. If your profile shows up now, it’s your lucky day.

This method is suggested on the program portal by Apple.

Exercise #3

It’s possible to set a value for the **Code Signing Identity** and **Provisioning Profile** at both the **Project** and **Target** level. And it’s possible to set different values in two places!

In theory this should not be a problem. If a value is not set at the **Target** level, it falls back to the value at the **Project** level. If a value is set at both places, the **Target** level value should have priority.

It seems however that xcode sometimes gets, hmm, confused. In fact if you followed the directions from the program portal, you will have **iPhone Developer** set at the **Project** level and **iPhone Distribution: Your Name** at the **Target** level for the identity.

Set the identity to **iPhone Distribution: Your Name** at the **Project** level. If your profile shows up, it’s your lucky day.

If not, do a **Clean All**, exit xcode, delete the contents of the **build** folder and restart xcode. If your profile shows up, it’s your lucky day.

Most likely by now your distribution profile shows up. If not, the move on to the next exercise that should also help with:

“no packaging or code signing during build”

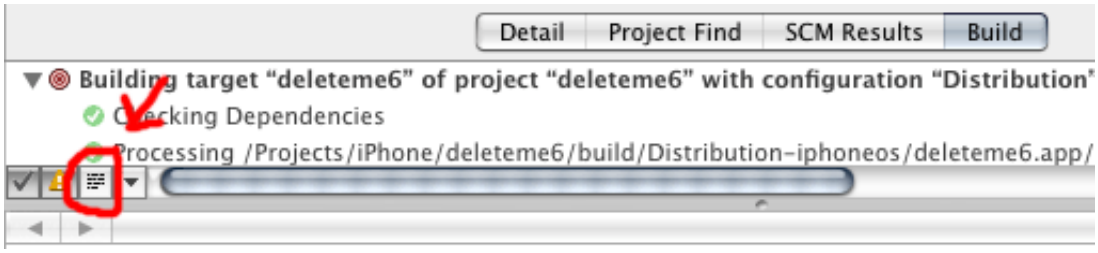
“code signing with the wrong certificate”

Exercise #4

The program portal instructs to verify code signing by looking at these two lines in the build out

> *ProcessingProductPackaging "**/Users/lajos/Library/MobileDevice/Provisioning Profiles/sketchinz.mobileprovision**" /Projects/iPhone/deleteme6/build/Distribution-iphoneos/deleteme6.app/embedded.mobileprovision …*
>
> *CodeSign /Projects/iPhone/deleteme6/build/Distribution-iphoneos/deleteme6.app*
> *cd /Projects/iPhone/deleteme6*
> */usr/bin/codesign -f -s "**iPhone Distribution: Lajos Kamocsay**"…*

By the way to look you can look at the build output in the **build** tab (shortcut: command+shift+B) by clicking this button:



xcode build output

Your app is correctly signed when:

1. product packaging uses the correct provisioning profile

2. product packaging outputs the embedded.mobileprovision into the correct .app build folder

3. code signing uses the correct identity

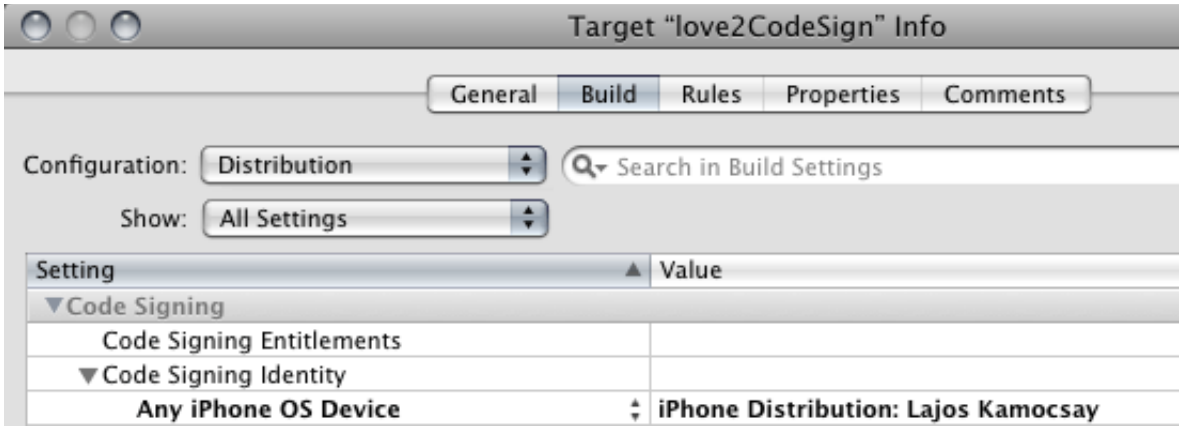
On a side note, I suggest installing the provisioning profiles by manually copying them into the **{YOUR_HOME_DIRECTORY}/Library/Mobile Provisions** folder rather than dropping them on xcode because when dropped on xcode the files are renamed to their ids. If you copy them by hand the file name will remain consistent with the build settings and the build output. (You can open up the certificates and look at the ids by the way, but why bother with remembering hex values at 3 in the morning.)

So how can xcode use the wrong certificates?

As discussed in **Exercise #3**, the **Code Signing Identity** and **Provisioning Profile** can be set at both the **Project** and **Target** levels. If the values are different, xcode seems to get conused sometimes. In my experience fixing the discrepancy had sofar eliminated the code signing issues from above. (There is one more issue coming later, the most evil of them all that needs even worse measures.)

In this exercise I will suggest something contrary to the instructions on the program portal. Instead of setting the **Code Signing Identity** and **Provisioning Profile** at the **Target** level, we'll only set them at the **Project** level. The **Target(s)** will inherit the settings the **Project** and life will be as sweet and naughty as chocolate cake.

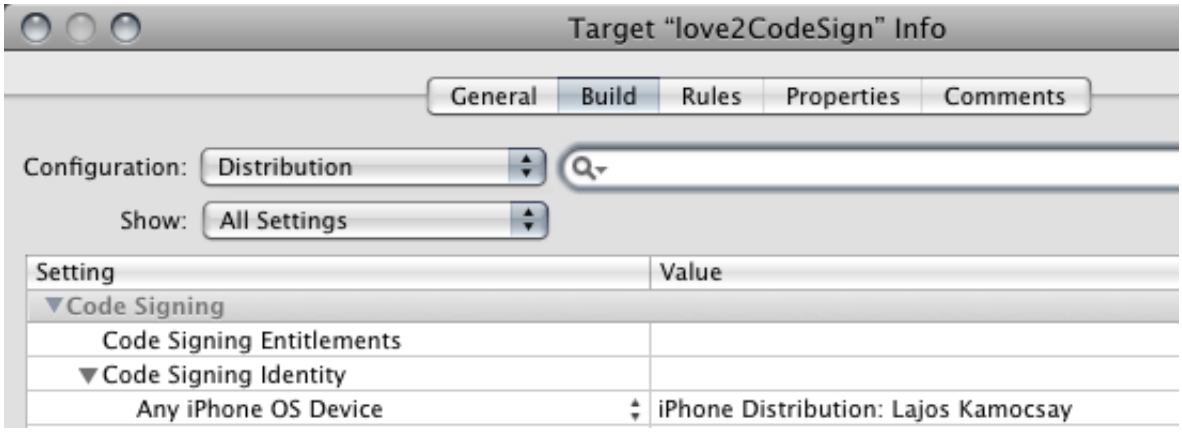
1. Make sure that there are no values set in the **Target Build info**. Xcode displays values set here in **bold**:



xcode target set

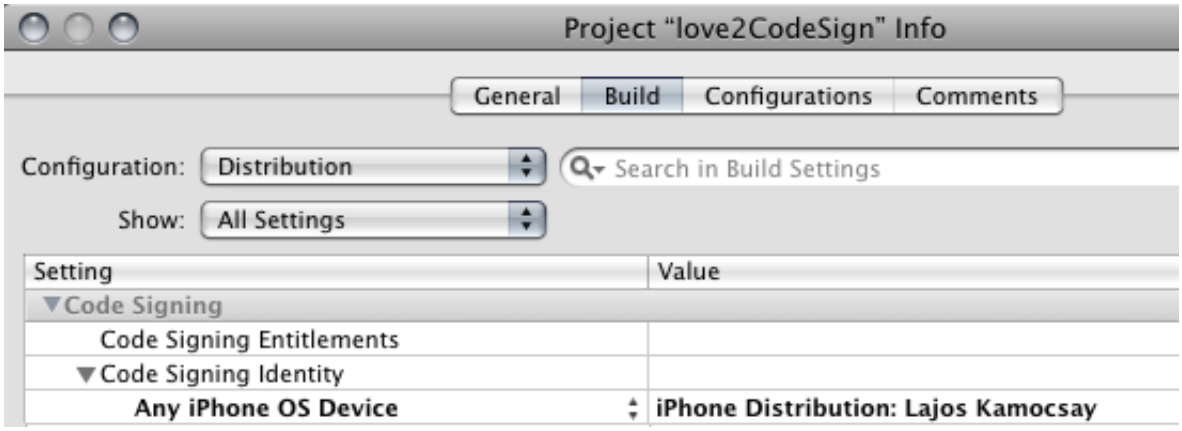
(You can also set the **Show** option to **Settings Defined at this Level**, and you want to make sure that **Code Signing Identity** and **Provisioning Profiles** don't show up there.)

2. To reset them, select the line where the value shows up and hit the 'delete' key. Both values should now appear in normal font weight:



xcode target unset value

3. Go to the **Project Build info**, and set the **Code Signing Identity** and **Provisioning Profile**.



xcode project distribution profile set

4. You might need to do the magical xcode voodoo dance of "Clean All", exit, delete everything in the build folder and restart xcode, but by now that should be second nature. Kind of like washing hands before one touches its own self.

The only situation this approach would not work conveniently is if there were multiple targets all with

their own provisioning profile. I have no idea why anyone would impose such hell on hirself.

In fact, I have no idea why the hell would anyone use a non-wildcard app ids. I guess some people like self torture while some others are following stupid arse executive orders. If you are using non-wildcard app ids and don't fall into either of those categories, this is your chance to redo your profiles with wildcard app ids (discussed somewhere above.)

And now let's look at the worst of them all:

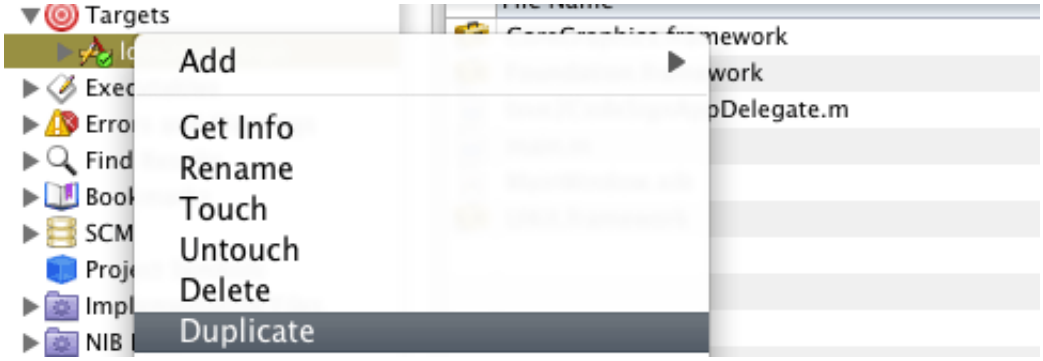
"product packaging uses wrong .app build folder"

Exercise #5

This one is pretty easy to recognize. The .app build folder on the product packagin line is pointing to a different configuration (eg. **Release**) or a completely different .app.

The only solution I've found for this one is to:

1. duplicate the target

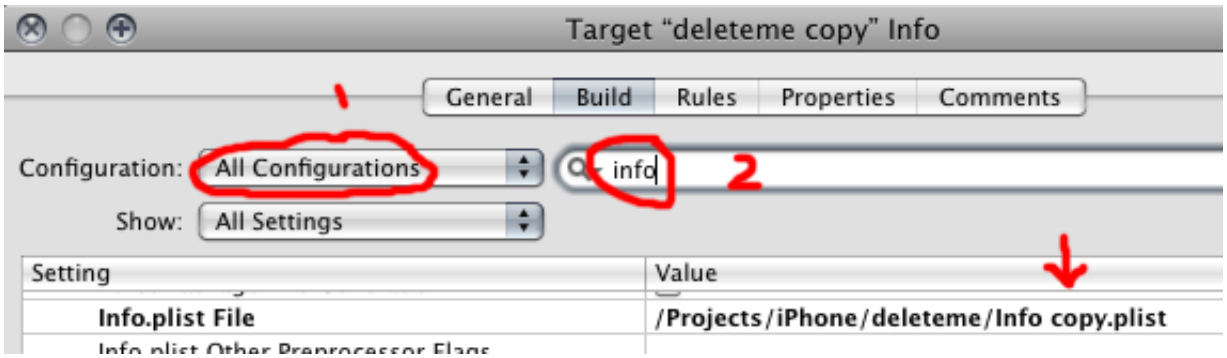


xcode duplicate target

2. delete the old target

3. rename the new target (take the " copy" off the end)

4. when duplicated, xcode creates a copy of the **Info.plist** file. To fix, go to the **Target Info**, select all configs, and search for info -> change **Info copy.plist** back to **Info.plist**. (You should also delete the **Info copy.plist** file from **Resources**.)



xcode target info copy.plist

Now try to "Clean All", delete old builds and restart xcode. Cross the fingers and build.

“no ProcessingProductPackaging line in build output”

This solution comes from Jonathan Mulcahy. Here’s his discription of the problem:

```
The app builds fine and runs on my iPhone fine. When I try to build
the Distribution app, it builds fine with no errors, but when I look
at my build log, I do have the CodeSign entry, but there is no log
entry for ProcessingProductPackaging part, so the
embedded.mobileprovision does not exist.
```

He also noticed that in the Code Signing Identity options it said “All Code Signing Certificate Identites” instead of the name of the mobileprovision. (click on image to see full size)

missing mobileprovision name

You should see the name of the mobileprovision, like in this case sketchinz:

mobileprovision name showing

So the solution:

Exercise #6

Remove the iphone SDK and reinstall it, preferably the latest version (at this time 2.2.1).

Thanks Jonathan for this contribution!

Final thoughts

- don’t give up- it will work
- feel free to contact me if you are still stuck or have any additional info to share with the world

Filed under:
apple, iphone by lajos

74 Responses to “how to fix iphone code signing errors”

1. 
   Bungler
   January 28th, 2009 at 4:10 pm

   Dear lajos,

   I am having an issue with the build process fro my app. The problem is that after a clean build, no where in the build log are mentioned ProcessingProductPackaging or CodeSign even tho I fallowed

all the official instructions and all of you unofficial recommendations. It seems like xcode is just not attempting to sign at all… Amy suggestions would be helpful. Thanks!

2. 
lajos
January 28th, 2009 at 10:19 pm

@Bungler-> Sent you an email.

3. 
Jonathan
January 29th, 2009 at 9:20 am

Hi lajos,

I just sent you an email, I'm having a similar problem as Bungler, I have no mention of ProcessingProductPackaging at all. I do see CodeSign however.

Thanks!

4. 
Matt Mendrala
February 7th, 2009 at 12:15 pm

Thanks a bunch for the info. I racked my brain for hours trying to figure out why my app couldn't be verified. It turned out that I had different code signing settings on my project and target. Once I set them to be the same everything worked great.

5. 
Jacob
February 16th, 2009 at 2:00 pm

Hello Lajos,

All your guidance has saved me hours of research. Thank you.

I got past my initial cert config, now that I have a successful build, I get the following when I try to deploy the app to my iPhone …. any idea why? Thank you.

Your mobile device has encountered an unexpected error (0xE800003A)
ApplicationVerificationFailed

6. 
lajos
February 16th, 2009 at 2:09 pm

@Jacob: 0xE800003A means that your phones doesn't have the .mobileprovision file installed that you signed your app with.
You can install it in the Organizer.