APPENDIX B

Troubleshooting Xcode

What You'll Learn in This Appendix:

- Logging in to the developer website
- Uploading an application to your device
- Installing your mobile provision
- Building your application to run on a device
- Updating mobile provisions
- Debugging the ambiguous matches error
- Uploading an application to your device
- Checking changes that do not show up
- Checking your executable

Although Cocoa is a joy to work with, Apple's code-signing process is difficult to use. Because it is a black box, problems are difficult to debug. Moreover its encryption will not prevent your applications from easily being cracked and distributed. Hopefully, this appendix will help you work around issues you encounter in what is commonly called "provisioning hell."

Logging in to the Developer Website

After signing up for the software development kit (SDK), you might still have trouble logging in to the developer website. It is possible that your Internet service provider (ISP) has done something that Apple's developer site rejects. This is likely the problem if, after entering a valid Apple ID and password, you are redirected to the login page but not told that the ID and password are invalid as shown in Figure B.1. This problem can be fixed by using an SSH tunnel (Secure Shell) to a computer running on a different ISP.

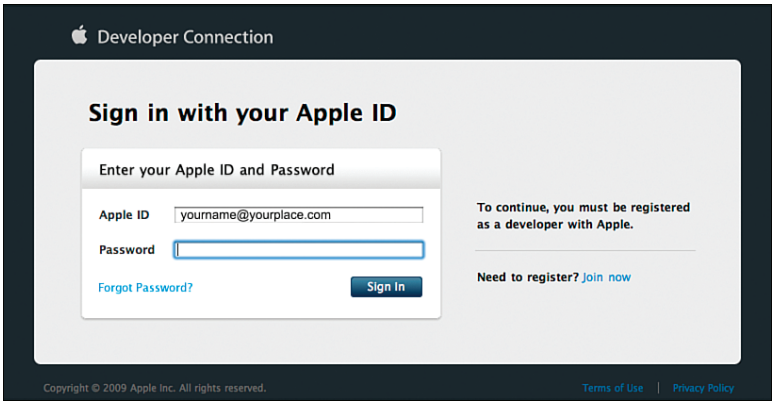

FIGURE B.1
The developer website login

To use an SSH tunnel to log in from a different ISP, follow these steps:

1. First, start the Terminal (in `/Applications/Utilities`) and run `ssh` with the `-D 9999` option:

   ```
   ssh -l user computer -D 9999
   ```

   (where *user* is your username and *computer* is the computer you're tunneling through). Log in and stay connected.

Did you Know?

If you receive the message `bind: Address already in use` when logging in, exit and try a different port, for instance 9998.

2. Enable the SOCKS Proxy in the System Preferences Application (choose Network, Ethernet, Advanced, Proxies, and enable the SOCKS Proxy).
3. Set the SOCKS Proxy Server to localhost and the port to 9999 (see Figure B.2), click OK, and then click Apply.
4. Log in (try a few times before giving up). Only log out of your SSH connection and disable to the SOCKS Proxy when you have finished using Apple's developer website.

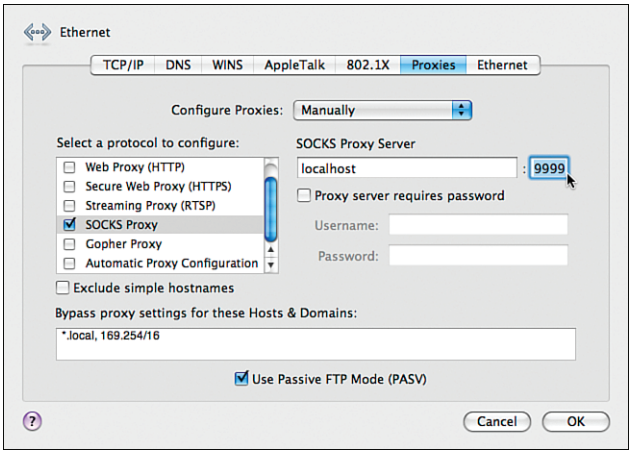

FIGURE B.2
Using a SOCKS Proxy Server

> The large size of the SDKs can be a problem if you have a slow, 100Kb/s connection.

Watch Out!

Obtaining Permission to Upload Your Application to Your Device

When you upload an application to your device, you must follow Apple's instructions very closely. Missing a single step can prevent your application from working.

Uploading code to your device involves two steps:

1. **Create a development certificate**—This permits your Mac to create an application to be uploaded to an iPhone.
2. **Create a provisioning profile**—This permits an iPhone or iPod Touch to receive the application you created.

Creating a Certificate Request

You should consider two factors while creating a certificate signing request:

- Messages created by the public key can only be decrypted by the private key.
- The originator of messages created by a private key can be validated by recipients who have the public key.

When you make a certificate signing request, you are sending your public key to Apple. Apple will then return a distribution certificate signed with its private key that contains your public key. Your computer can check that Apple issued the certificate because it knows Apple's public key. You must double-click the `.cer` file to install the development certificate. Your Mac can now build signed iPhone applications. The same process is used to create a distribution certificate. Because Apple does not have your private key, it cannot keep it for you, but without it, you can't build your application. Therefore, it is vital to keep a backup copy of each key.

To make a copy of your private key, follow these steps:

1. Launch the `Keychain Access` application in `/Application/Utilities`.
2. Choose the login keychain, and then the Keys category.
3. The keys associated with your certificates will have a disclosure triangle. Open the disclosure triangle to reveal the corresponding certificate.
4. Right-click (Ctrl-click) the key and choose the Export "KeyName" menu item.
5. Save each key with a `.p12` extension as shown in Figure B.3. You'll have to provide a password to encrypt them. You can install them by double-clicking the `.p12` files on another computer. Be careful not to install them twice because Xcode only signs your application correctly if you have installed a single copy of them.

FIGURE B.3
Saving a private key

Creating a Provisioning Profile

To create a provisioning profile, simply create the profile and specify which devices may install your application. To uniquely identify an iPhone or iPod Touch, Apple assigns each device it manufactures a unique device identifier (UDID). You need this identifier to create your provisioning profile.

To obtain your UDID, follow these steps:

1. Open Xcode's organizer and look for Identifier. (Note: Your iPhone must be attached to your computer to see this.) Alternatively, click Serial Number in iTunes' Device Summary page.
2. Download and install the WWDR Intermediate Certificate.

To see the entire process explained in full detail, go to: http://adcdownload.apple.com/iphone/iphone_developer_program_user_guide/iphone_developer_program_user_guide__standard_program_v2.4.pdf.

Installing Your Mobile Provision

After obtaining a mobile provision, you can install it by dragging it on top of the Xcode icon in the Dock. This will copy it to the Library/MobileDevice/Provisioning Profiles directory in your home directory. Each provision you install looks like `06B7817A-3CF3-49FD-A668-8BB547EC00DC.mobileprovision` and contains encrypted data.

Building Your Application to Run on a Device

Follow these steps to build your application to run on a device:

1. Set the Bundle identifier in your `Info.plist` file to the AppID you created at the Developer Program Portal (see Figure B.4).

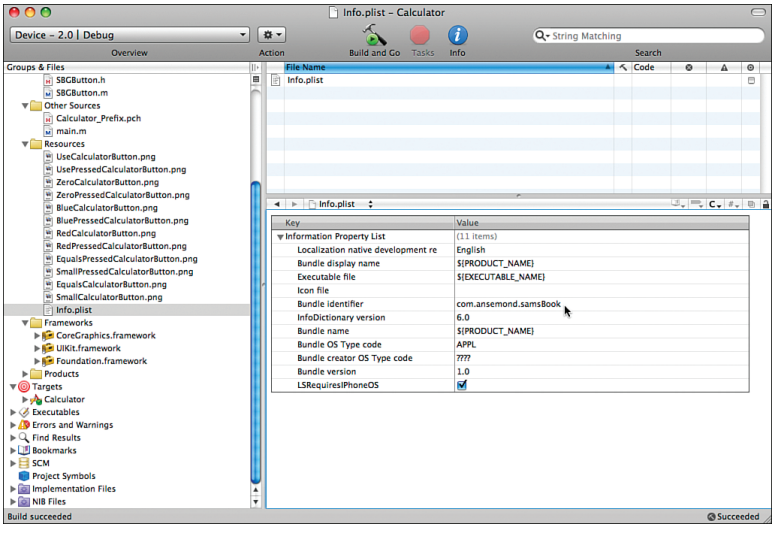

FIGURE B.4 Setting the Bundle identifier

2. Right-click (Ctrl-click) on the topmost item in the Groups & Files pane. This will reveal an Info window. Choose the Build settings, and choose the Active configuration (`Device – 2.0 | Debug` in Figure B.5).

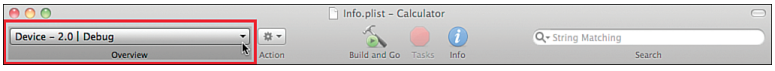

FIGURE B.5
The Active configuration

3. Open the Code Signing Identity section and replace Don't Code Sign with the provision you want to use. Always change the Active configuration or Xcode will not apply your settings correctly although it will show them as having been set. (see Figure B.6).

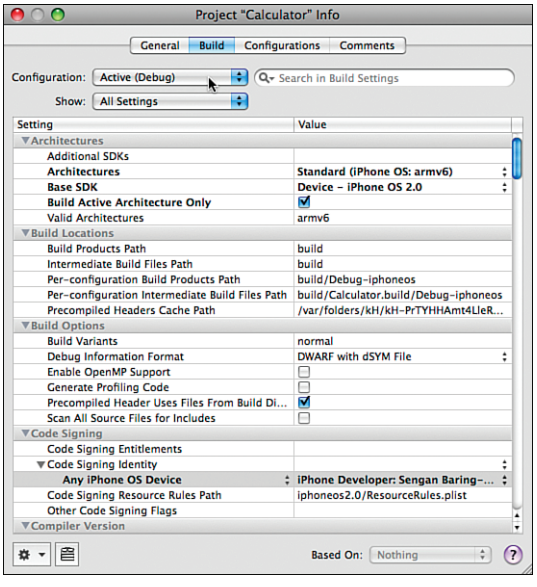

FIGURE B.6
Only change the Active configuration

Watch Out!

Make sure that the `Base SDK` is set to a valid SDK (one you have installed on your computer and that is compatible with the version of the OS on your iPhone).

Watch Out!

There is an identical window that is opened by choosing the Targets in the Groups & Files pane. These are the Target settings and they override the Project settings. If you set provisions there, and then try to change the project-level provisions, your project provision changes won't be used.

Updating Mobile Provisions

Problems can occur when switching between mobile provisions, be it updating provisions or switching to a different provision to build the App Store version of the application. Xcode does not always recognize a provision change. It will show the different settings, but still use the old provision. One option is to exit Xcode and start it up again each time you change a setting. Sometimes switching the provision to Don't Code Sign first helps; also quitting, restarting, and switching to the desired setting works.

Another possible solution is listed in the following steps:

1. Quit Xcode.
2. Move all the provisions from the Library/MobileDevice/Provisioning Profiles directory in your home directory to a different directory.
3. Open and then close Xcode to clear its profile cache.
4. Put all the provisions back into the Library/MobileDevice/Provisioning Profiles directory.
5. Open Xcode.
6. Open the target's Build settings and reset the provisioning profile settings for all configurations.

A third option is to modify the `projects.pbxproj` file in your project's *ProjectName*.xcode directory. (Be sure to save a copy first in case this does not work.) To do this, follow these steps:

1. Go to the Library/MobileDevice/Provisioning Profiles directory in your home directory and find the newly installed profile.
2. Copy the xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx number from the profile's filename (for example, 06B7817A-3CF3-49FD-A668-8BB547EC00DC) and look for all instances of `PROVISIONING_PROFILE[sdk=iphoneos*]`. These will be followed by a provisioning profile number.
3. Replace this profile number with the new profile number.

Watch Out!

You can only modify the `projects.pbxproj` file for the correct configurations. You can guess the configuration from the preceding lines in the `projects.pbxproj` file (`"CODE_SIGN_IDENTITY[sdk=iphoneos*]" = "iPhone Developer: Firstname Surname";` versus `"CODE_SIGN_IDENTITY[sdk=iphoneos*]" = "iPhone Distribution: CompanyName";`) and the subsequent line (`name = Release;`, `name = Debug;`, `name = Betatest;`, or `name = Appstore;`). One method that can save some time is to use `Betatest` for the ad hoc distribution and `Appstore` for the App Store distribution.

By the Way

If none of these methods work, you can create another project, copy all the files over, and set all the build options back up from scratch. This will take some time, but it will work. Keeping notes of what settings you've changed will help.

Debugging the Ambiguous Matches Error

If you get an error about ambiguous matches, it's possible that you installed the certificates multiple times. Delete the additional copies from the `Keychain Access` application.

Uploading Your Application to Your Device

Xcode sometimes is unable to upload even correctly signed applications. This section explains the steps you can take to fix the problem.

Power Cycling the iPhone

If your iPhone does not have enough memory to start the debug server and your application, Xcode will fail to upload and start an application to your device if a different application is running. It will also give up and fail for no apparent reason at other times. Turning the iPhone off by holding the top button and then swiping the red lock usually fixes this problem.

Every time you restart your iPhone, iTunes will start up and monopolize your computer. To prevent iTunes from opening go to iTunes Preferences, choose Devices, and then disable the Disable Automatic Syncing for iPhones and iPods check box (see Figure B.7). Now go to your device summary and disable the Open iTunes When This iPod Is Attached check box (see Figure B.8).

FIGURE B.7
Enabling automatic syncing

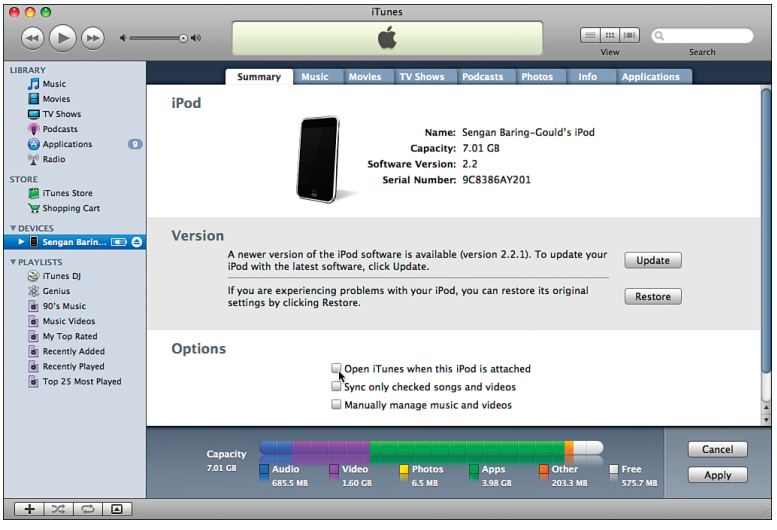

FIGURE B.8
Disabling opening iTunes when the iPhone is attached

If the app still won't load, delete its build directory, leave Xcode, reopen Xcode, and rebuild and try again. Xcode will refuse to install an app that requires an OS version greater than is installed on the iPhone. Check which SDK you are using by looking at the Active configuration in the upper left of the window shown in Figure B.5.

Watch Out!

If you delete your build directory and revert to an older snapshot, Xcode deletes all your files and then gives up, leaving you with an empty project directory. Don't panic! Xcode has put the files in the Trash folder. Furthermore, your snapshots are located in a folder within /var/folders.

Ensuring That the Application Was Really Signed

Another possible reason your application won't load is because the application wasn't signed. To fix this, first find your application in your project's build directory. For instance, for the debug version of the Calculator app, the build directory should contain the following:

```
Debug-iphoneos/Calculator.app/CodeResources
Debug-iphoneos/Calculator.app/embedded.mobileprovision
```

If it is not signed, first try emptying the caches and then rebuilding.

Rebuilding and Emptying Caches

If Xcode is not signing the application, it may think it has already done so. To reset Xcode, perform the following steps:

1. Choose the Build, Clean All Targets menu item.
2. Click the Xcode, Empty Caches menu item.
3. Quit Xcode and delete the build directory in your project.
4. Restart Xcode.

Look at the build results (the Build, Build Results menu item or (⇧Shift)-⌘-B) and make sure that they contain your provisioning profile (that looks like the `06B7817A-3CF3-49FD-A668-8BB547EC00DC.mobileprovision` file we used earlier) and the destination `embedded.mobileprovision` file.

For developer testing, the build results should also contain `iPhone Developer: *Firstname Surname*` in the `codesign` invocation.

For ad hoc distribution, the build results should contain `iPhone Distribution: *CompanyName*` in the `codesign` invocation and a reference to `Entitlements.plist` in the `ProcessingProductPackaging` invocation.

If these are lacking, make sure you set up your system as explained in the previous section, "Building Your Application to Run on a Device." If those settings are correct, check your keychain certificates.

Checking Your Keychain Certificates

Xcode will not sign your application or upload it if your keychain certificates are corrupted. To check them, perform the following steps:

1. Open the `Keychain Access` application.
2. The login keychain must be set to default: Right-click on it and choose the Make Keychain "login" Default menu item. This should clear the `A valid signing identity matching this profile could not be found on your keychain` error that appears in Xcode's Organizer window. Some Adobe products may change this setting.
3. In the login keychain, you should see:
   - `Apple Worldwide Developer Relations Certificate Authority` certificate
   - `iPhone Developer: Name Surname`
   - `iPhone Distribution: CompanyName`

   They should all be valid.
4. Check that Xcode can access the private keys. Click the My Certificates category, and open the disclosure triangle next to each certificate. There should be a key under each (see Figure B.9).
5. Right-click (Ctrl-click) the private key and choose Get Info to get an Access Control panel. Make sure the applications match those shown in Figure B.10.

FIGURE B.9 Keychain Access

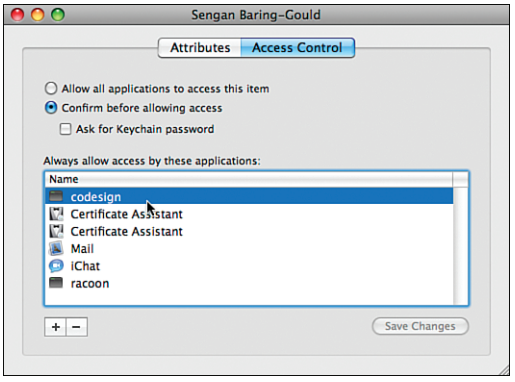

FIGURE B.10
Key Access Control

Did you Know?

Apple's Technical Q&A QA1618 at http://developer.apple.com/iphone/library/qa/qa2008/qa1618.html recommends that Online Certificate Status Protocol (OCSP) and Certificate Revocation List (CRL) are set to "Off" in Keychain Access's Preferences, Certificates menu.

Did you Know?

Apple sometimes expires your certificates. If they do so, and the new certificates don't work, you can try setting the system clock back to a date prior to the expiration.

Checking Your iPhone's Provisioning

A tool called the iPhone Configuration Utility is available from http://support.apple.com/downloads/. You can use this utility to ensure that your provisioning profile was correctly installed.

Old provisioning profiles can also be problematic. I've had to remove all the profiles on my device and reinstall new ones to upload my applications.

The iPhones Settings application also provides a section under "General" to list and remove provisioning profiles.

Finally, make sure that your device UDID was added to the list of supported devices in the Developer Program Portal. Make sure that the AppID in the Developer Program Portal matches the Bundle identifier you set in your `Info.plist` file.

Checking the Application Identifier

Some Application Identifiers do not work. When you encounter this problem, you'll see the following error in your Console Log:

```
entitlement 'application-identifier' has value not permitted by provisioning
profile
```

To work around this, try replacing the Application Identifier with a wildcard (*).

Checking for a Duplicate Distribution Profile

Apple's encryption system was not designed for small test applications. It is sometimes easier to reuse the same distribution profile for different applications. However, multiple applications may not share distribution profiles on a single iPhone.

To fix this problem, simply delete the test application from the iPhone, clean, and rebuild.

Creating Ad Hoc Profiles

Use Ad Hoc profiles to distribute your application to your beta testers, or within your small organization. To build an application using an Ad Hoc profile, perform the following steps:

1. Add a new file named `Entitlements.plist` of type "Code Signing, Entitlements" (see Figure B.11).

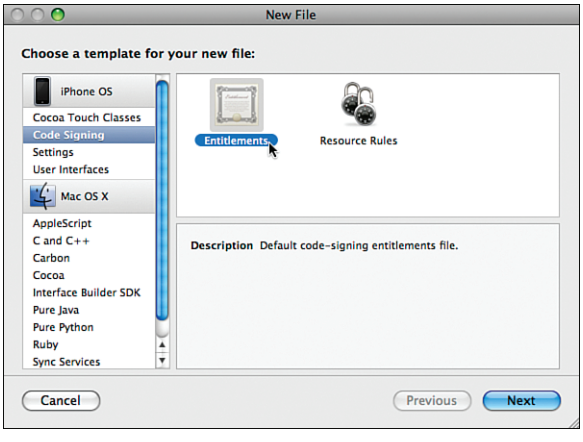

FIGURE B.11 Creating an `Entitlements.plist` file

2. Open the file and unclick the value (see Figure B.12).

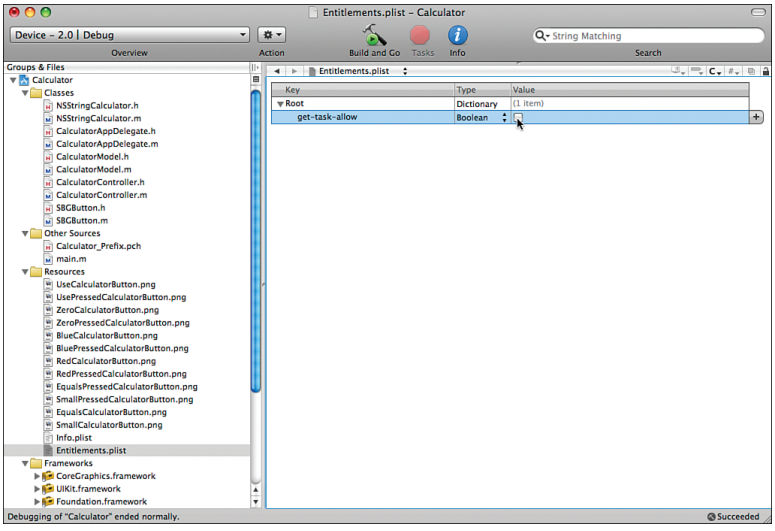

FIGURE B.12 `Entitlements.plist` file content

3. Add the file to the Code Signing Entitlements entry of the Code Signing section of the Targets' information panel.

You can add an image to be shown in iTunes to an ad hoc application by adding a 512x512 pixel iTunesArtwork JPEG file (without a `.jpg` or other file extension) to the Ad Hoc application's directory. Note that this only works for the Ad Hoc distribution builds installed via iTunes and will cause an error if you try to start the application from Xcode.

Checking Changes That Do Not Show Up

When switching between the device and the simulator, sometimes Xcode does not know whether or not it has built something. To force it to rebuild everything, choose the Build, Clean menu item (Shift-⌘-K), and rebuild. If this menu item is grayed out, Xcode thinks the app is running. If it is not running, you can force a rebuild by deleting the build directory in your application project folder and rebuilding.

Checking the App Store Executable

You can't test the executable you send to the App Store. However, Apple does provide clear instructions on how to check that your app was correctly code-signed by looking in the Build Results window. For App Store distribution, the build results should contain `iPhone Distribution: CompanyName` in the `codesign` invocation and a reference to `Entitlements.plist` in the `ProcessingProductPackaging` invocation.

Apple also provides an Application Loader, which verifies all the certificates, icons, and so on before submitting the file to Apple. You can download this application from the Manage Your Applications page in iTunes Connect (see Figure B.13).

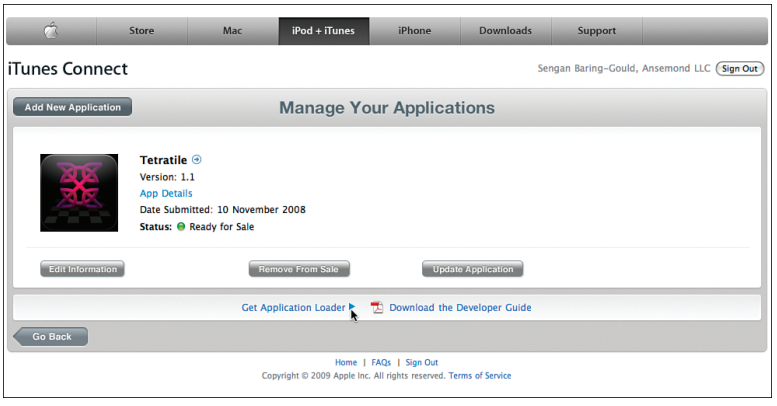

FIGURE B.13 Downloading the AppLoader

If the binary you submit is not correctly code-signed, Apple will reject it. Because it will be tested before your clients can download it, you needn't worry about a bad build hurting your reputation. However, you will lose time to market.

Did you Know?