Maintaining Your Signing Identities and Certificates

Code signing your app lets users trust that your app has been created by a source known to Apple and that it hasn't been tampered with. All apps must be code signed and provisioned to launch on a device, to use certain services, to be distributed for testing, or to be uploaded to iTunes Connect. Code signing uses cryptographic technology to digitally sign your app and installer package. You create signing identities—stored in your keychain—and certificates—stored in your developer account—to sign and provision your app. These assets uniquely identify you or your team, so it's important to keep them safe. This chapter covers common tasks that you perform to protect and maintain your signing identities and certificates over the lifetime of your project.

For the types of the certificates you use to develop, test, and distribute your app, refer to Your Signing Certificates in Depth.

Important: A team agent can perform all the tasks in this chapter. If you are a team admin or team member, see Team Privileges for the tasks you can perform.

About Signing Identities and Certificates

Code signing your app allows the operating system to identify who signed your app and to verify that your app hasn't been modified since you signed it. Your app's executable code is protected by its signature because the signature becomes invalid if any of the executable code in the app bundle changes. Note that resources such as images and nib files aren't signed; therefore, a change to these files doesn't invalidate the signature.

Code signing is used in combination with your App ID, provisioning profile, and entitlements to ensure that:

- Your app is built and signed by you or a trusted team member.
- Apps signed by you or your team run only on designated development devices.
- Apps run only on the test devices you specify.
- Your app isn't using app services you didn't add to your app.
- Only you can upload builds of your app to iTunes Connect.
- If you choose to distribute outside of the store (Mac only), the app can't be modified and distributed by someone else.

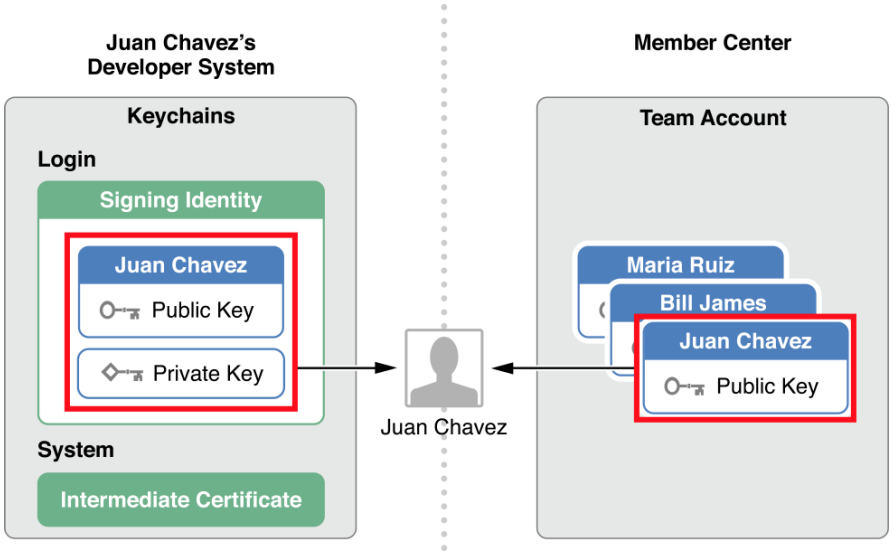
Code signing also allows your app's signature to be removed and re-signed by a trusted source. For example, you sign your app before uploading it to iTunes Connect, but Apple re-signs it before distributing it to customers. Also, you can re-sign and submit a fully tested development build of your app to the store.

Xcode uses your signing identity to sign your app during the build process. This *signing identity* consists of a public-private key pair that Apple issues. The public-private key pair is stored in your keychain, and used by cryptographic functions to generate the signature. The certificate stored in your developer account contains just the public key. An *intermediate certificate* is also required to be in your keychain to ensure that your certificate is issued by a *certificate authority*.

Signing requires that you have both the signing identity and the intermediate certificate installed in your keychain. When you install Xcode, Apple's intermediate certificates are added to your keychain for you. You use Xcode to create your signing identity and sign your app. Your signing identity is added to your keychain, and the corresponding certificate is added to your developer account.

Signing identities are used to sign your app or installer package. A *development certificate* identifies you, as a team member, in a development provisioning profile that allows apps signed by you to launch on devices. A *distribution certificate* identifies your team or organization in a distribution provisioning profile and allows you to submit your app to the store. Only a team agent or an admin can create a distribution certificate. You use the same development and distribution certificates for iOS, tvOS, and watchOS apps. You use different development and distribution certificates for Mac apps. For a complete list of certificate types, refer to Your Signing Certificates in Depth.

For an organization, other team members have their own signing identities installed on their Mac computers. The developer account contains a repository for all of the combined team assets but doesn't store any of the private keys.



Because the private key is stored locally on your Mac, protect it as you would an account password. Keep a secure backup of your public-private key pair. If the private key is lost, you'll have to create an entirely new identity to sign code. Worse, if someone else has your private key, that person may be able to impersonate you. In the wrong hands, someone might attempt to distribute an app that contains malicious code. Not only could that cause the app to be rejected, it could also mean your developer credentials could be revoked by Apple. Private keys are stored only in the keychain and can't be retrieved if lost.

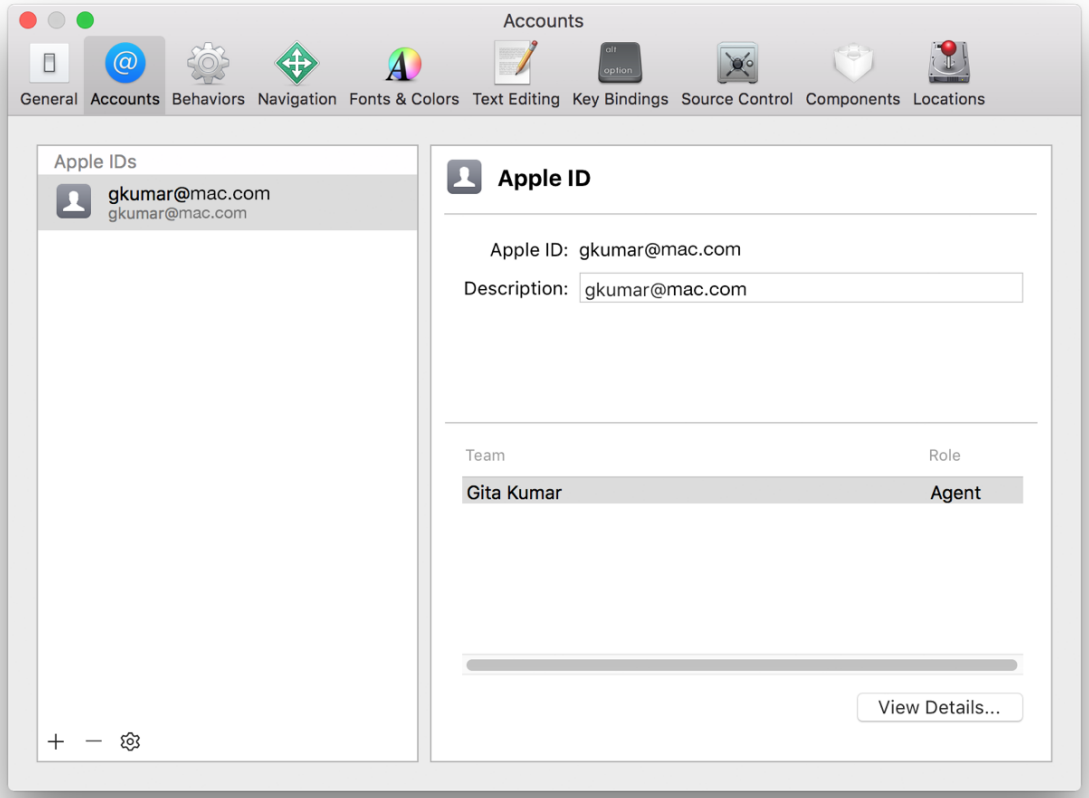
If you want to code sign your app using another Mac, you export your developer profile on the Mac you used to create your certificates and import it on the other Mac. You can also share distribution certificates among multiple team agents using this feature. (Team members should not share development certificates.)

Viewing Signing Identities and Provisioning Profiles

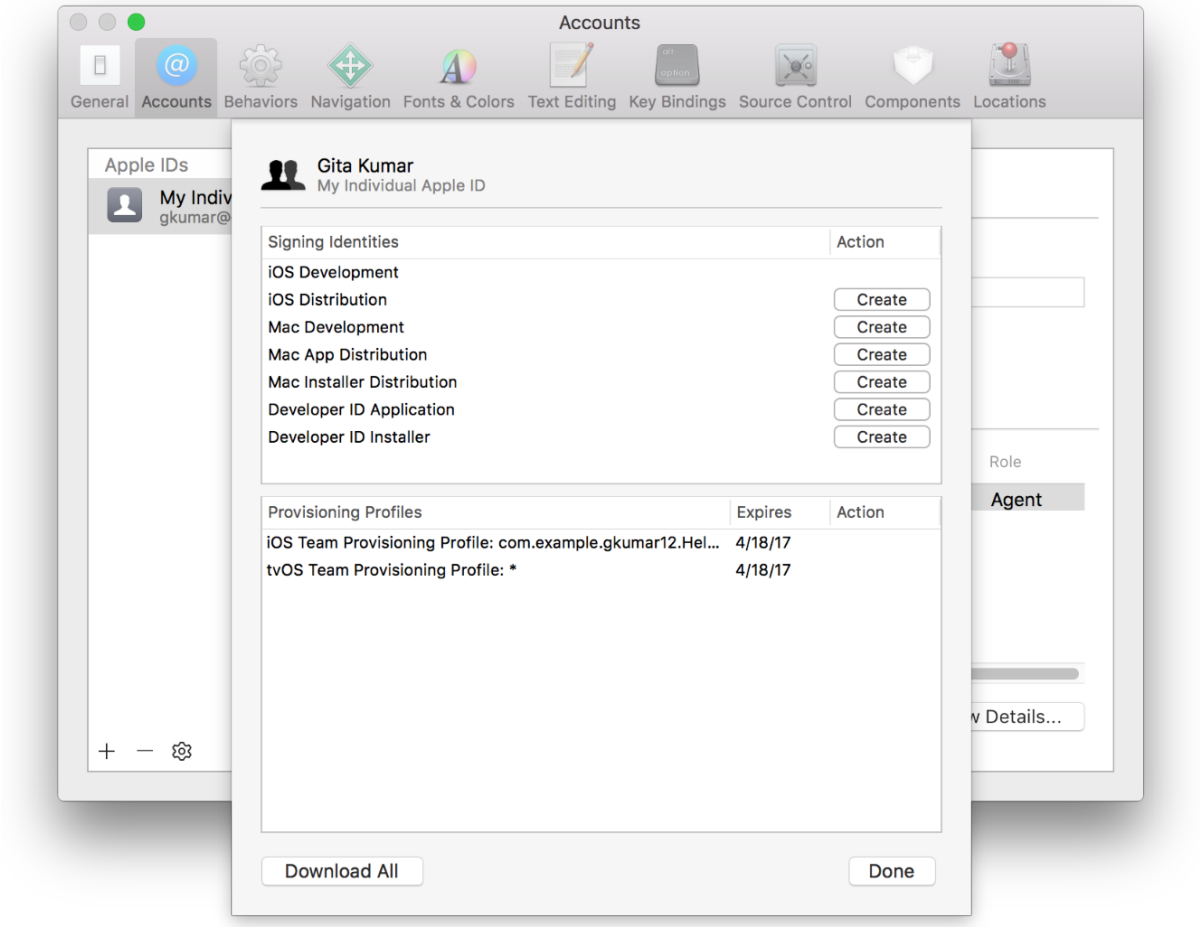
To verify or troubleshoot your certificates and profiles, view them in Xcode Accounts preferences. Although Xcode manages these assets for you, you may occasionally need to create or revoke specific certificates and to download your provisioning profiles.

To view account details

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.
3. Select the team you want to view, and click View Details.



In the dialog that appears, view your signing identities and provisioning profiles. If a Create button appears next to a certificate, it hasn't been created yet. If a Download button appears next to a provisioning profile, it's not on your Mac.



4. Click Done.

Creating Signing Identities

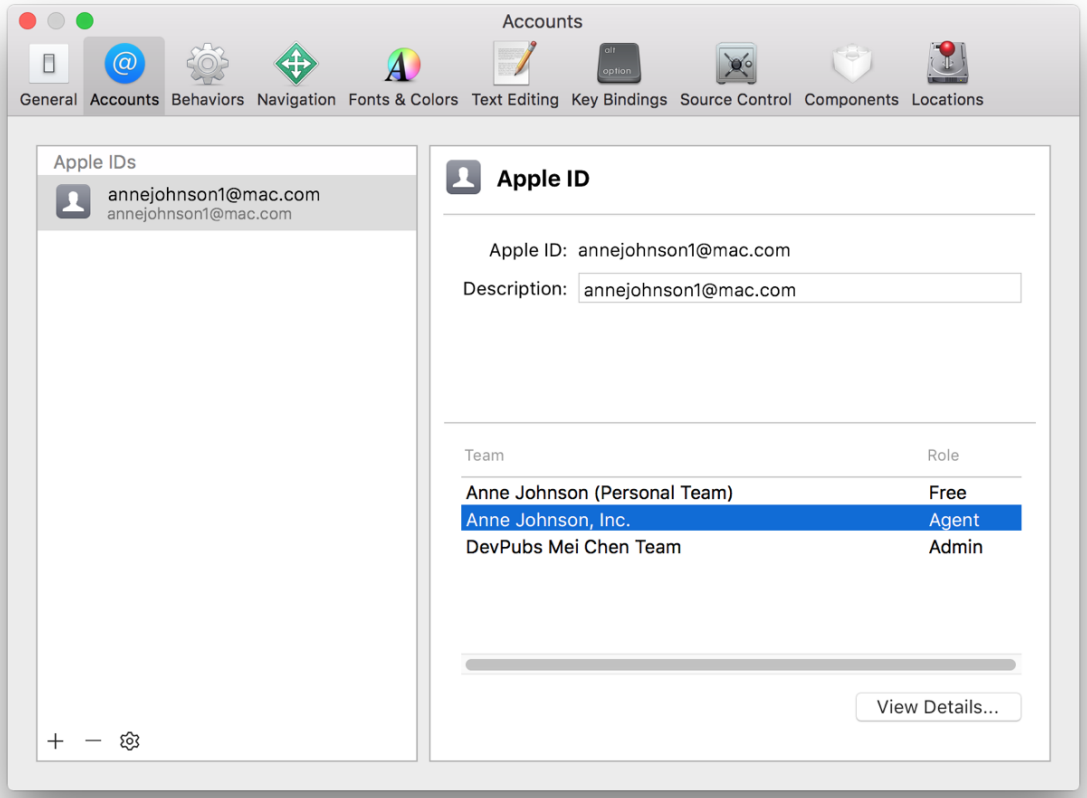
Before you can code sign your app, you create your development certificate and later, a distribution certificate to upload your app to iTunes Connect. You can create all the types of certificates and signing identities you need using Xcode. Xcode creates, downloads, and installs your signing identities for you.

All team members can create their own development certificate. Only a team agent or admin can create a distribution certificate. Only a team agent can create a Developer ID certificate. If you have an organization membership, read Managing Your Developer Account Team for a description of team roles and tasks that team agents perform on behalf of team members.

Xcode creates your development certificate for you when you need it. For example, when you assign your project to a team or create the team provisioning profile, as described in Configuring Identity and Team Settings, Xcode creates your development certificate and adds it to the team provisioning profile. Because of this, you typically create distribution certificates using the Xcode Preferences window.

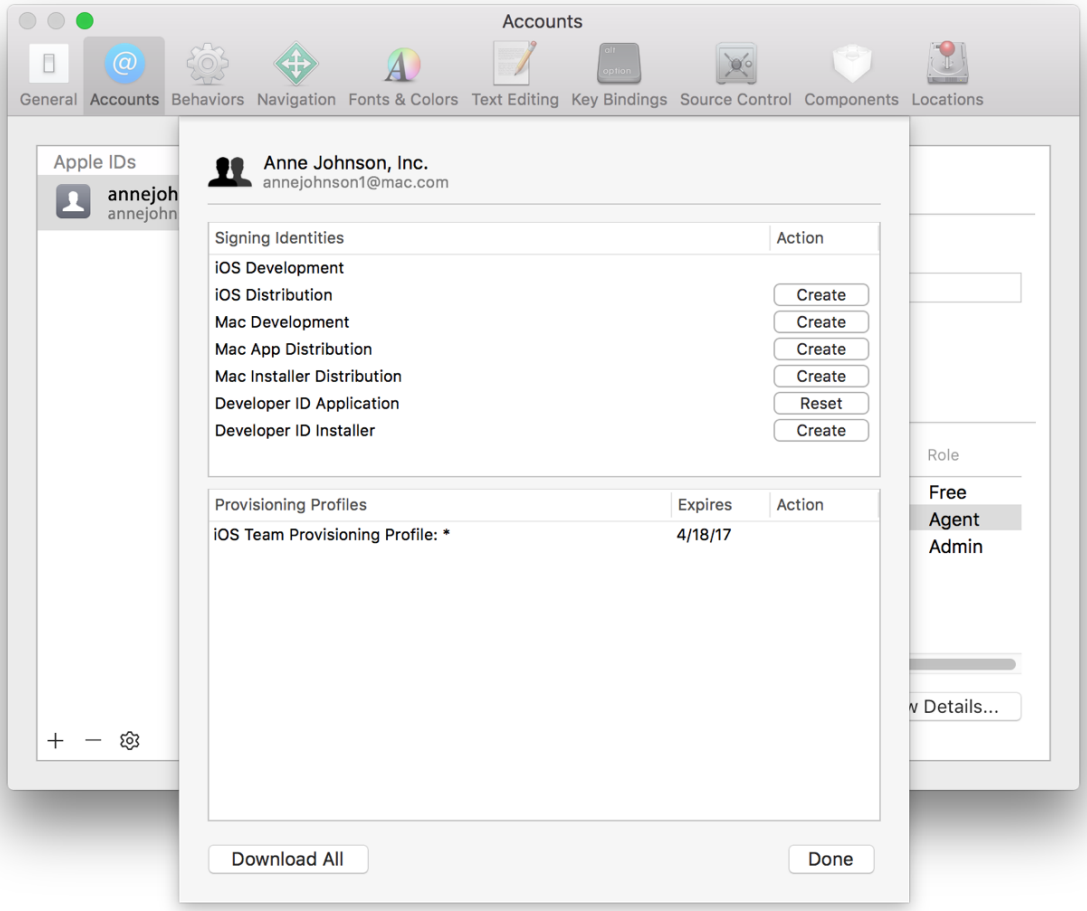
To create a signing identity

1. In the Xcode Preferences window, click Accounts.
2. Select the team you want to use, and click View Details.



3. In the dialog that appears, click the Create button in the row of the type of certificate you want to create.

   If the Create button is disabled, you are not allowed to create that type of signing identity. For a description of each type of certificate, refer to Table 14-2.



After the signing identity is created, the Create button disappears.

4. To return to Accounts preferences, click Done.

You can now export your signing identities to create a backup, as described in Exporting Your Developer Profile.

Verifying Your Steps

Verify that your certificates are correct and ready for use. Certificates must be valid in order to sign your app—and for a Mac app, to sign your installer package.

The first time you verify your certificates, verify them in Xcode, Keychain Access, and your developer account to learn where they're located and how they appear in each tool. Keychain Access and your developer account display the expiration dates of your signing identities and certificates. Later, you'll use Keychain Access for troubleshooting.

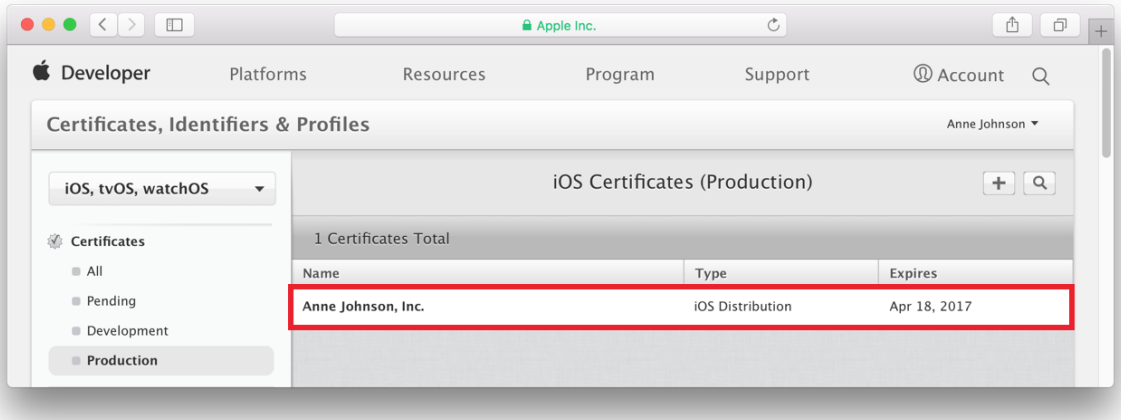
Verifying Using Your Developer Account

Your developer account should show the same certificates you see in Xcode and Keychain Access because it stores the public keys.

To verify signing certificates using your developer account

1. Sign in to developer.apple.com/account, and click Certificates, IDs & Profiles.
2. In the Certificates section of the sidebar, select Development or Production depending on the type of certificate you want to verify.

   The type and expiration date of the certificate should match the information that you view in Keychain Access.



Note: The name of the certificate in Keychain is different from the certificate type that appears in Xcode and your developer account. For the mapping between the certificate names and types that appear in the tools, refer to Table 14-2.

Verifying Using Keychain Access

Keychain Access shows the private and public keys for each of your signing identities.

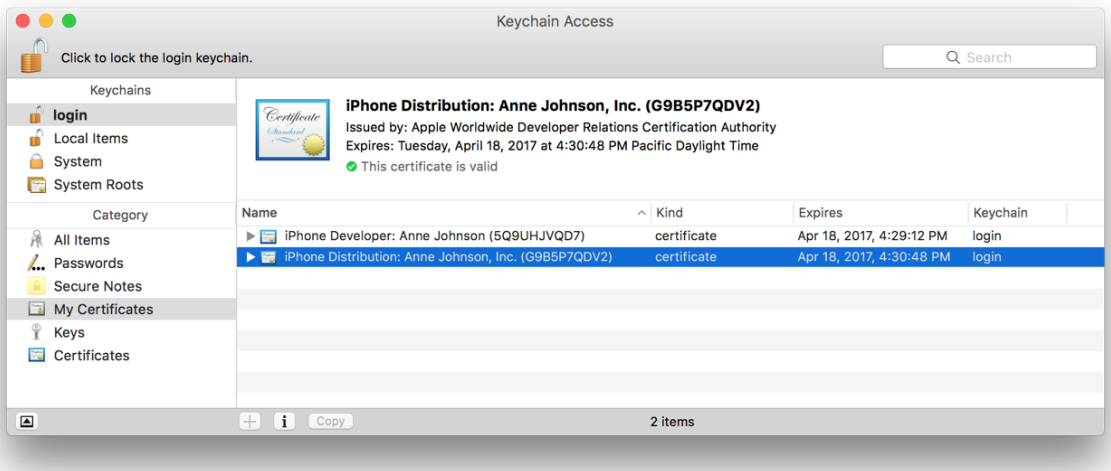
To verify signing identities using Keychain Access

1. Launch Keychain Access located in `/Applications/Utilities`.

   When you create a development or distribution certificate using Xcode, the certificate is automatically installed in your login keychain.

2. In the left pane, select “login” in the Keychains section and select Certificates in the Category section.

   Your development and distribution certificates appear in the Certificates category in Keychain Access. The name of the development certificate begins with the text “iPhone Developer” for iOS, tvOS, and watchOS apps, and “Mac Developer” for Mac apps, followed by your name (development certificates belong to a person).



Other types of certificates also appear in the Certificates category of Keychain Access.

Important: Keep your personal keychain items in your login keychain. If your certificates don't appear in the login keychain, it may not be the default keychain. The default keychain appears in bold in the Keychains column in Keychain Access. If the default keychain isn't login, select login in the Keychains column and choose File > Make Keychain "login" Default.

3. Verify that there's a disclosure triangle to the left of the certificate.

   If you click the disclosure triangle next to the certificate name, your private key appears. If the disclosure triangle doesn't appear, you're missing your private key. (Read No Code Signing Identities Found to fix this issue.)

4. Verify that the certificates are valid.

   When you select a certificate, a green circle containing a checkmark appears in Keychain Access above the list of certificates. The text next to the checkmark should read "This certificate is valid."

Troubleshooting

If the certificates shown in Xcode and Keychain Access don't match your certificates in your developer account, read Certificate Issues for information about how to resolve the discrepancies.

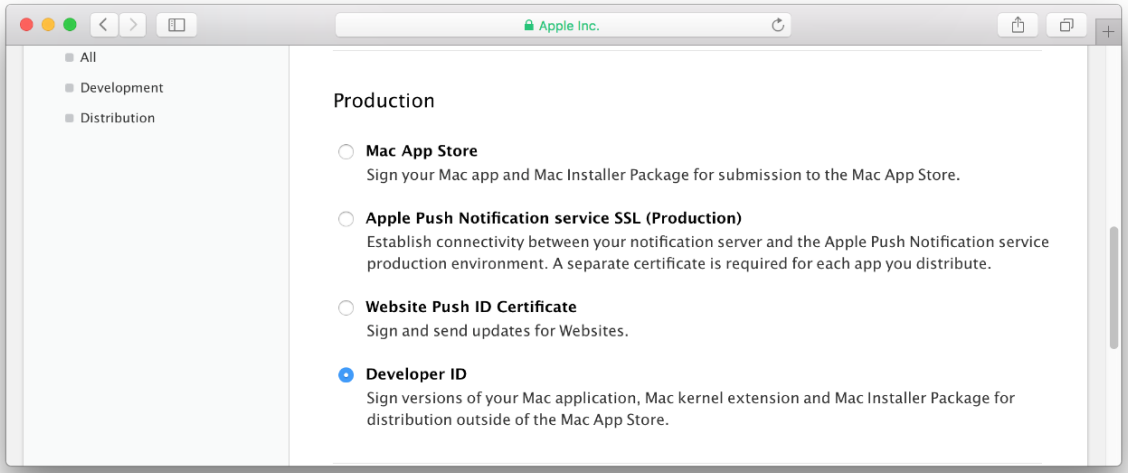
Creating Additional Developer ID Certificates

Developer ID certificates are used to distribute your app outside of the Mac App Store. Create your Developer ID certificates, along with other types of certificates, using Xcode, as described in Creating Signing Identities. If you want more Developer ID certificates, you can create up to five of each type using your developer account.

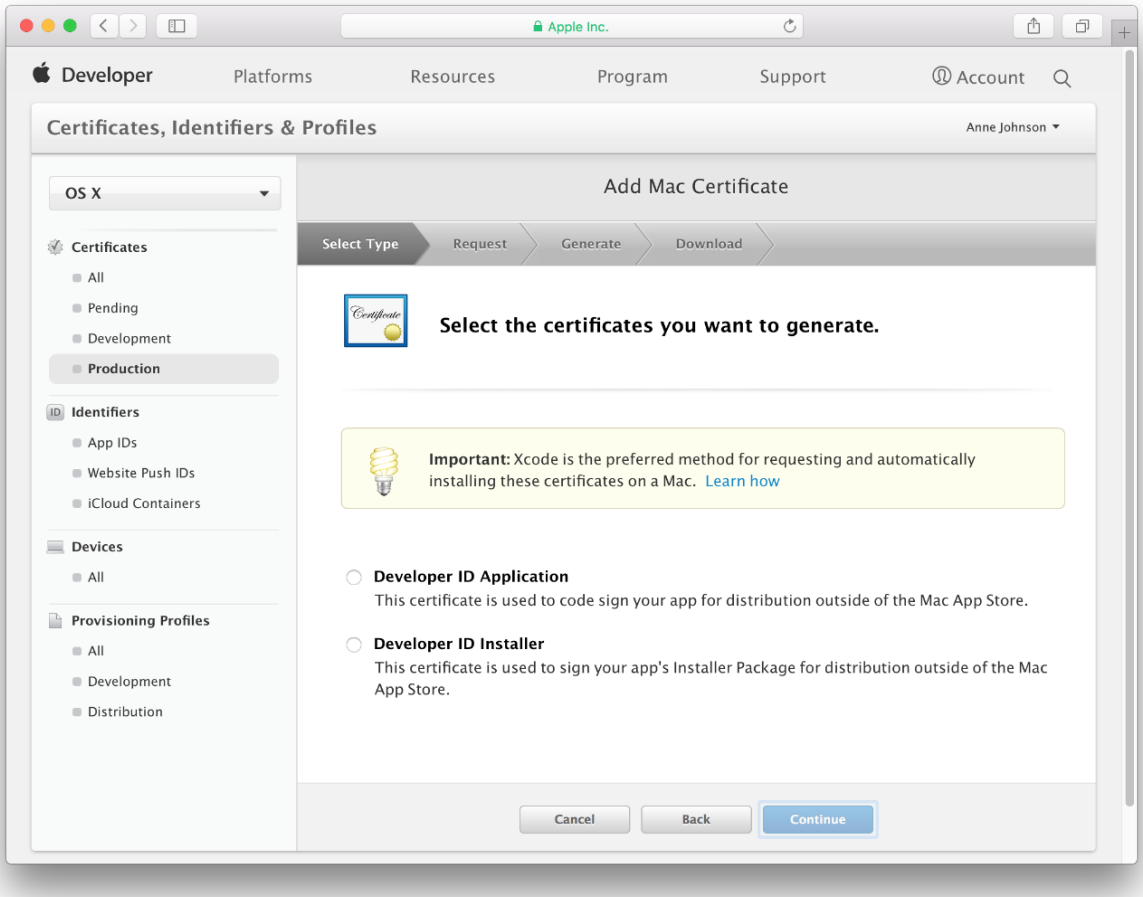
Important: Only team agents can create additional Developer ID certificates.

To create a Developer ID certificate

1. Sign in to developer.apple.com/account, and click Certificates, IDs & Profiles.
2. In the sidebar, choose OS X from the pop-up menu, and under Certificates, select All.
3. Click the Add button (+) in the upper-right corner.
4. Select Developer ID under Production, and click Continue.



5. Select the certificate type—Developer ID Application or Developer ID Installer—and click Continue.



6. Follow the instructions to create a certificate signing request (CSR) using Keychain Access, and click Continue.
7. Click Choose File.
8. Select a CSR file (with a `.certSigningRequest` extension), and click Choose.
9. Click Continue.
10. Click Download.

    The certificate file appears in your `Downloads` folder.

To install the Developer ID certificate in your keychain, double-click the downloaded certificate file (with a `.cer` extension). The Developer ID certificate appears in the My Certificates category in Keychain Access.

Installing Missing Intermediate Certificate Authorities

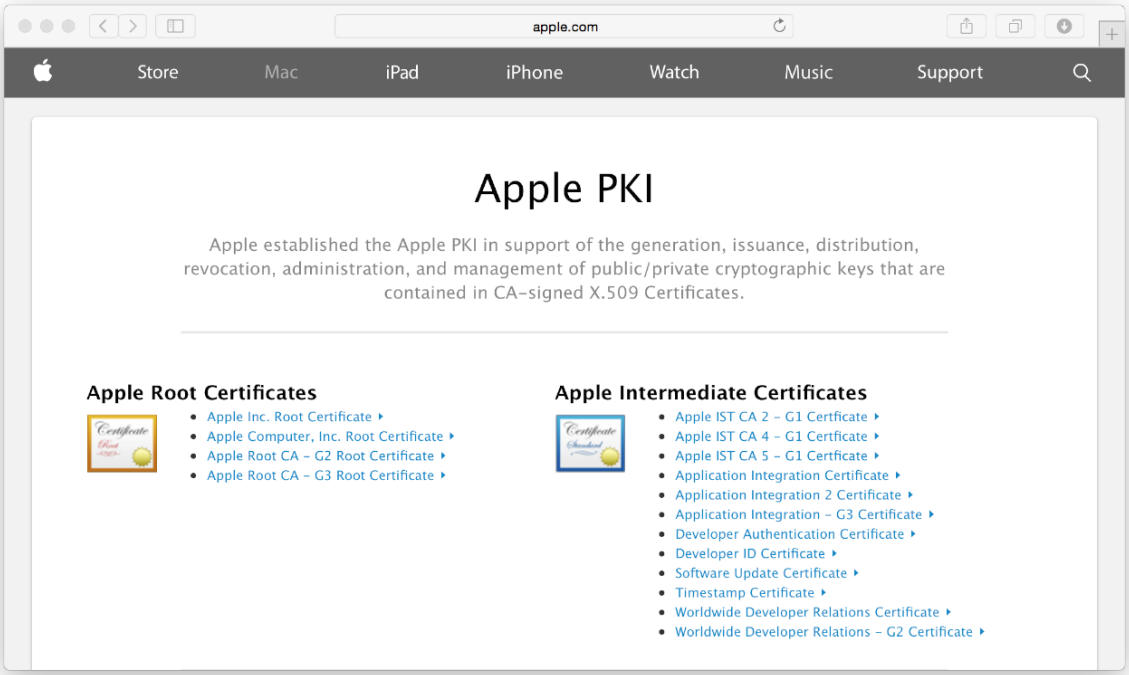
To use your certificates, the correct intermediate certificate needs to be in your keychain. An intermediate certificate ensures that your certificates were issued by a trusted source. The intermediate certificate, named Apple Worldwide Developer Relations Certification Authority, is installed in your system keychain when you install Xcode. The intermediate certificate for Developer ID certificates is called Developer ID Certification Authority. If you accidentally remove an intermediate certificate, you can install it again.

First try downloading your provisioning profiles in Xcode, as described in Downloading Provisioning Profiles in Xcode, to install missing intermediate certificates. If that doesn't work for you, download and install the missing intermediate certificate.

To install a missing intermediate certificate

1. Go to http://www.apple.com/certificateauthority.
2. Under Apple Intermediate Certificates, click the link for the intermediate certificate you're missing.

   A certificate file, with a `.cer` extension, appears in your Downloads folder.



3. Double-click the certificate file to install it in your system keychain.

Exporting and Importing Certificates and Profiles

After Xcode creates your certificates and profiles for you, export them to create a backup of all your assets. You do this to, for example, transfer your assets to another Mac you use for development or repair a certificate if the private key is missing. Downloading your profiles in Xcode won't replace a missing private key. Instead, import your certificates and profiles from a backup.

The export file, called a *developer profile*, contains the following team assets:

- Development certificates
- Distribution certificates
- Provisioning profiles

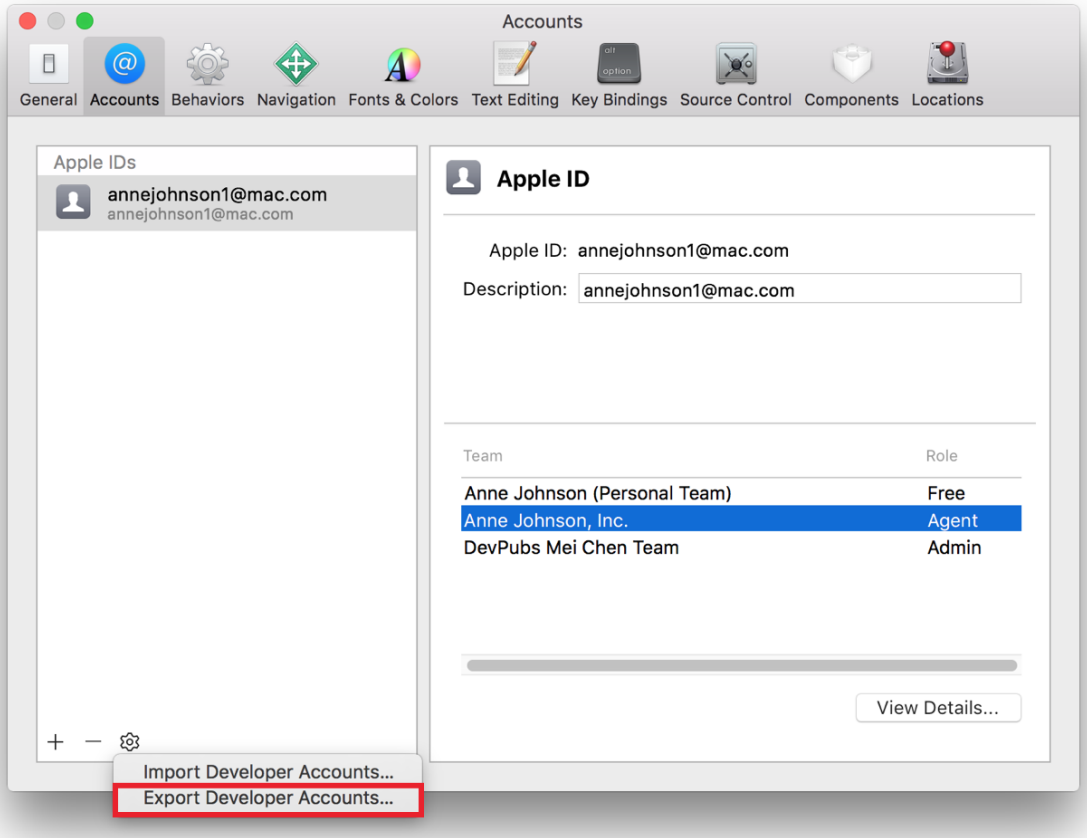
You can also export selected certificates to share with other team members. In this case, the export file contains just the certificates you select.

Exporting Your Developer Profile

Because the developer profile represents your credentials to sign and submit apps to the store, Xcode encrypts and password-protects the exported file.

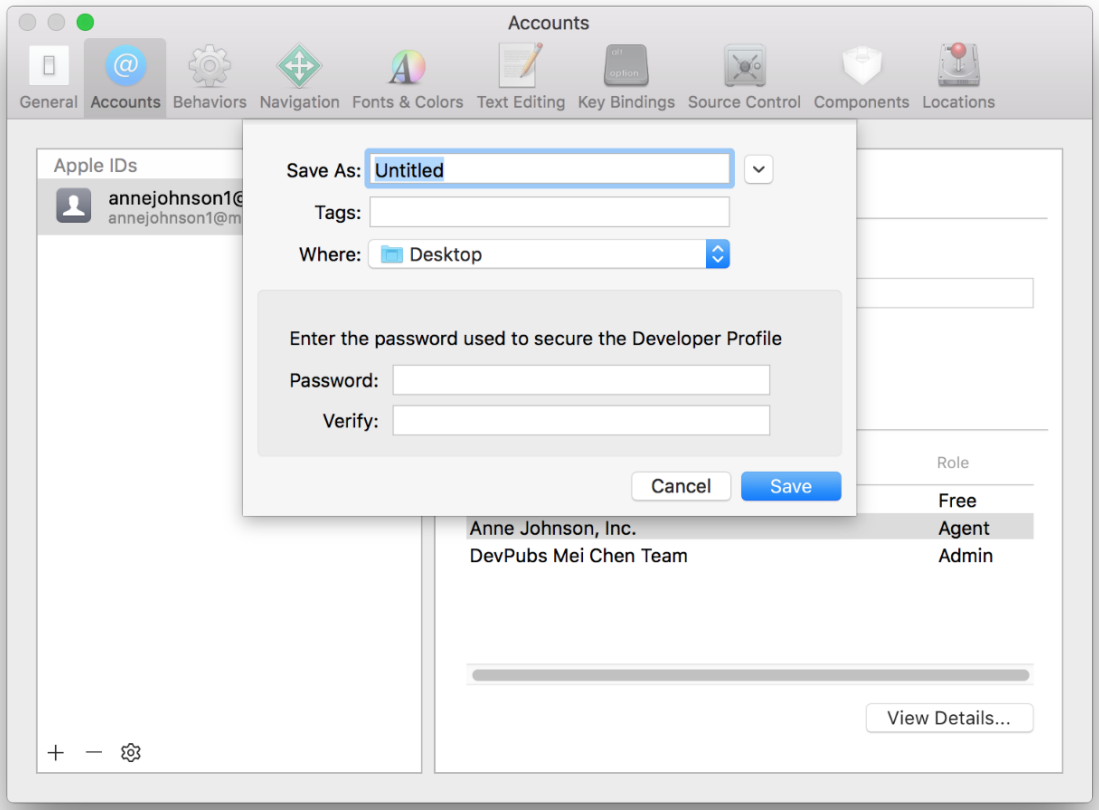
To export your developer account assets

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.
3. Click the Action button (the gear icon to the right of the Delete button) in the lower-left corner, and choose Export Developer Accounts from the pop-up menu.



4. Enter a filename in the Save As field and a password in both the Password and Verify fields.

   The file is encrypted and password protected.



5. Click Save.

   The file is saved to the location you specified with a `.developerprofile` extension.

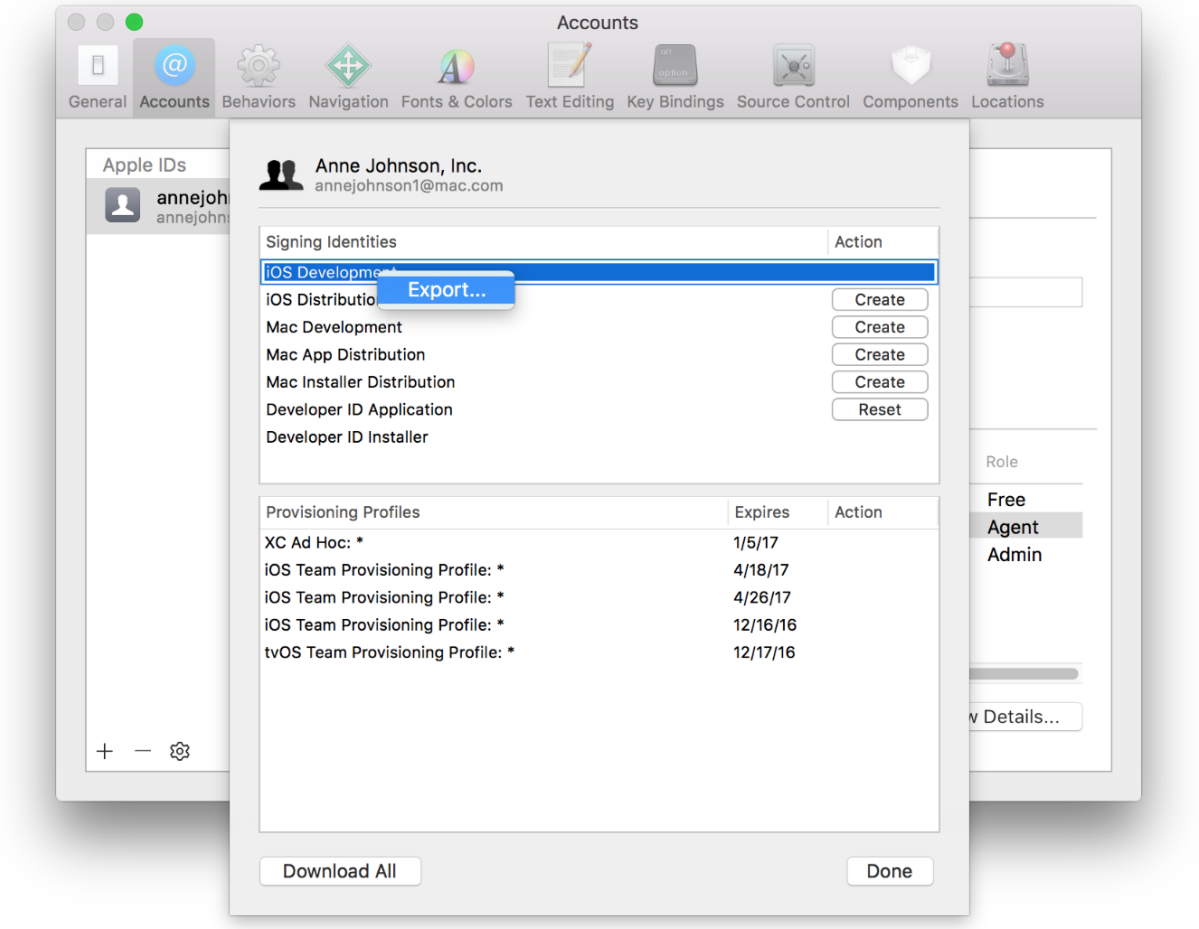
6. In the dialog that appears, click OK.

Exporting Selected Certificates

To export a few certificates and exclude the profiles, select the certificates in the details dialog.

To export selected certificates

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.
3. Select the team you want to view, and click View Details.
4. Control-click the certificate you want to export in the Signing Identities table and choose Export from the pop-up menu.



5. Enter a filename in the Save As field and a password in both the Password and Verify fields.

   The file is encrypted and password protected.

6. Click Save.

   The file is saved to the location you specified with a `.p12` extension.

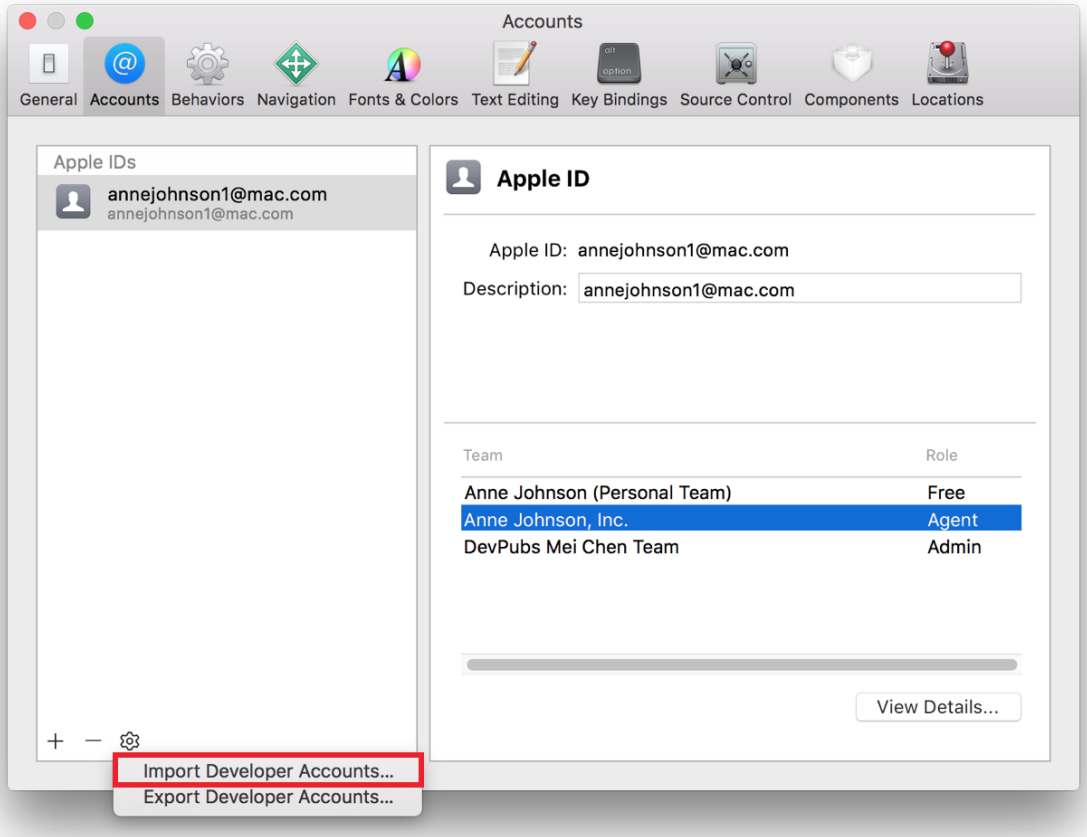
Alternatively, you can export certificates using the security(1) command-line utility.

Importing Your Developer Profile

You import your developer profile to restore missing private keys or when you want to switch to another Mac.

To import your developer account assets

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.
3. Click the Action button (the gear icon) in the lower-left corner, and choose Import Developer Accounts from the pop-up menu.



4. Locate and select the file containing your developer profile, and click Open.

   The file should have a `.developerprofile` extension.

5. Enter the password you used to encrypt the file, and click OK.
6. In the dialog that appears, click OK.

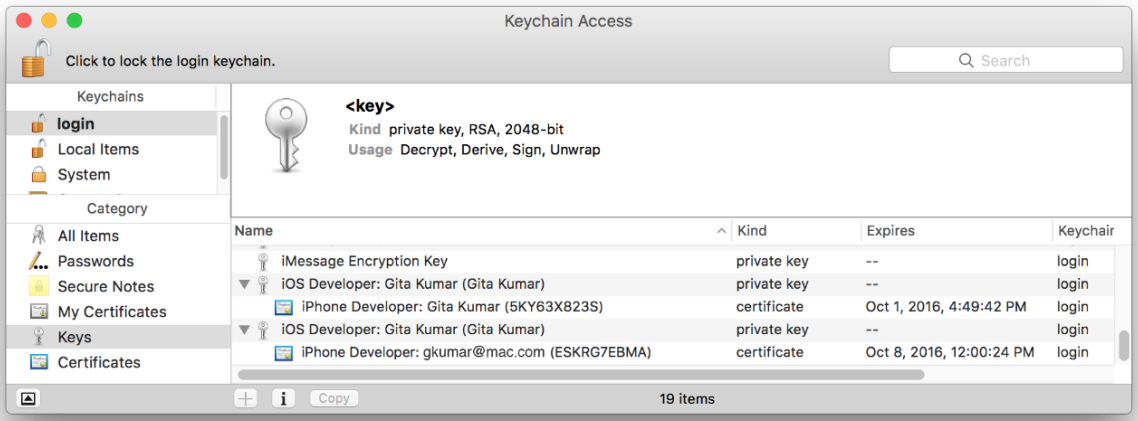
Removing Signing Identities from Your Keychain

You remove signing identities from your keychain if they're invalid, no longer used (perhaps they belong to a previous team you were a member of), or are missing the private key and consequently, aren't usable. If you're missing a private key and have a backup of your signing identities, import your developer profile, as described in Importing Your Developer Profile, immediately after removing the signing identities. If you remove signing identities for some other reason, read all the steps in Re-Creating Certificates and Updating Related Provisioning Profiles to avoid code signing issues later.

> **Warning:** You can't re-create a private key once you remove it from your keychain unless you import it from a developer profile file. Without the private keys, you can't sign apps using the certificates.

To remove signing identities from your keychain

1. Launch Keychain Access (located in `/Applications/Utilities`).
2. In the Category section, select Keys.
3. Click the disclosure triangles for all the private keys to reveal the associated certificates.



4. Select all of the private keys associated with the certificates that you want to remove.

   For how to recognize the type of certificate by the name as it appears in Keychain Access, refer to Table 14-2.

5. Select the corresponding public key for each private key.
6. Press Delete (on the keyboard), and when a dialog appears, click Delete.
7. In the Category section, select Certificates.
8. Select all of the certificates that you want to remove.

   The certificates won't have private keys.

9. Press Delete (on the keyboard), and when a dialog appears, click Delete.

Revoking Certificates

You revoke certificates when you no longer need them or when you want to re-create them because of another code signing issue (refer to Certificate Issues for the types of problems that can occur). You also revoke certificates if you suspect that they have been compromised. If you're a team admin for an organization, you may want to revoke development certificates of team members who no longer work on your project. Revoking certificates may invalidate provisioning profiles, so read all the steps in Re-Creating Certificates and Updating Related Provisioning Profiles to avoid code signing issues later.

> **Important:** Revoking development or distribution certificates doesn't affect apps that you've submitted to the store nor does it affect your ability to update them.

Revoking Privileges

Table 14-1 lists the types of certificates that each team member can revoke. Individual developers are the team agent for their one-person team, which means they have permission to revoke all types of development and distribution certificates except as indicated. For an organization, any team member can revoke his or her own development certificate, but a team member can only revoke distribution certificates if he or she is a team agent or admin.

Table 14-1 Team certificate revoking privileges

Type of certificate	Team agent	Team admin	Team member
Your development certificates: iOS Development Mac Development	✓	✓	✓
Other team admin and member certificates:			

iOS Development Mac Development	✓	✓	✗
The team agent's certificate: iOS Development Mac Development	✓	✗	✗
Store distribution certificates: iOS Distribution Mac App Distribution Mac Installer Distribution	✓	✓	✗
Developer ID certificates: Developer ID Application Developer ID Installer	✗	✗	✗
Push notification certificates: APNs Development iOS APNs Production iOS APNs Development Mac APNs Production Mac	✓	✓	✗
Pass certificate: Pass Type ID	✗	✗	✗

You can't revoke Developer ID or Pass Type ID certificates using your developer account. Instead, send a request to Apple at `product-security@apple.com` to revoke these types of certificates.

If Apple revokes your Developer ID certificate, users can no longer install apps that have been signed with that certificate. Instead of revoking a Developer ID certificate, you can create additional Developer ID certificates using your developer account, as described in Creating Additional Developer ID Certificates.

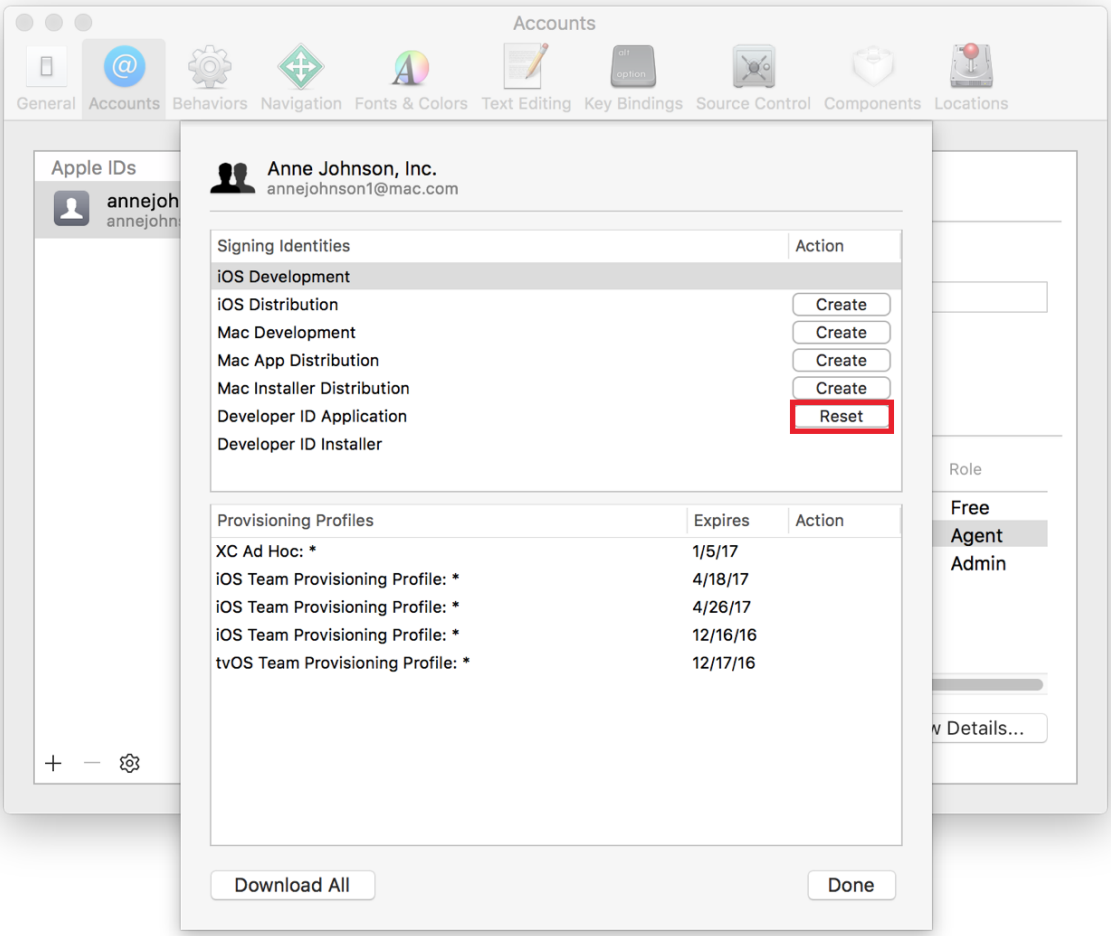
Resetting Certificates Using Xcode

If Xcode detects an issue with a signing identity, it displays an appropriate action in Accounts preferences. If Xcode displays a Create button, the signing identity doesn't exist in your developer account or on your Mac. If Xcode displays a Reset button, the signing identity is not usable on your Mac—for example, it is missing the private key. If you click the Reset button, Xcode revokes and re-creates the corresponding certificate.

Otherwise, you can use your developer account to revoke the certificate, as described in Revoking Certificates Using the Developer Account. If you are a team admin, use the developer account to revoke team member development certificates.

To reset a certificate using Xcode

1. Choose Xcode > Preferences, and click Accounts at the top of the window.
2. Select your team, and click View Details.
3. In the dialog that appears, click the Reset button in the Action column for the signing identity you want to revoke and re-create.



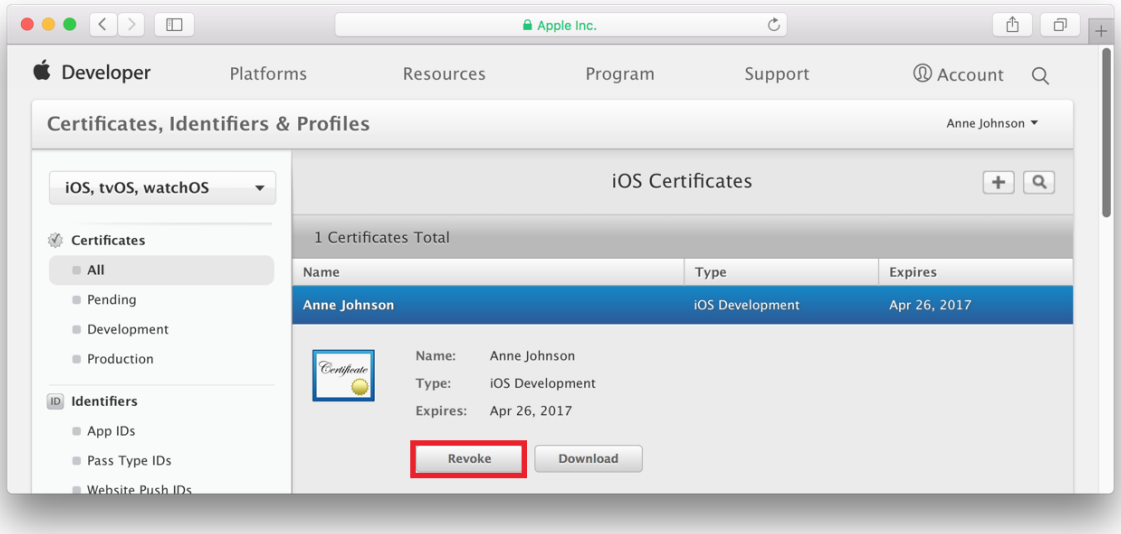
4. In the dialog that appears, click Done.

Revoking Certificates Using the Developer Account

Use the developer account to revoke all types of certificates belonging to your team. For example, revoke development certificates for other team members or distribution certificates that you no longer need or want to re-create.

To revoke a certificate using the developer account

1. Sign in to developer.apple.com/account, and click Certificates, IDs & Profiles.
2. Under Certificates, select All.
3. Select the certificate you want to revoke, and click Revoke.



4. In the dialog that appears, click Revoke.

Replacing Expired Certificates

When your development or distribution certificate expires, remove it and create a new certificate in Xcode. Follow the same steps to re-create certificates, as described in Re-Creating Certificates and Updating Related Provisioning Profiles. Use Keychain Access or your developer account to view the expiration dates of your signing identities and certificates, as described in Verifying Using Your Developer Account and Verifying Using Keychain Access.

Re-Creating Certificates and Updating Related Provisioning Profiles

Re-creating certificates and updating related provisioning profiles isn't a simple task, because these assets are related and reside on both your Mac and in your developer account. If you revoke a certificate, any provisioning profile that contains that certificate becomes invalid. Xcode will regenerate team provisioning profiles for you as needed, but you manage other types of provisioning profiles yourself. This section covers all the steps you perform to fully restore your code signing assets.

There are several reasons you might want to re-create your certificates and update related provisioning profiles. For example, you do this if:

- You accidentally removed one or more private keys from your keychain and don't have a backup to restore from
- You want to remove revoked or expired certificates and their related provisioning profiles
- You want to remove all the certificates and provisioning profiles from a previous team before you join another team
- You use multiple Mac computers for development or belong to multiple teams, and you want to set up your Mac for a new project

However, if you're experiencing certificate, provisioning, or build issues, review Certificate Issues first before performing these steps because removing your certificates is irreversible.

Choose the certificates you want to re-create. For example, if you experience problems running your app on a device, you may only need to re-create your development certificate. Keep in mind that re-creating a distribution certificate doesn't affect your development certificates or development provisioning profiles. Similarly, re-creating a development certificate doesn't affect your distribution certificate or distribution provisioning profiles.

Important: Re-creating your development or distribution certificates doesn't affect apps that you've submitted to the store nor does it affect your ability to update them.

To re-create your certificates and update related provisioning profiles

1. Revoke the certificates using your developer account, as described in Revoking Certificates.

   Provisioning profiles containing a revoked certificate become invalid. For example, if you revoke a team member development certificate, all the development provisioning profiles containing that certificate become invalid:

2. If necessary, remove the signing identities for these certificates from your keychain, as described in Removing Signing Identities from Your Keychain.

   If you revoke your own development or distribution certificate, remove the corresponding signing identity from your keychain. Otherwise, the owner of the certificate should remove the signing identity on his or her Mac.

3. Optionally, create new certificates using Xcode, as described in Creating Signing Identities.

   If you revoke a development certificate, Xcode displays a warning message below the Team pop-up menu on the General pane. Clicking the Fix Issue button creates a development certificate and regenerates the team provisioning profile used by that Xcode project so that it no longer appears invalid in your developer account.

4. Remove or regenerate other types of provisioning profiles that contain the revoked certificates, as described in Editing Provisioning Profiles in Your Developer Account.

   Xcode doesn't automatically regenerate distribution provisioning profiles or custom development provisioning profiles you create using your developer account.

5. If you changed provisioning profiles in your developer account, download provisioning profiles in Xcode, as described in Downloading Provisioning Profiles from Your Developer Account.
6. If necessary, install the modified provisioning profiles on your devices, as described in Verifying and Removing Provisioning Profiles on Devices.
7. Once the certificates are repaired on the primary Mac, export your developer profile, as described in Exporting Your Developer Profile.

If you're repairing multiple Mac computers, perform these additional steps on the other Mac computers:

1. Remove the signing identities from your keychain, as described in Removing Signing Identities from Your Keychain.
2. Import your developer profile, as described in Importing Your Developer Profile, that you created on the original Mac.

Your Signing Certificates in Depth

Your code signing identities, stored in your keychain, represent your Apple Developer Program development and distribution credentials. Become familiar with the names of these certificates, because they appear in menus,

and with the types of certificates, because they appear in lists, so that you don't accidentally remove them from your keychain or your developer account.

There are different types of signing certificates for different purposes. *Development certificates* identify a person on your team and are used to run an app on a device. During development and testing, you're required to sign all iOS, tvOS, and watchOS apps that run on devices, and Mac apps that use certain app services like iCloud and Game Center.

Distribution certificates identify the team and are used to submit your app to the store or for a Mac app, distribute it outside of the store. If you're an organization, distribution certificates can be shared by team members who have permission to submit your app. There are multiple kinds of distribution certificates, each associated with a specific method of distribution. Different code signing identities are also used for Mac apps.

Signing certificates are issued and authorized by Apple. You must have the intermediate certificate provided by Apple installed in your system keychain to use your certificate; otherwise, it's invalid. The intermediate certificates provided by Apple and installed by Xcode are:

- **Apple Worldwide Developer Relations Certification Authority.** Used to validate development and store certificates.
- **Developer ID Certification Authority.** Used to validate a Developer ID certificate for distribution outside of the Mac App Store.

Refer to Table 14-2 for the mapping between the type of the certificate, the name of the certificate as it appears in Keychain Access, and the purpose of each.

Your developer account displays the team name (or person's name) and type for each certificate. Xcode Accounts preferences displays the type of certificate in the Signing Identities column. Keychain Access and the Code Signing Identity build setting pop-up menu in Xcode display the name of the certificate.

There's one Mac or iOS development certificate per team member. Therefore, development certificate names contain the person's name. All other types of certificates are owned by the team (shared by multiple team members) and so contain the team name. Individual developers are a one-person team, and so your name and the team name are the same.

Table 14-2 Certificate types and names

Certificate type	Certificate name	Description
iOS Development	iPhone Developer: *Team Member Name*	Used to run an iOS, tvOS, or watchOS app on devices and use certain app services during development.
iOS Distribution	iPhone Distribution: *Team Name*	Used to distribute your iOS, tvOS, or watchOS app on designated devices for testing or to submit it to the store.
Mac Development	Mac Developer: *Team Member Name*	Used to enable certain app services for a Mac app during development and testing.
Mac App Distribution	3rd Party Mac Developer Application: *Team Name*	Used to sign a Mac app before submitting it to the Mac App Store.
Mac Installer Distribution	3rd Party Mac Developer Installer: *Team Name*	Used to sign and submit a Mac Installer Package, containing your signed app, to the Mac App Store.
Developer ID Application	Developer ID Application: *Team Name*	Used to sign a Mac app before distributing it outside the Mac App Store.
Developer ID Installer	Developer ID Installer: *Team Name*	Used to sign and distribute a Mac Installer Package, containing your signed app, outside the Mac App Store.

Other types of certificates appear in your developer account that are not covered in this document.

- For details on Pass Type ID certificates, read Interacting with Passes in Your App in *Wallet Developer Guide*.
- For details on Website Push ID certificates, read *Notification Programming Guide for Websites*.
- For details on VoIP Services certificates, read Voice Over IP (VoIP) Best Practices in *Energy Efficiency Guide for iOS Apps*.
- For details on Apple Pay certificates, read Configuring Your Environment in *Apple Pay Programming Guide*.

Recap

In this chapter, you learned how to maintain your development and distribution signing identities that you use throughout the lifetime of your app. You also learned how to identify the different types of certificates in Xcode, Keychain Access, and your developer account.