# The Care and Feeding of Developer ID

This thread has been locked by a moderator.

112

I regularly see folks run into problems with their Developer ID signing identities. Historically I pointed them to my posts on this thread, but I've decided to collect these ideas together in one place.

If you have questions or comments, start a new thread here on DevForums and tag it with *Developer ID* so that I see it.

**IMPORTANT** Nothing I write here on DevForums is considered official documentation. It's just my personal ramblings based on hard-won experience. There is a bunch of official documentation that covers the topics I touch on here, including:

- Xcode documentation
- Xcode Help
- Developer Account Help
- Developer > Support > Certificates

For a *lot* more information about code signing, see the Code Signing Resources pinned post.

Share and Enjoy

—

Quinn "The Eskimo!" @ Developer Technical Support @ Apple

```
let myEmail = "eskimo" + "1" + "@" + "apple.com"
```

## The Care and Feeding of Developer ID

Most Apple signing assets are replaceable. For example, if you accidentally lose access to your Apple Development signing identity, it's a minor inconvenience. Just use the Developer website to revoke your previous certificate and create a replacement. Or have Xcode do that for you.

**IMPORTANT** If you don't understand the difference between a certificate and a digital identity, and hence *signing identity*, read Certificate Signing Requests Explained before reading this post.

Some signing assets are *precious*. Losing access to such assets has significant consequences.

Foremost amongst those are Developer ID signing identities. These allow you to sign Mac products that ship independently. Anyone with access to your Developer ID signing identity can sign code as *you*. This has a number of consequences, both for you and for your relationship with Apple.

## Identify a Developer ID Signing Identity

A Developer ID signing identity consists of two parts: the certificate and the private key. There are two different flavours, identifiable by the subject name in the certificate:

- **Developer ID Application** — This is named `Developer ID Application: TTT`, where `TTT` identifies your team. Use this to sign code and disk images.
- **Developer ID Installer** — This is named `Developer ID Installer: TTT`, where `TTT` identifies your team. Use this to sign installer packages.

**Note** If you do KEXT development, there's a third flavour, namely a KEXT-enabled Developer ID Application signing identity. For more details, see KEXT Code Signing Problems.

This post focuses on traditional signing identities, where *you* manage the private key. Xcode Cloud introduced cloud signing, where signing identities are "stored securely in the cloud". These identities have the *Managed* suffix in Certificates, Identifiers & Profiles. For example, Developer ID Application Managed is the cloud signing equivalent of Developer ID Application. To learn more about cloud signing, watch WWDC 2021 Session 10204 Distribute apps in Xcode with cloud signing. To identify these certificates 'in the wild', see Identifying a Cloud Managed Signing Certificate.

## Limit Access to Developer ID

Anyone with your Developer ID signing identity can sign code as *you*. Given that, be careful to limit access to these signing identities. This is true both for large organisations and small developers.

In a large organisation, ensure that only folks authorised to ship code on behalf of your organisation have access to your Developer ID signing identities. Most organisations have some sort of release process that they use to build, test, and authorise a release. This often involves a continuous integration (CI) system. Restrict CI access to only those folks involved in the release process.

Even if you're a small developer with no formal release process, you can still take steps to restrict access to Developer ID signing identities. See *Don't Leak Your Private Key*, below.

In all cases, don't use your Developer ID signing identities for day-to-day development. That's what Apple Development signing identities are for.

## Create Developer ID Signing Identities as the Account Holder

Because Developer ID signing identities are precious, the Developer website will only let the Account Holder create them. For instructions on how to do this, see Developer Account Help > Create certificates > Create Developer ID certificates. For more information about programme roles, see Developer > Support > Program Roles.

**IMPORTANT** In an Organization team it's common for the Account Holder to be non-technical. They may need help getting this done. For hints and tips on how to avoid problems while doing this, see *Don't Lose Your Private Key* and *Don't Leak Your Private Key*, both below.

## Limit the Number of Developer ID Signing Identities You Create

Don't create Developer ID signing identities unnecessarily. Most folks only need to create one. Well, one Developer ID Application and maybe one Developer ID Installer. A large organisation might need more, perhaps one for each sub-unit, but that's it.

There are two reasons why this is important:

- The more you have, the more likely it is for one to get into the wrong hands. Remember that anyone with your Developer ID signing identity can sign code as you.
- The Developer website limits you to 5 Developer ID certificates.

**Note** I can never remember where this limit is actually documented, so here's the exact quote from this page:

*You can create up to five Developer ID Application certificates and up to five Developer ID Installer certificates using either your developer account or Xcode.*

## Don't Lose Your Private Key

There are two standard processes for creating a Developer ID signing identity:

- Developer website — See Developer Account Help > Create certificates > Create Developer ID certificates.
- Xcode — See Xcode Help > Maintaining signing assets > Manage signing certificates.

Both processes implicitly create a private key in your login keychain. This makes it easy to lose your private key. For example:

- If you do this on one Mac and then get a new Mac, you might forget to move the private key to the new Mac.
- If you're helping your Organization team's Account Holder to create a Developer ID signing identity, you might forget to export the private key from their login keychain.

It also makes it easy to accidentally leave a copy of the private key on a machine that doesn't need it; see *Don't Leak Your Private Key*, below, for specific advice on that front.

Every time you create a Developer ID signing identity, it's a good idea to make an independent backup of it. For advice on how to do that, see *Back Up Your Signing Identities*, below.

That technique is also useful if you need to copy the signing identity to a continuous integration system.

If you think you've lost the private key for a Developer ID signing identity, do a proper search for it. You might be able to find it on your old Mac, in a backup, in a backup for your old Mac, and so on. If you can find the private key, it'll save you a bunch of grief.

If you're absolutely sure that you previous private key is lost, use the Developer website to create a replacement signing identity.

If the Developer website won't let you create any more because you've hit the limit discussed above, talk to Developer Programs Support. Go to Apple > Developer > Contact Us and follow the path Development and Technical > Certificates, Identifiers, and Provisioning Profiles.

## Don't Leak Your Private Key

Anyone with your Developer ID signing identity can sign code as you. Thus, it's important to take steps to prevent its private key from leaking.

A critical first step is to limit access to your Developer ID signing identities. For advice on that front, see *Limit Access to Developer ID*, above.

In an Organization team, only the Account Holder can create Developer ID signing identities. When they do this, a copy of the identity's private key will most likely end up in their login keychain. Once you've exported the signing identity, and confirmed that everything is working, make sure to *delete* that copy of the private key.

Some organisations have specific rules for managing Developer ID signing identities. For example, an organisation might require that the private key be stored in a hardware token, which prevents it from being exported. Setting that up is a bit tricky, but it offers important security benefits.

Even without a hardware token, there are steps you can take to protect your Developer ID signing identity. For example, you might put it in a separate keychain, one with a different password and locking policy than your login keychain. That way signing code for distribution will prompt you to unlock the keychain, which reminds you that this is a significant event and ensures that you don't do it accidentally.

If you believe that your private key has been compromised, follow the instructions in the *Compromised Certificates* section of Developer > Support > Certificates.

## Back Up Your Signing Identities

Given that Developer ID signing identities are precious, consider making an independent backup of them. To back up a signing identity to a PKCS#12 (`.p12`) file:

1. Launch Keychain Access.
2. At the top, select My Certificates.
3. On the left, select the keychain you use for signing identities. For most folks this is the login keychain.
4. Select the identity.
5. Choose File > Export Items.
6. In the file dialog, select Personal Information Exchange (`.p12`) in the File Format popup.
7. Enter a name, navigate to your preferred location, and click Save.
8. You might be prompted to enter the keychain password. If so, do that and click OK.
9. You will be prompted to enter a password to protect the identity. Use a strong password and save this securely in a password manager, corporate password store, on a piece of paper in a safe, or whatever.
10. You might be prompted to enter the keychain password again. If so, do that and click Allow.
11. The end result is a `.p12` file holding your signing identity. Save that file in a secure location, and make sure that you have a way to connect it to the password you saved in step 9.

Remember to backup all your Developer ID signing identities, including the Developer ID Installer one if you created it.

To restore a signing identity from a backup:

1. Launch Keychain Access.
2. Choose File > Import Items.
3. In the open sheet, click Show Options.
4. Use the Destination Keychain popup to select the target keychain.
5. Navigate to and select the `.p12` file, and then click Open.
6. Enter the `.p12` file's password and click OK.
7. If prompted, enter the destination keychain password and click OK.

Alternatively, Xcode has a feature to export and import your developer account, including your Developer ID signing identities. Do this using the action menu in Xcode > Settings > Accounts. For the details, see Xcode Help > Maintaining signing assets > Export signing certificates and provisioning profiles.

## Revision History

- **2023-06-23** Added a link to Identifying a Cloud Managed Signing Certificate.
- **2023-06-21** First posted.

Developer ID

Reply

Posted 2 weeks ago by eskimo

Add a Comment