Fixing an untrusted code signing certificate

This thread has been locked by a moderator.

150

This post is a ‘child’ of Resolving errSecInternalComponent errors during code signing. If you found your way here directly, I recommend that you start at the top.

Share and Enjoy
—
Quinn “The Eskimo!” @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

Fixing an untrusted code signing certificate

If your code signing identity is set up correctly, selecting its certificate in Keychain Access should display a green checkmark with the text “This certificate is valid”. If it does not, you need to fix that before trying to sign code. There are three common causes of an untrusted certificate:

- Expired
- Missing issuer
- Trust settings overrides

Check for an expired certificate

If your code signing identity’s certificate has expired, Keychain Access shows a red cross with the text “… certificate is expired”. If you try to sign with it, `codesign` will fail like so:

```
% codesign -s "Apple Development" -f "MyTrue"
error: The specified item could not be found in the keychain.
```

If you use `security` to list your code signing identities, it will show the `CSSMERR_TP_CERT_EXPIRED` status:

```
% security find-identity -p codesigning

Policy: Code Signing
  Matching identities
  1) 4E587951B705280CBB8086325CD134D4CDA04977 "Apple Development: …" (CSSMERR_TP_CERT_EXPIRED)
     1 identities found

  Valid identities only
     0 valid identities found
```

The most likely cause of this problem is that… yep… your certificate has expired. To confirm that, select the certificate in Keychain Access and look at the Expires field. Or double click the certificate, expand the Details section, and look at the Not Valid Before and Not Valid After fields.

If your code signing identity’s certificate has expired, you’ll need to renew it. For information on how to do that, see Developer Account Help.

If your certificate hasn’t expired, check that your Mac’s clock is set correctly.

Check for a missing issuer

In the X.509 public key infrastructure (PKI), every certificate has an issuer, who signed the certificate with their private key. These issuers form a chain of trust from the certificate to a trusted anchor. In most cases the trusted anchor is a root certificate, a certificate that’s self signed. Certificates between the leaf and the root are known as intermediate certificates, or intermediates for short.

Your code signing identity’s certificate is issued by Apple. The exact chain of trust depends on the type of certificate and the date that it was issued. For example, in 2022 Apple Development certificates are issued by the Apple Worldwide Developer Relations Certification Authority — G3 intermediate, which in turn was issued by the Apple Root CA certificate authority.

If there’s a missing issuer in the chain of trust between your code signing identity’s certificate and a trusted anchor, Keychain Access shows a red cross with the text “… certificate is not trusted”. If you try to sign with it, `codesign` will fail like so:

```
% codesign -s "Apple Development" -f "MyTrue"
MyTrue: replacing existing signature
Warning: unable to build chain to self-signed root for signer "Apple Development: …"
MyTrue: errSecInternalComponent
```

The message `unable to build chain to self-signed root for signer` is key.

If you use `security` to list your identities, it will not show up in the `Valid identities only` list but there’s no explanation as to why:

```
% security find-identity -p codesigning

Policy: Code Signing
  Matching identities
  1) 4E587951B705280CBB8086325CD134D4CDA04977 "Apple Development: …"
     1 identities found

  Valid identities only
     0 valid identities found
```

IMPORTANT These symptoms can have multiple potential causes. The most common cause is a missing issuer, as discussed in this section. Another potential cause is a trust settings override, as discussed in the next section.

There are steps you can take to investigate this further but, because this problem is most commonly caused by a missing intermediate, try taking a shortcut by assuming that’s the problem. If that fixes things, you’re all set. If not, you have at least ruled out this problem.

Apple publishes its intermediates on the Apple PKI page. The simplest way to resolve this problem is to download all of the certificates in the Apple Intermediate Certificates list and use Keychain Access to add them to your keychain. Having extra intermediates installed is generally not a problem.

If you want to apply a more targeted fix:

1. In Keychain Access, find your code signing identity’s certificate and double click it.
2. If the Details section is collapsed, expand it.
3. Look at the Issuer Name section. Note the value in the Common Name field and, if present, the Organizational Unit field. For example, for an Apple Development certificate that’s likely to be *Apple Worldwide Developer Relations Certification Authority* and *G3*, respectively.
4. Go to the Apple PKI and download the corresponding intermediate. To continue the above example, the right intermediate is labelled *Worldwide Developer Relations - G3*.
5. Use Keychain Access to add the intermediate to your keychain.

Sometimes it’s not obvious which intermediate to choose in step 4. If you’re uncertain, download all the intermediates and preview each one using Quick Look in the Finder. Look in the Subject Name section for a certificate whose Common Name and Organizational Unit field matches the values from step 3.

Finally, double check the chain of trust:

1. In Keychain Access, select your code signing identity’s certificate and choose Keychain Access > Certificate Assistant > Evaluate.
2. In the resulting Certificate Assistant window, make sure that Generic (certificate chain validation only) is selected and click Continue.
   It might seem like selecting Code Signing here would make more sense. If you do that, however, things don’t work as you might expect. Specifically, in this case Certificate Assistant is smart enough to temporarily download a missing intermediate certificate in order to resolve the chain of trust, and that’ll prevent you from seeing any problems with your chain of trust.
3. The resulting UI shows a list of certificates that form the chain of trust. The first item is your code signing identity’s certificate and the last is an Apple root certificate. Double click the first item.
4. Keychain Access presents the standard the certificate trust sheet, showing the chain of trust from the root to the leaf. You should expect to see three items in that list:
   - An Apple root certificate
   - An Apple intermediate
   - Your code signing identity’s certificate

   If so, that’s your chain of trust built correctly.
5. Select each certificate in that list. The UI should show a green checkmark with the text “This certificate is valid”. If you see anything else, check your trust settings as described in the next section.

Check for a trust settings override

macOS allows you to customise trust settings. For example, you might tell the system to trust a particular certificate when verifying a signed email but not when connecting to a TLS server.

The code signing certificates issued by Apple are trusted by default. They do not require you to customise any trust settings. Moreover, customising trust settings might cause problems.

If code signing fails with the message `unable to build chain to self-signed root for signer`, first determine the chain of trust per the previous section then make sure that none of these certificates have customised trust settings. Specifically, for each certificate in the chain:

1. Find the certificate in Keychain Access.
   Note that there may be multiple instances of the certificate in different keychains. If that’s the case, follow these steps for each copy of the certificate.
2. Double click the certificate to open it in a window.
3. If the Trust section is collapsed, expand it.
4. Ensure that all the popups are set to their default values (Use System Defaults for the first, “no value specified” for the rest).
5. If they are, move on to the next certificate.
6. If not, set the popups to the default values and close the window. Closing the window may require authentication to save the trust settings.

Code Signing

Reply

Posted 2 months ago by eskimo

Add a Comment