Article

About the Rosetta Translation Environment

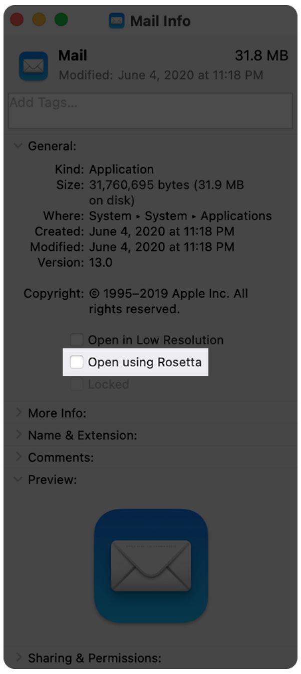
Learn how Rosetta translates executables, and understand what Rosetta can't translate.

Overview

Rosetta is a translation process that allows users to run apps that contain `x86_64` instructions on Apple silicon. Rosetta is meant to ease the transition to Apple silicon, giving you time to create a universal binary for your app. It is not a substitute for creating a native version of your app.

To the user, Rosetta is mostly transparent. If an executable contains only Intel instructions, macOS automatically launches Rosetta and begins the translation process. When translation finishes, the system launches the translated executable in place of the original. However, the translation process takes time, so users might perceive that translated apps launch or run more slowly at times.

The system prefers to execute an app's `arm64` instructions on Apple silicon. If a binary includes both `arm64` and `x86_64` instructions, the user can tell the system to launch the app using Rosetta translation from the app's Get Info window in the Finder. For example, a user might enable Rosetta translation to allow the app to run older plug-ins that don't yet support the `arm64` architecture.



Mail 31.8 MB
Modified: June 4, 2020 at 11:18 PM

General:
Kind: Application
Size: 31,760,695 bytes (31.9 MB on disk)
Where: System ▸ System ▸ Applications
Created: June 4, 2020 at 11:18 PM
Modified: June 4, 2020 at 11:18 PM
Version: 13.0

Copyright: © 1995–2019 Apple Inc. All rights reserved.

☐ Open in Low Resolution
☐ Open using Rosetta
☐ Locked

Important

The system prevents you from mixing `arm64` code and `x86_64` code in the same process. Rosetta translation applies to an entire process, including all code modules that the process loads dynamically.

For information on how to determine when your app is running under Rosetta translation, see

Determine Whether Your App Is Running as a Translated Binary.

What Can't Be Translated?

Rosetta can translate most Intel-based apps, including apps that contain just-in-time (JIT) compilers. However, Rosetta doesn't translate the following executables:

- Kernel extensions
- Virtual Machine apps that virtualize x86_64 computer platforms

Rosetta translates all `x86_64` instructions, but it doesn't support the execution of some newer instruction sets and processor features, such as AVX, AVX2, and AVX512 vector instructions. If you include these newer instructions in your code, execute them only after verifying that they are available. For example, to determine if AVX512 vector instructions are available, use the `sysctlbyname` function to check the `hw.optional.avx512f` attribute.

Determine Whether Your App Is Running as a Translated Binary

On Apple silicon, a universal binary may run either natively or as a translated binary. The system runs the native version whenever possible, but the user might opt to run the code using Rosetta to support older plug-ins.

To programmatically determine when a process is running under Rosetta translation, call the `sysctlbyname` function with the `sysctl.proc_translated` flag, as shown in the following example. The example function returns the value `0` for a native process, `1` for a translated process, and `-1` when an error occurs.

```
int processIsTranslated() {
   int ret = 0;
   size_t size = sizeof(ret);
   if (sysctlbyname("sysctl.proc_translated", &ret, &size, NULL, 0) == -1)
   {
      if (errno == ENOENT)
         return 0;
      return -1;
   }
   return ret;
}
```