Add iOS devices

Before you start

- Follow this guide to prepare the device.
- Find the UDID of the device and note it down.
- Make sure the Mac mini host has the Xcode version that is compatible with the iOS/iPadOS version of the device.

Import developer certificates and provisioning profiles

Make sure you have already export the appropriate developer certificates and provisioning profiles for the UDID of the device.

Import developer certificates

⚠ These steps require accessing the Mac mini host's screen and cannot be done via SSH.

Open **Terminal** on the Mac mini host and enter the following command:

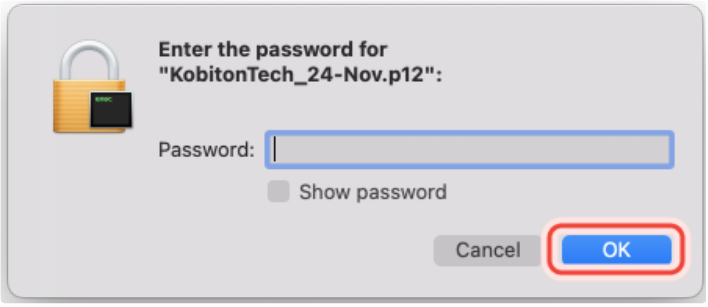
```
sudo security add-certificates -k /Library/Keychains/System.keychain <Path to the cert>/AppleWWDRCAG3.cer
```

Replace *<Path to the cert>* with the full path to the folder containing the Apple WWDR Intermediate certificate file. Enter the administration password if required.

Next, enter the following command, replacing *<Path to the cert>* with the full path to the *.p12* certificate file and *<Name of the cert>* with the filename.

```
sudo security import <Path to the cert>/<Name of the cert>.p12 -k /Library/Keychains/System.keychain -A
```

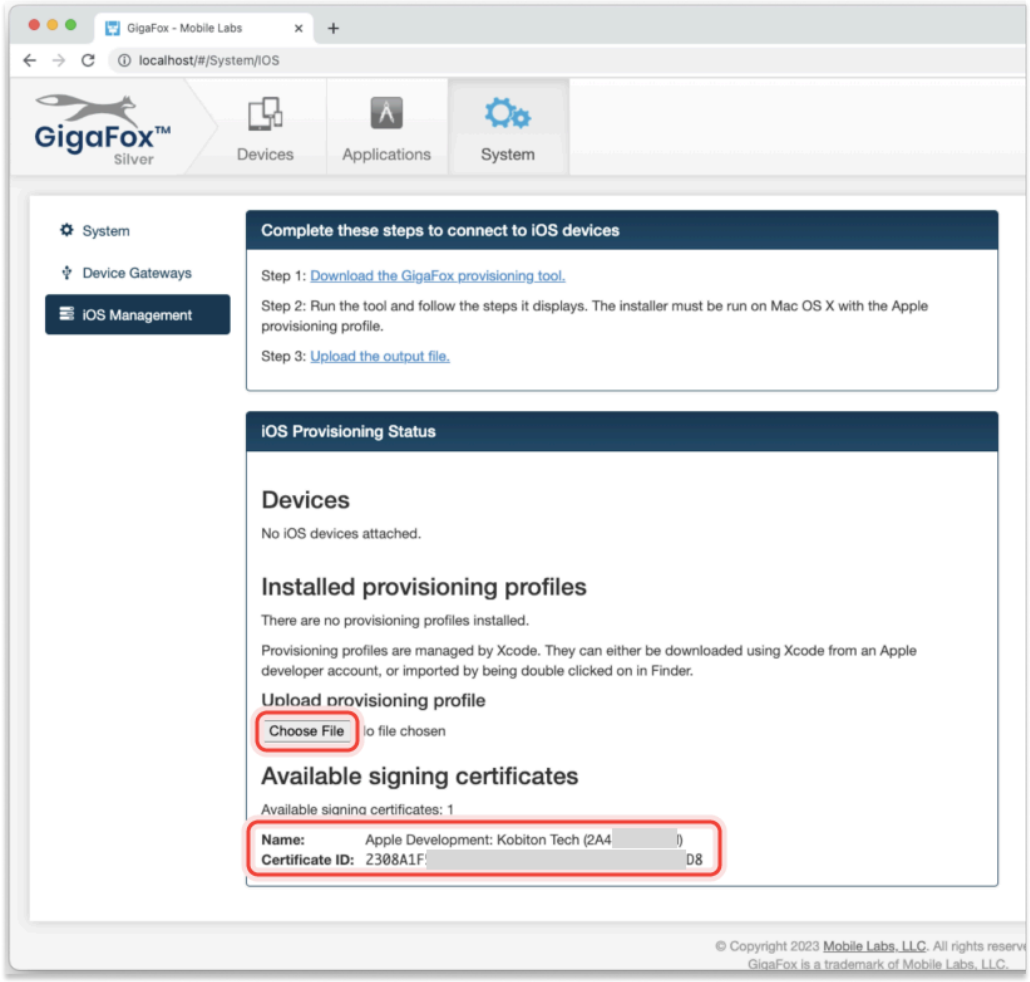
Enter the certificate password. If the certificate has no password, leave the field blank and click **OK**.



Repeat the above commands for each *.p12* file if there are multiple files.

Import provisioning profiles to deviceConnect

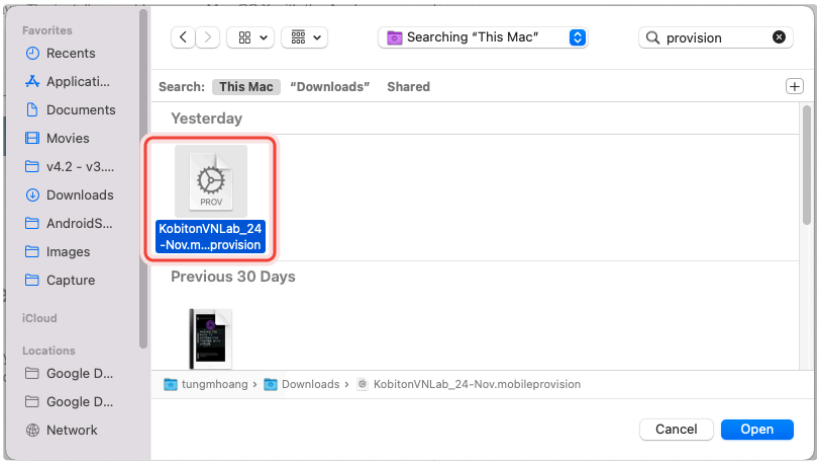
Open the Chrome browser, go to the address: *localhost/#/System/IOS* and log in.



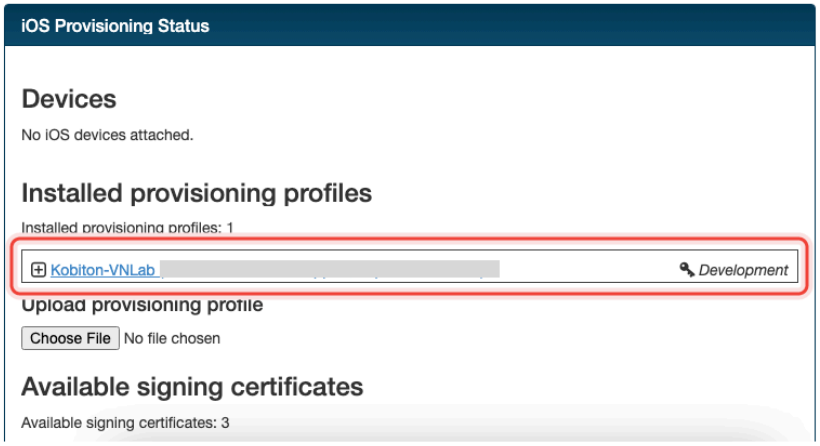
Under **Available signing certificates**, you can see all imported certificates from the above step.

Click **Choose File** under **Upload provisioning profile**

Select a *.mobileprovision* file, and click **Open** to upload it.



The uploaded profile should display under **Installed provisioning profiles**:



Important: Restart deviceConnect services to apply the new provisioning profiles.

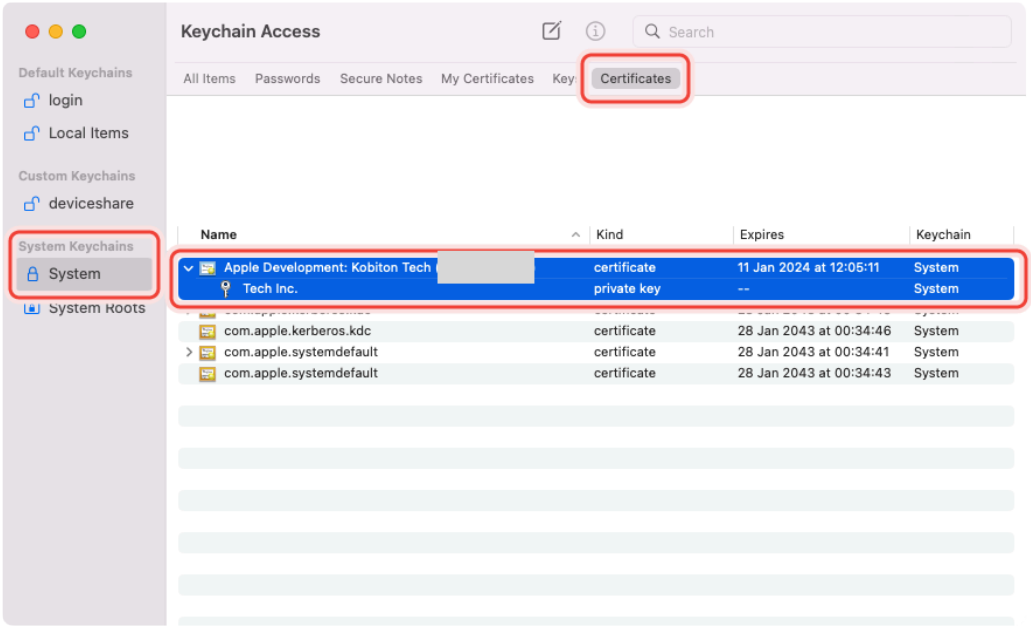
Import developer certificates and provisioning profiles to deviceShare

Skip this section if you do not use Kobiton app re-signing service.

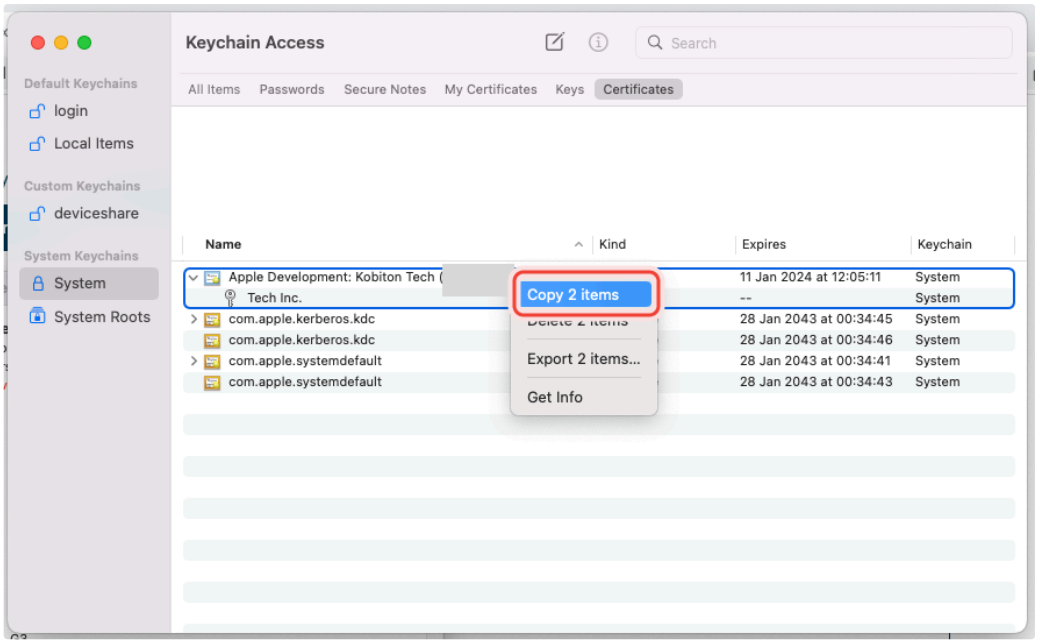
Before importing, if your deployment include multiple Mac mini hosts, make sure the Mac mini host has deviceShare installed by going to its installation folder and check the version. Only proceed if deviceShare is installed.

Open the **Keychain Access** app.

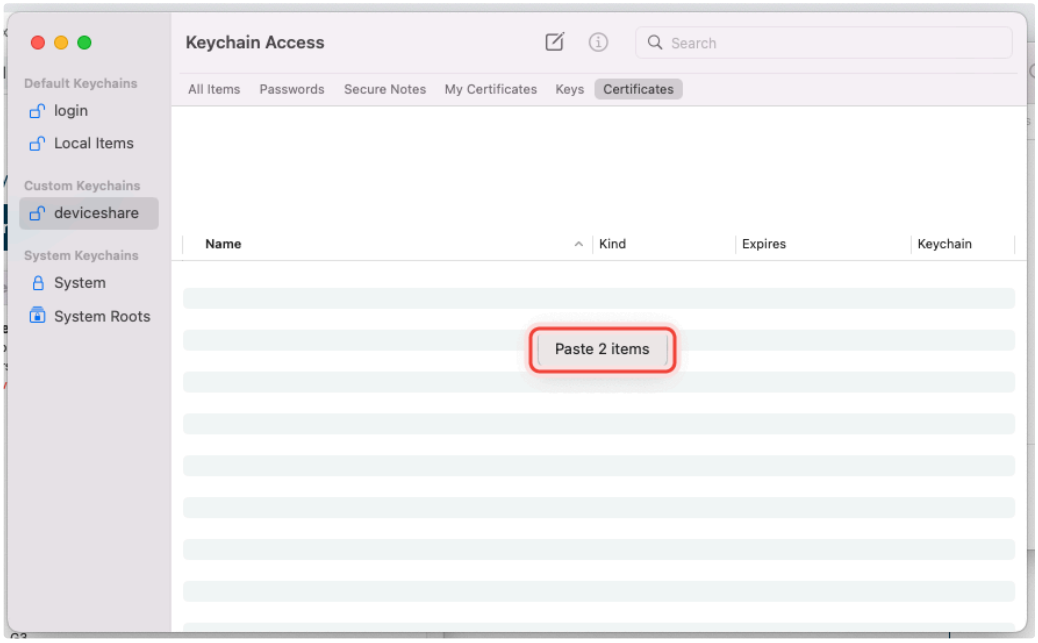
Select the **System** keychain, and then **Certificates**. You will see your **Apple Development** signing certificates along with all the other certificates. Expand all the **Apple Development** signing certificates to show the private key like the below:



Shift-click to select all the **Apple Development** certificates and their private key, then right-click and select **Copy items**.



Select the **deviceshare** keychain and then **Certificates**. Right-click the empty area and choose **Paste items**. You will be prompted to enter your login keychain password and the password for the *deviceshare* keychain for each certificate imported.



Verify that the certificates and keys are imported successfully into the *deviceshare* keychain.

Open the *deviceshare_config.toml* file located under */usr/local/kobiton/deviceshare/*.

Locate the line starting with `ios_provisioning_profile_paths`.

If the line is the same as below, skip this section as deviceShare is using the same folder with deviceConnect for provisioning profiles:

```
ios_provisioning_profile_paths = [
    "/usr/local/deviceconnect/ProvisioningProfiles"
]
```

If the line is the same as below instead, continue on the next step:

```
ios_provisioning_profile_paths = [
```

```
    "/usr/local/kobiton/deviceshare/provisioning_profiles"
]
```

Move all provisioning profile files into one folder and note down the location.

Open Terminal and execute the below command, where */path/to/profiles/* is the location of all the provisioning profile files:

```
cp -R /path/to/profiles/*.mobileprovision /usr/local/kobiton/deviceshare/provisioning_profiles
```

Restart deviceShare signing service to apply all the configurations above by running this command:

```
sudo /bin/launchctl unload -w /Library/LaunchDaemons/com.kobiton.deviceshare.signing.plist && sleep 5 && sudo /bi
```

Verify that the deviceShare signing service is running normally by executing the below command:

```
tail -100 /usr/local/kobiton/deviceshare/deviceshare_signing.log
```

A successful execution should show the output as below:

```
2022-02-24 23:23:20.873521 INFO  [deviceshare::logging] initialized log config from /usr/local/kobiton/devicesha
2022-02-24 23:23:20.873612 INFO  [deviceshare::signing::signingserver] attempting to connect to Kobiton signing
2022-02-24 23:23:20.873630 INFO  [deviceshare::signing::signingserver] authentication not enabled for Kobiton si
2022-02-24 23:23:20.873653 INFO  [deviceshare::signing::signingserver] attempting to connect to Kobiton signing
2022-02-24 23:23:20.873729 DEBUG [hyper::client::connect::http] connecting to 10.2.122.251:6000
2022-02-24 23:23:20.874310 DEBUG [hyper::client::connect::http] connected to 10.2.122.251:6000
2022-02-24 23:23:20.886689 INFO  [deviceshare::signing::signingserver] connected to Kobiton signing portal
.... truncated ...
2022-02-24 23:23:20.902941 DEBUG [deviceshare::signing::keychain] signing_certificates_all: elapsed: 0 ms
2022-02-24 23:23:20.905563 DEBUG [deviceshare::signing::signingserver] monitor_resource_changes: resources have
2022-02-24 23:24:20.927290 DEBUG [deviceshare::signing::signingserver] sending keepalive message
2022-02-24 23:24:20.943450 DEBUG [deviceshare::signing::signingserver] monitor_resource_changes: polling current
```

Connect Cambrionix hub to the host

> Skip this step if the Mac mini host or GEM already has a Cambrionix hub connected.

Make sure you use a supported model of Cambrionix hub.

Connect the Cambrionix hub to a power source. The power LED indicator of the Cambrionix hub should turn on.

For Standard mode, connect the Mac mini to the **host** port of the hub.

For Lightning mode, connect the Graphic Extension Manager (GEM) to the **host** port of the hub. Make sure you connect the Cambrionix hub to the blue USB 3.0 port on the GEM.

Refer to the hub model's user manual from Cambrionix for the exact host port location.

See below for an example of the SuperSync15 with the Host port visible.

Connect the device to the host

Make sure you have properly prepared the device for hosting on Kobiton.

For Standard mode, connect the mobile device to the Cambrionix hub attached to the Mac mini host.

For Lightning mode with iOS 16 and below, connect the mobile device to the Cambrionix hub attached to the GEM.

For Lightning mode with iOS 17 and above, connect the device to one of the USB ports of the Mac mini host to establish trust pairing first, then follow the next section before connecting the device to the Cambrionix hub attached to the GEM.

Check the device to see if it is charging after connecting. If it is not charging, the USB cable might be malfunctioning, or the Cambrionix hub is not connected to a power source.

Establish trust pairing between the device and the host

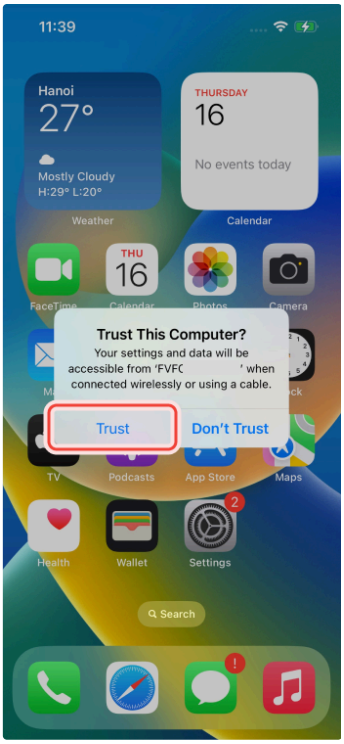
Access the Mac mini host directly or via screen sharing.

The steps to establish trust pairing vary between iOS 16 and below and iOS 17 and above.

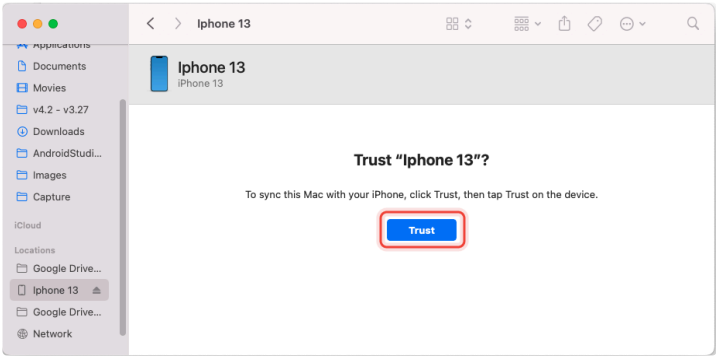
iOS 16 and below

> The steps in this section apply to both Standard and Lightning mode.

Check the device screen. Tap **Trust** on the *Trust this computer* pop-up:



Open **Finder** in the Mac mini host, select the connected device name, and choose **Trust**.



Unplug the device, then plug it in again. Wait until the device screen changes to the below before continuing (NOTE: there will also be an *automation running* overlay above the device screen):

iOS 17 and above

Note for air-gapped Mac mini hosts (no Internet access)

To control the iOS devices, deviceConnect needs to mount a *Developer Disk Image* (DDI), which is a `.dmg` archive included with Xcode that contains executables and other files needed by Xcode to support debugging and testing on iOS devices.

For iOS 17 and later, rather than Xcode providing a different DDI for every iOS version and device architecture, there is a generic DDI that Xcode must "personalize" for each device. The personalization process requires an Internet connection, as Xcode must use Apple's notarization servers to sign the personalized image. Without an Internet connection, Xcode can't personalize a DDI.

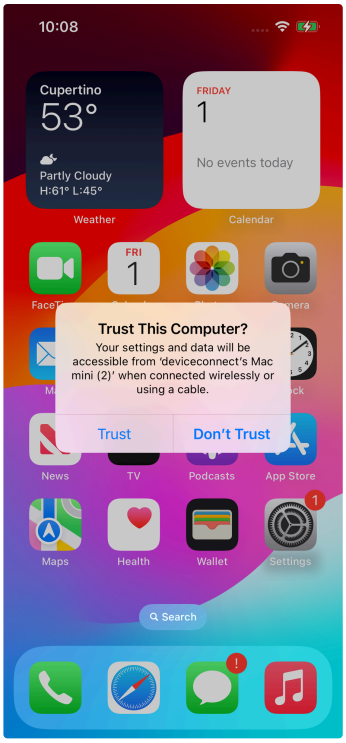
If the Mac mini host does not have Internet connection, follow the section pre-load DDI for air-gapped Mac mini before continuing with this section.

Follow the appropriate steps based on whether you are using Standard or Lightning mode

Standard mode:

Open Xcode on the Mac mini host, then navigate to **Window** → **Devices and Simulators**. Do this before continuing to the next step.

The *Trust this computer* prompt on the device screen appears, tap **Trust**.



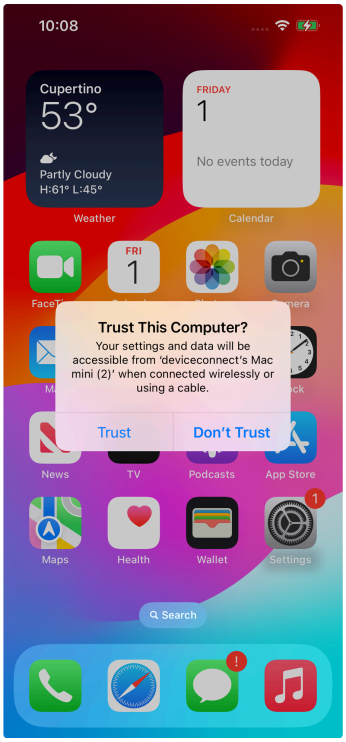
The *Trust this computer* prompts will reappear, tap **Trust** again. This time there should be no more **Trust** prompts.

Lightning mode:

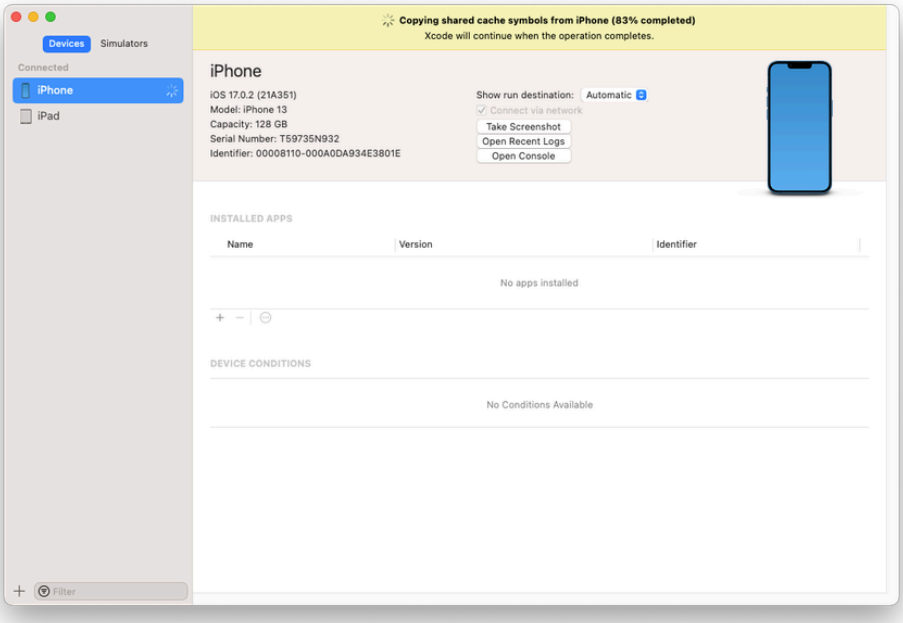
Open Xcode on the Mac mini host, then navigate to **Window** → **Devices and Simulators**. Do this before continuing to the next step.

Make sure you connect the device **to the Mac mini host** first.

The *Trust this computer* prompt on the device screen appears, tap **Trust**.



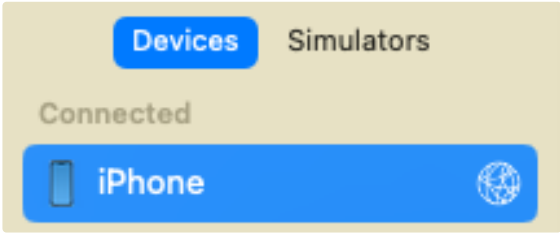
In the Mac mini host's screen, under the **Devices** tab of the **Devices and Simulators** screen, the iOS 17 devices should show up with a yellow warning message like the one below:



Unplug the device from the Mac mini host and plug it into the GEM.

The *Trust this computer* prompts will reappear, tap **Trust** again. This time there should be no more **Trust** prompts.

In Xcode's Devices and Simulators, the iOS 17 devices now display with a globe icon next to it like below:



Wait until the device screen changes to the below before continuing (NOTE: there will also be an *automation running* overlay above the device screen):

Pre-load DDI for air-gapped Mac mini hosts

> This section is only required for Mac mini hosts with no Internet access with iOS 17 and above devices.

Acquire any MacOS machine with Internet access. This will be referred to as the Internet Mac.

Ensure Xcode is installed on the Internet Mac. Make sure the Xcode version is compatible with the iOS 17 device. *Note:* Kobiton software such as deviceConnect and deviceShare do not need to be installed on the Internet Mac.

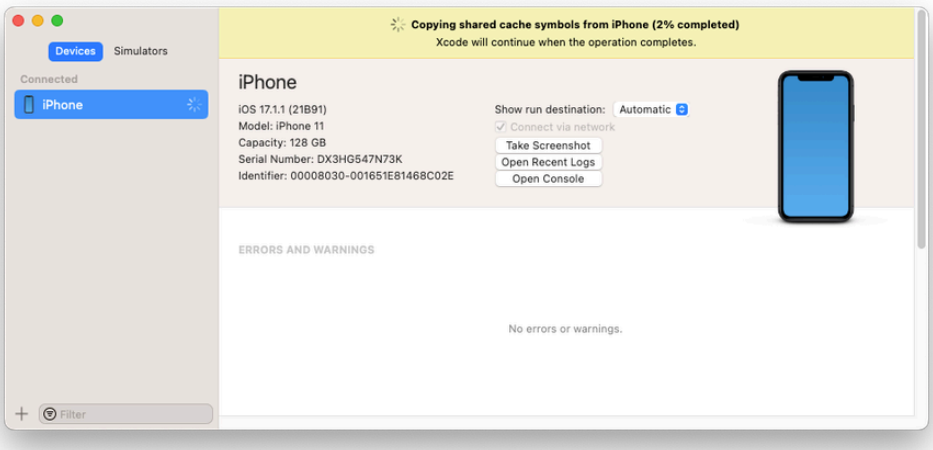
Unplug the iOS 17 device from the air-gapped Mac (Standard mode) or the GEM (Lightning mode) and connect it to the Internet Mac.

Open Xcode.

Tap Trust in the **Trust this computer** pop-up on the iOS 17 device. The **Trust this computer** prompts will reappear, tap **Trust** again. After this, there should be no more **Trust** prompts.

In the Xcode menu bar, select **Window** → **Devices and Simulators**. Select the iOS 17 device under the **Devices** tab.

The `Copying shared cache symbols...` message appears. Wait for this process to complete and the message to clear.



Unplug the device from the Internet Mac.

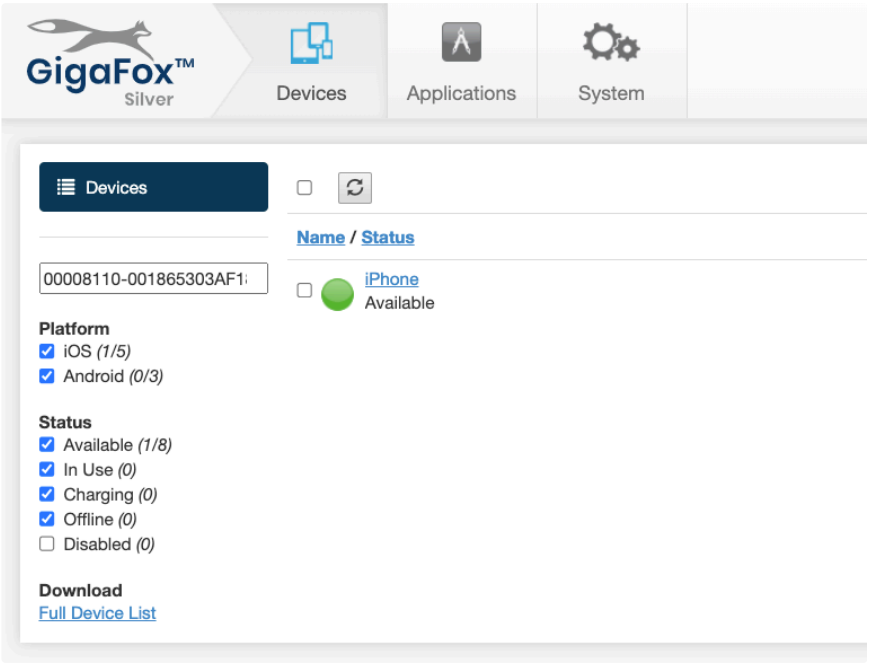
Continue with connecting iOS 17 and above devices to the air-gapped Mac mini hosts.

> ⚠ Apple has not published whether the personalized DDI will expire or how long it will last in an air-gapped environment. If connection errors occur and other troubleshooting steps do not resolve the issue, the personalized DDI may be expired and you will need to repeat this process.

Verify device is available in Kobiton

Open Chrome on the Mac mini, then open *localhost* and log in.

Navigate to **Devices**. The connected device displays as **Available**.



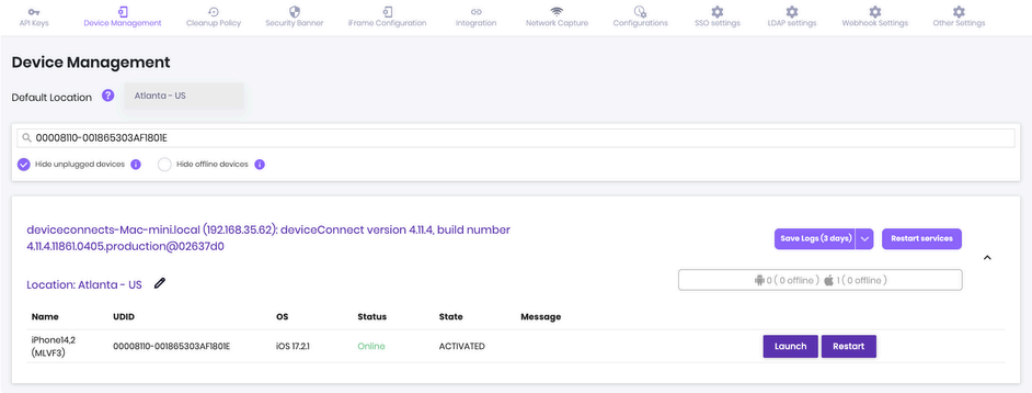
Still in Chrome, open the Kobiton web portal and log in using an account with ADMIN role.

Select the profile picture and choose **Settings**, then choose **Device Management**.

In the search bar, enter the device's UDID and select Enter to filter.

The device should appear in the filter result. If the state of the device is *Utilizing*, it is being cleaned up. Wait about 2-3 minutes for the cleanup to complete.

When the cleanup is done, the device state becomes *Online* and the **Launch** button is available. Select it to launch a Manual session on the device.



In the Manual session, try the following to verify if the device is working properly:

- Navigate around.
- Install an app.
- Browse the web (if the device has a Wi-Fi connection).
- Enable Lightning mode (if the device is configured for Lightning mode).

If all the above works, you have successfully added the device.