Visual Accessors for iOS New Feature Spotlight

AIQ 4.9 - Beta Release

Document Version 1.1

March 4, 2023

TOC

Visual Accessors - New Feature Spotlight

The ability to use visual accessors has been added to the Mobile Test Designer. Visual accessors allow Mobile Designer to incorporate images into your test scripts. Smart Tags Workbench and Mobile Blueprint have been enhanced so that images are defined as Smart Tags and can be added to a Mobile Blueprint. This new functionality is available for both Android and iOS mobile devices.

Mobile Designer enables script recording and playback, a simpler approach to mobile test design than traditional mobile scripting. Using visual accessors allows Mobile Designer to incorporate screenshots into scripts.

This documentation details how the image matching algorithm works, recommendations for getting the best results when using this feature, as well as information about setting up and configuring this new functionality.

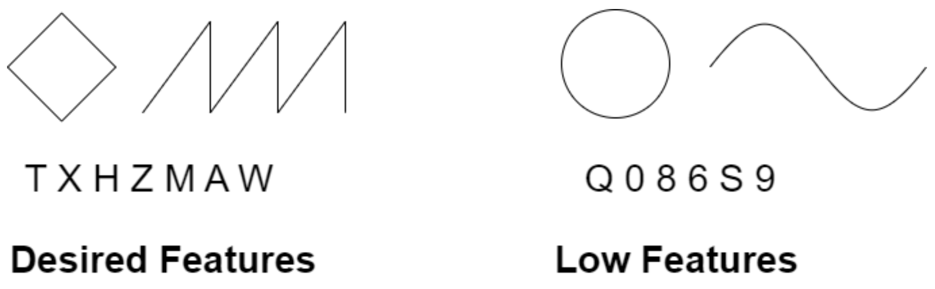
Visual accessors for mobile testing is new functionality that will be introduced in AIQ 4.9. This feature and this documentation are currently in Beta.

Understanding the Image Matching Algorithm

The following section details how the image-matching algorithm determines if the target and source images are a match. There are also some recommendations and considerations for getting the best results when using the visual accessor functionality.

How the Algorithm Works

Naturally, different images will have different numbers of distinguishing features. An example of this is shown below.



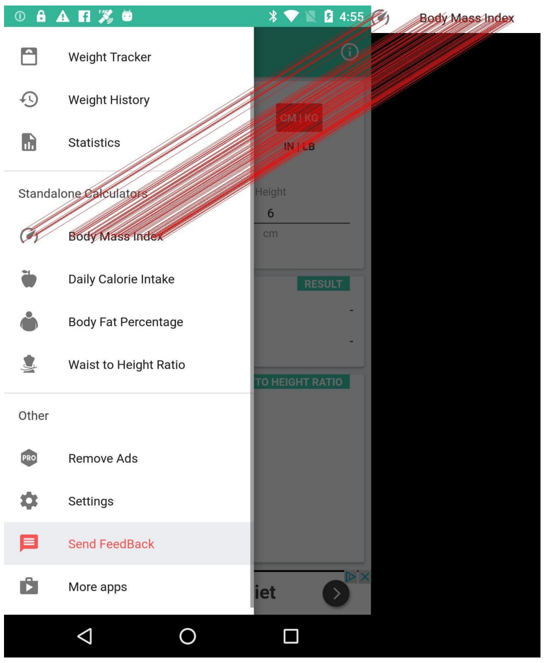
The images in the “Desired Features” set have a significant number of distinguishing features that set them apart from the other images in that set.

The images in the “Low Features” set have a smaller number of distinguishing features that set them apart from other images in that set.

Matching Process

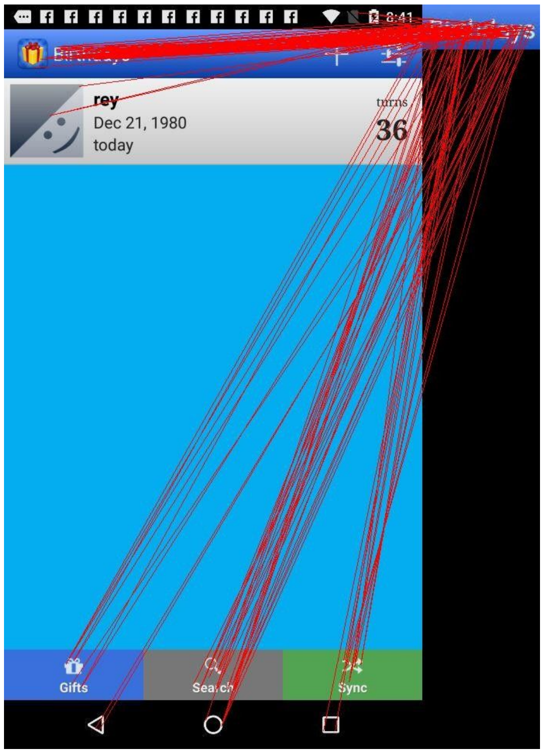
- The first thing the algorithm does is compute the distinguishing feature points for each image (target and source).

- Next, the algorithm applies a matcher to find similar features in both images. For the matcher, a possible set of different algorithms can be applied. In this case, it is a matcher that rejects outlier points that are chosen to reduce the probability of getting an incorrect match.



- Using the set of matched points, the algorithm constructs a region with the same dimensions as the target rectangle bounding.

- The similarity between the target and sub-region source is calculated with several parameters to determine if both are similar. If they are found to be similar, then the bounding box is returned.



Recommendations and Considerations

Here are some recommendations and considerations for getting the best results when using the visual accessor functionality.

- Multiple reference images for the same target. Along with the original-sized image, you should have several images from the same target with different sizes than the original size. Reference images should also include both images taken from the Mobile Designer IDE along with screen captures. Having reference images from both capture sources is recommended because some bit depth differences can affect results when the target image has limited distinguishing features

- Use images with less compression and better quality. The preferred image format is PNG, but formats such as JPG/JPEG will give good results. PNG is preferred because there is less information loss due to compression. Images with more detail will have a greater number of distinguishing features.

- Test your targets and possible sources. Before creating a full test run, test the set of targets and sources to validate if the correct targets are chosen. You can write a script to perform that task.

- Add context to images to help differentiate them from other possible matches. This can happen if the cropping area is too limited and the number of distinguishing features is limited.
  For example, an image with the word “ear” could have multiple matches depending on how the image is cropped. If the image is cropped exactly covering the word, the algorithm will also add “bear”, “spear”, “tear”, “boar” and more as matches. Even “ear” could have “oar” or “eor” as possible matches. So, when possible include some additional context in your image to help with exact matching.

Visual Accessors for iOS

The following sections contain information about setting up and configuring this new functionality.

- "iOS Environment Setup" on page 13
  - "Appium Server Setup for iOS" on page 14
  - CbI_iOS_Xcode_Main
  - "Install and Configure Appium WebDriverAgent" on page 18

The following sections contain an end-to-end example of using the visual accessor functionality with an example mobile app.

- "Visual Accessor Example for iOS" on page 23
  - "Creating the Configuration File" on page 24
  - "Configuring Access to a Repository" on page 28
  - "Taking Images for Smart Tags" on page 30
  - "Creating the Smart Tags File" on page 34

The following sections contain information on configuration steps that, depending on your existing environment, you may have to follow to use the visual accessor functionality.

This is the documentation for the visual accessors feature for testing iOS mobile applications. The documentation for testing Android mobile applications can be found on the Visual Accessors for Mobile Testing page.

iOS Environment Setup

Mobile Designer requires an Appium server and Xcode IDE to interact with apps.

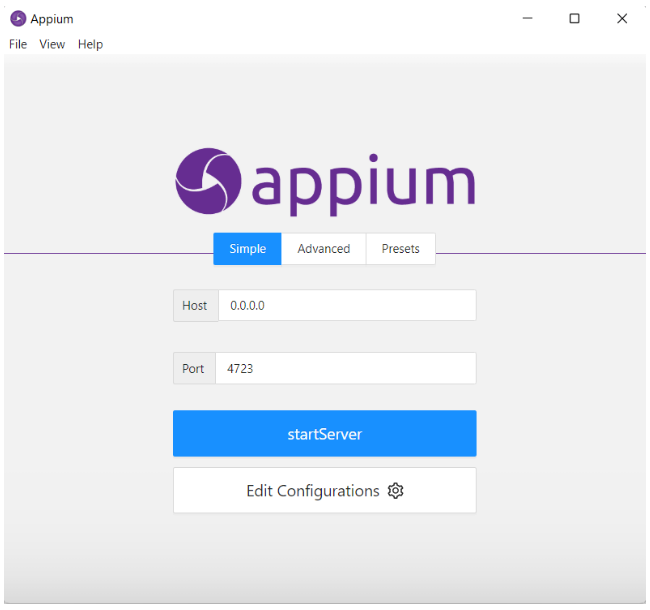
This guide is intended for iOS and iPadOS use only. There is a separate documentation for using visual accessors for Android apps.

Appium Server Setup for iOS

Appium is an open source test automation framework for use with native, hybrid and mobile web apps.

It drives iOS, Android, and Windows apps using the WebDriver protocol.

1. Download Appium Server from https://github.com/appium/appium-desktop/releases/tag/v1.22.3-4
2. Choose the MacOS installer.
3. Launch the installer.



Host 0.0.0.0

Port 4723

4. Follow the installation steps and select the default settings.

If you are running MacOS Ventura or newer versions please refer to "Troubleshooting Appium and MacOS Ventura" on page 59 in order to run Appium Server.

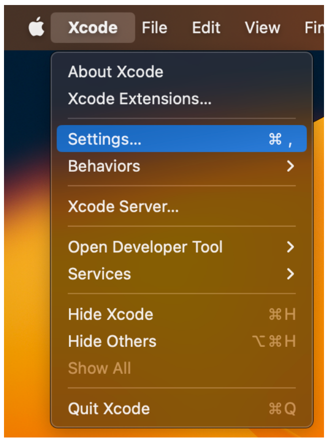
Setup Apple Developer account on Xcode

Follow these steps to setup an Apple Developer account on Xcode. Make sure you have the email and password associated with your account.

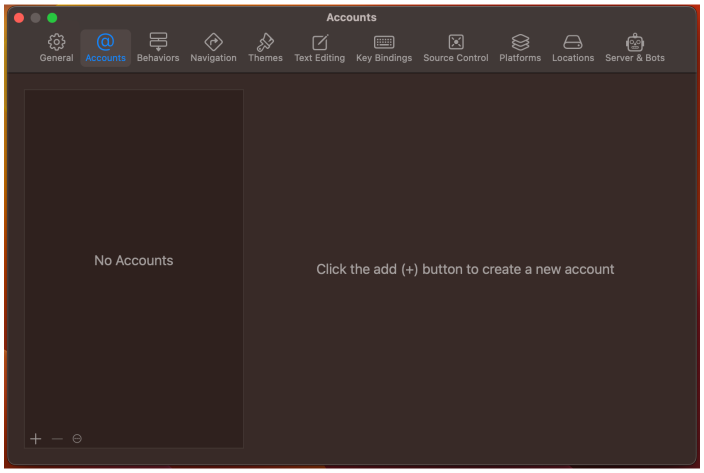
An Xcode account is required when signing a project. Signing is a process required by Apple when you want to create a build or run a project on a physical device. See "Signing Xcode project" on page 43 for more information.

If Two-Factor Authentication is enabled for the account, make sure you have access to at least one of the authentication methods. See Apple's information on two-factor authentication for more information.

1. Open **Xcode** and select **Settings**.



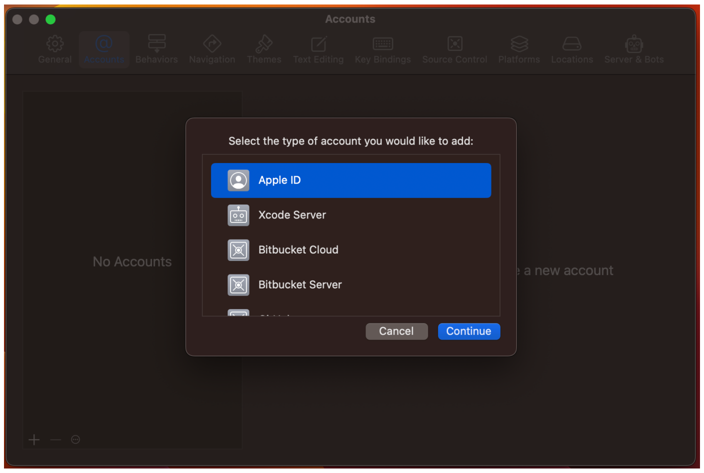
2. Navigate to the **Accounts** tab and click the add (+) button.



No Accounts

Click the add (+) button to create a new account

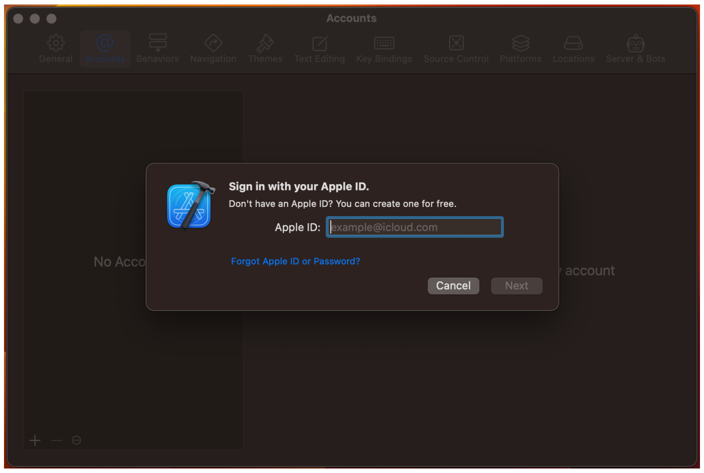
3. Select the **Apple ID** option and click **Continue**.



Select the type of account you would like to add:

- Apple ID
- Xcode Server
- Bitbucket Cloud
- Bitbucket Server

4. Enter the email associated with the **Developer Account** and click **Next**.



Sign in with your Apple ID.

Don't have an Apple ID? You can create one for free.

Apple ID: example@icloud.com

Forgot Apple ID or Password?

5. Type the password and click **Next**. Wait for the authentication process to finish.

6. If you have enabled two-factor authentication, enter the verification code. If necessary, follow any additional steps to get a verification code.

7. Select the **Developer Account** you just authenticated from the list of **Apple IDs**.

8. Click **Download Manual Profiles**.

.

Install and Configure Appium WebDriverAgent

WebDriverAgent is a WebDriver server implementation for iOS that can be used to remotely control iOS devices. It allows you to launch and kill applications, tap and scroll views or confirm view presence on a screen.

For more information see Appium's documentation on real device configuration at appium/appium-xcuitest-driver.

1. Open **Terminal**, and run the following command and copy the output path.

   ```
   echo "$(dirname "$(find "$HOME/.appium" -name
   WebDriverAgent.xcodeproj)")"
   ```

   An example of the command output: `/Users/USER/.appium/node_modules/appium-xcuitest-driver/node_modules/appium-web-driveragent`

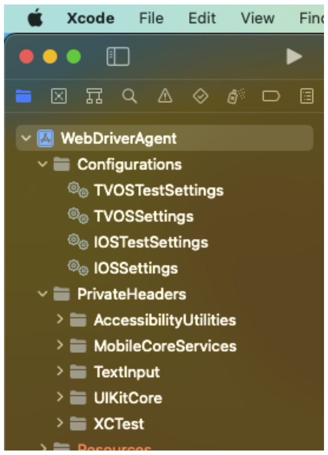
2. Navigate to the output path. The easiest way is running the **open** command on **Terminal** adding the path as parameter.

   For example:

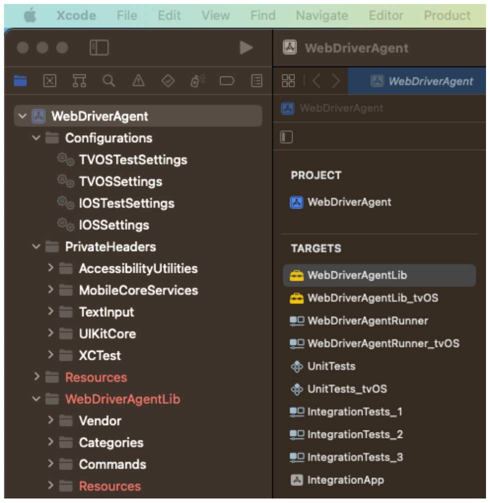
   ```
   open /Users/USER/.appium/node_modules/appium-xcuitest-
   driver/node_modules/appium-webdriveragent
   ```

3. Open the **WebDriverAgent.xcodeproj** on **Xcode**.

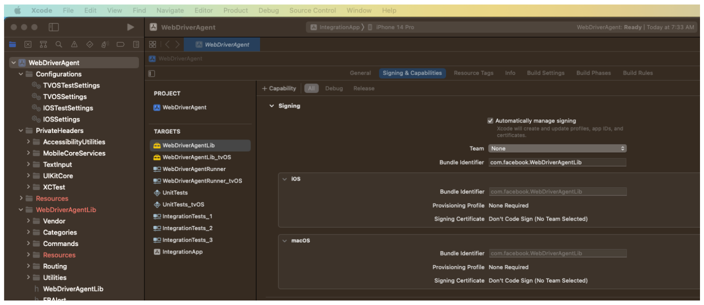
4. On the **Navigator** select **WebDriverAgent**.



5. Select **WebDriverAgentLib** from the list of targets.



6. Select the **Signing & Capabilities** tab.



WebDriverAgent: **Ready** | Today at 7:33 AM

Signing

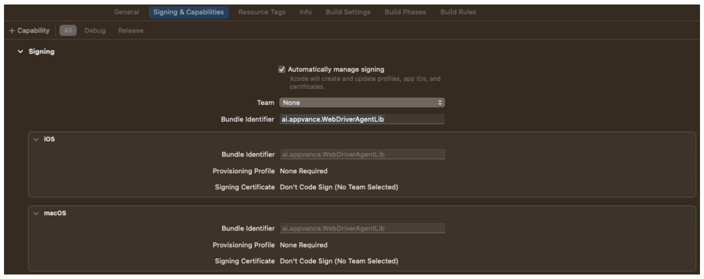
[✓] Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team: None
Bundle Identifier: com.facebook.WebDriverAgentLib

iOS
Bundle Identifier: com.facebook.WebDriverAgentLib
Provisioning Profile: None Required
Signing Certificate: Don't Code Sign (No Team Selected)

macOS
Bundle Identifier: com.facebook.WebDriverAgentLib
Provisioning Profile: None Required
Signing Certificate: Don't Code Sign (No Team Selected)

7. Edit the Bundle Identifier value to replace the **com.facebook** prefix. You can replace it for **ai.appvance** (you can use **com.nike** or so depending on the company).



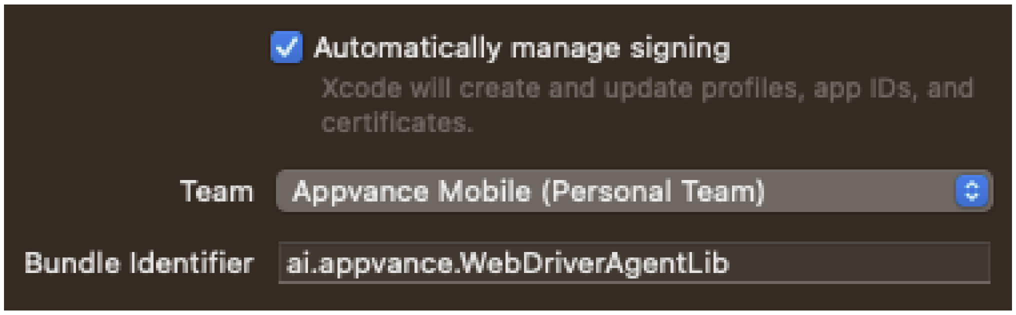
✓ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team: None
Bundle Identifier: ai.appvance.WebDriverAgentLib

iOS
Bundle Identifier: ai.appvance.WebDriverAgentLib
Provisioning Profile: None Required
Signing Certificate: Don't Code Sign (No Team Selected)

macOS
Bundle Identifier: ai.appvance.WebDriverAgentLib
Provisioning Profile: None Required
Signing Certificate: Don't Code Sign (No Team Selected)

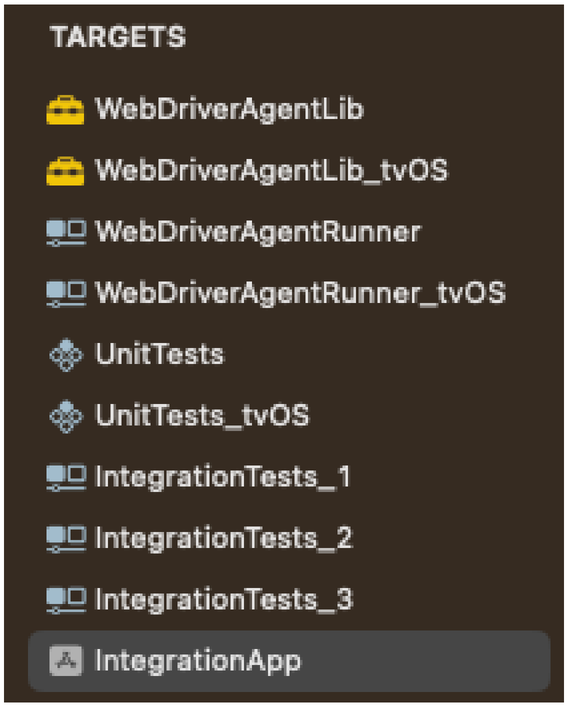
8. Make sure the **Team** field has the right value selected. This will depend on the owner of the app as well.



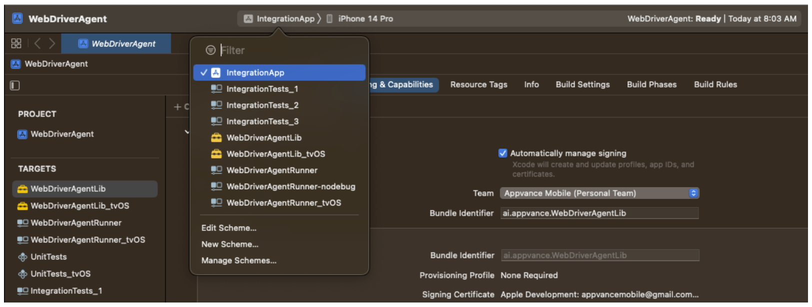
✓ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team: Appvance Mobile (Personal Team)
Bundle Identifier: ai.appvance.WebDriverAgentLib

9. Update the **Bundle Identifier** and **Team** values for all **Targets** in the project. Repeat the process in steps 5 through 8 for each target in the list.



10. To finish, select **WebDriverAgentLib** as destination the target and press **Command** + **B** or go to **Product / Build**.



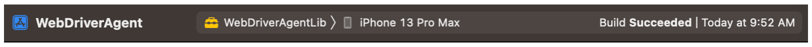
WebDriverAgent: **Ready** | Today at 8:03 AM

✓ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team: Appvance Mobile (Personal Team)
Bundle Identifier: ai.appvance.WebDriverAgentLib

Bundle Identifier: ai.appvance.WebDriverAgentLib
Provisioning Profile: None Required
Signing Certificate: Apple Development: appvancemobile@gmail.com...

11. A successful build displays the following message.



WebDriverAgent WebDriverAgentLib 〉 iPhone 13 Pro Max **Build Succeeded** | Today at 9:52 AM

Visual Accessor Example for iOS

To help you learn the process, this example uses iOS Zoo as the example mobile application. iOS Zoo is an educational zoo app which allow users to view the animals in a list and find the location of a particular animal.

The configuration file, the images for the smart tags and the smart tags themselves are going to correspond to the iOS Zoo app.

You are not required to use the iOS Zoo app for your configuration testing. You can apply this process to any app you want. Be sure to make the necessary adjustments based on the alternative app you choose.

iOS-ZooAppThis app can be downloaded from: https://-github.com/ShirleyDong/IOS-ZooApp

Creating the Configuration File

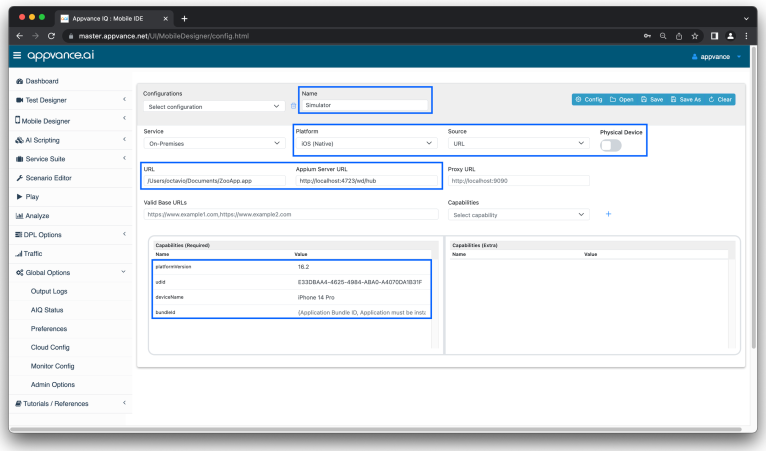
A Mobile Configuration file is required to describe which app will be launched during the Play/Recording/Blueprint and what is the target device and its capabilities.

To create a configuration file follow the steps below.

1. In AIQ navigate to **Mobile Designer** > **Mobile Configuration**.
2. In the Name field, enter **Simulator**.
3. Select **iOS (Native)** from the **Platform** dropdown.
4. Select **URL** from the **Source** dropdown.
5. Turn off the **Physical Device** switch.
6. Generate a build. See "Generating a build for iOS" on page 53 for instructions.
7. When the build is complete, paste the build path in the **URL** field.
8. Enter **16.2** for **platformVersion** field in the **Capabilities** section.
9. Enter **E33DBAA4-4625-4984-ABA0-A4070DA1B31F** the **udid** in the **Capabilities** section. You can get the UDID for any device you want to use. See "Getting a UDID" on page 46 for instructions.
10. Enter **iPhone 14 Pro** for the **platformVersion** in the **Capabilities** section.

11. The **bundleID** in the **Capabilities** section should be left empty since we are are using **URL**.

    This is an example of the configuration.



Configurations: Select configuration
Name: Simulator
Service: On-Premises
Platform: iOS (Native)
Source: URL
Physical Device
URL: /Users/octavio/Documents/ZooApp.app
Appium Server URL: http://localhost:4723/wd/hub
Proxy URL: http://localhost:9090
Valid Base URLs: https://www.example1.com,https://www.example2.com
Capabilities: Select capability

Capabilities (Required)

Name	Value
platformVersion	16.2
udid	E33DBAA4-4625-4984-ABA0-A4070DA1B31F
deviceName	iPhone 14 Pro
bundleId	{Application Bundle ID, Application must be insta

Capabilities (Extra)

Name	Value

For more information about where these values come from, refer to Mobile Configuration File for iOS.

12. Click **Save** and save the file on an easy-access location. Save the path.

    The configuration file will look like this:

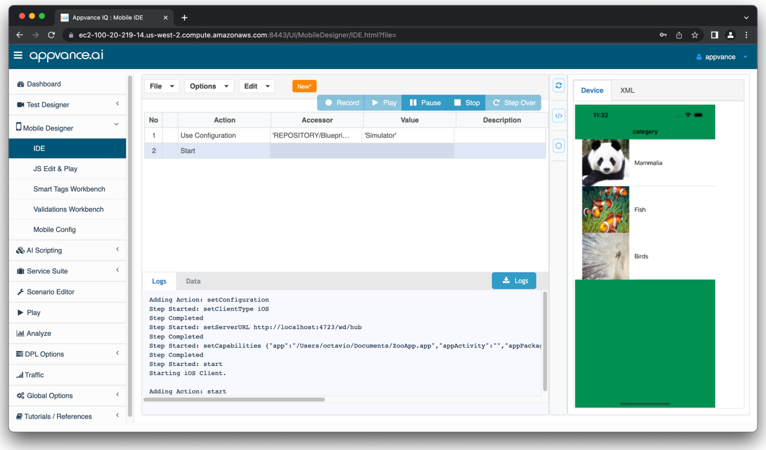
```
[
    {
        "configName":"Simulator",
        "app":"/Users/octavio/Documents/ZooApp.app",
        "appPackage":"",
        "proxyURL":"",
        "bundleId":"",
        "source":"path",
        "appActivity":"",
        "deviceMode":false,
        "service":"On-Premises",
        "serverURL":"http://localhost:4723/wd/hub",
        "platformName":"iOS",
        "validDomains":"",
        "capabilities":{
           "platformVersion":"16.2",
           "udid":"E33DBAA4-4625-4984-ABA0-
   A4070DA1B31F",
           "app":"/Users/octavio/Documents/ZooApp.app",
           "deviceName":"iPhone 14 Pro",
           "bundleId":""      }
     }
   ]
```

Testing the Configuration

Test the configuration to verify it is correct.

1. Start an Appium Server.
2. In AIQ navigate to **Mobile Designer** > **IDE** > **File,** > **New**,
3. Click **Record**.
4. Search for the configuration file just created and wait until the app is launched.

- If everything is well configured you will see something like the following image.



No		Action	Accessor	Value	Description
1		Use Configuration	'REPOSITORY/Bluepri…	'Simulator'	
2		Start			

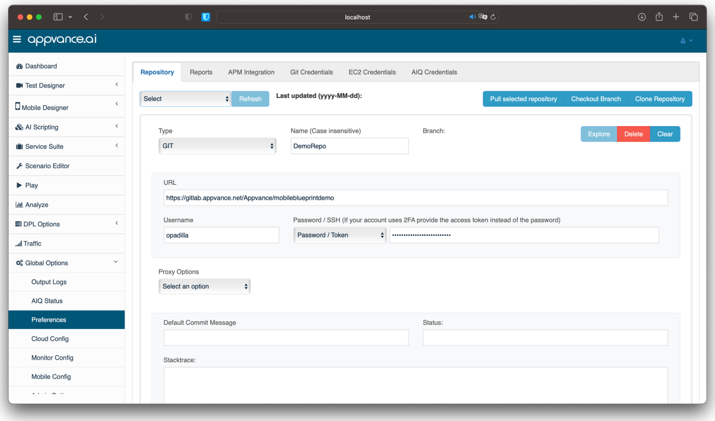
```
Adding Action: setConfiguration
Step Started: setClientType iOS
Step Completed
Step Started: setServerURL http://localhost:4723/wd/hub
Step Completed
Step Started: setCapabilities {"app":"/Users/octavio/Documents/ZooApp.app","appActivity":"","appPacka
Step Completed
Step Started: start
Starting iOS Client.

Adding Action: start
```

- If a problem occurs it is going to be printed in the **Log** tab at the bottom of the screen or in the **AppiumInspector**. The logs can be used to figure it out what the problem is. You may need to make some changes in the configuration file (typically there are errors related to incomplete names or copied parameters, do a comparison with the example provided in the snippet above if any trouble occurs).

Configuring Access to a Repository

In order to save images Appvance IQ has to connect to a git repository.

1. To configure a repository, in AIQ navigate to **Global Options > Preferences > Repository**.



Last updated (yyyy-MM-dd):

Type: GIT

Name (Case insensitive): DemoRepo

Branch:

URL: https://gitlab.appvance.net/Appvance/mobileblueprintdemo

Username: opadilla

Password / SSH (If your account uses 2FA provide the access token instead of the password): Password / Token ••••••••••••••••••••••••••

Proxy Options: Select an option

Default Commit Message:

Status:

Stacktrace:

Suggested parameters:

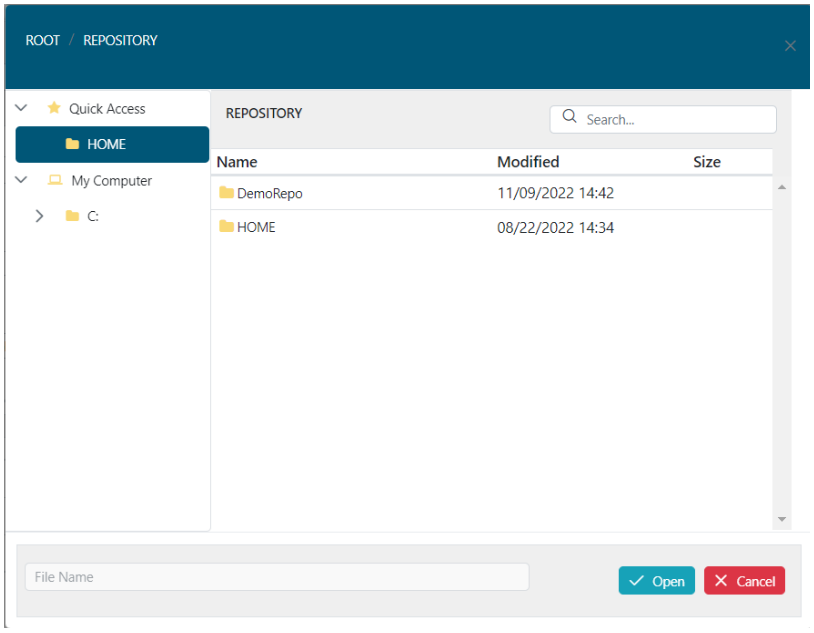
- **Type**: GIT
- **Name**: DemoRepo
- **URL**: https://gitlab.com/AIQBlueprints/mobileblueprint.git
- **Username**: GitLab user with access to the repository
- **Password**: GitLab user password or token

These parameters can be changed if the user has access to another repository.

2. Click **Clone**.

3. If the configuration is correct, the repository is successfully saved. If not, double check your configuration.

Anytime a file or document is saved or loaded it can be accessed from the repository in **ROOT > REPOSITORY > repository_name**.



Name	Modified	Size
DemoRepo	11/09/2022 14:42	
HOME	08/22/2022 14:34	

Taking Images for Smart Tags

Visual accessor functionality allows using graphical representations as possible Smart Tags. Images from the application can be taken when recording using the Mobile Designer IDE.

When creating the images for the smart tags it could be helpful to add many images from the same target taken in different methods and sizes too (preferably bigger than the original size). For example using the simulator you can take a screenshot and crop images by using an image editor or simply using MacOS Screenshot tool.

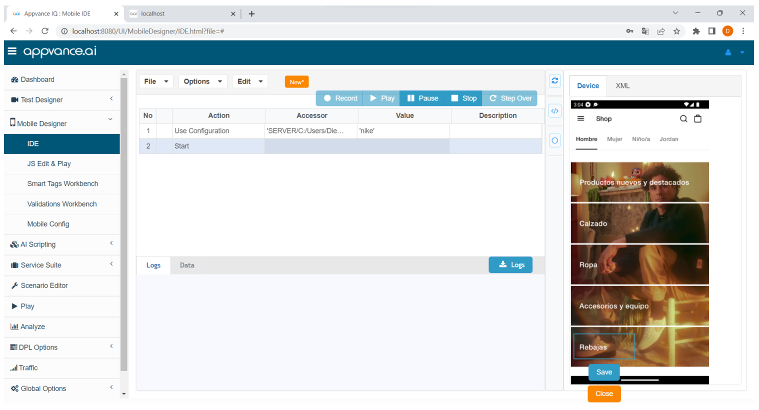
Images must be saved in a public URL or repository as the one included in the step 4. Getting access to a repository.

In the following example are going to capture three different images to use for smart tags.

To start creating the images follow the steps below.

1. Start the **Appium Server**
2. Go to **Appvance IQ / Mobile Designer / IDE**
3. Press **Record** and load the configuration file previously created in the 3. Creating the configuration file section.
4. Wait until it loads.
5. Once the app preview is visible you can click and drag to draw a square over any area of the app window. After doing this you will see the **Save** &

Close buttons as show in the following screenshot.



No		Action	Accessor	Value	Description
1		Use Configuration	'SERVER/C:/Users/Die…	'nike'	
2		Start			

6. When you click **Save** you will be prompted for a location to save the image. Save the image to a repository. If you do not save the image to a repository, you will not be able to properly create the smart tag for that image later.

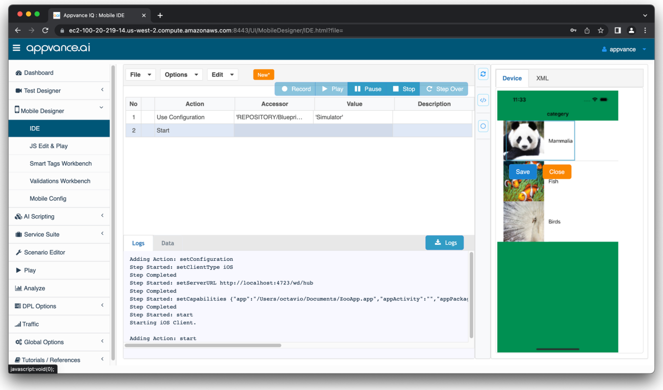
7. Example image 1 : Mammalia item

   a. Draw a square over the mammalia item in the home screen. Include both image and text.

   b. Click **Save**.

   c. Search for **ROOT > REPOSITORY > DemoRepo> iOS > ZooApp >**

images and save it as "**mammalia_item**".



No		Action	Accessor	Value	Description
1		Use Configuration	'REPOSITORY/Bluepri…	'Simulator'	
2		Start			

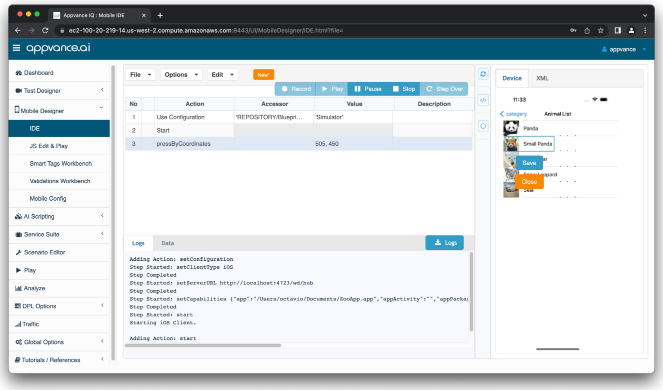
```
Adding Action: setConfiguration
Step Started: setClientType iOS
Step Completed
Step Started: setServerURL http://localhost:4723/wd/hub
Step Completed
Step Started: setCapabilities {"app":"/Users/octavio/Documents/ZooApp.app","appActivity":"","appPackag
Step Completed
Step Started: start
Starting iOS Client.

Adding Action: start
```

8. Example image 2: Small panda item

   a. Draw a square over the small panda item in the home screen. Include both image and text.

   b. Click **Save**.

   c. Search for **ROOT > REPOSITORY > DemoRepo> iOS > ZooApp > images** and save it as "**small_panda_item**".



No		Action	Accessor	Value	Description
1		Use Configuration	'REPOSITORY/Bluepri…	'Simulator'	
2		Start			
3		pressByCoordinates		505, 450	

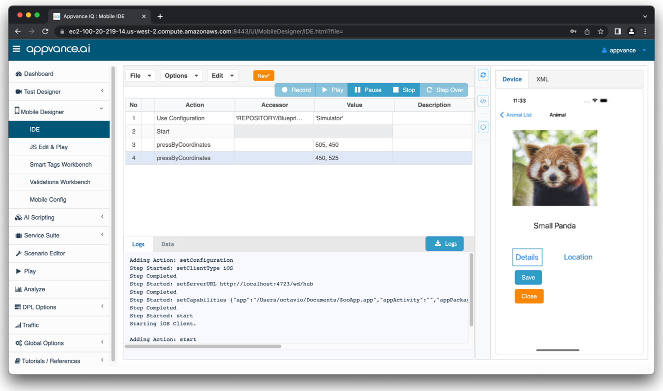
```
Adding Action: setConfiguration
Step Started: setClientType iOS
Step Completed
Step Started: setServerURL http://localhost:4723/wd/hub
Step Completed
Step Started: setCapabilities {"app":"/Users/octavio/Documents/ZooApp.app","appActivity":"","appPackag
Step Completed
Step Started: start
Starting iOS Client.

Adding Action: start
```

9. Example image 3: Details button

a. Draw a square over the Details button.

b. Click **Save**.

c. Search for **ROOT > REPOSITORY > DemoRepo> iOS > ZooApp > images** and save it as "**details_button**".



No	Action	Accessor	Value	Description
1	Use Configuration	'REPOSITORY/Bluepri...	'Simulator'	
2	Start			
3	pressByCoordinates		505, 450	
4	pressByCoordinates		450, 525	

```
Adding Action: setConfiguration
Step Started: setClientType iOS
Step Completed
Step Started: setServerURL http://localhost:4723/wd/hub
Step Completed
Step Started: setCapabilities {"app":"/Users/octavio/Documents/ZooApp.app","appActivity":"","appPacka
Step Completed
Step Started: start
Starting iOS Client.

Adding Action: start
```

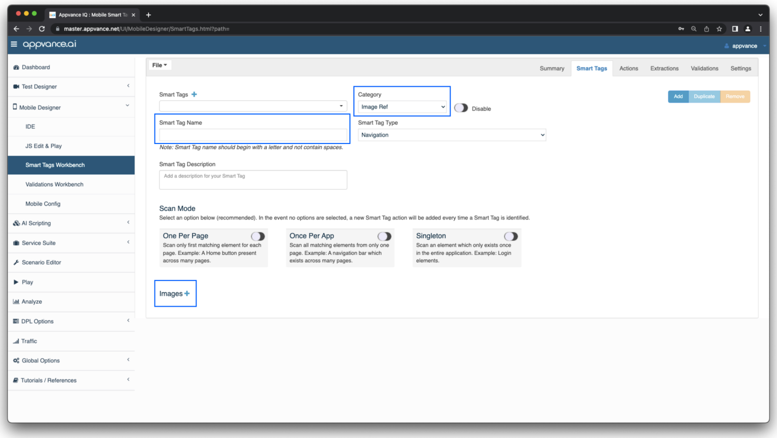
When saving images from the device preview try giving the file a meaningful name. This will be helpful when creating the smart tags.

Creating the Smart Tags File

A new type of Smart Tag is introduced, image reference, it can be provided from a URL or repository.

In order to create a Smart Tag follow the steps below.

1. Go to **Appvance IQ** / **Mobile Designer** / **Smart Tags Workbench**.
2. Change **Category** to **Image Ref**.
3. Fill in **Smart Tag Name**.
4. Click on the **+** button next to **Images** at the bottom of the form. You can type/paste a URL of the target image or browse your image by clicking on the **Browse** button.



Smart Tags +

Category
Image Ref

Smart Tag Name

Note: Smart Tag name should begin with a letter and not contain spaces.

Smart Tag Type
Navigation

Smart Tag Description

Add a description for your Smart Tag

Scan Mode

Select an option below (recommended). In the event no options are selected, a new Smart Tag action will be added every time a Smart Tag is identified.

One Per Page
Scan only first matching element for each page. Example: A Home button present across many pages.

Once Per App
Scan all matching elements from only one page. Example: A navigation bar which exists across many pages.

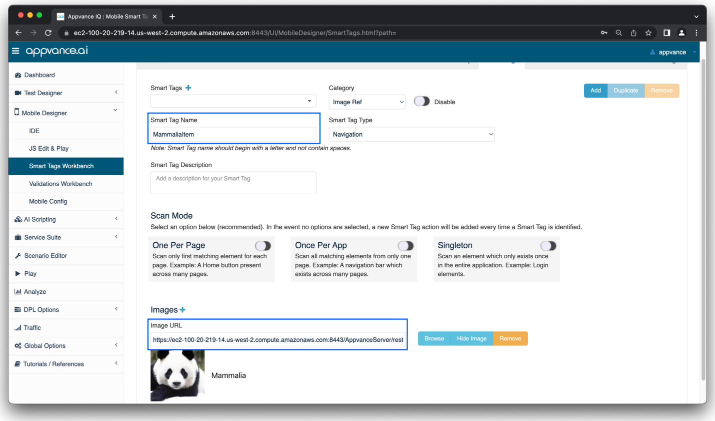
Singleton
Scan an element which only exists once in the entire application. Example: Login elements.

You can use the **Show Image** button next to each image URL in order to show the image and make sure you are using the right one.

To add more **Smart Tags** in the same file, click **Add** in the top right section and repeat the same steps. When all Smart Tags are created, save them in a file by clicking on the **File** / **Save** menu.

For our example we are going to create a smart tag for each image captured in step 5.

1. Smart tag for MammaliaItem



Smart Tags
Category
Image Ref
Disable
Smart Tag Name
MammaliaItem
Smart Tag Type
Navigation
Note: Smart Tag name should begin with a letter and not contain spaces.
Smart Tag Description
Add a description for your Smart Tag

Scan Mode
Select an option below (recommended). In the event no options are selected, a new Smart Tag action will be added every time a Smart Tag is identified.

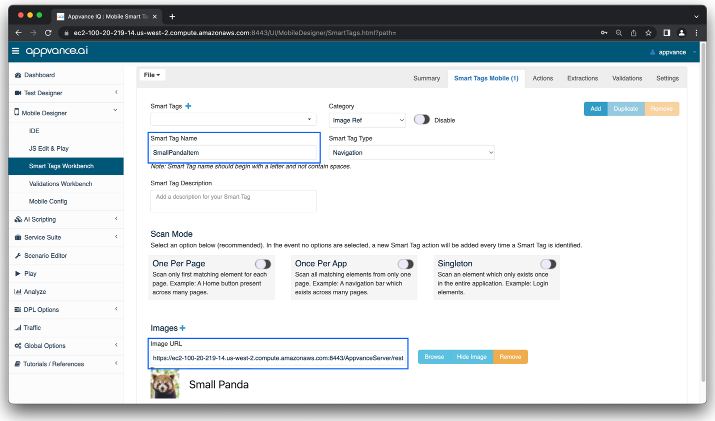
One Per Page
Scan only first matching element for each page. Example: A Home button present across many pages.

Once Per App
Scan all matching elements from only one page. Example: A navigation bar which exists across many pages.

Singleton
Scan an element which only exists once in the entire application. Example: Login elements.

Images
Image URL
https://ec2-100-20-219-14.us-west-2.compute.amazonaws.com:8443/AppvanceServer/rest
Mammalia

2. Smart tag for SmallPandaItem



Smart Tags
Category
Image Ref
Disable
Smart Tag Name
SmallPandaItem
Smart Tag Type
Navigation
Note: Smart Tag name should begin with a letter and not contain spaces.
Smart Tag Description
Add a description for your Smart Tag

Scan Mode
Select an option below (recommended). In the event no options are selected, a new Smart Tag action will be added every time a Smart Tag is identified.

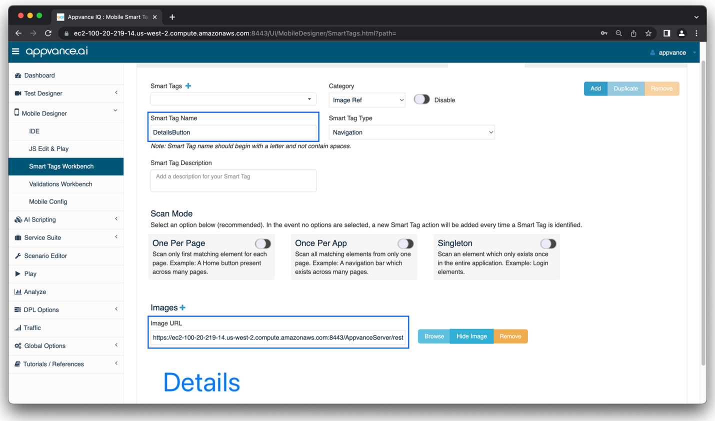
One Per Page
Scan only first matching element for each page. Example: A Home button present across many pages.

Once Per App
Scan all matching elements from only one page. Example: A navigation bar which exists across many pages.

Singleton
Scan an element which only exists once in the entire application. Example: Login elements.

Images
Image URL
https://ec2-100-20-219-14.us-west-2.compute.amazonaws.com:8443/AppvanceServer/rest
Small Panda

3. Smart tag for DetailsButton



Smart Tags

Category
Image Ref

Disable

Smart Tag Name
DetailsButton

Smart Tag Type
Navigation

Note: Smart Tag name should begin with a letter and not contain spaces.

Smart Tag Description
Add a description for your Smart Tag

Scan Mode

Select an option below (recommended). In the event no options are selected, a new Smart Tag action will be added every time a Smart Tag is identified.

One Per Page
Scan only first matching element for each page. Example: A Home button present across many pages.

Once Per App
Scan all matching elements from only one page. Example: A navigation bar which exists across many pages.

Singleton
Scan an element which only exists once in the entire application. Example: Login elements.

Images

Image URL
https://ec2-100-20-219-14.us-west-2.compute.amazonaws.com:8443/AppvanceServer/rest

4. Navigate to **File > Save As**.

5. Navigate to **ROOT > REPOSITORY > DemoRepo > iOS > ZooApp** and save the file as **Zoo-SmartTags**.

6. After saving the file you will get a file that look similar to the JSON below.

```
{
       "categoryId": "ImageRef",
       "threshold": "0.75",
       "smartTags": [
               {
                       "name": "DetailsButton",
                       "category": "ImageRef",
                       "type": "Navigation",
                       "tags": [
                               "https://-
mas-
ter-
.ap-
pvance.net/Ap-
pvanceServer-
/rest/pub-
lic/ap-
pvance/REPOSITORY/demoAnimals/iOS/ZooApp/images/details_
button.png"                        ],
                        "rules": [],
                        "info": {},
                        "disabled": false,
                        "description": "",
                        "singleton": false,
                        "scanOncePerApp": false,
                        "scanOnePerPage": false
                },
                {
                        "name": "MammaliaItem",
                        "category": "ImageRef",
                        "type": "Navigation",
                        "tags": [
                                "https://-
mas-
ter-
.ap-
pvance.net/Ap-
pvanceServer-
/rest/pub-
lic/ap-
pvance/REPOSITORY/demoAnimals/iOS/ZooApp/images/mammalia_
item.png"                        ],
                        "rules": [],
                        "info": {},
                        "disabled": false,
                        "description": "",
                        "singleton": false,
                        "scanOncePerApp": false,
                        "scanOnePerPage": false
                },
                {
                        "name": "SmallPandaItem",
                        "category": "ImageRef",
                        "type": "Navigation",
                        "tags": [
                                "https://-
mas-
ter-
.ap-
pvance.net/Ap-
pvanceServer-
/rest/pub-
lic/appvance/REPOSITORY/demoAnimals/iOS/ZooApp/images/small_
panda_item.png"                 ],
                       "rules": [],
                       "info": {},
                       "disabled": false,
                       "description": "",
                       "singleton": false,
                       "scanOncePerApp": false,
                       "scanOnePerPage": false
               }
       ],
       "actions": [],
       "customTags": [],
       "validators": [],
       "extractors": []
}
```

iOS Reference

The following sections contain information on configuration steps that, depending on your existing environment, you may have to follow to use the visual accessor functionality.

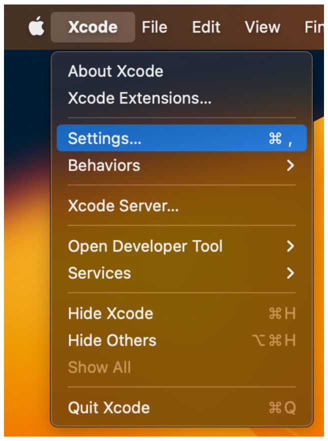
Setup Apple Developer account on Xcode

Follow these steps to setup an Apple Developer account on Xcode. Make sure you have the email and password associated with your account.

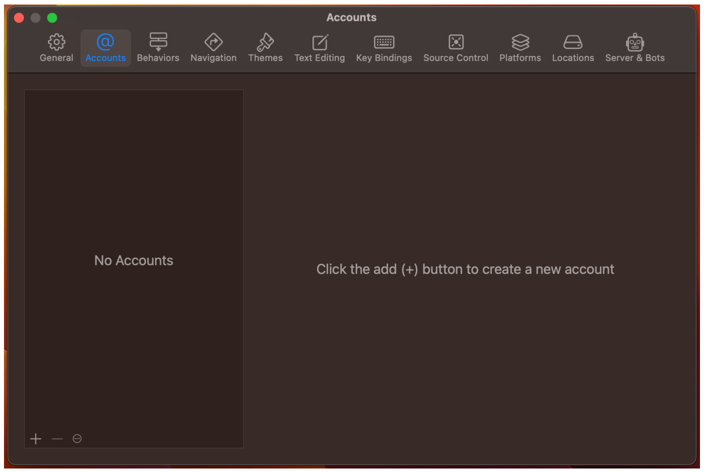
An Xcode account is required when signing a project. Signing is a process required by Apple when you want to create a build or run a project on a physical device. See "Signing Xcode project" on page 43 for more information.

If Two-Factor Authentication is enabled for the account, make sure you have access to at least one of the authentication methods. See Apple's information on two-factor authentication for more information.

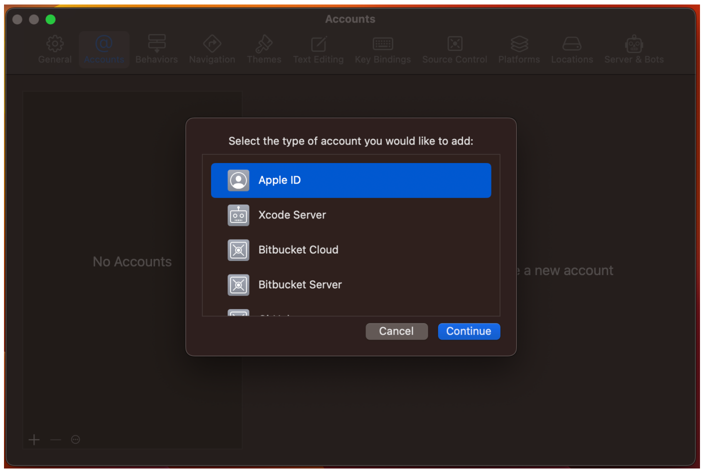
1. Open **Xcode** and select **Settings**.



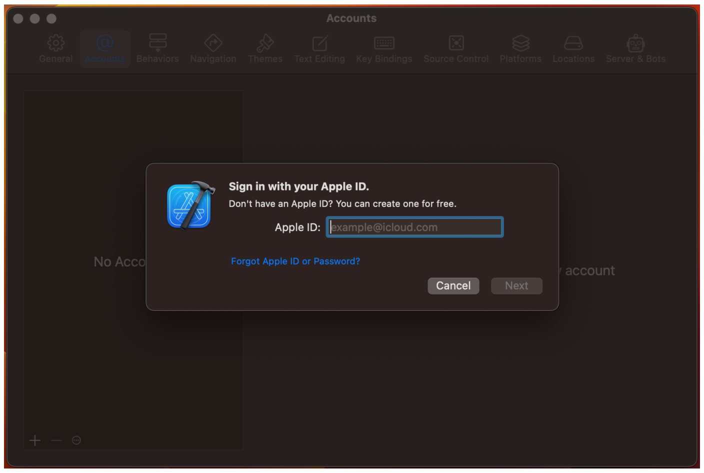
2. Navigate to the **Accounts** tab and click the add (+) button.



3. Select the **Apple ID** option and click **Continue**.



4. Enter the email associated with the **Developer Account** and click **Next**.



Sign in with your Apple ID.
Don't have an Apple ID? You can create one for free.
Apple ID: example@icloud.com
Forgot Apple ID or Password?

5. Type the password and click **Next**. Wait for the authentication process to finish.

6. If you have enabled two-factor authentication, enter the verification code. If necessary, follow any additional steps to get a verification code.

7. Select the **Developer Account** you just authenticated from the list of **Apple IDs**.

8. Click **Download Manual Profiles**.

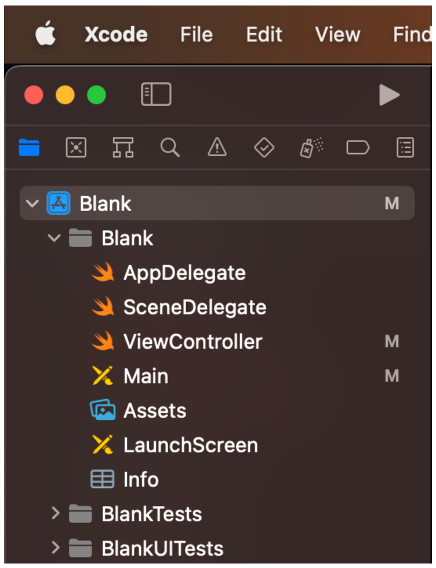
.

Signing Xcode project

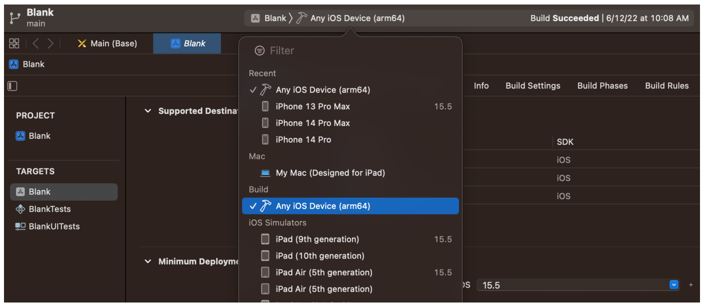
Signing is a process required by Apple when you want to create a build or run a project on a physical device.

Prerequisite: You must have an Apple Developer Account before completing this process. See "Setup Apple Developer account on Xcode" on page 40 for more information on that process.

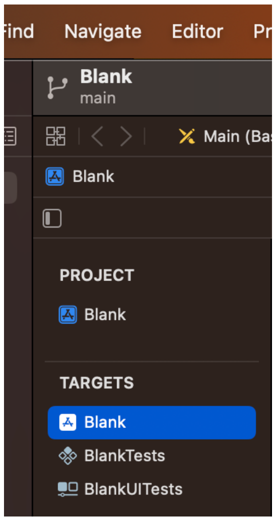
1. Open your app project using **Xcode** and select your project in the **Project Navigator**..



2. Select **Any iOS Device (arm64)** as the build target.

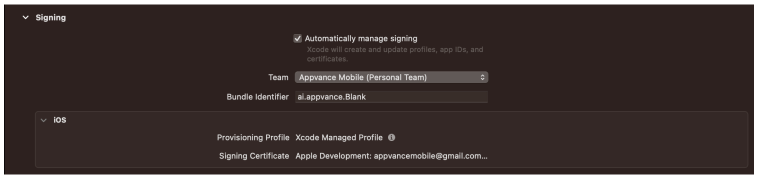


3. Select your project target from the **Targets** list.



4. Select the **Signing & Capabilities** tab.

5. In the **Team** field select your Apple Developer Account.



Signing

[✓] Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team: Appvance Mobile (Personal Team)
Bundle Identifier: ai.appvance.Blank

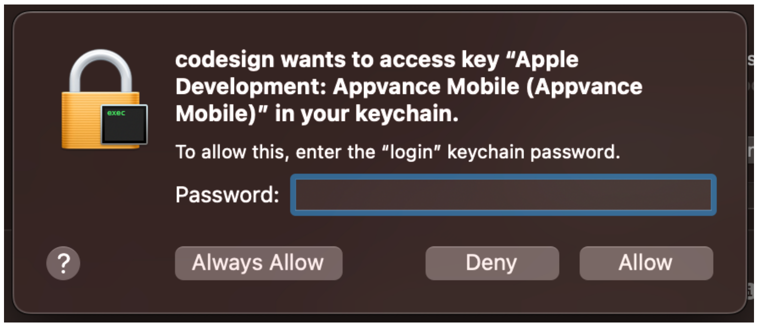
iOS

Provisioning Profile: Xcode Managed Profile
Signing Certificate: Apple Development: appvancemobile@gmail.com...

6. Verify that the **Bundle Identifier** matches your company domain. It should be written backwards. For example if your domain is `appvance.ai` you

should use ai.appvance and the app name. In this example it would be `ai.appvance.Blank`.

7. Build your project by pressing **Command + B** or navigating to the **Product / Build** menu. If you prompted for your password, enter it and click **Always Allow**.

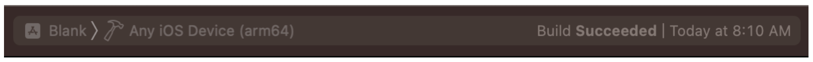


codesign wants to access key "Apple Development: Appvance Mobile (Appvance Mobile)" in your keychain.

To allow this, enter the "login" keychain password.

Password:

8. A successful build will display the following message..



Blank 〉 Any iOS Device (arm64)   Build **Succeeded** | Today at 8:10 AM

Getting a UDID

UDID is an acronym for Unique Device Identifier. It is a unique identifier that is calculated from different hardware values. The UDID is a feature of Apple devices running iOS, tvOS, watchOS, and macOS.

For the purposes of this example, it is used to determine the destination device to run tests.

> If you need to setup Xcode before this step, see "Setup Apple Developer account on Xcode" on page 40.

Getting a simulator UDID

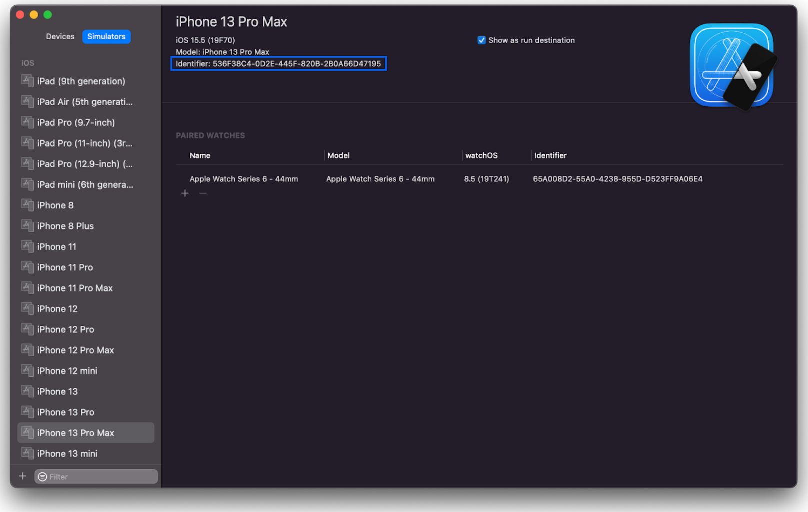
1. Open **Xcode**.

2. Go to the **Window** menu and select the **Devices and Simulators** option.

3. Select the **Simulators** tab in the upper left corner.

4. Pick the simulator you want to run the test on from the list of all available simulators on the left.

5. Look for the **Identifier** field on the detail screen. Right click on its value and select **Copy**.



iPhone 13 Pro Max

iOS 15.5 (19F70)
Model: iPhone 13 Pro Max
Identifier: 536F38C4-0D2E-445F-820B-2B0A66D47195

[✓] Show as run destination

PAIRED WATCHES

Name	Model	watchOS	Identifier
Apple Watch Series 6 - 44mm	Apple Watch Series 6 - 44mm	8.5 (19T241)	65A008D2-55A0-4238-955D-D523FF9A06E4

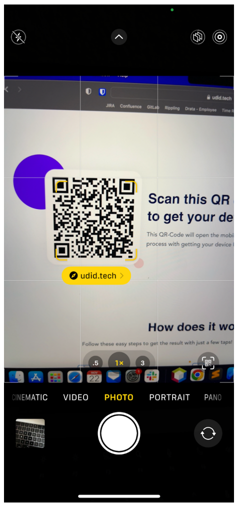
Getting a physical device UDID

These steps apply for both iPhone and iPad devices.

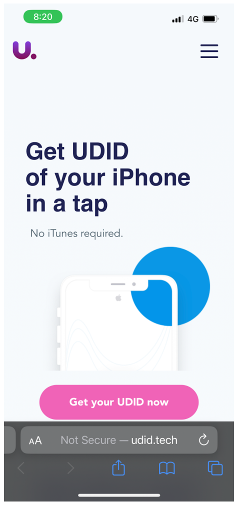
There are several ways to get your device UDID. Here's one of the easiest.

1. Get your device and open the Camera app.

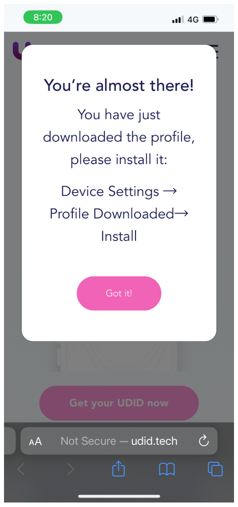
2. On your Mac go to https://udid.tech/ and scan the QR on the page. And follow the link.



3. You'll be taken to a web page. Tap on **Get your UDID** now.



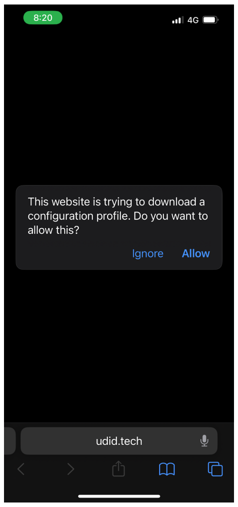
4. Tap on **Got it!**


You're almost there!

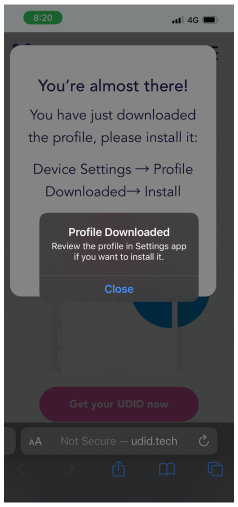
You have just downloaded the profile, please install it:

Device Settings → Profile Downloaded→ Install

5. Tap on **Allow** to permit the download.


This website is trying to download a configuration profile. Do you want to allow this?

6. Close the confirmation popup.



You're almost there!

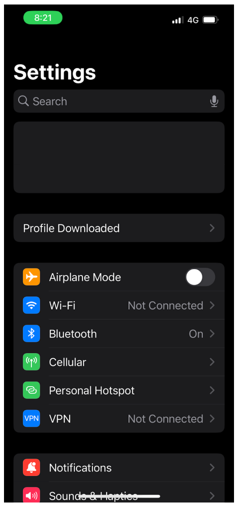
You have just downloaded the profile, please install it:

Device Settings → Profile Downloaded→ Install

Profile Downloaded

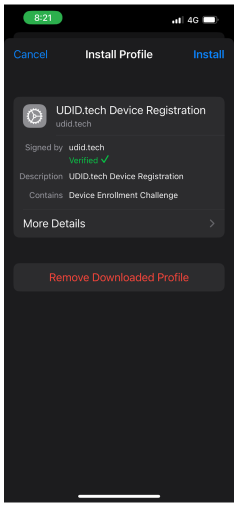
Review the profile in Settings app if you want to install it.

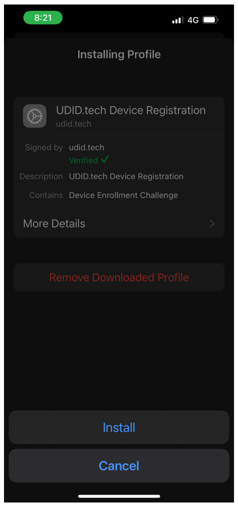
7. Go to your device **Settings** and tap on **Profile Downloaded**.



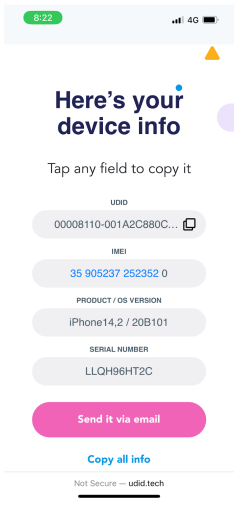
Airplane Mode
Wi-Fi Not Connected
Bluetooth On
Cellular
Personal Hotspot
VPN Not Connected

8. Select **Install** and then press **Install** again. If you have a passcode it's likely you will get a prompt to confirm your identity before installing the profile.


8:21
4G
Cancel
Install Profile
Install
UDID.tech Device Registration
udid.tech
Signed by udid.tech
Verified ✓
Description UDID.tech Device Registration
Contains Device Enrollment Challenge
More Details
Remove Downloaded Profile


8:21
4G
Installing Profile
UDID.tech Device Registration
udid.tech
Signed by udid.tech
Verified ✓
Description UDID.tech Device Registration
Contains Device Enrollment Challenge
More Details
Remove Downloaded Profile
Install
Cancel

8. You'll be taken to a web page with all your device information. Look for the UDID field and tap on it to copy its value.



8:22 4G

Here's your device info

Tap any field to copy it

UDID

00008110-001A2C880C...

IMEI

35 905237 252352 0

PRODUCT / OS VERSION

iPhone14,2 / 20B101

SERIAL NUMBER

LLQH96HT2C

Generating a build for iOS

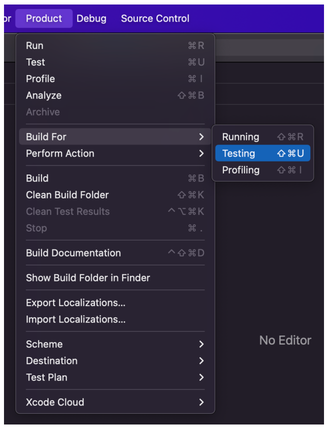
Generating build for iOS is quite simple but the steps are different for simulators and physical devices. The process is the same for iOS and iPadOS.

The following processes are detailed:

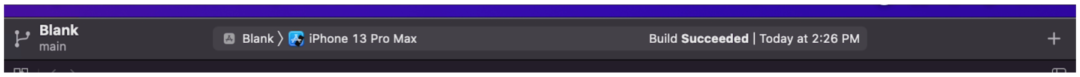
- Generating the Build for iOS Simulators
- Generating the Build for iOS Devices
- Retrieving the Build

Generating build for iOS Simulators

1. Open **Xcode**.
2. Open the project you are working on.
3. In the **Product** menu click on **Build For** and select **Testing**.



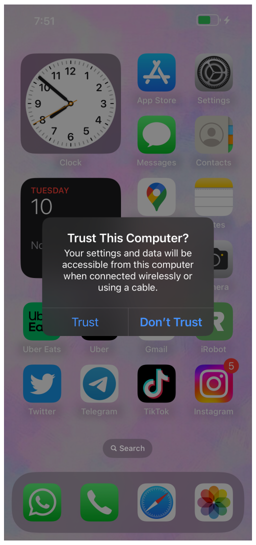
4. Verify that the build succeeds.



Generating build for iOS Devices

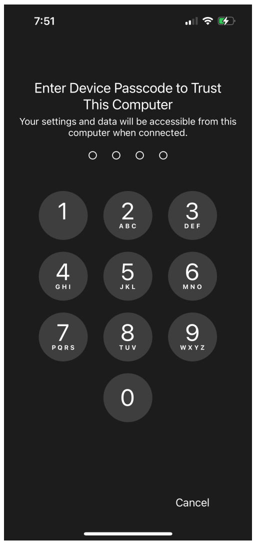
To successfully create a build targeting a physical device you need to have already "Setup Apple Developer account on Xcode" on page 40.

1. Open **Xcode**.
2. Open the project you are working on.
3. Pair your device with your computer.
   a. Connect the physical device to your Mac.
   b. Unlock your device and tap the **Trust** button in the alert.



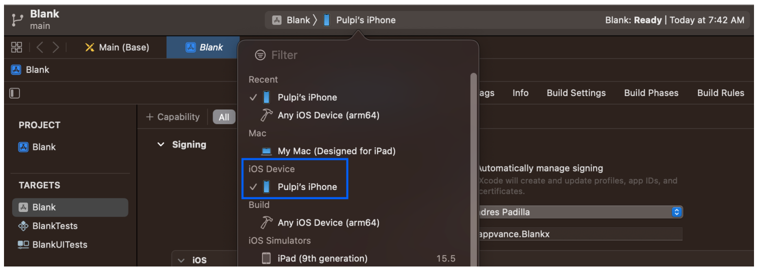
Trust This Computer?
Your settings and data will be accessible from this computer when connected wirelessly or using a cable.

   c. If asked, type the device passcode.

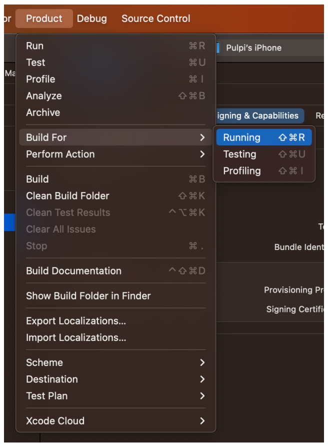

7:51
Enter Device Passcode to Trust This Computer
Your settings and data will be accessible from this computer when connected.
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0
Cancel

d. Wait for Xcode to finish the pairing.

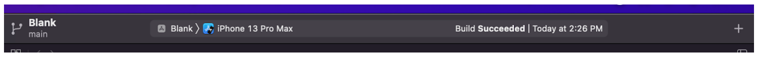
4. Select your device as the target device.



5. In the **Product** menu click on **Build For** and select **Running**.



6. Make sure the build succeeds.

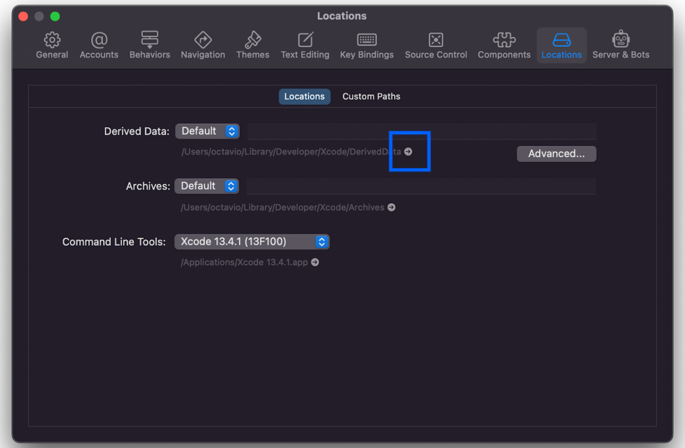


Build Succeeded | Today at 2:26 PM

Retrieving the build

1. Go to the Xcode menu and select **Preferences | Settings**.

2. Click the arrow button on the right of the **Derived Data** path.

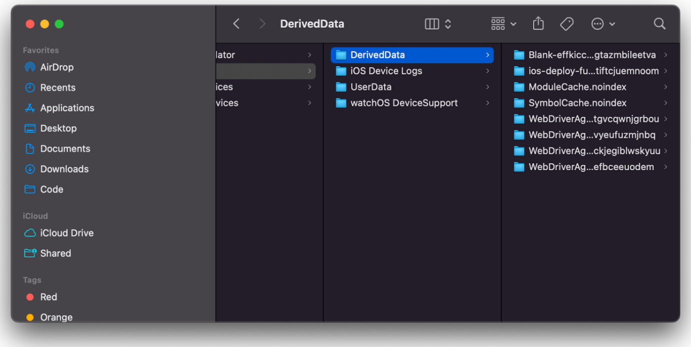


Derived Data: Default
/Users/octavio/Library/Developer/Xcode/DerivedData

Archives: Default
/Users/octavio/Library/Developer/Xcode/Archives

Command Line Tools: Xcode 13.4.1 (13F100)
/Applications/Xcode 13.4.1.app

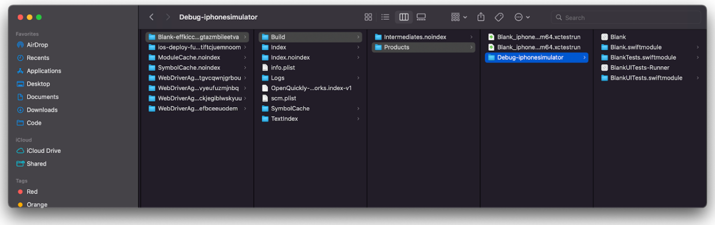
3. Look for the folder starting with the name of your project and open it. In case you find more than one folder, delete all folders and start over.



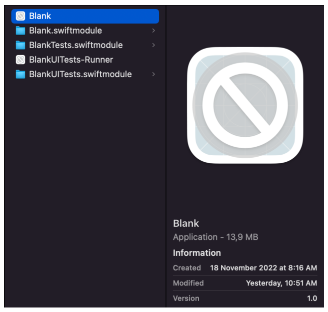
Navigate to

Build / Products / Debug-iphonesimulator if you built for simulator.

Build / Products / Debug-iphoneos if you built for device.



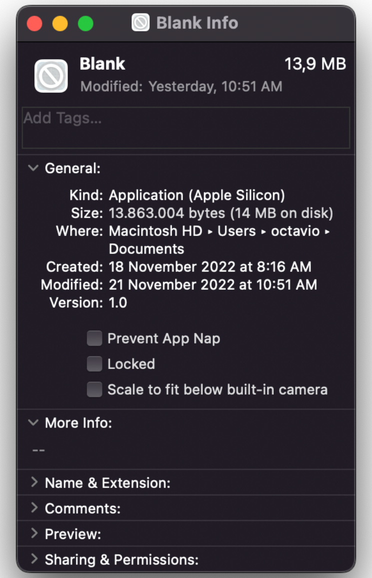
4. You should see a file with the name of your application.



5. Make sure to copy this file to a more accessible location. The **Documents** folder is a good option but you can use any location you want.

6. Once in the right location. **Right click** on the file and select **Get Info**.

7. In the **General** section look for the **Where** field. Right click and **Copy as Pathname**.



Blank 13,9 MB
Modified: Yesterday, 10:51 AM

General:
Kind: Application (Apple Silicon)
Size: 13.863.004 bytes (14 MB on disk)
Where: Macintosh HD ▸ Users ▸ octavio ▸ Documents
Created: 18 November 2022 at 8:16 AM
Modified: 21 November 2022 at 10:51 AM
Version: 1.0

8. To the text on your clipboard you should just append the name of the file to get the path to the build.

9. You should get something like this: `/Users/user/Documents/AppName.app`

Please save this path. It's going to be used when setting up the MD configuration.

Troubleshooting Appium and MacOS Ventura

Appium version 1.x is not compatible with MacOS Ventura. If you updated your Mac to Ventura you must follow the steps below in order to properly run Mobile Designer.

1. Open Terminal.
2. Install Appium version 2 by running the following command.

   ```
   npm install -g appium@next
   ```

3. Install Appiumdrivers for iOS and Android. To do this run the commands below.

   ```
   appium driver install xcuitest
   appium driver install uiautomator2
   ```

4. Update Python by running the following command.

   ```
   brew upgrade python
   ```

5. Install / update the Command Line Tools by running the following command.

   ```
   xcode-select --install
   ```

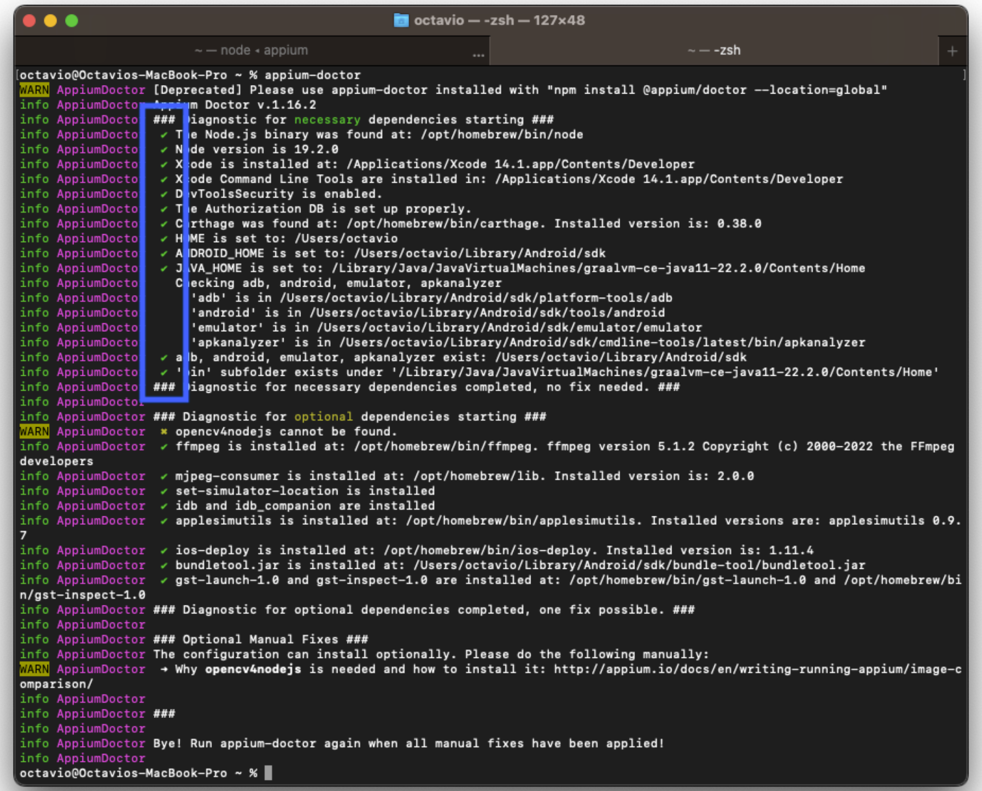
6. Update OpenCV by running the following command.

   ```
   brew install opencv
   ```

7. Execute Appium Doctor by running the following command.

```
appium-doctor
```

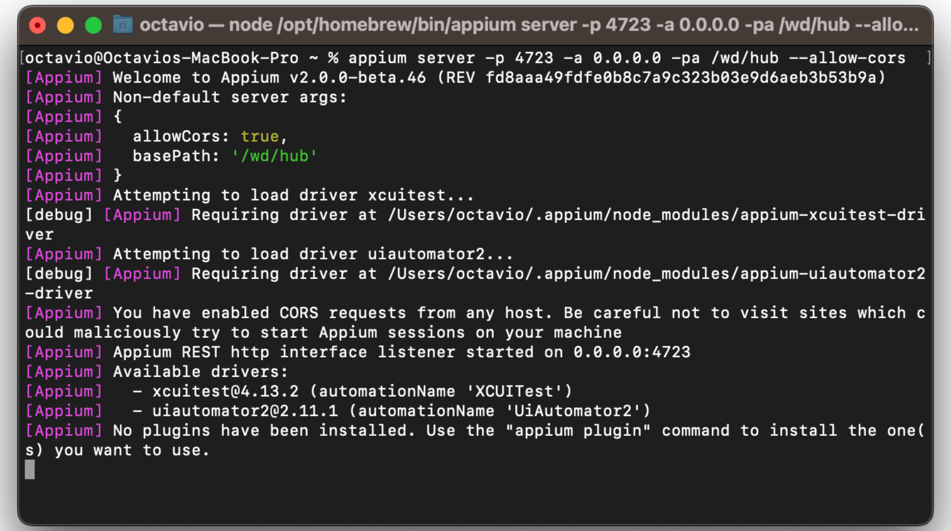
8. Validate that all required dependencies have a green tick.



```
octavio@Octavios-MacBook-Pro ~ % appium-doctor
WARN AppiumDoctor [Deprecated] Please use appium-doctor installed with "npm install @appium/doctor --location=global"
info AppiumDoctor Appium Doctor v.1.16.2
info AppiumDoctor ### Diagnostic for necessary dependencies starting ###
info AppiumDoctor  ✔ The Node.js binary was found at: /opt/homebrew/bin/node
info AppiumDoctor  ✔ Node version is 19.2.0
info AppiumDoctor  ✔ Xcode is installed at: /Applications/Xcode 14.1.app/Contents/Developer
info AppiumDoctor  ✔ Xcode Command Line Tools are installed in: /Applications/Xcode 14.1.app/Contents/Developer
info AppiumDoctor  ✔ DevToolsSecurity is enabled.
info AppiumDoctor  ✔ The Authorization DB is set up properly.
info AppiumDoctor  ✔ Carthage was found at: /opt/homebrew/bin/carthage. Installed version is: 0.38.0
info AppiumDoctor  ✔ HOME is set to: /Users/octavio
info AppiumDoctor  ✔ ANDROID_HOME is set to: /Users/octavio/Library/Android/sdk
info AppiumDoctor  ✔ JAVA_HOME is set to: /Library/Java/JavaVirtualMachines/graalvm-ce-java11-22.2.0/Contents/Home
info AppiumDoctor    Checking adb, android, emulator, apkanalyzer
info AppiumDoctor      'adb' is in /Users/octavio/Library/Android/sdk/platform-tools/adb
info AppiumDoctor      'android' is in /Users/octavio/Library/Android/sdk/tools/android
info AppiumDoctor      'emulator' is in /Users/octavio/Library/Android/sdk/emulator/emulator
info AppiumDoctor      'apkanalyzer' is in /Users/octavio/Library/Android/sdk/cmdline-tools/latest/bin/apkanalyzer
info AppiumDoctor  ✔ adb, android, emulator, apkanalyzer exist: /Users/octavio/Library/Android/sdk
info AppiumDoctor  ✔ 'bin' subfolder exists under '/Library/Java/JavaVirtualMachines/graalvm-ce-java11-22.2.0/Contents/Home'
info AppiumDoctor ### Diagnostic for necessary dependencies completed, no fix needed. ###
info AppiumDoctor
info AppiumDoctor ### Diagnostic for optional dependencies starting ###
WARN AppiumDoctor  ✖ opencv4nodejs cannot be found.
info AppiumDoctor  ✔ ffmpeg is installed at: /opt/homebrew/bin/ffmpeg. ffmpeg version 5.1.2 Copyright (c) 2000-2022 the FFmpeg
developers
info AppiumDoctor  ✔ mjpeg-consumer is installed at: /opt/homebrew/lib. Installed version is: 2.0.0
info AppiumDoctor  ✔ set-simulator-location is installed
info AppiumDoctor  ✔ idb and idb_companion are installed
info AppiumDoctor  ✔ applesimutils is installed at: /opt/homebrew/bin/applesimutils. Installed versions are: applesimutils 0.9.
7
info AppiumDoctor  ✔ ios-deploy is installed at: /opt/homebrew/bin/ios-deploy. Installed version is: 1.11.4
info AppiumDoctor  ✔ bundletool.jar is installed at: /Users/octavio/Library/Android/sdk/bundle-tool/bundletool.jar
info AppiumDoctor  ✔ gst-launch-1.0 and gst-inspect-1.0 are installed at: /opt/homebrew/bin/gst-launch-1.0 and /opt/homebrew/bi
n/gst-inspect-1.0
info AppiumDoctor ### Diagnostic for optional dependencies completed, one fix possible. ###
info AppiumDoctor
info AppiumDoctor ### Optional Manual Fixes ###
info AppiumDoctor The configuration can install optionally. Please do the following manually:
WARN AppiumDoctor  → Why opencv4nodejs is needed and how to install it: http://appium.io/docs/en/writing-running-appium/image-c
omparison/
info AppiumDoctor
info AppiumDoctor ###
info AppiumDoctor
info AppiumDoctor Bye! Run appium-doctor again when all manual fixes have been applied!
info AppiumDoctor
octavio@Octavios-MacBook-Pro ~ %
```

9. Run the following command to start the Appium server.

```
appium server -p 4723 -a 0.0.0.0 -pa /wd/hub --allow-
cor
```



octavio — node /opt/homebrew/bin/appium server -p 4723 -a 0.0.0.0 -pa /wd/hub --allo...

```
octavio@Octavios-MacBook-Pro ~ % appium server -p 4723 -a 0.0.0.0 -pa /wd/hub --allow-cors
[Appium] Welcome to Appium v2.0.0-beta.46 (REV fd8aaa49fdfe0b8c7a9c323b03e9d6aeb3b53b9a)
[Appium] Non-default server args:
[Appium] {
[Appium]   allowCors: true,
[Appium]   basePath: '/wd/hub'
[Appium] }
[Appium] Attempting to load driver xcuitest...
[debug] [Appium] Requiring driver at /Users/octavio/.appium/node_modules/appium-xcuitest-dri
ver
[Appium] Attempting to load driver uiautomator2...
[debug] [Appium] Requiring driver at /Users/octavio/.appium/node_modules/appium-uiautomator2
-driver
[Appium] You have enabled CORS requests from any host. Be careful not to visit sites which c
ould maliciously try to start Appium sessions on your machine
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
[Appium] Available drivers:
[Appium]   - xcuitest@4.13.2 (automationName 'XCUITest')
[Appium]   - uiautomator2@2.11.1 (automationName 'UiAutomator2')
[Appium] No plugins have been installed. Use the "appium plugin" command to install the one(
s) you want to use.
```

You must run the Appium Server using this command from now on.

After successfully updating to Appium version 2 you must "Install and Configure Appium WebDriverAgent" on page 18.