



# **Visual Accessors for Android New Feature Spotlight**

AIQ 4.9 - Beta Release

Document Version 1.1

March 4, 2023

© 2023 Appvance Inc. All rights reserved

# TOC

---

	1
<b>Visual Accessors - New Feature Spotlight</b> .....	<b>5</b>
<b>Understanding the Image Matching Algorithm</b> .....	<b>6</b>
How the Algorithm Works .....	6
Matching Process .....	6
Recommendations and Considerations .....	8
<b>Visual Accessors for Android</b> .....	<b>11</b>
<b>Setting up the Environment</b> .....	<b>13</b>
<b>Appium Server Setup for Android</b> .....	<b>14</b>
<b>Device Manager Setup</b> .....	<b>15</b>
<b>Android Studio Setup</b> .....	<b>16</b>
Installing Android Studio .....	16
Configuring Android Studio .....	16
Setting ANDROID_HOME .....	17
Setting Android Debug Bridge Location .....	18
<b>Adding App to Emulator</b> .....	<b>21</b>
<b>Required Fields</b> .....	<b>22</b>
Getting parameters from ADB .....	22
<b>Creating a Smart Tag File for Visual Accessors</b> .....	<b>25</b>
Considerations .....	26
<b>Generating Images for Smart Tags</b> .....	<b>27</b>

---

Considerations .....	28
<b>Creating a Mobile Blueprint using Visual Accessors .....</b>	<b>29</b>
<b>Android Mobile App Example .....</b>	<b>31</b>
<b>Selecting the Mobile Device and App in the Device Manager .....</b>	<b>32</b>
<b>Creating a Mobile Configuration File .....</b>	<b>36</b>
Verifying the Mobile Configuration File .....	39
Testing the Configuration .....	39
<b>Getting Access to a Repository .....</b>	<b>41</b>
<b>Taking Images as Possible Smart Tags .....</b>	<b>43</b>
<b>Creating the Smart Tag File .....</b>	<b>45</b>
<b>Mobile Blueprint using Visual Accessors .....</b>	<b>51</b>



# 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.



T X H Z M A W

**Desired Features**



Q 0 8 6 S 9

**Low Features**

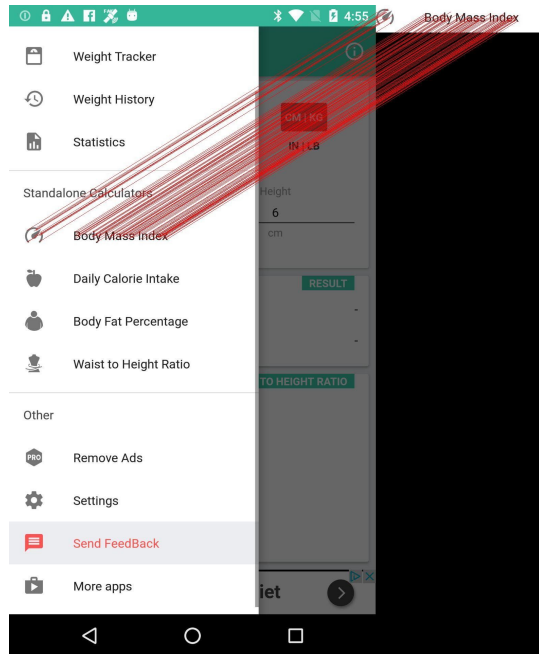
The images in the “Desired Features” set have a significant number of distinguishing features that set them apart from the other images in that set.

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

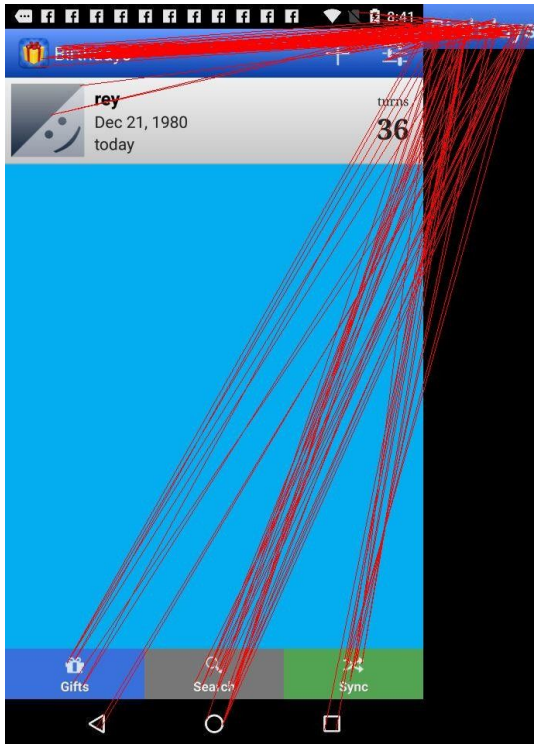
- The first thing the algorithm does is compute the distinguishing feature points for each image (target and source).

- 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 Android

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

- "Setting up the Environment" on page 13
  - "Appium Server Setup for Android" on page 14
  - "Device Manager Setup" on page 15
  - "Android Studio Setup" on page 16
  - "Adding App to Emulator" on page 21
  - "Required Fields" on page 22
- "Creating a Smart Tag File for Visual Accessors" on page 25
- "Generating Images for Smart Tags" on page 27
- "Creating a Mobile Blueprint using Visual Accessors" on page 29

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

- "Android Mobile App Example" on page 31
  - "Selecting the Mobile Device and App in the Device Manager" on page 32
  - "Creating a Mobile Configuration File" on page 36

- "Getting Access to a Repository" on page 41
- "Taking Images as Possible Smart Tags" on page 43
- "Creating the Smart Tag File" on page 45
- "Mobile Blueprint using Visual Accessors" on page 51



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

# Setting up the Environment

AIQ Mobile Designer requires an Appium server and a Device Manager to interact with mobile apps. The Device Manager functions as an emulator or bridge interface for a physical device connected to a computer.

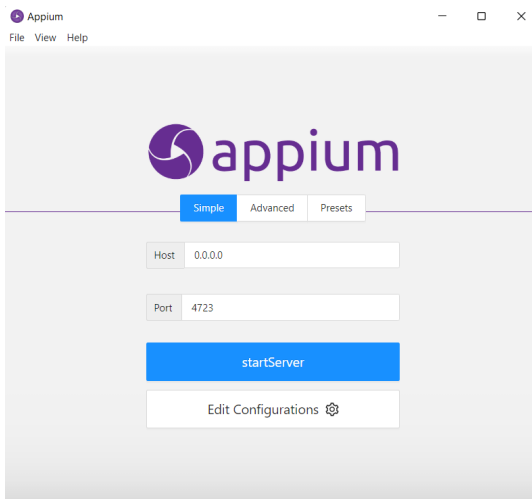
- "Appium Server Setup for Android" on the next page
- "Device Manager Setup" on page 15

# Appium Server Setup for Android

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 appropriate installer for your environment.
3. Launch the installer.



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

# Device Manager Setup

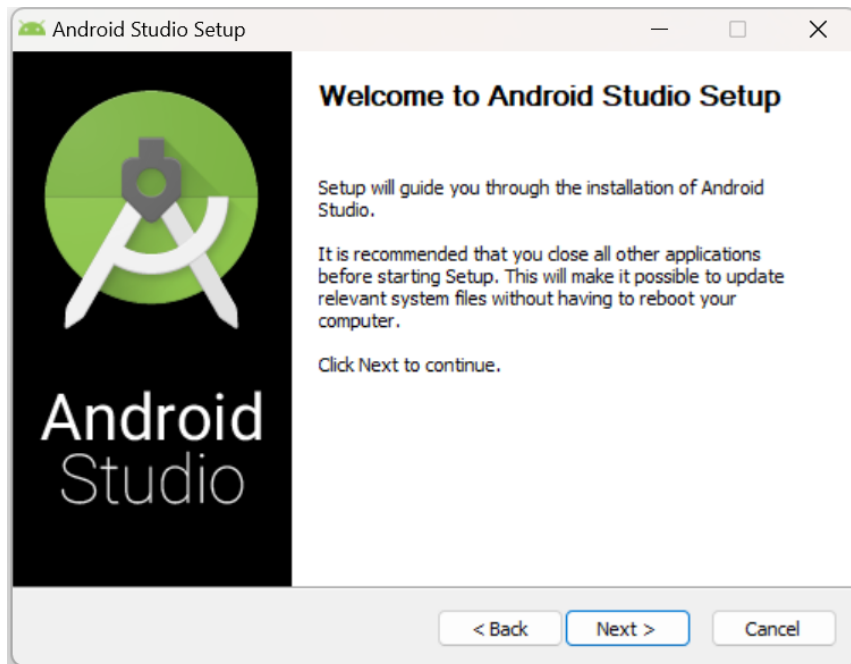
The Device Manager you use depends on your platform.

- Android applications can use Android Studio IDE. You can download it from: <https://developer.android.com/studio>.
- IOS applications use Xcode IDE. You can download it from, the Mac App Store or from <https://apps.apple.com/us/app/xcode/id497799835?mt=12>.

# Android Studio Setup

## Installing Android Studio

1. Download the appropriate installer for your environment from <https://developer.android.com/studio>.
2. Launch the installer.



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

## Configuring Android Studio



---

## Setting ANDROID\_HOME

Once Android Studio IDE is installed, add Global Environment Variable ANDROID\_HOME:

1. For Windows:

- a. Navigate to Environment Variables > Settings > System Variables > New.
- b. Add ANDROID\_HOME with value of `C://Users/Your_User-/Appdata/Local/Android/ SDK`



The above value assumes you accepted the default path during installation. If you specified a different path, enter that path for the value of ANDROID\_HOME.

2. For macOS:

- a. Open a Terminal and type: `echo export "ANDROID_HOME=/Users/yourName/Library/Android/sdk" >> ~/.bash_profile`
- b. Close Terminal and open a new one and type `echo $ANDROID_HOME` to verify that the value exists.

3. For Linux:

- a. Open a terminal in the same directory of installation and `sudo nano .bashrc`

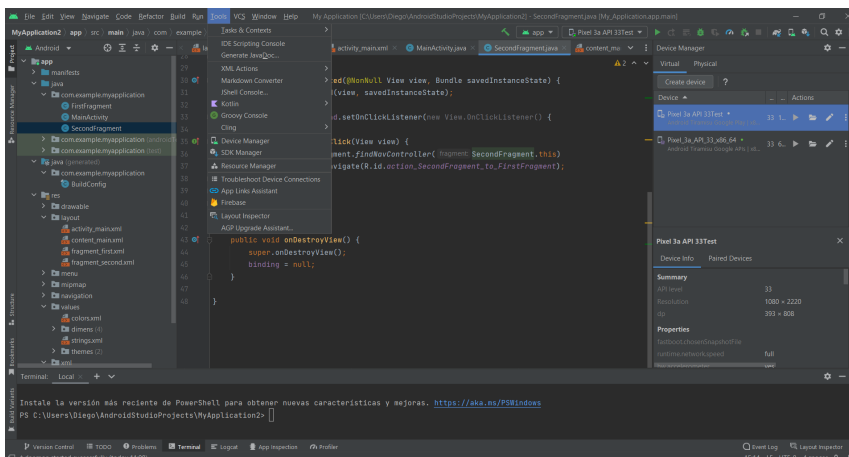
- b. Then add to the file: 

```
export ANDROID_HOME=  
E=${HOME}/Android/Sdk export PATH=  
H=${PATH}:${ANDROID_HOME}/platform-  
tools:${ANDROID_HOME}/tools
```
- c. Close the terminal and type in a new one `echo $ HOME`

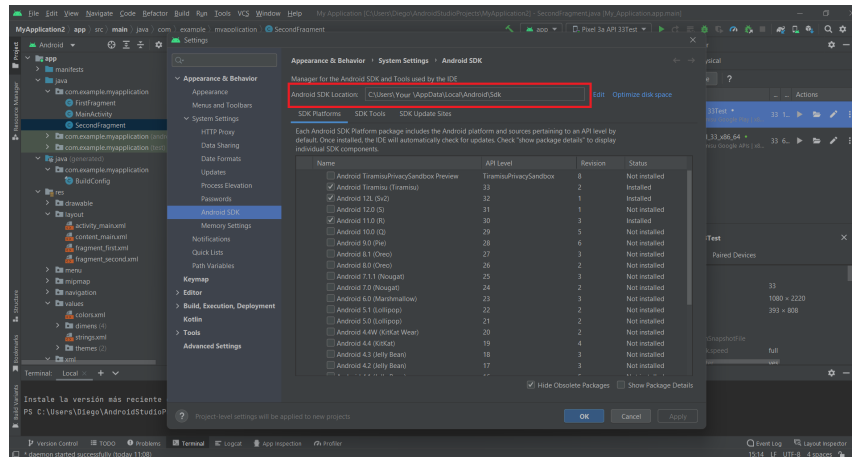
## Setting Android Debug Bridge Location

Android Debug Bridge (adb) is a command-line tool that lets you communicate with a device. The adb command facilitates a variety of device actions, such as installing and debugging apps. adb provides access to a Unix shell that you can use to run a variety of commands on a device.

1. Open Android Studio IDE.
2. In the **Tools** menu, select **SDK Manager**.



3. Copy the SDK Location. “path\_copied”



## 4. Copy the SDK Location. "path\_copied"

### a. For Windows:

- i. Open a cmd console
- ii. Type: `cd path_copied/platform-tools`
- iii. Type: `adb help`
- iv. Verify that it works.

### b. For Mac:

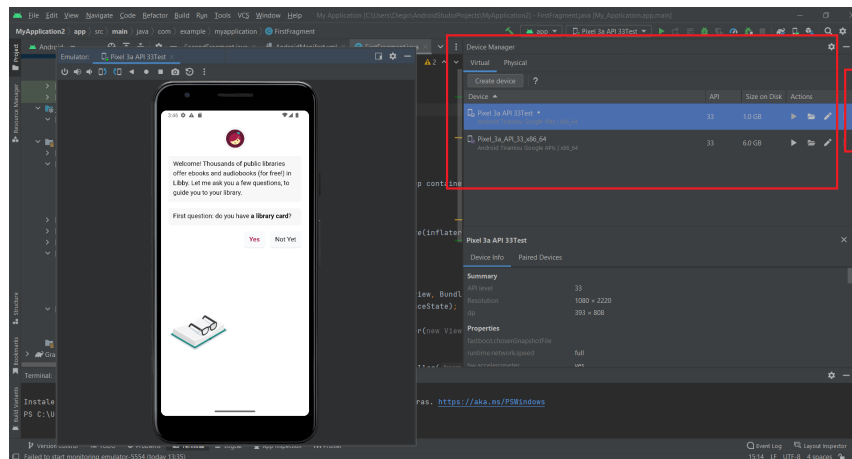
- i. Open terminal
- ii. Navigate to the directory: `~/Library/Android-  
/sdk/platform-tools/`

- iii. **Type:** `adb help`
  - iv. Verify that it works.
- c. For Linux:
- i. Perform the same command for the default directory in `~/Android/Sdk/platform-tools/adb` or a Terminal type `whereis adb` and look for the directory

# Adding App to Emulator

In Android Studio, the Device Manager interface helps you to configure an Emulator or Physical Device.

1. Choose between a Virtual or Physical Device.
2. Click **Create device** and follow the instructions.



3. When you finish the process, deploy the app by clicking the Play icon.
4. Install the app by using drag and drop from the apk or installing it from the Play Store. If you are using an emulator, be sure to install an Android version with Google Play Services.

## Required Fields

The minimum required parameters to set up a configuration file from the app are:

- appPackage
- appActivity
- udid
- platformVersion

The parameters can be obtained depending on the source of the app (self-app, downloaded from the app store)

Refer to "Creating a Mobile Configuration File" on page 36 for information on how to build a file configuration.

## Getting parameters from ADB

Parameters can be found using adb:

1. Navigate to the adb path. See "Android Studio Setup" on page 16 for information on the adb location.
2. Install the app in the emulator or physical device (the emulator can be downloaded from the app store or just drag and drop apk to the emulator).
3. Open the app in the Emulator/Physical Device
4. In the console/Terminal type: `adb shell pm list packages` to obtain the package name "appPackage" ( search for it in the list with a name similar to the app, for instance, `com.mobile.myapp`).

- 
5. For main activity "appActivity" type: `adb shell dumpsys package com.mobile.myapp`

Example result: **adb shell dumpsys package com.overdrive.mobile.android.libby**

Activity Resolver Table:

Schemes:

dewey-oauth:

3969bfc com.overdrive.mobile.android.libby/**com.overdrive.mobile.android.nautilus.ui.Activity\_Main** filter f2f78a6

Action: "android.intent.action.VIEW"

Category: "android.intent.category.DEFAULT"

Category: "android.intent.category.BROWSABLE" .....

extractNativeLibs=false

primaryCpuAbi=null

secondaryCpuAbi=null

cpuAbiOverride=null

versionCode=520000 minSdk=21 targetSdk=33

minExtensionVersions=[]

versionName=5.2.0

usesNonSdkApi=false

splits=[base]

apkSigningVersion=2

.....

UDID can be found just by typing: **adb devices**

Example: **adb devices**

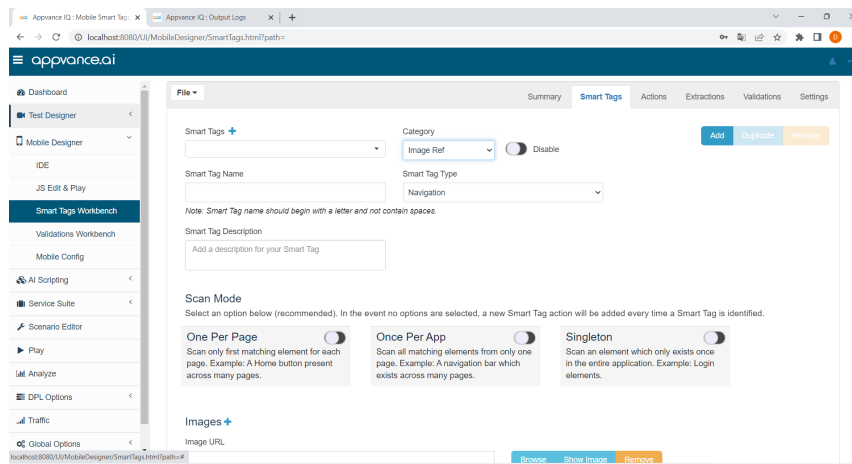
List of devices attached  
emulator-5554 device



# Creating a Smart Tag File for Visual Accessors

Visual accessors use a new type of Smart Tag called Image Reference. It can be provided from a URL or a repository.

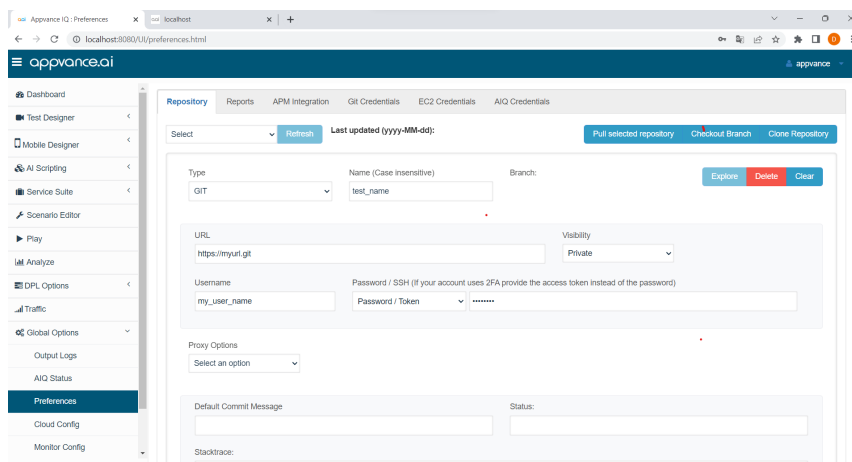
1. In AIQ navigate to **Mobile Designer > Smart Tags Workbench**.
2. In the **Category** field, select **Image Ref**.
3. Enter a **Smart Tag Name**.



4. Click on the plus next to Images.
5. You can enter a URL for the target in the **URL** field, or use the **Browse** button to upload an image from an existing repository.
6. 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 menu.

## Considerations

- Smart Tag images can have as many targets as necessary, just click on the plus icon next to **Images** to add more fields.
- if just a URL is provided it must be public access.
- If you are using a repository, it must be configured in AIQ, Navigate to **Global Options > Preferences > Repositories.**

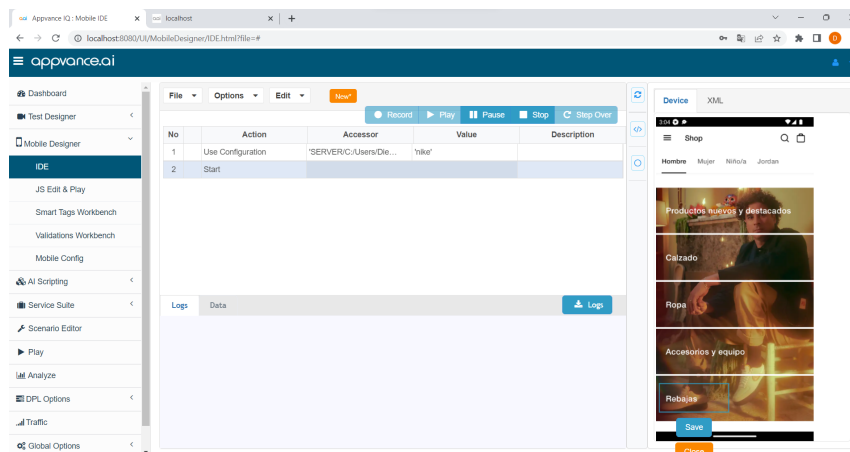


# Generating Images for Smart Tags

The Mobile Designer IDE allows you to play and record actions, besides cropping images from the device screen.

For taking images for your Smart Tags:

1. Open an emulator and start an Appium Server.
2. In AIQ navigate to **Mobile Designer**.
3. Click **Record**.
4. Load a file configuration. See Mobile Blueprint Enhancements for information on file configurations.
5. Click and navigate through the application and crop the images by drawing rectangles.



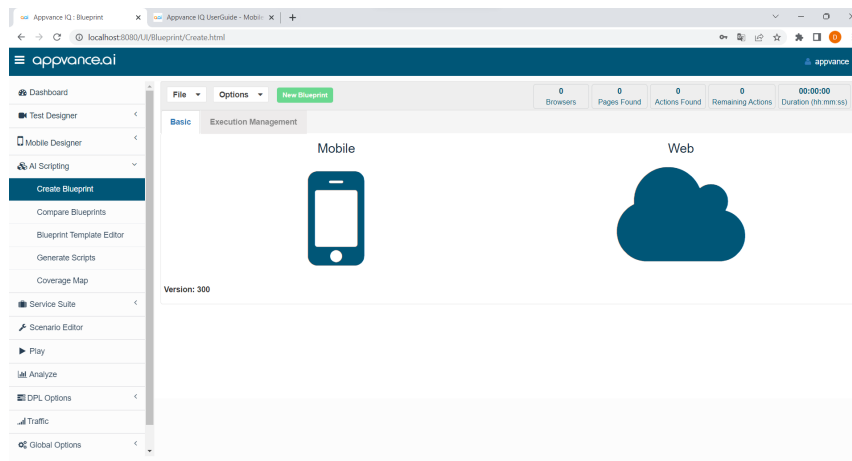
6. Click **Save** and save the images in your repository or local drive.

## Considerations

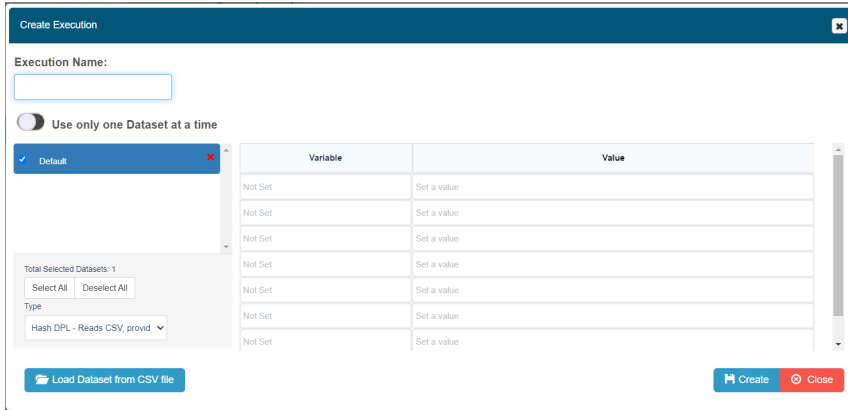
- It could be helpful to add multiple images from the same target taken using different methods at different sizes (preferably bigger than the original size).
- You can use the emulator to take a screenshot and crop images by using an image editor or just you could use basic cropping functionality from the OS.
- Images must be saved in a repository or a public URL.

# Creating a Mobile Blueprint using Visual Accessors

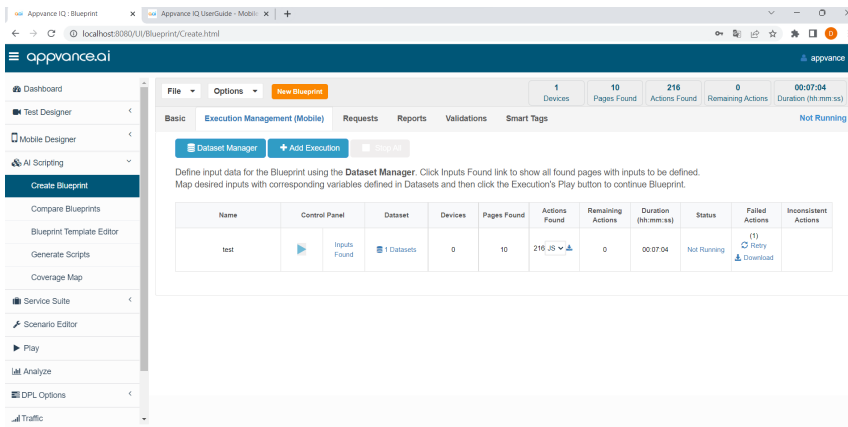
1. In Appvance IQ navigate to **AI Scripting** > **Create Blueprint**.
2. Click **Mobile** to create a mobile blueprint.



3. Load your device configuration file. Click **Browse**, search for the file and load it.
4. In **Scan Type** choose **Smart Tags** or **Smart Tags + Tags** and click **Next**.
5. Create a new execution and give it a name.
6. After clicking **Create**, the Blueprint will be running.



7. Once the Blueprint finishes, the execution results display.



# Android Mobile App Example

The following sections provide a complete end-to-end example of using the visual accessor functionality to test a mobile application.

The Android version of the Universe mobile app by Alpha Software is used for the example.

# Selecting the Mobile Device and App in the Device Manager

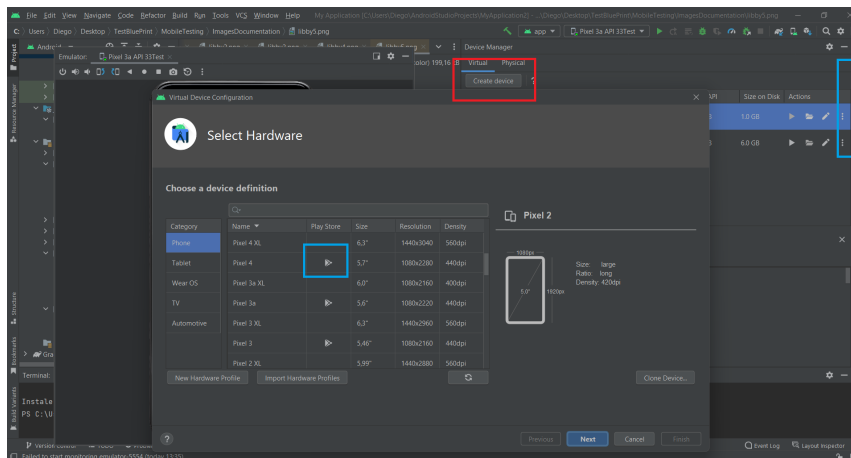
In this section you will select a mobile device to emulate and install the example mobile app. This example uses a Pixel 2a as the emulated mobile device.

1. Open Android Studio.
2. Click on **Device Manager > Create device > Virtual**.
3. Select **Category > Phone**.

In the Name column select Pixel 2a with Play Store 5.6" Resolution: 108 2210, 440dp.



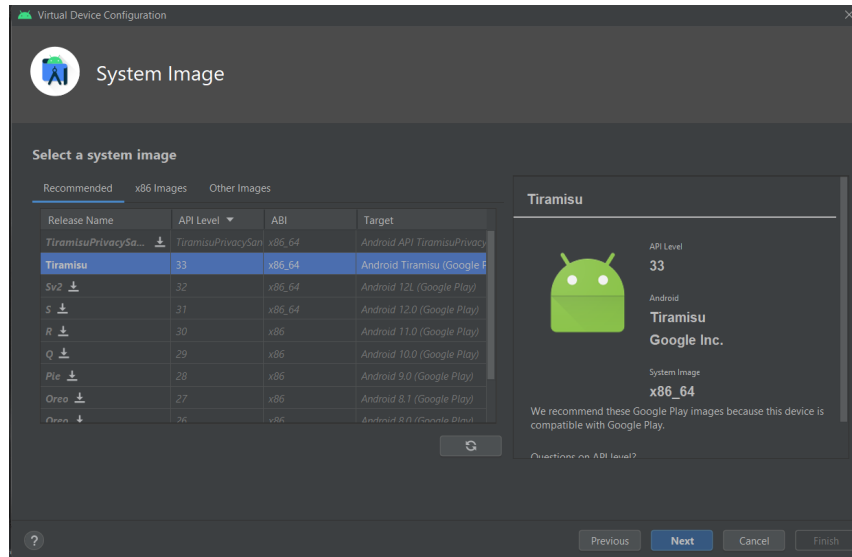
Note: Make sure the device has a Play Store icon (blue square) associated with it.



4. Click **Next**.
5. In the **Release Name** option select Tiramisu and click **Next**.

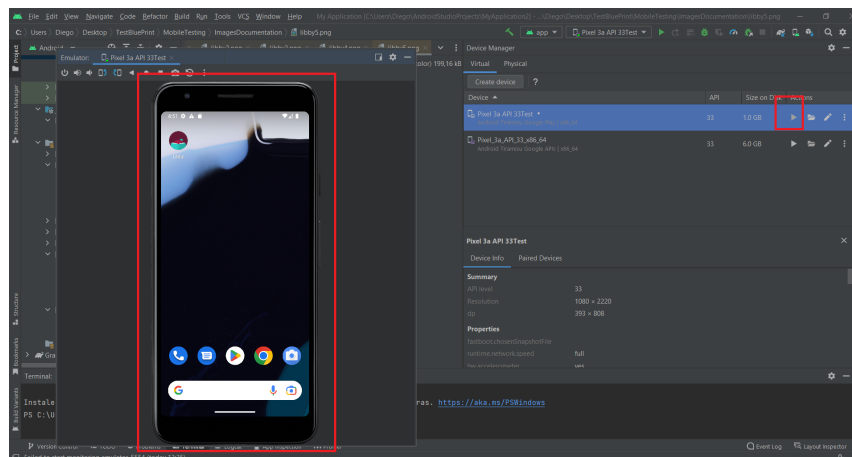


## Selecting the Mobile Device and App in the Device Manager



Note: The Android version of the emulator displays in the Device Manager.

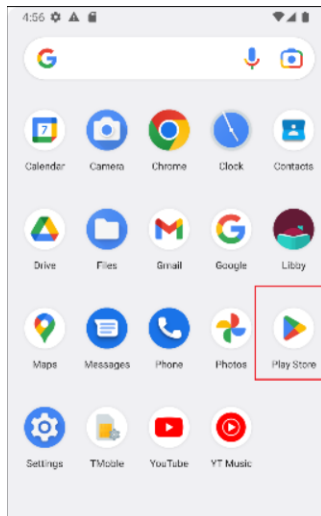
6. Accept the default settings. Be sure that Portrait is selected.
7. Click **Finish**.
8. Start the emulator by clicking the **Play** button.





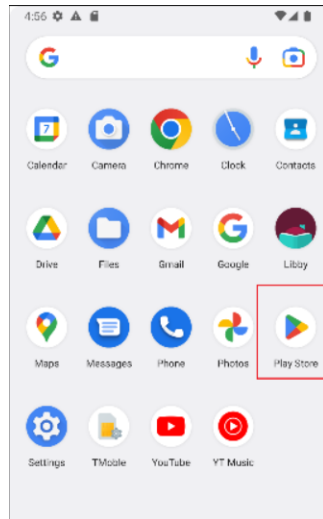
Note: Device Manager, Play button, and Emulator are highlighted in red.

9. Click at the center of the phone emulator and drag it up to show the available apps. Choose the Play Store.



Note: The emulated phone menu, Play Store highlighted in enclosed in red square.

10. Search Play Store for the Universe app by Alpha Software and Install it.

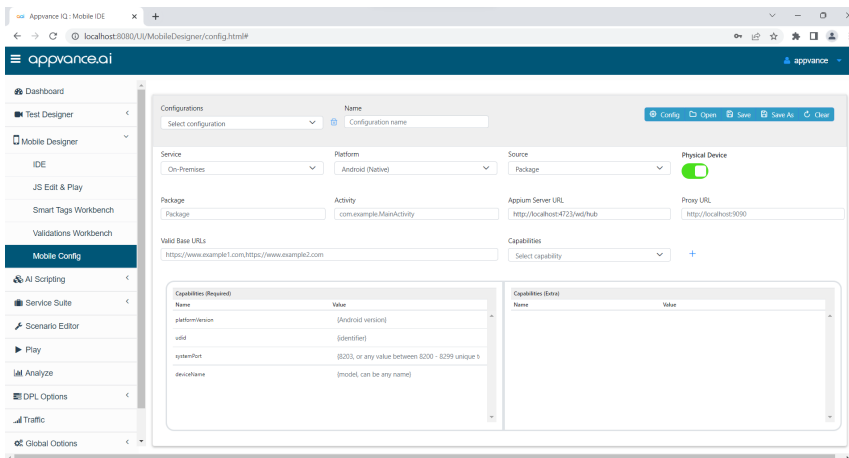


You are now ready for the next step, "Creating a Mobile Configuration File" on the next page.


# Creating a Mobile Configuration File

A Mobile Configuration file is required to describe which app will be launched during the Play/Recording/Blueprint.

1. In AIQ navigate to **Mobile Designer > Mobile Configuration**.



2. The first step is to get the basic parameters from the Device. Open a Console/Terminal in the adb directory.

 See "Setting up the Environment" on page 13 for information on how to get the directory if a variable path is not assigned to it.

3. Open the Universe app in the Device Emulator and perform the following steps:

- a. **UDID: adb devices**

List of devices attached  
emulator-5554 device

So: UDID=emulator-5554

b. appPackage: **adb shell pm list packages**

```
package:com.android.traceur
package:com.android.contacts
package:com.google.android.apps.messaging
package:com.ed.universe
package:com.android.internal.emulation.pixel_3a_xl
package:com.android.location.fused
package:com.android.vpndialogs
```

It returns a list with active packages, and searches for the app name, in this case, Universe (appPackage=**com.ed.universe**).

c. Activity/appActivity: **adb shell dumpsys package com.ed.universe**

Activity Resolver Table:

Non-Data Actions:

android.intent.action.MAIN:

237c878 com.ed.universe/.**MainActivity** filter 5a51b51

Action: "android.intent.action.MAIN"

Category: "android.intent.category.LAUNCHER"

.DayInDetailActivity:

7457fb6 com.ed.universe/com.edu.sundroid.DayInDetailActivity filter 24504b7

Action: ".DayInDetailActivity"

Category: "android.intent.category.DEFAULT"

versionCode=37 minSdk=21 targetSdk=32

minExtensionVersions=[]

versionName=1.7.4

usesNonSdkApi=false

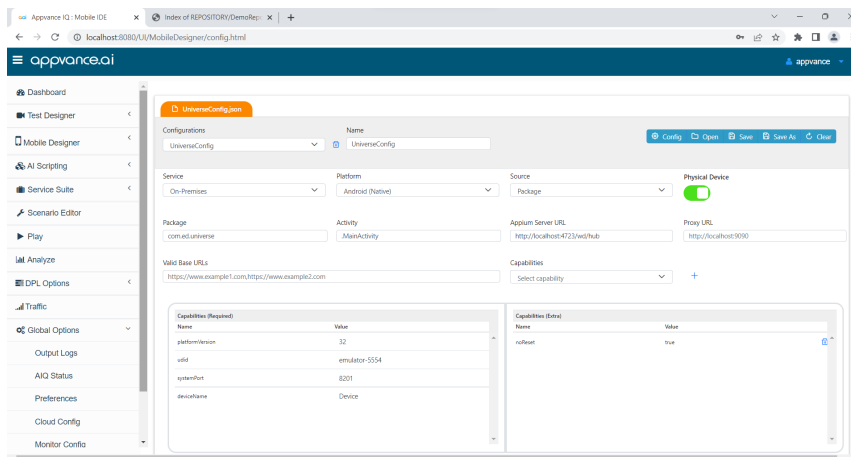
splits=[base, config.arm64\_v8a, config.en, config.xxhdpi]

apkSigningVersion=3

appActivity=**MainActivity**

Also, AndroidVersion appears as targetSDK = 32 ( it can be an upper version if OS is supporting it)

4. Now fill the Mobile Configuration Panel with the parameters
  - a. Name the configuration.
  - b. In **Source** select **Package**.
  - c. In Capabilities add "noReset" : "true" to prevent the app from asking repeatedly to log in.
  - d. Accept the default parameters for the remaining fields.
  - e. Save the configuration as "UniverseConfiguration".



## Verifying the Mobile Configuration File

The file configuration can be opened in text format:

```
[
  {
    "configName": "UniverseConfig",
    "app": "",
    "appPackage": "com.ed.universe",
    "proxyURL": "",
    "bundleId": "",
    "source": "package",
    "appActivity": ".MainActivity",
    "deviceMode": true,
    "service": "On-Premises",
    "serverURL": "http://localhost:4723/wd/hub",
    "platformName": "Android",
    "validDomains": "",
    "capabilities": {
      "platformVersion": "33",
      "udid": "emulator-5554",
      "systemPort": "8201",
      "deviceName": "Device",
      "noReset": "true"
    }
  }
]
```

## Testing the Configuration

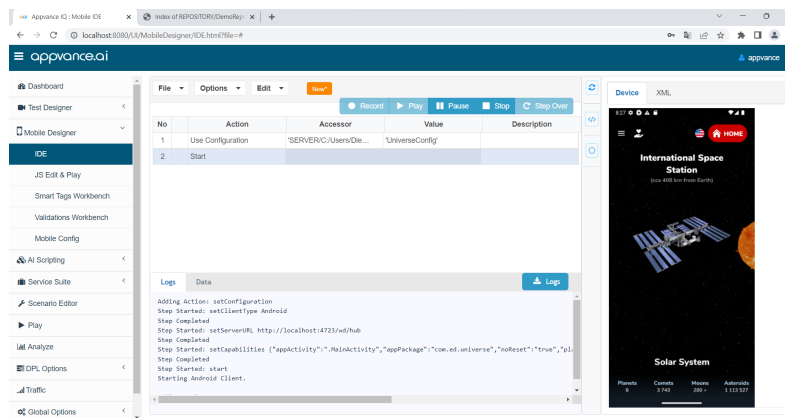
For testing, if the configuration is correct,

1. Start an Appium Server and an active emulator.
2. In Appvance IQ navigate to **Mobile Designer > IDE**.
3. In the **File** dropdown select **New**.

4. Click **Record**.

5. Add the configuration you created. Wait until the app is launched.

- If there is a problem with the configuration, it will display in the log or the Appium Inspector. Use the log file to fix any configuration issues. Typically errors are related to incomplete names or copied parameters. Use the example file above to help.
- If there are no configuration errors, you will see the emulator with the Universe app running in AIQ.



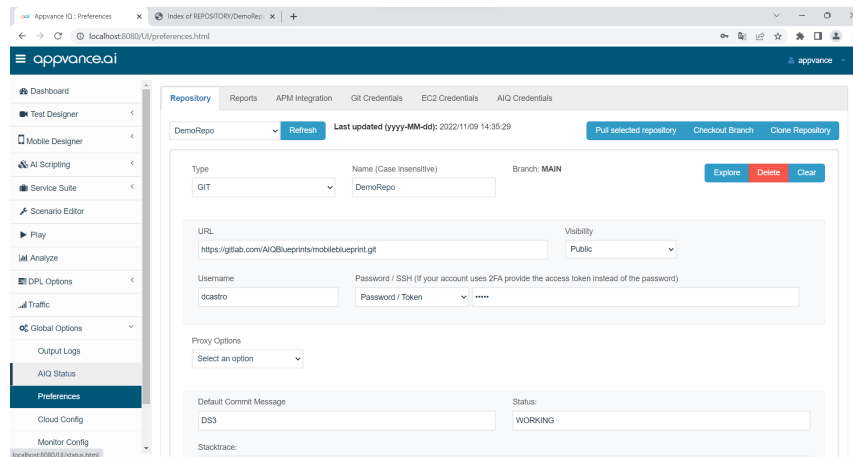
You are now ready to proceed to the next step "Getting Access to a Repository" on the facing page.



# Getting Access to a Repository

In order to save images Appvance IQ can connect to a repository.

## 1. In AIQ navigate to **Global Options > Preferences > Repository**.



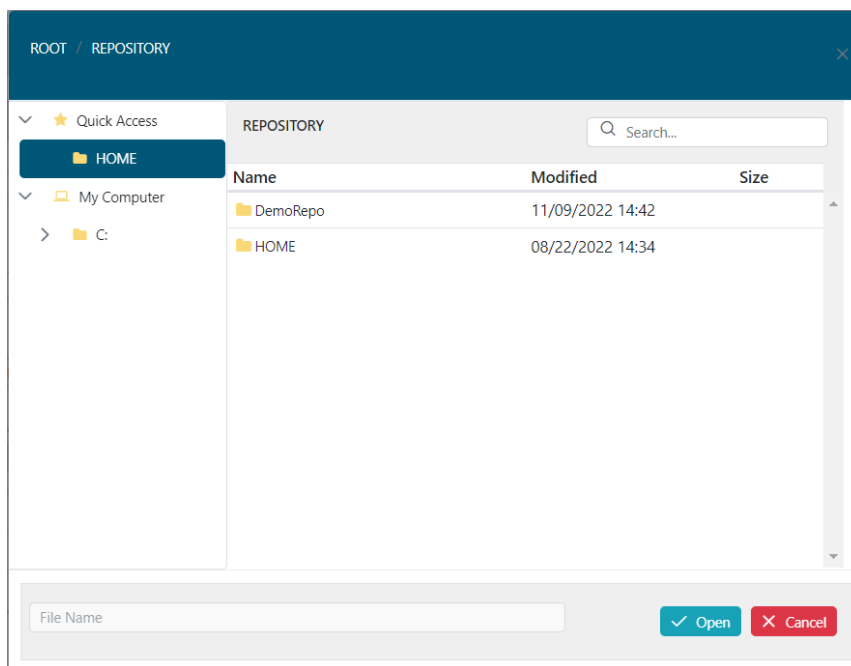
## 2. Suggested parameters:

- Name: DemoRepo
- URL: <https://gitlab.com/AIQBlueprints/mobileblueprint.git>
- Username: user name with access to the repository
- Password: user password, if not working please contact Mobile Designer Team for a Token
- Visibility: Public
- Type: GIT



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

3. Click **Clone Repository** (Repository Successfully saved if everything is correct, if not check your parameters )
4. Anytime a file or document is saved or loaded it can be accessed from the repository in ROOT > REPOSITORY > name\_repo.



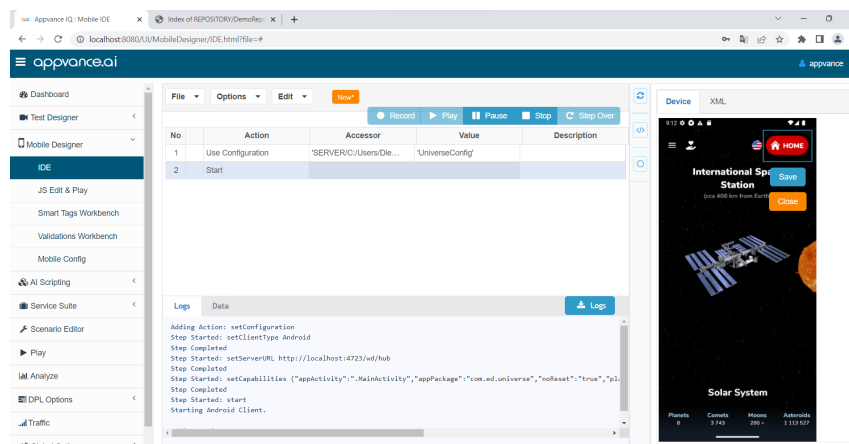
You are now ready to perform the next step, "Taking Images as Possible Smart Tags" on the facing page.

# Taking Images as Possible Smart Tags

Visual accessor functionality allows you to use graphical representations as possible Smart Tags. Images from the application can be obtained from Mobile Designer IDE in the Recording.

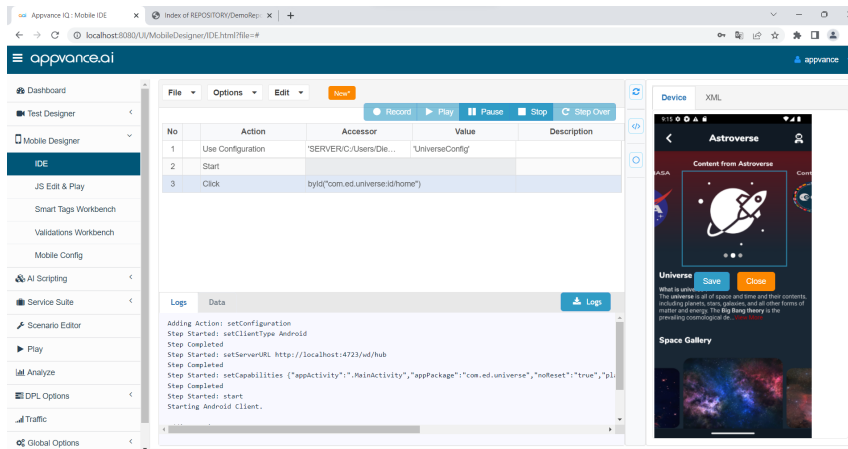
Follow these steps to get some images from the Universe app.

1. Start an Appium Server and an Emulator.
2. In AIQ navigate to **Mobile Designer > IDE**.
3. Click **Record** and load the configuration file that you created in "Creating a Mobile Configuration File" on page 36.
4. Wait until it loads and then draw a square around the HOME button.



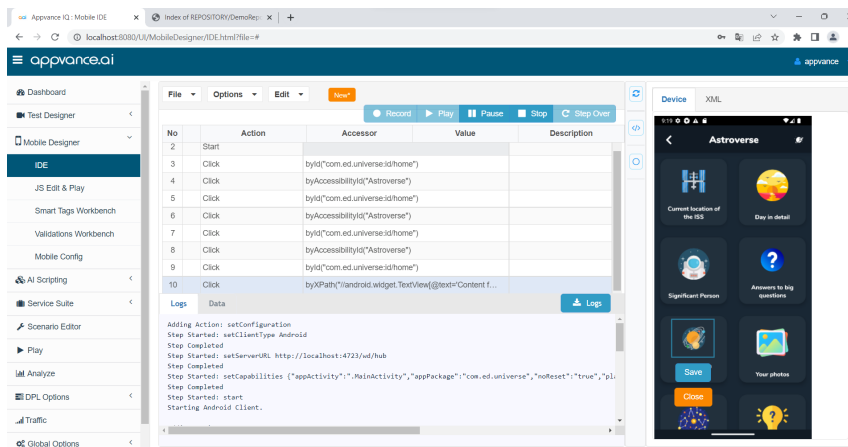
5. Click **Save** and search for **ROOT > REPOSITORY > DemoRepo > Targets** and save it as "UniverseHome".
6. Click on Home.

7. Draw a square around the Rocket image and save it as "UniverseRocket".



8. Click on **Content from Astroverse** located above Rocket Image.

9. Draw a square around the Solar System image and save it as "UniverseSolarSystem".



You are now ready for the next step, "Creating the Smart Tag File" on the facing page

# Creating the Smart Tag File

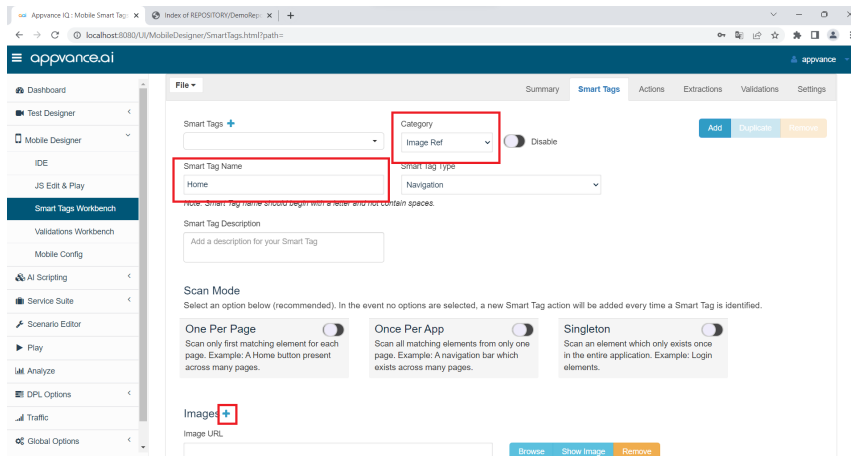
The following process demonstrates creating a Smart Tag File for the Universe app.

**Note:** You must have a connection to the repository to perform this task. See "Getting Access to a Repository" on page 41.

1. In AIQ navigate to **Mobile Designer > Smart Tags Workbench**.

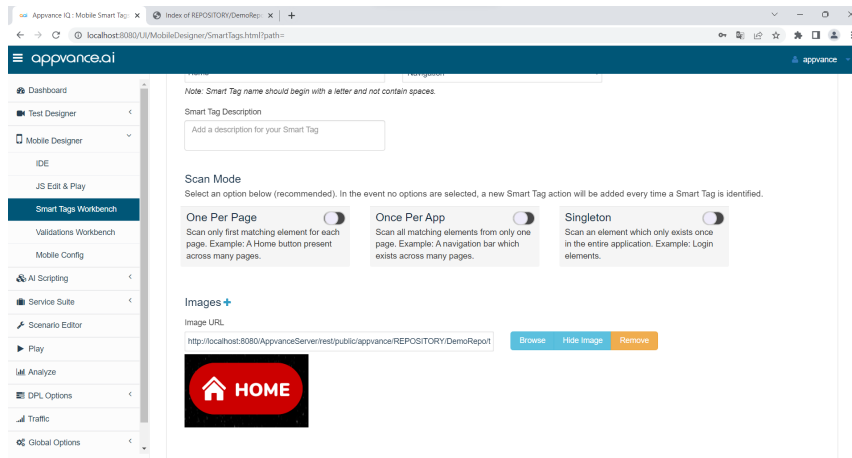
2. Define the following parameters:

- **Smart Tags Name** : Home
- **Category** : ImageRef
- **Smart Tag Type** : Navigation

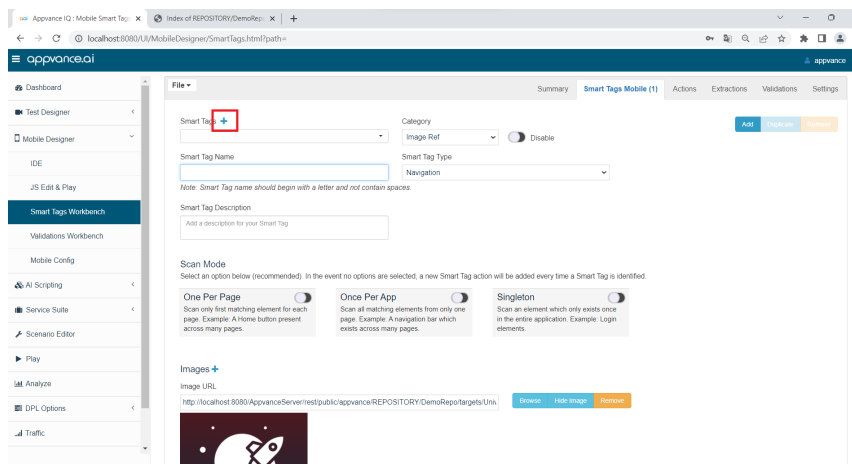


3. Click on the plus sign next to **Images**.

4. Click **Browse** and search in ROOT > REPOSITORY> DemoRepo > targets > UniversalHome.
5. Click **Show Image** to check if the correct image is loaded.

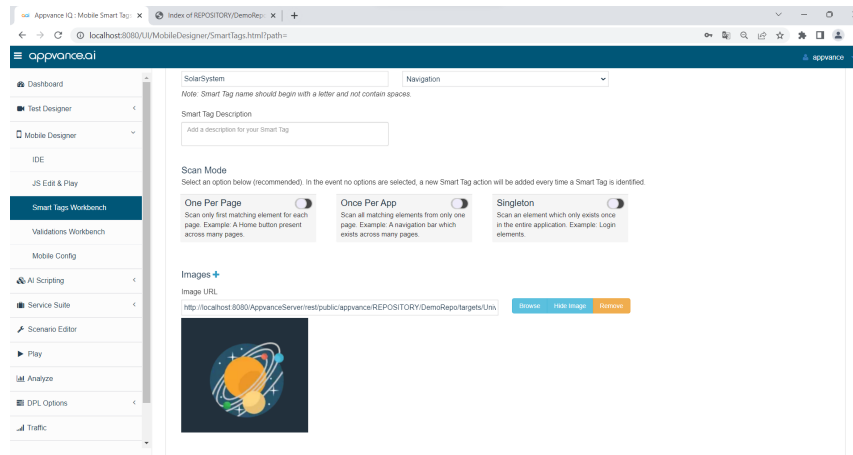


6. Click **Add**.
7. Click on the plus sign next to **Smart Tags**.



8. Create a new Smart Tag named "Rocket" in the **Smart Tags Name** field.

9. Click the plus sign next to **Images** and search for "UniverseRocket" in the repository.
10. Click **Add**.
11. Create a new SmartTag named "SolarSystem" and browse for the "UniverseSolarSystem" image.



12. Click **Add**.
13. In the File dropdown, select **Save** and save it as "UniverseSmarttagsDemo".
14. The saved file has the following structure:

```

{
  "categoryId": "ImageRef",
  "threshold": "0.75",
  "smartTags": [
    {

```

```
loc-
alhost:8080/Api-
vanceServer-
/rest/pub-
lic/api-
vance/REPOSITORY/DemoRepo/targets/universeHome.png"
    ],
    "rules": [],
    "info": {},
    "disabled": false,
    "description": "",
    "singleton": false,
    "scanOncePerApp": false,
    "scanOnePerPage": false
  },
  {
    "name": "Rocket",
    "category": "ImageRef",
    "type": "Navigation",
    "tags": [
      "http://-
loc-
alhost:8080/Api-
```



```

pvanceServer-
/rest/pub-
lic/ap-
pvance/REPOSITORY/DemoRepo/targets/UniverseRocket.png"
    ],
    "rules": [],
    "info": {},
    "disabled": false,
    "description": "",
    "singleton": false,
    "scanOncePerApp": false,
    "scanOnePerPage": false
},
{
    "name": "SolarSystem",
    "category": "ImageRef",
    "type": "Navigation",
    "tags": [
    "http://-

loc-
alhost:8080/Ap-
pvanceServer-
/rest/pub-
lic/ap-
pvance/REPOSITORY/De-
moRepo/targets/UniverseSolarSystem.png"
    ],
    "rules": [],
    "info": {},

```

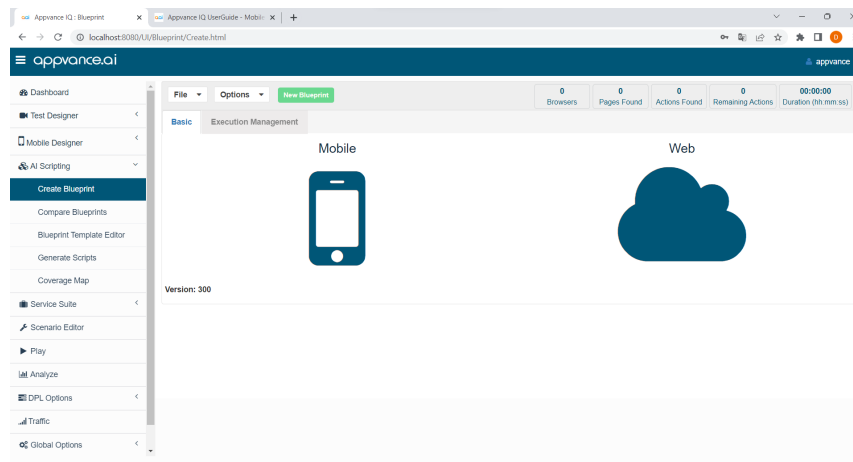
```
        "disabled": false,  
        "description": "",  
        "singleton": false,  
        "scanOncePerApp": false,  
        "scanOnePerPage": false  
    }  
  
    ],  
    "actions": [],  
    "customTags": [],  
    "validators": [],  
    "extractors": []  
}
```

You are now ready for the next step "Mobile Blueprint using Visual Accessors" on the facing page.

# Mobile Blueprint using Visual Accessors

This section demonstrates creating a Mobile Blueprint using visual accessors for the Universe app.

1. Start an Appium Server and an Emulator with the Universe app installed.
2. Open the app in the Emulator to verify everything is working correctly.
3. In AIQ navigate to **AI Scripting > Create Blueprint**.




4. Click **Mobile**.
5. Select the configuration file for your device. Click **Mobile Configuration > Browse** and load "UniverseConfiguration".

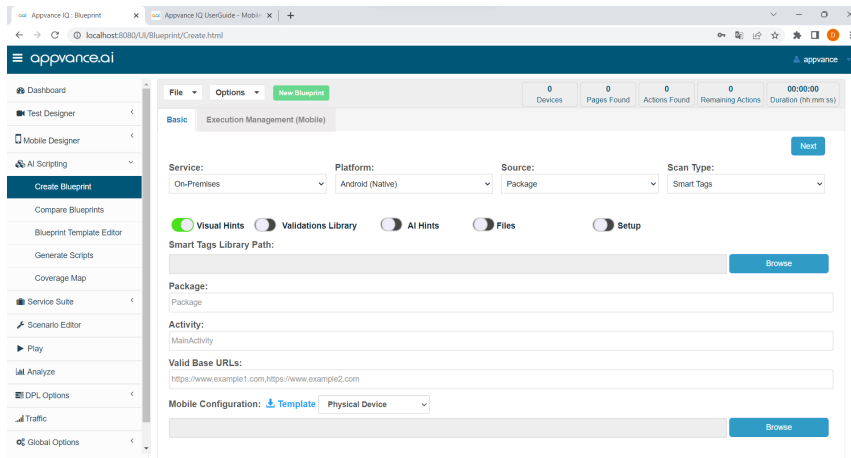


"UniverseConfiguration". is the file you created in "Creating a Mobile Configuration File" on page 36.

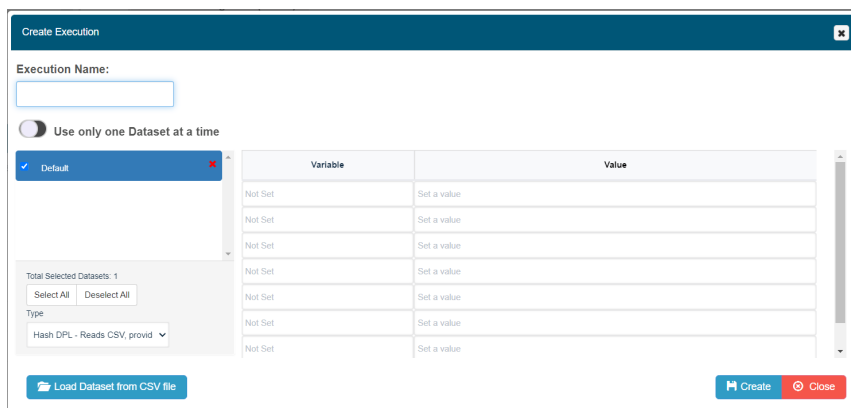
6. In **Scan Type** select **Smart Tags**.

7. Click **Browse** next to **Smart Tags Library Path** and load the "UniverseSmartTagsDemo" file.

 "UniverseSmartTagsDemo" is the file you created in "Creating the Smart Tag File" on page 45.

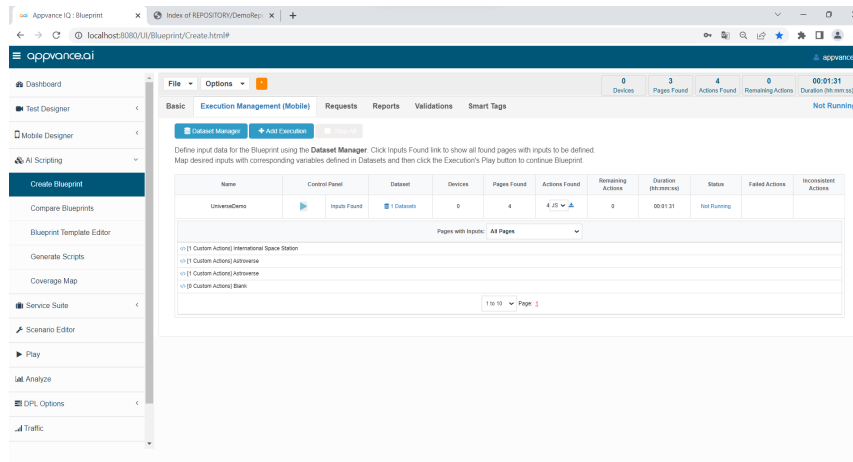


8. Click **Next**.
9. Create a new execution with the name "UniverseDemo".
10. Click **Create** The Blueprint will run.

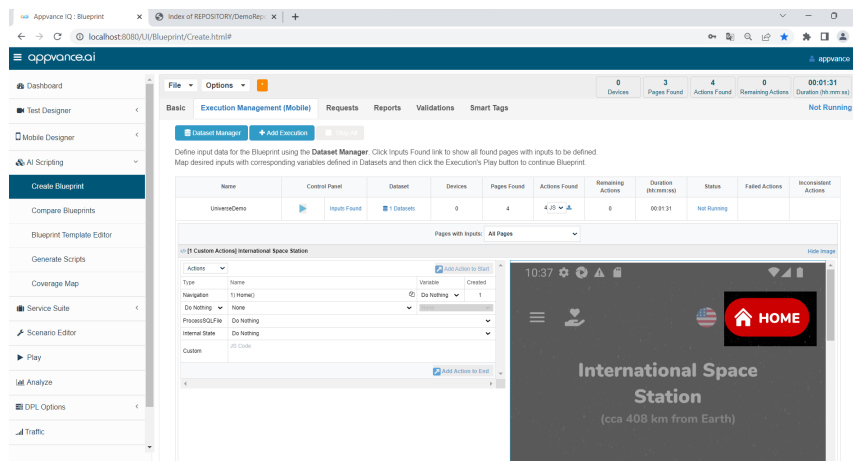


11. To view the search while the Blueprint is running refer to the Emulator.

12. Result : List of pages and actions found.

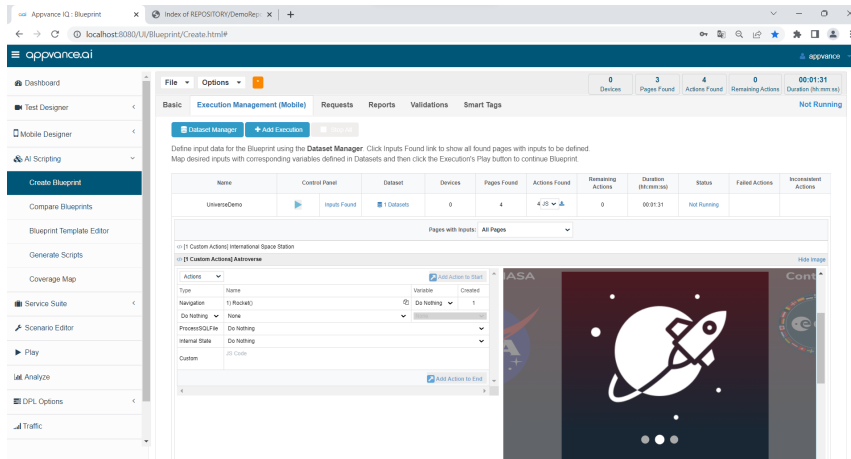


13. Result : Home Smart Tag found.

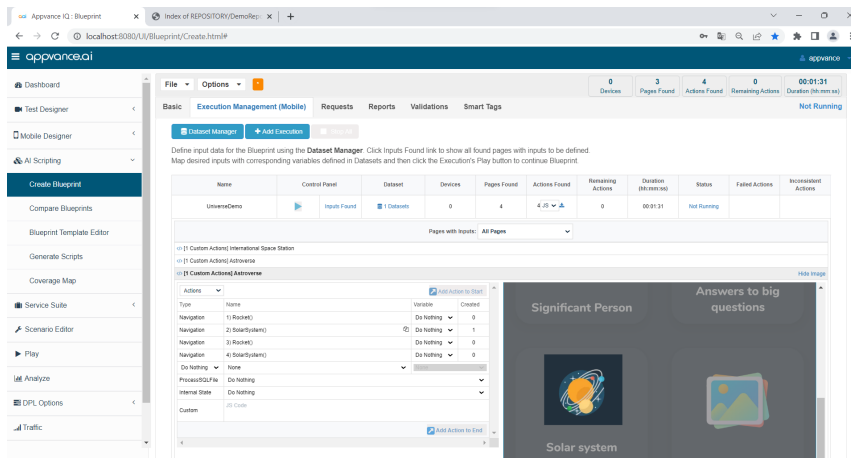


14. Result : Rocket Smart Tag found.

## Chapter 4



15. Result : Solar System Smart Tag found.



16. Save the results "Actions Found" and in **File** save the Blueprint.