appium / appium-xcuitest-driver

Watch 25 Star 65 Fork 39

Code Issues 24 Pull requests 2 Projects 0 Wiki Pulse Graphs

Appium iOS driver, backed by Apple XCUITest

496 commits 4 branches 46 releases 14 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

jlipps 2.3.0 Latest commit 10ba615 2 days ago

WebDriverAgent @ 8ff47c3	bump WDA to latest master	3 days ago
docs	Update WDA	3 months ago
lib	fix eslint	2 days ago
test	fix eslint	2 days ago
.eslintignore	Update eslint and fix problems	5 months ago
.eslintrc	Update eslint and fix problems	5 months ago
.gitignore	Allow use of xcconfig file for signing WDA	2 months ago
.gitmodules	Update system to get tests running	5 months ago
.jscsrc	initial commit	a year ago
.jshintrc	initial commit	a year ago
.npmignore	Allow use of xcconfig file for signing WDA	2 months ago
.travis.yml	Updating NodeJS versions in travis	10 days ago
LICENSE	Update eslint and fix problems	5 months ago
README.md	Add driver capability to prevent XCTest attachments creation	4 days ago
gulpfile.js	Add provisional support for real devices	4 months ago
index.js	Make sure that iOS 10 passes the tests	4 months ago
package.json	2.3.0	2 days ago

README.md

=# appium-xcuitest-driver

Missing functionality

- Setting geo location (https://github.com/appium/appium/issues/6856)
- Auto accepting alerts (https://github.com/appium/appium/issues/6863)
- Touch Actions

Known issues

- Unable to interact with elements on devices in Landscape mode (https://github.com/appium/appium/issues/6994)
- `shake` is not implemented due to lack of support from Apple
- `lock` is not implemented due to lack of support from Apple
- Setting geo-location not supported due to lack of support from Apple
- Through multi action API, `zoom` works but `pinch` does not, due to Apple issue.

External dependencies

In addition to the git submodules mentioned below (see Development), this package currently depends on `libimobiledevice` to do certain things. Install it with Homebrew,

```
brew install ideviceinstaller
```

There is also a dependency, made necessary by Facebook's WebDriverAgent, for the Carthage dependency manager. If you do not have Carthage on your system, it can also be installed with Homebrew

```
brew install carthage
```

`ideviceinstaller` doesn't work with iOS 10 yet. So we need to install ios-deploy

```
npm install -g ios-deploy
```

On some systems the default logger, `idevicesyslog`, does not work. You can install `deviceconsole` and specify its path with the `realDeviceLogger` capability (**note:** This path should be the path to the *executable* installed by the below command. It will be the directory created by the below command, followed by `/deviceconsole`).

```
npm install -g deviceconsole
```

For real devices we can use xcpretty to make Xcode output more reasonable. This can be installed by

```
gem install xcpretty
```

Sim Resetting

By default, this driver will create a new iOS simulator and run tests on it, deleting the simulator afterward.

If you specify a specific simulator using the `udid` capability, this driver will boot the specified simulator and shut it down afterwards.

If a udid is provided and the simulator is already running, this driver will leave it running after the test run.

In short, this driver tries to leave things as it found them.

You can use the `noReset` capability to adjust this behavior. Setting `noReset` to `true` will leave the simulator running at the end of a test session.

Real devices

Configuration

The `appium-xcuitest-driver` has provisional support for iOS real devices. Not all functionality is currently supported.

WebDriverAgent needs to be built with `development team` and `provisioning profile` installed on device. The easiest way to do this is to specify them in an xcconfig file, the path to which is passed in to the system using th `xcodeConfigFile` desired capability

```
DEVELOPMENT_TEAM = <Team ID>
CODE_SIGN_IDENTITY = iPhone Developer
```

The Team ID is a unique 10-character string generated by Apple that is assigned to your team. You can find your Team ID using your developer account (Sign in to developer.apple.com/account, and click Membership in the sidebar. Your Team ID appears in the Membership Information section under the team name.).

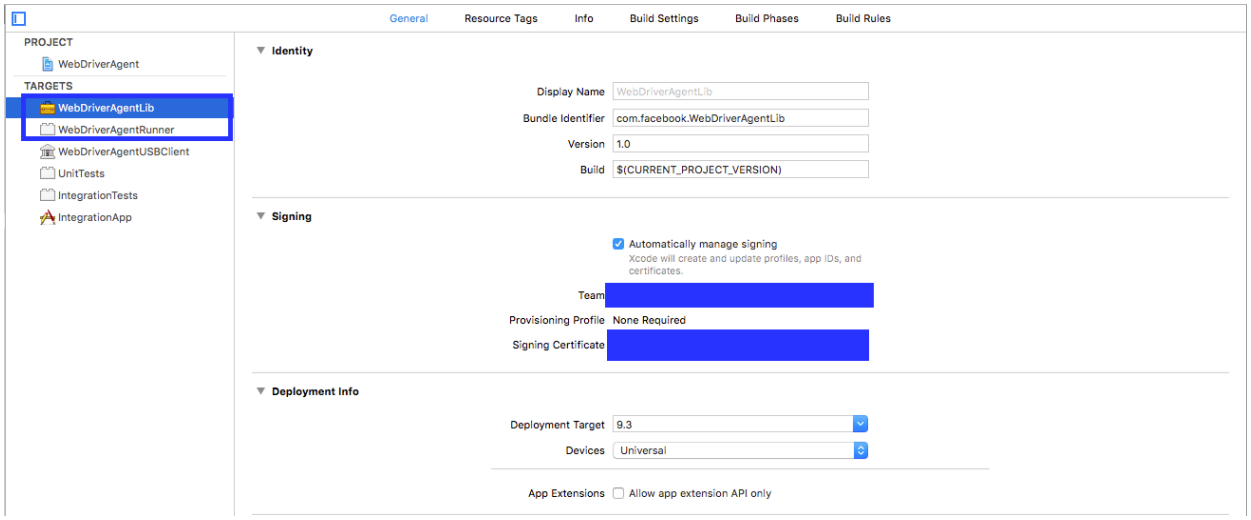
Manual configuration alternative

Alternatively, the profile can be manually associated with the project (keep in mind that this will have to be done each time the WebDriverAgent is updated):

- Open terminal go to `node_modules/appium-xcuitest-driver/WebDriverAgent` (this path is relative to your appium installation).

```
mkdir -p Resources/WebDriverAgent.bundle
sh ./Scripts/bootstrap.sh -d
```

- Open `WebDriverAgent.xcodeproj` in Xcode. Select your `development team` for **both** the `WebDriverAgentLib` and `WebDriverAgentRunner` targets. This should also auto select `Signing Ceritificate`. The outcome should look as shown below.



- Build `WebDriverAgent` once to verify all above steps worked.

```
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -destination 'id=<udid>' test
```

Last line on build output above command should be `Listening on USB`. Then you are all set!

Internally it also expects `idevicesyslog` to be installed. For iOS 10 you need to install it like this `brew install libimobiledevice --HEAD` and for iOS 9 `brew install libimobiledevice`

Known problems

Logger not working

If the system stops with log output like

```
[XCUITest] Waiting for WebDriverAgent to start on device
[debug] [XCUITest] Log file for xcodebuild test: /Users/user/Library/Developer/Xcode/DerivedData/WebDriverAgent-dmeyh
```

The culprit is usually the real device logger. By default the system uses `idevicesyslog`, which is installed with `libimobiledevice`, but on some machines this does not work. You can test by running `idevicesyslog` in a terminal window. If it fails, you can use `deviceconsole`, specifying the full path to the program with the `realDeviceLogger` capability.

Weird state

Note: Running WebDriverAgent tests on a real device is particularly flakey. If things stop responding, the only recourse is, most often, to restart the device. Logs in the form of the following *may* start to occur:

```
info JSONWP Proxy Proxying [POST /session] to [POST http://10.35.4.122:8100/session] with body: {       "desiredCapab
dbug WebDriverAgent Device: Jul 26 13:20:42 iamPhone XCTRunner[240]      <Warning>: Listening on USB
dbug WebDriverAgent Device: Jul 26 13:21:42 iamPhone XCTRunner[240]      <Warning>: Enqueue Failure: UI Testing Failu
dbug WebDriverAgent Device: Jul 26 13:21:57 iamPhone XCTRunner[240]      <Warning>: Enqueue Failure: UI Testing Failu
dbug WebDriverAgent Device: Jul 26 13:22:57 iamPhone XCTRunner[240]      <Warning>: Enqueue Failure: UI Testing Failu
```

Real device security settings

On some systems there are Accessibility restrictions that make the `WebDriverAgent` system unable to run. This is usually manifest by `xcodebuild` returning an error code `65`. A workaround for this is to use a private key that is not stored on the system keychain. See this issue and this Stack Exchange post

To export the key, use

```
security create-keychain -p [keychain_password] MyKeychain.keychain
security import MyPrivateKey.p12 -t agg -k MyKeychain.keychain -P [p12_Password] -A
```

where `MyPrivateKey.p12` is the private development key exported from the system keychain.

The full path to the keychain can then be sent to the Appium system using the `keychainPath` desired capability, and the password sent through the `keychainPassword` capability.

Desired Capabilities

Should be the same for Appium

Differences noted here

Capability	Description	Values
`noReset`	Do not destroy or shut down sim after test. Start tests running on whichever sim is running, or device is plugged in. Default `false`	`true`, `false`
`processArguments`	Process arguments and environment which will be sent to the WebDriverAgent server.	`{ args: ["a", "b", "c"], env: { "a": "b", "c": "d" } }` or `'{"args": ["a", "b", "c"], "env": { "a": "b", "c": "d" }}'`

Capability	Description	Values
`wdaLocalPort`	This value if specified, will be used to forward traffic from Mac host to real ios devices over USB. Default value is same as port number used by WDA on device.	e.g., `8100`
`showXcodeLog`	Whether to display the output of the Xcode command used to run the tests. If this is `true`, there will be **lots** of extra logging at startup. Defaults to `false`	e.g., `true`
`realDeviceLogger`	Device logger for real devices. It could be path to `deviceconsole` (installed with `npm install deviceconsole`, a compiled binary named `deviceconsole` will be added to `./node_modules/deviceconsole/`) or `idevicesyslog` (comes with libimobiledevice). Defaults to `idevicesyslog`	`idevicesyslog`, `/abs/path/to/deviceconsole`
`iosInstallPause`	Time in milliseconds to pause between installing the application and starting WebDriverAgent on the device. Used particularly for larger applications. Defaults to `0`	e.g., `8000`
`xcodeConfigFile`	Full path to an optional Xcode configuration file that specifies the code signing identity and team for running the WebDriverAgent on the real device.	e.g., `/path/to/myconfig.xcconfig`
`keychainPath`	Full path to the private development key exported from the system keychain. Used in conjunction with `keychainPassword` when testing on real devices.	e.g., `/path/to/MyPrivateKey.p12`
`keychainPassword`	Password for unlocking keychain specified in `keychainPath`.	e.g., `super awesome password`
`scaleFactor`	Simulator scale factor. This is useful to have if the default resolution of simulated device is greater than the actual display resolution. So you can scale the simulator to see the whole device screen without scrolling.	Acceptable values are: `'1.0', '0.75', '0.5', '0.33'` and `'0.25'`. The value should be a string.
`usePrebuiltWDA`	Skips the build phase of running the WDA app. Building is then the responsibility of the user. Only works for Xcode 8+. Defaults to `false`.	e.g., `true`

Capability	Description	Values
'preventWDAAttachments |Sets read only permissons to Attachments subfolder of WebDriverAgent root inside Xcode's DerivedData. This is necessary to prevent XCTest framework from creating tons of unnecessary screenshots and logs, which are impossible to shutdown using programming interfaces provided by Apple.|Setting the capability to true will set Posix permissions of the folder to 555 and false will reset them back to 755`|		

Development

This project has git submodules!

Clone with the `git clone --recursive` flag. Or, after cloning normally run `git submodule init` and then `git submodule update`

The `git diff --submodule` flag is useful here. It can also be set as the defaul `diff` format: `git config --global diff.submodule log`

`git config status.submodulesummary 1` is also useful.

Watch

```
npm run watch
```

Test

```
npm test
```

WebDriverAgent Updating

Updating FaceBook's WebDriverAgent is as simple as running updating the submodule and then committing the change:

```
git checkout -b <update-branch-name>
git submodule update --remote
git add WebDriverAgent
git commit -m "Updating upstream WebDriverAgent changes"
```

There is a chance that the update changed something critical, which will manifest itself ć `xcodebuild` throwing errors. The easiest remedy is to delete the files, which are somewhere like `/Users/isaac/Library/Developer/Xcode/DerivedData/WebDriverAgent-eoyoecqmiqfeodgstkwbxkfyagll`. This is also necessary when switching SDKs (e.g., moving from Xcode 7.3 to 8).