



# Silk Test 19.0

Testing Mobile Applications

**Micro Focus**  
**The Lawn**  
**22-30 Old Bath Road**  
**Newbury, Berkshire RG14 1QN**  
**UK**  
<http://www.microfocus.com>

**Copyright © Micro Focus 1992-2018. All rights reserved.**

**MICRO FOCUS, the Micro Focus logo and Silk Test are trademarks or registered trademarks of Micro Focus IP Development Limited or its subsidiaries or affiliated companies in the United States, United Kingdom and other countries.**

**All other marks are the property of their respective owners.**

**2018-06-06**


# Contents


<b>Testing Mobile Applications</b>	<b>4</b>
Android	4
Prerequisites for Testing Mobile Applications on Android	4
Testing Mobile Applications on Android	4
Testing Hybrid Applications on Android	5
Installing a USB Driver	6
Enabling USB-Debugging	7
Recommended Settings for Android Devices	7
Configuring the Android Emulator for Silk Test	7
Tested Configurations for Parallel Test Execution	9
iOS	11
Prerequisites for Testing Mobile Applications on iOS	11
Testing Native Mobile Applications on a Physical iOS Device	12
Testing Native Mobile Applications on an iOS Simulator	13
Testing Mobile Web Applications on a Physical iOS Device	14
Testing Mobile Web Applications on an iOS Simulator	14
Testing Hybrid Applications on iOS	15
Preparing an iOS Device for Testing	16
Preparing an iOS App for Testing	16
Installing the Silk Test Information Service on a Mac	17
Preparing a Mac to Test Mobile Applications on iOS	17
Using a Personal Team Profile for Testing on Physical iOS Devices	19
Editing the Properties of the Silk Test Information Service	20
Uninstalling the Silk Test Information Service from a Mac	21
Recommended Settings for iOS Devices	21
Running Existing Scripts on iOS Using XCUITest	22
Testing an Installed App	22
Recording Mobile Applications	22
Selecting the Mobile Device for Test Replay	23
Using Devices from Mobile Center	23
Using SauceLabs Devices	24
Editing Remote Locations	24
Connection String for a Mobile Device	25
Interacting with a Mobile Device	28
Releasing a Mobile Device	28
Releasing a Mobile Device After Recording	29
Releasing a Mobile Device After Replay	29
Troubleshooting when Testing Mobile Applications	29
How Can I Use Chrome for Android to Replay Tests?	34
Limitations for Testing Mobile Web Applications	35
Limitations for Testing Native Mobile Applications	36
Clicking on Objects in a Mobile Website	38
Using Existing Mobile Web Tests	38

# Testing Mobile Applications

Silk Test enables you to automatically test your native mobile applications (apps) and mobile web applications. Automatically testing your mobile applications with Silk Test provides the following benefits:

- It can significantly reduce the testing time of your mobile applications.
- You can create your tests once and then test your mobile applications on a large number of different devices and platforms.
- You can ensure the reliability and performance that is required for enterprise mobile applications.
- It can increase the efficiency of QA team members and mobile application developers.
- Manual testing might not be efficient enough for an agile-focused development environment, given the large number of mobile devices and platforms on which a mobile application needs to function.

 **Note:** To test native mobile applications or hybrid applications with Silk Test, you require a native mobile license. For additional information, see [Licensing Information](#).

 **Note:** Silk Test provides support for testing mobile apps on both Android and iOS devices.

For information on the supported operating system versions and the supported browsers for testing mobile applications, refer to the [Release Notes](#).

## Android

Silk Test enables you to test a mobile application on an Android device or an Android emulator.

### Prerequisites for Testing Mobile Applications on Android

Before you can test a mobile application (app) on an Android device or on an Android emulator, ensure that the following prerequisites are met:

- Ensure that Java is installed on the machine on which Silk Test is running, and that the path to Java is added to the *Path* environment variable. Silk Test requires either a JRE or a JDK for mobile testing. If Java is not installed, add `C:\Program Files (x86)\Silk\SilkTest\ng\jre\bin` to the *Path*.
- If you have created your own hybrid app by adding a web view to a native mobile app, add the following code to the app to make your app testable with Silk Test:


```
WebView.setWebContentsDebuggingEnabled(true);  
webView.getSettings().setJavaScriptEnabled(true);
```

- Silk Test does not support showing a live view of the device screen for Android 4.4. Micro Focus recommends using Android 5 or later.

### Testing Mobile Applications on Android

To test a mobile application on a physical Android device or on an Android emulator, perform the following tasks:

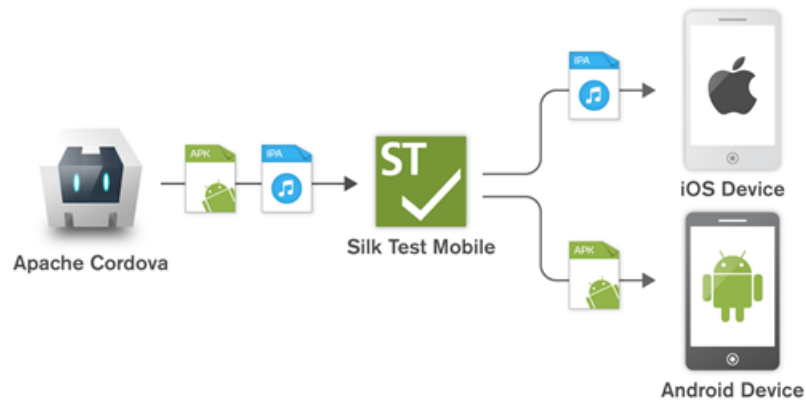
1. Ensure that you have met the prerequisites for testing mobile applications on Android.  
For additional information, see [Prerequisites for Testing Mobile Applications on Android](#).
2. If you want to test the mobile application on an Android emulator, configure the emulator settings for Silk Test.  
For additional information, see [Configuring the Android Emulator for Silk Test](#).

3. Start the Android emulator or connect the device to the machine on which Silk Test is installed.
4. If you want to test the mobile application on a physical Android device that you are using for the first time on this machine, install the appropriate Android USB Driver on the machine.  
For additional information, see [Installing a USB Driver](#).
5. If you want to test the mobile application on a physical Android device, enable USB-debugging on the Android device.  
For additional information, see [Enabling USB-Debugging](#).
6. Create a Silk Test project for your mobile application.
7. Create a test for your mobile application.
8. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
9. To test a mobile web application:
  - a) Select the **Web** tab.
  - b) Select the mobile browser that you want to use.
  - c) Specify the web page to open in the **Enter URL to navigate** text box.
10. To test a native mobile application or a Hybrid application:  
 **Note:** To test native mobile applications or hybrid applications with Silk Test, you require a native mobile license. For additional information, see [Licensing Information](#).
  - a) Select the **Mobile** tab.
  - b) Select the mobile device, on which you want to test the app, from the list.
  - c) Click **Browse** to select the app file or enter the full path to the app file into the **Mobile app file** text field.  
Silk Test supports HTTP and UNC formats for the path.  
Silk Test installs the app on the mobile device or emulator.
11. Click **OK**.  
An Android device or emulator must not be screen-locked during testing. To keep the device awake while it is connected to a machine, open the settings and tap **Developer Options**. Then check **Stay awake** or **Stay awake while charging**.
12. Use the **Recording** window to record the test against the mobile application.  
For additional information, see [Recording Mobile Applications](#).
13. When you have recorded all the actions, stop the recording.
14. Replay the test.
15. Analyze the test results.

## Testing Hybrid Applications on Android

*Hybrid applications* (apps) are apps that are run on the device, like native applications, but are written with web technologies, for example HTML5, CSS, and JavaScript.

Silk Test provides full browser support for testing debug hybrid apps that consist of a single web view, which is embedded in a native container. A common example of such a hybrid app would be an Apache Cordova application.



To prepare a non-debug hybrid app for testing, enable remote debugging in the app by adding the following code to the app:

```
WebView.setWebContentsDebuggingEnabled(true);
webView.getSettings().setJavaScriptEnabled(true);
```


To test non-debug hybrid apps without remote debugging enabled or hybrid apps that include more than one web view, enable the Silk Test fallback support by setting the option `OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT` to `TRUE`. For additional information, see *Setting Advanced Options*. With the fallback support enabled, Silk Test recognizes and handles the controls in a web view as native mobile controls instead of browser controls. For example, the following code clicks on a link when using browser support:

```
desktop.setOption(CommonOptions.OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT,
false);
desktop.<DomLink> find("//INPUT[@id='email']").click();
```

With the fallback support enabled, the following code clicks on the same link:

```
desktop.setOption(CommonOptions.OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT,
true);
desktop.<DomLink> find("//MobileTextField[@resource-id='email']").click();
```

Silk Test can detect web views that support Chrome remote debugging. Silk Test can detect web views with either the package `com.android.webview` or the package `com.google.android.webview`, which are the default packages on most Android devices.

 **Note:** Silk Test supports testing hybrid apps on Android 4.4 or later. To test hybrid apps on Android, Android System WebView version 51 or later is required.

The process for testing a hybrid app on Android is the same as the process for testing a mobile native application. For additional information, see [Testing Mobile Applications on Android](#).

## Installing a USB Driver

To connect an Android device for the first time to your local machine to test your mobile applications, you need to install the appropriate USB driver.

The device manufacturer might provide an executable with all the necessary drivers for the device. In this case you can just install the executable on your local machine. If the manufacturer does not provide such an executable, you can install a single USB driver for the device on the machine.

To install the Android USB driver:

1. Download the appropriate driver for your device.

For example, for information on finding and installing a USB driver for a Google Nexus device, see <http://developer.android.com/tools/extras/oem-usb.html>.

2. Connect your Android device to a USB port on your local machine.
3. From your desktop or **Windows Explorer**, right-click **Computer** and select **Manage**.
4. In the left pane, select **Device Manager**.
5. In the right pane, locate and expand **Other device**.
6. Right-click the device name, for example *Nexus 5x*, and select **Update Driver Software**. The **Hardware Update Wizard** opens.
7. Select **Browse my computer for driver software** and click **Next**.
8. Click **Browse** and navigate to the folder to which you have downloaded the USB driver.
9. Select the USB driver.
10. Click **Next** to install the driver.

## Enabling USB-Debugging

To communicate with an Android device over the Android Debug Bridge (adb), enable USB debugging on the device.

1. On the Android device, open the settings.
2. Tap **Developer Settings**.

The developer settings are hidden by default. If the developer settings are not included in the settings menu of the device:

  - a) Depending on whether the device is a phone or a pad, scroll down and tap **About phone** or **About Pad**.
  - b) Scroll down again and tap **Build Number** seven times.
3. In the **Developer settings** window, check **USB-Debugging**.
4. Set the USB mode of the device to **Media device (MTP)**, which is the default setting.

For additional information, refer to the documentation of the device.

## Recommended Settings for Android Devices

To optimize testing with Silk Test, configure the following settings on the Android device that you want to test:

- Enable USB-debugging on the Android device. For additional information, see [Enabling USB-Debugging](#)
- An Android device must be connected as a media device to the machine on which the Open Agent is running. The USB mode of the Android device must be set to **Media device (MTP)**.
- An Android device or emulator must not be screen-locked during testing. To keep the device awake while it is connected to a machine, open the settings and tap **Developer Options**. Then check **Stay awake** or **Stay awake while charging**.

## Configuring the Android Emulator for Silk Test



**Note:** When using an Android emulator, an additional adb server is running in addition to the one that is used by Silk Test. If the running adb servers have different versions, the connection between the Open Agent and the device might become unstable or even break. To avoid version mismatch errors, specify the path to the Android SDK directory by setting the environment variable `SILK_ANDROID_HOME`, for example to `C:\Users\\AppData\Local\Android\android-sdk`. If the information service was running during this change, use the Windows Service Manager to restart the Silk Test information service with the updated environment variable. If the environment variable is not set, Silk Test uses the adb version that is shipped with Silk Test.

When you want to test mobile applications on an Android emulator with Silk Test, you have to configure the emulator for testing:

1. Install the latest version of the Android SDK.

For information on how to install and configure the Android SDK, see [Get the Android SDK](#).

2. Install Android Studio 2.



**Tip:** You can skip installing Android Studio 2 and use the emulator provided with the Android SDK. However, Micro Focus recommends installing Android Studio 2 for improved emulator performance. The remaining steps in this topic require Android Studio 2 to be installed.

3. From Android Studio 2, start the **AVD Manager**.

4. Click **Create Virtual Device**.

5. Select a virtual device.

6. Click **Next**.

7. Download and select a system image of Android that includes Google APIs.

8. Click **Next**.

9. Configure the virtual device according to your requirements.

10. Click **Show Advanced Settings**.

11. Adjust the RAM size and the heap space used by the emulator to an amount that is manageable by your machine.



**Tip:** Micro Focus recommends using at least 1 GB RAM and 256 MB heap space.

12. Select **Auto** from the list in the **Emulated Performance** area.

The screenshot shows the 'Show Advanced Settings' dialog in the AVD Manager. The settings are as follows:

- AVD Name: Nexus 5 API 23
- AVD Id: Nexus\_5\_API\_23
- Device: Nexus 5, 4,95" 1080x1920 xxhdpi
- System Image: Marshmallow Android 6.0 x86\_64
- Startup size and orientation: Scale: Auto, Orientation: Portrait
- Camera: Front: None, Back: None
- Network: Speed: Full, Latency: None
- Emulated Performance: Graphics: Auto, Multi-Core CPU: 1 (Experimental)
- Memory and Storage: RAM: 1 GB, VM heap: 256 MB, Internal Storage: 800 MB, SD card: Studio-managed 100 MB
- Device Frame: Enable Device Frame (checked), Custom skin definition: nexus\_5
- Keyboard: Enable keyboard input (checked)

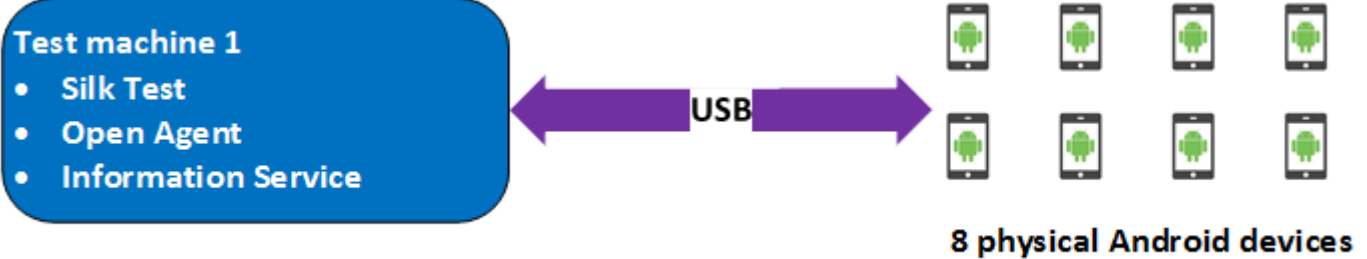


13. Click **Finish**.

## Tested Configurations for Parallel Test Execution

With Silk Test, you can run automated tests on multiple Android devices in parallel. The amount of Android devices that you are able to use in parallel depends on the available hardware. Micro Focus has successfully tested the following hardware configurations:

### Configuration with a single test machine

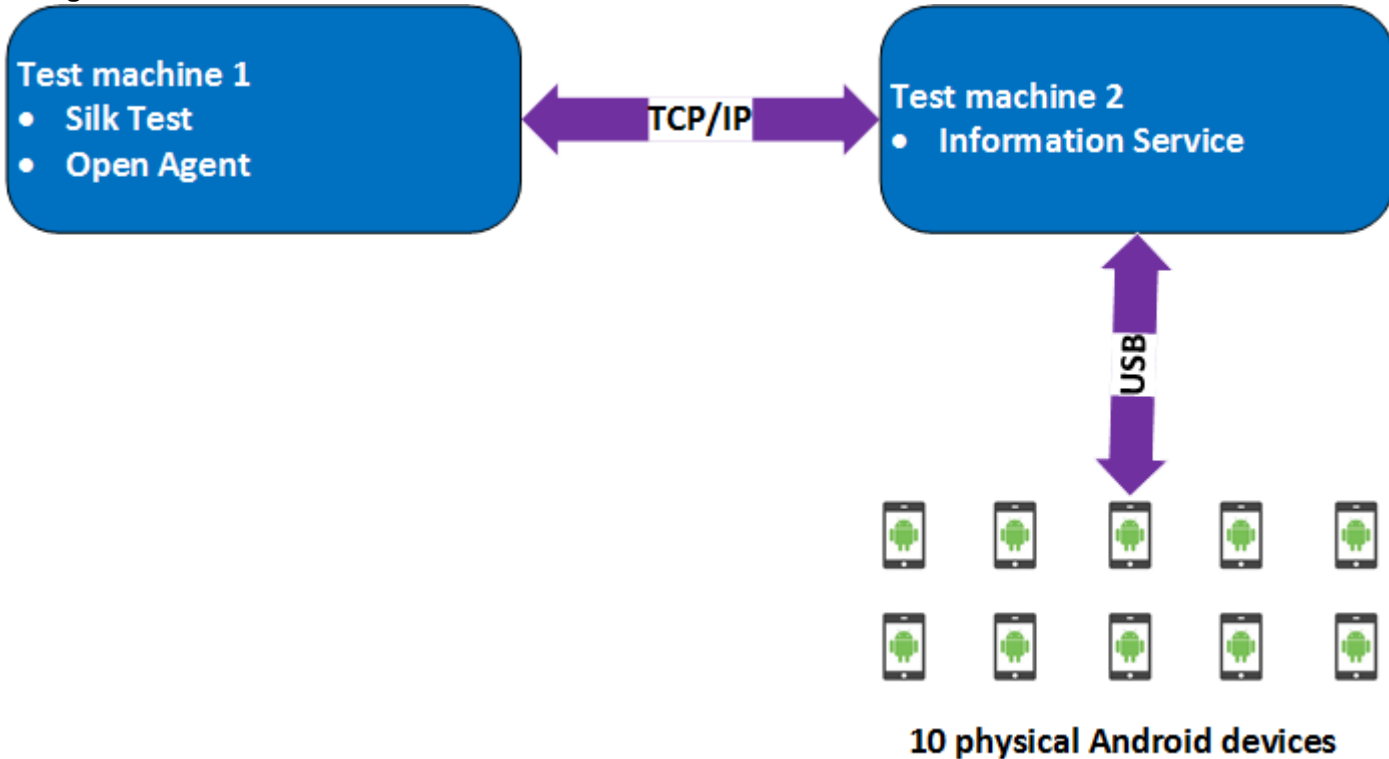


Using a single test machine directly connected to the Android devices through USB, we tested up to 8 physical Android devices in parallel.

The test machine was a Lenovo ThinkPad T450 with the following hardware specifications:

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 cores (4 threads)
- 8 GB RAM

### Configuration with two test machines



Here we are using two test machines, one with Silk Test installed and another, which is configured as a remote location for the first machine and has the Silk Test Information Service installed. Using such a configuration, we tested up to 10 physical Android devices in parallel.

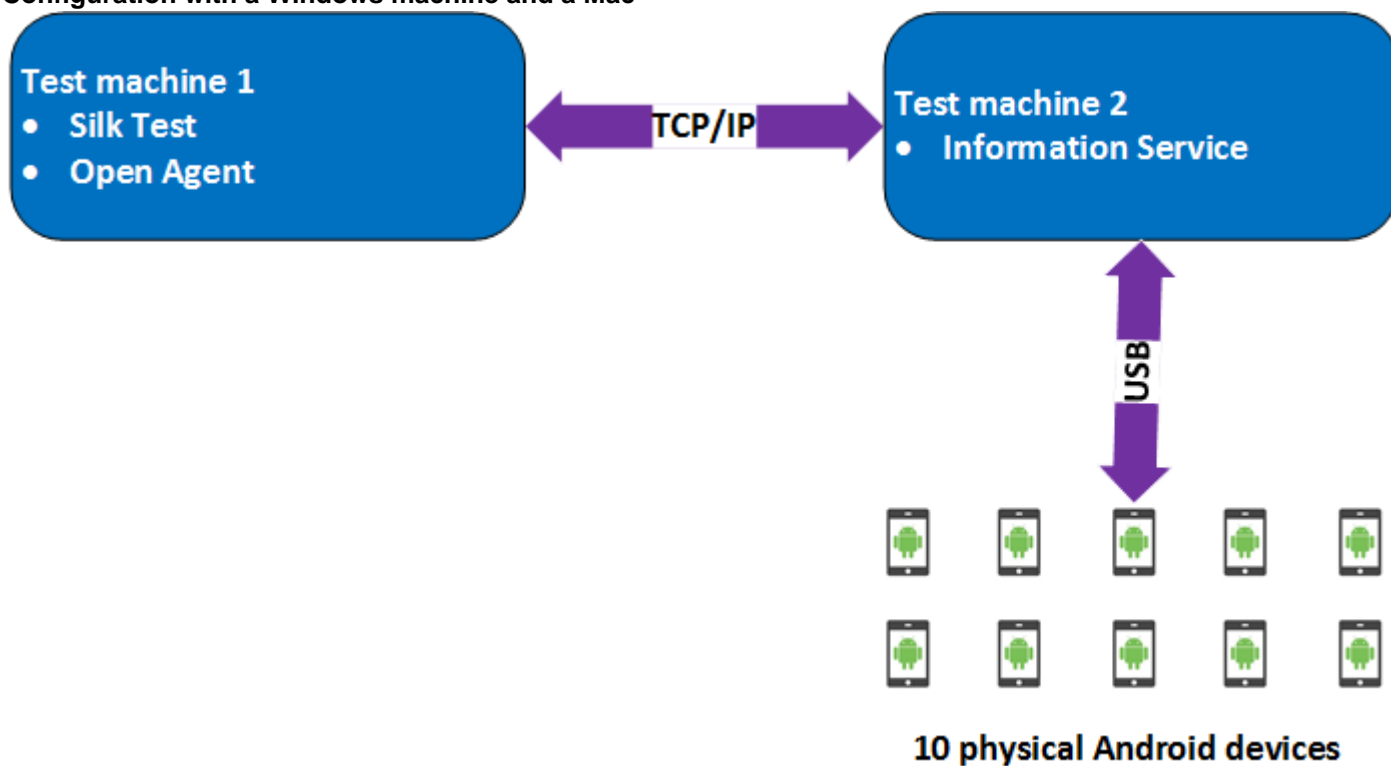
Test machine 1 was a Lenovo ThinkPad T450 with the following hardware specifications:

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 cores (4 threads)
- 8 GB RAM

Test machine 2 was a Dell Precision T1700 with the following hardware specifications:

- Intel® Core™ i7 - 4770 CPU @ 3.40 GHz
- 4 cores (8 threads)
- 16 GB RAM

### Configuration with a Windows machine and a Mac



Here we are using two test machines, a Windows machine with Silk Test installed and a Mac, which is configured as a remote location for the first machine and has the Silk Test Information Service installed. Using such a configuration, we tested up to 10 physical Android devices in parallel.

Test machine 1 was a Lenovo ThinkPad T450 with the following hardware specifications:

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 cores (4 threads)
- 8 GB RAM


Test machine 2 was an Apple Mac Mini with the following hardware specifications:


- Intel® Core™ i5 - 4782U CPU @ 2.60 GHz
- 2 cores (4 threads)
- 16 GB RAM


# iOS

Silk Test enables you to test a mobile application on an iOS device or an iOS Simulator.

Because of significant changes by Apple in iOS 9.3 in comparison to the previous versions of iOS, Silk Test supports testing mobile applications on iOS 9.3 or later. For a list of the supported iOS versions, refer to the [Release Notes](#).

 **Note:** Testing mobile applications on iOS 11 requires Xcode 9. When using Xcode 9 on a Mac, testing on physical devices and Simulators with iOS versions prior to iOS 11 that are connected to or running on this Mac is not supported. Use Xcode 8.3 to test physical devices and Simulators with iOS 9.3 and iOS 10.

 **Tip:** To test on iOS versions prior to iOS 9.3, you can use Silk Test 17.5. The following table shows the major changes when testing on iOS with Silk Test 19.0 in comparison to Silk Test 17.5:

Silk Test 19.0	Silk Test 17.5
Supports iOS 9.3, iOS 10, and iOS 11.	Supports iOS 8.1, 8.2, 8.3, 8.4, 9.0, 9.1, 9.2, and 9.3
Supports Xcode 8.3 and Xcode 9.	Supports Xcode 6 and Xcode 7.
Supports testing multiple physical iOS devices in the same user session.	Requires multiple user sessions on a Mac to test multiple physical iOS devices.
 <b>Tip:</b> If you have created multiple user sessions to test with Silk Test 17.5, Micro Focus recommends removing all user sessions except one.	

## Prerequisites for Testing Mobile Applications on iOS

Before you can test a mobile application (app) on an iOS device or on an iOS Simulator, ensure that the following prerequisites are met:

- The current version of the Silk Test information service is installed on the Mac. For additional information, see [Installing the Silk Test Infoservice on a Mac](#).
- If you want to test your application on a physical iOS device, ensure the following:
  - The device is connected to the Mac.
  - The device has a supported version of iOS. For a list of the supported iOS versions, refer to the [Release Notes](#).
- If you want to test your application on an iOS Simulator, ensure the following:
  - The iOS Simulator image is installed on the Mac.
  - The iOS Simulator image has a supported version of iOS. For a list of the supported iOS versions, refer to the [Release Notes](#).
- If you want to test your application on an physical iOS device, ensure that the same time zone is set on the device and the Mac.
- A supported version of Xcode is installed on the Mac.
- Silk Test is installed on a Windows machine.
- The Mac is located in the same network as the Windows machine and is added as a remote location to the Windows machine.
- To test a native mobile app on an iOS device, ensure that the `.ipa` file of your app has been signed with a developer account. For additional information, see [Preparing an iOS App for Testing](#).
- To test a native mobile app on an iOS Simulator, ensure that the app has been zipped. For additional information, see [Testing Native Mobile Applications on an iOS Simulator](#).

- To test a native mobile app on both an iOS device and an iOS Simulator, ensure that both the signed `.ipa` file and the zipped `.app` directory are located in the same folder.
- To test a native mobile app, ensure that the ID of the iOS device is associated with the developer profile which was used to sign the app.
- The iOS device must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings > General > Passcode Lock**.
- The Mac should not switch off the screen during testing, otherwise the **Playback Status** dialog box will not display anything.
- To test a mobile application on an iOS Simulator, deactivate the display sleep on the Mac during testing.
- To test a native mobile app on a physical iOS device, enable the UI automation on the device. For additional information, see [Preparing an iOS Device for Testing](#).
- To test a mobile web application with Apple Safari on a physical iOS device, activate the **Web Inspector**. For additional information, see [Preparing an iOS Device for Testing](#).
- Micro Focus recommends using iOS devices which have a Lightning connector. Silk Test does not support showing a live view of the device screen for iOS devices that are not connected to a Mac through a Lightning cable.

## Testing Native Mobile Applications on a Physical iOS Device



**Note:** To test native mobile applications or hybrid applications with Silk Test, you require a native mobile license. For additional information, see [Licensing Information](#).

For information on the prerequisites for testing mobile applications on iOS, see [Prerequisites for Testing Mobile Applications on iOS](#). For information on the known limitations when testing native mobile applications, see [Limitations for Testing Mobile Native Applications](#).

To test a native mobile application (app) or a hybrid application on a physical iOS device, perform the following tasks:

1. Prepare the iOS device for testing.  
For additional information, see [Preparing an iOS Device for Testing](#).
2. Prepare the app for testing.  
For additional information, see [Preparing an iOS App for Testing](#).
3. Prepare the Mac for testing. For additional information, see [Preparing a Mac for Testing Mobile Applications on iOS](#).
4. Add the Mac, to which the iOS device is connected, as a remote location to the Windows machine on which Silk Test is installed.  
For additional information, see [Editing Remote Locations](#).



**Note:** At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac. With Silk Test 17.5 Hotfix 1 or later, you are no longer required to use multiple user sessions on a Mac to test mobile applications on iOS.

5. Create a Silk Test project for your mobile application.
6. Create a test for your mobile application.
7. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
8. Select the **Mobile** tab.
9. Select the mobile device, on which you want to test the app, from the list.
10. Click **Browse** to select the app file or enter the full path to the app file into the **Mobile app file** text field.  
Silk Test supports HTTP and UNC formats for the path.  
Silk Test installs the app on the mobile device.


11. Click **OK**.

An iOS device or Simulator must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings > General > Passcode Lock**.


12. When you have recorded all actions, stop recording.

13. Replay the test.

14. Analyze the test results.

 **Note:** To test a native mobile app on both an iOS device and an iOS Simulator, ensure that both the signed `.ipa` file and the zipped `.app` directory are located in the same folder.

## Testing Native Mobile Applications on an iOS Simulator

 **Note:** To test native mobile applications or hybrid applications with Silk Test, you require a native mobile license. For additional information, see *Licensing Information*.

For information on the prerequisites for testing mobile applications on iOS, see [Prerequisites for Testing Mobile Applications on iOS](#). For information on the known limitations when testing native mobile applications, see [Limitations for Testing Mobile Native Applications](#).

To test a native mobile application (app) or a hybrid application on an iOS Simulator, perform the following tasks:

1. Prepare the Mac for testing. For additional information, see [Preparing a Mac for Testing Mobile Applications on iOS](#).

2. In the Xcode project of the app, compile the app for the iOS Simulator.

You can compile the app either from the Xcode UI or from the command line. For example, to compile the app through the command line for an iOS Simulator with iOS 10.0, execute the following command:


```
xcodebuild -sdk iphonesimulator10.0
```

3. Zip up the `.app` directory of the app into a `.zip` file.

By default, the `.app` directory is located in the directory `~/Library/Developer/Xcode/DerivedData`. You can click **File > Project Settings** in Xcode to see into which location the directory is stored.

4. Add the Mac, on which the iOS Simulator is installed, as a remote location to the Windows machine on which Silk Test is installed.

For additional information, see [Editing Remote Locations](#).

 **Note:** You can only test on one iOS Simulator that is installed on a Mac. Multiple Silk Test users cannot simultaneously test on multiple iOS Simulators that are installed on the same Mac.

5. Create a Silk Test project for your mobile application.

6. Create a test for your mobile application.

7. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.

8. Select the **Mobile** tab.

9. Select the iOS Simulator from the list.

10. Click **Browse** to select the zipped app file or enter the full path to the zipped app file into the **Mobile app file** text field.

Silk Test supports HTTP and UNC formats for the path.

Silk Test installs the app on the iOS Simulator.

11. Click **OK**.

An iOS device or Simulator must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings > General > Passcode Lock**.

12. When you have recorded all actions, stop recording.
13. Replay the test.
14. Analyze the test results.



**Note:** To test a native mobile app on both an iOS device and an iOS Simulator, ensure that both the signed `.ipa` file and the zipped `.app` directory are located in the same folder.

## Testing Mobile Web Applications on a Physical iOS Device

For information on the prerequisites for testing mobile applications on iOS, see [Prerequisites for Testing Mobile Applications on iOS](#). For information on the known limitations when testing mobile web applications, see [Limitations for Testing Mobile Web Applications](#).

To test a mobile web application on a physical iOS device, perform the following tasks:

1. Prepare the iOS device for testing.  
For additional information, see [Preparing an iOS Device for Testing](#).
2. Prepare the Mac for testing. For additional information, see [Preparing a Mac for Testing Mobile Applications on iOS](#).
3. Add the Mac, to which the iOS device is connected, as a remote location to the Windows machine on which Silk Test is installed.  
For additional information, see [Editing Remote Locations](#).



**Note:** At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac. With Silk Test 17.5 Hotfix 1 or later, you are no longer required to use multiple user sessions on a Mac to test mobile applications on iOS.

4. Create a Silk Test project for your mobile application.
5. Create a test for your mobile application.
6. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
7. To test a mobile web application:
  - a) Select the **Web** tab.
  - b) Select the mobile browser that you want to use.
  - c) Specify the web page to open in the **Enter URL to navigate** text box.
8. Click **OK**.  
An iOS device or Simulator must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings > General > Passcode Lock**.
9. When you have recorded all actions, stop recording.
10. Replay the test.
11. Analyze the test results.

## Testing Mobile Web Applications on an iOS Simulator


For information on the known limitations when testing mobile web applications, see [Limitations for Testing Mobile Web Applications](#).

To test a mobile web application on an iOS Simulator, perform the following tasks:

1. Prepare the Mac for testing. For additional information, see [Preparing a Mac for Testing Mobile Applications on iOS](#).

2. Add the Mac, on which the iOS Simulator is installed, as a remote location to the Windows machine on which Silk Test is installed.

For additional information, see [Editing Remote Locations](#).

 **Note:** At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac. With Silk Test 17.5 Hotfix 1 or later, you are no longer required to use multiple user sessions on a Mac to test mobile applications on iOS.

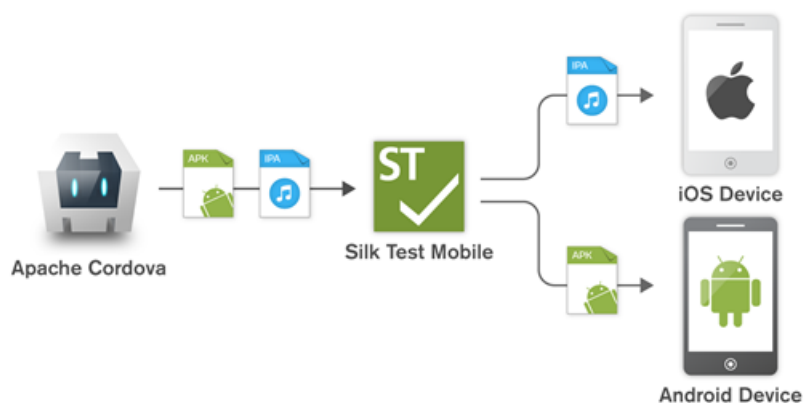
3. Create a Silk Test project for your mobile application.
4. Create a test for your mobile application.
5. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
6. To test a mobile web application:
  - a) Select the **Web** tab.
  - b) Select the mobile browser that you want to use.
  - c) Specify the web page to open in the **Enter URL to navigate** text box.
7. Click **OK**.

An iOS device or Simulator must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings > General > Passcode Lock**.
8. When you have recorded all actions, stop recording.
9. Replay the test.
10. Analyze the test results.

## Testing Hybrid Applications on iOS

*Hybrid applications* (apps) are apps that are run on the device, like native applications, but are written with web technologies, for example HTML5, CSS, and JavaScript.

Silk Test provides full browser support for testing debug hybrid apps that consist of a single web view, which is embedded in a native container. A common example of such a hybrid app would be an Apache Cordova application.



To test non-debug hybrid apps without remote debugging enabled or hybrid apps that include more than one web view, enable the Silk Test fallback support by setting the option `OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT` to `TRUE`. For additional information, see [Setting Advanced Options](#). With the fallback support enabled, Silk Test recognizes and handles the controls in a web view as native mobile controls instead of browser controls. For example, the following code clicks on a link when using browser support:

```
desktop.setOption(CommonOptions.OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT, false);  
desktop.<DomLink> find("//INPUT[@id='email']").click();
```

With the fallback support enabled, the following code clicks on the same link:

```
desktop.setOption(CommonOptions.OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT, true);
desktop.<DomLink> find("//MobileTextField[@resource-id='email']").click();
```

The process for testing a hybrid app on iOS is the same as the process for testing a mobile native application. For additional information, see [Testing Native Mobile Applications on a Physical iOS Device](#) or [Testing Native Mobile Applications on an iOS Simulator](#).

Before testing a hybrid app on an iOS device, ensure that the **Web Inspector** is activated on the device.

## Preparing an iOS Device for Testing



**Note:** To test native mobile applications or hybrid applications with Silk Test, you require a native mobile license. For additional information, see [Licensing Information](#).

To prepare the iOS device to test mobile applications:

1. Start Xcode on the Mac.
2. Connect the iOS device to the Mac.
3. On the iOS device, click **Settings > Developer**.



**Tip:** If the **Developer** menu is not displayed on the iOS device, restart the device and the Mac.

4. Activate **Enable UI Automation**.
5. To test a mobile web application on Apple Safari, click **Settings > Safari > Advanced**.
6. Activate the **Web Inspector**.
7. If you want to test on an iOS Simulator, enable **Rotate Device Automatically**.

You can enable this setting by using the **Silk Test Configuration Assistant** or by enabling it manually. To open the **Configuration Assistant** on a Mac, click on the Silk Test icon in the status menus and select **Configuration Assistant**. To enable the setting manually, perform the following actions:

- a) On the Mac, start the iOS Simulator.
- b) With Xcode 9 or later, expand the **Hardware** menu.  
With prior versions of Xcode, expand the **Debug** menu.
- c) Check **Rotate Device Automatically**.

8. If you want to test on an iOS Simulator with Xcode 9 or later, disable **Show Device Bezels**.

You can disable this setting by using the **Silk Test Configuration Assistant** or by disabling it manually. To open the **Configuration Assistant** on a Mac, click on the Silk Test icon in the status menus and select **Configuration Assistant**. To disable the setting manually, perform the following actions:

- a) On the Mac, start the iOS Simulator.
- b) Expand the **Window** menu.
- c) Uncheck **Show Device Bezels**.

## Preparing an iOS App for Testing

To be able to test a specific iOS app on a specific iOS device with Silk Test, consider the following:

- Test automation is only possible with iOS apps that can be installed manually on specific iOS devices. To be able to sign an iOS app, you require a membership in the *Apple Developer Program*. For additional information, see [Choosing a Membership](#). To test without having a membership in the *Apple Developer Program*, see [Using a Personal Team Profile for Testing on Physical iOS Devices](#).



**Note:** You cannot automatically test iOS apps that are created for publication in the App Store, as well as apps that can be installed manually on any iOS device.

- Before you can install and execute an iOS app on a specific iOS device, you have to register the iOS device with your Apple Developer account.



- You have to use Xcode to create an IPA file of the iOS app, which you can then install on the iOS device. To create IPA files for testing on a specific iOS device, members of the Apple Developer Program can use the *Archive* mechanism of Xcode, by using one of the following two options:
  - If you are a member of the *Apple Developer Enterprise Program*, you can use the **Save for Ad Hoc Deployment** option.
  - If you are a member of the Apple Developer Program, but not of the Apple Developer Enterprise Program, you can use the **Save for Development Deployment** option.

For additional information, see [Exporting Your App for Testing \(iOS, tvOS, watchOS\)](#).

To be able to test a specific iOS app on an iOS Simulator with Silk Test, use Xcode to create a ZIP file of the iOS app, which you can then install on the iOS Simulator. For additional information, refer to the Xcode documentation.

## Installing the Silk Test Information Service on a Mac



**Note:** To install the information service on a Mac, you require administrative privileges on the Mac.

To create and execute tests against Apple Safari on a Mac, or against mobile applications on an iOS or Android device that is connected to a Mac, install the Silk Test information service (information service) on the Mac, and then use the **Remote Locations** dialog box to connect a Windows machine, on which Silk Test is installed, to the Mac.

To install the information service on a Mac:

1. Ensure that a Java JDK is installed on the Mac.
2. If you want to test mobile applications on an iOS device, ensure that Xcode is installed on the Mac.
3. Access the information service setup file, `SilkTestInformationService<Version>-<Build Number>.pkg`.
  - If you have downloaded the information service setup file while installing Silk Test, open the folder `macOS` in the Silk Test installation directory, for example `C:\Program Files (x86)\SilkTest\SilkTest`.
  - If you have not downloaded the information service setup file while installing Silk Test, you can download the setup file from [Micro Focus SupportLine](#).
4. Copy the file `SilkTestInformationService<Version>-<Build Number>.pkg` to the Mac.
5. Execute `SilkTestInformationService<Version>-<Build Number>.pkg` to install the information service.
6. Follow the instructions in the installation wizard.
7. When asked for the password, provide the password of the currently signed in Mac user.
8. When Apple Safari opens and a message box asks whether to trust the SafariDriver, click **Trust**.



**Note:** You can only install the SafariDriver if you are directly logged in to the Mac, and not connected through a remote connection.

To complete the installation, the installer logs the current Mac user out. To verify that the information service was installed correctly, log in to the Mac and click on the Silk Test icon in the top-right corner of the screen to see the available devices and browsers.



**Tip:** If the Silk Test icon does not appear, restart the Mac.

## Preparing a Mac to Test Mobile Applications on iOS



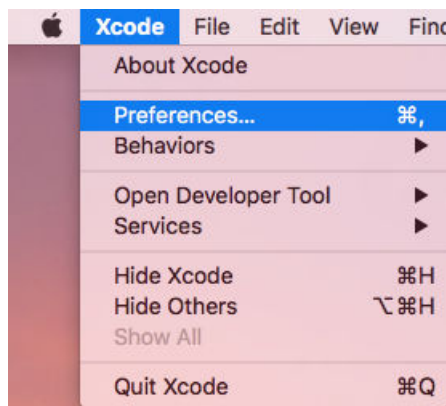
**Note:** To test native mobile applications or hybrid applications with Silk Test, you require a native mobile license. For additional information, see [Licensing Information](#).

To test mobile applications on iOS, you require a Mac to which you can connect the iOS device, or on which the iOS Simulator is running. This Mac requires Xcode to be installed. For additional information on the prerequisites for testing mobile applications on iOS, see [Prerequisites for Testing Mobile Applications on iOS](#).

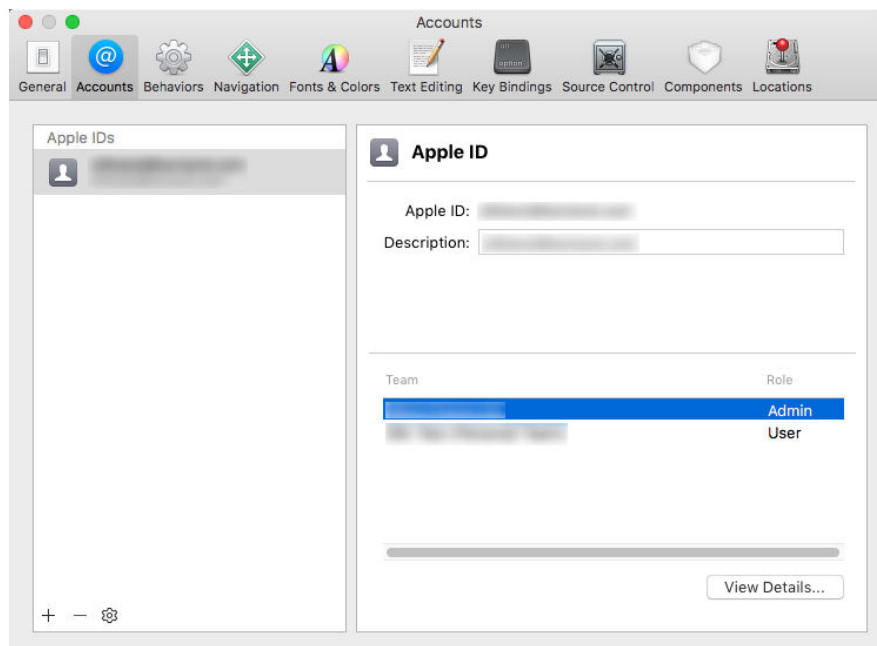
To execute iOS tests on a physical iOS device, follow the instructions in the **Silk Test Configuration Assistant** to configure the `WebDriverAgentRunner` Xcode project. To open the **Configuration Assistant**, click on the Silk Test icon in the status menus and select **Configuration Assistant**.

If for any reason you want to manually build the `WebDriverAgentRunner` Xcode project, perform the following actions:

1. Start Xcode on the Mac.
2. Select **Xcode > Preferences**.



3. In the **Preferences** window, select your account.
  - a) Select the **Accounts** tab.
  - b) Choose your **Apple ID**.
  - c) Choose your **Team**.
  - d) Click **View Details**.



4. Access the **Apple Member Center** and retrieve your development team.
5. In a terminal, navigate to `~/ .silk/silktest/conf/`.
6. Rename the xconfig file template `silktest.xconfig.sample` to `silktest.xconfig`.
7. Add your development team to the `silktest.xconfig` file.

```
DEVELOPMENT_TEAM = <your development team>
```

8. Execute the following commands in a terminal on the Mac to verify that you have prepared the `WebDriverAgentRunner` project correctly:

- a) Determine the unique device id (udid) of your physical iOS device:

```
idevice_id -l
```

- b) Navigate to the `WebDriverAgentRunner` project:

```
cd /Application/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver/WebDriverAgent
```

- c) Test that the `WebDriverAgent` can be built:

```
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -xconfig ~/ .silk/silktest/conf/silktest.xconfig -destination 'id=<udid>' test
```

Replace the `<udid>` with the unique device id that you have determined previously.



**Tip:** If the `xcodebuild` command fails, follow the instructions in the error message.

Additionally, open the Preferences window of the `WebDriverAgentRunner` project and ensure that the **Automatically manage signing** check box in the **General** tab is not checked.

9. *Optional:* In the `infoservice.properties` file, you can specify the port for the Silk Test Information Service or capabilities which are used during all test runs on the Mac.

For additional information, see [Editing the Properties of the Silk Test Information Service](#).

## Using a Personal Team Profile for Testing on Physical iOS Devices

If you have no membership in the Apple Developer Program, you can use a personal team profile to test an application on a physical iOS device:

1. On the Mac, navigate to `/Application/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver/WebDriverAgent`.
2. Open `WebDriverAgent.xcodeproj` project in Xcode.
3. From the **TARGETS** list, select the `WebDriverAgentLib` target:
  - a) Click the **General** tab.
  - b) Select **Automatically manage signing**.
  - c) Select your development team.

The **Signing Certificate** is automatically selected.

4. From the **TARGETS** list, select the `WebDriverAgentRunner` target:
  - a) Click the **General** tab.
  - b) Select **Automatically manage signing**.
  - c) Select your development team.

The **Signing Certificate** is automatically selected.

5. If Xcode fails to create a provisioning profile for the `WebDriverAgentRunner` target, manually change the bundle id for the target.
  - a) Click the **Build Settings** tab.
  - b) Change the **Product Bundle Identifier** to something that Xcode accepts.

For example, if the **Product Bundle Identifier** is `com.facebook.WebDriverAgentRunner`, change it to `io.appium.WebDriverAgentRunner` or `io.borland.WebDriverAgentRunner`.

c) Click the **General** tab.

The target should now have a provisioning profile.

6. Save the `WebDriverAgent.xcodeproj` project.

7. To verify that everything works as expected, build the project:

```
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -destination 'id=<udid>' test IPHONEOS_DEPLOYMENT_TARGET=10.3
```

8. To avoid problems during the reinstallation of the `WebDriverAgent` apps, permanently install an additional app that uses the same provisioning profile, on the device. For example, install the `IntegrationApp` of the `WebDriverAgent Xcode` project:

a) From the **TARGETS** list, select the `IntegrationApp` target.

b) Click the **General** tab.

c) Select **Automatically manage signing**.

d) Select your development team.

9. If Xcode fails to create a provisioning profile for the `IntegrationApp` target, manually change the bundle id for the target in the same way as described above for the `WebDriverAgentRunner` target.

10. After successfully configuring the `IntegrationApp` target, install and run the `IntegrationApp` on the physical iOS device:

a) Select the target and the iOS device.

b) Click **Play**.

Although the apps are successfully installed on the device, an error message like the following might appear in the console or the Appium log files:

```
2017-01-24 09:02:18.358 xcodebuild[30385:339674] Error Domain=com.apple.platform.iphoneros
Code=-12 "Unable to launch com.apple.test.WebDriverAgentRunner-Runner"
UserInfo={NSLocalizedString=Unable to launch com.apple.test.WebDriverAgentRunner-Runner,
NSUnderlyingError=0x7fa839cad60 {Error Domain=DTXMessage Code=1 "(null)"
UserInfo={DTXExceptionKey=The operation couldn't be completed. Unable to launch
com.apple.test.WebDriverAgentRunner-Runner because it has an invalid code signature, inadequate
entitlements or its profile has not been explicitly trusted by the user. : Failed to launch process with bundle
identifier 'com.apple.test.WebDriverAgentRunner-Runner'}}} 2017-01-24 09:02:18.358
xcodebuild[30385:339674] Error Domain=IDETestOperationsObserverErrorDomain Code=5 "Early
unexpected exit, operation never finished bootstrapping - no restart will be attempted"
UserInfo={NSLocalizedString=Early unexpected exit, operation never finished bootstrapping - no
restart will be attempted} Testing failed: Test target WebDriverAgentRunner encountered an error (Early
unexpected exit, operation never finished bootstrapping - no restart will be attempted)
The problem is that the developer is not trusted on the device. If you manually try to run the apps on the
device, you will see an Untrusted Developer message.
```

To solve this issue on the device, go to **Settings > General > Profiles** or **Settings > General > Device Management**, depending on the device type and the iOS version. Then trust the developer and allow the apps to be run.

## Editing the Properties of the Silk Test Information Service

You can use the `infoservice.properties` file to specify the port for the Silk Test Information Service or various capabilities. These capabilities are applied each time Silk Test executes a test on the machine on which the Silk Test Information Service is running.

1. Navigate to the directory in which the `infoservice.properties.sample` file is located.

For example, on a Mac, navigate to `~/ .silksilktest/conf/`. On a Windows machine, navigate to `C:\ProgramData\Silk\SilkTest\conf`.

2. Rename the file `infoservice.properties.sample` to `infoservice.properties`.
3. Specify a port that is not in use.

The default port for the Silk Test Information Service is 22901.

4. To specify capabilities, add the following line to the `infoservice.properties` file:

```
customCapabilities=<custom_capability_1>;<custom_capability_2>;...
```

#### Example: Running an iOS Simulator in a Specified Language

To always run a specific iOS Simulator on a Mac in the same language, for example Japanese, specify the custom capabilities *language* and *locale*. To do so, add the following line to the `infoservice.properties` file:

```
customCapabilities=language=ja;locale=ja_JP
```

## Uninstalling the Silk Test Information Service from a Mac

To uninstall the Silk Test information service (information service) from a Mac, for example if you no longer want to execute tests against Apple Safari on the Mac:

1. Create a new shell file, for example `uninstallInfoService.sh`.
2. Type the following code into the new file:

```
#!/bin/sh

if launchctl list | grep com.borland.infoservice ; then
    launchctl unload /Library/LaunchAgents/com.borland.infoservice.plist
    echo "unloading Launch Daemon"
fi

if [ -d "/Applications/Silk" ]
then
    sudo rm -rf /Applications/Silk
fi

if [ -f "/Library/LaunchAgents/com.borland.infoservice.plist" ]
then
    sudo rm /Library/LaunchAgents/com.borland.infoservice.plist
fi

if [ -f "/usr/local/bin/ideviceinstaller" ]
then
    sudo rm /usr/local/bin/ideviceinstaller
fi

exit 0
```

3. In the command line, type `chmod +x uninstallInfoService.sh` to make the shell file executable.
4. Execute the shell file from the command line.


## Recommended Settings for iOS Devices

To optimize testing with Silk Test, configure the following settings on the iOS device that you want to test:

- To make the testing reflect the actions an actual user would perform, disable AutoFill and remembering passwords for Apple Safari. Tap **Settings** > **Safari** > **Passwords & AutoFill** and turn off the **Names and Passwords** setting.

- The iOS device must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings > General > Passcode Lock**.

## Running Existing Scripts on iOS Using XCUITest

 **Attention:** Prior Silk Test versions used *Instruments* to automate iOS devices. With iOS 9.3, Apple has replaced the support for Instruments with support for the *XCUITest* framework, causing Silk Test to also no longer support *Instruments*. Because of this change, existing iOS test scripts might break when executed from the current version of Silk Test.

- The behavior of the `classname` attribute in XCUITest is different to the behavior in Instruments. In most cases, Silk Test will automatically handle this change. However, if an existing test script breaks because of such a `classname` attribute, you will have to record a new locator for the corresponding object.
- The object hierarchy has changed.

## Testing an Installed App

To test a native mobile app that is already installed on a device, an Emulator, or a Simulator, specify the app in the connection string.

1. Open an existing project that tests a native mobile app.
2. Open the **Edit Application Configuration** dialog box.
3. Replace the existing app in the connection string with one of the following:
  - If you want to test an iOS app, replace the app with the *bundleId*, for example replace `app=MyApp.ipa` with `bundleId=silktest.InsuranceMobile`.
  - If you want to test an Android app, replace the app with the *appActivity* and the *appPackage*. For example, replace `app=MyApp.apk` with `appActivity=.LoginActivity;appPackage=silktest.insurancemobile`.

For additional information, see [Connection String](#).

## Recording Mobile Applications

Once you have established the connection between Silk Test and a mobile device or an emulator, you can record the actions that are performed on the device. To record mobile applications, Silk Test uses a **Recording** window that provides the following functionality:

- Displays the screen of the mobile device or Android emulator which you are testing.
- When you perform an action in the **Recording** window, the same action is performed on the mobile device.
- When you interact with a control on the screen, the **Recording** window preselects the default action.
  - If the default action is a `Click`, and you left-click on the control, the action is executed. You can perform a right-click to show a list of the available actions against the control. You can then select the action that you want to perform and click **OK**.
  - If the default action is not a `Click`, a list of all the available actions against the control displays, and you can select the action that you want to perform or simply accept the preselected action by clicking **OK**.

When you have selected an action from the list, you can type values for the parameters of the selected action into the parameter fields. Silk Test automatically validates the parameters.

- During recording, Silk Test displays the mouse position next to the recording window. You can toggle the location to switch between displaying the absolute mouse position on the device display and the mouse position in relation to the active object.

- When you pause the recording, you can perform actions in the screen which are not recorded to bring the device into a state from which you want to continue recording.
- When you stop recording, a script is generated with your recorded actions, and you can proceed with replaying the test.

## Selecting the Mobile Device for Test Replay

You can define the mobile device that is used for the replay of a test in the following ways:

- If you execute a script from the command line or from a Continuous Integration (CI) server, specify the connection string in the application configuration of the script.
- If you execute a test from Silk Central, specify the mobile device in the **Mobile Device Selection** area of the **Deployment** tab of the execution definition in Silk Central instead of specifying a connection string. For additional information, refer to the [Silk Central Help](#).

You can use the connection string to specify a specific mobile device, or you can filter a subset of the available devices, for example if you have a device pool. The first matching device is used for replay. If not specified otherwise, mobile devices are matched by using the following rules, with declining priority:

- Matching mobile devices connected to the local machine are preferred over mobile devices connected to remote locations.
- If the browser type is specified in the connection string, newer browser versions are preferred over older versions.
- Newer platforms are preferred over older platforms.
- A physical device is preferred to an Emulator or Simulator.
- A device with a device name that is alphabetically later is preferred. For example, a device named "iphone 6" is preferred to a device named "iphone 5".

### Example: Connection string for an app on an Android device that is connected to a remote machine

To test the app `MyApp.apk` on an Android device that is connected to a remote machine, the connection string would look like the following:

```
"platformName=Android;deviceName=MotoG3;host=http://10.0.0.1;app=MyApp.apk"
```

### Example: Connection string for an app on an iOS Simulator on a Mac


```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=10.0.0.1;app=MyApp.ipa;isSimulator=true"
```

## Using Devices from Mobile Center

Mobile Center is a mobility gateway that enables you to manage the testing of your mobile devices.


To access the devices that are managed through the Mobile Center from Silk Test, perform the following actions:

1. Integrate Silk Test with Silk Central.  
For additional information, see *Integrating Silk Test with Silk Central*.
2. Configure Silk Central to use Mobile Center.


 **Note:** While installing Mobile Center, ensure that the appropriate Android SDK version is used. Ensure that the same version is used in Silk Test by setting the environment variable `SILK_ANDROID_HOME`, for example to `C:\Users\\AppData\Local\Android\android-sdk`. For additional information, refer to the *Silk Central Help*.


3. To test on iOS, ensure that the following IPA files are signed:


- HP4M-Agent.ipa
- HPMC-AgentLauncher.ipa
- WebDriverAgentRunner-Runner.ipa

 **Note:** Silk Test does not support testing iOS simulators through Mobile Center.

In the **Select Applications** dialog, you can now select the Mobile Center device on which you want to test.

 **Note:** You cannot test a mobile device with both Silk Test and Mobile Center at the same time. Restart a mobile device that you have tested with Silk Test, if you want to continue testing the device from Mobile Center.

 **Note:** When testing on a device that is managed through the Mobile Center, Silk Test does not support using the methods `typeKeys` or `setText` to type key codes like **ENTER**. Additionally, Silk Test does not support pressing the **Home** button on iOS devices.

 **Note:** When testing on an Android Emulator, disable the GPU HW Acceleration.

## Using SauceLabs Devices

SauceLabs provides an automated testing platform, enabling you to test on various mobile devices and mobile platform versions without having to purchase and maintain your own infrastructure.

To access SauceLabs devices through Silk Central, perform the following actions:

1. Ensure that Silk Test is integrated with Silk Central.  
For additional information, see *Integrating Silk Test with Silk Central*.
2. Ensure that Silk Central is configured to use SauceLabs.  
For additional information, refer to the *Silk Central Help*.

In the **Select Applications** dialog, you can now select the SauceLabs device on which you want to test.

## Editing Remote Locations

You can use the **Remote Locations** dialog box to add any browsers and mobile devices on a remote location to the set of applications that you can test.

1. Open the **Remote Locations** dialog box.
  - If you are using Silk Test Workbench, click **Tools > Edit Remote Locations**.
  - If you are using Silk4J, click **Silk4J > Edit Remote Locations**.
  - If you are using Silk4NET, click **Silk4NET > Edit Remote Locations**.
  - If you are using Silk Test Classic, click **Options > Edit Remote Locations**.
2. To add an additional remote location, perform the following actions:
  - a) Click on the arrow to the right of **Add Location** to specify whether you want to add a remote location which is using the Silk Test Information Service, or Silk Central.





**Note:** You can only configure one Silk Central as a remote location. If you have already configured the integration with Silk Central, Silk Central is listed in the remote locations list.

- b) Click **Add Location**. The **Add Location** dialog box appears.
- c) Type the URL of the remote location and the port through which Silk Test connects to the information service on the remote machine into the **Host** field.

The default port is 22901.

- d) *Optional:* Edit the name of the remote location in the **Name** field.

3. To edit an existing remote location, click **Edit**.
4. To remove a remote location, click **Remove**.
5. *Optional:* To reduce the amount of browsers and devices in the **Select Application** dialog, click **Do not show devices and browsers from this location**. The installed browsers and connected devices of the remote location will no longer be displayed in the **Select Application** dialog. By default, all installed browsers and connected devices of all remote locations are displayed in the **Select Application** dialog.
6. Click **OK**.

When you have added a remote location, the browsers that are installed on the remote location, including Apple Safari on a Mac, are available in the **Web** tab of the **Select Application** dialog box, and the mobile devices that are connected to the remote location are available in the **Mobile** tab of the **Select Application** dialog box.

## Connection String for a Mobile Device

The *connection string* specifies which mobile device is used for testing. When performing mobile testing, Silk Test uses the connection string to connect to the mobile device. The connection string is typically part of the application configuration. You can set the connection string when you configure your application under test. To change the connection string, you can use the **Edit Application Configuration** dialog box.



**Note:** If you execute a test from Silk Central, specify the mobile device in the **Mobile Device Selection** area of the **Deployment** tab of the execution definition in Silk Central instead of specifying a connection string. For additional information, refer to the [Silk Central Help](#).

You can use the connection string to specify a specific mobile device, or you can filter a subset of the available devices, for example if you have a device pool. The first matching device is used for replay. If not specified otherwise, mobile devices are matched by using the following rules, with declining priority:

- Matching mobile devices connected to the local machine are preferred over mobile devices connected to remote locations.
- If the browser type is specified in the connection string, newer browser versions are preferred over older versions.
- Newer platforms are preferred over older platforms.
- A physical device is preferred to an Emulator or Simulator.
- A device with a device name that is alphabetically later is preferred. For example, a device named "iphone 6" is preferred to a device named "iphone 5".

The following components are available for the connection string:

Component	Description
deviceName	The name of the mobile device. When testing on a physical mobile device, the device ID can be used instead. Supports wildcards. Case-insensitive.
platformName	Android or iOS. Required.
deviceId	<i>Optional:</i> The ID of the mobile device. Can be used instead of the device name, when testing on a physical mobile device. Supports wildcards. Case-insensitive.

Component	Description
platformVersion	<i>Optional:</i> The Android or iOS version. Specify the version to test only on mobile devices that have a specific Android or iOS version. Supports wildcards. Case-insensitive.
browserVersion	<i>Optional:</i> Can be used in combination with the browser type to test only on the specified browser version. Supports wildcards. Case-insensitive.
host	<i>Optional:</i> If not set, any specified remote location can be used as the host. Supports wildcards. Case-insensitive.
<ul style="list-style-type: none"> <li>app</li> <li>appActivity</li> <li>appPackage</li> </ul>	Required for testing native mobile applications on Android. Either the full path to the app, or a combination of <i>appActivity</i> and <i>appPackage</i> . For example <code>app=MyApp.apk</code> or <code>appActivity=.LoginActivity;appPackage=silktest.insurancemobile</code>
<ul style="list-style-type: none"> <li>app</li> <li>bundleId</li> </ul>	Required for testing native mobile applications on iOS. Either the full path to the app or the <i>bundleId</i> . For example <code>app=MyApp.ipa</code> or <code>bundleId=silktest.InsuranceMobile</code>
noReset	<i>Optional:</i> Can be set when testing native mobile applications. Is only valid if the <i>app</i> is specified. True if the app should not be reinstalled before testing. False if the app should be reinstalled before testing. The default value is False.
isSimulator	<i>Optional:</i> Used to specify that the test should only be executed on an iOS Simulator. The device name can be used instead.
isPhysicalDevice	<i>Optional:</i> Used to specify that the test should only be executed on a physical device. The device name can be used instead.

When using a pool of devices and to find out which device is actually used for testing, you can use the return value of the `generateConnectionString` method of `MobileDevice` class.

### Testing a mobile web application on a mobile device or on an Android Emulator

When testing a mobile web application on a mobile device or on an Android Emulator, the connection string consists of the following parts:

1. The mobile device name, for example `MotoG3`, or the device ID, for example `11111111`.



**Note:** If the device name is unique, Micro Focus recommends to use the device name in the connection string, because the device ID is less readable.

2. The platform name.
3. The browser version. This can only be used in combination with setting the browser type
4. The IP address or the host name of a specific remote machine, for example `10.0.0.1`. You can also use the name of a remote location that is specified in the **Edit Remote Locations** dialog box as the host name, for example `MyRemoteLocation`. When using the remote location name, you can also use wildcards. To test an Android device that is connected to the local machine, specify the IP address or the host name of the local machine.

#### Example: Connection string for any available Android device

```
"platformName=Android"
```

#### Example: Connection string for a browser on an Android device that is connected to the local machine

To test a mobile browser on an Android device that is connected to the local machine, the connection string should look similar to the following:

```
"deviceName=MotoG3;platformName=Android;host=localhost"
```

or

```
"platformName=Android;deviceId=11111111;host=localhost"
```

**Example: Connection string for a browser on an Android device that is connected to a remote machine**

To test a mobile browser on a remote Android device, the connection string should look similar to the following:

```
"deviceName=MotoG3;platformName=Android;host=10.0.0.1"
```

```
"deviceName=MotoG3;platformName=Android;host=MyRemoteLocation*"
```

**Example: Connection string for a browser on an iOS device that is connected to a Mac**

To test a mobile browser on a remote iOS device, the connection string would look like the following:

```
"deviceName=myiPhone6;platformName=iOS;host=10.0.0.1"
```

### Testing a native mobile application on a mobile device or on an Android Emulator

When testing a native mobile application on a mobile device or on an Android Emulator, the connection string consists of the following parts:

1. The mobile device name, for example *MotoG3*, or the device ID, for example *11111111*.



**Note:** If the device name is unique, Micro Focus recommends to use the device name in the connection string, because the device ID is less readable.

2. The platform name.
3. The IP address or the host name of a specific remote machine, for example *10.0.0.1*. You can also use the name of a remote location that is specified in the **Edit Remote Locations** dialog box as the host name, for example *MyRemoteLocation*. When using the remote location name, you can also use wildcards. To test an Android device that is connected to the local machine, specify the IP address or the host name of the local machine.
4. The name of the file of the app that you want to test, or the URL of the file, if the file is located on a web server. For example *C:/MyApp.apk* or *MyApp.ipa*.
  - Android apps are always *.apk* files.
  - iOS apps on a real device are always *.ipa* files.
  - iOS apps on a Simulator are either a zipped file or a directory with the name *app*.

**Example: Connection string for an app on an Android device that is connected to a remote machine**

To test the app *MyApp.apk* on an Android device that is connected to a remote machine, the connection string would look like the following:

```
"platformName=Android;deviceName=MotoG3;host=http://10.0.0.1;app=MyApp.apk"
```

**Example: Connection string for an app on an iOS device that is connected to a Mac**

To test the app *MyApp.ipa* on an iOS device that is connected to a remote machine, the connection string would look like the following:

```
"platformName=iOS;deviceName=MyiPhone;host=http://10.0.0.1;app=MyApp.ipa"
```

## Testing a mobile web application on an iOS Simulator

When testing a mobile web application on an iOS Simulator, the connection string consists of the following parts:

1. The platform name, which is iOS.
2. The platform version, for example 10.0.
3. The mobile device name, for example iPhone6.
4. The IP address or the host name of the Mac, on which the iOS Simulator is running.

### Example: Connection string for a browser on an iOS Simulator on a Mac

```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=10.0.0.1;isSimulator=true"
```

## Testing a native mobile application on an iOS Simulator

When testing a native mobile application on an iOS Simulator on a Mac, the connection string consists of the following parts:

1. The platform name, which is iOS.
2. The platform version, for example 10.0.
3. The mobile device name, for example iPhone6.
4. The IP address or the host name of the remote machine, for example 10.0.0.1.
5. The name of the app that you want to test, for example `MyApp.ipa`.

### Example: Connection string for an app on an iOS Simulator on a Mac

```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=10.0.0.1;app=MyApp.ipa;isSimulator=true"
```

# Interacting with a Mobile Device

To interact with a mobile device and to perform an action like a swipe in the application under test:

1. In the **Recording** window, click **Show Mobile Device Actions**. All the actions that you can perform against the mobile device are listed.
2. Select the action that you want to perform from the list.
3. To record a swipe on an Android device or emulator, move the mouse while holding down the left mouse button.
4. Continue with the recording of your test.

# Releasing a Mobile Device

When recording or playing back a test against a mobile device, the Open Agent instance takes ownership of the device. By doing so, the Open Agent is preventing other Silk Test users from using the device. To enable other Silk Test users to use the device after you have finished recording or replaying tests on the device, Silk Test automatically releases the device when the Silk Test client is closed, when an unattended test process finishes, or when the Open Agent is closed. You can also manually release the device.



**Note:** Releasing a mobile device will close the application under test (AUT) on the mobile device.

# Releasing a Mobile Device After Recording

Release a mobile device after recording to enable other Silk Test users to test on the device.

To release a mobile device after you have finished recording, perform one of the following actions:

- Stop the Open Agent from the System Tray.
- Close Silk Test. The device is only released by this action when parallel testing is enabled.



**Note:** Releasing a mobile device will close the application under test (AUT) on the mobile device.

# Releasing a Mobile Device After Replay

Release a mobile device after replay to enable other Silk Test users to test on the device.

To manually release a mobile device after replaying is complete, you can also perform one of the following:

- If you have tested a mobile web application, use the `Close` method or the `CloseSynchron` method of the `BrowserApplication` class. For additional information on these methods, refer to the API documentation.

```
webBrowser.close();
```

- If you have tested a mobile native application, use the `CloseApp` method of the `MobileDevice` class.

For example, type the following:

```
MobileDevice mobileDevice = desktop.find("//MobileDevice");  
mobileDevice.closeApp();
```

- Add the `desktop.detachAll()` statement to the test script.

A mobile device is automatically released if one of the following conditions is met:

- The Open Agent is closed.
- The test process stops during unattended testing. The device is only released by this action when parallel testing is enabled.
- Silk Test is closed. The device is only released by this action when parallel testing is enabled.



**Note:** Releasing a mobile device will close the application under test (AUT) on the mobile device.

# Troubleshooting when Testing Mobile Applications

## Why does the Select Application dialog not display my mobile devices?

If Silk Test does not recognize a mobile device or emulator, the **Mobile** tab in the **Select Application** dialog does not display the device or emulator. Additionally, the **Web** tab of the **Select Application** dialog does not display the mobile browsers that are installed on the device or emulator.

Silk Test might not recognize a mobile device or emulator for one of the following reasons:

Reason	Solution
The emulator is not running.	Start the emulator.
The Android Debug Bridge (adb) does not recognize the mobile device.	To check if the mobile device is recognized by adb: <ol style="list-style-type: none"><li>1. Navigate to the Android Debug Bridge (adb) in the Android SDK installation folder. If the Android SDK is not installed, navigate to <code>C:\Program Files</code></li></ol>

Reason	Solution
	<p>(x86)\Silk\SilkTest\ng\Mobile\windows\AndroidTools\platform-tools to use the adb that is installed with Silk Test.</p> <ol style="list-style-type: none"> <li>Hold <b>Shift</b> and right-click into the <b>File Explorer</b> window.</li> <li>Select <b>Open command window here</b>.</li> <li>In the command window, type <code>adb devices</code> to get a list of all attached devices.</li> <li>If your device is not listed, check if USB-debugging is enabled on the device and if the appropriate USB driver is installed.</li> <li>If you get an error, for example <code>adb server is out of date</code>, ensure that the adb version in <code>C:\Program Files (x86)\Silk\SilkTest\ng\Mobile\windows\AndroidTools\platform-tools</code> is the same as the adb version of your local Android SDK. For additional information, see <i>What can I do if the connection between the Open Agent and my device is unstable?</i></li> </ol>
The version of the operating system of the device is not supported by Silk Test.	For information on the supported mobile operating system versions, refer to the <a href="#">Release Notes</a> .
The USB driver for the device is not installed on the local machine.	Install the USB driver for the device on the local machine. For additional information, see <i>Installing a USB Driver</i> .
USB-debugging is not enabled on the device.	Enable USB-debugging on the device. For additional information, see <i>Enabling USB-Debugging</i> .



**Note:** If all previous solutions do not work, you could try to restart the device.

### Why does Silk Test search for a URL in Chrome for Android instead of navigating to the URL?

Chrome for Android might in some cases interpret typing an URL into the address bar as a search. As a workaround you can manually add a command to your script to navigate to the URL.

### What do I do if the adb server does not start correctly?

When the Android Debug Bridge (adb) server starts, it binds to local TCP port 5037 and listens for commands sent from adb clients. All adb clients use port 5037 to communicate with the adb server. The adb server locates emulator and device instances by scanning odd-numbered ports in the range 5555 to 5585, which is the range used by emulators and devices. Adb does not allow changing those ports. If you encounter a problem while starting adb, check if one of the ports in this range is already in use by another program.

For additional information, see <http://developer.android.com/tools/help/adb.html>.

### What can I do if the connection between the Open Agent and my device is unstable?

If you have installed the Android SDK or another tool that uses the Android Debug Bridge (adb), an additional adb server might be running in addition to the one that is used by Silk Test. If the running adb servers have different versions, the connection between the Open Agent and the device might become unstable or even break.

To avoid version mismatch errors, specify the path to the Android SDK directory by setting the environment variable `SILK_ANDROID_HOME`, for example to `C:\Users\<user>\AppData\Local\Android`

`\android-sdk`. If the information service was running during this change, use the Windows Service Manager to restart the Silk Test information service with the updated environment variable. If the variable is not set, Silk Test uses the adb version that is shipped with Silk Test.

### Why do I get the error: Failed to allocate memory: 8?

This error displays if you are trying to start up the emulator and the system cannot allocate enough memory. You can try the following:

1. Lower the RAM size in the memory options of the emulator.
2. Lower the RAM size of Intel HAXM. To lower the RAM size, run the `IntelHaxm.exe` again and choose **change**.
3. Open the **Task Manager** and check if there is enough free memory available. If not, try to free up additional memory by closing a few programs.

### Why do I get the error "Silk Test cannot start the app that you have specified" during testing on an iOS device?

This error might display for one or more of the following reasons:

Reason	Solution
The iOS device is not in developer mode.	<p>You can enable the developer mode in one of the following two ways:</p> <ul style="list-style-type: none"> <li>• Connect the device to a Mac on which Xcode is installed, and start the app that you want to test on the device.</li> <li>• Add your provisioning profiles to the device.               <ol style="list-style-type: none"> <li>1. Open Xcode.</li> <li>2. Select <b>Window &gt; Devices</b>.</li> <li>3. Right-click on the iOS device.</li> <li>4. Select <b>Show Provisioning Profiles</b>.</li> <li>5. Add your provisioning profiles.</li> </ol> </li> </ul>
You have recently updated the iOS version of the device.	<ol style="list-style-type: none"> <li>1. Open Xcode.</li> <li>2. Select <b>Window &gt; Devices</b>.</li> <li>3. Wait until Xcode has processed the symbol files.</li> </ol>
UI automation is not enabled on the iOS device.	<ol style="list-style-type: none"> <li>1. Select <b>Settings &gt; Developer</b>.</li> <li>2. Activate <b>Enable UI Automation</b>.</li> </ol>
The <b>Web Inspector</b> is not activated on the iOS device, while you are trying to test a mobile web application.	<ol style="list-style-type: none"> <li>1. Click <b>Settings &gt; Safari &gt; Advanced</b>.</li> <li>2. Activate the <b>Web Inspector</b>.</li> </ol>
The app that you want to test was not built for the iOS version of the iOS device on which you are testing.	<p>Use Xcode to build the app for the iOS version of the device.</p>
The <b>Software Update</b> dialog box is currently open on the iOS device.	<p>Close the dialog box and disable automatic software updates:</p> <ol style="list-style-type: none"> <li>1. Select <b>Settings &gt; App and iTunes Stores &gt; AUTOMATIC DOWNLOADS</b>.</li> <li>2. Deactivate <b>Updates</b>.</li> </ol>

### Why does my Android device display only the Back button in the dynamic hardware controls?

If the Android or the Android Emulator is screen-locked when you start testing, the device or Emulator might display only the button **Back** in the dynamic hardware controls.

To solve this issue, stop the Open Agent, restart the device, and change the device settings to no longer lock the screen.

### Why does my Android device or emulator no longer display a keyboard?

To support unicode characters, Silk Test replaces the standard keyboard with a custom keyboard. When testing is finished, the original keyboard is restored. If an error occurs during testing, the custom keyboard might still be active and cannot be replaced.

To solve this issue, manually reset the keyboard under **Settings > Language & input > Current Keyboard**.

### Why does my device not respond during testing?

If the device, emulator, or Simulator is screen-locked when you start testing, and Silk Test is unable to unlock the screen, the device, emulator, or Simulator might stop responding to any actions.

To solve this issue, stop the Open Agent and change the device settings to no longer lock the screen.

### Why can I not install the Information Service on a Mac?

When the **Allow apps downloaded from** setting in the **General** tab of the **Security & Privacy** system preferences pane is set to **Mac App Store and identified developers**, which is the default value, the following error message appears when opening the Information Service setup:

"SilkTestInformationService<version>.pkg" can't be opened because it is from an unidentified developer.

To solve this issue, use one of the following:

- Right-click the setup file and select **Open**. A warning message will appear, but you will still be able to open the file.
- Set the **Allow apps downloaded from** setting to **Anywhere**.
- After attempting to open the file, navigate to the **General** tab of the **Security & Privacy** system preferences pane and click **Open Anyway**.

### Why is the Recording window black when recording an Android app?

Android apps that require a higher level of security, for example apps that handle financial transactions, might have the [FLAG\\_SECURE](#) flag set, which prevents Silk Test from capturing the app. Silk Test relies on screenshots or on a video of the Android device during recording and will display a black screen of the device in the **Recording** window, if the Android app that you are testing has this flag set. To test such an app with Silk Test, you have to contact the app development team, and ask them to un-set the `FLAG_SECURE` flag during testing.

### Why does Silk Test not show a video when testing on an Android emulator?

If the emulator is using the graphic card of your computer for better rendering, the video capturing of Silk Test might not work. To solve this, emulate the graphics in software:

1. Open the **Android Virtual Device Manager**.
2. Click **Edit** in the **Actions** column of the emulator.
3. Select **Software** from the list in the **Emulated Performance** area of the **Virtual Device Configuration** dialog.



## What can I do if Silk Test does not show a video when testing in a cloud environment?

When testing in a cloud environment, showing a video might not work when recording or replaying a test, for example because required ports are not open.

To solve this issue, you can specify a list of WebDriver host URLs in the `infoservice.properties` file. For information on how to access this properties file, see *Editing the Properties of the Silk Test Information Service*. Add the option `infoservice.disableScreencastHosts` to the file, by typing the following:

```
infoservice.disableScreencastHosts=<URL_1>,<URL_2> , ...
```

For example:

```
infoservice.disableScreencastHosts=http://my-webdriver-server-url.com:80/wd/hub
```

You can specify URL patterns like `*my-webdriver-server-url.com` by using asterisks (\*) as wildcards.

Silk Test will show a series of screenshots instead of a video when recording and replaying on the specified hosts.

## How can I change the installed version of Xcode?

If the version of Xcode that you are using is not supported by Silk Test, for example when you upgrade to the latest version of Xcode, an error message might appear when testing on iOS.

To replace the installed version of Xcode with a supported version, download a supported Xcode version from <https://developer.apple.com/download/more/>, and replace the unsupported version with the downloaded version. For information about the supported Xcode versions, refer to the [Release Notes](#).

## What can I do if my Mac runs out of disk space?

Silk Test uses Instruments to automate iOS devices. This tool creates large log files in the `/Library/Caches/com.apple.dt.instruments` directory, which might fill up disk space on the Mac. To solve this issue, Micro Focus recommends regularly deleting these log files, either manually or by using a cronjob. For example, to delete the files each day at the same time, you could do the following:

1. Type `sudo crontab -e` into a Terminal. This opens an editor in which you can edit the crontab for root.
2. Add the following line to the crontab:

```
0 2 1 * * find /Library/Caches/com.apple.dt.instruments -mtime +10 -delete
```
3. Save the crontab.

In this example, all log files that are older than ten days will be deleted each day at 2 AM from the directory.

## Why does my test fail with the error message "Unable to sign WebDriver Agent for testing"?

When testing on a physical iOS device, this error usually means that during the build process the `WebDriverAgent` app could not be signed or that there is a problem with the provisioning profile.

You can check the actual problem with the following commands, which have to be executed at the Mac machine to which the device is connected:

```
cd /Applications/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -
destination 'id=<udid>' test
```

Verify that the folder `Resources` exists under `/Applications/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver` and that the folder contains the file `WebDriverAgent.bundle`. If not, create this folder and an empty `WebDriverAgent.bundle` file, for example by using the following command:

```
mkdir -p Resources/WebDriverAgent.bundle
```

## What can I do to prevent the Developer Tools Access from requesting to take control of another process?

When starting the execution of a test on iOS, a message box stating the following might appear: Developer Tools Access needs to take control of another process for debugging to continue. Type your password to allow this.

To avoid getting this message, execute the following command in a Terminal:

```
sudo /usr/sbin/DevToolsSecurity --enable
```

## Why are the rectangles wrong while testing a mobile web application on an iPad?

If the rectangles around controls are offset when testing a mobile web application on an iPad, you might have multiple browser tabs open and the Tab bar might be displayed. To fix this issue, close all tabs except one.

## Why can I no longer record or replay tests on my device after updating Silk Test?

When updating to a new version of Silk Test, some Appium apps on any physical mobile devices that have already been used for mobile testing with the previous version of Silk Test are updated automatically. If for any reason these apps are not automatically updated, you might experience difficulties when trying to record or replay tests on the device.

If you are experiencing such issues on a specific Android device after updating Silk Test, manually uninstall the following apps from the device:

- Appium Android Input Manager
- Appium Settings
- io.appium.uiautomator2.server
- io.appium.uiautomator2.server.test
- Unlock

If you are experiencing such issues on a specific iOS device after updating Silk Test, manually uninstall the WebDriverAgentRunner from the device.

## Why can I not record a mobile application?

Silk Test uses Appium to test mobile applications. Some network proxy settings set in Appium might interfere with recording Silk Test. You could try to deactivate the network proxy settings on the mobile device or Emulator.

# How Can I Use Chrome for Android to Replay Tests?

By default you can use the **Select Browser** dialog box to select the browser during replay.

If you execute a script from the command line or from a Continuous Integration (CI) server, you can specify the connection string in the application configuration of the script. To overwrite the browser that is specified in the application configuration, use the *silktest.configurationName* environment variable.

You can also use the property `browsertype` of the `BrowserApplication` class to set the type of the browser that is used during replay. However, the `browsertype` does not include an explicit value for Chrome for Android.

To specify that you want to use Chrome for Android as the browser, on which a test is replayed, set the `browsertype` to *GoogleChrome* and specify Android as the platform. When Android is specified, Silk Test uses Chrome for Android instead of Google Chrome on a desktop machine to execute the test.

## Examples

The following code sample shows how you can set the base state for a test to use Chrome for Android on a Nexus 7 by using the `silktest.configurationName`:

```
SET
silktest.configurationName="platformName=Android;deviceName=Nexus
7;host=10.0.0.1 - Chrome"
```


The following Java code sample shows how you can set the base state for a test to use Chrome for Android by using the `browserType`:

```
BrowserBaseState baseState = new
BrowserBaseState(BrowserType.GoogleChrome, "demo.borland.com/
InsuranceWebExtJS/");
baseState.setConnectionString("platformName=Android");
baseState.execute(desktop);
```

## Limitations for Testing Mobile Web Applications


The support for playing back tests and recording locators on mobile browsers is not as complete as the support for the other supported browsers. The known limitations for playing back tests and recording locators for mobile web applications are:

- The following classes, interfaces, methods, and properties are currently not supported for mobile web applications:
  - `BrowserApplication` class.
    - `CloseOtherTabs` method
    - `CloseTab` method
    - `ExistsTab` method
    - `GetActiveTab` method
    - `GetSelectedTab` method
    - `GetSelectedTabIndex` method
    - `GetSelectedTabName` method
    - `GetTabCount` method
    - `ImageClick` method
    - `OpenTab` method
    - `SelectTab` method
  - `DomElement` class.
    - `DomDoubleClick` method
    - `DomMouseMove` method
    - `GetDomAttributeList` method
  - `IKeyable` interface.
    - `PressKeys` method
    - `ReleaseKeys` method
- Silk Test does not support testing HTML frames and iFrames with Apple Safari on iOS.
- Recording in landscape mode is not supported for emulators that include virtual buttons in the system bar. Such emulators do not correctly detect rotation and render the system bar in landscape mode to the right of the screen, instead of the lower part of the screen. However, you can record against such an emulator in portrait mode.
- Only HTML attributes in the HTML DOM are supported in XPath expressions for mobile applications. Silk Test does not support properties in XPath expressions.

- If you are testing a mobile web application on Android, Silk Test does not support zooming.
- The following JavaScript alert-handling methods of the `BrowserWindow` class do not work when testing on the Original Android Stock (AOSP) Browser:
  - `AcceptAlert` method
  - `DismissAlert` method
  - `GetAlertText` method
  - `IsAlertPresent` method
- At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac.
- Before starting to test a mobile web application, ensure that no browser tab is open.
-  **Tip:** On iPads you can disable tabs in Apple Safari. Navigate to **Settings** > **Safari** and disable **Show Tab Bar** to do so.
- While testing a mobile web application, you can only have one browser tab open.
- Silk Test does not support testing mobile web applications that are opened by a native mobile application.

## Limitations for Testing Native Mobile Applications

The known limitations for playing back tests and recording locators on native mobile applications (apps) are:

- The following classes, interfaces, methods, and properties are currently not supported for native mobile applications:
  - `IKeyable` interface.
    - `PressKeys` method
    - `ReleaseKeys` method
  - `MobileDevice` class.
    - When testing a native mobile application on iOS, the `SetLocation` method is not supported.
    - When testing a native mobile application on an Android version prior to Android 6.0, you have to enable **Allow mock locations** to use the `SetLocation` method. To do so, open the settings of the Android device or emulator and tap **Developer Options**.
    - When testing a native mobile application on Android 6.0 or later, you have to set the app to **Appium Settings** to use the `SetLocation` method. To do so, open the settings of the Android device or emulator and tap **Developer Options** > **Select mock location app**. Then choose **Appium Settings**.
-  **Note:** The **Appium Settings** entry is only available if you have already executed a test with Appium on the Android device or emulator.
- When testing on iOS, the `Find` method does not work in the following cases:
  - If the `path` attribute is set.
  - If an attribute value is empty.
- When testing on iOS, the `getValue` method of the `XCUIElementTypeSwitch` class returns the strings `false` or `true` depending on the checked state, instead of returning the strings `0` and `1`.
- Recording in landscape mode is not supported for Android emulators that include virtual buttons in the system bar. Such emulators do not correctly detect rotation and render the system bar in landscape mode to the right of the screen, instead of the lower part of the screen. However, you can record against such an emulator in portrait mode.
- Only HTML attributes in the HTML DOM are supported in XPath expressions for mobile applications. Silk Test does not support properties in XPath expressions.

- At any given point in time, you can test on multiple physical iOS devices that are connected to the Mac, but only on one iOS Simulator that is running on the Mac. With Silk Test 17.5 Hotfix 1 or later, you are no longer required to use multiple user sessions on a Mac to test mobile applications on iOS.
- Silk Test does not support text recognition when testing native mobile applications on both Android and iOS.

Text recognition includes the following methods:

- `TextCapture`
- `TextClick`
- `TextExists`
- `TextRectangle`
- Silk Test does not support testing native mobile applications with multiple web views.
- When testing on iOS, the state of the `isVisible` property is always true, even if the element is not visible.
- When testing on iOS, a swipe action with multiple steps swipes to a point, releases the mouse pointer and then swipes to the next point. On prior versions of iOS, the action does not release the mouse pointer between the swipes.
- When testing on iOS, Silk Test does not support any multi-touch actions except pinch.
- When testing on iOS, Silk Test does not support the `PinchIn` method.
- When testing on iOS, you can only accept or dismiss alert dialog boxes. If no **Cancel** button is available and Silk Test cannot dismiss the dialog, the default action is to accept the dialog.
- When testing on Android, Silk Test does not provide automated synchronization for controls of the [Animation](#) class.
- When testing toasts on Android, the following limitations apply:
  - During recording, Silk Test always displays the rectangle for the toast in the lowest quarter of the **Recording** window, independent of the actual position of the toast.
  - During recording and replay, the detection of a toast by Silk Test always has a duration of five seconds, even if the toast appears in a shorter time period.
- When testing on iOS, Silk Test does not provide automated synchronization for controls that call the `UIView.animate` function or the `UIView.animateWithDuration` function.

You can workaround this issue by increasing the speed of the animation in the app delegate:

```
func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {
    //...
    if NSProcessInfo.processInfo().environment["automationName"] == "Silk
Test" {
        // Speed animations up (recommended)
        window!.layer.speed = 100;
    }
}
```

Micro Focus does not recommend disabling such animations completely, as this might change the applications behavior. However, if speeding up the animation does not resolve the synchronization issue, you could completely disable animations in the app delegate as follows:

```
func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {
    //...
    if NSProcessInfo.processInfo().environment["automationName"] == "Silk
Test" {
        UIView.setAnimationsEnabled(false)
    }
}
```

- When testing on iOS, the following additional limitations apply:
  - You might experience performance decreases while recording and replaying tests.

- Due to internal changes in iOS, the locators of some controls might have changed, and some of your existing tests might break.
- Text fields that are not in focus might not be recognized as text fields. To ensure that text fields are recognized correctly, set the focus on the text fields, for example by clicking on a text field before trying to interact with it.

## Clicking on Objects in a Mobile Website

When clicking on an object during the recording and replay of an automated test, a mobile website presents the following challenges in comparison to a desktop website:

- Varying zoom factors and device pixel ratios.
- Varying screen sizes for different mobile devices.
- Varying font and graphic sizes between mobile devices, usually smaller in comparison to a website in a desktop browser.
- Varying pixel size and resolution for different mobile devices.

Silk Test enables you to surpass these challenges and to click the appropriate object on a mobile website.

When recording a test on a mobile device, Silk Test does not record coordinates when recording a `Click`. However, for cross-browser testing, coordinates are allowed during replay. You can also manually add coordinates to a `Click`. Silk Test interprets these coordinates as the HTML coordinates of the object. To click on the appropriate object inside the `BrowserWindow`, during the replay of a test on a mobile device, Silk Test applies the current zoom factor to the HTML coordinates of the object. The device pixel coordinates are the HTML coordinates of the object, multiplied with the current zoom factor.

If the object is not visible in the currently displayed section of the mobile website, Silk Test scrolls to the appropriate location in the website.

### Example

The following code shows how you can test a `DomButton` with a fixed size of 100 x 20 px in your HTML page.

During replay on a different mobile device or with a different zoom factor, the `DomButton` might for example have an actual width of 10px on the device screen. Silk Test clicks in the middle of the element when using the code above, independent of the current zoom factor, because Silk Test interprets the coordinates as HTML coordinates and applies the current zoom factor.

## Using Existing Mobile Web Tests

Silk Test 17.0 or later uses a different approach to mobile web testing than previous versions of Silk Test. This change might result in your old mobile web tests no longer working on Silk Test 17.0 or later. This topic describes some of the changes that were introduced with Silk Test 17.0 and provides guidance on changing existing mobile web tests with Silk Test 17.0 or later.

The following changes for mobile web testing were introduced with Silk Test 17.0:

- With previous versions of Silk Test, you were able to test on iOS devices that were connected by USB to a Windows machine. With Silk Test 17.0 or later, you can only test on iOS devices that are connected to an OSX machine (Mac).
- If you have tested mobile web applications on an Android device with a previous version of Silk Test, you have to manually remove the proxy from the Android device to test a web application with Silk Test

17.0 or later. Silk Test 17.0 or later no longer requires a proxy, and if the proxy is set, the message Unable to connect to the proxy server displays on the device.

# Index

## A

- Android
  - configuring emulator 7
  - enabling USB-debugging 7
  - hybrid applications 5
  - installed apps, testing 22
  - installing USB drivers 6
  - mobile native applications, prerequisites 4
  - mobile web applications, prerequisites 4
  - parallel testing, tested configurations 9
  - recommended settings 7
  - releasing devices 28
  - releasing devices, recording 29
  - releasing devices, replay 29
  - testing 4
  - troubleshooting 29
- Android emulator
  - configuring 7
- Apple Safari
  - information service, installing 17, 21

## B

- browser type
  - Chrome for Android, setting 34
- browsertype
  - Chrome for Android, setting 34

## C

- capabilities
  - iOS 20
- Chrome for Android
  - browser type, setting 34
- Click
  - mobile web 38
- Configuration Assistant
  - automatic signing 17
- connection string
  - mobile devices 25
- cross-browser testing
  - remote locations, adding 24

## D

- device not connected
  - mobile 29

## E

- editing
  - remote locations 24
- Emulator
  - defining, playback 23
- emulators
  - testing 4

## H

- hybrid applications
  - Android 5
  - iOS 15

## I

- information service
  - editing 20
  - Mac, installing 17, 21
- installed apps
  - Android, testing 22
  - iOS, testing 22
- installing
  - information service, Mac 17, 21
- installing USB drivers
  - Android 6
- iOS
  - apps, preparing for testing 16
  - devices, preparing 16
  - hybrid applications 15
  - information service, installing 17, 21
  - installed apps, testing 22
  - Mac, preparing 17
  - mobile native applications, prerequisites 11
  - mobile web applications, prerequisites 11
  - native app, Simulator 13
  - native app, testing 12
  - recommended settings 21
  - releasing devices 28
  - releasing devices, recording 29
  - releasing devices, replay 29
  - testing 11
  - testing, no developer account 19
  - web app, Simulator 14
  - web app, testing 14
- iOS 10
  - existing scripts, executing 22

## L

- limitations
  - mobile web applications 35
  - native mobile applications 36

## M

- Mac
  - information service, installing 17, 21
- mobile
  - troubleshooting 29
- mobile applications
  - recording 22
  - testing 4
- mobile browsers
  - limitations 35
- Mobile Center
  - enabling 23
- mobile device
  - defining, playback 23
- mobile devices
  - interacting with 28
  - performing actions against 28



- mobile native applications
  - limitations 36
- mobile recording
  - about 22
- mobile testing
  - Android 4
  - connection string 25
  - iOS 11
  - native app, iOS Simulator 13
  - overview 4
  - releasing devices 28
  - remote locations, adding 24
  - web app, iOS 14
  - web app, iOS Simulator 14
- mobile testing devices
  - native app, iOS 12
- mobile web
  - Click 38
  - iOS 14
  - legacy tests 38
- mobile web applications
  - Android, prerequisites 4
  - iOS, prerequisites 11
  - limitations 35

## N

- native mobile applications
  - Android, prerequisites 4
  - iOS, prerequisites 11
  - limitations 36

## P

- parallel testing
  - tested configurations, Android 9
- playback
  - selecting device 23
- prerequisites
  - Android, mobile web applications 4
  - Android, native mobile applications 4
  - iOS, mobile web applications 11
  - iOS, native mobile applications 11

## R

- recording
  - mobile applications 22
  - no image displayed 29
  - releasing devices 29
- releasing devices

- mobile testing 28
  - recording 29
  - replay 29
- remote locations
  - adding 24
  - editing 24
- replay
  - releasing devices 29
  - selecting device 23
- running existing scripts
  - iOS 10 22

## S

- SauceLabs
  - enabling 24
- screencast
  - not working 29
- setting mobile device
  - playback 23
- Silk Central
  - Mobile Center, enabling 23
  - SauceLabs, enabling 24
- Simulator
  - defining, playback 23
  - mobile web applications, testing 14
  - native app, testing 13
  - testing 12

## T

- testing Apple Safari
  - information service, installing 17, 21
- troubleshooting
  - mobile 29

## U

- upload app
  - Mac 4

## V

- video
  - not displayed 29

## X

- xBrowser
  - Chrome for Android, setting 34