MICRO FOCUS

Silk Test 17.0

Testing Mobile Applications

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

2016-05-10

Contents

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 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:

- Enable USB-debugging on the Android device. For additional information, see *Enabling USB-Debugging*
- 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**.

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. 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*.
2. Start the Android emulator or connect the device to the machine on which Silk Test is installed.
3. 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*.
4. 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*.

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. 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.
9. To test a native mobile application:

   Note: To test native mobile 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.
10. 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**.
11. Use the **Recording** window to record the test against the mobile application.

    For additional information, see *Recording Mobile Applications*.
12. When you have recorded all the actions, stop the recording.
13. Replay the test.
14. Analyze the test results.

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**.
- To persist your changes for the Android emulator, uncheck the **Wipe user data** check box in the **Launch Options** dialog box of the emulator.

Configuring the Android Emulator for Silk Test

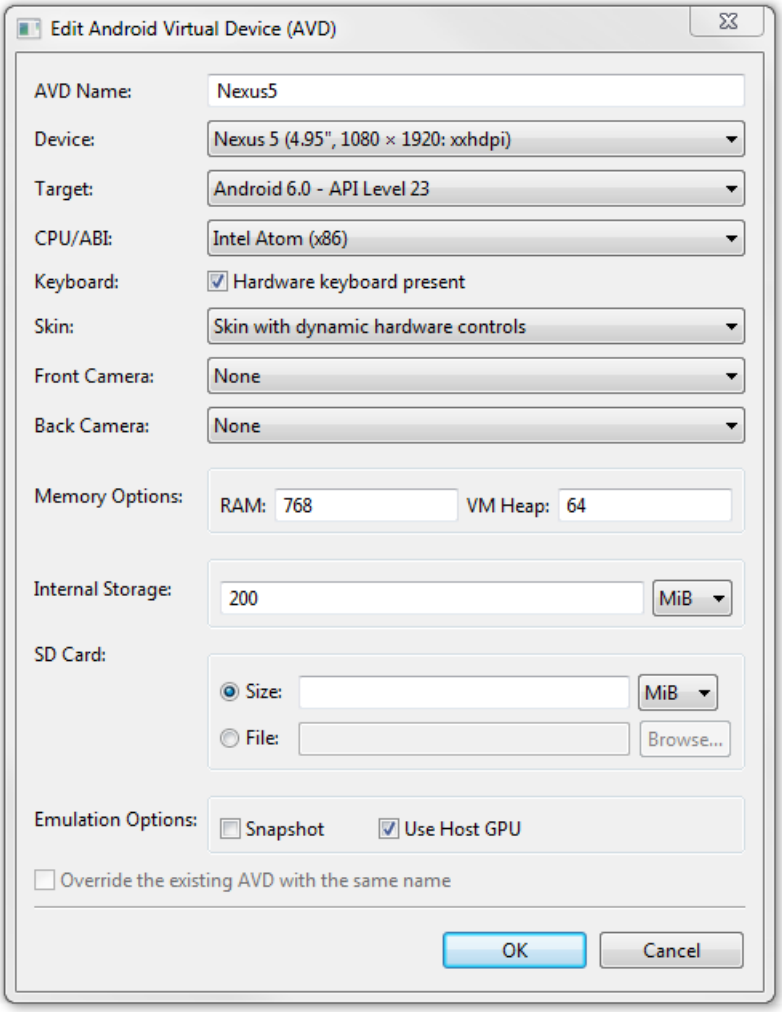
When you want to test mobile applications on an Android emulator with Silk Test, you have to configure the emulator for testing:

1. Install the Android SDK.

   For information on how to install and configure the Android SDK, see *Get the Android SDK*.
2. Start the **Android SDK Manager**.
3. For all Android versions that you want to test with the emulator, expand the version node and check the check box next to **Intel x86 Atom System Image**.
4. Click **Install** to install the selected packages.
5. Expand the **Extras** node and check the check box next to **Intel x86 Emulator Accelerator (HAXM)**.
6. Click **Install** to install the selected packages.
7. Review the *Intel Corporation license agreement*. If you accept the terms, select **Accept** and click **Install**. The **Android SDK Manager** will download the installer to the `extras` directory, under the main SDK directory. Even though the **Android SDK Manager** says `Installed` it actually means that the Intel HAXM executable was downloaded. You will still need to run the installer from the extras directory to get it installed.

8. Extract the installer inside the `extras` directory and follow the installation instructions for your platform.
9. Start the **Android Virtual Device Manager** to add a new Android Virtual Device (AVD).
10. Create a new virtual device.
11. Configure the virtual device according to your requirements.
12. Set the RAM size used by the emulator to an amount that is manageable by your machine.

    For example, set the RAM size for the emulator to `512`.
13. To enhance the speed of the transactions on the emulator, select the **Intel Atom (x86)** CPU in the **CPU/ABI** field.
14. Check the **Use Host GPU** check box in the emulation options.



15. Click **OK**.
16. *Optional:* To persist your changes for the Android emulator, uncheck the **Wipe user data** check box in the **Launch Options** dialog box of the emulator.

iOS

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

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 iOS device is connected to a Mac or the desired iOS Simulator image is installed on a Mac.
- The information service is installed on the Mac. For additional information, see *Installing the Silk Test Infoservice on a 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.
- If you want 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.
- If you want to test a mobile web application, use Xcode to build the `SafariLauncher.app`. The files that are required to build the app are located under `/Applications/Silk/Mobile/osx/Appium/node_modules/appium/node_modules/appium-ios-driver/node_modules/safari-launcher`. The output must be generated to the folder `/Applications/Silk/Mobile/osx/Appium/node_modules/appium/node_modules/appium-ios-driver/build/SafariLauncher`.
- The iOS device must not fall into sleep mode during testing. To turn the screen lock and password off, select **Settings** > **General** > **Passcode Lock**.
- 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*.

Testing Native Mobile Applications on a Physical iOS Device

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

To test a native mobile application (app) 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. 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, each user on a Mac can only test on one iOS device that is connected to the Mac or on one iOS Simulator that is running on the Mac. To test on multiple iOS devices or iOS Simulators on the same Mac, a user session on the Mac is required for each device and Simulator. For additional information, see *Testing Multiple iOS Simulators on the Same Mac* .

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. Select the **Mobile** tab.
8. Select the mobile device, on which you want to test the app, from the list.
9. 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.
10. 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**.
11. When you have recorded all actions, stop recording.
12. Replay the test.
13. 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 with Silk Test, you require a native mobile license. For additional information, see *Licensing Information*.

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

1. 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 9.2, execute the following command:

   ```
   xcodebuild -sdk iphonesimulator9.2
   ```
2. 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.
3. 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.
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. Select the **Mobile** tab.
8. Select the iOS Simulator from the list.
9. 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.

10. 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**.

11. When you have recorded all actions, stop recording.
12. Replay the test.
13. 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

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. 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, each user on a Mac can only test on one iOS device that is connected to the Mac or on one iOS Simulator that is running on the Mac. To test on multiple iOS devices or iOS Simulators on the same Mac, a user session on the Mac is required for each device and Simulator. For additional information, see *Testing Multiple iOS Simulators on the Same Mac* .

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 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**.

8. When you have recorded all actions, stop recording.
9. Replay the test.
10. Analyze the test results.

Testing Mobile Web Applications on an iOS Simulator

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

1. 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, each user on a Mac can only test on one iOS device that is connected to the Mac or on one iOS Simulator that is running on the Mac. To test on multiple iOS devices or iOS Simulators on the same Mac, a user session on the Mac is required for each device and Simulator. For additional information, see *Testing Multiple iOS Simulators on the Same Mac* .

2. Create a Silk Test project for your mobile application.
3. Create a test for your mobile application.
4. Record the actions that you want to execute in the test. When you start the **Recording** window, the **Select Application** dialog box opens.
5. 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.
6. 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**.
7. When you have recorded all actions, stop recording.
8. Replay the test.
9. Analyze the test results.

Preparing an iOS Device for Testing

Note: To test native mobile 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**.

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*.

  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 on 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

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. 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 `OSX` in the Silk Test installation directory, for example `C:\Program Files (x86)\Silk\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*.
2. Copy the file `SilkTestInformationService<Version>-<Build Number>.pkg` to the Mac.
3. Execute `SilkTestInformationService<Version>-<Build Number>.pkg` to install the information service.
4. Follow the instructions in the installation wizard.
5. When asked for the password, provide the password of the currently signed in Mac user.
6. When Apple Safari opens and a message box asks whether to trust the SafariDriver, click **Trust**.

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 a list of the available devices and browsers.

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

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**.

Testing Multiple iOS Devices or Simulators on the Same Mac

By default, Silk Test does not support testing multiple iOS devices or iOS Simulators that are connected to the same Mac. At any given time, any given user can only test on one iOS device or one iOS Simulator that is connected to the Mac. However, you can create multiple user sessions on a Mac to test multiple iOS devices that are connected to the Mac and multiple iOS Simulators that are running on the Mac in parallel.

1. For each iOS device or iOS Simulator, create a user with administrator privileges on the Mac.
   For example, to test on two iOS devices that are connected to the Mac and two iOS Simulators that are running on the Mac, you require four user sessions on the Mac with administrative privileges.
2. For each user, install the Silk Test information service (information service).
   For information on installing the information service, see *Installing the Silk Test Infoservice on a Mac*.
3. For each user, navigate to the directory `.silk/silktest/conf/` and rename the file `infoservice.properties.sample` to `infoservice.properties`.
4. For each user, edit the `infoservice.properties` file to specify a port that is not in use.
   The default port is 22901. Typically, you could increment the default port number by one for each user.
   For example, if you want to use four user sessions, use the ports 22901, 22902, 22903, and 22904.
5. For each user, logout and login again with the same user to activate the information service with the specified port.

Tip: To use the information service that corresponds to a user session, ensure that the user is logged on to the Mac, for example after a reboot of the Mac.

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

Note: Some low-level methods and classes are not supported for mobile web applications. To be able to correctly replay a test recorded against a mobile web application, uncheck the **Record native user input** option in the Browser options of Silk Test before recording against the mobile web application. For additional information, see *Limitations for Testing Mobile Web 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.

  Note: If no mobile device is connected to the machine and no emulator is started, the **Recording** window displays an error message. Connect your mobile device to the machine or start the emulator and then click **Refresh** in the **Recording** window.
- 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. 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**. You can type values for the parameters of the selected action into the parameter fields. Silk Test automatically validates the parameters.
- When you cannot directly interact with a control, for example because other controls are hiding the control, you can click **Toggle Object Hierarchy** in the **Recording** window to select the control from the control hierarchy tree.
- 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 test from the UI of Silk Test and the **Select Mobile Device** dialog box displays, the mobile device, Android Emulator, or iOS Simulator that is selected in the dialog box is used, and Silk Test ignores which mobile device is set in the test script.
- If the **Select Mobile Device** dialog box is disabled, because the **Don't show again** check box is checked, the application configurations in the individual test scripts determine the mobile device that is used to execute the tests.
- If you execute a script from the command line or from a Continuous Integration (CI) server, the application configurations of the individual scripts are used.

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 a iOS Simulator on a Mac

```
"platformName=iOS;platformVersion=9.2;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, you have to 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 actions:

Stop the Open Agent from the System Tray.

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 release a mobile device after replaying is complete, 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.

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

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 **Add Location**. The **Add Location** dialog box appears.
   b) Type a name for the remote location into the **Name** field.
   c) Type the URL of the remote location into the **URL** field.
   d) In the **Port** field, specify the port through which Silk Test connects to the information service on the remote machine.

      The default port is `22901`.
3. To edit an existing remote location, click **Edit**.
4. To remove a remote location, click **Remove**.
5. 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

The *connection string* uniquely identifies a remote browser or a mobile device. When performing remote testing or mobile testing, Silk Test uses the connection string to connect to the remote location or 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.

The following table shows the components of the connection string for each app:

	deviceName	platformName	platformVersion	host	app/bundleId/appActivity/appPackage	noReset
Native mobile on Android	Required. The device ID can be used instead.	Required.		Optional for local Android devices. Required for Android devices on remote machines.	Either the full path to the app, or a combination of *appActivity* and *appPackage*. For example `app=MyApp.apk` or `appActivity=.LoginActivity;appPackage=silktest.insurancemobile`.	Optional. Only valid if app 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.
Mobile web on Android	Required. The device ID can be used instead.	Required.		Optional for local Android devices. Required for Android devices on remote machines.		
Native mobile on iOS device	Required. The device ID can be used instead.			Required.	Either the full path to the app or the *bundleId*. For example `app=MyApp.ipa` or `bundleId=silktest.InsuranceMobile`.	Optional. Only valid if app 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.
Mobile web on iOS device	Required. The device ID can be used instead.			Required.		
Native mobile on iOS Simulator	Required.	Required.	Required.	Required.	Either the full path to the app or the *bundleId*. For example `app=MyApp.ipa` or `bundleId=silktest.InsuranceMobile`.	Optional. Only valid if app 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.

	deviceName	platformName	platformVersion	host	app/bundleId/appActivity/appPackage	noReset
Mobile web on iOS Simulator	Required.	Required.	Required.	Required.	Required.	
Remote Apple Safari or Microsoft Edge				Required.		

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. Required for Android.
3. The IP address or the host name of the remote machine, for example 10.0.0.1. If you want to test an Android device that is connected to the local machine, specify the IP address or the host name of the local machine.

Note: If you are testing web applications on an Android device that is connected to the local machine, you can use a connection string that includes only the device name. For example, the following is a valid connection string:

```
"MotoG3"
```

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"
```

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"
```

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. Required for Android.
3. The IP address or the host name of the remote machine, for example 10.0.0.1, or of the local machine, if you want to test an Android device that is connected to 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 9.2.
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 a iOS Simulator on a Mac

```
"platformName=iOS;platformVersion=9.2;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 9.2.
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 a iOS Simulator on a Mac

```
"platformName=iOS;platformVersion=9.2;deviceName=iPhone6;host=10
.0.0.1;app=MyApp.ipa;isSimulator=true"
```

Testing a remote browser

When testing a remote browser, the connection string consists of the following parts:

1. The IP address or the host name of the remote machine, for example 10.0.0.1.
2. *Optional:* The name of the browser on which you want to test, for example Safari.

Example: Connection string for Apple Safari on a remote Mac

```
"host=10.0.0.1 - Safari"
```

Troubleshooting when Testing Mobile Applications

Why does the Select Application dialog box not display my mobile browsers?

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: 1. Navigate to the Android Debug Bridge (adb) in the Android SDK installation folder. If the Android SDK is not installed, navigate to `C:\Program Files (x86)\Silk\SilkTest\ng\agent\plugins\com.microfocus.silktest.adb_17.0.0.<build number>\bin` to use the adb that is installed with Silk Test. 2. Hold **Shift** and right-click into the **File Explorer** window. 3. Select **Open command window here**. 4. In the command window, type `adb devices` to get a list of all attached devices. 5. If your device is not listed, check if USB-debugging is enabled on the device and if the appropriate USB driver is installed. 6. If you get the error `adb server is out of date`, an additional adb server might be running. For additional information, see *What can I do if the connection between the Open Agent and my device is unstable?*.
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 *Release Notes*.

Reason	Solution
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 *Installing a USB Driver*.
USB-debugging is not enabled on the device.	Enable USB-debugging on the device. For additional information, see *Enabling USB-Debugging*.

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 *SILKTEST_ANDROID_HOME*, for example to `C:\Users\<user>\AppData\Local\Android\android-sdk`. If the variable is not set, the adb version that is shipped with Silk Test is used.

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.	You can enable the developer mode in one of the following two ways: • Connect the device to a Mac on which Xcode is installed, and start the app that you want to test on the device.

Reason	Solution
	• Add your provisioning profiles to the device. **1.** Open Xcode. **2.** Select **Window** > **Devices**. **3.** Right-click on the iOS device. **4.** Select **Show Provisioning Profiles**. **5.** Add your provisioning profiles.
You have recently updated the iOS version of the device.	**1.** Open Xcode. **2.** Select **Window** > **Devices**. **3.** Wait unitl Xcode has processed the symbol files.
UI automation is not enabled on the iOS device.	**1.** Select **Settings** > **Developer**. **2.** Activate **Enable UI Automation**.
The **Web Inspector** is not activated on the iOS device, while you are trying to test a mobile web application.	**1.** Click **Settings** > **Safari** > **Advanced**. **2.** Activate the **Web Inspector**.
The app that you want to test was not built for the iOS version of the iOS device on which you are testing.	Use Xcode to build the app for the iOS version of the device.
The **Software Update** dialog box is currently open on the iOS device.	Close the dialog box and disable automatic software updates: **1.** Select **Settings** > **App and iTunes Stores** > **AUTOMATIC DOWNLOADS**. **2.** Deactivate **Updates**.

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**.

How Can I Use Chrome for Android to Replay Tests?

You can 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 a mobile device. When a mobile device is specified, Silk Test uses Chrome for Android instead of Google Chrome to execute the test.

Examples

The following Java code sample shows how you can set the base state for a test to use Chrome for Android on a Nexus 7:

```
BrowserBaseState basestate = New
BrowserBaseState(BrowserType.GoogleChrome, "demo.borland.com/
InsuranceWebExtJS/");
basestate.MobileDeviceName = "Nexus 7";
```

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 following list lists the known limitations for playing back tests and recording locators on mobile browsers:

- 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 hybrid applications (hybrid apps). However, you can use image recognition to click on objects in a hybrid app.
- 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 against Chrome for Android, Silk Test does not support zooming and scrolling.
- 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, each user on a Mac can only test on one iOS device that is connected to the Mac or on one iOS Simulator that is running on the Mac. To test on multiple iOS Simulators on the same Mac, a user session on the Mac is required for each Simulator. For additional information, see *Testing Multiple iOS Simulators on the Same Mac* .
- Before starting to test a mobile web application, ensure that no browser tab is open.
- While testing a mobile web application, you can only have one browser tab open.

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
- Silk Test does not support testing hybrid applications (hybrid apps). However, you can use image recognition to click on objects in a hybrid app.
- 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.

- At any given point in time, each user on a Mac can only test on one iOS device that is connected to the Mac or on one iOS Simulator that is running on the Mac. To test on multiple iOS devices or iOS Simulators on the same Mac, a user session on the Mac is required for each device and Simulator. For additional information, see *Testing Multiple iOS Simulators on the Same Mac* .
- 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`

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 with Silk Test 17.0

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

N

P

R

S

T

U

X