MICRO FOCUS

Silk Test 18.0

モバイル アプリケーションのテスト

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

2017-06-14

目次

モバイル アプリケーションのテスト

Silk Test では、ネイティブ モバイル アプリケーション（アプリ）およびモバイル Web アプリケーションを自動的にテストすることができます。Silk Test を使用してモバイル アプリケーションを自動的にテストすることには、次のメリットがあります。

- モバイル アプリケーションのテスト時間を大幅に減少させることができます。
- テストを一旦作成すれば、数多くの異なるデバイスやプラットフォーム上でモバイル アプリケーションをテストできます。
- エンタープライズ モバイル アプリケーションに要求される信頼性とパフォーマンスを確保できます。
- QA チームのメンバーおよびモバイル アプリケーションの開発者の効率を向上できます。
- モバイル アプリケーションは、多くのモバイル デバイスとプラットフォームで動作することを要求されるため、アジャイルにフォーカスした開発環境にとって手動テストは十分効率的とは言えない場合があります。

注: Silk Test を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

注: Silk Test は、Android および iOS デバイスの両方でのモバイル アプリのテストをサポートします。

モバイル アプリケーションのテストをサポートするオペレーティング システムとサポートするブラウザーについての情報は、『リリース ノート』を参照してください。

Android

Silk Test では、Android デバイスまたは Android エミュレータ上のモバイル アプリケーションをテストすることができます。

Android 上のモバイル アプリケーションのテストにおける前提条件

Android デバイスや Android エミュレータ上のモバイル アプリケーションをテストする前に、次の前提条件を満たしていることを確認してください。

- Silk Test を実行するマシンに Java インストールされており、Java へのパスが *Path* 環境変数に追加されていることを確認してください。 Silk Test は、JRE または JDK のいずれかをモバイル テストで必要とします。Java がインストールされていない場合は、C:¥Program Files (x86)¥Silk¥SilkTest¥ng¥jre¥bin を *Path* に追加してください。
- ネイティブ モバイル アプリに Web ビューを追加したハイブリッド アプリを作成した場合は、アプリを Silk Test でテスト可能にするために、次のコードをアプリに追加してください。

  ```
  WebView.setWebContentsDebuggingEnabled(true);
  webView.getSettings().setJavaScriptEnabled(true);
  ```
- Silk Test では、Android 4.4 のデバイス画面のライブ ビュー表示をサポートしていません。 このため、Android 5 以降を使用することを Micro Focus は推奨しています。

Android 上のモバイル アプリケーションのテスト

物理 Android デバイスや Android エミュレータ上のモバイル アプリケーションをテストするには、次のタスクを実行します。

1. Android 上のモバイル アプリケーションのテストにおける前提条件を満たすことを確認します。
   詳細については、「*Android 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。
2. Android エミュレータ上のモバイル アプリケーションをテストするには、Silk Test 用にエミュレータを設定します。
   詳細については、「*Silk Test 用に Android エミュレータを設定する*」を参照してください。
3. Silk Test をインストールしたマシンで Android エミュレータを開始するか、デバイスを接続します。
4. 物理 Android デバイスを初めて使用するマシンでテストする場合、適切な Android USB ドライバをマシンにインストールします。
   詳細については、「*USB ドライバのインストール*」を参照してください。
5. 物理 Android デバイス上でモバイル アプリケーションをテストする場合は、Android デバイスの USB デバッグを有効化します。
   詳細については、「*USB デバッグの有効化*」を参照してください。
6. モバイル アプリケーション用の Silk Test プロジェクトを作成します。
7. モバイル アプリケーション用のテストを作成します。
8. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
9. モバイル Web アプリケーションをテストするには：
   a) **Web** タブを選択します。
   b) 使用するモバイル ブラウザーを選択します。
   c) **移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。
10. ネイティブ モバイル アプリケーションまたはハイブリッド アプリケーションをテストするには：

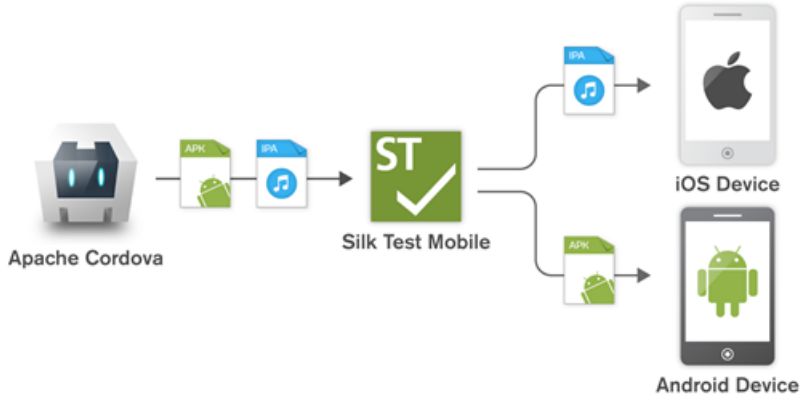
    注: Silk Test を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

    a) **モバイル** タブを選択します。
    b) アプリをテストするモバイル デバイスをリストから選択します。
    c) **参照** をクリックしてアプリ ファイルを選択するか、アプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。
       このパスでは、Silk Test は HTTP および UNC 形式をサポートします。
       Silk Test は、モバイル デバイスまたはエミュレータ上に指定したアプリをインストールします。
11. **OK** をクリックします。
    Android デバイスまたはエミュレータの画面が、テスト中にロックされないようにしてください。マシンに接続中にデバイスがロックされないようにするには、**開発者向けオプション** を開きます。**スリープモードにしない** または **充電中に画面をスリープにしない** をチェックします。
12. **記録** ウィンドウを使用して、モバイル アプリケーションに対するテストを記録します。
    詳細については、「モバイル アプリケーションの記録」を参照してください。
13. すべての操作の記録を終えたら、記録を停止します。
14. テストを再生します。
15. テスト結果を分析します。

Android 上のハイブリッド アプリケーションのテスト

ハイブリッド アプリケーション（アプリ）は、デバイス上で実行されるネイティブ アプリケーションのようなアプリですが、HTML5、CSS、JavaScript などの Web テクノロジを使用して記述されたアプリです。

Silk Test は、ネイティブ コンテナに埋め込まれた単一の Web ビューで構成されたデバッグ ハイブリッド アプリのテストにする完全なブラウザー サポートを提供します。このようなハイブリッド アプリの一般的な例は、Apache Cordova アプリケーションです。



非デバッグ ハイブリッド アプリをテスト可能にするには、次のコードをアプリに追加してアプリをリモート デバッグできるようにします。

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

リモート デバッグを有効化しない非デバッグ ハイブリッド アプリや、複数の Web ビューを含んだハイブリッド アプリをテストするには、Silk Test フォールバック サポートを有効化するために、オプション OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT を TRUE に設定します。詳細については、「詳細オプションの設定」を参照してください。フォールバック サポートを有効化すると、Silk Test は Web ビューのコントロールをブラウザー コントロールではなく、ネイティブ モバイル コントロールとして解決して処理します。たとえば、以下のコードは、ブラウザー サポートを使用したときのリンクのクリックです。

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

フォールバック サポートを有効化すると、同じリンクをクリックするコードは次のようになります。

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

Silk Test は、Chrome リモート デバッグをサポートする Web ビューを検出できます。 Silk Test は、*com.android.webview* パッケージまたは *com.google.android.webview* パッケージ（多くの Android デバイス上のデフォルト パッケージ）のいずれかを使用した Web ビューを検出できます。

注: Silk Test は、Android 4.4 以降でのハイブリッド アプリのテストをサポートします。Android でハイブリッド アプリをテストするには、Android システムの WebView バージョン 51 以降が必要です。

Android 上のハイブリッド アプリをテストする手順は、モバイル ネイティブ アプリケーションをテストする手順と同じです。詳細については、「*Android* 上のモバイル アプリケーションのテスト」を参照してください。

USB ドライバのインストール

モバイル アプリケーションをテストするために、ローカル マシンに最初に Android デバイスに接続するには、適切な USB ドライバをインストールする必要があります。

デバイスの製造元は、そのデバイスに必要なすべてのドライバをもった実行可能ファイルを提供している可能性があります。この場合、ローカル マシンにその実行可能ファイルをインストールするだけです。製造元がこのような実行可能ファイルを提供していない場合、マシン上にデバイスに対する単一の USB ドライバをインストールできます。

Android USB ドライバをインストールするには：

1. デバイス用の適切なドライバをダウンロードします。
   たとえば、Google Nexus デバイス用の USB ドライバを検索し、インストールする場合は、*http://developer.android.com/tools/extras/oem-usb.html* を参照します。
2. Android デバイスをローカル マシンの USB ポートに接続します。
3. デスクトップ、または **Windows Explorer** から、**コンピュータ** を右クリックし、**管理** を選択します。
4. 左側のペインで、**デバイス マネージャ** を選択します。
5. 右側のペインで、**その他のデバイス** を探して展開します。
6. デバイス名 (*Nexus 5x* など) を右クリックして、**ドライバ ソフトウェアの更新** を選択します。**ハードウェアの更新ウィザード** が開きます。
7. **コンピュータを参照してドライバ ソフトウェアを検索します** を選択して、**次へ** をクリックします。
8. **参照** をクリックし、USB ドライバをダウンロードしたフォルダに移動します。
9. USB ドライバを選択します。
10. **次へ** をクリックしてドライバをインストールします。

USB デバッグの有効化

Android Debug Bridge (adb) 上で Android デバイスと通信するために USB デバッグを有効化します。

1. Android デバイスで設定を開きます。
2. **開発者向けオプション** (Dev Settings) をタップします。
   開発者向けオプションは、デフォルトでは表示されません。開発者向けオプションがデバイスの設定メニューに含まれていない場合：
   a) 画面を下にスクロールさせて、デバイスが携帯電話の場合は **端末情報** を、タブレットの場合は **タブレット情報** をタップします。
   b) 再度画面を下にスクロールさせて **ビルド番号** を 7 回タップします。
3. **開発者向けオプション** ウィンドウで、**USB デバッグ** をオンにします。
4. デバイスの USB モードをデフォルトの設定である **メディア デバイス (MTP)** に設定します。
   詳細については、デバイスのドキュメントを参照してください。

Android デバイスの推奨設定

Silk Test を使用したテストを最適化するために、テストしたい Android デバイスで次の設定を行ってください。

- USB デバッグを Android デバイスで有効化します。詳細については、「*USB デバッグの有効化*」を参照してください。
- Android デバイスは、Open Agent を実行しているマシンに、メディア デバイスとして接続されている必要があります。Android デバイスの USB モードは、**メディア デバイス (MTP)** を設定します。
- Android デバイスまたはエミュレータの画面が、テスト中にロックされないようにしてください。マシンに接続中にデバイスがロックされないようにするには、**開発者向けオプション** を開きます。**スリープモードにしない** または **充電中に画面をスリープにしない** をチェックします。

Silk Test 用に Android エミュレータを設定する

Silk Test を使用して Android エミュレータ上でモバイル アプリケーションをテストする場合、テスト用にエミュレータを設定する必要があります。

1. Android SDK の最新版をインストールします。
   Android SDK のインストールと設定についての詳細は、「*Get the Android SDK*」を参照してください。

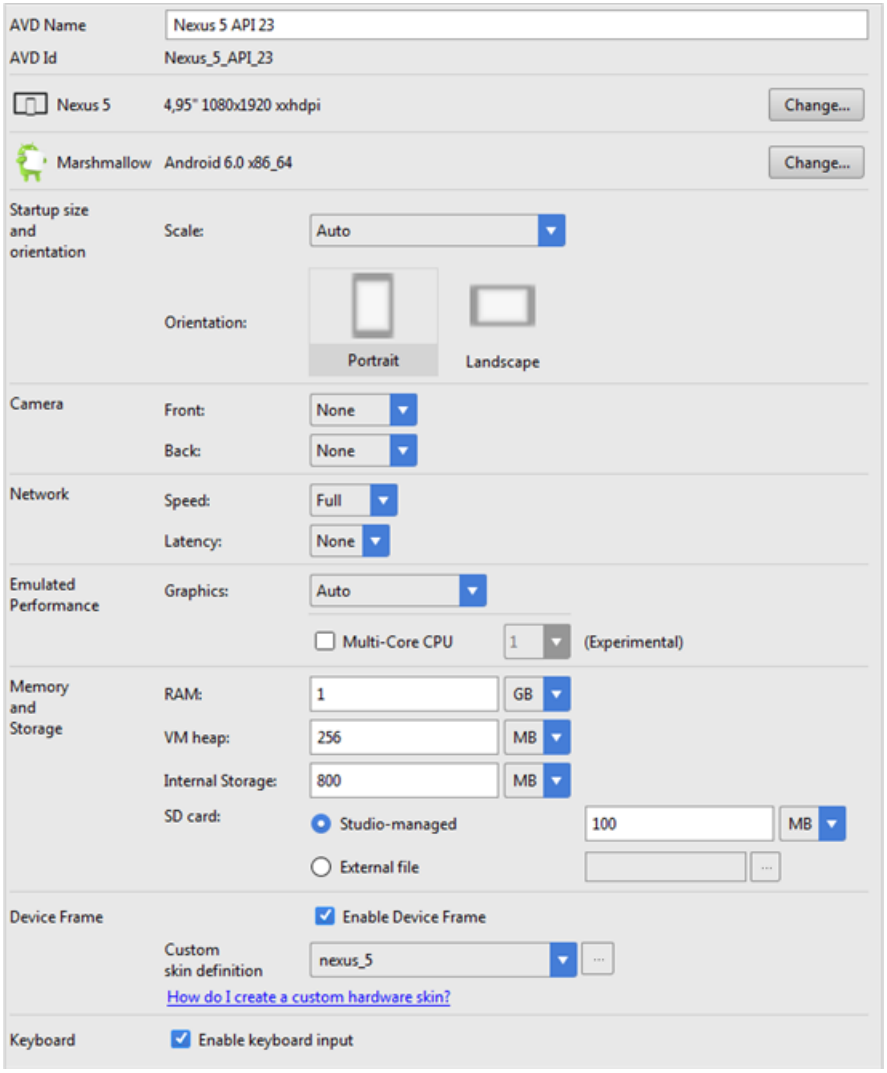
2. Android Studio 2 をインストールします。

   ヒント: Android Studio 2 のインストールをスキップして、Android SDK で提供されるエミュレータを使用することもできます。ただし、エミュレータのパフォーマンスを向上させるため、Android Studio 2 をインストールすることを、Micro Focus では推奨しています。このトピックの以下の手順では、Android Studio 2 がインストールされていることを前提としています。

3. Android Studio 2 から **AVD Manager** を開きます。
4. **Create Virtual Device** をクリックします。
5. 仮想デバイスを選択します。
6. **Next** をクリックします。
7. Google API を含む Android のシステム イメージをダウンロードして選択します。
8. **Next** をクリックします。
9. 要件に従って仮想デバイスを設定します。
10. **Show Advanced Settings** をクリックします。
11. エミュレータが使用する RAM サイズとヒープ領域を対象のマシンで管理可能な量に調整します。

    ヒント: 最低 1 GB の RAM と 256 MB のヒープ領域を使用することを、Micro Focus では推奨しています。

12. **Emulated Performance** 領域のリストで **Auto** を選択します。

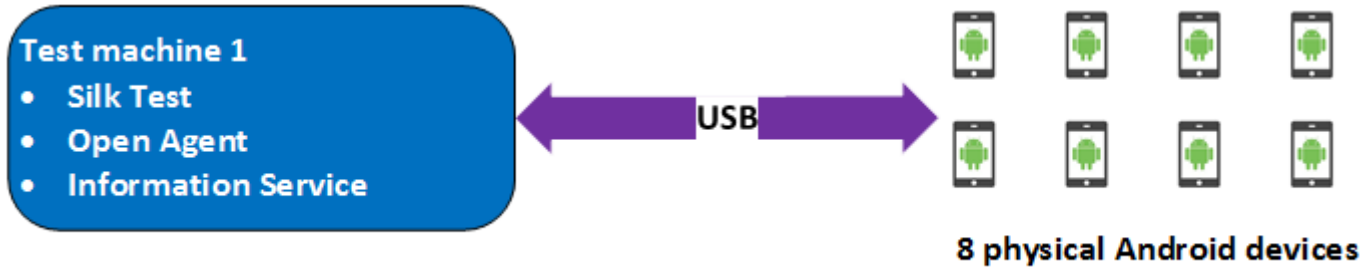


13. **Finish** をクリックします。

並列テスト実行のテスト済みの構成

Silk Test を使用して、複数の Android デバイス上で並列に自動テストを実行することができます。並列に実行できる Android デバイスの数は利用可能なハードウェアに依存します。 Micro Focus では、次のハードウェア構成でテストの実行を確認しています。

単一テスト マシンを使用した構成

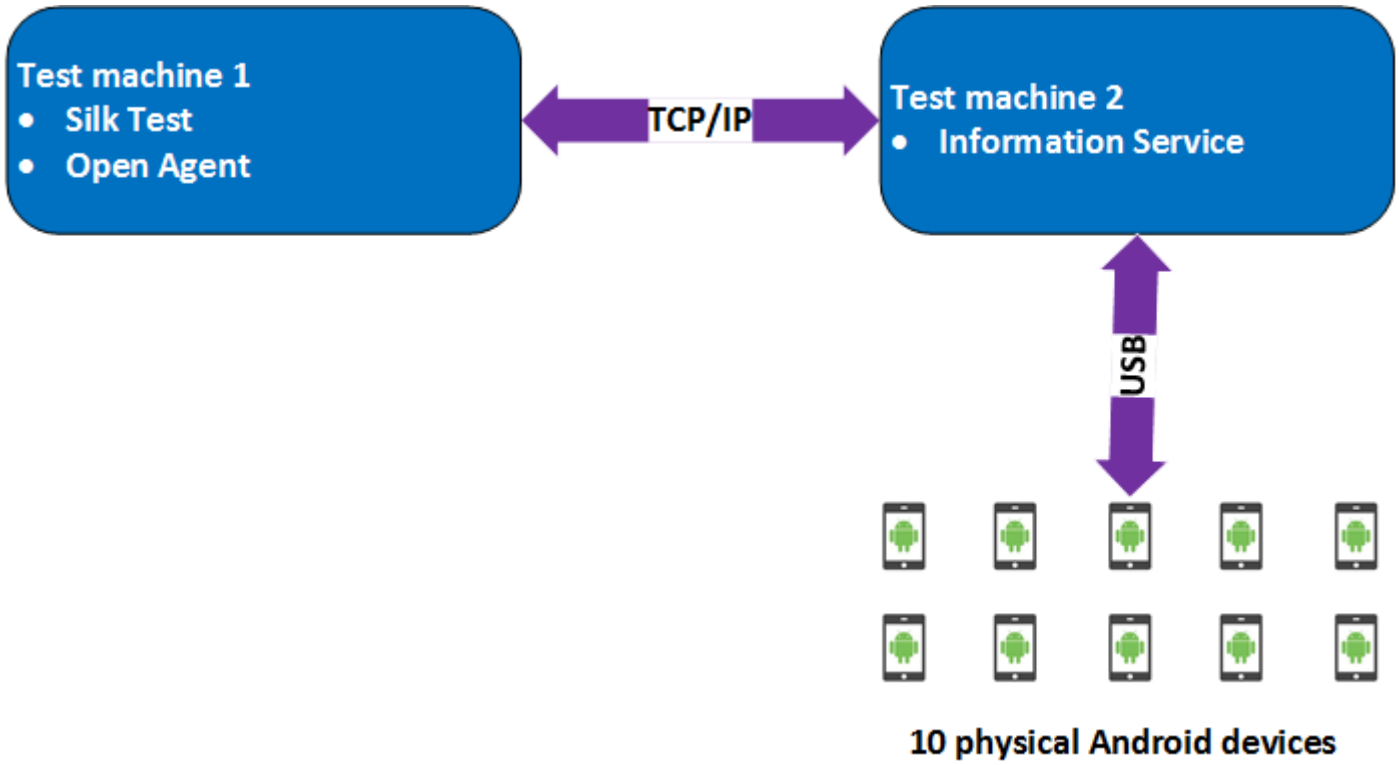


USB により Android デバイスを直接接続した単一のテスト マシンを使用して、最大 8 台の物理 Android デバイスまでの並列テストを行いました。

テスト マシンは、次のハードウェア仕様の Lenovo ThinkPad T450 です。

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 コア (4 スレッド)
- 8 GB RAM

2 台のテスト マシンを使用した構成



また、2 台のテスト マシンを使用してテストを行いました。 1 台に Silk Test をインストールし、もう 1 台には Silk Test Information Service をインストールして最初のマシンのリモート ロケーションとして設

定しました。このような構成を使用して、最大 10 台の物理 Android デバイスまでの並列テストを行いました。

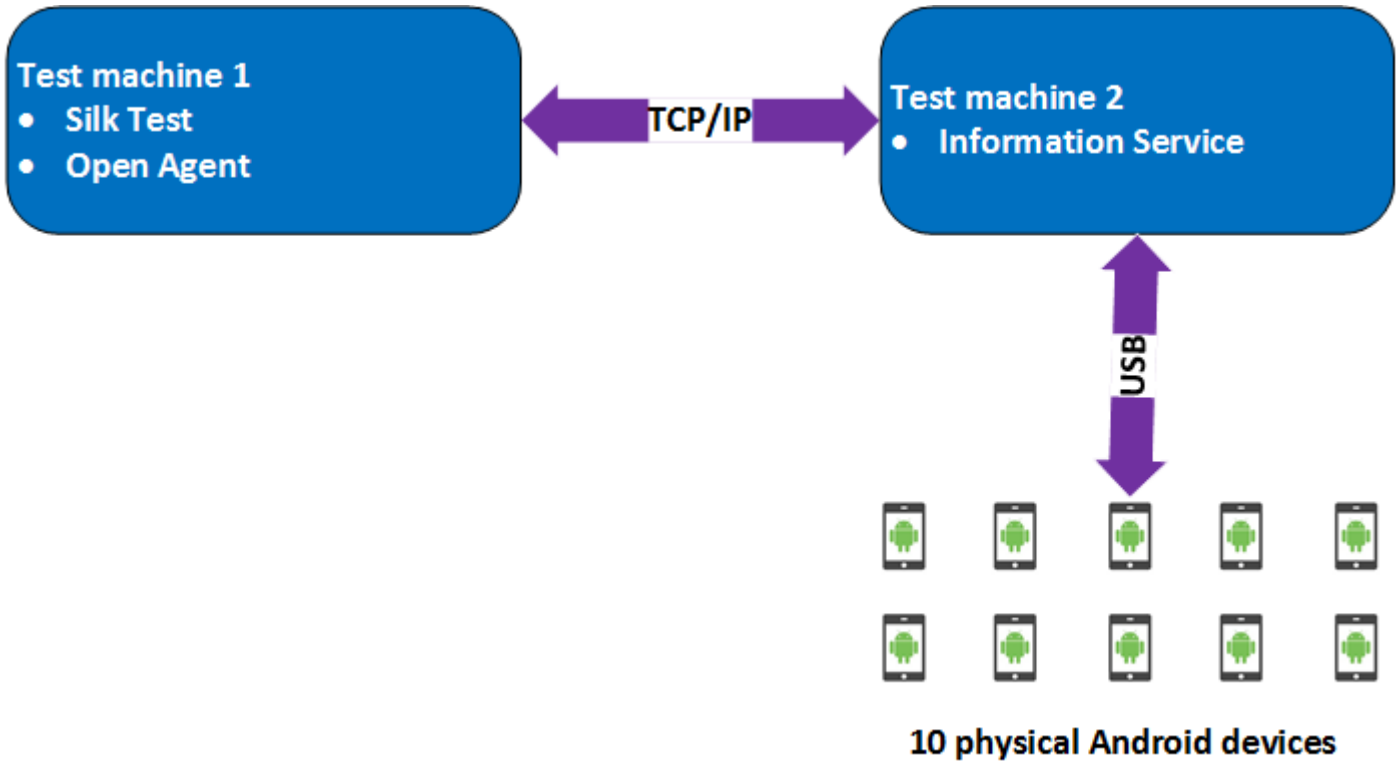
テスト マシン 1 は、次のハードウェア仕様の Lenovo ThinkPad T450 です。

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 コア (4 スレッド)
- 8 GB RAM

テスト マシン 2 は、次のハードウェア仕様の Dell Precision T1700 です。

- Intel® Core™ i7 - 4770 CPU @ 3.40 GHz
- 4 コア (8 スレッド)
- 16 GB RAM

Windows マシンと Mac を使用した構成



さらに、2 台のテスト マシンを使用したテストで、Windows マシンに Silk Test をインストールし、Mac には Silk Test Information Service をインストールして最初のマシンのリモート ロケーションとして設定しました。このような構成を使用して、最大 10 台の物理 Android デバイスまでの並列テストを行いました。

テスト マシン 1 は、次のハードウェア仕様の Lenovo ThinkPad T450 です。

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 コア (4 スレッド)
- 8 GB RAM

テスト マシン 2 は、次のハードウェア仕様の Apple Mac Mini です。

- Intel® Core™ i5 - 4782U CPU @ 2.60 GHz
- 2 コア (4 スレッド)
- 16 GB RAM

iOS

Silk Test では、iOS デバイスまたは iOS シミュレータ上のモバイル アプリケーションをテストすることができます。

iOS の以前のバージョンと比べて、iOS 9.3 では大幅な変更が Apple によってなされたため、Silk Test では iOS 9.3 以降のモバイル アプリケーションのテストをサポートします。サポートされている iOS のバージョンのリストについては、『リリース ノート』を参照してください。

ヒント: iOS の以前のバージョンをテストする場合は、Silk Test 17.5 を使用してください。Silk Test 18.0 を使用した iOS 上でのテストについて、Silk Test 17.5 でのテストとの主な違いを以下の表に示します。

Silk Test 18.0	Silk Test 17.5
iOS 9.3、iOS 10 をサポートします。	iOS 8.1、8.2、8.3、8.4、9.0、9.1、9.2、9.3 をサポートします。
Xcode 8.3 をサポートします。	Xcode 6、Xcode 7 をサポートします。
同じユーザー セッションでの複数の物理 iOS デバイスのテストをサポートします。 **ヒント:** Silk Test 17.5 を使用してテストするために、複数のユーザー セッションを作成している場合、1 つ以外のすべてのユーザー セッションを削除することを Micro Focus は推奨します。	複数の物理 iOS デバイスをテストする場合は、Mac 上に複数のユーザー セッションが必要です。

iOS 上のモバイル アプリケーションのテストにおける前提条件

iOS デバイスや iOS シミュレータ上のモバイル アプリケーションをテストする前に、次の前提条件を満たしていることを確認してください。

- 現在のバージョンの Silk Test Information Service が Mac 上にインストールされている。詳細については、「*Silk Test Information Service を Mac にインストールする*」を参照してください。
- 物理 iOS デバイス上でアプリケーションをテストする場合は、以下の項目を確認してください。
  - デバイスが Mac に接続されている。
  - デバイスが iOS のサポートされているバージョンである。サポートされている iOS のバージョンのリストについては、『リリース ノート』を参照してください。
- iOS シミュレータ上でアプリケーションをテストする場合は、以下の項目を確認してください。
  - iOS シミュレータのイメージが Mac 上にインストールされている。
  - iOS シミュレータのイメージが iOS のサポートされているバージョンである。サポートされている iOS のバージョンのリストについては、『リリース ノート』を参照してください。
- 物理 iOS デバイス上でアプリケーションをテストする場合は、デバイスと Mac の両方ともタイム ゾーンが同じである。
- サポートするバージョンの Xcode が Mac 上にインストールされている。
- Windows マシン上に Silk Test がインストールされている。
- Mac が Windows マシンと同じネットワークにあり、Windows マシンにリモート ロケーションとして追加されている。
- iOS デバイス上でネイティブ モバイル アプリをテストする場合は、Developer Account でサインされたアプリの .ipa ファイルが作成されている。詳細については、「*iOS アプリのテストの準備*」を参照してください。

- iOS シミュレータ上でネイティブ モバイル アプリをテストする場合は、ZIP 形式に圧縮したアプリが作成されている。詳細については、「*iOS シミュレータ上のネイティブ モバイル アプリケーションのテスト*」を参照してください。
- iOS デバイスと iOS シミュレータの両方でネイティブ モバイル アプリをテストする場合は、サインした .ipa ファイルと ZIP した .app ディレクトリの両方が同じフォルダに作成されている。
- ネイティブ モバイル アプリをテストする場合は、iOS デバイスの ID がアプリをサインするのに使用した Developer Profile に関連付けられている。
- iOS デバイスのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。
- テスト中に Mac の画面がオフにならないようにしてください。オフになると、**再生ステータス** ダイアログ ボックスに何も表示されなくなります。
- iOS シミュレータ上でモバイル アプリケーションをテストする場合は、テスト中に Mac のディスプレイ スリープが無効化されている。
- 物理 iOS デバイス上のネイティブ モバイル アプリをテストする場合は、デバイスで UI オートメーションが有効化されている。詳細については、「*iOS デバイスのテストの準備*」を参照してください。
- 物理 iOS デバイス上で Apple Safari を使用してのモバイル Web アプリケーションをテストする場合は、**Web インスペクタ** が有効化されている。詳細については、「*iOS デバイスのテストの準備*」を参照してください。
- Lightning コネクタを持つ iOS デバイスを使用することを Micro Focus は推奨しています。これは、Silk Test では、Lightning ケーブルで Mac に接続していない iOS デバイスのデバイス画面のライブ ビュー表示をサポートしていないためです。

物理 iOS デバイス上のネイティブ モバイル アプリケーションのテスト

注: Silk Test を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

iOS 上のモバイル アプリケーションのテストにおける前提条件については、「*iOS 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。ネイティブ モバイル アプリケーションをテストする際の既知の制限については、「モバイル ネイティブ アプリケーションのテストにおける制限事項」を参照してください。

物理 iOS デバイス上のネイティブ モバイル アプリケーション (アプリ) やハイブリッド アプリケーションをテストするには、次のタスクを実行します。

1. テストする iOS デバイスを準備します。

   詳細については、「*iOS デバイスのテストの準備*」を参照してください。
2. テストするアプリを準備します。

   詳細については、「*iOS アプリのテストの準備*」を参照してください。
3. テストする Mac を準備します。詳細については、「*iOS 上でのモバイル アプリケーションのテストのための Mac の準備*」を参照してください。
4. iOS デバイスが接続されている Mac を、Silk Test がインストールされている Windows マシンに、リモート ロケーションとして追加します。

   詳細については、「リモート ロケーションの編集」を参照してください。

   注: 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。Silk Test 17.5 Hotfix 1 以降を使用した場合、iOS 上のモバイル アプリケーションをテストするために、Mac 上で複数のユーザー セッションを使用する必要はありません。
5. モバイル アプリケーション用の Silk Test プロジェクトを作成します。
6. モバイル アプリケーション用のテストを作成します。

7. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
8. **モバイル** タブを選択します。
9. アプリをテストするモバイル デバイスをリストから選択します。
10. **参照** をクリックしてアプリ ファイルを選択するか、アプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。

    このパスでは、Silk Test は HTTP および UNC 形式をサポートします。

    Silk Test は、モバイル デバイス上に指定したアプリをインストールします。
11. **OK** をクリックします。

    iOS デバイスやシミュレータのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。
12. すべての操作の記録を終えたら、記録を停止します。
13. テストを再生します。
14. テスト結果を分析します。

注: iOS デバイスと iOS シミュレータの両方でネイティブ モバイル アプリをテストする場合は、サインした .ipa ファイルと ZIP した .app ディレクトリの両方が同じフォルダに作成されている。

iOS シミュレータ上のネイティブ モバイル アプリケーションのテスト

注: Silk Test を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

iOS 上のモバイル アプリケーションのテストにおける前提条件については、「*iOS 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。ネイティブ モバイル アプリケーションをテストする際の既知の制限については、「モバイル ネイティブ アプリケーションのテストにおける制限事項」を参照してください。

iOS シミュレータ上のネイティブ モバイル アプリケーション (アプリ) やハイブリッド アプリケーションをテストするには、次のタスクを実行します。

1. テストする Mac を準備します。詳細については、「*iOS 上でのモバイル アプリケーションのテストのための Mac の準備*」を参照してください。
2. アプリの Xcode プロジェクトで、iOS シミュレータ用にアプリをコンパイルします。

    Xcode UI からでも、コマンドラインからでもアプリをコンパイルできます。たとえば、iOS 10.0 で iOS シミュレータ用のアプリをコマンドラインでコンパイルするには、次のコマンドを実行します。

    ```
    xcodebuild  -sdk  iphonesimulator10.0
    ```
3. アプリの .app ディレクトリを .zip ファイルに Zip します。

    デフォルトでは、.app ディレクトリは、~/Library/Developer/Xcode/DerivedData ディレクトリにあります。Xcode で **File** > **Project Settings** をクリックすれば、ディレクトリがある場所を確認できます。
4. iOS シミュレータがインストールされている Mac を、Silk Test がインストールされている Windows マシンに、リモート ロケーションとして追加します。

    詳細については、「リモート ロケーションの編集」を参照してください。

    注: Mac にインストールされている 1 つの iOS シミュレータでのみテストを実行できます。複数の Silk Test ユーザーが、同じ Mac にインストールされている複数の iOS シミュレータ上で同時にテストを実行することはできません。
5. モバイル アプリケーション用の Silk Test プロジェクトを作成します。

6. モバイル アプリケーション用のテストを作成します。
7. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
8. **モバイル** タブを選択します。
9. リストから iOS シミュレータを選択します。
10. **参照** をクリックして Zip したアプリ ファイルを選択するか、Zip したアプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。

    このパスでは、Silk Test は HTTP および UNC 形式をサポートします。

    Silk Test は、iOS シミュレータ上に指定したアプリをインストールします。
11. **OK** をクリックします。

    iOS デバイスやシミュレータのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。
12. すべての操作の記録を終えたら、記録を停止します。
13. テストを再生します。
14. テスト結果を分析します。

注: iOS デバイスと iOS シミュレータの両方でネイティブ モバイル アプリをテストする場合は、サインした .ipa ファイルと ZIP した .app ディレクトリの両方が同じフォルダに作成されている。

物理 iOS デバイス上のモバイル Web アプリケーションのテスト

iOS 上のモバイル アプリケーションのテストにおける前提条件については、「*iOS 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。モバイル Web アプリケーションをテストする際の既知の制限については、「*モバイル Web アプリケーションのテストにおける制限事項*」を参照してください。

物理 iOS デバイス上のモバイル Web アプリケーションをテストするには、次のタスクを実行します。

1. テストする iOS デバイスを準備します。

    詳細については、「*iOS デバイスのテストの準備*」を参照してください。
2. テストする Mac を準備します。詳細については、「*iOS 上でのモバイル アプリケーションのテストのための Mac の準備*」を参照してください。
3. iOS デバイスが接続されている Mac を、Silk Test がインストールされている Windows マシンに、リモート ロケーションとして追加します。

    詳細については、「リモート ロケーションの編集」を参照してください。

    注: 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。Silk Test 17.5 Hotfix 1 以降を使用した場合、iOS 上のモバイル アプリケーションをテストするために、Mac 上で複数のユーザー セッションを使用する必要はありません。
4. モバイル アプリケーション用の Silk Test プロジェクトを作成します。
5. モバイル アプリケーション用のテストを作成します。
6. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
7. モバイル Web アプリケーションをテストするには：

    a) **Web** タブを選択します。

    b) 使用するモバイル ブラウザーを選択します。

    c) **移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。
8. **OK** をクリックします。

iOS デバイスやシミュレータのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。

9. すべての操作の記録を終えたら、記録を停止します。
10. テストを再生します。
11. テスト結果を分析します。

iOS シミュレータ上のモバイル Web アプリケーションのテスト

モバイル Web アプリケーションをテストする際の既知の制限については、「*モバイル Web アプリケーションのテストにおける制限事項*」を参照してください。

iOS シミュレータ上のモバイル Web アプリケーションをテストするには、次のタスクを実行します。

1. テストする Mac を準備します。詳細については、「*iOS 上でのモバイル アプリケーションのテストのための Mac の準備*」を参照してください。
2. iOS シミュレータがインストールされている Mac を、Silk Test がインストールされている Windows マシンに、リモート ロケーションとして追加します。

   詳細については、「リモート ロケーションの編集」を参照してください。

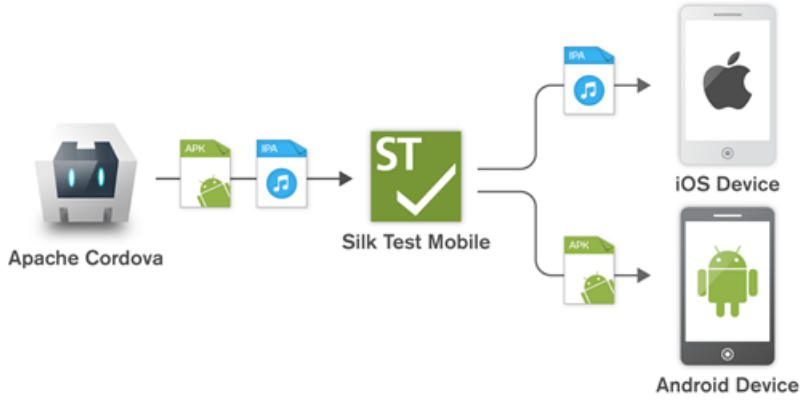
   注: 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。Silk Test 17.5 Hotfix 1 以降を使用した場合、iOS 上のモバイル アプリケーションをテストするために、Mac 上で複数のユーザー セッションを使用する必要はありません。
3. モバイル アプリケーション用の Silk Test プロジェクトを作成します。
4. モバイル アプリケーション用のテストを作成します。
5. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
6. モバイル Web アプリケーションをテストするには：
   a) **Web** タブを選択します。
   b) 使用するモバイル ブラウザーを選択します。
   c) **移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。
7. **OK** をクリックします。

   iOS デバイスやシミュレータのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。
8. すべての操作の記録を終えたら、記録を停止します。
9. テストを再生します。
10. テスト結果を分析します。

iOS 上のハイブリッド アプリケーションのテスト

ハイブリッド アプリケーション（アプリ）は、デバイス上で実行されるネイティブ アプリケーションのようなアプリですが、HTML5、CSS、JavaScript などの Web テクノロジを使用して記述されたアプリです。

Silk Test は、ネイティブ コンテナに埋め込まれた単一の Web ビューで構成されたデバッグ ハイブリッド アプリのテストにする完全なブラウザー サポートを提供します。このようなハイブリッド アプリの一般的な例は、Apache Cordova アプリケーションです。



リモート デバッグを有効化しない非デバッグ ハイブリッド アプリや、複数の Web ビューを含んだハイブリッド アプリをテストするには、Silk Test フォールバック サポートを有効化するために、オプション OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT を TRUE に設定します。詳細については、「詳細オプションの設定」を参照してください。フォールバック サポートを有効化すると、Silk Test は Web ビューのコントロールをブラウザー コントロールではなく、ネイティブ モバイル コントロールとして解決して処理します。たとえば、以下のコードは、ブラウザー サポートを使用したときのリンクのクリックです。

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

フォールバック サポートを有効化すると、同じリンクをクリックするコードは次のようになります。

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

iOS 上のハイブリッド アプリをテストする手順は、モバイル ネイティブ アプリケーションをテストする手順と同じです。詳細については、「物理 *iOS* デバイス上のネイティブ モバイル アプリケーションのテスト」または「*iOS* シミュレータ上のネイティブ モバイル アプリケーションのテスト」を参照してください。

iOS デバイス上のハイブリッド アプリをテストする前に、デバイスで **Web インスペクタ** が有効化されていることを確認してください。

iOS デバイスのテストの準備

注: Silk Test を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

モバイル アプリケーションをテストするために iOS デバイスを準備するには：

1. Mac 上で Xcode を起動します。
2. iOS デバイスを Mac に接続します。
3. iOS デバイスで、**設定** > **デベロッパ** をクリックします。

   ヒント: デベロッパ メニューが iOS デバイスに表示されていない場合は、デバイスと Mac を再起動します。
4. **Enable UI Automation** をオンにします。
5. Apple Safari 上でモバイル Web アプリケーションをテストするには、**設定** > **Safari** > **詳細** をクリックします。
6. **Web インスペクタ** をオンにします。

iOS アプリのテストの準備

Silk Test を使用して特定の デバイス上で特定の iOS アプリをテストできるようにするには、次の項目を考慮する必要があります。

- 特定の デバイスに手動でインストールできる iOS アプリに対してのみテストを自動化できます。iOS アプリにサインできるようにするために、*Apple Developer Program* のメンバーシップに登録する必要があります。詳細については、「メンバーシップの選択」を参照してください。*Apple Developer Program* のメンバーシップに登録せずにテストを行う場合は、「パーソナル チーム プロファイルを使用した物理 *iOS* デバイス上でのテスト 」を参照してください。

  注: App Store で配布するように作成された アプリや、任意の iOS デバイスに手動でインストールできるアプリを自動的にテストできません。
- 特定の デバイスで アプリをインストールして実行する前に、iOS デバイスを Apple Developer アカウントを使用して登録する必要があります。
- デバイスに iOS アプリをインストールするには、Xcode を使用してアプリの IPA ファイルを作成する必要があります。特定の iOS デバイスでのテスト用に IPA ファイルを作成するには、Apple Developer Program のメンバーとして Xcode の *Archive* 機能を使用します。この機能には 2 つのオプションがあります。

  - *Apple Developer Enterprise Program* のメンバーの場合、**Save for Ad Hoc Deployment** オプションを使用できます。
  - Apple Developer Enterprise Program のメンバー以外の Apple Developer Program のメンバーの場合、**Save for Development Deployment** オプションを使用できます。

  詳細については、「*Exporting Your App for Testing (iOS, tvOS, watchOS)*」を参照してください。

シミュレータ上の特定の アプリを Silk Test を使用してテストできるようにするには、Xcode を使用してアプリの Zip ファイルを作成してから iOS シミュレータ上にインストールします。詳細については、Xcode のドキュメントを参照してください。

Silk Test Information Service を Mac にインストールする

注: Information Service を Mac にインストールするには、Mac の管理者権限が必要です。

Mac 上の Apple Safari や、Mac に接続されている iOS や Android デバイス上のモバイル アプリケーションに対するテストを作成して実行するには、Mac に Silk Test Information Service (Information Service) をインストールしてから、**リモート ロケーション** ダイアログ ボックスを使用して、Silk Test をインストールした Windows マシンと Mac を接続する必要があります。

Information Service を Mac にインストールするには :

1. Java JDK が Mac 上にインストールされていることを確認します。
2. iOS デバイス上でモバイル アプリケーションをテストする場合は、Xcode が Mac 上にインストールされていることを確認します。
3. Information Service セットアップ ファイル（SilkTestInformationService<バージョン>-<ビルド番号>.pkg にアクセスします。

   - Silk Test のインストール時に Information Service セットアップ ファイルをダウンロードした場合は、Silk Test インストール ディレクトリ（C:¥Program Files (x86)¥Silk¥SilkTest など）の macOS フォルダを開きます。
   - Silk Test のインストール時に Information Service セットアップ ファイルをダウンロードしなかった場合は、*Micro Focus SupportLine* からセットアップ ファイルをダウンロードできます。
4. SilkTestInformationService<バージョン>-<ビルド番号>.pkg ファイルを Mac にコピーします。

5. SilkTestInformationService<バージョン>-<ビルド番号>.pkg を実行して、Information Service をインストールします。
6. インストール ウィザードの指示に従います。
7. パスワードを尋ねられた場合、現在サインインしている Mac ユーザーのパスワードを入力します。
8. Apple Safari が開き、SafariDriver を信頼するかどうかを尋ねるメッセージ ボックスが表示されたら、**信頼** をクリックします。

   注: リモート接続ではなく、直接 Mac にログインしている場合には、SafariDriver だけをインストールできます。

インストールを完了するために、現在の Mac ユーザーをログアウトします。Information Service が正しくインストールされていることを確認するには、Mac にログインし、画面の右上隅にある Silk Test アイコンをクリックして、利用可能なデバイスとブラウザーを表示させます。

ヒント: Silk Test アイコンが表示されない場合は、Mac を再起動してください。

iOS 上でのモバイル アプリケーションのテストのための Mac の準備

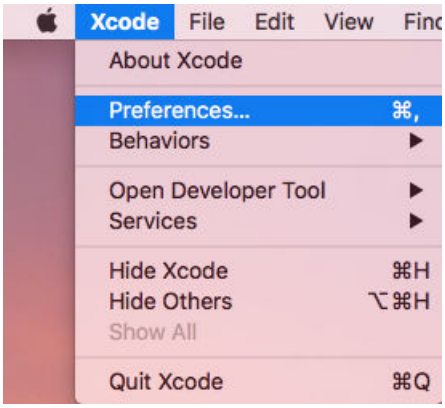
注: Silk Test を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

iOS 上でモバイル アプリケーションをテストするためには、iOS デバイスを接続した、または iOS シミュレータを実行している Mac が必要です。この Mac には Xcode がインストールされている必要があります。iOS 上のモバイル アプリケーションのテストにおける前提条件についての詳細は、「*iOS 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。

iOS テストを物理 iOS デバイス上で実行するには、**Silk Test Configuration Assistant** の指示に従って WebDriverAgentRunner Xcode プロジェクトを設定します。**Configuration Assistant** を開くには、ステータス メニューの Silk Test アイコンをクリックして、**Configuration Assistant** を選択します。

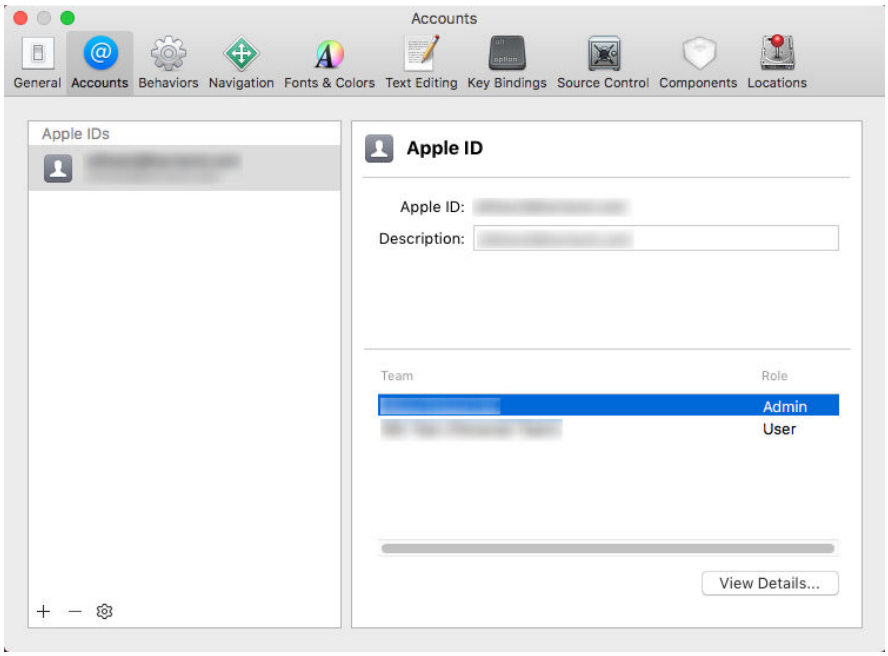
WebDriverAgentRunner Xcode プロジェクトを手動でビルドしたい場合には、次の手順を実行します。

1. Mac 上で Xcode を起動します。
2. **Xcode** > **Preferences** を選択します。



3. **Preferences** ウィンドウで、アカウントを選択します。
   a) **Accounts** タブを選択します。
   b) **Apple ID** を選択します。
   c) **Team** を選択します。

d) **View Details** をクリックします。



4. **Apple Member Center** にアクセスし、最新のプロビジョニング プロファイル ID と開発チーム ID を取得します。
5. .mobileprovision ファイルの名前から、プロビジョニング プロファイル ID をコピーします。

   注: プロビジョニング プロファイル ID はファイル拡張子を含まずに 36 文字の長さのファイル名です。
6. ターミナルで、~/.silk/silktest/conf/ に移動します。
7. xcconfig ファイル テンプレート silktest.xcconfig.sample を silktest.xcconfig という名前に変更します。
8. 開発チーム ID を silktest.xcconfig ファイルに追加します。

   ```
   DEVELOPMENT_TEAM = <your development team>
   ```
9. Mac 上のターミナルで次のコマンドを実行すると、WebDriverAgentRunner プロジェクトが正しく準備できたかどうかを確認できます。

   a) 物理 iOS デバイスの UDID（Unique Device ID）を見つけます。

   ```
   idevice_id -l
   ```

   b) WebDriverAgentRunner プロジェクトに移動します。

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

   c) WebDriverAgent がビルドできるかどうかをテストします。

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

   <udid> を先ほど見つけた UDID で置き換えます。

   ヒント: xcodebuild コマンドが失敗する場合には、エラー メッセージの手順に従ってください。さらに、WebDriverAgentRunner プロジェクトで Preferences ウィンドウを開き、**General** タブの **Automatically manage signing** チェック ボックスがオフになっていることを確認します。
10. 省略可能：infoservice.properties ファイルには、Silk Test Information Service のポートや、Mac 上でのすべてのテストの実行で使用されるケイパビリティを指定することができます。
    詳細については、「*Silk Test Information Service プロパティの編集*」を参照してください。

パーソナル チーム プロファイルを使用した物理 iOS デバイス上でのテスト

Apple Developer Program のメンバーシップに登録していない場合は、パーソナル チーム プロファイルを使用して、物理 iOS デバイス上のアプリケーションをテストできます。

1. Mac 上で、/Application/Silk/Mobile/common/Appium/node_modules/appium-xcuitestdriver/WebDriverAgent に移動します。
2. WebDriverAgent.xcodeproj プロジェクトを Xcode で開きます。
3. **TARGETS** リストから、WebDriverAgentLib ターゲットを選択します。
   a) **General** タブをクリックします。
   b) **Automatically manage signing** を選択します。
   c) 開発チームを選択します。

   Signing Certificate は自動的に選択されます。
4. **TARGETS** リストから、WebDriverAgentRunner ターゲットを選択します。
   a) **General** タブをクリックします。
   b) **Automatically manage signing** を選択します。
   c) 開発チームを選択します。

   Signing Certificate は自動的に選択されます。
5. WebDriverAgentRunner ターゲットに対するプロビジョニング プロファイルの作成に Xcode が失敗する場合は、ターゲットのバンドル ID を手動で変更します。
   a) **Build Settings** タブをクリックします。
   b) **Product Bundle Identifier** の値を Xcode が受け付ける適切な値に変更します。

      たとえば、**Product Bundle Identifier** が *com.facebook.WebDriverAgentRunner* の場合、*io.appium.WebDriverAgentRunner* や *io.borland.WebDriverAgentRunner* などに変更します。
   c) **General** タブをクリックします。

   これでターゲットにプロビジョニング プロファイルが設定されました。
6. WebDriverAgent.xcodeproj プロジェクトを保存します。
7. すべてが期待通り動作することを確認するために、プロジェクトをビルドします。

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

WebDriverAgentRunner のデバイスへのインストールに成功しても、次のようなエラー メッセージがコンソールや Appium ログ ファイルに出力される場合があります。
2017-01-24 09:02:18.358 xcodebuild[30385:339674] Error Domain=com.apple.platform.iphoneos Code=-12 "Unable to launch com.apple.test.WebDriverAgentRunner-Runner" UserInfo={NSLocalizedDescription=Unable to launch com.apple.test.WebDriverAgentRunner-Runner, NSUnderlyingError=0x7fa839cadc60 {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={NSLocalizedDescription=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)

この問題は、開発者がそのデバイスで信頼されていないため発生します。そのデバイス上で WebDriverAgentRunner アプリを手動で実行しようとすると、**信頼されていない開発元** メッセージが表示される場合があります。

デバイス上でこの問題を解決するには、**設定** > **一般** > **プロファイル** または **設定** > **一般** > **デバイス管理**（デバイスの種類と iOS のバージョンによって異なります）に移動します。そして、開発元を信頼し、WebDriverAgentRunner アプリの実行を許可します。

Silk Test Information Service プロパティの編集

infoservice.properties ファイルを使用して、Silk Test Information Service のポートやさまざまなケイパビリティ（Capabilities）を指定できます。これらのケイパビリティは、Silk Test Information Service を実行しているマシン上で Silk Test がテストを実行するたびに毎回適用されます。

1. infoservice.properties.sample ファイルがあるディレクトリに移動します。
   たとえば、Mac の場合は ~/.silk/silktest/conf/ に移動します。Windows マシンの場合は C:¥ProgramData¥Silk¥SilkTest¥conf に移動します。
2. infoservice.properties.sample ファイルの名前を infoservice.properties に変更します。
3. 使用していないポートを指定します。
   Silk Test Information Service のデフォルトのポートは 22901 です。
4. ケイパビリティを指定するには、次の行を infoservice.properties ファイルに追加します。

```
customCapabilities=<custom  capability_1>;<custom_capability_2>;...
```

例：指定した言語での iOS シミュレータの実行

Mac 上で iOS シミュレータを常に同じ言語で実行するようにするには、カスタム ケイパビリティ *language* および *locale* を指定します。たとえば日本語の場合、次の行を infoservice.properties ファイルに追加します。

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

Silk Test Information Service を Mac からアンインストールする

Mac 上の Apple Safari に対するテストを実行する必要がなくなった場合など、次の手順で Silk Test Information Service を Mac からアンインストールすることができます。

1. uninstallInfoService.sh のような新しいシェル ファイルを作成します。
2. 新しいファイルに以下のコードを入力します。

```
#!/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. コマンドラインで chmod +x uninstallInfoService.sh を実行し、シェル ファイルの実行可能にします。
4. コマンドラインからシェル ファイルを実行します。

iOS デバイスの推奨設定

Silk Test を使用したテストを最適化するために、テストしたい iOS デバイスで次の設定を行ってください。

- 実際にユーザーが行った操作をテストに反映させるために、Apple Safari の自動入力とパスワードの保存を無効化します。**設定** > **Safari** > **パスワードと自動入力** をタップし、**ユーザ名とパスワード** 設定をオフにします。
- iOS デバイスのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。

XCUITest を使用した iOS 上の既存のスクリプトの実行

注目: 以前のバージョンの Silk Test では、*Instruments* を使用して iOS デバイスを自動化していました。iOS 9.3 から、Instruments のサポートは *XCUITest* フレームワークのサポートに Apple によって置き換えられたため、Silk Test でも *Instruments* をサポートできなくなりました。この変更により、既存の iOS テスト スクリプトを Silk Test の現在のバージョンで実行できなくなる場合があります。

- XCUITest における classname 属性の動作は Instruments での動作と異なります。ほとんどの場合、Silk Test はこの変更に自動的に対応します。しかし、このような classname 属性が原因で既存のテスト スクリプトが動作しなくなった場合には、対応するオブジェクトの新しいロケーターを記録する必要があります。
- オブジェクトの階層が変更されました。

インストール済みアプリのテスト

デバイス、エミュレーター、またはシミュレータ上に既にインストールされているネイティブ モバイル アプリをテストするには、接続文字列でアプリを指定します。

1. ネイティブ モバイル アプリをテストする既存のプロジェクトを開きます。
2. **アプリケーション構成の編集** ダイアログ ボックスを開きます。
3. 次のいずれかの方法で、既存のアプリの指定部分を接続文字列で置き換えます。
   - iOS アプリをテストする場合は、*bundleId* を使ってアプリを指定します。たとえば、app=MyApp.ipa を bundleId=silktest.InsuranceMobile で置き換えます。
   - Android アプリをテストする場合は、*appActivity* と *appPackage* を使用してアプリを指定します。たとえば、app=MyApp.apk を appActivity=.LoginActivity;appPackage=silktest.insurancemobile で置き換えます。

詳細については、「接続文字列」を参照してください。

モバイル アプリケーションの記録

Silk Test とモバイル デバイスまたはエミュレータとの間の接続が一旦確立すると、デバイス上で実行する操作を記録できます。モバイル アプリケーションを記録するために、Silk Test では次の機能を持つ **記録** ウィンドウを使用します。

- テストするモバイル デバイスまたは Android エミュレータの画面を表示します。
- **記録** ウィンドウで操作を実行すると、モバイル デバイス上でも同じ操作が実行されます。
- 画面上のコントロールを操作すると、**記録** ウィンドウはデフォルトの操作を事前に選択します。
    - デフォルトの操作が Click の場合、コントロール上で左クリックすると、その操作が実行されます。右クリックすると、コントロールに対して利用可能な操作のリストが表示されます。この場合は、実行する操作を選択し、**OK** をクリックします。
    - デフォルトの操作が Click ではない場合、コントロールに対して有効なすべての操作がリストで表示されるので、実行したい操作を選択するか、単に **OK** をクリックして事前に選択された操作を受け入れます。

  リストから操作を選択する場合、選択した操作のパラメータの値をパラメータ フィールドに入力することができます。Silk Test は自動的にパラメータを検証します。
- 記録中、Silk Test は記録ウィンドウの隣にマウスの位置を表示します。その表示をクリックすると、デバイス画面に絶対的な位置とアクティブ オブジェクトに相対的な位置を切り替えることができます。
- 記録を一時停止すると、画面上での操作は記録されないため、デバイスを記録を続けたい状態に変更することができます。
- 記録を停止すると、記録した操作でスクリプトが生成されるため、続いてテストの再生を行うことができます。

テストを再生するモバイル デバイスの選択

テストを再生するために使用するモバイル デバイスを定義できます。

- スクリプトをコマンド ラインや CI サーバーから実行する場合は、スクリプトのアプリケーション構成で接続文字列を指定します。
- Silk Central からテストを実行する場合、接続文字列を指定する代わりに、Silk Central の実行定義の **配置** タブにある **モバイル デバイスの選択** 領域でモバイル デバイスを指定します。詳細については、『*Silk Central ヘルプ*』を参照してください。

デバイス プールなどがある場合に、特定のモバイル デバイスを指定したり、利用可能なデバイス群のサブセットをフィルタするために接続文字列を使用できます。最初に一致したデバイスが再生に使用されます。特に指定がない場合には、次のルールに従って一致したモバイル デバイスが使用されます（高い優先度順）。

- リモート ロケーションに接続されたモバイル デバイスよりも、ローカル マシンに接続されたモバイル デバイスが優先されます。
- ブラウザーの種類が接続文字列で指定されている場合、古いバージョンのブラウザーよりも、新しいバージョンのブラウザーが優先されます。
- 古いプラットフォームよりも、新しいプラットフォームが優先されます。
- 物理デバイスがエミュレータやシミュレータよりも優先されます。
- アルファベット順で後者のデバイス名のデバイスが優先されます。たとえば、"iphone 5"という名前のデバイスよりも、"iphone 6"という名前のデバイスが優先されます。

例：リモート マシンに接続されている Android デバイス上のアプリに対する接続文字列

リモート マシンに接続されている Android デバイス上で MyApp.apk アプリをテストするには、接続文字列は次のようになります。

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

例：Mac の iOS シミュレータ上のアプリに対する接続文字列

```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=10.0.0.
1;app=MyApp.ipa;isSimulator=true"
```

モバイル デバイスの操作

モバイル デバイスを操作したり、テスト対象アプリケーションでスワイプのような操作を実行するには、次の手順を実行します。

1. **記録** ウィンドウで、**モバイル デバイス操作の表示** をクリックします。モバイル デバイスに対して実行できるすべての操作がリストされます。
2. リストからリストから実行したい操作を選択します。
3. Android デバイスまたはエミュレータで、スワイプを記録するには、マウスの左ボタンを押したままマウスを動かします。
4. テストの記録を続行します。

モバイル デバイスの開放

モバイル デバイスに対するテストを記録または再生する場合、Open Agent インスタンスはそのデバイスの所有権を確保します。これによって、Open Agent は、他の Silk Test ユーザーがそのデバイスを使用することを防止します。デバイス上でのテストの記録や再生が終了した後に、他の Silk Test ユーザーがデバイスを使用できるようにするために、Silk Test クライアントが閉じたとき、無人のテスト プロセスが完了したとき、または Open Agent が閉じたときに、Silk Test は自動的にデバイスを開放します。また、デバイスを手動で開放することもできます。

注: モバイル デバイスを開放すると、モバイル デバイス上のテスト対象アプリケーション (AUT) は閉じられます。

記録後のモバイル デバイスの開放

他の Silk Test ユーザーがモバイル デバイスでテストできるようにするために、記録後にデバイスを開放します。

記録が完了した後にモバイル デバイスを開放するには、次のいずれかを実行します。

- システム トレイから Open Agent を停止する。
- Silk Test を閉じる。並列テストが有効な場合には、この操作でのみデバイスが開放されます。

注: モバイル デバイスを開放すると、モバイル デバイス上のテスト対象アプリケーション (AUT) は閉じられます。

再生後のモバイル デバイスの開放

他の Silk Test ユーザーがモバイル デバイスでテストできるようにするために、再生後にデバイスを開放します。

再生が完了した後に手動でモバイル デバイスを開放するために、次のいずれかを実行することもできます。

- モバイル Web アプリケーションのテストを再生した場合、BrowserApplication クラスの Close メソッド、または CloseSynchron メソッド を使用します。これらのメソッド の詳細については、 API ドキュメントを参照してください。

  ```
  webBrowser.close();
  ```
- モバイル ネイティブ アプリケーションのテストを再生した場合、MobileDevice クラスの CloseApp メソッドを使用します。

  たとえば、次のように入力します。

  ```
  MobileDevice mobileDevice = desktop.find("//MobileDevice");
  mobileDevice.closeApp();
  ```
- *desktop*.detachAll() ステートメントをテスト スクリプトに追加します。

次の条件を満たしている場合は、モバイル デバイスは自動的に開放されます。

- Open Agent を閉じる。
- 無人テスト中にテスト プロセスが停止する。並列テストが有効な場合には、この操作でのみデバイスが開放されます。
- Silk Test を閉じる。並列テストが有効な場合には、この操作でのみデバイスが開放されます。

注: モバイル デバイスを開放すると、モバイル デバイス上のテスト対象アプリケーション (AUT) は閉じられます。

リモート ロケーションの編集

リモート ロケーション ダイアログ ボックスを使用すると、リモート ロケーション上のブラウザやモバイル デバイスを、テストするアプリケーションのセットに追加できます。

1. **リモート ロケーション** ダイアログ ボックスを開きます。
   - Silk Test Workbench を使用している場合、**ツール** > **リモート ロケーションの編集** をクリックします。
   - Silk4J を使用している場合、**Silk4J** > **リモート ロケーションの編集** をクリックします。
   - Silk4NET を使用している場合、**Silk4NET** > **リモート ロケーションの編集** をクリックします。
   - Silk Test Classic を使用している場合、**オプション** > **リモート ロケーションの編集** をクリックします。
2. リモート ロケーションを追加するには、次の操作を実行します。
   a) **ロケーションの追加** の右側にある矢印をクリックして、Silk Test Information Service を使用してリモート ロケーションを追加するのか、Silk Central を追加するのかを指定します。

      注: Silk Central は 1 つだけリモート ロケーションとして設定できます。Silk Central との統合が既に設定されている場合は、リモート ロケーション リストに Silk Central が表示されます。

   b) **ロケーションの追加** をクリックします。**ロケーションの追加** ダイアログ ボックスが表示されます。
   c) Silk Test がリモート マシン上の Information Service に接続するためのリモート ロケーションの URL とポートを、**ホスト** フィールドに入力します。

      デフォルトのポートは 22901 です。
   d) 省略可能：**名前** フィールドにリモート ロケーションの名前を入力します。

3. 既存のリモート ロケーションを編集するには、**編集** をクリックします。
4. リモート ロケーションを削除するには、**削除** をクリックします。
5. 省略可能：**アプリケーションの選択** ダイアログに表示されるブラウザーやデバイスの数を減らすには、**このロケーション上のデバイスとブラウザーを非表示** をクリックします。そのリモート ロケーションにインストールされたブラウザーや接続されたデバイスが、**アプリケーションの選択** ダイアログに表示されなくなります。デフォルトでは、すべてのリモート ロケーションにインストールされたブラウザーや接続されたデバイスのすべてが、**アプリケーションの選択** ダイアログに表示されます。
6. **OK** をクリックします。

リモート ロケーションを追加すると、リモート ロケーション上にインストールされているブラウザ (Mac 上の Apple Safari など) が、**アプリケーションの選択** ダイアログ ボックスの **Web** タブで利用可能になり、リモート ロケーションに接続されているモバイル デバイスが、**アプリケーションの選択** ダイアログ ボックスの **モバイル** タブで利用可能になります。

モバイル デバイスの接続文字列

接続文字列 は、テストに使用するモバイル デバイスを指定します。モバイル テストを実行する場合、Silk Test は接続文字列を使用してモバイル デバイスに接続します。接続文字列は、アプリケーション構成の主要な一部です。テスト対象アプリケーションを構成するときに、接続文字列は設定されます。接続文字列を変更するには、**アプリケーション構成の編集** ダイアログ ボックスを使用します。

注: Silk Central からテストを実行する場合、接続文字列を指定する代わりに、Silk Central の実行定義の **配置** タブにある **モバイル デバイスの選択** 領域でモバイル デバイスを指定します。詳細については、『*Silk Central ヘルプ*』を参照してください。

デバイス プールなどがある場合に、特定のモバイル デバイスを指定したり、利用可能なデバイス群のサブセットをフィルタするために接続文字列を使用できます。最初に一致したデバイスが再生に使用されます。特に指定がない場合には、次のルールに従って一致したモバイル デバイスが使用されます（高い優先度順）。

- リモート ロケーションに接続されたモバイル デバイスよりも、ローカル マシンに接続されたモバイル デバイスが優先されます。
- ブラウザーの種類が接続文字列で指定されている場合、古いバージョンのブラウザーよりも、新しいバージョンのブラウザーが優先されます。
- 古いプラットフォームよりも、新しいプラットフォームが優先されます。
- 物理デバイスがエミュレータやシミュレータよりも優先されます。
- アルファベット順で後者のデバイス名のデバイスが優先されます。たとえば、"iphone 5"という名前のデバイスよりも、"iphone 6"という名前のデバイスが優先されます。

次のコンポーネントが接続文字列で使用できます。

コンポーネント	説明
deviceName	モバイル デバイスの名前。物理モバイル デバイス上でテストをする場合、デバイス ID を代わりに使用します。ワイルドカードをサポートします。大文字と小文字は区別されません。
platformName	Android または iOS。必須。
deviceId	省略可能：モバイル デバイスの ID。物理モバイル デバイスでテストをする場合に、デバイス名の代わりに使用します。ワイルドカードをサポートします。大文字と小文字は区別されません。
platformVersion	省略可能：Android または iOS のバージョン。特定の Android または iOS のバージョンのモバイル デバイス上でのみテストする場合に、バージョンを指定します。ワイルドカードをサポートします。大文字と小文字は区別されません。
browserVersion	省略可能：特定のブラウザーのバージョンでのみテストする場合に、ブラウザーの種類と共に使用します。ワイルドカードをサポートします。大文字と小文字は区別されません。

コンポーネント	説明
host	省略可能：設定しない場合は、任意のリモート ロケーションがホストとして使用されます。ワイルドカードをサポートします。大文字と小文字は区別されません。
• app • appActivity • appPackage	Android 上のネイティブ モバイル アプリケーションのテストでは必須です。アプリへのフルパス、または *appActivity* と *appPackage* の組み合わせで指定します。たとえば、app=MyApp.apk、appActivity=.LoginActivity;appPackage=silktest.insurancemobile などです。
• app • bundleId	iOS 上のネイティブ モバイル アプリケーションのテストでは必須です。アプリへのフルパス、または *bundleId* で指定します。たとえば、app=MyApp.ipa、bundleId=silktest.InsuranceMobile などです。
noReset	省略可能：ネイティブ モバイル アプリケーションをテストする場合に設定できます。*app* が指定されている場合にのみ有効です。テストの前にアプリを再インストールしない場合は True。テストの前にアプリを再インストールする場合は False を指定します。デフォルト値は、False です。
isSimulator	省略可能：iOS シミュレータ上でのみテストを実行する場合に指定します。デバイス名を代わりに使用できます。
isPhysicalDevice	省略可能：物理デバイス上でのみテストを実行する場合に指定します。デバイス名を代わりに使用できます。

デバイス プールを使用して、テストで実際に使用されるデバイスを確認するには、MobileDevice クラスの generateConnectionString メソッドの戻り値を使用できます。

モバイル デバイスまたは Android エミュレータ上のモバイル Web アプリケーションのテスト

モバイル デバイスまたは Android エミュレータ上でモバイル Web アプリケーションをテストする場合、接続文字列は次の要素から構成されます。

1. モバイル デバイス名（MotoG3 など）、またはデバイス ID（11111111 など）。

   注: デバイス ID は可読性に欠けるため、デバイス名がユニークであれば、デバイス名を接続文字列に使用することを Micro Focus では推奨します。
2. プラットフォーム名。
3. ブラウザーのバージョン。ブラウザーの種類の設定との組み合わせでのみ使用されます。
4. 特定のリモート マシンの IP アドレスまたはホスト名（10.0.0.1 など）。**リモート ロケーションの編集** ダイアログ ボックスで指定したリモート ロケーションの名前をホスト名として使用することもできます（*MyRemoteLocation* など）。リモート ロケーション名を使用する場合に、ワイルドカードを使用することもできます。ローカル マシンに接続されている Android デバイスをテストする場合は、ローカル マシンの IP アドレスまたはホスト名を指定します。

例：利用可能な任意の Android デバイスの接続文字列

```
"platformName=Android"
```

例：ローカル マシンに接続されている Android デバイス上のブラウザーに対する接続文字列

ローカル マシンに接続されている Android デバイス上でモバイル ブラウザーをテストするには、接続文字列は次のようになります。

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

または

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

例：リモート マシンに接続されている Android デバイス上のブラウザーに対する接続文字列

リモート Android デバイス上のモバイル ブラウザーをテストするには、接続文字列は次のようになります。

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

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

例：Mac に接続されている iOS デバイス上のブラウザーに対する接続文字列

リモート iOS デバイス上のモバイル ブラウザーをテストするには、接続文字列は次のようになります。

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

モバイル デバイスまたは Android エミュレータ上のネイティブ モバイル アプリケーションのテスト

モバイル デバイスまたは Android エミュレータ上でネイティブ モバイル アプリケーションをテストする場合、接続文字列は次の要素から構成されます。

1. モバイル デバイス名（MotoG3 など）、またはデバイス ID（11111111 など）。

   注: デバイス ID は可読性に欠けるため、デバイス名がユニークであれば、デバイス名を接続文字列に使用することを Micro Focus では推奨します。
2. プラットフォーム名。
3. 特定のリモート マシンの IP アドレスまたはホスト名（10.0.0.1 など）。**リモート ロケーションの編集** ダイアログ ボックスで指定したリモート ロケーションの名前をホスト名として使用することもできます（*MyRemoteLocation* など）。リモート ロケーション名を使用する場合に、ワイルドカードを使用することもできます。ローカル マシンに接続されている Android デバイスをテストする場合は、ローカル マシンの IP アドレスまたはホスト名を指定します。
4. テストするアプリのファイルの名前または、ファイルが Web サーバー上にある場合には、ファイルの URL。たとえば、C:/MyApp.apk や MyApp.ipa など。

   - Android アプリは、常に .apk ファイルを指定します。
   - 物理デバイス上の iOS アプリは、常に .ipa ファイルを指定します。
   - シミュレータ上の iOS アプリは、ZIP ファイルまたは、*app* という名前のディレクトリを指定します。

例：リモート マシンに接続されている Android デバイス上のアプリに対する接続文字列

リモート マシンに接続されている Android デバイス上で MyApp.apk アプリをテストするには、接続文字列は次のようになります。

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

例：Mac に接続されている iOS デバイス上のアプリに対する接続文字列

リモート マシンに接続されている iOS デバイス上で MyApp.ipa アプリをテストするには、接続文字列は次のようになります。

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

iOS シミュレータ上のモバイル Web アプリケーションのテスト

iOS シミュレータ上でモバイル Web アプリケーションをテストする場合、接続文字列は次の要素から構成されます。

1. プラットフォーム名（iOS）。
2. プラットフォームのバージョン（10.0 など）。
3. モバイル デバイス名（iPhone6 など）。
4. iOS シミュレータを実行している Mac の IP アドレスまたはホスト名。

例：Mac の iOS シミュレータ上のブラウザーに対する接続文字列

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

iOS シミュレータ上のネイティブ モバイル アプリケーションのテスト

Mac の iOS シミュレータ上でネイティブ モバイル アプリケーションをテストする場合、接続文字列は次の要素から構成されます。

1. プラットフォーム名（iOS）。
2. プラットフォームのバージョン（10.0 など）。
3. モバイル デバイス名（iPhone6 など）。
4. リモート マシンの IP アドレスまたはホスト名（10.0.0.1 など）。
5. テストするアプリの名前（MyApp.ipa など）。

例：Mac の iOS シミュレータ上のアプリに対する接続文字列

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

モバイル アプリケーションのテスト時のトラブルシューティング

[アプリケーションの選択] ダイアログにモバイル デバイスが表示されない理由

Silk Test がモバイル デバイスやエミュレータを認識できないと、**アプリケーションの選択** ダイアログの **モバイル** タブにはデバイスやエミュレータが表示されません。さらに、**アプリケーションの選択** ダイアログの **Web** タブにも、そのデバイスやエミュレータ上にインストールされているモバイル ブラウザーが表示されません。

Silk Test が、次の何れかが原因でモバイル デバイスまたはエミュレータを認識していない可能性があります。

原因	解決策
エミュレータが実行されていない。	エミュレータを開始します。
Android Debug Bridge (adb) がモバイル デバイスを認識しない。	モバイル デバイスが adb によって認識されているかどうか確認するには： 1. Android SDK をインストールしたフォルダで、Android Debug Bridge (adb) がある場所に移動します。Android SDK がインストールされていない場合、C:¥Program Files (x86)¥Silk¥SilkTest¥ng

原因	解決策
	¥Mobile¥windows¥AndroidTools¥platform-tools に移動して、Silk Test がインストールした adb を使用します。 2. **Shift** を押しながら、**ファイル エクスプローラ** ウィンドウで右クリックします。 3. **コマンド ウィンドウをここで開く** を選択します。 4. コマンド ウィンドウで、 adb devices を入力して、アタッチしたすべてのデバイスのリストを得ます。 5. デバイスがリストされない場合、USB デバッグがデバイスで有効化されていること、および適切な USB ドライバがインストールされていることを確認します。 6. 「adb server is out of date」のようなエラーが発生する場合は、C:¥Program Files (x86)¥Silk¥SilkTest¥ng¥Mobile¥windows¥AndroidTools¥platform-tools の adb のバージョンがローカルの Android SDK の adb のバージョンと一致していることを確認してください。詳細については、「*Open Agent* とデバイスとの接続が不安定な場合の対処方法」を参照してください。
デバイスのオペレーティング システムのバージョンを Silk Test がサポートしていない。	サポートするモバイル オペレーティング システムのバージョンについては、リリース ノート を参照してください。
デバイスの USB ドライバがローカル マシンにインストールされていない。	デバイスの USB ドライバをローカル マシンにインストールしてください。詳細については、「*USB* ドライバをインストールする」を参照してください。
USB デバッグがデバイスで有効化されていない。	USB デバッグをデバイスで有効化してください。詳細については、「*USB* デバッグの有効化」を参照してください。

注: どの解決策を適用しても解決できない場合は、デバイスを再起動してみてください。

URL に移動せずに Silk Test が Chrome for Android で URL を検索する理由

アドレス バーに入力された URL を、Chrome for Android が検索として解釈する場合があります。回避策として、URL に移動するコマンドをスクリプトに手動で追加できます。

adb サーバーが正しく起動しない場合にすべきこと

Android Debug Bridge (adb) サーバーが開始するとき、ローカル TCP ポート 5037 にバインドし、adb クライアントから送信されてくるコマンドをリッスンします。すべての adb クライアントは、ポート 5037 を使用して、adb サーバーと通信します。adb サーバーは、5555 から 5585 の範囲 (エミュレータやデバイスで使用される範囲) で奇数のポートをスキャンしてエミュレータやデバイス インスタンスを探します。adb はこれらのポートの変更を許しません。adb 開始中に問題が発生した場合、これらの範囲のポートの 1 つが、他のプログラムによって既に使用されているかどうか確認します。

詳細については、*http://developer.android.com/tools/help/adb.html* を参照してください。

Open Agent とデバイスとの接続が不安定な場合の対処方法

Android SDK、または Android Debug Bridge (adb) を使用するその他のツールをインストールしている場合、Silk Test が使用する　adb サーバー以外のサーバーが実行中の可能性があります。バージョンの異

なる adb サーバーが実行中の場合、Open Agent とデバイスとの接続が不安定になったり、接続できない場合があります。

このようなバージョンの不一致によるエラーを避けるには、環境変数 *SILKTEST_ANDROID_HOME* に Android SDK ディレクトリへのパス（C:¥Users¥<ユーザー>¥AppData¥Local¥Android¥android-sdk など）を指定してください。環境変数が設定されていない場合は、Silk Test に付属したバージョンの adb が使用されます。

エラー「メモリの割り当てに失敗しました：8」が発生する理由

エミュレータを開始しているときに、システムが十分なメモリを割り当てることができない場合に、このエラーが表示されます。以下を行ってみてください。

1. エミュレータのメモリ オプションの RAM サイズを下げる
2. Intel HAXM の RAM サイズを下げる RAM サイズを下げるには、IntelHaxm.exe を再度実行して、**Change** を選択します。
3. **タスク マネージャ** を開き、十分なフリー メモリが利用可能かどうかを確認します。不足している場合、プログラムを閉じてメモリを開放してください。

iOS デバイスのテスト時に「Silk Test は指定したアプリを開始できません」というエラーが発生する理由

このエラーが発生する原因として、以下の理由が考えられます。

原因	解決策
iOS デバイスがデベロッパ モードになっていない。	次の 2 種類の方法のいずれかで、デベロッパ モードを有効化できます。 • Xcode がインストールされている Mac にデバイスを接続し、テストするアプリをデバイスで開始します。 • プロビジョニング プロファイルをデバイスに追加します。 1. Xcode を開きます。 2. **Window** > **Devices** を選択します。 3. iOS デバイスをクリックします。 4. **Show Provisioning Profiles** を選択します。 5. プロビジョニング プロファイルを追加します。
デバイスの iOS のバージョンを最近更新した。	1. Xcode を開きます。 2. **Window** > **Devices** を選択します。 3. Xcode がシンボル ファイルを処理するまで待機します。
UI オートメーションが iOS デバイスで有効化されていない。	1. **設定** > **デベロッパ** を選択します。 2. **Enable UI Automation** をオンにします。
Web インスペクタ が iOS デバイスで有効化されていない（モバイル Web アプリケーションのテストの場合）。	1. **設定** > **Safari** > **詳細** をクリックします。 2. **Web インスペクタ** をオンにします。
テストするアプリがテストしようとしている iOS デバイスの iOS バージョン用にビルドされていない。	Xcode を使用してデバイスの iOS バージョン用にアプリをビルドします。
ソフトウェア・アップデート ダイアログ ボックスが iOS デバイス上で開いている。	ダイアログ ボックスを閉じ、ソフトウェアの自動アップデートを無効化します。

原因	解決策
	1. **設定** > **iTunes & App Store** > **自動ダウンロード** を選択します。 2. **アップデート** をオフにします。

Android デバイスの動的ハードウェア コントロールに戻るボタンだけが表示される理由

テストの開始時に Android デバイスや Android エミュレータの画面がロックされると、デバイスやエミュレータが動的ハードウェア コントロールに **戻る** ボタンだけを表示する場合があります。

この問題を解決するには、Open Agent を停止し、デバイスを再起動してから、デバイスの設定を画面のロックをしないように設定してください。

Android デバイスまたはエミュレータにキーボードが表示されなくなる理由

Unicode 文字列をサポートするために、Silk Test は標準キーボードをカスタム キーボードに置き換えます。そして、テストの完了時に元のキーボードに戻します。テスト中にエラーが発生すると、カスタム キーボードが設定されたまま、元に戻らない場合があります。

この問題を解決するには、**設定** > **言語と入力** > **現在のキーボード** を開き、手動で元のキーボードに戻してください。

テスト中にデバイスが応答しなくなる理由

テストの開始時に、デバイス、エミュレータ、シミュレータの画面がロックされると、Silk Test は画面のロックを解除できず、デバイス、エミュレータ、シミュレータが操作に応答しなくなる場合があります。

この問題を解決するには、Open Agent を停止し、デバイスの設定を画面のロックをしないように設定してください。

Information Service を Mac にインストールできない理由

システム環境設定の **セキュリティとプライバシー** で、**一般** タブの **ダウンロードしたアプリケーションの実行許可** 設定が **Mac App Store と確認済みの開発元からのアプリケーションを許可**（デフォルト値）に設定されている場合、Information Service セットアップを開いているときに次のエラー メッセージが表示されます。
"SilkTestInformationService<バージョン>.pkg" は、開発元が未確認のため開けません。

この問題を解決するには、次のいずれかを行います。

- セットアップ ファイルを右クリックして、**開く** を選択します。警告メッセージが表示されても、ファイルを開くことができます。
- **ダウンロードしたアプリケーションの実行許可** 設定を **すべてのアプリケーションを許可** に設定します。
- ファイルを開いた後、システム環境設定の **セキュリティとプライバシー** の **一般** タブを開き、**このまま開く** をクリックします。

Android アプリの記録時に記録ウィンドウが真っ暗になる理由

金融取引を処理するアプリなど、高レベルなセキュリティを必要とする Android アプリでは、Silk Test がアプリのキャプチャをできないようにするために、*FLAG_SECURE* フラグが設定されている可能性があります。 Silk Test は、記録時に Android デバイスのスクリーンショットやビデオを利用しますが、テストする Android アプリにこのフラグが設定されていると、**記録中** ウィンドウにはデバイスの真っ黒な画面が表示されます。Silk Test でこのようなアプリをテストするには、テスト中に FLAG_SECURE フラグを設定しないよう、アプリの開発チームに依頼してください。

Android エミュレータでのテスト時に Silk Test がビデオを表示しない理由

エミュレータがレンダリングにコンピュータのグラフィック カードを使用している場合、Silk Test のビデオ キャプチャが機能しない場合があります。この問題を解決するには、ソフトウェアでグラフィックでエミュレートします。

1. **Android Virtual Device Manager** を開きます。
2. エミュレータの **Actions** 列の **Edit** をクリックします。
3. **Virtual Device Configuration** ダイアログ **Emulated Performance** 領域で、リストから **Software** を選択します。

クラウド環境でのテスト時に Silk Test がビデオを表示しない理由

クラウド環境でテストする場合、必要なポートがオープンになっていないことなどが原因で、テストの記録や再生時にビデオの表示が機能しない場合があります。

この問題を解決するには、infoservice.properties ファイルに WebDriver ホストの URL リストを指定します。このプロパティ ファイルについての詳細は、「*Silk Test Information Service* プロパティの編集」を参照してください。infoservice.disableScreencastHosts オプションをファイルに追加し、次のように入力します。

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

例：

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

ワイルドカードとしてアスタリスク（*）を使用して、**my-webdriver-server-url.com* のような URL パターンを指定することができます。

この設定により、指定したホストでの記録と再生時に、Silk Test はビデオの代わりに連続したスクリーンショットを表示するようになります。

Xcode のインストール バージョンを変更する方法

Xcode の最新版にアップグレードしてしまった場合など、使用している Xcode のバージョンを Silk Test がサポートしていない場合、iOS でテストする際にエラー メッセージが表示される場合があります。

インストールした Xcode のバージョンをサポートするバージョンで置換するには、サポートするバージョンの Xcode を *https://developer.apple.com/download/more/* からダウンロードし、サポートされていないバージョンをダウンロードしたバージョンで置換します。サポートする Xcode のバージョンについての情報は、『リリース ノート』を参照してください。

Mac のディスクの空き容量がなくなった場合の対処

Silk Test は iOS デバイスの自動化に Instruments を使用します。このツールは、/Library/Caches/com.apple.dt.instruments ディレクトリに大きなログ ファイルを生成するため、Mac のディスクの空き容量を圧迫している場合があります。この問題を解決するために、手動であるいは cron ジョブを使用して、これらのログ ファイルを定期的に削除することを Micro Focus は推奨します。たとえば、毎日同じ時間にファイルを削除するには、次の手順を実行します。

1. ターミナルで「sudo crontab -e」を入力します。crontab を root として編集できるエディタが開きます。
2. 次の行を crontab に追加します。

   ```
   0 2 1 * * find /Library/Caches/com.apple.dt.instruments -mtime +10 -delete
   ```
3. crontab を保存します。

この例では、10 日より前のすべてのログ ファイルが、毎日午前 2 時にディレクトリから削除されます。

「Xcode build failed with code 65」というエラー メッセージによりテストが失敗する理由

物理 iOS デバイス上でテストを行う場合、通常このエラーは WebDriverAgent アプリのビルド プロセス中に、プロビジョニング プロファイルでサインされていないか問題があったことを意味します。

デバイスが接続されている Mac マシンで次のコマンドを実行して、実際の問題を確認できます。

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

/Applications/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver フォルダーにある Resources フォルダーが存在し、そのフォルダーに WebDriverAgent.bundle ファイルがあることを確認します。存在しない場合は、このフォルダーを作成し、空の WebDriverAgent.bundle ファイルを作成します。たとえば、次のコマンドを実行します。

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

Developer Tools Access による他のプロセスの制御の要求を抑える方法

iOS 上でテストの実行を開始すると、つぎのメッセージようなメッセージ ボックスが表示される場合があります。
Developer Tools Access は、デバッグを続けるためにほかのプロセスを制御する必要があります。 これを許可するには、パスワードを入力してください。パスワードを入力して許可します。

このメッセージを抑制するには、ターミナルで次のコマンドを実行します。

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

iPad 上のモバイル Web アプリケーションのテスト時に矩形領域がずれる理由

iPad 上のモバイル Web アプリケーションのテスト時にコントロールを囲む矩形領域がずれている場合、複数のブラウザー タブが開いており、タブ バーが表示されている場合があります。この問題を回避するには、1 つを残して、ほかのすべてのタブを閉じてください。

テストの再生に Chrome for Android を使用する方法

デフォルトでは、**ブラウザーの選択** ダイアログ ボックスを使用して、再生に使用するブラウザーを選択できます。

スクリプトをコマンド ラインや CI サーバーから実行する場合は、スクリプトのアプリケーション構成で接続文字列を指定できます。アプリケーション構成で指定したブラウザーを上書きするには、*silktest.configurationName* 環境変数を使用します。

BrowserApplication クラスの browsertype プロパティを使用して、再生に使用するブラウザーの種類を設定することもできます。ただし、browsertype は Chrome for Android の明示的な値を含みません。

テストを再生するブラウザーとして Chrome for Android を使用するように指定するには、browsertype に *GoogleChrome* を設定し、Android をプラットフォームとして指定します。Android が指定されると、デスクトップ マシン上の Google Chrome の代わりに Chrome for Android を使用して、Silk Test はテストを実行します。

使用例

次のサンプル コードは、*silktest.configurationName* を使用して Nexus 7 上の Chrome for Android を使用するテストの基本状態を設定する方法を示しています。

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

次の Java のサンプル コードは、browsertype を使用して Chrome for Android を使用するテストの基本状態を設定する方法を示しています。

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

モバイル Web アプリケーションのテストにおける制限事項

モバイル ブラウザ上でのテストの再生とロケーターの記録のサポートは、サポートされている他のブラウザほど完全なものではありません。モバイル Web アプリケーションに対するテストの再生とロケーターの記録の既知の制限事項を以下に示します。

- 次のクラス、インターフェイス、メソッド、プロパティは、モバイル Web アプリケーションでは現時点ではサポートされません。
  - BrowserApplication クラス。
    - CloseOtherTabs メソッド
    - CloseTab メソッド
    - ExistsTab メソッド
    - GetActiveTab メソッド
    - GetSelectedTab メソッド
    - GetSelectedTabIndex メソッド
    - GetSelectedTabName メソッド
    - GetTabCount メソッド
    - ImageClick メソッド
    - OpenTab メソッド
    - SelectTab メソッド
  - DomElement クラス。
    - DomDoubleClick メソッド
    - DomMouseMove メソッド
    - GetDomAttributeList メソッド
  - IKeyable インターフェイス。
    - PressKeys メソッド
    - ReleaseKeys メソッド
- Silk Test は、iOS 上の Apple Safari を使用した HTML フレームおよび iframe のテストをサポートしません。
- 横固定モードでの記録はシステム バーに仮想ボタンを含むエミュレータに対してサポートされません。このようなエミュレータは、回転を正しく検出せずに、横固定モードのシステム バーを画面の下部ではなく画面の右側に配置します。ただし、このようなエミュレータは縦固定モードで記録することができます。
- モバイル アプリケーションに対する XPath 式では、HTML DOM の HTML 属性だけがサポートされます。 Silk Test は、XPath 式のプロパティをサポートしません。
- Android 上でのモバイル Web アプリケーションのテストでは、Silk Test は、拡大縮小をサポートしません。
- BrowserWindow クラスの以下の JavaScript 警告処理メソッドが、Original Android Stock (AOSP) ブラウザー上でのテストでは機能しません。
  - AcceptAlert メソッド

    - DismissAlert メソッド
    - GetAlertText メソッド
    - IsAlertPresent メソッド
- 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。
- モバイル Web アプリケーションのテストを開始する前に、ブラウザーのタブが開いていないことを確認してください。

  ヒント: iPad 上で Apple Safari のタブを無効に出来ます。**設定** > **Safari** を選択して、**タブバーを表示** をオフにすると無効になります。
- モバイル Web アプリケーションのテスト中に、ブラウザーのタブは 1 つだけ開くことができます。
- Silk Test は、ネイティブ モバイル アプリケーションによって開かれたモバイル Web アプリケーションのテストをサポートしません。

ネイティブ モバイル アプリケーションのテストにおける制限事項

ネイティブ モバイル アプリケーションに対するテストの再生とロケーターの記録の既知の制限事項は次の通りです。

- 次のクラス、インターフェイス、メソッド、プロパティは、ネイティブ モバイル アプリケーションでは現時点ではサポートされません。
    - IKeyable インターフェイス。
        - PressKeys メソッド
        - ReleaseKeys メソッド
    - MobileDevice クラス。
        - iOS 上でのネイティブ モバイル アプリケーションのテスト時に、SetLocation メソッドはサポートされません。
        - Android 6.0 より前のバージョンの Android 上でのネイティブ モバイル アプリケーションのテスト時に、SetLocation メソッドを使用する場合は、**擬似ロケーションを許可** を有効にする必要があります。設定は、Android デバイスまたはエミュレータの設定を開き、**開発者向けオプション** をタップします。
        - Android 6.0 以降上でのネイティブ モバイル アプリケーションのテスト時に、SetLocation メソッドを使用する場合は、**Appium Settings** をアプリとして設定する必要があります。設定は、Android デバイスまたはエミュレータの設定を開き、**開発者向けオプション** > **仮の現在地情報アプリを選択** をタップします。そして、**Appium Settings** を選択します。

          注: Appium Settings という項目は、Android デバイスまたはエミュレータ上の Appium でテストを既に実行した場合にのみ表示されます。
        - iOS 上でのテスト時に、Find メソッドが次の状況では機能しません。
            - path 属性が設定されている場合。
            - 属性値が空の場合。
- iOS 上でのテスト時に、XCUIElementTypeSwitch クラスの getValue メソッドがチェック状態に応じて、文字列 0、1 ではなく、文字列 false、true を返します。
- 横固定モードでの記録はシステム バーに仮想ボタンを持つ Android エミュレータではサポートされません。このようなエミュレータは、回転を正しく検出せずに、横固定モードのシステム バーを画面の下部ではなく画面の右側に配置します。ただし、このようなエミュレータは縦固定モードで記録することができます。
- モバイル アプリケーションに対する XPath 式では、HTML DOM の HTML 属性だけがサポートされます。 Silk Test は、XPath 式のプロパティをサポートしません。

- 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。Silk Test 17.5 Hotfix 1 以降を使用した場合、iOS 上のモバイル アプリケーションをテストするために、Mac 上で複数のユーザー セッションを使用する必要はありません。
- Silk Test は、Android と iOS の両方とも、ネイティブ モバイル アプリケーションのテスト時にテキスト解決をサポートしません。

  テキスト解決は次のメソッドを含みます。

  - TextCapture
  - TextClick
  - TextExists
  - TextRectangle
- Silk Test は、複数の Web ビューを持つネイティブ モバイル アプリケーションのテストをサポートしません。
- iOS 上でのテスト時に、isVisible プロパティの状態は、要素が非表示であったとしても常に true になります。
- iOS 上でのテスト時に、複数のステップを持つスワイプ操作は、あるポイントへスワイプし、マウス ポインタをリリースした後、次のポイントへスワイプします。iOS の以前のバージョンでは、この操作はスワイプ間でマウス ポインタをリリースしません。
- iOS 上でのテスト時に、Silk Test はピンチ以外のマルチタッチ操作をサポートしません。
- iOS 上でのテスト時に、Silk Test は PinchIn メソッドをサポートしません。
- iOS 上でのテスト時に、警告ダイアログ ボックスに対して承認と解除のみを行えます。**キャンセル** ボタンは利用できないため、Silk Test はダイアログを解除できません。デフォルトの操作はダイアログの承認です。
- Android 上でのテスト時に、Silk Test は *Animation* クラスのコントロールに対する自動同期を行いません。
- iOS 上でのテスト時に、Silk Test は UIView.animate 関数または UIView.animateWithDuration 関数を呼び出すコントロールに対する自動同期を行いません。

  アプリケーション デリゲートでアニメーションの速度を速くすることで、問題が回避できる可能性があります。

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

  このようなアニメーションを完全に無効にすることは、アプリケーションの挙動が変わってしまう可能性があるため、Micro Focus では推奨していません。しかし、アニメーションの速度を速めても同期問題が解決できない場合は、次のようにしてアプリケーション デリゲートでアニメーションを完全に無効化することもできます。

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

モバイル Web サイトでのオブジェクトのクリック

自動テストの記録と再生中にオブジェクトをクリックするとき、モバイル Web サイトではデスクトップ Web サイトと比較して、次のような困難があります。

- 拡大/縮小率やデバイス ピクセル比が異なる
- さまざまなモバイル デバイスによって画面サイズが異なる
- モバイル デバイス間でのフォントとグラフィックサイズが異なる (通常、デスクトップ ブラウザの Web サイトよりも小さい)。
- さまざまなモバイルデバイスによってピクセル サイズと解像度が異なる

Silk Test は、このような困難をものともせずに、モバイル Web サイトの適切なオブジェクトをクリックできます。

モバイル デバイスでテストを記録するときに、Silk Test は Click の記録時に座標を記録しません。ただし、クロス ブラウザ テストの場合、再生中に座標が許されています。また、Click に座標を手動で追加することもできます。Silk Test は、これらの座標をオブジェクトの HTML 座標として解釈します。モバイル デバイスのテストの再生時に BrowserWindow の内側の適切なオブジェクトをクリックするために、Silk Test はオブジェクトの HTML 座標に現在の拡大/縮小率を適用します。デバイスのピクセル座標は、オブジェクトの HTML 座標に現在の拡大/縮小率をかけた座標です。

モバイル Web サイトの現在表示されている領域にオブジェクトが表示されていない場合、Silk Test は Web サイトの適切な位置にスクロールします。

例

HTML ページで 100 x 20 ピクセルの固定サイズの DomButton をテストするコードを以下に示します。

異なるモバイル デバイスまたは異なる拡大/縮小率で再生すると、たとえば DomButton は、デバイス画面上では実際は 10 ピクセルの幅かもしれません。しかし、現在の拡大/縮小率の影響は受けず、上記のコードを使用したときに Silk Test は要素の中央をクリックします。これは、Silk Test が座標を HTML 座標として解釈し、現在の拡大/縮小率を適用するためです。

既存のモバイル Web テストの使用方法

Silk Test 17.0 以降では、モバイル Web テストの扱いが、前のバージョンの Silk Test とは異なります。この変更により、以前のモバイル Web テストが Silk Test 17.0 以降では動作しなくなる可能性があります。このトピックでは、Silk Test 17.0 で行われた変更について説明し、既存のモバイル Web テストを Silk Test 17.0 以降で使用できるように変更する方法を説明します。

Silk Test 17.0 でモバイル Web テストに対して行われた変更は、以下の通りです。

- Silk Test の以前のバージョンでは、Windows マシンに USB で接続された iOS デバイスをテストすることができました。Silk Test 17.0 以降では、OS X マシン（Mac）に接続された iOS デバイスに対してのみテストすることができます。
- 以前のバージョンの Silk Test を使用して Android デバイス上のモバイル Web アプリケーションをテストしていた場合、Silk Test 17.0 以降で Web アプリケーションをテストするには、Android デバイスのプロキシ設定を手動で削除する必要があります。Silk Test 17.0 以降では、プロキシは使用しません。プロキシが設定されていると、「プロキシ サーバーに接続できません」というメッセージがデバイスに表示されます。

索引

し

す

せ

て

と

ね

は

ひ

ふ

へ

も

り