

Silk Test 14.0

Silk4NET ユーザー ガイド

Micro Focus
575 Anton Blvd., Suite 510
Costa Mesa, CA 92626

Copyright © Micro Focus 2013. All rights reserved. Silk Test は Borland Software Corporation に由来する成果物を含んでいます, Copyright © 2013 Borland Software Corporation (a Micro Focus company).

MICRO FOCUS, Micro Focus ロゴ、及びその他は Micro Focus IP Development Limited またはその米国、英国、その他の国に存在する子会社・関連会社の商標または登録商標です。

その他、記載の各名称は、各所有社の知的所有財産です。

2013-05-28

目次

ライセンス情報	7
Silk4NET	8
Silk Test 製品スイート	8
製品通知サービス	8
Silk4NET の新機能	9
Silk4NET の新しいスタート ページ	9
イメージ解決のサポート	9
Silk4NET のオブジェクト マップ	9
テスト対象アプリケーションの静的メソッドの呼び出し	9
テスト対象アプリケーションにコードを追加してカスタム WPF および Windows Forms コントロール	10
Silk4NET の Web ベースのトレーニング	10
Silk4NET でのワンクリック ブラウザ切り替え	10
テスト対象アプリケーションの静的メソッドの呼び出し	10
Mozilla Firefox のサポート	10
Google Chrome のサポート	10
Micro Focus へのお問い合わせ	11
Micro Focus SupportLine で必要な情報	11
Silk4NET 入門ガイド	12
基本状態	13
基本状態を変更する	13
アプリケーション構成	14
アプリケーション構成を変更する	14
アプリケーション構成エラー	15
Silk4NET プロジェクトの操作	16
Silk4NET プロジェクトの作成	16
Silk4NET テストの操作	18
プロジェクトへの Silk4NET テストの追加	18
Silk4NET テストの記録	19
記録中および再生中に除外される文字	20
Silk4NET テストの手動作成	20
Locator Spy を使用してロケーターまたはオブジェクト マップ項目をテスト メソッドに追加する	21
Silk4NET テストの実行	21
テスト結果の分析	22
TrueLog を使用したビジュアル実行ログ	23
TrueLog の有効化	23
TrueLog で非 ASCII 文字が正しく表示されない理由	24
Team Foundation Server での Silk4NET の使用	25
Silk4NET テストの TFS での実行	25
TFS で実行した Silk4NET テストの TrueLog ファイルの場所	25
スクリプト オプションの設定	27
TrueLog オプションの設定	27
記録オプションの設定	27
ブラウザの記録オプションの設定	28
カスタム属性の設定	29
無視するクラスの設定	30
記録/再生の対象とする WPF クラスの設定	30

同期オプションの設定	30
再生オプションの設定	31
詳細オプションの設定	32
Silk4NET サンプル テスト	33
オブジェクト解決	34
ロケーターの基本概念	34
オブジェクトタイプと検索範囲	34
属性を使用したオブジェクトの識別	35
ロケーター構文	35
ロケーターの使用	37
Find メソッドの使用	38
ロケーターを使用したオブジェクトの存在確認	38
1つのロケーターによる複数オブジェクトの識別	38
XPathのパフォーマンス問題のトラブルシューティング	39
Locator Spy	40
オブジェクト マップ	41
オブジェクト マップを使用する利点	41
オブジェクト マップのオン/オフの切り替え	41
複数のプロジェクトでの資産の使用	42
Web アプリケーションでのオブジェクト マップの使用	43
オブジェクト マップ項目名の変更	43
オブジェクト マップのロケーターの変更	44
テストアプリケーションからのオブジェクト マップの更新	45
オブジェクト マップ項目のコピー	46
オブジェクト マップ項目の追加	47
スクリプトからオブジェクト マップを開く	47
テストアプリケーションでのオブジェクト マップ項目のハイライト	48
スクリプトでのロケーターからオブジェクト マップ エントリへの移動	48
オブジェクト マップのエラーの検出	49
オブジェクト マップ項目の削除	49
オブジェクト マップのベストプラクティス	50
イメージ解決のサポート	51
イメージ クリックの記録	51
イメージ解決メソッド	51
イメージ資産	52
イメージ資産の作成	52
同じイメージ資産に複数のイメージを追加する	53
スクリプトから資産を開く	53
イメージ検証	53
イメージ検証の作成	54
記録中にイメージ検証を追加する	54
複数のプロジェクトでの資産の使用	55
テストの拡張	57
Windows DLL の呼び出し	57
スクリプトからの Windows DLL の呼び出し	57
DLL 関数の宣言構文	57
DLL 関数への引数の受け渡し	58
DLL 関数への文字列引数の受け渡し	59
DLL 名のエイリアス設定	59
DLL 関数呼び出しの表記規則	59
Microsoft ユーザー補助を使用したオブジェクト解決の向上	60
ユーザー補助の使用	60
ユーザー補助の有効化	61
テキスト解決のサポート	61



カスタム コントロール	62
動的呼び出し	63
テスト対象アプリケーションにコードを追加してカスタム コントロールをテストする	64
Adobe Flex カスタム コントロールのテスト	66
カスタム コントロールの管理	67
特定の環境のテスト	71
Adobe Flex のサポート	71
Adobe Flash Player で実行するための Flex アプリケーションの構成	71
Component Explorer の起動	72
Adobe Flex アプリケーションのテスト	72
Adobe Flex カスタム コントロールのテスト	72
Adobe Flex スクリプトのカスタマイズ	82
同一 Web ページ上の複数の Flex アプリケーションのテスト	82
Adobe AIR のサポート	82
名前またはインデックスを使用する Flex の Select メソッドの概要	83
FlexDataGrid コントロールでの項目の選択	83
Flex アプリケーションのテストの有効化	84
Adobe Flex アプリケーションのスタイル	96
Adobe Flash Player のセキュリティ制約に対応するための Flex アプリケーションの構成	97
Adobe Flex アプリケーションの属性	97
Java AWT/Swing のサポート	97
Java AWT/Swing アプリケーションの属性	98
Java メソッドの動的な呼び出し	98
Java AWT/Swing テクノロジ ドメインでの priorLabel の判別	99
Java SWT と Eclipse RCP のサポート	100
Java SWT クラス リファレンス	100
Java SWT カスタム属性	100
Java SWT アプリケーションの属性	101
Java メソッドの動的な呼び出し	101
.NET のサポート	102
Windows Forms のサポート	102
Windows Presentation Foundation (WPF) のサポート	106
Silverlight アプリケーションのサポート	113
Rumba のサポート	117
Rumba クラス リファレンス	118
Rumba の有効化と無効化	118
Rumba コントロールを識別するためのロケーター属性	118
Rumba での画面検証の使用	119
SAP のサポート	119
SAP クラス リファレンス	120
SAP アプリケーションの属性	120
SAP メソッドの動的な呼び出し	120
SAP コントロールの動的呼び出し	121
SAP の自動セキュリティ設定の構成	121
Windows API ベースのアプリケーションのサポート	121
Win32 クラス リファレンス	122
Windows API ベースのクライアント/サーバー アプリケーションの属性	122
Win32 テクノロジ ドメインにおける priorLabel の決定方法	122
xBrowser のサポート	123
xBrowser 用のテスト オブジェクト	123
xBrowser オブジェクト用のオブジェクト解決	123
xBrowser のページ同期	124
xBrowser における API 再生とネイティブ再生の比較	125

ブラウザの記録オプションの設定	126
マウス移動の詳細設定	127
xBrowser のブラウザ構成の設定	127
ロケータ生成プログラムを xBrowser 用に構成する	129
Internet Explorer 以外のブラウザでのスクリプトの再生	130
Google Chrome を使用したテスト再生の前提条件	130
Google Chrome を使用したテストの制限事項	131
xBrowser のよくある質問	132
Web アプリケーションの属性	136
xBrowser クラス リファレンス	136
64 ビット アプリケーションのサポート	136
サポートする属性の種類	137
Adobe Flex アプリケーションの属性	137
Java AWT/Swing アプリケーションの属性	137
Java SWT アプリケーションの属性	138
SAP アプリケーションの属性	138
Silverlight コントロールを識別するためのロケータ属性	138
Rumba コントロールを識別するためのロケータ属性	139
Web アプリケーションの属性	140
Windows Forms アプリケーションの属性	140
Windows Presentation Foundation (WPF) アプリケーションの属性	141
Windows API ベースのクライアント/サーバー アプリケーションの属性	142
動的ロケータ属性	142

ライセンス情報

評価版を使用している場合を除き、Silk Test を実行するにはライセンスが必要です。

ライセンス モデルは、使用しているクライアントとテストするアプリケーションをベースとします。利用可能なライセンス モデルは、次のアプリケーションの種類をサポートします。

ライセンス モデル	アプリケーションの種類
Web	Web アプリケーション (Java アプレット を含む)
Web + Flex	Web アプリケーション (以下を含む) <ul style="list-style-type: none">• Adobe Flex• Java アプレット
完全	<ul style="list-style-type: none">• Web アプリケーション (以下を含む)<ul style="list-style-type: none">• Adobe Flex• Java アプレット• Adobe Flex• Java AWT/Swing• Java SWT および Eclipse RCP• .NET (Windows Forms および Windows Presentation Foundation (WPF) を含む)• Rumba• Windows API ベース <p> 注: 完全版にアップグレードするには、www.borland.com へ移動してください。</p>
プレミアム	完全版でサポートされるすべてのアプリケーションの種類、および SAP アプリケーション。 <p> 注: プレミアム版にアップグレードするには、www.borland.com へ移動してください。</p>

Silk4NET

Silk4NET は Microsoft Visual Studio に対応した Silk Test プラグインです。Silk4NET を使用すると、機能テスト、回帰テスト、およびローカリゼーション テストの作成と管理を Visual Studio で直接、効率的に実行できます。Silk4NET で実行できる Visual Studio 内の作業は、以下のとおりです。

- Visual Basic .NET を使用してテストを開発します。
- C# を使用してテストを開発します。
- Microsoft テスト環境内のテスト計画の一環としてテストを実行できます。
- ビルドプロセスの一環としてテストを実行できます。
- テスト結果を表示します。

Silk4NET では、広範囲にわたる アプリケーション テクノロジーのテストがサポートされています。Silk4NET は、複雑なテストに適用した場合にも自動化の利点を実現されるように設計されており、開発者が使い慣れた環境に直接、テストの自動化機能を追加するため、テスト アプリケーションに加えられた変更に対応できます。

また、Silk4NET の強力なテスト フレームワークを使用すると、複数のテスト プロジェクトに対してテストの再利用が促進されるため、より高い投資収益率 (ROI) を達成できます。テストスイートの構築と保守に要する時間が短縮されるため、QA 担当者はテスト範囲を拡張し、アプリケーション品質を最適化することができます。

Silk Test 製品スイート

Silk Test 製品スイートには、以下のコンポーネントが含まれています。

- Silk Test Workbench : Silk Test Workbench は、新しいネイティブ品質テスト環境です。上級者用の .NET スクリプトと、より幅広い利用者がテストを行えるようにする使いやすいビジュアル テストが提供されます。
- Silk4NET : Silk4NET Visual Studio プラグインを使用すると、Visual Studio で直接 Visual Basic または C# のテスト スクリプトを作成できます。
- Silk4J : Silk4J Eclipse プラグインを使用すると、Eclipse 環境で直接 Java ベースのテスト スクリプトを作成できます。
- Silk Test Classic : Silk Test Classic は、従来の 4Test Silk Test 製品です。
- Silk Test Agent : Silk Test Agent は、テストのコマンドを GUI 固有のコマンドに変換するソフトウェア プロセスです。つまり、テストするアプリケーションをエージェントが動かし、監視しています。ホストマシン上で 1 つのエージェントをローカルに実行できます。ネットワーク環境では、任意の数のエージェントをリモート マシン上で実行できます。

インストールする製品スイートによって、使用できるコンポーネントが決まります。すべてのコンポーネントをインストールするには、完全インストール オプションを選択します。Silk Test Classic を除くすべてのコンポーネントをインストールするには、標準インストール オプションを選択します。

製品通知サービス

製品通知サービスはシステム トレイで実行されるアプリケーションで、Silk Test の更新が入手可能になった場合にそれを知らせます。また、更新ページに移動可能なリンクも提供します。

サービスの実行

システムトレイで、更新通知アイコンをクリックすると、製品通知サービスアプリケーションが起動します。

インストール済みバージョン 現在インストールされている Silk Test アプリケーションのバージョン番号を提供します。

更新バージョン 利用可能な場合、次回のマイナー更新のリンクとバージョン番号を提供します。

新しいバージョン 利用可能な場合、次回のフルリリースのリンクとバージョン番号を提供します。

設定 **設定** ボタンをクリックして、**設定** ウィンドウを開きます。通知サービスで更新をチェックするかどうかと、その頻度を選択します。

Silk4NET の新機能

このセクションでは、Silk4NET に対して行われた重要な改善と変更を示します。

Silk4NET の新しいスタート ページ

Silk4NET を開始すると表示される新しいスタート ページをエン트리 ポイントとして使用し、Silk4NET で使用できる機能を簡単に見つけることができます。このスタート ページを使用して、以下のオプションから選択し、処理を進めることができます。

- 新機能のリストを表示する。
- 共通の Silk4NET 機能に直接アクセスする。
- よく使用される機能に関するビデオを表示する。
- サンプル プロジェクトを開く。
- Silk4NET の操作を開始する方法が説明されているヘルプ トピックにアクセスする。
- リンクされているリソースを表示して、追加サポートとトレーニングを確認する。
- Silk Test コミュニティーで最新のフォーラム項目とブログ投稿を表示する。

スタート ページは、そのページを閉じるか、**スタートアップ時に開始画面を表示する** オプションを選択解除するまで開かれたままになります。

イメージ解決のサポート

イメージ解決メソッドでは、オブジェクト解決またはテキスト解決で識別できない、高度にカスタマイズされたコントロールを含むテスト アプリケーションを便利に操作できます。相対座標でのクリックの代わりにイメージ クリックを使用して、特定イメージをクリックできます。また、テスト対象アプリケーションの UI にあるオブジェクトのイメージ表現を使用して、オブジェクトが存在し、予想通りに表示されることを確認できます。

Silk4NET のオブジェクト マップ

Silk4NET でオブジェクト マップを使用できるようになりました。オブジェクト マップはテスト資産の一種であり、コントロールまたはウィンドウのロケーターではなく、コントロールまたはウィンドウに論理名 (エイリアス) を関連付ける項目が含まれています。コントロールがオブジェクト マップ資産に登録されると、スクリプトでのそのコントロールに対する参照はすべて、実際のロケーター名ではなく、そのエイリアスによって行われます。

テスト対象アプリケーションの静的メソッドの呼び出し

テスト対象アプリケーションの基になるオブジェクトが定義するすべてのメソッドを動的に呼び出せることに加えて、Windows Forms または WPF コントロールでは、テスト対象アプリケーションのオブジェク

トで定義されたすべての静的メソッドと、すべてのユーザー定義の静的メソッドを動的に呼び出せるようになりました。

テスト対象アプリケーションにコードを追加してカスタム WPF および Windows Forms コントロールをテストする

WPF Windows Forms アプリケーションをテストする場合、テスト対象アプリケーション (AUT) の実際のコントロールを操作する静的メソッドを含むコードを追加できます。このコードをスクリプトの再生時に AUT に挿入することも、コードを AUT にコンパイルすることもできます。その後、このコードを再生時にスクリプトから呼び出し、カスタム コントロールを操作できます。

Silk4NET の Web ベースのトレーニング

Silk4NET の新しい無料 Web ベース トレーニングは、[Micro Focus Training Store](#) で入手できます。

Silk4NET でのワンクリック ブラウザ切り替え

テスト再生とロケータ記録に使用されるブラウザの変更は、使用するブラウザのアイコンをクリックするだけで行えます。

テスト対象アプリケーションの静的メソッドの呼び出し

テスト対象アプリケーションの基になるオブジェクトが定義するすべてのメソッドを動的に呼び出せることに加えて、Windows Forms または WPF コントロールでは、テスト対象アプリケーションのオブジェクトで定義されたすべての静的メソッドと、すべてのユーザー定義の静的メソッドを動的に呼び出せるようになりました。

Mozilla Firefox のサポート

Silk Test では現在、以下のリリースで実行されているアプリケーションの再生をサポートしています。

- Mozilla Firefox 17
- Mozilla Firefox 18
- Mozilla Firefox 19
- Mozilla Firefox 20
- Mozilla Firefox 21

Google Chrome のサポート

Silk Test では現在、以下のリリースで実行されているアプリケーションの再生をサポートしています。

- Google Chrome 22
- Google Chrome 23
- Google Chrome 24
- Google Chrome 25
- Google Chrome 26
- Google Chrome 27

Micro Focus へのお問い合わせ

Micro Focus は、世界的規模のテクニカル サポートおよびコンサルティング サービスを提供します。すべての顧客のビジネスを成功に導くために、信頼できるサービスをタイムリーに提供するように Micro Focus はワールドワイドのサポート体制を整えています。

保守およびサポート契約を結んだすべてのお客様、および製品を評価中のお客様は、カスタマ サポートを受けることができます。弊社の熟練したスタッフが、可能な限り迅速に専門家としてお客様の質問にお答えします。

<http://supportline.microfocus.com/assistedservices.asp> にアクセスするか、またはメールを supportline@microfocus.com に送信して、Micro Focus SupportLine と直接連絡できます。

また、<http://supportline.microfocus.com> の Micro Focus SupportLine では、最新のサポートに関するニュースや、さまざまなサポート情報を得ることができます。このサイトに初めてアクセスした場合は、ユーザー登録が必要な場合があります。

Micro Focus SupportLine で必要な情報

Micro Focus SupportLine に連絡する際、可能な場合は以下の情報も提供してください。詳細な情報をご提供いただければ、Micro Focus SupportLine はより効果的なサポートが可能になります。

- 問題の原因と考えられるすべての製品の名前とバージョン番号。
- コンピュータのメーカーと機種。
- オペレーティング システム名とバージョン、プロセッサ、メモリの詳細などのシステム情報。
- 問題を再現する手順などの、問題の詳細な説明。
- 関係するエラー メッセージの正確な表現。
- シリアル番号。

これらの番号を見つけるには、Micro Focus から受信した Electronic Product Delivery Notice 電子メールの件名および本文を参照してください。

Silk4NET 入門ガイド

Silk4NET を使用するには、次の操作を実行します。

1. Silk4NET プロジェクトを作成します。
2. プロジェクトに Silk4NET テストを追加します。プロジェクトには記録されたテストおよび、手動でスクリプトを作成したテストを含めることができます。
3. テストを実行します。
4. テスト結果を分析します。

基本状態

アプリケーションの基本状態とは、各テストケースの実行開始前にアプリケーションに想定される既知の安定した状態です。アプリケーションは、各テストケースの実行が終了したあとに基本状態に戻る場合があります。大抵の場合この状態は、アプリケーションを最初に起動したときの状態になります。

Web アプリケーション構成 や標準構成用のクラスを作成するとき、Silk4NET は自動的に基本状態を作成します。

基本状態はテストの整合性を保障するための重要な一因です。各テストケースが安定した基本状態から開始することができることを保障することによって、あるテストケースのエラーによって、後続のテストケースが失敗しないことを保障することができます。

Silk4NET は、次の段階の間に、アプリケーションがその基本状態にあることを自動的に保障します。

- テストの実行前
- テストの実行中
- テストが成功裏に完了した後

 **注:** Silk4NET は、基本状態とすべての Silk4NET オプションを config.silk4net 構成ファイルに保存します。Silk4NET は、Silk4NET プロジェクトごとにこのようなファイルを作成します。

基本状態を変更する

必要に応じて、基本状態の実行可能ファイルの場所、作業ディレクトリ、ロケーター、URL を変更できます。たとえば、テスト用の Web サイト上で以前にテストしていたものを、本番の Web サイトに対してテストを行いたい場合には、基本状態の URL を変更すれば、新しい環境でテストが実行されるようになります。

1. **Silk4NET** の隣にあるドロップダウン矢印をクリックして、**アプリケーション構成の編集** を選択します。**アプリケーション構成の編集** ダイアログ ボックスが開き、既存のアプリケーション構成がリストされます。
2. **基本状態の編集** をクリックします。**基本状態の編集** ダイアログ ボックスが開きます。
3. **実行可能ファイル** テキスト ボックスに、テストするアプリケーションの実行可能ファイルの名前とファイルへのパスを入力します。
たとえば、Internet Explorer を指定する場合には、「C:¥Program Files¥Internet Explorer ¥IEXPLORE.EXE」と入力します。
4. 実行可能ファイルと組み合わせてコマンド ライン パターンを使用するには、**コマンド ライン引数** テキスト ボックスにコマンド ライン パターンを入力します。
5. テストするアプリケーションがディレクトリに依存する場合は、**作業ディレクトリ** テキスト ボックスにディレクトリを指定します。
たとえば、Java アプリケーションを起動するバッチ ファイルを使用する場合には、バッチ ファイルは JAR ファイルを相対パスで参照することができます。この場合、正しく相対パスが処理されるように作業ディレクトリを指定します。
6. **ロケーター** テキスト ボックスに、アプリケーションのメイン ウィンドウを識別する XPath ロケーター文字列またはオブジェクト マップ ID を入力します。
7. Web サイトをテストする場合は、**URL** テキスト ボックスに、テストを開始するときに起動する Web ページの Web アドレスを入力します。
8. **OK** をクリックします。

アプリケーション構成

アプリケーション構成は、テストするアプリケーションに Silk4NET が接続する方法を定義します。Silk4NET は、基本状態を作成するときに、アプリケーション構成を自動的に作成します。しかし、アプリケーション構成を追加したり、変更や削除をすることが必要になる場合があります。たとえば、データベースを変更するアプリケーションをテストしているときに、データベースの内容を確認するためにデータベースのビューアー ツールを使用する場合には、そのデータベースのビューアー ツール用のアプリケーション構成を追加する必要があります。

アプリケーション構成には次のものが含まれます：

- 実行可能ファイル パターン

このパターンに一致するすべてのプロセスは、テストに対して有効化されます。たとえば、Internet Explorer の実行可能パターンは *%IEXPLORE.EXE です。実行可能ファイルの名前が IEXPLORE.EXE で、任意のディレクトリに置かれているプロセスはすべて有効化されます。

- コマンド ライン パターン

コマンドライン パターンは、テストを行うために有効化されるプロセスの制約に使用される補足パターンで、コマンドライン引数の一部 (実行可能ファイル名の後ろ部分) をマッピングすることにより行います。コマンド ライン パターンを含むアプリケーション構成では、実行可能パターンとコマンド ラインパターンの両方に一致するプロセスのみが、テストに対して有効化されます。コマンド ラインパターンが定義されていない場合は、指定された実行可能ファイルパターンを持つすべてのプロセスが有効化されます。コマンドラインの使用は、Java アプリケーションに対して特に有益です。これは、ほとんどの Java プログラムが javaw.exe を使用して実行されるためです。つまり、典型的な Java アプリケーションに対してアプリケーション構成を作成する場合、実行可能パターンには *%javaw.exe が使用され、このパターンはすべての Java プロセスに一致します。このような場合、コマンドラインパターンを使用して、該当するアプリケーションのみがテストに対して有効化されるようにします。たとえば、アプリケーションのコマンド ラインが **com.example.MyMainClass** で終わる場合には、コマンドラインパターンに ***com.example.MyMainClass** を使用します。

アプリケーション構成を変更する

アプリケーション構成は、テストするアプリケーションに Silk4NET が接続する方法を定義します。Silk4NET は、基本状態を作成するときに、アプリケーション構成を自動的に作成します。しかし、アプリケーション構成を追加したり、変更や削除をすることが必要になる場合があります。たとえば、データベースを変更するアプリケーションをテストしているときに、データベースの内容を確認するためにデータベースのビューアー ツールを使用する場合には、そのデータベースのビューアー ツール用のアプリケーション構成を追加する必要があります。

1. Silk4NET の隣にあるドロップダウン矢印をクリックして、**アプリケーション構成の編集** を選択します。**アプリケーション構成の編集** ダイアログ ボックスが開き、既存のアプリケーション構成がリストされます。
2. アプリケーション構成をさらに追加するには、**追加** をクリックします。**新規アプリケーション構成** ウィザードが開きます。
 - a) テストするアプリケーションの種類を選択して **次へ** をクリックします。
 - b) テストしたいアプリケーションに対応する構成をクリックします。

テストするアプリケーションがリストに表示されない場合は、**キャプションを持たないプロセスを表示しない** チェック ボックスをオフにします。このオプションはデフォルトでオンになっており、キャプションが付いているアプリケーションのみをフィルタ処理するために使用します。
 - c) Web アプリケーションのテストを選択した場合、既存のブラウザ インスタンスを使用するか、新しいブラウザを起動するかを選択します。

- d) **終了** をクリックします。アプリケーションの実行可能なパターンが、**アプリケーション構成の編集** ダイアログ ボックスのアプリケーション構成のリストに追加されます。
3. アプリケーション構成を削除するには、該当するアプリケーション構成の隣にある **削除** をクリックします。
 4. アプリケーション構成を編集するには、**基本状態の編集** をクリックします。**基本状態の編集** ダイアログ ボックスが開きます。
 5. **OK** をクリックします。

アプリケーション構成エラー

プログラムをアプリケーションにアタッチできない場合、以下のエラー メッセージが表示されます。アプリケーション <Application Name> にアタッチするのに失敗しました。詳細については、ヘルプを参照してください。

この場合、以下の表に示されている 1 つ以上の問題が原因である可能性があります。

問題	原因	解決策
タイムアウト	<ul style="list-style-type: none"> • システムが遅すぎます。 • システムのメモリ サイズが小さすぎます。 	速いシステムを使用するか、現在使用しているシステムのメモリ使用量を減らします。
ユーザー アカウント制御 (UAC) の失敗	システムの管理者権限がありません。	管理者権限を持つユーザー アカウントでログインします。
コマンド ライン パターン	コマンド ライン パターンが固有すぎます。この問題は特に Java の場合に発生します。再生が意図したとおりに機能しないことがあります。	パターンから不明瞭なコマンドを削除します。

Silk4NET プロジェクトの操作

このセクションでは、Silk4NET プロジェクトの使用方法について説明します。

Silk4NET プロジェクトには、Silk4NET を使用してアプリケーションの機能をテストするために必要なリソースがすべて含まれています。

Silk4NET プロジェクトの作成

1. **Silk4NET > 新しいプロジェクト** または **ファイル > 新規作成 > プロジェクト** をクリックします。 **新しいプロジェクト** ダイアログ ボックスが表示されます。
2. **インストールされたテンプレート** で、**Visual Basic** または **Visual C#** をクリックし、**Silk4NET プロジェクト** をダブルクリックします。 **Silk4NET テストの作成** ダイアログ ボックスが開きます。
3. 次のいずれかのオプション ボタンをクリックして、Silk4NET テストの作成方法を選択します。

Silk4NET テストの記録 テスト対象アプリケーションに対する操作および検証を記録し、記録されたオートメーション ステートメントを含む新しいテストを生成します。

空の Silk4NET テストの作成 オートメーション ステートメントを後で入力できる空のテストを作成します。

4. **OK** をクリックします。空の Silk4NET テストを作成するように選択した場合は、Silk4NET プロジェクトを含む新しいソリューションが作成されます。また、このプロジェクトには、言語固有の以下のファイル名を使用して、Silk4NET テストも作成されます。

- UnitTest1.vb
- UnitTest1.cs

5. 新しい Silk4NET テストを記録するように選択した場合は、**新規アプリケーション構成** ウィザードが開きます。テストするアプリケーションのタイプを選択します。

- 標準アプリケーションをテストする場合は、**標準テスト構成** をクリックします。
- Web アプリケーションをテストする場合は、**Web サイト テスト構成** をクリックします。

6. **次へ** をクリックします。

7. 標準アプリケーションをテストするように選択した場合は、**新規標準構成** ダイアログ ボックスが開きます。テストしたいアプリケーションに対応する構成をクリックします。

テストするアプリケーションがリストに表示されない場合は、**キャプションを持たないプロセスを表示しない** チェック ボックスをオフにします。このオプションはデフォルトでオンになっており、キャプションが付いているアプリケーションのみをフィルタ処理するために使用します。

8. Web アプリケーションのテストを選択した場合、**新規 Web サイト構成** ダイアログ ボックスが開きます。**ブラウザの種類** リスト ボックスから、テストするブラウザの種類を選択します。

- a) **ブラウザ インスタンス** セクションで、次のオプション ボタンをクリックします。

既存のブラウザを使用する すでに開いているブラウザを使用する場合は、このオプション ボタンをクリックします。たとえば、テストしたい Web ページがすでにブラウザ上に表示されている場合などに、このオプションを使用します。

新しいブラウザを開始する テストを構成する際に、新しいブラウザ インスタンスを開始する場合には、このオプション ボタンをクリックします。次に、**ブラウズする URL** テキスト ボックスで、開く Web ページを指定します。

9. **終了** をクリックします。テスト メソッドの再生で Google Chrome の既存のインスタンスを選択した場合は、Silk Test Recorder がオートメーション サポートが含まれているかどうかをチェックします。

オートメーション サポートが含まれていない場合は、Silk Test Recorder が Google Chrome を再起動する必要があることを通知します。アプリケーションと **記録中** ダイアログ ボックスが開きます。



注: ソリューション エクスプローラーのコンテキスト メニューを使って Silk4NET プロジェクトを既存のソリューションに追加することもできます。

Silk4NET テストの操作


Silk4NET テストの使用方法について説明します。

AUT に対して行われたユーザーの操作を記録するか、Visual Basic または Visual C# のテスト クラスおよびメソッドのスクリプトを手動で記述することで、Silk4NET テストを新規に作成できます。

プロジェクトへの Silk4NET テストの追加

既存の Silk4NET またはテスト プロジェクトにのみ Silk4NET テストを追加できます。Silk4NET またはテスト プロジェクトが存在しない場合は、Silk4NET テストを作成する前に Silk4NET またはテスト プロジェクトを作成してください。

1. **Silk4NET > 新しいテスト** または **プロジェクト > 新しい項目の追加** をクリックします。

 **注:** ソリューションに複数の Silk4NET プロジェクトが存在する場合、**プロジェクトの選択** でリストから新しいテストを追加するプロジェクトを選択します。

新しい項目の追加 ダイアログ ボックスが開きます。

2. **インストールされたテンプレート** で次のいずれかをクリックします。

- プロジェクトが Visual Basic プロジェクトの場合は、**共通項目 > Silk4NET テスト** をクリックします。
- プロジェクトが Visual C# プロジェクトの場合は、**Visual C# アイテム > Silk4NET テスト** をクリックします。

Silk4NET テストの作成 ダイアログ ボックスが開きます。

3. 次のいずれかのオプション ボタンをクリックして、Silk4NET テストの作成方法を選択します。

Silk4NET テストの記録 テスト対象アプリケーションに対する操作および検証を記録し、記録されたオートメーション ステートメントを含む新しいテストを生成します。

空の Silk4NET テストの作成 オートメーション ステートメントを後で入力できる空のテストを作成します。

4. **OK** をクリックします。空の Silk4NET テストを作成するように選択した場合は、Silk4NET プロジェクトを含む新しいソリューションが作成されます。また、このプロジェクトには、言語固有の以下のファイル名を使用して、Silk4NET テストも作成されます。

- UnitTest1.vb
- UnitTest1.cs

5. Web アプリケーションのテストを選択した場合、**新規 Web サイト構成** ダイアログ ボックスが開きます。**ブラウザの種類** リスト ボックスから、テストするブラウザの種類を選択します。

a) **ブラウザ インスタンス** セクションで、次のオプション ボタンをクリックします。


既存のブラウザを使用する すでに開いているブラウザを使用する場合は、このオプション ボタンをクリックします。たとえば、テストしたい Web ページがすでにブラウザ上に表示されている場合などに、このオプションを使用します。

新しいブラウザを開始する テストを構成する際に、新しいブラウザ インスタンスを開始する場合には、このオプション ボタンをクリックします。次に、**ブラウズする URL** テキスト ボックスで、開く Web ページを指定します。

6. **終了** をクリックします。テスト メソッドの再生で Google Chrome の既存のインスタンスを選択した場合は、Silk Test Recorder がオートメーション サポートが含まれているかどうかをチェックします。


オートメーション サポートが含まれていない場合は、Silk Test Recorder が Google Chrome を再起動する必要があることを通知します。アプリケーションと **記録中** ダイアログ ボックスが開きます。

テストを記録するように選択した場合は、記録したテストがプロジェクトに追加されます。空のテストを追加するように選択した場合は、空の Silk4NET テストがプロジェクトに追加されます。

 **注:** ソリューション エクスプローラーのコンテキスト メニューを使って Silk4NET テストを Silk4NET またはテスト プロジェクトに追加することもできます。

Silk4NET テストの記録

1. Silk4NET > **新しいテスト** または **プロジェクト > 新しい項目の追加** をクリックします。

 **注:** ソリューションに複数の Silk4NET プロジェクトが存在する場合、**プロジェクトの選択** でリストから新しいテストを追加するプロジェクトを選択します。

新しい項目の追加 ダイアログ ボックスが開きます。

2. **インストールされたテンプレート** で次のいずれかをクリックします。

- プロジェクトが Visual Basic プロジェクトの場合は、**共通項目 > Silk4NET テスト** をクリックします。
- プロジェクトが Visual C# プロジェクトの場合は、**Visual C# アイテム > Silk4NET テスト** をクリックします。

Silk4NET テストの作成 ダイアログ ボックスが開きます。

3. **Silk4NET テストの記録** をクリックします。 **新規アプリケーション構成** ウィザードが開きます。

4. テストするアプリケーションのタイプを選択します。

- 標準アプリケーションをテストする場合は、**標準テスト構成** をクリックします。
- Web アプリケーションをテストする場合は、**Web サイト テスト構成** をクリックします。

5. Web アプリケーションのテストを選択した場合、**新規 Web サイト構成** ダイアログ ボックスが開きます。**ブラウザの種類** リスト ボックスから、テストするブラウザの種類を選択します。

a) **ブラウザ インスタンス** セクションで、次のオプション ボタンをクリックします。

既存のブラウザを使用する すでに開いているブラウザを使用する場合は、このオプション ボタンをクリックします。たとえば、テストしたい Web ページがすでにブラウザ上に表示されている場合などに、このオプションを使用します。

新しいブラウザを開始する テストを構成する際に、新しいブラウザ インスタンスを開始する場合には、このオプション ボタンをクリックします。次に、**ブラウズする URL** テキスト ボックスで、開く Web ページを指定します。

6. **終了** をクリックします。テスト メソッドの再生で Google Chrome の既存のインスタンスを選択した場合は、Silk Test Recorder がオートメーション サポートが含まれているかどうかをチェックします。オートメーション サポートが含まれていない場合は、Silk Test Recorder が Google Chrome を再起動する必要があることを通知します。アプリケーションと **記録中** ダイアログ ボックスが開きます。


7. 記録を行う、テスト対象アプリケーションとの対話を実行します。

詳細については、『Silk Test Recorder ヘルプ』を参照してください。

8. 記録が終了したら、**記録** ダイアログ ボックスの **記録の停止** をクリックします。**記録完了** ダイアログ ボックスが開きます。このダイアログ ボックスの **再生** をクリックすると、記録したテストを再生することができます。

- Visual Studio 2010 を使用している場合は、Visual Studio の **テスト ビュー** を使って、テストの再生および管理を行うことができます。
- Visual Studio 2012 を使用している場合は、Visual Studio の **テスト エクスプローラー** を使って、テストの再生および管理を行うことができます。

記録された対話は、プロジェクトにファイルとして追加されます。生成されたファイルのデフォルトのファイル名は、プロジェクトのデフォルトプログラミング言語に応じて、UnitTest<Index>.cs または UnitTest<Index>.vb になります。たとえば、Visual Basic プロジェクトの初回テストを記録している場合、生成されたファイルの名前は UnitTest1.vb になります。

 **注:** 新しいプロジェクトを作成し、そこに新しいテスト記録することもできます。

記録中および再生中に除外される文字

記録および再生中に Silk Test が無視する文字を以下に示します。

文字	コントロール
...	MenuItem
タブ	MenuItem
&	すべてのコントロール。アンパサンド (&) はアクセラレータとして使用されるため、記録されません。

Silk4NET テストの手動作成

1. Silk4NET テスをプロジェクトに追加します。
2. オプション : 特定のアプリケーション テクノロジーのコントロールのサポートを追加するには、以下の例に示すように、アプリケーション テクノロジーの名前空間を参照するテストの先頭にインポート ステートメントを含める必要があります。


```
'Visual Basic .NET
Imports SilkTest.Ntf.Wpf
Imports SilkTest.Ntf.XBrowser
Imports SilkTest.Ntf.Win32
```

```
//C#
using SilkTest.Ntf.Wpf;
using SilkTest.Ntf.XBrowser;
using SilkTest.Ntf.Win32;
```

3. テスト アプリケーションの基本状態を構成します。例 :

```
'Visual Basic .NET
Dim baseState = New BrowserBaseState(BrowserType.InternetExplorer,
"www.borland.com")
baseState.Execute()
```

```
//C#
BrowserBaseState baseState = new BrowserBaseState(BrowserType.InternetExplorer,
"www.borland.com");
baseState.Execute();
```

 **注:** 基本状態を使用すると、テストするアプリケーションがフォアグラウンドで実行中であることを保証できます。これにより、テストが常に同じアプリケーション状態で開始されることが保証され、信頼性が高まります。基本状態を使用するには、メインウィンドウの外観、およびテストするアプリケーションが実行されていない場合のアプリケーションの起動方法を指定する必要があります。基本状態の作成は任意です。ただし、ベストプラクティスとして、基本状態を作成することをお勧めします。

4. テスト アプリケーションの目的の機能をテストするクラスとメソッドを追加します。

Locator Spy を使用してロケーターまたはオブジェクト マップ項目をテスト メソッドに追加する

Locator Spy を使用して、ロケーターまたはオブジェクト マップ項目を手動でキャプチャし、ロケーターまたはオブジェクト マップ項目をテスト メソッドにコピーします。たとえば、**Locator Spy** を使って、GUI オブジェクトのキャプションや XPath ロケーター文字列を識別できます。そして、関係するロケーター文字列や属性をスクリプト内のテストメソッドにコピーします。

1. 変更したいテスト クラスを開きます。
2. **Silk4NET > ロケーターの記録** をクリックします。 **Locator Spy** が開きます。
3. 省略可能:オブジェクト マップ項目の代わりにロケーターを **ロケーター** 列に表示するには、**オブジェクト マップ識別子の表示** チェック ボックスをオフにします。

オブジェクト マップ項目名は、コントロールまたはウィンドウに対して、コントロールやウィンドウのロケーターではなく論理名 (エイリアス) を関連付けます。デフォルトでは、オブジェクト マップ項目名が表示されます。



注: このチェック ボックスをオンまたはオフにすると、変更が自動的にロケーターの詳細に反映されます。 **ロケーターの詳細** テーブルのエントリを更新するには、エントリをクリックします。

4. 記録するオブジェクトの上にマウスを移動します。 関連するロケーター文字列またはオブジェクト マップ項目は、**選択済みロケーター** テキスト ボックスに表示されます。
5. **Ctrl+Alt** を押してオブジェクトをキャプチャします。



注: **スクリプト オプション** ダイアログ ボックスの **全般記録オプション** ページで別の記録停止キー操作を指定した場合は、**Ctrl+Shift** を押してオブジェクトをキャプチャします。

6. 省略可能:**追加のロケーター属性の表示** をクリックすると、関係する属性のすべてが **ロケーター属性** テーブルに表示されます。
7. 省略可能:記録したロケーター属性は、**ロケーター属性** テーブルの別のロケーター属性で置き換えることができます。

たとえば、記録したロケーター属性が、次のようになっているとします。

```
/BrowserApplication//BrowserWindow//input[@id='loginButton']
```

ロケーター属性 テーブルに textContents Login がある場合、このロケーターを次のように手動で変更できます。

```
/BrowserApplication//BrowserWindow//input[@textContents='Login']
```

新しいロケーターは、**選択済みロケーター** テキスト ボックスに表示されます。

8. ロケーターをコピーするには、**ロケーターをクリップボードにコピー** をクリックします。
選択済みロケーター テキスト ボックスで、コピーするロケーター文字列の位置をマークし、マークしたテキストを右クリックして **コピー** をクリックすることもできます。
9. スクリプト内で、記録したロケーターを貼り付ける位置にカーソルを置きます。
たとえば、スクリプト内の Find メソッドの該当するパラメータにカーソルを置きます。
ロケーターを貼り付けるテスト メソッドでは、ロケーターをパラメータとして受け取れるメソッドを使用する必要があります。 **Locator Spy** を使用することで、クエリ文字列が正しいことが保障されます。
- 10.ロケーターまたはオブジェクト マップ項目をテスト ケースまたはクリップボードにコピーします。
- 11.閉じる をクリックします。


Silk4NET テストの実行

このトピックでは、Visual Studio で Silk4NET テストを実行する方法について説明します。

1. 選択したプロジェクトまたはソリューションで利用可能なすべてのテストを表示するには:
 - Visual Studio 2010 の場合、 **テスト > ウィンドウ > テスト ビュー** をクリックします。
 - Visual Studio 2012 の場合、 **テスト > ウィンドウ > テスト エクスプローラー** をクリックします。
2. **テスト ビュー** または **テスト エクスプローラー**（使用している Visual Studio のバージョンによって変わります）で、実行するテストを選択します。
3. 選択項目を右クリックし、次のいずれかをクリックします。
 - Visual Studio 2010 の場合、 **選択範囲の実行**。
 - Visual Studio 2012 の場合、 **選択したテストの実行**。

選択したプロジェクトまたはソリューションですべてのテストを実行するには、**テスト ビュー** または **テスト エクスプローラー**（使用している Visual Studio のバージョンによって変わります）で、**すべて実行** をクリックします。

4. TrueLog を使用してテスト結果を調べるには、**結果の検討** をクリックします。ダイアログ ボックスを閉じるには、**OK** をクリックします。

 **注:** Visual Studio は、テストの実行時にテストの実行に必要なコンポーネントを開始し、テスト実行の完了時に Open Agent やすべての開いているブラウザ ウィンドウを終了し、すべてをクリーンアップします。


テスト結果の分析

テストを実行後、テスト結果を確認し、テスト実行の成功または失敗を分析できます。

1. Silk4NET テストを実行します。実行が終了すると、**再生完了** ダイアログ ボックスが開きます。
2. **結果の検討** をクリックし、TrueLog を使用してテスト結果を調べます。Silk TrueLog Explorer が開きます。
3. Silk TrueLog Explorer で結果をクリックします。
テストに失敗すると、Silk TrueLog Explorer によってスクリーンショットがキャプチャされます。

TrueLog を使用したビジュアル実行ログ

TrueLog は、ビジュアルな検証を通じてテスト ケースの失敗の根本的な原因の分析を単純化するための強力なテクノロジーです。テストの結果は、TrueLog Explorer で検証できます。テストの実行中にエラーが発生すると、TrueLog はそのエラーが発生したスクリプトの行を簡単に特定し、問題を解決できるようにします。

 **注:** TrueLog は、スクリプトに対して単一のローカル エージェントまたはリモート エージェントのみをサポートしています。たとえば、1 つのマシンでアプリケーションをテストし、そのアプリケーションが別のマシンのデータベースにデータを書き込む場合のように、複数のエージェントを使用する場合は、スクリプトで使用された最初のエージェントに対してのみ TrueLog が書き出されます。リモート エージェントを使用する場合は、リモート マシンにも TrueLog ファイルが書き出されます。

TrueLog Explorer の詳細については、**スタート > プログラム > Silk > Silk Test > ドキュメント** にある *Silk TrueLog Explorer* ユーザー ガイド を参照してください。

Silk4NET で TrueLog を有効にして、Silk4NET テストの実行中にビジュアル実行ログを作成できます。TrueLog ファイルは、Silk4NET テストが実行されたプロセスの作業ディレクトリに作成されます。

デフォルトの設定では、スクリプトでエラーが発生した場合にのみスクリーンショットが作成され、エラーが発生したテスト ケースのログのみが作成されます。

TrueLog の有効化

新しい Silk4NET スクリプトでは、TrueLog がデフォルトで有効になっています。Visual Studio Unit Testing Framework を使用している既存の Silk4NET スクリプトで TrueLog を有効にするには、スクリプト内のすべてのテストクラスの TestClass 属性を SilkTestClass 属性で置き換える必要があります。

TrueLog を有効にするには、以下を実行します。

1. TrueLog を有効にするテスト クラスが含まれたスクリプトを開きます。
2. テストクラスに SilkTestClass 属性を追加します。

ディレクトリの TestResults サブディレクトリに TrueLog が作成されます。このディレクトリには、Visual Studio ソリューション ファイルおよび Visual Studio Unit Testing Framework の結果が格納されています。Visual Studio ソリューション ファイルには、Silk4NET スクリプトが格納されています。Silk4NET テストの実行が完了したら、ダイアログ ボックスが開きます。**結果の検討** をクリックすると、完了したテストの TrueLog を確認できます。

使用例

Visual Basic スクリプト内のクラスに対して TrueLog を有効にするには、以下のコードを使用します。

```
<SilkTestClass()> Public Class MyTestClass
    <TestMethod()> Public Sub MyTest()
        ' my test code
    End Sub
End Class
```

C# スクリプト内のクラスに対して TrueLog を有効にするには、以下のコードを使用します。

```
[SilkTestClass]
public class MyTestClass {
    [TestMethod]
    public void MyTest() {
```

```
// my test code  
}  
}
```

TrueLog で非 ASCII 文字が正しく表示されない理由

TrueLog Explorer は MBCS ベースのアプリケーションであるため、正しく表示するには、すべての文字列が MBCS 形式でエンコードされている必要があります。TrueLog Explorer でデータを表示およびカスタマイズすると、データが表示される前に、多数の文字列変換処理が発生することがあります。

UTF-8 でエンコードされた Web サイトをテストする場合は、文字列を含むデータをアクティブな Windows システム コード ページに変換できないことがあります。このような場合、TrueLog Explorer は変換できない非 ASCII 文字列を、構成可能な置換文字 (通常は「?」) で置き換えます。


TrueLog Explorer で非 ASCII 文字列を正確に表示するには、システム コード ページに適切な言語 (日本語など) を設定します。

Team Foundation Server での Silk4NET の使用

このセクションでは、Visual Studio Team Foundation Server (TFS) を使用して Silk4NET テストを実行する方法について説明します。

新機能、サポート対象のプラットフォームとバージョン、既知の問題、および回避策の詳細については、『Silk Test リリース ノート』(<http://supportline.microfocus.com/productdoc.aspx> で入手可能) を参照してください。

Silk4NET テストの TFS での実行

 **注:** TFS または Visual Studio の機能についての説明など、このタスクのステップの詳細な情報については、それぞれの製品のドキュメントを参照してください。

Silk4NET テストを実行するために、TFS を使用することができます。

1. Visual Studio で **チーム エクスプローラー ビュー** を開き、**チーム プロジェクトに接続** をクリックして TFS に接続します。
2. **チーム エクスプローラー ビュー** で、Silk4NET プロジェクトを TFS に追加します。
3. 省略可能: さらに、Silk4NET テストを Silk4NET プロジェクトに追加します。
4. **チーム エクスプローラー ビュー** で、テストを TFS にチェックインします。
5. Silk4NET プロジェクトを含んだソリューションを右クリックし、**追加 > 新しい項目 > テストの設定 > テストの設定** をクリックして新しいテスト設定ファイルを作成します。
6. テストの実行に使用するテスト コントローラを設定します。
7. テストの設定ファイルで、**データと診断** を選択します。
8. **Silk4NET TrueLog の有効化** をチェックして、Silk4NET TrueLog データ コレクタを有効化します。
9. 省略可能: テストの設定ファイルの **ロール** では、テストの実行に使用したいテスト コントローラを構成することができます。
10. **チーム エクスプローラー ビュー** で、新しいビルド定義を作成します。
11. テスト設定ファイルをビルド定義に追加します。
12. Silk4NET テスト プロジェクト アセンブリの自動テストがビルド後に実行されるように、ビルド定義を設定します。
13. **方法: テスト エージェントを設定して、デスクトップと対話するテストを実行する** の手順に従って、Silk4NET と AUT 間の対話を有効化します。
14. **チーム エクスプローラー ビュー** で、ビルド定義を実行して Silk4NET テストを実行します。
15. 省略可能: TrueLog ファイルを分析します。

TFS で実行した Silk4NET テストの TrueLog ファイルの場所


TFS で Silk4NET テストを実行する場合、実行が完了した後に **再生完了** ダイアログ ボックスは表示されず、そのテストの TrueLog ファイルはローカル マシンに出力されません。TFS で実行した Silk4NET テストの結果を分析するために生成した TrueLog ファイルの場所は次のようになります。

1. チーム エクスプローラーで、TrueLog ファイルの場所を特定したいビルドを右クリックします。
2. テストの実行に成功しました をクリックします。 **収集されたファイル** の下に、TrueLog ファイルが見つかります。
3. TrueLog ファイルをダブルクリックしてダウンロードします。

Silk4NET TrueLog データ コレクタの有効化についての情報は、*Silk4NET* テストの *TFS* での実行 を参照してください。

スクリプト オプションの設定

記録、ブラウザ、カスタム属性、無視するクラス、同期、および再生モードに関するスクリプト オプションを指定します。

 **注:** Silk4NET のプロジェクトごとに、Silk4NET によって config.Silk4net 構成ファイルが作成されます。Silk4NET によって、テスト対象のアプリケーションの基本状態とすべてのオプションがこのファイルに保存されます。その後、オプションが再生中に使用されます。

TrueLog オプションの設定

ビットマップをキャプチャして Silk4NET の情報を記録するように TrueLog を有効化します。

ビットマップとコントロールを TrueLog に記録すると Silk4NET のパフォーマンスに悪影響が出る場合があります。ビットマップをキャプチャして情報を記録すると TrueLog ファイルが大きくなるがあるので、エラーを含むテストケースのみを記録するように、詳細情報が必要なテスト ケース用に TrueLog オプションを調整できます。


テストの結果は、TrueLog Explorer で検証できます。TrueLog Explorer の詳細については、**スタート > プログラム > Silk > Silk Test > ドキュメント** にある *Silk TrueLog Explorer ユーザー ガイド* を参照してください。

TrueLog を有効にして、TrueLog が Silk4NET 用に収集する情報をカスタマイズするには、次の手順に従います。

1. **Silk4NET** をクリックし、**オプションの編集** を選択します。 **スクリプト オプション** ダイアログ ボックスが開きます。
2. **TrueLog** タブをクリックします。
3. **スクリーンショット モード** を選択します。
デフォルトは **なし** です。
4. 任意：**遅延** を設定します。
この遅延により、ビットマップが取られる前に Windows がアプリケーション ウィンドウを描画する時間を確保できます。キャプチャされたビットマップでアプリケーションが適切に描画されない場合は、遅延時間を増やしてください。
5. **OK** をクリックします。

記録オプションの設定

記録を一時停止するためのショートカット キーの組み合わせを設定したり、絶対値による指定やマウスの移動操作が記録されるかどうかを指定したりします。

 **注:** 以下の設定はすべて任意です。テスト メソッドの品質が向上する場合に、これらの設定を変更してください。

1. **Silk4NET** をクリックし、**オプションの編集** を選択します。 **スクリプト オプション** ダイアログ ボックスが開きます。
2. **記録** タブをクリックします。
3. 記録の一時停止に使用するショートカット キーの組み合わせとして Ctrl+Shift を設定するには、**OPT_ALTERNATE_RECORD_BREAK** チェック ボックスをオンにします。
デフォルトのショートカット キーの組み合わせは、Ctrl+Alt です。

- スクロール イベントの絶対値を記録するには、**OPT_RECORD_SCROLLBAR_ABSOLUT** チェックボックスをオンにします。
 - Web アプリケーション、Win32 アプリケーション、および Windows Forms アプリケーションのマウス移動操作を記録するには、**OPT_RECORD_MOUSEMOVES** チェックボックスをオンにします。たとえば、Adobe Flex や Swing など、xBrowser テクノロジー ドメインの子テクノロジー ドメインのマウス移動操作を記録することはできません。
 - マウスの移動操作を記録する場合は、**OPT_RECORD_MOUSEMOVE_DELAY** テキスト ボックスで、MouseMove 操作を記録する前に必要なマウスの静止時間をミリ秒で指定します。デフォルト値は、200 に設定されています。
 - 概して、TextClick 操作のほうが Click 操作よりも望ましいオブジェクトで、Click 操作ではなくテキストのクリック操作を記録するには、**OPT_RECORD_TEXT_CLICK** チェックボックスをオンにします。
 - 概して、ImageClick 操作のほうが Click 操作よりも望ましいオブジェクトで、Click 操作ではなくイメージのクリック操作を記録するには、**OPT_RECORD_IMAGE_CLICK** チェックボックスをオンにします。
 - オブジェクト マップを記録するには、**OPT_RECORD_OBJECTMAPS** チェックボックスをオンにします。
- 10OK をクリックします。

ブラウザの記録オプションの設定

記録中に無視するブラウザの属性や DOM 関数の代わりに、ユーザーの入力そのものを記録するかどうかを指定します。



注: 以下の設定はすべて任意です。テスト メソッドの品質が向上する場合に、これらの設定を変更してください。

- Silk4NET** をクリックし、**オプションの編集** を選択します。 **スクリプト オプション** ダイアログ ボックスが開きます。
- ブラウザ** タブをクリックします。
- ロケーター属性名除外リスト** グリッドで、記録中に無視する属性名を入力します。たとえば、height という名前の属性を記録しない場合には、height 属性名をグリッドに追加します。複数の属性名を指定する場合にはカンマで区切ります。
- ロケーター属性値除外リスト** グリッドで、記録中に無視する属性値を入力します。たとえば、x-auto という値を持つ属性を記録しない場合には、x-auto をグリッドに追加します。複数の属性値を指定する場合にはカンマで区切ります。
- DOM 関数の代わりにユーザーの入力そのものを記録するには、**OPT_XBROWSER_RECORD_LOWLEVEL** チェックボックスをオンにします。たとえば、DomClick の代わりに Click、SetText の代わりに TypeKeys を記録するには、このチェックボックスをオンにします。
アプリケーションでプラグインまたは AJAX を使用している場合は、ユーザーの入力そのものを使用します。アプリケーションでプラグインまたは AJAX を使用していない場合は、再生中にブラウザにフォーカスを設定したりブラウザをアクティブにしたりする必要がない高レベル DOM 関数を使用することをお勧めします。テストで DOM 関数を使用すると、より高速になり、信頼性も高まります。
- ロケーター属性値の最大長を設定するには、**属性値の最大の長さ** セクションのフィールドに長さを入力します。
実際の長さがこの制限を超えると、値は切り捨てられ、ワイルドカード (*) が付加されます。デフォルト値は、20 文字に設定されています。
- OK** をクリックします。

カスタム属性の設定

Silk4NET には、ロケーターが記録時に一意となり、メンテナンスが容易になるようにする、高度なロケーター生成メカニズムが備えられています。使用するアプリケーションやフレームワークに応じて、最適な結果を得るためにデフォルト設定を変更できます。それぞれのテクノロジーで使用できる任意のプロパティ (整数や倍精度の数値、文字列、項目識別子、列挙値) を、カスタム属性として使用できます。

頻繁には変更されない属性を利用して、適切に定義されたロケーターでは、メンテナンス作業が少なく抑えられます。カスタム属性を使用すると、caption や index などの他の属性を使用するよりも高い信頼性を得ることができます。これは、caption はアプリケーションを他の言語に翻訳した場合に変更され、index は他のオブジェクトが追加されると変更される可能性があるためです。

カスタム属性 タブのリスト ボックスに一覧表示されているテクノロジー ドメインの場合、任意のプロパティ (myCustomProperty を定義する WPFButton など) を取得し、それらのプロパティをカスタム属性として使用することもできます。最適な結果を得るために、テストで利用する要素にカスタム オートメーション ID を追加します。Web アプリケーションでは、操作する要素に <div myAutomationId= "my unique element name" /> などの属性を追加できます。また、Java SWT では、GUI を実装する開発者が属性 (testAutomationId など) をウィジェットに対して定義することによって、アプリケーション内でそのウィジェットを一意に識別できます。テスト担当者は、その属性をカスタム属性 (この場合は testAutomationId) のリストに追加し、その一意の ID によってコントロールを識別できます。この手法によって、ロケーターの変更に伴うメンテナンス作業を回避することができます。

caption のように、複数のオブジェクトで同じ属性値が共有されている場合、Silk4NET は、複数の利用可能な属性を "and" 操作で結合してロケーターを一意にするよう試み、一致したオブジェクトのリストを単一のオブジェクトになるまで絞り込んでいきます。それができなくなった場合には、索引を付加します。つまり、ロケーターは caption が xyz である *n* 番目のオブジェクトを探すことを意味します。

複数のオブジェクトに同じカスタム属性の値が割り当てられた場合は、そのカスタム属性を呼び出したときにその値を持つすべてのオブジェクトが返されます。たとえば、一意の ID として loginName を 2 つの異なるテキスト フィールドに割り当てた場合は、loginName 属性を呼び出したときに、両方のフィールドが返されます。

1. **Silk4NET** をクリックし、**オプションの編集** を選択します。 **スクリプト オプション** ダイアログ ボックスが開きます。
2. **カスタム属性** タブを選択します。
3. **テクノロジー ドメインを選択します** リスト ボックスから、テストするアプリケーションのテクノロジー ドメインを選択します。



注: Flex または Windows API ベースのクライアント/サーバー (Win32) アプリケーションには、カスタム属性を設定できません。

4. 使用する属性をリストに追加します。

カスタム属性が利用可能な場合は、ロケーター生成プログラムは、他の属性の前にそれらの属性を使用します。リストの順番は、ロケーター生成プログラムが使用する属性の優先順位を表しています。指定した属性が選択したオブジェクトに対して利用可能ではなかった場合には、Silk4NET はテストしているアプリケーションのデフォルトの属性を使用します。

複数の属性名を指定する場合にはカンマで区切ります。



注: Web アプリケーションにカスタム属性を含めるためには、HTML タグとして追加します。たとえば、bcauid という属性を追加するには、<input type='button' bcauid='abc' value='click me' /> と入力します。



注: Java SWT コントロールにカスタム属性を含めるには、org.swt.widgets.Widget.setData(String key, Object value) メソッドを使用します。



注: Swing コントロールにカスタム属性を含めるには、`SetClientProperty("propertyName", "propertyValue")` メソッドを使用します。

5. **OK** をクリックします。

無視するクラスの設定

記録や再生中に無視したいクラスの名前を指定します。

1. **Silk4NET** をクリックし、**オプションの編集** を選択します。 **スクリプト オプション** ダイアログ ボックスが開きます。
2. **無視するクラス** タブをクリックします。
3. **無視するクラス** グリッドで、記録や再生中に無視するクラスの名前を入力します。複数のクラス名を指定する場合にはカンマで区切ります。
4. **OK** をクリックします。

記録/再生の対象とする WPF クラスの設定

記録や再生の対象にしたい WPF クラスの名前を指定します。たとえば、*MyGrid* というカスタム クラスが WPF Grid クラスから継承された場合、*MyGrid* カスタム クラスのオブジェクトは記録や再生に使用できません。Grid クラスはレイアウト目的のためにのみ存在し、機能テストとは無関係であるため、Grid オブジェクトは記録や再生に使用できません。この結果、Grid オブジェクトはデフォルトでは公開されません。機能テストに無関係なクラスに基づいたカスタム クラスを使用するには、カスタム クラス (この場合は *MyGrid*) を **OPT_WPF_CUSTOM_CLASSES** オプションに追加します。これによって、記録、再生、検索、プロパティの検証など、すべてのサポートされる操作を指定したクラスに対して実行できるようになります。

1. **Silk4NET** をクリックし、**オプションの編集** を選択します。 **スクリプト オプション** ダイアログ ボックスが開きます。
2. **WPF** タブをクリックします。
3. **カスタム WPF クラス名** グリッドで、記録や再生中に公開するクラスの名前を入力します。複数のクラス名を指定する場合にはカンマで区切ります。
4. **OK** をクリックします。

同期オプションの設定

Web アプリケーションの同期およびタイムアウトの値を指定します。



注: 以下の設定はすべて任意です。テスト メソッドの品質が向上する場合に、これらの設定を変更してください。

1. **Silk4NET** をクリックし、**オプションの編集** を選択します。 **スクリプト オプション** ダイアログ ボックスが開きます。
2. **同期** タブを選択します。
3. Web アプリケーションを準備完了状態にするための同期アルゴリズムを指定するには、**OPT_XBROWSER_SYNC_MODE** リスト ボックスからオプションを選択します。同期アルゴリズムは、呼び出しが可能になる状態までの待機時間を設定します。デフォルト値は、**AJAX** に設定されています。
4. **同期除外リスト** テキスト ボックスに、除外するサービスまたは Web ページの URL 全体あるいは URL の一部を入力します。
AJAX フレームワークやブラウザによっては、サーバーから非同期にデータを取得するために、特殊な HTTP 要求を継続して出し続けるものがあります。これらの要求により、指定した同期タイムアウトの

期限が切れるまで同期がハングすることがあります。この状態を回避するには、HTML 同期モードを使用するか、問題が発生する要求の URL を **同期除外リスト** 設定で指定します。

たとえば、クライアントからデータをポーリングすることによってサーバー時間を表示するウィジェットを Web アプリケーションで使用する場合は、このウィジェットのトラフィックが永続的にサーバーに送信されます。このサービスを同期から除外するには、サービス URL を判別し、除外リストに入力します。

たとえば、以下のように入力します。

- http://example.com/syncsample/timeService
- timeService
- UICallBackServiceHandler

複数のエントリをカンマで区切って指定します。



注: アプリケーションで 1 つのサービスのみが使用されている場合、そのサービスでテストを無効にするには、サービス URL を除外リストに追加するのではなく、HTML 同期モードを使用する必要があります。

5. オブジェクトが準備完了状態になるまでの最大待機時間を指定するには、**OPT_SYNC_TIMEOUT** テキストボックスにミリ秒で値を指定します。
デフォルト値は、**300000** に設定されています。
6. 再生中にオブジェクトが解決されるまでの最大待機時間を指定するには、**OPT_WAIT_RESOLVE_OBJDEF** テキストボックスにミリ秒で値を入力します。
デフォルト値は、**5000** に設定されています。
7. エージェントがオブジェクトの解決を再試行するまでの最大待機時間を指定するには、**OPT_WAIT_RESOLVE_OBJDEF_RETRY** テキストボックスにミリ秒で値を入力します。
デフォルト値は、**500** に設定されています。
8. **OK** をクリックします。

再生オプションの設定

テストするオブジェクトがアクティブであることを確実にしたいかどうかや、デフォルトの再生モードを上書きしたいかどうかを指定します。再生モードは、コントロールがマウスやキーボードによって再生されるか、API で再生されるかを定義します。デフォルトモードを使用すると、最も信頼できる結果が得られます。他のモードを選択した場合は、すべてのコントロールが選択したモードを使用します。

1. **Silk4NET** をクリックし、**オプションの編集** を選択します。 **スクリプト オプション** ダイアログボックスが開きます。
2. **再生** タブを選択します。 **再生オプション** ページが表示されます。
3. **OPT_REPLAY_MODE** リストボックスから、以下のいずれかのオプションを選択します。
 - **デフォルト** : このモードを使用すると、最も信頼できる結果が得られます。デフォルトでは、各コントロールそれぞれが、マウスやキーボード (低レベル)、あるいは API (高レベル) モードのどちらかを使用します。デフォルトモードを使用すると、各コントロールがコントロールの種類に応じて適切なモードが使用されます。
 - **高レベル** : このモードを使用すると、API を使用して各コントロールが再生されます。
 - **低レベル** : このモードを使用すると、マウスやキーボードを使用して各コントロールが再生されません。
4. テストするオブジェクトがアクティブであることを確実にするには、**OPT_ENSURE_ACTIVE_OBJDEF** チェックボックスをオンにします。
5. 再生中にオブジェクトが有効になるまでの待機時間を変更するには、**オブジェクト有効化タイムアウト** セクションのフィールドに新しい時間を入力します。
この時間は、ミリ秒単位で指定されます。デフォルト値は 1000 です。

6. **OK** をクリックします。

詳細オプションの設定

Windows ユーザー補助を有効にするかどうか、テキストのキャプチャ中にウィンドウからフォーカスを外すかどうか、およびロケータ属性名で大文字小文字が区別されるかどうかを指定します。

1. **Silk4NET** をクリックし、**オプションの編集** を選択します。 **スクリプト オプション** ダイアログ ボックスが開きます。
2. **詳細設定** タブをクリックします。 **詳細オプション** ページが表示されます。
3. 標準の Win32 コントロールの解決に加えて、Microsoft ユーザー補助を有効にするには、**OPT_ENABLE_ACCESSIBILITY** チェック ボックスをオンにします。
4. テキストをキャプチャする前にウィンドウからフォーカスを外すには、**OPT_REMOVE_FOCUS_ON_CAPTURE_TEXT** チェック ボックスをオンにします。
テキストのキャプチャは、次のメソッドによる記録および再生中に実行されます。
 - TextClick
 - TextCapture
 - TextExists
 - TextRect
5. ロケータ属性名で大文字と小文字が区別されるように設定するには、**OPT_LOCATOR_ATTRIBUTES_CASE_SENSITIVE** チェック ボックスをオンにします。
6. **OPT_IMAGE_ASSET_DEFAULT_ACCURACY** リスト ボックスから、1 (低精度) から 10 (高精度) までの値を選択し、新しいイメージ資産のデフォルト精度レベルを設定します。
7. **OPT_IMAGE_VERIFICATION_DEFAULT_ACCURACY** リスト ボックスから、1 (低精度) から 10 (高精度) までの値を選択し、新しいイメージ検証資産のデフォルト精度レベルを設定します。
8. **OK** をクリックします。

Silk4NET サンプル テスト

Silk4NET サンプル テストは、Visual Studio ソリューションにパッケージ化されており、開いて表示したり、Silk Test サンプル アプリケーションに対して実行したりできます。

サンプル プロジェクトを開くには、*Silk4NET* スタート ページ にある **サンプル プロジェクトを開く** をクリックするか、Visual Studio で **ファイル > プロジェクトを開く** をクリックして、¥Users¥Public ¥Documents¥SilkTest¥samples¥Silk4NET を参照します。使用する Visual Studio バージョンに対応するフォルダを選択し、サンプルのソリューション ファイルを選択します。開く をクリックします。

インストールした Silk4NET サンプル アプリケーションに加え、一連の Silk4NET サンプル テストには、Web ベースの以下の Silk Test サンプル アプリケーション用のテストがいくつか含まれています。

Insurance Co. Web サイト	http://demo.borland.com/InsuranceWebExtJS/
Green Mountain Outpost Web	http://demo.borland.com/gmopost/

オブジェクト解決

Silk4NET では、識別されたオブジェクトのリテラル参照は **ロケーター** と呼ばれます。 Silk4NET では、テスト対象アプリケーション (AUT) のオブジェクトを検索して識別するためにロケーターを使用します。 ロケーターは、W3C (World Wide Web Consortium) によって定義された共通の XML ベース言語である XPath クエリ言語のサブセットを使用します。

ロケーターの基本概念

Silk4NET は、XPath クエリ言語のサブセットをサポートしています。

オブジェクトタイプと検索範囲

典型的なロケーターには、検索するオブジェクトのタイプと検索範囲が含まれます。 検索範囲は以下のいずれかです。

- //
- /

ロケーターは、ロケーターを指定する対象となるオブジェクトである、現在のオブジェクトに依存します。現在のオブジェクトは、アプリケーション UI のオブジェクト階層における位置を特定します。ファイルシステムと同じように、すべてのロケーターは、この階層における現在のオブジェクトの位置に依存します。

XPath 式は、現在のコンテキスト、つまり、Find メソッドを呼び出したオブジェクトの階層上における位置に依存します。ファイルシステムと同じように、すべての XPath 式は、この位置に依存します。

注:

HTML 要素に対するロケーターにおけるオブジェクトタイプは、HTML タグ名または、このオブジェクトに対して Silk4NET が使用するクラス名のいずれかになります。たとえば、ロケーター //a と //DomLink (ここで、DomLink は Silk4NET でのハイパーリンクに対する名前です) は同じです。HTML ベースでないテクノロジーの場合は、Silk4NET クラス名だけが使用されます。

例

- //a は、現在のオブジェクトに相対的なすべての階層にあるハイパーリンク オブジェクトを識別します。
- /a は、現在のオブジェクトの直下の子であるハイパーリンク オブジェクトを識別します。



注: <a> は、Web ページのハイパーリンクを表す HTML タグです。

例

以下のコード例は、ブラウザ内の最初のハイパーリンクを識別します。この例では、実行中のブラウザ インスタンスを参照するスクリプトに `browserWindow` という名前の変数が存在することを仮定しています。ここで、タイプは "a" で、現在のオブジェクトは `browserWindow` です。

VB

```
Dim link As DomLink = browserWindow.DomLink("//a")
```

```
C#
DomLink link = browserWindow.DomLink("//a");
```

属性を使用したオブジェクトの識別

オブジェクトのプロパティに基づいてオブジェクトを識別するために、ロケータ属性を使用することができます。ロケータ属性は、オブジェクトタイプ後に角かっこを使って指定します。

例

以下の例では、textContents 属性を使用して、テキスト *Home* を持つハイパーリンクを識別します。同じテキストを持つハイパーリンクが複数存在した場合は、ロケータは最初のオブジェクトを識別します。

VB

```
Dim link as DomLink = browserWindow.DomLink("//a[@textContents='Home']")
```

C#

```
DomLink link = browserWindow.DomLink("//a[@textContents='Home']");
```

ロケータ構文

Silk4NET は、UI コントロールを検索するために XPath クエリ言語のサブセットをサポートしています。

以下の表には、Silk4NET がサポートする構成子が一覧されています。



注: <a> は、Web ページのハイパーリンクを表す HTML タグです。

サポートするロケータ構成子	サンプル	説明
//	//a	現在のオブジェクトの子孫であるオブジェクトを識別します。 サンプルは、Web ページのハイパーリンクを識別します。
/	/a	現在のオブジェクトの直下の子であるオブジェクトを識別します。下位の階層レベルにあるオブジェクトは認識されません。 サンプルは、現在のオブジェクトの直下の子である Web ページのハイパーリンクを識別します。
属性	//a[@textContents='Home']	属性を指定してオブジェクトを識別します。 サンプルは、テキスト <i>Home</i> を持つハイパーリンクを識別します。
索引	サンプル 1: //a[3] サンプル 2: //a[@textContents='Home'][2]	複数のオブジェクトが検出された場合にオブジェクトの出現番号を指定

サポートするロケータ構成子	サンプル	説明
		して識別します。ロケータ内での索引は 1 から始まります。 サンプル 1 は 3 番目のハイパーリンクを、サンプル 2 はテキスト <i>Home</i> を持つ 2 番目のハイパーリンクを識別します。
論理演算子 : and、or、not、=、!=	<p>サンプル 1: // a[@textContents='Remove' or @textContents='Delete']</p> <p>サンプル 2: //a[@textContents!='Remove']</p> <p>サンプル 3: // a[not(@textContents='Delete' or @id='lnkDelete') and @href='*/delete']</p>	<p>論理演算子を使用して属性を組み合わせさせてオブジェクトを識別します。</p> <p>サンプル 1 はキャプション <i>Remove</i> または <i>Delete</i> のどちらかを持つハイパーリンクを識別し、サンプル 2 は <i>Remove</i> でないテキストを持つハイパーリンクを識別し、サンプル 3 はさまざまな論理演算子を組み合わせる方法を示しています。</p>
..	<p>サンプル 1: // a[@textContents='Edit']/..</p> <p>サンプル 2: // a[@textContents='Edit']/../a[@textContents='Delete']</p>	<p>オブジェクトの親を識別します。</p> <p>サンプル 1 はテキスト <i>Edit</i> を持つハイパーリンクの親を識別し、サンプル 2 はテキスト <i>Edit</i> を持つハイパーリンクと同列にあるテキスト <i>Delete</i> を持つハイパーリンクを識別します。</p>
*	<p>サンプル 1: // *[@textContents='Home']</p> <p>サンプル 2: /*/a</p>	<p>ハイパーリンク、テキストフィールド、またはボタンのような型を考慮せずにオブジェクトを識別します。</p> <p>サンプル 1 は型とは無関係に指定したテキスト コンテンツを持つオブジェクトを識別し、サンプル 2 は現在のオブジェクトの第 2 下位レベルにあるハイパーリンクを識別します。</p>

以下の表には、Silk4NET がサポートしていないロケータ構成子が一覧されています。

サポートしないロケータ構成子	サンプル
右辺、左辺ともに属性を指定して比較する。	//a[@textContents = @id]
属性名を右辺に指定することはサポートされません。属性名は左辺に指定する必要があります。	//a['abc' = @id]
複数のロケータを and あるいは or で結合する。	//a[@id = 'abc'] or ../Checkbox
複数の属性をかぎ括弧で指定する。	//a[@id = 'abc'][@textContents = '123'] (代わりに、//a [@id = 'abc' and @textContents = '123'] を使用してください)
複数の索引をかぎ括弧で指定する。	//a[1][2]
クラスあるいはクラス名の一部にワイルドカードを含むクラス・ワイルドカードを明示的に指定しない構成子。	//[@id = 'abc'] (代わりに、//*[@id = 'abc'] を使用してください)

サポートしないロケータ構成子	サンプル
	"//*/a[@id='abc']"

ロケータの使用

Silk4NET では、識別されたオブジェクトのリテラル参照はロケータと呼ばれます。必要に応じて、ロケータ文字列の短縮形をスクリプトで使用できます。スクリプトを再生すると、Silk4NET によって自動的に構文が展開されて完全なロケータ文字列が使用されます。スクリプトを手動で記述する場合、以下の順番で一部を省略できます。

- 検索範囲「//」。
- オブジェクトタイプの名前。Silk4NET のデフォルトはクラス名です。
- 属性を囲む角かっこ「[]」。

スクリプトを手動で記述する場合は、使用可能な最も短い形式を使用することをお勧めします。



注: オブジェクトを識別する場合は、完全なロケータ文字列がデフォルトでキャプチャされます。

以下のロケータは同じです。

- 最初の例では、完全なロケータ文字列が使用されています。

VB

```
_desktop.DomLink("//BrowserApplication//BrowserWindow//a[@textContents='Home']").Select()
```

C#

```
_desktop.DomLink("//BrowserApplication//BrowserWindow//a[@textContents='Home']").Select();
```

完全なロケータ文字列を確認するには、**Locator Spy** ダイアログ ボックスを使用します。

- 2 番目の例は、ブラウザ ウィンドウがすでに存在する場合に動作します。

VB

```
browserWindow.DomLink("//a[@textContents='Home']").Select()
```

C#

```
browserWindow.DomLink("//a[@textContents='Home']").Select();
```

または、次のように省略して記述できます。

VB

```
browserWindow.DomLink("@textContents='Home']").Select()
```

C#

```
browserWindow.DomLink("@textContents='Home']").Select();
```

識別のための実際の属性がないオブジェクトを検索するには、インデックスを使用します。たとえば、Web ページ上の 2 番目のハイパーリンクを選択するには、以下のように入力します。

VB

```
browserWindow.DomLink("[2]").Select()
```

C#

```
browserWindow.DomLink("[2]").Select();
```

さらに、その種類の最初のオブジェクトを検索する（このことは、オブジェクトに実際の属性がない場合に便利です）には、以下のように入力します。

VB


```
browserWindow.DomLink().Select()
```

C#

```
browserWindow.DomLink().Select();
```

Find メソッドの使用

.DomLink のようなメソッドを使用する代わりに、Find メソッドを使用してロケータを持つ単一のオブジェクトを識別することができます。

 **注:** Find メソッドは、完全ロケータのみ使用できます。省略形のロケータはサポートされません。

以下のように入力する代わりに：

VB

```
_desktop.DomLink("//a[@textContents='Home']").Select()
```

C#

```
_desktop.DomLink("//a[@textContents='Home']").Select();
```

以下のように入力することができます。

VB

```
_desktop.Find(Of DomLink)("//a[@textContents='Home']").Select()
```

C#

```
_desktop.Find<DomLink>("//a[@textContents='Home']").Select();
```

.DomLink メソッドは、Find メソッドで内部的に使用されます。 .DomLink メソッドを使用した方が、Find メソッドよりも簡潔に表現できるため、好まれます。

ロケータを使用したオブジェクトの存在確認

テスト対象アプリケーションにオブジェクトが存在するかどうかを確認するために、Exists メソッドを使用することができます。

以下のコードでは、テキスト *Log out* を持つハイパーリンクが Web ページに存在するかどうかを確認します。

VB

```
If (browserWindow.Exists( "//a[@textContents='Log out']" )) Then  
    ' do something  
End If
```

C#

```
if (browserWindow.Exists( "//a[@textContents='Log out']" )){  
    // do something  
}
```

1 つのロケータによる複数オブジェクトの識別

ロケータに一致する最初のオブジェクトを識別するだけでなく、そのロケータに一致したすべてのオブジェクトを識別したい場合には、FindAll メソッドを使用することができます。

例

以下のコード例では、Web ページのすべてのハイパーリンクを取得するために、FindAll メソッドを使用しています。

VB

```
Dim links As IList(Of DomLink) = browserWindow.FindAll(Of DomLink)("/a")
```

C#

```
IList<DomLink> links = browserWindow.FindAll<DomLink>("/a");
```

XPath のパフォーマンス問題のトラブルシューティング

複雑なオブジェクト構造を持つアプリケーションをテストする場合、パフォーマンスの問題やスクリプトの信頼性に関する問題が発生する場合があります。このトピックでは、記録中に Silk4NET が自動的に生成したロケータとは異なるロケータを使用することによって、スクリプトのパフォーマンスを改善させる方法について説明します。



注: 一般に、複雑なロケータを使用することは推奨しません。複雑なロケータを使用すると、テストの信頼性を損なう可能性があります。複雑なロケータは、テストアプリケーションの構造をほんの少し変更しただけで機能しなくなってしまう可能性があります。それにもかかわらず、スクリプトのパフォーマンスが要求を満たしていない場合には、より固有のロケータを使用することによってテストのパフォーマンスを向上できる可能性があります。

例として、MyApplication アプリケーションの要素ツリーを以下に示します。

```
Root
  Node id=1
    Leaf id=2
    Leaf id=3
    Leaf id=4
    Leaf id=5
  Node id=6
    Node id=7
      Leaf id=8
      Leaf id=9
    Node id=9
      Leaf id=10
```

以下の最適化手法のいくつかを使用して、スクリプトのパフォーマンスを改善させることができます。

- 複雑なオブジェクト構造内の要素を特定したい場合は、オブジェクト構造全体ではなく、その特定の部分だけを検索するようにします。たとえば、サンプルツリーの識別子 4 を持つ要素を検索する場合に `Root.Find("//Leaf[@id='4']")` というクエリーを使用している場合、`Root.Find("/Node[@id='1']/Leaf[@id='4']")` というクエリーで置き換えます。最初のクエリーでは、識別子 4 を持つリーフが、アプリケーションの要素ツリー全体から検索されます。最初のリーフが見つかった時点で返されます。2 番目のクエリーでは、識別子 1 を持つノードと識別子 6 を持つノードがある最初のレベルのノードがまず検索された後、識別子 4 を持つすべてのリーフが識別子 1 を持つノードのサブツリー内から検索されます。
- 同じ階層内の複数の項目を特定したい場合は、まずは階層を特定してからループ内で項目を特定します。`Root.FindAll("/Node[@id='1']/Leaf")` というクエリーを使用している場合、次のようなループで置き換えます。

```
Public Sub Main()
  Dim node As TestObject

  node = _desktop.Find("/Node[@id='1']")
```

```
For i As Integer = 1 To 4 Step 1
    node.Find("/Leaf[@id='"+i+"']")
Next
End Sub
```

Locator Spy

Locator Spy を使用すると、GUI オブジェクトのキャプションや XPath ロケータ文字列を識別できます。そして、関係する XPath ロケータ文字列や属性を、スクリプト内のメソッドにコピーできます。また、テストスクリプトで XPath ロケータ文字列の属性を手動で編集し、変更を **Locator Spy** で検証することができます。**Locator Spy** を使用することで、XPath クエリー文字列が正しいことが保障されます。



注: Locator Spy のロケータ属性テーブルには、ロケータで使用できるすべての属性が表示されます。Web アプリケーションの場合は、記録中に無視するように定義したすべての属性もテーブルに含まれます。

オブジェクト マップ

オブジェクト マップはテスト資産の一種であり、コントロールまたはウィンドウのロケータではなく、コントロールまたはウィンドウに論理名 (エイリアス) を関連付ける項目が含まれています。コントロールがオブジェクト マップ資産に登録されると、スクリプトでのそのコントロールに対する参照はすべて、実際のロケータ名ではなく、そのエイリアスによって行われます。

複数のテストで 1 つのオブジェクト マップ項目の定義を参照できるため、ユーザーがそのオブジェクト マップ定義を 1 回更新すると、オブジェクト マップ定義を参照するすべてのテストでそのオブジェクト マップ定義が Silk4NET によって更新されます。

例

以下の構成では、ロケータが使用されている `BrowserWindow` の定義が示されています。

```
__desktop.BrowserApplication("cnn_com").BrowserWindow("//  
BrowserWindow[1]")
```

オブジェクト マップ資産の名前は `cnn_com` です。オブジェクト マップのエイリアスによって置き換えることができるロケータは、以下のとおりです。

```
"//BrowserWindow[1]"
```

`BrowserWindow` のオブジェクト マップ エントリは `BrowserWindow` です。

結果的に、スクリプト内の `BrowserWindow` の定義は以下ようになります。

```
__desktop.BrowserApplication("cnn_com").BrowserWindow("BrowserWindow")
```

ロケータのインデックスが変更された場合、テスト スクリプトのロケータのすべての外観を変更する必要はなく、オブジェクト マップのエイリアスを変更するだけで済みます。Silk4NET によって、オブジェクト マップ定義を参照するすべてのテストが更新されます。

オブジェクト マップを使用する利点

オブジェクト マップには、以下の利点があります。

- 複雑なロケータ名がわかりやすい名前でも置き換えられるため、スクリプトが読みやすくなる。
- テスト アプリケーションが変更された場合に変わる可能性のあるロケータに依存しなくなる。
- オブジェクト マップ項目のロケータに加えられた変更を、対応するオブジェクト マップ項目を含むすべてのテストに適用することによって、テストのメンテナンスが簡単になる。

オブジェクト マップのオン/オフの切り替え


記録時に Silk4NET でロケータ名またはオブジェクト マップのエイリアスのいずれを使用するかを設定できます。

記録中にオブジェクト マップからエイリアスを使用するには、以下を実行します。

1. **Silk4NET > スクリプト オプション** をクリックします。
2. **記録** をクリックします。
3. **オブジェクト マップを記録する** をオンにします。

デフォルトで、Silk4NET は記録中にオブジェクト マップからのエイリアスを記録します。 **オブジェクト マップを記録する** 設定を オフ にすると、Silk4NET は記録中にロケーター名を記録します。必要に応じて、 **オブジェクト マップを記録する** 設定のオン/オフを切り替えることができます。ただし、ロケーターを使用してテストを記録した場合、オブジェクト マップ項目を使用するには、テストを記録し直す必要があります。

4. ロケーターの記録中にオブジェクト マップをマージする際に、Silk4NET で要素の追加の属性を使用したい場合、設定 **オブジェクト マップにロケーターのスマート マージを適用する** をチェックします。設定 **オブジェクト マップにロケーターのスマート マージを適用する** を無効にした場合は、Silk4NET は XPath だけを使用してマージします。ロケーターを既存のオブジェクト マップ エントリにマッピングする際に追加の属性を使用すると、記録したスクリプトのオブジェクト マップ ID の使用があいまいになる場合、設定 **オブジェクト マップにロケーターのスマート マージを適用する** に対して **いいえ** を選択します。


 **注: オブジェクト マップを記録する** 設定を有効にすると、Silk4NET 全体にわたって、ロケーター名の代わりにオブジェクト マップの項目名が表示されます。たとえば、**プロパティ ペイン**で **アプリケーション構成** カテゴリを表示する場合、ロケーター名ではなくオブジェクト マップ項目名が **ロケーター** ボックスに表示されます。

複数のプロジェクトでの資産の使用

Silk4NET では、イメージ資産、イメージ検証、およびオブジェクト マップが資産と呼ばれます。資産が配置されているプロジェクトのスコープ外でそれらの資産を使用する場合、資産を使用するプロジェクトから、資産を配置するプロジェクトに、プロジェクトの直接的な参照を追加する必要があります。

再生中に資産が使用されると、Silk4NET は、最初に現在のプロジェクト内でその資産を検索します。Silk4NET で現在のプロジェクト内に資産が検出されなかった場合、Silk4NET は現在のプロジェクトがプロジェクト参照を持つプロジェクトを追加検索します。それでも資産が見つからない場合、Silk4NET はエラーをスローします。

Silk4NET は、検出されたすべての資産の順番を保持します。

 **注:** 別のプロジェクトに依存関係として追加したプロジェクトのコードが、依存関係のあるプロジェクトのコードで参照されていない場合、ユーザーが依存関係のあるプロジェクトを作成すると、Visual Studio はそのプロジェクト依存関係を削除します。プロジェクト依存関係に置かれている資産を使用するには、依存関係のあるプロジェクトから、資産が置かれるプロジェクトのメンバーに、コード参照を追加する必要があります。このようなコード参照を追加することで、依存関係のあるプロジェクトを作成しているときに、Visual Studio によってプロジェクト依存関係が削除されないようにできます。たとえば、クラスまたは定数をプロジェクト依存関係に追加した後、依存関係のあるプロジェクトのコードでそのクラスまたは定数を呼び出すことができます。

複数のプロジェクトに同じ名前の資産が存在する場合に、現在のプロジェクトに含まれている資産を使用しないときは、資産を使用するメソッドで使用する特定の資産を定義できます。使用する資産を定義するには、メソッドを呼び出すときに、アセンブリ名を接頭辞として資産名に追加します。アセンブリ名は、デフォルトでプロジェクト名に設定されます。

例: プロジェクトの参照の追加

プロジェクト *ProjectA* にコード

```
'VB code  
ImageClick("imageAsset")
```

を呼び出すテストが含まれており、イメージ資産 *imageAsset* がプロジェクト *ProjectB* に置かれている場合、プロジェクトの直接的な参照を *ProjectA* から *ProjectB* に追加する必要があります。

例：特定の資産の呼び出し

ProjectA と *ProjectB* の両方に *anotherImageAsset* という名前のイメージ資産が含まれている場合に、*ProjectB* からイメージ資産を明示的にクリックする場合、次のコードを使用します：

```
'VB code  
ImageClick("ProjectB:anotherImageAsset")
```

Web アプリケーションでのオブジェクト マップの使用

デフォルトで、Web アプリケーションに対する操作を記録すると、Silk4NET は共通プロジェクトの記録中に、ネイティブ ブラウザのコントロール用に *WebBrowser* という名前のオブジェクト マップを作成し、各 Web ドメイン用にオブジェクト マップ資産を作成します。

印刷または設定用のメイン ウィンドウやダイアログ ボックスなど、Web ドメインに特有ではない共通のブラウザ コントロールの場合、*WebBrowser* という名前を使用して、現在のプロジェクトに追加のオブジェクト マップが生成されます。

オブジェクト マップで、オブジェクト マップのエントリのグループ化に使用される URL パターンを編集できます。パターンを編集すると、Silk4NET はそのパターンの構文検証を行います。パターンには、ワイルドカード * および ? を使用できます。

例

<http://www.borland.com> および <http://www.microfocus.com> で何らかの操作を記録した後、プリンタ ダイアログを開くと、次の 3 つの新しいオブジェクト マップ資産が **アセットブラウザ** に追加されます。

- WebBrowser
- borland_com
- microfocus_com



注: Silk4NET では、オブジェクト マップのないプロジェクトに対してのみ、新しいオブジェクト マップ資産が生成されます。バージョン 14.0 よりも前のバージョンの Silk4NET を使用して生成されたオブジェクト マップをすでに含む Silk4NET の Web アプリケーションに対する操作を記録すると、追加で記録されたエントリは既存のオブジェクト マップに保存されます。Web ドメインに対して追加のオブジェクト マップ資産が生成されることはありません。

オブジェクト マップ項目名の変更

オブジェクト マップでは、項目とロケータの名前を手動で変更できます。



警告: オブジェクト マップ項目の名前を変更すると、その項目を使用するすべてのスクリプトが影響を受けます。たとえば、**キャンセル** ボタンのオブジェクト マップ項目の名前を **CancelMe** から **Cancel** に変更すると、**CancelMe** を使用するすべてのスクリプトを、**Cancel** を使用するよう手動で変更する必要があります。

オブジェクト マップ項目は一意である必要があります。重複するオブジェクト マップ項目を追加しようとすると、オブジェクト マップ項目は一意である必要があることが Silk4NET から通知されます。


無効な文字またはロケータを使用すると、項目名またはロケータ テキストが赤で表示され、ツール ヒントにエラーの説明が表示されます。オブジェクト マップ項目として無効な文字には、¥、/、<、>、":、*、?、|、=、..、@、[,] があります。無効なロケータ パスは、空または不完全なロケータ パスです。

1. **ソリューション エクスプローラ** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 次のいずれか 1 つを選んでください：
 - 名前を変更するオブジェクト マップ項目を含むオブジェクト マップをダブルクリックします。
 - 名前を変更するオブジェクト マップ項目を含むオブジェクト マップを右クリックし、**開く** を選択します。

オブジェクト マップ項目および各項目に関連付けられたロケータの階層が、オブジェクト マップに表示されます。

3. 名前を変更するオブジェクト マップ項目に移動します。
たとえば、名前を変更する項目を検索するには、ノードの展開が必要な場合があります。
4. 名前を変更するオブジェクトをクリックしてから、オブジェクトを再度クリックします。
5. 使用する項目名を入力し、Enter を押します。
無効な文字を使用すると、項目名が赤で表示されます。
新しい名前が **項目名** リストに表示されます。
6. **CTRL+S** を押して、変更を保存します。

変更した項目名を既存のスクリプトで使用する場合は、新しい項目名を使用するようにスクリプトを手動で変更する必要があります。

 **注:** オブジェクト マップ ツリーに含まれるすべてのノードのすべての子ノードは、オブジェクト マップを保存するときにアルファベット順にソートされます。

オブジェクト マップのロケータの変更

スクリプトを記録するときに、ロケータは自動的にオブジェクト マップ項目に関連付けられます。ただし、より汎用的にするために、ロケータ パスを変更できます。たとえば、テスト アプリケーションで特定のコントロールに自動的に日付または時刻が割り当てられる場合、ワイルドカードを使用するようにそのコントロールのロケータを変更できます。ワイルドカードを使用すると、それぞれのテストで異なる日付または時刻が挿入される場合でも、各テストで同じロケータを使用できます。

1. **ソリューション エクスプローラ** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
 2. 次のいずれか 1 つを選んでください：
 - 変更するロケータを含むオブジェクト マップをダブルクリックします。
 - 変更するロケータを含むオブジェクト マップを右クリックし、**開く** を選択します。
- オブジェクト マップ項目および各項目に関連付けられたロケータの階層が、オブジェクト マップに表示されます。
3. 変更するロケータに移動します。
たとえば、変更するロケータを検索するには、ノードの展開が必要な場合があります。
 4. 変更するロケータ パスをクリックしてから、ロケータ パスを再度クリックします。
 5. 有効なロケータ パスがある場合は、使用する項目名とロケータ パスを入力して Enter を押すことができます。有効なロケータ パスを判別するには、以下のステップで説明するように、**Locator Spy** ダイアログ ボックスを使用します。
 - a) **Silk4NET > ロケータの記録** をクリックします。
 - b) 記録したいオブジェクトの上にマウスを移動して **Ctrl+Alt** を押します。Silk4NET によって、**ロケータ** テキスト フィールドにロケータ文字列が表示されます。
 - c) **ロケータの詳細** テーブルで、使用するロケータを選択します。
 - d) ロケータをコピーしてオブジェクト マップに貼り付けます。
 6. 必要に応じて、ニーズに合わせて項目名またはロケータ テキストを変更します。

無効な文字またはロケータを使用すると、項目名またはロケータ テキストが赤で表示され、ツール ヒントにエラーの説明が表示されます。

オブジェクト マップ項目として無効な文字には、¥、/、<、>、"、:、*、?、|、=、..、@、[,] があります。

無効なロケータ パスは、空または不完全なロケータ パスです。

7. **CTRL+S** を押して、変更を保存します。

変更したロケータ パスが既存のスクリプトによって使用されている場合は、新しいロケータ パスを使用するように、そのビジュアルテストまたはスクリプトを手動で変更する必要があります。

テスト アプリケーションからのオブジェクト マップの更新

テスト アプリケーションの項目が変化した場合は、**オブジェクト マップ UI** を使用してそれらの項目のロケータを更新できます。

1. ソリューション エクスプローラ で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。

2. 次のいずれか 1 つを選んでください：

- 使用するオブジェクト マップをダブルクリックします。
- 使用するオブジェクト マップを右クリックし、**開く** をクリックします。

オブジェクト マップ項目および各項目に関連付けられたロケータの階層が、オブジェクト マップに表示されます。

3. **ロケータの更新** をクリックします。 **Locator Spy** が表示され、Silk4NET によってテスト アプリケーションが開かれます。

4. 記録するオブジェクトの上にカーソルを合わせて、**CTRL+ALT** を押します。 Silk4NET の **ロケータ** テキスト フィールドにロケータ文字列が表示されます。

5. **ロケータの詳細** テーブルで、使用するロケータを選択します。

6. **ロケータ** テキスト フィールドに表示されているロケータから、使用しない属性を削除します。

7. **ロケータの検証** をクリックして、ロケータが機能することを検証します。

8. **ロケータをエディターに貼り付け** をクリックして、オブジェクト マップのロケータを更新します。

9. 変更されたオブジェクト マップを保存します。

AUT からオブジェクト マップ項目を更新するときに、オブジェクト マップ ツリーのリーフ ノードの XPath 表現のみを変更できます。 親ノードの XPath 表現を変更することはできません。 オブジェクト マップ ツリー内のより高いレベルのノードにある XPath 表現が更新後に整合しなくなると、エラー メッセージが表示されます。

例

たとえば、次の 3 つの階層レベルを持つオブジェクト マップ ID を含むオブジェクト マップ項目があるとします：

```
WebBrowser.Dialog.Cancel
```

これらの階層レベルに対応する XPath 表現は次のようになります：

```
/BrowserApplication//Dialog//PushButton[@caption='Cancel']
```

- 最初の階層レベル： /BrowserApplication
- 2 番目の階層レベル： //Dialog
- 3 番目の階層レベル： //PushButton[@caption='Cancel']

次のロケータを使用して、オブジェクト マップ項目を更新できます：

```
/BrowserApplication//Dialog//PushButton[@id='123']
```

- 最初の階層レベル : /BrowserApplication
- 2 番目の階層レベル : //Dialog
- 3 番目の階層レベル : //PushButton[@id='123']

2 番目のレベルの階層が一致しないため、次のロケータを使用してオブジェクト マップ項目を更新することはできません :

```
/BrowserApplication//BrowserWindow//PushButton[@id='9999999']
```

- 最初の階層レベル : /BrowserApplication
- 2 番目の階層レベル : //BrowserWindow
- 3 番目の階層レベル : //PushButton[@id='9999999']

オブジェクト マップ項目のコピー

オブジェクト マップ内、またはオブジェクト マップ間で、オブジェクト マップ エントリをコピーおよび貼り付けできます。たとえば、2 つの異なるテスト アプリケーションに同じ機能が存在する場合は、一方のオブジェクト マップの一部分をコピーして、他方のオブジェクト マップに貼り付けることができます。

1. **ソリューション エクスプローラ** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ フォルダ** をクリックします。
2. 次のいずれか 1 つを選んでください：
 - コピーするオブジェクト マップ項目を含むオブジェクト マップをダブルクリックします。
 - コピーするオブジェクト マップ項目を含むオブジェクト マップを右クリックし、**開く** を選択します。

オブジェクト マップ項目および各項目に関連付けられたロケータの階層が、オブジェクト マップに表示されます。

3. コピーするオブジェクト マップ項目に移動します。
たとえば、コピーするオブジェクト マップ項目を検索するには、ノードの展開が必要な場合があります。
4. 次のいずれか 1 つを選んでください：
 - コピーするオブジェクト マップ項目を右クリックし、**ツリーのコピー** を選択します。
 - コピーするオブジェクト マップ項目をクリックし、Ctrl+C を押します。
5. オブジェクト マップ階層で、コピーした項目を貼り付ける位置に移動します。
たとえば、階層の第 1 レベルに項目を組み込むには、項目リストの最初の項目の名前をクリックします。特定の項目の 1 レベル下にコピーする項目の位置を設定するには、コピーする項目の上にある項目をクリックします。
オブジェクト マップ間でコピーして貼り付けるには、オブジェクト マップ項目をコピーしたマップを終了し、オブジェクト マップ項目を貼り付けるオブジェクト マップを開いて編集する必要があります。
6. 次のいずれか 1 つを選んでください：
 - コピーしたオブジェクト マップ項目を貼り付けるオブジェクト マップ内の位置を右クリックし、**貼り付け** を選択します。
 - コピーしたオブジェクト マップ項目を貼り付けるオブジェクト マップ内の位置をクリックし、Ctrl+V を押します。

オブジェクト マップ項目が、階層内の新しい位置に表示されます。

7. **CTRL+S** を押して、変更を保存します。

移動したオブジェクト マップ項目を既存のスクリプトで使用する場合は、階層内の新しい位置を使用するようにスクリプトを手動で変更する必要があります。


オブジェクト マップ項目の追加

スクリプトを記録すると、オブジェクト マップ項目が自動的に作成されます。場合によっては、手動でオブジェクト マップ項目を追加することもできます。

1. **ソリューション エクスプローラ** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 次のいずれか 1 つを選んでください：
 - 名前を変更するオブジェクト マップ項目を含むオブジェクト マップをダブルクリックします。
 - 名前を変更するオブジェクト マップ項目を含むオブジェクト マップを右クリックし、**開く** を選択します。

オブジェクト マップ項目および各項目に関連付けられたロケータの階層が、オブジェクト マップに表示されます。

3. オブジェクト マップ階層で、オブジェクト マップ項目を追加する位置に移動します。
たとえば、階層の第 1 レベルに項目を組み込むには、項目リストの最初の項目の名前をクリックします。特定の項目の 1 レベル下に新しい項目の位置を設定するには、コピーする項目の上にある項目をクリックします。
4. **新規挿入** をクリックします。新しい項目が階層に追加されます。
5. 有効なロケータ パスがある場合は、使用する項目名とロケータ パスを入力して Enter を押すことができます。有効なロケータ パスを判別するには、以下のステップで説明するように、**Locator Spy** ダイアログ ボックスを使用します。
 - a) **Silk4NET > ロケータの記録** をクリックします。
 - b) 記録したいオブジェクトの上にマウスを移動して **Ctrl+Alt** を押します。Silk4NET によって、**ロケータ** テキスト フィールドにロケータ文字列が表示されます。
 - c) **ロケータの詳細** テーブルで、使用するロケータを選択します。
 - d) ロケータをコピーしてオブジェクト マップに貼り付けます。
6. 必要に応じて、ニーズに合わせて項目名またはロケータ テキストを変更します。
無効な文字またはロケータを使用すると、項目名またはロケータ テキストが赤で表示され、ツール ヒントにエラーの説明が表示されます。
オブジェクト マップ項目として無効な文字には、¥、/、<、>、"、:、*、?、|、=、..、@、[,] があります。
無効なロケータ パスは、空または不完全なロケータ パスです。
7. **CTRL+S** を押して、変更を保存します。

 **注:** オブジェクト マップ ツリーに含まれるすべてのノードのすべての子ノードは、オブジェクト マップを保存するときにアルファベット順にソートされます。

スクリプトからオブジェクト マップを開く

スクリプトを編集している際に、スクリプトのオブジェクト マップ エントリを右クリックし、**Silk4NET 資産を開く** を選択してオブジェクト マップを開くことができます。オブジェクト マップは GUI 上で開かれます。

例

```
// VB .NET code
<TestMethod()> Public Sub TestMethod1()
    With _desktop.Window("Untitled -
Notepad").TextField("TextField").TypeKeys("hello")
```

```
End With
End Sub

// C# code
[TestMethod]
public void TestMethod1()
{
    Window untitledNotepad = _desktop.Window("Untitled - Notepad");
    untitledNotepad.TextField("TextField").TypeKeys("hello");
}
```

上記のコード例で、Untitled - Notepad エントリをオブジェクト マップで開く場合は、Untitled - Notepad を右クリックします。オブジェクト マップで Untitled - Notepad.TextField エントリをオブジェクト マップで開く場合は、TextField を右クリックします。

テスト アプリケーションでのオブジェクト マップ項目のハイライト

オブジェクト マップ項目を追加または記録したあと、**ハイライト** をクリックして、テスト アプリケーションで項目をハイライトできます。変更する項目であることをオブジェクト マップ内で確認する場合などに項目をハイライトできます。

1. **ソリューション エクスプローラ** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 次のいずれか 1 つを選んでください：

- 使用するオブジェクト マップをダブルクリックします。
- 使用するオブジェクト マップを右クリックし、**開く** をクリックします。

オブジェクト マップ項目および各項目に関連付けられたロケータの階層が、オブジェクト マップに表示されます。

3. オブジェクト マップ階層で、テスト アプリケーションでハイライトするオブジェクト マップ項目を選択します。



注： テスト アプリケーションの 1 つのインスタンスのみが実行中であることを確認します。テスト アプリケーションの複数のインスタンスを実行すると、複数のオブジェクトがロケータと一致するためエラーになります。

4. **ハイライト** をクリックします。

テスト アプリケーションがオブジェクト マップに関連付けられていないと、**アプリケーションの選択** ダイアログ ボックスが表示されることがあります。この場合は、テストするアプリケーションを選択し、**OK** をクリックします。

Silk4NET によってテスト アプリケーションが開かれ、オブジェクト マップ項目を示すコントロールの周囲に緑のボックスが表示されます。

スクリプトでのロケータからオブジェクト マップ エントリへの移動

オブジェクト マップ エントリの **ID** 以外の情報、つまりコマンドが実行されたときに Open Agent が使用する生のロケータを参照したい場合は、以下の手順に従います。

1. スクリプトを開きます。

2. 識別するスクリプトの行の文字列内にカーソルを置いてください。
3. 右クリックして、**Silk4NET 資産を開く** を選択します。



注:

カーソルがオブジェクト マップ エントリではない文字列内にある場合でも、Silk4NET は、それがオブジェクト マップ エントリであるとみなし、期待した結果が得られない場合があります。

選択された適切な項目が表示された状態で、**オブジェクト マップ** ウィンドウが、ツリー ビューに表示されます。

オブジェクト マップのエラーの検出

無効な文字またはロケータを使用すると、項目名またはロケータ テキストが赤で表示され、ツール ヒントにエラーの説明が表示されます。**オブジェクト マップ** ウィンドウのツール バーを使用して、エラーに移動します。

1. **ソリューション エクスプローラ** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ フォルダ** をクリックします。
2. 次のいずれか 1 つを選んでください：

- トラブルシューティングするオブジェクト マップをダブルクリックします。
- トラブルシューティングするオブジェクト マップを右クリックし、**開く** を選択します。

オブジェクト マップ項目および各項目に関連付けられたロケータの階層が、オブジェクト マップに表示されます。

3. 赤色で表示された項目名またはロケータ テキストを探します。
4. 必要に応じて、ニーズに合わせて項目名またはロケータ テキストを変更します。

無効な文字またはロケータを使用すると、項目名またはロケータ テキストが赤で表示され、ツール ヒントにエラーの説明が表示されます。

オブジェクト マップ項目として無効な文字には、¥、/、<、>、"、:、*、?、|、=、..、@、[,] があります。

無効なロケータ パスは、空または不完全なロケータ パスです。

5. **CTRL+S** を押して、変更を保存します。

オブジェクト マップ項目の削除

テスト アプリケーションに存在しなくなったなどの理由により、オブジェクト マップから項目を削除できます。

1. **ソリューション エクスプローラ** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ フォルダ** をクリックします。
2. 次のいずれか 1 つを選んでください：

- 削除するオブジェクト マップ項目を含むオブジェクト マップをダブルクリックします。
- 削除するオブジェクト マップ項目を含むオブジェクト マップを右クリックし、**開く** を選択します。

オブジェクト マップ項目および各項目に関連付けられたロケータの階層が、オブジェクト マップに表示されます。

3. 削除するオブジェクト マップ項目に移動します。
たとえば、削除するオブジェクト マップ項目を検索するには、ノードの展開が必要な場合があります。
4. 次のいずれか 1 つを選んでください：

- 削除するオブジェクト マップ項目を右クリックし、**削除** を選択するか、そのオブジェクト マップ項目のすべての子項目も削除する場合は **ツリーの削除** を選択します。
- 削除するオブジェクト マップ項目をクリックし、**Del** を押すか、そのオブジェクト マップ項目のすべての子項目も削除する場合は **Ctrl+Del** を押します。

5. **CTRL+S** を押して、変更を保存します。

削除したオブジェクト マップ項目または子オブジェクトを既存のスクリプトで使用する場合は、そのオブジェクト マップ項目への参照をスクリプトで、手動で変更する必要があります。

オブジェクト マップのベスト プラクティス

ベスト プラクティスとして、テストを記録する前に、すべてのオブジェクト マップ項目を確認することをお勧めします。たとえば、すべてのオブジェクトをクリックしてすべてのウィンドウを開くテストをテスト アプリケーションで作成するとします。各オブジェクトにどのように名前が付けられているかを確認し、機能テストを記録する前に、必要な変更を加えることができます。オブジェクト項目名の確認と変更が完了したら、オブジェクト マップ項目に対して作成したテストを削除できます。

オブジェクト マップのツリーを移動するには、矢印キーを使用します。

イメージ解決のサポート

イメージ解決は、次の場合に使用できます。

- オブジェクト解決で識別できない、高度にカスタマイズされたコントロールを含むテスト アプリケーションを簡便に操作する場合。座標ベースのクリックの代わりにイメージ クリックを使用し、指定されたイメージをクリックできます。
- テスト対象アプリケーションのグラフィカル オブジェクト (グラフなど) をテストする場合。
- テスト対象アプリケーションの視覚的な UI のチェックを実行する場合。

認識されないコントロールをクリックするには、ImageClick メソッドとイメージ資産を使用します。テスト対象アプリケーションに、認識されないコントロールがあるかどうかを確認するには、VerifyAsset メソッドとイメージ検証を使用します。

イメージ解決メソッドは、Silk4NET でサポートされるすべてのテクノロジー ドメインでサポートされます。


イメージ クリックの記録


イメージ クリックの記録を行うと、大量のイメージが生成されてわかりにくくなるため、デフォルトではイメージ クリックの記録が無効となり、座標ベースのクリックの記録が優先されます。イメージ クリックの記録を有効にするには、以下のいずれかを実行します。

- **記録** ダイアログ ボックスで、'**ImageClick**' を記録する をオンにします。
- ツール > オプション > 記録 > 全般 をクリックし、'**ImageClick**' を記録する の値を はい に設定します。
- Silk4NET > オプションの編集 をクリックして、記録 タブを選択し、'**ImageClick**' を記録する セクションのチェック ボックスをチェックします。

イメージ クリックの記録を有効にすると、Silk4NET は、オブジェクト解決またはテキスト解決ができない場合に、ImageClick メソッドを記録します。イメージ クリックが記録されない場合でも、コントロール用にイメージ クリックをスクリプトに挿入できます。

ImageClick 操作を記録しない場合は、イメージ クリックの記録をオフに切り替えて通常のクリックまたはテキスト クリックを記録できます。

 **注:** 記録されたイメージは再利用されません。Silk4NET は、記録するイメージ クリックごとに新しいイメージ資産を作成します。

 **注:** イメージ クリックの記録は、Java AWT/Swing コントロールを使用するアプリケーションまたはアプレットではサポートされません。

イメージ解決メソッド

Silk4NET では、イメージ解決用に次のメソッドが用意されています。

メソッド	説明
ImageClick	資産で指定されたイメージの中央をクリックします。イメージが見つかるか、オブジェクト解決タイムアウト (同期オプションで定義可能) が経過するまで待機します。
ImageExists	資産で指定されたイメージが存在するかどうかを返します。
ImageRectangle	資産で指定されたイメージのオブジェクト相対矩形を返します。

メソッド	説明
ImageClickFile	ファイルで指定されたイメージをクリックします。
ImageExistsFile	ファイルで指定されたイメージが存在するかどうかを返します。
ImageRectangleFile	ファイルで指定されたイメージのオブジェクト相対矩形を返します。
VerifyAsset	検証資産を実行します。検証に合格しなかった場合、VerificationFailedException がスローされます。
TryVerifyAsset	検証資産を実行し、検証に合格したかどうかを返します。

イメージ資産

イメージ資産は、次の場合に使用できます。

- オブジェクト解決で識別できない、高度にカスタマイズされたコントロールを含むテストアプリケーションを簡単に操作する場合。座標ベースのクリックの代わりにイメージクリックを使用し、指定されたイメージをクリックできます。
- テスト対象アプリケーションのグラフィカルオブジェクト(グラフなど)をテストする場合。

イメージ資産は、イメージと、Silk4NET で資産を操作するために必要な追加情報で構成されます。

Silk4NET では、イメージ資産用に次のメソッドが用意されています。

メソッド	説明
ImageClick	指定されたイメージ資産の中央をクリックします。イメージが見つかるか、オブジェクト解決タイムアウト(同期オプションで定義可能)が経過するまで待機します。
ImageExists	指定されたイメージ資産があるかどうかを返します。
ImageRectangle	指定されたイメージ資産のオブジェクト相対矩形を返します。

イメージ資産は、プロジェクトの Image Assets フォルダに置く必要があります。 .imageasset ファイルは、埋め込みリソースにする必要があります。

イメージ資産の作成

イメージ資産は、次のいずれかの方法で作成できます。

- 新しいイメージ資産を既存のスクリプトに挿入。
- 記録時。
- メニューから。

新しいイメージ資産を で作成するには、以下のステップを実行します。

1. メニューで、**Silk4NET > 新規イメージ資産** をクリックします。
2. 資産のわかりやすい名前を **名前** フィールドに入力し、**Silk4NET イメージ資産** をダブルクリックします。イメージ資産の UI が開きます。
3. 資産にイメージを追加する方法を選択します。
 - 既存のイメージを使用する場合は、**参照** をクリックし、イメージファイルを選択します。
 - テスト対象アプリケーションの UI から新しいイメージをキャプチャする場合は、**キャプチャ** をクリックします。
4. 新しいイメージをキャプチャする場合は、キャプチャする画面領域を選択し、**選択範囲をキャプチャします** をクリックします。

5. 省略可能:オプション **クライアント領域のみ** を設定して、Silk4NET がイメージ検証と AUT の UI を比較するとき、実際に AUT の一部であるイメージの部分だけを考慮するように定義できます。

6. **精度レベル** を指定します。

精度レベルは、クリックされるイメージがテスト対象アプリケーションのイメージと異なってもよい度合いを定義し、これを超えて異なっている場合、Silk4NET はイメージが異なっていると判断します。これは、画面解像度が異なる複数のシステムまたはブラウザをテストする場合に役立ちます。誤検出を防ぐため、できるだけ精度レベルを高くすることを推奨します。デフォルトの精度レベル値は、6 です。デフォルトの精度レベル値は、オプションで変更できます。



注: 精度レベル を 5 未満に設定した場合、イメージの実際の色が比較で考慮されなくなります。イメージのグレースケール表現だけが比較されます。

7. イメージ資産を保存します。

新しいイメージ資産が、**ソリューション エクスプローラ** で現在のプロジェクトの下に表示され、これを使用してイメージクリックを実行できます。

同じイメージ資産に複数のイメージを追加できます。

同じイメージ資産に複数のイメージを追加する

テスト時に、異なるテスト構成を使用して、複数の環境の機能をテストする必要が生じることがよくあります。環境が異なると、イメージ資産にキャプチャしたイメージと実際のイメージが若干異なることがあり、イメージが存在するにもかかわらずイメージクリックが失敗することがあります。このような場合、同じイメージ資産に複数のイメージを追加できます。

イメージ資産に別のイメージを追加するには、以下を実行します。

1. イメージ資産に追加したいイメージをダブルクリックします。イメージ資産の UI が開きます。
2. UI の下部に表示されるプラス記号をクリックして、新しいイメージをイメージ資産に追加します。
3. イメージ資産を保存します。

新しいイメージが資産に追加されます。イメージクリックが呼び出されるたびに、一致するものに到達するまで、Silk4NET は資産のイメージとテスト対象アプリケーションの UI のイメージを比較します。デフォルトで、Silk4NET は資産に追加された順にイメージを比較します。



注: Silk4NET で比較するイメージの順番を変更するには、イメージ資産の UI の下部でイメージをクリックし、目的の場所にドラッグします。左から右の順に比較されます。最初に比較されるのは、最も左にあるイメージです。

スクリプトから資産を開く

スクリプトを編集しているときに、資産を右クリックして **Silk4NET 資産を開く** を選択し、資産を開くことができます。これにより、GUI で資産が開きます。

資産がシステム上のファイルへの参照である場合 (ImageClickFile によって参照される場合など)、ファイルはシステムのデフォルト エディターで開かれます。

イメージ検証

イメージ検証を使用して、テスト対象アプリケーション (AUT) の UI にイメージがあるかどうかをチェックできます。

イメージ検証は、イメージと、Silk4NET で資産を操作するために必要な追加情報で構成されます。

イメージ検証を実行するには、VerifyAsset メソッドを使用します。

イメージ検証資産は、プロジェクトの Verifications フォルダに置く必要があります。 .verification ファイルは、埋め込みリソースにする必要があります。

Silk4NET が AUT のイメージを見つけることができなかつた場合、イメージ検証は失敗します。この場合、スクリプトの実行は中断され、VerificationFailedException がスローされます。この動作を防止するには、TryVerifyAsset メソッドを使用します。

AUT 内でイメージ検証のロケータが見つからなかつた場合、Silk4NET は ObjectNotFoundException をスローします。

TrueLog Explorer で成功したイメージ検証を開くには、検証ステップの **情報** タブで **検証を開く** をクリックします。TrueLog Explorer で失敗したイメージ検証を開くには、検証ステップの **情報** タブで **相違点の表示** をクリックします。失敗したイメージ検証が、精度レベルを低くすれば成功すると判断された場合は、成功する精度レベルが提示されます。

イメージ検証の作成

イメージ検証は、次のいずれかの方法で作成できます。

- メニュー を使用。
- 記録時。

新しいイメージ検証をメニュー で作成するには、以下のステップを実行します。

1. **Silk4NET > 新規イメージ検証** をクリックします。
2. 資産のわかりやすい名前を **名前** フィールドに入力し、**Silk4NET イメージ検証** をダブルクリックします。イメージ検証の UI が開きます。
3. **識別** をクリックして、テスト対象アプリケーションの、検証するイメージを識別します。
4. 省略可能:最初にキャプチャしたイメージから変更されたために、テスト対象アプリケーションから同じイメージを再キャプチャする必要がある場合は、**再キャプチャ** をクリックします。
5. 省略可能:**検証** をクリックすると、イメージ検証が機能するかどうかをテストできます。
6. 省略可能:Silk4NET がイメージ検証とテスト対象アプリケーション (AUT) の UI を比較するときに考慮しない除外領域をイメージ検証に追加できます。
7. 省略可能:オプション **クライアント領域のみ** を設定して、Silk4NET がイメージ検証と AUT の UI を比較するときに、実際に AUT の一部であるイメージの部分だけを考慮するように定義できます。
8. **精度レベル** を指定します。

精度レベルは、検証されるイメージがテスト対象アプリケーションのイメージと異なってもよい度合いを定義し、これを超えて異なっている場合、Silk4NET はイメージが異なっていると判断します。これは、画面解像度が異なる複数のシステムまたはブラウザをテストする場合に役立ちます。誤検出を防ぐため、できるだけ精度レベルを高くすることを推奨します。デフォルトの精度レベル値は、オプションで変更できます。



注: 精度レベル を 5 未満に設定した場合、イメージの実際の色が比較で考慮されなくなります。イメージのグレースケール表現だけが比較されます。

9. イメージ検証を保存します。

新しいイメージ検証が**ソリューション エクスプローラ**に表示され、これを使用して、テスト対象アプリケーションの UI にイメージが存在するかどうかをチェックできます。

記録中にイメージ検証を追加する

イメージ検証をスクリプトに追加して、テスト対象アプリケーションの UI に認識されないコントロールがあるかどうかをチェックできます。スクリプトの記録中にイメージ検証を追加するには、以下のステップを実行します。

1. 記録を開始します。
2. 検証するイメージの上にマウスカーソルを移動して、**Ctrl+Alt** を押しながらかlickします。Silk4NET から、プロパティまたはイメージを検証するかどうかを尋ねられます。

3. **イメージ検証の作成または挿入** を選択します。
4. 次のいずれか 1 つのステップを行います：
 - イメージ検証の UI で新しいイメージ検証を作成するには、リストボックスから **新規** を選択します。
 - 既存のイメージ検証資産を挿入するには、リストボックスからイメージ検証資産を選択します。
5. **OK** をクリックします。
 - 新しいイメージ検証の作成を選択した場合は、イメージ検証の UI が表示されます。
 - 既存のイメージ検証の使用を選択した場合は、イメージ検証がスクリプトに追加されます。この場合、このトピックの残りのステップはスキップできます。
6. 新しいイメージ検証を作成するには、イメージ検証の UI で **検証** をクリックします。
7. AUT のイメージの上にマウスカーソルを移動して、**Ctrl+Alt** を押しながらクリックします。イメージ検証の UI に、新しいイメージ検証が表示されます。
8. **OK** をクリックします。新しいイメージ検証が現在のプロジェクトに追加されます。
9. 記録を続けます。

複数のプロジェクトでの資産の使用

Silk4NET では、イメージ資産、イメージ検証、およびオブジェクト マップが資産と呼ばれます。資産が配置されているプロジェクトの範囲外でそれらの資産を使用する場合、資産を使用するプロジェクトから、資産を配置するプロジェクトに、プロジェクトの直接的な参照を追加する必要があります。

再生中に資産が使用されると、Silk4NET は、最初に現在のプロジェクト内でその資産を検索します。Silk4NET で現在のプロジェクト内に資産が検出されなかった場合、Silk4NET は現在のプロジェクトがプロジェクト参照を持つプロジェクトを追加検索します。それでも資産が見つからない場合、Silk4NET はエラーをスローします。

Silk4NET は、検出されたすべての資産の順番を保持します。



注: 別のプロジェクトに依存関係として追加したプロジェクトのコードが、依存関係のあるプロジェクトのコードで参照されていない場合、ユーザーが依存関係のあるプロジェクトを作成すると、Visual Studio はそのプロジェクト依存関係を削除します。プロジェクト依存関係に置かれている資産を使用するには、依存関係のあるプロジェクトから、資産が置かれるプロジェクトのメンバーに、コード参照を追加する必要があります。このようなコード参照を追加することで、依存関係のあるプロジェクトを作成しているときに、Visual Studio によってプロジェクト依存関係が削除されないようにできます。たとえば、クラスまたは定数をプロジェクト依存関係に追加した後、依存関係のあるプロジェクトのコードでそのクラスまたは定数を呼び出すことができます。

複数のプロジェクトに同じ名前の資産が存在する場合に、現在のプロジェクトに含まれている資産を使用しないときは、資産を使用するメソッドで使用する特定の資産を定義できます。使用する資産を定義するには、メソッドを呼び出すときに、アセンブリ名を接頭辞として資産名に追加します。アセンブリ名は、デフォルトでプロジェクト名に設定されます。

例：プロジェクトの参照の追加

プロジェクト *ProjectA* にコード

```
'VB code  
ImageClick("imageAsset")
```

を呼び出すテストが含まれており、イメージ資産 *imageAsset* がプロジェクト *ProjectB* に置かれている場合、プロジェクトの直接的な参照を *ProjectA* から *ProjectB* に追加する必要があります。

例：特定の資産の呼び出し

ProjectA と *ProjectB* の両方に *anotherImageAsset* という名前のイメージ資産が含まれている場合に、*ProjectB* からイメージ資産を明示的にクリックする場合、次のコードを使用します：

```
'VB code  
ImageClick("ProjectB:anotherImageAsset")
```


テストの拡張


このセクションでは、テストの拡張方法について説明します。


Windows DLL の呼び出し


このセクションでは、DLL を呼び出す方法について説明します。DLL は Open Agent のプロセス内から、または AUT (テスト対象アプリケーション) から呼び出すことができます。これにより、テスト スクリプト内の既存のネイティブ DLL を再利用できます。

Open Agent 内の DLL 呼び出しは通常、AUT 内の UI コントロールと対話しないグローバル関数を呼び出す場合に使用されます。

AUT 内の DLL 呼び出しは通常、アプリケーションの UI コントロールと対話する関数を呼び出す場合に使用されます。これにより、Silk4NET は再生中に DLL 呼び出しを自動的に同期できます。

 **注:** 32 ビット アプリケーションでは 32 ビット DLL を、64 ビット アプリケーションでは 64 ビット DLL を呼び出すことができます。Open Agent は 32 ビットと 64 ビットの両方の DLL を実行できます。

 **注:** .NET Framework では、P/Invoke という DLL 呼び出しも組み込みでサポートされています。P/Invoke を Visual Basic スクリプト内で使用すると、このスクリプトを実行するプロセス内で DLL 関数を呼び出すことができます。ただし、AUT では Silk Test Workbench を使用して DLL 関数を呼び出すことができる一方で、自動同期は行われません。

 **注:** DLL を呼び出すには、C インターフェイスを使用する必要があります。同様に .dll というファイル拡張子の付いた .NET アセンブリを呼び出す場合は、DLL 呼び出し機能を使用しないで、.NET スクリプト内でアセンブリへの参照を追加します。

スクリプトからの Windows DLL の呼び出し

DLL の宣言を開始するには、DLL 属性を持つインターフェイスを使用します。宣言の構文は次のとおりです。

dllname スクリプトから呼び出す関数が含まれた DLL ファイルの完全パスの名前。DLL パス内の環境変数は自動的に解決されます。パス内のバックslashは 2 重 (¥¥) にする必要はありません。単一のバックslash (¥) を使用してください。

DllInterfaceName スクリプト内で DLL と対話するために使用される識別子。

FunctionDeclaration 呼び出そうとしている DLL 関数の関数宣言。

DLL 関数の宣言構文

DLL 関数の宣言は、一般に以下の形式を取ります。

戻り値のない関数の場合、宣言の形式は以下のとおりです、

return-type 戻り値のデータ型。

function-name 関数の名前。

arg-list 関数に渡される引数のリスト。

リストは以下のように指定します。

data-type	引数のデータ型。
identifier	引数の名前。

DLL 関数への引数の受け渡し

DLL 関数は C で記述されているため、これらの関数に渡す引数には適切な C データ型を指定する必要があります。次のデータ型がサポートされます。

次のデータ型を持つ引数または戻り値には、このデータ型を使用します。

- int
- INT
- long
- LONG
- DWORD
- BOOL
- WPARAM
- HWND

の型は、4 バイト値を持つすべての DLL 引数に対して有効です。

C データ型 long および int64 を持つ引数または戻り値には、このデータ型を使用します。の型は、8 バイト値を持つすべての DLL 引数に対して有効です。

C データ型 short および WORD を持つ引数または戻り値には、このデータ型を使用します。の型は、2 バイト値を持つすべての DLL 引数に対して有効です。

C データ型 bool を持つ引数または戻り値には、このデータ型を使用します。


String C で String となる引数または戻り値には、このデータ型を使用します。

C データ型 double を持つ引数または戻り値には、このデータ型を使用します。


C データ型 RECT を持つ引数には、このデータ型を使用します。は戻り値として使用できません。


C データ型 POINT を持つ引数には、このデータ型を使用します。POINT は戻り値として使用できません。


C データ型 HWND を持つ引数には、このデータ型を使用します。TestObject は戻り値として使用できませんが、戻り値型として Integer を持つ HWND を戻す DLL 関数を宣言できます。

 **注:** 渡された TestObject は インターフェイスを実装して、DLL 関数に渡される TestObject のウィンドウ ハンドルを Silk4NET が 判別できるようにする必要があります。 そうしないと、この DLL 関数を呼び出すときに、例外がスローされます。

List ユーザー定義の C 構造体の配列には、このデータ型を使用します。List は戻り値として使用できません。

 **注:** List をパラメーターとして使用する場合は、渡されるリストに、戻される内容を保持できるだけのサイズを確保する必要があります。

 **注:** C 構造体は List で表すことができます。この場合、すべてのリスト要素は構造体のメンバに対応しています。最初の構造体メンバは、リスト内の最初の要素で表されます。2 番目の構造体メンバは、リスト内の 2 番目の要素で表されます (以下同様)。

 **注:** DLL 関数に渡す引数の前には、いずれかの データ型を配置する必要があります。

DLL 関数への文字列引数の受け渡し

DLL 関数に渡している文字列、または DLL 関数から戻される文字列は、デフォルトでは Unicode Strings として処理されます。DLL 関数に ANSI String 引数が必要な場合は、DllFunctionOptions 属性の CharSet プロパティを使用します。

例

```
<Dll( "user32.dll" )> Public Interface IUserDll32Functions
    <DllFunctionOptions(CharSet:=CharacterSet.Ansi)> Function
    SendMessageA( _
        ByVal obj As TestObject, ByVal message As Integer , ByVal wParam
    As Integer , ByVal lParam As String ) As Integer
End Interface
```

DLL 呼び出しから String を ByRef 引数 として戻した場合、String のサイズが 256 文字以下であれば、デフォルトの動作に従います。戻される String が 256 文字を超えている場合は、作成された String を保持できるだけの長さを持つ、Visual Basic String を渡します。

例

```
1024 個の空白文字を含む String を作成するには、以下のコードを使用します。
Dim longEmptyString = New String ( " "c , 1024 )
```

この String を ByRef 引数として DLL 関数に渡します。すると、この DLL 関数は最大 1024 文字の String を戻します。

関数の戻り値として DLL から String が戻される場合、DLL は DLL 関数 FreeDllMemory を実装し、DLL 関数から戻される C String ポインターを受け入れて、以前に割り当てられたメモリーを解放する必要があります。このような関数が存在しない場合、メモリーはリークされます。

DLL 名のエイリアス設定

DLL 関数に、Visual Basic の予約語と同じ名前が付いている場合、または DLL 関数に名前ではなく序数が付いている場合は、宣言内でこの関数の名前を変更し、エイリアス ステートメントを使用して、宣言した名前と実際の名前をマッピングする必要があります。

例

たとえば、Exit ステートメントは Visual Basic コンパイラーで予約されています。したがって、関数 exit を呼び出すには、次のようにその関数を別の名前で宣言し、エイリアス ステートメントを追加する必要があります。

```
<Dll("mydll.dll")> Public Interface IMyDllFunctions
    <DllFunctionOptions(Alias:="exit")> Sub MyExit()
End Interface
```

DLL 関数呼び出しの表記規則

DLL 関数を呼び出す場合は、次に示す呼び出し規則がサポートされています。

- __stdcall
- __cdecl

DLL 関数を呼び出す場合は、__stdcall 呼び出し規則がデフォルトで使用されます。この呼び出し規則は、すべての Windows API DLL 関数で使用されます。

DLL 関数の呼び出し規則を変更するには、DllFunctionOptions 属性の CallingConvention プロパティを使用します。

例

次のコード例では、`__decl` 呼び出し規則を使用して DLL 関数を宣言します。

```
<Dll("msvcrt.dll")> Public Interface IMsVisualCRuntime
  <DllFunctionOptions(CallingConvention:=CallingConvention.Cdecl)>
  Function cos(ByVal input As Double) As Double
End Interface
```

Microsoft ユーザー補助を使用したオブジェクト解決の向上

Microsoft ユーザー補助を、クラスレベルでオブジェクトを簡単に認識するために使用することができます。Internet Explorer や Microsoft アプリケーションのいくつかのオブジェクトには、ユーザー補助を有効にすることで Silk4NET によってより良く認識されるようになるものがあります。たとえば、ユーザー補助を有効にしないと、Silk4NET は Microsoft Word のメニューバーや表示されるタブについて基本的な情報のみを記録します。しかし、ユーザー補助を有効にすると、Silk4NET はそれらのオブジェクトを完全に認識できるようになります。

例

ユーザー補助を使用しないと、Silk4NET は DirectUIHwnd コントロールを完全に認識できません。これは、このコントロールのパブリックな情報が存在しないためです。Internet Explorer は、2 つの DirectUIHwnd コントロールを使用しています。1 つはブラウザウィンドウの下部に表示されるポップアップです。このポップアップには、通常、次の情報が表示されます。

- Internet Explorer を既定のブラウザにしたいかどうかを尋ねるダイアログボックス。
- ダウンロード オプション（開く、保存、キャンセル）。

Silk4NET でプロジェクトを開始して、DirectUIHwnd ポップアップに対してロケータを記録すると、ユーザー補助が無効にしている場合、単一のコントロールのみが表示されます。ユーザー補助を有効にした場合には、DirectUIHwnd コントロールを完全に認識した情報が得られます。

ユーザー補助の使用

Win32 では、ジェネリックコントロールとして認識されるコントロールにユーザー補助 サポートが使用されます。Win32 は、コントロールを特定すると、ユーザー補助オブジェクトをコントロールのすべてのユーザー補助の子とともに取得しようとします。

ユーザー補助によって返されるオブジェクトは、AccessibleControl、Button、CheckBox のいずれかのクラスになります。Button および Checkbox は、そのクラス用に定義されたメソッドとプロパティの通常セットをサポートするので個別に扱われます。ユーザー補助によって返されるすべてのジェネリックオブジェクトの場合、クラスは AccessibleControl です。

例

ユーザー補助が有効になる前、アプリケーションのコントロール階層が次のようになっています。

- コントロール

- コントロール
- ボタン

ユーザー補助を有効にすると、階層は次のようになります。

- コントロール
 - コントロール
 - ユーザー補助コントロール
 - ユーザー補助コントロール
 - ボタン
- ボタン

ユーザー補助の有効化

Win32 アプリケーションをテストしているときに、でオブジェクトを認識できない場合は、最初にユーザー補助を有効にする必要があります。ユーザー補助は、オブジェクトの認識機能をクラス レベルで強化するためのものです。

のユーザー補助を有効にするには、以下の手順を実行します。

1. をクリックします。ダイアログ ボックスが表示されます。
2. **詳細設定** をクリックします。
3. **Microsoft ユーザー補助を使用する** オプションを選択します。ユーザー補助が有効になります。

テキスト解決のサポート

テキスト解決メソッドを使用して、オブジェクト解決で識別できない、高度にカスタマイズされたコントロールを含むテスト アプリケーションを便利に操作できます。座標ベースのクリックの代わりにテキストクリックを使用し、コントロール内に指定されたテキスト文字列をクリックできます。

たとえば、次の表の 2 行目の最初のセルを選択することをシミュレートできます。

CustomerName	FirstOrder	ID	IsActive	CreditCard
Bob Villa	01.01.2008	0	<input checked="" type="checkbox"/>	MasterCard
Brian Miller	02.01.2008	1	<input type="checkbox"/>	Visa
Caral Rudd	03.01.2008	2	<input checked="" type="checkbox"/>	American Ex...
Dan Rundgren	04.01.2008	3	<input type="checkbox"/>	MasterCard
Devie Yingstein	05.01.2008	4	<input checked="" type="checkbox"/>	Visa

セルのテキストを指定すると、次のコード行が生成されます。

テキスト解決メソッドは、次のテクノロジ ドメインでサポートされます。

- Win32
- WPF
- Windows Forms
- Java SWT と Eclipse
- Java AWT/Swing



注: Java アプレット、および Java バージョンが 1.6.10 以下である Swing アプリケーションの場合、テキスト解決は追加設定なしでサポートされます。Java バージョンが 1.6.10 以上の Swing アプリケーションの場合は、アプリケーションの起動時に次のコマンドライン要素を追加する必要があります。

```
-Dsun.java2d.d3d=false
```

例：

```
javaw.exe -Dsun.java2d.d3d=false -jar mySwingApplication.jar
```

- xBrowser

テキスト解決メソッド

次のメソッドにより、コントロールのテキストを操作できます。

- TextCapture** コントロール内のテキストを返します。子コントロールのテキストも返します。
- TextClick** コントロール内の指定テキストをクリックします。テキストが検出されるか、同期オプションで定義できるオブジェクト解決タイムアウトに達するまで待機します。
- TextRectangle** コントロール内の特定テキストの矩形、またはコントロールの領域を返します。
- TextExists** コントロール内またはコントロールの領域内に特定テキストが存在するかどうかを判断します。

テキスト クリックの記録

テキスト クリックの記録を有効にすると、は、相対座標でクリックを記録するのではなく、TextClick メソッドを記録します。通常の座標ベースのクリックよりも TextClick 記録の方が結果が良いコントロールには、この方法を使用します。テキスト クリックが記録されない場合でも、コントロール用にテキスト クリックをスクリプトに挿入できます。

TextClick 操作を記録しない場合は、テキスト クリックの記録をオフに切り替えて通常のクリックを記録できます。

テキスト解決メソッドでは、部分的に一致する単語よりも完全に一致する単語が優先されます。では、完全に一致する単語の前に部分的に一致する単語が画面に表示されていても、部分的に一致する単語よりも完全に一致した単語の出現が先に解決されます。完全に一致する単語がない場合は、部分的に一致する単語が画面に表示される順序で使用されます。


例

ユーザー インターフェイスには、テキスト「*the hostname is the name of the host*」が表示されているとします。画面には「hostname」が「host」より前に表示されていますが、次のコードでは「hostname」ではなく「host」がクリックされます。次のコードでは 2 回目の出現が指定され、単語「hostname」の部分文字列「host」がクリックされます。

カスタム コントロール

Silk4NET では、カスタム コントロールを扱うときに、以下の機能がサポートされます。

- 動的呼び出しを使用すると、テスト対象アプリケーション (AUT) 内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティの取得、またはプロパティの設定を Silk4NET で直接実行できます。
- Win32 ベースのアプリケーションでは、クラス マッピングを true に設定することで、カスタム コントロール クラスの名前を標準 Silk Test クラスの名前にマップできます。このようにすると、標準 Silk Test クラスでサポートされる機能をテストで使用できます。
- **カスタム コントロールの管理** ダイアログ ボックスを使用して、ロケータで利用できるカスタム コントロールの名前を指定したり、カスタム コントロールを操作する再利用可能なコードを作成することができます。

 **注:** カスタム コントロールでは、Click、TextClick、TypeKeys などのメソッドだけが、Silk4NET で記録できます。Adobe Flex アプリケーションをテストする場合を除き、カスタム コントロールのカスタム メソッドは記録できません。

動的呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティの取得、またはプロパティの設定を直接実行できます。また、このコントロールの Silk4NET API で使用できないメソッドおよびプロパティも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4NET API を通じて公開されていない場合に特に便利です。


オブジェクトの動的メソッドは Invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、GetDynamicMethodList メソッドを使用します。


オブジェクトの複数の動的メソッドは InvokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、GetDynamicMethodList メソッドを使用します。


動的プロパティの取得には GetProperty メソッドを、動的プロパティの設定には SetProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、GetPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.Invoke("SetTitle", "my new title")
```

 **注:** 通常、ほとんどのプロパティは読み取り専用で、設定できません。

 **注:** ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティを取得する場合、Reflection を使用します。

 **注:** DOM 要素のメソッドを動的に呼び出すことはできません。

動的呼び出しに関するよくある質問

このセクションでは、カスタム コントロールをテストするために動的にメソッドを呼び出すときの質問を示します。

Invoke メソッドを使用して呼び出せるメソッド

特定のテスト オブジェクトに対して、Invoke メソッドを使用して呼び出せるすべてのメソッドのリストを取得するには、GetDynamicMethodList を使用します。リストを表示するには、コンソールに出力したり、デバッガーで表示することなどができます。

呼び出しで複雑なオブジェクトが返されることが期待されるときに単純な文字列が返される理由

Invoke メソッドは単純なデータ型のみを返すことができます。複雑な型は文字列として返されます。Silk4NET は ToString メソッドを使用して、戻り値の文字列表現を取得します。個々のメソッドを呼び出し、最初のメソッドの呼び出しで返される複雑なオブジェクトのプロパティを読み取るには、Invoke ではなく、InvokeMethods を使用します。

複数の InvokeMethods 呼び出しを使用するときにスクリプトを単純化する方法

スクリプトで大量の InvokeMethods を使用すると、すべてのメソッド名を文字列として渡し、すべてのパラメータをリストとして渡す必要があるため、複雑になります。このような複雑なスクリプトを単純化するには、InvokeMethods を通じてコントロールを操作するのではなく、AUT の実際のコントロールを

操作する静的メソッドを作成します。詳細については、「テスト対象アプリケーションにコードを追加してカスタムコントロールをテストする」を参照してください。

テスト対象アプリケーションにコードを追加してカスタムコントロールをテストする

Windows Forms アプリケーションまたは WPF アプリケーションをテストし、複雑なカスタムコントロールまたは Invoke および InvokeMethods メソッドを使用するだけではテストできないカスタムコントロールをテストする場合は、テスト対象アプリケーション (AUT) の実際のコントロールを操作する静的メソッドを作成し、このコードを AUT に追加できます。

AUT にコードを追加することのメリットは、AUT のコードで、動的呼び出しメソッドによるメソッド呼び出しのリフレクション形式ではなく、通常のメソッド呼び出しを使用してコントロールを操作できるという点です。そのため、コードを作成する時に、コード補完と IntelliSense を使用できます。その後、AUT のコードを単純な呼び出しで呼び出し、該当するコントロールをパラメータとして渡すことができます。

AUT にコードを追加するには、次の方法があります。

- AUT でコードをコンパイルします。実装は簡単ですが、意図しない AUT の変更を行うこととなります。
- テスト スクリプトの LoadAssembly メソッドを使用して、実行時にコードを AUT に挿入します。AUT でコードをコンパイルする場合よりも作業は多くなりますが、挿入されたコードはテストコードの近くに配置されます。LoadAssembly は、クラス WPFWindow および FormsWindow で使用できます。

例 : UltraGrid Infragistics コントロールのテスト

この例では、UltraGrid コントロールの内容を取得する方法を示します。UltraGrid コントロールは、Infragistics が提供する NETAdvantage for Windows Forms ライブラリに含まれています。

UltraGridUtil クラスを作成するには、以下の操作を実行します。

1. C# または VB .NET で、新しいクラス ライブラリを作成します。新しいプロジェクト AUTExtensions を呼び出します。



注: クラス ライブラリは、AUT と同じバージョンの .NET バージョンを使用する必要があります。

2. 必要な依存関係への参照をプロジェクトに追加します。たとえば、Infragistics バージョン 12.2 の場合、次のアセンブリへの参照が必要です。

- Infragistics4.Shared.v12.2
- Infragistics4.Win.UltraWinGrid.v12.2
- Infragistics4.Win.v12.2

AUT で使用している Infragistics のバージョンが不明な場合は、Microsoft の **Process Explorer** ツールを使用して、AUT にロードされているアセンブリを確認できます。

- a. AUTExtensions プロジェクトで、次の内容を持つ新しいクラス UltraGridUtil を作成します。

```
' VB code
Public Class UltraGridUtil

    Public Shared Function GetContents(ultraGrid As
Infragistics.Win.UltraWinGrid.UltraGrid) As List(Of List(Of String))
    Dim contents = New List(Of List(Of String))
    For Each row In ultraGrid.Rows
        Dim rowContents = New List(Of String)
```



```

        For Each cell In row.Cells
            rowContents.Add(cell.Text)
        Next
        contents.Add(rowContents)
    Next
    Return contents
End Function

End Class

// C# code
using System.Collections.Generic;

namespace AUTExtensions {

    public class UltraGridUtil {

        public static List<List<string>>
        GetContents(Infragistics.Win.UltraWinGrid.UltraGrid grid) {
            var result = new List<List<string>>();
            foreach (var row in grid.Rows) {
                var rowContent = new List<string>();
                foreach (var cell in row.Cells) {
                    rowContent.Add(cell.Text);
                }
                result.Add(rowContent);
            }
            return result;
        }
    }
}
}

```



注: Shared 修飾子によって、GetContents メソッドが静的メソッドになります。

3. AUTExtensions プロジェクトを構築します。
4. 再生中に、AUT にアセンブリをロードします。
 - Silk4NET プロジェクトで、既存のテスト スクリプトを開くか、新しいテスト スクリプトを作成します。
 - AUTExtensions プロジェクトを Silk4NET プロジェクトへの参照として追加します。
 - 次のコードをテスト スクリプトに追加します。

```
' VB code
mainWindow.LoadAssembly(GetType(UltraGridUtil).Assembly.Location)
```

```
// C# code
mainWindow.LoadAssembly(typeof(UltraGridUtil).Assembly.Location);
```

5. 注入したコードの静的メソッドを呼び出して、UltraGrid の内容を取得します。

```
'VB code
Dim ultraGrid = mainWindow.Control("@automationId='my grid'")
Dim contents As IList =
mainWindow.Invoke("AUTExtensions.UltraGridUtil.GetContents", ultraGrid)
```

```
// C# code
Dim ultraGrid = mainWindow.Control("@automationId='my grid'");
Dim contents As IList =
mainWindow.Invoke("AUTExtensions.UltraGridUtil.GetContents", ultraGrid);
```

AUT へのコードの追加に関するよくある質問

このセクションでは、カスタム コントロールをテストするために AUT にコードを追加するときの質問を示します。

LoadAssembly メソッドを使用して AUT に挿入したコードが AUT で更新されない理由

AUT 内のコードが、LoadAssembly メソッドを使用して AUT に挿入したコードによって置き換えられない場合、アセンブリがすでに AUT にロードされている可能性があります。アセンブリをアンロードすることはできないため、AUT を閉じてから、再開する必要があります。

メソッドを呼び出すと入力引数の型が一致しない理由

何らかのメソッドを呼び出したときに、入力引数の型が一致しないことを示すエラーが表示される場合は、呼び出すメソッドは見つかりましたが、引数が正しくありません。スクリプトで正しいデータ型を使用していることを確認します。

スクリプトで LoadAssembly メソッドを使用してアセンブリを AUT にロードする場合にこのエラーが発生するもう 1 つの理由として、AUT が使用するバージョンとは異なるサードパーティ ライブラリのバージョンに対してアセンブリが作成されている可能性があります。この問題を修正するには、プロジェクトで参照されているアセンブリを変更します。AUT で使用されているサードパーティ ライブラリのバージョンが不明な場合は、Microsoft の **Process Explorer** ツールを使用できます。

アセンブリをコピーできないときにコンパイル エラーを修正する方法

LoadAssembly メソッドで AUT にコードを追加しようとしたときに、次のコンパイル エラーが発生することがあります。

```
Could not copy '<assembly_name>.dll' to '<assembly_name>.dll'. The process cannot access the file.
```

このコンパイル エラーは、アセンブリがすでに AUT にロードされていて、上書きできないために発生します。

このコンパイル エラーを修正するには、AUT を閉じて、再度スクリプトをコンパイルします。

Adobe Flex カスタム コントロールのテスト

Silk4NET では、Flex カスタム コントロールのテストがサポートされています。デフォルトで、Silk4NET では、カスタム コントロールの個別のサブコントロールに対する記録および再生のサポートが提供されません。

カスタム コントロールをテストする場合、以下のオプションが存在します。

- 基本サポート

基本サポートでは、動的呼び出しを使用して、再生中にカスタム コントロールと対話します。作業量が少なく済むこのアプローチは、テスト アプリケーションにおいて、Silk4NET が公開しないカスタム コントロールのプロパティおよびメソッドにアクセスする場合に使用します。カスタム コントロールの開発者は、コントロールのテストを容易にすることのみを目的としたメソッドおよびプロパティをカスタム コントロールに追加することもできます。ユーザーは、動的呼び出し機能を使用してこれらのメソッドやプロパティを呼び出すことができます。

基本サポートには以下のような利点があります。

- 動的呼び出しでは、テスト アプリケーションのコードを変更する必要がありません。
- 動的呼び出しを使用することによって、ほとんどのテストのニーズを満たすことができます。

基本サポートには以下のような短所があります。

- ロケーターには、具体的なクラス名が組み込まれません (たとえば、Silk4NET では「// FlexSpinner」ではなく「//FlexBox」と記録されます)。

- 記録のサポートが限定されます。
- Silk4NET では、イベントを再生できません。

例を含む動的呼び出しの詳細については、「Flex メソッドの動的呼び出し」を参照してください。

• 高度なサポート

高度なサポートでは、カスタム コントロールに対して、特定のオートメーション サポートを作成できます。この追加のオートメーション サポートによって、記録のサポートおよびより強力な再生のサポートが提供されます。高度なサポートには以下のような利点があります。

- イベントの記録と再生を含む、高レベルの記録および再生のサポートが提供されます。
- Silk4NET では、カスタム コントロールが他のすべての組み込み Flex コントロールと同様に処理されます。
- Silk4NET API とシームレスに統合できます。
- Silk4NET では、ローケーターで具体的なクラス名が使用されます (たとえば、Silk4NET では「// FlexSpinner」と記録されます)。

高度なサポートには以下のような短所があります。

- 実装作業が必要です。テスト アプリケーションを変更し、Open Agent を拡張する必要があります。

カスタム コントロールの管理

Silk4NET が専用サポートを提供していないカスタム コントロールに対応するカスタム クラスを作成できます。カスタム クラスを作成すると、以下の利点があります。

- スクリプトのローケーターが効率化されます。
- カスタム コントロールと対話するための再利用可能コードを簡単に記述できます。

例 : UltraGrid Infragistics コントロールのテスト

カスタム グリッド コントロールが Silk4NET で汎用クラス Control として認識されるとします。Silk4NET のカスタム コントロール サポートを使用すると、以下の利点があります。

カスタム コントロール クラス名をローケーターで使用できるため、オブジェクトの認識率が高まります。

複数のオブジェクトが Control として認識されることがあります。ローケーターには、特定のオブジェクトを識別するためのインデックスが必要です。たとえば、オブジェクトはローケーター // Control[13] を使用して識別できます。このコントロールのカスタム クラス (クラス UltraGrid など) を作成する場合は、ローケーター //UltraGrid を使用できます。カスタム クラスを作成することによって、テスト対象アプリケーションが変更された場合にオブジェクト識別子が変わりやすい、大きな数字のインデックスを使用する必要がなくなります。

スクリプト内のコントロールに、再利用可能な再生操作を実行できます。

カスタム クラスを使用している場合、ユーザー インターフェイスにカスタム コントロールを指定すると生成されるクラスであるカスタム クラスに以下のコードを追加することで、グリッドのコンテンツをメソッド内に取り込む動作をカプセル化できます。

通常は、以下のいずれかの方法で、メソッドをカスタム コントロール クラスに実装できます。

- Click、TypeKeys、TextClick、および TextCapture などのメソッドを使用できます。
- AUT のオブジェクトで動的にメソッドを呼び出せます。
- AUT に追加したメソッドを動的に呼び出せません。これは、この例で説明されている手法です。

以下のコードを使用して、「テスト対象アプリケーションにコードを追加してカスタム コントロールをテストする」の例で定義されている静的メソッドを呼び出すことができます。メソッド GetContents が、生成されたクラス UltraGrid に追加されます。

```
' VB code
Partial Public Class UltraGrid

    Public Function GetContents() As IList
        Return
        Invoke("AUTExtensions.UltraGridUtil.GetContents", Me)
    End Function

End Class
```

```
// C# code
public partial class UltraGrid {

    public System.Collections.IList
    GetContents() {
        return (System.Collections.IList)
        Invoke("AUTExtensions.UltraGridUtil.GetContents", this);
    }

}
```

クラスをカスタム コントロールとして定義すると、Dialog クラスのように、すべての組み込みクラスの場合と同じ方法でそのクラスを使用できます。

```
' VB code
Dim ultraGrid As UltraGrid =
mainWindow.UltraGrid("@automationId='my
grid'")
Dim contents = ultraGrid.GetContents()
```

```
// C# code
UltraGrid ultraGrid =
mainWindow.UltraGrid("@automationId='my
grid'");
IList contents = ultraGrid.GetContents();
```

カスタム コントロールのサポート

Silk4NET が専用サポートを提供していないカスタム コントロールに対応するカスタム クラスを作成するには、以下を実行します。

1. Silk4NET > **カスタムコントロールの管理** をクリックします。 **カスタムコントロールの管理** ダイアログ ボックスが開きます。
2. Silk4NET **カスタム コントロール コードのディレクトリ** フィールドで、任意の名前を入力するか、 **参照** をクリックして、カスタム コントロールを含めるディレクトリを選択します。
3. 新しいカスタム クラスを作成するテクノロジー ドメインのタブをクリックします。
4. **追加** をクリックします。
5. 次のいずれかをクリックします。
 - **新しいカスタム コントロールの識別** をクリックし、 **オブジェクトの識別** ダイアログ ボックスを使ってアプリケーション内のカスタム コントロールを直接選択します。
 - **新しいカスタム コントロールの追加** をクリックし、カスタム コントロールを手動でリストに追加します。

新しい行がカスタム コントロールのリストに追加されます。

6. カスタム コントロールを手動でリストに追加するように選択した場合は、以下を実行します。
 - a) **Silk Test 基本クラス** 列で、クラスの取得元となる既存の基本クラスを選択します。
このクラスは、ご使用のカスタム コントロールのタイプに最も一致率が高くなければなりません。
 - b) **Silk Test クラス** 列で、クラスの参照に使用する名前を入力します。
この名前は、ロケーターに表示されます。たとえば、//Control[13] でなく //UltraGrid を入力します。
7. Win32 アプリケーションの場合のみ：**クラス マッピング** 列で、クラス マッピングを有効に設定し、カスタム コントロール クラスの名前を標準 Silk Test クラスの名前にマップします。
カスタム コントロール クラスを標準 Silk Test クラスにマップすると、テストの際に標準 Silk Test クラスでサポートされている機能を使用できます。
8. **OK** をクリックします。
9. スクリプトの場合のみ：
 - a) カスタム コントロール用のクラスにカスタム メソッドおよびプロパティを追加します。
 - b) スクリプト内で新しいクラスのカスタム メソッドおよびプロパティを使用します。



注: 有効なクラスを追加すると、そのクラスは **Silk Test 基本クラス** リストで使用できるようになります。追加したクラスは、基本クラスとして再使用できます。



注: カスタム メソッドおよびプロパティは記録されません。



注: スクリプト ファイル内のカスタム クラスまたは基本クラスの名前を変更しないでください。スクリプト内に生成されたクラスを変更した場合、予期しない動作を起こすことがあります。カスタム クラスにプロパティおよびメソッドを追加する場合にのみスクリプトを使用してください。それ以外の変更をカスタム クラスに加える場合は **カスタム コントロールの管理** ダイアログ ボックスを使用してください。


カスタム コントロール オプション

Silk4NET > カスタム コントロールの管理

Silk4NET **カスタム コントロール コードのディレクトリ** で、新しいカスタム クラスをその中に生成するディレクトリを定義します。

次の **カスタム コントロール** オプションが使用できます。

オプション	説明
Silk Test 基本クラス	自分のクラスの派生元として使用する既存の基本クラスを選択します。このクラスは、ご使用のカスタムコントロールのタイプに最も一致率が高くなければなりません。
Silk Test クラス	クラスの参照に使用する名前を入力します。この名前は、ロケーターに表示されます。
カスタムコントロールのクラス名	マッピングされているクラスの完全修飾クラス名を入力します。クラス名には、ワイルドカード ? および * を使用できます。
クラス マッピング	このオプションは Win32 アプリケーションの場合のみ使用できます。カスタムコントロール クラスの名前を標準 Silk Test クラスの名前にマップするには、クラス マッピングを true に設定します。カスタムコントロール クラスを標準 Silk Test クラスにマップすると、テストの際に標準 Silk Test クラスでサポートされている機能を使用できます。

 **注:** 有効なクラスを追加すると、そのクラスは **Silk Test 基本クラス** リストで使用できるようになります。追加したクラスは、基本クラスとして再使用できます。

例 : UltraGrid Infragistics コントロールのオプションの設定

UltraGrid Infragistics コントロールをサポートするには、次の値を使用します。

オプション	値
Silk Test 基本クラス	Control
Silk Test クラス	UltraGrid
カスタム コントロールのクラス名	Infragistics.Win.UltraWinGrid. UltraGrid

特定の環境のテスト

Silk4NET では、複数の種類の環境でのテストがサポートされています。


Adobe Flex のサポート

Silk4NET は、Internet Explorer、Mozilla Firefox、スタンドアロンの Flash Player を使用した Adobe Flex アプリケーション、および Adobe Flex 4.x 以降で作成した Adobe AIR アプリケーションのテストを組み込みでサポートしています。

Silk4NET では、Adobe Flex 3.x および 4.x アプリケーションにおいて複数のアプリケーション ドメインもサポートされているため、サブアプリケーションをテストできます。Silk4NET では、ロケーター階層ツリーの各サブアプリケーションが、関連するアプリケーション ドメイン コンテキストを持つアプリケーション ツリーとして認識されます。Adobe Flex 4.x サブアプリケーションでは、ロケーター属性テーブルのルートレベルで SparkApplication クラスが使用されます。Adobe Flex 3.x サブアプリケーションでは、FlexApplication クラスが使用されます。

サポートするコントロール

Adobe Flex のテストで記録および再生できるコントロールの完全なリストについては、「Flex クラス リファレンス」を参照してください。

 **注:** Silk Test Flex オートメーション SDK は、Adobe Flex のオートメーション API に基づいていません。Silk Test オートメーション SDK は、Adobe Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、Flex オートメーション API の typekey ステートメントでは、すべてのキーはサポートされません。テキスト入力ステートメントを使用してこの問題を解決できます。Flex オートメーション API の詳細については、『Adobe Flex リリース ノート』を参照してください。

Adobe Flash Player で実行するための Flex アプリケーションの構成

Adobe Flex アプリケーションを Flash Player で実行するには、以下のいずれか、または両方の条件が満たされている必要があります。

- Flex アプリケーションを作成する開発者は、アプリケーションを EXE ファイルとしてコンパイルする必要があります。アプリケーションは、ユーザーが起動すると、Flash Player で開きます。Windows Flash Player は、<http://www.adobe.com/support/flashplayer/downloads.html> からインストールします。
 - ユーザーが、Windows Flash Player Projector をインストールしている必要があります。ユーザーは、Flex の .SWF ファイルを開いた場合に Flash Player で開くように構成できます。Adobe Flex 開発者スイートをインストールしないと、Flash Player をインストールしても Windows Flash Projector はインストールされません。Windows Flash Projector は、<http://www.adobe.com/support/flashplayer/downloads.html> からインストールします。
1. Windows 7 および Windows 2008 R2 では、管理者として実行されるように Flash Player を構成します。以下のステップを実行します。
 - a) Adobe Flash Player プログラム ショートカットまたは FlashPlayer.exe ファイルを右クリックして、**プロパティ** をクリックします。
 - b) **プロパティ** ダイアログ ボックスで、**互換性** タブをクリックします。

- c) **管理者としてこのプログラムを実行する** チェック ボックスをオンにして、**OK** をクリックします。
2. コマンド プロンプト (cmd.exe) で以下のコマンドを入力して、Flash Player で .SWF ファイルを起動します。

```
"<Application_Install_Directory>%ApplicationName.swf"
```

デフォルトで、<SilkTest_Install_Directory> は Program Files¥Silk¥Silk Test にあります。

Component Explorer の起動

Silk Test には、Component Explorer というサンプルの Adobe Flex アプリケーションが含まれています。Component Explorer は、Adobe オートメーション SDK および Silk Test 固有のオートメーション実装を使用してコンパイルされており、テスト用に事前に構成されています。

Internet Explorer で、<http://demo.borland.com/flex/SilkTest14.0/index.html> を開きます。デフォルトブラウザでアプリケーションが起動します。

Adobe Flex アプリケーションのテスト

Silk Test は、Adobe Flex アプリケーションのテストを組み込みでサポートしています。Silk Test では、いくつかのサンプル Adobe Flex アプリケーションを提供しています。サンプル アプリケーションには、<http://demo.borland.com/flex/SilkTest14.0/index.html> からアクセスできます。


新機能、サポート対象のプラットフォームとバージョン、既知の問題、および回避策の詳細については、『Silk Test リリース ノート』(<http://supportline.microfocus.com/productdoc.aspx> で入手可能) を参照してください。

独自の Adobe Flex アプリケーションをテストする前に、Adobe Flex 開発者は以下のステップを実行する必要があります。

- Adobe Flex アプリケーションのテストの有効化
- テスト可能な Adobe Flex アプリケーションの作成
- Adobe Flex コンテナのコーディング
- カスタム コントロールのオートメーション サポートの実装

独自の Adobe Flex アプリケーションをテストするには、以下のステップを実行します。

- ローカルの Flash Player のセキュリティ設定の構成
- テストの記録
- テストの再生
- Adobe Flex スクリプトのカスタマイズ
- カスタム Adobe Flex コントロールのテスト

 **注:** Adobe Flex アプリケーションを読み込み、Flex オートメーション フレームワークを初期化する
とき、テストを実行するマシンおよび Adobe Flex アプリケーションの複雑度に応じて、多少の時間
がかかる場合があります。アプリケーションが完全に読み込まれるように、ウィンドウのタイムアウト
値を高い値に設定します。

Adobe Flex カスタム コントロールのテスト

Silk4NET では、Flex カスタム コントロールのテストがサポートされています。デフォルトで、Silk4NET
では、カスタム コントロールの個別のサブコントロールに対する記録および再生のサポートが提供されま
す。

カスタム コントロールをテストする場合、以下のオプションが存在します。

- 基本サポート

基本サポートでは、動的呼び出しを使用して、再生中にカスタム コントロールと対話します。作業量
が少なく済むこのアプローチは、テスト アプリケーションにおいて、Silk4NET が公開しないカスタム

コントロールのプロパティおよびメソッドにアクセスする場合に使用します。カスタムコントロールの開発者は、コントロールのテストを容易にすることのみを目的としたメソッドおよびプロパティをカスタムコントロールに追加することもできます。ユーザーは、動的呼び出し機能を使用してこれらのメソッドやプロパティを呼び出すことができます。

基本サポートには以下のような利点があります。

- 動的呼び出しでは、テストアプリケーションのコードを変更する必要がありません。
- 動的呼び出しを使用することによって、ほとんどのテストのニーズを満たすことができます。

基本サポートには以下のような短所があります。

- ロケーターには、具体的なクラス名が組み込まれません (たとえば、Silk4NET では「// FlexSpinner」ではなく「//FlexBox」と記録されます)。
- 記録のサポートが限定されます。
- Silk4NET では、イベントを再生できません。

例を含む動的呼び出しの詳細については、「Flex メソッドの動的呼び出し」を参照してください。

- 高度なサポート

高度なサポートでは、カスタムコントロールに対して、特定のオートメーションサポートを作成できます。この追加のオートメーションサポートによって、記録のサポートおよびより強力な再生のサポートが提供されます。高度なサポートには以下のような利点があります。


- イベントの記録と再生を含む、高レベルの記録および再生のサポートが提供されます。
- Silk4NET では、カスタムコントロールが他のすべての組み込み Flex コントロールと同様に処理されます。
- Silk4NET API とシームレスに統合できます。
- Silk4NET では、ロケーターで具体的なクラス名が使用されます (たとえば、Silk4NET では「// FlexSpinner」と記録されます)。

高度なサポートには以下のような短所があります。

- 実装作業が必要です。テストアプリケーションを変更し、Open Agent を拡張する必要があります。

Flex メソッドの動的呼び出し

動的呼び出し機能を使用して Silk4NET が対象としないコントロールのメソッドを呼び出したり、プロパティを取得/設定することができます。この機能は、カスタムコントロールを使用したり、カスタマイズせずに Silk4NET がサポートするコントロールを使用する場合に有効です。

 **注:** 通常、ほとんどのプロパティは読み取り専用で、設定できません。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4NET がサポートするコントロールのメソッドとプロパティ。
- Flex API で定義されているすべてのパブリック メソッド
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティ。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4NET 型

Silk4NET 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point など) が含まれます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4NET 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、C# では null が、VB では Nothing が返されます。

テスト アプリケーションでのカスタム コントロールの定義

通常、テスト アプリケーションには、アプリケーションの開発中に追加されたカスタム コントロールがすでに含まれています。テスト アプリケーションにすでにカスタム コントロールが含まれている場合は、「動的呼び出しを使用して Flex カスタム コントロールをテストする」または「オートメーション サポートを使用してカスタム コントロールをテストする」に進んでください。

この手順では、Flex アプリケーション開発者が Flex で Spinner カスタム コントロールを作成する方法を示します。このトピックで作成する Spinner カスタム コントロールは、カスタム コントロールの実装およびテストのプロセスを説明するために、いくつかのトピックで使用されています。

Spinner カスタム コントロールは、以下のグラフィックに示すように、2 つのボタンと 1 つのテキスト フィールドを含んでいます。



ユーザーは、**Down** をクリックしてテキスト フィールドに表示されている値を 1 減分させ、**Up** をクリックしてテキスト フィールドの値を 1 増分させることができます。

カスタム コントロールには、設定および取得が可能なパブリックの CurrentValue プロパティが用意されています。

1. テスト アプリケーションで、コントロールのレイアウトを定義します。
たとえば、Spinner コントロール タイプでは、以下のように記述します。

```
<?xml version="1.0" encoding="utf-8"?>
<customcontrols:SpinnerClass xmlns:mx="http://www.adobe.com/2006/mxml"
xmlns:controls="mx.controls.*" xmlns:customcontrols="customcontrols.*">
  <controls:Button id="downButton" label="Down" />
  <controls:TextInput id="text" enabled="false" />
  <controls:Button id="upButton" label="Up"/>
</customcontrols:SpinnerClass>
```

2. カスタム コントロールの実装を定義します。
たとえば、Spinner コントロール タイプでは、以下のように記述します。

```
package customcontrols
{
    import flash.events.MouseEvent;

    import mx.containers.HBox;
    import mx.controls.Button;
    import mx.controls.TextInput;
    import mx.core.UIComponent;
    import mx.events.FlexEvent;

    [Event(name="increment", type="customcontrols.SpinnerEvent")]
    [Event(name="decrement", type="customcontrols.SpinnerEvent")]

    public class SpinnerClass extends HBox
    {
        public var downButton : Button;
        public var upButton : Button;
```

```

public var text : TextInput;
public var ssss: SpinnerAutomationDelegate;
private var _lowerBound : int = 0;
private var _upperBound : int = 5;

private var _value : int = 0;
private var _stepSize : int = 1;

public function SpinnerClass() {
    addEventListener(FlexEvent.CREATION_COMPLETE,
creationCompleteHandler);
}

private function creationCompleteHandler(event:FlexEvent) : void {
    downButton.addEventListener(MouseEvent.CLICK, downButtonClickHandler);
    upButton.addEventListener(MouseEvent.CLICK, upButtonClickHandler);
    updateText();
}

private function downButtonClickHandler(event : MouseEvent) : void {
    if(currentValue - stepSize >= lowerBound) {
        currentValue = currentValue - stepSize;
    }
    else {
        currentValue = upperBound - stepSize + currentValue - lowerBound
+ 1;
    }

    var spinnerEvent : SpinnerEvent = new
SpinnerEvent(SpinnerEvent.DECREMENT);
    spinnerEvent.steps = _stepSize;
    dispatchEvent(spinnerEvent);
}

private function upButtonClickHandler(event : MouseEvent) : void {
    if(currentValue <= upperBound - stepSize) {
        currentValue = currentValue + stepSize;
    }
    else {
        currentValue = lowerBound + currentValue + stepSize -
upperBound - 1;
    }

    var spinnerEvent : SpinnerEvent = new
SpinnerEvent(SpinnerEvent.INCREMENT);
    spinnerEvent.steps = _stepSize;
    dispatchEvent(spinnerEvent);
}

private function updateText() : void {
    if(text != null) {
        text.text = _value.toString();
    }
}

public function get currentValue() : int {
    return _value;
}

```

```

public function set currentValue(v : int) : void {
    _value = v;
    if(v < lowerBound) {
        _value = lowerBound;
    }
    else if(v > upperBound) {
        _value = upperBound;
    }
    updateText();
}

public function get stepSize() : int {
    return _stepSize;
}

public function set stepSize(v : int) : void {
    _stepSize = v;
}

public function get lowerBound() : int {
    return _lowerBound;
}

public function set lowerBound(v : int) : void {
    _lowerBound = v;
    if(currentValue < lowerBound) {
        currentValue = lowerBound;
    }
}

public function get upperBound() : int {
    return _upperBound;
}

public function set upperBound(v : int) : void {
    _upperBound = v;
    if(currentValue > upperBound) {
        currentValue = upperBound;
    }
}
}
}
}

```

- 3.** コントロールが使用するイベントを定義します。
 たとえば、Spinner コントロール タイプでは、以下のように記述します。

```

package customcontrols
{
    import flash.events.Event;

    public class SpinnerEvent extends Event
    {
        public static const INCREMENT : String = "increment";
        public static const DECREMENT : String = "decrement";

        private var _steps : int;

        public function SpinnerEvent(eventName : String) {

```

```

        super(eventName);
    }

    public function set steps(value:int) : void {
        _steps = value;
    }

    public function get steps() : int {
        return _steps;
    }
}

```

次のステップでは、テスト アプリケーションのオートメーション サポートを実装します。

動的呼び出しを使用した Flex カスタム コントロールのテスト

Silk4NET では、動的呼び出しを使用したカスタム コントロールの記録と再生のサポートが提供されており、これにより再生中にカスタム コントロールを操作できます。作業量が少なく済むこのアプローチは、テスト アプリケーションにおいて、Silk4NET が公開しないカスタム コントロールのプロパティおよびメソッドにアクセスする場合に使用します。カスタム コントロールの開発者は、コントロールのテストを容易にすることのみを目的としたメソッドおよびプロパティをカスタム コントロールに追加することもできます。

1. コントロールでサポートされている動的メソッドのリストを取得するには、GetDynamicMethodList メソッドを使用します。
2. オブジェクトの動的メソッドは Invoke メソッドを使用して呼び出します。
3. オブジェクトの複数の動的メソッドは InvokeMethods メソッドを使用して呼び出します。
4. コントロールでサポートされている動的プロパティのリストを取得するには、GetPropertyList メソッドを使用します。
5. 動的プロパティの取得には GetProperty メソッドを、動的プロパティの設定には SetProperty メソッドを使用します。

例

この例では、以下の図に示すように、2 つのボタンと 1 つのテキスト フィールドを含む Spinner カスタム コントロールをテストします。



ユーザーは、**Down** をクリックしてテキスト フィールドに表示されている値を 1 減分させ、**Up** をクリックしてテキスト フィールドの値を 1 増分させることができます。

カスタム コントロールには、設定および取得が可能なパブリックの CurrentValue プロパティが用意されています。

Spinner の値を 4 に設定するには、以下のように入力します。

```

Dim spinner = Desktop.Find("//
FlexBox[@className=customcontrols.Spinner]")
spinner.SetProperty("CurrentValue", 4)

```

オートメーション サポートを使用したカスタム コントロールのテスト

カスタム コントロールに対して、特定のオートメーション サポートを作成できます。この追加のオートメーション サポートによって、記録のサポートおよびより強力な再生のサポートが提供されます。オートメ

ーション サポートを作成するには、テスト アプリケーションを変更し、Open Agent を拡張する必要があります。

Silk4NET でカスタム コントロールをテストする前に、以下のステップを実行します。

- テスト アプリケーションでのカスタム コントロールの定義
- オートメーション サポートの実装

テスト アプリケーションを変更してオートメーション サポートを組み込んだあと、以下のステップを実行します。

スクリプトの場合は、スクリプトを記録して、カスタム コントロールに合わせて手動で変更を加えます。

たとえば、以下のコードでは、オートメーションの委譲に実装されている「Increment」メソッドを使用して、Spinner の値を 3 増分する方法を示しています。

```
_desktop.TestObject("//FlexSpinner[@caption='index:1']").Invoke("Increment", 3)
```

以下の例は、Spinner の値を 3 に設定する方法を示しています。

```
_desktop.TestObject("//FlexSpinner[@caption='index:1']").SetProperty("CurrentValue", 3)
```

カスタム コントロールのオートメーション サポートの実装

カスタム コントロールをテストする前に、カスタム コントロールの ActionScript でオートメーション サポート（オートメーションの委譲）を実装し、テスト アプリケーションにコンパイルします。

以下の手順では、Flex のカスタム Spinner コントロールを使用して、カスタム コントロールのオートメーション サポートの実装方法を示します。Spinner カスタム コントロールは、以下のグラフィックに示すように、2 つのボタンと 1 つのテキスト フィールドを含んでいます。



ユーザーは、**Down** をクリックしてテキスト フィールドに表示されている値を 1 減分させ、**Up** をクリックしてテキスト フィールドの値を 1 増分させることができます。

カスタム コントロールには、設定および取得が可能なパブリックの CurrentValue プロパティが用意されています。

1. カスタム コントロールの ActionScript でオートメーション サポート（オートメーションの委譲）を実装します。

オートメーションの委譲の実装の詳細については、Adobe Live ドキュメント (http://livedocs.adobe.com/flex/3/html/help.html?content=functest_components2_14.html) を参照してください。

この例では、オートメーションの委譲によって、「increment」および「decrement」メソッドに対してサポートが追加されます。オートメーションの委譲のコード例は以下のとおりです。

```
package customcontrols
{
    import flash.display.DisplayObject;
    import mx.automation.Automation;
    import customcontrols.SpinnerEvent;
    import mx.automation.delegates.containers.BoxAutomationImpl;
    import flash.events.Event;
    import mx.automation.IAutomationObjectHelper;
    import mx.events.FlexEvent;
    import flash.events.IEventDispatcher;
    import mx.preloaders.DownloadProgressBar;
    import flash.events.MouseEvent;
    import mx.core.EventPriority;
```

```

[Mixin]
public class SpinnerAutomationDelegate extends BoxAutomationImpl
{
    public static function init(root:DisplayObject) : void {
        // register delegate for the automation
        Automation.registerDelegateClass(Spinner, SpinnerAutomationDelegate);
    }

    public function SpinnerAutomationDelegate(obj:Spinner) {
        super(obj);
        // listen to the events of interest (for recording)
        obj.addEventListener(SpinnerEvent.DECREMENT, decrementHandler);
        obj.addEventListener(SpinnerEvent.INCREMENT, incrementHandler);
    }

    protected function decrementHandler(event : SpinnerEvent) : void {
        recordAutomatableEvent(event);
    }

    protected function incrementHandler(event : SpinnerEvent) : void {
        recordAutomatableEvent(event);
    }

    protected function get spinner() : Spinner {
        return uiComponent as Spinner;
    }

    //-----
    //  override functions
    //-----

    override public function get automationValue():Array {
        return [ spinner.currentValue.toString() ];
    }

    private function replayClicks(button : IEventDispatcher, steps : int) : Boolean
    {
        var helper : IAutomationObjectHelper =
Automation.automationObjectHelper;
        var result : Boolean;
        for(var i:int; i < steps; i++) {
            helper.replayClick(button);
        }
        return result;
    }

    override public function replayAutomatableEvent(event:Event):Boolean {
        if(event is SpinnerEvent) {
            var spinnerEvent : SpinnerEvent = event as SpinnerEvent;
            if(event.type == SpinnerEvent.INCREMENT) {
                return replayClicks(spinner.upButton, spinnerEvent.steps);
            }
            else if(event.type == SpinnerEvent.DECREMENT) {
                return replayClicks(spinner.downButton, spinnerEvent.steps);
            }
            else {

```

```

        return false;
    }
}
else {
    return super.replayAutomatableEvent(event);
}
}

// do not expose the child controls (i.e the buttons and the textfield) as
individual controls
override public function get numAutomationChildren():int {
    return 0;
}
}
}
}

```

2. Open Agent にオートメーションの委譲を導入するために、カスタム コントロールを記述する XML ファイルを作成します。

クラス定義ファイルには、インストルメント化されたすべての Flex コンポーネントについての情報が含まれています。このファイルでは、記録中にイベントを送信でき、再生中にイベントを受け取ることができるコンポーネントについての情報が提供されます。クラス定義ファイルには、サポートされているプロパティの定義も含まれています。

Spinner カスタム コントロールの XML ファイルは以下のようになります。

```

<?xml version="1.0" encoding="UTF-8"?>
<TypeInfoInformation>
  <ClassInfo Name="FlexSpinner" Extends="FlexBox">
    <Implementation
      Class="customcontrols.Spinner" />
    <Events>
      <Event Name="Decrement">
        <Implementation
          Class="customcontrols.SpinnerEvent"
          Type="decrement" />
        <Property Name="steps">
          <PropertyType Type="integer" />
        </Property>
      </Event>
      <Event Name="Increment">
        <Implementation
          Class="customcontrols.SpinnerEvent"
          Type="increment" />
        <Property Name="steps">
          <PropertyType Type="integer" />
        </Property>
      </Event>
    </Events>
    <Properties>
      <Property Name="lowerBound" accessType="read">
        <PropertyType Type="integer" />
      </Property>
      <Property Name="upperBound" accessType="read">
        <PropertyType Type="integer" />
      </Property>
      <!-- expose read and write access for the currentValue property -->
      <Property Name="currentValue" accessType="both">
        <PropertyType Type="integer" />
      </Property>
    </Properties>
  </ClassInfo>
</TypeInfoInformation>

```



```

        <Property Name="stepSize" accessType="read">
            <PropertyType Type="integer" />
        </Property>
    </Properties>
</ClassInfo>
</TypeInfo>

```

3. サポートされている Flex コントロールのすべてのクラス、およびそのメソッドとプロパティを記述するすべての XML ファイルが格納されるフォルダに、カスタム コントロールの XML ファイルを配置します。

Silk Test には、サポートされている Flex コントロールのすべてのクラス、およびそのメソッドとプロパティを記述するいくつかの XML ファイルが含まれています。これらの XML ファイルは <Silk Test_install_directory>%ng%agent%plugins %com.borland.fastxd.techdomain.flex.agent_<version>%config%automationEnvironment フォルダ内にあります。

独自の XML ファイルを提供する場合は、XML ファイルをこのフォルダにコピーする必要があります。Open Agent が起動して、Adobe Flex のサポートを初期化する場合、このディレクトリの内容が読み込まれます。

Flex の Spinner サンプル コントロールをテストするには、CustomControls.xml ファイルをこのフォルダにコピーする必要があります。Open Agent が現在実行されている場合は、ファイルをフォルダにコピーしたあと、Open Agent を再起動します。

Flex クラス定義ファイル

クラス定義ファイルには、インストール化されたすべての Flex コンポーネントについての情報が含まれています。このファイルでは、記録中にイベントを送信でき、再生中にイベントを受け取ることができるコンポーネントについての情報が提供されます。クラス定義ファイルには、サポートされているプロパティの定義も含まれています。

Silk Test には、Flex の共通コントロールおよび特殊化されたコントロールのすべてのクラス、イベント、およびプロパティを記述するいくつかの XML ファイルが含まれています。これらの XML ファイルは <Silk Test_install_directory>%ng%agent%plugins %com.borland.fastxd.techdomain.flex.agent_<version>%config%automationEnvironment フォルダ内にあります。

独自の XML ファイルを提供する場合は、XML ファイルをこのフォルダにコピーする必要があります。Silk Test Agent が起動して Adobe Flex のサポートを初期化するとき、このディレクトリの内容が読み込まれます。

XML ファイルの基本的な構造は以下のとおりです。

```

<TypeInfo>
<ClassInfo>
<Implementation />
<Events>
<Event />
...
</Events>
<Properties>
<Property />
...

```

```
</Properties>
```

```
</ClassInfo>
```

```
</TypeInfoInformation>
```

Adobe Flex スクリプトのカスタマイズ

手動で Flex スクリプトをカスタマイズできます。Flex オブジェクトのプロパティに対して Verify 関数を使用して、手動で検証を挿入できます。各 Flex オブジェクトには、検証可能な一連のプロパティがあります。検証に使用できるプロパティのリストについては、「Flex クラス リファレンス」を参照してください。

1. Flex アプリケーションのテストを記録します。
2. カスタマイズするスクリプト ファイルを開きます。
3. 追加するコードを手動で入力します。

同一 Web ページ上の複数の Flex アプリケーションのテスト

同じ Web ページに複数の Flex アプリケーションが存在する場合、Silk4NET は、Flex アプリケーションの ID またはアプリケーションの size プロパティを使用して、テスト対象アプリケーションを特定します。同じページに複数のアプリケーションが存在し、それらのサイズが異なる場合、Silk4NET は、size プロパティを使用して操作実行対象のアプリケーションを特定します。追加の操作は必要ありません。

以下の場合、Silk4NET は、JavaScript を使用して Flex アプリケーションの ID を検索し、操作実行対象のアプリケーションを特定します。

- 単一の Web ページ上に複数の Flex アプリケーションが存在する場合。
- これらのアプリケーションのサイズが同じである場合。



注: この場合、ブラウザ マシンで JavaScript が有効になっていないと、スクリプト実行時にエラーが発生します。

1. JavaScript を有効にします。
2. Internet Explorer では、以下のステップを実行します。
 - a) ツール > インターネット オプション を選択します。
 - b) セキュリティ タブをクリックします。
 - c) レベルのカスタマイズ をクリックします。
 - d) スクリプト作成 セクションの アクティブ スクリプト で、有効にする をクリックして OK をクリックします。
3. 「Adobe Flex アプリケーションのテスト」の手順に従います。



注: Web ページにフレームが存在し、アプリケーションが同じサイズである場合、この方法は動作しません。

Adobe AIR のサポート

Silk4NET がサポートする Adobe AIR でのテストは、Flex 4 コンパイラを使用してコンパイルされたアプリケーションのみです。サポートされているバージョンの詳細については、リリース ノートで最新の情報を確認してください。

Silk Test には、サンプルの Adobe AIR アプリケーションが含まれています。 <http://demo.borland.com/flex/SilkTest14.0/index.html> にあるサンプル アプリケーションにアクセスして、使用する Adobe AIR アプリケーションをクリックしてください。オートメーションあり、またはオート

メーシオンなしのアプリケーションを選択できます。AIR アプリケーションを実行するには、Adobe AIR ランタイムをインストールする必要があります。

名前またはインデックスを使用する Flex の Select メソッドの概要

Flex の Select メソッドは、選択するコントロールの Name または Index を使用して記録できます。デフォルトで、Silk4NET では、コントロールの名前を使用して Select メソッドが記録されます。ただし、コントロールのインデックスを使用して Select イベントを記録するように環境を変更したり、名前を使用した記録とインデックスを使用した記録を切り替えたりすることができます。


以下のコントロールでは、インデックスを使用して Select イベントを記録できます。

- FlexList
- FlexTree
- FlexDataGrid
- FlexAdvancedDataGrid
- FlexOLAPDataGrid
- FlexComboBox

デフォルト設定は、コントロールの名前を使用する ItemBasedSelection (Select イベント) です。インデックスを使用するには、IndexBasedSelection (SelectIndex イベント) を使用するように AutomationEnvironment を変更する必要があります。これらのクラスのいずれかの動作を変更するには、以下のコードを使用して FlexCommonControls.xml、AdvancedDataGrid.xml、または OLAPDataGrid.xml ファイルを変更する必要があります。これらの XML ファイルは <Silk Test_install_directory>%ng%agent%plugins%com.borland.fastxd.techdomain.flex.agent_<version>%config%automationEnvironment フォルダ内にあります。対応する xml ファイルで、以下の変更を行います。

```
<ClassInfo Extends="FlexList" Name="FlexControlName"
EnableIndexBasedSelection="true" >
...
</ClassInfo>
```

この変更では、FlexList::SelectIndex イベントの記録に IndexBasedSelection が使用されています。コードの EnableIndexBasedSelection= を false に設定するか、またはこのブール値を削除すると、記録で名前が使用される設定に戻ります (FlexList::Select イベント)。

 **注:** これらの変更内容を有効にするには、アプリケーションを再起動する必要があります。アプリケーションを再起動すると、Silk Test Agent も自動的に再起動されます。

FlexDataGrid コントロールでの項目の選択

FlexDataGrid コントロールの項目は、インデックス値または内容値を使用して選択します。

1. インデックス値を使用して FlexDataGrid コントロールの項目を選択するには、SelectIndex メソッドを使用します。
たとえば、FlexDataGrid.SelectIndex(1) のように入力します。
2. 内容値を使用して FlexDataGrid コントロールの項目を選択するには、Select メソッドを使用します。
必要な形式の文字列を使用して、選択する行を識別します。項目と項目の間は、縦線文字 (|) で区切る必要があります。少なくとも 1 つの項目を 2 つのアスタリスク (*) で囲む必要があります。これにより、クリックが実行される項目が識別されます。
構文は FlexDataGrid.Select("*Item1* | Item2 | Item3") です。

Flex アプリケーションのテストの有効化


Flex アプリケーションをテストに対して有効化するには、Adobe Flex 開発者は Flex アプリケーションに以下のコンポーネントを組み込む必要があります。

- Adobe Flex オートメーション パッケージ
- Silk Test オートメーション パッケージ

Adobe Flex オートメーション パッケージ

開発者は、Flex オートメーション パッケージを使用して、オートメーション API を使用する Flex アプリケーションを作成できます。Flex オートメーション パッケージは、Adobe の Web サイト (<http://www.adobe.com>) からダウンロードできます。パッケージには、以下の内容が含まれています。

- オートメーション ライブラリ : automation.swc ライブラリおよび automation_agent.swc ライブラリは、Flex フレームワーク コンポーネントの委譲の実装です。automation_agent.swc ファイルおよび関連するリソースバンドルは、汎用的なエージェント メカニズムです。Silk Test Agent などのエージェントは、これらのライブラリの上に構築されます。
- サンプル

 **注:** Silk Test Flex オートメーション SDK は、Flex のオートメーション API に基づいています。Silk Test オートメーション SDK は、Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、Flex オートメーション API の typekey ステートメントでは、すべてのキーはサポートされません。テキスト入カステートメントを使用してこの問題を解決できます。Flex オートメーション API の詳細については、『*Adobe Flex リリース ノート*』を参照してください。

Silk Test オートメーション パッケージ

Silk Test の Open Agent は、Adobe Flex オートメーション エージェント ライブラリを使用しています。FlexTechDomain.swc ファイルに、Silk Test 固有の実装が含まれています。

以下のいずれかの方法を使用して、アプリケーションをテストに対して有効化できます。

- Flex アプリケーションへのオートメーション パッケージのリンク
- 実行時の読み込み

Flex アプリケーションへのオートメーション パッケージのリンク

テストする予定の Flex アプリケーションを事前にコンパイルする必要があります。機能テスト クラスは、コンパイル時にアプリケーションに埋め込まれ、アプリケーションは実行時に自動テストに関する外部依存関係を持ちません。

コンパイル時にアプリケーションの SWF ファイルに機能テスト クラスを埋め込むと、SWF ファイルのサイズが大きくなります。SWF ファイルのサイズが重要でない場合は、機能テストと展開に同じ SWF ファイルを使用します。SWF ファイルのサイズが重要である場合は、2 つの SWF ファイルを生成します。1 つは機能テスト クラスが埋め込まれたファイル、もう 1 つは機能テスト クラスが埋め込まれていないファイルです。展開には、テスト クラスが埋め込まれていない SWF ファイルを使用します。

include-libraries コンパイラ オプションを指定してテストのために Flex アプリケーションを事前にコンパイルする場合は、以下のファイルを参照します。

- automation.swc
- automation_agent.swc
- FlexTechDomain.swc
- automation_charts.swc (アプリケーションでグラフおよび Flex 2.0 を使用する場合のみインクルード)
- automation_dmv.swc (アプリケーションでグラフおよび Flex 3.x 以降を使用する場合にインクルード)

- automation_flasflexkit.swc (アプリケーションで埋め込みの Flash コンテンツを使用する場合にインクルード)
- automation_spark.swc (アプリケーションで新しい Flex 4.x コントロールを使用する場合にインクルード)
- automation_air.swc (アプリケーションが AIR アプリケーションである場合にインクルード)
- automation_airspace.swc (アプリケーションが AIR アプリケーションであり、新しい Flex 4.x コントロールを使用する場合にインクルード)

Flex アプリケーションの最終リリースバージョンを作成する場合は、これらの SWC ファイルへの参照なしでアプリケーションを再コンパイルします。オートメーション SWC ファイルの使用の詳細については、『*Adobe Flex リリース ノート*』を参照してください。

アプリケーションをサーバーに展開しないで、ファイル プロトコルを使用して要求したり、Adobe Flex Builder 内で実行したりする場合は、各 SWF ファイルをローカルの信頼済みサンドボックスに組み込む必要があります。このためには、追加の構成情報が必要です。コンパイラの構成ファイルを変更するか、またはコマンド ライン オプションを使用して、構成情報を追加します。



注: Silk Test Flex オートメーション SDK は、Flex のオートメーション API に基づいています。Silk Test オートメーション SDK は、Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、オートメーション コードを使用してアプリケーションがコンパイルされ、連続的に SWF ファイルが読み込まれる場合、メモリ リークが発生して、最終的にアプリケーションでメモリが不足します。Flex Control Explorer サンプル アプリケーションは、この問題の影響を受けます。回避策として、Explorer が読み込むアプリケーションの SWF ファイルをオートメーション ライブラリを使用してコンパイルしない方法があります。たとえば、Explorer のメイン アプリケーションのみをオートメーション ライブラリを使用してコンパイルします。SWFLoader の代わりにモジュール ローダーを使用する方法もあります。Flex オートメーション API の詳細については、『*Adobe Flex リリース ノート*』を参照してください。

テストのための Flex アプリケーションの事前コンパイル

アプリケーションをテスト用に事前コンパイルするか、または実行時の読み込みを使用することによって、アプリケーションをテストに対して有効化できます。

1. 以下のコードを構成ファイルに追加することによって、コンパイラの構成ファイルに automation.swc、automation_agent.swc、および FlexTechDomain.swc ライブラリをインクルードします。

```
<include-libraries>
...
<library>/libs/automation.swc</library>
<library>/libs/automation_agent.swc</library>
<library>pathinfo/FlexTechDomain.swc</library>
</include-libraries>
```



注: アプリケーションでグラフを使用する場合は、automation_charts.swc ファイルも追加する必要があります。

2. コマンド ライン コンパイラで include-libraries コンパイラ オプションを使用して、automation.swc、automation_agent.swc、および FlexTechDomain.swc ライブラリの場所を指定します。


構成ファイルは以下の場所にあります。


Adobe Flex 2 SDK : <flex_installation_directory>/frameworks/flex-config.xml

Adobe Flex Data Services : <flex_installation_directory>/flex/WEB-INF/flex/flex-config.xml

以下の例では、automation.swc ファイルと automation_agent.swc ファイルがアプリケーションに追加されています。

```
mxmmlc -include-libraries+=./frameworks/libs/automation.swc;../frameworks/libs/automation_agent.swc;pathinfo/FlexTechDomain.swc MyApp.mxml
```

 **注:** コマンドラインで include-libraries オプションを明示的に設定すると、既存のライブラリに対して追加されるのではなく、既存のライブラリが上書きされます。コマンドラインで include-libraries オプションを使用して automation.swc ファイルと automation_agent.swc ファイルを追加する場合は、+= 演算子を使用します。これにより、インクルードされる既存のライブラリが上書きされるのではなく、インクルードされる既存のライブラリに対して追加されます。

 **注:** Silk Test Flex オートメーション SDK は、Flex のオートメーション API に基づいています。Silk Test オートメーション SDK は、Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、オートメーションコードを使用してアプリケーションがコンパイルされ、連続的に SWF ファイルが読み込まれる場合、メモリリークが発生して、最終的にアプリケーションでメモリが不足します。Flex Control Explorer サンプルアプリケーションは、この問題の影響を受けます。回避策として、Explorer が読み込むアプリケーションの SWF ファイルをオートメーションライブラリを使用してコンパイルしない方法があります。たとえば、Explorer のメインアプリケーションのみをオートメーションライブラリを使用してコンパイルします。SWFLoader の代わりにモジュールローダーを使用する方法もあります。Flex オートメーション API の詳細については、『*Adobe Flex リリースノート*』を参照してください。

実行時の読み込み

Silk Test Flex オートメーションランチャを使用して、実行時に Flex オートメーションサポートを読み込むことができます。このアプリケーションは、オートメーションライブラリを使用してコンパイルされており、SWFLoader クラスを使用してユーザーのアプリケーションを読み込みます。これにより、SWF ファイルにオートメーションライブラリをコンパイルしなくても、アプリケーションが自動的にテストに対して有効化されます。Silk Test Flex オートメーションランチャは、HTML および SWF のファイル形式で利用できます。

制限事項

- Flex オートメーションランチャアプリケーションは、自動的にルートアプリケーションとなります。ユーザーのアプリケーションをルートアプリケーションにする必要がある場合は、Silk Test Flex オートメーションランチャを使用してオートメーションサポートを読み込むことができません。
- 外部ライブラリを読み込むアプリケーション（他の SWF ファイルライブラリを読み込むアプリケーション）をテストするには、自動テストに特別な設定が必要です。実行時に読み込まれるライブラリ（ランタイム共有ライブラリ（RSL）を含む）は、読み込むアプリケーションの ApplicationDomain に読み込まれる必要があります。アプリケーションで使用される SWF ファイルが異なるアプリケーションドメインに読み込まれた場合、自動テストの記録と再生が正しく動作しません。以下に、同じ ApplicationDomain に読み込まれるライブラリの例を示します。

```
import flash.display.*;

import flash.net.URLRequest;

import flash.system.ApplicationDomain;

import flash.system.LoaderContext;

var ldr:Loader = new Loader();

var urlReq:URLRequest = new URLRequest("RuntimeClasses.swf");
```

```
var context:LoaderContext = new LoaderContext();
context.applicationDomain = ApplicationDomain.currentDomain;
loader.load(request, context);
```

実行時の読み込み

1. Silk¥Silk Test¥ng¥AutomationSDK¥Flex¥<version>¥FlexAutomationLauncher ディレクトリの内容を、テストする Flex アプリケーションのディレクトリにコピーします。
2. Windows Explorer で FlexAutomationLauncher.html を開き、ファイルパスへの接尾辞として以下のパラメータを追加します。

```
?automationurl=YourApplication.swf
```

YourApplication.swf は Flex アプリケーションの SWF ファイルの名前です。

3. ファイルパスへの接頭辞として file:/// を追加します。
たとえば、ファイルの URL に ?automationurl=explorer.swf などのパラメータが含まれている場合は、以下のように入力します。


```
file:///C:/Program%20Files/Silk/Silk Test/ng/sampleapplications/Flex/3.2/
FlexControlExplorer32/FlexAutomationLauncher.html?automationurl=explorer.swf
```

コマンドラインを使用した構成情報の追加

コマンドライン コンパイラを使用して automation.swc、automation_agent.swc、および FlexTechDomain.swc ライブラリの場所を指定するには、include-libraries コンパイラ オプションを使用します。

次の例では、automation.swc ファイルと automation_agent.swc ファイルがアプリケーションに追加されます。

```
mxmlc -include-libraries+=../frameworks/libs/automation.swc;../frameworks/libs/
automation_agent.swc;pathinfo/FlexTechDomain.swc MyApp.mxml
```

 **注:** アプリケーションでグラフを使用する場合は、include-libraries コンパイラ オプションに automation_charts.swc ファイルも追加する必要があります。

コマンドラインで include-libraries オプションを明示的に設定すると、既存のライブラリに対して追加されるのではなく、既存のライブラリが上書きされます。コマンドラインで include-libraries オプションを使用して automation.swc ファイルと automation_agent.swc ファイルを追加する場合は、+= 演算子を使用します。これにより、インクルードされる既存のライブラリが上書きされるのではなく、インクルードされる既存のライブラリに対して追加されます。

Flex Builder プロジェクトに自動テストサポートを追加するには、include-libraries コンパイラ オプションに automation.swc および automation_agent.swc ライブラリも追加する必要があります。

Flex アプリケーションにパラメータを渡す

以下の手順に従って、Flex アプリケーションにパラメータを渡すことができます。

実行する前に Flex アプリケーションにパラメータを渡す

オートメーション ライブラリを使用して、実行する前に Flex アプリケーションにパラメータを渡すことができます。

1. 適切なオートメーション ライブラリを使用して、アプリケーションをコンパイルします。
2. パラメータの指定には、通常どおり標準的な Flex のメカニズムを使用します。

Flex オートメーション ランチャを使用して、実行時に Flex アプリケーションにパラメータを渡す

このタスクを開始する前に、実行時の読み込みに対応するようにアプリケーションを準備します。

1. FlexAutomationLauncher.html ファイルを開くか、または例として FlexAutomationLauncher.html を使用してファイルを作成します。
2. 以下のセクションに移動します。

```
<script language="JavaScript" type="text/javascript">
    AC_FL_RunContent(eef
        "src", "FlexAutomationLauncher",
        "width", "100%",
        "height", "100%",
        "align", "middle",
        "id", "FlexAutomationLauncher",
        "quality", "high",
        "bgcolor", "white",
        "name", "FlexAutomationLauncher",
        "allowScriptAccess","sameDomain",
        "type", "application/x-shockwave-flash",
        "pluginspage", "http://www.adobe.com/go/getflashplayer",
        "flashvars", "yourParameter=yourParameterValue"+
"&automationurl=YourApplication.swf"
    );
</script>
```

 **注:** 「src」、 「id」、 および 「name」 の 「FlexAutomationLauncher」 の値は変更しないでください。

3. 「yourParameter=yourParameterValue」 に、独自のパラメータを追加します。
4. 「& automationurl=YourApplication.swf」 の値として、テストする Flex アプリケーションの名前を渡します。
5. ファイルを保存します。

テスト可能な Flex アプリケーションの作成

Flex 開発者は、Flex アプリケーションを可能なかぎりテストしやすくするためのテクニックを利用できます。以下のテクニックがあります。

- オブジェクトに対するわかりやすい ID の指定
- オブジェクトの重複の回避

オブジェクトに対するわかりやすい ID の指定

テストしやすいアプリケーションを作成するには、スクリプト内でオブジェクトを識別しやすくする必要があります。テストするすべてのコントロールに対して、わかりやすい文字列を使用した ID プロパティの値を設定できます。

オブジェクトに対してわかりやすい ID を指定するには：

- テスト可能なすべての MXML コンポーネントに対して ID を指定して、その Flex コントロールの参照時にテスト スクリプトで一意的 ID が使用できるようにします。
- これらの ID は、ユーザーがテスト スクリプト内でそのオブジェクトを容易に識別できるように、可能な限り人間が理解しやすい文字列にします。たとえば、TabNavigator 内の Panel コンテナの id プロパティは、panel1 や p1 ではなく submit_panel とします。

Silk4NET を使用する場合、id や childIndex などの特定のタグに基づいて、オブジェクトに対して自動的に名前が設定されます。id プロパティに値がない場合、Silk4NET では、childIndex プロパティなどの他のプロパティが使用されます。id プロパティに値を割り当てると、テスト スクリプトを読みやすくすることができます。

オブジェクトの重複の回避

自動エージェントの処理は、実行中にオブジェクト インスタンスの一部のプロパティが変更されないことを前提としています。実行時に Silk4NET によってオブジェクト名として使用されている Flex コンポーネント プロパティを変更すると、予期しない結果が発生する可能性があります。たとえば、automationName プロパティのない Button コントロールを作成し、最初は label プロパティに値を設定しないで、その後、label プロパティに値を設定した場合、問題が発生することがあります。この場合、Silk4NET では、automationName プロパティが設定されていない場合は Button コントロールを識別するためにコントロールの label プロパティの値が使用されます。あとから label プロパティの値を設定したり、既存の label の値を変更すると、Silk4NET ではこのオブジェクトを新しいオブジェクトとして識別し、既存のオブジェクトを参照しなくなります。


重複オブジェクトを回避するには：

- エージェントにおいて、オブジェクトの識別にどのプロパティが使用されているかを理解し、実行時にそれらのプロパティを変更しないようにします。
- 記録されたスクリプトに含まれているすべてのオブジェクトに対して、人間が理解しやすい一意の id プロパティまたは automationName プロパティを設定します。

Flex の AutomationName プロパティと AutomationIndex プロパティ

Flex オートメーション API には、automationName プロパティと automationIndex プロパティが用意されています。automationName を指定すると、Silk4NET では、記録されたウィンドウ宣言の名前としてこの値が使用されます。わかりやすい名前を指定すると、そのオブジェクトを Silk4NET で識別しやすくなります。ベストプラクティスとして、アプリケーションのテストに含まれているすべてのオブジェクトの automationName プロパティに値を設定することをお勧めします。

automationIndex プロパティを使用して、オブジェクトに対して一意のインデックス値を割り当てます。たとえば、2つのオブジェクトが同じ名前を共有している場合は、インデックス値を割り当てて、2つのオブジェクトを識別します。

 **注:** Silk Test Flex オートメーション SDK は、Flex のオートメーション API に基づいています。Silk Test オートメーション SDK は、Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、オートメーション コードを使用してアプリケーションがコンパイルされ、連続的に SWF ファイルが読み込まれる場合、メモリ リークが発生して、最終的にアプリケーションでメモリが不足します。Flex Control Explorer サンプル アプリケーションは、この問題の影響を受けます。回避策として、Explorer が読み込むアプリケーションの SWF ファイルをオートメーション ライブラリを使用してコンパイルしない方法があります。たとえば、Explorer のメイン アプリケーションのみをオートメーション ライブラリを使用してコンパイルしま

す。SWFLoader の代わりにモジュール ロダーを使用する方法もあります。Flex オートメーション API の詳細については、『*Adobe Flex リリース ノート*』を参照してください。

Flex クラス定義ファイル

クラス定義ファイルには、インストール化されたすべての Flex コンポーネントについての情報が含まれています。このファイルでは、記録中にイベントを送信でき、再生中にイベントを受け取ることができるコンポーネントについての情報が提供されます。クラス定義ファイルには、サポートされているプロパティの定義も含まれています。

Silk Test には、Flex の共通コントロールおよび特殊化されたコントロールのすべてのクラス、イベント、およびプロパティを記述するいくつかの XML ファイルが含まれています。これらの XML ファイルは `Silk Test_install_directory>¥ng¥agent¥plugins¥com.borland.fastxd.techdomain.flex.agent_<version>¥config¥automationEnvironment` フォルダ内にあります。

独自の XML ファイルを提供する場合は、XML ファイルをこのフォルダにコピーする必要があります。Silk Test Agent が起動して Adobe Flex のサポートを初期化するとき、このディレクトリの内容が読み込まれます。

XML ファイルの基本的な構造は以下のとおりです。

```
<TypeInfoInformation>
<ClassInfo>
<Implementation />
<Events>
<Event />
...
</Events>
<Properties>
<Property />
...
</Properties>
</ClassInfo>
</TypeInfoInformation>
```

Flex の automationName プロパティの設定

automationName プロパティは、テストに表示されるコンポーネント名を定義します。このプロパティのデフォルト値は、コンポーネントの種類に応じて異なります。たとえば、Button コントロールの automationName は、Button コントロールのラベルです。automationName がコントロールの id プロパティと同じ場合もありますが、常に同じであるわけではありません。

一部のコンポーネントでは、automationName プロパティの値は、Flex によってそのコンポーネントを認識しやすい属性に設定されています。これにより、テスト担当者は、テストでコンポーネントを認識しやすくなります。通常、テスト担当者は、アプリケーションの基になるソース コードにアクセスできないため、コントロールの表示されるプロパティによってそのコントロールを認識できるようにすることは有用です。たとえば、「Process Form Now」というラベルが設定された Button は、テストで `FlexButton("Process Form Now")` と表示されます。

新しいコンポーネントを実装する場合や、既存のコンポーネントから派生する場合は、automationName プロパティのデフォルト値をオーバーライドできます。たとえば、UIComponent では、automationName の値は、デフォルトでコンポーネントの id プロパティに設定されます。ただし、一部のコンポーネントでは、独自の方法を使用して値が設定されます。たとえば、Flex Store サンプルアプリケーションでは、コンテナを使用して製品のサムネイルが作成されています。コンテナのデフォルトの automationName はコンテナの id プロパティと同じ値となるため、あまり役立ちません。そのため、Flex Store では、製品のサムネイルを生成するカスタム コンポーネントで明示的に automationName を製品名に設定して、アプリケーションをテストしやすくしています。

以下の CatalogPanel.mxml カスタム コンポーネントの例では、automationName プロパティの値をカタログに表示される項目名に設定しています。これにより、デフォルトのオートメーション名を使用するよりもサムネイルを認識しやすくなります。

```
thumbs[i].automationName = catalog[i].name;
```

以下の例では、ComboBox コントロールの automationName プロパティを「Credit Card List」に設定しています。このように設定すると、通常、テスト ツールでは、スクリプトにおいて id プロパティではなく「Credit Card List」を使用して ComboBox が識別されます。

```
<?xml version="1.0"?>
<!-- at/SimpleComboBox.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      [Bindable]
      public var cards: Array = [
        {label:"Visa", data:1},
        {label:"MasterCard", data:2},
        {label:"American Express", data:3}
      ];

      [Bindable]
      public var selectedItem:Object;
    ]]>
  </mx:Script>
  <mx:Panel title="ComboBox Control Example">
    <mx:ComboBox id="cb1" dataProvider="{cards}"
      width="150"
      close="selectedItem=ComboBox(event.target).selectedItem"
      automationName="Credit Card List">
```

```

/>
<mx:VBox width="250">
<mx:Text width="200" color="blue" text="Select a type of credit card." />
<mx:Label text="You selected: {selectedItem.label}"/>
<mx:Label text="Data: {selectedItem.data}"/>
</mx:VBox>
</mx:Panel>
</mx:Application>

```

automationName プロパティの値を設定すると、オブジェクト名が実行時に変更されないことが保証されます。このことは、予期しない結果の回避に役立ちます。

automationName プロパティの値を設定すると、テストでは、デフォルト値ではなく、その値が使用されます。たとえば、Silk4NET では、デフォルトで、スクリプトにおいて Button コントロールの label プロパティがボタンの名前として使用されます。この場合、ラベルが変更されると、スクリプトが動作しなくなります。automationName プロパティの値を明示的に設定することによって、このような事態を回避できます。

ラベルがなく、アイコンがあるボタンは、インデックス番号によって記録されます。この場合は、automationName プロパティをわかりやすい文字列に設定して、テスト担当者がスクリプトでボタンを認識できるようにします。automationName プロパティの値を設定したあとは、コンポーネントのライフサイクル全体を通して値を変更しないでください。項目レンダラでは、automationName プロパティではなく automationValue プロパティを使用します。automationValue プロパティを使用するには、createAutomationIDPart() メソッドをオーバーライドして、automationName プロパティに割り当てる新しい値を返します。以下に例を示します。

```

<mx:List xmlns:mx="http://www.adobe.com/2006/mxml">
<mx:Script>
<![CDATA[
import mx.automation.IAutomationObject;
override public function
createAutomationIDPart(item:IAutomationObject):Object {
var id:Object = super.createAutomationIDPart(item); id["automationName"] =
id["automationIndex"];
return id;
}
]]>
</mx:Script>
</mx:List>

```

このテクニックを使用して、任意のコンテナまたはリスト形式コントロールの子にインデックス値を追加します。子が自分自身のインデックスを指定する方法はありません。

名前またはインデックスを使用するように Flex の Select メソッドを設定

Flex の Select メソッドは、選択するコントロールの Name または Index を使用して記録できます。デフォルトで、Silk Test では、コントロールの名前を使用して Select メソッドが記録されます。ただし、コントロールのインデックスを使用して Select イベントを記録するように環境を変更したり、名前を使用した記録とインデックスを使用した記録を切り替えたりすることができます。

1. インデックスを使用するように変更するクラスを特定します。

以下のコントロールでは、インデックスを使用して Select イベントを記録できます。

- FlexList
- FlexTree
- FlexDataGrid
- FlexOLAPDataGrid
- FlexComboBox
- FlexAdvancedDataGrid

2. 変更するクラスに関連する XML ファイルを特定します。

上記のコントロールに関連する XML ファイルには、FlexCommonControls.xml、AdvancedDataGrid.xml、または OLAPDataGrid.xml があります。

3. 変更するクラスに関連する XML ファイルに移動します。

XML ファイルは、<Silk Test_install_directory>%ng%agent%plugins
%com.borland.fastxd.techdomain.flex.agent_<version>%config%automationEnvironment フォルダにあります。

4. 対応する XML ファイルで、以下の変更を行います。

```
<ClassInfo Extends="FlexList" Name="FlexControlName"  
EnableIndexedSelection="true" >  
...  
</ClassInfo>
```

たとえば、「FlexControlName」として「FlexList」を使用し、FlexCommonControls.xml ファイルを変更できます。

この変更では、FlexList::SelectIndex イベントの記録に IndexedSelection が使用されています。



注: コードの EnableIndexedSelection= を false に設定するか、またはこのブール値を削除すると、記録で名前が使用される設定に戻ります (FlexList::Select イベント)。

5. これらの変更内容を有効にするには、Flex アプリケーションおよび Open Agent を再起動します。

Flex コンテナのコーディング

コンテナは、ユーザー対話（ユーザーが Accordion コンテナの次のページに移動したなど）の記録、およびテスト スクリプト内でのコントロールに対する一意の場所の提供の両方の目的で使用されるため、他の種類のコントロールとは異なります。

オートメーション階層におけるコンテナの追加と削除

通常、自動テスト機能のスクリプトでは、ネストされたコンテナについての詳細情報は少なく抑えられます。テストの結果やコントロールの識別に影響がないコンテナは、スクリプトから削除されます。削除対象となるコンテナは、HBox、VBox、Canvas などの、レイアウトの目的でのみ使用されるコンテナです。ただし、ViewStack、TabNavigator、Accordion などの複数ビュー ナビゲータ コンテナで使用されている場合は削除されません。このような場合、コンテナはオートメーション階層に追加されて、ナビゲーションに使用されます。

多くの複合コンポーネントでは、Canvas や VBox などのコンテナを使用して、子が整理されます。これらのコンテナは、アプリケーション上では視覚的な効果を持ちません。この結果、これらのコンテナでは、ユーザー操作は実行されず、操作を視覚的に記録する必要もないため、通常、これらのコンテナはテストから除外されます。テストからコンテナを除外することによって、関連するテスト スクリプトが簡潔になり、読みやすくなります。

コンテナを記録から除外するには、コンテナの `showInAutomationHierarchy` プロパティを `false` に設定します（子は除外されません）。このプロパティは、UIComponent クラスによって定義されているため、UIComponent のサブクラスであるすべてのコンテナにこのプロパティが存在します。階層で表示されないコンテナの子は、階層内でそのコンテナの次に上位の親の子として表示されます。

`showInAutomationHierarchy` プロパティのデフォルト値は、コンテナの種類に応じて異なります。Panel、Accordion、Application、DividedBox、Form などのコンテナではデフォルト値は `true` であり、Canvas、HBox、VBox、FormItem などのコンテナではデフォルト値は `false` です。

以下の例では、VBox コンテナがテスト スクリプトの階層に組み込まれています。

```
<?xml version="1.0"?>

<!-- at/NestedButton.mxml -->

<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Panel title="ComboBox Control Example">
    <mx:HBox id="hb">
      <mx:VBox id="vb1" showInAutomationHierarchy="true">
        <mx:Canvas id="c1">
          <mx:Button id="b1" automationName="Nested Button 1" label="Click Me" />
        </mx:Canvas>
      </mx:VBox>
      <mx:VBox id="vb2" showInAutomationHierarchy="true">
        <mx:Canvas id="c2">
          <mx:Button id="b2" automationName="Nested Button 2" label="Click Me 2" />
        </mx:Canvas>
      </mx:VBox>
    </mx:HBox>
  </mx:Panel>
</mx:Application>
```

複数ビュー コンテナ

TabNavigator や Accordion コンテナなどの複数ビュー コンテナ内の複数のタブに同じラベルを使用しないでください。同じラベルを使用することもできますが、このことは、通常、推奨 UI 設計プラクティスとして推奨されていません。このようにすると、テスト環境においてコントロールの識別に問題が発生することがあります。

Flex 自動テスト ワークフロー

Flex アプリケーションのテストの Silk4NET ワークフローは、以下のとおりです。

- 自動テストの初期化
- 自動テストの記録
- 自動テストの再生

Flex 自動テストの初期化

ユーザーが Flex アプリケーションを起動すると、以下の初期化イベントが発生します。

1. オートメーション初期化コードによって、コンポーネントの委譲クラスがコンポーネントのクラスに関連付けられます。
2. コンポーネントの委譲クラスは、IAutomationObject インターフェイスを実装します。
3. AutomationManager のインスタンスがミックスインの init() メソッドで作成されます。(AutomationManager はミックスインです。)
4. SystemManager によってアプリケーションが初期化されます。コンポーネント インスタンスおよび対応する委譲インスタンスが作成されます。委譲インスタンスによって、目的のイベントに対するイベント リスナーが追加されます。
5. Silk4NET FlexTechDomain はミックスインです。FlexTechDomain の init() メソッドで、FlexTechDomain が SystemManager.APPLICATION_COMPLETE イベントに登録されます。イベントを受信すると、FlexTechDomain インスタンスが作成されます。
6. FlexTechDomain インスタンスが、同じマシン上の記録および再生機能に登録する Silk Test Agent に TCP/IP ソケット経由で接続します。
7. FlexTechDomain は、自動環境についての情報を要求します。この情報は XML ファイルに格納され、Silk Test Agent から FlexTechDomain に転送されます。

Flex 自動テストの記録

ユーザーが Silk4NET で Flex アプリケーションの新しいテストを記録すると、以下のイベントが発生します。

1. Silk4NET によって Silk Test Agent が呼び出されて、記録が開始されます。Agent は、このコマンドを FlexTechDomain インスタンスに転送します。
2. FlexTechDomain は、beginRecording() を呼び出すことによって、AutomationManager に対して記録の開始を通知します。AutomationManager は、SystemManager からの AutomationRecordEvent.RECORD イベントに対するリスナーを追加します。
3. ユーザーがアプリケーションを操作します。たとえば、ユーザーが Button コントロールをクリックしたとします。
4. ButtonDelegate.clickEventHandler() メソッドによって、プロパティとしてクリック イベントと Button のインスタンスが指定された AutomationRecordEvent イベントがディスパッチされます。
5. AutomationManager は、XML 環境情報に基づいて、クリック イベントのどのプロパティを格納するかを決定します。値が適切な型または書式に変換されます。記録イベントがディスパッチされます。
6. FlexTechDomain イベント ハンドラがイベントを受信します。AutomationManager.createID() メソッドが呼び出されて、ボタンの AutomationID オブジェクトが作成されます。このオブジェクトは、オブジェクト識別用の構造体を提供します。AutomationID 構造体は、AutomationIDParts の配列になっています。AutomationIDParts は、IAutomationObject を使用して作成されます。(Button コントロールの UIComponent.id、automationName、automationValue、childIndex、および label プロパティが読み込まれて、オブジェクトに格納されます。XML 情報に、label プロパティを Button の識別に使用できることが指定されているため、label プロパティが使用されます。)
7. FlexTechDomain は、AutomationManager.getParent() メソッドを使用して、Button の論理的な親を取得します。アプリケーション レベルまでの各レベルで、親コントロールの AutomationIDParts オブジェクトが収集されます。

8. すべての AutomationIDParts が AutomationID オブジェクトの一部として組み込まれます。
9. FlexTechDomain は、Silk4NET への呼び出しでこの情報を送信します。
10. ユーザーが記録を停止すると、FlexTechDomain.endRecording() メソッドが呼び出されます。

Flex 自動テストの再生

ユーザーが Silk4NET で **再生** ボタンをクリックすると、以下のイベントが発生します。

1. 各スクリプト呼び出しにおいて、Silk4NET は Silk Test Agent に接続し、実行されるスクリプト呼び出しの情報を送信します。この情報には、完全なウィンドウ宣言、イベント名、およびパラメータが含まれています。
2. Silk Test Agent は、その情報を FlexTechDomain に転送します。
3. FlexTechDomain は、ウィンドウ宣言情報と共に AutomationManager.resolveIDToSingleObject を使用します。AutomationManager は、説明情報 (automationName、automationIndex、id など) に基づいて、解決したオブジェクトを返します。
4. Flex コントロールが解決されると、FlexTechDomain は AutomationManager.replayAutomatableEvent() を呼び出して、イベントを再生します。
5. AutomationManager.replayAutomatableEvent() メソッドによって、委譲クラスの IAutomationObject.replayAutomatableEvent() メソッドが呼び出されます。委譲では、IAutomationObjectHelper.replayMouseEvent() メソッド (または replayKeyboardEvent() などの他のいずれかの再生メソッド) を使用してイベントが再生されます。
6. スクリプトに検証がある場合、FlexTechDomain は AutomationManager.getProperties() を呼び出して、検証する必要がある値にアクセスします。

Adobe Flex アプリケーションのスタイル

Adobe Flex 3.x で開発されたアプリケーションについて、Silk4NET ではスタイルとプロパティを区別しません。この結果、スタイルはプロパティとして公開されます。ただし、Adobe Flex 4.x の Spark という接頭辞が付いているすべての新しい Flex コントロール (SparkButton など) では、スタイルがプロパティとして公開されません。この結果、Flex 4.x コントロールの GetProperty() メソッドおよび GetPropertyList() メソッドでは color や fontSize などのスタイルが返されず、text や name などのプロパティのみが返されます。

GetStyle(string styleName) メソッドは、スタイルの値を文字列として返します。どのようなスタイルが存在するかを確認するには、Adobe ヘルプ (http://help.adobe.com/en_US/FlashPlatform/reference/actionsript/3/package-detail.html) を参照してください。

スタイルが設定されていない場合は、再生中に StyleNotSetException が発生します。

FlexTree などの Flex 3.x コントロールでは、GetProperty() を使用してスタイルを取得できます。GetStyle() を使用することもできます。Flex 3.x コントロールでは、GetProperty() メソッドと GetStyle() メソッドの両方が動作します。

色スタイルの計算

Flex では、色は数値として表されます。色は、以下の式を使用して計算できます。

```
red*65536 + green*256 + blue
```

例

以下のスクリプト例では、Spark アプリケーションの ButtonBar がフォント サイズ 12 を使用しているかどうかを検証しています。

Adobe Flash Player のセキュリティ制約に対応するための Flex アプリケーションの構成

Adobe Flash Player 10 では、セキュリティ モデルが以前のバージョンから変更されています。Flash Player を使用するテストを記録する場合、記録は想定どおりに動作します。ただし、テストを再生する場合は、特定の状況で高レベルのクリックが行われると、予期しない結果が発生します。たとえば、**ファイル参照** ダイアログ ボックスをプログラムから開くことができません。このシナリオの再生を試みると、セキュリティ制約が原因でテストに失敗します。

このセキュリティ制約を回避するには、ダイアログ ボックスを開くボタンに対して低レベルのクリックを実行します。低レベルのクリックを作成するには、click メソッドにパラメータを追加します。

たとえば、SparkButton::Click() の代わりに SparkButton::Click(MouseButton.Left) を使用します。パラメータを指定しない Click() は高レベルのクリックとして再生され、パラメータを指定したクリック (ボタンなど) は低レベルのクリックとして再生されます。

1. Flash Player を使用するテストを記録します。
2. Click メソッドに移動して、パラメータを追加します。
たとえば、**ファイルを開く** ダイアログ ボックスを開くには、以下のように指定します。

```
SparkButton("@caption='Open File Dialog...']").Click(MouseButton.Left)
```

テストを再生すると、想定どおりに動作します。

Adobe Flex アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Flex アプリケーションがサポートする属性は次のとおりです。

- automationName
- caption (automationName と同様)
- automationClassName (FlexButton など)
- className (実装クラスの完全修飾名。例: mx.controls.Button)
- automationIndex (FlexAutomation のビューでのコントロールのインデックス。例: index:1)
- index (automationIndex と同様。ただし、接頭辞はなし。例: 1)
- id (コントロールの ID)
- windowId (id と同様)
- label (コントロールのラベル)
- すべての動的ロケーター属性





注: 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

Java AWT/Swing のサポート

Silk4NET は、Java AWT/Swing コントロールを使用するアプリケーションまたはアプレットのテストを組み込みでサポートしています。Java AWT/Swing を使用するアプリケーションまたはアプレットを設定すると、Silk4NET は標準の AWT/Swing コントロールのテストのサポートを組み込みで提供します。

 **注:** Java AWT/Swing アプリケーションまたはアプレットに埋め込まれた Java SWT コントロールや、Java SWT アプリケーションに埋め込まれた Java AWT/Swing コントロールもテストできます。

 **注:** イメージ クリックの記録は、Java AWT/Swing コントロールを使用するアプリケーションまたはアプレットではサポートされません。

サンプル アプリケーション

Silk Test は、サンプルの Swing テスト アプリケーションを提供しています。サンプル アプリケーションを <http://supportline.microfocus.com/websync/SilkTest.aspx> からダウンロードします。サンプル アプリケーションをインストールしたあと、**スタート > プログラム > Silk > Silk Test > Sample Applications > Java Swing > Swing Test Application** をクリックします。

新機能、サポート対象のプラットフォームとバージョン、既知の問題、および回避策の詳細については、『Silk Test リリース ノート』(<http://supportline.microfocus.com/productdoc.aspx> で入手可能) を参照してください。

サポートするコントロール


Java AWT/Swing のテストで使用可能なコントロールの完全な一覧については、「Java AWT/Swing クラス リファレンス」を参照してください。

Java AWT/Swing アプリケーションの属性

ロケーターが作成される時、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Java AWT/Swing でサポートされる属性には以下のものがあります。

- caption
- priorlabel : 隣接するラベル フィールドのテキストによってテキスト入力フィールドを識別します。通常、フォームのすべての入力フィールドに、入力の目的を説明するラベルがあります。caption のないコントロールの場合、自動的に属性 **priorlabel** がロケーターに使用されます。コントロールの **priorlabel** 値 (テキスト入力フィールドなど) には、コントロールの左側または上にある最も近いラベルの caption が使用されます。
- name
- accessibleName
- *Swing* のみ : すべてのカスタム オブジェクトの定義属性は、ウィジェットに `SetClientProperty("propertyName", "propertyValue")` で設定されます。

 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Java メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティの取得、またはプロパティの設定を直接実行できます。また、このコントロールの Silk4NET API で使用できないメソッドおよびプロパティも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4NET API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは `Invoke` メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、`GetDynamicMethodList` メソッドを使用します。

オブジェクトの複数の動的メソッドは `InvokeMethods` メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、`GetDynamicMethodList` メソッドを使用します。

動的プロパティの取得には `GetProperty` メソッドを、動的プロパティの設定には `SetProperty` メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、`GetPropertyList` メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを `String` 型の入力パラメータとして設定する必要がある `SetTitle` というメソッドを呼び出すには、次のように入力します：

```
control.Invoke("SetTitle", "my new title")
```



注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。



注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティを取得する場合、`Reflection` を使用します。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4NET がサポートするコントロールのメソッドとプロパティ。
- SWT、AWT、または Swing ウィジェットのすべてのパブリック メソッド
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティ。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- プリミティブ型 (`boolean`、`integer`、`long`、`double`、`string`)
プリミティブ型 (`int` など) とオブジェクト タイプ (`java.lang.Integer` など) の両方がサポートされます。プリミティブ型は必要に応じて拡大変換されます。たとえば、`long` が必要な場所で `int` を渡すことができます。
- 列挙型
列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが列挙型 `java.sql.ClientInfoStatus` のパラメータを必要とする場合、`REASON_UNKNOWN`、`REASON_UNKNOWN_PROPERTY`、`REASON_VALUE_INVALID`、または `REASON_VALUE_TRUNCATED` の文字列値を使用できます。
- リスト
リスト、配列、または可変長引数のパラメータを持つメソッドを呼び出すことができます。リストの要素がターゲットの配列型に代入可能の場合、配列型への変換は自動的に行われます。
- その他のコントロール
コントロール パラメーターは、`TestObject` として渡したり、返したりできます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4NET 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、C# では `null` が、VB では `Nothing` が返されます。

Java AWT/Swing テクノロジ ドメインでの `priorLabel` の判別

Java AWT/Swing テクノロジ ドメインで `priorLabel` を判別するには、ターゲット コントロールと同じウィンドウ内のすべてのラベルおよびグループを考慮する必要があります。判別の基準は、次のとおりです。

- priorLabel の候補とみなされるのは、コントロールの上または左にあるラベル、およびコントロールが属しているグループのみです。
- コントロールの親が JViewport または ScrollPane の場合、アルゴリズムはこのコントロールを含むウィンドウが親であるかのように機能し、外側の要素はどれも関連しないとみなされます。
- 最も単純なケースでは、コントロールに最も近いラベルが priorLabel として使用されます。
- 2 つのラベルがコントロールから等距離にあり、1 つがコントロールの左、もう 1 つが上にある場合は、左側のラベルが優先します。
- 適したラベルがない場合は、最も近いグループのキャプションが使用されます。

Java SWT と Eclipse RCP のサポート

Silk Test は、SWT (Standard Widget Toolkit) コントロールのウィジェットを使用したアプリケーションのテストの組み込みサポートを提供します。Java SWT/RCP アプリケーションを構成すると、Silk Test は、標準的な Java SWT/RCP コントロールのテストのサポートを自動的に提供します。

Silk Test は、以下をサポートします。

- Java AWT/Swing アプリケーションに埋め込まれた Java SWT コントロール、および Java SWT アプリケーションに埋め込まれた Java AWT/Swing コントロールのテスト。
- Java SWT アプリケーションのテスト。
- レンダリングに SWT ウィジェットを使用する Eclipse ベースのアプリケーション。Silk Test は、Eclipse IDE ベース、および RCP ベースの両方のアプリケーションをサポートします。

新機能、サポート対象のプラットフォームとバージョン、既知の問題、および回避策の詳細については、『Silk Test リリース ノート』(<http://supportline.microfocus.com/productdoc.aspx> で入手可能) を参照してください。

サポートするコントロール

SWT テストで使用できるウィジェットの完全なリストについては、「Java SWT クラス リファレンス」を参照してください。

Java SWT クラス リファレンス

Java SWT アプリケーションを設定すると、Silk4NET は標準の Java SWT コントロールのテストのサポートを組み込みで提供します。

Java SWT カスタム属性

カスタム属性をテスト アプリケーションに追加して、テストをより安定させることができます。たとえば、Java SWT では、GUI を実装する開発者が属性 ('silkTestAutomationId' など) をウィジェットに対して定義することによって、アプリケーション内でそのウィジェットを一意に識別することができます。これにより、Silk4NET を使用するテスト担当者は、その属性 (この場合は 'silkTestAutomationId') をカスタム属性のリストに追加すると、その一意の ID によってコントロールを識別できるようになります。カスタム属性を使用すると、caption や index のような他の属性よりも高い信頼性を得ることができます。これは、caption はアプリケーションを他の言語に翻訳した場合に変更され、index は定義済みのウィジェットより前に他のウィジェットが追加されると変更されるためです。

複数のオブジェクトに同じカスタム属性の値が割り当てられた場合は、そのカスタム属性を呼び出したときにその値を持つすべてのオブジェクトが返されます。たとえば、一意の ID として 'loginName' を 2 つの異なるテキストフィールドに割り当てた場合は、'loginName' 属性を呼び出したときに、両方のフィールドが返されます。

Java SWT の例

以下のコードを使用して、テストするアプリケーションにボタンを作成する場合：

```
Button myButton = Button(parent, SWT.NONE);  
  
myButton.setData("SilkTestAutomationId", "myButtonId");
```

テストの XPath クエリ文字列に属性を追加するには、以下のクエリを使用します。

```
Dim button =  
desktop.PushButton("@SilkTestAutomationId='myButton'")
```

Java SWT アプリケーションをカスタム属性のテストに対して有効化にするには、開発者はカスタム属性をアプリケーションに含める必要があります。属性を含めるには `org.swt.widgets.Widget.setData(String key, Object value)` メソッドを使用します。

Java SWT アプリケーションの属性

ロケーターが作成される時、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Java SWT がサポートする属性は次のとおりです。

- caption
- すべてのカスタム オブジェクト定義属性



注: 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Java メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティの取得、またはプロパティの設定を直接実行できます。また、このコントロールの Silk4NET API で使用できないメソッドおよびプロパティも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4NET API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは `Invoke` メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、`GetDynamicMethodList` メソッドを使用します。

オブジェクトの複数の動的メソッドは `InvokeMethods` メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、`GetDynamicMethodList` メソッドを使用します。

動的プロパティの取得には `GetProperty` メソッドを、動的プロパティの設定には `SetProperty` メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、`GetPropertyList` メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを `String` 型の入力パラメータとして設定する必要がある `SetTitle` というメソッドを呼び出すには、次のように入力します：

```
control.Invoke("SetTitle", "my new title")
```



注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。



注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティを取得する場合、`Reflection` を使用します。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4NET がサポートするコントロールのメソッドとプロパティ。
- SWT、AWT、または Swing ウィジェットのすべてのパブリック メソッド
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティ。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- プリミティブ型 (boolean、integer、long、double、string)
プリミティブ型 (int など) とオブジェクト タイプ (java.lang.Integer など) の両方がサポートされます。プリミティブ型は必要に応じて拡大変換されます。たとえば、long が必要な場所で int を渡すことができます。
- 列挙型
列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが列挙型 java.sql.ClientInfoStatus のパラメータを必要とする場合、REASON_UNKNOWN、REASON_UNKNOWN_PROPERTY、REASON_VALUE_INVALID、または REASON_VALUE_TRUNCATED の文字列値を使用できます。
- リスト
リスト、配列、または可変長引数のパラメータを持つメソッドを呼び出すことができます。リストの要素がターゲットの配列型に代入可能の場合、配列型への変換は自動的に行われます。
- その他のコントロール
コントロールパラメーターは、TestObject として渡したり、返したりできます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4NET 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、C# では null が、VB では Nothing が返されます。

.NET のサポート

Silk Test は、以下の .NET アプリケーションのテストを組み込みでサポートしています。

- Windows Forms (Win Forms) アプリケーション
- Windows Presentation Foundation (WPF) アプリケーション
- Microsoft Silverlight アプリケーション

サポートされているバージョンの詳細については、**スタート > プログラム > Silk > Silk Test > リリース ノート** をクリックして、リリース ノートを表示してください。

Windows Forms のサポート

Silk4NET は、.NET スタンドアロン アプリケーションとノータッチ Windows Forms (Win Forms) アプリケーションのテストを組み込みでサポートしています。ただし、スタンドアロン アプリケーションでは、side-by-side 実行はサポートされていません。Silk4NET では、以下に埋め込まれているコントロールを記録し、再生することができます。

- Framework バージョン 2.0
- Framework バージョン 3.0
- Framework バージョン 3.5

新機能、サポート対象のプラットフォームとバージョン、既知の問題、および回避策の詳細については、『Silk Test リリース ノート』(<http://supportline.microfocus.com/productdoc.aspx> で入手可能) を参照してください。

オブジェクト解決

アプリケーション中の要素に指定された name が使用可能な場合、ロケーターの automationId 属性として使用されます。この結果、多くのオブジェクトは、この属性のみを使用して一意に識別できます。

サポートするコントロール

Win Forms テストで使用可能な記録/再生コントロールの完全な一覧については、「Windows Forms クラス リファレンス」を参照してください。

Windows Forms クラス リファレンス


Windows Forms アプリケーションを設定すると、Silk4NET は標準の Windows Forms コントロールのテストのサポートを組み込みで提供します。

Windows Forms アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Windows Forms アプリケーションがサポートする属性は次のとおりです。

- automationid
- caption
- windowid
- priorlabel (caption のないコントロールの場合、自動的に priorlabel が caption として使用されます。caption のあるコントロールの場合、caption を使う方が簡単な場合があります。)

 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Windows Forms メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティの取得、またはプロパティの設定を直接実行できます。また、このコントロールの Silk4NET API で使用できないメソッドおよびプロパティも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4NET API を通じて公開されていない場合に特に便利です。

オブジェクトの動的メソッドは Invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、GetDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは InvokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、GetDynamicMethodList メソッドを使用します。

動的プロパティの取得には GetProperty メソッドを、動的プロパティの設定には SetProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、GetPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.Invoke("SetTitle", "my new title")
```



注：通常、ほとんどのプロパティは読み取り専用で、設定できません。



注：ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティを取得する場合、Reflection を使用します。

Invoke メソッド

Windows Forms または WPF コントロールでは、Invoke メソッドを使用して、以下のメソッドを呼び出すことができます。

- MSDN が定義するコントロールのパブリック メソッド。
- MSDN が定義する静的パブリック メソッド。
- ユーザーが定義する任意の型の静的パブリック メソッド。

Invoke メソッドの最初の例

Silk4NET の DataGrid 型のオブジェクトでは、MSDN が System.Windows.Forms.DataGrid 型に定義しているすべてのメソッドを呼び出すことができます。

System.Windows.Forms.DataGrid クラスのメソッド IsExpanded を呼び出すには、次のコードを使用します。

```
//VB .NET code  
Dim isExpanded As Boolean = dataGrid.Invoke("IsExpanded", 3)
```

```
//C# code  
bool isExpanded = (bool) dataGrid.Invoke("IsExpanded", 3);
```

Invoke メソッドの 2 番目の例

AUT 内の静的メソッド String.Compare(String s1, String s2) を呼び出すには、次のコードを使用します。

```
//VB .NET code  
Dim result as Integer = (Integer)  
mainWindow.Invoke("System.String.Compare", "a", "b")
```

```
//C# code  
int result = (int) mainWindow.Invoke("System.String.Compare", "a", "b");
```

Invoke メソッドの 3 番目の例

この例では、ユーザーが生成したメソッド GetContents を動的に呼び出す方法を示します。

テスト対象アプリケーション (AUT) のコントロールの操作に使用するコードを作成できます (この例では UltraGrid)。UltraGrid の内容を取得するために、複雑な動的呼び出しを作成するのではなく、新しいメソッド GetContents を生成し、この新しいメソッドを動的に呼び出すことができます。

Visual Studio で、AUT 内の次のコードによって GetContents メソッドを UltraGridUtil クラスのメソッドとして定義します。

```
//C# code, because this is code in the AUT  
namespace UltraGridExtensions {
```



```

public class UltraGridUtil {
    /// <summary>
    /// Retrieves the contents of an UltraGrid as nested list
    /// </summary>
    /// <param name="grid"></param>
    /// <returns></returns>
    public static List<List<string>>
    GetContents(Infragistics.Win.UltraWinGrid.UltraGrid grid) {
        var result = new List<List<string>>();
        foreach (var row in grid.Rows) {
            var rowContent = new List<string>();
            foreach (var cell in row.Cells) {
                rowContent.Add(cell.Text);
            }
            result.Add(rowContent);
        }
        return result;
    }
}

```

UltraGridUtil クラスのコードを AUT に追加する必要があります。これは、次のようにして行います。

- アプリケーション開発者は、クラスのコードを AUT にコンパイルできます。アセンブリがすでにロードされている必要があります。
- テストの実行時に AUT にロードされる新しいアセンブリを作成できます。

アセンブリをロードするには、次のコードを使用します。

```
FormsWindow.LoadAssembly(String assemblyFileName)
```

フルパスを使用して、アセンブリをロードできます。例：

```
mainWindow.LoadAssembly("C:/temp/ultraGridExtensions.dll")
```

Location メソッドを使用してアセンブリの場所を見つけることもできます。

```
//VB.NET code
```

```
Dim assemblyLocation =
GetType(UltraGridExtensions.UltraGridUtil).Assembly.Location
mainWindow.LoadAssembly(assemblyLocation)
```

```
//C# code
```

```
string assemblyLocation =
typeof(UltraGridExtensions.UltraGridUtil).Assembly.Location;
mainWindow.LoadAssembly(assemblyLocation);
```

UltraGridUtil クラスのコードが AUT 内にある場合は、次のコードをテスト スクリプトに追加して、GetContents メソッドを呼び出すことができます。

```
var contents = (IList)
mainWindow.Invoke("UltraGridExtensions.UltraGridUtil.GetContents",
ultraGrid);
```

Invoke メソッドを呼び出す mainWindow オブジェクトは、AUT を特定しているだけなので、同じ AUT の他のオブジェクトに置き換えてもかまいません。

InvokeMethods メソッド

Windows Forms または WPF コントロールでは、InvokeMethods メソッドを使用して、ネストされたメソッドのシーケンスを呼び出すことができます。以下のメソッドを呼び出すことができます。

- MSDN が定義するコントロールのパブリック メソッド。
- MSDN が定義する静的パブリック メソッド。
- ユーザーが定義する任意の型の静的パブリック メソッド。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4NET がサポートするコントロールのメソッドとプロパティ。
- MSDN が定義するコントロールのパブリック メソッドとプロパティ。
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティ。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4NET 型

Silk4NET 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point や Rect など) が含まれます。

- 列挙型

列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが .NET 列挙型 System.Windows.Visibility のパラメータを必要とする場合は、値が Visible、Hidden、または Collapsed の文字列を使用できます。

- .NET 構造体とオブジェクト

.NET 構造体とオブジェクトパラメータはリストとして渡す必要があります。リスト内の要素は、テストアプリケーションの .NET オブジェクトで定義されているコンストラクターの 1 つと一致しなければなりません。たとえば、メソッドが .NET 型 System.Windows.Vector のパラメータを必要とする場合、2 つの整数値を持つリストを渡すことができます。これが機能するのは、System.Windows.Vector 型が 2 つの整数値を引数に取るコンストラクターを持つためです。

- その他のコントロール

コントロールパラメーターは、TestObject として渡したり、返したりできます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4NET 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、C# では null が、VB では Nothing が返されます。

Windows Presentation Foundation (WPF) のサポート

Silk4NET は、Windows Presentation Foundation (WPF) アプリケーションのテストを組み込みでサポートしています。Silk4NET は、スタンドアロン WPF アプリケーションをサポートしており、.NET バージョン 3.5 以降に組み込まれているコントロールを記録し、再生できます。

新機能、サポート対象のプラットフォームとバージョン、既知の問題、および回避策の詳細については、『Silk Test リリース ノート』(<http://supportline.microfocus.com/productdoc.aspx> で入手可能) を参照してください。

サポートするコントロール

WPF テストで使用可能なコントロールの完全な一覧については、「[WPF クラス リファレンス](#)」を参照してください。

Silk4NET WPF がサポートするすべての WPF クラスは、WPFWindow や WPFListBox のように接頭辞 *WPF* で始まります。

WPF コントロールでサポートされるメソッドとプロパティは、実際の実装とランタイム状態によって異なります。メソッドとプロパティは、対応するクラスに対して定義されたリストと異なる場合があります。特定の状況でサポートされるメソッドとプロパティを判別するには、以下のコードを使用します。

- `GetPropertyList()`
- `GetDynamicMethodList()`

WPF の詳細については、[MSDN](#) を参照してください。


WPF クラス リファレンス

WPF アプリケーションを設定すると、Silk4NET は標準の WPF コントロールのテストのサポートを組み込みで提供します。

Windows Presentation Foundation (WPF) アプリケーションの属性

WPF アプリケーションがサポートする属性は次のとおりです。

- *automationId*
- *caption*
- *className*
- *name*
- すべての動的ロケータ属性。

 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード `?` および `*` をサポートしています。

動的ロケータ属性の詳細については、「[動的ロケータ属性](#)」を参照してください。

オブジェクト解決

WPF スクリプト内のコンポーネントを識別するために、*automationId*、*caption*、*className*、あるいは *name* を指定できます。アプリケーション中の要素に指定された *name* が利用可能な場合、ロケータの *automationId* 属性として使用されます。この結果、多くのオブジェクトは、この属性のみを使用して一意に識別できます。たとえば、*automationId* を持つロケータは、以下のようになります：`//`

```
WPFButton[@automationId='okButton']"
```

automationId や他の属性を定義した場合、再生中に *automationId* だけが使用されます。*automationId* が定義されていない場合には、コンポーネントを解決するのに *name* が使用されます。*name* も *automationId* もどちらも定義されていない場合には、*caption* 値が使用されます。*caption* が定義されていない場合は、*className* が使用されます。*automationId* は非常に役立つプロパティであるため、使用することを推奨します。

属性の種類	説明	例
<i>automationId</i>	テスト アプリケーションの開発者によって提供された ID	<code>//WPFButton[@automationId='okButton']"</code>
<i>name</i>	コントロールの名前。Visual Studio デザイナは、デザイナー上で作成された	<code>//WPFButton[@name='okButton']"</code>

属性の種類	説明	例
	すべてのコントロールに自動的に名前を割り当てます。アプリケーション開発者は、アプリケーションのコード上でコントロールを識別するために、この名前を使用します。	
caption	コントロールが表示するテキスト。複数の言語にローカライズされたアプリケーションをテストする場合、caption の代わりに automationId や name 属性を使用することを推奨します。	//WPFButton[@automationId='Ok']"
className	WPF の .NET 単純クラス名 (名前空間なし)。クラス名属性を使用すると、Silk4NET が解決する標準 WPF コントロールから派生したカスタム コントロールを識別するのに役立ちます。	//WPFButton[@className='MyCustomButton']"

Silk4NET は、*automationId*、*name*、*caption*、*className* 属性をこの表に示した順番に使用して WPF コントロールのロケータを記録時に作成します。たとえば、コントロールが *automationId* と *name* を持つ場合、Silk4NET がロケータを作成する際には *automationId* が使用されます。

以下の例では、アプリケーション開発者がアプリケーションの WPF ボタンに対して *name* と *automationId* を XAML コードに定義する方法を示します。

```
<Button Name="okButton" AutomationProperties.AutomationId="okButton"
Click="okButton_Click">Ok</Button>
```

WPFItemsControl クラスから派生したクラス

Silk4NET は、2つの方法を使用して WPFItemsControl から派生したクラス (WPFListBox、WPFTreeView、WPFMenu など) を操作することができます。

- コントロールでの作業
ほとんどのコントロールには、標準的なユースケースのためのメソッドやプロパティがあります。項目は、テキストや索引によって識別されます。
- WPFListBoxItem、WPFTreeViewItem、WPFMenuItem などの個々の項目での作業
高度なユースケースの場合、個々の項目を使用します。たとえば、リスト ボックスの特定の項目のコンテキスト メニューを開いたり、項目に相対的な場所をクリックしたりする場合に個々の項目を使用します。

カスタム WPF コントロール

一般的に、Silk4NET では、すべての標準 WPF コントロールの記録と再生がサポートされています。

Silk4NET は、カスタム コントロールが実装された方法を基にカスタム コントロールを処理します。次の方法を使用してカスタム コントロールを実装することができます。

- UserControl から派生したクラスを定義する

複合コントロールを作成する典型的な方法です。Silk4NET は、これらのユーザー コントロールを WPFUserControl として認識し、含まれるコントロールを完全にサポートしています。

- ListBox などの標準 WPF コントロールから派生したクラスを定義する

Silk4NET は、これらのコントロールを派生元の標準 WPF コントロールのインスタンスとして扱います。ユーザー コントロールの振る舞いがその基底クラスの実装と大きく異なる場合には、子の記録、再生、解決は機能しない可能性があります。

- テンプレートを使用して視覚デザインを変更した標準コントロールを使用する

低レベルの再生が機能しない可能性があります。その場合には、「高レベル」再生モードに切り替えます。

Silk4NET は、一般的に機能テストに無関係なコントロールは除外します。たとえば、レイアウトを目的として使用されるコントロールは含まれません。しかし、カスタム コントロールが除外されたクラスから派生している場合、除外されたコントロールを記録/再生の対象とするためには、関連する WPF クラスの名前を指定します。

WPF メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティの取得、またはプロパティの設定を直接実行できます。また、このコントロールの Silk4NET API で使用できないメソッドおよびプロパティも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4NET API を通じて公開されていない場合に特に便利です。


オブジェクトの動的メソッドは Invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、GetDynamicMethodList メソッドを使用します。


オブジェクトの複数の動的メソッドは InvokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、GetDynamicMethodList メソッドを使用します。

動的プロパティの取得には GetProperty メソッドを、動的プロパティの設定には SetProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、GetPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.Invoke("SetTitle", "my new title")
```

 **注:** 通常、ほとんどのプロパティは読み取り専用で、設定できません。

 **注:** ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティを取得する場合、Reflection を使用します。

Invoke メソッド

Windows Forms または WPF コントロールでは、Invoke メソッドを使用して、以下のメソッドを呼び出すことができます。

- MSDN が定義するコントロールのパブリック メソッド。
- MSDN が定義する静的パブリック メソッド。
- ユーザーが定義する任意の型の静的パブリック メソッド。

Invoke メソッドの最初の例

Silk4NET の DataGrid 型のオブジェクトでは、MSDN が System.Windows.Forms.DataGrid 型に定義しているすべてのメソッドを呼び出すことができます。

System.Windows.Forms.DataGrid クラスのメソッド IsExpanded を呼び出すには、次のコードを使用します。

```
//VB .NET code
Dim isExpanded As Boolean = dataGrid.Invoke("IsExpanded", 3)

//C# code
bool isExpanded = (bool) dataGrid.Invoke("IsExpanded", 3);
```

Invoke メソッドの 2 番目の例

AUT 内の静的メソッド String.Compare(String s1, String s2) を呼び出すには、次のコードを使用します。

```
//VB .NET code
Dim result as Integer = (Integer)
mainWindow.Invoke("System.String.Compare", "a", "b")

//C# code
int result = (int) mainWindow.Invoke("System.String.Compare", "a", "b");
```

Invoke メソッドの 3 番目の例

この例では、ユーザーが生成したメソッド GetContents を動的に呼び出す方法を示します。

テスト対象アプリケーション (AUT) のコントロールの操作に使用するコードを作成できます (この例では UltraGrid)。UltraGrid の内容を取得するために、複雑な動的呼び出しを作成するのではなく、新しいメソッド GetContents を生成し、この新しいメソッドを動的に呼び出すことができます。

Visual Studio で、AUT 内の次のコードによって GetContents メソッドを UltraGridUtil クラスのメソッドとして定義します。

```
//C# code, because this is code in the AUT
namespace UltraGridExtensions {
    public class UltraGridUtil {
        /// <summary>
        /// Retrieves the contents of an UltraGrid as nested list
        /// </summary>
        /// <param name="grid"></param>
        /// <returns></returns>
        public static List<List<string>>
        GetContents(Infragistics.Win.UltraWinGrid.UltraGrid grid) {
            var result = new List<List<string>>();
            foreach (var row in grid.Rows) {
                var rowContent = new List<string>();
                foreach (var cell in row.Cells) {
                    rowContent.Add(cell.Text);
                }
                result.Add(rowContent);
            }
            return result;
        }
    }
}
```

UltraGridUtil クラスのコードを AUT に追加する必要があります。これは、次のようにして行います。

- アプリケーション開発者は、クラスのコードを AUT にコンパイルできます。アセンブリがすでにロードされている必要があります。
- テストの実行時に AUT にロードされる新しいアセンブリを作成できます。

アセンブリをロードするには、次のコードを使用します。

```
FormsWindow.LoadAssembly(String assemblyFileName)
```

フルパスを使用して、アセンブリをロードできます。例：

```
mainWindow.LoadAssembly("C:/temp/ultraGridExtensions.dll")
```

Location メソッドを使用してアセンブリの場所を見つけることもできます。

```
//VB.NET code
Dim assemblyLocation =
GetType(UltraGridExtensions.UltraGridUtil).Assembly.Location
mainWindow.LoadAssembly(assemblyLocation)
```

```
//C# code
string assemblyLocation =
typeof(UltraGridExtensions.UltraGridUtil).Assembly.Location;
mainWindow.LoadAssembly(assemblyLocation);
```

UltraGridUtil クラスのコードが AUT 内にある場合は、次のコードをテスト スクリプトに追加して、GetContents メソッドを呼び出すことができます。

```
var contents = (IList)
mainWindow.Invoke("UltraGridExtensions.UltraGridUtil.GetContents",
ultraGrid);
```

Invoke メソッドを呼び出す mainWindow オブジェクトは、AUT を特定しているだけなので、同じ AUT の他のオブジェクトに置き換えてもかまいません。

InvokeMethods メソッド

Windows Forms または WPF コントロールでは、InvokeMethods メソッドを使用して、ネストされたメソッドのシーケンスを呼び出すことができます。以下のメソッドを呼び出すことができます。

- MSDN が定義するコントロールのパブリック メソッド。
- MSDN が定義する静的パブリック メソッド。
- ユーザーが定義する任意の型の静的パブリック メソッド。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4NET がサポートするコントロールのメソッドとプロパティ。
- MSDN が定義するコントロールのパブリック メソッドとプロパティ。
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティ。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4NET 型

Silk4NET 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point や Rect など) が含まれます。

- 列挙型

列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが .NET 列挙型 System.Windows.Visibility のパラメータを必要とする場合は、値が Visible、Hidden、または Collapsed の文字列を使用できます。

- .NET 構造体とオブジェクト

.NET 構造体とオブジェクトパラメータはリストとして渡す必要があります。リスト内の要素は、テストアプリケーションの .NET オブジェクトで定義されているコンストラクターの 1 つと一致しなければなりません。たとえば、メソッドが .NET 型 System.Windows.Vector のパラメータを必要とする場合、2 つの整数値を持つリストを渡すことができます。これが機能するのは、System.Windows.Vector 型が 2 つの整数値を引数に取るコンストラクターを持つためです。

- WPF コントロール

WPF コントロールパラメータは TestObject として渡すことができます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4NET 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、C# では null が、VB では Nothing が返されます。
- すべてのその他の型の場合は文字列

返された .NET オブジェクトに対して ToString を呼び出せば、文字列表現を取得できます。

例

たとえば、アプリケーション開発者が次のメソッドとプロパティを持つ Calculator カスタム コントロールを作成したとします。

```
public void Reset()
public int Add(int number1, int number2)
public System.Windows.Vector StrechVector(System.Windows.Vector
vector, double
factor)
public String Description { get;}
```

テスト担当者は、テスト内からメソッドを直接呼び出すことができます。例：

```
customControl.Invoke("Reset")
Dim sum as Integer = customControl.Invoke("Add", 1, 2)
' the vector can be passed as list of integer
Dim vector = New List(Of Integer)
vector.Add(3)
vector.Add(4)
' returns "6;8" because this is the string representation of the .NET
object
Dim stretchedVector As String = customControl.Invoke("StrechVector",
vector, 2.0)
Dim description As String = customControl.GetProperty("Description")
```

記録/再生の対象とする WPF クラスの設定

Silk4NET は、一般的に機能テストに無関係なコントロールは除外します。たとえば、レイアウトを目的として使用されるコントロールは含まれません。しかし、カスタム コントロールが除外されたクラスから派生している場合、除外されたコントロールを記録/再生の対象とするためには、関連する WPF クラスの名前を指定します。

記録や再生の対象にしたい WPF クラスの名前を指定します。たとえば、MyGrid というカスタム クラスが WPF Grid クラスから継承された場合、MyGrid カスタム クラスのオブジェクトは記録や再生に使用できません。Grid クラスはレイアウト目的のためだけに存在し、機能テストとは無関係であるため、Grid オ

プロジェクトは記録や再生に使用できません。この結果、Grid オブジェクトはデフォルトでは公開されません。機能テストに無関係なクラスに基づいたカスタム クラスを使用するには、カスタム クラス (この場合は *MyGrid*) を **OPT_WPF_CUSTOM_CLASSES** オプションに追加します。これによって、記録、再生、検索、プロパティの検証など、すべてのサポートされる操作を指定したクラスに対して実行できるようになります。

1. **Silk4NET > スクリプト オプション** をクリックします。
2. **オプション** メニュー ツリーの **記録** の隣にあるプラス記号 (+) をクリックします。 **記録** オプションが右側のパネルに表示されます。
3. **WPF** をクリックします。
4. **カスタム WPF クラス名** グリッドで、記録や再生中に公開するクラスの名前を入力します。複数のクラス名を指定する場合にはカンマで区切ります。
5. **OK** をクリックします。

Silverlight アプリケーションのサポート

Microsoft Silverlight (Silverlight) は、リッチ インターネット アプリケーションを記述し、実行するためのアプリケーション フレームワークで、Adobe Flash と同様の機能と目的を備えています。Silverlight の実行時環境は、大部分の Web ブラウザでプラグインとして使用できます。

Silk4NET は、Silverlight アプリケーションのテストを組み込みでサポートしています。Silk4NET は、ブラウザ内部と同様ブラウザ外部でも実行される Silverlight アプリケーションをサポートしており、.NET バージョン 3.5 以降でコントロールを記録し、再生できます。

Silverlight をベースとする以下のアプリケーションがサポートされます。

- Internet Explorer で実行される Silverlight アプリケーション
- Mozilla Firefox 4.0 以降で実行される Silverlight アプリケーション
- Out-of-Browser Silverlight アプリケーション


サポートするコントロール

Silk4NET は、Silverlight コントロールの記録と再生をサポートしています。

Silverlight テストで使用可能なコントロールの完全な一覧については、「*Silverlight* クラス リファレンス」を参照してください。

前提条件

Silverlight アプリケーションのテストを Microsoft Windows XP 上でサポートするには、サービスパック 3 をインストールし、Windows 7 で提供される MSUIA (Microsoft User Interface Automation) の Windows XP 用の更新プログラムを適用する必要があります。更新プログラムは、<http://www.microsoft.com/download/en/details.aspx?id=13821> からダウンロードできます。

 **注:** Silverlight サポートには、MSUIA のインストールが必須です。Windows OS 上で Silverlight サポートが機能しない場合は、利用中のオペレーティング システムに一致した MSUIA の更新プログラムを <http://support.microsoft.com/kb/971513> からダウンロードしてインストールしてください。


Silverlight クラス リファレンス

Silverlight アプリケーションを設定すると、Silk4NET は標準の Silverlight コントロールのテストのサポートを組み込みで提供します。

Silverlight コントロールを識別するためのロケータ属性

Silverlight コントロールでサポートされているロケータ属性は次のとおりです。

- *automationId*
- *caption*
- *className*
- *name*
- すべての動的ロケータ属性

 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード ? および * をサポートしています。


動的ロケータ属性の詳細については、「動的ロケータ属性」を参照してください。

Silverlight スクリプト内のコンポーネントを識別するために、*automationId*、*caption*、*className*、*name*、または任意の動的ロケータ属性を指定できます。*automationId* はアプリケーション開発者が設定します。たとえば、*automationId* を持つロケータは、以下のようになります：

```
// SLButton[@automationId="okButton"]
```

automationId は一般に非常に有用で安定した属性であるため、使用することを推奨します。

属性の種類	説明	例
<i>automationId</i>	テスト対象アプリケーションの開発者によって設定される識別子。Visual Studio デザイナは、デザイナー上で作成されたすべてのコントロールに自動的に <i>automationId</i> を割り当てます。アプリケーション開発者は、アプリケーションのコード上でコントロールを識別するために、この ID を使用します。	<pre>// SLButton[@automationId="okButton"]</pre>
<i>caption</i>	コントロールが表示するテキスト。複数の言語にローカライズされたアプリケーションをテストする場合、 <i>caption</i> の代わりに <i>automationId</i> や <i>name</i> 属性を使用することを推奨します。	<pre>//SLButton[@caption="Ok"]</pre>
<i>className</i>	Silverlight コントロールの .NET 単純クラス名 (名前空間なし)。 <i>className</i> 属性を使用すると、Silk4NET が解決する標準 Silverlight コントロールから派生したカスタム コントロールを識別するのに役立ちます。	<pre>// SLButton[@className='MyCustomButton']</pre>
<i>name</i>	コントロールの名前。テスト対象アプリケーションの開発者によって設定されます。	<pre>//SLButton[@name="okButton"]</pre>

 **注目:** XAML コードの *name* 属性は、ロケータ属性 *name* ではなく、ロケータ属性 *automationId* にマップされます。

Silk4NET は、*automationId*、*name*、*caption*、*className* 属性をこの表に示した順番に使用して Silverlight コントロールのロケータを記録時に作成します。たとえば、コントロールが *automationId* と *name* を持つ場合、*automationId* が固有の場合は Silk4NET がロケータを作成する際に使用されません。

以下の表は、アプリケーション開発者がテキスト「Ok」を持つ Silverlight ボタンをアプリケーションの XAML コードに定義する方法を示しています。

オブジェクトの XAML コード	Silk Test からオブジェクトを検索するためのロケータ
<code><Button>Ok</Button></code>	<code>//SLButton[@caption="Ok"]</code>
<code><Button Name="okButton">Ok</Button></code>	<code>//SLButton[@automationId="okButton"]</code>
<code><Button AutomationProperties.AutomationId="okButton">Ok</Button></code>	<code>//SLButton[@automationId="okButton"]</code>

オブジェクトの XAML コード	Silk Test からオブジェクトを検索するためのロケータ
<Button AutomationProperties.Name="okButton">Ok< /Button>	//SLButton[@name="okButton"]

Silverlight メソッドの動的呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティの取得、またはプロパティの設定を直接実行できます。また、このコントロールの Silk4NET API で使用できないメソッドおよびプロパティも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4NET API を通じて公開されていない場合に特に便利です。

オブジェクトの動的メソッドは Invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、GetDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは InvokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、GetDynamicMethodList メソッドを使用します。

動的プロパティの取得には GetProperty メソッドを、動的プロパティの設定には SetProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、GetPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.Invoke("SetTitle", "my new title")
```



注： 通常、ほとんどのプロパティは読み取り専用で、設定できません。



注： ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティを取得する場合、Reflection を使用します。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4NET 型。

Silk4NET 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point、Rect など) が含まれます。

- 列挙型。

列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが .NET 列挙型 System.Windows.Visibility のパラメータを必要とする場合には、Visible、Hidden、Collapsed の文字列値を使用できます。

- .NET 構造体とオブジェクト。

.NET 構造体とオブジェクトパラメータはリストとして渡します。リスト内の要素は、テスト アプリケーションの .NET オブジェクトで定義されているコンストラクターの 1 つと一致しなければなりません。たとえば、メソッドが .NET 型 System.Windows.Vector のパラメータを必要とする場合、2 つの整数値を持つリストを渡すことができます。これが機能するのは、System.Windows.Vector 型が 2 つの整数値を引数に取るコンストラクターを持つためです。

- その他のコントロール。

コントロールパラメータは TestObject として渡すことができます。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- MSDN が定義する AutomationElement クラスのすべてのパブリック メソッドとプロパティ。詳細については、<http://msdn.microsoft.com/ja-jp/library/system.windows.automation.automationelement.aspx> を参照してください。
- MSUIA が公開するすべてのメソッドとプロパティ。利用可能なメソッドとプロパティは「パターン」で分類されます。パターンとは、MSUIA 固有の用語です。すべてのコントロールは、いくつかのパターンを実装します。一般的なパターンについての概要およびすべての利用可能なパターンについては、<http://msdn.microsoft.com/ja-jp/library/ms752362.aspx> を参照してください。カスタムコントロールの開発者は、MSUIA パターン セットを実装することによって、カスタム コントロールのテスト サポートを提供できます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4NET 型の場合は正しい値。
- 戻り値を持たないすべてのメソッドの場合は、NULL が返されます。
- すべてのその他の型の場合は文字列。

この文字列表現を取得するには、テスト対象アプリケーションの返された .NET オブジェクトに対して ToString メソッドを呼び出します。

例

Silverlight の TabItem。これは TabControl の項目です。

```
tabItem.Invoke("SelectedItemPattern.Select")  
mySilverlightObject.GetProperty("IsPassword")
```

Silverlight でのスクロール

Silk4NET では、Silverlight コントロールに応じて、2 種類のスクロール方法とプロパティを提供します。

- 1 つめの種類のコントロールには、それ自体でスクロール可能なコントロールが含まれ、スクロールバーは子として明示的に表示されません。たとえば、コンボ ボックス、ペイン、リスト ボックス、ツリー コントロール、データグリッド、オート コンプリット ボックスなどがあります。
- 2 つめの種類のコントロールには、それ自体ではスクロール不可能なコントロールが含まれ、スクロール用にスクロールバーが子として表示されます。たとえば、テキスト フィールドがあります。

Silk4NET にこのような違いがあるのは、Silk4NET のコントロールがこの 2 通りの方法でスクロールを実装するためです。

スクロールをサポートするコントロール

この場合、スクロール方法とプロパティは、スクロールバーを含むコントロールで使用できます。したがって、Silk4NET ではスクロールバー オブジェクトは表示されません。

使用例

以下のコマンドでは、リスト ボックスが一番下までスクロールされます。

```
listBox.SetVerticalScrollPercent(100)
```

以下のコマンドでは、リスト ボックスが 1 ユニットずつ下方へスクロールされます。

```
listBox.ScrollVertical(ScrollAmount.SmallIncrement)
```

スクロールをサポートしないコントロール

この場合、スクロールバーが表示されます。コントロール自体で可能なスクロール方法とプロパティはありません。水平スクロールバーと垂直スクロールバーの各オブジェクトを使用すると、対応する API 関数でパラメータとして増分または減分、または最終位置を指定することでコントロール内をスクロールできます。増分または減分として ScrollAmount 列挙の値を使用できます。詳細については、Silverlight の製品マニュアルを参照してください。最終位置は、オブジェクトの位置に関連し、アプリケーション設計者によって定義されます。

使用例

以下のコマンドでは、テキスト ボックス内の垂直スクロールバーが 15 の位置までスクロールされます。

```
textBox.SLVerticalScrollBar().ScrollToPosition(15)
```

以下のコマンドでは、テキスト ボックス内の垂直スクロールバーが一番下までスクロールされます。

```
textBox.SLVerticalScrollBar().ScrollToMaximum()
```

Silverlight 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

Silk4NET で Silverlight アプリケーションの内部を確認できず、記録時に緑色の四角形が描画されない。次の理由により、Silk4NET は Silverlight アプリケーションの内部を確認できなくなっています。

原因	解決策
使用している Mozilla Firefox のバージョンが 4.0 以前です。	Mozilla Firefox 4.0 以降を使用してください。
使用している Silverlight のバージョンが 3 以前です。	Silverlight 3 (Silverlight Runtime 4) または Silverlight 4 (Silverlight Runtime 4) を使用してください。
使用している Silverlight アプリケーションがウィンドウレスモードで実行されています。	Silk4NET は、ウィンドウレス モードで実行される Silverlight アプリケーションをサポートしません。このようなアプリケーションをテストするには、Silverlight アプリケーションが実行されている Web サイトを変更する必要があります。したがって、Silverlight アプリケーションがホストされている HTML または ASPX ファイルのオブジェクト タグの windowless パラメータを false に設定する必要があります。 以下のコードは、windowless パラメータを false に設定する例を示します。 <pre><object ...> <param name="windowless" value="false"/> ... </object></pre>

Rumba のサポート

Rumba は、世界トップクラスの Windows デスクトップ端末エミュレーション ソリューションです。Silk Test は、Rumba の記録および再生を組み込みでサポートしています。

Rumba でのテスト時には、以下の点を考慮してください。

- Rumba のバージョンは、Silk Test のバージョンと互換性がある必要があります。バージョン 8.1 以前の Rumba はサポートされていません。
- Rumba のグリーン スクリーンの周囲にあるコントロールはすべて WPF の基本機能 (または Win32) を使用しています。
- サポートされている Rumba デスクトップ タイプは、以下のとおりです。
 - メインフレーム ディスプレイ
 - AS400 ディスプレイ

Rumba テストで使用できる記録および再生のコントロールの完全な一覧については、「*Rumba クラス リファレンス*」を参照してください。

Rumba クラス リファレンス

Rumba アプリケーションを設定すると、Silk4NET は標準の Rumba コントロールのテストのサポートを組み込みで提供します。

Rumba の有効化と無効化

Rumba は、世界トップクラスの Windows デスクトップ端末エミュレーション ソリューションです。Rumba は、メインフレーム、ミッドレンジ、UNIX、Linux、および HP サーバーとの接続ソリューションを提供します。

サポートの有効化

Rumba スクリプトを記録および再生する前に、サポートを有効にする必要があります。

1. Rumba デスクトップ クライアント ソフトウェアバージョン 8.1 以降をインストールします。
2. **スタート > プログラム > Silk > Silk Test > 管理 > Rumba プラグイン > Silk Test Rumba プラグインの有効化** をクリックします。

サポートの無効化

スタート > プログラム > Silk > Silk Test > 管理 > Rumba プラグイン > Silk Test Rumba プラグインの無効化 をクリックします。


Rumba コントロールを識別するためのロケーター属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。サポートされている属性は次のとおりです。

caption	コントロールが表示するテキスト。
priorlabel	フォームの入力フィールドには通常入力の目的を説明するラベルがあるため、 priorlabel の目的は隣接するラベル フィールド RumbaLabel のテキストによってテキスト入力フィールド RumbaTextField を識別することです。テキスト フィールドの同じ行の直前にラベルがない場合、または右側のラベルが左側のラベルよりテキスト フィールドに近い場合、テキスト フィールドの右側にあるラベルが使用されます。
StartRow	この属性は記録されていませんが、手動でロケーターに追加することができます。 StartRow を使用して、この行で始まるテキスト入力フィールド、 RumbaTextField を識別します。

StartColumn この属性は記録されていませんが、手動でロケータに追加することができます。**StartColumn** を使用して、この列で始まるテキスト入力フィールド、**RumbaTextField** を識別します。

すべての動的ロケータ属性。 動的ロケータ属性の詳細については、「動的ロケータ属性」を参照してください。

 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード ? および * をサポートしています。

Rumba での画面検証の使用


Rumba に対する画面検証を自動的に挿入するには、**オプション** ダイアログ ボックスで **記録 > 全般 > 画面検証を記録する** をオンにします。


画面検証を手動で挿入するには、以下を実行します。

1. テストで、**検証タイプのロジックの作成** ボタンをクリックし、**テスト ロジック デザイナ - 検証** を開きます。
2. **次へ** をクリックします。
3. **画面のコンテンツ** を選択します。
ツール > オプション > 記録 > Rumba > 除外オブジェクト で特定されるすべての除外オブジェクトが使用されます。この手順を完了後に、テストの **プロパティ** ウィンドウでこれらをさらにカスタマイズできます。
4. **次へ** をクリックします。
5. **識別** ボタンをクリックします。
6. 識別する Rumba 画面でコントロールを選択します。画面全体がキャプチャされます。
7. **次へ** をクリックします。
8. **完了** をクリックします。

SAP のサポート

Silk4NET は、Windows ベースの GUI モジュールを基にした SAP クライアント/サーバー アプリケーションのテストを組み込みでサポートしています。

 **注:** Silk4NET のプレミアム ライセンスを所有している場合にのみ、Silk4NET で SAP アプリケーションをテストできます。ライセンス モードについての詳細は、「ライセンス情報」を参照してください。

 **注:** Internet Explorer や Firefox で SAP NetWeaver を使用する場合、Silk4NET は、xBrowser テクノロジ ドメインを使用してアプリケーションをテストします。

 **注:** 最新のバージョンについての情報と既知の問題についてはリリース ノートを確認してください。

サポートするコントロール


SAP のテストで利用可能な記録および再生コントロールの完全な一覧については、「SAP クラス リファレンス」を参照してください。

サポートされている属性の一覧については、「SAP アプリケーションの属性」を参照してください。

SAP クラス リファレンス

SAP アプリケーションを設定すると、Silk4NET は標準の SAP コントロールをテストするためのサポートを組み込みで自動的に提供します。

SAP クラス リファレンスに含まれるクラス (インクルードしたプロパティとメソッドを含む) は、Silk4NET から直接アクセス可能な SAP オートメーション モジュールの一部です。


 **注:** そのインターフェイスや、インターフェイスの基本的なアルゴリズムおよび動作は、Silk4NET によって制御されません。

SAP アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

SAP がサポートする属性は次のとおりです。

- automationId
- caption

 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

SAP メソッドの動的な呼び出し

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4NET がサポートするコントロールのメソッドとプロパティ。
- SAP オートメーション インターフェイスによって定義されているすべての public メソッド
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティ。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4NET 型

Silk4NET 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point や Rect など) が含まれます。

- UI コントロール

UI コントロールは、TestObject として渡したり、返したりできます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4NET 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、C# では null が、VB では Nothing が返されます。

SAP コントロールの動的呼び出し

Silk4NET で SAP コントロールに対する操作を記録できない場合、SAP で利用できるレコーダーで操作を記録してから、記録されたメソッドを Silk4NET スクリプトで動的に呼び出すことができます。これによって、記録できない SAP コントロールに対する操作を再生できます。

1. コントロールに対して実行する操作を記録するには、SAP で利用できる **SAP GUI スクリプト作成** ツールを使用できます。
SAP GUI スクリプト作成 ツールの詳細については、SAP のドキュメントを参照してください。
2. 記録された操作を **SAP GUI スクリプト作成** ツールによって保存された場所から開き、記録されたメソッドを確認します。
3. Silk4NET で、記録されたメソッドをスクリプトから動的に呼び出します。

使用例

たとえば、SAP UI で *Test* というラベルが付いた、ボタンとリストボックスの組み合わせである特別なコントロールを押し、コントロールのサブメニュー *subsub2* を選択する操作を再生したい場合、SAP で利用できるレコーダーでこの操作を記録できます。結果のコードは、次のようになります。

```
session.findById("wnd[0]/usr/cntlCONTAINER/shellcont/shell").pressContextButton "TEST"  
session.findById("wnd[0]/usr/cntlCONTAINER/shellcont/shell").selectContextMenuItem "subsub2"
```

これにより、Silk4NET で、スクリプト内で次のコードを使用して、メソッド `pressContextButton` と `selectContextMenuItem` を動的に呼び出すことができます。

```
.SapToolBarControl("shell ToolbarControl").Invoke("pressContextButton",  
"TEST")  
.SapToolBarControl("shell ToolbarControl").Invoke("selectContextMenuItem",  
"subsub2")
```

このコードを再生すると、SAP UI のコントロールが押され、サブメニューが選択されます。

SAP の自動セキュリティ設定の構成

SAP アプリケーションを起動する前に、セキュリティ警告設定を構成する必要があります。このようにしないと、テストで SAP アプリケーションが再生されるたびにセキュリティ警告「スクリプトから GUI に接続しようとしています」が表示されます。

1. Windows の **コントロール パネル** で **SAP システム設定** を選択します。 **SAP システム設定** ダイアログ ボックスが開きます。
2. **デザイン選択** タブで、**スクリプトが実行中 SAP GUI に追加されるとき通知** をオフにします。

Windows API ベースのアプリケーションのサポート

Silk4NET は、Microsoft Windows API ベースのアプリケーションのテストを組み込みでサポートしています。アクセシビリティを有効にすると Microsoft のアプリケーションのいくつかのオブジェクトが Silk4NET によってより詳細に認識されます。たとえば、アクセシビリティを有効にしないと、Silk4NET は Microsoft Word のメニューバーおよびバージョン 7.0 より後の Internet Explorer に表示されるタブについて基本的な情報のみを記録します。ただし、アクセシビリティを有効にすると、Silk4NET によってそれらのオブジェクトがすべて認識されます。必要な場合、新しいウィンドウを定義すると、Silk4NET によるオブジェクトの認識を向上させることもできます。

新機能、サポート対象のプラットフォームとバージョン、既知の問題、および回避策の詳細については、『Silk Test リリース ノート』(<http://supportline.microfocus.com/productdoc.aspx> で入手可能) を参照してください。

サポートするコントロール

Windows ベースのテストで利用可能な記録および再生コントロールの完全な一覧については、「Windows API ベースのクラス リファレンス」を参照してください。

Win32 クラス リファレンス

Win32 アプリケーションを設定すると、Silk4NET は標準の Windows API ベースのコントロールのテストのサポートを組み込みで提供します。

Windows API ベースのクライアント/サーバー アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Windows API ベースのクライアント/サーバー アプリケーションがサポートする属性は次のとおりです。

- caption
- windowid
- priorlabel : 隣接するラベル フィールドのテキストによってテキスト入力フィールドを識別します。通常、フォームのすべての入力フィールドに、入力の目的を説明するラベルがあります。caption のないコントロールの場合、自動的に属性 **priorlabel** がロケーターに使用されます。コントロールの **priorlabel** 値 (テキスト ボックスなど) には、コントロールの左側または上にある最も近いラベルの caption が使用されます。



注: 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Win32 テクノロジ ドメインにおける priorLabel の決定方法

Win32 テクノロジ ドメインにおいて priorLabel を決定する場合、同じウィンドウ内のすべてのラベルとグループが対象のコントロールとみなされます。以下の条件に従って、コントロールが決定されます。

- コントロールの上または左側にあるラベル、およびコントロールを囲むグループが priorLabel の候補とみなされます。
- 最も単純なケースでは、コントロールに最も近いラベルが priorLabel として使用されます。
- コントロールからの距離が等しい 2 つのラベルが存在する場合、次の条件に基づいて priorLabel が決定されます。
 - 一方のラベルがコントロールの左側にあり、他方が上にある場合、左側のものが優先されます。
 - 両方のラベルがコントロールの左側にある場合、上にあるものが優先されます。
 - 両方のラベルがコントロールの上にある場合、左側のものが優先されます。
- 最も近いコントロールがグループ コントロールである場合、まずグループ内のすべてのラベルが上記の規則に従って決定されます。グループ内に適切なラベルが見つからない場合は、グループのキャプションが priorLabel として使用されます。

xBrowser のサポート

xBrowser テクノロジ ドメインを使用して、以下を使用する Web アプリケーションをテストします。

- Internet Explorer
- Mozilla Firefox
- Google Chrome
- 埋め込みブラウザ コントロール

xBrowser テクノロジ ドメインでは、プレーン HTML ページのテスト以外に、AJAX ページもサポートされています。AJAX ページを使用する場合は、オブジェクト認識と同期を行うために、他の高度な方法が必要となります。



注: Internet Explorer を使用し、Web アプリケーションのテストを記録する必要があります。別のサポート対象ブラウザを使用するテストを作成するには、Internet Explorer で記録して別のブラウザで再生します。または、**オブジェクトの識別** ダイアログ ボックスを使用して使用するサポート対象ブラウザでロケータを識別し、そのブラウザに対するテストを手動で作成できます。



注: Web アプリケーションを記録または再生する前に、システムにインストールされているすべてのブラウザアドオンを無効にします。Internet Explorer でアドオンを無効にするには、**ツール > インターネット オプション** をクリックし、**プログラム** タブをクリックし、**アドオンの管理** をクリックし、アドオンを選択してから **無効にする** をクリックします。

サポート対象バージョン、既知の問題、回避策の詳細については、リリース ノート を参照してください。

サンプル アプリケーション

Silk Test のサンプル Web アプリケーションには、以下の URL からアクセスします。

- <http://demo.borland.com/InsuranceWebExtJS/>
- <http://demo.borland.com/gmopost>

xBrowser 用のテスト オブジェクト

Silk4NET では、以下のクラスを使用して Web アプリケーションがモデル化されます。

クラス	説明
BrowserApplication	は、Web ブラウザのメイン ウィンドウを公開し、タブ化するための方法を提供します。
BrowserWindow	は、タブおよび埋め込みブラウザ コントロールへのアクセスを提供し、異なるページに移動するための方法を提供します。
DomElement	は、Web アプリケーションの DOM ツリー (フレームを含む) を提供し、すべての DOM 属性へのアクセスを提供します。一部の DOM 要素では、特殊なクラスを使用できます。

xBrowser オブジェクト用のオブジェクト解決

xBrowser テクノロジ ドメインでは、動的オブジェクト解決がサポートされています。

つまり、テストでロケータ文字列を使用して、オブジェクトの検索と識別が行われます。一般的なロケータには、`"/LocatorName[@locatorAttribute='value']"` のようにロケータ名と少なくとも 1 つのロケータ属性が含まれます。

ロケータ名 Java SWT などの他の種類のテクノロジーでは、テスト オブジェクトのクラス名を使用してロケータ名が作成されます。xBrowser では、DOM 要素のタグ名もロケータ名として使用できます。以下のロケータは、同じ要素を示しています。

1. タグ名を使用した場合：`"//a[@href='http://www.microfocus.com']"`
2. クラス名を使用した場合：`"//DomLink[@href='http://www.microfocus.com']"`


再生速度を最適化するには、クラス名ではなくタグ名を使用します。

ロケータ属性 すべての DOM 属性は、ロケータ文字列属性として使用できます。たとえば、要素 `<button automationid='123'>Click Me</button>` はロケータ `"//button[@automationid='123']"` を使用して識別できます。

ロケータの記録 Silk4NET では、テストケースを記録したり、**オブジェクトの識別** ダイアログ ボックスを使用するときに、組み込みロケータ生成プログラムが使用されます。特定のアプリケーションの結果を向上するように、ロケータ生成プログラムを構成することができます。

xBrowser のページ同期

同期は、すべてのメソッド呼び出しの前後に実行されます。メソッド呼び出しは、同期条件が満たされるまで開始せず、終了もしません。

 **注:** プロパティのアクセスは同期されません。

同期モード

Silk4NET には、HTML および AJAX 用の同期モードがあります。


HTML モードを使用すると、すべての HTML ドキュメントが対話的な状態になることが保証されます。このモードでは、単純な Web ページをテストすることができます。Java Script が含まれるより複雑なシナリオが使用される場合は、以下の同期関数を使用して、手動でスクリプトを記述することが必要になることがあります。

- WaitForObject
- WaitForProperty
- WaitForDisappearance
- WaitForChildDisappearance

AJAX モードでは、ブラウザがアイドル状態に類似した状態になるまで待機します。このことは、AJAX アプリケーションまたは AJAX コンポーネントを含むページに対して特に効果的です。AJAX モードを使用すると、同期関数を手動で記述する必要がなくなるため、スクリプト（オブジェクトの表示または非表示を待機したり、特定のプロパティ値を待機するなど）の作成処理が大幅に簡略化されます。また、この自動同期は、スクリプトを手動で適用しないで記録と再生を正常に行うための基礎となります。

トラブルシューティング


AJAX の非同期の特性のため、ブラウザが完全にアイドル状態になることはありません。このため、Silk4NET でメソッド呼び出しの終了が認識されず、特定のタイムアウト時間が経過したあとで、タイムアウト エラーが発生することがまれにあります。この場合は、少なくとも、問題が発生する呼び出しに対して、同期モードを HTML に設定する必要があります。

 **注:** 使用するページ同期メソッドにかかわらず、Flash オブジェクトがサーバーからデータを取得し、計算を実行してデータをレンダリングするテストでは、手動でテストに同期メソッドを追加する必要があります。メソッドを追加しないと、Silk4NET は、Flash オブジェクトが計算を完了するまで待機しません。たとえば、`Thread.sleep(millisecs)` を使用します。

AJAX フレームワークやブラウザによっては、サーバーから非同期にデータを取得するために、特殊な HTTP 要求を継続して出し続けるものがあります。これらの要求により、指定した同期タイムアウトの期

限が切れるまで同期がハングすることがあります。この状態を回避するには、HTML 同期モードを使用するか、問題が発生する要求の URL を **同期除外リスト** 設定で指定します。

監視ツールを使用して、同期の問題により再生エラーが発生するかどうかを判断します。たとえば、FindBugs (<http://findbugs.sourceforge.net/>) を使用して、AJAX 呼び出しが再生に影響を及ぼしているかどうかを判断できます。次に、問題が発生するサービスを **同期除外リスト** に追加します。

 **注:** URL を除外すると、指定した URL を対象とする各呼び出しに対して同期が無効になります。その URL に対して必要な同期は、手動で呼び出す必要があります。たとえば、WaitForObject をテストに手動で追加する必要がある場合があります。手動で数多くの呼び出しを追加することを避けるために、可能なかぎり、最上位の URL ではなく、具体的に対象を絞って URL を除外します。

ページ同期設定の構成

スクリプト オプション ダイアログ ボックスでは、各テストのページ同期設定を個別に構成したり、すべてのテストに適用するグローバルオプションを設定したりできます。

ビジュアル テストの個別の設定を構成するには、テストを記録し、次に、グローバル再生値を上書きするステップを挿入します。たとえば、タイム サービスを除外するには、以下のように入力します。

```
desktop.setOption(CommonOptions.OPT_XBROWSER_SYNC_EXCLUDE_URLS,
    Arrays.asList("timeService"));
```

xBrowser における API 再生とネイティブ再生の比較

Silk4NET では、Web アプリケーション用に API 再生とネイティブ再生がサポートされています。アプリケーションでプラグインまたは AJAX を使用している場合は、ユーザーの入力そのものを使用します。アプリケーションでプラグインまたは AJAX を使用していない場合は、API 再生を使用することをお勧めします。

ネイティブ再生には以下のような利点があります。

- ネイティブ再生では、マウス ポインタを要素上に移動し、対応する要素を押すことによって、エージェントはユーザー入力をエミュレートします。この結果、再生はほとんどのアプリケーションで変更なしで動作します。
- ネイティブ再生では、Flash や Java アプレットなどのプラグイン、および AJAX を使用するアプリケーションをサポートしていますが、高レベルの API 記録はサポートしていません。

API 再生には以下のような利点があります。

- API 再生では、Web ページが onmouseover や onclick などの DOM イベントによって直接実行されます。
- API 再生を使用するスクリプトでは、ブラウザをフォアグラウンドで実行する必要はありません。
- API 再生を使用するスクリプトでは、要素をクリックする前に、要素が表示されるようにスクロールする必要はありません。
- 一般的に、高レベルのユーザー入力は再生中にポップアップ ウィンドウやユーザー対話の影響を受けないため、API スクリプトの信頼性は高くなります。
- API 再生は、ネイティブ再生よりも高速です。

API 再生とネイティブ再生の関数の違い

DomElement クラスには、API 再生とネイティブ再生に対して異なる関数が備えられています。

以下の表に、API 再生とネイティブ再生で使用する関数を示します。

	API 再生	ネイティブ再生
マウス操作	DomClick	Click
	DomDoubleClick	DoubleClick

	API 再生	ネイティブ再生
	DomMouseMove	MoveMouse
		PressMouse
		ReleaseMouse
キーボード操作	使用不可	TypeKeys
特殊な関数	Select	使用不可
	SetText	
	など	

ブラウザの記録オプションの設定


カスタム属性、記録中に無視するブラウザ属性、DOM 関数の代わりに、ユーザーの入力そのものを記録するかどうかを指定します。

Silk4NET には、ロケーターが記録時に一意となり、メンテナンスが容易になるようにする、高度なロケーター生成メカニズムが備えられています。使用するアプリケーションやフレームワークに応じて、最適な結果を得るためにデフォルト設定を変更できます。それぞれのテクノロジーで使用できる任意のプロパティ (整数や倍精度の数値、文字列、項目識別子、列挙値) を、カスタム属性として使用できます。


xBrowser アプリケーションでは、任意のプロパティを取得し、カスタム属性として使用することもできます。最適な結果を得るために、テストで利用する要素にカスタム オートメーション ID を追加します。

1. **Silk4NET > スクリプト オプション** をクリックします。
2. **オプション** メニュー ツリーの **記録** の隣にあるプラス記号 (+) をクリックします。 **記録** オプションが右側のパネルに表示されます。
3. **xBrowser** をクリックします。
4. Web アプリケーションのカスタム属性を追加するには、**カスタム属性** テキスト ボックスに、使用する属性を入力します。

カスタム属性を使用すると、caption や index のような他の属性よりも高い信頼性を得ることができます。これは、caption はアプリケーションを他の言語に翻訳した場合に変更され、index は定義済みのウィジェットより前に他のオブジェクトが追加されると変更される可能性があるためです。

 **注:** Web アプリケーションにカスタム属性を含めるためには、HTML タグとして追加します。たとえば、myAutomationId という属性を追加するには、`<input type='button' myAutomationId='abc' value='click me' />` と入力します。

複数のオブジェクトに同じカスタム属性の値が割り当てられた場合は、そのカスタム属性を呼び出したときにその値を持つすべてのオブジェクトが返されます。たとえば、一意の ID として loginName を 2 つの異なるテキスト フィールドに割り当てた場合は、loginName 属性を呼び出したときに、両方のフィールドが返されます。

 **注:** 属性名の長さは、62 文字までという制限があります。

5. **ロケーター属性名除外リスト** テキスト ボックスで、記録中に無視する属性名を入力します。このリストを使用して、サイズ、幅、高さ、スタイルなどの頻繁に変更される属性を指定します。ワイルドカード '*' および '?' を **ロケーター属性名除外リスト** で使用できます。たとえば、height という名前の属性を記録しない場合には、height 属性名をリストに追加します。複数の属性名を指定する場合にはカンマで区切ります。
6. **ロケーター属性値除外リスト** テキスト ボックスで、記録中に無視する属性値を入力します。たとえば、x-auto という値を持つ属性を記録しない場合には、x-auto 属性値をリストに追加します。

一部の AJAX フレームワークでは、ページが再読み込みされるたびに変わる属性値が生成されます。このリストを使用して、そのような値を無視します。このリストでワイルドカードを使用することもできます。

複数の属性名を指定する場合にはカンマで区切ります。

7. DOM 関数の代わりにユーザーの入力そのものを記録するには、**ネイティブなユーザー入力を記録する** リスト ボックスから、**はい** を選択します。

たとえば、DomClick の代わりに Click を記録し、SetText の代わりに TypeKeys を記録するには、**はい** を選択します。

アプリケーションでプラグインまたは AJAX を使用している場合は、**はい** を指定して、ユーザーの入力そのものを使用します。アプリケーションでプラグインまたは AJAX を使用していない場合は、再生中にブラウザにフォーカスを設定したりブラウザをアクティブにしたりする必要がない高レベル DOM 関数を使用することをお勧めします。テストで DOM 関数を使用すると、より高速になり、信頼性も高まります。

8. **OK** をクリックします。

マウス移動の詳細設定

マウス移動イベントを使用する Web アプリケーション、Win32 アプリケーション、および Windows Forms アプリケーションでマウス移動操作を記録するかどうかを指定します。たとえば、Adobe Flex や Swing など、xBrowser テクノロジ ドメインの子ドメインのマウス移動イベントを記録することはできません。

1. **Silk4NET > スクリプト オプション** をクリックします。
2. **オプション** メニュー ツリーの **記録** の隣にあるプラス記号 (+) をクリックします。 **記録** オプションが右側のパネルに表示されます。
3. **記録** をクリックします。
4. マウス移動操作を記録するには、OPT_RECORD_MOUSEMOVES オプションをオンにします。
Silk4NET では、スクリプトを短くするために、マウスが置かれた要素またはその親が変化するマウスの移動イベントのみが記録されます。
5. マウスの移動操作を記録する場合、MoveMouse 操作が記録される前に、どのくらいの間マウスが不動状態になければならないかを、**マウスの移動記録遅延** テキスト ボックスにミリ秒単位で指定します。
デフォルト値は、200 に設定されています。
マウスの移動操作は、この時間、マウスが静止している場合にのみ記録されます。遅延を短くすると、予期しないマウスの移動操作が増加します。遅延を長くすると、操作を記録するためにマウスを静止しておく必要があります。
6. **OK** をクリックします。

xBrowser のブラウザ構成の設定

いくつかのブラウザ設定は、テストを継続的に安定して実行するのに役立ちます。設定を変更しなくても Silk4NET は動作しますが、ブラウザ設定を変更するにはいくつかの理由があります。

再生速度を向上させる 読み込みに時間を要する Web ページではなく、about:blank をホームページとして使用する

ブラウザの予期しない動作を回避する

- ポップアップ ウィンドウや警告ダイアログ ボックスを無効にする
- オート コンプリート機能を無効にする
- パスワード ウィザードを無効にする

ブラウザの誤動作を防止する 不要なサードパーティ製プラグインを無効にする

以下のセクションでは、対応するブラウザにおけるこれらの設定場所について説明します。

Internet Explorer

ブラウザ設定は、**ツール > インターネット オプション** にあります。以下の表に、調整できるオプションの一覧を示します。

タブ	オプション	構成	コメント
全般	ホーム ページ	about:blank に設定します。	新しいタブの起動時間を最小限に抑えます。
全般	タブ	<ul style="list-style-type: none"> 複数のタブを閉じるときの警告を無効にします。 新しいタブを作成したとき、新しいタブに切り替えます。 	<ul style="list-style-type: none"> 予期しないダイアログ ボックスが表示されないようにします。 このようにしないと、新しいタブを開くリンクが正しく再生されない場合があります。
プライバシ	ポップアップ ブロック	ポップアップ ブロックを無効にします。	Web サイトで新しいウィンドウを開くことができることを確認します。
コンテンツ	オートコンプレット	完全にオフにします。	<ul style="list-style-type: none"> 予期しないダイアログ ボックスが表示されないようにします。 キー入力するときに予期しない動作を回避します。
プログラム	アドオンの管理	最低限必要なアドオンのみを有効にします。	<ul style="list-style-type: none"> サードパーティ製アドオンにはバグが含まれていることがあります。 Silk4NET と互換性がない可能性があります。
詳細設定	設定	<ul style="list-style-type: none"> Internet Explorer の更新について自動的に確認する を無効にします。 スクリプトのデバッグを使用しない (Internet Explorer) を有効にします。 スクリプトのデバッグを使用しない (その他) を有効にします。 自動クラッシュ回復機能を有効にする を無効にします。 スクリプト エラーごとに通知を表示する を無効にします。 すべての ...警告する 設定を無効にします。 	予期しないダイアログ ボックスが表示されないようにします。

Mozilla Firefox

Mozilla Firefox では、タブを「about:config」に移動して、すべての設定を編集することができます。以下の表に、調整できるオプションの一覧を示します。オプションが存在しない場合は、表を右クリックして **新規作成** 選択すると作成できます。

オプション	値	コメント
app.update.auto	false	予期しない動作を回避します (自動更新を無効にします)。
app.update.enabled	false	予期しない動作を回避します (一般の更新を無効にします)。
app.update.mode	0	予期しないダイアログ ボックスが表示されないようにします (新規更新のプロンプトを表示しません)。

オプション	値	コメント
app.update.silent	true	予期しないダイアログ ボックスが表示されないようにします (新規更新のプロンプトを表示しません)。
browser.sessionstore.resume_from_crash	false	予期しないダイアログ ボックス (ブラウザのクラッシュ後の警告) が表示されないようにします。
browser.sessionstore.max_tabs_undo	0	パフォーマンスを向上させます。Session Restore サービスにより追跡される閉じられたタブの数を制御します。
browser.sessionstore.max_windows_undo	0	パフォーマンスを向上させます。Session Restore サービスにより追跡される閉じられたウィンドウの数を制御します。
browser.sessionstore.resume_session_once	false	予期しないダイアログ ボックスが表示されないようにします。次回ブラウザが起動したときに最後に保存されたセッションを復元するかどうかを制御します。
browser.shell.checkDefaultBrowser	false	予期しないダイアログ ボックスが表示されないようにします。Mozilla Firefox がデフォルト ブラウザであるかどうかを確認します。
browser.startup.homepage	「about:blank」	新しいタブの起動時間を最小限に抑えます。
browser.startup.page	0	ブラウザの起動時間を最小限に抑えます (初期タブに開始ページは表示されません)。
browser.tabs.warnOnClose	false	予期しないダイアログ ボックス (複数のタブを閉じるときの警告) が表示されないようにします。
browser.tabs.warnOnCloseOtherTabs	false	予期しないダイアログ ボックス (他のタブを閉じるときの警告) が表示されないようにします。
browser.tabs.warnOnOpen	false	予期しないダイアログ ボックス (複数のタブを開くときの警告) が表示されないようにします。
dom.max_chrome_script_run_time	180	予期しないダイアログ ボックス (XUL コードの実行時間が長すぎる場合の警告、秒単位のタイムアウト) が表示されないようにします。
dom.max_script_run_time	600	予期しないダイアログ ボックス (スクリプト コードの実行時間が長すぎる場合の警告、秒単位のタイムアウト) が表示されないようにします。
dom.successive_dialog_time_limit	0	予期しない このページでこれ以上ダイアログを表示させない ダイアログ ボックスが表示されないようにします。
extensions.update.enabled	false	予期しないダイアログ ボックスが表示されないようにします。拡張の自動更新を無効にします。

Google Chrome

Google Chrome のブラウザ設定を変更する必要はありません。Silk4NET により、適切なコマンドラインパラメータが指定され、自動的に Google Chrome が起動します。

ロケータ生成プログラムを xBrowser 用に構成する

Open Agent には、ロケータが記録時に一意となり、メンテナンスが容易になるようにする、高度なロケータ生成メカニズムが備えられています。使用するアプリケーションやフレームワークに応じて、最適な結果を得るためにデフォルト設定を変更できます。

頻繁には変更されない属性を利用して、適切に定義されたロケータでは、メンテナンス作業が少なく抑えられます。カスタム属性を使用すると、caption や index などの他の属性を使用するよりも高い信頼性を得ることができます。これは、caption はアプリケーションを他の言語に翻訳した場合に変更され、index は他のオブジェクトが追加されると変更される可能性があるためです。

最適な結果を得るために、テストで利用する要素にカスタムオートメーションIDを追加することもできます。Webアプリケーションの場合は、利用した要素に<div myAutomationId="my unique element name" />のような属性を追加できます。この手法によって、ロケーターの変更に伴うメンテナンス作業を回避することができます。

1. **Silk4NET > オプションの編集** をクリックしてから、**カスタム属性** タブをクリックします。
2. カスタムオートメーションIDを使用する場合、**テクノロジー・ドメインを選択します** リストボックスから、**xBrowser** を選択してから、IDをリストに追加します。
カスタム属性リストには、ロケーターに適した属性が含まれます。カスタム属性が利用可能な場合は、ロケーター生成プログラムは、他の属性の前にそれらの属性を使用します。リストの順番は、ロケーター生成プログラムが使用する属性の優先順位を表しています。指定した属性が選択したオブジェクトに対して利用できない場合は、Silk4NETはxBrowserのデフォルトの属性を使用します。
3. **ブラウザー** タブをクリックします。
4. **ロケーター属性名除外リスト** グリッドで、記録中に無視する属性名を入力します。
たとえば、このリストを使用して、size、width、height、styleなどの頻繁に変更される属性を指定します。ロケーター属性名除外リストでは、ワイルドカード'*'および'?'を使用できます。
複数の属性名を指定する場合にはコンマで区切ります。
5. **ロケーター属性値除外リスト** グリッドで、記録中に無視する属性値を入力します。
一部のAJAXフレームワークでは、ページが再読み込みされるたびに変わる属性値が生成されます。このリストを使用して、そのような値を無視します。このリストでワイルドカードを使用することもできます。
複数の属性値を指定する場合にはコンマで区切ります。
6. **OK** をクリックします。
以上で、テストケースを記録したり、手動で作成する準備ができました。

Internet Explorer 以外のブラウザでのスクリプトの再生

Webアプリケーションのスクリプトは、Internet Explorer を使用して作成する必要があります。ただし、必要に応じて別のサポート対象ブラウザでスクリプトを再生できます。

サポート対象バージョン、既知の問題、回避策の詳細については、リリース ノート を参照してください。

1. テストする Web アプリケーションのスクリプトを記録するか、または、手動でスクリプトを作成します。
スクリプトを記録するには、Internet Explorer を使用する必要があります。
2. **Silk4NET > アプリケーション構成の編集** をクリックします。**アプリケーション構成の編集** ダイアログボックスが開きます。
3. **ブラウザーの設定** をクリックします。**基本状態の編集** ダイアログボックスが開きます。
4. **ブラウザーの種類** セクションにあるブラウザー アイコンをクリックして、使用するブラウザーを選択します。
5. **OK** をクリックします。

Google Chrome を使用したテスト再生の前提条件

コマンドラインパラメータ

Google Chrome を使用してテストを再生またはロケーターを記録する場合は、以下のコマンドを使用して Google Chrome を起動します。

```
%LOCALAPPDATA%\%Google%\Chrome\Application\chrome.exe  
--enable-logging
```

```
--log-level=1
--disable-web-security
--disable-hang-monitor
--disable-prompt-on-repost
--dom-automation
--full-memory-crash-report
--no-default-browser-check
--no-first-run
--homepage=about:blank
--disable-web-resources
--disable-preconnect
--enable-logging
--log-level=1
--safebrowsing-disable-auto-update
--test-type=ui
--noerrdialogs
--metrics-recording-only
--allow-file-access-from-files
--disable-tab-closeable-state-watcher
--allow-file-access
--disable-sync
--testing-channel=NamedTestingInterface:st_42
```

ウィザードを使用してアプリケーションに追加する場合は、これらのコマンドラインパラメータは、基本状態に自動的に追加されます。テストを開始したときに、適切なコマンドラインパラメータなしで Google Chrome のインスタンスがすでに実行されている場合、は Google Chrome を終了して、コマンドラインパラメータを使用してブラウザを再起動しようとします。ブラウザを再起動できない場合は、エラーメッセージが表示されます。



注: クロスドメインのドキュメントを記録または再生する場合は、コマンドラインパラメータ `disable-web-security` が必要です。

Google Chrome を使用したテストの制限事項

Google Chrome を使用した再生テストと記録ロケータのサポートは、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。以下のリストに、Google Chrome を使用した再生テストと記録ロケータの既知の制限事項をリストします。

- Silk Test は、Google Chrome を使用した xBrowser ドメインの子テクノロジー ドメインのテストをサポートしていません。たとえば、Adobe Flex または Microsoft Silverlight は Google Chrome ではサポートされていません。
- Silk Test は、Google Chrome のネイティブ サポートは提供しません。内部 Google Chrome 機能をテストすることはできません。たとえば、テストで、Win32 でナビゲーションバーにテキスト追加して現在表示されている Web ページを変更することはできません。回避策として、API コールを使用して Web ページ間を移動できます。Silk Test は警告ダイアログなどのダイアログ ボックスをサポートしています。
- Google Chrome のページ同期は、サポートされている他のブラウザほど高度なものではありません。同期モードを変更しても、Google Chrome の同期は影響を受けません。
- Silk Test は、Google Chrome を使用してアプリケーションをテストする際のメソッド `TextClick` と `TextSelect` をサポートしていません。
- Silk Test は、Google Chrome の認証ダイアログ ボックスの **ログイン** および **キャンセル** ボタンをサポートしていません。以下の回避策のいずれかを使用して、この制限事項を回避できます。
 - テストする Web サイトの URL にユーザー名とパスワードを指定します。たとえば、Web サイト `www.example.com/loginrequired.html` にログインするには、以下のコードを使用します。
`http://myusername:mypassword@example.com/loginrequired.html`

- TypeKeys を使用して、ダイアログ ボックスにユーザー名とパスワードを入力します。たとえば、以下のコードを使用します。

```
desktop.find("//Window[@caption='Authentication Required']/Control[2]").TypeKeys("myusername")
desktop.find("//Window[@caption='Authentication Required']/Control[1]").TypeKeys("mypassword<Enter>")
```



注: Control[2] はユーザー名のフィールドで、Control[1] はパスワードのフィールドです。2 番目の TypeKeys の末尾の <Enter> キーで、ダイアログ ボックスのエントリを確認します。

- Silk Test は、Google Chrome メニューを使用して Google Chrome の **印刷** ダイアログ ボックスが開かれたことは認識しません。Google Chrome でダイアログ ボックスを開く動作を追加してテストするには、TypeKeys メソッドを使用して **Ctrl+Shift+P** を送信する必要があります。Internet Explorer はこのショートカットを認識しません。したがって、最初に Internet Explorer にテストを記録してから、手動で **Ctrl+Shift+P** を押す操作をテストに追加する必要があります。
- 2 つの Google Chrome ウィンドウが同時に開いているときに、2 番目のウィンドウが最初のウィンドウから解除された場合、Silk Test は解除された Google Chrome ウィンドウの要素を認識しません。たとえば、Google Chrome を起動して、2 つのタブを開きます。次に、最初のタブから 2 番目のタブを解除します。Silk Test は 2 番目のタブの要素を認識しなくなっています。Silk Test を使用している場合に、複数の Google Chrome ウィンドウで要素を認識するには、**CTRL+N** を使用して新しい Google Chrome ウィンドウを開きます。
- Google Chrome を使用して Web アプリケーションをテストしている場合に、**Google Chrome を閉じた際にバックグラウンド アプリケーションの処理を続行する** チェックボックスがチェックされていると、Silk Test は Google Chrome を再起動してオートメーション サポートを読み込むことができません。

xBrowser のよくある質問

このセクションでは、Web アプリケーションをテストするときに発生することがある質問のコレクションを示します。

要素のテキストに使用されるフォント タイプの確認方法

属性名を「:」で区切ると、DOM 要素の currentStyle 属性のすべての属性にアクセスできます。

Internet Explorer 8 以前 wDomElement.GetProperty("currentStyle:fontName")

Internet Explorer 9 またはそれ以降および Mozilla Firefox などの他のすべてのブラウザ wDomElement.GetProperty("currentStyle:font-name")

textContent、innerText、および innerHtml の違い


- textContents は、書式設定のみを目的とする要素およびその子要素に含まれるすべてのテキストです。
- innerText は、要素およびその子要素に含まれるすべてのテキストを返します。
- innerHtml は、要素に含まれるすべてのテキスト (html タグも含む) を返します。

以下の html コードについて検討します。

```
<div id="mylinks">
  This is my <b>link collection</b>:
  <ul>
    <li><a href="www.borland.com">Bye bye <b>Borland</b> </a></li>
    <li><a href="www.microfocus.com">Welcome to <b>Micro Focus</b> </a></li>
  </ul>
</div>
```

以下の表に、返されるプロパティの詳細を示します。

コード	返される値
<code>browser.DomElement("//div[@id='mylinks']").GetProperty("textContents")</code>	This is my link collection:
<code>browser.DomElement("//div[@id='mylinks']").GetProperty("innerText")</code>	This is my link collection:Bye bye Borland Welcome to Micro Focus
<code>browser.DomElement("//div[@id='mylinks']").GetProperty("innerHTML")</code>	This is my link collection: Bye bye Borland Welcome to Micro Focus

 **注:** Silk Test 13.5 以降では、要素の textContents プロパティを通して取得されるテキスト内の空白類は、すべてのサポートするブラウザにおいて等しくトリムされます。一部のブラウザのバージョンでは、Silk Test 13.5 以前の Silk Test バージョンでは空白類の処理が異なります。OPT_COMPATIBILITY オプションを 13.5.0 より低いバージョンに設定することによって、以前の動作に戻すことができます。

innerText をカスタム クラス属性として構成したが、ロケーターで使用されない

ロケーター文字列に使用する属性には最大長があります。InnerText は長くなりすぎる傾向があり、ロケーターで使用できない場合があります。可能な場合は、textContents を代わりに使用してください。

クロスブラウザ スクリプトの作成時に必要な処置

クロスブラウザ スクリプトを作成する場合は、以下の 1 つまたは複数の問題に遭遇する場合があります。

- 属性値が異なる。たとえば、Internet Explorer の色が "# FF0000" として、Mozilla Firefox の色が "rgb(255,0,0)" として返されます。
- 属性名が異なる。たとえば、Internet Explorer 8 以前のバージョンではフォント サイズ属性が "fontSize" と呼ばれ、Internet Explorer 9 以降および Mozilla Firefox などの他のすべてのブラウザでは "font-size" と呼ばれます。
- 一部のフレームワークで異なる DOM ツリーがレンダリングされることがある

現在使用しているブラウザを確かめるには

BrowserApplication クラスには、ブラウザの種類を返すプロパティ "browsertype" があります。このプロパティをロケーターに追加することで、どのブラウザに一致させるかを定義できます。

新機能、サポート対象のプラットフォームとバージョン、既知の問題、および回避策の詳細については、『Silk Test リリース ノート』(<http://supportline.microfocus.com/productdoc.aspx> で入手可能) を参照してください。

使用例

ブラウザの種類を取得するには、次のコードをロケーターに入力します。

```
browserApplication.GetProperty("browsertype")
```

また、BrowserWindow には、現在のウィンドウのユーザー エージェント文字列を返すメソッド GetUserAgent があります。

安定したクロスブラウザ テストを実現するために最適なロケーター

組み込みロケーター生成プログラムでは、安定したロケーターの作成が試みられます。ただし、情報を使用できない場合、高品質のロケーターを生成することは困難です。この場合、ロケーター生成プログラムでは、階層形式の情報およびインデックスが使用されます。その結果、直接的な記録／再生には適していても、安定した日常的な実行には適さない脆弱なロケーターが生成されます。さらに、クロスブラウザ テストでは、いくつかの AJAX フレームワークで異なるブラウザに対して異なる DOM 階層がレンダリングされることがあります。

この問題を回避するには、アプリケーションの UI 要素にカスタム ID を使用します。

アプリケーションのログ出力に正しくないタイムスタンプが含まれる

この方法によって、同期に関して予期しない結果が発生する場合があります。この問題を回避するには、HTML 同期モードを指定します。

新しいページに移動したあと、スクリプトがハングする

この問題は、AJAX アプリケーションによりブラウザがビジー（サーバー プッシュ／ActiveX コンポーネントの接続が開いている）のままになっている場合に、発生することがあります。HTML 同期モードを設定してください。他のトラブルシューティングのヒントについては、「xBrowser のページ同期」のトピックを参照してください。

正しくないロケーターが記録されている

マウスを要素上に移動したときに、要素の属性が変更することがあります。Silk4NET によってこのシナリオの追跡が試行されますが、失敗することがあります。影響を受ける属性を特定し、それが Silk4NET で無視されるように構成してください。

Internet Explorer で要素を囲む四角形の位置が正しくない

- 拡大率が 100% に設定されていることを確認します。このようにしないと、四角形が正しく配置されません。
- ブラウザ ウィンドウの上に通知バーが表示されていないことを確認します。Silk4NET では、通知バーを処理できません。

Link.Select で、Internet Explorer で新しく開いたウィンドウにフォーカスが設定されない

この制限は、ブラウザの構成設定を変更することで修正できます。新しく開いたウィンドウが常にアクティブ化されるようにオプションを設定します。

DomClick(x, y) が Click(x, y) のように動作しない

アプリケーションで onclick イベントを使用しており、座標を必要とする場合、DomClick メソッドは動作しません。代わりに、Click を使用します。

FileInputField.DomClick() でダイアログが開かない

代わりに、Click を使用します。

マウス移動設定がオンになっているにもかかわらず、すべての操作が記録されない理由

多くの無用な MoveMouse 操作がスクリプトに影響を及ぼさないように、Silk4NET では以下の操作が行われます。

- マウスが一定時間静止している場合にのみ、MoveMouse 操作が記録されます。
- マウスを要素上に移動したあとで操作が行われていることが確認された場合にのみ、MoveMouse 操作が記録されます。場合によっては、スクリプトに手動操作を追加することが必要となることがあります。
- Silk4NET は、Web アプリケーション、Win32 アプリケーション、および Windows Forms アプリケーションに対してのみ、マウス移動の記録をサポートします。Silk4NET は、Adobe Flex や Swing など、xBrowser テクノロジ ドメインの子テクノロジ ドメインのマウス移動を記録することはできません。

xBrowser API で公開されていない機能が必要な場合の対処方法

ExecuteJavaScript() を使用して、JavaScript コードを Web アプリケーションから直接実行できます。この方法は、ほとんどすべての問題の回避策となります。

ロケーターでクラスとスタイルの属性が使用されない理由

これらの属性は AJAX アプリケーションで頻繁に変更され、ロケーターの安定性が損なわれることがあり、無視リストに含まれています。ただし、多くの場合、これらの属性を使用してオブジェクトを識別できるため、アプリケーションで使用することに意味がある場合があります。

再生中にダイアログが認識されない

スクリプトを記録するときに、Silk4NET はいくつかのウィンドウを Dialog として認識します。スクリプトをクロス ブラウザ スクリプトとして使用する場合は、ブラウザによっては Dialog が認識されないため、Dialog を Window に置き換える必要があります。

たとえば、スクリプトに以下の行があるとします。

```
/BrowserApplication//Dialog//PushButton[@caption='OK']
```

クロス ブラウザ テストを可能にするには、次のように行を書き換えます。

```
/BrowserApplication//Window//PushButton[@caption='OK']
```

ハンドル無効エラーが表示される理由

このトピックでは、Web アプリケーションをテストしたときに、Silk4NET に「このオブジェクトのハンドルは無効になりました。」というエラー メッセージが表示された場合の対処法について説明します。

このメッセージは、たとえば WaitForProperty などのメソッドを呼び出したオブジェクトが何らかの理由で消失していることを示しています。たとえば、Web アプリケーションでメソッドを呼び出しているときに、何らかの理由でブラウザが新しいページに移動した場合、以前のページのすべてのオブジェクトは自動的に無効になります。

この問題の原因が、組み込みの同期機能である場合もあります。たとえば、テスト対象のアプリケーションにショッピング カートが含まれていて、このショッピング カートに品物を追加したとします。ユーザーは次のページが読み込まれ、ショッピング カートのステータスが品物がある状態に変わるまで待機しています。品物を追加するという操作からの戻り時間が短すぎた場合、最初のページのショッピング カートはステータスが変わるまで待機しますが、その間も新しいページは読み込まれています。したがって、最初のページのショッピング カートは無効になります。この動作によって、ハンドル無効エラーが発生します。

この問題を回避するには、2 番目のページでのみ有効なオブジェクトが表示されるまで待機してから、ショッピング カートのステータスを確認するようにしてください。このオブジェクトが有効になるとすぐに、ショッピング カートのステータスを確認できるようになり、2 番目のページで正しく検証されるようになります。

Internet Explorer 10 で Click の記録が異なる理由

Internet Explorer 10 の DomElement で Click を記録し、DomElement が Click の後で破棄された場合、記録動作が予期したとおりにならないことがあります。別の DomElement が最初の DomElement の下にある場合、Silk Test では、1 つの Click が記録されるのではなく、Click、MouseMove、および ReleaseMouse が記録されます。

予期しない記録動作を回避する方法は、テスト対象のアプリケーションによって異なります。通常は、記録されたスクリプトから不必要な MouseMove イベントと ReleaseMouse イベントを削除すれば十分です。

Web アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Web アプリケーションがサポートする属性は次のとおりです。

- caption (次のワイルドカードをサポート：? および *)
- すべての DOM 属性 (次のワイルドカードをサポート：? および *)



注: 各ブラウザによって、空のスペースの処理に違いがあります。この結果、「textContent」および「innerText」属性は正規化されています。空のスペースのあとに別の空のスペースが続く場合、空のスペースはスキップされるか、または 1 文字の空白で置き換えられます。空のスペースとは、検出されたスペース、キャリッジリターン、改行、タブのことです。また、このような値に一致するものも正規化されます。例：

```
<a>abc  
abc</a>
```

以下のロケーターを使用します。

```
//A[@innerText='abc abc']
```

xBrowser クラス リファレンス

xBrowser アプリケーションを設定すると、Silk4NET は標準の xBrowser コントロールのテストのサポートを組み込みで提供します。

64 ビット アプリケーションのサポート

Silk4NET では、以下のテクノロジーについて、64 ビット アプリケーションのテストがサポートされています。

- Windows Forms
- Windows Presentation Foundation (WPF)
- Microsoft Windows API ベース
- Java AWT/Swing
- Java SWT

サポートするバージョン、既知の問題、および回避策についての最新の情報は、リリース ノートを確認してください。

サポートする属性の種類

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。必要に応じて、以下のいずれかの方法を使用して属性の種類を変更できます。

- 他の属性の種類と値を手動で入力する。
- **推奨属性リスト** の値を変更して、デフォルトの属性の種類に対して別の設定を指定する。

Adobe Flex アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Flex アプリケーションがサポートする属性は次のとおりです。

- automationName
- caption (automationName と同様)
- automationClassName (FlexButton など)
- className (実装クラスの完全修飾名。例 : mx.controls.Button)
- automationIndex (FlexAutomation のビューでのコントロールのインデックス。例 : index:1)
- index (automationIndex と同様。ただし、接頭辞はなし。例 : 1)
- id (コントロールの ID)
- windowId (id と同様)
- label (コントロールのラベル)
- すべての動的ロケーター属性



注: 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。


動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

Java AWT/Swing アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Java AWT/Swing でサポートされる属性には以下のものがあります。

- caption
- priorlabel : 隣接するラベル フィールドのテキストによってテキスト入力フィールドを識別します。通常、フォームのすべての入力フィールドに、入力の目的を説明するラベルがあります。caption のないコントロールの場合、自動的に属性 **priorlabel** がロケーターに使用されます。コントロールの **priorlabel** 値 (テキスト入力フィールドなど) には、コントロールの左側または上にある最も近いラベルの caption が使用されます。
- name
- accessibleName
- *Swing* のみ : すべてのカスタム オブジェクトの定義属性は、ウィジェットに `SetClientProperty("propertyName", "propertyValue")` で設定されます。


 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード? および * をサポートしています。

Java SWT アプリケーションの属性

ロケータが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケータがテスト内のオブジェクトを識別する方法が決定されます。

Java SWT がサポートする属性は次のとおりです。

- `caption`
- すべてのカスタム オブジェクト定義属性


 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード? および * をサポートしています。

SAP アプリケーションの属性

ロケータが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケータがテスト内のオブジェクトを識別する方法が決定されます。

SAP がサポートする属性は次のとおりです。


- `automationId`
- `caption`

 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード? および * をサポートしています。

Silverlight コントロールを識別するためのロケータ属性

Silverlight コントロールでサポートされているロケータ属性は次のとおりです。

- `automationId`
- `caption`
- `className`
- `name`
- すべての動的ロケータ属性


 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード? および * をサポートしています。

動的ロケータ属性の詳細については、「動的ロケータ属性」を参照してください。

Silverlight スクリプト内のコンポーネントを識別するために、`automationId`、`caption`、`className`、`name`、または任意の動的ロケータ属性を指定できます。`automationId` はアプリケーション開発者が設定します。たとえば、`automationId` を持つロケータは、以下のようになります：`// SLButton[@automationId="okButton"]`

`automationId` は一般に非常に有用で安定した属性であるため、使用することを推奨します。

属性の種類	説明	例
automationId	テスト対象アプリケーションの開発者によって設定される識別子。Visual Studio デザイナは、デザイナー上で作成されたすべてのコントロールに自動的に <i>automationId</i> を割り当てます。アプリケーション開発者は、アプリケーションのコード上でコントロールを識別するために、この ID を使用します。	// SLButton[@automationId="okButton"]
caption	コントロールが表示するテキスト。複数の言語にローカライズされたアプリケーションをテストする場合、 <i>caption</i> の代わりに <i>automationId</i> や <i>name</i> 属性を使用することを推奨します。	//SLButton[@caption="Ok"]
className	Silverlight コントロールの .NET 単純クラス名 (名前空間なし)。 <i>className</i> 属性を使用すると、Silk4NET が解決する標準 Silverlight コントロールから派生したカスタム コントロールを識別するのに役立ちます。	// SLButton[@className='MyCustomButton']
name	コントロールの名前。テスト対象アプリケーションの開発者によって設定されます。	//SLButton[@name="okButton"]

 **注目:** XAML コードの *name* 属性は、ローケータ属性 *name* ではなく、ローケータ属性 *automationId* にマップされます。

Silk4NET は、*automationId*、*name*、*caption*、*className* 属性をこの表に示した順番に使用して Silverlight コントロールのローケータを記録時に作成します。たとえば、コントロールが *automationId* と *name* を持つ場合、*automationId* が固有の場合は Silk4NET がローケータを作成する際に使用されません。

以下の表は、アプリケーション開発者がテキスト「Ok」を持つ Silverlight ボタンをアプリケーションの XAML コードに定義する方法を示しています。

オブジェクトの XAML コード	Silk Test からオブジェクトを検索するためのローケータ
<Button>Ok</Button>	//SLButton[@caption="Ok"]
<Button Name="okButton">Ok</Button>	//SLButton[@automationId="okButton"]
<Button AutomationProperties.AutomationId="okButton">Ok</Button>	//SLButton[@automationId="okButton"]
<Button AutomationProperties.Name="okButton">Ok</Button>	//SLButton[@name="okButton"]

Rumba コントロールを識別するためのローケータ属性


ローケータが作成されるとき、属性の種類はアプリケーションが使用するテクノロジドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ローケータがテスト内のオブジェクトを識別する方法が決定されます。サポートされている属性は次のとおりです。

- caption** コントロールが表示するテキスト。
- priorlabel** フォームの入力フィールドには通常入力の目的を説明するラベルがあるため、**priorlabel** の目的は隣接するラベル フィールド **RumbaLabel** のテキストによってテキスト入力フィールド **RumbaTextField** を識別することです。テキスト フィールドの同じ行の直前にラベルがない場合、または右側のラベルが左側のラベルよりテキスト フィールドに近い場合、テキスト フィールドの右側にあるラベルが使用されます。

StartRow この属性は記録されていませんが、手動でロケータに追加することができます。**StartRow** を使用して、この行で始まるテキスト入力フィールド、**RumbaTextField** を識別します。

StartColumn この属性は記録されていませんが、手動でロケータに追加することができます。**StartColumn** を使用して、この列で始まるテキスト入力フィールド、**RumbaTextField** を識別します。

すべての動的ロケータ属性。 動的ロケータ属性の詳細については、「動的ロケータ属性」を参照してください。


 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード ? および * をサポートしています。

Web アプリケーションの属性

ロケータが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケータがテスト内のオブジェクトを識別する方法が決定されます。

Web アプリケーションがサポートする属性は次のとおりです。

- caption (次のワイルドカードをサポート : ? および *)
- すべての DOM 属性 (次のワイルドカードをサポート : ? および *)

 **注:** 各ブラウザによって、空のスペースの処理に違いがあります。この結果、「textContent」および「innerText」属性は正規化されています。空のスペースのあとに別の空のスペースが続く場合、空のスペースはスキップされるか、または 1 文字の空白で置き換えられます。空のスペースとは、検出されたスペース、キャリッジ リターン、改行、タブのことです。また、このような値に一致するものも正規化されます。例 :

```
<a>abc  
abc</a>
```

以下のロケータを使用します。


```
//A[@innerText='abc abc']
```

Windows Forms アプリケーションの属性

ロケータが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケータがテスト内のオブジェクトを識別する方法が決定されます。

Windows Forms アプリケーションがサポートする属性は次のとおりです。

- automationid
- caption
- windowid
- priorlabel (caption のないコントロールの場合、自動的に priorlabel が caption として使用されます。caption のあるコントロールの場合、caption を使う方が簡単な場合があります。)

 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード ? および * をサポートしています。

Windows Presentation Foundation (WPF) アプリケーションの属性

WPF アプリケーションがサポートする属性は次のとおりです。

- *automationId*
- *caption*
- *className*
- *name*
- すべての動的ロケータ属性。



注: 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード ? および * をサポートしています。

動的ロケータ属性の詳細については、「動的ロケータ属性」を参照してください。

オブジェクト解決

WPF スクリプト内のコンポーネントを識別するために、*automationId*、*caption*、*className*、あるいは *name* を指定できます。アプリケーション中の要素に指定された *name* が利用可能な場合、ロケータの *automationId* 属性として使用されます。この結果、多くのオブジェクトは、この属性のみを使用して一意に識別できます。たとえば、*automationId* を持つロケータは、以下のようになります：
`// WPFButton[@automationId='okButton']"`

automationId や他の属性を定義した場合、再生中に *automationId* だけが使用されます。*automationId* が定義されていない場合には、コンポーネントを解決するのに *name* が使用されます。*name* も *automationId* もどちらも定義されていない場合には、*caption* 値が使用されます。*caption* が定義されていない場合は、*className* が使用されます。*automationId* は非常に役立つプロパティであるため、使用することを推奨します。

属性の種類	説明	例
<i>automationId</i>	テスト アプリケーションの開発者によって提供された ID	<code>//WPFButton[@automationId='okButton']"</code>
<i>name</i>	コントロールの名前。Visual Studio デザイナは、デザイナー上で作成されたすべてのコントロールに自動的に名前を割り当てます。アプリケーション開発者は、アプリケーションのコード上でコントロールを識別するために、この名前を使用します。	<code>//WPFButton[@name='okButton']"</code>
<i>caption</i>	コントロールが表示するテキスト。複数の言語にローカライズされたアプリケーションをテストする場合、 <i>caption</i> の代わりに <i>automationId</i> や <i>name</i> 属性を使用することを推奨します。	<code>//WPFButton[@automationId='Ok']"</code>

属性の種類	説明	例
className	WPF の .NET 単純クラス名 (名前空間なし)。クラス名属性を使用すると、Silk4NET が解決する標準 WPF コントロールから派生したカスタム コントロールを識別するのに役立ちます。	//WPFButton[@className='MyCustomButton']"

Silk4NET は、*automationId*、*name*、*caption*、*className* 属性をこの表に示した順番に使用して WPF コントロールのロケータを記録時に作成します。たとえば、コントロールが *automationId* と *name* を持つ場合、Silk4NET がロケータを作成する際には *automationId* が使用されます。

以下の例では、アプリケーション開発者がアプリケーションの WPF ボタンに対して *name* と *automationId* を XAML コードに定義する方法を示します。


```
<Button Name="okButton" AutomationProperties.AutomationId="okButton"
Click="okButton_Click">Ok</Button>
```

Windows API ベースのクライアント/サーバー アプリケーションの属性

ロケータが作成されるとき、属性の種類はアプリケーションが使用するテクノロジー ドメインに基づいて自動的に割り当てられます。属性の種類と値によって、ロケータがテスト内のオブジェクトを識別する方法が決定されます。


Windows API ベースのクライアント/サーバー アプリケーションがサポートする属性は次のとおりです。

- *caption*
- *windowid*
- *priorlabel* : 隣接するラベル フィールドのテキストによってテキスト入力フィールドを識別します。通常、フォームのすべての入力フィールドに、入力の目的を説明するラベルがあります。 *caption* のないコントロールの場合、自動的に属性 ***priorlabel*** がロケータに使用されます。コントロールの ***priorlabel*** 値 (テキスト ボックスなど) には、コントロールの左側または上にある最も近いラベルの *caption* が使用されます。

 **注:** 属性の名前は、大文字小文字が区別されます。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケータ属性は、ワイルドカード ? および * をサポートしています。

動的ロケータ属性

再生中にコントロールを識別するために、事前に定義されたロケータ属性のセット (*caption* や *automationId* など。テクノロジー ドメインに依存します) をロケータに使用できます。しかし、動的プロパティを含む、コントロールのすべての属性をロケータ属性として使用することもできます。特定のコントロールで使用可能なプロパティのリストを取得するには、*GetPropertyList* メソッドを使用します。返されたプロパティはすべて、ロケータを使用してコントロールを識別するのに使用できます。

 **注:** 特定のプロパティの実際の値を取得するには、*GetProperty* メソッドを使用します。この値はロケータで使用できます。

例

Silverlight アプリケーションのダイアログ ボックスにあるボタンを識別する場合、以下のように入力します。

```
browser.Find("//SLButton[@IsKeyboardFocused=true]")
```

または

```
Dim button = dialog.SLButton("@IsKeyboardFocused=true")
```

これが機能するのは、Silk4NET により Silverlight ボタン コントロールの IsDefault というプロパティが公開されるためです。

例

Silverlight アプリケーションのフォント サイズ 12 のボタンを識別する場合、以下のように入力します。

```
Dim button = browser.Find("//SLButton[@FontSize=12]")
```

または

```
Dim button = browser.SLButton("@FontSize=12")
```

これが機能するのは、テスト対象アプリケーションの基になるコントロール (この場合、Silverlight ボタン) が FontSize というプロパティを持つためです。

索引

記号

- .NET のサポート
 - Silverlight 113
 - Windows Forms の概要 102
 - Windows Presentation Foundation(WPF)の概要 106
 - 概要 102

数字

- 64 ビット アプリケーション
 - サポート 136

A

- Adobe Flex
 - Adobe AIR のサポート 82
 - automationIndex プロパティ 89
 - automationName プロパティ 89, 90
 - Component Explorer 72
 - Flash Player 設定 71
 - FlexDataGrid コントロール 83
 - Select メソッド 83, 93
 - アプリケーションの作成 88
 - アプリケーションの事前コンパイル 85
 - アプリケーションの有効化 84
 - オートメーション パッケージのリンク 84
 - 概要 71
 - カスタム コントロール 66, 72, 81, 90
 - カスタム コントロールの実装 78, 81, 90
 - カスタム コントロールの定義 74
 - カスタム コントロールのメソッドの呼び出し 77
 - クラス定義ファイル 81, 90
 - 構成情報の追加 87
 - コンテナ 93
 - コンテナのコーディング 93
 - 実行時のパラメーター渡し 88
 - 実行時の読み込み 86, 87
 - 実行前のパラメーター渡し 87
 - スクリプトのカスタマイズ 82
 - スタイル 96
 - セキュリティ設定 97
 - 属性 97, 137
 - テスト 72
 - テストの記録 95
 - テストの再生 96
 - テストの初期化 95
 - パラメータを渡す 87
 - 複数のアプリケーションのテスト 82
 - 複数ビュー コンテナ 94
 - メソッドの呼び出し 73
 - ワークフロー 95
- AJAX アプリケーション
 - スクリプトのハング 134
 - ブラウザ設定 126
- API 再生
 - ネイティブ再生との比較 125

C

- Chrome
 - 構成設定 127
 - 前提条件 130

- Chrome
 - クロスブラウザ スクリプト 133
- Component Explorer
 - Adobe Flex 72
- Customer Care 11

D

- dll
 - 関数の宣言構文 57
 - 関数への引数の受け渡し 58
 - 関数への文字列引数の受け渡し 59
 - 規則の変更 59
 - 名前のエイリアス設定 59
- DOM 関数 125
- dynamicInvoke
 - Adobe Flex 73
 - Java AWT 98, 101
 - Java Swing 98, 101
 - SAP 120
 - Silverlight 115
 - Windows Forms 103
 - Windows Presentation Foundation (WPF) 109

E

- Eclipse
 - 概要 100
- Eclipse RCP
 - サポート 100

F

- FAQ
 - xBrowser 132
- Firefox
 - 構成設定 127
 - スクリプトの再生 130
 - ロケーター 134
- Firefox
 - クロスブラウザ スクリプト 133
- Flash Player
 - アプリケーションを開く 71
 - セキュリティ設定 97
- Flex
 - Adobe AIR のサポート 82
 - automationIndex プロパティ 89
 - automationName プロパティ 89, 90
 - Component Explorer 72
 - Flash Player 設定 71
 - FlexDataGrid コントロール 83
 - Select メソッド 83, 93
 - アプリケーションの作成 88
 - アプリケーションの事前コンパイル 85
 - アプリケーションの有効化 84
 - オートメーション パッケージのリンク 84

概要 71

カスタム コントロール

実装 78

定義 74, 81, 90

カスタム コントロールの実装 78, 81, 90

カスタム コントロールの定義 74

カスタム コントロールのメソッドの呼び出し 77

クラス定義ファイル 81, 90

構成情報の追加 87

コンテナ 93

実行時のパラメーター渡し 88

実行時の読み込み 86, 87

実行前のパラメーター渡し 87

スクリプトのカスタマイズ 82

スタイル 96

セキュリティ設定 97

属性 97, 137

テスト 72

テストの記録 95

テストの再生 96

パラメータを渡す 87

複数のアプリケーションのテスト 82

複数ビュー コンテナ 94

メソッドの呼び出し 73

ワークフロー 95

G

Google Chrome

構成設定 127

スクリプトの再生 130

制限事項 131

前提条件 130

I

innerHTML 132

innerText 132, 133

Internet Explorer

位置が正しくない四角形 134

クロスブラウザ スクリプト 133

Internet Explorer

link.select のフォーカスの問題 134

構成設定 127

ロケーター 134

Internet Explorer 10

予期しない Click 動作 136

invoke

SAP 120

Windows Forms 103

Windows Presentation Foundation (WPF) 109

InvokeMethods

Adobe Flex 73

Java AWT 98, 101

Java Swing 98, 101

SAP 120

Silverlight 115

Windows Forms 103

Windows Presentation Foundation (WPF) 109

invoke メソッド

呼び出し可能なメソッド 63

J

Java AWT

概要 97

属性 98, 137

属性の種類 98, 137

メソッドの呼び出し 98, 101

Java Swing

概要 97

属性 98, 137

メソッドの呼び出し 98, 101

Java SWT

ウィジェット 100

概要 100

カスタム属性 29

カスタム属性の記録 100

サポート 100

属性の種類 101, 138

メソッドの呼び出し 98, 101

Java SWT クラス リファレンス 100

Java AWT/Swing

priorlabel 99

L

LoadAssembly

アセンブリをコピーできない 66

Locator Spy

オブジェクト マップ項目をテスト メソッドに追加する 21

概要 40

ロケーターをテスト メソッドに追加する 21

M

Microsoft ユーザー補助

オブジェクト解決の向上 60

Mozilla Firefox

構成設定 127

O

OPT_ALTERNATE_RECORD_BREAK

オプション 27

OPT_ENABLE_ACCESSIBILITY

オプション 32

OPT_ENSURE_ACTIVE_OBJDEF

オプション 31

OPT_LOCATOR_ATTRIBUTES_CASE_SENSITIVE

オプション 32

OPT_RECORD_MOUSEMOVE_DELAY

オプション 27

OPT_RECORD_MOUSEMOVES

オプション 27

OPT_RECORD_SCROLLBAR_ABSOLUT

オプション 27

OPT_REMOVE_FOCUS_ON_CAPTURE_TEXT

オプション 32

OPT_REPLAY_MODE

オプション 31

OPT_WAIT_RESOLVE_OBJDEF 30

OPT_WAIT_RESOLVE_OBJDEF_RETRY 30

OPT_XBROWSER_RECORD_LOWLEVEL 28
OPT_XBROWSER_SYNC_EXCLUDE_URLS 30
OPT_XBROWSER_SYNC_MODE 30
OPT_XBROWSER_SYNC_TIMEOUT 30

P

priorlabel
Java AWT/Swing テクノロジ ドメイン 99
priorLabel
Win32 テクノロジ ドメイン 122

R

Rumba
class reference 118
画面検証の使用 119
サポートの有効化と無効化 118
ロケータ属性 118, 139
Rumba ロケータ属性
コントロールの識別 118, 139

S

SAP
概要 119
カスタム属性 29
クラス リファレンス 120
セキュリティ設定 121
属性の種類 120, 138
メソッドの呼び出し 120
SAP コントロール
メソッドを動的に呼び出す 121
SetText 28
Silk4NET
基本的なワークフロー 12
詳細 8
テスト 18
テストの手動作成 20
プロジェクト 16
Silk4NET テスト
記録 19
実行 21
手動作成 20
Silk4NET テストの追加
Silk4NET プロジェクト 18
Silk4NET プロジェクト
テストの追加 18
Silk4NET プロジェクトの作成
Visual Studio 16
Silverlight
概要 113
クラス リファレンス 113
サポート 113
スクロール 116
属性の種類 113, 138
トラブルシューティング 117
メソッドの呼び出し 115
ロケータ属性 113, 138
SupportLine 11
Swing
概要 97
属性 98, 137
メソッドの呼び出し 98, 101

T

Team Foundation Server
Silk4NET テストで使用する 25
TrueLog の場所 25
textContents 132
TFS
Silk4NET テストで使用する 25
TrueLog
Team Foundation Server 25
構成 27
非 ASCII 文字の置換 24
ビジュアル実行ログの作成 23
不正な非 ASCII 文字 24
有効化 23, 27
TrueLog Explorer
TrueLog の有効化 23
構成 27
ビジュアル実行ログの作成 23
有効化 27
TrueLog の有効化
TrueLog Explorer 23
TypeKeys 28

W

Web アプリケーションの
xBrowser テスト オブジェクト 123
WebSync 11
Web アプリケーション
カスタム属性 29, 126
サポートされている属性 136, 140
ハンドル無効エラー 135
Win32
priorLabel 122
Win32 クラス リファレンス 122
Windows
64 ビット アプリケーションのサポート 136
属性の種類 107, 141
Windows API
サポート 121
Windows API ベース
64 ビット アプリケーションのサポート 136
Windows API ベースのサポート
概要 121
Windows Forms
64 ビット アプリケーションのサポート 136
概要 102
カスタム属性 29
クラス リファレンス 103
属性の種類 103, 140
メソッドの呼び出し 103
Windows Presentation Foundation (WPF)
64 ビット アプリケーションのサポート 136
WPFItemsControl クラス 108
概要 106
カスタム コントロール 108
クラスの公開 112
クラス リファレンス 107
メソッドの呼び出し 109
ロケータ属性 107, 141

- Windows アプリケーション
 - カスタム属性 29
- Windows クラス リファレンス 122
- Works Order 番号 11
- WPF
 - 64 ビット アプリケーションのサポート 136
 - WPFIItemsControl クラス 108
 - カスタム コントロール 108
 - クラスの公開 30, 112
 - クラス リファレンス 107
 - メソッドの呼び出し 109
 - ロケーター属性 107, 141
- WPF アプリケーション
 - カスタム属性 29
- WPF クラスの公開 30
- WPF ロケーター属性
 - コントロールの識別 107, 141

X

- xBrowser
 - Internet Explorer で四角形の位置が正しくない 134
 - 機能の公開 135
 - クロスブラウザ スクリプト 133
 - 認識されないダイアログ 135
- xBrowser
 - API とネイティブ再生 125
 - DomClick が Click のように動作しない 134
 - FAQ 132
 - FieldInputField.DomClick でダイアログが開かない 134
 - innerText がロケーターで使用されない 133
 - link.select のフォーカスの問題 134
 - textContent、innerText、innerHTML 132
 - 新しいページへの移動 134
 - オブジェクト解決 123
 - オブジェクト マップ 43
 - 概要 123
 - カスタム属性 29, 126
 - 記録オプション 126
 - クラス リファレンス 136
 - 再生オプション 125
 - 属性の種類 136, 140
 - タイムスタンプ 134
 - 正しくないロケーターの記録 134
 - テストオブジェクト 123
 - フォント タイプの検証 132
 - ブラウザ構成設定 127
 - ブラウザの種類の違い 133
 - ページ同期 124
 - マウス移動の記録 134
 - マウス移動の詳細設定 127
 - ロケーター生成プログラムを構成する 129
 - ロケーターにないクラスとスタイル 135
 - ロケーターの記録 134
- xBrowser のページ同期 124
- XPath
 - 概要 34
 - クエリ文字列の作成 40
 - トラブルシューティング 39
- XPath のトラブルシューティング 39

あ

- アプリケーション構成
 - エラー 15
 - 削除 14
 - 追加 14
 - 定義 14
 - トラブルシューティング 15
 - 変更 14

い

- イメージ解決
 - 概要 51
 - メソッド 51
 - 有効化 51
- イメージ クリック
 - 記録 51
- イメージ クリックの記録
 - 概要 51
- イメージ検証
 - 概要 53
 - 記録中に追加する 54
 - 作成 54
 - 他のプロジェクトでの使用 42, 55
- イメージ資産
 - 概要 52
 - 作成 52
 - 他のプロジェクトでの使用 42, 55
 - 複数のイメージを追加する
 - 複数のイメージを追加する
 - イメージ資産 53
- イメージのチェック
 - 概要 53

お

- オブジェクト
 - 存在の確認 38
- オブジェクト タイプ
 - ロケーター 34
- オブジェクト解決
 - Exists メソッド 38
 - FindAll メソッド 38
 - Find メソッドの使用 38
 - 概要 34
 - 属性の使用 35
 - 複数オブジェクトの識別 38
 - ユーザー補助を使用して向上する 60
- オブジェクト解決の向上
 - ユーザー補助 60
- オブジェクト マップ
 - Web アプリケーション 43
 - xBrowser 43
 - オフに切り替え 41
 - オンに切り替え 41
 - 概要 41
 - 記録 41
 - 項目のコピー 46
 - 項目の削除 49
 - 項目の追加 47

- 項目の名前変更 43
- スクリプトから開く 47
- スクリプトでのロケータからオブジェクト マップへの移動 48
- 他のプロジェクトでの使用 42, 55
- ベストプラクティス 50
- 利点 41
- オブジェクト マップ項目
 - エラーの検出 49
 - コピー 46
 - 削除 49
 - 識別 44, 48
 - 追加 47
 - テスト アプリケーションからの更新 45
 - テスト アプリケーションでの検索 48
 - 名前の変更 43
 - ハイライト 48
 - ロケータの変更 44
- オブジェクトを解決する
 - xBrowser 123
- オプション
 - 詳細設定 32
 - ブラウザの記録オプションの設定 126
- オプションの指定
 - スクリプト 27

か

- カスタマー ケア 11
- カスタム コントロール
 - Adobe Flex の動的呼び出し 77
 - AUT にコードを追加する 64
 - AUT へのコードの追加にかんする FAQ 66
 - Flex のテスト 66, 72
 - 概要 62
 - カスタム クラスの作成 68
 - 管理 67
 - サポート 68
 - 挿入したコードが AUT で使用されない 66
 - ダイアログ ボックス 69
 - 定義 81, 90
 - 動的呼び出しに関する FAQ 63
 - 呼び出しで予期しない文字列が返される 63
- カスタム コントロールのテスト
 - AUT にコードを追加する 64
- カスタム属性
 - 設定 29, 126

き

- 基本状態
 - 定義 13
 - 変更 13
- 記録
 - Silk4NET テスト 19
 - イメージ検証を追加する 54
 - オブジェクト マップ 41
 - 詳細設定 27
- 記録停止キー 27

く

- クラス
 - 公開 30
 - 無視 30

- クラスの無視 30
- クラス名
 - Locator Spy で探す 21

け

- 検索範囲
 - ロケータ 34

こ

- 更新 8
- コントロールの識別
 - Locator Spy 40
 - 動的ロケータ属性 142

さ

- 再生
 - オプション 31
- 再生
 - 認識されないダイアログ 135
- サンプル スクリプト
 - 使用法 33
 - 場所 33

し

- 資産
 - スクリプトから開く 53
- 準備
 - Silk4NET 12
- 詳細設定
 - オプション 32
- ショートカット キーの組み合わせ 27
- 除外される文字
 - 記録 20
 - 再生 20
- シリアル番号 11

す

- スクリプト
 - オブジェクト マッピング 41
 - オプションの指定 27
 - ロケータからオブジェクト マップへの移動 48
- スクリプトの再生
 - Firefox 130
 - Google Chrome 130
- スクロール イベント 27
- スタイル
 - Flex アプリケーション 96

せ

- 製品サポート 11
- 製品の更新 8
- セキュリティ設定
 - SAP 121
- 前提条件

Google Chrome 130

そ

属性除外リスト

設定 126

属性値

Locator Spy で探す 21

属性の種類

Adobe Flex 97, 137

Java AWT 98, 137

Java Swing 98, 137

Java SWT 101, 138

SAP 120, 138

Silverlight 113, 138

Web アプリケーション 136, 140

Windows 107, 141

Windows Forms 103, 140

xBrowser 136, 140

概要 137

た

ダイアログ

認識しない 135

タイムスタンプ 134

ダウンロード 11

て

テキスト解決

概要 61

テキスト クリックの記録

概要 61

テスト

Silk4NET 18

拡張 57

結果の分析 22

テストの実行

Silk4NET 21

テスト メソッド

オブジェクト マップ項目を追加する 21

ロケーターを追加する 21

と

同期オプション 30

動的呼び出し

AUT にコードを追加する際の FAQ 66

FAQ 63

概要 63

スクリプトの単純化 63

入力引数の型が一致しない 66

予期しない戻り値 63

動的ロケーター属性

詳細 142

トラブルシューティング

Silverlight 117

に

入力引数の型が一致しない

動的呼び出し 66

ね

ネイティブ再生

API 再生との比較 125

ネイティブなユーザー入力

概要 125

記録 126

は

ハンドル無効エラー

トラブルシューティング 135

ひ

ビジュアル実行ログの作成

TrueLog 23

TrueLog Explorer 23

ふ

ブラウザ

詳細設定の設定 28

ブラウザ構成設定

xBrowser 127

ブラウザの記録オプション 126

ブラウザの記録オプションの設定 126

ブラウザの種類

GetProperty 133

使用法 133

プロジェクト

Visual Studio の作成 16

プロジェクト

Silk4NET 16

Silk4NET テストの追加 18

プロジェクトの依存関係

追加 42, 55

分析

テスト結果 22

ま

マウス移動操作 27

マウス移動の詳細設定 127

め

メソッドの動的呼び出し

Adobe Flex 73

Adobe Flex カスタム コントロールのテスト 77

Java AWT 98, 101

Java Swing 98, 101

Java SWT 98, 101

SAP 120

Silverlight 115

Windows Forms 103

Windows Presentation Foundation (WPF) 109

メソッドを動的に呼び出す

SAP コントロール 121

ゆ

- ユーザー補助
 - オブジェクト解決の向上 60

よ

- 予期しない Click 動作
 - Internet Explorer 136
- よくある質問
 - AUT にコードを追加する 66
 - 動的呼び出し 63
- 呼び出し
 - Java AWT 98, 101
 - Java SWT 98, 101
 - Swing 98, 101

ら

- ライセンス
 - 利用可能なライセンスの種類 7

れ

- 連絡先情報 11

ろ

- ロケーター

- xBrowser 134
- xBrowser 記録オプションの設定 126
- xBrowser 内で不正 134
- オブジェクト タイプ 34
- オブジェクト マップでの変更 44
- オプションの設定 126
- 基本概念 34
- 検索範囲 34
- 構文 35
- サポートしない構成子 35
- サポートする構成子 35
- サポートするサブセット 37
- スクリプトでのオブジェクト マップ エントリへの移動 48
- 属性 28
- 属性の使用 35
- マッピング 41
- ロケーター生成プログラム
 - xBrowser 用に構成する 129
- ロケーター属性
 - Rumba コントロール 118, 139
 - WPF コントロール 107, 141
 - Silverlight コントロール 113, 138
 - 除外される文字 20
 - 動的 142

わ

- ワーク オーダー番号 11