

Ranorex

変更に強い RanoreXPath の設計方法

Last updated: 2026.08

目次

内容

Ranorex	1
変更に強い RanoreXPath の設計方法	1
目次	2
1 はじめに	4
1.1 本書について	4
2 RanoreXPath に影響を与える主な要因	5
2.1 ブラウザーのアップデート	5
2.2 Windows OS のアップデート	5
2.3 テスト対象アプリケーションのアップデート	5
2.4 動的な属性の使用	5
3. メンテナンス工数を抑える RanoreXPath 設計	6
3.1. メンテナンス工数を抑える RanoreXPath とは	6
3.1.1. RanoreXPath の構造	6
3.1.2. メンテナンス工数を抑える RanoreXPath の作り方	7
3.1.2.1. 変更されにくい属性を使用する	7
3.1.2.2. 属性値を正規表現で指定する	9
3.1.2.3. パス構造を簡略化する	10
3.2. メンテナンス工数を抑える方法	13
3.2.1. 速度と堅牢性の設定	13
3.2.1.1. 速度と堅牢性とは	13
3.2.1.2. 設定の効果（「堅牢性」重視型）	14
3.2.1.3. 有効なケース	14
3.2.2. 重み付けルールの設定	15
3.2.2.1. 重み付けルールとは	15
3.2.2.2. 設定の効果	15
3.2.2.3. 有効なケース	15
3.3. RanoreXPath 設計の進め方と推奨設定	16
3.3.1. 基本的な進め方	16
3.3.2. ケース別の推奨設定	17
4. RanoreXPath の設定手順	18
4.1. 速度と堅牢性の設定	18
4.1.1. 速度と堅牢性の設定	18

4.1.2.	ソリューション設定の共有	19
4.2.	重み付けルールの設定	19
4.2.1.	重み付けルールの設定.....	19
4.2.2.	重み付けルール of 共有.....	23

1 はじめに

1.1 本書について

ブラウザ/Window OS/テスト対象アプリケーションなどのアップデートや属性の特性などにより、テスト対象アプリケーションのオブジェクト情報が意図せず変化し、RanoreXPath の修正が必要となることで、メンテナンス工数が発生する場合があります。本書では、メンテナンス工数を抑えるための RanoreXPath の設計方法について説明します。

本書は、Ranorex の基本機能や操作を理解し、RanoreXPath に関する基礎知識をお持ちの方を対象としています。
(弊社で開催している無償ハンズオンセミナーの内容レベル)



メモ

無償ハンズオンセミナーの詳細については、以下のリンクをご確認ください。

- ハンズオンセミナー（初級）
<https://ranorex.techmatrix.jp/handson-details/>

2 RanoreXPath に影響を与える主な要因

2.1 ブラウザーのアップデート

ブラウザのアップデートにより、Web ページの内部構造（DOM 構造）やオブジェクト情報が変化することで、Ranorex が取得する UI ツリー構造や属性情報に影響を与える場合があります。

2.2 Windows OS のアップデート

Windows OS のアップデートにより、テスト対象アプリケーションの UI 情報を取得する仕組みが変化することで、Ranorex が取得する UI ツリー構造や属性情報に影響を与える場合があります。

2.3 テスト対象アプリケーションのアップデート

テスト対象アプリケーションのアップデート（改修など）により、画面構成やコントロール、属性情報が変更されることで、既存の RanoreXPath で対象のオブジェクトを認識できなくなる場合があります。

2.4 動的な属性の使用

RanoreXPath に動的な属性が使用されている場合は、アプリケーションの状態変化にともない属性値が変化することで、既存の RanoreXPath で対象のオブジェクトを認識できなくなる場合があります。

3. メンテナンス工数を抑える RanoreXPath 設計

3.1. メンテナンス工数を抑える RanoreXPath とは

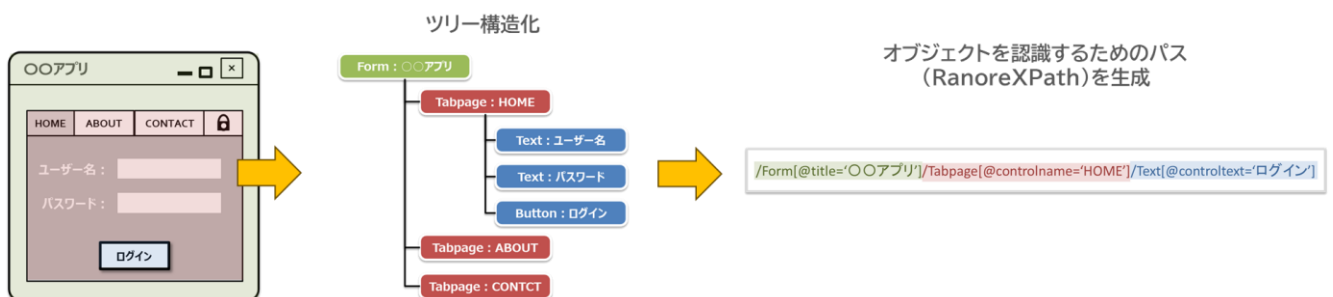
3.1.1. RanoreXPath の構造

Ranorex は、テスト対象アプリケーション内にあるボタンや入力フォームなどのオブジェクトを認識するために、RanoreXPath というパスを使用します。RanoreXPath には、アプリケーション内部で保持しているシステム情報が使用されます。

Web アプリケーションでは、DOM 情報が使用され、デスクトップアプリケーションやモバイルアプリケーション（ネイティブアプリケーション）では、アプリケーション内のコントロールやプロパティ情報が使用されます。

RanoreXPath で使用されるアプリケーションの内部情報は、ツリー構造で表示され、Ranorex Spy ツールを使用することで確認ができます。このツリー内の各階層を要素と呼び、複数の要素が組み合わさり、RanoreXPath が構成されます。

テスト対象アプリケーションの内部情報を基にオブジェクトを認識：



RanoreXPath の構造については、以下のユーザーガイドに記載されています。

- RanoreXPath の基本

<https://support.ranorex.com/hc/en-us/articles/38037966014225-RanoreXPath-Basics>

3.1.2. メンテナンス工数を抑える RanoreXPath の作り方

RanoreXPath のメンテナンス工数を抑えるためには、ブラウザー/Window OS/テスト対象アプリケーションなどのアップデートや属性の動的な変化などの、さまざまな変更要因の影響を受けにくいパス設計をおこなうことが重要であり、その対応として、以下のような方法があります。

- 変更されにくい属性を使用する
- 属性値を正規表現で指定する
- パス構造を簡略化する



メモ

既存の RanoreXPath を編集する場合は、パスエディター画面を使用することで対応可能です。
詳細は以下のユーザーガイドをご参照ください。

- Integrated Ranorex Spy
<https://support.ranorex.com/hc/en-us/articles/43052619754257-Integrated-Ranorex-Spy>
- パスエディターの基本
<https://support.ranorex.com/hc/en-us/articles/43499019175825-Path-Editor-Basics>

3.1.2.1. 変更されにくい属性を使用する

アプリケーションの特性上、ユーザーインターフェース（UI）の操作によって入力フォームやボタンなどの位置が変わることや、開発の過程で UI が変更される場合があります。こうした場合、リポジトリに登録されている既存のオブジェクトに使われている RanoreXPath（要素や属性の情報）が変わることで、オブジェクトが正しく認識されなくなり、結果として、メンテナンスが必要になります。

このメンテナンスにかかる工数を減らすには、RanoreXPath に使用する属性情報を、位置や UI の変更に影響されにくいものにすることが効果的です。そうすることで、RanoreXPath の耐久性を高め、安定した認識が可能になります。

1. 本設定が有効なケース

- 一部のテスト対象オブジェクトの RanoreXPath に動的な属性が使用されている場合
- 開発において、UI の変更が頻繁に発生する場合

2. 設定のポイント

例として、以下のリポジトリアイテムに登録されている RanoreXPath において、**text** 属性の値

(**img_w3bnz2p1t58q2xi0**) が、「一部のテスト対象オブジェクトの RanoreXPath に動的な属性が使用されている場合」のケースに当てはまる場合、**text** 属性を無効にし、動的でない他の属性 (**controlname**) を使用することで、RanoreXPath を堅牢にすることができます。

変更前の RanoreXPath :

```
/form[@controlname='RxMainFrame']/?/?/tabpage[@controlname='RxTabDynamic']  
/text[@text='img_w3bnz2p1t58q2xi0']
```

対象のリポジトリアイテムのパスエディター画面のツリーの要素から、対象の要素 (**text** 要素) を選択し、属性の編集画面にて、**text** 属性のチェックを無効にし **controlname** 属性などの他の属性を有効にします。



変更後の RanoreXPath :

```
/form[@controlname='RxMainFrame']/?/?/tabpage[@controlname='RxTabDynamic']  
/text[@controlname='txtBoxID']
```



注意

属性を変更した後の RanoreXPath は一意のオブジェクトを認識する必要があります。1 つの属性値では一意のオブジェクトにならない場合、複数の属性を有効にし、各属性を組み合わせて設定することも可能です。

3.1.2.2. 属性値を正規表現で指定する

自動生成された RanoreXPath の中に、特定の形式の動的な属性値が存在する場合は、その形式に合わせて属性値を正規表現で設定することが可能です。属性値が変わった場合も、設定した正規表現に一致することにより、対象の要素を認識することができます。

1. 本設定が有効なケース

- 属性値の一部が特定の形式で変化する場合
- 安定して認識できる代替の属性が存在しない場合

2. 設定のポイント

例として、以下のリポジトリアイテムに登録されている RanoreXPath において、**text** 属性の値

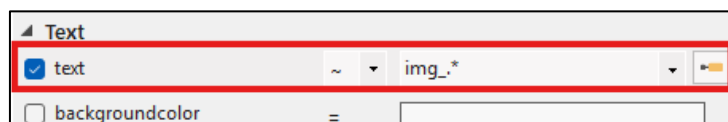
(**img_w3bnz2p1t58q2xi0**) が、「属性値の一部が特定の形式で変化する場合」のケースに当てはまる場合、その形式に合わせた正規表現で設定することで、RanoreXPath を堅牢にすることができます。

※**text** 属性の値 (**img_w3bnz2p1t58q2xi0**) は、“**img_**”から後ろの値が動的に変化します。

変更前の RanoreXPath :

```
/form[@controlname='RxMainFrame']/?/?/tabpage[@controlname='RxTabDynamic']  
/text[@text='img_w3bnz2p1t58q2xi0']
```

対象のリポジトリアイテムのパスエディター画面のツリーの要素から、対象の要素 (**text** 要素) を選択し、属性の編集画面にて、text 属性の演算子を、**~ (Regex Match)** に変更し、属性値を指定する入力フォームに、正規表現を使用した値を (**img_.***) を設定します。



変更後の RanoreXPath :

```
/form[@controlname='RxMainFrame']/?/?/tabpage[@controlname='RxTabDynamic']  
/text[@text='img_.*']
```



ヒント

属性値の先頭の文字列が固定値であり、その文字列と一致するように指定する場合は **> (Starts With)** を使用します。属性値の最後の文字列が固定値であり、その文字列と一致するように指定する場合は **< (Ends With)** を使用します。

正規表現については、以下のユーザーガイドに記載されています。

- 正規表現構文

<https://support.ranorex.com/hc/en-us/articles/38071614574353-Regex-Syntax>

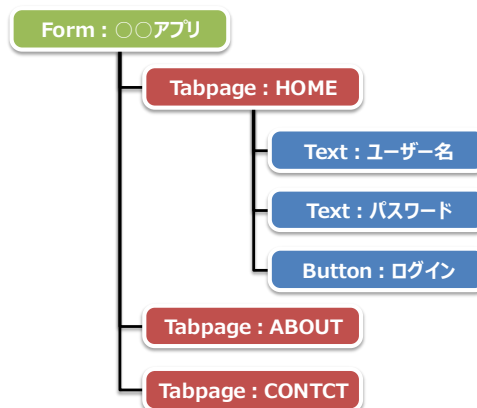
3.1.2.3. パス構造を簡略化する

アプリケーションのアップデートにより画面に変更が加えられることで、ツリー構造が変わり、リポジトリアイテムに登録された既存の RanoreXPath がオブジェクトを認識できなくなる場合があります。このような場合、ワイルドカード演算子を使用してパス構造を簡略化することで、パス構造の変更に柔軟に対応できます。

アップデート前のアプリケーションとツリー構造：



(アプリケーションの画面)

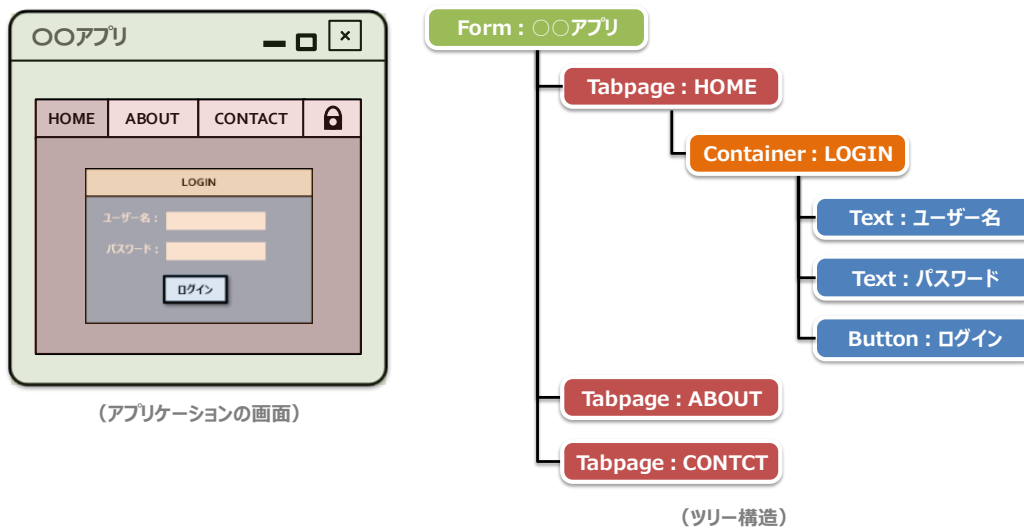


(ツリー構造)

```
/Form[@title='○○アプリ']/TabPage[@controlname='HOME']/Text[@controltext='ログイン']
```

(ログインボタンを認識する RanoreXPath)

アップデート後のアプリケーションとツリー構造：



```
/Form[@title='〇〇アプリ']/TabPage[@controlname='HOME']/Container[@controlname='HOME']/Text[@controltext='ログイン']
```

(ログインボタンを認識するRanoreXPath)



注意

ワイルドカードを広範囲に使用すると、意図しないオブジェクトを検出したり、検索範囲が広がることでパフォーマンスに影響が出る可能性があります。そのため、一意に対象のオブジェクトを特定できることを確認したうえで適用することを推奨します。

1. 本設定が有効なケース

- オブジェクトの階層構造が変更された/今後変更される可能性のある場合

2. 本設定のポイント

例として、前述した RanoreXPath の場合、タブページ内の画面構成に対して、今後変更される可能性がある場合は、**// (any descendants)** を使用することで、画面変更の影響を受けない RanoreXPath にします。

変更前の RanoreXPath：

```
/form[@title='〇〇アプリ']/tabpanel[@controlname='HOME']/button[@controltext='ログイン']
```

変更後の RanoreXPath：

```
/form[@title='〇〇アプリ']/tabpanel[@controlname='HOME']//button[@controltext='ログイン']
```



ヒント

中間階層の変化する範囲があらかじめ限定されている場合は、/*（any）や/?（any optional）を使用することでも、階層構造を簡略化して RanoreXPath を定義することができます。

- /*（any） : 対象の要素（階層）の内部情報を問わない
- /?（any optional） : 対象の要素（階層）の内部情報および存在有無を問わない
- //（descendant-or-self） : 設定した条件で使用する属性

ワイルドカードの使用例については、以下のユーザーガイドに記載されています。

- RanoreXPath の例

<https://support2.ranorex.com/ja/userguide/ranorex-studio-advanced/ranorexpath/ranorexpath-syntax-examples/>

3.2. メンテナンス工数を抑える方法

RanoreXPath の自動生成に関する設定として、「速度と堅牢性の設定」と「重み付けルールの設定」があります。これらの設定は、RanoreXPath を生成する前に、テスト対象アプリケーションの特性や運用方針に応じてあらかじめ設定しておくことで、変更に強いパスを生成しやすくなり、後からパスを修正する工数を抑えることができます。

設定	概要	参照先
速度と堅牢性の設定	対象オブジェクトに対する認識速度と認識精度のバランスを調整する設定。	3.2.1
重み付けルールの設定	RanoreXPath に使用する属性の優先度を調整する設定。	3.2.2

3.2.1. 速度と堅牢性の設定

3.2.1.1. 速度と堅牢性とは

テスト実行時におけるオブジェクトの認識の速さや、アプリケーションの画面の変更にともなうオブジェクトの認識の強さは、RanoreXPath の構造に大きく依存しますが、RanoreXPath は「速度」と「堅牢性（壊れにくさ）」の 2 つの軸を基にバランスを見て生成されます。

どちらの設定に重点を置くかは、テスト対象アプリケーションの内部構造や改修頻度に合わせて調整する必要がありますが、テスト自動化の長期運用において、ブラウザや Windows OS、テスト対象アプリケーションのアップデートが発生した際の RanoreXPath のメンテナンス工数を最小限に抑えるために、変更に強い RanoreXPath を生成することが重要です。テスト実行時におけるオブジェクトの認識の速さに影響しない場合は、将来発生する可能性のあるメンテナンス工数を抑えるために、原則として「堅牢性」重視の設定をおこなうことを推奨します。

3.2.1.2. 設定の効果（「堅牢性」重視型）

テスト対象オブジェクトを認識できる最低限の情報（要素、属性）を使用した RanoreXPath を生成します。テスト対象アプリケーションのアップデートによって見た目や内部構造に変更が入った場合も、テストが壊れにくい堅牢な RanoreXPath となります。



注意

「堅牢性」重点にした場合、画面の変更に強いオブジェクトになりますが、最低限の情報のみで RanoreXPath が生成されるため、初期設定時に比べてオブジェクト認識速度が低速になる場合があります。

3.1.1.1. 有効なケース

- テスト対象アプリケーションのアップデートにより、内部構造に変更が発生する可能性がある場合
- ブラウザーや Windows OS のアップデート後に、テスト対象オブジェクトの認識が安定しない場合



ヒント

「速度」重視型にした場合、テスト対象オブジェクトの親要素から子要素までの各階層情報が省略されずに、多くの要素が含まれた RanoreXPath を生成します。そのため、親要素から子要素までの各階層情報を使用してテスト対象オブジェクトを検出するため、オブジェクトの検索速度が向上します。

※「速度」重視型にした場合、オブジェクト認識速度は向上しますが、テスト対象オブジェクトの親要素から子要素までの各階層の要素は省略されないため、初期設定時に比べて画面の変更に弱い RanoreXPath になります。

設定の際は、以下のケースで有効です。

- テスト対象アプリケーションのアップデートが想定されない場合
- オブジェクト認識速度の向上を優先したい場合

3.1.2. 重み付けルールの設定

3.1.2.1. 重み付けルールとは

Ranorex で自動生成された RanoreXPath に使用される属性には、動的な情報が含まれる場合がありますが、テスト実行の度に属性値が変わることでオブジェクトが検出できずに失敗する場合があります。そのため、対象のオブジェクトの RanoreXPath を修正し、他のユニークな属性に変更するなどの修正が必要となります。

RanoreXPath の生成時に、どの属性を使用するかは「重み付けルール」の設定で管理されます。重み付けルールでは、各属性に数値が設定されており、数値が高い属性から優先的に使用されます。



注意

対応範囲が小規模な場合は RanoreXPath を手動で修正することで対応が可能ですが、対応範囲が広範囲に及ぶ場合は、重み付けルールを設定のうえ、RanoreXPath を再取得することを推奨します。

3.1.2.2. 設定の効果

テスト対象オブジェクトに動的な属性が存在することで、アプリケーションの状態変化にともないオブジェクト認識に失敗する場合や、特定の属性を優先的に使用したい場合に、重み付けルールを使用することで動的な属性を RanoreXPath に含めないようにすることができます。これにより、動的な属性を持つオブジェクトに対しても、柔軟な対応が可能となります。

3.1.2.3. 有効なケース

- 複数のテスト対象オブジェクトの RanoreXPath に動的な属性が使用されている場合
- テスト対象アプリケーションの特徴に応じて、特定の属性を優先的に使用したい場合

3.2. RanoreXPath 設計の進め方と推奨設定

RanoreXPath のメンテナンス工数を抑えるためには、変更に強いパス構造を理解したうえで、目的や状態に応じて適切な設計方法を選択することが重要です。以下では、RanoreXPath 設計における基本的な進め方と、代表的なケースごとの推奨される設定について説明します。

3.2.1. 基本的な進め方

Step1. 変更に強い RanoreXPath 構造を理解する

- RanoreXPath に影響を与える主な要因を理解する
- 変更に強い（メンテナンス工数を抑える）RanoreXPath の考え方を理解する

Step2. 変更に強い RanoreXPath を作成する

- 変更されにくい属性を使用する
- 属性値を正規表現で指定する
- パス構造を簡略化する

Step3. ケースに応じた RanoreXPath の生成設定をおこなう

- 速度と堅牢性の設定をおこなう
- 重み付けルールの設定をおこなう

3.2.2. ケース別の推奨設定

ケース	推奨される設定
一部のテスト対象オブジェクトの RanoreXPath に動的な属性が使用されている場合	変更されにくい属性をパスエディターから指定
属性値の一部が特定の形式で変化する場合	属性値を正規表現で指定
一部のオブジェクトの階層構造が変更された/今後変更される可能性のある場合	パス構造の簡略化
テスト対象アプリケーションのアップデートにより、内部構造に変更が発生する可能性がある場合	速度と堅牢性の設定（堅牢性重視型）
ブラウザや Windows OS のアップデート後に、テスト対象オブジェクトの認識が安定しない場合	速度と堅牢性の設定（堅牢性重視型）
オブジェクト認識速度の向上を優先したい場合	速度と堅牢性の設定（速度重視型）
複数のテスト対象オブジェクトの RanoreXPath に動的な属性が使用されている場合	重み付けルールの設定
テスト対象アプリケーションの特徴に応じて、特定の属性を優先的に使用したい場合	特定の属性をパスエディターから指定 重み付けルールの設定

4. RanoreXPath の設定手順



ヒント

テスト自動化を成功させる方法の 1 つとして、一度にすべてのテストケースを作成せずに、段階的に作成することが重要です。

まずは基本的な画面操作や、影響度や使用頻度の高い操作に対してテストケースを作成し、アプリケーションの内部構造や生成される RanoreXPath の特性を理解したうえで、どのような設定をおこなうかを検討することを推奨します。

すべてのテストケースを作成した後で、RanoreXPath の設定やリポジトリに登録したオブジェクト（RanoreXPath）を修正しようとすると、本来修正しなくても良いパスも修正することになり、メンテナンス工数が掛かります。初期段階で適切な設定をおこなうことで、将来的にパスを修正する工数も軽減されます。

4.1. 速度と堅牢性の設定

4.1.1. 速度と堅牢性の設定

- (1) Ranorex Studio のメニューバーから、設定（=）ボタンをクリックします。
- (2) 詳細タブを開きます。
- (3) RanoreXPath の設定の速度/堅牢性のスライダーを、堅牢性にスライドします。

※速度重視型の場合は、速度にスライドします



4.1.2. ソリューション設定の共有

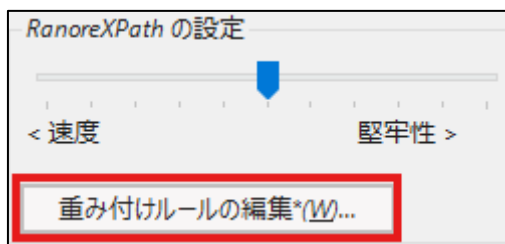
「速度と堅牢性の設定」などをおこなったソリューションの設定内容は、既存の設定ファイルのインポートが可能なため、別環境にエクスポートすることが可能です。

- (1) Ranorex Studio で対象のソリューションを開きます。
- (2) 設定画面左下のエクスポートをクリックし、設定ファイルを任意の場所に保存します。
- (3) 別環境に設定ファイルを移動した後、手順(1) –(2)を実施します。
- (4) 設定画面の左下の「インポート」をクリックし、手順(3)でエクスポートした設定ファイルを選択します。

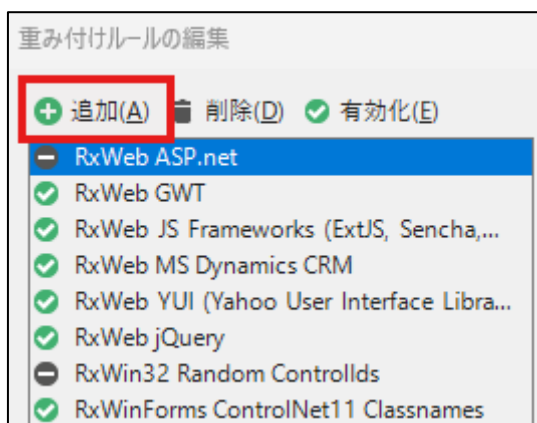
4.2. 重み付けルールの設定

4.2.1. 重み付けルールの設定

- (1) 詳細タブを開き、重み付けルールの編集をクリックします。



- (2) 重み付けルールの編集画面で、追加をクリックします。



(3) 重み付けルールの編集画面で、追加をクリックします。

追加された Unnamed Rule を選択し、以下のルールの設定をおこないます。

- ・ 名前 : 新たに追加するルールの名前 (任意)
- ・ 機能 (Capability) : 対象の属性が指定されている機能 (例: control) (※ 1)
- ・ 属性 (Attribute) : 対象の属性の名前 (例: controltext) (※ 1)
- ・ 重み付け : 対象の属性の重み (例: 200) (※ 2)

重み付けルールの編集

+ 追加(A) - 削除(D) - 無効化(I)

ルールターゲット

名前: Unnamed Rule

機能(Capability): control

属性(Attribute): controltext

重み付け: 200 属性の一覧の表示...

マッチ条件

☐ すべて ☒ いずれか

ルール条件

+ 条件の追加

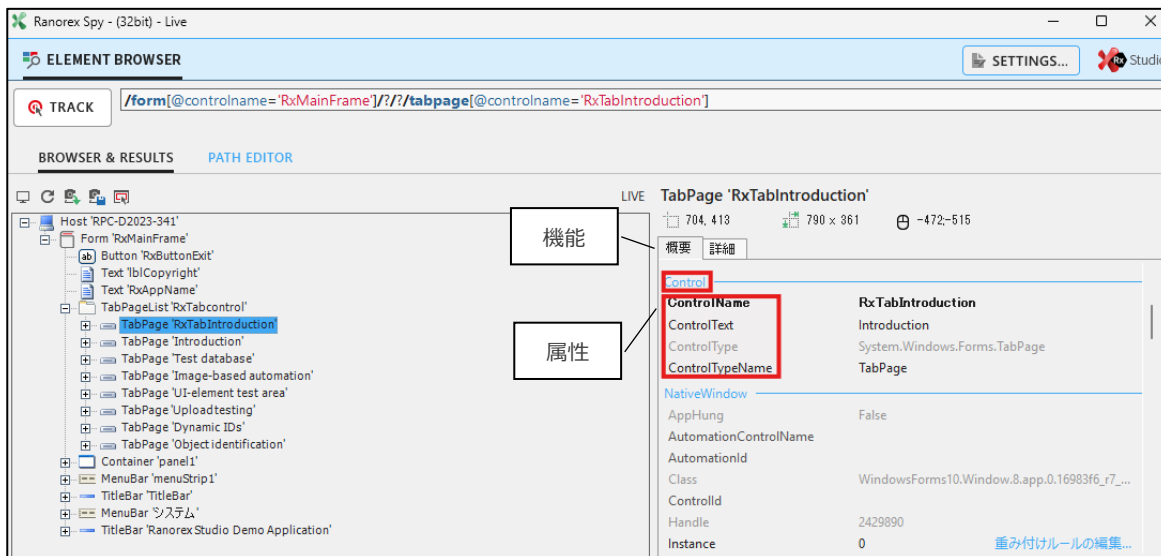
(4) 重み付けルールを設定後、Ranorex Spy で対象のオブジェクトを再度トラッキングすると、指定した属性が RanoreXPath に使用されます。例として、controltext 属性を指定した場合は以下のようなパスとなります。

**/form[@controlname='RxMainFrame']/?/?/tabpage[@controlname='RxTabUIElement
s']/?/?/button[@controltext='Reset']**

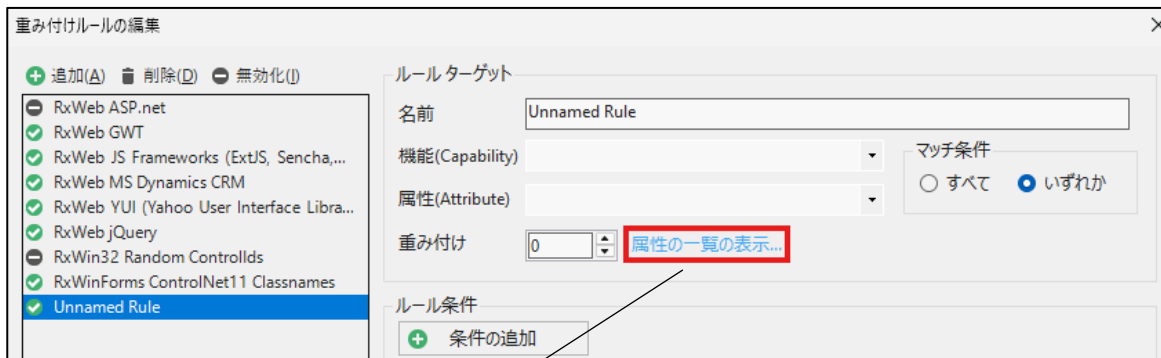


メモ

※ 1 機能 (Capability) および属性 (Attribute) は、Ranorex Spy でテスト対象オブジェクトをトラッキングした際の、概要タブから確認することができます。



※ 2 各属性に設定されている重み付け (数値) は、属性の一覧の表示から確認することができます。



属性の一覧				
機能(Capability)	属性(Attribute)	説明	重み付け	タイプ
WinForms Plugin				
Control	ControlName	The name of the Windows Forms control.	150	String
	ControlText	The text of the Windows Forms control.	100	String
	ControlType	The full type name of the Windows Forms control.	50	String
	ControlTypeName	The (short) type name of the Windows Forms control.	105	String
ControlNet11	ControlName	The name of the Windows Forms control.	150	String
WinForms.Core Plugin				
WinformsCoreCont...	ControlName	The name of the Windows Forms control.	150	String
	ControlText	The text of the Windows Forms control.	100	String
	ControlType	The full type name of the Windows Forms control.	50	String
	ControlTypeName	The (short) type name of the Windows Forms control.	105	String
	FullTypeLineage	The full type name lineage of the Windows Forms control.	90	String[]



ヒント

重み付けルールでは、条件の追加を使用することで、対象となる属性の条件を細かく設定し、ルールを適用したいケースのみに制限することができます。

- ・ マッチ条件 : すべての条件に一致するか、いずれか 1 つの条件に一致するかの条件
- ・ ルール条件（ソース） : 設定した条件で使用する属性が参照する要素
 - ・ self = 対象の要素自身
 - ・ parent = 対象の要素の親要素
 - ・ toplevel = 対象の要素のトップレベルの祖先の要素
- ・ ルール条件（属性） : 設定した条件で使用する属性
- ・ ルール条件（正規表現） : 選択した属性値にマッチさせるための正規表現

重み付けルールおよび正規表現については、以下のユーザーガイドに記載されています。

- ・ 動的な UI 要素

<https://support.ranorex.com/hc/en-us/articles/38041596836753-Introduction-to-Dynamic-UI-Elements>

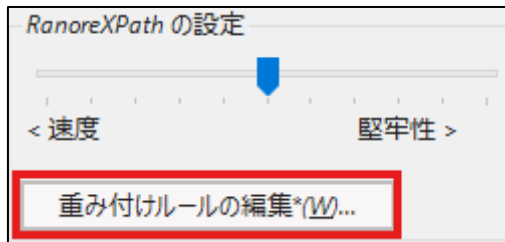
- ・ 正規表現構文

<https://support.ranorex.com/hc/en-us/articles/38071614574353-Regex-Syntax>

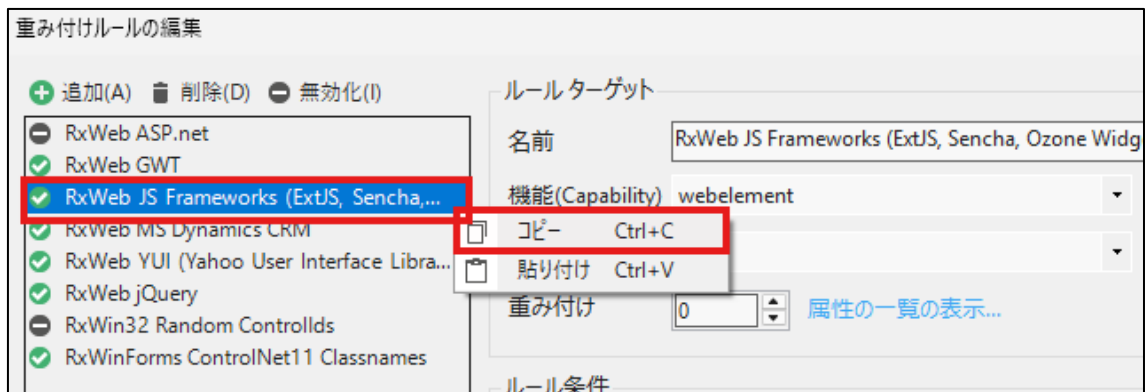
4.2.2. 重み付けルールの共有

Ranorex Studio は、重み付けルールを XML ファイルとして保存します。重み付けルールの共有は、重み付けルールの編集より、ルールの内容をコピー & ペーストすることでおこなうことができます。

- (1) Ranorex Studio のメニューバーから、設定 (=) ボタンをクリックします。
- (2) 詳細タブを開き、重み付けルールの編集をクリックします。



- (3) 共有したいルールを右クリックし、コピーをクリックします。



- (4) 任意のテキストエディタに貼り付け、ファイルを保存し、別環境にファイルを格納します。

```
<rules>
<rule>
  attribute="id"
  capability="webelement"
  conditionsoperator="or"
  enabled="True"
  name="RxWeb JS Frameworks (ExtJS, Sencha, Ozone Widget ,...)"
  setweight="0"
  <condition>
    attribute="id"
    match="^ext-.*$.*"
    negate="False"
    source="self"
  </condition>
  <condition>
    attribute="id"
    match="[a-z]{4}(-[a-z]*(-[a-z]*)?)"
    negate="False"
    source="self"
  </condition>
</rule>
</rules>
```

- (5) 別環境でファイルの内容をコピーした後、手順(1) - (2)を実施します。



テクマトリックス株式会社 ソフトウェアエンジニアリング事業部
Ranorex テクニカルサポートセンター

E-MAIL ranorex-support@techmatrix.co.jp