

WinActor

---

## シナリオ作成ガイドライン

**Version 2.00**

**2019/06/10**

**NTT DATA CORPORATION**

## Release note

2017/04/01	新規作成	池田
2017/07/10	制定	小山
2019/06/10	表題及び本文の全面改訂（旧：WinActor シナリオ構築規約）	中原/久門

### **ご注意**

- ・ 本書の内容の一部または全部を、営業目的で無断利用・無断配付することは禁止されています。
- ・ 本書の内容の一部または全部を、無断で転載することは禁止されています。
- ・ 本書の内容は、将来予告なしに変更することがあります。
- ・ 本書の内容は、万全を期して作成しておりますが、ご不審な点や誤り、記載漏れなどお気づきの点がありましたら、発行元までご連絡ください。
- ・ 本システムを利用したことにより発生したユーザの損害およびユーザが第三者に与えた損害については、上記にかかわらず責任を負いません。

複製厳禁・無断転載禁止

---

## Table of contents

<b>1 はじめに .....</b>	<b>4</b>
1.1 本書の目的 .....	4
1.2 本書の位置付け .....	4
1.3 本書の更新 .....	4
<b>2 シナリオ構成 .....</b>	<b>5</b>
2.1 シナリオの構成要素 .....	5
2.2 シナリオ作成の基本方針 .....	6
2.2.1 メインタブ .....	6
2.2.2 タブ .....	6
2.2.3 サブルーチン .....	6
2.2.4 グループ .....	7
<b>3 命名ルール .....</b>	<b>8</b>
3.1 タブ .....	8
3.2 サブルーチン .....	8
3.2.1 サブルーチンのコメント欄 .....	9
3.3 グループ .....	9
3.3.1 グループのコメント欄 .....	9
3.4 ノード/ライブラリ .....	10
3.4.1 ノード/ライブラリのコメント欄 .....	10
3.5 変数 .....	10
<b>4 禁止事項 .....</b>	<b>12</b>
<b>5 シナリオ作成ポイント .....</b>	<b>13</b>
5.1 シナリオ全体に関するポイント .....	13
5.2 ノード/ライブラリ使用上のポイント .....	16
5.2.1 ノード「分岐」「多分岐」「繰り返し」「後判定繰返」 .....	16
5.2.2 ノード「例外処理」 .....	16
5.2.3 ノード「サブルーチングループ」「サブルーチン呼び出し」 .....	17
5.2.4 ノード「画像マッチング」 .....	18
5.2.5 ノード「ウィンドウ状態待機」 .....	18
5.2.6 ノード「文字列送信」 .....	18
5.2.7 IE モード(ライブラリ「自動記録アクション」) .....	18
5.3 特定操作を実現する上での作成ポイント .....	19
5.3.1 ウィンドウを識別できない画面への操作 .....	19
5.3.2 エミュレーションの操作 .....	19
5.3.3 アプリケーション起動やファイル起動の操作 .....	20

# 1 はじめに

## 1.1 本書の目的

本書は、品質の高い WinActor シナリオ(以下、シナリオと呼ぶ)を作成することを目的としている。

「品質の高いシナリオ」とは、以下の観点を満たしているものを指す。

- シナリオ実行中にエラーで停止せず、想定した実行結果となること
- シナリオからどのような操作を行っているか容易に読み取れること
- シナリオ修正が容易であること
- OS やソフトウェアのアップデートによるシナリオ実行への影響を最小限にすること
- シナリオに無駄な操作が含まれないこと

また、本書の想定する読者として、シナリオ作成に従事する方とする。

## 1.2 本書の位置付け

本書は、NTT アドバンステクノロジー株式会社の RPA ツール、WinActor を個人としてではなく組織として導入する場合を想定している。組織で導入する場合、シナリオ作成・保守・運用は複数人で行う為、属人的ではなく、統一的なルールに従う必要がある。

したがって本書は、WinActorにてシナリオ作成する際に準拠することを想定しており、品質確認時には作成したシナリオが本書を満たしているかを確認する。

ただし、本書はシナリオ作成の基本となり得る方式を記載しているものであるため、シナリオを作成する各団体に適宜ルールの追加、修正、取捨選択を行うことで最適化をする必要がある。

## 1.3 本書の更新

本書は、適用業務や環境及び社内規約を鑑み、シナリオ作成者・ユーザからの意見を反映し、適宜見直し改善していく必要がある。

## 2 シナリオ構成

### 2.1 シナリオの構成要素

シナリオは以下の要素から構成される。

- ① メインタブ：                   メインの操作を配置する  
シナリオは「開始」から順番に実行し、「終了」に到達すると完了する
- ② タブ：                         サブルーチンを配置する
- ③ サブルーチン：               「同一画面やアプリケーションへの一連の操作」または「シナリオの中で何度も必要とされる定型的な操作」を定義する
- ④ グループ：                   自動化対象作業における 1 つの操作を定義する
- ⑤ ノード：                     WinActor における 1 つの操作を定義する

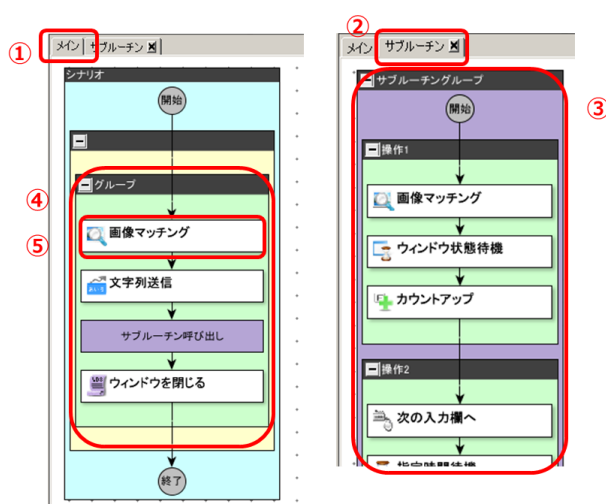


図 2-1 シナリオ構成

## 2.2 シナリオ作成の基本方針

### 2.2.1 メインタブ

- シナリオはできるだけシンプルにすること  
メインタブ内のノードが多くなり、見づらい場合には適宜サブルーチンを作成すること。

### 2.2.2 タブ

- 1つのサブルーチンにつき、1つのタブを作成すること  
タブが多くなった場合の各サブルーチンへの移動は、パレットエリア「サブルーチンタブ」から行うと良い。

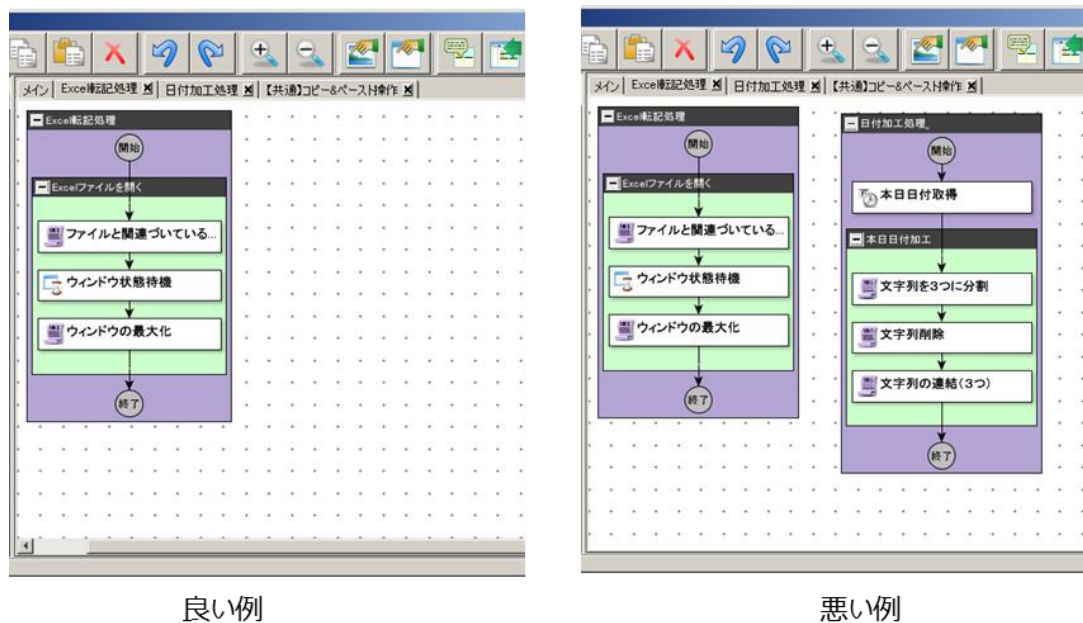


図 2-2 タブ使用例

### 2.2.3 サブルーチン

- 「同一画面やアプリケーションへの一連の操作」または「シナリオで何度も必要とされる定型的な操作」をサブルーチンとして作成すること  
1つのサブルーチン内のノードが多くなり見づらい場合は、サブルーチンから更に別サブルーチンへ分割すること。

※「画面やアプリケーション単位での一連の操作」例

- ・ IE で検索し、結果から情報を取得する
- ・ Excel から必要な項目を取得し、別 Excel へ転記する
- ・ 社内システムから Excel ファイルをダウンロードする

## 2.2.4 グループ

- 自動化対象作業における 1 つの操作ごとにグループを作成すること

操作は必ずしもノード 1 つで実現できるものではなく、操作を実現するための関連ノードを同じグループに含めること。

グループに属するノードが 1 つになる場合に限り、グループ化せずに配置してもよい。

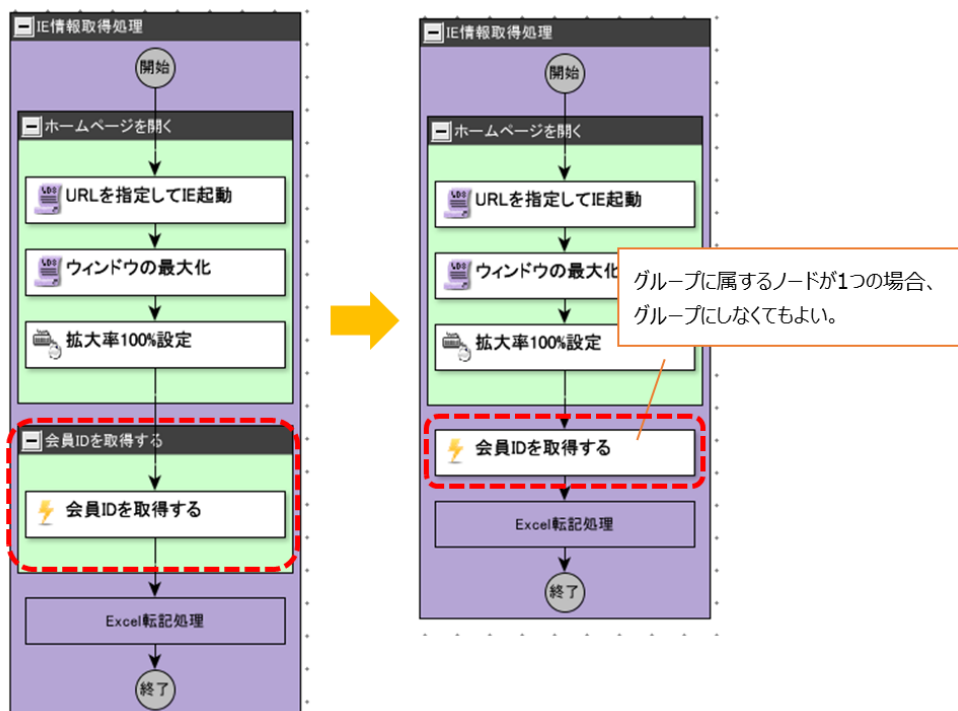


図 2-3 グループ使用例(ノード 1 つの場合)

自動化対象作業における 1 つの操作とは、作業マニュアルの 1 つの手順や作業担当者からみた 1 つの操作を指し、以下のように、WinActor における 1 つの操作(ノード)とは異なる。

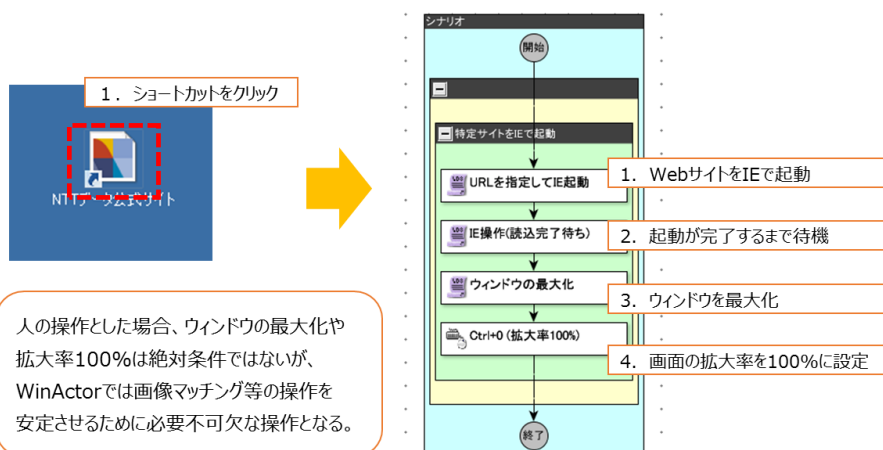


図 2-4 人の操作と WinActor の操作の違い(IE 起動)

## 3 命名ルール

### 3.1 タブ

- タブ名は、そのタブに配置しているサブルーチンと同名にすること
  - ※ 後述する 3.2 のうち「呼び出し元サブルーチン名も記載する」ルールは適用しなくて良い。
  - ※ 共通的に使用するサブルーチンの場合、タブ名は省略せず、サブルーチンと同じにすること。

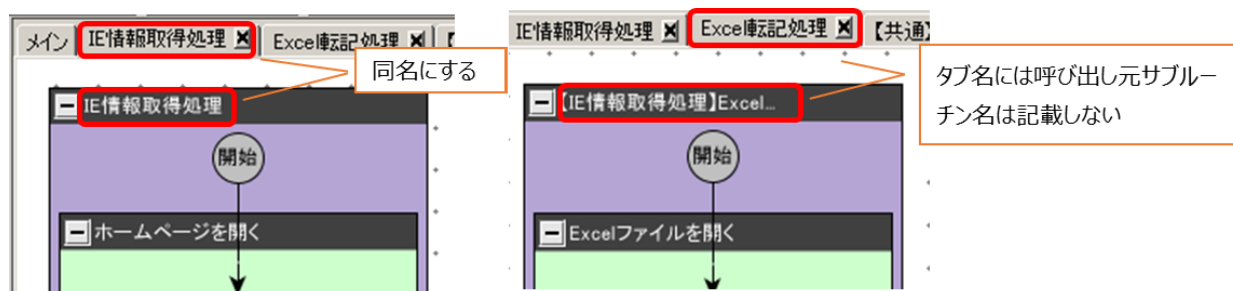


図 3-1 タブの命名ルール例

### 3.2 サブルーチン

- サブルーチン名は「○○処理」または「○○操作」にすること
- サブルーチンから別サブルーチンを呼び出す場合、呼び出し先のサブルーチン名は「【呼び出し元サブルーチン名】呼び出し先サブルーチン名」にすること

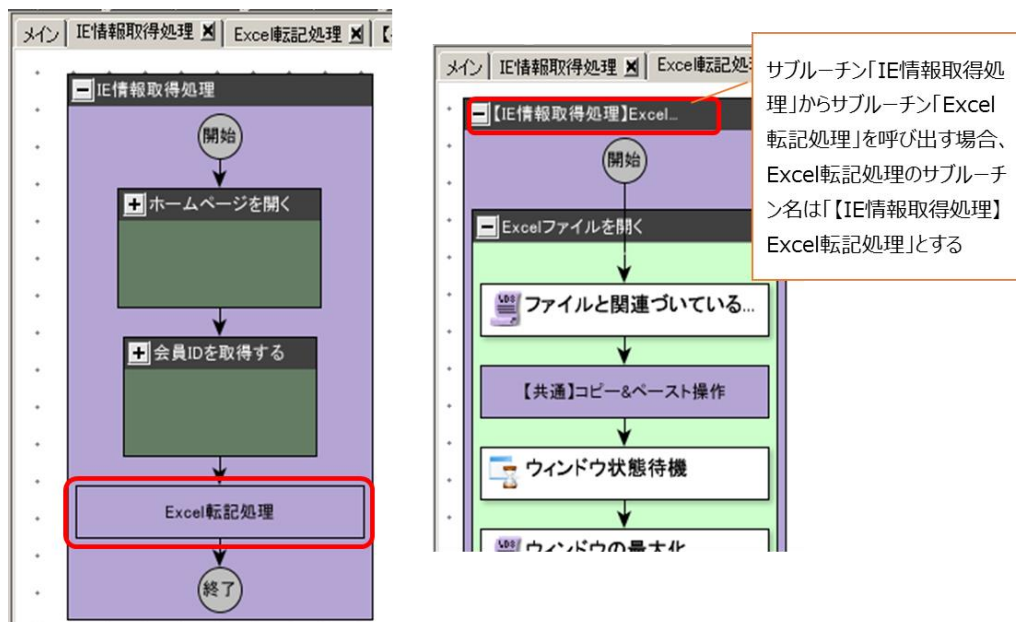


図 3-2 サブルーチンの命名ルール例



- 複数の箇所から呼び出されるサブルーチンの名前は、【共通】呼び出し先サブルーチン名にすること

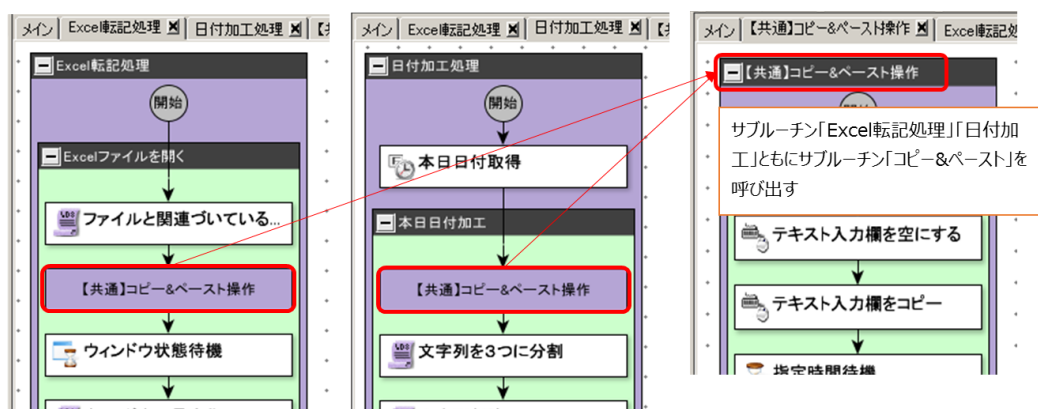


図 3-3 共通サブルーチンの命名ルール例

### 3.2.1 サブルーチンのコメント欄

- 名前欄だけではサブルーチンの操作・目的が判断できない場合はコメント欄に記載すること

## 3.3 グループ

- グループ名は「対象+動作」で記載すること

(例)

- ・○○ボタンを押下する/○○ボタン押下
- ・顧客 ID を入力する/顧客 ID 入力

作業マニュアル等がある場合、グループ名は作業マニュアルの手順/操作と紐付く名前にしてもよい。その場合、グループ名は「対象+動作」となっていなくてもよい。

### 3.3.1 グループのコメント欄

- 名前欄だけでは、グループの操作・目的が判断できない場合はコメント欄に記載すること

### 3.4 ノード/ライブラリ

- ノード/ライブラリ名は基本的に「対象+動作」で記載すること

（例）

- ・○○ボタンを押下する/○○ボタン押下
- ・顧客 ID を入力する/顧客 ID 入力

対象のノード/ライブラリがグループに属しており、グループ名から操作が連想できる場合には、グループ名のみを変更し、ノード/ライブラリは変更しなくてもよい。

- ノード「サブルーチン呼び出し」の名前は呼び出し先サブルーチン名とすること

※ 3.2 のうち「呼び出し元サブルーチン名も記載する」ルールは適用しなくて良い。

#### 3.4.1 ノード/ライブラリのコメント欄

- 名前欄だけでは、ノード/ライブラリの操作・目的が判断できない場合はコメント欄に記載すること
- ライブラリの名前を変更する場合は、コメントにもとのライブラリ名を残しておくこと

### 3.5 変数一覧

#### 3.5.1 グループ名

- 変数グループは操作対象や一連の操作など、区別が明確となるように作成すること
  - ・ 1 つの変数グループに属する変数の数が多かった場合、適宜グループを細分化すること
  - ・ ファイル(Excel,PDF など)に関する変数は、対象ファイルが分かる変数グループ名にすること
  - ・ ブラウザやその他アプリケーション操作に関する変数は、対象画面や一連の操作が分かる変数グループ名にすること
- 共通的に使用する変数は 1 つのグループにすること

画像マッチングの状態取得結果に使用する変数(「結果」など)やカウンタのように、特定の処理やファイルではなく共通的に使用する変数は、1 つのグループとすること。

グループ名	変数名	現在値	初期化しない	初期値
共通	結果		<input type="checkbox"/>	
共通	チェック済判定		<input type="checkbox"/>	
定義ファイル	■ 定義ファイル 取...		<input type="checkbox"/>	取得地域
定義ファイル	■ 定義ファイル 取...		<input type="checkbox"/>	取得データ
定義ファイル	定義ファイル ファイ...		<input type="checkbox"/>	
定義ファイル	定義ファイル 最終行		<input type="checkbox"/>	
定義ファイル	定義ファイル 行番号		<input type="checkbox"/>	
定義ファイル	定義ファイル 取得...		<input type="checkbox"/>	
定義ファイル	定義ファイル 取得...		<input type="checkbox"/>	

複数のノードにて共通して使用する変数は、1つのグループにする。  
ここではグループ名「共通」

図 3-4 共通変数のグループ設定例

### 3.5.2 変数名

- 変数名は【対象】\_【内容】で記載する
  - ・ 【対象】はファイル名や画面名が分かるように記載する  
(例)  
「会員情報ファイル.xlsx のファイルパス」を設定する変数の場合  
⇒ 「会員情報\_ファイルパス」
  - ・ 【内容】には「ファイルパス」「取得値」など、変数の役割が分かるように記載する
  - ・ 【対象】\_【内容】だけでは対象が明確にならない場合、【対象】\_【内容】(補足)の形式で補足情報を記載する  
(例)  
会員情報\_取得値(会員 ID)、会員情報\_取得値(氏名) など
- 不要な変数は全て「結果」とする  
画像マッチングの状態取得結果や文字列分割した際の不要な値など、その後シナリオ内で使用しない変数の名前は全て「結果」とする
- 変数はインプット元(取得元)を基準とした変数名とする  
例えば、Excel から取得し IE に転記する変数の場合は、取得元である Excel を基準にした変数名(「Excel\_取得値」など)を設定する ※「IE\_設定値」としない。
- 定数(固定値)には接頭語を付与する
  - ・ ファイルパスのように、格納している値が変更されない変数(定数)の場合、定数であることが分かるように接頭語を付与すること  
(「●」を接頭語とした場合、「●会員情報\_ファイルパス」など)
  - ・ 定数は、各変数グループの先頭に配置すること

### 3.5.3 コメント

- 変数に関する補足がある場合はコメント欄に記載する
  - ・ 変数名だけでは使用用途の判断が難しい場合、コメント欄に補足説明を記載すること
  - ・ true/false を格納する変数の場合は、true/false となる条件をコメントに記載すること  
(例)  
コメント 「画面が表示された場合「true」、教示されなかった場合「false」を格納する。」

## 4 禁止事項

以下に禁止事項を記す。

禁止事項はシナリオに組み込むことで品質が著しく低下するため、使用を認めないものとする。

### ■ 「%変数%」は使用しない

変数に格納されている値を使用する際、一部のノードでは「%変数%」の形式を使用することができるが、この形式で変数を設定すると、変数一覧から該当変数の使用箇所を確認することができない。

【参考】

WinActor\_操作マニュアル.pdf 「7.1.6 %変数名%の利用」

### ■ ノード「Excel 操作」は使用しない

ノード「Excel 操作」はシナリオ実行のたびに新規の Excel プロセスを起動する等の動作が行われるため、シナリオの予期せぬ動作につながる恐れがある。

※ライブラリ「Excel 操作(データ一覧連携)」も同ノードのため使用しない。

### ■ ノード「文字列送信」でリターンキー送信は使用しない

文字列送信のプロパティ画面にて、「リターンキー送信」のチェックボックスにチェックを入れると、意図しない箇所でリターンキーが送信される場合がある。

文字列送信後にリターンキーを入力(改行)したい場合は、エミュレーション等を使用すること。

【参考】

WinActor\_操作マニュアル.pdf 「6.4.4.2 文字列送信のプロパティ」

### ■ ライブラリ「Excel 操作(全て閉じる)」「IE 操作(全て閉じる)」は使用しない

これらのライブラリは、未保存のファイル/ウィンドウに対しても保存確認ダイアログを表示せずに閉じてしまう場合(WinActor のバージョンにより異なる)があり、予期せぬ動作につながる恐れがある。

## 5 シナリオ作成ポイント

以下にシナリオ作成ポイントを記す。

シナリオ作成ポイントは、より高品質なシナリオを作成するために気をつけるべきことやテクニックであり、「シナリオ全体に関するポイント」「ノード/ライブラリ使用上のポイント」「特定操作を実現する上での作成ポイント」から構成される。

### 5.1 シナリオ全体に関するポイント

#### ■ 使用ノードの優先順序

操作の実現方法が複数考えられる場合は、下記の優先順序によるシナリオ作成を推奨する。

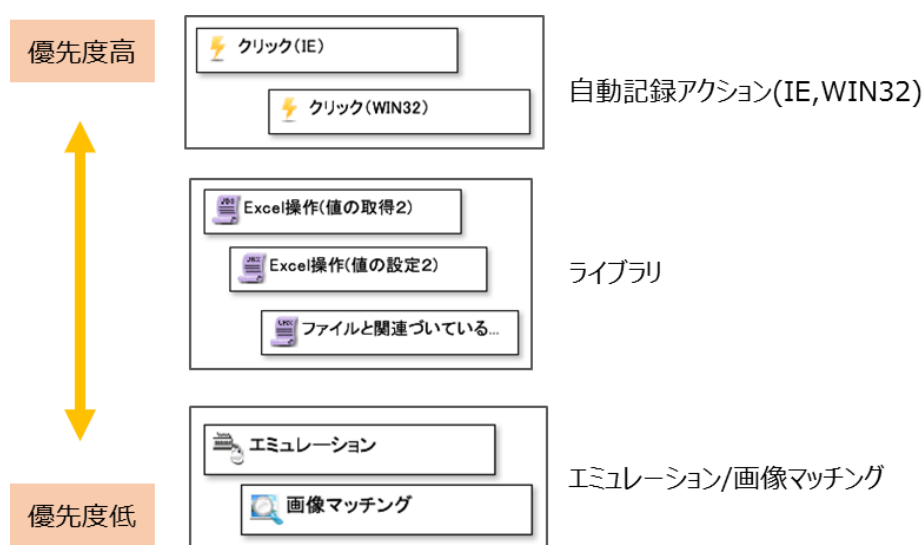


図 5-1 使用ノードの優先順序の例

#### ■ シナリオのメンテナンス等で変更する可能性がある値には、変数の使用を推奨する

例えばファイルパスに関して「値⇒○○」による指定を行っている場合、ファイルの格納場所が変更した際に、該当ノード全てのファイルパスを変更する必要がある。

そのような場合は「値⇒○○」による直接の値設定は行わず、変数一覧の初期値等を使用することで値の一元管理を行うと良い。

また、定期的に変更しなければならないパスワードや、変更される可能性の高いファイルパスなどについては別途設定ファイルを設け、シナリオ内で設定ファイル(Excel ファイル等)から設定値を読み込む仕組みにすることでメンテナンス性の向上を図ると良い。

---

**■ シナリオ全体を例外処理で困う**

シナリオ実行中のエラーによってシナリオが一時停止しないよう、シナリオ全体をノード「例外処理」の正常系に入れること。また、異常系にはエラーの内容を取得できる処理を設定すること。

**【エラーを取得するライブラリ例】**

- ・ エラー情報収集
- ・ 画面キャプチャ(デスクトップ)
- ・ デバッグ：ウィンドウ状態保存
- ・ デバッグ：変数値保存
- ・ SPV エラー情報収集

※エラー情報は、クリップボードに設定されるため保存されませんのでご注意ください。

**■ 不要な部品は削除する**

使用していないノード、変数、ウィンドウ識別ルール、イメージ等はシナリオ内に残さないこと。

**【参考】**

WinActor\_操作マニュアル.pdf 「5.4 変数一覧画面」

「5.9 イメージ管理画面」

「5.10 ウィンドウ識別ルール画面」

**■ 1025 文字以上の文字列は変数に格納できない(WinActor Ver6 では文字数制限を解除できるため、解除することで対応可能)**

表の全コピー等で 1025 文字以上の値を扱う可能性がある場合は、変数への格納を行わないシナリオにすること。

- 1025 文字以上の文字列は切り捨てられるが、WinActor 上でエラーは発生しないことがあるため、シナリオの不具合を検知しづらい
- 1025 文字以上の文字列を加工する場合は、メモ帳や Excel 等の機能を使用すると良い

## ■ シナリオの動作が不安定な場合には待機処理を入れる

シナリオの動作が不安定になる要因の 1 つとして、画面遷移等のタイミングでシナリオの操作が早すぎることを考えられる。そのため、動作が不安定になりそうな箇所(画面遷移やアプリケーション起動の直後)には適当な位置に待機処理を組み込むこと。

待機方法は以下のように複数パターンが考えられるため、状況に応じて使い分けること。

### ① ノード「ウィンドウ状態待機」

画面遷移やアプリケーション起動が完了するまで待機させることができる。

### ② ライブラリ「IE 操作(読み込み完了待ち)」

IE 上の画面遷移等で読み込みが発生している間待機させることができる。

### ③ ノード「繰り返し」+ ノード「画像マッチング」or ライブラリ

読み込みが発生しないウィンドウ内の変化やメール監視、フォルダ監視を行う場合は、「繰り返し + ノード/ライブラリ」の組み合わせにより待機させることができる。

※ 画像マッチングの失敗等により無限ループになることを避けるため、「N 回繰り返ししても待機が終了しない場合、強制的に繰り返しを終了する」のような操作も加えると良い。

### ④ ノード「指定時間待機」

①～③のいずれも対応できない場合は、指定時間待機を使用する。

エミュレーションの場合、プロパティ内に設定している操作全体の始めと最後に 300 ミリ秒程度の待機時間を設定すること。また、各操作(Ctrl+C などの単位)の間にも適宜待機時間を設定すること。

## ■ ウィンドウ識別ルールの最適化

ウィンドウタイトルが可変のウィンドウを操作する場合、ウィンドウ識別ルールの識別方式は「一致する」ではなく「を含む」等を指定すること。

## 5.2 ノード/ライブラリ使用上のポイント

### 5.2.1 ノード「分岐」「多分岐」「繰り返し」「後判定繰返」

- 分岐や繰り返し等の条件式では、不等号は『<』『≤』を使用すること

### 5.2.2 ノード「例外処理」

- 例外処理は、予期せぬエラーに対する操作を行う場合のみ使用すること  
ノード「分岐」「多分岐」による判断が可能な箇所には、例外処理は極力使用しないこと。

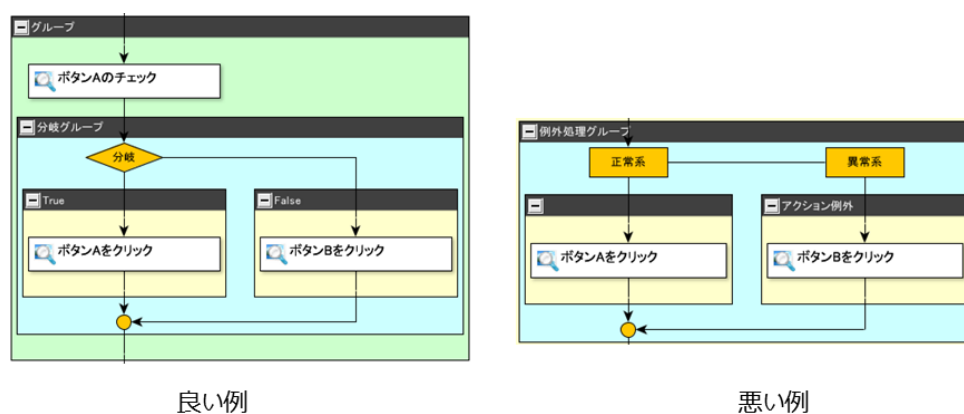


図 5-2 分岐箇所では例外処理を使用している例



### 5.2.3 ノード「サブルーチングループ」「サブルーチン呼び出し」

- サブルーチンのみで使用する変数がある場合は、ローカル変数や引数の活用を検討する

#### 【注意】

サブルーチン呼び出しの中で引数に値を設定しているときに、サブルーチン内のローカル変数を追加/削除すると、サブルーチン呼び出しに設定した引数の値が削除されるため、注意すること。

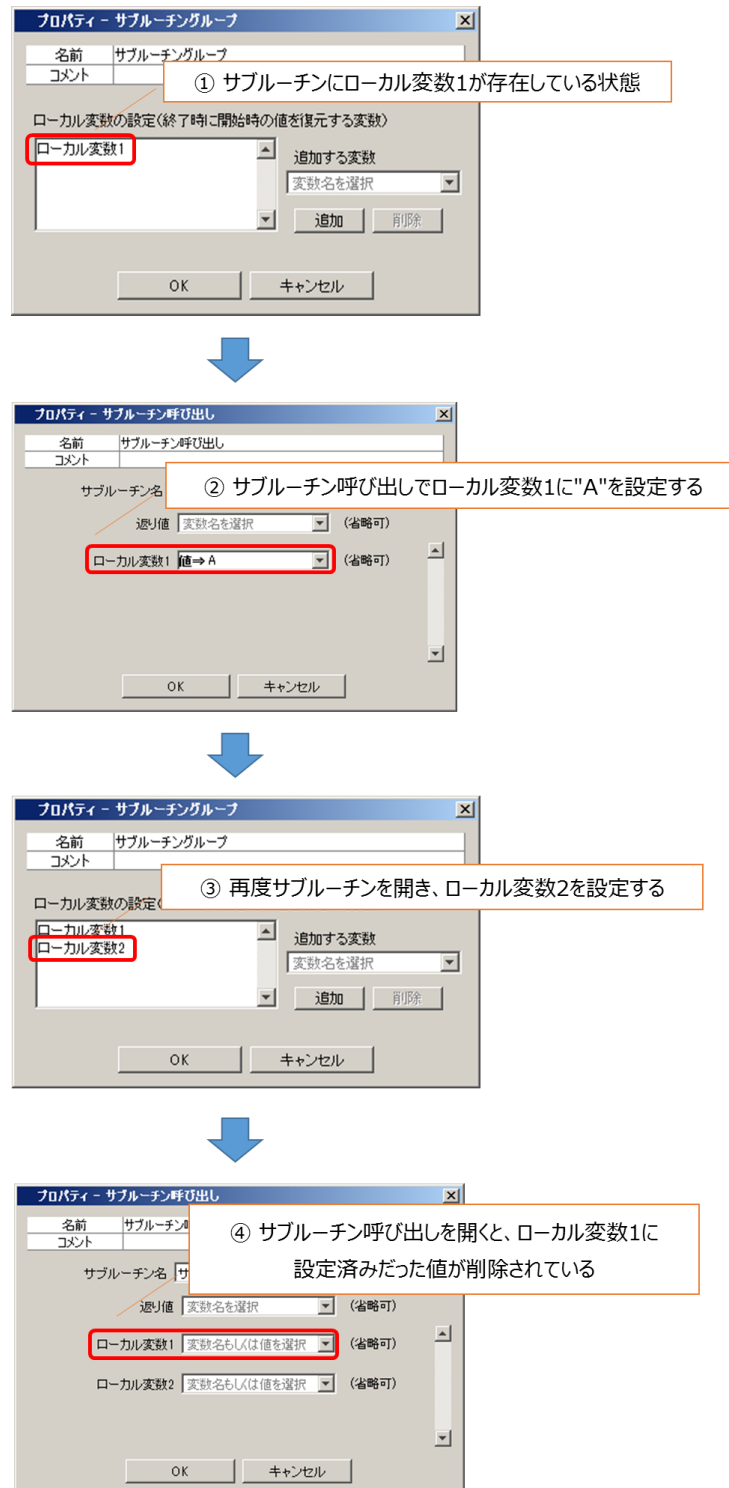


図 5-3 サブルーチン呼び出し内の引数が削除される事象の例

#### 5.2.4 ノード「画像マッチング」

- 画像マッチングを使用する際は、画面サイズや拡大率を記録時と同じにする動作もシナリオに組み込む
- マッチ率は、誤認識を防ぐためになるべく高く設定する
- マッチング箇所(プロパティ：マッチング画像)はできるだけ狭い範囲で囲む



図 5-4 マッチング画像の設定例

#### 5.2.5 ノード「ウィンドウ状態待機」

- プロパティの【画面の変化】は、待機後のシナリオ操作に応じて設定すること

#### 5.2.6 ノード「文字列送信」

- コマンドプロンプトのように、ライブラリやエミュレーション「Ctrl+V」による文字設定ができない場合のみ使用すること

#### 5.2.7 IE モード(ライブラリ「自動記録アクション」)

- 詳細設定タブを確認し、「name」や「id」も取得できている場合は、それらの要素にもチェックを入れること。また、以下にも注意すること。
  - 「name」や「id」によって対象を一意に特定できる場合は、「tag index」のチェックは外す。
    - ※ Webサイトの仕様次第で「name」「id」が一意でない場合もあるため注意する。
- 「tag index」にチェックを入れる場合、操作対象ページの更新等で「tag index」が変化しないことを必ず確認する

## 5.3 特定操作を実現する上での作成ポイント

### 5.3.1 ウィンドウを識別できない画面への操作

- 記録対象アプリケーション選択によるウィンドウ識別ができない画面へのキーボード操作は、【ウィンドウ識別名】を「(スクリーン)」としたエミュレーションを使用する  
※ ライブラリ「エミュレーションで文字列送信」でも同様の操作が可能だが、半角英数字以外の操作を行うことができないため、エミュレーションで統一すること。

### 5.3.2 エミュレーションの操作

- 1つのエミュレーションには1つの操作を設定することを推奨する  
(例)コピー、ペースト、全選択、その他ショートカット など  
複数の操作を1つのエミュレーションで設定すると、内部の操作が見えづらくなる。  
1つのエミュレーションで複数の操作を設定する場合には、操作が推測しやすいようなノード名にするなど工夫をすること。

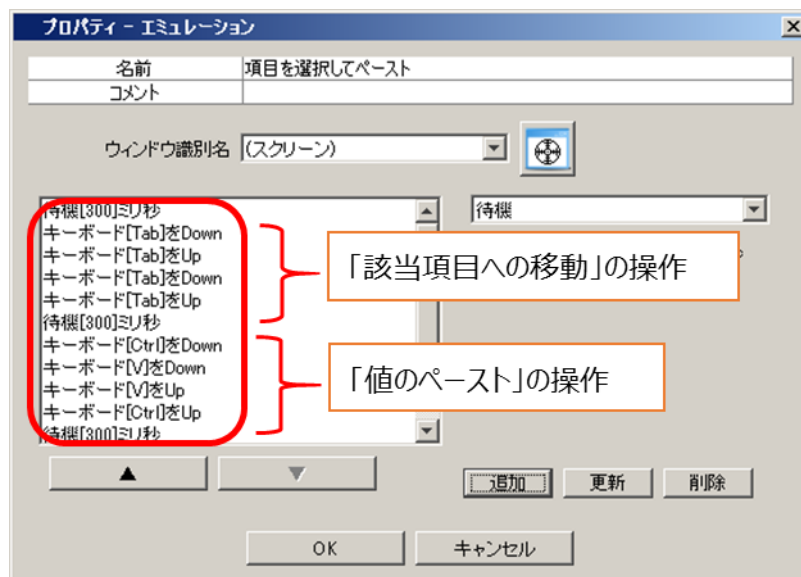


図 5-5 エミュレーション(複数操作)の設定例

### 5.3.3 アプリケーション起動やファイル起動の操作

- 起動操作を実現するノード/ライブラリには、大きく分けて「起動完了を待つ」「起動処理のみを行い、完了は待たない」の 2 つがある。起動対象(ファイル、アプリケーション等)に応じて適切な方法で実現すること

※ 使用アプリケーションによっては、下記のパターンを満たさない場合がある。  
その場合は別のノード/ライブラリを使用すること。

#### 【起動完了を待つノード/ライブラリ例】

- ・ ライブラリ「ファイルと関連づいているアプリ起動」
  - ・ ライブラリ「Excel 開く(前面化)」
- など

#### 【起動完了を待たないノード/ライブラリ例】

- ・ ノード「コマンド実行」
  - ・ ライブラリ「Explorer でファイル開く」
  - ・ ライブラリ「コマンド実行」
  - ・ ライブラリ「URL を指定して IE 起動」
- など

---

シナリオ作成ガイドライン Version 2.00

---

	2019 年 06 月	Version 2.00 発行
発行者	株式会社NTTデータ 社会基盤ソリューション事業本部	
	ソーシャルイノベーション事業部	
	東京都江東区豊洲 3-3-3	
	豊洲センタービル	
	Tel. 050-5546-7720	
	E-mail. WinActor_support@kits.nttdata.co.jp	

---

本書は無断で他に転用しないようお願いします。