

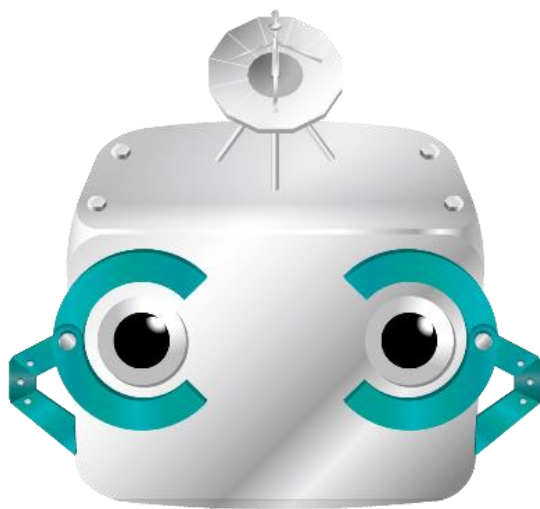
パソコン設定



# SetROBO for Kitting

## 入門ガイド

Ver.2.4



SETROBO

# 目次

I.はじめに	2
II.キッキングの自動化について	2
III.本書にて使用する画面・アイコンの説明	3
IV.事前準備	5
V.注意	7
STEP01 SetROBO for Kittingの起動と終了	8
・SetROBOの起動方法とプロジェクト作成方法、終了方法、プロジェクトの開き方について	
STEP02 指定のフォルダを開く	12
・指定のフォルダを開くコードと、[画面を直接開く][画面が開くまで待つ]について	
STEP03 インターネットオプション ホームページ変更	16
・インターネットオプションにある[ホームページ]のアドレスを変更へ変更するコードと、 [変数] [キー入力][チェックプロパティ] について	
STEP04 音量ミキサーのスピーカーをミュートに設定	22
・音量ミキサーのスピーカーをミュートに設定するコードと、[条件分岐] [存在チェック] について	
STEP05 コンピュータ名の変更① ～操作の記録～	26
・コンピュータ名を変更するコードと、[操作の記録] について	
STEP06 コンピュータ名の変更② ～記録後の編集～	31
・STEP5で記録したコードの[編集] について	
STEP07 CSVファイルからデータを読み込みコンピュータ名の変更を行う	35
・[CSVファイルからデータを読み込む方法]と[入力フォーム] について	
STEP08 画像を比較して操作する方法	38
・[記録ができないコントロールなどを操作する方法] について	
STEP09記録が取れないコントロールの記述方法について	46
・[コントロールの記録が取れない場合の記述方法] について	
STEP10 セットアップ用パッケージの作成	51
・[セットアップ用パッケージ作成方法] について	
付録1 Windows11ショートカットコマンド一覧	55
付録2 Windows10ショートカットコマンド一覧	66
付録3 画像比較ライブラリについて	77
付録4 画像比較ライブラリのメソッド一覧	78
付録5 CSVファイル操作ライブラリについて	81
付録6 共通変数について	82
付録7 ログの保存先の変更方法について	84
【ご案内】SetROBOサポートページをチェック！！	85
変更点	87

# I.はじめに

「SetROBO for Kitting」入門ガイドは、はじめてSetROBO for Kittingを使用する方が、キッティングシナリオの作成に必要な基本の操作や知識を身に付けることを、目的としています。

## ●用語の使い方

本文中では、「Microsoft® Windows® 11」を「Windows 11」、「Microsoft® Windows® 10」を「Windows 10」、「Microsoft® Windows® 7」を「Windows 7」と記述しています。また、本文中で使用している用語は、基本的に実際の画面に表示される名称に則って記述しています。

## ●本書の前提

本書では、「Windows 10」と「SetROBO for Kitting」がインストールされているパソコンを前提に画面を再現しています。その他のOSをお使いの場合、一部画面名やボタン名が異なることもありますが、基本的に同じ要領を進めることができます。

なお、「Windows 11」でも動作可能ですが、一部動作が出来ないSTEPもありますので、その場合は「Windows 10」でご検証ください。

# II.キッティングの自動化について

キッティングの自動化は、人が手動で行うには効率が悪い、もしくは人的ミスの多い箇所を中心に行うことをおすすめしております。もちろん、すべて自動化出来ることが望ましいのですが、実際にすべて自動化を行うことは逆にコストパフォーマンスを下げてしまいます。

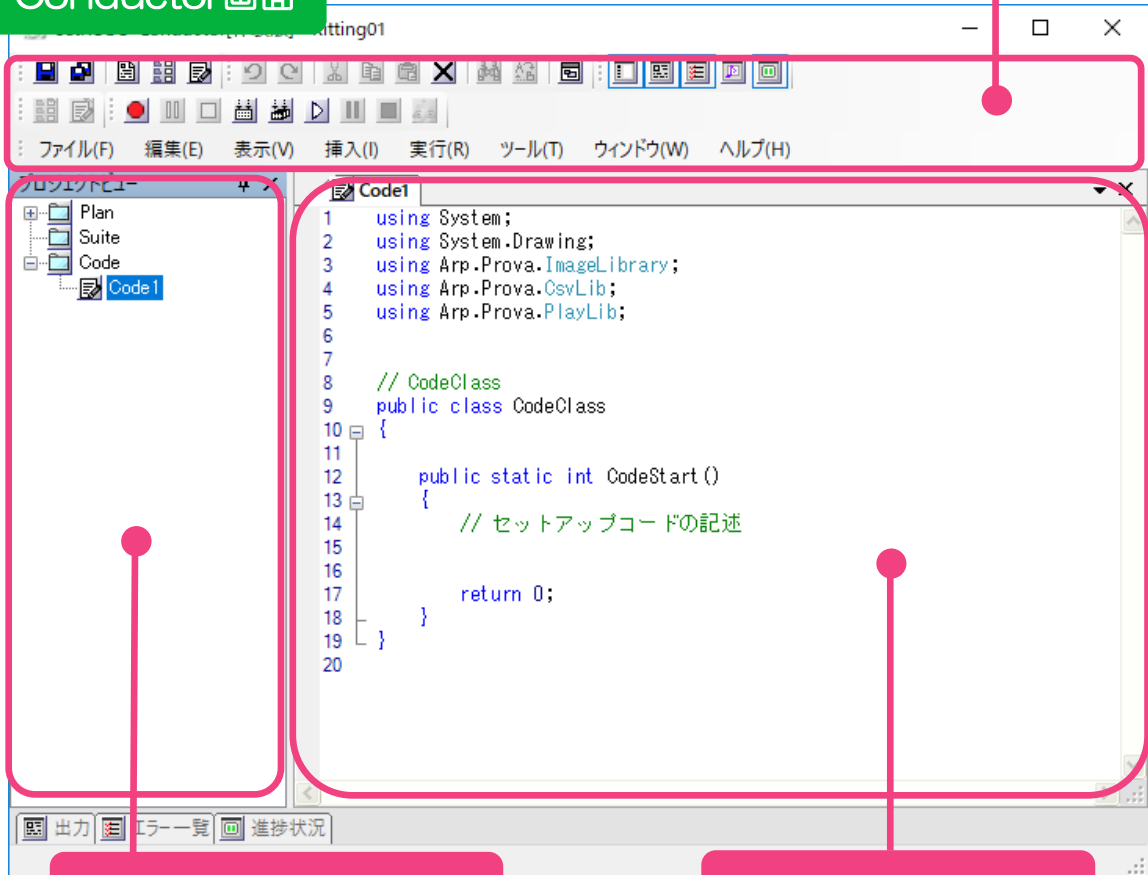
キッティング自動化を行う上で大切なことは、手動で行った方がメリットがある項目と自動化した方がメリットがある項目を切り分けた上で、自動化を実施することです。手動と自動の融合で、コストパフォーマンスを最大限に引き出すことが最も重要です。

## Ⅲ.本書にて使用する画面・アイコンの説明

### 操作メニュー

記録・再生・保存などのメニューがあります。

### Conductor画面



### プロジェクトビュー

プロジェクトのアイテムが表示されます。

「Code」は、キッキングシナリオを作成する項目です。  
キッキング項目ごとに管理できます。

「Plan」は、作成したCodeをまとめてセットアップファイル（実行ファイル）を作成する項目です。

「Suite」は、繰り返し使用する複数の「Code」をまとめることのできる項目です。

### アイテム詳細ビュー

PlanやCode等、プロジェクトビューで選択したアイテムの詳細が表示されます。

「Code」の詳細ビューは、実際にキッキングのシナリオを記述し作成する作業スペースとなります。

## Codeエリア

## Code名

## 閉じるボタン

表示しているCode  
を閉じます。

任意に付けたCode名を  
表示します。

```
1 using System;
2 using System.Drawing;
3 using Arp.Prova.ImageLib;
4 using Arp.Prova.CsvLib;
5 using Arp.Prova.PlayLib;
6
7
8 // CodeClass
9 public class CodeClass
10 {
11
12     public static int CodeStart ()
13     {
14         // セットアップコードの記述
15         |
16         //ごみ箱画面を開く
17         PlayLib.ProcessStart("Explorer",":::{645FF040-5081-101B-9F08-00AA002F954E}","");
18
19         //ごみ箱画面が表示するまで待つ
20         PlayLib.WaitWindow("ごみ箱", true ,80);
21
22         //ごみ箱内の左上を右クリック
23         PlayLib.Window("ごみ箱").Unknown("DirectUIHWND", "DirectUIHWND&3").RightClick(1, 50);
24
25         // 1秒待つ
26         PlayLib.Sleep(1000);
27
28         //ポップアップメニューの"ごみ箱を空にする(B)"をクリック
29         PlayLib.PopupMenu().Click("ごみ箱を空にする(B)");
30
31         // 1秒待つ
32         PlayLib.Sleep(1000);
33
34         //確認ウィンドウで"はい(Y)"ボタンをクリック
35         PlayLib.Window("*/の削除").Button("はい(Y)").Click();
36
37         // 1秒待つ
38         PlayLib.Sleep(1000);
39
40         //ごみ箱画面を閉じる
41         PlayLib.Window("ごみ箱").Close();
42
43         return 0;
44     }
45 }
46
```

## コード行数表示エリア

コードの行数を表示します。

Code内に記述されているコード  
の行数を表示します。

## コードの記述エリア

キッキングのシナリオを記述します。

基本的には「//セットアップコードの記述」  
から「return 0;」の間にキッキングシナ  
リオのコードを記述していきます。  
それ以外の場所に記述すると正しく動作しな  
い場合がありますのでご注意ください。

## アイコン

## コード貼り付けのときのポイント

### ! コード位置修正

・・・コードの位置が崩れるので、[TAB]キーで整えてください。

### ! 改行修正

・・・コードが改行されてしまうと正しく実行できませんので、改行しないよう修正してください。

### ! ¥ 注意

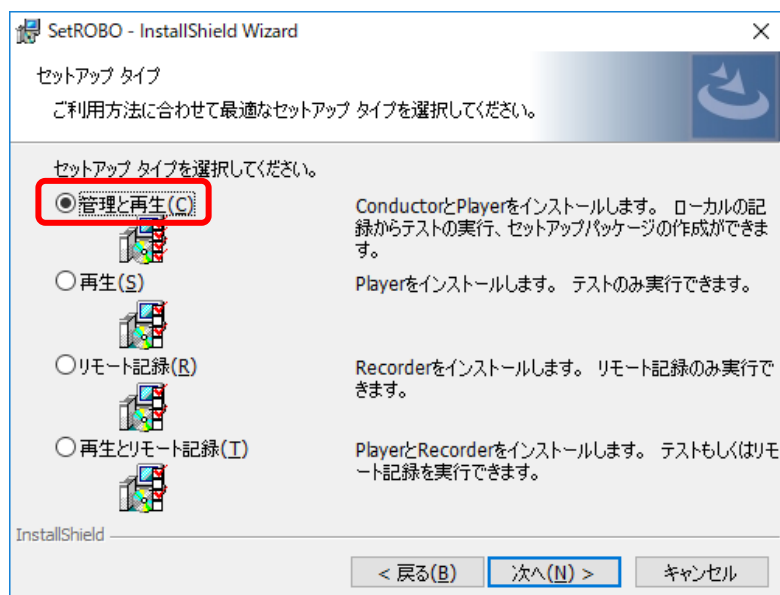
・・・コードをコピーして貼り付けした際に、¥の文字コードが正しくない文字コードに  
変換されてしまうことがあります。  
¥の部分はコード貼り付け後に改めて入力しなおしてください。

## IV. 事前準備

### ① SetROBO for Kittingのインストール

SetROBO for Kittingのインストール手順については、別紙の「SetROBO for Kittingインストールガイド」をご参照ください。

なお、インストール中に表示する「セットアップタイプ」を選択する画面では、必ず「管理と再生(C)」を選択してインストールしてください。

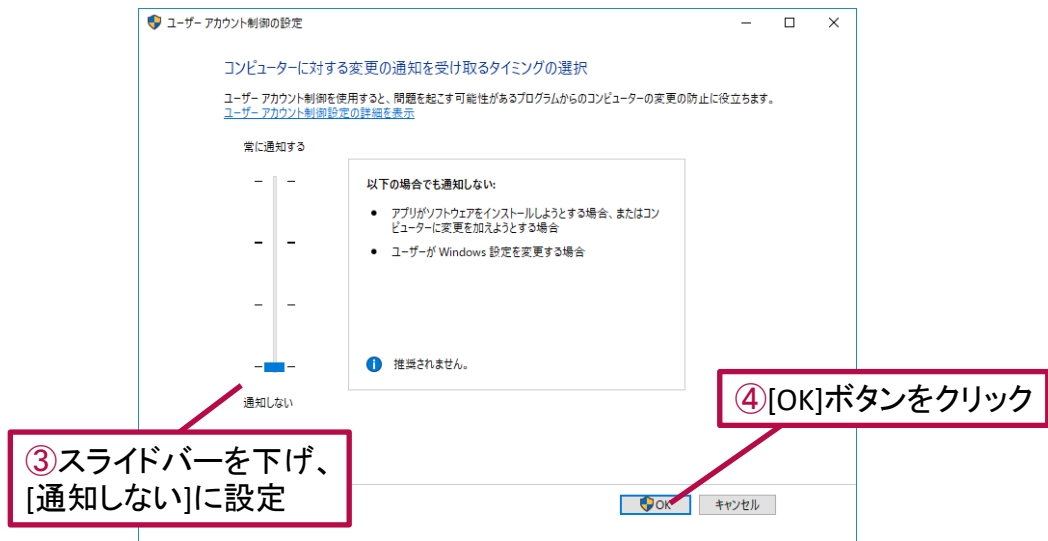


### ② ユーザーアカウント制御 (UAC) を無効に設定

ユーザーアカウント制御を無効にします。

※ユーザーアカウント制御が有効な場合、SetROBO for Kittingが正しく動作しないことがあります。  
なお、この機能は、実行終了時にSetROBO for Kittingで有効に設定することもできます。



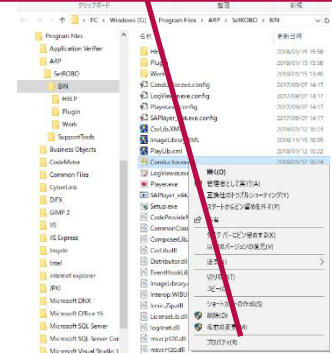


「通知しない」に設定後は、必ず再起動を行ってください。  
**※再起動を行わないと設定が反映されません。**

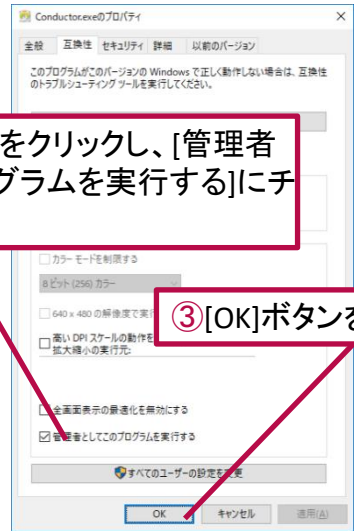
## ③ SetROBO for Kitting Conductorを管理者として実行する設定

Conductorを管理者として実行する設定を行います。  
**※Conductorを管理者として実行しない場合、正しく動作しないことがあります。**

①インストールフォルダー下の  
 BIN¥Conductor.exeを右クリックし  
 [プロパティ]をクリック



②[互換性]タブをクリックし、[管理者としてこのプログラムを実行する]にチェックを入れる



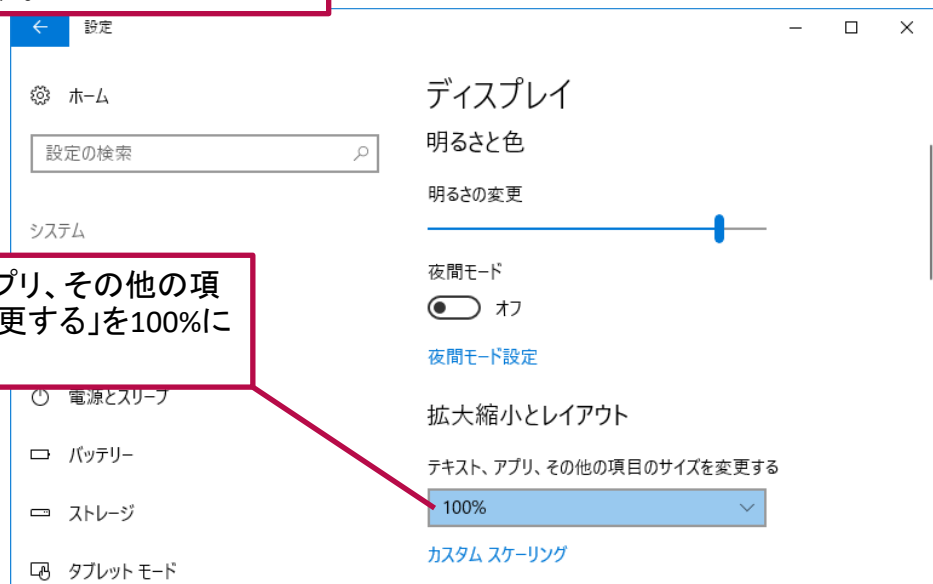
③[OK]ボタンをクリック

## ④ 画面拡大率を100%に設定

Conductorを実行するPC、画像検索に使用する画像を取得するPCの「拡大率」は、100%に設定する必要があります。

(拡大率が100%でない場合、Clickの処理や画像検索が正しく行われません)。以下の手順で拡大率を100%に設定してください。

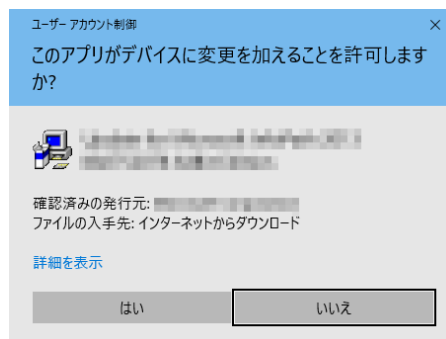
①「設定」の「システム->ディスプレイ」画面を開きます。



②「テキスト、アプリ、その他の項目のサイズを変更する」を100%にします。

## V.注意

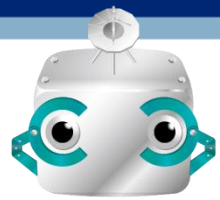
- ①ユーザーアカウント制御(UAC)が表示するメッセージについては、自動操作することができません。  
※UACメッセージ表示中は、全てのアプリケーションが一時停止となります。  
「通知しない」に設定しても表示する場合は、手動で操作を行ってください。



- ②コードを実行するPCのAeroが有効になっている場合、ウィンドウ名・コントロール名・コントロール識別子がサンプルコードと異なる場合や、Conductorの記録時に正しく取得出来ない事があります。ご注意ください。(必要に応じて、Aeroを無効にする、テーマをAero以外から選択する等の対応を行ってください。)

# STEP01

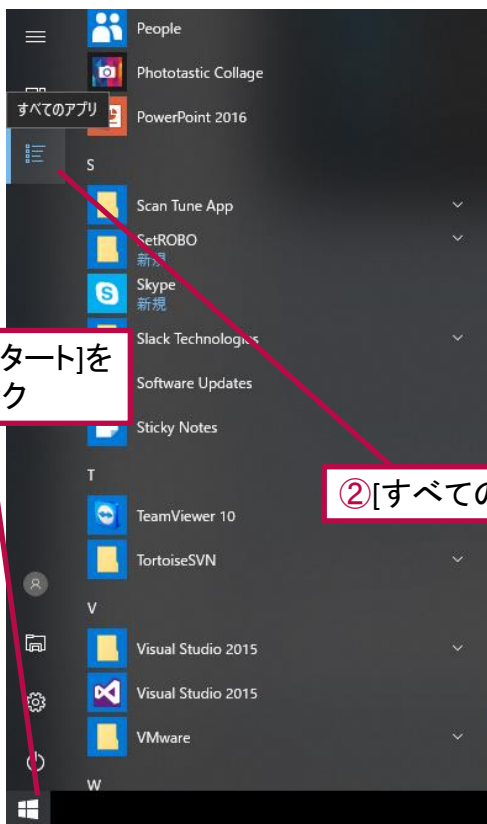
## SetROBO for Kittingの 起動と終了



このSTEPでは、SetROBO for Kittingの起動方法とプロジェクト作成方法、終了方法、プロジェクトの開き方を解説します。

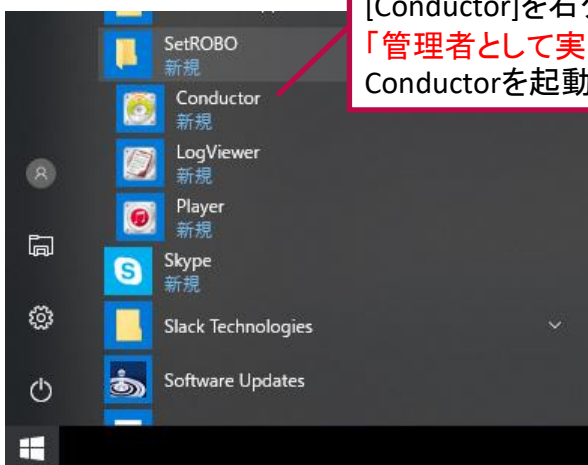
### 1 「スタート」メニューから、SetROBO for Kittingの「Conductor」を起動する

【注意】 Conductor起動の際は、必ず「管理者として実行」から起動してください



①[スタート]をクリック

②[すべてのアプリ]をクリック



③[SetROBO]を開き、[Conductor]を右クリックし、「管理者として実行」からConductorを起動

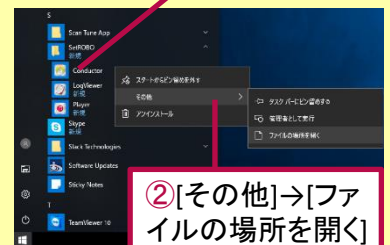
### 👉 プロジェクトとは

「プロジェクト」とは、キittingを自動化する配布用実行ファイル「セットアップ用パッケージ」の生成に必要な複数のファイルをまとめたものです。作成した「プロジェクト」内で、キittingシナリオを作成し、実行形式のファイルを生成することができます。

### 👉 ショートカットアイコンの作成

SetROBOは、「Conductor」を、デスクトップにショートカット作成することで、デスクトップから起動することができます。

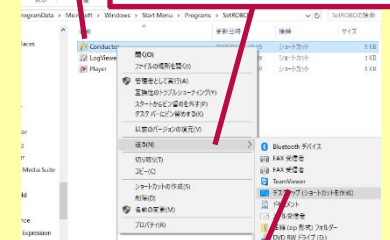
①[Conductor]を右クリック



②[その他]→[ファイルの場所を開く]をクリック

③ [Conductor]を右クリック

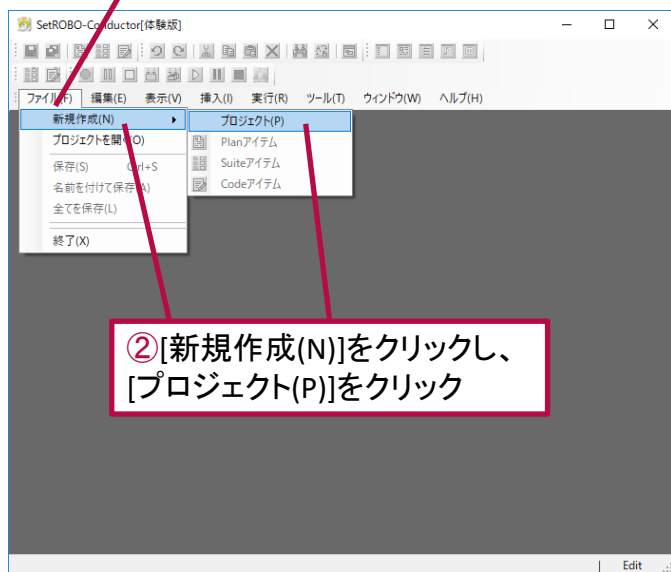
④ [送る(N)]をクリック



⑤ [デスクトップ]をクリック

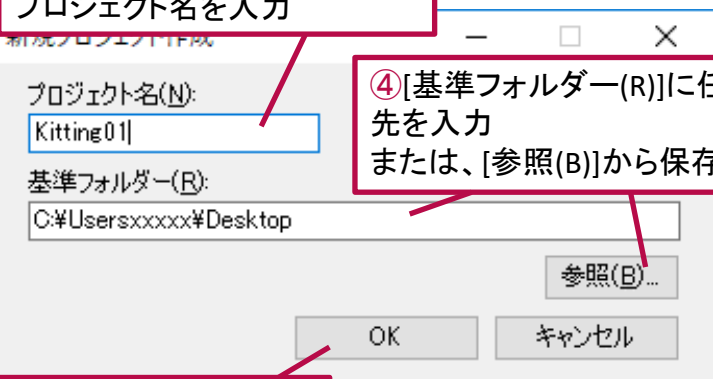
## 2 プロジェクトを新規作成する

① [ファイル(F)]をクリック



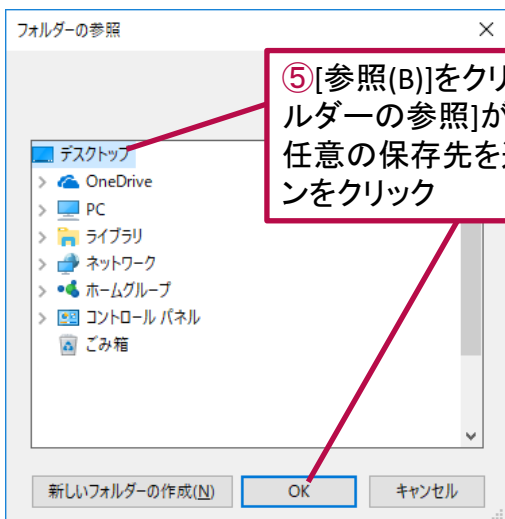
② [新規作成(N)]をクリックし、  
[プロジェクト(P)]をクリック

③ [プロジェクト名(N)]に任意の  
プロジェクト名を入力



④ [基準フォルダー(R)]に任意の保存  
先を入力  
または、[参照(B)]から保存先を選択

⑥ [OK]ボタンをクリック  
するとプロジェクトが作  
成される



⑤ [参照(B)]をクリックすると[フォル  
ダーの参照]が表示される  
任意の保存先を選択し、[OK]ボタ  
ンをクリック

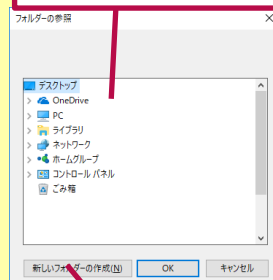
👉 プロジェクト名につ  
いて

「プロジェクト名」の命名規則は特にありません。ただ、実際に行う作業とは関係ない名前を付けてしまうと、あとで分からなくなってしまふ場合があります。プロジェクト名は、実際に行う作業に結びつく名前を付けるようにしましょう。

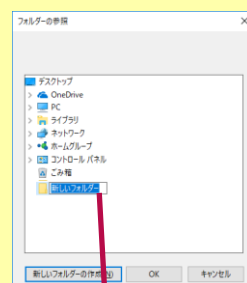
👉 [フォルダーの参照]  
で新しいフォルダー  
を作成するには

保存先を指定する際に、保存先フォルダーを作成していない場合、[フォルダーの参照]から新規フォルダーを作成できます。

① 保存先フォルダーを  
作成する場所を選択



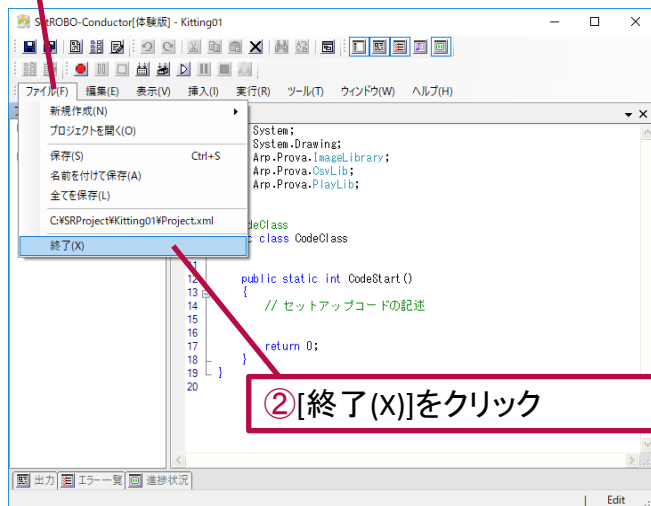
② [新しいフォルダーの  
作成(N)]ボタンをクリック




③ 任意のフォルダー名を  
入力

### 3 SetROBO for Kittingを終了する

①[ファイル(F)]をクリック



②[終了(X)]をクリック

 [保存]と[名前を付けて保存]、[全てを保存]について

SetROBOにある[保存(S)]と[名前を付けて保存(A)]、[全てを保存(L)]は、保存する内容が異なりますので注意してください。

#### 保存(S)

[保存]は、現在[アイテム詳細ビュー]に表示している内容を、保存します。

#### 名前を付けて保存(A)

[名前を付けて保存]は、現在[アイテム詳細ビュー]に表示している内容を別名で保存します。

#### 全てを保存(L)

[全てを保存]は、プロジェクト内の内容をすべて保存します。

 未保存の状態の場合

コードが未保存の場合、[アイテム詳細ビュー]のタブに[\*]が表示されます。

[未保存時]

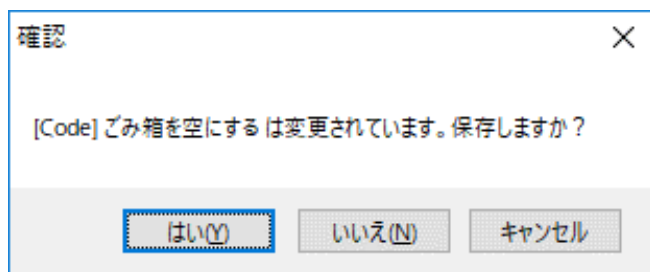
```
code1 *
1 using System;
2 using Arp.Prova.PlayLib;
3
```

[保存時]

```
code1
1 using System;
2 using Arp.Prova.PlayLib;
3
```

### ■プロジェクトが保存されていない場合

プロジェクトが保存されていない場合、保存の有無の確認メッセージが表示されます。

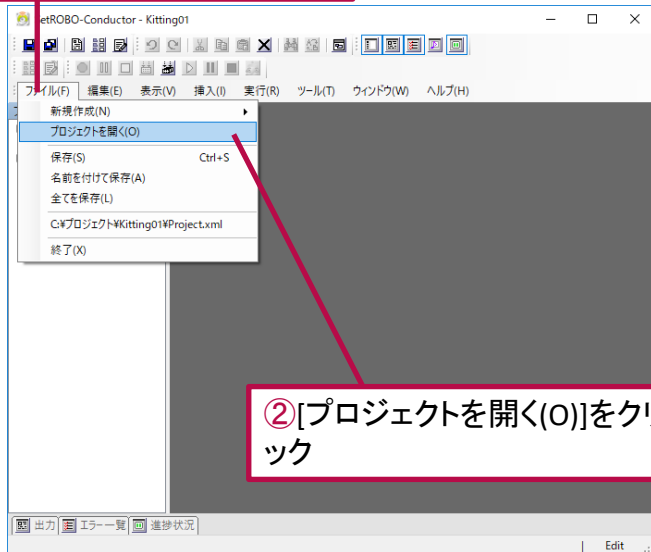


確認メッセージが表示した場合は、状況に合わせてボタンをクリックしましょう。

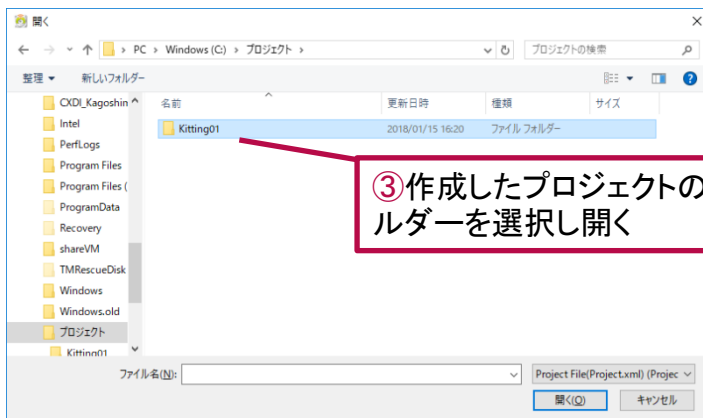
- [はい]をクリックした場合... 変更を保存して終了します。
- [いいえ]をクリックした場合... 変更を保存せず終了します。
- [キャンセル]をクリックした場合... 終了せずに、元の画面に戻ります。

## 4 プロジェクトの開き方

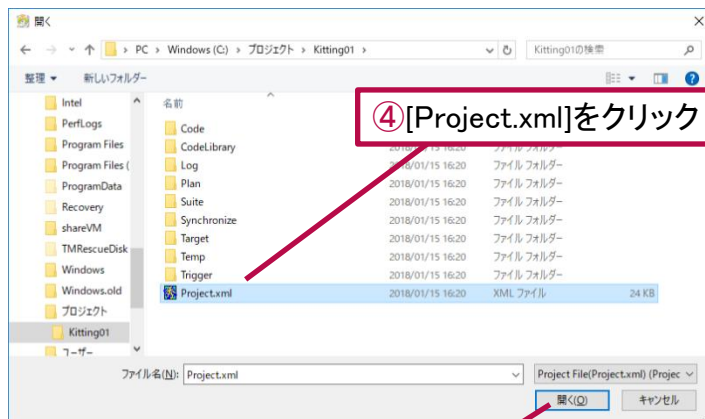
### ① [ファイル(F)]をクリック



### ② [プロジェクトを開く(O)]をクリック



### ③ 作成したプロジェクトのフォルダーを選択し開く



### ④ [Project.xml]をクリック

### ⑤ [開く(O)]をクリックするとプロジェクトが開く

## プロジェクトについて

作成したプロジェクトの内容について、説明します。

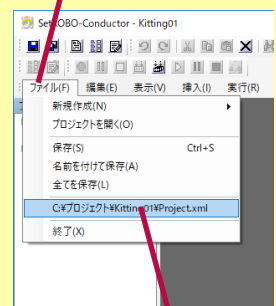
プロジェクトを作成すると、このようなファイルやフォルダー群を出力します。この中に、作成したコードや動作確認した時のログ等を格納していきます。

このファイル群を、Conductorで開く際に必要なのが、左記で選択し開いている[Project.xml]です。

## プロジェクトをより簡単に開く方法

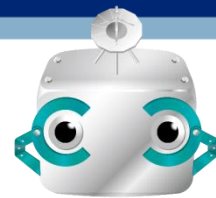
左記のような開き方が基本なのですが、普段使用するプロジェクトの場合、より簡単に開く方法があります。

### ① [ファイル(F)]をクリック



### ② 開きたい[Project.xml]のパスをクリック

作成したプロジェクトのXMLファイルをフルパスで表示します。どこに保存したかを確認し次回プロジェクトを開く際は、簡単に開きましょう。



# STEP02

## 指定のフォルダを開く

それでは、早速キittingシナリオを作成していきましょう。このSTEPでは、指定のフォルダを開くコードを記述するとともに、[画面を直接開く][画面が開くまで待つ]方法について解説します。

**ProcessStart関数を使いましょう**

[ProcessStart]関数は、下記のように直接画面を開いたり、プログラムを実行することができます。画面を開くのに、デスクトップ画面から順に遷移していくシナリオを作成してしまうと、大量のコードの記述が必要となり、実行時のキitting時間も増加します。直接開くことができる画面や、直接実行できるプログラムは、できる限り[ProcessStart]関数を使用しましょう。

**ProcessStart関数の使い方**

[ProcessStart]関数は、まず[PlayLib.ProcessStart]と記述し、その次の括弧の中に実行させたい内容を記述します。最後に[;]を記述します。(コードの最後には、基本的に[;]を記述してください)  
**[PlayLib.ProcessStart(内容);]**

画面を開く際の内容は、画面ごとに決まったコードがあります。

[SetROBO]フォルダを開く場合は、**[["@C:¥Program Files¥ARP¥SetROBO"];]**というコードを記述します。括弧内に記述した内容によってさまざまな画面を開くことができます。

画面ごとの決まったコードについては、本書後半の「ショートカットコマンド一覧」を確認してください。

### 1 Code1を開く

①[Code]の[+]をクリック

②[Code1]をダブルクリック

③[Code1]が[アイテム詳細ビュー]に表示される

### 2 [SetROBO]フォルダを開くコードを記述する

[SetROBO]フォルダを直接開くコード

```
PlayLib.ProcessStart(@"C:¥Program Files¥ARP¥SetROBO");
```

※コード内の文字列で記述する[¥]は、[¥¥]と記述する必要があるが、文字列の前に[@]を記述することによって、[¥]を省略することができます。

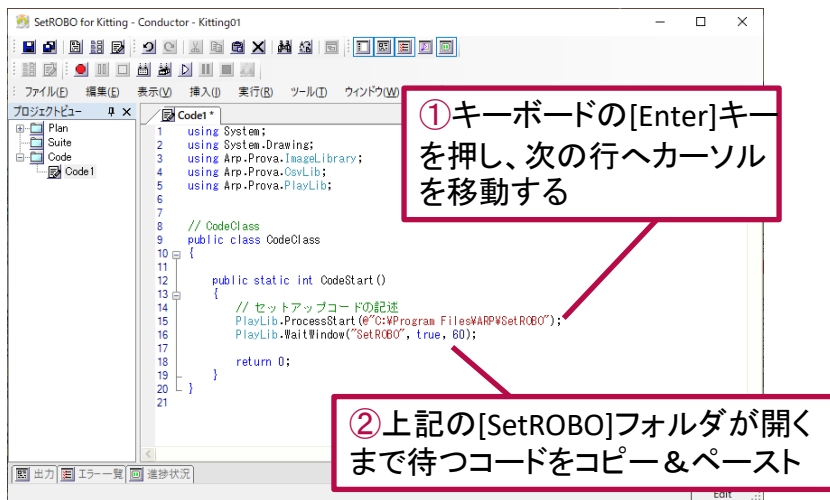
①[// セットアップコードの記述]の下をクリック

②上記の[SetROBO]フォルダを直接開くコードをコピー&ペースト

### 3 ごみ箱画面が開くまで待つコードを記述する

[SetROBO]フォルダが開くまで待つコード

```
PlayLib.WaitWindow("SetROBO", true, 60);
```



### WaitWindow関数について

[WaitWindow]関数は、指定した画面が表示または表示しなくなるまで待つ際に使用します。

[PlayLib.WaitWindow]を記述し、その次の括弧の中に内容を記述します。

**[PlayLib.WaitWindow(内容);]**

指定する内容は主に3つあり、左から順に[画面名]、[表示/非表示]、[待機時間]です。各内容の間には、必ず[, ]で区切って記述します。

1つ目の[画面名]は、対象の画面名を記述します。記述方法は、[" "]で囲みます。例："SetROBO"

2つ目の[表示/非表示]は、対象の画面が[表示するまで待つ]または[表示しなくなるまで待つ]を指定します。指定方法は、表示[true]/非表示[false]です。

3つ目の待機時間は、対象の画面が表示または非表示となるまで何秒待つかを指定します。(秒で指定)  
**[("SetROBO", true, 60);]**

確実に次の動作を行えるので、ぜひ使いましょう。

### Sleepについて

[Sleep]とは、次のコードの実行を、指定した時間待機するコードです。

[PlayLib.Sleep]を記述し、その次の括弧の中に待機する時間を記述します。

時間はmsec(ミリセック 1000msec = 1秒)で記述します。

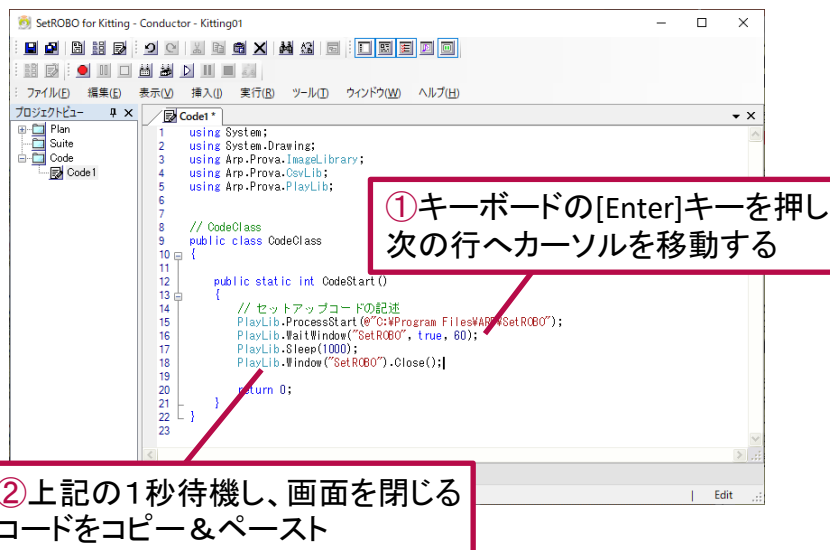
次の動作を行う際に、SetROBOが早く動作してしまい、正しく実行できない場合などに使用します。

### 4 ごみ箱を空にし、画面を閉じるコードを記述する

1秒待機し、画面を閉じるコード

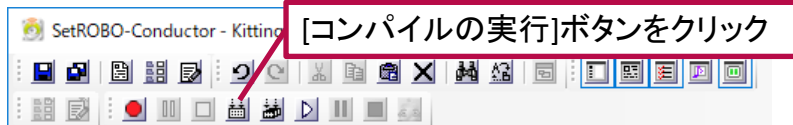
🚨 コードの位置修正

```
PlayLib.Sleep(1000);  
PlayLib.Window("SetROBO").Close();
```



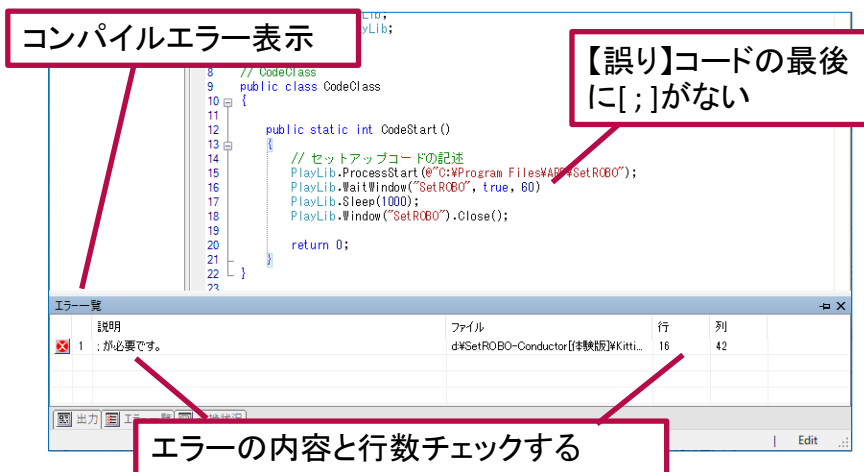
## 5 記述したコードをコンパイルする

コンパイルとは、記述したコードをコンピュータ上で実行可能な形式に変換することです。  
実行可能な形式に変換することで、記述したコードを動作させることができます。



### ■コンパイルした時にエラーが表示された場合

コンパイルを実行し、コードに問題がない場合は、何もメッセージを表示しません。逆に、記述したコードに誤りがある場合は、エラーメッセージを表示します。  
エラーが表示した場合は、エラーの内容を確認し正しく修正しましょう。



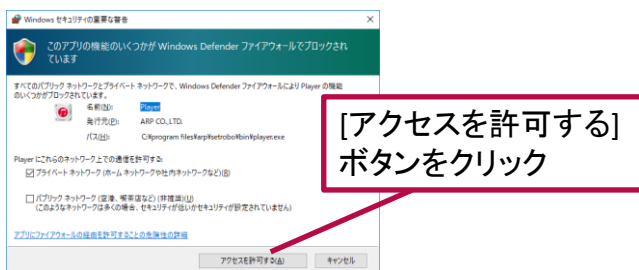
## 6 コードを実行する

作成したコードを実行します。実行すると記述したコードを上から順に動作していきます。

また、進行状況は実行時に起動する[SetROBO-Player]で確認できます。※SetROBO-Playerはタスクバーに表示します。

なお、Windowsファイアウォールの警告が表示した場合は、「アクセスを許可する」ボタンをクリックしてください。

※許可しても失敗した場合は、再度、コードを実行してください。



## コンパイル・実行で自動保存

STEP1の③にて[保存]について説明しましたが、コンパイルを実行すると、実行可能な形式に変換するとともに、自動で保存を行います。  
また、コードの実行を行った場合でも、同様に自動で保存を行います。

## エラー時の確認方法

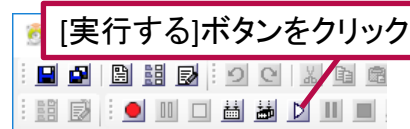
エラーが表示した場合、エラー一覧の[説明]と[行]、[列]を確認しましょう。

まず、[行]を確認し、その行の何列目を[列]で確認します。(列は一番左の列を1行目とします)

行	列
13	38

最後にエラーの原因を[説明]に記載していますので、修正を行いましょう。

説明  
: が必要です。



## 7 コードの内容について

実行したコードの内容です。

### ■ 「SetROBO」フォルダを開く

```
PlayLib.ProcessStart(@"C:\Program Files\ARP\SetROBO");
```

### ■ 「SetROBO」フォルダが表示するまで待つ

```
PlayLib.WaitWindow("SetROBO", true, 60);
```

### ■ 1秒待つ

```
PlayLib.Sleep(1000);
```

[1000msec]の間、待つ

### ■ ごみ箱画面を閉じる

```
PlayLib.Window("SetROBO").Close();
```

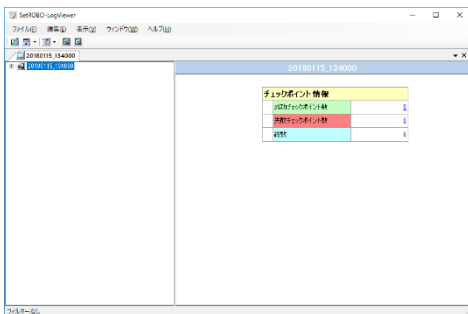
画面:ごみ箱画面

閉じる

## 8 ログを確認する

実行が終了すると、[SetROBO-LogViewer]画面が表示します。実行したログは、[SetROBO-LogViewer]画面に表示します。

なお、ログについてはSTEP3の⑩で説明します。



[×]ボタンをクリックし、閉じる

## コメントについて

コードを記述する際に、記述したコードが何をする処理なのかを分かりやすくするために、コードの上にコメントを入れておきます。コメントの入れ方は、先頭に[//]を入れ、そのあとに内容を書き込みます。

```
// 「SetROBO」フォルダを開く  
PlayLib.ProcessStart(@"C:\Program Files\ARP\SetROBO");
```

コメントを入れないと、あとでコードを見た際に何をしているか分かりにくくなります。

コメントは、できる限り入れるように心がけましょう。

## 正しく実行できない場合について

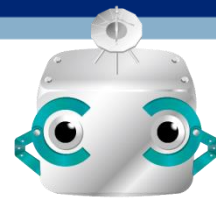
実行時に、正しく動作できない場合があります。

例えば、指定のフォルダが存在しない場合、アプリケーションが見つからないエラーが発生し、フォルダの画面が表示されないため、規定の待機時間表示を待ち、その後エラーとなって終了します。

日時	種別	情報
2022/01/13 12:00:00	状態	再生データの受信中です
2022/01/13 12:00:00	状態	実行開始待ちです
2022/01/13 12:00:00	状態	再生データの実行中です
2022/01/13 12:00:00	動作	ProcessStart(C:\Program Files\ARP\SetROBO)が実行されました。
2022/01/13 12:00:00	エラー	ファイルが見つかりません。【C:\Program Files\ARP\SetROBO】が見つかりません。
2022/01/13 12:00:00	動作	WaitWindow(SetROBO, True, 60)が実行されました。
2022/01/13 12:00:00	動作	Window("SetROBO").Close()が実行されました。
2022/01/13 12:00:00	エラー	ウィンドウ["SetROBO"]が見つかりませんでした。
2022/01/13 12:00:00	状態	待機状態です
2022/01/13 12:00:00	状態	切替されました(相手は 127.0.0.1)

なお、待機時間については、メニューの[ツール]⇒[オプション]から設定できます。

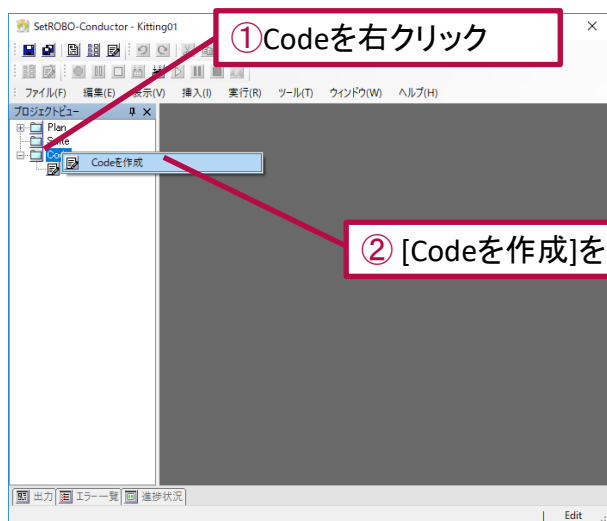
オプション	
[Player]通信設定   Conductor設定   環境情報設定   メール通知設定	
待機時間設定	
ウィンドウ探索待機時間	30 秒
コントロール探索待機時間	30 秒
チェックポイントモード待機時間	10 秒
ボタンクリック時の有効待機時間	30 秒
クリック動作前の待機時間	0 秒



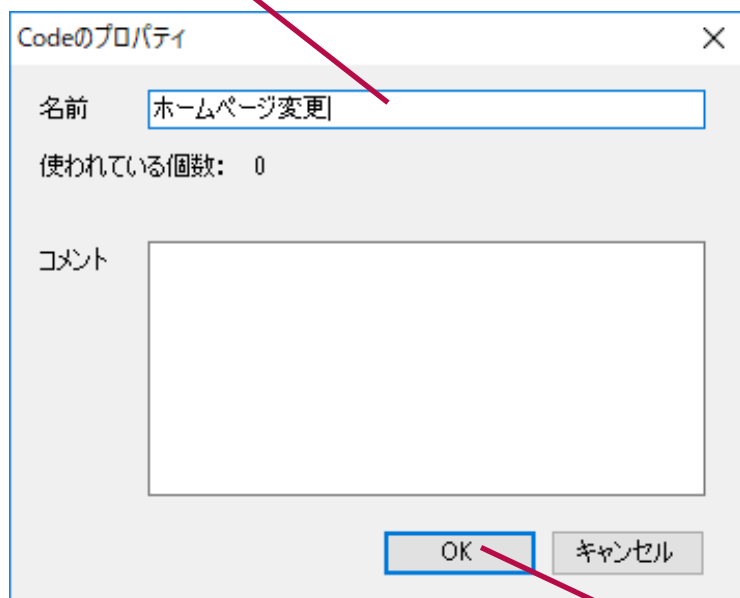
このSTEPでは、インターネットオプションにある[ホームページ]のアドレスを変更するコードを記述するとともに、[変数][キー入力][チェックプロパティ]について解説します。

※Windows 11ではこの設定がありませんので、検証を行う場合はWindows 10でお願いします。

## 1 新規のCodeを作成し、開く



③ [名前]に任意の名前を入力



④ [OK]ボタンをクリック

### Codeの管理について

[Ⅲ. Conductor画面の説明]でも説明しましたが、Codeは、キittingシナリオを作成する項目です。複数のシナリオを作成する際に、1つのCodeに複数のシナリオを記述することもできますが、そうしてしまうと、コード量が増え修正が大変になります。また、[セットアップファイル]を作成する際にも、シナリオによって手順の必要・不必要が発生し、コードの切り出し作業を行うこととなります。

そのようなことを防ぐために、Codeは項目ごとに管理することをオススメします。項目ごとにCodeを作成し管理することによって、Codeの編集も楽になり、[セットアップファイル]もシナリオによって、Codeの追加や削除ができ、色々なシナリオを簡単に作成することができます。

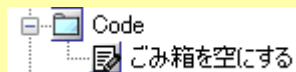
### Code名の変更方法

Code名は、下記の手順で変更できます。

① [Code1]をクリック



② 再度[Code1]をクリックし、Code名を入力



③ [Enter]キーを押す

## 2 インターネットのプロパティを開き、表示を待つ

### 変数について

変数とは、データを入れる「箱」のようなものです。複数回同じ値を記述する場合や取得した値を格納したりと、様々なところで使用できます。基本的な変数の格納について説明します。

数値を格納する場合、まず、「int」と記述します。これは「この箱は整数値用です」と宣言しています。その次に、箱の名前として、任意の文字（英数字）を記述します。箱の名前は、分かりやすい名前をつけましょう。箱の宣言を記述したら、次に、箱の中身の値を記述します。 [= ] を記述し、その後ろに値（数値）と ; ] を記述します。

```
int basicWait = 1000;
```

これで、変数「basicWait」に値(1000)が入った状態になりました。この変数を下記のように使用します。

```
PlayLib.Sleep(basicWait);
```

文字列を格納する場合もほぼ同じです。ただ数値を格納する場合と違う点が2点あります。1点目は、初めに「int」ではなく、「String」と宣言します。「String」は、文字列を格納する場合に使用します。2点目は、文字列を記述する際に「"」で文字列を囲います。

```
String homePage = "http://www.google.co.jp/";
```

これで、変数「homePage」ができましたので、この変数を下記のように使用します。

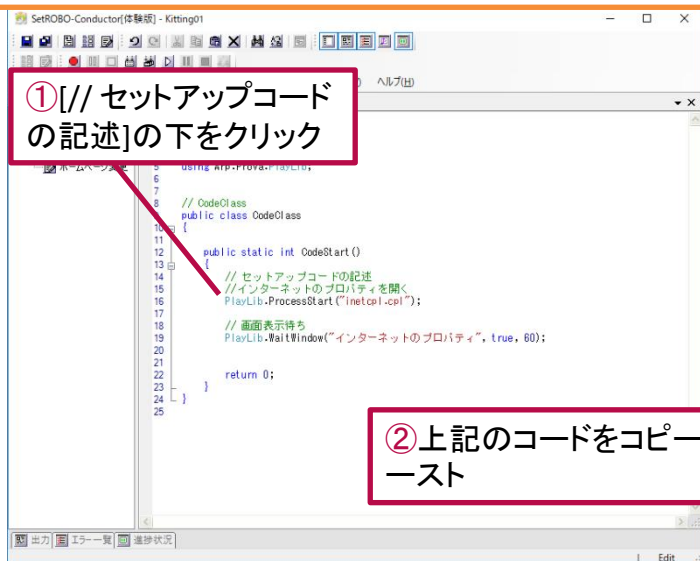
```
PlayLib.Window("画面名").Edit("エディット名").Input(homePage);
```

### インターネットのプロパティを直接開き、画面を待つコード

```
//インターネットのプロパティを開く  
PlayLib.ProcessStart("inetctl.cpl");
```

❗コード位置修正

```
// 画面表示待ち  
PlayLib.WaitWindow("インターネットのプロパティ", true, 60);
```



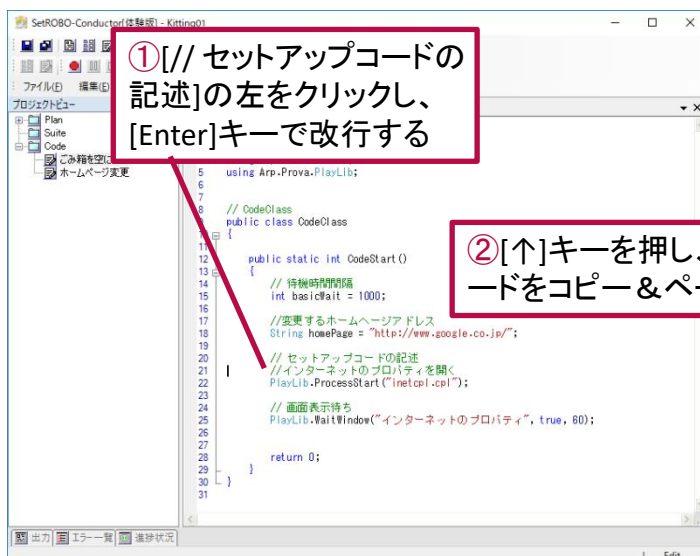
## 3 変数を宣言する

### 待機時間とホームページアドレスの変数のコード

❗コード位置修正

```
// 待機時間間隔  
int basicWait = 1000;
```

```
//変更するホームページアドレス  
String homePage = "http://www.google.co.jp/";
```



左記のコード内に登場した [Input] は、文字を入力する際に使用します。

記述する方法は、[PlayLib.] の後に [Window("画面名")] と [Edit("入力対象の名前")] を記述します。そして最後に、[Input("入力する値")] を記述します。

左記の入力するコードを例にすると、Window名が [Window("インターネットのプロパティ")]、入力対象の名前が [Edit("複数のホーム ページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)))]、入力する値が [Input("http://www.google.co.jp/")] となります。  
 ※入力対象が入力欄の場合は、基本的に [Edit] を使用します。

なお、入力対象の名前は、基本的に入力欄の上に記述されてます。もし分からない場合は、[記録] 機能を使用しましょう。  
 ※記録機能は、STEP05で説明します。

## 4 ホームページ設定の入力と1秒待つ

### ホームページ設定の入力と、1秒待つコード

! 改行修正

```
//ホームページ設定の入力
PlayLib.Window("インターネットのプロパティ").Edit("複数のホーム ページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)").Input(homePage);

//1秒待つ
PlayLib.Sleep(basicWait);
```

! コード位置修正

① [画面表示待ち] コードの下を改行

② 上記のコードをコピー & ペースト

```
using Arp.Prova.PlayLib;

public class CodeClass
{
    public static int CodeS
    {
        // 待機時間間隔
        int basicWait = 1000;

        // 変更するホームページアドレス
        String homePage = "http://www.google.co.jp/";

        // セットアップコードの記述
        // インターネットのプロパティを開く]
        PlayLib.ProcessStart("inetcp1.cpl");

        // 画面表示待ち
        PlayLib.WaitWindow("インターネットのプロパティ", true, 60);

        // ホームページ設定の入力
        PlayLib.Window("インターネットのプロパティ").Edit("複数のホーム ページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)").Input(homePage);

        // 1秒待つ
        PlayLib.Sleep(basicWait);
    }
}
```

// ホームページ設定の入力  
 PlayLib.Window("インターネットのプロパティ").Edit("複数のホーム ページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)").Input(homePage);

## 5 ホームページ設定で入力した値をチェックする

### Sleepとホームページアドレスの変数のコード

! 改行修正

```
//ホームページURLの確認
PlayLib.Window("インターネットのプロパティ").Edit("複数のホーム ページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)").CheckProperty("text", homePage);

//1秒待つ
PlayLib.Sleep(basicWait);
```

! コード位置修正

① [1秒待つ] コードの下を改行

② 上記のコードをコピー & ペースト

```
public static int codeStart()
{
    // 待機時間間隔
    int basicWait = 1000;

    // 変更するホームページアドレス
    String homePage = "http://www.google.co.jp/";

    // セットアップコードの記述
    // インターネットのプロパティを開く]
    PlayLib.ProcessStart("inetcp1.cpl");

    // 画面表示待ち
    PlayLib.WaitWindow("インターネットのプロパティ", true, 60);

    // ホームページ設定の入力
    PlayLib.Window("インターネットのプロパティ").Edit("複数のホーム ページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)").Input(homePage);

    // 1秒待つ
    PlayLib.Sleep(basicWait);

    // ホームページURLの確認
    PlayLib.Window("インターネットのプロパティ").Edit("複数のホーム ページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)").CheckProperty("text", homePage);

    // 1秒待つ
    PlayLib.Sleep(basicWait);
}
```

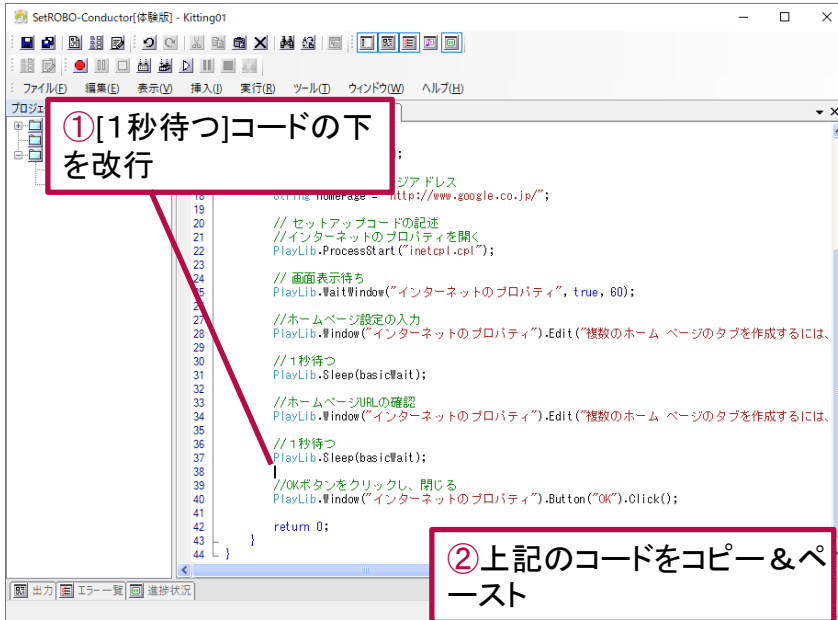
// ホームページURLの確認  
 PlayLib.Window("インターネットのプロパティ").Edit("複数のホーム ページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)").CheckProperty("text", homePage);

## 6 OKボタンをクリックし、閉じる

### OKボタンをクリックするコード

🚨 コード位置修正

```
//OKボタンをクリックし、閉じる  
PlayLib.Window("インターネットのプロパティ").Button("OK").Click();
```



## 👉 CheckProperty関数について

前ページのコード内に登場した[CheckProperty]関数は、指定した箇所の値や状態をチェックする際に使用します。

記述する方法は、[PlayLib.]の後に[Window("画面名")]とチェック対象["チェック対象の名前"]を記述します。そして最後に、[CheckProperty("対象プロパティ", "期待値");]を記述します。

チェックできる対象プロパティは下記の内容となります。

**Text :**

テキスト表示されている文字列の確認

**Enabled :**

ボタン等の状態が有効か無効かを確認

**Visible :**

対象のものが可視状態であるかどうかどうか確認

**Checked :**

チェックボックス等の状態を確認

**ItemCount :**

対象アイテムの数を確認

各プロパティの期待値は、下記の内容となります。

**Text : 実際の値**

**Enabled : true または false**

**Visible : true または false**

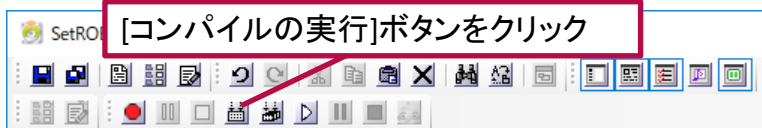
**Checked : 0 または 1**

**ItemCount : 実際の値**

[true]は、正しい場合に使用し、[false]は、誤りの場合に使用し、[0]はチェックOFF状態、[1]はチェックON状態に使用します。

trueまたはfalseを記述する場合、[" "]で囲わないでください。

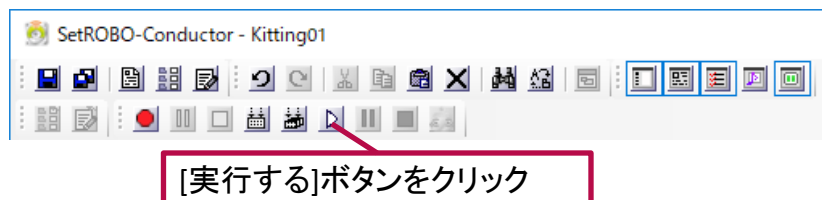
## 7 記述したコードをコンパイルする



もし、コンパイルでエラーが表示した場合は、STEP2の⑤を確認ください。

## 8 コードを実行する

【注意】コードを実行すると、ホームページ初期表示がGoogleに変更されますのでご注意ください。



## 9 コードの内容について

実行したコードの内容です。

### ■ 待機時間間隔

```
int basicWait = 1000;
```

数値[1000]を変数basicWaitの中に入れる

整数値型

変数名

### ■ 変更するホームページアドレス

```
String homePage = "http://www.google.co.jp/";
```

文字列型

変数名

URLを変数homePageの中に入れる

### ■ インターネットのプロパティを開く

```
PlayLib.ProcessStart("inetctl.cpl");
```

### ■ 画面表示待ち

```
PlayLib.WaitWindow("インターネットのプロパティ", true, 60);
```

画面: インターネットのプロパティ画面

対象: 「複数の...」入力欄

### ■ ホームページ設定の入力

```
PlayLib.Window("インターネットのプロパティ").Edit("複数のホームページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)").Input(homePage);
```

上部で宣言した変数の値を入力する

### ■ 1秒待つ

```
PlayLib.Sleep(basicWait);
```

上部で宣言した変数の値の時間待つ

### ■ ホームページURLの確認

```
PlayLib.Window("インターネットのプロパティ").Edit("複数のホームページのタブを作成するには、それぞれのアドレスを行で分けて入力してください(R)").CheckProperty("text", homePage);
```

入力されているテキストが、上部で宣言した変数の値と同じかチェックする

### ■ 1秒待つ

```
PlayLib.Sleep(basicWait);
```

### ■ OKボタンをクリックし、閉じる

```
PlayLib.Window("インターネットのプロパティ").Button("OK").Click();
```

クリックする

画面: インターネットのプロパティ画面

対象: 「OK」ボタン

## Clickについて

[Click]とは、メニューやボタン、チェックボックス等、何かをクリックする際に使用します。

[Click]の次の[()]は、ボタン等の対象物内の座標を指定できます。座標が空の場合は、対象物の中央をクリックし、指定されている場合は、指定の場所をクリックします。指定方法は、[Click(X座標, Y座標)]とカンマ区切りで記述します。

```
PlayLib.Window("〇〇画面").Button("OK").Click(12, 5);
```

通常のクリック以外にもいろいろな種類があります。

### DbClick : ダブルクリック

(例)入力欄の内容をダブルクリックする

```
PlayLib.Window("〇〇画面").Edit("Edit").DbClick(12, 5);
```

### RightClick : 右クリック

(例)入力欄を右クリックし、ポップアップメニュー「貼り付け(P)」をクリックする

```
PlayLib.Window("〇〇画面").Edit("Edit").RightClick(118, 70);  
PlayLib.PopupMenu().Click("貼り付け(P)");
```

また、メニュー内の操作の際は、下記のように[->]を使用して記述します。

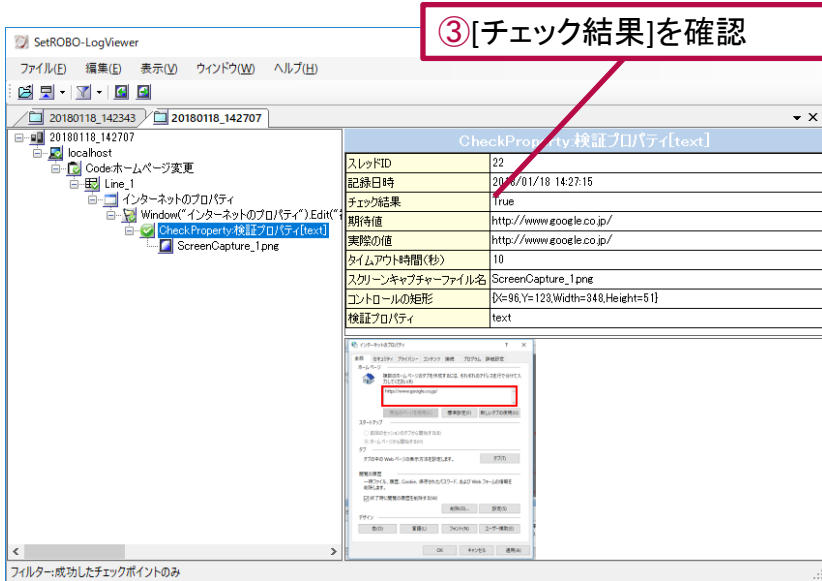
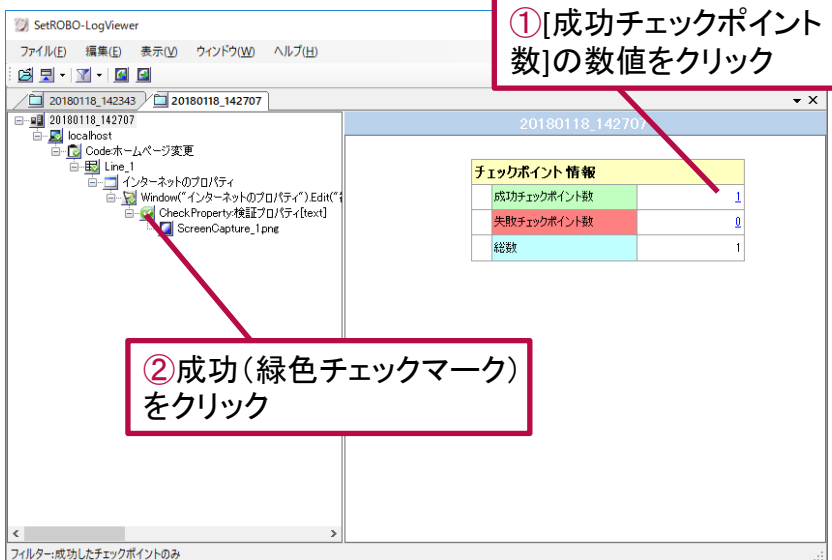
```
PlayLib.Window("〇〇画面").Menu().Click("ファイル(F)->開く(O)");
```

## 10 ログを確認する

[SetROBO-LogViewer]画面にて、CheckPropertyで指定した内容のチェック結果を、チェックポイント情報に表示します。

成功チェックポイント数：期待通りのチェック結果となった数  
失敗チェックポイント数：期待と異なるチェック結果となった数

☑: 成功マーク    ❌: 失敗マーク



チェック結果は、[True]と[False]という結果を表示します。

True：成功    False：失敗

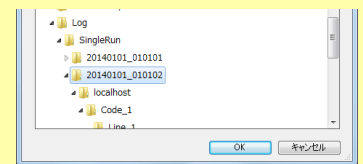
失敗の場合は、[チェック結果]の下にある[期待値]と[実際の値]が異なる結果となった場合です。また失敗の場合は、自動でスクリーンキャプチャーを取得し表示しますので、こちらも確認してください。

※初期設定は、失敗時のみスクリーンキャプチャーを取得します。

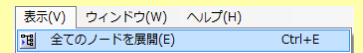
## SetROBO-LogViewerについて

実行結果のログは、作成したプロジェクトフォルダー内の[Log]⇒[SingleRun]フォルダー内に日時別で格納されます。

格納されているログを確認する場合は、[SetROBO-LogViewer]を開き、下記の手順でログを開きます。[ファイル]⇒「ログを開く(O)」⇒日時別のログフォルダーを選択し、[OK]ボタンをクリック

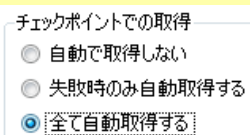


ログの内容を確認する際、左記ではチェックポイント情報から詳細を確認しましたが、実行した内容すべてのログを確認する場合は、メニュー[表示]の[全てのノードを展開(E)]をクリックしてください。



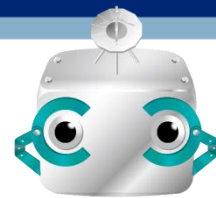
※一度[チェックポイント情報]からツリービューを展開した場合、フィルターがかかりすべて表示できません。その場合は、下記の手順でフィルターのクリアを行ってください。  
メニュー[表示]⇒[フィルターのクリア]をクリック

また、エビデンスとして、チェックした際にスクリーンキャプチャーを取得する設定を変更する場合は、Conductorにて、下記の手順で設定を変更して下さい。  
メニュー[ツール]⇒[オプション]⇒[チェックポイントでの取得]⇒任意の設定に変更⇒[OK]をクリック



# STEP04

## 音量ミキサーのスピーカーをミュートに設定



このSTEPでは、音量ミキサーのスピーカーをミュートに設定するコードを記述するとともに、[条件分岐] [存在チェック] について解説します。

### 体験版の制限について

体験版では以下の制限があります。

①1つのプロジェクトにつき、Codeは3つまで作成可能

②1つのCodeは100行まで

③実行可能形式ファイル (Setup package) 作成不可

このSTEPで、新規にCodeを作成すると、プロジェクト内のCodeが3つになります。体験版では同じプロジェクト内に、これ以上Codeは作成できません。

従って、次のSTEPでは、新しいプロジェクトを作成して、そのプロジェクト内にCodeを作成してください。

### 画面名のワイルドカードについて

このSTEPで記述している画面名[音量ミキサー\*]の\*は、ワイルドカードという特殊文字です。

[\*]は、「任意の文字」を意味します。**Window("音量ミキサー\*")**と記述した場合は、「音量ミキサー」のあとの文字は何でもよいことになります。

音量ミキサー画面は、画面名がサウンドドライバの種類によって変わる為、画面名が固定ではありません。そういった場合に使用することで、さまざまなPC環境でこのコードを使用することができます。ぜひ、活用しましょう。

## 1 新規のCodeを作成し、開く

①Codeを右クリック

② [Codeを作成]をクリック

Codeのプロパティ

名前 音量ミキサー-OFF

使われている回数: 0

③ [名前]に任意の名前を入力

④ [OK]ボタンをクリック

## 2 デバイス名を確認

PC毎にスピーカーの名前が異なる場合がありますので、事前に音量ミキサー画面を開きデバイス名を確認します。デバイス名は後程使用しますので、半角スペースなどを含む正確なデバイス名をメモしておいてください。

※半角スペースなど、正確な名前をメモしてください。

音量ミキサー - Realtek Digital Output (Realtek High Definition Audio)

デバイス(D) アプリケーション

Realtek Digital Output

① デバイス名を確認

### 3 音量ミキサー画面を開き、表示を待つ

音量ミキサー画面を直接開き、画面を待つコード

❗コード位置修正

```
//音量ミキサー画面を開く
PlayLib.ProcessStart("sndvol");

// 画面表示待ち
PlayLib.WaitWindow("音量ミキサー*", true, 30);
```

### 👉条件分岐について

条件分岐とは、ある条件の場合にAの処理を、別の条件の場合にBの処理を行うための分岐です。  
今回は、if文という制御文を使用して条件を分岐します。

音量ミキサーでは、デバイスの下にあるボタンが、ミュートの時とミュート解除の時で異なるボタン名となります。

🔊 **ミュート{デバイス名}**  
🔊 **ミュート解除{デバイス名}**  
※{デバイス名}については、**本STEPの②をご確認ください。**

今回は、音量ミキサーをミュートに設定するため、ミュート解除の時に表示する「ミュート {デバイス名}」ボタンをクリックしてミュートにします。従って、以下の動きをするようにコードを記述します。

「もしミュート {デバイス名} ボタンが存在したらクリックする。」  
※ミュート {デバイス名} ボタンが存在しない場合は何もしない。

本STEPのif文は、以下のように記述します。

```
if(条件A == 条件B)
{
    条件を満たした場合の処理
}
先ほどの内容をあてはめてコードにすると、
if(PlayLib.Window("音量ミキサー").WaitUIControl("UIButton", "ミュート {デバイス名}", true, 1) == true)
{
    ミュート{デバイス名}クリック
}
となります。
```

左右の条件が一致した場合(==)、{}内の処理を行い、条件が一致しない場合は、{}内に入らず、次の動作を行います。条件については、次ページで説明します。

### 4 if文で条件分岐を行う

if文を使用し、条件分岐を行います。  
今回は、デバイスのスピーカーボタンが[ミュート {デバイス名}]となっている場合のみ、ボタンをクリックします。

※{デバイス名}は②で確認したデバイス名を記述してください。

if文で条件分岐を行うコード

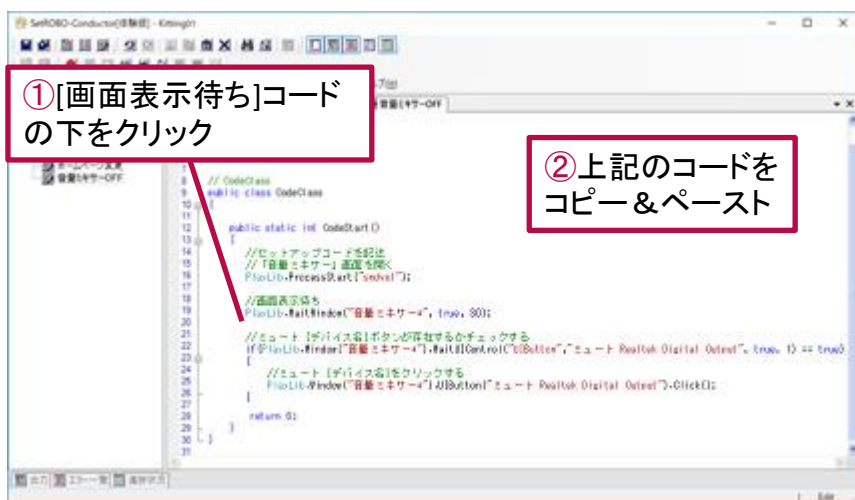
❗コード位置修正

❗改行修正

```
//ミュート{デバイス名}ボタンが存在するかチェックする
if(PlayLib.Window("音量ミキサー").WaitUIControl("UIButton", "ミュート {デバイス名}", true, 1) == true)
{
    //ミュート {デバイス名}をクリックする
    PlayLib.Window("音量ミキサー").UIButton("ミュート {デバイス名}").Click();
}
```

①[画面表示待ち]コードの下をクリック

②上記のコードをコピー&ペースト



## WaitUIControl関数について

if文の条件内で記述している[WaitUIControl]関数について説明します。

この関数は、本来「コントロールが表示するまで待つ」際に使用します。例えば、「インストール中、[次へ]ボタンが表示するまで待つ」場合や「認証後、チェックボックスが表示するまで待つ」場合などに使用できます。

[WaitUIControl]は、下記のように記述します。

**PlayLib.Window("画面名").WaitUIControl("コントロールの種類", "コントロール名", 期待値, タイムアウト時間);**

期待値は、[true]または[false]を指定します。  
**true** : コントロールが存在することを確認する

**false** : コントロールが存在しないことを確認する

本STEPでは、if文の中の条件部分で[WaitUIControl]を使用しています。

```
if(PlayLib.Window("音量ミキサー").WaitUIControl("UIButton", "ミュート{デバイス名}", true, 1) == true)
```

```
{  
    ミュート{デバイス名}クリック処理  
}
```

「音量スピーカー画面の[ミュート {デバイス名}]ボタンが存在しているか」を確認し、その結果ボタンが存在すれば「true」、ボタンが存在しなければ「false」となります。

if文内の「==」の条件は、左側が「true」になり、右側の「true」と一致するので、{}内の処理を行います。

このように、この関数は「条件分岐」の際にも使用することができます。

## 5 ミュート解除 {デバイス名}の存在をチェックする

ミュート解除 {デバイス名}が存在するかチェックするコード **! 改行修正**

```
//ミュート解除 {デバイス名}が存在するかチェックする  
PlayLib.Window("音量ミキサー*").CheckUIControlExist("UIButton", "ミュート解除 {デバイス名}", true);
```

**! コード位置修正**

① [ ]の下をクリックし改行

② 上記のコードをコピー&ペースト

## 6 音量ミキサー画面を閉じる

音量ミキサー画面を閉じるコード

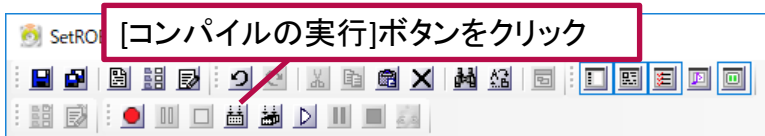
**! コード位置修正**

```
//音量ミキサー画面を閉じる  
PlayLib.Window("音量ミキサー*").Close();
```

① [ミュート解除 {デバイス名}が存在するかチェックする]コードの下をクリックし、改行

② 上記のコードをコピー&ペースト

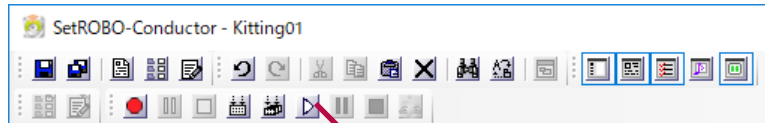
## 7 記述したコードをコンパイルする



もし、コンパイルでエラーが表示した場合は、STEP02の⑤を確認してください。

## 8 コードを実行する

**【注意】** コードを実行すると、音量スピーカーをミュート設定に変更されますのでご注意ください。  
なお、オーディオデバイスがインストールされていない環境では、正しく動作いたしません。



## 9 コードの内容について

実行したコードの内容です。

### ■音量ミキサー画面を開く

```
PlayLib.ProcessStart("sndvol");
```

### ■画面表示待ち

```
PlayLib.WaitWindow("音量ミキサー*", true, 30);
```

### if文で条件分岐

画面:「音量ミキサー\*」

「ミュート {デバイス名} ボタン」が存在することを確認

### ■ミュート {デバイス名} ボタンが存在するかチェックする

```
if(PlayLib.Window("音量ミキサー*").WaitUIControl("UIButton", "ミュート {デバイス名}", true, 1) == true)
```

左側の条件と右側の条件がイコールの場合 { } 中の処理をする

```
{
    ■ミュート {デバイス名} をクリックする
    PlayLib.Window("音量ミキサー*").UIButton("ミュート {デバイス名}").Click();
}
```

クリック

対象:「ミュート {デバイス名}」ボタン

### ■ミュート解除 {デバイス名} が存在するかチェックする

```
PlayLib.Window("音量ミキサー*").CheckUIControlExist("UIButton", "ミュート解除 {デバイス名}", true);
```

「ミュート解除 {デバイス名}」ボタンが存在することを確認

### ■音量ミキサー画面を閉じる

```
PlayLib.Window("音量ミキサー*").Close();
```

閉じる

## WaitUIControl関数とWaitControl関数について

前ページで[WaitUIControl]関数について説明しましたが、実はもう一つ同じ機能を持つ[WaitControl]という関数があります。

この2つの関数の違いは、対象コントロールの種類の違いとなります。

SetROBOには記録機能があり、記録を行って取得できるコントロールは、

[WaitControl] 関数のような「UI」が付いていない名前の関数が使用できます。

逆に、記録を行ったがコントロール名が取得できない場合（[Unknown]と表示する場合）は、

[WaitUIControl] 関数のような「UI」が名前についているコントロールを使用することができます。

なお、記録機能については次のステップで説明します。

## CheckUIControlExist関数について

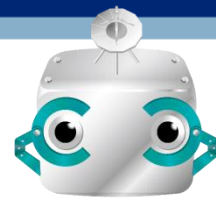
[CheckUIControlExist]関数は、コントロールが存在するかをチェックします。記述方法は、下記のとおりです。

**PlayLib.Window("画面名").CheckUIControlExist("コントロールの種類", "コントロール名", 期待値);**

期待値は、[true]または[false]で指定します。

**true** : コントロールが存在することを確認する

**false** : コントロールが存在しないことを確認する



このSTEPでは、コンピュータ名を変更するコードを記述するとともに、[操作の記録] について解説します。

### 1 プロジェクトを新規作成する

SetROBO for Kitting体験版では、Codeが3つまでしか作成できない為、新しくプロジェクトを作成します。作成方法は、[STEP01]の[②プロジェクトを新規作成する]を参照ください。

### 2 Code1の名前を変更し開く

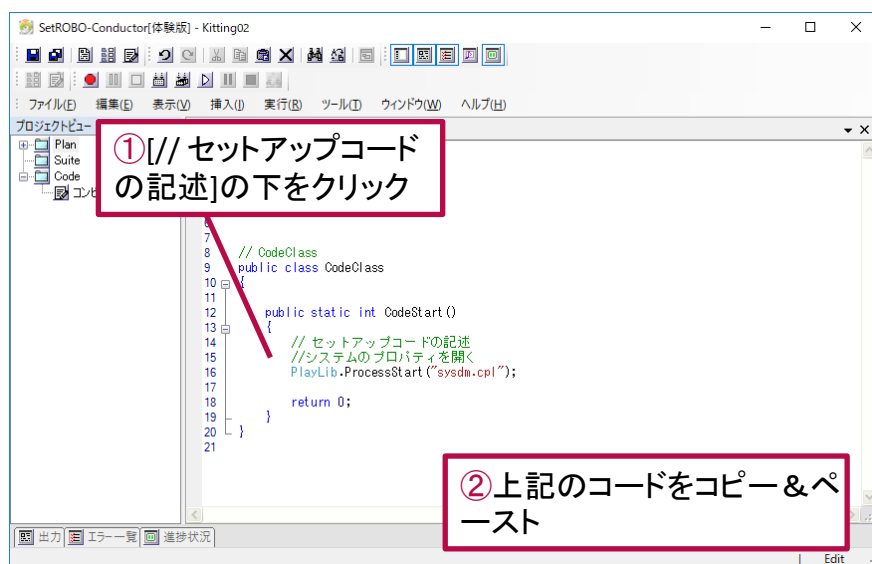
Code1の名前を「コンピュータ名の変更」に変更します。変更方法は、[STEP03]の[①新規のCodeを作成し、開く]を参照ください。

### 3 システムのプロパティを開くコードを記述する

システムのプロパティを直接開くコード

📌 コード位置修正

```
//システムのプロパティを開く
PlayLib.ProcessStart("sysdm.cpl");
```



### 👉 コードの記録について

SetROBOでは、操作した内容を記録しコード化することができます。キittingシナリオを一からコード記述するのは大変です。その為、[ProcessStart]関数で目的の画面近くまで表示した後から記録機能を使用し、キittingシナリオのベースを作ります。

記録の仕方は、[記録を開始する]ボタンをクリックすると[記録]モードとなり記録が始まります。記録を停止する際は、[記録を終了する]ボタンをクリックするだけで、非常に簡単です。

ただし、操作の記録は、行った操作すべてを記録しますので、手順と違う操作を行ってしまった場合でも記録されます。

また、クリック時にマウスが動いてしまうと、クリックではなく、ドラック&ドロップと記録されることもあります。

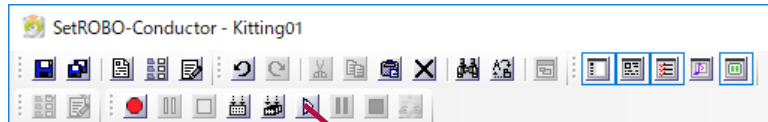
その為、コードの記録時は慎重に操作を行い、もし誤って記録した箇所については、記録停止後に修正を行ってください。

記録機能はあくまでも補助的な機能で、この機能だけでシナリオ作成はできませんが、全体的なシナリオ作成のベース作りには非常に有効な機能です。

ぜひご活用ください。

## 4 コードを実行する

コードを実行すると、「システムのプロパティ」画面が表示されます。

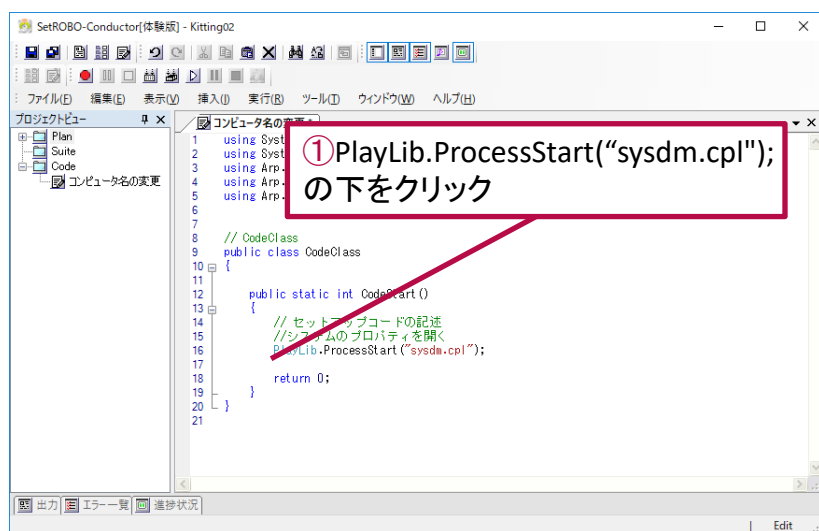


②[実行する]ボタンをクリック

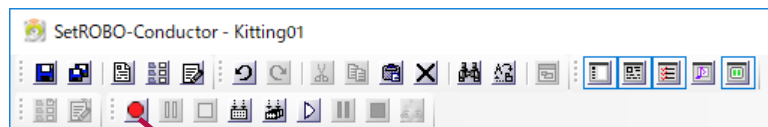
## 5 [記録を開始する]ボタンをクリック

表示した「システムのプロパティ」画面内を操作し、コードの記録を行います。

コードを記述する位置を指定し、記録を開始します。



①PlayLib.ProcessStart("sysdm.cpl");  
の下をクリック



②[記録を開始する]ボタンをクリック



③[記録を開始する]ボタンをクリックすると[Record]モードになります

## 記録のポイント

前ページでも触れましたが、記録機能はすべての操作を記録しますので、余計な操作まで記録してしまいます。しかし、記録機能を使用する際に、いくつかのポイントを押えておくと、比較ミスなく、且つ早くシナリオ作成を行うことができます。

記録を行う際のポイントは下記の通りです。

①ゆっくり操作して、キッキングシナリオ以外の手順は行わない

②クリックする際は、マウスを動かさないようクリックする

③正しく記録が出来ない、または誤って記録した箇所は、削除後に再度その箇所だけ記録を取り直す

この3点を押えて記録すれば、比較的簡単にコードが作成できます。

慌てず、確実な操作でミスをおさず記録しましょう。

## 記録が途中で止まった場合について

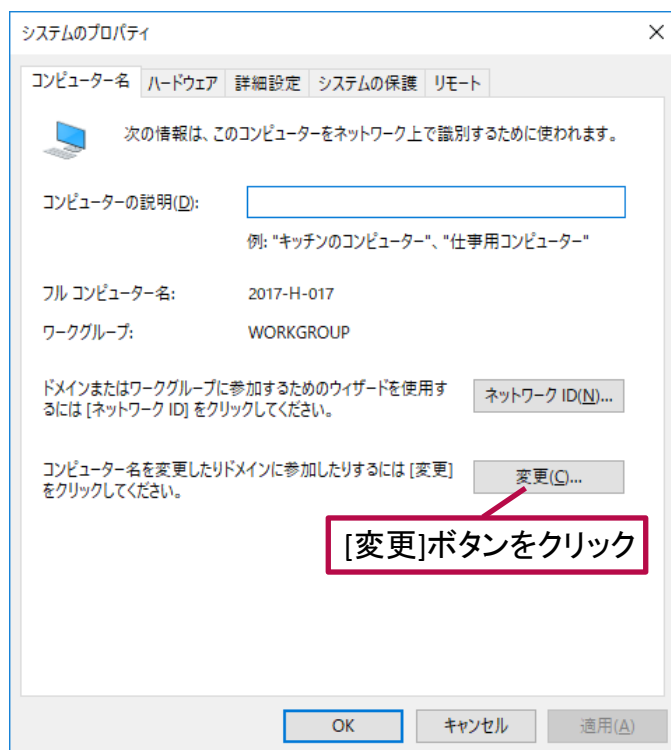
非常に稀ですが、記録操作を行っている際に、記録が取れなくなる場合があります。

その際、一度停止し、再度記録状態にして操作を行うと記録が取れるようになります。

記録を行う際は、きちんとコードが記述されているかConductorを確認しながら行いましょう。

## 6

## [変更]ボタンをクリック



## 文字入力について

キッキングでは、キーボードで文字を入力することがあると思います。SetROBOの記録機能は、キーボードで文字を入力する操作も、基本的にコードの記録を行うことができます。たとえば、「ABC」と入力すると下記のような形で記録します。

```
PlayLib.Window("〇〇").Edit("〇〇").Input("ABC");
```

また、本STEPで行うコンピュータ名の変更では、[Ctrl]+[A]キーで全選択をし、[Delete]キーを使用して削除します。そのようなショートカットコマンドにも対応しており、記録を行うと下記のような形で記録します。

```
PlayLib.Window("コンピュータ名/ドメイン名の変更").Edit("コンピュータ名(C:").Input("{Ctrl_Down}A{Ctrl_Up}{Del}PC01");
```

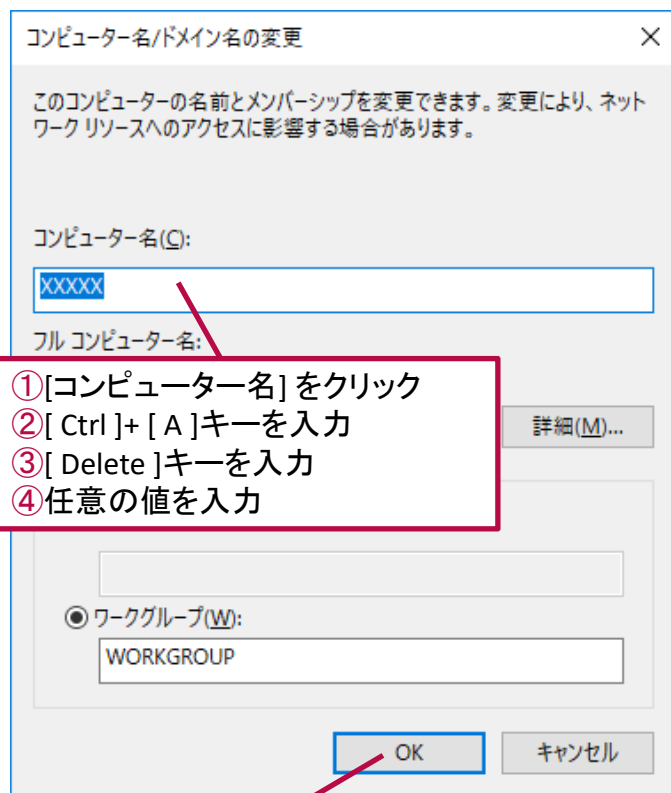
この記録機能では、ほとんどのキーボードの記録を行えますが、下記の点は記録を行えません。

- [Windows]キーを使用したショートカット操作 ([Windows]キーのみは記録可能)

その為、[Windows]キーを使用したショートカットは行わずに、別の方法を使用してキッキングを行いましょう。

## 7

## [コンピュータ名]の値を入力



- ① [コンピュータ名] をクリック
- ② [ Ctrl ]+ [ A ]キーを入力
- ③ [ Delete ]キーを入力
- ④ 任意の値を入力

- ⑤ [OK]ボタンをクリック

## 8 [OK]ボタンをクリック

コンピューター名/ドメイン名の変更

**i** これらの変更を適用するには、お使いのコンピューターを再起動する必要があります

再起動する前に、開いているファイルを保存して、すべてのプログラムを閉じてください。

OK

[OK]ボタンのクリック

## 9 [閉じる]ボタンをクリック

システムのプロパティ

コンピューター名 ハードウェア 詳細設定 システムの保護 リモート

次の情報は、このコンピューターをネットワーク上で識別するために使われます。

コンピューターの説明(D):   
例: "キッチンのコンピューター"、"仕事用コンピューター"

フルコンピューター名: 2017-H-017  
ワークグループ: WORKGROUP

ドメインまたはワークグループに参加するためのウィザードを使用するには [ネットワーク ID] をクリックしてください。 ネットワーク ID(N)...

コンピューター名を変更したりドメインに参加したりするには [変更] をクリックしてください。 変更(O)...

**[閉じる]ボタンのクリック**

⚠ 変更はコンピューターの再起動後に有効になります。

閉じる キャンセル 適用(A)

## 10 [閉じる]ボタンをクリック

Microsoft Windows

これらの変更を適用するにはコンピューターを再起動する必要があります。

再起動する前に、開かれているファイルをすべて保存して、プログラムをすべて閉じる必要があります。

今すぐ再起動する(R) 後で再起動する(L)

[後で再起動する]ボタンのクリック

## 再起動について

本STEPでも、Windowsから再起動を求めるメッセージが表示するシナリオがありますが、キッキングには、再起動を行う場面が多々あります。

SetROBOで再起動を行う場合、Windowsの再起動機能は使用せず、必ずSetROBOで再起動を行ってください。

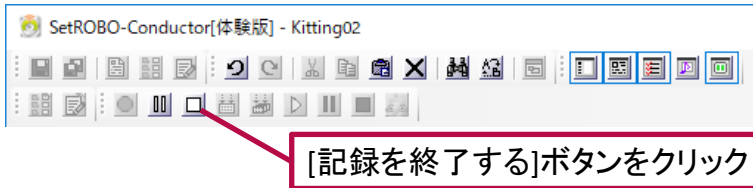
Windowsの機能を使用して再起動をしてしまうと、SetROBOが終了する前に再起動がかかってしまう場合があります。そうなると、Logが正しく出力されず、結果を確認することができなくなってしまいます。

その為、再起動を行う際は必ずSetROBOにて、コードの最後に下記のコードを記述し、再起動を行ってください。

```
PlayLib.ProcessStart("shutdown", "-r -t 5");
```

最後の[-t 5]という数字は、再起動までの時間の指定です。5秒位間をあげると、SetROBOが終了し、Logの出力が完了した状態で再起動が行えます。

## 11 [記録を終了する]ボタンをクリック



## 12 記録したコードについて

記録したコードの内容です。

```
PlayLib.Window("システムのプロパティ").Button("変更(C)...").Click(45, 10);
PlayLib.Sleep(1122);
PlayLib.Window("コンピューター名/ドメイン名の変更").Edit("コンピューター名(C):").Click(32, 10);
PlayLib.Sleep(1320);
PlayLib.Window("コンピューター名/ドメイン名の変更").Edit("コンピューター名(C):")
.Input("{Ctrl_Down}A{Ctrl_Up}{Del}PC01");
PlayLib.Sleep(1587);
PlayLib.Window("コンピューター名/ドメイン名の変更").Button("OK").Click(41, 12);
PlayLib.Sleep(15320);
PlayLib.Window("コンピューター名/ドメイン名の変更").Button("OK").Click(25, 6);
PlayLib.Sleep(1362);
PlayLib.Window("システムのプロパティ").Button("閉じる").Click(44, 9);
PlayLib.Sleep(1923);
PlayLib.Window("Microsoft Windows").Button("後で再起動する(L)").Click(47, 11);
```

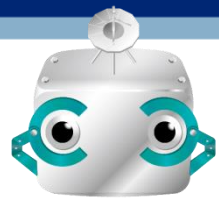
上記コードの内容と違うコードが含まれている場合は、そのコードを削除してください。  
入力時に別のキーを押してしまった場合は、Input()内のコードを上記と同じように修正してください。

なお、Clickの括弧内の値やSleepの括弧内の値は、上記と異なります。  
そのコードについては、削除しないでください。

**【注意】**本STEPで使用するコードには含まれていませんが、コードの記録を行った際に、「PlayLib.Window("OOO").Unknown(xxxxx)」と表示することがあります。

通常ですと、記録したコントロールの種類を表示しますが、SetROBOの記録機能が対応していないコントロールを記録した場合は、「Unknown」と表示します。  
対応していないコントロールを記録した場合は、「UIコントロール」という方法で記述することで動作が可能となります。

「Unknown」と表示した際の記述方法については、「STEP09」をご確認ください。



このSTEPでは、STEP05で記録したコードの[編集]について解説します。  
この編集は、あくまでも一例ですが、編集する際の基本として覚えてください。

### 1 コードにコメントを記入する

記録したコードにコメントを記入します。  
あとでこのコードを見たときに、何をしているのかわかるように、下記のように一つ一つのコードに分かりやすく記述しましょう。

```
//システムのプロパティを開く
PlayLib.ProcessStart("sysdm.cpl");

//システムのプロパティ画面の変更ボタンをクリック
PlayLib.Window("システムのプロパティ").Button("変更(C)...").Click(45, 10);
PlayLib.Sleep(1122);

//コンピューター名/ドメイン名の変更画面のコンピューター名をクリック
PlayLib.Window("コンピューター名/ドメイン名の変更").Edit("コンピューター名(C):").Click(32,10);
PlayLib.Sleep(1320);

//コンピューター名/ドメイン名の変更画面のコンピューター名の値を削除し変更
PlayLib.Window("コンピューター名/ドメイン名の変更").Edit("コンピューター名(C):").Input("{Ctrl_Down}A{Ctrl_Up}{Del}PC01");
PlayLib.Sleep(1587);

//コンピューター名/ドメイン名の変更画面のOKボタンをクリック
PlayLib.Window("コンピューター名/ドメイン名の変更").Button("OK").Click(41, 12);

//コンピューター名の変更の反映処理を行っており待機時間が長い為、長めのSleepで待機する
PlayLib.Sleep(15320);

//コンピューター名/ドメイン名の変更画面のOKボタンをクリック
PlayLib.Window("コンピューター名/ドメイン名の変更").Button("OK").Click(25, 6);
PlayLib.Sleep(1362);

//システムのプロパティ画面を閉じる
PlayLib.Window("システムのプロパティ").Button("閉じる").Click(44, 9);
PlayLib.Sleep(1923);

//Microsoft Windows画面の後で再起動するボタンをクリック
PlayLib.Window("Microsoft Windows").Button("後で再起動する(L)").Click(45, 10);
```

### 👉 記録後の編集ポイント

前のSTEPでも説明しましたが、コードの記録を行ったままだと、不要なコードが記述されていたり、チェックが入っていない、固定の条件しかできないなど、そのままではキッキングに使用することはできません。そこで、必ず編集を行います。

本STEPでも行いますが、コードの編集には、下記のポイントがあります。

- ①Sleepの時間を修正する
- ②変数を使用する
- ③画面を待つコードを使用する
- ④チェックを行うコードを使用する
- ⑤条件分岐を使用する
- ⑥外部ファイルの読込を使用する
- ⑦「Unknown」箇所の修正

あれ？みなさんお気づきでしょうか？  
この編集のポイントは、⑥⑦以外はもうすでにみなさんが今までのSTEPで行ってきた内容ですね。

ぜひ今度は、本書以外のシナリオも作成してみましょ

## 2 変数を宣言し編集する

待機時間とコンピュータ名の変数のコードを追加・編集します。

### 待機時間とコンピュータ名の変数のコード

! コード位置修正

```
// 待機時間間隔
int basicWait = 1000;
//長めの待機時間間隔
int longWait = 15000;

//コンピュータ名の変数
String PCName = "PC01";
```

① [// セットアップコードの記述]の下に、  
[待機時間とコンピュータ名の変数の  
コード]をコピーし張り付ける

② Sleep()の中にbasicWait  
を記述

③ 長めの待機が必要な場合は  
Sleep()の中にlongWaitを記述

④ コンピュータ名の入力箇所を以下の様に  
変更する

【変更前】

```
Input("{Ctrl_Down}A{Ctrl_Up}{Del}PC01");
```

【変更後】

```
Input("{Ctrl_Down}A{Ctrl_Up}{Del}" + PCName);
```

### 3 画面が開くまで待つコードを記述する

以下のコードを追加します。

システムのプロパティ画面が表示するまで待つコード

! コード位置修正

```
//システムのプロパティ画面が表示するまで待つ  
PlayLib.WaitWindow("システムのプロパティ", true ,60);
```

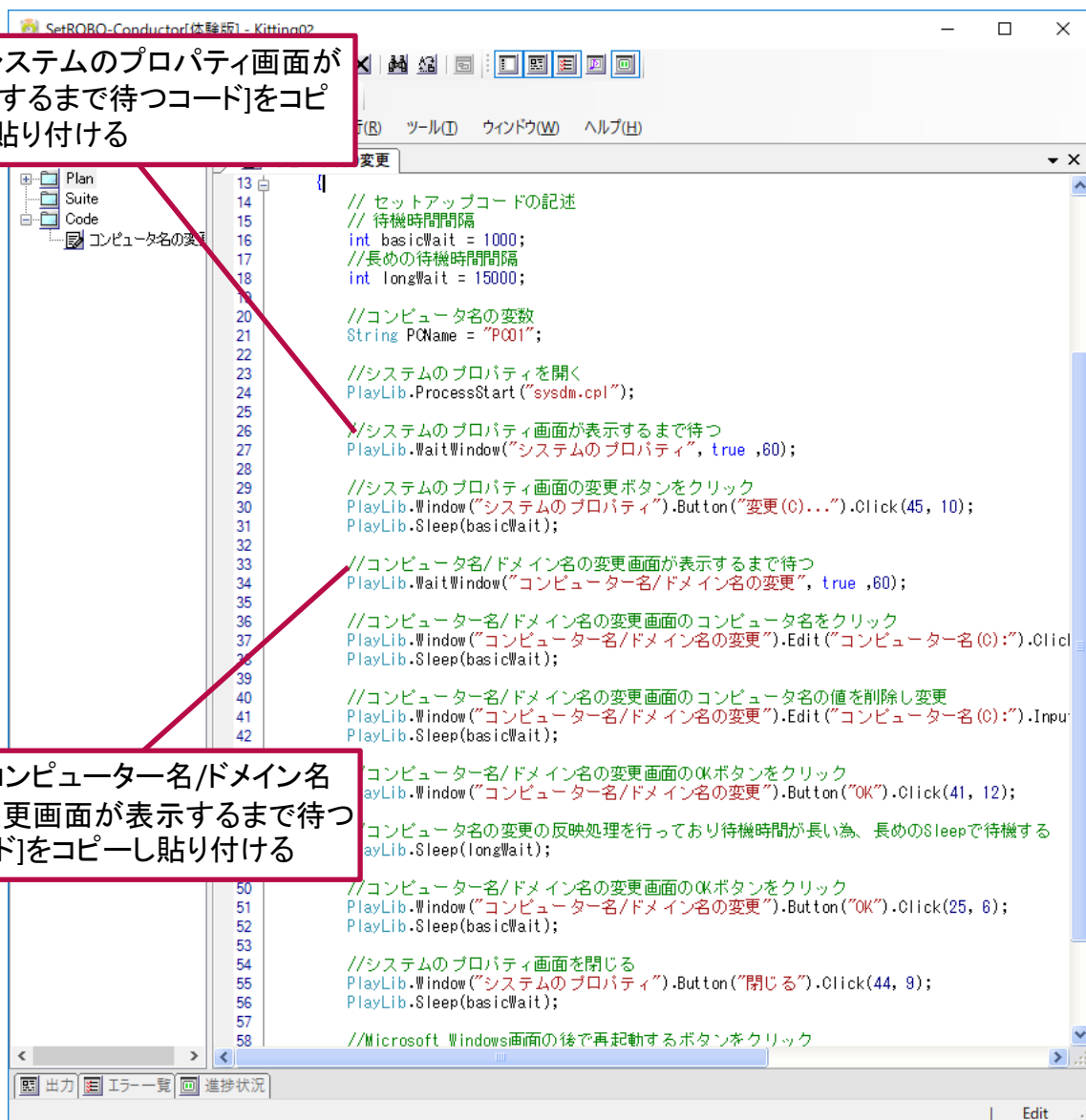
コンピュータ名/ドメイン名の変更画面が表示するまで待つコード

! コード位置修正

```
//コンピュータ名/ドメイン名の変更画面が表示するまで待つ  
PlayLib.WaitWindow("コンピュータ名/ドメイン名の変更", true ,60);
```

①[システムのプロパティ画面が表示するまで待つコード]をコピーし貼り付ける

②[コンピュータ名/ドメイン名の変更画面が表示するまで待つコード]をコピーし貼り付ける

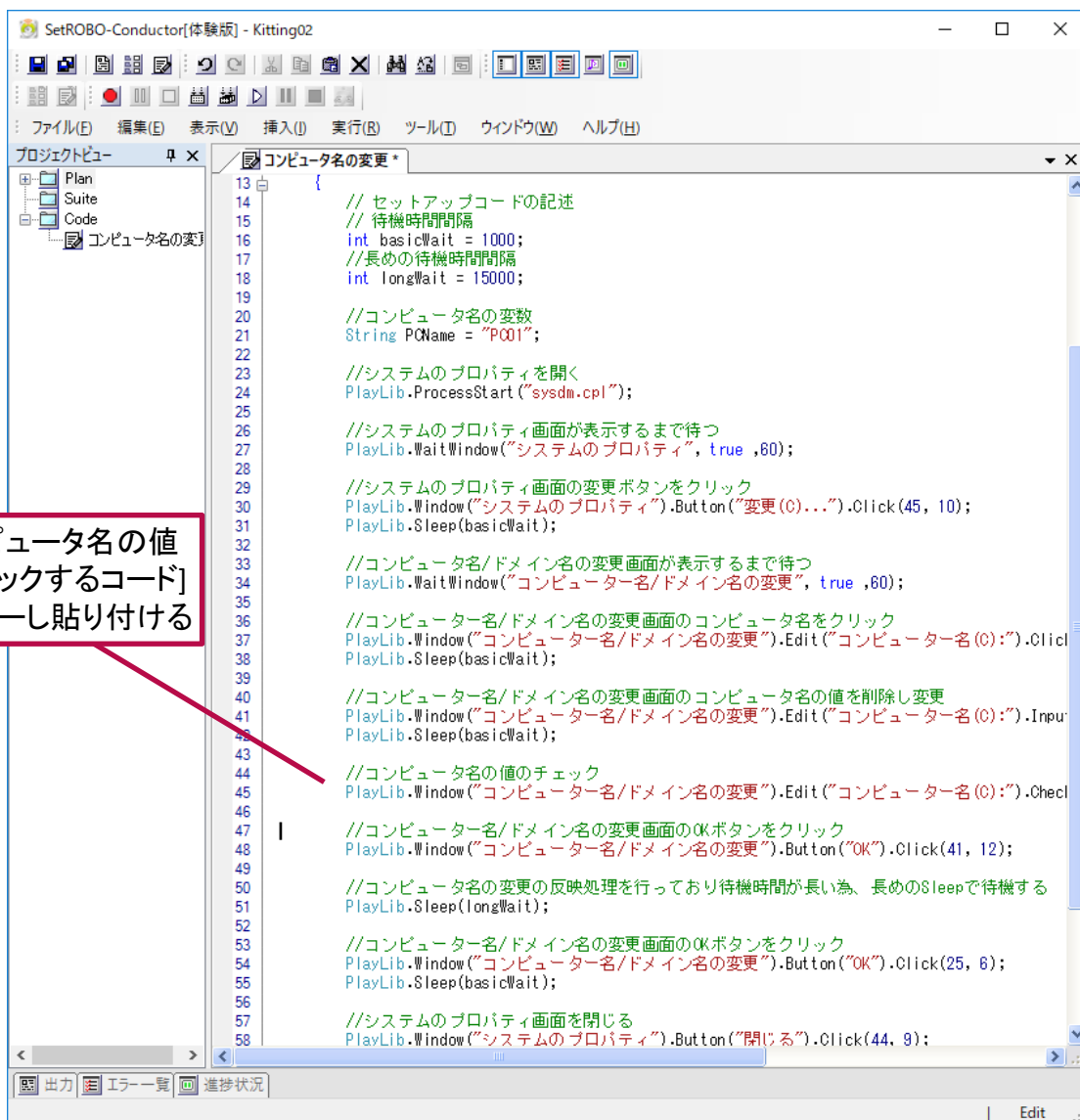


## 4 コンピュータ名の値をチェックする

🔔 コード位置修正

コンピューター名の値をチェックするコード

```
//コンピューター名の値のチェック  
PlayLib.Window("コンピューター名/ドメイン名の変更").Edit("コンピューター名(C):").CheckProperty("Text", PCName);
```



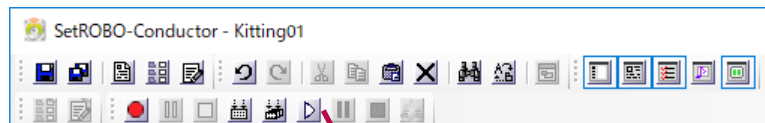
[コンピューター名の値  
をチェックするコード]  
をコピーし貼り付ける

## 5 [実行する]ボタンをクリック

記録したコードを実行します。実行する際に、コンピューター名の記述箇所を変更して、実行してください。

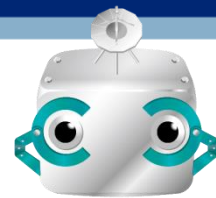
```
//コンピューター名の変数  
String PCName = "PC01";
```

任意の名前に変更



[実行する]ボタンをクリック

【注意】コンピューター名を変更しないでコードを実行すると、コンピューター名が同一となってしまう為、途中で失敗してしまいます。コンピューター名を変更してから実行してください。

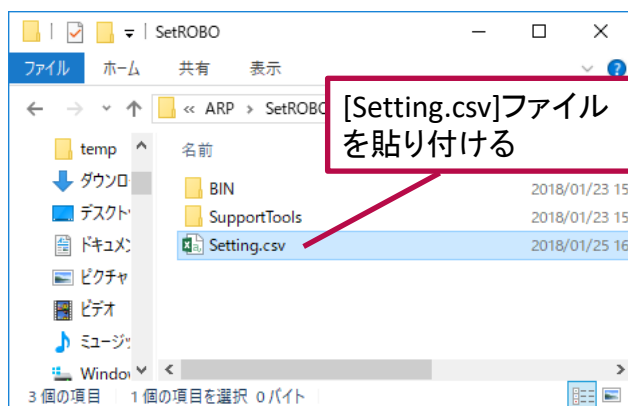


このSTEPでは、STEP05・STEP06で作成したコンピュータ名を変更するコードをベースに、[CSVファイルからデータを読み込む方法]と[入力フォーム]について解説します。

## 1 CSVファイルの配置

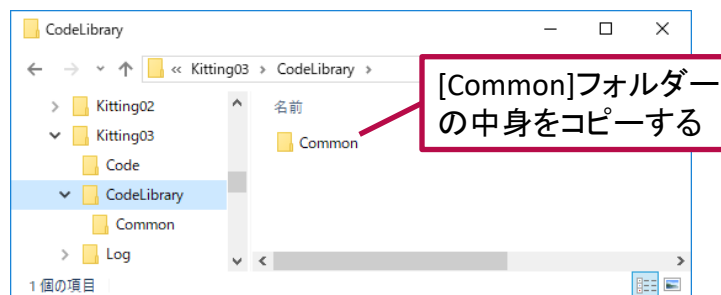
Webサイト「SetROBO for Kitting入門ガイド」ページに公開されている「SetROBO for Kitting入門ガイドで使用するファイル(zipファイル)」の中にある「Setting.csv」ファイルをコピーし、SetROBO for Kittingがインストールされているフォルダ内の下記の場所に貼り付けます。

C:\Program Files\ARP\SetROBO\Setting.csv



## 2 ライブラリの配置

付録として同梱されているCommonフォルダの中身を、STEP05で作成したプロジェクトフォルダ内のCodeLibrary > Commonフォルダの中にコピーします。



### 👉 CSVファイルについて

キittingでは、PC毎別々の設定を行うことがあります。その際に、コードの中に別々の設定を記述していくのはとても大変です。

SetROBOは、外部に配置したCSVファイルを読み込むことができます。予め、CSVファイルに各PCの設定内容を記述しておくことで別々の設定を行うことができます。個々の設定を行う際は、CSVファイルからの取り込みを使用しましょう。

### 👉 CSVファイルの内容を編集しよう

本STEPでは、[入力フォーム]を表示させて、入力した[番号]の行にある[コンピュータ名]を設定します。

本STEPのCSVファイルの内容は、[番号][コンピュータ名]の二列となっています。【番号】は、上から連番で10まで振られています。11以降も同じように設定することができます。また、【コンピュータ名】も同様に、番号の横に任意の値を設定することができます。

CSVファイルの内容は、本STEPの内容以外にも色々な形で使用することができます。詳細な説明については、SetROBOホームページ「お問合せフォーム」よりお問合せ下さい。

### 3 CSVファイルから値を読み込む

CSVファイルから値を読み込む為にコードを変更します。

### 👉 CSVファイルの読 込先について

本STEPでは、CSVファイルの置き場を、SetROBO for Kittingがインストールされているフォルダ内の、実行ファイル（exe）がある階層の一つ上の階層に配置しています。これは、コード内にてその場所読込に行くという内容が記述されているからです。実際のコードは下記の通りです。

```
string sFilePath =  
Path.GetDirectoryName(Appli  
cation.ExecutablePath);  
sFilePath =  
Path.Combine(Path.GetDirect  
oryName(sFilePath), "[ファイ  
ル名]");
```

この読込先は任意で変更ができます。例えば、今後配布用の[セットアップ用パッケージ]を作成し、そのフォルダ内（実行ファイルと同じ階層）に配置する場合は、下記のコードを記述します。

```
string sFilePath =  
Path.GetDirectoryName(Appli  
cation.ExecutablePath) +  
"¥¥[ファイル名]";
```

または、読込先を固定の場所にする場合は、下記のようにパスとファイル名を直接記述しても読込可能です。

```
string sFilePath =  
"C:¥¥Users¥¥Windows10¥¥De  
sktop¥¥PC-Kitting01¥¥[ファイ  
ル名]";
```

なお、コード内で記述する[¥]は、[¥¥]と記述して下さい。また、ファイル名は任意の値が設定できますので、分かりやすい名前にしましょう。

#### ライブラリを読み込むコード

🚨 コード位置修正

```
//Windowsのライブラリ  
using System.IO;  
using System.Windows.Forms;  
  
// 入力画面用ライブラリ  
using InputFormLib;
```

#### CSVファイルから値を読み込むコード

🚨 コード位置修正

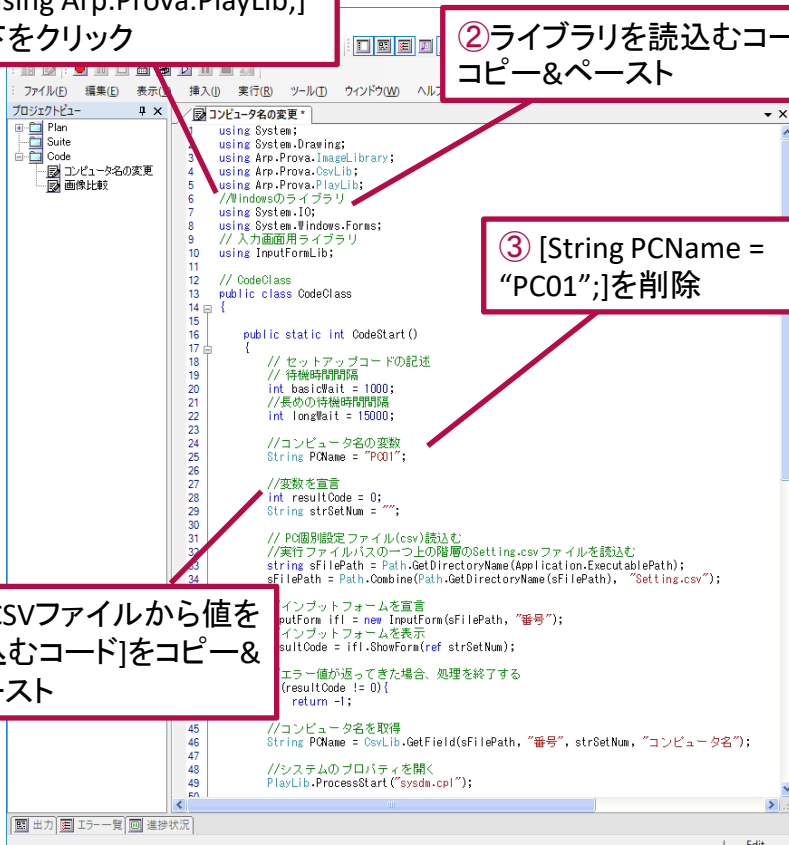
```
//変数を宣言  
int resultCode = 0;  
String strSetNum = "";  
  
// PC個別設定ファイル(csv)読み込む  
//実行ファイルパスの一つ上の階層のSetting.csvファイルを読み込む  
string sFilePath = Path.GetDirectoryName(Application.ExecutablePath);  
sFilePath = Path.Combine(Path.GetDirectoryName(sFilePath), "Setting.csv");  
  
//インプットフォームを宣言  
InputForm ifl = new InputForm(sFilePath, "番号");  
//インプットフォームを表示  
resultCode = ifl.ShowForm(ref strSetNum);  
  
//エラー値が返ってきた場合、処理を終了する  
if(resultCode != 0){  
    return -1;  
}  
//コンピュータ名を取得  
String PCName = CsvLib.GetField(sFilePath, "番号", strSetNum, "コンピュータ名");
```

① [using Arp.Prova.PlayLib;]  
の下をクリック

② ライブラリを読み込むコードを  
コピー&ペースト

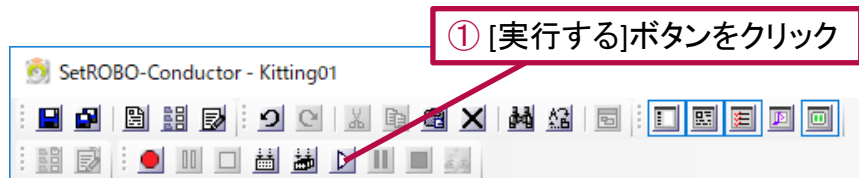
③ [String PCName =  
"PC01;"]を削除

④ [CSVファイルから値を  
読み込むコード]をコピー&  
ペースト

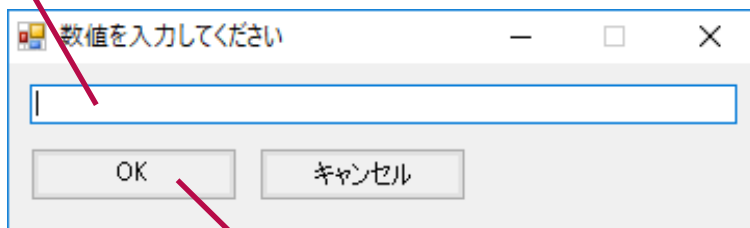


## 4 コードを実行する

コードを実行すると、入力フォーム画面が表示します。  
入力フォームに入力した値は、CSVファイルの番号と紐付き、一致した番号の行の設定を行います。  
本STEPでは、入力フォームに数値を入力し、OKボタンをクリックするとCSVファイルのコンピュータ名の値を自動で読み込みコンピュータ名の変更が始まります。



② 入力欄をクリックし、[1]を入力

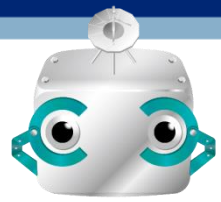


③ [OK]ボタンをクリック

**【注意】** コードを実行した際にCSVファイルが存在しない場合は、エラーとなります。  
また、入力フォーム画面で入力した値が、CSVファイルに存在しない場合もエラーとなります。

なお、この入門ガイドで使用している入力フォームは、簡易版となりますので、実際にご使用の際は、SetROBOホームページの「サポートページ」に掲載している入力フォームをご使用ください。

また、CSVファイルとの紐付け方法は、入力フォーム以外にPCの製造番号（シリアルナンバー）をキーにする方法や、MACアドレスをキーにする方法などさまざまな方法があります。  
こちらについても、SetROBOホームページの「サポートページ」をご確認ください。



このSTEPでは、記録ができないコントロールなどを操作する際に、画像を比較して一致した個所の操作する方法について解説します。

なお、この機能の詳しい内容は、付録2を参照して下さい。

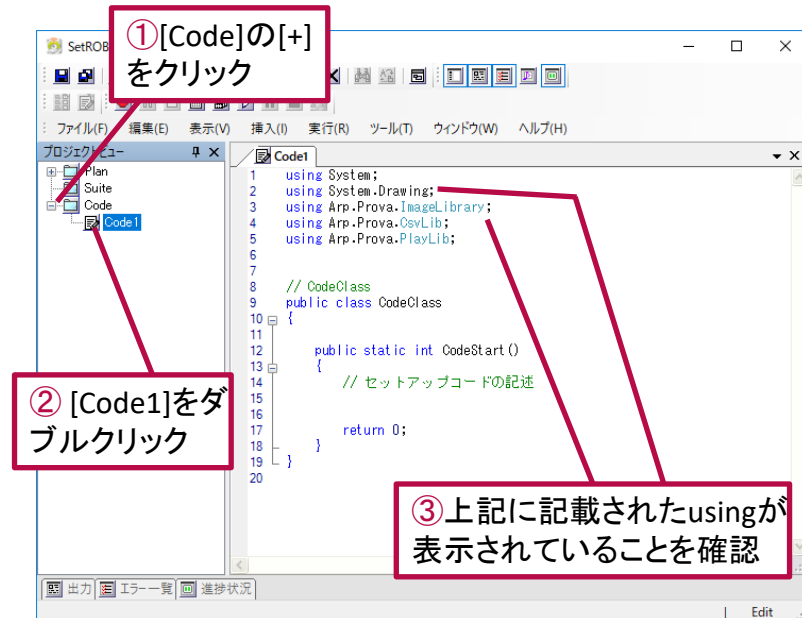
## 1 事前確認

画像を比較して操作を行うために、事前確認を行います。

SetROBO for Kitting Conductorを実行し、作成したプロジェクト内のコードを表示します。

表示したコード内に以下の using が表示されていることを確認します。

- using Arp.Prova.ImageLibrary;
- using System.Drawing;



### 各usingについて

「using Arp.Prova.ImageLibrary;」は、画像比較用ライブラリをロードして使用するためのものです。

「using System.Drawing;」は、画像の座標を扱うPoint型を使用するためのものです。

### usingの記載について

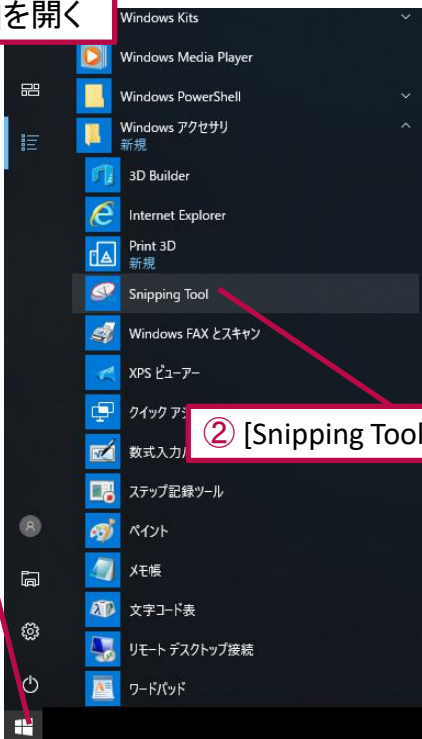
画像比較用に使用するusingは、SetROBOのバージョン「1.9.0」から自動的に挿入されるようになっています。

バージョン「1.9.0」より前のバージョンのSetROBOを使用している場合、「1.9.0」以上のバージョンにバージョンアップして下さい。

## 2 比較画像の準備

画像の比較を行うための、比較元画像を取得します。  
今回は、OS標準装備されている「Snipping Tool」を使用します。

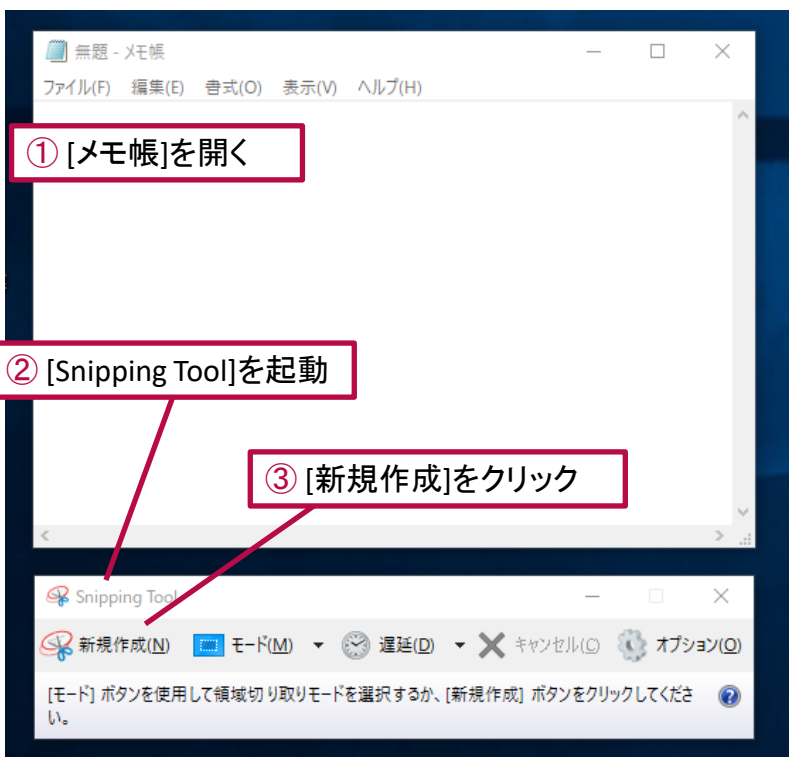
① [スタート]を開く



② [Snipping Tool]を選択

例として「メモ帳」の閉じるボタンを比較します。  
比較するために必要な「メモ帳」の閉じるボタンの画像を「Snipping Tool」を使用して用意します。

① [メモ帳]を開く



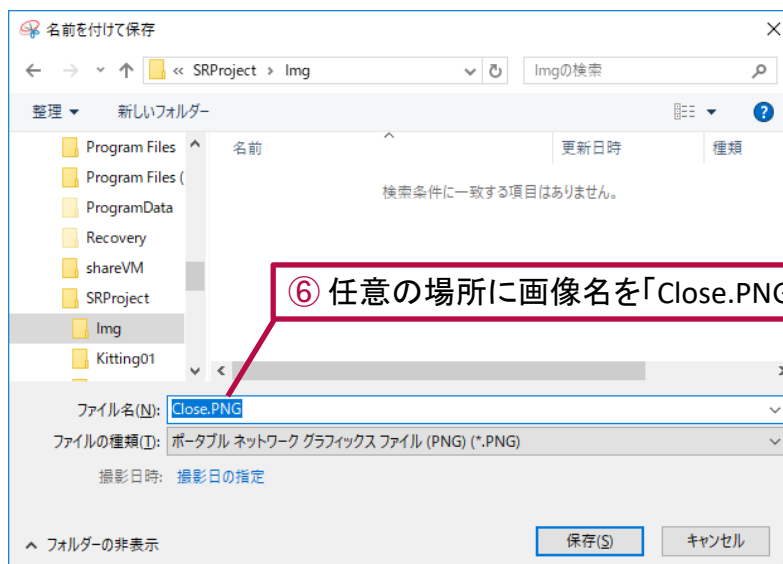
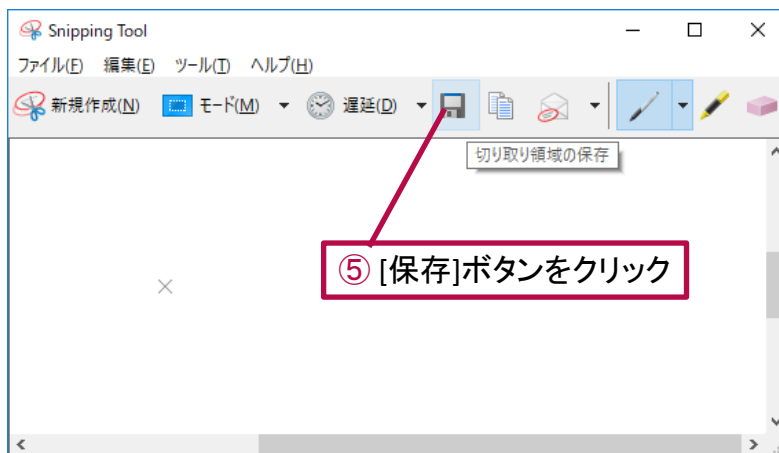
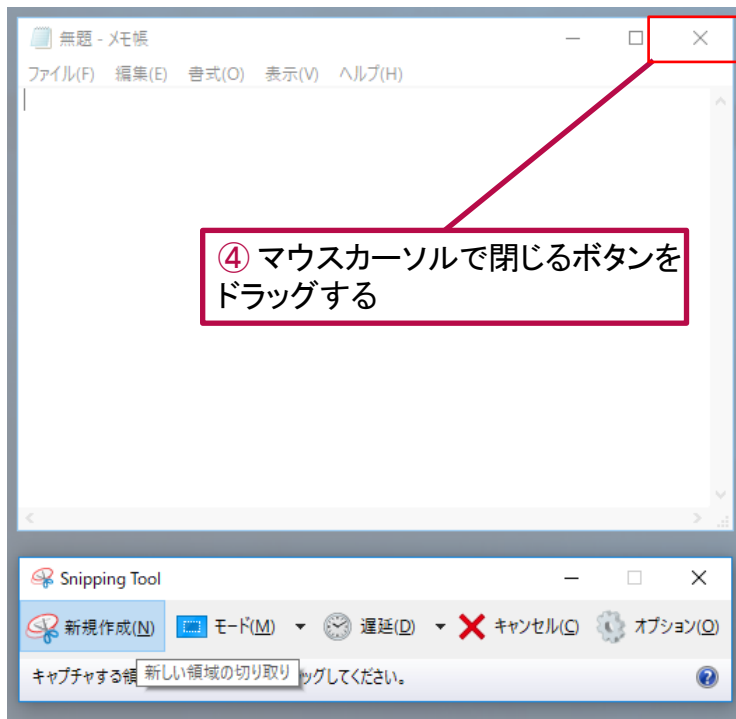
## Snipping Toolについて

Snipping Toolは、OS純正のツールで、画面をキャプチャーするツールです。

フリーハンドや四角形での範囲の指定、ウィンドウや画面全体などさまざまな形式でパソコン上の画面を範囲選択し、キャプチャーすることができます。

詳しい使い方に関しては、以下のリンクを参照して下さい。

<https://support.microsoft.com/ja-jp/help/13776/windows-use-snipping-tool-to-capture-screenshots>



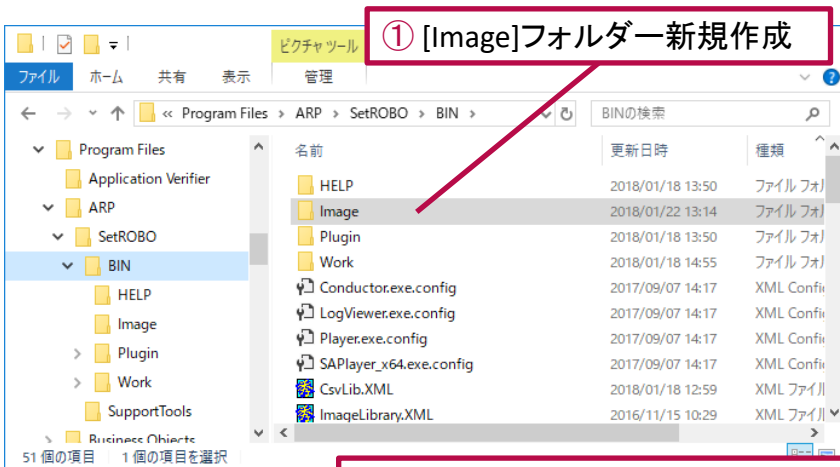
### 3 比較画像の配置

比較画像の準備ができましたら、用意した比較画像を配置します。今回は、SetROBO for Kitting Conductorが格納されているフォルダ内にImageフォルダーを新規に作成し、作成したImageフォルダ内に画像を配置します。

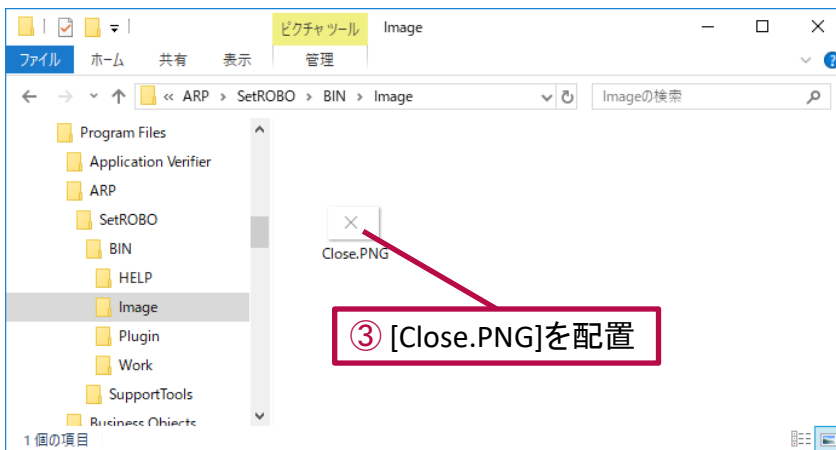
配置する場所は、以下の用になります。

#### 格納場所

C:\ProgramFiles\ARP\SetROBO\BIN\Image\Close.PNG



#### ② [Image]フォルダーをダブルクリック

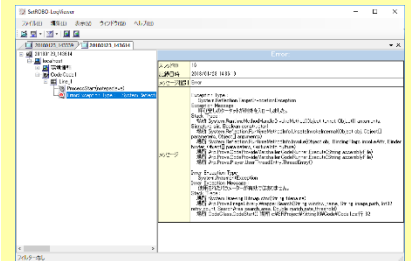


### 比較画像の格納場所について

基本的に比較画像の格納場所は、任意で指定できます。

任意で格納できますが、画像比較用のコードを作成するときに任意で格納した画像のパスを正しく指定して下さい。

指定が間違っている場合は、「SetROBO-LogViewer」でエラー表示されますので正しく指定して下さい。



## 4 画像比較用のコード作成する

画像比較できるようにコードを修正します。

まず、必要なライブラリをロードし、必要な定数を用意します。

### ライブラリを読み込むコード

```
//Windowsのライブラリ  
using System.IO;  
using System.Windows.Forms;
```

Windows上のパス情報を扱うため追加

アプリケーションのパス情報を扱うため追加

### 比較用画像のパスを用意するコード

```
// 比較用画像のパスを取得  
string CloseImagePath = Path.GetDirectoryName(  
    Application.ExecutablePath) + @"%Image%Close.PNG";
```

### ! ¥ 注意

本コードをコピーして貼り付けする際は、¥の部分を一旦全て削除して入力しなおしてください。

① [using Arp.Prova.PlayLib;]の下をクリック

```
1 using System;  
2 using System.Drawing;  
3 using Arp.Prova.ImageLibrary;  
4 using Arp.Prova.CsvLib;  
5 using Arp.Prova.PlayLib;  
6 //Windowsのライブラリ  
7 using System.IO;  
8 using System.Windows.Forms;  
9  
10 // CodeClass  
11 public class CodeClass  
12 {  
13  
14     public static int CodeStart ()  
15     {  
16         // セットアップコードの記述  
17  
18         // 比較用画像のパスを取得  
19         string CloseImagePath = Path.GetDirectoryName(  
20             Application.ExecutablePath) + @"%Image%Close.PNG";  
21  
22         return 0;  
23     }  
24 }  
25
```

② 上記のライブラリを読み込むコードをコピー&ペースト

③ 上記の比較用画像のパスコードをコピー&ペースト  
\*¥は一旦全て削除して入力しなおしてください。

## Windowsのライブラリについて

Windows上のパス情報とアプリケーションのパス情報を扱うため、左側で記載したライブラリをロードする必要があります。指定せずに画像比較用のコードを修正するとコンパイル時にエラーになります。

using System.IO;  
をロードしなかった場合、以下のようなエラーになります。

エラー一覧	
説明	
名前 'Path' は現在のコンテキスト内に存在しません。	

また、using System.Windows.Forms;をロードしなかった場合、以下のようなエラーになります。

エラー一覧	
説明	
名前 'Application' は現在のコンテキスト内に存在しません。	

## 4 画像を比較するように修正する

自動で「メモ帳」を起動し、起動した「メモ帳」に対して比較対象の「閉じる」ボタンが存在するか検索し、比較するようにコードを修正します。

### メモ帳を起動するコード

```
// メモ帳を開く  
PlayLib.ProcessStart("notepad.exe");  
// 1秒待機  
PlayLib.Sleep(1000);
```

### メモ帳の閉じるボタンを検索するコード

```
// 開いたメモ帳の閉じるボタンを検索するだけの場合は以下のように記載します。  
bool ret = ImageLibrary.Search(CloseImagePath);
```

### 指定した画像と合致するか確認するコード

```
// 指定した画像と合致する箇所があった場合、TRUEが返ってくる  
PlayLib.CheckValue(ret, true, true);
```

### 開いたメモ帳を閉じるコード

```
// 開いたメモ帳を閉じる  
ImageLibrary.Click(CloseImagePath);
```

① 上記のメモ帳を起動するコードをコピー&ペースト

② 1秒待機コード追加

③ 上記のメモ帳の閉じるボタン検索コードをコピー&ペースト

④ 上記の指定した画像と合致したか確認するコードをコピー&ペースト

⑤ 上記のメモ帳を閉じるコードをコピー&ペースト

## 👉 Searchメソッドについて

画像比較用ライブラリ「ImageLibrary.dll」で提供しているメソッドの一つです。

処理内容は、指定した画像と合致する箇所を検索し、指定した画像が見つければbool値のTRUEを返す。見つからなかった場合、bool値のFALSEを返します。

Searchメソッドの引数は、以下のとおりになります。

- ① ウィンドウ名
- ② 画像ファイルパス
- ③ 検索回数
- ④ 検索エリア
- ⑤ 一致率

引数の詳しい説明は、P54ページの付録3を参照して下さい。

## 👉 Clickメソッドについて

画像比較用ライブラリ「ImageLibrary.dll」で提供しているメソッドの一つです。

処理内容は、指定した画像と合致する箇所を検索し、見つかった場合には、その座標に対してマウスの左クリックを行います。

Clickメソッドの引数は、以下のとおりになります。


- ① ウィンドウ名
- ② 画像ファイルパス
- ③ 検索回数
- ④ 検索エリア
- ⑤ 一致率

引数の詳しい説明は、P54ページの付録3を参照して下さい。

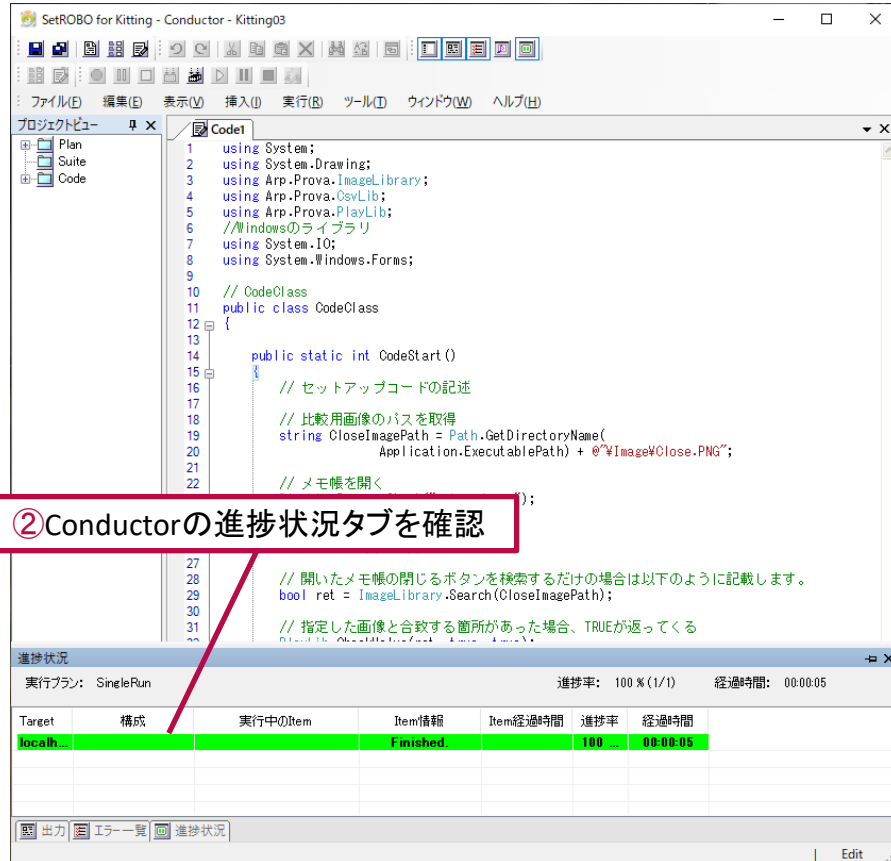
## 5 コードを実行する

コードを実行すると、「メモ帳」が自動的に立ち上がり比較用に用意した画像と一致する箇所を検索します。比較用の画像が見つかったかどうか判断して「メモ帳」を自動的に閉じます。

① [実行する]ボタンをクリック



② Conductorの進捗状況タブを確認



```
1 using System;
2 using System.Drawing;
3 using Arp.Prova.ImageLibrary;
4 using Arp.Prova.CsvLib;
5 using Arp.Prova.PlayLib;
6 //Windowsのライブラリ
7 using System.IO;
8 using System.Windows.Forms;
9
10 // CodeClass
11 public class CodeClass
12 {
13
14     public static int CodeStart ()
15     {
16         // セットアップコードの記述
17
18         // 比較用画像のパスを取得
19         string CloseImagePath = Path.GetDirectoryName(
20             Application.ExecutablePath) + @"\Image\Close.PNG";
21
22         // メモ帳を開く
23     };
24
25     // 開いたメモ帳の閉じるボタンを検索するだけの場合は以下のように記載します。
26     bool ret = ImageLibrary.Search(CloseImagePath);
27
28     // 指定した画像と合致する箇所があった場合、TRUEが返ってくる
```

進捗状況  
実行プラン: SingleRun 進捗率: 100 % (1/1) 経過時間: 00:00:05

Target	構成	実行中のItem	Item情報	Item経過時間	進捗率	経過時間
localhost			Finished.		100 %	00:00:05

コードを実行した際に以下のように表示される場合は、実行結果が期待した内容になっていないことを示します。


進捗状況						
実行プラン: SingleRun 進捗率: 100 % (1/1) 経過時間: 00:00:04						
Target	構成	実行中のItem	Item情報	Item経過時間	進捗率	経過時間
localhost			Finished.		100 % (1/1)	00:00:04

## 6 コード実行結果を確認する

コードを実行すると、「SetROBO-Player」と「SetROBO-LogViewer」が自動的に立ち上がり、実行結果が確認できます。

「SetROBO-Player」では、どのような動作をしたかが時系列で表示されます。

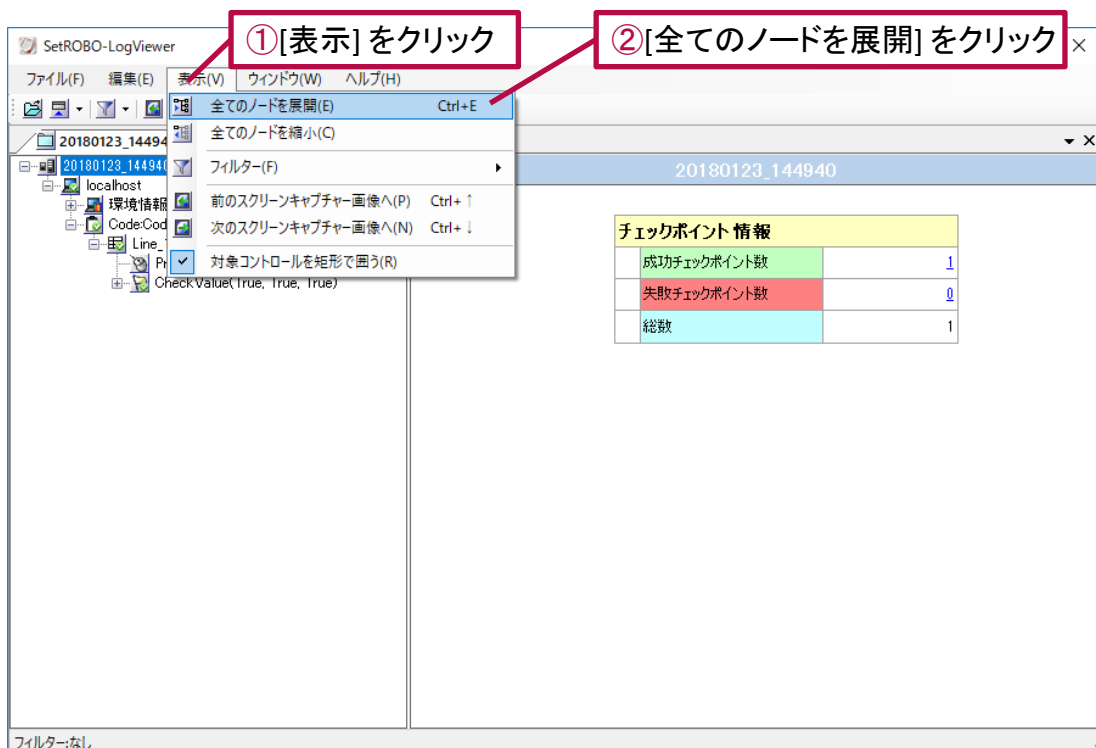
「SetROBO-LogViewer」では、左側にコードを実行した環境情報やコード内で実行された動作をツリー状で表示し、右側にテスト結果の成功・失敗数が表示されます。



日時	種別	情報
2022/01...	状態	接続されました(相手は 127.0.0.1)
2022/01...	状態	再生データの受信中です
2022/01...	状態	実行開始待ちです
2022/01...	状態	再生データの実行中です
2022/01...	動作	ProcessStart(notepad.exe)
2022/01...	動作	CheckValue(True, True, True)
2022/01...	スクリ	スクリーンキャプチャーを取りました(ScreenCapture_1.png)
2022/01...	チェック	結果 成功, 期待値 True, 実測値 True, タイムアウト 0秒, 比較値1 True, 比較値2 True
2022/01...	スクリ	スクリーンキャプチャーを取りました(ScreenCapture_1.png)
2022/01...	状態	待機状態です
2022/01...	状態	切断されました(相手は 127.0.0.1)

成功チェックポイント数：期待通りのチェック結果となった数  
失敗チェックポイント数：期待と異なるチェック結果となった数

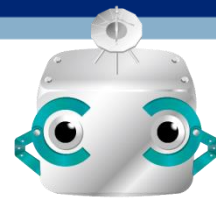
🟢：成功マーク    🚫：失敗マーク



①[表示] をクリック

②[全てのノードを展開] をクリック

チェックポイント情報	
成功チェックポイント数	1
失敗チェックポイント数	0
総数	1



このSTEPでは、「操作の記録」機能を使用した際、コントロールの記録が取れない場合の記述方法について解説します。

STEP4、STEP5でも少し触れましたが、操作対象の画面やコントロールによって「操作の記録」を行った際に記録が取れない（コントロール名が「Unknown」と記録される状態）の記述方法について解説します。

記録が取れないコントロールを記録した際の例：

```
PlayLib.Window("デバイスとプリンター").Unknown("DirectUIHWND", "DirectUIHWND&1").Click(73, 14);
```

## 1 事前準備

記録が取れないコントロールの場合、そのコントロールが、どの種類でどのような名前なのかが分かりません。

その際に、使用するのが「Inspect」というツールです。

「Inspect」は、Unknownと記録したコントロールにマウスカーソルを当てるだけで、操作対象のコントロールの種類や正式名称を教えてください。

まず事前準備として、「Inspect」を入手しましょう。入手方法は、下記の手順となります。

①SetROBOサポートページより「参考資料」ページを開く

<https://www.setrobo.jp/support/Reference/index.html>

②「参考資料」ページ内のドキュメント

「InspectObjectsインストール手順について(pdfファイル)」をダウンロードし、その内容に従い

「Inspect」を入手してください。

### Inspectについて

Inspectとは、マイクロソフト社が提供する開発者向けキット「Windows SDK for Windows10」の中に含まれている機能の一部で、exe単体で動作するアプリケーションです。

Inspectは、デスクトップ上に表示している画面やコントロールの情報を取得することが出来ます。マウスで調べたい画面やコントロールにフォーカスを当てるだけで、その種類や名前を取得し、Inspectツール内に表示します。

操作の記録で取得出来ないコントロールの場合は、画面やコントロールの情報が分からないとシナリオ作成が出来ませんので、Inspectを使用し、画面やコントロールの種類を調べ、その情報を基にシナリオを記述することで、捜査の記録ができない箇所も自動化することが可能となります。

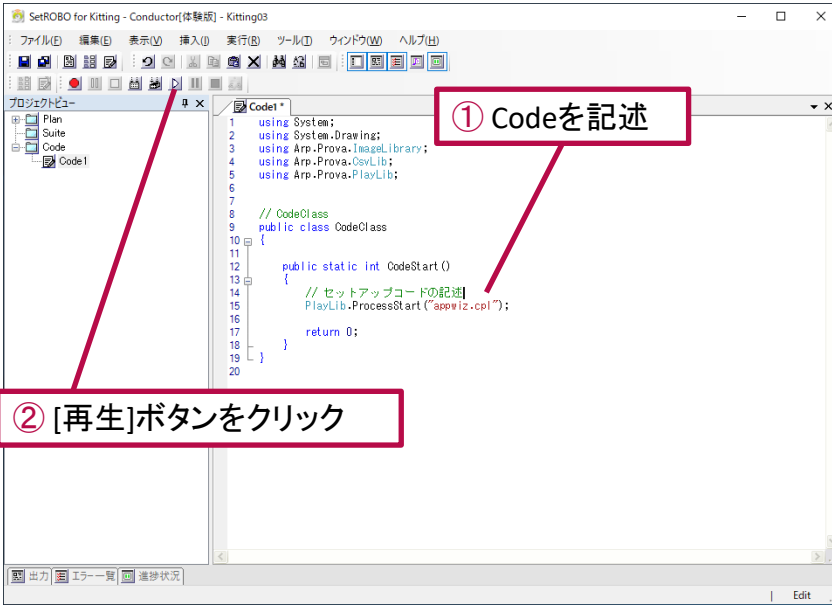
## 2 「プログラムと機能」画面を開く

今回は、「プログラムと機能」画面を使用して、リンクを操作する記述とボタンを操作する記述を試してみたいと思います。

まずは、任意のCodeを作成し、ProcessStart関数とショートカットコマンドを使用して、「プログラムと機能」画面を表示しましょう。

## 「プログラムと機能」画面を起動するコード

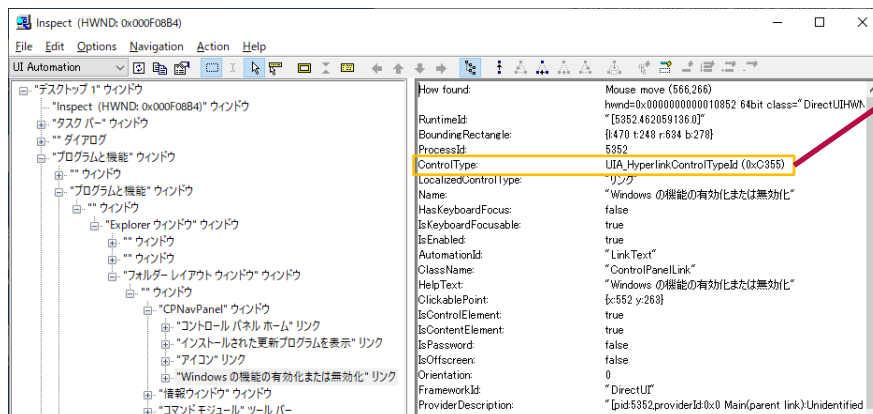
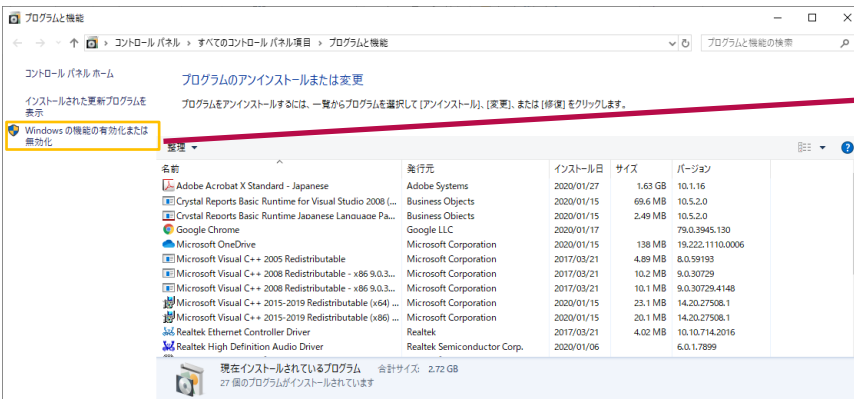
```
PlayLib.ProcessStart("appwiz.cpl");
```



## 3 「Inspect」を起動し、コントロールの種類を調べる

まずは、Inspectを起動し「Windowsの機能の有効化または無効化」リンクを調べてみましょう。

※「Inspect」の使用方法についても、SetROBOホームページ内、「サポートページ」にある「参考資料」ページ内のドキュメント「InspectObjectsインストール手順について(pdfファイル)」をご確認ください。



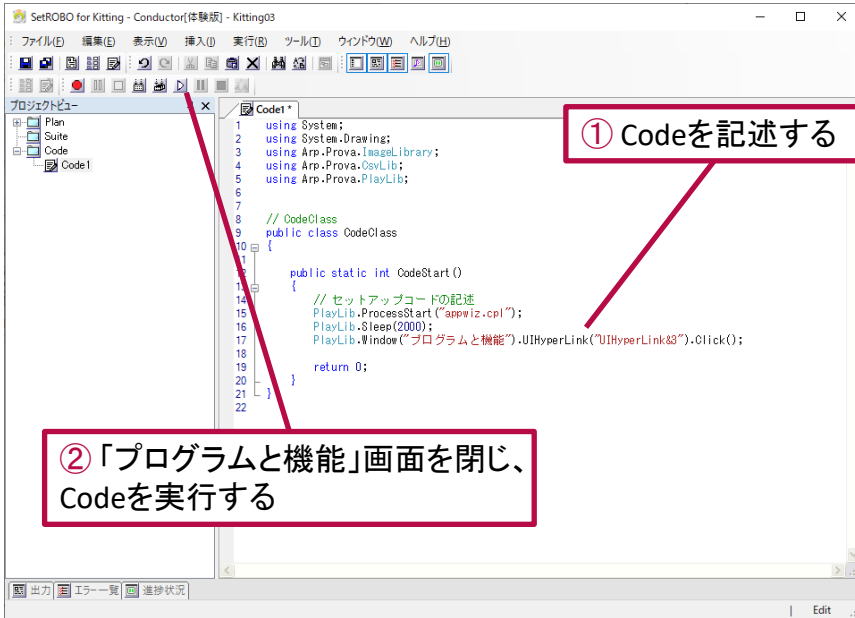
## 4 Codeを記述する

Inspectで調べた結果、コントロールは「UIHyperLink」ということが分かりましたので、それを基にCodeを記述します。

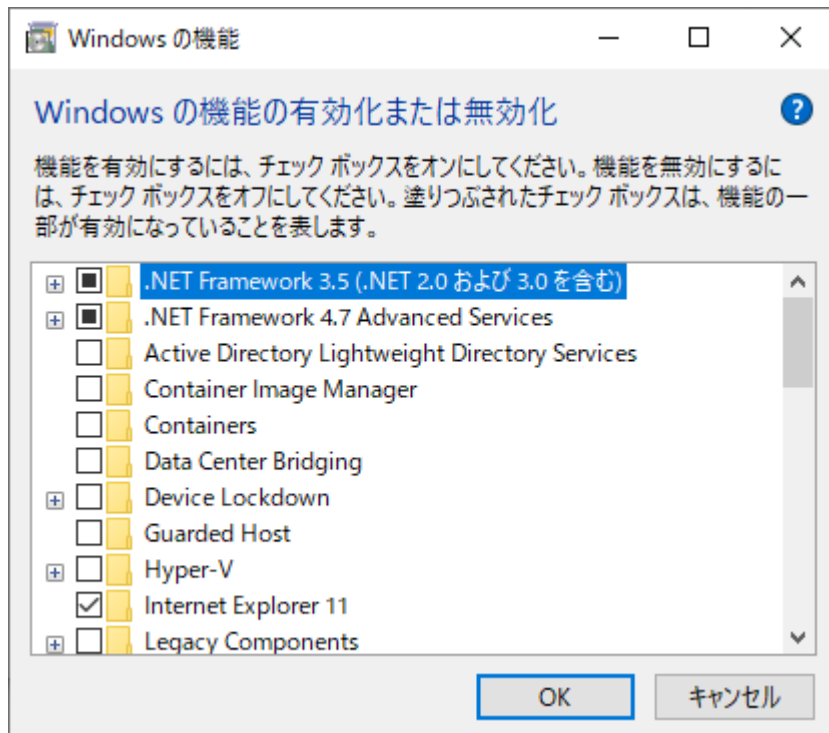
※UIHyperLink関数の記述方法は、Conductorヘルプの「PlayLib関数>コントロール指定」をご確認ください。

### SleepとHyperLinkのコード

```
PlayLib.Sleep(2000);  
PlayLib.Window("プログラムと機能").UIHyperLink("UIHyperLink&3").Click();
```



「Windowsの機能」画面が表示します



## UIHyperLinkの「&番号」について

UIHyperLinkの「&」の後ろにしている番号について説明します。  
「&」の後ろに指定する番号とは、何番目のHyperLinkかを指定しています。

「プログラムと機能」画面の場合、左上側からリンクが下記の順番で表示しています。

- ①コントロールパネルホーム
- ②インストールされた更新プログラムを表示
- ③Windowsの機能の有効化または無効化

今回クリックしたいリンクは、左上から3番目にありますので、「&3」と指定したことで、「Windowsの機能の有効化または無効化」リンクをクリックすることができました。

また、「&番号」はUIHyperLinkだけでなく、ボタンを指定するUIButtonや、入力欄を指定するUIEditでも使用します。ただし、どの関数も番号のルールは同じなので、操作対象のコントロールが左上から何番目にあるかを確認し記述することで指定できます。

本章の最後に、操作対象のコントロール名かどうかを判断し、見つけたら操作するサンプルを記載しています。ぜひ、確認してみてください。

## 5 UIButtonのコントロールを調べる

次に、Button系で記録ができないコントロールを調べて、記述してみましょう。  
「プログラムと機能」画面のインストールされたアプリケーション一覧から、任意のアプリケーションを選択し「アンインストール」ボタンをクリックしてみましょう。

### 【注意】

アプリケーション一覧から選択するアプリケーションは、削除してもよいアプリケーション、または削除の有無を確認するメッセージが表示するアプリケーションを選択してください。

### ※例 Google Chromeの場合

名前	発行元	インストール日	サイズ	バージョン
Adobe Acrobat X Standard - Japanese	Adobe Systems	2020/01/27	1.63 GB	10.1.16
Canon Generic Plus LIPSLX Printer Driver Uninstaller	Canon Inc.	2020/01/15		7, 1, 0, 0
Crystal Reports Basic Runtime for Visual Studio 2008 (...)	Business Objects	2020/01/15	69.6 MB	10.5.2.0
Crystal Reports Basic Runtime Japanese Language Pa...	Business Objects	2020/01/15	2.49 MB	10.5.2.0
CyberLink Power2Go 12	CyberLink Corp.	2017/03/21	201 MB	12.0.4719.55
FileZilla Client 3.46.3	FileZilla Project	2020/01/15	27.1 MB	3.46.3
Google Chrome	Google LLC	2020/01/17		79.0.3945.130
Lhaplus		2020/01/15		
Microsoft Office Home and Business 2016 - ja-jp	Microsoft Corporation	2020/01/27		16.0.12325.20344
Microsoft Visual C++ 2005 Redistributable	Microsoft Corporation	2017/03/21	4.89 MB	8.0.59193
Microsoft Visual C++ 2008 Redistributable - x86 9.0.3...	Microsoft Corporation	2017/03/21	10.2 MB	9.0.30729
Microsoft Visual C++ 2008 Redistributable - x86 9.0.3...	Microsoft Corporation	2017/03/21	10.1 MB	9.0.30729.4148
Microsoft Visual C++ 2015-2019 Redistributable (x64) ...	Microsoft Corporation	2020/01/15	23.1 MB	14.20.27508.1
Microsoft Visual C++ 2015-2019 Redistributable (x86) ...	Microsoft Corporation	2020/01/15	20.1 MB	14.20.27508.1

How found: Mouse move (762,272)

RuntimeId: hwnd=0x000000000000B051E 64bit class="DirectUIHWI"

BoundingRectangle: [5352.45; 66448.0]

ProcessId: [1:705; 265; r:795; b:291]

ControlType: UIA\_ButtonControlTypeId (0xC350)

LocalizedControlType: ボタン

Name: アンインストール

HasKeyboardFocus: false

IsKeyboardFocusable: true

IsEnabled: true

AutomationId: "{8E2267F1-30BD-489B-9DD6-2887121113E8}"

ClassName: "AJOSplitButton"

HelpText: "このプログラムをアンインストールします。"

ClickablePoint: [x:750; y:278]

IsControlElement: true

IsContentElement: true

IsPassword: false

IsOffscreen: false

Orientation: 0

FrameworkId: "DirectUI"

ProviderDescription: "[pid:5352;providerId:0x0; Main(parent link):Unidentified"

Value.IsReadOnly: true

Value.Value: ""

LegacyIAccessible.ChildId: 0

LegacyIAccessible.DefaultAction: "押す"

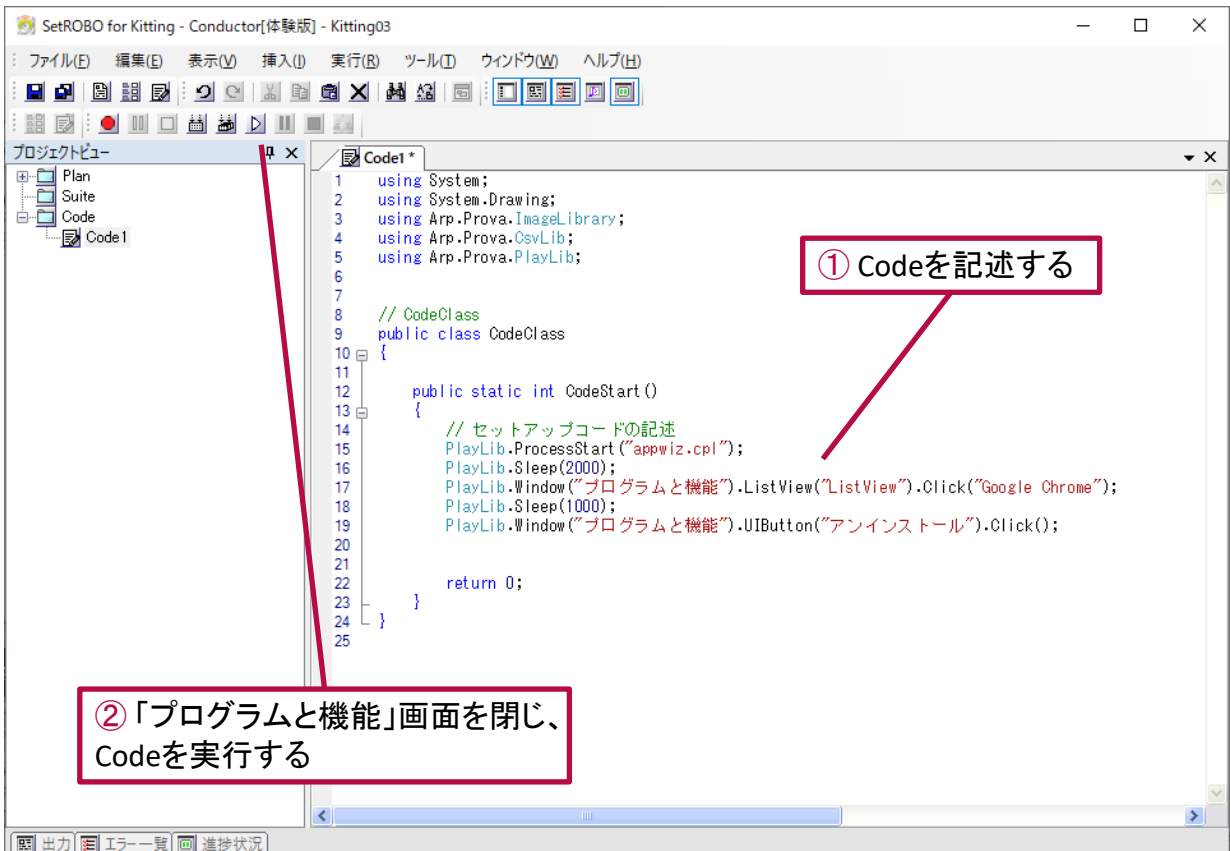
## 6 Codeを記述する

Inspectで確認した内容を基に、UIButtonコントロールを記述しましょう。

「プログラムと機能」画面の起動～アンインストールをクリックするコード

```
PlayLib.ProcessStart("appwiz.cpl");
PlayLib.Sleep(2000);

PlayLib.Window("プログラムと機能").ListView("ListView").Click("Google Chrome");
PlayLib.Sleep(1000);
PlayLib.Window("プログラムと機能").UIButton("アンインストール").Click();
```



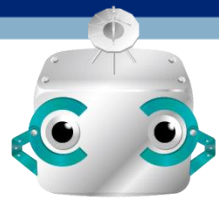
### 【「&番号」の番外編】

「&番号」の番号を指定するコードの場合、操作対象の文字列かどうかを比較し、一致したら操作することができます。

下記のように、for文で番号をカウントアップして文字列を取得し、操作対象の文字列と操作対象の文字列が一致するか比較することで正確に操作することが可能となります。

```
for(int i = 1; i < 10; i++)
{
    string linkName = PlayLib.Window("プログラムと機能").UIHyperLink("UIHyperLink&" + i).Text;

    if(linkName == "Windows の機能の有効化または無効化")
    {
        PlayLib.Window("プログラムと機能").UIHyperLink("UIHyperLink&" + i).Click();
        break;
    }
}
```



このSTEPでは、セットアップ用パッケージの作成方法について解説します。

※この機能は、体験版では使用できません。

### セットアップ用パッケージについて

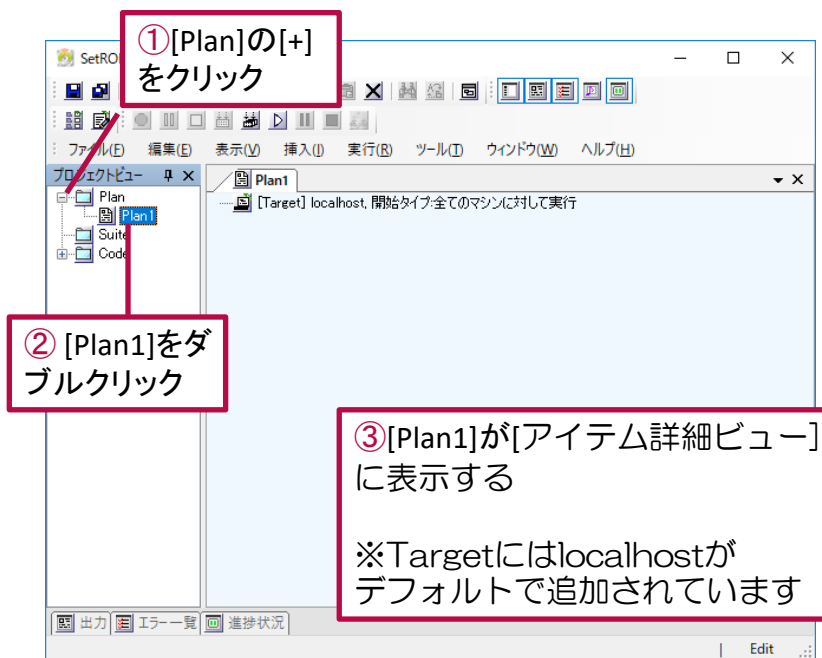
セットアップ用パッケージとは、作成したコードを実行するための実行形式ファイルです。

セットアップ用パッケージは、SetROBO for KittingがインストールされていないPCに配布して、ライセンス不要で実行できます。

### セットアップ用パッケージの出力

セットアップ用パッケージはPlan名のフォルダーに出力されます。

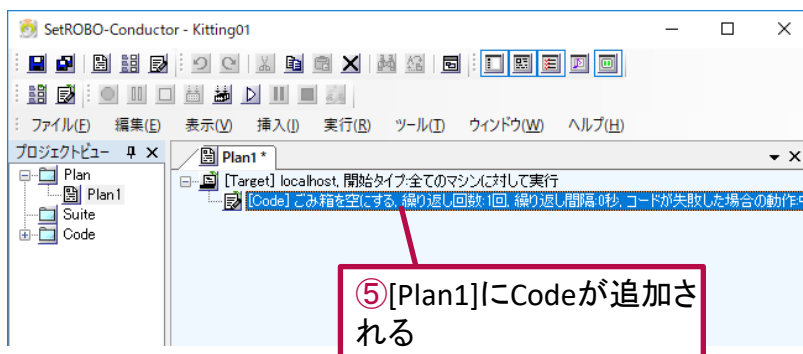
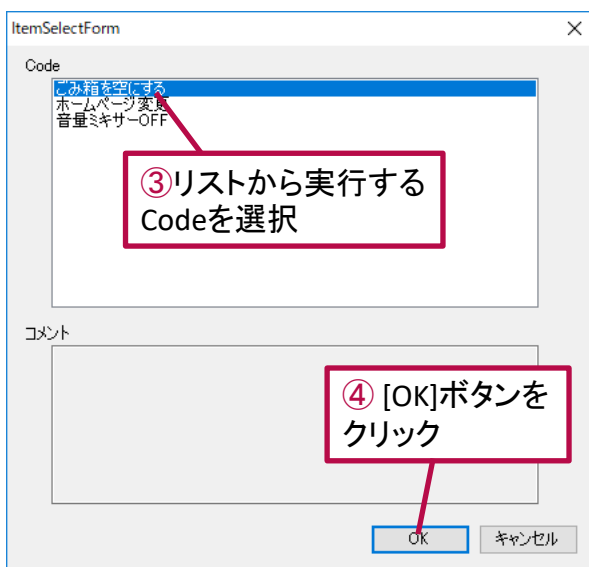
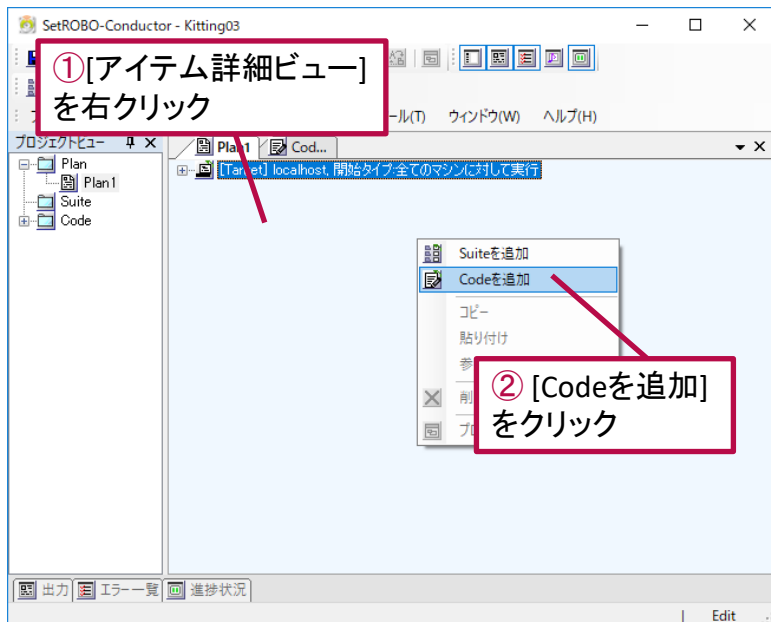
## 1 Plan1を開く



## 2 PlanにCodeを追加する

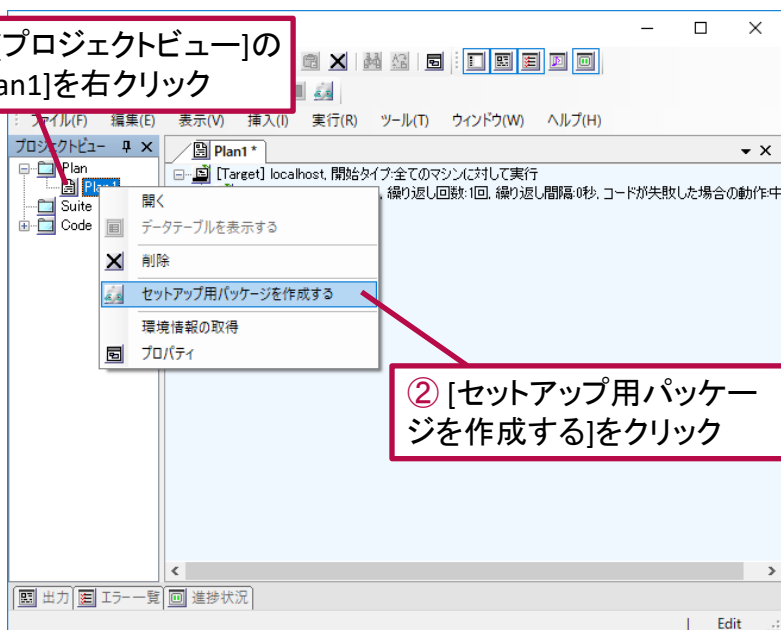
作成したCodeの中から実行したいものをPlanに追加します。

※下記方法以外にも、プロジェクトビューにある作成したCodeを選択し、ドラッグ&ドロップでPlanに追加することも可能です。

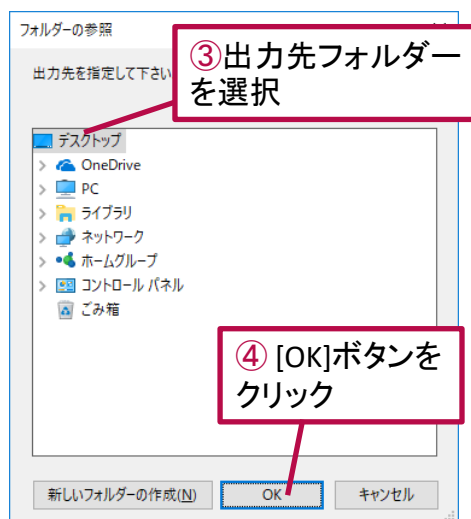


### 3 セットアップ用パッケージを作成する

① [プロジェクトビュー]の [Plan1]を右クリック

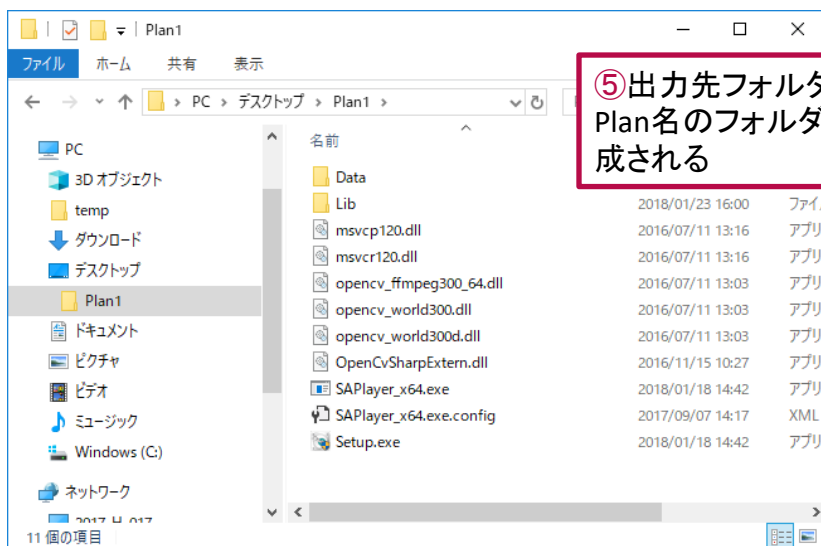


② [セットアップ用パッケージを作成する]をクリック



③ 出力先フォルダーを選択

④ [OK]ボタンをクリック



⑤ 出力先フォルダーに Plan名のフォルダーが作成される

## 4 セットアップ用パッケージを実行する

作成したセットアップ用パッケージフォルダーをUSBメモリなどの外部ストレージにコピーします。USBメモリをPCに接続し、セットアップ用パッケージフォルダー内にある「Setup.exe」を実行します。

### 👉 セットアップ用パッケージ使用時の注意

セットアップ用パッケージは、別のUSBメモリなどにコピーしても実行可能です。USBメモリを使用できないPCの場合は、ローカルハードディスクにコピーしても実行可能です。但し、CD-Rなどの書き込み不可能なメディアにコピーした時は実行することはできません。

シナリオ作成PC



セットアップ用パッケージフォルダー



USBメモリ



Target PC

実行!

Plan作成時に選択したCodeが順番に実行されます。

# 付録1 Windows11ショートカットコマンド一覧

	名前	実行コード	備考
ア行	iSCSI イニシエーターのプロパティ	PlayLib.ProcessStart("control", "/name Microsoft.iSCSIInitiator");	
	アカウント	PlayLib.ProcessStart("ms-settings:accounts");	
	アカウント情報	PlayLib.ProcessStart("ms-settings:privacy-accountinfo");	
	アクションセンター	PlayLib.ProcessStart("ms-actioncenter:editquickactions");	
	アクセシビリティ	PlayLib.ProcessStart("ms-settings:easeofaccess");	
	アクセス許可の検索	PlayLib.ProcessStart("ms-settings:search-permissions");	
	アクティビティの履歴	PlayLib.ProcessStart("ms-settings:privacy-activityhistory");	
	新しいコンテンツの保存先	PlayLib.ProcessStart("ms-settings:savelocations");	
	アプリと機能	PlayLib.ProcessStart("ms-settings:appsfeatures");	
	アプリの診断	PlayLib.ProcessStart("ms-settings:privacy-appdiagnostics");	
	イーサネット	PlayLib.ProcessStart("ms-settings:network-ethernet");	
	位置情報	PlayLib.ProcessStart("ms-settings:privacy-location");	
	イベントビューアー	PlayLib.ProcessStart("eventvwr");	
	色	PlayLib.ProcessStart("ms-settings:colors");	
	色の管理	PlayLib.ProcessStart("control", "/name Microsoft.ColorManagement");	
	インターネットのプロパティ	PlayLib.ProcessStart("inetcpl.cpl");	
	インデックスのオプション	PlayLib.ProcessStart("control", "/name Microsoft.IndexingOptions");	
	印刷の管理	PlayLib.ProcessStart("printmanagement.msc");	

名前	実行コード	備考
Windows Defender ファイアウォール	PlayLib.ProcessStart("Firewall.cpl");	
Windows Helloセットアップ(セキュリティキー認証)	PlayLib.ProcessStart("ms-settings:signinoptions-launchsecuritykeyenrollment");	
Windows Helloセットアップ(顔認証)	PlayLib.ProcessStart("ms-settings:signinoptions-launchfaceenrollment");	
Windows Helloセットアップ(指紋認証)	PlayLib.ProcessStart("ms-settings:signinoptions-launchfingerprintenrollment");	
Windows Insider Program	PlayLib.ProcessStart("ms-settings:windowsinsider");	
Windows Update	PlayLib.ProcessStart("ms-settings:windowsupdate");	
Windows セキュリティ	PlayLib.ProcessStart("ms-settings:windowsdefender");	
Windows バックアップ	PlayLib.ProcessStart("ms-settings:backup");	
Windows メモリ診断	PlayLib.ProcessStart("mdsched");	
Windows モビリティ センター	PlayLib.ProcessStart("mblctr");	
Windowsツール	PlayLib.ProcessStart("control", "admintools");	
Windowsの検索	PlayLib.ProcessStart("ms-settings:search");	
Web サイト用のアプリ	PlayLib.ProcessStart("ms-settings:appsforwebsites");	
エクスプローラーのオプション	PlayLib.ProcessStart("control", "folders");	
Xbox Game Bar	PlayLib.ProcessStart("ms-settings:gaming-gamebar");	
ODBC データ ソース アドミニストレーター	PlayLib.ProcessStart("control", "odbc32.cpl");	
オーディオ	PlayLib.ProcessStart("ms-settings:easeofaccess-audio");	
オフライン マップ	PlayLib.ProcessStart("ms-settings:maps");	
オプションの更新プログラムを表示	PlayLib.ProcessStart("ms-settings:windowsupdate-optionalupdates");	

名前	実行コード	備考
オプション機能	PlayLib.ProcessStart("ms-settings:optionalfeatures");	
音楽ライブラリ	PlayLib.ProcessStart("ms-settings:privacy-musiclibrary");	
音声によるアクティブ化	PlayLib.ProcessStart("ms-settings:privacy-voiceactivation");	
音声認識	PlayLib.ProcessStart("control", "/name Microsoft.SpeechRecognition");	
音声認識(アクセシビリティ)	PlayLib.ProcessStart( "ms-settings:easeofaccess-speechrecognition");	
音声認識(プライバシーとセキュリティ)	PlayLib.ProcessStart("ms-settings:privacy-speech");	
音声認識(時刻と言語)	PlayLib.ProcessStart("ms-settings:speech");	
音量ミキサー	PlayLib.ProcessStart("sndvol");	
音量ミキサー(サウンド)	PlayLib.ProcessStart("ms-settings:apps-volume");	
開発者向け	PlayLib.ProcessStart("ms-settings:developers");	
回復	PlayLib.ProcessStart("control", "/name Microsoft.Recovery");	
回復(システム)	PlayLib.ProcessStart("ms-settings:recovery");	
拡大鏡	PlayLib.ProcessStart("ms-settings:easeofaccess-magnifier");	
家族とその他のユーザー	PlayLib.ProcessStart("ms-settings:otherusers");	
カメラ(Bluetoothとデバイス)	PlayLib.ProcessStart("ms-settings:camera");	
カメラ(プライバシーとセキュリティ)	PlayLib.ProcessStart("ms-settings:privacy-webcam");	
カラー フィルター	PlayLib.ProcessStart("ms-settings:easeofaccess-colorfilter");	
カレンダー	PlayLib.ProcessStart("ms-settings:privacy-calendar");	
キーボード	PlayLib.ProcessStart("ms-settings:easeofaccess-keyboard");	

カ行

名前	実行コード	備考
キーボードのプロパティ	PlayLib.ProcessStart("control", "/name Microsoft.Keyboard");	
キオスク	PlayLib.ProcessStart("ms-settings:assignedaccess");	
記憶域	PlayLib.ProcessStart("control", "/name Microsoft.StorageSpaces");	
記憶域(システム)	PlayLib.ProcessStart("ms-settings:storagesense");	
既定のアプリ	PlayLib.ProcessStart("ms-settings:defaultapps");	
既定のプログラム	PlayLib.ProcessStart("control", "/name Microsoft.DefaultPrograms");	
再起動のスケジュール	PlayLib.ProcessStart("ms-settings:windowsupdate-restartoptions");	
キャプチャ	PlayLib.ProcessStart("ms-settings:gaming-gamedvr");	
グラフィック	PlayLib.ProcessStart("ms-settings:display-advancedgraphics");	
クリップボード	PlayLib.ProcessStart("ms-settings:clipboard");	
ゲーム モード	PlayLib.ProcessStart("ms-settings:gaming-gamemode");	
言語と地域	PlayLib.ProcessStart("ms-settings:regionlanguage");	
更新の履歴	PlayLib.ProcessStart("ms-settings:windowsupdate-history");	
高度なオプション(その他のMicrosoft 製品の更新プログラムを受け取る)	PlayLib.ProcessStart("ms-settings:windowsupdate-activehours");	
個人用設定	PlayLib.ProcessStart("ms-settings:personalization");	
このPCへのプロジェクション	PlayLib.ProcessStart("ms-settings:project");	
コマンドプロンプト	PlayLib.ProcessStart("cmd");	
ゴミ箱	PlayLib.ProcessStart("Explorer", "::[645FF040-5081-101B-9F08-00AA002F954E]");	
コントラスト テーマ	PlayLib.ProcessStart("ms-settings:easeofaccess-highcontrast");	

サ行

名前	実行コード	備考
コンピューターの管理	PlayLib.ProcessStart("compmgmt.msc", "/s");	
コンピューターの簡単操作センター	PlayLib.ProcessStart("control", "/name Microsoft.EaseOfAccessCenter");	
コンポーネント サービス	PlayLib.ProcessStart("comexp.msc");	
サービス	PlayLib.ProcessStart("services.msc");	
サインイン オプション	PlayLib.ProcessStart("ms-settings:signinoptions");	
サインイン オプション(動的ロック)	PlayLib.ProcessStart("ms-settings:signinoptions-dynamiclock");	
サウンド	PlayLib.ProcessStart("mmsys.cpl");	
サウンド(システム)	PlayLib.ProcessStart("ms-settings:sound");	
視覚効果	PlayLib.ProcessStart("ms-settings:easeofaccess-visualeffects");	
資格情報マネージャー	PlayLib.ProcessStart("control", "/name Microsoft.CredentialManager");	
システム	PlayLib.ProcessStart("ms-settings:");	
システムのプロパティ	PlayLib.ProcessStart("sysdm.cpl");	
システムの復元	PlayLib.ProcessStart("rstrui");	
システム構成	PlayLib.ProcessStart("msconfig");	
視線制御	PlayLib.ProcessStart("ms-settings:easeofaccess-eyecontrol");	
自動再生	PlayLib.ProcessStart("control", "/name Microsoft.AutoPlay");	
自動再生(Bluetoothとデバイス)	PlayLib.ProcessStart("ms-settings:autoplay");	
字幕	PlayLib.ProcessStart("ms-settings:easeofaccess-closedcaptioning");	
集中モード(ゲームをプレイしているとき)	PlayLib.ProcessStart("ms-settings:quietmomentsgame");	

名前	実行コード	備考
集中モード(ディスプレイを複製しているとき)	PlayLib.ProcessStart("ms-settings:quietmomentspresentation");	
集中モード(次の時間帯)	PlayLib.ProcessStart("ms-settings:quietmomentsscheduled");	
集中モード	PlayLib.ProcessStart("ms-settings:quiethours");	
詳細オプション	PlayLib.ProcessStart("ms-settings:windowsupdate-options");	
職場または学校にアクセスする	PlayLib.ProcessStart("ms-settings:workplace");	
所在地情報	PlayLib.ProcessStart("telephon.cpl");	
診断 & フィードバック	PlayLib.ProcessStart("ms-settings:privacy-feedback");	
診断 & フィードバック(診断データを表示する)	PlayLib.ProcessStart("ms-settings:privacy-feedback-telemetryviewer");	
スクリーンショットとアプリ	PlayLib.ProcessStart("ms-settings:privacy-graphicscaptureprogrammatic");	
スクリーンショットの境界線	PlayLib.ProcessStart("ms-settings:privacy-graphicscapturewithoutborder");	
スクリーンセーバーの設定	PlayLib.ProcessStart("control", "desk.cpl,1");	
スタート	PlayLib.ProcessStart("ms-settings:personalization-start");	
スタートアップ	PlayLib.ProcessStart("ms-settings:startupapps");	
ストレージ センサー	PlayLib.ProcessStart("ms-settings:storagepolicies");	
すべてのサウンド デバイス	PlayLib.ProcessStart("ms-settings:sound-devices");	
スマホ同期	PlayLib.ProcessStart("ms-settings:mobile-devices");	
セキュリティが強化された Windows Defender ファイアウォール	PlayLib.ProcessStart("WF.msc");	
セキュリティとメンテナンス	PlayLib.ProcessStart("wscui.cpl");	
Set up a new connection	PlayLib.ProcessStart("rasphone", "-a");	

	名前	実行コード	備考
	全般	PlayLib.ProcessStart("ms-settings:privacy-general");	
タ行	ダイヤルアップ	PlayLib.ProcessStart("ms-settings:network-dialup");	
	ダウンロード フォルダー	PlayLib.ProcessStart("ms-settings:privacy-downloadsfolder");	
	タスク	PlayLib.ProcessStart("ms-settings:privacy-tasks");	
	タスク バー	PlayLib.ProcessStart("ms-settings:taskbar");	
	タスクスケジューラー	PlayLib.ProcessStart("Taskschd.msc");	
	タッチ	PlayLib.ProcessStart("ms-settings:devices-touch");	
	タッチ キーボード	PlayLib.ProcessStart("ms-settings:personalization-touchkeyboard");	
	地域	PlayLib.ProcessStart("intl.cpl");	
	地図のダウンロード	PlayLib.ProcessStart("ms-settings:maps-downloadmaps");	
	通知(システム)	PlayLib.ProcessStart("ms-settings:notifications");	
	通知(プライバシーとセキュリティ)	PlayLib.ProcessStart("ms-settings:privacy-notifications");	
	通話履歴	PlayLib.ProcessStart("ms-settings:privacy-callhistory");	
	ディスク クリーンアップ-(C:)	PlayLib.ProcessStart("cleanmgr");	
	ディスクとボリューム	PlayLib.ProcessStart("ms-settings:disksandvolumes");	
	ディスプレイ	PlayLib.ProcessStart("ms-settings:display");	
	テーマ	PlayLib.ProcessStart("ms-settings:themes");	
	手書き入力と入力の個人用設定	PlayLib.ProcessStart("ms-settings:privacy-speechtyping");	
	テキスト カーソル	PlayLib.ProcessStart("ms-settings:easeofaccess-cursor");	

名前	実行コード	備考
テキストのサイズ	PlayLib.ProcessStart("ms-settings:easeofaccess-display");	
デバイス	PlayLib.ProcessStart("ms-settings:bluetooth");	
デバイス マネージャー	PlayLib.ProcessStart("devmgmt.msc");	
デバイスとプリンター	PlayLib.ProcessStart("control", "printers");	
デバイスの検索	PlayLib.ProcessStart("ms-settings:findmydevice");	
デバイスの使用状況	PlayLib.ProcessStart("ms-settings:deviceusage");	
電源	PlayLib.ProcessStart("ms-settings:batterysaver");	
電源オプション	PlayLib.ProcessStart("powercfg.cpl");	
電話をかける	PlayLib.ProcessStart("ms-settings:privacy-phonecalls");	
同期センター	PlayLib.ProcessStart("mobsync");	
ドキュメント	PlayLib.ProcessStart("ms-settings:privacy-documents");	
トラブルシューティング	PlayLib.ProcessStart("ms-settings:troubleshoot");	
ナレーター	PlayLib.ProcessStart("ms-settings:easeofaccess-narrator");	
ナレーター (サインイン後にナレーターを開始する)	PlayLib.ProcessStart("ms-settings:easeofaccess-narrator-isautostartenabled");	
入力	PlayLib.ProcessStart("ms-settings:typing");	
ネットワークとインターネット	PlayLib.ProcessStart("ms-settings:network");	
ネットワークと共有センター	PlayLib.ProcessStart("control", "/name Microsoft.NetworkAndSharingCenter");	
ネットワークの詳細設定	PlayLib.ProcessStart("ms-settings:network-advancedsettings");	
ネットワーク接続	PlayLib.ProcessStart("ncpa.cpl");	

ナ行

	名前	実行コード	備考
8行	バージョン情報	PlayLib.ProcessStart("ms-settings:about");	
	背景	PlayLib.ProcessStart("ms-settings:personalization-background");	
	配信の最適化	PlayLib.ProcessStart("ms-settings:delivery-optimization");	
	配信の最適化のアクティビティ モニター	PlayLib.ProcessStart("ms-settings:delivery-optimization-activity");	
	配信の最適化の詳細オプション	PlayLib.ProcessStart("ms-settings:delivery-optimization-advanced");	
	パフォーマンス モニター	PlayLib.ProcessStart("perfmon.msc");	
	BitLocker ドライブ暗号化	PlayLib.ProcessStart("control", "/name Microsoft.BitLockerDriveEncryption");	
	ピクチャ	PlayLib.ProcessStart("ms-settings:privacy-pictures");	
	日付と時刻	PlayLib.ProcessStart("timedate.cpl");	
	日付と時刻(時刻と言語)	PlayLib.ProcessStart("ms-settings:dateandtime");	
	ビデオ	PlayLib.ProcessStart("ms-settings:privacy-videos");	
	ビデオの再生	PlayLib.ProcessStart("ms-settings:videoplayback");	
	ファイル システム	PlayLib.ProcessStart("ms-settings:privacy-broadfilesystemaccess");	
	ファイルの自動ダウンロード	PlayLib.ProcessStart("ms-settings:privacy-automaticfiledownloads");	
	ファイル履歴	PlayLib.ProcessStart("control", "/name Microsoft.FileHistory");	
	VPN	PlayLib.ProcessStart("ms-settings:network-vpn");	
	フォルダー	PlayLib.ProcessStart("ms-settings:personalization-start-places");	
	フォント	PlayLib.ProcessStart("control", "/name Microsoft.Fonts");	
	フォント(個人用設定)	PlayLib.ProcessStart("ms-settings:fonts");	

	名前	実行コード	備考
	プライバシーとセキュリティ	PlayLib.ProcessStart("ms-settings:privacy");	
	プリンターとスキャナー	PlayLib.ProcessStart("ms-settings:printers");	
	プロキシ	PlayLib.ProcessStart("ms-settings:network-proxy");	
	プログラムと機能	PlayLib.ProcessStart("appwiz.cpl");	
	ペンとWindows Ink	PlayLib.ProcessStart("ms-settings:pen");	
	他のデバイス	PlayLib.ProcessStart("ms-settings:privacy-customdevices");	
マ行	Microsoft IME	PlayLib.ProcessStart("ms-settings:regionlanguage-jpnime");	
	マイク	PlayLib.ProcessStart("ms-settings:privacy-microphone");	
	マウス ポインターとタッチ	PlayLib.ProcessStart("ms-settings:easeofaccess-mousepointer");	
	マウス(Bluetoothとデバイス)	PlayLib.ProcessStart("ms-settings:mousetouchpad");	
	マウス(アクセシビリティ)	PlayLib.ProcessStart("ms-settings:easeofaccess-mouse");	
	マウスのプロパティ	PlayLib.ProcessStart("main.cpl");	
	マルチタスク	PlayLib.ProcessStart("ms-settings:multitasking");	
	無線	PlayLib.ProcessStart("ms-settings:privacy-radios");	
	メール	PlayLib.ProcessStart("ms-settings:privacy-email");	
	メールとアカウント	PlayLib.ProcessStart("ms-settings:emailandaccounts");	
	メッセージング	PlayLib.ProcessStart("ms-settings:privacy-messaging");	
ヤ行	夜間モード	PlayLib.ProcessStart("ms-settings:nightlight");	
	ユーザーアカウント	PlayLib.ProcessStart("control", "userpasswords2");	

	名前	実行コード	備考
	ユーザーアカウント(コントロールパネル)	PlayLib.ProcessStart("control", "/name Microsoft.UserAccounts");	
	ユーザーアカウント制御の設定	PlayLib.ProcessStart("UserAccountControlSettings");	
	ユーザーの情報	PlayLib.ProcessStart("ms-settings:yourinfo");	
	USB	PlayLib.ProcessStart("ms-settings:usb");	
	[要求に応じて表示]ページを表示する	PlayLib.ProcessStart("ms-settings:windowsupdate-seekerondemand");	
ラ行	ライセンス認証	PlayLib.ProcessStart("ms-settings:activation");	
	リソースモニター	PlayLib.ProcessStart("resmon");	
	リモート デスクトップ	PlayLib.ProcessStart("ms-settings:remotedesktop");	
	RemoteApp とデスクトップ接続	PlayLib.ProcessStart("control", "/name Microsoft.RemoteAppAndDesktopConnections");	
	連絡先	PlayLib.ProcessStart("ms-settings:privacy-contacts");	
	ローカル セキュリティ ポリシー	PlayLib.ProcessStart("secpol.msc");	
	ロック画面	PlayLib.ProcessStart("ms-settings:lockscreen");	
ワ行	ワーク フォルダー	PlayLib.ProcessStart("control", "/name Microsoft.WorkFolders");	

## 付録2 Windows10ショートカットコマンド一覧

	名前	実行コード	備考
ア行	iSCSI イニシエーターのプロパティ	PlayLib.ProcessStart("control", "/name Microsoft.iSCSIInitiator");	
	アカウント情報	PlayLib.ProcessStart("ms-settings:privacy-accountinfo");	
	アクションセンター	PlayLib.ProcessStart("ms-actioncenter:editquickactions");	
	アクセス許可	PlayLib.ProcessStart("ms-settings:cortana-permissions");	
	アクセス許可と履歴	PlayLib.ProcessStart("ms-settings:search-permissions");	
	アクティビティの履歴	PlayLib.ProcessStart("ms-settings:privacy-activityhistory");	
	新しいコンテンツの保存先を変更する	PlayLib.ProcessStart("ms-settings:savelocations");	
	アプリと機能	PlayLib.ProcessStart("ms-settings:appsfeatures");	
	アプリとブラウザー コントロール	PlayLib.ProcessStart("windowsdefender://appbrowser/");	
	アプリの音量とデバイスの設定	PlayLib.ProcessStart("ms-settings:apps-volume");	
	アプリの診断	PlayLib.ProcessStart("ms-settings:privacy-appdiagnostics");	
	位置情報	PlayLib.ProcessStart("ms-settings:privacy-location");	
	色	PlayLib.ProcessStart("ms-settings:personalization-colors");	
	色の管理	PlayLib.ProcessStart("control", "/name Microsoft.ColorManagement");	
	イベントビューアー	PlayLib.ProcessStart("eventvwr");	
	印刷の管理	PlayLib.ProcessStart("printmanagement.msc");	
	インターネットのプロパティ	PlayLib.ProcessStart("inetcpl.cpl");	
	インデックスのオプション	PlayLib.ProcessStart("control", "/name Microsoft.IndexingOptions");	

名前	実行コード	備考
イーサネット	PlayLib.ProcessStart("ms-settings:network-ethernet");	
ウイルスと脅威の防止	PlayLib.ProcessStart("windowsdefender://threat/");	
Windows Update	PlayLib.ProcessStart("ms-settings:windowsupdate");	
Windows Update -アクティブ時間の変更	PlayLib.ProcessStart("ms-settings:windowsupdate-activehours");	
Windows Update -更新の履歴を表示する	PlayLib.ProcessStart("ms-settings:windowsupdate-history");	
Windows Update -更新プログラムのチェック	PlayLib.ProcessStart("ms-settings:windowsupdate-action");	
Windows Update -再起動のオプション	PlayLib.ProcessStart("ms-settings:windowsupdate-restartoptions");	
Windows Update -詳細オプション	PlayLib.ProcessStart("ms-settings:windowsupdate-options");	
Windows Insider Program	PlayLib.ProcessStart("ms-settings:windowsinsider");	
Windows セキュリティ	PlayLib.ProcessStart("ms-settings:windowsdefender");	
Windows Defender ファイアウォール	PlayLib.ProcessStart("Firewall.cpl");	
Windows の検索	PlayLib.ProcessStart("ms-settings:cortana-windowssearch");	
Windows Hello セットアップ (顔認証)	PlayLib.ProcessStart("ms-settings:signinoptions-launchfaceenrollment");	
Windows メモリ診断	PlayLib.ProcessStart("mdsched");	
Windows モビリティ センター	PlayLib.ProcessStart("mblctr");	
Web サイト用のアプリ	PlayLib.ProcessStart("ms-settings:appsforwebsites");	
エクスプローラーのオプション	PlayLib.ProcessStart("control", "folders");	
Xbox ネットワーク	PlayLib.ProcessStart("ms-settings:gaming-xboxnetworking");	
オフライン マップ	PlayLib.ProcessStart("ms-settings:maps");	

名前	実行コード	備考
オプション機能	PlayLib.ProcessStart("ms-settings:optionalfeatures");	
音声によるアクティブ化	PlayLib.ProcessStart("ms-settings:privacy-voiceactivation");	
音声認識	PlayLib.ProcessStart("control", "/name Microsoft.SpeechRecognition");	
音声認識(簡単操作)	PlayLib.ProcessStart( "ms-settings:easeofaccess-speechrecognition");	
音声認識(時刻と言語)	PlayLib.ProcessStart("ms-settings:speech");	
音声認識(プライバシー)	PlayLib.ProcessStart("ms-settings:privacy-speech");	
音量ミキサー	PlayLib.ProcessStart("sndvol");	
オーディオ	PlayLib.ProcessStart("ms-settings:easeofaccess-audio");	
ODBC データ ソース アドミニストレーター	PlayLib.ProcessStart("control", "odbc32.cpl");	
開発者向け	PlayLib.ProcessStart("ms-settings:developers");	
回復	PlayLib.ProcessStart("control", "/name Microsoft.Recovery");	
回復(更新とセキュリティ)	PlayLib.ProcessStart("ms-settings:recovery");	
拡大鏡	PlayLib.ProcessStart("ms-settings:easeofaccess-magnifier");	
家族とその他のユーザー	PlayLib.ProcessStart("ms-settings:otherusers");	
カメラ	PlayLib.ProcessStart("ms-settings:privacy-webcam");	
カラー フィルター	PlayLib.ProcessStart("ms-settings:easeofaccess-colorfilter");	
カレンダー	PlayLib.ProcessStart("ms-settings:privacy-calendar");	
管理ツール	PlayLib.ProcessStart("control", "admintools");	
カーソルとポインター	PlayLib.ProcessStart( "ms-settings:easeofaccess-cursorandpointersize");	

カ行

名前	実行コード	備考
記憶域	PlayLib.ProcessStart("control", "/name Microsoft.StorageSpaces");	
キオスク モードを設定する	PlayLib.ProcessStart("ms-settings:assignedaccess");	
既定のアプリ	PlayLib.ProcessStart("ms-settings:defaultapps");	
既定のプログラム	PlayLib.ProcessStart("control", "/name Microsoft.DefaultPrograms");	
機内モード	PlayLib.ProcessStart("ms-settings:network-airplanemode");	
キャプチャ	PlayLib.ProcessStart("ms-settings:gaming-gamedvr");	
共有エクスペリエンス	PlayLib.ProcessStart("ms-settings:crossdevice");	
キーボード	PlayLib.ProcessStart("ms-settings:easeofaccess-keyboard");	
キーボードのプロパティ	PlayLib.ProcessStart("control", "/name Microsoft.Keyboard");	
クリップボード	PlayLib.ProcessStart("ms-settings:clipboard");	
グラフィックの設定	PlayLib.ProcessStart("ms-settings:display-advancedgraphics");	
言語	PlayLib.ProcessStart("ms-settings:regionlanguage");	
ゲーム モード	PlayLib.ProcessStart("ms-settings:gaming-gamemode");	
ゲームを全画面表示で プレイしているとき	PlayLib.ProcessStart("ms-settings:quietmomentsgame");	
ゲーム バー	PlayLib.ProcessStart("ms-settings:gaming-gamebar");	
この PC へのプロジェクション	PlayLib.ProcessStart("ms-settings:project");	
コマンドプロンプト	PlayLib.ProcessStart("cmd");	
Cortana に話しかける	PlayLib.ProcessStart("ms-settings:cortana-talktocortana");	
コンピューターの 簡単操作センター	PlayLib.ProcessStart("control", "/name Microsoft.EaseOfAccessCenter");	

	名前	実行コード	備考
	コンピューターの管理	PlayLib.ProcessStart("compmgmt.msc", "/s");	
	コンポーネント サービス	PlayLib.ProcessStart("comexp.msc");	
	ごみ箱	PlayLib.ProcessStart("Explorer", ":{645FF040-5081-101B-9F08-00AA002F954E}");	
サ行	サインイン オプション	PlayLib.ProcessStart("ms-settings:signinoptions");	
	サウンド	PlayLib.ProcessStart("mmsys.cpl");	
	サウンド(システム)	PlayLib.ProcessStart("ms-settings:sound");	
	サウンド デバイスを管理する	PlayLib.ProcessStart("ms-settings:sound-devices");	
	サービス	PlayLib.ProcessStart("services.msc");	
	資格情報マネージャー	PlayLib.ProcessStart("control", "/name Microsoft.CredentialManager");	
	システム	PlayLib.ProcessStart("control", "/name Microsoft.System");	
	システム構成	PlayLib.ProcessStart("msconfig");	
	システムの復元	PlayLib.ProcessStart("rstrui");	
	システムのプロパティ	PlayLib.ProcessStart("sysdm.cpl");	
	視線制御	PlayLib.ProcessStart("ms-settings:easeofaccess-eyecontrol");	
	集中モード	PlayLib.ProcessStart("ms-settings:quietmomentshome");	
	詳細	PlayLib.ProcessStart("ms-settings:search-moredetails");	
	詳細情報	PlayLib.ProcessStart("ms-settings:cortana-moredetails");	
	職場または学校にアクセスする	PlayLib.ProcessStart("ms-settings:workplace");	
	所在地情報	PlayLib.ProcessStart("telephon.cpl");	

名前	実行コード	備考
診断 & フィードバック	PlayLib.ProcessStart("ms-settings:privacy-feedback");	
自動再生	PlayLib.ProcessStart("control", "/name Microsoft.AutoPlay");	
自動再生(デバイス)	PlayLib.ProcessStart("ms-settings:autoplay");	
字幕	PlayLib.ProcessStart("ms-settings:easeofaccess-closedcaptioning");	
状態	PlayLib.ProcessStart("ms-settings:network-status");	
スクリーン セーバーの設定	PlayLib.ProcessStart("control", "desk.cpl,1");	
スタート	PlayLib.ProcessStart("ms-settings:personalization-start");	
スタートアップ	PlayLib.ProcessStart("ms-settings:startupapps");	
スタートメニューに 表示するフォルダーを選ぶ	PlayLib.ProcessStart("ms-settings:personalization-start-places");	
ストレージ	PlayLib.ProcessStart("ms-settings:storagesense");	
ストレージ センサーを 構成するか、今すぐ実行する	PlayLib.ProcessStart("ms-settings:storagepolicies");	
スマホ同期	PlayLib.ProcessStart("ms-settings:mobile-devices-addphone-direct");	
セキュリティが強化された Windows Defender ファイアウォール	PlayLib.ProcessStart("WF.msc");	
セキュリティとメンテナンス	PlayLib.ProcessStart("wscui.cpl");	
セキュリティの概要	PlayLib.ProcessStart("windowsdefender:");	
セキュリティ プロバイダー	PlayLib.ProcessStart("windowsdefender://providers/");	
設定の同期	PlayLib.ProcessStart("ms-settings:sync");	
Set up a new connection (ダイヤルアップネットワーク)	PlayLib.ProcessStart("rasphone", "-a");	
全般	PlayLib.ProcessStart("ms-settings:privacy-general");	

各行

名前	実行コード	備考
タスク	PlayLib.ProcessStart("ms-settings:privacy-tasks");	
タスク スケジューラ	PlayLib.ProcessStart("Taskschd.msc");	
タスク バー	PlayLib.ProcessStart("ms-settings:taskbar");	
タッチパッド	PlayLib.ProcessStart("ms-settings:devices-touchpad");	
他のデバイス	PlayLib.ProcessStart("ms-settings:privacy-customdevices");	
タブレット モード	PlayLib.ProcessStart("ms-settings:tabletmode");	
ダイヤルアップ	PlayLib.ProcessStart("ms-settings:network-dialup");	
地域	PlayLib.ProcessStart("intl.cpl");	
地域(時刻と言語)	PlayLib.ProcessStart("ms-settings:regionformatting");	
通知	PlayLib.ProcessStart("ms-settings:privacy-notifications");	
通知とアクション	PlayLib.ProcessStart("ms-settings:notifications");	
通話履歴	PlayLib.ProcessStart("ms-settings:privacy-callhistory");	
次の時間帯	PlayLib.ProcessStart("ms-settings:quietmomentsscheduled");	
手書き入力と入力の個人用設定	PlayLib.ProcessStart("ms-settings:privacy-speechtyping");	
テーマ	PlayLib.ProcessStart("ms-settings:themes");	
ディスクのクリーンアップ : ドライブの選択	PlayLib.ProcessStart("cleanmgr");	
ディスプレイ(簡単操作)	PlayLib.ProcessStart("ms-settings:easeofaccess-display");	
ディスプレイ(システム)	PlayLib.ProcessStart("ms-settings:display");	
ディスプレイを複製しているとき	PlayLib.ProcessStart("ms-settings:quietmomentspresentation");	

名前	実行コード	備考
デバイス セキュリティ	PlayLib.ProcessStart(" windowsdefender://hardware/");	
デバイスとプリンター	PlayLib.ProcessStart(" control", "printers");	
デバイスの暗号化	PlayLib.ProcessStart(" ms-settings:deviceencryption");	
デバイスの検索	PlayLib.ProcessStart(" ms-settings:findmydevice");	
デバイスのパフォーマンスと 正常性	PlayLib.ProcessStart(" windowsdefender://perfhealth/");	
デバイス マネージャー	PlayLib.ProcessStart(" devmgmt.msc");	
電源オプション	PlayLib.ProcessStart(" powercfg.cpl");	
電源とスリープ	PlayLib.ProcessStart(" ms-settings:powersleep");	
電話	PlayLib.ProcessStart(" ms-settings:mobile-devices");	
電話をかける	PlayLib.ProcessStart(" ms-settings:privacy-phonecalls");	
データ使用状況	PlayLib.ProcessStart(" ms-settings:datausage");	
トラブルシューティング	PlayLib.ProcessStart(" ms-settings:troubleshoot");	
同期センター	PlayLib.ProcessStart(" mobsync");	
ドキュメント	PlayLib.ProcessStart(" ms-settings:privacy-documents");	
ナレーター	PlayLib.ProcessStart(" ms-settings:easeofaccess-narrator");	
入力	PlayLib.ProcessStart(" ms-settings:typing");	
ネットワーク接続	PlayLib.ProcessStart(" ncpa.cpl");	
ネットワークと共有センター	PlayLib.ProcessStart(" control", "/name Microsoft.NetworkAndSharingCenter");	
背景	PlayLib.ProcessStart(" ms-settings:personalization-background");	

ナ行

ハ行

名前	実行コード	備考
ハイコントラスト	PlayLib.ProcessStart("ms-settings:easeofaccess-highcontrast");	
配信の最適化	PlayLib.ProcessStart("ms-settings:delivery-optimization");	
バックアップ	PlayLib.ProcessStart("ms-settings:backup");	
バックグラウンド アプリ	PlayLib.ProcessStart("ms-settings:privacy-backgroundapps");	
バッテリー	PlayLib.ProcessStart("ms-settings:batterysaver");	
バッテリー残量に影響を及ぼしているアプリを表示する	PlayLib.ProcessStart("ms-settings:batterysaver-usedetails");	
バージョン情報	PlayLib.ProcessStart("ms-settings:about");	
パフォーマンス モニター	PlayLib.ProcessStart("perfmon.msc");	
日付と時刻	PlayLib.ProcessStart("timedate.cpl");	
日付と時刻(時刻と言語)	PlayLib.ProcessStart("ms-settings:dateandtime");	
表示スケールの詳細設定	PlayLib.ProcessStart("ms-settings:display-advanced");	
BitLocker ドライブ暗号化	PlayLib.ProcessStart("control", "/name Microsoft.BitLockerDriveEncryption");	
ビデオ	PlayLib.ProcessStart("ms-settings:privacy-videos");	
ビデオの再生	PlayLib.ProcessStart("ms-settings:videoplayback");	
ピクチャ	PlayLib.ProcessStart("ms-settings:privacy-pictures");	
ファイアウォールとネットワーク保護	PlayLib.ProcessStart("windowsdefender://network/");	
ファイル システム	PlayLib.ProcessStart("ms-settings:privacy-broadfilesystemaccess");	
ファイルの自動ダウンロード	PlayLib.ProcessStart("ms-settings:privacy-automaticfiledownloads");	
ファイル履歴	PlayLib.ProcessStart("control", "/name Microsoft.FileHistory");	

名前	実行コード	備考
ファミリーのオプション	PlayLib.ProcessStart(" windowsdefender://family/");	
フォント	PlayLib.ProcessStart(" control", "/name Microsoft.Fonts");	
フォント(個人用設定)	PlayLib.ProcessStart(" ms-settings:fonts");	
VPN	PlayLib.ProcessStart(" ms-settings:network-vpn");	
Bluetooth とその他のデバイス	PlayLib.ProcessStart(" ms-settings:bluetooth");	
プリンターとスキャナー	PlayLib.ProcessStart(" ms-settings:printers");	
プロキシ	PlayLib.ProcessStart(" ms-settings:network-proxy");	
プログラムと機能	PlayLib.ProcessStart(" appwiz.cpl");	
ペンと Windows Ink	PlayLib.ProcessStart(" ms-settings:pen");	
マイク	PlayLib.ProcessStart(" ms-settings:privacy-microphone");	
Microsoft IME	PlayLib.ProcessStart(" ms-settings:regionlanguage-jpnime");	
マウス(簡単操作)	PlayLib.ProcessStart(" ms-settings:easeofaccess-mouse");	
マウス(デバイス)	PlayLib.ProcessStart(" ms-settings:mousetouchpad");	
マウスのプロパティ	PlayLib.ProcessStart(" main.cpl");	
マルチタスク	PlayLib.ProcessStart(" ms-settings:multitasking");	
無線	PlayLib.ProcessStart(" ms-settings:privacy-radios");	
メッセージング	PlayLib.ProcessStart(" ms-settings:privacy-messaging");	
メール	PlayLib.ProcessStart(" ms-settings:privacy-email");	
メールとアカウント	PlayLib.ProcessStart(" ms-settings:emailandaccounts");	

マ行

	名前	実行コード	備考
	モバイル ホットスポット	PlayLib.ProcessStart("ms-settings:network-mobilehotspot");	
ヤ行	夜間モードの設定	PlayLib.ProcessStart("ms-settings:nightlight");	
	USB	PlayLib.ProcessStart("ms-settings:usb");	
	ユーザー アカウント	PlayLib.ProcessStart("control", "/name Microsoft.UserAccounts");	
	ユーザー アカウント (自動ログオン設定)	PlayLib.ProcessStart("control", "userpasswords2");	
	ユーザー アカウント制御の設定	PlayLib.ProcessStart("UserAccountControlSettings");	
	ユーザーの情報	PlayLib.ProcessStart("ms-settings:yourinfo");	
ラ行	ライセンス認証	PlayLib.ProcessStart("ms-settings:activation");	
	リソース モニター	PlayLib.ProcessStart("resmon");	
	RemoteApp とデスクトップ接続	PlayLib.ProcessStart("control", "/name Microsoft.RemoteAppAndDesktopConnections");	
	リモート デスクトップ	PlayLib.ProcessStart("ms-settings:remotedesktop");	
	連絡先	PlayLib.ProcessStart("ms-settings:privacy-contacts");	
	ロック画面	PlayLib.ProcessStart("ms-settings:lockscreen");	
	ローカル セキュリティ ポリシー	PlayLib.ProcessStart("secpol.msc");	
ワ行	Wi-Fi	PlayLib.ProcessStart("ms-settings:network-wifi");	
	ワーク フォルダー	PlayLib.ProcessStart("control", "/name Microsoft.WorkFolders");	

## 付録3 画像比較用ライブラリについて

画像比較ライブラリの名称は、「ImageLibrary.dll」といいます。

「ImageLibrary.dll」は、SetROBO内部に組み込まれたライブラリであり、画像を検索し一致した画像に対してアクション（操作）を行うためのライブラリです。

あらかじめ、検索元画像（検索対象の画像）を用意しておき、画像比較ライブラリを実行すると検索元画像と同じ画像を画面から検索し、一致した場合は、一致した画像の中心座標に対して指定したアクションを行います。  
一致しなかった場合は、エラーとなりその旨を返します。

また、画像検索の際の画像一致率を変更できるため、背景が微妙に変わるなど画像検索ならではの問題をクリアすることができます。

「ImageLibrary.dll」は、画像ベースで操作対象のコントロールの操作を行うことができるので、シナリオコード作成の際に指定したコントロールの種類・コントロール名を気にすることなくシナリオコードを作成することができ、対象コントロールの操作や正解・不正解などの判定を行うことができます。

### ●イメージ

検索元画像



検索対象画面



# 付録4 画像比較用ライブラリのメソッド一覧

## ①Search

指定した画像と合致する箇所を検索し、見つかった場合にはTRUE、見つからなかった場合にはFALSEを返します。

- public static bool Search(string image\_path)
- public static bool Search(string window\_name, string image\_path)
- public static bool Search(string window\_name, string image\_path, int retry\_count)
- public static bool Search(string window\_name, string image\_path, int retry\_count, ImageLibrary.SearchArea search\_area, double match\_rate\_threshold)

### ○各パラメータの説明

window\_name: ウィンドウ名

ウィンドウ名を指定すると、そのウィンドウの範囲内でのみ検索します。

全画面検索を行う場合は、「String.Empty」を指定してください。

※ウィンドウ名は、基本的に全画面検索「String.Empty」を指定することをおすすめします。

image\_path: 画像ファイルパス

画像ファイルパスに指定した画像と合致する箇所を検索します。

※ファイルパスは、絶対パスで指定してください。

retry\_count: 検索回数

該当箇所が見つからなかった場合に再検索を行う回数を指定します。

search\_area: 検索エリア

検索する範囲を指定します。

ImageLibrary.SearchArea.ALL = 0(画面全体)

ImageLibrary.SearchArea.UPPER(画面全体の上半分)

ImageLibrary.SearchArea.LOWER(画面全体の下半分)

ImageLibrary.SearchArea.LEFT(画面全体の左半分)

ImageLibrary.SearchArea.RIGHT(画面全体の右半分)

ImageLibrary.SearchArea.UPPER\_LEFT(画面全体を4分割しての左上)

ImageLibrary.SearchArea.UPPER\_RIGHT(画面全体を4分割しての右上)

ImageLibrary.SearchArea.LOWER\_LEFT(画面全体を4分割しての左下)

ImageLibrary.SearchArea.LOWER\_RIGHT(画面全体を4分割しての右下)

match\_rate\_threshold: 一致率

検索時の一致率を指定します。

デフォルトは、0.8です。1を指定すると完全一致になり、値を小さくするにつれ曖昧な検索になります。

## 付録4 画像比較用ライブラリのメソッド一覧

### ②Click

指定した画像と合致する箇所を検索し、見つかった場合にはその座標に対してマウスの左クリックを行います。

- public static bool Click(string image\_path)
- public static bool Click(string image\_path, int offset\_x, int offset\_y)
- public static bool Click(string window\_name, string image\_path)
- public static bool Click(string window\_name, string image\_path, int offset\_x, int offset\_y)
- public static bool Click(string window\_name, string image\_path, int retry\_count)
- public static bool Click(string window\_name, string image\_path, int offset\_x, int offset\_y, int retry\_count)
- public static bool Click(string window\_name, string image\_path, int retry\_count, ImageLibrary.SearchArea search\_area, double match\_rate\_threshold)
- public static bool Click(string window\_name, string image\_path, int offset\_x, int offset\_y, int retry\_count, ImageLibrary.SearchArea search\_area, double match\_rate\_threshold)

#### ○各パラメータの説明

※Searchと同様のパラメータに加え、以下のパラメータを利用する

offset\_x: x座標方向へのオフセット

画像の左上の座標からのx座標方向(右方向)へのオフセットを指定します。  
指定したオフセットでクリックを行います。

offset\_y: y座標方向へのオフセット

画像の左上の座標からのy座標方向(下方向)へのオフセットを指定します。  
指定したオフセットでクリックを行います。

### ③DoubleClick

指定した画像と合致する箇所を検索し、見つかった場合にはその座標に対してマウスの左ダブルクリックを行います。

- public static bool DoubleClick(string image\_path)
- public static bool DoubleClick(string image\_path, int offset\_x, int offset\_y)
- public static bool DoubleClick(string window\_name, string image\_path)
- public static bool DoubleClick(string window\_name, string image\_path, int offset\_x, int offset\_y)
- public static bool DoubleClick(string window\_name, string image\_path, int retry\_count)
- public static bool DoubleClick(string window\_name, string image\_path, int offset\_x, int offset\_y, int retry\_count)
- public static bool DoubleClick(string window\_name, string image\_path, int retry\_count, ImageLibrary.SearchArea search\_area, double match\_rate\_threshold)
- public static bool DoubleClick(string window\_name, string image\_path, int offset\_x, int offset\_y, int retry\_count, ImageLibrary.SearchArea search\_area, double match\_rate\_threshold)

#### ○各パラメータの説明

※Clickと同様

## 付録4 画像比較用ライブラリのメソッド一覧

### ④RightClick

指定した画像と合致する箇所を検索し、見つかった場合にはその座標に対してマウスの右クリックを行います。

- public static bool RightClick(string image\_path)
- public static bool RightClick(string image\_path, int offset\_x, int offset\_y)
- public static bool RightClick(string window\_name, string image\_path)
- public static bool RightClick(string window\_name, string image\_path, int offset\_x, int offset\_y)
- public static bool RightClick(string window\_name, string image\_path, int retry\_count)
- public static bool RightClick(string window\_name, string image\_path, int offset\_x, int offset\_y, int retry\_count)
- public static bool RightClick(string window\_name, string image\_path, int retry\_count, ImageLibrary.SearchArea search\_area, double match\_rate\_threshold)
- public static bool RightClick(string window\_name, string image\_path, int offset\_x, int offset\_y, int retry\_count, ImageLibrary.SearchArea search\_area, double match\_rate\_threshold)

○各パラメータの説明

※Clickと同様

### ⑤GetPoint

指定した画像と合致する箇所を検索し、見つかった場合にはその座標を引数のPoint型の参照で返します。(ウィンドウを指定した場合でもウィンドウ内での座標ではなく、ディスプレイ全体を見たときの座標を取得します。)

※Point型を使用する際は、名前空間(コード上部)に「System.Drawing」を指定してください。

指定方法: using System.Drawing;

また、座標を返すために下記コードを記述する前に「Point pt」を指定してください。

指定方法: Point pt;

- public static bool GetPoint(string image\_path, out Point pt)
- public static bool GetPoint(string window\_name, string image\_path, out Point pt)
- public static bool GetPoint(string window\_name, string image\_path, int retry\_count, out Point pt)
- public static bool GetPoint(string window\_name, string image\_path, int retry\_count, ImageLibrary.SearchArea search\_area, double match\_rate\_threshold, out Point pt)

○各パラメータの説明

※Searchと同様

# 付録5 CSVファイル操作ライブラリについて

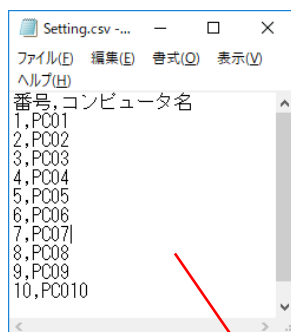
CSVファイル操作ライブラリの名称は、「CsvLib.dll」といいます。

「CsvLib.dll」は、SetROBO内部に組み込まれたライブラリであり、CSVファイルに対してアクション（操作）を行うためのライブラリです。

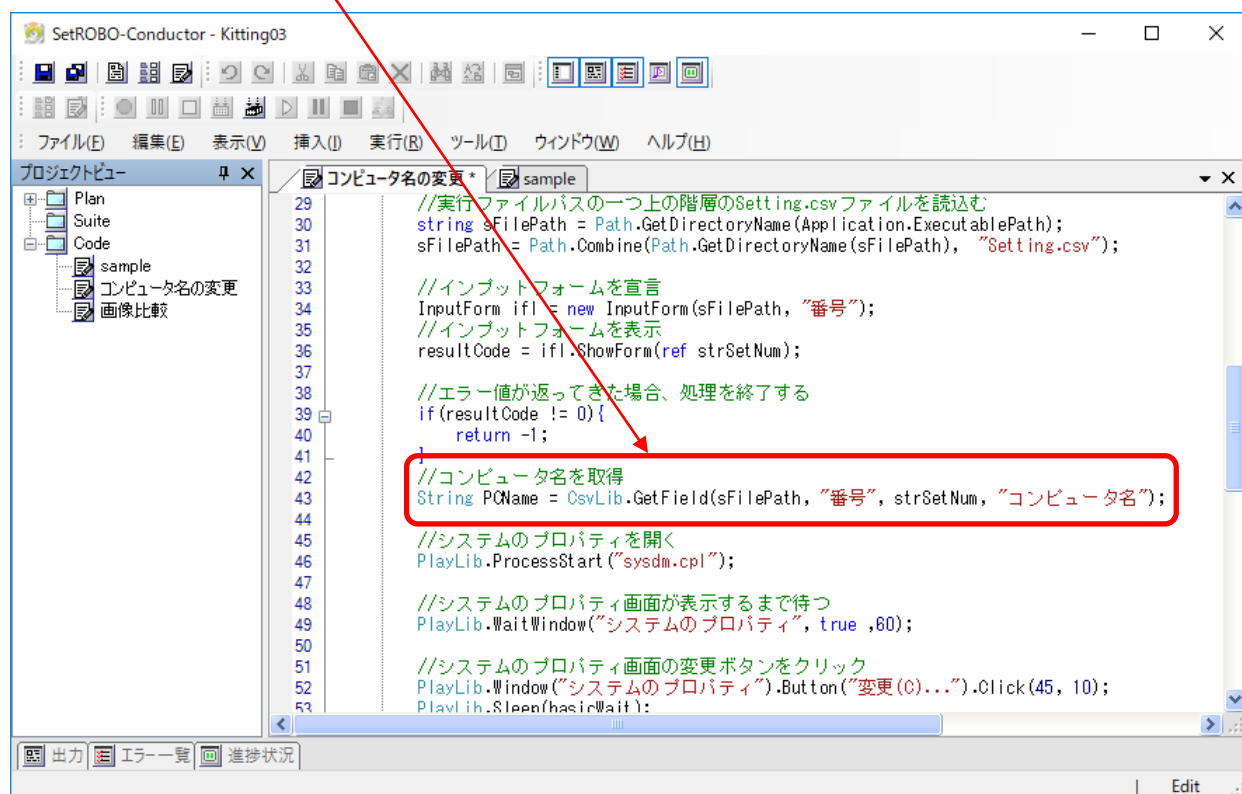
あらかじめ設定値などが書かれたCSVファイルを用意し、そのCSVファイルに対して、設定値の取得や書き込みを行います。

## ●イメージ

### 設定ファイル



### 設定ファイルから値を取得するコードの例 (STEP7参照)



CSVファイル操作ライブラリの詳細な使用方法は、Conductorのヘルプをご覧ください。

## 付録6 共通変数について

共通変数とは、Code間・Plan間をまたいで使用可能な変数のことです。共通変数の値は、Code間・Plan間をまたいだ場合でも保持されます。

共通変数の値の格納と取得を行うためには、Code内で関数を記載していただく必要があります。

①共通変数の値を格納するための関数(共通変数の型によって異なります。)

- PlayLib.GlobalSetInt(string name, int value, out GlobalErrorCode errCode)
- PlayLib.GlobalSetDouble(string name, double value, out GlobalErrorCode errCode)
- PlayLib.GlobalSetString(string name, string value, out GlobalErrorCode errCode)
- PlayLib.GlobalSetBool(string name, bool value, out GlobalErrorCode errCode)

②共通変数の値を取得するための関数(共通変数の型と、共通変数が設定されていない場合にデフォルトの値を設定するかどうかによって異なります。)

- PlayLib.GlobalGetInt(string name, out int value, out GlobalErrorCode errCode)
- PlayLib.GlobalGetInt(string name, int defaultValue, out int value, out GlobalErrorCode errCode)
- PlayLib.GlobalGetDouble(string name, out double value, out GlobalErrorCode errCode)
- PlayLib.GlobalGetDouble(string name, double defaultValue, out double value, out GlobalErrorCode errCode)
- PlayLib.GlobalGetString(string name, out string value, out GlobalErrorCode errCode)
- PlayLib.GlobalGetString(string name, string defaultValue, out string value, out GlobalErrorCode errCode)
- PlayLib.GlobalGetBool(string name, out bool value, out GlobalErrorCode errCode)
- PlayLib.GlobalGetBool(string name, bool defaultValue, out bool value, out GlobalErrorCode errCode)

上記関数①によって格納された共通変数の値を、別のCodeから関数②によって取得することができます。

また、別Planから共通変数を引き継ぐ場合は、Planのオプションにて、引継ぎ元のPlan名を選択した状態でセットアップパッケージを出力、実行することで共通変数の値を引き継ぐことができます。

各関数の使用方法やPlanのオプション設定の詳細については、Conductorのヘルプをご覧ください。

# 付録6 共通変数について

## ●イメージ

### 共通変数の値の格納を行うコードの例

```
Code1 Code2
1 using System;
2 using System.Drawing;
3 using Arp.Prova.ImageLibrary;
4 using Arp.Prova.CsvLib;
5 using Arp.Prova.PlayLib;
6
7
8 // CodeClass
9 public class CodeClass
10 {
11
12     public static int CodeStart ()
13     {
14         // セットアップコードの記述
15         PlayLib.GlobalErrorCode errCode;
16         bool result1 = PlayLib.GlobalSetInt ("a", 100, out errCode);
17         PlayLib.TestLog(errCode);
18
19         return 0;
20     }
21 }
22
```

### 共通変数の値の取得を行うコードの例

```
Cod... Code2
1 using System;
2 using System.Drawing;
3 using Arp.Prova.ImageLibrary;
4 using Arp.Prova.CsvLib;
5 using Arp.Prova.PlayLib;
6
7
8 // CodeClass
9 public class CodeClass
10 {
11
12     public static int CodeStart ()
13     {
14         // セットアップコードの記述
15         int num = 0;
16         PlayLib.GlobalErrorCode errCode;
17         bool result2 = PlayLib.GlobalGetInt ("a", out num, out errCode);
18         PlayLib.TestLog(num); // num = 100
19         PlayLib.TestLog(errCode);
20
21         return 0;
22     }
23 }
24
```

# 付録7 ログの保存先の変更方法について

ConductorからCodeを実行した際やセットアップパッケージを実行した際に出力するログの保存先フォルダを変更することができます。

## ①ConductorからCodeを実行した際のログの出力先の設定

「オプション」>「Conductor設定」タブ内の

- ・ログ保存先(ConductorからCode実行した場合の、Player以外のログ)
- ・ログ保存先(ConductorからCode実行した場合の、Playerのログ)

にて設定可能です。

未設定時(設定項目が空欄のままの時)のパスは、

「[Conductor.exeがあるフォルダ]¥Work¥SystemLog」になります。

## ②セットアップパッケージを実行した際のログの出力先の設定

「オプション」>「SetRobo設定」タブ内の

- ・ログ保存先(セットアップパッケージを実行した場合のログ)

にて設定可能です。

未設定時(設定項目が空欄のままの時)のパスは、

「[セットアップパッケージのフォルダ]¥Work¥SystemLog」になります。

### ①ConductorからCodeを実行した際の設定

オプション

Player通信設定 Conductor設定 環境情報設定 SetRobo設定

記録設定

操作タイミングを記録する

ログ保存先(ConductorからCode実行した場合の、Player以外のログ)

C:\Logs

参照

ログ保存先(ConductorからCode実行した場合の、Playerのログ)

参照

スクリーンキャプチャー出力先(Player実行時)

参照

OK キャンセル

### ②セットアップパッケージを実行した際の設定

オプション

Player通信設定 Conductor設定 環境情報設定 SetRobo設定

出力ファイル設定

出力ファイル名(デフォルトはSetup.exe)

Setup.exe

ログ保存先(セットアップパッケージを実行した場合のログ)

C:\Logs

参照

スクリーンキャプチャー出力先(セットアップパッケージ実行時)

参照

OK キャンセル

# 【ご案内】SetROBO for Kittingサポートページをチェック！！

SetROBO for Kittingサポートページには、いろいろなキittingシナリオのサンプルコードがたくさんあります。

サンプルコードを活用して、簡単にキittingシナリオを作成しましょう。

## SetROBO for Kittingサポートページ:

<http://www.setrobo.jp/support/>



SetROBO for Kitting

SetROBO for Office

導入事例

サポートページ

ご利用環境について

お問い合わせ

## サポートページ

# SetROBO

ホーム ▾ サポートページ

### サンプルコード

SetROBO for Kittingで使用可能なサンプルコード集です。  
サンプルコードはキittingしたい内容ごとになっています。  
対象OSの目的に合うコードをダウンロードし、SetROBOの「Code」に張り付けてご使用ください。

#### サンプルコード一覧

##### ■ Windows10用はこちら

Windows10用のサンプルコード集

##### ■ Windows7用はこちら

Windows7用のサンプルコード集

##### ■ 参考資料

- ・ショートカット一覧集
- ・UIコントローラー一覧集
- ・InspectObjectsインストール手順について
- ・サーバ接続&コピー用&SetROBO実行バッチサンプル
- ・サーバ接続&サーバ上にログ出力&バッチ削除バッチサンプル
- ・実行結果のCSVファイルをマージするバッチサンプル

### Q&A

SetROBOでよくあるお客様からのご質問をまとめました。

※これからも定期的に内容を更新していきます！

下記のリンクを選択すると「Q & A」が表示します。

左側「Q & A」の「+」をクリックするとQ&A一覧が表示します。

##### ■ SetROBO Q&Aはこちら

### お問い合わせはこちら

製品に関するお問い合わせは、下記お問い合わせフォームからご連絡頂けますようお願い致します。



お問い合わせフォームはこちら >>

▶ SetROBO for Kitting

▶ SetROBO for Office

▶ 導入事例

▶ サポートページ

▶ ご利用環境について

お問い合わせ

### 新着情報

2018.9.1

・平成29年度補正 IT導入補助金についてのお知らせ

2018.6.18

・Windows10用サンプルコードページ掲載のお知らせ

2018.6.1

・「SetROBO RPA Solution」リリースのお知らせ

2018.2.15

・SetROBOの新バージョン「1.9.1」リリースのお知らせ

2017.10.6

・「産業交流展2017」アープブースのお知らせ

Software Development  
株式会社 アープ コーポレートサイト

## サンプルコード例：インターネットオプション-詳細設定

```
using System;
using Arp.Prova.PlayLib;

// このサンプルコードは、「インターネットのプロパティ」の「詳細設定」タブ内の設定を行います。
public class CodeClass
{
    public static int CodeStart()
    {
        // 待機時間間隔
        int basicWait = 1000;

        /*****ここから設定の変更を行います*****/
        // 「インターネットのプロパティ」画面を開く
        PlayLib.ProcessStart("inetcp1.cpl");

        // 「インターネットのプロパティ」画面が開くまで待つ(待ち時間80秒)
        if(PlayLib.WaitWindow("インターネットのプロパティ", true, 80) == true)
        {
            // 「詳細設定」タブをクリック
            PlayLib.Window("インターネットのプロパティ").TabControl("TabControl").Click("詳細設定");
            PlayLib.Sleep(basicWait);

            // 「セキュリティ->DOM ストレージを有効にする」項目のチェックがOFF(0)の場合、チェックする
            if((int)PlayLib.Window("インターネットのプロパティ").UITreeView("UITreeView").GetNodeProperty("Checked", "セキュリティ->DOM ストレージを有効にする") == 0)
            {
                // 「セキュリティ->DOM ストレージを有効にする」項目をクリック
                PlayLib.Window("インターネットのプロパティ").TreeView("TreeView").Click("セキュリティ->DOM ストレージを有効にする");
            }

            // 「セキュリティ->DOM ストレージを有効にする」の項目にチェックが入っているかチェック
            if((int)PlayLib.Window("インターネットのプロパティ").UITreeView("UITreeView").CheckNodeProperty("Checked", "セキュリティ->DOM ストレージを有効にする", 1) == false)
            {
                // 「セキュリティ->DOM ストレージを有効にする」の項目にチェックが入っていなかった場合、ログを出力し処理を終了する
                PlayLib.TestLogError("「セキュリティ->DOM ストレージを有効にする」項目にチェックが入っていませんでした。");
                return -1;
            }

            // 「インターネットのプロパティ」画面を閉じる
            PlayLib.Window("インターネットのプロパティ").Button("OK").Click();
        }
        else
        {
            // 「インターネットのプロパティ」画面が開かなかった場合、ログを出力し処理を終了する
            PlayLib.TestLogError("「インターネットのプロパティ」画面が開きませんでした。");
            return -1;
        }

        return 0;
    }
}
```

## サンプルコード例：Adobe Acrobat Readerのアンインストール

```
using System;
using System.Drawing;
using Arp.Prova.ImageLibrary;
using Arp.Prova.DevLib;
using Arp.Prova.PlayLib;

// CodeClass
public class CodeClass
{
    public static int CodeStart()
    {
        // Adobe Acrobat Reader DC - Japanese のアンインストール
        // 【注意】 アンインストール手順は、アプリケーションやバージョンによって異なります。
        // それぞれのアプリケーション用のアンインストール用コードを作成してください。

        // 待機時間間隔
        int basicWait = 1000;

        /*****ここから設定の変更を行います*****/
        // 「プログラムと機能」画面を開く
        PlayLib.ProcessStart("appwiz.cpl");

        // 「プログラムと機能」画面が開くまで待つ(待ち時間80秒)
        if(PlayLib.WaitWindow("プログラムと機能", true, 80) == true)
        {
            PlayLib.Sleep(basicWait);

            // 「Adobe Acrobat Reader DC - Japanese」をダブルクリック
            PlayLib.Window("プログラムと機能").ListView("ListView").DbClick("Adobe Acrobat Reader DC - Japanese");
            PlayLib.Sleep(basicWait);

            // アンインストールメッセージ表示後の「はい(Y)」ボタンが表示されるまで待つ(待ち時間80秒)
            if(PlayLib.Window("プログラムと機能").WaitControl("Button", "はい(Y)", true, 80) == true)
            {
                PlayLib.Sleep(basicWait);

                // 「はい(Y)」ボタンをクリック
                PlayLib.Window("プログラムと機能").Button("はい(Y)").Click();
                PlayLib.Sleep(basicWait);
            }

            // 「Windowsインストーラ」画面が消えるまで待つ(待ち時間180秒)
            PlayLib.WaitWindow("Windows インストーラ", false, 180);
            PlayLib.Sleep(basicWait);

            // アンインストールを実施し、「Adobe Acrobat Reader DC - Japanese」画面が消え終わったら次の動作をさせる
            if(PlayLib.WaitWindow("Adobe Acrobat Reader DC - Japanese", true, 10) == true)
            {
                PlayLib.Sleep(basicWait);

                PlayLib.WaitWindow("Adobe Acrobat Reader DC - Japanese", false, 80);
                PlayLib.Sleep(basicWait);
            }

            // アンインストール後、「Adobe Acrobat Reader DC - Japanese」が存在していない事をチェック
            PlayLib.Window("プログラムと機能").ListView("ListView").CheckCellProperty("Exists", "Adobe Acrobat Reader DC - Japanese", 0, false, 10);

            // 「プログラムと機能」画面を閉じる
            PlayLib.Window("プログラムと機能").Close();
        }
        else
        {
            // 「プログラムと機能」画面が開かなかった場合
            // ログを出力し処理を終了する
            PlayLib.TestLogError("「プログラムと機能」画面が開きませんでした。");
            return -1;
        }

        return 0;
    }
}
```

## - 変更点 -

### 【Ver.1.0からの変更点】

#### 1) 誤記修正

P18 「CheckProperty関数について」にて、期待値の値に誤りがありましたので修正いたしました。  
「Checked」の設定値 (誤)true または false ⇒ (正)0 または 1

### 【Ver.1.1からの変更点】

#### 1) 誤記修正

P38 「付録1 Windows7ショートカットコマンド一覧」にて、実行コードに誤りがありましたので修正いたしました。

スクリーンセーバーの設定

(誤) PlayLib.ProcessStart("rundll32.exe" + "shell32.dll, Control\_RunDLL Desk.cpl,,1");  
⇒ (正) PlayLib.ProcessStart("rundll32.exe", "shell32.dll, Control\_RunDLL Desk.cpl,,1");

### 【Ver.1.2からの変更点】

- 1) 全体的な文章の見直し
- 2) STEP08の追加

### 【Ver.1.3からの変更点】

- 1) STEP8 画像を比較して操作するを追記
- 2) 付録2 画像比較ライブラリについてを追記
- 3) 付録3 画像比較ライブラリのメソッド一覧を追記

### 【Ver.1.4からの変更点】

- 1) 全体的に、画像をWindows10の画面に変更
- 2) 付録1 Windows10ショートカット一覧を追記
- 3) 付録4 CSVファイル操作ライブラリについてを追記

### 【Ver.1.5からの変更点】

- 1) IV 事前準備 拡大率についての項目を追記
- 2) V 注意 Aeroについての項目を追記

### 【Ver.1.6からの変更点】

- 1) SetROBOの名称をSetROBO for Kittingに変更

### 【Ver.1.7からの変更点】

- 1) 付録3 画像比較用ライブラリのメソッド一覧を更新

### 【Ver.1.8からの変更点】

- 1) STEP9 記録が取れないコントロールの記述方法についてを追記
- 2) 付録1 Windows7ショートカットコマンド一覧を削除し、Windows10ショートカットコマンド一覧の内容を更新

## - 変更点 -

### 【Ver.1.9からの変更点】

1)STEP2 P13 ④注意書き追加、及びCodeの内容を変更

(旧)PlayLib.Window("ごみ箱").Unknown("DirectUIHWND", "DirectUIHWND&3").RightClick(1,50);  
⇒(新)PlayLib.Window("ごみ箱").Unknown("DirectUIHWND", "DirectUIHWND&2").RightClick(1,50);

2)STEP3 P21 「SetROBO-LogViewerについて」の内容を一部変更

3)STEP7 P37 注意書き及び説明文の内容を一部変更

### 【Ver.2.0からの変更点】

1)STEP4 音量ミキサーのボタン名が環境により異なる場合ある為一部内容修正

### 【Ver.2.1からの変更点】

1)STEP4 文章の見直し修正

2)STEP8 P39 注意書きを追加

### 【Ver.2.2からの変更点】

1)付録5 「共通変数について」を追加

2)付録6 「ログの保存先の変更方法について」を追加

### 【Ver.2.3からの変更点】

1)Windows 11の対応のため説明・シナリオを一部修正

2)STEP2 「ゴミ箱を空にする」コードから「指定のフォルダを開く」コードに変更

3)付録1 「Windows11ショートカットコマンド一覧」を追加

4)上記に伴い、付録の採番を修正

# SetROBO for Kitting 入門ガイド



## SetROBO for Kitting入門ガイド Ver.2.4

【本書に対するお問合せ】

株式会社アープ

Mail : [sales-ml@arp-corp.co.jp](mailto:sales-ml@arp-corp.co.jp)

【発行】

株式会社アープ

東京都千代田区神田小川町3-8 神田駿河台ビル6F