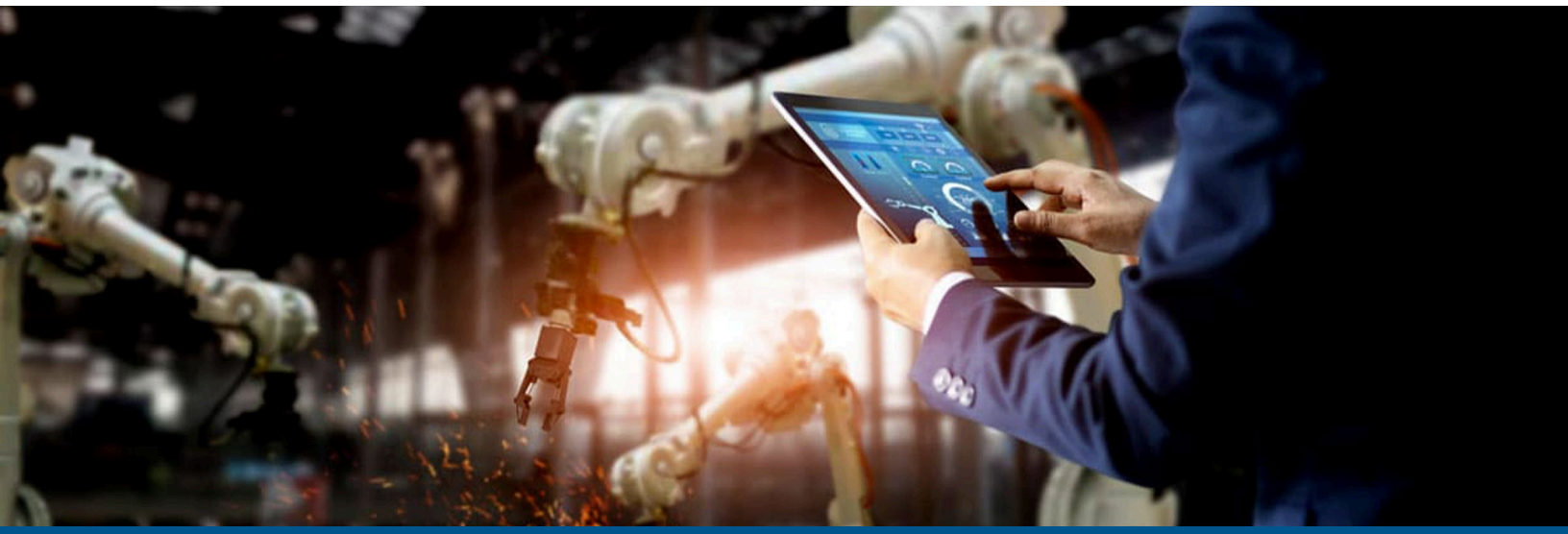


Kofax RPA
ユーザー ガイド

バージョン: 11.5.0

日付: 2023-10-02

KOFAX

目次

はじめに

このガイドは、『Kofax RPA のヘルプ』の PDF バージョンです。

関連ドキュメント

Kofax RPA のドキュメント セットには次の場所からアクセスできます。[1]

https://docshield.kofax.com/Portal/Products/RPA/11.5.0-nlfihq5gwr/RPA.htm

ドキュメント セットには、次のようなリソースがアルファベット順で含まれています。

Kofax RPA 管理者ガイド
Kofax RPA での管理タスクについて説明します。

Kofax RPA のベストプラクティス ガイド
Kofax RPA 環境で口ボット ライフサイクル マネジメントを使用しながらパフォーマンスを最適化し、成功を確実にするために推奨される方法とテクニックを提供します。

Kofax RPA Desktop Automation サービス ガイド
リモート コンピューターで Desktop Automation を使用するために必要な Desktop Automation サービスを設定および管理する方法について説明します。

Kofax RPA 開発者ガイド
RoboServer で口ボットを実行するために使用される Java および .NET API のプログラマー ユーザー ガイドが含まれています。また、製品で提供される Management Console REST サービスに関する情報が含まれています。

Kofax RPA 口ボット構築の開始ガイド
Kofax RPA を使用して口ボットを構築するプロセスを実行するためのチュートリアルを提供します。

[1] オンラインのドキュメント セットにアクセスするにはインターネットに接続する必要があります。インターネットに接続せずにアクセスする方法については、『インストール ガイド』を参照してください。

Kofax RPA Document Transformation スタート ガイド
OCR、抽出、フィールドの書式設定、検証などを含む Kofax RPA 環境の Document Transformation 機能を使用する方法について説明します。

Kofax RPA のヘルプ
Kofax RPA の使用方法について説明しています。ヘルプは、『Kofax RPA ユーザー ガイド』という PDF 形式のドキュメントとしても提供されています。

Kofax RPA インストール ガイド
Kofax RPA およびそのコンポーネントを開発環境にインストールする方法について説明します。

Kofax RPA Java API documentation (Kofax RPA Java API ドキュメント***)***
開発者が Kofax RPA で使用できる Kofax RPA Java API パッケージおよびクラスへのアクセスを提供します。

Kofax RPA API は、元の製品名である「RoboSuite」に対する詳細な参照を含んでいます。RoboSuite の名前は下位互換性を確保するために残されています。API ドキュメントの中では、RoboSuite という用語は Kofax RPA と同じ意味で使われています。

Kofax RPA リリース ノート
その他の Kofax RPA ドキュメントからは入手できない最新の詳細やその他の情報が含まれています。

Kofax RPA 技術仕様
サポートされるオペレーティング システムおよびその他のシステム要件に関する情報が含まれています。

Kofax RPA アップグレード ガイド
Kofax RPA やそのコンポーネントを新しいバージョンにアップグレードする手順が含まれています。

Kofax RPA ユーザー ガイド
Kofax RPA とそのコンポーネントの使用手順が記載されています。*Kofax RPA* のヘルプ トピックに加えて、ヘルプに記載されていない詳細な内容が含まれています。

トレーニング

Kofax は、Kofax RPA ソリューションを最大限に活用するために、教室でのトレーニングとコンピュータでのトレーニングを提供しています。利用可能なトレーニング オプションとスケジュールの詳細については、https://learn.kofax.com/ の Kofax 教育ポータルを参照してください。

また、https://smarthub.kofax.com/ の Kofax Intelligent Automation SmartHub にアクセスして、追加のソリューション、ロボット、コネクタなどを見つけることもできます。

Kofax 製品のヘルプの入手

[Kofax Knowledge Portal (Kofax ナレッジ ポータル)] リポジトリにある記事の内容は定期的に更新され、Kofax 製品の最新情報について参照できます。製品に関してご不明の点がある場合は、Knowledge Portal (ナレッジ ポータル) で情報を検索することをお勧めします。

[Kofax Knowledge Portal] にアクセスするには、https://knowledge.kofax.com にアクセスしてください。

[Kofax Knowledge Portal] は Google Chrome、Mozilla Firefox、または Microsoft Edge 向けに最適化されています。

[Kofax Knowledge Portal] は以下の内容を提供します。

- 強力な検索機能で必要な情報をすぐに見つけることができます。
  [Search (検索)] ボックスに目的の語句を入力し、検索アイコンをクリックしてください。
- 製品情報、設定の詳細、リリース情報などのドキュメント。
  記事を見つけるには、Knowledge Portal のホームページにアクセスし、製品に該当するソリューション ファミリを選択するか、[View All Products (すべての製品を表示)] ボタンをクリックします。

Knowledge Portal のホームページからは、次の操作を実行できます。

- Kofax Community (Kofax コミュニティ) へのアクセス (全カスタマー)。
  [Resources (リソース)] メニューで、**[Community (コミュニティ)]** リンクをクリックします。
- Kofax Customer Portal (Kofax カスタマー ポータル) へのアクセス (一部のカスタマーのみ)。
  [Support Portal Information (サポート ポータルの情報)] ページに移動し、**[Log in to the Customer Portal (カスタマー ポータルにログイン)]** をクリックします。
- Kofax Partner Portal (Kofax パートナー ポータル) へのアクセス (一部のパートナーのみ)。
  [Support Portal Information] ページに移動し、**[Log in to the Partner Portal (パートナー ポータルにログイン)]** をクリックします。
- Kofax サポート コミットメント、ライフサイクル ポリシー、電子フルフィルメントの詳細、セルフサービス ツールへのアクセス。
  [Support Details (サポートの詳細)] ページに移動し、適切な記事を選択します。

第 1 章

概要

Kofax RPA は、アプリケーション統合および Robotic Process Automation (RPA) 向けのプラットフォームです。接続用に構築されていないアプリケーションを統合し、クラウド/SaaS アプリケーションとオンプレミス システム、レガシー システムと最新の Web アプリケーション、バック オフィス システムとパートナー Web サイトなど、異種のシステム間のプロセスを自動化できます。

Design Studio ビジュアル エディターを使用することで、統合を行うアプリケーションやデータ ソースをクリック スルーして、自動化されたワークフローを標準的な手順で作成することができます。

Kofax RPA では、これらのワークフローをロボットと呼びます。ロボットを構築すると、アプリケーションを統合することによって、アプリケーション間を自由に移動できます。アプリケーションへのログイン、ページのデータ部分の抽出、データのフォームまたは検索ボックスへの入力、メニュー選択、複数のページのスクロールが可能です。構築したロボットは、データベース、ファイル、API、Web サービス、およびその他のロボットにアクセスして、特定のアプリケーションからデータをエクスポートし、別のアプリケーションにロードすることもできます。また、Document Transformation Service の OCR エンジンを使用してデータを認識し、必要に応じてデータを変換することができます。

Kofax RPA 環境の Document Transformation 機能は、OCR、抽出、フィールドの書式設定、検証などの機能を備えています。

Kofax RPA の Desktop Automation を使用して、ネットワーク コンピュータ上の Windows および Java アプリケーションを自動化することができます。Desktop Automation は、デスクトップまたはターミナル上のアプリケーションを制御することにより、手動プロセスを置き換えます。詳細については、ロボットの構築 を参照してください。

構築されたロボットは、Management Console のリポジトリにアップロードされます。ここから、ロボットを RoboServer で一括実行するのようにスケジュールすることができます。また、Java および C# API 経由のオンデマンドで実行したり、調整可能な REST サービスで実行したりすることもできます。この REST サービスは、ロボットがリポジトリに追加され、Kofax RPA Kapplet という特殊用途のエンドユーザー Web アプリケーションとして公開されると利用可能になります。

また、Management Console を使用することで、負荷分散、フェールオーバー、RoboServer の健全性の監視、ユーザー ロールおよび権限の管理を行うこともできます。

CEF への移行とロボット名の変更について

Kofax では、最新の Web サイトおよび Web アプリケーションには Chromium ブラウザ (Desktop Automation) を使用し、静的な Web サイトには WebKit ベースのブラウザ (Web Automation) を使用することをお勧めします。

WebKit から **Chromium** に移行する理由

Kofax RPA バージョン 10.3.0 以降では、Web サイトおよび Web アプリケーションを操作する 2 つの方法として、WebKit ベースのブラウザを使用する方法 (Web オートメーション ロボットを使用)、および Chromium ブラウザを使用する方法 (Desktop Automation ロボットを使用) が用意されています。リアルタイム データを処理するように Web サイト テクノロジが進化するにつれて、最新の動的 Web サイトおよび Web アプリケーションを処理するための機能を求める需要が高まっています。対照的に、WebKit ベースのブラウザは、当初、Web サイト全体の状態をロボットに保持し、ロボットが時間を遡って、過去に状態が保存された任意の時点から実行できるように設計されています。(つまり WebKit ブラウザは Web ページのリアルタイムデータを処理する仕様となっていません)

「ブラウザのタイプ」を参照してください。

既存のロボットを **Chromium** に移行するタイミング

新しいロボットは Chromium ブラウザを使用して構築することをお勧めしますが、Kofax は、静的な Web サイトを操作するために WebKit ベースのブラウザを引き続きサポートしています。ターゲットとなる Web サイトが大幅に変更されない限り、既存の WebKit で構築されたロボットを引き続き使用することができます。時間とともに、ターゲットとなる Web サイトの変更に伴って WebKit で構築された既存のロボットでパフォーマンスの問題や実行の失敗が発生した場合は、それらの Web サイトを Chromium ブラウザに移行することを検討してください。必要な機能が Chromium ブラウザで構築されたロボットで利用できない場合は、Kofax テクニカル サポートに解決策をお問い合わせください。不足している機能は、製品の今後のリリースで追加される予定です。

Chromium ブラウザでトレーニングを受けるには、Kofax Education に連絡するか、こちらでトレーニング コースを確認してください。

RPA ロボット名を変更する理由

WebKit から Chromium への移行により、Kofax RPA ロボット名に次の 2 つの変更が加えられました。

新しいロボット名	元のロボット名
ベーシック エンジン ロボット	Web オートメーション ロボット
ロボット	Desktop Automation ロボット

この名前変更により、Kofax RPA 内で開いている Web サイトのリアルタイム変更対応が促進されます。より多くの相互作用が一連の動的データに依存するようになったため、ロボット (これまでは Desktop Automation ロボットと呼ばれていました) を使用することにより、製品の将来のリリースにおいて Kofax RPA が果たす主導的な役割は強化されます。

Kofax RPA ドキュメントの変更点

名前の変更と Chromium ブラウザへの継続的な移行が反映されるように、このヘルプ システムと Kofax RPA ドキュメント セット全体が更新されました。ヘルプ システムでは、Desktop Automation セクションの名前が「ロボットの構築」に変更されました。Web オートメーション セクションの名前が「ベーシック エンジン ロボット」に変更されて、「リファレンス」セクションに配置されました。

ブラウザのタイプ

Kofax RPA はロボットを使用して Web サイトを操作し、そこからデータを抽出します。この目的のために、ロボットはChromium および WebKit という特定のテクノロジを基盤にした組み込みブラウザを使用します。バージョン 11.2.0 以降の Kofax RPA は、WebKit ベースのブラウザから Chromium ブラ

ウザに移行しているため、新しいロボットの作成には Chromium ブラウザを使用することをお勧めします。CEF への移行とロボット名の変更についてを参照してください。

Chromium ブラウザ

このセクションでは、ロボット (旧称、Desktop Automation ロボット) で使用される推奨ブラウザについての情報を示します。

Chromium ブラウザは、Kofax RPA に実装されている最新のブラウザ エンジンです。このブラウザは、状態がサーバー側に存在する Web サイト、つまり状態がロボットの外部にある最新の Web サイトで使用することが推奨されます。このブラウザでのステップの実行は前方への移動のみに限定され、Web ページを移動するには、ブラウザのツール (戻るボタンや進むボタンなど) を使用する必要があります。このブラウザは定期的に更新され、JavaScript などの最新の Web テクノロジーで構成されます。このブラウザの使用方法の詳細については、Web サイトのアクセスを参照してください。

Chromium ブラウザでトレーニングを受けるには、Kofax Education に連絡するか、こちらでトレーニング コースを確認してください。

WebKit ベースのブラウザ

このセクションでは、ベーシック エンジン ロボット (旧称は Web オートメーション ロボット) で使用されるブラウザについての情報を示します。

WebKit ベースのブラウザ
このブラウザでは、Web ページの状態がロボットの内部に保存されます。これは、Web コンテンツ (クライアント側の状態) をクライアント側で制御できる、リソースへの依存度が低いレガシー Web サイト向けのブラウザです。WebKit ベースのブラウザでは、async 関数、await 演算子などの JavaScript 機能を含む、最新の Web 標準およびテクノロジがサポートされていません。

命名規則

Kofax RPA では、Management Console および Design Studio に対して、次の命名規則が定められています。

共通ルール

- ファイル拡張子を含めて 243 文字を超える名前は使用できません。
- 次の句読点は使用しないでください: /、\、<、>、 : 、"、|、?、または *。
- 空の名前は使用できません。
- フォルダ名は、親フォルダ内で一意である必要があります。
- ファイルの絶対パスには 255 文字の制限があります。
- 11.2.0 より古いバージョンで作成されたバックアップ/プロジェクトに含まれるアイテムの名前が命名規則に準拠していない場合、プロジェクトのバックアップ/インポートを復元すると、これらのアイテムの名前が自動的に変更されます。
- ファイル名に、次に示すようなシステムで予約された名前を使用することはできません:
  CON、PRN、AUX、NUL、COM1、COM2、
  COM3、COM4、COM5、COM6、COM7、COM8、COM9、LPT1、LPT2、LPT3、LPT4、LPT5、LPT6、LPT7、

LPT8、LPT9。また、これらの名前の直後に拡張子を付けて使用しないでください。
- ファイル名またはフォルダ名に、ASCII 制御文字 0 ~ 31 を使用することはできません。

タイプとデバイス マッピング名に関する特定のルール
- タイプ名には、文字、数字、アンダースコア以外の文字は使用できません。
  また、Management Console のデバイス マッピング名にもこのような文字を使用しないでください。
- 名前の先頭に数字を使用しないでください。
- 名前 `local` は予約された名前であり、デバイス マッピング名には使用できません。
- デバイス マッピング名で `true` または `false` を使用しないでください。

重複した名前
- プロジェクト内で重複した名前を使用できるのは、ベーシック エンジン ロボット (旧称、Web Automation ロボット)、テキスト/リソース、およびフォルダのみです。
- ロボット、スニペット、タイプ、リソース、データベース マッピング名など、親フォルダ内の名前の重複は許可されていません。

ファイル名でのピリオド「.」の使用
- ピリオド「.」で開始または終了するファイルやフォルダを使用または作成することはできません。
- ファイル名をピリオドのみで構成することはできません。

ファイル名でのスペースの使用
- ファイル名やフォルダ名をスペースで開始または終了することはできません。
- タイプ名およびデバイス マッピング名にスペースを使用することはできません。
- リソース ファイル名とフォルダ名には、1 行に複数のスペースを含めることができます。

第 2 章

チュートリアル

このセクションには、Kofax RPA でさまざまなタスクを実行するために役立つビデオ チュートリアルへのリンクが含まれています。これらのチュートリアルは、製品の概要を提供するだけでなく、最初のプロジェクトを作成する場合のガイドとしても機能します。これらのチュートリアルを続行する前に、Kofax RPA を適切にインストールしてセットアップしてください。

ビデオ チュートリアルを表示するには、インターネット接続が必要です。

- ビギナー チュートリアル - RPA の概要
- ビギナー チュートリアル - ベーシック エンジン ロボット
- ビギナー チュートリアル - Kapplet
- ビギナー チュートリアル - タイプ

第 3 章

Design Studio

Design Studio は、ロボットとタイプを作成するためのアプリケーションです。Design Studio で、ロボットをデバッグしたり、データベース データ登録する必要があるタイプのデータベース テーブルを作成することもできます。

ロボット開発のための統合開発環境 (IDE) である Design Studio は、ロボットとタイプの設計に必要なものがすべて揃っています。

ロボットは、独自の構文 (構造) とセマンティクス (意味) を持つ理解しやすいビジュアル プログラミング言語でプログラミングされます。ロボットの構築をサポートするために、Design Studio では対話型のビジュアル プログラミング、完全デバッグ機能、プログラム状態の概要、コンテキストからのアクセスが容易なオンライン ヘルプなどの強力なプログラミング機能が提供されています。

Design Studio では、ロボット変数によってデータ抽出および入力に使用されるタイプを作成することもできます。Design Studio タイプ エディターを使用して、実際のデータをモデル化したタイプを設計できます。最も一般的なケースで、タイプはロボットがデータ ソースから抽出するデータを保持するように設計されています。

Design Studio について

Design Studio は、ロボットを作成し、タイプをデザインするためのプログラミング環境です。ロボットの作成には、独自の構文とセマンティクスを持つ特殊用途のプログラミング言語を使用します。他のプログラミング環境と同様に、Design Studio で使用される概念は、ロボットのデザイナーとして Design Studio の動作を完全に把握するために理解する必要があります。このセクションでは最も重要な概念を定義しているため、必要に応じてこのセクションに戻って参照することをお勧めします。Design Studio に触れ、ロボットの作成を開始すると、Design Studio の概念がより明確に理解できます。

ロボット

Design Studio の最も重要な概念はロボットです。ロボットとは、データ ソース (通常は Web サイトやデスクトップ アプリケーション) に関連するタスクの一部を実行するように設計されたプログラムのことを指しますが、Excel や PDF ドキュメントであることや、Kofax TotalAgility や Kofax SignDoc などの別の Kofax 製品であることもあります。

Kofax RPA では、ベーシック エンジン ロボットと、ロボットという 2 つのタイプのロボットを作成できます。主な機能は、ロボットが Web サイトを操作するために使用するブラウザ テクノロジーです。

ロボット
これらのロボットは、Web サイトを操作し、サーバー/アプリケーション側に配置された状態になっているリモート アプリケーションを自動化するように設計します。この場合、状態はロボットの外部になり

ます。主要なロボット タイプはロボットであり、自動化する必要のあるすべてのプロセスにロボットを使用することを強くお勧めします。

Kofax RPA では、ネットワーク接続しているコンピュータ上の Windows および Java アプリケーションを対象とする作業プロセスを自動化できるロボットを作成できます。ロボットのワークフローは、順々に実行される一連のステップです。これらのステップは、自動化された Web サイトやアプリケーションをユーザーが操作する方法をモデル化します。詳細については、ロボットの構築 および ロボット構築の概要 を参照してください。

ロボットの Document Transformation 機能を使用すると、画像およびテキスト ドキュメントから情報を抽出して使用することができます。Document Transformation アクションでは、選択したプロジェクトを使用してグラフィカル ドキュメントまたは PDF ドキュメントを処理します。プロジェクトは、OCR やその他の指定された操作を実行してドキュメントを処理および変換するモジュールです。詳細については、Document Transformation を参照してください。

ベーシック エンジン ロボット
これらのロボットの当初の目的は、ロボットの内部に状態が配置されているステートレス アプリケーションを自動化することでした。ベーシック エンジン ロボットには制限があるため、自動化する必要のあるすべてのプロセスにはロボット を使用することをお勧めします。ただし、ターゲットとなるアプリケーションや Web サイトが大幅に変更されない限り、既存のベーシック エンジン ロボットを引き続き使用することができます。このロボット タイプの詳細については、ベーシック エンジン ロボットを参照してください。

追加のソリューション、ロボット、コネクタなどを見つけるには、https://smarthub.kofax.com/ の Kofax Intelligent Automation SmartHub にアクセスしてください。

ロボット実行モード

Kofax RPA Design Studio は、デザイン時に以下の 2 つのロボット実行モードをサポートします：最小実行 (ダイレクト) およびスマート再実行 (フル) です。このトピックでは、この 2 つのモードについて詳しく説明します。

新しいベーシック エンジン ロボット の作成時に、新規ロボット ウィザードで実行モードを選択できます。実行モードを表示または変更するには、ロボットの設定のデザイン モード タブを使用します。選択されたロボットの実行モードは、デザイン モードでの実行にのみ影響し、デバッグ モードや RoboServer の実行には影響しません。

スマート再実行モードは、ロボット でワークフローをサポートする唯一のモードです。ワークフローを最初に実行すると、戻された変数の状態がキャッシュされます。ワークフローが更新された場合にも、キャッシュされた変数は更新されません。ワークフローを変更しても、キャッシュされた変数の状態は更新されません。ワークフローから戻された変数値を更新するには、ロボット全体を再実行する必要があります。

スマート再実行 (フル)

スマート再実行モードにおいて、デザイン モードでのロボットの実行方法は、実行時またはデバッグ モードで実行される方法と同様です。これは、ベーシック エンジン ロボット のデフォルトのモードです。

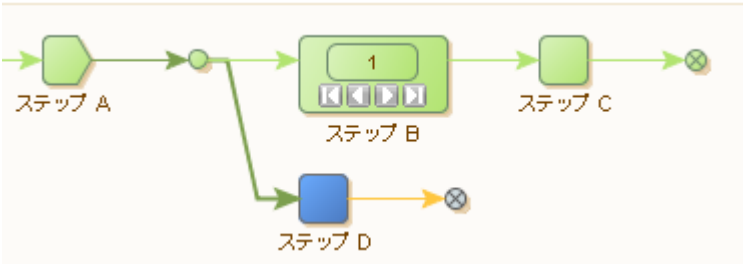
一例として、以下のロボットでステップ C をクリックすると、値判定ステップでテストが失敗した場合に、トライ構造の一番下の分岐を通ってロボットが自動的に実行されます。

[値判定] ステップの青色の感嘆符アイコンは、[次のトライステップへ移動] へのエラー処理がトリガーされたことを示します。

ループ内のステップをクリックすると、ループ ステップによって、選択したイテレーションまでを含むすべてのステップが実行されます。

以下の例では、ステップ C をクリックするとループのイテレーションが 3 回実行されます。

さらに、ループの下の分岐のステップをクリックすると、ループのすべてのイテレーションが実行されます。例として、以下の口ボットをご覧ください。



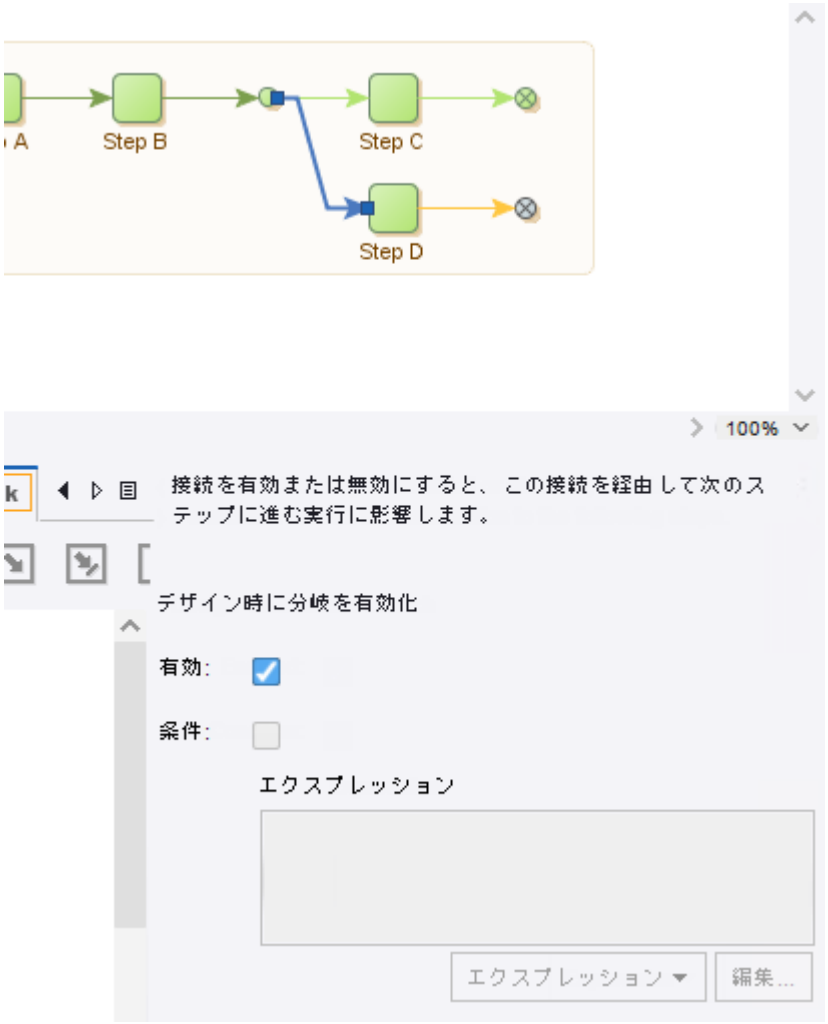
ステップ D をクリックすると、ステップ A の実行と ステップ B および C の繰り返し (ループ B の反復回数と同じ回数) が実行され、最終的にステップ D で停止します。

スマート再実行モードが特に便利なのは、グローバル変数で作業する場合や、正確な変数に応じたステップが口ボット内で後続する場合などです。このモードは、ウェブサービス (REST または SOAP) 呼び出しのペイロードを構築したり、Excel ドキュメントを作成したりするのに便利です。XML または Excel ドキュメントはグローバル変数に埋め込まれますが、そのコンテンツはループの実行中に追加されます。ドキュメントを生成するループの下の分岐で、口ボットはドキュメント全体を取り出し、それをウェブ サービスまたは同等のサービスにポストします。この場合、ドキュメントが確実に埋め込まれているため、デザイン モードでウェブ サービスの呼び出しをテストする場合に、スマート再実行モードによる口ボットの構築が容易となります。

スマート再実行では、ウェブサイト、データベース、またはウェブ サービスによる外部とのやりとりがキャッシュされます。実行結果を格納するための前提条件 (ロードする URL を決定する変数など) が変更されるまで、キャッシングによってステップの再実行が回避されます。スマート再実行モードには、最小実行モードよりも高いメモリ フットプリントがあります。

外部と重要なやりとりを行う大型口ボットや長時間稼働する口ボットには、スマート再実行モードはお勧めしません。こうした口ボットでは、実行時間が長くなり、メモリの使用量が高くなり過ぎてしまいます。

ロボットのデザイン時の実行時間を短縮するために、Design Studio で分岐を右クリックしてロボットを無効にすることもできます。デバッグ モードでも同様の設定を適用することができます。さらに、選択したイテレーションで指定された条件に基づいて、分岐を無効にすることもできます。



接続設定

また、ロボットの設定の [デザイン モード] タブには、[外部の再実行を回避] オプションがあります。このオプションにチェックを入れると、前の実行でキャッシュされた結果が使用できない場合でも、ステップが再実行されないようにすることができます。この場合、ロボットの編集は可能ですが、作業対象となる現在の入力状態は表示されません。このオプションは、外部環境とのやりとりの再実行を避ける場合にのみ使用します (たとえば、再実行によりパートナーのシステムでデータに不整合が生じる、または重複が発生するといった場合など)。

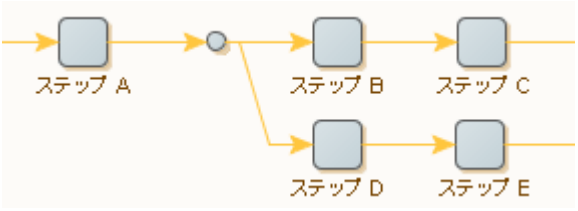
スマート再実行モードでは、一部のステップが利用できません。使用できないステップの一覧については、Kofax RPA の制限トピックの「実行モード」の項を参照してください。

最小実行 (ダイレクト)

最小実行 (ダイレクト) モードは、従来の Design Studio 実行モードです。9.5 以前のバージョンで記述された口ボットはすべてこの実行モードを使用します。

口ボット グラフの最小実行モードでステップをクリックすると、ダイレクト パスにない以前の分岐とイテレーションをスキップし、Design Studio はそのステップへの最短のダイレクト パスを選択します。

次の例を考えてみましょう。



通常、口ボットは実行時にステップ E へ到達する前にステップ A、B、C および D を実行します。ただしデザイン モードでは、ステップ E をクリックするとステップ A および D の実行のみが行われます。

同様に、ステップがループ内にある場合は、選択されたイテレーションのみが実行されます。

イテレーション カウンタが 3 に設定されているため、ステップ C をクリックするとステップ A、B および C のみが一度実行され、ステップ B では第 3 のイテレーションが選択されます。

最小実行モードは、できるだけ少ないステップを実行するように最適化されています。複雑なウェブサイトとやり取りするステップなど、実行にかなりの時間がかかる大きな口ボットやステップがあるとき、このモードが便利です。一般的に、データ収集を行う多くの場合または外部サイトとの重要なやりとりを行う口ボットには、最小実行が推奨されます。

最小実行モードの欠点は、デフォルトのパスを使用してステップに口ボットを直接実行できない場合に、ユーザーが指定されたステップへのパスを選択する必要があることです。たとえば、パスのトライ ステップによって口ボットの一番上の分岐が妨げられることがあります。

次の例をご覧ください。

ステップ C をクリックした場合に、[値判定] ステップでテストが失敗すると、最小実行モードは続行されません。この場合、ステップ C に対する実行パスを指定するには、ユーザーが最初にトライ ステップの一番下の分岐を明示的にクリックする必要があります。

ステップ

ロボットは複数のステップで構成されており、これがロボット プログラム内の構築ブロックとなります。

ステップはロボット状態で動作し、ステップの構成に従って処理を行います。ステップには入力ロボット状態があり、出力ロボット状態を生成します。

ステップには、ステップ名、基盤となるアクション、およびステップ タイプに応じたさまざまなプロパティがあります。ステップ名は、「見出しの抽出 (Extract Headline)」や「検索ページの読み込み (Load Search Page)」など、ステップの内容を表す名前にします。ステップは、ステップが実行するアクションです。たとえば、「ロボットの値を抽出」アクションによって、Web ページから要素の値を抽出して、変数に格納できます。

ファインダーは、ステップが動作するページ内の要素 (HTML タグまたは Excel セル) を検出します。ステップには単一のエレメントを必要とするものがあり、また他のステップ アクションには複数のエレメントの処理を実行できるものもあります。

ステップは実行可能な要素です。実行されたステップはロボット状態を入力として受け入れ、ファインダーとステップを順番に適用し、出力ロボット状態を生成します。出力ロボット状態は次のステップに渡され、その入力ロボット状態になります。ステップの中には "termed loop" アクションというものがあり、こうしたアクションを持つステップを「ループ ステップ」と呼びます。ループ ステップは 0 個以上の出力ロボット状態を生成し、それぞれのロボット状態に対して後続のステップを 1 回実行します。

拡張可能なグループ ステップでステップをグループ化することができます。

概念としての変数

変数は、Design Studio の重要な概念です。

すべての変数はデフォルトの初期値と関連付けることができます。この初期値は、ロボットが明示的に再割り当てしない限り保持されます。再割り当ては、実行中に値が抽出されたり操作されたりするときに多く発生します。ほとんどのロボットは、変数の値を呼び出し元に返して出力するか、データベースに挿入して出力します。また、ロボットは、入力から値を受け取るようにマークされた特定の変数に割り当てられる入力値を取得することができます。これらの変数は、入力変数と呼ばれます。

複雑な変数またはシンプルな変数として各変数を定義できます。これらのタイプは両方とも、通常はロボット内部で一時変数として使用され、ロボットが実行を終了した時点で変数の値は存在しなくなります。また、値を出力しない限り、ロボット間で変数を共有することはできません。

シンプルな変数

シンプルな変数は属性を定義しませんが、単一の値のタイプのみを表します。したがって、簡単なタイプの変数にはテキスト文字列などの単一の値が含まれ、ユーザー名などの変数名のみによって参照されます。簡単なタイプの変数は組み込みであり、編集したり、作成したりできません。

- 一時データを抽出するときや、グローバル カウンターとして便利です。
- 入力変数に対しては、簡単なタイプを使用できません。
- シンプルなタイプの値を出力できません。

複雑な変数

複雑な変数は一連の属性を定義します。複雑な各変数はいくつかの (名前付きの) 値を表します。通常、"title" などの各属性を "Book" などの別個の変数として参照し、Book.title などの完全修飾属性名を使用してその値を表します。必要に応じて、Design Studio 内でコンプレックス タイプを作成できます。

複雑な変数は、さまざまな方法で出力されます。たとえば、Web サイトからニュースを抽出するロボットは、ニュース変数の値を出力する場合があります。それぞれのニュース変数には、見出し、本文、日付、作成者などの属性を備えたコンプレックス タイプが含まれ、出力される各ニュース値は、各名前付き属性に対して一意である可能性のある副値で構成されます。

入力変数が含まれるロボットの場合、入力変数は、入力変数に割り当てられる値を含むロボット入力の一部として指定する必要があります。たとえば、http://amazon.com で本を注文するショッピング ロボットは、ユーザーと本の情報が含まれている入力値に依存する場合があります。これらの入力値は、タイプ "User" および "BookInfo" の "user" および "bookInfo" と呼ばれる、ロボットの 2 つの入力変数に割り当てられる場合があります。次の図は、ロボットが入力値を受け入れて出力値を生成する方法を示しています。

この図は、ロボットの入出力値を示しています。入力値は入力変数に割り当てられ、一部の変数の値が出力値になります。入力から割り当てたり、値を出力したりできるのは、コンプレックス タイプの変数のみです。

変数とタイプを構成する方法については、変数の設定とタイプを設定を参照してください。

ライブラリとロボット プロジェクト

ロボットとタイプは、ライブラリに整理されています。ライブラリは、ロボット定義、タイプ定義、および含まれているロボットを実行するのに必要な他のファイルの集合体です。ライブラリは、ロボットの展開ユニットとして機能します。RoboServer など、実行時環境でロボットを分配および展開する場合は、ライブラリを使ってロボットと必要なファイルをバンドルします。

Design Studio では、必要に応じて、1 つ以上のロボット プロジェクトで作業ができます。ロボット プロジェクトの目的は、ロボット ライブラリを開発することです。ロボット プロジェクトには、ロボット ライブラリでの作業に役立つその他のファイルと同様に、指定したロボット一式を開発するロボット ライブラリを含まれています。ライブラリに配置されたファイルには、特別なライブラリ プロトコルを使ったロボットでアクセスすることもできます。

このように、ロボットを開発する場合にはロボット プロジェクトで作業を行うことになり、またロボット ライブラリがユーザーの作成したロボットを分配または展開する手段になります。詳細については、「プロジェクトとライブラリ」を参照してください。

シェア プロジェクトは Management Console に配備され、ローカル Design Studio コンピュータのプロジェクトに接続されます。Management Console プロジェクトは複数の Design Studio 間で共有できます。マイ プロジェクト ペインの Management Console セクションには、シェア プロジェクト ファイルのステータスの視覚的な表示および説明付きのヒントが表示されます。

Design Studio ユーザー インターフェース

このトピックは、Design Studio ユーザー インターフェイスについて説明し、以下のエレメントなどへのツアーを始めます。

- メニュー バー
- ツールバー
- マイ プロジェクト
- エディター ビュー
- ロボット エディター
- タイプ エディター
- テキスト エディター
- Design Studio のウィンドウ
- ステータス バー

Design Studio メイン ウィンドウを表示するには、アクティブ化された有効なライセンスが必要です。ライセンスについては、『Kofax RPA インストール ガイド』を参照してください。

メニュー バー

メニュー バーは、Design Studio ウィンドウの上部にあります。

使用可能なメニューおよび含まれるアイテムは、エディター ビューで開いたファイルのタイプに基づいて決まります。開いているファイルがない場合でも、以下のメニューは常に利用できます (無効になるアイテムもあります)。

- [ファイル] メニューには、ファイル、ロボット、プロジェクトなどを操作するアイテムが含まれています。
- ツール メニューを使用すると、データベース テーブルの生成 (タイプ用)、または Management Console へのロボットの展開 (ロボット用) など、ファイルの種類に応じたタスクの実行が可能になります。
- [設定] メニューには、デフォルト設定を変更したり、プロキシ サーバーまたはデータベース接続を定義したりするアイテムがあります。
- [ウィンドウ] メニューには、[レイアウト初期化] や [レイアウトを保存] など、ユーザー インターフェイスのレイアウトを変更するアイテムが含まれています。
- [ヘルプ] メニューには、オンライン ヘルプ、ドキュメント、およびテクニカル サポート情報へのリンクが含まれています。

ロボットなどのファイルを開くと、[編集]、[表示]、[デバッグ] メニューが使用できるようになります。

- 編集メニューには、開いているファイルに対して実行できる一連の編集操作が用意されています。使用可能なアクションはファイルのタイプによって異なりますが、常に [元に戻す] および [やり直し] アクションが含まれます。

- 表示メニューを使うと、表示でアクションを実行したり、デフォルトでは開いていない追加の表示を開くことができます。
- デバッグ メニューには、デバッガに関連するアクションが含まれています。

ベーシック エンジン ロボット ファイルを開くと、モードが [デバッグ] に変化し、[ブレークポイント] メニューが表示されます。ブレークポイント メニューには、ブレークポイントの追加や削除など、デバッガのブレークポイントに関連するアクションが含まれています。

ベーシック エンジン ロボットの実行を許可するために [実行の準備] をクリックすると、ほぼすべてのメニュー項目が使用できるようになります。このアクションをクリックすると、ロボットが実行モードになり、編集中にロボットを実行できるようになります。ベーシック エンジン ロボットの実行準備が整っていない場合でも、ロボットの編集は実行できます。ただし、ステップ実行の結果を確認したり、ロボットをデザインするために操作するアプリケーションからデータを取得して使用したりすることはできません。また、デバッグ モードでは複数のベーシック エンジン ロボットを同時に実行できるのに対し、デザイン モードでは 1 つのベーシック エンジン ロボットのみが、実行する権限を持ちます。デザイン モードでの実行権限を持つロボットはデバッグ モードでも同時に実行することができるため、2 つのモードを切り替えることができます。詳細については、「デバッグ モード」を参照してください。

ツールバー

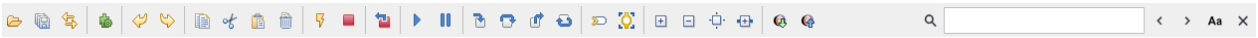
ツールバー ボタンを使用して、メニューで使用できるオプションにすばやくアクセスします。

アクティブなツールバー ボタンとそのツールチップは、Kofax RPA のバージョン、ロボット タイプ、およびアクティブな エディターによって異なります。Design Studio のユーザー インターフェイスも参照してください。

ツールバーのボタンは、ロボット タイプに対応しています。

- ロボットのツールバー
- ベーシック エンジン ロボットのツールバー

ロボットのツールバー



ボタン	ツールチップ	説明
	プロジェクトを開く	プロジェクトを開きます。
	すべて保存	Design Studio で変更されたすべてのファイルを保存します。
	すべて更新	プロジェクト内の変更されたアイテムをすべて更新して、[マイ プロジェクト] ビューを更新します。
	ロボット設定	[ロボットの設定] ウィンドウを開きます。
	元に戻す	最後の変更を削除します。

ボタン	ツールチップ	説明
	やり直し	最後のアクションを繰り返します。また、元に戻すアクションを元に戻します。
	コピー	選択したコンテンツをクリップボードにコピーします。
	切り取り	選択したコンテンツを切り取ります。
	貼り付け	クリップボードの内容を、選択した場所に貼り付けます。
	削除	選択したコンテンツを削除します。
	実行の準備	選択したロボットが実行権限を取得できるように準備します。このボタンを使用してロボットを実行し、ロボットで完全な編集オプションを有効にします。「実行の準備」を参照してください。
	リセット	内部状態をリセットします。 ロボットがスタンドアロン モードで実行されている場合は、ローカル アプリケーションを含む内部状態とデバイスがリセットされます。
	ロボットの終了	ロボットを停止し、最後まで実行したり結果を返したりせずに終了します。
	実行を開始	現在のフロー ポイントからワークフローの実行を開始します。
	停止 - 一時停止	ローカル ハブからの応答を待機している実行を停止し、現在のステップの実行が完了した後に、後続の実行を一時停止します。
	ステップ	次のフロー ポイントに対して実行します。次のフロー ポイントが折りたたまれたステップ内にある場合、ステップは展開されます。「ステップ」を参照してください。
	ステップ オーバー	現在のフロー ポイントの直後のステップを実行します。該当するステップがない場合は、次のフロー ポイントに到達するまで実行されます。 新しく挿入されたステップは、ツールバーの [ステップ] または [ステップ オーバー] をクリックしない限り実行されません。

ボタン	ツールチップ	説明
	ステップ アウト	次のように、現在のフロー ポイントを含むスコープからステップ アウトします。 • 現在のフロー ポイントがロボットの実行の最上位レベルにある場合は、ロボットの最後まで実行されます。 • 現在のフロー ポイントが複合ステップの実行内にある場合は、複合ステップの直後にフロー ポイントを離れて停止します。 • ロボットが終了し、ベーシック エンジン ロボットから呼び出された場合、実行はロボットを離れ、結果またはエラーとともにベーシック エンジン ロボットに戻ります。 ロボットがスタンドアロン モードで実行されている場合、戻るためのベーシック エンジン ロボットがないため、このボタンは使用できません。
	次のイテレーションに移動	現在のフロー ポイントがループ ステップ内にあるときに有効になります。ボタンを押し、同じフロー ポイントに再度達するまで実行します。フロー ポイントが一部のイテレーションでスキップされた場合、ループは複数回実行されます。これ以上実行するイテレーションがなくなると、実行はループ ステップ外のフロー ポイントで停止します。
	指定したロケーションへ移動	ステップに直接移動するためにステップのロケーション コードを貼り付けたり、ステップのロケーション コードをコピーしたりするためのダイアログ ボックスを開きます。 ステップに移動するには、ロボットを開いて [指定したロケーションへ移動] ボタンをクリックするか、Ctrl+G を押します。コードをダイアログ ボックスに貼り付けます。このアクションは、特定のステップのエラーがログに記録されている場合に役立ちます。このログからコードを使用して、ロボット内のステップに移動します。 ステップのロケーション コードをコピーするには、ロボット内でステップを選択し、[指定したロケーションへ移動] ボタンをクリックするか、Ctrl+G を押します。コードをダイアログ ボックスからコピーします。これは、ステップのロケーションを他のユーザーと共有する必要がある場合に役立ちます。
	現在のフロー ポイントに移動	現在のフローポイントの場所を表示します。
	展開	選択したすべてのステップを展開します。何も選択しない場合は、すべてのステップが展開されます。選択したステップまたはすべてのステップが展開されている場合、ボタンは使用できません。
	折りたたむ	選択したすべてのステップを折りたたみます。何も選択しない場合は、すべてのステップが折りたたまれます。選択したステップまたはすべてのステップが折りたたまれている場合、ボタンは使用できません。
	選択以外すべて折りたたむ	選択したステップを除くすべてのステップを折りたたみ、選択したすべてのステップを展開します。選択がない場合、ボタンは使用できません。
	フローを展開	選択したステップのフローを展開します。グループ、ループ、条件付きステップなど、フロー ラインを含むすべてのステップを展開します。何も選択しないと、ロボットの構造全体が展開されます。ロボットのフロー全体が展開されている場合、ボタンは使用できません。
	Management Console からロボットをダウンロード	ロボットをダウンロードします。

ボタン	ツールチップ	説明
	Management Console にロボットをアップロード	ロボットをアップロードします。

ベーシック エンジン ロボットのツールバー

ベーシック エンジン ロボットには、デザイン用およびデバッグ オプション用の 2 つのツールバーがあります。

- デザイン ツールバー
- デバッグ ツールバー

デザイン ツールバー

ボタン	ツールチップ	説明
	プロジェクトを開く	プロジェクトを開きます。
	すべて保存	ワークフローのすべての変更を保存します。
	すべて更新	開いているすべてのロボットを更新します。
	ロボット設定	[ロボットの設定] ウィンドウを開きます。「ベーシック エンジン ロボットの設定」を参照してください。
	実行の準備	選択したロボットが実行権限を取得できるように準備します。このボタンを使用してロボットを実行し、ロボットで完全な編集オプションを有効にします。「実行の準備」を参照してください。
	更新	選択したロボットを更新します。
	停止 - 一時停止	ローカル ハブからの応答を待機している実行を停止し、現在のステップの実行が完了した後に、後続の実行を一時停止します。
	ステップ	次のフロー ポイントに対して実行します。次のフロー ポイントが折りたたまれたステップ内にある場合、ステップは展開されます。「ステップ」を参照してください。
	元に戻す	ロボットへの最後の変更を削除します。
	やり直し	最後のアクションを繰り返します。また、元に戻すアクションを元に戻します。

ボタン	ツールチップ	説明
	切り取り	選択したコンテンツを切り取ります (削除)。
	コピー	ワークフローで選択したコンテンツをクリップボードにコピーします。
	ステップの前に貼り付け	クリップボードの内容をワークフローの選択した場所に貼り付けます。
	削除	選択したコンテンツを削除します。
	選択したステップの前にステップを挿入	選択したステップの前に新しいステップを挿入します。
	選択したステップの後にステップを挿入	選択したステップの後に新しいステップを挿入します。
	選択したステップからブランチを追加	選択したステップに新しいブランチを挿入します。
	グループ化	ステップをグループ化します。
	グループを解除	ステップのグループを削除し、個別のステップにします。
	選択した範囲からスニペットを作成	選択したコンテンツをコピーしてスニペットを作成します。詳細については、スニペットを参照してください。
	スニペットをグループに変換	ステップをスニペットにグループ化します。
	ステップまたは接続を上へ移動	選択したステップまたは接続を上に移動します。
	ステップまたは接続を下へ移動	選択したステップまたは接続を下に移動します。
	すべて広げる	選択したすべてのステップを展開します。何も選択しない場合は、すべてのステップが展開されます。選択したステップまたはすべてのステップが展開されている場合、ボタンは使用できません。
	すべて閉じる	選択したすべてのステップを折りたたみます。何も選択しない場合は、すべてのステップが折りたたまれます。選択したステップまたはすべてのステップが折りたたまれている場合、ボタンは使用できません。

ボタン	ツールチップ	説明
	デバッグモードに切替	デバッグ モード ビューを表示します。「ベーシック エンジン ロボット のデバッグ」を参照してください。
	現在のステップからデバッグ開始	選択した場所からデバッグ ツールを実行します。
	Management Console からロボットをダウンロード	ロボットをダウンロードします。
	Management Console にロボットをアップロード	ロボットをアップロードします。

デバッグ ツールバー

ボタン	ツールチップ	説明
	プロジェクトを開く	プロジェクトを開きます。
	すべて保存	ワークフローのすべての変更を保存します。
	すべて更新	開いているすべてのロボットを更新します。
	実行	現在のフロー ポイントからワークフローの実行を開始します。
	シングル ステップ	一度に 1 つのステップを実行します。「シングル ステップ」を参照してください。
	ステップ	次のフロー ポイントに対して実行します。次のフロー ポイントが折りたたまれたステップ内にある場合、ステップは展開されます。
	ステップ アウト	グループまたはスニペットからステップ アウトします。グループを離れ、グループ ステップの次のステップにデバッガーを進めます。
	停止 - 一時停止	ローカル ハブからの応答を待機している実行を停止し、現在のステップの実行が完了した後に、後続の実行を一時停止します。
	デバッグを再開	ローカル アプリケーションを含む内部状態とデバイスをリセットします。
	ブレークポイントの切り替え	デバッグの際に Design Studio はブレークポイントで停止します。 「ブレークポイントの使用」を参照してください。

ボタン	ツールチップ	説明
	選択したステップ上のブレークポイントを除去	選択したステップのブレークポイントを除去します。
	すべてのブレークポイントの除去	すべてのブレークポイントを除去します。
	すべて広げる	選択したすべてのステップを展開します。何も選択しない場合は、すべてのステップが展開されます。選択したステップまたはすべてのステップが展開されている場合、ボタンは使用できません。
	すべて閉じる	選択したすべてのステップを折りたたみます。何も選択しない場合は、すべてのステップが折りたたまれます。選択したステップまたはすべてのステップが折りたたまれている場合、ボタンは使用できません。
	デザイン モードでこのロケーションに移動	デバッグ ロケーションから [デザイン] モードに戻ります。「デバッグ ロケーションからデザイン モードに戻る」を参照してください。

検索

Design Studio の検索機能を使用して、ロボット内のテキストを検索します。[検索] フィールドに検索するテキストを入力して、Enter キーを押します。検索結果がロボット グラフで強調表示され、[検索結果] パネルに一覧表示されます。

ボタン	ツールチップ
	検索
Aa	大文字と小文字を区別する
<	前へ
>	次へ
×	閉じる

検索結果

現在の結果が選択された状態で、すべての結果が [検索結果] パネルに一覧表示されます。

次のルールが適用されます。

- テキスト検索では、ロボット グラフに表示できるテキストがすべて検索されます。
- すべての検索結果は黄色でハイライト表示されますが、現在の結果はオレンジでハイライト表示されます。
- 検出されたテキストをロボット グラフに表示できない場合は、この結果を含むコンポーネントの展開ボタンがハイライト表示されます。
- F3 キーを押すなど、現在の結果を変更した場合は、ロボット グラフは検出されたテキストまでスクロールします。

- ロボットを変更すると検索が更新され、現在の検索結果がハイライト表示ではなくなります。更新後に F3 キーを押すと、現在の結果 (多くの場合、新しい結果) がハイライト表示されます。
- 式工ディターを開いており、検索結果が含まれている場合は、これらの結果の最初の部分が現在の結果になります。式工ディターを閉じるまで、このダイアログ ボックス内の検索結果をナビゲートします。式工ディター内でテキストを変更すると、このダイアログ ボックスを閉じて再度開くまで、このダイアログ ボックス内の検索結果がすべてのハイライト表示ではなくなります。
- 矢印 › ‹ をクリックするなどの操作を行って検索結果を移動してリストの末尾/先頭に達した場合、移動はリストの先頭/末尾から再度開始されます。

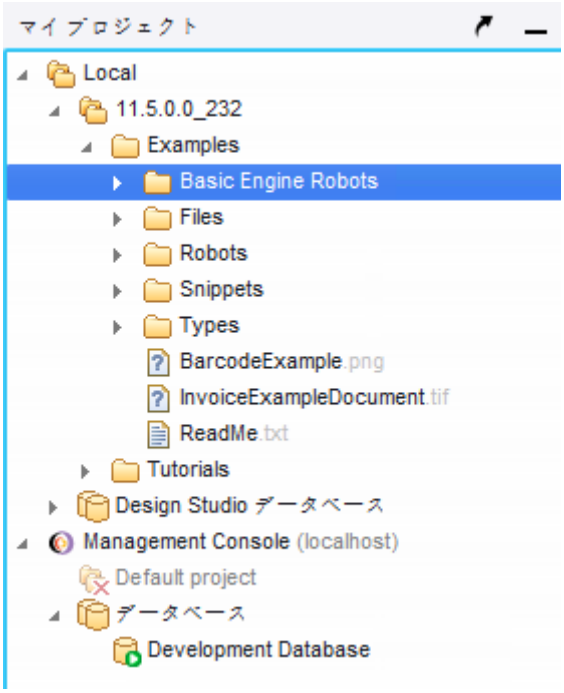
検索結果のナビゲート

次のボタンとショートカットを使用して、検索結果間をナビゲートします。

- 次の検索フィールドに移動する矢印ボタン。
- カーソルが検索ボックス内にある場合は、Enter キーを押すと、検索結果リストが下にスクロールします。
- キーボード ショートカット: 前方に移動する場合は F3 キーを押し、後方に移動する場合は Shift + F3 キーを押します。
- 検索をクリアするには、Esc キーまたは閉じる ✕ ボタンを使用します。

マイ プロジェクト

[マイ プロジェクト] ペインは、Design Studio メイン ウィンドウのツールバーの下にあります。



[マイ プロジェクト] ペインは、次の 2 つの主要なフォルダで構成されています。[ローカル] および Management Console。

- ローカル フォルダ

- Management Console フォルダ

ローカル フォルダ

[ローカル] フォルダには、Design Studio のローカル プロジェクトとデータベースを表す、展開/折りたたみ可能なツリー構造が表示されます。[Design Studio データベース] フォルダには、Design Studio のローカル データベースが表示されます

- このツリーで または をクリックすると、対応するサブツリーが展開または縮小されます。このフォルダには、必要な数の開いているプロジェクトを含めることができます。
- [ローカル] フォルダ内のフォルダまたはファイルを右クリックすると、コンテキスト メニュー オプションが表示されます。標準の Windows メニュー オプションに加えて、このメニューには次の RPA プロジェクト関連のオプションが表示されます。

アクション	説明
新しいプロジェクト	[新しいプロジェクト] ウィンドウを開いて、新しいプロジェクトを作成します。
プロジェクトを開く	プロジェクトの参照や検索を行うための [プロジェクトを開く] ウィンドウが開きます。
すべて閉じる	開いているすべてのプロジェクトとサブフォルダを閉じます。
すべて保存	ツリーのすべてのファイルを保存します。
開く	エディターでファイルを開きます。
更新またはすべて更新	開いているすべてのファイルを更新します。
名前の変更	ファイルの名前を変更するための [ファイル名変更] ダイアログ ボックスを開きます。
移動	ファイルの場所を変更するための [ファイルの移動] ダイアログ ボックスを開きます。
コピー	ファイルを別の場所にコピーするための [ファイル コピー] ダイアログ ボックスを開きます。
削除	ファイルを削除するよう求めるメッセージを表示します。
ファイルの場所を開く	コンピュータ上の格納先のフォルダを開きます。
ロボット ライブラリ ファイルの生成	ロボット ライブラリ ファイルを作成するためのダイアログ ボックスを開きます。このオプションは、プロジェクト フォルダでのみ使用できます。 詳細については、『Kofax RPA 開発者ガイド』 の「ロボット ライブラリ」セクションを参照してください。
アップロード	ファイルのアップロードに使用できるオプションが表示される [Management Console へアップロード] ダイアログ ボックスを開きます。「Management Console へのアップロード」を参照してください。
URL をコピー	このオプションは、ベーシック エンジン ロボット 、スニペット、ロボット 、タイプ、データベース マッピング、オートメーション デバイス マッピング、およびリソースで使用できます。 ファイルの URL をクリップボードにコピーします。URL を開く に示されているように、URL を使用して Design Studio にあるファイルを開きます。

アクション	説明
依存関係を表示	このオプションは、ベーシック エンジン ロボット、スニペット、およびロボットで使用できます。 このファイルに関連付けられているすべてのタイプ、スニペット、ロボットなど、ファイルの既存の依存関係がすべて表示されている [依存関係] ダイアログ ボックスを開きます。 依存関係がローカル プロジェクトに存在しない場合、それらは、リモートのシェア プロジェクトに存在していても、取り消し線付きで表示されます。
ロボットのエクスポート	アクション ステップをロボットに変換するためのオプションを含む [ロボットのエクスポート] ダイアログ ボックスが開きます。「古い Desktop Automation アクション ステップの変換」を参照してください。
アップグレード	ベーシック エンジン ロボットをアップグレードするための [アップグレード] ダイアログ ボックスを開きます。 このアクションにより、ベーシック エンジン ロボットを最新バージョンにアップグレードします。また、クラシック ブラウザで作成したベーシック エンジン ロボットを、WebKit ベースのブラウザを使用するベーシック エンジン ロボットにアップグレードすることもできます。「ベーシック エンジン ロボットのアップグレード」を参照してください。

Management Console フォルダ

[Management Console] セクションには、接続している Management Console のロボット プロジェクトとデータベースを表す、展開および折りたたみ可能なツリー構造が表示されます。

- このツリーで または をクリックすると、対応するサブツリーが展開または縮小されます。
- 最上位の Management Console フォルダまたはサブフォルダを右クリックして、コンテキスト メニュー オプションを表示します。標準の Windows メニュー オプションに加えて、このメニューには次の Management Console オプションが表示されます。

アクション	説明
開く	エディターでファイルを開きます。
更新	開いているすべてのファイルを更新します。
すべてを更新に含める、または更新に含める	選択した Management Console のすべてのプロジェクトを更新に含めるか、1 つのプロジェクトのみを含めます。
すべてを更新から除外、または更新から除外	選択した Management Console のすべてのプロジェクトを更新から除外するか、1 つのプロジェクトのみを除外します。 すべてのプロジェクトの更新には時間がかかる場合があるため、不要になったプロジェクトを更新から除外することをお勧めします。 デフォルトでは、すべてのプロジェクトが更新から除外され、[Management Console] セクションで (更新から除外) とマークされます。
切断	Management Console から切断し、Management Console への接続をリセットします。 Design Studio を使用している場合は、このオプションを使用してユーザーまたはロールを切り替えます。このアクションでは、更新後など、Management Console に再度接続するときに再認証が要求されます。 ブラウザは元のユーザーを記憶しているため、次のユーザーがログインする前に、元のユーザーは Management Console からログアウトする必要があります。

アクション	説明
プロパティ	[Design Studio 設定] ダイアログ ボックスを開きます。
アップロード	新しいファイルを Management Console にアップロードするための [同期] ダイアログ ボックスを開きます。
ダウンロード	Management Console から新しいファイルをダウンロードするための [同期] ダイアログ ボックスを開きます。
同期	Management Console と同期するための [同期] ダイアログ ボックスを開きます。

[Management Console] セクションのプロジェクトがコンピュータの Design Studio と共有されている場合、それらのプロジェクトは [Management Console] セクションにのみ含まれます。

シェア プロジェクト内のファイルの状態に応じて、ダウンロード、更新、および削除されたファイルに表示される内容は異なる場合があります。ユーザーは、さまざまな方法によってローカル プロジェクトを Management Console プロジェクトと 同期 させることができます。

以下の表に、それぞれのステータスのプロジェクト ファイルを示します。[Management Console] セクションには、ヒントや同期の問題に関する説明が表示されます。

アイコン	説明	意味
SimpleExtract.robot	淡色表示のロボット アイコンと名前	シェア プロジェクト内のオブジェクトは、接続された Management Console に存在しています が、Design Studio にはダウンロードされていません。
LoadHelp2.snippet	標準アイコンとオブジェクト名	シェア プロジェクト内のオブジェクトは、リモートの Management Console と同期されています。
~~JSON~~.robot	オブジェクト名に打ち消し線が付いた標準アイコン	このオブジェクトは、コンピュータのプロジェクトで削除されています。
Extracting.robot	名前が太字の通常のアイコン	ファイルがローカルで変更されているため、同期する必要があります。
JSONTest.robot	プラス記号が付いた、名前が太字のアイコン	ローカル プロジェクトの新しいファイルが、Management Console にアップロードされていません。
Article.type	感嘆符付きの黄色のサインがあるオブジェクト アイコン	ローカル コピーとリモート プロジェクトが競合しています。 たとえば、オブジェクトがリモート Management Console で削除されていることが考えられます。 同期時に、競合を解決する方法を選択します。

データベース フォルダと接続

[データベース] フォルダには、接続されている Management Console のデータベースが表示されます。

Management Console との接続を設定するには、**[設定]** > **[Design Studio 設定]** > **[Management Console]** の順に移動します。

データベースは、Management Console のデータベース マッピングでフェッチされます。

データベースを Design Studio の [Management Console] セクションに表示するには、Design Studio ユーザーと共有するクラスタ データベースにデータベース マッピングが存在している必要があります。

マッピングされないクラスタ データベースは、Design Studio に表示されません。

データベース マッピング、タイプ、およびドライバーは、Management Console と Design Studio のコピー間の接続が次のイベント中に確立または更新されるときにのみ Management Console からフェッチされます。

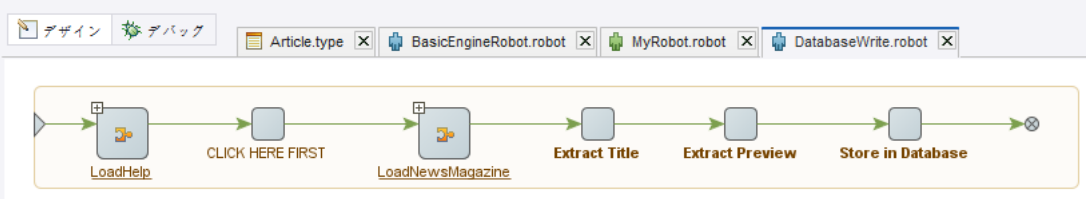
- Management Console 接続の Design Studio への追加。
- 既存の Management Console 接続での Design Studio の開始。
- Management Console 接続の更新。

Management Console 接続を更新するには、[Management Console] セクション ツリーの Management Console ノードを選択し、**[更新]** をクリックします。

エディター ビュー

ロボットとタイプの編集には、エディター ビューを使用します。エディターは同時に複数開くことができますが、表示されるエディターは 1 つだけです。エディターはエディタ ビューの上部にタブとして表示され、タブをクリックすると別のエディターに切り替わります。エディターには 3 種類あります。

- ロボットを編集するロボット エディター。
- 1 つ以上の属性を含むコンプレックス タイプを編集するタイプ エディター。
- プレーン テキスト ファイルを編集するテキスト エディター。



ロボット エディター

ロボットを編集するには、ロボット エディターを使用します。ロボットを開くと、新たなタブに配置された新しいロボット エディターにそのロボットが表示されます。ツールバーとボタンの詳細については、ツールバーを参照してください。

ロボット のエディター
ロボット エディターには、ステップのワークフロー、変数、式、ツリー モードの設定が含まれています。メニュー下のツールバーのボタンは、ワークフローのステップをナビゲートするのに使用します。[実行を開始]、[ステップ]、[ステップ オーバー]、および [ステップ アウト] ボタンを使用してワーク

フローを進めます。ロボットの実行を一時停止またはリセットします。詳細については、「ロボットの編集」を参照してください。

ベーシック エンジン ロボット のエディター

ロボット エディターには、デザイン (デフォルト モード) とデバッグの 2 種類のモードがあります。ロボット エディターの左隅にあるモード ボタンをクリックしてモードを選択します。選択するモードによって、オプションの外観と使用できる項目が異なります。

ベーシック エンジン ロボットの場合、各ビューはサブビューで構成されています。デザイン モードの場合、サブビューは次のとおりです。

- ロボット ビュー
- アプリケーション ビュー
- プロパティ ペイン
- 状態ペインとデータの状態ペイン
- フレーム ビュー

ロボットで作業しているときは、さまざまなステータス メッセージが Design Studio の左下隅に表示されます。これらのメッセージの意味については、ステータス バーを参照してください。

ロボット ビュー

ロボット ビューは、ロボット エディターの上部にあるタブの下に配置されています。ロボット ビューには、ロボット プログラム (ロボットを構成するステップと接続) が表示されます。このビューでロボット ステップをナビゲートします。ステップを選択し、ステップの削除、移動、変更、またはグループ化などを実行して、構造を編集します。

現在のステップ

ロボット ビューには、「現在のステップ」という概念があります。現在のステップにより、この実行と Design Studio 内の位置をマークします。

現在のステップは緑色でマークされます。ステップをクリックすると、そのステップまでのロボットが実行されます。選択したステップが現在のステップになります。実行がステップに達すると、そのステップが新しい現在のステップとなって緑色で表示されます。クリックしたステップまで実行が到達できない場合 (HTML ページが読み込まれない場合など)、有効なステップで実行が停止し、これが新たな現在のステップになります。すでにロボットが実行しているステップをクリックした場合には、実行は発生せずに新しいステップが新しい現在のステップになります。

現在の実行パス

現在の実行パスは、現在のステップに到達するために実行されたロボット内のパスです。

アイテムの選択

一連のステップを選択するには、Ctrl キーを押したままアイテムをクリックします。現在選択されているステップと接続を選択解除するには、ロボットの外側の任意の場所をクリックします。

アクションの編集

ロボット エディターを使用すると、ステップに対して長い範囲のアクションを実行できます。こうしたアクションには、コピー、貼り付け、切り取り、削除などの標準の編集アクションや、デザイン モードでのロボットの実行に関するアクションがあります。現在のステップ (他のステップを選択していない場合)、または選択したステップに対してアクションを実行できます。対応するツールバー ボタンをクリックするか、選択したエレメントのコンテキスト メニューを使って、アクションを実行します。

詳細については、「ロボットの編集」(ロボット の場合) および「一般編集」(ベーシック エンジン ロボット の場合) を参照してください。

レコーダー ビューとアプリケーション ペイン

レコーダー ビュー (ロボット の場合) またはアプリケーション ペイン (ベーシック エンジン ロボット の場合) は、ロボット エディターのロボット ビューの下にあります。

レコーダー ビュー (ロボット)。

[アプリケーションを開く] ウィンドウを含むタブと、利用可能なエレメントを含むツリーを表示します。インターフェイスでエレメントを選択したり、イメージを選択し、選択したエレメントまたはイメージを右クリックして、ステップを挿入できます。レコーダー ビューの下部には、アプリケーション ウィンドウの左上隅を基準としたマウスの座標と、デバイス状態のライブ ストリーミング ステータスが表示されます。詳細については、ロボットの編集を参照してください。

アプリケーション ペイン (ベーシック エンジン ロボット)

アプリケーション ペインには、現在のロボット状態の一部が表示されます。ロボット状態によっては、表示する際にページをロードする必要があります。表示された状態は、現在のステップへの入力状態です。アプリケーション ペインには、現在のロボット状態におけるウィンドウのページ ビューが表示されます。URL からロードすると、それぞれ 1 つのページが含まれているウィンドウが複数開く場合があります。現在のウィンドウはオレンジ色のボックスでマークされています。

開いたページに非 HTML エレメントが含まれる場合、コンテンツのタイプに応じて、ページをプレビューできます。[プレビュー] ボタンを使用して、コンテンツのタイプを変更します。CSV、JSON、テキスト、Excel、XML、およびバイナリのコンテンツをプレビューして、これらにステップを適用できます。

アプリケーション ビューで適用された XSLT 変換を使って XML コンテンツを表示するには、**[ロボット設定]** > **[基本]** タブを選択します。[デフォルト オプション] の横にある [設定] をクリックし、[レガシー] タブに移動して、[フォーマット処理] 設定で [クラシック ローディング] を選択し、**[XML から HTML へ変換]** オプションをオフにします。

現在のステップの状態の Cookie ビューを表示するには、ビュー メニューから Cookie ウィンドウを開きます。Cookie を使用する Web ページがロボットでロードされるときに、Cookie がこのリストに追加されます。

同様に、ビュー メニューから認証情報ウィンドウを開いて、現在の状態の認証情報を確認できます。

プロパティ ペイン

このトピックの情報は、ベーシック エンジン ロボット にのみ適用されます。

プロパティ ペインには、現在のステップの設定が表示されます。タブをクリックすると、プロパティを表示して、編集することができます。

[基本] タブ

このタブには、ステップの名前と関連するコメントが含まれます。コメントを付けたステップは、ロボット ビューに太字の名前で表示されます。マウス ポインタをステップ上に置くと、コメントが表示されます。

[ファインダー] タブ
このタブには、ステップのアクションが使用する必要があるタグや範囲を検索するためにステップが使用するタグ ファインダーまたは範囲ファインダーが含まれています。ページ ビューで要素を右クリックして、ファインダーを設定します。詳細については、「タグ ファインダーの使用」を参照してください。

[アクション] タブ
このタブでは、ステップのアクションを表示し、設定することができます。使用可能なアクションの説明については、ステップ アクションとデータ コンバータを参照してください。

[エラー処理] タブ
このタブには、このステップの実行中に発生する エラーを処理する 方法をコントロールするプロパティが含まれています。

状態ペインとデータの状態ペイン

ロボット の状態ペインおよびベーシック エンジン ロボット のデータの状態ペインには、変数およびその他の要素のリストが表示されるタブがあります。

状態ペイン (ロボット)
ワークフローの実行状態と、入力、変数、ファインダー、データベース、戻り値、出力値などの含まれる要素を表示します。詳細については、「ロボットの編集」を参照してください。

データの状態ペイン (ベーシック エンジン ロボット)
[変数]
リストから変数を選択すると、関連付けられている詳細がタブの右側に表示されます。このタブには、ロボット実行の現在のステップの変数値が表示されますが、これらを編集することはできません。

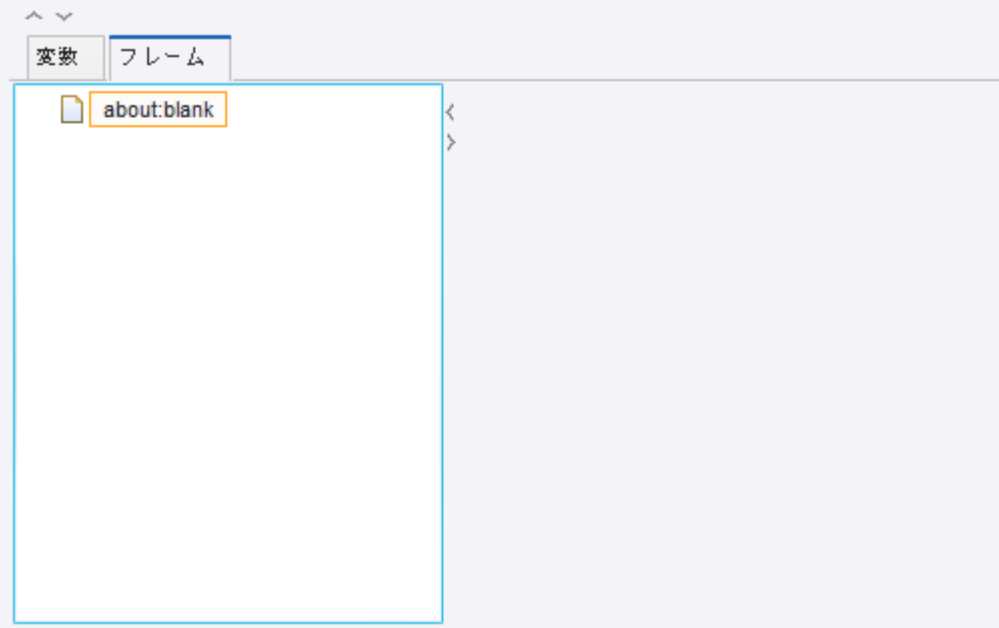
- 変数リストを右クリックして、変数タイプのリストにアクセスします。このリストを使用して、変数タイプを追加または除去することができます。また、このリストを使用して、選択した変数を除去できます。
- [編集] をクリックして、初期変数値を変更するか、変数リストのアイテムをダブルクリックします。表示されるダイアログ ボックスには、ステップを実行する前の変数の値が表示されます。これらの値は編集することができます。

ロボットの記述とテストを行うときは、初期入力変数の値を使用します。実稼働環境でロボットを実行するときは、入力変数が、ロボットを実行しているアプリケーションによって決定された値に初期化されます。

> アプリケーションが値を提供しない場合、ロボットの実行が失敗します。

変数の初期値は、ロボットの起動時 (たとえば、開始ステップ) に変数が保持している値です。実稼働環境でロボットを記述、テスト、および実行するときに、これらの値が適用されます。

[フレーム]
[フレーム] タブは、[変数] タブの隣にあります。



フレーム タブには、ツリー内のすべての最上位ブラウザ フレームとそのすべてのサブフレームが表示されます。また、このタブには、選択したフレームの詳細を表示するプレビュー パネルが含まれてます。フレームの概要が表示されるのは、[フレーム] タブのみです。フレーム ツリーの最上位フレームのラベルは、ページ ビューのタブ タイトルと同じです。フレームに表示された HTML ページにタイトルがある場合はタイトルが表示され、タイトルがない場合は URL が表示されます。サブフレームのラベルは名前で表示されます (名前がないものは無名)。

フレーム ツリー内のノードは、次のようなさまざまな修飾子を持ちます：

- ラベルの周りのオレンジ色のボックス：フレームは現在のウィンドウです。
- ラベルの周りの灰色のボックス：フレームは現在ページ ビューで選択されています。
- ラベルの周りの明るい灰色の背景色：フレームはページ ビューで開いています。
- ラベルとアイコンが淡色表示: フレームにはビューがありません (ビューポートの高さがゼロまたは幅がゼロ)。

[フレーム] ツリーの横にある [フレーム] プレビュー パネルには、ツリー フレームの選択内容に関する詳細が表示されます。表示される詳細は、URL と、1263 × 1024 などのフレーム サイズで重ねて表示されるフレームのブラウザ ビューの小さなレンダリングです。URL ブロッキングによってフレームがブロックされている場合、プレビューとツリーで ⊘ が表示されます。

[フレーム] プレビュー パネルは、デフォルトのブラウザ エンジンを使ってデザインされたロボットでのみ利用できます。

フレーム ビュー アクション

フレーム ツリーのノードに関連したアクションは多数存在します。

- [現在のウィンドウとして設定]:「現在のウィンドウに設定」ステップを、選択済みのノード名でフレームを開くように設定されたロボットに挿入します (名前はノード上のツールチップに表示されます)。
- [ウィンドウを閉じる]: ロボットにステップを挿入し、フレームを閉じます。

- [開く/閉じる]: ページ ビュー (タブ) でフレームを開閉します。最上位のフレームは常に開いているため、最上位にないフレームでのみ動作します。このコマンドは、ロボットにステップを挿入しないことに注意してください。
- [**URL** をブロック]: フレームの URL ブロック パターンを編集するためのダイアログ ボックスを開き、ロボットの [ブロックする URL のパターン] リストにこのパターンを追加します。
- [ブラウザ ビュー内を選択]: フレームを定義したブラウザ ビューでフレーム エレメントを選択します。エレメントを含むフレームがページ ビューで開いていない場合は、このフレームが開きます。

これらのアクションは、ページ ビューのブラウザ ビュー タブでも利用できます。

タイプ エディター

タイプ エディターを使用して、ロボットの新しいタイプの作成と、現在編集されているタイプの設定を行います。属性テーブルの下にあるボタンを使用して、タイプへの新しい属性の追加、属性の除去、順序の変更、および属性の設定を行うことができます。

詳細については、タイプを設定を参照してください。

テキスト エディター

テキスト エディターは、ReadMe ファイルなどのプレーン テキスト ファイル (.txt) を編集するためのシンプルなエディターです。エディターで開くことができる拡張子は、.txt、.java、.jsp、.js、.log、.html、.xml、および .csv です。エディターは、これらの拡張子のファイルがファイル内容について示す情報を使用しません。ファイルはすべて、プレーン テキスト ファイル (シンタックス ハイライトなし) として扱われます。

Design Studio のウィンドウ

Design Studio のウィンドウはドッキング可能です。ウィンドウを移動したり、サイズを変更したり、端にドッキングしたりできます。これにより、保存して後で使用できるカスタム レイアウトが作成されます。

- ウィンドウのドッキングを解除するには、[フローティングの切り替え] ボタンをクリックします。フローティング ウィンドウのタイトル バーをドラッグするとウィンドウを移動できます。ウィンドウをドッキングするには、 ボタンをクリックします。
- 変更を保存するには、メニューで **[ウィンドウ]** > **[レイアウトを保存]** をクリックします。

保存されたすべてのレイアウトは、[ウィンドウ] ドロップダウン メニューの上部のリストに表示されます。

- レイアウトをアップロードするには、いずれかのレイアウトをクリックして、[レイアウトをロード] を選択します。
- リストからアイテムを削除するには、レイアウトをクリックして、[レイアウトの削除] を選択します。
- Design Studio のウィンドウ レイアウトをデフォルトにリセットするには、**[ウィンドウ]** > **[レイアウト初期化]** をクリックします。

Design Studio を終了すると現在のレイアウトが自動的に保存されて、Design Studio を再び開くとレイアウトがリセットされます。

ステータス バー

ロボットで作業しているときに、次のステータス メッセージが左下隅に表示されることがあります。

ステータス メッセージ	説明	表示モード
準備完了	ロボットは実行する準備が整っています。	デザインとデバッグ
実行を準備中	ロボットが実行を開始しようとしています。	デバッグ
実行を完了中	ロボットが実行を終了しようとしています。	デザインとデバッグ
実行中	ロボットが実行されています。	デザインとデバッグ
開始場所まで実行中	デザイン モードの [現在のステップからデバッグ開始] ボタンで開始すると、ロボットが現在選択されているステップに移動します。	デバッグ
開始場所に到達しました	デザイン モードの [現在のステップからデバッグ開始] ボタンで選択したステップにロボットが到達しました。	デバッグ
開始場所に到達できませんでした	デザイン モードの [現在のステップからデバッグ開始] ボタンで選択したステップにロボットが到達できませんでした。	デバッグ
実行は正常に完了しました	ロボットはすべてのステップを正常に実行しました。	デバッグ
実行は完了しましたがエラーが発生しました	ロボットの実行がエラーで終了しました。	デバッグ
停止しました	[一時停止] ボタンをクリックした後に、ロボットが停止しました。	デバッグ
API 例外の報告後に停止しました	API エラーが報告された後、ロボットが停止しました。	デザインとデバッグ
ブレークポイントで停止しました	[ブレークポイントの切り替え] ボタンで設定したブレークポイントでロボットが停止しました。	デバッグ
次の表示可能なステップで停止しました	現在の位置マーカーの後に次の表示可能なステップが見つかったときに、ロボットが停止しました。このメッセージは、[シングル ステップ] ボタンを使用したときに表示されます。	デバッグ
次のステップで停止しました	グループ内で次のステップが非表示になっている場合でも、現在の位置マーカーの後の次のステップでロボットが停止しました。このメッセージは、「ステップ」ボタンを使用したときに表示されます。	デバッグ
指定したステップで停止しました	ロボットは指定したステップで停止しました。このメッセージは、[ステップ アウト] ボタンを使用したときに表示されます。	デバッグ

プロジェクトとライブラリ

Design Studio で作業する場合、任意の数のプロジェクトを必要に応じて開くことができます。プロジェクトの目的は、ロボットの集合体とロボットが必要とするファイルを含んだライブラリを開発することにあります。通常は、ロボットを使用するユーザーの企業におけるそれぞれの用途に対して 1 つのプロジェクトといったように、ロボットの用途ごとにプロジェクトを作成します。2 つのプロジェクトでファイルを共有することはできません。タイプは常に 1 つのプロジェクトに属し、タイプのスコープはそのタイプが属するプロジェクトのものに限定されます。

プロジェクトとは、ファイル システム内の任意の場所にあるフォルダです。プロジェクト フォルダには任意の名前を付けることができますが、Library サブフォルダを含める必要があります。

名前の付け方について、詳しくは命名規則を参照してください。

Library

このフォルダにはプロジェクトのライブラリが含まれます。

ロボット ライブラリから [Library] フォルダにロードされたファイルなど、ロボットが使用するロボット ファイル、タイプ ファイル、およびその他のファイルすべてを配置します。適切なサブフォルダを使って、[Library] フォルダ内のファイルを整理することができます。

次の例は、ニュース サイトからニュースを抽出し、また株式サイトから株式相場を抽出するための ロボット ライブラリを開発するプロジェクトに対して、NewsAndStocksProject と名前を付けたプロジェクト フォルダを表しています。

```
NewsAndStocksProject/
    Library/
       News/
          CNN.robot
          Reuters.robot
          News.type
       Stocks/
          Nasdaq.robot
          NYSE.robot
          Stocks.type
```

このプロジェクトには、ロボットとタイプ ファイルが、それぞれニュース フォルダと株式サブ フォルダとして分けられた [Library] フォルダがあることに注意してください。

Design Studio を閉じると、開かれているプロジェクトとファイルが記憶されます。次回 Design Studio を開いたときに、同じプロジェクトとファイルが開きます。

現在のプロジェクト

Design Studio では、多くのプロジェクトを使用できますが、RoboServer などの Kofax RPA の他のアプリケーションは必ず特定のプロジェクト (現在のプロジェクト) で動作します。Kofax RPA をインストールすると、デフォルトのプロジェクトが作成されます。このプロジェクトは現在のプロジェクトとして選択されます。初めて Design Studio を開くと、この現在のプロジェクトが唯一のプロジェクトとして開かれた状態になります。Design Studio を閉じる前にプロジェクトをすべて閉じると、次回 Design Studio を開いたときに、選択された現在のプロジェクトが開きます。

現在のプロジェクト選択を変更するには、[設定] アプリケーションを使用して [プロジェクト] タブの [現在のプロジェクト フォルダ] プロパティに新規作成プロジェクト フォルダへのパスを指定し、[OK] をクリックして設定を閉じます。詳細については、『Kofax RPA 開発者ガイド』を参照してください。

シェアプロジェクト

シェア プロジェクトは Management Console に展開され、ローカル Design Studio コンピュータのプロジェクトに接続されるプロジェクトです。Management Console プロジェクトは複数の Design Studio 間で共有できるため、複数のユーザーがプロジェクトを編集できます。シェア プロジェクトが Management Console 上のプロジェクトと同期していない場合、マイ プロジェクト ペインの Management Console セクションにはプロジェクト内の各オブジェクトのステータスが視覚的に表示されます。ローカル コピーを Management Console に展開されたものと同期する場合には、異なる方法を用いることができます。

ロボット プロジェクトの操作

以下の手順を使用して、プロジェクトを開いたり、閉じたり、作成したりします。

- 既存のプロジェクトを開くには、[ファイル] メニューから [プロジェクトを開く] を選択し、プロジェクト フォルダを選択します。[プロジェクトを開く] ウィンドウが表示されます。
- プロジェクトを閉じるには、[マイ プロジェクト] ビューでプロジェクトを右クリックします。[プロジェクト] ウィンドウが表示されます。[閉じる] をクリックします。

  [ファイル] メニューからすべてのプロジェクトを閉じることもできます。
- 新しいプロジェクトを作成するには、次の手順を実行します。

1. [ファイル] メニューから [新しいプロジェクト] を選択します。

   [新規作成プロジェクト] ウィンドウが表示されます。
2. プロジェクトの名前とロケーションを入力します。
3. [終了] をクリックします。

   指定したロケーションに新しいプロジェクトが作成されます。プロジェクトのフォルダ名は、プロジェクトに割り当てた名前と同じです。

   例

   名前 MyProject とロケーション`C:/KofaxRPAProjects` を入力した場合は、次のフォルダが作成されます。

   `C:/KofaxRPAProjects/MyProject`

   `C:/KofaxRPAProjects/MyProject/Library`

ロボット ファイルの整理

RoboServer などのランタイム環境でロボット ライブラリを分配およびデプロイするときに、ロボット ライブラリ ファイルと呼ばれる単一のファイルにロボット ライブラリをパックすることができます。

これにより、現在のエディターでファイルのロボット ライブラリに含まれるすべてのファイルがパックされ、単一のファイルとして指定した名前で結果が保存されます。ロボット ライブラリ ファイルを作成する前に、最新の変更を含めるために、ロボットやタイプなど開いているすべてのファイルを保存します。

RoboServer がロボット ライブラリ ファイルを使用できるようにして、ロボット ライブラリからロボットを実行することができます。詳細については、『Kofax RPA 開発者ガイド』を参照してください。

1. Design Studio で、ロボットやタイプなど、開いているすべてのプロジェクト ファイルを保存します。
2. [ツール] メニューから [ロボット ライブラリ ファイルの生成] を選択します。

[ロボット ライブラリ出力ファイル選択] が表示されます。

3. ライブラリに使用する場所に移動します。
   ツールバーのアイコンを使用して、詳細ビューまたはリスト ビューに変更したり、1 レベル上に移動したり、新しいフォルダを作成したりします。
4. [ファイル名] フィールドにライブラリの名前を入力します。
5. [**OK**] をクリックします。
   システムにより、ロボット ライブラリ ファイルが生成されます。
6. [**OK**] をクリックします。

シェア プロジェクトの使用

1 つまたは複数の Management Console に接続すると、[マイ プロジェクト] ペインの [Management Console] セクションに、アクセス可能なすべての Management Console に展開されたすべてのプロジェクトが表示されます。ローカル コンピュータにプロジェクトとオブジェクトがダウンロードされていない場合、これらのプロジェクトとオブジェクトはリストに表示されますが、使用することはできません。Design Studio はこうしたプロジェクトをトラックしません。

Management Console へのプロジェクトのアップロード

プロジェクトを Management Console にアップロードするには、次の手順を実行します。

1. ローカル フォルダ でプロジェクトを右クリックし、コンテキスト メニューで [アップロード] を選択するか、プロジェクトを選択し [ツール] メニューで [**Management Console** へアップロード] をクリックします。
2. [Management Console へアップロード] ウィンドウで、ファイルのアップロード先の Management Console とプロジェクトを選択します。
   このプロジェクトをシェア プロジェクトとして保持し、Design Studio と Management Console の間で同期する場合は、[これを記憶する **(**シェア プロジェクトとして**)**] をクリックします。
3. [アップロード] をクリックして手順を完了します。

プロジェクトを Management Console にアップロードすると、[管理] > [プロジェクト] タブにプロジェクトが表示され、選択した Management Console の [リポジトリ] タブにすべてのプロジェクト ファイルが表示されます。

Management Console からのプロジェクトのダウンロード

Management Console からプロジェクトをダウンロードするには、次の手順を実行します。

1. マイ プロジェクト ペインの Management Console セクションの下のプロジェクトを右クリックし、コンテキスト メニューで [ダウンロード] を選択するか、プロジェクトを選択し、[ツール] メニューで [**Management Console** からダウンロード] をクリックします。
2. [プロジェクトの名前と場所を選択] ウィンドウでプロジェクトの名前とロケーションを選択します。
3. [終了] をクリックしてプロジェクトをダウンロードします。

Management Console からプロジェクトをダウンロードすると、プロジェクトが ローカル フォルダに表示され、ローカルでプロジェクト ファイルを編集できるようになります。

プロジェクトの同期

コンピュータでシェア プロジェクトのファイルを編集した後、ローカル ファイルと Management Console にデプロイされたファイルを同期することができます。複数のユーザーがシェア プロジェクトにアクセスできるため、同期の競合が発生する可能性があります。Design Studio に、競合の種類とその解決方法を知らせるメッセージと説明が表示されます。タイプとスニペットなどの依存したファイルが原因で、ロボットが適切に機能しなくなる場合があります。[ダウンロード] を使用してプロジェクトを同期した場合、ファイルは Management Console からダウンロードされ、ローカルで加えた変更が失われます。[アップロード] を使用した場合、ローカル ファイルが Management Console にアップロードされ、他のユーザーが加えた変更が失われます (ただし、これらの変更は他のユーザーのコンピュータに保存されていることもあります)。競合が発生している状態で、ユーザー自身または他のユーザーが加えた変更が失われた場合、Design Studio により、同期オプションを選択するための [同期] ウィンドウが表示されます。

次の表は、同期の例を示しています。

ステータス	同期オプション	結果
シェア プロジェクトのファイルを自分のコンピュータで編集しています。Management Console の同じプロジェクトにアクセスできる他のユーザーの中に、ファイルを編集したユーザーはいません。	アップロード	変更は、Management Console でシェア プロジェクトにアップロードされます。[同期] を選択した場合、デフォルト オプションでは、変更が Management Console にアップロードされます。
Management Console でシェア プロジェクトのファイルが変更されます。ファイルを編集したユーザーと変更点を確認できます。	ダウンロード	Management Console からの変更済みファイルが、ローカル プロジェクトにダウンロードされます。
自分のコンピュータでシェア プロジェクトのファイルを編集しています。ファイルを編集しているときに、他のユーザーが同じファイルを編集して Management Console にアップロードしています。	同期	これは競合が発生した状態であり、保持する変更を決定する必要があります。[同期] ウィンドウで、変更を Management Console にアップロードするか、Management Console からファイルをダウンロードするか、またはファイルを Management Console と同期せずに保持するかを選択できます。

データベースの操作

Design Studio を使用してデータベースを操作できます。詳しくは、以下のトピックを参照してください。

- データベースのマッピング
- タイプとデータベース
- データベース警告
- データベース テーブルの生成
- データベースへのデータ格納

データベースのマッピング

ロボットは、さまざまなデータベース アクセス ステップ (データベース データ登録など) を介してデータベースにアクセスすることが必要な場合があります。これらのステップには、名前付きデータベースへのリファレンスを提供する必要があります。RoboServer でロボットが正常に実行されるようにするには、ロボットが使用する名前付きのデータベースに RoboServer からアクセスできる必要があります。

Design Studio でのロボットの設計中は、RoboServer からは利用できないローカル データベースを使用すると便利です。ロボットのデプロイ前に、データベースにアクセスするさまざまなステップ上の名前付きデータベースを変更することを覚えている必要はありません。Design Studio には、この問題を解決するために追加された抽象化のレイヤーである、データベース マッピングがあります。このマッピング メカニズムにより、ロボットのデータベース アクセス ステップで名前付きデータベースは Design Studio データベースにマッピングされます。Design Studio 内からロボットを実行する限り、データベースにアクセスするステップの名前付きデータベースは、このマッピングで指定された Design Studio データベースにマッピングされます。Design Studio のユーザーは、データベースにアクセスするステップ上の名前付き参照先データベースをロボットのデプロイ前に変更する必要はありません。ロボットの設計やテストと同時にローカル データベースを使用できます。

また、データベース マッピングを使用すると、Design Studio のユーザーは、異なるデータベースでロボット ストア値を簡単に作成できます。この操作は、異なるデータベースをポイントするようにマッピングを再設定するだけで済みます。

データベース マッピングは小規模の設定ファイルで、マッピング先のデータベースや、ユーザーがマッピングと参照先データベースを正しく設定できるようなさまざまな警告を Design Studio で表示するかどうかを定義します。マッピングの名前は、設定ファイルのファイル名です。つまり、ファイル名 "objectdb" を使ってマッピングを作成した場合、マッピングがポイントするデータベースは、ロボットで "objectdb" という名前でアクセス可能になります。

データベース名は複数の Management Console で同一となる場合がありますが、Design Studio でデータベース マッピングを作成する際に区別するために、リストのデータベース名には Management Console の名前が含まれています。

次の手順は、Design Studio でデータベース マッピングを作成するいくつかの方法を示しています。

1. [ファイル] メニューから [新しいデータベース マッピング] を選択します。
   ウィザードが表示されます。
2. データベースとプロジェクトを選択し、[次へ] をクリックします。
3. 一意のデータベース マッピングの名前を入力し、[終了] をクリックします。
   ウィザードが完了すると、選択したプロジェクトとフォルダにマッピングが作成されます。

データベース ビュー

1. データベース ビューで、プロジェクトと関連付けるデータベースを右クリックします。
2. [プロジェクトに追加] を選択し、データベースを追加するプロジェクトを選択します。
3. データベース マッピングに使用する一意の名前を入力します。これはマッピング ファイルの名前で、この名前でデータベースにアクセスします。
   名前の候補が表示されることに注意してください。これは、デフォルトのデータベース名であり、Design Studio とは別に、他の Kofax RPA アプリケーションでこのデータベースにアクセスするときに使用する名前です。

マッピングされていないデータベース

Design Studio では、マッピングがないデータベースを使用するロボットを開くと、警告が表示されます。

1. マッピングされていないデータベースを使用するロボットを開きます。

   警告が表示され、ロボットで参照されているデータベースの名前を持つマッピングが推奨されます。これにより、他のデータベースを定義している開発者から送信されたロボットを変更することなく、すぐに実行できます。
2. ウィザードの手順を完了します。

タイプとデータベース

ロボットが変数の値をデータベースに書き込む場合、これらの変数のタイプで、どの属性が値をデータベースに格納するのに使うキーの一部になるのか、定義する必要があります。値に対するデータベース キーは、データベース キーの一部としてマークされた属性のセキュアなハッシュとして計算されます。

また、属性定義の一部にストレージ名を指定することもできます。これは、属性を格納するときに使用するオプションの別名です。

[データベース データ登録] アクションを使用してデータベース ストレージに値を保存するときは、適切なデータベース テーブルが使用可能なデータベースに存在する必要があります。テーブルには、タイプの属性に一致するカラムを入れる必要があります。

⚠ データベース データ登録または SQL 実行ステップで使用されるデータベースのテーブルを削除したり変更したりしないでください。

Design Studio による適切なデータベース テーブルの設定のサポート方法については、データベース テーブルの生成を参照してください。Design Studio でのデータベース接続の設定については、Design Studio での設定を参照してください。

データベース警告

データベース警告は、データベース マッピング、ロボット、および参照されるデータベースを正しく設定する場合に便利です。警告システムでは、タイプ検証問題、不足したテーブル、不足したデータベース マッピングなどの潜在的な問題が自動的に監視され、問題が発生した場合、警告メッセージがステータス バーに表示されます。

また、警告システムでは、ロボットで使用されるデータベース名が監視され、不足したマッピングの作成がサポートされます。システムではデータベースの表面的な監視が実行されます。つまり、データベースに継続的に ping 送信をしてデータベースがオンラインかどうかを確認するのではなく、必要に応じて情報を更新します。システムでは関連するデータベース テーブルのテーブル構造がキャッシュされ、このキャッシュはデータベース クエリの数が過剰にならないように警告を計算するために使用されます。キャッシュは、Design Studio でキャッシュが要求される何らかのイベントの発生が認識された場合に再作成されます。データベース テーブルの外部修正またはデータベースの利用可能性は監視されません。単一のデータベース、またはすべてのデータベースに対しデータベース キャッシュを再構築するオプションがあります。このオプションは、データベース ビュー、および該当する場合は警告で利用できます。たとえば、データベースのキャッシュを再作成するには、データベース ビューを右クリックし、[更新] を選択します。

データベース テーブルの生成

抽出した変数値をデータベース データ登録するには、データベースに一致するテーブルを作成します。Design Studio は、作成したタイプを検査して適切な SQL を生成することにより、これらのテーブルの作成をサポートします。一部のタイプの変数の値を保存するときは、そのタイプを表すテーブルがデータベースに存在している必要があります。

データベース エラーを回避するため、変数名には予約名を使用しないでください。詳細については、命名規則 を参照してください。

1. ロボットまたはタイプを開き、[ツール] メニューで [データベース テーブルの生成] を選択します。
   [データベース テーブルの生成] ウィンドウが表示されます。

   組み込みの Kofax RPA 開発用データベースを使用する場合は、データベース テーブルを作成する前にデータベースを起動します。データベースを起動するには、[**Design Studio** データベース] の下の [開発用データベース] 項目を右クリックし、[ローカルの開発用データベースを起動] を選択します。
2. [データベース] リストからデータベースを選択します。リストには、作成したデータベースが含まれます。データベースの作成方法については、データベースのマッピングを参照してください。
3. データベース テーブルを作成するタイプを選択します。
4. [**SQL** を生成] をクリックします。
   システムにより、テーブルを作成するための SQL ステートメントが提示されます。
5. 既存のデータベースで各 'CREATE TABLE' ステートメントの前に 'DROP TABLE' ステートメントを含めてテーブルを再作成するには、[存在する場合テーブルを削除] を選択して、対応するテーブルがすでに存在する場合は削除されるようにします。
6. ステートメントを変更、実行、または保存します。
   表示された SQL は推奨される提示です。必要に応じて、ニーズに合わせてステートメントを変更できます。たとえば、ショート テキスト属性の列タイプを "VARCHAR(255)" から "VARCHAR(50)" に変更してデータベース領域を節約したり、自動増分プライマリ キーを追加したりできます。ただし、通常の状況では、テーブル名や列名を変更したり、列を除去したりする必要はありません。

データベースへのデータ格納

このセクションでは、Kofax RPA データベース ストレージの仕組みについて説明します。

オブジェクト キー

データベースのタイプに対して作成したテーブルには、自分のタイプの各属性に対応する列と、以下の 7 つの監査データ フィールドがあります。ObjectKey、RobotName、ExecutionId、FirstExtracted、LastExtracted、 ExtractedInLastRun、および LastUpdated。ObjectKey はテーブルの主キーになるため、最も重要なフィールドです。

"ObjectKey" という名前の理由は、以前 Kofax RPA で使用されていた用語にあります。以前は、タイプと変数は「オブジェクト」と呼ばれていました。新しい用語を使うのであれば、"ObjectKey" を "ValueKey" と呼びます。しかし、名前を変更すると後に互換性の問題が多発することから、古い名前を保持することが許容されています。

タイプの ObjectKey は、そのタイプの変数から抽出された値を、データベースに格納されているときに一意に識別するためものです。タイプの値を一意に識別するものを把握しておく必要があります。車のリポジトリを構築する場合は、VIN コードだけで十分に各車両を一意に識別できるでしょう。野球の結果を収集する場合は、各試合を一意に識別するために、年、チーム名、球場、日付が必要になるでしょう。

タイプを構築するときには、ObjectKey の計算方法を選択することができます。新しい属性の作成時に「データベース キーの一部」オプションにチェックを入れて選択します。先ほどの車の例では、VIN コードのみがデータベース キーの一部としてマークされた属性となり、野球の試合を例にとると、年、チーム名、球場、および日付の属性は、すべてデータベース キーの一部としてマークされます。

ロボットの開発者は、データベース データ登録アクションで直接キーを指定し、タイプに定義されているデフォルトのアルゴリズムをオーバーライドすることもできます。

データベース キーの一部ではない属性は、非キー フィールドと呼ばれます。たとえば、車に価格の属性がある場合、価格が変更されても、同一の車と見なされます。

データベース データ登録

Kofax RPA には、データベース内の値を管理する次のいくつかのアクションがあります: データベース データ登録、データベース データ抽出、キーの計算、データベース データ削除、データベース照会、および SQL 実行。検索とデータベース照会の操作はシンプルですが、データベース データ登録と SQL の実行は、値を格納するだけではありません。

データベースデータ登録と SQL の実行は、テーブルに新しい値を挿入したり、または以前に格納された既存の値を更新したりできます。以下が操作についての詳細なリストです。

1. 変数の値を格納するとき、ObjectKey は、変数のタイプが「データベース キーの一部」にマークされている属性の変数値に基づいて計算されます。ロボット開発者がアクションでキーを指定した場合は、そのキーが代わりに使われます。
2. 計算キーを使って、その値がすでにデータベースに存在するかどうかをチェックします。
3. 値が存在しない場合は、データベースに新しい行が挿入されます (この ObjectKey の下)。
4. 値がすでに存在する場合は、更新され、すべての非キー属性がテーブル (この ObjectKey の下) に書き込まれます。

データベース データ登録または SQL 実行ステップで使用されるデータベースのテーブルを削除したり変更したりしないでください。

監査データ フィールド

値が挿入されるたびに、7 つの監査データ フィールドすべてが更新されます。更新時は、一部のフィールドのみが変更されます。次のテーブルに概要を示します。

フィールド	説明	変更されるタイミング
ObjectKey	この値の主キー。	挿入
RobotName	この値を格納したロボットの名前。	挿入と更新

フィールド	説明	変更されるタイミング
ExecutionId	この値を格納した口ボット実行の実行 ID。	挿入と更新
FirstExtracted	値が初めて格納された時間。	挿入
LastExtracted	値が最後に格納された時間。	挿入と更新
LastUpdated	値が最後に更新された日付。	更新
ExtractedInLastRun	値が最新の実行で抽出されたかどうか ('y' と 'n' を使用)。	挿入と更新

各口ボットの実行後 (口ボットがデータベースデータ登録を使用)、この口ボットによって以前に収集され、この実行中に保存されないすべての値は、ExtractedInLastRun が "n" に、LastUpdated が "now" に設定されます。これは、値が最新の実行中に Web サイト上で見つからなかったことを示します。

値が直前の実行で見つかり、変更された非キー フィールドがない場合、LastUpdated は更新されません。ただし、直前の実行で値が見つからず、その前の実行では見つかっていた場合は、非キー フィールドが変更されていない場合でも、LastUpdated は更新されます。これは、値がサイトから削除され、その後に再び表示されていることを意味します。

ハーベスト テーブル

Kofax RPA で作成されたテーブルは、口ボットがデータを収集しているため、しばしばハーベスト テーブルと呼ばれます。

口ボットが最後に実行されたときに Web サイトで利用可能だった情報を確認するには、次の SQL コマンドを使用できます。

SELECT * FROM table WHERE ExtractedInLastRun = 'y'

口ボットがテーブルにデータを格納するのと同時に、テーブルに対して同時にクエリを実行すると、結果は、前回の実行からのデータと、実行中の口ボットがこれまでに保存したデータが混合した状態になります。安定したデータセットに対してクエリを実行できるように、データをハーベスト テーブルから別のプロダクション テーブルにコピーすることをお勧めします。

データベースにデータを格納するのに口ボットを使用するソリューションは多数ありますが、ほとんどは次のテーブルに示す 3 つのシナリオのいずれかに該当します。

シナリオ	説明
Web サイトと照合するリポジトリ (小さなデータ セット)	Web サイト上のアイテムと 1 対 1 で照合するリポジトリという考え方です。 これを実現する最も簡単な方法は、口ボットの実行ごとに、すべての行を削除したプロダクション テーブルを作成し、`ExtractedInLastRun='y'` となっているレコードをハーベスト テーブルからこのテーブルにコピーします。これは小さなデータ セットでは適切に機能します。

シナリオ	説明

Web サイトと照合するリポジトリ (大きなデータ セット)	上記と同様ですが、ロボットの実行後にすべてのデータをコピーするには、データセットが大きすぎます。代わりに、生じた変更に基づき、ロボットの各実行後にプロダクション テーブルを更新することにします。
	ここでは LastUpdated フィールドが便利です。更新された値にはすべて、ロボットの開始時刻より大きい LastUpdated フィールド値があります。開始時刻は、データベースのログ テーブルから取得することも、ロボットでどこかに格納することもできます。
	削除された値を検出するには、次のコマンドを使います。
	`SELECT * FROM table WHERE LastUpdated > 'StartTime' AND ExtractedInLastRun = 'n'`
	新しい値を検出する場合
	`SELECT * FROM table WHERE LastUpdated > 'StartTime' AND ExtractedInLastRun = 'y' AND FirstExtracted > 'StartTime'`
	更新した値を検出する場合
	`SELECT * FROM table WHERE LastUpdated > 'StartTime' AND ExtractedInLastRun = 'y' AND FirstExtracted < 'StartTime'`
	次にプロダクション テーブルを更新します。

履歴データ	デフォルトの設定では、値が最初に抽出されたタイミングと最後に更新されたタイミングを確認できますが、値が見つかったロボットの実行を確認することはできません。 この場合、ロボットの実行後、ハーベスト テーブルからすべてのデータを別のテーブルにコピーする必要がありますが、新しいテーブルでは、ObjectKey を主キーにすることはできません。代わりに、RUN_ID という列を追加で作成し、ObjectKey とともに使用して複合の主キーを作成します。RUN_ID が必要ない場合は、自動インクリメントされた列を作成し、それをセカンダリ テーブルの主キーとして使います。各実行前にハーベスト テーブルから行を削除します。

すべての監査データ フィールドをプロダクション テーブルにコピーする必要はありません。プロダクション テーブルの更新には ObjectKey のみが必須です。

競合に関する注意点

同じタイプの値を同じデータベースに格納する複数のロボットがある場合、次のことに注意してください。

- 値が格納されるたびに、RobotName 列が更新されます。2 つのロボットが同じ値 (ObjectKey によって識別) を格納している場合、ロボットが実行された後に最後のものだけが表示されます。
- 2 つのロボットが同時に同じ値を登録すると、エラーが発生します。いずれのロボットも、値がテーブルにないことを発見し、値を挿入しようとしますが、そのうちの 1 つだけが成功します。同じ値であるため、多くの場合、このエラーは無視できます。
- 同じロボットを同時に 2 回実行し、ロボットがデータベースにデータを格納すると、ExtractedInLastRun 列が適切に機能しなくなります。1 番目のロボットの実行が終了すると、ロボットは格納されていないすべての値の ExtractedInLastRun を "n" に更新します。これには、これまでに 2 番目のロボットによって格納されたすべての値が含まれます。その後、2 番目のロボットが終了すると、1 番目のロボットによって格納されたすべての値の ExtractedInLastRun を "n" に設定し、最初の実行を完全に否定します。

値の関係

ストレージ システムには、値間の関係を自動管理する手段がありません。Person タイプと Address タイプの値があり、これらをリンクさせたい場合は、このリンクを維持する必要があります。

リンクを作成する一番簡単な方法は、Person 値の ObjectKey を、この個人にリンクする Address 値の外部キーにすることです。

ObjectKey がタイプから自動的に計算される場合、ObjectKey の計算アクションを使用してキーを生成し、各アドレス値に割り当ててから格納することができます。

格納された値の間に接続がある口ボットを構築するときは注意が必要です。Person 値を格納するときにエラーが発生した場合は、Address 値が格納されていないことを確認します。

ObjectKey に関する警告

MySQL または Oracle を使用する場合は、以下の重要な ObjectKeys のルールを確認してください。

- Oracle では、空の文字列は null として格納されます。
- MySQL のタイム スタンプにはミリ秒の精度はありません。

これらの 2 つの事例では、データがデータベースに格納されるときにデータが失われる可能性があります。ObjectKey は、指定の変数のデータに基づいて口ボット内で計算されます。後にデータベースから値をロードし、ObjectKey を再計算しようとすると、データベース キーの一部としてマークされた属性のいずれかでデータ損失が生じた場合に、ObjectKey が異なります。

変数の設定

新しいベーシック エンジン 口ボット を作成する場合は、通常、最初に変数を追加して、設定を行います。変数の初期値の変更など、口ボットの存続期間中はいつでも変数を再設定できます。変数のドロップダウン リストには、選択したステップと条件に一致する変数のみが含まれます。

口ボット では、ベーシック エンジン 口ボット からの入力を使用し、設定を追加した同一の変数タイプを使用できます。

変数を追加して設定した後に検証エラーが発生した場合は、「変数の検証エラー」を参照してください。

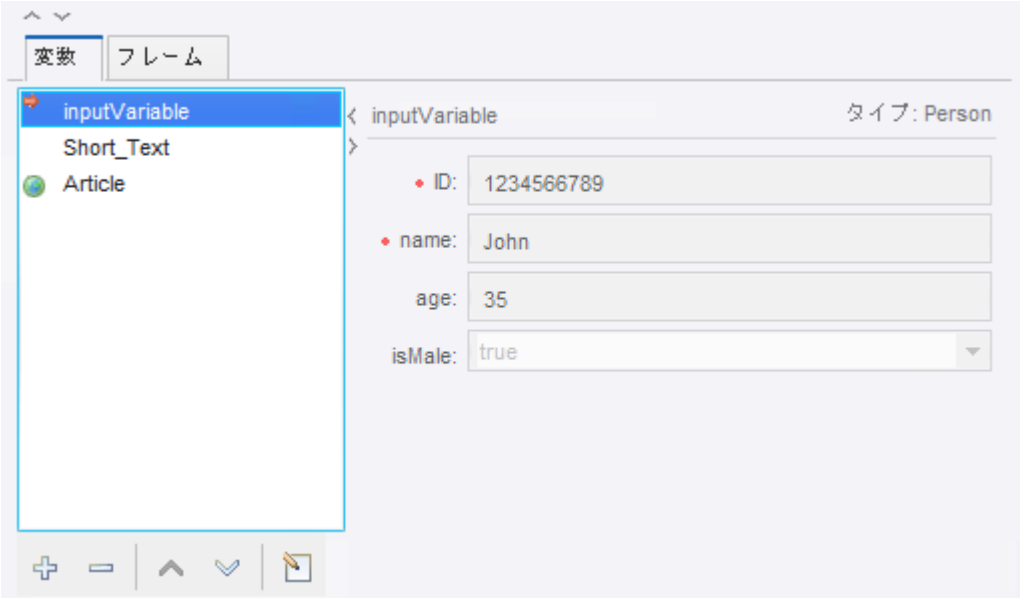
1. ベーシック エンジン 口ボット エディターの右下隅にある [変数] を選択します。

   指定した変数は、口ボットの最初のステップへの入力として提供される、口ボット状態の一部になります。

   [変数] タブには、変数のリストとともに、選択した変数の詳細が表示されます。変数の横にあるアイコンは、次の変数タイプを示しています。

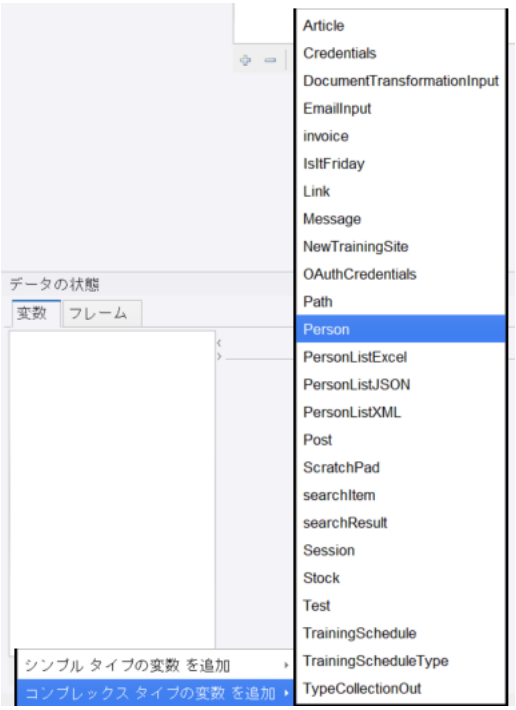
   - 入力変数
   - グローバル変数

   次の [変数] タブには、1 つの入力変数、1 つの通常の変数、1 つのグローバル変数の 3 つの変数が含まれています。



この [変数] タブには、現在のステップの変数の値が表示されます。

2. 新しい変数を追加するには、[追加] をクリックするか、変数を右クリックしてタイプを選択します。



[変数を編集] ダイアログ ボックスが表示されます。

> タイプを選択した状態で右クリックする方法で変数を追加した場合、あらかじめ選択されたタイプが含まれたウィンドウが開きます。

また、このダイアログ ボックスで変数をダブルクリックするか、  ボタンをクリックして、既存の変数を設定します。

3. [変数を編集] ダイアログ ボックスで、変数の名前を入力します。
   この名前は命名標準に準拠している必要があります。たとえば、スペースを使用することはできません。変数名が無効である場合、[**OK**] をクリックするとメッセージが表示されます。無効な名前を変更するか、[キャンセル] をクリックします。

> 変数の設定ウィンドウを使用して、初期値を編集します。言い換えると、変数ビューの場合と同様に、ダイアログ ボックスには現在の値が表示されません。指定した値は、実行の開始時に使用されます。

4. 変数タイプを選択します。
5. タイプに基づいて、入力フィールドに入力します。
   これらのフィールドを使用して、変数と初期値を指定します。変数に名前を手動で設定する必要はありません。
6. [**OK**] をクリックします。
   名前を入力しなかった場合は、タイプ名から名前を生成するように促されます。
7. [グローバル] および [パラメータとして使用] チェック ボックスを使用して、ロボットへの入力またはグローバルとして変数を設定します。
   変数が入力として使用される場合、RoboServer でロボットを実行するときに、その変数の値をロボットに提供することができます。入力変数の場合、属性に対して入力された値については、テスト入力値だとみなし、Design Studio でロボットを操作しているときにのみ使用する必要があります。RoboServer でロボットが実行されているとき、入力値は、ロボットを実行するクライアントによって提供された値で上書き (置換) されます。簡単なタイプの変数は一時変数としてロボット内部で使用されるため、これらの変数は入力として使用できないことに注意してください。
8. ロボットの実行全体で変数に値を保持する場合は、[グローバル] を選択します。
   グローバル変数を使用すると、カウンターを作成して、イテレーションおよび分岐全体で他の種類の計算を実行できます。また、グローバル変数は、コンマ区切りの値で構成されるテキストの集積など、イテレーションや分岐全体でデータを集積するために使用できます。
   この変数は、ループ イテレーションおよび分岐全体で値が保持されない通常の変数とは異なります。

> Design Studio では、グローバル変数の値は、現在のステップに到達するためにどのようなステップを実行したかによって決まります。ステップを正しい順序で実行するように留意しないと、値は、ロボットが実際に実行される場合の値と異なってしまいます。

9. 変数を除去するには、変数を右クリックし、[除去] を選択します。
   または、変数を選択し、リストの下にある [削除] をクリックします。

変数の検証エラー

検証エラーは、無効な変数と入力に対して発生します。このトピックでは、これらのエラーの対処方法について説明します。

エラーまたは結果	推定される原因	解決方法
「タイプが見つかりませんでした」	変数で使用されているタイプが見つかりません。	見つからないタイプの名前を使用してタイプを作成するか、[設定変数] ウィンドウを開いて変数に別のタイプを選択します。
「無効なタイプです」	変数で使用されるタイプに関する設定の問題。	タイプに関する問題を解決するか、[設定変数] ウィンドウから変数に別のタイプを選択します。
「タイプと互換性がありません」	変数には属性に割り当てられた初期値があり、その後、変数のタイプが変更されました。 1 つ以上の値を属性に割り当てることができなくなりました。これは、タイプから属性が削除されたり、属性タイプが変更されたりすることで、以前の割り当て値の互換性が現在は失われている場合に発生します。	割り当てられていない値を消去します。 1. [設定変数] ウィンドウを開き、[OK] をクリックしてウィンドウを閉じます。 割り当てられていない値が破棄されたことを示すメッセージが表示されます。 2. [OK] をクリックして値を破棄し、検証エラーを解決します。 たとえば、初期値が属性タイプ Short Text の属性に割り当てられている場合でも、変数タイプが変更され、属性タイプがブール値になると、古い値を割り当てることができなくなります。
「整数の最大値を超えるサイズは使用できません: 8639044608。」	[変数を追加] ダイアログ ボックスで、バイナリ タイプの入力変数またはパラメータを追加し、最大サイズを超えるファイルをロードすると、Design Studio が閉じます。	対象のファイルを、最大サイズの 8639044608 バイトよりも小さいファイルに置き換えます。

タイプを設定

タイプ エディターのメイン ビューでは、タイプのさまざまなプロパティを編集できます。これには、タイプの属性、タイプに付加されたコメント (オプション) 、およびストレージ名 (オプション) が含まれます。

タイプには、有効な名前が必要です。この名前は、対応するタイプのファイル名で構いません。この名前には、文字、数字、アンダースコアのみを含める必要があります。名前は、文字またはアンダースコアで始まる必要があります。さらに、この名前は、名前の大文字と小文字を含め、プロジェクト内で一意である必要があります。たとえば、1 つのプロジェクトに「MyType」と「mytype」という 2 種類の名前を含めることはできません。タイプの名前は、ストレージ名が明確に設定されている場合を除き、ストレージ名として使用されます (以下を参照)。

タイプには、タイプ エディターで設定できる以下のプロパティが含まれます。

属性

タイプに追加される属性がテーブルに表示されます。新しい属性を追加するには、テーブルの下の [新しい属性の追加と設定] をクリックします。属性を除去するには、属性が入力されている行を選択して、[属性除去] ボタンをクリックします。属性を設定するには、[属性を設定] ボタンをクリックします。属性の追加または設定を行う際には、[属性の設定] ダイアログが自動的にポップ アップします。スペース節約

のため、テーブルのすべての列がデフォルトで表示されるわけではありません。表示する列を変更するには、任意の列の名前を右クリックします。

コメント

ここでは、オプションのコメントをタイプに追加できます。コメントは、タイプ エディターでのみ表示されます。

ストレージ名

ここでは、このタイプの変数から値 (例：データベースのテーブル名、または XML のタグ名) を格納する際に使用する名前を設定できます。このフィールドを空欄にすると、タイプの名前が、ストレージの代わりに使用されます。タイプの使用目的によっては、ストレージ名に制約が加えられることがあります。たとえば、タイプの値がデータベース データ登録されることになっている場合、使用する予定のデータベースのいくつかのキーワードと同じストレージ名を使用することは避ける必要があります。

属性の設定

[基本] タブ

このタブには、属性の基本プロパティが含まれます。

名前

属性の名前。この名前は、大文字と小文字を含めて、タイプ内で一意である必要があります。たとえば、「Location」と「location」という 2 種類の名前を持つ属性を含めることはできません。なお、名前には文字、数字、アンダースコアのみを含めることができ、先頭は文字またはアンダースコアとする必要があります。さらに、属性でストレージ名が設定されていない場合、ストレージ名としてこの名前が使用されます (データベースの行の名前、CSV ファイルの列のヘッダー、または XML のタグ名)。タイプの使用目的によっては、名前にその他の制約が生じる場合があります。例として、使用を意図するデータベースのキーワードと同じ属性の名前の使用は避ける必要があります。

タイプとデフォルト値

属性タイプのリストから属性の属性タイプを選択し、属性のデフォルト値を設定します。

レコード タイプのフィールド タイプ

派生レコード タイプの対応するフィールドを選択します。たとえば、属性のタイプが日付の場合、フィールド タイプをDate、Time、または DateTime (デフォルト) のいずれかに指定できます。

必須

このオプションにチェックを入れると、以下の 2 つが有効になります。

- 属性に値がない場合 (例外がスローされた場合)、タイプの変数は (ファイルまたはデータベースに) 保存されない。必須でない属性に値が指定されていない場合、例外やエラーはスローされませんが、ファイルには何も保存されません。
- 該当するタイプの入力変数に属性の値が必須となる、またはロボットの実行が開始されない。

データベース キーの一部

データベースにタイプの値を保存する場合は、このタイプの値をキーの下に保存する必要があります。値のキーは、データベース キーの一部である属性の安全なハッシュとして計算されます。データベースに値を保存する際には、適切なキーを選択することが重要です。選択したキー属性が、タイプのすべての値において一意であるようにする必要があります。理想的なキーの例として、製品番号および URL が挙げられます。データベース データ登録されたデータがある場合、このオプションを変更する際には細心の注意を払う必要があります。変更を行うと、ロボットがデータベースの既存の値を再検索 (更新) で

きなくなることがあります。すべての口ボットに適切な設定が行われていても、1 つの口ボットに別のキー計算が必要となる場合は、口ボットのステップでキー フィールドを変更する必要があります。

コメント
このフィールドには、属性の詳細を説明するためのオプションのコメントを追加できます。

[詳細] タブ

このタブには、属性の詳細プロパティが含まれます。

ストレージ名
これは、データベースの行名、CSV ファイルの列ヘッダー、XML のタグ名など、属性を格納するときに使用するオプションの別の名前です。このフィールドを空欄にすると、[名前] プロパティの値がストレージに自動的に使用されます。タイプの使用目的によっては、ストレージ名にその他の制約が生じる場合があります。例として、タイプの値をデータベース データ登録しようとする場合、使用を意図するデータベースのキーワードと同じストレージ名の使用は避ける必要があります。

表示
このオプションは、属性が Design Studio の口ボットで表示されるようにする場合に選択します。

格納可能
このオプションは、タイプの値を保存する際に、この属性を保存する必要がある場合に選択します。

先に区切り記号を表示
タイプを Design Studio の口ボットで使用する際に、区切り記号がこの属性の前に表示されるようにする場合、このオプションにチェックを入れます。

区切り記号のタイトル
区切り記号の名前。

タイプ属性

タイプを有効にするには、タイプ内の属性も正しく追加および設定する必要があります。各属性には名前とタイプをする必要があります。利用可能な属性タイプを次の表に示します。

属性タイプ	説明
バイナリ	バイナリ データ。任意のバイト配列。
ブール値	ブール値で、"true" または "false" 。
文字	"A" などの単一の文字。
国	ドイツの "DE" など、ISO-3166 規格で定義されている国コード。
通貨	ユー口の "EUR" など、ISO-4217 規格で定義されている通貨コード。
日付	"2021/04/25 10:33:06.0" のような yyyy-mm-dd hh:mm:ss.n という形式を使用する日付。また、Date、Time、DateTime を含むレコード タイプの変数を参照してください。

属性タイプ	説明
Excel	Excel ドキュメント。ドキュメントをプレビューして基本的な Excel 操作を実行できることを除き、これはバイナリ データと同じです。
HTML	HTML クリップ。ブラウザ ウィンドウでクリップをプレビューできることを除いてロング テキストと同じです。
イメージ	イメージ。イメージはプレビューできることを除いてバイナリ データと同じです。
整数	12 のような整数。可能な範囲は、-9223372036854775808 ~ 9223372036854775807 で、両端を含む。
JSON	JSON 値は JSON テキストまたは JSON シンプル タイプのいずれかになります。JSON シンプル タイプとは JSON リテラル、数字、または文字列のいずれかを指します。
言語	ドイツ語の "de" など、ISO-639 規格で定義されている言語コード。
ロング テキスト	ロング テキスト複数行のテキスト ボックスに表示されます。
数値	12.345 のような数値。可能な範囲は ±2.2×10-308 から ±1.8×10308 で、精度は 15 桁をわずかに上回る。
パスワード	パスワード。パスワードの文字をアスタリスクで表示するパスワード フィールドに表示されます。
PDF	PDF 文書。PDF 文書はプレビューできることを除いてバイナリ データと同じです。
プロパティ	それぞれが名前と値のペアであるプロパティのリストを表すテキスト。詳細については、プロパティ属性のタイプを参照してください。
セッション	Cookie、認証などを含むセッション。
ショート テキスト	ショート テキスト。1 行のテキスト フィールドに表示されます。
XML	XML ドキュメント。適格な XML 文書のみが許容されることを除いてロング テキストと同じです。

プロパティ属性のタイプ

プロパティ属性タイプの属性には、プロパティのリストを表すテキストが含まれます。リストの各プロパティは名前と値のペアです。

プロパティ属性のタイプは、動的に変化する可能性のあるプロパティ リストを表すのに便利です。プロパティのセットが固定されている場合、通常、各プロパティを代わりに属性として表わすことになります。

プロパティ属性タイプの属性が含む可能性のあるプロパティのリストの例は、次の通りです。

"productName" =「油圧バルブ」"productNumber" = "53563-433" "productVendor" = "American Valves Inc." "productWeight" = "3.45" ...

各プロパティは別の行に存在する必要があります。プロパティの行は、プロパティ名、その後に "="、そしてプロパティの値の順に並べられます。特定のプロパティは、リストに最大で 1 回しか存在できません。プロパティ名を空にできませんが、値は空にできます。

名前と値は、引用符あり、または引用符なしで指定します。引用符を使用する場合、バックスラッシュ文字 (\) を使用して特殊文字を入力することができます。

- \n は改行。
- \r はキャリッジ リターン。
- \f は改ページを表します。
- \t は水平タブ。
- \b はバックスペース。
- \」は二重引用符。
- \' は一重引用符。
- \\ はバックスラッシュ。
- \uxxxx は xxxx エンコードの Unicode 文字。xxxx は 4 つの 16 進数の値。

名前や値に引用符を使用しない場合、名前と値の最初と最後のすべてのスペースは除去され、空の値を指定することはできません。特殊文字を入力するのに、バックスラッシュ表記を使用することはできません。

プロパティ リストには、空行やコメント行を含めることができます。コメント行は 2 本のスラッシュ (//) で開始します。

Design Studio の設定

Design Studio の設定を開くには、メニューの [設定] をクリックします。Design Studio の設定ウィンドウの次のタブを使用して、Design Studio のユーザー設定を行います。

- 一般設定
- テキスト ファイル
- ベーシック エンジン ロボット エディター
- ロボット エディター
- Desktop Automation
- ローカル データベース
- プロキシ サーバー
- 証明書
- Management Console

一般

Design Studio 設定ウィンドウを開くと、[一般] タブがデフォルトで表示されます。このタブを使用して、Design Studio の一般ユーザー設定を行います。

以下の表で、[一般] タブのオプションについて説明します。

オプション	説明
Design Studio への切り替え時	他のアプリケーションあらから Design Studio に切り替えたときに何が起こるかを示します。 [すべてのプロジェクトを更新] を選択した場合 (デフォルト)、Design Studio は、開いているプロジェクトのファイルについて、Design Studio に前回フォーカスしてから変更が加えられたかどうかを確認します。競合する変更が検出された場合、Design Studio はその解決を支援し、[マイ プロジェクト] ビュー全体を更新します。 [ファイル エディターのみを更新] を選択した場合、Design Studio は、開いているプロジェクトのファイルについて、Design Studio に前回フォーカスしてから変更が加えられたかどうかを確認します。競合する変更が検出された場合、Design Studio はその解決を支援しますが、[マイ プロジェクト] ビューは更新しません。ユーザーが現在のエディターで開いているファイルの競合を解決した場合、(このファイルの) ビューが部分的に更新されることはありますが、プロジェクト全体のビューは更新されません。 [更新しない] が選択されている場合、競合する変更の確認や更新は行われません。
デフォルト実行モード	新しく作成されたすべてのロボットについて、デフォルトの**ロボット実行モード**を指定します。
最近開いたプロジェクトの最大数	最近開いた Design Studio プロジェクトのローカル履歴に含める最大数をリストします。リストにアクセスするには、**[ファイル]** > **[最近開いたプロジェクト]** を選択します。
最近開いたファイルの最大数	最近開いた Design Studio ファイルのローカル履歴に含める最大数をリストします。リストにアクセスするには、**[ファイル]** > **[最近開いたファイル]** を選択します。
起動時にプロジェクトを開く	選択すると、最も最近開かれたプロジェクトが Design Studio の起動時に再度開かれます。
起動時にウェルカム スクリーンを表示	選択すると、ウェルカム スクリーンが Design Studio の起動時に表示されます。
バックアップ ファイルを生成	選択すると、保存されているファイルが変更されるたびにバックアップ ファイルが作成されます。バックアップ ファイル名の最後には波形記号 (~) が付きます。
ドキュメントの場所	選択すると、Kofax RPA ドキュメント セットがオフラインモードで使用できます。
ドキュメントの取得通知を表示	選択すると、インターネット アクセスなしで Kofax RPA からオンライン ドキュメントにアクセスしようとすると、「ヘルプとドキュメントの取得」という警告が表示されます。
URL 経由で開いたリモート ファイルを同期	Management Console に保存されているファイルと、**URL を使用して Design Studio で開いたファイル**を同期するかどうかを指定します。[なし] に設定すると、Design Studio で開いたファイルのバージョンが、Management Console に保存されているバージョンと一致しないことがあります。 デフォルトでは、[毎回確認する] に設定されています。

テキスト ファイル

[テキスト ファイル] タブを使用して、Design Studio で使用されるテキスト ファイルの設定を行います。

次の表に、[テキスト ファイル] タブのオプションを示します。

オプション	説明
デフォルトのファイル エンコード	テキスト ファイルのデフォルトのエンコードを指定する。
デフォルトの行区切り記号	テキスト ファイルのデフォルトの行区切り記号を指定する。
デフォルトのタブ サイズ	テキスト ファイルのデフォルトのタブ サイズを指定する。

ベーシック エンジン ロボット エディター

このタブを使用して、ベーシック エンジン ロボット エディターのユーザー設定を行います。

次のオプションが利用可能です。

オプション	説明
ステップのツールチップを表示	このチェック ボックスを選択すると、ベーシック エンジン ロボット ステップのツールチップが表示されます。
ステップでエラー処理を表示	選択すると、カスタム エラー処理を含むベーシック エンジン ロボット ステップに赤い感嘆符マークが表示されます。
デフォルトのズーム比	ベーシック エンジン ロボットを ロボット エディターで開いたときに適用されるズーム比を割り当てます。ズーム比は ロボット エディターの右下隅で手動で調整できます。
ソース ビューのフォント	ソース ビュー (Design Studio の一番下のセクション) のテキスト フォントのポイント サイズを指定します。
よく使用するステップ	Design Studio のベーシック エンジン ロボット ビューで接続を右クリックしたときに使用可能な [ステップを挿入] メニューに直接表示されるステップをリストします。リストにステップを追加するには、 をクリックし、ステップを選択します。リストからステップを削除するには、1 つ以上のステップを選択して をクリックします。ステップを並べ替えるには、ステップを選択し、矢印を使用してリスト内で上下に移動します。

ロボット エディター

このタブを使用して、ロボット エディターのユーザー設定を行います。

次のオプションが利用可能です。

オプション	説明
表示される出力値の最大数	状態ビューに含める各タイプの出力値の最大数が一覧で表示されます。 Design Studio で ロボットが [出力値] ステップを実行すると、すべての値が収集されますが、状態ビューには最新の値のみが表示されます。必要な数の出力値を表示するには、このオプションを設定します。

Desktop Automation

このタブを使用して、リモート コンピュータのユーザー設定を行います。

次のオプションが利用可能です。

オプション	説明
コマンド タイムアウト (秒)	オートメーション デバイスのコマンドからの応答を Design Studio で待機する必要がある時間を指定します。このオプションは、ロボットでのターミナルの自動化および Web サイトのブラウジングにのみ適用されます。 コマンドとは、マウス ボタンをクリックする、アプリケーションを開く、「該当するロケーション」ガードを追加するといったオートメーション デバイスに送信される命令です。コマンドが指定した時間内に完了できない場合、サービスによって通知が送信され、ロボットの実行が停止します。 ガード チョイス ステップの場合、この設定はワークフローでのガードの呼び出しに適用されますが、ガードが満たされるまでの待機はこのタイムアウトとは無関係のため、無制限に待機し続ける可能性があります。マウス移動 ステップと抽出ステップの使用時に、同様の状況が発生します。コマンドはフィールドで指定されたタイムアウト以内にデバイスで呼び出される必要がありますが、ロボットはコマンドの完了を最大 240 秒間待機します。
ローカル ハブ TLS 構成設定 詳細については、TLS コミュニケーションを使用を参照してください。	
デフォルトを使用	Design Studio とオートメーション デバイス間の TLS 接続に対し Kofax RPA で提供されるファイルを使用します。
秘密鍵ファイル	Design Studio コンピュータ上に存在するローカル ハブで使用される秘密鍵ファイルへのパス。
公開鍵ファイル	基盤となる秘密鍵で署名される公開キー ファイルへのパス。
信頼済み証明書フォルダ	信頼されている証明書を保存するフォルダ。

ローカル データベース

[ローカル データベース] タブを使用して、Design Studio でデータベースを作成します。Design Studio で作成されたデータベースは、Design Studio でのみ使用できます。

データベースを Design Studio とサーバーで使用できるようにするには、データベースを Management Console で設定する必要があります。

作成された接続のリストが左ペインに表示されます。リスト下のボタンを使用して、新しい接続の作成、接続の除去、接続順序の変更を行うことができます。現在選択されている接続は、[ローカル デー

タベース] ウィンドウの右側で設定されています。フィールド名、ホスト、タイプ、スキーマ、ユーザー名、およびパスワードは必須で、指定する必要があります。

さまざまなデータベース タイプが Management Console で定義され、起動時に Design Studio に自動的に配信されます。新しいデータベース タイプは、Management Console で作成する必要があります。

オプション	必須	説明
名前	はい	Kofax RPA のデータベース名を一意に識別する名前。名前はデータベースの内部参照に使用され、英数字とアンダースコアのみを含めることができます。
ホスト	はい	データベース サーバーのホスト名。IP アドレス、または完全修飾ドメイン名になります (例：myhost.kofax.com)。
タイプ	はい	データベースのタイプ (例：Oracle)。Management Console ではさまざまなタイプのデータベースを設定し、Design Studio の起動時にそれらが自動的に提供されます。
スキーマ	はい	データベース スキーマ (またはカタログ) の名前。
ユーザー名	はい	データベースのユーザー名。
パスワード	はい	データベースのパスワード。
最大アクティブ接続数	はい	Kofax RPA (RoboServer または Design Studio) で作成されるデータベースへの同時接続の最大数。接続は接続プールで管理されます。つまり、新しい接続を作成する前に既存の接続が再利用されます。
最大待機接続数	はい	許可される待機接続の最大数。負荷が高いときに多くの接続が作成された場合、それらは不要になったときに自動的に閉じられます。

現在の接続をテストするには、[テスト接続] をクリックします。

このアクションは、データベースとの接続のみをテストします。データベースに適切な権限があるかどうかの判別は、このテストでは行われません。

Oracle への接続：Oracle データベースを使用している場合、[ユーザ名] フィールドにユーザ名とロールを入力する必要があります。たとえば、ユーザ名が "sys"、ロールが "sysdba" の場合、[ユーザ名] フィールドに "sys as sysdba" と入力する必要があります。

プロキシ サーバー

[プロキシ サーバー] タブを使用して、Design Studio で使用できる数のプロキシ サーバーを指定します。

オプション	説明
プロキシ サーバーを使用	選択すると、プロキシ サーバーの使用が有効になります。
ホスト	プロキシ サーバーのホスト名。IP アドレス、または完全修飾ドメイン名になります (例：myproxy.kofax.com)。
ポート	プロキシ サーバーのポート番号。デフォルトのプロキシ サーバー ポート 8080 を使用する場合は空白のままにします。
ユーザー名	プロキシ サーバーでログインが必要な場合に使用するユーザー名。
パスワード	プロキシ サーバーでパスワードが必要な場合に使用するパスワード。

オプション	説明
除外ホスト	ここで、プロキシ サーバーが使用されないホスト名のリストを指定できます。1 つの行に 1 つのホスト名を指定するか、ワイルドカード (*) を使用して 1 つの行に 1 つのホスト名パターンを指定できます。各ホスト名は IP アドレス、または完全修飾ドメイン名になります (例 : www.kofax.com)。

プロキシ サーバーの下の [インポート] ボタンを使用して、プロキシ サーバーのリストをインポートします。ファイルには任意の数のプロキシ サーバー定義を保持でき、それぞれ次の形式に準拠する必要があります。

```
proxyName.proxyServerUse = true
proxyName.proxyServerHost = host name or IP address
proxyName.proxyServerPort = port number
proxyName.proxyServerUserName = user name
proxyName.proxyServerPassword = password
proxyName.proxyServerExcludedHostNames = list of hosts
```

proxyName は特定のプロキシ サーバーを識別するための名前です。各プロキシ サーバーには独自の一意の proxyName がある必要があります。

複数のプロキシ サーバーが指定されると、ロボットの実行ごとに新しいプロキシ サーバーが選択されます。

個別のロボットに対しプロキシ サーバーを指定することもできます。これは、Design Studio のロボットの設定ウィンドウでロボットを設定するときに行われます。そのようなプロキシ サーバーによってここで指定されているプロキシ サーバーが上書きされます。詳細については、ロボット設定を参照してください。さらに、プロキシ サーバーはプロキシ切替アクションによるロボットの実行中にプロキシ切替されます。

詳細については、プロキシ サービスの使用を参照してください。

証明書

[証明書] タブで、ロボットが使用できるクライアント証明書のリストを指定できます。

プロパティ

ID
証明書の一意の識別文字列。

発行先
証明書の発行先である X-509 サブジェクト名。

発行元
証明書発行者の X-509 サブジェクト名。自己署名証明書では、[発行先] フィールドと [発行元] フィールドに同じ名前が含まれています。

期限切れ
証明書の有効期限。

エイリアス
証明書ストレージ内の証明書の名前。このフィールドが空の場合、証明書は自動的に選択されます。

デフォルト
チェック マークはデフォルトの証明書を示します。デフォルトの証明書は 1 つだけ選択できます。

証明書テーブルの下にあるボタンをクリックすると、証明書を追加、削除、および設定できます。

Management Console

[Management Consoles] タブを使用して、Management Consoles への接続を設定します。URL は一意とする必要がありますが、異なるプロトコル、ユーザー名、およびパスワードを使用して、同じ Management Console への複数の接続を構成できます。初めて Design Studio を開始してライセンス サーバーを指定すると、そのサーバーが Management Consoles のリストに自動的に追加されます。

名前

Management Console の名前。

URL

Management Console に接続するための URL。`HTTP` または `HTTPS`、およびポート番号を入力してプロトコルを指定します。例：`http://localhost:50080/`このフィールドには IP アドレスを使用することもできます。

プロキシ サーバーを使用

Management Console が外部サーバーに接続するときに経由するプロキシ サーバーを使用する場合に選択します。

- [ホスト名]: プロキシ サーバーのホスト名
- [ポート]: プロキシ サーバーのポート
- [ユーザー名]: 認証が必要な場合にプロキシ サーバーに接続するためのユーザー名
- [プロキシ パスワード]: 認証が必要な場合にプロキシ サーバーに接続するためのパスワード

詳細については、プロキシ サーバーセクションを参照してください。

DB 警告を表示

これを選択すると、欠落したテーブルなどのデータベースの警告が、ロボット エディターの上部に表示されます。

JDBC ドライバーを承認

選択すると、JDBC ドライバーが Management Console から Design Studio に分配されます。ユーザーがこのオプションを無効にする必要はほぼありません。

プライマリとして使用

現在の Management Console をプライマリ Management Console として使用する場合に選択します。このオプションは、パスワード ストア、ロボット ファイル システム、および Kofax TotalAgility 設定にアクセスするときに接続する Management Console に影響します。ローカルにのみ存在し、Management Console と同期していないプロジェクト内のロボットはプライマリとマークされている Management Console を使用します。ロボットがシェア プロジェクト内にある場合、プロジェクトの同期先の Management Console に接続されます。

ライセンス サーバーとして使用

現在の Management Console をライセンス サーバーとして使用する場合に選択します。このオプションは 1 つの Management Console にのみ適用でき、Design Studio への認証にどの Management Console が使用されるかに影響します。

Management Console へのアップロード

ロボットおよびロボットが使用するタイプとスニペットを Management Console にパブリッシュするには、プロジェクト ビューでロボットを右クリックし、[アップロード] をクリックします。このダイアログ ボックスは、ローカルの非共有プロジェクトからファイルを Management Console のシェア プロジェクトにアップロードする際に表示されます。詳細については、シェア プロジェクトの使用 を参照してください。

Management Console

リストからいずれかの Management Console を選択します。このリストには、Design Studio を接続した Management Console が含まれています。

プロジェクト

アップロードするプロジェクトを選択します。

記憶する (シェア プロジェクトとして)

このオプションを選択して、選択した Management Console プロジェクトにプロジェクトをリンクします。

URL でファイルを開く

特別な URL を使用することにより、Design Studio でファイルをすばやく開くことや、ロボットまたはスニペット内の特定の場所に移動することができます。この URL は、`kofaxrpa:` プリフィックスによって示されます。これには開くファイルへのパスと、オプションでステップのロケーション コードが含まれています。

この機能は、さまざまなケースで役に立ちます。たとえば、Management Console で実行されているロボットのエラーが報告された場合に、Management Console のログ ビューから直接 Design Studio でロボットを簡単に開き、エラーのあるステップをすばやく見つけることができます。

前提条件:

- この機能がサポートされているファイルは、ロボット、ベーシック エンジン ロボット、スニペット、タイプ、データベース マッピング、オートメーション デバイス マッピング、およびリソースです。
- ファイルがリモート プロジェクトの一部である場合、URL は Management Console に保存されているファイルの URL です。リンクのコピー元の Design Studio と同じ Management Console に接続されている任意の Design Studio で開くことができます。
- ファイルがローカル プロジェクトのみに保存されている場合、URL はそのローカル プロジェクトを認識している Design Studio で開くことができますが、ファイル バージョンの同期は行われません。

このトピックは、次のサブトピックに分かれています。

Design Studio またはブラウザからファイルを開く

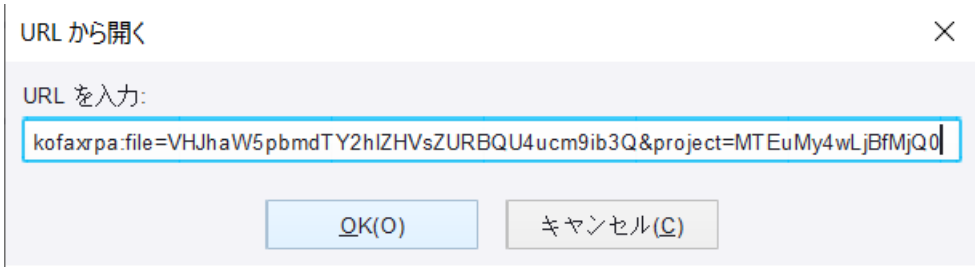
URL を取得する

Design Studio 内からファイルやステップへの URL を取得するには、以下の 2 つの方法があります。

- [マイ プロジェクト] ツリーで、ファイルを右クリックして [**URL** をコピー] をクリックすると、URL がクリップボードにコピーされます。
- ロボット、ベーシック エンジン ロボット、またはスニペットの編集時は、ステップを右クリックして [**URL** をコピー] をクリックすると、選択したステップへの URL がクリップボードにコピーされます。

URL を開く

- Design Studio から直接 (すべてのオペレーティング システムでサポート)
  1. URL をコピーします。
  2. Design Studio のメニューで、**[ファイル]** > **[URL から開く]** の順にクリックします。
  3. 表示されるダイアログ ボックスで、URL を貼り付けて [**OK**] をクリックします。

     Design Studio で、URL によって指定されているファイルが開きます。URL にロケーション コードが含まれている場合は、指定されている場所に移動します。
- ブラウザから (Windows でのみサポート)
  1. URL をクリックするか、コピーして Web ブラウザに貼り付けます。
     ブラウザで、Design Studio を開くよう求められます。
  2. [確認] をクリックして続行します。
     Design Studio が URL で指定されているファイルとともに開きます。URL にロケーション コードが含まれている場合は、指定されている場所に移動します。

Management Console からファイルを開く

Management Console 内から Design Studio 内のファイルを開くには、いくつかの方法があります。

- [リポジトリ] > [ロボット]/[タイプ]/[スニペット] で、ファイルのコンテキスト メニューから [**Design Studio** で開く] をクリックします。
- [ログ ビュー] > [ロボット メッセージ] で、エラー メッセージ エントリのコンテキスト メニューから [**Design Studio** で開く] をクリックします。

  ロボットのステップに関してエラーが報告されている場合は、そのロボットを使用するベーシック エンジン ロボットも開きます。
- [リポジトリ] > [ロボット] で、ロボットのコンテキスト メニューから [ドキュメントを生成] をクリックします。

  生成されたドキュメントから、ヘッダーのロボット名の横にある をクリックしてロボットを開くか、ロボット概要のステップ名の横にある をクリックして特定のステップを開くことができます。

その後、ブラウザで Design Studio を開くよう求められます。Design Studio が URL で指定されているファイルとともに開きます。URL にロケーション コードが含まれている場合は、指定されている場所に移動します。

ファイル バージョンの同期

ファイルが Management Console に保存されている場合は、最初にファイルを同期するよう求められます。同期を拒否しても、ファイルを Design Studio で開くことはできますが、Management Console に保存されているバージョンと一致しない可能性があります。この動作は、**[Design Studio 設定]** > **[一般]** > **[URL 経由で開いたリモート ファイルを同期]** で調整できます。

ファイル編集

どのようなファイルでも URL を使用して開くことや、特定の場所に移動することができますが、編集に関して以下のような違いがあります。

- Design Studio で直接編集できるファイルは、新しいタブで開いて、そのまま編集できます。
- Design Studio で直接編集できないファイルは、[マイ プロジェクト] ツリーで選択します。
- デバイス マッピングやデータベース マッピングなど、ダイアログ ボックスで編集 (設定) するファイルの場合は、編集を許可するよう求めるメッセージが表示されます。

第 4 章

Desktop Automation サービス

Design Studio と Management Console を使用した Desktop Automation サービスを設定し、リモートコンピューター上でロボットを自動化して実行できます。リモート Desktop Automation は、Windows オペレーティング システムでのみサポートされます。他のシステム要件が適用されます。

- Desktop Automation サービスがインストールされていない場合の要件と手順については、『Kofax RPA インストール ガイド』を参照してください。
- Desktop Automation サービスのインストール後の設定と管理については、『Kofax RPA Desktop Automation サービス ガイド』を参照してください。
- オートメーション デバイスの準備。
- オートメーション デバイス マッピングの要件 を使用して、オートメーション デバイス マッピングを追加または編集します。
  - オートメーション デバイス マッピングの追加します。
  - オートメーション デバイス マッピングの編集 。
- RoboServer デバイス マッピング を作成します。
- [Design Studio 設定] > [Desktop Automation] タブでユーザー設定を行います。

オートメーション デバイスの準備

ロボットが実行できるオートメーション デバイスは次のとおりです。ローカルの Windows コンピュータ、ロボットが実行されているコンピュータ、またはロボットが到達できる IP アドレスにあるリモートの Windows コンピュータ。リモート コンピュータは、ロボットがそのコンピュータを制御できるようにするための Desktop Automation サービスを実行している必要があります。

ターミナル セッションを自動化する方法については、ターミナル エミュレータの自動化 を参照してください。

1. プロジェクトのワークフローでリモート コンピュータに接続するには、最初にデバイス マッピングを作成します。この操作を行うには、**[ファイル]** > **[新しいオートメーション デバイス マッピング]** をクリックします。新しいマッピングを作成してから、ロボットの編集に進みます 。「オートメーション デバイス マッピングの追加」を参照してください。
2. ベーシック エンジン ロボット で「ロボットを呼び出す」ステップの [アクション] タブの [デバイス] フィールドにある [追加] をクリックして、デバイス リファレンスを入力します。
3. [デバイスの追加] ウィンドウで、[静的リファレンス]、[動的リファレンス]、または [トリガー リファレンス] を選択します。
   - 静的リファレンスは、選択する 1 つ以上のオートメーション デバイス マッピングを作成したことを意味します。ロボットにより、選択したマッピングと関連付けられているデバイスが自動化されます。マッピング情報は Windows デバイスの自動化に必要ですが、組み込みブラウザおよ

びロボット用の組み込み Excel ドライバーで操作するターミナルの自動化には必要ありません。マッピングを変更する場合、Design Studio の [更新] をクリックして、接続を更新します。ローカル Desktop Automationを使用する場合、静的リファレンスを使用します。

- 動的リファレンスは、リモート デスクトップ プロトコル (RDP) 接続などを使用して、オートメーション デバイスにシングル ユーザー モードで接続できるようにするリファレンスのタイプです。デバイスに接続ステップで使用するマッピング名を指定します。その他のすべての接続パラメータは、ロボット ワークフロー内で指定します。「ロボットの呼び出し」ステップでロボットが動的リファレンス接続を使用した後に、デバイスを接続すると、接続は維持され、このリファレンス (およびこのデバイス) はロボットの次の「ロボットを呼び出す」ステップで使用できるようになります。

  ワークフローの実行中は、同じデバイスに一度しか接続できません。たとえば、ループ内のデバイスに接続する場合、接続がすでに確立されているときには、ロボットがループ内の接続ステップをスキップすることを確認します。

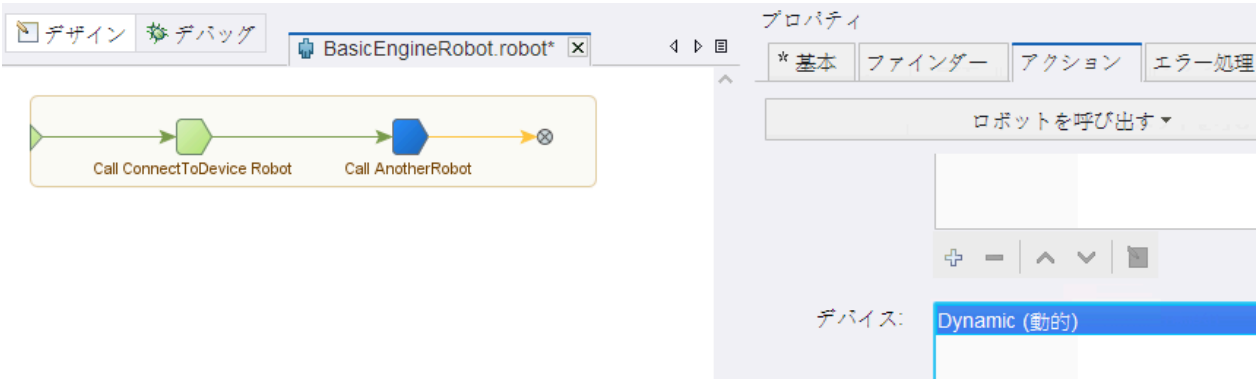
- トリガー リファレンスで参照は、選択する 1 つ以上のオートメーション デバイス マッピングを作成したことを意味します。自動化されたアテンデッド オートメーション ロボットは選択したマッピングに関連付けられたデバイスに接続します。デバイス マッピングを作成する場合、デバイスのホスト、ポート、およびトークンを指定します。「オートメーション デバイス マッピングの追加」を参照してください。

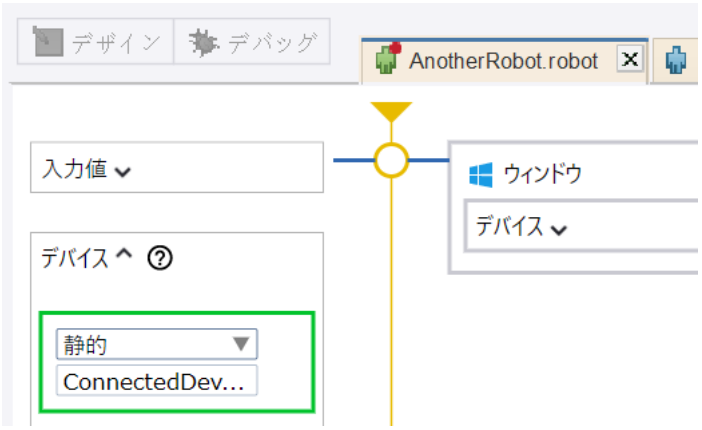
  組み込みのブラウザで Web サイトにアクセスしたり、組み込み Excel ドライバーで Excel スプレッドシートを操作したりするなど、ローカル コンピュータで実行される操作に対する参照は指定しないでください。`"local"` の参照は、他の参照が指定されているかどうかに関係なく、デフォルトで常に利用可能です。

4. 前のステップでオートメーション デバイスを追加した場合は、ロボットを編集します 。ロボットの編集 および デバイス の定義を参照してください。

ロボット間で動的デバイスを渡す

ロボット内の静的デバイスは、接続がすでに確立されているデバイスです。ロボットを呼び出す ステップにベーシック エンジン ロボットの [デバイス] プロパティで動的マッピングが指定されており、デバイスがすでに接続されている場合は、この動的マッピングが静的デバイスに渡されます。つまり、デバイスがすでに動的マッピングを使用して接続されている場合は、ロボットの [デバイス] ペインで静的デバイスの使用を選択することで、動的マッピングを使用して確立された接続を再利用できます。これは、このロボットを別のベーシック エンジン ロボットで再利用する必要がある場合に便利です。





オートメーション デバイス マッピングの要件

Design Studio のプロジェクトからデバイスにアクセスするためのデバイス マッピングを設定します。

- [デバイス マッピング] オプションは、Design Studio が Desktop Automation インスタンスに直接接続する場所です。
- [Management Console ベースのデバイス マッピング] オプションは、Management Console を使用して、プールから利用可能な Desktop Automation インスタンスを、その後 Design Studio が接続する Design Studio に割り当てます。

ターミナル エミュレータを自動化する場合は、リモート コンピューターでの Desktop Automation サービスの使用や Management Console でのデバイス マッピングの作成を行わないようにしてください。「ターミナル エミュレータの自動化」を参照してください。

[Management Console ベースのデバイス マッピング] オプションを使用すると、ネットワーク インフラストラクチャが変更された場合でも、作成したロボットが、指定したオートメーション デバイスを使

用して専用の Management Console で動作するようになるため、このオプションを使用することをお勧めします。

- オートメーション デバイス マッピングの名前は文字またはアンダースコアで始まり、文字、数字、およびアンダースコアのみが含まれるようにする必要があります。また、「false」または「true」という名前にすることはできません。「 」
- Design Studio で作成するマッピングの名前は、Management Console でのマッピングの名前と一致させる必要があります。
- Design Studio で使用されるラベルは、設計時に使用するデバイスと一致させる必要があります。
- Management Console のラベルは、本番環境で使用されるデバイスと一致させる必要があります。

Design Studio でロボットを開発およびデバッグするには、[デバイス マッピングの設定] ダイアログ ボックスの [デバイス マッピング] オプションを使用します。

オートメーション デバイス マッピングの追加

1. [ファイル] メニューで [新しいオートメーション デバイス マッピング] をクリックするか、[プロジェクト] リストでプロジェクトを右クリックして、**[**新規作成**]** > **[**オートメーション デバイス マッピング**]** を選択します。
2. [ファイル] メニューから [オートメーション デバイス マッピング] ウィザードを開始した場合は、名前を入力し、デバイスが関連付けられるプロジェクトを選択します。
3. それ以外の場合は、オートメーション デバイスの名前を入力し、**[**次へ**]** をクリックします。
   Management Console ベースのマッピングについては、Design Studio で作成するマッピングの名前が Management Console でのマッピングの名前と一致する必要があります。
4. [オートメーション デバイス マッピング] の設定ステップで、[**Management Console** ベースのデバイス マッピング] または [デバイス マッピング] を選択します。

   Management Console ベースのデバイス マッピング
   このオプションにより、Management Console でマッピングを使用してオートメーション デバイスに接続します。次のように設定します。
   - [**Management Console**]: マッピングに使用する Management Console を選択します。
   - [クラスタ名]: 選択した Management Console のクラスタ名を入力します。
   - [必須ラベル]: オートメーション デバイスに 1 つまたは複数のラベルを入力します。指定するラベルは、設計時に使用するデバイスと一致する必要があります。ラベルはコンマで区切る必要があります。

   デバイス マッピング
   このオプションにより、オートメーション デバイスに直接接続します。次のように設定します。
   - [ホスト]: オートメーション デバイス ホスト名または IP アドレスを入力します。
   - [ポート]: オートメーション デバイスに接続するコマンド ポート番号を入力します。
   - [トークン]: 選択したオートメーション デバイスのリモート ハブ設定で指定されているトークンを入力します。

オートメーション デバイス マッピングの編集

1. [オートメーション デバイス マッピング] を編集するには、プロジェクトでデバイス マッピングをダブルクリックするか、デバイス マッピングを右クリックして [設定] を選択します。

2. [デバイス マッピングの設定] ウィンドウで、[**Management Console** ベースのデバイス マッピング] または [デバイス マッピング] を選択します。

Management Console ベースのデバイス マッピング
このオプションにより、Management Console でマッピングを使用してオートメーション デバイスに接続します。次のように設定します。

- [**Management Console**]: マッピングに使用する Management Console を選択します。
- [クラスタ名]: 選択した Management Console のクラスタ名を入力します。
- [必須ラベル]: オートメーション デバイスの 1 つまたは複数のラベルを入力します。ラベルはコンマで区切る必要があります。

デバイス マッピング
このオプションにより、オートメーション デバイスに直接接続します。次のように設定します。

- [ホスト]: オートメーション デバイス ホスト名または IP アドレスを入力します。
- [ポート]: オートメーション デバイスに接続するときのポート番号を入力します。
- [トークン]: 選択したオートメーション デバイスのリモート ハブ設定で指定されているトークンを入力します。

ローカル Desktop Automation サービスの設定

ローカル Desktop Automation 機能を使用すると、自動化するデバイスと同じコンピュータ上でロボットを設計して実行することができるため、コンピュータの自動化プロセスが迅速かつ容易になります。ローカル Desktop Automation は、Windows オペレーティング システムでのみサポートされます。ローカル オートメーション デバイス上で一度に実行できるロボットは 1 つだけです。この機能は開発のみを目的としています。

ローカル Desktop Automation を有効にするには、以下の手順を実行します。

1. Desktop Automation サービスと Design Studio を同じコンピュータにインストールします。これは自動化するアプリケーションが実行されるコンピュータでもあります。詳細については、『Kofax RPA インストール ガイド』を参照してください。

   お使いのコンピュータにデュアル モニタが設定されている場合は、自動化されたアプリケーションを 1 台のモニタで開き、Design Studio を別のモニタで開くことができます。

2. 『Kofax RPA Desktop Automation サービス ガイド』の説明に従って Desktop Automation サービスを設定します。Desktop Automation サービス設定ウィンドウでプロパティを指定する場合は、[シングル ユーザー] オプションを選択して Design Studio から自動化されたアプリケーションへの直接接続を設定することをお勧めします。マッピングに使用するトークンを忘れずに入力します。

終了したら、以下の手順を実行します。

1. ローカル コンピュータにインストールされている Desktop Automation サービスへのマッピングを作成します。
   a. 作業中のプロジェクトを右クリックし、**[新規作成]** > **[オートメーション デバイス マッピング]** をクリックします。

   b. オートメーション デバイス マッピングの追加の説明に従ってフィールドを記入し、[終了] をクリックします。
2. Design Studio を開きます。
3. ロボットを作成します。
   a. **[**ファイル**]** > **[**新規ロボット**]** をクリックします。
   b. ロボットの名前を指定し、プロジェクトを選択します。[終了] をクリックします。

      エディター ウィンドウの新しいタブに新しいロボットが表示されます。最初にベーシック エンジン ロボット から新しいロボットを呼び出す必要があるため、この時点ではロボット ワークフローを編集できません。
4. 既存のベーシック エンジン ロボットを開くか、**[**ファイル**]** > **[**新しい **Web** オートメーション ロボット**]** をクリックして新しいロボットを作成します。
5. アクション ステップを挿入します。
   a. [アクション] タブで [アクションを選択] をクリックし、[ロボットを呼び出す] を選択します。
   b. [ロボット] ドロップダウン リストで、手順 4 で作成したロボットを選択します。

      同じタブで、入力値と戻り変数を設定します。
      - [デバイス] プロパティでプラス アイコンをクリックし、[静的リファレンス] を選択して、ステップ 1 で作成したマッピングを選択します。
      - **[OK]** をクリックします。
6. ツールバーの [実行を開始] をクリックして、新しく追加されたアクション ステップを実行します。

   [ロボットを呼び出す] ステップを実行した後に、ワークフロー自体を編集できます。編集を行うためには、ツールバーで [ロボットにステップ] をクリックします。

   ロボットが表示されているタブが開き、エディターがアクティブになります。タイトル バーには、ローカル Desktop Automation モードであることが示されます。これで、ロボットの設計を開始できるようになりました。
7. [レコーダー ビュー] で、自動化するアプリケーションを含むタブを選択します。アプリケーションはすでにコンピュータ上で開いている必要があります。または、アプリケーションを開くロボットに [開く] アクション ステップを追加できます。これで、アプリケーションで実行するステップを作成できます。
   - コンテキスト メニューやドロップダウン メニューなど、ポインタを削除すると消えるアプリケーション要素を自動化する必要がある場合は、バンドル ステップを使用します。バンドル ステップは、自動化されたアプリケーション上で実行するいくつかのステップを接続し、最初のステップから順番に実行されるシーケンスに変換します。
     - 既存のステップをバンドル ステップにラップするには、エディターで、消える要素が使用されているステップを選択し、グループを右クリックして [バンドル ステップで囲む] をクリックします。また、バンドル ステップをワークフロー内に直接挿入し、必要なステップを追加することもできます。
     - 右クリックまたは左クリックのアクションを含むバンドル ステップ、またはアプリケーション コンポーネントをポイントするバンドル ステップを挿入するには、[レコーダー ビュー] で、アプリケーションの必須のコンポーネントを右クリックし、**[Smart Focus** メニュー クリック**]** をクリックします。[右]、[左]、[ホバー] をそれぞれクリックします。

     アクション ステップをバンドル ステップに追加するには、ステップ内のフロー ポイントを右クリックして選択を行います。一部のステップはバンドル ステップ内部では使用できませんが、バンドル ステップの前または後のロボットに追加できます。

> バンドル ステップをその最初からプログラム内の特定のフロー ポイントまで実行するには、フロー ポイントをダブルクリックするか右クリックして [ここまで実行] をクリックします。ツールバーの [ステップ オーバー] または [実行を開始] ボタンを使用すると、バンドル ステップは常に最初から最後まで実行されます。

- 新しく追加されたアクション ステップをすぐに実行してストリーミングします。ステップを追加する前に、[レコーダー ビュー] で [自動実行] をクリックします (ボタンの円が赤色になります)。ステップが新しいアプリケーションまたはダイアログ ボックスを開くと、それぞれのタブがストリーム ビューに表示され、アクティブなタブになります。自動実行を停止するには、[自動実行] をもう一度クリックします。
- ポインタを削除すると非表示になるアプリケーション要素を操作するには、以下のホットキーの組み合わせを使用します。
  - ストリーミング ビューを一時停止するには、Ctrl+Shift+Alt+P を押します。
  - たとえば別のアプリケーションとしてのみ表示されるアプリケーションを操作する場合など、ストリーミング ビューのタブを切り替えるには、Ctrl+Shift+Alt+T を押します。

> これらのキーの組み合わせは、フォーカスがない場合にも機能します。

8. ロボット内の複数のアプリケーションを自動化する場合は、アプリケーションの間でフォーカスを切り替えます。デフォルトでは、実行が開始すると、フォーカスはロボット内で自動化されている最初のアプリケーションに設定されます。フォーカスを変更したり他の自動化されたアプリケーションに切り替えるには、それぞれのアプリケーションに [クリック] アクション ステップを追加します。アプリケーション内でクリックするステップを追加することも、Windows タスクバーのアプリケーションをクリックするステップを追加することもできます。
9. 変更を保存します。作成したワークフローを実行するには、[実行を開始] ボタンをクリックします。

   ワークフローの実行が開始すると、フォーカスはすぐに自動化されたアプリケーションに切り替えられ、ストリームのステータスは [ライブ] に変更されます。これは [レコーダー ビュー] の右下角に表示されます。

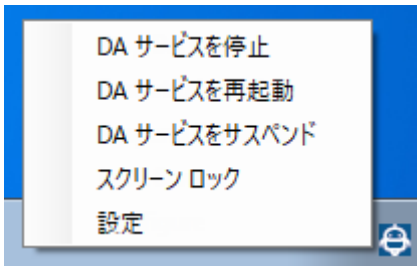
   実行が完了した後にフォーカスをロボット ワークフローに戻すには、ワークフローの内部をクリックします。ストリームのステータスは [一時停止中] に変更されます。ストリームが一時停止すると、アプリケーション ステータスは [レコーダー ビュー] で更新されません。

> ロボット ステップの実行中は、誤った実行操作を防ぐためにキーボードとマウスの操作が自動的に無効になり、ステップが完了するとこの無効化が解除されます。実行中にキーボードとマウスを使用する必要がある場合は、[Esc] を押します。

ローカル Desktop Automation モードで作成されたロボットを編集し、他のロボットと同様にリモート コンピュータで実行できます。

Desktop Automation サービスの管理

Desktop Automation サービスのショートカット メニューをクリックして、次のオプションにアクセスします。



これらのオプションを使用して、リモート コンピューターで実行されている Desktop Automation サービスを管理します。

- **DA** サービスを停止: サービスを停止します。これによりリモート デバイスは使用できなくなります。Desktop Automation サービスが実行されているコンピュータは、Management Console のリストから削除されます。
- **DA** サービスを再起動: サービスを停止してから、起動します。ロボットまたは Design Studio ではデバイスとの接続が失われるため、復元するにはリロードする必要があります。
- **DA** サービスをサスペンド: デバイスをサスペンドします。サービスをサスペンドすると、Management Console ではサスペンド状態と表示されます。サービス操作を復元するには、ユーザーまたは管理者はデバイスで Desktop Automation サービスを手動で起動する必要があります。
  サスペンド状態ではロボットは DA サービスを利用できませんが、状態情報は ping メカニズムを介して Management Console に送信され、**[管理]** > **[デバイス]** セクションにデバイスが表示されます。このコマンドは、何らかの理由でサービスまたはサービスを実行しているコンピュータの設定を変更する必要が生じた場合に役立ちます。
- スクリーン ロック: リモート デバイスのスクリーンをロックします。このオプションを使用する前に、Desktop Automation サービスをインストールして設定する必要があります。『Desktop Automation サービス ガイド』の「スクリーン ロックの使用」を参照してください。
- 設定: Desktop Automation サービスの設定ダイアログ ボックスを開きます。『Kofax RPA Desktop Automation サービス ガイド』を参照してください。

第 5 章

ロボットの構築

ロボットを構築して、以下のアプリケーションを含むプロセスを自動化します。

- Web サイトと Web アプリケーション
- ネイティブ Windows アプリケーション
- ネイティブ Java アプリケーション
- レガシー ターミナル アプリケーション
- Citrix クライアントなど、Windows システムに GUI を表示する他のアプリケーション

詳細については、ロボット構築の概要を参照してください。また、『ロボット構築の開始ガイド』も参照してください。

ロボット構築の概要

Kofax RPA では、Web サイト操作を自動化するロボット、およびネットワーク接続しているコンピューター上の Windows アプリケーションや Java アプリケーションに関連する作業プロセスを作成します。Design Studio には、この目的のための専用のワークフロー言語とステップが用意されています。Design Studio のロボット タイプの説明については、「ロボット」を参照してください。

ロボット

ロボットのワークフローは、順々に実行される一連のステップです。これらのステップは、自動化されているアプリケーションをユーザーが操作する方法をモデル化しています。

ロボット はスタンドアロン モードで実行することや、ベーシック エンジン ロボット から呼び出すことができます。

スタンドアロン モードのロボット

ロボット エディターを使用して、スタンドアロン ロボットを構築し、実行します。「ロボットの編集」および「ツールバー」も参照してください。

- Design Studio で、シンプル入力タイプおよびコンプレックス入力タイプのロボットを作成して実行します。
- Management Console で、コンプレックス入力タイプのロボットをキューに入れて実行します。

  シンプル入力タイプのロボットは、Management Console ではスタンドアロン モードで実行できません。

- 以前にベーシック エンジン ロボットから呼び出されたロボットがあり、それをスタンドアロンで実行したい場合は、出力値ステップを挿入することが出力を取得するための唯一の方法となります。

ベーシック エンジン ロボットから呼び出されたロボット

ロボットを呼び出すという名前の専用アクション ステップを使用して、ベーシック エンジン ロボットからロボットを呼び出します。

ベーシック エンジン ロボットには、それぞれ独自のワークフローを持つ複数の「ロボットを呼び出す」ステップを含めることができます。複数のベーシック エンジン ロボットで 1 つの ロボットを再利用できるため、複数のロボットを同時に操作すると時間を大幅に短縮できます。「ロボットを呼び出す」ステップを使用するロボットを他の Kofax RPA ロボットとして実行するには、スケジュール、API、Kapplet を使用するか、開発またはテスト中に手動で実行します。

ワークフロー

ワークフローは Design Studio で編集します。Design Studio には、ロボットと自動化中のアプリケーションのビュー、ロボット状態の詳細、およびロボットを手動で制御するためのボタンが配置された専用ツールバーが表示されます。詳細については、ロボットの編集 を参照してください。

メニューの [ヘルプ] ボタンから、関連する各ドキュメントおよびロボットの構築プロセスの概要が記載されたスタート ガイドへのリンクを利用できます。

ステップ

ステップは、ロボットのワークフローを構築する基本的なブロックです。ロボットには、終了ポイントがない一部のステップを除いたすべてのステップに、エントリ ポイントと終了ポイントが 1 つずつあります。一部のステップは単純なステップで、マウス移動やキー プレスなどの 1 つのアクションのみを実行します。複合ステップと呼ばれるその他のステップには、追加のステップが含まれていることがあります。複合ステップは、同じようなステップのグループ化、または分岐、および実行を続行する方法を制御するその他の方法の処理に使用します。ステップの完全なリストについては、ロボットのステップ を参照してください。

通常、ステップは、詳細かつより小さなタスクを処理します。たとえば、すべてのステップ タイプで固有のエラー処理はありません。代わりに、専用ステップが特に実行時のエラーの処理のために存在します。

デバイス

ロボットを使用する目的の 1 つとして、アプリケーションを自動で制御することが挙げられます。アプリケーションは、ネットワークによるリモート アクセスが可能なデバイス (コンピュータ、サーバー、または仮想マシン) で実行されます。ロボットは、リモート デバイスで実行される Desktop Automation サービスと接続することで自動化を実行します。ただし、デバイスがターミナルまたはその他の組み込みドライバーを実行している場合は、ロボットから直接接続されます。デバイスの処理とエージェントの設定の詳細については、『Kofax RPA Desktop Automation サービス ガイド』を参照してください。

アプリケーション ツリー

Kofax RPA は、アプリケーション ツリーを生成するいくつかの方法を提供します。デフォルトでは、Kofax RPA は、ロボットが動作しているアプリケーションのタイプ (Windows アプリケーション、ターミナル、組み込みブラウザなど) を検出し、このアプリケーションのツリーを自動的に形成します。一部の Windows アプリケーションについては、Kofax RPA に 拡張サポートが用意されています。たとえば、Design Studio で Internet Explorer モードの Microsoft Edge を使用している場合、Kofax RPA で Internet Explorer の拡張サポートが有効になり、DOM (Document Object Model) ツリーを取得することで、アプリケーション ツリーでより正確な結果を得ることができます。

Kofax RPA がアプリケーションから直接受け取る属性と Kofax RPA が追加する属性を区別するために、「拡張属性」のセットが提供されています。これは、異なる属性間の名前の競合を防ぐためのものです。Kofax RPA は、拡張属性として境界ボックス (x、y、幅、高さ) を追加します。拡張属性は、「`der_`」という接頭辞が付いたツリーに表示され、ファインダーで抽出する際に使用できます。

次の表に、アプリケーション ツリーで使用可能な拡張属性のリストとその説明を示します。

拡張属性	クラス	説明
der_x	一般	要素の左上隅の X 座標。
der_y	一般	要素の左上隅の Y 座標。
der_width	一般	要素の境界ボックスの幅。
der_height	一般	要素の境界ボックスの高さ。
der_tree_mode	一般	アプリケーションのツリーの生成方法を定義するツリー モードを指定します。 ISA で生成されたツリーには「ISA」を使用し、DAS ツリーには「Windows」または「None」を使用します。他のドライバーの場合は「Auto」を使用します。
der_original_node_name	一般	記述されているエンティティから派生したツリー内のノード名を指定します。この名前が XML に準拠していない場合は、正規化され、元の派生名がこの属性の値として渡されます。
der_index	テーブル表示	列インデックスを指定します。 この拡張属性をファインダーで使用しないことをお勧めします。この属性の値は、バージョン間で変更される可能性があるか、動的ソースに基づいて生成できます。 たとえば、これらのフィールドの可能なソースは、ユーザーまたはデータベース管理者がいつでも変更できる SQL クエリ (およびストアド プロシージャ) の出力です。
der_rpa_type	テーブル表示	データベース内のデータ タイプに最も一致する RPA データ タイプを指定します。 この拡張属性をファインダーで使用しないことをお勧めします。この属性の値は、バージョン間で変更される可能性があるか、動的ソースに基づいて生成できます。
der_rendered	CEF	ページに要素がレンダリングされる場合は、「y」(「はい」) に設定されます。
der_isOffscreen	CEF	要素が画面に表示されていない場合は、「true」に設定されます。要素を表示するには、ページをスクロールする必要があります。
der_isa	CEF	ISA によって要素が生成される場合は、「y」(「はい」) に設定されます。
der_value	CEF	「email」、「text」、「number」、「range」、「tel」、「time」、「url」、「search」、「date」、「datetime-local」、「week」、「color」、「month」、「textarea」の入力要素に使用されます。
der_checked	CEF	「radio」および「checkbox」の入力要素に使用されます。要素が選択されているのか、選択解除されているのかに応じて、「true」または「false」になります。

拡張属性	クラス	説明
der_visible	Windows	ネイティブの Windows グラフィック要素に使用されます。要素が表示可能かどうかに応じて、「true」または「false」になります。
der_subdriver	Windows	サブツリーを生成した Windows サブドライバーを指定します。可能な値は、「excel」、「ie」、「sap」、または「java」です。
der_SubWindow	Windows	Internet Explorer モードの Microsoft Edge ウィンドウのサブツリーが外部コンポーネントから生成されることを示します。可能な値は「Silverlight」と「JavaApplet」です。
der_handle	電子メール	電子メールを識別するために使用される内部参照。 この拡張属性をファインダーで使用しないことをお勧めします。この属性の値は動的に生成されます。
der_is_big_value	データベース	列が BLOB または同等のデータベース タイプである場合は、「true」に設定されます。

特定のアプリケーションで問題が発生した場合は、拡張アプリケーション サポートをオフにすることができます。

ロボット とベーシック エンジン ロボット の比較

Kofax RPA は本来、HTML ページが主に静的な場合に一度に HTML にアクセスするように設計されました。これらのケースでは、アプリケーション (Web ページ) の状態を口ボットで内部的に追跡できます。一方、口ボット は、状態がアプリケーションに存在する新しい動的な Web サイトおよびリモート アプリケーションを自動化するように設計されています。この場合、状態は口ボットの外部になります。

口ボットでのステップの実行は、前方への移動に限定されます。リモート コンピュータを自動化している場合、実行の状態はリモート デバイス上にあるため、「値を抽出」ステップと「値の変換」ステップのグループ以外のステップについてはワークフローで前のステップに戻ってから元に戻すことはできません。その結果、ワークフローの設計時に、新しく挿入されたステップは 口ボットの編集 で明示的に実行を選択するまで実行されません。

分岐は、ベーシック エンジン 口ボットで設計されているため、口ボットには存在しません。複合ステップの一部としてのみ発生します。

分岐は、条件などの複合ステップの一部としてのみ発生します。分岐は代替分岐であるため、ワークフローの実行時に選択される分岐は 1 つだけです。これは、分岐が順番に実行され、各分岐の開始時に状態が戻るベーシック エンジン 口ボットとは異なります。

口ボットでは、エラーの処理方法が指定されていないステップもあるため、エラーの処理はステップによって異なります。代わりに、トライ-キャッチ ステップ ではそのスコープ内で発生するエラーがキャッチされ、それらの処理方法が定義されます。

通常は、口ボットを設計する際に、自動化を行うアプリケーションのユーザー インターフェイスをユーザーがどのように操作するのかを考慮します。たとえば、テキスト フィールドにテキストを入力する必要がある場合、最初にフィールドをクリックしてから、そのテキストを入力するステップを挿入します。

口ボットには、口ボットの設計者がオートメーションを設計してアプリケーションの外部状態を判断し、適切に対応するための機能があります。たとえば、ボタンのクリックがボタンが表示されるまで

待機できるようになります。または、ステップでアプリケーションがすでに開始されていることを検出し、別のインスタンスが開始されないようにすることができます。ロボット ワークフローの設計時は、ガードとファインダーがアプリケーションの特定の状態を待機するために使用され、これにより ロボットが必要な要素を見つけ、それらと予想どおりに相互作用するようになります。ガードの詳細については ガード チョイス、ファインダーの詳細については ファインダーをそれぞれ参照してください。

プロセスの自動化を開始する方法については、はじめにを参照してください。また、アクション ステップのリストについては、ロボットのステップを参照してください。

はじめに

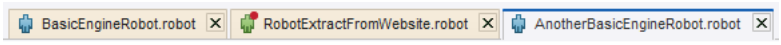
次の説明は、Kofax RPA をいずれかのコンピューターにインストールしていることを前提としています。Kofax RPA の使用方法については、『Kofax RPA インストール ガイド』の「クイック スタート ガイド」を参照してください。

1. デバイスを自動化する方法については、Desktop Automation サービス を参照してください。
2. ターミナル アプリケーションを自動化する方法については、ターミナル エミュレータの自動化 を参照してください。
3. インストールと設定タスクが完了した後に、Design Studio を開きます。
   インストールおよび設定タスクの多くは、管理者ロールまたは管理者と同様のロールによって実行されます。
4. ロボットを作成します。
   a. **[ファイル] > [新規ロボット]** をクリックします。
   b. ロボットの名前を指定し、プロジェクトを選択します。
   c. [終了] をクリックします。
      エディター ウィンドウの新しいタブに新しいロボットが表示されます。ベーシック エンジン ロボットは青いアイコン で識別され、ロボットは緑色のアイコン で識別されます。
      ロボット ワークフローの基本的な編集を実行します。

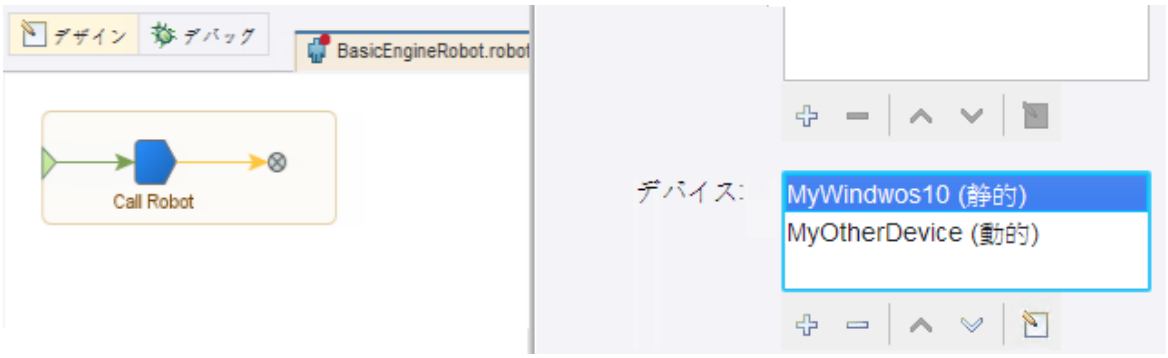
   ロボットを実行するには、外部アプリケーションを自動化するだけでなく、データを操作し、レコーダー ビューでロボットの状態を追跡して、状態ペインで変数の状態を追跡し、ベーシック エンジン ロボットからロボットを呼び出します。
5. 既存のベーシック エンジン ロボットを開くか、**[ファイル] > [新しいベーシック エンジン ロボット]** をクリックして新しいロボットを作成します。
6. ベーシック エンジン ロボットの実行を許可するには、ツールバーまたは [アプリケーション] ペインで [実行の準備] をクリックします。
7. あるロボットから別のロボットに実行権限を割り当てるには、必要なロボットのタブを開き、**[実行の準備]** をクリックします。

実行権限を持つことができるのは、一度に 1 つのベーシック エンジン ロボットだけです。デバッグ モードでは複数のベーシック エンジン ロボットを実行できますが、デザイン モードでは 1 つのベーシック エンジン ロボットのみが実行する権限を持ちます。デザイン モードでの実行権限を持つロボットは、2 つのモードを切り替えて、デバッグ モードでも同時に実行することができます。詳細については、「デバッグ モード」を参照してください。

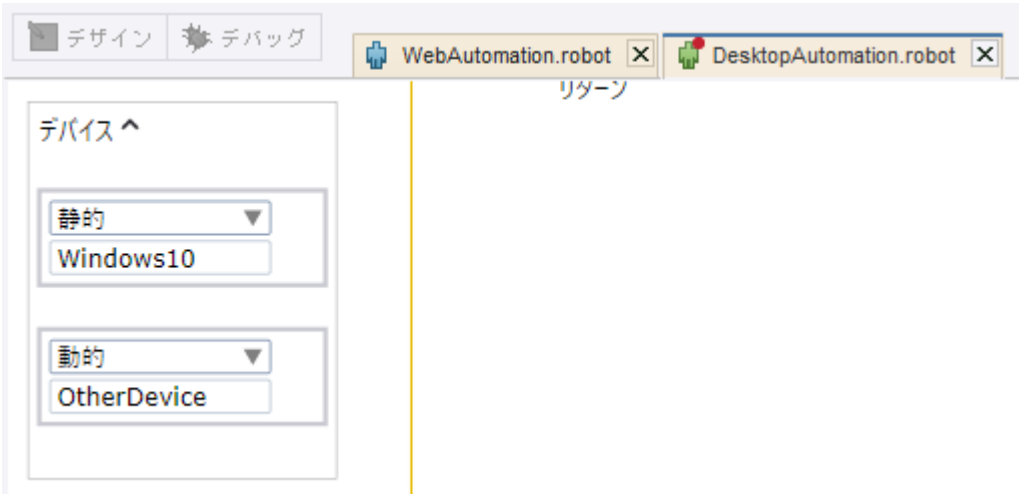
ロボットが実行権限を保持している場合は、そのロボットのタブに赤い点が表示されます。現在実行中の他のすべてのロボットは、次の図に示すようにハイライト表示されます。



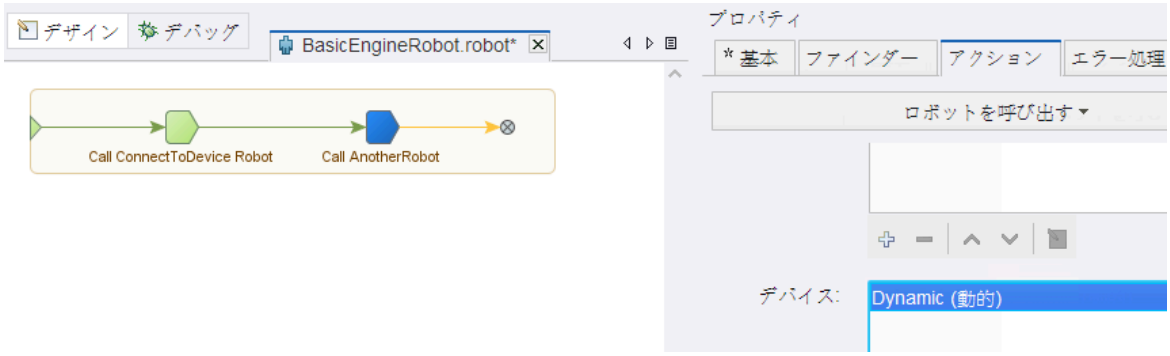
8. ベーシック エンジン ロボットに[ロボットを呼び出す]ステップを挿入します。
   a. このステップの [ロボット] ドロップダウン リストで、手順 4 で作成したロボットを選択します。
      後から、**[ロボット]** ドロップダウン リストで **[新規作成]** をクリックして新しいロボットを作成することもできます。ベーシック エンジン ロボットの値とマッピングを指定してから、新しいロボットを作成すると、新しいロボットはこれらのプロパティを継承します。
   b. 入力値、戻り変数、およびデバイスを設定します。オートメーション デバイスの準備および「オートメーション デバイス マッピング」を参照してください。ターミナル アプリケーションのみを自動化する場合は、デバイスの設定をスキップします。
9. 実行が許可されている場合に、**[ロボットを呼び出す]**ステップに対してワークフローを実行してから、ツールバーの **[ロボットにステップ]** をクリックします。
   ロボットのタブが開き、ロボットの編集と実行ができるようになります。
10. 外部アプリケーションを自動化するには、[デバイス] ボックスでオートメーション デバイスを指定します。ターミナル アプリケーションのみを自動化する場合は、このステップをスキップします。
    - ベーシック エンジン ロボットおよびロボットで指定したデバイスの数が一致している必要があります。

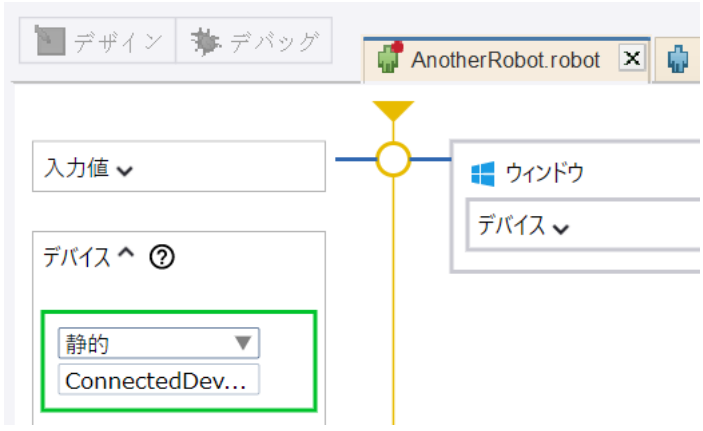


    - ロボットで設定したデバイス名は、ベーシック エンジン ロボットのデバイス名と異なる場合があります。ロボット エディターでは、必要に応じてコンテキスト メニューからデバイスの名前を変更できます。ここでデバイスの名前を変更すると、そのデバイスが使用されるすべてのステップで名前の変更が反映されます。



- ロボット内の静的デバイスは、接続がすでに確立されているデバイスです。ベーシック エンジン ロボットの [デバイス] プロパティの [ロボットを呼び出す] ステップに動的マッピングが指定されており、そのデバイスが以前に動的マッピングを使用して接続されていた場合、ロボットの [デバイス] ペインでは、使用する静的デバイスを選択し、動的マッピングで確立された接続を再利用できます。これは、このロボットを別のベーシック エンジン ロボットで再利用する必要がある場合に便利です。





11. ロボットのオートメーション ワークフローを設計し、ワークフローを実行して動作を確認します。

Kofax RPA バージョン 10.7.0 以前からアップグレードする場合は、古い Desktop Automation アクション ステップの変換 を参照してください。

12. ロボットの設計が完了した後に、そのロボットを実行して、デバイスを自動化します。
    - ロボットからステップ アウトして、ベーシック エンジン ロボットでの作業に切り替えるには、ロボットのワークフロー全体を実行し、ツールバーの [ステップ アウト] をクリックします。

      ベーシック エンジン ロボットで、**[ロボットを呼び出す]** ステップが実行済みと表示されます。
    - 最後まで実行しない終了する場合、または結果を返さないでロボットを終了する場合は、ツールバーの [ロボットの終了] をクリックします。

      ベーシック エンジン ロボットで、**[ロボットを呼び出す]** ステップが実行中ではないと表示されます。

古い Desktop Automation アクション ステップの変換

Kofax RPA バージョン 10.7 0 以前では、Desktop Automation アクション ステップに含まれるロボット ワークフローを編集する際に、スタンドアロンの Desktop Automation エディターを使用していました。Kofax RPA 11.0.0 以降では、バージョン 10.7.0 以前で作成された Desktop Automation アクション ステップを実行することができます。ただし、ワークフローを編集するには、ステップから新しいロボットにワークフローを抽出し、Desktop Automation ステップを、新しいロボットを参照する「ロボットを呼び出す」ステップに変更する必要があります。

ベーシック エンジン ロボットと、Desktop Automation ステップを含むスニペットをエクスポートできます。

ロボットにエクスポートした後は、最初の Desktop Automation ステップに戻ることはできません。

1. エクスポートする前に、ロボットを開いて Desktop Automation ステップを選択し、ステップ プロパティで **[プレビュー]** をクリックしてロボットをプレビューして、エクスポート後にどのような状態となるかを確認します。
2. プロジェクト ツリーから変換してエクスポートします。
    - アクション ステップをシングル ロボットのロボットに変換するには、Desktop Automation ステップを含むベーシック エンジン ロボットを右クリックします。
    - 複数のロボットを変換するには、[プロジェクト] ビューで任意のフォルダを右クリックするか、Desktop Automation ステップを含む複数のベーシック エンジン ロボットを選択して右クリックします。

      抽出可能な Desktop Automation ステップが一覧で表示されたダイアログ ボックスが表示されます。
3. **[ロボットのエクスポート]** をクリックします。
4. 必要に応じて、選択したロボットにわかりやすい新しい名前を割り当て、ロボットをプレビューして、そのロボットを含むベーシック エンジン ロボットの Desktop Automation ステップを表示します。
    - 選択したファイルの名前を変更するには、 をクリックします。
    - エクスポート後に作成される、選択したロボットをプレビューするには、 をクリックします。
    - 選択した Desktop Automation ステップをそのステップが含まれるベーシック エンジン ロボットで表示するには、 をクリックします。

ワークフロー プレビュー ウィンドウでズーム レベルを変更すると、ロボット エディターのズーム レベルが変更されます。

5. [次へ] をクリックして、現在のプロジェクト内にあるエクスポートされたロボットの場所を選択します。
6. [終了] をクリックしてダイアログ ボックスを閉じ、エクスポートを開始します。
   エクスポートの概要が表示され、エクスポートしたロボットおよびスニペットの数が一覧で表示されます。ベーシック エンジン ロボットで設定されたすべてのデバイスは、自動的にロボットに表示されます。

ロボット設定

ロボットを構成するには、次に説明する各プロパティを使用します。**[ロボットの設定]** ウィンドウは、ツールバーの **[ロボット設定]** をクリックするか、Ctrl+R を押すと表示されます。また、**[ファイル]** メニューから **[ロボット設定]** を選択することもできます。

[基本] タブ

ロボット コメント

ロボットについてのコメントを入力します。

ロボットのサムネイル

ロボットの画像を追加します。

[ロード] をクリックして、必要なファイルがあるフォルダに移動します。推奨される画像タイプは PNG です。画像をアップロードすると、自動的に 20x20 ピクセルにスケーリングされます。ファイルを開き、**[OK]** をクリックして、ロボットを保存します。

異なるタイプのファイルをアップロードしようとすると、エラー メッセージが表示されます。

画像を削除するには、**[ロボットの設定]** ウィンドウを開き、ロボットのサムネイルの近くにあるアスタリスクをダブルクリックします。ロボットのサムネイルがデフォルトとして設定され、プレビュー ウィンドウに「画像なし」というテキストが表示されます。

[OK] をクリックしてロボットを保存します。

ロボット タグ

ロボット用に 1 つ以上のタグを作成します。タグは、Management Console の **[リポジトリ]** > **[ロボット]** ページにある [タグ] 列に表示されます。タグを使用して、Management Console にあるロボットのリストをフィルタリングできます。タグには文字、数字、および下線を含めることができます。タグには 255 文字を使用できます。255 文字以上の文字を入力すると、最初の 255 文字のみが保存されます。同一のタグを 2 つ入力すると、赤い警告アイコンが表示されます。

手動の処理時間

このオプションでは、選択したロボットによって実行時に行われるタスクをユーザーが実行する場合の所要時間を分単位で指定できます。Kofax Analytics for RPA の [概要] レポートの「節約された人手の処理時間」テーブルに、指定した値とロボットの実際の実行時間の差が表示されます。

バージョン

保存されたロボットのバージョンおよびロボットの編集を最後に行った Design Studio のバージョンを表示します。

ロボットの編集

ロボットの編集については、次のトピックを参照してください。

- エディター
- ロボット定義
- レコーダー ビュー
- アプリケーション レベルのアクション
- 検索結果
- コメント
- 状態
- 出力ログ
- ヒント

エディター

エディターには、ワークフローと変数、式、およびツリー モードの設定などのパラメータが含まれています。

アクションとナビゲーション

- エディターのウィンドウを切り離して移動し、編集しやすくします。
- ロボットの実行を一時停止またはリセットします。
- ツールバーのボタンを使用して、ワークフローのステップをナビゲートします。
- [実行を開始]、[ステップ]、[ステップ オーバー]、および [ステップ アウト] ボタンを使用してワークフローを進めます。
- 次のキーの組み合わせを使用して、複数のステップを選択します。
  - Shift + クリック: ステップの範囲を選択します。
  - Ctrl + クリック: ステップの選択を追加または削除します。
  - Ctrl + Shift +クリック: ステップの選択の範囲を追加します。

ツールバー ボタン

ロボットのボタンのリストについては、ツールバーを参照してください。

エディターの拡大または縮小

Web ブラウザと同じ方法でワークフロー ビューを拡大または縮小できます。

- 拡大するには、Ctrl + (プラス記号) を押すか、Ctrl を押したままマウス ホイールを上にスクロールします。
- 縮小するには、Ctrl - (マイナス記号) を押すか、Ctrl を押したままマウス ホイールを下にスクロールします。

ロボット定義

ツールバー、[ロボット定義] ペイン、およびコンテキスト メニューを使用して、次のロボット定義を追加、設定、および変更します。

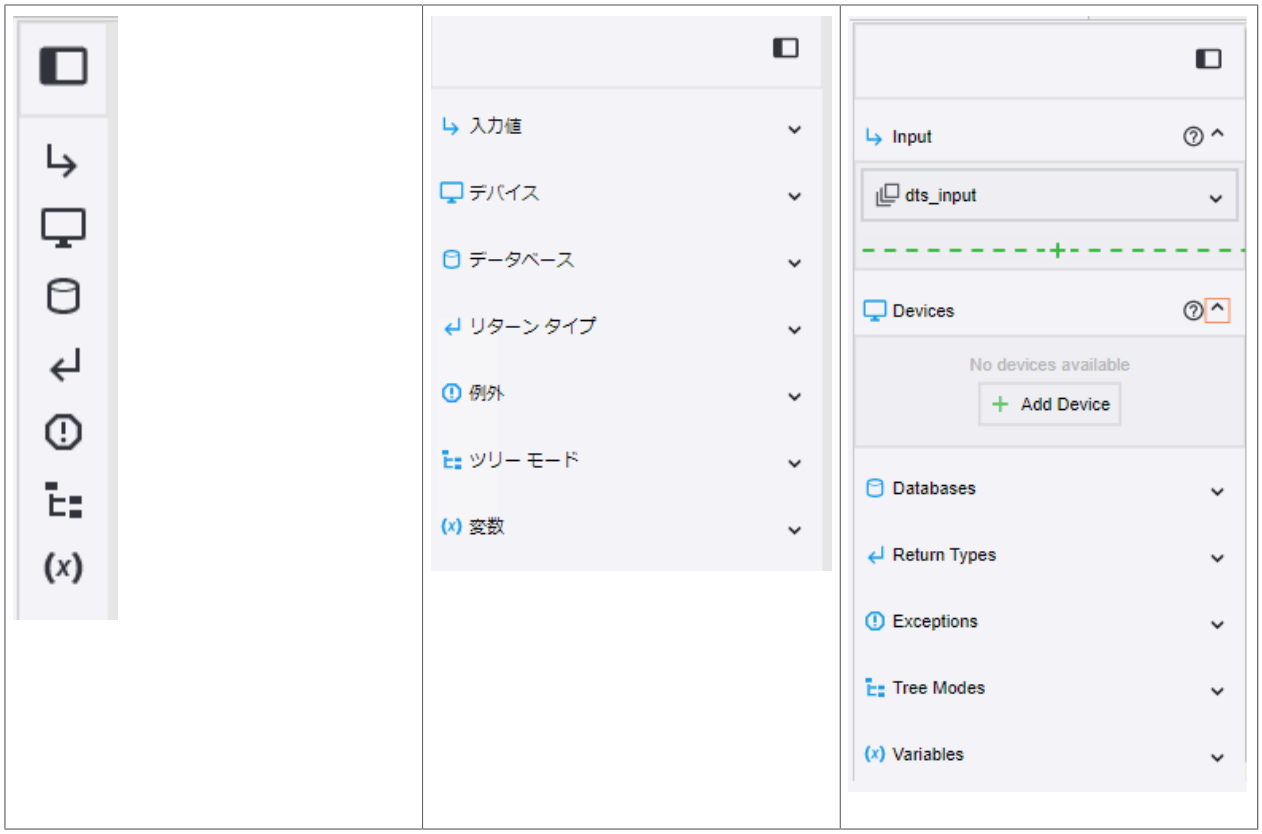
- 入力値: テスト値を含む、ロボットが受け取る入力パラメータの定義を表示します。
- デバイス: ロボットが使用する必要なデバイスを定義します。
- データベース: ロボットが使用する必要なデータベース マッピングを定義します。
- リターン タイプ: ロボットがベーシック エンジン ロボットに配信するリターン タイプのリストが表示されます。
- 例外処理: 発生した例外を表示します。
- ツリー モード: 配信するさまざまなアプリケーションのアプリケーション ツリー タイプを設定します。
- 変数: ロボット変数を設定します。

ツールバーとペイン

アイコンをクリックすることで、ツールバーからロボット定義のペインまでビューを折りたたんだり展開したりすることができます。各ビューには、定義領域が同じ順序で一覧表示されます。

- ロボット定義ツールバーのアイコンをクリックして、対応する定義領域に直接移動します。
- 上下のキャレット マークをクリックすることで、各定義と関連要素を展開したり折りたたんだりすることができます。定義を追加および表示したり、**[?]** 記号をクリックして関連トピックに直接移動して『Kofax RPA のヘルプ』のトピックにアクセスしたりするには、キャレット マークを上矢印にする必要があります。
- 領域に定義がある場合は、既存の定義の後に緑色のプラス記号 `---+---` が表示されます。定義を追加するには、緑色のプラス記号をクリックします。
- 領域に定義がない場合は「利用可能なデバイスがありません」などのメッセージが表示され、その後に `+ Add Device` などのボタンが表示されます。ボタンをクリックして最初の定義を追加します。

次の図は、ロボット定義ツールバー、ペイン、および定義を追加および編集する展開された領域を示しています。

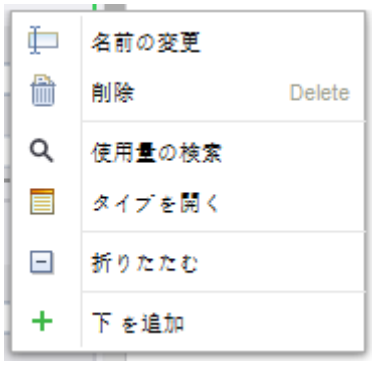


コンテキスト メニュー

定義、パラメータ、または [ロボット定義] ペイン内で右クリックして、コンテキスト メニューを表示します。

オプションは右クリックした場所によって異なります。

次の例は、レコード タイプの入力パラメータのメニューがどのように表示されるかを示しています。



次の表に、すべてのコンテキスト メニュー オプションと、それらのオプションが使用できる条件に関する説明を示します。

ボタン	ツールチップ	説明
	名前の変更	選択した要素の名前を変更します。
	削除	選択した要素を削除します。 [削除] オプションは、[例外] 領域を除くすべての領域で使用できます。
	値の更新	レコード値の値を更新します。このオプションは、値が最後に編集されてから値のタイプが変更された場合にのみアクティブになります。
	使用量の検索	[入力値]、[デバイス]、[変数] の場合、[使用量の検索] 機能を使用して、使用されている場所を特定することができます。 このオプションはテキスト検索と似ていますが、特定の要素のインスタンスのみが [検索結果] パネルのリストに表示される点が異なります。
	タイプを開く	Design Studio タイプ エディターでタイプを開きます。パラメータまたは変数定義からタイプを開きます。 [タイプを開く] は、[入力値] と [変数] の定義および [リターン タイプ] で使用できます。 「タイプ」を参照してください。
	マイ プロジェクト ビューで選択	[マイ プロジェクト] ビューで対応するマッピング ファイルを選択 (検索) すると、[マイ プロジェクト] ビューでファイルをダブルクリックして編集できるようになります。 このオプションは、[デバイス] 定義および [データベース] 定義で使用できます。
	展開	アクティブな定義内のフィールドとエントリを展開します。
	折りたたむ	フィールドを折りたたんで、選択した要素のエントリを開きます。
	追加	新しい要素を追加します。 この右クリック オプションを使用すると、定義に基づいて、一致するテキストが表示されます。 例: [入力値を追加]、[デバイスの追加]、および [データベースの追加]。
	下を追加	現在の選択の下に新しい要素を追加します。 [下を追加] オプションは、[例外] 領域を除くすべての領域で使用できます。 • 既存のエントリの間に新しい定義を追加するには、前の定義を選択し、[下を追加] を選択して新しい定義を挿入します。 • 新しい定義を上部に追加するには、[ロボット定義] ペインの上部にカーソルを置き、コンテキスト メニューの [追加] オプションを使用して要素を追加します。

レコーダー ビュー

[アプリケーションを開く] ウィンドウを含むタブと、利用可能な要素を含むツリーを表示します。インターフェイスで要素を選択するか、イメージを選択し、選択した要素またはイメージを右クリックして、ステップを挿入します。

ビューには、デバイス状態のライブ ストリーミング ステータスが表示されます。アプリケーション レベルのアクションも参照してください。

レコーダー ビューのツールバー

ボタン	ロールオーバー テキスト	説明
		デバイス状態のライブ ストリーミングを一時停止または再開します。 クリックするとレコーダー ビューでストリーミングが一時停止または再開します。 • ストリーミングがアクティブな場合、ステータスは ストリーム ライブ になります。 • ストリーミングが一時停止されている場合、ステータスは ストリーム 一時停止中 になります。
	選択した範囲に対してファインダーを作成	[レコーダー ビュー] で選択した要素のファインダーを作成し、ロボット内の選択したファインダーを置き換えます。 これは、選択したファインダーが現在のフロー ポイントの直後のステップ上にある場合にのみ有効になります。
	検出された次の要素を表示	ファインダーに一致する次の要素を表示します。 ツールチップには、一致した要素の数が表示されます。
	クリックに一致する次のノードを選択	選択を [レコーダー ビュー] での選択に一致する次のノードに移動します。
	ズーム レベルの選択	[レコーダー ビュー] でズーム インまたはズーム アウトします。
	1 レベル外側のタグを選択	選択したノードの親ノードに選択を変更します。
	1 レベル内側のタグを選択	選択した最初の子ノードを選択します。
	最も外側のタグを選択	アプリケーション ツリーのトップノードに対する選択を変更します。
	最も内側のタグを選択	選択した最後の子ノードを選択します。
	前の同位層ノードを選択	アプリケーション ツリーで同じレベルにある前のノードを選択します。
	次の同位層ノードを選択	アプリケーション ツリーで同じレベルにある次のノードを選択します。

ボタン	ロールオーバー テキスト	説明
	サブ ツリーを XML としてコピー	ツリー ビューで選択したサブ ツリー要素をコピーします。
		[レコーダー ビュー] から挿入されたステップをすぐに実行します。 1 つまたは複数のステップを追加する前にこのボタンをクリックします。 ステップが新しいアプリケーションまたはダイアログを開くと、それぞれのタブがストリーム ビューに表示され、アクティブなタブになります。 自動実行を停止するには、このボタンをもう一度クリックします。

座標

レコーダー ビューの下部には、アプリケーション ウィンドウの上部を基準としたマウスの座標が表示されます。

ビューで要素を選択すると、選択した要素の上部を基準とした座標、および下部のバーの要素へのパスが表示されます。

カラー	説明
■	マウス座標は、ウィンドウの左上隅を基準として表示します。
■	紫色の領域を選択する場合、紫色のボックスの横にある数字は、紫色の領域の左上隅を基準にしたマウスの移動量を示しています。 イメージ ファインダーを使用するために画像上で領域が選択されている場合、紫色の領域のサイズがピクセル単位で表示されます。 • 黒色のボックスの横にある数字は、ウィンドウの上部を基準にしたマウスの座標を示しています。 • 緑色のボックスの横にある数字は、緑色の領域の上部を基準にしたマウスの移動量を示しています。 • 緑色の領域は、選択した紫色の領域を含む要素を示しています。
■	ツリーまたは画像上で要素をクリックすると、緑色の領域の左上隅を基準にしたマウスの移動量がピクセル単位で表示されます。
■	ツリー上で要素をクリックすると、オレンジ色の領域の左上隅を基準にしたマウスの移動量がピクセル単位で表示されます。
■	ロボットにループが含まれている場合、ループを含む親要素と、ループが現在オンになっている要素を示します。

タグは、現在の状態対応する色の付いたボックスでマークされます。

- 緑色のボックスは、選択済みのプライマリ タグを示します。
- オレンジ色のボックスは、1 つ下のレベルのタグを示します。
- 青色のボックスは、さらに 1 つ下 (3 番目) のタグを示します。

レコーダー ビューのすべてのタグ パスはインタラクティブです。

- パスのタグを左クリックして、ノードを選択済みのタグにします。
- タグを右クリックしてノードのコンテキスト メニューを開きます。

要素間の切り替え

異なるレベルの要素を切り替えるには、次のボタンを使用します。

- 最も外側のタグを選択
- 1 レベル内側のタグを選択
- 1 レベル外側のタグを選択
- 最も内側のタグを選択
- 前の同位層ノードを選択
- 次の同位層ノードを選択

アプリケーション ビューのテーブルのセル要素を選択できないことがあります。これらの場合、アプリケーション ツリーのセル要素を選択したり、ツリー ビューからステップを追加したりします。

レコーダー ビューの拡大または縮小

[レコーダー ビュー] は、ツールバーのズーム レベルを選択するか、Web ブラウザと同じ方法で拡大および縮小できます。

- 拡大するには、Ctrl + (プラス記号) を押すか、マウス ホイールを上にスクロールします。
- 縮小するには、Ctrl - (マイナス記号) を押すか、マウス ホイールを下にスクロールします。

アプリケーション レベルのアクション

アプリケーション レベルのアクションとはアプリケーション全体に適用されるアクションのことで、このアクションを使用するには、[レコーダー ビュー] のアプリケーション タブを右クリックします。これらには、次のアクションとステップが含まれます。

- テキストを入力
- キープレス
- スクロール
- ガード
- トリガー
- ツリー モード
- アプリケーション アクション
  - フォーカス: このアクションは、リモート デスクトップ アプリケーションで使用できます。
  - Excel、Document Transformation ブラウザ、Web サイト ブラウザ、ターミナル エミュレータなど、ローカルで実行されているアプリケーションの場合、アプリケーション アクションのリストは異なります。詳細については、関連トピックを参照してください。
- デベロッパー ツールを起動

検索結果

検索 の結果の概要を表示します。

コメント

ステップ、グループ ステップ、変数、ファインダー、入力タイプ、条件分岐など、オートメーション ワークフローで選択可能なアイテムのコメントを表示します。

コメントを作成または変更するには、アイテムをクリックして、[コメント] ウィンドウでメモを追加または変更します。ここでは [元に戻す] および [やり直し] ボタンを使用します。コメントは、ウィンドウの外側をクリックすると自動的に保存されます。

コメントを含むアイテムには、コメント記号が付きます。

状態

ワークフローの実行状態と、入力、変数、ファインダー、データベース、戻り値、出力値などの含まれる要素を表示します。関連項目: 状態ペインとデータの状態ペイン。

コンテキスト依存のポップアップ メニューが表示されます。ポップアップ メニュー オプションを使用してステップを挿入すると、クリックした要素からの情報を使用して自動的に設定されます。このオプションは、手動でステップを入力して設定するよりも高速かつ効率的です。

また、[状態] ペインのエントリまたはフィールドを右クリックすると、次のようなタスクを実行するためのポップアップ メニュー オプションが表示されます。

- 値をコピーする。
- 値を貼り付ける。

  貼り付けオプションは、[値の設定] オプションでは変更できないバイナリ値の場合に特に便利です。「値をクリア」も参照してください。
- 値を設定する。
- 値をクリア

  [クリア] オプションは、新しいバイナリ値を貼り付ける前にバイナリ フィールドをクリアする場合に便利です。クリップボードにコピーしたバイナリ項目を、空のバイナリ フィールドに貼り付けます。
- 選択した項目の開いているフィールドまたは要素をすべて折りたたみます。

  ペイン内のすべての要素を折りたたむには、[状態] を右クリックし、[すべて折りたたむ] を選択します。
- 次のステップを挿入します。(関連項目: ステップ。)
  - 割り当て先
  - 変換
  - [データベース データ登録]、[データベース データ抽出]、または [データベースから削除]
  - キーの計算
  - レコード タイプの変数の [出力値]

    たとえば、変数を右クリックし、[ステップを挿入] > [出力] を選択して、[出力値] ステップを実行します。結果が、[状態] ペインの出力値の下に表示されます。
- 変数または入力値を一時的に変更します。

  このオプションは、ロボットの操作に慣れたプログラム担当者がデバッグ中に使用する場合に便利です。詳細については、以下を参照してください。

変数または入力値フィールドを一時的に変更するには、次の手順を実行します。

1. 値をダブルクリックするか、値の上にポインタを移動させてポップアップ メニューから [値の設定] を選択して、値のテキスト フィールドを開きます。
2. 新しい値を入力します。
   この値は、要素の有効なエントリを表すテキスト エントリである必要があります。たとえば、ブール値には true または false のいずれかが必要です。
3. 変更を確定するには、Enter キーを押します。
   Enter キーを押さずにテキスト フィールドの外側をクリックすると、値は変更されずにフィールドが閉じます。(元の値に戻ります。)
   一時的な変更は、ロボットによって値が変更されるか、実行がリセットされるまで有効になります。

Kofax RPA でバイナリ変数にイメージが含まれていることが検出されると、そのイメージがビューに表示されます。Kofax RPA では GIF、JPEG、BMP、TIFF、および PNG 形式のイメージが検出されます。イメージ ツールチップには、イメージの MIME タイプと、そのサイズ (幅と高さ) が表示されます。

出力ログ

ワークフロー実行メッセージが含まれます。

ヒント

ワークフローのエラーにフォーカスを合わせる
ロボットのフロー ポイントからステップが挿入されていて、そのステップに 1 つまたは複数のエラーが含まれている場合は、展開されたステップで最初のエラーの場所が示されます。ステップのエラーが原因でロボットの実行が停止した場合は、そのステップは展開され、最初のエラーの場所が表示されます。「「～で囲む」」ステップを使用してワークフローにステップが挿入された場合は、それらのステップが展開されます。

記録されていないインスタント マウス クリックの実行

ロボット ワークフローに追加せずに、要素にインスタント マウス クリックを実行します。このアクションは、不要なタブや誤って開いたタブを閉じる場合に役立ちます。[レコーダー ビュー] の要素を右クリックし、[記録されないインスタント クリック] をクリックして、使用するマウス クリックのタイプを選択します。

組み込みアプリケーションを閉じる
ロボットの編集中に、ブラウザ、Excel などの組み込みアプリケーションのアプリケーション タブを閉じるには、タブの右上隅にある [閉じる] ボタンをクリックします。このアクションはステップではなく、タブはウィンドウと同様に閉じます。

入力とリターン タイプ

ロボット定義ペインで [入力値] パラメータと [リターン タイプ] を設定します。

入力

[ロボット定義] ペインの [入力値] 領域を使用して、ロボットの入力パラメータを定義します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

ロボット がベーシック エンジン ロボット または Management Console から呼び出されたときに、入力値のリストが渡される場合があります。ロボットは、これらの値にアクセスするために、[ロボット定義] ペインの [入力値] 領域で定義された入力パラメータの一致リストを使用します。

入力パラメータの数は入力値の数と一致する必要があります。また、それらのタイプは、対応する値のタイプと一致する必要があります。入力値は、並び順に入力パラメータにバインドされます。

ロボット内で、エクスプレッションでパラメータを使用します。

- エクスプレッションのパラメータは、変数を使用する場合と同じように、名前で参照する必要があります。
  例：固定値 Age という入力パラメータがある場合は、この値を Age という変数として使用できます。
- 入力パラメータがロボットによって変更されることはありません。また、割り当てステップの [変数] など、値が変数に割り当てられる場所では入力パラメータを使用できません。

シンプル タイプの入力を持つロボットについては、実行や実行のスケジュールを行うことはできず、ベーシック エンジン ロボットからのみ呼び出すことができます。

入力を追加する

1. [入力値] 領域が展開されていない場合は、下矢印キャレット マークをクリックすると上矢印に変わります。
2. 定義が存在しない場合は、**[入力値を追加]** ボタンをクリックするか、コンテキスト メニューを右クリックして **[入力値を追加]** を選択します。
3. 定義が存在する場合は、次のいずれかの方法を使用します。
   - 既存のエントリの後に追加するには、緑色のプラス記号をクリックします。
   - 既存のエントリの間に追加するには、前の入力を選択し、**[下を追加]** を選択して新しいデータベースを挿入します。
   - 既存のリストの先頭に追加するには、先頭にカーソルを置き、コンテキスト メニューから **[入力値を追加]** オプションを選択します。
4. ドロップダウン リストから、入力の **[タイプ]** を選択します。
5. 入力値に名前を付けて Enter を押します。

6. ロボットがスタンドアロン モードで編集されている場合は、**[値判定]** キャレット マークをクリックして、[値判定] パラメータの入力を完了します。
   「入力エディターと検証」を参照してください。このフィールドはスタンドアロン モードでのみ表示されます。

入力を削除する

データベースを削除するには、[ロボット定義] ペインの [入力値] 領域を使用します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

入力パラメータを削除するには、入力パラメータを右クリックし、**[削除]** をクリックします。

レコード タイプを更新する

タイプを変更し、ロボットがそのタイプを入力に使用するようにする場合は、入力定義のレコード タイプの値を変更する必要があります。

レコード タイプの値を更新するには、**[値の更新]** コンテキスト メニュー オプションを使用します。

レコード値の更新が必要な変更と、[値の更新] オプションの機能に関する説明を以下に示します。

- フィールドを削除すると、そのフィールドの値も削除されます。
- フィールドを追加すると、そのフィールドはレコード値に追加され、フィールドのタイプのデフォルト値が入力されます。
- フィールドのタイプを変更すると、フィールドの値は新しいタイプのデフォルト値に変更されます。

入力エディターと検証

ロボット定義ペインの [入力値] 領域の各入力定義には、[値判定] があります。

キャレット マークをクリックすると、パラメータの種類に応じたエディターが開きます。

- 整数タイプと数値タイプの場合は、シンプルなテキスト フィールド エディターが開きます。
- テキストが非表示となっていない場合、パスワードはテキスト フィールドです。
- ブール値エディターは、次の 2 つの値を含む選択ボックスです: True または False。
- Text 値エディターは、サイズ変更可能なテキスト領域です。
- バイナリ値エディターは、値をコピー、貼り付け、クリア、ロードするボタンを持つカスタム エディターです。
  - バイナリ値が画像として認識されると、エディターに表示されます。
  - このエディターには、値のサイズと、その MIME タイプが表示されます (可能な場合)。
- Date 値エディターは、日付の入力や、カレンダー アイコンをクリックして日付の選択を行うためのフィールドです。
- Time 値エディターは、時間の入力や、時間アイコンをクリックして時間を現在の時間に設定する変更を行うためのフィールドです。
- DateTime 値エディターは、日付および時間のタイプのエディターを組み合わせたものです。

無効な定義を作成した場合、エディターでは間違ったエントリの輪郭が赤で囲まれ、赤い感嘆符が表示されます。ポインタを赤い感嘆符の上に移動すると、エラーの説明を示すポップアップ テキストが表示されます。

整数タイプ、数値タイプ、日付タイプ、および時間タイプの場合、間違った値を入力して工ディターを閉じると、値は編集前の値に自動的に戻ります。

その他のタイプでは、レコード タイプの値を変更するという特殊な場合を除き、間違った値を入力しても工ラーは発生しません。「レコード タイプを更新する」を参照してください。

リターン タイプ

[ロボット定義] ペインの [リターン タイプ] 領域を使用して、ロボットによって返される値のタイプを定義します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

ロボット を実行して、実行が戻るステップに到達すると、ロボットは、そのロボットを呼び出しているベーシック エンジン ロボットに値を返します。このような値は、ロボットの戻り値と呼ばれます。

ロボットがスタンドアロン モードで呼び出された場合、戻り値は無視されます。

ロボットのすべての [戻る] ステップでは、[出力値] の定義の数とタイプに一致する値のリストが返されます。値のリストが返されない場合は、このルールに違反するステップに対して工ラーが表示されます。

リターン タイプを追加する

1. [リターン タイプ] 領域が展開されていない場合は、下矢印をクリックすると上矢印に変わります。
2. 定義が存在しない場合は、**[リターン タイプを追加]** ボタンをクリックするか、コンテキスト メニューを右クリックして **[リターン タイプを追加]** を選択します。
3. 定義が存在する場合は、次のいずれかの方法を使用します。
   - 既存のエントリの後に追加するには、緑色のプラス記号をクリックします。
   - 既存のエントリの間に追加するには、前のリターン タイプを選択し、**[下を追加]** を選択して新しいリターン タイプを挿入します。
   - 既存のリストの先頭に追加するには、先頭にポインタを置き、コンテキスト メニューから **[リターン タイプを追加]** オプションを選択します。
4. ドロップダウン リストから、リターン タイプを選択します。

リターン タイプを削除する

データベースを削除するには、[ロボット定義] ペインの [リターン タイプ] 領域を使用します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

[リターン タイプ] 領域で、[リターン タイプ] パラメータを右クリックし、**[削除]** をクリックします。

リターン タイプを削除する際は、対応する出力値もすべての [戻る] ステップから削除する必要があります。

デバイス

[ロボット定義] ペインの [デバイス] 領域を使用して、ロボットで使用されるデバイスを定義します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

次の関連トピックを参照してください。

- Management Console に登録されている デバイス を表示する
- オートメーション デバイス マッピングの要件
- オートメーション デバイスの準備

デバイスを追加する

1. [デバイス] 領域が展開されていない場合は、下矢印をクリックすると上矢印に変わります。
2. 定義が存在しない場合は、**[デバイスの追加]** ボタンをクリックするか、コンテキスト メニューを右クリックして **[デバイスの追加]** を選択します。
3. 定義が存在する場合は、次のいずれかの方法を使用します。
   - 既存のエントリの後に追加するには、緑色のプラス記号をクリックします。
   - 既存のエントリの間に追加するには、前のデバイスを選択し、**[下を追加]** を選択して新しいデバイスを挿入します。
   - 既存のリストの先頭に追加するには、先頭にポインタを置き、コンテキスト メニューから **[デバイスの追加]** オプションを選択します。
4. ドロップダウン リストから、**[参照タイプ]** を選択します。
   - 静的デバイスとトリガー デバイスは、ロボットのスタンドアロン編集中に使用される値を持ちます。この値はプロジェクト デバイスと呼ばれ、ロボットのプロジェクトからのデバイス マッピングです。
   - 動的デバイスは、[デバイスに接続] ステップを実行することで値を取得するため、編集中のプロジェクト デバイス値は持ちません。
   - ベーシック エンジン ロボットから呼び出されたロボットの場合は、動的マッピングを静的デバイスに渡すことができます。「ロボット間で動的デバイスを渡す」を参照してください。

   [静的]、[動的]、および [トリガー] タイプの参照の詳細については、**[?]** アイコンをクリックして『Kofax RPA のヘルプ』の「オートメーション デバイスの準備」トピックに移動します。
5. デバイスに名前を付けて、Enter キーを押します。
   - ロボットで設定したデバイスの名前は、ベーシック エンジン ロボットのデバイス名と異なる場合があります。ロボット エディターでは、必要に応じてコンテキスト メニューからデバイスの名前を変更できます。ここでデバイスの名前を変更すると、デバイスが使用されるすべてのステップに名前の変更が反映されます。
   - ベーシック エンジン ロボットおよびロボットで定義したデバイスの数が一致している必要があります。
6. ロボットがスタンドアロン モードで実行される場合は、**[プロジェクト デバイス]** 下矢印をクリックして、対応するデバイスを選択します。

   プロジェクト デバイスは、Management Console からスタンドアロン モードでロボットを実行するときに使用されます。

デバイスを削除する

定義を削除するには、[ロボット定義] ペインの [デバイス] 領域を使用します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

[デバイス] 領域でデバイスを右クリックし、**[削除]** をクリックします。

ロボットからデバイスを削除すると、指定されたデバイスのみがロボットから削除されます。

ロボットをリセットするデバイスの変更

変更によってどのタイミングでリセットがトリガーされるか、およびリセットがどのように処理されるかに関する説明を以下に示します。

- デバイス定義を削除すると、ロボットはリセットされます。
- スタンドアロン モードでのリセットは、ハブ接続が更新されて、内部状態、現在のフロー ポイント、変数などがリセットされることを意味します。

  リセットは、ロボットを閉じて再度開き、実行を準備することと同じ意味を持ちます。すべての内部状態がリセットされ、Chromium 組み込みブラウザがリセットされて、デバイスへの接続が閉じてから、再確立されます。

  リセットされない唯一の状態は、ロボットがアクセスしているデバイス (ロボットがコンピューター上で開いたアプリケーションなど) です。この場合、接続は再確立されないため、デバイスの状態を手動でリセットする必要があります。
- デバイスを追加した場合、ロボットのリセットはトリガーされませんが、[レコーダー ビュー] でのストリーミングは停止します。デバイスが完全に定義された後に、[レコーダー ビュー] で [再開] を押してストリーミングを再開する必要があります。あるいは、実行を開始するか、ロボットのシングル ステップを実行してストリーミングを再開することもできます。
- プロジェクト デバイスに外部から変更が加えられた場合は、実行がリセットされます。これには、プロジェクト デバイスの削除やプロジェクト デバイスのプロパティの変更といった操作が含まれます。

データベース

ロボット がデータベース ステップでデータベースにアクセスするときに、ロボットはデータベースの名前を使用します。これは、ロボットがベーシック エンジン ロボットから呼び出されるときにデータベース ステップで使用されるマッピングです。

データベース マッピングを入力として渡すこのアプローチでは、ロボットは単一の特定のデータベースにバインドされておらず、どのデータベース マッピングがデータベースへの入力として指定されているかに応じて異なるデータベースを使用できるため、ロボットを簡単に再利用できます。

ロボットで複数のデータベースを使用するには、[ロボット定義] ペインの [データベース] 領域で複数のデータベース名を指定し、対応する数の実際のデータベース マッピングを渡します。

データベース名は、変数および入力パラメータと同じ命名規則に従いますが、名前が変数および入力パラメータ名と重複する可能性があります。ロボットを呼び出すステップで指定された実際のデータベース マッピングは異なる場合や同じである場合もありますが、ロボット内のデータベース名は異なっている必要があります。

ロボットがベーシック エンジン ロボットから呼び出された場合、データベース マッピングはベーシック エンジン ロボットの [ロボットを呼び出す] ステップによって提供されます。同じロボットに対して複数の呼び出しがある場合、異なる呼び出しによって同じロボットに異なるマッピングが渡されることがあります。ロボットがスタンドアロン モードの Management Console から呼び出された場合は、ロボットに定義された [プロジェクト データベース] が使用されます。

ロボットのデータベース名を管理するには、次の手順を実行します。

- データベース定義の追加
- データベース定義の削除

次の関連トピックも参照してください。

- Management Console でデータベースを設定するには、データベース タブを使用します。
- データベースをマッピングする方法については、データベース マッピング を参照してください。
- ロボットで使用できるデータベース ステップのリストを表示するには、データベース ステップを参照してください。

データベース定義の追加

ロボット定義ペインの [データベース] 領域を使用して、データベースを定義します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

1. [デバイス] ペインが展開されていない場合は、下矢印をクリックすると上矢印に変わります。
2. 定義が存在しない場合は、**[データベースの追加]** ボタンをクリックするか、コンテキスト メニューを右クリックして **[データベースの追加]** を選択します。
3. 定義が存在する場合は、次のいずれかの方法を使用します。
   - 既存のエントリの後に追加するには、緑色のプラス記号をクリックします。
   - 既存のエントリの間に追加するには、前のデータベースを選択し、**[下を追加]** を選択して新しいデータベースを挿入します。
   - 既存のリストの先頭に追加するには、先頭にポインタを置き、コンテキスト メニューから **[データベースの追加]** オプションを選択します。
4. データベースに名前を付け、Enter キーを押します。
5. ロボットがスタンドアロン モードで実行される場合は、**[プロジェクト データベース]** 下矢印をクリックして、対応するマッピングされたデータベースを選択します。

   フィールドが空の場合、現在マッピングされているデータベースはありません。データベース マッピングを参照してください。

   使用可能なデータベースをこのフィールドで定義すると、データベースのステップで使用できるようになります。これらのステップを実行すると、ロボットは実際のマッピングを通じてデータベースにアクセスします。

   [データベース照会] ステップ内でデータベース定義を変更すると、エディターでは実行がリセットされます。

データベース定義の削除

データベースを削除するには、[ロボット定義] ペインの [データベース] 領域を使用します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

[データベース] 領域でデータベース定義を右クリックし、**[削除]** をクリックします。

ロボットからデータベース定義を削除しても、指定したデータベースがロボットから削除されるだけです。

例外

[ロボット定義] ペインの [例外] 領域を使用して、ロボットがスローする可能性があり、呼び出し元にエラーとして返される例外のリストを表示します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

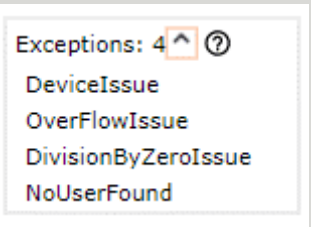
ロボットの実行中にエラーが発生すると、例外がスローされます。例外により、予期せず発生したイベントが通知されます。

ただし、例外を トライ-キャッチ ステップで処理して、その後もロボットの実行を継続できる場合もあります。

- ロボットが例外を処理するように設定されていない場合、実行は再開されず、例外は処理のためにベーシック エンジン ロボットに渡されます。
- Management Console で呼び出しを行うと、エラーがログに書き込まれます。
- Web サービスによって呼び出された場合は、返された JSON または XML にエラーが表示されます。

例: 処理できない例外

この例では、ロボットが 3 つの事前定義の例外 (`DeviceIssue`、`OverFlowIssue`、`DivisionByZeroIssue`)、および 1 つのカスタム例外 (`NoUserFound`) という 4 つの例外をスローしています。例外のリストはロボットに関する情報を提供するために表示され、編集することはできません。



ツリー モード

[ロボット定義] ペインの [ツリー モード] 領域を使用して、ロボットで使用される [ツリー モード] を定義します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

[ツリー モード] により、アプリケーションのツリーの生成方法を定義します。

[ツリー モード] の定義は、[レコーダー ビュー] でアプリケーション タブのタイトルまたはコンポーネントを右クリックして作成することができます。

ツリー モードの管理については、次のトピックを参照してください。

- アプリケーション ツリーの生成
- ツリー モードの追加
- ツリー モードの変更
- ツリー モードの削除
- ISA オプション
- Windows オプション

アプリケーション ツリーの生成

Kofax RPA は、アプリケーション ツリーを生成するいくつかの方法を提供します。デフォルトでは、Kofax RPA は、ロボットが動作しているアプリケーションのタイプ (Windows アプリケーション、ターミナル、組み込みブラウザ) を検出し、このアプリケーションのツリーを自動的に形成します。

また、[ツリー モード] は、自動化されたアプリケーションのオプションを選択する場合に役立ちます。たとえば、組み込みのブラウザで作業しているときに特定の要素のツリー モードを ISA オプション に変更して、堅牢なファインダーを作成することができます。

ツリー モードの追加

1. [ツリー モード] 領域が展開されていない場合は、下矢印をクリックすると上矢印に変わります。
2. 定義が存在しない場合は、**[ツリー モードを追加]** ボタンをクリックするか、コンテキスト メニューを右クリックして **[ツリー モードを追加]** を選択します。
3. 定義が存在する場合は、次のいずれかの方法を使用します。
   - 既存のエントリの後に追加するには、緑色のプラス記号をクリックします。
   - 既存のエントリの間に追加するには、前のツリー モードを選択し、**[下を追加]** を選択して新しいエントリを挿入します。
   - 既存のリストの先頭に追加するには、先頭にポインタを置き、コンテキスト メニューから **[ツリー モードを追加]** オプションを選択します。
4. ドロップダウン リストから、次の ファインダー エントリを入力します。
   a. **[デバイス]** ドロップダウン リストからデバイスを選択します。
   b. **[アプリケーション]** 名を入力します。
   c. **[コンポーネント]** 名を入力します。
5. **[ツリー モード]** ドロップダウン リストから、次のモードのいずれかを選択します。
   - ISA: プログラム インターフェイスのグラフィカル表示からインターフェイス要素を決定する、ユーザー インターフェイス (UI) 認識が有効になります。このモードは、オートメーション API を提供しないプログラムで使用できます。
   - [Windows]: Windows オートメーション API を使用して生成されたウィジェット ツリーにオプションを提供します。

- [ツリーなし]: アプリケーション ツリーを生成しません。ロボットがツリーを抽出しようとした際に不安定な動作をするアプリケーションには、このオプションを使用します。

ツリー モードの変更

アプリケーションおよびコンポーネント レベルでツリー モードを変更します。

- コンポーネントのツリー モードを変更すると、そのコンポーネントのネストされた要素はすべて、選択したモードを使用して自動的に設定されます。
- コンポーネントのツリー モードを変更すると、ファインダーの [コンポーネント] フィールドのアスタリスクが、選択したツリー モードに置き換わります。

  コンポーネント タグの `der_tree_mode` プロパティには、次のいずれかの値を入力します。`None`、`ISA`、`Windows`, `Auto`。
- コンポーネントのツリー モードをアプリケーションレベル モードに戻すには、**[コンポーネント]** フィールドにアスタリスクを挿入します。

ツリー モードの削除

定義を削除するには、[ロボット定義] ペインの [ツリー モード] 領域を使用します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

[ツリー モード] 領域でデバイスを右クリックし、**[削除]** をクリックします。

ISA オプション

ISA (インテリジェント スクリーン オートメーション) オプションは、UI 要素と図形を自動的に検索することで、拡張されたスクリーン認識機能を提供します。

アテンデッド オートメーションでインテリジェント スクリーン オートメーション (ISA) を使用しないでください。

ISA は、制限付きのアプリケーションまたは Citrix などのオートメーションなしの API を自動化するのに役立ちます。また、このオプションを使用して、リモート デスクトップ アプリケーションや SAP などの低速アプリケーションを自動化することもできます。

一般的に、このオプションは、アプリケーション ツリーが正しく生成されないアプリケーション、またはファインダーを作成することが困難なアプリケーションに使用します。

認識された各 UI 要素には、ファインダーで使用する複数の属性が含まれています。

共通の属性

- `der_x`, `der_y`, `der_width`,`der_height`: エレメントの座標とサイズ。
- `lt_16_5`, `rb_15_4`: エレメントの左上および右下座標を基準にして計算された数値。プログラムの状態が異なると要素の座標が変化する場合には、ファインダーが正しく動作しない可能性があります。

  たとえば、選択すると太字になるテキストの座標が、選択を解除すると変化する場合などがあります。このプロパティでは、座標の変化に伴う問題が排除されるため、信頼性の高いファインダーを作成できます。UI エレメントに `name` プロパティがある場合、アクション ステップを挿入すると、

ファインダーによってデフォルトで使用されます。それ以外の場合、ファインダーではプロパティ `lt_16_5` が使用されます。

次の表に、サポートされる UI エレメントを示します。

UI エレメント名	ウィジェット名	要素の属性	注意
ダイアログ ボックス、フレーム、ペイン、その他	`container`	該当なし	他のエレメントを含む一般的な親エレメント。 たとえば、ダイアログ ボックス全体またはこのダイアログ ボックス内のフォームをコンテナにすることができます。Kofax RPA によって認識される特別なコンテナの 1 つがテーブルです。
要素	`element`		他のエレメントを含まない一般的な UI エレメント。
アイコン	`icon`	`label`	テキストを含まないグラフィック エレメント。
テーブル	`table`		テーブルはアプリケーション ツリー内で個別の要素として表されますが、次のような制限があります。 • 結合されたセルは認識されません。 • ヘッダーは通常のセルと同じです。 • それぞれのセルは、アプリケーション ツリー内の `item` として定義されます。 • アイテムには、チェック ボックス、テキスト ボックス、その他の要素などの他の UI 要素を含めることができます。
行	`row`		テーブル ツリーの行。
アイテム	`item`		テーブル ツリーの行にあるアイテム。
テキスト ボックス	`textbox`	`label`	テキスト フィールド。
チェック ボックス	`checkbox`	`label`	チェック ボックス。
オプション ボタン。	`radiobutton`	`label`	オプションまたはラジオ ボタン。
テキストとテキスト ラベル	`linklabel`	`name` `inverted`	これは、ダイアログ ボックスのサブタイトルなどのスタンドアロンのテキスト要素、またはチェック ボックスのラベルなどの UI 要素のラベルのいずれかです。 • 名前: テキストが UI エレメント ラベルとして認識される場合、UI エレメントにはラベル名を値とする `label` プロパティが含まれます。 次の例では、`textbox` 要素に `label="Name"` プロパティが含まれています。  • `inverted`: OCR によって検出されたテキストが暗く、背景が白い場合、`inverted` は `0` に設定されます。テキストが白で、背景が暗い場合、`inverted` は `1` に設定されます。

ISA のヒント

- スクリーン フォント、背景色と前景色、テキスト サイズなど、さまざまな要素によって認識結果は異なります。認識結果を改善する場合は、明るい背景や大きなフォント サイズでの等幅の黒色フォントを使用するなど、カラー スキームを変更してみます。
- テキスト フィールドでは選択されて反転色になったテキストが正しく認識されないことがあります。認識結果を改善するには、値を抽出する前にテキスト フィールドの範囲選択を除去します (他の場所をクリックするなど)。
- 状態によって異なって認識される可能性のある動的コンテンツ フォームとテキスト フィールドを操作する場合は、ツリーの凍結ステップを使用してアプリケーション ツリーの状態を保持し、認識結果を向上させます。
- 要素から値を抽出するには、Windows オートメーション API を使用する場合と同様に、ここから値を抽出アクションを使用します。要素の値が ISA モードで認識されない場合は、画像からテキストを抽出アクションを実行してください。
- `ocr.cfg` ファイルで拡張 OCR 設定を編集します。詳細については、拡張 OCR 設定を参照してください。
- TTF フォントまたは UI スクリーンショットのいずれかを使用して、Tesseract が文字セットを認識できるようにトレーニングします。詳細については、Tesseract のトレーニングを参照してください。
- SAP アプリケーションを自動化する前に、SAP GUI オプションでシステム依存テーマを設定すると、認識結果が向上します。
- リモート デスクトップ プロトコル (RDP) を介して Windows 自動デバイスにリモートで接続する場合、解像度と色深度はコンピュータとリモート デバイス間のネットワーク接続速度に依存します。ロボットがイメージ ファインダーとIntelligent Screen Automation (ISA) を使用している場合、異なるロボットの実行で異なる接続速度が問題を引き起こす可能性があります。
  - イメージ ファインダーおよび ISA では、同一のアプリケーション状態で、ピクセル単位で同一の画像を常に受信することが重要です。
  - 「開く」ステップの RDP 接続には、明示的に指定された同じ解像度と色深度パラメータ (`g`: デスクトップ ジオメトリ (WxH) および`a`: 接続の色深度) を常に使用してください。

    例: `rdp://user1:MyPassword@MyDesktop?g=640x480&a=16`

UI 認識言語の変更または追加

デフォルトでは、ISA は、`ocr.cfg` ファイルで指定した言語を使用します。UI 認識言語を追加または変更する場合は、OCR エンジンと言語の変更トピックを参照してください。

コンピューターの ISA UI 認識言語を変更するには、次の手順を実行します。

1. テキスト エディターで、次のように `lib` の該当するディレクトリにある `isa_v1.cfg` ファイルを開きます。
   - インストールされた Desktop Automation サービスを使用する Windows ベースの自動化されたコンピューター:

     `C:\ProgramData\Kofax RPA\[ビルド番号]\lib\tessdata`.

     自動化されたコンピュータに複数のバージョンの Desktop Automation サービスがインストールされている場合は、ISA を使用するデスクトップを自動化するサービスのバージョンを確認します。『Kofax RPA Desktop Automation サービス ガイド』の [Windows] タブを参照してください。

   - 組み込みブラウザを使用するローカルの Windows ベースのコンピューター:

Kofax RPA のインストール ディレクトリの `nativelib\hub\windows-x64\[ビルド番号]\lib`。
ビルド番号は、プログラムのバージョンごとに異なります。
例：`C:\Program Files\Kofax RPA 11.5.0.0\nativelib\hub\windows-x64\622\lib`

- 組み込みブラウザを使用するローカルの Linux ベースのコンピューター:
  `nativelib/hub/linux-x64/<build number>/lib` in the Kofax RPA installation directory.
  例：`Kofax_RPA_11.5.0.0/nativelib/hub/linux-x64/533/lib`

2. `ocr_language` パラメータ内の `default` を選択した言語に置き換えます。複数の言語を使用する場合は、`ocr_language=eng+jpn` のように + 記号を使用して別の言語を追加します。
3. ファイルを保存して閉じます。
4. 変更を有効化するには、Desktop Automation サービスを再起動します。

デフォルトの認識言語に戻すには、`ocr_language` パラメータ内で `default` を指定して、前述の手順を実行します。

Tesseract の UI 認識言語を追加する

Tesseract OCR エンジンを使用する場合、ISA はデフォルトで英語 UI を自動化できます。UI 認識の言語を追加するには、次の手順を使用します。

認識言語の変更については、『Kofax RPA Desktop Automation サービス ガイド』の「デフォルトの OCR 言語の変更」を参照してください。スクリーン認識で複数の言語を同時に使用すると、ロボットの実行が遅くなり、認識結果が低下することに注意してください。

OmniPage OCR エンジンには、Kofax RPA インストールでサポートされているすべての言語が含まれています。

1. 必要な言語の `.traineddata` ファイルを GitHub の Web サイトからダウンロードします。
   たとえば、日本語のファイルは `jpn.traineddata` です。
   上記のリンクでファイルが入手できない場合は、メイン リポジトリ ページでバージョン 3.04 を検索してください。
   Tesseract バージョン 3.05.00 は Kofax RPA 11.5.0 と互換性があります。ただし、RPA は Tesseract 3.04.00 トレーニング ファイルを使用します。

   ダウンロードした `.traineddata` ファイルが、使用する `teserract.dll` ファイルのバージョンと互換性があることを確認します。互換性については、traineddata ページの下部に公開されている `README.md` ファイルを参照してください。
   間違ったトレーニング データを使用してもエラー メッセージは表示されませんが、**OCR** の結果が低下する可能性があります。

2. ダウンロードした `.traineddata` ファイルを適切なフォルダにコピーします。
   - インストールされた Desktop Automation サービスを使用する Windows ベースの自動化されたコンピューター:
     Desktop Automation サービス インストール ディレクトリの `DesktopAutomationService\lib\tessdata`。

例：`C:\Program Files\Kofax RPA DesktopAutomation 11.5.0.0\DesktopAutomationService\lib\tessdata`

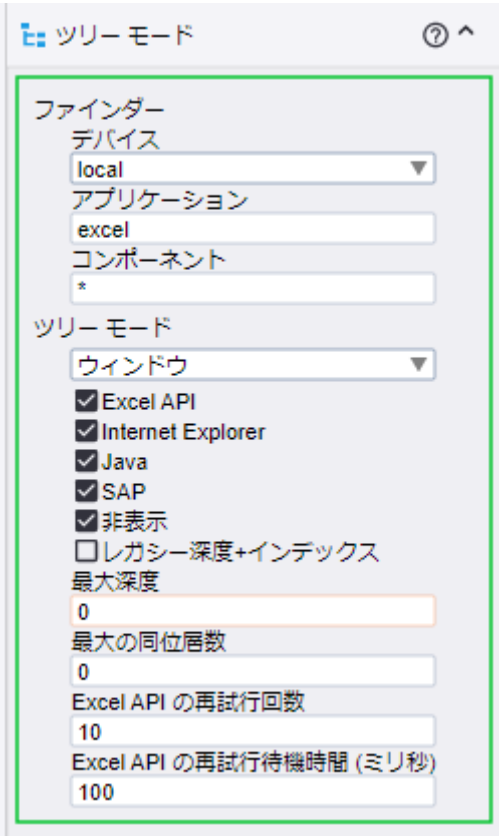
- 組み込みブラウザを使用するローカルの Windows ベースのコンピューター:
  Kofax RPA のインストール ディレクトリの `nativelib\hub\windows-x64\[ビルド番号]\lib\tessdata` にコピーします。
  ビルド番号は、プログラムのバージョンごとに異なります。
  例：`C:\Program Files\Kofax RPA 11.5.0.0\nativelib\hub\windows-x64\622\lib\tessdata`
- 組み込みブラウザを使用するローカルの Linux ベースのコンピューター:
  Kofax RPA インストール ディレクトリの `nativelib/hub/linux-x64/<build number>/lib/tessdata`。
  例：`Kofax_RPA_11.5.0.0/nativelib/hub/linux-x64/533/lib/tessdata`

3. 前述の「UI 認識言語の変更または追加」タスクの説明に従って、UI 認識言語を変更します。

Windows オプション

Windows アプリケーション用のウィジェット ツリーを生成するために、Kofax RPA は、Windows に組み込まれた UI オートメーション フレームワークを使用します。Kofax RPA は、一部のアプリケーションの拡張サポートを提供します。

Kofax RPA の拡張機能を使用してアプリケーションを操作している際に問題が発生した場合は、[Windows] ツリー モードでの選択を解除し、アプリケーションをオフにします。



Excel

スプレッドシートからセルを取得するなどの、Microsoft Excel の拡張サポートを有効にします。

次の手順で Excel を使用して作業する場合は、Excel ワークブックを [編集] モードのままにしないでください (Excel プログラム ウィンドウの左下隅に「編集」という単語が表示されます)。

リモート デバイスで Excel にテキストを入力を使用している場合、[編集] モードを終了するには、次のいずれかを実行します。

- キー プレス ステップを使用して Enter キーを押します。
  Excel の編集モードが終了し、現在のセルの下のセルが直接選択されます。
- キープレス ステップを使用して Tab キーを押します。
  編集モードが中止され、現在のセルの右のセルが選択されます。
- 別のセルをクリックします。
- キープレス ステップを使用して F2 キーを押します。

詳細については、 Microsoft のドキュメントを参照してください。

Internet Explorer

Microsoft Edge を [Internet Explorer] モードで使用すると、ドキュメント オブジェクト モデル (DOM) ツリーを取得するための拡張 Internet Explorer のサポートが有効になります。このモードでは、アプリ

ケーション ツリーでより正確な結果が得られます。また、Kofax RPA には、Web サイトにアクセスするための組み込みの Chromium Web ブラウザが用意されています。

Internet Explorer の拡張サポートを使用する場合は、次の点に注意してください。

- Internet Explorer の拡張サポートが無効になっている場合は、UI オートメーションで Internet Explorer からオブジェクト ツリーを取得します (これは、Microsoft の `Inspect.exe` を使用して取得するツリーと同じです)。
- Internet Explorer の拡張サポートが有効で、ロボットが Internet Explorer ウィンドウを検出した場合:
  - DOM を横断し、JSON の表現を作成する Java スクリプトの一部が注入されます。これは、Internet Explorer ページのコンテキストで実行されます。
  - JSON は、ページ内の「kapowDataElement」ID を持つ入力タグに配置されます。
  - ロボットは JSON コンテンツを解析し、そのコンテンツからサブツリーを作成します。

Java

状況によっては、Java Access Bridge が機能しないことがあります。このオプションの選択を解除すると、レガシー モードに切り替えることができる場合があります。また、[Java] オプションは、[Internet Explorer] モードの Microsoft Edge の Java アプレットにも影響を与えます。

SAP

SAP は他の Windows アプリケーションとは異なる方法で処理されます。SAP を使用した操作は、低速となる可能性のあるスクリプト API に依存します。このオプションをオフにすると、SAP サポートがオフになります。

一部の要素のノードはアプリケーション ツリーで使用できない場合があります。ツリー内のすべてのノードを使用できるようにするには、次の手順を実行します。

1. 製品のインストール フォルダから Desktop Automation サービスを実行しているコンピューターに `RegSAPSurrogate.reg` ファイルをダウンロードして保存します。
   ファイルは次の場所にあります。`[パス]\Kofax RPA Desktop Automation 11.5.0.0\DesktopAutomationService\bin`。
2. このファイルを実行し、警告を受け入れます。
3. サービス エージェントを再起動します。

非表示

アプリケーションのツリー全体を抽出することを指定します。デフォルトでは、このオプションが選択されています。この選択を解除すると、Kofax RPA は、多くの エレメントを含むリスト ボックスやテーブルなど、オフスクリーンとして報告される エレメントをスキップします。選択を解除すると、ツリーの抽出に必要な時間が短縮されます。

レガシーの深度とインデックス

Kofax RPA 10.2 で使用されていたアプリケーション ツリーに「深度」属性と「インデックス」属性を追加します。

深度属性およびインデックス属性を持つコンポーネント ファインダーを使用する Kofax RPA バージョン 10.2 で作成されたロボットを使用する場合は、Kofax RPA 10.3 以降のリリースで動作するロボットに [レガシー深度+インデックス] を選択します。

- [最大深度]: ビュー内のすべてのアプリケーション ウィンドウに各ノードが表示する、ネストされた要素の最大数を設定します。
- 最大の同位層数: ビュー内のすべてのアプリケーション ウィンドウに各ノードが表示する、同位層要素の最大数を設定します。

制限なしの場合は 0 を指定します。

Excel API 再試行カウント
Windows API から Excel ウィンドウにアクセスする際の再試行の回数を指定します。Excel の応答が遅い場合は、この回数を増やします。

Excel API 再試行の待ち時間
Windows API から Excel ウィンドウにアクセスする際の再試行の間隔をミリ秒単位で指定します。

変数

ロボット内で、[ロボット定義] ペインの[変数] 領域を使用して変数を定義します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

- 変数を追加する
- 変数を削除する
- ロボットの変数
- ロボット間で変数を使用および編集するためのヒント

変数を追加する

1. [変数] 領域が展開されていない場合は、下矢印をクリックすると上矢印に変わります。
2. 定義が存在しない場合は、**[変数を追加]** ボタンをクリックするか、コンテキスト メニューを右クリックして **[変数を追加]** を選択します。
3. 定義が存在する場合は、次のいずれかの方法を使用します。
   - 既存のエントリの後に追加するには、緑色のプラス記号をクリックします。
   - 既存のエントリの間に追加するには、前の変数を選択し、**[下を追加]** を選択して新しいデータベースを挿入します。
   - 既存のリストの先頭に追加するには、先頭にポインタを置き、コンテキスト メニューから **[変数を追加]** オプションを選択します。
4. ドロップダウン リストから、入力の **[タイプ]** を選択します。
5. 変数に名前を付けて、Enter キーを押します。

変数を削除する

定義を削除するには、[ロボット定義] ペインの [変数] 領域を使用します。ナビゲーションとコンテキスト メニューのオプションについては、ロボットの編集 を参照してください。

[変数] 領域で、変数を右クリックし、**[削除]** をクリックします。

ロボットの変数

ロボットの変数は、ベーシック エンジン ロボットで使用される変数とは異なります。ロボット変数には、シンプル タイプまたはレコード タイプがあります。

- 変数と同じタイプの値を変数に割り当てます。
- レコード変数のフィールドを同じタイプの値に割り当てます。

変数を使用する [レコーダー ビュー] のショートカット メニューで、レコード タイプ変数からフィールドを選択することもできます。

デフォルト値を持つシンプル タイプの変数

レコード タイプも含めて、すべてのタイプにはデフォルト値があります。レコード タイプのデフォルト値は、すべてのフィールドがそのタイプのデフォルト値を持った値です。

変数には、変数のタイプに応じたデフォルト値があります。シンプル タイプ変数のデフォルト値は次のとおりです。

- バイナリ：空
- ブール値: False
- 整数：0
- 数値：0.0
- パスワード：空
- テキスト："
- Date: 1601-01-01
- Time: 00:00 PM
- DateTime: デフォルト値は、デフォルトのタイプの Date と Time の値で構成され、デフォルトのタイム ゾーンはロボットが実行される場所に対応します。(「ロボットでの日付と時刻の処理」と「DateTime を抽出」を参照してください。)

ベーシック エンジン ロボットで設定されたカスタムのデフォルト値は、テスト値として扱われます。この値がデフォルト値としてロボットに転送されることはありません。

デフォルト値とは異なる初期値を設定するには、ロボット内のフロー ポイントを右クリックし、**[割り当てと変換]** > **[割り当て]** をクリックします。

Date、Time、DateTime を含むレコード タイプの変数

Date、Time、および DateTime タイプのフィールドを含むレコード タイプの場合、タイプ エディターでタイプを作成するときに、シンプル タイプの Date の属性に対して、派生したレコード タイプの対応するフィールドにタイプを指定できます。

Date、Time、または DateTime のレコード フィールド タイプを含むタイプを定義するには、Date タイプの属性で変数を作成し、その属性のレコード フィールド タイプを定義します。

タイプ	説明
Date	Date は、時刻やタイム ゾーンを参照せずに日付を表すタイプです。

タイプ	説明
Time	Time は、日付とタイム ゾーンを参照せずに時刻を表すタイプです。
DateTime	DateTime は、ISO-8601 カレンダー システムのタイム ゾーンで日付と時刻を表すタイプです。

これらのタイプはベーシック エンジン ロボットの Date タイプに対応し、日付と時刻を処理するためにロボットでのみ使用されるよう設計されています。

- Date タイプと Time タイプには、それぞれ日付と時刻が含まれます。
- DateTime には、日付と時間、およびタイム ゾーンが含まれます。
- シンプルな Date 変数がベーシック エンジン ロボットからロボットの Date タイプの入力パラメータに渡された場合は、入力値の日付部分のみが渡されます。

  入力パラメータのタイプが Time の場合、入力値の時刻部分のみが渡されます。
- DateTime は、ロボットの入力変数および出力パラメータとして使用できます。
  - ベーシック エンジン ロボットの Date タイプ変数は、ロボットの DateTime 入力変数に入力として渡すことができます。
  - DateTime 値は、ベーシック エンジン ロボットのシンプル タイプの Date 変数への出力として返すことができます。

ロボット間で変数を使用および編集するためのヒント

ロボット ワークフローでは、ベーシック エンジン ロボット からの入力を使用し、何らかの設定を追加した同一の変数タイプを使用できます。ベーシック エンジン ロボット内で変数を設定するには、[変数] タブを使用します。「変数の設定」を参照してください。

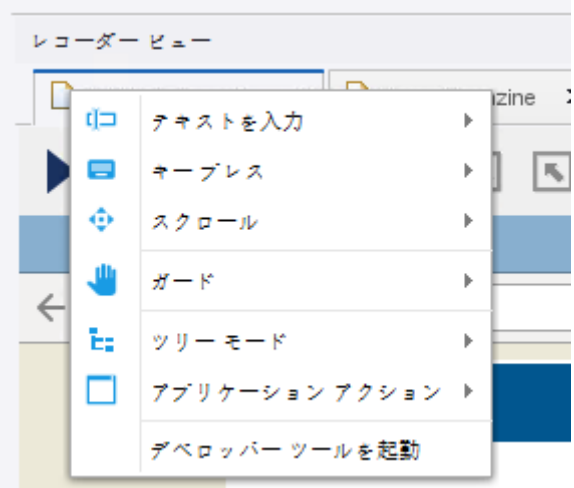
ロボットで使用される変数は、さまざまな方法で作成できます。ベーシック エンジン ロボットの [変数] タブに移動するために作業を中断することなく、[ロボット定義] ペインの [変数] 領域から新しいロボット変数を作成します。

- ロボットの編集中に新しいタイプを作成し、ロボットの実行状態を妨げることなく、そのタイプをロボットの新しい変数のタイプとして使用します。この操作を行うには、タイプ エディターで新しいタイプを作成します。その後、新しいタイプが使用可能になり、そのタイプを使用して新しい変数を追加できるようになります。
- ローカル変数を作成し、ステップのグループ化で使用します。ステップでローカル変数を使用する場合、ステップをそのローカル変数のあるグループに含めます。「変数の設定」を参照してください。
- ロボットのパスワード タイプ変数の値は、ベーシック エンジン ロボットで作成されたパスワード タイプ変数との間で転送できます。ロボット ワークフローのパスワード タイプ変数の値を手動で割り当てることはできません。
- レコード タイプはコンプレックス タイプの変数から派生する場合があります。

デベロッパー ツールを起動

Design Studio で Chromium Embedded Framework (CEF) セッションを使用する場合に、このメニュー オプションを使用して Chrome DevTools にアクセスします。

このオプションは、ロボットが CEF アプリケーション タブで開いている場合にのみ表示されます。このオプションにはアイコンがなく、他のアプリケーションのコンテキスト メニューでは省略されます。[デベロッパー ツールを起動] オプションは、アクティブな CEF タブ ウィンドウのコンテキスト メニューに表示されます。



このオプションを選択すると、Design Studio の上または下の別のウィンドウで Chrome DevTools が開きます。この動作は Windows のセキュリティ制限によるものであり、RPA 製品の機能ではありません。

- ウィンドウがすでに開いている場合は、何も起こりません。
- 複数の開発ツールを開くには、複数の CEF タブをアクティブにし、それぞれのタブから [デベロッパー ツールを起動] オプションを選択します。
- CEF タブを閉じると、対応する開発ツール ウィンドウが閉じます。

ファインダー

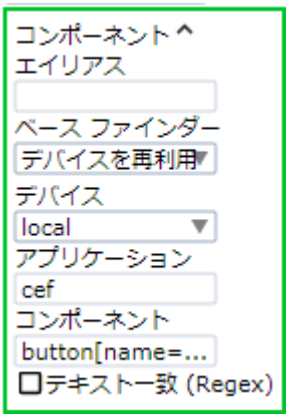
ファインダーはロボット の重要な概念です。ファインダーは、あるアクションの実行を行う工レメントを検索する方法を示します。例:

- アプリケーションを開く場合に、ファインダーによって、どのデバイスでそのアプリケーションを開くかを指定します。
- ボタンをクリックする場合は、ファインダーによって、どこをクリックするかを指定します。
- また、テキストを抽出する場合は、ファインダーによって、テキストを探す場所を指定します。

[レコーダー ビュー] で右クリックして、アクション ステップを挿入すると、ファインダーが自動的に作成されます。Design Studio は、アクションを実行しようとする工レメントを確実に見つけるためのファインダーの作成を試行します。

エディター ビューをクリックしてステップを直接挿入した場合、ファインダーは空であるため、アクションのターゲットを指定するようにファインダーを設定する必要があります。

デバイス、アプリケーション、またはコンポーネントというラベルの付いたボックスを展開して、ステップで使用したファインダーを必要に応じて確認することができます。



ファインダーのタイプ

ファインダーには主に 4 つのタイプがあります。

1. デバイス ファインダー
2. アプリケーション ファインダー
3. コンポーネント ファインダー
4. イメージ ファインダー

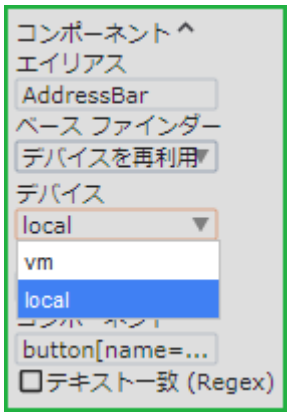
各ファインダーは下のものほど複雑で、より具体的な要素を得ることができます。

デバイス ファインダー

最も単純なファインダーは、デバイス ファインダーです。デバイス ファインダーに含まれるのは、アクセス可能なデバイスを選択するデバイス セレクターのみです。

デバイス セレクターは、ロボットがアクセスできるデバイス名を含むドロップダウン リストです。デバイスは、ベーシック エンジン ロボット の「ロボットを呼び出す」ステップの [アクション] タブにある [デバイス] にリストされます。

ドロップダウン リストには、他のファインダーのエイリアスも含まれています。いずれかのエイリアスを選択すると、このファインダー内のデバイス セレクターが再利用されます。

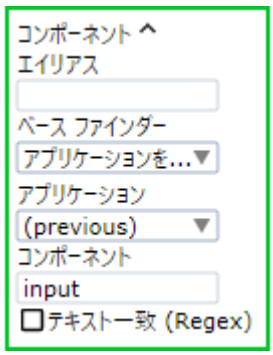


デバイス ファインダーは、開くなどでアプリケーションを開くデバイスを選択する際に使用します。

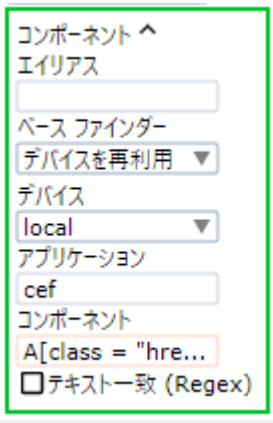
アプリケーション ファインダー

アプリケーション ファインダーは最初に特定のデバイスを選択し、次にこのデバイス上の特定のアプリケーションを選択します。アプリケーション ファインダーには、デバイス セレクターとアプリケーション セレクターが必要です。

アプリケーション ファインダーが別のアプリケーション ファインダーをベースとしている場合は、この別のアプリケーション ファインダーのデバイス セレクターとアプリケーション セレクターが再利用されます。



アプリケーション ファインダーがデバイス ファインダーをベースとしている場合は、デバイス セレクターが再利用されるため、アプリケーション セレクターを指定する必要があります。



アプリケーション セレクターは、セレクター構文で説明している CSS セレクター構文を使用します。

検出済みのアプリケーションを再利用する場合は、すべてのアプリケーションに検索を再実行するのではなく、このアプリケーションの内部ハンドルが使用されます。これにより、アプリケーションの複数の新しいインスタンスが同じ名前で起動された場合でも、アプリケーションのターゲット設定を正しく行うことができます。

ロボットが cef エンジン ベースの組み込みブラウザを使用する場合、アプリケーション ファインダーでは、1 つのブラウザ ウィンドウが開いているときは [アプリケーション] プロパティで「cef」が使用されます。ブラウザの複数のインスタンスが開いているときは、ファインダーにはアプリケーション名が含まれます (`name="kofax_mainwindow"` など)。

アプリケーション ファインダーは、キープレスでキー プレスを受信するアプリケーションを選択するために使用することができます。

一部の設定では、アプリケーション セレクターは無視されます。たとえば、デバイスにキー プレスが無差別に送信されると、その時点でフォーカスされている (前面にある) あらゆるアプリケーションがキー プレスを受け取ります。このような場合は、アクションの実行時に、必要なアプリケーションがフォーカスされていることを確認する必要があります。

コンポーネント ファインダー

最も一般的なファインダーはコンポーネント ファインダーです。コンポーネント セレクターは、特定のデバイスおよびアプリケーションを選択するのではなく、入力フィールド、ボタン、アイコン、テーブル、メニューなど、アプリケーション内の特定のコンポーネントを検索します。

コンポーネント ファインダーは次のいずれかをベースとしています。

- デバイス
- 検出されたアプリケーション
- 検出されたコンポーネント

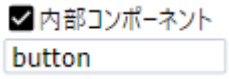
コンポーネント ファインダーがデバイスをベースとしている場合は、アプリケーション セレクターおよびコンポーネント セレクターを指定する必要があります。アプリケーションを再利用する場合は、コンポーネント セレクターのみを指定する必要があります。

コンポーネント セレクターは、セレクター構文で説明しているのと同じ CSS セレクター構文を使用します。

検出済みのコンポーネントを再利用する場合は、すべてのコンポーネントに検索を再実行するのではなく、このコンポーネントの内部ハンドルが使用されます。これにより、コンポーネントの位置が変更さ

れ、実行時間が短縮されている場合でも、コンポーネントのターゲット設定を正しく行うことができます。

コンポーネントを再利用するファインダーには、オプションの [内部コンポーネント] フィールドが追加されています。このフィールドを使用すると、テーブル内のセルや、モーダル ダイアログ ボックス内のボタンなど、すでに検出されているコンポーネント内のコンポーネントを検出できます。

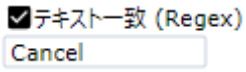


ただし、名前付きタグ (再利用するコンポーネント) が画像またはテキストの一致に基づいている場合、内部コンポーネント機能は機能しません。

内部コンポーネント セレクターも、セレクター構文で説明している CSS セレクター構文を使用します。

すべてのタイプのコンポーネント ファインダーに、オプションの [テキスト一致 (Regex)] セレクターがあります。前述のセレクターを使用して、見つかったコンポーネントの中で一致するテキスト コンテンツを検索します。含まれるコンテンツのみで変化するエレメントを識別するには、[テキスト一致 (Regex)] が便利です。たとえば、コンポーネント セレクターで OK ボタンとキャンセル ボタンが検出された場合、テキスト セレクターは「キャンセル」というテキストを識別して、的確にキャンセル ボタンを見つけ出します。テキスト セレクターは、正規表現の構文を使って記述されます。テキスト セレクターには、次の検索モードがあります。

- [下位層のないテキスト]: 選択した 1 つのノード要素内でテキストを検索します。ネストされた要素または子要素は無視されます。
- [下位層のあるテキスト]: 選択したノードとそのすべての下位層となる子要素内でテキストを検索します。
- [ツリー]: アプリケーション ツリーを XML に変換し、選択したノード内でタグと属性を含めて全文検索を実行します。



イメージ ファインダー

一部のコンポーネント ファインダーには、追加のイメージ セレクターがあります。これらのコンポーネント ファインダーもイメージ ファインダーと呼ばれます。

必要な要素がアプリケーション ツリー内に簡単に見つからなくても、その要素が視覚的な外見を持つ場合には、イメージ ファインダーを使うことができます。

イメージ ファインダーは、ファインダーごとに 3000 件の一致に制限されています。3000 個のインスタンスが見つかった後、ロボットは一致する画像の検索を停止します。

イメージ ファインダーはデフォルトでは使用しませんが、コンポーネント ファインダーを使用するあらゆるステップで、その代替となります。イメージ ファインダーの使い方:

1. 検索するグラフィック エレメントの周囲のレコーダー ビューで、長方形の選択 (紫色でマーク) をドラッグします。この長方形を変更するには、辺をドラッグするか、新たな長方形を描きます。
2. 選択範囲内を右クリックしてステップを挿入します。ステップには、選択によって作成されたイメージ ファインダーが含まれます。

3. カラーの変更などの変更が発生した場合にも、ロボットに画像を検索させるように設定するには、[ファジー画像検索] を選択します。このオプションには、次の 2 つのパラメータがあります。
   - [閾値]: 一致した画像間の差異のレベルを指定します。100 は完全一致を意味し、0 は差異が非常に大きいことを示します。値を小さくするほど、検索の厳密性が低くなります。カラーの変化のように、予想される差異がわずかである場合は、90 ~ 100 の範囲の数値を指定します。値を試して、画像が一致するようにします。
   - [近さ]: 他の一致を探すときに無視される、見つかった一致に含まれる画像のパーセンテージを指定します。「0」は、複数の一致がほぼ完全にオーバーラップする可能性があることを意味し、「100」は、一致に含まれる画像のどの部分も別の一致の中に存在しない (オーバーラップしない) ことを意味します。このフィールドには 100 を指定することをお勧めします。

   [ファジー画像検索] のパラメータにはデフォルト値がなく、[ファジー画像検索] チェックボックスをオフにした場合、指定した値は保存されません。

イメージ ファインダー内のイメージ セレクターは編集できませんが、上記のステップ 1 の説明に従って置き換えることができます。

イメージ セレクターを除去し、イメージ ファインダーを標準のコンポーネント ファインダーに変換するには、[画像] フィールドで該当する選択項目を削除します。

リモート デスクトップ プロトコル (RDP) を介して Windows 自動デバイスにリモートで接続する場合、解像度と色深度はコンピュータとリモート デバイス間のネットワーク接続パラメータに依存します。たとえば、ロボットがイメージ ファインダーとインテリジェント スクリーン オートメーション (ISA) を使用している場合、ロボットの実行ごとに接続速度が異なると問題が生じる可能性があります。イメージ ファインダーおよび ISA では、同一のアプリケーション状態で、ピクセル単位で同一の画像を常に受信することが重要です。「開く」ステップの RDP 接続には、明示的に指定された同じ解像度と色深度パラメータ (g: デスクトップ ジオメトリ (WxH) および a: 接続の色深度) を常に使用してください。例:

```
rdp://user1:MyPassword@MyDesktop?g=640x480&a=16
```

別の接続によって Desktop Automation サービス コンピュータの画面のグラフィック パラメータが変更され、イメージ ファインダー オブジェクトに影響を及ぼす可能性があることに注意してください。

セレクター構文

アプリケーション セレクターおよびコンポーネント セレクターは、CSS セレクターの構文を使用します。これにより、どの エレメントの選択が必要かを詳細に示すことができます。

セレクターの一般的なパターンは次のとおりです：

```
elementName[attributeName="attributeValue"]
```

たとえば、"Documents" というタイトルを持つ `explorer.exe` アプリケーション ウィンドウを検索するには、次のパターンを使います：

```
explorer.exe[title="Documents"]
```

セレクター パターンは、上下関係を示す不等号 (>) および上位層 - 下位層関係を示す空白スペースを用いてネストすることもできます。たとえば、ウィンドウ 要素の下位のどこかに存在する、ツールバー要素の子エレメントであるボタンを見つけるには、次のようなパターンを使用します：

ウィンドウ ツールバー > ボタン

アドバンスド セレクター構文

Kofax RPA は、多くの高度なセレクター構文をサポートしています。次の表に、サポートされている演算子とその動作の説明を示します。

パターン	意味
*	任意のエレメント
E	タイプ E のエレメント
E[foo]	"foo" 属性の E エレメント
E[foo="bar"]	"foo" 属性値が "bar" と正確に等しい E エレメント
E[foo~="bar"]	"foo" 属性値が空白で区切られた値のリストの E エレメント、その 1 つは "bar" と全く同様
E[foo^="bar"]	"foo" 属性値が文字列 "bar" で正確に始まる E エレメント

パターン	意味
E[foo$="bar"]	"foo" 属性値が文字列 "bar" で正確に終わる E エレメント
E[foo*="bar"]	"foo" 属性値にサブストリング "bar" を含む E エレメント
E:root	E エレメント (ドキュメントのルート)
E:nth-child(n)	E エレメント (親エレメントから n 番目の子エレメント)
E:nth-last-child(n)	E エレメント (親エレメントの、最後から数えて n 番目の子エレメント)
E:nth-of-type(n)	E エレメント (同タイプの n 番目の同位層)
E:nth-last-of-type(n)	E エレメント (最後のエレメントから数えて、そのタイプの n 番目の同位層エレメント)
E:first-child	E エレメント (親エレメントの最初の子エレメント)
E:last-child	E エレメント (親エレメントの最後の子エレメント)
E:first-of-type	E エレメント (同タイプの最初の同位層)
E:last-of-type	E エレメント (同タイプの最後の同位層)
E F	E エレメントの下位層 F エレメント
E > F	E エレメントの子 F エレメント
E + F	E エレメントの直後の F エレメント
E ~ F	E エレメントの後のF エレメント

複数の属性

アプリケーション ウィンドウを一意に識別するには、以下のように、セレクター内でパターンを伴った複数の属性を使用できます。

```
element[attribute1=”value1”][attribute2=”value2”][attribute3=”value3”]
```

たとえば、visible = "true" で "Save" から始まる名前を持つボタンを見つける場合は、次のようになります：

```
button[visible=”true”][name^=”Save”]
```

再利用可能なファインダー

信頼性の高いファインダーを作成することは、自動化プロセスの安定性において重要です。これは場合によっては困難で、またファインダー内の各セレクターを手動で修正する必要があります。ファインダーの仕組みに満足できたら、さまざまな状況で再利用できます。

再利用のもう一つの理由は、一貫性を保つためです。同じエレメントに対して多数のステップでアクションを実行する場合、すべてのステップで同じファインダーを使用するのが合理的です。これにより、どのようなエレメントであっても矛盾が起きないようになります。

ファインダーの再利用には、次の 3 つの方法があります：

- ファインダーをコピー＆ペーストする
- 前のファインダーを参照する

- ファインダーを名前で参照する

ファインダーをコピー＆ペーストする

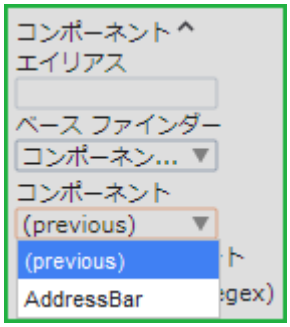
再利用において、ファインダーのコピーと貼り付けは最も脆弱な方法です。この方法は、空のファインダーでファインダーの作成を始めたくない場合には実用的です。

ファインダーをコピーするには、エディター ペインでファインダーを選択し、[**Ctrl+C**] を押すか、ツールバーのコピー ボタン をクリックします。

コピーしたファインダーを貼り付けるには、上書きするファインダーを選択し、**Ctrl + V** を押すか、ツールバーの貼り付けボタン をクリックします。

前のファインダーを参照する

前のファインダーの参照は、ファインダーを再利用するうえで最も有用かつ一般的な方法です。この参照を行ったものは、ファインダーの上から 2 番目のフィールドの再利用ドロップダウン リストに [**(直前の)**] とマークされます。この場合、現在のファインダーでは、直前に使用されたファインダーのセレクターが再利用されています。



一連のファインダーで前のファインダーを使用している場合、その各ファインダーは、前のものではない、最初のファインダーの設定を共有していることになります。チェーン内のすべてのファインダーを編集するには、最初のファインダーを編集します。

レコーダー ビューから作成された多くのステップでは、前のファインダーの参照がファインダーに自動的に含まれます。

Kofax RPA 11.3.0 以降、以前のすべてのファインダーは、ループ ステップ本体の開始時にクリアされます。ループ ステップを含むロボットが以前のバージョンの製品で作成されている場合は、以前のファインダーを名前付きファインダーに置き換えることをお勧めします。

名前付きファインダーの参照

名前付きファインダーの参照は、再利用するファインダーが前にあるファインダーではない場合に便利です。

ファインダーに名前を付けて、再利用できるようにするには、[エイリアス] フィールド (ファインダー上部の最初のフィールド) にわかりやすい名前を入力します。名前付きファインダーが 1 つ以上ある場合、後続のファインダーの再利用ドロップダウン リストにオプションとして表示されます。

ファインダーが別のファインダーを再利用する場合は、そのファインダーの設定を共有します。名前付きファインダーを編集すると、そのファインダーを参照するすべてのファインダーに影響します。

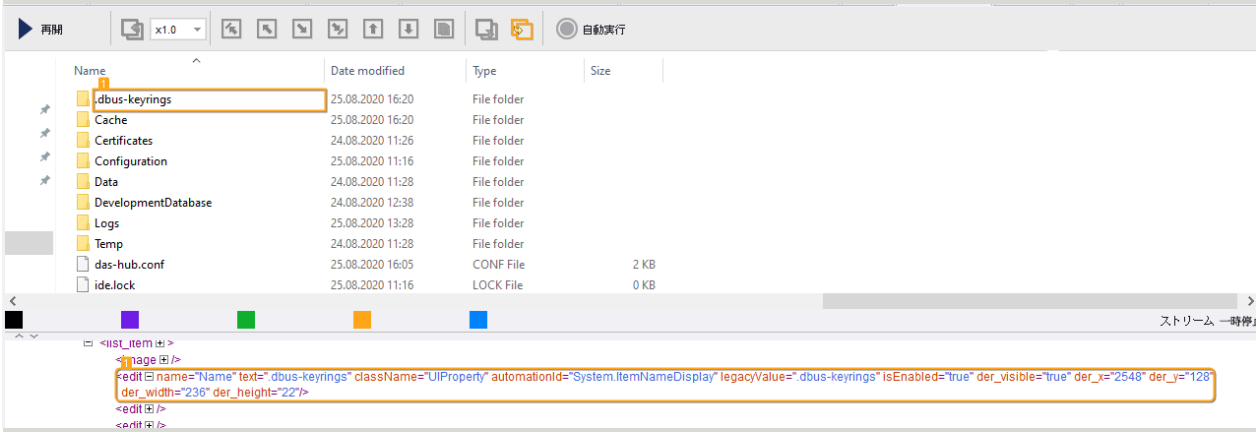
例: ファインダーの例

以下は、それぞれ説明されているように、さまざまな要素を検索するためのファインダーの例です。

ファインダーによって複数のノードが検索された場合、[見つかった次の口ケーションを表示] ボタンをクリックすると、その工レメントを順々に表示することができます。

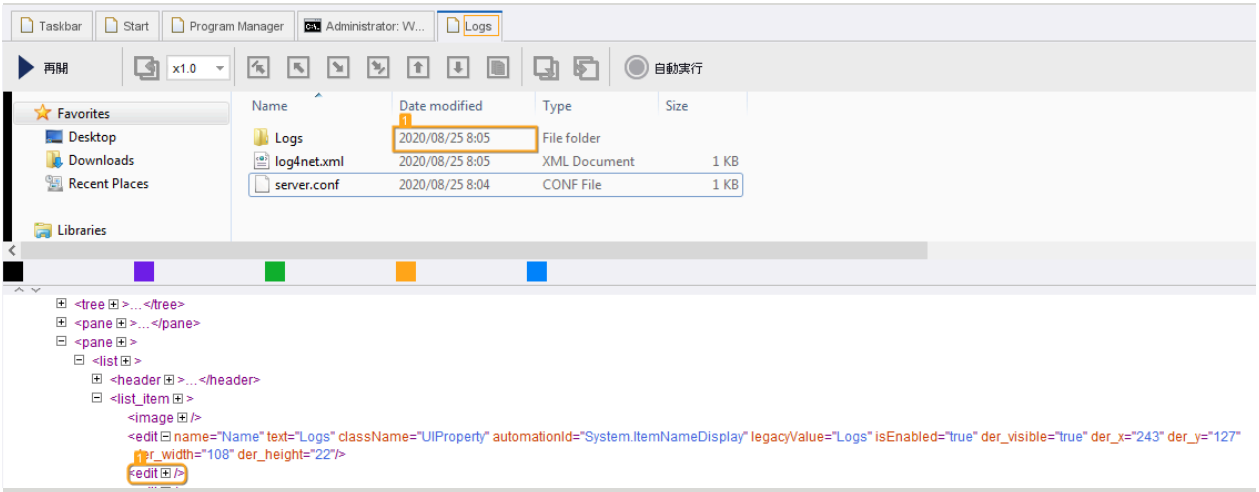
"list" の子である最初の "edit" エレメントの検索

- セレクター：":nth-of-type"
- ファインダー："list edit:nth-of-type(1)"



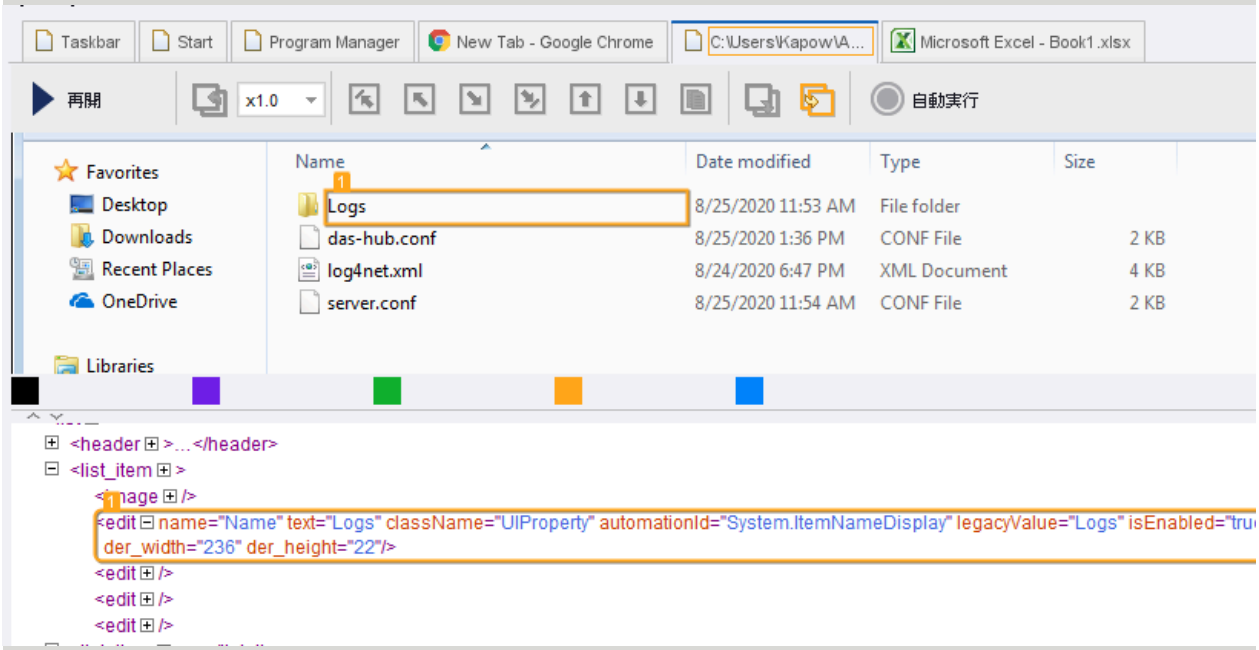
"list" の子である 2 番目の "edit" エレメントの検索

- セレクター：":nth-of-type"
- ファインダー："list edit:nth-of-type(2)"



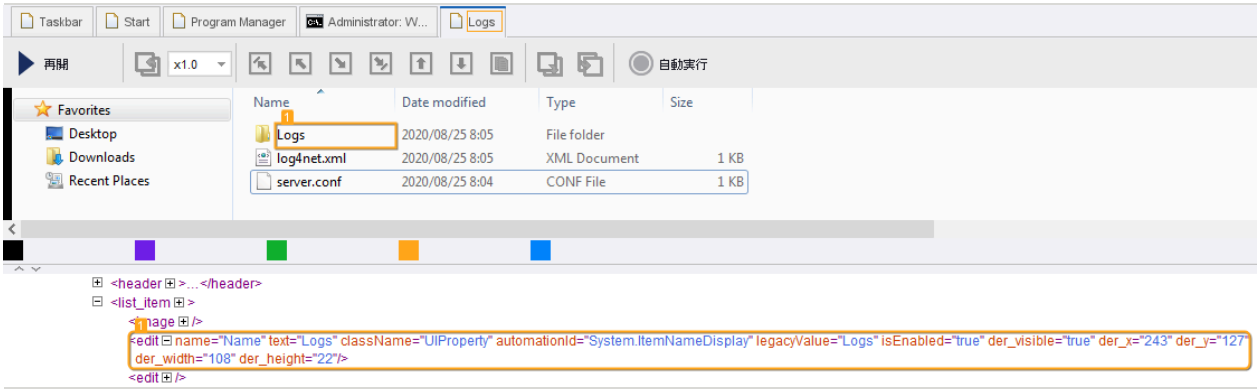
2 番目の子である "edit" エレメントの検索

- セレクター：":nth-child"
- ファインダー："list edit:nth-child(2)"



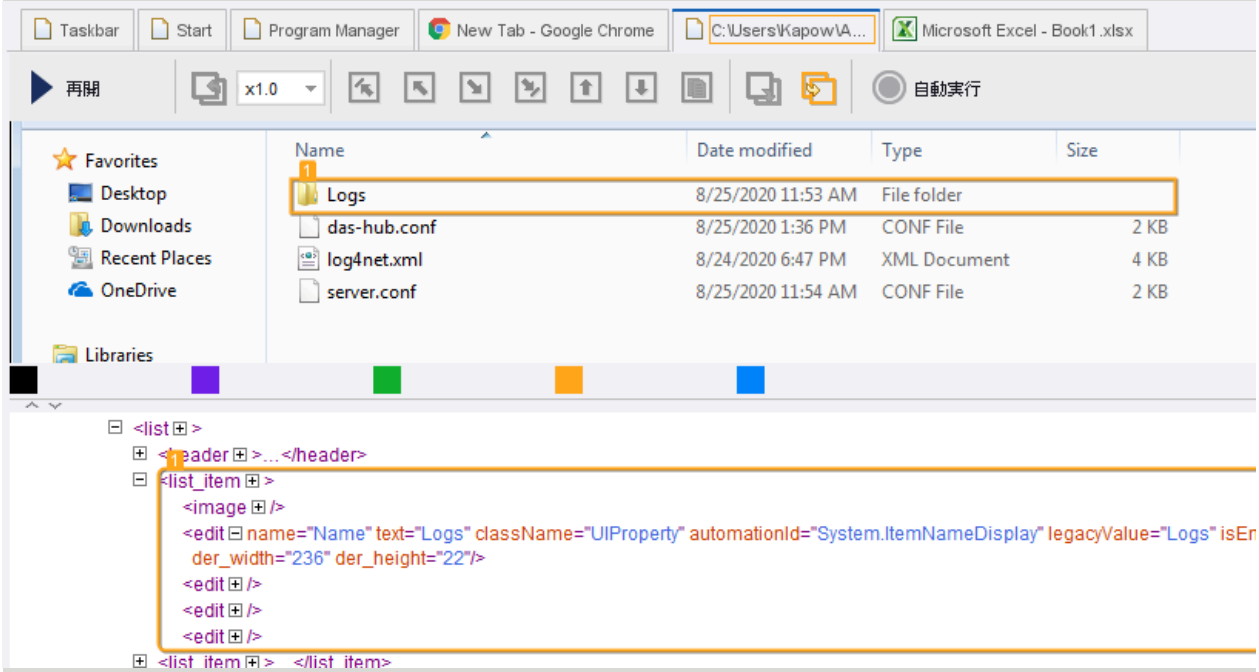
"list" の下位層である "edit" エレメントの検索

- セレクター：<space>
- ファインダー："list edit"



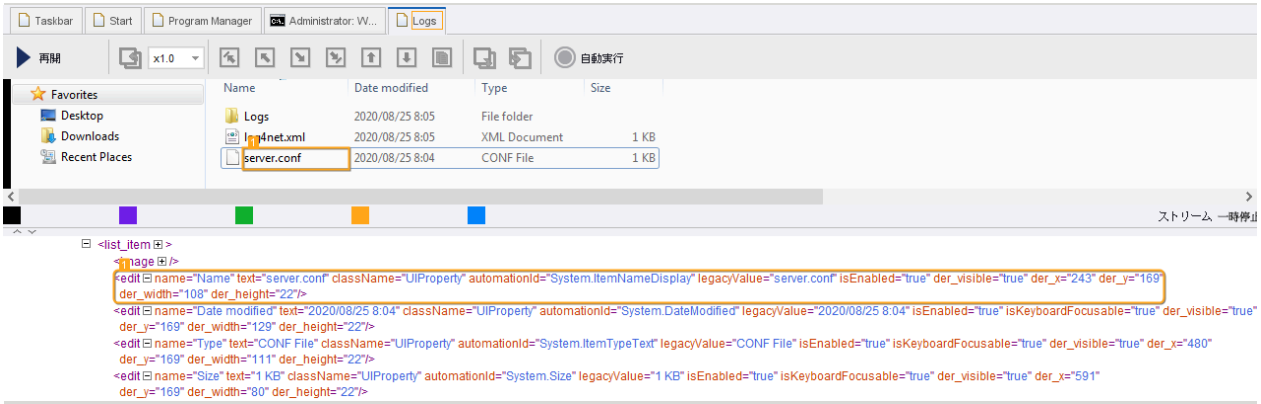
"list" の子である **"list_item"** エレメントの検索

- セレクター：>
- ファインダー："list > list_item"



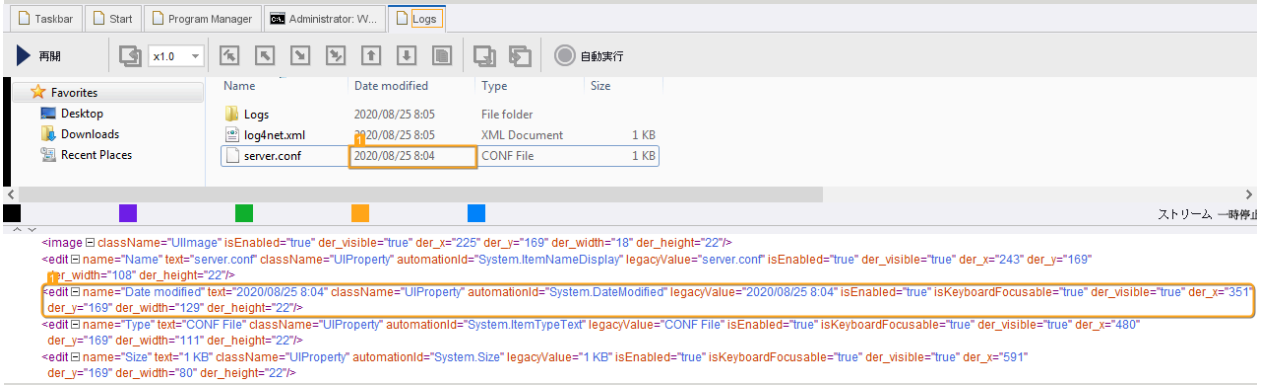
"server.conf" 値に **"text"** 属性を持つ**"edit"** エレメントの検索

- セレクター：<attribute>
- ファインダー："edit[text="server.conf"]"



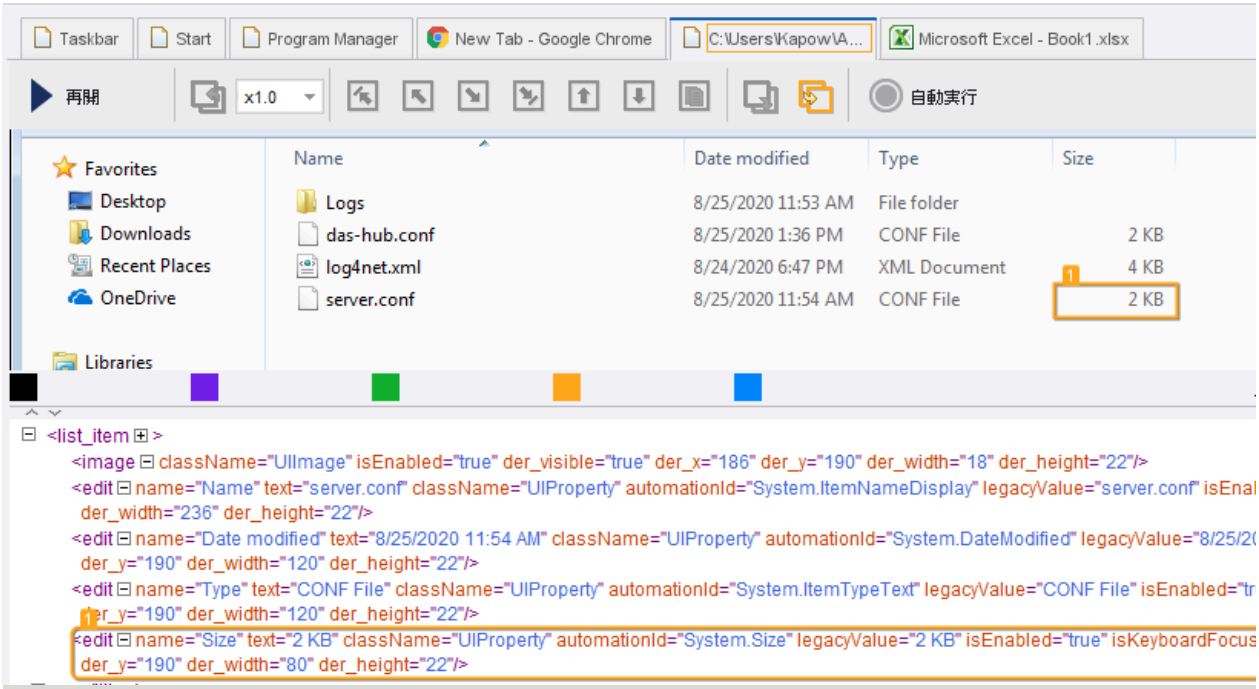
"server.conf" 値に **"text"** 属性を持つエレメントの直後にある **"edit"** エレメントの検索

- セレクター：+
- ファインダー："edit[text="server.conf"] + edit"



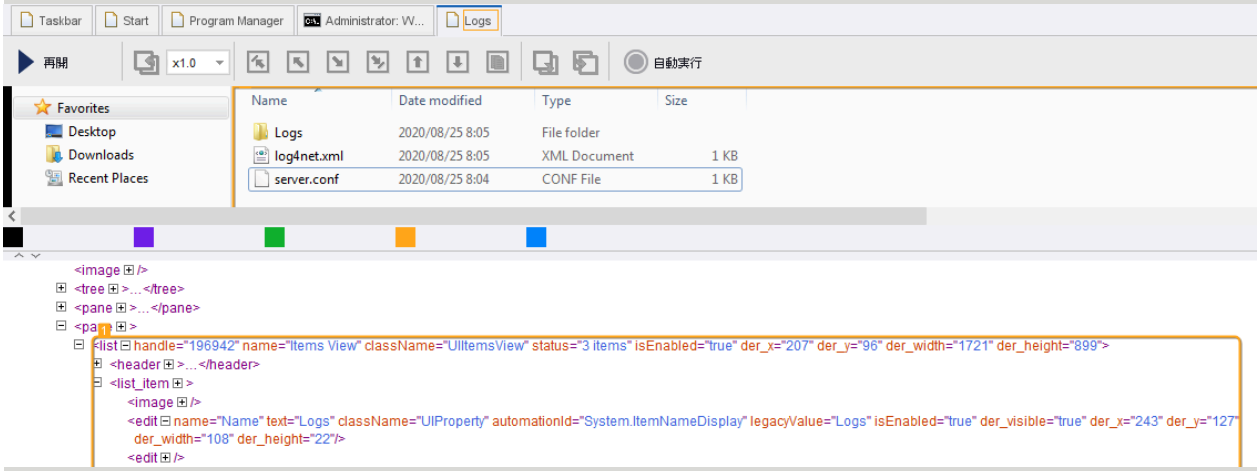
"server.conf" 値に **"text"** 属性を持つエレメントの後の任意の位置にある **"edit"** エレメントの検索

- セレクター：~
- ファインダー："edit[text="server.conf"] ~ edit"



任意の位置にある最初の **"list"** エレメントの検索

- セレクター：":first-child"
- ファインダー："list:first-child"

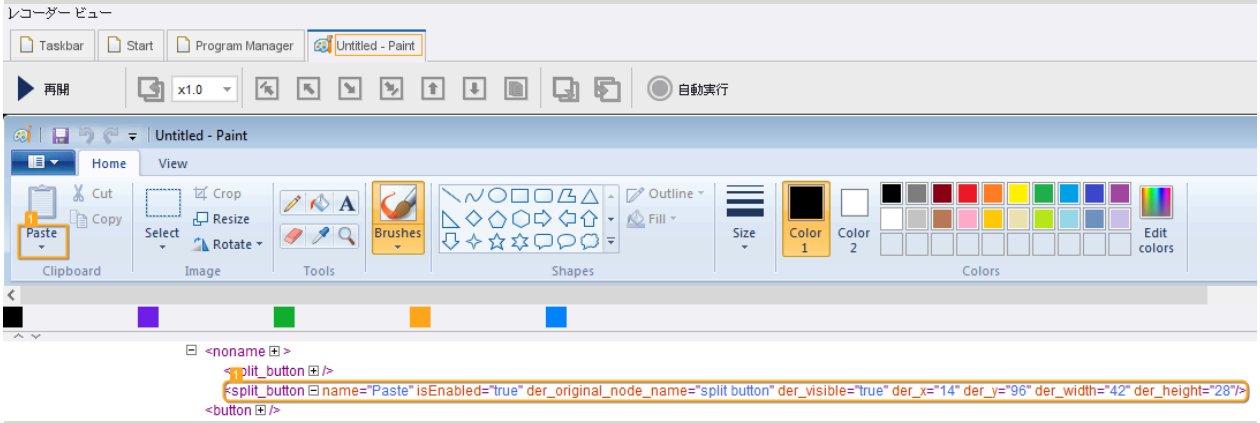


ファインダーでの複数の属性の使用
次の例では、Windows ペイント アプリケーションの貼り付けボタンをクリックするために、2 つの属性を組み合わせたファインダーを使用しています。

ペースト アイコンに同じ名前がありますが、`isEnabled` フラグはありません。Design Studio により、貼り付けドロップダウン ボタン用にこのファインダーが生成されます。

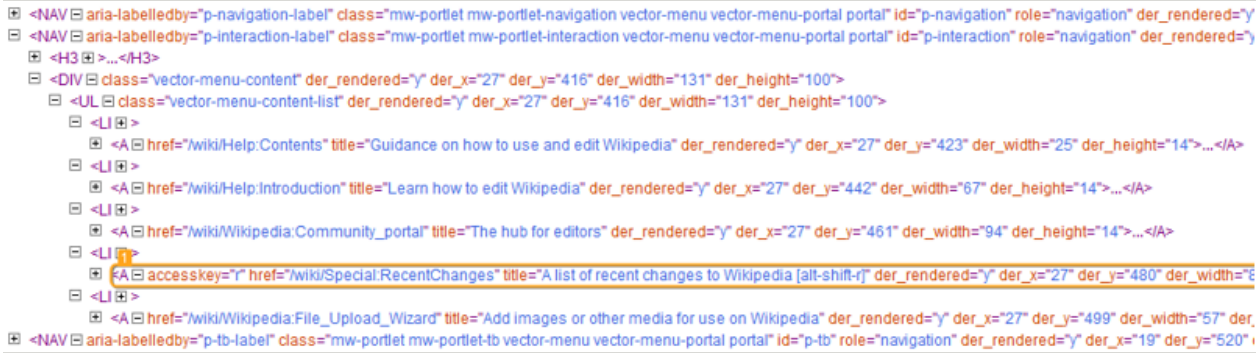
- アプリケーション：mspaint.exe

- Finder: split_button[isEnabled="true"][name="Paste"]



「p-interaction」という値の「id」属性を持つ「NAV」要素内にある、「accesskey」属性を持つ「A」要素の検索

ファインダー：NAV[id="p-interaction"] A[accesskey]



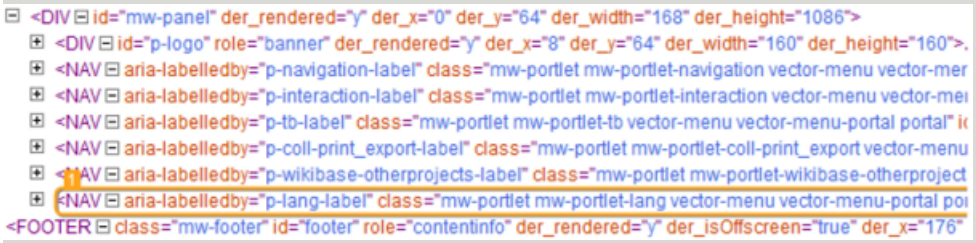
スペースで区切られた値のリスト (値のうちの 1 つが「mw-portlet-interaction」) を含む「class」属性を持つ「NAV」要素の検索

ファインダー：NAV[class~="mw-portlet-interaction"]

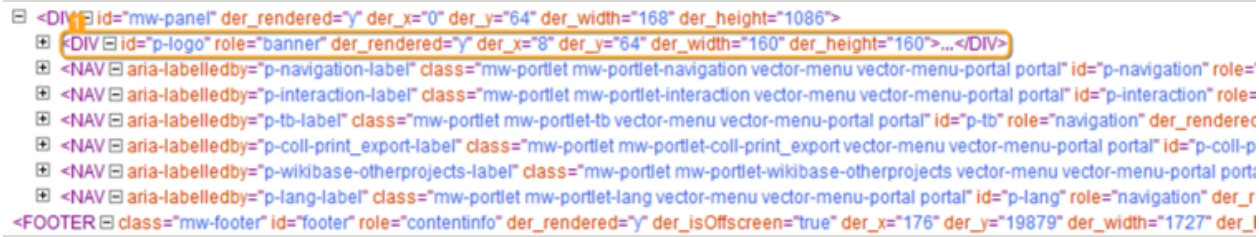
「arai-labelledby」属性の値に「coll-print」という部分文字列を含む「NAV」要素の検索

ファインダー：`NAV[aria-labelledby*="coll-print"]`

「**mw-panel**」という値の「**id**」属性を持つ「**DIV**」要素の最後の子要素の検索
ファインダー：DIV[id="mw-panel"] > *:last-child



「**mw-panel**」という値の「**id**」属性を持つ「**DIV**」要素の最後の「**DIV**」子要素の検索
ファインダー：DIV[id="mw-panel"] > DIV:last-of-type



Finder Updater Tool

Finder Updater Tool を使用して、プロジェクト内のすべてのロボットの名前を検索および置換することにより、ファインダー名を変更できます。また、このツールを使用して、ロボットを Internet Explorer から Microsoft Edge に移行できます。

Finder Updater Tool を起動するには、Kofax RPA インストール フォルダの `/bin` サブフォルダにある `ApplicationFinderUpdaterTool.exe` をダブルクリックします。

1. 必要な情報を入力します。
   - 更新するプロジェクト: [参照] をクリックして、更新するロボットを含むプロジェクトを選択します。
   - 置き換えるファインダーのタイプ: [アプリケーション ファインダー] と [コンポーネント ファインダー] のどちらかを選択します。
   - 正規表現を使用: 該当する場合は、このチェック ボックスを選択します。
   - 置き換えるファインダー: このフィールドには、デフォルトの「iexplore.exe」値が含まれています。ツールは、選択したプロジェクト内のすべてのロボットでこのファインダー名を検索します。

それ以外の場合は、置き換えたいファインダー名を入力します。

正規表現の使用を選択した場合は、「^iexp.*」などのワイルドカードを使用して必要な値を入力します。

- ファインダーの置き換え: このフィールドには、デフォルトの「msedge.exe」値が含まれています。ツールはこの値を使用して、選択したプロジェクト内のすべてのロボットのファインダー名を置き換えます。

  それ以外の場合は、カスタマイズされたファインダー置換名を入力します。
- バックアップの作成: Finder Updater Tool を使用するときは注意して、常にバックアップを作成してください。

2. [テスト] をクリックします。ツールはテスト置換を開始し、行われるすべての変更を [ログ] フィールドに表示します。
3. テスト後、[実行] をクリックして実際の置換を開始します。ツールは値を置き換え、プロジェクト フォルダに存在するライブラリ フォルダのバックアップを作成します。

ロボットのステップ

次のテーブルでは、利用可能なロボットのステップが、それらのステップが属するカテゴリに従ってグループ化されています。利用可能なステップのアルファベット順のリストを取得するには、後続のトピックを参照してください。

また、アプリケーション アクションとコンポーネント アクションを参照してください。

カテゴリー	利用可能なステップ
割り当てと変換	割り当て 値の変換
条件と制御	条件 トライ-キャッチ スロー ガード チョイス グループ リターン
ループ	ループ 条件付きループ 要素の繰り返し データベース照会 電子メールごとに ディレクトリの反復 ブレイク コンテニュー

カテゴリー	利用可能なステップ
アプリケーション	ブラウザ Windows Excel ターミナル Document Transformation PDF 電子メール 電子メールごとに ディレクトリの反復
データベース	データベース照会 データベース データ登録 データベース データ抽出 データベースから削除 キーの計算 SQL 実行
ファイル システム	ファイルの読み込み ファイル出力 ファイル システム アクション ディレクトリの反復
出力値	出力値 ファイル出力 ログ出力
統合	KTA クラウド AI カスタム アクション
リモート デバイス	デバイスに接続 デバイスからの切断 リモート デバイス アクション RDP ログイン トリガー チョイス 通知 クリップボードから抽出 クリップボードへ割り当て
抽出	ツリーを XML として抽出 画像抽出 画像からテキスト抽出 値を抽出

カテゴリー	利用可能なステップ
マウスとキーボード	キープレス テキストを入力 マウス プレス マウス移動 スクロール クリック
その他	ツリーの凍結

アプリケーション アクションとコンポーネント アクション

Kofax RPA では、現在の状態の特定のリモート アプリケーションまたはローカル アプリケーションで使用可能なアクション ステップに簡単にアクセスすることができます。アプリケーション ステップとは、レコーダー ビューのアプリケーション タブを右クリックして使用できるアクションです。コンポーネント ステップとは、レコーダー ビューまたはツリー ビューの要素を右クリックして使用できるアクションです。

- リモート デスクトップ アプリケーションの場合、選択したアプリケーションにフォーカスするためのフォーカス アプリケーション アクションが追加されています。たとえば、このステップを使用して、最小化されたアプリケーションを前面に移動できます。
- ローカルで実行されているアプリケーションの場合、アプリケーション アクションとコンポーネント アクションのリストの内容は異なります。アクションの説明は、それぞれのアクションが適用されるアプリケーションまたはステップと同じトピックに記載されています。必要に応じて、次の表を使用して適切なトピックを確認してください。

ローカルで実行されているアプリケーション	アクション リストのリンク	
Chromium ブラウザ	アプリケーション アクション	コンポーネント アクション
データベース	-	コンポーネント アクション
Document Transformation ブラウザ	アプリケーション アクション	コンポーネント アクション
電子メール	アプリケーション アクションは、プロパティの説明の一部として説明されています。	コンポーネント アクションは、プロパティの説明の一部として説明されています。
Excel	アプリケーション アクション	コンポーネント アクション
PDF	アプリケーション アクション	コンポーネント アクション
ターミナル	アプリケーション アクション (ターミナルのタイプごとに異なるアプリケーション アクション セットを使用できます)	-
Windows	アプリケーション アクション	コンポーネント アクション

割り当て

このステップでは、値を変数に割り当てます。[エクスプレッション] フィールドの値は変数のタイプと一致する必要があります。

このステップは、値を右クリックして [ステップを挿入] > [割り当て先] をクリックすることにより、ロボット ワークフローまたは [状態] ペインから直接追加できます。

プロパティ

名前
ステップの名前。

変数
変数の名前。

エクスプレッション
変数値。このフィールドではエクスプレッションとその他の変数を使用できます。詳細については、エクスプレッションを参照してください。

クリップボードへ割り当て

このステップでは、値をオートメーション デバイスのクリップボードに割り当てます。

プロパティ

名前
ステップの名前。

デバイス
オートメーション デバイスの名前を選択します。

コンテンツ
クリップボードにコピーする値を指定します。このフィールドで変数名を指定できます。

ブラウザ

このステップを使用して、組み込みブラウザの 1 つで Web サイトを開きます。詳細については、Web サイトのアクセス を参照してください。

また、Chromium 組み込みブラウザで Cookie を使用する場合にも参照ステップを使用します。詳細については、「Chromium 組み込みブラウザでの Cookie の管理」を参照してください。

プロパティ

ブラウザ
使用するブラウザ エンジンを選択します。Kofax RPA は、Chromium ベースのブラウザを提供します。

アクション
ブラウザが実行する必要があるアクションを選択します。

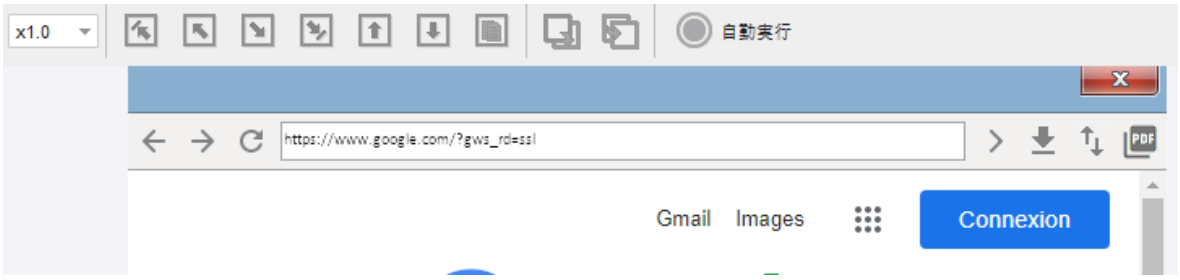
ページ読込
- アプリケーション名

アプリケーションで名前属性を設定します。この必須パラメータによって、ページの読み込み時にアプリケーションの `name` タグに値が追加されます。`name` タグは、アプリケーションを識別でき、堅牢で信頼性の高いファインダーを作成するために役立ちます。

エラーを回避するために、ロボット内のアプリケーションにはそれぞれ異なるアプリケーション名を割り当てます。

開かれたページにあるリンクをクリックすることにより、ロボットによって新しいブラウザ ウィンドウにページがロードされると、新しく開いたブラウザ アプリケーションの `name` タグには、たとえば `name="mainpage(2)"` のように、親アプリケーションの名前と括弧で囲まれた数字が含まれます。アプリケーション名を設定アプリケーション アクションを使用して、開いているブラウザ アプリケーションに名前を割り当てることができます。

- **URL**
  開く Web サイトのパスを指定します。パスでスラッシュを使用します。例:
  `https://www.kofax.com`
- [画面サイズ]
  このオプションを選択して、ブラウザ ウィンドウの幅と高さをピクセル単位で指定します。デフォルトのウィンドウ サイズは 1920x1200 で、ツールバーは含まれません。
  例: デフォルトの画面サイズを使用した場合、ブラウザのページがページのサイズに収まらないため、スクロールバーを使用する必要があります。レコーダー ビューのブラウザ ページのサイズを縮小し、割り当てられたスペースの中央に配置するには、[画面サイズ] を 640x480 に設定します。
- [ユーザー エージェント]
  このオプションを選択すると、HTTP リクエストの「User-Agent」ヘッダーで組み込みブラウザが送信するものを指定できます。「User-Agent」文字列には、Web ブラウザ名、オペレーティング システム、デバイス タイプなどに関する情報が含まれます。
- [言語リストを承認]
  このオプションが選択されていない場合、ページは Web サイトのデフォルト言語でロードされます。選択した場合、指定された言語コードに応じた言語でのページのロードを試みます。その言語が Web サイトで使用されていない場合は、Web サイトのデフォルトの言語が使用されます。
  デフォルト コード: en-US
  例: フランス語で Web サイトを表示するには、このオプションを fr-FR に設定します。



- [ロード状態を無視]
  ブラウザでこのオプションを選択すると、広告バナーの読み込みなどのバックグラウンドで行われる読み込み要求が無視されます。
  デフォルトでは、CEF ブラウザはそのロード状態を分析し、ブラウザが Web ページをアクティブにロードしている場合、または別の場所にリダイレクトしている場合は、ロケーション ベースの

ガードの実行を許可しません。ブラウザは、Web ページがロードされ、ブラウザが非ロード状態に切り替わるまで待機します。

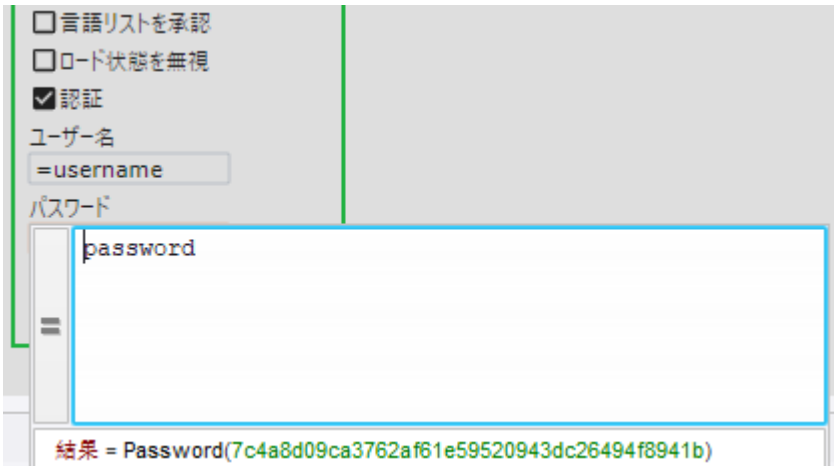
[ロード状態を無視] プロパティの設定により、ユーザーは、ブラウザが使用する URL または部分文字列を指定して、ブラウザが Web ページを完全にロードする前に、該当するロケーションのガード、該当しないロケーションのガード、ロケーションが消失した場合のガード、およびツリーの変更停止のガードが実行可能かどうかを決定できます。

ユーザーは、コンマ区切りの URL 部分を指定できます。Web ページのロード中にブラウザで Web ページの URL とユーザーが指定した文字列との一致が検出された場合に、開くアクションまたはロード状態を変更するアクション (ボタンをクリックする、またはマウスを動かすといったアクション) の後にいずれかのガードを実行する必要がある場合は、ロケーション ベースのガードを実行します。

- [認証]

  Web ページ認証用にユーザー クレデンシャルを指定します。ページが読み込まれる前にポップアップ ダイアログ ボックスに資格情報を入力する必要がある認証です。Web ページで認証を必要としない場合、これらの設定は無視されます。

  例: Web サイトにクレデンシャルを指定するには、ベーシック エンジン ロボット に `username` および `password` 変数を追加し、その値を指定します。次に、これらの変数を入力値として追加して、ロボット に渡します。ロボット内で、これらの変数を [入力値] フィールドに追加してバインドします。その後、[認証] チェック ボックスをオンにします。[ユーザー名] フィールドと [パスワード] フィールドに、それぞれ `username` 変数と `password` 変数を追加します。



  これにより、Web ページがロードされたときにユーザー入力なしで認証がバイパスされるようになります。

- セキュリティ
  - ロード エラーを無視: デフォルトでは、ロボットは提示された SSL 証明書を検証するように設定されています。発生する可能性のある証明書エラーを無視するには、このオプションを選択します。
  - [クライアント証明書]: Web サイトでクライアント (ユーザー) の認証にクライアント証明書が使用されている場合は、このオプションを選択して、クライアント証明書を Web サイト サーバーに渡すことができます。

    [証明書ストレージ] で、証明書ストレージ ファイルを含むバイナリ変数を指定します。[ストレージ パスワード] で、証明書ストレージのパスワードを含むパスワード変数を指定します。
- [**PDF** 設定]

Web ページを PDF 形式で保存するための設定を指定します。

- 背景グラフィック : 選択すると、Web ページの背景グラフィックを保存します。
- [ヘッダーとフッター] : 選択すると、PDF ドキュメントにヘッダーとフッターを保存します。
- [スケール] : スケール レベルをパーセンテージで指定します。
- [用紙サイズ] : PDF ドキュメントの用紙サイズを選択します。
- [レイアウト] : ドキュメントの横向きまたは縦向きレイアウトを選択します。
- [マージン] : ドキュメントのページ マージンを指定します。[カスタム] を選択した場合、各マージンをピクセルで入力します。

> Web ページを PDF として保存する場合は、ブラウザ ウィンドウの右側にある [**PDF**] ボタンを使用します。ボタンが画面に表示されていない場合は、右端までスクロールする必要が生じる場合があります。

- ヘッダー

  このオプションのパラメータは、HTTP 要求で送信されるカスタムのページ ヘッダーを作成するために役立ちます。ヘッダーは、コロンで区切られた `name:value` のペアで指定する必要があります。`name` パラメータの後のコロン以下を省略することで、値なしでヘッダーを指定することができます。複数のヘッダーを指定するには、各ヘッダーを別々の行に配置します。次の例では、2 つのペアと 2 つの空のヘッダーを設定します。

```
header0:value0
header2:
header3:value3
header4
```

> 以前のバージョンの Kofax RPA で作成されたロボットは、Kofax RPA バージョン 11.5.0 以降で開くと、このパラメータを使用するように自動的に更新されます。

ページ生成

Web ページを作成し、新しいブラウザ タブで開くステップを追加します。たとえば、このステップを使用して、ロボットについての説明が記載されたページを作成することができます。

- アプリケーション名

  アプリケーションを識別するための名前を指定します。この必須パラメータによって、ページの読み込み時にアプリケーションの `name` タグに値が追加されます。アプリケーション名の追加は、堅牢で信頼性の高いファインダーを作成するために役立ちます。

> エラーを回避するために、ロボット内のアプリケーションにはそれぞれ異なるアプリケーション名を割り当てます。

- [コンテンツ]: 新しいページの HTML コードを指定します。
  例: `<html><body><p>このページには、ロボットの情報が表示されます</p></body></html>`
- ページ **URL**新しいページの URL を指定します。
  例: http://www.createdpage.com

[ダウンロードを待機]

このアクションを使用すると、ダウンロードされるファイルの待機期間を有効にすることができます。これは、Web ブラウザでファイルを表示できるのと同じように、ファイルのダウンロードが完了したかどうか、またはアクティブなダウンロードの数を知る必要がある場合に役立ちます。

最大待機時間は 300 秒 (5 分) です。待機時間に達しても、ダウンロードが完了していない場合は、依然としてアクティブなダウンロードの数がステップによって返されます。この場合、ダウンロードを続行するか (ダウンロード サイクルのままにするか)、終了するかを決定できます。

また、ダウンロードに時間がかかりすぎる場合やダウンロードが不要になった場合は、明示的にダウンロードをキャンセルできます。

- 設定
  - [アクティブなダウンロード]: 待機期間を有効にするには、このオプションを選択します。
  - [ダウンロードをキャンセル]: アクティブなダウンロードをすべてキャンセルするには、このオプションを選択します。[ブラウザ/ダウンロードを待機] ステップ セットを「アクティブなダウンロード」に設定した後、新しい [ブラウザ/ダウンロードを待機] ステップ セットを追加して、すべてのアクティブなダウンロードをキャンセルします。
- [秒]: 待機時間を指定します。デフォルトは 60 秒です。また、最小および最大待機時間はそれぞれ 0 秒と 300 秒です。この時間に達すると、ステップによって、以下の [アクティブなダウンロードの数] で指定した変数でダウンロードのステータスが返されます。ダウンロード プロセスがまだアクティブである場合、変数は進行中のダウンロードの数を示します。すべてのダウンロードが完了した場合、この数は 0 になります。
- [**RFS** へのアップロードを待機]: ファイルがロボット ファイル システムに正常に保存されたことを示す通知が届くまで待機するには、このオプションを選択します。システムが通知を受信すると、ダウンロードは停止し、ロボットは次のステップの実行を継続します。Robot File System にアップロードまたはダウンロードできるファイルの最大サイズは、使用可能なメモリによって制限されます。
- [結果]:

  [アクティブなダウンロードの数]: アクティブなダウンロードの数に関する情報を含む整数変数を指定します。

バンドル

このステップは、ローカル Desktop Automation モードでの使用を目的としています。Design Studio および Desktop Automation サービスは、シングル ユーザー モードに設定する必要があります。

バンドル ステップは、コンテキスト メニューやドロップダウン メニューなど、ポインタを削除すると消えるアプリケーション要素を自動化する必要がある場合に使用します。バンドル ステップは、自動化されたアプリケーション上で実行するいくつかのステップを接続し、最初のステップから順番に実行されるシーケンスに変換します。

既存のステップをバンドル ステップにラップするには、エディターで、消える要素が使用されているステップを選択し、グループを右クリックして [バンドル ステップで囲む] をクリックします。また、バンドル ステップをワークフロー内に直接挿入し、必要なステップを追加することもできます。

アクション ステップをバンドル ステップに追加するには、ステップ内のフロー ポイントを右クリックして選択を行います。

一部のステップはバンドル ステップ内部では使用できませんが、バンドル ステップの前または後のロボットに追加できます。

キーの計算

このステップでは変数値のキーを計算することができます。値はユーザー定義のコンプレックス タイプ、つまり .type ファイルで指定されたタイプである必要があります。値のキーを把握しておくことは、値を別の値に (たとえば、別のテーブルのセカンダリ キーとして) リンクさせる必要がある場合、またはファイルに保存されたデータにリンクさせる必要がある場合に役立つことがあります。

プロパティ

変数
キーを計算するユーザー定義のコンプレックス タイプ変数を選択します。

キー (出力値)
計算されたキーが格納される文字列タイプ変数。

クリック

クリック ステップは、ロボットで最も一般的に使用されるアクションの 1 つです (ターミナルを自動化していない場合)。クリック ステップを使用することで、ロボットは、プログラムの開始と終了、プログラム インターフェイスの使用、テキストの選択、およびユーザーがポインティング デバイスで実行可能なその他の多くのアクションの実行を行うことができます。クリック ステップは、1 つのステップでマウス ポインタを必要な場所に移動し、クリックします。ドラッグアンドドロップ操作には、マウス プレス ステップおよびマウス移動ステップを使用します。

さらに、自動化されたコンピュータでハードウェアのキーボードとマウスをシミュレートすることもできます。詳細については、『Kofax RPA インストール ガイド』の「Desktop Automation サービスのインストール」を参照してください。

プロパティ

コンポーネント
クリック ステップのコンポーネント ファインダー。

ボタン
ポインティング デバイスの [標準ボタン] または [計算されたボタン] を選択します。

- [標準ボタン]: 左、中央、右。
- 計算されたボタン: このオプションを選択して、ロボットの実行中にマウス ボタンのクリックを判定します。[エクスプレッション] フィールドに、仮想キー コードまたはスペースで区切られた入力仕様のリストを指定します。エクスプレッションの結果は、マウス ボタンの 1 つを表す 0 から 2 までの値である必要があります。この機能は、Windows オペレーティング システムでのみサポートされます。仮想キー コードのリストについては、Microsoft のドキュメントを参照してください。

例
左マウス ボタンに「0」、右マウス ボタンに「1」、中央マウス ボタンに「2」を使用します。

カウント
アクションを実行する回数を指定します。たとえば、ダブルクリックの場合は 2 を指定します。

修飾子
キー修飾子を選択します。

- [固定キー修飾子]: Shift、Ctrl、Alt の 3 つのスタンダードなキー修飾子が含まれます。
- **[**計算されたキー修飾子**]** (Desktop Automation サービスのみ): このオプションを選択する場合は、修飾子に仮想キー コードの記号定数名を指定します。
  表示されるテキスト ボックスには、Shift、Ctrl、Alt のキー コードのみが入力可能です。たとえば、VK_LSHIFT キー コードは左 Shift キーを、VK_RCONTROL は右 Ctrl キーを、VK_MENU は Alt キーを表します。全キー コードのリストについては、Microsoft のドキュメントを参照してください。

オフセット

- なし: 座標オフセットを使用せず、選択した エレメントの中央に移動します。以下に対応します。
  x=0、y=0 を中央に相対的に設定
- 使用: 次のオプションを使用して、オフセットをピクセルで指定します。
  次を基準
  このオプションは、オフセットを計算する起点を指定します。
  - 左上: ウィンドウ、または x=0 および y=0 で選択された エレメントの左上隅。
  - 上: ウィンドウ、または y=0 で選択された エレメントの上枠の中央。
  - 右上: ウィンドウ、または y=0 で選択された エレメントの右上隅。
  - 左: ウィンドウ、または x=0 で選択された エレメントの左枠の中央。
  - 中央: ウィンドウまたは選択された エレメントの中央。
  - 右: ウィンドウ、または選択された エレメントの右枠の中央。
  - 左下: ウィンドウ、または x=0 で選択された エレメントの左下隅。
  - 下: ウィンドウ、または選択された エレメントの下枠の中央。
  - 右下: ウィンドウ、または選択された エレメントの右下隅。

  X
  選択された起点を基準とする水平オフセットを指定します。正数はマウスを起点の右に移動します。負数はマウスを起点の左に移動します。

  Y
  選択された起点を基準とする垂直オフセットを指定します。正数はマウスを起点から下に移動します。負数はマウスを起点から上に移動します。

クラウド AI

クラウド AI ステップを使用すると、アマゾン ウェブ サービス、Microsoft Azure、および Google Cloud という 3 つのクラウド サービス プロバイダの AI 機能に簡単にアクセスできます。この機能は 3 つのプロバイダ間でわずかに異なりますが、各プロバイダで次のことを実行できます。

- テキストを別の言語に翻訳する
- 画像を分析し、その中のオブジェクトを特定する

ステップの機能は、選択したクラウド サービス プロバイダによって異なります。クラウド サービスの機能が Amazon、Microsoft、または Google によって変更された場合、クラウド AI ステップは機能を停止することがあります。

ステップは、ステップを実行する API キーに依存します。これらのキーは Kofax RPA の一部ではなく、クラウド プロバイダから取得する必要があります。

画像の分析

3 つのプロバイダはどれも画像の分析が可能で、結果とともに JSON 形式の文字列を返します。

サポートされている画像フォーマットと要件については、各プロバイダのドキュメントを参照してください。

アマゾン ウェブ サービス

Amazon には、「アクセス キー ID」という名前の 2 つの API キーと、対応する「シークレット アクセス キー」が必要です。この API キー ペアには、Amazon Translate サービスへのアクセス権を付与する必要があります。詳細については、Amazon AWS のドキュメントを参照してください。

Amazon サービスを使用する前に、リージョンを指定してください。リージョンはサービスに対して、リクエストを (地理的に) 処理する場所を指示します。ロボットが実行されている場所に近いリージョンを選択することをお勧めします。サポートされているリージョンのリストは、各サービスのドキュメントに記載されています。Amazon はこのリストを必要に応じて変更することができます。

Amazon サービスには、3 つの異なる画像分析方法があります。

- ラベルの検出: さまざまな種類のオブジェクトを検出します。
- 保護ギアの検出: ヘルメットなどの保護ギアを検出します。
- 顔の検出: 顔とその特徴を検出します。

詳細については、アマゾン ウェブ サービスのドキュメントを参照してください。

Microsoft Azure

Azure を画像分析に使用するには、エンドポイントと API キーが必要です。API タイプが「Computer Vision」の「Cognitive Services」サービスを作成して、エンドポイントと API キーを取得します。

API キーは、画像の分析とテキストの翻訳で異なります。
Azure は、要求元に最も近いリージョンにリクエストを自動的に転送します。

画像分析を設定して、計算済みの特定の詳細を返すようにすることができます。これらの設定は、ステップのオプションでオンとオフを切り替えることができます。

詳細については、Microsoft Azure のドキュメントを参照してください。

Google Cloud

Google Cloud を画像分析に使用するには、「Cloud Vision API」へのアクセスを有効にして、API キーを作成します。Google は、要求元に最も近いデータセンターにリクエストを自動的にルーティングします。

画像分析を設定して、計算済みの特定の詳細を返すようにすることができます。これらの設定は、ステップのオプションでオンとオフを切り替えることができます。

詳細については、Google Cloud のドキュメントを参照してください。

テキストの分析

3 つのサービス プロバイダには、ある言語から別の言語にテキストを翻訳するオプションが用意されています。また、Azure にはテキストを文字変換するオプションがあります。

アマゾン ウェブ サービス

Amazon には、「アクセス キー ID」という名前の 2 つの API キーと、対応する「シークレット アクセス キー」が必要です。この API キー ペアには、Amazon Translate サービスへのアクセス権を付与する必要があります。詳細については、Amazon AWS のドキュメントを参照してください。

クラウド AI ステップを作成する場合は、入力の言語コードを指定します。

次の場所にある、サポートされている言語のリストを参照してください。

https://docs.aws.amazon.com/translate/latest/dg/what-is.html#what-is-languages

Microsoft Azure

Azure をテキスト分析に使用するには、場所と **API** キーが必要です。API タイプが「Translator」の「Cognitive Services」サービスを作成して、場所と API キーを取得します。

> API キーは、画像の分析とテキストの翻訳で異なります。

Azure プロバイダは、テキストの翻訳と文字変換を実行できます。クラウド AI ステップを作成する場合は、入力の言語を選択します。サポートされている言語と文字変換は、設計時に Azure からダウンロードされます。

詳細については、Microsoft Azure のドキュメントを参照してください。

Google Cloud

Google Cloud をテキスト分析に使用するには、「Cloud Translation API」へのアクセスを有効にして、API キーを作成します。

クラウド AI ステップを作成する場合は、[ターゲット言語] フィールドで入力テキストの言語コードを指定します。

次の場所にある、現在サポートされている言語のリストを参照してください。

https://cloud.google.com/translate/docs/languages

ステップ プロパティ

次のセクションでは、[クラウド AI] ステップ プロパティについて説明します。

> オプションの数と名前は、クラウド サービス プロバイダによって異なります。

データ タイプ

- [画像]: 画像を処理するには、このオプションを選択します。
- [テキスト]: テキストを分析するには、このオプションを選択します。

ソース
- [画像]: [ローカル ファイル]、[バイナリ データ] (変数)、[**RFS**]、および [**URL**] から選択します。ソースのタイプを選択した後に、ファイルのパスまたは URL を指定するか、変数を選択します。
- [テキスト]: テキスト フィールドにテキストを入力するか、テキストを貼り付けます。あるいは、テキスト変数を指定します。

プロバイダ
クラウド サービス プロバイダを選択します。
プロバイダを選択した後に、サービスに接続する場合に必要となるキーおよび他の設定を指定します。上記の各サービスの説明を参照してください。

アクション
アクションを選択し、選択したアクションに対して 1 つ以上のオプションを指定します。

言語
- [画像]: サービス プロバイダがサポートする言語の中から 1 つを選択します。出力 JSON ファイルには、選択した言語のセクションが含まれています。
- [テキスト]: 分析するテキストの言語を選択します。文字変換の場合は、[ソース言語] と [ターゲット スクリプト] を指定します。

結果
結果を保存するテキスト形式の変数を指定します。画像の場合、出力テキストには、JSON ドキュメントの作成に使用できる JSON マークアップが含まれています。

条件

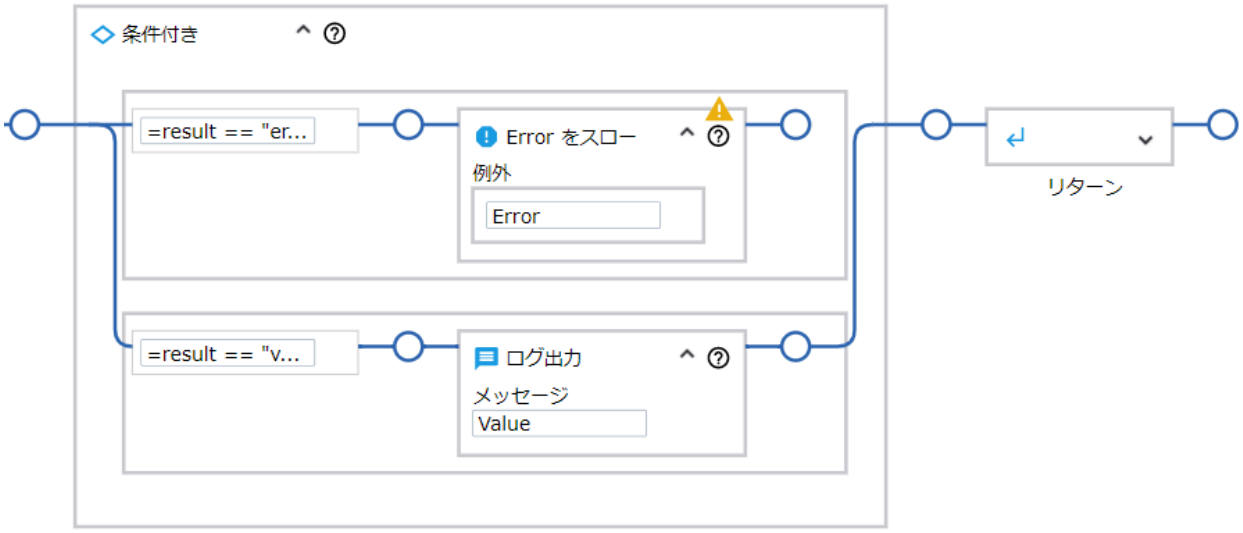
このステップでは、ロボットでのステップの実行に影響するブール値の条件を指定できます。このステップは頻繁に、ループ 内で使用されます。

プロパティ

名前
ステップの名前。

条件
それぞれがロボットの分岐に関連付けられている、ブール条件の変更可能なリスト。

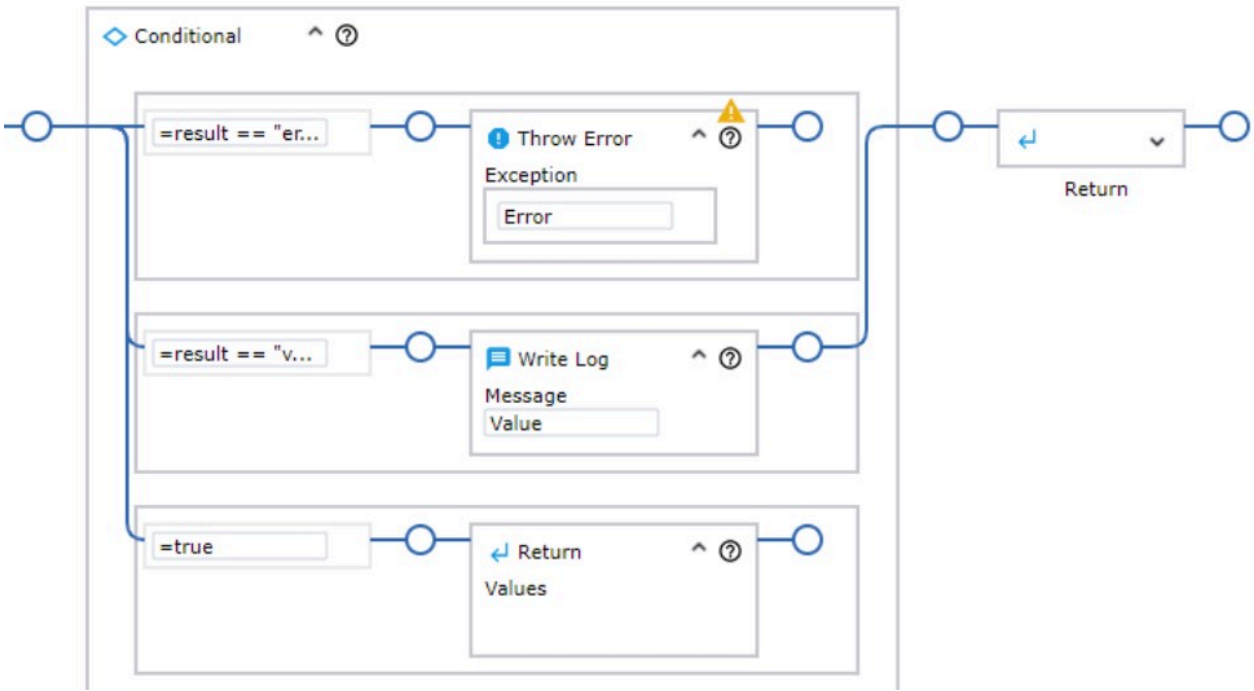
動作
ステップを実行すると、true と評価される最初の条件に属する分岐が実行されます。他のすべての分岐はスキップされます。一致する分岐がない場合、ステップは何も実行せず、ロボットは次のステップに進みます。
例 **1**



この例では、文字列の特定の値がチェックされます。一致する場合は、スロー エラー ステップでエラーがスローされます。それ以外の場合、文字列は別の特定の値に対してチェックされ、メッセージはログ出力ステップでログに記録されます。いずれの条件も true とならない場合も含め、すべての場合において、ロボットは戻るステップに進みます。

次の例に示すように、true という条件で条件分岐を作成することができます。上記の条件のいずれにも当てはまらない場合、この分岐は常に実行されます。1 つ以上の分岐が常に実行されるため、いずれの条件にも一致しない場合、ステップを暗黙的にスキップすることはできません。

例 **2**



この例は前の例と似ていますが、最初の 2 つの分岐のどちらにも一致しない場合、実行は戻るステップに進み、ロボットはすぐに終了します。

デバイスに接続

このステップを使用して、リモート デバイスに接続できます。Desktop Automation プロパティを設定する方法については、『Kofax RPA Desktop Automation サービス ガイド』を参照してください。

プロパティ

名前
ステップの名前。

デバイス
使用する動的リファレンス名を選択します。このリファレンス名は、「ロボットを呼び出す」ステップの [デバイス] プロパティで指定します。リストには動的リファレンスのみが表示されます。

ホスト
オートメーション デバイス名または IP アドレスを入力します。

ポート
オートメーション デバイスで使用するコマンド ポートを指定します (デフォルトは 49998)。このポートは、Desktop Automation サービスの設定で指定されています。

トークン
トークン名を指定します。トークンは、Desktop Automation サービスの設定の **[シングル ユーザー]** タブの **[トークン]** フィールドで選択したオートメーション デバイスに指定した名前と一致させる必要があります。

Desktop Automation サービスは、シングル ユーザー モードである必要があります。

タイムアウト
接続試行タイムアウトを秒単位で指定します。

試行数
接続試行数を指定します。各接続試行間には数秒の遅れがあります。

値の変換

このステップを使用して、手動で指定した値を変換したり、抽出ステップとともに抽出した値を変換したりできます。

ロボットのワークフローまたは状態ペインで値を右クリックし、**[ステップを挿入]** > **[変換]** の順にクリックすると、このステップを直接追加できます。

手動で指定した値の変換を実行するには、ドロップダウン リストから [値の変換] オプションを選択する必要があります。抽出された値の変換を実行するには、ドロップダウン リストからいずれかの抽出ステップを選択します。

プロパティ

値の変換
変換するソース値。

エクスプレッションを評価
値での使用が推奨される変換関数のリストが含まれています。特定の値に対して推奨される関数リストは、値の種類に応じて制限されます。変換関数を指定するには、ステップのコンテキスト メニューで **[エクスプレッションを評価]** をクリックし、次のいずれかを実行します。

- 推奨関数のリストから必要な関数を選択します。
- [プレーン] をクリックして、必要な関数を手動で入力します。たとえば、整数を表すテキストを抽出して整数変数に格納する必要がある場合は、エクスプレッション `$initial.integer()` を使用できます。

サポートされている関数と例の詳細については、エクスプレッションを参照してください。
コンバータは、1 つ以上のエクスプレッションを評価ステップを含めることができます。

現在のインを保存
変換された値を指定されたタイプの変数に保存します。値のタイプは変数のタイプと一致する必要があり、次のいずれかになります：整数、ブール値、数値、またはテキスト。

コンバータは、1 つ以上の現在のイン ステップを保存を含めることができます。たとえば、同じコンバータ グループ内で個人のフルネームを抽出し、それぞれ名と姓などの 2 つの変数に保存する必要がある場合に役立ちます。

コンバータ グループには、次の一連のアクション ステップを含めることもできます。

- 条件ステップ
- グループ ステップ
- スロー ステップ
- ログの書き込みステップ

例:「**hello world**」というエクスプレッションを大文字にする

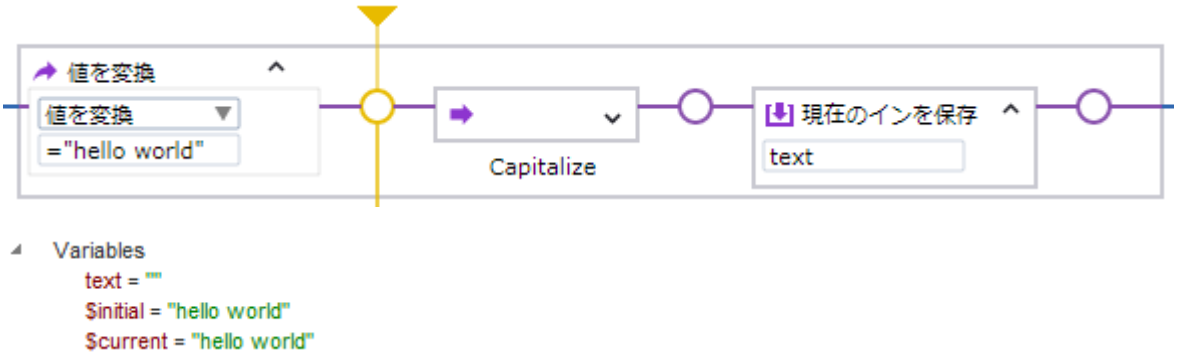
次の例は、「hello world」というエクスプレッションを大文字にし、テキスト タイプの変数に格納する方法を示します。

> コンバータ グループ内でワークフロー状態の変更を元に戻す必要がある場合、または特定の先行フロー ポイントの状態を確認する必要がある場合は、必要な先行フローポイントを実行することにより、変換ステップのグループ内に戻ることができます。また、現在のフロー ポイントの前のワークフローを変更すると、ワークフローの状態が自動的に調整されるため、変換ステップを再実行する必要はありません。

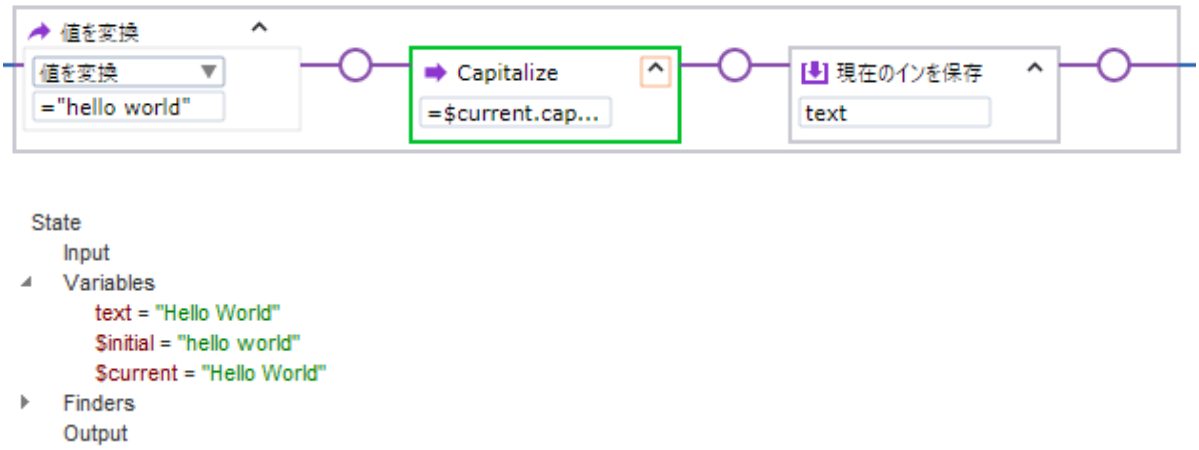
1. オートメーション ワークフローに [値の変換] ステップを追加し、次のように設定します。
   a. [値の変換] の下で**"hello world"**と入力します。

b. エクスプレッションの後に、フロー ポイントを右クリックして、**[エクスプレッションを評価] > Capitalize: (Text) -> Text** をクリックします。

2. 最初の部分を実行すると、 **"hello world"**エクスプレッションの (指定された) 初期値が変数 **$initial** と **$current** に割り当てられます。



1. 実行を続けると、エクスプレッションはコンバータによって大文字になり (次の例では **= $current.capitalize()**)、変数 **$current** には新しい値**"Hello World"**が割り当てられますが、**$initial** の変数値は変わらず残ります。$current 変数の新しい値**"Hello World"**は、[現在のインを保存] が実行された後、テキスト タイプの変数 **text** に保存されます。



$initial と $current 変数の詳細については、データの変換を参照してください。

コンポーネント セレクターのコピー

コンポーネント セレクターをアプリケーション ツリー ビューまたはタグ パスからコピーして、エディター ビュー内のステップのコンポーネント フィールドに貼り付けることができます。ツリー ビューまたはタグ パスで要素を選択し右クリックして、[コンポーネント セレクターのコピー] をクリックしてから使用するセレクター タイプに応じて、[コンパクト コンポーネント セレクターのコピー] または [フル パス コンポーネント セレクターのコピー] をクリックします。

コンパクト コンポーネント セレクターは、以下のように指定の要素に関する最小限の一意のセレクタをコピーします:

```
TR[class="odd"]:nth-of-type(1) > TD[class=" "][der_rendered="y"]:nth-of-type(4)
```

フル パス コンポーネント セレクターは、セレクターをアプリケーション ツリーのルートからコピーします。これは、同じ要素について次のようになります。

```
window > _document_ > HTML > BODY > DIV:nth-of-type(3) > DIV:nth-of-type(2) > DIV:nth-
of-type(2)
    > DIV > TABLE > TBODY > TR:nth-of-type(1) > TD:nth-of-type(4)
```

セレクターはクリップボードにコピーされます。[Ctrl+V] でこれをコンポーネント フィールドに貼り付けることができます。

セレクター構文の詳細については、セレクター構文を参照してください。

コピー

アプリケーション ツリー ビューから要素をコピーして、要素を必要に応じて使用できます。これは、ツリー ビューから特定のアプリケーション要素を使用するカスタムのコンポーネント ファインダーを記述する必要がある場合に有効です。

ツリー ビューで、要素を選択して右クリックし、[コピー] をクリックして、必要な要素のタイプに応じて [サブ ツリー] または [ノード] をクリックします。

選択したサブツリー要素をコピーするには、[Ctrl+C] を使うこともできます。

[サブ ツリー] アクションは、ルート要素をすべてのサブ要素とともに以下のような XML 文字列としてコピーします。

```
<DIV class="teaser__description" der_rendered="y" der_x="1103" der_y="294"
 der_width="270" der_height="20">
 <SPAN class="teaser__nowrap" der_rendered="y" der_x="1103" der_y="295" der_width="135"
 der_height="17">
   Text
   <SPAN class="teaser__age" der_rendered="y" der_x="1222" der_y="299" der_width="16"
 der_height="13">
     0+
   </SPAN>
 </SPAN>
</DIV>
```

[ノード] アクションは、上記と同じ要素の場合、以下のようにルート要素のみを XML 文字列としてコピーします。

```
<DIV class="teaser__description" der_rendered="y" der_x="1103" der_y="294"
 der_width="270" der_height="20"/>
```

この要素はクリップボードにコピーされます。

カスタム アクション

[カスタム アクション] ステップでは、Connector を使用して口ボット内からデータを処理するために外部リソースを利用できます。Connector は、口ボットが使用できる 1 つ以上のアクションを定義し、アクションを実装するために必要なすべてのリソースを含むパッケージです。

Connector は、外部プログラムの呼び出し、シェル コマンド、またはプライベート インスタンスで JavaScript と Python を実行するアクションを定義できます。これらのアクションは、Design Studio の [カスタム アクション] ステップのアクションとして利用できます。

追加のソリューション、ロボット、コネクタなどを見つけるには、https://smarthub.kofax.com/ の Kofax Intelligent Automation SmartHub にアクセスしてください。

Connector の設定

このセクションでは、Connector を [カスタム アクション] ステップで使用するための設定方法について説明します。

ロボットで Connector を使用する前に、[RoboServer 設定] アプリケーションの [セキュリティ] タブで [**Connector** の使用を許可] を選択します。[RoboServer 設定] は、Windows のスタート メニューから起動できます。詳細については、『Kofax RPA 管理者ガイド』の「RoboServer 設定」を参照してください。

Connector の作成

Connector は、拡張子が `.connector` の .zip アーカイブとして配布されます。[カスタム アクション] ステップでロボットを作成する前に、ZIP を使用して Connector をアーカイブし、ファイル拡張子を `.connector` に変更して、プロジェクトに追加します。プロジェクトを Management Console にアップロードすると、使用できる Connector が **[リポジトリ]** > **[リソース]** タブで確認できるようになります。次に、必要に応じて、Connector を RoboServer および自動化されたデスクトップ コンピュータにアップロードします。

各 .zip ファイルには、ルートに `manifest.json` ファイルが含まれている必要があります。ファイルには、Connector で使用可能なすべてのアクションを定義するマニフェストが含まれている必要があります。マニフェストの説明は以下を参照してください。

各アクションは、一連のパラメータ、応答、およびコマンド ラインを定義します。パラメータはロボット設計者に提示され、ロボットが入力する必要があります。コマンド ラインは、パラメータの組み合わせを使用して構成され、要求の実行に使用されます。要求の出力は解析され、応答で定義された変数を満たすために使用されます。

Management Console への Connectors のアップロード

Design Studio で [カスタム アクション] ステップを含むロボットのデザインを完了したら、Management Console にアップロードします。プロジェクトを Management Console にアップロードすると、使用できる Connector が [リポジトリ] の [リソース] タブで確認できるようになります。次に、必要に応じて、Connector を RoboServer および自動化されたデスクトップ コンピュータにアップロードします。

Python および NodeJS を使用している Connector は、オンデマンドでロードされます。これらを使用していない場合は、リソースは不要です。それぞれの Connector は独自の `python` または `node.js` インスタンスを受け取るため、ユーザーは必要に応じて、バージョンが異なるこのタイプの複数の Connector を実行できます。

プログラムの実行 (タイプ：プログラム)

プログラム アクションは、抽出された Connector ファイルに設定された現在の作業ディレクトリを使用して、指定されたプログラムを直接呼び出します。実行可能ファイルが Connector パッケージの一部で

ある場合、マニフェストは相対パス (`./executable` など) を使用してこれを明示的にする必要があります。この方法を使用しなかった場合、ロボットが実行可能ファイルを見つけることができない可能性があります。

追加の構成 (PATH 設定または Linux ダイナミック ローダー設定) が必要な場合、シェル ラッパーを使用して設定をセットアップし、実行可能ファイルを呼び出すことをお勧めします。パラメータは、追加のエスケープをせずにプログラムに直接渡されます。

シェルコマンドの実行 (タイプ：shell)

シェル インターフェイスは、シェルを起動して、抽出された Connector ファイル ディレクトリに設定された現在の作業ディレクトリでコマンド ラインを直接実行します。このコマンド ラインの処理はシェル自体によって処理されるため、プラットフォームに応じて異なります。

- Windows では、このアクションによって `CMD/C` が呼び出され、コマンド ラインのすべての要素が直接シェルに渡されます。
- Linux では、このアクションによって `bash -c` が呼び出され、コマンド ラインの各要素が個別の引数として渡されます。パラメータは、追加のエスケープをせずにシェルに直接渡されます。詳細については、`bash -c` オプションに関するドキュメントを参照してください。

JavaScript の実行 (タイプ：ノード)

Node.js JavaScript の要求は `function()` コンテキストにラップされて実行されます。コードに `require` ステートメントが含まれている場合、それらは Connector パッケージのルートにある `node_modules` ディレクトリを使用して解決されます。

デフォルトでは、パラメータは文字列に変換され、コマンド ラインに挿入される前にエスケープされます。コマンド ラインはエスケープされません。

インターフェイスが同期している場合、ロボットは JavaScript から返された値と例外値を受け取ります。この 2 つのフィールドのうち 1 つだけが空でない値を持ちます。オブジェクトは、ロボットに返される前に JSON にシリアル化されます。

特定の Node.js 実行可能ファイルは、ルートに配置することでパッケージに埋め込むことができます。実行可能ファイルが存在しない場合、Kofax RPA に含まれている Node.js LTS のバージョンが使用されます。

Python の実行 (タイプ：python)

Python の要求は、`exec()` ステートメントと永続的な `private global()` 辞書を使用して実行されます。Connector が独自のモジュールを提供できるように、Connector パッケージのルートが sys.path の前に追加されます。

デフォルトでは、パラメータは文字列に変換され、コマンド ラインに挿入される前にエスケープされます。コマンド ラインはエスケープされません。

Python の `exec()` ステートメントはトップ レベルで `return` ステートメントを許可しないため、要求はその結果を代わりにグローバル `rpa_return` 変数に割り当てる必要があります。この変数は、リクエスト間のグローバル スコープから削除されます。インターフェイスが同期している場合、ロボットは `rpa_return` の変数と例外値を受け取ります。この 2 つのフィールドのうち 1 つだけが空でない値を持ちます。オブジェクトは、ロボットに返される前に JSON にシリアル化されます。

`RPAConnector` 名は内部使用のために予約されています。モジュール名として使用したり、 Connector パッケージ内に `RPAConnector` というディレクトリを作成しないでください。

Kofax RPA はデフォルトの Python インタープリターを使用します。この設定は、マニフェストで「python3.8」や「/usr/bin/python3」などの別の名前またはパスを使用することでオーバーライドできます。指定されたインタープリターはシステムが解決します。

パッケージは、使用する Python バージョンに合わせて設定する必要があります。サポートされている Python のバージョンについては、『Kofax RPA 技術仕様』 を参照してください。

ローカル デバイスでの Connector の実行

ロボットのローカル デバイスでカスタム Connector を使用するには、ローカル デバイス RoboServer の [**RoboServer** 設定] アプリケーションで [**Connector** の使用を許可] オプションを選択します。

コネクタのタイムアウト

デフォルトでは、ロボットはコネクタが処理を完了して結果を取得するまで待機します。デフォルトの [カスタム アクション] ステップのタイムアウト値は 240 秒に設定されています。

コネクタがこの時間を超えることが予想される場合は、非同期またはバックグラウンド スレッドで処理を実行するようにコネクタを変更し、コネクタに (部分的な) 結果と完了をポーリングする 2 番目の [カスタム アクション] ステップを追加することをお勧めします。このようにして、ロボットはコネクタと並行してステータスの問い合わせを迅速に実行し、結果を処理します。

この処理が実行されない場合は、タイムアウトをより高い値に設定します。この設定を行うには、installation\bin フォルダ内の common.conf ファイルを編集します。次の行を追加します:

- wrapper.java.additional.<nr>=-Dtimeout.action.seconds=<timeout>

<nr> を、ファイル内にすでに存在する wrapper.java.additional 設定の次のシーケンス番号に置き換えます。

<timeout> を必要なタイムアウト値 (秒) に置き換えます。

マニフェスト

Manifest.json ファイルには、次の JSON 要素が含まれている必要があります。

フィールド	ステータス	説明
アクション	必須	アクションの配列。
名前	必須	カスタム アクション ステップで表示されている Connector の名前。
python	任意	String Python アクションで実行する実行可能ファイルを指定します。この設定はすべてのプラットフォームに適用されますが、次のプラットフォーム固有の設定のいずれかを使用して上書きすることができます。 デフォルト:「python」

フィールド	ステータス	説明
python-windows	任意	String 記述されている場合、Windows プラットフォームで Python アクションに対して実行する実行可能ファイルを指定します。
python-linux	任意	String 記述されている場合、Windows プラットフォームで Linux アクションに対して実行する実行可能ファイルを指定します。
python-support	任意	整数 使用される Python のメジャー バージョンを指定します。 たとえば、Python バージョン 3.x の場合、値は 3 です。

アクション

次の表はアクションの形式を詳しく説明しています。

アクション形式

フィールド	ステータス	説明
名前	必須	アクションの名前。
タイプ	必須	アクションのタイプ。次のいずれかである必要があります: 「`program`」、「`shell`」、「`node`」または 「`python`」
parameters	任意	カスタム アクション ステップで入力フィールドとして表される一連のパラメータ。

フィールド	ステータス	説明
レスポンス	任意	カスタム アクション ステップで出力フィールドとして表される一連のレスポンス。このフィールドを省略すると、アクションの定義に応じてデフォルトのレスポンス セットが使用されます。デフォルトのレスポンスは次のとおりです。 • 「shell」および「python」アクションの場合、returncode タイプの整数が追加されます。 • 「stdout」および/または「stder」アクションの場合、対応するテキスト フィールドが結果セットに追加されます。 • 「node」および「python」アクションの場合、result および error という 2 つの文字列フィールドが追加されます。 「shell」および「program」アクションの場合、wait フィールドを falseに設定して、レスポンスを受け取らないようにすることができます。この場合、アクションはバックグラウンドで開始および実行されます。「node」または「python」アクションをレスポンスなしに設定することはできないことに注意してください。
commandline	必須	要求コマンドのパラメーターを表す文字列のセット。「プログラム」アクションの場合、最初のパラメーターが実行可能ファイルになります。パラメーターは %n エスケープを使用して挿入できます (パラメーター配列から n th番目の値を挿入します)。%% を使用して、パーセント記号を挿入します。 パラメータ置換後に空の要素は削除されます。 「node」および「python」要求の場合、完全なスクリプトを送信できます。
prune	任意	ブール値 空の文字列に展開された要素をコマンド ライン配列から削除することを示します。 デフォルト: true

フィールド	ステータス	説明
wait	任意	ブール値 ロボットがアクションの終了を待つことを示します。「node」アクションおよび「python」アクションの場合、またはアクションにレスポンス配列がある場合、この設定は無視されます。 デフォルト: true
stdout	任意	ブール値 変数にある「program」または「shell」アクションからの標準出力ストリームのキャプチャを有効にします。「wait」が false の場合、このフィールドは無視されます。 デフォルト: false
stderr	任意	ブール値 変数にある「program」または「shell」アクションからの標準エラー ストリームのキャプチャを有効にします。「wait」が false の場合、このフィールドは無視されます。 デフォルト: false

パラメータ
次の表はパラメータの形式を詳しく説明しています。

パラメータ形式

フィールド	ステータス	説明
名前	必須	パラメータの名前。この名前は、カスタム アクション ステップに表示されます。
タイプ	必須	パラメータのタイプ。このタイプは、次のいずれかである必要があります。「string」または「number」「number」パラメータは整数に制限されます。
デフォルト	任意	パラメータのデフォルト値。 デフォルト: パラメータのタイプに応じて、「」または 0。

フィールド	ステータス	説明
最小	任意	数値 パラメータの最小許容値。この属性は、「number」パラメータでのみサポートされています。他のタイプでは無視されます。 デフォルト: -2147483648
最大	任意	数値 パラメータの最大許容値。この属性は、「number」パラメータでのみサポートされています。他のタイプでは無視されます。 デフォルト: +2147483647
任意	任意	ブール値 オプションのパラメータは、ロボットに割り当てられていないか、空のままにすることができます。 デフォルト: false
エスケープ	任意	ブール値 値を引用符で囲み、特殊文字をエスケープします。この属性は現在、「node」および「python」文字列パラメータでサポートされています。他のすべての状況では無視されます。 デフォルト: true

数値パラメータの場合、デフォルト値は最小値と最大値の間にある必要があります (両端を含む)。3 つの設定はすべて、-2147483648 から +2147483647 の範囲内である必要があります。

レスポンス

次の表は、レスポンス形式を示します。

レスポンス形式

フィールド	ステータス	説明
名前	必須	レスポンスの名前。この名前は、カスタム アクション ステップに表示されます。
タイプ	必須	レスポンスのタイプ。このタイプは、次のいずれかである必要があります。「string」または「number」「number」レスポンスは整数に変換されます。

フィールド	ステータス	説明
任意	任意	ブール値 出力にレスポンスが存在する必要がないことを示します。レスポンスが省略された場合、変数にはデフォルト値が入力されます。 デフォルト: `false`
デフォルト	任意	パラメータのデフォルト値。 この値は、オプションのパラメータがレスポンスから省略された場合に使用されます。 デフォルト: パラメータのタイプに応じて、「」または 0。

アプリケーションのレスポンスの浮動小数点数値は整数に変換されます。

manifest.json の例

次の Linux ベースの例では、対応する関数を呼び出して 2 つの数値を加算するさまざまなコネクタ (シェル、ノード、python、プログラム) を使用しています。関数パラメータは、この関数に渡される順序で定義されます(例: `First number` および `Second number` の順序)。計算の結果は、`Sum` 値に割り当てられた変数に格納されてロボットに戻されます。出力値は、コネクタごとに異なった方法で割り当てられます。

python.connector

このコネクタを機能させるには、コネクタを備えたロボットが実行されているコンピュータに最新の Python バージョンをインストールします。

```
{"actions": [
    {
      "name": "Adder",
      "type": "python",
      "parameters": [
        {
          "name": "First number",
          "type": "number"
        },
        {
          "name": "Second number",
          "type": "number",
          "min": 1,
          "default": 1000
        }
      ],
      "response": [
        {
          "name": "Sum",
          "type": "number"
        }
      ],
      "commandline": [
        "rpa_return = { 'Sum': %1 + %2 }"
      ]
    }
  ],
  "name": "Adder Function (python)"
}
```

shell.connector

```
{
  "actions": [
    {
      "name": "Adder",
      "type": "shell",
      "parameters": [
        {
          "name": "First number",
          "type": "number"
        },
        {
          "name": "Second number",
          "type": "number",
          "min": 1,
          "default": 1000
        }
      ],
      "response": [
        {
          "name": "Sum",
          "type": "number"
        }
      ],
      "commandline": [
 "echo \"{ \\\"Sum\\\": $((%1 + %2)) }\""
      ]
    }
  ],
  "name": "Adder Function (shell)"
}
```

node.connector

```
{
  "actions": [
    {
      "name": "Adder",
      "type": "node",
      "parameters": [
        {
          "name": "First number",
          "type": "number"
        },
        {
          "name": "Second number",
          "type": "number",
          "min": 1,
          "default": 1000
        }
      ],
      "response": [
        {
          "name": "Sum",
          "type": "number"
        }
      ],
      "commandline": [
        "return { Sum: %1 + %2 };"
      ]
    }
  ],
  "name": "Adder Function (node)"
```

```
}
```

program.connector

このコネクタは、コネクタによって呼び出される実行可能プログラム「add」を使用します。

```
{
  "actions": [
    {
      "name": "Adder",
      "type": "program",
      "parameters": [
        {
          "name": "First number",
          "type": "number"
        },
        {
          "name": "Second number",
          "type": "number",
          "min": 1,
          "default": 1000
        }
      ],
      "response": [
        {
          "name": "Sum",
          "type": "number"
        }
      ],
      "commandline": {
      "windows":["./add.exe", "%1", "%2"],
      "linux":["./add", "%1", "%2"],
    }
  ],
  "name": "Adder Function (program)"
}
```

レスポンスの定義

プログラム

応答として送信される JSON データは、JavaScript Object Notation (JSON) Data Interchange Format 標準 RFC-8259 に準拠する必要があります。プログラム コネクタは、この標準で要求されているように、出力が必ず UTF-8 でエンコードされているようにする必要があります。

プログラムのアクションは次のとおりです。

- アクションに応答配列がある場合、待機フィールドは無視され、プログラムが終了するまで口ボットは待機します。プログラムの標準出力はキャプチャされ、JSON 文字列として解析され、変数にはこの文字列の各フィールドが入力されます。プログラムが有効な JSON 文字列を生成しない場合、口ボットは失敗します。
- アクションにレスポンス配列がなく、待機フィールドが false の場合、ステップにはレスポンス変数がありません。
- アクションにレスポンス配列がなく、待機フィールドが true の場合、ステップには次のレスポンス変数があります。
  - `rc`: プログラムの戻りコード。
  - `stdout`: プログラムの標準出力ストリーム (`stdout` が `true` の場合のみ提供されます)。
  - `stderr`: プログラムの標準エラー ストリーム (`stderr` が `true` の場合のみ提供されます)。

Connector の構成に関係なく、標準出力ストリームと標準エラー ストリームは、INFO レベルで Kofax RPA ログに記録されます。

シェル

応答として送信される JSON データは、JavaScript Object Notation (JSON) Data Interchange Format 標準 RFC-8259 に準拠する必要があります。シェル コネクタは、この標準で要求されているように、出力が必ず UTF-8 でエンコードされているようにする必要があります。

シェル アクションは、Windows では CMD.EXE、Linux では bash を使用して実行されます。

- アクションに応答配列がある場合、待機フィールドは無視され、シェルが終了するまで口ボットは待機します。シェルの標準出力はキャプチャされ、JSON 文字列として解析され、変数にはこの文字列の各フィールドが入力されます。プログラムが有効な JSON 文字列を生成しない場合、口ボットは失敗します。
- アクションにレスポンス配列がなく、待機フィールドが false の場合、ステップにはレスポンス変数がありません。
- アクションにレスポンス配列がなく、待機フィールドが true の場合、ステップには次のレスポンス変数があります。
  - `rc`: シェルの戻りコード。
  - `stdout`: シェルの標準出力ストリーム (`stdout` が `true` 場合のみ提供されます)。
  - `stderr`: シェルの標準エラー ストリーム (`stderr` が `true` 場合のみ提供されます)。

Connector の構成に関係なく、標準出力ストリームと標準エラー ストリームは、INFO レベルで Kofax RPA ログに記録されます。

NodeJS

コマンド ラインは、関数のコンテキストで実行されます。コマンドは、結果を口ボットに返すために return ステートメントを使用する必要があります。

- アクションにレスポンス ブロックがある場合、オブジェクトまたは JSON 文字列を返すことが期待され、変数にはこのオブジェクトの各フィールドが入力されます。例外がスローされ、処理されない場合、口ボットは失敗します。
- アクションにレスポンス ブロックがない場合、ステップには次の 2 つの応答変数があります。

  結果: ステートメントが正常に完了した場合、この変数はステートメントによって返された値を受け取ります。

  エラー: 例外がスローされ、JavaScript コードで処理されない場合、この変数には例外のテキストが含まれます。

Python

コマンド ラインは `exec()` 関数を使用して実行されます。このコマンドは、口ボットに結果を返すために、グローバル変数 `rpa_return` に値を割り当てる必要があります。

1. アクションにレスポンス ブロックがある場合、`rpa_return` はオブジェクトまたは JSON 文字列を含める必要があり、変数にはこのオブジェクトの各フィールドが入力されます。例外がスローされ、処理されない場合、口ボットは失敗します。

1. アクションにレスポンス ブロックがない場合、ステップには次の 2 つの応答変数があります。
   - 結果: ステートメントが正常に完了した場合、この変数は `rpa_return` の値を受け取ります。

- エラー: 例外がスローされ、Python コードで処理されない場合、この変数には例外のテキストが含まれます。

実装の詳細

.zip ファイルの次の要素は予約されています。

タイプ	パス	説明
ファイル	/manifest.json	マニフェスト
ファイル	/node	Linux プラットフォーム用の Node.js 実行可能ファイル。このファイルが存在しない場合、Kofax RPA に含まれている Node.js インスタンスが使用されます。
ファイル	/node.exe	Windows プラットフォーム用の Node.js 実行可能ファイル。このファイルが存在しない場合、Kofax RPA に含まれている Node.js インスタンスが使用されます。
ディレクトリ	/node_modules	Node.js モジュールの場所。
ディレクトリ	/RPAConnector	内部使用のために予約されています。
ディレクトリ	/node_modules/RPAConnector	内部使用のために予約されています。

データベース

このトピックでは、ロボットで使用できるデータベース ステップを一覧表示します。

- データベースから削除
- SQL 実行
- データベース データ抽出
- データベース照会
- データベース データ登録

データベースから削除

このステップを使用して、データベースから値を削除します。データベース内のレコードを変更するには、SQL 実行ステップを使用します。選択したデータベースからデータを抽出するには、データベース照会ステップを使用します。データベースにデータを保存するには、「データベース データ登録」ステップを使用します。

このステップでは、書き込み先のテーブルがタイプと構成の定義と一致することが検証され、いずれかが一致しない場合はエラーが生成されます。データベースにアクセスできる限り、ロボットの編集中は警告も表示されます。この場合は、ステップを右クリックし、[テーブルをデータベースに (再) 生成] を選択して SQL ダイアログ ボックスを開き、Design Studio からテーブルを更新できます。

デバッグ モードで、[次の場合停止] で [値が返されるか、または値が保存されます] が選択されている場合、[データベースから削除] ステップを使用するロボットは、呼び出し元のベーシック エンジン ロボットに実行が戻るまで停止しません。つまり、呼び出されたロボットは、デバッガーで停止する前にデータベースの複数の値を削除する可能性があります。

プロパティ

データベース マッピング
「ロボットの呼び出し」ステップで指定されたデータベースからデータベース マッピングを選択します。

変数
値の読み取り元の変数を選択します。この変数は必要なタイプである必要があります。

キー
データベース内の値のキーを計算する方法を選択します。[タイプ内に定義されたキー] が選択されている場合、タイプ エディターで [データベース キーの一部] としてマークされている属性を指定する必要があります。そうでない場合、値が見つかりません。また、[計算されたキー] を選択してキーを計算する式を指定することもできます。

監査データ
選択すると、値を含む監査データ フィールドが含まれます。詳細については、「データベースへのデータ格納」を参照してください。

SQL 実行

選択したデータベースで SQL ステートメントを実行するには、このステップを使用します。`create`、`update`、`insert`、`delete`、`drop` などの SQL コマンドを使用して、データベースのレコードを変更できます。`call` コマンドを使用してストアド プロシージャを実行することもできます (例: `call add_record("name")`)。データベースからレコードを読み取るには、「データベース照会」ステップを使用します。データベースにデータを保存するには、「データベース データ登録」ステップを使用します。

プロパティ

以下のプロパティを使用して「SQL 実行」ステップを設定します。

データベース マッピング
「ロボットの呼び出し」ステップで指定されたデータベースからデータベース マッピングを選択します。

SQL ステートメント
選択したデータベースで実行する SQL ステートメントを入力します。

ロボットは入力したステートメントを評価しません。事前に SQL ステートメントを確認してください。そうしないとステップを実行してステートメントの有効性を確認するしかありません。
「データベース データ登録」または「SQL 実行」ステップで使用されるデータベースのテーブルを削除または変更しないでください。

変更された行

選択した場合、整数型の変数を指定して、SQL ステートメントによって変更された行の数をステップの出力として取得できます。

タイムアウト

選択した場合、SQL ステートメントの送信タイムアウトを秒単位で指定できます。値はゼロより大きい必要があります。タイムアウトは DBMS を通じて強制され、タイムアウトに達するとデータベース固有のエラー メッセージとともに `DeviceIssue` 例外がスローされます。

それぞれの DBMS タイプおよび JDBC ドライバーに応じて、タイムアウトの処理方法が異なる場合があります。たとえば、一部のシステムにおいては、特定のステートメントの途中でのタイムアウトの発生が許可されていないこともあります。

ステートメント タイプ

リストからステートメント タイプを選択します。

準備されたステートメント

このステートメント タイプを使用して、効率よく SQL ステートメントを繰り返し実行します。準備されたステートメントは、実際の値の代わりにパラメータ プレースホルダを使用して記述されたクエリです (例: `SELECT * FROM MyTable WHERE string = ?`)。任意の SQL ステートメントを指定できますが、[入力値] パラメータのみを使用するように制限されています。また、DBMS および JDBC ドライバーが準備されたステートメントをサポートしていることを確認してください。

ストアド プロシージャ コール

このステートメント タイプを使用して、ストアド プロシージャを呼び出します。例: `CALL db.Procedure(?)`、`? = CALL db.Procedure(?)`、ここで、「?」はパラメータのプレースホルダとして機能します。ストアド プロシージャ コールは 「CALL」 で開始し、必要に応じて戻り値パラメータを指定する必要があります。また、[入力値] パラメータと [出力値] パラメータの両方をサポートします。

パラメータ

準備されたステートメントとストアド プロシージャ コール ステートメントの両方のタイプに、さまざまなパラメータを追加できます。クエリ内のクエスチョン マーク (?) はパラメータのプレースホルダであり、パラメータの順序に対応しています。

例

使用するクエリ:

```
SELECT * FROM MyTable WHERE string = ? AND int = ?
```

パラメータ:

- content
- 123

クエリは自動的に `SELECT * FROM MyTable WHERE string = content AND int = 123` に変換されます。

このステップで定義するパラメータは、式として使用できます。これにより、クエリ パラメータを動的に定義できます。ユーザー入力によるエラーやインジェクションを防ぐために、すべてのシナリオでこのパラメータ置換を使用することをお勧めします。

ストアド プロシージャ コール ステートメントの場合、[入力値]、[出力値]、または [入力値/出力値] パラメータを選択できます。

- [入力値] パラメータは、式としてロボットに入力されます。それらの値は、クエリに配置される前に評価されます。
- [出力値] パラメータは、値を返すために使用され、SQL 変数タイプの選択もサポートします。パラメータ タイプと SQL 変数タイプは一致する必要があります。一致しないと、エラーが返されます。詳細については、このセクションの後半を参照してください。
- [入力値/出力値] パラメータも SQL 変数タイプの選択をサポートし、[出力値] パラメータとして機能します。ただし、選択した変数の現在の値を [入力値] として使用します。

[出力値] または [入力値/出力値] パラメータ タイプを選択すると、[手動パラメータの詳細] チェック ボックスが表示され、必要に応じて SQL パラメータ タイプを選択し、そのスケールを指定できます。通常、これらの詳細は自動的に取得されますが、一部の JDBC ドライバーはこれをサポートしていないため、タイプを手動で設定する必要があります。

リストから SQL 変数タイプを選択するときは、それが [ストアド プロシージャ コール] フィールドで定義されたパラメータ タイプと一致していることを確認してください。必要な SQL 変数タイプがリストに見つからない場合は、ロボットで使用できないことを意味します。

スケールは、NUMERIC または DECIMAL 値の区切り記号の右側の桁数を表します。NUMERIC および DECIMAL タイプのパラメータにはスケールが必須であり、指定する必要があります。たとえば、パラメータ タイプ NUMERIC(24,12) を入力すると、スケール値は 12 になります。

[出力値] または [入力値/出力値] パラメータから返される値は、それぞれのロボット値にマップされます。次の表に、ロボット パラメータ タイプとそれぞれの SQL 変数タイプを示します。

ロボット パラメータ タイプ	**SQL** 変数タイプ
整数	BIGINT、INTEGER、SMALLINT、TINYINT
Number	DECIMAL、DOUBLE、FLOAT、NUMERIC、REAL
テキストまたはパスワード	CHAR、CLOB、DATALINK、LONGVARCHAR、LONGNVARCHAR、NCH
ブール値	BIT、BOOLEAN
バイナリ	BINARY、BLOB、LONGVARBINARY、VARBINARY
Time	TIME、TIME_WITH_TIMEZONE
Date	DATE
DateTime	TIMESTAMP、TIMESTAMP_WITH_TIMEZONE

構成された JDBC ドライバーは、ロボットの値をクエリ値に変換します。変換プロセスは、JDBC のバージョン、DBMS のタイプ、バージョン、設定など、さまざまな要因によって異なります。パラメータを適用できるように、式の値を変換するか、クエリを手動で調整する必要がある場合があります。

特に、日付と時刻の値は手動による変換が必要になる場合があります。これは、デフォルトでタイム ゾーン付きの日付と時刻の値をサポートする JDBC ドライバーがほとんどないためです。以下は、DBMS ごとの値の例です。

- **IBM DB2**

  入力された日時の値の形式をタイム ゾーンなしの文字列にする必要がある場合があります。

  例: yyyy-MM-dd HH:mm:ss.SS
- **Microsoft SQL Server**

入力された日時の値の形式を文字列にする必要がある場合があります。

例: yyyy-MM-dd HH:mm:ss.SSSSSSS xxx

[入力値/出力値] パラメータに Number 変数タイプを使用すると、ストアド プロシージャで明示的な変換を行っても、[出力値] が [入力値] の精度を達成する場合があります。

可能であれば [出力値] パラメータを使用し、それ以外の場合は [入力値] パラメータに対して適切で正確な値を使用します。

例: パラメータが [入力値/出力値] として設定されている場合、NUMERIC 値を 「1.2」 に設定するストアド プロシージャはその値を 「1.0」 として返すことがあり、初期値の精度は 1 (0.0 または 1 など) のみでした。

- **MySQL**
  日時の値の形式をタイム ゾーンなしの文字列にする必要がある場合があります。

  例: yyyy-MM-dd HH:mm:ss
- **Oracle**
  日時の値の形式を文字列にする必要がある場合があります。

  例: dd-MMM-yyyy HH:mm:ss.SS a xxx

  タイムゾーンなしの例: dd-MMM-yyyy HH:mm:ss.SS a

  日時の形式はロケールに大きく依存する可能性があることに注意してください。

  ストアド プロシージャから整数全体を返すことはサポートされていない場合があります。これを修正するには、ロボットの式で round() メソッドを使用します。

  以前のバージョンの JDBC ドライバーでは、パラメータ タイプとスケールを自動的に判別できない場合があることに注意してください。
- **PostgreSQL**
  日時の値の形式を文字列にする必要がある場合があります。

  例: yyyy-MM-dd HH:mm:ss.SS xxx

  また、クエリ内で変換が必要になる場合があります。

  例: to_timestamp(?, 'yyyy-MM-dd hh24:mi:ss.us xxx')

  JDBC ドライバーのすべてのバージョンで、パラメータ タイプとスケールを自動的に判別できない場合があることに注意してください。すべてが「その他」タイプとして報告されますが、これはロボットではサポートされていません。変数タイプを手動で選択した場合でも、クエリの各パラメータを明示的にキャストする必要がある場合があります。

  例: 'call db.proc(? :: character varying)'。ここでパラメータは VARCHAR タイプです。

  デフォルトでは、関数呼び出しのみがサポートされています。実際のストアド プロシージャを使用するには、Management Console で接続 URL プロパティ 'escapeSyntaxCallMode' を 'call' または 'callIfNoReturn' に設定する必要がある場合があります。

データベース データ抽出

このステップを使用して、データベース内の値を検索します。データベース内のレコードを変更するには、SQL 実行ステップを使用します。選択したデータベースからデータを抽出するには、データベース照会ステップを使用します。データベースにデータを保存するには、「データベース データ登録」ステップを使用します。

このステップでは、書き込み先のテーブルがタイプと構成の定義と一致することが検証され、いずれかが一致しない場合はエラーが生成されます。データベースにアクセスできる限り、ロボットの編集中は

警告も表示されます。この場合は、ステップを右クリックし、[テーブルをデータベースに (再) 生成] を選択して SQL ダイアログ ボックスを開き、Design Studio からテーブルを更新できます。

デバッグ モードで、[次の場合停止] で [値が返されるか、または値が保存されます] が選択されている場合、[データベース データ抽出] ステップを使用するロボットは、呼び出し元のベーシック エンジン ロボットに実行が戻るまで停止しません。

ロボットのワークフローまたは状態ペインで値を右クリックし、**[ステップを挿入]** > **[データベース データ抽出]** の順にクリックすると、このステップを直接追加できます。

プロパティ

データベース マッピング
「ロボットの呼び出し」ステップで指定されたデータベースからデータベース マッピングを選択します。

変数
値の読み取り元の変数を選択します。この変数はコンプレックス タイプである必要があります。

キー
データベース内の値のキーを計算する方法を選択します。[タイプ内に定義されたキー] を選択すると、タイプ エディターで [データベース キーの一部] とマークされている属性がキーの計算に使用されます。それ以外の場合は、[計算されたキー] を選択するときにキーを計算する式を指定します。

空の属性だけを上書き
選択すると、選択した複雑な変数の空のフィールドのみがデータベースのデータで上書きされます。クリアすると、複雑な変数の既存の値が置き換えられます。デフォルトでは選択されていません。

監査データ
選択すると、値を含む監査データ フィールドが含まれます。詳細については、「データベースへのデータ格納」を参照してください。

データベース照会

このステップを使用すると、選択したデータベースからデータを抽出できます。データベース内のレコードを変更するには、SQL 実行 ステップを使用します。データベースにデータを保存するには、「データベース データ登録」ステップを使用します。

このステップは、アプリケーションをレコーダー ビューで開き、抽出された 1 行を表示します。行のデータは、テーブル ビューとツリー ビューで使用できます。このステップは、クエリから返された行に対してイテレーションを行うループ ステップとして機能します。イテレーションを行うたびに、次の行のために抽出されたデータでビューの内容が更新されます。すべての行のイテレーションが完了する (またはロボットが例外やブレイク ステップによってループを終了する) と、アプリケーションは自動的に閉じます。また、ロボットが実行を停止した場合にもアプリケーションは閉じます。

ロボットは、イテレーション変数と続行およびブレーク ステップを使用することで、他のループ ステップと同様にループの実行を制御できます。

抽出ステップを使用すると、通常どおりにアプリケーション ツリーから返されたデータを抽出できます。

データベース照会ステップは、[データベース] および [ループ] カテゴリに属します。

プロパティ

以下のプロパティを使用して「データベース照会」ステップを設定します。

データベース マッピング
「ロボットの呼び出し」ステップで指定されたデータベースからデータベース マッピングを選択します。

SQL ステートメント
`select * from departments;` など、選択したデータベースで実行する SQL ステートメントを入力します。

ロボットは入力したステートメントを評価しません。事前に SQL ステートメントを確認してください。そうしないとステップを実行してステートメントの有効性を確認するしかありません。

アプリケーション名
ステップで開くアプリケーションの名前を設定します。この必須パラメータで指定された名前は、アプリケーションの名前属性とタイトル属性で使用されます。

エラーを回避するために、ロボット内のアプリケーションにはそれぞれ異なるアプリケーション名を割り当てます。

イテレーション変数 **(**オプション**)**
イテレーション データを格納する変数を指定します。

タイムアウト
選択した場合、SQL ステートメントの送信タイムアウトを秒単位で指定できます。値はゼロより大きい必要があります。タイムアウトは DBMS を通じて強制され、タイムアウトに達するとデータベース固有のエラー メッセージとともに `DeviceIssue` 例外がスローされます。

それぞれの DBMS タイプおよび JDBC ドライバーに応じて、タイムアウトの処理方法が異なる場合があります。たとえば、一部のシステムにおいては、特定のステートメントの途中でのタイムアウトの発生が許可されていないこともあります。

ステートメント タイプ
ステートメント タイプの詳細については、SQL 実行を参照してください。

コンポーネント アクション

長いバイナリ (BLOB) と長いテキスト (CLOB) の各データベース タイプの列は、アプリケーション ツリーには読み取られません。代わりに、[長いバイナリ値を読み取る] または [長いテキスト値を読み取る] コンポーネント アクションをそれぞれ使用して、それらのタイプの列から値を読み取る必要があります。これらのアクションでは、長いデータを読み取る変数と、null データが読み取られた場合にロボットが例外をスローするかどうかを示すフラグを指定できます。このフラグは、デフォルトではオフであり、null データが空のデータとして扱われることを示しています。

データベース データ登録

このステップを使用して、変数のデータをデータベースに保存します。このステップは、ベーシック エンジン ロボットのデータベース データ登録ステップと同じように機能します。データベース内のレコードを変更するには、SQL 実行ステップを使用します。選択したデータベースからデータを抽出するには、データベース照会ステップを使用します。

このステップでは、書き込み先のテーブルがタイプと構成の定義と一致することが検証され、いずれかが一致しない場合はエラーが生成されます。データベースにアクセスできる限り、ロボットの編集中は警告も表示されます。この場合は、ステップを右クリックし、[テーブルをデータベースに (再) 生成] を選択して SQL ダイアログ ボックスを開き、Design Studio からテーブルを更新できます。

デバッグ モードで、[次の場合停止] で [値が返されるか、または値が保存されます] が選択されている場合、[データベース データ登録] ステップを使用するロボットは、呼び出し元のベーシック エンジン ロボットに実行が戻るまで停止しません。つまり、呼び出されたロボットは、デバッガーで停止する前にデータベースに複数の値を保存する可能性があります。

ロボットのワークフローまたは状態ペインで値を右クリックし、**[ステップを挿入]** > **[データベース データ登録]** の順にクリックすると、このステップを直接追加できます。

プロパティ

データベース マッピング
「ロボットの呼び出し」ステップで指定されたデータベースからデータベース マッピングを選択します。

変数
値の読み取り元の変数を選択します。この変数はコンプレックス タイプである必要があります。

キー
データベース内の値のキーを計算する方法を選択します。[タイプ内に定義されたキー] を選択すると、タイプ エディターで [データベース キーの一部] とマークされている属性がキーの計算に使用されます。それ以外の場合は、[計算されたキー] を選択するときにキーを計算する式を指定します。

監査データ
選択すると、値を含む監査データ フィールドが含まれます。詳細については、「データベースへのデータ格納」を参照してください。このステップは、監査データが選択されている限り、ベーシック エンジン ロボットの 「データベース データ登録」ステップのみと互換性があります。

デバイスからの切断

このステップでは、デバイスへの接続ステップで接続したリモート デバイスから切断します。

プロパティ

デバイス
切断するデバイス名を選択します。リストには、動的リファレンス名のみが表示されます。

Document Transformation

Document Transformation ステップでは、画像およびテキスト ドキュメントから情報を抽出して使用することができます。Kofax RPA Document Transformation Service は、.png、.jpeg、.jpg、.tif、tiff、.pdf、および .txt ファイルを処理できます。複数のドキュメントは、.zip アーカイブまたはファイルを含むフォルダへのパスのいずれかの形で送信することができます。Kofax Transformation でドキュメント分離機能を使用すると、Kofax RPA は、Document Transformation ブラウザで移動できる複数のドキュメントを受信します。

Kofax RPA Document Transformation Service は、Sentiment プロジェクトを使用して自然言語処理 (NLP) リクエストを処理し、テキストのムード (ポジティブまたはネガティブなど) を検出し、会社名、個人名などのエンティティを抽出することもできます。Sentiment プロジェクトを使用して顧客レビューを処理し、顧客がサービスに満足しているかどうかを理解できます。また、記事内の会社に関するすべての言及を検索するためにも使用できます。Sentiment プロジェクトは、KTT バージョン 7.0.0 以降で使用できます。詳細については、事前定義済みのプロジェクトの Sentiment プロジェクトを参照してください。

Document Transformation ワークフロー

Document Transformation アクションでは、選択したプロジェクトを使用してグラフィカル ドキュメントまたは PDF ドキュメントを処理します。プロジェクトは、OCR やその他の指定された操作を実行してドキュメントを処理および変換するモジュールです。

処理結果はロボットに戻され、レコーダー ビューの Document Transformation Browser で開きます。サービスは、抽出されたすべての情報を含む要素ツリーを形成します。複数ページのドキュメントでは、Document Transformation ブラウザのツールバーの [前へ] および [次へ] ボタンを使用してページを移動できます。詳細については、「DT ブラウザとアクション ステップ」を参照してください。

ツリーの要素には、プロジェクトで定義された OCR 結果やその他の抽出結果の信頼度が含まれています。confidence 属性には、0 から 1 までの値を含めることができます。一番高い信頼度は 1 です。

<word text="Engineer" confidence="0.981818" der_x="342" der_y="176" der_width="56" der_height="13"/>

der_x などの拡張属性を使用して要素を見つけることができます。これはファインダーで使用できます。

変換されたドキュメントを エディターで開くと、変換結果の検証を実行するかどうかを決定できます。検証なしで変換結果に満足できた場合は、ドキュメントのデータを抽出して使用することができます。

検証は Document Transformation Thin Client によって実行されます。指定された Thin Client にドキュメントを送信するには、Document Transformation Browser の [] をクリックします。一意の URL が生成され、ロボットに返されます。ロボットは URL を抽出し、それを使用して電子メールなどを介して検証ユーザーにドキュメントを送信します。検証ユーザーは URL をクリックし、資格情報を入力します。その後、抽出されたデータを含むドキュメントが開きます。検証ユーザーは、変換されたドキュメントを調べ、必要であれば、ドキュメント内の抽出された情報を修正します。

ドキュメントの検証時に、ユーザーはオンライン ラーニングを有効にして、同じようなドキュメントでのフィールド認識の確率を上げることができます。この機能は、請求書などのサンプル ドキュメントのレイアウトの記憶に基づいています。自動フィールド入力を使用、ドキュメントに正しい値を手動で入力または選択することにより、ユーザーはナレッジ ベースに貢献します。これによってユーザーが次回同様のドキュメントを表示する際に、抽出結果が改善されます。

検証が終了すると、検証ユーザーはそのドキュメントを有効であるとマークします。有効であるとマークされたドキュメントは、**Document Transformation** アクションの [コールバック] オプションで指定された口ボットの引数として使用されます。

Document Transformation を開くステップのメタ データを含めるには、[ステップを移行] を実行した後に口ボットを閉じて開きます。

ステップ プロパティ

アクション

Kofax RPA Document Transformation Service を使用して、実行するアクションを選択します。

サービス **URL**

必要に応じて、Document Transformation Service を実行しているコンピュータの URL とポートを指定します。サービスがローカルにインストールされている場合は、このフィールドに `localhost` と入力します。URL には、`http://` または `https://` プレフィックスが含まれている必要があります。`https` を使用する場合、Web ホスティング サービスにはよく知られている認証機関によって受け入れられた証明書が必要です。

プロジェクトのタイプ

- デフォルトのプロジェクト: このオプションは、事前定義済みのプロジェクトのセットを提供します。事前定義済みのプロジェクトを参照してください。
- カスタム プロジェクト: このオプションを選択する場合、[カスタム プロジェクトのパス] でドキュメントを処理するプロジェクトへのパスを指定します。

  [カスタム プロジェクトのパス] は、Document Transformation Service サーバーを参照します。絶対パス、または`Document Transformation\Projects` フォルダからの相対パスのいずれかにすることができます。

ドキュメント ソース

口ボットが処理するドキュメントを検索する方法を選択します。

- ローカル ファイル: [ファイル名] で処理するドキュメントのパスを入力します。口ボットを実行しているコンピュータからアクセス可能なイメージ ファイル、.zip アーカイブ、ファイルを含むフォルダ、またはその他のサポートされている形式のファイルのいずれかのフル パスを指定します。
- 口ボット ファイル システム: 構成済みのファイル システムへのパスとファイル名 (**Myshare/doctotransform.pdf** など) を入力します。ファイル システム名は、Management Console 内の[口ボット ファイル システム] セクションでの指定に対応している必要があります。
- バイナリ変数: ドキュメントを含むバイナリ変数を指定します。

複数のドキュメントのパスを指定すると、Document Transformation ブラウザのツールバー ボタンを使用して複数のドキュメント間を移動することができます。

メタデータ

追加のデータを Document Transformation Service に渡すには、このオプションを選択します。

このデータは XValues として入力ドキュメントに追加されるため、Document Transformation Service プロジェクトで使用することができます。プロジェクトは通常、このデータを使用して分析を調整または制御します。このオプションの一般的な使用例としては、言語設定または顧客 ID などが挙げられます。XValues は、処理されたドキュメントが Document Transformation Service から返された後に、デバイス ツリーで使用することができます。

特定のプロジェクトでサポートされている値を確認する場合は、Document Transformation Service プロジェクトの開発者に問い合わせてください。

複数のキーと値のペアを [メタデータ] プロパティに追加できます。
いずれかの [キー] がリストに複数回表示されている場合は、最後の [値] が使用されます。

検証 **URL**
シン クライアント サービスの URL を指定するには、このオプションを選択します。このプロパティは、処理されたドキュメントを検証の目的で送信するために必要です。URL は、Document Transformation Service の `ValidationService` プロパティで指定します。URL は次のようになります。

```
http://localhost:50082
```

コールバック
ドキュメントの検証後にシン クライアント サービスで呼び出す必要がある口ボットを指定するには、このオプションを選択します。検証が完了すると、口ボットは実行のために Management Console のキューに登録されます。

- [口ボット プロジェクト]: 呼び出す口ボットが存在するプロジェクト(デフォルトのプロジェクトなど)を指定します。
- [口ボット名とパス]: 口ボットがプロジェクトのフォルダ内にある場合は、その口ボットの名前とパス(`MyRobot.robot`、`folder/subfolder/MyRobot.robot` など) を指定します。

検証が完了すると、[コールバック] オプションで指定された口ボットに返されるドキュメントには、ドキュメントのバッチを検証したユーザーの名前が含まれます。この情報は、KDTS-ValidatingUser という名前の XValue として提供されます。

コールバック ロボットをキューに格納する Management Console を見つけるために、Document Transformation ステップが設定された口ボットは、ステップが実行される RoboServer 用に設定された Management Console URL を使用します。Design Studio で口ボットを実行した場合、この口ボットは Design Studio の設定 で「プライマリ」とマークされた Management Console の URL を使用します。組み込みの Management Console で口ボットを実行した場合、この口ボットは `-mcUrl` パラメータで設定された URL を使用します。

これらの設定済み URL では、Management Console を実行しているコンピュータのホスト名または IP アドレスを使用する必要があります。Document Transformation サービスが Management Console にアクセスできず、コールバック ロボットがキューに格納されなくなるため、「localhost」を使用しないでください。

事前定義済みのプロジェクト

Kofax RPA によってインストールされた KTT Project Builder のカスタム プロジェクトおよび Kofax RPA によって提供された変換プロジェクトを編集できます。Project Builder を開くと、そのドキュメントにアクセスできます。

バーコード プロジェクト
このプロジェクトの目的は、ドキュメントからすべてのバーコードを抽出することです。
バーコード プロジェクトの設定を変更するには、次のステップを実行します。

1. Kapow_Barcodes.fpr プロジェクト ファイルを見つけます。
2. このファイルを Project Builder で開きます。

3. 左側のプロジェクト ツリーでクラス [デフォルト] を選択します。
4. 目のシンボルをクリックして詳細を開きます。
5. [ロケータ] で、BL バーコード ロケータをダブルクリックします。デフォルトでは、ロケータはバーコード タイプを自動検出するように設定されています。
6. [タイプ] の下にある [自動検出] オプションをオフにして、特定のタイプを選択します。
7. デフォルトでは、ロケータは方向を自動検出するように設定されています。[方向] の下にある [自動検出] オプションをオフにして、特定の方向を選択します。
8. デフォルトでは、ロケータはドキュメントのすべてのページのバーコードを検索するように設定されています。バーコードの検出対象を特定のページのセットに制限するには、[領域] タブを選択し、[ロケータを有効化する対象] の設定を変更します。
9. プロジェクトの編集が終了したら、すべてのダイアログ ボックスを閉じ、[プロジェクト] タブの [プロジェクトを保存] をクリックします。

請求書プロジェクト **(**請求書の消費税および請求書の **VAT)**

これらのプロジェクトは米国からの請求書を抽出するように設計されており、消費税もサポートしています。これらのプロジェクトでベンダーを適切に抽出するには、ERP マスター データを設定する必要があります。ベンダーおよび内部会社のマスター データは、csv ファイルとして提供する必要があります。プロジェクトには、会社固有のベンダーに適合できる `vendors.csv` ファイルと `internal_venders.csv` ファイルが含まれています。

マスター データを指定して設定するには、以下の手順を実行します。

前提条件

- ベンダー ファイルは、`Vendors.csv` というセミコロンで区切られたドキュメントです。ファイルには次の列が必要です。
  - VendorID (必須)
  - CompanyCode (オプション)
  - Name (必須)
  - Street (必須)
  - City (必須)
  - ZIP (必須)
  - PostBox (オプション)
  - Country (必須、2 文字の国コード)
  - FIDNumber (オプション)
  - Phone (オプション)
  - Fax (オプション)
  - URL (オプション)
  - Email (オプション)
- 内部のベンダー ファイルは `Vendors_Internal.csv` という名前にする必要があります。これは、`Vendors.csv` と同じ列を持つセミコロンで区切られたファイルです。内部ベンダーは、顧客のエンタープライズ内部のベンダーです。このファイルは、ベンダー結果から、通常の外部請求書の請求先住所と混同しやすい内部ベンダーを除外するために使用されます。

1. `Kapow_Invoices_SalesTax.fpr` プロジェクト ファイルまたは `Kapow_Invoices_VAT.fpr` プロジェクトファイルを見つけて、Project Builder で開きます。

2. [プロジェクト設定] を開きます。
3. [データベース] タブを開きます。
   パスが正しくないため、2 つの Fuzzy データベース項目に赤色のフラグが付いていることに注意してください。
4. [**Vendors**] をダブルクリックします。
5. ネットワーク共有上の `Vendors.csv` のパスを選択します。
6. [**OK**] をクリックしてダイアログ ボックスを閉じます。ファイルがインポートされます。
7. **Vendors_Internal** をダブルクリック。
8. ネットワーク共有上の `Vendors_Internal.csv` のパスを選択します。
9. [**OK**] をクリックしてダイアログ ボックスを閉じます。ファイルがインポートされます。
10. ファイルの編集が終了したら、すべてのダイアログ ボックスを閉じ、[プロジェクト] タブの [プロジェクトを保存] をクリックします。

オンライン ラーニング
デフォルトでは、同様のドキュメントでのフィールド認識の確率を上げるために役立つオンライン ラーニングは請求書プロジェクトでのみ有効にできます。トレーニング ドキュメントが保存されるフォルダへのパスを指定する場合、そのフォルダがすでに存在していることを確認します。フォルダが存在しない場合、作成することを求める通知を受け取ります。続行するには、[はい] をクリックします。

言語プロジェクト
このプロジェクトの目的は、ドキュメントが記述される言語を特定することです。
このプロジェクトは設定可能ではありません。

OCR プロジェクト
このプロジェクトの目的は、ドキュメントのフル テキスト OCR の結果を返すことです。デフォルトではこのプロジェクトに検証プロセスが含まれていないことに注意してください。
OCR 認識言語をデフォルト (英語) から変更するには、以下の手順を実行します。

1. Kapow_OCR.fpr プロジェクト ファイルを見つけます。
2. このファイルを Project Builder で開きます。
3. [プロジェクト設定] をクリックします。
4. [プロジェクト設定] ダイアログ ボックスで、[認識] タブを選択します。
5. [**FineReader**] ページ プロファイルを選択します。
6. 希望の言語を確認します。
7. すべてのダイアログ ボックスを閉じ、[プロジェクト] タブの [プロジェクトを保存] をクリックします。

US 住所抽出プロジェクト
このプロジェクトの目的は、ドキュメントからすべての US 住所を抽出することです。
このプロジェクトは設定可能ではありません。

Sentiment プロジェクト
このプロジェクトの目的は、テキストの雰囲気 (ポジティブまたはネガティブなど) を推測し、会社名、個人名などのエンティティを識別することです。変換されたドキュメントでは、ムードは -1 から 1 までの数字で `Sentiment` フィールドに表示されます。-1 は完全にネガティブで、1 は完全にポジティブです。たとえば、`0.257545` はわずかにポジティブなテキストを表します。
デフォルトでは、プロジェクトは英語のテキストを処理します。Sentiment プロジェクトの言語バンドルは、3 つの `.msi` インストーラーで個別に分配されます。

- Kofax NLP 規定欧米語 バンドル: 英語、フランス語、ドイツ語、ポルトガル語、スペイン語
- Kofax NLP 規定欧米語以外の欧米語 バンドル: オランダ語、イタリア語、ルーマニア語
- Kofax NLP 追加の言語 バンドル: 日本語、韓国語、標準中国語

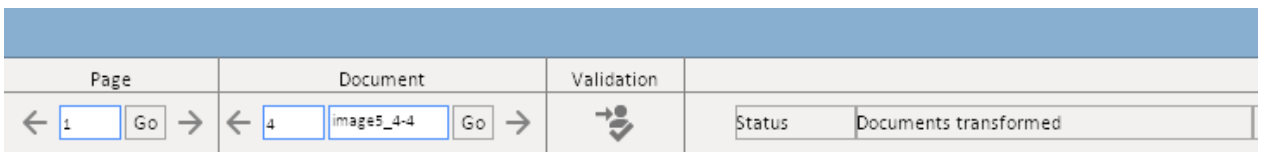
言語バンドルはデフォルトではインストールされません。使用可能な言語を使用するには、該当する言語バンドルをインストールします。たとえば、英語を使用するには、Kofax NLP Western Default Language Bundle をインストールします。
バンドルは Windows プログラムとしてインストールされ、オプションはありません。言語バンドルを削除するには、コントロールパネルから [プログラムと機能] または [アプリと機能] を開き、バンドルを選択して [アンインストール] をクリックします。
認識言語をデフォルト (英語) から変更するには、以下の手順を実行します。

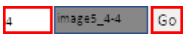
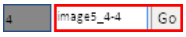
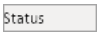
1. Project Builder で Sentimentプロジェクトを開きます。
2. [プロジェクト設定] をクリックします。
3. [プロジェクト設定] ダイアログ ボックスで、[プロパティ] ボタンをクリックします。
4. デフォルトの英語オプションをクリアします。
5. 言語を選択します。
6. すべてのダイアログボックスを閉じます。
7. プロジェクト ツリーで、**Default Project Class** の定義を選択します。
8. スクロール ダウンして、言語リストから目的の言語を選択します。
9. [プロジェクト] タブの [プロジェクトを保存] をクリックします。

カスタマイズ済みのプロジェクト
このオプションを選択するときは、ドキュメントを処理するプロジェクトのパス (`c:\rpa\ocr` など) を [プロジェクト名] プロパティに指定します。プロジェクト リンクは、Design Studio を実行しているコンピュータではなく、Document Transformation ホスト上のローカルにアクセス可能なフォルダである必要があります。

DT ブラウザとアクション ステップ

DT (Document Transformation) ブラウザは変換結果を表示し、ドキュメント内の抽出されたデータを処理するのに役立ちます。次の表は、DT ブラウザのツールバーの要素について説明します。



ボタン	説明
複数ページのドキュメント内を移動するのに役立つ [ページ] セクション	
	複数ページのドキュメントで 1 つ前のページに戻ります。
	複数ページのドキュメントで 1 つ先のページに進みます。
	複数ページのドキュメントで指定のページに移動します。
複数または分割されたドキュメント内を移動するのに役立つ [ドキュメント] セクション	
	前のドキュメントに移動します。
	次のドキュメントに移動します。
	番号で指定されたドキュメントに移動します。
	名前で指定されたドキュメントに移動します。 • 通常のドキュメントの場合は、ファイル名を拡張子なしで指定します。例：mydocument.pdf ファイルの場合、mydocument で指定。 • 分割されたドキュメントの場合は、拡張子なしのファイル名の後にアンダースコア (_) とページ範囲を接尾辞として付加します。たとえば、4 ページから成る mydocument.pdf ファイルが 2 ページに分割されている場合、分割されたドキュメントはそれぞれ mydocument_1-2 および mydocument_3-4 という名前にします。
ドキュメントの検証に役立つ [検証] セクション	
	指定された Document Transformation Thin Client サーバーに手動で検証するドキュメントを送信します。
	変換されたドキュメントの状態 (エラーの説明がある場合)。

アプリケーション アクション

ツールバーに表示されるすべてのアクションは、DT ブラウザのタブを右クリックしたときに表示されるコンテキスト メニューからも利用できます。次の表に、コンテキスト メニューからのみ使用できるアクションの一覧とその説明を示します。

アクション ステップ	説明
前のページ	前のページに移動します。
次のページ	次のページに移動します。
ページに移動	複数ページのドキュメントで指定のページに移動します。
ページ数	現在アクティブなドキュメントのページ数を取得します。
前のドキュメント	前のドキュメントに移動します。

アクション ステップ	説明
次のドキュメント	次のドキュメントに移動します。
ドキュメントに移動	バッチ内の指定されたドキュメントに移動します。
ドキュメント数	バッチ内のドキュメントの数を取得します。
閉じる	アプリケーション ウィンドウを閉じます。
送信	処理のためにドキュメントを DTS サーバーに送信します。
ドキュメントを取得	バッチから特定のドキュメントを抽出します。 Document Transformation Service プロジェクトがドキュメント分離を使用するように設定されている場合、このアクションを使用してサブドキュメントを抽出できます。 1. ステップのプロパティで、抽出するサブドキュメントの名前またはそのインデックス (バッチ内の番号) を選択します。[インデックス] オプションは [名前] オプションよりも優先されるため、両方を指定した場合は、ドキュメントの選択にインデックスが使用されます。 2. ドキュメントを抽出するファイル形式を選択します。[画像を取得] (.tif) および/または [**XDoc** を取得] (.xdc)。 3. 結果を保存するバイナリ形式の変数を指定します。 その後、たとえば、ドキュメントを作成アクションで Kofax TotalAgility に送信したり、ファイルの書き込みステップを使用してローカルまたはリモート コンピュータ上のファイルにデータを書き込んだりするなど、さまざまな方法でこのバイナリ データを使用できます。 たとえば、ファイルの書き込みステップを使用する場合は、抽出されたドキュメントを含むファイルのファイル名については次のことを考慮します。 • 「画像を取得」のみを選択した場合は、拡張子 **.tif** を使用する。 • 「XDoc を取得」のみを選択した場合は、拡張子 **.xdc** を使用する。 • 両方の形式を選択した場合は、拡張子 **.zip** を使用する。この場合、抽出された 2 つの形式のドキュメントを含むアーカイブが作成されます。 .xdc 形式は、Document Transformation Service インストールに含まれる XDoc Browser アプリケーションで読み取ることができます。

アクション ステップ	説明
検証ユーザーの設定	ドキュメントを検証するためのユーザーまたはグループを指定するステップを挿入します。ユーザーまたはグループは、次のいずれかの形式で指定できます。 • 名前 • ドメイン\名前 • name@domain.com 検証用のユーザーまたはグループを割り当ててドキュメントを送信すると、指定したユーザーまたはグループにのみドキュメントが表示されます。

コンポーネント アクション

アクション	説明
フィールド値を取得	ドキュメント フィールドの値を、指定した変数に抽出します。

電子メール

このステップは、電子メール サーバーでフォルダを開いてそれらをツリーに表示し、電子メールにアクセスする場合に役立ちます。開いたフォルダは、アプリケーション ツリーに表示され、レコーダー ビューにテーブルとして表示されます。メッセージごとに、このテーブルには次のプロパティが列として一覧表示されます。

- 送信者: John Doe <j.doe@organization.com> という形式の送信者。
- 件名: メッセージの件名。
- 受信済み: 2023-05-17T15:27:33+01:00 (ISO-8601) という形式の、メッセージが受信された日付。
- サイズ: バイト単位のメッセージのサイズ。
- 添付ファイルあり: メッセージに添付ファイルがあるかどうかを示します
- 既読: メッセージが既読としてマークされているかどうかを示します。
- 添付ファイルの数: 添付ファイルの数。

メッセージのあるツリーとテーブルには、メッセージの移動や削除など、ステップの実行後に発生する可能性のある電子メール フォルダへの変更は反映されません。

「電子メール フォルダを開く」アクションで電子メール ステップをステップ オーバーする場合、電子メール フォルダのロードを停止することはできません。

また、[電子メール] ステップを使用すると、電子メールを HTML として開くことで、電子メールを開封せずに電子メールの添付ファイルを抽出できます。

プロパティ

アクション: 電子メール フォルダを開く

このアクションは、電子メール サーバーに接続し、選択した電子メール フォルダを開きます。

アプリケーション名

アプリケーションの名前を指定します。

エラーを回避するために、ロボット内のアプリケーションにはそれぞれ異なるアプリケーション名を割り当てます。

プロトコル

電子メール サーバーへの接続に使用する電子メール プロトコルを選択します。選択肢は「imap」、「pop3」、およびそれらのセキュアなバリアント「imaps」、「imaps+tls」、および「pop3s」です。オプション「imap+tls」は、便宜的な TLS を必要とする電子メール サーバー用です。

電子メール サーバー

電子メール サーバーの名前またはアドレスを指定します。

電子メール サーバー ポート (オプション)

電子メール サーバーのポート番号を指定します。指定しない場合は、選択したプロトコルの標準ポートが使用されます。

アカウント名とパスワード

電子メール サーバーにログインするためのクレデンシャルを入力します。

メールボックスのパス (オプション)

電子メール サーバーで開くフォルダの名前。ネストされたフォルダのパスの形式である場合があります (この場合、パス内の要素はスラッシュで区切る必要があります)。受信電子メールが保存されているフォルダを開くには、パスや名前を指定しないでください。

例: `Junk E-Mail、Subscriptions/Mailing lists.`

POP3 はフォルダをサポートしていないため、POP3 または POP3S プロトコルを使用する場合は、このフィールドを空白のままにします。

開いているフォルダは、アプリケーション ツリーに表示されます。フォルダには、0 個以上のメッセージを含めることができます。各メッセージには、フォルダ内の場所を示す "der_index" プロパティがあります。インデックスは 1 から始まる連続する値です。

添付ファイルを取得するには、[添付ファイルを取得] コンポーネント アクションを選択します。

アプリケーション アクション

次のアプリケーション アクションが使用できます。

アクション	説明
電子メール フォルダを閉じる	アプリケーションを閉じます。

コンポーネント アクション

次のコンポーネント アクションが使用できます。

アクション	説明
メッセージをコピー	**IMAP** および **IMAPS** プロトコルの場合。選択したメッセージを指定したターゲット フォルダにコピーします。このアクションを使用できるかどうかは、電子メール サーバーでのコピー操作のサポートに応じて異なります。
メッセージを削除	電子メール メッセージを削除します。

アクション	説明
メッセージの内容を取得	RFC 822 形式のメッセージとヘッダーを含む文字列値を返します。
メッセージを HTML として開く	Chromium 組み込みブラウザを使用して、電子メールを HTML Web ページとして開きます。開いている電子メール テーブル内の行を操作します。
メッセージを移動	**IMAP** および **IMAPS** プロトコルの場合。選択したメッセージを、指定したターゲット フォルダに移動します。このアクションを使用できるかどうかは、電子メール サーバーでの移動操作のサポートに応じて異なります。
メッセージを既読にする	**IMAP** および **IMAPS** プロトコルの場合。メッセージに既読または未読のフラグを立てます。 チェック ボックス **[既読]** をオンにしてメッセージを既読としてマークするか、チェック ボックスをオフにしてメッセージを未読としてマークします。
添付ファイルを取得	添付ファイルのバイナリ データを取得し、変数に保存します。このアクションは、添付ファイルが大きい場合に便利です。 必要なデータを取得するには、次のプロパティを設定します。 • インデックス: 抽出する添付ファイルの、0 から始まるインデックス。 • 添付ファイル データ: 結果の添付ファイル データを格納するバイナリ変数。

POP3 または **POP3S** プロトコルに関する制限事項

- フォルダはサポートされていません。[電子メール フォルダを開く] アクションのフォルダ名は、空または「INBOX」にする必要があります。
- 一部の POP3 サーバーには、他のインターフェイスの POP3 接続を介して行われた変更が反映されません。たとえば、ロボットが POP3 プロトコルを使用して Gmail アカウントからのメッセージを削除した場合でも、Web ブラウザ インターフェイスには引き続きそのメッセージが表示されることがあります。ロボットの場合、メッセージは削除されます。
- メッセージを削除ステップを行うとメッセージに削除のマークが付きますが、アプリケーションを閉じるまでメッセージはサーバー上に残ります。
- ほとんどの POP3 サーバーでは、メッセージがダウンロードされるとメッセージは削除されます。
- 「既読」または「未読」フラグは設定できません。

アクション**:** 電子メールを **HTML** として開く

このアクションにより、Chromium 組み込みブラウザを使用して電子メール メッセージが HTML Web ページとして開きます。MIME 形式の電子メール ファイル (`.eml`) のみがサポートされます。このアクションではテキスト値を使用します。

[電子メール] ステップの前に、[ファイルの読み取り] ステップと [割り当て] ステップを挿入します。次のアクションを実行します。

1. [ファイルの読み取り] ステップを設定して、`.eml` ファイルからバイナリ データを読み取り、保存します。

2. [割り当て] ステップを設定して、バイナリをテキストに変換します。
   エクスプレッションの例を次に示します。`=text(mail_bin, "utf-8")`。ここで、`mail_bin` は [ファイルの読み取り] ステップで指定したバイナリ変数です。

プレーン テキストや HTML などの電子メールのテキスト バリアントは、結果として得られる Web ページにレンダリングされます。Web ページの上部に、ヘッダー (件名など) を含むテーブルが表示されます。その下のテーブルには、利用可能な添付ファイルが一覧で表示されます。

アプリケーション名
アプリケーションの名前を指定します。

電子メール メッセージ
開く電子メールを、先頭に等号を付けたテキスト値として指定します。

すべてのヘッダーを含める
結果として得られる Web ページのすべてのヘッダーを表示する場合は、このオプションを選択します。
デフォルトでは、ステップには送信者、宛先、件名、日付、メッセージ ID などの基本的なヘッダー情報のみが含まれます。

ヘッダー値をデコード
ヘッダー値を Unicode にデコードする場合に選択します。これは電子メール クライアントに共通するもので、英語以外のテキストを表示する場合に役立ちます。
デフォルトでは、このオプションは選択されています。

インライン **CID** 画像タグ
コンテンツ指定の画像をタグとしてインライン化するかどうかを指定します。電子メールのテキスト形式にのみ適用されます。
デフォルトでは、タグはインライン化されます。

添付ファイルをインライン化
添付ファイルを完全にインライン化するかどうかを指定します。このオプションを選択すると、添付ファイル テーブルには、ブラウザを使用してデータをダウンロードするために使用できるデータ URL が含まれます。
添付ファイルを表示するには、その URL を右クリックし、[ターゲット抽出] コンポーネント アクションを選択して、バイナリ変数を指定してから、[ステップ オーバー] をクリックします。
大きな添付ファイルを表示するためにこのオプションを使用することはお勧めしません。

アクション**:** 電子メールから添付ファイルを抽出
このアクションにより、電子メールを開かずに電子メールから添付ファイルを抽出し、バイナリ変数に保存します。このアクションを使用して、大きな添付ファイルを抽出します。
前のアクションで説明したように、[ファイルの読み取り] と [割り当て] ステップを設定します。

電子メール メッセージ
開く電子メールを、先頭に等号を付けたテキスト値として指定します。
MIME 形式の電子メール ファイル (`.eml`) のみがサポートされます。

インデックス
抽出する添付ファイルの、0 から始まるインデックス。

結果: 添付ファイル データ
結果の添付ファイル データを格納するバイナリ変数を指定します。

テキストを入力

このステップでは、ロボットがテキストをテキスト フィールドに入力できます。必要なテキストをステップの [テキスト] フィールドに直接入力したり、変数からのテキストを使用したりできます。これはアプリケーション レベルのステップで、[アプリケーション] タブを右クリックすると利用できます。テキスト フィールドをクリックしない場合、またはテキストを入力する前に適切なファインダーを作成していない場合は、このステップにより、アプリケーション ツリーの利用可能な最初のテキスト フィールドにテキストが挿入されます。

選択したフィールドを右クリックし、ショートカット メニューの [テキストの置き換え] を選択することで、そのフィールド内にテキストを入力できます。このコマンドでは、ファインダーと選択したフィールド内のテキストを置き換えるのに必要なすべてのアクションを含む入力ステップが作成されます。

以下のステップで Excel 上の操作をするときは、リモート デバイス上の Excel で「テキストを入力」ステップを使用する際に、Excel のスプレッドシートを編集モード (「編集」という語句が Excel のプログラム ウィンドウの左下隅に表示される状態) のままにしないでください。編集モードを終了するには、以下の手順を実行します。

- キー プレス ステップを使用して Enter キーを押します。Excel の編集モードが終了し、現在のセルの下のセルが直接選択されます。
- キープレス ステップを使用して Tab キーを押します。編集モードが中止され、現在のセルの右のセルが選択されます。
- 別のセルをクリックします。
- キープレス ステップを使用して F2 キーを押します。

詳細については、 Microsoft のドキュメントを参照してください。

仮想入力ドライバーでテキストを入力します
自動化されたデバイスで仮想入力ドライバーを有効にすると、Windows のデバイス オートメーションの [テキストを入力] ステップでは、テキストの入力にこのドライバーが自動的に使用されます。
テキストはハードウェア キーボードを介して入力され、オペレーティング システムによるキーボード レイアウトにより動作が決まります。物理キーボードによるテキストの入力は、アプリケーションのアクティブなキーボード レイアウトを使用してのみ可能です。すべての Unicode 文字を入力することはできません。
仮想入力ドライバーを有効または無効にする方法については、『Kofax RPA Desktop Automation サービス ガイド』を参照してください。

プロパティ

名前
ステップの名前。

ファインダー
[デバイス]: オートメーション デバイスの名前を選択します。

[アプリケーション]: アクションが実行されるアプリケーションの名前を指定します。

テキスト
テキストを直接入力するか、テキストのある変数を指定します。変数名の前に等号を付ける必要があります (例：`=EnterTextVariable`)。

エクスプレッションを評価

このステップを使用できるのは、値を抽出、クリップボードから抽出、ツリーを XML として抽出、値を変換、DateTime を抽出、および DateTime の書式設定のコンテキストのみです。このステップには、特定の値での使用が推奨される変換関数のリストが含まれています。特定の値に対して推奨される関数リストは、値の種類に応じて制限されます。変換関数を指定するには、ステップのコンテキスト メニューで **[エクスプレッションを評価]** をクリックし、次のいずれかを実行します。

- 推奨関数のリストから必要な関数を選択します。
- [プレーン] をクリックして、必要な関数を手動で入力します。たとえば、整数を表すテキストを抽出して整数変数に格納する必要がある場合は、エクスプレッション `$initial.integer()` を使用できます。

サポートされている関数と例の詳細については、エクスプレッションを参照してください。

Excel

組み込み Excel ドライバーを使用して Excel ワークブックで操作を実行するには、このステップを使用します。

ナビゲーションのヒント、アクション メニュー、グラフ シートの情報については、ロボットでの Excel の使用 を参照してください。

プロパティ

アクション
ファイルを作成
新しい Excel ワークブックを作成する場合に選択します。

ファイルを開く
既存の Excel ワークブックを開く場合に選択します。

ファイル アクセス
ソースへのアクセス方法を選択します。

直接アクセス
ファイルからソースを選択します。

RFS 経由
リモート ファイル共有ソースからソースを選択します。

変数から
変数からソースを選択します。

ワークシートのパス
Excel ワークブックへのフル パスを次のように指定します。

- 直接アクセスまたは RFS ソースの場合は、ワークブックへのフル パスを入力します。
  例: `c:/documents/myworkbook.xlsx`
- 変数ソースの場合は、データを含むバイナリ タイプの変数を選択します。

表示可能エリア
Excel ワークブックを開いたときに表示する行と列の数を指定する場合に選択します。
デフォルトは 45 行および 20 列です。指定した列数は最小の値です。
Excel ドライバーは、指定した数以上の列をロードし、画面全体の幅を埋めます。
この領域の行とセルの高さまたは幅が 0 に設定されていてレコーダー ビューに表示されない場合は、これらに `Hidden="true"` 属性が設定されています。

拡張ツリー
ファインダーで使用するために、可視領域外のセルをアプリケーション ツリーに含める場合に選択します。
可視領域外のセルを操作するには、アプリケーションとコンポーネントのアクションを使用する必要があります。
このオプションを使用すると、Excel が「使用中」であると見なす領域内のすべてのセルがアプリケーション ツリーに含まれます。これらのセルが可視領域の外側にある場合は、`Visible = "false"` 属性が設定されています。
ツリーに含まれる行と列の数を減らすには、[行] と [列] のオプションを使用します。
拡張ツリー領域は、可視領域全体を含むように動的に調整されます。

このオプションを使用すると、ツリーが非常に大きくなり、パフォーマンスが低下する可能性があります。

クリップボードから抽出

このステップでは、情報をクリップボードから変数に抽出します。このステップは、組み込みの Chromium ブラウザでの使用がサポートされていません。

プロパティ

デバイス
オートメーション デバイスの名前を選択します。

エイリアス
コンポーネントのエイリアス。

エクスプレッションを評価
オプション。値での使用が推奨される変換関数のリストが含まれています。特定の値に対して推奨される関数リストは、値の種類に応じて制限されます。変換関数を指定するには、ステップのコンテキストメニューで **[エクスプレッションを評価]** をクリックし、次のいずれかを実行します。

- 推奨関数のリストから必要な関数を選択します。
- [プレーン] をクリックして、必要な関数を手動で入力します。

サポートされている関数と例の詳細については、 エクスプレッション。

1 つ以上のエクスプレッションの評価ステップを追加できます。

現在のインを保存
抽出/変換された値を格納する、指定されたタイプの変数の名前。

1 つ以上の現現在のイン ステップを保存を追加できます。

クリップボード抽出ステップには、次の一連のアクション ステップを含めることもできます。

- 条件ステップ
- グループ ステップ
- スロー ステップ
- ログの書き込みステップ

DateTime を抽出

このステップを使用すると、Date、Time、または DateTime 値のテキスト表現を Date、Time、または DateTime 値に変換できます。このステップは、レコーダー ビューから挿入することも、ロボットのワークフローから直接挿入することもできます。ステップを挿入する際には、レコーダー ビューから、抽出する要素を右クリックし、**[ここから値を抽出]** > **[テキスト]** の順にクリックして、抽出した値を格納する DateTime 変数を選択することをお勧めします。この場合、抽出パターンやターゲット タイプの決定など、ほとんどの設定手順が自動的に実行されます。

プロパティ

ターゲット タイプ
Date、Time、または DateTime 値を抽出する結果タイプ。変換された値は、**$current** 変数に格納されます。

デフォルトの日付
[ターゲット タイプ] が [**DateTime**] に設定されている場合にのみ表示されます。パターンが時刻パターンである場合に使用するデフォルトの日付です。

- 日付なし: デフォルトの日付は指定されません。これを選択すると、ロボットが抽出する日付を発見できなかった場合は抽出が失敗します。
- 現在の日付: これを選択すると、日付が抽出されなかった場合は現在の日付が使用されます。
- 計算済み: これを選択すると、日付が抽出されなかった場合は、変数や関数などのエクスプレッションを使用して日付が計算されます。

[デフォルトの日付] プロパティは、Web ページやアプリケーションに、1 つの日付とその日付に関連付けられた複数の時刻のインスタンスが含まれている場合に便利です。この場合、日付を変数に抽出してから、日付変数をデフォルトの日付として使用して時刻のインスタンスを抽出できます。抽出した各時刻は、最初に抽出した日付に対応する日付部分を持つ DateTime 値となります。

タイム ゾーン
[ターゲット タイプ] が [**DateTime**] に設定されている場合にのみ表示されます。抽出した DateTime 値で使用するタイム ゾーンです。

- デフォルト: 抽出用のタイム ゾーンが見つかった場合は、それが使用されます。タイム ゾーンが見つからない場合は、ロボットが実行されているタイム ゾーンが代わりに使用されます。
- ゾーン **ID**:「Europe/Rome」、「America/Los_Angeles」、「CET」、「Universal」など、事前定義済みのタイム ゾーン ID のセット。
- オフセット:「Z」、「+01:00」、「-08:00」など、事前定義済みのゾーン オフセットのセット。
- 計算済み: 定数またはエクスプレッションからタイム ゾーンを取得します。このタイム ゾーンは、ゾーン ID またはオフセットのどちらかです。事前定義済みのタイム ゾーンと同じにすることもできますが、[計算済み] オプションでは、「+1」、「-08:30:15」など、追加の形式も使用できます。

パターン
Date、Time、または DateTime 値の抽出に使用できるさまざまな形式。

- 事前定義: ISO などのさまざまな日付形式に基づく標準の抽出形式。すべての事前定義済みの形式のリストについては、docs.oracle.com の Web サイトを参照してください。
- デフォルトのパターン: yyyy-M-d など、選択して使用できる一般的なパターンのセット。[ターゲット タイプ] を選択すると、このリストから使用できるパターンが変化することに注意してください。
- 計算済み: 定数として、またはエクスプレッションを使用して作成できるパターン。このオプションは、[デフォルトのパターン] オプションと同じ形式を提供します。次の表に示すように、パターンは、日付の各部分に対応する文字および文字の組み合わせで構成されます。

文字	説明	例
z	タイム ゾーン ID	`z`、`zz`、`zzz` は CET に、`zzzz` は CET と中央ヨーロッパ標準時の両方に一致します
Z	タイム ゾーン オフセット	`Z`、`ZZ`、`ZZZ` は +0100 に、`ZZZZ` は GMT+01:00 に、`ZZZZZ` は +01:00 に一致します
Y	年	`yyyy` は 2022 に、`yy` は 22 に一致します
M	月	`M` は 7 または 07 に、`MM` は 07 に、`MMM` は Jul に、`MMMM` は July に一致します
d	日	`d` は 1 または 01 に、`dd` は 01 に一致します
E	曜日	`EEE` は Mon (月) に、`EEEE` は Monday (月曜日) に一致します
h	時間 (1 ~ 12 AM または PM)	`H` は 0 または 23 に一致します。`h` と `hh` は 01 または 12 に一致し、通常は AM または PM を示す **a** パターンを必要とします

文字	説明	例
a	AM または PM	`a` は AM または PM に一致します
m	分	`mm` は 00 または 43 に一致します
s	秒	`ss` は 00 または 43 に一致します
S	秒の端数	`SSS` は 123 に一致します

注意:

- パターンには、日付の各部分を区切る「-」(ハイフン)、「/」(スラッシュ) などの文字とスペースを含めることができます。
- テキストを一重引用符で囲むと、パターンにテキストを含めることができます。
- パターンの文字数によって、結果の形式が決まります。たとえば数字の場合、文字数が 1 であれば、結果には任意の数の数字が含まれることになります (`d` の結果は 1、01、31 のいずれにもなります)。それ以外の場合、文字数は結果の長さを表します。たとえば月名の場合、3 文字は短縮形 (Jul)、4 文字は完全形 (July) を表します。

  詳細については、docs.oracle.com の Web サイトを参照してください。

ロケール
使用可能なロケールのリストが含まれています。ロケールは言語と地域を指定します。言語と地域はハイフンで区切る必要があります (「en-US」)。日付を識別する方法は言語によって異なりますが、このオプションを使用すると、月名など、言語の特性を反映できます。

タイム ゾーンはロケールの影響を受けないことに注意してください。

画像抽出

このステップでは、画面の選択領域から画像抽出し、バイナリ形式の変数で保存します。[レコーダー ビュー] で画像を選択するには、マウスボタンを押し、四角形で囲まれた選択範囲を描画します。四角形を修正するには、側面をドラッグするか、新しい四角形を描画するだけです。

プロパティ

コンポーネント
[エイリアス]: ファインダーのエイリアス。

[ベース ファインダー]: 使用するコンポーネントを指定します。

[コンポーネント]: ボタン、ウィンドウ、ペインなど、アプリケーション コンポーネント名を設定します。

[コンテンツ]: エレメントを値で検索する正規表現。このパラメータは通常、ロボットのブラウザを使用するときに使用します。

[画像]: 画像として保存される選択領域のスクリーンショット。

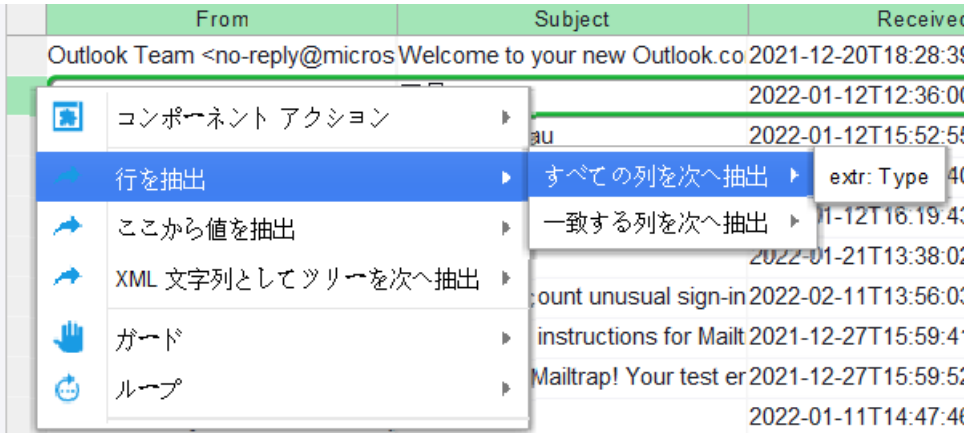
出力変数
画像を保存するバイナリ形式の変数を指定します。

表のセル エレメントから画像抽出することはできません。

行を抽出

[電子メール] ステップまたは [データベース] グループ ステップを実行するときに、[レコーダー ビュー] の行に基づいてデータを抽出できます。テーブルの列に対応する属性名とタイプを使用してコンプレックス タイプが作成され、属性名またはストレージ名のいずれかが列名と一致していることを確認します。

[電子メール] ステップまたは [データベース] 関連ステップのいずれかを実行すると、[レコーダー ビュー] で開いているテーブルの必要な行に移動できます。必要な行の左側にある四角いインジケータをクリックして、列のタイプが行の抽出用に作成したコンプレックス タイプと一致しているかどうかを確認します。必要に応じてタイプを編集します。



すべての列を抽出

行のすべての列からデータを抽出するには、行の左にある四角いインジケータを右クリックし、**[行を抽出]** > **[すべての列を次へ抽出]** > **[extr:]<**コンプレックス タイプ名**>** の順に選択します。各セルについて、ロボット ワークフローに **[データの抽出]** ステップが表示されます。抽出したデータを設定し、適切なタイプに変数を割り当てることは可能です。

一致する列を抽出

属性名とそのタイプのみに一致する列からデータを抽出するには、行の左にあるインジケータを右クリックし、**[行を抽出]** > **[一致する列を次へ抽出]** > **[extr:]<**コンプレックス タイプ名**>** の順に選択します。ロボットのワークフローに、選択したコンプレックス タイプに一致するセルの数と同じ数の [データの抽出] ステップが表示されます。データを編集および設定し、ロボットのワークフローの次のステップに進みます。

長いバイナリ タイプまたは長いテキスト タイプのデータを含むフィールドは、[行を抽出] ステップでは抽出できません。セルを右クリックし、**[コンポーネント アクション]** > **[長いバイナリ値を読み取る]** または **[コンポーネント アクション]** > **[長いテキスト値を読み取る]** の順にそれぞれ選択することにより、バイナリ データを手動で抽出できます。

ロボットのツリー ビューでは、長いバイナリ タイプおよび長いテキスト タイプの属性が `der_is_big_value="true"` であることに注意してください。

プロパティ

エイリアス
エイリアスまたはファインダーを追加します。

ベース ファインダー
使用するコンポーネント (アプリケーション、コンポーネント、またはデバイス) を指定します。

デバイス
デバイスを選択します。

アプリケーション
アプリケーションの名前をセットします。

コンポーネント
コンポーネントの名前を設定します。

テキスト一致 **(Regex)**
下位層の有無にかかわらず、ツリー内で正確なテキストを検索できます。**[Regex]** ボックスに入力して、検索基準を指定します。

コンポーネント アクション

アクション	説明
[長いバイナリ値を読み取る]	セルから長いバイナリ タイプのデータを抽出します。
[長いテキスト値を読み取る]	セルから長いテキスト タイプのデータを抽出します。

画像からテキスト抽出

このステップは、選択した OCR エンジンを使用して画像からテキストを抽出するのに役立ちます。

Tesseract エンジン (デフォルト) または OmniPage エンジンを選択して、画像からテキストをキャプチャできます。Tesseract の場合は、英語のみがインストールに含まれています。OmniPage には、インストールでサポートされるすべての言語が含まれています。バージョン 11.1 より前の Kofax RPA で作成された口ボットは、Tesseract エンジンを使用します。

認識言語の変更の詳細については、『Kofax RPA Desktop Automation サービス ガイド』の「デフォルトの OCR 言語の変更」を参照してください。

組み込みブラウザで操作する場合のように、Desktop Automation サービスを使用しないでこのステップを使う場合は、`ocr.cfg` ファイルの OCR 設定を変更します。詳細については、拡張 OCR 設定を参照してください。

表のセル エレメントからテキストを抽出することはできません。

プロパティ

名前
ステップの名前。

変数
抽出されたテキストを保存する変数を指定します。

テキストのフォント サイズ
- 小：12 ピクセル未満のフォント。
- 中 (デフォルト): 12 ピクセルから 24 ピクセルまでのフォント サイズ。
- 大：24 ピクセルを超えるフォント。

フォント サイズの選択は、テキスト分析および認識の速度に影響します。たとえば、大きい画像を分析する場合、[大] を選択すると、[中] と比べて分析が 2 倍または 3 倍加速します。[小] を選択すると、[中] と比べて認識速度が 2 倍または 3 倍低下します。さまざまな設定を試してみて、速度と認識結果が最適になるものを選択してください。

画像の二値化
- 自動：Tesseract アルゴリズムが画像のテキスト認識準備に使用されます。
- カスタム：Kofax RPA アルゴリズムが画像のテキスト認識準備に使用されます。詳細については、テキスト認識の微調整を参照してください。

  閾値のデルタ
  なし

  Positive
  - Small
  - Medium
  - Large

  Negative
  - Small
  - Medium
  - Large

テキスト認識の微調整

以下の情報は Tesseract エンジンにのみ適用されます。

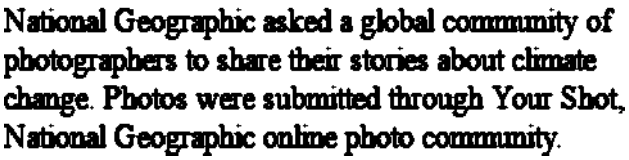
デフォルトで、Kofax RPA では、ほとんどの場合許容可能な結果を生成する Tesseract アルゴリズムが OCR に使用されます。テキストが識別される前、アルゴリズムによって画像が黒白画像に変換され、テキストが認識されるようにその他の調整が行われます。認識可能なテキストが背景に紛れ、認識結果がよくない場合、[画像のバイナリ化] オプションで [カスタム] に変更し、許容可能な結果が生成されるように [閾値のデルタ] オプションを調整できます。

以下は、認識のために画面からコピーされた画像です。

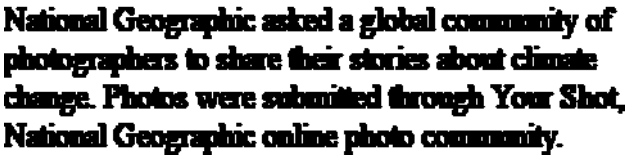
National Geographic asked a global community of photographers to share their stories about climate change. Photos were submitted through Your Shot, National Geographic online photo community.

以下は、テキスト認識のための Kofax RPA アルゴリズムからの内部画像調整結果です。各画像に一連の [閾値のデルタ] オプションでラベルが付けてあります。複雑な場合、別のオプションを試してみて、認識結果が最適になるものを選択してください。

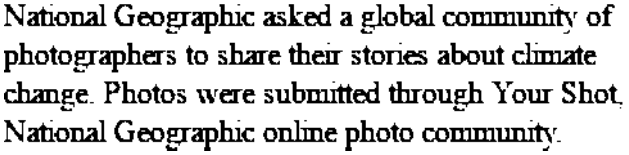
閾値のデルタ：なし


National Geographic asked a global community of photographers to share their stories about climate change. Photos were submitted through Your Shot, National Geographic online photo community.

閾値のデルタ：**Positive Medium**


National Geographic asked a global community of photographers to share their stories about climate change. Photos were submitted through Your Shot, National Geographic online photo community.

閾値のデルタ：**Negative Medium**


National Geographic asked a global community of photographers to share their stories about climate change. Photos were submitted through Your Shot, National Geographic online photo community.

拡張 OCR 設定は、`ocr.cfg` ファイルで編集できます。詳細については、拡張 OCR 設定を参照してください。

拡張 OCR 設定

Kofax RPA は、画像からテキストを抽出し、制限付きまたはオートメーションなしの API を使用するアプリケーションを自動化する光学文字認識 (OCR) 機能を提供します。

OCR は複雑な処理であるため、スクリーン フォント、背景色と前景色、テキスト サイズなど、さまざまな要素によって認識結果は異なります。Kofax RPA では、認識結果の変更に使用可能な構成設定を含む `ocr.cfg` ファイルがインストールされます。ファイルには構成設定の詳細な記述が含まれています。`ocr.cfg` ファイルは、以下のように Kofax RPA のインストール ディレクトリに配置されています。

- Desktop Automation サービスがインストールされた Windows ベースの自動化されたコンピュータでは、

  Desktop Automation サービス インストール ディレクトリの `DesktopAutomationService\lib`。

  例：

```
C:\Program Files\Kofax RPA DesktopAutomation 11.5.0.0
\DesktopAutomationService\lib
```

- 組み込みブラウザを使用するローカルの Windows ベースのコンピュータでは、
  Kofax RPA のインストール ディレクトリの `nativelib\hub\windows-x64\[ビルド番号]\lib`*。
  例：
  `C:\Program Files\Kofax RPA 11.5.0\nativelib\hub\windows-x64\166\lib`
- 組み込みブラウザを使用するローカルの Linux ベースのコンピュータでは、
  `nativelib/hub/linux-x64/<build number>/lib` in the Kofax RPA installation directory.例：
  `Kofax RPA_11.5.0.0/nativelib/hub/linux-x64/166/lib`

* ビルド番号は、プログラムのバージョンごとに異なります。

OCR エンジンと言語の変更

OCR エンジンの変更

Kofax RPA は Tesseract (デフォルト) または OmniPage エンジンを使用して、画像からテキストをキャプチャします。OCR エンジンをデフォルトから変更するには、次の手順を実行します。

1. コンピュータ上で `ocr.cfg` ファイルを見つけます。
2. テキスト エディターで `ocr.cfg` を開き、`engine_type` オプションを見つけます。
3. `engine_type = omnipage` のように、`omnipage` などの OCR エンジンを値として指定します。
   デフォルトの OCR エンジン (Tesseract) を使用する場合は、`engine_type` オプションの値として `tesseract` を指定するか、このオプションから値を削除します。

OCR 言語の変更

1. コンピュータ上で `ocr.cfg` ファイルを見つけます。
2. テキスト エディターで `ocr.cfg` を開き、`default_language` オプションを見つけます。
3. `eng` を `jpn` などの別の言語コードに置き換えるか、複数の言語を使用する場合は、`default_language=eng+jpn` のように + 記号を使用して `jpn` を追加します。言語コードは ISO 639-3 または ISO 639-1 形式である必要があります。ファイルを保存して閉じます。

OmniPage には、インストールでサポートされるすべての言語が含まれています。Tesseract の場合は、英語のみがインストールに含まれています。Tesseract で UI 認識言語をさらに追加するには、ツリー モードにある「Tesseract の UI 認識言語の変更または追加」セクションを参照してください。

Desktop Automation サービスの OCR エンジンと言語設定は、サービスを実行する各コンピュータの Desktop Automation サービス設定ウィンドウで個別に指定します。『Kofax RPA Desktop Automation サービス ガイド』を参照してください。

画像の前処理

以下の情報は Tesseract エンジンにのみ適用されます。

画像に実際の OCR プロセスを開始する前に、特定のアルゴリズムを使用して画像の前処理を行います。使用するアルゴリズムは ocr.cfg ファイルの `preparation` 設定で定義されます。デフォルトでは、`normal` に設定されています。

デフォルトの前処理アルゴリズムで満足できる結果が得られない場合は、別のアルゴリズムに切り替えることができます。そのためには、`preparation` の値を `10.2` に変更して、変更を保存します。

Tesseract のトレーニング

Kofax RPA は、Tesseract OCR エンジンまたは OmniPage OCR エンジンを使用して画像からテキストをキャプチャし、インテリジェント スクリーン オートメーション (ISA) を実行します。OmniPage には、インストールでサポートされるすべての言語が含まれています。Tesseract の場合は、英語のみがインストールに含まれています。Tesseract の言語は、各言語に対応する `.traineddata` ファイルを提供することで変更が可能です。

特定の言語または文字の認識に問題が発生した場合、Tesseract をトレーニングしてフォントを適切に読み取らせることができます。

トレーニング データの準備用に Kofax RPA で提供されるスクリプトは、Linux オペレーティング システム向けに設計されています。現在、Tesseract バージョン 3.4.0 が使用されています。

前提条件

トレーニング データを作成する前に、システムが以下の前提条件を満たしていることを確認します。

システムが **Ubuntu** ベースの場合のシステム要件

次のように、`sudo apt-get install` コマンドを使用してライブラリをインストールします。

```
sudo apt-get install libicu-dev libpango1.0-dev libcairo2-dev git
```

トレーニングの前提条件

Kofax RPA インストール ディレクトリ内の `nativelib/hub/linux-x64/<hub_id>/tools/tesseract_train/bin` に移動して、`prepare.sh` スクリプトを実行します。例：

```
$ cd /home/user88/Kofax_RPA/nativelib/hub/linux-x64/574/tools/tesseract_train/bin
$ ./prepare.sh
```

自動トレーニング

このモードは、認識対象の UI で TTF フォント ファイルが使用されている場合に選択します。このモードは、手動トレーニング モードよりもシンプルです。対象フォントのトレーニング データ ファイルを作成するには、`tesseract_train/bin` フォルダ内の、言語コード、フォント名、フォント ファイル ディレクトリを指定する `tesseract_auto.sh` スクリプトを以下のように実行します。

スクリプトを `tesseract_train/bin` 作業ディレクトリから実行していることを確認します。

```
$ ./tesstrain_auto.sh --lang eng --fontlist 'Envy Code R' --fonts_dir ..
```

スクリプトを実行すると、以下のメッセージが表示されます。

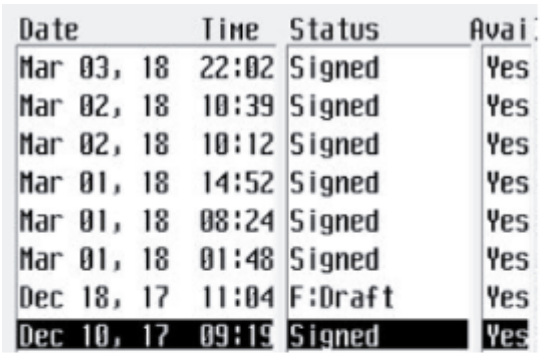
```
Moving /tmp/tmp.OtEqYbS3qV/eng/eng.traineddata to ../output
Completed training for language 'eng'
```

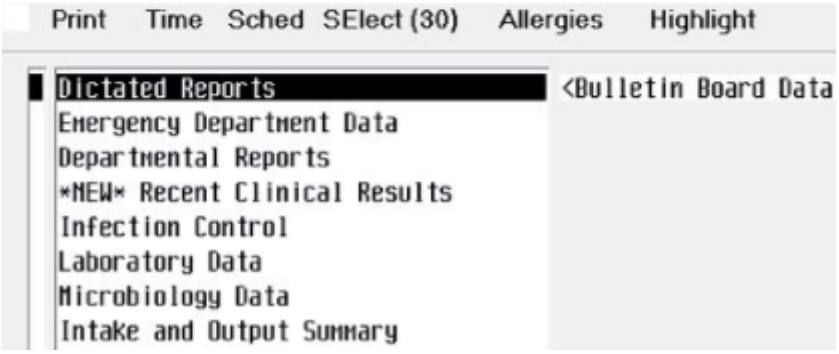
トレーニングしたデータ ファイルを Kofax RPA で使用できるようになります。『Kofax RPA Desktop Automation サービス ガイド』の「デフォルトの OCR 言語の変更」と、ツリー モードにあるトピックインテリジェント スクリーン オートメーションの「UI 認識言語の変更または追加」を参照してください。

手動トレーニング

このモードは、UI で TTF フォント ファイルが使用されていない場合 (つまり自動モードを適用できない場合) で、ロボットの認識対象となる文字がすべてアルファベットの UI スクリーン ショットが多数ある場合に選択します。トレーニング画像ファイルがスクリプトによって自動的に作成される自動モードとは異なり、トレーニング画像は手動で作成する必要があります。このファイルの作成には、時間と労力が必要になります。

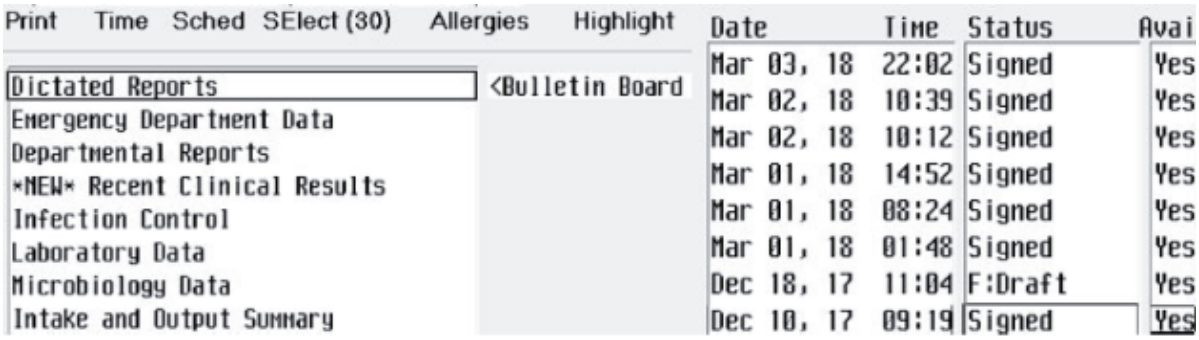
Tesseract 用のトレーニング データ ファイルを作成するには、以下の手順を実行します。ファイルには、最終トレーニング データ ファイルに必要なすべての文字 (大文字および小文字、数字、コンマ、ピリオドなど) を含む必要があります。以下の例は、次の UI で使用するトレーニング データの作成方法を示しています。



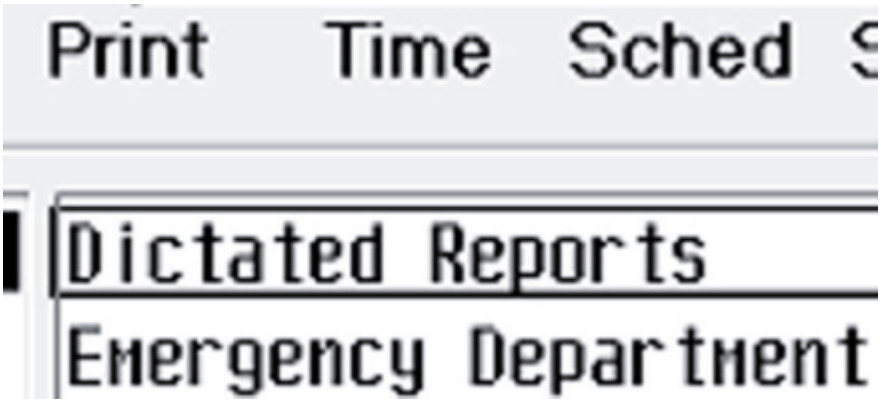


1. 使用する完全な文字セットを決定します。トレーニング ファイルの作成時には、各文字に対するサンプルの最小数は 5 であることに注意してください。最も頻繁に使用される文字の場合、追加でサンプルを含めます。
2. 単一の TIFF ファイルに、トレーニングで使用する UI スクリーン ショットの全部分を貼り付けます。この操作では使用する画像エディターは任意です。この例では、ターゲットのアルファベットを 10 ~ 15 の英字に限定します。実際の操作では、すべての文字のサンプルを用意します。

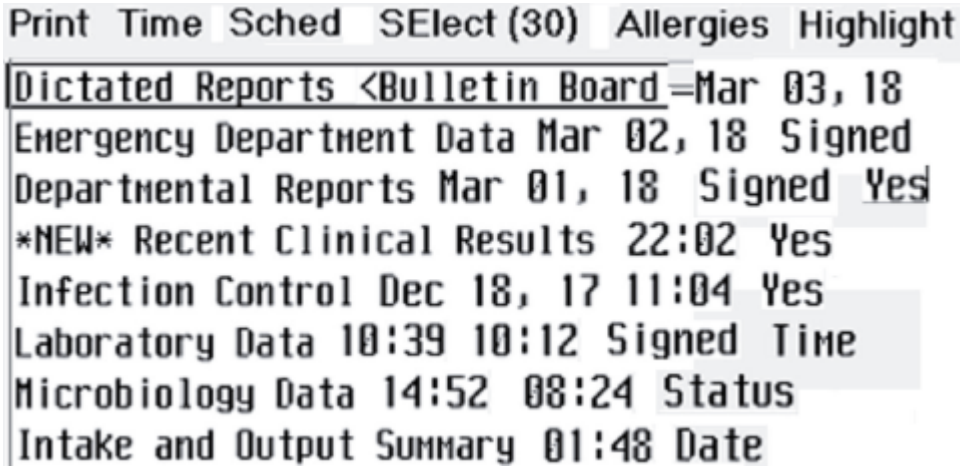
3. 反転色の領域を選択して、色を反転させて通常色にします。



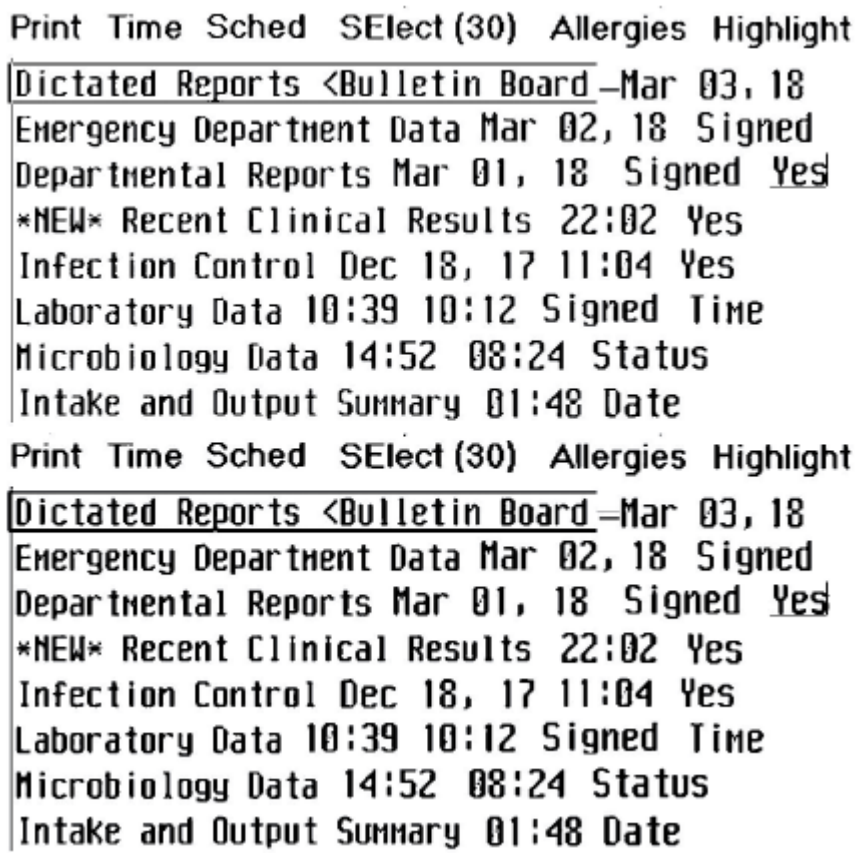
4. 大文字の高さが 36 ピクセルになるように、キュービック補間を使用して画像の大きさを調節します。この例では、画像の大きさを 2.97 倍に拡大しました (画像の一部のみを表示)。



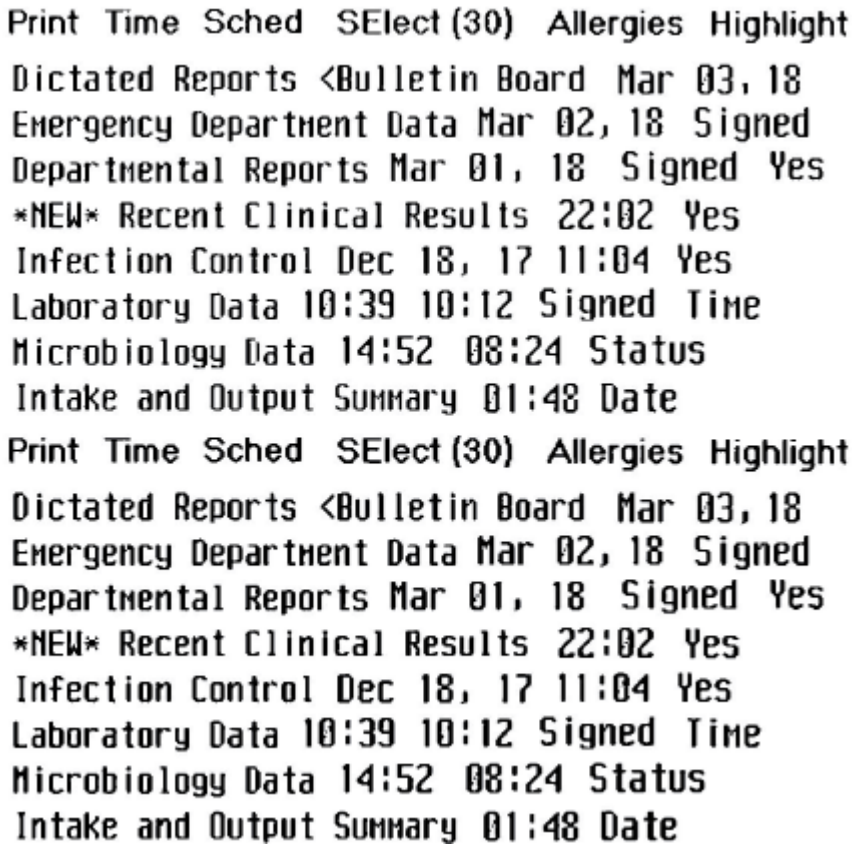
5. 単語を調節して、テキスト領域間に大きな空白がなくてもテキスト行を容易に検出できるようにします。以下の例のように、余分と思われるテキストを除去します (ページに合わせて縮小しています)。



6. 画像をグレースケールに変換し、テキストの質が最も良くなるしきい値の色効果を適用します。適切なしきい値を選択することは難しい可能性があります。複数の異なるしきい値を適用することを考慮し、結果の画像を単一の TIFF ファイルにコピーします。トレーニング画像に、表現の異なる同じ文字が複数含まれることになります。この例では GIMP エディターで 125 および 150 のしきい値を適用して、画像を 1 つのファイルにコピーしました。画像の上にある文字のほうが、下にある文字よりも薄いことが分かります (ページに合わせて縮小しています)。



7. 以下の例のように、手動でノイズを除去します (ページに合わせて縮小しています)。



8. 画像を圧縮せずに TIF または TIFF 形式で `MyFont.tif` のような名前で保存します。
9. ボックス ファイルを作成します。ボックス ファイルは、画像を囲む境界ボックスの座標を使用して、トレーニング画像内の文字を 1 行ごとに 1 文字ずつリストするテキスト ファイルです。GitHub で Tesseract プロジェクトの「Training Tesseract - Make Box Files」ページを参照してください: https://github.com。

   ボックスのテキストをコピーし、新規ファイルに貼り付けて `MyFont.box` のような名前にします。

   この例では、ボックス ファイルは以下の行から開始します。

```
P 15 1076 39 1108 0
r 41 1076 53 1100 0
i 57 1076 62 1108 0
n 68 1076 89 1100 0
t 92 1076
...
```

10. `tesseract_train/bin` フォルダに移動し、言語コード、TIF ファイルとボックス ファイルへのパスを指定する `tesstrain_manual.sh` スクリプトを以下のように実行します。

```
$ ./tesstrain_manual.sh --lang eng --box_file ../MyFont.box --
training_image ../MyFont.tif
```

    スクリプトを実行すると、以下のメッセージが表示されます。

```
Moving /tmp/tmp.OtEqYbS3qV/eng/eng.traineddata to ../output
```

トレーニングしたデータ ファイルを Kofax RPA で使用できるようになります。

詳細については、GitHub Web サイトの Tesseract wiki ページを参照してください。

ツリーを XML として抽出

このステップでは、アプリケーション ツリーの一部を抽出して、XML 文字列として変数に格納します。

Chromium の組み込みブラウザで「ツリーを XML として抽出」ステップを使用した場合、サポートされていない記号、誤ったタグ、または誤った属性を使用していると、不正な形式の HTML から無効な XML が生成されることがあります。この場合、無効な要素は以下のように置き換えられ、XML が有効になります。

- サポートされていない記号は、最初の記号以外、すべてアンダースコアに置き換えられます。サポートされていないタグで属性が始まる場合、ステップの実行時に属性の先頭にアンダースコアが追加されます。
- タグが正しくない場合は、次の形式で元のタグ名に特別な属性が追加されます。
  `kapow:original_tag_name=<元のタグ名>`
- 属性が正しくない場合は、次の形式で元の属性名に特別な属性が追加されます。
  `kapow:attr_<正しくない属性>_<インデックス>=<元の属性名>`

名前内の Unicode 文字が正しい場合は、変換または置換されません。

プロパティ

拡張属性を含める
拡張属性を含めるか除外するかを選択できます。たとえば、HTML ノードから純粋な HTML コードを抽出することが目的である場合は、レコーダー ビューにステップを挿入するときにショートカット メニューの [拡張属性を除外する] を選択するか、ワークフロー ビューのステップで [拡張属性を含める] オプションをオフにします。拡張属性の詳細については、ロボット構築の概要 の「アプリケーション ツリー」を参照してください。

エクスプレッションを評価
オプション。値での使用が推奨される変換関数のリストが含まれています。特定の値に対して推奨される関数リストは、値の種類に応じて制限されます。変換関数を指定するには、ステップのコンテキスト メニューで **[エクスプレッションを評価]** をクリックし、次のいずれかを実行します。

- 推奨関数のリストから必要な関数を選択します。
- [プレーン] をクリックして、必要な関数を手動で入力します。

サポートされている関数と例の詳細については、 エクスプレッション。
1 つ以上の [エクスプレッションを評価] ステップを追加できます。

現在のインを保存
抽出/変換された値を格納する、指定されたタイプの変数の名前。

1 つ以上の現現在のイン ステップを保存を追加できます。

ツリーを XML として抽出ステップには、次の一連のアクション ステップを含めることもできます。

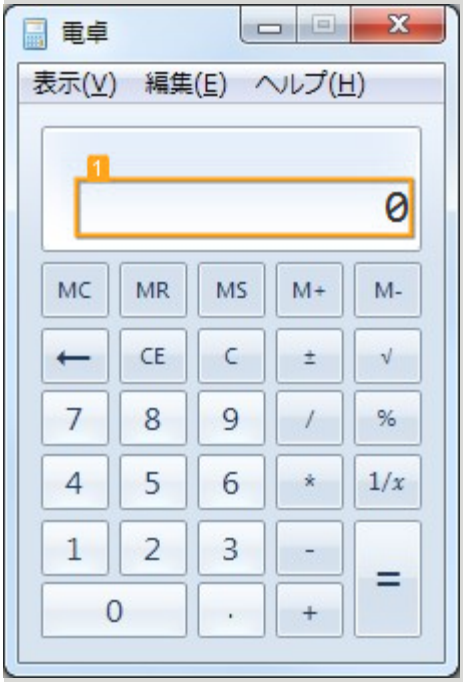
- 条件ステップ

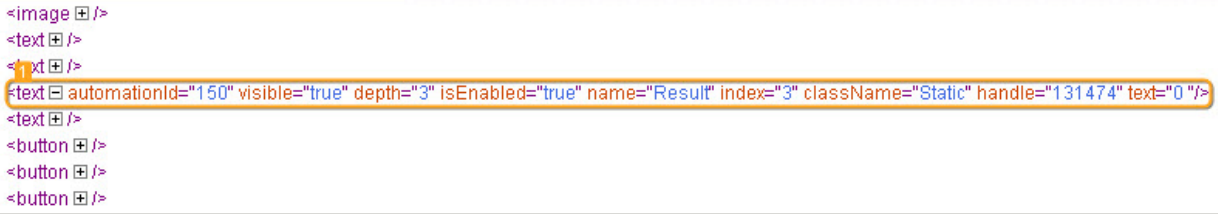
- グループ ステップ
- スロー ステップ
- ログの書き込みステップ

例: 選択したツリーと XML 文字列

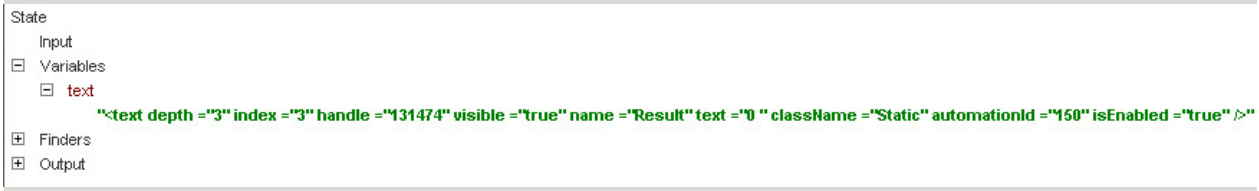
以下の図に、XML 文字列として変数にエクスポートされた計算機ウィジェット ツリーの一部を示します。

変数にエクスポートされた属性の順序は、レコーダー ビューのアプリケーション ツリーと異なる場合があります。





以下は、変数にエクスポートされた XML 文字列です。



値を抽出

このステップでは、異なる工レメントの値を抽出します。

プロパティ

コンポーネント
ステップのコンポーネント ファインダー。

抽出タイプ
抽出する情報のタイプを選択します。

- [属性]: 指定された属性の値を抽出します。
- [拡張属性]: 選択した拡張属性の値を抽出します。接頭辞 `der_` のない属性の名前を指定します。
- [テキスト]: 選択したコンポーネントの直下の工レメントからテキストを抽出します。すべての子工レメントからテキストを抽出するには、[すべての下位層を含める] を選択します。

エクスプレッションを評価
オプション。値での使用が推奨される変換関数のリストが含まれています。特定の値に対して推奨される関数リストは、値の種類に応じて制限されます。変換関数を指定するには、ステップのコンテキストメニューで **[エクスプレッションを評価]** をクリックし、次のいずれかを実行します。

- 推奨関数のリストから必要な関数を選択します。
- [プレーン] をクリックして、必要な関数を手動で入力します。たとえば、整数を表すテキストを抽出して整数変数に格納する必要がある場合は、エクスプレッション `$initial.integer()` を使用できます。

サポートされている関数と例の詳細については、エクスプレッションを参照してください。
1 つ以上のエクスプレッションを評価ステップ を追加できます。

現在のインを保存
抽出/変換された値を格納する、指定されたタイプの変数の名前。

1 つ以上の現現在のイン ステップを保存を追加できます。たとえば、同じステップ内で個人のフルネームを抽出し、それぞれ名と姓などの 2 つの変数に保存する必要がある場合に役立ちます。

値の抽出ステップには、次の一連のアクション ステップを含めることもできます。

- 条件ステップ
- グループ ステップ
- スロー ステップ
- ログの書き込みステップ

ファイル システム アクション

[ファイル システム アクション] ステップを使用して、ファイルへの直接アクセスまたは ロボット ファイル システム (RFS) によるファイル、ディレクトリ、およびその他のアイテムに対する操作を実行します。

このステップは、次の 3 つのレベルで操作を行います。

- デバイス レベルのアクションにはコンテキストはなく、パラメータによって完全に制御されます。
- アプリケーション レベルのアクションは、各ディレクトリがアプリケーションによって表されるディレクトリ レベルで動作します。
- コンポーネント レベルのアクションは、デバイス ツリーの `<item>` ノードで表されるファイルまたはディレクトリで動作します。

利用可能なコマンドは、どのレベルで発生しても同じように機能しますが、パラメータの数が異なります。

ローカル Desktop Automation モードでのローカル ファイル システムへのアクセスを含むファイル システム アクセスを有効にするには、RoboServer 設定アプリケーションの [セキュリティ] タブで [ファイル システムとコマンド ラインのアクセスを許可] オプションを選択します。

RoboServer がローカル ファイル システムにアクセスできないように設定されている場合、ロボットは実行できますが、ステップがローカル デバイス上で [直接アクセス] に設定した [ファイル アクセス] を使用するとエラーが発生します。ただし、このステップは、[ファイル アクセス] が [RFS 経由] に設定されているローカル デバイス上で機能します。リモート デバイス上では、[直接アクセス] と [RFS 経由] を実行できます。

アプリケーションとディレクトリ

ロボットは [ディレクトリを一覧表示] アクションを使用して、ディレクトリ内の各オブジェクトの `<item>` ノードとそのオブジェクトの一部のコア プロパティを含むアプリケーションを開きます。

- アプリケーションは静的であり、ディレクトリのコンテンツが変更された場合でも更新されません。
- コンテンツを更新するには、ロボットでアプリケーションを閉じてから再度開くか、[更新] アクションを使用してディレクトリのコンテンツを再スキャンします。
- [ディレクトリを一覧表示] アクションにより、新しいアプリケーションが開きます。このアクションは、アプリケーションに名前属性を設定するために必要な [アプリケーション名] パラメータを提供します。
- アプリケーションを開く操作と閉じる操作は非同期で行われます。ディレクトリを開いて閉じるロボットは、操作が完了するまで待機するか、アプリケーション名を使用して以前のアプリケーションを見つけないようにする必要があります。

  エラーを回避するために、ロボット内のアプリケーションにはそれぞれ異なるアプリケーション名を割り当てます。

エラー処理

いずれかのアクションでエラーが発生すると、`DeviceIssue` 例外が生成されます。これらのエラーは、「トライ-キャッチ」ステップを使用してキャッチできます。

[最後のエラーを取得] アクションを使用して、ファイル システム ドライバーが実行した最後のアクションによって報告されたエラーを取得できます。

アクションとプロパティ

アクション

- ディレクトリを一覧表示: ディレクトリのコンテンツを表す新しいアプリケーションを開きます。このアクションは、完了するまでに長い時間がかかる可能性があります。

  [ファイル アクセス]、[ディレクトリ]、および [アプリケーション名] パラメータを指定します。
- ファイル削除: ファイルを削除します。

  [ファイル アクセス] および [ファイル] パラメータを指定します。
- ディレクトリを作成: ディレクトリを作成します。このアクションは、基盤となるストレージのディレクトリ作成セマンティクスを使用します。

  [ファイル アクセス] および [ディレクトリ] パラメータを指定します。
- ディレクトリを削除: ディレクトリを削除します。ディレクトリが空でない場合、基盤となるストレージによっては、このアクションが失敗する可能性があります。

  [ファイル アクセス] および [ディレクトリ] パラメータを指定します。
- 存在する: オブジェクトが存在するかどうかをテストし、結果をブール値として返します。

  ファイル アクセス パラメータとアイテム パラメータ、および結果を [結果] フィールドに格納する変数を指定します。
- ロック ステータス: Windows ファイルの読み取り/書き込みロックのテストを実行し、数値ステータス インジケータを返します。ロック ステータスは、RFS および Linux では使用できません。
  - `1` ファイルは存在しますが、ロックされていません。
  - `0` ファイルが存在し、ロックされています (読み取りおよび書き込み)。
  - `-1` ファイルが存在しません。
  - `-2` 機能がサポートされていません。
  - `-3` 別の理由で失敗しました。(根本的なエラーを取得するには、[最後のエラーを取得] ステップを使用します。)
- ファイル コピー: ファイルを新しい場所にコピーします。RFS とローカル ファイル システムの間でファイルをコピーする場合にも使用します。ソース パラメータおよびターゲット パラメータには、ファイルの名前を含める必要があります。このアクションは、基盤となるストレージのコピー セマンティクスを使用します。

  ファイル名を含むパスを使用して [ソース] および [ターゲット] のパラメータを指定し、ファイル アクセスを指定します。

  > ⚠ ファイル `c:\a\b\c.txt` をディレクトリ `d:\destination` にコピーする場合、[ソース] フィールドには `c:\a\b\c.txt` が含まれている必要があり、[ターゲット] フィールドには `d:\destination\c.txt` が含まれている必要があります。[ターゲット] フィールドでファイル名を省略した場合、またはターゲット ディレクトリが存在しない場合、結果が予測できない可能性があります。

- 移動: オブジェクトを移動または名前変更します。ソース パラメータおよびターゲット パラメータには、オブジェクトの名前を含める必要があります。このアクションを使用して、RFS とローカル ファイル システム間でオブジェクトを移動することはできません。実装では、基盤となるストレージの移動セマンティクスを使用します。

  オブジェクト名を含むパスを使用して、[ファイル アクセス]、[ソース]、および [ターゲット] パラメータを指定します。

- 名前の変更: 現在の場所にあるオブジェクトの名前を変更します。
  [ファイル アクセス]、[アイテム]、および [新しい名前] パラメータを指定します。
- タイプを取得: オブジェクトのタイプを取得します。オブジェクトが存在しない場合は、`DeviceIssue` 例外をスローします。このアクションは、次のような値を返します。
  - `file` オブジェクトはファイルです。
  - `directory` オブジェクトはディレクトリです。
  - `other` オブジェクトは存在しますが、ファイルまたはディレクトリではありません。オブジェクトの正確なタイプは基盤となるストレージによって異なりますが、デバイスやシンボリック リンクなどのその他のオブジェクトである場合もあります。

  [ファイル アクセス] パラメータと [アイテム] パラメータ、および結果を [タイプ] フィールドに格納する変数を指定します。
- パスを取得: アイテムへのパスを取得します。
- すべてのディレクトリを閉じる: ロボットによって開かれたディレクトリに関連付けられているすべてのアプリケーションを閉じます。
- ディレクトリを閉じる: アプリケーションを閉じます。
- カウント: コンポーネント ファインダーに一致するオブジェクトの数をカウントします。このアクションにより、存在のチェックとワイルドカード検索を実行します。
- 最後のエラーを取得: ファイル システム ドライバーが実行した最後のアクションからエラー メッセージを取得します。
  [エラー] フィールドに結果を格納する変数を指定します。
- 更新: アプリケーションによって表されるディレクトリのコンテンツをスキャンして再ロードし、ツリーを更新します。このアクションは、完了するまでに長い時間がかかる可能性があります。

プロパティ

- ファイル アクセス: [直接アクセス] または [RFS 経由] のいずれかを選択します。
- ディレクトリ: ディレクトリのパスと名前を指定します。
- ファイル: ファイルのパスと名前を指定します。
- アイテム: アイテムのパスと名前を指定します。
- アプリケーション名: アプリケーションで名前属性を設定します。
- 結果: アクションの結果を保存する変数を指定します。

ディレクトリ ツリー

[ディレクトリを一覧表示] アクションで開かれた各ディレクトリは、独自のツリーを持つアプリケーションによって表されます。

ツリーには次の構造があります。

```
<fs title="…" driver="…" name="…">
    <directory path="…" count="…">
        <item name="…" size="…" extension="…" type="…" created="…" modified="….">
        <item name="…" size="…" extension="…" type="…" created="…" modified="….">
        <item name="…" size="…" extension="…" type="…" created="…" modified="….">
        ….
```

次の表は、ツリー内のノード (要素とも呼ばれます)、属性、および子要素を定義します。

fs ノード	説明	タイプ
title	アプリケーションのタイトル。これは、ディレクトリのパスに設定されます。	属性
driver	固定: fs。	属性
name	アプリケーションの名前。 この属性は、[ディレクトリを一覧表示] アクションによって設定されます。	属性
directory	ディレクトリを表します。	子要素

directory ノード	説明	タイプ
path	[ディレクトリを一覧表示] アクションで使用されるパスが含まれます。	属性
count	ディレクトリ内のオブジェクトの数。	属性
item	各アイテムの要素は、ディレクトリ内のオブジェクトを表します。 これらの要素の順序は定義されておらず、[更新] アクションの後に変更される可能性があります。	繰り返しの子要素

item ノード	説明	タイプ
name	オブジェクトの名前。	属性
size	基盤となるストレージによって報告された、オブジェクトのサイズ (バイト単位)。	属性
extension	ディレクトリの場合は空です。オブジェクトの拡張子 (ファイルの場合) は小文字に変換されます。	属性
type	file、directory、または other。	属性
created	基盤となるストレージによって報告された、オブジェクトが作成された時刻。 この値は、ISO8601 表記でローカル時間として表されます。	属性
modified	基盤となるストレージによって報告された、オブジェクトが最後に変更された時刻。 この値は、ISO8601 表記でローカル時間として表されます。	属性

フォーカス

これは、選択したアプリケーションをフォーカスするアプリケーション レベルのステップです。このステップを使用して、たとえば、最小化された状態で開始したアプリケーションを最前面に表示することができます。このステップは、リモート デバイス上のアプリケーションで使用できます。

フォーカス ステップを挿入するには、[レコーダー ビュー] のアプリケーション タブを右クリックし、**[アプリケーション アクション]** > **[フォーカス]** を選択します。Kofax RPA により、ステップを実行するために必要なファインダーとガードが自動的に挿入されます。

電子メールごとに

このステップは、アプリケーションをレコーダー ビューで開き、選択した電子メール フォルダから 1 つのメッセージを表示します。メッセージのデータは、テーブル ビューとツリー ビューで使用できます。このステップは、電子メール フォルダのメッセージに対してイテレーションを行うループ ステップとして機能します。イテレーションを行うたびに、次のメッセージのために抽出されたデータでビューの内容が更新されます。すべてのメッセージのイテレーションが完了する (または口ボットが例外やブレーク ステップによってループを終了する) と、アプリケーションは自動的に閉じます。また、口ボットが実行を停止した場合にもアプリケーションは閉じます。

口ボットは、イテレーション変数と続行およびブレーク ステップを使用することで、他のループ ステップと同様にループの実行を制御できます。

メッセージごとに、このテーブルには次のプロパティが列として一覧表示されます。

- 送信者: John Doe <j.doe@organization.com> という形式の送信者。
- 件名: メッセージの件名。
- 受信済み: 2023-05-17T15:27:33+01:00 (ISO-8601) という形式の、メッセージが受信された日付。
- サイズ: バイト単位のメッセージのサイズ。
- 添付ファイルあり: メッセージに 1 つ以上の添付ファイルがあるかどうかを示します。
- 既読: メッセージが既読としてマークされているかどうかを示します。
- 添付ファイルの数: 添付ファイルの数。

[電子メールごとに] ステップは、[ループ] および [アプリケーション] カテゴリに属します。

プロパティ

アプリケーション名
アプリケーションに名前を付けます。

エラーを回避するために、口ボット内のアプリケーションにはそれぞれ異なるアプリケーション名を割り当てます。

プロトコル
電子メール サーバーへの接続に使用する電子メール プロトコルを選択します。選択肢は「imap」、「pop3」、およびそれらのセキュアなバリアント「imaps」、「imaps+tls」、および「pop3s」です。オプション「imap+tls」は、便宜的な TLS を必要とする電子メール サーバー用です。

電子メール サーバー
電子メール サーバーの名前またはアドレスを指定します。

電子メール サーバー ポート (オプション)
電子メール サーバーのポート番号を指定します。指定しない場合は、選択したプロトコルの標準ポートが使用されます。

アカウント名とパスワード
電子メール サーバーにログインするためのクレデンシャルを入力します。

メールボックスのパス (オプション)
電子メール サーバーで開くフォルダの名前。ネストされたフォルダのパスの形式である場合があります (この場合、パス内の要素はスラッシュで区切る必要があります)。受信電子メールが保存されているフォルダを開くには、パスや名前を指定しないでください。
例: `Junk E-Mail`、`Subscriptions/Mailing lists`.

POP3 はフォルダをサポートしていないため、POP3 または POP3S プロトコルを使用する場合は、このフィールドを空白のままにします。

取得バッチ サイズ (オプション)
この設定により、メール サーバーから一度に取得するメッセージの数が決まります。イテレーションごとに 1 つのメッセージを表示するループには影響しません。ステップの実行で エラーが発生した場合にのみ、設定の値を変更する必要があります。ロボットのメモリが不足している場合は値を減らし、メール サーバーから要求の実行回数が多すぎると報告された場合は値を増やします。デフォルト値は 100 です。

イテレーション変数 (オプション)
イテレーション データを格納する変数を指定します。

コンポーネント アクション
次のコンポーネント アクションが使用できます。

アクション	説明
メッセージをコピー	**IMAP** および **IMAPS** プロトコルの場合。選択したメッセージを指定したターゲット フォルダにコピーします。このアクションを使用できるかどうかは、電子メール サーバーでのコピー操作のサポートに応じて異なります。
メッセージを削除	電子メール メッセージを削除します。
メッセージの内容を取得	RFC 822 形式のメッセージとヘッダーを含む文字列値を返します。
メッセージを HTML として開く	Chromium 組み込みブラウザを使用して、電子メールを HTML Web ページとして開きます。開いている電子メール テーブル内の行を操作します。
メッセージを移動	**IMAP** および **IMAPS** プロトコルの場合。選択したメッセージを、指定したターゲット フォルダに移動します。このアクションを使用できるかどうかは、電子メール サーバーでの移動操作のサポートに応じて異なります。

アクション	説明
メッセージを既読にする	**IMAP** および **IMAPS** プロトコルの場合。メッセージに既読または未読のフラグを立てます。 チェック ボックス **[既読]** をオンにしてメッセージを既読としてマークするか、チェック ボックスをオフにしてメッセージを未読としてマークします。
添付ファイルを取得	添付ファイルのバイナリ データを取得し、変数に保存します。このアクションは、添付ファイルが大きい場合に便利です。 必要なデータを取得するには、次のプロパティを設定します。 • インデックス: 抽出する添付ファイルの、0 から始まるインデックス。 • 添付ファイル データ: 結果の添付ファイル データを格納するバイナリ変数。

POP3 または **POP3S** プロトコルに関する制限事項

- フォルダはサポートされていません。
- 一部の POP3 サーバーには、他のインターフェイスの POP3 接続を介して行われた変更が反映されません。たとえば、ロボットが POP3 プロトコルを使用して Gmail アカウントからのメッセージを削除した場合でも、Web ブラウザ インターフェイスには引き続きそのメッセージが表示されることがあります。ロボットの場合、メッセージは削除されます。
- メッセージを削除ステップを行うとメッセージに削除のマークが付きますが、アプリケーションを閉じるまでメッセージはサーバー上に残ります。
- ほとんどの POP3 サーバーでは、メッセージがダウンロードされるとメッセージは削除されます。
- 「既読」または「未読」フラグは設定できません。

DateTime の書式設定

このステップを使用すると、Date、Time、または DateTime 値を Text 値に変換できます。このステップは、ロボットのワークフローからのみ挿入できます。

プロパティ

パターン

Date、Time、または DateTime 値の書式設定に使用できるさまざまな形式。

- 事前定義: ISO などのさまざまな日付形式に基づく標準の抽出形式。すべての事前定義済みの形式のリストについては、docs.oracle.com の Web サイトを参照してください。
- デフォルトのパターン: yyyy-M-d など、選択して使用できる一般的なパターンのセット。[ターゲット タイプ] を選択すると、リストから使用できるパターンが変化することに注意してください。
- 計算済み: 定数として、またはエクスプレッションを使用して作成できるパターン。このオプションは、[デフォルトのパターン] オプションと同じ形式を提供します。次の表に示すように、パターンは、日付の各部分に対応する文字および文字の組み合わせで構成されます。

文字	説明	例
G	期間 (BC または AD)	`G` は BC または AD に、`GGGG` は Before Christ (紀元前) または Anno Domini (紀元) に、`GGGGG` は B または A に一致します
z	タイム ゾーン ID	`z`、`zz`、`zzz` は CET に、`zzzz` は CET と中央ヨーロッパ標準時の両方に一致します
Z	タイム ゾーンの UTC オフセット	`Z`、`ZZ`、`ZZZ` は +0100 に、`ZZZZ` は GMT+01:00 に、`ZZZZZ` は +01:00 に一致します
Y	年	`yyyy` は 2022 に、`yy` は 22 に一致します
Q	四半期	`Q` は 1 に、`QQ` は 01 に、`QQQ` は Q1 に一致します
M	月	`M` は 7 または 07 に、`MM` は 07 に、`MMM` は Jul に、`MMMM` は July に一致します
w	週番号	`w` は 3 または 12 に、`ww` は 12 に一致します
d	日	`d` は 1 または 01 に、`dd` は 01 に一致します
E	曜日	`EEE` は Mon (月) に、`EEEE` は Monday (月曜日) に一致します
h	時間 (1 ~ 12 AM または PM)	`H` は 0 または 23 に一致します。`h` と `hh` は 01 または 12 に一致し、通常は AM または PM を示す **a** パターンを必要とします
a	AM または PM	`a` は AM または PM に一致します
m	分	`mm` は 00 または 43 に一致します
s	秒	`ss` は 00 または 43 に一致します
S	秒の端数	`SSS` は 123 に一致します

注意:

- パターンには、日付の各部分を区切る「-」(ハイフン)、「/」(スラッシュ) などの文字とスペースを含めることができます。
- `'Day' d 'of the month' MMMM 'in the year' yyyy` のように、テキストを一重引用符で囲むと、「Day 3 of the month July in the year 2022」というように、パターンにテキストを含めることができます。
- パターンの文字数によって、結果の形式が決まります。たとえば数字の場合、文字数が 1 であれば、結果には任意の数の数字が含まれることになります (`d` の結果は 1、01、31 のいずれにもなります)。それ以外の場合、文字数は結果の長さを表します。たとえば月名の場合、3 文字は短縮形 (Jul)、4 文字は完全形 (July) を表します。
  詳細については、docs.oracle.com の Web サイトを参照してください。

ロケール
使用可能なロケールのリストが含まれています。ロケールは言語と地域を指定します。言語と地域はハイフンで区切る必要があります (「en-US」)。日付を識別する方法は言語によって異なりますが、このオプションを使用すると、月名など、言語の特性を反映できます。たとえば、ロケールを en-US に設定すると、`MMMM d, yyyy` の結果は July 4, 2021 になります。fr-FR に設定すると、結果は juillet 4, 2021 になります。
タイム ゾーンはロケールの影響を受けないことに注意してください。

ツリーの凍結

ツリーの凍結ステップは、ワークフローでのステップの実行時にレコーダー ビューでアプリケーション ツリー更新を凍結するグループ ステップです。ツリーの凍結ステップ内のステップの実行時にアプリケーション ツリーはリロードされません。実行フローがツリーの凍結グループ外になると、アプリケーションツリーがリロードされます。このステップにより、テーブル、スプレッドシート、フォームなど、静的ウィンドウでの周期的な操作の実行中にパフォーマンスを大幅に向上させることができます。

ツリーの凍結ステップは、Chromium 組み込みブラウザでの使用を想定していません。組み込みブラウザでツリーの凍結ステップを使用すると、ツリーの凍結ステップ内に含まれる左クリック ステップ内での実行の進行に伴い、ツリーが更新されます。オートメーション デバイスでツリーの凍結ステップを使用すると、ツリーの凍結ステップ内に含まれる左クリック ステップ内で実行が進行しても、ツリーは維持されます (更新されません)。

グループ

このステップは、複数のステップを 1 つのグループに統合します。グループ ステップでは、グループ内でのみ利用可能なローカル変数を作成できます。ステップでローカル変数を使用する場合、ステップをそのローカル変数のあるグループに含めます。グループでステップを作成したり、カットアンドペースト操作を使用して既存のステップをグループに含めたりできます。

ガード チョイス

ガード チョイス ステップは、複数のアクションと関連付けられた多くの条件を設定する際に使用します。条件またはガードのいずれかが最初に満たされると、その関連付けられているアクションまたはステップが実行されます。これは一般的に、ボタンなどのインターフェイス エレメントに移動してクリックをする前に、エレメントが存在することを確認するために使用されます。無制限に待機しないように、タイムアウト ガードが追加されます。Kofax RPA では、ロケーションとタイムアウト ガードを使用して、ロボットが必要なエレメントを見つけ、設計どおりに動作するようにできます。

多くの場合、Kofax RPA では、クリック ステップまたは押すステップなどのステップの挿入時に自動的にガードが追加されます。次のガードが利用可能です。

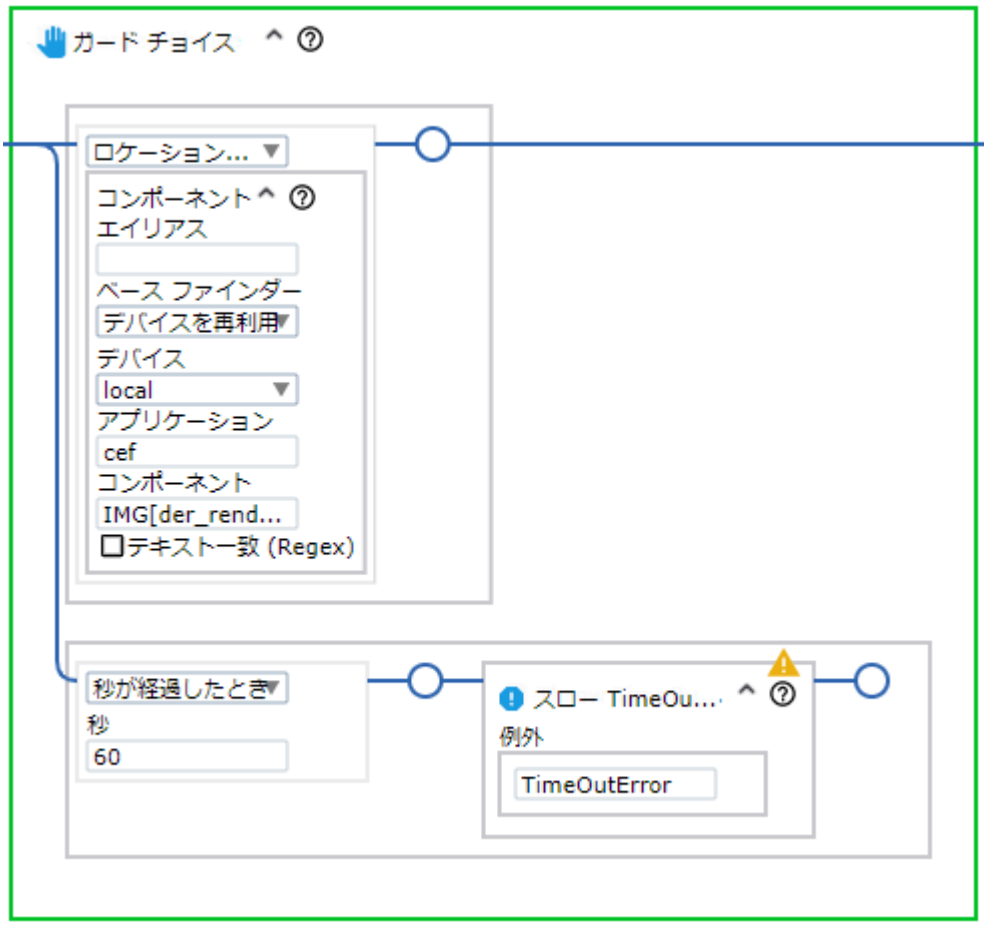
時間経過
これは、ロボットで次のステップを実行する前に指定された時間待機するタイムアウト ガードです。

デフォルトでは、以下のステップが [レコーダー ビュー] を通して追加されると、ステップに 60 秒のガードが挿入されます: クリック、マウスのスクロール、テキスト入力、値抽出、コンテンツ抽出、画像抽出、画像からテキスト抽出、ガード チョイス、およびキー プレス。

該当するアプリケーション
アプリケーションが指定したファインダーで見つかるようにするアプリケーション ガード。アプリケーションが見つからない場合は、見つかるまで待機します。複数のアプリケーションが見つかった場合は、アプリケーションが 1 つになるまで待機します。

該当しないアプリケーション
アプリケーションが指定したファインダーで見つからないようにするアプリケーション ガード。

該当するロケーション
エレメントが指定したファインダーで見つかるようにするロケーション ガード。



該当しないロケーション
エレメントが指定したファインダーで見つからないようにするロケーション ガード。

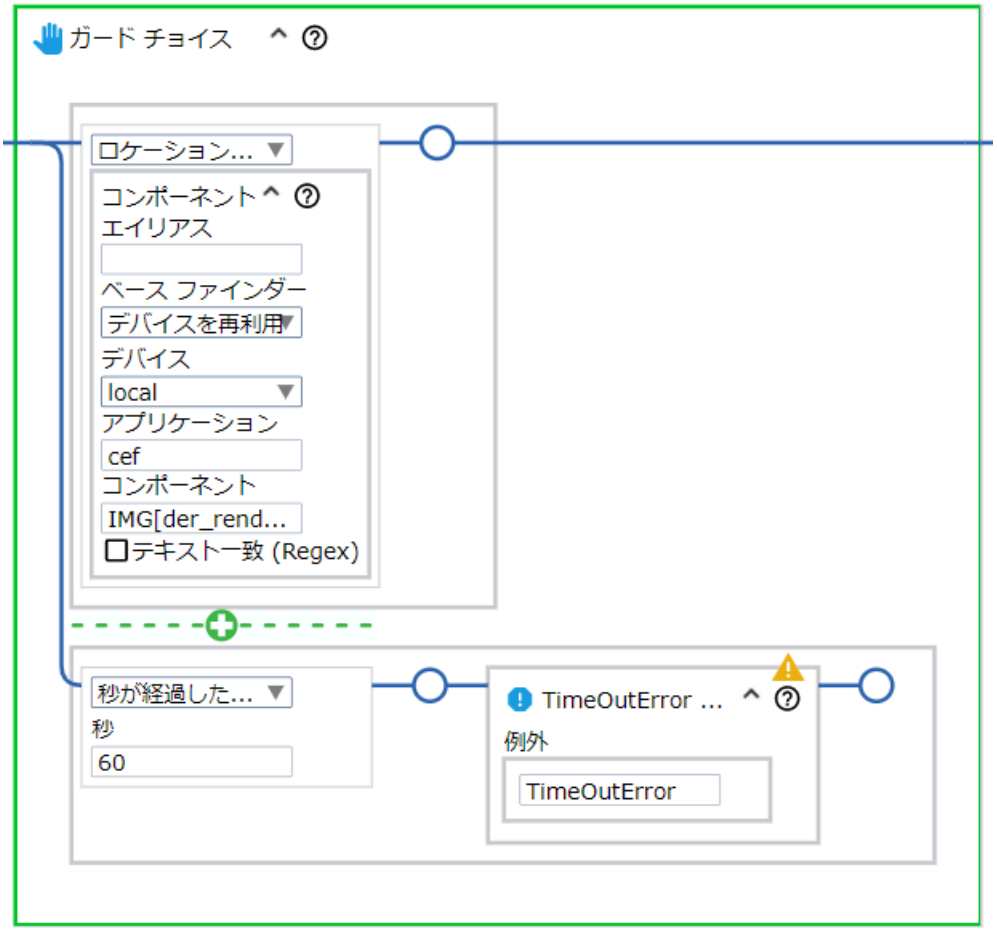
取り除かれたロケーション
指定したファインダーでエレメントを見つけ、次のステップの実行前にエレメントが除去されるまで待機します。

ツリーの変更停止
アプリケーション ツリーまたは Web ページ ツリーが変更されてからステップが実行されるまでに、指定された時間だけ待機するタイムアウト ガード。タイムアウトはミリ秒単位で指定されます。

ファインダーは以下のロケーション ガードごとに指定する必要があります。

手動でのガードの追加

ガード チョイス ステップにガードを手動で追加するには、ガード フレームの一番下の行にマウスを置き、緑のプラス記号が表示されるまでプラス記号をクリックしてガードを追加します。次の例をご覧ください。



ヒントとテクニック

ドキュメントを直接開く
最初にアプリケーションを起動してからドキュメントを開く必要はありません。代わりに、ドキュメントを開くだけで、関連付けられているプログラムが起動します。

アクセシビリティ検出を監視

- アプリケーションを開くとき：Desktop Automation サービスを使用して Adobe Reader を開くと、アクセシビリティ設定アシスタント ウィンドウが表示されます。通常これは 1 回限りの設定ですが、オンデマンドで生成されるデスクトップの場合、ロボットでの処理が必要になることがあります。

- ファイルを開くとき：PDF ファイルを開くとき、Adobe Reader でスクリーン リーダーでドキュメントを処理する必要があるかどうかの確認が表示されます。

テキストをフィールドに貼り付けたら、次のステップにガードを使用します。

ガードでは、テキスト フィールドのコンテンツが次のアクションのために貼り付けたものと一致することが検証されます。一致しない場合、貼り付け操作の直後に Enter を押しても、値が完全にフィールドに貼り付けられないことがあります。

このヒントはパスワード フィールドに適用されません。

KTA

このステップは、Kofax TotalAgility (KTA) サーバーに接続し、新しいジョブの開始、ジョブのステータスを取得、新しいドキュメントの作成、ジョブの変数を取得、またはジョブ イベントの提起を行います。

RPA を使用するには、Kofax TotalAgilityで [複数ログオンを許可] を設定します。

プロパティ

アクション
タスクに応じて、次のアクションから選択します。[ジョブの生成]、[ジョブのステータスを取得]、[ドキュメントの生成]、[ジョブの変数を取得]、[ジョブ イベントの提起]。

Kofax TotalAgility サーバー
使用する Kofax TotalAgility インストールの名前を指定します。新しい Kofax TotalAgility インストールの追加と設定を実行するには、Management Console 内の**[管理]** > **[設定]** > **[KTA** 設定**]** セクションに移動します。詳細情報は、KTA 設定を参照してください。

アクション

ジョブの作成

このアクションは、Kofax TotalAgility でジョブを開始します。RPA から開始できるジョブは、リリースされたバージョンのプロセスに限ります。このステップは、プロセス実行の完了を待機せず、Kofax TotalAgility からのジョブ ID を変数に割り当て、実行結果の監視を可能にします。

最初に、プロセスが含まれる Kofax TotalAgility カテゴリを選択します。該当するカテゴリの Kofax TotalAgility プロセスのドロップダウン リストが表示されます。次に、開始するプロセスを選択します。プロセスに属するパラメータのリストが表示されます。Kofax TotalAgility では、すべてのパラメータにデフォルト値があります。ロボットのデフォルト値を上書きするには、ここで必要なパラメータを選択し、新しい値を入力します。

ジョブ ステータスの取得
このアクションにより、KTA ステップで特定のジョブのステータス情報を取得します。ジョブは、ジョブの生成ステップから返されたステータスなどのジョブ ID によって識別されます。ステータス情報は、次の 2 つのフィールドで構成されます: 番号と説明 (次の表の「値」と「テキストとしてフォーマット」) 。

値	テキストとしてフォーマット
0	アクティブ
1	完了
2	終了
3	サスペンド
4	完了待ち
5	ロック
6	評価の準備待ち
7	保留
8	完了待ち
9	ケースの完了待ち
10	完了待ちの停止
11	ケースの完了待ちの停止

ドキュメントの生成
このアクションは、Kofax TotalAgility でドキュメントを作成します。ドキュメント ID が ロボットに返されます。このドキュメント ID は、ドキュメント タイプの変数として [KTA ジョブを作成] ステップに渡すことができます。作成できるドキュメント タイプのリストについては、Kofax TotalAgility のドキュメントを参照してください。

次のパラメータは RPA から設定できません。

- チェックリスト
- データ バックボーン
- ダイナミック コンプレックス
- XML 式

ジョブの変数を取得

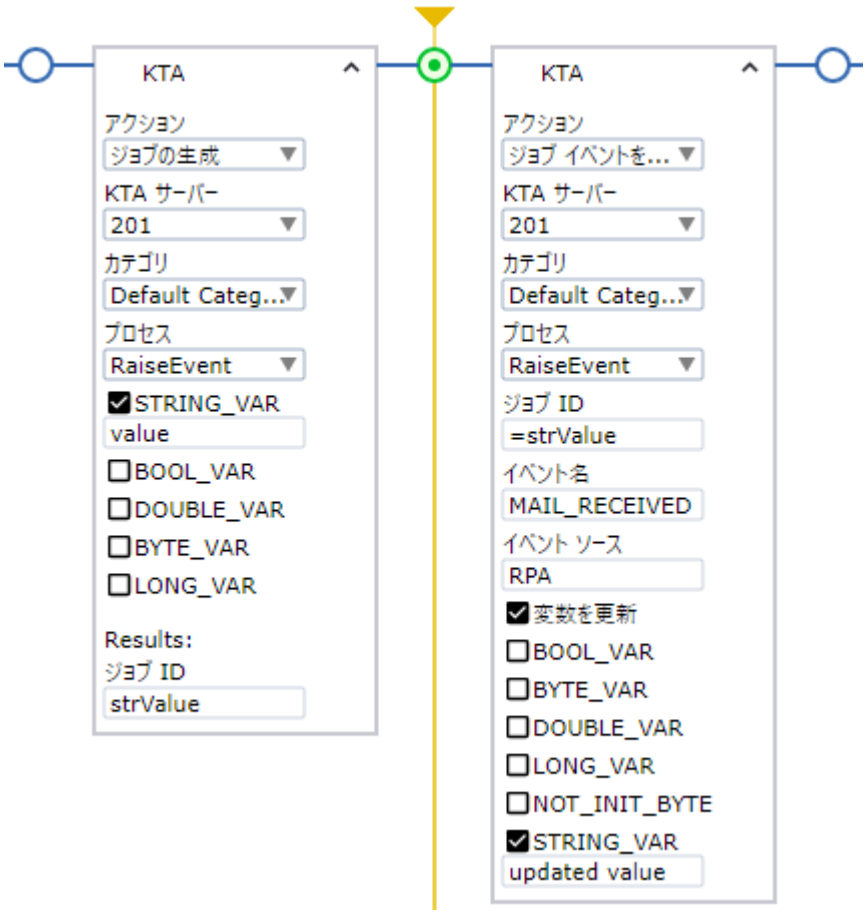
このアクションにより、KTA ステップで特定の Kofax TotalAgility 変数の値を取得します。値を取得する Kofax TotalAgility 変数を指定するには、次のフィールドに入力します。

- ジョブ **ID**: Kofax TotalAgility ジョブ ID を入力します。
- 変数の **ID**: Kofax TotalAgility 変数 ID を入力します。

返された情報は、[値] フィールドおよび [変数が見つかりました] フィールドで構成される [結果] に表示され、ジョブで変数が見つかったかどうかが示されます。

ジョブ イベントの提起

このアクションを使用して、イベントを Kofax TotalAgility に送信し、Kofax TotalAgility 変数を変更します。



Kofax TotalAgility ジョブが作成された後に、[ジョブ イベントの提起] アクションを使用して、次のフィールドに入力します。

- ジョブ **ID**: Kofax TotalAgility ジョブ ID を入力します。
  [ジョブの生成] アクションで使用した値と同じ値を入力できます。
- [イベント名]: Kofax TotalAgility イベント名を入力します。

  イベント名は、大文字と小文字が区別されません。

- [イベント ソース]: Kofax TotalAgility イベント ソースを入力します。

変数を変更するには、最初に [変数を更新] オプションをオンにします。

次に、リストから変数を確認し、値を変更します。

[ジョブ イベントの提起] アクションでは、初期値だけでなく、すべての Kofax TotalAgility 変数を変更できることに注意してください。

ドキュメント検証の例

PDF ドキュメントを検証するには、次の手順を実行します。

1. [**KTA**] ステップをロボットに追加します。
2. [ファイルの読み込み] ステップを追加し、PDF ドキュメントを変数 `DocData` にロードします。
3. [**KTA**] ステップで、[ドキュメントの生成] アクションを追加して、ドキュメントを Kofax TotalAgility のデータベースに保存します。以下のオプションを記入します。他のオプションは Kofax TotalAgility サーバーに依存します。
   - ドキュメント データ: 処理するドキュメントで `DocData` 変数を指定します。
   - **MIME** タイプ: ドキュメントの形式 (`application/pdf`) を指定します。
   - ドキュメント **ID**: Kofax TotalAgility によってドキュメントに割り当てられた ID を指定します。
4. [**KTA**] ステップで、[ジョブの生成] アクションを使用して、ドキュメントを検証する Kofax TotalAgility プロセスを開始します。
   - プロセス: Kofax TotalAgility で作成した、ドキュメントを処理できるプロセス名を指定します。
   - **MYKTADOCUMENT**: ドキュメント ID を渡すこの Kofax TotalAgility プロセスのパラメータ。
   - **[ジョブ ID]**: Kofax TotalAgility によってプロセスに割り当てられた ID を指定します。
5. [ジョブのステータスを取得] ステップを追加して、ジョブが確実に成功するようにします。以下のオプションを記入します。
   - ジョブ **ID**: KTA ステップの [ジョブの生成] アクションの ID を指定します。
   - 値: ジョブ ステータス値を含む変数を指定します。
   - テキストとしてフォーマット: ステータスの説明を含む変数を指定します。

[ジョブのステータスを取得] ステップによって「評価の準備待ち」ステータスが返される場合は、Kofax TotalAgility のアクティビティを手動または自動で完了するように設定できます。アクティビティが完了すると、[ジョブのステータスを取得] ステップを実行した後に「完了」ステータスが返されます。

ループ ステップ

このセクションでは、ロボットのループ ステップについて説明します。

次の 6 つのループ ステップがあります: ループ ステップ、条件付きループ ステップ、要素の繰り返しステップ、電子メールごとにステップ、ディレクトリの反復ステップ、およびデータベース照会ステップ (データベース カテゴリにも属しています)。

以下はすべてのループ ステップに共通です。

1. すべてのループ ステップには、オプションのイテレーション変数があります。
2. ブレーク ステップを使用して、ループ ステップを終了できます。
3. コンテニュー ステップを使用して、次のイテレーションにスキップすることができます。

イテレーション変数

すべてのループ ステップには、オプションのイテレーション変数があります。これは、次の特性を持つステップで定義できる整数変数です。

- 変数の初期値が 0 で、つまり、開始イテレーションでは変数が 0 である。

- ループの各イテレーションの最後に 1 つずつ増加する。
- 変数がループ ステップに対してローカルであり、ループ外からアクセスできない。
- 変数が読み取り専用で、つまり、割り当てステップを使用して変数を変更することができない。

ループ内のイテレーション変数を参照するには、[イテレーション変数] を選択し、イテレーションを格納する変数の名前を入力します。

前のファインダーを参照する

Kofax RPA 11.3.0 以降、以前のすべてのファインダーは、ループ ステップ本体の開始時にクリアされます。ループ ステップを含むロボットが以前のバージョンの製品で作成されている場合は、以前のファインダーを名前付きファインダーに置き換えることをお勧めします。

ブレイク

このステップを使用して、 ループ ステップを終了できます。ループ ステップ内でのみ使用する必要があります。1 つのループ内で複数のブレーク ステップを使うことができます。

コンテニュー

このステップでは、ループステップで次のイテレーションにスキップできます。ループ ステップ内でのみ使用する必要があります。

ディレクトリの反復

この手順を使用して、ディレクトリ内のすべてのエントリ (ファイル、ディレクトリ、およびその他のタイプのオブジェクト) を反復処理します。

このステップは、アプリケーションをレコーダー ビューで開き、最初のエントリを表示します。エントリのプロパティは、テーブル ビューとツリー ビューで使用できます。このステップは、ディレクトリ内のエントリを反復処理するループ ステップとして機能します。反復処理を行うたびに、次のエントリのプロパティでビューの内容が更新されます。

ループは、最初の反復処理の前にディレクトリ内のエントリのスナップショットを取得し、このスナップショットに対して反復処理を行います。ループ中にディレクトリの内容を変更しても、反復処理には影響しません。アプリケーションは静的であり、エントリのプロパティがループ内のステップによって変更されても更新されません。たとえば、エントリを削除すると、アプリケーションに表示されます。エントリの名前を変更すると、最初の名前で表示されます。エントリを作成しても、アプリケーションには表示されません。

すべてのエントリのイテレーションが完了する (またはロボットが例外やブレイク ステップによってループを終了する) と、アプリケーションは自動的に閉じます。また、ロボットが実行を停止した場合にもアプリケーションは閉じます。

ロボットは、イテレーション変数と続行およびブレーク ステップを使用することで、他のループ ステップと同様にループの実行を制御できます。

抽出ステップを使用すると、通常どおりにアプリケーション ツリーからプロパティを抽出できます。

ローカル Desktop Automation モードでのローカル ファイル システムへのアクセスを含むファイル システム アクセスを有効にするには、RoboServer 設定アプリケーションの [セキュリティ] タブで [ファイル システムとコマンド ラインのアクセスを許可] オプションを選択します。

RoboServer がローカル ファイル システムにアクセスできないように設定されている場合、ロボットは実行できますが、ステップがローカル デバイス上で [直接アクセス] に設定した [ファイル アクセス] を使用するとエラーが発生します。ただし、このステップは、[ファイル アクセス] が [**RFS** 経由] に設定されているローカル デバイス上で機能します。リモート デバイス上では、[直接アクセス] と [RFS 経由] が使用できます。

ディレクトリの反復ステップは、アプリケーション、ファイル システム、およびループのカテゴリに属します。

プロパティ

以下のプロパティを使用して「ディレクトリの反復」ステップを設定します。

名前
ステップの名前。

デバイス
使用するリファレンス名を選択します。このリファレンス名は、「ロボットを呼び出す」ステップの [デバイス] プロパティで指定します。

ファイル アクセス
ファイルのアクセス方法を指定します。
指定したローカル デバイスまたはリモート デバイスのディレクトリ内のエントリに対して反復処理を行うには、[直接アクセス] を選択します。
ロボット ファイル システムのディレクトリ内のエントリを反復処理するには、[**RFS** 経由] を選択します。

ディレクトリの名前
エントリが反復処理されるディレクトリへのパスを指定します。
たとえば、`C:\Directoryname` です。

アプリケーション名
ステップで開くアプリケーションの名前を設定します。
この必須パラメータで指定された名前は、アプリケーションの名前属性とタイトル属性で使用されます。

エラーを回避するために、ロボット内のアプリケーションにはそれぞれ異なるアプリケーション名を割り当てます。

イテレーション変数 **(**オプション**)**
イテレーション データを格納する変数を指定します。

要素の繰り返し

要素の繰り返しステップは、アプリケーション ツリー内のノードを反復処理します。[スコープ ファインダー] と呼ばれるコンポーネント ファインダーがあります。これは、ツリーの一部を特定し、反復するノードを検出します。イテレーションは、スコープ ノードの下にないノード (スコープ ファインダ-によって検出されたノード) についてはループしません。[スコープ ファインダー] は、ステップ内のすべてのファインダーと似ています。これは、このステップが動作するツリーの一部を検出するためです。スコープ ファインダーは常に、要素の繰り返しステップの中で一意となる名前を持つ必要があります。

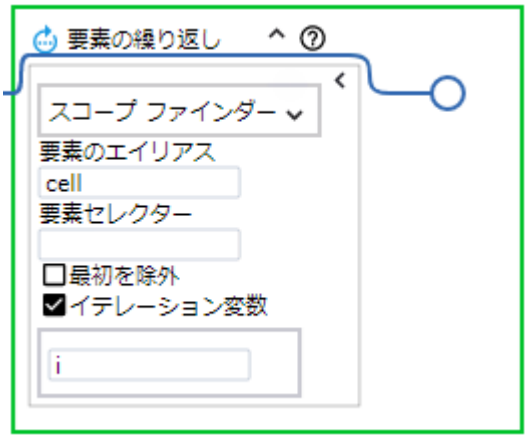
要素の繰り返しステップの要素ファインダーは、ファインダーの名前と "> DIV" などの相対セレクターで設定されています。スコープ ファインダーと同様に、要素ファインダーは常に、要素の繰り返しステップの中で一意となる名前を持つ必要があります。

相対セレクターは、ループする エレメントを検出するために使用されます。このセレクターは、別のセレクターと結合されて実際の新しいセレクターを形成するようになっているため、相対セレクターと呼ばれます。スコープ ファインダーにセレクター `"DIV[class="someClass"]"` があり、要素ファインダーのセレクターが `"> DIV"` の場合、結合セレクターは `"DIV[class="someClass"] > DIV"` のようになります。反復する エレメントを検出するために使用される実際のファインダーはスコープ ファインダーと本質的に同じですが、セレクターがこの新しい結合セレクターに置き換えられます。

要素の繰り返しステップは次のように動作します。開始イテレーションでは、結合されたファインダーにより最初に検出された エレメントは、要素ファインダーの名前となる エレメントです。次のイテレーションで検出される エレメントは、前の エレメントの次に見つかる エレメントです。繰り返しステップの実行中にツリーが変更され、新しい エレメントがツリーに表示された場合、現在のイテレーションの エレメントの前または後に現れるかどうかに応じて、この新しい エレメントがその後のイテレーションに含まれる場合と含まれない場合があります。

要素ファインダーで見つかった エレメントは、他のファインダーのリファレンスとしてループの本文内で使用されます。

以下は、要素の繰り返しステップの一例です。



レコーダー ビューの [ループ] メニュー

「要素の繰り返し」ステップは、他のステップと同様にエディター ペインに直接挿入できますが、レコーダー ビュー内で右クリックし、ショートカット メニューを使用して挿入する方が簡単です。[ループ] と呼ばれるメニュー項目を選択します。ループには次の 4 つのサブメニューがあります。

- すべての同位層
- 各 <タグ名> 同位層
- 各 <タグ名> (<レベル>) 上位層
- テーブル行繰り返し

これらのサブメニューは、現在選択されているエレメントに応じてさまざまなエレメントをループする要素の繰り返しステップを作成するのに役立ちます。Design Studio では、要素の繰り返しステップにガード チョイス ステップが挿入されます。これにより、ループ ステップが実行される前にスコープ ファインダーがツリー内に確実に表示されるようになります。

すべての同位層
このメニュー項目は、選択されたタグのすべての同位層をループする要素の繰り返しステップを挿入します。スコープ ファインダーは、選択されたエレメントの親 (スコープ ノードと呼ばれる) を検出し、エレメント セレクターは "> *" となります。つまり、スコープ ノードの下にある任意の子ノードを検出します。

各 **<タグ名>** 同位層
このメニュー項目は、選択されたタグのすべての同位層を同じタグ名でループする要素の繰り返しステップを挿入します。スコープ ファインダーは、選択されたエレメントの親 (スコープ ノードと呼ばれる) を検出し、選択されたエレメントがタグ名 P を持つ場合、エレメント セレクターは "> P" となります。

各 **<タグ名> (<レベル>)** 上位層

このメニュー項目は、ロボットで組み込みブラウザを使用する場合にのみ使用します。このコマンドは、ループのための適切な上位層ノードを検索します。そうすることで、自身に似た同位層ノードが複数あるノード、または以下のタグ名のエレメントを持つ上位層ノードを探します。"TR"、"LI"、"TD"、"TH"、"DD"、"OPTION"、"PARAM"以下の場合、2 つのノードが類似します。

- タグ名が同じで、クラス属性がない。
- タグ名が同じで、同じクラス属性である。

次の HTML で内部の P タグを 1 つ選択すると、そのループは囲んでいる DIV タグ内の 1 つの P タグを繰り返さずに、P タグを含む DIV タグをループします。

例：

```
<DIV>
  <DIV>
    <P>1</P>
  </DIV>
  <DIV>
   <P>2</P>
  </DIV>
</DIV>
```

メニュー項目の丸括弧内のレベルは、選択されたエレメントのいくつ上のレベルに実際の要素の繰り返しが配置されているかを示します。上記の例では、このレベルは 1 です。メニュー項目に表示されるタグ名は、実際の要素の繰り返しのタグ名です。

テーブル行繰り返し
このメニュー項目は、ロボットで組み込みブラウザを使用する場合にのみ使用します。このコマンドは、表のすべての行を繰り返し処理する要素の繰り返しステップを挿入します。

要素の繰り返しステップの使用

要素の繰り返しステップを使用する際には、いくつかの点を考慮する必要があります。

イテレーションのスキップ
一部の初期ノードや各 2 番目のノードなど、一部のエレメントをループ中にスキップする場合は、ループ内の最初のステップとして、テストが true である場合に継続される条件付きステップを挿入します。たとえば、次の条件では、ループはすべての偶数回の繰り返しをスキップします。

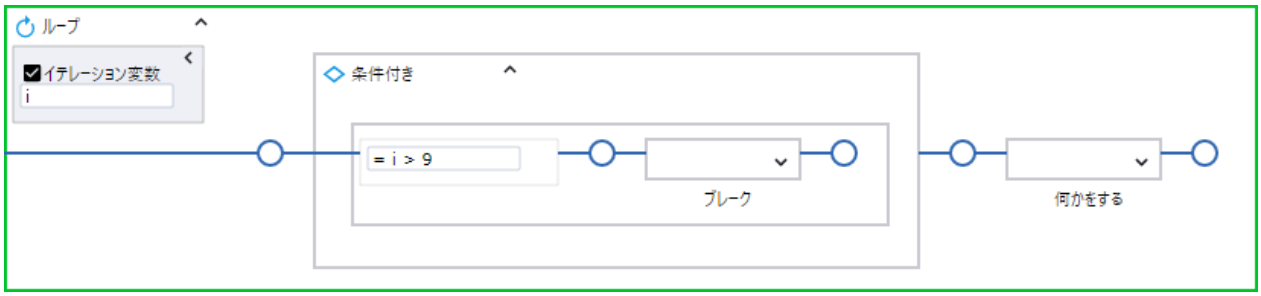
```
=i % 2 == 0
```

ループ ステップ内のファインダー
Kofax RPA は、検出されたエレメントに相対的な要素を自動的に検出します。レコーダー ビューで要素を右クリックしてアクションを挿入するときに、この要素が検出された指定エレメント内にある場合、生成されたファインダーは、次のように検出された要素に対して相対的になります。

結果は下図のようになります。実際の結果は、選択したエレメントと、要素ファインダーに付けた名前によって異なります。

ループ

ループ ステップは、ループを終了するブレーク ステップ、または次のイテレーションへスキップするコンテニュー ステップを伴うステップのグループです。条件でループするには、ループ内で条件を使用します。デバイスで何かが発生するまで待機してループするには、代わりにガード チョイスを使用します。

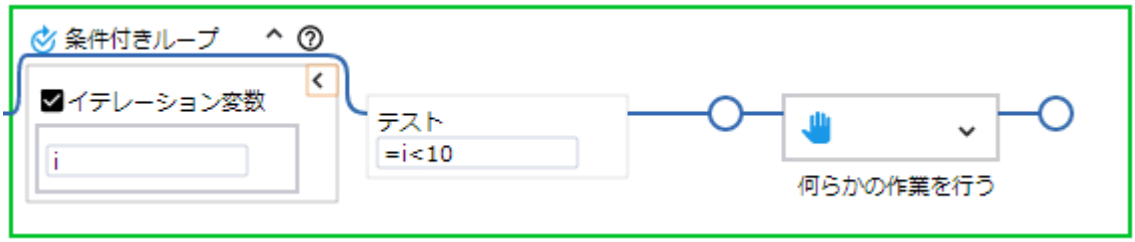


[次のイテレーションに移動] ボタンを押して、同じフロー ポイントに再度到達するまで実行します。フロー ポイントが一部のイテレーションでスキップされた場合、ループは複数回実行できます。これ以上イテレーションがなくなると、実行はループ ステップ外のフロー ポイントで停止します。

条件付きループ

条件付きループ ステップは、追加のプロパティを持つ以下のループ ステップに似ています。[テスト]これは、ステップの各イテレーションの前に評価されるテストです。テストが true である場合、ステップの本文が実行されます。テストが false である場合、ループの実行は終了します。

以下は、「何らかの作業を行う」ステップを 10 回実行し、テストが失敗したときにループを終了する条件付きループ ステップです。



電子メール (旧ステップ)

このステップは廃止されています。このステップを使用する既存のロボットは引き続き実行することができますが、新しいメール送信ステップを作成することはできません。代わりに、電子メール ステップを使用してください。

このステップは、メール サーバーで 1 つ以上のフォルダを開いてそれらをツリーに表示し、電子メールにアクセスする場合に役立ちます。開いているフォルダは、アプリケーション ツリーに表示されます。ネストされたフォルダは表示されませんが、個別に開くことができます。ツリーには、メッセージごとに、受信者と「cc」のアドレスおよび送信者が一覧表示されます。次のヘッダーが含まれています。送信者、宛先、メッセージ ID、日付、件名、およびコンテンツ タイプ。

プロパティ

アプリケーション アクション: メールボックスを開く
このアクションは、選択したメール ボックスに接続して、開きます。

プロトコル
メール サーバーとの接続に使用するメール プロトコルを選択します。選択肢は「imap」、「pop3」、およびそれらの安全なバリアント「imaps」、「imaps+tls」および「pop3s」です。

メール サーバー
メール サーバーの名前またはアドレスを指定します。

メール サーバー ポート (オプション)
メール サーバーのポート番号を指定します。指定しない場合は、選択したプロトコルの標準ポートが使用されます。

アカウント名とパスワード
メール サーバーにログインするためのクレデンシャルを入力します。

メールボックスのパス (オプション)
メール サーバーで開くフォルダ名を 1 つ以上入力します。パス内に複数のフォルダ名を含める場合は、スラッシュで区切る必要があります。受信メールが保存されているフォルダを開くには、パスや名前を指定しないでください。

例: `Junk E-Mail`、`Subscriptions/Mailling lists`.

`Junk E-Mail;Subscriptions/Mailling lists` のようにセミコロンで区切って複数のパスを指定すると、複数のフォルダを開くことができます。

POP3 はフォルダをサポートしていないため、pop3 または pop3s プロトコルを使用する場合は、このフィールドを空白のままにします。

電子メール ポーリング間隔 (秒) (オプション)
開いているメール フォルダが連続して更新される場合の間隔を示す秒数。省略した場合、フォルダは 15 秒間隔で 1 分あたり 4 回ポーリングされます。この値は、プロトコルが imap または imaps に設定されている場合にのみ関係します。

開いているフォルダは、アプリケーション ツリーに表示されます。フォルダには、0 個以上のメッセージを含めることができます。ネストされたフォルダは表示されませんが、個別に開くことはできます。メッセージごとに、受信者と「cc」のアドレスおよび送信者が一覧表示されます。次のヘッダーが含まれています。送信者、宛先、メッセージ `ID`、日付、件名、およびコンテンツ タイプ。各メッセージには、フォルダ内の場所を示す [インデックス] プロパティがあります。インデックスは 1 から始まる連続する値です。メッセージを追加または削除すると、インデックスが変更されます。

アプリケーション ツリーに電子メール メッセージを入力するのは、比較的遅いプロセスです。これは非同期で実行されます。ツリーへのメッセージの読み込みは、メールボックスを開くアクションによって開始されますが、その後しばらく続く場合があります。また、imap および imaps プロトコルの場合、メールボックスは定期的にポーリングされ、ツリーに新しいメッセージを追加したり、削除されたメッセージを削除したりします。

これにより、ロボットがメッセージを処理している間に、ツリー内のメッセージの数が変化する可能性があります。特に、メッセージの「`index`」プロパティは、新しいメッセージが到着した場合、またはメッセージが削除された場合に変化する可能性があります。index プロパティを使用して、複数のアクションが実行されるメッセージを選択する場合は、次の手順を実行します。ファインダーでインデックスを含むメッセージを選択し、「`handle`」プロパティを抽出してから、後続のファインダーでこの値を使用します。`handle` プロパティは、1 つのセッションのメールボックス内のメッセージごとに一意であり、変更されません。

アプリケーション アクション**:** メールボックスを閉じる
このアプリケーション アクションにより、開いているすべてのフォルダを含むアプリケーションが閉じます。

アプリケーション アクション**:** フォルダを開く **(imap**、**imap+tls** および **imaps** プロトコルの場合**)**
このアプリケーション アクションは、パラメータとしてフォルダ パスを受け取ります。このアクションを行うと、メール サーバー上に追加のメール フォルダが開き、ツリー ビューのツリーに追加されます。

コンポーネント アクション

次のコンポーネント アクションが使用できます。

フォルダを閉じる **(imap**、**imap+tls** および **imaps** プロトコルの場合**)**
このアクションは、メール フォルダを閉じます。

電子メールのコピーと電子メールの移動 **(imap**、**imap+tls** および **imaps** プロトコルの場合**)**
このアクションは、選択したメール メッセージを、指定したターゲット フォルダにコピーまたは移動します。このアクションを使用できるかどうかは、メール サーバーでのコピーおよび移動操作のサポートに応じて異なります。

電子メール コンテンツを取得
このアクションは、RFC 822 形式のメッセージとヘッダーを含む文字列値を返します。

電子メールを削除
このアクションは、メール メッセージを削除します。

POP3 または **POP3S** プロトコルに関する制限事項

- フォルダはサポートされていません。メール ボックスを開くアクションのフォルダ名は、空または「INBOX」にする必要があります。
- 開いているフォルダの内容を更新することはできません。「INBOX」の変化を監視するようにロボットを設定した場合、ロボットは受信トレイを開く、検査する、閉じるという処理をループ内で実行します。
- サーバーに更新のポーリングは行われないため、接続がタイムアウトする場合があります。時間がかかる可能性のある他のタスクを実行している場合、ロボットはメール ボックスを閉じ、メール操作が再度実行されたときにメール ボックスを再度開きます。
- 一部の POP3 サーバーには、他のインターフェイスの POP3 接続を介して行われた変更が反映されません。たとえば、ロボットが POP3 プロトコルを使用して Gmail アカウントからのメールを削除した場合でも、Web ブラウザ インターフェイスには引き続きそのメールが表示されることがあります。ロボットの場合、電子メールは削除されます。
- 「電子メールを削除」ステップを行うと、メールに削除のマークが付き、ツリーから削除されますが、フォルダまたはメール ボックスを閉じるまでメールはサーバーに残ります。

マウス移動

このステップでは、マウスを画面の指定された場所に移動します。マウス座標は、ウィンドウの左上隅を基準としています。X は左から右に移動する横軸で、Y は上から下に移動する縦軸です。このステップは、マウス ホバー アクションとドラッグアンドドロップ操作に使用できます。

さらに、自動化されたコンピュータでハードウェアのキーボードとマウスをシミュレートすることもできます。詳細については、『Kofax RPA インストール ガイド』の「Desktop Automation サービスのインストール」を参照してください。

プロパティ

オフセット

- なし: 座標オフセットを使用せず、選択した工レメントの中央に移動します。以下に対応します。
  x=0、y=0 を中央に相対的に設定
- 使用: 次のオプションを使用して、オフセットをピクセルで指定します。
  次を基準
  このオプションは、オフセットを計算する起点を指定します。
  - 左上: ウィンドウ、または x=0 および y=0 で選択された工レメントの左上隅。
  - 上: ウィンドウ、または y=0 で選択された工レメントの上枠の中央。
  - 右上: ウィンドウ、または y=0 で選択された工レメントの右上隅。
  - 左: ウィンドウ、または x=0 で選択された工レメントの左枠の中央。
  - 中央: ウィンドウまたは選択された工レメントの中央。
  - 右: ウィンドウ、または選択された工レメントの右枠の中央。
  - 左下: ウィンドウ、または x=0 で選択された工レメントの左下隅。
  - 下: ウィンドウ、または選択された工レメントの下枠の中央。
  - 右下: ウィンドウ、または選択された工レメントの右下隅。

X
選択された起点を基準とする水平オフセットを指定します。正数はマウスを起点の右に移動します。負数はマウスを起点の左に移動します。

Y
選択された起点を基準とする垂直オフセットを指定します。正数はマウスを起点から下に移動します。負数はマウスを起点から上に移動します。

通知

このステップでは、自動化されたデバイスのタスクバーの通知領域にメッセージが表示されます。このステップは、アテンデッド オートメーション ロボットのトリガー チョイスで使用するために設計されています。ロボットによってトリガー イベントが阻止されていて、ロボットの実行中にユーザーがキーボードやマウスを使用できない場合は、このステップにより、ロボットの実行中に起きている内容がユーザーに通知されます。このステップには次のパラメータがあります。

プロパティ

- [デバイス]: デバイス名。
- [タイトル]: 通知メッセージのタイトル。
- [メッセージ]: 通知メッセージ。
- [アイコン]: アイコンなし、情報、警告、およびエラーから選択します。

開く

「開く」ステップは非推奨です。このステップは他の複数のステップに置き換えられ、それぞれが個別のアプリケーション専用になりました。各ステップには、専用アプリケーションを自動化するために必要な一連のオプションがあります。

- Web サイトを開く「ブラウザ」ステップ。
- スプレッドシートの操作に役立つ「Excel」ステップ
- Document Transformation画像およびテキスト ドキュメントから情報を抽出して使用することができる「Document Transformation」ステップ。
- Kofax TotalAgility (KTA) サーバーの操作に役立つ「KTA」ステップ。
- ターミナルに接続して自動化する「ターミナル」ステップ。
- Windows デスクトップを操作して、Windows アプリケーションを実行するのに役立つ「Windows」ステップ。

「開く」ステップでロボットを使用することはできますが、新しい「開く」ステップを作成することはできないことに注意してください。

プロパティ

デバイス
アプリケーションを開くデバイスを選択します。

URI

- 開くアプリケーションまたは Web サイトへのパスを指定します。パスでスラッシュを使用します。
  例:
  - `C:/Program Files/SAP/FrontEnd/SAPgui/saplogon.exe`
  - `="C:/Program Files/SAP/FrontEnd/SAPgui/saplogon.exe"`
  - `https://www.google.com`
  - `about://version`

  詳細については、Web サイトのアクセスを参照してください。
- 組み込み Windows アプリケーションの場合、`calc.exe` などのプロセス名を指定できます。
- 組み込み Excel ドライバーを開くには、次のように指定します。
  スプレッドシートを新規作成する場合は `excel://new`
  既存のスプレッドシートを開く場合は `excel://<スプレッドシートのフル パス>/<スプレッドシート名>.xlsx`
  詳細については、「Excel の補足ドキュメント」を参照してください。
- RDP 接続の場合、以下を指定します。
  `rdp://<ドメイン>\<ユーザー名>:<パスワード>@<ホスト名>?<param1>=<value1>&<param2>=<value2>`
  利用可能なパラメータの場所は、以下のとおりです。
  - d: ドメイン (URL にユーザー名の一部としてドメインを入力)
  - g: デスクトップ ジオメトリ (WxH)
  - a: 接続カラーの深度

- Z: ログイン中に別の画面を閉じるまで待機する秒数を指定します (下記参照)
- kapow-keep-awake: RDP セッションを維持するために送信するダミーのキーストロークの間隔を秒単位で指定します。デフォルトは 30 秒です。キーストロークを無効にするには、「`kapow-keep-awake=0`」のようにゼロ (0) を指定します。

例: `rdp://admin:AdminPassword@Server1`

パスワードまたはホスト名に、URL で受け入れられない文字 (バックスラッシュなど) が含まれている場合は、エンコードする必要があります。

接続しているシステムがユーザーのログイン時に別の画面を表示するように設定されている場合は、`rdp://admin:AdminPassword@Server1?Z=3` のように Z パラメータを使用してログイン中に別の画面を閉じるまで待機する秒数を設定します。この画面を閉じないと、アクションが失敗することがあります。

- RDP 接続を使用した [開く] ステップでは、接続が確立されるまで待機してからロボットの実行を継続します。リモート接続に失敗すると、エラー メッセージが表示されます。
- RDP 接続には、明示的に指定された同じ解像度と色深度のパラメータを常に使用してください。Windows 8 および Windows 10 は、32 ビット以下の色の深度をサポートしません。そのため RDP から送信される接続カラーの深度を変更する要求は、これらの Windows バージョンでは無視されます。

また、Chromium 組み込みブラウザで Cookie を使用する場合にも、「開く」ステップを使用します。詳細については、「Chromium 組み込みブラウザでの Cookie の管理」を参照してください。

出力値

このステップはロボットから値を返却します。

依存ロボット (ベーシック エンジン ロボットから呼び出される) をスタンドアロン ロボットに変換する場合、戻るステップを出力値ステップに置き換えます。

プロパティ

名前
ステップの名前。

変数
変数の名前。保存されている値を返却する必要がある変数を入力します。

[出力値] ステップの実行後、すべての値が収集されますが、状態ビューには最新の値のみが表示されます。Design Studio の **[Design Studio 設定]** > **[ロボット エディター]** メニューに表示される出力値の最大数を変更できます。

詳細については、「ロボット エディター 」を参照してください。

PDF

PDF のステップは、SignDoc 機能を使用して PDF ドキュメントと署名ドキュメントからコンテンツを抽出するために役立ちます。

CentOS/Red Hat Enterprise Linux 7.x オペレーティング システムでは、PDF 抽出機能はサポートされていません。

[レコーダー ビュー] には、PDF ドキュメント ツリーの単一ページと抽出されたテキストが表示されます。ロボットは、[アプリケーション アクション] メニューで利用できる [次のページ]、[前のページ]、および [ページに移動] アクションを使用して、ドキュメント内を移動します。メニューは、[レコーダー ビュー] のアプリケーション タブを右クリックすると表示されます。

テキスト抽出の結果は、PDF データの内部データと構造に依存します。テキストは、PDF ドキュメントの書式設定およびデータの基になるアクセシビリティに基づいて分割され、ページ境界の外側にあるテキストや、重複する要素によって非表示になっているテキストが含まれる場合があります。必要なアクセシビリティ データが (通常は古い) PDF ドキュメントに含まれていない場合は、OCR を使用してテキストを抽出するために画像からテキスト抽出ステップを使用する必要がある場合があります。

[テキストを抽出] アプリケーション アクションおよび [テキストを抽出] コンポーネント アクションを使用して、ページの特定の領域から構造化テキストを抽出することができます。

プロパティ

アクション

[開く] を選択して PDF ファイルをロードします。

ドキュメント ソース

- ローカル ファイル: **[ファイルのパス]** フィールドで、ローカル ファイル システム内のファイルのパスを指定します。
- ロボット ファイル システム: [ファイルのパス] フィールドで、ロボット ファイル システム内のファイルのパスを指定します。
- バイナリ: バイナリ形式の PDF ドキュメントを含む変数またはエクスプレッションを指定します。

パスワード

必要に応じて、このオプションを選択して、PDF にアクセスするためのパスワードを指定します。

ページ番号

必要に応じて、ドキュメントを開いた後に表示する物理ページを指定します。このプロパティが指定されていない場合は、最初のページが表示されます。

アプリケーション アクション

アクション	説明
ページに移動	指定したページに移動します。
次のページ	次のページに移動します。

アクション	説明
前のページ	前のページに移動します。
テキストを抽出	ページ領域からテキストを抽出して、選択した変数に挿入します。 テキストを抽出する場合は、次のオプションを指定します。 • **X**: ページの左端からの水平オフセットを設定します。 • [**Y**]: ページの上端からの垂直オフセットを設定します。 • [幅]: 幅を指定します。 • 高さ: 高さを指定します。 • 包括: 境界ボックスに、指定した領域を超えた文字が含まれるようにする場合に選択します。 デフォルトでは、このオプションは選択されていません。つまり、境界ボックスには、指定した領域内に完全に収まる文字のみが含まれます。 すべてのユニットはデバイス ツリーの座標内にあります。
画像を挿入	ローカル フォルダからドキュメント内の選択したページに JPEG または PNG 画像を挿入します。 RFS フォルダはサポートされていません。 画像は、ページの左上隅を基準として、画像の左上隅の X 座標と Y 座標に基づいて配置されます。 サポートされている単位は次のとおりです。 • デバイス ツリーの座標 • インチ/センチメートル • ページ サイズに対する相対値 (パーセンテージで指定) 以下のルールに注意してください。 • 負の値は反対側のマージンを指定します。X 値が -1 インチの場合、画像は右マージンから 1 インチの位置に配置されます。 • いずれかの寸法 (幅または高さ) が 0 の場合、画像は画像の他の寸法とアスペクト比に基づいて配置されます。 • アスペクト比に基づいて画像のサイズを変更すると、画像は指定した領域の中央に配置されます。 • これらの計算の後に、X、Y、幅、および高さの最終値がページの寸法または他のページ変換に対して検証されることはありません。このため、画像の一部がページ内に収まらない可能性があります。 画像を挿入する場合は、次のオプションを指定します。 • [画像のパス]: 画像のフル パスを入力します。 • [ユニット]: 画像の座標とサイズの単位を指定します。 • [**X**]: ページの左端からの水平オフセットを設定します。 • [**Y**]: ページの上端からの垂直オフセットを設定します。 • [幅]: 画像の幅を指定します。 • [高さ]: 画像の高さを指定します。 • [縦横比を維持]: 指定した座標内に指定のサイズの画像を挿入するときに、元のアスペクト比を維持します。

アクション	説明
画像を挿入 (変数)	変数からドキュメント内の選択したページに画像を挿入します。画像を挿入する場合は、[画像を挿入] アクションと同じオプションを指定しますが、画像パスを指定する代わりに、画像を含んだバイナリ変数の名前を指定します。
名前を付けて保存	ドキュメントのコピーを保存するステップを挿入します。PDF ファイルを保存するフル パスを指定します。
変数に保存	ドキュメントのコピーをバイナリ変数に保存します。
閉じる	閉じる
SignDoc アクション	
SignDoc で署名	SignDoc セッションを作成し、PDF ドキュメントをすぐに送信します。
SignDoc で署名 (テンプレート)	SignDoc テンプレートに基づいて SignDoc セッションを作成し、すぐに PDF ドキュメントを送信します。
SignDoc 署名フィールドを挿入	PDF ドキュメントの現在のページに署名フィールドを挿入します。フィールドはページには表示されませんが、アプリケーション ツリーには SignDoc フィールドとして表示されます。
SignDoc プロパティを取得	SignDoc セッションのプロパティを照会します。
SignDoc リクエストを完了	SignDoc セッションを閉じて、SignDoc パッケージの処理方法を指定します。

SignDoc を使用してドキュメントに証明する方法については、ドキュメントへの署名を参照してください。

コンポーネント アクション

アクション	説明
テキストを抽出	PDF ドキュメントの選択したコンポーネントからテキストを変数に抽出します。 テキストを抽出する場合は、次のオプションを指定します。 • 幅 (オプション): テキストを抽出する領域の幅を設定します。 • 高さ (オプション): テキストを抽出する領域の高さを設定します。 • 包括: 境界ボックスに、指定した領域を超えた文字が含まれるようにする場合に選択します。 デフォルトでは、このオプションは選択されていません。つまり、境界ボックスには、指定した領域内に完全に収まる文字のみが含まれます。 すべてのユニットはデバイス ツリーの座標内にあります。
SignDoc アクション	
SignDoc 署名フィールドを挿入	コンポーネント ファインダーに基づいて署名フィールドを挿入します。フィールドはページには表示されませんが、アプリケーション ツリーには SignDoc フィールドとして表示されます。

アクション	説明
フィールドを更新する	サポート対象であると SignDoc で識別されたPDF ドキュメント内のフォーム フィールドの属性を更新します。サポートされているフィールドは、アプリケーション ツリーの SignDoc ノードの下にリストされます。
SignDoc 署名者の割り当て	PDF ファイルに既存の署名フィールドに署名者を割り当てます。これらのフィールドはページには表示されませんが、アプリケーション ツリーには SignDoc フィールドとして表示されます。

キープレス

このアクションでは、指定したキーを押します。次を右クリックすると使用できるアプリケーション レベルのステップです。

- アプリケーションのタブ。
- アプリケーションまたは Web サイトのテキスト フィールド。
- ロボットのプログラム ポイント。

仮想入力ドライバーを使用する

自動化されたデバイスで仮想入力ドライバーを有効にすると (「仮想入力ドライバーをアクティブにする」を参照)、Windows のデバイス オートメーションの [キー プレス] ステップでは、テキストの入力にこのドライバーが自動的に使用されます。キーはハードウェア キーボードを介して入力されるため、Ctrl + Alt + Del などのシステムのみの組み合わせが動作します。計算されたキーを使用する場合、「u」(キー アップ イベント用) 以外のフラグはサポートされません。ドライバーは、6 つ以上のキーを同時に押すことをサポートしていません。『Kofax RPA Desktop Automation サービス ガイド』を参照してください。

プロパティ

名前
ステップの名前。

ファインダー
[デバイス]: オートメーション デバイスの名前を選択します。

[アプリケーション]: アクションが実行されるアプリケーションの名前を指定します。

キー
[標準キー] または [計算されたキー] を選択します。

- [標準キー]: 文字、数字、句読点、方向キー、ファンクション キーなど、標準キーボードのキーから選択します。
- **[計算されたキー]** (Desktop Automation サービスのみ): このオプションは、キーボードのキーのオプションが不十分な場合に選択します。[キー コード] フィールドに、仮想キー コードまたはスペースで区切られた入力仕様のリストを指定します。この機能は、Windows オペレーティング システムでのみサポートされます。

  仮想キー コードはシンボリック定数名で、たとえば「左マウス ボタン」の場合は VK_LBUTTON になります。仮想キー コードのリストについては、Microsoft のドキュメントを参照してください。

入力仕様は、1 つ以上の keydown イベントまたは keyup イベントのシーケンスです。入力仕様を追加する場合は、以下のプレフィックスを使用して仮想キー コードまたはスキャン コードを指定します。

- **v**: 仮想キー コード (v0xXX など)
- **s**: スキャン コード (s0xXX など)

デフォルトでは、入力指定は keydown の仮想キー イベントです。このデフォルトを上書きするには、入力仕様に **f** フラグを追加し、カンマで区切ります。次のフラグがサポートされています：**u** (keyup)、**s** (スキャン コード)、**e** (拡張キー)、**U** (Unicode)。

例

- `v0x30 v0x30,fu` 計算されたキーは 0 キーを押してから離します。`v0x30` 入力仕様は keydown イベントで、`v0x30,fu` は keyup イベントです。
- `v0x5b v0x52 v0x52,fu v0x5b,fu` 計算されたキーは、Run コマンド (Win + R) 用です。左の Win キー、次に R キーを押してから、両方のキーを離します。`v0x5b` と `v0x52` は keydown イベントで、`v0x52,fu` と `v0x5b,fu` は keyup イベントです。
- `s0x04c1,fU s0x04c1,fUu` 計算されたキーは、キリル文字 Ӿ 用です。`0x04c1` コードは Ӿ の Unicode ですが、`s0x04c1,fU` はスキャン コード、keydownイベントで、 `s0x04c1,fUu` はスキャン コード、keyup イベントです。

修飾子
[キー] プロパティで [計算されたキー] を選択した場合、[修飾子] プロパティは無視されるため、設定する必要はありません。

キー修飾子を選択します。

- [固定キー修飾子]: Shift、Ctrl、Alt の 3 つのスタンダードなキー修飾子が含まれます。
- [計算されたキー修飾子]: このオプションを選択する場合、修飾子に対し仮想キー コードの記号定数名を指定します。
  表示されるテキスト ボックスに入力できるのは、Shift、Ctrl、Alt、Windows キーのキー コードのみです。適用可能なキー コードは次のとおりです。
  - 左右の Shift キーを表す VK_LSHIFT と VK_RSHIFT
  - 左右の Ctrl キーを表す VK_LCONTROL と VK_RCONTROL
  - 左右の Alt キーを表す VK_LMENU と VK_RMENU
  - 左右の Windows キーを表す VK_LWIN と VK_RWIN

カウント
アクションを実行する回数を指定します。形式は等号と数字になります (例：`=1`)。

マウス プレス

マウス プレス ステップは、マウス移動ステップと組み合わせてドラッグ & ドロップ操作を行えるよう設計されています。その他のマウス クリック操作については、クリック ステップを参照してください。

プロパティ

アプリケーション
マウス クリックのアプリケーション ファインダー。

アクション
3 つのマウス アクションが含まれます。それぞれのアクションは、現在のカーソル位置で開始されます。これが予期した場所であることを確認するか、場所を変更するには、マウス移動 ステップを使用します。

- [クリック]: アプリケーション インターフェイスでクリックします。
- [プレス]: マウス ボタンを解除せずに押します。
- [リリース]: マウス ボタンを解除します。

ボタン
ポインティング デバイスの [標準ボタン] または [計算されたボタン] を選択します。

- [標準ボタン]: 左、中央、右。
- [計算されたボタン]: このオプションを選択して、ロボットの実行中にマウス ボタンのクリックを判定します。[エクスプレッション] フィールドに、仮想キー コードまたはスペースで区切られた入力仕様のリストを指定します。エクスプレッションの結果は、マウス ボタンの 1 つを表す 0 から 2 までの値である必要があります。この機能は、Windows オペレーティング システムでのみサポートされます。仮想キー コードのリストについては、Microsoft のドキュメントを参照してください。

例
左マウス ボタンに「0」、右マウス ボタンに「1」、中央マウス ボタンに「2」を使用します。

修飾子
キー修飾子を選択します。

- [固定キー修飾子]: Shift、Ctrl、Alt の 3 つのスタンダードなキー修飾子が含まれます。
- [計算されたキー修飾子]: このオプションを選択する場合は、修飾子に仮想キー コードの記号定数名を指定します。
  表示されるテキスト ボックスには、Shift、Ctrl、Alt のキー コードのみが入力可能です。たとえば、VK_LSHIFT キー コードは左 Shift キーを、VK_RCONTROL は右 Ctrl キーを、VK_MENU は Alt キーを表します。全キー コードのリストについては、Microsoft のドキュメントを参照してください。

RDP ログイン

この手順を使用して、RDP 接続を介してデバイスに接続します。詳細については、RDP 接続の使用 を参照してください。

RDP ログイン ステップは、接続が確立されるまで待機してからロボットの実行を続行します。リモート接続に失敗すると、エラー メッセージが表示されます。

プロパティ

アクション
ステップで実行するアクションを選択します。

ホスト
接続するホスト名を指定します。

アカウント
RDP 接続のアカウント名を指定します。

ドメイン
RDP 接続のドメイン名を指定します。

パスワード
リモート デスクトップで認証するパスワードを指定します。

パスワードまたはホスト名に、URL で受け入れられない文字 (バックスラッシュなど) が含まれている場合は、エンコードする必要があります。

試行数
ステップが接続の確立を試行する回数を指定します。システムの負荷または起動の遅延が原因で接続が失敗したことに気付いた場合は、このプロパティを 1 より大きい値に設定します。

デスクトップ サイズ
デスクトップ ジオメトリ (WxH) を設定します。

色深度
接続の色深度 (16、32 など) を設定します。

RDP 接続には、明示的に指定された同じ解像度と色深度のパラメータを常に使用してください。Windows 10 は、32 ビット以下の色の深度をサポートしません。そのため RDP から送信される接続カラーの深度を変更する要求は、この Windows バージョンでは無視されます。

ログオン ダイアログの破棄タイムアウト
ログイン中に別の画面を閉じるまで待機する秒数を指定します接続しているシステムがユーザーのログイン時に別の画面を表示するように設定されている場合は、オプションでログイン中に別の画面を閉じるまで待機する秒数を設定します。この画面を閉じないと、アクションが失敗することがあります。

キープアウェイク キーを押す間隔
RDP セッションを維持するために送信するダミーのキーストロークの間隔を秒単位で指定します。デフォルトは 30 秒です。キー ストロークを無効にするには、ゼロ (0) を指定します。

追加のパラメータ
セッションの追加パラメータを指定します。以下のパラメータを利用できます。

パラメータ	説明
`/kbd:<言語>`	新しいセッションを設定するときに、指定したキーボードが選択されます。Windows キーボード レイアウトの ID またはキーボード レイアウトの完全な名前のいずれかを指定します。ロボットが既存のセッションに接続する場合、RDP サーバーはこのパラメータを無視します。 例: `/kbd:german` または `/kbd:0x407` を指定すると、ドイツ語のキーボード レイアウトが選択されます。

ファイルの読み込み

ファイルの読み取りステップは、リモート デスクトップ上のファイルからバイナリ変数にデータを抽出します。このステップは、ロボット ファイル システムのファイルからの読み込みに使用できます。

ローカル Desktop Automation モードでのローカル ファイル システムへのアクセスを含むファイル システム アクセスを有効にするには、RoboServer 設定アプリケーションの [セキュリティ] タブで [ファイル システムとコマンド ラインのアクセスを許可] オプションを選択します。

RoboServer がローカル ファイル システムにアクセスできないように設定されている場合、ロボットは実行できますが、ステップがローカル デバイス上で [直接アクセス] に設定した [ファイル アクセス] を使用すると エラーが発生します。ただし、このステップは、[ファイル アクセス] が [**RFS** 経由] に設定されているローカル デバイス上で機能します。リモート デバイス上では、[直接アクセス] と [RFS 経由] が使用できます。

プロパティ

デバイス
使用するリファレンス名を選択します。このリファレンス名は、「ロボットの呼び出し」ステップの [必要なデバイス] プロパティで指定します。

ファイル アクセス
ファイルのアクセス方法を指定します。
指定のローカル デバイスまたはリモート デバイス上のファイルから読み込むには、[直接アクセス] を選択します。
ロボット ファイル システム上のファイルから読み込むには、[**RFS** 経由] を選択します。

ファイル名
データを抽出するファイルへのパスを指定します。

変数
抽出したデータを保存するバイナリ変数を指定します。

リモート デバイス アクション

デバイス アクション ステップを使用して、リモート コンピュータ上で実行中の Desktop Automation サービスでいくつかのアクションを実行できます。

プロパティ

名前
ステップの名前。

デバイス
サービスを管理するリモート デバイスを選択します。

アクション

- [**DA** サービスをサスペンド]: デバイスをサスペンドします。サービス操作を復元するには、ユーザーまたは管理者はデバイスで Desktop Automation サービスを手動で起動する必要があります。
  サスペンド状態ではロボットは DA サービスを利用できませんが、状態情報は ping メカニズムを介して Management Console に送信され、デバイスは **[管理]** > **[デバイス]** セクションに表示されます。このコマンドは、何らかの理由でサービスまたはサービスを実行しているコンピュータの設定を変更する必要が生じた場合に役立ちます。

- [**DA** サービスを停止]: サービスを停止します。これによりリモート デバイスは使用できなくなります。Desktop Automation サービスが実行されているコンピュータは、Management Console のリストから削除されます。
- [**DA** サービスを再起動]: サービスを停止してから、起動します。ロボットまたは Design Studio ではデバイスとの接続が失われるため、復元するにはリロードする必要があります。
- [スクリーン ロック]: リモート デバイスのスクリーンをロックします。設定とパスワードが必要です。『Kofax RPA Desktop Automation サービス ガイド』を参照してください。
- マシンを再起動: Desktop Automation サービスを実行中のコンピュータを再起動します。
- マシンをシャットダウン: Desktop Automation サービスを実行中のコンピュータをシャットダウンします。

リターン

これは、変数値を出力するロボット実行における最終ステップです。オートメーション ワークフローの [リターン タイプ] セクションのタイプと同じ順序で変数をリターン ステップで指定します。変数値を出力しない場合は、ステップを空のままにします。

複数のリターン ステップを使用できますが、最初のリターン ステップが実行されると、ロボット実行は停止します。これは、条件付きステップで条件と出力変数値をチェックし、これらのステップが条件に適合しているかどうかを確認する場合に有用です。条件が一致しない場合、ロボットは実行を続けます。

依存ロボット (ベーシック エンジン ロボットから呼び出される) をスタンドアロン ロボットに変換する場合は、戻るステップを 出力値 ステップに置き換えます。

プロパティ

名前
ステップの名前が含まれます。

値
出力する変数と値を指定します。変数の順序は、[リターン タイプ] セクションのタイプのリストと一致する必要があります。すべての変数が同じタイプの場合、順序は重要ではありません。

スクロール

このステップでは、プログラム ウィンドウをスクロールします。これはアプリケーション レベルのステップで、[アプリケーション] タブを右クリックすると利用できます。このステップを挿入する前に、ウィンドウで適切な エレメントを選択する必要があります。

さらに、自動化されたコンピュータでハードウェアのキーボードとマウスをシミュレートすることもできます。詳細については、『Kofax RPA インストール ガイド』の「Desktop Automation サービスのインストール」を参照してください。

プロパティ

名前
ステップの名前を指定します。

量
スクロールする量を指定します。このフィールドの値は、マウス スクロール ホイールのノッチの数と等しくなります。最初に、1 ノッチはテキスト エディターの 3 行のテキストに等しくなります。このパラメータは、オートメーション デバイスのマウス プロパティ ウィンドウで変更できます。正数も負数も使用できます。たとえば、[ダウン] を選択し、負数を使用すると、エレメントが上にスクロールします。
このステップを組み込みブラウザで使用する場合、[量] オプションはスクロールするピクセル数を意味します。

スクロールするエレメントを正しく選択してください。エレメントを選択できない場合、まずエレメントをクリックしてから、スクロールを使用します。

現在のインを保存

このステップを使用できるのは、値を抽出、クリップボードから抽出、ツリーを XML として抽出、および値を変換のコンテキストのみです。変換された値を指定したタイプの変数に保存します。値のタイプは変数のタイプと一致する必要があり、次のいずれかになります：整数、ブール値、数値、またはテキスト。

ターミナル

このステップを使用して、ターミナルに接続してターミナルを自動化できます。詳細については、ターミナル エミュレータの自動化 および 基本 ターミナル チュートリアル を参照してください。

プロパティ

エミュレータ
ターミナルのエミュレーターを選択します。Kofax RPA は、3270、5250、6530 およびストリーム ベース (VT100 および ANSI) ターミナルとの接続および通信に対応しています。

アクション
ステップで実行するアクションを選択します。[接続] または [接続 **(SSH)**] のいずれかを設定できます。

ホスト
必要に応じて、ターミナル名または IP アドレスとポート番号を指定します。例：`TerminalServer:25`

クレデンシャル
[アクション] リストで [接続 **(SSH)**] を選択した場合、このプロパティのターミナルに接続するためのユーザー名とパスワードを指定できます。

オプション
色オプションのサポート、トレース ファイルの作成、バッファされた行数の設定など、選択したターミナルの接続オプションを指定できます。詳細については、ターミナル エミュレータの自動化 を参照してください。

スロー

このステップは、例外をスローしてエラーを表示し、ロボット ワークフローの別の場所でそのエラーを処理します。

その他のワークフロー ステップでエラーが発生した場合も、例外がスローされます。ワークフローのロジックで見つかったエラーと、ステップで見つかったエラーは、同じ方法で処理されます。その他のワークフロー ステップでスローされた例外のリストについては、トライ-キャッチを参照してください。[スロー] ステップで、[使用量の検索] オプションを使用して [例外] を検索します。

スローされた例外は、[キャッチ] (Catch) 分岐に指定した例外を含む最も近い トライ-キャッチ によってキャッチされ、処理されます。そのような トライ-キャッチ ステップがない場合、例外はワークフロー内で "not handled" に設定されます。その場合、ロボット ワークフローおよび「ロボットの呼び出し」ステップ の実行は停止し、「ロボットの呼び出し」ステップの [エラー処理] タブで指定したとおりにエラーが処理されます。

スロー ステップは通常、タイムアウト ガードとともに使用します。デバイスとの意図した相互作用 (たとえば、「Location Found ガード」によって設定) が可能でない場合にタイムアウト エラーが発生します。タイムアウトが発生すると、その他のことを行うことができ、その結果回復することがあります。回復できない場合、スロー ステップを使用して、失敗を体系的に伝えます。これにより、トライ-キャッチを追加し、エラーを適切に処理 (デバイスとの相互作用を取り消すなど) することができます。

ワークフローの異なる場所で (つまり異なる スロー ステップで) 同じ例外名を同様のエラーに使用すると、同じ トライ-キャッチ ステップですべてのエラーを処理することが可能です。そのため、例外名には、すべての詳細ではなく、エラー状況の分類を提供する必要があります。

このステップでは例外をスローし、ロボット実行が停止します。このステップは、ロボットの設計およびデバッグ時に便利です。たとえば、60 秒のタイムアウト ガードによってアクションなしで 60 秒待機するタイミングを確認する場合は、スロー ステップを「60 秒のタイムアウトが経過しました。」などのテキストとともにタイムアウト ガードに挿入します。実行中に表示されるメッセージは、ガードが 60 秒待機し、何も起こらなかったことを意味します。

スロー ステップは トライ-キャッチ ステップのファイナル ブロックに挿入することはできません。

プロパティ

名前
ステップの名前が含まれます。

例外
例外の名前。この名前は、変数名ルールに従っている必要があります。命名規則を参照してください。

トリガー チョイス

このステップはアテンデッド オートメーション機能の一部です。このステップでは、トリガーを選択し、トリガーによって起動されるアクション ステップを挿入します。トリガー イベントがトリガー チョイス ステップ内で阻止された際に実行されるアクション ステップを 1 つ以上挿入します。

トリガー イベントが検出されると、ロボットは自動化されたデバイスを引き継いで、ユーザーがマウスやキーボードを使用できないようにすることがあります。ユーザーに、ロボットで実行されたアクションを知らせるには、通知を使用します。

トリガーのヒント

- トリガー チョイス ステップを挿入するには、[レコーダー ビュー] タブを右クリックして、メニューで [トリガー] を選択します。

- トリガー チョイス ステップの [コンポーネント クリック] イベントを挿入するには、レコーダー ビューで要素を右クリックして、メニューで [トリガー] を選択します。
- ロボットで使用できるトリガーは 1 つのみです。
- スニペットでトリガーを使用することはできません。
- トリガー内にトリガーを作成することはできません。

以下のトリガー イベントが利用できます。

トリガー

アプリケーションを開く
指定されたアプリケーションが開くと、ロボットにより、選択されたアクションが実行されます。アクションをトリガーするトリガーの名前とアプリケーションを指定します。

アプリケーションを閉じる
指定されたアプリケーション閉じると、ロボットにより、選択されたアクションが実行されます。アクションをトリガーするトリガーの名前とアプリケーションを指定します。

コンポーネントをクリックする
指定されたコンポーネントをクリックすると、ロボットにより、選択されたアクションが実行されます。トリガーの名前、アクションをトリガーするアプリケーション、およびアプリケーション内のコンポーネントを指定します。また、コンポーネントをクリックするマウス ボタンを選択します。

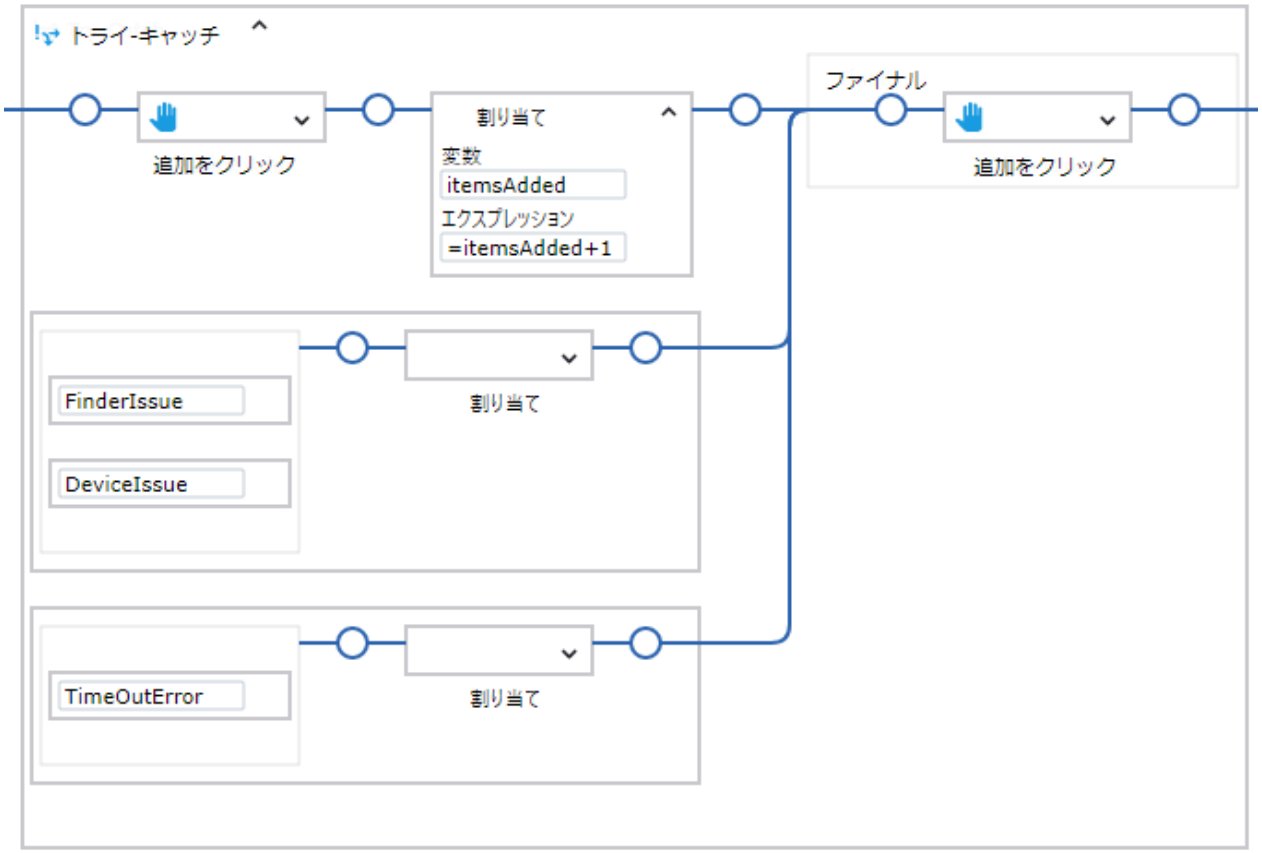
ホット キーを押す
指定されたキーまたはキーの組み合わせを押すと、ロボットにより、選択されたアクションが実行されます。トリガーの [名前] を指定し、リストから [キー] を選択します。また、Shift、Ctrl、Alt の 3 つのスタンダードなキー修飾子から選択します。

トライ-キャッチ

このステップでは、アクションを実行し、そのアクションによって発生することがある 1 つ以上の例外をキャッチします。このステップは、3 つの部分に分割された多くの分岐から構成されています。

- トライ分岐：実行するアクションを指定する最上部分。
- キャッチ分岐：トライ分岐のアクションの実行時にスローされる可能性がある 1 つ以上の例外と、それが発生した場合に実行するアクションを指定します。複数のキャッチ分岐を設定することができ、それぞれのキャッチ分岐は同じ方法で処理される任意の数の例外をリストできます。
- ファイナル分岐：実行するアクションを指定します。この分岐は、トライとキャッチの実行結果に関係なく、常に最後に実行されます。

例外は、スローによって明示的にスローされるか、その他のステップで実行中にエラーが発生するためスローされます。スローされた例外は、事前定義の例外と呼ばれます。[トライ-キャッチ] ステップのキャッチ分岐で、[使用量の検索] コンテキスト メニュー オプションを使用して [例外] を検索します。



プロパティ

名前
ステップの名前が含まれます。

トライ
実行するアクションを指定します。アクションの結果として例外が予想される場合、キャッチ ブロックで例外を指定します。

例外
キャッチすることが予想される 1 つ以上の例外を指定します。

キャッチ分岐はそれぞれ例外のリスト、およびその右側にある、トライ分岐の実行でこれらのいずれかの例外がスローされた場合に実行するアクションで構成されます。

各例外には、スロー ステップで使用される例外名、または事前定義された例外名に対応する名前が付けられます。

キャッチ分岐で例外が追加または編集されると、エディターでは トライ分岐内でスローされる可能性があり、かつキャッチ分岐にまだリストされていない例外が提案されます。

ファイナル
トライ-キャッチ ステップを終了する直前に実行するアクションを指定します。

実行

トライ-キャッチ ステップの実行は、その他のステップよりも少し複雑です。最も一般的な実行ケースは最もシンプルで、最初に説明します。最も複雑なケースはファイナル分岐にステップが含まれている (空ではない) 場合です。

すべてのケースで、トライ-キャッチ ステップの実行は、トライ分岐を実行することから始まります。これは正常に終了するか、いずれかのステップによってスローされる例外によって終了できます。

最も一般的なケース：ファイナル分岐が空

トライ分岐が正常に終了
実行は、トライ-キャッチ ステップ全体の後のステップで続行されます。つまり、キャッチ分岐はこの場合は実行されません。

トライ分岐がスローされた例外で終了
例外をスローするステップから、実行は例外をリストするキャッチ分岐の開始を直接続行します。

より複雑なケース：ファイナル分岐が空

トライ分岐がスローされた例外で終了しても、キャッチ分岐はその例外をリストしない
このケースは、トライ-キャッチ ステップ自体が例外をスローしたように扱われ、その他のステップが例外をスローした場合と同じ方法で処理されます。ここにリストされているすべてのケースが適用されます。

> この戦略 (「トライ-キャッチ ステップ自体が例外をスローしたように扱われる」) は、その他多くのケースで使用されます。

すべての トライ-キャッチ ステップに空のファイナル分岐がある場合、ワークフロー ロジックで周囲の トライ-キャッチ ステップの一致するキャッチ分岐が検索され、どの トライ分岐にこの トライ-キャッチ ステップが含まれているかが検索されます。そのようなキャッチ分岐が周囲の トライ-キャッチ ステップに見つからない場合、例外はワークフロー内で "not handled" に設定されます。こうした場合、ロボット ワークフローおよび含まれている「ロボットの呼び出し」ステップ の実行は停止し、「ロボットの呼び出し」ステップの [エラー処理] タブで指定したとおりにエラーが処理されます。

トライ-キャッチ ステップにファイナル分岐も含まれている場合、実行は同様ですが、1 度に 1 つの "スロー" が実行されます。

トライ分岐がスローされた例外で終了し、当該のキャッチ分岐もスローされた例外で終了する
キャッチ分岐でスローされた例外は同じ トライ-キャッチ ステップのキャッチ分岐では処理されません。代わりに、これは トライ-キャッチ ステップ自体がその例外をスローしたように扱われます。詳細については、前のケースの説明を参照してください。

ネストされた トライ-キャッチ ステップに関する注意
トライ-キャッチ ステップによって処理される例外は、周囲の トライ-キャッチ ステップによって処理されません。例外を処理できるキャッチ分岐が見つかると、例外は完全に処理されたものとみなされ、"forgotten" になります。キャッチ分岐の実行が開始し、通常の方法で続行します。そのため、各例外は一度だけ処理されます。

最も複雑なケース：ファイナル分岐が空ではない
この場合、実行の状態に関係なく、ファイナル分岐のステップは実行が トライ-キャッチ ステップで終了する直前に実行されます。次のケースで、これがどのように動作するかを上記のケースごとに詳述します。

トライ分岐が正常に終了
例外はファイナル分岐のステップで続行します。その後のことは、ファイナル分岐の実行がどのように終了するかによって異なります。

- ファイナル分岐の実行が正常に終了すると、実行は トライ-キャッチ ステップ全体の後のステップで続行されます。
- 例外がファイナル分岐の実行中にスローされる場合、これは トライ-キャッチ ステップ自体がその例外をスローしたように扱われます。

トライ分岐はスローされた例外で終了し、キャッチ分岐は正常に終了する
キャッチ分岐の実行後、ロジックは前のケースのとおりです。

トライ分岐がスローされた例外で終了しても、キャッチ分岐はその例外をリストしない
この場合、例外は "remembered" となり、実行はファイナル分岐のステップで続行されます。その後のことは、ファイナル分岐の実行がどのように終了するかによって異なります。

- ファイナル分岐の実行が正常に終了すると、実行は トライ-キャッチ ステップ自体が "remembered" の例外を再度スローするかのように続行されます。
- 例外がファイナル分岐の実行時にスローされると、これは同じ トライ-キャッチ ステップのキャッチ分岐によって処理されません。代わりに、これは トライ-キャッチ ステップ自体がその例外 (つまり、ファイナル分岐によってスローされた例外) をスローしたように扱われます。"remembered" の例外はこの時点では事実上 "forgotten" です。

トライ分岐はスローされた例外で終了し、当該のキャッチ分岐もスローされた例外で終了する
これは、"remembered" の例外が トライ分岐ではなく、キャッチ分岐によってスローされたものであることを除き、前のケースと同様に処理されます。上記のように、トライ分岐によってスローされた例外がキャッチ分岐の実行開始時に完全に処理され、"forgotten" になります。

事前定義の例外

ステップで実行中にエラーが発生した場合、次のいずれかの例外がスローされます。これらの例外は必要に応じて スロー ステップによって明示的にスローすることもできます。

ステップ エラーのためにスローされると、事前定義の例外にはその問題を説明するメッセージが含まれます。このメッセージは、例外がロボット ワークフローの トライ-キャッチ ステップで処理されず、「ロボットを呼び出す」ステップ の実行が終了された場合に利用できるようになります。

すべての「内部」例外は事前定義されているため、名前を変更することはできません。「ユーザー定義」例外の名前は、タイムアウトが参照しているステップの種類によっては、ユーザーが変更することも可能です。たとえば、[InputNameTimeOut] または [LoginTimeOut] に変更できます。

- `TimeOutError`: 実行がタイムアウトした場合にスローされます。
- `FinderIssue`: ファインダーで要素が見つからなかった場合にスローされます。
- `DeviceIssue`: ステップの実行を妨げるデバイスまたはドライバーの問題の場合にスローされます。
- `IncorrectValueIssue`: `"one".substring(-1)` の `-1` など、エクスプレッションの値が使用場所で適切でない場合にスローされます。
- `ExtractIssue`: 抽出ステップで抽出に失敗した場合にスローされます。
- `DivisionByZeroIssue`: エクスプレッションの評価中にゼロ除算 (またはゼロ剰余) が発生した場合にスローされます。
- `OverFlowIssue`: エクスプレッションの評価でオーバーフローが発生した場合にスローされます。

- `ConversionIssue`: エクスプレッションの評価中に `"one".integer()`. など、タイプ間の変換が失敗した場合にスローされます。
- `DisappearedIssue`: アクションを実行する対象のコンポーネントまたはアプリケーションが見つからなかった場合にスローされます。

エクスプレッションがステップの一部である場合は、ステップの実行で次の例外がスローされることがあります。

- `IncorrectValueIssue`
- `DivisionByZeroIssue`
- `OverflowIssue`
- `ConversionIssue`

以下の表に、ステップ、ファインダー、およびその他のワークフロー エレメントでスローされる可能性のある例外をリストします。エクスプレッションの問題は、エクスプレッションを持つステップによってスローされる問題です。

ワークフロー エレメント	例外
ステップ	
クリック	DeviceIssue、FinderIssue、エクスプレッションの問題
テキストを入力	DeviceIssue、FinderIssue、エクスプレッションの問題
キープレス	DeviceIssue、FinderIssue、エクスプレッションの問題
スクロール	DeviceIssue、FinderIssue、エクスプレッションの問題
マウス移動	DeviceIssue、FinderIssue、エクスプレッションの問題
クリップボードへ割り当て	DeviceIssue、エクスプレッションの問題
割り当て	エクスプレッションの問題
値を抽出	DeviceIssue、FinderIssue、エクスプレッションの問題、ExtractIssue
クリップボードから抽出	DeviceIssue
画像抽出	DeviceIssue、FinderIssue、エクスプレッションの問題、ExtractIssue
ツリーを XML として抽出	DeviceIssue、FinderIssue、エクスプレッションの問題、ExtractIssue
画像からテキスト抽出	DeviceIssue、FinderIssue、エクスプレッションの問題
ループ	なし
条件	エクスプレッションの問題
グループ化	なし
With	DeviceIssue、FinderIssue、エクスプレッションの問題
ガード チョイス	上記の表にリストされているガードによって異なります
トライ-キャッチ	なし
ブレイク	なし
スロー	なし

ワークフロー エレメント	例外
Return	エクスプレッションの問題
開く	DeviceIssue、エクスプレッションの問題
デバイスに接続	DeviceIssue、エクスプレッションの問題
リモート デバイス アクション/スクリーンのロック コマンド	DeviceIssue、エクスプレッションの問題
リモート デバイス アクション/その他	DeviceIssue
エクスプレッション	
任意のエクスプレッション	IncorrectValueIssue、DivisionByZeroIssue、OverFlowIssue、Conver
ガード	
時間経過	エクスプレッションの問題、IncorrectValueIssue
該当するアプリケーション	エクスプレッションの問題、DeviceIssue、FinderIssue
該当しないアプリケーション	エクスプレッションの問題、DeviceIssue、FinderIssue
該当するロケーション	エクスプレッションの問題、DeviceIssue、FinderIssue
該当しないロケーション	エクスプレッションの問題、DeviceIssue、FinderIssue
取り除かれたロケーション	エクスプレッションの問題、DeviceIssue、FinderIssue
ツリー変更停止	エクスプレッションの問 題、IncorrectValueIssue、DeviceIssue、FinderIssue
ファインダー	
デバイス ファインダー	DeviceIssue
アプリケーション ファインダー	DeviceIssue、FinderIssue、エクスプレッションの問題
コンポーネント ファインダー	DeviceIssue、FinderIssue、エクスプレッションの問題

ガード チョイス ステップは、ステップで使用されるガードに応じて、以下の例外をスローします。

ガード	例外
時間経過	エクスプレッションの問題
その他	DeviceIssue、FinderIssue、エクスプレッションの問題

Windows

このステップは、Windows デスクトップで作業し、Windows アプリケーションを実行するのに役立ちます。

プロパティ

次のプロパティを指定して、アプリケーションを実行します。

デバイス
アプリケーションを開くデバイスを選択します。

アクション
実行するアクションを選択します。

実行可能
- 実行するアプリケーションのパスを指定します。ローカル ドライブのパスまたはネットワーク パスを使用できます。
- 組み込み Windows アプリケーションの場合、`calc` などのプロセス名を指定できます。
  次の例は、電卓アプリケーションを起動するためのいくつかの方法を示しています。
  - `calc`
  - `calc.exe`
  - `C:\\Windows\\System32\\calc.exe`
  - `C:\Windows\System32\calc`
  - `C:/Windows/System32/calc`
  - `C://Windows//System32//calc.exe`

  ネットワーク パスを次のように指定します。

  `\\MyServer\shared\reportform.exe`

作業ディレクトリ (オプション)
アプリケーションの作業ディレクトリのパスを指定します。たとえば、同じ名前のアプリケーションが別のフォルダにある場合は、このパスを使用します。

引数 (オプション)
必要に応じて、アプリケーションの引数を指定します。たとえば、Web ブラウザ アプリケーションに「kofax」と指定した場合、ブラウザでは https://www.kofax.com Web サイトが開きます。

インタラクティブ (オプション)
このオプションを選択すると、ロボットによって開始されたプログラムがバックグラウンド アプリケーションとして開始されることはなくなり、ロボットがこのプログラムを操作できるようになります。このオプションは、Windows が Kofax RPADesktop Automation サービスをバックグラウンド アプリケーションとして処理する場合に役立ちます。したがって、一部のアプリケーションがバックグラウンドで開始されると、ロボットはこれらを操作できなくなります。

最大化を開始 (オプション)
最大化されたウィンドウでプログラムを開始します。

Kofax RPA インストールをバージョン 11.2 にアップグレードした場合は、この手順によって、既存のすべてのロボットに[インタラクティブ] (デフォルトでは未選択) および [最大化を開始] (デフォルトで選択) のオプションが自動的に追加されます。

ステップの例

次の例は、メモ帳アプリケーションを使用して `C:\Temp` にある `text.txt` ファイルを開く場合に役立つ Windows ステップ プロパティを示しています。
- アクション: 実行
- 実行可能: `notepad.exe`
- 作業ディレクトリ: `C:\Temp`
- 引数: `text.txt`

アプリケーション アクション

Internet Explorer モードの Microsoft Edge を使用している場合は、[レコーダー ビュー] の [Internet Explorer モードの Microsoft Edge] タブを右クリックすると表示される [アプリケーション アクション] メニューで、次のアクションを使用できます。これらのアクションは、Desktop Automation サービスが実行されているコンピュータで Internet Explorer モードの Microsoft Edge が開かれている場合にのみ表示され、アプリケーション全体に適用されます。

Desktop Automation サービスを実行中のコンピュータで開いている Internet Explorer モードの Microsoft Edge の表示スケーリングは自動的に 100% に設定され、これは変更できません。

アクション	説明
フォーカス	選択したアプリケーションをフォーカスするステップを追加します。

JavaScript の実行	現在の Internet Explorer モードの Microsoft Edge ページで JavaScript を実行するステップを追加します。
	• **JavaScript**実行するコードを入力します。
	• **JS** 実行結果: 結果を保存するバイナリ形式の変数を指定します。
	返された実行結果がテキスト タイプでない場合は、自動的にテキストに変換されます。
	  • 「未定義」、「オブジェクト」、「null」は空の文字列に変換されます
	  • 「ブール値」は、「true」または「false」に変換されます
	  • 「数値」は、数値の文字列表現に変換されます
	• [結果を待機] を選択すると、コードが実行され、結果が [**JS** 実行結果] 変数に書き込まれるまでロボットは待機します。
	タイムアウトは 60 秒です。この期間内にアクションが完了しない場合は、タイムアウト エラーが表示されます。また、JavaScript コードにエラーが含まれる場合にもエラーが表示されます。
	このオプションを選択しない場合、ロボットはコードの実行が終了するまで待機せず、すぐに次のステップに進みます。結果は収集されず、変数に書き込まれません。さらに、このコードで作成された定義またはオブジェクトの対象範囲はこのステップに限定され、後続の JavaScript の実行ステップでは使用できません。

コンポーネント アクション

Internet Explorer モードの Microsoft Edge を使用する場合は、[コンポーネント アクション] メニューで次のアクションを使用できます。これらのアクションを使用できるのは、Desktop Automation サービスを実行しているコンピュータで Internet Explorer モードの Microsoft Edge が開いている場合のみです。

Desktop Automation サービスを実行中のコンピュータで開いている Internet Explorer モードの Microsoft Edge の表示スケーリングは自動的に 100% に設定され、これは変更できません。

アクション	説明
キーボード フォーカスを設定	キーボード入力を使用するには、選択したコンポーネントにキーボード フォーカスを設定します。
オプション選択	このオプションは、Web ページが Internet Explorer モードの Microsoft Edge に読み込まれている場合にのみ使用できます。その Web サイトで互換表示が無効になっていることを確認します。 Web サイトがイントラネット Web サイトの場合は、[このページを互換表示で開く] が選択されていないことを確認してください。 メニューから単一のオプションまたは複数のオプションを選択できるようにするステップを追加します。 • アイテムのリスト、アイテムのグループ、またはドロップダウン リストから単一のオプションを選択できます。 この操作を行うには、レコーダー ビューのメニューをクリックし、ツリー ビューの [選択] タグで必要な [オプション] タグを選択します。 • Ctrl キーを押しながらクリックする操作と同様に、アイテムのリストまたはアイテムのグループから複数のオプションを選択できます。 リストまたはグループから単一のオプションをすでに選択している場合は、このリストまたはグループの別のアイテムに「オプション選択」ステップを追加し、ステップのプロパティで **[複数選択]** をオンにします。ステップを実行すると、両方のオプションが選択された状態で表示されます。さらにオプションを選択するには、選択する追加オプションごとに同じ手順を実行します。 複数の選択を解除する必要がある場合は、同じ設定を使用します。

ステップの更新

デバイス情報の更新などによって、Windows ステップで使用されるバックグラウンド情報が変更された場合は、変更が行われたことを示す警告が表示されます。コンテキスト メニューで [ステップの更新] をクリックすると、新しい情報を使用して Windows ステップを更新できます。

ファイル出力

ファイルの書き込みステップは、バイナリ変数のデータをリモート デスクトップのファイルに書き込みます。この手順を使用して、ロボット ファイル システム上のファイルに書き込むことができます (例については、『Kofax RPA 管理者ガイド』を参照してください)。

ローカル Desktop Automation モードでのローカル ファイル システムへのアクセスを含むファイル システム アクセスを有効にするには、RoboServer 設定アプリケーションの [セキュリティ] タブで [ファイル システムとコマンド ラインのアクセスを許可] オプションを選択します。

RoboServer がローカル ファイル システムにアクセスできないように設定されている場合、ロボットは実行できますが、ステップがローカル デバイス上で [直接アクセス] に設定した [ファイル アクセス] を使用するとエラーが発生します。ただし、このステップは、[ファイル アクセス] が [**RFS** 経由] に設定されているローカル デバイス上で機能します。リモート デバイス上では、[直接アクセス] と [RFS 経由] が使用できます。

ファイル出力ステップは、ファイル システムと出力のカテゴリに属します。

プロパティ

デバイス
使用するリファレンス名を選択します。このリファレンス名は、「ロボットの呼び出し」ステップの [必要なデバイス] プロパティで指定します。

コンテンツ
ファイルに書き込む値を指定します。このフィールドで変数名を指定できます。

たとえば、ロボット ファイル システム上のファイルに書き込みを行う場合、このフィールドには `="data".binary("utf-8")` を含めることができます。ここで `"data"` はそのファイルに書き込まれる文字列です。ファイル コンテンツはバイナリである必要があります。

ファイル アクセス
ファイルのアクセス方法を指定します。

指定のローカル デバイスまたはリモート デバイス上のファイルに書き込むには、[直接アクセス] を選択します。

ロボット ファイル システム上のファイルに書き込むには、[**RFS** 経由] を選択します。

ファイル名
データを書き込むファイルへのパスを指定します。

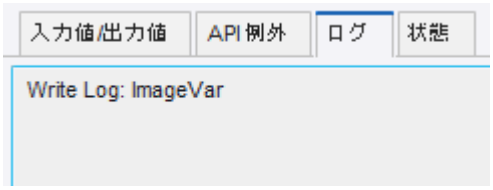
たとえば、ロボット ファイル システム上のファイルに書き込むには、ファイルのパスが設定されたファイル システム名で始まる必要があり、`myshare/myfile.bin` のような形式になります。

使用するファイル システム名は、Management Console 内の[ロボット ファイル システム] セクションでの指定に対応している必要があります。

ログ出力

ログの書き込みステップは、事前定義されているメッセージを従うステップの直後にログに書き込みます。

Design Studio で [ログ] タブを有効にするには、ロボットをデバッグ モードで実行します。



Management Console でメッセージを表示するには、**[ログ ビュー]** > **[ロボット実行]** タブに移動し、ログ出力ステップを含むロボットをダブルクリックします。

プロパティ

メッセージ
ログに書き込むテキスト、変数、または式のいずれかを指定します。

ターミナル エミュレータの自動化

Kofax RPA は、ターミナル エミュレータによるホストへの接続をサポートします。ホスト システム プラットフォームに基づいてターミナル エミュレータを構成します。

ターミナル エミュレータを自動化するには、プラットフォームに基づいて次の設定リストから選択します。

- Unix、Linux、および Windows (vt100 シリーズおよび ANSI) ターミナルの構成
- Mainframe (tn3270) ターミナルの構成
- iSeries (tn5250) ターミナルの構成
- NonStop (tn6530) ターミナルの構成

コマンド タイムアウトの設定
次のいずれかの方法で、ターミナルを自動化するためのコマンド タイムアウトを設定します。

- Design Studio でワークフローを実行する場合は、[Design Studio 設定] ウィンドウの [**Desktop Automation**] タブ。
- RoboServer 実行の場合は、[**RoboServer** 設定] ウィンドウの [セキュリティ] タブにある [オートメーション デバイス] セクション。

『Kofax RPA 管理者ガイド』の「ランタイム」 >「セキュリティ」を参照してください。

フォントの取得
Kofax RPA は、ターミナルのスクリーンをレンダリングするためのフォントを提供します。

```
C:\Program Files\Kofax RPA 11.5.0\nativelib\hub\windows-x32\{nnn}\fonts
```

ここでは、{nnn} は内部目的用の 3 桁の数字です。
使用可能なフォントにない文字を使用するホストとのターミナル接続が確立されると、文字は画面上で四角形としてレンダリングされます。これは、同じ Kofax RPA フォント ディレクトリにある `fontlist.txt` の TrueType フォントにパスを追加することで修正します。
例：`C:\TerminalFonts\MyTerminalFont.ttf`

SSH 認証オプションの設定

セキュア シェル (SSH) プロトコルを使用してアプリケーションを自動化する場合は、以下の 4 つの方法のいずれかでユーザーを認証します。

1. SSH クライアントがパスワードの入力を要求するようにします。「テキストを入力」ステップを使用し、キープレス アクションを実行してパスワードを入力します。
2. 暗号化されていない秘密鍵は、Design Studio および RoboServer のファイル システムに配置します。[ターミナル] ステップの [オプション] フィールドに `PublicKeyFile=[パスのキー]` という行を追加します。

   キーファイルは PuTTY 形式にすることが重要です。Linux では、**puttygen** を使ってオープンな ssh 秘密鍵を変換します。
3. 暗号化された秘密鍵は、Design Studio および RoboServer のファイル システムに配置します。[オプション] フィールドに `PublicKeyFile=[パスのキー]` を追加します。SSH クライアントがパスワードの入力を要求します。パスワードを入力するには、「テキストを入力」ステップを使用し、キープレス アクションを実行します。
4. ローカル ファイル システムで暗号化キーと、Windows では Pageant、Linux では SSH-agent を使用します。
   - Windows では、RoboServer を実行している同じユーザーとして Pageant を実行し、キーを追加します。
   - Linux では、`ssh-agent` と `ssh-add` を実行します。
   - `SSH_AUTH_SOCK` と `SSH_AGENT_PID` 環境変数を RoboServer および Design Studio 環境にコピーします。
   - 次の行を `RoboServer.conf` および `DesignStudio.conf` に追加します。
     - `set.SSH_AUTH_SOCK=<value as outputted from ssh-agent>`
     - `set.SSH_AGENT_PID=<value as outputted from ssh-agent>`

     または、RoboServer および Design Studio を起動する前に環境変数を設定します。

以前の認証方法を、最も安全性の低いものから最も高いものまで以下に示します。

1. ロボットを読むすべてのユーザーがパスワードを抽出してシステムにログオンできる。
2. 攻撃者は、ファイル システムからプライベート キーを取得する必要がある。(攻撃者が Management Console からロボットを入手した場合、アクセスするだけでは不十分です。)
3. 攻撃者は、ロボットとプライベート キー ファイルが必要になる。
4. 攻撃者は、アクセスのためにプライベート キーのパスワードを取得する必要がある。

Unix、Linux、および Windows (vt100 シリーズおよび ANSI) ターミナルの構成

ストリームベースのターミナルとして接続するには、ターミナル ステップでエミュレータに対して [接続 **(SSH)**] を選択します。SSH プロトコルは、暗号化と認証の両方を提供します。

基盤となる Kofax RPA SSH クライアントは PuTTY に基づいており、PuTTY がアクセスできる任意のシステムにアクセス可能になるように設定できます。PuTTY セッションと同じパラメータをすべて設定しますが、PuTTY クライアントでセッションを定義する必要はなく、[オプション] フィールドにオプションのパラメータを追加することでセッション設定を定義します。

接続オプション

ターミナルは、サーバー上のコード ページを自動的には検出しません。ASCII 以外の文字を送受信するには、文字のエンコードとデコードに使用するコード ページを指定する必要があります。コー

ド ページを指定しない場合、ロボットはサーバーのロケールが UTF-8 に設定されているとみなします。

コード ページを指定するには、ロボット ステップの [オプション] フィールドで `LineCodePage` パラメータを使用します。

Kofax RPA ターミナルは、文字エンコーディングに "libicu" を使用して構築されています。コード ページとエイリアスを調べるには、http://demo.icu-project.org/icu-bin/convexp?s=ALL にある `libicu` コンバータ エクスプローラを使用します。サポートされている文字セットは、ICU ライブラリに基づいています。詳細については、ICU のドキュメントを参照してください。

ストリームベースのターミナルで一般的に使用されるパラメータのリストを以下の表に示します。

パラメータ	説明
TermWidth	ターミナルの幅を設定します (デフォルトは 80)。
TermHeight	ターミナルの高さを設定します (デフォルトは 24)。
AutoWrap	自動行折り返しを制御します (デフォルトは N)。行折り返しを有効にするには Y に設定します。このパラメータは vr100 (Unix vt100) ターミナルにのみ適用されます。
TerminalType	ターミナルのタイプを設定します (デフォルトは "xterm")。
LineCodePage	文字のエンコードとデコードに使用するコード ページを指定します (デフォルトは UTF-8)。

セッションに関するパラメータを設定するには、[ターミナル] ステップの [オプション] フィールドにパラメータを入力します。オプションはアンパサンド (&) で区切ります。

以下の接続オプションがサポートされています。

パラメータ	説明
SessionFile	デフォルトの PuTTY 設定ファイルを設定します。このオプションを指定しない場合は、bin ディレクトリ内の `session.plink` ファイルが使用されます。
PublicKeyFile	認証の SSH キー ファイルを指定します。このファイルは PuTTY の秘密鍵の形式で指定する必要があります。
SSHUser	URI の `user@host` のユーザー名の部分にオーバーライドします。ユーザーを指定しなかった場合、ロボットにはダイアログがインタラクティブに表示されます。
SSHPassword	URI の `user:password@host` のパスワードの部分にオーバーライドします。このオプションを指定しなかった場合、ロボットにはダイアログがインタラクティブに表示されます。
SSHHostKey	ホストを設定する SSH ホスト キーを設定します。この設定を使用して、値がホストから送信されたホスト キーと一致しない場合には、接続は拒否されます。この設定を省略し、PuTTY 設定ファイルを使用して設定したホスト キーがない場合、ロボットにダイアログがインタラクティブに表示されるため、ロボットはキーを承認するダイアログを処理する必要があります。ロボットは Windows のレジストリを無視するため、ホスト キーを格納しないことに注意してください。
SSHLog	トラブルシューティングに使用する SSH レベルのログを有効にします。
NetworkTrace	ホストと交換されたすべてのデータを含むトレース ファイルを作成します。

アプリケーション アクション

[アプリケーション アクション] メニューは、[レコーダー ビュー] の [ターミナル] タブを右クリックすると表示されます。メニューには、vt100 シリーズなどのストリームベースのターミナル用の次のアクションが含まれています。

アクション	説明
閉じる	[ターミナル] ウィンドウを閉じます。

Mainframe (tn3270) ターミナルの構成

tn3270 ターミナルとして接続するには、「ターミナル」ステップを挿入し、[**Mainframe (tn3270)**] を選択して、ホスト名、接続タイプ、接続オプションなど、必要なすべてのパラメータを指定します。Mainframe tn3270 ターミナルは、IBM zSeries プラットフォームに基づいています。

SSH の代わりに、tn3270 ターミナルは SSL および TLS プロトコルをサポートします。これらのプロトコルは、認証なしで接続を暗号化します。

キーボードのサポート

Kofax RPA はデフォルトで、基盤となるターミナル エミュレータのデフォルトである 3279-4 カラー 80x43 ターミナルをサポートしています。従来の tn3270 ターミナルは、標準キーボードでは利用できない特殊なキーを備えたキーボードを使用していました。ロボットでこれらのキーを再作成するには、キープレス の [計算されたキー] アクションで次のキーを使用します。

- VK_RETURN
- VK_TAB
- VK_BACKTAB
- VK_Up
- VK_Down
- VK_Left
- VK_Right
- VK_F1
- VK_F2
- VK_F3
- VK_F4
- VK_F5
- VK_F6
- VK_F7
- VK_F8
- VK_F9
- VK_F10
- VK_F11
- VK_F12
- VK_F13
- VK_F14
- VK_F15
- VK_F16
- VK_F17
- VK_F18
- VK_F19
- VK_F20

- VK_F21
- VK_F22
- VK_F23
- VK_F24
- VK_ATTENTION
- VK_BACKSPACE
- VK_CLEAR
- VK_DELETE
- VK_DUPLICATE
- VK_HOME
- VK_INSERT
- VK_PA1
- VK_PA2
- VK_PA3

文字エンコード

ホストの文字エンコードを指定するには、[オプション] フィールドに `charset` パラメータを追加します。

例: `charset=cp930`

Kofax RPA は、tn3270 ターミナルの次の文字セットをサポートしています。

文字セット名	ホスト コード ページ パラメータ
belgian	500
belgian-euro	1148
bracket	037
brazilian	275
chinese-gb18030	1388
cp1047	1047
cp870	870
finnish	278
finnish-euro	1143
french	297
french-euro	1147
german	273
german-euro	1141
greek	423
hebrew	424
icelandic	871
icelandic-euro	1149

文字セット名	ホスト コード ページ パラメータ
italian	280
italian-euro	1144
japanese-kana	930
japanese-latin	939
norwegian	277
norwegian-euro	1142
russian	880
simplified-chinese	935
slovenian	870
spanish	284
spanish-euro	1145
swedish	278
swedish-euro	1143
thai	1160
traditional-chinese	937
turkish	1026
uk	285
uk-euro	1146
us-euro	1140
us-intl	037

ターミナル エミュレーション

別のターミナルをエミュレートするには、`model` パラメータを [オプション] フィールドに追加します。

例: `model=3279-5`

Kofax RPA は、次のターミナル モデルに対応しています。

ターミナル モデル	寸法
3279-2	80x24
3279-3	80x32
3279-4	80x43 (デフォルト)
3279-5	132x27

待機時間の指定

ホストが一時的に利用できない場合 (ホストに到達できない場合、または接続が失敗する場合) にドライバーの待機時間を指定するには、オプション フィールドに `connecttimeout` パラメータを追加して、タイムアウトを秒単位で入力します。

例: `connecttimeout=5`

このパラメータを省略する場合、デフォルトのタイムアウトは 20 秒です。

解決できない理由によって、または接続を再試行できないために接続が失敗した場合、ドライバーはすぐに失敗します。

アプリケーション アクション

[アプリケーション アクション] メニューは、[レコーダー ビュー] の [ターミナル] タブを右クリックすると表示されます。メニューには、tn3270 ターミナルの次のアクションが含まれます。

アクション	説明
閉じる	[ターミナル] ウィンドウを閉じます。
ステップを移動 (行または列)	アプリケーション ツリーで使用されている行または列の座標に基づいて、指定した位置にカーソルを移動します。各パラメータは、水平方向および垂直方向の移動を容易にするためのオプションです。

iSeries (tn5250) ターミナルの構成

tn5250 ターミナルとして接続するには、「ターミナル」ステップを挿入し、[**iSeries (tn5250)**] を選択して、ホスト名、接続タイプ、接続オプションなど、必要なすべてのパラメータを指定します。tn5250 ターミナルは、IBM iSeries プラットフォームに基づいています。

SSH の代わりに、tn5250 ターミナルは SSL および TLS プロトコルをサポートします。これらのプロトコルは、認証なしで接続を暗号化します。

デフォルトと異なるポートを使用するには、[ホスト] フィールドでポートを指定します (`TerminalHost:11625` など)。

Kofax RPA は 80x24 と 132x27 のモードのターミナルをサポートしています。

接続オプション

接続オプションは「ターミナル」ステップの [オプション] フィールドで指定します。オプションはアンパサンド (&) で区切ります。オプションの文字列にパーセント (%) エスケープ文字を含めることができます。

値 `env.TERM` は、URL によってモードが切り替わった後のクライアント ターミナル タイプです。デフォルトのターミナル タイプは、"IBM-3179-2" です。`env.TERM` パラメータは、tn5250 ターミナルに対してのみ有効であることに注意してください。

Kofax RPA は、次のターミナル タイプに対応しています。

- "IBM-3477-FC"
- "IBM-3477-FG"
- "IBM-3180-2"
- "IBM-3179-2" (デフォルト)
- "IBM-3196-A1"
- "IBM-5292-2"
- "IBM-5291-1"
- "IBM-5251-11"
- "IBM-5555-B01" (DBCS が有効)
- "IBM-5555-C01" (DBCS が有効)

文字エンコード

ホストの文字エンコードを指定するには、[オプション] フィールドに `LineCodePage` パラメータを追加します。

例：`LineCodePage=cp838`

tn5250 ターミナルの場合、Kofax RPA は ICU ライブラリでサポートされているすべての EBCDIC コード ページをサポートします。`LineCodePage` 設定は、ホストへの接続に使用されるターミナル エミュレータのコード ページ設定またはターミナル セッションの `QCHRID` プロパティの値と一致する必要があります。

この表には、頻繁に使用されるコード ページが示されています。

文字セット名	ホスト コード ページ パラメータ
US/Canada	cp037 (デフォルト)
Multinational	cp500
Thai	cp838
Japanese	cp930, cp939
Simplified Chinese	cp935
Korean	cp933
EBCDIC 273 とユーロ通貨の更新	cp1141

デバイスの割り当て

デフォルトでは、5250 ホストが、リクエストされたターミナル タイプに対応している最初のデバイスを割り当てるか、新しいデバイスの作成を試行します。ロボットが特定のデバイスに接続する必要がある場合は、[オプション] フィールドに `env.DEVNAME=<device>` パラメータを追加します。

例：iSeries ホストの DEVROBOT デバイスに接続するには、パラメータ `env.DEVNAME=DEVROBOT` を追加します。

リクエストされたデバイスが存在しないか、そのターミナル タイプに対応していない場合、または使用できない (使用中または `varied off` の状態) 場合は、接続が失敗します。

強化されたインターフェイス サポート

プログラムできないワークステーションがホスト上で実行されている場合に、強化されたインターフェイスを有効にするには、[オプション] フィールド に `enhanced=on` クエリ パラメータを追加します。

この設定を使用すると、継続的に入力するフィールドのサポートが有効になります。

拡張インターフェイスが有効の場合、DirectDraw 表面ウィンドウとグラフィカル機能はサポートされません。

キーボードのサポート

従来の 5250 ターミナルは、標準キーボードでは利用できない特殊なキーを備えたキーボードを使用していました。ロボットでこれらのキーを再作成するには、キープレス の [計算されたキー] アクションで次のキーを使用します。

- VK_RETURN
- VK_ENTER
- VK_TAB
- VK_BACKTAB
- VK_UP

- VK_DOWN
- VK_LEFT
- VK_RIGHT
- VK_CLEAR
- VK_BACKTAB
- VK_F1
- VK_F2
- VK_F3
- VK_F4
- VK_F5
- VK_F6
- VK_F7
- VK_F8
- VK_F9
- VK_F10
- VK_F11
- VK_F12
- VK_F13
- VK_F14
- VK_F15
- VK_F16
- VK_F17
- VK_F18
- VK_F19
- VK_F20
- VK_F21
- VK_F22
- VK_F23
- VK_F24
- VK_ROLLDN
- VK_ROLLUP
- VK_BACK
- VK_HOME
- VK_END
- VK_INSERT
- VK_DELETE
- VK_RESET
- VK_PRINT
- VK_HELP
- VK_SYSREQ

- VK_CLEAR
- VK_REFRESH
- VK_FIELDEXIT
- VK_TESTREQ
- VK_TOGGLE
- VK_ERASE
- VK_ATTENTION
- VK_DUPLICATE
- VK_FIELDMINUS
- VK_FIELDPLUS
- VK_PREVWORD
- VK_NEXTWORD
- VK_PREVFLD
- VK_NEXTFLD
- VK_FIELDHOME
- VK_EXEC
- VK_MEMO
- VK_COPY_TEXT
- VK_PASTE_TEXT
- VK_NEWLINE
- VK_ERASE_EOF
- VK_KANJI

詳細については、製造元の 5250 ターミナルのドキュメントを参照してください。

自動化を実行できるようにするために、Kofax RPA は次のキーを提供します。

- VK_VIRTUAL_FIELDEXIT
  このキーは VK_FIELDEXIT として機能しますが、機能するのは、フィールドの最初の位置にカーソルが置かれていない場合に限られます。このキーを使用して、フィールドが不完全な場合に終了するか、カーソルが次のフィールドに移動した場合に終了を抑制します。
- VK_DBCS_SPACE
  DBCS 空間を入力します。

アプリケーション アクション

[アプリケーション アクション] メニューは、[レコーダー ビュー] の [ターミナル] タブを右クリックすると表示されます。メニューには、tn5250 ターミナルの次のアクションが含まれます。

アクション	説明
閉じる	[ターミナル] ウィンドウを閉じます。
ステップを移動 (行または列)	アプリケーション ツリーで使用されている行または列の座標に基づいて、指定した位置にカーソルを移動します。各パラメータは、水平方向および垂直方向の移動を容易にするためのオプションです。

アクション	説明
フィールドに入力 (テキスト)	指定したテキストをフィールドに入力し、[FieldExit] キーを使用して次のフィールドに進みます。

NonStop (tn6530) ターミナルの構成

tn6530 ターミナルとして接続するには、「ターミナル」ステップを挿入し、[**Nonstop (tn6530)**] を選択して、接続タイプ、クレデンシャル、接続オプションなどのパラメータを指定します。tn6530 ターミナルは、Hewlett-Packard Enterprise (HPE) NonStop プラットフォームに基づいています。

⚠ この構成は、ホストが HPE NonStop プラットフォームで実行されている場合にのみ使用してください。Unix、Linux、および Windows プラットフォームについては、「Unix、Linux、および Windows (vt100 シリーズおよび ANSI) ターミナルの構成」を参照してください。

Kofax RPA は TN6530-8 ターミナルを 80x24 画面でエミュレートします。ターミナルは対話モードで開始します。

接続オプション

「ターミナル」ステップの [オプション] フィールドで接続オプションを指定します。オプションはアンパサンド (&) で区切ります。オプションの文字列にパーセント (%) エスケープ文字を含めることができます。

例：`NetworkTrace=MyLogfile.log&LineBuffer=2048`

`tn6530` の [接続] アクションは、[ホスト] フィールドで別のポートが明示的に指定されていない限り (例: `TerminalHost:11625`)、Telnet サービス (ポート 23) に接続します。

以下のオプションがサポートされています。

- `NetworkTrace=<logfile>`
  ホストと交換されたすべてのデータを含むトレース ファイルを作成します。
- `ColorMap=<color-map>`
  ターミナルをカラー サポートに設定し、オプションでデフォルトのカラー マップを提供します。
- `LineBuffer=<lines>`
  ターミナルのバッファリングされる行数を設定します。これらの行は、ページングおよびスクロールに使用できます。有効な番号は 24 ~ 65536 です。指定しない場合、ターミナルは 1024 行のバッファを保持します (対話モードのみ)。

`ssh6530` の [接続 **(SSH)**] アクションは、[ホスト] フィールドで別のポートが指定されていない限り (例: `TerminalHost:11625`)、Telnet サービス (ポート 22) に接続します。SSH ログインのクレデンシャルを指定するには、「ターミナル」ステップで [クレデンシャル] オプションを使用します。

以下のオプションがサポートされています。

- `SessionFile=<template>`
  デフォルトの PuTTY 設定ファイルを設定します。このオプションを指定しない場合は、bin ディレクトリ内のデフォルト `session.plink` ファイルが使用されます。
- `PublicKeyFile=<file>`
  認証の SSH キー ファイルを指定します。このファイルは PuTTY の秘密鍵の形式で指定する必要があります。
- `SSHUser=<user>`
  ロボット ステップの [ユーザー名] フィールドのユーザー名をオーバーライドします。ユーザーを指定しなかった場合、ロボットにはダイアログがインタラクティブに表示されます。

- `SSHPassword=<password>`
  ステップの [パスワード] フィールドのパスワードをオーバーライドします。このオプションを指定しなかった場合、ロボットにはダイアログがインタラクティブに表示されます。
- `SSHHostKey=<hostkey>`
  ホストを設定する SSH ホスト キーを設定します。この設定を使用して、値がホストから送信されたホスト キーと一致しない場合には、接続は拒否されます。
  この設定を省略し、PuTTY 設定ファイルを使用して設定したホスト キーがない場合、ロボットにダイアログがインタラクティブに表示されるため、ロボットはキーを承認するダイアログを処理する必要があります。ロボットは Windows のレジストリを無視するため、ホスト キーを格納しないことに注意してください。
- `SSHLog=<logfile>`
  トラブルシューティングに使用する SSH レベルのログを有効にします。
- `NetworkTrace=<logfile>`
  ホストと交換されたすべてのデータを含むトレース ファイルを作成します。
- `ColorMap=<color-map>`
  ターミナルをカラーをサポートするように設定し、オプションでデフォルトのカラー マップを提供します。
- `LineBuffer=<lines>`
  ターミナルのバッファリングされる行数を設定します。これらの行は、ページングおよびスクロールに使用できます。有効な番号は 24 ~ 65536 です。指定しない場合、ターミナルは 1024 行のバッファを保持します (対話モードのみ)。

その他のオプションは、`SessionFile` テンプレートの設定を置き換えます。`SessionFile` テンプレートで設定されていないオプションは無視されます。

ログ ファイルの作成時に、`logfile` パラメータは、次の表に示す置き換えを実行します。

パターン	置き換え	例
$P	ロボットを実行する処理の PID (値はプラットフォームに依存します)	12928
$T	UNIX エポック形式の UTC 時刻	1524649335
$D	ISO 8601 形式でのローカル時刻	20180425T114215
$H	ホストの IP アドレス	127.0.0.1
$P	ホストのポート	22
$$	単独のドル記号 ($)	$

キーボードのサポート

以下の 6530 キーは、保護ブロック モードでサポートされます。

- Return、Shift + Return、Ctrl + Return
- Home、Ctrl + Home
- End
- Backspace
- Tab、Shift + Tab
- Insert、Alt + Insert、Ctrl + Insert

- Delete、Alt + Delete、Ctrl + Delete
- Alt + 2、Shift + Alt + 2
- 方向キー
- ファンクション キー F1 ~ F16。(互換性のため、Alt+F1 ~ F1-F6 も F11 ~ F16 にマッピングされます)
- 6530 ファンクション キー: Alt + Up、Alt + Down、PgUp、Alt + PgUp、PgDn、Alt + PgDn

Kofax RPA は対話モードで次の 6530 キーをサポートします。

- Return、Shift + Return
- ファンクション キー F1 ~ F16。(互換性のため、Alt+F1 ~ F6 は F11 ~ F16 にマッピングされます)

さらに、次のキーが キープレス の [計算されたキー] アクションでサポートされています。

- VK_CLOSE
  このキーは、セッションを終了して TN6530 ターミナルを閉じるのに使用されます。
- VK_F11 ~ VK_F16
  これらのキーは、F11 ~ F16 ファンクション キーにマッピングされた 6530 のキーの組み合わせ Alt+F1 ~ Alt+F6 に対応して機能します。

カラー サポート

`ColorMap=` 設定は、ターミナルをモノクロから設定不可のカラー モードに切り替えます。ホストはエスケープ コードを送信してカラー マップを更新できます。初期カラー マップを接続文字列で設定するには、32 バイトの文字列を、32 個の属性の組み合わせに対するカラー マッピングを指定する 16 進数形式で指定します。属性と色のマッピング スキームの説明については、製造元の 6530 のドキュメントを参照してください。

既知の問題と制限

- 対話モードと非保護モードのサポート範囲は限定されます。
- 拡張カラー サポートと EM3270 モードは利用できません。
- ターミナルに対するローカル操作を実行するコマンドがサポートされません。

メッセージ ラインで、サポートされていないエスケープ シーケンスおよびキーが「`**ERRORS:`」マーカーでトラッキングされます。

アプリケーション アクション

[アプリケーション アクション] メニューは、[レコーダー ビュー] の [ターミナル] タブを右クリックすると表示されます。メニューには、tn6530 ターミナルの次のアクションが含まれます。

アクション	説明
閉じる	[ターミナル] ウィンドウを閉じます。
ステップを移動 (行または列)	アプリケーション ツリーで使用されている行/列の座標に基づいて、指定した位置にカーソルを移動します。各パラメータは、水平方向および垂直方向の移動を容易にするためのオプションです。

基本 ターミナル チュートリアル

ターミナルの前提条件と設定については、ターミナル エミュレータの自動化を参照してください。

このチュートリアルでは、5250 ターミナルに接続してログインし、コマンドを実行してターミナルから情報を抽出します。

1. 既存の口ボット を開くか、新規に作成します。
2. 既存のプロジェクトを開くか、(スマート再実行 (フル) モードで) 新しいプロジェクトを作成し、「口ボットの呼び出し」ステップを使用して新しいベーシック エンジン 口ボット を追加します。このステップで開く口ボットの名前を指定します。
3. ターミナルにログインするための口グイン名とパスワードを格納する変数を追加します。また、出力テキストを格納する変数を追加します。出力値を伴う変数をリターン ステップに追加します (`=textVariableName`)。
4. 「口ボットを呼び出す」ステップを実行して、ツールバーの [口ボットにステップ] ボタンをクリックします。
5. 口ボット ワークフローで、必須のパラメータを含む「ターミナル」ステップを追加します。このチュートリアルでは、5250 ターミナルに接続します。一般的に、接続文字列は：`tn5250://<hostname>:<portnumber>?env.TERM=<terminal type>` です。`env.TERM` パラメータは、tn5250 ターミナルに対してのみ有効であることに注意してください。
   - エミュレータ: iSeries(tn5250)
   - アクション: 接続
   - ホスト: `terminal5250_server:11623`
   - オプション: `env.TERM=IBM-3477-FC`

   ここで、`terminal5250_server` はターミナル名または IP アドレス、`:11623` はターミナル接続ポート番号です。`env.TERM` パラメータを省略した場合、エミュレータはデフォルトのターミナル タイプ (IBM-3179-2) に接続されます。ターミナル エミュレータの自動化の「サポートされている tn5250 ドライバー ターミナル」を参照してください。

   「ターミナル」ステップを使用してターミナルに口ボット ワークフローを再実行するには、口ボットを終了し、Design Studio で「口ボットを呼び出す」ステップを使用してベーシック エンジン 口ボットを更新して、[口ボットにステップ] をクリックします。ターミナルを閉じずに口ボットを再実行すると、別のターミナル ウィンドウが開き、口ボットの実行に失敗することがあります。

6. ターミナルに Enter キーの入力が必要な場合、レコーダー ビューでターミナルを右クリックし、キー プレス ステップを選択します。デフォルトでは、Enter キーが選択されます。
7. レコーダー ビューでターミナルの [ユーザー ID] フィールドを右クリックし、**[テキストを入力]** > **[変数から]** > **login** を選択し、ユーザー名を含む変数を選択します。[ステップ] をクリックして、テキスト フィールドにこのテキストをタイプします。
8. パスワード フィールドに移動するには、キー プレス ステップを追加し、「キー」フィールドで、**[標準キー]** > **[タブ]** を選択します。[ステップ] をクリックします。カーソルが Password フィールドに移動します。
9. ターミナル ウィンドウの Password フィールドを右クリックし、**[テキストを入力]** > **[変数から]** を選択し、パスワードを使って変数を選択します。テキスト フィールドにこのテキストをタイプするには、[ステップ] をクリックします。
10. ログインするには、レコーダー ビューでターミナルを右クリックし、[キー プレス] (デフォルトで Enter) を選択します。

11. ターミナルに Enter キーの入力が必要な場合、レコーダー ビューでターミナルを右クリックし、[キー プレス] を選択します。デフォルトでは、Enter キーが選択されます。
12. 必要なコマンドを実行した後、ターミナル ウィンドウから情報が抽出できます。テキスト行を抽出するには、行を右クリックし、**[ここから値を抽出]** > **[テキスト]** > **<変数名>** を選択します。[ステップ] をクリックします。ワークフロー状態ビューの [変数] 分岐に抽出した値が表示されます。
    ターミナル ウィンドウ全体のスクリーン ショットを取得するには、(バイナリ変数を戻るステップ (`=binaryVariableName`) に事前に追加する必要があります)、レコーダー ビューで画面の要素を選択し、[画像を次へ抽出] > **<バイナリ変数名>** をクリックします。後から、ベーシック エンジン ロボットでバイナリ変数の情報を画像に変換することができます。[ステップ] をクリックします。

ターミナルから情報を抽出すると、ベーシック エンジン ロボット エディター ウィンドウに戻り、抽出した情報を使用することができます。

Web サイトのアクセス

ロボット ワークフローを作成する際に、組み込みブラウザで Web サイトを開いて、アクション ステップを使用して情報を抽出し、サイトをナビゲートすることができます。組み込みブラウザは、Chromium (CEF) などの選択されたエンジンに基づいています。ナビゲートするには、ロボット ステップを使用します。アプリケーション用のブラウザ選択の詳細については、ブラウザのタイプを参照してください。

Kofax RPA の Chromium ブラウザは、次のプロトコルをサポートしています。

- `http:`
- `about:`
- `https:`
- `file:`

Web サイトを開くには、ブラウザ ステップを挿入し、**[ブラウザ]** リストでブラウザ エンジンを選択して、アクションとして **[ページ読込]** を選択してから、アドレスなどの必要なすべてのパラメータを **[URL]** フィールドに指定します。ページの読み込みを含むステップが完了するか、Web ページ上で他の変更が発生した場合は、ツリーの変更停止ガードを使用します。待機時間に制限を設けるため、ツリーの変更停止ガードにタイムアウトを追加することをお勧めします。。詳細については、ガード チョイスを参照してください。

- Web サイトのブラウズのコマンド タイムアウトは、Desktop Automation でワークフローを実行するための Design Studio 設定ウィンドウにある [**Design Studio**] タブ、または RoboServer 実行のための [**RoboServer** 設定] アプリケーションの [セキュリティ] タブにある [オートメーション デバイス] セクションで設定します。『Kofax RPA 管理者ガイド』の「ランタイム」>「セキュリティ」を参照してください。
- 現在のところ、組み込みブラウザを使用している場合は、1 つまたは複数のロボット内でシステム クリップボードを使用してコンテンツをコピーしたり、貼り付けたりすることはできません。コンテンツをコピーして貼り付ける代わりに、「値を抽出」ステップを使用することをお勧めします。

次のリンクを使用して、サブトピック間をすばやく移動できます。

ブラウザ インターフェイス

組み込みブラウザには、次のコントロールが含まれています。

ボタン	説明
	戻る: 1 ページ前に戻ります。
	進む: 1 ページ先に進みます。
	リロード: 現在のページをリロードします。ページの読み込み中、ボタンにはブラウザがビジー状態であることを示すクロスが表示されます。
	**[URL]** フィールド: 現在読み込まれているページの URL、またはそのページに移動する前に貼り付けた URL が表示されます。
	ナビゲート: URL フィールドに入力した URL に移動します。ボタンが見えない場合は、[レコーダー ビュー] ウィンドウを右にスクロールしてください。
	ページの保存ボタン: 現在開いているページを HTML 形式で保存します。ボタンが見えない場合は、[レコーダー ビュー] ウィンドウを右にスクロールしてください。詳細については、以下を参照してください。
	プロキシ設定: プロキシ設定ダイアログ ボックスを開きます。ボタンが見えない場合は、[レコーダー ビュー] ウィンドウを右にスクロールしてください。詳細については、以下を参照してください。
	**PDF** に印刷: 現在開いている Web ページを PDF ファイルに保存します。ボタンが見えない場合は、[レコーダー ビュー] ウィンドウを右にスクロールしてください。詳細については、以下を参照してください。

組み込みブラウザ内の **URL**

ブラウザ ウィンドウでは、ツールバーに URL テキスト フィールドが表示されます。これは、読み込まれた Web ページの URL を示します。URL を選択し、アクション ステップを使用して変数に値を抽出することができます。ツリー ビューには URL も表示されます。

- URL フィールドのテキストを選択するには、クリックを使用して URL フィールドをクリックします。
- アドレスを変更するには、クリック ステップを使用して URL フィールドをクリックし、テキストを入力ステップまたは変数を使用してアドレスを入力します。
- 入力した URL に移動するには、URL フィールドの右側にある [ナビゲート] ボタンをクリックします。

プロキシを構成する

デフォルトでは、Desktop Automation サービスのすべてのロボットは Kofax RPA グローバル プロキシ設定を使用します。Desktop Automation サービスは、Design Studio および Management Console と同じプロキシ設定を使用します。プロキシ サーバーのプロパティの詳細については、「Design Studio でのプロキシ サーバーの設定」および「Management Console でのプロキシ サーバーの設定」を参照してください。

Desktop Automation サービスの組み込みブラウザのローカル プロキシ設定は、Kofax RPA グローバル プロキシ設定よりも優先順位が高いことに注意してください。タスクでローカル プロキシ設定を使用する必要がない場合、ロボットは Kofax RPA グローバル プロキシ設定を使用することを確認してください。

ロボットで組み込みブラウザのプロキシ設定を変更するには、ブラウザのツールバーの [プロキシ設定] ボタンをクリックします。以下のプロキシ オプションがあります。

- [ダイレクト]: プロキシは使用されません。
- [固定]: ホスト、ポート、迂回リストなどの固定プロキシ設定を指定します。
- [PAC]: プロキシ自動設定スクリプト ファイルの URL を指定します。
- [自動]：Web プロキシ自動検出プロトコルなど、ネットワークで自動プロキシ設定が提供されている場合は、このオプションをクリックします。
- [システム]: ロボットを実行しているコンピュータからプロキシ設定をコピーするには、このオプションを選択します。

プロキシ設定を完了してから [**OK**] をクリックして設定を保存し、ダイアログ ボックスを閉じます。

プロキシの認証のネゴシエート

認証のネゴシエートは、プロキシ サーバーで使用することができます。プロキシ サーバーがロボット用に設定されている場合、サーバーでネゴシエート プロトコルがサポートされていると、このプロトコルが自動的に使用されます。

PDF に印刷

ブラウザのツールバーの [**PDF** に印刷] ボタンを使用して、Web ページを PDF 形式で保存できます。

1. クリック ステップを使用して [**PDF** に印刷] ボタンをクリックします。
   このステップを実行すると、ブラウザで [ページを **PDF** ドキュメントとして保存] ダイアログ ボックスが開きます。

2. 「テキストの置き換え」ステップを使用して、ファイル名を含むフル パスを指定します。デフォルトでは、ファイルは現在のユーザーに構成された一時ファイルフォルダに保存されます。
   [**OK**] をクリックします。
   - 必要に応じて、ダイアログ ボックスで [ロボット ファイル システム] を選択して、ファイルをロボット ファイル システムに保存できます。[**RFS** ファイル名] フィールドに、設定したファイル システムへのパスとファイル名を入力します (**myshare/downloaded.pdf** など)。ファイル システム名は、Management Console 内の [ロボット ファイル システム] セクションでの指定に対応している必要があります。
3. [保存] をクリックします。

ページの向き、用紙サイズ、縮尺などの PDF ファイルの設定を調整するには、ブラウザ ステップの [**PDF** 設定] プロパティで必要なオプションを指定します。

開いている HTML ページまたはドキュメントをハードウェア プリンタに送信する場合、プリンタ選択用のポップアップ ダイアログ ボックスの処理には制限があるため、代わりに[ページを **PDF** ドキュメントとして保存] ダイアログ ボックスが自動的に開きます。これにより、ドキュメントをファイル システムまたはロボット ファイル システムに PDF 形式で保存して、後で使用できるようになります。

HTML でのページの保存

ブラウザ ツールバーの [ページの保存] ボタンを使用すると、現在開いているページを HTML 形式で保存できます。

1. クリック ステップを使用して [ページの保存] ボタンをクリックします。
   このステップを実行すると、ブラウザで [名前を付けて保存] ダイアログ ボックスが開きます。
2. 「テキストの置き換え」ステップを使用して、ファイル名を含むフル パスを指定します。デフォルトでは、ファイルは現在のユーザーに構成された一時ファイルフォルダに保存されます。
   [**OK**] をクリックします。
3. [保存] をクリックします。

アプリケーション アクション

アプリケーション レベルのアクションとはアプリケーション全体に適用されるアクションのことで、[レコーダー ビュー] のアプリケーション タブを右クリックして使用できます。Chromium 組み込みブラウザのアプリケーション レベルのアクションのリストには、次のアクションが含まれます。

アクション	説明
[アプリケーション名を設定]	ブラウザ アプリケーションの `name` タグに値を追加します。アプリケーションを識別でき、堅牢で信頼性の高いファインダーの作成に役立つ名前を指定します。開かれたページにあるリンクを使用して新しいブラウザ ウィンドウに Web ページがロードされると、新しく開いたブラウザ アプリケーションの `name` タグには、たとえば `name="mainpage(2)"` のように、親アプリケーションの名前と括弧で囲まれた数字が含まれます。 エラーを回避するために、ロボット内のアプリケーションにはそれぞれ異なるアプリケーション名を割り当てます。

アクション	説明
ウィンドウを閉じる	開いているアプリケーション ウィンドウを閉じるステップを追加します。
ウィンドウを複製	アプリケーション タブを複製し、新しいタブで開くステップを追加します。アプリケーションの状態は保持されます。
JavaScript の実行	メイン ウィンドウ コンテキストの現在のページで JavaScript を実行するステップを追加します。 • **JavaScript**: 実行するコードを入力します。 • **JS** 実行結果: 結果を保存するバイナリ形式の変数を指定します。 • [結果を待機] を選択すると、コードが実行され、結果が [「JS 実行結果」] 変数に書き込まれるまでロボットは待機します。 デフォルトのタイムアウトは 30 秒です。この期間内にアクションが完了しない場合は、例外がスローされます。JavaScript コードにエラーが含まれる場合も、例外がスローされます。 このオプションをオフにすると、ロボットはコードの実行が開始された直後に (実行の完了を待たずに) 次のステップに進みます。
ナビゲート	指定した URL を開くステップを追加します。[**URL**] フィールドに URL を入力します。
再ロード	現在の Web ページを再ロードします。
戻る	1 ページ前に戻ります。ブラウザ ツールバーの [戻る] ボタンをクリックする操作と同様です。
進む	1 ページ先に進みます。ブラウザ ツールバーの **[進む]** ボタンをクリックする操作と同様です。

コンポーネント アクション

コンポーネント アクションとは選ばれたコンポーネントまたは要素に適用されるアクションのことで、[レコーダー ビュー] またはツリー ビューで要素を右クリックすることで使用できます。Chromium 組み込みブラウザのコンポーネント レベルのアクションのリストには、次のアクションが含まれます。

RPA は CEF ブラウザを自動的に検出し、適切なツリー モードに切り替えますが、右クリックして **[ツリー モード]** > **[ISA]** を選択すると、アプリケーション全体および特定の要素のツリー モードを ISA に変更できます。

アクション	説明
リンクを新しいウィンドウで開く	Web ページに存在するリンクで使用されます。選択したリンクを [レコーダー ビュー] の新しいタブで開くステップを追加します。
指定タグまでスクロール	選択した要素にページをスクロールするステップを追加して、ウィンドウの中央に要素が表示されるようにします。

アクション	説明
オプション選択	メニューから単一のオプションまたは複数のオプションを選択できるようにするステップを追加します。 • アイテムのリスト、アイテムのグループ、またはドロップダウン リストから単一のオプションを選択できます。 折りたたまれたドロップダウン リストからオプションを選択する必要がある場合は、レコーダー ビューのメニューをクリックし、ツリー ビューで必要なオプションを選択します。 • Ctrl キーを押しながらクリックする操作と同様に、アイテムのリストまたはアイテムのグループから複数のオプションを選択できます。 リストまたはグループから単一のオプションをすでに選択している場合は、このリストまたはグループの別のアイテムに「オプション選択」ステップを追加し、ステップのプロパティで **[複数選択]** をオンにします。ステップを実行すると、両方のオプションが選択された状態で表示されます。さらにオプションを選択するには、選択する追加オプションごとに同じ手順を実行します。 複数の選択を解除する必要がある場合は、同じ設定を使用します。
ターゲット抽出	データ URL 値を持つ href 属性を使用したアンカー HTML 要素に適用できます。これにより、ユーザーはデータ URL 値をバイナリ変数に抽出できます。値は、必要に応じて base64 形式からバイナリにデコードされます。 バイナリ変数の利用可能なタイプには、バイナリ、画像と PDF が含まれます。[状態] ペインで画像をプレビューできること、およびターゲット抽出ステップで保存した変数から [レコーダー ビュー] で PDF を使用して PDF を開くことができることを除き、これらはすべて同等です。

Chrome Inspector を使用したデバッグ

Kofax RPA は、「参照」ステップで Chrome Inspector を使用して、Chromium ブラウザ エンジンをデバッグする操作をサポートします。Inspector を使用すると、Web サイトとブラウザ間の相互作用、および Web コンテンツの処理中に発生したエラーを分析するために必要な情報を抽出して保存できます。情報は HTTP アーカイブ (HAR) 形式で保存されます。ログをファイルに保存するには、次の手順を実行します。

1. `<Kofax RPA インストール フォルダ>\nativelib\hub\windows-x64\[hub バージョン番号]\node_modules\cef` にある `cef.cfg` ファイルをテキスト エディターで開きます。

   例：`C:\Program Files\Kofax RPA 11.5.0.0\nativelib\hub\windows-x64\1267\node_modules\cef`

2. `show_dev_tools` プロパティを `true` に設定し、ファイルを保存して、ロボットを再ロードします。

   Web ページをロードする ブラウザ ステップを実行したら、該当する Web ページに接続されている **[Chrome DevTools]** ウィンドウが開きます。

3. **[Chrome DevTools]** ウィンドウで、[ネットワーク] タブを選択し、[ログの保存] をクリックして、Ctrl + R を押します。ネットワーク トレースがウィンドウにロードされます。

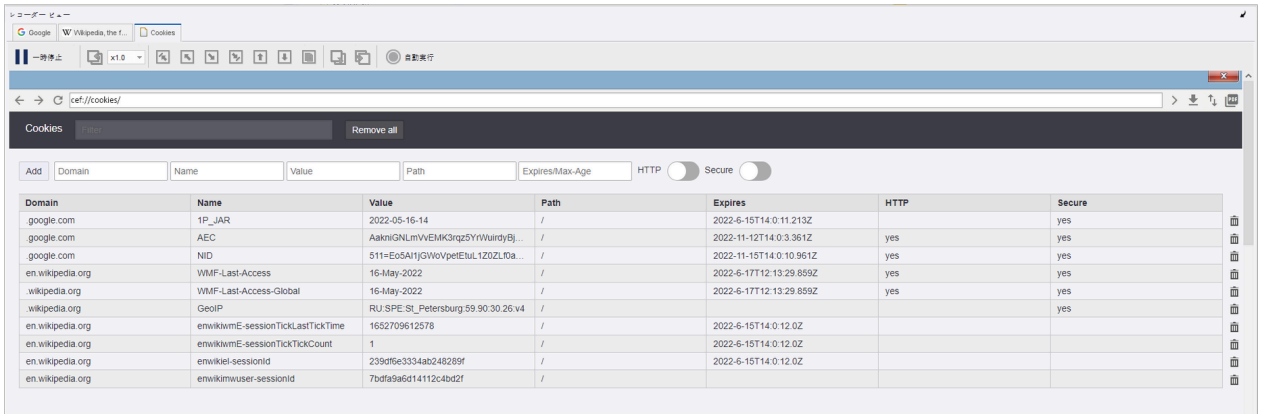
4. トレースを右クリックし、[コンテンツ付きの **HAR** としてすべて保存] を選択して、ネットワークトレースをファイルに保存します。HTTP アーカイブ インスペクターでファイルを開くことができます。

Chromium 組み込みブラウザでの Cookie の管理

CookieStore から Cookie を監視、追加、または削除するには、ロボットの組み込みブラウザを使用します。

[Cookie] ページでの Cookie の管理

Chromium 組み込みブラウザで Cookie を管理するには、「ブラウザ」ステップを挿入して、[Cookie] ページを開きます。次に [ブラウザ] で [**Chromium**] を選択し、[**URL**] フィールドに `cef://cookies` と入力して、ステップを実行します。



CookieStore への新しい Cookie の追加

1. 対応するテキスト ボックスで Cookie 属性を指定します。
   - Domain
   - Name
   - Value
   - Path
   - Expires/Max-Age
2. HTTP 制限を有効にするには、[**HTTP**] トグルをクリックします。
3. 暗号化接続 (HTTPS) を有効にするには、[セキュア] をクリックします。
4. Cookie を追加するには [追加] ボタンをクリックしてします。

Cookie のフィルタリング

[Cookie] ページのすべての Cookie は任意の属性値でフィルタリングできます。

Cookie を属性でフィルタリングするには、[フィルタ] フィールドに属性値を入力します。

Cookie の削除

[Cookie] ページを使用して、CookieStore から Cookie を個別にまたは一括で削除します。

- CookieStore から Cookie を削除するには、行の最後にある [削除] ボタンをクリックします。
- CookieStore からすべての Cookie を削除するには、[すべて除去] ボタンをクリックします。

「ブラウザ」ステップによる Cookie の管理

リクエストによって直接 Cookie を追加または削除するには、「ブラウザ」ステップを使用します。

- CookieStore に新しい Cookie を追加するには、「ブラウザ」ステップの [**URL**] フィールドに次のテキストを入力します (値を置き換えます)。

  ```
  cef://cookies/create?domain=MyDomain.com&name=MyCookie&value=MyValue&path=/MyPath&expires=2020-12-31T20:30:10.000Z&http=on&secure=off
  ```

- CookieStore から Cookie を削除するには、「ブラウザ」ステップの [**URL**] フィールドに次のテキストを入力します。

  ```
  cef://cookies/remove?domain=.MyDomain.com&name=MyCookie&value=MyValue&path=/MyPath&expires=2020-12-31T20:30:10.000Z&http=on&secure=off
  ```

- すべての Cookie を CookieStore から削除するには、「ブラウザ」ステップの [URL] フィールドに次のテキストを入力します。

  ```
  cef://cookies/removeall
  ```

Cookie からの値の抽出

[Cookie] ページでは、Cookie 値をテキスト変数に抽出できます。

コンテキスト メニューを使用して、Cookie 属性を右クリックしてその値を抽出します。

アテンデッド オートメーション

アテンデッド オートメーションは、リモート デバイスのイベントに対応するトリガー ロボットを作成してリモート コンピュータを自動化する方法です。次のタスクと情報を使用して、トリガーを持つロボットを使用するための開発環境および本番環境を設定します。

要件と設定については、Desktop Automation サービス を参照してください。

前提条件: ロボットの開発中に、リモート デバイスの Desktop Automation サービスを **[**シングル ユーザー**]** モードに設定して Design Studio からデバイスにアクセスできるようにします。

1. トリガー ロボット用にシンプルな直接接続デバイス マッピングを作成します。
2. [デバイスの追加] ダイアログ ボックスを使用して、作成したマッピングを「トリガー リファレンス」として「ロボットの呼び出し」ステップの [デバイス] に追加します。
3. トリガー チョイスを ロボットに追加します。

4. 開発が完了した後に、リモート デバイスで Desktop Automation サービスの設定を開き、次の手順を実行します。
   - [シングル ユーザー] オプションをオフにして、ロボットを展開した Management Console を指定します。
   - [**Windows**] タブを開き、[パッケージをロック] を選択します。

トリガーを持つロボットを Management Console にアップロードした後、ロボットを [トリガー マッピング] セクションのユーザーとラベルにマッピングします。こうすると、作成したマッピングに基づいて、Management Console から Desktop Automation サービスにトリガーのリストが提供されるようになります。リモート デバイスにトリガー イベントが検出されると、Desktop Automation サービスは Management Console に通知を送信し、ロボットはプログラムされたいくつかのステップを実行します。たとえば、特定のアプリケーションを開いた場合にデータを挿入または抽出するようにロボットをプログラムすることができます。トリガー チョイス ステップを使用して、特定のイベントが検出されたときに開始するトリガーおよびアクションを定義します。

手順の詳細については、「アテンデッド オートメーション - はじめに」を参照してください。

トリガーを持つロボットに対する禁止事項

- トリガーを持つロボットをスケジュールに追加したり、トリガーを持つロボットを手動で開始したりしないでください。スケジュールに従って、または手動で開始されたロボットは、無制限に (または指定された RoboServer タイムアウト期間の間) 待機し続けます。これは、どのリモート Desktop Automation サービスでも、トリガーが有効にならないためです。
- トリガーを持つロボットを Kapplet に追加しないでください。
- トリガーを持つロボットでインテリジェント スクリーン オートメーション (ISA) を使用しないでください。

Desktop Automation サービスを構成するときには、ラベルを使用してオートメーション デバイスを区別できます。たとえば、アクティブなユーザーが使用するコンピュータに Desktop Automation サービスがインストールされ、手動タスクでユーザーを支援するためにアテンデッド ロボットを実行する必要がある場合、「対話型」などの Desktop Automation サービスにラベルを個別に追加できますコンピュータがアクティブに使用されておらず、コンピューター上でアテンデッド (通常の) ロボットを実行できる場合、「非対話型」のようなラベルを追加できます。

Management Console でトリガー イベントを持つロボットにユーザーとラベルを割り当てるには、トリガー マッピング を使用します。

また、Management Console の [リポジトリ] > [ロボット] セクションでマッピングを割り当て、トリガーの停止やアクティブ化を行うこともできます。

トリガー イベントが検出されると、ロボットはユーザーがマウスやキーボードを使用できないようにすることがあります。ユーザーに、ロボットによって実行されたアクションを知らせるには、通知ステップを使用します。

Management Console でトリガーがサスペンドされている場合、ロボットはリモートの Desktop Automation サービスによってトリガーされません。トリガー情報の更新はリモート デバイスとの接続中に実行されて、それ以降は 1 分おきに実行されることに注意してください。トリガーがサスペンドと表示されていても、Management Console で引き続き実行されている場合があります。この場合は、トリガー イベントが引き続き検出されることがあります。

アテンデッド オートメーション - はじめに

Kofax RPA では、ネットワーク接続しているコンピュータ上の Windows および Java アプリケーションを対象とする作業プロセスを自動化できるロボットを作成して、これらのアプリケーションの制御を自動化することができます。

アテンデッド オートメーションは、リモート デバイスのイベントに対応するトリガー ロボットを作成してリモート コンピュータを自動化する方法です。

設定情報の詳細については、Desktop Automation サービス を参照してください。

トリガーを持つロボットを使用するには、Desktop Automation サービスの設定ウィンドウの **[Windows]** タブで **[パッケージをロック]** を選択します。

次のタスクを使用して、トリガーを持つロボットを使用するための開発環境および本番環境を設定します。

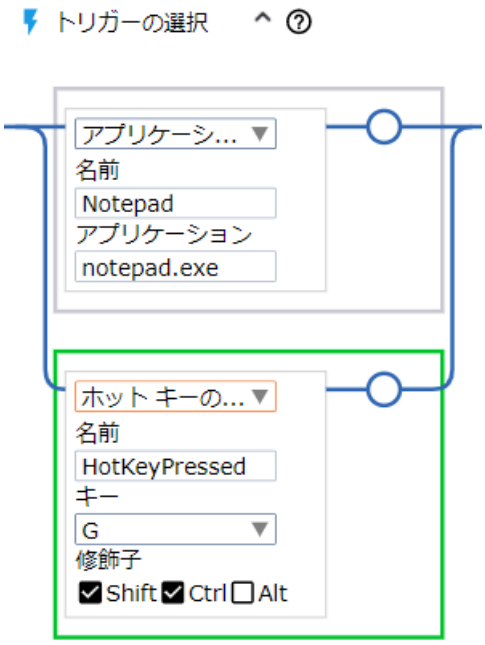
トリガーを持つロボットの開発

1. 既存のベーシック エンジン ロボットを開くか、**[ファイル]** > **[新しい Web オートメーション ロボット]** をクリックして新しいロボットを作成します。
2. [ロボットを呼び出すステップ] を追加して、次の手順を実行します。
   - ロボット プロパティの **[アクション]** ペインで、**[ロボット]** リストの **[新規作成]** を選択します。ロボットの名前を指定し、**[終了]** をクリックします。
   - **[デバイス]** の下に、トリガー デバイスを追加します。選択するデバイスは、Management Console ベースのマッピングではなく、デバイス マッピングである必要があります。

   詳細については、「はじめに」を参照してください。
3. 自動化されたコンピュータで Desktop Automation サービス セットアップ ウィンドウを開き、**[シングル ユーザー]** を選択してトークンを指定します。
4. **[ロボットを呼び出すステップ]** を右クリックし、[タブでロボット '<ロボット名>' を開きます] を選択して、作成したロボットを開きます。

   ロボットが新しいタブで開きます。
5. 開いたロボットに切り替えて、次の手順を実行します。
   a. トリガー チョイス ステップを追加します。
   b. トリガーの種類 (アプリケーションの開始、ホット キーなど) を選択し、名前を付けます。
   c. ステップでアプリケーションまたはホット キーを指定します。

      トリガーの選択ステップでは、複数のタイプのトリガーを使用できます。次の画像で定義されているように、ワークフローは、ユーザーが notepad.exe を開くか、Shift+Ctrl+G キーの組み合わせを押した場合にのみ進行します。



設計中は、トリガーが開始されたときにのみワークフローが進行することに注意する必要があります。トリガーの選択ステップは、Desktop Automation サービスを実行しているコンピュータで、ユーザーが指定されたトリガーを実行するまで待機します。ロボットの設計中にトリガーの選択ステップの進行状況を確認するには、ロボットを実行し、DAS を実行するデスクトップに切り替えて、アプリケーションを開くかホット キーを押すなどして、選択したトリガーを開始します。Design Studio でロボットに切り替え、フロー ポイントがトリガーの選択ステップを通過したことを確認します。最初にワークフローの必要なすべての部分を作成し、ロボットの構築の最後にトリガーの選択を追加して、トリガーの開始を前後に切り替える必要がないようにすることをお勧めします。

トリガー イベントが阻止されると、ユーザーのデスクトップはロボット ワークフローの残りの部分でロックされ、すべてのステップが実行された後にのみ制御が戻ります。単一のロボット ワークフロー内に複数のトリガーがある場合、ワークフローは 2 番目のトリガーの選択で待機しますが、待機中はロボットがデスクトップへのアクセスをブロックするため、待機時間が数秒間以内となるようにしてください。また、アテンデッド オートメーション アクションを通知ステップで囲むことで、ロボットに制御があり、ロボットがデスクトップ上でいくつかのタスクを実行しているという情報をユーザーに示すこともできます。

ロボットのデプロイ

1. ロボット ワークフローの設計が完了した後で、**[プロジェクト]** ビューでロボットを右クリックし、**[アップロード]** をクリックして両方のロボットを Management Console にアップロードします。
2. Management Console を開き、**[リポジトリ]** > **[ロボット]** に移動します。これにより、アップロードされたロボットが、対応するトリガーとともにリポジトリに表示されます。

		フォ...	名前	タイプ	プロジェクト...	タグ	バー...	サ...	スケジ...	入力...	返されるタ...
☐	⋮		email_step	ベーシック...	Default project		11.3.0.0	7.45 KB			mail
☐	⋮		email_step_...	ロボット	Default project		11.3.0.0	53.3 KB			

3. **[リポジトリ]** > **[デバイス マッピング]** に移動し、**[デバイス マッピングを作成]** (左上隅の + 記号) をクリックします。ロボットの構築時に使用したものと同じデバイス マッピング名を指定します。
   詳細については、「デバイス マッピング」を参照してください。
4. **[リポジトリ]** > **[トリガー マッピング]** に移動して、トリガー ユーザー マッピングを作成する場合は **[ユーザー]** タブを選択し、トリガー ラベル マッピングを作成する場合は **[ラベル]** タブを選択します。その後、左上隅の + 記号をクリックして、新しいトリガー マッピングを作成します。このガイドでは、トリガー ユーザー マッピングを作成します。
   詳細については、「トリガー マッピング」を参照してください。
5. アテンデッド オートメーション ロボットとユーザーを選択して、DAS を実行しているデスクトップ上の個々のユーザー アカウントにトリガーを関連付けます。

トリガー マッピングを作成すると、実行中の Desktop Automation サービスはマッピングされたユーザーによって実行されたトリガーの監視を開始し、阻止された場合、関連するロボットを開始します。

Management Console でロボットを実行する前に、Desktop Automation サービスの設定でシングル ユーザー モードをオフにし、再起動して変更を有効にします。これにより、Desktop Automation サービスが Management Console と通信し、デスクトップ上のトリガーを監視できるようになります。

ドキュメントへの署名

ロボット ワークフローの「PDF」ステップには、署名付きドキュメントを作成する SignDoc が統合されています。

必要条件

- SignDoc 標準。サポートされているソフトウェア バージョンについては、『Kofax RPA 技術仕様』ドキュメントを参照してください。
- SignDoc ユーザー アカウントおよび対応する API キー。

ワークフロー

SignDoc を使用するロボット ワークフローは、次のようになります。

1. PDF ドキュメントを開く (「PDF」ステップ)。
2. SignDoc 署名セッション ([**SignDoc** で署名] または [**SignDoc** で署名 **(テンプレート)**] アプリケーション アクション) を開始します。
3. ドキュメントの処理:
   - PDF ドキュメントを参照して、署名フィールドを追加します ([**SignDoc** 署名フィールドを挿入] アクション)。

- SignDoc ([**SignDoc** 署名者の割り当て] および [フィールドを更新する] コンポーネント アクション) で処理するために PDF ドキュメントのフィールドを操作します。

4. 制御を SignDoc に転送します ([**SignDoc** リクエストを完了] アプリケーション アクション)。

ロボットは開いている PDF ドキュメントごとに SignDoc セッションを 1 つ開くことができます。複数の PDF ドキュメントが署名のために開かれている場合は、複数の SignDoc セッションが終了することがあります。

アクション

SignDoc で署名

このステップは、SignDoc セッションを作成し、PDF ドキュメントをすぐに送信します。このステップの前にロボットによって PDF ファイルが変更された場合、ドキュメントは保存されます。それ以外の場合、元のドキュメントは SignDoc に送信されます。

プロパティ	説明
SignDoc サーバーの URL	SignDoc サーバーの URL (プロトコル + ホスト + ポート)
API キー	サーバーにアクセスするための API キー。API キーに関連付けられているユーザー アカウントが署名パッケージの所有者になります。
SignDoc パッケージ ID (オプション)	このパッケージに割り当てられた SignDoc のパッケージ ID。省略した場合、SignDoc によって一意の ID が割り当てられます。
名前	署名者/レビュー担当者に表示される SignDoc パッケージの名前。
説明	署名者/レビュー担当者に表示される SignDoc パッケージの説明。
件名	署名者/レビュー担当者に送信される通知メールに使用される件名。
メッセージ	署名者/レビュー担当者に送信される通知メールに使用される本文。
ドキュメントの説明	ドキュメントの説明。これは、SignDoc の署名者/レビュー担当者に表示されます。
ドキュメント ファイル名	ドキュメントの名前。これは、署名者/レビュー担当者がドキュメントをダウンロードするときに表示されます。
レビュー担当者 (オプション)	レビュー担当者のリスト。詳細については、レビュー担当者 を参照してください。
署名順序を適用	ドキュメントを順番に処理する必要があることを示します。このオプションが選択されていない場合は、すべての署名者とレビュー担当者が同時に処理を実行できます。詳細については、順序 を参照してください。
ドキュメントを閉じるときにパッケージを保持	ロボットが PDF ドキュメントを (手動、またはエラーによって) 閉じた場合、あるいは終了した場合に、署名パッケージを SignDoc サーバーに残しておく必要があることを示します。このオプションが選択されていない場合は、ロボットの終了時にパッケージが削除されます。
有効期限 (日数) (オプション)	署名パッケージの有効期限 (日数)。
フォーム フィールドをロック	PDF ドキュメントの既存のすべてのフィールドに読み取り専用のマークを付ける必要があることを示します。これは、SignDoc 内の署名セッション中のドキュメントにのみ影響します。「フィールドの更新」アクションを使用して、フィールドごとに読み取り専用属性を変更することができます。

プロパティ	説明
存在する場合にオーバーライド	セッションの開始時に既存のパッケージを破棄する必要があることを示します。この設定は、[SignDoc パッケージ ID] オプションを使用している場合のみ有効です。

レビュー担当者

プロパティ	説明
名前	レビュー担当者の名前。
電子メール	レビュー担当者の電子メール アドレス。
表示言語 (オプション)	SignDoc がレビュー担当者と通信するために使用する構成済み言語をオーバーライドします。サポートされている BCP 47 言語コードの中の 1 つを指定してください。詳細については、表示言語 を参照してください。
電子署名への同意が必要	レビュー担当者が電子署名への同意ページに同意する必要があることを示します。
GDPR への同意が必要	レビュー担当者が GDPR への同意ページに同意する必要があることを示します。
順序	レビュー担当者の相対的な順序。詳細については、順序 を参照してください。

SignDoc で署名 (テンプレート)

SignDoc テンプレートに基づいて SignDoc セッションを作成し、すぐに PDF ドキュメントを送信します。このステップの前にロボットによって PDF ファイルが変更された場合、ドキュメントは保存されます。それ以外の場合、元のドキュメントは SignDoc に送信されます。

プロパティ	説明
SignDoc サーバーの URL	SignDoc サーバーの URL (プロトコル + ホスト + ポート)
API キー	サーバーにアクセスするための API キー。API キーに関連付けられているユーザー アカウントが署名パッケージの所有者になります。
SignDoc パッケージ ID (オプション)	このパッケージに割り当てられた SignDoc のパッケージ ID。省略した場合、SignDoc によって一意の ID が割り当てられます。
テンプレート	パッケージのベースとなる SignDoc テンプレートの名前または ID。
名前 (オプション)	署名者/レビュー担当者に表示される SignDoc パッケージの名前。
説明 (オプション)	署名者/レビュー担当者に表示される SignDoc パッケージの説明。
件名 (オプション)	署名者/レビュー担当者に送信される通知メールに使用される件名。
メッセージ (オプション)	署名者/レビュー担当者に送信される通知メールに使用される本文。
ドキュメントの説明	ドキュメントの説明。これは、SignDoc の署名者/レビュー担当者に表示されます。
ドキュメント ファイル名	ドキュメントの名前。これは、署名者/レビュー担当者がドキュメントをダウンロードするときに表示されます。
レビュー担当者 (オプション)	レビュー担当者のリスト。詳細については、レビュー担当者 を参照してください。
署名順序を適用	ドキュメントを順番に処理する必要があることを示します。このオプションが選択されていない場合は、すべての署名者とレビュー担当者が同時に処理を実行できます。詳細については、順序 を参照してください。

プロパティ	説明
ドキュメントを閉じるときにパッケージを保持	ロボットが PDF ドキュメントを (手動、またはエラーによって) 閉じた場合、あるいは終了した場合に、署名パッケージを SignDoc サーバーに残しておく必要があることを示します。このオプションが選択されていない場合は、ロボットの終了時にパッケージが削除されます。
有効期限 (日数) (オプション)	署名パッケージの有効期限 (日数)。
フォーム フィールドをロック	PDF ドキュメントの既存のすべてのフィールドに読み取り専用のマークを付ける必要があることを示します。これは、SignDoc 内の署名セッション中のドキュメントにのみ影響します。「フィールドの更新」アクションを使用して、フィールドごとに読み取り専用属性を変更することができます。
存在する場合にオーバーライド	セッションの開始時に既存のパッケージを破棄する必要があることを示します。この設定は、[SignDoc パッケージ ID] オプションを使用している場合のみ有効です。

オプション パラメータの [名前]、[説明]、[件名]、および [メッセージ] を使用して、テンプレートの設定を上書きできます。

ロボットは、テンプレートから作成されたパッケージに PDF ドキュメントを挿入します。以前に SignDoc ステップで作成されたドキュメントに基づくテンプレートを使用している場合、ロボットはそのテンプレートから作成されたドキュメントを置き換えます。この場合、ドキュメントのプロパティはすべて保持されます。

SignDoc 署名フィールドを挿入 (アプリケーション アクション)

PDF ドキュメントの現在のページに署名フィールドを挿入します。これらのフィールドはページには表示されませんが、アプリケーション ツリーには SignDoc フィールドとして表示されます。

プロパティ	説明
名前	署名フィールドの名前。
説明	署名フィールドの説明。この値は、署名者がフィールドに署名するときにツールチップとして表示されます。
ラベル (オプション)	署名フィールドのラベル。SignDoc のクライアント インターフェイスには説明が表示され、管理インターフェイスには ID が使用されます。この値は PDF ビューアに表示されます。
署名者の名前	署名者の名前。
署名者の電子メール	署名者の電子メール アドレス。
表示言語 (オプション)	SignDoc が署名者と通信するために使用する設定済みの言語を上書きします。サポートされている BCP 47 言語コードの中の 1 つを指定してください。詳細については、表示言語 を参照してください。
電子署名への同意が必要	署名者が電子署名への同意ページに同意する必要があることを示します。
GDPR への同意が必要	署名者が GDPR への同意ページに同意する必要があることを示します。
署名者の順序	署名者の相対的な順序。詳細については、順序 を参照してください。
許可された署名モード (オプション)	フィールドで許可される署名モードを制限します。どの署名モードも選択されていない場合、SignDoc は設定済みのデフォルト署名モードを使用します。
ユニット	署名用のボックスを作成するための X、Y、幅、および高さの単位を指定します。詳細については、寸法 を参照してください。

プロパティ	説明
X Y	署名フィールドの左上隅の位置を設定します。
幅 高さ	署名フィールドの寸法を設定します。
署名が必要	署名が必要であることを示します。
SignDoc ID (オプション)	署名フィールドの ID。省略した場合、SignDoc によって一意の ID が割り当てられます。
補足ドキュメント	署名者に補足ドキュメントを提供するよう求めます。詳細については、補足ドキュメントを参照してください。

署名者が複数回追加された場合は、最初の出現箇所以降の [表示言語]、[電子署名への同意が必要]、[GDPR への同意が必要]、および [署名者の順序] フィールドは無視されます。

補足ドキュメントに、同じ署名者、同じドキュメント タイプに対するリクエストが複数含まれている場合 (補足ドキュメント)、これらのリクエストは SignDoc パッケージ内で結合されます。いずれかの出現箇所が必須に設定されている場合、このリクエストは必須に設定されます。ファイルの最大数は、すべての出現箇所の最大数に設定されます。

補足ドキュメント

プロパティ	説明
ドキュメント タイプ	リクエストされたドキュメントのタイプ。
必須	署名者がドキュメントをアップロードする必要があることを示します。
名前 (オプション)	SignDoc に示されているドキュメント タイプの名前。その他のタイプ (SignDoc では汎用) を除くすべてのドキュメント タイプで、このパラメータは無視されます。
説明 (オプション)	リクエストの説明 (手順を含む)。その他のタイプ (SignDoc では汎用) を除くすべてのドキュメント タイプで、このパラメータは無視されます。
ファイルの最大数 (オプション)	アップロード可能なドキュメントの最大数。その他のタイプ (SignDoc では汎用) を除くすべてのドキュメント タイプで、このパラメータは無視されます。

使用可能なドキュメント タイプのリストは、SignDoc サーバーから取得されます。Design Studio でローカライズされているのは、SignDoc の事前定義されたドキュメント タイプの名前のみです。

SignDoc 署名フィールドを挿入 (コンポーネント アクション)

このアクションにより、コンポーネント ファインダーに基づいて署名フィールドを挿入します。これらのフィールドはページには表示されませんが、アプリケーション ツリーには SignDoc フィールドとして表示されます。

署名フィールドは、コンポーネント ファインダーによって配置された領域に表示されますが、位置はパラメータで調整できます。

プロパティ	説明
名前	署名フィールドの名前。
説明	署名フィールドの説明。この値は、署名者がフィールドに署名するときにツールチップとして表示されます。

プロパティ	説明
ラベル (オプション)	署名フィールドのラベル。このフィールドが設定されていない場合、SignDoc のクライアント インターフェイスには説明が表示され、管理インターフェイスには ID が使用されます。この値は PDF ビューアに表示されます。
署名者の名前	署名者の名前。
署名者の電子メール	署名者の電子メール アドレス。
表示言語 (オプション)	SignDoc が署名者と通信するために使用する設定済みの言語を上書きします。サポートされている BCP 47 言語コードの中の 1 つを指定する必要があります。詳細については、表示言語 を参照してください。
電子署名への同意が必要	署名者が電子署名への同意ページに同意する必要があることを示します。
GDPR への同意が必要	署名者が GDPR への同意ページに同意する必要があることを示します。
署名者の順序	署名者の相対的な順序。詳細については、順序 を参照してください。
許可された署名モード (オプション)	フィールドで許可される署名モードを制限します。どの署名モードも選択されていない場合、SignDoc は設定済みのデフォルト署名モードを使用します。
水平オフセット (オプション)	コンポーネントを基準にして、署名フィールドを左または右にシフトするスペースの量。
垂直オフセット (オプション)	コンポーネントを基準にして、署名フィールドを上下にシフトするスペースの量。
目的の幅 (オプション)	署名フィールドの幅を上書きします。この値が指定されていない場合、フィールドの幅はコンポーネントの幅になります。
目的の高さ (オプション)	署名フィールドの高さを設定します。この値が指定されていない場合、フィールドの高さはコンポーネントの高さになります。
最小の幅 (オプション)	署名フィールドの最小幅を指定します。フィールドの幅がこれよりも小さい場合、フィールドは右に拡張されます。[目的の幅] が設定されている場合、このパラメータは無視されます。
最小の高さ (オプション)	署名フィールドの最小の高さを指定します。フィールドの高さがこれよりも小さい場合、フィールドは下に拡張されます。[目的の高さ] が設定されている場合、このパラメータは無視されます。
署名が必要	署名が必要であることを示します。
SignDoc ID (オプション)	署名フィールドの ID。省略した場合、SignDoc によって一意の ID が割り当てられます。
補足ドキュメント	署名者に補足ドキュメントを提供するよう求めます。詳細については、補足ドキュメントを参照してください。

署名者が複数回追加された場合は、最初の出現箇所以降の [表示言語]、[電子署名への同意が必要]、[GDPR への同意が必要]、および [署名者の順序] フィールドは無視されます。

すべての調整は、ツリー座標で指定する必要があります。PDF ドライバーは、ツリー座標の基礎として 144 DPI 解像度を使用します。

補足ドキュメントに、同じ署名者、同じドキュメント タイプに対するリクエストが複数含まれている場合 (補足ドキュメント)、これらのリクエストは SignDoc パッケージ内で結合されます。いずれかの出現箇所が必須に設定されている場合、このリクエストは必須に設定されます。ファイルの最大数は、すべての出現箇所の最大数に設定されます。

SignDoc 署名者の割り当て (コンポーネント アクション)

PDF ファイルに既存の署名フィールドに署名者を割り当てます。これらのフィールドはページには表示されませんが、アプリケーション ツリーには SignDoc フィールドとして表示されます。

このステップは、SignDoc フィールドと FormBox フィールドで使用できます。このステップにより、SignDoc で署名者が作成され、設定されます。既存の署名者を割り当てるには、「フィールドの更新」アクションを使用します。

プロパティ	説明
署名者の名前	署名者の名前。
署名者の電子メール	署名者の電子メール アドレス。
表示言語 (オプション)	SignDoc が署名者と通信するために使用する設定済み言語を上書きします。サポートされている BCP 47 言語コードの中の 1 つを指定する必要があります。詳細については、表示言語 を参照してください。
電子署名への同意が必要	署名者が電子署名への同意ページに同意する必要があることを示します。
GDPR への同意が必要	署名者が GDPR への同意ページに同意する必要があることを示します。
署名者の順序	署名者の相対的な順序。詳細については、順序 を参照してください。
許可された署名モード (オプション)	フィールドで許可される署名モードを制限します。どの署名モードも選択されていない場合、SignDoc は設定済みのデフォルト署名モードを使用します。
署名が必要	署名が必要であることを示します。
補足ドキュメント	署名者に補足ドキュメントを提供するよう求めます。詳細については、補足ドキュメントを参照してください。

署名者が複数回追加された場合は、最初の出現箇所以降の [表示言語]、[電子署名への同意が必要]、[GDPR への同意が必要]、および [署名者の順序] フィールドは無視されます。

補足ドキュメントに、同じ署名者、同じドキュメント タイプに対するリクエストが複数含まれている場合 (補足ドキュメント)、これらのリクエストは SignDoc パッケージ内で結合されます。いずれかの出現箇所が必須に設定されている場合、このリクエストは必須に設定されます。ファイルの最大数は、すべての出現箇所の最大数に設定されます。

コンポーネントが SignatureText タイプの SignDoc フィールドである場合、このフィールドは更新されます。それ以外の場合、ステップはページ上で、まったく同じ寸法を持つ、このようなフィールドを見つけようとします。このフィールドが見つからない場合、ステップは失敗します。

使用可能なドキュメント タイプのリストは、SignDoc サーバーから取得されます。Design Studio でローカライズされているのは、SignDoc の事前定義されたドキュメント タイプの名前のみです。

フィールドを更新する (コンポーネント アクション)

このステップは、サポート対象であると SignDoc で識別された PDF ドキュメント内のフォーム フィールドの属性を更新します。サポートされているフィールドは、アプリケーション ツリーの SignDoc ノードの下に表示されます。

プロパティ	説明
必須 (オプション)	必須属性を変更します。
読み取り専用 (オプション)	読み取り専用属性を変更します。
チェック済み (オプション)	チェック ボックスの状態を設定します。他のタイプのフィールドでは無視されます。

プロパティ	説明
説明 (オプション)	フィールドの説明を変更します。この値は、署名者がフィールドを操作するときにツールチップとして表示されます。
ラベル (オプション)	フィールドのラベルを変更します。この値は PDF ビューアに表示されます。
値 (オプション)	テキスト フィールドの値を設定します。他のタイプのフィールドでは無視されます。
割り当て先 (オプション)	電子メール アドレスまたは名前で識別される署名者にフィールドを割り当てます。署名者は、セッション内ですでに認識されている必要があります。既存の割り当てを削除するには、特別な値「Any」を使用します。

フィールドの変更は SignDoc 署名セッション中に行われます。変更は PDF ドキュメント自体には適用されず、アプリケーション ツリーの外部には表示されません。

SignDoc リクエストを完了

「SignDoc リクエストを完了」ステップは SignDoc セッションを閉じて、SignDoc パッケージの処理方法を判別します。

プロパティ	説明
リクエストを閉じる	セッションは閉じますが、SignDoc サーバーにはドラフト モードのパッケージが残ります。このオプションは、[**SignDoc** で署名] ステップと [**SignDoc** で署名 **(**テンプレート**)**] ステップの [ドキュメントを閉じるときにパッケージを保持] オプションを上書きします。
スケジュール リクエスト	セッションを閉じて署名手順を開始するように、SignDoc パッケージにスケジュールを設定します。
リクエストのキャンセル	セッションを閉じて、SignDoc サーバーから SignDoc パッケージを削除します。これ以上のアクションは実行されません。

SignDoc プロパティを取得

このステップは、SignDoc セッションのプロパティを照会します。

プロパティ	説明
プロパティ	プロパティの名前
デフォルト値 (オプション)	失敗時に返される値。この値が指定されておらず、プロパティを取得できない場合、エラーが生成されます。
値	結果の変数

サポートされているプロパティ

- [セッション中]: PDF ドキュメントにアクティブな SignDoc セッションがすでに存在する場合は、「Y」を返します。それ以外の場合は「N」を返します。
- [パッケージ **ID**]: SignDoc のパッケージ ID を返します。
- [パッケージの **URL**]: REST API コールで使用する SignDoc パッケージの URL を返します。

パッケージ URL には、SignDoc で構成されたホスト名が含まれています。このホスト名は、ロボットが接続に使用するホスト名と一致しない場合があります。

アクティブな SignDoc セッションがない場合は、[セッション中] を除くすべてのプロパティがエラーを返します。

表示言語

次に、SignDoc が [表示言語] フィールドに受け入れる値を示します。

BCP 47 コード	言語
de	ドイツ語
en	英語
es	スペイン語
fr	フランス語
it	イタリア語
nl	オランダ語
pt-BR	ポルトガル語 (ブラジル)
ja	日本語

このリストは今後の SignDoc バージョンで拡張される可能性があります。

順序

[**SignDoc** で署名] で [署名順序を適用] オプションが選択されている場合、または [**SignDoc** で署名 **(**テンプレート**)**] ステップで指定されたテンプレートが順次署名するように設定されている場合、レビュー担当者と署名者が PDF ドキュメントに署名するよう順次招待されます。それぞれの参加者は、前の参加者が処理したドキュメントを受け取ります。署名者とレビュー担当者が処理される順序は、[順序] フィールドに基づきます。つまり、順序値が最も低い署名者またはレビュー担当者が最初に処理されます。複数のユーザーの順序値が同じである場合は、ロボットがパッケージに追加した順序で処理されます。テンプレート内のレビュー担当者と署名者は、ロボットによって追加されたレビュー担当者と署名者の前に処理されます。

[署名順序を適用] オプションが選択されていない場合、レビュー担当者と署名者は同時に招待されます。これらのすべての参加者に PDF ドキュメントが元の状態で送信されます。

寸法

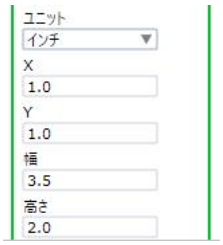
署名フィールドの場所と寸法は、ロボットが指定します。計算には以下の単位を使用できます。

- [ツリー座標]: アプリケーション ツリーで使用され、Design Studio に表示される座標。
- **[インチ]** または **[センチメートル]**: PDF ドキュメントの物理的な寸法。

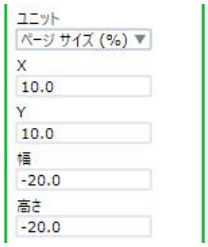
- [ページ サイズ **(%)**]: PDF ドキュメントの寸法を基準とする相対的なサイズ。

値は、ページの左上隅を基準にして計算されます。フィールドに負の値が含まれている場合は、ページに適した寸法を基準にして計算されます。[幅] または [高さ] が 0 に設定されている場合は、代わりにページの寸法が使用されます。

例
次の例では、ロボットは左上隅から下および右に 1 インチの位置にある、3.5 インチ x 2 インチの領域を選択します。



次の例では、ロボットはすべての辺の 10% マージンを除いた、ページ全体を選択します。



TLS コミュニケーションを使用

Kofax RPA は、オートメーション デバイスと RoboServer または Design Studio の間に TLS 通信を設定する手段を提供します。通信には、通信を暗号化するための証明書が使用されます。暗号化では、接続の保護にはパブリック - プライベート キー構造が使用されます。

TLS バージョン 1.1 または 1.2 が必要な場合は、.NET インストールがバージョン 4.7.2 以降に更新されていることを確認してください。

パスワードで保護された証明書はサポートされません。

キー ファイルは、"pem" 形式にする必要があります。これは、openSSL では最も一般的な形式です。

Desktop Automation サービスのインストール パッケージには、4 つのファイルを含む `certificates` フォルダと、`serverCa` フォルダが含まれています。

`ca.cert.pem` は Kofax RPA が作成した秘密鍵で署名された公開鍵ファイルです。キーのトラスト チェーンのルート証明書として機能します。`kofax.das.ca.cert.pem` は、ルート 秘密鍵で署名され

た別の署名付き証明書です。この 2 つのファイルは serverCa フォルダに存在します。特定のセキュリティ要件がない場合は、これらのファイルはそのまま使用できます。

kofax.local.das.pem は、RoboServer と Design Studio に存在するローカル ハブによって使用される秘密鍵ファイルです。kofax.local.das.cert.pem は、Kofax.das.ca.cert.pem に対して基盤となる秘密鍵で署名されたパブリック キーです。

kofax.remote.das.pem は、Desktop Automation サービスが使用する秘密鍵ファイルです。kofax.remote.das.cert.pem は、kofax.das.ca.cert.pem に対して基盤となる秘密鍵で署名されたパブリック キーです。

これらのファイルでは、オートメーション デバイスと RoboServer または Design Studio のコードは同じです。

オートメーション デバイスには、ca.cert ファイルを含む serverCa フォルダとともに、kofax.remote.* ファイルが含まれている必要があります。

RoboServer または Design Studio には、kofax.local.* ファイルと ca.cert ファイルを含んだ ca フォルダが含まれている必要があります。

独自の証明書を使用する場合、信頼できる認証局の証明書 (署名済みパブリック キー ファイル) が ca フォルダ (オートメーション デバイスの serverCa) に含まれている必要があります。Node.js は、オートメーション デバイスからの証明書の信頼性をチェックします。ca/serverCa フォルダの証明書に対して署名済み証明書のチェーンがある場合、証明書は信頼されます。オートメーション デバイスからの証明書が信頼されている場合は、同じ方法で Desktop Automation サービスは RoboServer または Design Studio からの証明書を検証します。

RoboServer または Design Studio には、信頼できる証明書がのみが必要です。複数のオートメーション デバイスから同じ証明書を使用することができるため、オートメーション デバイス名は証明書の "common name" と一致する必要はありません。

証明書を自分の有効な証明書または独自の証明書に変更するには、2 つの方法があります。

- 推奨する方法
  1. 新しいサーバー証明書を取得し、秘密鍵とパブリック キーをデバイス上のフォルダにコピーしてオートメーション デバイスにインストールします。
  2. **[Desktop Automation** サービス**]** ウィンドウの **[**証明書**]** タブを使用して、証明書のパスを新しいパスに変更します。
     詳細については、「[Desktop Automation] タブ」と「Desktop Automation サービス」を参照してください。
  3. Design Studio の新しいクライアント証明書を取得し、Design Studio を実行しているコンピュータにインストールします。Design Studio 設定ウィンドウの [**Desktop Automation**] タブを開き、クライアント証明書へのパスを指定します。
  4. RoboServer の新しいクライアント証明書を取得してインストールします。RoboServer ダイアログ ボックスの [セキュリティ] タブで、新しい証明書へのパスを指定します。
- 別の方法
  - カスタム証明書の名前を、kofax.local.das.pem、kofax.local.das.cert.pem など、適切な Kofax RPA の名前に変更します。元の証明書を新しい証明書で上書きします。

ロボットでの日付と時刻の処理

一般的な形式から日付と時刻を抽出して、統一された日付と時刻の形式にすることができます。また、Date、Time、および DateTime 変数を作成して割り当てたり、日付と時刻を比較および変換したりすることもできます。

このトピックでは、機能に関する一般的な情報、およびロボットで日付と時刻を処理するための詳細な説明を含むこのヘルプの他のトピックへのリンクを提供します。

日付と時刻を操作するためには、次のような機能を使用できます。

- タイプ: Date、Time、および DateTime
- コンバータ ステップ: DateTime を抽出、DateTime の書式設定
- Date、Time、および DateTime のタイプに基づく関数

タイプ

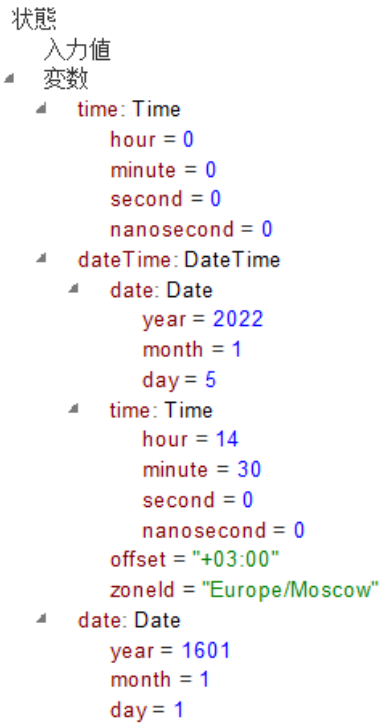
Date タイプは、時刻またはタイム ゾーンを参照せずに日付を表すタイプです。Date タイプの変数のデフォルト値は、1601-01-01 です。

Time タイプは、日付とタイム ゾーンを参照せずに時刻を表すタイプです。Time タイプの変数のデフォルト値は、00:00:00 です。

DateTime タイプは、ISO-8601 カレンダー システムのタイム ゾーンで日付と時刻を表すタイプで、これは基本的にタイム ゾーンで日付と時刻の値です。DateTime 変数のデフォルト値は、その Date および Time コンポーネントのデフォルトのタイプで構成され、デフォルト タイム ゾーンはロボットが実行される場所に対応します。

これらのタイプは、ベーシック エンジン ロボットのシンプル タイプの Date に対応します。これらのタイプを使用するには、最初に Date 型の属性を持つコンプレックス タイプを作成する必要があります。次に、**[**レコード タイプのフィールド タイプ**]** を Date、Time、または DateTime (デフォルト) のいずれかに指定する必要があります。追加の手順については、「Date、Time、DateTime を含むレコード タイプの変数」を参照してください。また、ロボット エディターで Date、Time、または DateTime タイプの変数を直接追加して、ロボット内で使用することもできますが、この場合、変数の結果値はベーシック エンジン ロボットに転送されません。

次に示すように、Date、Time、および DateTime の値は、 [状態] ペインの [変数] セクションに反映されます。ロボットの編集中に値をダブルクリックして新しい値を入力することで、[状態] ペインで変数の値を一時的に変更できます。他の値と同様に、`year`、`hour`、`offset` など、Date、Time、および DateTime の値の個々の部分を変更できます。



コンバータ ステップ

次の 2 つのコンバータ ステップを使用して、ロボットで日付と時刻を処理することができます。

- DateTime を抽出ステップは、テキスト表現を Date、Time、または DateTime 値に変換します。
- DateTime の書式設定ステップは、Date、Time、または DateTime 値を Text 値に変換します。

関数

Ctrl キーを押しながらスペース キーを押すと、エクスプレッション エディターで使用可能な関数の完全なリストを直接取得できます。使用可能な関数の詳細については、「エクスプレッション」のトピックを参照してください。

Date、Time、DateTime の各タイプには、コンポーネントへのアクセスに役立つ関数のセットがいくつか用意されています。

- **Date**: `year`、`month`、および `day`。
- **Time**: `hour`、`minute`、`second`、および `nanosecond`。
- **DateTime**: `date`、`time`、`offset`、および `zoneID`。

これらのコンポーネントはフィールド値ではないため、エクスプレッションのフィールドとしてアクセスすることや割り当てステップのフィールドで設定することができないことに注意してください。

他の関数を使用することで、それぞれのタイプの値を作成し、そのタイプの値をテキストに変換できます。たとえば、`with` 関数を使用して DateTime コンポーネントの値を変更したり、`plus` および `minus`

関数を使用して年、月、または日数を加算または減算したり、`text` 関数を使用して値をテキスト表現に変換したりすることなどができます。

例

- `year(Date)` という関数は、現在の日付の年を返します。
  `year(date(2022, 3, 4))` は year = 2022 を返します。
- `hour(Time)` という関数は、現在の時刻の時間を返します。
  `hour(time(14, 50, 45))` は hour = 14 を返します。
- `withDay(Date, Integer)` という関数は、日付が変更された日付のコピーを返します。
  `date(2022, 3, 4).withDay(10)` は、year = 2022、month = 3、**day = 10** を返します。
- `plusDays(Date, Integer)` という関数は、指定された日数が追加された日付のコピーを返します。
  `date(2022, 3, 4).plusDays(70)` は、year = 2022、**month = 5**、**day = 13** を返します。
- `text(DateTime, Text, Text)` という関数は、指定されたパターンと口ケール (ハイフンで区切られた言語コードと国コードの組み合わせ) に一致する DateTime の日付と時刻のテキスト表現を返します。
  DateTime が 2021-12-20T00:30:00+03:00 である場合、`text("EEEE MMM d yy HH:mm", "en-US")` は "Monday Dec 20 21 00:30" を返します

タイム ゾーンの操作

一連の関数を使用して、タイム ゾーンを操作することができます。たとえば、現在のタイム ゾーンの取得やタイム ゾーンの変更ができます。次の 2 つのプロパティがタイム ゾーンに関連しています。`zoneID` および `Offset`。

zoneID

`zoneID` は、"Europe/Paris" (ヨーロッパ/パリ) や "MET" などのタイム ゾーンを識別するために使用されます。このプロパティは、夏時間を考慮する必要がある場合、または、あるタイム ゾーンから別のタイム ゾーンに DateTime 値を変換する必要がある場合に役立ちます。ゾーン ID とオフセットは相互に関連しており、一方を変更すると他方も変更される可能性があります。

オフセット
`Offset` は、+01:00 や -05:00 など、グリニッジ/UTC からのタイム ゾーン オフセットを識別するために使用されます。`Offset` の形式は次のとおりです。

- `Z` - UTC の場合
- `+h`
- `+hh`
- `+hh:mm`
- `-hh:mm`
- `+hhmm`
- `-hhmm`
- `+hh:mm:ss`
- `-hh:mm:ss`
- `+hhmmss`
- `-hhmmss`

`Offset` を設定すると `zoneID` が UTC に変更され、オフセットが 0 または Z でない場合は、その後にオフセットが表示されます。たとえば、オフセットを +00 に設定すると UTC というゾーン ID となり、オフセットを +05 に設定すると UTC +05:00 というゾーン ID になります。

また、DateTime 値にオフセットまたはゾーン ID を設定すると、新しいゾーンと同じインスタンスを表す DateTime 値になります。つまり、日付と時刻が変更される可能性があります。これは、関数 `withzoneID` および `withOffset` を使用して設定できます。

日付と時刻を変更せずにタイム ゾーンを変更するには、次のような新しい DateTime 値を作成します。

```
dateTime(dateTime1.date(), dateTime1.time(), "Europe/Paris")
```

[状態] ビューからオフセットまたはゾーン ID を設定するとゾーン ID またはオフセットがそれぞれ変更されるため、相互に整合性が保たれますが日付と時刻は変更されません。

データの変換

ロボットの変換機能を使用することで、データ変換とロボット開発が容易になります。この機能を使用すると、手動で入力したデータを自動的に変換し、抽出ステップとともに、抽出したデータを変換してから結果を変数に保存できます。

次のステップのグループは「コンバータ グループ」と総称されます。

- 値の変換
- 値を抽出
- クリップボードから抽出
- ツリーを XML として抽出

次のコンバータ ステップは、前のステップ内でのみ使用できます。

- DateTime を抽出
- DateTime の書式設定

これらのステップを使用して、データを抽出、変換およびフォーマットできます。これらのステップには、エクスプレッションの評価ステップを追加できます。また、現在のインを保存ステップは、変換された値を変数に格納するためにコンバータ ステップとともに使用します。

ロボットの他のステップとは対照的に 、コンバータ グループ内のワークフロー状態への変更を元に戻す必要がある場合、または特定の先行フロー ポイントの状態を確認する必要がある場合は、必要な先行フロー ポイントを実行して、ステップ内に戻ることができます。また、現在のフロー ポイントの前のワークフローを変更すると、ワークフローの状態が自動的に調整されるため、ステップを再実行する必要はありません。

変換と抽出は、次の組み込み変数を使用して実行されます。

- **$current** 変数には初期値が含まれています。この変数の値は実行中に変更されます。数値をテキスト値に変換するなど、値のタイプも変更できます。
- **$Initial** 変数にも初期値が含まれていますが、値は実行中に変更されないため、再変換など、必要に応じて初期値を再利用できます。

エクスプレッション

このセクションでは、ロボットのエクスプレッションおよびその編集方法や評価方法について説明します。

ロボット ステップのプロパティの多くは、プレーン値 (数値など) またはエクスプレッションとして指定できます。エクスプレッションは評価され、この評価の結果は、プレーン値がプロパティの値として直接利用されるプロパティに対して用いられます。たとえば、クリック ステップのカウント プロパティには 2 などの数値を指定できますが、`clickCount` などのエクスプレッションとして指定することもできます。`clickCount` はステップの範囲内で定義された変数であり、ロボットの他の場所で値が与えられます。

ロボットのエクスプレッションは、Java、C#、JavaScript などの最も一般的なプログラミング言語の式と非常によく似ています。これらは、定数、変数、演算 (加算、減算、乗算、比較演算、論理演算など)、および関数で構成されています。エクスプレッションの例を以下に示します :

- `(1 + 2)*3`
- `x > 0 || x <= 6`
- `max(x, 10)`
- `"hello".substring(3)`

エクスプレッションにはタイプが追加され、これらのタイプは変数と同じものとなります。つまり、演算と関数は特定のタイプのオペランドにしか適用されないことを意味します。オペランドの型によっては、評価されるときに特定の型の値を返します。たとえば、整数タイプの 2 つのオペランドを加えると、`1 + 2` のように整数タイプの結果が得られ、`3` と評価されます。オペランドのタイプが数値の場合は結果が数値タイプとなり、たとえば `1.0+2.0` は、`3.0` と評価されます。エクスプレッションが評価され、エクスプレッションのタイプ エラーがロボットでエラーとして報告される前に、エクスプレッションのタイプの静的チェックが行われます。

エクスプレッションの評価によってワークフローの状態が変わることはありません。つまり、ユーザーはエクスプレッションの中の変数に値を割り当てることはできません。変数に値を割り当てることができるのはステップだけです。たとえば、変数割り当て ステップによって変数に値が割り当てられ、この値はエクスプレッションの評価から得られたものとなります。

次のセクションでは、エクスプレッションのさまざまなコンポーネントと式エディターについて説明します。

- 定数
- 変数
- 演算
- 関数
- 数値の制限
- エクスプレッション エディター

定数

定数は以下の簡単なタイプの値になります。

タイプ	例
整数	42 -17
数値	3.14159 -.33
ブール値	TRUE FALSE
テキスト	"Hello" "First"

テキスト値が終了してしまうため、テキスト値に二重引用符 (") を含めることはできません。代わりに、テキスト値に二重引用符が必要な場合には \" を使用します。バックスラッシュ記号 (\) は通常、エクスプレッションで直接記述できないテキスト値の特殊文字に用います。特殊文字：

文字	説明
`\n`	改行 組み込みブラウザで改行を使用するには、以下を使用します。 `\r\n`
`\r`	キャリッジ リターン
`\f`	フォーム フィード
`\"`	二重引用符
`\t`	タブ
`\b`	バックスペース
`\\`	バックスラッシュ文字自体
`\uXXXX`	コード化された 16 進数の Unicode 文字。たとえば「`\u002A`」は、「`*`」の別の表記です。

変数

エクスプレッションの変数は、エクスプレッションのロケーションでスコープ内のワークフローに定義された任意の変数または入力パラメータとなります。入力パラメータのスコープはワークフロー全体となるため、そのパラメータは常にスコープ内となります。ワークフローの最上位レベルで定義されている場合、または グループ ステップ 内にある場合、変数はスコープ内となります。

演算

演算は、演算子と複数のオペランドから構成されるエクスプレッションです。`1+2` というエクスプレッションでは、`+` が演算子で、`1` と `2` がオペランドです。したがって、演算子は、エクスプレッションが評価されるときにオペランドの値に対して実行される演算を定義します。このセクションでは、エクスプレッションで発生する演算子について説明します。ほとんどの場合、これらの演算子が実行する演算

は複雑ではありません。プログラミング言語のエクスプレッションに詳しい場合は、この説明を省略して、以下のサマリーテーブルを参照してください。

算術演算
エクスプレッションは、加算 (+)、減算 (-)、乗算 (*)、除算 (/)、およびモジュロ (%) に対応しています。各演算は、整数タイプと数値タイプの任意の組み合わせを持つことができる 2 つのオペランドを取ります。オペランドに 1 つ以上の数値タイプが含まれている場合、結果タイプも数値になります。それ以外の場合は整数タイプになります。

加算演算 (+) を用いるとき、オペランドの 1 つがテキストで、もう 1 つのオペランドが整数、数値、ブール タイプまたはテキスト タイプの場合、結果タイプはテキストになります。たとえば、`"a" + 1` は、テキスト `"a1"` と評価されます。テキスト タイプではないオペランドの値がテキストに変換され、2 つのオペランドの値が結果のテキストに連結されます。減算演算 - は、x が整数タイプまたは数値タイプで `-x` のように、数値の否定として用いることもできます。演算子 `%` はモジュロ演算子、または剰余演算子と呼ばれます。この演算子は 2 つのオペランドを除算した後の余りを返します。たとえば、`5 % 2` は `1` を返します。より正確には、次のように数学的に定義されます：

```
x % y = x - trunc(x / y) * y
where
trunc(x) = sgn(x) * floor(|x|)
```

算術演算を評価すると、例外がスローされることがあります。これは、数値が大きすぎる場合のオーバーフローなど、結果が数値の制限を超えている場合に、加算、減算、乗算、および除算の演算で発生する可能性があります。この場合、`OverflowIssue` 例外がスローされます。第 2 オペランドの値がゼロの場合、除算と剰余演算子は `DivisionByZeroIssue` 例外をスローします。例：

- `17 % 2` は `1` と評価されます。
- `-17.3 % 2.0` は `-1.3` と評価されます。

等価演算子
ワークフロー エクスプレッションでは 2 つの等価演算子があります。

- `==` 値が別の値と等しいかどうかを判定します。
- `!=` 1 つのオペランドの値が別のオペランドの値と等しくないかどうかを判定します。

これらの演算子は、あらゆるタイプのオペランドで動作しますが、オペランドのタイプは同じである必要があります。たとえば、数値を整数と比較することはできません。

関係演算子
関係演算子は、一方のオペランドが他方のオペランドより小さいか大きいかを判定します。オペランドは Integer、Number、Date、Time、DateTime タイプの数値である必要があります。エクスプレッション内のオペランドのタイプは同一である必要があります。関係演算子には 4 種類あります：

演算子	説明
<	未満
<=	以下
>	より大きい
>=	以上

論理演算子
2 つの二項演算 (2 つのオペランドを取る) 論理演算子があります：AND (`&&`)、OR (`||`)。また、1 つの単項演算 (1 つのオペランドを取るもの) があります：NOT (`!`)。これらの演算子はブール タイプのオペランドに対して定義され、リターン タイプもブール タイプになります。`&&` 演算子は、両方のオペランドの値が TRUE であれば TRUE を返し、それ以外の場合には FALSE を返します。`||` 演算子は、オペランドの値

のうち 1 つまたは両方が TRUE であれば TRUE を返し、両方が FALSE の場合には FALSE を返します。! 演算子は、オペランドの値が FALSE の場合は TRUE を返し、オペランドが TRUE の場合は FALSE を返します。

`&&` および `||` 演算子の評価は、他の多くの演算子の評価とは若干異なります。通常はオペランドの評価が実行される前にすべてのオペランドが評価されますが、`&&` および `||` 演算子については、第 1 オペランドが最初に評価されます。演算子の結果を判断するのに十分であれば、2 番目の引数は評価されません。たとえば、`x==1 || x==2` において、`x` が 1 の場合、エクスプレッションの 2 番目の部分 (`x==2`) は評価されません。

条件付き演算子
条件付き演算子は 3 つのオペランドを取り、次の形工クスプレッションを取ります。

```
<Op>?<Op>:<Op>
```

ここで、`<Op>` は、制限付きで任意のオペランドになります。たとえば、`x` の値が `1` であれば、`x==1?0:1` は `0`、それ以外の場合は 1 になります。第 1 オペランドのタイプはブール タイプである必要があり、他の 2 つのオペランドは任意のタイプにすることができますが、同じである必要があります。

条件付き演算子の評価も、他の多くの演算子の評価とは若干異なります。条件付き演算子の場合、最初に第 1 オペランドが評価され、その値に応じて、他の 2 つのオペランドのうちの 1 つだけが評価されます。第 1 オペランドが TRUE (または FALSE) の場合、第 2 (または第 3) のオペランドが評価され、結果がこの評価の結果になります。これは、評価されていないオペランドに評価工ラーが発生しても、例外がスローされないことを意味します。たとえば、`x == 0.0?` で、`1.0: 1/X` では、`x` に `0.0` の値がある場合、`1/x` は評価されず、`DivisionByZeroIssue` 例外はスローされません。

演算子の概要
次の表は、エクスプレッション演算子の一覧です。

演算子	説明	例
+	加算またはテキスト連結	`1+2` `"hello " + name`

-	減算または否定	`1-2`
		`5-2.9`
		`-5`

*	乗算	`42*2` `1.0*17`
/	除算	`1/2` `1/2.0`
%	剰余演算	`x % 2` `2.5 % 1.0`
==,!=	等しい、等しくない	`true == false` `x != 0`
<,<=	未満、以下	`0 < 1` `1.0 <= 0.0`
>,>=	より大きい、以上	`0 > 1` `1.0 >= 0.0`
&&,||	論理 AND、論理 OR	`true || x` `false && y`

演算子	説明	例
!	論理 NOT	`!true`
_?_:_	条件付き演算子	`x>0? x: 0`

丸括弧
丸括弧を用いてエクスプレッションの評価順序を決定することで、エクスプレッションから得られる結果を別のものにすることができます。たとえば、エクスプレッション `1+2*3` は 7 と評価されますが、次のように丸括弧を挿入すると：`(1+2)*3` は、隣接する演算子の前で括弧の内容が評価されるため、結果は 9 に変わります。

関数

エクスプレッションには、関数呼び出しを含めることもできます。関数を呼び出すには 2 つの方法があります。1 つ目は直接関数呼び出しと呼ばれ、次のようになります。`f(<Op>,…,<Op>)`、たとえば `max(1,2)`。もう 1 つはメソッド関数呼び出しと呼ばれ、次のようになります。`<Op>.f(<Op>,…,<Op>)`、たとえば `1.max(2)`。この 2 つの方法は次のような関係になっています。

<Op1>.f(<Op2>,...,<Opn>) は f(<Op1>,...,<Opn>) と同じです。

オペランドは特定のタイプを持つ必要があり、結果タイプはオペランドのタイプに応じて異なる点において、関数は演算子に似ています。たとえば、2 つの数値で最大値を決定する関数 `max` は、整数タイプまたは数値タイプのオペランドで呼び出すことができ、戻り値のタイプはオペランドのタイプと同じになります。

評価中、関数が想定外の不正なオペランド値を取得すると、`IncorrectValueIssue` 例外がスローされます。

Kofax RPA は以下の関数を提供します。

- 数値
- テキスト
- DateTime
- 日付
- 時刻
- 変換

数値関数

関数	結果タイプ	例
abs(Integer)	整数	`abs(-5)` は 5 を返します。
abs(Number)	数値	`abs(-2)` は 2 を返します。
ceil(Number)	整数	指定された数値以上の最小の整数を返します。`ceil(2.4)` は 3 を返します。
computeMD5(binary: Binary)	テキスト	バイナリ入力の MD5 チェックサムを計算します。

関数	結果タイプ	例
floor(Number)	整数	指定された数値以下の最小の整数を返します。 `floor(8.7)` は 8 を返します。
round(Number)	整数	算術規則に基づいて、指定された数値よりも大きいまたは小さい最小の整数を返します。 `round(1.4)` は 1 を返します。 `round(1.6)` は 2 を返します。
trunc(Number)	整数	小数部を含まない数値を返します。 `trunc(4.8732)` は 4 を返します。
max(Integer, Integer)	整数	`max(4, 9)` は 9 を返します。
max(Number, Number)	数値	`max(12, 99)` は 99 を返します。
min(Integer, Integer)	整数	`min(23, 47)`は 23 を返します。
min(Number, Number)	数値	`min(15.01, 32.5)`は 15.01 を返します。
random()	数値	0.0 以上 1.0 未満の乱数を返します。 `random()`
random(Integer, Integer)	整数	第 1 オペランドの値以上、第 2 オペランドの値以下のランダムな整数を返します。 `Random(5, 96)`

テキスト関数

関数	結果タイプ	例
ampersandDecode(Text)	テキスト	`ampersandDecode(">")` は > を返します。
ampersandEncode(Text)	テキスト	`ampersandEncode(">")` は `>` を返します。
base64Decode(Text)	バイナリ	`base64Decode("SGVsbG8=").text("UTF8")` は `Hello` を返します。
binary(Text, Text)	バイナリ	テキスト値をバイナリに変換します。 `"Hello".binary("UTF8")`
boolean(Text)	ブール値	テキストは「true」または「false」のいずれかになります。 `boolean("true")` は、ブール型の「true」を返します。
capitalize(Text)	テキスト	各単語の最初の文字が大文字で、残りの文字が小文字であるテキストを返します。 `capitalize("hello world")` は `Hello World` を返します。
contains(Text, Text)	ブール値	テキストに指定したテキストが含まれていることを確認します。 `contains("I have found it", "have")` は `true` を返します。

関数	結果タイプ	例
endsWith(Text, Text)	ブール値	`endsWith("Why portraits have fascinated us for millennia", "millennia")` は `true` を返します。
indexOf(Text, Text)	整数	`indexOf("Why portraits have fascinated us for millennia", "millennia")` は 37 を返します。
integer(Text)	整数	`integer("7")`は、整数型の 7 を返します。
length(Text)	整数	記号の数を計算して、テキストの長さを返します。 `length("Hello")` は 5 を返します。
matches(text: Text, regex: Text)*	ブール値	テキストが正規表現に一致するかどうかをチェックします。 `matches("My phone number is 123-45-678", "(.*)( \\d{3}-\\d{2}-\\d{3})")` は `true` を返します。
number(Text)	数値	`number(987)` は 987.0 を返します。
password(Text)	パスワード	テキストをパスワードに変換します。これは、ロボットの開発中にアプリケーションにログインする際にパスワードが必要な場合に役立ちます。セキュリティ上の理由から、この関数を本番環境で使用しないでください。 `password("123abc!@")` は `Password=074f4d84dfe28d...` を返します。
removeNonPrintable(Text)	テキスト	印刷不可文字を含まないテキストを返します。 `removeNonPrintable("Text with non-printable characters")`
replaceAll(text: Text, regex: Text, replacement: Text)*		正規表現に一致するすべてのサブテキストを、指定された置換文字に置き換えます。 正規表現は、$ を使用した複数の一致検索および参照をサポートしています。 `"345-84-7735".replaceAll(")` は `345-84-7735` を返します。 円記号を工スケープする必要があることに注意してください。
startsWith(Text, Text)	ブール値	`startsWith("To be or not to be", "To")` は true を返します。
substring(Text, Integer)	テキスト	`"workflow".substring(5)` は `low` を返します。
substring(Text, Integer, Integer)	テキスト	`substring("DesignStudio",1,8)` は `esignSt` を返します。
toJSON(Text)	テキスト	`toJSON("Hello")` はテキストとして `"Hello"` を返します。

関数	結果タイプ	例
toLowerCase(Text)	テキスト	`toLowerCase("START")` は `start` を返します。
toUpperCase(Text)	テキスト	`toUpperCase("sun")` は `SUN` を返します。
trim(Text)	テキスト	テキスト文字列の両端から、スペースと空白文字をすべて削除します。 `trim(" Let's start ")` は `Let's start` を返します。
unquote(text: Text)	テキスト	テキストから引用符を除去します。 `"\"hello\"".unquote()` は、二重括弧なしで `hello` を返します。

* この関数の実行には、使用されるテキストと正規表現に応じて長い時間がかかることがあります。たとえば、テキストに余分なゼロを大量に入力すると、`matches` 関数が極めて長時間実行されます。最後に `A` を 1 つ追加すると、`matches("00000000000000000000000", "(0*)*A")` のようにほぼ即座に true を返します。エクスプレッションを変更するたびに、以前の評価が取り消され (まだ実行中の場合)、新しい評価が開始されます。

DateTime 関数

DateTime 関数の結果には、日付、時刻、タイム ゾーン ID、およびオフセットが含まれます。表現は指定されたパラメータによって異なります。デフォルトでは、特定のタイム ゾーンが手動で指定されていない限り、コンピュータのローカル タイム ゾーンが使用されます。これは、式が評価されるときに決定されます。

関数	結果タイプ
dateTime()	DateTime 現在の日付 (年、月、日)、時刻 (時、分、秒)、オフセット、および zoneID を返します。
dateTime(date: Date, time: Time)	DateTime 指定の日付と時刻を返します。 例 `dateTime(date(2022, 1, 10), time(12, 0, 5))` は、日付 (年 = 2022、月 = 1、日 = 10)、時刻 (時 = 12、分 = 0、秒 = 5、ナノ秒 = 0)、zoneID = "[DateTime のタイム ゾーン]" を返します。
dateTime(date: Date, time: Time, zone: Text)	DateTime 指定した日付、時刻、およびタイム ゾーン ID またはオフセットを返します。 例 `dateTime(date(2010,11,08), time(12,44,00), zoneId(Dublin))`

関数	結果タイプ
dateTime(text: Text, pattern: Text)	DateTime コンピュータのデフォルトの口ケールを使用して、Text 値を DateTime 値に変換します。 例 `dateTime("2022-01-02 12:30", "yyyy-MM-dd HH:mm")`
dateTime(text: Text, pattern: Text, locale: Text)	DateTime パターンと口ケールを使用して、テキストから DateTime 値を作成します。 例 `dateTime("January 1, 2022 12:30", "MMMM d, yyyy HH:mm ", "en-US").`
dateTime(text: Text, pattern: Text, defaultDate: Date)	DateTime パターンと、テキストに日付が含まれていない場合に使用するデフォルトの日付を使用して、Text 値を DateTime 値に変換します。 例 `dateTime("12:30", "HH:mm", date())` ここで、日付は現在の日付です。
dateTime(text: Text, pattern: Text, dafaultDate: Date, locale: Text)	DateTime パターン、デフォルトの日付、および口ケールを使用して、テキストから DateTime 値を作成します。パターンによってテキストと日付が一致する場合は口ケールが使用され、デフォルトの日付は使用されません。テキストに日付が含まれていない場合は、デフォルトの日付が使用されます。 例 `dateTime(text, "[MMMM d, yyyy] HH:mm", date(), "en-US")`。ここで、`text` は抽出する日付を含む変数であり、`date()` は現在の日付です。パターンは両方のテキストに一致します。`January 1, 2022 12:30` および `12:30`。最初のケースでは、`January1, 2022` という日付で DateTime 値が返されます。2 番目のケースでは、現在の日付で返されます。どちらの場合も、時刻は `12:30` です。
offset(dateTime: DateTime)	テキスト DateTime 値のオフセットを返します。 例 `offset(dateTime())` は、使用しているコンピュータにおける現在の日付のオフセットを `+01:00` のような形で返します。
zoneId(dateTime: DateTime)	テキスト DateTime 値のタイム ゾーンを返します。 例 `zoneId(dateTime())` は、現在の日付の `zoneID` の値を `Europe/Paris` のような形で返します。

関数	結果タイプ
withYears(dateTime: DateTime, years: Integer)	DateTime 年を変更した日付と時刻を新たに返します。 例 `dateTime(date(2021, 7, 20), time(12, 13, 14)).withYears(2022)` は、年が 2022 である DateTime 値を返します。
withMonth(dateTime: DateTime, months: Integer)	DateTime 月を変更した日付と時刻を新たに返します。例については、「withYears」を参照してください。
withDay(dateTime: DateTime, days: Integer)	DateTime 日を変更した日付と時刻を新たに返します。例については、「withYears」を参照してください。
withHour(dateTime: DateTime, hours: Integer)	DateTime 時間を変更した日付と時刻を新たに返します。例については、「withYears」を参照してください。
withMinute(dateTime: DateTime, minutes: Integer)	DateTime 分を変更した日付と時刻を新たに返します。例については、「withYears」を参照してください。
withSecond(dateTime: DateTime, seconds: Integer)	DateTime 秒を変更した日付と時刻を新たに返します。例については、「withYears」を参照してください。
withNano(dateTime: DateTime, nanos: Integer)	DateTime ナノ秒を変更した日付と時刻を新たに返します。例については、「withYears」を参照してください。
withOffset(dateTime: DateTime, offset: Text)	DateTime 指定したオフセットの日付と時刻を新たに返します。オフセットを変更すると、オフセットとゾーン ID が互いに矛盾しないように、ゾーン ID が変更される場合があります。 例 (オフセットが +1 の場合) `withOffset(dateTime(), "+3")` は、オフセットが +3 の時刻を返します。これは、時刻が現在の時刻よりも 2 時間進んでいることを意味します。また、ゾーン ID は UTC +03:00 (ヨーロッパ/モスクワ) に変更されます。
withZoneId(dateTime: DateTime, zoneID: Text)	DateTime 指定したタイム ゾーンの日付と時刻を新たに返します。ゾーン ID を変更すると、ゾーン ID とオフセットが互いに矛盾しないように、オフセットが変更される場合があります。 例 `withZoneId(dateTime(), "Europe/Moscow")` は、指定されたゾーン ID における現在の時刻を返します。

関数	結果タイプ
plusYears(dateTime: DateTime, amount: Integer)	DateTime 指定した年数を加えた日付と時刻を新たに返します。 例 `dateTime(date(2021, 7, 20), time(12, 13, 14)).plusYears(1)` は、ロボットが実行されるタイムゾーンにおける、日時 2022-07-22 12:13:14.0 の DateTime 値を返します。
plusMonths(dateTime: DateTime, amount: Integer)	DateTime 指定した月数を加えた日付と時刻を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"plusDays" を参照してください。
plusWeeks(dateTime: DateTime, amount: Integer)	DateTime 指定した週数を加えた日付と時刻を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"plusDays" を参照してください。
plusDays(dateTime: DateTime, amount: Integer)	DateTime 指定した日数を加えた日付と時刻を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。 `plusDays(70)` を使用した次の例では、3 月は 31 日、4 月は 30 日あるという前提に基づき、まず 3 月 4 日に 27 日を追加し、さらに残りの 43 日を追加して 5 月 13 日としています。 例 `dateTime(date(2022, 3, 4), time(12, 13, 14)).plusDays(70)` は、日付 (年 = 2022、月 **= 5**、日 **= 13**)、時刻 (時 = 12、分 = 13、秒 = 14、ナノ秒 = 0)、zoneID = "[DateTime のタイム ゾーン]" を返します。
plusHours(dateTime: DateTime, amount: Integer)	DateTime 指定した時間数を加えた日付と時刻を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"plusDays" を参照してください。
plusMinutes(dateTime: DateTime, amount: Integer)	DateTime 指定した分数を加えた日付と時刻を新たに返します。 例 `dateTime(date(2022, 7, 20), time(12, 13, 14)).plusMinutes(7)` は、日付 (年 = 2022、月 = 7、日 = 20)、時刻 (時間 = 12、分 **= 20**、秒 = 14、ナノ秒 = 0)、zoneID = "[DateTime のタイム ゾーン]" を返します どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。

関数	結果タイプ
plusSeconds(dateTime: DateTime, amount: Integer)	DateTime 指定した秒数を加えた日付と時刻を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"plusDays" を参照してください。
plusNanos(dateTime: DateTime, amount: Integer)	DateTime 指定したナノ秒数を加えた日付と時刻を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"plusDays" を参照してください。
minusYears(dateTime: DateTime, amount: Integer)	DateTime 指定した年数を差し引いた日付と時刻を新たに返します。 例 `dateTime(date(2022, 7, 20), time(12, 13, 14)).minusYears(10)` は、ロボットが実行されるタイムゾーンにおける、日時 2012-07-22 12:13:14.0 の DateTime 値を返します。
minusMonths(dateTime: DateTime, amount: Integer)	DateTime 指定した月数を差し引いた日付と時刻を新たに返します。 例 `dateTime(date(2022, 7, 20), time(12, 13, 14)).minusMonths(10)` は、ロボットが実行されるタイムゾーンにおける、日時 2021-09-20 12:13:14 の DateTime 値を返します。
minusWeeks(dateTime: DateTime, amount: Integer)	DateTime 指定した週数を差し引いた日付と時刻を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。例については、「minusMonths」を参照してください。
minusDays(dateTime: DateTime, amount: Integer)	DateTime 指定した日数を差し引いた日付と時刻を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。`minusDays(20)` を使用した次の例では、3 月は 31 日、2 月は 29 日あるという前提に基づき、まず 3 月 14 日から 14 日を差し引き、さらに残りの 6 日を差し引いて 2 月 23 日としています。 例 `dateTime(date(2020, 3, 14),time(12, 13, 14)).minusDays(20)` は、日付 (年 = 2020、月 = **2**、日 = **23**)、時刻 (時間 = 12、分 = 13、秒 = 14、ナノ秒 = 0)、zoneID = "[DateTime のタイム ゾーン]" を返します。

関数	結果タイプ
minusHours(dateTime: DateTime, amount: Integer)	DateTime 指定した時間数を差し引いた日付と時刻を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"minusDays" を参照してください。
minusMinutes(dateTime: DateTime, amount: Integer)	DateTime 指定した分数を差し引いた日付と時刻を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"minusDays" を参照してください。
minusSeconds(dateTime: DateTime, amount: Integer)	DateTime 指定した秒数を差し引いた日付と時刻を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"minusDays" を参照してください。
minusNanos(dateTime: DateTime, amount: Integer)	DateTime 指定したナノ秒数を差し引いた日付と時刻を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"minusDays" を参照してください。
yearsUntil(from: DateTime, to: DateTime)	整数 最初の DateTime 引数から 2 番目の DateTime 引数までの年数を計算します。 例 `dateTime(date(2022, 1, 20), time(12, 13, 14)).yearsUntil(dateTime(date(3785, 11, 20), time(12, 13, 14)))` は 1763 を返します。
monthsUntil(from: DateTime, to: DateTime)	整数 最初の DateTime 引数から 2 番目の DateTime 引数までの月数を計算します。 例 `dateTime(date(2022, 1, 20), time(12, 13, 14)).monthsUntil(dateTime(date(2022, 11, 20), time(12, 13, 14)))` は 10 を返します。
daysUntil(from: DateTime, to: DateTime)	整数 最初の DateTime 引数から 2 番目の DateTime 引数までの日数を計算します。例については、「monthsUntil」を参照してください。

関数	結果タイプ
hoursUntil(from: DateTime, to: DateTime)	整数 最初の DateTime 引数から 2 番目の DateTime 引数までの時間数を計算します。例については、「monthsUntil」を参照してください。
minutesUntil(from: DateTime, to: DateTime)	整数 最初の DateTime 引数から 2 番目の DateTime 引数までの分数を返します。例については、「monthsUntil」を参照してください。
secondsUntil(from: DateTime, to: DateTime)	整数 最初の DateTime 引数から 2 番目の DateTime 引数までの秒数を返します。例については、「monthsUntil」を参照してください。
millisUntil(from: DateTime, to: DateTime)	整数 最初の DateTime 引数から 2 番目の DateTime 引数までのミリ秒数を返します。例については、「monthsUntil」を参照してください。
nanosUntil(from: DateTime, to: DateTime)	整数 最初の DateTime 引数から 2 番目の DateTime 引数までのナノ秒数を返します。例については、「monthsUntil」を参照してください。
dayOfYear(dateTime: DateTime)	整数 年の日にちを返します。 例 `dateTime(date(2022, 3, 1), time(12, 13, 14)).dayOfYear()` は日 = 60 を返します
weeksUntil(from: DateTime, to: DateTime)	整数 最初の DateTime 引数から 2 番目の DateTime 引数までの週数を返します。 例 `dateTime(date(2022, 1, 20), time(12, 13, 14)).weeksUntil(dateTime(date(2025, 11, 20), time(12, 13, 14)))` は 200 を返します。
text(dateTime: DateTime)	テキスト DateTime 値をテキスト表現に変換します。 例 `text(dateTime())` は、たとえば 2022-02-10T13:46:56.907958+01:00 [ヨーロッパ/パリ] のような形式で現在の日付と時刻を返します。

関数	結果タイプ
text(dateTime: DateTime, pattern: Text)	テキスト DateTime 値を、指定したパターンと口ケールに一致するテキスト表現に変換します。パターンは、次のようなさまざまな方法で指定できます。 • `"d/M/yyyy HH:mm"` • `"d-M-yyyy HH:mm:ss"` 例 (DateTime が 2022-12-20T00:30:00+03:00 の場合) `text("d/M/yyyy HH:mm")` は `20/12/2022 00:30` を返します。

text(dateTime: DateTime, pattern: Text, locale: Text)	テキスト
	指定したパターンと口ケールに一致する、DateTime の日付と時刻のテキスト表現を返します。
	• パターンは、次のようなさまざまな方法で指定できます。
	  • `"d/M/yyyy HH:mm"`
	  • `"d-M-yyyy HH:mm:ss"`
	  • `"EEEE MMM d yy HH:mm"` (`EEEE` は完全な曜日を、`MMM` は 3 文字に省略した月の名前を返します。その他も同様です)
	• 口ケールは、`en-US`、`fr-FR`、`ja-JP`、`ru-RU` のように指定します。
	例 (現在の日付が 20.12.2021 00:30:00) である場合
	`text("EEEE MMM d yy HH:mm", "en-US")` は `Monday Dec 20 21 00:30` を返します。
	`text("EEE MMMM d yyyy HH:mm", "en-US")` は `Mon December 20 2021 00:30` を返します。
	`text("yyyy'年'MM'月'd'日' HH:mm:ss","ja-JP")` は `2021年12月20日 00:30:00` を返します。

fromExcelDate(excelDate: Number)	DateTime DateTime 値を数値として表現された Excel の日付に変換します。 例 (ExcelDate 値が 44969 の場合) `ExcelDate.fromExcelDate()` は、year=2023、month=2、day=12 を返します。
toExcelDate(dateTime: DateTime)	Number DateTime 値を数値として表現された Excel の日付に変換します。 例 (現在の日付が 08.02.2022 18:45:00 である場合) `dateTime().toExcelDate()` は 44600.760508449071494396775960922224 を返します。

日付関数

関数	結果タイプ
date()	日付 現在の日付を返します。
date(year: Integer, month: Integer, day: Integer)	日付 指定した年、月、および日の日付を返します。
date(DateTime)	日付 以前に DateTime が指定されていた場合に、DateTime の日付値を返します。
date(text: Text, pattern: Text)	日付 パターンを使用して、Text 値を Date 値に変換します。 例 `date("2022-01-02", "yyyy-MM-dd")`
date(text: Text, pattern: Text, locale: Text)	日付 パターンと口ケールを使用して、Text 値を Date 値に変換します。 例 `date("January 1, 2022", "MMMM d, yyyy", "en-US")`
year(date: Date)	整数 現在の日付の年を返します。
month(date: Date)	整数 現在の日付の月 (1 ~ 12) を返します。
day(date: Date)	整数 現在の日付の日 (1 ~ 31) を返します。
withYear(date: Date, years: Integer)	日付 年を変更した日付を新たに返します。 例 `date(2022, 3, 4).withYear(2037)` は、**year = 2037**、month = 3、day = 4 を返します。
withMonth(date: Date, months: Integer)	日付 月を変更した日付を新たに返します。 `date(2022, 3, 4).withMonth(5)` は、year = 2022、**month = 8**、day = 4 を返します。
withDay(date: Date, days: Integer)	日付 日を変更した日付を新たに返します。 例 `date(2022, 3, 4).withDay(10)` は、year = 2022、month = 3、**day = 10** を返します。

関数	結果タイプ
plusYears(date: Date, amount: Integer)	日付 指定した年数を加えた日付を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"plusDays" を参照してください。
plusMonths(date: Date, amount: Integer)	日付 指定した月数を加えた日付を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"plusDays" を参照してください。
plusWeeks(date: Date, amount: Integer)	日付 指定した週数を加えた日付を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"plusDays" を参照してください。
plusDays(date: Date, amount: Integer)	日付 指定した日数を加えた日付を新たに返します。 どの「plus」関数でも、暦に基づく繰り越し計算が使用されます。`plusDays(70)` を使用した次の例では、3 月は 31 日、4 月は 30 日あるという前提に基づき、まず 3 月 4 日に 27 日を追加し、さらに残りの 43 日を追加して 5 月 13 日としています。 例 `date(2022, 3, 4).plusDays(70)` は、year = 2022、**month = 5**、**day = 13** を返します。
minusYears(date: Date, amount: Integer)	日付 指定した年数を差し引いた日付を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。 `date(2022, 3, 14).minusYears(20)` は、**year = 2002**、month = 3、day = 14 を返します。
minusMonths(date: Date, amount: Integer)	日付 指定した月数を差し引いた日付を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"minusDays" を参照してください。
minusWeeks(date: Date, amount: Integer)	日付 指定した週数を差し引いた日付を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。例については、"minusDays" を参照してください。

関数	結果タイプ
minusDays(date: Date, amount: Integer)	日付 指定した日数を差し引いた日付を新たに返します。 どの「minus」関数でも、暦に基づく繰り越し計算が使用されます。`minusDays(20)` を使用した次の例では、3 月は 31 日、2 月は 29 日あるという前提に基づき、まず 3 月 14 日から 14 日を差し引き、さらに残りの 6 日を差し引いて 2 月 23 日としています。 例 `date(2022, 3, 14).minusDays(20)` は、年 = 2022、月 **= 2**、日 **= 23** を返します。
yearsUntil(from: Date, to: Date)	整数 最初の Date 引数から 2 番目 の Date 引数までの年数を返します。 例 (現在の年が 2022 年である場合) `date().yearsUntil(date(2023, 11, 12))` は 1 を返します。
monthsUntil(from: Date, to: Date)	整数 最初の Date 引数から 2 番目 の Date 引数までの月数を返します。例については、「yearsUntil」を参照してください。
daysUntil(from: Date, to: Date)	整数 最初の Date 引数から 2 番目 の Date 引数までの日数を返します。例については、「yearsUntil」を参照してください。
weeksUntil(from: Date, to: Date)	整数 最初の Date 引数から 2 番目 の Date 引数までの週数を返します。例については、「yearsUntil」を参照してください。
isLeapYear(date: Date)	ブール値 年がうるう年の場合は「true」を返し、そうでない場合は「false」を返します。 例 `date(2020, 3, 4).isLeapYear` は、2020 年がうるう年であったため、`true` を返します。
text(date: Date)	テキスト Date 値をテキスト表現に変換します。 例 `text(date(2000, 12, 25))` は `2000-12-25` を返します。

関数	結果タイプ
text(date: Date, pattern: Text)	テキスト Date 値を、指定したパターンに一致するテキスト表現に変換します。 例 `text(date(2000, 12, 25), "M/YY/d")` は `12/00/25` を返します。
text(date: Date, pattern: Text, locale: Text)	テキスト Date 値を、指定したパターンとロケールに一致するテキスト表現に変換します。 例 `text(date(2023, 12, 12), "MMMM d, yyyy", "de-DE")` は `Dezember 12, 2023.` を返します。

時刻関数

関数	結果タイプ
time()	時刻 現在の時刻を返します。
time(hours: Integer, minutes: Integer, seconds: Integer)	時刻 指定した時、分、および秒の時刻を返します。
time(hours: Integer, minutes: Integer, seconds: Integer, nanos: Integer)	時刻 指定した時、分、秒、およびナノ秒の時刻を返します。
time(dateTime: DateTime)	時刻 以前に DateTime が指定されていた場合に、DateTime の時刻値を返します。
time(text: Text, pattern: Text, locale: Text)	時刻 指定したパターンと ロケールを使用して、Text 値をTime 値に変換します。 例 `time("January 1, 2022 12:30", "MMMM d, yyyy hh:mm a", "en-US")`
hour(time: Time)	整数 現在の時刻の時間を返します。
minute(time: Time)	整数 現在の時刻の分を返します。
second(time: Time)	整数 現在の時刻の秒を返します。
nanosecond(time: Time)	整数 現在の時刻のナノ秒を返します。

関数	結果タイプ
withHour(time: Time, hours: Integer)	時刻 時間を変更した時刻を新たに返します。 例 `time(14, 50, 45).withHour(2)` は、時 = **2**、分 = 50、秒 = 45、ナノ秒 = 0 を返します。
withMinute(time: Time, minutes: Integer)	時刻 分を変更した時刻を新たに返します。例については、「withHour」を参照してください。
withSecond(time: Time, seconds: Integer)	時刻 秒を変更した時刻を新たに返します。例については、「withHour」を参照してください。
withNano(time: Time, nanos: Integer)	時刻 ナノ秒を変更した時刻を新たに返します。例については、「withHour」を参照してください。
plusHours(time: Time, amount: Integer)	時刻 指定した時間数を加えた時刻を新たに返します。 例 `time(14, 50, 45).plusHours(2)` は、時 = **16**、分 = 50、秒 = 45、ナノ秒 = 0 を返します。
plusMinutes(time: Time, amount: Integer)	時刻 指定した分数を加えた時刻を新たに返します。例については、「plusHours」を参照してください。
plusSeconds(time: Time, amount: Integer)	時刻 指定した秒数を加えた時刻を新たに返します。例については、「plusHours」を参照してください。
plusNanos(time: Time, amount: Integer)	時刻 指定したナノ秒数を加えた時刻を新たに返します。例については、「plusHours」を参照してください。
minusHours(time: Time, amount: Integer)	時刻 指定した時間数を差し引いた時刻を新たに返します。 例 `time(14, 50, 45).minusHours(2)` は、時 = **12**、分 = 50、秒 = 45、ナノ秒 = 0 を返します。
minusMinutes(time: Time, amount: Integer)	時刻 指定した分数を差し引いた時刻を新たに返します。例については、「minusHours」を参照してください。
minusSeconds(time: Time, amount: Integer)	時刻 指定した秒数を差し引いた時刻を新たに返します。例については、「minusHours」を参照してください。
minusNanos(time: Time, amount: Integer)	時刻 指定したナノ秒数を差し引いた時刻を新たに返します。例については、「minusHours」を参照してください。

関数	結果タイプ
hoursUntil(from: Time, to: Time)	整数 最初の Time 引数から 2 番目の Time 引数までの時間数を返します。 例 `time(9, 41, 35).hoursUntil(time(15, 34, 12))` は 5 を返します。
minutesUntil(from: Time, to: Time)	整数 最初の Time 引数から 2 番目の Time 引数までの分数を返します。例については、「hoursUntil」を参照してください。
secondsUntil(from: Time, to: Time)	整数 最初の Time 引数から 2 番目の Time 引数までの秒数を返します。例については、「hoursUntil」を参照してください。
millisUntil(from: Time, to: Time)	整数 最初の Time 引数から 2 番目の Time 引数までのミリ秒数を返します。例については、「hoursUntil」を参照してください。
nanosUntil(from: Time, to: Time)	整数 最初の Time 引数から 2 番目の Time 引数までのナノ秒数を返します。例については、「hoursUntil」を参照してください。
toSecondsOfDay(time: Time)	整数 時刻を秒に変換します。 例 `time(1, 0, 0).toSecondsOfDay()` は 3600 を返します。
toNanosOfDay(time: Time)	整数 時刻をナノ秒に変換します。 例 `time(1, 0, 0).toNanosOfDay()` は 3600000000000 を返します。
text(time: Time)	テキスト Time 値をテキスト表現に変換します。 例 `hour(time(10, 15, 00))` は、10:15:00 を返します。
text(time: Time, pattern: Text)	テキスト Time 値を、指定したパターンに一致するテキスト表現に変換します。 例 `text(time(10, 15, 00), "hh:m a")` は 10:15 AM を返します。

関数	結果タイプ
text(time: Time, pattern: Text, locale: Text)	テキスト Time 値を、指定したパターンとロケールに一致するテキスト表現に変換します。 例 `text(time(10, 15, 00), "hh:m a", "de-DE")` は 10:15 vorm を返します。

変換関数

変換関数は、あるタイプから別のタイプへ値を変換します。オペランドの値が、結果タイプの値に変換可能な値を表さない場合、変換が失敗することがあります。

関数	結果タイプ
ampersandEncode(text: Text)	テキスト テキストをアンバンサンド エンコードします。 例 `ampersandEncode("&")` は `&` を返します。
ampersandDecode(text: Text)	テキスト テキストをアンバンサンド デコードします。 例 `ampersandDecode(">")` は > を返します。
base64Encode(binary: Binary)	テキスト MIME (RFC 2045) 用の Base64 転送エンコードを使用する Base64 エンコード。 例 `binary.base64Encode()`。ここで、`binary` はバイナリ値を持つ変数です。
base64Decode(text: Text)	バイナリ MIME (RFC 2045) 用の Base64 転送エンコードを使用する Base64 デコード。 例 `base64Decode("SGVsbG8=").text("UTF8")` は `Hello` を返します。
boolean(text: Boolean)	ブール値 テキストは「true」または「false」のいずれかに一致する必要があります。 例 `boolean(false:Boolean)` は `false` を返します。
integer(Number)	整数 数値は 1.0 などの整数値である必要があります。
integer(Text)	整数 テキストは、"42" などの整数のテキスト表現である必要があります。

関数	結果タイプ
number(Integer)	数値 `Number(17)` は 17.0 を返します。
number(Text)	数値 テキストは、"17.7" などの数字のテキスト表現である必要があります。
text(Integer)	テキスト
text(Number)	テキスト
text(Boolean)	テキスト
text(binary: Binary, charsetName: Text)	テキスト テキストの二進表現をテキスト タイプの値に変換します。`UTF8` などの文字セットを引数として指定します。
binary(text: Text, charsetName: Text)	バイナリ テキスト値をバイナリ値に変換します。`UTF8` などの文字セットを引数として指定します。 例 `"Hello".binary("UTF8").text("UTF8")` は `Hello` を返します。ここで、`"Hello".binary("UTF8")` は UTF8 文字エンコードを使用してエンコードされたバイナリ値に評価されます。例のようにバイナリ値がテキストに変換される場合、バイナリ値は同じ文字エンコード (この場合は UTF8) を使用してエンコードする必要があります。

関数	結果タイプ
toJSON()	パスワード以外のあらゆるタイプの値を、JSON オブジェクトとしてフォーマットされたテキスト値に変換します。パスワード タイプ属性を含むレコード タイプは JSON に変換できません。 変換された値の例 • 5 は "5" となります • 1.2 は "1.2" となります • true は "true" となります • "Hello" は "\"Hello\"" となります • バイナリ値は、バイナリ値の Base 64 エンコードになります • レコード値、R(a = 5, b = true, t = "Hello") は "{\"a\":5, \"b\":true,\"t\":\"Hello\"" となります ワークフロー状態ビューには、引用符の前の逆スラッシュが表示されません。上に示したレコードの値の変換方法は、エクスプレッションに記述する方法です。 toJSON 関数は、[1,2,3] のような配列を生成することはありません。ロボットで JSON 値を使用する場合は、文字列を連結することによりリストを作成できます。このようにすると、ロボットからベーシック エンジン ロボットに値を返すことができます。 詳細については、JSON の使用を参照してください。

次の表は、変換関数の一覧と説明です。これらは、text(2) などの直接関数呼び出しと 2.text() などのメソッド関数呼び出しという 2 とおりの記述が可能です。最初のバリアントでは関数は直接呼び出されますが、2 番目のバリアントではエクスプレッション モードでエクスプレッションを記述する場合に、自動テキスト入力を使用できます。詳細については、後述のエクスプレッション モードを参照してください。

例	
関数	以下の結果に評価
ampersandEncode("<b/>") または "<b/>".ampersandEncode()	<b/>
ampersandDecode("<b/>") または "<b/>".ampersandDecode()	<b/>
base64Encode(bin) または bin.base64Encode() bin は "Hello".binary("ASCII") の結果のバイナリ値を保持している変数です	SGVsbG8=

例	
関数	以下の結果に評価
base64Decode("SGVsbG8=").text("UTF8") または "SGVsbG8=".base64Decode().text("UTF8")	Hello
• "true".boolean() • "false".boolean() • "False".boolean()	• Boolean true • Boolean false • 「true」または「false」のいずれにも一致しないため、ConversionIssue 例外をスローします
• integer(2.0) または 2.0.integer() • integer("2") または "2".integer	2
• number(2) または 2.number() • number("2") または "2".number()	2.0
number(".1E10") または ".1E10".number()	1.0E9
"Hello".binary("UTF8").text("UFT8") ここで "Hello".binary("UTF8") は UTF8 文字エンコードを使用してエンコードされたバイナリ値と評価します。例のようにバイナリ値がテキストに変換される場合、バイナリ値は同じ文字エンコード (この場合は UTF8) を使用してエンコードする必要があります。	Hello
"hello".toJSON()	"hello"
17.7.toJSON()	17.7
true.toJSON()	TRUE
Company.toJSON() ここで"Company"はテキストタイプ、整数タイプ、ブールのフィールド タイプを含むレコード タイプの変数です。 Variables company: Company name = "Acme Corp." url = "www.acme-corp.com" revenue = 1000000000 CEO = "Marvin Acme" fictional = true	次の値のテキスト値: {"name":"Acme Corp.", "url":"www.acme-corp.com", "revenue":1000000000, "CEO":"Marvin Acme", "fictional":true}

数値の制限

次に示す数値の制限を考慮してください。

整数値
表現できる最大の整数は 1E34-1 です。これは 9 が 34 個並んだ数値です。この数字に対して 1 を加えると、`OverflowIssue` の例外が発生します。同様に、表現できる最小の整数値は -1E34+1 です。したがって整数値はすべて、-1E34+1 から 1E34-1 (どちらも含む) の範囲内である必要があります。

数値
数値の表現は、小数点以下を含め 34 桁の数を使用する IEEE 754R Decimal128 と、-2147483648 ~ +2147483648 の範囲の指数の表現に一致します。エクスプレッションの評価で範囲外の数値になった場合、`OverflowIssue` 例外が発生します。

詳細については、数値の制限を参照してください。

次のように、指数で数値を入力することができます。

1.234E5、0.1E-2, 1000.0e42、など。

基本数の後には、任意の `E` または `e` および整数の指数が続きます。[データの状態] ビューまたはエクスプレッション エディターに数値が表示される際には、次のルールに従って正規化されます。

- 他の 10 進数がない場合、小数点の後にゼロが 1 つ表示されます (1.0E12 など)。
- 指数は、数値が 9 以上の場合にのみ表示されます。つまり、次のようになります。
- • 1.0E8 は 100000000 と表示されますが、1.0E9 は 1.0E9 と表示されます
  • 1.0E-8 は 000000001 と表示されますが、1.0E -9 は 1.0E-9 と表示されます。
- 数字は最大 34 桁まで表示されます (数字は四捨五入されて最大 34 桁となります)。たとえば、9.99999999999999999999999999999999 という数字はそのまま表示されます。この数字に 9 をもう 1 つ追加すると、10.0 と表示されます。実際には、数値は四捨五入されているため、内部的にも 10.0 として表されます。したがって、数値を一方向に入力しながら別の方法で表示することができるため、10.0 や 0.1E2 など、数値の入力方法に柔軟性を持たせることができます。

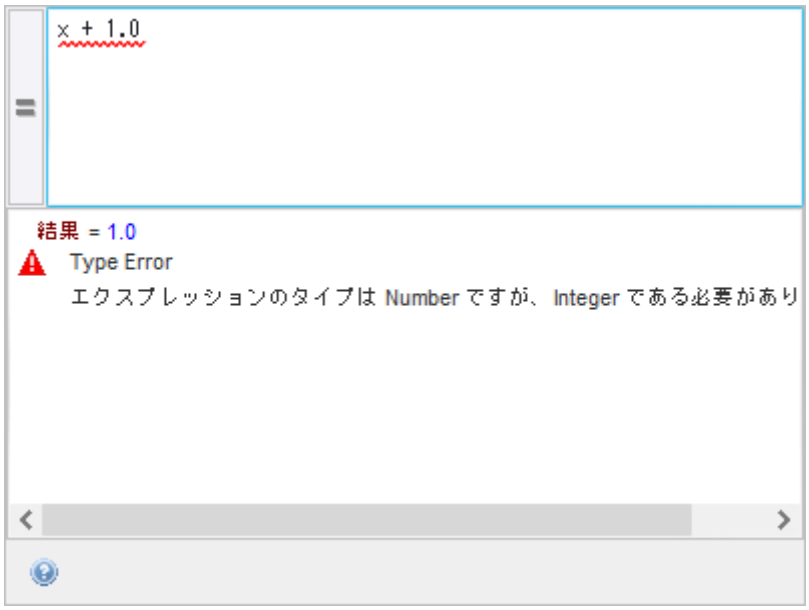
また、変換関数 `text(Number)` も上記のルールに従って記述された数値を返します。

エクスプレッション エディター

エクスプレッション エディターは、ロボットで入力フィールドをクリックした場合、および指定したフィールドがエクスプレッションの入力をサポートしている場合に開く、インタラクティブなエディターです。エディターは 2 つのペインで構成されています。上部のペインはエクスプレッションの入力や編集を行う入力ペインで、下部のペインは結果ペインです。結果ペインには、エクスプレッションの評価結果、エラーメッセージ、またはその両方が表示されます。次のエラーが表示されることがあります：

- Parse errors: エクスプレッションの構文が不正です。
- Type errors: エクスプレッションにタイプ エラーがある、または結果が正しいタイプとなっていません。
- 評価 errors: ゼロによる除算など、エクスプレッションの評価中にエラーが発生しました。

次の例は、`x + 1.0` というエクスプレッションです。ここで `x` は整数タイプの変数であるとします。エクスプレッションには正しいタイプが付けられていますが、結果は整数タイプの変数に割り当てられるものであるため、エクスプレッションの結果タイプは想定外であるというエラー メッセージが表示されます。



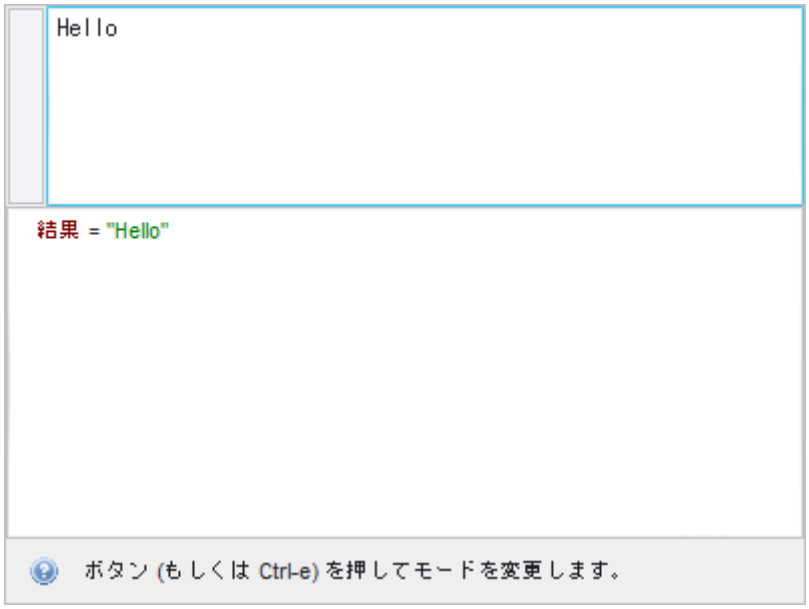
右クリックして [値のコピー] を選択すると、エディターの下部ペインからエクスプレッションの結果とエラー メッセージをコピーすることができます。値がレコード タイプの場合、結果はツリーとして表示され、各属性値は個別にコピーできます。パスワードとバイナリ値はコピーできません。

エクスプレッション エディターを閉じるには、エディターの外側をクリックするか、[Esc] を押します。

エディター モード
エディターには、上部ペインの左側にエクスプレッション モードと値モードを切り替えるモード ボタンがあります。上図のエディターは、エクスプレッション モードです。エディターが値モードになっているとき、モード ボタンは空白となります。エディターがエクスプレッション モードのとき、ボタンには等号 (=) が表示されます。キーボード ショートカット [Ctrl-E] を使って、2 つのモードを切り替えることができます。

値モード
値モードでは、入力された値は単にテキスト、ブール値、数値などの値として解釈され、評価は行われません。結果パネルに結果が表示されます。表示できる唯一のエラーは、結果タイプが不正な場合です。

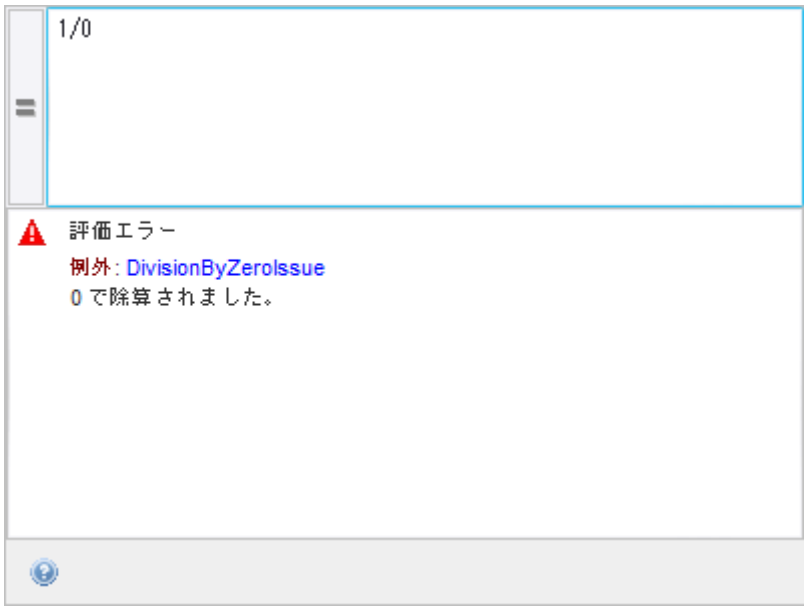
値モードで入力された値の先頭に「=」がある場合、エディターはエクスプレッション モードに切り替わります。



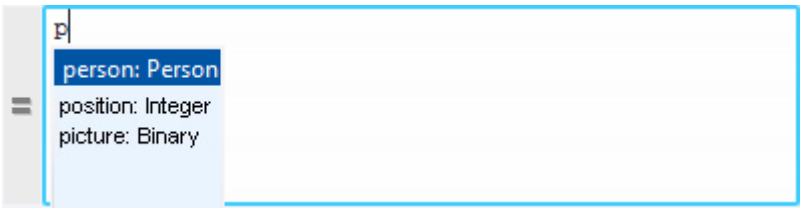
エクスプレッション モード

エクスプレッション モードでは、入力ペインに入力したすべての内容がエクスプレッションとして解釈され、構文とタイプのエラーについてチェックを行うことができます。現在のフロー ポイントでエクスプレッションを編集していてエラーがない場合は、評価され、その結果が表示されます。エクスプレッションの状態が常にわかるように、入力の途中で評価が行われます。現在のフロー ポイントにないエクスプレッションを編集する場合は、変数が含まれていないか、値が現在の状態に依存していないか評価されます。

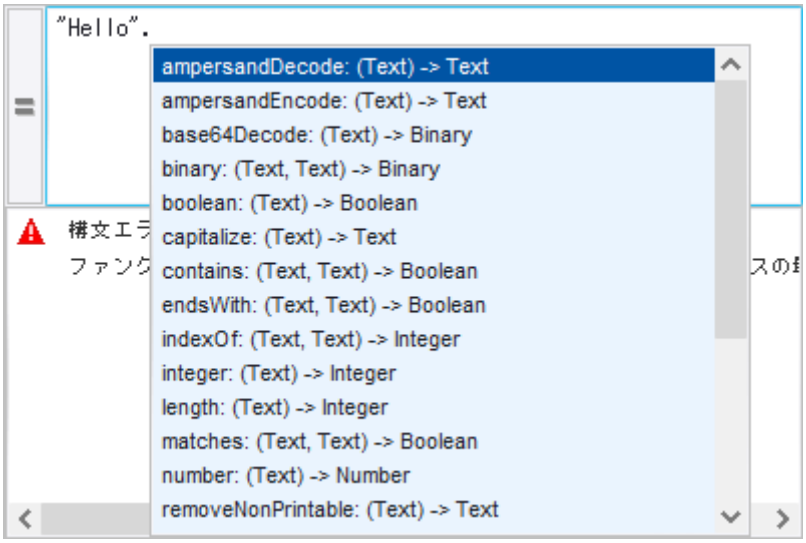
エクスプレッションの評価中に、たとえばゼロによる除算などのエラーが発生した場合、例外の名前と問題を説明するメッセージが結果ペインに表示されます。例外の名前を右クリックし、[値のコピー] を選択してその名前をコピーすることができます。



エクスプレッション モードでのテキスト補完は、変数名、フィールド名、関数名の入力に役立ちます。補完ヘルプがあるものを入力すると、テキスト補完ウィンドウが自動的に表示されます。たとえば、変数名の入力を開始し、すでに入力したものから始まる変数がある場合、補完ウィンドウが表示されます。レコード タイプの変数の後でドット (.) を押すと、補完ウィンドウには、レコード タイプのフィールドに対応した補完オプションのリストが表示されます。次の例は、"p" と入力した後の補完ヘルプを示します。

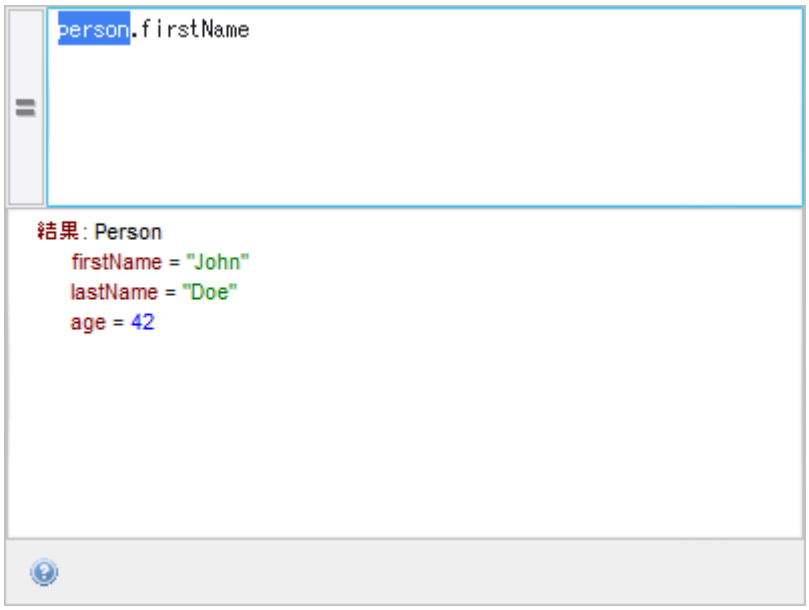


シンプル タイプのサブエクスプレッションの後にドット (.) を押すと、最初の引数が同じシンプル タイプを持つ関数に対応する補完オプションのリストが、補完ウィンドウに表示されます。



補完リストのオプション間を移動するには、矢印キーまたはマウスを使用します。補完リストでオプションを選択するには、[Enter]、[Tab] を押すか、オプションをダブルクリックします。何も入力せずに補完ウィンドウを開くには、[Ctrl+Space] を押します。補完ウィンドウを閉じるには、ウィンドウの外側をクリックするか、[Esc] を押します。

入力ペインでエクスプレッションの一部が選択され、その選択が適切なサブエクスプレッション (それ自身で評価可能) に相当する場合、このサブエクスプレッションの値が下の図のように結果ペインに表示されます。



入力ペインで関数の名前を選択すると、この関数の説明が結果ペインに表示されます。

数値の制限

ロボット およびベーシック エンジン ロボット の数値形式は異なります。ベーシック エンジン ロボットでは、変数に格納される値は倍精度です (IEEE 754 規格で規定された binary64)。ロボットでは、変数は IEEE 754R decimal128 形式で数値を格納し、34 桁の 10 進数と #2147483647 ~ +2147483648 (= 2^31) の指数範囲が使用されます。ロボットに値を格納するときに四捨五入が発生するため、精度が低下することがあります。たとえば、数値が 34 桁を超え、最後の数字が `.5` の場合、四捨五入されるので、 `.5` は `.0` になります。またたとえば、12345678901234567890123456789012345678901234567890.0 は、12345678901234567890123456789012350000000 になります。大きな整数の数値をとることもできますが、有効桁数は 34 桁です。

利用できる数値は以下のとおりです：最大 9.9999999999...E2147483647 から最小 0.1E-2147483646 までの数値これらの数値は、エクスプレッションに `"0.1E-2147483646".number()` を書き込むなどして、テキストから数値に変換できる最大値と最小値です。乗算を使ってより大きな数値を得ることもできますが、オーバーフロー エラーが発生する可能性があります。数値の表記には以下のような制約があります：

`"9.9999999999…9E2147483647".number()` は、数値が 34 桁以上ある場合 1.0E2147483648 に変換され、34 桁未満の場合は 9.9999999999...9E2147483647 に変換されます。

`"0.1E-2147483646".number()` は、1.0E-2147483647 に変換されます。

RDP 接続の使用

リモート デスクトップ セッションを使用するセッション管理の方が標準ログイン セッションよりも優先する場合は、RDP 接続を介してデバイスに接続できます。

前提条件
RDP を使用するには、環境がスクリーン ロック機能の要件に適合する必要があります。『Kofax RPA Desktop Automation サービス ガイド』の「ホストの設定」を参照してください。

RDP を使用するステップ

1. Desktop Automation サービスをリモート デバイスにインストールし、設定します。「Desktop Automation サービス」を参照してください。
2. サービスがシングル ユーザー モードで動作するように設定し、トークンを指定します。
3. ロボットで、「RDP ログイン」ステップを挿入します。
4. RDP 接続に必要なすべてのオプションを指定します。
   RDP ログイン ステップは、接続が確立されるまで待機してからロボットの実行を続行します。リモート接続に失敗すると、エラー メッセージが表示されます。
5. 接続しているシステムがユーザーのログイン時に別の画面を表示するように設定されている場合は、「RDP ログイン」ステップの「ログオン ダイアログの終了タイムアウト」で、ログイン中に別の画面が閉じるまで待機する秒数を設定します。
   この画面を閉じないと、アクションが失敗することがあります。

RDP の維持
Windows では、長時間待機状態にある RDP セッションが切断または終了するように設定されている場合があります。切断を回避するため、ロボットの RDP 接続サービスは数秒に 1 回、ダミーのキーストローク シグナルをリモート セッションに送信します。

キーストロークの間隔のデフォルト値は 30 秒です。

時間間隔を変更するには、「RDP ログイン」ステップの [キープアウェイク キーを押す間隔] オプションで間隔を秒単位で指定します。

このオプションをオフにするには、ゼロ (0) を指定します。

特定のデバイスへの RDP 接続は同時に 1 つしか存在できませんが、異なるデバイスへの RDP 接続は複数併存できます。同じデバイスに新たに RDP を接続すると、このデバイス上の既存の RDP 接続はすべて閉じます。

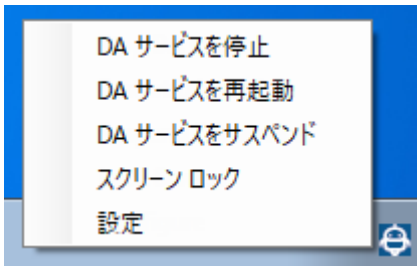
デバイスが静的リファレンスの場合は、RDP 接続が確立された後に、レコーダー ビューに使用可能なアプリケーションが表示されます。

デバイスが動的リファレンスの場合は、リモート デバイスを自動化するために「デバイスに接続」が必要になります。タイムアウトの設定と再試行を実行し、RDP 接続の確立を確認することをお勧めします。このステップを実行すると、レコーダー ビューに使用可能なアプリケーションが表示されます。

RDP 接続が確立されていることを確認するには、Windows のタスクマネージャで `kapowlock` プロセスを探します。プロセスが、ロボットを実行しているコンピュータ上のプロセスのリストに存在する場合、接続はアクティブです。

Desktop Automation サービスの管理

Desktop Automation サービスのショートカット メニューをクリックして、次のオプションにアクセスします。



これらのオプションを使用して、リモート コンピューターで実行されている Desktop Automation サービスを管理します。

- **DA** サービスを停止: サービスを停止します。これによりリモート デバイスは使用できなくなります。Desktop Automation サービスが実行されているコンピュータは、Management Console のリストから削除されます。
- **DA** サービスを再起動: サービスを停止してから、起動します。ロボットまたは Design Studio ではデバイスとの接続が失われるため、復元するにはリロードする必要があります。
- **DA** サービスをサスペンド: デバイスをサスペンドします。サービスをサスペンドすると、Management Console ではサスペンド状態と表示されます。サービス操作を復元するには、ユーザーまたは管理者はデバイスで Desktop Automation サービスを手動で起動する必要があります。

  サスペンド状態ではロボットは DA サービスを利用できませんが、状態情報は ping メカニズムを介して Management Console に送信され、**[管理]** > **[デバイス]** セクションにデバイスが表示されます。このコマンドは、何らかの理由でサービスまたはサービスを実行しているコンピュータの設定を変更する必要が生じた場合に役立ちます。
- スクリーン ロック: リモート デバイスのスクリーンをロックします。このオプションを使用する前に、Desktop Automation サービスをインストールして設定する必要があります。『Desktop Automation サービス ガイド』の「スクリーン ロックの使用」を参照してください。
- 設定: Desktop Automation サービスの設定ダイアログ ボックスを開きます。『Kofax RPA Desktop Automation サービス ガイド』を参照してください。

第 6 章

Management Console

Management Console は、Kofax RPA の監視機能および管理機能を提供する Web ベースのインターフェイスを備えたアプリケーションです。

Management Console では、特に、次のことができます。

- リポジトリを使用した共同作業および共有の有効化
- ユーザーが表示できるメッセージを投稿します。
- ユーザー、ユーザー ロール、権限の一元管理
- ライセンス情報とパスワードの管理
- Design Studio からリポジトリへのロボットの展開
- リポジトリからのロボットのスケジュール設定
- 抽出したデータの参照とエクスポート
- 本番環境の結果およびエラーの詳細ログへのアクセス
- 複数の RoboServer のクラスタの設定
- 他の Kofax 製品やサービスへの接続

概要

Management Console は、Web ベースのユーザー インターフェイスを備えたアプリケーションで、ネットワーク上の任意のコンピュータから簡単にアクセスできます。Management Console に接続するためのアドレスは、コンピュータ名とポート (`http://computername:port` など) で構成されます。

Management Console は、2 つの方法で展開できます。

- 組み込みモード: インストール後にデモやテストを目的として Management Console をすばやく起動するための方法。組み込みモードについては、以下の組み込み Management Console の構成を参照してください。
- スタンドアロン モード: 本番環境の Tomcat に展開された Management Console のスタンドアロン バージョン。このモードでは、Kofax RPA によって、Docker 展開用のツールも提供されます。スタンドアロンの Management Console の詳細は、『Kofax RPA 管理者ガイド』の「Tomcat 管理コンソール」を参照してください。

すべての RoboServer は Management Console に自動的に登録される必要があります。そのため、RoboServer を起動するときに Management Console の URL と共有シークレット、およびクラスタ名を指定する必要があります (コマンド ライン、または RoboServer 設定アプリケーションの **[一般]** タブを使用します)。詳細については、「RoboServer の開始 」を参照してください。

ライセンス キーによっては、高可用性などの一部の機能が利用できないことがあります。

組み込み Management Console の構成

組み込みモードでは、Management Console と RoboServer が同時に起動し、デフォルトで、Web サーバーはポート 50080 でデータの送受信を行います。Web インターフェイス ポートおよびその他のパラメータは、RoboServer 設定アプリケーションの [**Management Console**] タブで設定します。

プロトコルとポート

別のポートで HTTP および HTTPS を介してアクセスできるように Web サーバーを設定することができます。プロトコルが有効になっている場合は、ポート番号を選択する必要があります。デフォルトのポートは、50080 (HTTP) および 50443 (HTTPS) です。

HTTPS を有効にするには、JKS 形式のサーバー証明書を `webserver.keystore` というファイルに保存する必要があります。

また、埋め込み Management Console に JDBC ドライバーをアップロードすることができるユーザーを制限できます。選択可能なオプションは、「許可されていません」(すべてのユーザーは JDBC ドライバーのアップロードが不可能)、「localhost の管理者」(管理者はローカルのマシンから Management Console にアクセスする場合にドライバーをアップロード可能)、そして、「すべてのホストの管理者」(管理者は常に JDBC ドライバーをアップロード可能) です。

ユーザー管理

Management Console を使用する理由の 1 つとして、ロボットの実行を調整して、多くのクライアントが一般にアクセスできるようにすることが挙げられます。他のマシンから Management Console への不正アクセスに対する潜在的なセキュリティ リスクを軽減するため、組み込みモードではデフォルトでユーザー管理が有効化されており、デフォルトの admin スーパーユーザー パスワードを使用できます (ユーザー名 - `admin`、パスワード - `admin`)。詳細については、ユーザーおよびグループを参照してください。

Management Console に初めてログインする際は、デフォルトの `admin` ユーザー パスワードを使用します。ログイン後に、Management Console の 「管理」 の下にある [ユーザーおよびグループ] ページでパスワードを変更し、他のユーザーとグループを作成します。ロボットを Design Studio から Management Console に発行する場合、およびブラウザから Web インターフェイスにアクセスする場合に、これらの資格情報を使用します。`admin` ユーザーのパスワードを変更するには、次の手順を実行します。

1. 左側のペインの [管理] 項目を展開し、[ユーザーおよびグループ] をクリックします。
2. [ユーザー] タブで、`admin` ユーザーを選択し、 をクリックします。
3. 新しいパスワードを入力し、確認のためにもう一度入力して、[**OK**] をクリックします。
   詳細については、ユーザーおよびグループを参照してください。

メッセージの投稿

このオプションは、管理者が利用できます。

Management Console のホームメニュー オプションを使用して、管理メッセージを投稿および管理します。

1. 新しいメッセージを開始するには、**[メッセージの投稿]** ボタンをクリックします。
2. メッセージを入力します。
   テキスト入力フィールドのサイズを変更するには、矢印が表示されるまでポインタを下端に置いたままにしてから、ポインタを上下にドラッグします。
3. 優先度を選択します。
   メッセージを投稿すると、メッセージとともにこの優先度に対応する色が表示されます。
4. (オプション) 有効期限を入力するか、ポップアップ カレンダーから有効期限を選択します。
   このフィールドを空白のままにすると、メッセージは手動で削除するまで残ります。
5. 投稿をクリックします。

有効期限が切れる前にメッセージを削除するには、メッセージを選択してゴミ箱アイコンをクリックします。

Management Console の開始

本番環境では、Management Console の Tomcat バージョンをセットアップすることをお勧めします。詳細については、『Kofax RPA 管理者ガイド』を参照してください。

組み込みモードで、次のように Management Console を起動します。

Windows

スタート メニューの [**Management Console** を起動] の項目を使用します。必要なパラメータはすべて、デフォルトのスーパーユーザー名とパスワードを使用して組み込み Management Console を起動するスタート メニューのショートカットに含まれています。

コマンド ラインから Management Console を開始するには、インストール フォルダの `bin` サブフォルダで次のコマンドを実行します。

```
RoboServer.exe -p 50000 -MC -mcUrl http://ServerName:port
```

Linux

コマンド ラインから Management Console を開始します。これは RoboServer プログラムの一部で、インストール ディレクトリの下の `bin` ディレクトリにあります。

```
$./RoboServer -p 50000 -MC -mcUrl http://ServerName:port
```

これにより、ポート 50000 でソケットをリッスンする RoboServer が起動し、設定したポートの Web インターフェイスを介して Management Console 機能が提供されます (ポートは、**[Management Console]** タブの RoboServer 設定アプリケーションで設定します)。詳細については、組み込み Management Console の構成を参照してください。

コマンド ラインを使用して RoboServer を開始し、Management Console に登録することもできます。

```
RoboServer.exe -p 50000 -MC  -mcUrl http://ServerName:port -ss <MC Shared Secret> -cl
 "Production"
```

これにより、ポート `50000` で RoboServer が開始され、指定した共有シークレットを使用して `Production` クラスタの下の `ServerName:port` の Management Console に登録されます。

Management Console の起動後に、ブラウザで開きます。Windows では、スタートメニューにある Management Console の項目をクリックします。すべてのプラットフォームで、ブラウザを開き、`http://ServerName:port/` に移動します。初めての起動時は、デフォルトの資格情報を使用してログインし、ライセンス条項に同意して、ライセンス キーを含むライセンス情報を入力します。後でライセンス情報を変更する必要がある場合は、[管理] > [ライセンス] で変更できます。

Management Console をセットアップした後に、パラメータ `-p 50000` を使用して、必要な数の RoboServer を起動します。この RoboServer のみを [RoboServer] セクションに追加する必要があります。

ユーザー インターフェース

Management Console には次のメニュー オプションがあります。

ホーム

このメニュー オプションを使用して、メッセージを表示および投稿します。

スケジュール

このメニューを使用して、スケジュールを作成および管理します。スケジュールは、1 つ以上のロボットを実行するための手順で、通常、指定された時点で繰り返されます。スケジュールを使用して、設定済みのサーバーにロボットを渡すことで実行される、ロボット実行のタイミングに関するプランを設計します。

リポジトリ

このメニューを使用して、作業オブジェクトの管理、マッピングによる他のコンポーネントへの接続、ロボット ファイル システムの追加などを行います。ロボット、タイプの定義、およびリソースは、Design Studio から Management Console のリポジトリにアップロードするか、Management Console の Web インターフェイスを介して手動でアップロードできます。アップロードしたロボットは、スケジュール、Kapplet の一部として実行したり、Kofax RPA Java や C# API を使用してロボットを実行するクライアント コードを介して実行したりできます。また、API を使用して、リポジトリに対してプログラムでクエリを実行したり、リポジトリを更新したりできます。

データ ビュー

このメニューを使用して、ロボットがデータベースに保存したデータの表示や、このデータの Excel、XML、または CSV へのエクスポートを行います。

ログ ビュー

このメニューを使用して、RoboServer、スケジュール、ロボット、およびその他のログの確認を行います。

管理

このメニューを使用して、管理タスクに関連する Management Console の設定を行います。また、このメニューでは、RoboServer のクラスタとその設定に加えて、プロジェクトと権限を管理することができます。また、このメニューでは、共有シークレットの生成、ライセンスの設定、バックアップやプロジェクトの作成/復元を行うこともできます。

設定

このメニューを使用して、サーバーとデータベースの設定、Process Discovery、KTA など、Management Console のその他の設定を構成します。

Management Console のほとんどのセクションでは、特定のタイプのアイテムがテーブルに表示されます。表示できるアイテムが多数ある場合は、複数のページに分割されます。[ページごとのアイテム] 設定を使用して、同時に表示するアイテムの数を選択できます。この数はブラウザ ウィンドウの高さに自動的に適合しないため、テーブルの下の空白スペースに必ずしも最後のアイテムが表示されるとは限りません。ページに表示されている列を選択するには、右側の をクリックして、リストから列を選択します。

表は、並び替えとフィルタリングをサポートしています。列の見出しをクリックすると、その列で並び替えできます。再び同じ列の見出しをクリックすると、並び替え順を反転できます。並べ替えは、ページごとではなく、テーブル全体に対して実行されます。「ページ」が複数存在する場合、並べ替え順を変更するとまったく異なる行のセットが表示されることがあります。開いているページに応じて、[フィルタ] フィールドに名前を入力するか、最初に [フィルタ] をクリックしてからフィルタ基準とする項目を選択し、その名前を入力して、リストをフィルタリングできます。詳細については フィルタリング を参照してください。

Management Console の各セクションの下に表示されるウィンドウはサイズ変更できます。デフォルト設定を復元するには、**[プロファイル]** > **[設定]** > **[すべての設定をリセット]** に移動します。

ユーザー メニュー

右上隅に、次の機能を含むユーザー メニューがあります。

- プロファイル
  現在のユーザーに関する一般的な情報が含まれます。
- 設定
  ユーザー インターフェイスの言語設定と、すべてのカスタム設定をデフォルト値にリセットするオプションが含まれます。ユーザー インターフェイスのローカル言語設定は、管理者が割り当てた言語設定よりも優先されます。

  「ログ ビュー」を除くすべてのテーブルでの日付の表示形式は、ブラウザの設定によって異なります。英語に設定されている場合、ブラウザの言語設定を使用して英語の方言が決定されます。たとえば、言語リストの一番上に [英語 (英国)] と表示されている場合、日付はイギリス英語の標準の日付形式に従って書式設定されます。ブラウザの設定で何も指定がない場合は、デフォルトで英語 (アメリカ合衆国) が使用されます。
  「ログ ビュー」セクションの日付設定については、RoboServer を参照してください。

- パスワードの変更
  クリックすると、パスワードを変更できます。

- ユーザー **API** トークン
- 製品について
  このバージョンの Management Console、サーバー時間、およびローカル時間に関する一般情報が含まれます。
- ログアウト
  クリックすると、現在のユーザー プロファイルからログアウトします。

また、ユーザー メニューの左側にはヘルプ ボタンがあり、『Kofax RPA のヘルプ』を開くことができます。

ホーム

ホームページで、管理者によって投稿されたメッセージを表示します。

スケジュール

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、開発者、および VCS サービス ユーザー。

Management Console でスケジュールを管理するには、[スケジュール] メニューを使用します。スケジュールは、ロボットの選択とそれらを実行するための計画 (ジョブ) を示します。スケジュールの実行とは、選択したロボットを (並列で、または順番に) 実行することを意味し、必要に応じてロボットを事前実行または事後実行することができます。

連続して実行されるスケジュール間の間隔を選択する場合は、前のスケジュールの実行中に次のスケジュールされた実行が発生すると、次にスケジュールされた実行は開始されないことに注意してください。

[スケジュール] メニューの上部にある [プロジェクト] ドロップダウン リストで、表示するスケジュールを含むプロジェクトを選択します。各スケジュールに関する情報の表示方法を次のように変更できます。

- [フィルタ] テキスト フィールドにスケジュール名またはスケジュール名の一部を入力して、テーブル内のスケジュールのリストをフィルタリングすることができます。詳細については、フィルタリングを参照してください。
- 右側の メニュー アイコンを使用して、スケジュールに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、スケジュールごとに次のテーブル列が表示されます。

列	説明
アクティブ	アクティブである場合、スケジュールは計画どおりに実行されます。次のような複数の理由により、スケジュールを非アクティブにする必要がある場合があります。 • スケジュールによって実行されている機能が現在不要である。 • ロボットでエラーが見つかったため、次回の実行前にエラーを修正したいと考えている。 • 実行するたびに手動でスケジュールをトリガーする。これは、準備タスクやクリーンアップ タスクなどの一部のロボットやスケジュールに適しています。
開始/停止	アイコン または をクリックすると、スケジュールを開始または停止できます。 アイコンは、スケジュールのステータス (実行中または停止) に基づいて変化します。 複数のスケジュールがアクティブ化されている場合、開始/停止アイコンには、選択した各スケジュールのステータスが表示されます。
名前	スケジュールの名前。
プロジェクト名	スケジュールが属するプロジェクトの名前 (すべてのプロジェクトを表示する場合に役立ちます)。
ジョブ カウント	合計およびアクティブなジョブの総計。すべてのジョブがアクティブな場合は、アクティブなジョブの数が一覧表示されます。3 つのジョブのうち 2 つがアクティブな場合は、2 (3) と表示されます。
次の実行	次回のスケジュール実行が計画されている時間。
前の実行	スケジュールが前回実行された時間。
間隔	同じスケジュールの連続した 2 つの実行における計画された間隔。
合計実行数	スケジュールが実行された回数。
エラー	スケジュールの最終実行中のスケジュール エラーの数。スケジュール エラーには、ロボットのエラーは含まれません。 エラーとロボット エラーを表示するには、[設定] -> [RoboServer ログ データベース]の [一般] タブでログ データベースを使用して設定されていること、および [RoboServer] -> [設定] -> [ログ] の RoboServer でデータベース ログが有効になっていることを確認します。
ロボットのエラー	このスケジュールによってロボット実行で発生したロボットのエラーの数。 スケジュールされたロボットで RoboServer が動作しているときにロボットによってエラーが報告された場合、RoboServer はエラーからの回復を試みるあいだ、スケジューラと RoboServer は停止されません。
オプションの列	
合計ジョブ数	スケジュールにおけるジョブの合計数。
アクティブなジョブ	スケジュールにおけるアクティブなジョブの合計数。
作成者	スケジュールを作成したユーザーの名前。

列	説明
変更者	スケジュールを最後に変更したユーザーの名前。
コミット メッセージ	コミットについて説明した概要。
リビジョン番号	スケジュールのリビジョンの回数。
クラスタ	スケジュールが実行されるクラスタ。
優先度	スケジュール内の他のジョブに対するこのジョブの実行の優先度。

新しいスケジュールの作成

1. 新しいスケジュールを作成するには、左上隅の + 記号をクリックします。
   「プロジェクトを選択」ダイアログ ボックスが表示されます。
2. プロジェクトを選択して [**OK**] をクリックします。
   複数の新しいタブが開きます。
3. [基本] タブには、スケジュールの設定に必要なすべての項目が含まれます。
   - 名前: スケジュールの名前。
   - アクティブ: このスケジュールをアクティブにするには、このオプションを選択します。
   - シンプル/Cron: スケジュールのタイム プランを定義する方法を選択します。
   - 繰り返し間隔: シンプル スケジュールにのみ利用できます。スケジュールの 2 回にわたる連続した実行間の、希望の時間の間隔。「1 分」、「3 時間」などの単位付きの整数で入力します。
   - パターン: Cron スケジュール専用。スケジュールを実行するタイミングを定義するパターン。
   - サーバーのタイム ゾーン: cron スケジュール専用。デフォルトでは、サーバーのタイム ゾーンがスケジュールの実行に使用されます。このオプションを使用すると、特定の UTC タイム ゾーンを選択することもできます。
   - 開始時間: スケジュールを開始するローカルの時間。
   - 日付を選択: スケジュールを開始する必要がある日付。
   - ジョブの優先順位: 同じ必要なリソースのためにキューに留まっている他のジョブに対する、このジョブの実行の優先度。この優先度は、スケジュール内のすべての口ボット ジョブに適用されます。詳細については、「ジョブの優先順位の設定」を参照してください。
   - ジョブのタイムアウト: ジョブがキュー内に留まる時間のタイムアウト。タイムアウトに達するまでにジョブにリソースを割り当てて実行できない場合、ジョブはキューから削除されます。このタイムアウトは、スケジュール内のすべての口ボット ジョブに適用されます。詳細については、「ジョブの優先順位の設定」を参照してください。
   - 前処理口ボット: スケジュールの開始前に実行される口ボットの名前。
   - 後処理口ボット: スケジュールされた他のすべての口ボットの実行後に実行される口ボットの名前。
   - クラスタで実行: このスケジュールを実行するクラスタの名前。
   - コミット メッセージ: コミットについて説明した概要。
4. [詳細] タブでは、ランタイム制約を設定できます。
   - 実行時間の制限: スケジュールの各口ボットの最大実行時間を設定します。口ボットがこの期間にわたって実行された場合、サービスが停止し、エラーがログに記録されます。デフォルトの値は `-1` で、これは時間に制限がないことを示します。

- 抽出された値の制限: 各ロボットが出力できる値の最大数を選択します。ロボットがこの数を超える値を生成すると、サーバーはロボットを停止し、エラーがログに記録されます。デフォルトの値は `-1` で、これは数に制限がないことを示します。

  制限はスケジュール全体ではなく、各ロボットの実行に個別に適用されます。制限に達するとロボットは停止しますが、スケジュールは続行されます。

- 電子メール通知を使用: チェックすると、ロボットに不具合が発生したとき毎回、電子メールを受信します。スケジュールにおいて複数のロボットに不具合が発生すると、スケジュールが実行される度に各ロボットにつき 1 通のメールが送信されます。電子メール通知は、SMTP サーバーを設定し、[電子メール アドレス] フィールドに目的の電子メール アドレスを入力した場合にのみ機能します。
- 電子メール アドレス: 通知が送信される、コンマ区切りの電子メールのリスト。最初にリストされているアドレスが、送信者および受信者のアドレス両方として使用されます。フィールドには 255 文字まで入力できます。このフィールドの制限文字数を超えると、スケジュールは保存されません。

5. [スケジュールされたジョブ] タブには、スケジュールがトリガーされた場合に実行されるジョブのリストが表示されます。
   - ジョブを順次実行: [スケジュールされたジョブ] タブにリストされた順序でロボットを実行する場合に選択します。
   - ジョブ名: ジョブの表示名。ロボットのアップロード時に表示名の属性で指定した名前です。
   - アクティブ: アクティブの場合、スケジュールが実行されるときにジョブが実行されます。以下のような場合には、スケジュール内の単一のジョブを非アクティブの状態にする必要があります。
     - ジョブによって実行される機能が現在必要ない場合。
     - ロボットでエラーが検出され、これらのエラーを修正する前にスケジュールが実行されないようにする場合。
     - 実行するたびに毎回手動でジョブをトリガする場合。

     スケジュールへのジョブの追加

     スケジュールにジョブを追加するには、次の手順を実行します。

     a. 左上隅の + 記号をクリックします。

        ジョブの生成をガイドするダイアログ ボックスが表示されます。

     b. [ジョブ タイプの選択] ステップで、オプションを選択します。利用できるオプションには、以下が含まれます。
        - シングル ロボット
        - 名前を基準としたロボットのグループ

ジョブ タイプ	説明
シングル ロボット	シングル ロボットを実行するジョブを追加します。複数のロボットを 1 つずつ追加します。入力をロボットに渡すために、必要なパラメータを指定できます。
ロボットのグループ	パス名が指定した基準に一致する任意の数のロボットを実行するジョブを追加します。ロボット グループは、スケジュールを開始するたびに評価されるため、選択した基準と一致する新しいロボットが自動的に実行されます。

スケジュールに対するアクション

ロボットの ⋮ コンテキスト メニューをクリックして、次のアクションから選択します。

- コピーの作成: スケジュールを作成するためのウィザードを開きます。新しい名前を指定し、スケジュールに変更を加えて、[**OK**] をクリックします。
- 編集: パラメータが入力済みの、スケジュール作成と同じウィザードが開きます。
- 表示: ログ ビューの表示内容と同一の、スケジュール実行に関連する情報を表示するために複数のビュー機能を使用することができます。

[スケジュール] ページで、次のアクションを実行します。

- 1 つまたは複数のスケジュールを選択し、**[アクティブ化 & 非アクティブ化]** アイコン をクリックして、スケジュールをオン/オフにします。
- 1 つまたは複数のスケジュールを選択し、**[実行/停止]** アイコン をクリックして、スケジュールを手動で実行または停止にします。スケジュールを停止すると、実行しているすべてのロボットが可能な限りすぐに停止します。すべてのロボットが実行を停止するまで、スケジュールは「実行中」と表示されます。

  すでに実行中のスケジュールを実行しようとすると、実行中のスケジュール名を含むエラーが表示されます。
- 1 つまたは複数のスケジュールを選択し、**[スケジュールを削除]** アイコン をクリックして、スケジュールを削除します。
- スケジュールを選択し、該当する **[開始/停止]** アイコン または をクリックして、スケジュールを手動で開始または停止します。

別のスケジュール作成方法

[ロボット] セクションからスケジュールを作成することもできます。この操作を行うには、必要なロボットのコンテキスト メニューをクリックし、**[スケジュールの作成]** を選択します。

シングル ロボットの追加

シングル ロボットの追加のウィザードには、1 ~ 4 つのステップが含まれます。これは、選択したロボットによって異なります。

1. [ロボットの選択] ステップで表示名を指定して、ドロップダウン リストからロボットを選択します。
2. ロボットが使用するすべてのスクリプトとタイプがすでにアップロードされていて、ロボットに入力変数がない場合は、[次へ] をクリックしてから [保存] をクリックします。

   このロボットに関連付けられているすべてのスクリプトまたはタイプ ファイル、あるいはその両方が Management Console リポジトリに存在することを確認してください。存在しない場合は、必ずすぐにアップロードしてください。

   スケジュールに追加されたロボットでは、入力タイプのデフォルト値のみがアップロードされます。Design Studio で行った、変数内のデフォルト タイプの値に対する変更は、Design Studio でのみ使用できます。変数タイプの値を確認し、ロボットをスケジュールにアップロードする場合に、必要に応じて変更します。
3. このスケジュールの一部として実行する際に使用するロボット入力を設定します。

4. 属性がバイナリ、画像、PDF、Excel タイプである場合は、ドロップダウン リストを使用することですでにアップロードされているリソースを選択することができます。または、[アップロード] をクリックしてアップロードすることも可能です。

   必須の属性は、赤でマークされています。
5. 必須のフィールドにすべて入力して、[保存] をクリックします。

   さらに、⋮ コンテキスト メニューからジョブに対して次のアクションを実行できます。
   - [コピーの作成]: 既存のジョブのコピーを作成します。
   - [編集]: ジョブを編集します。
   - [非アクティブ化]/[アクティブ化]: ジョブをオフ/オンにします。

   ロボットの実行順序を並べ替えるには、ロボットを順序内の新しい位置にドラッグします。

   ジョブを削除するには、テーブルでジョブを選択し、左上隅の 🗑 ごみ箱アイコンをクリックします。

ロボットのグループの追加

特定の基準に一致するパス名を持ったすべてのロボットを実行する単一のジョブを追加するように選択できます。この基準はスケジュールの実行時に評価されるため、ジョブの生成後にアップロードしたロボットは、設定した基準を満たす場合に追加されます。

この基準は実行時に評価されるため、スニペット/タイプの欠落などのエラーは、ログ ビューのスケジュール実行ログにエラーとして記録されます。入力変数を使用するロボットが実行される場合も、条件に一致するため、同じことが言えます。

ℹ スケジュールに追加されたロボットでは、入力タイプのデフォルト値のみがアップロードされます。Design Studio で行った、変数内のデフォルト タイプの値に対する変更は、Design Studio でのみ使用できます。変数タイプの値を確認し、ロボットをスケジュールにアップロードする場合に、必要に応じて変更します。

1. [基準の設定] ステップで表示名を指定して、一致基準を次の中から選択します。「ロボット パスに次が含まれる」、「ロボット パスが次のパターンと一致する」、または「ロボット パスが次で開始する」

   下部のリストには、選択した基準に一致する 1 つ以上のロボットが表示されます。[表示名] フィールドは、編集を行わない限り、条件の選択に応じて変更されます。ただし、編集を開始するとこの自動命名は無効になります。
2. 照合するパターンを指定します。
3. [次へ] をクリックします。

   設定した基準に一致するロボットがない場合でも、一致するロボットを後でアップロードできるため、[次へ] をクリックします。

   ロボットの数が多い場合は、選択した基準に一致するロボットのリストが更新されるまでに時間がかかることがあります。

   ℹ ジョブを順次実行するように設定したスケジュールにこのジョブ タイプを追加すると、基準に一致するロボットのグループ内での順序を制御できなくなります。ジョブは順次実行されますが、1 つのグループに属するロボットは常に同時に実行されます。

Cron スケジュール

「cron」 スケジュールは、6 つまたは 7 つのサブフィールドで構成され、これらのサブフィールドは、ロボットを実行するタイミングを記述します。サブフィールドを 1 つまたは複数のスペースで区切って、以下の項目を記述します。

1. 秒
2. 分
3. 時
4. 日
5. 月
6. 曜日
7. 年 (オプション フィールド)

Cron スケジュールの完全形の例は「0 0 12 ? * WED」で、その意味は「毎水曜日の午後 12:00」です。

個々のサブフィールドには範囲やリストを含めることができます。たとえば、先の例の曜日フィールド (つまり、"WED") は "MON-FRI"、"MON, WED, FRI"、または "MON-WED,SAT" などにも置き換えることができます。

「このフィールドで可能なすべての値」を意味するワイルドカード ("*" 文字) を使用することもできます。たとえば、先の例の月フィールド内の "*" 文字は、「毎月」の意味になります。曜日フィールド内の "*" 文字は、「毎日」という意味になります。

各フィールドに対する一連の有効値は、次のようになります。

- 秒：0 ~ 59 の数。
- 分：0 ~ 59 の数。
- 時：0 ~ 23 の数。
- 日：1 ~ 31 の数 (該当する月の日数を考慮)。
- 月：1 ~ 12 の数、または文字列
  JAN、FEB、MAR、APR、MAY、JUN、JUL、AUG、SEP、OCT、NOV、および DEC。
- 曜日：1 ~ 7 の数 (1 は日曜日)、または文字列 SUN、MON、TUE、WED、THU、FRI、および SAT。
- 年：数。

さらに、以下のような特殊文字も使用できます。

	定義
/	値にインクリメントを指定します。たとえば、"0/15" を分フィールドに入力すると、「0 分から開始して 15 分毎」の意味になります。分フィールドに「3/20」と入力すると、「1 時間のうち、3 分から開始される 20 分毎」という意味になります。つまり、「3、23、43」を指定した場合と同じことになります。
?	日フィールドおよび曜日フィールドにのみ使用でき、「指定値なし」を意味します。これら 2 つのフィールドのうち一方に何かを指定する必要があり、他方には不要な場合に便利です。詳細は、下の各例を参照してください。

	定義
L	日フィールドおよび曜日フィールドに限り使用できます。"L" は "last" の省略形ですが、その意味は 2 つのフィールド間で異なります。 たとえば、日フィールドの「L」は「月の最終日」を意味し、1 月では 31 日、うるう年以外の年の 2 月では 28 日になります。 「L」を曜日フィールドで単独で使用した場合、「7」または「SAT」を意味しますが、曜日フィールド内で別の値の後に使用した場合、「月の最後の xxx 日」を意味します。たとえば、「6L」または「FRIL」はどちらも「月の最後の金曜日」を意味します。 「L」という文字を使用する場合は、混乱を招く原因となるため、リストや値の範囲を指定しないようにしてください。
#	曜日フィールドに使用して、その月の「第 n 番目の」X 曜日を指定します。たとえば、"6#3" または "FRI#3" の値の意味は、「その月の第 3 金曜日」です。
W	その日に一番近い平日 (月曜 ~ 金曜) を指定します。例として、日フィールドの値に「15W」と指定した場合、その意味は、「その月の 15 日に一番近い平日」になります。

例

Cron スケジュール	説明
0 0 12 * * ?	毎日午後 12 時 (正午) に作動
0 15 10 ? * *	毎日午前 10:15 に作動
0 15 10 * * ?	毎日午前 10:15 に作動
0 15 10 * * ? *	毎日午前 10:15 に作動
0 15 10 * * ? 2005	2005 年中は毎日午前 10:15 に作動
0 * 14 * * ?	毎日午後 2 時から午後 2:59 までの間、毎分作動
0 0/5 14 * * ?	毎日午後 2 時から午後 2:55 までの間、5 分毎に作動
0 0/5 14,18 * * ?	午後 2 時から午後 2:55 までの間、5 分毎に作動 および、毎日午後 6 時から午後 6:55 までの間、5 分毎に作動
0 0-5 14 * * ?	毎日午後 2 時から午後 2:05 までの間、毎分作動
0 10,44 14 ? 3 WED	3 月の毎水曜日の午後 2:10 および午後 2:44 に作動。
0 15 10 ? * MON-FRI	毎週月曜日、火曜日、水曜日、木曜日、および金曜日の午前 10:15 に作動
0 15 10 15 * ?	毎月 15 日の午前 10:15 に作動
0 15 10 L * ?	毎月最終日の午前 10:15 に作動
0 15 10 ? * 6L	毎月最終金曜日の午前 10:15 に作動
0 15 10 ? * 6L	毎月最終金曜日の午前 10:15 に作動
0 15 10 ? * 6L 2002-2005	2002、2003、2004、および 2005 年の毎月最終金曜日の午前 10:15 に作動
0 15 10 ? * 6#3	毎月第 3 金曜日の午前 10:15 に作動

ジョブの優先順位の設定

Desktop Automation サービス、ライセンス ユニット、RoboServer 実行スロットなどの必要なリソースが利用可能になったときに、ジョブがキューで実行されるようにスケジュールを設定します。

スケジュールを作成するには、「スケジュール」を参照してください。

タスク ビューでキューイングのステータスを表示し、ログ ビューの「タスク メッセージ」ログを使用して履歴を表示します。

優先度の処理

ジョブの設定方法によって、処理の優先度が決まります。

- より高い優先度でスケジュールされたジョブは、より低い優先度のジョブよりも前に実行されます。
- 優先度の高いジョブは、必要なリソースに優先的にアクセスできます。
- 数分間キューに入れられた優先度の高いジョブは、キューに入ったばかりの別の優先度の高いジョブよりも前に実行されます。
- 「ジョブを順次実行」を選択した場合は、以下の内容が適用されます。
  - シングル ロボットを含むスケジュールを作成すると、スケジュール内の各ロボットの後にタイムアウトが発生します。
  - ロボットのグループを含むスケジュールを作成すると、ロボットは同時に実行されます。タイムアウトが発生すると、すべてのロボットがタイムアウトし、スケジュールもタイムアウトします。
- 各スケジュールには設定可能なタイムアウトがあり、タイムアウトはスケジュール内のすべてのロボット ジョブに適用されます。
  - タイムアウトに達する前にジョブが実行されなかった場合、ジョブはキューから削除されます。
  - 300 秒や 600 秒などのタイムアウトが設定された 2 つのジョブがあり、それらが同時に実行されるようにスケジュールされている場合は、タイムアウトが短いジョブがタイムアウトが長いジョブの前に実行されます。
  - スケジュールの競合が発生した場合、優先度は同じであってもタイムアウトが異なるジョブは、キュー内で費やした時間に基づいて優先度が高くなります。ジョブがキュー内に長く留まるほど、同じ優先度を持つ別のジョブよりも先に実行される可能性が高くなりますが、タイムアウトは長くなります。

優先度の低いジョブが長時間キューに留まっており、タイムアウトに近づいている場合、それらのジョブをより高い優先度に変更することはできません。優先度の低いジョブは、実行スロットが利用できない場合、キューから削除されます。

スケジュールを設定する

優先度とタイムアウトを決定するパラメータを使用して、各スケジュールを設定します。

1. スケジュールの作成中に、**[ジョブの優先順位]** パラメータを設定します。
   Management Console では、次のような優先度の低いジョブから高いジョブまでを利用できます。
   - 最小
   - 低
   - 中
   - 高
   - 最大
2. **[ジョブのタイムアウト]** パラメータを設定して、スロットとリソースが使用できない場合にジョブのキューイングを停止するタイミングを決定します。

リソースの予約と実行

Management Console はキュー内のジョブを処理する準備ができると、次のシーケンスを実行します。

1. 最も優先度の高いジョブから開始します。
2. 最初のジョブを実行するための RoboServer が利用可能かどうかを判断します。
3. ライセンスの分配と RoboServer 閾値を考慮します。
   ライセンスと閾値のパラメータについては、「一般」を参照してください。

   ロボットは、RoboServer に送信された場合にのみ CRE ライセンスを使用します。キューに留まっている間は、ロボットがライセンスを予約したり使用したりすることはありません。

4. ジョブで Desktop Automation サービス マシンでのリソース予約が必要である場合、Management Console は予約を作成します。
5. すべての条件を満たすと、ジョブは実行のために RoboServer に送信されます。

リポジトリ

Management Console は、ロボット、タイプ、スニペット、リソース、デバイス マッピング、OAuth 資格情報、およびその他の作業オブジェクトのリポジトリを保持します。[リポジトリ] メニューは、オブジェクトの管理に役立ちます。

ロボット
ロボットの管理、API: を使用してロボットを実行するコードの生成、REST サービスとしてロボットを実行する方法、SOAP リクエストの開始、ロボットのドキュメント ドキュメントの作成について説明します。

タイプ
タイプをアップロード、削除、および管理する方法について説明します。

スニペット
スニペットをアップロードおよび削除する方法について説明します。

リソース
既存のリソースを管理し、新しいリソースをアップロードする方法について説明します。

デバイス マッピング
ロボット用の既存のマッピング済みオートメーション デバイスと、新しいデバイスを作成する方法について説明します。

データベース マッピング
既存のデータベース マッピングと、新しいデータベース マッピングを作成する方法について説明します。

電子メール トリガー
新しい電子メール トリガーを設定する方法について説明します。

トリガー マッピング
さまざまなタイプのトリガー マッピングとその使用方法について説明します。

OAuth
新しい OAuth アプリケーションおよびユーザーを追加および管理する方法について説明します。

パスワード ストア
パスワード リポジトリに関するすべての情報を確認できます。

CyberArk
CyberArk と、外部パスワード マネージャーとして CyberArk を使用する方法について説明します。

ロボット ファイル システム
ロボット ファイル システムの機能と、新しいファイル システムを作成および設定する方法について説明します。

ロボット

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、開発者、API、DAS クライアント、VCS サービス ユーザー、および Kapplet サービス ユーザー。

[ロボット] セクションには、Management Console リポジトリにアップロードされたロボットが一覧表示され、プロジェクトごとにリポジトリ内のロボットを管理するために役立ちます。

スケジュールで実行するには、アップロード機能を使用して Design Studio から Management Console リポジトリにロボットをアップロードするか、「ロボットをアップロード」ボタンを使用して Management Console から直接アップロードする必要があります。ロボットがアップロードされると、リポジトリにコピーされます。したがって、Design Studio でロボットに変更を加えた場合は、後で再度アップロードする必要があります。ロボットに関連付けられたスケジュールでは、次回の実行時に新しいバージョンのロボットが使用されます。

各ロボットは、プロジェクトに属します。指定のプロジェクト内では、同じ名前を持つ 2 つの異なるロボットを使用することはできません。異なるプロジェクトには、同じ名前のロボットを含めることができます。Design Studio では、ロボットにタグを割り当てることができ、同じタグを使用することで Management Console のリストのフィルタリングに使用できます。

[ロボット] セクションの上部にある [プロジェクト] ドロップダウン リストで、表示するロボットを含むプロジェクトを選択できます。各ロボットに関する情報の表示方法を次のように変更できます。

- [フィルタ] テキスト フィールドにロボット名またはタグ名を入力して、テーブル内のロボットのリストをフィルタリングすることができます。詳細については、フィルタリングを参照してください。
- 右側の メニュー アイコンを使用して、ロボットに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、ロボットごとに次のテーブル列が表示されます。

列	説明
フォルダ	ロボット ファイルの保存に指定されているフォルダの名前。デフォルトでは、ファイルは Root フォルダに保存されます。新しいフォルダを作成してファイルを保存できます。このフォルダ名は一意である必要があります。フォルダ名は、選択したフォルダが Root 以外の場合、列に表示されます。フォルダは、ログ ビューのロボット実行ログ内でロボット名の一部としても表示されます。ロボットを削除する際は、空のフォルダを削除できます。
名前	ロボットの名前。ロボットでリポジトリに存在しないタイプまたはスニペットが使用されている場合、名前は赤でマークされます。
タイプ	ロボットのタイプ: ロボットまたはベーシック エンジン ロボット。
プロジェクト名	ロボットが属するプロジェクトの名前 (すべてのプロジェクトを表示する際に便利)。
タグ	ロボットに割り当てられたタグ。
バージョン	ロボットを編集する際に最後に使用した Kofax RPA バージョン。
サイズ	ロボットのサイズはバイトで表記されます。
スケジュール	ロボットを実行するスケジュールの名前。
入力タイプ	ロボットの入力変数で使用されるタイプ。ロボットを実行するには、これらのタイプに関連した ".type" ファイルが存在する必要があります。
返されるタイプ	ロボットによって返された値のタイプ 。API を介してロボットを実行すると、これらのタイプの値が返される場合があります。ロボットを実行するには、これらのタイプに関連した ".type" ファイルが存在する必要があります。
保存されたタイプ	ロボットによってデータベース データ登録された値のタイプ。ロボットを実行するには、これらのタイプに関連した ".type" ファイルが存在する必要があります。
トリガー	ロボットがマッピングされているトリガー名。
ラベル	ロボットがマッピングされているラベルの名前。
使用されるスニペット	このロボットによって使用されたスニペットの名前。ロボットでスニペット A が使用され、スニペット A でスニペット B が使用されている場合は、スニペット A のみがここに表示されます。
マッピング	ロボットのユーザーとラベルのマッピングを表示します。
最終変更日	ロボットの最近の変更の日付。
インポート元	インポートされたプロジェクトまたは復元されたバックアップの一部であるロボットをインポートしたユーザーのユーザー名。
インポート時間	インポートされたプロジェクトまたは復元されたバックアップの一部であるロボットをインポートした日時。
参照先ロボット	ベーシック エンジン ロボットが呼び出しているロボットの名前。
オプションの列	
ロボット ID	自動的に生成されたロボットの ID。
作成者	ロボットを最初にアップロードしたユーザーのユーザー名。
変更者	ロボットを最後に変更したユーザーのユーザー名。
コミット メッセージ	コミットについて説明した概要。

列	説明
リビジョン番号	ロボットのリビジョンの回数。

ロボットのアップロード

1. Management Console にロボットを追加するには、左上隅の + 記号をクリックします。
   「ロボットのアップロード」 ダイアログ ボックスが表示されます。
   ロボットをアップロードする別の方法は、Design Studio のアップロード機能の 1 つを使うことです。これは、Design Studio は必要なタイプとスニペットをアップロードすること以外は、まったく同じように動作します。Management Console のリポジトリに複数のプロジェクトが含まれる場合は、ロボットをアップロードするプロジェクトを選択するように求められます。
2. ペーパー クリップ記号をクリックして、アップロードするロボット ファイルを選択し、コンピュータ上のファイルを選択してから、[**Open**] をクリックします。
   - 既存のロボットと同じ名前のロボットをアップロードする場合、[存在する場合にオーバーライド] を選択して既存のロボットを置換します。
   - ロボットをアップロードするフォルダを選択します。デフォルトでは、すべてのファイルはルート フォルダに保存されます。
   - [コミット メッセージ] フィールドでコミットの説明を追加できます。
3. [送信] をクリックします。
   ロボットがテーブルに表示されます。
   Management Console にアップロードしたロボットは、「すぐに実行」オプションを使用して直接実行する、スケジュールの一部として実行する、Java/.NET API を介して実行する、RESTサービスとして実行するなど、さまざまな方法で実行できます。下記の「ロボットのアクション」を参照してください。
   Kapplet の一部としてロボットを実行する方法については、Kapplets を参照してください。

ロボットに不足しているスニペットとタイプの追加

ロボットをアップロードすると、[名前]、[必要なタイプ]、[使用されるスニペット] の各列のデータが赤で表示される場合があります。これは、ロボットの実行に必要なスニペットやタイプの一部がアップロードされていないことを意味します。これを行うには、いずれかのテーブル列の赤いテキストにカーソルを合わせて、アップロード メニューを表示します。そのメニューの右側にあるアップロード記号 をクリックすると、タイプまたはスニペットのアップロード ダイアログ ボックスが開きます。

ロボットのアクション

ロボットの ⋮ コンテキスト メニューをクリックすると、次のアクションが一覧表示されます。

- [すぐに実行]: RoboServer でロボットの実行をすぐに開始します。この機能は、入力を取得するロボットには利用できません。
- [フォルダの設定]: 選択したロボットを Management Console のフォルダに追加します。
- スケジュールの作成: スケジュール作成用のウィザードを開きます。この方法で追加したロボットに入力が必要な場合は、後で追加する必要があります。
- **[API]**: RoboServer 上でロボットを実行するためのサンプル Java または C# コードのウィンドウを開きます。
- **[REST]**: ロボットを REST サービスとして呼び出すためのウィンドウを開きます。

Java/.NET API を介してロボットを起動するか、REST サービスとしてロボットを起動するときに、Management Console にロボットを実行するために必要なスロットがない場合、空きスロットがないというメッセージが表示されます。ロボットはキューに格納されません。したがって、API を使用してロボットを開始する場合は、十分なスロットが利用可能なときにロボットが必ず実行されるようにロボットをスケジュールするようにします。たとえば、実行中のロボットの数を常にカウントして、空きスロットが十分にある場合にのみ新しいロボットを起動することができます。また、ロボットの実行を試行するループを作成して、スロット不足によりロボットを実行できない場合は、待機してから再試行するように設定することもできます。

- [**SOAP**]: SOAP でロボットを呼び出すためのウィンドウを開きます。
- [ロボットのパスワード アクセスの追加/編集]: ロボットのアクセス トークンを作成または編集するためのダイアログ ボックスを開きます。このトークンは、ロボットの特定のバージョンに対応しています。ここでは、トークンの説明を編集し、使用可能なパスワード エントリのリストからトークンにパスワード エントリを割り当てることもできます。パスワード ストアの [パスワード エントリ] タブでは、トークンにパスワード エントリを直接割り当てることができます。
- [リソース アクセス トークンを取得]: このロボットのアクセス トークンをコピーするためのダイアログ ボックスを開きます。たとえば、このアクセス トークンを使用して、ロボットがパスワード ストアまたはロボット ファイル システムにアクセスできるようにすることができます。
- [ユーザーにマッピング]: ロボット内のトリガーを持つロボットに対して使用できます。ロボットをマッピングするユーザーを選択できます。
- [ラベルにマッピング]: ロボットをマッピングするラベルを選択できます。ここに新しいラベルを入力するか、既存のラベルを選択できます。
- [トリガーをサスペンド]: ロボット内のトリガーを持つロボットに対して使用できます。クリックすると、ロボットのトリガーが無効になります。
- [トリガーのアクティブ化]: ロボット内のトリガーを持つロボットに対して使用できます。クリックすると、サスペンドされたロボットでトリガーが有効になります。
- [ダウンロード]: ロボットのコピーをリポジトリからダウンロードし、ファイル システムに保存します。
- [ドキュメントを生成]: ロボットが実行するアクションなど、ロボットの概要を生成します。
- [**Design Studio** で開く]: Design Studio でロボットを開きます。
- [ロボットの実行を表示]: ログ ビューの表示内容と同一の、ロボットの実行に関するログ情報を表示します。
- [ロボットのエラーを表示]: ログ ビューの表示内容と同一の、ロボット エラーに関するログ情報を表示します。

リポジトリからロボットを削除するには、テーブルでロボットを選択し、左上隅の ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。ロボットは、実行に使用されるスケジュールから自動的に除去されます。ファイル システムにロボットのコピーがない場合は、完全に失われます。

API:

[リポジトリ] > **[ロボット]** セクションで、ロボットの ⋮ コンテキスト メニューから **[API]** をクリックして、Java または .NET のコードを生成する [コード生成] ウィンドウにアクセスします。

API を使用してロボットの実行を開始する前に、『Kofax RPA 開発者ガイド』の関連する章を読み、API の仕組みを理解しておくことをお勧めします。

REST

ロボットを REST サービスとして実行できます。これにより、ロボットを任意のプログラミング言語から呼び出すことも、JavaScript を使用してブラウザから直接呼び出すこともできます。

[リポジトリ] > **[ロボット]** セクションにある、ロボットの ⋮ コンテキスト メニューで、[**REST**] をクリックします。

- 使用する方法 (GET または POST) を選択し、リクエストとレスポンスのフォーマットを設定します。入力を必要とするロボットは、POST を使用して呼び出される必要があります。入力なしのロボットは、GET または POST のいずれかを使用して呼び出される必要があります。
- [フォーマット] ボタンを使用すると、テストを行う際のリクエストとレスポンスのフォーマットを設定できます。ただし、サービスをコードから呼び出す場合、フォーマットは `Accept` および `Content-Type` HTTP ヘッダーによって制御されます。`Accept` ヘッダーは、必要なレスポンス フォーマットを指定します。また、`Content-Type` ヘッダーはリクエスト フォーマットを指定します。
- サービス ウィンドウの [リクエスト] ペインを使用して、リクエストを構築します。[テスト サービス] をクリックしてロボットを実行します。すると、ウィンドウの [レスポンス] ペインに結果が表示されます。

REST サービスは、REST Web サービス呼出アクションを使用してロボットから簡単に呼び出すことができます。

ロボットの REST Web サービス呼出ステップで OAuth を使用する必要がある場合は、Management Console の OAuth ページで OAuth アプリケーションとユーザーを登録し、Management Console のスケジュールまたは REST リクエストを使用してロボットを起動する際にその OAuth アプリケーションとユーザー名を使用してください。必要な情報はすべて、登録済みの OAuth アプリケーションから取得されます。

プロジェクトまたはロボット名に非 ASCII 文字が含まれる場合は、URL が適切にエンコーディングされることを確認してください (UTF-8 URL エンコーディング)。これはロボットで自動的に行われますが、サービスをコードから呼び出す場合は、開発者が URL をエンコーディングする必要があります。

サービスとして実行されるロボットは、ロボットで初めて API 例外が生成されたときに停止します。これは、ロボットによって生成された API 例外に関係なく実行を継続する、スケジュールされたロボットとは異なります。

サービスとして実行される各ロボットは、リクエスト スレッドを使用します。Management Console が RoboServer に埋め込まれて実行されている場合は、最大 100 件のリクエスト スレッドを使用できます。これらの 100 件のスレッドは、Management Console にアクセスするユーザー、Design Studio からのアップロード、リポジトリ API など、すべてのタイプの HTTP 要求に使用されます。より多くの同時 REST サービスを実行する必要がある場合は、スタンドアロン バージョンの Management Console を Tomcat にインストールして、リクエスト スレッドの数を制御できるようにします。

SOAP

ロボットは、SOAP リクエストを開始することで、その他のコンピューターにインストールされているプログラムと通信し、必要な情報を送信して、レスポンスを返すことができます。

[リポジトリ] > [ロボット] セクションで、ロボットの ⋮ コンテキスト メニューから [**SOAP**] をクリックして、SOAP 要求を編集およびテストするためのウィンドウにアクセスします。

入力フォーマット

「正常」または「フラット」は、SOAP リクエスト メッセージの構成を指しています。たとえば、ロボット myRobot の入力変数として var1 と var2 が予想され、どちらのタイプにも属性 attr1 および attr2 が含まれている場合、「正常」の SOAP メッセージは次のようになります。

```
<myRobot>
   <var1>
     <attr1>Some value</attr1>
     <attr2>Another value</attr2>
   </var1>
   <var2>
     <attr1>More input</attr1>
     <attr2>and some more</attr2>
   </var2>
 </myRobot>
```

「フラット」構造では、以下のような SOAP メッセージを必要とします。

```
<myRobot>
    <var1__attr1>Some value</var1__attr1>
    <var1__attr2>Another value</var1__attr2>
    <var2__attr1>More input</var2__attr1>
    <var2__attr2>and some more</var2__attr2>
 </myRobot>
```

フラット構造は、互換性の理由から導入されました。

WSDL URL

このロボットが属するプロジェクトの WSDL の URL。注意：この URL は、同じプロジェクトのロボットすべてと同一です。

リクエスト **URL**

ロボットを実行する際、HTTP POST リクエストは、この URL に送信される必要があります。

SOAP アクション

ロボットを実行する際、SOAPAction という HTTP ヘッダーが、表示されている値とともに示されます。

リクエスト

このフィールドは、例の SOAP メッセージで事前に入力されます。すべての入力属性には、デフォルト/テスト値があります。この値は、[テスト サービス] をクリックする前に編集できます。

レスポンス

ロボット実行からの出力を含む編集できないフィールド。

入力パラメータにエラーがある場合や、ロボット実行の際にエラーが発生した場合、"SOAP Fault" メッセージが表示されます (エラーの理由と詳細の一部が含まれます)。

重要なメモ

- プロジェクト名には、WSDL で許可されていない文字を含めることができます。そのため、WSDL/SOAP メッセージではプロジェクト名が異なることがあります。具体的には、英数字 (a ~ z、A ~ Z、0 ~ 9) 以外のすべての文字が _ に置き換えられます。
- 同じように、ロボット名は違う形で表示されることがあります。これらは、プロジェクト名と同じように変換されます。ただし、ロボット名が変更されると、特殊な接尾語 (例：_1234) も追加されます。

- 現在、SOAP 1.1 がサポートされています。

ロボットのドキュメント

ロボットのドキュメント機能を使用すると、ロボットの構造やロボットが実行するアクションなど、ロボットの概要を生成できます。これは、ロボットが実行するアクションのシーケンスを調べたり、ロボットに関する技術的な詳細 (他のロボット、スニペット、リモート デバイスなどへの依存関係など) を共有したりする必要がある場合に役立ちます。生成されたドキュメントは、保存したり印刷したりすることができます。

ロボットの概要を生成するには、ロボットの ⋮ コンテキスト メニューで [ドキュメントを生成] をクリックします。

デフォルトでは、ロボットのドキュメント機能が有効になっています。無効にするには、**RoboServer** 設定アプリケーションで [ドキュメント リクエストの禁止] を選択する必要があります。詳細については、「RoboServer の開始」を参照してください。

ロボットのドキュメントを生成するには、Management Console のバージョンと、この機能が有効になっている、対応する RoboServer が一致する必要があります。

このトピックは、次のサブトピックに分かれています。

- ヘッダー
- ロボットの詳細と依存関係
- ロボット構造
- ロボットのステップの説明
- ロボットのドキュメントの保存と印刷

ヘッダー

概要のヘッダーには、ロボット タイプに応じて、次の情報を含めることができます。

- [ロボット名]: ロボットの名前と Design Studio でロボットを開くためのボタン。
- [プロジェクト名]: ロボットが属しているプロジェクトの名前。
- [ブラウザ エンジン]: ベーシック エンジン ロボットのみ。ロボットの構築に使用されるブラウザ エンジン。
- [ロボット コメント]: Design Studio のロボットの設定ダイアログで指定されている場合は、ロボットに関するコメント。

ロボットの詳細と依存関係

概要のサブヘッダーには、ロボット タイプに応じて、次の情報を含めることができます。

- [タイプ]: ロボットのタイプ: ロボット またはベーシック エンジン ロボット 。
- [バージョン]: ロボットが最後に変更および保存された製品バージョン。
- [バージョン別に保存]: ロボットがこれまでに変更および保存された各製品バージョン。
- [データベースに保存されているタイプ]: データベースに保存される変数として使用されるタイプの名前。これは、データベース データ登録ステップで実行されます。
- [タグ]: Design Studio のロボットの設定ダイアログで指定されている場合は、ロボットに割り当てられているタグ。

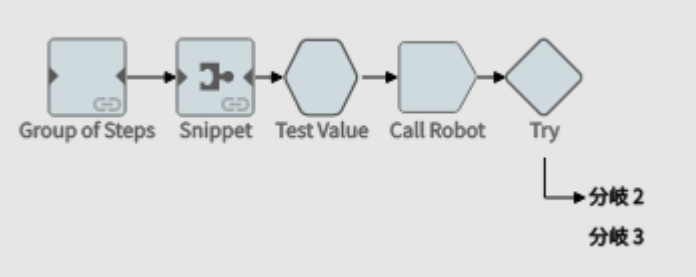
- [マップ済みのデバイス]: ロボットがマップされているリモート デバイス。
- [使用済みのロボット]: ベーシック エンジン ロボットが呼び出しているロボット 。
- [使用済みのタイプ]: ロボットによって使用されたすべてのタイプの名前。
- [返されるタイプ]: 抽出されたデータを含むタイプの名前。
- [入力タイプ]: 入力変数として使用されるタイプの名前。
- [出力タイプ]: 出力変数として使用されるタイプの名前。
- [トリガー]: ロボットがマップされているトリガーの名前。
- [使用済みのスニペット]: ロボットが使用しているスニペットの名前。
- [作成者]: ロボットを最初にアップロードしたユーザーのユーザー名。
- [最終変更日]: ロボットの最新の変更日。
- [変更者]: ロボットを最後に変更したユーザーのユーザー名。

ロボットによって使用される Connector に関する情報は、ロボットのドキュメントには含まれません。

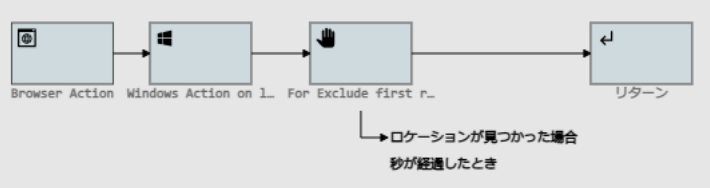
ロボット構造

概要では、ロボットのワークフローが複数のセクションに分割される場合があります。各セクションは、ロボット内の各セグメントに対応しています。ロボット構造では、Design Studio でロボットを開いたときの表示と同じように、ロボットのワークフローを示します。構造内のすべての要素は名前で識別され、Design Studio の場合と同じ図形またはアイコンが使用されています。

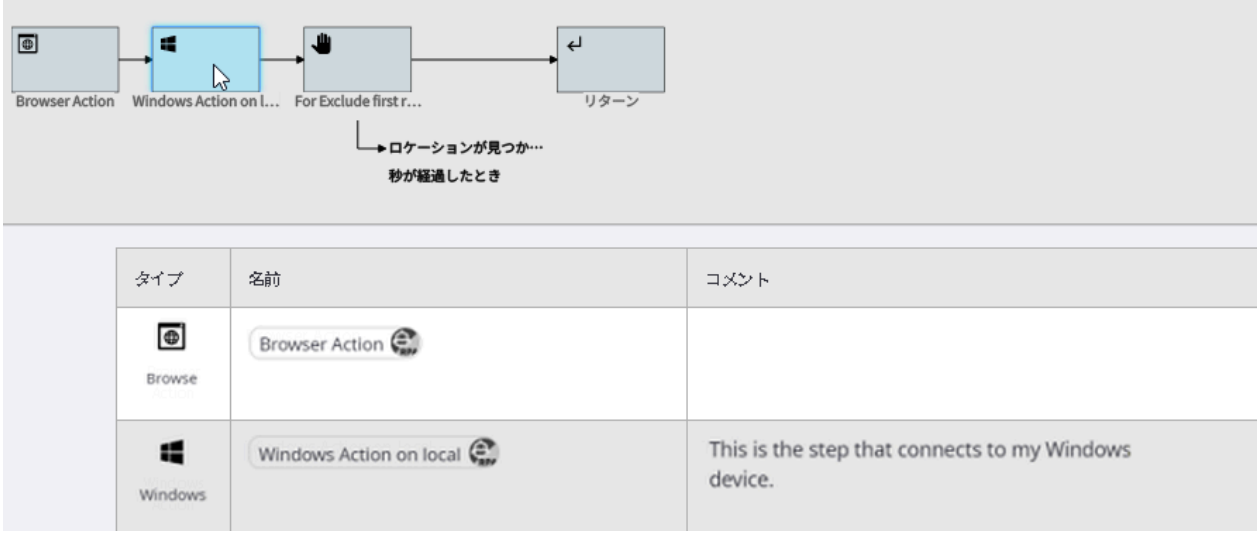
たとえば、ベーシック エンジン ロボットのほとんどのアクション ステップは四角形で表示され、トライ ステップは菱形で表示されます。



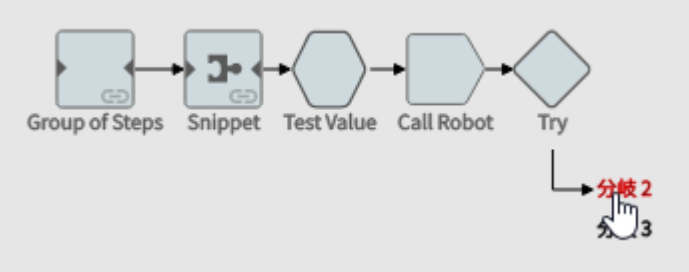
ロボットのアクションは、さまざまなアイコンで識別されます。



ロボット構造内の任意のステップにポインターを移動すると、テーブル内で強調表示され、ステップの説明が表示されます。ステップ名の横にあるをクリックすると、Design Studio でロボットが開き、このステップに直接移動できます。



ロボット構造の一部の要素は、クリック可能です。つまり、要素をクリックすると、概要内でその要素を説明している該当セクションに移動できます。たとえば、グループ ステップ、スニペット、および分岐は、ロボットの概要に独自のセクションが用意されており、ロボット構造でグループ ステップ要素、スニペット、または分岐の名前をクリックするか、概要を下にスクロールすると、そのセクションに移動できます。クリック可能な要素にポインターを移動すると、カーソルが指のポインターに変わります。



分岐が空の場合、その分岐はクリックできず、概要に個別のセクションがないことに注意してください。

次の表では、ロボットのドキュメントで使用されている要素とグラフィックについて説明します。

——→	ワークフローの方向。
···→	ワークフローの継続。
@分岐 <番号> (ベーシック エンジン ロボットの場合) <分岐名> (ロボットの場合)	分岐。この要素をクリックすると、概要で分岐について説明しているセクションにリダイレクトされます。
🔗	グループ リンク。グループ ステップに表示されます。グループをクリックすると、概要でそのグループについて説明しているセクションにリダイレクトされます。

	Design Studio を開くためのボタン。クリックすると、Design Studio でロボットが開き、このステップに直接移動します。詳細については、「URL でファイルを開く」を参照してください。

ロボットのステップの説明

ロボットのステップを説明するテーブルには、次の情報が含まれています。

タイプ	ステップの基盤となるアクション。
名前	ステップまたはステップのグループの名前。 ロボットを呼び出すステップの場合は、呼び出されるロボットの名前も含まれます。
コメント	Design Studio で指定されている場合は、ステップに関するコメント。

ロボットのドキュメントの保存と印刷

右下隅の [印刷] をクリックすると、ロボットのドキュメントを印刷したり、PDF 形式で保存したりできます。このボタンをクリックすると、ブラウザ ウィンドウが開き、出力形式 (A4 推奨) やレイアウトなどの設定を行うことができます。PDF として保存すると、概要セクション間のナビゲーションが PDF ドキュメントに保存されます。そのため、Management Console と同様に、ロボット構造内のさまざまな要素をクリックして、概要の各セクションにジャンプすることができます。

Firefox では、ロボットのドキュメントの印刷および PDF 形式での保存はサポートされていません。

タイプ

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、開発者、API、VCS サービス ユーザー、および Kapplet サービス ユーザー。

[タイプ] セクションには、Management Console のリポジトリにアップロードしたタイプが一覧表示されます。スケジュールが実行されると、そのスケジュールにリンクされたロボットが RoboServer 上で実行されます。多くのロボットは、入力値または出力値のいずれかの定義として、あるいは両方の定義としてタイプが必要になります。これらのタイプは、ロボットと同じプロジェクトのリポジトリに追加する必要があります。

タイプをリポジトリにアップロードすると、リポジトリにコピーされます。したがって、後で Design Studio のタイプに変更を加えた場合は、再度アップロードする必要があります。それぞれのタイプ名は各プロジェクト内で一意である必要があるため、同じ名前の複数のタイプについては、別々のプロジェクト内にある場合のみアップロードできます。

[タイプ] セクションの上部にある [プロジェクト] ドロップダウン リストで、表示するタイプを含むプロジェクトを選択できます。次のように、各タイプに関する情報の表示方法を変更できます。

- [フィルタ] テキスト フィールドにフィルタを適用して、テーブル内のタイプのリストをフィルタリングします。詳細については、フィルタリングを参照してください。
- 右側の メニュー アイコンを使用して、タイプに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。

- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、タイプごとに次のテーブル列が表示されます。

列	説明
フォルダ	タイプの保存に指定されているフォルダの名前。デフォルトでは、ファイルは Root フォルダに保存されます。新しいフォルダを作成してファイルを保存できます。このフォルダ名は一意である必要があります。フォルダ名は、選択したフォルダが Root 以外の場合、列に表示されます。タイプを削除する際は、空のフォルダを削除できます。
名前	タイプの名前。
サイズ	タイプのサイズはバイトで表記されます。
プロジェクト名	タイプが属するプロジェクトを表示します (すべてのプロジェクトを表示する際に便利です)。
最終変更日	このタイプの最終変更のタイム スタンプ。
オプションの列	
作成者	タイプを最初にアップロードしたユーザーのユーザー名。
変更者	タイプを最後に変更したユーザーのユーザー名。
コミット メッセージ	コミットについて説明した概要。
リビジョン番号	タイプのリビジョンの回数。
インポート元	インポートされたプロジェクトまたは復元されたバックアップの一部であるタイプをインポートしたユーザーのユーザー名。
インポート時間	インポートされたプロジェクトまたは復元されたバックアップの一部であるタイプをインポートした日時。

タイプのアップロード

1. Management Console にタイプを追加するには、左上隅の + 記号をクリックします。
   「タイプのアップロード」 ダイアログ ボックスが表示されます。
   タイプは Design Studio から暗黙的にアップロードすることもできます。これは、Design Studio を使用して、タイプを使用するロボットをアップロードする際に起こります。Design Studio はロボットおよびタイプ間の依存関係について認識しているため、常にロボットとともに必要なタイプをアップロードします。
2. ペーパー クリップ記号をクリックして、アップロードするタイプ ファイルを選択し、コンピュータ上のファイルを選択してから、[**Open**] をクリックします。
   - 既存のタイプと同じ名前のタイプをアップロードする場合は、[存在する場合にオーバーライド] を選択して既存のタイプを置換します。
   - タイプをアップロードするフォルダを選択します。デフォルトでは、すべてのファイルはルート フォルダに保存されます。
   - [コミット メッセージ] フィールドでコミットの説明を追加できます。
3. [送信] をクリックします。
   タイプがテーブルに表示されます。

さらに、⋮ コンテキスト メニューからタイプに対して次のアクションを実行できます。

- [フォルダの設定]: タイプが配置されているフォルダを変更します。
- [データベース テーブルの生成]: 抽出した値をデータベースに格納するためのデータベース テーブルを作成します。
- [ダウンロード]: タイプのコピーをコンピュータにダウンロードします。
- [**Design Studio** で開く]: Design Studio でタイプを開きます。

タイプを削除するには、テーブルでタイプを選択し、左上隅の 🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。タイプがロボットまたはスニペットによって使用されている場合、確認メッセージには、使用回数が含まれます。ロボットまたはスニペットによって使用されているタイプを削除する場合、そのロボット (または、そのスニペットを使用しているロボット) は実行できなくなります。コンピュータにタイプのコピーがない場合、タイプは完全に失われます。

タイプからのデータベース テーブルの生成

抽出された変数値をデータベースに格納するには、一致するデータベース テーブルを作成します。Management Console にアップロードされた 1 つ以上のタイプからデータベース テーブルを作成するには、以下の手順を実行します。

1. **[**リポジトリ**]** > **[**タイプ**]** でタイプを選択します。
2. このタイプの ⋮ コンテキスト メニューをクリックし、[データベース テーブルの生成] をクリックします。
   [テーブル生成] ダイアログ ボックスが表示されます。
3. [テーブル生成] ダイアログ ボックスで、プロジェクトに定義された設定済みデータベース マッピングを 1 つ選択し、テーブルをドロップする SQL コードを生成するかどうかを選択します。[**SQL** を生成] をクリックします。
   選択したデータベース上で、選択したタイプからテーブルを生成する ([テーブルを削除する **SQL** を生成] を選択した場合はテーブルをドロップする) SQL コードを含む、SQL エディターが開きます。表示された SQL は推奨 SQL です。必要に応じてステートメントを変更できます。準備を整えてから、[**SQL** 実行] をクリックします。
   SQL コードを実行すると、実行状態 (成功または失敗とその説明) とともにメッセージが表示されます。[**OK**] をクリックしてメッセージを閉じ、ウィンドウを閉じます。

スニペット

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、開発者、API、VCS サービス ユーザー、および Kapplet サービス ユーザー。

スニペット セクションには、Management Console のリポジトリにアップロードしたスニペットが一覧表示されます。スケジュールが実行されると、そのスケジュールにリンクされたロボットが RoboServer 上で実行されます。一部のロボットは、そのロボットと同じプロジェクトのリポジトリで利用可能なスニペットを使用します。

スニペットをリポジトリにすると、リポジトリにコピーされます。したがって、後で Design Studio のスニペットに変更を加えた場合は、再度アップロードする必要があります。それぞれのスニペット名は各プロジェクト内で一意である必要があるため、同じ名前の複数のスニペットについては、別々のプロジェクト内にある場合のみアップロードできます。

[スニペット] セクションの上部にある [プロジェクト] ドロップダウン リストで、表示するスニペットを含むプロジェクトを選択できます。各スニペットに関する情報の表示方法を次のように変更できます。

- [フィルタ] テキスト フィールドにフィルタを適用して、テーブル内のスニペットのリストをフィルタリングします。詳細については、フィルタリングを参照してください。
- 右側の メニュー アイコンを使用して、スニペットに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、スニペットごとに次のテーブル列が表示されます。

名前	説明
フォルダ	スニペットの保存に指定されているフォルダの名前。デフォルトでは、ファイルは Root フォルダに保存されます。新しいフォルダを作成してファイルを保存できます。このフォルダ名は一意である必要があります。フォルダ名は、選択したフォルダが Root 以外の場合、列に表示されます。スニペットを削除する際は、空のフォルダを削除できます。
名前	スニペット名。
プロジェクト名	スニペットが属するプロジェクト名 (すべてのプロジェクトを表示する際に便利)。
入力タイプ	スニペットの入力変数で使用されるタイプ。このスニペットをロボットで使用するには、これらのタイプに関連した ".type" ファイルが存在する必要があります。
返されるタイプ	スニペットによって返された値のタイプ。このスニペットをロボットで使用するには、これらのタイプに関連した ".type" ファイルが存在する必要があります。
保存されたタイプ	スニペットによってデータベース データ登録された値のタイプ。このスニペットをロボットで使用するには、これらのタイプに関連した ".type" ファイルが存在する必要があります。
使用されるスニペット	このスニペットによって使用されたスニペットの名前。スニペットでスニペット A が使用され、スニペット A でスニペット B が使用されている場合は、スニペット A のみがここに表示されます。
サイズ	スニペットのサイズはバイトで表記されます。
作成者	スニペットを最初にアップロードしたユーザーのユーザー名。
コミット メッセージ	コミットについて説明した概要。
リビジョン番号	スニペットのリビジョンの回数。
最終変更日	このスニペットの最終変更のタイム スタンプ。
オプションの列	
変更者	スニペットを最後に変更したユーザーのユーザー名。
インポート元	インポートされたプロジェクトまたは復元されたバックアップの一部であるスニペットをインポートしたユーザーのユーザー名。
インポート時間	インポートされたプロジェクトまたは復元されたバックアップの一部であるスニペットをインポートした日時。

スニペットのアップロード

1. スニペットを追加するには、左上隅の + 記号をクリックします。
   「スニペットのアップロード」 ダイアログ ボックスが表示されます。
   また、スニペットは Design Studio から暗黙的にアップロードできます。これは、Design Studio を使用して、スニペットを使用するロボットをアップロードする際に発生します。Design Studio はロボットおよびスニペット間の依存関係について認識しているため、常にロボットとともに必要なスニペットをアップロードします。
2. ペーパー クリップ記号をクリックして、アップロードするスニペット ファイルを選択し、コンピュータ上のファイルを選択してから、[**Open**] をクリックします。
   - 既存のスニペットと同じ名前のスニペットをアップロードする場合は、[存在する場合にオーバーライド] を選択して既存のスニペットを置換します。
   - スニペットをアップロードするフォルダを選択します。デフォルトでは、すべてのファイルはルート フォルダに保存されます。
   - [コミット メッセージ] フィールドでコミットの説明を追加できます。
3. [送信] をクリックします。
   スニペットがテーブルに表示されます。

さらに、⋮ コンテキスト メニューからスニペットに対して次のアクションを実行できます。

- [フォルダの設定]: スニペットが配置されているフォルダを変更します。
- [ダウンロード]: スニペットのコピーをコンピュータにダウンロードします。
- [**Design Studio** で開く]: Design Studio でスニペットを開きます。

スニペットを削除するには、テーブルでスニペットを選択し、左上隅の ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。スニペットがロボットまたはスニペットで使用されている場合、確認メッセージには使用回数が含まれます。ロボットで使用されているスニペットを削除すると、そのロボットは実行できなくなります。スニペットを削除する際は、空のフォルダを削除できます。

リソース

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、開発者、API、VCS サービス ユーザー、および Kapplet サービス ユーザー。

[リソース] セクションには、Management Console リポジトリにアップロードされたリソースが表示されます。これらのリソースは、バイナリ属性とともに入力変数を持つスケジュール設定されたロボットの入力として使用できます。このような変数をロボットに追加およびロードする方法についての情報は、「ロボットでのローカル ファイルの使用」を参照してください。

[リソース] セクションの上部にある [プロジェクト] ドロップダウン リストで、表示するリソースを含むプロジェクトを選択できます。各リソースに関する情報の表示方法を次のように変更できます。

- [フィルタ] テキスト フィールドにフィルタを適用して、テーブル内のリソースのリストをフィルタリングすることができます。詳細については、フィルタリングを参照してください。
- 右側の メニュー アイコンを使用して、リソースに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。

- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、リソースごとに次の情報が表示されます。

列	説明
名前	リソースの名前。
プロジェクト名	リソースが属するプロジェクト名 (すべてのプロジェクトを表示する際に便利)。
フォルダ	リソースの保存に指定されているフォルダの名前。デフォルトでは、ファイルは Root フォルダに保存されます。新しいフォルダを作成してファイルを保存できます。このフォルダ名は一意である必要があります。フォルダ名は、選択したフォルダが Root 以外の場合、列に表示されます。リソースを削除する際は、空のフォルダを削除できます。
サイズ	リソースのサイズはバイトで表記されます。
最終変更日	リソースの最終変更のタイム スタンプ。
変更者	リソースを最後に変更したユーザーのユーザー名。
リビジョン番号	リソースのリビジョンの回数。
コミット メッセージ	コミットについて説明した概要。
作成者	リソースを最初にアップロードしたユーザーのユーザー名。
インポート元	インポートされたプロジェクトまたは復元されたバックアップの一部であるリソースをインポートしたユーザーのユーザー名。
インポート時間	インポートされたプロジェクトまたは復元されたバックアップの一部であるリソースをインポートした日時。

リソースのアップロード

1. リソースを追加するには、左上隅の + 記号をクリックします。
   「リソースのアップロード」 ダイアログ ボックスが表示されます。
2. ペーパー クリップ記号をクリックして、アップロードするリソース ファイルを選択し、コンピュータ上のファイルを選択してから、[**Open**] をクリックします。
   - 既存のリソースと同じ名前のリソースをアップロードする場合、[存在する場合にオーバーライド] を選択して既存のリソースを置換します。
   - リソースをアップロードするフォルダを選択します。デフォルトでは、すべてのファイルはルート フォルダに保存されます。
   - [コミット メッセージ] フィールドでコミットの説明を追加できます。
3. [送信] をクリックします。
   リソースがテーブルに表示されます。

さらに、⋮ コンテキスト メニューからリソースに対して次のアクションを実行できます。

- [ダウンロード]: リソースのコピーをコンピュータにダウンロードします。

リソースを削除するには、テーブルでリソースを選択し、左上隅の ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

デバイス マッピング

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、および開発者。

「デバイス マッピング」セクションには、ロボットで使用可能なマッピング済みのオートメーション デバイスが表示されます。Design Studio でこれらのデバイスのマッピングを作成することができます。

仕組み

ロボットの起動後に、[Management Console] > [管理] > [デバイス] にリストされたデバイスの IP アドレスまたはホスト名にロボットが直接アクセスすることはできません。

ロボットは、デバイス マッピング名とラベルを使用します。同じラベルのデバイスが複数ある場合、ロボットは 「デバイス マッピング」 セクションにリストされているデバイスのリストを上から順に確認して、いずれかのデバイスを選択します。

利用可能なデバイスを要求するコールは Hazelcast を介して実行され、利用可能な場合は毎回同じデバイスを返す可能性があります。Management Console または ロボットが命令を強制したり制御したりすることはありません。複数のラベルを使用すると、ロボットが単一のデバイスに結び付けられることはなくなるため、デバイスが停止している場合や使用中の場合にエラーが発生しなくなり、冗長性が実現されます。しかし、ロボットで指定されたデバイス マッピングのラベルに関連付けられた任意のデバイスは使用することができます。

次のように、各デバイス マッピングの情報の表示方法を変更できます。

- [フィルタ] テキスト フィールドにフィルタを適用して、テーブル内のデバイス マッピングのリストをフィルタすることができます。詳細については、フィルタリングを参照してください。
- 右側の メニュー アイコンを使用して、デバイス マッピングに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、デバイス マッピングごとに次の情報が表示されます。

列	説明
名前	デバイス マッピングの名前。
ラベル	デバイス マッピングのラベル。ラベルを使用して、ロボットで自動化するデバイスをフィルタリングすることができます。

新しいオートメーション デバイス マッピングの作成

1. 新しいデバイス マッピングを作成するには、左上隅の + 記号をクリックします。
   「デバイス マッピングの追加」ダイアログ ボックスが表示されます。
2. デバイス マッピングの名前とラベル (または複数のラベル) を指定します。
   ラベルはコンマで区切る必要があります。
3. [保存] をクリックします。

さらに、⋮ コンテキスト メニューからデバイス マッピングに対して次のアクションを実行できます。

- [編集]:「デバイス マッピングの追加」 ダイアログ ボックスと同じフィールドが含まれます。

Management Console からデバイス マッピングを削除するには、テーブルでデバイス マッピングを選択し、左上隅の 🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

データベース マッピング

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、および開発者。

「データベース マッピング」セクションには、選択したプロジェクトのデータベース マッピングが一覧表示されます。マッピングを使用してロボットをクラスタ内の別のデータベースにリンクしたり、このセクションで新しいマッピングを作成したりすることができます。Kofax RPA をインストールすると、デフォルトのデータベース マッピング「`objectdb`」がデフォルトのプロジェクトに追加されます。これは、[本番] クラスタのデフォルトの開発用データベースに指定されています。

データベース マッピングの詳細については、Design Studio の章のデータベースのマッピングを参照してください。

データベース マッピングを作成する場合は、次のことに注意してください。

- 1 つのプロジェクトに同じ名前の複数のデータベース マッピングを含めることはできません。
- データベース マッピングの名前には、空白、括弧、ハイフンを使用できます。たとえば、`"Development Database (MySQL)"` は有効なデータベース マッピング名です。

「データベース マッピング」セクションの上部にある [プロジェクト] ドロップダウン リストで、表示するデータベースを含むプロジェクトを選択できます。各データベース マッピングに関する情報の表示方法を次のように変更できます。

- [フィルタ] テキスト フィールドにフィルタを適用して、テーブル内のデータベース マッピングのリストをフィルタリングします。詳細については、フィルタリングを参照してください。
- 右側の ▮▮▮ メニュー アイコンを使用して、データベース マッピングに表示するテーブル列を選択します。
- 右側の ↻ 更新アイコンをクリックして、表示された情報を更新します。
- 右側の ↺ リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、データベース マッピングごとに次の情報が表示されます。

列	説明
名前	データベース マッピング名
プロジェクト名	マッピングが割り当てられているプロジェクトの名前 (すべてのプロジェクトを表示する際に便利です)。
データベース	マッピングが使用されるデータベースの名前。
クラスタ	データベース マッピングが関連付けられているクラスタの名前。

新しいデータベース マッピングの作成

1. 新しいデータベース マッピングを作成するには、左上隅の + 記号をクリックします。
   「データベース マッピングを追加」ダイアログ ボックスが表示されます。
2. マッピングを割り当てるプロジェクトを指定します。
3. データベース マッピングをマッピングするクラスタとデータベースを選択します。
4. [保存] をクリックします。

さらに、⋮ コンテキスト メニューからデータベース マッピングに対して次のアクションを実行できます。

- [編集]:「データベース マッピングを追加」ダイアログ ボックスと同じフィールドが含まれます。

Management Console からデータベース マッピングを削除するには、テーブルでデータベース マッピングを選択し、左上隅の 🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

ℹ データベースの最終マッピングを削除すると、クラスタがシェア データベースを Design Studio に提供するように選択されている場合でも、このデータベースは Design Studio で利用できなくなります。

電子メール トリガー

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、および開発者。

この 「電子メール トリガー」 ウィンドウには、電子メール トリガー機能用に設定された電子メール トリガーが一覧表示されます。このウィンドウを使用して、新しいトリガーの追加や不要になったトリガーの削除を行います。

電子メール トリガーに指定したロボットには、ロング テキスト属性を持つコンプレックス タイプの変数が 1 つ含まれている必要があります。ロボットは、電子メールのすべてのヘッダー、本文、および 1 つ以上の添付ファイルを読み取ることができます。すべての添付ファイルは Base64 形式にエンコードされます。ロボットに変数が含まれていない場合、電子メールは処理されず、ロボットを起動するトリガーとしてのみ使用されます。

「電子メール トリガー」 セクションの上部にある [プロジェクト] ドロップダウン リストで、表示する電子メール トリガーを持つプロジェクトを選択します。それぞれの電子メール トリガーに関する情報の表示方法を次のように変更できます。

- 右側の ||| メニュー アイコンを使用して、電子メール トリガーに表示するテーブル列を選択します。
- 右側の ↻ 更新アイコンをクリックして、表示された情報を更新します。
- 右側の ↺ リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、電子メール トリガーごとに次の情報が表示されます。

列	説明
電子メール トリガー	電子メール トリガーの名前。
割り当てられた電子メール フォルダ	電子メール サーバー上のフォルダ。このフォルダに表示される電子メール メッセージにより、トリガーが開始されます。
ロボット	電子メール トリガーを使用するロボット。
プロジェクト	電子メール トリガーを含むプロジェクト。

新しい電子メール トリガーを作成

1. 新しい電子メール トリガーを追加するには、左上隅の + 記号をクリックします。
「電子メール トリガー マッピングを作成」 ダイアログ ボックスが表示されます。
2. 次のパラメータを指定します。
   a. トリガー名: 電子メール トリガーの名前を入力します。
   b. プロジェクトを選択: 電子メール トリガーを含めるプロジェクトを選択します。
   c. 電子メール アカウントを選択: トリガー用の電子メール アカウントを選択します。詳細については、電子メール アカウント を参照してください。
   d. ロボットの選択: トリガーを使用するロボットを選択します。
   e. 優先度: 処理の優先度 (最小、低、中、高、最大) を選択します。この優先度は、ロボットがキューに入れられた場合に、電子メール トリガーのあるロボット間で設定されます。
   f. タイムアウト: 電子メール処理のタイムアウトを選択します。指定したタイムアウト中に処理されなかった電子メールは、[受信トレイ] の [タイムアウト] フォルダに移動されます。

      RPA によって受信トレイ内に以下のフォルダが作成されます。
      - エラー: 処理中にエラーが発生した電子メールが含まれます。
      - 終了: 正常に処理された電子メールが含まれます。
      - 処理中: 処理中の電子メールが含まれます。
      - タイムアウト: 指定したタイムアウト中に処理されなかった電子メールが含まれます。

   g. 電子メール フォルダを選択: 電子メール サーバー上のフォルダを選択します。このフォルダに表示される電子メール メッセージにより、トリガーが開始されます。電子メール サーバーの [受信トレイ] の下にサブフォルダのリストが表示されている場合に、サブフォルダの 1 つを選択すると、選択したサブフォルダに表示されるメッセージのみが口ボットの実行をトリガーします。この場合、[受信トレイ] に配置されたメッセージは口ボットの実行をトリガーしません。
      電子メール サーバーのサブフォルダのリストが表示されていない場合に、[受信トレイ] を選択すると、[受信トレイ] とそのサブフォルダに表示されるすべてのメッセージが口ボットをトリガーします。
3. [**OK**] をクリックします。

さらに、⋮ コンテキスト メニューから電子メール トリガーに対して次のアクションを実行できます。

- 編集:「電子メール トリガー マッピングを作成」ダイアログ ボックスと同じフィールドが含まれます。

Management Console から電子メール トリガーを削除するには、テーブルで電子メール トリガーを選択し、左上隅の 🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

トリガー マッピング

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、開発者、DAS クライアント、および VCS サービス ユーザー。

「トリガー マッピング」セクションには、使用可能なトリガー マッピングが表示されます。トリガーは、ロボットのアテンデッド オートメーション機能の一部です。トリガー ロボットは、リモート デバイス上のイベントに反応するロボットです。トリガーを持つロボットを Management Console にアップロードした後に、ロボットをユーザーやラベルにマッピングできます。その後、Management Console は、作成したマッピングに基づいて、トリガーのリストを Desktop Automation サービスに提供します。リモート デバイスにトリガー イベントが検出されると、Desktop Automation サービスは Management Console に通知を送信し、ロボットはプログラムされたいくつかのステップを実行します。たとえば、特定のアプリケーションを開いた場合にデータを挿入または抽出するようにロボットをプログラムすることができます。

Kofax RPA は、ロボットのユーザーとラベルのマッピングをサポートしています。ユーザー マッピングを使用できるのは、トリガーを持つロボットのみです。[ロボット] セクションで、ロボットのトリガーを有効または無効にできます。

「トリガー マッピング」セクションには、次の 2 つのタブがあります。ユーザーとラベル。[ユーザー] タブには、ユーザー マッピングが割り当てられたロボットが表示されます。[ラベル] タブには、ラベルが割り当てられたロボットが表示されます。トリガーを持つロボットにのみ、ユーザー マッピングを割り当てることができます。

各マッピングに関する情報の表示方法を次のように変更できます。

- フィルタを適用して、テーブル内のトリガー マッピングのリストをフィルタリングします。選択したタブに応じて、ロボット名とユーザー/ラベル名でフィルタリングします。トリガー マッピングを作成する際に、ロボットのリストをフィルタリングできます。
- 右側の ↻ 更新アイコンをクリックして、表示された情報を更新します。
- 右側の ↺ リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

ユーザー

ユーザー マッピングごとに次の情報が表示されます。

列	説明
ロボット名	ユーザー マッピングのベースとなるロボットの名前。
トリガー	関連付けられているトリガーの名前。
プロジェクト名	ユーザー マッピングが属しているプロジェクトの名前。

列	説明
ユーザー名	ロボットがマッピングされているユーザー。

ラベル

ラベル マッピングごとに次の情報が表示されます。

列	説明
ロボット名	ラベル マッピングのベースとなるロボットの名前。
プロジェクト名	ラベル マッピングが属しているプロジェクトの名前。
ラベル	ロボットがマッピングされているラベル。

ユーザー マッピングの作成

1. [ユーザー] タブの + 記号をクリックします。
   「トリガー ユーザー マッピングを追加」ダイアログ ボックスが表示されます。
2. [ロボット] タブで、マッピングのベースとなるロボットを選択します。[ユーザー] タブで、ロボットをマッピングするユーザーを選択します。[**OK**] をクリックします。
   新しいマッピングがテーブルに追加されます。複数のロボットを選択した場合は、複数の新しいマッピングが追加されます。

ラベル マッピングの作成

1. [ラベル] タブの + 記号をクリックします。
   「トリガー ラベル マッピングを追加」ダイアログ ボックスが表示されます。
2. マッピングのベースとなるロボットを選択し、ロボットをマッピングするラベルを追加します。[**OK**] をクリックします。
   新しいマッピングがテーブルに追加されます。複数のロボットを選択した場合は、複数の新しいマッピングが追加されます。

さらに、⋮ コンテキスト メニューからトリガー マッピングに対して次のアクションを実行できます。

- [編集]:「トリガー ユーザー マッピングを追加」ダイアログ ボックスと同じフィールドが含まれます。

Management Console からトリガー マッピングを削除するには、テーブルでトリガー マッピングを選択し、左上隅の 🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

OAuth

このセクションは、次のロールを持つユーザーが利用できます: 管理者、プロジェクト管理者、開発者、および Kapplet サービス ユーザー。

このセクションには、Management Console を使用して認証される OAuth アプリケーションとユーザーが含まれます。ユーザーの資格情報をスケジュール内のロボットの入力として使用することで、ロボットはユーザー名とパスワードにアクセスしなくても、認証されたユーザーの代わりに API にアクセスできるようになります。

OAuth によって保護されている API にアクセスするロボットの作成および管理方法についての詳細は、OAuth セクションを参照してください。

この OAuth セクションには、次の 2 つのタブが含まれます。[アプリケーション] と [ユーザー] という 2 つのタブが含まれます。

- 新しい OAuth アプリケーションを追加するには、アプリケーションの追加を参照してください。
- 既存の OAuth アプリケーションに新しいユーザーを追加するには、ユーザーの追加を参照してください。

[OAuth] セクションの上部にある [プロジェクト] ドロップダウン リストで、表示する OAuth アプリケーションを含むプロジェクトを選択できます。

各アプリケーションまたはユーザーの情報の表示方法は、次のように変更できます。

- 表示するテーブル列を選択するには、右側の メニュー アイコンを使用します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

アプリケーション

デフォルトでは、アプリケーションごとに次の情報が表示されます。

列	説明
名前	アプリケーションの名前。
サービス プロバイダ	Web サービスのプロバイダ。
プロジェクト名	アプリケーションが属するプロジェクト名 (すべてのプロジェクトを表示する際に便利)。
変更者	アプリケーションを最後に変更したユーザーのユーザー名。
コミット メッセージ	コミットについて説明した概要。
リビジョン番号	アプリケーションのリビジョンの回数。

ユーザー

デフォルトでは、ユーザーごとに次の情報が表示されます。

列	説明
名前	ユーザーの名前。
アプリケーション	ユーザーが属するアプリケーション。

OAuth アプリケーションの追加後は、Management Console のスケジュールまたは REST のリクエストを使用してロボットを起動する際に OAuth アプリケーションとユーザー名を指定することで、OAuth アプリケーション情報をすべて使用できます。

パスワード ストア

この機能は、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、およびパスワード ストア クライアント。

パスワード ストアは、機密情報を開示せずにさまざまなシステムへのアクセス権を付与するパスワード リポジトリです。パスワード ストア機能には、使用可能な割り当て済みのパスワード エントリが表示されます。ここでパスワードを作成および変更できます。

パスワード ストアには、次の 2 つのタブが含まれます。

- パスワード (パスワードの管理 を参照してください。)
- パスワード アクセス (パスワード アクセスの管理 を参照してください。)

エントリを表示

各タブの上部にある [プロジェクト] ドロップダウン リストで、表示するパスワード ストア エントリのあるプロジェクトを選択します。

各パスワード ストア エントリに関する情報の表示方法を次のように変更します。

- エントリを表示するテーブル列を選択するには、右側の メニュー アイコンを使用します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下のナビゲーション メニューを使用してページ間を移動します。

パスワード ストアの使用

パスワード ストアを使用する一般的な手順は次のとおりです。

1. 選択した Management Console で、**[リポジトリ]** > **[パスワード ストア]** に移動し、パスワード タブでパスワード エントリを作成します。
2. Design Studio で、**[設定]** > **[Design Studio 設定]** に移動します。
3. **[Management Consoles]** タブで、パスワードを保存する Management Console を指定し、**[プライマリとして使用]** を選択します。
4. ロボットを展開する準備ができたら、ロボットを Management Console にアップロードします。
5. ロボットをアップロードしたら、Management Console に移動し、[パスワード アクセス] タブを選択します。
6. 次のいずれかの方法で、ロボットの新しいパスワード アクセス エントリを作成します。
   - ロボットのコンテキスト メニューで **[リソース アクセス トークンを取得]** をクリックして、ロボットのアクセス トークンを取得して使用します。
   - 手順 1 で作成した同じパスワード エントリを使用します。

ロボット、またはタイプ、スニペットなどのコンポーネントを Management Console にアップロードするたびに、ロボットの新しいパスワード アクセス エントリを作成する必要があります。以前のエントリはパスワード アクセス リストに保持されますが、これらは手動で削除できます。

パスワードの管理

このタブでは、ターゲット システムにアクセスするためのパスワードを作成および変更します。

該当する [「フィルタ」] テキスト フィールドにエントリを入力して、ユーザー名またはターゲット システムのリストを検索およびフィルタします。詳細については、「フィルタリング 」を参照してください。

[パスワード ストア]の一般的な使用手順については、パスワード ストアの使用を参照してください。

デフォルトでは、パスワード エントリごとに次のテーブル列が表示されます。

列	説明
ユーザー名	ターゲット システムにアクセスするユーザー。
ターゲット システム	ターゲット システムの説明。
プロジェクト名	パスワード エントリを含むプロジェクト。

新しいパスワードを作成する

1. [リポジトリ] メニューから [パスワード ストア] を選択します。
2. [パスワード] タブを選択します。
3. + 記号をクリックして、新しいパスワードを作成します。
   「パスワード エントリ」ダイアログ ボックスが表示されます。
4. 次のフィールドに入力します。
   - プロジェクト: パスワード エントリを含めるプロジェクトを選択します。
   - ユーザー名: ターゲット システムにアクセスするためのユーザー名を入力します。
   - ターゲット システム: アクセスするシステムの説明を入力します。これは、このターゲット システムを他のターゲット システムと区別するために必要です。

     パスワード取得 ステップを挿入する際に、[ターゲット システム] の値が、パスワード エントリの [ターゲット システム] の値と一致している必要があります。

   - [パスワード]: パスワードを入力します。
   - [パスワードを再入力してください]: パスワードを再入力します。
5. **[OK]** をクリックしてパスワードを保存します。
   新しいパスワード エントリがテーブルに表示されます。

パスワードを変更する

⋮ コンテキスト メニューからパスワード エントリに対して次のアクションを実行します。

- 編集: [「パスワード入力」] ダイアログ ボックスで同じフィールドを編集します。
- パスワード エントリを移動: ドロップダウン リストからプロジェクトを選択して、パスワード エントリをプロジェクトから別のプロジェクトに移動します。パスワード ストア アクセスはプロジェクトベースです。アクセス権のあるプロジェクトに割り当てられたパスワード エントリを表示できま

す。Management Console 管理者は、プロジェクト間でパスワード エントリを移動して、プロジェクト管理者がパスワード エントリを管理し、ターゲット システムへのアクセス権を付与することができるようにできます。

- パスワード エントリを移動: [エントリの結合] オプションを選択して、同一のユーザー名とターゲット システムを持つエントリを組み合わせます。

パスワード エントリを削除するには、テーブルでパスワード エントリを選択し、ごみ箱アイコンをクリックします。削除を確認するプロンプトが表示されます。

パスワード アクセスの管理

このタブで、パスワード アクセス エントリを作成および変更します。

例: このオプションを使用して、ロボットへのアクセス権を付与します。

[パスワード ストア]の一般的な使用手順については、パスワード ストアの使用を参照してください。

デフォルトでは、パスワード アクセス エントリごとに次のテーブル列が表示されます。

列	説明
パスワード アクセス トークン	特定のロボットのリソース アクセス トークン。
説明	テーブルのエントリを識別する説明。
プロジェクト名	パスワード アクセス エントリを含むプロジェクト。
パスワード エントリ	アクセスを許可するパスワード エントリ。

新しいパスワード アクセスの作成

1. [リポジトリ] メニューから [パスワード ストア] を選択します。
2. [パスワード アクセス] タブを選択します。
3. + 記号をクリックして、新しいアクセス エントリを作成します。
   「パスワード アクセス エントリ」ダイアログ ボックスが表示されます。
4. 次のフィールドに入力します。
   - プロジェクト: パスワード エントリを含めるプロジェクトを選択します。
   - [パスワード アクセス トークン]: ロボット (Management Console にアップロードする必要があります) へのアクセスを許可するには、[ロボット] セクションに移動して、必要なロボットのコンテキスト メニューから [リソース アクセス トークンを取得] をクリックし、トークンをコピーして [パスワード アクセス トークン] フィールドに入力します。
   - [説明]: リスト内のエントリを識別するための説明を入力します。
   - [パスワード エントリ]: パスワード ストアへのアクセスを許可するユーザー名またはターゲット システムを選択します。
5. [**OK**] をクリックしてパスワード アクセスを保存します。
   新しいパスワード アクセス エントリがテーブルに表示されます。

パスワード アクセスの変更

パスワード アクセスを変更するには、次のアクションを実行します。

- ⋮ コンテキスト メニューから [編集] を使用して、[パスワード アクセス エントリ] ダイアログ ボックスと同じフィールドを変更します。
- パスワード アクセス エントリを削除するには、テーブルでエントリを選択し、🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

CyberArk

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、およびプロジェクト管理者。

CyberArk は、Kofax RPA でサポートされるサードパーティの情報セキュリティ ソフトウェアです。CyberArk は外部パスワード マネージャとして機能します。Kofax RPA の組み込みパスワード マネージャの代わりに使用してください。

CyberArk をインストールするには、『Kofax RPA インストール ガイド』を参照してください。

CyberArk を設定するには、「CyberArk 構成」および「CyberArk アプリケーション」を参照してください。

パスワード ストアと同様に、CyberArk は、機密情報を開示せずに各種システムにアクセス権を付与するために設計されています。[CyberArk] セクションには、使用可能な割り当て済みのパスワード ストア エントリが表示されます。ここでは、新規エントリの作成および既存のエントリの編集を行うことができます。

このセクションには 2 つのタブがあります。

- パスワード
- パスワード アクセス

CyberArk の使用

以下は、CybeArk を使用するための一般的なステップです。

1. 選択した Management Console で、**[リポジトリ]** > **[CyberArk]** に移動し、[パスワード] タブでパスワード エントリを作成します。
2. Design Studio で、**[設定]** > **[Design Studio 設定]** の順に移動し、[**Management Consoles**] タブで Management Console を指定してパスワードを保存して、[プライマリとして使用] を選択します。
3. ロボットの展開準備が整ったら、Management Console にアップロードする必要があります。ロボットがアップロードされた後に、Management Console の [パスワード アクセス] タブで、ロボットのアクセス トークン (ロボットのコンテキスト メニューで **[リソース アクセス トークンを取得]** をクリックして取得可能) を使用して、アップロードされたロボットの新しいパスワード アクセス エントリを作成します。また、手順 1 で作成したパスワード エントリを使用します。

❗ ロボット、またはタイプ、スニペットなどのコンポーネントを Management Console にアップロードするたびに、ロボットの新しいパスワード アクセス エントリを作成する必要があります。以前のエントリはパスワード アクセス リストに保持されますが、これらは手動で削除できます。

パスワード

このタブでは、特定のターゲット システムにアクセスするためのパスワード エントリを作成できます。

デフォルトでは、パスワード エントリごとに次のテーブル列が表示されます。

列	説明
ユーザー名	ターゲット システムにアクセスするユーザーのユーザー名。
ターゲット システム	アクセスするターゲット システムの説明。
プロジェクト名	パスワード エントリを含むプロジェクト。
アカウント名	必須の CyberArk アカウント名に対応するアカウント名。
アプリケーション ID	CyberArk アプリケーション内のアプリケーション。
安全	CyberArk アカウント エントリの安全な名前。

新しいパスワード エントリの作成

1. 新しいパスワード エントリを作成するには、+ 記号をクリックします。
   「パスワード エントリ」ダイアログ ボックスが表示されます。
2. 以下のフィールドを記入します。
   - プロジェクト: パスワード エントリを含めるプロジェクトを選択します。
   - ユーザー名: 指定のターゲット システムにアクセスするためのユーザー名を入力します。
   - ターゲット システム: アクセスするシステムの説明が表示されます。これは、このターゲット システムを他のターゲット システムと区別するために必要です。

     パスワード取得 ステップを挿入する際に、ステップの [ターゲット システム] のプロパティ値が、パスワード エントリの [ターゲット システム] の値と一致している必要があります。

   - アプリケーション **ID**: CyberArk キーストア リストからアプリケーションを選択します。
   - 安全: CyberArk アカウント エントリの安全な名前を入力します。
   - アカウント名 : 必須の CyberArk アカウント名に対応するアカウント名を入力します。
3. [保存] をクリックします。
   新しいパスワード エントリがテーブルに表示されます。

さらに、⋮ コンテキスト メニューからパスワード エントリに対して次のアクションを実行できます。

- 編集:「パスワード入力」ダイアログ ボックスと同じフィールドが含まれます。
- パスワード エントリを移動: あるプロジェクトから別のプロジェクトにパスワード エントリを移動します。パスワード ストア アクセスはプロジェクトベースです。アクセス権のあるプロジェクトに割り当てられたパスワード エントリを表示できます。Management Console 管理者は、プロジェクト間でパスワード エントリを移動して、プロジェクト管理者がパスワード エントリを管理し、ターゲット システムへのアクセス権を付与することができるようにできます。[エントリの結合] オプションを使用

して、同一のユーザー名、ターゲット システム、アプリケーション ID、アカウント名、安全性を持つエントリをマージします。

パスワード エントリを削除するには、テーブルでパスワード エントリを選択し、🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

パスワード アクセス

このタブでは、パスワード アクセス エントリを作成して、特定のロボットにパスワード ストアへのアクセスを許可するなどの操作を実行できます。

デフォルトでは、パスワード アクセス エントリごとに次のテーブル列が表示されます。

列	説明
パスワード アクセス トークン	特定のロボットのリソース アクセス トークン。
説明	テーブルのエントリを識別する説明。
プロジェクト名	パスワード アクセス エントリを含むプロジェクト。
パスワード エントリ	アクセスを許可するパスワード エントリ。

新しいパスワード アクセス エントリの作成

1. 新しいアクセス エントリを作成するには、+ 記号をクリックします。
   「パスワード アクセス エントリ」ダイアログ ボックスが表示されます。
2. 以下のフィールドを記入します。
   - プロジェクト: パスワード エントリを含めるプロジェクトを選択します。
   - [パスワード アクセス トークン]: 特定のロボット (Management Console にアップロードする必要があります) へのアクセスを許可するには、ロボット セクションに移動して、必要なロボットのコンテキスト メニューから [リソース アクセス トークンを取得] をクリックし、トークンをコピーしてここに入力します。
   - [説明]: リスト内のエントリを識別するための説明を指定します。
   - [パスワード エントリ]: パスワード ストアへのアクセスを許可するユーザー/ターゲット システムを選択します。
3. [保存] をクリックします。
   新しいパスワード エントリがテーブルに表示されます。

さらに、⋮ コンテキスト メニューからパスワード エントリに対して次のアクションを実行できます。

- 編集:「パスワード アクセス エントリ」ダイアログ ボックスと同じフィールドが含まれます。

パスワード アクセス エントリを削除するには、テーブルでエントリを選択し、🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

ロボット ファイル システム

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、および RoboServer。

Robot File System セクションには、ロボットが使用または提供するデータの共有および保存用に設定されたファイル システムのリストが表示されます。新たにファイル システムの設定を追加するには、必要な資格情報を提供し、そのファイル システムにアクセスできるロボットとユーザーのリストを定義します。

ファイル システムは、ローカルの Windows フォルダ、Windows ファイル共有、SFTP、および FTP サーバーにすることができます。FTP の場合、FTPS がサポートされています。Robot File System にアップロードまたはダウンロードできるファイルの最大サイズは、使用可能なメモリによって制限されます。

ロボット ファイル システムのサーバーを構成する方法に関する詳細については、『Kofax RPA 管理者ガイド』の「ロボット ファイル システム サーバーのセットアップ」を参照してください。また、このガイドには、一般的な設定と使用手順の例が含まれています。

デフォルトでは、**admin**、**RPA Administrator**、**Project Administrator** のロールを持つユーザーのみがロボット ファイル システムを管理できます。ユーザー ロールの詳細については、ユーザーおよびグループを参照してください。

[ロボット ファイル システム] セクションの上部にある [プロジェクト] ドロップダウン リストで、表示するファイル システムを含むプロジェクトを選択できます。次のように、各設定ファイル システムに関する情報の表示方法を変更できます。

- [フィルタ] テキスト フィールドにフィルタを適用して、テーブル内のファイル システムのリストをフィルタリングすることができます。詳細については、フィルタリングを参照してください。
- 右側の メニュー アイコンを使用して、ファイル システムに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、設定したファイル システムごとに次の情報が表示されます。

列	説明
名前	ロボット データの保存および共有用に設定されたファイル システムの名前。
プロジェクト名	ファイル システムが関連付けられたプロジェクト。
パス	ファイル システムへのパス。
ユーザー名	ファイル システムにアクセスできるユーザー。
プロジェクト範囲	選択されている場合、このファイル システムはプロジェクトのすべてのロボットからアクセスできます。選択されていない場合、ファイル システムは設定で選択された特定のロボットのみからアクセスできます。
オプションの列	
ID	ロボット データの保存および共有用に設定されたファイル システムの ID。

新しいロボット ファイル システムの設定

1. 新しいファイル システムを追加するには、左上隅の + 記号をクリックします。
   複数の新しいタブが開きます。

2. [一般情報] タブで、次の情報を指定します。
   - [プロジェクト]: ファイル システムを含めるプロジェクトを指定します。
   - [プロジェクト範囲]: このオプションを選択すると、プロジェクトのすべての口ボットがファイル システムにアクセスできるようになります。または、ステップ 3 に示すように、特定の口ボットのみがファイル システムにアクセスできるようにすることもできます。
   - [ファイル システム名]: 新しいファイル システム設定の名前を入力します。
     例：`RFS1`
   - [パス]: ファイル システムのパスを指定します。

ファイル システム	説明	パスの例
Windows		`Folder\Subfolder1\Subfolder2`
Windows ファイル共有	ファイル共有は **[名前]** フィールドに指定された口ボット ファイル システムにマッピングされます。サブフォルダは使用できません。	`\\WindowsServer\FileShareFolder`
FTP または FTPS サーバー	パスは `ftp://` または `ftps://` で始まるようにする必要があります。	`ftp://website.com:9551/guest`
	FTP クライアントは、FTP サーバーが UTF-8 エンコーディングをサポートしているかどうか検出を試行します。この自動検出をオーバーライドするには、`utf8:`*`option`* を追加します。ここで *`option`* は次のとおりです。 • `on` で UTF-8 サポートを強制します • `off` で UTF-8 サポートを無効にします • `auto` で自動検出を実行します	`ftp://website.com:9551/guest,utf8:on`
SFTP サーバー	パスは `sftp://` で始まるようにする必要があります。	`sftp://ssh.domain.com:9551/guest`
	SSH キー認証を使用するには 2 つの方法があります: • **[パス]** フィールドに `publickey` を追加して、Robot File System サーバー上で設定されたデフォルトの SSH キーを使用します。 • `publickey:<path-to-the-private-key>` を **[パス]** フィールドに追加することで、プライベート SSH キーを使用して認証します。 DSA、Ed25519、および ECDSA キーがサポートされています。RSA キーは廃止予定であるため、置き換える必要があります。 SSH キー認証が設定されている場合、[パスワード] は無視されます。	`sftp://ssh.domain.com:9951/guest,publickey:c:\keys\ssh-domain-com_ed25519.key`

   - [ユーザー名] と [パスワード]: ファイル システムにアクセスするための資格情報を入力します。
     Windows ユーザー名は `username@domain` の形式に適合している必要があります。
     クレデンシャルは、口ボットにファイル システムに対する書き込みまたは読み取りを許可する保護されたサービスでのみ使用されます。

- 変更を保存します。

3. プロジェクト内の特定のロボットからファイル システムにアクセスできるようにするには、名前でマッピングするか、ロボットへのアクセス トークンを追加します。この方法はロボット名が変わらない限り機能します。⋮ をクリックして作業中のファイル システムのコンテキスト メニューを開き、[編集] をクリックします。
   - ロボットの現在および将来のすべてのバージョンがファイル システムにアクセスできるようにするにはロボット名を使用します。

     [ロボット マッピング] タブで、ファイル システムへのアクセス権を持つロボットを選択します。このタブに表示されるようにするには、ロボットを Management Console と同期する必要があります。

     変更を保存します。
   - 現在のバージョンのロボットのみがファイル システムにアクセスできるようにし、ロボットに対する変更がファイル システムに書き込まれないようにするには、ロボット アクセス トークンを使用します。トークンは特定バージョンのロボットに対応するため、ロボットに変更があると、ファイル システムへの変更はできなくなります。
     a. Management Console のナビゲーション メニューの **[ロボット]** セクションで、⋮ をクリックして必要なロボットのコンテキスト メニューを開き、**[リソース アクセス トークンを取得]** を選択します。

        新しいダイアログ ボックスが表示されます。
     b. トークンをコピーしてダイアログ ボックスを閉じます。
     c. [ロボット ファイル システム] セクションに戻り、ファイル システムをクリックして編集し、[認証されたアクセス トークン] タブで、コピーしたアクセス トークンを追加します。
4. 変更を保存します。

ファイル システムを削除するには、テーブルでファイル システムを選択し、左上隅の 🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

データ ビュー

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、開発者、および VCS サービス クライアント。

「データ ビュー」 メニューを使用すると、プロジェクトのデータベース データを表示したり、ロボットによって抽出されたデータを エクスポートしたりできます。このビューを使用して、Design Studio または Management Console を使用するタイプから作成されたデータベース テーブルを表示することもできます。

プロジェクトに定義されたデータベース マッピングのテーブルの 1 つを表示するには、メニューの上部にある [データベース ナビゲーション] をクリックし、ドロップダウン リストから プロジェクト > データベース マッピング > テーブル を選択します。

データベース接続が確立されていない場合、またはデータベースのテーブルが作成されていない場合、データベース マッピング名は非アクティブとなります (展開できません)。

データベース マッピング名をクリックすると、ツリーが展開され、データベースに含まれるさまざまなテーブルが表示されます。テーブルを選択すると、タイプの名前、それらのタイプに関連付けられたロボット、タイプ属性などのタイプ データが右側ペインに表示されます。テーブルからデータを直接コ

ピーすることができます。右側の メニュー アイコンを使用して、データベース テーブルに表示する列を選択できます。

データが読み込まれると、ログ ビューのフィルタと同じように機能する一部のフィルタが利用可能になります。たとえば、「含む」、「含まない」、「いずれか」、「次で始まる」などの条件でフィルタリングを行うことができます: `binary` および `longtext` 属性をフィルタすることはできません。

データのエクスポート

データ グリッドの上にある [エクスポートの保存場所:] ドロップダウン リストで、ロボットによって抽出されたデータをエクスポートする形式を[**XML**]、[**CSV**]、または [**Excel**] の中から選択できます。

以前のエクスポートを表示したり、再度ダウンロードしたりするには、[エクスポート] をクリックします。

デフォルトでは、エクスポートごとに次の情報が表示されます。

列	説明
作成済み	エクスポートが作成された日付。
テーブル	エクスポートが作成されたテーブルの名前。
データベース	テーブルが属しているデータベースの名前。
カタログ	テーブルが属しているデータベース カタログ (該当する場合)。
ダウンロード	エクスポートをダウンロードする場合にクリックします。

エクスポートは、次回 Management Console を起動したときに自動的に削除されます。エクスポート数が 100 件を超えると、最も古いエクスポートが削除されます。CSV または XML にエクスポートできる行の数に制限はありませんが、システムがメモリ不足にならないように、Excel ファイルは 10000 レコードに制限されています。

ナビゲーション ツリーの各データベースの下に表示されるスキーマ/カタログのリストは、クラスタ設定 データベースの構成で使用されるクレデンシャルを持つユーザーの権限によって制御されます。

ログ ビュー

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、および開発者。

[ログ ビュー] メニューには、Management Console および RoboServer のデータベース ロギングによって作成されたログが含まれます。Management Console ログと RoboServer ログがどちらもログ データベースに書き込まれるため、Management Console 設定でログ データベースを設定し、RoboServer でロギング クラスタ設定を使用してデータベース ロギングを有効にする必要があります。

[スケジュール実行] および [スケジュール メッセージ] は、スケジュールに基づいて情報を報告する Management Console のログです。残りのログの [ロボット実行]、[ロボット メッセージ]、[ロボットの概要]、[**RoboServer** メッセージ]、[**DAS** メッセージ]、および [タスク メッセージ] は、RoboServer のステータスおよびロボットとロボットの実行に関する情報を含む RoboServer のログです。

すべての実行とメッセージは統一されたタイム ゾーンに従ってログに記録されるため、Management Console、Desktop Automation サービス、および RoboServer で設定されたタイム ゾーンは考慮されません。

各ログのページ レイアウトは同一ですが、それぞれのログの情報の表示方法は次のように変更できます。

- [フィルタ] テキスト フィールドにフィルタを適用して、テーブル内のメッセージのリストをフィルタリングすることができます。フィルタ アイコンをクリックすると、「列フィルタ」 ダイアログ ボックスに一部のフィルタが表示されます。

  たとえば、ログの種類に応じて、既存のプロジェクトやオーファン プロジェクトでフィルタリングしたり、時間枠、合計実行時間、実行結果、エラー メッセージの重大度などでフィルタリングしたりすることができます。

  また、フィルタリングを参照してください。

  「ログ ビュー」フィルタでの日付の表示形式は、Management Console の設定言語によって異なります。英語に設定されている場合、ブラウザの言語設定を使用して英語の方言が決定されます。たとえば、言語リストの一番上に [英語 (英国)] と表示されている場合、フィルタの日付はイギリス英語の標準の日付形式に従って書式設定されます。ブラウザの設定で何も指定がない場合は、デフォルトで英語 (アメリカ合衆国) が使用されます。

- 右側の メニュー アイコンを使用して、ログに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。ページごとに選択した件数以上の結果がある場合には、データ グリッド下のコントロールを使用して次のページに移動することができます。

RoboServer ログ データベースでログの保持ポリシーを設定するには、ログを保持する日数とロボット実行におけるメッセージの数を指定します。ログは、最も古いメッセージを削除することで毎日消去されます。デフォルトの値は、10 日と 500 件のメッセージに設定されています。

[スケジュール実行]

スケジュールの開始時および終了時など、実行する各スケジュールの実行情報が表示されます。

コンテキスト メニューを使用して、個別のスケジュール メッセージ、またはこのスケジュール実行の一部として実行されたロボットに移動します。

コンテキスト メニューを使用して、スケジュール実行を削除できます。削除する場合、メッセージは必ず削除する必要がありますが、実行情報は必要に応じて保持できます。この実行およびメッセージ、または現在のフィルタに一致するすべての実行またはメッセージを削除することができます。多数の実行またはメッセージの削除には、しばらく時間がかかることがあります。

[スケジュール メッセージ]

指定のスケジュール実行の個別のメッセージ エントリを表示します。スケジュール実行が失敗した理由に関する詳細情報を確認できます。

指定したスケジュール実行のロボットのエラーを調べるには、 コンテキスト メニューから [エラーのあるロボットを表示] を選択します。

1 つ以上のレコードを削除できます。削除する場合は、このメッセージのみ、現在のフィルタに一致するすべてのメッセージ、またはすべての RoboServer ログ メッセージを削除するように選択できます (一致する結果が 500 件を超える場合は、パフォーマンス上の理由から、フィルタに一致するメッセージを削除するオプションは無効になります)。

重要度フィルタで値を選択した場合、結果は選択した値以上となることに注意してください。たとえば、重要度フィルタを「警告」に設定した場合、結果には警告の他にエラーや致命的なエラーも含まれます。

[ロボット実行]

各ロボット実行の情報を表示します。⋮ コンテキスト メニューでは、同じロボットの実行をすべて表示したり、この実行が実行されたときにこのロボットが受け取った入力を表示したりすることもできます。

実行とメッセージを削除できます。削除する時には常にメッセージを削除する必要がありますが、必要であれば実行情報を維持することができます。この実行およびメッセージ、または現在のフィルタに一致するすべての実行またはメッセージを削除することができます。多数の実行またはメッセージの削除には、しばらく時間がかかることがあります。

「ログ ビュー」でロボットの入力を表示する場合は、「ロギング」の「ロボットの入力をデータベースに記録」を参照してください。

[ロボット メッセージ]

ロボット実行に属する個別のエラー メッセージを表示します。⋮ コンテキスト メニューを使用すると、このメッセージが属する実行に移動したり、Design Studio でロボットを開いてエラーの原因となったステップを検索したりできます。

また、[ロケーション コード] 列からステップのロケーション コードをコピーし、それを Design Studio で使用してエラーの原因となったステップに直接移動することもできます。

- ロボットの場合は、ツールバーの をクリックします。
- ベーシック エンジン ロボットの場合は、[編集] メニューの **[ステップを移動]** > **[指定したロケーションへ移動]** をクリックします。

> エラーをログに記録する必要があるすべてのステップで、[エラーとしてログ記録] オプションが有効になっていることを確認してください。Design Studio では、ステップ プロパティの [エラー処理] タブで [エラーとしてログ記録] が選択されていることを確認します。
>
> RQL API を使用している場合に、ステップ エラーが API 経由で返され、Management Console にログ記録されるようにするには、[エラー処理] タブの [**API** 例外] オプションも選択されていることを確認してください。
>
> デフォルトでは、両方のオプションが選択されています。

重要度フィルタで値を選択した場合、結果は選択した値以上となることに注意してください。たとえば、重要度フィルタを「警告」に設定した場合、結果には警告の他にエラーや致命的なエラーも含まれます。

[ロボットの概要]

これは、すべてのロボットのこれまでの実行 (データが取得できる限り) の簡易的な概要ビューです。⋮ コンテキスト メニューを使用して、特定のロボットのすべての実行に移動できます。

[RoboServer メッセージ]

RoboServer または Management Console からの一般的なメッセージを表示します。

1 つ以上のレコードを削除できます。削除する場合は、このメッセージのみ、現在のフィルタに一致するすべてのメッセージ、またはすべての RoboServer ログ メッセージを削除するように選択できます (一致する結果が 500 件を超える場合は、パフォーマンス上の理由から、フィルタに一致するメッセージを削除するオプションは無効になります)。

重要度フィルタで値を選択した場合、結果は選択した値以上となることに注意してください。たとえば、重要度フィルタを「警告」に設定した場合、結果には警告の他にエラーや致命的なエラーも含まれます。

[DAS メッセージ]
Desktop Automation サービスから受信したイベントを表示します。詳細については、Desktop Automation サービスを参照してください。

重要度フィルタで値を選択した場合、結果は選択した値以上となることに注意してください。たとえば、重要度フィルタを「警告」に設定した場合、結果には警告の他にエラーや致命的なエラーも含まれます。

[タスク メッセージ]
キューイング タスクの状態を通知する個々のメッセージを表示します。タスクについて表示されるメッセージ タイプは、キューイング、実行中、完了、タイムアウト、および削除です。すべてのメッセージに対して、[詳細] 列に説明が表示されます。

Desktop Automation サービス、ライセンス ユニット、RoboServer 実行スロットなどの必要なリソースを取得できなかった場合、タスクはタイムアウトします。「タスク ビュー」のスケジュールまたは対応するタスクが停止した場合、RoboServer が停止した場合、あるいは Management Console がオフラインになった場合、タスクは削除されます。

詳細については、ジョブの優先順位の設定を参照してください。

管理

このメニューを使用して、Management Console を管理します。

タスク ビュー

Management Console が実行するロボットとその他のタスクを表示します。

RoboServer

RoboServer とクラスタを追加または削除し、クラスタ設定を構成します。実行中のロボットを表示します。

プロジェクト

プロジェクトを作成および削除します。

高可用性ノード

設定した高可用性ノードを表示します。

デバイス

Management Console で登録した利用可能なオートメーション デバイスを表示します。

ユーザーおよびグループ

新しいユーザー グループを作成します。Management Console ユーザーに関する情報を表示します。

サービス認証

Management Console でユーザーを認証する共有シークレットを生成し、追加のセキュリティを提供します。

バックアップ

バックアップを作成および復元し、プロジェクトをインポート/エクスポートします。

ライセンス

ライセンス情報を入力するか、現在のライセンスと使用中の Design Studio 接続クライアントに関する情報を表示します。

タスク ビュー

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、およびプロジェクト管理者。

「タスク ビュー」には、すべてのサーバーとロボット全体の現在アクティビティの概要が表示されます。このセクションには、スケジュールされたロボットと、Management Console スケジューラによって開始された事前処理タスクおよび事後処理タスクが表示されます。過去のロボット実行の結果については、ログを確認してください。

更新の遅延によって、実行時間の短いロボットがこのビューに表示されない (または、ほとんど表示されない) 場合があります。

「タスク ビュー」セクションの上部にある [クラスタ] ドロップダウン リストで、表示するロボットとタスクを含むクラスタを選択できます。次のように、情報の表示方法を変更できます。

- 表示するテーブル列を選択するには、右側の メニュー アイコンを使用します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、現在実行中のすべてのロボットに次のテーブル列が表示されます。

列	説明
名前	実行しているタスク名。
スケジュール	タスクが実行されているスケジュールの名前。
プロジェクト名	タスクが属しているプロジェクトの名前。

列	説明
ステータス	タスクは、以下の状態のいずれかになります。 キューに登録済み タスクはキューに登録され、実行を待っています。タスクは、以下の理由によってキューに登録されています。 • サーバーなし 現在、クラスタにサーバーはありません。または、すべてのサーバーはオフラインです。タスクは、サーバーが追加された際またはオフラインのサーバーがオンラインに切り替わった際に実行を開始します。 • キャパシティなし すべてのサーバーは最大キャパシティで実行しています。タスクは、その他のタスクが終了して、キャパシティが利用できるようになった際、またはその他のサーバーがクラスタに追加された際に実行を開始します。 • その他のタスク待ち タスクは、その他のタスクが終了するのを待っています。事後処理は、すべての口ボットが終了するまで実行されません。また、口ボットは、事前処理が終了するまで実行されません。また、スケジュールの口ボットが連続実行に設定されている場合、口ボットは、以前の口ボットが完了するまでキューに登録された状態に留まります。 実行中 タスクは現在実行中です。 取得済み タスクは RoboServer に送信済みですが、実行は開始されていません。
最後のステータス変更	最後のステータス変更の時間。
入力値	実行中の口ボットの入力の概要。
優先度	他のタスクのキューにおける、タスクの実行優先順位 (最小、低、中、高、または 最大)。
満期	タスク キューイングがタイムアウトに達するまでの残り時間。

RoboServer

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、KTA クライアント、および RoboServer。Kapplet サービスのユーザーは、クラスタと設定を表示できます。プロジェクト管理者はクラスタと設定を変更できません。

[RoboServer] セクションでは、Management Console で認識されているクラスタと RoboServer を管理できます。RoboServer は自動的に Management Console に登録され、オンラインになると RoboServer のリストに表示され、オフラインになるたびにリストから削除されます。組み込みモードの場合、デフォルトでは、リストには 1 つの RoboServer を持つ 1 つのクラスタが含まれます (このクラスタで Management Console 機能も実行されます)。複数の RoboServer とクラスタがある大規模なセットアップでは、ライセンスで許可されている場合、スタンドアロンの Web コンテナに Management Console を展開することをお勧めします。『Management Console』の設定に関する詳細については、『Kofax RPA 管理者ガイド』を参照してください。

RPA バージョン 11.2 以降では、RoboServer は UTC 時間でログを書き込みます。11.2 より前のバージョンの RoboServer は、デフォルトではローカル サーバー時間でログを書き込みます。そのため、11.2 以降とそれより前のバージョンの両方の RoboServer が同じログ データベースにログを記録すると、タイムスタンプに不整合が生じる可能性があります。11.2 より前のバージョンの RoboServer を Management Console バージョン 11.2 以降に接続する場合は、`RoboServer.conf` ファイル内の次のオプションをコメント解除することで、ローカル サーバー時間ではなく UTC 時間でログ メッセージを書き込むように設定できます。

```
wrapper_java_additional.41=-DwriteLogdbUtc=true
```

RoboServer は、このパラメータをサポートするフィックスパック バージョンに更新する必要があります。詳細については、対応する RPA フィックスパックの ReadMe ファイルを参照してください。

各クラスタ/RoboServer の情報の表示方法を次のように変更できます。

- 右側の メニュー アイコンを使用して、クラスタ/RoboServer に表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。

デフォルトでは、次のテーブル列が各クラスタ/RoboServer に対して表示されます。

列	説明
クラスタ	クラスタの名前。名前の後にタグが続きます。 • **Client**: RoboServer はクライアントとして Management Console に接続します。 • **Service**: RoboServer はサービスとして実行されます。 • **Service SSL**: RoboServer はサービスとして実行され、SSL 接続を使用します。

列	説明
アクション	この列には、使用可能なアクションがリストされます。 クラスタの場合 • クラスタ設定: クラスタ設定のタブを開きます。 • **RoboServer** に **CRE** を割り当て または **RoboServer** に **KCU** を割り当て: [ライセンス ユニットの割り当て] ペインを開きます。ここで、クラスタ内の各 RoboServer の CRE または KCU ライセンス ユニットの数を手動で設定できます。 RoboServer のライセンス ユニット数を 0 (スライダーの一番左の位置) に設定すると、この RoboServer がライセンスの自動配布を使用することを示します。このデフォルト設定は、クラスタに設定されたライセンス ユニットが、クラスタ内のオンライン RoboServer 間で均等に配布されることを意味します。 スライダーを動かして、希望するライセンス ユニット数を手動で設定します。この数は `licenseLimit` パラメータよりも優先されます。 • 削除: クラスタを削除します。 **RoboServer** の場合 • スレッド ダンプ: 選択した RoboServer にリクエストを送信して、完全なスレッド ダンプを実行します。 • **RoboServer** を停止: 選択した RoboServer を停止/再開するダイアログ ボックスを開きます。RoboServer をシャットダウンする方法を選択し、必要に応じてシャットダウンのタイムアウトを指定して、[**OK**] をクリックします。タイムアウトはミリ秒単位で指定します。-1 に設定すると、シャットダウンが即時に強制実行されます。 • すべての口ボットを停止: 選択した RoboServer で実行中のすべての口ボットを停止します。 • 削除: 選択したオフライン RoboServer を削除します。
サーバー	RoboServer の名前または IP アドレス、ポート、および ID 番号。
バージョン	実行中の RoboServer 上のソフトウェアのバージョン。
ステータス	クラスタの場合、クラスタの状態が表示されます。 • 初期化中 • シャット ダウン中 • 実行中 • 一時停止中 • 無効な設定 • RoboServer に設定を送信中 • スケジュールと口ボットが終了するまで待機します • 現在実行中の口ボットが終了するまで待機します • すべての口ボットとスケジュールを停止し、すぐに設定を適用します クラスタ設定を変更すると、「クラスタ設定を適用」 ダイアログ ボックスが表示され、続行するためのオプションが示されます。 **RoboServer** の場合、サーバーがオンラインまたはオフラインのいずれであるかが示されます。「Duplicate」のステータスは、同一の RoboServer が 2 つ存在することを示します。RoboServer がオフラインの場合は、⋮ コンテキスト メニューから [削除] をクリックして削除できます。

列	説明
CRE/KCU	CRE および KCU ライセンスの詳細については、『Kofax RPA インストール ガイド』の「同時ロボット実行ライセンス」および「Kofax RPA 計算単位」を参照してください。 **CRE** ベースのライセンスの場合 静的ライセンス分配のモードでは、この列はこのクラスタで同時に実行できるロボットの数を示します。CRE は、クラスタ内のオンライン RoboServer 間で均等に配分されます。CRE は不可欠なユニットであり、1 つの CRE を複数の RoboServer 間で分割することはできません。たとえば、クラスタ内に 6 つの CRE と 5 つの RoboServer がある場合、各 RoboServer は CRE を 1 つ取得するため、1 つの CRE が未使用のままになります。 クラスタ内の CRE の数は、RoboServer の数以上である必要があります。クラスタ内に存在する RoboServer の数よりもクラスタに割り当てた CRE の数が少ない場合、クラスタは無効になります。 動的ライセンスの分配モードでは、この列はクラスタに割り当てられた CRE ライセンスの総数を示します。RoboServer はリクエストごとにクラスタからライセンスを受け取ります。リクエストされた数のライセンスがある場合、RoboServer は、その数のライセンスを取得します。このモードの場合、RoboServer は Management Console とのみ通信し、API 呼び出しなどの他のリクエストをブロックします。 動的ライセンスの分配モードで、Management Console の再起動後に RoboServer と Management Console 間の接続を再確立するには、RoboServer を再起動してください。 RoboServer が実行できる最大同時ロボットの数は、利用可能な CPU の量と、RoboServer が処理するデータを取得するために必要な速度によっても異なります。詳細については、『Kofax RPA 管理者ガイド』の「本番構成」を参照してください。 CRE の数を調整するには、[クラスタに **CRE** を割り当て] をクリックします。このアクションを行うと、[ライセンス ユニットの割り当て] ペインが開き、ライセンス単位の数を調整して、使用可能な単位の総数と残りの数を確認できます。 動的ライセンスの分配モードでは、ライセンス ユニットが自動的に分配されます。これは `licenseLimit` パラメータよりも優先されます。 **KCU** ベースのライセンスの場合 このクラスタに割り当てられている KCU の数を示します。クラスタの KCU は、クラスタ内のオンライン RoboServer 間で均等に配分されます。クラスタの KCU を調整するには、[クラスタに **KCU** を割り当て] をクリックします。このアクションを行うと、[ライセンス ユニットの割り当て] ペインが開き、ライセンス単位の数を調整して、使用可能な単位の総数と残りの数を確認できます。
ライセンス タイプ	クラスタのライセンス タイプ: プロダクションまたは非プロダクション。
オプションの列	
実行中のロボット	現在、RoboServer で実行されている ロボットの数。
キューに格納されたロボット	RoboServer でキューに格納されたロボットの数。

列	説明
最大ロボット数	RoboServer で同時に実行されるロボットの最大数。クラスタ設定で設定できます。
アップタイム	RoboServer のアップタイム。サーバーを起動または再起動した時刻を確認できます。
コマンド ライン	RoboServer を起動したコマンド ライン。
CPU 数	RoboServer プロセスに割り当てられている CPU の数。たとえば、CPU アフィニティが割り当てられている場合などがあります。
メモリ制限	RoboServer が実行されている JVM に割り当てられているメモリの最大容量。
期間 (累計)	RoboServer が「制限を超過」の状態になっていた合計時間を示します。
制限を超過	サーバーがそのメモリ閾値 (デフォルトの 80%) を越えて動作しているかどうかを示します。この制限に到達した場合、RoboServer は、ロボットを起動する代わりに、キューに登録します。
最大キュー数	RoboServer でキューに登録できるロボットの最大数。クラスタ設定で設定できます。
最終更新	Management Console が RoboServer から最新のステータス更新を受け取った時刻を示します。
一時プロファイリング	特定の RoboServer に対してプロファイリングが一時的に有効化されているかどうかを示します。RoboServer が再起動されると、この設定はクリアされます。

新しいクラスタの作成

1. 新しいクラスタを作成するには、左上隅の + 記号をクリックします。
   「クラスタの追加」 ダイアログ ボックスが表示されます。
2. クラスタの名前を指定します。「クラスタの追加」 ダイアログ ボックスを閉じた後は、名前を変更できません。
   本番クラスタを作成する場合は、本番ライセンスからライセンス ユニットを割り当てることができます。同様に、非本番クラスタを作成する場合は、非本番ライセンスからライセンス ユニットを割り当てることができます。
3. RoboServer と Management Console 間の接続タイプを指定します。
4. ライセンスの分配モードを選択します。
5. [**OK**] をクリックします。
   新しいクラスタがテーブルに表示されます。

クラスタにさまざまな Kofax RPA 製品バージョンの RoboServer を含めて、ロボットを徐々に更新することができます。クラスタでは、古いバージョンのロボットは最も近い利用可能なバージョンの RoboServer に転送されます。例については、閾値の RoboServer バージョンを参照してください。

負荷分散とフェールオーバー

クラスタでロボットを実行する必要があるときは、利用可能なスロット数が最も多い RoboServer が検索されます。利用可能なスロット数は、RoboServer ですでに実行されているロボットの数と、同時に実行できるロボットの数 (クラスタ設定で設定した同時ロボットの最大数) に基づいて計算されます。

クラスタ内のいずれかの RoboServer がオフラインになると、KCU が残りの RoboServer 間で自動的に均等に配分されます。

RoboServer またはクラスタのアクション

RoboServer またはクラスタでアクションを実行するには、[アクション] 列の ⋮ コンテキスト メニューを使用します。アクションの説明については、前の表の [アクション] 行を参照してください。

実行中のロボット ビュー

クラスタまたは RoboServer の名前をクリックすると、ビューが開き、選択したクラスタ内のすべての RoboServer からこのクラスタ内でロボットを実行する方法、または選択した RoboServer でロボットを実行する方法についての詳細が表示されます。

このビューの情報の表示方法を次のように変更できます。

- さまざまなフィルタを適用して、テーブル内のロボットのリストをフィルタリングします。ロボット名、プロジェクト名、実行 ID、ロボット URL でフィルタリングすることができます。フィルタリングでは大文字と小文字が区別され、入力したテキストがロボット名、プロジェクト名、実行 ID、またはロボット URL のいずれかの部分文字列として含まれるロボットが選択されます。
- 右側の ||| メニュー アイコンを使用して、表示するテーブル列を選択します。

デフォルトでは、実行中または直前に完了したロボットごとに、次のテーブル列が表示されます。

列	説明
ロボット名	ロボットの名前。
サーバー	ロボットを実行している RoboServer の名前
プロジェクト名	ロボットが属しているプロジェクトの名前。**[リポジトリ]** > **[ロボット]** セクションにプロジェクトのリストが表示されます。
ロボット URL	ロボットを識別する URL。RoboServer の実行要求を作成する場合に、file:// URL または Library:/ を指定できます (これは、ロボットをファイル システムまたはライブラリのいずれからロードする必要があるかどうかを示します)。 • ファイル システムの URL: `file://C:/Kofax RPA/Robots/Library/Input.robot` • ライブラリの URL: `Library:/Input.robot` ロボットの実行要求は、次の例のようになります。 `Request request = new Request("Library:/Input.robot")`
開始時間クライアント	ロボットが開始された時刻。この時刻は、Management Console を実行しているブラウザのタイム ゾーンで表示されます。
実行 ID	ロボットの実行 ID。
現在のステップ	ロボットが現在実行されているステップ。

列	説明

ステータス	ロボットの現在のステータス。
	• 実行中 - 現在実行中。
	• 待機中 - 実行可能になるまで待機中。
	• 完了 - RoboServer での実行が終了しました。このステータスは、次のようなロボットに割り当てられます。
	  • 正常に完了したロボット
	  • エラー状態で完了したロボット
	  • 失敗したロボット
	  • 強制的に停止されたロボット
	完了ステータスを持つロボットは、完了してから 1 分後にテーブルから削除されます。

オプションの列	
ロケーション コード	Design Studio に表示可能なステップに割り当てられているコード。
ステップ実行時間	現在のステップ実行時間 (秒単位)
実行されるステップ数の制限	ロボットが実行できるステップの最大数を表示します。制限に到達すると、ロボットは停止します。
KCU 待機時間	KCU ポイント (その時点での) がすでに使用されているため、ロボットが実行できなかった時間。
ロード済みバイト数	ロボットの実行中にロードされたバイト数。
最終出力時間	最後の抽出が実行された時刻。
送信された電子メール	ロボットが送信した電子メールの数。
実行されたステップ	ロボットが実行したステップの数。
実行パス	ロボットが実行したステップのシーケンス。
KCU ポイント コスト	ロボットを実行するために使用した KCU ポイント。KCU ポイント コストは、Design Studio に示されている KCU 使用量と等しくなります。
抽出された値の制限	オブジェクト抽出数の上限。このプロパティで指定した上限よりも多くのオブジェクトをロボットが抽出した場合、エラー メッセージが生成されるか、ロボットが停止します。
実行時間の制限	ロボット実行の合計時間の上限。この時間制限内にロボットが完了しなかった場合、エラー メッセージが生成され、ロボットが停止します。このプロパティ値は秒単位で指定されます。
出力数	ロボットが生成したオブジェクトの数。
ロボット ライブラリ	ロボット ライブラリのタイプ。次のタイプがあります。 • Design Studio ロボット ライブラリ • 組み込みファイルベースのロボット ライブラリ • リポジトリ ロボット ライブラリ • URL ファイルベースのロボット ライブラリ • URL フォルダベースのロボット ライブラリ 詳細については、『Kofax RPA 開発者ガイド』を参照してください。また、このヘルプ システムの「ロボット ライブラリ」も参照してください。

列	説明
切断時に停止	設定した場合、Management Console との接続が失われると口ボットは停止します。このフラグは、Java API で口ボットを実行する場合にのみ使用します。詳細については、『Kofax RPA 開発者ガイド』の「setStopOnConnectionLost」を参照してください。
API 例外時に停止	設定した場合、API 例外が生成されると口ボットは停止します。
停止中	口ボットがシャットダウン プロセス中かどうかを示します。

クラスタの状態の変更

[ステータス] 列で、クラスタを次のいずれかの状態に設定できます。実行中または一時停止

クラスタの状態	説明
実行中	通常クラスタは、想定通りに実行されるスケジュールおよび個別の口ボットによって動作します。クラスタ設定は RoboServer に適用されないため、スケジュール実行中に設定が変更されることはありません。
一時停止	クラスタは新しい実行要求を受け入れることができず、RoboServer は現在のすべての実行を完了することができません。クラスタ設定は変更できます。たとえば、データベース マッピングは変更できます。

新しい設定をクラスタに適用すると、「クラスタ設定を適用」 ダイアログ ボックスが表示され、続行するためのオプションが示されます。

実行中と一時停止以外にも、クラスタは次のいずれかの状態になります。

- 初期化中
- シャット ダウン中
- 無効な設定
- RoboServer に設定を送信中
- スケジュールと口ボットが終了するまで待機します
- 現在実行中の口ボットが終了するまで待機します
- すべての口ボットとスケジュールを停止し、すぐに設定を適用します

クラスタの状態は、クラスタ レベルにのみ関連します。このようにクラスタの状態は、Management Console がクラスタでタスクを実行できるタイミングを制御する方法です。ただし、個別の RoboServer は制御しません。これは、API から起動された口ボットが、一時停止状態に切り替わる際に停止しないことを意味します。そのため、RoboServer の設定は、API 口ボットの実行中に更新することができます。ただし、口ボットの設定は実行中に更新されないことが保証されています。たとえば、1 つ以上の API 口ボットの実行中にデータベースが更新された場合、口ボットは開始時に設定されたデータベースを使用します。そして、次回の実行時に、口ボットは新しい設定を使用します。

クラスタ設定の構成

クラスタ設定は、クラスタ内の各 RoboServer に送信されます。たとえば、クラスタ設定を使用すると、RoboServer で実行されている口ボットが使用可能なデータベースを設定できます。

[管理] > **[RoboServer]** に移動し、クラスタの **[アクション]** 列で、⋮ コンテキスト メニューをクリックし、**[クラスタ設定]** を選択します。

RoboServer に設定を送信するには、最初に設定を適用する必要があります。設定を変更した後に、[**OK**] をクリックすると設定が自動的に適用されます。その後、オプションを含んだ「クラスタ設定を適用」ダイアログ ボックスが表示されます。この画面で [キャンセル] をクリックしても、適用済みの設定には影響がないことに注意してください。設定を適用した後に、クラスタを手動で実行中の状態にする必要があります。

一般

このタブを使用して、CRE ライセンスの分配と閾値の RoboServer バージョンを定義します。

[ライセンス] と [閾値のバージョン] オプションは、KCU ライセンスが使用されている場合、無効になります。

ライセンスの分配

CRE ライセンスの分配モードを選択します。

オプション	説明
静的	このモードでは、CRE はクラスタ内のオンライン RoboServer 間で均等に分配されます。CRE は不可欠なユニットであり、1 つの CRE を複数の RoboServer 間で分割することはできません。たとえば、クラスタ内に 6 つの CRE と 5 つの RoboServer がある場合、各 RoboServer は CRE を 1 つ取得するため、1 つの CRE が未使用のままになります。 クラスタ内の CRE の数は、RoboServer の数以上である必要があります。クラスタ内に存在する RoboServer の数よりもクラスタに割り当てた CRE の数が少ない場合、クラスタは無効になります。 CRE の数を調整するには、[クラスタに **CRE** を割り当て] をクリックします。この操作により、[ライセンス ユニットの割り当て] ペインが開き、ライセンス ユニットの数を調整して、使用可能なユニットの合計数と残りのユニット数を確認することができます。

オプション	説明
動的	このモードでは、RoboServer はリクエストごとにクラスタからライセンスを受け取ります。リクエストされた数のライセンスがある場合、RoboServer は、その数のライセンスを取得します。このモードの場合、RoboServer は Management Console とのみ通信し、API 呼び出しなどの他のリクエストをブロックします。RoboServer と Management Console の間の接続を確立する際に問題が発生した場合は、サーバー ログをデバッグ レベルに切り替えて、ログを確認してください。詳細については、「ロギング」を参照してください。 動的ライセンスの分配モードは Kofax RPA バージョン 10.2 以降でサポートされています。バージョン 10.7 以降は、インストール直後にこのモードをサポートします。動的ライセンスの分配を使用するには、10.2 から 10.6 までのバージョンに、対応するバージョンの最新のフィックス パックをインストールします。『Kofax RPA アップグレード ガイド』の「動的ライセンスの分配モードを有効にする」を参照してください。 動的ライセンスの分配モードで、Management Console の再起動後に RoboServer と Management Console 間の接続を再確立するには、RoboServer を再起動してください。

閾値の RoboServer バージョン

レガシー ロボットを以後アップグレードしない Kofax RPA のバージョン番号を定義します。ロボットを円滑にアップグレードしたり、テストが実施されていない RoboServer でロボットが実行されないようブロックしたりするのに役立ちます。

製品バージョンが閾値を下回るロボットは、最も新しいバージョンの RoboServer に転送されます。閾値を上回るすべてのロボットは、厳密にバージョンが一致する場合に、RoboServer 間で分配されます。つまり、製品バージョンが閾値より高いロボットのみが、同じ製品バージョンの RoboServer で実行されます。

たとえば、閾値が 10.2.0.0 に設定されている場合、クラスタにアップロードされる複数のランダムな RoboServer 間での複数のランダムなロボットの分配は次のようになります。

	10.2.0.0 RoboServer	10.3.0.0 RoboServer	10.5.0.0 RoboServer	10.7.0.0 RoboServer
9.7.0.0 ロボット	一致	-	-	-
10.1.0.0 ロボット	一致	-	-	-
10.3.0.0 ロボット	-	厳密なバージョン一致	-	-
10.4.0.0 ロボット	-	-	-	-
10.7.0.4 ロボット	-	-	-	-

製品バージョンが閾値を下回るすべてのロボットは、同等またはそれ以上のバージョンの RoboServer と正常に一致します。

10.3.0.0 のロボットは、製品バージョンが閾値を超えており、クラスタ内に同じバージョンの RoboServer があるため、厳密にバージョンが一致します。

10.4.0.0 のロボットには一致するバージョンがなく、製品バージョンが閾値を超えており、クラスタ内に同じバージョンの RoboServer がないため、実行されません。

10.7.0.4 のロボットには一致するバージョンがなく、製品バージョンが閾値を超えており、クラスタ内に同じバージョンの RoboServer がないため、実行されません。

デフォルトでは、閾値は現在の Management Console のバージョンに設定されます。

バックアップを使用して、Kofax RPA の新しいバージョンにアップグレードした場合、RoboServer の閾値のバージョンは自動的に更新されません。ロボットを確実に実行できるようにするには、**[クラスタ設定]** > **[一般]** > **[閾値のバージョン]** の順に移動し、手動で RoboServer の閾値のバージョンを指定してください。

KCU ライセンスを使用する古いロボットを新しい RoboServer で実行する場合 (たとえば、10.3.2.2 の RoboServer を Management Console 11.2 環境に追加する場合) は、新しいクラスタを作成し、古い RoboServer ノードをそのクラスタ内に配置します。「RoboServer」の「新しいクラスタの作成」を参照してください。

データベース

クラスタ設定の [データベース] タブで、データベースを追加または削除できます。

クラスタで定義したデータベースは、**[リポジトリ]** > **[データベース マッピング]** (データベース マッピングを参照) を介して、クラスタ RoboServer で実行されているロボットが使用できます。データベース タイプは、[設定] メニューで定義します。

Design Studio にシェア データベースを提供するクラスタが選択されていて、これらのデータベースを指すデータベース マッピングがリポジトリに存在する場合、クラスタで定義されたデータベースは Design Studio でも使用できます。

データベースのプロパティについての詳細は、データベース接続を参照してください。

特定のデータベースに接続するには、Kofax RPA にこのデータベース タイプの JDBC ドライバーが必要となります。このドライバーは、データベースのプロバイダーからダウンロードできます。たとえば、Microsoft SQL Server データベースで使用される `sqljdbc4.jar` は Microsoft の Web サイトからダウンロードできます。Kofax RPA で使用するドライバーを指定するには、**[設定]** > **[データベース ドライバー]** > **[ドライバー JAR ファイルのアップロード]** で Management Console にドライバーをアップロードします。

デフォルトでは、Management Console にドライバー JAR ファイルをアップロードするには、localhost からこの Management Console に管理者ユーザーとして接続している必要があります (このように接続していない場合、[ドライバー JAR ファイルのアップロード] ボタンは無効になります)。この設定は、**[RoboServer 設定]** > **[Management Console]** > **[JDBC ドライバー アップロード]** で変更できます。重要: Management Console を実行しているユーザーとして RoboServer 設定を開始します。RoboServer 設定を変更した後に、Management Console を再起動して変更を有効にします。

アクション

- [新しい開発用データベース] ボタンを使用すると、事前にインストールされている Kofax RPA 開発データベースへのデータベース接続が作成されます。
- [新しいデータベース] ボタンをクリックすると、新しいデータベース接続が作成されます。

新しいデータベース

1. クラスタ設定の [データベース] タブで、[新しいデータベース] を選択します。
2. 以下のデータベースの詳細をすべて入力します。
   - 名前
   - ホスト
   - ポート
   - スキーマ

     データベース プロバイダで「スキーマ」と「データベース」の概念が同等である場合 (同じように使用される場合)、このフィールドにスキーマの名前を指定します。それ以外の場合は、データベースの名前を指定します。この場合、デフォルトのスキーマが使用されます。
   - タイプ
   - ユーザー名
   - パスワード
   - 最大アクティブ接続数
   - 最大待機接続数
3. [**OK**] をクリックします。

プロキシ サーバー

RoboServer で使用するプロキシ サーバーのリストを設定することができます。1 つのプロキシ サーバーのみが定義されている場合は、そのサーバーが使用されます。プロキシ サーバーのリストが定義されている場合は、最初の接続に対してプロキシ サーバーがランダムに選択されます。

プロキシの詳細については プロキシ サービスの使用を、また、プロキシ サーバーのプロパティについては Design Studio、プロキシ サーバーを参照してください。

レガシー プロパティ ファイルからプロキシ サーバーをインポートするには、プロキシ サーバーのカテゴリを選択して、[ファイルからプロキシ サーバーをインポート] をクリックします。この操作により、インポートするファイルのコンテンツを貼り付けるためのウィンドウが開きます。ファイルには、プロキシ サーバートピックの下部で説明されているフォーマットを使用する必要があります。

1. クラスタ設定の [プロキシ サーバー] タブで、[新しいプロキシ] を選択します。
2. 以下のプロキシ サーバーの詳細をすべて入力します。
   - ホスト
   - ポート
   - ユーザー名
   - パスワード
   - 除外 (除外ホスト名)
3. [**OK**] をクリックします。

ロギング

クラスタ RoboServer のログ オプションを有効にするには、**[設定]** > **[一般]** > **[RoboServer ログ データベース]** に移動し、適切なオプションを選択します。

次の関連トピックも参照してください。

- 「クラスタ設定の構成」を参照してください。
- 「データベース」を参照してください。
- 「RoboServer ログ データベース」を参照してください。
- 「ログ ビュー」を参照してください。

データベースに記録

このオプションを使用して、データベース ロギングを有効にします。RoboServer は、Management Console 設定で定義されている RoboServer ログ データベースにログを記録します。

ロボットの入力をデータベースに記録

このオプションを使用して、ロボットの入力をデータベースに記録します。

「ロボットの入力をデータベースに記録」 オプションを有効にした後に、ログの [ログ ビュー] > [ロボット実行] > [入力値] 列で入力を表示する場合は、該当する各クラスタの入力ビューを有効にします。

- **[管理]** > **[RoboServer]** に移動します。
- クラスタの **[アクション]** 列で、コンテキスト メニューから ⋮ を選択し、**[クラスタ設定]** > **[ログ]** > **[ロボットの入力をデータベースに記録]** の順に選択します。

> [設定] で「[ロボットの入力をデータベースに記録]」オプションのみを有効にした場合、[ログ ビュー] の [入力値] 列は空白のままになります。

Log4J に記録

通常の `log4j2.properties` ファイルを使用してログの出力先を制御するには、このオプションを使用します。このファイルは、Kofax RPA アプリケーション データ フォルダの構成ディレクトリにあります。

例：`C:\Users\name.lastname\AppData\Local\Kofax RPA\11.5.0\Configuration`

『Kofax RPA インストール ガイド』の「アプリケーション データ フォルダ」および『Kofax RPA 管理者ガイド』の「Management Console 監査ログ」を参照してください。

デフォルトの `log4j2.properties` ファイルの場合は、すべてのロボット実行情報、ロボット メッセージ、およびサーバー メッセージがファイルにログが記録されます。

ログは、アプリケーション データ フォルダのログ フォルダに配置されます。より高度な設定では、Windows イベント ログ、ローテーション ファイル、syslog への格納などを設定できます。

[Log4J に記録] オプションを有効にすると、RoboServer は 3 つの異なるロガーを使用してログを記録します。

ロガー	説明
`logger.servermessagelog.name = kapow.servermessagelog` `logger.servermessagelog.level = {TRACE,INFO,DEBUG,ERROR...}`	特定のロボット実行に結び付けられていないサーバー ログに使用します。

ロガー	説明
`logger.robotrunlog.name = kapow.robotrunlog` `logger.robotrunlog.level = {TRACE,INFO,DEBUG,ERROR...}`	実行時間を含めて、ロボットの開始と停止に関する情報をログに記録するために使用します。
`logger.robotmessagelog.name = kapow.robotmessagelog` `logger.robotmessagelog.level = {TRACE,INFO,DEBUG,ERROR...}`	エラー処理に関する情報など、ロボット メッセージをログに記録するために使用します。ログに出力するようにプロファイリングを設定している場合は、プロファイルも含まれます。

ロボットの入力を **Log4j** に記録

`log4j2.properties` ファイルを使用してロボット入力をログに記録するには、このオプションを使用します。

電子メールに記録

ロボットがエラー メッセージを記録するようにするか、サーバーにエラー メッセージがあるときに電子メールを送信するには、このオプションを使用します。

プロパティ	説明
電子メールの送信元	電子メールで使用される送信元アドレス。
電子メールの送信先	電子メールを受け取るユーザー。複数のアドレスを追加するには、アドレスをカンマで区切ります。

プロファイリング

ロボットのプロファイリングを有効化すると、個々のステップおよびロボット全体の実行時間を確認することができます。これは、速度の遅いロボットがあり、そのパフォーマンスを向上させる必要がある場合に非常に役立ちます。

プロファイリングは、以下のプロパティを使用して設定します。

プロパティ	説明
出力タイプ	[概要] を選択した場合は、実行をまとめた、1 つのステートメントのみが、各ロボットの実行のプロファイリング ログに書き込まれます。 [詳細] を選択すると、ステップ実行時間が [閾値] プロパティで定義されているものを超える場合、ロボットで実行された各ステップのプロファイリング ログに詳細なステートメントが書き込みされます。 詳細ログには、ロボットのパフォーマンスを分析および改善するのに役立つ次の情報が含まれています。 • リモート デバイスの待機時間 (ミリ秒単位)。 • 実行されたステップの名前。ロボット のステップでは、ステートメントの前にロボット名が付きます。 • ステップ実行時間 (ミリ秒単位)。 • KCU ポイント コスト。 • 利用可能な KCU ポイントの待機時間。 • ロボットの実行時間 (ミリ秒単位)。 • ロボットのロード時間 (ミリ秒単位)。 • 合計待機時間の要約。
出力ターゲット	プロファイリング情報の送信先を制御します。
閾値	このしきい値は、プロファイリング情報に必ず含まれるようにするためのステップの最小の実行時間 (ミリ秒) を定義します。
URL のログ	これをチェックすると、ページの URL が、ページのロード待ち時間の前に表示されます。

ロボットの実行

このタブを使用して、RoboServer ロボット実行のプロパティを定義します。同時に実行できるロボットの数と、各 RoboServer でキューに格納することができるロボットの数を指定します。

CRE ライセンスを使用している場合、クラスタ内で同時に実行できるロボットの最大数は、ライセンスによって定義されます。

プロパティ	説明
最大同時ロボット数	それぞれの RoboServer で同時に実行できるロボットの最大数。 ロボットが CPU やメモリを大量に消費し、ロボットを実行するとメモリの閾値に達することが多い場合は、この最大数を下げます。 デフォルトは 5 です。

プロパティ	説明
最大キュー格納可能ロボット数	それぞれの RoboServer のキューに格納できるロボットの最大数。 ロボットを API コールを介して開始する場合は、それぞれの RoboServer でキューに格納できるロボットの数をこの値で決定します。 この設定は、クラスタを呼び出さずに RoboServer を直接呼び出すために、Java API の `SocketBasedObjectRQLProtocol` などの非推奨のメソッドでのみ使用します。廃止予定のメソッドの使用は推奨されません。 デフォルトは 10 です。

プロジェクト

このオプションは、次のロールを持つユーザーが利用できます: 管理者、プロジェクト管理者、KTA クライアント、VCS サービス ユーザー、および RoboServer。

Management Console のプロジェクトを設定および作成するには、[管理] メニューの [プロジェクト] オプションを使用します。

プロジェクトは、ロボット、タイプ、スニペット、リソース、スケジュール、およびその他の作業アセットを分割するための手段です。ロボットは、そのロボットが属するプロジェクトに含まれるタイプ、スニペット、およびリソースにのみアクセスできます。ロボットの名前、タイプ、およびその他のオブジェクトをプロジェクト内で区別する必要があります。

デフォルトでは、Management Console には「デフォルトのプロジェクト」という単一のプロジェクトが含まれています。

各プロジェクトに関する情報の表示方法を次のように変更します。

- 右側の メニュー アイコンを使用して、プロジェクトに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、プロジェクトごとに次のテーブル列が表示されます。

列	説明

名前	プロジェクトの名前。一意である必要があります。
	プロジェクト名は、誰にログファイルの表示を許可するかを決めるために、ログ テーブルにおいて外部キーとして使用できます。
	プロジェクトの名前を変更した場合は、既存のすべてのロボットを実行し、このプロジェクトに属するロボット メッセージを更新して、新しい名前を反映させる必要があります。新しい名前が反映されていない場合は、プロジェクトを基準にログをフィルタリングしてもログが表示されません。
	Management Console がログ データベースに接続されている場合は、プロジェクトの名前を変更すると、Management Console によりログ データベースのロボット実行とメッセージ エントリの名前が自動的に変更されます。ただし、Management Console が接続されていない場合、または更新中に接続が切断された場合、管理者は以下の SQL を手動で実行してログ テーブルを更新する必要があります。
	`UPDATE ROBOT_RUN SET projectName = '<newName>' where projectName = '<oldName>';`
	`UPDATE ROBOT_MESSAGE SET projectName = '<newName>' where projectName = '<oldName>';`
	`<oldName>` は以前のプロジェクト名で、`<newName>` は新しいプロジェクト名を意味します。

説明	プロジェクトの説明。
REST を認証	REST/SOAP リクエストの認証が有効かどうかを示します。
オプションの列	
インポート元	プロジェクトをインポートした、またはバックアップの一部として復元したユーザーのユーザー名。
インポート時間	プロジェクトがバックアップの一部としてインポートまたは復元された日時。
権限	プロジェクトに設定されたプロジェクト権限。
REST クラスタ	ロボットが REST サービスとして呼び出されたときに、このプロジェクトのロボットを実行するために使用されるクラスタ。

新しいプロジェクトの作成

1. Management Console にログインします。
2. [管理] メニューをクリックし、[プロジェクト] オプションを選択します。
3. 左上隅のプラス記号をクリックして新しいプロジェクトを作成します。
   「プロジェクトを追加」 ダイアログ ボックスが表示され、デフォルトの開始位置として [基本] タブが開きます。
4. [基本] ペインで、プロジェクトの名前を指定し、説明を入力します。
5. [権限] タブを選択します。
   a. プラス記号をクリックして権限を追加します。
   b. プロジェクト ロールとセキュリティ グループを入力します。
      プロジェクト ロールの説明については、ユーザーおよびグループを参照してください。

   Management Console をスタンドアロンの Web コンテナにデプロイする場合は、グループ メンバーシップ (LDAP グループなど) に基づいてユーザーのプロジェクト権限を設定することがで

きます。詳細については、『Kofax RPA 管理者ガイド』の「プロジェクト権限」にある「Tomcat Management Console」セクションを参照してください。

6. [サービス] タブを選択してサービスを構成します。
   a. [サービス クラスタ] については、リストから選択します。
   b. [プロジェクトでサービス クラスタのみを使用する] については、チェック ボックスをオンまたはオフにします。

      サービス クラスタは、現在のプロジェクトで REST サービスを実行します。REST サービスは常に、選択されたサービス クラスタを使用します。
      - このチェック ボックスをオンにした場合、プロジェクトの [サービス クラスタ] を選択する必要があります。
      - Management Console では他のすべてのクラスタが非表示になり、コードを生成する場合、および口ボット メニューから口ボットを実行する場合に、すべてのスケジュールに対して選択されたサービス クラスタが使用されます。
   c. [**REST/SOAP** リクエストの認証] については、チェック ボックスをオンまたはオフにします (有効または無効にします)。

      デフォルトでは、REST/SOAP サービスは基本認証を使用して保護されます。
      - XMLHttpRequest を使用してブラウザからサービスを直接呼び出す場合は、認証を無効にます。無効にしない場合、JavaScript ソース ファイルの口グイン資格情報が公開されます。
      - Java、Ruby、C# などのプログラミング言語から REST/SOAP サービスを呼び出すときに、資格情報を安全に保存できる場合は認証を有効にしてサービスを保護します。
   d. [アクセス-制御-許可オリジン] については、別のドメインのリソースを処理するためのクライアント アクセスのヘッダーを入力します。

      REST/SOAP サービスは呼び出し元の Web ページが置かれている Web サーバーと別の場所にある場合、ブラウザから呼び出す際には制限が発生します。
      - 別のドメインから REST/SOAP サービスを呼び出す場合 (CORS、Cross-Origin Resource Sharing とも呼ばれる)、アクセス制御ヘッダーを含める必要があります。
      - クロス ドメイン方式で REST/SOAP サービスを呼び出す場合は、リクエストを生成したページがロードされたドメインを指定する必要があります。

        例**:** http://example.com のページに、http://kofax.com に存在するサービスへの要求を生成する JavaScript を含んだページがある場合、http://kofax.com からのサービス応答にヘッダー「Access-Control-Allow-Origin: http://example.com」が含まれている必要があります。もしくは、このヘッダーがない場合は、ブラウザの組み込みセキュリティ メカニズムにる、CORS (Cross Origin Response) の処理が妨げられます。
      - ワイルドカードとして * を使用することも可能です。これは、すべてのドメインがクロスドメイン方式で REST/SOAP サービスを呼び出すことができることを意味します。
7. [リポジトリ] タブを選択して、リポジトリ アクセスを設定し、オブジェクトを同期します。
   a. [設定を有効化] をクリックして、リポジトリを設定するためのフィールドをアクティブ化します。

      このオプションにより、GitHub や GitLab などのバージョン管理システムで作業オブジェクトを制御できます。このオプションの詳しい使用例については、『Kofax RPA ベスト プラクティス ガイド』を参照してください。
   b. [**URL**] フィールドに、リポジトリへのパスを入力します。
      - リポジトリに直接リンクするには、リポジトリへのパスを入力します。

例:`https://gitrepos/example.git/`

- ユーザー名とパスワードを含む URL で Synchronizer を使用するには、ユーザー名とパスワードをリポジトリへのパスと共に入力します。

  例:`https://username:password@gitlab.com/project_name/repo_name.git`

  この URL エントリは、以前はオンライン リポジトリと同期する唯一の方法であった SSH キー ペア認証方法をバイパスします。詳細については、『Kofax RPA ベスト プラクティス ガイド』の「同期の開始」を参照してください。

c. [ブランチ] フィールドに、使用するブランチの名前を入力します。

例: 2 つの Management Console を使用したセットアップがあり、片方は本番用、もう片方は開発用である場合は、専用の開発ブランチを設定することができます。

d. リポジトリをオブジェクト変更の唯一のソースにするには、[読み取り専用] を選択します。

本番用 Management Console 内の同期されたプロジェクトに属するオブジェクトが変更されないようにするために、このオプションを選択することをお勧めします。

読み取り専用モードでは、Management Console に含まれるすべてのオブジェクトがリポジトリから取得されます。同期を開始する前に、同期するオブジェクトが Management Console で空であることを確認してください。空でない場合、エラー メッセージが表示されます。

オブジェクトの内容を削除するには、同期するオブジェクト リストから必要なオブジェクトを選択し、[選択したオブジェクトを削除] をクリックします。重要なデータを失わないように、オブジェクトを削除するときは注意してください。

e. [同期するオブジェクト] については、同期に含めるオブジェクトを選択します。

8. [**OK**] をクリックして変更を保存します。

プロジェクトを変更する

1. Management Console にログインします。
2. [管理] メニューをクリックし、[プロジェクト] オプションを選択します。
3. プロジェクトのリストから、変更するプロジェクトを選択します。

   プロジェクトを削除すると、プロジェクトに関連付けられたロボット、タイプ、スニペット、リソース、およびスケジュールもすべて削除されます。

4. プロジェクトを削除するには、プロジェクトを選択して、アイコンをクリックします。
5. プロジェクトを編集するには、コンテキスト メニューから [編集] を選択します。
   「プロジェクトの編集」 ダイアログ ボックスが表示されます。このダイアログ ボックスには、選択したプロジェクトを編集するためのタブとフィールドが含まれています。
6. 必要に応じてプロジェクトを編集します。
7. [**OK**] をクリックして変更を保存します。

高可用性ノード

このセクションは、次のロールを持つユーザーが利用できます: 管理者およびプロジェクト管理者。

このセクションには、設定されているすべての高可用性ノードが一覧表示されます。高可用性モードを有効にして、複数の Management Console がクラスタとして連携するように設定する方法については、『Kofax RPA 管理者ガイド』の「高可用性」を参照してください。

高可用性ノードの情報の表示方法は次のように変更できます。

- 表示するテーブル列を選択するには、右側の メニュー アイコンを使用します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。

デフォルトでは、設定したすべてのノードに次のテーブル列が表示されます。

列	説明
ノード ID	設定済みの Management Console ノードの ID。
インターフェイス	ポートを含む Management Console ノードの IP アドレス。
ステータス	ノードのステータス: 実行中、シャットダウン中、またはシャットダウン。
接続先	現在このノードに接続しているかどうかに応じて、「Yes」または「No」と表示されます。

デバイス

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、および RoboServer。

このセクションには、Management Console に登録されている利用可能なオートメーション デバイスが表示されます。詳細については、オートメーション デバイス マッピングの要件を参照してください。

Management Console にオートメーション デバイスを登録するには、Desktop Automation サービスで Management Console の詳細を指定します。[シングル ユーザー] オプションは空のままにします。『Kofax RPA Desktop Automation サービス ガイド』を参照してください。

デバイス/クラスタの情報の表示方法は、次のように変更できます。

- 右側の メニュー アイコンを使用して、デバイス/クラスタに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。

デフォルトで、各クラスタ/デバイスに対して、次の情報が表示されます。

列	説明
クラスタ	クラスタの名前。
デバイス	オートメーション デバイスの IP アドレスまたは名前、およびデバイスへの接続に使用するポート。

列	説明
ステータス	クラスタの場合 現在のクラスタの状態。詳細については、クラスタの状態の変更を参照してください。 デバイスの場合 デバイスの現在のステータス。ステータスは、次のとおりです。 • 利用可能：Desktop Automation サービスが実行されており、デバイスへの接続はありません。 • 使用中: Desktop Automation サービスが実行されており、Design Studio または RoboServer がデバイスに接続しています。オートメーション デバイスでは、1 つの接続のみを確立できることに注意してください。 • オフライン：リモート デバイスがオフになっているか、Desktop Automation サービスが起動していません。 • 予約済み: Management Console がデバイスをロボットに渡し、ロボットがデバイスに接続すると、デバイスは「予約済み」としてマークされます。
ラベル	デバイス マッピングのラベル。ラベルを使用して、ロボットで自動化するデバイスをフィルタリングすることができます。
ユーザー名	Desktop Automation サービスを実行しているユーザーの名前。
ドメイン	Desktop Automation サービスを実行しているユーザーのドメイン名。
バージョン	ロボットを編集する際に最後に使用した Kofax RPA バージョン。
実行 ID	特定のオートメーション デバイスを使用したロボット実行 ID。
ロボット名	実行中のロボットの名前。

ユーザーおよびグループ

このセクションは、次のロールを持つユーザーが利用できます: 管理者およびプロジェクト管理者。

管理者は、Management Console およびプロジェクトへのアクセス権を付与されたユーザーとグループを管理します。このセキュリティ モデルはロールベースです。ユーザーを追加した後に、1 つ以上のプロジェクトのロールに関連付けられている 1 つ以上のグループにそのユーザーを追加します。

このセクションには 2 つのタブがあります。

- [ユーザー] タブでは、ユーザーを作成、編集、削除できます。また、ユーザーのパスワードをリセットできます。
- [グループ] タブでは、グループを作成、削除、編集できます。

Kofax RPA のユーザー名とグループ名は、Microsoft Windows のログオン名ルールに従う必要があります。名前には、次の文字は使用できません。

" / \ [ ] : ; | = , + * ? < >

ビューをカスタマイズする

各タブに関する情報の表示方法をカスタマイズします。

- [フィルタ] テキスト フィールドにフィルタを適用して、テーブル内のリストをフィルタリングします。「フィルタリング」を参照してください。
- メニュー アイコンを使用して、表示されるテーブル列を選択します。
- 更新アイコンをクリックして、表示された情報を更新します。
- リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、ナビゲーション メニューを使用してページ間を移動します。

ユーザー

デフォルトでは、ユーザーごとに次のテーブル列が表示されます。

列	説明
ユーザー名	ユーザーの名前。
ユーザーのオリジン	作成方法に応じたユーザーのオリジン。 • `unknown`: ユーザーはバックアップの復元後に作成されました。 外部 ID プロバイダー (SAML、LDAP、SiteMinder など) を使用していない場合、ユーザーを選択して [内部オリジンを設定] をクリックすることで、`unknown` オリジンを `internal` オリジンに変更できます。 • `internal`: ユーザーは [ユーザーおよびグループ] ページで手動で作成されました。 • `saml`: ユーザーは SAML 経由でのログイン後に作成されました。 • `site minder`: ユーザーは SiteMinder 経由でのログイン後に作成されました。 • `ldap#{ldapDirectoryIdentifier}`: ユーザーは LDAP 経由でのログイン後に作成されました。 外部 ID プロバイダーおよびユーザー認証の詳細については、『Kofax RPA 管理者ガイド』の「Tomcat Management Console」>「高度な構成」を参照してください。 SAML、LDAP、または SiteMinder などの外部 ID プロバイダを使用する場合、選択したユーザーの一部のフィールドを変更できません。
フル ネーム	ユーザーのフル ネーム。
電子メール	ユーザーの電子メール アドレス
ログイン カウント	このユーザーが行ったセッションの数。
前回のログイン	ユーザーが最後にログインした日時。
最後の IP アドレス	ユーザーがログインした最後の IP アドレス。
グループ	ユーザーが所属するグループ。

新しいユーザーを作成

ユーザーをグループに追加する前に、グループを作成します。ユーザーは、少なくとも 1 つのプロジェクト内で閲覧者ロールが付与されたグループに追加されるまでログインできません。

1. [ユーザー] タブで、+ 記号をクリックします。
   「新しいユーザーを作成」 ダイアログボックスが表示されます。
2. ユーザーのユーザー名、パスワード、フル ネーム、および電子メールを指定します。
3. ユーザーが所属する 1 つまたは複数のグループを選択します。
4. [**OK**] をクリックします。
   「Dev」ユーザーがテーブルに表示されます。

⋮ コンテキスト メニューからユーザーを編集します。

ユーザー パスワードをリセットする

1. [ユーザー] タブでユーザーを選択し、 をクリックします。
   パスワードをリセットするダイアログ ボックスが表示されます。
2. 新しいパスワードを入力し、確認のためにもう一度入力して、**[OK]** をクリックします。
   電子メールを送信して、ユーザーがパスワードの変更に関する通知を受け取るように選択することができます。「送信元アドレス」は、通知が送信されるように事前設定する必要があります。

グループ

デフォルトでは、グループごとに次の情報が表示されます。

列	説明
グループ名	グループの名前。
説明	グループの説明。
ユーザー数	グループに含まれるユーザーの数。
プロジェクトで使用	このグループを使用するプロジェクト。

新しいグループを作成

1. [グループ] タブで、+ 記号をクリックします。
   「新しいグループを作成」 ダイアログ ボックスが表示されます。
2. グループ名と説明を指定します。
3. このグループに含めるユーザーを選択します。
4. [**OK**] をクリックします。
   「Developers」グループがテーブルに表示されます。

⋮ コンテキスト メニューからグループを編集します。

組み込みロール

Management Console には、ユーザーに割り当てることができる組み込み済みロールが用意されています。ロールはユーザーまたはサービスにマッピングされます。ユーザー権限は、ユーザーが所属するセキュリティ グループに割り当てられたロールに基づいて計算されます。組み込み済みロールを変更したり、その他のロールを追加したりすることができます。

ユーザーは、自身に割り当てられていない権限を含むロールを割り当てることはできません。たとえば、プロジェクト管理者は Kapplet 管理者ロール、Kapplet ユーザー ロール、および Process Discovery クライアント ロールを割り当てることはできません

サービス ロールは API アプリケーションでのみ使用することを目的としているため、ブラウザの Management Console への対話型ログインには使用しないでください。

- プロジェクト管理者: 1 つまたは複数のプロジェクトを管理し、これらのプロジェクトのグループにロールを割り当てる権限を持ちます。また、RoboServer およびクラスタ設定を変更せずに表示する権限を持ちます。プロジェクト管理者は、RPA 管理者グループのメンバーではありません (詳細については、このセクションの後半を参照してください)。
- 開発者: リポジトリ内のすべてのリソース タイプをアップロード、ダウンロード、表示する権限があります。このロールにより、スケジュールの作成、編集、削除、ロボットの実行、および実行ログとクラスタの表示を行う権限が付与されます。
- 閲覧者: [スケジュール]、[リポジトリ]、[データ ビュー]、[ログ ビュー]、および一部の [設定] を表示できます。このロールにより、[管理者] セクションの下で制限付きアクセスが付与されますが、ロボットを変更したり実行したりする権限は付与されません。
- **API** (サービス ロール): リポジトリ API を使用してリポジトリの読み取りおよびリポジトリへの書き込みを行う権限を付与します。このロールでは、REST を使用したロボットの実行は許可されませんが、RQL を使用したロボットの実行は許可されます。
- サービス認証 **API** (サービス ロール): リポジトリ API を使用して、リポジトリの読み取りおよびリポジトリへの書き込みを行います。ユーザーは OAuth 認証方法を使用してログインします。
- **RoboServer** (サービス ロール): リポジトリからの読み取りのみができます。このロールは、クラスタにアクセスする場合、リポジトリ アイテムを取得する場合、およびパスワード ストアからパスワードを要求する場合に RoboServer で使用されます。
- **Kapplet** 管理者: Kapplets から Management Console のプロジェクトへの読み取り/書き込みアクセス権を付与します。Kapplets では、このロールを持つユーザーは Kapplet と Kapplet テンプレートを管理できます。Kapplets では、このロールを持つユーザーは、Kapplet を管理し、これらのテンプレートに必要なロボットを含むプロジェクトの Kapplet テンプレートを作成および管理できます。

  このロールを持つユーザーは、他の権限がない場合、Management Console にアクセスできません。

  詳細については、Kapplet ユーザー管理 と ユーザーとユーザー グループ を参照してください。
- **Kapplet** ユーザー: Kapplets から Management Console のプロジェクトへの読み取り専用アクセス権を付与します。Kapplets では、このロールを持つユーザーは、アクセス権のあるプロジェクトに属するロボットの Kapplets のみを表示および実行できます。

  このロールを持つユーザーは、他の権限がない場合、Management Console にアクセスできません。

  詳細については、Kapplet ユーザー管理 と ユーザーとユーザー グループ を参照してください。
- **Kapplet** サービス ユーザー (サービス ロール): リポジトリからの読み取りのみができます。これは特定のプロジェクトで利用できるロボット、タイプ、スニペット、リソースに関する情報の取得にのみ使用されるロールなので、Kapplets と Management Console の間の通信のみを目的に使用します。このロールはすべての Management Console プロジェクトに自動的に適用されます。

- パスワード ストア クライアント (サービス ロール): このアドオン ロールにより、Management Console のパスワード ストアへのアクセス権を付与します。このロールは、開発者ロールと同様に、他のロールの上位に提供されます。
- **DAS** クライアント ユーザー (サービス ロール): このロールを持つユーザーはリモートの Desktop Automation サービス (DAS) クライアント用に作成され、DAS API へのアクセス権のみを持ちます。DAS クライアント ユーザーには DAS を Management Console に提示し、DAS 構成を取得する権限があります。
- **VCS** サービス ユーザー (サービス ロール): シンクロナイザーに特別な権限セットを付与します。このロールにより、リソースを追加、編集、削除する権限を付与します。他のユーザーの代理で展開を実行して VCS で「deployer」機能を使用できるのはこのロールのみです。
- **Process Discovery** クライアント (サービス ロール): このロールを使用すると、Process Discovery コンポーネントを Management Console と連携させることができます。
- **KTA** クライアント (サービス ロール): このロールを使用すると、KTA コンポーネントを Management Console と連携させることができます。

組み込み **admin** ユーザー

admin は、すべてにアクセスできるスーパーユーザーです。admin は RPA 管理者グループのメンバーではありません。また、どのグループのメンバーにも属しません。 このユーザーはデフォルトの admin ユーザーのパスワードを使用できます。このユーザーは、admin のユーザー名とパスワードを変更できます。

LDAP 統合セットアップでは、管理者グループは LDAP 設定の一部として定義されます。admin はログインを行い、開発者、プロジェクト管理者、RoboServer などのロールにマッピングする LDAP グループを定義できます。

内部ユーザー設定では、admin ユーザーは初期開始時に作成します。また、このユーザーはログインおよび管理者や開発者などのユーザーの作成が可能です。

admin は、初期ユーザーであることに加えて、次のような特別な権限を持ちます。

- [RoboServer] セクションで、admin は RoboServer ノードをクリックし、対応する RoboServer からスタック トレースをリクエストすることができます。
- admin のみがインポート バックアップを作成できます。
- パスワード ストアで、admin はパスワードを別のプロジェクトに移動できます。

admin のパスワードをリセットする

admin のデフォルトの名前とパスワードは次のとおりです。

- ユーザー名: `admin`
- パスワード: `admin`

管理者名とパスワードを変更するには、次の手順を実行します。

1. **[ユーザー]** タブでユーザーを選択し、 をクリックします。
   パスワードをリセットするダイアログ ボックスが表示されます。
2. 新しいパスワードを入力し、確認のためにもう一度入力して、**[OK]** をクリックします。
   ユーザーがパスワード変更に関する通知を受信できるように、電子メールの送信を選択できます。「送信元アドレス」は、通知が送信されるように事前設定する必要があります。

組み込みグループ

RPA 管理者グループに属するユーザーは、特別な admin ユーザー権限を除く、すべてのプロジェクトに対するすべての権限を持ちます。RPA 管理者ユーザーは、任意のプロジェクトに対して新しい管理者とユーザーを作成します。ユーザーを管理者にするには、そのユーザーをこのグループに追加します。

- RPA 管理者グループは、内部ユーザー管理が有効な場合に表示され、デフォルトでは空の状態です。
- 10.7 より前のバージョンで作成されたバックアップを復元する場合、管理者ロールを持つユーザーは RPA 管理者グループのメンバーになります。

ユーザー管理の原則

Management Console は、任意のライセンスを持つ RoboServer に組み込まれ、スタンドアロンの Tomcat サーバー上で実行されます (エンタープライズ ライセンスが必要です)。Tomcat の Management Console の詳細は、『Kofax RPA 管理者ガイド』の「Tomcat 管理コンソール」を参照してください。

Management Console を組み込みモードで実行すると、ユーザー管理がデフォルトで有効になり、他のコンピューターから Management Console への不正アクセスを防ぐセキュリティが提供されます。

ライセンスと Management Console の実行方法に応じて、次のようにユーザー アクセスを管理します。

- 内部ユーザー管理: 組み込みモードとスタンドアロン モードで利用可能。
- 外部ユーザー管理 (LDAP、SAML、または CA シングル サインオン): エンタープライズ ライセンスのスタンドアロン モードでのみ利用可能。

Tomcat サーバー上でエンタープライズ バージョンを実行すると、Management Console は常にマルチユーザー モードになります。Management Console (埋め込みモードなど) でユーザーを管理するか、企業の LDAP サーバーからユーザー クレデンシャルを取得するかを選択できます。認証方法は「ユーザーのオリジン」列に表示されます。

ログイン試行回数のチェック

デフォルトでは、ユーザーが行ったログイン試行回数と次の試行までの待ち時間のチェックが無効化されています。

1. この機能を有効にするには、`authentication.xml` ファイル内のコードを編集します。

   このファイルは次の場所にあります。`<Tomcat installation folder>\WebApps\Management Console\WEB-INF\spring` にある authentication.xml ファイル内の次のセクションを編集します。以下はコード サンプルです。

```
   <bean id="loginAttemptService"

 class="com.kapowtech.scheduler.server.spring.security.LoginAttemptService" lazy-init="true">
       <constructor-arg type="boolean" value="false"/>
       <constructor-arg type="int" value="3"/>
       <constructor-arg type="int" value="10"/>
   </bean>
```

2. 最初の値を **true** に設定します。
3. 2 番目と 3 番目の値を必要に応じて指定します。

2 番目と 3 番目の値は、ログイン試行回数 (この例では 3) と次の試行までの待ち時間 (この例では 10) です。

サービス認証

このセクションは、管理者が利用できます。

「サービス認証」セクションでは、次のクライアントを Management Console で認証し、追加のセキュリティを提供するために使用される共有シークレットを指定します。

- Kapplets
- RoboServer
- Desktop Automation サービス (DAS)
- Synchronizer
- Robot File System (RFS)
- Document Transformation Service (DTS)

アクション
クライアントの ⋮ コンテキスト メニューをクリックすると、次のアクションが一覧表示されます。

- 共有シークレットを表示: 特定のクライアントの共有シークレットとその生成日を表示します。
- 共有シークレットを変更: 新しい共有シークレットを生成します。
- トークンを取り消す: 選択したクライアントの共有シークレットを取り消します。

Management Console は、初回起動時に共有シークレットを自動的に生成します。生成されたシークレットは Management Console データベースに保存され、**[共有シークレットを変更]** ボタンをクリックして新しいシークレットを明示的に生成するまで、変更されることはありません。共有シークレットが Management Console で変更された場合は、新しいシークレットをコピーしてクライアントに貼り付けます。

共有シークレットを取り消すには、次のアクションを実行します。

1. 必要なクライアントのコンテキスト メニューを開き、**[共有シークレットを変更]** を選択します。
2. **[トークンを取り消す]** を選択します。
3. クライアントの共有シークレットを更新し、必要に応じてクライアントを再起動します。

バックアップ

このセクションは、次のロールを持つユーザーが利用できます: 管理者およびプロジェクト管理者。

[バックアップ] セクションを使用して、Management Console の設定のすべてまたは一部をファイルにコピーすることにより、データをバックアップまたはエクスポートします。必要であれば、ファイルを使用して復元またはインポート操作を行うことができます。

復元できるのは、Kofax RPA バージョン 9.7.10 またはそれ以降で作成されたバックアップです。以前のバージョンのデータの切り替えのヘルプについては Kofax RPA テクニカル サポートまでご連絡ください。

バックアップの作成/復元とプロジェクトのエクスポート/インポートにおける違いを理解することが重要です。

バックアップの作成と復元

バックアップには、リポジトリ内のすべてのスケジュールとすべてのプロジェクトを含む、すべての Management Console の設定が含まれており、全体の復元のみを実行できます。バックアップは、データ損失後や、Kofax RPA の以降のバージョンにアップグレードする際のシステムの復元に使用することができます。アップグレード後のシステムの復元に使用する場合には、Management Console の以前のインスタンスのバックアップを作成し、新しいインスタンスへと復元します。

ユーザー管理がオフになっている Management Console バックアップを復元した後、デフォルトのスーパーユーザー資格情報を使用して Management Console にログインします (ユーザー: `admin`、パスワード: `admin`)。ユーザー管理を有効にして Management Console からバックアップを復元すると、デフォルトの `admin` ユーザーはバックアップのスーパーユーザーに置き換えられます。復元した Management Console で指定された資格情報を使用します。

プロジェクトのエクスポートとインポート

[プロジェクトのエクスポート] 機能を使用して、単一のプロジェクトに関する情報を含むファイルを作成します。これは、プロジェクト内のスケジュール、ロボット、タイプ、リソース、OAuth、およびロボット ファイル システムで構成されます。こうしたファイルは、スケジュールやロボットなどを Kofax RPA システム間でコピーするために使用できます。たとえば、テスト環境から本番環境に移行する場合などが挙げられます。ターゲット システム上の既存のプロジェクトにインポートすることも可能です。この場合、ファイルのアイテムはプロジェクトにマージされ、名前が一致する場合は既存のアイテムが上書きされます。また、一部のアイテムのみを含める選択的なインポートを行うことも可能です。

バックアップの作成

1. バックアップを作成するには、**[管理]** > **[バックアップ]** で、[バックアップの作成] をクリックします。
2. [作成] をクリックします。
3. バックアップの準備が整うと、コンピュータにダウンロードされます。

プロジェクトのエクスポート

1. プロジェクトをエクスポートするには、**[管理]** > **[バックアップ]** で、[プロジェクトのエクスポート] をクリックします。
2. エクスポートするプロジェクトを選択します。
3. [エクスポート] をクリックします。
4. プロジェクトのエクスポートの準備が整うと、コンピュータにダウンロードされます。

プロジェクトの名前別のエクスポート

また、プロジェクトを名前別にエクスポートするには、次の URL を使用します。たとえば、API を使用してプロジェクトをバックアップする必要がある場合は、このエクスポート方法が便利です。

```
http://user:password@localhost:8080/ManagementConsole/secure/BackupProjectByName?projectName=MyProject
```

この URL の次の部分を実際のデータで置き換えます:

- ユーザー:パスワード: Management Console にアクセスするためのユーザー クレデンシャルです。
- **localhost:8080/ManagementConsole**: Management Console の URL です。
- **MyProject**: エクスポートするプロジェクトの名前です。

バックアップの復元

バージョン 9.5.0 以降では、クリーンアップの閾値が記録数から日数に変更されているため、記録数が 50000 を超える以前のバックアップ (以前のデフォルト) の場合、日数は 10 に設定されます。記録数が少なければ、日数も少なくなります。

1. バックアップを復元するには、**[管理]** > **[バックアップ]** で、[バックアップの復元] をクリックします。
2. ペーパー クリップ記号をクリックして、復元するバックアップ ファイルを選択し、コンピュータ上のファイルを選択してから、[**Open**] をクリックします。
3. [結合] または [リセット] を選択します。
   - バックアップ設定を現在のバージョンの Management Console に結合する場合、[結合] を選択します。

     バックアップに不足している設定、つまりバックアップの作成時に以前のバージョンに存在しなかった設定では、カスタム値が保持されます。
   - バックアップの復元前にすべての設定をリセットする場合、[リセット] を選択します。

     バックアップに不足している設定、つまりバックアップの作成時に以前のバージョンに存在しなかった設定は、デフォルト値にリセットされます。

     [結合] または [リセット] モードを使用してバックアップを復元すると、以前に作成したすべてのクラスタが削除されます。クラスタを保持するには、[バックアップの復元] ウィンドウで [既存のクラスタを保持] チェックボックスをオンにします。既存のクラスタの名前と、バックアップのクラスタの名前を異なるものにすることが重要です。
4. [復元] をクリックします。

   バックアップに含まれるアイテムの名前が命名規則に準拠していない場合、バックアップを復元すると、これらのアイテムの名前が自動的に変更されます。
5. インストールが完了した後に、[閉じる] をクリックします。

プロジェクトのインポート

1. プロジェクトをインポートするには、**[管理]** > **[バックアップ]** で、[プロジェクトのインポート] をクリックします。
2. インポートするアイテムを選択します。

   利用できるオプションには、以下が含まれます。
   - スケジュールのインポート
   - ロボット、タイプ、スニペットのインポート
   - リソースのインポート
   - OAuth のインポート
   - 権限のインポート

- ロボット ファイル システムのインポート
- トリガー マッピングのインポート
- メール トリガーのインポート

3. ペーパー クリップ記号をクリックして、インポートするプロジェクト ファイルを選択し、コンピュータ上のファイルを選択してから、[**Open**] をクリックします。
   - 同じ名前のプロジェクトがすでに存在している場合に、新しくインポートしたプロジェクトで置き換えるには、[存在する場合はプロジェクトを削除] を選択します。このオプションを選択しない場合、変更は既存のプロジェクトに結合されます。
4. [インポート] をクリックします。
   プロジェクトに含まれるアイテムの名前が命名規則に準拠していない場合、プロジェクトをインポートすると、これらのアイテムの名前が自動的に変更されます。
5. インストールが完了した後に、[閉じる] をクリックします。

ライセンス

このセクションは、次のロールを持つユーザーが利用できます: 管理者およびプロジェクト管理者。

Kofax RPA Management Console は、ライセンスが設定されている場合にのみ起動します。

[Management Console] > **[管理者]** > **[ライセンス]** セクションでは、次のタブを使用してライセンスを表示および管理します。

- ライセンスの詳細
- Design Studio シート
- ライセンス キー
- ライセンス サーバー

ライセンスの詳細

[ライセンスの詳細] タブを使用して、Kofax RPA ライセンスに関連するすべての詳細を表示します。

この情報は、ライセンスと設定の機能のみを表示するようにカスタマイズされています。

たとえば、Analytics のライセンスを所有している場合は、アイテム 「Kofax RPA Analytics: Yes」 と表に表示されます。

「ライセンスの詳細」は自動的に更新されませんが、ユーザーがライセンス表示に切り替えたとき、またはライセンス方式が変更されたときに再計算されます (Management Console から要求されます)。

「ライセンスの詳細」テーブルには、ライセンスのタイプと設定に基づいて、次のような読み取り専用の詳細が一覧で表示されます。

- **[機能]** には、ライセンスで利用できる機能が一覧で表示されます。
- **[使用中]** には、この Management Console で使用される機能とユニットが表示されます。
  たとえば、この合計はユーザー管理に登録されているユーザーの数を示します。CRE などの Management Console によって配布されるユニットにより、設定された数の本番ユニットと非本番ユニットが予約されます。
  ライセンスがライセンス サーバーを使用して設定されている場合、「使用中」の合計には予備ライセンスは含まれません。予約ライセンスは、使用中の場合にのみ表示されます。

- **[使用可能 (未使用)]** には、ライセンスに残っているライセンス ユニットが表示されます (ユニットの総数ではありません)。

  ユニットの総数は、他のテナントが使用するライセンス ユニットに依存するため、直接計算できない場合があります。ライセンス キーを使用して設定されている場合、未使用のユニットの数が表示されます。これは、ライセンス キーで設定されたユニットの数から使用中のユニットの数を差し引いたものです。
- **[有効期限]** には、該当する場合、ライセンス機能の有効期限が表示されます。

  個々の機能の日付は異なる場合があります。ライセンス キーを使用してライセンスを取得した場合、すべての機能に同じ有効期限が表示されます。

  Kofax TotalAgility 統合ライセンス サーバーで設定されたライセンスでは、Kofax RPA の「ライセンスの詳細」にシステム有効期限は表示されません。『Kofax TotalAgility Designer のヘルプ』の「統合」トピックを参照してください。

ライセンスの更新

Kofax RPA は、次の 2 つのライセンス タイプをサポートします。

- ステーション ライセンスでは、最大ユーザー数や CRE など、限られた数のユニットが提供されます。ユニットが使用されなくなった場合は、リセットして使用可能なユニットの一部にすることができます。
- システム ライセンスは無制限のライセンスを提供し、機能がいつ利用可能になるかを示します。

ライセンスのタイプと設定 (ライセンス キーまたはサーバー) に基づいて、Management Console は定期的に使用状況を更新し、ライセンスの可用性を確認します。

- 起動時に、Management Console はすべてのライセンス タイプを呼び出して、可用性と現在の使用状況を確認します。
- ステーション ライセンスの場合、Management Console はユーザー数をカウントし、ステーション API を使用して報告を行います。運用中は、次のように定期的に更新および可用性を確認します。
  - 使用状況を 5 ~ 10 分ごとに更新します (間隔は設定可能)。

    これは、ユーザー数などの限定されたライセンス単位を考慮します。無制限のリソースが設定されている場合、ユニット数はサーバーに報告されません。
  - 新しいユニットが要求されると、ステーション ライセンスのユニット消費量が更新されます。

    たとえば、新しいユーザーを作成すると、Management Console は、可用性を確認するための新しいユニットの要求によって、使用中のユニットの数を更新します。ユニットを割り当てることができない場合、新しいユーザーを作成アクションは拒否されます。
- ライセンス キー設定の場合、次のようなシステム ライセンスの変更を Management Console に反映するには再起動が必要です。
  - 高可用性などのシステム ライセンス機能の削除。
  - システム ライセンスが変更された場合。
- システム ライセンス サーバー設定の場合、機能の可用性とライセンスの変更が 10 分ごとにチェックされます。ライセンスは、Kofax TotalAgility 環境内のサーバーによって管理されます。
- シャットダウン時には、すべてのステーション ライセンスが 「0」 の値で呼び出され、リソースが解放されます。

ライセンスの予約

Kofax RPA ライセンスがライセンス サーバーで設定されている場合、Kofax TotalAgility「ライセンス API キー」は、使用を開始したときにのみ予約ライセンスを報告します。

たとえば、5 人のユーザーと 3 つの予備ライセンスがある場合、5 人以上を使用するまで、「ライセンスの詳細」には「5」ユーザーと表示されます。6 番目のライセンスを使用すると、予約を利用していることが表示され、ライセンスの合計は「6」になります。Kofax TotalAgility は、割り当てられた後の予約ライセンスのみを報告します。

Design Studio シート

「Design Studio シート」タブを使用して、使用中のシート (ライセンス ユニット) に関する情報を表示します。

この表には、この Management Console からのライセンスを持つ Design Studio クライアントの一覧と、ユーザーがログインした最後の IP アドレスが表示されます。

- 表示するテーブル列を選択するには、 メニュー アイコンを使用します。
- 更新アイコンをクリックして、表示された情報を更新します。

ライセンス キー

「ライセンス キー」タブを使用して、ライセンス キーで設定された Kofax RPA のライセンスに関する情報を入力および表示します。

キーを使用して Kofax RPA にライセンスが付与されている場合、「ライセンス キー」タブのラベルの横にアイコン が表示されます。

Management Console には、本番キーおよび非本番キーという 2 つのライセンス キーを入力できます。

本番環境が開発/テスト環境から完全に分離されている場合は、本番ライセンス キーを含む Management Console と、非本番ライセンス キーを含む Management Console という 2 つの Management Console をインストールする必要があります。

本番環境と非本番環境が混在する環境では、単一の Management Console に両方のキーが含まれることがあります。

製品をインストールし、ライセンス キーを入力すると、「ライセンス キー」タブに次の情報が表示されます。

- 管理者の名前と電子メール アドレス。
  これら 2 つのフィールドは更新することができます。残りのフィールドは読み取り専用です。
- 会社名
- 本番ライセンス キー
- 非本番ライセンス キー

ライセンス サーバー

Kofax RPA が Kofax TotalAgility ライセンス サーバーで統合ライセンスを使用して設定されている場合は、「ライセンス サーバー」タブを使用してライセンス情報を入力および表示します。このオプションは、インストールに Kofax TotalAgility が含まれている場合にのみ使用できます。

Kofax TotalAgility サーバーを使用して Kofax RPA にライセンスが付与されている場合、「ライセンス サーバー」タブのラベルの横にアイコン が表示されます。

Kofax TotalAgility を配布ゲートウェイとして使用すると、それぞれの Kofax RPAManagement Console は、Kofax TotalAgility ライセンス Web API を使用してライセンスの使用状況を取得および報告します。

設定情報については、『Kofax RPA インストール ガイド』の第 3 章「ライセンス情報を提供する」を参照してください。

「ライセンス サーバー」ダイアログ ボックスには次の情報が含まれています。

KTA サーバーの URL	ライセンス サーバーとして機能する Kofax TotalAgility インストールの URL。
ライセンス API キー	Kofax TotalAgility ライセンス API キー。 • Management Console が 1 つの HA クラスタの一部として機能している場合、すべての Management Console に同じライセンス API キーを割り当てる必要があります。 • Management Console が独立して動作していても、同じライセンス サーバーを使用している場合は、すべての Management Console でライセンス サーバー上の別のライセンス API キーを使用する必要があります。
予約する本番 CRE の数	Management Console によって起動時に割り当てられる本番 CRE ライセンスの数。 この設定の目的は、複数の Management Console クラスタが同じ Kofax TotalAgility ライセンス サーバーと通信する場合に、Management Console 間で CRE を分散するようにすることです。 管理者は、この設定を使用して、CRE の数を異なる Management Console クラスタに分散できます。
予約する非本番 CRE の数	予約する非本番 CRE ライセンスの数。 無制限の CRE ライセンスが使用可能な場合、この数は無視され、無制限の数が使用されます。
ユーザーの最大数	Management Console が使用できるユーザーの最大数。 使用可能なライセンスの数に関係なく、この最大値によって制限が設定されます。この目的は、複数の Management Console クラスタが同じライセンスとライセンス サーバーを使用する場合にライセンスを配布するようにすることです。 指定しない場合、ユーザー数は無制限になります。
Design Studio シートの最大数	Management Console が使用できる Design Studio シートの最大数。 最大ユーザー数と同じように機能します。

設定

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、プロジェクト管理者、および RoboServer。

[設定] メニューを使用して、追加の Management Console 設定を行います。

Management Console (ポート番号やセキュリティ設定など) への接続方法に影響を与えるその他のオプションは、『Kofax RPA 管理者ガイド』の「組み込み Management Console の設定」および「RoboServer パラメータ」で説明されているように、RoboServer 設定アプリケーションで設定します。

一般

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、およびプロジェクト管理者。

このセクションでは、データベースの設定、プロキシ サーバー、ロボット ファイル システム サーバーなどを設定できます。

ログイン

このタブでは、ユーザーに電子メール通知を送信して、パスワードを紛失した場合、またはパスワードを忘れた場合に復元できるように設定できます。

前提条件: SMTP サーバーは事前にしておく必要があります。[設定] > [一般] > [SMTP サーバー] を参照してください。

RoboServer 認証

⚠ この設定は非推奨になりました。

このタブでは、設定済みのクラスタに属する RoboServer でロボットを実行する場合に Management Console が使用するクレデンシャル (ユーザー名とパスワード) を設定します。Management Console は、すべての RoboServer で同じクレデンシャルのセットを使用します。これらのクレデンシャルは、『Kofax RPA 管理者ガイド』の「リクエスト認証」セクションで説明されているように、設定済みのすべての RoboServer で設定が一致している必要があります。

RoboServer ログ データベース

このタブでは、Management Console が使用するログ データベースを設定して、スケジュール実行、スケジュール メッセージ、ロボット実行、ロボット メッセージ、ロボット タグ、Desktop Automation サービス メッセージ、およびクラスタ設定でデータベース ログが有効なクラスタに属するすべての RoboServer を保存できます。すべてのログは Management Console によって処理されるため、RoboServer から RoboServer ログ データベースへの直接接続は必要ありません。

ℹ ロギングにデータベースを使用するには、新しいデータベース (スキーマ) を作成するか、既存のデータベースが利用できるかどうかを確認して、データベース サーバーを準備する必要があります。テーブルの作成、テーブルの削除、インデックスの作成、インデックスの削除、データベースでの選択、挿入、更新、および削除を実行する権限を持ったユーザー名とパスワードを取得する必要があります。

Management Console と RoboServer はどちらも、開始時にログ テーブルを自動的に作成します (ログ テーブルが存在しない場合)。ただし、データベース テーブルを作成するためのスクリプトを使用して作成することも可能です。

[データベースに記録] を選択し、ログ データベースを設定します。

プロパティ	説明
ホスト	データベース サーバーのホスト名。IP アドレス、完全修飾ドメイン名 (myhost.kofax.com)、または作業用コンピュータの名前 (PC1X1XXX) などを指定できます。
ポート	データベース サーバーのポート。
スキーマ	データベース スキーマ (またはカタログ) の名前。
タイプ	データベースのタイプ (例: Oracle)。さまざまなタイプのデータベースが Management Console で設定され、Design Studio の起動時に自動的に提供されます。
ユーザー名	データベース用のユーザー名。

プロパティ	説明
パスワード	データベース用のパスワード。

現在の設定をテストするには、[テスト] をクリックします。

このアクションは、データベースとの接続のみをテストします。データベースに適切な権限があるかどうかの判別は、このテストでは行われません。

完了した後に、[保存] をクリックします。

RoboServer ログ データベースには、以下のクリーンアップ閾値を設定できます。

プロパティ	説明
ロボットおよびスケジュールの統計情報の保持日数	ロボットおよびスケジュール統計を維持する日数を指定します。定義するクリーンアップ設定に応じて、古いデータは毎日削除されます。
ロボット実行の最大メッセージ数	単一のロボット実行におけるメッセージ最大件数を指定します。ロボットのメッセージの件数がこのしきい値を超えると、ロボットのメッセージのロギングはその実行に対して停止します。 クリーンアップでは、削除された実行の最も古いロボット実行およびメッセージが削除されます。ログ データベースにおいてパフォーマンス上の問題が発生する場合は、このしきい値を下げてください。より多くの履歴メッセージを保存するには、このしきい値を上げることができます。

データベース テーブルを作成するためのスクリプト

データベースにテーブルを作成してドロップするための SQL スクリプトは、Kofax RPA インストール ディレクトリの `documentation\sql` ディレクトリにあります。たとえば、Windows システムでは、`C:\Program Files\Kofax RPA 11.5.0\documentation\sql` にあります。スクリプト ファイルの名前には、スクリプトが対象とするデータベースの名前が含まれます。

SQL スクリプトは Design Studio をインストールする時に Kofax RPA とともにインストールされます。

`sql` ディレクトリには、以下のように異なるスクリプトを持つ 4 つのサブディレクトリが含まれています。

- `Kapplet`: Kapplets テーブルの作成およびドロップを行うためのスクリプト
- `logdb`: logdb テーブルの作成および削除を行うためのスクリプト
- `mc`: Management Console テーブルの作成およびドロップを行うためのスクリプト
- `statistics`: Kofax Analytics for RPA でレポートを作成するために必要な統計テーブルを作成および削除するためのスクリプト。

Management Console では、Quartz というサードパーティのスケジューリング コンポーネントが使用されます。Quartz もまた、その他のプラットフォーム テーブルに存在する多数のテーブルを必要とします。これらのテーブルは、Management Console の開始時に自動的に作成されますが、スクリプトを使用して手動で作成することも可能です。

以下は、Quartz 検証スクリプトです。

```
select count(*) from QRTZ_SIMPLE_TRIGGERS;
select count(*) from QRTZ_BLOB_TRIGGERS;
select count(*) from QRTZ_CRON_TRIGGERS;
select count(*) from QRTZ_TRIGGER_LISTENERS;
select count(*) from QRTZ_CALENDARS;
select count(*) from QRTZ_FIRED_TRIGGERS;
select count(*) from QRTZ_LOCKS;
select count(*) from QRTZ_PAUSED_TRIGGER_GRPS;
select count(*) from QRTZ_SCHEDULER_STATE;
select count(*) from QRTZ_JOB_LISTENERS;
select count(*) from QRTZ_TRIGGERS;
select count(*) from QRTZ_JOB_DETAILS;
```

Analytics データベース

このタブでは、Kofax Insight Analytics ダッシュボードで使用するデータベースへの接続を設定できます。

[**Analytics** データベースに統計情報を記録] を選択して、RoboServer ログ データベースと同じ方法で Analytics データベースを設定します。設定をテストすることができます。完了した後に、[保存] をクリックします。

次のクリーンアップ閾値は、Analytics データベース用に設定できます。

プロパティ	説明
RoboServer 統計情報の保持日数	統計を保持する日数を指定します。定義するクリーンアップ設定に応じて、古いデータは毎日削除されます。

新しいデータベース (スキーマ) を作成するか、既存のデータベースが利用できることを確認して、データベース サーバーを準備する必要があります。データベースでテーブルの作成、テーブルの破棄、インデックスの作成、インデックスの破棄、作成、挿入、更新、および削除を実行する権限があるユーザー名とパスワードを取得する必要があります。詳細については、RoboServer ログ データベース を参照してください。

Analytics 機能の可用性は、使用するライセンスによって異なります。Analytics のライセンスがある場合は、Management Console の [設定] メニューに [Analytics データベース] セクションが表示されます。

SMTP サーバー

このタブでは、Management Console でパスワード変更や口ボットの失敗通知などの電子メール通知を送信するために使用する RoboServer の SMTP サーバー設定を定義できます。

SMTP サーバーは、以下のプロパティを使用して設定します。

プロパティ	説明
ホスト	SMTP サーバーのホスト名。
ポート	SMTP サーバーのポート。
ユーザー名	SMTP サーバーで認証が必要な場合は、ユーザー名を入力します。

プロパティ	説明
パスワード	SMTP サーバーで認証が必要な場合は、パスワードを入力します。
暗号化	• なし: 資格情報および電子メールは、クリア テキストで送信されます。 • **TLS**: TLS 暗号化が使用されます。このオプションは、SMTP サーバーに信頼できる証明書がある場合にのみ使用できます (サーバーに自己署名証明書がある場合は、キーツール ユーティリティを使用して JVM のトラストストアにエクスポートおよびインポートする必要があります)。STARTTLS SMTP 拡張を使用します。 • **SMTPS**: SMTP over SSL電子メールの送信に使用する、セキュリティで保護された通信チャネルを確立します。これが **SMTP** サーバーでサポートされることはめったにありませんが、サーバーで自己署名証明書が使用されている場合でも機能します。
送信元アドレス	通知の送信元の電子メール アドレス。

プロキシ サーバー

このタブでは、外部 API にアクセスするための OAuth セキュリティ トークンを生成する場合など、Management Console が外部サーバーに接続するときに経由するプロキシ サーバーを指定できます。

プロパティ	説明
プロキシ サーバーを使用	プロキシ サーバーを使用する場合に選択します。選択しない場合は、直接接続が使用されます。
ホスト	プロキシ サーバーのホスト名。
ポート	プロキシ サーバーのポート。
ユーザー名	プロキシ サーバーで認証が必要な場合は、ユーザー名を入力します。
パスワード	プロキシ サーバーで認証が必要な場合は、パスワードを入力します。

ロボット ファイル システム サーバー

このタブでは、RoboServer および Design Studio インスタンスからドライブへのすべてのマッピングを処理するロボット ファイル システム サーバーを設定できます。ロボット ファイル システム サーバーは、以下のプロパティを使用して設定します。

プロパティ	説明
ロボット ファイル システム サーバーを使用	ロボット ファイル システム サーバーの使用を有効にする場合に選択します。
URL	ロボット ファイル システム サーバーの URL。たとえば、`http://myserver.mydomain:8080/rfs` など。

ベース URL

このタブを使用して、Management Console ベース URL を定義します。通常、ベース URL は自動的に設定されるため、変更する必要はありません。

オフライン モードでこのドキュメントを利用するには、[ローカルのドキュメントを使用] を選択してください。[ローカル ドキュメント ベース **URL**] で、ドキュメントが含まれている Tomcat Web サイトの URL を指定します。

例: `http://localhost:8080/KofaxRPAManagementConsoleDocumentation`

[ローカルのドキュメントを使用] が選択されている場合、Management Console では、インターネットに接続されている場合でもデフォルトでオフライン バージョンのドキュメントが使用されます。

オフライン ドキュメントについての詳細は、『Kofax RPA インストール ガイド』を参照してください。

DAS の設定

このタブでは、Desktop Automation サービスが Management Console に ping を送信する間隔をミリ秒単位で設定できます。最初の ping で、Management Console は Management Console に ping を送信する頻度に関する情報を Desktop Automation サービスに渡します。

Design Studio

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、およびプロジェクト管理者。

データベースやデバイスを Design Studio で使用できるようにするには、このセクションで選択を行います。

シェア データベースは、Management Console ライセンスを取得することで、アクティブ化されたすべての Design Studio クライアントに送信されます。「Production」という名前のデフォルトのクラスタが、シェア データベース用に自動的に作成されます。

Design Studio に送信するデータベース

- 特定のクラスタを選択すると、このクラスタのデータベースを Design Studio で利用できるようになります。データベースがそのクラスタに存在することを確認してください。これは、**[リポジトリ] > [管理] > [RoboServer] > [クラスタ設定] > [データベース]** タブで確認できます。
- [すべて] を選択すると、すべてのクラスタのデータベースを Design Studio で利用できるようになります。
- [マップ済み] を選択すると、Design Studio ユーザーがアクセスできるプロジェクトにマップされているデータベースのみを指定できます。このデータベースを指すデータベース マッピングが存在することを確認してください。データベース マッピングは、**[リポジトリ]** > **[データベース マッピング]** セクションで確認できます。
- データベースに対して [なし] が選択されている場合は、データベース タイプとドライバーだけを Design Studio で利用できます。

Design Studio で利用できる特別なセットのデータベースを定義するには、特殊クラスタを作成できます。このアプローチは、プロダクション データベースを Design Studio に送信しないことを希望する場合に便利です。

Design Studio のデータベース マッピングの詳細については、データベースのマッピングを参照してください。

Design Studio で許可されるデバイス

- 特定のクラスタを選択すると、このクラスタのデバイスを Design Studio で利用できるようになります。
- [すべて] を選択すると、すべてのデバイスを Design Studio で利用できるようになります。

- [なし] を選択すると、デバイスは Design Studio で利用できなくなります。

RoboServer のロールを持つユーザーと RPA 管理者グループに属するユーザーは、選択したクラスタに関係なく、すべてのデバイスにアクセスできます。

データベース タイプ

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、およびプロジェクト管理者。

このセクションでは、Apache Derby (開発用データベース)、PostgreSQL、Oracle、MySQL、Microsoft SQL Server のいずれかのデータベース タイプを定義できます。

既存のデータベース タイプを編集するには、それぞれのタブを選択します。新しいタイプを作成するには、[新しいタイプ] タブで次のプロパティを指定して、[保存] をクリックします。例については、データベース Microsoft SQL Server データベース タイプの追加を参照してください。

プロパティ	説明
名前	データベース タイプを識別する名前。
JDBC ドライバー	ドライバーの JDBC ドライバー クラス (ドライバーは、データベース ドライバーの下にアップロードする必要があります)。
接続 URL テンプレート	任意のタイプのデータベースの接続 URL を定義するテンプレート文字列。テンプレート文字列で使用可能な変数は次のとおりです。 **${ServerName}** データベース サーバーのサーバー名 (ホスト) を定義します。 **${Schema}** データベースのスキーマ (または、データベース ベンダーに応じてデータベース/カタログ) を定義します。 接続文字列テンプレートは、`jdbc:mysql://${ServerName}/${Schema}` のように表します。この文字列は、デフォルトのポート (ポートの指定なし) および ${ServerName} で指定されたサーバーで動作し、${Schema} で指定されたスキーマを使用する MySQL データベースの接続文字列を定義します。変数は、特定のタイプのデータベースを作成したときに指定した値です (「クラスタ データベース」セクションを参照してください)。
SQL フレーバー	リストからデータベース タイプを選択します。 サポートされるデータベースの詳細については、Kofax RPA 11.5.0 のドキュメント サイトの『Kofax RPA 技術仕様』ドキュメントを参照してください。https://docshield.kofax.com/Portal/Products/RPA/11.5.0-nlfihq5gwr/RPA.htm。

データベース タイプは、Design Studio クライアントにも送信されます。

Kofax RPA では、「SQL フレーバー」リストで指定されたデータベース タイプ以外の追加はサポートされていません。ただし、データベース タイプを変更して、標準以外のポートで実行されているデータベース サーバー、または別の認証方法を必要とするデータベース サーバーを定義するタイプを追加できます。

データベース Microsoft SQL Server データベース タイプの追加

この手順を使用して、変更したデータベース タイプを Management Console に追加します。例として、Windows 統合セキュリティを必要とする Microsoft SQL Server を追加する方法を示します。

1. Management Console で、**[設定]** > **[データベース タイプ]** を選択します。
2. [新しいタイプ] タブで、次の値を指定します。
   - 名前：Microsoft SQL Server (統合セキュリティ)
   - JDBC ドライバー：com.microsoft.sqlserver.jdbc.SQLServerDriver
   - 接続 URL テンプレート：jdbc:sqlserver://${ServerName};databaseName=${Schema};integratedSecurity=true
   - SQL フレーバー: [**Microsoft SQL Server**] を選択します。
3. [保存] をクリックして、データベース タイプを追加します。
4. Microsoft の Web サイトから "Microsoft JDBC Driver 4.0 for SQL Server" をダウンロードして、ディスク上のフォルダに抽出します。
5. `sqljdbc_auth.dll` を Kofax RPA インストール フォルダの `\lib` ディレクトリにコピーします。ファイルは、`Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu\auth\x64` または `\86` フォルダの下の抽出したフォルダ内にあります。
   他のコンピュータで RoboServer が実行されている場合は、各 RoboServer に `sqljdbc_auth.dll` をインストールする必要があります。
6. `sqljdbc.jar` ファイルを Management Console にアップロードします。[設定] で [データベース ドライバー] を選択し、[ドライバー **JAR** ファイルのアップロード] をクリックします。対応する JDBC ドライバーを参照します。.jar ファイルは、`Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu` の下にある抽出したフォルダ内にあります。
7. Kofax RPA Management Console を再起動します。

これで、Management Console の **[管理]** > **[RoboServer]** セクションで新しく作成したデータベース タイプを使用するデータベースに接続できるようになりした。

Windows 統合セキュリティを必要とする **Microsoft SQL Server** に関する注意

- 必要なデータベース .dll ファイルをコピーしない場合は、64 ビットではなく 32 ビット バージョンの auth .dll を使用します。また、Management Console を再起動しない場合は、「データベースへの接続エラー：このドライバーは統合認証向けに設定されていません。」というエラー メッセージが表示されることがあります。
- サーバーがドメインの一部ではない場合、「データベースへの接続エラー：ログインに失敗しました。信頼できないドメインからログインが試行されたため、Windows 認証では使用できません。」というエラー メッセージが表示されることがあります。

Kofax RPA でデータベースを使用することに関する詳細については、次のトピックを参照してください。

- データベース タイプ
- データベース ドライバー
- データベースの操作

データベース ドライバー

このトピックは、次のロールを持つユーザーを対象としています: 管理者およびプロジェクト管理者。

[データベース ドライバー] オプションを使用して、データベース JDBC ドライバーをアップロードし、インストールされている JDBC ドライバーのリストを表示します。これらのドライバーは必須であり、データベース タイプに基づいています。

- データベースとドライバーのインストールの詳細については、『Kofax RPA インストール ガイド』を参照してください。
- データベースとドライバーの設定の詳細については、『Kofax RPA 管理者ガイド』を参照してください。

Oracle JDBC ドライバーは、組み込みの Management Console、RoboServer、および Design Studio マシンにインストールされている場合にサポートされます。Oracle JDBC ドライバーを Management Console にアップロードしないでください。詳細については、『Kofax RPA インストール ガイド』の「ロボットを動かす前に Management Console を設定します」を参照してください。

同じ Management Console にアップロードする必要があるデータベース ドライバーは、タイプごとに 1 つのみです。たとえば、同じ Management Console で 2 つの異なる MySQL ドライバーを同時に使用することはできません。同じタイプの複数のドライバーをアップロードした場合は、最初のドライバーのみが使用されます。

データベース タイプと同じように、データベース ドライバーは Design Studio クライアントに送信されます。MySQL データベースで Management Console を実行する場合、Tomcat では MySQL ドライバーが必要になります。つまり、MySQL データベースは、[ログ ビュー] の Management Console で使用する場合や、接続をテストする場合に機能します。ただし、MySQL データベースをすべての RoboServer で動作させるには、MySQL ドライバーをアップロードして、RoboServer と Design Studio に送信できるようにする必要があります。

エンタープライズ Management Console では、Derby JDBC ドライバーは分配されません。Derby JDBC ドライバーのダウンロード情報については、Apache Derby の Web サイトを参照してください。エンタープライズ Management Console では、MySQL または別のエンタープライズクラス データベースを使用することをお勧めします。

ヒント

特定のデータベース タイプでは、より大きなファイルを格納するために変数設定を変更する必要がある場合があります。

例: MySQL データベースでは、「`max_allowed_packet`」変数の値を増やすことが必要になる場合があります。多くのインストールでは、この値は 1 MB に設定されています (つまり、1 MB よりも大きいデータベース ドライバーを格納することはできません)。

ドライバーをアップロードするときに問題が発生した場合は、データベースのドキュメントを参照してください。問題を特定するには、エラー メッセージと Management Console ログを確認してください。

セキュリティに関する注意

デフォルトでは、ローカル ホストから Management Console にアクセスする際に、JDBC ドライバーをアップロードできるのは管理者だけです。管理者としてアクセスしない場合、「ドライバー JAR ファイルのアップロード」 ボタンは無効になります。

- 組み込みモードで Management Console を実行している場合、**RoboServer** 設定でこの動作を変更できます。詳細については、『Kofax RPA 管理者ガイド』の「組み込み Management Console の設定」セクションを参照してください。
- Tomcat で Management Console を実行している場合は、『Kofax RPA Kofax RPA 管理者ガイド』の「Tomcat 管理コンソール」にある「セキュリティ」セクションを参照してください。

Process Discovery Analyzer

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、およびプロジェクト管理者。

Kofax RPAProcess Discovery Analyzer は、記録された生データを処理し、Kofax Analytics for RPA ダッシュボードの絞り込みデータを生成するためのものです。

Process Discovery Analyzer データベースの接続設定、ランタイム パラメータ、クラスタ設定を指定し、Process Discovery Analyzer ページでデータベース プロビジョニングを実行するための資格情報を指定します。編集が完了したら、[保存] をクリックして設定を保存します。

オプション	説明
スケジュールの設定	Analyzer の実行方法を選択します。 • [特定時に実行]: [特定時に実行] フィールドで Analyzer を実行するには、時間値のコンマ区切りリストを `hh:mm` 形式で設定します。時間は、以下のように指定できます: 1:00、15:30 • [定期的に実行]: [実行間隔 **(**時間**)**] フィールドで、Analyzer の実行間隔を設定します。

オプション	説明
クラスタ設定	Process Discovery Analyzer のクラスタ設定を指定します。詳細については、Process Discovery Analyzer クラスタ を参照してください。 クラスタ設定を変更した場合は、マスター ノードを再起動します。別のコンピュータをマスター ノードとして割り当てた場合は、現在のマスター ノードおよび新しく割り当てたマスター ノードを両方とも再起動します。たとえば、現在、クラスタに A、B、および C という 3 つのノードが含まれていて、「A」がマスター ノードになっているとします。この状況で「B」をマスター ノードとして割り当てた場合は、「A」と「B」を再起動します。 • [マスター アドレス]: IP アドレス (IPv4 のみ)、完全修飾ドメイン名 (`myhost.company.com` など)、またはクラスタのマスター アプリケーションをホストしているコンピュータのコンピュータ名。ネットワークで DHCP が使用されている場合は、マスター アドレスとしてドメイン名を指定します。 • [ネットワーク パターン]: これは、クラスタ コンピュータを `192.168.*.*` や `10.*.*.*` などの特定のサブネットにバインドするために役立つオプションのパラメータです。異なる IP アドレスを持つ複数のネットワーク インターフェイスをコンピュータで実行している場合は、パターンを入力します。 次のポートは、標準ポートがネットワークでブロックされている場合にのみ変更する必要があります。 • [マスター ポート]: マスター アプリケーション ホストのポートを指定します。 • [マスター **WebUI** ポート]: マスター アプリケーション ホストのアクティビティ モニター Web インターフェイスにアクセスするポートを指定します。 • [ワーカー **WebUI** ポート]: ワーカー アプリケーション ホストのアクティビティ モニター Web インターフェイスにアクセスするポートを指定します。 • マスター メモリ **(GB)**: マスター プロセスの最大メモリ容量を指定します。マスター ノードの Analyzer ログで `outofmemory` タイプのエラーが発生した場合は、この設定の値を大きくしてマスター ノードを再起動します。 • ワーカー メモリ **(GB)**: プロセスの開始時に Apache Spark がワーカー プロセス用に予約するメモリの量を指定します。Process Discovery で使用される Apache Spark クラスタ テクノロジーにより、ワーカー ノードと同様の処理能力とメモリ量を持つコンピュータが暗黙的に使用されます。ワーカー ノードごとに RAM の空き容量が異なる場合は、ノードで利用可能な空きメモリの最小量を指定します。ノードにワーカー プロセス用に予約するための十分なメモリがない場合、ノードは起動されますが、データ分析には使用されません。ノードを確認するには、ブラウザで `http://[ワーカーの IP アドレス]:[ワーカーの UI ポート]` を開き、ノードで実行中の エグゼキューターがあることを確認します。 マスター ノードとワーカー ノードのメモリ量は、デフォルト設定よりも少なくしないでください。

オプション	説明
データベースの設定	Process Discovery Analyzer に対し絞り込みデータを保存するためのデータベース接続設定を指定します。 • [ホスト]: データベース サーバー名または IP アドレス。Process Discovery の展開に Docker を使用する場合、これが、Process Discovery が Docker コンテナーで実行されるコンピュータとなります。 • [ポート]: データベースにアクセスするためのポートを指定します。 • スキーマ: スキーマの名前。 • [タイプ]: データベースのタイプを指定します。このパラメータは変更できません。 • [ユーザー名]: Analyzer でデータベースにアクセスするためのアカウント名。 • [パスワード]: この Analyzer でデータベースにアクセスするためのアカウントのパスワード。 • テスト ボタン: データベースへの接続を確認します。
データベースの詳細設定	Analyzer 実行設定を指定します。 このセクションの設定を変更する場合は、Kofax テクニカル サポートにお問い合わせください。 • [接続タイムアウト (秒)]: データベース操作のタイムアウト設定。 • [バッチ サイズの読み取り]: 1 つのトランザクションで Agent データベースから読み取られたイベントの数。 • [バッチ サイズの書き込み]: 1 つのトランザクションで Analyzer データベースに書き込まれたイベントの数。
データベースのプロビジョニング	資格情報を入力し、データベース スキーマを作成し、[データベースの設定] で指定されたユーザーにアクセス権を付与します。 [データベースの設定] タブで設定を保存するまで、データベースのプロビジョニング ボタンは無効になります。 • データベース管理者の名前: データベースにアクセスできる管理者アカウントの名前を指定します。 • データベース管理者のパスワード: 管理者アカウントを保護するためのパスワードを指定します。 • スキーマの作成 ボタン: 指定した資格情報を使用して、データベースでスキーマを作成します。スキーマが存在する場合は、既存のスキーマを上書きするか、キャンセルするよう求められます。 • ユーザーにアクセス権を付与 ボタン: 必要なデータベース アクセス権を [データベースの設定] で指定したユーザーに付与します。ユーザーが存在する場合は、既存のユーザーを再作成するか、キャンセルするように求められます。ユーザーを再作成すると、以前のユーザーのアクセス権およびその他のユーザー設定は失われます。

Process Discovery グループ

このセクションは、次のロールを持つユーザーが利用できます: 管理者、閲覧者、およびプロジェクト管理者。

このセクションには、Process Discovery グループ設定が含まれています。ここで指定したグループにより、`finacial_dept`、`hr_dept`、`sales_dept` など、インストールされている Process Discovery Agents とコンピュータが論理的に組み合わされます。Process Discovery Agents をデプロイする場合は、ここで作成したグループの名前を指定します。

新しいグループを作成するには、左側のメニューの [**Process Discovery** グループ] をクリックし、[新しいグループを作成] タブで必要なパラメータを指定して、[保存] をクリックします。指定した Process Discovery グループ名の新しいタブが [**Process Discovery** グループ] セクションに表示され、新しく作成したグループ タブに自動的に移動されます。

既存のグループの設定を編集している場合は、編集後に [保存] をクリックして設定を保存します。

オプション	説明
グループ設定	一般的なグループ パラメータを指定します。 • [グループ名]: 新しいグループ名を指定するか、既存のグループ名を編集します。 • [レコード]: このグループのアクションの記録を有効にします。 • [分析]: このグループのアクション分析を有効にします。
記録設定	[アプリケーション無視リスト]: これは、データの記録および処理時に Process Discovery によって無視されるプログラムのプロセス名のリストです (コンマ区切り、大文字と小文字の区別なし)。プログラム プロセス名は、Windows タスク マネージャで確認できます。無視されるプログラムとして、作業プロセスには通常含まれないメッセンジャを指定します。このリストを指定して、データから余分な「ノイズ」を除去することにより、検出されたプロセスに対する Agent と Analyzer のパフォーマンスおよび信頼性を向上させます。
データベースの設定	Agent がデータにアクセスして格納するための新しいデータベース パラメータを指定します。 • [アドレス]: データベース サーバー名または IP アドレス。 • [ポート]: データベースにアクセスするためのポートを指定します。 • スキーマ: スキーマの名前。 • [タイプ]: データベースのタイプを指定します。このパラメータは変更できません。 • [ユーザー名]: このグループの Agent がデータベースにアクセスするためのアカウント名。 • [パスワード]: このグループの Agent がデータベースにアクセスするためのアカウントのパスワード。 • [テスト] ボタン: データベースへの接続を確認します。

オプション	説明
Agent データベースのプロビジョニング	資格情報を入力し、データベース スキーマを作成し、[データベースの設定] で指定されたユーザーにアクセス権を付与します。これらのオプションは、既存のグループで使用できます。 設定を保存するまで、[データベースのプロビジョニング] ボタンは無効になります。 • [データベース管理者の名前]: データベースにアクセスできる管理者アカウントの名前を指定します。 • [データベース管理者のパスワード]: 管理者アカウントを保護するためにパスワードを指定します。 • [スキーマの作成] ボタン: 指定のクレデンシャルを使用して、データベースでスキーマを作成します。スキーマが存在する場合は、既存のスキーマを上書きするか、キャンセルするよう求められます。 • [ユーザーにアクセス権を付与] ボタン: 必要なデータベース アクセス権を [データベースの設定] で指定したユーザーに付与します。ユーザーが存在する場合は、既存のユーザーを再作成するか、キャンセルするように求められます。ユーザーを再作成すると、以前のユーザーのアクセス権およびその他のユーザー設定は失われます。
分析設定	このグループの Analyzer 実行設定を指定します。詳細については、以下の「分析設定の説明」セクションを参照してください。 • [プロセス インスタンスの最小数]: プロセスを形成する、類似した一連のユーザー アクションの最小数を設定します。 • [プロセス インスタンスのアクション最小数]: 検出するプロセス インスタンス内のユーザー アクションの最小数を設定します。 • [プロセス インスタンス内の順次ノイズ アクションの最大数]: プロセス インスタンスに関連付けられていない連続するユーザー アクションの最大数を設定します。 • [プロセス インスタンスの非アクティブ状態の最大間隔 (分)]: 検出されたプロセス インスタンスでユーザー非アクティビティ期間の最大値を設定します。 • [プロセス インスタンスのアプリケーション最大数]: プロセス インスタンス内のさまざまなアプリケーションの最大数を設定します。 • [スクリーンショットをぼかす]: このオプションが選択されている場合、Analyzer は、スクリーンショット上のテキストを Insight ダッシュボード ビューで認識できないようにします。Agent データベース内のスクリーンショットはそのままになります。スクリーンショットに個人情報、パスワード、クレジット カード番号などの機密データが含まれる場合、このオプションを選択することを強くお勧めします。

オプション	説明
記録の詳細設定	このグループの記録の詳細設定を指定します。 • [非 **UIA** アプリケーション]: Process Discovery が UI オートメーション API ではなく ISA (インテリジェント スクリーン オートメーション) を使用するプログラムのプロセス名のリスト (コンマ区切り) を入力します。デフォルトでは、このフィールドには、一部の通信プログラム、Java プログラム、Chrome を除くすべてのブラウザ、および ISA を使用するときにより信頼性の高い結果を示すその他のプログラム名前が含まれます。Kofax RPA は Chrome ブラウザのネイティブ UI オートメーション サポートを実装しているため、環境内のデフォルト ブラウザとして Chrome を使用することをお勧めします。 • [**Agent SQLite DB** 最大サイズ **(MB)**]: Agent が実行されているコンピュータにローカルに保存されるデータの最大サイズを設定します。データベースへの接続を確立できない場合、Agent は収集した情報をローカルに保存します。保存されたデータが制限を超えると、Agent は最も古いデータの上書きを開始します。デフォルト設定では、2 ~ 3 日間のデータ収集用のストレージが提供されます。長期間にわたるネットワーク障害が頻繁に発生する場合は、サイズの上限を引き上げてください。 • [最小 **UIA** ツリー不変期間 **(ms)**]: プログラム ツリーが変化しなくなってからツリーが記録されるまで Agent が待機する時間です。タイムアウトが短いほど Agent がツリーを記録する時間が短縮され、より適切な結果を得ることができます。ユーザーのコンピュータ上のプログラムの応答が遅くなる場合は、この数値を大きくしてください。デフォルト設定を変更する場合は、十分検討するようにしてください。

オプション	説明
分析の詳細設定	このグループの分析出力の詳細設定を指定します。以下の設定のデフォルト値を変更する場合は、十分検討するようにしてください。 • [演算スコアの最大距離] (0 ~ 100): グループ内のアクションの数に基づいてユーザー アクションを比較するための近接スコアを設定します。 • [アクション ブロック スコアの最小値] (0 ~ 100): 隣接するアクション ブロックを組み合わせるための類似スコア閾値を指定します。数値を小さくすると、Analyzer の許容性が高まり、より多くのブロックが結合されるようになります。数値を大きくすると、アクション ブロックを結合するときの Analyzer の精度が高まります。 • [ストロング ペア スコアの最小値] (0 ~ 100): プロセスの検出中に複数のアクションをペアとして結合するためのユーザー アクション類似スコアを設定します。数値を小さくすると、検出されるプロセス数を増やすことができますが、誤検出も増えます。数値を大きくすると、その逆となります。 • [プロセス インスタンス間の最大距離] (0 ~ 100): 検出されたプロセス インスタンスをクラスタに結合するための最大差スコアを設定します。指定した数値が大きいほど、類似していると見なされて結合されるプロセスは増えますが、類似性が低いプロセスがクラスタに結合される可能性は大きくなります。数値を小さくすると、作成されるクラスタは増えて、その中に含まれるプロセス インスタンスは少なくなります。 • [比較のためのブロック内のアクション]: ブロックの比較と結合に使用される、これらのブロック内のユーザー アクションの最大数を設定します。数値を小さくすると、Analyzer の「ノイズ」アクションに対する許容度が高まり、より多くのプロセスが検出されるようになりますが、プロセスが短くなって、長いプロセスが見逃される可能性が生じます。数値を大きくすると、類似しないアクションが含まれているブロックが Analyzer によって結合される可能性が高くなります。 • [**ISA** での **X** 方向の位置の最大差 **(px)**]: 類似したアクションを検出したときに別の要素としてマークされる、x 軸の最大要素オフセットをピクセル単位で設定します。この設定は、ISA テクノロジを使用して検出されたプロセスに適用されます。この数値を大きくすると、より多くのアクションが類似していると見なされてプロセスの誤検出が増える可能性が生じます。数値を小さくすると、その逆となります。 • [**ISA** での **Y** 方向の位置の最大差 **(px)**]: 類似したイベントを検出したときに別の要素としてマークされる、Y 軸の最大要素オフセットをピクセル単位で設定します。この設定は、ISA テクノロジを使用して検出されたプロセスに適用されます。この数値を大きくすると、より多くのアクションが類似していると見なされてプロセスの誤検出が増える可能性が生じます。数値を小さくすると、その逆となります。

分析設定の説明

このセクションで使用される用語については、Process Discovery 用語集を参照してください。

- [プロセス インスタンスの最小数]: この数により、収集されたデータで同様なアクションのシーケンスが発生した場合に、シーケンスがプロセスとしてマークされるようにする回数を指定します。この数は、収集するデータの量 (人数やデータ収集プロセスの期間など) と一致している必要があります。ユーザーの数が多く、データの収集に相当の時間がかかる場合は、より大きな数を指定することにより、分析の品質の向上、自動化に不利となる短い繰り返しアクションの除外、レポートのサイズの縮小、および分析時間の短縮を図ることができます。選択した部門の作業の概要を把握した上で、この数を変更することをお勧めします。20 人以上の部門で 2 週間以上かけて収集されたデータに関しては、15 という数値を最初に指定し、レポートで検出されたプロセスに応じて数値を増減することがで

きます。このプロパティとその他のプロパティの値が異なる同じデータを分析して、レポートを調整することをお勧めします。

- [プロセス インスタンスのアクション最小数]: このプロパティにより、Analyzer によって検出され、プロセスとしてマークされる必要のある固有のユーザー アクション (マウス クリック、テキスト フィールドへのテキストの入力、テキストのコピーなど) を指定します。5 と 15 など、一意のアクション数が異なるプロセス を検出する場合は、このフィールドに小さい方の数 (5) を指定する必要があります。
- [プロセス インスタンス内の順次ノイズ アクションの最大数]: これは、Analyzer がこのインスタンスを検出し、アクションのシーケンスをプロセス インスタンスに分割する場合に、プロセス インスタンスとは無関係の一部のユーザー アクションを判別するために役立つ追加の設定です。この設定により、通常のアクションの過程で、誤ったクリックやタイプ ミス、重要なインスタント メッセージ、同じアプリケーション内の他のタスクなどにユーザーが気を取られている場合でも、プロセス インスタンスを検出できます。プロセス インスタンスには、指定した数を超える連続したノイズ アクションを含めることはできません。たとえば、通常の一連のアクションの実行中に他のユーザーが何らかの操作を開始して、このフィールドで指定したノイズ アクションの数を超えた場合、Analyzer はこのプロセス インスタンスが終了したと見なします。この設定の数を増やすと、結合したより大きなプロセス インスタンスが検出される可能性があります。数を減らすと、プロセス インスタンスが断片化されます。
- [プロセス インスタンスの非アクティブ状態の最大間隔 **(**分**)**]: この設定により、コンピュータでのユーザーの非アクティブ状態、つまりユーザーがコンピュータから離れている場合の最大持続時間を指定します。ユーザーが非アクティブ状態である期間が指定した値を超えると、プロセスが検出されなくなります。指定する値には、検出するプロセスの非アクティビティ期間より若干長い値を指定する必要があります。
- [プロセス インスタンスのアプリケーション最大数]: このプロパティには、プロセス インスタンスに関係するアプリケーションの最大数を含める必要があります。社内ユーザーが通常のタスクを実行する場合に使用するアプリケーションの数を考慮した上で、デフォルト値を変更してください。アプリケーションの数が多い場合、データの信頼性が低下する可能性があります。

KTA 設定

このセクションは、次のロールを持つユーザーが利用できます: 管理者、プロジェクト管理者、および RoboServer。

このセクションを使用して、Kofax TotalAgility インストールを参照するロボットの実行中に Management Console が使用する資格情報を設定します。

新しい Kofax TotalAgility インストールを追加するには、[新しい **KTA** 設定] タブで次のプロパティを指定します。

プロパティ	説明
KTA インストール名	新しい TotalAgility インストールの名前を指定します。
URL	TotalAgility を実行しているコンピュータの URL を指定します。 HTTP を介した TotalAgility との接続は、統合 Windows 認証が有効な状態で設定した場合のみサポートされます。HTTPS を介した TotalAgility との接続は、手動ログオン オプションを使用して設定した場合のみサポートされます。
ユーザー名	選択したインストールに接続するユーザー名を入力します。これは TotalAgility へのログインに使用するユーザー名にする必要があります。

プロパティ	説明
パスワード	選択したインストールに接続するパスワードを入力します。これは TotalAgility へのログインに使用するパスワードにする必要があります。

[テスト] をクリックして接続をテストします。テスト接続を実行すると、Management Console では実行状態 (成功または失敗とその説明) を示すメッセージが表示されます。

変更を有効化するには、[保存] をクリックします。

電子メール アカウント

このセクションは、次のロールを持つユーザーが利用できます: 管理者およびプロジェクト管理者。

このセクションには、電子メール トリガー機能用に設定した電子メール アカウントが一覧表示されます。このセクションを使用して、新しいアカウントの追加や不要になったアカウントの削除を行います。

[「電子メール アカウント」] セクションの上部にある [プロジェクト] ドロップダウン リストで、表示する電子メール アカウントを持つプロジェクトを選択します。それぞれの電子メール アカウントに関する情報の表示方法を次のように変更できます。

- 右側の メニュー アイコンを使用して、電子メール アカウントに表示するテーブル列を選択します。
- 右側の 更新アイコンをクリックして、表示された情報を更新します。
- 右側の リセット アイコンをクリックして、カスタム列の設定をリセットします。
- ページごとに表示するアイテムの数を選択し、右下隅のナビゲーション メニューを使用してページ間を移動します。

デフォルトでは、電子メール トリガーごとに次の情報が表示されます。

列	説明
名前	電子メール アカウントの名前。
説明	アカウントの説明。
プロジェクト名	電子メール アカウントにリンクされたプロジェクト。

電子メール アカウントを追加

1. 新しい電子メール アカウントを追加するには、左上隅の + 記号をクリックします。
   「新しい電子メール アカウントを作成」 ダイアログ ボックスが表示されます。
2. アカウント タイプを選択し、次のプロパティを指定します。

   Google Gmail と **Microsoft 365** アカウントのプロパティ

プロパティ	説明
名前	変数の名前を指定します。
説明	アカウントの説明を指定します。
プロジェクトに割り当て	電子メール アカウント用の Kofax RPA プロジェクトを選択します。

プロパティ	説明
OAuth を使用	OAuth ユーザー資格情報を使用して追加のセキュリティを提供する場合は、このオプションを選択します。このオプションを選択すると、パスワード フィールドが非アクティブになります。
OAuth ユーザー	OAuth ユーザー資格情報の使用を選択すると、このフィールドがアクティブになります。ドロップダウン リストには、選択した電子メール アカウント プロバイダとプロジェクトに関連付けられているすべての OAuth ユーザーが含まれています。リストから必要な OAuth ユーザーを選択します。
電子メール	アカウントのログイン名を指定します。
パスワード	アカウントのログイン パスワードを指定します。[2]

IMAP アカウント

プロパティ	説明
名前	変数の名前を指定します。
説明	アカウントの説明を指定します。
プロジェクトに割り当て	電子メール アカウント用のプロジェクトを選択します。
便宜的な **TLS** を使用	TLS を使用して電子メール アカウント プロバイダとの接続を暗号化するには、このオプションを選択します。
ホスト	IMAP ホスト名を指定します。
ポート	選択した IMAP ホストのポート番号を入力します。
プロトコル	IMAPS (IMAP over SSL) または IMAP を使用するプロトコルを選択します。
電子メール	電子メール アドレスを指定します。
パスワード	指定した電子メール アドレスのパスワードを入力します。

3. [テスト] をクリックして接続をテストします。テスト接続を実行すると、Management Console では実行状態 (成功または失敗とその説明) を示すメッセージが表示されます。
4. 変更を有効化するには、[保存] をクリックします。

さらに、⋮ コンテキスト メニューから電子メール アカウントに対して次のアクションを実行できます。

- 編集:「[新しい電子メール アカウントを作成]」 ダイアログ ボックスと同じフィールドが含まれます。

リストからアカウントを削除するには、アカウントを選択し、🗑 ゴミ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

[2] 電子メール アカウント プロバイダによっては、アプリのパスワードを作成してこのフィールドに指定し、サーバーでアカウントがブロックされないようにする必要が生じることがあります。詳細については、電子メール プロバイダのドキュメントを参照してください。

CyberArk 構成

このセクションは、次のロールを持つユーザーが利用できます: 管理者およびプロジェクト管理者。

このセクションを使用して、CyberArk パスワード マネージャにアクセスする場合に使用する Management Console の資格情報を設定します。

CyberArk アプリケーションをアップロードする方法については、CyberArk アプリケーションを参照してください。Kofax RPA で CyberArk を使用する方法については、CyberArkを参照してください。

以下のプロパティを使用してクレデンシャルを設定します。

プロパティ	説明
CyberArk URL	CyberArk 統合クレデンシャル プロバイダ ホストの URL を指定します。
CyberArk ポート	CyberArk 統合クレデンシャル プロバイダ ホストの ポート番号を指定します。
IIS アプリケーション名	インターネット インフォメーション サービス (IIS) で指定された CyberArk 統合クレデンシャル プロバイダ アプリケーションの名前を指定します。
サーバー証明書	CyberArk 統合クレデンシャル プロバイダの TLS サーバー証明書をアップロードします。これは、CyberArk 統合クレデンシャル プロバイダ Web サービスに自己署名証明書を使用する場合にのみ必要です。 • 新しい証明書をアップロードするには、[証明書の変更] ボタンをクリックします。 • 証明書を削除するには、[証明書の削除] × ボタンをクリックします。

[テスト] をクリックして接続をテストします。テスト接続を実行すると、実行状態を含むメッセージが表示されます。

CyberArk アプリケーション

このセクションは、次のロールを持つユーザーが利用できます: 管理者およびプロジェクト管理者。

このセクションでは、CyberArk アプリケーションをアップロードします。

新しい **CyberArk** アプリケーションの追加

1. + 記号をクリックします。
   「CyberArk アプリケーション」 ダイアログ ボックスが表示されます。
2. 次のフィールドに入力して、資格情報を設定します。
   - [アプリケーション **ID**]: CyberArk アプリケーション リストで指定されている ID を入力します。
   - キー ストア ファイルのアップロード: ペーパー クリップ記号をクリックして、アップロードするキーストア ファイルを選択します。これは、CyberArk でアプリケーションを認証するために必要な証明書と秘密鍵のペアを含むキーストア ファイルです。
   - [パスワードの保存]: キーストアのパスワードを入力します。

データのセキュリティを最大限に確保するには、常に強力なパスワードを使用します。

- [秘密鍵のパスワード]: キーストアへのプライベート キー パスワードが存在する場合は、入力します。

3. [保存] をクリックします。

JMX

Management Console は、JMX プロトコルを介して少量の情報を提供します。JMX を介して表示される情報は試験的なものであり、パブリックな API ではありません。

Management Console は、次の MBean を公開します。

名前	説明
DistributionController	スケジュールの実行とクラスタ設定に関する情報。
FileBackedDataExporter	データ ビューから作成されたエクスポートに関する情報。
RobotLibraryGenerator	Management Console 内でロボット ライブラリの内部キャッシュを制御できるようにします。
TaskQueuePerformanceTracker	クラスタ化された 2 つの Management Console インスタンス間で分配されたタスク キューをプロファイリングできるようにします。

MBean は、JConsole と Java VisualVM (+MBean プラグイン) を介してアクセス可能です。これらは、JDK 1.6+ またはその他の JMX クライアントに含まれています。

OAuth

OAuth は、Twitter および Facebook など人気のある多くの API で使用されている認証のオープン標準です。これは、Kofax RPA が、ユーザーのログイン資格情報に直接アクセスすることなく、データにアクセスしたり、ユーザーの代わりにアクションを実行したりする場合に、アプリケーションとロボットにその手段を提供します。たとえば、ロボットは、@Kofax Twitter パスワードへの明示的なアクセス権がなくても、Twitter API を使用して @Kofax などのユーザーの最新の発言を、そのユーザーによって提供されたアクセス トークンを使用して抽出することができます。

Management Console は、キーストアに安全に格納されるアクセス トークンを生成するために使用します。アクセス トークンは、スケジュールにおけるロボットへの入力として渡すことができます。通常通り、実際の API 呼び出しを行い、返されたデータを処理するロボットは Design Studio に作成されます。

サポートされているサービス プロバイダ

OAuth で使用される用語として、サービス プロバイダは、Web サービスのプロバイダを意味します。Kofax RPA は、以下のサービス プロバイダーをサポートしています。各サービス プロバイダーの API の文書は、関連する Web サイトを参照してください。

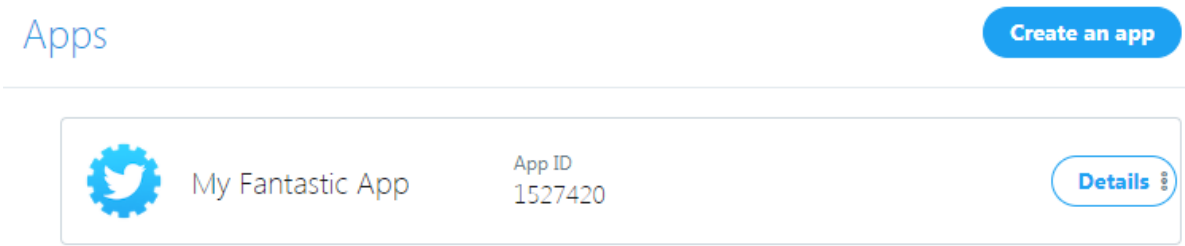
- Dropbox
  https://www.dropbox.com/developers
- Facebook
  https://developers.facebook.com
- Google
  https://console.developers.google.com/apis/library
- Kintone
  https://kintone.dev
- LinkedIn
  https://developer.linkedin.com
- Microsoft Azure AD 2.0
  https://docs.microsoft.com
- Salesforce
  https://developer.salesforce.com
- Twitter
  https://developer.twitter.com

アプリケーションの追加

サービス プロバイダーの API にアクセスするには、そのサービス プロバイダーでアプリケーションを作成する必要があります。アプリケーションを作成すると、コンシューマ キー (API キーまたはアプリケーション キーとも呼ばれます) およびコンシューマ シークレット (API シークレットまたはアプリケーション シークレットとも呼ばれます) が生成されます。

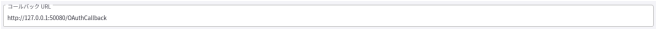
アプリケーションを作成するには、サービス プロバイダの開発者コミュニティにログインし、[新しいアプリケーションの作成] またはそれと同等のオプションを選択して必要な情報を入力します。開発者のサービス プロバイダーのドキュメンテーションへのリンクは、サポートされているサービス プロバイダを参照してください。このトピックでは、Twitter を使用してこれを実行する方法を説明します。

1. 最初に https://developer.twitter.com/en/apps/ にログインし、必要に応じてアカウントを作成してから、右上隅の **[アプリの作成]** をクリックします。



2. アプリケーションの名前および説明などの必要な情報を入力し、Twitter の利用規約を読んでから同意を選択します。

   フォームのフィールドの 1 つに、[コールバック **URL**] があります。これは、ユーザーが自身の代わりとしてアプリケーションが Twitter アカウントと対話することを承認した後に、Twitter がブラウザをリダイレクトする宛先の URL です。このフィールドは、Management Console が展開されたフォルダ下のパス「OAuthCallback」に設定する必要があります。たとえば、組み込み型の Management Console で実行している場合は、`http://localhost:50080/` で実行されます。この場合、コールバック URL は、`http://localhost:50080/OAuthCallback` に指定されます。ただし、一部のサービス プロバイダでは、`localhost` を含むコールバック URL が許可されていないことに注意してください。Twitter はこのようなプロバイダの 1 つであるため、代わりに `http://127.0.0.1:50080/OAuthCallback` を使用します。

   または、Management Console を実行しているマシンのホスト名または非ループバック IP アドレスを指定する必要があります (これは、一部のサービス プロバイダで必要となります)。このページは、認証ユーザーのブラウザでロードされるため、パブリックのホスト名または IP アドレスにすることはできません。

   > IP アドレスが 127.0.0.1 以外の場合、すべてのサービス プロバイダは HTTPS 接続を必要とします。また、この場合は、Management Console SSL を使用するように設定する必要があります。詳細については、SSL の使用に関する Tomcat のドキュメントを参照してください。

   アプリケーションを作成すると、アプリケーションの概要が表示されます。これらの値の一部を Management Console にコピーする必要があります。

3. ブラウザで Management Console を開きます。コールバック URL として入力したものと同じ IP アドレスまたはホスト名を使用します。

   以下の例では、ブラウザを http://127.0.0.1:50080/ に指定しています。

4. **[**リポジトリ**]** > **[OAuth]** に移動し、左上隅にある + 記号をクリックします。

   新しいアプリケーション ダイアログ ボックスが表示されます。

5. プロジェクトを選択し、次の情報を指定します。
   - **[**名前**]**: アプリケーションの名前を指定します。サービス プロバイダでアプリケーションを作成したときに使用した名前と同じである必要はありません。
   - **[**サービス プロバイダ**]**: サービス プロバイダを選択します。上記の例を実行するには、Twitter を選択します。
   - **[**コンシューマ キー**]** と **[**コンシューマ シークレット**]**: サービス プロバイダから提示されたアプリケーションの概要ページからこれらの値をコピーします。Twitter の例の手順については、ステップ 2 を参照してください。
   - **[**コールバック **URL]**: コールバック URL を指定します。Twitter の例の手順では、`http://127.0.0.1:50080/OAuthCallback` と入力します。
   - **[**スコープ**]**: 一部のサービス プロバイダでは、ユーザーがアプリケーションにアクセスを許可する API の部分など、スコープを指定する必要があります。たとえば、Google にアクセスする際、アプリケーションに Google Analytics Data API へのアクセスを許可する必要がある場合は、スコープとして `https://www.google.com/analytics/feeds/` を指定する必要があります。Twitter はスコープ フィールドを使用しないため、例の手順に従っている場合は、このフィールドを空白のままにします。
   - [コミット メッセージ]: コミットの説明を追加できます。

6. [**OK**] をクリックします。
   これで、Management Console での OAuth アプリケーションのセットアップが完了しました。

さらに、⋮ コンテキスト メニューからアプリケーションに対して次のアクションを実行できます。

- [編集]: **[**新しいアプリケーション**]** ダイアログ ボックスと同じフィールドが含まれます。
  セキュリティ上の理由から、コンシューマ シークレットは「(暗号化)」と表示されます。
- ユーザーの追加: アプリケーションにユーザーを追加するには、「ユーザーの追加」を参照してください。

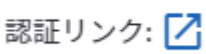
コンシューマ キーまたはコンシューマ シークレットを編集し、このアプリケーションに関連付けられている OAuth ユーザーがいる場合は、すべてのユーザーを削除する必要があります。この操作を行うには、**[**ユーザーを削除**]** ボタンをクリックします。このボタンは認証情報を編集した後に使用可能になります。次に、このアプリケーションに新しいユーザーを追加します。

アプリケーションを削除するには、テーブルでジョブを選択し、左上隅の 🗑 ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

ユーザーの追加

このトピックでは、OAuth アプリケーションのユーザーを追加する方法を示します。

1. **[**ユーザー**]** タブを選択し、左上隅にある + 記号をクリックします。
   複数のタブを持つダイアログ ボックスが表示されます。
   または、**[**アプリケーション**]** タブで希望するアプリケーションを選択し、⋮ コンテキスト メニューをクリックして、**[**ユーザーの追加**]** を選択します。
2. [アプリケーションとユーザー名を選択] タブで、アプリケーションを選択してユーザー名を指定します。
   このユーザー名は、サービス プロバイダが使用するユーザー名と一致している必要はありません。これは、Management Console の内部でのみ使用されます。
   - テナント。OAuth アプリケーションが Microsoft Azure AD 2.0 で作成されている場合、[テナント] という追加のパラメータも存在します。
     シングルテナント アプリケーションの場合、ユーザーを作成するにはテナント ID が必要です。Microsoft Azure ポータルに表示されるアプリケーションの **[**概要**] > [**要点**]** ページから [ディレクトリ **(**テナント**) ID**] をコピーします。
     マルチ テナント アプリケーションの場合は、必要なテナントの ID を指定するか、このパラメータを空のままにするかを選択できます。後者の場合、指定したクレデンシャルを持つテナントの ID がデフォルトで使用されます。
3. [次へ] をクリックします。
4. [承認] タブで、[承認リンク] をクリックします。

   これにより、サービス プロバイダの Web サイトが開きます。Twitter では、以下のように表示されます。

5. ユーザー名およびパスワードを入力して、[アプリの承認] をクリックします。
   サービス プロバイダーにより、コールバック URL にフォワーディングされます。承認に成功した場合は、OAuth 承認で続行ページが表示されます。
6. [ブラウザ] タブを閉じて、Management Console に戻ります。ウィザードで、[次へ] をクリックします。
   [アクセス トークンを取得] タブに、ユーザーの代わりにサービス プロバイダへアクセスするために使用可能なアクセス トークンが表示されます。これらのトークンは Management Console のキーストアに安全に保存されており、スケジュールに対する入力として使用できます。

   後のステップで構築するロボットのテスト入力値として、サンプル アクセス トークンが必要になります。Notepad などのテキスト エディターに値をコピーします。セキュリティ上の理由から、[終了] をクリックした後に非暗号化フォームのキーストアから値を取得することはできません。

   Twitter では、アクセス トークンおよびアクセス トークン シークレットを取得します。OAuth 2.0 を使用するサービス プロバイダーは、アクセス トークン シークレットを使用しないため、アクセス トークンのみを返します。一部のサービス プロバイダーは、更新トークンを追加で返します。これは、サービス プロバイダーによって返されたアクセス トークンが短期用途である場合のみ使用されます。その後、ユーザーが Management Console 経由で再認証を行わなくても、ロボットは更新トークンを使用して新しいアクセス トークンを取得できます。サービス プロバイダーの API に対してロボットを作成するには、ウィザードの最終ステップで表示されるトークンすべてをコピーする必要があります。
7. [終了] をクリックします。
   ユーザー エントリが [OAuth] セクションに表示されます。

ユーザーを削除

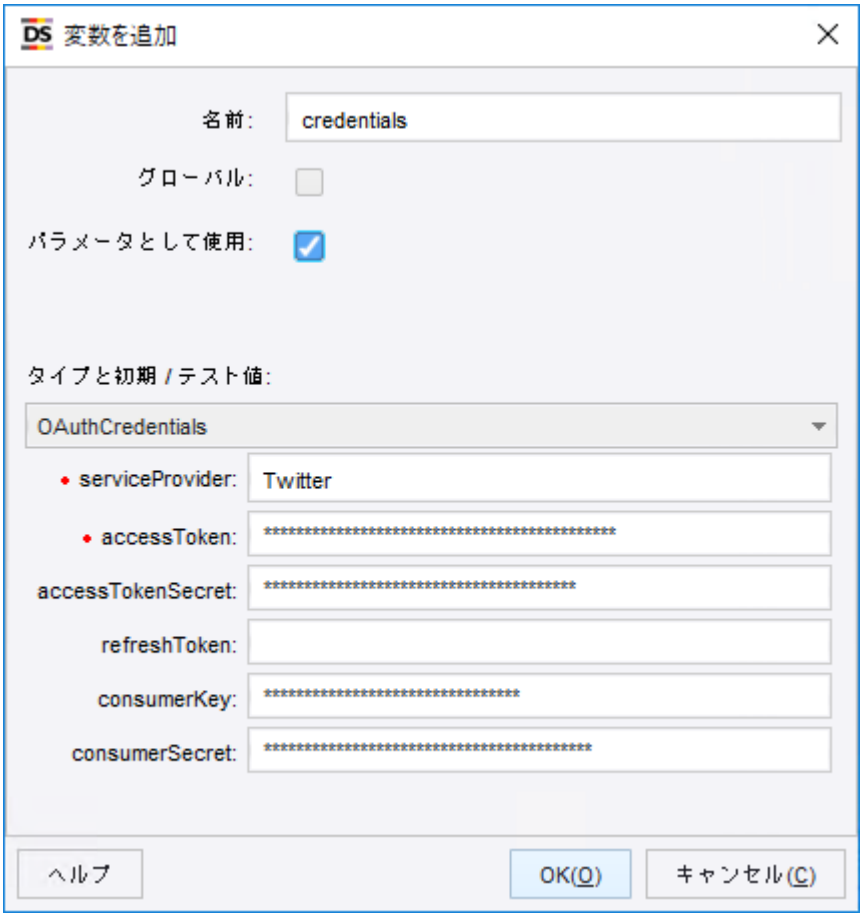
ユーザーを削除するには、テーブルでユーザーを選択し、左上隅の ごみ箱アイコンをクリックします。削除を確認するためのダイアログが表示されます。

コンシューマ キーまたはコンシューマ シークレットを編集する場合は、このアプリケーションに関連付けられているすべての OAuth ユーザーを削除する必要があります。詳細については アプリケーションの追加 を参照してください。

ロボットの書き込み

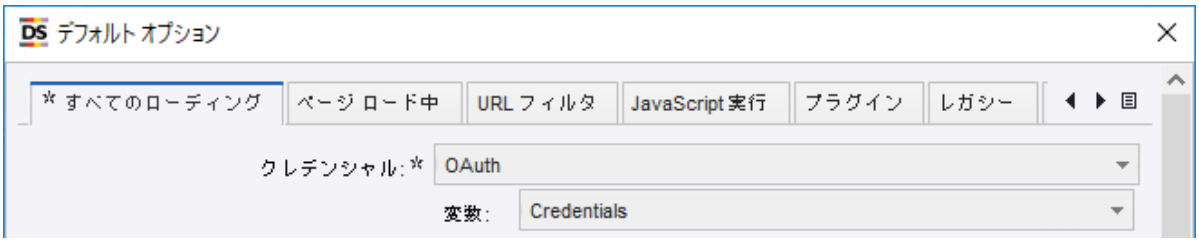
次の手順は、OAuth を認証メカニズムとして使用する REST API にアクセスするロボットを作成する方法を示しています。たとえば、Twitter REST API を使用して、ユーザーおよびユーザーがフォローしているユーザーを認証することで最新のステータスを取得します。

1. Design Studio を起動して、新しいベーシック エンジン ロボットを作成します。
   ウィザードには、URL を入力しないでください。認証するまでは、REST API にアクセスすることはできません。
2. ロボットに対して、コンプレックス タイプの新しい変数 OAuthCredentials を追加し、[パラメータとして使用] を選択します。
3. [**serviceProvider**] フィールドに、Twitter と入力します。
4. Management Console でユーザー認証プロセスを実行したときに取得したアクセス トークンとアクセス トークン シークレットを入力します。また、Twitter アプリケーションのコンシューマ キーおよびコンシューマ シークレットを入力します。



5. [**OK**] をクリックします。

6. [ロボット設定] をクリックします。
   [ロボットの設定] ウィンドウが表示されます。
7. [基本] タブで、[設定] をクリックします。
8. [すべてのローディング] タブで、[クレデンシャル] を検索し、[標準] のユーザー名/パスワード認証から [**OAuth**] に切り替えます。
9. 先ほど追加した入力変数を選択します。



   この操作により、ユーザーのタイムラインの最新のステータスを含む、返された XML が表示されるようになります。
10. 両方のダイアログで [**OK**] をクリックします。
    ロボットが OAuth を使用し、Design Studio での実行時に、指定した資格情報が使用されるようになりました。これで、Twitter API にアクセスできるようになります。たとえば、ユーザーが投稿した最新の各ツイートを表示するには、URL `https://api.twitter.com/1.1/statuses/user_timeline.json` にアクセスします。

資格情報を持つスケジュール ロボット

次の手順は、「ロボットの書き込み」で作成したロボットのスケジュールを設定する方法を示しています。

1. 「ロボットの作成」で作成したベーシック エンジン ロボットを Management Console にアップロードします。
2. ブラウザで Management Console を開き、新しいスケジュールを作成します。
3. [スケジュールされたジョブ] タブを選択します。
   [シングル ロボット] を選択し、ベーシック エンジン ロボットをスケジュールに追加して、[入力値を設定] ステップに達するまで [次へ] をクリックします。
4. このスケジュールでロボットを実行する際に使用する資格情報を持つ OAuth ユーザーを選択します。
   この操作により、ロボットが RoboServer で実行されているときに、サービス プロバイダ、アクセス トークン、コンシューマ キー、コンシューマ シークレットのフィールドが自動入力されます。
5. [次へ] をクリックし、[保存] をクリックしてスケジュールを保存します。
   これでロボットを実行する準備が整いました。

アウト オブ バンド アプリケーション

一部のサービス プロバイダーは、コールバックを使用せずに OAuth アプリケーションを認証するためのメカニズムを提供しています。これは、アウトオブバンドとして知られています。Management Console も、アウトオブバンド認証をサポートしています。サービス プロバイダで作成されたアプリ

ケーションは、帯域外アプリケーションとして登録する必要があります。たとえば、Twitter では、これはコールバック URL フィールドを空白のままにすることで行います。

1. Twitter アプリケーションを作成する場合は、コールバック URL のフィールドを空白のままにします。
2. Management Console にアプリケーションを登録する際には、ここでコールバック URL フィールドを空白のままにします。

   アプリケーションにユーザーを追加する際に、認証リンクをクリックしても、Management Console にリダイレクトされることはありません。代わりに、サービス プロバイダでは、検証機能または PIN が示されます。

3. [検証者] フィールドで、サービス プロバイダから提供される PIN を入力します。
4. [次へ] をクリックします。

   ユーザーが正常に認証されると、[アクセス トークンを取得] タブが開き、[アクセス トークン] および [アクセス トークン シークレット] フィールドに入力されます。[終了] をクリックします。

ユーザー API トークン

このユーザー メニュー アイテムを使用して、API アクセスを検証します。

新しいユーザー トークンを作成するには、**[トークンを作成]** をクリックし、トークン名を入力します。トークンは、作成ステップで 1 回だけ表示できます。

トークン名のリストは Management Console で入手できますが、トークンはセキュリティ上の理由から非表示になっています。

「RPA」は予約済みの名前です。そのため、トークンにこの名前を使用することはできません。

トークンをコピーし、API アプリケーションに挿入します。

トークンを削除するには、**[削除]** をクリックします。

⚠ API を使用し、Robot File System、Document Transformation Service、またはパスワード ストアを使用するロボットがあるアプリケーションを更新して、ユーザー名とパスワードの代わりにユーザー API トークンを要求に含めることをお勧めします。

フィルタリング

Management Console の多くのセクションに含まれる [フィルタ] フィールドは、項目のリストをフィルタリングして、名前とパスを基準に表示する場合に役立ちます。[ロボット] および [トリガー マッピング] セクションには、ワイルドカードを使用しないフィルタリング テキスト フィールドが含まれることに注意してください。たとえば、ロボットのリストをタグでフィルタリングするには、[タグ] テキスト ボックスにタグ名または名前の一部を入力します。**[管理]** > **[RoboServer]** などの一部のセクションでリストをフィルタリングするには、まず [フィルタ] をクリックして表示する項目を選択し、次にその名前を入力します。

[フィルタ] フィールドを使用してリストをフィルタリングするときは、次のルールに注意してください。

- デフォルトでは、すべてのフィルタで大文字と小文字が区別されません。[フィルタ] フィールドに文字を入力するときに、大文字と小文字は区別されません。
- ? は 1 文字に、* は 0 以上の文字にマッチします。
- スラッシュなし、ワイルドカード文字なしで検索文字列をタイプすると、項目の名前にマッチする部分文字列として検索を行います。たとえば、検索文字列 "foo" は、入っているフォルダに関係なく、ロボット `foo` および `foobar` にマッチすることになります。
- スラッシュなし、1 つまたは複数のワイルドカード文字付きで検索文字列をタイプすると、完全な項目の名前にマッチするパターンとして検索を行います。たとえば、"foo*" の場合は、入っているフォルダに関係なく、ロボット `foo` および `foobar` にマッチしますが、`xxxfoo` にはマッチしません。検索文字列 "foo?ar" は、ロボット `foobar` にマッチします。
- スラッシュを含む検索文字列をタイプすると、項目のフルパス、つまりフォルダおよび名前にマッチするように検索を行います。たとえば、検索文字列 "sub1/foo" の場合、`sub1/foo` にマッチしますが、`sub1/foobar` にはマッチしません。検索文字列 "sub1/foo*" の場合、`sub1/foo` および `sub1/foobar` にマッチします。

ユーザー パスワードの暗号化

Management Console には、ユーザー パスワードを使用および保存する「bcrypt」パスワード ハッシュ関数が実装されています。この関数は、10 回のログ ラウンドを使用するように設定されたデフォルトの「bcrypt」メカニズムに基づいているため、パスワードを 2^10 回ハッシュします。

シンクロナイザー

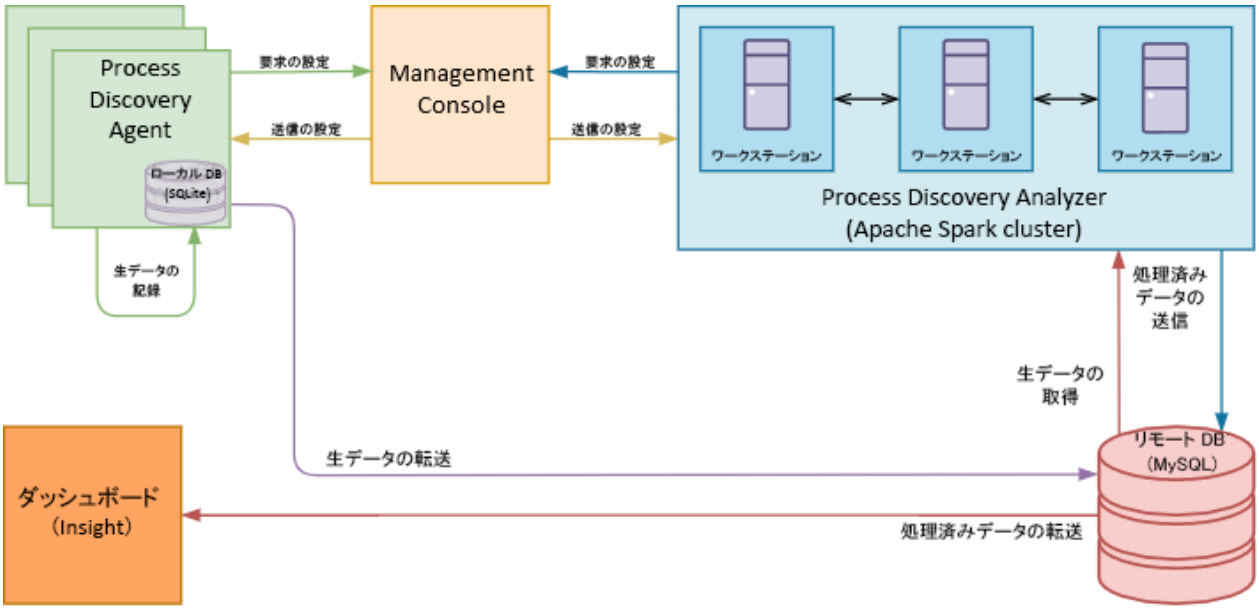
シンクロナイザーを使用して、Management Console リポジトリをファイルベースの Git リポジトリに接続します。

シンクロナイザーを起動するには、Kofax RPA インストール フォルダの `/bin` サブフォルダにある `Synchronizer.exe` をダブルクリックします。シンクロナイザーの認証と設定の詳細については、『Kofax RPA のヘルプ』の「サービス認証」と『Kofax RPA ベスト プラクティス ガイド』の「同期の開始」を参照してください。

第 7 章

Process Discovery

Kofax RPAProcess Discovery は、ユーザーのアクティビティとプロセスのボトルネックを監視するために必要なデータを収集し、本番環境での Kofax RPA の使用に関する意思決定を行う場合に役立ちます。



Process Discovery コンポーネント

Process Discovery は Windows UI オートメーション API に依存しています。UI オートメーション API クライアントは、同じコンピュータ上で Process Discovery Agent と同時に実行しないでください。

Kofax Analytics for RPA ダッシュボードの Process Discovery ビューには、Process Discovery Agents によって収集されたデータに基づく情報を含んだレポートが表示されます。このビューには、データに関するさまざまな視覚的および分析的表現がチャートやテーブルとして表示されます。

製品には、Kofax RPA Process Discovery コンポーネントを正常にインストールするために必要なファイルが含まれます。次のテーブルをご覧ください。

コンポーネント	必要なインストール ファイル
Linux • .deb パッケージ • Analyzer をインストールして実行するための Dockerfile **Windows** Process Discovery Analyzer .msi インストーラ	`KofaxRPAProcessDiscoveryAnalyzer-11.5.0.0.[ビルド番号].zip` Process Discovery Analyzer は、記録された生データを処理し、Kofax Analytics for RPA ダッシュボード ビューの絞り込みデータを生成するためのものです。
Process Discovery Agent	`KofaxRPAProcessDiscoveryAgent-11.5.0.0.[ビルド番号].msi` Process Discovery Agents は、コンピュータにインストールして、指定したアプリケーションのユーザー アクティビティ情報の収集や、データベースへのデータの格納を実行します。

Kofax Analytics for RPA	• 次のような Kofax Insight インストール ファイル:
	`KofaxInsightSetup_6.5.0.0.0.<build>_x64.msi`
	• 次のような Kofax Analytics for RPA プロジェクトの `zip` ファイル。
	`KofaxAnalyticsforRPA-<バージョン>.zip`

Process Discovery を使用するには、次のインストール手順および設定手順を実行します。

1. 環境に応じて、次のいずれかを実行します。
   - Linux での Docker を使用した Analyzer のデプロイ
   - Windows への Process Discovery Analyzer のデプロイ
   - Linux への Process Discovery Analyzer のデプロイ
2. Process Discovery Agents 。
3. Analytics 。
4. データベース 。

Process Discovery 用語集

- イベント: Agent によってユーザーのデスクトップに記録されたアクション。たとえば、Acrobat で PDF を左クリックする、Excel のコンテキスト メニューからオプションを選択する、Enter キーを押す、などのアクションが含まれます。
- シーケンス: 1 人のユーザーが実行した一連のイベント (時間順に並べ替えられます)。
- タスク: 特定の目標を達成するためのシーケンス。ユーザーによって、タスクを構成するシーケンスの開始と終了が示されていることがあります。タスクはエージェントで使用される用語で、プロセス インスタンスと同様の働きをします。
- ステップ: 複数のタスクを達成するために実行される個別のアクション。各ステップは、確認されたイベントに基づいて Analyzer によって検出されます。これらのイベントは検出されたステップに関連付けられます。
- プロセス: 複数のタスクを達成する、有向グラフに並べられた一連のステップ。各プロセスは、確認されたシーケンスに基づいて Analyzer によって検出されます。プロセスとタスクが一致する場合もあれば、一致しない場合もあります。

- プロセス インスタンス: プロセスに適合する、確認されたシーケンス。プロセス インスタンスはタスクと同様の働きをし、この用語は Analyzer、および RPA ダッシュボードの Kofax Analytics の Process Discovery ビューで使用されます。

Process Discovery Agents

Process Discovery Agents は、コンピュータにインストールして、指定したアプリケーションのユーザー アクティビティ情報の収集や、データベースへのデータの格納を実行します。

データ収集が中断されないようにするために、Agent プログラムをウイルス対策ソフトウェアの例外リストに含めます。

Management Console で、Agent の割り当て先となる、コンピュータを論理的に結合するグループを 1 つ以上作成します (`financial_dept`、`hr_dept`、`sales_dept` など)。Process Discovery Agents をデプロイするときに、Management Console で作成したグループの名前を指定します。

Agent をインストールする前にグループを作成するには、次の手順を実行します。

1. Management Console の左側のペインで [設定] を展開し、[**Process Discovery** グループ] をクリックします。
2. [新しいグループを作成] タブでグループ名を入力し、他の設定を指定して、[保存] をクリックします。詳細については、Process Discovery グループを参照してください。
   このアクションにより、新しいグループが作成され、新しく作成したグループ設定を含むタブが開きます。
3. [**Agent** データベースのプロビジョニング] セクションで [スキーマの作成] と [ユーザーにアクセス権を付与] をクリックして、データベース プロビジョニングを実行します。

Agent を手動でインストールするには、`KofaxRPAProcessDiscoveryAgent-11.5.0.0.[ビルド番号].msi` を実行して、ウィザードに示される手順を実行します。

Process Discovery Agent の以前のバージョンは、新しいバージョンの Agent をインストールすると削除されます。

インストール後に初めて Agent を実行すると、Process Discovery Agent 設定ウィンドウが開きます。Management Console の URL、Process Discovery のグループ名、クレデンシャルなどのパラメータを入力して、Management Console にログインします。[記録の開始] をクリックします。

スクリプト ファイルを使用してインストール プロセスを自動化するために、Agent のサイレント インストールを実行することもできます。

Process Discovery Agents の構成

Agent 設定を編集するには、通知領域で Process Discovery Agent アイコン を右クリックして、[設定] を選択します。ヘルプ トピックを開くには、通知領域で Process Discovery Agent アイコン を右クリックして、[ヘルプ] をクリックします。

Process Discovery Agent のサイレント インストール

スクリプト ファイルを使用してインストール プロセスを自動化するために、Agent のサイレント インストールを実行できます。デフォルトの場所に Kofax RPA Process Discovery Agent のサイレント インストールを実行するには、管理者権限で次のコマンドを実行します。

```
msiexec /qn /I KofaxRPAProcessDiscoveryAgent-11.5.0.0.[ビルド番号].msi
MCURL="mc_url" MCUSER="username" MCPW="password" GROUP="group"
```

ここで、"mc_url"、"username"、および "password" は、Management Console に接続するための URL、ユーザー名、およびパスワードです。また、"group" は、**[管理]** > **[設定]** の [**Process Discovery** グループ] セクションの Management Console に作成されたグループ名です。

Agent をインストールする場所を指定するには、次のコマンドを実行します。

```
msiexec /qn /i KofaxRPAProcessDiscoveryAgent-11.5.0.0.[ビルド番号].msi
MCURL="mc_url" MCUSER="username" MCPW="password" GROUP="group"
INSTALLDIR="dir"
```

ここで、"dir" は Agent をインストールするパスです。

インストール プロセスをログに記録するには、ログ ファイルを指定します。

```
msiexec /qn /i KofaxRPAProcessDiscoveryAgent-11.5.0.0.[ビルド番号].msi /l
RPAmsilog.txt
```

インストール後に Agent を起動するには、次のコマンドを指定します。

```
"start /d [インストール ディレクトリ] KofaxRPAProcessDiscoveryAgent.exe"
```

ここで、[インストール ディレクトリ] は、プログラムをインストールするディレクトリです。インストール時に INSTALLDIR オプションを指定していた場合は、その値を [インストール ディレクトリ] に使用します。

Agent 設定を編集する場合、通知領域で Process Discovery Agent アイコン を右クリックして、[設定] を選択します。ヘルプ トピックを開くには、通知領域にある Process Discovery Agent アイコン を右クリックして、[ヘルプ] をクリックします。

Process Discovery Agent の構成

Process Discovery Agent はコンピュータにインストールされ、指定したアプリケーションのユーザー アクティビティ情報を収集したり、データをデータベースに格納したりします。

スクリーン リーダーやウイルス対策ソフトウェアなどの一部の支援テクノロジ アプリケーションは、Process Discovery Agent を検出して、警告を生成するか、Agent の実行を阻止することがあります。Agent の実行を許可するには、Process Discovery Agent を安全なアプリケーションとして承認するか、警告が表示されたときに [今後このダイアログボックスを表示しない] オプションを選択します。

設定ウィンドウ

接続設定を編集する場合、通知領域で Process Discovery Agent アイコン を右クリックして、[設定] を選択します。開かれる [**Kofax RPA Process Discovery Agent**] ウィンドウには、Management Console 設定とステータス メッセージが含まれます。このウィンドウには、次のボタンも含まれます:

- 終了: Agent を停止してプログラムを完全に終了します。
- ヘルプ: ヘルプの「Process Discovery Agent の設定」トピックを開きます。
- 製品について: Agent のバージョン情報およびログ ファイルの情報が記載されたウィンドウを開きます。
- 記録の開始と記録の停止: ユーザー アクションの記録を開始または停止します。

Management Console 接続設定
Management Console に接続するための接続設定が含まれます。

- [**URL**]: 接続する Management Console の URL とポート (形式は次の例のとおり):
  - 組み込み Management Console: `http://localhost:50080`
  - Tomcat 上の Management Console: `http://192.168.0.101:8080/ManagementConsole`
- [グループ]: Process Discovery グループ名 (Management Console で指定)。
- [ユーザー]: ログイン ユーザー名。
- [パスワード]: ログイン パスワード。

ステータス
次の表には、Agent 記録と接続ステータスが一覧表示されています。

コンポーネント	ステータス
エージェント	開始されていません: Agent は実行していません。 記録を開始しています: Agent は開始しており、すべての必要な接続が現在確立中です。 ローカル ストレージに記録中: Agent は実行中ですが、Management Console またはデータベースとの接続を確立できません。Agent は収集された情報を実行中のコンピュータに保存しています。Agent は定期的に再接続を試みます。 リモート ストレージに記録中: Agent が実行中で、すべての接続は確立され、収集されたデータは Agent に保存されています。 記録を無効化: Agent は実行中ですが、ユーザー アクションは記録されません。記録は Management Console 内のこの記録モード設定で無効化されています。 失敗: Agent は最後の試行でユーザー アクションを記録できませんでした。

コンポーネント	ステータス
Management Console	接続されていません: Agent は Management Console に接続されていません。以下の原因が考えられます: • 記録が開始されていない。 • Management Console URL が指定されていない。 • Process Discovery グループが指定されていない。 接続中: Management Console への接続は現在確立中です。 接続済み: Agent は Management Console に接続されています。 接続に失敗しました: Agent は最後の試行で Management Console に接続できませんでした。以下の原因が考えられます: • 指定されたホストに接続できません。Management Console サーバが稼働していることを確認します。 • 指定されたポートが開いていないか、到達できません。Management Console が指定されたポートで実行していることを確認します。 • ユーザー名かパスワードが無効です。Management Console のユーザー名とパスワードを正しく指定します。 • Management Console がリソース パスをサポートしていません。正しく指定されていることを確認します。 Agent は定期的に再接続を試みます。 誤設定: このメッセージが表示される場合は、Management Console の Process Discovery グループ設定が正しいかを確認します。Agent は、定期的に Management Console に接続して設定を適用しようと試みます。 **Management Console** のバージョンが **Agent** のバージョンと一致しません: Agent は現在のバージョンの Management Console とともに機能することができません。Management Console と Agent のバージョンを一致させる必要があります。Agent は定期的に再接続を試みます。
データベース サーバー	接続されていません: Agent がデータベースに接続されていません。Management Console への接続が確立されていません。 接続中: データベースへの接続は現在確立中です。 接続済み: Agent はデータベースに接続されています。 接続に失敗しました: Agent は最後の試行でデータベースに接続できませんでした。以下の原因が考えられます: • データベースのアドレスが Management Console のProcess Discovery グループの設定で不正に指定されています。 • データベースに未定義のスキーマがあります。再作成するか、Management Console の設定を変更します。 • データベース スキーマが以前のバージョンの Agent 用です。スキーマを再作成するか、Management Console の設定を変更します。 Agent は定期的にデータベースの再接続を試みます。

記録の開始および停止

記録を開始または停止するには、[記録の開始] か [記録の停止] をそれぞれクリックします。

Management Console 設定が設定されている場合、[記録の開始] をクリックして Agent を開始します。Agent は、最初に必ず Management Console に接続してグループ設定の取得を試みます。次

に、Agent はデータベースに接続します。接続が確立されると Agent は記録を開始し、数秒後にウィンドウが最小化されシステム トレイに表示されます。これ以外では、Agent は以下を実行します:

- Management Console 設定が設定されていない場合、Agent はローカルで記録を開始し、すべての収集されたデータを保存します。
- Management Console 設定が定義されているものの、Agent が Management Console に接続できなかった場合、Agent はローカルで記録を開始しすべての収集されたデータを保存しながら、接続が確立されるまで定期的に再接続を試みます。
- Agent がデータベースに接続できない場合、すべての収集されたデータをローカルに保存し、定期的に再接続を試みます。データベースへの接続が確立されると、収集されたデータはデータベースに移動されます。

記録中、Management Console またはリモート データベースへの接続が失われた場合、Agent は接続が確立されるまで再接続を試みます。

Agent の終了
Agent を停止し、プログラムを完全に終了するには、[終了] をクリックします。

設定ウィンドウの [閉じる] ボタンをクリックすると、ウィンドウは最小化されシステム トレイに表示されます。

バージョン情報ウィンドウ

このウィンドウは、Agent のバージョン情報およびログ ファイルの情報を表示します。

[**Kofax RPA Process Discovery Agent** について] ウィンドウを開くには、通知領域で Process Discovery Agent アイコン を右クリックして [製品について] を選択するか、設定ウィンドウの [製品について] をクリックします。

Agent のバージョン
Agent のバージョンは、[**Kofax RPA Process Discovery Agent** について] ウィンドウのバージョン ラベルの横に表示されます。

ログを開く

Agent のログを開きます。

クラッシュ ダンプ ディレクトリを開く
ディスク上のメモリ情報ファイルがあるディレクトリを開きます。このダンプ ファイルは、開発者がクラッシュの原因をデバッグする際に役立ちます。

ドキュメントを開く
Kofax RPA ヘルプで Agent のドキュメントを開きます。

タスクの開始

Agent のショートカット メニューの [タスクの開始] オプションは、フォームの入力、顧客の要求への対応、レポート準備など、従業員が実行するタスクの開始を示します。タスクが終了したら、ユーザーは [タスクの停止] をクリックする必要があります。タスクのパフォーマンスは、RPA ダッシュボードの Kofax Analytics の Process Discovery ビューに表示されます。このオプションを使用すると、ビューでプロセスの検出タイプに [手動] とマークされます。タスクの開始および停止によって、作業の完了まで実行されたアクションをより適切に解析できます。

たとえば、従業員が実行するプロセスが Process Discovery ビューに表示されない場合、またはプロセスが複数に分割されて表示される場合は、この機能を使用して、アナライザーがこのプロセスを見つけられるようにすることができます。複数のユーザーが同様のタスクを実行する場合は、プロセスごとにこの機能 1 回だけ使用します。ユーザーが [タスクの開始] オプションをクリックした後に、より適切な結果を得るためには、タスクの実行時に中断などの「ノイズ」アクションを発生させないようにする必要があります。電話や電子メール メッセージなど、他のタスクによって作業が中断された場合は、タスクをキャンセルして最初からやり直してください。

タスクのキャンセル

開始されたタスクをキャンセルするには、Agent のショートカット メニューの [タスクのキャンセル] オプションをクリックします。

ヘルプを開く

Agent のドキュメントを開くには、通知領域で Process Discovery Agent アイコン を右クリックし、[ヘルプ] を選択するか、Agent の設定ウィンドウで [ヘルプ] をクリックします。または、[**Kofax RPA Process Discovery Agent** について] ウィンドウで [ドキュメントを開く] をクリックします。

トラブルシューティング

次の手順は、Agent の実行時に発生する一般的な問題を解決する場合に役立ちます。

Agent の実行中に、一般的なコンピュータまたはユーザーのコンピュータ上の一部のプログラムの応答が遅くなる場合は、次の手順を実行します。

1. Management Console を開き、左側のペインの [設定] で [**Process Discovery** グループ] をクリックします。
2. コンピュータが属するグループのタブを開きます。
3. [記録の詳細設定] を見つけて、[最小 **UIA** ツリー不変期間 **(ms)**] でタイムアウト期間を長くします。コンピュータが応答するようになるまで、50 ms 刻みで数値を大きくします。

Agent の実行中にユーザーのコンピュータ上の特定のプログラムの応答が遅くなった場合は、次のいずれかを実行します。

- プログラムがビジネス プロセスの一部ではなく、Agent が無視できる場合は、Management Console でコンピュータが属する Process Discovery グループを開き、[記録設定] の [アプリケーション無視リスト] にプログラムの実行可能ファイルの名前を入力します。
- プログラムがビジネス プロセスの一部であり、Agent がそのプログラムを記録する必要がある場合は、それぞれのグループ タブの [記録の詳細設定] の下にある [非 **UIA** アプリケーション] リストにプログラム名を含めるか、プログラムが応答するようになるまで、上記の全般的に応答が遅いコンピュータに対する手順を実行します。これらの変更によって、プロセスの分析結果が変わる可能性があることに注意してください。

ビジネス プロセス内で Microsoft Excel を使用する場合は、入手可能な最新バージョンを使用することで、より効率的にプロセスを検出できます。

Process Discovery Analyzer

Process Discovery Analyzer は、記録された生データを処理し、Kofax Analytics for RPA ダッシュボード ビューの絞り込みデータを生成するためのものです。

Analyzer は、ステータスをコンソール ウィンドウおよび Analyzer ログ ファイルに出力するコマンド ライン アプリケーションです。

分析が中断されないようにするために、Analyzer プログラムをウイルス対策ソフトウェアの例外リストに含めます。

Management Console から設定を取得

起動中、Process Discovery Analyzer は Management Console に接続し、必要な設定と Process Discovery グループ に関する情報を受信して、解析を開始します。

Management Console URL と認証設定をセットアップして、Management Console に接続します。Process Discovery Analyzer オプション セクションでは、Analyzer の展開方法に応じてオプションを設定する方法について説明します。

データの解析

Process Discovery Analyzer は、Process Discovery グループの Agent が収集したすべての記録されたデータを処理します。すべての Process Discovery グループは個別に解析を実行します。分析設定は、Management Console の Process Discovery グループ メニューで指定します。

Management Console の接続が確立された後、Analyzer はデータベースへの接続を確認します。データベースへの接続が問題なく確立された場合、Analyzer は解析を開始します。

Analyzer が Management Console または Analyzer データベースに接続できない場合、解析は開始されず、エラーが報告されます。その理由の 1 つとして、Analyzer と Management Console のバージョンが異なることが考えられます。

Management Console のグループに対して [分析] オプションがオンになっていない場合、または Process Discovery グループ データベースに Analyzer を接続できない場合は、分析からグループが省略されます。

すべてのグループのデータ分析が終了すると、Analyzer はスタンバイ モードになり、Management Console の設定で指定したスケジュールに沿って次の実行を待機します。

Process Discovery Agents がデータを収集するときに、毎日 Process Discovery Analyzer を実行することをお勧めします。一日の終わりに Analyzer を実行して、毎朝新しいデータを取得するようにスケジュールできます。分析には時間がかかります。データ量によって数時間から数日間かかることがあります。データの処理時間を短縮するには、Analyzer クラスタのノード数を増やします。詳細については、Process Discovery Analyzer クラスタ を参照してください。

ログ ファイル

Analyzer とクラスタ ログ ファイルの場所は、オペレーティング システムと実行モードに応じて異なります。次のログ ファイルが作成されます。

- `analyzer.log`: 一般的な Process Discovery Analyzer のログ ファイル。
- `spark_worker.log`: Analyzer クラスタ ワーカーのログ ファイル。
- `spark_master.log`: Analyzer クラスタ マスターのログ ファイル (マスター ノードでのみ作成)。

Windows アプリケーション
Analyzer が Windows アプリケーションとしてインストールされている場合、ログ ファイルは Analyzer Application Data フォルダ内にあります。例：

`C:\Users\UserName\AppData\Local\KofaxRPAProcessDiscoveryAnalyzer\[バージョン]\Logs`

Windows サービス
Analyzer が Windows サービスとしてインストールされている場合、ログ ファイルは Program Data フォルダ内にあります。例：

`C:\ProgramData\KofaxRPAProcessDiscoveryAnalyzer\<version>\Logs`

Linux アプリケーション
Analyzer が Linux にインストールされている場合、ログ ファイルは次の場所にあります。

`/home/KofaxRPAProcessDiscoveryAnalyzer/[バージョン]/Logs`

Process Discovery Analyzer オプション

Docker を使用しない **Process Discovery Analyzer** の展開
Analyzer のオプションと説明のリストを表示するには、以下のようにコンソール ウィンドウで `KofaxRPAProcessDiscoveryAnalyzer.exe` (Windows プラットフォーム) または `KofaxRPAProcessDiscoveryAnalyzer` (Linux プラットフォーム) を `-h` または `--help` パラメータとともに実行します:

```
KofaxRPAProcessDiscoveryAnalyzer.exe --help
```

Process Discovery Analyzer オプション

オプション	説明
`-h, --help`	ヘルプ メッセージを表示して終了します。
`--version`	Analyzer のバージョンを表示します。
`--mc-url`	次の例のように接続する Management Console URL とポート: • 組み込み Management Console: `http://localhost:50080` • Tomcat 上の Management Console: `http://192.168.0.101:8080/ManagementConsole`
`--mc-user`	Management Console のユーザー名。
`--mc-password`	Management Console のパスワード。
`--locale`	ロケール (「ja」など) を明示的に指定します。デフォルトでは、ロケールはシステム設定から取得されます。
`--log`	ロギング レベルの設定。使用する値は、`CRITICAL`、`ERROR`、`WARNING`、`INFO`、`DEBUG` です。
`--open-doc`	Analyzer ドキュメントを開きます。
`--dump`	このパラメータが指定されている場合、Analyzer はデバッグのために一部の生データを一時ディレクトリに記録します。

Process Discovery Analyzer により前回実行時の設定が保存されます。次回にオプションを指定しないで Analyzer を起動した場合は、前回の実行時に保存された設定が使用されます。
Windows アプリケーションおよび Windows サービスの設定は個別に保存されます。

Analyzer のデフォルトの設定

```
--mc-url=localhost:50080
--log=info
--locale=<system settings>
```

Linux での Docker を使用した Analyzer のデプロイ

この手順では、Docker を使用して Linux ベースのシステムのスタンドアロン サーバーに Process Discovery Analyzer をデプロイするための基本的な手順を説明します。

1. MySQL サーバーを起動します。
   MySQL サーバーは、Docker を使用してデプロイすることもできます。公式の Docker Hub Web サイトで最新の MySQL サーバー イメージを入手することができます。
2. Management Console を起動し、Process Discovery Analyzer のパラメータを指定します。次のフィールドで、デフォルト設定が使用可能な場合はその設定を使用します。
   a. Management Console で、[設定] メニューを展開し、[**Process Discovery Analyzer**] をクリックします。
   b. Process Discovery Analyzer データベース接続設定の指定[3]ランタイム パラメータ、クラスタ設定、データベースのルート パスワードを指定したりすることはできません。
   c. [データベースのプロビジョニング] タブで [スキーマの作成] と [ユーザーにアクセス権を付与] をクリックして、データベース プロビジョニングを実行します。詳細については、Process Discovery Analyzerを参照してください。
3. `KofaxRPAProcessDiscoveryAnalyzer-11.5.0.0.[ビルド番号].zip` をダウンロードして、すべてのファイルをアーカイブからコンピュータ上のフォルダに抽出します。
4. https://www.docker.com から Docker をダウンロードし、コンピューターにインストールします。
5. 現在のユーザーを Docker グループに追加して、すべての Docker コマンドで `sudo` を入力するように求められることを回避できます。たとえば、「kofax」ユーザーを追加するには、`/etc/group` ファイルで `docker:x:<n>` を `docker:x:<n>:kofax` に置き換えます。ログアウトしてから再度ログインするか、再起動して、権限を更新します。
6. アーカイブから Dockerfile のあるフォルダに移動します。
7. 次のコマンドを実行して、Analyzer Docker イメージを構築します。

   ```
   docker build --tag process_discovery_analyzer_11.5.0.0 .
   ```
8. コンテナーの環境変数を指定して、Analyzer を起動します。Analyzer は、次のデフォルト設定で起動します。

   ```
   MC_URL=localhost:50080
   LOG=info
   ```

[3] これは、MySQL サーバーを実行しているコンピュータ、または MySQL サーバーを使用する Docker コンテナの IP アドレスまたはドメイン名です。アドレスとして `localhost` または `127.0.0.1` を指定したり、

```
LOCALE=<system settings>
```

次の環境変数を設定できます。

- MC_URL: Management Console の URL
- MC_USER: ユーザー名
- MC_PASSWORD: パスワード
- LOG: ロギング レベルの設定。使用する値は、CRITICAL、ERROR、WARNING、INFO、DEBUG です。
- ロケール: Analyzer の優先言語を設定します。
- ダンプ: 一部の中間データを一時フォルダにダンプします。

Analyzer を起動するには、次のように docker run コマンドを使用して、コンテナの起動時に環境変数を指定します。

```
docker run -it [-e <ENV VAR>=<VALUE>] process_discovery_analyzer_11.5.0.0
```

環境変数 DUMP を指定するには、次のパターンを使用します。

```
-e DUMP=True
```

Analyzer のオプションとデフォルト設定のリストについては、Process Discovery Analyzer オプションを参照してください。

Analyzer のコンテナ ID を取得するには、次のコマンドを実行します。

```
docker ps
```

Analyzer の実行を停止するには、次のコマンドを使用します。

```
docker stop <container-id>
```

Analyzer に新しい設定を適用するには、まずコンテナを停止してから、新しいパラメータでコンテナを再起動します。

Windows への Process Discovery Analyzer のデプロイ

以下の情報を使用して、Docker を使用せずに Windows で Process Discovery Analyzer を実行します。

Process Discovery Analyzer をデプロイする前に、Process Discovery Analyzer を実行しているコンピュータに Java ランタイム環境がインストールされており、JRE インストール フォルダを指定する JAVA_HOME 変数がシステム変数に追加されていることを確認してください。

Docker を使用せずに Process Discovery Analyzer を Windows にデプロイするには、実行中の MySQL サーバーが必要です。

1. MySQL サーバーを起動します。
2. Management Console を起動して開き、Process Discovery Analyzer パラメータを指定します。次のフィールドで、デフォルト設定が使用可能な場合はその設定を使用します。
   a. Management Console で、[設定] メニューを展開し、[**Process Discovery Analyzer**] をクリックします。
   b. Process Discovery Analyzer データベース接続設定の指定[4]ランタイム パラメータ、クラスタ設定、データベースのルート パスワードを指定したりすることはできません。

[4] これは、MySQL サーバーを実行しているコンピュータの IP アドレスまたはドメイン名です。アドレスとして localhost または 127.0.0.1 を指定したり、

c. [データベースのプロビジョニング] タブで [スキーマの作成] と [ユーザーにアクセス権を付与] をクリックして、データベース プロビジョニングを実行します。詳細については、Process Discovery Analyzerを参照してください。

3. KofaxRPAProcessDiscoveryAnalyzer-11.5.0.0.[ビルド番号].zip をダウンロードして、アーカイブをコンピュータ上のフォルダに抽出します。
4. 抽出されたファイルに移動して、KofaxRPAProcessDiscoveryAnalyzer-11.5.0.0.[ビルド番号].msi を実行します。
   - デフォルトでは、Process Discovery Analyzer は Windows アプリケーションとして C:\Program Files (x86)\Kofax RPA Process Discovery Analyzer 11.5.0.0.[ビルド番号]\ にインストールされます。インストール パスは、必要に応じて変更できます。

     新しいバージョンの Analyzer をインストールすると、以前のバージョンの Process Discovery Analyzer は削除されます。

     インストールした後に、KofaxRPAProcessDiscoveryAnalyzer.exe に移動して Analyzer を実行します。デフォルト以外のオプションを指定するには、コマンド プロンプト ウィンドウでオプションを指定して Analyzer を実行します。詳細については、Process Discovery Analyzer を参照してください。
   - Process Discovery Analyzer を Windows サービスとしてインストールする場合は、インストーラの [インストール先フォルダ] のステップで [**Windows** サービスとして登録] を選択します。

     Windows の起動時に Process Discovery Analyzer を起動するには、[自動起動] を選択します。

     インストールが完了すると、Analyzer が自動的に起動します。サービスにデフォルト以外のオプションを指定するには、次の手順を実行します。

     a. **Control Panel** > **Administrative Tools** > **Services** の順に移動します。
     b. [**Kofax RPA Process Discovery Analyzer**] サービスを右クリックし、[停止] を選択してサービスを停止します。
     c. [**Kofax RPA Process Discovery Analyzer**] サービスを右クリックし、[プロパティ] を選択します。
     d. [スタート パラメータ] フィールドにパラメータを指定し、[スタート] をクリックしてサービスを開始します。

     Process Discovery Analyzer を Windows アプリケーションとして起動するには、まず [**Kofax RPA Process Discovery Analyzer**] サービスを停止してから、KofaxRPAProcessDiscoveryAnalyzer.exe に移動して、上記の手順に従って実行します。

Analyzer のデフォルトの設定

```
MC_URL=localhost:50080
LOG=INFO
LOCALE=<system settings>
```

Analyzer のオプションとデフォルト設定のリストについては、Process Discovery Analyzer オプションを参照してください。

Linux への Process Discovery Analyzer のデプロイ

以下の情報を使用して、Docker を使用せずに Linux で Process Discovery Analyzer を実行します。

Process Discovery Analyzer をデプロイする前に、Process Discovery Analyzer を実行しているコンピューターに Java ランタイム環境がインストールされており、JRE インストール フォルダを指定する JAVA_HOME 変数がシステム変数に存在していることを確認してください。

1. MySQL サーバーを起動します。
2. Management Console から、Process Discovery Analyzer パラメータを指定します。
   a. **[設定]** メニューを展開し、**[Process Discovery Analyzer]** をクリックします。
   b. Process Discovery Analyzer データベース接続設定の指定[5]ランタイム パラメータ、クラスタ設定、データベースのルート パスワードを指定したりすることはできません。
   c. [データベースのプロビジョニング] タブで [スキーマの作成] と [ユーザーにアクセス権を付与] をクリックして、データベース プロビジョニングを実行します。詳細については、Process Discovery Analyzerを参照してください。
3. KofaxRPAProcessDiscoveryAnalyzer-11.5.0.0.[ビルド番号].zip をダウンロードして、ディスク上のフォルダに抽出します。

   デフォルトでは、Analyzer は /opt/KofaxRPA/ProcessDiscoveryAnalyzer にインストールされます。
4. システムに Analyzer をインストール、アップグレード、またはアンインストールするには、適切なコマンドを実行します。
   - Debian ベースのシステムに Analyzer をインストールまたはアップグレードするには、次のコマンドを使用します。

     ```
     sudo dpkg -i KofaxRPAProcessDiscoveryAnalyzer-11.5.0.0.[ビルド番号].deb
     ```
   - Analyzer をアンインストールするには、sudo dpkg -P kofax-rpa-processdiscoveryanalyzer を実行します。
5. Analyzer を実行するには、/opt/KofaxRPA/ProcessDiscoveryAnalyzer に移動して、次のコマンドを実行します。

   ```
   ./KofaxRPAProcessDiscoveryAnalyzer
   ```

Analyzer のデフォルトの設定

```
MC_URL=localhost:50080
LOG=INFO
LOCALE=<system settings>
```

Analyzer のオプションとデフォルト設定のリストについては、Process Discovery Analyzer オプションを参照してください。

Process Discovery Analyzer クラスタ

機械学習アルゴリズムを使用した自動プロセス検出を行って Process Discovery のデータ フローが増大した場合は、より多くの処理能力が必要となります。効率的かつ効果的なデータ処理に対するニーズを合理化するために、Process Discovery は、高パフォーマンス ビッグ データ分析のための Analyzer クラスタ設定を提供します。

[5] これは、MySQL サーバーを実行しているコンピュータの IP アドレスまたはドメイン名です。アドレスとして localhost または 127.0.0.1 を指定したり、

Process Discovery Analyzer クラスタは Apache Spark テクノロジを基盤にしています。クラスタは、Process Discovery Analyzer アプリケーションを実行するクラスタ ノードとして機能する複数のコンピュータで構成された分散コンピューティング フレームワークです。Spark クラスタは、マスター ノードと 1 つ以上のワーカー ノードで構成されます。マスターおよび各ワーカー ノードには、同じ Analyzer アプリケーションがインストールされます。マスター ノードは、クラスタの作業を調整し、Analyzer ワーカー プロセスを実行してデータ処理に参加します。ワーカー ノードは、Analyzer ワーカー プロセスを実行してデータを分析します。異なる OS で実行されている Analyzer クラスタ ノードを混在させることができます。マスター ノードを割り当てるには、Management Console でそのアドレスを指定します。

Apache Spark クラスタ テクノロジーにより、ワーカー ノードと同様の処理能力とメモリ量を持つコンピュータが暗黙的に使用されます。クラスタ ノードとして使用するコンピュータ間で RAM と処理能力に大きな違いがある場合、Analyzer を実行するには、大量の RAM が搭載されたコンピュータでコンテナー (Docker コンテナーなど) を使用します。このようにすることで、処理能力が低いノードと同様の特性を持つ複数のクラスタを作成できます。

ノードの中で最も処理能力の高いコンピュータをマスター ノードとして使用して、高い負荷を分散させることをお勧めします。

ノードの最小 (デフォルト) 数は 1 ノードです。Process Discovery Agents が収集するデータの量と分析時間の要件に応じて、必要な数のクラスタ ノードを追加できます。たとえば、Analyzer のデータ処理時間を短縮する場合は、クラスタにノードを追加してから処理時間を再度確認します。処理時間は、Kofax Analytics for RPA ビュー内の Process Discovery ビューのステータスレポートに表示されます。

2 倍の数のノードを追加した場合でも、ノードの調整およびノード間でのデータの送信にはある程度の時間がかかるため、処理時間は半分にはならないことに注意してください。この場合、ネットワーク遅延を削減すると、クラスタのパフォーマンスが向上します。また、非力なサーバーを複数追加するよりも、大容量の RAM を持つ 1 つの強力なサーバーをノードとして追加する方が効率的です。Analyzer はシステムで利用可能なすべての処理能力を使用するため、クラスタ内のノードとして専用のコンピュータを使用することをお勧めします。

Process Discovery Analyzer クラスタの設定

1. Process Discovery Analyzer の **[<data conref="../../All_Variables/variables.xml#variables/ProcDiscAnal"></data>]** > **[クラスタ設定]** で、マスター ノードの割り当て、ネットワーク パターンの指定 (オプション)、その他の設定など、必要なすべてのパラメータを指定します。詳細については、Process Discovery Analyzer を参照してください。
2. a. Analyzer クラスタ ワーカー ノードとして使用するコンピュータに Process Discovery Analyzer をインストールし、設定して、起動します。
   b. すべてのワーカー ノードを起動した後に、Analyzer クラスタ マスター ノードとして使用するコンピュータに Process Discovery Analyzer をインストールし、設定して、起動します。

   ノードで Analyzer アプリケーションのインスタンスを起動するときに、Analyzer クラスタ設定を定義した Management Console アドレスを指定します。必要に応じてその他のパラメータを指定します。詳細については、Process Discovery Analyzer を参照してください。

   すべてのノードが実行された後に、必要に応じてワーカー ノードを追加、削除、設定することができます。変更は、次の Analyzer の実行時に適用されます。

いずれかのワーカー ノードに障害が発生した場合は、基盤となる Apache Spark テクノロジがデータを保持し、作業ノード間で負荷を分散します。Analyzer の設定を変更した場合は、マスター ノードを再起

動します。別のコンピュータをマスター ノードとして割り当てた場合は、現在のマスター ノードおよび新しく割り当てたマスター ノードを両方とも再起動します。たとえば、現在、クラスタに A、B、および C という 3 つのノードが含まれていて、「A」がマスター ノードになっているとします。この状況で「B」をマスター ノードとして割り当てた場合は、「A」と「B」を再起動します。

マスター ノードの Analyzer ログで `outofmemory` タイプのエラーが発生した場合は、Management Console の **[設定]** > **[Process Discovery Analyzer]** > **[クラスタ設定]** を開き、[マスター メモリ **(GB)**] 設定のメモリ容量を増やして、マスター ノードを再起動します。詳細については、Process Discovery Analyzerの「クラスタ設定」を参照してください。

クラスタ ノードの監視

Apache Spark クラスタには、ノードのアクティビティを監視するツールが含まれます。環境にクラスタをセットアップした後に、マスター ノード ダッシュボードを開いて、すべてのワーカー ノードが稼働していることを確認します。このダッシュボードには、実行中のアプリケーションおよび完了したアプリケーションに関するいくつかの基本情報、およびクラスタ ワーカーのリストが表示されます。リスト内のアプリケーション ID をクリックすると、アプリケーションの詳細を表示できます。ブラウザでマスター ノード ダッシュボードを開くには、マスターのアドレスの後に、Management Console の [マスター **WebUI** ポート] オプションで指定したポート番号を続けて入力します。例：

```
10.10.0.15:8080
```

リストのワーカー ID をクリックすると、ワーカー ダッシュボードを開くことができます。ブラウザでワーカー ダッシュボードを直接開くには、ワーカー アドレスの後に、Management Console の [ワーカー **WebUI** ポート] オプションで指定したポート番号を続けて入力します。例：

```
10.10.0.11:8081
```

マスター ノードとワーカー ノードのログは、一般的な Analyzer ログ ファイルと同じ場所にあります。システム内のログ ファイルを見つける方法についてには、Process Discovery Analyzerの「ログ ファイル」セクションを参照してください。

Apache Spark クラスタの詳細については、https://spark.apache.org/ にある Apache Spark のドキュメントを参照してください。

Analytics

Kofax Analytics for RPA ダッシュボードの Process Discovery ビューには、Process Discovery Agents によって収集されたデータに基づく情報を含んだレポートが表示されます。このビューには、データに関するさまざまな視覚的および分析的表現がチャートやテーブルとして表示されます。システム管理者、ビジネス プロセス マネージャ、およびその他の利害関係者は、このインターフェイスを使用して分析情報を可視化できます。

Insight ダッシュボードでデータを正しく表示するには、RoboServer および Management Console のインスタンスを実行中のコンピュータのタイム ゾーンに応じて、時刻が Java で正しく設定されていることを確認してください。最新版に更新されたタイム ゾーンについては、Oracle web サイトにある「Timezone Data Versions in the JRE Software」を参照してください。必要に応じて、Oracle「タイム ゾーン アップデート ツール」を使用してタイム ゾーン情報を更新します。

データベース

Kofax RPAProcess Discovery は、MySQL データベースを使用するように設計されています。サポートされている MySQL のバージョンについては、Kofax RPA 11.5.0 次のドキュメント サイトにある『Kofax RPA 技術仕様書』 を参照してください: https://docshield.kofax.com/Portal/Products/RPA/11.5.0-nlfihq5gwr/RPA.htm。

パフォーマンスの問題を防ぐためのセーフガードとして、これらの手順に従うことをお勧めします。

- MySQL データベースには SSD などの高速ディスク ドライブを備えた専用サーバーを使用します。基盤となるハードウェアが強力でない場合は、1 つの Process Discovery グループに対して 100 を超えるエージェントを同時に実行しないことをお勧めします。グループあたりのエージェントの最大数は、記録されたユーザー アクティビティの強度と、これらのアクティビティが発生するアプリケーションによって異なります。
- Management Console で指定された MySQL データベース名の長さの上限は、104 文字です。
- Process Discovery Agent の 1 つのインスタンスは、ユーザーごとに 1 日あたり約 500 MB のデータを記録します。

  MySQL サーバーに十分なストレージ容量があることを確認してください。
- 構成ファイルに保存されているデフォルトの MySQL サーバー設定の一部を変更します。ファイル名と場所は、オペレーティング システムと MySQL 配布パッケージのバージョンによって異なる場合があります。

  たとえば、Windows では、MySQL 8 の構成ファイルは次の場所にあります。

  `C:\ProgramData\MySQL\MySQL Server 8.0\data\my.ini`

  - 「`max_allowed_packet`」 を少なくとも 16 MB (「`max_allowed_packet=16MB`」) に設定します。
  - 「`innodb_buffer_pool_size`」 を使用可能な RAM の 80% に設定します。

    たとえば、MySQL サーバーに合計 8.0 GB の RAM があり、5.0 GB の RAM が使用可能な場合、「`innodb_buffer_pool_size`」 を 4096 MB (「`innodb_buffer_pool_size=4096MB`」) に設定する必要があります (5.0GB * 1024 * 0.8 = 4096 MB)。
  - バイナリ ログ機能を無効にします。

    有効にすると、エージェントがデータベースに記録するデータ量が 2 倍になります。

    無効にするには、構成ファイルの `[mysqld]` セクションに 「skip-log-bin」 行を追加します。
  - 接続の最大数を 1000 (「`max_connections=1000`」) に設定します。

  MySQL サーバーを再起動して、構成ファイルで行った変更を適用します。
- データ保持。

  エージェント データが不要になったら、特定の Discovery グループに対して Management Console で指定されたエージェント データベース スキーマを削除します。そのために、MySQL Workbench などの任意のデータベース管理ツールを使用できます。

  同じ手順に従って Process Discovery Analyzer データを保存します。

第 8 章

Kofax Analytics for RPA

Kofax Analytics for RPA は Kofax RPA の拡張機能で、Kofax RPA ロボットと Process Discovery Agent が動作している間に収集されたデータに基づいてグラフィカルなビジネス インテリジェンス ダッシュボードを生成します。

Kofax Analytics for RPA ビューは、事前定義された Kofax レコードと指標の値に基づいています。Insight Studio 内の Dashboard Designer を使用してカスタム ビューを追加する場合は、製品に付属する事前定義済みのビュー、レコード、または指標を変更しないでください。既存のビューのコピーを作成した後に、設定をカスタマイズできます。詳細については、Kofax Web サイトの『Kofax Analytics Project Customizations アプリケーション ノート』を参照してください。

タイム ゾーンの設定

Kofax Analytics for RPA ダッシュボードでデータが正しく表示されるようにするには、RoboServer の時刻がタイム ゾーンに従って Java で正しく設定されていること、および Management Console がコンピュータで実行されていることを確認してください。タイム ゾーンの最新の更新内容については、Oracle Web サイトの「*Timezone Data Versions in the JRE Software*」を参照してください。必要に応じて、Oracle「タイム ゾーン アップデート ツール」を使用してタイム ゾーン情報を更新します。

Kofax Analytics for RPA のインストールと設定

このセクションの指示に従って、新しい Kofax Analytics for RPA のインストールを実行します。

以前のバージョンの Kofax Analytics for RPA からアップグレードする場合は、以前のバージョンからのアップグレードを参照してください。

Kofax Analytics for RPA 製品には、RPA コンポーネントと組み込みビューを正常にインポートするために必要なプロジェクト バンドル ファイルの `KofaxAnalyticsforRPA-<バージョン>.zip` が含まれています。

SSL

最初に SSL なしで Kofax Analytics for RPA インストールを実行した場合でも、後から SSL に切り替えられます。SSL に切り替えるには、[既存のデータベースを使用] オプションを使用して Kofax Insight を再インストールします。

インストールと設定のチェックリスト

次のチェックリストは、Kofax Analytics for RPA のセットアップに必要な手順の概要を示しています。

1. 設定を確認し、必要に応じて Kofax RPA で次のパラメータを設定します。設定は、Kofax RPA Management Console の **[設定]** > **[一般]** タブにあります。

   ロボット実行の分析の場合
   - RoboServer ログ データベース: [データベースに記録] を選択し、データベース パラメータを指定します。
   - Analytics データベース: [**Analytics** データベースに統計情報を記録] を選択し、データベース パラメータを指定します。Analytics スキーマの作成に必要な SQL スクリプトは、Kofax RPA インストール ディレクトリの `documentation\sql\statistics` ディレクトリにあります。

   Process Discovery の分析の場合
   Process Discovery を展開します。詳細については、Process Discovery を参照してください。

2. Insight データベース (admin、メタデータ、およびデータ) を作成して構成します。以下の「データベース」トピックを参照してください。
3. Kofax Insight をインストール 。
4. プロジェクトのインポートと設定 。

データベース

新規または既存のデータベース管理システム インスタンスを使用して、Kofax Analytics for RPA で使用する次のデータベースを作成します。

- [メタデータ]: 指標定義や計算ロジックなどの設定情報を格納します。`rpa_analytics_meta` などの名前を割り当てます。
- [データ]: 処理されたレコードと指標を格納します。`rpa_analytics_data` などの任意の名前を割り当てることができます。
- 管理: ユーザー、ロール、フィルタリング、アラート、ログなどに関連する Insight の管理データを格納します。`rpa_analytics_admin` などの名前を割り当てます。

rpa_analytics_admin、rpa_analytics_meta、および rpa_analytics_data など、データベースごとに一貫した命名規則を使用することをお勧めします。

⚠ 製品ドキュメントの推奨事項に従って、すべての Kofax RPA 製品データベースを作成して保守運用してください。データベースの変更またはカスタマイズを検討している場合は、Kofax に確認したうえで続行するようにしてください。確認せずに実行した場合、結果が予測不能になり、ソフトウェアが動作しなくなる可能性があります。

Oracle データベースの場合、すべてのデータベース スキーマがすでに配置されている状態で、製品のインストールに進む必要があります。

Kofax RPA で Oracle データベースを使用して分析データを収集し、新しいバージョンの Kofax RPA へのアップグレード中に同じデータベースを指定する場合は、データベース内のテーブルを手動で削除して作成する必要があります。データベース テーブルの作成については、『Kofax RPA のヘルプ』の「RoboServer ログ データベース」および「データベース テーブルを作成するためのスクリプト」のトピックを参照してください。

Kofax Insight をインストール

Kofax Insight は、ロボット実行データと Kofax RPA Process Discovery Agents によって収集されたデータに基づいて、グラフィカルなビジネス インテリジェンス ダッシュボードを作成します。

インストール手順の詳細については、『Kofax Insight インストール ガイド』を参照してください。Kofax Analytics for RPA は、Kofax Insight バージョン 6.5 以降で動作します。

Insight インストール プロセスを開始する前に、Kofax Insight が実行されているコンピュータに MySQL Connector/NET をインストールします。MySQL Connector/NET は MySQL Web サイトからダウンロードできます。

1. `KofaxInsightSetup_6.5.0.0.0.<build>_x64.msi` などの Kofax Insight インストール ファイルを実行します。Insight のビルド番号は、『Kofax Insight リリースノート』の「バージョン情報」セクションに記載されています。デフォルトのオプションを使用して、Kofax Insight をインストールします。

   Kofax Insight セットアップ ウィザードを完了した後、またはフィックス パック インストール ファイルを実行して Kofax Analytics for RPA の設定手順全体を実行した後に、最新のフィックス パックをインストールします。

   インストール ウィザードが完了すると、[**Kofax Insight** セットアップ ウィザードの完了] ウィンドウが表示されます。
2. [終了] をクリックします。Insight Installation Manager が起動します。[終了] をクリックする前に [**Insight Installation Manager** を起動する] オプションをオフにした場合は、後で [スタート] メニューから Installation Manager を起動できます。[スタート] > [**Insight 6.5.0**] > [管理] > [**Installation Manager**] の順に移動します。

   Installation Manager はインストール プロセスを続行し、Insight 管理データベースやその他のパラメータなど、Kofax Insight のセットアップと設定を支援します。詳細については、『Kofax Insight インストール ガイド』を参照してください。

Insight のインストールが完了すると、Insight Installation Manager の最終画面にいくつかのオプションが表示されます。

ダッシュボード プロジェクトを設定するには、[**Admin Console**] をクリックして次のセクションに進みます。

プロジェクトのインポートと設定

ここで示されている手順を使用して、Kofax RPA ダッシュボード プロジェクトを設定します。

デフォルトでは、[ユーザーのログインおよびパスワードは **Insight** プラットフォーム内で定義されます。] が設定されることに注意してください。プロジェクトの [ユーザーは明示的にロールにリンク] オプション。認証タイプは後で変更できます。

1. Insight Admin Console を起動します。
   - Insight Installation Manager を閉じていない場合は、[インストールは完了しました] ウィンドウで、[**Admin Console**] をクリックします。
   - Insight Installation Manager を閉じた場合は、**Start** > **All Programs** > **Insight 6.5.0** > **Admin Console** の順に移動します。
2. Insight のインストール中に指定した管理者のログイン資格情報を入力します。
   Admin Console を初めて起動する場合は、[ライセンス] タブでライセンス情報を入力するように求められます。
3. ナビゲーション パネルで [プロジェクト] をクリックし、右クリックして [新しいプロジェクト] を選択します。
   [新しいプロジェクト] ウィンドウが表示されます。
4. プロジェクト名として `KofaxRPA` と入力し、[**OK**] をクリックします。
   新しいプロジェクトの作成 [KofaxRPA] ウィンドウが表示されます
5. [**Import from file**] タブをクリックし、[ファイル選択] をクリックします。[**File is located on the client computer and will be copied to the server for processing**] を選択します。
   [ファイル アップロード] ウィンドウが表示されます。
6. `KofaxAnalyticsforRPA-<`バージョン`>.zip` に移動し、[開く] をクリックします。
   ファイルがインポートされるまで待ちます。
7. メタ データベースとデータ データベースのデータベース接続情報を設定します。

データベース名フィールドには、プロジェクト名およびデータベース末尾である `Kofax_RPA_meta` などが自動的に事前入力されます。要件に合わせて名前を編集します。データベース名にはスペースを含めないでください。

> サーバーの詳細がメタ データベースとデータ データベースでほぼ同一である場合は、[**Same server as the meta database**] を選択して、[メタ] セクションから [データ] セクションにエントリをコピーします。エントリがコピーされたら、必ず [データ] セクションのデータベース名を更新してください。
> データベースがまだ存在していないため、このステップでは [接続] をクリックしないでください。

8. [**OK**] をクリックして、プロジェクトをインポートします。
9. データベースの作成について確認するよう要求されたら、データベースがすでに存在している場合でも [はい] をクリックします。Insight により、データベース内に必須のテーブルが作成されます。
   プロジェクトのインポートが進行している間は、画面にインジケータが表示されます。
   Kofax RPA の組み込みのビュー、レコード、および指標がインストールに追加されます。
10. 新しく作成したプロジェクトを展開し、[データ ソース] に次のデータベースがあることを確認します。
    - **Data DB**: Insight データ ウェアハウス
    - **Analytics**: Kofax RPA Management Consoleの **[設定]** > **[一般]** > **[Analytics** データベース**]** で設定したレポート データベース
    - ログ: Kofax RPA で設定したログ データベース
    - **erd_dean**: Process Discovery データベース

    プロジェクトを展開するには、プロジェクト名の右側にある矢印をクリックします。
11. サーバー名、データベース名、ユーザー、パスワードを指定して、各データベースに接続するためのパラメータを設定します。[接続] をクリックして接続をテストします。接続が成功すると、円形のインジケータが緑色に変わります。

> Analyzer データベースは、Kofax RPA Management Console の **[設定]** > **Process Discovery Analyzer [**データベースの設定**]** で設定します。

12. [保存] をクリックして設定を保存します。
13. [ユーザー] を展開し、Kofax RPA ビューアに直接アクセスしてレポートを表示できる組み込みユーザーのパスワードを設定します。[保存] をクリックしてパスワードを保存します。次のようなユーザーが存在します。
    - **KAFK**: このユーザーは、Process Discovery および RPA ダッシュボードにアクセスできます。
    - **RPA**: このユーザーは RPA ダッシュボードにのみアクセスできます。
    - **PD**: このユーザーは、Process Discovery ダッシュボードにのみアクセスできます。
    - **KAFK_admin**: このユーザーは、バージョン 1.0 との下位互換性を確保するために用意されており、管理者と同様の権限を持ちます。
14. Insight Admin Console で [**Studio**] をクリックして Insight Studio を開くか、Web ブラウザを開いて次の URL を入力します。
    `http://<server>/Insight/Studio/`

15. [実行プラン] を展開し、インポートされた Kofax RPA プロジェクトの [**1** 分のプラン] をクリックします。
    この計画は、5 分ごとにデータをロードするようにスケジュールされています。
16. [データのロード] をクリックし、[日付範囲] タブで次の操作を行います。
    a. その日収集されたデータをロードして表示するには、当日の日付と翌日の日付を選択します。
    b. [**Load Data**] をクリックします。
       前の日に収集されたデータをロードするには、期間全体をロードするのではなく、1 日ごとにデータをロードします。1 日分を超えるデータをロードすると負荷が増加し、プログラムでデータをロードできなくなる可能性があります。

設定が完了したら、ダッシュボードの使用を開始できます。

以前のバージョンからのアップグレード

Kofax Analytics for RPA 2.8.0 は Kofax Insight 6.5.0 以降を使用するように設計されています。次の表に、Kofax Analytics for RPA バージョンの互換性に関する情報を示します。

Kofax Analytics for RPA のバージョン	**Kofax RPA** のバージョン	**Kofax** のバージョン **Insight**
Kofax Analytics for RPA 2.3.0	Kofax RPA 11.0.0	KofaxInsight 6.2.0 以降
Kofax Analytics for RPA 2.4.0	Kofax RPA 11.0.0、11.1.0	KofaxInsight 6.2.0 以降
Kofax Analytics for RPA 2.5.0	Kofax RPA 11.2.0	KofaxInsight 6.3.0 以降
Kofax Analytics for RPA 2.6.0	Kofax RPA 11.3.0	Kofax Insight 6.3.0 以降
Kofax Analytics for RPA 2.7.0	Kofax RPA 11.4.0	Kofax Insight 6.5.0 以降

Kofax Insight のアップグレードについては、『Kofax Insight インストール ガイド』にある「Insight のアップグレード」の章を参照してください。アップグレードを実行する前に、Kofax サポートに問い合わせることもできます。

Windows 認証

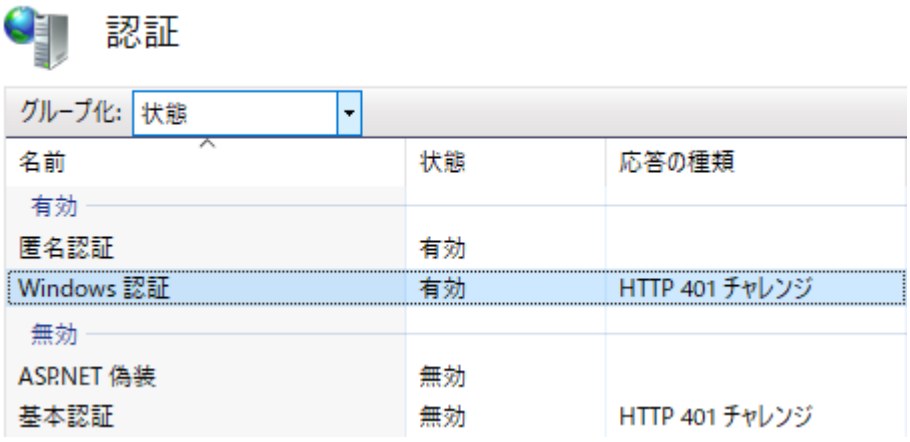
Windows ユーザー認証を使用している場合は、以下のアカウントにすべての Kofax Analytics for RPA データベースへのアクセス権を付与する必要があります。

1. Kofax Analytics for RPA インストールの実行に使用するユーザー アカウント
2. IIS アプリケーション プールのアカウント/ID
3. Insight Scheduler サービスのアカウント/ID

Insight の Windows 認証の設定

このセクションの手順を使用して、Kofax Insight に Windows 認証を設定します。

Insight を設定する前に、Web アプリケーション (デフォルトの Web サイト) に対して IIS で Windows 認証を選択します。



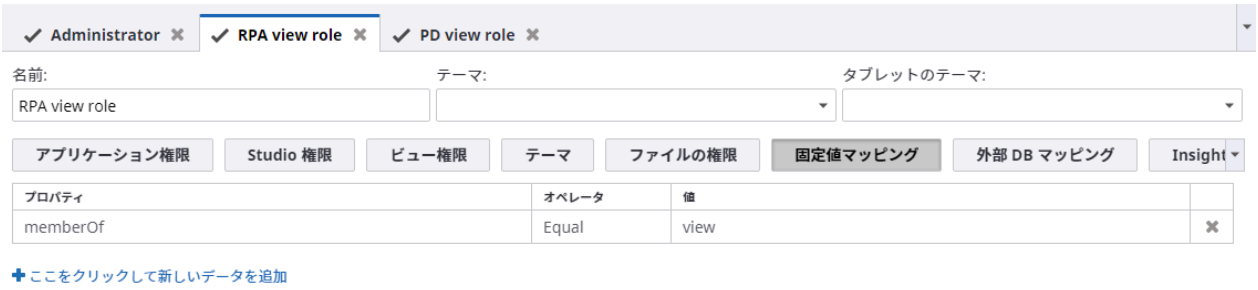
1. **[スタート]** > **All Programs** > **Insight 6.5.0** > **Administration** > **Admin Console** の順に移動します。
2. ナビゲーション パネルで [認証] をクリックします。
3. [認証方法] タブをクリックして、次のいずれかを選択します。
   a. [ユーザーのプロパティは環境から取得されます]: **Windows**
   b. **And then user roles and access rights are determined by comparing these values to**: **Fixed values**
4. ユーザー識別子 **(UID)**: ユーザーの ID を取得する方法を指定します。ID は、特定のユーザーのログインに対して一定である必要があります。通常、これは Active Directory ドメインとユーザー名を参照するセッション プロパティ (ID) です。
   a. ナビゲーション パネルで、[ユーザー マッピング] をクリックします。
   b. ユーザー ID (UID) の [ユーザー マッピング] タブで、「セッションのプロパティ」を [**ID**] に設定します。
5. [ユーザー名] および [電子メール] のセッション プロパティを設定します。
   a. [ユーザー名] の [ユーザー マッピング] タブで、「セッションのプロパティ」を [**FullName**] に設定します。
      [ユーザー名] は、ユーザー アカウントの表示名です。通常、これは *Identity*、*Name*、*FullName*、*displayName* などの Active Directory のプロパティの 1 つ、または別の便利なプロパティです。ドメイン管理者は、利用可能なすべての Active Directory プロパティを提供できます。
   b. 電子メールの [ユーザー マッピング] タブで、[セッションのプロパティ] を [電子メール アドレス] に設定します。
      電子メールは、ユーザー アカウントの電子メール アドレスです。これはセルフ サブスクリプションにのみ使用されます。通常、これは Active Directory プロパティの電子メールまたは電子メール アドレスです。ドメイン管理者は、利用可能なすべての Active Directory プロパティを提供できます。

マッピング ロール

ロールは、テーマ、日付形式など、事前定義された一連の管理設定を定義します。また、ロールは、プロジェクトおよびダッシュボード ビューへの特定のアクセス権を定義します。それぞれのロールに対

してマッピング ルールを記述する必要があります。通常は、Active Directory プロパティの *memberOf* が使用されます。サンプル図では、*admin* を含む、Active Directory プロパティ *memberOf* を持つユーザーが、KAFK 管理者ロールに割り当てられています。ドメイン管理者は、利用可能なすべての Active Directory プロパティを提供できます。

マッピング グリッドの各行は **AND** オペランドを使用します。[ロール] リストの複数のロールがユーザー アカウントの条件に一致する場合、一致するすべてのロールのアクセス権が結合されますが、他の設定 (テーマや日付形式など) は、リストの最上位の一致するロールによって割り当てられます。



Kofax Analytics for RPA の使用

Kofax Analytics for RPA を使用すると、ログ データベースおよび Analytics データベースから取得したデータを、ビューアを介してインタラクティブなダッシュボードで表示できます。

Kofax Analytics for RPA のダッシュボードは、複数の組み込みビューで構成されています。各ビューには、Kofax RPA データベースのデータを分析するためのグラフィック エレメントとその他のコンポーネントがあります。

Viewer

Viewer を使用して、Kofax Analytics for RPA プロジェクトに含まれるダッシュボード ビューを表示します。これらのビューには、データに関するさまざまな視覚的表現および分析的表現がチャートやテーブルを使用して表示されます。システム管理者、ビジネス プロセス マネージャ、およびその他の利害関係者は、このインターフェイスを使用して分析情報を可視化できます。

ビューアのデータは 5 分ごとに更新されることに注意してください。

Viewer を開く

認証方法に応じて、Viewer を開く手順は異なります。認証タイプに応じて、このセクションの手順を実行します。

Insight ユーザー認証

次のいずれかを実行して、ビューアを開きます。

- **Start** > **All Programs** > **Insight 6.5.0** > **Viewer** の順に移動します。
- Web ブラウザを開き、次の URL を入力します。

  `http://[サーバー]/Insight/View/`

ここで［サーバー］は、Insight を実行しているコンピュータの名前または IP アドレスです。

ログイン ウィンドウが表示されます。Insight Admin Console で、ユーザー名 (KAFK、RPA、または PD) の 1 つと、そのユーザーに設定されているパスワードを入力します。[ログイン] をクリックします。上記のユーザーの 1 人だけがビューアに直接アクセスできることに注意してください。ユーザーの設定に関する詳細については、プロジェクトのインポートと設定を参照してください。

IIS 設定で、Web サイトのバインディング ホスト名が空白または localhost に設定されていることを確認してください。空白または localhost に設定されていない場合、ログイン エラーが発生する可能性があります。

正しく表示されるようにするには、Insight Admin Console で適切な認証方法の設定を確認してください。「認証なし」の設定はサポートされていないため、ブラウザ ウィンドウでエラーまたは予期しない結果が発生する可能性があります。詳細については、プロジェクトのインポートと設定を参照してください。

また、ログインしているユーザーに関連付けられているロールのデフォルトのビューとしてメインが選択されていない場合は、次のエラーが表示されることがあります。

```
You can specify view name parameter in Admin Tool
```

メインをデフォルトのビューとして設定することにより、問題を解決します。

Windows ユーザー認証

次のいずれかを実行して、ビューアを開きます。

- **Start** > **All Programs** > **Insight** > **Viewer** の順に移動します。
- Web ブラウザを開き、次の URL を入力します。

  `http://[サーバー]/Insight/View/`

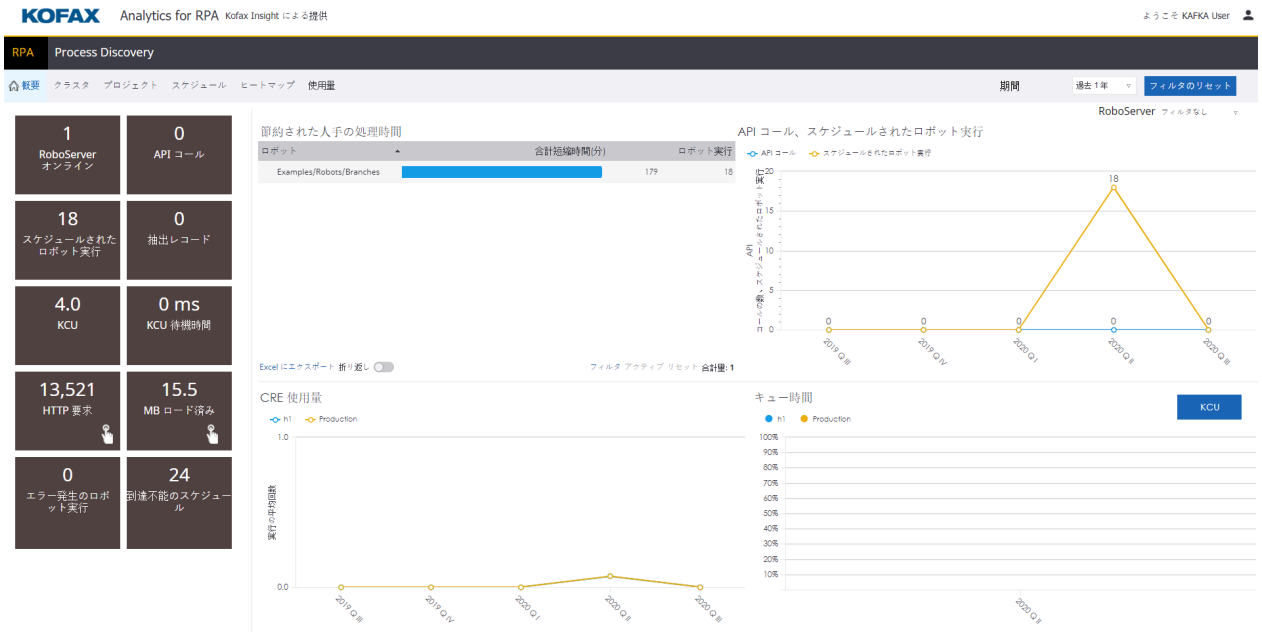
  ここで［サーバー］は、サーバーの名前です。

  IIS 設定で、Web サイトのバインディング ホスト名が空白または localhost に設定されていることを確認してください。空白または localhost に設定されていない場合、ログイン エラーが発生する可能性があります。

Viewer の画面レイアウトとナビゲーション

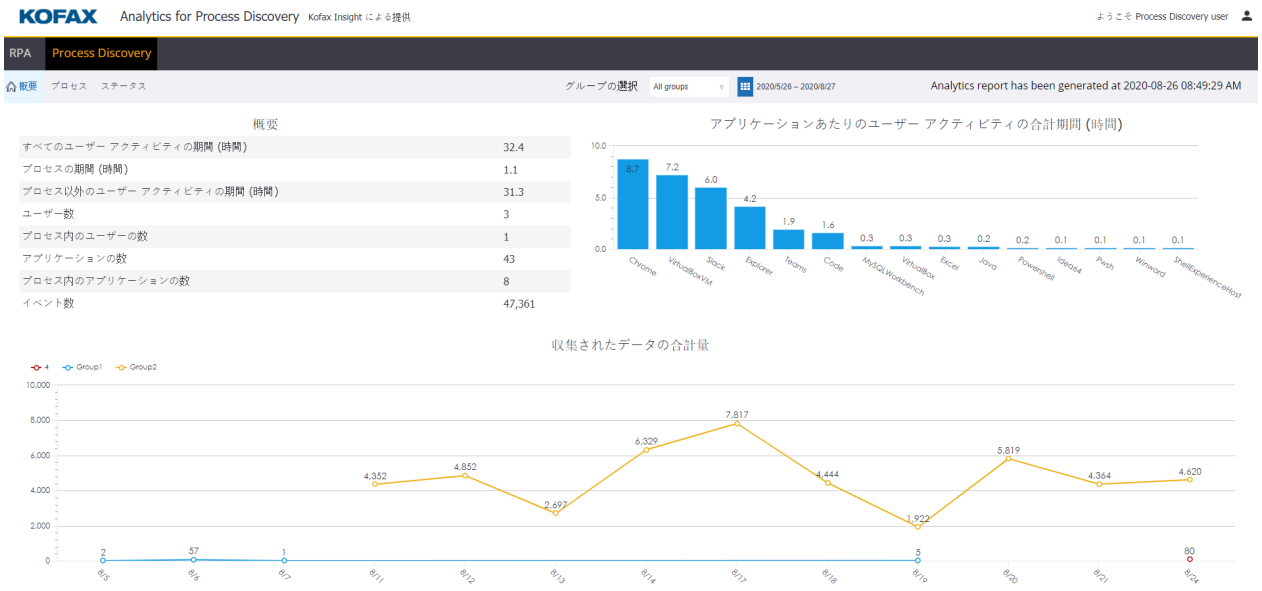
ツールバー メニュー メニューを使用して、**RPA** や **Process Discovery** などの各セクションや、選択したセクションのさまざまなビューにアクセスします。頻繁に使用するビューをブックマークしてログアウトするには、追加のユーザー レベルの項目を使用します。
コンテキスト メニューとチャート オプション グラフやグリッドを右クリックして、[リセット]、[Pivot]、[ズームイン]、[表示] などの他のオプションにアクセスします。これらのオプションは、チャートの種類または右クリックした領域に応じて異なります。

RPA ビュー



RPA ビュー フィルタ
[期間] リストの時間と、[**RoboServer**] リストでロボットが実行されている RoboServer でレポートをフィルタリングできます。

Process Discovery ビュー



Process Discovery フィルタ入力

Process Discovery のグループ名と日付で結果をフィルタリングします。

[**Process Discovery** グループ] リストでグループ名を選択します。[フィルタなし] を選択すると、すべてのグループの情報がビューに表示されます。

Viewer の使用

ビューアでの作業中に、次のことができます。

- リスト内のプロジェクト、ロボット、またはスケジュールの名前をクリックして、アクティビティの詳細にドリルダウンする。
- ダッシュボード上のコンポーネントを操作する。
- グラフのポイントにカーソルを合わせ、詳細を表示する。
- グラフ名をダブルクリックして、ブラウザ ウィンドウ全体に展開する。
- グラフ名の横をクリックしてドラッグし、リスト表示に切り替えて元に戻す。
- [期間] リストで統計の期間を選択する。
- フィルタのデフォルト値を復元するため、[フィルタのリセット] をクリックする。

Excel にエクスポート

[Excel にエクスポート] 機能を備えたグリッドでは、グリッドのコンテンツを Microsoft Excel にエクスポートするための設定を行うことができます。

1. [**Excel** にエクスポート] が有効になっているグリッドで、[**Excel** にエクスポート] をクリックします。

   [**Excel** にエクスポート] ウィンドウが表示されます。

2. [コンテンツをエクスポート] オプションを選択します。
   - [現在のページおよびトップ ドリルダウン]
   - **All pages of a grid. top drill down level. Could take some time.**
   - **All pages of a grid, all drill down levels. Could take much longer.**
3. [エクスポート形式] オプションを選択します。
   - [未フォーマットの **XML** ファイル、**Excel** で読み込み可]
   - **Formatted Excel file**
   - [未フォーマットの **CSV** ファイル]
   - [未フォーマットの **TSV** ファイル]
4. 必要に応じて、[有効] を選択し、フラット エクスポートを使用します。これにより、ドリルダウンまたは拡張可能なデータの詳細が保持されます。
5. [**OK**] をクリックし、プロンプトが表示されたら、.xml ファイルを保存するか、開きます。

Kofax Analytics for RPA ビュー

このセクションでは、Kofax Analytics for RPA プロジェクトで表示される各ビューの情報について説明します。ビュー上のさまざまなオブジェクトを操作して、詳細な情報を取得できます。Kofax Analytics for RPA では、RPA ビューと Process Discovery ビューが用意されています。

RPA ビュー

RPA ビューには、Kofax RPA の分析情報が表示されます。

- 概要
- クラスタ
- プロジェクト
- スケジュール
- ヒートマップ
- 使用量

概要

概要では、Kofax RPA システムのパフォーマンス パラメータがグラフィックで表示されるため、容量のボトルネックやシステム使用率の問題をすばやく確認できます。[期間] リストで期間を選択し、[**RoboServer**] リストで [RoboServer] を選択することで、データをフィルタリングできます。

左側のペインのタイルのパーセントは、選択した期間および選択した期間の前の同じ期間との差を示します。たとえば、期間として [過去 1 時間] を選択した場合、パーセントには過去 1 時間とその前の 1 時間との差が表示されます。値の横にある三角形は、値が減少すると下向きになり、値が増加すると上向きになることに注意してください。

- API コール、スケジュールされたロボット実行、抽出されたレコード、HTTP 要求、MB ロード済みの場合。値がゼロより大きい場合、パーセントは緑色になり、値が増加したことを示します。
- KCU、KCU 待機時間、エラー発生のロボット実行、到達不能のスケジュールの場合。値がゼロより小さい場合、パーセントは緑色になり、値が減少したことを示します。

ネットワークの詳細を表示するには、この記号 のタイルをクリックします。

節約された人手の処理時間
このグラフには、Design Studio の [ロボットの設定] ダイアログ ボックスの [手動の処理時間] オプションで指定した値と、指定した期間中のすべてのロボット実行に対して計算されたロボット実行の実際の継続時間との差に関する情報が表示されます。[手動の処理時間] オプションでは、選択したロボットによって実行時に行われるタスクをユーザーが実行する場合の所要時間を指定できます。

API コール、スケジュールされたロボット実行
API または手動で開始されたロボット実行の数。

CRE 使用量
利用可能な最大ライセンスのうち、CRE ライセンスを使用して実行されたロボットの割合。CRE の詳細については、『Kofax RPA インストール ガイド』の「同時ロボット実行ライセンス」を参照してください

概要レポートには、選択した Management Console で使用されるライセンスに応じて、CRE 使用量または KCU 使用量の情報が表示されます。

キュー時間
合計実行時間とキュー時間の比率 (パーセント)。

KCU 使用量
クラスタごとの利用可能な最大ライセンスのうち、KCU を使用して実行されたロボットの割合。

待機時間
合計実行時間と KCU 待機時間の比率 (パーセント) (KCU ライセンスが使用されている場合のみ表示)。

KCU の健全性

KCU (Kofax RPA 演算ユニット) の健全性は、API コールとスケジュールの 2 つの指標に基づいて計算されます。KCU の詳細については、『Kofax RPA インストール ガイド』の「Kofax RPA 演算ユニット」を参照してください。

API コールの KCU の健全性は、ロボット実行からの KCU 待機時間に基づいて計算されます。
- ロボット実行の最大 KCU 待機時間が 1 秒を超えると、KCU の健全性が黄色になります。
- ロボット実行の平均 KCU 待機時間が 1 秒を超えると、KCU の健全性が赤色になります。

スケジュールの KCU の健全性は、統計から次のように算出されます (ゼロ ベースは 1 分あたり 2 秒、ピーク制限は 1 分あたり 10 秒)。
- データ ポイントの 10% にピーク制限を超える KCU 待機時間がある場合、またはデータ ポイントの 50% にゼロ ベースを超える KCU 待機時間がある場合、KCU の健全性は黄色になります。
- データ ポイントの 50% にピーク制限を超える KCU 待機時間がある場合、またはデータ ポイントの 90% にゼロ ベースを超える KCU 待機時間がある場合、KCU の健全性は赤色になります。

クラスタ

このビューには、クラスタ レベルでの Kofax RPA システムのパフォーマンス パラメータがグラフィックで表示され、各 RoboServer のグラフも表示されます。RoboServer の起動時に `-serverName` パラメータで RoboServer の名前を指定した場合、ビューには RoboServer 名が表示されます。それ以外の場合は、RoboServer の IP アドレスが表示されます。

RoboServer の名前の色は、最後のアクティビティに応じて次のように変化します。

- 緑: 値が 60 秒を超える場合。
- オレンジ: 値が 60 秒以上で、5 分より小さい場合。
- 赤: 値が 5 分以上で、1 日より小さい場合。
- グレー: 値が 1 日以上の場合。

リストで RoboServer を選択することで、結果をフィルタリングできます。RoboServer を選択すると、[CPU] と [メモリ] のグラフに、最小、最大、平均の使用状況が表示されます。

プロジェクト

このビューには、主要なロボット実行の指標がグラフィックで表示され、プロジェクトのロボットの健全性に関する概要が示されます。想定どおりに動作しないロボットを特定し、修正することができます。このビューでは、次の列と行が使用できます。

フィールド名	説明
プロジェクト	
プロジェクト	プロジェクト名を表示します。
スケジュール済み	選択した期間中の、プロジェクトでスケジュールされたロボット実行の数。
API コール	選択した期間中の API コールの数。
健全性	プロジェクトの全体的な健全性を表すカラー サークル。詳細については、ロボットの健全性と色 を参照してください。
ロボット	
ロボット	選択したプロジェクトのロボット名とパス。
実行時間	1 回のロボット実行の平均実行時間。
合計短縮時間(分)	選択した期間中の手動および自動 (ロボットによる) タスク実行時間の推定差 (分単位)。
抽出レコード	1 回のロボット実行中に抽出されたレコードの平均数。
KCU	1 回のロボット実行中に使用された KCU ポイントの平均数。
エラー	1 回のロボット実行での平均エラー数。
タグ	
タグ	ロボットのタグの名前。タグをクリックすると、選択したタグを含むすべてのロボットのリストが開きます。

ロボットの健全性と色

ロボットの健全性は、エラー数、実行時間、抽出されたレコード数という 3 つの値に基づいて計算されます。選択した期間のロボット実行ごとに、平均と偏差が計算されます。すべての値が平均プラス/マイナス偏差の範囲に収まっている場合、ロボット実行は緑色になります。それ以外の場合、ロボット実行は赤色になります。

ロボットのすべてのロボット実行が緑色の場合、ロボットの健全性は緑色になります。すべてではなく、1 つ以上のロボット実行が赤色の場合、ロボットの健全性は黄色になります。すべてのロボット実行が赤色の場合、ロボットの健全性は赤色で表示されます。

ロボットのリストの行をクリックすると、実行時間、抽出レコードの数、エラーと警告、ロボットの実行の詳細など、選択したロボットの情報が表示されます。

スケジュール

このビューには、システムでの負荷分散の概要を示す、同時スケジュール実行の数がグラフで表示されます。

スケジュール実行の詳細を表示するには、スケジュールを選択し、[スケジュール] ペインで [詳細] をクリックします。[スケジュールの詳細] ウィンドウが表示されます。[スケジュール実行] リストでスケジュール実行をクリックして、[ロボット実行] で詳細を表示します。

スケジュールの健全性と色

スケジュールの健全性は、エラー数、実行時間、抽出レコード数という 3 つの値に基づいて計算されます。選択した期間のスケジュール実行ごとに、平均と偏差が計算されます。すべての値が平均プラス/マイナス偏差の範囲に収まっている場合、スケジュール実行は緑色になります。それ以外の場合、スケジュール実行は赤色になります。

スケジュールのすべてのスケジュール実行が緑色の場合、スケジュールの健全性は緑色になります。すべてではなく、1 つ以上のスケジュール実行が赤色の場合、スケジュールの健全性は黄色になります。すべてのスケジュール実行が赤色の場合、スケジュールの健全性は赤色で表示されます。

ヒートマップ

RPA ダッシュボードには、ロボット実行とスケジュール実行に関する色分けされたヒートマップが表示されます。ヒートマップのしきい値は、レポートの上にある [重大ステータスのしきい値の入力] オプションで指定します。レポートの上にマウス カーソルを合わせると、実行数がセルに表示されます。以下の色が利用できます。

色	値
赤	指定したしきい値を超過
オレンジ	指定したしきい値の 4 分の 3 を超過、指定したしきい値以下
黄	指定したしきい値の半分を超過、4 分の 3 以下
緑	指定したしきい値の半分以下

使用量

ライセンスの使用状況とアプリケーションの使用量のほか、クラスタごとに整理されたロボットの実行情報が表示されます。

アプリケーションの使用量

プロジェクト
利用可能なプロジェクトを表示します。

ロボットで見つかったタグ
選択したプロジェクト内のロボットに割り当てられたタグを表示します。各行には、タグが付いているロボットの数が表示されます。タグ名をクリックすると、そのタグが割り当てられたロボットが表示されます。

ロボットで使用されているアプリケーション
選択したプロジェクト内のロボットによって呼び出されたアプリケーションを表示します。各行には、アプリケーションを実行したロボットの数が表示されます。アプリケーション名をクリックすると、ロボット名が表示されます。

ロボットで使用される **URL**
選択したプロジェクト内のロボットがアクセスするドメイン URL を表示します。各行には、URL にアクセスしたロボットの数が表示されます。URL をクリックすると、ロボットがアクセスした具体的なページのリストが表示されます。

ライセンス使用状況

選択した Management Console で使用されているライセンスに応じて、レポートには KCU または CRE 使用量の情報が表示されます。

KCU 使用量
クラスタごとの利用可能な最大ライセンスのうち、KCU を使用して実行されたロボットの割合。

KCU 使用量のピーク
選択した期間の KCU 使用量のピークを表示します。

KCU 待機時間のピーク**(**秒**)**
選択した期間の KCU 待機時間のピークを表示します。

KCU 待機時間の累積
選択した期間の合計 KCU 待機時間。

CRE 使用量
利用可能な最大ライセンスのうち、CRE ライセンスを使用する同時ロボット実行の割合。

CRE 使用量のピーク
選択した期間の同時ロボット実行のピークをクラスタ別に表示します。

キュー時間
使用可能なライセンスを得るためにロボットがキューに格納されていた平均時間。

キューに格納されたロボット
キューに格納されたロボットの最大数。

ロボットの実行

ロボットの実行の累積
ロボット実行の合計数。

ロボットの実行時間の累積
ロボットの合計実行時間。

ロボットのエラー
ロボットのエラーの合計数。

キュー時間 (秒)
ロボットがキューに格納されていた合計時間。

Process Discovery ビュー

Process Discovery ビューは、Process Discovery Agent が収集し、Analyzer で処理されたデータに基づく情報を含むレポートです。

フィルタとステータス
- [グループの選択]: このフィルタを使用して、選択した Process Discovery グループのデータを表示します。すべてのグループのデータを表示するには、[すべてのグループ] を選択します。
- [日付]: 日付範囲を指定します。デフォルトでは、範囲は過去 30 日です。
- [一般的なステータス]: レポートの現在のステータスを表示します。「Analytics レポートは <日付と時刻> に生成されました」と表示された場合、Analyzer は分析を正常に終了しており、データを評価する準備が整っています。その他のステータスである場合は、レポートが未完了であることを示します。詳細については、[ステータス] ビューを開いてください。

Process Discovery ビューでは、次のチャートとグラフを使用できます。

概要

Analyzer によって処理されたすべてのデータの概要が表示されます。このビューでは、次の情報を利用できます。

概要
- [すべてのユーザー アクティビティの期間 (時間)]: 検出されたユーザー アクティビティの合計期間。
- [プロセスの期間 (時間)]: 検出されたプロセスの期間。
- [プロセス以外のユーザー アクティビティの期間 (時間)]: プロセスを形成しないユーザー アクティビティの期間。
- [ユーザー数]: 分析されたユーザーの合計数。
- [プロセス内のユーザーの数]: プロセスを形成するアクションを実行したユーザーの数。
- [アプリケーションの数]: 検出されたアプリケーションの合計数。
- [プロセス内のアプリケーションの数]: プロセス内で使用されたアプリケーションの数。
- [イベント数]: 処理されたイベントの数。

アプリケーションあたりのユーザー アクティビティの合計期間 (時間)
この棒グラフは、各アプリケーションでの作業の所要を検出するために役立ちます。

収集されたデータの合計量
このグラフには、分析されたイベントの合計数に関する情報が表示されます。

プロセス

このビューには、検出されたプロセスに関する情報が表示されます。ウィンドウの左下隅にある [**Excel** にエクスポート] をクリックすると、検出されたプロセスのリスト、プロセス インスタンスのリスト、お

よびプロセス インスタンスのステップのリストをスプレッドシートにエクスポートできます。また、[プロセスの詳細] ウィンドウおよび [プロセスのインスタンスの詳細] ウィンドウで [**PDF** にエクスポート] をクリックすると、プロセス インスタンスのリストとスクリーンショットを含むステップのリストを PDF ファイルにエクスポートできます。

概要

検出されたプロセスの概要を表示します。

- [プロセスの合計期間 **(**時間**)**]: 検出されたすべてのプロセスの期間。
- [プロセスの数]: 検出されたプロセスの数。
- [プロセス内のアプリケーションの数]: プロセス内で使用されたアプリケーションの数。
- [プロセス内のユーザーの数]: プロセスを形成するアクションを実行したユーザーの数。

[アプリケーションあたりのプロセス内のユーザー アクティビティの期間 **(**時間**)]**

チャートには、プロセス内の各アプリケーションでの作業の所要時間が表示されます。

[プロセスあたりの平均時間 **(**分**)]**

チャートには、プロセス内のプロセス インスタンスの平均継続期間が表示されます。チャートの右側にあるプロセス名をクリックして、チャート内のプロセスを表示または非表示にできます。

[プロセスの期間 **(**時間**)]**

チャートには、検出されたプロセスの合計時間が表示されます。チャートの右側にあるプロセス名をクリックして、チャート内のプロセスを表示または非表示にできます。

[発見されたプロセス**]**

発見されたプロセスを一覧表示します。各プロセスに対して、次の情報が表示されます。プロセスの詳細を表示するには、リスト内のプロセスをクリックします。

- [タイトル]: プロセスの名前。プロセス名は自動的に作成され、「Process」という単語と序数で「Process 22」などのように構成されます。
- [最も使用されているアプリケーション]: プロセスで最も使用されているアプリケーションの名前。
- [実行の量]: 検出されたプロセス インスタンスの数。
- [合計時間 **(**時間**)**]: プロセスのすべてのインスタンスの合計期間。
- [一致率 **(%)**]: プロセス インスタンスの平均一致率。
- [検出タイプ]: プロセスが検出された方法を示します。[自動] または [補助あり] のいずれかになります。ユーザー アシスタンスを使用して 1 つ以上のプロセス インスタンスが検出された場合、検出タイプは [補助あり] と表示されます。ユーザーが Agent のメニューの [タスクの開始] オプションおよび [タスクの停止] オプションを使用してタスクの開始と終了を指定した場合、[補助あり] タイプのプロセスが表示されます。
- [平均時間 **(**分**)**]: プロセス インスタンスの平均継続期間。
- [ステップ]: プロセス インスタンスの平均ステップ数。
- [ユーザー]: 選択したプロセスを実行したユーザーの数。

  [プロセスの詳細**]**

  [発見されたプロセス] リストでプロセスをクリックすると、[プロセスの詳細] ウィンドウが表示されます。ウィンドウには、選択したプロセスの詳細情報とプロセス インスタンスのリストが表示されます。各プロセス インスタンスに対して、次の情報が表示されます。列のタイトルをクリックして、プロセス インスタンスのリストを並べ替えることができます。

  - [インスタンス]: インスタンス番号。
  - [一致率 **(%)**]: 各インスタンスの一致率。

- [検出タイプ]: プロセス インスタンスの検出タイプ。
- [ステップ]: 検出されたプロセス インスタンスのステップ数。
- [期間 (分)]: プロセス インスタンスの期間。

[プロセスのインスタンスの詳細]

[プロセスの詳細] ウィンドウでプロセス インスタンスをクリックすると、[プロセスのインスタンスの詳細] ウィンドウが表示されます。このウィンドウには、選択したプロセス インスタンスに関する詳細情報と、入手可能な場合はスクリーンショット付きのステップのリストが表示されます。

ステータス

このレポートには、Analyzer イベントと Agent のステータスのリストが表示されます。このレポートを使用して、一般的なステータスが「Analytics レポートは <日付と時刻> に生成されました」の内容と異なる場合に、Analyzer で発生したエラーをトレースします。

[**Agents** 記録のステータス] リストを使用して、エージェントのアクティビティを追跡できます。Agent は日付順に並べ替えられ、最新のアクティビティが一番上に表示されます。このリストは、オフになっている Agent 、または正しくセットアップされていない Agent を検出するのに役立ちます。

第 9 章

Kapplets

Kofax RPA Kapplets は、ロボットを実行するためのシンプルで直感的な Web インターフェイスを提供するアプリケーションです。ロボットやその構築方法に関する知識がなくても、Kapplets を使用してロボットを操作することができます。

Kapplets の Web ブラウザ インターフェイスへのアクセス権は管理者によって提供され、これらのアクセス権はロールに基づいています。ログインするときは、管理者によって割り当てられたロールと資格情報を使用してください。

関連項目: Kofax RPA の制限。

Kapplets へのログイン

初めてブラウザで Kapplets を開いたときは、Management Console を使用して Kapplets を認証するための共有シークレットを入力する必要があります (インストール中にすでに設定されている場合を除きます)。

この操作を行うには、次の手順を完了させます。

1. **Management Console** > **[管理]** > **[サービス認証]** に移動します。
2. Kapplets のコンテキスト メニューから、**[共有シークレットを表示]** を選択して、共有シークレットをコピーします。
3. Kapplets ページで、**[管理者の場合は、ここをクリックして OAuth Kapplets シークレットを入力してください]** をクリックします。
4. コピーした共有シークレットを入力し、**[更新]** をクリックします。
5. **[Management Console にログイン]** をクリックします。

Management Console で共有シークレットが変更されている場合は、Kapplets のログイン ページで共有シークレットを更新する必要があります。

共有シークレットを設定した後、次に Kapplets にログインするときは、次のようになります。

- Kapplets を開いて、**[Management Console にログイン]** をクリックします。Management Console のログイン ページにリダイレクトされます。

  クレデンシャルを入力して **[ログイン]** をクリックすると、自動的にリダイレクトされ、Kapplets にログインします。
- すでに Management Console にログインしている場合は、Kapplets を開くと、自動的に Kapplets にログインします。

Kapplet ユーザー管理

Kofax RPA Kapplets セキュリティ モデルはロールベースです。ユーザーは、Management Console プロジェクト内のリポジトリへの特定のアクセス権と、Kapplet、テンプレート、スケジュール、およびワークスペースを管理するための Kofax RPA Kapplets 内の特定の権限を持っている必要があります。

たとえば、Management Console で [Kapplet ユーザー] ロールが割り当てられている場合、Kapplets インターフェイスでは、管理者権限を持っている場合でも、行えるのは Kapplet の実行だけです。しかし、テンプレートを作成するには、[Kapplet 管理者] ロールが割り当てられている必要があります。これら 2 つのロールにより、Management Console リポジトリへのアクセスが制限されます。

ユーザーは、Management Console で設定されたクレデンシャルを入力して、Management Console を使用して Kofax RPA Kapplets にログインします。最初に、管理者はユーザーとグループを作成し、Management Console のプロジェクトに権限を割り当てる必要があります。したがって、後で Kofax RPA Kapplets で作業するユーザーは、ロボット、タイプなどが含まれる Management Console プロジェクト内のリポジトリにアクセスするための特定の権限を取得します。

これらのユーザーは、次の 2 つのユーザー ロールを使用できます。

ロール	説明
Kapplet 管理者	このロールを持つユーザーは、Kapplets から Management Console のプロジェクトへの読み取り/書き込みアクセス権を取得します。Kofax RPA Kapplets では、このロールを持つユーザーは Kapplet を管理し、これらのテンプレートに必要なロボットを含むプロジェクトの Kapplet テンプレートを作成および管理できます。
Kapplet ユーザー	このロールを持つユーザーは、Kapplets から Management Console のプロジェクトへの読み取り専用アクセスを取得します。Kofax RPA Kapplets では、このロールを持つユーザーは、アクセス権のあるプロジェクトに属するロボットの Kapplets のみを表示および実行できます。

アクセス権のないプロジェクトに属するロボットが Kapplet に少なくとも 1 つ含まれている場合は、警告アイコンと、この Kapplet が無効になっていることを示すツールチップが表示されます。つまり、Kapplet を実行するには、これらのロボットが含まれているすべてのプロジェクトに対する権限を持っていること、およびロボットが Management Console リポジトリ内に存在することが必要となります。

Management Console でユーザー、グループ、およびプロジェクトの権限を設定するには、管理者が次の手順を使用する必要があります。

1. Management Console に管理者としてログインし、ユーザーとグループを作成します。ユーザーをグループに追加します。
2. **[プロジェクト]** セクションでプロジェクトを作成 (または既存のプロジェクトを編集) し、[権限] タブでセキュリティ グループにプロジェクト ロールを割り当てます。

   [Kapplet 管理者] または [Kapplet ユーザー] を選択します。

   「プロジェクト」および 「ユーザーおよびグループ」も参照してください。

Kapplet はユーザーの権限に応じてユーザーによって構築およびメンテナンスされ、Kapplets 側のグローバル管理者によって割り当ておよび変更されます。Management Console でユーザーを作成した後に、管理者はユーザーとグループにグローバル権限とカスタム権限を割り当てる必要があります。

管理者は、Management Console を介してスーパーユーザー (デフォルト名: `admin`、パスワード: `admin`) として Kapplets にログインし、次のオプションのいずれかを実行する必要があります。

- 最初に、ユーザーに Kapplets にログインするよう依頼します。これらのユーザーとそのグループのデータは、Management Console と Kapplets の間で同期されます。

  新しいユーザーが Kapplets にログインするとアクセスは拒否されますが、Kapplets に新しいユーザー アカウントが作成されます。

  管理者は、これらのユーザーに対して適切な権限を割り当てることができ、それによりユーザーは Kapplets で作業できるようになります。
- [ユーザー グループ] ページに移動し、左上隅にある + 記号をクリックして、Management Console のグループと同じ名前のグループを作成します。

  作成した各グループに権限を割り当てます。

Kapplets では、次の 2 種類の権限が利用できます。

- グローバル権限: システム全体のレベルで管理され、個々のユーザーまたはユーザーのグループに割り当てることができ、各ユーザーが権限を継承します。

権限	説明
なし	Kapplets アプリケーションにアクセスすることはできません。
ユーザー	Kapplet を実行し、これらの Kapplet のスケジュールを作成して、作成されたスケジュールの結果のみを表示できます。
開発者	Kapplet とテンプレートを作成、実行、編集、および削除します。スケジュールを作成し、任意のユーザーが作成したすべてのスケジュールの結果を表示できます。
管理者	Kapplet、テンプレート、ワークスペースを管理し、スケジュールを作成します。任意のユーザーが作成したすべてのスケジュールの結果を表示できます。 権限を割り当てます。

- カスタム権限: 特定のワークスペースのユーザー グループにのみ割り当てられます。ユーザーはユーザー グループからこれらの権限を継承します。

ユーザーとこのユーザーが属するグループに異なる権限が付与されている場合、ユーザーは最も広範な権限を取得します。

以下に、ユーザーが Kapplet で作業するために必要なロールと権限の組み合わせの例を示します。

既存の Kapplet を実行するには、ユーザーは次のことを行う必要があります。

- Management Console で、この Kapplet で使用されるロボットを含むプロジェクトに対する Kapplet ユーザー ロールを持ちます。
- Kapplets では、最低限の権限としてユーザー グローバル権限を持っています。

テンプレートを作成するには、ユーザーは次のことを行う必要があります。

- Management Console で、Kapplet で使用されるロボットを含むプロジェクトに対する Kapplet 管理者ロールを持ちます。
- Kapplets では、最低限の権限として開発者のグローバル権限を持っています。

詳細については、ユーザーとユーザー グループを参照してください。

バックアップ、復元、移行

Kapplets は、Management Console に存在するロボットに依存します。ほとんどの場合、Kapplets のバックアップまたは復元は、対応する Management Console のバックアップまたは復元に依存します。データが利用可能であることを確認するには、次のシーケンスを使用します。

- 最初に、Management Console をバックアップまたは復元します。
- Kapplets によって Management Console のデータが更新されるまで、5 ~ 10 分間 (または Kapplets サーバーでの設定に従って) 待ちます。
- Kapplets をバックアップまたは復元します。

Kapplets のバックアップ、復元、移行を行う場合は、次の Kofax RPA メソッドを使用できます。以前のバージョンからデータを移行する場合は、指定されたバージョンのシーケンスと手順に従ってください。

バージョン 11.3.0 以降からの復元

11.3.0 以降のバージョンで作成されたバックアップは、**Kapplets** > **[バックアップ]** タブから復元できます。「バックアップ」を参照してください。

バージョン 11.0.0、11.1.0、および 11.2.0 からの復元

これらのバージョンでは、Kapplets のバックアップ機能はなく、Management Console の標準バックアップ ファイルには Kapplets データは含まれていません。

バージョン 11.0.0、11.1.0、および 11.2.0 で作成された Kapplets は、最初にバージョン 11.3.0 に移行する必要があります。将来の Kapplets バージョンで使用できる Kapplets バックアップ ファイルを取得するには、既存の Kapplets データベースを削除せずに Kapplets をバージョン 11.3.0 にアップグレードします。

あるバージョンから別のバージョンに段階的に復元することなく、以前のバージョン (11.0.0 ~ 11.2.0) から 11.3.0 に直接移行できます。11.3.0 より前のバージョンから 11.4.0 以降のバージョンに直接移行することはできません。

SQL サーバーと Kofax RPA を同時にアップグレードまたは移行しないでください。制約エラーが発生する可能性があります。代わりに、次のシーケンスを使用します。

- バージョン 11.3.0 への Kapplets のアップグレード。
- **Kapplets** > **[バックアップ]** タブからバックアップを作成します。
- SQL サーバーをアップグレードまたは移行します。

- Kapplets バージョン 11.4.0 以降をインストールします。
- 作成したバックアップを使用して復元します。

バージョン 11.0.0 ~ 11.2.0 で作成された Kapplets をアップグレードするには、次の手順を実行します。

1. 既存のデータベースをバックアップして、必要に応じて復元バージョンを作成します。(データベース ベンダーのガイドラインに従ってください。)
2. Kofax RPA バージョン 11.3.0 の Kapplets に切り替えて、既存のデータベースを使用するように設定します。

   `kapplets.xml` ファイルでデータベースの URL を設定する必要があります。`kapplets.xml` ファイルの例については、『Kofax RPA インストール ガイド』の「Kapplets のインストール」>「Tomcat サーバーのインストール」を参照してください。

   一部の入力パラメータとその値は、スケジュールされたレガシー Kapplets を含むバックアップからは復元されません。

   Kapplets では、セッション、国、通貨、言語フィールドの復元はサポートされていません。

3. バージョン 11.3.0 では、**[管理]** > **[バックアップ]** > **[バックアップの作成]** を選択してバックアップ ファイルを作成します。
4. バックアップ ファイルを使用して Kapplets をバージョン 11.4.0 以降に復元するには、**[管理]** > **[バックアップ]** を選択し、ターゲットの 11.3.0 バックアップ ファイルを選択します。詳細については、バックアップを参照してください。

バージョン 10.6.0 および 10.7.0 からの復元

バージョン 10.6.0 および 10.7.0 では、Kapplets と Management Console の個別のバックアップ機能は使用できません。代わりに、Kapplets のテンプレート、設定、プロジェクトなどがすべて、Management Console の標準バックアップ ファイルに含まれています。

バージョン 10.6.0 および 10.7.0 で作成されたレガシー バックアップは、次の方法で復元できます。

1. **[管理]** > **[バックアップ]** > **[バックアップの復元]** ボタンを使用して、バックアップ ファイルを RPA Management Console に復元します。

   レガシー バックアップは、Management Console で最初に復元された場合にのみ、Kapplets で正しく復元されます。
2. **Kapplets** > **[バックアップ]** タブから Management Console バックアップ ファイルを復元します。
3. プロジェクト、テンプレート、ユーザーを設定します。
   - ユーザーも、ユーザー グループ内で Kapplets に移動されます。これらのユーザー グループおよびユーザーには、グローバル管理者が権限を割り当てるまで、権限が付与されません。
   - Management Console バックアップのオブジェクトは、次のように Kapplets オブジェクトに移動されます。

Management Console バックアップ オブジェクト	**Kapplets** オブジェクト
プロジェクト	ワークスペース
Master Kapplet	テンプレート

Management Console バックアップ オブジェクト	Kapplets オブジェクト
インストールされた Kapplet	「お気に入り」とマークされた Kapplet

Kapplets の Management Console ロールに相当するロール

この転送は、Management Console バックアップが Kapplets にインポートされた場合にのみ行われます。

Management Console グループのロール	Kapplets グループの権限
Kapplet ユーザー	ユーザー
開発者	開発者
Kapplet 管理者	開発者
プロジェクト管理者	管理者
RPA 管理者	管理者
Kapplet サービスのユーザー	なし

ユーザー インターフェース

次のトピックでは、Kofax RPA Kapplets のユーザー インターフェイスと機能を紹介します。

Kofax RPA Kapplets メイン ウィンドウを表示するには、有効でアクティブ化されたライセンスが必要です。ライセンスについては、『Kofax RPA インストール ガイド』を参照してください。

Kofax RPA Kapplets には Web ベースのユーザー インターフェイスがあります。これを使用すると、コンピュータだけではなく、スマートフォンやタブレットなどのモバイル デバイスから Kapplets を操作することができます。

メイン ウィンドウは次のユーザー インターフェイス要素で構成されています。

- メイン メニュー
- ツールバー
- ユーザー メニュー

メイン メニュー

メイン メニューは、Kofax RPA Kapplets ウィンドウの左側に配置されています。メニューを最小化してアイコンにしたり、元に戻したりする場合は、ウィンドウの左上隅にある ≡ をクリックします。

Kapplets は、8 つの機能領域に分かれています。

Kapplet

どのユーザー グループでも利用できます。Kapplet を作成、編集、および実行するには、このページを使用します。Kapplet を作成する前に、テンプレートを作成する必要があります。

テンプレート

このページは、開発者、管理者、およびグローバル管理者が使用できます。テンプレートを作成および編集するには、このページを使用します。Kapplet はテンプレートに基づいています。

スケジュール

このページは、すべてのユーザー グループが使用できます。Kapplet のスケジュールを作成および編集するには、このページを使用します。

履歴

このページは、すべてのユーザー グループが使用できます。Kapplet の実行に関する情報を確認したり、Excel にエクスポートしたりするには、このページを使用します。

ユーザー

このページは、管理者のみが利用できます。このページは、次のように Kapplets のユーザーを管理する場合に使用します。

- ユーザーに関する情報を編集する
- ユーザーにグローバル権限を割り当てる
- ユーザー グループの割り当てを変更する
- ユーザーを有効化、無効化、または除去する

ユーザー グループ

このページは、管理者のみが利用できます。このページは、次のようにユーザー グループを作成および管理する場合に使用します。

- 新しいユーザーグループを作成する
- ユーザー グループに新しいユーザーを追加する
- ユーザー グループにグローバル権限を割り当てて、ワークスペース内でローカル権限を割り当てる

ワークスペース

このページは、管理者のみが利用できます。このページは、次の用途に使用します。

- ワークスペースを作成、管理、および削除する
- ワークスペース内でユーザー グループにローカル権限を割り当てる

管理

このページは、管理者のみが利用できます。このページは、次の用途に使用します。

- Kapplets のバックアップ ファイルを作成および復元する
- レガシーの Kapplets バックアップ ファイルを復元する
- 共有シークレットを提供する

ツールバー

ツールバーは、各ページの表示領域の上部に配置されています。ツールバーに表示される要素の組み合わせは、選択したページに応じて異なる場合があります。

- 選択したページに新しいアイテムを作成するには、[新規作成] + ボタンを使用します。
- タブのアイテムを更新するには、[データの再ロード] ↻ ボタンをクリックします。
- 選択したページのアイテムを検索するには、検索ボックスを使用します。

Kapplets のほとんどのページでは、特定のタイプのアイテムがテーブルに表示されます。表示できるアイテムが多数ある場合は、複数のページに分割されます。ページの分割は、ページの下部にある [ページごとのアイテム] 設定で管理されます。

表示されたアイテムのテーブルでは、フィルタを使用できます。[設定] ボタンをクリックして、フィルタ オプションのリストを表示します。フィルタ オプションのセットは、選択したページに応じて異なる場合があります。たとえば、[**Kapplet**] ページと [テンプレート] ページには、Kapplets およびテンプレートのフィルタ基準として使用できるワークスペースのリストも表示されます。

ユーザー メニュー

ユーザー メニューには次の機能があります。

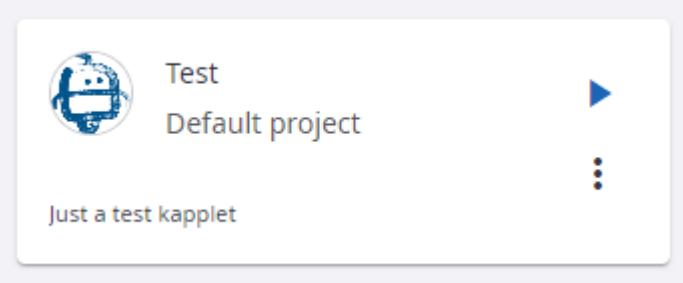
- プロファイル
  現在のユーザーに関する一般的な情報が含まれます。
- 設定
  ユーザー インターフェイスの言語設定と、すべてのカスタム設定をデフォルト値にリセットするオプションが含まれます。ユーザー インターフェイスのローカル言語設定は、管理者が割り当てた言語設定よりも優先されます。
- キーボード ショートカット
  Kapplets で使用できるキーボード ショートカットが含まれています。
- ログアウト
  クリックすると、Kapplets からログアウトできます。Management Console からログアウトするためのオプションを含むページが表示されます。

Kapplets

Kapplets はすべてテンプレートに基づいています。Kapplet は、開発者、管理者、およびグローバル管理者が作成および編集できます。

Kapplets ページまたは [テンプレート] ページのいずれかを使用して Kapplets を作成します。テンプレートがない場合は、Kapplet の前にテンプレートを作成します。Kapplet を作成すると、テンプレートとは別のオブジェクトになります。テンプレートまたは Kapplet への変更は、それらの間で共有されません。

Kapplet のオブジェクトは次のようになります。



[Kapplet] ページから作成

1 つ以上のテンプレートが存在する場合は、[**Kapplet**] ページで Kapplet を作成できます。

1. メイン メニューから [**Kapplet**] オプションを選択します。
2. **[Kapplet]** ページで、左上隅にある + 記号をクリックします。
3. [新しい **Kapplet** の作成] ウィンドウで次の情報を設定します。
   - [名前]: Kapplet の名前を入力します。
   - [テンプレート名]: ドロップダウン リストからテンプレートを選択します。
   - [ワークスペース]: ドロップダウン リストからワークスペースを選択します。
   - [説明]: 短い説明を入力して、Kapplet に関する情報を追加します。
   - [アイコン]: アイコンを追加することで、Kapplet を一意にして区別しやすくします。デフォルトでは、アイコンはテンプレートから継承されます。アイコンを変更するには、[ギャラリー] から画像を選択するか、新しい画像をアップロードします。アップロードされた画像に、他の画像と区別しやすくするためのタグを設定します。

     画像 [ギャラリー] では、`.png` および `.jpeg` ファイルのみがサポートされます。
4. ウィンドウの右下隅にある [保存] をクリックします。

[テンプレート] ページから作成

テンプレートが多数ある場合は、[テンプレート] ページから検索し、必要なテンプレートを見つけて Kapplet を簡単に作成できる場合があります。

1. メイン メニューから [テンプレート] オプションを選択します。
2. [テンプレート] ページで、新しい Kapplet に使用するテンプレートを見つけます。
   新しいテンプレートを作成するには、テンプレートを参照してください。
3. テンプレートの [新しい **Kapplet** の作成] ボタンをクリックします。
   [新しい **Kapplet** の作成] ウィンドウのフィールドの値は、テンプレートから継承されます。
4. [名前] フィールドに Kapplet の名前を入力します。
5. 説明を変更する場合は、[説明] フィールドに新しい説明を入力します。
6. ウィンドウの右下隅にある [保存] をクリックします。

7. [**Kapplet**] ページをクリックして、新しい Kapplet エントリを表示します。
   Kapplet を実行するには、Kapplet の実行を参照してください。

Kapplet でのアクションの実行

[コンテキスト メニューを開く] ⋮ ボタンをクリックして、Kapplet でアクションを実行します。

- 履歴の表示: [履歴] ページを開きます。
- お気に入りに追加: Kapplet をお気に入りに追加します。お気に入りの Kapplet のみを表示するには、[設定] アイコンをクリックし、[お気に入り] を選択します。
- お気に入りから除去: Kapplet をお気に入りから削除します。
- テンプレートに移動: Kapplet のテンプレート ページを開きます。
- テンプレートから更新: Kapplet をテンプレートから更新します。この Kapplet に関連付けられたすべてのスケジュールは、更新後に削除されます。
- 編集: [Kapplet の編集] ページを開きます。
- 削除: Kapplet を削除します。
- スケジュールの作成: [新しいスケジュールを作成] ページを開きます。スケジュールを参照してください。
- スケジュールを表示: [スケジュール] ページを開きます。

Kapplet のロボットを除去または変更すると、この Kapplet は無効になります。次の通知が表示されます。

Kapplet を有効にするには、[コンテキスト メニューを開く] から [テンプレートから更新] を選択します。

Kapplet の実行

Kapplet を実行するには、次の手順を実行します。

- Kapplet の [**Kapplet** の実行] ▶ ボタンをクリックして、[開始] ページを開きます。入力パラメータがある場合は [開始] ページで入力パラメータを設定し、ウィンドウの右下隅にある [実行] をクリックします。OAuth を使用する Kapplet を実行している場合は、Management Console に保存されている OAuth 資格情報を再利用できます。保存された OAuth 情報を使用するには、**Management Console** タブをクリックし、[**OAuth** ユーザー] で適切な資格情報を選択します。

  [実行] をクリックすると、Management Console はロボットの実行リクエストを受け取り、ロボットを開始しようとします。ライセンス、RoboServer スロットなどの必要なリソースが利用可能な場合、選択したプロジェクトに指定されたクラスタで、ロボットが開始されます。必要なリソースが利用できない場合、ロボットは 10 分のタイムアウトでキューに入れられます。10 分経過してもリソースが利用できない場合、ロボットはキューから削除され、エラー メッセージがログに記録されます。
- Kapplet が実行されると、[履歴] ページが開きます。

テンプレート

[テンプレート] ページを使用して、Kapplet のベースとして機能するテンプレートを作成および編集します。テンプレートは、開発者、管理者、およびグローバル管理者が作成できます。

新しいテンプレートの作成

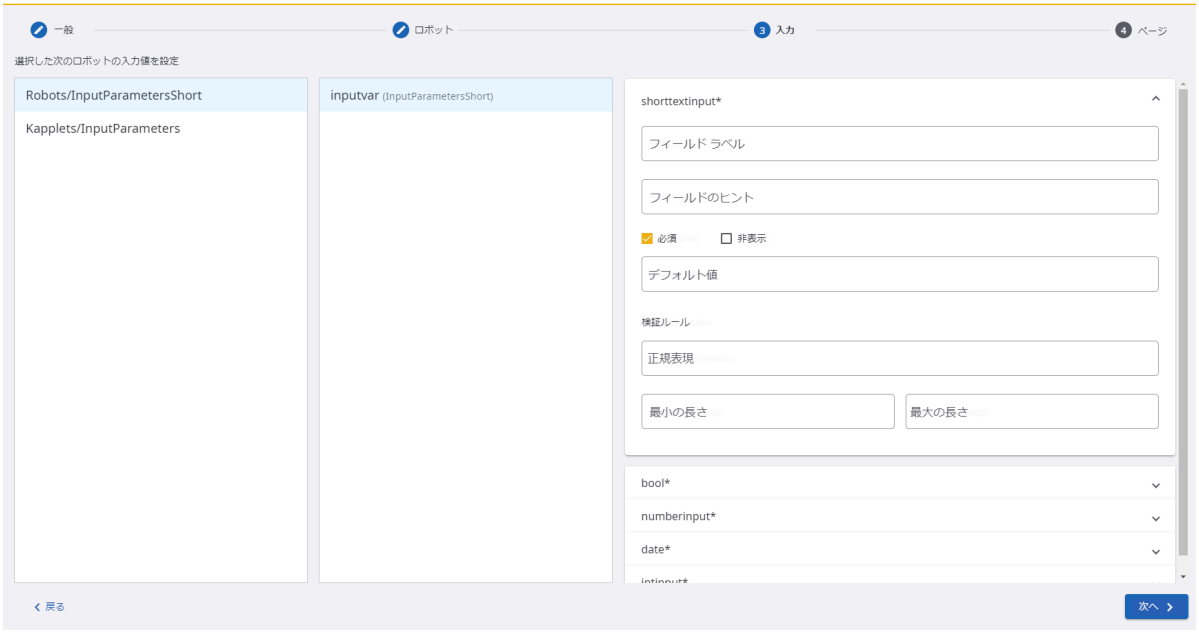
1. [テンプレート] ページに移動します。
2. 左上隅のプラス記号をクリックして新しいテンプレートを作成します。
3. [一般] ページで次の情報を設定します。
   - 名前: テンプレート名を入力します。
   - [ワークスペース]: ドロップダウン リストからワークスペースを選択します。
   - [説明]: 短い説明を入力して、テンプレートに関する情報を追加します。
   - **[タイムアウト値]** と **[時間単位]**: 希望するタイムアウト値を設定し、時間単位を選択して、ロボットが RoboServer のキューに留まる時間を指定します。このタイムアウトは、テンプレート内のすべてのロボットに適用されます。デフォルトは 600 秒です。
   - [アイコン]: アイコンを追加することで、テンプレートを一意にして区別しやすくします。デフォルトでは、アイコンはワークスペースから継承されます。アイコンを変更するには、[ギャラリー] から画像を選択するか、新しい画像をアップロードします。アップロードされた画像に、他の画像の中から特定しやすくするためのタグを設定します。

     画像 [ギャラリー] では、`.png` および `.jpeg` ファイルのみがサポートされます。

4. [次へ] をクリックします。
5. [ロボット] ページで、1 つまたは複数のロボットを選択します。
   ロボット リストが長すぎる場合は、検索フィールドを使用して必要なロボットを見つけます。
   - 複数のロボットを選択する場合は、ツールバーの [選択したロボット] ボタンをクリックして、選択したロボットを表示します。選択したロボットの番号がボタンに表示されます。
   - [選択したロボット] ペインの [順次実行] ボタンを使用して、ロボットの実行順序を定義します。
   - [設定] ペインで、Management Console プロジェクトを基準としてロボットをフィルタします。
6. [次へ] をクリックします。
7. [入力] ページで、選択したロボットの入力パラメータを設定します。

   選択したロボットのいずれかに入力パラメータが含まれている場合は、プロシージャにこのステップが追加されます。

   [入力] ページで入力パラメータと検証ルールを設定すると、それらは自動的に検証されます。エラーが返された場合は、入力パラメータおよび/または対応する検証ルールを修正してください。入力が有効な場合、[次へ] ボタンがアクティブになります。
   - ロボットを選択して、その変数タイプを表示します。
   - ロボット変数が表示されたウィンドウで変数をクリックし、その属性を設定します。
   - 変数ウィンドウの隣のウィンドウに、すべての属性がリストされます。

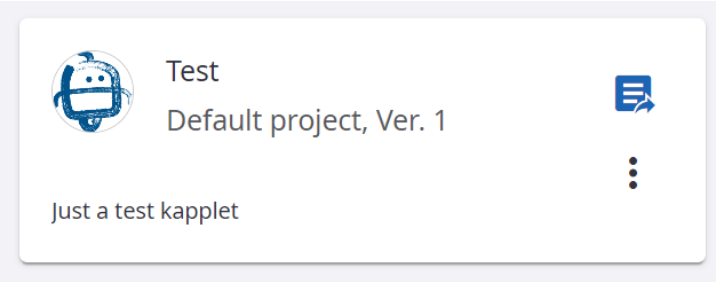


- [入力] ペインで、[開始] ページに表示する [フィールド ラベル] および [フィールドのヒント] を入力します。
- 必要に応じて [必須] および [非表示] チェック ボックスを選択して、[開始] ページにフィールドを表示する方法を設定します。
- パラメータを個別に設定します。パラメータのセットは、各属性によって異なります。

8. [次へ] をクリックします。
9. [ページ] ペインで、必要に応じて次の設定を行います。
   - [開始] ページでページ タイトルを編集し、[ページ テキスト] フィールドにページの説明を入力します。
   - [フィールドの順序] リストでアイテムをドラッグして、フィールドの順序を変更します。
   - [ステータス ページ] でページ タイトルを編集します。
   - 実行結果をテーブルに表示するには、ツールバーの [テーブル ページの追加] ≣ アイコンをクリックします。
     - [テーブル ページ] でページのタイトルおよび説明を編集します。
     - ドロップダウン リストでロボットを選択し、パラメータを設定します。[返されるタイプ] リストは、選択したロボットの出力タイプで構成されています。
     - パラメータをクリックして、テーブルのフィールドを定義します。
     - [利用可能なコミット アクション] オプションを使用して、各ロボットにアクションをコミットするためのそれぞれ異なるロボットを割り当てます。詳細については、コミット アクションのロボットの選択を参照してください。
   - 実行結果をグラフに表示するには、ツールバーの [グラフ ページの追加] ⊕ アイコンをクリックします。
     - [チャート ページ] でページのタイトルおよび説明を編集します。
     - ドロップダウン リストでロボットを選択し、グラフに表示するパラメータを設定します。

- 次のパラメータを選択します。
  - グラフのタイプ
  - 配色
  - ラベル軸
  - データ軸
- 他のグラフ パラメータを設定します。
- [プレビュー] 👁 ボタンをクリックしてグラフをプレビューします。

10. [保存] をクリックしてテンプレートを保存します。
    テンプレートのオブジェクトは次のようになります。テンプレートから Kapplet を作成するには、Kapplet の作成を参照してください。



テンプレートの編集

テンプレートを修正または削除するには、[コンテキスト メニューを開く] ⋮ ボタンをクリックします。

[入力] ページで入力パラメータと検証ルールを編集すると、それらは自動的に検証されます。エラーが返された場合は、入力パラメータおよび/または対応する検証ルールを修正してください。入力が有効な場合、[次へ] ボタンがアクティブになります。

コミット アクションのロボットの選択

2 つのロボットを連続で使用するときに、メイン ロボットの出力データを 1 つ以上のサポートされるロボットの入力データとして使用することが必要になる場合があります。サポートされる各ロボットは、1 つの入力と 1 つの出力タイプのみを持つように設計する必要があります。メイン ロボットに複数の出力タイプがある場合は、アクションをコミットするために他のロボットを割り当てます。入力と出力のタイプは、サポートされるロボットで一致する必要があります。次の手順では、ロボット間でデータを渡す方法について説明します。

1. [テンプレート] ページに移動します。
2. 新しいテンプレートを作成するか、既存のテンプレートを編集します。
3. [ページ] ペインに到達するまで構成ペインの中を進みます。
4. [返されるタイプ] リストを探します。このリストでは、選択したロボットの出力タイプを確認できます。
   アクションのコミットに利用可能なタイプと利用できないタイプは、別々のカードに表示されます。

5. 使用可能なタイプの右側にある [利用可能なコミット アクション] ペインで、ドロップダウン リストから口ボットを選択します。
   利用可能な返されるタイプに、すべての口ボットを一度に割り当てることができます。

   入力タイプと出力タイプを同じにする必要があります。使用するタイプの前に、「コミット アクションに使用できる口ボットはありません」という通知が表示されているかどうかをよく確認してください。

6. メイン ロボットを実行します。
7. メイン ロボットの実行が終了したら、[履歴] ページを選択します。
   - [テーブル ページ] タブで、前の手順で設定したサポートされる口ボットのリストを表示します。
   - [コミット アクションの選択] チェックボックスをオンにして、1 つ以上のサポートされる口ボットを選択します。
   - テーブルの右上隅にある [コミット アクション] ボタンがアクティブになったら、それをクリックして実行結果を表示します。

   同じ口ボットを 2 回実行することはできません。

スケジュール

[スケジュール] ページでは、Kapplets でのスケジュールを作成および管理できます。スケジュールは、Kapplets を実行するための計画を示したものです。毎日正午、毎週金曜日の午後 4 時 50 分など、事前定義された間隔で Kapplets を実行できます。スケジュールのリストが広範囲にわたる場合は、[検索] 入力フィールドを使用して必要な Kapplet を検索します。このページは、すべてのユーザー グループが使用できます。

デフォルトでは、スケジュールごとに次のテーブル列が表示されます。

列	説明
Kapplet の名前	Kapplet の名前。
ワークスペース	Kapplet が属しているワークスペースの名前。
次の実行	次回のスケジュール実行が計画されている時間。
スケジュール タイプ	スケジュールの連続した 2 つの実行における計画された間隔。
実行時間	前回のスケジュール実行の実行時間。
ステータス	スケジュール実行のステータス。
停止日	スケジュールが停止する日付。
所有者	スケジュールを作成したユーザー。

デフォルトでは、このリストは Kapplet 名のアルファベット順で並べられています。必要な列をクリックすることで、項目の順序を変更できます。リストからスケジュールを削除するには、スケジュールのコンテキスト メニュー ⋮ から **[削除]** をクリックします。このメニューから、スケジュールを編集、一時停止、または再開できます。

新しいスケジュールを作成

新しいスケジュールを作成するには、**[スケジュール]** ページに移動し、次の手順を実行します。

1. 左上隅の + 記号をクリックします。
2. 新しいスケジュールのワークスペースと Kapplet を選択します。[次へ] をクリックします。
3. [スケジュール] の間隔を選択します。
   選択内容に応じて、スケジュールを適切に設定するように新しいフィールドが動的に表示されます。可能なオプションは、次のとおりです。
   - [スケジュール]: スケジュールの間隔。
   - [スケジュールされた最初の日]: スケジュールを開始する必要がある ローカル時間。
   - [トリガー日時]: 間隔が「毎週」および「毎月」の場合に利用できます。スケジュールされた実行が開始される時間。
   - [**Kapplet** の実行頻度]: 間隔が「毎週」および「毎月」の場合に利用できます。スケジュールの実行を開始する曜日または日付を選択します。
   - [パターン]: 「cron」スケジュール専用。スケジュールを実行する必要があるタイミングを定義するパターン。
   - [スケジュールされた最後の日]: 「一回のみ」以外の間隔が設定されているすべてのスケジュールなど、すべての定期的なスケジュールに対して利用できます。
   - [期間]: 間隔が「毎時」の場合に利用できます。スケジュールの連続した 2 つの実行間の希望の間隔 (時間単位)。1 と 23 の間の整数値を入力します。
   - [実行に失敗した場合]: スケジュールの実行に失敗した場合に実行するアクションを選択します。
     間隔が「一回のみ」のスケジュールでは、[なし] のみを選択できます。
     「一回のみ」以外の間隔のすべてのスケジュールでは、リストされているアクションのいずれかを選択できます。ただし、[なし] を選択すると、すべての一時停止アクションが利用できなくなり、1 つまたは複数の一時停止アクションを選択すると、[なし] が利用できなくなります。
     - なし
       実行の失敗を無視してスケジュールを実行します。
     - 失敗した場合は一時停止
       実行の開始または終了に失敗した場合、スケジュールを一時停止します。
     - タイムアウトした場合は一時停止
       ロボットの実行結果にタイムアウトの実行が含まれている場合、スケジュールを一時停止します。
     - エラーで終了した場合は一時停止
       実行が終了したが、エラーで完了したロボットが含まれている場合、スケジュールを一時停止します。
4. [入力パラメータ] ペインでは、ロボットの入力パラメータを入力できます。
5. [保存] をクリックしてスケジュールを保存します。

別のスケジュール作成方法

[**Kapplet**] ページからスケジュールを作成することもできます。この操作を行うには、Kapplet を選択し、コンテキスト メニューから **[スケジュールの作成]** をクリックします。

スケジュールを表示

Kapplet にスケジュールが割り当てられている場合は、Kapplet ページの [Kapplet] ボックスに表示されます。この場合は、 [コンテキスト メニューを開く] ⋮ ボタンをクリックし、[スケジュールを表示] を選択すると、この Kapplet に関連付けられている既存のすべてのスケジュールのリストを表示できます。

履歴

[履歴] ページでは、Kapplet に関する実行情報を表示できます。テーブル内の Kapplet の名前をクリックして、実行情報を開きます。このページは、すべてのユーザー グループが使用できます。

デフォルトでは、エンティティごとに次のテーブル列が表示されます。

列	説明
アイコン	Kapplet のアイコン。
名前	Kapplet の名前。
日付	Kapplet の実行が現在の状態になった日時。
状態	Kapplet の実行の状態。
作成者	Kapplet を作成したユーザー。

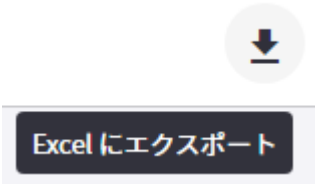
[設定] の実行状態を使用して実行情報をフィルタできます。デフォルトでは、設定のすべての状態が選択されています。

画像	名前	説明
	進行中	Kapplet が実行中です。
	失敗	Kapplet の実行が開始されましたが、完了できませんでした。
✓	終了	Kapplet の実行は正常に完了しました。
⚠	エラーで終了	Kapplet の実行は完了しましたがエラーが発生しました。
	タイム アウト	Kapplet の実行結果には、タイム アウトした実行が含まれます。
×	キャンセル	Kapplet の実行が停止されました。

Kapplet の結果を Excel にエクスポート

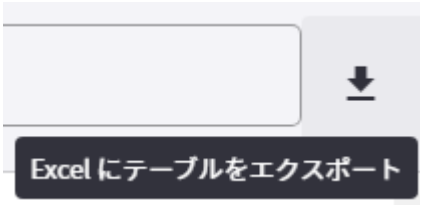
[履歴] ページから、Kapplet の実行結果を Excel ファイルにエクスポートできます。これを行うには、実行のリストで、正常に実行された Kapplet をクリックします。実行が成功し、エクスポートするテーブル データがある場合にのみ、ダウンロードできます。

- 実行結果全体をエクスポートするには、右側のダウンロード ボタンをクリックします。



Kapplet の実行結果に複数のテーブルがある場合、それらのテーブルは、作成された Excel ファイルでは個別のタブとして表示されます。

- 複数のテーブルがある場合に特定のテーブルをダウンロードするには、それぞれの [テーブル ページ] タブで、下のダウンロードボタンをクリックします。



ユーザーとユーザー グループ

これらのセクションは、管理者が利用できます。

ユーザー

[ユーザー] ページを使用して、Kapplets アプリケーションにアクセスできるユーザーを表示および管理します。

管理者はシステムによって自動的に作成されるデフォルトのユーザーであり、管理者を削除することはできません。

ユーザーが Kapplets にログインすると、Kapplets の [ユーザー] ページの表に表示されます。ユーザーの設定方法については、Kapplet ユーザー管理を参照してください。

列	説明
有効	⏻ をクリックして、Kapplets へのユーザー アクセスを有効または無効にします。
フル ネーム	ユーザーのフル ネーム。
ユーザー名	Management Console を使用して Kapplet にログインするために使用されるユーザー名。
電子メール	ユーザーの電子メール。
タイプ	ユーザーのタイプ。「Management Console」は、ユーザーが Management Console で設定されていることを意味します。
グループ	ユーザーがアクセスできる Management Console グループのリスト。

列	説明
権限	Kapplets のグローバル権限がユーザーに割り当てられます。

[ユーザー] に対するアクション

ユーザーの ⋮ コンテキスト メニューをクリックすると、次のアクションが一覧表示されます。

- 編集: ダイアログ ボックスが開き、ユーザーにグローバル権限を割り当てたり、ユーザーを有効または無効にしたり、このユーザーを利用可能なグループに割り当てることができます。
- [削除]: リストからユーザーを削除します。

ユーザー グループ

[ユーザー グループ] ページを使用して、ユーザー グループを作成および編集し、グローバル権限とカスタム権限を割り当てます。

列	説明
名前	ユーザー グループの名前。
タイプ	グループのタイプ。
権限	このグループに割り当てられた Kapplets のグローバル権限。
説明	グループの説明。

デフォルトでは、次の 3 つのグループが利用可能です。

- 英語 UI
- フランス語 UI
- 日本語 UI

ユーザーがこれらの言語のいずれかで Kapplet ユーザー インターフェイスを表示できるようにするには、必要なユーザー グループを選択し、コンテキスト メニューから [編集] をクリックし、[グループを編集] ページでユーザーを選択して、変更を保存します。次にこのユーザーが Kapplet にログインすると、選択した言語でユーザー インターフェイスが表示されます。

新しいユーザー グループの作成

新しいユーザー グループを作成するには、**[ユーザー グループ]** ページに移動し、次の手順を実行します。

1. 左上隅の + 記号をクリックします。
2. **[**新しいグループを作成**]** ウィンドウで次の情報を設定します。
   - 名前: グループの名前を入力します。これは、Management Console で作成されたセキュリティ グループの名前と一致している必要があります。
   - [説明]: 必要に応じてグループの説明を追加します。
   - グローバル メンバーの権限グループのグローバル権限を選択します。
   - [ユーザー]: ユーザーをグループに追加します。
3. 保存] をクリックします。

さらに、コンテキスト メニューの ⋮ ボタンをクリックすると、グループに対して次のようなアクションを実行できます。

- 編集: [グループを編集] ページを開きます。
- [削除]: ユーザー グループをリストから削除します。

詳細については、Kapplet ユーザー管理を参照してください。

Kapplets ロールの転送

以前の製品リリースでユーザーに割り当てられていた Kapplets に関連するすべての Management Console ロールは、Management Console のバックアップを Kapplets にインポートすることで、Kapplet に転送できます。バックアップのロールは、Kapplet で同等のロールに変換されます。

また、Kapplets はスタンドアロン アプリケーションであるため、ユーザーに Management Console の Kapplet に関連するロールが割り当てられている場合、ユーザーが Kapplets にログインしても自動的に転送されません。新しいロールまたはロールの変更は、グローバル管理者によって明示的に確認される必要があります。つまり、Kofax RPA 側のユーザーに割り当てる必要があります。

ワークスペース

[ワークスペース] ページでは、ワークスペースを作成および編集できます。ワークスペースは、意味、および部門のニーズに基づいて Kapplet とテンプレートを 1 か所にまとめる論理グループとして機能します。ワークスペースは、管理者およびグローバル管理者が作成できます。

新しいワークスペースの作成

ワークスペースを作成するには、[ワークスペース] ページに移動して、次の手順を実行します。

1. 左上隅の + 記号をクリックします。
2. [新しいワークスペースの作成] ウィンドウで次の設定を行います。
   - [名前]: ワークスペースの名前を入力します。
   - [説明]: 短い説明を入力して、ワークスペースに関する情報を追加します。
   - [アイコン]: アイコンを追加することで、ワークスペースを一意にして区別しやすくします。[ギャラリー] から画像を選択するか、新しい画像をアップロードします。アップロードされた画像に、他の画像と区別しやすくするためのタグを設定します。

     画像 [ギャラリー] では、`.png` および `.jpeg` ファイルのみがサポートされます。

3. 変更を有効化するには、[保存] をクリックします。

ページの [設定] ペインに、使用可能なワークスペースのリストが表示されます。たとえば、カスタム権限を持つユーザーには、[**Kapplet**] ページの [設定] ペインにワークスペースのリストが表示されます。

テンプレートと Kapplet を追加できるのは、[テンプレート] ページと [**Kapplet**] ページのワークスペースのみです。

ワークスペースでのアクション

ワークスペースの ⋮ コンテキスト メニューをクリックすると、次のアクションが一覧表示されます。

- 編集: [ワークスペースの編集] ページを開きます。
- [削除]: ワークスペースと、このワークスペースに関連付けられている Kapplet、テンプレート、スケジュールを削除します。

管理

Kofax RPA Kapplets [管理] ページで、Kapplets バックアップ ファイルを作成および復元し、共有シークレットを指定します。これらのタブは、管理者のみが使用できます。

バックアップ

このタブを使用して、Kofax RPA バージョン 11.3.0 以降で作成された Kapplets のバックアップ ファイルを作成および復元します。以前のバージョンについては、「バックアップ、復元、移行」を参照してください。

バックアップを作成するには、**[**バックアップの作成**]** をクリックします。

zip 形式のバックアップ ファイルがコンピュータ上に自動的に作成されます。バックアップには、Kapplets、テンプレート、スケジュール、ワークスペース、およびユーザーとグループが含まれます。バックアップには Kapplets の履歴は含まれません。

バックアップを復元するには、コンピューターから **[**ファイル選択**]** をクリックします。

以前に保存したバックアップ ファイルを含むウィンドウが表示されるので、そこで必要なバックアップ ファイルを選択して開くことができます。

共有シークレット

このタブを使用して、共有シークレットを入力します。共有シークレットにより、Kapplets のユーザーを Management Console を使用して認証し、セキュリティを強化します。

共有シークレットは、Management Console で指定されたものと一致する必要があり、Management Console の サービス認証 セクションからコピーできます。

シークレットが Management Console で提供されていないまたは変更されている場合、管理者がこのタブまたはログイン時の専用フィールドでシークレットを手動でコピーして Kapplets に貼り付けるまで、データは Management Console から Kapplets に転送されません。

第 10 章

リファレンス

このセクションでは、次のトピックについての情報を示します。

- ベーシック エンジン ロボット
- Excel の補足ドキュメント
- RoboServer
- プロキシ サービスの使用
- Kofax RPA の制限

ベーシック エンジン ロボット

このセクションでは、ベーシック エンジン ロボットの作成に関連するさまざまなタスクと、これらのロボットで使用されるタイプについて説明します。

ベーシック エンジン ロボットの当初の目的は、ロボットの内部に状態が配置されているステートレス Web サイトおよびアプリケーションを自動化することでした。多くのロボットは、ナビゲーション部分と抽出部分の 2 つに分けられます。

ナビゲーションは、「コンテンツがある場所に移動する」ということに関係しています。ナビゲーションは、主にページ読み込みとフォーム送信に関連する部分です。Design Studio でナビゲートする場合、通常はクリック アクションを使ってウェブ ページ間を移動します。

抽出は、「適切なコンテンツを取得する」ということに関係しています。抽出では、主にコンテンツの選択、コピー、および正規化を行います。Design Studio で抽出を行う場合、通常はタグ判定アクションを使って不要なコンテンツをスキップし、抽出アクションで変数にコンテンツをコピーして、また、コンテンツを正規化するためのデータ コンバータで正しい日付や数値の形式などの必要なフォーマットを取得します。抽出されたら、[データベース データ登録] または [値返却] アクションで値を出力します。

ステップは、アプリケーションへのログイン、Web ページからのデータの抽出、フォームまたは検索ボックスへのデータの入力、メニューの選択、および複数のページのスクロールに役立ちます。ロボットは、データベース、ファイル、API、Web サービス、およびその他のロボットにアクセスして、あるアプリケーションからデータをエクスポートし、別のアプリケーションにロードすることもできます。また、必要に応じて途中でデータを変換することもできます。

多くのロボットには、見た目の似た複数のページを読み込んだり、見た目の似た複数の表を抽出したりするためのタグ繰り返しアクションなど、上記以外のアクションが含まれています。

はじめに

次の手順は、ベーシック エンジン ロボット の作成方法を示しています。

1. Design Studio を開きます。
2. **[ファイル] > [新しいベーシック エンジン ロボット]** をクリックします。
3. ロボットの名前を指定し、ロボットを保存するプロジェクトを選択して、[次へ] をクリックします。
4. 必要に応じて、ロボットを開始する URL を入力します。開始 URL を入力すると、この URL を含むページ読込ステップがロボットに自動的に挿入されます。
5. 使用するデザイン時の実行モードを次の中から選択します。最小実行 (ダイレクト) およびスマート再実行 (フル) です。デフォルトのモードはスマート再実行です。
6. [終了] をクリックします。

ロボットの編集と実行を開始するには、アプリケーション ビューまたはツールバーの [実行の準備] をクリックして、ロボットの実行準備を行う必要があります。このアクションをクリックすると、ロボットが実行モードになり、編集中にロボットを実行できるようになります。

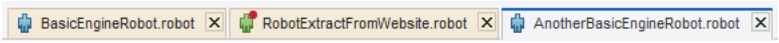
追加のソリューション、ロボット、コネクタなどを見つけるには、https://smarthub.kofax.com/ の Kofax Intelligent Automation SmartHub にアクセスしてください。

実行の準備

ロボットの完全な編集を行うと同時にロボットを実行するには、ロボットに実行権限を与える必要があります。アプリケーション ビューまたはツールバーで [実行の準備] をクリックすると、ベーシック エンジン ロボット に実行権限が与えられます。

実行権限の準備はベーシック エンジン ロボットからのみ可能ですが、[ロボットを呼び出す] ステップで [ロボットにステップ] をクリックすると、ベーシック エンジン ロボットから呼び出されるロボット に実行権限が渡されます。実行権限をベーシック エンジン ロボットに戻して編集を続行するには、ロボットからステップ アウトする必要があります。

実行権限を持つことができるロボットは一度に 1 つだけです。ロボットが実行権限を保持している場合は、このロボットのタブに赤い点が表示されます。現在実行中の他のすべてのロボットは、以下のようにタブがハイライト表示されます。



実行権限がなくてもロボットの編集は可能ですが、レコーダー ビュー/アプリケーション ペインでロボットの状態を追跡したり、状態ペイン/データの状態ペインで変数の状態を追跡したりすることはできません。

また、デバッグ モードでは複数のベーシック エンジン ロボットを同時に実行できるのに対し、デザイン モードでは 1 つのベーシック エンジン ロボットのみが、実行する権限を持ちます。デザイン モードでの実行権限を持つロボットはデバッグ モードでも同時に実行することができるため、2 つのモードを切り替えることができます。詳細については、「デバッグ モード」を参照してください。

一般編集

このトピックでは、Design Studio でのベーシック エンジン ロボットの編集に関する一般的なヒントを示します。このヒントは、ロボット ビューでロボットに変更を加える場合、タイプ エディターでタイプに変更を加える場合、またはテキスト エディターでテキストに変更を加える場合に適用されます。Design Studio で作業をする際には、便利なキーボード ショートカットを使用することもできます。

コピー、貼り付け、または切り取り

Design Studio でアイテムの切り取り、コピー、貼り付けを行うには、キーボード ショートカットを使用します。

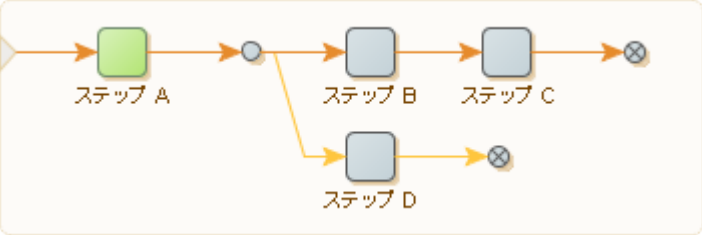
- **Ctrl-C** コピー
- **Ctrl-V** 貼り付け
- **Ctrl-X** 切り取り

さらに、ステップのファインダー一覧などの多くのリストでは、**Ctrl-Shift-C** を使ってリスト内のすべてのアイテムをコピーすることができます。

ステップのグループ化とグループ解除

ステップをグループ化するには、複数のステップを選択して、ツールバーの [グループ化] をクリックします。また、ステップを右クリックして、リストから選択することもできます。

選択をグループ化できない場合もあります。グループ ステップには、必ず 1 つの入力接続と 1 つの出力接続が必要です。またこれは、グループ化するステップの選択でも同様です。唯一の例外は、選択されたステップに出ていく接続がない場合です。この場合、選択範囲はグループ化できますが、最上位の End ステップはグループの最後に接続される必要があります。次の例をご覧ください。



以下は、このロボットでグループ化できるステップの一例です。

- 全ステップ。
- ステップ A、ステップ B などの、任意のシングル アクション ステップ。
- 分岐ポイント、ステップ B、ステップ C およびステップ C の後の終了ステップ。

以下は、グループ化できないステップの一例です。

- 分岐ポイントとステップ B (1 つ以上の入力接続)
- ステップ B、C、D および 2 つのエンド ステップ (複数の出力接続)

作成したグループ ステップにステップをドラッグするか、コピー (カット) および貼り付けオプションを使用して、作成したグループ ステップにステップを追加できます。

ステップまたはステップのコレクションをグループ解除するには、次のいずれかの操作を実行します。グループ ステップからドラッグするか、グループ ステップを折りたたみ、折りたたまれたグループス

テップをクリックして、ツールバーの [グループを解除] ボタンをクリックします (または、コンテキスト メニューで [グループを解除] を選択します)。

- ステップをグループ ステップの外にドラッグします。
- グループ ステップを折りたたみ、それを選択して、ツールバーの [グループを解除] ボタンをクリックします。
- グループ ステップを折りたたみ、それを右クリックして、コンテキスト メニューで [グループを解除] を選択します。

グループ化とグループを解除は逆の動作です。選択したステップをグループ化してから直ちにグループ解除を行っても、ロボットの構造は変更されません。

すべてのグループに対してこのアクションを実行するには、ツールバーの [展開する] と [折りたたむ] を使用します。

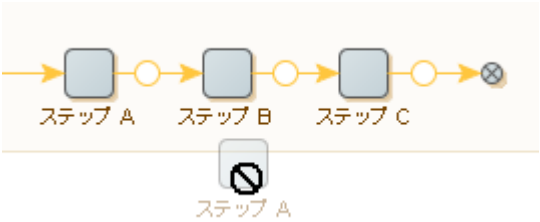
選択した 1 つ以上のグループに対してこのアクションを実行するには、コンテキスト メニューの [展開する] と [折りたたむ] を使用します。

ステップのコンテキスト メニューの [展開する] オプションと [閉じる] オプションでも同じ動作を行うことができますが、その範囲は選択済みのグループ ステップに制限されます。

ドラッグ & ドロップ

アクションに加えて、ドラッグ & ドロップを使ってロボットのエレメントを直接編集できます。ステップをドラッグすると、有効なドロップ位置を示す特別なインジケーターが表示されます。一度に複数のステップの選択や移動を行うこともできます。

- 接続エンドポイントを移動するには、接続を選択し、最後にあるハンドルの 1 つにマウスを移動します。次に、ハンドルをクリックして、新しい場所に移動します。ハンドルをクリックするとすぐに特別なインジケーターが表示され、接続できる箇所が示されます。
- ドラッグ & ドロップ アクションを中止するには、マウスをロボットの外に移動させ、次の図に示すようにマウス ボタンを離します。



複数の分岐を持つロボットの要素をドラッグ & ドロップすることはできません。

新規接続の追加

マウスを使って新しい接続を作成することもできます。ステップの終わり付近にポインターを置くと、インジケーターが表示されます (緑色の輪が付いたオレンジ色の円)。インジケーターをクリックすると、新しい矢印が現れます。マウスの左ボタンを押したままマウスを動かすと、新しい接続がマウスの動きを追います。新たなインジケーターが表示されるので、新しい接続エンド ポイントでマウスの左ボタンを離してドロップします。

変更の元に戻すとやり直し

ロボットの編集中は、すべてのアクションを元に戻したり、やり直したりすることができます。  をクリックするか、Ctrl-Z を選んでアクションを元に戻します。同様に、  をクリックするか、Ctrl-Y を押してアクションをやり直します。

ステップの検証

ロボットを編集すると、ロボット ビューにより各ステップの検証が行われます。無効なステップには赤い下線が引かれています。無効なステップにマウスを移動すると、エラーの説明が表示されます。

[ステップを挿入] メニューによく使用するステップを追加

ロボットの編集では、他の作業よりも多くのステップを使用する場合があります。よく使用するステップを挿入するのに必要な時間を最小限に抑えるために、ステップを [ステップを挿入] メニューに直接追加できます。[ステップを挿入] メニューにステップを追加するには、[Design Studio の設定] ダイアログ ボックスで [ベーシック エンジン ロボット エディター] を開きます。リストにステップを追加するには、[よく使用するステップ] の  をクリックし、ステップを選択します。リストからステップを削除するには、1 つ以上のステップを選択して  をクリックします。ステップを並べ替えるには、ステップを選択し、矢印を使用してリスト内で上下に移動します。

キーボード ヘルプ

キーボード ショートカットを使用すると、製品のさまざまな領域に簡単かつ効率的に移動できます。通常、ポインタをオプションに合わせると、ツールチップにショートカットが表示されます。

単一行テキスト フィールド

説明	キーボード ショートカット
前/次の語に移動	Ctrl + 左矢印、Ctrl + 右矢印
フィールドの先頭/末尾に移動	Home/End
すべて選択	Ctrl + A
すべて選択解除	矢印キー
選択を左/右へ拡張	Shift+左矢印、Shift+右矢印
選択を先頭/末尾へ拡張	Shift+Home、Shift+End
選択を前/次の語へ拡張	Ctrl+Shift+左矢印、Ctrl+Shift+右矢印
選択をコピー	Ctrl + C
選択を切り取り	Ctrl + X
クリップボードから貼り付け	Ctrl + V
次の文字を削除	削除
前の文字を削除	バックスペース

複数行テキスト フィールド

説明	キーボード ショートカット
行頭/行末へ移動	Home、End
前/次の語に移動	Ctrl+左矢印/右矢印

説明	キーボード ショートカット
テキストの先頭/末尾へ移動	Ctrl+Home/End
上/下へブロック移動	PgUp、PgDn
左へブロック移動	Ctrl + PgUp
右へブロック移動	Ctrl + PgDn
すべて選択	Ctrl + A
選択をコピー	Ctrl + C
選択を切り取り	Ctrl + X
選択したテキストを貼り付け	Ctrl + V
次の文字を削除	削除
前の文字を削除	バックスペース
改行を挿入	Enter

ボタン

説明	キーボード ショートカット
アクティベート	スペースバー

ドロップダウン メニュー

説明	キーボード ショートカット
メニューをトグル アップ/ダウン	Alt+Up/Down

複数行テキスト フィールド

説明	キーボード ショートカット
エントリを広げる	右矢印
エントリを閉じる	左矢印
エントリを広げる/閉じるの切り替え	Enter
1 つのエントリを上/下へ移動	上矢印、下矢印
最初のエントリへ移動	Home
最後の可視エントリへ移動	終了
垂直方向にブロック移動	PgUp、PgDn
左へブロック移動	Ctrl + PgUp
右へブロック移動	Ctrl + PgDn
垂直方向にブロック展開	Shift + PgUp、Shift + PgDn
すべて選択	Ctrl + A
単一選択	Ctrl + スペースバー

リスト

説明	キーボード ショートカット
リスト内を移動	上矢印、下矢印
リストの先頭へ移動	Home
リストの末尾へ移動	終了
すべてのエントリを選択	Ctrl + A
切り取り	Ctrl + X
コピー	Ctrl + C
すべてのエントリをコピー	Ctrl + Shift + C
貼り付け	Ctrl + V

表

説明	キーボード ショートカット
次のセルへ移動	右矢印
前のセルへ移動	左矢印
行の最初のセルへ移動	Home
行の最後のセルへ移動	終了
表の最初のセルへ移動	Ctrl + Home
表の最後のセルへ移動	Ctrl + End
すべてのセルを選択	Ctrl + A

ショートカットの中には、ツールチップとして表示されないものもあります。このようなショートカットの一覧については、以下の表を参照してください。

メイン ビューで開いている エディター

説明	キーボード ショートカット
プロジェクト ビューで現在のファイルを選択	Alt+F1

ブラウザ トレーサ

説明	キーボード ショートカット
フォーカスされているビューで検索	Ctrl+F
次の検索結果に移動	F3
特定の行に移動	Ctrl+G

データ テーブル ビュー **(Excel**、**CSV**、データベースの結果**)**

説明	キーボード ショートカット
選択範囲を変更	矢印キー
選択範囲を最上部に変更	Page Up または Ctrl+Home
選択範囲を最下部に変更	Page Down または Ctrl+End
選択範囲を最初の列に変更	Home
選択範囲を最後の列に変更	End

編集可能なテキスト エリア **(**ロング テキスト変数値など**)**

説明	キーボード ショートカット
検索	Ctrl+F
次の検索結果に移動	F3
元に戻す	Ctrl+Z
やり直し	Ctrl+Y
切り取り	Ctrl + X
コピー	Ctrl + C
貼り付け	Ctrl+P
選択を変更	Up または Down

選択したファイルを含むプロジェクト ビュー

説明	キーボード ショートカット
選択したファイルを開く	Return

ロボット ビュー

説明	説明
選択を変更	矢印キー
ズームを 100% にリセット	Ctrl+0 または Ctrl+テンキー 0
選択のクリア	Esc
選択したエッジ (ステップではありません) まで実行	Return
ステップ名またはイテレーションの編集中:変更を確定	Return
ステップ名またはイテレーションの編集中:変更を破棄する	Esc
検索バー内:検索	Return
検索バー内:検索結果をクリア	Esc

ロボット エディター

説明	説明
拡大する	Ctrl+「+」(一部のキーボード) または Ctrl+テンキーのプラス
縮小する	Ctrl+「-」または Ctrl+テンキーのマイナス
ズームを 100% にリセット	Ctrl+0 または Ctrl+テンキー 0
検索結果をクリア	Esc
状態ビューで値の編集中:変更を承認	Return
オートコンプリート ポップアップ ウィンドウが開いている間:選択を変更	Up または Down
オートコンプリート ポップアップ ウィンドウが開いている間:選択を承認	Return、Tab、または Space
オートコンプリート ポップアップ ウィンドウが開いている間:オートコンプリートを破棄	Esc
ローカル Desktop Automation 実行中:ストリーミングを一時停止	Ctrl+Shift+Alt+P (グローバル)
ローカル Desktop Automation 実行中:ストリーミング ビューの次のタブに切り替え	Ctrl+Shift+Alt+T (グローバル)
オートコンプリートが有効なフィールド (エクスプレッション、Catch 句など) の編集中:オートコンプリート ポップアップ ウィンドウを開く	Ctrl+スペース
ポップアップ テキスト エディター ウィンドウが開いている間 (ステップ名、エクスプレッション、変数など):変更を承認してエディター ウィンドウを閉じる	Tab (修飾キー) または Esc
検索バー内:検索	Return
ストリーミング ビュー内:拡大する	Ctrl+「+」(一部のキーボード) または Ctrl+テンキーのプラス
ストリーミング ビュー内:縮小する	Ctrl+「-」または Ctrl+テンキーのマイナス
ストリーミング ビュー内:ズームを 100% にリセット	Ctrl+0 または Ctrl+テンキー 0
ストリーミング ビュー内:ビューを移動	矢印キー
ストリーミング ビュー内:次の同位層を選択	Ctrl+Alt+Down
ストリーミング ビュー内:前の同位層を選択	Ctrl+Alt+Up
ストリーミング ビュー内:最も内側のタグを選択	Ctrl+Alt+End
ストリーミング ビュー内:最も外側のタグを選択	Ctrl+Alt+Home
ストリーミング ビュー内:1 レベル内側のタグを選択	Ctrl+Alt+右

説明	説明
ストリーミング ビュー内:1 レベル外側のタグを選択	Ctrl+Alt+左

ロボットの状態

ロボットが実行されると、主に以下の 4 つのエレメントからなるロボット状態で動作します。

- ウィンドウ
- 変数
- Cookie
- 認証

ウィンドウ エレメントは現在開いているウィンドウに対応し、それぞれにページが含まれています。このページは、HTML ページ、スプレッドシート、XML ページなどである可能性があります。このページには、ウィンドウにロードされたページのタイプに応じた特定のページ タイプがあり、ページ ビューの外観とロボットに挿入できるステップはこのタイプによって異なります。少なくとも 1 つのウィンドウが常に開いており、1 つのウィンドウがカレント ウィンドウとしてマークされます。変数エレメントには、変数の現在の値が含まれます。Cookie および認証エレメントは、それぞれ HTTP Cookie と HTTP 認証で、Web サーバーとの通信中に受信されます。

デバッグ モード

このトピックでは、デバッグ モードの概要と、デザイン モードとデバッグ モードの相互連関について説明します。

デバッグ モードは、ベーシック エンジン ロボットをデバッグするための特殊なモードです。ベーシック エンジン ロボットを編集した後、編集したロボットを RoboServer で実行するのと同じ方法で実行することをお勧めします。つまり、ロボットの機能をすべてテストするために、場合によっては異なる入力値を使用して、ロボットを最後まで実行することをお勧めします。これはデバッガーで実行できます。

デバッガーで 1 つまたは複数のロボットを実行すると同時に、デザイン モードで他のロボット、スニペット、またはタイプを編集することは可能ですが、このトピックで後述するように、編集プロセスがデバッガーでのロボットの実行に影響を与える場合があります。

ロボットの編集とデバッグとの相互連関をよりよく理解するために、デバッグ プロセスの状態に関する次の用語を理解することが重要です。

- デバッガーの起動時には、ロボットはデバッガーで実行準備完了状態になっており、実行されてはいません。デバッガーでロボットの実行を開始する前に、[入力値/出力値] タブでロボットの入力値を変更できます。ただし、デザイン モードで編集したロボットの入力値も変更されることに注意してください。つまり、デバッグ モードでロボットを変更すると、デザイン モードでも変更されます。
- デバッガーをクリックして実行すると、ロボットの実行が開始します。ブレークポイントが設定されている場合は、ロボットがこれらのブレークポイントの 1 つに達すると、ロボットは一時停止しますが、まだ状態を保持しているため、実行は継続されます。クリックしてデバッガーを再度実行すると、一時停止したところから実行が再開されます。

デバッガーでロボットを実行しているときに、デザイン モードでロボットを編集することはできません。デザイン モードでロボットを編集するには、ロボットが準備完了状態であるか、一時停止中である必要があります。デバッガーで一時停止中のロボットを編集すると、デバッグの実行が停止し、ロボッ

トが準備完了状態に戻ります。このとき、デバッグの状態も失われ、出力値やログ メッセージなどのデバッグ実行に関する情報も失われます。

ロボットがデバッガーで実行されているとき、そのロボットが依存するリソース (ロボット、スニペット、タイプなど) を編集できます。これを行うと、タブを [デバッグ] に切り替えるまで、ロボットはデバッガーで実行を続けます。タブを切り替えると、デバッガーはロボットの実行を停止して準備完了状態に戻り、デバッグの状態が失われます。タブを切り替える前にデバッガーでロボットの実行が終了した場合、状態は失われず、デバッガーで確認できます。

ロボットがデバッガーで実行されているときに、そのロボットが依存していない他のロボット、スニペット、またはタイプを変更しても、デバッガーでのロボットの実行には影響しません。

さまざまなデバッグ オプションの詳細については、次のトピックを参照してください。

- ベーシック エンジン ロボット のデバッグ
- デザイン モードで現在のステップからデバッグ
- デバッグ ロケーションからデザイン モードに戻る
- ブレークポイントの使用 (特定のステップでデバッガーを停止させるブレークポイントを設定する機能を提供)
- シングル ステップ (デバッグ モードで一度に 1 つのステップを実行する機能を提供)
- ステップ (グループ ステップにステップして各ステップをデバッグする機能を提供)

ベーシック エンジン ロボット のデバッグ

1. デバッグ モードに切り替えるには、Design Studio でツールバーの **[デバッグ]** > **[デバッグ モードに切替]** をクリックするか、[デバッグ] ボタンをクリックします。
2. ロボットのデバッグを開始するには、[実行] をクリックします。
3. ロボット ビューでは、デバッグ モードでのロボット実行を見ることができます。
   メイン パネルに、デバッグ プロセスの結果がさまざまなタブに分割されて表示されます。
   - [入力値/出力値]: デバッグ中に使用されたすべての変数と、返されたすべての値のリスト。
   - [**API** 例外]: デバッグ中に報告された API 例外のリスト。
   - [ログ]: デバッグ中に生成された処理ログ。ループ フォーム アクションなど、特に実行に時間がかかる一部のアクションでは、ステータス情報がこのログに書き込まれます。設定によっては、ステップ エラーもログ記録されます。
   - [状態]: デバッグ プロセスが一時的に停止されるときは常に [状態] タブに現在のステップに入力される口ボット状態が表示されます。[状態] タブには複数のサブタブが含まれています。
     - [変数]: 変数をリストします。
     - [ウィンドウ]、[**Cookie**]、および [認証]: 状態が関連付けられているダイアログとともに表示されます。
     - [ローカル ストレージ] と [セッション ストレージ]: ローカルに保持された HTML5 オブジェクトを表示します。
     - [**API** 例外]: 現在のステップで生成されます。すべての API 例外 (および関連するエラー) に対し、 [ステップを移動] ボタンをクリックして、エラーが生成されたステップ (デザイン モー

ド) に移動できます。エラーが生成されたステップが Design Studio の現在のステップになります。

- サマリー: デバッグ プロセス中に返された、またはデータベースに書き込まれた変数の数と、生成された API 例外の概要。
- [次の場合停止]: デバッグ プロセスを一時的に停止するために必要な条件を指定します。
- [スキップするステップ]: データベース データ登録、データベース データ削除、SQL 実行、コマンド ライン実行、メール送信など、スキップするステップを選択します。

4. デバッグを終了するには、[停止] をクリックします。

   デバッグは任意のタイミングで終了できます。

5. 特定のイベントが発生したときにデバッグを終了するには、[次の場合停止] アクションを入力します。

   ここで、値が返されたとき、API 例外が報告されたとき、およびブレークポイントに到達したときに、デバッグを終了するかどうかを選択できます。

   ロボットの実行が完了すると、デバッグは常に終了します。

   デバッグが終了した場合、ロボット エディターの下部にあるステータス バーに停止した原因が表示されます。

   ロボットの実行が完了する前にデバッグが終了した場合、[状態] タブで現在のロボット状態を確認できます。[変数]、[Windows]、[Cookie]、および [認証] サブタブには、Design Studio での状態ビューの場合と同じように、ロボット状態が表示されます。API 例外が報告されたために実行が停止した場合、[API 例外] サブタブに API 例外が表示されます。

6. ロボットの実行が完了する前にデバッグが停止した場合は、[実行] をクリックして、デバッグを再開します。

   [デバッグを再開] をクリックして、デバッグを再開することもできます。クリックすると、現在のデバッグ プロセスが停止し、デバッガーでは、ロボットの起動時に新しいデバッグ操作を開始する用意が整います。

   > Design Studio で現在のロボットを変更した場合、または別のロボットに交換した場合は、デバッグが自動的に再起動されます。

7. ロボットに入力変数がある場合は、[入力] パネルでその変数を編集できます。[停止] をクリックして値を変更し、Enter キーを押して、新しい入力値でデバッグを再開します。

   デバッグの実行中に入力変数を編集することはできません。入力変数を変更するには、最初にデバッグを再起動する必要があります。

デザイン モードで現在のステップからデバッグ

1. 現在のステップからデバッグを開始するには、デザイン モードで [現在のステップからデバッグ開始] をクリックします。

   ロボット エディターがデバッグ モードに切り替わり、デザイン モードの現在のステップまで可能な限り直接的に実行します。

   そのステップに到達すると、デバッグが停止します。

2. [実行] をクリックして、このステップからデバッグを続行します。

   この機能は、特定の分岐やループ アクションの特定のイテレーションなど、ロボットの一部をデバッグするときに便利です。

[現在のステップからデバッグ開始] をクリックしたときに、Design Studio がすでにロボットをデバッグ中である場合は、デバッグを再開してそのステップまで実行する必要があり、ユーザーに許可が求められます。

3. [**OK**] をクリックします。

デバッグ ロケーションからデザイン モードに戻る

デバッグ中に [Design Studio ウィンドウ内のこのロケーションに移動] または [ステップを移動] を使用して所定のロケーションに戻ることができます。以下の手順は、この 2 つのオプションでデザイン モードに戻る方法を示しています。

1. あるロケーションでロボットのデバッグが停止した場合は、デバッグ モードで [ステップを移動] をクリックして、同じロケーションでデザイン モードに切り替えます。

   これにより、デザイン モードでその ロケーションを詳しく調べて、その ロケーション周辺のステップを変更したり、ロボットの他の部分の変更を可能にします。

   または、デザイン モードに切り替えて、値が返された ロケーションに移動できます。
2. [入力値/出力値] タブで、[出力値] パネルの値を選択し、右下の [ステップを移動] をクリックします。

   この操作は、値が正しく抽出されておらず、その理由を知りたい場合に便利です。

   また、デザイン モードに切り替えて、API 例外が返された ロケーションまたはエラーが発生した ロケーションに移動することができます。
3. [API 例外] タブ、または [状態] タブの [API 例外] サブタブに API 例外が表示された場合は、ロケーション コードの横にある [ステップを移動] をクリックし、API 例外が生成された ロケーションに移動します。特定のエラーの右側にある [ステップを移動] をクリックし、そのエラーが発生した ロケーションに移動します。

   この操作は、エラーの理由を特定して、問題を修正する場合に便利です。
4. デザイン モードで操作が完了したら、デバッグを再開できます。ロボットを変更していない場合は、[実行] をクリックできます。

   ロボットを変更した場合は、デバッグが自動的に再起動されるため、デバッグを直接再開することはできません。その代わり、[現在のステップからデバッグ開始] をクリックして、デザイン モードで現在の ロケーションから新しいデバッグ セッションを開始できます。

ブレークポイントの使用

デバッグしているときに、ブレークポイントを設定して、特定のステップで Design Studio を停止することができます。

1. ロボット ビューで、ステップを右クリックし、[ブレークポイントの切り替え] を選択します。

   ステップで、ブレークポイントはブレークポイント アイコンで示されます。

   デバッグの際に Design Studio はブレークポイントで停止します。
2. 特定のアクションが発生したときにデバッグを停止するには、[次の場合停止] パラメータを入力します。
3. デバッグを再開するには、[実行] をクリックします。
4. ブレークポイントを除去するには、ステップを右クリックし、[ブレークポイントの切り替え] を選択します。
5. 複数のステップからすべてのブレークポイントを除去するには、1 つまたは複数のステップを選択し、[ブレークポイントの除去] をクリックします。

選択したステップからすべてのブレークポイントが除去されます。

6. ロボットのすべてのブレークポイントを除去するには、[すべてのブレークポイントの除去] をクリックします。

   ロボットからすべてのブレークポイントが除去されます。

シングル ステップ

シングル ステップを使用して、デバッグ モードでステップを 1 つずつ実行することができます。シングル ステップは、実行を詳細に調べる場合に便利です。

Design Studio で新しいデバッグを開始する用意ができているとき、または Design Studio がデバッグ中に停止したときに、シングル ステップを使用できます。

1. 次のステップを実行するには、[シングル ステップ] をクリックします。

   そのステップが実行されたら、実行が停止します。
2. [シングル ステップ] を再度クリックして、次のステップを実行します。
3. 通常の実行を再開するには、任意のステップで [実行] をクリックします。

ステップ

1. グループ ステップを使用して、複数のステップが含まれるグループを作成した場合は、他のステップでステップを作成したときと同じように、グループ ステップでブレークポイントを作成します。
   - シングル ステップを使用してグループを折り畳むと、Design Studio では、そのグループをシングル ステップと見なすため、グループ化されたステップのいずれにも入れなくなります。
   - グループを展開すると、シングル ステップの使用時にグループ内のステップにアクセスできます。
   - シングル ステップ アイコンではなく、ステップ アイコンを使用してグループを折りたたむと、デバッガーからグループに入って各ステップにアクセスすることができます。
2. グループから出て、デバッガーをグループ ステップの次のステップに進めるには、[ステップ アウト] をクリックします。

ステップとデータ コンバータ

Design Studio では、各ステップとデータ コンバータについて簡単な説明が表示されます。説明に関連付けられたステップまたはデータ コンバータについてさらに情報を表示するには、説明の横にある [詳細] をクリックします。また、ヘルプ をクリックすると、選択したステップまたはデータ コンバータに関連付けられたオンスクリーン ヘルプが表示されます。

抽出アクションのようないくつかのアクションは、データ コンバータのリストを通して抽出されたテキストを実行することができ、結果を変数にソートします。

ベーシック エンジン ロボットのステップとそのオプションについては、ステップを参照してください。

データ コンバータは、ユーザーが定義したパラメータに基づき、抽出されたテキストを処理します。たとえば、[数値を抽出] データ コンバータは、数字を含んだ入力テキストを受け取り、同じ数字を含むテキストを標準フォーマットで出力します。

データ コンバータは、テキストを入力値として取り込み、別のテキストとして出力するため、データ コンバータを連続して接続し、あるデータ コンバータの出力が次のデータ コンバータへの入力となるようにできます。最終出力は、データ コンバータ リストにある最後のデータ コンバータのテキスト出力値になります。たとえば、データ コンバータのリストでコンバータ [大文字に変換] の後に [スペースを除去] データ コンバータがある場合、リストへの入力テキストが "R oboMa ker" であると出力は "ROBOMAKER" になります。

利用可能なデータ コンバータについては、データ コンバータを参照してください。

ステップ

このトピックでは、利用可能なステップの概要を説明します。

最も一般的に使用されるステップを、Design Studio のロボット ビューで接続を右クリックしたときに使用可能な [ステップを挿入] メニューに直接追加することができます。詳細については、一般編集を参照してください。

標準

このカテゴリには最もよく使われるステップが含まれます。

ステップ	説明
変数の割当	値を変数に割り当てます。
ロボットの呼び出し	ベーシック エンジン ロボットからロボットを呼び出すステップを作成します。
ページ生成	新しいページを作成します。
Desktop Automation	廃止予定です。「ロボットの呼び出し」を参照してください。
変数を開く	ビューで変数属性 (またはシンプル タイプの変数) を開きます。
ページ読み込み	URL から Web ページを読み込みます。
値返却	ロボットから値を返却します。
データベース データ登録	値をデータベース データ登録します。
値判定	ブール値に応じてステップ以降の実行を停止または続行します。

変数割り当て/変換

このカテゴリには最もよく使われるステップが含まれます。

ステップ	説明
変数の割当	値を変数に割り当てます。
変数の変換	変数の値をデータ コンバータで変換し、結果を同じ変数または別の変数に保存することによって 1 つ以上の変数の値を変換します。
XML の変換	XSLT を使用して XML を変換します。

ブラウザ セッション

このカテゴリには、ブラウザ セッション全体の保存と復元および Cookie と HTML 5 Web ストレージの抽出と操作を行うためのステップが含まれます。

ステップ	説明
セッションの保存	後で別の口ボット実行によって復元するためにセッションを変数に保存します。
セッションの復元	以前、別の口ボット実行によって保存されたセッションを変数から復元します。
Cookie 抽出	このアクションは、名前、ドメイン、およびパスのパターンに一致する Cookie の値を抽出します。
Cookie 作成	このアクションは、指定したドメイン、パス、名前、および (必要に応じて) 値の Cookie を作成します。
Cookie 除去	このアクションは、名前、ドメイン、パス、および値のパターンに一致する 1 つ以上の Cookie を除去します。
Web ストレージ抽出	ローカル ストレージまたはセッション ストレージあるいはその両方からデータを抽出します。データは JSON 形式で変数に保存されます。
Web ストレージ読み込み	ローカル ストレージまたはセッション ストレージあるいはその両方にデータを読み込まれます。データは JSON 形式で指定する必要があります。
Web ストレージ消去	ローカル ストレージ内またはセッション ストレージ内あるいはその両方のデータを消去します。

ブラウザ ウィンドウ

このカテゴリにはブラウザ ウィンドウを開く、選択する、閉じるという操作のためのステップが含まれます。

ステップ	説明
新しいウィンドウを開く	新しいウィンドウを作成します。
カレント ウィンドウ設定	次以降のステップが操作するウィンドウなど、別のウィンドウを現在のウィンドウとして設定します。
ウィンドウを閉じる	ウィンドウを閉じます。

Web サービス呼び出し

このカテゴリには REST Web サービスおよび SOAP Web サービスを呼び出すためのステップが含まれます。

ステップ	説明
REST Web サービス呼出	REST Web サービスを呼び出し、結果を現在のウィンドウに読み込むか、変数に保存します。
SOAP Web サービス呼出	SOAP XML リクエストを Web サービスに送信し、SOAP XML レスポンスを返します。

マウスのクリック/移動

このカテゴリには、ブラウザ ビュー内の要素に対するマウス クリックやマウス移動、およびブラウザ ビュー内の要素からのマウス クリックやマウス移動を模倣するステップが含まれます。

ステップ	説明
クリック	検出されたタグ上のマウス クリックをエミュレートします。
マウスオーバー	検出されたタグへのマウス移動をエミュレートします。
マウスアウト	検出されたタグからのマウス移動をエミュレートします。

ステップ	説明
スクロール	ドキュメントまたはタグのスクロールをエミュレートします。
指定タグまでスクロール	検出されたタグをスクロールして表示する操作を エミュレートします。

データベース

このカテゴリにはデータベース内の項目の保存、取得、照会または削除を行えるステップが含まれます。

ステップ	説明
データベース データ登録	値をデータベース データ登録します。
データベース データ抽出	データベース内の値を検索します。
キーの計算	選択された変数の値を保存するために使用されるキーを計算します。
データベース データ削除	データベース内の値を削除します。
データベース照会	SQL クエリをデータベースに送信し、結果を順次ループします。
SQL 実行	データベースで SQL ステートメントを実行します。

フォームへのデータ入力

このカテゴリには Web フォームにデータを入力するためのステップが含まれます。

ステップ	説明
テキストを入力	フォームのテキスト フィールドにテキストを入力します。
パスワード入力	フォームのパスワード フィールドにパスワードを入力します。
キー プレス	フォームでキーを押す操作をエミュレートします。
オプション選択	フォームのドロップダウン ボックスまたはリスト ボックスからオプションを選択します。
複数オプション選択	フォームのリスト ボックスから複数のオプションを選択します。注意：このステップはリスト ボックスでのみ使用できます。ドロップダウン ボックスでは使用できません。
チェックボックス設定	フォームのチェックボックスを選択または選択解除します。
範囲値の設定	下回ることのできない下限の数値と、超えることのできない上限の数値を設定します。
ラジオ ボタン選択	フォームのラジオ ボタンを選択します。
ファイル選択	フォームのファイル フィールドでアップロードするファイルを選択します。

抽出

このカテゴリにはデータを抽出するためのステップが含まれます。データは、Web サイトあるいは PDF、CSV、および Excel などの他の書式から、テキスト形式または HTML 形式で抽出されます。画像抽出したり、属性値やリンク URL など、HTML ソースや XML ソースに関する特定のデータを抽出したりすることもできます。

ステップ	説明
抽出	テキストを抽出し、それをデータ コンバータのリストで変換して、結果を変数に保存します。

ステップ	説明
バイナリ コンテンツ抽出	ブラウザ ビューからバイナリ コンテンツを抽出します。
セル値抽出	Excel ページからコンテンツを抽出し、それをデータ コンバータのリストで変換して、結果を変数に保存します。
データ行の列を抽出	現在の範囲内のセルからデータを抽出して変数に入れます。
電子メールから抽出	保存された電子メールから情報を抽出します。
フォームパラメータを抽出	検出されたタグ内のフォーム URL からパラメータから抽出します。
Flash コンテンツ抽出	廃止予定です。
PDF から抽出	変数に含まれている PDF ドキュメントからテキストを抽出します。
画像抽出	画像抽出し、それを変数またはファイルに保存します。オプションで、画像のコンテンツ タイプおよびファイル名を別の変数に保存することができます。
JSON 抽出	JSON ファインダーが見つけた JSON 値の一部を、抽出して JSON 値として変数に格納します。
パス抽出	ファインダーによって検出された要素の絶対パスを抽出します。
プロパティ名抽出	JSON ファインダーが見つけた JSON 値のプロパティ名を抽出して、変数に格納します。
スクリーンショット抽出	現在のページから画像抽出し、それを変数に保存します。
選択済オプション抽出	選択されたオプションのテキストまたは値を抽出し、それをデータ コンバータのリストで変換して、結果を変数に保存します。
ソース抽出	プレビューしたデータを変数に保存します。
タグ属性抽出	検出されたタグからタグ属性を抽出し、それをデータ コンバータのリストで変換して、結果を変数に保存します。
ターゲット抽出	URL ターゲットからデータを抽出し、それを変数またはファイルに保存します。オプションで、読み込まれたデータのコンテンツ タイプおよびファイル名を別の変数に保存することができます。
URL 抽出	検出されたタグから URL を抽出し、それを変数に保存します。

ファイル システム

このカテゴリにはファイル システムにアクセスするためのステップが含まれます。ファイルとディレクトリの読み込み、書き込み、変更を行ったり、ディレクトリ内のファイルをループしたり、指定されたファイルの有無を判定したりすることができます。

ステップ	説明
ファイル読み込み	ファイルからブラウザ ウィンドウまたは変数へデータを読み込みます。
ファイル繰り返し	ディレクトリ内のファイルを順次ループします。
ファイル出力	新しいファイルを書き込むか、既存のファイルに追加します。
ファイル有無判定	特定のファイルの有無に応じてステップ以降の実行を停止または続行します。
ファイル情報取得	ファイル システム内のファイルに関するメタデータをフェッチします。
ファイル コピー	ロボットが実行されるローカル ファイル システム上のファイルをコピーします。コピー先ファイルが存在すると、このステップはエラーを生成します。

ステップ	説明
ファイル削除	指定されたファイルまたはディレクトリを削除します。
ディレクトリ作成	新しいディレクトリを作成します。
ファイル名変更	ロボットが実行されるローカル ファイル システム上のファイルまたはディレクトリの名前を変更します。変更後 (新しい名前) のファイルが存在すると、このステップはエラーを生成します。

ループ

このカテゴリにはループするためのステップが含まれます。HTML 構造、ウィンドウ、カンマ区切りの値、フォーム値、Excel 範囲を順次ループしたり、ドメイン全体をクロールしたりすることができます。HTML 構造のループには、2 つのオプションがあります: タグ繰り返しとタグ パス繰り返し。[タグ繰り返し] ステップの方が単純です。このステップ アクションは検出されたタグの直下のタグを順次ループするために使用されるのに対して、[タグ パス繰り返し] は検出されたタグ内の範囲内の深度にある同様のタグを順次ループすることができます。[次のステップ] リンクなどによって結合された複数のページを順次ループするには、[繰り返し] ステップと [次のステップ] ステップを使用する必要があります。

選択した要素にループするサブ要素が含まれていない場合、すべての 繰り返しステップがエラーをスローします。たとえば、検出されたタグにループするタグが含まれていない場合、[タグ パス繰り返し] ステップはエラーをスローします。

ただし、[タグ繰り返し] ステップを使用してディレクトリ内のファイルをループする際にディレクトリにファイルが存在しない場合、エラーは見なされず、エラーはスローされません。

ステップ	説明
タグ繰り返し	検出されたタグ内部の最上レベルに含まれているタグを順次ループします。
タグ パス繰り返し	検出されたタグ内部の任意のレベルに含まれているタグを順次ループします。
URL 繰り返し	検出されたタグに含まれている URL を順次ループします。
セレクト オプション繰り返し	フォームのドロップダウン ボックス内またはリスト ボックス内のオプションを順次ループし、各イテレーションで 1 つのオプションを選択します。
ラジオ ボタン繰り返し	ラジオ ボタンのグループを順次ループし、各イテレーションで 1 つのラジオ ボタンを選択します。検出されるタグはグループ内のラジオ ボタンの 1 つである必要があります。
フィールド値ループ	指定された値を順次ループし、各イテレーションでテキスト フィールドに 1 つの値を入力します。
繰り返し	[次へ] ステップと組み合わせて繰り返しループを作成します。
次へ	[繰り返し] ステップを使用して作成された繰り返しループ内の別のイテレーションをリクエストします。
Excel 内ループ	検出された範囲内の行、列、セルまたは Excel ページ内のすべてのシートをループします。
データ行繰り返し	CSV ファイル内の各データ行をループします。
プロパティ繰り返し	JSON オブジェクトのすべてのプロパティをループします。
JSON アイテム繰り返し	タグのグループをループします。
ファイル繰り返し	ディレクトリ内のファイルを順次ループします。

ステップ	説明
ブラウザウィンドウ繰り返し	ブラウザ ウィンドウを順次繰り返し処理し、各ブラウザ ウィンドウを順番に現在のウィンドウに設定します。
部分文字列繰り返し	指定された区切りでテキストを分割し、分割された部分を順次ループします。
イテレーション取得	囲みループ ステップの現在のイテレーションを取得します。

ページ読み込み

このカテゴリには、指定された URL からページを読み込むためのステップまたはすでに抽出されているコンテンツに基づいてページを新規作成するためのステップが含まれます。必要に応じて、基本的な HTTP レベルでページ読み込みリクエストを指定することもできます。

ステップ	説明
ページ読み込み	URL から Web ページを読み込みます。
ページ生成	新しいページを作成します。
HTTP 通信	選択されたメソッドの HTTP 通信リクエストを実行します。
変数を開く	ビューで変数属性 (またはシンプル タイプの変数) を開きます。
Excel 形式表示	ダウンロードした Excel コンテンツを Excel ビューで開きます。
JSON 形式表示	ダウンロードした JSON コンテンツを JSON ビューで開きます。
XML 形式表示	ダウンロードした XML コンテンツを XML ビューで開きます。
CSV 形式表示	ダウンロードした CSV コンテンツを CSV ビューで開きます。
ソース抽出	プレビューしたデータを変数に保存します。

スナップショット生成

このカテゴリには Web ページのオフライン スナップショットを保存するためのステップが含まれます。ページとそのリソースのオフライン HTML コピーを保存するには、[スナップショット生成] を使用します。複数の相互リンクされた HTML ページを保存するには、[ページ再描画] と [CSS 再描画] を使用します。

ステップ	説明
スナップショット生成	フレームとリソースを含む現在のウィンドウのスナップショットを作成します。
ページ再描画	現在のウィンドウの HTML コンテンツを抽出し、さらにスタイル シート、画像、その他のページへのリンクを書き換えて出力します。
CSS 再描画	ページ再描画のヘルパーとして機能します。このアクションの役割は、指定されたスタイル シート内の他のスタイル シートまたは画像へのリンクを再描画ことです。

ページの変更

このカテゴリには、コンテンツの除去、置き換え、挿入などによって現在の Web ページを変更するためのステップが含まれます。

ステップ	説明
タグ挿入	新しいタグを挿入します。
タグ置き換え	検出されたタグを新しいタグに置き換えます。

ステップ	説明
タグ除去	検出されたタグからタグを除去します。除去規則は以下の順番で実行されます。1 つ以上の除外規則に一致するタグはすべて除去されません。除去規則をまったく定義しないと、デフォルトですべてのタグが除去されます。
タグ範囲除去	タグの範囲を除去します。
タグ非表示化	検出されたタグを非表示にします。
タグ表示化	検出されたタグが表示されるようにします。
テキスト分割	検出されたタグ内のテキストを複数の部分に分割します。
テーブル行除去	入力 <table> タグから、指定された数の列 (<td> タグと <th> タグ) を持っていないすべての行 (<tr> タグ) を除去します。
テーブル行列入れ替え	テーブルの <td> タグを左上から右下の対角線に沿って反転することによって入力 <table> タグを入れ替え (反転させ) ます。
セル結合解除	余分なセルを挿入して rowspan と colspan を除去することによってテーブルを正規化します。元のセルのコンテンツは新しいセルにコピーされます。

出力値

このカテゴリには、このロボットを呼び出した API に値を返したり、電子メールを送信したり、ファイルまたはログに書き込みを行ったりするためのステップが含まれます。

ステップ	説明
値返却	ロボットから値を返却します。
メール送信	電子メールを送信します。Design Studio のデザイン モードで実行中は、電子メールが送信されないことに注意してください。
ファイル出力	新しいファイルを書き込むか、既存のファイルに追加します。
ログ出力	メッセージをログに書き込みます。このステップ アクションはロボットをデバッグするときに役立ちます。

判定

このカテゴリには、特定の条件が満たされた場合に現在の分岐以降の実行を停止するなど、判定用の条件付きステップが含まれます。この条件は、検出されたタグのコンテンツ、変数、または指定したウィンドウの有無などに依存します。

ステップ	説明
タグ判定	検出されたタグのコンテンツに応じて、現在の分岐以降の実行を停止または継続します。
URL 判定	検出されたタグに含まれている URL 応じて、現在の分岐以降の実行を停止または継続します。
値判定	ブール値に応じてステップ以降の実行を停止または継続します。
変数判定	1 つ以上の変数値に応じてステップ以降の実行を停止または継続します。
行判定	テーブル行内の列の数を判定します。
ウィンドウ判定	特定のウィンドウの有無に応じてステップ以降の実行を停止または続行します。
ページ タイプ判定	ページのタイプに応じてステップ以降の実行を停止または続行します。

ステップ	説明
セル タイプ判定	空白または検出された範囲の番号などのセル タイプを判定し、範囲内のすべてのセルが指定されたタイプであるかどうかに応じてステップ以降の実行を停止または続行します。
JSON タイプ判定	JSON 値のタイプを判定します。

Excel

このカテゴリには Excel のページ専用に設計されたステップが含まれます。

ステップ	説明
セル値抽出	Excel ページからコンテンツを抽出し、それをデータ コンバータのリストで変換して、結果を変数に保存します。
シート名抽出	スプレッドシート ドキュメントのシートの名前を抽出し、それを変数に保存します。
ハイパーリンク抽出	スプレッドシートのセルからハイパーリンクを抽出します。
Excel 内ループ	スプレッドシートのさまざまな要素を順次ループします。
HTML として抽出	スプレッドシート ドキュメントの一部を HTML テーブルとして抽出し、それを変数に保存します。
セルのコンテンツ設定	指定されたコンテンツをスプレッドシートのセルに挿入します。
セルの値設定	セルの値を設定します。
列のコンテンツ設定	コンプレックス タイプの変数からスプレッドシートの列のコンテンツを設定します。
行のコンテンツ設定	コンプレックス タイプの変数からスプレッドシートの行のコンテンツを設定します。
セルのフォーマット設定	スプレッドシートの 1 つ以上のセルの書式を設定します。
シート名設定	シート名を設定します。
セルのハイパーリンク設定	セルにハイパーリンクを挿入します。
列の幅設定	スプレッドシートの列の幅を設定します。
行の高さ設定	行の高さをポイントで設定します。
プロパティ情報設定	スプレッドシートのプロパティ情報の値を設定します。
シート挿入	スプレッドシートに新しいシートを挿入します。
行挿入	スプレッドシートに 1 つ以上の行を挿入します。
列挿入	スプレッドシートに 1 つ以上の列を挿入します。
シート除去	選択されたシートをスプレッドシートから除去します。
行除去	選択された行をスプレッドシートから除去します。
列除去	選択された列をスプレッドシートから除去します。
セル タイプ判定	1 つ以上のセルのタイプを判定します。
名前付き範囲設定	検出された範囲を 名前付き範囲としてマークし、次以降のステップで範囲を検索するときにそれを参照として使用できるようにします。

ステップ	説明
評価モードの設定	Excel 変数の Excel 値の自動評価オプションを変更して、サポートされていない関数を、エラーを発生させずにセルに入力できるようにします。

JSON

このカテゴリには JSON 値を管理するためのステップが含まれます。

ステップ	説明
JSON 抽出	JSON ファインダーが見つけた JSON 値の一部を、抽出して JSON 値として変数に格納します。
プロパティ名抽出	JSON ファインダーが見つけた JSON 値のプロパティ名を抽出して、変数に格納します。
プロパティ繰り返し	JSON オブジェクトのすべてのプロパティをループします。
JSON アイテム繰り返し	タグのグループをループします。
JSON 設定	検出された JSON 値全体を新しい JSON 値に置き換えます。
プロパティ名設定	検出された JSON オブジェクトのプロパティ名を新しい名前に置き換えます。
JSON 挿入	JSON オブジェクトに新しいプロパティ、または JSON 配列に新しいアイテムを挿入します。
JSON 除去	検出された JSON を JSON 値から除去します。
JSON タイプ判定	JSON 値のタイプを判定します。
名前付き JSON 設定	検出された JSON を 名前付き JSON としてマークします。

XML

このカテゴリには XML に関連したステップが含まれます。

ステップ	説明
抽出	テキストを抽出して変数に保存します
タグ属性抽出	検出されたタグからタグ属性を抽出し、それをデータ コンバータのリストで変換して、結果を変数に保存します。
タグ繰り返し	タグのグループをループします。
タグ パス繰り返し	検出されたタグのサブツリー内にある特定のタイプのすべてのタグをループします。
タグ設定	検出されたタグ全体を新しいコンテンツに置き換えます。
コンテンツ設定	タグ上の指定したコンテンツを設定します。
テキスト設定	検出されたタグのコンテンツをテキストに置き換えます。
名前付きタグ設定	検出されたタグの名前を新しい名前に置き換え、オプションで、検出されたタグの属性を新しいタグにコピーします。
属性設定	特定の名前や値によって、検知タグ上で、属性を挿入したり更新したりします。
コンテンツ挿入	検出されたタグに対応するドキュメントに、指定されたコンテンツを挿入します。

ステップ	説明
タグ除去	検出されたタグを親ノードから除去します。
コンテンツ除去	タグのすべてのコンテンツを除去します。
属性除去	XML 内のタグから属性を除去します。
タグ判定	現在の分岐以降も実行を続行する必要があるかどうかを判定します。
名前付きタグ設定	検出されたタグを名前付きタグとしてマークし、次以降のステップでタグを検索するときにそれを参照として使用できるようにします。

その他

このカテゴリにはその他のさまざまなステップが含まれます。

ステップ	説明
名前付きタグ設定	検出されたタグを名前付きタグとしてマークし、次以降のステップでタグを検索するときにそれを参照として使用できるようにします。
名前付き範囲設定	検出された範囲を名前付き範囲としてマークし、次以降のステップで範囲を検索するときにそれを参照として使用できるようにします。
名前付き JSON 設定	検出された JSON を 名前付き JSON としてマークします。
名前付きタグ/範囲のクリア	選択されている名前付きタグまたは名前付き範囲、またはすべての名前付きタグおよび名前付き範囲のマークを解除し、次以降のステップでそれらが指定されないようにします。
コメント	何もしません。
待機	指定された時間だけ待機します。
ブラウザ再開	指定の待機条件が満たされるか、ブラウザが待機状態になった場合に (どちらか早い方を優先) ブラウザを再開し、実行します。
終了	エラーを生成せずにロボットの実行を終了します。
エラー生成	エラーを生成します。
コマンド ライン実行	コマンド ラインまたはシェル スクリプトを実行します。RoboServer に、この操作を行うための十分な権限があることを確認してください。
プロキシ切替	プロキシ サーバーを切り替えます。
JavaScript の実行	JavaScript を実行します。
パスワード取得	Management Console の パスワード ストア からパスワードを取得します。

変数の割当

このアクションは、値を変数に割り当てます。多くの場合、この値はシンプルなタイプです。値のソースが別の変数である場合、コンプレックス タイプがフィールド (test.result など) ではなく変数自体 (test など) である値が、変数のリストから選択されます。すべての場合において、ターゲットの変数のタイプは受信値と適合します。

プロパティ

以下のプロパティを使用して、「変数割り当て」アクションを設定します。

値
変数に割り当てる値。値セレクターを使用して値を指定します。

変数
値を割り当てる変数を指定します。

分岐ポイント

分岐ポイントは実行が複数の分岐に分かれるロボット内のポイントを示します。

ロボットの実行が分岐ポイントに到達すると、個々の分岐が順番に実行されます。ただし、分岐の実行がエラーで終了する場合を除きます。その場合、実行は、エラーが発生したステップのステップ ビューの [エラー処理] タブの下で指定されたポイントから再開されます。

発信接続に番号の注釈が付いており、その番号が示す順に分岐を実行する場合を除いて、分岐は上から下の順に実行されます。

各分岐は、1 つの分岐から次の分岐まで存続するグローバル変数を除いて、同じ状態 (ページ ビューのページ、Cookie など) で実行されます。外部の世界に加えられる変更も分岐をまたがって存続します。このような変更には、ファイルへの書き込み、データベースへの保存、Web サイトでのフォームの送信など、現実世界に影響を与えるものがあります (Amazon で本を購入するなど)。

分岐ポイントはプロパティを持っておらず、必要とされているかどうかに応じて自動的に挿入または削除されます。分岐から End ステップが除去され、1 つの分岐だけが残ると、分岐ポイントは自動的に除去されます。

キーの計算

このステップでは変数値のキーを計算することができます。値はコンプレックス タイプ、つまり、Number のような組み込み済みシンプル タイプの 1 つではなく、.type ファイルで指定されたタイプである必要があります。タイプの少なくとも 1 つの属性を「データベース キーの一部」としてマークする必要があります。値のキーを把握しておくことは、値を別の値に (たとえば、別のテーブルのセカンダリ キーとして) リンクさせる必要がある場合、またはファイルに保存されたデータにリンクさせる必要がある場合に役立つことがあります。

プロパティ

変数
キーを計算する変数を選択します。レガシー ロボット (バージョン 7.2 より古いバージョン) を使用している場合、変数のタイプは、レガシーの目的でのみ存在している専用のデータベース出力タイプの種類ではなく、標準タイプである必要があります。

キー (出力値)
計算キーが保存される変数。変数には、シンプル タイプの変数とコンプレックス タイプの変数の属性の 2 種類があります。

ロボットの呼び出し

このアクションを行うと、ベーシック エンジン ロボット から  ロボット を呼び出すために必要な「ロボットを呼び出す」ステップが作成されます。詳細については、はじめにを参照してください。

VDI またはリモート デスクトップを使用してアプリケーションを自動化する場合は、まず、オートメーション デバイスへの参照を設定して指定してください。『Kofax RPA Desktop Automation サービス ガイド』を参照してください。

プロパティ

以下のプロパティを使用して「ロボットを呼び出す」ステップを設定します。

ロボットの選択
使用するロボットを指定するか、新しいロボットを作成します。
ベーシック エンジン ロボットの値とマッピングを指定してから、新しいロボットを作成すると、新しいロボットはこれらのプロパティを継承します。

入力値
ロボットに入力値を指定します。

戻り変数
「ロボットを呼び出す」ステップからの出力値を保持する変数を割り当てます。

デバイス
「ロボットを呼び出す」ステップを使用するために、オートメーション デバイスに接続する方法を指定します。[静的リファレンス]、[動的リファレンス]、または [トリガー リファレンス] を選択できます。詳細については、オートメーション デバイスの準備を参照してください。
[静的リファレンス] または [トリガー リファレンス] を選択した場合は、使用するオートメーション デバイス マッピングの要件を指定します。
[動的リファレンス] を選択した場合は、「デバイスに接続」ステップで使用するマッピング名を指定します。ロボットで動的リファレンス接続が使用されており、デバイスが接続されている場合は、その接続が維持され、ロボットの次の「ロボットを呼び出す」ステップで使用できるようになります。

- 「ロボットを呼び出す」ステップ、およびロボットで指定したデバイスの数は一致している必要があります。
- ロボットで設定したデバイス名は、「ロボットを呼び出す」ステップのデバイス名と異なる場合があります。

データベース
ロボットで使用する 1 つ以上のデータベースを選択します。たとえば、「SQL 実行」ステップで指定したマッピングを使用できます。

REST Web サービス呼出

REST Web サービス呼出アクションにより、インターネット上のさまざまなソースからデータを取得します。このステップでは、ソースの URI に対して行われたリクエストによって、HTML、XML、JSON、またはその他の形式でフォーマットされたペイロードを含む応答が返されます。レスポンスは現在のページとして HTML 形式で提示されるか、変数に保存されます。

Web サービスがフォールトを返した場合、アクションはメッセージを返しません。その代わり、アクションは標準エラー処理メカニズムを使用して処理できるエラーを生成します。

このアクション ステップでサポートされている認証には、Negotiate、NTML、Digest、Basic、OAuth が含まれます。ステップ プロパティの [オプション] の下にある [詳細] をクリックすると、[すべてのローディング] タブでこのステップの認証方法を設定できます。プロトコルの詳細については、Web 認証を参照してください。ロボット全体またはこのステップのみのネゴシエート プロトコル パラメータを設定する場合、spn.txt ファイルは使用されません。

プロパティ

以下のプロパティを使用して REST Web サービス呼出アクションを設定できます。

URL
パラメータを除く Web サービスのベース URL。URL セレクターを使用して複数の方法で URL を指定できます。

リクエスト
ここでは、実行されるリクエストのタイプを指定します。REST は、次の 5 つの基本操作をサポートしています。

GET
データをクエリするために使用されます。GET リクエストには、リクエストで渡す名前と値のペアとして、複数のパラメータやファイルを指定できます。「+」をクリックして、新しいパラメータを追加するか、ファイルをアップロードします。

POST
データの選択されている部分のアップデートに使用されます。POST リクエストでは、複数のパラメータを name/value ペアとして指定するか、リクエストの本文全体を渡すことができます。パラメータ付きでリクエストを指定する場合は、パラメータのエンコードに POST (application/x-www-form-urlencoded) または MULTIPART (multipart/form-data) のどちらを使用するかを選択する必要があります。リクエストの本文全体 ('raw') を渡す場合は、リクエスト データのコンテンツ タイプを指定する必要があります。

POST リクエストと PUT リクエストでは、MULTIPART エンコーディングを選択してファイル アップロードを有効にすることができます。ファイル アップロード パラメータの値としてバイナリ変数が選択されている場合は、バイトがそのまま送信されます。Base 64 エンコーディングを使用する場合は、パラメータの値をエクスプレッション base64Encode(data) にする必要があります。data はバイナリ値が含まれた変数の名前です。その場合は、値 base64 を Content Transfer Encoding として指定することをお勧めします。そうしない場合は、このフィールドを通常通り空白のままにすることができます。

PUT
データを置き換えるために使用されます。PUT リクエストを指定するさまざまな方法については、POST の説明を参照してください。

DELETE
データを削除するために使用されます。DELETE リクエストには、リクエストで渡す名前と値のペアとして、複数のパラメータやファイルを指定できます。「+」をクリックして、新しいパラメータを追加するか、ファイルをアップロードします。

PATCH
データを修正するために使用されます。PATCH リクエストには、リクエストで渡す名前と値のペアとして、複数のパラメータやファイルを指定できます。「+」をクリックして、新しいパラメータを追加するか、ファイルをアップロードします。

受け入れ
レスポンスとして受け入れられるコンテンツ タイプ。デフォルトでは、あらゆるタイプのレスポンスが受け入れられます。値セレクターを使用して、受け入れられるコンテンツ タイプを複数の方法で指定できます。

エンコーディング
リクエスト内の特殊文字のエンコードに使用されるエンコーディング。レスポンスのデコーディングに使用されるエンコーディングは、[ページ読み込み] タブの step オプションを使用して制御されます。

出力値
ここでは、Web サービス呼び出しの出力がどうなるかを選択します。

ブラウザへの読み込み
結果がページ読み込みアクションの結果のように現在のウィンドウに読み込まれます。以下で説明するオプション プロパティを使用して、ブラウザの挙動を設定することができます。

変数への保存
選択されている変数に結果が保存されます。

プリエンプティブ認証
認証ヘッダーの事前送信を有効にして、認証プロセス時間を短縮します。プリエンプティブ認証が有効になっている場合は、サーバーが「401 未承認」応答を返す前に、Design Studio から基本認証またはダイジェスト認証応答が送信されます。プリエンプティブ認証を使用するには、以下の [オプション] でデフォルトのクレデンシャルを設定する必要があります。
ダイジェスト認証を使用する場合、各アルゴリズムにはセッションと非セッションの 2 つのバリエーションがあります。非セッション アルゴリズム (SHA-256 など) は <name> と定義し、非セッション アルゴリズム (SHA-256-sess など) は <name>-sess のように定義します。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。[オプション] ダイアログ ボックスでアスタリスクが付いているオプションは、ロボットの設定のオプションよりも優先されます。その他のすべてのオプションは、ロボットに対して指定されているものと同様になります。

SOAP Web サービス呼出

SOAP Web サービス呼出アクションは SOAP XML リクエストを Web サービスに送信し、Web サービスの SOAP XML レスポンスを返します。レスポンスは現在のページとして XML (または HTML) 形式で提示されるか、XML 変数の値としてセットされます。

SOAP リクエストは非常に複雑になる可能性があるため、通常は外部ツールを使用してリクエストを生成し、それを SOAP XML リクエスト プロパティに貼り付けます。エクスプレッションから XML を指定することによってリクエストを動的に変更し、リテラル値をエクスプレッションに置き換えるテンプレート SOAP リクエストを作成することができます。

Web サービスが SOAP フォールトを返した場合、アクションはメッセージを返しません。その代わり、アクションは標準エラー処理メカニズムを使用して処理できるエラーを生成します。

プロパティ

以下のプロパティを使用して SOAP Web サービス呼出アクションを設定できます。

Web サービス URL

ここでは Web サービス操作の場所が指定されます。Web サービスは通常 HTTP プロトコルを使用します。値セレクターを使用して複数の方法で値を指定することができます。

SOAP アクション

このプロパティにはオプションの SOAP アクションを含めることができます。値セレクターを使用して複数の方法で値を指定することができます。SOAP アクションは HTTP ヘッダの一部として送信されます。SOAP アクションは通常、リクエストされたアクションを指定する URL です。

SOAP リクエスト

このプロパティには有効な SOAP XML リクエストを含める必要があります。デフォルトでは、XML をリテラルで指定できます。SOAP XML リクエストを動的に作成するには、[エクスプレッション] から XML を選択するか、[変数] から XML を選択します。

SOAP バージョン

このプロパティは SOAP リクエストの送信に使用する SOAP の指定のバージョンを指定します。SOAP 1.1 と SOAP 1.2 がサポートされています。SOAP 1.1 を指定する場合は、Content-Type が text/xml に設定され、(オプションの) SOAP アクションが追加の HTTP ヘッダを使用して設定されます。SOAP 1.2 を指定する場合は、Content-Type が application/soap+xml に設定され、(オプションの) SOAP アクションが Content-Type HTTP ヘッダのアクションパラメータとして設定されます。

プリエンプティブ認証

認証ヘッダーの事前送信を有効にして、認証プロセス時間を短縮します。プリエンプティブ認証が有効になっている場合は、サーバーが「401 未承認」応答を返す前に、Design Studio から基本認証またはダイジェスト認証応答が送信されます。プリエンプティブ認証を使用するには、ロボットの設定でデフォルトのクレデンシャルを設定する必要があります。

ダイジェスト認証を使用する場合、各アルゴリズムにはセッションと非セッションの 2 つのバリエーションがあります。非セッション アルゴリズム (SHA-256 など) は <name> と定義し、非セッション アルゴリズム (SHA-256-sess など) は <name>-sess のように定義します。

出力値

SOAP レスポンスを XML ページとして出力するか、XML Variable の値として出力するかを選択します。Kofax RPA 7.2 またはそれ以前のバージョンで作成されたロボットでは、これが「HTML ページとして結果を出力」になる可能性があります。これは、XML が HTML 表現に変換されることを意味しています。

オプション

ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

プロキシ切替

このアクションは以降のステップで使用されるプロキシ サーバーを変更します。そうすることで、ロボットは実行中に複数のプロキシ サーバーの間で切り替えたり、指定されたプロキシを使用したりすることができます。

このアクションを自動選択で使用するには、アクションが切り替えるプロキシ サーバーのリストを指定する必要があります。

プロキシ切替アクションを実行するたびに新しいプロキシ サーバーがリストから選択されます。プロキシ サーバーはリストに出現する順番に選択されます。最初のプロキシ サーバーはプロキシ切替アクションの初回の実行時にランダムに選択されます。

詳細については、プロキシ サービスの使用を参照してください。

プロパティ

以下のプロパティを使用してプロキシ切替アクションを設定できます：

Cookie の除去
プロキシが変更されたとき、すべての Cookie を除去するかどうかを指定します。匿名を保つためにプロキシが使用されている場合は、プロキシを変更するときに Cookie を除去する必要があります。

自動選択
オンにした場合、ステップはプロパティ ファイルから次の (ラウンド ロビン) プロキシを選択します。ステップはプロキシを選択する前に、プロキシに接続できるかどうかをテストします。自動選択のチェックが外れている場合、ステップは以下で説明するプロパティで手動で指定されたプロキシを使用します。

ホスト名
プロキシのホスト名を指定します。

ポート番号
プロキシのポート番号を指定します。

ユーザー名
プロキシの認証で使用するユーザー名を指定します。

パスワード
プロキシの認証で使用するパスワードを指定します。

除外ホスト
プロキシから除外する必要のあるホストが含まれたリストを指定します。1 つの行に 1 つのホスト名を指定するか、ワイルドカード (*) を使用して 1 つの行に 1 つのホスト名パターンを指定できます。

名前付きタグ/範囲のクリア

このアクションは選択されている名前付きタグまたは名前付き範囲、またはすべての名前付きタグおよび名前付き範囲のマークを解除し、以降のステップでそれらに名前を持たせないようにします。

プロパティ

以下のプロパティを使用して名前付きタグ/範囲のクリア アクションを設定できます：

消去する名前付きタグ**/**範囲
マークを解除する名前付きタグまたは名前付き範囲を指定します。指定の対象は、名前で選択された 1 つの名前付きタグまたは名前付き範囲、あるいは現在のウィンドウ内のすべての名前付きタグと名前付き範囲です。

Web ストレージ消去

Web ストレージ消去ステップはローカル ストレージまたはセッション ストレージあるいはその両方にあるデータを消去します。ローカル ストレージとセッション ストレージは、通常は Cookie に保存できる大量のデータを存続させるために一部の Web サイトによって使用されます。

関連するステップ アクション
Web ストレージ読み込みステップを使用して、新しいデータをローカル ストレージまたはセッション ストレージあるいはその両方に読み込むか、既存のデータを置き換えることができます。新しい値で上書きされない限り、既存のデータは除去されません。
Web ストレージ抽出ステップは、ローカル ストレージまたはセッション ストレージあるいはその両方からデータを抽出するために使用されます。

プロパティ

ローカル ストレージの消去
これがチェックされていると、ストレージ項目がローカル ストレージから消去されます。ブラウザでは、ローカル ストレージは、永続 Cookie と同様に、通常、ブラウザ セッションをまたがって存続します。

セッション ストレージの消去
これがチェックされていると、ストレージ項目がセッション ストレージから消去されます。ブラウザでは、セッション ストレージはセッション Cookie と同様に、通常はブラウザ ウィンドウまたはブラウザ タブが存在する限り、存続します。

キー パターン
特定のキーを持つ保存済みの項目のみを消去する場合は、関心の対象であるキーと照合するパターンを指定することができます。このフィールドを空白のままにすると、保存項目のキーに関係なく、すべての保存項目が消去されます。パターンを指定する場合、パターンはキー全体と照合ことに注意してください。

ドメイン パターン
特定のドメインに属する保存項目のみを消去する場合は、関心の対象であるドメインと照合するパターンを指定することができます。このフィールドを空白のままにすると、すべてのドメインの保存項目が消去されます。パターンを指定する場合、パターンはドメイン全体と照合する必要があることに注意してください。

クリック

このアクションは見つかったタグへのマウス クリックを エミュレートします。

これは、リンクに従う、フォームを送信するなど、ナビゲーションに使用する最も一般的なアクションです。このアクションは、何かをクリックしたときにロボットがブラウザと同じアクションを実行する必要があるときに使用することができます。見つかったタグに応じて、クリック アクションは、必要とされるページ読み込み、フォーム送信、JavaScript 実行などを実行します。

クリックの代わりにマウスの動きを エミュレートするには、マウス オーバーアクションおよびマウス アウトアクションを使用します。

プロパティ

ダブルクリック
ダブルクリックまたはシングル クリックのどちらをエミュレートするかを指定します。

右クリック
右クリックまたは左クリックのどちらをエミュレートするかを指定します。

座標
自動に設定されていると、ブラウザはクリックの対象となる該当する座標を選択します。あるいは、クリックする座標を正確に指定することもできます。これらの座標は、クリックされるコンポーネント (画像やボタンなど) の左上を基準としてピクセル単位で指定されます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

ウィンドウを閉じる

このアクションはウィンドウを閉じます。

Design Studio では、ウィンドウのタブを右クリックし、[ウィンドウを閉じる] を選択することによって、ウィンドウを閉じるステップを簡単に挿入することができます。

プロパティ

以下のプロパティを使用してウィンドウを閉じるアクションを設定できます：

閉じるウィンドウ
閉じる必要のあるウィンドウを指定します (ウィンドウを特定する方法の説明を参照してください)。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

変数の変換

このアクションでは、1 つまたは複数の値を変換します。データ コンバータで値を処理し、その結果を同一または別の変数に保存します。変数の変換のアクションを使用して、特定の Web サイトから抽出した値を、返される変数内で必要な値に変換します。入力変数の値を、特定の Web サイトで使用される値に変換する場合も、このアクションを使用します。

変換するすべての変数について、変換リストへの変換を追加します。各変換で、選択した変数の値をとり、それを選択したデータ コンバータに渡し、その結果を別の (または同一の) 選択した変数に書き込みます。

これら 2 つの選択した変数は同一の変数の場合と異なる変数の場合があり、どちらになるかは変換した変数値の保存場所によって決まります。変数値は、変換においてデータ コンバータを選択しないことで、1 つの変数から別の変数に簡単にコピーできます。

プロパティ

変数の変換のアクションは、次の各プロパティを使用して設定できます。

変換
使用する変換のリストを指定します。

変換のプロパティ

変数の変換のアクションは、次の各プロパティを使用して設定できます。

変換元
変換する値を保持している変数を指定します。

変換
変数の値に適用するデータ コンバータのリストを指定します。このリストは空になる場合があります (1 つの変数の値を別の変数にコピーするだけの場合など) 。

変換先
変換の結果を保存する変数を指定します。これは「変換元」プロパティで指定した変数と同じ場合もありますが、異なる場合もあります。

ファイル コピー

このアクションでは、ロボットを実行する場所のローカル ファイル システムで、ファイルをコピーします。

このアクションを行うのは、「デザイン モードでの実行」オプションを選択している場合、Design Studio 内のデザイン モードで実行中に限られるため注意してください。

プロパティ

ファイル コピーのアクションは、次の各プロパティを使用して設定できます。

ソース ファイル
これはコピーするソース ファイルのファイル システム パスか、またはファイル URL です。これは、値セレクターを使用して、さまざまな方法で指定できます。パスは絶対である必要があります。ドライブ

の名前 (該当する場合)、ディレクトリへのディレクトリ パスを含めます。あるいは、パスを `file:/C:/temp/myFile` などのファイル URL にすることもできますが、この場合はパスを URL エンコードする必要があります。区切り記号の / および \ は互いに入れ替えて使用できます。

リンク先ファイル

これはリンク先ファイルのファイル システム パスか、またはファイル URL です。これは、値セレクターを使用して、さまざまな方法で指定できます。パスは絶対である必要があります。ドライブの名前 (該当する場合)、ディレクトリへのディレクトリ パスを含めます。あるいは、パスを `file:/C:/temp/myFile` などのファイル URL にすることもできますが、この場合はパスを URL エンコードする必要があります。区切り記号の / および \ は互いに入れ替えて使用できます。

デザイン モードで実行

これが有効な場合、アクションは Design Studio 内のデザイン モードで実行されます。これが無効な場合、デザイン モードでロボットをナビゲートしても、アクションで何も起こりません。

Cookie 作成

Cookie Creator は、指定されたドメイン、パス、名前、および値 (オプション) で Cookie を作成し、これを現在の Cookie のセットに追加します。指定されたドメイン、パス、および名前の Cookie がすでに有る場合は、古い Cookie が新規作成の Cookie に置き換えられます。

プロパティ

Cookie 作成のアクションは、次の各プロパティを使用して設定できます。

ドメイン

Cookie のドメインを指定します。ドメインは、値セレクターを使用してさまざまな方法で指定できます。

パス

Cookie のパスを指定します。パスは、値セレクターを使用してさまざまな方法で指定できます。

名前

Cookie の名前を指定します。名前は、値セレクターを使用してさまざまな方法で指定できます。

値

Cookie の値を指定します。値セレクターを使用して複数の方法で値を指定することができます。このプロパティはオプションです。

セキュリティ

選択すると、Cookie は HTTPS 経由で所定のドメインから読み込まれるときに送信されます。設定していない場合は、Cookie は HTTP 経由で所定のドメインから読み込まれるときに設定されます。

HTTP 限定

選択すると、Cookie は HTTP 限定の Cookie になります。つまり、Cookie は HTTP (または HTTPS) 送信要求のときに限り使用されます。ただし、その値はクライアント側のスクリプト (JavaScript など) では利用できません。

ページ生成

ページ生成のアクションでは、現在のウィンドウで古いページを置き換える新しいページを作成します。ページの処理過程は、ページ読み込みのステップの動作と同様です。また、JavaScript の実行がオプ

ションで無効ではない場合、新規作成ページの HTML 内にある JavaScript が実行されます。ページ生成のアクションは、たとえば、XML ドキュメントなどの非 HTML ページを読み込むためにも使用します。

プロパティ

ページ生成のアクションは、次の各プロパティを使用して設定できます。

コンテンツ
新しいページのコンテンツは、値セレクターを使用してさまざまな方法で指定できます。たとえば、変数からコンテンツを取得することが可能で、変数がテキストやバイナリ コンテンツを伴う場合も同様です。コンテンツのタイプ (HTML など) は自動的に検出されます。自動検出が不十分である場合、またはコンテンツを別の方法で読み込む場合 (たとえば、HTML ドキュメントをプレーンテキストとして読み込む場合) は、オプションでコンテンツのタイプの検出を無効にすることができます。

ページの **URL**
ここで、新しいページのページ URL を指定します。これは特に、関連するリンクやページ内のリソース リファレンスを解決するために使用します。これは、値セレクターを使用して、さまざまな方法で指定できます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションは、ロボットのオプションよりも優先されます。オプション ダイアログでアステリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションはロボットに対して指定されているものと同様になります。

ファイル削除

このアクションではロボットが実行されるローカル ファイル システム上のファイルまたはディレクトリを削除します。

このアクションを行うのは、「デザイン モードでの実行」オプションを選択している場合、Design Studio 内のデザイン モードで実行中に限られることに注意してください。

プロパティ

次のプロパティを使用して [ファイル削除] アクションを設定できます。

ファイルまたはディレクトリ
これは、削除されるファイルまたはディレクトリのファイル システム パスまたはファイル URL です。これは、値セレクターを使用して、さまざまな方法で指定できます。パスは絶対である必要があります。ドライブの名前 (該当する場合)、ディレクトリへのディレクトリ パスを含めます。あるいは、パスを file:/C:/temp/newDir などのファイル URL にすることもできます。その場合はパスを URL エンコードする必要があります。区切り記号の / および \ は互いに入れ替えて使用できます。

空でないディレクトリを削除
これはディレクトリを削除する場合にのみ意味を持つオプションです。これが選択されていないと、アクションは空のディレクトリのみを削除します。これが選択されていると、アクションはすべてのファ

イルおよびサブディレクトリを含むディレクトリを削除し、すべてのサブディレクトリに対してもこの操作を再帰的に実行します。

デザイン モードで実行
これが有効になっていると、アクションは Design Studio 内部のデザイン モードでも実行されます。これが無効になっていると、ロボットをデザイン モードでナビゲートするときにアクションは何も実行しません。

データベース データ削除

このステップでは、データベースに以前保存された値を削除することができます。データベース接続の設定は「設定」で行います。

プロパティ

データベース
削除する値が保存されているデータベースを指定します。変数、エクスプレッションまたはコンバータを使用して、デザイン時に値を選択またはハード コーディングするか、ランタイム時に動的にデータベース名を作成します。ロボットが実行されたときに、その名前を持つデータベースが存在しない場合はエラーが発生します。

変数
削除する値が含まれたコンプレックス タイプの変数を選択します。

キー
削除する値の一意のキーを指定します。キーは (変数タイプで属性を「データベース キーの一部」としてマークすることによって) 変数で定義することや、値セレクターを使用して定義することもできます (値パラメータを除く)。

デザイン モードで実行
これが有効になっていると、ステップは Design Studio 内部のデザイン モードでも実行されます。これが無効になっていると、ロボットをデザイン モードでナビゲートするときにステップは何も実行しません。

Desktop Automation

このステップは廃止されているため、代わりに「ロボットを呼び出す」ステップを使用する必要があります。

古い *Desktop Automation* アクション ステップの変換

Kofax RPA バージョン 10.7 0以前では、Desktop Automation アクション ステップに含まれるロボット ワークフローを編集する際に、スタンドアロンの Desktop Automation エディターを使用していました。Kofax RPA 11.0.0 以降では、バージョン 10.7.0 以前で作成された Desktop Automation アクション ステップを実行することができます。ただし、ワークフローを編集するには、ステップから新しいロボット にワークフローを抽出し、Desktop Automation ステップを、新しいロボットを参照する「ロボットを呼び出す」ステップに変更する必要があります。

エクスポートは、ベーシック エンジン ロボット および Desktop Automation ステップを含むスニペットで実行できます。

エクスポート プロセスを元に戻すことはできないため、ロボットにエクスポートを行った後に最初の Desktop Automation ステップに戻すことはできません。

1. アクション ステップをロボット に変換するには、プロジェクト ツリーで Desktop Automation ステップを含む ベーシック エンジン ロボット を右クリックし、[ロボットのエクスポート] をクリックします。シングル ロボットを開いて Desktop Automation ステップを選択した場合、ステップのプロパティで [プレビュー] をクリックすると、後に現れるロボット をプレビューすることができます。
   - 複数のロボットで作業する場合のために、複数のロボット を一度にエクスポートできます。プロジェクト ビューで任意のフォルダを右クリックするか、Desktop Automation ステップを含む複数のベーシック エンジン ロボット を選択して右クリックし、[ロボットのエクスポート] をクリックします。

   新しいダイアログ ボックスが表示され、抽出元となる、検索されたすべての Desktop Automation ステップが一覧表示されます。
2. 必要に応じて、後に現れるロボット にわかりやすい名前を新規に割り当てて、後に現れるロボットをプレビューし、含まれるベーシック エンジン ロボット の Desktop Automation ステップを確認できます。
   - 選択したファイルの名前を変更するには、 をクリックします。
   - エクスポート後に作成される、選択したロボット をプレビューするには、 をクリックします。
   - 含まれるベーシック エンジン ロボット で選択した Desktop Automation ステップを表示するには、 をクリックします。

   ワークフロー プレビュー ウィンドウでズーム レベルを変更すると、ロボット エディターのズーム レベルも変更されます。
3. [次へ] をクリックして、現在のプロジェクト内にあるエクスポートされたロボットの場所を選択します。
4. [終了] をクリックしてダイアログ ボックスを閉じ、エクスポートを開始します。
   エクスポートの概要が表示されて、エクスポートしたロボットおよびスニペットの数が一覧表示されます。ベーシック エンジン ロボット で設定された必要なデバイスは、ロボット に自動的に表示されます。

テキスト分割

[テキスト分割] アクションは、パターンと式を使用して、検出されたタグに含まれているテキストを複数の部分に分割します。このアクションは、分割されたテキストの部分を次以降のステップでループする場合に便利です。

プロパティ

次のプロパティを使用して [テキスト分割] アクションを設定できます。

パターン
検出されたタグ内のテキストと照合されるパターンを指定します。パターンの一致ごとに出力エクスプレッションフィールドのエクスプレッションが評価されます。次に、検出されたタグが タグに置

き換えられます。このタグには、パターンの一致ごとに別の タグに囲まれたエクスプレッションの結果が含まれます。

大文字と小文字を無視
このプロパティをオンにすると、パターンの照合で大文字と小文字の区別が無視されます。

出力エクスプレッション
このフィールドにはパターンの一致ごとに評価されるエクスプレッションが含まれます。

例

検出されたタグがこのページの "Kofax RPA" というテキストであり、

```
<html>
  <body>
    <p>
      Kofax RPA
    </p>
  </body>
</html>
```

パターンが "\S+\s?"(1 つ以上の非スペース文字に続けてオプションの 1 つのスペース) に設定されており、出力エクスプレッションが "$0" (一致したテキスト全体) に設定されている場合、出力値は次のようになります。

```
<html>
  <body>
    <p>
      <span>
        <span>Kofax</span>
        <span>RPA</span>
      </span>
    </p>
  </body>
</html>
```

コメント

[コメント] アクションでは何も実行しません。

このアクションはロボットにコメントを追加するときに便利です。

プロパティ

このアクションにはプロパティがありません。

ステップ終了

[ステップ終了] はロボット内の分岐の終了点をマークします。

このステップをクリックすると分岐内のすべてのステップを実行できるため、このステップは分岐の終了点に配置するマーカーとして便利です。このマーカーがない場合、分岐の最後のステップを実行できるようにするには、別のステップを終了点に挿入する必要があります。

[ステップ終了] はいずれのアクションとも関連がなく、その実行に関して言えば、なし アクションとともに実行されるアクション ステップに相当します。[ステップ終了] は削除できませんが、複数の分岐が同じ [ステップ終了] を共有することはできます。

[ステップ終了] によって口ボットの実行が停止されることはありません。口ボットの実行を停止させるには、アクション ステップを [終了] アクションとともに使用する必要があります。

パスワード入力

このアクションはフォームのパスワード フィールドにパスワードを入力します。

このアクションはテキストを入力アクションと似ています。ただし、固定パスワードが指定されたときに Design Studio および保存された口ボット ファイルでパスワードがユーザーに表示されない点が異なります。

検出されたタグはパスワード フィールドである必要があります。

パスワード フィールドに対して登録されたイベント ハンドラーがある場合、パスワードを入力すると、JavaScript の実行がトリガーされる可能性があることに注意してください。

プロパティ

次のプロパティを使用して [パスワード入力] アクションを設定できます。

入力するパスワード
パスワード フィールドに入力するパスワードを指定します。値セレクターを使用して複数の方法で値を指定することができます。

次によりフォーカスを取得
選択されているタグに口ボットがフォーカスを設定する方法を示します。

入力する前にすべてのテキストを選択
新しいテキストを入力するときに上書きされるようにするには、フィールド内の既存のテキストを選択する必要がある場合があります。
このオプションを使用して、新しいパスワードを入力する前に、パスワード フィールドの既存のテキストを選択します。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームの工レメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

テキストを入力

このアクションは検出されたタグにテキストを入力します。検出されたタグはテキスト フィールド、テキスト エリア、パスワード フィールド、ファイル フィールド、非表示フィールド、またはコンテンツが編集可能なタグである必要があります。

パスワード フィールドに固定パスワードを入力する場合は、このアクションの代わりにパスワード入力アクションを使用することを検討してください。 これは、Design Studio および保存されたロボット ファイルでパスワードをユーザーに見られることを防止するためです。

フィールドまたはタグに対して登録されたイベント ハンドラーがある場合、テキストを入力すると、JavaScript の実行がトリガーされる可能性があることに注意してください。

プロパティ

次のプロパティを使用して [テキストを入力] アクションを設定できます。

入力するテキスト
入力するテキストを指定します。値セレクターを使用して複数の方法で値を指定することができます。

次によりフォーカスを取得
一部の Web サイトでは、テキストを入力する前にフォーカスを入力フィールドに設定する必要があります。入力フィールドにフォーカスを設定するには、[次によりフォーカスを取得] オプションを使用します。

入力する前にすべてのテキストを選択
新しいテキストを入力したときに既存のテキストが確実に上書きされるように、フィールド内の既存のテキストを選択する必要が生じることがあります。フィールド内の既存のテキストを選択するには、[入力する前にすべてのテキストを選択] オプションを使用します。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

コマンド ライン実行

このアクションはコマンド ライン (外部プログラム) を実行します。

プロパティ

次のオプションを使用して [コマンド ライン実行] を設定できます。

コマンド ライン
実行するコマンド ライン。値セレクターを使用して値を設定します。Windows では "cmd.exe /C" の引数としてコマンド ラインを使用できます。その他のプラットフォームでは、"/bin/sh -c" の引数としてコマンド ラインを使用します。

抽出
プログラムがコンソールにテキストを書き込むと、テキスト抽出はここで設定されます。オプションを以下に示します。

- "Nothing" は何も抽出しません。
- "Stdout" は stdout に書き込まれたテキストを抽出し、変数に保存します。

- "Stderr" は stderr に書き込まれたテキストを抽出し、変数に保存します。
- "Separate stdout and stderr" は stdout と stderr に書き込まれたテキストを抽出し、2 つの別々の変数に保存します。
- "Joined stdout and stderr" は stdout に書き込まれたテキストを抽出し、1 つの変数に保存します。

プログラムが非 ASCII 文字をコンソールに書き込む場合は、テキストの読み込みに使用されるエンコーディングを指定できます。Windows の西ヨーロッパ バージョンでは、たいていの場合、コンソールは "Latin-1, MS-dos, with Euro" とも呼ばれる cp858 を使用します。その他のプラットフォームでは、たいていの場合、コンソール テキストを読み取るときに utf-8 をエンコーディングに使用する必要がありますが、これは環境に固有です。

ここに終了コードを保存

終了コードの保存先となる変数を指定します。プログラムは実行を終了すると、実行の状態を示す終了コードを返します。0 は成功を意味し、その他の値は何らかのエラーを示しますが、エラーの意味はプログラムに固有です (ただし、ある程度の合意はあります。たとえば、値 2 は通常「ファイルが見つかりません」を意味します)。変数が指定されていない場合、終了コードは破棄されます。

デザイン モードで実行

これが有効になっていると、アクションは Design Studio 内部のデザイン モードでも実行されます。これが無効になっていると、ロボットをデザイン モードでナビゲートするときにアクションは何も実行しません。

プログラムは作業ディレクトリと環境を Design Studio から継承します。

ステップにタイムアウトはなく、ステップは外部プログラムが完了するまで待機します。

JavaScript の実行

このアクションは現在のページ上の JavaScript またはユーザー自身のカスタム JavaScript を実行します。

ほとんどのステップは関連する JavaScript を操作の一部として自動的に実行するため、JavaScript を実行する特別な必要がない場合、通常は [JavaScript の実行] アクションを使用する必要はありません。

[JavaScript の実行] アクションは HTML に JavaScript を含める次の方法をサポートしています。

- 実行される JavaScript の複数行を含めることができる <script> タグ。外部 JavaScript ファイルを参照することもできます。
- 特殊属性としてタグに出現する可能性があるイベント ハンドラ。onClick や onMouseOver のように必ず「on」で始まります。JavaScript からイベント ハンドラーをアタッチして HTML ソース上で非表示にすることもできます。
- JavaScript を指定する JavaScript の URL: `<a href="javascript:open()">` などの URL が存在する可能性のある、タグ属性の値を持った JavaScript プロトコル

プロパティ

次のプロパティを使用して [JavaScript の実行] アクションを設定できます。

JavaScript
このプロパティによって、実行する JavaScript を指定します。

- すべての **<script>** タグ内の **JavaScript** は、現在のページ内のすべての <script> タグを実行します。
- 選択された **<script>** タグ内の **JavaScript** は 1 つの検出された <script> タグを実行します。
- **URL** 内の **JavaScript** は JavaScript: Protocol による指定に従って JavaScript URL 内の JavaScript を実行します。
- イベント ハンドラー内の **JavaScript** はタグに含まれたイベント ハンドラー内の JavaScript を実行します。実行する必要のある特定のイベント ハンドラをドロップダウン ボックスから選択する必要があります。
- カスタム **JavaScript** はユーザー自身のカスタム JavaScript を実行します。
- エクスプレッションからのカスタム **JavaScript** はユーザー自身のカスタム JavaScript を実行します。これは、固定テキストの代わりにエクスプレッションを入力できる点を除いて、カスタム Javascript と同様です。JavaScript 関数のリストについては、「JavaScript を使用して変換」を参照してください。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

SQL 実行

[SQL 実行] アクションでは、SQL ステートメントをデータベースに送ります。オプションで、影響を受けた行の数を変数に保存します。データベースが返したその他の結果は無視されます。言い換えると、このアクションの目的は SQL のステートメントの挿入、アップデート、および削除です。このアクションを使用して、ストアド プロシージャなどを呼び出すこともできます。SQL の指定にはエクスプレッションを使用します。

- Design Studio でデザイン モードを使用している場合、SQL ステートメントは実行されません。
- SQL ステートメントは、ステップごとに 1 つずつ、個別のトランザクション内で実行されます。複数の連続する SQL 実行の呼び出し、またはデータベース照会の呼び出しにまたがるトランザクションを使用することはできません。
- 「 SQL 実行」は現在、準備済みの (バインド) ステートメントまたは呼び出し可能なステートメントをサポートしていません。

SQL ステートメント内の `use <Some Other DB>` コマンドは実行しないでください。これ以降のすべての SQL コマンド (同一の口ボット内など) の結果が変更されてしまいます。

プロパティ

[SQL 実行] アクション は、次の各プロパティを使用して設定できます。

データベース
このアクションでクエリを送るデータベースを、Design Studio で利用できるデータベースのドロップダウン リストを使用して選択します。

SQL
このフィールドには、有効な SQL ステートメントがエクスプレッションの形式で含まれる必要があります。このエクスプレッションの値が、選択したデータベースに送られます。

タイムアウト
SQL ステートメントを送信するタイムアウトを秒単位で指定します。変数または値のオプションを選択して、希望するタイムアウトを定義します。値はゼロより大きい必要があります。
タイムアウトは DBMS を通じて強制され、タイムアウトに達すると、[SQL 実行] アクションの実行が停止されて、データベース固有のエラー メッセージが表示されます。

それぞれの DBMS タイプおよび JDBC ドライバーに応じて、タイムアウトの処理方法が異なる場合があります。たとえば、一部のシステムにおいては、特定のステートメントの途中でのタイムアウトの発生が許可されていないこともあります。

変更された行
SQL ステートメントの影響を受けた行の数を保存する、テキストまたは整数の変数を選択します。これはオプションです。

デザイン モードで実行
これが有効になっていると、ステップは Design Studio 内部のデザイン モードでも実行されます。これが無効な場合、デザイン モードでロボットをナビゲートしても、ステップで何も起こりません。

抽出

[抽出] アクションでは、テキストを抽出してそれを変数に保存します。

テキストのみ、またはタグを含めて全部など、抽出するコンテンツを指定する場合があります。テキストを保存する前に、データ コンバータのリストを使用して処理することができます。また、オプションで先頭および末尾のスペースを削除できます。

「抽出」アクションの最も簡単な使用方法は、1 つの検知タグから抽出することです。また、タグ範囲から抽出することもできます。この場合、1 つの検知タグから別の検知タグへとすべてのタグを使用します。

プロパティ

[抽出] アクションは、次の各プロパティを使用して設定できます。

抽出元
検知タグの抽出する部分を指定します。
- 見つかったタグ: 検知タグの全体を抽出するように指定します。

- タグの範囲: タグの一定の範囲を抽出するように指定します。開始タグおよび終了タグを選択したり、これらのタグを範囲に含めるかを選択できます。

次を抽出

抽出するコンテンツを指定します。

- テキストのみ: テキストだけを抽出するように指定します。
- 構造化テキスト: テキストだけを抽出し、ブラウザに表示される形式と同様の形式でテキストを構造化します。システムで見出しの場所を推測し、テキストを前後に挿入できます。次のオプションを設定できます。

  位置指定されたテーブルとイメージを含める

  テキストの右端または左端に位置合わせされた表および画像を出力テキストに含めるよう指定します。このプロパティを無効にすると、目的のコンテンツが削除されることがあります。

  URL を含める

  リンク タグ内の実際の URL を出力テキストに含めるよう指定します。

  イメージ テキストの代替要素を含める

  画像のテキスト表現を出力テキストに含めるよう指定します。

  フォーム フィールドを含める

  フォーム フィールドのテキスト表現を出力テキストに含めるよう指定します。

  見出しの前にこれを挿入

  このアクションで見出しの場所を推測し、指定のテキストを見出しの前に挿入するように指定します。

  見出しの後にこれを挿入

  このアクションで見出しの場所を推測し、指定のテキストを見出しの後に挿入するように指定します。

- [高度な構造化テキスト]: テキストだけを抽出し、ブラウザに表示される形式と同様の形式でテキストを構造化します。タグの名前は任意のテキストに変換できます。次のオプションを設定できます。

  位置指定されたテーブルとイメージを含める

  テキストの右端または左端に位置合わせされた表および画像を出力テキストに含めるよう指定します。このプロパティを無効にすると、目的のコンテンツが削除されることがあります。

  URL を含める

  リンク タグ内の実際の URL を出力テキストに含めるよう指定します。

  イメージ テキストの代替要素を含める

  画像のテキスト表現を出力テキストに含めるよう指定します。

  フォーム フィールドを含める

  フォーム フィールドのテキスト表現を出力テキストに含めるよう指定します。

  タグ変換

  使用するタグ変換を指定します。タグ変換の形式は tag=text です。たとえば、"<h1>=<head1>" および "</h1>=</head1>" は、HTML の見出しレベル 1 を特殊な <head1> タグに変換します。変換の右側は任意です。通常のタグにする必要はありません。

- **HTML**: HTML の全体を抽出するように指定します。

HTML を書式設定
HTML をプリティプリントするように指定します。

URL をエンコード
属性値の URL を HTML エンコードするように指定します。これを強く推奨します。その理由は、さまざまなブラウザで一貫して機能するように、標準に準拠した HTML を生成する必要があるためです。ただし、URL の認識や比較のための処理が単純な HTML では、URL をエンコードせずにそのままにすることが必要な場合もあります。

相対 **URL** を抽出
すべての URL を相対で抽出することを指定します。URL のベース部分がある場合には除去されます。

- **XML**: XML の全体を抽出するように指定します。これはページが XML ページの場合に限り機能します。

  XML 宣言を含める
  XML 宣言 (たとえば、<?xml version="1.0" encoding="UTF-8"?>) がある場合は、これを抽出した XML に含めるように指定します。つまり、XML ドキュメントの一部を抽出して、適切な宣言をトップにして、新規 XML ドキュメントを取得できます。

コンバータ
テキストを処理するデータ コンバータのオプション リスト。

スペースの除去
選択した場合、テキストの先頭および末尾のスペースを除去してから、テキストを変数に保存します。

変数
抽出したテキストを保存する変数を指定します。

HTML として抽出

[HTML として抽出] アクションでは、スプレッドシートのドキュメントの一部を HTML テーブルとして抽出し、変数に保存します。ステップの範囲ファインダーで、抽出する対象を決定します。これは 1 つのシートの一部の場合もあり、情報のプロパティすべての場合もあります。

プロパティ

[HTML として抽出] アクションは、次の各プロパティを使用して設定できます。

ヘッダーを含める
このプロパティで、生成したテーブルにスプレッドシートのヘッダー (1、2、3、4 および A、B、C、D など) を含めるかを指定します。

次を抽出
セルから抽出する対象を指定します。ただし、情報のプロパティには無効です。次の 3 つから選択できます。

- [書式設定された値] では、書式設定された表示を抽出するように指定します。たとえば、数は表示されるとおりに抽出して、余分な小数点を伴う内部表現は抽出しません。
- [プレーン値] では、内部表現を抽出するように指定します。たとえば、数は全桁で抽出し、日付は 1900 年 1 月 1 日以降の日数として抽出します。書式設定された値とプレーンな値で差がないセル、たとえば、テキスト値や論理値を含むセルでは、抽出した値は [書式設定された値] と同じです。

- [式] は、式を抽出するように指定します。式を含まないセルでは、抽出される値は [プレーン値] の場合と同じです。

変数
生成した HTML テーブルを保存する変数。

バイナリ コンテンツ抽出

このアクションでは、ブラウザ ビューからバイナリ コンテンツを抽出します。ステップでバイナリ コンテンツをブラウザ ビューに読み込んだ場合、このデータは表示されません。ただし、次のステップの [バイナリ コンテンツ抽出] を使用して、データをバイナリ変数に抽出できます。

プロパティ

[バイナリ コンテンツ抽出] アクションは、次の各プロパティを使用して設定できます。

データを次の場所に保存
バイナリ コンテンツを保存する変数を指定します。これはバイナリ変数である必要があります。

コンテンツ タイプ
データのコンテンツ タイプを保存する変数。

ファイル名
データを読み込んだ元の場所に基づいて、元のファイル名を保存するための変数 (URL に基づくファイルの名前など) 。この名前は、たとえば、ロボットの次のステップで、ファイルへの保存が発生する場合に必要です。

セル値抽出

[セル値抽出] アクションでは、スプレッドシートのドキュメントからコンテンツを抽出し、それをデータ コンバータのリストに通し、その結果を変数に保存します。

プロパティ

[セル値抽出] アクションは、次の各プロパティを使用して設定できます。

次を抽出
セルから抽出する対象を指定します。次の 3 つから選択できます。

- [書式設定された値] では、書式設定された表示を抽出するように指定します。たとえば、数は表示されるとおりに抽出して、余分な小数点を伴う内部表現は抽出しません。
- [プレーン値] では、内部表現を抽出するように指定します。たとえば、数は全桁で抽出し、日付は 1900 年 1 月 1 日以降の日数として抽出します。書式設定された値とプレーンな値で差がないセル、たとえば、テキスト値や論理値を含むセルでは、抽出した値は [書式設定された値] と同じです。
- [式] は、式を抽出するように指定します。式を含まないセルでは、抽出される値は [プレーン値] の場合と同じです。

コンバータ
コンテンツを処理するデータ コンバータのオプション リスト。

変数
値を割り当てる変数。

データ行の列を抽出

このアクションで、現在の範囲内のセルからデータを抽出して変数に入れます。範囲は、データ行繰り返しステップで作成した CSV ファイルの 1 行です。[データ行の列を抽出] ステップを使用するには、最初にデータ行繰り返しステップを作成および設定します。

プロパティ

[ファインダー] タブ

- コンテキスト：データ行繰り返しステップからのセルの範囲の名前。
- 列：列の名前またはそのインデックス (1 から開始)。

[アクション] タブ

- コンバータ：コンテンツを処理するデータ コンバータのオプション リスト。
- 変数：値を割り当てる変数。
- [列がない場合エラー]: 選択すると、行が短すぎる場合にエラーが生成されます。たとえば、4 列の行を取得しようとして 3 列しか取得されなかった場合は、エラーが生成されます。

Cookie 抽出

Cookie 抽出アクションでは、現在の Cookie のセットから Cookie の値を抽出します。Cookie は、ドメイン、パス、および名前の正規表現を使用して選択します。複数の Cookie がパターンに一致した場合、最初に一致した Cookie から値を抽出します。

プロパティ

Cookie 抽出アクションは、次の各プロパティを使用して設定できます。

ドメイン パターン
Cookie のドメインに一致するパターンを指定します。パターンは Cookie のドメイン全体に一致する必要があります。

パス パターン
Cookie のパスに一致するパターンを指定します。パターンは Cookie のパス全体に一致する必要があります。

名前パターン
Cookie の名前に一致するパターンを指定します。パターンは Cookie の名前全体に一致する必要があります。

変数
抽出した値を保存する変数を、ここで指定します。

フォームパラメータを抽出

[フォームパラメータを抽出] アクションでは、検知タグ内のフォームの URL からパラメータから抽出して、値を変数に保存します。

プロパティ

[フォームパラメータを抽出] アクションは、次の各プロパティを使用して設定できます。

フォーム パラメータの名前
フォーム パラメータの名前を指定します。

コンバータ
抽出した値を変数に保存する前に適用するデータ コンバータ。

変数
得られた値を保存する変数。

URL で使用するエンコード
URL で使用する文字エンコードを指定します。抽出した値に使用できない文字が含まれる場合は、エンコードを、URL を含むページ、または URL が作成されたフォームを含むページで使用されるエンコードに変更してください。フォームの URL で最も一般的に使用されるエンコードは Unicode (UTF-8) および Latin-1 (ISO-8859-1) です。

例
この検知タグは次のように考えます。

```
<a href="http://www.abc.com/search?author=Johnson">
  Search
</a>
```

フォーム パラメータの名前が "author" にセットされている場合、値 "Johnson" を抽出して、選択した変数に保存します。

電子メールから抽出

このアクションは、保存された電子メールから情報を抽出します。このステップでは、入力として、文字列または MIME 形式で `.eml` ファイルに保存された電子メール メッセージを含むテキスト変数を使用します。Outlook `.msg` 形式からの抽出はサポートされていません。

メッセージ ファイルに HTML バージョンとプレーン テキスト バージョンが含まれている場合、このステップは、電子メールのフレームを 2 つ作成します (HTML 表現とプレーン テキスト表現)。

抽出された電子メールはブラウザで開かれ、HTML ファイルまたはプレーン テキスト ファイルからの抽出と同様に、情報を抽出できます。画像以外の添付ファイルはアンカー タグに追加されます。ユーザーは添付ファイルを変数に追加したり、添付ファイルのタイプが Kofax RPA でサポートされている場合は、開いて編集したりすることができます。

たとえば、添付された PDF、Excel、または HTML ドキュメントを編集できます。電子メールに添付ファイルとして画像が含まれている場合は、インライン画像要素として追加されます。[画像からテキストを抽出] ステップを使用するなど、画像を開いてそこから情報を抽出することもできます。

Unicode 文字を含む `.eml` ファイルで [ファイル読み込み] ステップを使用すると、エンコードが正しく処理されず、エンコードが不適切なテキスト値が発生することがあります。この問題が発生した場合は、バイナリ変数を用いた [ファイル読み込み] ステップを使用し、その後コンバータ ステップでエンコーディングを手動で設定します。データ コンバータ の「バイナリをテキストに変換」オプションを参照してください。

プロパティ

次のプロパティを使用して、[電子メールから抽出] アクションを設定します。

電子メールを含む変数

文字列または MIME 形式の電子メールを含むテキスト変数を選択します。

すべてのヘッダーを含める

デフォルトでは、ステップには送信者、宛先、件名、日付、メッセージ ID などの基本的なヘッダー情報のみが含まれます。このオプションを選択すると、抽出された電子メールにメッセージ ヘッダー全体が含まれます。

オプション

- ロードする画像

  電子メールとともにロードされる画像をフィルタリングします。**[なし]**、**[すべて]**、および **[URL** に依存**]** から選択します。

  [**URL** に依存] を選択した場合は、+ 記号をクリックし、URL をフィルタリングするための条件を指定します。
- [詳細]

  [詳細] をクリックして、ステップのデフォルト オプションを編集します。

PDF から抽出

このアクションでは、選択したバイナリ変数内にバイナリ データとして含まれる PDF ドキュメントから、テキストおよび画像抽出します。

通常は、ターゲット抽出ステップを使用して、PDF ドキュメントをダウンロードして変数に入れます。[PDF から抽出] アクションから出力されるのは、PDF ドキュメントから抽出したテキストおよび画像を含む HTML ページです。これ以降の各ステップで、必要な情報をページから抽出できます。方法は他の HTML ページと同様です。

次のことに注意してください:

- PDF ドキュメントにはテーブルや段落などの構造情報は含まれず、テキストやグラフィックの位置のみが含まれますが、これらがテーブルや段落のように見える場合があります。これにより、PDF ドキュメントから必要な情報を抽出するのが難しくなる場合があります。ただし、「PDF から抽出」ステップは、ヒューリスティックスを使用することで、利用可能な位置情報に基づき、テキストを HTML のパラグラフにグループ化します。
- [PDF から抽出] ステップでは、フォームに入力されたデータを抽出できません。フォーム データを抽出できるようにするには、サードパーティ ツールを使用してドキュメントをフラット化する必要があります。

プロパティ

[PDF からテキスト抽出] アクションは、次の各プロパティを使用して設定できます。

PDF 変数

PDF ドキュメントをバイナリ データとして含むバイナリ変数。

画像を含める

埋め込まれた画像抽出するかを指定します。PDF ドキュメントからすべての画像やグラフィックを抽出できないこともあり、これは元のドキュメントへの組み込み方法に応じて異なります。

Form XObjects を含める

このオプションで、PDF から Form XObjects を抽出できます。Form XObjects は、PDF ファイル内のオブジェクトをグループ化します。オブジェクトには、テキスト、画像、ベクター要素などが含まれるこ

とがあります。Form XObjects は通常、ドキュメント内で複数回参照されるオブジェクトを保存するために使用します。

位置を含める
各テキストの位置を抽出するかを指定します。これらの各位置が、ドキュメントの構造を引き出すために有効な場合があります。

フォーマットを含める
テキストのフォーマット (フォントの名前、サイズなど) を抽出するかを指定します。各位置と同様に、フォーマットはドキュメントの構造を引き出すために有効な場合があります。

テキストのマージ
デフォルトで、PDF から HTML を生成したコンバータは、同一ラインにあるテキストを 1 つの HTML 要素にマージします。PDF ドキュメント内で異なるテキストとして表示される場合も同様です。この機能は通常は望ましいものですが、場合によっては別の作用を及ぼします。つまり、元は離れた場所にあったテキストがマージされてすぐ隣に表示されることがあります。この機能をオフしておくことが望ましい典型的な例は、ドキュメントに複数の列が含まれる場合です。この機能をオフにすると、列の構造を維持しようとします。

ハイパーリンク抽出

このアクションで、スプレッドシート内のセルからハイパーリンクを抽出します。

プロパティ

[ハイパーリンク抽出] アクションは、次の各プロパティを使用して設定できます。

コンバータ
コンテンツを処理するデータ コンバータのオプション リスト。

変数
値を割り当てる変数。

画像抽出

このアクションでは、画像を検知タグから抽出して、変数またはファイルに保存します。

また、オプションで、実際のコンテンツ タイプと抽出した画像ファイルの名前を他の変数に保存できます。

プロパティ

[画像抽出] アクションは、次の各プロパティを使用して設定できます。

次の場所に保存
抽出した画像の保存場所を指定します。以下の 2 つから選択できます。

- 変数
  抽出したデータを保存する変数を指定します。変数のタイプは画像またはバイナリである必要があります。抽出した画像のプレビューを変数ビューで見ることができるので、特化した画像変数の使用を推奨します。

ファイル

データを書き込むファイルを指定します

ファイル名

ファイルの名前と拡張子を指定します。

自動

このオプションでは、次の方針でファイルの名前を自動的に生成します。

1. まず、レスポンスのコンテンツ配置ヘッダーにファイルの名前のパラメータが有るかを確認し、有る場合はその名前を使用します。
2. 次に、URL にファイルの名前が含まれるかを確認し、有る場合はその名前を使用します。
3. 上のオプションがどれも成功しなかった場合は、エラーが生成されます。

値、変数、エクスプレッション、コンバータ

値セレクターを使用してさまざまな方法で値を指定することができます。

ディレクトリ

ファイルを置くディレクトリを指定します。値セレクターを使用して複数の方法で値を指定することができます。

ディレクトリを作成

存在しないディレクトリをすべて、指定したパスに作成するかを指定します。このオプションを選択すると、ディレクトリが作成されます。このオプションを選択しない場合は、ディレクトリが存在している必要があり、存在しないときにエラーが生成されます。

上書きの方針

選択したファイルがすでにある場合にどうするかを指定します。

ファイルを上書きする

既存のファイルがある場合は置き換えられます。

ファイルを上書きしない

既存のファイルは置き換えられなくなります。ファイルが存在する場合は、エラーが生成されます。

新しいファイルを作成

このオプションで、新規ファイルが必ず作成されるようになります。選択した名前のファイルがすでに存在する場合は、新しい固有のファイルの名前が、その新規ファイルに対して生成されます。この新しいファイルの名前は、選択した元のファイルの名前に、拡張子の直前の末尾にシリアル番号を追加したものになります。たとえば、元のファイルの名前 myImage.png に _1 を追加して myImage_1.png になります。

メタ データを次の場所に保存

抽出した画像についてのメタ データを保存する変数を指定します。

コンテンツ タイプ

画像のコンテンツ タイプを保存するオプション変数を指定します。たとえば、コンテンツ タイプは次のようになります: image/gif

ファイル名
抽出した画像のファイルの名前を保存するオプション変数を指定します。画像をファイルに保存する場合、そのファイルの名前は実際に使用するファイルのフルパスになります。画像が変数に読み込まれると、そのファイルの名前は、元のリソースのファイルの名前 (URL またはレスポンスのコンテンツ配置ヘッダーから取得) になります。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

JSON 抽出

[JSON 抽出] ステップでは、JSON ファインダーが見つけた JSON 値の一部を、抽出して JSON 値として変数に格納します。抽出した値自体は、オブジェクト、配列、基本値などの有効な JSON である必要があります。たとえば、名前/値のペア "answer" の場合: 42 は、ビューで選択できる場合でも抽出することはできません。このような名前 / 値のペアを選択して抽出しても、値 (42) が抽出されるだけです。

プロパティ

[JSON 抽出] アクションは、次の各プロパティを使用して設定できます。

コンバータ
テキストを処理するデータ コンバータのオプション リスト。

変数
抽出したテキストを保存する変数を指定します。

パス抽出

「パス抽出」ステップは、ステップのファインダーが検知した要素の絶対パスを抽出します。このパスは、一般的な意味でのパスと考えることができます。HTML および XML では、html.body.div.text などのタグ パスです。Excel では、Sheet1!A5:B7 などの範囲です。JSON では、@top:.a[5].b などの JSON ファインダーで使用されるパスです。抽出されるパスは常に絶対パスです。つまり、名前付きタグ、範囲などからの相対パスではなく、ワイルドカード (*) 記号が含まれることはありません。この種類のパスは、HTML/XML でのタグ パス表示や Excel でのセル範囲表示など、ページ表示の最下部にあるターゲット ビューで確認できます。

抽出したパスは、「パス」プロパティ内の他のステップのファインダーで、パスを含む変数を参照して、利用される場合があります。

プロパティ

「パス抽出」アクションは、次の各プロパティを使用して設定できます。

変数
抽出したパスを保存する変数を指定します。

プロパティ名抽出

「プロパティ名抽出」ステップでは、JSON ファインダーが見つけた JSON 値のプロパティ名を、抽出して変数に入れます。見つけた項目は、オブジェクト上のプロパティの宣言 (名前と値のペア) である必要があります。たとえば、"answer" ：42 のようになります。

プロパティ

[プロパティ名抽出] アクションは、次の各プロパティを使用して設定できます。

コンバータ
テキストを処理するデータ コンバータのオプション リスト。

変数
抽出した名前を保存する変数を指定します。

スクリーンショット抽出

このステップは、デフォルト (WebKit) ブラウザで使用する必要があります。

このアクションでは、ページ全体またはその一部を画像として抽出し、それを変数に保存します。

タグ ファインダーで、抽出する領域を指定します。タグ パス * を使用して、ページ全体を抽出します。

プロパティ

「スクリーンショット抽出」アクションは、次の各プロパティを使用して設定できます。

パディング
抽出用のパディングは以下のとおりです。正の値で領域が大きくなり、負の値で小さくなります。値セレクターを使用してさまざまな方法で値を指定することができます。

左 **(px)**
左マージン (ピクセル単位)。負の値にすることもできます。

上 **(px)**
上マージン (ピクセル単位)。負の値にすることもできます。

右 **(px)**
右マージン (ピクセル単位)。負の値にすることもできます。

下 **(px)**
下マージン (ピクセル単位)。負の値にすることもできます。

変数
値を割り当てる変数。画像またはバイナリの変数です。

画像形式
画像の形式使用できる値は PNG、JPG、BMP、および GIF です。

画像の読み込み
「ページ読み込み」ステップなどで行っていない場合に、すべての画像をページに読み込みます。

タイムアウト **(ms)**
画像の読み込みに使用するタイムアウト (ミリ秒単位)。

選択済オプション抽出

「選択済オプション抽出」アクションでは、<select> タグから選択されているオプションを抽出し、それを変数に保存します。

オプションのテキストまたはその値を抽出できます。テキストを保存する前に、データ コンバータのリストを使用して処理することができます。また、オプションで先頭および末尾のスペースを削除できます。

プロパティ

[抽出] アクションは、次の各プロパティを使用して設定できます。

値を抽出
選択した場合、選択されたオプションのテキスト自体ではなく、そのオプションの値が抽出されます。

コンバータ
テキストを処理できるデータ コンバータのオプション リスト。

スペースの除去
選択した場合、テキストの先頭および末尾のスペースを除去してから、テキストを変数に保存します。

変数
抽出したテキストを保存する変数を指定します。

シート名抽出

「シート名抽出」アクションでは、スプレッドシート ドキュメントの名前を抽出し、それを変数に保存します。このシートの識別には範囲ファインダーを利用します。

プロパティ

「シート名抽出」アクションは、次のプロパティを使用して設定できます。

変数
シート名を保存する変数。これはテキスト変数である必要があります。

ソース抽出

このアクションでは、プレビューしたデータを変数に保存します。

このステップはプレビュー限定で機能します。

プロパティ

「ソース抽出」アクションは、次の各プロパティを使用して設定できます。

ソース
抽出元のソースのタイプ。バイナリまたはテキストを選択します。ソースがテキストの場合、エンコーディングを明確に定義できます。デフォルトのエンコーディングは、Web サーバーによって提供されたエンコーディングです。

データの抽出先
データの抽出先となる変数。

タグ属性抽出

このアクションでは、検知タグからタグ属性を抽出し、それをデータ コンバータのリストに通し、その結果を変数に保存します。

プロパティ

「タグ属性抽出」アクションは、次の各プロパティを使用して設定できます。

タグ属性の名前
抽出するタグ属性の名前。

コンバータ
属性の値に適用するデータ コンバータ。

変数
結果を保存する変数。

例
この検知タグは次のように考えます。

```
<img
src="mypicture.gif">
```

「タグ属性の名前」が "src" に設定され、「変数」プロパティが "TemporaryData.shortText0" に設定されたと想定します。これによって、値 "mypicture.gif" が "TemporaryData" 変数の "shortText0" 属性に保存されます。

ターゲット抽出

このアクションでは、データを対象 URL から抽出して、変数またはファイルに保存します。選択した変数に実際のデータを保存できない場合 (PDF ファイルを XML 変数に保存しようとするときなど)、アクションを実行するときにエラーが生成されることがあります。

また、オプションで、抽出したデータの実際のコンテンツ タイプとファイルの名前を、ユーザー指定の変数に保存できます。

プロパティ

「ターゲット抽出」アクションは、次の各プロパティを使用して設定できます。

ロケーション
このプロパティで、抽出元になる対象 URL を指定します。

URL
表示されるテキスト フィールドに、URL を直接入力します。HTTP プロトコルを使用する Standard URL が短縮形で記載されることがあるので注意してください。たとえば、"http://www.kofax.com" の代わりに "www.kofax.com" と記載されることがあります。

見つかったタグの **URL**
見つかったタグに URL を含めるように指定します。

変数内の **URL**
指定した変数から URL を読み込むように指定します。

エクスプレッションから **URL** を作成
開く URL としてエクスプレッションを指定します。

コンバータの **URL**
開く URL として出力値を使用するデータ コンバータのリストを指定します。

クリックして読み込んだ **URL**
見つかったノードをクリックし、その結果として読み込まれた URL を使用するように指定します。たとえば、見つかったノードをクリックした結果がフォーム送信となった場合、「ページ読み込み」ステップで読み込まれたデータが、フォーム送信の結果となります。

データを保存
抽出したデータの保存場所を指定します。以下の 2 つから選択できます。

変数
抽出したデータを保存する変数を指定します。この変数は、次のいずれかである必要があります。バイナリ、イメージ、PDF、テキスト、HTML、XML、または Excel です。

ファイル
データを書き込むファイルを指定します。

> ロボットがダイレクト (最小実行) デザイン モードの場合、「ファイルに保存」は、デバッグ モード内で、RoboServer に対してのみ実行されます。
> ロボットがフルで実行 (スマート再実行) デザイン モードの場合、「ファイルに保存」は、デザイン モードとデバッグ モードで実行されます。

ファイル名
ファイルの名前と拡張子を指定します。

自動
このオプションでは、次の方針でファイルの名前を自動的に生成します。

1. まず、レスポンスのコンテンツ配置ヘッダーにファイルの名前のパラメータがあるかどうかを確認し、ある場合はその名前を使用します。
2. 次に、URL にファイルの名前が含まれるかどうかを確認し、ある場合はその名前を使用します。
3. 上のオプションがどれも成功しなかった場合は、エラーが生成されます。

値、変数、エクスプレッション、コンバータ
値セレクターを使用してさまざまな方法で値を指定することができます。

ディレクトリ
ファイルを置くディレクトリを指定します。値セレクターを使用して複数の方法で値を指定することができます。

ディレクトリを作成
存在しないディレクトリをすべて、指定したパスに作成するかどうかを指定します。このオプションを選択すると、ディレクトリが作成されます。このオプションを選択しない場合は、ディレクトリが存在している必要があり、存在しないときにエラーが生成されます。

オーバーライド戦略
選択したファイルがすでに存在している場合の処理を指定します。

ファイルを上書きする
既存のファイルは置き換えられます。

ファイルを上書きしない
既存のファイルは置き換えられなくなります。ファイルがすでに存在する場合は、エラーが生成されます。

新しいファイルを作成
新規ファイルが必ず作成されるようになります。選択した名前のファイルがすでに存在する場合は、新しい固有のファイルの名前が、そのファイルに対して生成されます。この新しいファイルの名前は、選択した元のファイルの名前に、拡張子の直前の末尾にシリアル番号を追加したものになります。たとえば、元のファイルの名前 myData.dat に _1 を追加して myData_1.dat になります。

メタ データを次の場所に保存
抽出したデータについてのメタ データを保存する変数を指定します。

コンテンツ タイプ
データのコンテンツ タイプを保存するオプション変数を指定します。たとえば、画像の場合、コンテンツ タイプは以下のようになります。

image/gif

プレーン テキストの場合、コンテンツ タイプは以下のようになります。

text/plain; charset=iso-8859-1

ファイル名
抽出したデータのファイルの名前を保存するオプション変数を指定します。データをファイルに保存する場合、そのファイルの名前は実際に使用するファイルのフル パスになります。データが変数に読み込まれると、そのファイルの名前は、元のリソースのファイルの名前 (URL またはレスポンスのコンテンツ配置ヘッダーから取得) になります。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

URL 抽出

タグから URL を抽出して、それを変数に保存します。URL が相対の場合は、現在のページの URL を使用して、絶対 URL に変換します。

URL は属性から、たとえば、href 属性付きの <a> タグから抽出できます。また、タグをクリックしたときのように抽出できます。たとえば、onClick イベント ハンドラで新規ページを読み込んだり、フォームを送信したりするボタンの場合などです。タグが <form> タグの場合、アクション属性の値が抽出されるか、またはフォーム送信に対応する URL (および、POST メソッドの使用時にも) が抽出されます。

プロパティ

「URL 抽出」アクションは、次の各プロパティを使用して設定できます。

抽出方法
抽出を行う方法を決めます。

自動
URL の抽出方法を自動的に決めます。

タグ属性から
以下のタグについては、URL をタグの関連属性から直接抽出できます。

- <a>
- <area>
- <form>
- <frame>
- <iframe>
- <script>
- <img>
- <input type="image">
- <param>
- <link>
- <meta>
- タグに Background 属性がある場合の <body>、<table>、<tr>、<td> または <th>

「タグ属性から」の抽出には、さらに 2 つの別のプロパティがあります。

JavaScript URL の実行
タグ属性に JavaScript URL が含まれ、このプロパティのチェックボックスをオンにすると、非 JavaScript URL が読み込まれると見なして JavaScript URL が実行されます。ただし、実際には読み込みを行いません。このプロパティのチェックボックスをオフにした場合、JavaScript URL 自体が抽出されます。

絶対 **URL** に変換
このチェックボックスを選択すると、相対 URL が絶対 URL に変換されます。

読み込まずにクリック

タグをクリックしたときのように URL を抽出できます。ただし、読み込みは行いません。これは、onClick / onMouseDown / onMouseUp のイベント ハンドラ付きのタグや、フォームを送信するボタンを使用する場合に便利です。

読み込まずにフォーム送信

フォームを送信したときのように URL を抽出できます。ただし、実際の要求を Server に送信することはありません。このタイプの抽出は、<form> タグに限り適用できます。[送信] ボタンを使用して送信するには、代わりに「読み込まずにクリック」抽出を選択します。この場合は [送信] ボタンを検知タグとして使用します。

変数

抽出された URL を保存するための変数。

次の時に続行

ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、「待機基準の使用」を参照してください。

オプション

ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

Web ストレージ抽出

「Web ストレージ抽出」ステップでは、現在のブラウザ セッションから、ローカル ストレージやセッション ストレージのデータを抽出します。ローカル ストレージやセッション ストレージは、Web サイトで使用することがあり、それによって Cookie に保存可能な量を超える大量データを維持します。抽出した Web ストレージ データは JSON 形式で変数に保存されます。抽出したデータを口ボットでさらに処理するには、変数を入力として使用するように設定されていれば、ページ生成 ステップを利用できます。JSON はブラウザ表示に読み込まれ、他のステップを、たとえば、抽出した値をループするために追加できます。

プロパティ

「Web ストレージ抽出」アクションは、以下のプロパティを使用して設定できます。

ローカル ストレージを含める

オンにすると、ローカル ストレージのストレージ アイテムが抽出データに含まれるようになります。ブラウザでは、ローカル ストレージは永続 Cookie と同様に、通常、ブラウザ セッションをまたがって存続します。

セッション ストレージを含める

オンにすると、セッション ストレージのストレージ アイテムが抽出データに含まれるようになります。ブラウザでは、セッション ストレージはセッション Cookie と同様に、通常、ブラウザ ウィンドウまたはブラウザ タブが存在する限り、存続します。

キー パターン

特定のキーのある保存アイテムだけを抽出する場合は、対象のキーにマッチするパターンを指定できます。このフィールドが空欄の場合、キーに関係なくすべてのストレージ アイテムが含まれるようになります。パターンを指定すると、キー全体とマッチする必要があるので注意してください。

ドメイン パターン

特定のドメインに属する保存アイテムだけを抽出する場合は、対象のドメインにマッチするパターンを指定できます。このフィールドが空欄の場合、すべてのドメインのストレージ アイテムが含まれるようになります。パターンを指定すると、ドメイン全体とマッチする必要があるので注意してください。

出力値

抽出したストレージを保存する変数を指定します。ストレージは JSON 形式で抽出されます。

データベース データ抽出

過去にデータベース データ登録したコンプレックス タイプの値を検知します。値を検知しない場合はエラーが生成されます。値を検知すると、その値が読み込まれます。データベース接続の設定は「設定」で行います。

プロパティ

「データベース データ抽出」アクションは、次の各プロパティを使用して設定できます。

データベース

値を検知するデータベース。値は、設計時に選択またはハード コーディングできます。あるいは、データベースの名前は、実行時に変数、エクスプレッション、またはコンバータを使用して動的に構築できます。ロボットの実行時にこの名前のデータベースが存在しない場合は、エラーが発生します。

変数

検知した値の読み込み先となる変数を選択します。値を保存した後で、保存可能な属性がそのタイプに追加された場合は、「データベース データ抽出」を使用してその値を読み込むことはできません。この変数は標準のコンプレックス タイプである必要があります。7.2 より前に存在した特殊なデータベース出力タイプではありません。

キー

検知する値に対する固有キー。このキーは、値を保存したときに使用したものである必要があります。このキーは「データベース キーの一部」としてタイプで定義された可能性があります。または、変数、エクスプレッション、またはコンバータを使用しても定義できます。所定のキーを持つ値が存在する場合、その値をデータベースから読み込んで変数に入れます。所定のキーで保存された値がない場合、エラーが生成されます。このエラーは他のエラーと同様に処理できます。

空の属性だけを上書き

このオプションが有効な場合、空の属性だけを、データベースから読み込んだ値で上書きします。これによって、「データベース データ抽出」の使用前にデータを抽出できるようになり、抽出した属性の値を上書きしないようにできます。

ブラウザ ウィンドウ繰り返し

このアクションでは、各オープン ブラウザ ウィンドウ (フレームやポップアップ ウィンドウを含む) を、現在のウィンドウ、つまり、以降の各ステップで処理するウィンドウとして次々と選択しながらループします。

データ行繰り返し

このアクションは、CSV ファイル内の各データ行をループします。このステップを利用するには、ファイル読み込みステップを使用して、CSV ファイルを読み込みます。プレビュー ウィンドウで、[アクションを選択] をクリックして、[開く] を選択します。このアクションでは自動的に「CSV 形式表示」ステップを追加します。これで、「データ行繰り返し」ステップおよびデータ行の列を抽出ステップを利用できます。

プロパティ

範囲名
アクションがループするセル範囲の名前を指定します。

- 自動：Design Studio が自動的に名前を割り当てます。
- 名前付き：範囲に名前を付けます。

このステップの結果は、データ行の列を抽出ステップでのデータ抽出に利用できる、(1 行の) 各セルの名前付き範囲です。

ファイル繰り返し

このアクションでは、ディレクトリ内の各ファイルをループします。

このアクションは、指定するディレクトリ内の各ファイルをループしますが、オプションで、サブディレクトリに含まれる各ファイルもループします (直接または間接的に)。各イテレーションで、ファイルパスを含む現在のファイルのファイル名が、選択した変数に保存されます。

ファイルの名前のパターンを指定すると、このパターンにマッチするファイル名のファイルだけが含まれるようになります。パターンは、パスのないファイルの名前全体に対してマッチします。

選択した要素にループするサブ要素が含まれていない場合、すべての [要素の繰り返し]ステップがエラーをスローします。ただし、[タグ繰り返し] ステップを使用してディレクトリ内のファイルをループする際にディレクトリにファイルが存在しない場合、エラーは見なされず、エラーはスローされません。

プロパティ

「ファイル繰り返し」アクションは、次の各プロパティを使用して設定できます。

ディレクトリの名前
ループするディレクトリの名前。名前は、値セレクターを使用してさまざまな方法で指定できます。名前は絶対ディレクトリ名である必要があります。該当する場合はドライブ名とディレクトリへのパスを含めます。

サブディレクトリを含める
このオプションがオンになっている場合は、ディレクトリのサブディレクトリに (直接または間接的に) あるファイルを含めます。オフになっている場合は、ディレクトリ直下にあるファイルだけを含めます。

ファイルの名前のパターン
ここでパターンを指定すると、このパターンにマッチするファイル名のファイルだけが含まれるようになります。パターンは、パスのないファイルの名前全体に対してマッチします。

ファイルの名前をここに保存
各イテレーションでの現在のファイルの名前を保存する変数。保存されるのはファイルの名前全体です。該当する場合は、ドライブ名とディレクトリ パスを含めます。

JSON アイテム繰り返し

「JSON アイテム繰り返し」アクションは、JSON 配列の全項目をループします。各イテレーションで、適切な項目が名前付き JSON としてマークされます。

「JSON アイテム繰り返し」アクションは、グローバル変数には無効です。

プロパティ

「JSON アイテム繰り返し」アクションは、次の各プロパティを使用して設定できます。

名前
2 つのオプションがあります。「自動」または「名前付き」。
自動: 項目名に番号が割り振られます。自動で割り振られた最初の項目名は 1 になり、次に 2、3 と続きます。
名前付き: 固定的な、明確に記述された名前を項目に与えます。
詳細については、名前付きタグ、範囲、JSON を参照してください。

既存の名前付きアイテムを保持
オンにすると、既存の名前付き項目が保持されます。オフの場合、これらは名前付き項目としてマークされずに、このステップの後で見つかった項目だけが名前付き項目となります。

セレクト オプション繰り返し

このアクションは、ドロップダウン ボックスまたはリスト ボックス内のオプションをループし、各イテレーションで 1 つのオプションを選択します。

検出されたタグは <select> タグである必要があります。

<select> タグに対して登録されたイベント ハンドラーがある場合、オプションを選択すると、JavaScript の実行がトリガーされる可能性があることに注意してください。

ループしないでオプションを選択するには、オプション選択または複数オプション選択アクションを使用します。

プロパティ

「セレクトオプション繰り返し」アクションは、次のプロパティを使用して設定できます。

オプションをスキップ
いずれかのオプションをループ内でスキップする必要がある場合は、ここで指定する必要があります。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

プロパティ繰り返し

「プロパティ繰り返し」アクションは、JSON オブジェクトのすべてのプロパティ (名前と値のペア) をループします。各イテレーションで、適切なプロパティが名前付き JSON としてマークされます。

「プロパティ繰り返し」アクションは、グローバル変数では機能しません。

プロパティ

「プロパティ繰り返し」アクションは、次のプロパティを使用して 設定できます。

名前
2 つのオプションがあります。「自動」または「名前付き」。
自動: 項目名に番号が割り振られます。自動で割り振られた最初の項目名は 1 になり、次に 2、3 と続きます。
名前付き: 固定的な、明確に記述された名前を項目に与えます。
詳細については、名前付きタグ、範囲、JSON を参照してください。

既存の名前付きアイテムを保持
オンにすると、既存の名前付き項目が保持されます。オフの場合、これらは名前付き項目としてマークされずに、このステップの後で見つかった項目だけが名前付き項目となります。

ラジオ ボタン繰り返し

このアクションは、各イテレーションでラジオ ボタンの 1 つを選択して、ラジオ ボタンのグループをループします。

検出されるタグはグループ内のラジオ ボタンの 1 つである必要があります。すべてのラジオ ボタンを含むタグではありません。

ボタンに対して登録されたイベント ハンドラーがある場合、ラジオ ボタンを選択すると、JavaScript の実行がトリガーされる可能性があることに注意してください。

ループしないでラジオ ボタンを選択するには、ラジオボタン選択アクションを使用します。

プロパティ

「ラジオボタン繰り返し」アクションは、次のプロパティを使用して設定できます。

これらのラジオ ボタンをスキップ
ループ中にスキップするラジオ ボタンを選択します。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

タグ繰り返し

「タグ繰り返し」アクションは、タグのグループをループします。各イテレーションで、適切なタグが名前付きタグとしてマークされます。

多くの場合、ループは表のすべての <tr> タグをループするなど、最初から最後までループする必要があります。ただし、シーケンス内の最後の n 個のタグなど、一部のタグだけをループするように「タグ繰り返し」アクションを設定することもできます。

「タグ繰り返し」アクションは、タグ パス繰り返しアクションに似ています。大きな違いは、「タグ繰り返し」アクションでは検出されたタグの直下のタグのみが検出され、「タグ パス繰り返し」アクションではサブツリー全体が検索されることが挙げられます。

プロパティ

「タグ繰り返し」アクションは、次のプロパティを使用して設定できます。

タグ
ループするタグの名前 (例：**tr**)。

インクルード **class**
結果に含めるノードの class を指定します。論理積 (AND) はスペースで、論理和 (OR) は | で表されます。論理積は論理和に優先します。たとえば、class1 class2 は、class1 および class2 であるノードを示し、class1 | class2 | class3 は、class1、class2、または class3 のいずれかのノードを示しますが、class1 class2 | class3 class4 は、class1 および class2、または class3 および class4 であるノードを示します。

除外する **class**
結果から除外するノードの class を指定します。論理積 (AND) はスペースで表され、論理和 (OR) は | で表されます。また、class が存在しないことを明示的に示す場合は $ が使用されます。論理積は論理和に優先します。たとえば、class1 class2 は、class1 および class2 であるノードを示し、class1 | $ は、class1 か class のないノードのいずれかを示します。class1 | class2 | class3 は、class1、class2、または class3 のいずれかのノードを示しますが、class1 class2 | class3 class4 は、class1 および class2、または class3 および class4 であるノードを示します。

最初のタグ番号
ループに含める最初のタグの番号。番号を最初のタグから順方向にカウントするか、最後のタグから逆方向にカウントするかを指定することができます。

最後のタグ番号

ループに含める最後のタグの番号。番号を最初のタグから順方向にカウントするか、最後のタグから逆方向にカウントするかを指定することができます。

タグ番号をインクリメント

ループ スキップ タグを作成します。たとえば、2 のインクリメントを指定すると、ループは 2 番目のタグをすべてスキップします。

逆方向にループ

一致するタグを逆順にループさせることを選択します。同じタグを順方向に逆順でループするようにループが実行されることに注意してください。つまり、「最初のタグ番号」とは、ループするときに最初にアクセスするタグではなく、ループするタグの選択の最初のタグということになります (実際には一番最後のタグです)。

前にあるインクルード タグ

各出力値に含める名前付きタグの前にある (同じ名前の) タグの数。

後ろにあるインクルード タグ

各出力値に含める名前付きタグの後ろにある (同じ名前の) タグの数。

タグの名前

「自動」と「名前付き」という 2 つのオプションがあります。「自動」はタグ名として番号を与えます。「自動」で最初に与えられる番号は 1、次に与えられる番号は 2 で、以下同様です。このステップの前に「自動」により追加的に番号を与えられたタグが (同じページに) 挿入されると、番号は変わることがあります。「名前付け」はタグに明確に指定された固定の名前を与えますが、これにはいくつかの利点があります。

- 適切に選んだ名前を付けると、そのタグがどのようなタグであるかを容易に思い出せます。
- 明確に名前付きタグは、名前付きの別のタグがその前に挿入されても影響を受けません。
- 名前付きタグ設定 においてすでに使われているのと同じ名前を使用する場合、その名前は単純に新しいタグを参照することになります (ステートフルなページ内ループに便利です)。

既存の名前付きタグを保持

このオプションを選択すると、既存の名前付きタグが、各イテレーションの結果をマークする名前付きタグとともに保持されます。このオプションを選択していない場合は、既存の名前付きタグが除去され、各出力状態にはイテレーションの結果を示す名前付きタグのみが含まれます。

例

この検知タグは次のように考えます。

```
<tbody>
  <tr>...
  <tr>...
  <tr>...
  <tr>...
  <tr>...
</tbody>
```

タグ名が "tr"、最初のタグ番号が 0 (最初から)、最後のタグ番号が 1 (最後から) に設定されている場合、「タグ繰り返し」アクションにより <tr> タグ、0、1、2、3 をループします。各イテレーションにおいて、アクションにより設定された名前付きタグは、以下に示す適切な <tr> タグになります。

イテレーション	名前付きタグ
1	<tr> タグ 0
2	<tr> タグ 1
3	<tr> タグ 2
4	<tr> タグ 3

タグ パス繰り返し

「タグ パス繰り返し」アクションにより、検出されたタグのサブツリー内にある特定のタイプのすべてのタグをループします。各イテレーションで、適切なタグが名前付きタグとしてマークされます。

「タグ パス繰り返し」アクションは、タグ繰り返しアクションによく似ています。大きな違いは、「タグ繰り返し」アクションでは検出されたタグの直下のタグのみが検出され、「タグ パス繰り返し」アクションではサブツリー全体が検索されるということが挙げられます。

プロパティ

「タグ パス繰り返し」アクションは、次のプロパティを使用して設定できます。

タグ パス

ループするタグのパスを指定します。タグ パスはタグ ファインダーの場合と同様に指定します。サブツリー全体について一致するタグがすべて検出されます。

インクルード **class**

結果に含めるタグの class を指定します。論理積 (AND) はスペースで、論理和 (OR) は | で表されます。論理積は論理和に優先します。たとえば、class1 class2 は、class1 および class2 であるタグを示し、class1 | class2 | class3 は、class1、class2、または class3 のいずれかのタグを示しますが、class1 class2 | class3 class4 は、class1 および class2、または class3 および class4 であるタグを示します。

除外する **class**

結果から除外するタグの class を指定します。論理積 (AND) はスペースで表され、論理和 (OR) は | で表されます。また、class が存在しないことを明示的に示す場合は $ が使用されます。論理積は論理和に優先します。たとえば、class1 class2 は、class1 および class2 であるタグを示し、class1 | $ は、class1 か class のないタグのいずれかを示します。class1 | class2 | class3 は、class1、class2、または class3 のいずれかのタグを示しますが、class1 class2 | class3 class4 は、class1 および class2、または class3 および class4 であるタグを示します。

最初のタグ番号

ループに含める最初の一致するタグの番号。番号を最初のタグから順方向にカウントするか、最後のタグから逆方向にカウントするかを指定することができます。

最後のタグ番号

ループに含める最後の一致するタグの番号。番号を最初のタグから順方向にカウントするか、最後のタグから逆方向にカウントするかを指定することができます。

タグ番号をインクリメント

ループ スキップ タグを作成します。たとえば、2 のインクリメントを指定すると、ループは 2 番目のタグをすべてスキップします。

逆方向にループ

一致するタグを逆順にループさせることを選択します。同じタグを順方向に逆順でループするようにループが実行されることに注意してください。つまり、「最初のタグ番号」とは、ループするときに最初にアクセスするタグではなく、ループするタグの選択の最初のタグということになります (実際には一番最後のタグです)。

タグの名前

「自動」と「名前付き」という 2 つのオプションがあります。「自動」はタグ名として番号を与えます。「自動」で最初に与えられる番号は 1、次に与えられる番号は 2 で、以下同様です。このステップの前に「自動」により追加的に番号を与えられたタグが (同じページに) 挿入されると、番号は変わることがあります。「名前付け」はタグに明確に指定された固定の名前を与えますが、これにはいくつかの利点があります。

- 適切に選んだ名前を付けると、そのタグがどのようなタグであるかを容易に思い出せます。
- 明確に名前付きタグは、名前付きの別のタグがその前に挿入されても影響を受けません。
- 名前付きタグ設定 においてすでに使われているのと同じ名前を使用する場合、その名前は単純に新しいタグを参照することになります (ステートフルなページ内ループに便利です)。

既存の名前付きタグを保持

このオプションを選択すると、既存の名前付きタグが、各イテレーションの結果をマークする名前付きタグとともに保持されます。このオプションを選択していない場合は、既存の名前付きタグが除去され、各出力状態にはイテレーションの結果を示す名前付きタグのみが含まれます。

例

この検知タグは次のように考えます。

```
<table>
  <tbody>
    <tr>
      <td> 1 </td>
      <td> 2 </td>
    </tr>
  </tbody>
</table>
```

「タグパス」を td に設定します。開始イテレーションでは名前付きタグが <td> 1 </td> に設定され、次のイテレーションでは、名前付きタグは <td> 2 </td> になります。

この検出されたタグを次のように考察します。

```
<table>
  <tbody>
    <tr>
      <td>
        <table>
          <tbody>
            <tr>
              <td> 1 </td>
              <td> 2 </td>
            </tr>
          </tbody>
        </table>
      </td>
      <td> 3 </td>
    </tr>
  </tbody>
</table>
```

「タグパス」を tr.td に設定します。開始イテレーションでは、名前付きタグを次のように設定します。

```
<td><table><tbody><tr><td> 1
</td><td> 2
</td></tr></tbody></table></td>
```

次のイテレーションでは、名前付きタグが以下となります。

```
<td> 3 </td>
```

部分文字列繰り返し

このアクションは、テキストを指定された区切り記号のパターンで分割し、その部分をループして、各イテレーションで選択された変数に次のテキスト部分を割り当てます。

プロパティ

「部分文字列繰り返し」ステップは、次のプロパティを使用して設定できます。

入力

分割する文字列は、値セレクターを使用してさまざまな方法で指定できます。タグの内容を分割する必要がある場合は、まず抽出のステップを使用してタグ テキストを変数に抽出する必要があります。

区切り記号

テキストを分割する区切り記号を指定します。

下にある "," を区切り記号として使用してテキストを分割する例を参照してください。

出力値

各イテレーションでテキスト部分が格納される変数を指定します。

空の出力値をスキップ

選択すると、ループは長さゼロのテキストが出力されたイテレーションをスキップします。たとえば、テキスト "a,b,,c" をループする場合、このプロパティがチェックされていると、ループに含まれるイテレーションは 3 つだけになります ("a"、"b"、"c") が出力される)。「空の出力をスキップ」プロパティが選択されていない場合、ループには 4 つのイテレーションが含まれます ("a"、"b"、""、"c" が出力される)。

例

次の入力テキストがあるものとします。

```
apple,pear,banana,grape,kiwi,pineapple
```

フルーツをイテレートし、各イテレーションで何らかのアクション、たとえばフルーツの名前をデータベースに格納するとします。

区切り記号として、,

を指定し、出力変数として Fruit.name を選択します。

開始イテレーションでは、Fruit.name 変数に apple という値が格納され、2 つ目のイテレーションでは pear という値が格納されます。ループ全体には 6 つのイテレーションが含まれ、最後のイテレーションでは Fruit.name 変数に pineapple という値が含まれます。

URL 繰り返し

「URL 繰り返し」アクションは、検出されたタグに含まれる URL をループし、オプションで重複する URL をスキップします。URL を含むタグは、検出されたタグ内の範囲内の深度に配置できます。各イテレーションで、適切なタグが名前付きタグとしてマークされます。

プロパティ

「URL 繰り返し」アクションは、次のプロパティを使用して設定できます。

URL タグ
ループする URL を持つ HTML タグを指定します。

最初の **URL** 番号
ループに含める最初の URL の番号。番号を最初の URL から順方向にカウントするか、最後の URL から後ろ逆方向にカウントするかを指定することができます。

最後の **URL** 番号
ループに含める最後の URL の番号。番号を最初の URL から順方向にカウントするか、最後の URL から後ろ逆方向にカウントするかを指定することができます。

逆方向にループ
一致するタグを逆順にループさせることを選択します。同じタグを順方向に逆順でループするようにループが実行されることに注意してください。つまり、「最初のタグ番号」とは、ループするときに最初にアクセスするタグではなく、ループするタグの選択の最初のタグということになります (実際には一番最後のタグです)。

URL パターン
各 URL と照合するパターン。このアクション プロパティは、URL をスキップするかどうかを決定します。パターンは URL 全体と照合する必要があります。パターンが指定されていない場合、パターンの照合は行われません。

アクション
「パターンに一致する URL をスキップする」に設定すると、パターンに一致するすべての URL がスキップされます。「パターンに一致しない URL をスキップする」に設定すると、パターンに一致しないすべての URL がスキップされます。

重複 **URL** をスキップ
重複 URL (複数の同一の URL) をスキップしてループさせないかどうかを指定します。同じ URL を複数回ループさせないようにするには、このプロパティを選択します。(これは通常の場合です。)

タグの名前
「自動」と「名前付き」という 2 つのオプションがあります。「自動」はタグ名として番号を与えます。「自動」で最初に与えられる番号は 1、次に与えられる番号は 2 で、以下同様です。このステップの前に「自動」により追加的に番号を与えられたタグが (同じページに) 挿入されると、番号は変わることがあります。「名前付け」はタグに明確に指定された固定の名前を与えますが、これにはいくつかの利点があります。

- 適切に選んだ名前を付けると、そのタグがどのようなタグであるかを容易に思い出せます。
- 明確に名前付きタグは、名前付きの別のタグがその前に挿入されても影響を受けません。
- 名前付きタグ設定 においてすでに使われているのと同じ名前を使用する場合、その名前は単純に新しいタグを参照することになります (ステートフルなページ内ループに便利です)。

既存の名前付きタグを保持
このオプションが選択されると、既存の名前付きタグは、各イテレーションの結果をマークしている名前付きタグとともに保持されます。このオプションを選択していない場合は、既存の名前付きタグが除去され、各出力状態にはイテレーションの結果を示す名前付きタグのみが含まれます。

エラー生成

このアクションはエラーを生成します。

Web サイトで特定のアクションを実行しようとしたときにそのサイトからエラー ページが送信された場合など、エラーと見なされる状況が検出された場合に役立ちます。「エラー生成」アクションを使用すると、ロボットの実行中に発生するその他のエラーのように処理されるロボット エラーを生成できます。

特定の状況が発生したことを記録し、現在の分岐に沿って実行を継続することがまさに重要である場合は、代わりにログ出力の使用を検討してください。

プロパティ

「エラー生成」アクションは、次のプロパティを使用して設定できます。

エラー メッセージ
エラーに含めるエラー メッセージ。エラー メッセージは、値セレクターを使用してさまざまな方法で指定できます。エラー メッセージが指定されていない場合は、デフォルトのエラー メッセージが使用されます。

ファイル情報取得

このアクションにより、ファイル システム内のファイルに関するメタデータがフェッチされます。

プロパティ

「ファイル情報取得」アクションは次のプロパティを使用して設定できます。

ファイル名
探索するファイルの名前。名前は、値セレクターを使用してさまざまな方法で指定できます。名前は絶対パス名である必要があります。該当する場合はドライブ名とファイルへのディレクトリ パスを含めます。

最終変更日時の格納先
最後に変更を行った日付を格納する変数を指定します。

格納サイズ
ファイルのサイズを格納する変数をバイト単位で指定します。これはディスク上でのファイル サイズではなく、ファイル内容の実際のサイズです。

イテレーション取得

このアクションは、エンクローズするループ ステップの現在のイテレーションを取得し、変数に格納します。

このアクションは、オブジェクトのリストをループするときに、抽出している現在のオブジェクトの番号が重要である場合に有用です。

プロパティ

「イテレーション取得」アクションは次のプロパティを使用して設定できます。

繰り返しステップ
イテレーションが取得されるループ ステップの番号。ロボット内の最初のループ ステップから順方向にカウントするか、または直ちにエンクローズするループ ステップから逆方向にカウントします。たとえば、"2" が入力され、"From Last" が選択される場合、内側から 2 番目にあるエンクローズするループ ステップのイテレーションが選択されます。

イテレーションをここに格納
イテレーションを格納する変数。

グループ

グループ ステップは別のステップを含んだ状態で設計を行いますが、グループ ステップを 1 つのステップとしてまとめることで、別のステップを非表示にすることができます。これはロボットを構造化する場合に役立ちます。グループ ステップ内のステップをグループ化することは、ロボットの実行に影響しません。

グループ ステップの左上隅には、広げる/閉じるアイコン (+/-) があります。このアイコンをクリックすると、グループ ステップを展開または折りたたむことができます。グループ ステップを折りたたむと、通常のステップと同様の見た目となり、内部のすべてのステップが非表示になります。グループ ステップを展開すると、グループ化されていない場合に表示されるレイアウトと同様に、中に含まれるステップが表示されます。グループ ステップは、ツールバーの [すべて展開]/[すべて折りたたむ] ボタンを使用して展開したり折りたたんだりすることもできますが、このアクションはロボット内のすべてのグループ ステップに対して適用されます。また、[グループを展開]/[グループを折りたたむ] によるアクションは、選択内のすべてのグループに対して適用されます。

ステップをグループ化するには、グループ化するステップを選択し、ステップのツールバー、[編集] メニュー、またはコンテキスト メニューの [グループ] アクションを使用します。サブグラフを形成するステップのみをグループ化することができます。つまり、すべてのステップを選択内の別のステップに直接接続する必要があります。(選択外のステップから) 選択内に入れる接続は 1 つだけです。選択を選択外のステップにつなげる接続は、すべてグループの最後に接続されます。

ステップのグループ化を解除するには、グループ化を解除するグループ ステップを選択し、ステップのツールバー、[編集] メニュー、またはコンテキスト メニューの [グループ解除] アクションを使用します。

(グループが展開されている状態では) グループ ステップの最初と最後に三角形のマーカーが表示されます。これはグループ ステップの開始と終了を示し、これらのマーカーの 1 つを右クリックすると表示されるコンテキスト メニューを使用することで、グループの先頭に分岐を挿入するなどのグループ ステップで選択したアクションを実行できます。

タグ非表示化

このアクションは検出されたタグを非表示にします。

非表示化は、タグのスタイル属性を設定することで実行されます。つまり、タグはページ内に保持されますが、スタイルの使用によりページ上では見えないように設定されます。

このアクションは、ページからの切り抜きをするときタグが見えない方がよい場合には特に便利です。タグを除去する代わりに非表示にすることで、特定の構造と内容を持つページに依存することのある JavaScript などの破壊が避けられます。

プロパティ

「タグ非表示化」アクションは、次のプロパティを使用して設定できます。

レイアウトを適切に調整する
このオプションが選択されている場合に非表示化が実行されると、タグが占有していたスペースが使用されるようにページのレイアウトが調整されます。このオプションが選択されていない場合に非表示化が実行されると、タグはそれ以前と同じスペースを占有し続けますが、ページ内には表示されません。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

列挿入

このアクションは、スプレッドシートに 1 つ以上の列を挿入します。

プロパティ

「列挿入」アクションは、次のプロパティを使用して設定できます。

挿入位置
このオプションは、列を挿入する場所を指定します。次のオプションがあります。

- 最初
- 前 (デフォルト)
- 後ろ
- 最後

セル範囲が選択されている場合、[前] オプションは範囲の前に列を挿入し、[後] オプションは範囲の後に列を挿入します。[最初] オプションは範囲内の最初に列を挿入し、[最後] オプションは範囲内の最後に列を挿入します。

列数
挿入する列数

名前付き範囲として設定
範囲名のオプションは以下のとおりです。

- 自動 (デフォルト)
- 名前付き：このオプションを選択する場合は、範囲の名前を指定します。

範囲名
「自動」、「名前付き」という 2 つのオプションがあります。「自動」は範囲名として番号を与えます。「自動」で最初に与えられる番号は 1、次に与えられる番号は 2 となります (以下同様)。「自動」で番号が与えられる他の範囲が (同じページの) このステップの前に挿入されると、番号が変わることがあります。名前付けにより、範囲に対して明示的に指定された固定の名前が付けられますが、これには次のような利点があります。

- 適切に選んだ名前を付けると、その範囲がどのような範囲であるかを容易に思い出せます。
- 明確に名前付けされた範囲は、名前付きの別の範囲がその前に挿入されても影響を受けません。
- すでに使われているのと同じ名前を使用する場合、その名前は単純に新しい範囲を参照することになります (ステートフルなページ内ループに便利です)。

コンテンツ挿入

このアクションは、検出されたタグに対応するドキュメントに、指定されたコンテンツを挿入します。

このステップは XML 変数に対してのみ機能します。このステップは、XML の正当性の確認も工ンティティの解決もしません。

プロパティ

「コンテンツ挿入」アクションは、次のプロパティを使用して設定できます。

新しいコンテンツ
挿入する新しいコンテンツです。

タグの挿入場所
検出されたタグに対応する新しいタグを挿入する場所を選択します。次のオプションがあります。

- 検出されたタグの最初の子工レメントとして
- 検出されたタグの最後の子工レメントとして
- 検出されたタグの前
- 検出されたタグの後

タグをテキスト ノードに挿入することはできません。代わりに、囲んでいるタグを選択する必要があります。

名前付きタグとして設定
項目に名前付きタグ設定するとき、このプロパティを選択します。

自動
項目の名前として番号を与えます。自動で項目に与えられる最初の番号は 1、次に与えられる番号は 2 であり、以下同様です。

名前付き
固定的な、明確に記述された名前を項目に与えます。

詳細については、名前付きタグ、範囲、JSON を参照してください。

JSON 挿入

このアクションは、JSON オブジェクトに新しいプロパティ (名前と値のペア) を挿入し、または JSON 配列に新しい項目を挿入します。

このステップは JSON 変数に対してのみ機能します。

プロパティ

「JSON 挿入」アクションは、次のプロパティを使用して設定できます。

挿入対象

このオプションは、挿入する新しいプロパティを定義します。このオプションを使用する場合、検出された JSON 値はオブジェクトである必要があり、それ以外はエラーが発生します。新しいプロパティは、以下の 2 つのプロパティで定義します。

オブジェクト プロパティ

このオプションは、挿入する新しいプロパティを定義します。このオプションを使用した場合、検出された JSON 値はオブジェクトである必要があり、それ以外はエラーが発生します。新しいプロパティは、以下の 2 つのプロパティで定義します。

名前

新しいプロパティの名前です。これは有効なプロパティ名である必要があり、引用符などはエスケープする必要があります。

値

これはプロパティの値ですが、有効な JSON 値である必要があります。たとえば、テキストを挿入するときは引用符で囲む必要があります。

配列アイテム

このオプションは、挿入する新しい項目を定義します。このオプションを使用する場合、検出された JSON 値は配列である必要があり、それ以外はエラーが発生します。新しいプロパティは、次のプロパティによって定義します。

値

これは項目の値ですが、有効な JSON 値である必要があります。たとえば、テキストを挿入するときは引用符で囲む必要があります。

挿入位置

検出された JSON 値に対応する新しい JSON を挿入する場所を選択します。以下のオプションがあります。

前

プロパティまたは項目を検出された JSON の前に挿入します。

最初

プロパティまたは項目を JSON オブジェクトまたは配列における最初のプロパティとして挿入します。

最後

プロパティまたは項目を JSON オブジェクトまたは配列における最後のプロパティとして挿入します。

後ろ

プロパティまたは項目を検出された JSON の後ろに挿入します。

名前付き **JSON** として設定
項目に名前付き JSON を設定するとき、このプロパティを選択します。

自動
項目名に番号が割り振られます。自動で割り振られた最初の項目名は 1 になり、次に 2、3 と続きます。

名前付き
固定的な、明確に記述された名前を項目に与えます。

詳細については、名前付きタグ、範囲、JSON を参照してください。

行挿入

このアクションは、スプレッドシートに 1 つまたは複数の行を挿入します。

プロパティ

「行挿入」アクションは、次のプロパティを使用して設定できます。

挿入位置
このオプションは、行を挿入する場所を指定します。次のオプションがあります。

- 最初
- 前 (デフォルト)
- 後ろ
- 最後

セル範囲が選択されている場合、[前] オプションは範囲の前に行を挿入し、[後] オプションは範囲の後に行を挿入します。[最初] オプションは範囲内の最初に行を挿入し、[最後] オプションは範囲内の最後に行を挿入します。

行数
挿入する行数

名前付き範囲として設定
範囲名のオプションは以下のとおりです。

- 自動 (デフォルト)
- 名前付き：このオプションを選択する場合は、範囲の名前を指定します。

範囲名
2 つのオプションがあります。「自動」または「名前付き」。
自動: 項目名に番号が割り振られます。自動で割り振られた最初の項目名は 1 になり、次に 2、3 と続きます。
名前付き: 固定的な、明確に記述された名前を項目に与えます。
詳細については、名前付きタグ、範囲、JSON を参照してください。

シート挿入

このアクションは、スプレッドシートに新しいシートを挿入します。

プロパティ

「シート挿入」アクションは次のプロパティを使用して設定できます。

挿入位置
このオプションにより、シートを挿入する場所を指定します。次のオプションがあります。

- 最初
- 前 (デフォルト)
- 後ろ
- 最後

名前
このオプションは、挿入されたシートの名前を指定します。

タグ挿入

「タグ挿入」アクションは、新しいタグを挿入します。JavaScript の実行がオプションで無効になっていない場合、新しいタグの HTML 内にある JavaScript が実行されます。このステップの使用は、最小実行モードでのみサポートされています。

プロパティ

「タグ挿入」アクションは、次のプロパティを使用して設定できます。

新しいタグの **HTML**
新しいタグの HTML を指定します。HTML の指定は、以下で説明する複数の方法で行うことができます。

タグの挿入場所
検出されたタグに対応する新しいタグを挿入する場所を選択します。次のオプションがあります。

- 検出されたタグの最初の子エレメントとして
- 検出されたタグの最後の子エレメントとして
- 検出されたタグの前
- 検出されたタグの後

以下のルールに注意してください。

- <html>、<head>、または <body> タグの前に新しいタグを挿入すると、そのタグは、ドキュメントに <body> タグが含まれている場合には <body> の最初のタグになります。それ以外の場合、そのタグは <html> の最初のタグになります。
- <head> タグの後ろに新しいタグを挿入すると、そのタグは <body> の最初のタグになります。
- <html> または <body> タグの後ろに新しいタグを挿入すると、そのタグは <body> の最後のタグになります。
- HTML タグをテキスト ノードに挿入することはできません。代わりに、囲んでいるタグを選択する必要があります。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレー

ムの エレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション

ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

HTML の指定

新しいタグの HTML は以下の方法で指定できます。

HTML

単純に新しいタグの HTML を記述します。

エクスプレッションからの **HTML**

結果が新しいタグの HTML として使用されるエクスプレッションを記述します。

エクスプレッションとパターンから **HTML** を作成

検出されたタグと照合されるパターンと、生成される結果が新しいタグの HTML として使用されるエクスプレッションを記述します。検出されたタグのパーツを使用して新しいタグの HTML を作成する場合は、この方法で HTML を指定します。

パターン

「タグ挿入」アクションのために検出されたタグと照合されるパターン。パターンは検出されたタグの全体と一致している必要があり、一致しない場合はエラーが発生します。

照合対象

検出されたタグのどの部分とパターンを照合するかを指定します。

- 「テキストのみ」は、パターンが検出されたタグ中のテキストとのみ照合します。
- "HTML" は、パターンが検出されたタグの HTML と照合します。

大文字と小文字を無視

このチェックボックスをオンにすると、パターンが入力と照合されるときに大文字と小文字の区別は無視されます。たとえば、"KoFaX" は "kOfax" と同じであると見なされます。

エクスプレッション

このフィールドには、生成される結果が新しいタグの作成に使用されるエクスプレッションが入ります。エクスプレッションは、$n 表記を使用して、「パターン」フィールド内のパターンのサブマッチを参照することができます。たとえば、エクスプレッションに $1 を挿入すると、パターン内の最初のサブマッチ (つまり、パターン内の最初の丸括弧のペアに囲まれたコンテンツと一致するテキスト) を得ることができます。

XML 変数から **HTML** を変換

内容が HTML に変換されて新しいタグの HTML として使用される XML 変数を選択します。

ファイル読み込み

このアクションは、ファイルをブラウザまたは変数に読み込みます。

プロパティ

「ファイル読み込み」アクションは、次のプロパティを使用して設定できます。

ファイル名
このプロパティは、読み込むファイルへのパスを指定します。パスはエクスプレッションまたはコンバータを使用して設定できます。

出力値
ファイルの内容をどうするか、つまり、ブラウザに読み込むかまたは変数に保存するかを指定します。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。詳細については、「待機基準の使用」を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

ページ読込

このアクションは、現在のページにあるリンク、または他の場所から得られた URL からページを読み込みます。通常は、現在のページにあるリンクから読み込む方が簡単です。それには、クリック アクションを使用してリンクをクリックします。

プロパティ

「ページ読み込み」アクションは、次のプロパティを使用して設定できます。

ロケーション
このプロパティで、開く URL を指定します。URL は、URL セレクターを使用して複数の方法で指定できます。

URL
表示されるテキスト フィールドに、URL を直接入力します。HTTP プロトコルを使用する標準の URL は省略できます。たとえば、http://www.kofax.com の代わりに、www.kofax.com と入力できます。

見つかったタグの **URL**
見つかったタグに URL を含めるように指定します。見つかったタグは、以下のタイプのいずれかである必要があります。

- <a href="URL">
- <area href="URL">
- <frame src="URL">
- <iframe src="URL">
- <script src="URL">
- <param value="URL">
- <meta http-equiv="Refresh" content="...; url=URL">

変数内の **URL**
指定した変数から URL を読み込むように指定します。

エクスプレッションから **URL** を作成
開く URL としてエクスプレッションを指定します。

コンバータの **URL**
開く URL として出力値を使用するデータ コンバータのリストを指定します。

読み込み先
このプロパティで、ページの読み込み先ウィンドウを指定します。既存ウィンドウ、新規ウィンドウのどちらも指定できます。以下のオプションがあります。

- 「自動」は、ブラウザの読み込み先と同じウィンドウにページを読み込みます。検出されたタグ内の URL からページが読み込まれると、検出されたタグ上の "target" 属性などが考慮されます。
- 「既存ウィンドウ」は、選択した既存ウィンドウにページが読み込まれることを指定します (ウィンドウ識別方法の説明を参照してください)。
- 「新しいウィンドウを開く」は、新規ウィンドウにページが読み込まれることを指定します。新しいウィンドウの名前は任意に指定でき、また、新規ウィンドウを開くためのウィンドウを登録することができます (ウィンドウ識別方法の説明を参照してください)。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

[スタイルを無視]
ロードされた HTML ページの CSS スタイルを無視するには、このオプションを選択します。このオプションは、アプリケーション ビューの下にあります。

Web ストレージ読み込み

「Web ストレージ読み込み」ステップは、ローカル ストレージまたはセッション ストレージあるいはその両方にデータを読み込みます。ローカル ストレージとセッション ストレージは、通常は Cookie に保存できる大量のデータを存続させるために一部の Web サイトによって使用されます。

読み込むデータはデータ フィールドで指定できますが (方法は下の例を参照してください)、このステップは、Web ストレージ抽出ステップを使用して抽出されたストレージ データの読み込みにも使用できます。

プロパティ

「Web ストレージ読み込み」アクションは、次のプロパティを使用して設定できます。

データ

ローカル ストレージまたはセッション ストレージあるいはその両方に読み込むデータを指定します。通常、このデータは「Web ストレージ抽出」アクションを使用した結果を保存する変数から取得されますが、個別に入力または生成することもできます。

データは JSON 形式で指定し、またオブジェクトの配列として指定する必要があります。各オブジェクトは次のプロパティを含む必要があります。

ストレージのタイプ

ストレージのタイプは、「セッション」または「ローカル」のいずれかである必要があります。セッション ストレージ は、現在のウィンドウに読み込まれ、セッション Cookie と同様に、最上位のウィンドウが存在する限りは存続します。ローカル ストレージ は、永続 Cookie と同様に、すべてのブラウザー ウィンドウ間で共有されます。

ドメイン

ストレージへのアクセスを有するサイトを含むドメインです。

ストレージ

ストレージを構成するアイテムの配列です。各アイテムは次のプロパティを有するオブジェクトです。

キー

ストレージ内のアイテムを探すために使用する名前です。

値

保存された値であり、文字列タイプである必要があります。

例

この例では、ローカル ストレージ内に値 "http://help.kofax.com" を有する「ヘルプ」と命名された 1 つのストレージ アイテムと、セッション ストレージ内の 2 つのストレージ アイテム、つまり、値 "Kofax RPA" を有する「製品」および値 "11" を有する「バージョン」を定義してみます。すべてのストレージ アイテムはドメイン "www.kofax.com" のために定義されます。

以下のデータを「Web ストレージ読み込み」ステップに入力します。

```
    [
            { "storage-type": "local",
              domain: "www.kofax.com",
              storage: [
                { key: "help",
                  value: "http://help.kofax.com"
                }
              ]
            },

            { "storage-type": "session",
              domain: "www.kofax.com",
              storage: [
                { key: "product",
                  value: "Kofax RPA"
                },
                { key: "version",
                  value: "11"
                }
              ]
            }
        ]
```

フィールド値ループ

このアクションは値のリストをループし、各イテレーションでテキスト フィールドに 1 つ入力します。

検出されるタグは、"text" または "password" タイプの <textarea> タグまたは <input> タグである必要があります。

<input> または <textarea> タグにイベント ハンドラが登録されている場合、値を入力すると、JavaScript の実行がトリガされます。

ループさせずにテキストを入力するには、テキストを入力アクションを使用します。

プロパティ

「フィールド値ループ」アクションは、次のプロパティを使用して設定できます。

値
ループする値。これは次のいずれかにすることができます：

a...z
a から z の文字。アルファベット順。

aa..zz
a から z の 2 文字の組み合わせ。アルファベット順。

数値範囲
整数の範囲です。範囲の端点を定義する数字と、各イテレーションで実行するステップのサイズを指定する必要があります。例： "From" を 0、"To" を 100、"Step" を 10 に設定すると、0、10、20、30、40、50、60、70、80、90、100 の数列でループするステップになります。

値のリスト
ループする値を明示的に指定します。値はカンマで区切るか、別の行にする必要があります。二重引用符で囲うことができます。値セレクターを使用して、変数などからリストを取得できます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

Excel 内ループ

「Excel 内ループ」アクションは、スプレッドシートのさまざまなエレメントをループします。この場合のエレメントは、シート、列、行、セルであり、ステップの範囲ファインダーで特定します。各イテレーションで、適切なエレメントが名前付き範囲としてマークされます。

プロパティ

「Excel 内ループ」アクションは、次のプロパティを使用して設定できます。

ループ オーバー
アクションがループする工レメントの種類を決定します。これには 4 つの可能性があります。

シート
アクションは、スプレッドシート文書のシートをループします。これを選択する場合、範囲ファインダーは必要ありません。

列
アクションは、範囲ファインダーによって検出される範囲の列をループします。

行
アクションは、範囲ファインダーによって検出される範囲の行をループします。

セル
アクションは、範囲ファインダーによって検出される範囲のセルをループします。

開始インデックス
ループに含める最初の工レメントの数字。数字を最初のタグから順方向にカウントするか、最後のタグから逆方向にカウントするかを指定することができます。

最終インデックス
ループに含める最後の工レメントの数字。数字を最初のタグから順方向にカウントするか、最後のタグから逆方向にカウントするかを指定することができます。

増数
ループで工レメントをスキップさせます。たとえば、増数を 2 に指定した場合、ループは 1 つおきに工レメントをスキップします。

逆方向にループ
逆順で一致する工レメントをループすることを選択します。同じ工レメントを順方向に逆順でループするようにループが実行されることに注意してください。これは、開始インデックスが、ループする工レメント選択の最初の工レメントを指し、ループ時に最初に通過する工レメントではないことを意味します (逆順の場合は最後の工レメントになります)。

範囲名
「自動」、「名前付き」という 2 つのオプションがあります。「自動」は範囲名として番号を与えます。「自動」で最初に与えられる番号は 1、次に与えられる番号は 2 であり、以下同様です。このステップの前に「自動」により追加的に番号を与えられた範囲が (同じページに) 挿入されると、番号は変わることがあります。

- 適切に選んだ名前を付けると、その範囲がどのような範囲であるかを容易に思い出せます。
- 明確に名前付けされた範囲は、名前付きの別の範囲がその前に挿入されても影響を受けません。
- 「名前付き範囲設定」で同じ名前を使用すると、その名前は、新しい範囲を参照するように設定されます (ステートフルなページ内ループに便利です)

パスワード取得

このアクションにより、Management Console のパスワード ストアからユーザー パスワードを取得し、変数に格納します。このステップは、機密情報を開示せずに使用するように設計されています。取得したパスワードをパスワード タイプ変数に格納し、ロボットで使用することができます。

パスワード情報を取得する前に、Management Console 管理者は、特定のロボットのリソース アクセス トークンを使用して Management Console 内にパスワード アクセス エントリを作成する必要があります。

ロボット、またはタイプ、スニペットなどのコンポーネントを Management Console にアップロードするたびに、ロードの新しいパスワード アクセス エントリを作成する必要があります。以前のエントリがパスワード アクセス リストに保持されます。管理者はこれらを手動で削除できます。

プロパティ

[パスワード取得] アクションは、次のプロパティを使用して設定できます。

ユーザー名
パスワードを取得するユーザーの名前を指定します。

ターゲット システム
パスワードを取得する外部システムの名前を指定します。このフィールドに入力する値は、パスワード入力の [ターゲット システム] プロパティの値と一致する必要があります。

管理者の設定により、ターゲット システムは、同じユーザーに対して異なるシステム (プロダクション、プリプロダクション、開発など) でパスワードを提供することができます。また、顧客が 1 つのパーティションにクレジット チェックのオートメーションを持ち、別のパーティションに会計要約機能を持つ場合、ターゲット システムを使用して、仮想マシンの異なる部分へのアクセスを区分することができます。

プロジェクト名
パスワードを取得するプロジェクトの名前を指定します。これにより、ユーザー名やターゲット システムなど、同じ資格情報を使用するプロジェクトをシームレスに操作できます。

以前のバージョンのロボットを RoboServer で実行する場合は、次の点に注意してください。

- 単一プロジェクトの場合: [プロジェクト名] フィールドを空白のままにして、エントリが検索基準に適合する場合、[パスワード取得] アクションは Management Console からパスワードを取得します。
- 複数のプロジェクトの場合: **[プロジェクト名]** フィールドを空白のままにし、エントリが検索基準に適合する場合、Management Console はエラーを返します。

変数
取得したパスワードを格納するパスワード タイプ変数の名前。

ディレクトリ作成

このアクションは、ロボットが実行されるローカル ファイル システムに新しいディレクトリを作成します。

このアクションを行うのは、「デザイン モードでの実行」オプションを選択している場合、Design Studio 内のデザイン モードで実行中に限られることに注意してください。

プロパティ

「ディレクトリ作成」アクションは、次のプロパティを使用して設定できます。

ディレクトリ
作成されるディレクトリのファイル システムのパスかファイルの URL です。これは、値セレクターを使用してさまざまな方法で指定できます。パスは絶対である必要があります。ドライブの名前 (該当する場合)、ディレクトリへのディレクトリ パスを含めます。代わりに、file:/C:/temp/newDir などのファイルの URL を指定することもできます。この場合は、エンコードされた URL である必要があります。区切り記号の / および \ は互いに入れ替えて使用できます。

ディレクトリが存在する場合はエラーを生成
ディレクトリがすでに存在する場合、アクションによって エラーが生成されます。

ディレクトリを作成
ディレクトリを作成する前に、パスに必要なディレクトリを作成するかどうかを指定します。選択されず、パスにディレクトリが存在しない場合、アクションは失敗します。

デザイン モードで実行
これが有効になっていると、アクションは Design Studio 内部のデザイン モードでも実行されます。これが無効になっていると、ロボットをデザイン モードでナビゲートするときにアクションは何も実行しません。

スナップショット生成

「スナップショット生成」ステップは、現在のウィンドウに存在するページを取得して、ファイル システムに格納します。これには、スナップショットが生成された時のようにそのページを表示するのに必要なすべてのスタイルシートや画像が含まれます。コンテンツが動的に生成されていたとしても、後でページを表示するのに、JavaScript やサーバーとの通信は必要ありません。

このステップは、最小実行 (ダイレクト) モードでのみサポートされています

関連するステップ アクション

相互にリンクされるページを大量にダウンロードし、共有リソースを再利用し、オフライン スナップショットの間のリンクを保持するには、「ページ再描画」ステップの使用を検討してください。「ページ再描画」ステップを使用するロボットには、ページとダウンロードするリソースの URL をフィードする外部のコントローラ アプリケーションが必要になります。

プロパティ

「スナップショット生成」ステップは、次のプロパティを使用して設定できます。

出力値フォルダ
スナップショットを出力するフォルダ。メイン ページ (現在のウィンドウのドキュメントに対応) は、index.html という名前のファイルに出力されます。

リソースのダウンロード
有効になっている場合、以前のステップにまだ存在しない画像やその他のリソースが、スナップショットの一部としてダウンロードして保存されます。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

マウス アウト

このアクションは、検出されるタグから離れるマウスの動きをエミュレートします。

このアクションは、マウス カーソルが合わされた時に開き、離れた時に閉じる JavaScript ベースのメニューを閉じるのに便利です。

タグに近付くマウスの動きをエミュレートするには、「マウスオーバー」アクションを使用します。動きではなくマウス クリックをエミュレートするには、「クリック」アクションを使用します。

プロパティ

「マウスアウト」アクションは、次のプロパティを使用して設定できます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

マウス オーバー

このアクションは、検出されるタグに近付くマウスの動きをエミュレートします。

このアクションは、マウス カーソルを合わせた時に開く JavaScript ベースのメニューをトリガするのに便利です。

タグから離れるマウスの動きをエミュレートするには、「マウスアウト」アクションを使用します。動きではなくマウス クリックをエミュレートするには、「クリック」アクションを使用します。

プロパティ

「マウスオーバー」アクションは、次のプロパティを使用して設定できます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

新しいウィンドウを開く

このアクションは新しいウィンドウを作成します。

プロパティ

「新しいウィンドウを開く」アクションは、次のプロパティを使用して設定できます。

ウィンドウ名
もし存在する場合は、新しいウィンドウの名前。新しいウィンドウに名前を付けない場合は、フィールドを空のまま残します。

ウィンドウ オープナー
もし存在する場合は、新しいウィンドウを開く元として登録されるウィンドウ (ウィンドウ識別方法に関する説明を参照してください)。ウィンドウを開く関係は、ウィンドウで JavaScript を実行する場合、重要になります。

次へ

このアクションは、[繰り返し] アクションを使用して作成した繰り返しループで別のイテレーションを要求します。

「次へ」アクションを使用すると、反復ループの各イテレーションで、別のイテレーションをリクエストすることができます。「次のステップ」のウィンドウ、ページなどが「繰り返し」ステップに戻され、次のイテレーションの「繰り返し」ステップからの出力値となります。与えられたイテレーションで「次のステップ」が実行されない場合、そのイテレーションが最後になり、反復ループは終了します。

詳細は、「繰り返し」アクションを参照してください。

プロパティ

このアクションにはプロパティがありません。

セル結合解除

このアクションは、rowspan や colspan が使用されている時に追加のセルを入れてセルの結合を解除します。これによって、テーブルの列や行のイテレーションが簡単になります。セル結合解除は、CSS を使用して作成されているのではなく、HTML ソースに colSpan や rowSpan が使用されている場合にのみ動作します。

JavaScript またはフォームの送信はページの構造に依存している可能性があるため、このアクションによってページが変更されると、JavaScript またはフォームの送信が機能しなくなる可能性があります。ただし、このアクションは通常、そこから先へのナビゲーションが発生しないデータ テーブルに適用されるため、これが問題になることはほとんどありません。

プロパティ

以下のプロパティを使用して [タグ ファインダー] を設定できます。

フィールドからコピーしない
このオプションを選択すると、追加のセルが挿入される時に、すべてのフォーム フィールド (input、select、button、textarea) がコピーされません。

例

colSpan

前

1	2
3	
4	5

後ろ

1	2
3	3
4	5

rowSpan

前

1	2	X
3		X
	4	X
5	6	X

後ろ

1	2	X
3	2	X
3	4	X
5	6	X

colSpan と **rowSpan**

前

1	2	3		4
5				6
7				
8	8	8	8	8

後ろ

1	2	3	3	4
5	5	3	3	6
7	7	3	3	6
8	8	8	8	8

古いステップ

より新しいバージョンの Kofax RPA では、いくつかのステップが置き換えられり、除去されたりしています。

ヘルプを得ようとしているステップは、以前のバージョンのプログラムの 1 つで作成されている可能性があります。対応する Kofax RPA のバージョンのオンライン ヘルプを参照してください。

より新しいバージョンの Kofax RPA で口ボットを編集する場合は、可能であれば既存のステップの 1 つで古いステップを置き換えます。アップグレードの手順については、『Kofax RPA リリース ノート』および『Kofax RPA アップグレード ガイド』を参照してください。Kofax RPA の使用において発生した問題を解決するには弊社のカスタマー サポート ポータルを使用してください。

変数を開く

このアクションは、ビューで変数の属性、あるいは単純なタイプの変数を開きます。ビューでこれを操作することができます。これによって、変数の内容を視覚的に調整して、Web サービスのパラメータに必要な内容などを正確かつ簡単に作成することができます。

このステップは、XML、JSON、Excel 変数に対してのみ動作します。

プロパティ

「変数を開く」アクションは、次のプロパティを使用して設定できます。

変数
このプロパティは、どの変数を開くか指定します。これによって読み込むには、変数は XML、JSON、または Excel である必要があります。9.3 以前の口ボットは、現在非推奨の XML 属性タイプを使用しており、開くには、新しい XML タイプにアップグレードする必要があります。単純なタイプの変数については、変数を編集して新しい XML タイプを選択することでこれが完了します。コンプレックス タイプの変数については、関連する .type のファイルは、古いものではなく新しい XML 属性タイプを使用するように変更する必要があります。

キープレス

このアクションは、キーボードの特定のキー プレスを工ミュレートします。

任意のタグを検出し、フォーカスを仮定してキーボード イベントを受信することができます。

このアクションは、<TAB> などのキー プレスを送信するのに便利です。

プロパティ

「キー プレス」アクションは、次のプロパティを使用して設定できます。

押下するキー
押されるキー。

事前定義
リストから選択します。事前定義されるキーのリストには、最も一般的に使用されるキーが含まれます。

値
押されるキーの値を指定します。値は、10 進数の Qt キー コードの 1 つである必要があります。Qt で使用されるキーの名前については、Qt のドキュメントを参照してください。

変数
押されるキーのキー コードを含む変数を指定します。キー コードは、10 進数の Qt キーコードの 1 つである必要があります。

エクスプレッション
エクスプレッションを指定します。

コンバータ
コンバータを指定します。

エレメントにフォーカス
キーを押す前のエレメントにフォーカスします。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

データベース照会

[データベース照会] アクションは SQL クエリをデータベースに送信し、結果全体をループします。エクスプレッションを使用して SQL を指定する必要があります。結果ループの個々のイテレーションで、結果セット内の現在の行の値を変数に割り当てることができます。

- SQL エディターには、結果セットの結果が 20 件のみ表示されます。
- SQL クエリは、ステップごとに 1 つずつ、個別のトランザクション内で実行されます。複数の連続するデータベース照会の呼び出し、または SQL 実行の呼び出しにまたがるトランザクションを使用することはできません。
- ストアド プロシージャの呼び出しは、「データベース照会」ステップではサポートされません。

プロパティ

以下のプロパティを使用して [データベース照会] アクションを設定できます。

データベース

このアクションでクエリを送るデータベースを、Design Studio で利用できるデータベースのドロップダウン リストを使用して選択します。

SQL クエリ

このフィールドには有効な SQL クエリがエクスプレッションの形式で含まれている必要があります。このエクスプレッションの値が、選択したデータベースに送られます。[編集] ポップアップ ダイアログを使用して、SQL クエリをテストし、出力のサンプルを表示することができます。

タイムアウト

SQL クエリを送信するタイムアウトを秒単位で指定します。変数または値のオプションを選択して、希望するタイムアウトを定義します。値はゼロより大きい必要があります。

タイムアウトは DBMS を通じて強制され、タイムアウトに達すると、[データベース照会] アクションの実行は停止し、データベース固有のエラー メッセージが表示されます。

それぞれの DBMS タイプおよび JDBC ドライバーに応じて、タイムアウトの処理方法が異なる場合があります。たとえば、一部のシステムにおいては、特定のステートメントの途中でのタイムアウトの発生が許可されていないこともあります。

変数マップ

結果列から変数へのマッピングを指定します。プラス符号をクリックして新しいマッピングを追加し、マイナス符号をクリックして既存のマッピングを除去します。マッピングは列名と変数名から構成されます。列名は SQL クエリによって返される列の名前と一致している必要があり、変数名は既存の変数のリストから選択されます。列のタイプは、選択した変数のタイプと一致している必要があることに注意してください。一致していないと、実行中にエラーが生成される可能性があります。つまり、テキスト列を整数変数に保存しようとすると、エラーが発生します。

ループ中に取得

このプロパティが有効になっていると、イテレーションごとにループによって結果行が必要とされる場合にのみデータベースから結果行が取得されます。実行については以下の注記を参照してください。

デザイン モードでの最初の行

Design Studio のデザイン モードでのみ使用されるこのプロパティは、(1 からカウントして) アクセス可能な最初のイテレーションまたは照会結果行の番号です。実行については以下の注記を参照してください。

デザイン モードで使用する行数

Design Studio のデザイン モードでのみ使用されるこのプロパティは、イテレーションが利用できる結果行の最大数を指定します。実行については以下の注記を参照してください。

注意：[ループ中に取得] が無効または有効のどちらに設定されているかに応じて、結果行の取得は 2 つの異なる方法で行われます。

- 無効：最初のイテレーションが実行される前にすべての結果行が取得され、メモリに保存されます。したがって、データベース接続は可能な限り最短の時間だけ予約され、結果はループの一部であるいずれのステップ (データベース データ登録ステップなど) の影響も受けません。一方、利用可能なメモ

リはエラーなしで処理できる結果行の数を制限します(リリース 8.3 まではこれが利用可能な唯一のオプションでした)。

- 有効：ループのイテレーションの実行に結果行が必要とされるたびに、データベースから 1 度に 1 つずつ結果行が取得されます。したがって、ステップは非常に多数の結果行を処理できますが、ループのすべてのイテレーションが実行を終了するまで、データベース接続はオープン状態にとどまります。その影響で、ループの実行中に SQL クエリによって参照されるデータベース テーブルに変更を加えると、結果が影響を受ける可能性があります。ただし、特定の状況で実際に変化が目に見えるかどうかは、多くの要因が重なり合って決まります。

Design Studio のデバッグ モードでは、ループ中に取得を行うと、ブレークポイントで、またはシングルステッピング中に実行が停止している間、データベース接続はオープン状態にとどまります。アクティブでない接続に対するタイムアウトがデータベースで設定されている場合は、長い中断の後、ロボットの実行を再開すると、データベース エラーが発生することがあります。

Design Studio のデザイン モードでは、結果行が常にループの開始前に取得されるため、[ループ中に取得] が事実上、無効になります。このようになるのは、異なるイテレーション間でインタラクティブに切り替えを行えるようにするためです。この操作に使用されるメモリの量を制限するには、[デザイン モードでの最初の行] および [デザイン モードで使用する行数] を組み合わせて、読み込む結果行のサブセットを指定します。たとえば、

- デザイン モードでの最初の行 = 301
- デザイン モードで使用する行数 = 100 の場合、

ループは結果行 301 から 400 を繰り返します (ただし、SQL クエリがそれだけ多くの行を返すことを条件とします)。

HTTP 通信

「HTTP 通信」アクションは HTTP リクエストを Web サーバーに送信します。レスポンスが処理される方法はメソッドによって異なりますが、一般に、ステータス コードとレスポンス ヘッダーがページ読み込みオプションの一部として定義された変数で返されます。

プロパティ

「HTTP 通信」アクションを、そのプロパティによって設定することができます。

ロケーション

このプロパティで、開く URL を指定します。URL は、URL セレクターを使用してさまざまな方法で指定できます。URL セレクターの詳細については、ページ読込 トピックの 「URL」セクションを参照してください。

メソッド

ここでは、使用するメソッドを指定します。

- GET は GET HTTP リクエストの実行に使用されます。
- POST は POST HTTP リクエストの実行に使用されます。POST リクエストでは、複数のパラメーターを名前/値ペアとして指定するか、リクエストの本文全体を渡します。パラメーター付きでリクエストを指定する場合は、パラメーターのエンコードに POST (application/x-www-form-urlencoded) または MULTIPART (multipart/form-data) のどちらを使用するかを選択する必要があります。リクエストの本文全体 (未加工) を渡す場合は、リクエスト データのコンテンツ タイプを指定する必要があります。
- PUT は PUT HTTP リクエストの実行に使用されます。PUT リクエストでは、複数のパラメーターを名前/値ペアとして指定するか、リクエストの本文全体を渡します。

以下の設定は GET リクエスト、POST リクエスト、PUT リクエストに共通です。

パラメータ
ここでは、複数のパラメーターを名前/値ペアとして指定できます。新しいパラメータを追加するには '+' をクリックします。

POST リクエストと PUT リクエストでは、MULTIPART エンコーディングを選択してファイル アップロードを有効にすることができます。ファイル アップロード パラメータの値としてバイナリ変数が選択されている場合は、バイトがそのまま送信されます。Base 64 エンコーディングを使用する場合は、パラメータの値をエクスプレッション base64Encode(data) にする必要があります。data はバイナリ値が含まれた変数の名前です。その場合は、値 base64 を Content Transfer Encoding として指定することをお勧めします。そうしない場合は、このフィールドを通常通り空白のままにすることができます。

受け入れ
これはレスポンスとして受け入れられるコンテンツ タイプです。デフォルトでは、あらゆるタイプのレスポンスが受け入れられます。値セレクターを使用して、受け入れられるコンテンツ タイプを複数の方法で指定できます。

エンコーディング
これはリクエスト内の特殊文字のエンコードに使用されるエンコーディングです。

次の場所に保存
これは結果の保存先となる必須変数の名前です。

- HEAD は HEAD HTTP リクエストの実行に使用されます。HEAD リクエストはレスポンスの一部としてデータを返さないため (ステータス コードとレスポンス ヘッダーのみを返します)、ページ ロード オプションの一部として定義された変数を使用してこのデータにアクセスします。GET リクエストは HEAD リクエストのシミュレーションを行うために使用できます。その結果、ヘッダー情報が受信されると直ちに破棄される GET HTTP リクエストが送信されます。つまり、レスポンス全体は読み込まれません。
- OPTIONS は OPTIONS HTTP リクエストの実行に使用されます。OPTIONS リクエストはレスポンスの一部としてデータを返さないため (ステータス コードとレスポンス ヘッダーのみを返します)、ページ ロード オプションの一部として定義された変数を使用してこのデータにアクセスします。レスポンスには通常、サーバーによって実装され、リクエストされたリソース (URL) に適用可能なオプションの機能を示すヘッダー フィールドが含まれます。以下に例を示します。OPTIONS、TRACE、GET、HEAD

オプション
ロボットのオプションはステップ自身のオプションで上書きできます。変更されたオプションはロボットの設定のオプションを上書きし、[オプション] ダイアログに表示されるときにアステリスクでマークされます。

追加のリクエスト ヘッダーが必要な場合は、それを [オプション] の下でも設定できます。

属性除去

このアクションは XML 内のタグから属性を除去します。

このステップは XML 変数に対してのみ機能します。このステップは、XML の正当性の確認もエンティティの解決もしません。

プロパティ

以下のプロパティを使用して「属性除去」アクションを設定できます。

属性名
除去する属性の名前。

列除去

このアクションは選択された列をスプレッドシートから除去します。

プロパティ

このアクションにはプロパティがありません。

コンテンツ除去

このアクションはタグのすべてのコンテンツを除去します。つまり、タグのコンテンツを消去し、タグそのものだけを残します。

このステップは XML 変数に対してのみ機能します。このステップは、XML の正当性の確認も工ンティティの解決もしません。

プロパティ

「コンテンツ除去」アクションはプロパティを持っていません。

Cookie 除去

「Cookie 除去」アクションは現在の Cookie のセットから 1 つ以上の Cookie を除去します。

プロパティ

以下のプロパティを使用して「Cookie 除去」アクションを設定できます。

ドメイン パターン
Cookie のドメインに一致するパターンを指定します。パターンは Cookie のドメイン全体に一致する必要があります。

パス パターン
Cookie のパスと照合するパターンを指定します。パターンは Cookie のパス全体に一致する必要があります。

名前パターン
Cookie の名前と照合するパターンを指定します。パターンは Cookie の名前全体に一致する必要があります。

値パターン
Cookie の値と照合するパターンを指定します。パターンは Cookie の値全体と照合させる必要があります。

例
ドメイン ".kofax.com"、パス "/"、値 "123" を持つ "PREF" という名前の Cookie があるとします。この Cookie は名前パターンを "PREF" に設定することによって除去することができます。ただし、そうすると、Cookie のドメイン、パスまたは値に関係なく、"PREF" という名前を持つすべての Cookie が除去されます。したがって、この Cookie のみを除去する必要がある場合は (これが、この名前、ドメイン、パスを持つ唯一の Cookie であると仮定して)、ドメイン パターンを ".kofax.com"、パス パターンを "/"、名前パターンを "PREF"、値パターンを "123" に設定します。
すべての Cookie を無条件で除去するには、すべてのパターンを ".*" に設定します。

JSON 除去

このアクションは、見つかった JSON を JSON 値から除去します。このアクションによって、オブジェクトからのプロパティ (名前/値ペア) の除去、配列からの項目の除去、または JSON 値全体の除去 (この場合、結果は空の JSON 値になります) を行います。

このステップは JSON 変数に対してのみ機能します。

プロパティ

「JSON 除去」アクションはプロパティを持っていません。

行除去

このアクションは選択された行をスプレッドシートから除去します。

プロパティ

このアクションにはプロパティがありません。

シート除去

このアクションは選択されたシートをスプレッドシートから除去します。

プロパティ

このアクションにはプロパティがありません。

テーブル行除去

このアクションは特定の数の列 (<td> タグと <th> タグ) を持っていない入力 <table> タグ内のすべての行 (<tr> タグ) を除去します。

プロパティ

以下のプロパティを使用して「テーブル行除去」アクションを設定できます。

列の最小数
テーブル行に常に存在する必要のある列の最小数を入力します。

列の最大数
テーブル行に存在する列の最大数を入力します。

[列の最小数; 列の最大数] の範囲に収まらない数の列を持つすべての行は除去されます。

最終的にテーブルのすべての行が除去された場合、「テーブル行除去」アクションはエラーを生成します。

例
次の入力を考えてみましょう。

```
<table>
  <tbody>
    <tr><td> 1 </td></tr>
    <tr><td> 2 </td><td> 2 </td></tr>
    <tr><td> 3 </td><td> 3 </td><td> 3 </td></tr>
  </tbody>
</table>
```

以下の条件を仮定します。

- 列の最小数が 1 に設定されている
- 列の最大数が 2 に設定されている

この設定では、3 つの <td> タグを持つ最後の行が除去されます。

タグ除去

「タグ除去」アクションは検出されたタグをそのタグの親ノードから除去します。

このステップは XML 変数に対してのみ機能します。このステップは、XML の正当性の確認もエンティティの解決もしません。

プロパティ

このアクションにはプロパティがありません。

タグ範囲除去

「タグ範囲除去」アクションはタグの範囲、つまり 1 つのタグから別のタグの間のすべてのタグを除去します。

プロパティ

以下のプロパティを使用して「タグ範囲除去」アクションを設定できます。

開始タグ
除去するタグ範囲の開始タグを指定します。

開始タグを除去
開始タグを除去するかどうかを指定します。

終了タグ
除去するタグ範囲の終了タグを指定します。

終了タグを除去
終了タグを除去するかどうかを指定します。

タグ除去

「タグ除去」アクションは、2 つの規則セットに基づいて指定された条件に一致するタグを、検出されたタグから除去します。

設定

「タグ除去」アクションは、除去するタグを定義する規則セットおよび除去規則からタグを除外するための規則セットを定義することによって設定されます。

リストへの規則の追加とリストからの規則の除去

2 つのリストのどちらかに規則を追加するには、リストの下の '+' をクリックします。規則の除去は、リスト内の規則を選択し、下の '-' をクリックすることによって行われます。規則の順序を変更するには、リストの下の矢印を使用します。規則を追加した後、リストの下の [編集] ボタンを使用して規則を編集することができます。

除去規則のリストで規則をまったく定義しないと、除外規則のいずれとも一致しないすべてのタグが除去されます。

タグ照合規則の設定

リストの下の '+' または [編集] をクリックして開くタグ マッチャー規則エディターでは、除去するタグまたは除去対象から除外するタグを照合するための 3 つの異なる方法の選択肢が提示されます。

[タグ マッチャー] リストから以下のいずれかのオプションを選択します。

- この名前を持つタグは、`タグ名` に指定された名前と名前が完全に一致するすべてのタグを照合します。
- この名前と属性を持つタグは、`タグ名` に指定された名前と名前が完全に一致し、属性の条件に一致するすべてのタグを照合します。
- このパターンと一致するタグは、タグ属性を含む開始タグ (< と > の間のすべて) の正規表現を照合します。タグ照合パターンを反転させて、パターンと一致しないすべてのタグを照合することもできます。
- このパターンと一致するテキストは、検出されたタグ内部のテキストの正規表現と照合します。正規表現を空にすると、すべてのテキストと照合します。
- コメントはすべてのコメントと照合します。

タグ照合方法のプロパティ

「子工レメントを含む」オプション

この規則で「子工レメントを含む」チェックボックスをオン (デフォルト) にすると、子工レメントが含まれます。このオプションを除去規則でオンにすると、タグ (およびそれに対応する終了タグ) だけでなく、タグのすべてのコンテンツも除去されます。このオプションを除外規則でオンにすると、(たとえ子工レメントのいずれかが除去規則に一致していても) タグだけでなく、タグの子工レメントも残ります

除外規則では、このオプションが最優先されます。つまり、タグの子工レメントを含めるように設定されている除去規則の 1 つにタグが一致していて、そのタグのいずれかの子タグが除外規則の 1 つに一致している場合、その子タグは (子工レメントを含めて) 残ります。

「この名前を持つタグ」照合方法のオプション
タグの名前
この規則によって照合されるタグの正確な名前を指定します。

「この名前と属性を持つタグ」照合方法のオプション
タグの名前
この規則によって照合されるタグの正確な名前を指定します。属性条件のみに基づいてタグを照合する必要がある場合は、このオプションを空のままにすることができます。

属性名
照合する属性の正確な名前を指定します。

属性値パターン
指定された属性の属性値と一致する必要があるパターンを指定します。

「このパターンと一致するタグ」照合方法のオプション
タグの名前
この規則によって照合されるタグの正確な名前を指定します。

パターンを反転
このパターンに適合しないすべてのタグに規則を照合させるには、このオプションを選択します。

「このパターンと一致するテキスト」照合方法のオプション
パターン
テキストと一致する必要のあるパターンを指定します。このパターンを空のままにすると、すべてのテキストが一致します。

「コメント」照合方法のオプション
この方法はオプションを持っておらず、常にすべてのコメントと一致します。除去する特定のコメントを指定するには、タグ ファインダーを使用します。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

ファイル名変更

このアクションではロボットが実行されるローカル ファイル システム上のファイルまたはディレクトリの名前を変更します。

このアクションを行うのは、「デザイン モードでの実行」オプションを選択している場合、Design Studio 内のデザイン モードで実行中に限られることに注意してください。

プロパティ

以下のプロパティを使用して「ファイル名変更」アクションを設定できます。

ファイルまたはディレクトリ

これは、名前を変更するファイルまたはディレクトリのシステム パスあるいはファイル URL です。これは、値セレクターを使用してさまざまな方法で指定できます。パスは絶対である必要があります。ドライブの名前 (該当する場合)、ディレクトリへのディレクトリ パスを含めます。あるいは、パスを file:/C:/temp/oldFile.txt などのファイル URL にすることもできます。その場合はパスを URL エンコードする必要があります。区切り記号の / および \ は互いに入れ替えて使用できます。

新しい名前

これは、ファイルまたはディレクトリの新しい名前です。../SomeOtherFolder/newText.txt のようなディレクトリ構造を含む新しい名前を指定することによってファイルを異なる場所へ移動するときに使用できる点で、これは実質的に相対パスです。

デザイン モードで実行

これが有効になっていると、アクションは Design Studio 内部のデザイン モードでも実行されます。これが無効になっていると、ロボットをデザイン モードでナビゲートするときにアクションは何も実行しません。

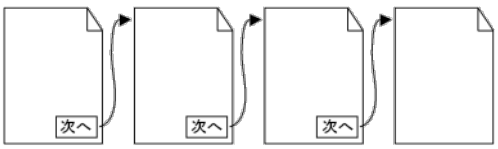
繰り返し

このアクションでは、「次へ」アクションとともに繰り返しループを作成します。

[繰り返し] アクションは繰り返しループの開始位置をマークします。続くステップでは、[次へ] アクションを使用して、ループの別のイテレーションをリクエストすることができます。[次へ] ステップのウィンドウ、ページなどは、[繰り返し] ステップに送り返され、次のイテレーションの [繰り返し] ステップからの出力になります。指定したイテレーションで [次へ] ステップが実行されない場合、そのイテレーションが最後になり、反復ループは終了します。

ロボットは、パスが完了するまで、[次へ] ステップでパスに従って続行されることに注意してください。また、このパスでループする結果を提供するデータベース クエリがある場合、ロボットはデータベース クエリの結果をループしてから、[繰り返し] ステップに戻ります。

[繰り返し] アクションは、以下のように [次ページ] リンクで結合されたページをループする場合に特に役立ちます。



[繰り返し] アクションの使用例については、次ページにリンクしている各ページを参照してください。

タグ書き換え

「タグ置き換え」アクションは検出されたタグを新しいタグに置き換えます。JavaScript の実行がオプションで無効になっていない場合、新しいタグの HTML 内にある JavaScript が実行されます。

プロパティ

以下のプロパティを使用して「タグ置き換え」アクションを設定できます。

新しいタグの **HTML**
新しいタグの HTML を指定します。HTML の指定は、以下で説明する複数の方法で行うことができます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

HTML の指定

新しいタグの HTML は以下の方法で指定できます。

HTML
単純に新しいタグの HTML を記述します。

エクスプレッションからの **HTML**
生成される結果が新しいタグの HTML として使用されるエクスプレッションを記述します。

エクスプレッションとパターンから **HTML** を作成
検出されたタグと照合されるパターンと、生成される結果が新しいタグの HTML として使用されるエクスプレッションを記述します。検出されたタグのパーツを使用して新しいタグの HTML を作成する場合は、この方法で HTML を指定します。

パターン
「タグ書き換え」アクションで検出されたタグと照合されるパターン。パターンは検出されたタグの全体と一致している必要があり、一致しない場合はエラーが発生します。

照合対象
検出されたタグのどの部分とパターンを照合するかを指定します。

- 「テキストのみ」は、検出されたタグ内のテキストのみとパターンを照合するよう指定します。
- "HTML" は、パターンが検出されたタグの HTML と照合します。
  - 一致の検索を実行する前に HTML の書式を自動設定するには、[HTML の書式設定] を選択します。
  - URL を含む属性値が常にアンパサンドでエンコードされるようにするには、[URL をエンコード] を選択します。
  - 一致の検索を実行する前に絶対 URL を相対 URL に変換するには、[相対 URL を抽出] を選択します。

大文字と小文字を無視
このチェックボックスをオンにすると、パターンが入力と照合されるときに大文字と小文字の区別は無視されます。たとえば、"KoFaX" は "kOfax" と同じであるとと見なされます。

エクスプレッション
このフィールドには、その結果が新しいタグの HTML として使用されるエクスプレッションが含まれます。エクスプレッションは、$n 表記を使用して、「パターン」フィールド内のパターンのサブマッチを参照することができます。たとえば、$1 をエクスプレッションに入力して、パターン内の最初のサブマッチ (つまり、パターン内の最初の 1 組の括弧内のコンテンツと一致するテキスト) を得ることができます。

XML 変数から HTML を変換
内容が HTML に変換されて新しいタグの HTML として使用される XML 変数を選択します。

オプション
[デフォルト オプションを使用] プロパティにチェックが入っていない場合に使用するオプションを設定します。

セッションの復元

「セッションの復元」アクションは、セッションの保存アクションを使用して、以前別の口ボット実行によって変数に保存されたセッションを復元します。セッションの再利用の詳細については、「セッションの保存」のヘルプを参照してください。

指定されたパラメーターにセッションが (まだ保存されていないなどで) 存在しない場合、「セッションの復元」アクションはエラーを生成します。

プロパティ

以下のプロパティを使用して「セッションの復元」アクションを設定できます。

復元元
変数
セッション変数を選択します。

[セッション プール] オプションはステップから除去されました。[セッション プール] オプションが指定されている古い口ボットを開くと、ツールチップ付きの警告が表示されます。

ブラウザ再開

このアクションは、指定の待機条件が満たされるか、ブラウザが待機状態になった場合に (どちらか早い方を優先) ブラウザを再開し、実行します。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

値返却

このアクションはロボットから値を返却します。

値はこのロボット実行を開始したクライアントに返却されます。このクライアントは、通常、Kofax RPA API を使用して RoboServer でロボットを実行する Java コードまたは .Net コードです。「値返却」ステップは、値を Kapplet 結果および REST レスポンスとして返却するときにも使用されます。

<data conref="../../All_Variables/variables.xml#variables/KKat"></data> のバージョン 7.2 より前のバージョンでは、返却された値を、データベースに値を保存したストレージ環境によって処理することもできます。<data conref="../../All_Variables/variables.xml#variables/KKat"></data> バージョン 7.2 以降では、値は「データベース データ登録」アクションを使用してデータベース データ登録されます。

プロパティ

以下のプロパティを使用して「値返却」アクションを設定できます。

[基本] タブ変数
保存されている値を返却する必要がある変数を選択します。

エラー処理タブ必須属性がない場合
ここでは、1 つ以上の必須属性がない場合に発生する動作を次の中から選択します: [エラー生成]、[ログ メッセージを出力]、または [無視]。

ページ再描画

「ページ再描画」ステップは、現在のウィンドウにある HTML ページを取り込み、そのページの HTML コンテンツおよびそのページに存在する可能性のあるすべてのフレームを抽出し、さらに、他のページへのリンクおよびそのページが依存する画像、スタイル シート、その他のリソースの URL を出力します。後から、抽出時とまったく同じ状態のページをオフラインで表示することができます。

抽出された HTML は、すでにページとページのフレームを読み込み、追加のコンテンツを生成する可能性のあるすべての JavaScript を実行したことによって取得された結果に相当するため、すべての JavaScript とイベント ハンドラーは抽出された HTML から除去されます。まず、ユーザー指定の変換に従ってページのすべての URL が書き換えられ、その後、書き換えられた URL が相対 URL に変換されます。インライン スタイル シート内の URL も書き換えられます。

含まれている URL がステップによって出力される外部スタイル シートは、同様の変換を適用し、スタイル シートで参照されるインポートされたスタイル シートと画像の URL を書き換える「CSS 再描画」ステップを使用して実行する必要があります。

「ページ再描画」ステップは、ロボット上で書き換えられるページ、スタイル シート、その他のリソースの URL をフィードする外部コントローラーを持つロボットで使用することを意図しています。

関連するステップ アクション

ページの簡易オフライン スナップショット生成には、「スナップショット生成」ステップを使用できます。このステップ アクションではロボットを外部アプリケーションで制御する必要はありませんが、1 つのステップで、必要なすべてのリソースをダウンロードしてファイル システムに保存し、完全なスタンドアロンのスナップショットを作成します。

「ページ再描画」ステップと異なり、「スナップショット生成」ステップでは、異なるスナップショット間のリンクを保持せず、スナップショット間の共有リソースを再利用しません。

このステップの実行はライセンス キーによって制御されます。

プロパティ

以下のプロパティを使用して「ページ再描画」ステップを設定できます。

元のページ **URL**

現在のウィンドウにあるページの元の URL が含まれた変数を指定します。これはページの読み込みに使用された URL です。リクエストされたページと異なるページへサーバーがリダイレクトした場合、ページの現在の URL は元の URL と異なる可能性があります。

データ コンバータ

ページの URL に対して実行する変換を指定するデータ コンバータ。これを使用して URL からファイル システム上の場所への変換を指定することができます。データ コンバータは、ステップが元のページ URL を基準とした相対 URL に自動的に変換する絶対 URL (ファイル URL である可能性もある) を出力する必要があります。高度な URL の書き換えには、[JavaScript を使用して変換] データ コンバータを推奨します。

抽出されたページ

抽出されたページの保存先となる変数。ステップは現在のウィンドウにあるページの HTML および個々のフレームの HTML を抽出します。抽出された HTML は、各ページの元の URL および書き換えられた URL が含まれた JSON 形式で出力されます。ただし、メイン ページのみが指定された元の URL を持ちます。

JSON 出力値をウィンドウに読み込むには、JSON をコンテンツのソースとして含む変数の名前を指定して「ページ生成」ステップを使用します。ステップの [オプション] で、コンテンツ タイプが JSON であり、エンコーディングが UTF-8 であることを明示的に示す必要が生じることもあります。

URL

抽出された URL の保存先となる変数。ステップは、ページとページのフレームによって直接リンクされているすべてのページ、画像、スタイル シートおよびその他のリソースの URL を抽出します。リンクされているスタイル シートとページ自体にも URL が含まれている可能性があることに注意してください。それらの URL はリストに含まれません。

URL は JSON 形式で出力され、元の URL と各 URL の書き換えられた絶対 URL が提供されます。また、URL が出現するコンテキストによって決定される URL のタイプが提供されます。たとえば、<IMG> タグ内のすべての URL はタイプ IMAGE でマークされます。

利用可能なタイプは以下の通りです。

PAGE

アンカー タグ内のリンク。ページがまだ読み込まれていないため、これは、そのページのコンテンツ タイプに関する情報を意味するものではないことに注意してください。

IMAGE

イメージ。

STYLESHEET

外部 CSS スタイル シート。

RESOURCE
フレーム内の PDF または Flash オブジェクトなどのバイナリ リソースなど。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

CSS 再描画

「CSS 再描画」ステップは、「ページ再描画」ステップと組み合わせて使用します。このステップ アクションは、スタイル シート内で検出されたすべての URL をユーザー指定の変換に従って書き換え、さらにそれら を、スタイル シート URL を基準とした相対 URL になるように変換します。

「CSS 再描画」ステップは、「ページ再描画」ステップによって検出された外部スタイル シートの URL に対して適用することを意図しています。これら 2 つのステップの URL 変換 (データ コンバータのリストによって指定される) を同じにすることが重要です。

このステップの実行はライセンス キーによって制御されます。

プロパティ

スタイル シート **URL**
読み込まれるスタイル シートの URL。「ページ再描画」ステップの出力から取得された場合、この URL はタイプ STYLESHEET の抽出 URL の originalURL プロパティにあります。

データ コンバータ
ページの URL に対して実行する変換を指定するデータ コンバータ。これを使用して URL からファイル システム上の場所への変換を指定することができます。データ コンバータは、ステップが後に元のページ URL を基準とした相対 URL に自動的に変換する絶対 URL (ファイル URL である可能性もある) を出力する必要があります。高度な URL の書き換えには、[JavaScript を使用して変換] データ コンバータを推奨します。上で説明したように、データ コンバータが、スタイル シート URL を出力した「ページ再描画」ステップと同じ方法で URL を変換することが重要です。

出力値
書き換えられたスタイル シートの保存先となる変数。

URL
抽出された URL の保存先となる変数。ステップは、読み込まれたスタイル シートによって直接参照されるすべての画像およびインポートされたスタイル シートの URL を抽出します。インポートされたスタイル シート自体にも URL が含まれている可能性があることに注意してください。それらの URL はリストに含まれません。

URL は JSON 形式で出力され、元の URL および各 URL の書き換えられた絶対 URL が提供されます。また、URL が出現するコンテキストによって決まる URL のタイプも提供されます。たとえば、インポートステートメント内のすべての URL はタイプ STYLESHEET でマークされます。

利用可能なタイプは以下の通りです。

IMAGE

イメージ。

STYLESHEET

インポートされた CSS スタイル シート。

JSON 出力値をウィンドウに読み込むには、JSON をコンテンツのソースとして含む変数の名前を指定して「変数を開く」ステップを使用します。ステップの [オプション] で、コンテンツ タイプが JSON であり、エンコーディングが UTF-8 であることを明示的に示す必要が生じることもあります。

オプション

ステップの読み込みオプションをロボットのオプションよりも優先させることができます。オプション ウィンドウでアステリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

セッションの保存

「セッションの保存」アクションでは、ブラウザ セッションを保存します。ロボット間でセッションを転送するために、セッションを含む変数をデータベース データ登録することができ、他のロボットがそれをデータベースから読み込んで復元できます。Kofax RPA API を活用したソリューションを行う場合、セッションをロボットからの出力値として保持して、その後、それを別のロボットへの入力値として使用できます。

デフォルトでは、完全なブラウザの状態を保存します。これによって、必要なスペースが大きくなりすぎる場合は、ページ、ページの URL、および関連する Cookie と認証で構成される、部分的な状態を保存できます。

セッションは変数に保存されて、「セッションの復元」アクションを使用して、別のロボットの実行によって後で復元できます。

プロパティ

「セッションの保存」アクションは、次の各プロパティを使用して設定できます。

保存場所

変数

セッション変数を選択します。

[セッション プール] オプションはステップから除去されました。[セッション プール] オプションが指定されている古いロボットを開くと、ツールチップ付きの警告が表示されます。

完全なブラウザの状態を保存

完全なブラウザの状態を保存するか、または、ページ、ページの URL、および関連する Cookie と認証で構成される、部分的な状態を保存するかを、選択します。

スクロール

このアクションでは、ページやタグのスクロールをエミュレートします。これが特に有効なのは、ページまたはコンテンツのタグをスクロールして初めてフル コンテンツが表示されるサイトです。スクロール ステップでは、ブラウザ表示内でのスクロール バーの位置を変更しませんので注意してください。

最も一般的なところでは、「スクロール」ステップの使用目的は、ドキュメント全体のスクロールです。この場合、タグ ファインダをステップに追加することはできません。独自のスクロール バーを備える特定のタグをスクロールするには、そのタグを選択するタグ ファインダを追加します。

プロパティ

「スクロール」アクションは、次の各プロパティを使用して設定できます。

水平スクロール
現在のスクロール位置を基準とする、右へのスクロールのピクセル数を指定します。左にスクロールするには、負の数を指定します。値セレクターを使用してさまざまな方法で値を指定することができます。

垂直スクロール
現在のスクロール位置を基準とする、下へのスクロールのピクセル数を指定します。上にスクロールするには、負の数を指定します。値セレクターを使用して複数の方法で値を指定することができます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームの エレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

指定タグまでスクロール

このステップは、デフォルト (WebKit) ブラウザで使用する必要があります。

このアクションでは、検知タグまでスクロールして表示させます。このアクションが特に有効なのは、特殊なページ領域が、ブラウザを操作するユーザーに見えるときだけ、画像を含むフル コンテンツが表示されるサイトです。

プロパティ

「指定タグまでスクロール」アクションは、次の各プロパティを使用して設定できます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームの エレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

ファイル選択

このアクションでは、ファイル フィールドにアップロードするファイルを選択します。

検知タグは、タイプ ファイルの <input> タグになります。

プロパティ

「ファイル選択アクション」は、次の各プロパティを使用して設定できます。

選択するファイル
アップロードするファイルを指定します。ファイルは、以下のいずれかの場所から取得する必要があります。

変数に含まれるファイル
ファイルの内容は変数から読み取ります。

ファイル名
ファイルの名前を指定します。値セレクターを使用して複数の方法で値を指定することができます。

ファイルのコンテンツ タイプ
ファイルのコンテンツ タイプを指定します。指定の方法は以下を参照してください。

ファイル コンテンツ
ファイルのコンテンツを含む変数を選択します。

ローカル ファイル システム内のファイル
ファイルの場所は、ローカルのファイル システム内です。

ファイル名
ファイルの名前を指定します。値セレクターを使用して複数の方法で値を指定することができます。

ファイルのコンテンツ タイプ
ファイルのコンテンツ タイプを指定します。指定の方法は以下を参照してください。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

ファイル コンテンツ タイプの指定

ファイルのコンテンツ タイプは、以下の方法で指定できます。

自動
コンテンツ タイプをコンテンツから自動的に判断します。

変数から
コンテンツ タイプをオブジェクト内の変数で指定します。たとえば、GIF 画像には "image/gif" というテキストが変数に含まれています。

事前定義
コンテンツ タイプをリストから選択します。

カスタム
コンテンツ タイプを直接指定します。たとえば、テキスト コンテンツを "text/plain" と指定します。

複数オプション選択

このアクションでは、リスト ボックスで複数のオプションを選択します。

この検知タグは、"multiple" の属性が有るなど、複数選択が有効な <select> タグである必要があります。

<select> タグに対して登録されたイベント ハンドラーがある場合、オプションを選択すると、JavaScript の実行がトリガーされる可能性があることに注意してください。

ドロップダウン ボックスやリスト ボックスで、オプションを 1 つ選択するには、「オプション選択」アクションを使用します。各オプションをループするには、「セレクト オプション繰り返し」アクションを使用します。

プロパティ

「複数オプション選択」アクションは、次の各プロパティを使用して設定できます。

選択オプション
選択するオプション。すべてのオプションを選択するか、選択するオプションのみを指定します。どのオプションも選択しない場合は、オプションのリストは空欄のままになります。

既存の選択肢を維持
リスト ボックス内の既存の選択肢を維持するかを指定します。これを選択しない場合で、「選択オプション」プロパティでいずれのオプションも選択していない場合、リスト ボックス内のすべての既存の選択肢が選択されず、新たな選択が行われません。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

オプション選択

このアクションでは、ドロップダウン ボックスやリスト ボックスで複数のオプションを選択します。

検索対象のタグはドロップダウン / リスト ボックスの <select> タグになります。選択するオプションに対する <option> タグではありません。

<select> タグに登録済みのイベント ハンドラがある場合は、オプションを設定すると JavaScript を実行させることがあるので、注意してください。

リスト ボックスで一度に複数のオプションを選択するには、複数オプション選択アクションを使用します。各オプションをループするには、「セレクト オプション繰り返し」アクションを使用します。

プロパティ

オプション選択アクションは、次の各プロパティを使用して設定できます。

選択オプション
選択するオプション。これは、値セレクターを使用してさまざまな方法で指定できます。この値は、<select> タグ内の <option> タグに対応する必要があります。また、フォームで表示される値か、またはドロップダウン / リスト ボックス内のオプションで示される値に一致する場合があります。具体的には、照合は以下の方法で実行されます。

- 値が <option> タグの "value" 属性と正確に一致する場合、その最初の <option> タグが選択されます。
- または、先頭や末尾の空白文字にかかわらず、値が <option> タグの "value" 属性と一致する場合、その最初の <option> タグが選択されます。
- または、先頭や末尾の空白文字にかかわらず、値が <option> タグ内のテキストと一致する場合、その最初の <option> タグが選択されます。

大文字と小文字を区別
オンにすると、オプション値の照合の際に大文字と小文字が区別されます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

ラジオ ボタン選択

このアクションでは、1 つのラジオ ボタンを選択することで、同じラジオ ボタン グループ内の他のラジオ ボタンを選択しないことになります。

検知タグは、タイプ ラジオ ボタンの <input> タグになります。

ラジオ ボタン グループ内のラジオ ボタンに登録済みのイベント ハンドラが有る場合は、ラジオ ボタンを選択すると JavaScript を起動させることがあるので、注意してください。

ラジオ ボタン グループ内の各ラジオ ボタンをループするには、「ラジオボタン繰り返し」アクションを使用します。

プロパティ

「ラジオ ボタン選択」アクションは、次の各プロパティを使用して設定できます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームの エレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを ロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

メール送信

「メール送信」アクションではメールを送信します。

メールは Design Studio 内のデザインモードでの実行中には送信されません。

このステップは、SMTP 成功コードが返された場合にのみ正常に実行できます。SMTP エラーが発生すると、ステップの実行が完全にブロックされます。ステップで複数の受信者に電子メールを送信することになっていて、そのうちの 1 人から SMTP エラーが返された場合、ステップは実行されず、どの受信者も電子メールを受け取れません。

プロパティ

「メール送信」アクションは、次の各プロパティを使用して設定できます。

[メッセージ] タブ
このタブに含まれるプロパティで、メッセージの内容、送信者と受信者を指定できます。

送信元アドレス
メールの送信元アドレス (送信者アドレス) を指定します。値セレクターを使用して複数の方法で値を指定することができます。

宛先アドレス
メールの送信先アドレス (受信者アドレス) を指定します。To アドレスは必ず指定します。複数のアドレスを指定する場合はコンマで区切ってください。値セレクターを使用して複数の方法で値を指定することができます。

CC アドレス

メールの CC アドレス (受信者アドレス) を指定します。このアドレスはオプションですが、複数のアドレスを指定する場合はコンマで区切ってください。値セレクターを使用して複数の方法で値を指定することができます。

件名

メールの件名を指定します。値セレクターを使用して複数の方法で値を指定することができます。

メッセージ

メールの本文を指定します。値セレクターを使用して複数の方法で値を指定することができます。

メッセージ タイプ

メッセージ本文のタイプとして、テキストまたは HTML を指定します。

[サーバー] タブ

このタブに含まれるプロパティで、使用するメール サーバーを設定できます。

メール サーバー

メールの送信に使用するメール サーバーを指定します。値セレクターを使用して複数の方法で値を指定することができます。

ポート

メールの送信に使用するメール サーバーのポート番号を指定します。適切なポート番号は、一番多いのが SSL を使用しない場合で 25、SSL を使用する場合で 465 です。値セレクターを使用して複数の方法で値を指定することができます。

ユーザー

メール送信時の認証に使用するユーザー名を指定します。空欄になっていると認証を行いません。値セレクターを使用して複数の方法で値を指定することができます。

パスワード

メール送信時の認証に使用するパスワードを指定します。値セレクターを使用して複数の方法で値を指定することができます。

暗号化

暗号化を使用するかを指定します。

- なし：資格情報および電子メールは、クリア テキストで送信されます。
- TLS: TLS 暗号化が使用されます。現在は TLS バージョン 1.2 が使用されています。
  接続後に、STARTTLS コマンドを使用して通信チャネルを TLS にアップグレードします。これは、SMTP サーバーに信頼できる証明書がある場合のみ機能します。サーバーが自己署名証明書を使用する場合は、keytool ユーティリティを使用してエクスポートしてから JVM のキーストアにインポートする必要があります。
- SMTPS: SMTP over SSLクレデンシャルおよびメールの送信に使用する、セキュリティで保護された通信チャネルを確立します。これは SMTP サーバーがサポートすることは少ないですが、サーバーの証明書が自己署名であっても機能します。

[添付ファイル] タブ

このタブに含まれるプロパティで、メールに添付ファイルを追加できます。

添付ファイルを含める

メッセージに添付ファイルを追加するには、このオプションをオンにします。

内容
含める添付ファイルの内容値セレクターを使用して複数の方法で値を指定することができます。

コンテンツ タイプ
添付ファイルのタイプを指定します。「自動」を選択すると、ファイルの名前の拡張子からタイプを判断します。

ファイル名
添付ファイルのタイプを指定します。指定しない場合は、デフォルトの名前が生成されます。

証明書のエクスポートとインポート

この手順は、SMTP 証明書をエクスポートしてから JVM キーストアにインポートする方法について説明します。これらの手順は、TLS を使用して電子メールの送信ステップを設定し、SMTP サーバーが自己署名証明書を使用する場合にのみ実行してください。詳細については、上記の「暗号化」セクションを参照してください。

1. 次の場所にある既存の証明書ストア **cacerts** をバックアップします。
   `C:\Program Files\Kofax RPA <バージョン番号>\jre\lib\security`
2. STARTTLS コマンドで指定された証明書をダンプします。
   たとえば、OpenSSL ツールを使用して次のコマンドを実行すると、ダンプを実行できます。
   ```
   C:\Program Files\OpenSSL-Win64\bin>openssl s_client -showcerts
   -starttls smtp -crlf -connect smtp.gmail.com:587
   ```
   必要に応じてサーバーとポート番号を変更します。
3. 新しい証明書ファイル (または複数のファイル) を作成します。
   a. ダンプされた証明書 (BEGIN および ENG ヘッダーを含む) の全コンテンツをコピーし、メモ帳を使用して新しいファイルに貼り付けます。
   b. 新しいファイルに .cer 拡張子を選択します。例: ABCD.cer
      複数の証明書を作成する場合は、個別のファイルとして保存しますが、ABCD1.cer、ABCD2.cer などの元の順序を保持してください。
4. .cer ファイルを次の場所に保存します。
   `C:\Program Files\Kofax RPA <バージョン番号>\jre\lib\security`
5. .cer ファイルを Kofax RPA JVM キーストアにインポートします。
   管理者としてコマンド プロンプトを実行し、次のようなコマンドを実行して、ダンプされた証明書を cacerts ファイルにインポートします。
   ```
   C:\Program Files\Kofax RPA <version number> \jre\bin>keytool -import -
   trustcacerts -alias ABCD -file ..\lib\security\ABCD.cer -keystore ..\lib\security
   \cacerts -storepass changeit
   ```
6. プロンプトが表示されたら、**yes** と入力します。
7. 複数の証明書がある場合は、手順 4 から 6 を繰り返します。コマンドの `alias` パラメータを必ず変更してください。
8. Design Studio を再起動して、デバッグ モードで TLS を有効にしてメール送信ステップをテストします。

属性設定

このアクションでは、特定の名前や値によって、検知タグ上で、属性を挿入したり更新したりします。検知タグに、所定の名前が付いた属性が含まれる場合、その値が特定の値で更新されます。それ以外の場合は、指定の値による新しい属性が挿入されます。

このステップは XML 変数に対してのみ機能します。このステップは、XML の正当性の確認もエンティティの解決もしません。

プロパティ

「属性設定」アクションは、次の各プロパティを使用して設定できます。

属性名
設定または更新する属性の名前。

値
属性の値。これは、XML などの所定のタイプのドキュメント内の属性値をエンコードするルールに従って、正確にエンコードされます。

詳細については、「値セレクター」セクションを参照してください。

評価モードの設定

このアクションは、Excel 変数の Excel 値の評価オプションを変更します。

Excel 変数データにのみ使用します。

このステップにより、サポートされていない関数をセルに入力しても、エラーを発生しなくなります。

プロパティ

以下のプロパティを使用して [評価モードの設定] アクションを設定できます。

評価
このプロパティによって、自動評価を有効にするか無効にするかを指定します。
自動評価を有効または無効にするには、`true` または `false` を入力します。
次のいずれかのオプションを選択します。
- 値 - 値を直接設定します。
- 変数 - 値を保存する変数を指定します。
- エクスプレッション - 値を指定するエクスプレッションを作成します。
- コンバータ - 値を設定するデータ コンバータを指定します。

チェックボックス設定

このアクションによって、チェックボックスをオンにしたりオフにしたりします。

検知タグは、タイプ チェックボックスの <input> タグになります。

チェックボックスに登録済みのイベント ハンドラがある場合は、チェックボックスを設定すると JavaScript を実行させることがあるので、注意してください。

プロパティ

「チェックボックス設定」アクションは、次の各プロパティを使用して設定できます。

チェックボックスを以下に設定
たとえば、オンにする、オフにするなど、設定が必要なチェックボックスの状態を指定します。この状態は、値セレクターを使用して指定します。結果の値は、チェックボックスをオンにした場合は "true"、"on"、"1"、または "checked" です。チェックボックスをオフにした場合は "false"、"off"、"0"、または "unchecked" です。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションを口ボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、口ボットの設定のオプションに優先します。その他のすべてのオプションは、口ボットに対して指定したものと同様になります。

列の幅設定

このアクションで、スプレッドシート内の列の幅を設定します。幅の指定には文字を使用します。

プロパティ

「列の幅設定」アクションは、次の各プロパティを使用して設定できます。

幅
このプロパティで、列の幅を指定します。次のオプションがあります。

- 値 - 値を直接設定します
- 変数 - 幅の値を保存する変数を指定します。
- エクスプレッション - 幅を指定するエクスプレッションを作成します。
- コンバータ - 列の幅を設定するデータ コンバータを指定します。

コンテンツ設定

このアクションでは、タグ上の指定したコンテンツを設定します。

このステップは XML 変数に対してのみ機能します。このステップは、XML の正当性の確認も工ンティティの解決もしません。

プロパティ

「コンテンツ設定」アクションは、次の各プロパティを使用して設定できます。

コンテンツ変更モード
このプロパティで、変更するタグを指定します。[既存のタグを設定] および [ルートタグを設定] の各オプションから設定できます。

新しいコンテンツ
挿入する新しいコンテンツです。

セルのコンテンツ設定

このアクションは、Excel と同じように、指定されたコンテンツをスプレッドシートのセルに挿入します。セルは、挿入されたコンテンツのタイプを取得します。つまり、コンテンツが数値の場合は Number タイプ、ブール値が入力された場合は Boolean タイプを取得します。

プロパティ

「セルのコンテンツ設定」アクションは、次の各プロパティを使用して設定できます。

内容
このプロパティで、セルのコンテンツを指定します。次のオプションがあります。
- 値 - 値を直接設定します
- 変数 - 値を保存する変数を指定します。
- エクスプレッション - 値を指定するエクスプレッションを作成します。
- コンバータ - 値を設定するデータ コンバータを指定します。

形式
このプロパティで、セルの形式を指定します。次のオプションがあります。
- 値 - 値を直接設定します
- 変数 - 値を保存する変数を指定します。
- エクスプレッション - 値を指定するエクスプレッションを作成します。
- コンバータ - 値を設定するデータ コンバータを指定します。

Excel のセルで数字の後に D という文字を入力すると、その文字が消えてしまいます。これを回避するには、次のように数字の前に一重引用符を追加します。'12358D。

列のコンテンツ設定

このアクションは、コンプレックス タイプの変数からスプレッドシートの列のコンテンツを設定します。検出された範囲の最初のセルを先頭に、前の属性値の下に新しい変数の属性値が連続的に挿入されます。

プロパティ

以下のプロパティを使用して [列のコンテンツ設定] アクションを設定できます。

変数
このプロパティはセルの値を保存する変数の名前を指定します。

日付書式
挿入されるいずれかのセルの値が日付である場合は、このプロパティを使用してセルの日付書式を指定することができます。次のオプションがあります。
- 値 - 直接書式を設定します。
- 変数 - 書式を保存する変数を指定します。
- エクスプレッション - 書式を指定するエクスプレッションを作成します。
- コンバータ - 書式を設定する データ コンバータ を指定します。

行のコンテンツ設定

このアクションは、コンプレックス タイプの変数からスプレッドシートの行のコンテンツを設定します。

プロパティ

以下のプロパティを使用して [行のコンテンツ設定] アクションを設定できます。

変数
このプロパティは行の値を保存する変数の名前を指定します。

日付書式
挿入されるいずれかのセルの値が日付である場合は、このプロパティを使用してセルの日付書式を指定することができます。次のオプションがあります。
- 値 - 直接書式を設定します。
- 変数 - 書式を保存する変数を指定します。
- エクスプレッション - 書式を指定するエクスプレッションを作成します。
- コンバータ - 書式を設定する データ コンバータ を指定します。

カレント ウィンドウ設定

このアクションは、別のウィンドウをカレント ウィンドウ、つまり後続のステップが動作するウィンドウとして選択します。

Design Studio で [カレント ウィンドウ設定] ステップを簡単に挿入するには、ウィンドウのタブを右クリックして [現在のウィンドウとして設定] を選択します。

プロパティ

[カレント ウィンドウ設定] アクションは、[ウィンドウ設定] プロパティを使用して設定することができます。

ウィンドウまたはフレーム
このプロパティを使用して、現在のウィンドウまたはフレームとして選択する必要があるウィンドウまたはフレームを定義します (ウィンドウの識別方法については、ウィンドウの識別を参照してください)。
以下のオプションを使用して、現在のウィンドウを設定します。
- ウィンドウ名**:** ウィンドウのタブに表示されるウィンドウの名前でウィンドウまたはフレームを検索します。
- ウィンドウ **ID** がパターンと一致する範囲: 指定したパターンに一致する ID でウィンドウまたはフレームを検索します。

- **HTML** がパターンに一致している場所: 指定したパターンに一致する HTML コードでウィンドウまたはフレームを検索します。

タグ ファインダーで見つかったフレーム
タグ ファインダーを用いて、特定のフレームを現在のフレームとして定義するには、このプロパティを使用します。詳細については、「タグ ファインダーの使用」セクションを参照してください。

セルのフォーマット設定

このアクションはスプレッドシートの 1 つ以上のセルの書式を設定します。

プロパティ

以下のプロパティを使用して [セルのフォーマット設定] アクションを設定できます。

形式
挿入するデータの書式。次のオプションがあります。
- 値 - 直接書式を設定します。
- 変数 - 書式を保存する変数を指定します。
- エクスプレッション - 書式を指定するエクスプレッションを作成します。
- コンバータ - 書式を設定する データ コンバータ を指定します。

セルのハイパーリンク設定

このアクションはセルにハイパーリンクを挿入します。

プロパティ

以下のプロパティを使用して [セルのハイパーリンク設定] アクションを設定できます。

URL
このプロパティはリンクのアドレスを指定します。次のオプションがあります。
- 値 - リンク アドレスを直接設定します。
- 変数 - リンク アドレスを保存する変数を指定します。
- エクスプレッション - リンク アドレスを指定するエクスプレッションを作成します。
- コンバータ - リンク アドレスを設定する データ コンバータ を指定します。

デフォルトのリンク スタイルを使用
このプロパティは、リンクのデフォルト スタイル、つまりテキストに下線を引き、文字色を青にするスタイルをセルに設定するかどうかを指定します。

プロパティ情報設定

このアクションはスプレッドシートのプロパティ情報の値を設定します。

プロパティ

以下のプロパティを使用して [プロパティ情報設定] アクションを設定できます。

名前
このプロパティはプロパティ情報の名前を指定します。

値
このプロパティは値のコンテンツを指定します。次のオプションがあります。
- 値 - 値を直接設定します
- 変数 - 値を保存する変数を指定します。
- エクスプレッション - 値を指定するエクスプレッションを作成します。
- コンバータ - 値を設定するデータ コンバータを指定します。

JSON 設定

このステップは JSON 値の検出された部分全体を新しい JSON 値に置き換えます。新しい値は、オブジェクト { "answer": 42 } や引用文字列 "Life, the Universe and Everything" のような有効な JSON 値である必要があります。

このステップは JSON 変数に対してのみ機能します。

プロパティ

以下のプロパティを使用して [JSON 設定] アクションを設定できます。

新しいコンテンツ
新規の JSON 値。
この値は、値セレクターの拡張バージョンを使用してさまざまな方法で指定できます。値セレクターには、値を指定するための通常の 4 つの方法と、非常に有用な追加のセレクター「変数から生成」が含まれています。このセレクターはコンプレックス タイプの変数から JSON を自動的に生成します。たとえば、名前と年齢という 2 つの属性を持つタイプの変数があり、これらの変数の値が "Joe" と 23 である場合、生成される JSON は{ "name" : "Joe", "age" : 23 } のようになります。

名前付き JSON 設定

このアクションを実行すると、検出された JSON が名前付き JSON としてマークされるため、後続のステップで JSON 値の他の部分を見つける場合の参照として使用できます。

このアクションは、複数のステップがこの JSON と関連する値を操作する必要がある場合に便利です。

プロパティ

以下のプロパティを使用して [名前付き JSON 設定] アクションを設定できます。

名前
2 つのオプションがあります。「自動」または「名前付き」。
自動: 項目名に番号が割り振られます。自動で割り振られた最初の項目名は 1 になり、次に 2、3 と続きます。

名前付き: 固定的な、明確に記述された名前を項目に与えます。
詳細については、名前付きタグ、範囲、JSON を参照してください。

既存の名前付きアイテムを保持
オンにすると、既存の名前付き項目が保持されます。オフの場合、これらは名前付き項目としてマークされずに、このステップの後で見つかった項目だけが名前付き項目となります。

名前付き範囲設定

このアクションは、見つかった範囲を名前付き範囲としてマークし、ます。これにより、後続のステップで範囲を検索するときに参照として使用できるようになります。

このアクションは、複数のステップがこの範囲と関連する範囲を操作する必要がある場合に便利です。

プロパティ

以下のプロパティを使用して [名前付き範囲設定] アクションを設定できます。

範囲名
「自動」、「名前付き」という 2 つのオプションがあります。「自動」は範囲名として番号を与えます。「自動」で最初に与えられる番号は 1、次に与えられる番号は 2 となります (以下同様)。「自動」で番号が与えられる他の範囲が (同じページの) このステップの前に挿入されると、番号が変わることがあります。名前付けにより、範囲に対して明示的に指定された固定の名前が付けられますが、これには次のような利点があります。

- 適切に選んだ名前を付けると、その範囲がどのような範囲であるかを容易に思い出せます。
- 明確に名前付けされた範囲は、名前付きの別の範囲がその前に挿入されても影響を受けません。
- 「名前付き範囲設定」で同じ名前を使用すると、その名前は、新しい範囲を参照するように設定されます (ステートフルなページ内ループに便利です)

既存の名前付き範囲を保持
このプロパティがオンになっていると、既存の名前付き範囲が保持されます。このプロパティがオフになっていると、既存の名前付き範囲については名前付き範囲としてのマークが解除され、検出された範囲のみがこのステップ以降の名前付き範囲になります。

名前付きタグ設定

このアクションは、見つかったタグを名前付きタグとしてマークし、後続のステップでタグを検索するときに参照として使用できるようにします。

このアクションは、複数のステップがこのタグと関連するタグを操作する必要がある場合に便利です。

プロパティ

以下のプロパティを使用して [名前付きタグ設定] アクションを設定できます。

タグの名前
「自動」、「名前付き」という 2 つのオプションがあります。「自動」を選択すると、タグ名に番号が割り振られます。自動で割り振られた最初のタグ名は 1 になり、次に 2、3 と続きます。自動で番号を割り振られた他のタグが (同じページの) このステップの前に挿入されると、番号が変更されることがあります。「名前付き」を選択すると、タグに対して明示的に指定した固定の名前を付けることができますが、これには次のような利点があります。

- 適切に選んだ名前を付けると、そのタグがどのようなタグであるかを容易に思い出せます。
- 明確に名前付きタグは、名前付きの別のタグがその前に挿入されても影響を受けません。

- 名前付きタグ設定 においてすでに使われているのと同じ名前を使用する場合、その名前は単純に新しいタグを参照することになります (ステートフルなページ内ループに便利です)。

既存の名前付きタグを保持
このプロパティがオンになっていると、既存の名前付きタグが保持されます。このプロパティがオフになっていると、既存の名前付きタグについては名前付きタグとしてのマークが解除され、検出されたタグのみがこのステップ以降の名前付きタグになります。

プロパティ名設定

このアクションは検出された JSON オブジェクトのプロパティ名を新しい名前に置き換えます。プロパティの値は変更されません。

このステップは JSON 変数に対してのみ機能します。

プロパティ

以下のプロパティを使用して [プロパティ名設定] アクションを設定できます。

名前
プロパティの新しい名前。これは有効なプロパティ名である必要があり、引用符などはエスケープする必要があります。

範囲値の設定

このアクションは、下回ることのできない下限の数値と、超えることのできない上限の数値を設定します。

プロパティ

以下のプロパティを使用して [名前付き範囲設定] アクションを設定できます。

範囲コントロールの値の指定先
コントロールの値を指定します。たとえば、Web ページの音量制御スライダーに適用できます。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

行の高さ設定

このアクションは行の高さをポイントで設定します。

プロパティ

以下のプロパティを使用して [行判定] アクションを設定できます。

高さ
このプロパティによって、値セレクター の拡張バージョンを使用して、行の高さを指定します。次のオプションがあります。
- 値 - 行の高さを直接設定します。
- 変数 - 行の高さを保存する変数を指定します。
- エクスプレッション - 行の高さを指定するエクスプレッションを作成します。
- コンバータ - 行の高さを設定する データ コンバータ を指定します。

シート名設定

このアクションはシート名を設定します。

プロパティ

「コンテンツ設定」アクションは、次の各プロパティを使用して設定できます。

名前
このプロパティはシートの名前を指定します。次のオプションがあります。
- 値 - 名前を直接設定します。
- 変数 - 名前を保存する変数を指定します。
- エクスプレッション - 名前を指定するエクスプレッションを作成します。
- コンバータ - 名前を設定する データ コンバータ を指定します。

タグ設定

このステップは検出されたタグ全体を新しいコンテンツに置き換えます。このステップは [コンテンツ設定] ステップと似ていますが、後者はタグのコンテンツのみを置き換えるのに対して、[タグ設定] ステップではタグ自体を置き換えます。

このステップは XML 変数に対してのみ機能します。このステップは、XML の正当性の確認もエンティティの解決もしません。

プロパティ

以下のプロパティを使用して [タグ設定] アクションを設定できます。

新しいコンテンツ
タグ全体の新しいコンテンツ。

名前付きタグ設定

このアクションは検出されたタグの名前を新しい名前に置き換え、オプションで、検出されたタグの属性を新しいタグにコピーします。タグの子ノードは保持されます。

このステップは XML 変数に対してのみ機能します。このステップは、XML の正当性の確認もエンティティの解決もしません。

プロパティ

以下のプロパティを使用して [名前付きタグ設定] アクションを設定できます。

タグの名前
タグの新しい名前。これは、当然、指定されたドキュメント タイプの有効なタグ名である必要があります。

属性を消去
このプロパティがオンになっていると、検出されたタグのすべての属性が除去されます。オフになっていると、属性が新しいタグにコピーされます。

テキスト設定

このステップは検出されたタグのコンテンツをテキストに置き換えます。このステップは [コンテンツ設定] ステップと似ていますが、後者は指定されたテキストをマークアップとして挿入するのに対して、[テキスト設定] ステップはテキストをアンパサンド エンコードされたテキストまたは CDATA として挿入します。

このステップは XML 変数に対してのみ機能します。このステップは、XML の正当性の確認もエンティティの解決もしません。

プロパティ

以下のプロパティを使用して [テキスト設定] アクションを設定できます。

テキスト
挿入するテキスト。この値は、値セレクターの拡張バージョンを使用していくつかの方法で指定できます。

エンコーディング
テキストをエンコードする方法。2 つの選択肢があります。
- 「アンパサンド」は挿入するときにテキストをアンパサンド エンコードするよう指定します。たとえば、< は < になります。
- "CDATA" はテキストを CDATA セクションに挿入するよう指定します。

セルの値設定

このアクションは、セルのタイプが事前設定されている場合に、セルの値を設定します。

プロパティ

以下のプロパティを使用して [セルの値設定] アクションを設定できます。

タイプ
このプロパティは値のタイプを指定します。次のオプションがあります。
- テキスト
- 数値
- 論理
- 日付
- 式

値
このプロパティによって、値セレクター の拡張バージョンを使用して値の内容を指定します。次のオプションがあります。
- 値 - 値を直接設定します
- 変数 - 値を保存する変数を指定します。
- エクスプレッション - 値を指定するエクスプレッションを作成します。
- コンバータ - 値を設定するデータ コンバータを指定します。

形式
このプロパティによって、値セレクター の拡張バージョンを使用してセルのフォーマットを指定します。次のオプションがあります。
- 値 - 書式を直接設定します。
- 変数 - 書式を保存する変数を指定します。
- エクスプレッション - 書式を指定するエクスプレッションを作成します。
- コンバータ - 書式を設定する データ コンバータ を指定します。

スニペット ステップ

スニペット ステップはスニペットを ロボットに挿入するために使用されます。スニペットは、1 つの ロボットまたは複数の ロボット全体で複数回再利用できる、ロボットのサブパートです。スニペットは、複数の結合されたステップおよび変数のセットという 2 つの部分から構成されます。スニペット ステップを介してスニペットが ロボットに挿入されると、スニペットのステップがスニペット ステップの場所に挿入され、スニペットの変数が ロボットの変数に追加されます。

スニペット ステップは グループ ステップ と似ています。グループ ステップと同様に、スニペット ステップの左上隅には展開/縮小 (+/-) アイコンがあります。このアイコンをクリックするとスニペット ステップが展開/縮小され、グループ ステップと同様に、挿入されるステップは 1 つの入口ポイントと 1 つの出口ポイントを持ちます。ただし、グループ ステップと異なり、スニペット ステップ内部のステップを変更すると、対応するスニペットが変更され、そのスニペットを参照するその他のすべてのスニペット ステップによってスニペット ステップに含まれるステップが変更されます。

プロパティ

スニペット ステップは、[基本] タブと [スニペット] タブの次のプロパティを使用して設定します。

ステップ名
これはステップの名前です。ここにステップ名を入力すると、ステップの下の ロボット ビューに名前が表示されます。ステップ名が入力されていない場合は、代わりにスニペットの名前が表示されます。この場合、名前に下線が付きます。

ステップのコメント
これはスニペット ステップを説明するコメントです。これはスニペット自体を説明するコメントではありません。スニペットのコメントは、スニペットがエディターで開かれているときにスニペット上で直接編集する必要があります。

スニペット
これはスニペットの名前です。スニペット ステップが含まれている ロボットと同じプロジェクト内部のその名前のスニペットを参照します。別の名前を選択すると、新しいスニペットがスニペット ステップの場所に挿入されます。スニペットにコメントがある場合は、ここに表示されます。

[開く] をクリックすると、ステップが参照するスニペットを開くことができます。スニペットが新しいタブで開きます。

XML データ マッパーを開く(X)
[XML データ マッパー] ダイアログを開きます。詳細については、「XML データ マッパー」セクションを参照してください。

終了

このアクションはエラーを生成せずにロボットの実行を終了させます。つまり、ロボット内の残りのイテレーションや分岐、ステップはどれも実行されません。

データベース データ登録

このステップは変数をデータベース データ登録する操作を行います。Design Studio の [設定] でデータベース接続を設定する必要があります。

このステップをロボットで使用する前に、必ずデータベースへのデータの保存のセクションをお読みください。

⚠ 「データベース データ登録」または「SQL 実行」ステップで使用されるデータベースのテーブルを削除したり変更したりしないでください。

プロパティ

以下のプロパティを使用して [データベース データ登録] を設定できます。

データベース
どのデータベースに変数を保存するかをコントロールします。値については、選択することやデザイン時にハードコーディングすることができます。あるいは、変数、式またはコンバータを使用して、ランタイム時に動的にデータベース名を作成することもできます。ロボットの実行時にその名前のデータベースが存在しない場合は、エラーが発生します。

変数
保存する変数を選択します。この変数はシンプル タイプの変数ではなく、正規変数である必要があります。

キー
保存される変数の一意のキー。キーは (「データベース キーの一部」として変数をマークすることによって) タイプで定義することも、変数、式またはコンバータを使用して定義することもできます。指定されたキーを持つ値がすでに存在する場合、このステップは既存のレコードを上書き (SQL 更新) します。このキーを持つ値が存在しない場合は、新しいレコードが挿入されます。

デザイン モードで実行
これが有効になっていると、ステップは Design Studio 内部のデザイン モードでも実行されます。これが無効になっていると、ロボットをデザイン モードでナビゲートするときにステップは何も実行しません。

セル タイプ判定

このアクションは 1 つまたは複数のセルのタイプを判定します。このタイプは次の Excel のデータ タイプです: テキスト、数値、論理値 (ブール値)、空白、エラー、数式。これらは、基本的に Excel 関数の ISTEXT、ISNUMBER などに相当します。範囲ファインダーによって複数のセルが選択されている場合は、検出されたすべてのセルが判定の条件を満たしている必要があります。

プロパティ

以下のプロパティを使用して [セル タイプ判定] アクションを設定できます。

条件

これには以下を満たすような条件が含まれます。

Is Blank

これは検出されたセルが空白の場合にのみ true になります。これは Excel 関数の ISBLANK に相当します。

Is Text

これは検出されたすべてのセルにテキストが含まれている場合にのみ true になります。これは Excel 関数の ISTEXT に相当します。

Is Number

これは検出されたすべてのセルに数値が含まれている場合にのみ true になります。これは Excel 関数の ISNUMBER に相当します。

Is Logical

これは検出されたすべてのセルにブール値が含まれている場合にのみ true になります。これは Excel 関数の ISLOGICAL に相当します。

Is Error

これは検出されたすべてのセルにエラーが含まれている場合にのみ true になります。これは Excel 関数の ISERROR に相当します。

Is Formula

これは検出されたすべてのセルに数式が含まれている場合にのみ true になります。これは Excel 関数の ISFORMULA に相当します。

If

判定の基準となる厳密な条件を指定します。また、指定された条件の反転も可能です。デフォルトでは「条件が満たされていないか」を判定します。この条件が選択されており、条件の評価が false である場合、実行は Do プロパティによる指定の影響を受けます。しかし、条件の評価が true である場合、実行は現在の分岐以降も継続します。この条件の代わりに、反転した結果が得られる「条件が満たされているか」を判定することもできます。

Do

条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り

[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

ファイル有無判定

このアクションは特定のファイルの有無を判定して、現在の分岐以降も実行を続行するか、他のことを実行する必要があるかを決定します。

プロパティ

以下のプロパティを使用して [ファイル有無判定] アクションを設定できます。

ファイル名
探索するファイルの名前。名前は、値セレクターを使用してさまざまな方法で指定できます。名前は絶対パス名である必要があります。該当する場合はドライブ名とファイルへのディレクトリ パスを含めます。

If
判定の基準となる厳密な条件を指定します。条件は「ファイルが存在する」または「ファイルが存在しない」のどちらかです。この条件が満たされている場合、実行は Do プロパティによる指定の影響を受けます。条件が満たされていない場合、実行は現在の分岐以降も継続します。

Do
条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り
[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

JSON タイプ判定

このアクションは JSON 値のタイプを判定します。次のタイプがあります: オブジェクト、配列、文字列、整数、浮動小数、ブール値、Null。これらのタイプの他に、2 つ以上の一般的なタイプをテストすることもできます: シンプルおよび数値

プロパティ

以下のプロパティを使用して [JSON タイプ判定] アクションを設定できます。

条件
これには以下を満たすような条件が含まれます。

Is Object
これは検出された JSON 値がオブジェクトである場合にのみ true になります。これは {...} 形式の JSON 値です。これは JavaScript のオブジェクトと見なされるため、配列のオブジェクトを返す可能性もある JavaScript タイプの操作とは異なります。

Is Array
これは検出された JSON 値が配列である場合にのみ true になります。これは [...] 形式の JSON 値です。

Is Simple
これは検出された JSON 値がシンプル タイプである場合にのみ true になります。シンプル タイプとはオブジェクトでも配列でもない任意の JSON 値です。

Is Integer
これは検出された JSON 値が整数である場合にのみ true になります。このタイプは任意精度整数ですが、JSON 整数値を他の整数の表現に変換するときは常に注意する必要があります。たとえば、整数変数では、変換の結果オーバーフローが起きる可能性があります。

Is Float
これは検出された JSON 値が浮動小数である場合にのみ true になります。このタイプは、23.345E-42 などの非整数の任意精度数値ですが、JSON 浮動小数値を他の何らかの数値表現に変換するときは常に注意する必要があります。たとえば、数値変数では、変換の結果オーバーフローが起きる可能性があります。

Is Number
これは検出された JSON 値が整数または浮動小数である場合にのみ true になります。

Is String
これは検出された JSON 値が文字列である場合にのみ true になります。文字列の先頭と末尾は引用符 (") でなければならず、さまざまな文字は「、\、/、b、f、n、r、t、uXXXX などのバックスラッシュ (\) で エスケープする必要があります。

Is Boolean
これは検出された JSON 値がブール値である場合にのみ true になります。ブール値は 2 つの値 true または false のいずれかです。

Is Null
これは検出された JSON 値が null 値である場合にのみ true になります。

これらの条件は相互排他的ではありません。たとえば整数値では "Is Number" と "Is Integer" が true です。

If
判定の基準となる厳密な条件を指定します。また、指定された条件の反転も可能です。デフォルトでは「条件が満たされていないか」を判定します。この条件が選択されており、条件の評価が false である場合、実行は Do プロパティによる指定の影響を受けます。しかし、条件の評価が true である場合、実行は現在の分岐以降も継続します。この条件の代わりに、反転した結果が得られる「条件が満たされているか」を判定することもできます。

Do
条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り

[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

ページ タイプ判定

このアクションは現在のウィンドウ内のページのタイプを判定します。現在次の 5 つのページ タイプがあります: HTML、XML、JSON、Excel、バイナリ。

プロパティ

以下のプロパティを使用して [ページ タイプ判定] アクションを設定できます。

ページ タイプ
これには、テストを行う次のようなページ タイプが含まれます。

HTML
これは現在のウィンドウ内のページが HTML ページである場合にのみ true になります。これは、必ずしもソースが HTML であったことを意味するものではなく、ページが読み込まれた後、HTML ページに渡され、または変換され、ページ ビューなどで HTML ページとして提示されたことを意味しているにすぎません。

XML
これは現在のウィンドウ内のページが XML ページである場合にのみ true になります。

JSON
これは、現在のウィンドウ内のページが JSON ページである場合にのみ true になります。

Excel
これは現在のウィンドウ内のページがスプレッドシート ドキュメントである場合にのみ true になります。

バイナリ
これは現在のウィンドウ内のページがバイナリ ページである場合にのみ true になります。

If
判定の基準となる厳密な条件を指定します。また、指定された条件の反転も可能です。デフォルトでは「条件が満たされていないか」を判定します。この条件が選択されており、条件の評価が false である場合、実行は Do プロパティによる指定の影響を受けます。しかし、条件の評価が true である場合、実行は現在の分岐以降も継続します。この条件の代わりに、反転した結果が得られる「条件が満たされているか」を判定することもできます。

Do
条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り
[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

行判定

[行判定] アクションはテーブル行内の列の数を判定します。

検出されたタグは <tr> タグである必要があります。[列の最小数] と [列の最大数] によって定義される区間を指定し、列の数が指定された区間外 (または区間内) である場合に何を実行するかを選択します。

プロパティ

以下のプロパティを使用して [行判定] アクションを設定できます。

列の最小数
テーブル行内の列の最小数を入力します。

列の最大数
テーブル行内の列の最大数を入力します。

If
判定の基準となる厳密な条件を指定します。条件は区間と「行が一致する」または「行が一致しない」のどちらかです。行内の列の数が指定された区間に収まっていれば、行は区間と一致します。指定された条件が満たされている場合、実行は Do プロパティによる指定の影響を受けます。条件が満たされていない場合、実行は現在の分岐以降も継続します。

Do
条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り
[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

タグ判定

[タグ判定] アクションは判定を行い、現在の分岐以降も実行を続行してよいか、他のことを実行する必要があるかを決定します。判定では検出されたタグを パターン と照合します。

[タグ判定] アクションの使用は、ロボットの条件付き実行を検出するうえで最も一般的な方法です。

プロパティ

以下のプロパティを使用して [タグ判定] アクションを設定できます。

パターン
検出されたタグと照合するパターン。パターンは検出されたタグ全体と照合する必要があります。

照合対象
検出されたタグのどの部分とパターンを照合するかを指定します。

- テキストのみは、検出されたタグ内のテキストのみとパターンを照合することを指定します。
- **HTML** は、検出されたタグの HTML とパターンを照合することを指定します。

大文字と小文字を無視
このオプションをオンにすると、パターンの照合で大文字と小文字の区別が無視されます。

If
判定の基準となる厳密な条件を指定します。条件は「検出されたタグとパターンが一致する」または「検出されたタグとパターンが一致しない」のどちらかです。この条件が満たされている場合、実行は Do プロパティによる指定の影響を受けます。条件が満たされていない場合、実行は現在の分岐以降も継続します。

Do
条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り
[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

コンバータ
これらの データ コンバータ (選択されている場合) は、パターン照合を行う前に、検出されたタグのテキストまたは HTML に適用されます。データ コンバータから何も出力されない場合は、エラーが生成されます。

URL 判定

このアクションは検出されたタグに含まれた URL を判定して、現在の分岐以降も実行を続行するか、他のことを実行する必要があるかを決定します。

URL は以下のいずれかのタイプのタグに該当する必要があります。

- <a href="URL">
- <area href="URL">
- <frame src="URL">
- <iframe src="URL">
- <script src="URL">
- <param value="URL">
- <meta http-equiv="Refresh" content="...; url=URL">

プロパティ

以下のプロパティを使用して [URL 判定] アクションを設定できます。

URL パターン
検出されたタグ内の URL が一致する必要があるパターン。パターンは URL 全体と照合する必要があります。

大文字と小文字の区別を無視
このプロパティをオンにした場合、パターンの照合では大文字と小文字の区別が無視されます。

If
判定の基準となる厳密な条件を指定します。条件は「パターンが URL と一致する」または「パターンが URL と一致しない」のどちらかです。指定された条件が満たされている場合、実行は Do プロパティによる指定の影響を受けます。条件が満たされていない場合、実行は現在の分岐以降も継続します。

Do
条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り
[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

値判定

このアクションはブール値を判定して、現在の分岐以降も実行を続行するか、他のことを実行する必要があるかを決定します。このアクションは、グローバル変数の値をチェックする必要があるときに役立つ機能の 1 つです。たとえば、このアクションを使用してカウンターが指定された値を超えたかどうかをチェックすることができます。

プロパティ

以下のプロパティを使用して [値判定] アクションを設定します。

条件
条件が含まれます。条件は true または false のどちらかを評価する必要があります。その他のすべての値は、アクションが実行されたときにエラーを生成します。条件の指定は、デフォルトではエクスプレッションを入力して行います。ただし、値セレクター の内のオプションを使用することもできます。

If
判定の基準となる厳密な条件を指定します。また、指定された条件の反転も可能です。デフォルトでは「条件が満たされていないか」を判定します。この条件が選択されており、条件の評価が false である場合、実行は Do プロパティによる指定の影響を受けます。しかし、条件の評価が true である場合、実行は現在の分岐以降も継続します。この条件の代わりに、反転した結果が得られる「条件が満たされているか」を判定することもできます。

Do
条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り
[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

例
以下の例では、式を使用して条件を指定しています。これが条件を指定するデフォルトの方法です。
指定された長さを持つテキストであるかどうかを判定します。

```
ScratchPad.shortText1.length() == 28
```

複数の値を一度に判定します。

```
PersonInput.isMale && PersonInput.isMarried
```

変数判定

このアクションは、1 つ以上の変数を判定して、現在の分岐以降も実行を続行するか、他のことを実行する必要があるかを決定します。このアクションは、抽出された変数の値が有効かどうかをチェックするのに役立ちます。たとえば、このアクションを使用して、抽出された変数が入力変数の値と一致するかどうかをチェックすることができます。

判定を指定するには、1 つ以上の変数条件を追加します。個々の変数条件は選択されている変数の値を別の値と比較します。比較の結果および if プロパティの選択肢に応じて、実行は現在の分岐以降も継続するか、Do プロパティの指示の影響を受けます。

プロパティ

以下のプロパティを使用して [変数判定] アクションを設定できます。

条件
変数条件のリストが含まれます。変数条件を設定する方法の詳細については、以下の説明を参照してください。

If
変数条件が判定されたときに何を実行するかを決定します。

どの条件も満たされていない
1 つ以上の変数条件が満たされていれば、実行は現在の分岐以降も継続します。どの条件も満たされていなければ、実行は Do プロパティの指示の影響を受けます。

いずれかの条件が満たされていない
すべての変数条件が満たされている場合にのみ、実行は現在の分岐以降も継続します。1 つ以上の条件が満たされていなければ、実行は Do プロパティの指示の影響を受けます。

いずれかの条件が満たされている
どの変数条件も満たされていなければ、実行は現在の分岐以降も継続します。1 つ以上の変数条件が満たされていれば、実行は Do プロパティの指示の影響を受けます。

すべての条件が満たされている
1 つ以上の変数条件が満たされていなければ、実行は現在の分岐以降も継続します。すべての変数条件が満たされていれば、実行は Do プロパティの指示の影響を受けます。

Do
条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り
[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

変数条件を設定する

変数条件は選択されている変数の値を別の値と比較します。以下のプロパティを利用して変数条件を設定します。

[基本] タブ

変数
値が判定の対象となる変数 (または変数属性)。

演算子
変数の値を他の値と比較するときに使用する演算子。利用可能な演算子については、以下の説明を参照してください。

値
変数値を比較する対象となる値。値セレクターを使用してさまざまな方法で値を指定することができます。

大文字と小文字の区別を無視
このオプションが選択されている場合、大文字と小文字の区別を無視して比較が実行されます。

[詳細] タブ

変数が空の場合に必ず正とする
このオプションが選択されていて、選択されている変数が値を持っていなければ、比較の結果に関係なく、変数条件は必ず満たされます。

値が空の場合に必ず正とする
このオプションが選択されていて、選択対象の値が空であれば、比較の結果に関係なく、変数条件は必ず満たされます。このオプションは、抽出された変数の値を入力変数の値と比較するときに便利です。入力変数が特定の属性の値を持っていない場合、抽出された変数の属性値が入力変数の属性値と一致するかどうかの判定は通常、省略されます。このオプションを選択すると、この判定が省略されます。

演算子プロパティでは以下の演算子が利用できます。

演算子	説明
=	2 つの値が等しいかどうかを判定します。
<>	2 つの値が等しくないかどうかを判定します。
<	最初の値が 2 番目の値未満かどうかを判定します。
<=	最初の値が 2 番目の値以下かどうかを判定します。
>=	最初の値が 2 番目の値以上かどうかを判定します。
>	最初の値が 2 番目の値より大きいかどうかを判定します。

演算子	説明
contains	最初の値に 2 番目の値の 1 回以上の出現が含まれているかどうかを判定します。判定は値のテキスト表現に対して行われます。 最初の値が空である場合、判定の条件が満たされることはありません。また、最初の値が空でなく、2 番目の値が空である場合、判定の条件は必ず満たされます。
does not contain	最初の値に 2 番目の値の出現が含まれていないかどうかを判定します。判定は値のテキスト表現に対して行われます。注意：最初の値が空である場合、判定の条件は必ず満たされます。また、最初の値が空でなく、2 番目の値が空である場合、判定の条件が満たされることはありません。
is contained in	最初の値が 2 番目の値に 1 回以上出現するかどうかを判定します。判定は値のテキスト表現に対して行われます。注意：2 番目の値が空である場合、判定の条件が満たされることはありません。また、2 番目の値が空でなく、最初の値が空である場合、判定の条件は必ず満たされます。
starts with	最初の値が 2 番目の値から始まるかどうかを判定します。判定は値のテキスト表現に対して行われます。注意：最初の値が空である場合、判定の条件が満たされることはありません。また、最初の値が空でなく、2 番目の値が空である場合、判定の条件は必ず満たされます。
ends with	最初の値が 2 番目の値で終了するかどうかを判定します。判定は値のテキスト表現に対して行われます。注意：最初の値が空である場合、判定の条件が満たされることはありません。また、最初の値が空でなく、2 番目の値が空である場合、判定の条件は必ず満たされます。

演算子 '<>'、'<'、'<='、'>='、'>' の正確な意味は、選択されている変数/属性のタイプによって異なります。たとえば、タイプがショート テキストまたはロング テキストの場合、比較は辞書学的に行われるのに対して、タイプが数値または整数の場合、比較は数値的に行われます。

ウィンドウ判定

このアクションは、現在の分岐以降も実行を続行するか、他のことを実行する必要があるかを決定するために、特定のウィンドウの有無を判定します。

ウィンドウを選択できるようにするには、[ウィンドウ判定] アクションを設定するときにウィンドウが存在している必要があります。

プロパティ

以下のプロパティを使用して [ウィンドウ判定] アクションを設定できます。

ウィンドウ
有無を確認するウィンドウ (ウィンドウを特定する方法を参照してください)。

If
判定の基準となる厳密な条件を指定します。条件は「ウィンドウが存在する」または「ウィンドウが存在しない」のどちらかです。この条件が満たされている場合、実行は Do プロパティによる指定の影響を受けます。条件が満たされていない場合、実行は現在の分岐以降も継続します。

Do
条件と if プロパティの組み合わせによって現在の分岐以降の実行を続行しないと判定された場合に、何を実行するかを決定します。

エラー処理の指定通り
[エラー処理] タブで、実行される内容を詳細に指定します。

後続のステップ全てをスキップ
現在の分岐以降の実行を停止します。

XML の変換

XML の変換アクションでは、XSLT スタイルシートを使用して、XML 変数に含まれている XML ドキュメントを変換します。このスタイルシートは、アクションの一部として指定されます。変換結果は、XML、HTML、または Long Text タイプの変数に格納されます。

XML の変換ステップは、XSLT バージョン 1.0 をサポートしています。

スタイルシートが生成する出力は、選択した出力変数において格納できるタイプである必要があります。つまり、出力値が XML 変数に格納される場合は、そのスタイルシートでは <xsl:output method="xml"> を指定する必要があります。出力値が HTML 変数に格納される場合は、そのスタイルシートでは <xsl:output method="html"> を指定する必要があります。出力値をテキスト変数に格納する場合、出力メソッドは XML、HTML として格納され、テキストはすべてテキストとして格納されます。

一般的なユースケースは、Web サイトから XML を XML 変数に格納するために ターゲット抽出アクションを使用してから、XML の変換アクションを使用してデータを変換し、同じ XML 変数に格納する場合です。最後に、ページ生成アクションは、XML 変数から変換した HTML 変数を選ぶことで XML ドキュメントを表示するページの作成に使用できます。これにより、標準の抽出アクションを使用しての、変換したドキュメントからのデータの抽出が容易になります。

プロパティ

XML の変換アクションは、以下のプロパティを使用して設定されます。

入力変数
変換に対する入力を含む XML 変数を選択します。HTML または Long Text 変数を選択できますが、有効な XML が含まれている必要があります。

XSLT スタイルシート
変換に使用する XSLT スタイルシートを指定します。多くの場合、このスタイルシートは固定の XML として使用されます。ただし、[エクスプレッションから XML] または [変数から XML] を選択することでスタイルシートを動的に作成することができます。

出力変数
変換結果が格納される必要のある変数を選択します。XML、HTML、Long Text のタイプの変数が許可されています。XSLT スタイルシートは、選択した変数で格納できる出力値を作成する必要があります。結果は、XML 入力として選択されたものと同じ変数に格納できます。

テーブル行列入れ替え

テーブル行列入れ替えアクションは、テーブルを入れ替えます。テーブルを入れ替えるということは、行間および列間の順番を維持しながら、行を列に変え、列を行に変えることを意味します。

このアクションにはプロパティがありません。

例
たとえば、3 つの行と 4 つの列のある (ヘッダーを含む) 以下のテーブルを見てください。

名前	John	Michael	Ross
身長	1.80	1.56	2.09
体重	75	93	64

このテーブルを入れ替えると、以下の 4 つの行と 3 つの列のある (ヘッダーを含む) 以下のテーブルが生成されます。

名前	身長	体重
John	1.80	75
Michael	1.56	93
Ross	2.09	64

テーブルを 2 回入れ替えても、元のテーブルに戻るとは限りません。

トライ

トライ ステップは、特定の作業を完了させるために複数の代替的なアプローチを試行する必要のある場合に使用します。

トライ ステップは、複数の分岐につながるため、分岐ポイントに似ています。ただし、分岐ポイントと異なる理由は、先行する分岐がエラー処理オプションの次のトライステップへ移動 (または、レガシ ロボットでは「後方に送る」) を有効化する場合にのみ、最初のものを越えて分岐が実行されるためです。

タグ表示化

このアクションは、見つかったタグ非表示化を解除します。

非表示の解除は、タグがページで表示されるように、タグのスタイル属性を設定することで行います。

次の時に続行
ステップに待機基準を追加します。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときやメイン フレームのエレメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。詳細については、待機基準の使用 を参照してください。

オプション
ステップのオプションをロボットのオプションよりも優先させることができます。オプション ダイアログでアスタリスクが付いているオプションは、ロボットの設定のオプションに優先します。その他のすべてのオプションは、ロボットに対して指定したものと同様になります。

CSV 形式表示

このアクションでは、ダウンロードした CSV コンテンツが CSV ビューで開きます。

このステップは、ダウンロードした CSV コンテンツのみで動作します。

プロパティ

CSV 形式表示アクションは、以下のプロパティを使用して設定できます。

ヘッダーを使用
このプロパティは、CSV ドキュメントの最初の行が列の名前を定義する場合にチェックします。

区切り文字
CSV ドキュメントの解析に使用される区切り文字。

引用文字
CSV ドキュメントの解析に使用される引用文字。

その他のエスケープ文字
CSV パーサーは、別の二重引用文字が続く場合、エスケープ文字として二重引用文字 (") を解釈します。その他のエスケープ文字を使用する場合は、ユーザーはここで指定できます。

エンコーディング
この CSV ドキュメントに使用されるエンコーディング。

トップ ラインをスキップ
CSV ファイルの上部からスキップするラインの数を設定します。これは、システムが、実際の CSV データの一部でない CSV ファイルの開始部分で多数のラインを自動的に加えている場合に特に便利です。

下部をスキップ
CSV ファイルの下部からスキップするラインの数を設定します。これは、システムが、実際の CSV データの一部ではない多数のラインを CSV ファイルの最初に自動的に加えている場合に特に便利です。

Excel 形式表示

このアクションでは、ダウンロードした Excel コンテンツが Excel ビューで開きます。

このステップは、ダウンロードした Excel コンテンツのみで動作します。

プロパティ

このアクションにはプロパティがありません。

JSON 形式表示

このアクションでは、ダウンロードした JSON コンテンツが JSON ビューで開きます。

このステップは、ダウンロードした JSON コンテンツのみで動作します。

プロパティ

JSON 形式表示アクションは、以下のプロパティを使用して設定されます。

エンコーディング
この JSON ドキュメントに使用されるエンコーディング。

XML 形式表示

このアクションでは、ダウンロードした XML コンテンツが XML ビューで開きます。

このステップは、ダウンロードした XML コンテンツのみで動作します。

プロパティ

このアクションにはプロパティがありません。

待機

このアクションは、指定した期間にわたり待機します。待機は、Design Studio の Design モードでの実行中ではなく、ランタイム実行中にのみ実行されます。

プロパティ

待機アクションは、以下のプロパティを使用して設定されます。

秒
待機する秒数。秒の値は正数である必要があります。また、この値は、値セレクターを使用してさまざまな方法で指定できます。

ファイル出力

このアクションは、新しいファイルを出力するか、既存のファイルに追加します。

オプションの [デザイン モードで実行] が選択されている場合は、Design Studio においてデザイン モードで実行している際にのみファイルが書き込まれます。

指定のファイルが存在するかどうかをテストするには、ファイル有無判定アクションを使用します。

CSV (コンマ区切りの値) ファイルを書き込む必要がある場合は、以下の 2 つの方法で行うことができます。「CSV に追加」データ コンバータを使用して各ラインを作成して、ファイル出力で 1 度に 1 ラインずつファイルを書き込みます。あるいは、必要なファイル コンテンツを 1 度に 1 ラインずつグローバル変数で構築し (再び「CSV に追加」データ コンバータを利用)、ファイル出力を末尾で使用して (別の分岐で)、1 つのピースでファイルを書き込みます。

ファイルの内容を保持するグローバル変数を変更するには、変数自体を指す [変数を取得] データ コンバータから入力を取得する [変数の割当] アクションを使用します。これは、「CSV に追加」データ コンバータの後に続きます。

プロパティ

ファイル出力アクションは、以下のプロパティを使用して設定できます。

ファイル名

ファイル名。名前は、値セレクターを使用してさまざまな方法で指定できます。名前は絶対パス名である必要があります。該当する場合はドライブ名とファイルへのディレクトリ パスを含めます。

ファイル コンテンツ

ファイルに書き込むコンテンツコンテンツは、値セレクターを使用してさまざまな方法で指定できます。このコンテンツは、バイナリ データまたはテキストのいずれかです。後者の場合、[ファイル エンコーディング] プロパティで選択される文字のエンコーディングは、バイトとしてテキストをエンコーディングするのに使用されます。CSV、XML、および JSON ファイルにファイル コンテンツを指定するためにコンバータを使用する方法については、「データ コンバータ」セクションを参照してください。

ファイル エンコーディング

ファイルに書き込むコンテンツがテキストである場合、このプロパティは、テキストをバイトしてエンコーディングするために使用する文字エンコーディングを指定します。

ファイルに追加

新しいファイルが常に作成される (同じ名前を持つ既存のファイルを上書きする) ようにする必要があるかどうか、または、ファイルがすでに存在する場合に、コンテンツがファイルに追加されるようにする必要があるかどうかを指定します。ファイルに追加は、ファイルの形式に関係なく、既存のファイルの末尾に新しいコンテンツのみを追加します。そのため、このオプションは、テキスト変数とともにのみ使用される必要があります。

ディレクトリを作成

新しいファイルを作成する前に、そのファイル パス上に必要なディレクトリを作成するかどうかを指定します。選択されず、パスにディレクトリが存在しない場合、アクションは失敗します。

デザイン モードで実行

これが有効になっていると、アクションは Design Studio 内部のデザイン モードでも実行されます。これが無効になっていると、ロボットをデザイン モードでナビゲートするときにアクションは何も実行しません。

ログ出力

このアクションでは、ログ システムに情報を書き込む機会が生まれます。

ログ システムが何であるかは、ロボットが実行されている仕組みによって変わります。ロボットが Design Studio においてデザイン モードで実行されている場合は、ログ情報が [ログ] ウィンドウに表示されます。これは、[表示] メニューから開くことができます。ロボットがデバッグ モードで実行されている場合、ログは [ログ] タブに表示されます。ロボットが RoboServer を使用して実行されている場合、ログ情報はメッセージ データベースまたは他の場所に送信される可能性があります。これは、Kofax RPA の設定によります。

プロパティ

ログ出力アクションは、以下のプロパティを使用して設定されます。

メッセージ

このフィールドには、ログに書き込まれるメッセージが含まれます。この値は、値セレクターを使用してさまざまな方法で指定できます。

データ コンバータ

このセクションでは、利用可能なデータ コンバータの概要を説明します。

以下のデータ コンバータの説明では、いずれも Kofax RPA の中心的なテキスト操作概念であるパターンと式について解説します。これら 2 つの概念の説明については、パターンとエクスプレッションを参照してください。

Design Studio のコンバータは、テキスト タイプの値しか処理できないため、エクスプレッションの出力値が別のタイプに割り当てられると、コンバータの出力でエラーが発生することがあります。たとえば、エクスプレッションから日付書式を使用して `now()` 関数の値を Excel に入力すると、正しい書式になりますが、コンバータから `now()` 関数の値を追加すると、正しくない出力になります。

標準

このカテゴリーには最もよく使われるデータ コンバータが含まれます。テキストを処理する最も一般的な方法は [抽出] を使用する方法です。[抽出] ではパターンを利用してテキストを抽出することができます。エクスプレッションを評価するには、[エクスプレッションを評価] を使用します。テキストを追加するには [テキストを追加] を使用します。変換規則のリストを利用してテキストを変換するには、[リストを使用して変換] を使用します。変数の値を取得するには、[変数を取得] を使用します。

アクション	説明
抽出	このデータ コンバータはパターンを利用して入力テキストからテキストを抽出します。抽出する部分を 1 組の括弧でマークする必要があります。
エクスプレッションを評価	このデータ コンバータはエクスプレッションの結果を出力します。INPUT 表記を利用して、データ コンバータへの入力テキストをエクスプレッションで使用することができます。
テキストの追加	このデータ コンバータは入力テキストの前または後ろにテキストを追加します。
リストを使用して変換	このデータ コンバータは、変換リストを使用して入力テキストを出力値テキストに変換します。
変数を取得	このデータ コンバータは変数の値をフェッチします。入力テキストは無視されます。

抽出

このカテゴリーには抽出用のデータ コンバータが含まれます。最もよく使用されるのは [抽出] です。[抽出] ではパターンを利用してテキストを抽出することができます。より高度な抽出機能が必要な場合は、[アドバンスド抽出] を使用します。同じテキストに対して [アドバンスド抽出] の機能を複数回適用するには、[抽出リスト] を使用します。

アクション	説明
抽出	このデータ コンバータはパターンを利用して入力テキストからテキストを抽出します。抽出する部分を 1 組の括弧でマークする必要があります。
アドバンスド抽出	このデータ コンバータは入力テキストとパターンを照合し、エクスプレッションの結果を出力します。
抽出リスト	このデータ コンバータはパターンのすべての一致を見つけて、個々の一致についてエクスプレッションを評価します。出力テキストはエクスプレッションの結果のリストです。

テキスト書式設定

このカテゴリーにはさまざまな種類のテキスト書式設定用のデータ コンバータが含まれます。テキストを追加するには [テキストを追加] を使用します。テキストの検索と置換を行うには、[テキストを置換] を使用します。パターンを利用してテキストの検索と置換を行うには、[パターンを置換] を使用します。入力テキストからスペースを除去するには、[スペースを除去] を使用します。すべての特殊文字を除去するには、[特殊文字を除去] を使用します。すべての印刷不可文字を除去するには、[印刷不可文字を除去] を使用します。大文字と小文字を入れ替えるには、[小文字へ変換]、[大文字へ変換]、または [キャピタライズ] を使用します。

アクション	説明
テキストの追加	このデータ コンバータは入力テキストの前または後ろにテキストを追加します。
テキストを置換	このデータ コンバータは入力テキスト内の一致するテキストを検索し、置き換えます。
パターンを置換	このデータ コンバータは、パターンのすべての一致をエクスプレッションの結果に置き換えます。
スペースを除去	このデータ コンバータは入力テキストからスペースを除去します。
特殊文字を除去	このデータ コンバータは入力テキスト内のすべての特殊文字をスペースに置き換えます。
印刷不可文字を除去	このデータ コンバータはすべての印刷不可文字を除去します。
小文字変換	このデータ コンバータは入力テキスト内のすべての文字を小文字に変換します。
大文字変換	このデータ コンバータは入力テキスト内のすべての文字を大文字に変換します。
キャピタライズ	このデータ コンバータは入力テキストをキャピタライズします。つまり、すべての語の最初の文字を大文字にして、その他の文字を小文字にします。
テキストの引用符を除去	このデータ コンバータは二重引用符または単一引用符で囲まれたテキストの引用符を除去します。

数値の処理

このカテゴリーには数値処理用のデータ コンバータが含まれます。テキストから数値を抽出するには、[数値を抽出] を使用します。すでに標準の数値書式になっている数値を書式設定するには、[数値を書式設定] を使用します。

アクション	説明
数値を抽出	このデータ コンバータは入力テキストから数値を抽出し、標準の数値書式で数値を出力します。
数値を書式設定	このデータ コンバータは標準の数値書式になっている数値を再び書式設定します。

日付の処理

このカテゴリーには日付処理用のデータ コンバータが含まれます。テキストから日付抽出するには、[日付抽出] を使用します。テキストから年を抽出するには、[年を抽出] を使用します。すでに標準の日付書式になっている日付を書式設定するには、[日付を書式設定] を使用します。2 つの日付の間の時間を取得するには、[日付間の時間を取得] を使用します。日付を変更するには、[日付を変更] を使用します。

アクション	説明
日付抽出	このデータ コンバータは入力テキストから日付抽出し、標準の日付書式で日付を出力します。
年を抽出	このデータ コンバータは入力テキスト内の日付から年を抽出します。
日付を書式設定	このデータ コンバータは標準の日付書式になっている日付を再び書式設定します。
日付間の時間を取得	このデータ コンバータでは、2 つの日付の差を求めることができます。入力テキスト内の日付を、指定された日付と比較し、選択された単位で日付の差を計算します。
日付を変更	このデータ コンバータは、日付の選択部分に時間を加算または減算することによって入力日付を変更します。この操作によって日付の当該部分がオーバーフローまたはアンダーフローした場合は、日付の他の部分もそれに従って更新されます。変更が実行されるタイム ゾーンを指定することができます。
Excel の日付に変換	日付を標準の日付書式から Excel 書式へ変換します。
Excel の日付から変換	日付を Excel 書式から標準の日付書式へ変換します。

ブール値処理

このカテゴリには、ブール値を処理するためのブール コンバータが含まれています。

アクション	説明
ブール値の抽出	このデータ コンバータはパターンを検索し、ブール値を返します。

HTML の処理

このカテゴリーには HTML 処理用のデータ コンバータが含まれます。テキストからすべての HTML タグを除去するには、[タグ除去] を使用します。HTML を構造化プレーンテキストに変換するには、[HTML からテキストへ変換] を使用しますHTML を再び書式設定 (プリティプリント) するには、[HTML を書式設定] を使用します。特定のタグが入力テキストに出現する回数をカウントするには、[タグをカウント] を使用します。

アクション	説明
タグ除去	このデータ コンバータは入力テキストから HTML タグを除去します。
HTMLをテキストに変換	このデータコンバータは入力 HTML テキストをプレーン テキストに変換し、ブラウザに表示される形式と同様の形式でテキストを構造化します。
HTML を書式設定	このデータ コンバータは入力 HTML テキストを再び書式設定 (プリティプリント) します。
カウント タグ	このデータ コンバータは、[タグ名] フィールドに入力された名前と完全に一致する名前を持つタグの数をカウントします。入力文字列内に対応する終了タグがあるタグのみをコンバータにカウントさせる場合は、[終了タグが必要] チェックボックスにチェックを入れます。

出力書式の処理

このカテゴリーには出力書式処理用のデータ コンバータが含まれます。個々のデータ コンバータは、オブジェクトを書式設定して特定の出力書式に変換し、それを (以前作成された部分結果であるはずの) 入力テキストに追加することができます。あるいは、選択された出力書式の規則に従って任意のテキストを入力テキストに追加することもできます。

アクション	説明
CSV に追加	このデータ コンバータは変数を CSV 形式に変換し、入力テキストに追加します。

エンコーディングとデコーディング

このカテゴリにはエンコードおよびデコード用のデータ コンバータが含まれています。アンパサンドをデコードまたはエンコードするには、[アンパサンド デコード] または [アンパサンド エンコード] を使用します。URL をエンコードまたはデコードするには、[URL エンコード] または [URL デコード] を使用します。バイナリ データを Base64 エンコードまたは Base64 デコードするには、[Base64 エンコード] または [Base64 デコード] を使用します。

アクション	説明
アンパサンド エンコード	このデータ コンバータは、特定の文字を「&<特定の文字>」に置き換えるアンパサンド エンコーディングで文字をエンコードします。
アンパサンド デコード	このデータ コンバータは、すべての HTML アンパサンド エンコーディングを実際の文字にデコードします。
URL エンコード	このデータ コンバータは URL エンコーディングで文字をエンコードします。
URL デコード	このデータ コンバータはすべての URL エンコーディングをデコードして、それらに対応する実際の文字に変換します。
Base64 エンコード	このデータコンバータは Base64 エンコーディングを使用してデータをエンコードします。
Base64 デコード	このデータ コンバータは Base64 エンコードされたデータをデコードします。
バイナリをテキストに変換	バイナリ変数をテキストに変換します。入力テキストは無視されます。
テキストをバイナリに変換	テキスト変数をテキスト変数のバイナリの 16 進数表現に変換します。入力テキストは無視されます。

その他

このカテゴリーにはその他さまざまなデータ コンバータが含まれます。

アクション	説明
エクスプレッションを評価	このデータ コンバータはエクスプレッションの結果を出力します。INPUT 表記を利用して、データ コンバータへの入力テキストをエクスプレッションで使用することができます。
リストを使用して変換	このデータ コンバータは、変換リストを使用して入力テキストを出力値テキストに変換します。
JavaScript を使用して変換	このコンバータでは JavaScript を使用して変換を定義することができます。入力は INPUT 変数で利用でき、変換の結果を OUTPUT 変数に割り当てる必要があります。
If Then	If Then データ コンバータでは、コンバータの出力値を決定する if-then ルールのリストを指定することができます。
変数を取得	このデータ コンバータは変数の値をフェッチします。入力テキストは無視されます。
プロパティを取得	このデータ コンバータは変数に含まれているプロパティ リストからプロパティの値を取得します。

アクション	説明
絶対 URL に変換	このデータ コンバータは URL を絶対 URL に変換します。
相対 URL に変換	相対 URL に変換
MD5 チェックサム計算	このデータ コンバータは入力テキストの MD5 チェックサムを計算します。

廃止予定

このカテゴリーには新しいバージョンに置き換えられたデータ コンバータや使用されていないデータ コンバータが含まれます。これらのデータ コンバータは Design Studio の古いバージョンを使用して書かれたロボットとの後方互換性を保つ目的でのみ利用可能になっています。これらのデータ コンバータを新しいロボットで使用しないでください。

テキストの追加

このデータ コンバータでは、入力テキストの前後へのテキストの追加ができます。

プロパティ

追加テキストのデータ コンバータの設定には、以下のプロパティを使用します。

追加するテキスト

このフィールドには、入力テキストに追加するためのテキストが含まれます。

追加位置

テキストの追加位置を入力テキストの前にするか後にするか選択します。

説明

データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

CSV への追加

このデータ コンバータは、変数を CSV フォーマットに変換し、入力テキストに追加します。

このデータ コンバータを使用した、完全な CSV ファイルのコンテンツの作成および書き込み方法についてのヘルプは、フファイル出力アクションを参照してください。

プロパティ

CSV への追加データ コンバータは、以下のプロパティを使用して設定します。

フォーマットする変数

入力テキストの末尾に追加する前に、CSV としてフォーマットされる変数を指定します。CSV には、「保存可能」としてマークされていない、変数のすべての属性が含まれます。

ヘッダーを作成

これにチェックを入れると、選択された変数に一致するヘッダーの行が作成されます。変数の属性のストレージ名 (またはデフォルトでは名前) は、このヘッダーの列タイトルとして使用されます。このプロパティにチェックを入れない場合、変数の属性の値を含んだデータの行が作成されます。

データ フォーマット ロケール

データの行を作成する場合、日付に必要なロケールを、日付の書式設定データ コンバータと同じ様に指定する必要があります。

データ フォーマット パターン

データの行を作成する際、必要なデータ フォーマット パターンを、フォーマット データのデータ コンバータと同じ様に指定する必要があります。

フィールド区切り文字

必要な、行の個別のフィールド間の区切り記号これは、コンマまたは TAB 文字で区切ることができます。コンマを選択すると、オプションとしてフィールド値を引用符 (」文字) で入力することができます。この場合、フィールド値の引用符文字は 2 重になります (」が「」となります)。引用符で閉じたコンマ区切りの値については、コンマの後に空白文字を入れるかどうかを指定できます。

説明

データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

アドバンスド抽出

このデータ コンバータでは、パターンやエクスプレッションを使用した柔軟な方法で、入力テキストの操作ができます。パターンは入力テキストからテキスト ピースを抽出するために使用でき、このテキスト ピースをエクスプレッションで使用して出力テキストを構築することができます。

プロパティ

アドバンスド抽出データ コンバータの設定には、以下のプロパティを使用します。

パターン

入力テキストと一致するパターンが含まれます。パターンは、入力テキスト全体で一致している必要があることに注意してください。一致していない場合、コンバータが失敗し、エラーが生成されます。

大文字と小文字を無視

このオプションを選択すると、パターンの一致で大文字と小文字が区別されなくなります。つまり、大文字と小文字に関係なくパターンは入力テキストに一致します。

出力エクスプレッション

結果がデータ コンバータの出力値となるエクスプレッションが含まれます。エクスプレッションは、$n 表記を使用することでパターンのサブマッチを確認することができます。たとえば、エクスプレッションに $1 を入力して、パターンの最初のサブマッチ (すなわち、パターンの括弧の最初の部分の内容に一致するテキスト) を取得することができます。

説明

データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

アンパサンド デコード

このデータ コンバータは、すべての HTML アンパサンド エンコーディングを実際の文字にデコードします。

プロパティ

アンパサンド デコードのデータ コンバータの設定には、以下のプロパティを使用します。

NBSP を通常の空白文字に変換
  を、改行なしの空白文字ではなく通常の空白文字に変換するように指定します。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
    &amp;alt; &amp;gt; &amp;amp; &amp;quot;
```

このデータ コンバータは以下を出力します。

```
    < > & "
```

アンパサンド エンコード

このデータ コンバータは、HTML アンパサンド エンコーディングで文字をエンコードします。

詳細については、以下を参照してください。

http://www.w3.org/TR/html401/charset.html#h-5.3.2

プロパティ

アンパサンド エンコードのデータ コンバータの設定には、以下のプロパティを使用します。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
    < > & " á ç a
```

このデータ コンバータは以下を出力します。

```
    &amp;alt; &amp;gt; &amp;amp; &amp;quot; &amp;aacute; &amp;ccedil; a
```

Base64 デコード

このデータ コンバータは、Base64 エンコード テキスト データをバイナリにデコードします。

プロパティ

Base64 デコードのデータ コンバータの設定には、以下のプロパティを使用します。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
yv66vg==
```

このデータ コンバータは以下を出力します。

```
CA FE BA BE
```

base64 デコードのコンバータは、ダッシュ ("-") およびアンダースコア ("_") 文字をサポートしていません。そのため、Base64 でコーディングを行う前に、これらの文字を置き換える必要があります。"-" の代わりに "+" を、"_" の代わりに "/" を使用します。

Base64 エンコード

このデータ コンバータは Base64 エンコーディングを使用してバイナリデータを文字列にエンコードします。

プロパティ

以下のプロパティを使用して Base64 エンコード データ コンバータを設定できます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
CA FE BA BE
```

このデータ コンバータは以下を出力します。

```
yv666vg==
```

ブール コンバータ

ブール コンバータは任意のパターンを変換し、パターンが見つかれば、ブール値の true または false を返します。パターンが見つからなければ、値は返されず、代わりに警告メッセージが [テスト入力値] ウィンドウの上部に表示されます。

ブール コンバータを使用する
以下の方法でコンバータにアクセスできます。

- 抽出アクションを利用してブール データを抽出する
  [アクション] タブで **[抽出]** > **[抽出]** を選択し、[コンバータ] の下のプラス記号をクリックして、**[ブール値処理]** > **[ブール値の抽出]** を選択します。
- [ページ] ビューで右クリック メニューを使用する

抽出元のテキストを右クリックし、[抽出] メニューで適切なオプションを選択します。

プロパティ

以下のプロパティを利用してブール コンバータを設定します。

フォーマット
定義されたパターンと予想される結果、true または false を表示します。

パターン
[パターン] リストには、yes、no などのデフォルトの推奨パターンが含まれています。**[**パターン**]** リストの右側にあるドロップダウン ボックスをクリックして、[値]、[変数]、[エクスプレッション]、または [コンバータ] から選択することもできます。

大文字と小文字を無視
このオプションを選択すると、パターンの内容の大文字と小文字の区別がアクションで無視されます。

値
リストから [true] または [false] を選択します。

テスト入力値
テストの入力値を入力します。

テスト出力値
定義されたフォーマットに従って対応する出力が表示されます。

変換規則

- デフォルトでは、入力テキストがブール値そのものである場合、出力値は入力値と同じになります。出力では大文字と小文字が区別されることに注意してください。この規則は小文字の true および false にのみ適用されます。
- フォーマットの順番は問題になります。つまり、アクションは上から順に一致するフォーマットを検索します。一致するパターンが見つかると、見つかったパターンの下にある他のパターンに関係なく、結果が返されます。

キャピタライズ

このデータ コンバータは入力テキストをキャピタライズします。これは、すべての語の最初の文字を大文字にして、その他の文字を小文字にすることを意味しています。

プロパティ

以下のプロパティを使用してキャピタライズ データ コンバータを設定できます:

キャピタライズしない語
キャピタライズしない語を含めます。語はカンマで区切られます。たとえば、"in" および "the" という語をキャピタライズしない場合は、プロパティを "in, the" に設定します。

キャピタライズする語の最小の長さ
キャピタライズする必要のある語の最小の長さ。たとえば、このプロパティが 3 に設定されている場合、3 文字未満の語はすべてキャピタライズされません。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
    'hello WORLD'
```

出力テキストは次のようになります:

```
    'Hello World
```

MD5 チェックサム計算

MD5 チェックサム計算データ コンバータは入力テキストの MD5 チェックサムを計算します。

このデータ コンバータは、後のいずれかの時点でデータの変更を簡単に検出できるように、テキストまたはバイナリ データのデジタル指紋を作成するのに役立ちます。MD5 チェックサムはバイナリ データにのみ適用できるため、MD5 を適用する前に、文字列入力をエンコードする必要があります。テキストから生成されたチェックサムを外部の MD5 サービスによって生成されたチェックサムと比較する場合は、外部のサービスがロボットと同じエンコーディングを使用していることを確認してください。

バイナリ データが含まれた属性からテキストを読み取ったときに出力される 16 進数テキスト表現を含むあらゆるテキストを入力として渡すことができます。データ コンバータの出力値は、32 桁の 16 進数として表現される、テキストの 128 ビット MD5 チェックサムです。実用上、この数は入力テキストの一意の識別子と見なすことができます。つまり、実際問題として 2 つのテキストの MD5 チェックサムが同じになることは決してありません。

プロパティ

以下のプロパティを使用して MD5 チェックサム計算データ コンバータを設定できます：

エンコーディング
md5 を適用する前のテキストのエンコードに使用されるエンコーディング。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
    The quick brown fox jumps over the lazy dog
```

出力値は以下のようになります：

```
    9E107D9D372BB6826BD81D3542A419D6
```

入力テキストをわずかに変更しても、まったく異なる出力値が生成されます。たとえば、"dog" の 'd' を 'h' に変更して入力が以下のようになると：

```
    The quick brown fox jumps over the lazy hog
```

出力値は以下のようになります：

```
5681F8C64F7CA70B12E0B80435265203
```

バイナリ・テキスト変換

選択されたエンコーディングを使用して、バイナリ変数のコンテンツをテキストに変換します。入力テキストは無視されます。このアクションは、HTML ファイル、XML ファイルまたはテキストファイルがバイナリ変数を介してロボットに入力として渡されるときに使用されます。

プロパティ

以下のプロパティを利用してバイナリをテキストに変換するデータ コンバータを設定します:

変数
取得する値が含まれた変数。これはバイナリ属性である必要があります。

エンコーディング
バイナリ変数のバイトのデコードに使用されるエンコーディング。テキストに変換できるバイナリ コンテンツは、たいていの場合、UTF-8、Latin-1 (ISO-8859-1)、または Latin-1 (Windows-1252) でエンコードされます。この変換によって生成されるテキストに文字「#」が含まれている場合は、選択されたエンコーディングが間違っているか、文字が表示不能です。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

Excel 日付からの変換

このデータ コンバータは Excel 数値の日付を標準形式に変換します。

プロパティ

以下のプロパティを使用して Excel 日付からの変換データ コンバータを設定できます :

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
1
```

出力テキストは次のようになります:

```
1900-01-01 00:00:00.0
```

HTMLをテキストに変換

このデータ コンバータは HTML 入力テキストをプレーン テキストに変換し、ブラウザに表示される形式と同様の形式でテキストを構成します。

プロパティ

以下のプロパティを使用して HTML をテキストに変換するデータ コンバータを設定します:

位置指定されたテーブルとイメージを含める
テキストの右端または左端に位置合わせされた表および画像を出力テキストに含めるよう指定します。このプロパティを無効にすると、目的のコンテンツが削除されることがあります。

URL を含める
リンク タグ内の実際の URL を出力テキストに含めるよう指定します。

イメージ テキストの代替要素を含める
画像のテキスト表現を出力テキストに含めるよう指定します。

フォーム フィールドを含める
フォーム フィールドのテキスト表現を出力テキストに含めるよう指定します。

見出しの前にこれを挿入
このデータ コンバータが見出しの位置を推測し、その見出しの前に指定されたテキストを挿入するよう指定します。

見出しの後にこれを挿入
このデータ コンバータが見出しの位置を推測し、その見出しの後に指定されたテキストを挿入するよう指定します。

アンパサンド エンコーディングを保持
アンパサンド エンコーディングをデコードしないよう指定します。スクリプトおよびスタイル シートに含まれたテキストは保持されます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

テキスト・バイナリ変換

選択されたエンコーディングを使用して、テキスト変数のコンテンツをテキストのバイナリの 16 進数表現に変換します。このステップを使用して、テキストをバイナリとしてエンコードし、ファイル アップロードに使用することができます。

プロパティ

以下のプロパティを利用してテキストをバイナリに変換するデータ コンバータを設定します:

エンコーディング
テキストのエンコードに使用されるエンコーディング。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

Excel 日付への変換

このデータ コンバータは日付をスプレッドシートで同じ日付を表す Excel 数値に変換します。

プロパティ

以下のプロパティを使用して Excel 日付への変換データ コンバータを設定できます :

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
    1900-01-01 00:00:00.0
```

出力テキストは次のようになります:

```
    1.0
```

小文字変換

このデータ コンバータは入力テキスト内のすべての文字を小文字に変換します。

プロパティ

以下のプロパティを使用して小文字変換データ コンバータを設定できます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
    "HELLO World"
```

出力テキストは次のようになります:

```
    "hello world"
```

大文字変換

このデータ コンバータは入力テキスト内のすべての文字を大文字に変換します。

プロパティ

以下のプロパティを使用して大文字変換データ コンバータを設定できます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合、

```
    "hello World"
```

出力テキストは次のようになります:

```
    "HELLO WORLD"
```

JavaScript を使用して変換

このデータ コンバータは、JavaScript を使用した変換を指定できます。このデータ コンバータは、たとえば、URL のリライトなどの高度なテキスト操作の作業や複雑な計算を行うときに便利です。制限については、「Kofax RPA の制限」を参照してください。

コンバータへの入力は、暗黙的に定義された INPUT 変数で可能です。変換の実施結果は OUTPUT 変数に割り当てる必要があります。ヘルパー関数を定義して、必要に応じて呼び出すことができます。このコンバータでは、JavaScript からブラウザ状態にアクセスできないので、注意してください。

プロパティ

「JavaScript を使用して変換」データ コンバータは、次の各プロパティを使用して設定できます。

スクリプト

実行する JavaScript。これはリテラルで指定するか、または値セレクターを使用してさまざまな方法で作成できます。

説明

データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例: JavaScript 変換

例 1

数のコンマ区切りリストの平均を計算するには、計算を行う次のスクリプトで「JavaScript を使用して変換」データ コンバータを設定します。

```
var sum = 0;
var numbers = INPUT.split(",");
for (var i = 0; i < numbers.length; i++) {
    sum += parseInt(numbers[i]);
}

OUTPUT = sum / numbers.length;
```

JavaScript は INPUT 変数から "23,22,25,31,24" などの数のリストを読み取ります。組み込み済みの JavaScript split() 関数を使用して、これを各コンマで分けます。数の全体で繰り返して合計を算出します (parseInt() 関数を使用して、文字列を整数に変換します。それ以外の場合は、合計の算出ではなく、文字列の連結を行います)。最後に平均を計算して、その結果を OUTPUT 変数に割り当てます。

データ コンバータへの入力値が文字列 "23,22,25,31,24" の場合は、コンバータの出力値は 25 になります。

例 2

「$10.50,$13,$21.75,$7" などの金額のコンマ区切りリストの最大値を計算するには、計算を行う次のスクリプトで「JavaScript を使用して変換」データ コンバータを設定します。

```
function getNumber(amountWithDollarSign) {
    return parseFloat(amountWithDollarSign.substring(1));
}

var amountsWithDollarSigns = INPUT.split(",");
var max = getNumber(amountsWithDollarSigns[0]);
```

```
    for (var i = 1; i < amountsWithDollarSigns.length; i++) {
        max = Math.max(max, getNumber(amountsWithDollarSigns[i]));
    }

    OUTPUT = max;
```

上記の JavaScript では、getNumber() という名前のヘルパー関数を定義しています。これが金額の前に付いているドル記号を除去して、残りを浮動小数点数に変換します。この関数はスクリプト内で繰り返し呼び出されます。組み込み済みの JavaScript 関数 Math.max() を使用して 2 つの数のうちの最大値を見つけます。各イテレーションで、それまでに見つけた最高値の数を次の数と比較します。

最後に、見つかった最高値の数を OUTPUT 変数に保存して、これがデータ コンバータの出力値になります。

文字列操作

次の各メソッドは、文字列オブジェクトを変換するときに便利です。regexp は // の中に記載しますが、文字列は「」の中に記載するので注意してください。regexp の終わりのグローバル g 属性は、メソッドをすべての一致に適用することを示します。

メソッド	説明
string.charAt(n)	指数 n を添えて文字を返します。
string.charCodeAt(n)	指数 n を添えた文字の Unicode 値を返します。
string.concat(value1, value2, ...)	1 つまたは複数の値を連結して文字列にします。
String.fromCharCode(c1, c2, ...)	文字の Unicode エンコードを指定する整数から、新規文字列を作成します。
string.indexOf(substring) string.indexOf(substring, start)	string.start 内のサブ文字列のインデックスを返して、検索を開始するインデックスを指定します (0 は文字列の最初の文字で、string.length-1 は最後の文字です。デフォルトは 0 です)。
string.lastIndexOf(substring) string.lastIndexOf(substring, start)	string.start 内のサブ文字列の最後の出現位置を返して、検索を開始するインデックスを指定します (0 は文字列の最初の文字で、string.length-1 (デフォルト) は最後の文字です)。
string.length	文字列の文字の長さ。
string.match(regexp)	正規表現 regexp で、一致する文字列を検索します。regexp に (g) で指定したグローバル属性がない場合は最初の一致だけを返しますが、グローバル属性がある場合は一致するすべての結果を含む配列を返します。
string.replace(regexp, replacement) string.replace(substring, replacement)	サブ文字列または正規表現で、一致する文字列を検索します。最初の出現を置換します。regexp に (g) で指定するグローバル属性がある場合は、すべての出現を置換します。
string.search(regexp)	正規表現で最初に一致した最初の文字のインデックスを返します。
string.slice(start, end)	start から end-1 までのすべての文字を含む文字列を返します。
string.split(delimiter, limit)	区切り記号は文字列または正規表現で、文字列を分ける場所を指定します。文字列の配列を返します。配列の長さは上限を超えません。
string.substr(start, length)	インデックス開始で始まり、指定の長さであるサブ文字列のコピーを返します。
string.substring(from, to)	from の位置で始まり、to-1 の位置で終わるサブ文字列を返します。

メソッド	説明
string.toLowerCase()	小文字に変換された文字列のコピーを返します。
string.toUpperCase()	すべての文字が大文字に変換された文字列のコピーを返します。

Math オブジェクト

以下の各プロパティおよびメソッドは、数学的な計算をするときに便利です。すべての角度はラジアンで測定されます。

プロパティ / メソッド	説明
Math.E	オイラー数を返します。
Math.LN10	10 の自然対数を返します。
Math.LN2	2 の自然対数を返します。
Math.LOG10E	10 を底とする E の対数を返します。
Math.LOG2E	2 を底とする E の対数を返します。
Math.PI	円周率を返します。
Math.SQRT1_2	1/2 の平方根を返します。
Math.SQRT2	2 の平方根を返します。
Math.abs(x)	絶対値を返します。
Math.acos(x)	逆余弦を計算します。
Math.asin(x)	逆正弦を返します。
Math.atan(x)	逆正接を計算します。
Math.atan2(y, x)	点に対する角度を計算します。入力は通常の (x,y) 座標を表しますが、順序は逆です。
Math.ceil(x)	数を切り上げます。
Math.cos(x)	余弦関数。
Math.exp(x)	e を x 乗します。
Math.floor(x)	数を切り捨てます。
Math.log(x)	自然対数を計算します。
Math.max(x1, x2, ...)	数の最大値を返します。
Math.min(x1, x2, ...)	数の最小値を返します。
Math.pow(x,y)	x の y 乗を計算します
Math.random()	0 ~ 1 の間の乱数を返します
Math.round(x)	最も近い整数に四捨五入します。
Math.sin(x)	正弦関数。
Math.sqrt(x)	平方根を計算します。
Math.tan(x)	正接関数。

数値

String(number) を使用して数を文字列に変換できます。Number(string) を使用すれば逆も可能です。数オブジェクトの便利なメソッドがいくつかあります。

メソッド	説明
number.toExponential(digits)	小数点の後にくる桁数を指定します。指数表記の数の文字列表現を返します。
number.toFixed(digits)	小数点の後にくる桁数を指定します。指数表記を使用しない数の文字列表現を返します。
number.toPrecision(precision)	有効数字の数を指定します。指定した数の有効数字で数の文字列表現を返します。
number.toString(base)	指定した進数を使用して数の文字列表現を返します。

リストを使用して変換

このデータ コンバータは、変換リストを使用して入力テキストを出力値テキストに変換します。データ コンバータは、特定の Web サイトで使用されるテキストをここで使用するテキストに変換したり、その逆に変換するのに役立ちます。たとえば、データ コンバータを使用して、国の名前と国のコードとを変換できます。

変換リストの指定

変換リストの指定は変換フィールドで行います。たとえば、国の名前と国のコードとの変換リストは次のようになります。

```
"Australia" = "AUS"
"Austria" = "AUT"
"Belgium" = "BEL"
"Brazil" = "BRA"
"Canada" = "CAN"
"China" = "CHN"
"Denmark" = "DNK"
"Egypt" = "EGY"
"Finland" = "FIN"
"France" = "FRA"
"Germany" = "DEU"
"Hungary" = "HUN"
"Iceland" = "ISL"
"India" = "IND"
"Ireland" = "IRL"
...
```

この変換リストに従って、たとえば、入力テキスト "Australia" は "AUS" に変換され、入力テキスト "Austria" は "AUT" に変換されます。

バックスラッシュ文字 (\) を使用すると、テキストに特殊文字を入力できます。

- \n は改行。
- \r はキャリッジ リターン。
- \f はドキュメント作成。
- \t は水平タブ。
- \b はバックスペース。

- \」は二重引用符。
- \' は一重引用符。
- \\ はバックスラッシュ。
- \uxxxx は xxxx エンコードの Unicode 文字。xxxx は 4 つの 16 進数の値。

テキスト前後の引用符は省略できます。その場合は、テキストの開始と終了箇所のスペースはすべて除去されます。テキストは空にできません。また、バックスラッシュ表記を特殊文字の入力に使用できません。

変換リストには空行やコメント行を含めることができます。コメント行は 2 本のスラッシュ (//) で開始します。

変換が一致しない場合の処理

「一致変換なし」オプションで、入力テキストがどの変換値にも一致しない場合の動作が決定されます。

- エラー生成では、データ コンバータがエラーを生成します。このエラーは、データ コンバータを使用するステップの設定に従って処理されます。
- テキスト変換なしでは、入力テキストを変換しません。つまり、入力テキストを変換せずに、出力テキストとして使用します。
- デフォルト テキストに変換では、入力テキストを、「デフォルト テキスト」フィールドで指定するテキストに変換します。

注意 : [デフォルト テキスト] フィールドでは、引用符なしでテキストを指定します。テキスト内に引用符文字が必要な場合は、直接入力できます (\」表記を使用しないでください)。

その他のプロパティ

「リストを使用して変換」データ コンバータは、このほかにも次の各プロパティを使用して設定できます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

カウント タグ

「カウント タグ」データ コンバータは、入力テキストに特定のタグが出現した回数をカウントします。これはタグの名前と正確に一致します。

プロパティ

「カウント タグ」データ コンバータは、次の各プロパティを使用して設定できます。

タグの名前
このフィールドに、カウントするタグの名前 ("tr" など) を入力します。

終了タグが必要
これを選択した場合、開始タグと終了タグを含むタグだけをカウントします。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

エクスプレッションを評価

このデータ コンバータはエクスプレッションを評価する機会を提供します。

INPUT 表記を利用して、データ コンバータへの入力テキストをエクスプレッションで使用することができます。以下に示す例を参照してください。

プロパティ

次のプロパティを利用して エクスプレッションを評価 データ コンバータを設定します。

エクスプレッション
結果がデータ コンバータの出力値となるエクスプレッションが含まれます。エクスプレッションは INPUT 表記を利用して入力テキストを参照することができます。このトピックの後半にある例を参照してください。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
次のエクスプレッションを使用して現在の日付時刻を取得することができます。

```
now()
```

入力テキストが「The time is」であり、現在の時刻を追加する必要がある場合は、次のエクスプレッションを使用することができます。

```
INPUT + time(now())
```

数値計算を実行することもできます。たとえば、整数属性 Item.price に 95 が含まれており、数値属性 Customer.discount に 25.0 が含まれている場合は、割引額を計算することができます。

```
(Item.price * Customer.discount)/100
```

同様に、入力テキストが "10" であれば、次のエクスプレッションを使用して、その値に 20 を乗じることができます。

```
toInteger(INPUT) + 20
```

抽出

このデータ コンバータでは、パターンを使用して簡単な方法で、入力テキストから 1 つのテキストを抽出できます。抽出する部分は、1 組の括弧でマークしてください。(抽出したテキストはパターンでの最初のサブマッチです)。

複数のテキストを抽出する場合や複数の操作オプションを指定する場合など、さらに高度な抽出オプションを設定するには、アドバンスド抽出データ コンバータを使用してください。

プロパティ

「抽出」データ コンバータは、次の各プロパティを使用して設定できます。

パターン
入力テキストと一致するパターンが含まれます。抽出する入力テキストの部分は、1 組の括弧でマークしてください。パターンは、入力テキスト全体で一致している必要があることに注意してください。一致していない場合、コンバータが失敗し、エラーが生成されます。

大文字と小文字を無視
このオプションを選択すると、パターンの一致で大文字と小文字が区別されなくなります。つまり、大文字と小文字に関係なくパターンは入力テキストに一致します。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストの最初の語を抽出するには、次のパターンを使用してください。

```
(\S*)\s?.*
```

これにより、スペース以外の文字の最初の連続が抽出されます。

日付抽出

このデータ コンバータは、日付を検知して抽出します。抽出した日付は、標準的な日付の形式で出力されます。

注意：すでに標準的な日付の形式になっているものを変更するには、[日付を書式設定] データ コンバータを代わりに使用します。

プロパティ

[日付抽出] データ コンバータは、次の各プロパティを使用して設定できます。

基本
フォーマット
日付の各形式。試される順番で並んでいます。入力に一致する最初の日付形式が適用されます。一致しない場合、データ コンバータはエラーを生成します。リスト上部にある '+' 記号をクリックして、新しい日付の形式を追加します。データ コンバータは、次の 2 種類の日付の形式をサポートします。フォーマット パターンおよび相対日付。フォーマット パターンには、MM/dd yyyy hh:mm のようなパターンの日付の仕様があります。月、日、または年の各フィールドに指定がない場合は、今日の日付に関連して、この日付が過去に属するか未来に属するかによって決められます。「時間軸の方向」プロパティの説明を参照してください。フォーマット パターンには、以下のプロパティがあります。

パターン
抽出する日付の形式を指定するパターン。このトピック内の後にある「相対日付パターンの構文」セクションを参照してください。

ロケール
入力で使用するロケールを指定します。これはたとえば、入力に "Monday, 25 May 2009" のように、月や曜日の名前が含まれる場合に使用します。

デフォルトの日付

このオプションを使用して、現在の日付を除く日付を指定し、不完全な日付を解決します。デフォルトでは、このオプションは現在の日付にセットされています。

アドバンスド

このタブには、未来や過去の日付を指定するオプションが含まれます。入力が関連すると考えられる日付。デフォルトでは、これは現在の日付と時間を取得する エクスプレッション now() です。

時間軸の方向

抽出する日付が過去であるか、未来であるかを指定します。これによってデータ コンバータは、形式パターンに月や年が欠落している場合、または相対日付から抽出する場合に、欠落する情報を補充できます。たとえば、「3 時間前」から抽出する場合、「次を基準」の中で指定する日付から、3 時間が差し引かれるようにするためには、時間軸の方向は「過去の日付」にセットする必要があります。一方で、「5 日で」などの入力から抽出する場合は、時間軸の方向は「未来の日付」にセットする必要があります

デフォルト タイム ゾーン

入力テキスト内の日付のデフォルト タイム ゾーン。タイム ゾーンを選択しない場合、デフォルト タイム ゾーンを使用しません。タイム ゾーンを選択した場合、入力テキストにタイム ゾーンが見つからないときに、このタイム ゾーンを使用します。

結果のタイム ゾーン

日付を変換するタイム ゾーン。タイム ゾーンを選択しない場合、変換しません。タイム ゾーンを選択した場合、入力テキストで検知した日付をそのタイム ゾーンから (またはデフォルト タイム ゾーン。上記参照) このタイム ゾーンに変換します。入力テキスト内の日付にタイム ゾーンがなく、デフォルト タイム ゾーンを選択しない場合、変換しません。

定数

数で指定するのではなく、数を記入する場合のある相対日付から抽出するための、言語依存の定数を指定します。たとえば、「1 時間前にアップデート済み」という入力から日付抽出するには、'HOURS hour[s] ago' というパターン付きの相対日付の形式を指定する必要があります。定数 'an = 1' が定義されていることが重要です。

説明

データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

フォーマット パターンの構文

以下の各パターンを組み合わせて、「パターン」プロパティ内のパターンを作成できます。

パターン	説明
yy	正確な 2 桁の年
yyy	年
yyyy	正確な 4 桁の年
YYYY[6]	正確な 4 桁の週年
G	紀元の表記 (AD または BC)

[6] 週年 (WEEK YEAR) を使用する場合、年の最初の日を含む週の年が新しい年として設定されます。

パターン	説明
MM	1 桁または 2 桁の月、月の略称、または完全な名前
dd	1 桁または 2 桁の日付
EEE	曜日の短い名前 (たとえば、Monday の代わりに Mon)。
EEEE	曜日の完全な名前 (Monday、Tuesday など)。
hh または HH	1 桁または 2 桁の時間
mm	1 桁または 2 桁の分
ss	1 桁または 2 桁の秒
a	AM または PM の表記
Z	タイム ゾーンの識別子 (たとえば、"PST"、"中央ヨーロッパ標準時"、または "GMT +02:00")
*	任意の文字数をスキップします
スペース	1 つまたは複数の空白をスキップします
その他の文字	正確にその文字をスキップします

曜日の各パターン ("EEE" および "EEEE") を使用して、日付パターン ("dd") を使用しない場合、月および年の各パターン ("MM" および "yy"/"yyy"/"yyyy") は使用できません。この場合、得られる日付は、次の、パターンに一致する名前が付いた日です。たとえば、パターンが 'EEEE' で、入力が次のテキストの場合:

```
Wednesday
```

得られる日付は次の水曜日です。

曜日パターンを日付パターンとともに (また、多くの場合、月および年パターンとともに) 使用する場合は、曜日を破棄します。たとえば、パターンが 'EEE, dd/MM/yyy' で、入力が次のテキストの場合:

```
Mon, 16/03/2003
```

得られる日付は '2003-03-16 00:00:00.0' (月曜日かどうかは無視されます)。

注意 : 'EEE' パターンは、曜日を示す短い名前に一致します (たとえば、Mon、Tue など)。パターンが曜日の名前の全体に一致する場合は、'EEEE' パターンを使用します。たとえば、入力が次のテキストの場合:

```
Thus, let us meet on Wednesday
```

'EEEE' のパターンを使用する必要があるため、'EEE' のパターンでは 'Thu' に一致し、日付抽出で次の木曜日が検知されます。

例: フォーマット パターンおよび一致する日付

フォーマット パターン	一致する日付
dd/MM-yyy	7/6-78 24/12-2001 1/jan-2001

フォーマット パターン	一致する日付
dd.MM yyy	4. jan 1993 4. january 93
yyyy G	2000 AD

相対日付パターンの構文

以下の各「日付」フィールドは、相対日付の「パターン」プロパティ内のパターンで使用できます。

「日付」フィールド	説明
`SECONDS`	秒
`MINUTES`	分
`HOURS`	時
`DAYS`	日
`MONTHS`	月
`YEARS`	年

「前」などの時間の表記はステップで自動的に認識されません。そのため、過去の相対日付抽出するには、[詳細] タブにある [時間軸の方向] リスト内の [過去の日付] を選択します。ロボットは、抽出した数を現在の時間から差し引きます。

未来の日付抽出するには、[詳細] タブにある [時間軸の方向] リスト内の [未来の日付] を選択します。次に、ロボットは抽出した数を現在の時間に加えます。

たとえば、「123 秒前」の文字列から正確な時間を知りたい場合、次のように指定します。

- [基本] タブで、`SECONDS sec[s] ago` を選択します。これは、[次を基準] 内の [パターン] フィールドおよび `now()` にあります。
- [詳細] タブで、[時間軸の方向] リストにある `Past date` を選択します。

  ステップは、現在の時間から 123 秒を差し引きます。

例: 相対日付パターン

フォーマット パターン	一致する日付
`HOURS hour[s] ago`	4 時間前 1 時間前 (定数 an = 1 が定義されている場合)
`HOURS hour[s] and MINUTES minute[s] ago`	3 時間 5 分前
`[HOURS hour[s] ]MINUTES minute[s] ago`	4 時間 1 分前 15 分前

抽出リスト

「リスト抽出」データ コンバータは、パターンに従ってテキストをフィルタして、すべての一致を連結したテキストを返します。

アドバンスド抽出データ コンバータでの照合がパターンの最初のインスタンスに限られるのに対して、このデータ コンバータは、一致がある場合にそれぞれの一致を連続して返します。

パターンを入力テキスト全体と照合する必要はないことに注意してください。これはアドバンスド抽出データ コンバータの要件となります。実際には、このデータ コンバータの最も多い用途は、明確に定義された各点で開始および終了するパターンを利用することです (つまり、大きな入力テキスト内に埋め込まれた特定のテキストに対して照合を実行する場合、パターンの開始と終了箇所に ".*" が含まれる必要はないことがほとんどです)。

プロパティ

「リスト抽出」データ コンバータは、次の各プロパティを使用して設定できます。

パターン
入力に対して照合されるパターンを入力します。

大文字と小文字を無視
このチェックボックスをオンにすると、パターンに対する照合は大文字と小文字の区別なしで実行されます。たとえば、"KoFaX" は "KOFAX" および "kofax" と同じであると見なされます。

出力エクスプレッション
出力値テキストを指定するエクスプレッションを入力します。

出力値区切り記号
任意のテキストを入力して、出力値テキスト内で連続するパターン一致を区切るための、区切り記号を示します。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

数値を抽出

このデータ コンバータは、番号を見つけて抽出し、それを標準的な番号の形式で出力します。

注意：すでに標準的な番号の形式になっているものを変更するには、数値を書式設定データ コンバータを代わりに使用します。

プロパティ

「数値を抽出」データ コンバータは、次の各プロパティを使用して設定できます。

フォーマット パターン
抽出する番号の形式を指定するパターンを含みます。デフォルト パターンの 1 つを使用するか、以下に示すパターン指定の詳細を参照してください。

小数点の記号
抽出する番号に使用できる小数点記号、たとえば "." を含みます。複数の区切り記号を指定できます。

桁区切り記号
抽出する番号に使用できる千の桁記号、たとえば "," を含みます。複数の区切り記号を指定できます。

マイナス記号
番号のマイナス記号 (通常は "-") として使用する文字を含みます。

次を乗ずる
抽出した番号に掛ける乗算の係数を指定します。

整数に変換
このフィールドのチェックボックスをオンにすると、抽出した番号が整数に変換されます。

定数
抽出する番号の前後に置かれることのある定数の定義を含みます。それぞれの定数について、名前 (たとえば、kilo) と値 (たとえば、1000) を定数の位置 (抽出する番号の前か後) とともに示します。定数の名前が正確に、抽出する番号の前または後に来る定数を示すように注意してください。たとえば、定数を kilo=1000.0 および double=2.0 と設定します。入力 "2 kilo" によって、番号 2000.0 が抽出されますが、入力 "2 double kilo" では double kilo という名前の定数がないため、番号 2.0 が抽出されます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

書式パターンを指定

書式パターンは、番号書式を指定する非常にフレキシブルな方法を提供します。ただし、パターンを指定するルールは理解しにくい場合があるため、必要なフォーマットに最適なデフォルト パターンを見つけて、そのデフォルト パターンを変更して試すのが、簡単な解決方法になる場合があります。

次の特殊文字をパターンに使用できます。

特殊文字	説明
0	数字。
#	ゼロ以外の数字が表示される。
.	小数点記号、すなわち小数点記号フィールドで指定される文字。
,	千の桁記号、すなわち千の桁記号フィールドで指定される文字。
-	マイナス記号、すなわちマイナス記号フィールドで指定される文字。
E	科学的な表記法では、対数と指数を区別する。

パターンでは、「小数点の記号」フィールドの入力内容にかかわらず、常に '.' の文字が小数点記号の選択に使用されます。番号の書式を変更すると、'.' の文字が小数点記号フィールドの文字に置き換えられます。同じことが千の桁記号とマイナス記号に当てはまります。

正の数と負の数に対して別々のパターンを指定することができます。セミコロン (';') で区切られた 2 つのパターンを指定することにより行います。たとえば、マイナス記号の文字を負の数の前に置くデフォルトではなく、負の数を括弧で囲みたい場合は、"#,##0.00;(#,##0.00)" というパターンを使用してください。

入力に大きな指数を伴う指数表記を使用する場合 (たとえば、値 6.023E23 など)、このような大きな数値を整数に変換すると不適切な結果が生じる可能性があるため、通常は、[整数に変換] をオンにしないでください。

例: 番号の抽出

この入力は次のように考えます。

```
Price is USD 33,555.77.
```

「フォーマット パターン」を「###0.0」、「小数点の記号」を「.」、「桁区切り記号」を「,」、「マイナス記号」を「-」、「次を乗ずる」を「1.0」、「整数に変換」のチェックをオフ、定数を設定しないに設定すると、番号 33555.77 が抽出されます。

上の例では、「整数に変換」のチェックボックスをオンにすると、番号 33556 が抽出されます。

ここでは、この入力を次のように考えます。

```
Price is USD 10.5 mill.
```

「フォーマット パターン」を「0.000」、「小数点の記号」を「.」、「桁区切り記号」を「,」、「マイナス記号」を「-」、「次を乗ずる」を「1.0」、「整数に変換」のチェックボックスをオン、定数を mill.=1000000.0 および bill.=1000000000.0 に設定すると、番号 10500000 が抽出されます。

上の例では、「整数に変換」のチェックボックスがオフになっている場合、番号 1.05E7 が抽出されます。

年を抽出

「年抽出」データ コンバータは、入力テキスト内の日付から年を抽出します。日付は、年しか含まれないなど、不完全な場合があります。

プロパティ

「年抽出」データ コンバータは、次の各プロパティを使用して設定できます。

ロケール

日付で使用するロケールを指定します。

データ フォーマット パターン

年を抽出する日付の形式を指定するパターンを含みます。構文の説明は以下を参照してください。

将来の月の最大値

これから先の月数の最大値です。このフィールドが有効になるのは、'dd' および 'MM' のパターンを使用するときで年の明示がない場合に限られます。

将来の日付の最大値

これから先の日数の最大値です。このフィールドが有効になるのは、'dd' および 'yyy' のパターンを使用するときで月の明示がない場合に限られます。

説明

データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

日付の形式パターンの構文

以下の各パターンを組み合わせて、「日付の形式パターン」プロパティ内のパターンを作成できます。

パターン	説明
yy	正確な 2 桁の年
yyy	年
yyyy	正確な 4 桁の年
YYYY[7]	正確な 4 桁の週年
MM	1 桁または 2 桁の月、月の略称、または完全な名前
dd	1 桁または 2 桁の日付
EEE	曜日の短い名前 (たとえば、Monday の代わりに Mon)。
EEEE	曜日の完全な名前 (たとえば、Monday)。
hh または HH	1 桁または 2 桁の時間
mm	1 桁または 2 桁の分
ss	1 桁または 2 桁の秒
a	AM または PM の表記
Z	タイム ゾーンの識別子 (たとえば、"PST"、"中央ヨーロッパ標準時"、または "GMT+02:00")
*	任意の文字数をスキップします (これは、英字 a ~ z および A ~ Z をスキップする場合に使用します)
スペース	1 つまたは複数の空白をスキップします
その他の英字以外の文字	その文字をスキップします (英字をスキップするには * を使用します)

曜日の各パターン ("EEE" および "EEEE") を使用して、日付パターン ("dd") を使用しない場合、月および年の各パターン ("MM" および "yy"/"yyy"/"yyyy") は使用できません。この場合、得られる日付は、次の、パターンに一致する名前が付いた日です。たとえば、パターンが 'EEEE' で、入力が次のテキストの場合:

水曜日

見つかった年は、次の水曜日の年です。

曜日パターンを日付パターンとともに (また、多くの場合、月および年パターンとともに) 使用する場合は、曜日を破棄します。たとえば、パターンが 'EEE, dd/MM/yyy' で、入力が次のテキストの場合:

見つかった年は、"2003" です。

'EEE' パターンは、曜日を示す短い名前に一致します (たとえば、Mon、Tue など)。パターンが曜日の名前の全体に一致する場合は、'EEEE' パターンを使用します。たとえば、入力が次のテキストの場合:

[7] 週年 (WEEK YEAR) を使用する場合、年の最初の日を含む週の年が新しい年として設定されます。

それでは、水曜日 (Wednesday) に会いましょう

'EEE' のパターンでは 'Thu' に一致し、「年を抽出」データ コンバータで次の木曜日の年が検知されるため、'EEEE' のパターンを使用する必要があります。

例: データ フォーマット パターン

以下に、形式パターンおよびマッチする日付を例示します。

データ フォーマット パターン	一致する日付
dd/MM-yyy	7/6/1978 24/12-2001 1-Jan-01
dd.MM yyy	4. jan 1993 4. january 93

日付を書式設定

このデータ コンバータは、日付の書式を再設定します。入力するテキストは、"2001-02-25 14:32:49.0" のような標準的な日付書式の形式にする必要があります。

注意：日付を標準的な形式に変換する場合は、日付抽出データ コンバータを使用します。

プロパティ

「日付を書式設定」データ コンバータは、次のプロパティを使用して設定できます。

ロケール
このフィールドは、日付を書式設定するロケールを指定します。

フォーマット パターン
日付書式を指定するパターンを含みます。デフォルト パターンの 1 つを使用するか、以下に示すパターン指定の詳細を参照してください。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

書式パターンを指定

書式パターンは、日付書式を指定する非常にフレキシブルな方法を提供します。ただし、パターンを指定するルールは理解しにくい場合があるため、単純に必要なフォーマットに最適なデフォルト パターンを見つけて、そのデフォルト パターンを変更して試してみた方が簡単かもしれません。

次の特殊文字をパターンに使用できます。

特殊文字	説明	タイプ	例
y,yy,yyy	年	数値	96
yyyy	年	数値	1996

特殊文字	説明	タイプ	例
YYYY[8]	週年	数値	2009
M	月	数値	7
MM	月	数値	7
MMM	月	テキスト	7 月
MMMM	月	テキスト	7 月
d,dd	日にち	数値	10
(d)ddd	日にち	数値	(0)010
E,EE,EEE	曜日	テキスト	火曜日
EEEE	曜日	テキスト	火曜日
D,DD,(D)DDD	年内日数	数値	(0)189
(F)F	月内曜日数	数値	(0)2 (7 月の第 2 週)
w,(w)ww	年内週数	数値	(0)27
(W)W	月内週数	数値	(0)2
(H)H	時刻 (0-23)	数値	(0)4 または (0)12
(k)k	時刻 (1-24)	数値	(0)4 または (0)12
(K)K	am/pm 時刻 (0-11)	数値	(0)0
(h)h	am/pm 時刻 (1-12)	数値	(0)5 または (0)12
(m)m	分	数値	(0)30
(s)s	秒	数値	(0)55
S,SS,(S)SSS	ミリ秒	数値	(0)978
a	am/pm マーカー	テキスト	PM
z,zz,zzz	タイム ゾーン	テキスト	PST
(z)zzzz	タイム ゾーン	テキスト	太平洋標準時
G	時代識別子	テキスト	AD
'	入力内にないテキストの最初/最後	区切り記号	'o''clock' -> o'clock
''	一重引用符	リテラル	"EEEE" ->「金曜日」

A-Z または a-z の範囲の文字ではないパターン内の文字はすべてテキストとして扱われます。たとえば、':'、'.'、' '、'#'、'@' などの文字は、一重引用符で囲まれていなくても結果の日付に表示されます。

パターンが空の場合は、選択したロケールのデフォルトの日付書式が使用されます。

例: 日付出力値

ロケールが「英語 (米国)」の場合の日付出力の例：

[8] 週年 (WEEK YEAR) を使用する場合、年の最初の日を含む週の年が新しい年として設定されます。

フォーマット パターン	出力データの例
yyyy.MM.dd G 'at' hh:mm:ss z	1996.07.10 AD at 15:08:56 PDT
EEE, MMM d, ''yy	Wed, July 10, '96
h:mm a	12:08 PM
hh 'o''clock' a, zzzz	12 o'clock PM, Pacific Daylight Time
K:mm a, z	0:00 PM, PST
yyyyy.MMMMM.dd GGG hh:mm aaa	1996.July.10 AD 12:08 PM

HTML を書式設定

このデータ コンバータは入力 HTML テキストを再び書式設定 (プリティプリント) します。

プロパティ

「HTML を書式設定」データ コンバータは、次のプロパティを使用して設定できます。

終了タグの不足を許容
選択すると、終了タグがオプションのタグ (<p> タグなど) は自動的に終了します。ステップから直接出力される HTML では、このオプションを選択する必要はありません。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

数値を書式設定

このデータ コンバータは番号の書式を再設定します。入力するテキストを、標準的な番号書式、たとえば、"12378.64" のような形式にする必要があります。

注意: 番号の書式を標準的な番号書式に変換する必要がある場合は、数値を抽出データ コンバータを使用します。

プロパティ

「数値を書式設定」データ コンバータは、次のプロパティを使用して設定できます。

フォーマット パターン
番号書式を指定するパターンを含みます。デフォルトパターンの 1 つを使用するか、以下に示すパターン指定の詳細を参照してください。

小数点の記号
番号に使用する小数点記号、すなわち、'.' または ',' のような、番号の整数部分と小数部分の間の区切り記号類です。

桁区切り記号
番号に使用する千の桁記号、すなわち、',' またはスペースのような、番号の整数部分にある千単位の区切り記号類です。

マイナス記号
番号のマイナス記号 (通常は "-") として使用する文字を含みます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

書式パターンを指定

書式パターンは、番号書式を指定する非常にフレキシブルな方法を提供します。ただし、パターンを指定するためのルールは理解しにくいこともあるため、必要なフォーマットに最適なデフォルトのパターンを見つけて、そのパターンを変更して試してみることをお勧めします。

次の特殊文字をパターンに使用できます。

特殊文字	説明
0	数字。
#	ゼロ以外の数字が表示される。
.	小数点記号、すなわち小数点記号フィールドで指定される文字。
,	千の桁記号、すなわち千の桁記号フィールドで指定される文字。
-	マイナス記号、すなわちマイナス記号フィールドで指定される文字。
E	科学的な表記法では、対数と指数を区別する。

パターンでは、「小数点の記号」フィールドに何が入力されているかにかかわらず、常に '.' の文字が小数点記号を選択するために使用されます。番号の書式を変更すると、'.' の文字が小数点記号フィールドの文字に置き換えられます。同じことが千の桁記号とマイナス記号に当てはまります。

正の数と負の数に対して別々のパターンを指定することができます。セミコロン (';') で区切られた 2 つのパターンを指定することにより行います。たとえば、マイナス記号の文字を負の数の前に置くデフォルトではなく、負の数を括弧で囲む必要がある場合は、"#,##0.00;(#,##0.00)" というパターンを使用することができます。

プロパティを取得

このデータ コンバータは変数に含まれているプロパティ リストからプロパティの値を取得します。

この変数は、プロパティ変数タイプである必要があります。

データ コンバータへの入力テキストは無視されます。

プロパティ

「プロパティを取得」データ コンバータは、次のプロパティを使用して設定できます。

プロパティ変数
プロパティのリストを含む変数。

プロパティ名
取得するプロパティの名前。

プロパティがない場合はデフォルト値を使用
プロパティが存在しない場合の処理を決定します。このオプションを選択すると、デフォルト値が代わりに使用されます。このオプションを選択されていないと、エラーが生成されます。

デフォルト値
プロパティが存在しない場合に使用するデフォルト値。デフォルト値を代わりに使用する必要があります。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

日付間の時間を取得

このデータ コンバータは、2 つの日付の差分を検出することを可能にします。入力テキスト内の日付を指定された日付と比較し、日付間の差分を計算します。

差分は、選択された単位、たとえば、日または週で計算されます。入力するテキストを、標準的な日付書式、たとえば、"2001-02-25 14:32:49.0" のような形式にする必要があります。

プロパティ

「日付間の時間を取得」データ コンバータは、次のプロパティを使用して設定できます。

他の日付
入力日付と比較する日付を指定します。日付は、値セレクターを使用してさまざまな方法で指定できます。日付を標準的な日付書式、たとえば、"2001-02-25 14:32:49.0" のような書式にする必要があります。

差分を取得する単位
差分を取得する単位を選択します。

差分を整数で取得
差分を整数に丸めるかどうかを指定します。

符号付きの差分を取得
差分を符号付きにするか符号なしにするかを指定します。差分を符号付きにする必要がある場合、入力日付が他の日付より後なら正、逆なら負となります。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
コンバータへの入力が "2008-03-01 12:00:00.0" で、他の日付が "2008-02-28 00:00:00:00.0" に設定され、差分を日数と分数/小数にする場合、結果は "2.5" になります。コンバータがうるう年をどのように考慮しているかについて注意してください。

変数を取得

このデータ コンバータは変数の値をフェッチします。入力テキストは無視されます。

プロパティ

「変数を取得」データ コンバータは、次のプロパティを使用して設定できます。

変数
取得する値が含まれた変数。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

If Then

If Then データ コンバータでは、コンバータの出力値を決定する if-then ルールのリストを指定することができます。

このリストは、複数の if 条件で構成されている場合があり、下端には常にデフォルト値を提供する 1 つの else ブロックがあります。

基本条件
条件タイプの「含む」、「含まない」、「先頭」、および「末尾」は、入力文字列が与えられた文字列を含んでいるか、与えられた文字列を含んでいないか、与えられた文字列で始まっているか、与えられた文字列で終わっているかのチェックをします。
条件タイプの「パターンと一致」および「パターンと不一致」は、入力文字列が特定のパターンと一致しているか、一致していないかのチェックをします。

基本条件のプロパティ

If Then データ コンバータの基本条件は、次のプロパティを使用して設定できます。

含む

含まない

先頭

末尾
入力テキストと照合する 1 つのテキスト値を入力します。

パターンと一致
パターンと不一致
これらのフィールドには、入力テキストと照合する 1 つのパターンを入力します。入力テキストの全体が入力パターンと一致するか/一致しないかの比較が行われることに注意してください。

テキスト値
上のプロパティの値が入力テキストと一致する場合は出力テキストを指定します。この値は、値セレクターを使用してさまざまな方法で指定できます (コンバータは使用しません)。

大文字と小文字を無視
オンにすると、第 1 のプロパティの値との照合は大文字/小文字の区別なしで行われます。たとえば、"KoFaX" は "KOFAX" や "kofax" に等しいとみなされます。

Else のプロパティ

If Then データ コンバータの Else のステートメントは、次のプロパティを使用して設定できます。

テキスト値
どの条件も入力テキストと一致しなかった場合に出力テキストを指定します。この値は、値セレクターを使用してさまざまな方法で指定できます (コンバータは使用しません)。このフィールドを空白のままにしておくと、If Then コンバータは空のテキストを返します。
If Matches Pattern において、Then 属性の式は、$n 表記を使用して、前にある If Matches Pattern フィールド内のパターンのサブマッチを参照することができます。
他のすべての条件の場合は、INPUT のキーワードを使用して入力テキストを参照することができます。

その他のプロパティ

If Then データ コンバータは、次のプロパティを使用して追加的に設定できます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが "911" のときに出力テキストを "Porsche 911" にする必要があるとします。あるいは、入力テキストが "911" 以外のときはそのままにしておくものとします。
この場合、If Then データ コンバータを次のように設定する必要があります。

- If Matches
  - If Matches: 911
  - その場合は (エクスプレッション): "Porsche " + $0
  - 大文字と小文字を区別しない：[オフにしておく]
- Else
  - その場合は (エクスプレッション): $0

絶対 URL に変換

「URL 絶対化」データ コンバータは、ロボットの現在の URL を使用して、相対 URL を絶対 URL に変換します。

プロパティ

「URL 絶対化」データ コンバータは、次のプロパティを使用して設定できます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
これらの例では、ロボットの現在の URL は http://www.kofax.com です。
- 入力テキストが「~hello」である場合、出力テキストは「http://www.kofax.com/~hello」になります。
- 入力テキストが「hello」である場合、出力テキストは「http://www.kofax.com/hello」になります。
- 入力テキストが「test1/test2/../test」である場合、出力テキストは「http://www.kofax.com/test1/test」になります。

相対 URL に変換

「URL 相対化」データ コンバータは、ロボットの現在の URL を使用して、絶対 URL を相対 URL に変換します。

プロパティ

「URL 相対化」データ コンバータは、次のプロパティを使用して設定できます。

ベース **URL**
入力 URL を相対化する URL。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
これらの例では、ロボットの現在の URL は http://www.kofax.com です。
- 入力テキストが「http://www.kofax.com/~hello」である場合、出力テキストは「~hello」になります。
- 入力テキストが「http://www.kofax.com/hello」である場合、出力テキストは「hello」になります。
- 入力テキストが「http://www.kofax.com/test1/test2/../test」である場合、出力テキストは「test1/test」になります。

日付を変更

このデータ コンバータは、日付の選択した部分に対して加算や減算を使用して日付を修正します。

加算や減算によって日付の選択した部分のオーバーフローやアンダーフローが生じる場合、それに従って、日付のその他の部分が更新されます。

入力するテキストは、"2001-02-25 14:32:49.0" のような標準的な日付書式の形式にする必要があります。

プロパティ

量
日付に加算したり減算したりする量。この量は、値セレクターを使用して指定します。値は整数である必要があります。

修正する日付の部分
加算または減算の対象となる日付の部分を選択します。

関数
その量を加算するか減算するか選択します。

タイム ゾーン
入力日付のタイム ゾーン。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
コンバータへの入力が "2008-02-28 10:45:00.0" で 2 日加算される場合、結果は "2008-03-01 10:45:00.0" になります。月がどのように更新されたか注目してください。2008 年のうるう日が考慮されました。

印刷不可文字を除去

[印刷不可文字を除去] データ コンバータはすべての印刷不可文字を除去します。

具体的には、以下の文字を除去します。
- ASCII 32 以下のすべての文字。ただし、タブ (#x0009)、ライン フィード (#x000A)、キャリッジ リターン (#x000D) は除きます。
- #xD800 ~ #xDFFF および #xFFFE ~ #xFFFF の範囲にあるすべての文字。

プロパティ

以下のプロパティを使用して [印刷不可文字を除去] データ コンバータを設定できます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

スペースを除去

このデータ コンバータは入力テキストからスペースを除去します。改行禁止スペース (HTML では   と表記される) はスペースとして扱われることに注意してください。

プロパティ

以下のプロパティを使用して [スペース除去] データ コンバータを設定できます。

すべてのスペースを除去
入力テキストからすべてのスペースを除去します。

先頭と末尾のスペースを除去
先頭と末尾のスペースがなくなるようにテキストをトリムします。

複数スペースを単一スペースに置き換え
入力テキスト内のすべての複数スペースを単一スペースに置き換えます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
入力テキストが次の場合：

```
"   hello   world    "
```

かつ、[複数スペースを単一スペースに置き換え] が選択されている場合、出力テキストは次のようになります:

```
" hello world "
```

[先頭と末尾のスペースを除去] が選択されている場合、出力テキストは次のようになります:

```
"hello   world"
```

[複数スペースを単一スペースに置き換え] と [先頭と末尾のスペースを除去] が選択されている場合、出力テキストは次のようになります:

```
"hello world"
```

かつ、[すべてのスペースを除去] が選択されていると、出力テキストは次のようになります:

```
"helloworld"
```

特殊文字を除去

[特殊文字除去] データ コンバータは入力テキスト内のすべての特殊文字をスペースに置き換えます。通常の文字、数字または数字の前に出現するカンマおよびピリオド以外のすべての文字は特殊文字と見なされ、スペースに置き換えられます。

[特殊文字除去] データ コンバータを適用した後、[スペース除去] データ コンバータを使用して、望ましくないスペースを除去することができます。

プロパティ

以下のプロパティを使用して [特殊文字除去] データ コンバータを設定できます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

タグ除去

[タグ除去] データ コンバータは入力テキストから HTML タグを除去します。

プロパティ

以下のプロパティを使用して [タグ除去] データ コンバータを設定できます。

これらのタグを除去
除去する必要のあるタグを指定します。
- 「すべてのタグ」はすべてのタグを除去するよう指定します。オプションで、テキスト内のアンパサンド エンコーディングを残します。
- 「選択されているタグ」は選択されているタグのみを除去するよう指定します。タグ名は "html,body" のように、カンマで区切ります。オプションで、テキスト内のアンパサンド エンコーディングを残します。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

パターンを置換

パターン置き換えデータ コンバータは、パターンの一致をエクスプレッションの結果に置き換えます。

プロパティ

以下のプロパティを使用してパターン置き換えデータ コンバータを設定できます。

パターン
入力テキストを対象として検索するパターンを指定します。このパターンを入力テキスト全体と照合する必要はないことに注意してください。

大文字と小文字を無視
このプロパティをオンにすると、パターン照合で大文字と小文字の区別が無視されます。

置換エクスプレッション
パターンと一致したテキストの部分を置き換える結果を生成するエクスプレッションを指定します。

すべてを置き換え
このプロパティをオンにすると、パターンのすべての出現箇所が [置換エクスプレッション] に置き換えられます。オフにした場合は、最初の出現箇所のみが置き換えられます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

テキストを置換

このデータ コンバータは入力テキスト内の一致するテキストを検索し、置き換えます。

プロパティ

以下のプロパティを使用して [テキスト置き換え] データ コンバータを設定できます。

このテキストを検索
入力テキストを対象として検索するテキスト。

このテキストに置き換え
置き換えに使用されるテキスト。

大文字と小文字を無視
このプロパティをオンにすると、テキスト照合で大文字と小文字の区別が無視されます。

すべてを置き換え
このプロパティをオンにすると、テキストのすべての出現箇所が新しいテキストに置き換えられます。オフにした場合は、最初の出現箇所のみが置き換えられます。

ワード全体のみを照合
このプロパティをオンにすると、大きいワードの一部ではなく、ワード全体の出現箇所のみに対してテキスト置き換えが行われます。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

テキストの引用符を除去

このデータ コンバータは、一重引用符または二重引用符で閉じられたテキストの引用符を削除します。テキスト内のエスケープ引用符は、エスケープされません。

プロパティ

「テキストの引用除去」データ コンバータは、以下のプロパティを使用して設定されます。

説明
データ コンバータのリストに表示する説明を入力します。説明がない場合は生成されます。

例
入力テキストが次の場合：

```
"Bob"
```

出力テキストは次のようになります：

```
Bob
```

入力テキストが次の場合：

```
"Robert \"Bob\" Jones"
```

出力テキストは次のようになります：

```
Robert "Bob" Jones
```

入力テキストが次の場合：

```
'Bob'
```

出力テキストは次のようになります：

```
Bob
```

入力テキストが次の場合：

```
Bob
```

出力テキストは次のようになります：

```
Bob
```

URL デコード

このデータ コンバータはすべての URL エンコーディングをデコードして、それらに対応する実際の文字に変換します。

エンコード済みの文字は、%HH の形式で表されます。この HH は、16 進数バイト値です。エンコード済みの文字はまず、この表記からバイトにデコードされます。そのバイトは、選択した文字エンコーディングを使用して文字に変換されます。

プロパティ

URL デコードのデータ コンバータは、以下のプロパティを使用して設定できます。

文字コード
バイトを文字に変換するのに使用する文字エンコーディング。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
選択した文字エンコーディングが UTF-8 である場合、入力テキストは以下のようになります。

```
x%2By%3Dz
```

データ コンバータは以下を出力します。

```
x+y=z
```

URL エンコード

このデータ コンバータは URL エンコーディングで文字をエンコードします。

エンコードされる必要のある文字はまず、選択した文字エンコーディングを使用してバイトに変換されます。そのバイトは %HH 形式で表されます。この HH は、16 進数バイト値です。

プロパティ

URL エンコードのデータ コンバータは、以下のプロパティを使用して設定できます。

文字コード
文字をバイトに変換するのに使用する文字エンコーディング。

説明
データ コンバータのリストに表示する説明を入力します。説明に入力がない場合、説明のないコンバータが生成されます。

例
選択したエンコーディングが UTF-8 である場合、入力テキストは以下のようになります。

```
x+y=z
```

データ コンバータは以下を出力します。

```
x%2By%3Dz
```

スニペット

スニペットは、複数のロボットで再利用できるステップのグループです。スニペットは、ロボットとは別のファイルで管理されます。1 つのロボットでスニペットの内容が変更されると、同じスニペットを使用する他のロボットで自動的に更新されます。スニペットはスニペット ステップを使ってロボットに挿入され、インラインで編集されます。ロボットに挿入されていないスニペットのコンテンツを編集することはできません。

ロボット内部のスニペット ステップは、多くの点でグループ ステップに似ています。グループ ステップ内のステップはロボットの一部ですが、スニペット ステップ内のステップは別のファイルに保持され、同じプロジェクト内の他のロボットで再利用することができます。ロボットが参照するスニペットがプロジェクトに存在しない場合は、ロボットは不完全となり、実行できません。

再利用可能なスニペットに変換するステップのグループを選択したら、「選択した範囲からスニペットを作成」をクリックします。1 つのグループ ステップのみが選択された場合、「グループをスニペットに変換」をクリックして、再利用可能なスニペットに変換することができます。スニペットは、スニペット ステップを選択した後、「スニペットをグループに変換」アイコンをクリックして、ロボットに簡単に埋め込むことができます。

スニペットは、スニペットを利用するロボットの変数一式に含まれる一連の変数も定義することができます。

スニペットに説明を付けることができます。この説明はスニペット エディターで編集し、ロボットでそのスニペットが実行されると表示されます。

ベーシック エンジン ロボットのアップグレード

この手順では、ベーシック エンジン ロボットを最新バージョンにアップグレードする方法について説明します。

1. プロジェクト ツリー ビューで、[ベーシック エンジン ロボット] を右クリックし、[アップグレード] をクリックします。
   - WebKit ベースのブラウザで作成されたベーシック エンジン ロボットは、青色の アイコンで識別されます。これらのロボットは、最新のバージョンにアップグレードすることができます。
   - クラシック ブラウザで作成されたベーシック エンジン ロボットは、オレンジ色の アイコンで識別されます。これらのロボットは、WebKit ベースのブラウザを使用するようにアップグレードすると、アイコンが青色に変わります。

     クラシック ブラウザが削除されたため、このブラウザを使用して新しいロボットを作成したり、Design Studio で既存のロボットを実行したりすることはできなくなったことに注意してください。RoboServer バージョン 11.5.0 以降では、このようなロボットを実行することもできません。

     ロボットを正常に実行するには、ロボットをアップグレードする必要があります。アップグレード後に、ロボットを設計および編集できるようになります。ロボットに赤い感嘆符が付いているかどうかを確認します。これは、ロボットに非推奨のアクション ステップとデータ コンバータが含まれている可能性があることを示します。一部のアイテムを、現在の Kofax RPA のバージョンに関連する同様のアイテムに置き換えることができます。置き換えるアイテムがない場合は、ロ

ボットを再設計する必要があります。詳細については 廃止されたアクション ステップとデータ コンバータ を参照してください。

ℹ または、最上位のフォルダを選択し、[アップグレード] をクリックすることで、クラシックと WebKit の両方のベーシック エンジン ロボットをすべて一度にアップグレードできます。

もう 1 つのオプションは、ロボットをダブルクリックして、Design Studio で優先するベーシック エンジン ロボットを開くことです。アップグレードが必要な場合は、黄色の警告メッセージが表示されます。ロボットをアップグレードするには、[アップグレード] をクリックします。

[アップグレード] ダイアログ ボックスが表示されます。

2. 表示されたリストにアップグレードするアイテムが含まれていることを確認し、[次へ] をクリックして続行します。
3. ウィザードにより、ロボットとそれに関連付けられたスニペットをバックアップします。バックアップ ファイルを格納するフォルダを選択し、[終了] をクリックしてアップグレードを開始します。

廃止されたアクション ステップとデータ コンバータ

以下の表に、廃止されたアクション ステップとその代替可能なステップを示します。

廃止されたアクション ステップ	代替として使用
REST Web サービス呼出 (旧バージョン)	REST Web サービス呼出
Web サービスの呼び出し (旧バージョン)	SOAP Web サービス呼出
ページをクロール	代替なし
ページをクロール (旧バージョン)	代替なし
ページ生成 (旧バージョン)	ページ生成
テーブル分割	代替なし
タグの分割	代替なし
テキスト分割 (旧バージョン)	テキスト分割
SQL 実行 (旧バージョン)	SQL 実行
Cookie 抽出 (旧バージョン)	Cookie 抽出
CSV 抽出 (旧バージョン)	ファイルをページとしてロードし、各フィールドに通常のループとエクストラクタを使用します。
Excel から抽出 (旧バージョン)	セル値抽出
PDF から抽出 (旧バージョン)	PDF から抽出
JSON 抽出 (旧バージョン)	JSON 抽出
タグ属性抽出 (旧バージョン)	タグ属性抽出
PDF からテキスト抽出 (旧バージョン)	PDF から抽出
タグの検索	代替なし
ファイル繰り返し (旧バージョン)	ファイル繰り返し

廃止されたアクション ステップ	代替として使用
URL 繰り返し (旧バージョン)	URL 繰り返し
XML タグ繰り返し	タグ繰り返し
戻る (レガシー)	代替なし
進む	代替なし
データのロード (旧バージョン)	ページ読込 および バイナリ コンテンツ抽出
ページのロード (旧バージョン)	ページ読込
ループ フォーム	代替なし
データベース照会 (旧バージョン)	データベース照会
HTTP 通信 (古いバージョン)	HTTP 通信
Cookie 除去 (旧バージョン)	Cookie 除去
タグ除去 (旧バージョン)	タグ除去
タグの復元	代替なし
タグの保存	代替なし
複数オプション選択 (旧バージョン)	複数オプション選択
トップ タグの設定 (旧バージョン)	代替なし
フォーム送信	クリック
タグ判定 (旧バージョン)	タグ判定
URL 判定 (旧バージョン)	URL 判定
ログ出力 (旧バージョン)	ログ出力

以下の表に、廃止されたデータ コンバータとその代替となる可能性のあるデータ コンバータを示します。

廃止されたデータ コンバータ	代替として使用
JSON に追加 (旧バージョン)	代替なし
XML に追加	代替なし
高度な抽出 (旧バージョン)	アドバンスド抽出
タグをカウント (旧バージョン)	カウント タグ
エクスプレッションを評価 (旧バージョン)	エクスプレッションを評価
日付抽出 (旧バージョン)	日付抽出
抽出元	抽出
リストを抽出 (旧バージョン)	抽出リスト
If Then (旧バージョン)	If Then
パターンを置き換え (旧バージョン)	パターンを置換

ベーシック エンジン ロボットの設定

ロボットを構成するには、次に説明する各プロパティを使用します。**[ロボットの設定]** ウィンドウは、ツールバーの **[ロボット設定]** をクリックするか、Ctrl+R を押すと表示されます。また、**[ファイル]** メニューから **[ロボット設定]** を選択することもできます。

[基本] タブ

デフォルト オプション

ロボットのステップに対するデフォルト オプションを設定します。詳細については、ベーシック エンジン ロボットの設定のデフォルト オプションを参照してください。

ロボット コメント

ロボットについてのコメントを入力します。

ロボットのサムネイル

ロボットの画像を追加します。

[ロード] をクリックして、必要なファイルがあるフォルダに移動します。推奨される画像タイプは PNG です。画像をアップロードすると、自動的に 20x20 ピクセルにスケーリングされます。ファイルを開き、**[OK]** をクリックして、ロボットを保存します。

異なるタイプのファイルをアップロードしようとすると、エラー メッセージが表示されます。

画像を削除するには、**[ロボットの設定]** ウィンドウを開き、ロボットのサムネイルの近くにあるアスタリスクをダブルクリックします。ロボットのサムネイルがデフォルトとして設定され、プレビュー ウィンドウに「画像なし」というテキストが表示されます。**[OK]** をクリックしてロボットを保存します。

ロボット タグ

ロボット用に 1 つ以上のタグを作成します。タグは、Management Console の [リポジトリ] > [ロボット] ページにある [タグ] 列に表示されます。タグを使用して、Management Console にあるロボットのリストをフィルタリングできます。タグには文字、数字、および下線を含めることができます。タグには 255 文字を使用できます。255 文字以上の文字を入力すると、最初の 255 文字のみが保存されます。同一のタグを 2 つ入力すると、赤い警告アイコンが表示されます。

手動の処理時間

このオプションでは、選択したロボットによって実行時に行われるタスクをユーザーが実行する場合の所要時間を分単位で指定できます。Kofax Analytics for RPA の [概要] レポートの「節約された人手の処理時間」テーブルに、指定した値とロボットの実際の実行時間の差が表示されます。

[詳細] タブ

デフォルト待機

ロボットのデフォルトの待機基準を指定します。待機基準とそのオプションについての情報は、待機基準の使用を参照してください。

プライベート **HTTP** キャッシュを有効化

プライベート HTTP キャッシュを有効にするには、このオプションを選択します。Cache-Control: private とマークされた、サーバーから受けたページには、特定のクライアントに特有の情報が含まれ、

これらはグローバル HTTP キャッシュ内に保存されません。このようなページを絶対にキャッシュしないためには、このオプションを無効にしてください。このようなページをロボット専用のキャッシュに保存するには、このオプションを有効にしてください。プライベート HTTP キャッシュ有効化の欠点は、ロボット 1 つあたりのメモリー使用量が増えることです。同一サーバーで大量のロボットを実行している場合は、このオプションを無効にするとメモリー領域を節減できます。

プライベート **HTTP** キャッシュ サイズ

このプロパティでは、プライベート HTTP キャッシュに使用する最大メモリー量を指定します。サイズはキロバイトで指定します。この数値は高く設定するように注意してください。クラウドで実行中のロボット インスタンスのそれぞれが、他の状態に加えて、このメモリー量を潜在的に使用するからです。HTTP キャッシュに保存したすべてのページは圧縮されます。そのため、テキスト コンテンツのシンプルなページでは、必要なメモリーはごくわずかです。また、Cache-Control: private などを含むページだけが、常にプライベート HTTP キャッシュに保存されます。非プライベートキャッシュとしてマークしたページは、すべてのロボットで共有されているグローバル HTTP キャッシュに保存されます。

プロキシ サーバー

プロキシ サーバーを使用

このオプションは、ロボット のプロキシ設定には影響しません。ロボットのプロキシ設定を指定するには、「プロキシを構成する」を参照してください。グローバル プロキシ設定を指定するには、「プロキシ サーバー」を参照してください。

この特定のロボットが行うすべてのページおよびデータの読み込みに使用する、オプションのプロキシ サーバーを指定します。このプロパティを使用する機会はわずかです。Kofax RPA のインストールの全体に対して 1 つまたは複数のプロキシ サーバーを指定することをお勧めします。[Design Studio 設定] ウィンドウから指定できます。詳細については、プロキシ サーバーを参照してください。特定のロボットに対して指定したプロキシサーバーは、Design Studio 設定で指定したプロキシ サーバーを無効にします。

デザイン モード

デフォルト (WebKit) ブラウザ エンジンを使用するロボットに対して [デザイン モード実行] を選択します。使用できるオプションは以下のとおりです。

- 最小実行 (ダイレクト)
- スマート再実行 (フル)

[外部の再実行を回避] オプションは、[スマート再実行] で利用でき、前の実行のキャッシュ結果を使用できない場合であっても、ステップを絶対に再実行しないようにします。このオプションを使用できるのは、外界との相互作用で再実行を回避する必要がある場合に限られます。たとえば、これによってパートナーのシステムでデータの相違や重複が起きるような場合です。

バージョン

保存されたロボットのバージョンおよびロボットの編集を最後に行った Design Studio のバージョンを表示します。

ベーシック エンジン ロボットの設定のデフォルト オプション

[ロボット設定] -> [基本] -> [デフォルト オプション] ダイアログ ボックスで、ベーシック エンジン ロボットのデフォルト オプションを設定できます。

[すべてのローディング] タブ

このタブには、一般的な読み込みのプロパティが含まれ、ページ読み込みとその他のタイプの読み込みに使用されます。

クレデンシャル

デフォルト: スタンダード

クレデンシャルとして、標準のユーザー名/パスワードのクレデンシャルか、OAuth クレデンシャルのいずれかを使用できます。

[スタンダード] を選択する場合は、次のプロパティを使用できます。

ユーザー名

このプロパティは、ログインに使用するユーザー名を指定します。値セレクターを使用してさまざまな方法で値を指定することができます。このユーザー名は、HTTP および FTP ベースのログインにのみ使用されることに注意してください。これらのログイン タイプでは、通常、ブラウザでポップアップ ウィンドウのプロンプトが開き、より一般的に使用されるフォーム ベースのログイン方法とは異なります。

パスワード

このプロパティは、ログインに使用するパスワードを指定します。値セレクターを使用してさまざまな方法で値を指定することができます。このパスワードは、HTTP および FTP ベースのログインにのみ使用されることに注意してください。これらのログイン タイプでは、通常、ブラウザでポップアップ ウィンドウのプロンプトが開き、より一般的に使用されるフォーム ベースのログイン方法とは異なります。

詳細については、Web 認証を参照してください。

代わりに、[**OAuth**] クレデンシャルを使用することができます。OAuth は、よく利用される REST API の多くで好まれる認証メカニズムです。Design Studio および Management Console での OAuth の使用方法については、OAuth を参照してください。

クライアント証明書

デフォルト: 自動

このプロパティは、HTTPS URL から読み込まれる時にどこでクライアント証明書を取得するか定義します。クライアント証明書は、直接与えることもできますし、「HTTPS クライアント証明書」で説明されているようにインストールされた証明書の 1 つを参照することもできます。以下のオプションがあります。

- 自動: 「デフォルト」とマークされた、インストール済みの証明書の 1 つを選択します。証明書がインストールされていない場合、またはインストールされたどの証明書も「デフォルト」としてマークされていない場合、接続にクライアント証明書は使用されません。
- インストール済み証明書: インストール時に定義された ID を指定することで、インストール済みの証明書の中の 1 つを選択します。
- 変数からの証明書: バイナリ変数の値として証明書を指定します。証明書のパスワードも、別の変数の値として指定する必要があります。証明書ファイルに複数の証明書が含まれている場合は、[エイリア

ス] フィールドで必要な証明書を選択します。デフォルト値の [なし] の場合、証明書のエントリが指定されず、選択が自動的に行われます。

- 変数からの **ID**: 変数の値として ID を指定し、インストールされた証明書の 1 つを選択します。

SSL/TLS 暗号化プロトコル

デフォルト: セキュアな TLS

このプロパティは、HTTPS URL からロードするときに使用するモードを指定します。これは、最新の TLS プロトコルがサポートされていない場合、あるいはクライアントが提供する脆弱なプロトコルが受け入れない場合などに、使用する SSL/TLS バージョンに応じてさまざまな結果が生成されるサイトがあるため、設定が必要です。次の 2 つのオプションを利用できます。

- セキュアな **TLS**: 安全なプロトコルと暗号のみを使用するように、暗号化された接続を制限します。このオプションは、TLS 1.3 および TLS 1.2 をサポートします。
- セキュリティの低い **TLS**: 制限を適用しないで、サポートされているすべてのプロトコルと暗号を使用します。このオプション は、TLS 1.3、TLS 1.2、TLS 1.1、TLS 1.0、および SSL3 をサポートします。

SSL3、TLS 1.0、および TLS 1.1 は、ベーシック エンジン ロボットの「REST Web サービス呼出」および「SOAP Web サービス呼出」ステップではデフォルトでブロックされます。プロトコルを有効にするには、C:\Program Files\Kofax RPA [バージョン]\jre\conf\security\java.security に移動し、`jdk.tls.disabledAlgorithms` パラメータから `SSLv3, TLSv1, TLSv1.1` という値を削除します。

```
jdk.tls.disabledAlgorithms=SSLv3, TLSv1, TLSv1.1, RC4, DES, MD5withRSA, DH keySize
 < 1024, \
EC keySize < 224, 3DES_EDE_CBC, anon, NULL, \
include jdk.disabled.namedCurves
```

SSL 証明書を検証

デフォルト: 選択済み

このオプションを選択すると、ロボットは、提示される SSL 証明書を検証します。

セキュリティ上の理由から、データの漏洩やロボットのパフォーマンスの問題を防ぐためのセーフガードとして、このオプションを常に選択しておくことをお勧めします。

認証方法

デフォルト: NTLM

使用する認証プロトコルを選択します。[**NTLM**] および [ネゴシエート] から選択できます。[ネゴシエート] を選択した場合は、特定のネゴシエート プロトコル パラメータを追加できます。詳細については、Web 認証を参照してください。

HTTP User Agent

デフォルト: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (Gecko のような KHTML) Chrome/43.0.2357.134 Safari/537.36

このプロパティは、HTTP の User-Agent ヘッダの値として送信する正確なテキストを指定します。変数から値をランダムに取得するような形で User-Agent ヘッダを変化させると、リモートの Web サーバーへの他のリクエストにうまく馴染ませることができます。

言語

デフォルト: 英語 (アメリカ合衆国) (en_US)

このプロパティは、JavaScript による問い合わせ時および読み込みの実行時に、どのブラウザ言語が表示されるか指定します。

画面サイズ

デフォルト: 1280 x 1024

このプロパティは、JavaScript によって問い合わせられる場合に表示される画面サイズを指定します。

この **URL** から参照

デフォルト: 未指定

このプロパティは、何かを読み込む時に、どこからアクションが参照されたように表示させたいか URL で指定します。URL を指定しない場合、アクションは、ロボットの現在のページから参照されて表示されます。

Cookie を有効化

デフォルト: 選択済み

このプロパティは、Cookie を有効にするかどうかを指定します。

HTTP キャッシュ

デフォルト: 有効

このプロパティは、ロボットにどのように HTTP キャッシュを使用させるか指定します。

- 有効にする: HTTP キャッシュが有効になり、HTTP キャッシュのルールに基づいて HTTP レスポンスをキャッシュします。
- 無効：HTTP キャッシュを無効にします。
- **[アグレッシブ]** は、キャッシュ ディレクティブを上書きし、上書きしなければキャッシュされないリソースのキャッシュが有効化されます。[アグレッシブ] オプションは、レイテンシの大きなサイトのパフォーマンスを高めるのに便利な場合があります。

最大試行回数

デフォルト: 1

このプロパティは、読み込みエラーが発生する場合に、アクション実行を試行する回数を指定します。最小値は「1」です。

試行間隔 **(**秒**)**

デフォルト: 5.0

このプロパティは、アクションの各実行の間で待機する秒数を指定します。

試行タイムアウト **(**秒**)**

デフォルト: 60.0

このプロパティは、タイムアウト前にアクション実行の各試行が許容される秒数を指定します。値はゼロより大きい必要があります。

送信する追加ヘッダー

デフォルト: リストから

このプロパティにより、オプションの変数または送信する追加の HTTP ヘッダーを含むリストを指定します。ヘッダは、HTTP メッセージと同じ形式のテキストで表示される必要があります。

受信したステータス コードをここに保存

デフォルト: (なし)

このプロパティは、受け取る HTTP レスポンス ステータス コードを格納するオプション変数を指定します。コードは整数であり、受け取るヘッダが取得される同じレスポンスに対応します。

受信したヘッダーをここに保存

デフォルト: (なし)

このプロパティは、受け取る HTTP レスポンス ヘッダを格納するオプション変数を指定します。ヘッダは、HTTP メッセージと同じ形式のテキストで表示されます。

ロード エラーを無視

デフォルト: 未選択

このプロパティは、ページやリソースの読み込みが失敗する時のエラーを無視するかどうか指定します。

[ページ ローディング] タブ

このタブには、ページの読み込みに特に使用されるプロパティが含まれます。

ページ コンテンツ タイプ

デフォルト: 自動

このプロパティは、読み込まれるページのコンテンツ タイプを指定します。通常、[自動] 設定で十分ですが、URL に応じて、アクションで読み込まれる全ページ、またはその一部に対してのみ、直接コンテンツ タイプを指定することもできます。

メタ リダイレクトに従う

デフォルト: 選択済み

このプロパティは、<meta> タグのリダイレクション、つまり読み込まれるページの <meta> によって定義されるリダイレクションに従うかどうか指定します。

フレームをロード

デフォルト: 選択済み

このプロパティは、ページのフレームを自動的に読み込むかどうか指定します。

ロードするイメージ

デフォルト: なし

このプロパティは、ページの画像を自動的に読み込むかどうか指定します。通常、ロボットは画像を読み込む必要はありませんが、画像の読み込みにページ ナビゲーションに必要な副次的効果があると考える場合は、ページの画像の読み込みを選択できます。その場合、URL に応じて、ページのすべての画像を読み込むか、その一部を読み込むか選択できます。

ページの変更

デフォルト: (なし)

このプロパティを使用すると、読み込まれたページを、解析前にオンザフライで修正することができます。これは、文法エラーの修正、解析に関するその他の問題の解決、タグ除去や変更などの実行に便利です。

この変更は、解析前にページに適用する 1 つ以上のデータ コンバータを指定して実行されます。すべてのページに適用するデータ コンバータか、URL に応じて個々のページに適用するデータ コンバータのいずれかを指定することができます。

ページの変更に使用する最も一般的なデータ コンバータは、テキスト置き換え、パターン置き換え、タグ除去です。データ コンバータを設定するときは、アンパサンド エンコードなどのデコードの前に、

データ コンバータがページの元の未処理のテキストに適用されることに注意してください。したがって、標準ブラウザのソース表示機能などを使用して、このテキストを取得することをお勧めします。データ コンバータ設定ウィンドウで、左下隅の入力領域にテキストを貼り付けて、コンバータでテキストに対する目的のアクションが実行されるかどうかをテストできます。

JavaScript を変更する場合は、代わりに [JavaScript 実行] タブの [JavaScript 変更] プロパティを使用します。

タイムアウトの場合はページを出力

デフォルト: 選択済み

このプロパティは、アクションがタイムアウトした時に実行される内容を指定します。無効になっている場合、イベントがタイムアウトしてアクションが失敗します。有効になっている場合、それまでに受信された内容が出力値となります。このプロパティに関して、次の内容に注意してください。

- このプロパティのデフォルトが "false" である古いデフォルト ブラウザ (WebKit) ロボットを新しいバージョンの Kofax RPA で開くと、ロボットのブラウザ構成で [タイムアウトの場合はページを出力] が false に設定されます。
- 新しいデフォルト ブラウザのロボットを作成する時は、[タイムアウトの場合はページを出力] プロパティはデフォルトで true に設定されます。

[URL フィルタリング] タブ

このタブは、広告フレームの読み込みをブロックするといった、ブロックする URL の設定を管理します。このタブの設定は、ベーシック エンジン ロボットに対してのみ有効です。

URL をフィルタリング

デフォルト: 選択済み

このプロパティは、特定の URL の読み込みをブロックするかどうか指定します。ブロックされる URL は、「含まれている URL パターン」および「除外された URL パターン」のリストで、それぞれパターンとして指定されます。次のタグで生じる URL のみブロックされる可能性があります。

含まれている **URL** パターン

デフォルト: 未指定

指定されると、これらのパターンに一致する URL のみブロックされなくなります。各パターンは別の行に記述される必要があります。以下で指定される [ブロックする URL のパターン] の 1 つと一致する場合、これらのパターンの 1 つと一致する URL はブロックされます。このプロパティは、通常、単一のドメインの URL のみ一致するパターンを指定して、そのドメインのフレームとスクリプトのみ読み込まれるようにするために使用されます。

- `<frame src="URL">`
- `<iframe src="URL">`
- `<script src="URL">`

URL がブロックされると、リクエストは実行されず、コンテンツは空のまま残されます。フレームや iframe の場合、それでもページに新しいウィンドウが表示され、読み込みが実行されなかった理由を説明するメッセージが示されます。ページ ビューのウィンドウのタブにある赤いブロック アイコンは、URL がブロックされたことを示します。

ブロックする **URL** のパターン

デフォルト: [デフォルトに設定] をクリックして、ブロックする URL のデフォルト リストを表示します。

このプロパティは、ブロックする URL を指定します。これは、各行にパターンを 1 つ記載したパターン リストを作成して指定されます。

[JavaScript の実行] タブ

このタブには、JavaScript の実行に使用されるプロパティが含まれます。これらのプロパティを使用すると、デフォルトの自動実行が正しく動作しない場合に、JavaScript 実行をカスタマイズできます。

JavaScript の実行

デフォルト: 選択済み

このプロパティによって、JavaScript を実行するかどうか指定します。

アラート メッセージを無視

デフォルト: 未選択

一般的には、ロボットは、このプロパティを設定してアラートを無視して、以下の [無視されたアラート メッセージをここに保存] プロパティを、無視されたアラート メッセージが適切な変数に格納されるように設定し、アラート メッセージを処理します。その後のステップで、この変数をテストして、アラート メッセージが含まれている場合は適切なアクションを実行できます。

無視されたアラート メッセージをここに保存

デフォルト: (なし)

このプロパティは、無視されたアラート メッセージが格納される変数を指定します。これは、上記の [アラート メッセージを無視] オプションが選択されている時のみ関係します。

キー プレス間の遅延 **(ms)**

デフォルト: 0

このプロパティは、キーボードでのユーザー入力を エミュレートする時にキー プレス間の待機時間をミリ秒で指定します。これは、フォームにテキストを入力するステップにのみ関係します。

JavaScript 変更

デフォルト: 未指定

[JavaScript 変更] は、実行前に JavaScript に適用されるデータ コンバータのオプションのリストです。変更は、実行されるすべての JavaScript のイベント ハンドラ、内部および外部のスクリプトに対して適用されます。データ コンバータは、たとえば、サポートされていない言語構造を使用するブラウザで JavaScript が正しく動作しないときに、JavaScript に変更や修正を加える場合に便利です。

データ コンバータを設定する時は、元の JavaScript に適用されることに注意してください。このため、インライン JavaScript の場合は標準ブラウザのソース表示機能を使用したり、外部 JavaScript の場合はそのファイルをダウンロードしたりして、このテキストを取得することが推奨されます。データ コンバータ設定ウィンドウで、左下隅の入力領域にテキストを貼り付けて、コンバータでテキストに対する目的のアクションが実行されるかどうかをテストできます。

このオプションは、ロードするページで JavaScript が実行される方法に影響します。

JavaScript ポリフィル

デフォルト: 未指定

デフォルトの Kofax RPA ブラウザ (WebKit) は、最新のステートメント、組み込み関数、オブジェクトなどの JavaScript 機能の一部をサポートしていない可能性があります。新しい機能のサポートを有効にするために、Kofax RPA では事前定義済みまたはカスタムの JavaScript ポリフィルをロードすることができます。

ポリフィルは、最新の機能をネイティブにサポートしていないブラウザに最新機能を提供するコード (通常は Web 上の JavaScript) です。

エラーが発生した場合、JavaScript コンソールには、存在しない JavaScript オブジェクトが表示されます。この情報に従って、必要なポリフィルが見つかり、適用して エラーを解決できます。

[追加] (+) をクリックして、ブラウザでサポートするオブジェクトまたは API を選択します。また、特定の JavaScript オブジェクトまたは API をサポートするカスタム コードを含めることもできます。カスタムの実装を含めるには、[追加] (+) をクリックし、リストから [カスタム] を選択します。[カスタム] ダイアログ ボックスには、[名前] と [コード] の 2 つのペインが含まれています。[名前] ペインでコード実装の名前を指定し、[コード] ペインに JavaScript コードを貼り付けます。

JavaScript オブジェクトの実装コードは、ページがロードされる前に実行されます。

事前定義済み JavaScript ポリフィルの事前定義済みポリフィルのリストを参照してください。

しかし、最新 JavaScript 構築のなかにはポリフィルを適用しても エラーが解決されない場合が多くあります。Kofax RPA の制限にある JavaScript の既知の問題のリストを参照してください。

プラグイン タブ

このタブには、ブラウザを使用する時にシミュレートされるプラグインを追加して設定するパラメータが含まれます。

サポート シミュレーション
デフォルト: リストから

- [リストから]: + 記号をクリックしてリストからプラグインを選択します。
- [**JSON** 変数値から]: JSON 変数を使用して独自のプラグインをコンストラクトします。
  詳細については、JSON 変数値からのプラグイン シミュレーション を参照してください。

[レガシー タブ]

このタブには、ほとんどの場合、変更されないプロパティが含まれます。レガシー プロパティは、古いバージョンの製品との後方互換性を保証するために実装されています。新しい機能が製品に導入され、以前実行されていた方法と競合する場合、オプションがこのタブに追加され、古い ロボットの動作と後方互換性が保証されます。このタブでは、デフォルト設定は最新の方法を示し、その他の設定は古い方法を示します。

フォーマット処理

デフォルト: 非 HTML をダウンロード

このオプションは、さまざまなドキュメント形式をどのように処理するか指定します。

非 **HTML** をダウンロード
Kofax RPA は、使用するすべてのサポートされる非 HTML コンテンツを読み込みます。CSV、JSON、テキスト、Excel、XML、およびバイナリのコンテンツをプレビューして、これらにステップを適用できます。[プレビュー] ボタンを使用して、コンテンツのタイプを変更します。

クラシック ローディング
従来のブラウザ エンジンに対して各種ドキュメント形式をどのように処理するか指定できます。

JSON

デフォルト: 変換しない

このプロパティは、Web サービスを呼び出す時に、一般的なレスポンス タイプの 1 つである JSON の処理方法を指定します。JSON を XML に変換できるため、標準的な方法を用いて Design Studio で簡単に処理できます。さらに、HTML に変換することもできます。HTML は XML よりも読みやすい形式ですが、自動抽出には向きません。

XML から HTML へ変換

デフォルト: 未選択

このプロパティは、XML ドキュメントを HTML ドキュメントに変換するか、そのまま維持するか指定します。これは以前は唯一のオプションであったため、変換されたドキュメントで動作する古いロボットで主に使用されます。新しいロボットは、通常、XML 構造で直接動作させる方が便利です。

XSLT 変換が適用されている XML コンテンツをアプリケーション ビューで表示するには、**[ロボット設定] > [デフォルト オプション] > [設定] > [レガシー] > [フォーマット処理] > [クラシック ローディング]** を選択し、[**XML から HTML へ変換**] オプションをオフにします。

Excel から HTML へ変換

デフォルト: 未選択

このプロパティは、Excel ドキュメントを HTML ドキュメントに変換するか、そのまま維持するかを指定します。これは以前は唯一のオプションであったため、変換されたドキュメントで動作する古いロボットで主に使用されます。より高速でスプレッドシートに近い表示やユーザー インターフェイスを提供するため、新しいロボットは、通常、Excel ドキュメントで直接動作させる方が便利です。

CSV

デフォルト: HTML へ変換

このプロパティは、CSV ドキュメントを HTML ドキュメントに変換するか、そのままテキストとして維持するかを指定します (PRE タグ)。これは以前は唯一のオプションであったため、テキスト表現で動作する古いロボットで主に使用されます。新しいロボットは、通常、CSV の HTML テーブル表現で直接動作させる方が便利です。そうすることで、Design Studio のすべての能力を利用できます。CSV ドキュメントは、カンマ (,) を区切り文字、二重引用符 (") を引用文字、バックスラッシュ (\) をエスケープ文字として使用して、エンコードされます。読み込むドキュメントがこの表記法に従っていない場合、Convert to Text オプションを使用し、「CSV 抽出」ステップなどによりテキストとしてドキュメントで作業する必要があります。

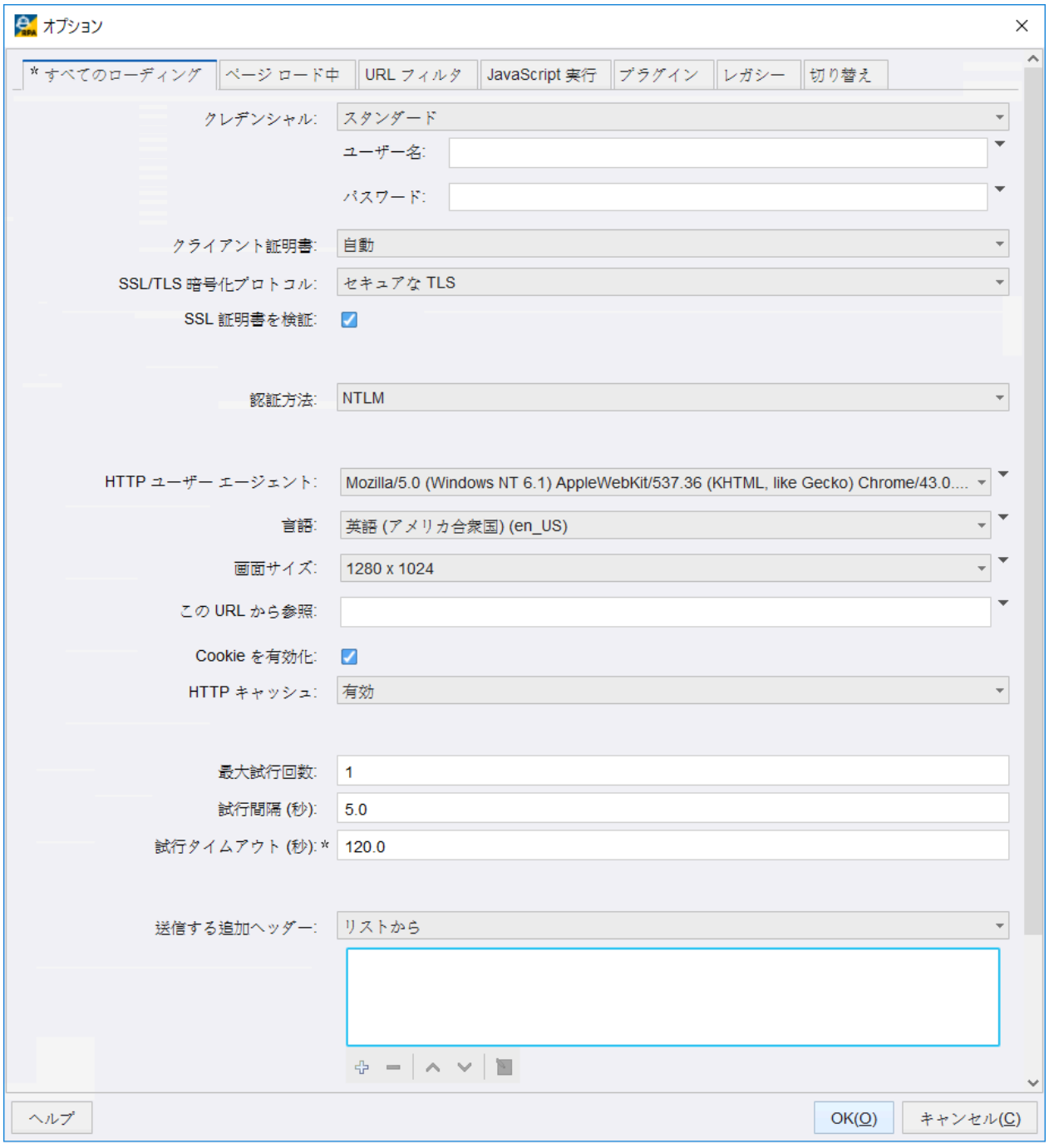
切り替えタブ

このタブには、クラシック ブラウザから WebKit (デフォルト) ブラウザにアップグレードされたロボットの切り替えログが含まれます。このログには、アップグレード中に変更されたすべての値がリストされます。詳細については ベーシック エンジン ロボットのアップグレード を参照してください。

デフォルトの口ボット設定から変更を表示

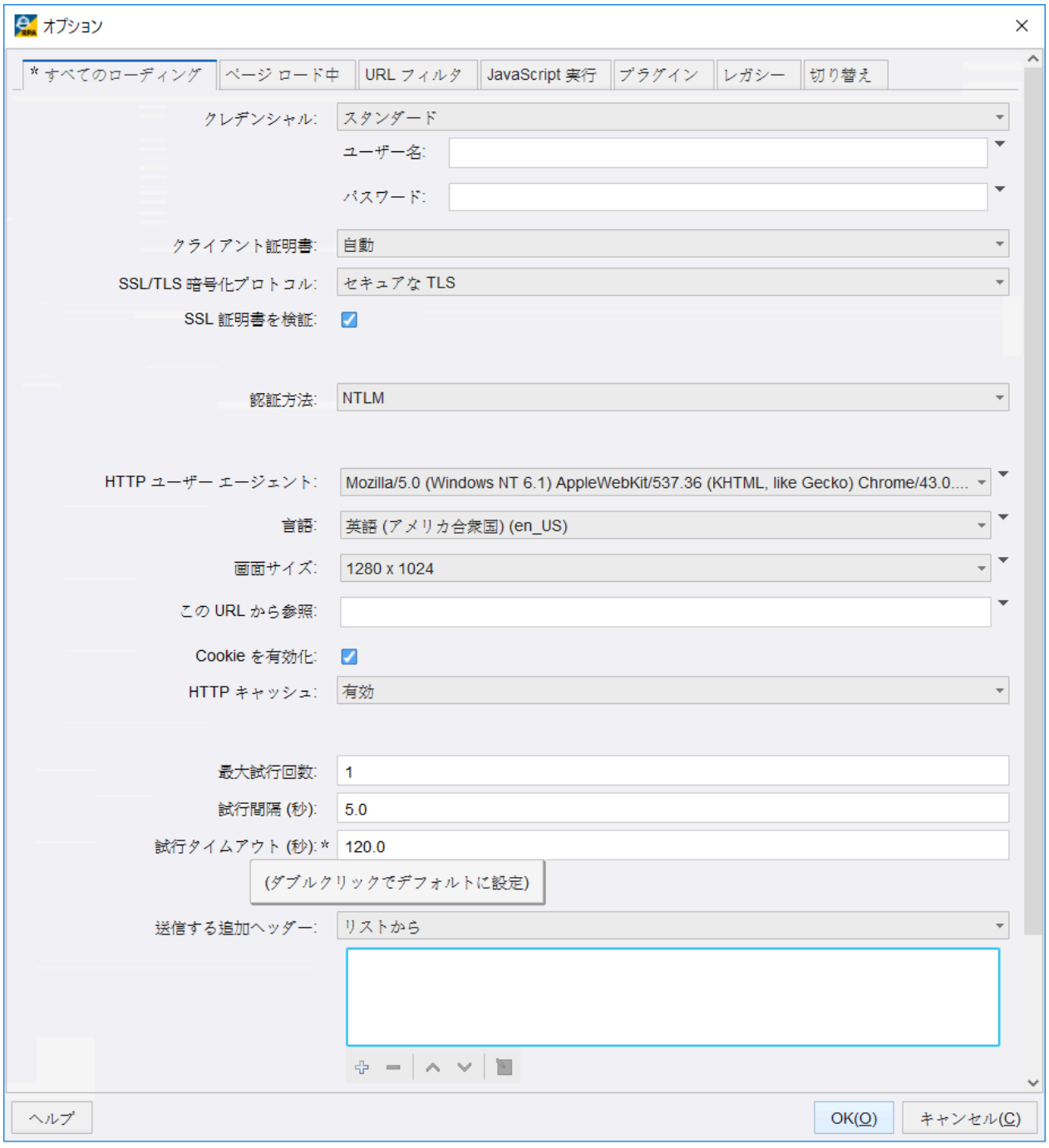
口ボット設定の非標準部分を表示することが難しい場合があります。たとえば、ページ読み込みステップにデフォルト値の 60.0 よりも長いタイムアウトが割り当てられている場合があります。非標準の設定を調べるには、Design Studio の内部を移動して、デフォルトの設定から変更されているプロパティを見つけることが必要な場合があります。また、すべてのプロパティのデフォルト値を覚えておく必要があります。

Design Studio で、変更の表示を使用すると、これらの手動の手順を実行しなくて済みます。次の図に示すように、特定のデフォルト値が定義されているプロパティは、値が変更されると、プロパティ名の横のアスタリスク * でマークされます。



タブにもアスタリスクが表示されていることに注意してください。このアスタリスクは、タブのいずれかのプロパティが変更されたことを示します。

プロパティ名またはアスタリスクを右クリックし、値をデフォルト値にリセットします。通常、コンテキスト メニューにはデフォルト値が表示されます。



上記の図は、[オプション] ダイアログ ボックスの [各試行のタイムアウト] プロパティを示しています。

変更の表示は、ステップ オプションの設定に使用された場合、[オプション] ウィンドウで異なる動作をします。通常、次の 2 つの場所でオプションを設定できます。

- ロボットの設定から
- これらのオプションに依存している場合があるステップで

いずれの場所でも、ボタンをクリックし [オプション] ウィンドウを開いてオプションを設定しますが、それぞれの状況に対して、デフォルト値は異なります。ロボットの設定からウィンドウを開いた場合、デフォルト値はアプリケーションの固定デフォルト値であり、Design Studio によって提供される値が表示されます。ステップの設定からウィンドウを開いた場合、デフォルト値は、ロボットの設定の下で定義されている値 (ロボットのデフォルト値) です。つまり、ステップでオプション値が明示的に変更されていない場合、ステップは、ロボットの設定からオプション値を継承します。たとえば、ロボットの設定で [Cookie の有効化] オプションがオフになっている場合、ステップでこの設定を明示的に変更していない限り、このオプションに依存するすべてのステップでもこの設定が使用されます。ステップ オプションのアスタリスクは、ロボットの設定で定義されている値とは異なる値がステップで使用されることを示します。ステップで使用される値が、アプリケーションのデフォルト値と必ずしも異なるとは限りません。

ステップの [オプション] ダイアログのアスタリスクが、ロボットの設定の [オプション] ウィンドウの場合とは異なる意味を持つ別のケースがあります。ステップの [オプション] ウィンドウのオプションの場合、アスタリスクは、任意のオプションが意図的に固定値に設定されていることを意味します。この固定値が、ロボットの設定の対応する値と必ずしも異なるとは限りません。また、この値は、ロボットの設定の対応する値を変更しても影響を受けません。たとえば、最初にステップの [各試行のタイムアウト] プロパティが 120 に、ロボットの設定の下にある対応する値が 60 にそれぞれ設定されている場合、アスタリスクがオプションの横に表示されます。ロボットの設定の値が 120 に変更され、2 つの値が実際に同じになった場合でも、ステップの値は引き続きアスタリスクでマークされます。ロボットの値が 120 から再度変更された場合は、ステップの値は 120 のまま変化しません。コンテキスト メニューを使用して、またはダブルクリックしてステップの値をデフォルト値 (ロボットの設定からの値) に戻した場合、ステップの値にはロボットの構成の値が使用され、以降は、変更があった場合はそれらの変更に従った値になります。

デフォルト値が他の設定の選択に依存するもう 1 つの状況は、エラー処理ステップの設定に適用されます。

ロバストなベーシック エンジン ロボットの作成

Web サイトは予告なしに変更されることがあります。ロバストとは、Web サイトの変更にロボットが適切に対処できるかを示すために用いられる用語です。ロボットが処理できる変更が増えて、多くの変更が適切に機能するようになるほど、ロボットのロバスト性は高まります。

ロバストなロボットを作成するには、対象の Web サイトを分析し、登録フォームが間違って入力された場合など、さまざまな状況におけるロボットの反応について理解する必要があります。また、ロバストなロボットを作成する際に Web サイトのロジックのリバースエンジニアリングが必要になることもあります。そのためには、通常、Web サイトの内容を精査するしかありません。

ロバストに対しては以下の 2 つのアプローチがあり、それぞれ目的が異なります。

- 可能な限り成功させる。
- 不完全な場合に失敗するようにする。

ニュースのタイプ変数を抽出するロボットでは、可能な限り成功させるということは、可能な限り多くのニュース アイテムを抽出するということを意味します。ベーシック エンジン ロボットでは、条件付きアクション、トライ ステップ、およびデータ コンバータを使って、さまざまなレイアウト、欠落した情報、および不適切にフォーマットされたコンテンツに対処することができます。

オーダーを提出するタイプのロボットでは、不完全な場合の失敗とは、フィールドを正しく入力する方法が不明確であったり、オーダーの結果ページが正確なレイアウトと一致しない場合などに、ただちに失敗するようにすることを意味します。この場合、「失敗」とは API 例外を生成することではありませ

ん。エラーや障害の原因を記述する専用の値を口ボットが返すようにすることを意味します。入力変数を取る口ボットでは、成功よりも頻繁に失敗が発生することあります。ベーシック エンジン 口ボットでは、専用のエラー タイプ変数、エラー処理、および条件付きアクションを使って予期しない状況を検出し、処理することができます。

口ボットをよりロバストにするための Design Studio 技術の詳細については、以下のセクションを参照してください。

- 適切に構造化された口ボットの記述
- エラーの処理
- デバッグ モード
- ステップ アクション
- データ コンバータ

適切に構造化された口ボットの記述

各口ボットはプログラムであるため、適切に構造化されたベーシック エンジン 口ボットの記述は重要です。次の理由から、適切に構造化された口ボットを記述することが重要になります。

- 口ボットを文書化することが容易になる。
- 口ボットのメンテナンスが容易になる。
- 口ボットの操作が容易になる。

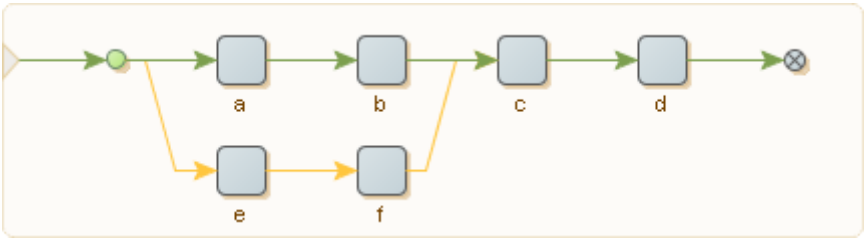
適切に構造化された口ボットを記述すると、副次的に Design Studio で口ボットがより高速にロードできるようになります。そのため、通常は口ボット ビューで口ボットを編集すると、応答がより高速になります。

適切に構造化されたベーシック エンジン 口ボットを記述するための主なツールとして、スニペット ステップとグループ ステップの 2 つがあります。両ステップ タイプでは、口ボットの一部分を取り出し、説明的な名前を付けたり、単一のステップにまとめたりします。このため、口ボットの一部分の機能を詳細に知らなくても、口ボットの全体的な構造に注意を向けることが可能になります。この概念は、メソッド、関数、プロシージャなど、他のプログラミング言語の概念と似ています。

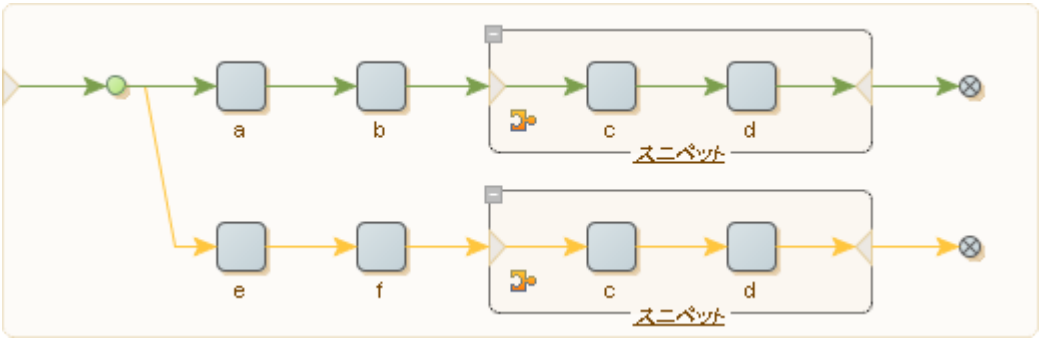
グループ ステップを使用して、明示的に定義されたタスクを実行する複数のステップをまとめたり、非表示にしたりできます。「サイト X へのログイン」や「エラーのレポート」など、ステップに説明的な名前を付けます。グループ ステップの名前は、グループ内のステップの動作について説明する比較的短く説明的な名前にすることが重要です。適切な名前を付けることができなかった場合、明示的に定義されたタスクをグループが実行できない原因になる可能性があります。グループ ステップを導入すると、名前で口ボットのその部分の動作を表せるため、口ボットの文書化に役立ちます。

スニペットは主に口ボット間の機能の共有に導入されていますが、単一の口ボット内で使用して口ボットの構造化を支援することもできます。ステップの開始時に結合する、口ボットの各部分からの接続など、口ボット内に複数の分岐で使用されるステップのコレクションがある場合、このようなステップの共有は、ステップを含むスニペットの導入によって置き換えることができます。

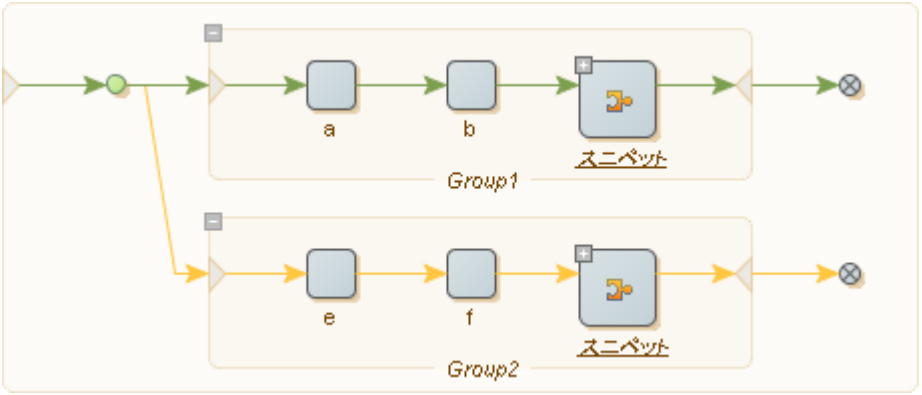
次の口ボット構造では、接続を結合する代わりに、スニペットとグループが使用されています。



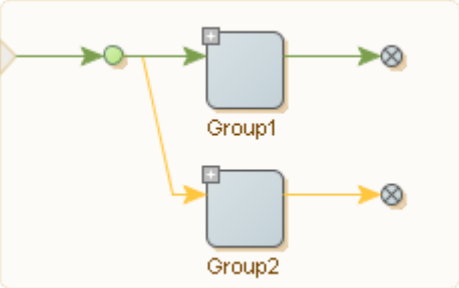
最後の 2 つのステップ **c** とステップ **d** は、ステップ **a** とステップ **e** で始まる 2 つの分岐で共有されます。実際には、はるかに大きなロボットと、このようにステップを共有する 3 つ以上の分岐が存在して、関連するステップがさらに離れている場合があります。その結果、ロボットの全体像を把握することが難しくなることがあります。ロボットを適切に構造化するために、まず次のようにステップ **c** とステップ **d** を含むスニペット ステップを導入できます。



スニペット ステップ内のステップを編集し、2 つの分岐で変更が共有されるようにできます。両方の分岐をグループ ステップに入れて、ロボットをさらに構造化することができます。



最後に、2 つのグループ ステップを使用して、次の簡単なロボットを記述します。



結果として、この口ボットは、2 つのタスクのうち 1 つを Group1、もう 1 つを Group2 として実行します。これらの 2 つのグループに説明的な名前を付けることにより、口ボットの構造が元の口ボットよりも論理的になります。

これは明らかに非常に簡単な例ですが、口ボットのサイズが特定のサイズを超えて、口ボット ビューに多くの接続が表示されている場合は、口ボットが複雑になりすぎている可能性があります。上記のように口ボットを再構造化すると、口ボットの全体像を管理しやすくなります。

エラーの処理

口ボットのステップでは、ステップが実行されたときにエラーが生成される場合があります。たとえば、タグ ファインダーが処理対象のタグを見つけることができない場合、またはステップがエラーを生成した場合、こうしたエラーが発生します。テストが失敗したときにエラーが発生したかのように動作するテスト ステップを設定することができます。口ボットのデフォルト動作では、エラーを即座に報告してログに記録し、失敗したステップ以降のステップの実行を省略します。ただし、口ボットのステップの「エラー処理」プロパティを設定して、この動作を変更することができます。たとえば、エラーを生成するステップをスキップしたり、別の分岐を試したりするように口ボットを設定できます。

このヘルプ システムで説明しているエラー処理動作は、口ボットのランタイム実行 (RoboServer やデバッグ モードでの実行など) に適用されます。Design Studio のデザイン モードでの実行には適用されません。デザイン モードでは、通常、エラーが即座に報告され、以降のステップの実行が中止されます。例外的に、エラーが発生したときにステップを「無視して続行」するように設定している場合は、Design Studio は、ランタイム実行時と同様に、エラーを無視して次のステップを実行します。

API 例外と口グ エラーを処理する方法を次に示します。

1. ステップ ビューの [エラー処理] タブで、エラー処理オプションを選択します。
    a. 口ボットの呼び出し元にエラーを報告するために、[**API** 例外] を選択します。これは、いずれかの API を介してクライアントにより口ボットが RoboServer で実行されている場合に最も有用です。この場合、API を介して呼び出し元にエラーが RobotErrorResponse として送信され、少なくともデフォルトの RQLHandler を使用している場合、呼び出し側で例外が発生します。口ボットがその他の方法で実行される場合の詳細については、エラー処理を参照してください。
    b. エラーを口グに記録するには、[エラーとしてログ記録] を選択します。口ボットが Design Studio または RoboServer で実行されているかどうかに応じて、ログは異なる方法で記録されます。

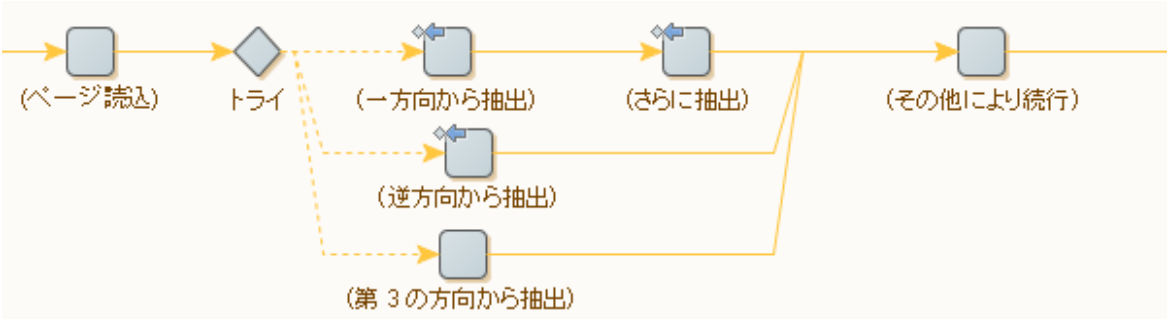
チェック ボックスのオン/オフで選択します。チェック ボックスがアスタリスク * でマークされている場合は、デフォルト以外の値に設定されていることを示します。詳細については、アスタリスクを除去して、デフォルト値に戻す方法を説明しているデフォルトからの変更を表示を参照してください。デフォルト値が適用された場合 (つまり、アスタリスクがない場合)、デフォルト値は、エラーを処理する方法によって異なることに注意してください。

2. [次の処理] フィールドで、リストからオプションを選択します。
   この値により、エラーの発生後、ロボットの実行を続行する方法と場所が定義されます。可能なオプションは、次のセクションで例を使って説明します。詳細については、「エラー処理」セクションを参照してください。

別のエラー処理方法

エラー処理には複数の方法があります。エラー処理の概要については、条件とエラー処理を参照してください。

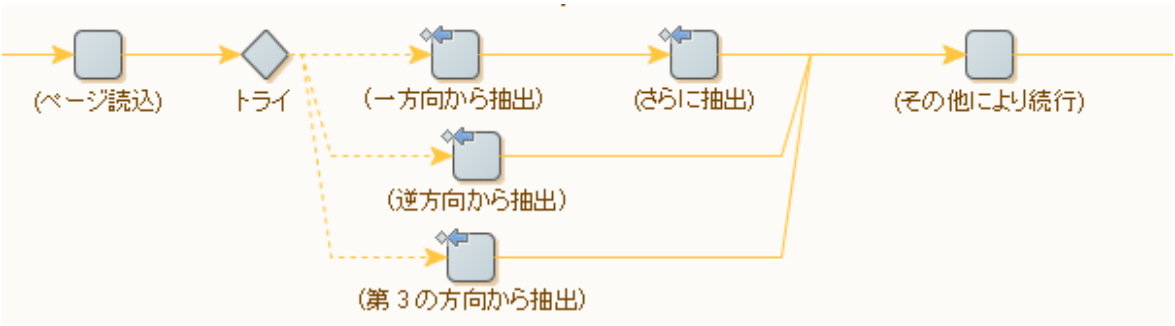
Web ページの一部にさまざまな構造とレイアウトがあるものの、その部分が 3 つのケースのいずれかに該当する場合を仮定します。各ケースには、抽出する情報があります。これは、1 度に 1 つのケースの抽出を試行することによって実行できます。失敗した場合には次のケースを試行し、3 番目のケースまで試行します。これで成功することが想定できます。



抽出ステップの (次のトライステップへ移動) アイコンに注目してください。抽出に失敗すると、トライ ステップの次の分岐が実行されます (トライ ステップから出る分岐が成功すると、次の分岐は実行されません)。抽出ステップでは同時に以下の 2 つのことが実行されます。ステップで Web ページから抽出し、完了できない場合には、次のアプローチを試行するようにします。1 つ目の分岐の 2 つのステップのいずれかが失敗すると、2 つ目の分岐が実行されることに注意してください。これは、分岐の「成功条件」をステップの組み合わせで表現する方法の一例です。

抽出の「第 3 の方法」が必ず機能する場合は、このアプローチが最も効果的です (たとえば、Web ページから実際にデータを抽出するのではなく、固定のデフォルト値を適用する場合など)。最初の 2 つの分岐がアクセスしたように、3 番目の分岐が Web ページにアクセスした場合、成功すると仮定するのは正しくない可能性があります。次回、ロボットを実行したときに、Web サイトが大幅に変更されて 3 つの方法がどれも成功しない可能性もあるため、ロボットが合理的な方法で対応できるようにする必要があります。

最も簡単な対応方法は、問題を呼び出し元に報告してログに記録し、抽出と以降の操作をすべて放棄することです。これは、最初の 2 つの分岐と同様に、3 番目の分岐が作業に失敗したかどうかをトライ ステップに通知することによって実現できます。



(トライ ステップでは、[後続のステップすべてをスキップ] とは、レポートやログの後に追加のアクションを実行しないことを意味します)。

あるいは、トライ ステップで、問題を以前のトライ ステップに戻して処理することもできます。詳細については、「トライ-キャッチ」を参照してください。

一般的なケースにおけるショートカット

トライ ステップと「次のトライステップへ移動」エラー処理は、非常に柔軟なツールです。適切な方法で使用することで、さまざまな方法でエラーを処理することができます。このトピックでは、シンプルで一般的なケースをいくつか紹介します。実際には、これらのケースは非常に一般的であるため、特殊なエラー処理オプションによってもサポートされています。

後続のステップ全てをスキップ

多くの場合、ロボットは Web ページで任意のエレメントの処理が必要になります。つまり、エレメントが存在する場合には処理が必要ですが (データの抽出など)、存在しない場合には、エレメントの処理をスキップできます。エレメントが存在しなくてもエラーではなく、想定内の状況です。これは、ロボットでは次のように表すことができます。ステップ A はエレメントが存在するかどうかテストします。ステップ B と C は続けて処理を実行しますが、これはステップ A の成功に依存します。

ステップ A が成功しなかった場合 (エレメントがウェブページにない場合)、[次のトライステップへ移動] ( ) エラー処理は、トライ ステップに通知を送信します (この例では名前がありません)。これにより、2 番目の空の分岐が実行され、その後、トライ ステップで始まる分岐全体の実行が完了します。したがって、ステップ A が成功しなければ、ステップ B と C は実行されません。

この状況は一般的なものであるため、固有のエラー処理オプションである [後続のステップ全てをスキップ] をショートカットとして導入します。以下のように、この例を単純にすることができます。

ステップ A のエラー処理は次のように設定します。これはすべての新しいステップのデフォルト設定です。

厳密に言えば、トライ ステップで示されているのとまったく同じ動作にするには、[**API** 例外] と [エラーとしてログ記録] のチェック ボックスをクリアする必要があります。その理由は、エラー処理を行う 2 つの方法では、これらのチェックボックスのデフォルト値が異なるためです。

ステップ B がステップ A に類似していた場合 (つまり、ステップ B にも [次のトライステップへ移動] エラー処理があった場合) 、この同じショートカットを使うことができることに注意してください。

無視して続行

何らかの条件を満たす場合、アクション (抽出など) の実行をして、それ以外の場合はスキップする必要があることもあります。後続のステップは、この結果に依存しません (または、結果に対して適切なデフォルトが事前に設定されています)。これは、次のように表すことができます。

ステップ A が成功しなかった場合、[次のトライステップへ移動] ( ) エラー処理により、(名前のない) トライ ステップからの 2 番目の空の分岐が実行されます。この後、ステップ A に入力されたのと同じロボット状態でステップ B で実行が継続され、ステップ A が効率的にスキップされます。

これは、ステップ A でエラー処理オプション [無視して続行] ( ) を使って、トライ ステップなしで行うこともできます。

また、無視された場合でも状況の記録を可能にすることができます。これは、以下のように、ステップ A でエラー処理を設定することで実現できます。

トライ ステップを使用した方法でも、同様の手順で設定できます。

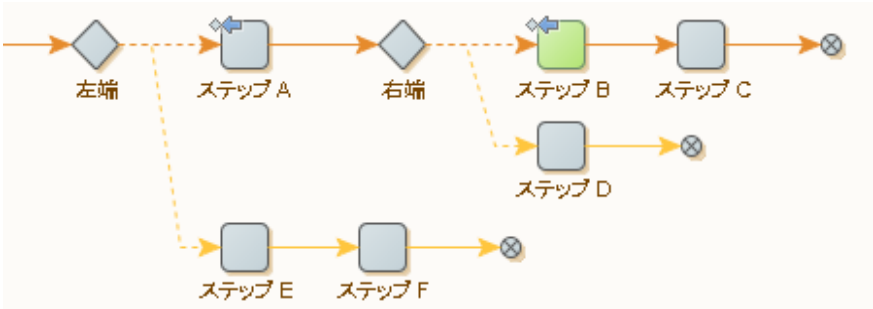
At ターゲット

[次のトライステップへ移動]

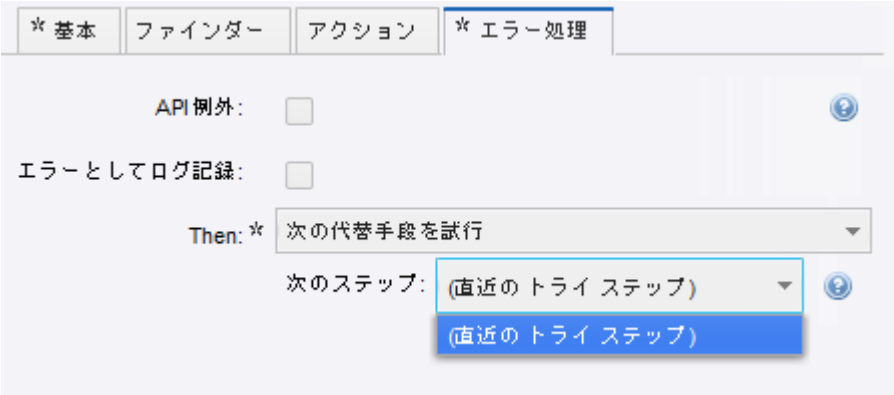
「次のトライステップへ移動」によるエラー処理は、その性質からトライ ステップと呼ばれます。このトライ ステップは、現在の実行パスの前のステップ、つまり現在のステップに至る実行を含むステップの中の 1 つである必要があります。それ以外の場合、次のトライステップへ移動の「次のトライステップへ移動」の部分は意味をなさなくなります。

現在の実行パスに複数のトライ ステップがあるロボットについて考えます。この状況で、ロボットはこれらのトライ ステップのそれぞれに関連する「現在の分岐」の実行中で、トライ ステップごとに「次の分岐」は異なります。たとえば、次のロボットでは、ステップ B でエラーが発生した場合、「次のトライステップへ移動」は次のいずれかのアプローチを使用して実行を続行します。

- トライステップの右端にある次の分岐。実行ではステップ C をスキップし、ステップ D を続行します
- トライ ステップの左端にある次の分岐。これにより実行ではステップ C と D をスキップし、ステップ E を続行します



[エラー処理] タブでオプションを定義できます。次の例では、ステップ B に対するエラー処理設定を示します。



デフォルトは最も近いトライ ステップですが、実行パスでその他のトライ ステップを選択できます。トライ ステップは名前で参照されます。複数のトライ ステップの名前が同じ場合、その名前の最も近いもの (右端) を指します。これは、トライ-キャッチ に記載されているように活用できます。

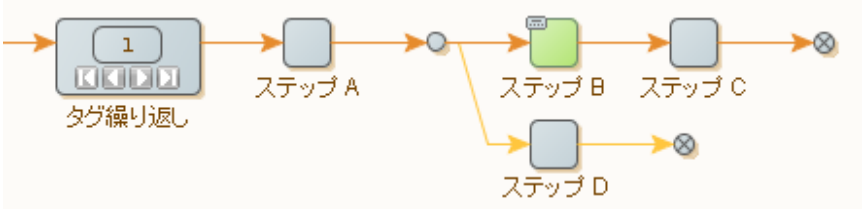
この例では「次のトライステップへ移動」エラー処理との関連で At ターゲットについてのみ説明していますが、そのような参照はルーピングに記載されている「次のイテレーションへ移動」および「ループ終了」エラー処理と使用できます。これらの場合には、参照はトライ ステップではなく、ループ ステップに移動します。

ルーピング

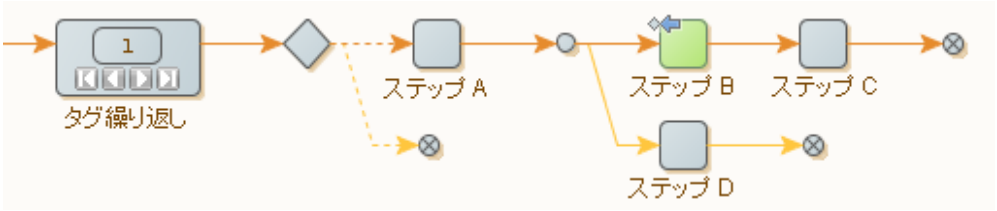
エラーが発生した場合やテストが失敗した場合などは、ループの現在のイテレーションまたはループ全体の実行を中止するのが適切な反応になります。これは、2 つの特殊なエラー処理オプションによってサポートされています。

次のイテレーションへ移動

後続の口ボット、ステップ B には、「次のイテレーションへ移動」に対する エラー処理が含まれます。このステップの実行中にエラーが発生すると、現在のループ イテレーションの実行が停止されます。ステップ C と D は実行されず、代わりにイテレート オーバーするループ ステップのうち、次のタグを反映する口ボット状態にあるステップ A において実行が継続されます。



このエラー処理オプションは、[次のトライステップへ移動] やトライ ステップを利用するのと同じ効果が得られるショートカットです。



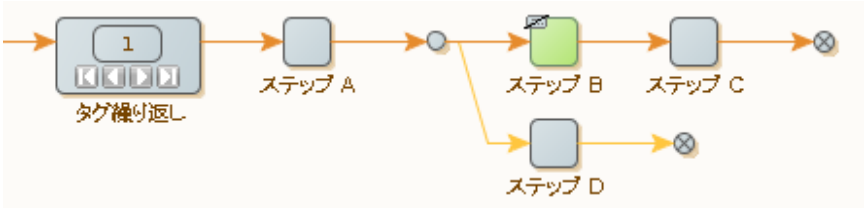
他のトライ ステップが干渉する恐れがあるため、この変換には一般的に At ターゲットの使用が必要であることに注意してください。

口ボット内のループ ステップの後にループステップが続く場合は、次のイテレーションに進むステップを選択することができます。

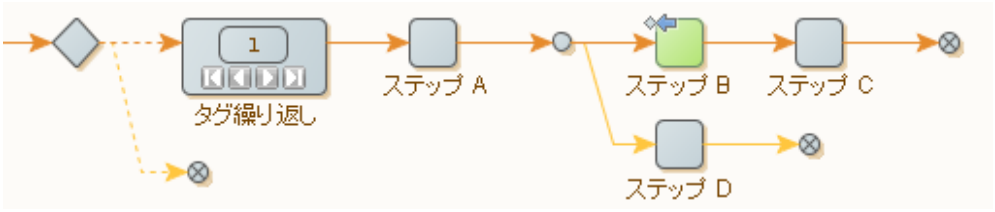
「次のイテレーションへ移動」は、「繰り返し - 次」のループでは機能しません。これら 2 つの場合において、ブラウザの状態については、「次」という言葉がまったく別の意味を持ちます。

ループ終了

「次のイテレーションへ移動」でループの 1 回のイテレーションを完了する代わりに、「ループ終了」を使ってループ全体を途中停止することができます。



このエラー処理オプションは、ショートカットです。以下の口ボットには同じ効果があります。



「次のイテレーションへ移動」とは異なり、「ループ終了」は、「繰り返し - 次」のループと共に機能します。

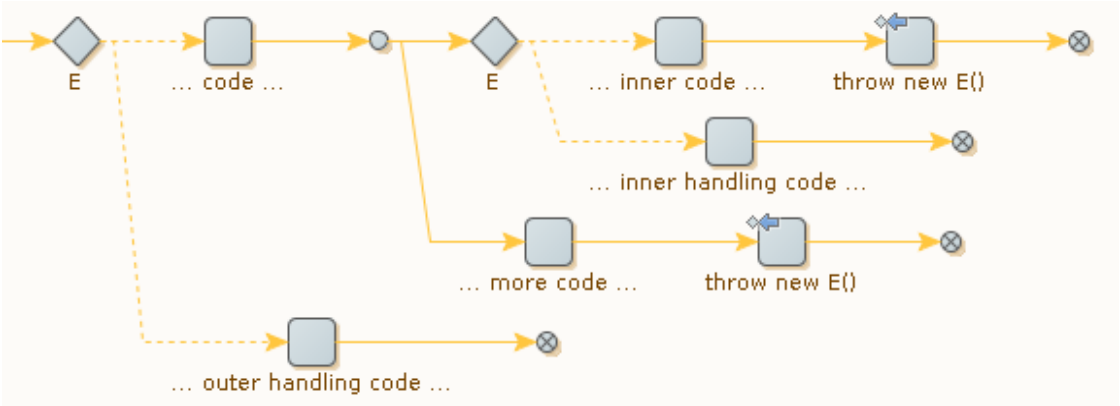
トライ-キャッチ エラー処理

[次のトライステップへ移動] エラー処理をターゲット トライ ステップへの明示的な [次のステップ] リファレンスと併用すると、ステップは名前で識別されます。ほとんどの場合、ターゲット ステップとその名前の細かい区別は重要ではありませんが、Java や C# の try-catch 構文に似た、例外処理機能を提供するのに悪用される可能性があります。

これらのプログラミング言語では、"try" と "catch" の間にあるコードのセクションには、特別なエラー処理があります。このセクションで特定のエラーが通知された場合 (名前付きの「例外」を「スローする」ことで)、同様に名前をつけた "catch" に続くコードの一部が実行されます。try-catch 構文は入れ子にできるため、名前を付けた「例外」は必ず一致する名前が付いた最も内側にある "catch" により処理されます。例 :

```
try {
   ... code ...
   try {
     ... inner code ...
     throw new E(); // caught by innermost "catch"
   }
   catch (E e) {
     ... inner handling code ...
   }
   ... more code ...
   throw new E(); // caught by outermost "catch"
 }
 catch (E e) {
   ... outer handling code ...
 }
```

口ボットでは、同じようなことをトライ ステップで行うことができます。「次のステップ」で選択されるトライステップ名は、(現在の実行パス上にある) 選択された同じ名前のトライステップの直ぐ側の次のステップを指していることを覚えておきましょう。同じ実行パス上であっても、複数のトライ ステップに同じ名前を使うことが許されています。したがって、各 try-catch 構文は、「例外」と同じ名前のトライ ステップでモデル化されます。トライ ステップには 2 つの分岐があり、1 つは "try" 構文のコード部分、もう 1 つは "catch" 構文のコード部分用です。



Java/C# 構文と Design Studio の用語は、以下の表のように対応しています。

Java / C# 構文	**Design Studio** で使用される要素
try { ...code...}	トライ ステップの 1 番目の分岐 (コードに対応するステップ)
例外の名前	トライ ステップの名前
throw new E() (トライのコード中)	E における [次のトライステップへ移動] によるエラー処理
catch E { ...code...}	"E" という名前のトライ ステップの 2 番目の分岐 (コードに対応するステップ)

したがって、中心となる考え方は、トライ ステップをエラー処理に使うときは、処理するエラー状況の名前からトライ ステップに名前を付けます。メリットは以下のとおりです。

- 名前を付けると、各トライ ステップの目的を明確にできます。
- エラーが一般的なレベルで (ロボットで左側にあるトライ ステップを使用して) 処理されたとき、場合によっては (同じ名前の 2 番目のトライ ステップを使用して) 特殊な処理をするのが簡単になります。

ロボット ビューでのエラー処理の識別

ロボット ビューでは、特別なエラー処理が含まれるロボット ステップが小さなシンボルでマークされます。このシンボルは、ステップに対して定義されているエラー処理のタイプに基づいています。このシンボルにより、ユーザーはカスタム エラー処理が含まれるステップを視覚的に識別できます。ステップにカスタム エラー処理のマークを付ける必要がない場合は、Design Studio の **[Design Studio** 設定**]** > **[**ベーシック エンジン ロボット エディター**]** メニューでこの機能を無効にできます。

詳細については、「ベーシック エンジン ロボット エディター 」を参照してください。

エラー処理

ステップの実行中にエラーが発生した場合、エラーはステップの設定の [エラー処理] タブで指定された方法で処理されます。テスト アクションでも、条件が失敗する場合は、同じ処理を適用することがあります。エラーまたは失敗したテストの処理には 2 つの側面があります。1 つはインシデントを報告または ログ記録する方法と場所で、もう 1 つはベーシック エンジン ロボットの実行を続行する方法と場所です。

プロパティ

次のプロパティを利用してエラー処理を設定します。

API 例外

このプロパティではインシデントをロボットの呼び出し元に報告するかどうかを指定します。報告が行われる方法は、ロボットが実行される方法によって異なります。

- デザイン モードでは (以下で説明する [無視して続行] を使用してインシデントが処理される場合を除いて) インシデントが常に報告されるため、ロボットが Design Studio のデザイン モードで実行される場合、このプロパティは効力を持ちません。
- ロボットが Management Console で実行されている場合、API 例外はカウントされエラーとしてログに記録されますが、実行は停止されません。このシナリオでは、[エラーとしてログ記録] と同じ値 (チェックボックスのオンまたはオフ) を使用することを推奨します。
- ロボットがいずれかの API を介してクライアントで実行され、RoboServer 上で実行されると、レポートは次のように API を介して呼び出し元に送り返されます。

  ```
  RobotErrorResponse
  ```

  少なくともデフォルトの RQLHandler を使用しているときは、これによって呼び出し側で例外が発生します。(プロパティの名前が「例外」になっているのは、そのためです) 。

このプロパティはオンまたはオフになっている可能性がありますが、プロパティが明示的に非デフォルト値に設定されていることを意味するアステリスク (*) でマークされていることもあります。これについては、アステリスクを除去してデフォルト値に戻す方法も説明している [デフォルトからの変更の表示] で説明します。デフォルト値が適用される (つまりアステリスクがない) 場合は、Then プロパティの下で行われた選択に従ってデフォルトが変化することに注意してください。

エラーとしてログ記録

このプロパティではインシデントをログ記録するかどうかを指定します。これは報告とは異なります (前項を参照してください)。

- ロボットが Design Studio のデザイン モードで実行される場合、インシデントは [表示] メニューで [ログを表示] を選択したときに表示されるログに追加されます。
- ロボットがデバッグ モードで実行される場合、インシデントは [ログ] タブに追加されます。
- ロボットが (Management Console または API を介して) RoboServer で実行されると、インシデントは、指定した RoboServer のクラスタ設定で定義された場所に記録されます。その場所は、たいていの場合、Management Console が使用しているデータベースと同じデータベースです。

このプロパティはオンまたはオフになっている可能性がありますが、上記の [API 例外] で説明したように、アステリスク (*) でマークされていることもあります。

次の処理

このプロパティでは、インシデント発生後のロボット実行方法と実行場所を指定します。次のオプションが利用可能です。

後続のステップ全てをスキップ **(**デフォルト**)**

エラーが発生した (またはテスト条件が失敗した) ステップに続くステップは実行されません。その他の点では、ステップが問題なく実行されたかのように実行が続行されます。

無視して続行

エラーが発生した (またはテスト条件が失敗した) ステップがスキップされ、そのステップに続くステップから実行が通常通り続行されます。

次のトライステップへ移動

トライ ステップから出る分岐でこのプロパティを使用することができます。トライ ステップからの現在の分岐の実行が中止され、そのトライ ステップからの次の分岐上の最初のステップから実行が続行されます。そこでは、最初の分岐と同じロボット状態で実行が続行されます。

現在の分岐がそのトライ ステップからの最後の分岐である場合、[次のトライステップへ移動] は無効ではありませんが、エラー (「すべての代替手段が失敗しました」) が発生し、その エラーはそのトライ ステップで エラー処理が設定されている方法に従って処理されます。

トライ ステップがネストされている場合は、現在のステップへの実行パス上にあるトライ ステップから該当するトライ ステップを選択することができます。

後方に送る **(**レガシー**)**

このオプションは、8.0 より前の Design Studio のバージョンで作成されたロボットとの後方互換性を保つためにのみ存在します。このオプションについては別途説明します。

次のイテレーションへ移動

ループの内部で、つまり (Repeat-Next ループを除く) 何らかのループ ステップの後で、このプロパティを使用することができます。現在のループ イテレーションの実行は中止され、次のイテレーションから実行が続行されます。つまり、次の looped-over 項目に対応するロボット状態で、ループ ステップの後の最初のステップから実行が続行されます。

ループがネストされている場合は、次のイテレーションへの移動元となる該当するループを選択することができます。

ループ終了

ループの内部で、つまり (Repeat-Next ループを含む) 何らかのループ ステップの後で、このプロパティを使用することができます。ループの実行は中止されます。したがって、現在のイテレーションは終了されず、次以降のイテレーションはまったく実行されません。これは、ループの通常の終了後と同じステップから実行が続行されることを意味しています。

ループがネストされている場合は、現在のステップへの実行パス上にあるループから該当するループを選択することができます。

「*At...*」 でのステップ リファレンス

エラー処理では、エラーが発生したときに実行を続行するさまざまな方法を選択することができます。以下に挙げる 3 つの方法では、実行を再開する場所となる実行パス上の前のステップ (トライ ステップまたは繰り返しステップのどちらか) を特定する必要があります。ユーザーによる以下の選択に応じて、実行は、選択されたステップまたは選択されたステップの後から続行されます。

- 次のトライステップへ移動
- 次のイテレーションへ移動
- ループ終了

優先ステップを特定するときは、(適切な種類の)「最も近いステップ」を選択するか、名前で特定のステップを選択することができます。ただし、さまざまな理由により、優先ステップが選択リストに表示されないこともあります。

- 名前付きステップのみを選択できます。ステップに名前がない場合 (トライ ステップのデフォルト)、ステップを選択リストに表示するには、ステップに名前を付ける必要があります。ステップが適切な種類の最も近いステップである場合は、「最も近いステップ」を選択することによってそのステップにアクセスすることもできます。

- [次のイテレーションへ移動] を [繰り返し] ステップと組み合わせて使用することはできないことに注意してください。したがって、[繰り返し] ステップは [次のイテレーションへ移動] の選択リストに表示されません。
- 特定先は必ず選択された名前を持つ (適切な種類の) 最も近いステップになるため、ステップ名が選択リストに複数回表示されることは決してありません。ロボットを変更するときはこのことを念頭に置いてください!
- ロボットを通って現在のステップに至るパスが複数ある場合は、選択された名前を持つ (適切な) ステップが現在のステップに至るすべてのパス上に存在する必要があります。存在しない場合、名前は選択リストに表示されません。

最後の 2 つの規則は Java または C# の トライ-キャッチ 構造体に非常によく似た高度な方法によるエラー処理の使用をサポートしています。このトピックの詳細については、トライ-キャッチ セクションを参照してください。これらのプログラミング言語では、エラー発生点で名前付き「例外」を「スロー」し、周囲のコードのどこかで (名前で) それを「キャッチ」します。ロボットでは、トライ ステップで似たような処理を実行できます。Java または C# の用語と Design Studio の用語の対応を以下の表に示します。

Java/C# の用語	**Design Studio** で使用するもの
トライ	トライ ステップの最初の分岐
例外の名前	トライ ステップの名前
throw E	E における [次のトライステップへ移動] によるエラー処理
catch E	名前 "E" を持つトライ ステップの 2 番目の分岐

エラー処理に対するこの考え方は、おそらく大規模なロボットで最も有用でしょう。この方法でエラー処理を使用する場合、トライ ステップの名前は、個々のトライ ステップの 2 番目の分岐が処理する例外状況の種類を示します。

後方に送る *(*レガシー*)*

「後方に送る (レガシー)」は、エラー処理のオプションの 1 つです。これは、8.0 よりも前のバージョンの Design Studio で作成したロボットに、後方互換性を提供するためにあります。以降のロボットでは使用しません。

「後方に送る」は、別のエラー処理オプションである「次のトライステップへ移動」と同様に、トライ ステップから抜け出す分岐の実行に影響します。ただし、動作が大きく異なります。大きな違いとしては、現在のステップに続くステップを実行しないことです (「後続のステップ全てをスキップ」と同様)。これは別として、トライ ステップからの現在の分岐の実行は継続されます。ただし、エラーはトライ ステップに「後方に送られて」記憶されます。これにより、現在の分岐の実行が正常に終了すると、トライ ステップの次の分岐の実行は継続します。この部分は、遅れて実行されるものの「次のトライステップへ移動」の効果と同様で、ロボット状態について同じコメントが適用されます。

現在の分岐が、そのトライ ステップからの最後の分岐である場合、「後方に送る」はエラーにはなりませんが、エラーになります (「すべての候補が失敗」)。このエラーは、トライ ステップに対するエラー処理の設定方法に従って処理されます。

前のトライ ステップが検出されなかった場合は、エラーはロボットの開始に「後方に送られて」API 経由で報告され、ロボットの実行が終了する直前に記録されます。

「後方に送る」は遅延が発生するため使用しにくく、「次のトライステップへ移動」の使用が推奨されます。8.0 よりも前のバージョンの Design Studio で作成したロボットを開くと、「後方に送る」の代わ

りに「次のトライステップへ移動」を使用するように自動的に修正されます (ロボットからの結果を変更せずに修正ができる場合) 。

条件とエラー処理

ロボットは、さまざまなケースでさまざまなアプローチを使用できます。ケースは明示テストに基づき、条件の評価と、処理を必要とするエラーの発生のいずれかに区別されます。

このトピックの例は、最小実行 (ダイレクト) デザインタイム実行モードに基づいています。

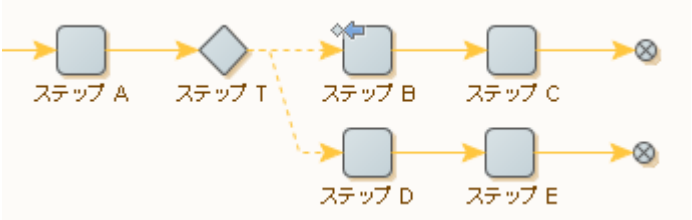
条件によって、入力ロボット状態 (HTML ページに特定のタグがあるなど) のコンテンツに基づき実行のフローが変更されます。エラー処理とは、特定のエラーの発生時に実行のフローを変えることです。たとえば、存在が想定されるアンカー タグが HTML ページに見つからず、クリックできない場合などです。多くの場合、状況は次の 2 通りになります : アンカー タグが見つかった場合 (条件) にはクリックする。または、ロボットがアンカー タグのクリックを試行してエラーを処理できる (アンカー タグが見つからなかった場合)。一部のケースでは、一般的に条件と考えられているものは非常に複雑でそれ自体を記述することはできません (たとえば、「この特定のページがエラーなしでロードできる場合」という条件)。そのような場合、ページを読み込み、エラーを条件が失敗したことの指標と見なします。

その他のエラーは、ロボットまたはアクセス中の Web サイトに実際に問題があることを示しています。たとえば、Web サイトがダウンし、ページのロード中にエラーが発生した場合、または HTML ページのページ レイアウトを動的に変更したために、タグ ファインダーで必要なタグが見つからなかった場合などが挙げられます。特定のエラーは一部の状況では失敗した条件であり、その他の状況では実際のエラーと考えられます。解釈はロボットによって異なります。

このように条件実行とエラー処理間の境界が不明確なため、Design Studio では両方の機能を統一された方法で提供しています。ステップごとに、エラー発生時に何をするかを設定できます。さらに、特定の条件に基づくテスト アクションを含むステップでは、同じアプローチを再利用します。つまり、条件が一致しない場合、(デフォルトの) アクションはエラーが発生した場合のように適用されます。

ロボットのステップごとに、エラーへの必要な対応を設定できます。ここでは、2 つの有用なエラー処理オプションについて説明します。その他のオプションについては、エラーの処理方法を参照してください。最初のオプションは、トライ ステップに密接に関連しています。

トライ ステップでは複数の分岐が出ているため、分岐ポイントに似ています。トライ ステップは分岐ポイントとは異なります。最初の分岐より後の分岐は、先行する分岐のステップで「次のトライステップへ移動」オプションに基づいて処理されるエラーが発生した場合にのみ実行されるためです。次のロボットについて、通常ステップでそれぞれ 1 つのロボット状態が出力されるものとします。



アイコンは、ステップ B が「次のトライステップへ移動」によってエラーを処理するように設定されていることを示します。

ステップ B が正常に実行されると、ステップ実行は次のようになります：「A、T、B、C」。T から出ている最初の分岐がエラーなしで実行されるため、2 番目の分岐はまったく実行されません。

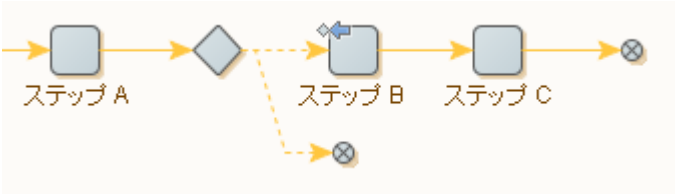
一方で、ステップ B でエラーが発生すると、ステップの実行は次のようになります：
「A、T、B、T、D、E」。ステップ B のエラーが処理された後、実行は後続のステップで続行されず、代わりにトライ ステップから出ている次の分岐の開始時に続行されます。

トライ ステップからの各分岐は、その点から続行する可能な 1 つの方法を表します。各分岐の開始に近いステップでは、分岐に沿った実行が実行可能なアプローチであるかどうかが検証されます。実行可能でない場合は「次のトライステップへ移動」が実行されます。分岐が現在のケースに対して適切であると判別された場合、後のステップが実際の作業を行います。分岐の開始の近くの判定ステップは、テスト ステップか、またはエラーが発生した場合にこの分岐は続行経路ではないことを示すステップにすることができます。トライ ステップから出ている分岐は多数にすることもできます。

Java、JavaScript、C# などの通常のプログラミング言語と同様に、先行する口ボットは "if-then-else" 構成に似ています。トライ ステップの後の最初の分岐には条件 ("if" の部分) と "then" の部分が含まれ、最後の分岐には "else" の部分が含まれています。2 つ以上の分岐がある場合、最初と最後の間の分岐は "else-if" の部分と同様です。

最初の分岐がエラーを発生させる可能性がある何らかのアクションを実行しようとする場合、例は "トライ-キャッチ" 構成ともいえます。最初の分岐が "try" の部分で、2 番目の分岐が "catch" の部分と同様です。

もう 1 つのエラー処理オプション、「後続のステップ全てをスキップ」では、共通した特殊ケースを表すさらにコンパクトな方法を提供しています。これについて次の口ボットで例示します。エラーが発生する可能性があるステップは最初の分岐の最初のステップで、2 番目の分岐は何も発生しません。



結果として、エラーが発生した場合にはステップ B より後のステップの実行がスキップされます。トライ ステップなしで、エラー処理オプション「後続のステップ全てをスキップ」(デフォルト) を以下の方法で使用しても同様の結果になります。

ロケーションとロケーション コード

エラーが処理された際に、口ボットの呼び出し元に報告を返したり、記録したりすることができます。いずれの場合も、メッセージには、エラーが発生したステップの口ケーションと口ケーション コードとともにエラーの簡単な説明が含まれます。

エラーが発生したステップの口ケーションは、開始ステップからそのステップに到達するのに必要なステップ (イテレーション番号を含む) のリストにあります。次の口ボットを考えてみましょう。

ステップ B の 2 回目のイテレーションでステップ C がエラーを報告すると、その口ケーションは以下のように記述されます。「ステップ A - ステップ B [2] - ステップ C」(この口ケーションには、ステップ名とイテレーション番号がハイフンで区切られて挿入されていることに注意してください)分岐ポイントは省略されています。

口ケーション コードは口ケーションに似ていますが、各ステップの名前は、名前の重複を避けるためにそのステップに対して一意の識別子に置き換えられます。前述の口ケーションの例では、口ケーション コードは以下のようになります。`{a-i1-a}`Design Studio の口ケーション コードを使用して、エラーを報告したステップに直接進みます ([編集] メニューの **[ステップを移動]** > **[指定した口ケーションへ移動]** を使用)。

ロケーションとロケーション コードのイテレーション回数が 0 であるため、開始イテレーションは次のようになります。`{a-i0-a}`

接続と実行フロー

接続を使用して、ステップ間の実行フローを決定します。

このトピックの例は、最小実行 (ダイレクト) デザインタイム実行モードに基づいています。

次のようなシンプルな口ボットがあるとします。

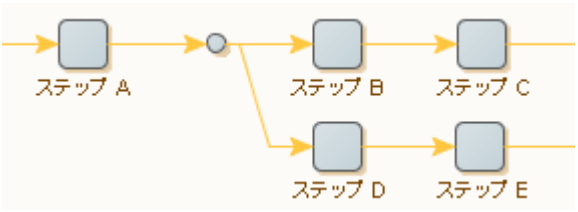
この口ボットは、次の 3 つのステップから構成されています : ステップ A、ステップ B、およびステップ C。エラーが発生せず、各ステップで 1 つの出力口ボット状態が生成される場合、口ボットは次のように実行されます。最初の口ボット状態が生成され、ステップ A (最初のステップ) への入力として使用されます。ステップ A により出力口ボット状態が生成されます。この出力口ボット状態はステップ B の入力口ボット状態です。ステップ B はステップ C の入力口ボット状態である口ボット状態を生成します。ステップ C が実行され、出力口ボット状態が生成されると、実行は完了します。つまり、ステップの実行は次のようになります。「A、B、C」。

ステップは実行時に出力口ボット状態を生成しないことがあります。これは、エラーまたはテストステップが原因で実行が口ボット内の別のステップで続行された際に発生します (条件とエラー処理を参照)。

ループ アクションを含むステップでは入力状態を複数回処理することがあり、そのたびに異なる口ボット状態が出力されます。ステップ B にループ アクションが含まれる次のような口ボットがあるとします。

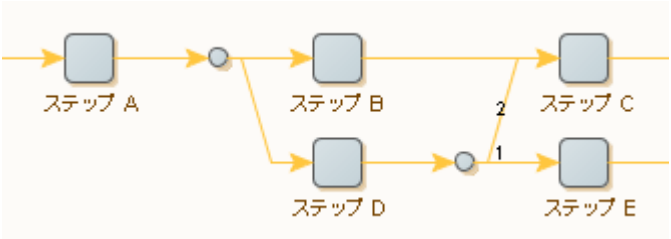
エラーまたはテスト ステップがなく、ステップ B が 3 つの口ボット状態を出力し、その他のすべてのステップが 1 つの口ボット状態を出力する場合、ステップは次の順序で実行されます。「A、B[1]、C、D、B[2]、C、D、B[3]、C、D」。B[N] はステップ B に含まれるループ アクションの N 回目のイテレーションを表します。ステップ B によって出力された口ボット状態は別の口ボット状態です。各イテレーションにより新しい口ボット状態が出力されます。そのため、ステップ C は実行されるたびに新しい入力口ボット状態を受け取ります。

1 つのステップを複数のステップに接続できます。これは「分岐」と呼ばれます。次のような口ボットがあるとします。



この口ボットで、ステップ A の後に分岐ポイントが続き、接続は 2 つの分岐に分かれています。1 つの分岐はステップ B とステップ C から構成され、もう 1 つの分岐はステップ D とステップ E から構成されます。分岐ポイントから出るすべての分岐は次々に実行されます。そのため、エラーまたはテスト ステップで管理フローが変更されず、各ステップで 1 つの出力口ボット状態が生成される場合、先行口ボットは次のように実行されます : A、B、C、D、E。ただし、ステップ B とステップ D はそれぞれステップ A によって生成されたものと同じ出力口ボット状態のコピーを受け取ります。

分岐は複雑な方法で結合できます。次のような口ボットがあるとします。



この口ボットは、接続が明示的に順序付けられる様子を示しています。この口ボットでは、ステップ D の分岐が数字によって指定された順序で実行されます。ステップ E はステップ C の前に実行されます。順序が数字によって指定されていない場合、接続はトップダウンで実行されます。そのため、テスト ステップがなく、エラーが発生せず、各ステップで 1 つの出力口ボット状態が生成される場合、口ボットは次のように実行されます : A, B, C, D, E, C。ステップ C が初めて実行されると、ステップ B によって生成された出力口ボット状態を受け取ります。ステップ C が 2 回目に実行されると、ステップ D によって生成された出力口ボット状態を受け取ります。

状況に応じて、複数の分岐のうち 1 つだけを選択 (実行) する必要がある場合があります。条件とエラー処理トピックでは、これを行う方法について説明します。

ブラウザ トレーサ

この機能は、ベーシック エンジン ロボットでのみ使用できます。

ブラウザ トレーサは Design Studio のツールメニューから利用できます。ブラウザ トレーサは、Design Studio の HTTP トラフィックと、デフォルトの (WebKit) ブラウザ エンジンを使用するロボットの JavaScript コンソール メッセージをトレースできます。ブラウザ トレーサは、ロボットをパフォーマンス チューニングしたり、現状のままでは機能しないサイトの回避策を考えたりするのに役立ちます。

トレーシング

トレーシングを開始するには、[記録] ボタンを有効にします。記録中、特に JavaScript レコーディングを使用している場合は、膨大な量のデータが収集されるため、通常よりも動作が遅くなることがあります。目的のトラフィックのトレースが完了したら、必ず [記録] ボタンを無効にしてください。

HTTP トレース

HTTP トレースは HTTP トラフィックを表示します。トレース項目を選択すると、その HTTP イベントに関する詳細情報がトレースの下の詳細ビューに表示されます。詳細ビューには、リクエスト ヘッダとレスポンス ヘッダおよび送信されたリクエスト データとレスポンス データが表示されます。通常は、POST リクエストにのみリクエスト データが含まれます。

保留中の読み込みは青で表示されます。

URL をブロック

HTTP リストで項目を右クリックし、[**URL** をブロック リストに追加] を選択することによって、特定の URL をブロックすることができます。URL パターンを編集するための [URL ブロック パターンを設定] ダイアログ ボックスが開きます。詳細については、URL ブロックを参照してください。

JavaScript トレース

個々の JavaScript トレースの下に、現在選択されているトレース項目の JavaScript ソース コードからのコンソール メッセージが表示されます。トレース項目を選択すると、対応するソース コード行がソース ビューでハイライト表示されます。トレース項目はハイライト表示されているソース コード行の実行のランタイム結果です。個々のソース コード行は、当然、複数回実行できます。その場合は、すべてが同じソース コード行に対応する複数のトレース項目が生成されます。

トレース項目をたどることは、1 つの JavaScript コードが動作する仕組みの理解に役立ちます。

HTTP トレースにしか関心がない場合は、JavaScript のトレーシングをオフにすることができます。この操作を行うには、[トレース] メニューで [JavaScript トレースを有効化] のチェック マークを外します。

トレース セッションの保存とロード

後で読み込むためにトレース セッションを保存することができます。トレース セッションには Design Studio トレースとプロキシ トレースおよび JavaScript トレースと HTTP トレースが含まれます。トレース セッションが大きいため、後で詳しく調べる場合、または誰かにトレース セッションを電子メールで送信する場合は、トレース セッションを保存すると便利です。

Design Studio から送信されるバグ レポートには、存在する場合、現在のトレース セッションが自動的に含まれます。

ダウンロードされたコンテンツが "Page Changes" または "JavaSript changes" によって変更された場合、ブラウザ トレーサは [レスポンス] タブの変更されたコンテンツが表示される場所に余分な行を追加します。URL は `Rewritten` から始まります。

トレース レスポンスの保存
トレース レスポンスはパスによって保存することができます。

[ファイル] > **[保存]** または **[ファイル]** > **[名前を付けて保存]** をクリックして [保存] ダイアログ ボックスを開き、すべての本文の応答を保存するディレクトリを選択します。

すべてのリクエストは、そのパスによって選択されたフォルダのドメイン ディレクトリに保存されます。

URL:

```
https://chedlink.chedraui.com.mx/Artus/g9/projects/main.php
```

ディスク:

```
c:\somedir\chedlink.chedraui.com.mx\Artus\g9\projects\main.php
```

パターン

パターンとは、テキストを記述する形式的な方法です。たとえば、テキスト "32" は、2 桁を含むテキストとして記述することができます。また一方で、"12" や "00" といった 2 桁の数字を含む他のテキストも、2 桁の数字を含むテキストとして記述することができます。これは、パターン \d\d として表すことができます。このパターンは、テキストに 2 つの数字が含まれ、2 桁に限定されることを示す規則的な表記方法です (d は数字を表す記号)。この場合、「これらのテキストはこのパターンと一致する」と言うことができます。(Design Studio のパターンは Perl5 の構文に従います。)

パターンは、通常の文字と特殊記号で構成されています。特殊記号にはそれぞれ特別な意味があります。たとえば、特殊記号 "."(ドット) は任意の単一文字を表し、"a"、"b"、"1"、"2" などのすべての単一文字と照合されます。

以下の表に、最も一般的に使用される特殊記号の概要を示します。

特殊記号	説明
.	"a"、"1"、"/"、"?"、"." などの任意の単一文字。
\d	"0"、"1" ~ "9" などの任意の 10 進数。
\D	"0"、"1" ~ "9" を除いた "." に等しい、任意の非数字。
\s	" " や改行などの空白文字。
\S	空白 (" " や改行など) を除いた "." に等しい、空白以外の任意の文字。
\w	"a" ~ "z"、"A" ~ "Z"、"0" ~ "9" などの任意の単語文字列 (英数字)。
\W	"a" ~ "z"、"A" ~ "Z"、"0" ~ "9" を除いた "." に等しい、任意の単語文字列 (英数字)。

例

- パターン ".an" は、"mcan" ではなく、"can" と "man" のような "an" で終わるすべての 3 桁のテキスト長と照合されます。

- パターン "\d\d\s\d\d" は、2 桁の数字で始まり空白を 1 つ挟んで 2 桁の数字で終わる、5 桁のテキスト長 ("01 23" や "72 13" など。 "01 2s" は当てはまらない) と照合されます。
- "." または "\" のような特殊文字を通常の文字として機能させたい場合は、その前に "\" (バックスラッシュ) を付けてエスケープさせることができます。"." の文字に正確に照合させたい場合は、任意の単一文字の代わりに "\." と記述します。

  たとえば、パターン "m\.n\\o" は、テキスト "m.n\o" にのみ一致します。
- 以下の括弧を使って、パターンをサブパターンに整理できます。"(" および ")" 。

  たとえば、パターン "abc" は、"(a)(bc)" として整理できます。
- 単一文字はすべて、サブパターンとみなされます。

  たとえば、パターン "abc" では、単一文字 "a"、"b"、"c" はサブパターンとみなされます。

サブパターンは、パターン演算子を適用するときに便利です。次の表に、利用可能なパターン演算子の概要を示します。

演算子	説明
?	前のサブパターンまたは空のテキストに一致。
*	前のサブパターンの繰り返し回数、または空のテキストに一致。
+	先行するサブパターンの 1 つ以上の繰り返しに一致。
{m}	先行するサブパターンの m 回の繰り返しと正確に一致。
{m,n}	先行するサブパターンの m ~ n 回 (両端を含む) の繰り返しに一致。
{m,}	先行するサブパターンの m 回以上の繰り返しと正確に一致。
a|b	エクスプレッション a が一致するもの、またはエクスプレッション b が一致するものに一致。

例

- ".*" は、"Design Studio"、"1213"、「」(空のテキスト) などの任意のテキストに一致
- "(abc)*" は、"abca" ではなく、「」、"abc"、"abcabc"、および "abcabcabc" などのテキスト "abc" の繰り返し回数に一致
- "(\d\d){1,2}" は、"12" や "6789" のような 2 桁または 4 桁の数字のいずれかに一致しても、"123" には一致しない
- "(good)?bye" は、"goodbye" と "bye" に一致
- "(good)|(bye)" は、"good" と "bye" に一致

他の特殊文字と同様に、文字の前に "\" バックスラッシュを追加して、パターン演算子に現れる特殊文字をエスケープすることができます。

サブパターンは、テキストから特定のテキスト部分を抽出するときに便利です。カッコを使ってサブパターンを作成すると、そのサブパターンに一致するテキスト部分を抽出することができます。たとえば、パターン "abc (.*) def (.*) ghi" を考えてみましょう。このパターンには、括弧でまとめられた 2 つのサブパターンがあります。このパターンをテキスト "abc 123 def 456 ghi" と照合した場合、これらのサブパターンの最初はテキスト "123" と一致し、2 番目のサブパターンはテキスト "456" と一致します。エクスプレッション (エクスプレッションを参照) では、"$1" と "$2" を記述することで、これらのサブパターンの一致を参照できます。たとえば、"X" + $1 + "Y"+ $2 + "Z" というエクスプレッションは、結果 "X123Y456Z" を生成します。これは、Design Studio で非常に重要な抽出手法です (これについては、「エクスプレッションの試行」で説明しています)。

デフォルトでは、繰り返しパターン演算子 (*, +, {...}) は、可能な限り前のパターンの繰り返し回数に一致します。演算子の後に "?" を置くことで、できるだけ繰り返し回数の少ない演算子に変換することができます。たとえば、パターン ".*(\d\d\d).*" を考えてみましょう。このパターンをテキスト "abc 123 def 456 ghi" と照合した場合、最初の * 演算子は可能な限り多くの繰り返しと照合されるため、サブパターン "(\d\d\d)" はテキストの 2 番目の数字 ("456") と一致します。パターンが ".*?(\d\d\d).*" になるように * 演算子の後に "?" を置くと、*? 演算子はできるだけ少ない繰り返し回数で照合されるため、サブパターン "(\d\d\d)" はテキストの最初の数字 ("123") と一致します。

ご自分でパターンを試行してみてください。試行を行う場合の最適な方法は、Design Studio を起動し、[タグ判定] アクションなどのパターンを入力できる位置を見つけることです。次に、パターン フィールドの右側にある [編集] ボタンをクリックして、[パターン エディター] ウィンドウを開きます。

[パターン エディター] では、パターンを入力し、[入力値] パネルのテスト入力値テキストと一致するかどうかテストできます。ウィンドウを開いたとき、Design Studio は通常、所定のステップが現在の入力ロボット状態で実行された場合に、パターンが一致するテキストにテスト入力値テキストを設定します。しかし、テスト入力値テキストを自分で編集して、他の入力でパターンを試すこともできます。パターンをテストするには、[テスト] ボタンをクリックします。照合の結果は、[出力値] パネルに表示されます。

パターン エディターの [シンボル] ボタンは、パターンに特殊記号を入力する場合にとても便利です。このボタンをクリックするとメニューが表示され、パターンに挿入するシンボルを選択できます。このようにシンボルを選択する方法であれば、特殊記号とその意味をすべて覚える必要はありません。

使用可能な特殊記号やパターンの詳細については、「パターン」を参照してください。

エクスプレッション

エクスプレッションからは通常、テキストが求められます。たとえば、エクスプレッション

"The author of the book " + Book.title + " is " + Book.author + "." は、変数 Book.title と Book.author にテキスト "Gone with the Wind" と "Margaret Mitchell" がそれぞれ含まれている場合、テキスト "The author of the book Gone with the Wind is Margaret Mitchell." が求められます。

また、エクスプレッション内で数値を計算することもできます。たとえば、変数 Book.price に本の価格が含まれている場合、次のエクスプレッションを使用して、これに 100 を掛けることができます。

Book.price * 100

以下の表に、最も一般的に使用されるサブエクスプレッション タイプの概要を示します。利用可能なすべてのサブエクスプレッション タイプの完全な概要については、下記のリファレンスを参照してください。

一般的に使用されるサブエクスプレッション タイプ

サブエクスプレッション タイプ	表記	説明
テキスト定数	"text" または >>text<<	指定したテキスト (例： "Margaret Mitchell" または >>Margaret Mitchell<<) に評価します。
変数	variablename.attributename	指定した変数の値を求めます。たとえば、"Book.author" から "Margaret Mitchell" を求めます。

サブエクスプレッション タイプ	表記	説明
現在の URL	URL	現在のページの URL を求めます。
サブパターン マッチ	$n	関連付けられているパターンのサブパターンでマッチしたテキストを求めます (ある場合)。たとえば、これは以下のようにアドバンスド抽出データ コンバータで使用されます。$0 はパターン全体でマッチしたテキストを求めます。
関数	func(args)	指定した関数を指定した引数に渡し、その結果をテキストに変換して求めます。

引用符による表記または >>text<< 表記 (例： "Margaret Mitchell" または >>Margaret Mitchell<<) を使用してテキスト定数を指定できます。引用符による表記を使用し、引用文字がテキスト内に表示されるようにする場合、それを 2 つの引用符で記述する必要があります。たとえば、テキスト "This is some ""quoted"" text" を取得するには、"This is some "quoted" text" と記述します。>>text<< 表記を使用する場合、">>」と "<<" を除く、すべてのテキストを表示できます。そのため、>>This is some "quoted" text<< などのように、引用を直接記述することができます。>>text<< 表記は、HTML など、多くの引用文字が含まれている長いテキストに有用です。

以下の表に、エクスプレッションで最も一般的に使用される関数を示します。

関数	説明
toLowerCase(arg)	引数を小文字に変換します。
round(arg)	引数を整数に四捨五入します。

たとえば、エクスプレッション "The discount is " + round((Item.oldPrice - Item.newPrice) / Item.oldPrice) + "%." は、アイテムの古い価格が $99.95、新しい価格が $89.95 の場合、"The discount is 10%." が求められます。

リファレンス

エクスプレッション

エクスプレッション タイプ	表記	説明	例
定数	"text"	固定テキスト。 バックスラッシュ文字 (\) を使用すると、特殊文字を入力できます。 \n は改行 \r はキャリッジ リターン \f はフォーム フィード \t は水平タブ \b はバックスペース \」は二重引用符 \' は一重引用符 \\ はバックスラッシュ \uxxxx は xxxx エンコードの Unicode 文字。xxxx は 4 つの 16 進数の値。	"This is some \"quoted \」text." "This a text with line break \n, tab \t and a unicode \u0035 character"

エクスプレッション タイプ	表記	説明	例
定数	>>text<<	固定テキスト。この表記には引用符文字などあらゆるものを含めることができます。ただし、終了シンボル (<<) を除きます。バックスラッシュ (\) 文字は、特殊文字の入力に使用できません。	>>This is some "quoted" text.<<
変数	variable.attribute 変数	変数の値。変数がコンプレックス タイプの場合、属性の名前も必要になります。	Book.title + 整数
入力テキスト	INPUT	ある場合は、エクスプレッションへの入力テキスト。	INPUT
連結	expr1 + expr2	2 つのエクスプレッション expr1 および expr2 の連結。	"Title:"+ Book.title
サブパターン マッチ	$n	n > 0 の場合、パターン内のサブパターン n に一致するテキスト。n = 0 の場合、パターン全体に一致するテキスト。 注意：このエクスプレッションが意味を持つのは、エクスプレッションに関連するパターンがある場合に限ります。	$1 詳細については、パターンと エクスプレッションの試行 を参照してください。
数式	演算子：+, -, *, /, %	数式の結果。	Book.price * 100
ブール式	演算子：&& (and), || (or)	ブール式の結果。	performTransactions && Book.price < 30
条件式	condition ? expr1 : expr2	条件の評価結果が true の場合は expr1 の値を使用し、それ以外の場合は expr2 の値を使用します。	Book.price < 30 ? "cheap" : "expensive"
等価	演算子：== (等しい)、!= (等しくない)	これらの演算子は、あらゆるタイプのオペランドで動作しますが、オペランドのタイプは同一である必要があります。たとえば、数値を整数と比較することはできません。	true == false style != "none"
関係	演算子：< (未満)、<= (以下)、> (より大きい)、>= (以上)	関係演算子は、一方のオペランドが他方のオペランドより小さいか大きいかを判定します。 オペランドは整数タイプまたは数値タイプの数値で、エクスプレッション内のオペランドのタイプは同一である必要があります。	0 < 1 1.0 >= 0.0
関数	func(expr)	エクスプレッション expr の結果に適用される func 関数。利用できる関数については、このテーブルの後の「関数」のセクションを参照してください。	capitalize(Book.title)
現在の URL	URL	現在の URL	URL

エクスプレッション タイプ	表記	説明	例
現在のウィンドウ	WINDOW	現在のウィンドウの固有 ID。	WINDOW
ロボット名	Robot.name	ロボット名。	Robot.name
実行 ID	Robot.executionId	ロボットの現在の実行 ID。	Robot.executionId
実行エラー	Robot.executionErrors	最も近いトライ ステップの、前の分岐で発生した実行エラー。 このエクスプレッションは、トライステップの 2 番目以降の分岐で使用できます。	Robot.executionErrors

関数

関数	説明
abs(arg)	数の絶対値を返します。
base64Decode(arg)	Base64 でエンコードされたデータをデコードします。
base64Encode(arg)	バイナリ データを Base64 エンコーディングでエンコードします。
binaryToText(data[, encoding])	バイナリ データをデコードしてテキストにします。指定のエンコーディングが有れば使用します。無い場合は、デフォルトで UTF-8 エンコーディングを使用します。
capitalize(arg)	すべての単語の最初の文字を大文字に、他の文字をすべて小文字にします。
ceil(arg)	数を一番近い整数に切り上げます。
collapseSpaces(arg)	スペースが 2 つ連続しないようにします。
contains(source, key)	ソースに指定されたキーが含まれるかを返します。
date()	現在の日付を標準的な日付の形式 (yyyy-mm-dd) で返します。
day(args)	引数として与えられた日付の日を返します。
endsWith(source, key)	ソース文字列が指定されたキーで終わる場合は true を返し、それ以外の場合は false を返します。
floor(arg)	数を最も近い整数に切り捨てます。
guid()	ランダムに生成された、グローバルな固有 ID (GUID) を返します。
hexDecode(arg)	16 進エンコードのデータをデコードします。
hexEncode(arg)	バイナリ データを 16 進エンコーディングでエンコードします。
indexOf(source, key)	ソース内のキーの最初のインデックスを返します。無い場合は -1 を返します。
length(arg)	テキスト内の文字数をカウントするか、バイナリ データが与えられた場合はバイト数をカウントします。
max(a, b)	2 つの数のうちの大きい方の値を返します。

関数	説明
md5(arg)	引数として与えられたバイナリ データの MD5 チェックサムを計算します。
min(a, b)	2 つの数のうちの最小値を返します。
month(args)	引数として与えられた日付の月を返します。
now()	現在の日付と時刻を返します。
random()	0 ~ 1 の間の乱数を返します。
removeSpaces(args)	SPACE、\t、\n などの、引数内の空白文字をすべて除去します。
replacePattern(source, pattern, newText)	テキスト "source" 内に "pattern" パターンが発生する度に、テキスト "newText" に置き換えます。パターンの一致は大文字と小文字を区別しません。
replaceText(source, oldText, newText)	テキスト "source" 内にテキスト "oldText" が発生する度に、テキスト "newText" に置き換えます。"oldText" との一致は大文字と小文字を区別しません。
resolveURL(arg)	現在の URL を使用して、相対の URL を絶対に変換します。
round(arg)	最も近い整数に四捨五入します。
shortTime(arg)	引数として与えられた日付に対して、秒の小数部が無い時間 (hh:mm:ss) を返します。
substring(source, startIndex)	startIndex からソース文字列の末尾までを範囲とするソース文字列の一部を返します。文字列の先頭文字のインデックスが 0 になります。
substring(source, startIndex, endIndex)	startIndex から endIndex までを範囲とするソース文字列の一部を返します。文字列の先頭文字のインデックスが 0 になります。
startsWith(source, key)	ソース文字列が指定されたキーで始まる場合は true を返し、それ以外の場合は false を返します。
textToBinary(text[, encoding])	テキストをバイナリ データにエンコードします。指定のエンコーディングが有れば使用します。無い場合は、デフォルトで UTF-8 エンコーディングを使用します。
time(arg)	引数として与えられた日付の時間 (hh:mm:ss.fff) を返します。
toInteger(args)	テキストを整数に変換します。計算に含めたい場合に便利です。
toNumber(args)	テキストを浮動小数点数に変換します。計算に含めたい場合に便利です。
toLowerCase(arg)	テキスト内のすべての文字を小文字に変換します。
toUpperCase(arg)	テキスト内のすべての文字を大文字に変換します。
trim(arg)	テキスト文字列の両端から、スペースと空白文字をすべて削除します。
urlDecode(arg)	URL にエンコードされたテキストをデコードします。
urlEncode(arg)	テキストを URL にエンコードします。
weekday(arg)	引数として与えられた日付の曜日の名前を (英語で) 返します。
year(args)	引数として与えられた日付の年を返します。

関数	説明
toColumn(arg)	整数を Excel の列テキストに変換します。
toIndex(arg)	Excel の列テキストを整数に変換します。
iteration(arg)	現在のイテレーション数が含まれます。

コンバータで使用できるのは、テキスト タイプの値を返すエクスプレッションのみです。詳細については、「データ コンバータ」を参照してください。

エクスプレッションの試行

自分でエクスプレッションを試すことをお勧めします。エクスプレッションを試す最適な方法は、Design Studio を起動し、既存の口ボットを開くことです。

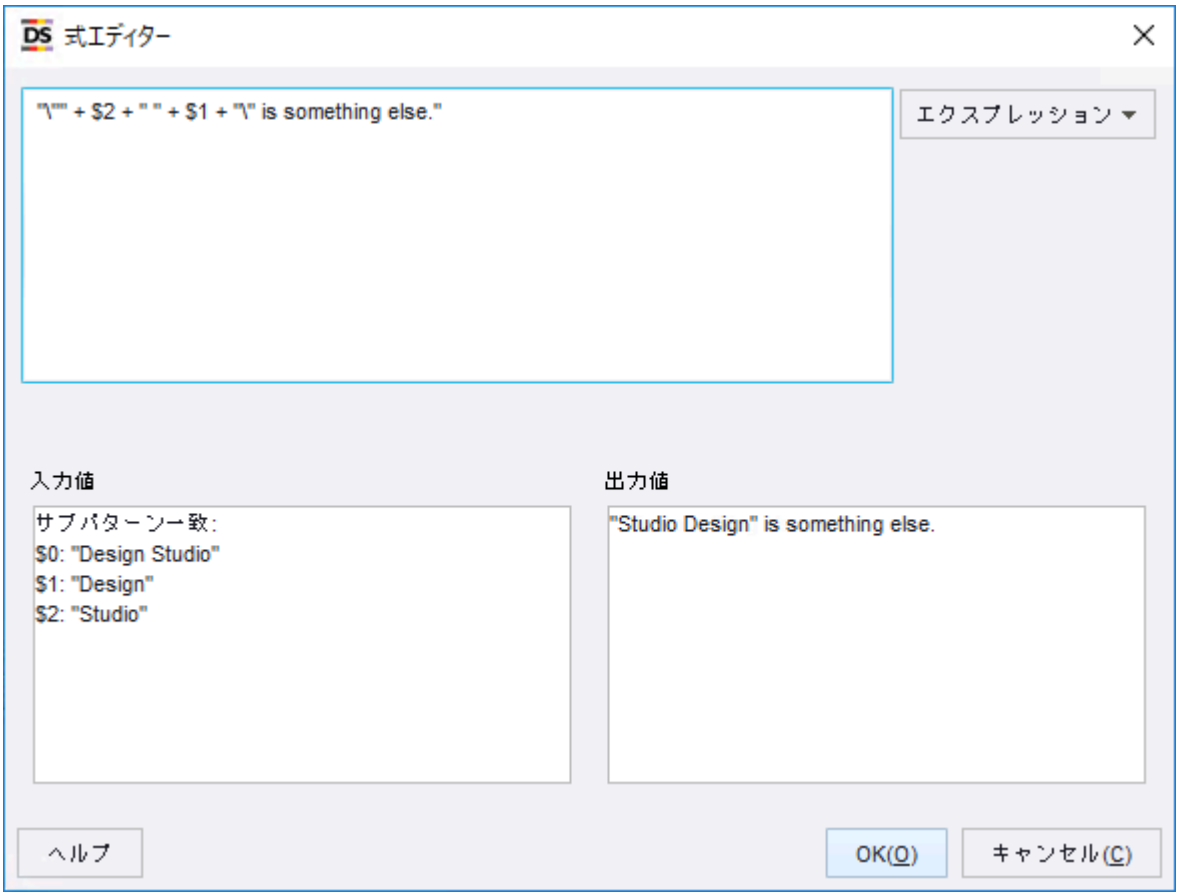
1. Design Studio で、現在のステップの [抽出] アクションを選択します。
2. アドバンスド抽出データ コンバータを追加します。
3. 設定 アイコンをクリックして、データ コンバータを設定します。
   [アドバンスド抽出設定] ウィンドウが表示されます。

示されている例では、入力テキストの一部を抽出するために $n 表記が使用されていることに注意してください。

4. 左側にあるテキスト領域の入力テキストを変更します。
5. 次に、パターン プロパティを変更します。
6. [出力エクスプレッション] プロパティを変更します。
   エクスプレッションを入力して、右側の領域の結果を確認します。

エクスプレッションの編集

1. [アドバンスド抽出設定] ウィンドウの [出力エクスプレッション] フィールドで [編集] をクリックします。
   式エディターが表示されます。



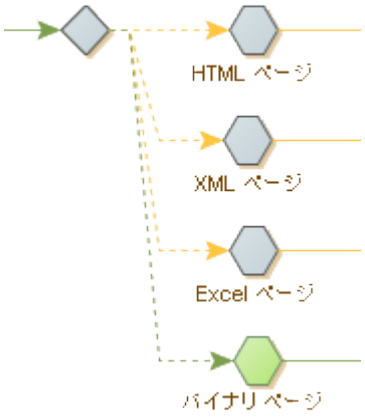
2. [エクスプレッション] フィールドで、エクスプレッションを入力するか、[エクスプレッション] をクリックして、リストから 1 つ選択します。オプションには、定数、変数、演算子、特殊文字、関数、ページ プロパティ、追加のサブエクスプレッション関数が設定されたロボット プロパティが含まれます。
   入力および出力セクションにエクスプレッション値が表示されます。

   テスト機能は、Design Studio のどの場所でも利用できるわけではありません。

3. [**OK**] をクリックします。

ページ タイプの判定

トライ ステップを作成して、ロードされたページのタイプを特定することができます。有効なページ タイプは、HTML、XML、Excel、バイナリなどです。



1. Designer では、ページ読み込みステップの後、トライステップを挿入します。
2. 分岐でアクション ステップを挿入し、**[テスト]** > **[ページ タイプ判定]** を選択します。
3. [アクション] タブの [ページ タイプ] フィールドで **[HTML]** を選択します。
4. [基本] タブでステップ名を **[HTML** ページ] に変更します。
5. トライ ステップを選択し、[ブランチを追加] をクリックします。
6. ページ タイプを **[XML]** に、ステップ名を **[XML** ページ] にそれぞれ設定して、手順 2 ~ 5 を繰り返します。
7. ページ タイプを **[Excel]** に、ステップ名を **[Excel** ページ] にそれぞれ設定して、手順 2 ~ 5 を繰り返します。
8. ページ タイプを [バイナリ] に、ステップ名を [バイナリ ページ] にそれぞれ設定して、手順 2 ~ 4 を繰り返します。

   ロボットを実行すると、ページ タイプがハイライト表示されます。

タグ ファインダーの使用

HTML/XML ページでタグを見つけるには、タグ ファインダーを使用します。タグ ファインダーはステップで使用するのが最も一般的です。ステップでタグ ファインダーを使うことでアクションを適用するタグを見つけることができます。現在のステップのタグ ファインダーのリストは、ステップ ビューの [タグ ファインダー] タブにあります。

タグ パス

タグ パスは、タグのページ内での位置についての短いテキスト表現です。以下のタグ パスについて考えます。

```
    html.body.div.a
```

このタグ パスは、<html> タグ内の <body> タグ内にある タグ内の <a> タグを表します。

タグ パスは、同じページにある複数のタグと照合することができます。たとえば、上のタグ パスは、このページのすべての <a> タグと照合されますが、下の図の 3 番目のタグは除きます。

```
<html>
   <body>
```

```
    <div>
      <a href="url...">Link 1</a>
      <a href="url...">Link 2</a>
    </div>
    <p>
      <a href="url...">Link 3</a>
    </p>
    <div>
      <a href="url...">Link 4</a>
      <a href="url...">Link 5</a>
      <a href="url...">Link 6</a>
    </div>
  </body>
</html>
```

インデックスを使って、そのレベルの同じタイプのタグの中から特定のタグを参照することができます。以下のタグ パスについて考えます。

```
    html.body.div[1].a[0]
```

このタグ パスは、<html> タグ内の <body> タグにある 2 番目の タグ内の 1 番目の <a> タグを表します。したがって、上記のページでは、このタグ パスは "Link 4" の <a> タグとのみ照合されます。タグパスのインデックスは 0 から始まることに注意してください。タグ パス上のタグにインデックスが指定されていない場合、上記の最初のタグのパスで見たように、パスはそのレベルの該当するタイプのあらゆるタグと照合されます。インデックスが負の場合、照合されるタグは、インデックス -1 に相当する最後の一致タグから逆方向にカウントされます。以下のタグ パスについて考えます。

```
    html.body.div[-1].a[-2]
```

このタグ パスは、<html> タグ内の <body> タグにある最後の タグ内の最後から 2 番目の <a> タグを表します。したがって、上記のページでは、このタグ パスは "Link 5" の <a> タグとのみ照合されます。

アスタリスク ('*') を使って、任意のタイプの、任意の順番のタグを表すことができます。以下のタグ パスについて考えます。

```
    html.*.table.*.a
```

このタグ パスは、<html> タグ内の任意の場所に配置された <table> タグ内の任意の場所に配置された <a> タグを表します。任意のタグ パスの前にはアスタリスクがあると見なされるため、"* .table" の代わりに "table" を書いて、ページ上の任意のテーブル タグを表すことができます。唯一の例外はピリオド ('.') で始まるタグ パスで、これはタグ パスの前にアスタリスクがないことを意味します。この場合、タグ パスはページの最初の (トップレベルの) タグから照合する必要があります。

アスタリスクを使用すると、レイアウトに関連するタグなど、時間の経過とともに変化する可能性がある重要でないタグを無視することができるため、ページの変更に対して安定したタグ パスを作成することができます。しかし、アスタリスクを使うと、誤って別のタグを見つける可能性も高くなります。

可能なタグのリストは、次のタグ パスのように、"|" で区切って指定することができます。

```
    html.*.p|div|td.a
```

このタグ パスは、<html> タグ内の任意の場所に配置された <p>、 、または <td> タグ内の <a> タグを表します。

タグ パスでは、ページ上のテキストはキーワードの "text" を使用して、他のタグと同様に表されます。技術的には、テキストはタグではありませんが、タグ パスでは同様に処理されて表示されます。例として、以下の HTML を考えます。

```
<html>
   <body>
     <a href="url...">Link 1</a>
     <a href="url...">Link 2</a>
   </body>
 </html>
```

タグ パス "html.body.a[1].text" は、テキスト "Link 2" を表します。

タグ ファインダーのプロパティ

このトピックでは、タグ ファインダーの設定に使用するプロパティについて説明します。

検索範囲

名前付きタグに関連するタグの検索場所を指定します。デフォルト値は「ページ内の任意の場所」であり、名前付きタグはタグの検索に使われません。

タグ パス

タグ パスについては、タグ パスを参照してください。

属性名

タグには、"align" などの特定の属性が必要です。

属性値

タグには、特定の値がある属性が必要です。[属性名] プロパティが設定されている場合、属性値はその属性名に関連付けられます。

これらの値は大文字と小文字が区別されます。

- 「テキストと等しくする」は、属性値が指定したテキストと完全に一致することを指定します。テキストは属性値の全体と一致する必要があることに注意してください。
- 「テキストを含む」は、属性値に指定されたテキストが含まれることを指定します。
- 「テキストで始まる」は、属性値が指定したテキストで始まることを指定します。
- 「以下のテキストで終了」は、属性値が指定したテキストで終わることを指定します。
- 「次のパターンに一致」は、属性値が指定したパターンと一致することを指定します。パターンは属性値全体と一致する必要があることに注意してください。
- 「テキストと等しくない」は、属性値が指定したテキストと等しくないことを指定します。
- 「テキストを含まない」は、属性値に指定されたテキストが含まれないことを指定します。
- 「テキストで開始しない」は、属性値が指定したテキストで始まらないことを指定します。
- 「テキストで終了しない」は、属性値が指定したテキストで終わらないことを指定します。
- 「パターンと一致しない」は、属性値が指定したパターンと一致しないとを指定します。

タグ パターン

".*.*Stock Quotes.*.*" など、内部のタグもすべて含めてタグが一致する必要があるパターン。ロボットのパフォーマンスに大きな影響を及ぼす可能性があるため、このプロパティを使用する場合は注意が必要です。タグ パターンは、1 つの一致するタグを検索するためにページ全体に何度も適用されることがあるためです。これを回避するには、照合対象プロパティに「テキストのみ」を選択する方法があります。

照合対象

タグパターンは、テキストまたはタグの HTML 全体と照合する必要があります。デフォルトでは、パフォーマンスの高速化のため、テキストのみと照合します。

タグ深度

一致するタグが互いに入れ子になっている場合、どのタグを使用するかを判断します。デフォルト値は [範囲内の深度] です。この値は、一致するタグをすべて受け入れます。[最も外側のタグ] を選択した場合、最も外側のタグのみが受け入れられ、同様に、[最も内側のタグ] を選択すると、最も内側のタグのみが受け入れられます。

タグ番号

複数のタグがタグ パスなどの基準と一致する場合、どのタグを使うかを決定します。使用するタグの数を、一致する最初のタグから順方向、または一致する最後のタグから逆方向のいずれかで数えて指定します。たとえば、タグ パスを "table" に、タグ属性プロパティを "align = center" に設定し、タグ パターン プロパティを ".*Business News.*" に設定すると、タグ ファインダーは、<table> タグで中央揃えされ、"Business News" というテキストが含まれる最初のものが検出されます。

タグ ファインダーの設定

タグ ファインダーのタグ パスを定義するには、複数の方法があります。ブラウザ\HTML\DOM パス ビューを使用してアクションを作成するか、または右クリックして [このタグを使用] または [このタグのみを使用] を選択すると、Design Studio によりパスが自動的に作成されます。または、タグ パスを手動で定義できます。

ページ ビューで、[検索] をクリックし、[検索する文字列] フィールドと [検索範囲] フィールドに入力して、タグ ファインダーによって見つかったタグを表示します。

次のいずれかの方法でタグ ファインダーを設定します。

- コンテキスト メニューからタグ ファインダーを設定するには、ページ ビューでタグを右クリックします。コンテキスト メニューで [このタグを使用] を選択するか、ツールバーの をクリックした場合は、簡単なモードのタグ パスを使用して右クリックしたタグを見つけるようにタグ ファインダーが設定されます。同様に、メニューから別のアクションを選択した場合は、対応するステップが選択され、右クリックしたタグを見つけるようにタグ ファインダーが設定されます。
- タグ ファインダーを自動的に構成するには、ページ ビューでタグを選択し、コンテキスト メニューで [このタグのみを使用] をクリックするか、ツールバーの をクリックします。これにより、簡単なモードのタグ パスを使用して選択したタグを見つけるためのタグ ファインダーが設定されます。
- ステップ アクション用にタグ ファインダーを設定するには、新しいステップを選択します。一部のアクションを選択すると、通常そのアクションに使用されるタグを見つけるようにタグ ファインダーが設定されます。たとえば、フォーム送信アクションでは、1 つのタグ ファインダーを使用して、"form" へのタグ パスが設定され、ページの最初の <form> タグが特定されます。

フォーム送信

フォーム送信はロボットの一般的なタスクです。たとえば、検索フォームを送信して、抽出する検索結果を取得したり、注文フォームを送信して、注文トランザクションを実行したりすることが必要な場合があります。実際にフォームを送信する必要はないが、単にフォーム送信を表す URL を作成したり、フォームの現在の値を変更したりすることが必要な場合もあります。

Design Studio でフォーム送信に推奨される最も簡単な方法は、一般的なブラウザでフォームを送信する方法に似ています。

1. フォームの詳細を入力します。
   次のアクションを使用できます。
   - テキストを入力

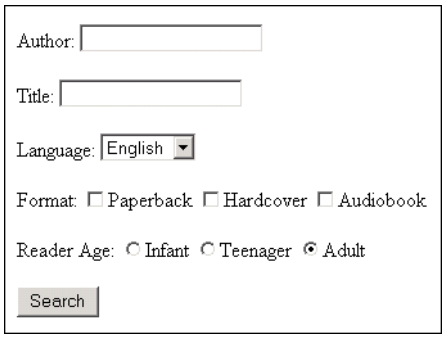
- パスワード入力
- キー プレス
- オプション選択
- 複数オプション選択
- チェックボックス設定
- 範囲値の設定
- ラジオ ボタン選択
- ファイルの選択

2. フォーム送信ボタンをクリックします。
   次のアクションを使用して、フィールド値 (テキスト入力) オプションまたはラジオ ボタンをループすることができます。
   - フィールド値ループ
   - セレクト オプション繰り返し
   - ラジオ ボタン繰り返し

フォームの基本

以下のような書籍検索フォームの例を考えます。最初に HTML で示し、次にブラウザで示します。

```
<html>
     <body>
       <form action="http://www.books.com/search.asp" method="get">
         Author:
         <input type="text" name="book_author">
         <p>
         Title:
         <input type="text" name="book_title">
         <p>
         Language:
         <select name="book_language">
         <option value="lang_0" selected>English</option>
         <option value="lang_1">French</option>
         <option value="lang_2">German</option>
         <option value="lang_3">Spanish</option>
         </select>
         <p>
         Format:
         <input type="checkbox" name="book_format" value="format_pb">Paperback
         <input type="checkbox" name="book_format" value="format_hc">Hardcover
         <input type="checkbox" name="book_format" value="format_ab">Audiobook
         <p>
         Reader Age:
         <input type="radio" name="reader_age" value="age_inf">Infant
         <input type="radio" name="reader_age" value="age_teen">Teenager
         <input type="radio" name="reader_age" value="age_adult" checked>Adult
         <p>
         <input type="submit" value="Search">
       </form>
     </body>
   </html>
```



フォームにはフィールドが多数あります。たとえば、サンプル フォームの最初の <input> タグは、"book_author" という名前のフィールドを定義します。フィールド名は、通常、ユーザーがブラウザで見るものとは異なることに注意してください。たとえば、"book_author" フィールドは、"book_author" ではなく、ブラウザに "Author" という名前で表示されます。

フィールドは、複数のタグで定義することができます。たとえば、"book_format" フィールドは、サンプル フォームの 3 つの <input> タグで定義されます。同じフィールド名を使用し、フィールド タイプ (テキスト フィールド、ラジオ ボタン、チェック ボックスなど) が同じタグは、同じフィールドを定義します。

フィールドには、1 つ以上の値を割り当てることができます。たとえば、"book_format" フィールドには、ペーパーバックを選択するための値 "format_pb" を割り当てることができます。フィールド名と同様に、フィールドに割り当てられた値は、通常、ユーザーがブラウザで見る値とは異なることに注意してください。たとえば、ペーパーバックを選択すると、ユーザーには値 "format_pb" ではなく、テキストの "Paperback" が表示されます。フィールド タイプによっては、複数のフィールドに同時に複数の値を割り当てることができます。たとえば、"book_format" はチェック ボックス フィールドであるため、"book_format" フィールドに値 "format_pb" と値 "format_hc" を割り当て、ペーパーバックとハードカバーを選択できます。

ほとんどのフィールドにはデフォルト値が入っています。デフォルト値は、フォームのフィールドに最初に割り当てられる値です。たとえば、"book_language" フィールドには "selected" 属性があるため、デフォルト値 "lang_0" が選択されています。

フォームは、フィールドの現在の値を Web サイトに送ることによって送信されます。現在の値が 1 つ以上あるフィールドのみが送信されます。たとえば、サンプルフォームの "book_format" フィールドのチェックボックスのいずれもチェックされていない場合、フィールドに対する値は送信されません。

ブラウザでは、フォーム送信は通常、ユーザーが送信ボタンをクリックすると発生します。送信ボタンには、通常の送信ボタンとイメージ送信ボタンの 2 種類があります。通常の送信ボタンは、<button> タグまたは <input> タグで "type" 属性を "submit" に設定することで定義されます。通常の送信ボタンにフィールド名と値がある場合、ボタンをクリックすると、そのフィールドが指定された値で送信されます。

画像送信ボタンは、<input> タグで "type" 属性を "image" に設定することで定義されます。画像送信ボタンは、"button name.x" と "button name.y" という名前で 2 つのフィールドを定義しますが、ボタン名は、<input> タグの "name" 属性に含まれる名前になります。<input> タグに "name" 属性がない場合、フィールドの名前は "x" と "y" になります。画像送信ボタンがクリックされると、これらの 2 つのフィールドには、マウスがクリックされた画像内の位置を表す x 座標と y 座標が割り当てられます。一部の Web サイトでは、ユーザーがクリックした位置に応じて異なる動作のイメージ マップを作成するためにこのボタンが使用されています。

JavaScript を使用するフォームもあります。たとえば、<form> タグは フォームが送信される前に実行される JavaScript を含む "onsubmit" 属性を持つことがあります。同様に、<input> タグは、ユーザーがフィールドをクリックしたときに実行される JavaScript を含む "onclick" 属性を持つことができます。ロボットはこの JavaScript を自動的に実行します。

パフォーマンス上の理由から、フォーム送信時に JavaScript の実行を無視することができます。これには、フォーム送信ステップのオプションで JavaScript の実行オプションの選択を解除する必要があります。

ステップ アクションの決定

最も単純なフォーム送信方法は、適切なアクションを使用して、フォームに入力することです。より複雑な送信の場合は、フォームをループ スルーすることで必要な結果を得ることができます。

本の検索例フォームがあるとします。

- すべての言語で利用でき、全年齢を対象にした本を検索する場合、サイトでそのような一般検索はできないことがあります。言語と年齢の組み合わせごとにフォーム送信を行いながら、言語と読者年齢をループ スルーできます。これを実行するには、次のループ フォーム アクションを使用します。
  - フィールド値ループ
  - セレクト オプション繰り返し
  - ラジオ ボタン繰り返し
- ループ フォーム アクションではフォームは送信されないため、フォーム送信ボタンのそれぞれをクリックするクリック アクションを別個に続けることで実行する必要があります。
- フィールド値の組み合わせをループ オーバーするには、フォームを送信するクリック アクションの前に、複数のステップとループ フォーム アクションを交互に配置します。
- フォーム送信先を表す URL を作成するには、フォーム送信ボタンに URL 抽出アクションを使用します。

フォームでのループの使用

フォームでは次の 3 つのループ アクションを使用できます: フィールド値ループ、セレクト オプション繰り返し、ラジオ ボタン繰り返し。これらの 3 つのアクションは、テキスト入力 ("text" タイプの INPUT エレメントと TEXTAREA エレメント)、オプション (SELECT エレメント)、ラジオ ボタン ("radio" タイプの INPUT エレメント) という 3 種類のフォーム コントロールに対応しています。これらのループの使用方法については、次のビデオを確認するか、以下のセクションをお読みください。

フォームをループ オーバーするには、ループ オーバーするフォーム コントロールとその順序 (この順序により、出力値が生成される順序が決まります) を決定する必要があります。次に、対応するフォーム アクションのループがある各フォーム コントロールについて、ステップを挿入します。ステップを挿入するには、ページ ビューのコントロールを右クリックして、コンテキスト メニューから **[ループ] > [<フォーム アクション>]** を選択します。この [<フォーム アクション>] は適切なアクションに相当します。たとえば、コントロールがテキスト入力コントロールの場合は、**[ループ]** > **[フィールド値ループ]** を選択します。

ループ フォーム アクションを実行するたびに、HTML ページのフォーム コントロール エレメントで値が変更されます。この変更は、ブラウザ内において手動で行う変更に相当します。フォーム コントロールに JavaScript イベントが付加されている場合、イベントが発生し、一部の JavaScript が実行されます。この JavaScript によって、SELECT エレメントのオプションなどのフォームが変更される場合があります。この場合、コントロールをループ オーバーする正しい順序を注意して選択し、ロボットが適切な

オプションを必要なときに利用できるようにする必要があります。通常、ブラウザ内において手動で行うときに使用する順序を選択すると、適切に動作するはずです。

ロボットにループ フォーム アクションのすべてのステップを挿入したら、フォームのいずれかの送信ボタンをクリックするクリック アクションのステップを追加する必要があります。

ファイルのアップロード

一部のフォームには、ファイルのアップロードに使用できるファイル フィールドが含まれています。ファイル フィールドは、次のような、<input> タグのタイプのファイルで定義されます。

```
<INPUT type="file" name="attachedFile">
```

ファイル選択アクションでは、ファイル フィールドを使用してファイルをアップロードする方法が 2 つあります。

1. 1 つ目は、ファイル システムからファイルをアップロードする方法です。このアップロードを実行するには、リストから [ローカル ファイル システム内のファイル] を選択し、ファイル名を入力します。フォームを送信すると、指定したファイルがファイル システムからロードされ、フォーム送信の一部としてアップロードされます。

   ファイル名は、ドライブ名 (ある場合) とファイルへのディレクトリ パスを含む絶対ファイル名である必要があります。

2. 2 つ目の方法はファイルをアップロードする最も一般的な方法で、ファイル システムからファイルをロードする代わりに、アップロードするファイル コンテンツを指定します。これを行うには、リストから [変数に含まれるファイル] を選択します。次に、ファイル コンテンツ リストからファイル コンテンツを保持している変数を選択することもできます。通常、以前にターゲット抽出アクションを使用してファイルをダウンロードしたバイナリ変数から、または以前に抽出したテキストを含む変数からコンテンツを取得します。

オプションで、ファイルのコンテンツ タイプとファイル名を指定できます。コンテンツ タイプは、MIME タイプのコンテンツである必要があり、オプションで文字セットが続きます。あらかじめ定義されたいずれかのコンテンツ タイプを使用できます。コンテンツ タイプを属性から取得するか、カスタム コンテンツ タイプを指定します。たとえば、画像の場合、コンテンツ タイプは以下のようになります。

```
image/gif
```

プレーン テキストの場合、コンテンツ タイプは以下のようになります。

```
text/plain; charset=iso-8859-1
```

ターゲット抽出を使用してファイルをダウンロードする場合、ダウンロードしたデータのコンテンツ タイプとファイル名を他の変数に格納できることに注意してください。これで、ファイル選択アクションでファイルをアップロードするときに、この情報を使用できます。

ページ ビューでのコンテキスト メニューの使用

フォーム送信 アクションとループ フォーム アクションを選択して設定するためのショートカットとして、ページ ビューでコンテキスト メニューを使用することができます。

これは古いロボットでのみ利用可能な、現在は使用されていない機能です。

1. 現在のステップで [**Submit Form**] または [**Loop Form**] アクションを選択するには、ページ ビューで <form> タグ内を右クリックします。
2. コンテキスト メニューの [フォーム] サブメニューで、[フォーム送信の使用] または [ループ フォームの使用] を選択します。
3. 現在のステップにフォーム送信 アクションまたはループ フォーム アクションが含まれる場合、ページ ビューの [フォーム] サブメニューで、フィールドを右クリックし、[フォーム] サブメニューの [フィールドへのアサインメントの追加] を選択します。

   ダイアログ ボックスが表示されます。
4. フィールド値を割り当てます。
5. 現在のステップにループ フォーム アクションが含まれる場合、[フォーム] サブメニューで、フィールドを右クリックし、[ループ オーバー フィールドの追加] を選択してフィールドをループします。
6. ダイアログ ボックスが表示され、ここでフィールドに [ループする値を持つ **1** つのフィールド] フィールド グループを設定できます。
7. 現在のステップに [**Submit Form**] または [**Loop Form**] アクションが含まれる場合、ページ ビューで送信ボタンを右クリックします。
8. [フォーム] サブメニューで、[サブミットの選択] をクリックします。

ページのループ スルー タグ

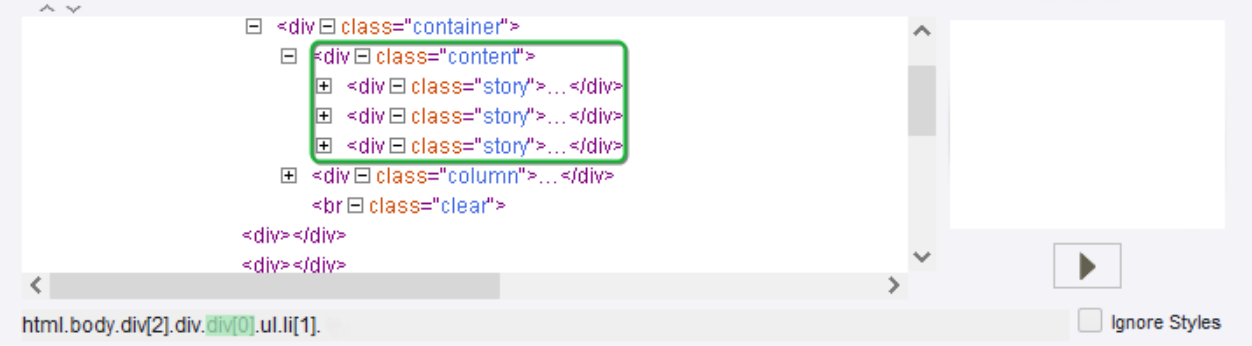
ロボットは、各エレメントで何らかのアクションを実行するために、ページ上のエレメントをループする必要が頻繁にあります。たとえば、検索の各結果から、あるいは表の各行から取得することができる特定のプロパティの抽出が必要な場合があります。以下のトピックでは、その方法について説明します。

- class の同じタグのループ
- class の異なるタグのループ

複数の異なるタイプのループ ステップでは、同じ状況を処理する方法がいくつもあります。

class の同じタグのループ

いくつかの方法でループを設定できます。第 1 の方法は、実行可能であれば最も簡単な方法で、同じ class 属性を共有しているタグをループします。



各 div エレメントには属性 `class="story"` があります。

class の同じタグをループできるかどうかを判定するには、HTML ビューのエレメントを見つけます。上記の場合、属性 `class="story"` を持つ 3 つの div タグをループできます。

1. 最初のタグを右クリックし、**[ループ]** > **[クラス付タグ繰り返し]** > **story** を選択します。
   これにより、ロボットにタグ パス繰り返しステップが作成され、指定した class を持つページですべてのエレメントがループされます。
2. ループ ステップで、矢印を使用して、ループに正しいタグが含まれていることを確認します。
   指定した class を使用するページ内の他のタグを、ループに含めたくない場合があります。簡単な修正を加えて、これらのタグをループから除外できます。
3. 選択したクラスのタグを除外するには、エディターの **[アクション]** タブで、**[ループ]** > **[タグ パス繰り返し]** の順に選択します。
   HTML ビューをレビューします。
   タグ パス繰り返しでは、ページ全体が見つかったタグとして自動的に含まれています。
4. 見つかったタグを変更して、ロボットが強制的に指定した別のタグ内のタグのみをループするようにします。
   ループが正常に作成されたら、ループの各イテレーションに対して、[タグ パス繰り返し] ステップの後に追加されたステップが繰り返されます。
   各イテレーションに対して、ループ ステップの後のステップが実行されます。
   ロボット ビューが表示されている上記の例では、ループの各イテレーションに対して、ロボットが 2 つのテキスト (タイトルとプレビュー) を抽出し、それらの値を返しています。

class の異なるタグのループ

一般的なシナリオでは、class が同じタグをループしますが、実際のシナリオはそのように単純ではありません。多くの場合は、class が同じでないタグをループする必要があります。ここで、別のシナリオが必要になります。

class を含むタグ繰り返しが失敗するほとんどのケースで、タグ繰り返しステップは非常に効率的です。タグ繰り返しステップは、見つかったタグ内に直接存在するすべてのタイプのタグをループします。これを行うには、右クリックして挿入することに加えて、追加の設定が少し必要です。このステップの使用方法を次に示します。

[タグ繰り返し] を使用して、指定したタイプの各タグ (見つかったタグ内に直接存在する) をループします。

見つかったタグには 3 つの div タグが含まれていますが、すべての div タグが同じクラスを持つわけではないことに注意してください。このシナリオでは、[タグ繰り返し] を使用して、この差異に対応します。

1. 空の新しいステップ を挿入し、[タグ繰り返し] アクションを選択します。
2. ページ ビューで、見つかったタグを選択します。
3. [ステップ アクション ビュー] の [タグ] フィールドで、タグのタイプを選択します。
4. タグ繰り返しステップの後にステップを追加します。
   これらのステップは、ループの各イテレーションに対して繰り返されます。

HTML ページのループ

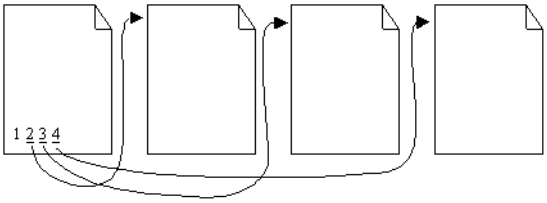
ロボットは頻繁にページをループする必要があります。たとえば、検索要求の結果が含まれる Web サイトでは、多くの場合、各ページに 20 件といったように複数のページにわたる検索結果が表示されます。検索結果を取得するには、ページをループして、一度に 1 ページを処理する必要があります。以下のトピックでは、この方法について説明します。

- 最初のページにある他のすべてのページへのリンク
- 次ページにリンクしている各ページ

最初のページにある他のすべてのページへのリンク

最初のページに他のすべてのページへの直接リンクが含まれている場合、リンクを使用して、最初のページからすべてのページに直接アクセスできます。

最初のページには、最初のページへのリンクが含まれている場合もあります。



この例では、ロボットからの抜粋に示したように、タグ繰り返しステップを使用してページを簡単にループできます。

この図では、ロボットは、フォーム送信 ステップで表される検索リクエストから結果ページをループしています。

フォーム送信ステップからページ処理ステップへの直接接続で示されているように、最初の結果ページが直接処理されます。

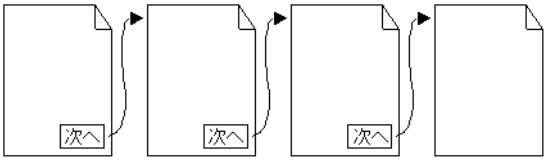
残りのページは、2 番目の分岐のタグ繰り返しアクションを使用してループします。

タグ判定ステップでは、複数のページがあることが確認されます。

- 最初のページが複数のページにリンクされている場合、次の処理を実行します。
  1. ページへのリンクが含まれているタグをループします。
  2. クリック アクションを使用して各ページを読み込みます。
  3. ページの処理を続行します。
- 最初のページが最初のページ自身にリンクされている場合、次の処理を実行します。
  1. この最初のリンクをスキップするようにタグ繰り返しアクションを設定します。
     最初のページが処理されるのは 1 回だけです。

次ページにリンクしている各ページ

ページが以降のページにリンクされている場合です。通常、ページ下部に次のページへのリンクがあります。



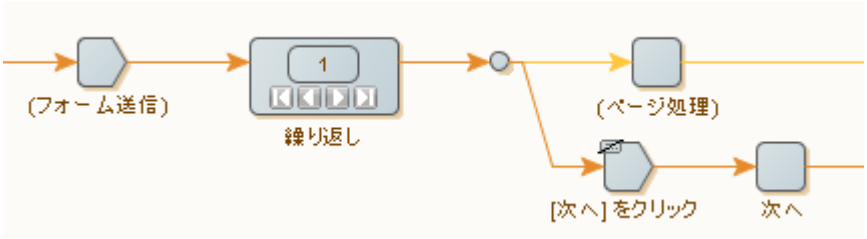
繰り返しアクションを使用して、こうしたページをループします。このアクションは、「次へ」という名前の別のアクションにより、指定されているページをループします。

「繰り返し」と「次へ」は、連携して使用しないと効果がありません。

1. 最初のページで、「繰り返し」ステップを追加します。
2. 必要に応じて、追加のアクションを挿入します。
3. 「次のステップ」を挿入します。

   ロボット実行が「次のステップ」に到達すると、「繰り返し」ステップに戻り、ステップの別のイテレーションを実行します。ページは「繰り返し」ステップに転送され、各イテレーションで新しいページがロードされます。

   必要に応じて、「繰り返し」ステップと「次のステップ」の間に他のループを追加し、ページから追加の情報を抽出できます。

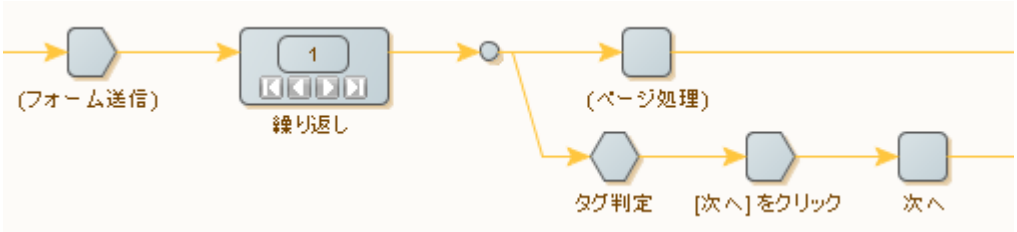
   ここで、これまでのように、フォーム送信ステップで表される検索リクエストから結果ページをループします。

   フォーム送信ステップは、最初の結果ページを出力します。このページを繰り返しアクションに提供します。繰り返しアクションの最初の分岐で、現在のページを処理します。2 番目の分岐で、次のページのリンクをクリックして、次のページを読み込みます。次のアクションでは、ページが繰り返しアクションに戻され、次のイテレーションへ移動で出力されます。最後のページに到達すると、クリック アクションによりエラーが生成されます。これを実行するには、ループを終了するようにクリック ステップを設定します。クリック ステップでは、[エラー処理] タブで「次に」プロパティを「ループ終了」に設定すると、ループが終了します。
4. ループを終了するには、[エラー処理] タブで、[次の処理] プロパティを [ループ終了] に設定します。

   プロセスで次のページが見つからない場合、プロセスが終了します。

   詳細については、エラーの処理を参照してください。

   以下のロボットの例は、最後のページを処理する別の方法を示しています。

   2 番目の分岐でタグ判定アクションを使用して、最後のページにいつ到達したかを検出できます。タグ判定アクションで、たとえば、テキスト "Next" が含まれる <a> タグを検索して、ページに次のページへのリンクが含まれていることをチェックします。ページにそのようなリンクが含まれて

いる場合、このページを読み込んで、次のアクションに提供します。最後のページに到達すると、タグ判定アクションは 2 番目の分岐で実行を停止し、新しいページが繰り返しアクションに提供されなくなるため、ループが終了します。

次のページへのリンクを見つけるときは、注意が必要な場合があります。一般的な間違いは、最初のページ、以降のページ、最後のページの間でページのレイアウトがわずかに異なっているために、次のページへのリンクではなく、一部のページで前のページへのリンクを見つけてしまうことです。もう 1 つの一般的な間違いは、最後のページを検出しないことです。ステップのタグ ファインダーの適切な動作のために、慎重な設定が必要になる場合があります (タグ ファインダーの使用を参照してください)。

Design Studio でロボットを操作する場合、繰り返しアクションのイテレーション間を適切に往復できない場合があります。Design Studio が適切に動作しているかどうかが不明な場合は、[更新] をクリックして更新します。

待機基準の使用

[次の時に続行] オプションの待機基準は、ロボットを迅速かつ信頼性の高い方法で構築するのに役立つ強力な方法です。

待機基準は、デフォルトのブラウザ エンジンを使用しているときに利用できます。

ブラウザの実行開始を要求するすべてのロボット ステップが一連の基準で設定され、アクション (クリックやページ読み込みなど) の処理が完了してロボットを続行できるようになったタイミングを正確に特定できます。

組み込みアルゴリズムと組み合わせて [次の時に続行] 機能を使用すると、ロボットを続行するために必要なときにのみブラウザ ステップを実行できます。また、ユーザーは、ページ上の要素をポイントしてクリックし、ブラウザ ステップの新しい停止基準を作成できます。

次のステップで待機基準を指定できます。

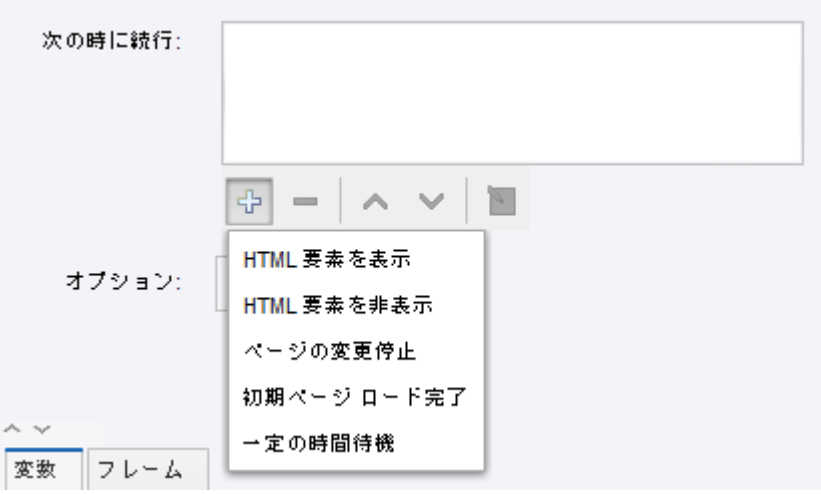
- クリック
- ウィンドウを閉じる
- ページ生成
- パスワード入力
- テキストを入力
- JavaScript の実行
- URL 抽出
- セレクト オプション繰り返し
- ラジオ ボタン繰り返し
- タグ挿入
- ファイル読込
- ページ読み込み
- フィールド値ループ
- マウス アウト
- マウス オーバー

- キー プレス
- タグ書き換え
- スクロール
- 指定タグまでスクロール
- ファイル選択
- 複数オプション選択
- オプション選択
- ラジオ ボタン選択
- チェックボックス設定

Kofax RPA バージョン 9.6 以降で作成されたすべてのロボットでは、デフォルト待機が [ページは 500 ミリ秒の間、変更を停止します] に設定されています。この設定は、[ロボットの設定] ウィンドウの [詳細] タブに表示されます。待機基準が有効化されたすべてのステップには、デフォルトの [500 ms 間ページが変更されていません] 待機基準と、[Result View] に表示される有効な [再開] ボタンがあります。この待機基準は [待機] ビューに常に灰色で表示され、常に満たされています。その他のすべてのブラウザ ステップには、デフォルトの [初期ページ読み込み完了] 待機基準と無効化された [再開] ボタンがあります。この待機基準は、[待機] ビューに常に灰色表示されて満たされています。

待機基準の追加

ステップの待機基準を指定するには、[次の時に続行] フィールドの [+] をクリックし、基準を選択します。



待機基準が有効化されたステップの実行後にブラウザ ビューやソース ビューから待機基準を追加するには、ブラウザ ビューまたはソース ビューを右クリックして、メニューから [次を待機] を選択し、基準を選択します。基準が追加された後、ステップが再実行されます。これは、待機基準の設定ウィンドウの [ページ読込を再実行] ボタンで示されます。

基準を追加したら、リストの下のパネルを使って、待機基準を追加、除去、上へ移動、下へ移動、および編集することができます。[次の時に続行] リストの待機基準を右クリックすると、基準のコピー、切り取り、貼り付けを実行するためのオプション メニューが表示されます。

複数の待機基準をステップに追加することができます。待機基準がいくつかある場合、いずれかの待機基準を満たすと実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機している

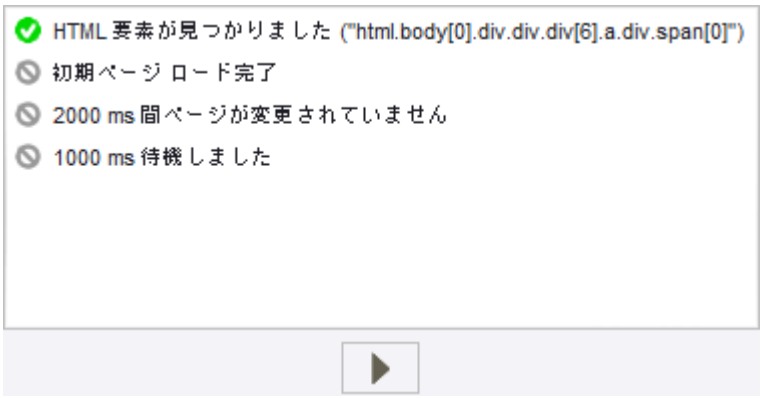
ときやメイン フレームの工レメントを待機しているときなど、[初期ページ読み込み完了] が設定されている場合は、条件を満たす複数の待機基準がある可能性があります。

ショートカット メニューから待機基準が無効化されたステップに待機基準を追加すると、待機基準が有効化された以前のステップに基準が追加されます。たとえば次の例のように、抽出ステップの後に待機基準の追加を試みると、

ページ読み込みステップに基準が追加されます。

待機ビュー

[待機] ビューには、待機基準の実行結果と無効化された待機基準が表示されます。



リストの基準を右クリックすると、ショートカット メニューが開きます。基準を有効化、無効化、および削除したり、選択した基準のプロパティを開いたりできます。[**HTML** 要素を表示] の場合、[ブラウザ] ビューの DOM で見つかった工レメントを選択できます。

また、このビューには、ページが完全にロードされたかどうかが表示されます。ページが完全にロードされた場合は、[再開] ボタンが無効化されます。タイムアウトが短いためにページが完全にロードされていない場合は、[再開] ボタンが無効化されるため、タイムアウトを延長する必要があります。

待機基準が満たされた後、ブラウザ操作を再開するには、[待機] ビューの [再開] ボタンを使用します。複数の待機基準があり、そのいずれかが満たされている場合に [再開] ボタンをクリックすると、満たされた待機基準が灰色の記号でマークされ、次の待機基準が満たされるまでブラウザの動作が継続します。すべての待機基準が満たされた場合、[再開] ボタンをクリックすると オプションダイアログ ボックスの [各試行のタイムアウト] オプションで指定されている時間内にページのロードが開始されます。

[再開] ボタンは、待機基準が有効化されたステップに対して有効化されます。

デフォルトの待機基準の場合、[再開] ボタンを必要な回数クリックすることができます。デフォルト以外の待機基準の場合、[再開] ボタンを一度だけクリックできます。

アイコンなしで待機基準が表示された場合、その基準は満たされていません。

待機基準のプロパティ

[初期ページ読み込み完了] 以外の各待機基準には設定があります。待機基準を設定するには、[待機] ビューまたは [次の時に続行] ビューで基準をダブルクリックするか、[次の時に続行] ビューで [] をクリックするか、または [待機] ビューで基準を右クリックして [プロパティ] を選択します。

待機基準の削除

待機基準の設定ウィンドウで待機基準を無効化または有効化することができます。デフォルトでは、すべての基準が有効になっています。待機基準を無効にするには、[基準が有効になるのを待機] チェック ボックスをオフにします。待機基準を無効化すると、ステップの実行中に待機基準が考慮されません。

待機基準を右クリックし、ショートカット メニューを使用して、基準を無効化および有効化することができます。

待機基準を無効化または有効化した後、前のステップが再実行されます。

基準が満たされたときすべての保留ロードを無視

[初期ページ ロード完了] 以外の各待機基準には、待機基準が満たされたときにページのロードを停止する [基準が満たされたときすべての保留ロードを無視] オプションがあります。このオプションは、待機基準がすでに満たされていても、タイマーが実行を継続していて、ロードが停止しない場合に役立ちます。デフォルトでは、このオプションは選択されていません。このオプションによってブラウザが停止した場合、[待機] ビューの緑のアイコンに警告マークが追加されます。

HTML 要素を表示

この基準は、指定された HTML エレメントが DOM ツリーに存在するときに満たされます。この基準設定は、「ステップの設定」の [タグ ファインダー] タブと類似していますが、追加のプロパティ [検知された要素は次である必要がある] には、次の 2 つのオプションが含まれています。

- 有効にする: このオプションを選択した場合、`result = !element.disabled;` のときに実行を停止する必要があります。
- 表示: このオプションを選択した場合は、`result = style.display !== "none" && style.visibility !== "hidden";` のときに実行を停止する必要があります。

[HTML 要素を表示] 基準が満たされている場合、[待機] ビューのオプション メニューで [ブラウザ ビュー内を選択] コマンドを使用するときに、ブラウザ ビューおよびソース ビューでこの基準がマークされます。

HTML エレメントを非表示

この基準は、指定された HTML エレメントが DOM ツリーに存在しないときに満たされます。この基準の設定は、「ステップの設定」のタグ ファインダー タブに類似しています。ただし、追加のプロパティ [最初の要素の検知] には、2 つのオプションが含まれています。

- [ページ内に要素が見つかりました]: ロボットは、要素がページに表示されるまで待機し、その後、要素が DOM ツリーに表示されなくなるまで待機します。
- [一定の時間待機]: ロボットは、指定した時間待機し、要素が DOM に存在するかどうかを確認します。
  - DOM ツリーにエレメントが存在する場合、ロボットは、そのエレメントが表示されなくなるまで待機します。
  - DOM ツリーにエレメントが存在しない場合は、エレメントがページ読み込みの開始として DOM に表示されていなくても、待機基準が満たされ、ロボットは次のステップに進みます。

ページの変更停止

この基準は、指定された時間内に DOM ツリーが変更されない場合に満たされます。時間を設定するには、基準のプロパティを開き、[タイムアウト **(ms)**] テキスト ボックスにタイムアウトをミリ秒単位で指定します。

初期ページ読み込み完了

この待機基準は、Javascript onload イベントの場合のように初期ページ読み込みが完了したときに満たされます。

この基準には [基準が満たされたときすべての保留ロードを無視] オプションがありませんが、デフォルトでは、すべてのロードが満たされるとロードが停止します。

一定の時間待機

この待機基準は、実行を指定された時間待機しているときに満たされます。時間を設定するには、基準のプロパティを開き、[待機 **(ms)**] テキスト ボックスに時間をミリ秒単位で指定します。

Kofax RPA 9.6 以降での古い口ボット

9.6 よりも前の Kofax RPA バージョンで作成されたデフォルトのブラウザ ロボットを開く場合は、[ロボットの設定] ダイアログボックスの [詳細] タブにある [デフォルト待機] 設定を参照してください。

以前のリリースで作成されたロボットの [デフォルト待機] 設定は [バージョン 9.6 以前のデフォルト待機を使用] です。このようなロボットの場合、この設定を [ページは 500 ミリ秒の間、変更を停止します] に変更できます。

- [デフォルト待機] が [バージョン 9.6 以前のデフォルト待機を使用] に設定されている場合、新しい待機基準をステップに追加すると、以下の警告が表示されます。

  待機基準を使用する場合、ロボット設定で [デフォルト待機] を [ページは 500 ミリ秒の間、変更を停止します] に設定する必要があります。
- [デフォルト待機] が [バージョン 9.6 以前のデフォルト待機を使用] に設定されていて、[レガシー タイミング] 待機基準を使用するようにステップが設定されている場合、[デフォルト待機] を [ページは 500 ミリ秒の間、変更を停止します] に変更すると、エラーが表示されます。

  [レガシーの待機基準を使用する際は、ロボット設定で 'デフォルト待機' を 'バージョン 9.6 以前のデフォルト待機を使用' に設定する必要があります。]

9.6 への既存の口ボットのアップグレード

Kofax RPA 9.6 (9.3、9.4、9.5) よりも前のバージョンで作成されたロボットを開く場合、[タイマー イベントの最大待機時間] および [タイマー イベントをリアルタイムで待機] 設定に応じて、異なるステップを実行して、ロボットで新しい待機基準を使用する必要があります。

[タイマー イベントの最大待機時間] および **[タイマー イベントをリアルタイムで待機]** 設定はデフォルトではありません。

ロボットで [タイマー イベントの最大待機時間] および [タイマー イベントをリアルタイムで待機] 設定がデフォルトとして設定されていない場合、Design Studio により、編集不可の [レガシー タイミング] 基準がロボットに追加されます。

この待機基準は、ステップの実行後、常に緑になります。新しい待機基準を追加すると、[レガシー タイミング] が自動的に除去され、新しい待機基準を使用できるようになります。

[タイマー イベントの最大待機時間] および **[タイマー イベントをリアルタイムで待機]** 設定はデフォルトです。

ロボットで [タイマー イベントの最大待機時間] および [タイマー イベントをリアルタイムで待機] 設定がデフォルトとして設定されている場合、既存のロボットを 9.6 にアップグレードするときに、次の手順を実行して新しいデフォルト設定に切り替える必要があります。**[ファイル]** > **[ロボット設定]** ダイアログ ボックスに移動し、[詳細] タブをクリックし、[バージョン **9.6** 以前のデフォルト待機を使用] チェック ボックスをオフにします。これで、ロボットで新しい待機基準が使用できるようになります。

ページ読み込みの修正
ロボットで [バージョン 9.6 以前のデフォルト待機を使用] または [レガシー タイミング] オプションを使用するときに、ページのロード後、[ページ読み込み完了] メッセージが表示されてもページが完全にロードされない場合があります。この現象は、前のページ読み込みが中止された場合に発生することがあります。ページ読み込みを修正するには、[バージョン 9.6 以前のデフォルト待機を使用] オプションを [ページの変更停止] オプションに変更することで除去し、ステップで新しい待機基準を使用します。

HTML からのコンテンツの抽出

HTML ページ内のタグからコンテンツを抽出する方法として Design Studio には、次の 6 つのステップがあります：

- 抽出アクションは、タグからテキスト コンテンツを抽出する場合に使用し、オプションで HTML タグも含めることができます。
- **URL** 抽出アクションは、URL を含むタグ属性から URL を抽出し、その URL を完全なものにするために使用します。
- タグ属性抽出アクションは、タグ属性の値を抽出するために使用します。
- ターゲット抽出アクションは、画像や PDF ファイルなどのバイナリ データを抽出するために使用しますが、あらゆる種類のバイナリ データを処理します。
- フォームパラメータを抽出アクションは、見つかったタグのフォーム URL からパラメータから抽出し、その値を変数に格納するために使用します。
- 選択済みオプション抽出アクションは、選択済みオプションを <select> タグから抽出し、変数に格納するために使用します。

抽出したコンテンツを再度書式設定 (または正規化) するには、抽出とタグ属性抽出を用いて、データ コンバータのリストを設定します。

[**PDF** から抽出] アクションは、選択した属性のバイナリ データとして含まれる PDF ドキュメントからテキストを抽出する場合に使用します。データを抽出し、ロボットがデータにアクセスできるように構造化された形式のデータを含む HTML ページを生成します。このアクションは、実際にデータを抽出する前の初期ステップで使用され、生成された HTML をループしてテキストを抽出することができます。

テキスト抽出

1. [アクション] タブで、[抽出] を選択します。
2. 製品名や価格などの短いテキストを抽出するには、[テキストのみ] として抽出します。
   これにより、タグの間にあるテキストが抽出されます。

3. セクションや見出しなどがある長いテキストを抽出するときは、プレーン テキストとして選択できます。ブラウザに表示される状態に近い状態でテキストを表示する場合は、テキストを [構造化テキスト] として抽出します。
4. 見出しに付いているかっこなど、特別なマークアップとともに抽出するには、[構造化テキスト] を選択します。
   構造化テキストは、特別なマークアップの基本的なサポートを備えています。
5. [構造化テキスト] オプションでマークアップの要件を満たすことができない場合は、[高度な構造化テキスト] を選択します。
   このオプションを使用すると、HTML タグのマッピングを専用のマークアップに設定できます。

バイナリ データの抽出

ターゲット抽出アクションを使用して、バイナリ データを抽出できます。

[アクション タブ] で、[ターゲット抽出] を選択します。

URL データがロードされ、変数に格納されるか、ファイルに直接保存されます。

通常、バイナリ変数を使用して、ロードされたデータを格納します。バイナリ変数の利用可能なタイプには、バイナリ、画像、PDF、セッションが含まれます。これらのタイプは、画像、PDF、セッションの各データ タイプがプレビュー可能であるという点を除き、すべて同等です。

ページ ビューでコンテキスト メニューを使用

1. 抽出元のテキストを右クリックするか、ロード元のリンクを右クリックします。
2. 抽出コンテキスト メニューから適切なオプションを選択します。

一般的なタスクの実行

テキストの一部のみを抽出

タグ内のテキストを一部のみ抽出するには、タグ内のテキストにパターンを使用します。たとえば、次のテキストから名前 "Bob Smith" を抽出するとします: "The article is written by Bob Smith."抽出するには、抽出データ コンバータ (抽出ステップと混同しないでください) を使用します。抽出データ コンバータは、このトピックで説明しているとおりに設定する必要があります。

この例では、使用されているパターンは ".*by\s(.*)\." です。これは、"by" とピリオドの間のテキストは、サブパターンによって照合されることを意味します。詳細については、パターンを参照してください。

1. 抽出設定を開き、[基本] タブを選択します。
2. [パターン] フィールドに、抽出するテキスト パターンを入力します。
   パターン プロパティを、かっこでくくられているサブパターンで抽出対象のテキストを照合し、テキスト全体に対して照合するように設定します。

コンテンツの変換

コンテンツを正規化するには、テキストを別のテキストに置換するなど、変換を使用します。たとえば、「US」から「United States」に正規化するなど、国コードを自然言語の説明に正規化します。

- プレーン テキスト変換の場合は、[リストを使用して変換] データ コンバータを使用します。

- パターンまたはエクスプレッションに基づいた変換の場合は、[If Then] データ コンバータを使用します。

番号の抽出と書式設定

1. コンテンツから番号を抽出するには、[数値を抽出] データ コンバータを追加します。
2. 番号の追加の書式設定を実行するには、[数値の書式設定] データ コンバータを使用します。

テキストから日付抽出

日付抽出は、番号の抽出と同じように実行する必要があります。

1. テキストから日付抽出するには、ロボットに [日付抽出] データ コンバータを追加します。
   日付抽出では、パターンを使用して日付抽出します。パターンは、テキスト全体ではなく、日付のみに一致する必要があります。抽出された日付は、標準の日付書式に変換されます。
2. 追加の日付書式設定を実行するには、日付の書式設定データ コンバータを使用します。

見つかったタグ内のタグのサブセットを抽出

単一のタグではなく、タグの範囲から抽出する必要がある場合があります。

たとえば、記事の本文を抽出する場合について考えてみます。この本文は独自のタグ内にある個々のセクションで構成され、記事のタイトルと作成者についての情報は他のタグに含まれています。記事のタイトルと作成者なしに、本文のみを抽出するには、抽出アクションを使用してテキストを抽出し、本文に適用されているタグの範囲のみが抽出されるように抽出アクションを設定します。

1. [アクション] タブで、[抽出] を選択します。
2. 範囲の最初のタグを指定します。
3. 範囲の最後のタグを指定します。

ロボットでのローカル ファイルの使用

ロボットを使用して、HTML、Excel、CSV、標準のテキスト ファイルなど、多くのタイプのファイルをロードすることができます。そのため、ロボットでは、さまざまなソースからデータを抽出できます。

- ロボットでネイティブにロードできるファイル タイプは、HTML、XML、Excel、および JSON です。
- 他のファイル タイプ (プレーン テキスト、CSV、PDF) もロードできますが、ロボットで処理される前に HTML に変換されます。

ファイル タイプをロードする手順は 2 つあります。ファイルがインターネット上にある場合、ページ読み込みアクションでファイルの URL を指定するか、クリック アクションでファイルへのリンクをクリックすることで、ファイルがロードされます。この操作では、ファイルがページ ビューに自動的にロードされます。ファイルがシステム上にある場合は、次の方法でファイルをロードし、ロボットを Management Console にアップロードしてスケジュールしたとき、または Kapplet に追加したときにもファイルを利用できるようにします。

PDF 以外のすべてのファイル タイプは同じ方法でロードされます。ファイルをロボットに追加するには、以下の手順に従います。

1. [変数の追加] フォームで、バイナリ タイプの変数をロボットに追加します。

   PDF や HTML などの他の変数タイプも使用できますが、これらの変数タイプはバイナリ タイプよりも柔軟性がなく、ユーザー入力が許可されない場合があります。

2. 名前を入力します。
3. [タイプと初期 / テスト値] の値では、リストからオプションを選択します。
4. 必要に応じて、[グローバル] と [パラメータとして使用] オプションを選択します。

   Management Console では、ロボットがスケジュールされるか、Kapplet で使用される場合にのみ、[パラメータとして使用] のオンとオフの違いが重要になります。入力変数はユーザーによる定義が可能であるため、ロボットを実行するたびにファイルは交換可能になります。一方、ロボットを実行するときにファイルが毎回同じである必要がある場合は、入力変数を使用する必要はありません。

5. [ロード] をクリックしてテスト ファイルをロードし、[**OK**] をクリックします。
   [パラメータとして使用] を選択しなかった場合、このテスト ファイルが最終ファイルになります。
   バイナリ タイプの属性を持つ変数はロボットに追加されます。この変数は入力変数として定義されるため、ユーザーは他のファイルを Kapplet やスケジュールに入力できます。
   テスト ファイルは属性にロードされます。

スニペットの作成と再利用

スニペットは次の 3 つの方法で作成できます。

1. ステップの選択範囲から：
   (単一のグループ ステップではなく、グループ化できるステップである必要があります)
   a. 1 つまたは複数のステップを選択し、[選択した範囲からスニペットを作成] をクリックします。
   b. 新しいスニペットの名前を入力します。
   c. 選択したステップが含まれる、その名前のスニペットを作成します。
2. グループ ステップをスニペット ステップに変換する：
   a. グループ ステップを選択し、[編集] メニューで [グループをスニペットに変換] をクリックします。
   b. 新規作成スニペットの名前を入力します。
   c. グループ ステップが含まれる、その名前のスニペットを作成します。
3. 新しいスニペットからスニペットを作成する：
   a. [ファイル] メニューから [新しいスニペット] を選択します。
   b. 新しいスニペットの名前を入力します。
      プロジェクトに空のスニペットが表示され、スニペット エディターが開きます。

      この エディター内でスニペットのコンテンツ (スニペット内のステップ) を編集することはできません。

   c. 必要に応じて、説明と参照される変数を編集します。

変数とスニペット

ロボットのすべてのステップと同じように、スニペットのステップでも変数を使用できます。スニペットのステップは、常にロボット内で編集されます。このコンテキストにおいて、ロボットで定義された変数をスニペットで使用することができます。別のロボットでスニペットを再利用するには、スニペットを使用する各ロボットで、スニペットのステップによって使用される変数を定義する必要があります。

スニペットは自身の変数を定義することができます。スニペット自身のエディターでスニペットを開き、スニペットで変数を定義します。スニペットにすでにステップが含まれる場合、スニペットが編集されたロボットに存在していた変数を使用すると、ステップは、赤のフラグでマークされます。

スニペットが変数を定義している場合、ロボットでスニペットを使用すると、ロボットの一連の変数にスニペット変数が自動的に追加されます。

ロボットには、ロボットが使用するスニペットで定義されている変数と同じ名前の変数定義を含めることはできません。同じ名前の変数定義が含まれている場合、変数タイプが一致する必要があります。

ロボットからスニペットを除去すると、そのスニペットがインポートした変数も除去されます。

スニペットのベスト プラクティス

スニペットのベストプラクティスを検討します。

- スニペットのステップに、スニペット内のステップの実行に使用するデフォルト以外のロボット構成を設定します。このようにすれば、スニペットを使うロボットごとに設定をする手間を省略することができます。
- スニペットをロボットに挿入するときは、スニペットで定義された変数の名前が、ロボットで定義された変数と競合しないように注意してください。Design Studio は、スニペットで定義された変数の名前が、ロボットで定義された変数と同じ名前になっている状況を処理できません。変数を別のコンテキストで使用する必要がある場合に、スニペットで変数を定義するのが適切な手法です。これにより、スニペットの再利用が容易になります。
- スニペットの記述では、スニペットに必要な名前付きタグやウィンドウのコンテキストを文書化します。
- スニペット内部にスニペットが含まれているときには注意してくだい。スニペットには、他のスニペットを参照するスニペット ステップを含めることができます。ただし、スニペットに循環参照 (スニペット自体が含まれている循環参照など) を含めることはできません。スニペットに循環参照が含まれている場合、Design Studio がエラーを報告します。

セッションの再利用

セッションとは、ウェブサイト ブラウジングの結果を指し、ブラウジングの過程で取得されたページ、そのページの URL、Cookie および認証が含まれます。しかし、必要な情報へ容易に到達できるセッションの取得には、ログインなどの多数のナビゲーション ステップが必要になります。

ロボットの実行が頻繁過ぎてその応答時間を大幅に短くする必要がある場合、ロボットの適切なセッションの作成には必要以上に多くの時間がかかります。しかし、セッションの取得が行われロボットおよびロボットの実行間で共有されると、時間が大幅に節約されることになります。

セッションの再利用には以下の 2 つのステップを使用します。

1. [セッションの保存] アクション：セッションを変数に保存します。
2. [セッションの復元] アクション：変数からセッションを復元します。

例

Web サイトにログインし、データを収集して返すロボットがあると仮定します。ただし、収集しようとするデータは、リンク先ページに広く分散しています ([次のページ] リンクを使用している場合など)。ロボットの最初の呼び出しでサイトにログインして最初のページのデータを返し、その後続の呼び出しではそれぞれ新たなデータのかたまり (次のページ) を返すようにする必要があるとします。また、ログインしているユーザーのセッションをロボットの呼び出し間で共有する必要があり、さらに、返したデータ量が記録されるようにしたいとします。ロボットは以下の例のようになります。

ロボットが呼び出されると、まず入力変数からセッションを復元しようとします。セッションが存在する場合はそのセッションが使用され、次のステップが次のページ リンクをクリックして新しいデータ ページを取得します。セッションがロボットに渡されない場合にはステップが失敗し、データが見つかる可能性のあるサイトの関連ページにログインし、ナビゲートする第 2 の選択肢が実行されます。

ロボットの実行が 2 つの代替分岐のうちの 1 つを経由すると、[セッションの保存] ステップに達します。これにより、次回ロボットを呼び出す場合に使用するセッションが保存されます。しかし、この呼び出しを行うには、セッションをロボットの呼び出し元に返す必要があります。これは、セッションを含む変数の値を返す通常の値返却ステップ、[リターン セッション] ステップによって処理されます (この変数は、[セッションの保存] ステップがセッションを保存した属性タイプ [セッション] の属性を持つタイプ)。最終的にロボットがデータの最後に到達すると (ページに次のページへのリンクが存在しない場合)、[次へをクリック] によってエラーが生成されます。[エラー処理] を [後続のステップ全てをスキップ] に設定しているため、これはロボットに無視されますが、[API 例外] にチェック マークを付けている場合は、呼び出し元で例外が発生します。たとえば、ロボットが Java から呼び出されている場合、このチェック マークを使用することでデータの最後に達したことを把握することができます。

セッションの保存後は、ロボットの残りのステップがテーブルをループして各行の値を返すといった方法によって、ページからデータを抽出します。

Design Studio では、ロボットの自然なフローによってロボットの実行が制御されることはありません。ロボットの実行は、ユーザーの操作によって制御されます。

1. セッションを保存するには、[セッションの保存] ステップの後に続くステップを選択します。
2. [セッションの復元] アクションを選択します。

既存のタイプの修正

あるタイプの変数を使ったロボットを記述した後にそのタイプを変更する必要がある場合には、注意が必要です。不適切な変更を行うと、ロボットは動作を停止することがあります。

既存のロボットの変数ですでに使用されているタイプに対して以下のいずれかの変更を行う場合は、変数を使用することができなくなる恐れがあるため、慎重に行ってください。ただし、ロボットは変数なしでロードすることもできます。

- タイプの名前を変更する。
- タイプを削除する。
- ロボット内で、タイプの 1 つ以上の変数によってそのデフォルトの属性とは異なる値が割り当てられている場合に、タイプの属性を除去または名前を変更します。
- ロボット内で、タイプの 1 つ以上の変数によって新しい属性タイプに対応しない値が割り当てられている場合に、属性の属性タイプを変更します。

ユーザーが上記の変更を行っている間にロボットが開いている場合は、ロボット エディターの上部に赤のステータス バーが現れ、問題を説明するテキストが表示されます。ステータス バーには、不正な変数を除去してロボットをリロードするためにクリック ボタンも表示されます。また、タイプに適切な変更を加えて問題を解決することもできます。問題を解決すると、ロボットに戻って作業を続けることができるようになります。

タイプに対する以下の変更は、そのタイプの変数を除去しなくてもロボットへ自動的に反映されますが (リロードが必要です)、ロボットの実行時にエラーが発生することがあります。このエラーは後から修正することができます。

- 属性名の変更。
- 属性の必須プロパティに対する FALSE から TRUE への変更。
- 必須プロパティが TRUE に設定されている新しい属性の追加。
- 変数に値が割り当てられている属性の削除または名前変更。
- 変数に値が割り当てられている属性の属性タイプの変更。

既存のロボットに影響を与えることなく、必要に応じて以下の変更を加えることができます。

- 属性の必須プロパティに対する TRUE から FALSE への変更。
- コメントの変更 (位置に無関係)。
- 必須プロパティが FALSE に設定されている新しい属性の追加。

アプリケーション ビューでの変数の使用

アプリケーション ビューには、ロードされた HTML ページや JSON ドキュメントなど、現在のロボット状態の一部が表示されます。特定の簡単なタイプ (XML、JSON、および Excel) の変数または属性をアプリケーション ビューのタブに表示することもできます。アプリケーション ビューに変数が表示されている場合は、アプリケーション ビューでロードされる他のドキュメントの場合と同じように変数を操作できます。たとえば、変数を抽出、テスト、ループ オーバーできるほか、多くの場合、変数を変更することもできます。

たとえば、いくつかの XML を入力として取得したり、出力値として返したりする Web サービスを呼び出します。次に、変数を表示しているウィンドウのコンテンツで動作するステップで変更した XML 変数を使用して入力 XML を作成します。入力 XML が適切な形式である場合、Web サービスのステップに入力としてフィードします。この Web サービスのステップに別の XML 変数の応答を保存してから、この応答をループ オーバーして、データを抽出することができます。

変数を開く

ウィンドウで変数 (または属性) を使用するには、最初に新しいウィンドウで変数を開く必要があります。変数を開くには、変数を開くステップを使用します。

変数を開く最も簡単な方法は、変数ビューで変数を右クリックし、メニュー オプションの **[ステップを挿入] > [変数を開く]** を選択することです。

1. 変数ビューで、変数を右クリックし、**[ステップを挿入] > [変数を開く]** を選択します。

   このステップを実行すると、新しいウィンドウに変数のコンテンツが表示されます。このように、変数を開くステップの動作は、ページ読み込みステップ アクションによく似ています。

   変数がすでに開いている場合、新しいウィンドウは表示されませんが、変数を含むウィンドウが新しいカレント ウィンドウになります。この点では、変数を開くステップの動作は、ページ読み込みステップ アクションとは異なり、カレント ウィンドウ設定ステップに似ています。

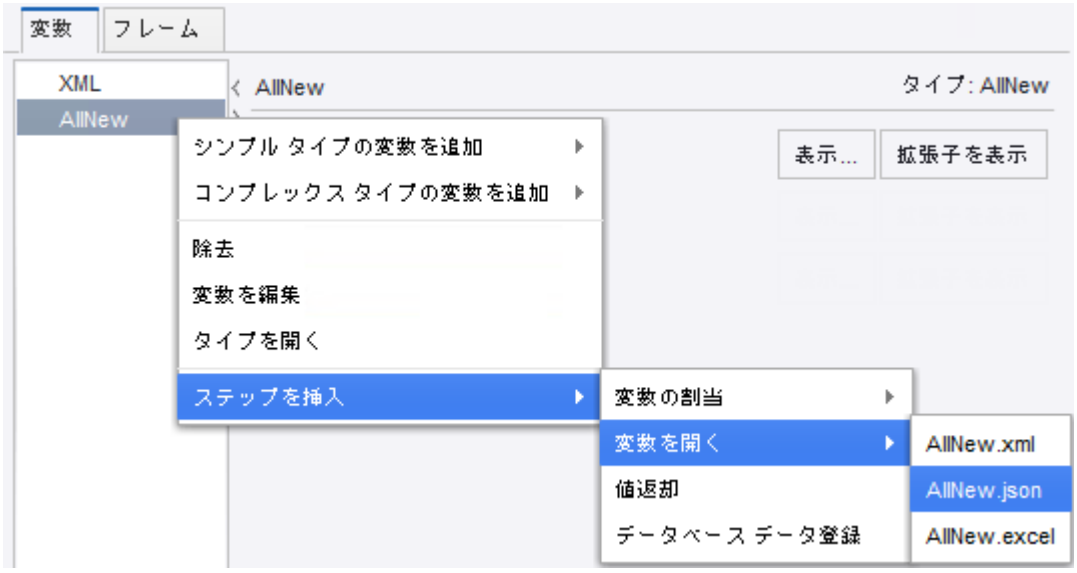
   このステップは変数を開くと呼ばれるものの、変数の属性もウィンドウで開かれることがあるタイプの場合には、変数の属性についても機能します。

   変数 (または属性) を開いたら、URL からロードしたドキュメント (XML ドキュメントなど) で使用するときのように変数を使用できます。

   ビューを右クリックし、変数で操作するステップを挿入できます。ステップの挿入は、変数または URL からロードされた (または、開かれた) かどうかに関係なく、カレント ウィンドウで機能します。実際に異なる点は、URL からロードされたドキュメントは変更できない場合があり、変更不可とみなされる点のみです。ドキュメントを変更するには、最初にドキュメントを変数に抽出してから変更する必要があります。

2. [変数] タブで、[XML] または [すべて新規作成] を右クリックし、設定のオプションを選択します。

   次の図は、コンプレックス タイプの JSON 変数の属性を開く方法を示しています。



   変数を開くステップは、ロボットの現在のステップの前に挿入されます。

3. 変数を右クリックし、変数で操作するステップを挿入します。

変数の変更

XML 変数と JSON 変数は、変更することができます。両変数タイプには、変数の変更に使用できるさまざまな専用ステップがあります。たとえば、属性設定により、既存の属性の値を設定したり、XML タグに新しい属性を追加したりできます。また、プロパティ名設定ステップにより、JSON オブジェクトのプロパティ名を変更できます。変数の割り当てのような変数を直接操作するステップを使用して、変数の割当もできます。この場合、ビューに変更が反映されます。

現在のウィンドウを通じて XML 変数を変更するステップは次のとおりです。

- タグ設定
- コンテンツ設定
- テキスト設定
- 名前付きタグ設定
- 属性設定
- コンテンツ挿入
- タグ除去
- コンテンツ除去
- 属性除去

現在のウィンドウを通じて JSON 変数を変更するステップは次のとおりです。

- JSON 設定
- プロパティ名設定
- JSON 挿入
- JSON 除去

XML または JSON タイプの変数が現在のウィンドウに表示されると、ウィンドウのコンテキスト メニュー (右クリック メニュー) で、ステップを挿入するためのメニュー オプションが利用できるようになります。指定したタイプに関連するステップ アクションのみが表示されます。ビューの現在の選択が関連していない場合、一部のステップ アクションは無効化される場合があります。たとえば、選択したタグに属性がない場合、属性除去は有効化されません。

JSON の使用

JSON (JavaScript Object Notation) は、次のような JavaScript リテラル表記法に類似している比較的簡単なデータ交換形式です。{ "x" : 5 , "y" : 7 }。

JSON はテキスト形式ですが、ロボットでは XML を表す場合と同じように、JSON 構造を表したり、表示したりします。JSON は、独自のページ タイプを使った独自のデータ形式として (HTML、XML、Excel の場合と同じように) 処理されます。前のバージョンの Design Studio の場合と同じように、XML に変換されることはありません。ページ タイプ判定ステップでは、現在のウィンドウのコンテンツが JSON であることを確認できます。

アプリケーション ビューで JSON は、URL から、または簡単なタイプの JSON の変数/属性からロードされます。アプリケーション ビューと変数ビューで開かれる JSON 変数を表示するための専用ビューに加えて、JSON のみで動作する専用ステップを利用できます。

JSON テキストの例を次に示します。

```
{ "answer" : 42,
   "people" : [ { "firstName" : "Arthur",
                  "lastName" : "Dent" },
                { "firstName" : "Ford",
                  "lastName" : "Prefect" } ] }
```

JSON の用語

JSON テキストは、オブジェクト、あるいは `{ "a" : 5 }` や `[1, 2, 3]` などの配列です。JSON 値は JSON テキストまたは JSON シンプル タイプのいずれかになります。ここでは、JSON シンプル タイプは JSON リテラル、数値、文字列のいずれかになります。JSON のリテラルは FALSE、NULL または TRUE です。FALSE と TRUE をブール値といいます。数値は、整数または浮動小数点数のいずれかです。数値の精度やサイズに制限はありませんが、別の式に変換された場合には、その式の制限が必ず考慮されます。たとえば、整数が整数変数に抽出された場合には、値が -2^{63} と 2^{63} -1 の間となる必要があります。この間の値とならない場合には、抽出ステップでエラーが生じます。JSON 文字列の始まりと終わりには二重引用符 (") を付ける必要があり、", \ を除くすべての Unicode 文字または制御文字を含めることができます (これらの文字は \"、\\ および \r などの \ を用いてエスケープすることができます)。JSON 形式は、https://www.ietf.org の RFC 4627 に記載されています。

JSON Syntax

JSON Text = JSON Object | JSON Array

JSON Object = {} | { Properties }

JSON Array = [] | [items]

Properties = Property,Properties

Property = String :JSON Value

Items = JSON Value,Items JSON Value = JSON Text | String | Number | false | null | true

String = " " | " Characters "

Characters = Character Characters

Character = あらゆる Unicode 文字を含む (例外として「、\ や制御文字 | \」| \\ | \/ | \b | \f | \n | \r | \t | \u などの 4 hex digitsNumber = C 数値または Java 数値によく似た数値)

JSON MIME タイプ

JSON テキストの MIME メディア タイプは以下のとおりです。

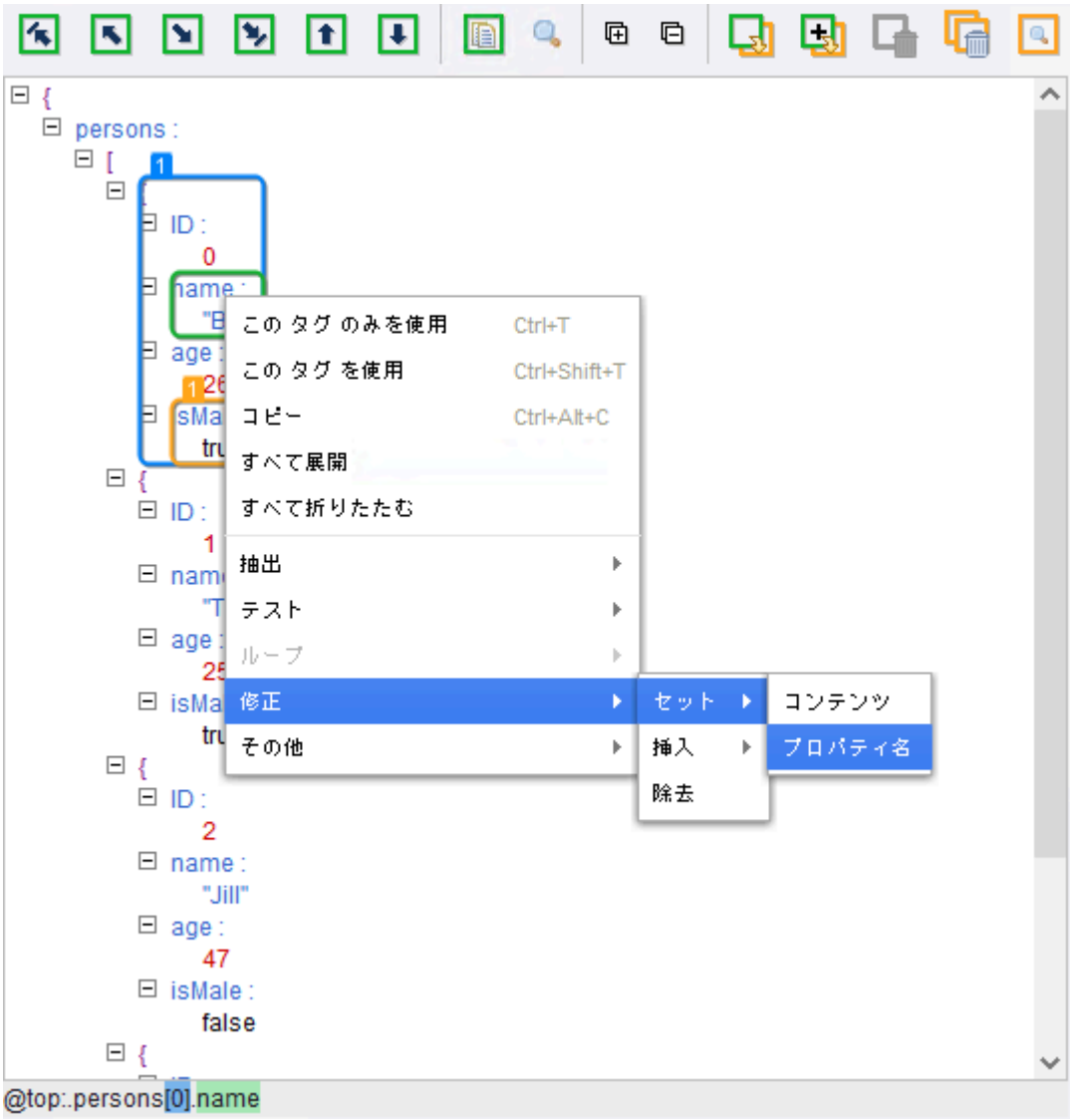
```
application/json
```

厳密に言えば、すべての JSON 値がこの MIME タイプに対して有効というわけではありません。JSON の受け入れや返却を行うサービスの開発者は、さらに自由な JSON 値の受け入れや返却を行う可能性もあります。Kofax RPA では、JSON に対して、こうしたより自由なアプローチを採るという選択をしました。そのために、JSON 変数には JSON 値が含まれ、JSON 表示にはその値が表示されます。

データが URL からロードされ、MIME タイプが application/json である場合、ロードされた JSON が JSON ページ ビューに表示されます。それ以外の場合は、データが JSON を表すように指定することができます。この指定を行うには、[ページ読み込み] ステップで、[ページ コンテンツ] タイプを JSON に設定します。またこの方法は、MIME タイプの使用が可能なソースからデータがロードされていない場合にも利用できます (ページ生成ステップなど)。

JSON とステップ アクション

多くのステップは JSON 上でのみ機能します。現在のウィンドウに表示されるデータは、JSON である必要があります (また、JSON が XML に変換されたレガシー形式の JSON ではない必要があります)。これらのステップは、ステップ ビューのアクション タブにあるステップ アクション セレクタの JSON というステップ カテゴリにあります。ただし、このステップ アクションを選択する一番簡単な方法は、現在のウィンドウに JSON が含まれている場合に、アプリケーション ビューでコンテキスト メニュー (右クリック メニュー) を使うことです。次に例示する図を参照してください。



JSON 値から 2 つのステップが抽出できます。

- JSON 抽出。このステップは、常に JSON 値を抽出します。たとえば、ビュー内の選択がプロパティである場合、これが抽出されるプロパティの値になります。マークアップとテキストには区別がなく

データ形式がより単純であるということを除き、これは HTML や XML から抽出される 抽出ステップと多くの点で類似しています。
- プロパティ名抽出。このステップは、プロパティの名前を抽出します。

以下の 2 つのステップは、JSON テキストをループ オーバーします。

- プロパティ繰り返し。このステップは、JSON オブジェクト の各プロパティをループ オーバーします。
- JSON アイテム繰り返し。このステップは、JSON 配列の各 JSON 値をループします。

両方のステップは、各イテレーションについて、該当する JSON 値の一部を名前付き JSON として設定します (名前付きタグに対しても同様)。この設定は、変数に対してイテレートする場合は、全体に適用されません。イテレーション対象のリストからアイテムを除去する場合のように、イテレーション中に変数の値を変更すると、イテレーションの失敗につながる値の変更が生じる可能性があるため、

以下の 4 つのステップで JSON を修正できます (JSON が変数にある場合のみ)。

- JSON 設定。選択した JSON 値の一部分を新しい JSON 値に置き換えます。
- プロパティ名設定。選択したプロパティでプロパティ名を新しい名前に設定します。
- JSON 挿入。JSON オブジェクトに新しいプロパティ、または JSON 配列に新しいアイテム (JSON 値) を挿入します。新しいプロパティまたはアイテムを挿入する位置には、最初または最後などの複数の選択肢があります。完全なリストについては、ステップに関する参考文書をお読みください。
- JSON 除去。JSON オブジェクトからプロパティ、または JSON 配列からアイテムなど、JSON 値の選択部分を除去します。

最後に、さらに以下の 2 つのステップが JSON で動作します。

- JSON 判定。このステップは、JSON 値の "type" がオブジェクト、配列、文字列などであるかどうかを判定します。
- 名前付き JSON 設定。このステップは、名前付きタグ設定 や 名前付き範囲設定 など、他のタイプのデータに対応するステップ アクションに似ています。このステップ アクションは、後続のステップで JSON 値の他の部分を見つける場合にリファレンスとして使用できるように、JSON 値の一部に対して名前付きリファレンスを定義します。ビュー内に青色のボックスとして表示されます。

JavaScript オブジェクトとしての JSON

「JavaScript を利用して変換」コンバータを含むステップ内のコンバータ スタックについて考えます。このコンバータは、コンバータで使用される JavaScript で使用するために名前付き INPUT という変数で、前のコンバータからの出力値にアクセスします。INPUT 変数の値は、常に文字列です。

次の表は、INPUT 変数の可能な変換値を示しています。

INPUT 値	**JavaScript (OUTPUT =)**	結果 **(OUTPUT** 値**)**
5	OUTPUT = INPUT	5
5	OUTPUT = INPUT + 3	53
5	OUTPUT = eval(INPUT)	5
5	OUTPUT = eval(INPUT) + 3	8
5	OUTPUT = eval(INPUT + 3)	53

INPUT 値	JavaScript (OUTPUT =)	結果 (OUTPUT 値)
5	OUTPUT = eval(INPUT + 「 + 3")	8
[1,2,3]	OUTPUT = INPUT[0]	[
[1,2,3]	OUTPUT = eval(INPUT)[0]	1
{ “a”: 5 }	OUTPUT = eval(INPUT).a	“Syntax Error”
{ “a”: 5 }	OUTPUT = eval("var x=" + INPUT + "; x;").a;	5

JSON を JavaScript に変換する場合は、以下のことに注意してください。

- INPUT は文字列値になる変数です。したがって、INPUT で実行する操作はすべて文字列操作です。たとえば、+ は文字列を連結するものです。これが、上記の例で INPUT + 3 が 53 になる理由です。
- 関数 "eval" は正しい JavaScript のみを入力として受け入れます。このため、上記最後の例では、構文上正しくない JavaScript 行 {"a":5} と異なり、正しい構文である var x = {"a":5} では正常に結果が返ります。

ブラウザ ウィンドウ アクション

ウィンドウでは、ロボットの HTML、XML、その他のコンテンツを保持します。1 つ以上のウィンドウが常に開いている場合、その中の 1 つのウィンドウが現在のウィドウになります。つまり、これは、ステップアクションが動作しているページを含むウィンドウです。Design Studio では、各ウィンドウがタブに表示され、現在のウィンドウが黄色い長方形でマークされます。

ウィンドウでは、複数のページを同時に処理することができます。ただし、ステップは 1 度に単一のページのみで動作します。そのため、現在のページではなく別のページで作業する場合は、常に現在のウィンドウを変更する必要があります。

ブラウザ ウィンドウで適切なステップを使用すると、次の操作を行うことができます。

- 新しいウィンドウを開くアクションを使用して新しいウィンドウを開く
- カレント ウィンドウ設定アクションを使用して現在のウィンドウを設定
- ウィンドウを閉じるアクションを使用して、ウィンドウを閉じる

Design Studio で [カレント ウィンドウ設定] ステップを挿入する簡単な方法は、ウィンドウのタブを右クリックし、[現在のウィンドウとして設定] を選択する方法です。

他のページをロードするページ (たとえば、[フレーム セット]-tag を含むページなど) をロードすると、それぞれのページは自動的に個別のウィンドウにロードされます。

各ウィンドウでは、1 つ以上の名前付きタグまたは範囲を利用できます。それぞれの名前付きタグまた範囲は、特定のウィンドウに属します。

ウィンドウの識別

一部のステップ (たとえば、上で言及したもの) は、特定のウィンドウで動作するように設定されます。このウィンドウは、以下の 3 つの方法で識別されます。

- ウィンドウのタブで表示されている名前またはウィンドウのタブの数
- 見つかったタブ
- ウィンドウ名に一致するパターン

名前は、ロボットの変更を考えた場合に 2 つの候補において、より安定しています。また、(最も控えめに言えば) ステップが異なるウィンドウを開く、異なるパスから到達できる場合にもそう言えます。よって、名前は、ウィンドウを識別する好ましい方法と言えます。変数を表示するウィンドウでは、照合は動作しません。これは、これらの名前が修正されるためです。

ただし、場合によっては、名前は、ロボットが実行されるたびに異なります。たとえば、一部の Web サイトはフレームに基づいていますが、これらのフレームには毎回異なる名前が付けられます (ただしフレーム セットの構造は維持されます)。ウィンドウ名がフレーム名から派生するため、ウィンドウ名はこのような場合に有用ではありません。そのため、そのウィンドウは、それぞれの番号によって参照される必要があります。これらの状況下では、対象のステップへと続くロボットからのすべてのパスが、同じウィンドウ構造およびウィンドウ番号になるようにすることが重要です。

また、ウィンドウの識別には、タグを使用することが可能です。検出されたタグは FRAME 要素、IFRAME 要素、OBJECT 要素または EMBED 要素である必要があります。Design Studio では、フレームのリストがフレーム ビューにツリーとして表示されます。カレント ウィンドウ設定 アクションで [「ウィンドウ #<番号> (ウィンドウまたはフレーム)」] オプションを使用して、見つかったタグで現在のウィンドウを設定します。

ウィンドウの識別には、以下の 2 つの代替の方法があります。

- ウィンドウ名にパターンを指定する
- ウィンドウの (テキストまたは HTML) コンテンツにパターンを指定する

両方の場合において、パターンは、単一のウィンドウの名前のみで照合できるように十分に正確である必要があります。

追加情報

このセクションでは、ベーシック エンジン ロボット の機能とパラメータに関する追加情報を示します。

プロトコル

プロトコルは RoboServer と通信するメカニズムを定義します。現在、Kofax RPA は 3 つのプロトコルに対応しており、それぞれ独自のメリットとデメリットが存在します。

プロトコル	説明
ソケット	TCP ソケットを使用して RoboServer と通信します。これはシンプルな低レベルのプロトコルであり、プラットフォームに応じて、XML または Java の二進表現を使用します。以下のプロパティの説明を参照してください。
ランダム配分	プロトコルのリストを与えると、クライアントがリクエストを行うたびに、現在利用可能としてマークされているかどうかに基づきプロトコルの中の 1 つが選択されます。リストで指定した 1 つ以上の RoboServer が利用できる場合は、これによって簡単なフェールオーバーが実行されます。ランダム配分プロトコルは明示的なロード バランシングを実行しているわけではありませんが、その目的のために使用できます。以下のプロパティの説明を参照してください。

プロパティ

プロトコルは次のプロパティを使用して設定します。

ソケット

ホスト名
RoboServer が配置されているホスト マシンの名前。

ポート番号
RoboServer がリスニングするポート番号。デフォルトのポート番号は 50000 です。

ランダム配分プロトコル

切断されたら再試行
このオプションを有効にすると、透過的なフェイルオーバーがサポートされます。リクエスト処理中に RoboServer との接続が失われると、このプロトコルによって、リクエストがリストの別の RoboServer に再送信されます。これが正しく動作するには、そのロボットが「べき等」であること、つまり、ロボットの起動が繰り返されても一台のロボットと同じ効果を持つことが必要です。通常、これは、アクセスするサイトで永久的な変更を行わないロボットに当てはまります。

プロトコル
リクエストを配分するプロトコルのリスト。ランダム配分ポリシーは、個々のリクエストの取り扱いに対してランダムにプロトコルを選択します。

Kofax RPA にインストールされているプラグインに応じて、他のプロトコルを使用することもできます。

ロボット ライブラリ

ロボット ライブラリは、Kofax RPA のロボットおよびタイプのコレクションです。ロボットを実行する要求を受け付けると、RoboServer は、そのロボットと関連するタイプをロボット ライブラリの中から検索します。

ロボット ライブラリは、RoboServer API を介してロボットを実行する場合のみ使用できます。

ロボット ライブラリ	説明
デフォルト ロボット ライブラリ	RoboServer は、現在のプロジェクト ロボット ライブラリを使用してロボットおよびタイプを検索します。 そのため、開発中のロボットが修正されるごとに、その変更がすぐに利用できて非常に便利です。
URL にあるロボット ライブラリ ファイル	RoboServer は、URL からライブラリを 1 度読み込み、その後しばらくの間ライブラリをキャッシュします。キャッシュ タイムアウトは、展開記述子にあるキャッシュ タイムアウト属性で制御できます。以下のプロパティの説明を参照してください。
要求に埋め込まれたロボット ライブラリ	このオプションで、RoboServer に送られるすべての要求の中にロボット ライブラリを埋め込みます。その後、RoboServer は、要求を満たすために必要なロボットおよびタイプをこのライブラリを使用して抽出します。

ロボット ライブラリ	説明
URL のロボット ライブラリ フォルダ	指定した URL に関連するロボットを検索するように、RoboServer に指示を与えます。 以下のプロパティの説明を参照してください。

プロパティ

以下のプロパティを使用してライブラリを設定します。

URL にあるロボット ライブラリ ファイル

URL
ロボット ライブラリ パッケージの場所。

キャッシュを許可
このオプションを有効にすると、RoboServer でロボット ライブラリをキャッシュできるようになります。RoboServer で URL のコンテンツのキャッシュが許可されていない場合、RoboServer は要求を受信するたびに URL のコンテンツを取得します。この機能は、ロボット開始までの時間を削減できるため、大きなロボット ライブラリをお使いの場合に非常に便利です。

キャッシュのタイムアウト
RoboServer がロボット ライブラリをキャッシュするために許可された秒数。「キャッシングを許可」プロパティが有効でない場合は、この設定も無効になります。

URL のロボット ライブラリ フォルダ

URL
関連ロボットを検索する URL。

JSON 変数値からのプラグイン シミュレーション

JSON 変数を使用して独自のプラグインをコンストラクトすることができます。以下は、JSON 構造の例です。

```
[
  {
    "name" : "Shockwave Flash",
    "description" : "Shockwave Flash 18.0 r0",
    "filename" : "NPSWF32_18_0_0_232.dll",
    "mimeTypes" : [
                    {
                      "type" : "application/x-shockwave-flash",
                      "description" : "Adobe Flash movie",
                     "suffixes" : "swf"
                    },
                    {
                      "type" : "application/futuresplash",
                      "description" : "FutureSplash movie",
                      "suffixes" : "spl"
                    }
                  ]
  },
  {
    "name" : "Silverlight Plug-In",
```

```
    "description" : "Silverlight Plug-In 5.1.40416.0",
    "filename" : "npctrl.dll",
    "mimeTypes" : [
                    {
                      "type" : "application/x-silverlight",
                      "description" : "npctrl",
                      "suffixes" : "scr"
                    },
                    {
                      "type" : "application/x-silverlight-2",
                      "description" : "",
                      "suffixes" : ""
                    }
                  ]
  }
]
```

次のコードを使用して、プラグイン シミュレーションで使用する JSON 変数を生成することができます。コードを HTML ファイルに保存してブラウザで開いてください。生成されるテキストは、Design Studio の [オプション] ウィンドウの [プラグイン] タブで、JSON 変数に直接コピーできます。

```
<!doctype html>
<html>
<body>
<div id="plugins"></div>
</body>
<script>
var plugins=[];
for(var n=0; n<navigator.plugins.length; n++) {
          plugins.push({});
          plugins[n].name=navigator.plugins[n].name;
          plugins[n].description=navigator.plugins[n].description;
          plugins[n].filename=navigator.plugins[n].filename;
          plugins[n].mimeTypes=[];
          for(var m=0; m<navigator.plugins[n].length; m++) {
                   plugins[n].mimeTypes.push({});
                   plugins[n].mimeTypes[m].type=navigator.plugins[n][m].type;
                    plugins[n].mimeTypes[m].description=navigator.plugins[n]
[m].description;
                    plugins[n].mimeTypes[m].suffixes=navigator.plugins[n][m].suffixes;
          }
}
var json = document.getElementById("plugins");
json.innerHTML= JSON.stringify(plugins);
</script>
</html>
```

名前付きタグ、範囲、JSON

名前付きタグ、範囲、JSON は、その他のタグ、範囲、一部の JSON テキストを検出するためにそれぞれ使用できるマーカーです。ステップは、ファインダーを使用して、処理する エレメントを発見します (ステップが処理するコンテンツの種類に応じて、HTML/XML タグ、Excel 範囲、名前付き JSON のいずれか)。ファインダーは、名前付きタグ、範囲、JSON の参照に応じて、前のステップによって検出される内容に基づくことができます。

すべての名前付きタグ/範囲/JSON はウィンドウに属します。各ウィンドウは、任意の数の名前付きタグ/範囲を持つことができますが、適切なタイプのみです。HTML/XML コンテンツのあるウィンドウは名前付きタグ、スプレッドシート コンテンツを持つウィンドウは名前付き範囲、JSON のあるウィンドウは名前付き JSON です。

名前付きタグ/範囲は多くのステップによって設定されます。たとえば、ループ ステップは、通常、ループの現在のイテレーションのマーカーとして名前付きタグ/範囲を使用します。明示的にタグ、範囲、JSON に名前を付けるには、名前付きタグ設定、名前付き JSON 設定、名前付き範囲設定アクションも使用することができます。これは以降のステップでファインダーを簡素化する場合に便利です。

Design Studio では、ウィンドウの名前付きタグや範囲は、青いボックスを使用して表示されます。現在のウィンドウで右クリックしてタグや範囲にアクションを実行する場合、可能な場合は必ずウィンドウの名前付きタグまたは範囲を使用して検索するために新しいステップのタグまたは範囲ファインダーが自動的に設定されます。

タグ ファインダー、範囲ファインダー、JSON ファインダー

ファインダーは HTML/XML ページ上のタグ、スプレッドシート ドキュメントのセルの範囲、または JSON 構造体の要素を検索するために使用されます。ファインダーは、ステップが操作の対象とするページの部分を特定するためにステップで使用されます。現在のステップのファインダーのリストはステップ ビューの [ファインダー] タブにあります。

検索の対象となるページの種類によってファインダーの形状は大きく異なるため、それぞれの種類に対するファインダーについて個別に説明しています。HTML/XML ページで使用されるファインダーの種類の詳細についてはタグ ファインダーを、スプレッドシートのコンテンツで使用されるファイルの種類の詳細については範囲ファインダーを、JSON ファインダーの詳細については JSON ファインダーを参照してください。

タグ ファインダー

タグ ファインダーは HTML/XML ページ上のタグを検索するために使用されます。タグ ファインダーはステップで使用され、ステップが適用されるタグを検索する方法を定義します。現在のステップのタグ ファインダーのリストはステップ ビューの [ファインダー] タブにあります。スプレッドシートのコンテンツを操作の対象とするステップは、タグ ファインダーではなく、範囲ファインダーを使用します。

タグ パスを理解する

タグ ファインダーを理解するうえでタグ パスの概念は重要です。タグ パスは、タグのページ内での位置についての短いテキスト表現です。以下のタグ パスについて考えます。

このタグ パスは、<html> タグ内の <body> タグ内にある タグ内の <a> タグを表します。

html.body.div.a

タグ パスは、同じページにある複数のタグと照合することができます。たとえば、上記のタグ パスは、このページ上の 3 番目のタグを除くすべての <a> タグと一致します。

```
<html>
  <body>
    <div>
      <a href="url...">Link 1</a>
      <a href="url...">Link 2</a>
    </div>
    <p>
      <a href="url...">Link 3</a>
    </p>
    <div>
      <a href="url...">Link 4</a>
      <a href="url...">Link 5</a>
      <a href="url...">Link 6</a>
    </div>
```

```
  </body>
</html>
```

インデックスを使って、そのレベルの同じタイプのタグの中から特定のタグを参照することができます。以下のタグ パスについて考えます。

html.body.div[1].a[0]

このタグ パスは、<html> タグ内の <body> タグにある 2 番目の タグ内の 1 番目の <a> タグを表します。したがって、上記のページでは、このタグ パスは "Link 4" の <a> タグとのみ照合されます。インデックスが 0 から始まることに注意してください。タグ パス上のタグにインデックスが指定されていない場合、上記の最初のタグのパスで見たように、パスはそのレベルの該当するタイプのあらゆるタグと照合されます。インデックスが負の場合、照合されるタグは、インデックス -1 に相当する最後の一致タグから逆方向にカウントされます。以下のタグ パスについて考えます。

html.body.div[-1].a[-2]

このタグ パスは、<html> タグ内の <body> タグにある最後の タグ内の最後から 2 番目の <a> タグを表します。したがって、上記のページでは、このタグ パスは "Link 5" の <a> タグとのみ照合されます。

アスタリスク ('*') を使って、任意のタイプの、任意の順番のタグを表すことができます。たとえば、次のタグ パスがあったとします。

html.*.p|div|td.a

このタグ パスは、<html> タグ内の任意の場所に配置された <p>、 、または <td> タグ内の <a> タグを表します。

タグ パスでは、ページ上のテキストはキーワードの "text" を使用して、他のタグと同様に表されます。技術的には、テキストはタグではありませんが、タグ パスでは同様に処理されて表示されます。例として、以下の HTML を考えます。

```
<html>
  <body>
    <a href="url...">Link 1</a>
    <a href="url...">Link 2</a>
  </body>
</html>
```

タグ パス "html.body.a[1].text" は、テキスト "Link 2" を表します。

タグ ファインダーのプロパティ

以下のプロパティを使用して [タグ ファインダー] を設定できます。

検索範囲

このプロパティでは、名前付きタグとの関連でタグを検索する場所を指定することができます。デフォルト値は「ページ内の任意の場所」であり、名前付きタグはタグの検索に使われません。

タグ パス

このプロパティでは、前のセクションで説明したタグ パスを指定できます。タグ パスは、値セレクターを使用してさまざまな方法で指定できます。

属性名
このプロパティでは、たとえば "align" などの特定の属性を持つタグを指定できます。

属性値
このプロパティでは、特定の値を持つ属性を持つタグを指定できます。[属性名] プロパティが設定されている場合、属性値はその属性名に関連付けられます。
これらの値は大文字と小文字が区別されます。

- 「テキストと等しくする」は、属性値が指定したテキストと完全に一致することを指定します。テキストは属性値の全体と一致する必要があることに注意してください。
- 「テキストを含む」は、属性値に指定されたテキストが含まれることを指定します。
- 「テキストで始まる」は、属性値が指定したテキストで始まることを指定します。
- 「以下のテキストで終了」は、属性値が指定したテキストで終わることを指定します。
- 「次のパターンに一致」は、属性値が指定したパターンと一致することを指定します。パターンは属性値全体と一致する必要があることに注意してください。
- 「テキストと等しくない」は、属性値が指定したテキストと等しくないことを指定します。
- 「テキストを含まない」は、属性値に指定されたテキストが含まれないことを指定します。
- 「テキストで開始しない」は、属性値が指定したテキストで始まらないことを指定します。
- 「テキストで終了しない」は、属性値が指定したテキストで終わらないことを指定します。
- 「パターンと一致しない」は、属性値が指定したパターンと一致しないとを指定します。

タグ パターン
このプロパティでは、(タグ内部のすべてのタグを含む) タグが一致する必要がある「.***Stock Quotes.***.*」などのパターンを指定できます。このプロパティはロボットのパフォーマンスに大きな影響を及ぼす可能性があるため、使用する際には注意が必要です。タグ パターンは、1 つの一致するタグを検索するためにページ全体に何度も適用されることがあるためです。これを回避するには、照合対象プロパティに「テキストのみ」を選択する方法があります。

照合対象
このプロパティでは、「タグ パターン」がテキストまたはタグの HTML 全体と照合することを指定できます。デフォルトでは、パフォーマンスの高速化のため、テキストのみと照合します。

タグ深度
このプロパティは、一致するタグ同士が互いの内部に含まれている場合に、どちらのタグを使用するかを決定します。デフォルト値はすべての一致するタグを受け入れる「範囲内の深度」です。「最も外側のタグ」を選択した場合は、最も外側のタグのみが受け入れられ、同様に、「最も内側のタグ」を選択した場合は、最も内側のタグのみが受け入れられます。

タグ番号
このプロパティは、複数のタグがタグ パスおよび他の条件と一致した場合に、どのタグを使用するかを決定します。使用するタグの数を、一致する最初のタグから順方向、または一致する最後のタグから逆方向のいずれかで数えて指定します。

例

たとえば、タグ パスを "table" に設定し、属性名プロパティを "align"、タグ属性プロパティを "align" に設定して、 属性値プロパティをテキストを "center" にした「固定テキスト」およびタグ パターン プロパティを ".*Business News.*" に設定すると、タグ ファインダーは、<table> タグで中央揃えされ、"Business News" というテキストが含まれる最初のものが検出されます。

範囲ファインダー

範囲ファインダーはスプレッドシート上のセルまたはセルの範囲を検索するために使用されます。範囲ファインダーはステップで使用され、ステップが適用されるセルを検索する方法を定義します。現在のステップの範囲ファインダーのリストはステップ ビューの [ファインダー] タブにあります。HTML ページまたは XML ページに働きかけるステップは、範囲ファインダーではなく、タグ ファインダーを使用します。

範囲ファインダーを設定するときは、さまざまな開始点を選択することができます。

指定範囲を検索

ほぼ通常の Excel 構文を使用して (範囲で) セルまたはセルの範囲を指定します。(Excel と異なり) シート名を指定する必要があることに注意してください。

範囲は、値セレクターを使用してさまざまな方法で指定できます。

名前付き範囲で検索

以前定義された名前付き範囲を開始点として指定します (範囲)。名前付き範囲は、名前付き範囲設定ステップまたはExcel 内ループステップなどで定義されている可能性があります。

範囲を開始点として選択した後、[使用する] プロパティで指定されている複数の方法で範囲を調整し、範囲を小さくしたり大きくしたりすることができます。詳細については、以下を参照してください。

最後に [結合セルの左上のセルを使用する] プロパティで、スプレッドシート上の結合セルを処理する方法を決定します。Excel では隣接するセルを視覚的に「結合」して、1 つの値を持つより大きいセルを形成できることを念頭に置いてください。Excel は、大きい「結合セル」が最も左上のサブセルと同じセル番地を持つと見なし、「結合セル」の値がそのセル番地に (のみ) 存在すると見なします。この処理は Kofax RPA によって正確に再現されますが、特にイテレーションの一部として自動抽出を行う場合、この処理は必ずしも便利ではありません。したがって、[結合セルの左上のセルを使用する] が有効になっていて、範囲が「結合セル」内の 1 つのサブセルを参照する場合は、目的のコンテンツを取得しやすくするために、「結合セル」の最上部、左端のサブセルを参照するように範囲が変更されます。

呼出範囲

範囲ファインダーを「指定された範囲を検索」に設定する場合は、セル (またはセルの範囲) を参照するテキストを書きます。そのような参照はスプレッドシート ビューの下部のセル範囲ビューにも表示されます (セル範囲ビューで参照を入力することもできます)。基本的な形式は Excel の数式からわかりますが、Kofax RPA は拡張機能を備えています。

以下の例はバリエーションを示しています：

Sheet1!B3

コア要素とそれらの要素を組み合わせる方法：シート名 ("Sheet1")、区切りの感嘆符、列名 ("B")、行番号 ("3")。この例では、指定されたシートの 1 つのセルを参照します。

B3
スプレッドシート ビューの下部のセル範囲ビューでセル範囲参照を入力するときに現在表示されているシートを参照する場合は、シート名を省略することができます。ただし、Enter キーを押すと直ちにシート名が参照に追加されます。ロボットの実行中は「現在表示されているシート」という概念がないため、範囲ファインダーでは毎回シート名を入力する必要があります。そのため、以下のすべての例にはシート名が含まれています。

Sheet3!B3:F14
1 つのシートからセルの範囲を参照する方法：シート名の後で、左上隅と右下隅をコロンで区切って指定します。(複数のシートにまたがる範囲を参照することはできません。)

Sales!C
指定されたシートの列 "C" のすべて。このようなオープンエンドな範囲は Excel では許可されませんが、さまざまなサイズの Excel 文書に対応する必要のあるロボットでは非常に便利です。ロボットは、特定のドキュメントを処理する場合、実際にデータが含まれている文書のエリアにロボット自身を自動的に制限します。オープンエンドな範囲は Excel 内ループステップの範囲ファインダーでよく使用されます。

Sales!C:H
指定されたシートの列 C から列 H のすべて。上で説明したオープンエンドな範囲です。

Suppliers!14
オペレーションな範囲は行にも適用できます。この例では、指定されたシートの行 14 のすべてを参照します。

Suppliers!14:29
指定されたシートの行 14 から行 29 までのすべてを参照する固定範囲。

'Total Sales'!B3
シート名に空白または特定の特殊文字が含まれている場合は、それを単一引用符で囲む必要があります。シート名に単一引用符が含まれている場合は、単一引用符を 2 つの単一引用符文字として入力する必要があります。規則は Excel の数式でセル参照にシート名が含まれているときとまったく同じです。

!B3
Kofax RPA では、Excel の「文書プロパティ」(たとえば、文書の作成者や作成日) を名前が空白の特殊なシートの形式で利用できます。したがって、この例は、文書プロパティの 1 つの値を参照します (プロパティの名前は隣接するセル "!A3" で利用できます)。これは Excel の拡張機能です。

JSON ファインダー

JSON テキスト内の必要なデータを検索するには、JSON ファインダーを使用します。現在のステップのファインダー リストは、ステップ ビューの [ファインダー] タブにあります。

JSON およびその関連用語の詳細については、JSON を使用するを参照してください。

JSON ファインダーのプロパティ

JSON ファインダーは、以下のプロパティを使用して設定できます。

検索範囲
このプロパティで、JSON 要素を検索する場所を指定できます。デフォルト値は「JSON 全域」で、これは検索で名前付き JSON を使用しないことを意味します。

この名前付き **JSON**
このプロパティは、**[検索範囲]** のリストで [名前付き **JSON** で] を選択したときに使用します。このプロパティで、選択した「名前付き JSON」内の検索を指定するか、または使用する「名前付き JSON」の名前を指定することができます。

パス
このプロパティで、JSON 要素へのパスを指定できます。タグ パスは、値セレクターを使用してさまざまな方法で指定できます。

JSON パス エクスプレッションは、常に、XPath エクスプレッションが XML ドキュメントとの組み合わせで使用されるのと同様に JSON 構造を参照します。JSON パス エクスプレッションは、JavaScript とよく似て、ドット表記を使用します (例：`personnel.person[0].name`)。`@top:` 要素は必須で、JSON の先頭から検索を行うようにファインダーに命じます。

例

JSON パス

以下は簡単な JSON 構造と、パス例および可能性のある結果を示す表です。

```
{
  "personnel" : {
                  "person" : [
                                {
                                  "ID" : 0,
                                  "name" : "Bob",
                                  "age" : 26,
                                  "isMale" : true
                                },
                                {
                                  "ID" : 1,
                                  "name" : "Ted",
                                  "age" : 25,
                                  "isMale" : true
                                },
                                {
                                  "ID" : 2,
                                  "name" : "Jill",
                                  "age" : 47,
                                  "exam" : "553213-3",
                                  "isMale" : true
                                },
                                {
                                  "ID" : 3,
                                  "name" : "Rick",
                                  "age" : 50,
                                  "exam" : "553225-3",
                                  "isMale" : true
                                }
                              ]
                }
}
```

XPath	**JSON** パス	結果
/personnel/person[2]/name	@top:.personnel.person[1].name	Ted
/personnel	@top:.personnel	"personnel" からすべての情報を抽出する

JSON の名前に次の文字が含まれている場合、角括弧で囲まれた新しいファインダーが使用されます。

```
.
"
..
[
]
\
```

たとえば、`{ "main.key": 42 }` のように、JSON の名前にドットが含まれている場合、[アプリケーション] ペインの要素を右クリックして「JSON 抽出」ステップを挿入すると、そのステップは次のファインダーを自動的に挿入します。

```
@top:["main.key"]
```

2 つの構文は同等なものとして使用でき、以前に作成したすべての JSON ファインダーは、新しいバージョンの RPA で機能します。

JSON 要素から情報セットを抽出する場合は、JSON から XML ページを作成し、テキスト エクスプレッションを使用して必要な情報を抽出することができます。たとえば、上の JSON から XML ページを作成する場合は、XML で `item[1]` を選択し、`".*<name>"+TheInput+"</name>.*"` のような エクスプレッションを実行すると、`1Ted25true` と同様の結果を得ることができます。

名前付き **JSON** の検索

次の例において、名前付き JSON は "a" を検出するための JSON ファインダーで使用できる JSON テキストの一部です。

次の JSON テキスト：

{ "a" : [{ "b" : [1,2,3] }], "c" :42 }

において、名前付き JSON に

"b" : [1,2,3]

をマークさせることができ、こうして、JSON ファインダーに以下のプロパティによる検索を行わせることができます。

検索範囲：名前付き JSON で

この名前付き JSON: 1

パス：[1]

これで、このファインダーはリスト内の番号 2 を検出します。

パターン

パターンを使用すると、テキストを表現することができます。たとえば、テキスト "32" は、2 桁を含むテキストとして記述することができます。しかし、その他のテキストも "12" や "00" などの 2 桁の数字を含みます。これらのテキストはパターンに一致すると言うことができます。(Design Studio のパターンは Perl5 の文法に従います。)

パターンは、通常の文字と特殊記号で構成されています。特殊記号にはそれぞれ特別な意味があります。たとえば、特殊記号 "."(ドット) は、任意の 1 文字を意味し、"a"、"b"、"1"、"2" など、単一のすべての文字に一致します。

特殊記号

パターン内では、次の特殊記号を使用することができます。

特殊記号	説明
.	任意の 1 文字 ("a"、"1"、"/"、"?"、"." など)。
\d	任意の 10 進数の数字 ("0"、"1" ... "9" など)。
\D	10 進数の数字以外の任意の文字 ("0"、"1" ... "9" を除外した "." と同じ)。
\s	任意の空白文字 (" "、タブ、リターンなど)。また、空白文字に関する注意を参照してください。
\S	任意の非空白文字 (" "、タブ、リターンなどを除外した "." と同じ)。
\w	任意の英数字 ("a" ... "z"、"A" ... "Z"、"0"..."9")。
\W	英数字以外の任意の文字 ("a" ... "z"、"A" ... "Z"、"0" ... "9" を除外した "." と同じ)。
\n	改行文字。
\r	キャリッジ リターン文字。
\t	タブ文字。
[abc]	a、b、c の中の任意の文字。
[^abc]	a、b、c 以外の任意の文字。
[a-z]	a から z までの範囲の任意の文字。
a|b	サブパターン a が一致するすべて、またはサブパターン b が一致するすべてに一致します。

"." または "\" のような特殊文字を通常の文字として機能させたい場合は、その前に "\" (バックスラッシュ) を付けて工スケープさせることができます。このため、単一の任意の文字ではなく、実際の "." 文字と一致させたい場合は、"\." と表記する必要があります。

以下の括弧を使って、パターンをサブパターンに整理できます。"(" および ")" 。パターン "abc" は "(a)(bc)" として整理できます。サブパターンは、パターン演算子を適用するときに便利です。

例: 簡単なサンプル パターン

以下はパターンとパターンの照合対象の例です。

パターン	一致内容
.an	"an" で終了する 3 文字のすべてのテキスト ("can" や "man" に一致しても "mcan" は一致しません)。
\d\d\s\d\d	2 桁の数字で始まり空白を 1 つ挟んで 2 桁の数字で終わる、5 桁のテキスト長 ("01 23" や "72 13" など。 "01 2s" は当てはまらない) に一致します。
m\.n\\o	テキスト "m.n\o" に一致します
(good)|(bye)	"good" および "bye" に一致しても "goodbye" には一致しません。

空白文字に関する注意

パターン内の空白文字は、常に同じ方法で処理されるとは限りません。これは旧バージョンとの互換性を確保するためです。場合によっては、空白文字が 1 つのスペースに圧縮されることもあります。たとえば、タグ テキストに複数の空白文字が連続して含まれている場合、それらは 1 つの空白文字に変換されます。

このような空白文字の圧縮は、次の場所で発生します。

- タグ パターンを使用するベーシック エンジン ロボットのファインダー。
- パターンを使用してタグの出現を待機するベーシック エンジン ロボットの HTML 待機基準。

その他の場合には、空白文字の圧縮は発生しません。

反復演算子

以下の演算子記号は、前の文字、記号、サブパターンを反復します。

特殊記号	説明
{m}	先行するサブパターンの m 回の繰り返しと正確に一致。
{m,n}	先行するサブパターンの m ~ n 回 (両端を含む) の繰り返しに一致。できる限り多くのサブパターンに一致します。
{m,n}?	先行するサブパターンの m ~ n 回 (両端を含む) の繰り返しに一致。できる限り少ないサブパターンに一致します。
{m,}	先行するサブパターンの m 回以上の繰り返しと正確に一致。できる限り多くのサブパターンに一致します。
{m,}?	先行するサブパターンの m 回以上の繰り返しと正確に一致。できる限り少ないサブパターンに一致します。
?	前のサブパターンか空テキスト。{0,1} の簡略記法です。
*	前のサブパターンの繰り返し回数、または空のテキストに一致。{0,} の簡略記法です。できる限り多くのサブパターンに一致します。
*?	前のサブパターンの繰り返し回数、または空のテキストに一致。{0,}? の簡略記法です。できる限り少ないサブパターンに一致します。
+	先行するサブパターンの 1 つ以上の繰り返しに一致。{1,} の簡略記法です。できる限り多くのサブパターンに一致します。
+?	先行するサブパターンの 1 つ以上の繰り返しに一致。{1,}? の簡略記法です。できる限り少ないサブパターンに一致します。

これらの演算子は、前の文字、記号、サブパターンを反復します。

例: 反復演算子の使用例

いくつか反復演算子を使用するサンプル パターンと一致する内容を挙げます。

パターン	一致内容
.*	任意のテキスト ("hello"、"1213"、および「」(空テキスト) など)
(abc)*	テキスト "abc" の任意の回数の繰り返しに一致します (「」、"abc"、"abcabc"、"abcabcabc" などと一致しますが "abca" には一致しません)。
(.*)(.*)	"abc" と一致します。最初のサブパターンが "abc" と一致し、2 番目のサブパターンが「」(空テキスト) と一致します。
(.*?)(.*)	"abc" と一致します。最初のサブパターンが「」(空テキスト) と一致し、2 番目のサブパターンが "abc" と一致します。

パターン	一致内容
(.+?)(.*)	"abc" と一致します。最初のサブパターンが "a" と一致し、2 番目のサブパターンが "bc" と一致します。
\w*\d	"abc1abc1" と一致します。\w* が "abc1abc" と一致し、\d が "1" と一致します。
\w*?\d	"abc1" と一致しますが "abc1abc1" には一致しません。 "\w*?" は "abc" とだけ一致し、残りが \d と一致できないためです。
(\d\d){1,2}	2 桁または 4 桁の数字のいずれかに一致します ("12" および "67" などと一致しますが "123" とは一致しません)。
(good)?bye	"goodbye" および "bye" 。

グループ化に関する詳細

"(" と ")" がサブパターンを新しいサブパターンにグループ化できるのを確認してきました。しかし、これは別の目的に使用できます。エクスプレッションの中でこれらのグループを使用できるのです。エクスプレッションで使用できないグループ化を行うには、"(?:" を使用することができます。

\n を使用すると、パターンの中の他のグループを参照できます。ここでの n はグループの番号です。

例: グループ化の例

いくつかグループ化を使用するサンプル パターンと一致する内容を挙げます。

パターン	一致内容
a(bc)	"abc=abc" と一致しますが、"abc=ab" とは一致しません。
a(?:bc)	"abc" と一致しますが、エクスプレッションで "bc" を参照することはできません。
(.*)=\1	エクスプレッションでは、グループを $1 として参照できます。

ライブラリ プロトコル

⚠ ライブラリ プロトコルは非推奨になりました。

代わりに Robot File System を使用するようにロボットを変更します。

ベーシック エンジン ロボットを変更して、ファイルを入力として取得するか、ファイルを Web サーバーに配置します。

Kofax RPA の非標準ライブラリ プロトコルを使用して、ベーシック エンジン ロボットが属している、ロボット ライブラリ内のファイルを参照することができます。

たとえば、ファイル MyPage.html がロボット ライブラリ フォルダ内のフォルダ MyFolder に存在する場合は、以下の URL を使用してそのファイルを参照することができます。

library:/MyFolder/MyPage.html

ロボット ライブラリがフォルダとして表現されているか、ロボット ライブラリ ファイルにパックされているかに関わらず、この方法は機能します。

値セレクター

値セレクター使用して、必要に応じて異なる方法で値を指定します。

値セレクターの右側のドロップダウン メニューを使用して、値を指定するいずれかの方法を選択します (すべての方法が必ず使用できるわけではありません)。

値
固定値を入力または選択します。

変数
変数の値を選択します。

エクスプレッション
エクスプレッションを入力します。

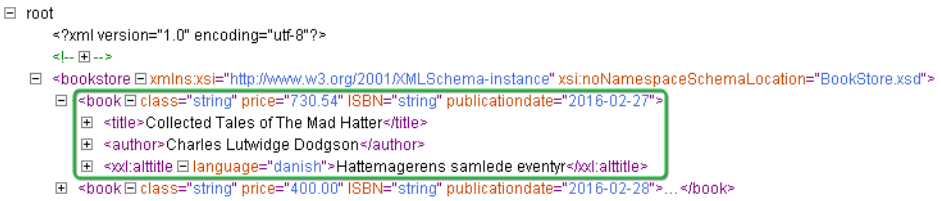
コンバータ
出力が値として使用される、データ コンバータのリストを選択します。(データ コンバータに対する入力テキストは空です。)

XML データ マッパー

XML データ マッパーを使用することで、XML ドキュメントで指定されたデータ レコードを適切な構造の変数に簡単にマッピングできます。

データ マッピングの作成

1. ソース ビューで、関連する XML 要素を選択します。



   選択に関連した問題についての詳細は、データの選択を参照してください。
2. 選択した XML 要素のコンテキスト メニューの [XML データ マッパーで抽出] を使用してマッピングを設定します。
3. [抽出] リストで、[**XML** データ マネージャ] を選択します。
   [XML データ マネージャ] ウィンドウが開きます。
   a. 利用可能なリストからソースをマッピングします。
   b. [ターゲット変数] エリアで、ターゲット変数およびタイプを選択します。
   c. ウィンドウの右下のエリアは、個別のエンティティ マッピングの詳細の設定用に確保されています。
      詳細情報は、設定を参照してください。
   ランタイムでマッピングを実行するステップ シーケンスが自動的に生成されます。
4. [**OK**] をクリックします。

ステップは自動的に作成されます。

詳細情報については、ステップの自動生成を参照してください。

データの選択

データの選択は簡単ですが、選択とマッパーが相互作用する仕組みを知ることが重要です。

[ソース] の列で、XML 要素を選択してエンティティを変数属性にマッピングします。マッパーを使用して、以下のエンティティを変数属性にマッピングします。

- 選択した要素自体の属性。[ソース] の列では、このような属性はプレーン テキストの名前として一覧表示されます。
- 自身にサブ要素を含まない、選択した要素のサブ要素のコンテンツ。[ソース] の列では、このような要素は、名前が < および > で囲まれる、XML のような形式で表示されます。
- 自身にサブ要素を含まない、選択した要素のサブ要素の属性。[ソース] の列では、このような属性は、要素名および属性名が < と > で閉じらている XML に似た形式で表示されます。

セクションにマッパーを有効化するには、最低でも 1 つのマッピング可能な要素が、選択した要素に関連付けられる必要があります。

設定

設定ステップでは、多くのユーザー操作が行われます。設定ステップは、マッピングが必要な各ソースのエンティティの適切なターゲット変数の属性を識別することを目的としています。利用できるすべてのソースをマッピングする必要はありません。

1. [ターゲット変数] セクションで、[新しい変数値を作成] を選択します。
2. 変数の名前を入力します。
3. [新しいタイプを作成] を選択します。
4. タイプの名前を入力します。
   ソース エンティティに一致する属性名が自動的に生成され、マッピングが提案されます。
   - 既存のタイプから新しい変数値を作成するには、[既存のタイプを使用] を選択します。タイプがソースによって提案された名前と一致する属性名を指定する場合、システムはソース エンティティのマッピングを試行します。
   - 既存の変数を使用して、新しいタイプに関連付けることができます。ソース エンティティに一致する属性名が自動的に生成され、マッピングが提案されます。
   - 既存の変数および既存のタイプを使用できます。システムは、タイプの属性に基づいて自動的にソースをマッピングしようとします。

   新しいマッピングが開始されると、システムは自動的に適切な変数およびタイプの選択肢の作成を試行します。提案は、XML の属性タグ名に基づいて作成されます。変数に同じ名前が付いている場合は、変数とタイプが提案されます。一致する変数が存在しても、タイプの名前がタグ名のように付けられている場合、システムは、このタイプを使用して新しい変数を作成するように提案します。
5. エンティティをマッピングするには、ソース リストで、エンティティを選択します。

6. ターゲット リストにおいて、属性を選択します。
   ソースおよびターゲットを接続するラインでは、接続が表示されます。
   単一のソースは、1 つ以上のターゲットにマッピングできます。
7. マッピング全体を正常なものにするためにソース エンティティが必要な場合は、ソース エンティティの隣のチェック ボックスを選択します。ソース エンティティに必須のマークを付けた場合は、抽出中にそのフィールドが存在しないと口ボットが失敗します。
   関連付けは、作成されたときのように簡単に解決できます。関連付けを消去するには、ターゲットの近くのラインをマウスでポイントし、表示される赤の十字をクリックします。
8. 利用できるソース エンティティに適したターゲット属性がない場合は、ソース要素を選択して、[の追加] をクリックします。
   [属性の追加] ウィンドウが表示されます。
9. 属性の名前とタイプを入力します。
   属性を除去するには、属性を選択して [の除去] をクリックします。

   この操作は、マッパーの現在の呼び出しに追加されている属性でのみ可能です。これはたとえば、対応するタイプ ファイルにおいてまだ保存されていない属性などです。

10. ターゲット属性を設定するには、属性を選択します。
    設定属性が表示されます。
    データが受理できるフォームになるように、データ コンバータをマッピングと関連付けることができます。
    ソース エンティティが XML 属性である場合は、[スペースの除去] を選択できます。
    - ターゲット属性名およびタイプは変更できます。
    - データが受理できるフォームになるようにデータ コンバータをマッピングと関連付けます。
    - 選択して、XML 属性のスペースのトリムをクリアします。

ステップの自動生成

1. データ マッピングの設定が完了したら、[**OK**] をクリックします。
   システムは、暗黙の抽出を実行して、口ボットにステップを挿入するのに必要な実際のステップを自動生成します。この手順は常に、マッパーで作成された各関連付けに対して、正確に 1 つの抽出またはタグ属性抽出ステップを生成します。

2. 選択した元の XML 属性を識別する、名前付きタグでデータ マッピングをアンカーします。
   このような名前付きタグが存在しない場合、タ [グ名称設定] ステップが生成され、実際のデータ マッピングの前に口ボットに挿入されます。

   場合によっては、このような名前付きタグ設定は不要です。これは、適した名前付きタグがすでに所定の位置に存在するためです。これは、マッピングがループの一部である場合によく発生します。

データ マッピングの編集

データ マッピングが設定されると、データ マッパーを使用することで、これを開いて編集できます。データ マッピングはグループ ステップと関連付けられているため、マッパーを再び開く機能もグループ ステップにつながっています。

1. データ マッピングを開くには、グループ ステップを右クリックして、[**XML** データ マッパーを開く] を選択します。

   これは、グループ ステップが現在のステップである場合 (緑) のみ動作します。緑でない場合は、マッパーをサポートするための XML 情報がありません。

   または、関連付けられているステップ ビューに現在存在するグループ ステップのマッパーを開くことができます。そして [**XML** データ マッパーを開く] をクリックします。
2. 開くと、マッパーはデータ マッピングの作成で説明されている通りに動作します。
3. 必要に応じて、データ マッピング ステップを手動で編集します。

   XML データ マッパーと互換性があるのは、特定の変更のみです。データ マッピングを開く操作は、多数の条件に影響を受けます。特に、グループ ステップが実際のデータ マッピングであることが重要です。
   - グループ ステップには、分岐、ループ、またはそれと同等のフォームのコントロール構造を含むことはできません。
   - グループ ステップには、最低でも 1 つの抽出またはタグ属性抽出ステップが含まれる必要があります。
   - 抽出はすべて、同じ名前付きタグを逆参照する必要があります。
   - 抽出にはすべて、同じ変数の対象属性がある必要があります。
   - この変数の指定の属性については、1 つのみの抽出があります。
   - アスタリスク '*' が発生する場合は、タグ ファインダーを使用できる抽出はありません。
   - 対象となるすべての属性は、基になるタイプに存在する必要があります。

URL ブロック

[URL ブロック パターンを設定] ウィンドウでは、URL ブロック パターンを指定して、このパターンをロボットのブロックする URL のパターンのリストに追加できます。パターンを指定してから [OK] をクリックすると、ロボットが再実行されます。ブロックされた URL のリストを表示するには、**[**ファイル**]** > **[**ロボット設定**]** に移動し、[基本] タブの [設定] をクリックして、[**URL** フィルタリング] を選択します。

パターン エディターでは特殊記号を使用することや、パターンを編集することができます。詳細については、パターンを参照してください。

Web 認証

Kofax RPA のベーシック エンジン ロボットは、ネットワーク上でさまざまな認証を使用できます。認証設定は、[デフォルト オプション] ウィンドウの [すべてのローディング] タブの [クレデンシャル] で指定されます。標準または OAuth のいずれかの資格情報を使用できます。詳細については、OAuth を参照してください。

[標準の資格情報] オプションを使用すると、Kofax RPA では、基本、ダイジェスト、NTLM、ネゴシエート プロトコルがサポートされます。Windows システムでは、Kofax RPA はセキュリティ サポート プロバイダー インターフェイス (SSPI) を使用して、セキュリティ サービスを呼び出しアプリケーションに提

供します。Unix では、Kofax RPA は、ネゴシエート プロトコルおよび開発した専用の NTLM サポート実装にジェネリック セキュリティ サービス API (GSS-API) ライブラリを使用します。

クロスプラットフォーム認証を使用するには、お使いの Linux インストールに、ジェネリック セキュリティ サービス API (GSS-API) ライブラリが含まれている必要があります。詳細については、『Kofax RPA インストール ガイド』の「Dependencies and Prerequisites」(依存関係と要件) のセクションを参照してください。

リモート ネットワーク リソースへのアクセスをユーザーに付与する必要がある場合、これらのリソースにアクセスするためにも、委任をセットアップする必要があります。認証および委任ルールの設定についての詳細は、msdn.microsoft.com および support.microsoft.com の Microsoft によるドキュメントを参照してください。

Kofax RPA は、ログイン処理中に認証のタイプを自動的に検出します。また、多くの場合、必要な形式で認証パラメータを提供します。NTLM プロトコルについては、SPN (サービス プリンシパル名) 文字列は常に以下のようになります。`HTTP/HostName:port`ネゴシエート プロトコルについては、SPN にはポート番号がある場合とない場合が考えられます。

場合によっては、ネゴシエート プロトコルの認証パラメータを明示的に指定する必要があります。この指定は、[ベーシック エンジン ロボットの設定のデフォルト オプション] ダイアログ ボックスの [すべてのローディング] タブの [認証方法] オプションで行うか、または spn.txt ファイルを使用して行うことができます。

[デフォルト オプション] ダイアログ ボックスでのネゴシエート プロトコル パラメータの指定

1. [ロボットの設定] ウィンドウの [基本] タブから [デフォルト オプション] ダイアログ ボックスを開きます。
2. [認証方法] リストで [ネゴシエート] を選択し、[追加] (+) をクリックします。[設定:][ネゴシエート認証の設定] ダイアログ ボックスが開きます。
   - [URL ホスト] フィールドに、接続する Web サイトのアドレスを `HTTP://<host name>:<port>/<page>` 形式で入力します。たとえば、`http://localhost:123/index.html` のように入力します。Kofax RPA は、入力したアドレスから `http://localhost:123` などのホスト名を抽出します。`<port>` はオプションの TCP ポート番号で、単一のホスト コンピュータ上の同じサービスの複数のインスタンスを区別するための非標準ポート番号を指定できます。ポート 80 および 433 は省略されています。
   - [サーバー] フィールドに、サーバー/サービスの名前を完全修飾ドメイン名 (FQDN) の形式で指定します。たとえば、`localhost:123` または `computer.global.companyname.com:1433` のように指定します。

   Kofax RPA が WebKit ブラウザに Web サイトをロードし、サーバーがプロトコルのネゴシエートを開始すると、WebKit は、ユーザーが [URL ホスト] フィールドで指定したパラメータでホスト名を照合しようとします。一致した場合、WebKit は次の形式で SPN 文字列を作成します。`HTTP/Server`。ここで、`Server` は、[サーバー] フィールドにオプションのポート パラメータが指定された FQDN となります。たとえば、[`HTTP/computer.global.companyname.com:1433`] など
3. 指定したアカウントの委任使用をオンに切り替える場合は、[代理認証可能] を選択します。
4. [保存] をクリックします。

プロキシの認証のネゴシエート

また、認証のネゴシエートはプロキシ サーバーで使用することもできます。spn.txt ファイル (このセクションで後述) または特定のロボットの構成設定を使用して、ネゴシエート プロトコルでプロキシ サー

バーを使用するように設定できます。spn.txt を使用すると、この設定がすべてのロボットで使用されます。

spn.txt を使用したネゴシエート プロトコル パラメータの設定

以下に説明する形式に従って spn.txt を作成し、ファイルをアプリケーション データフォルダの `Configuration` フォルダ (例: `C:\Users\user.name\AppData\Local\Kofax RPA\11.5.0.0\Configuration`) に配置します。

ファイルには、独立して指定できる 3 つのパラメータが含まれます。

spn.txt のファイル形式

パラメータ	説明	例
<host>:<port>::allow_port=[true|false]	SPN にポート番号を含めるかどうかを指定します。 `false` (デフォルト): ポート番号を含めない `true`: ポート番号を含める	`localhost::allow_port=true`
<host>:<port>::delegate=[true|false]	指定したアカウントの委任使用をオンに切り替えます。 `false` (デフォルト): 委任を使用しない `true`: 委任を使用する	`localhost::delegate=true`
<host>:<port>=<FQDN>	ホスト名をサーバーの完全修飾ドメイン名 (FQDN) の形式で入力します。このパラメータは、`allow_port` パラメータを上書きします。また、Kofax RPA は、ここで指定された通りの文字列を使用します。	`localhost=COMPUTERNAME.companyname.com`

ネゴシエート プロトコルが使用されている環境内に複数の Web サイトがあり、これらのサイトが異なるポート番号および異なるアプリケーション プール ID を使用して同じホスト名で実行されている場合、たとえば `allow_port` を `true` に設定し、SPN に非標準ポートを指定できます。

```
  localhost:8888::allow_port=true
```

また、`localhost:8888=server:555` のような、SPN サーバーを割り当てるためにマスクの一部としてポートを使用することもできます。

ロギング

Web 認証の際にエラーが発生した場合は、以下のように `log4j2.properties` ファイルの Webkit ロギングをオンに切り替えることができます。

```
 logger.webkit.name = webkit
 logger.webkit.level = TRACE
```

ログ ファイルには、使用される認証プロパティおよび SPN 文字列についての情報が含まれている必要があります。ログ ファイルの以下の行を探します。

```
Setting SPN to : 'Service Principal Name (SPN)'
Delegate : [ON|OFF]
```

```
Non-standard port : [ON|OFF]
Setting NTLM SPN to : 'SPN string'
```

詳細については、ロギングを参照してください。

事前定義済み JavaScript ポリフィル

次に、事前定義された JavaScript ポリフィルのリストを示します。詳細については、「ベーシック エンジン ロボットの設定のデフォルト オプション」トピックの「[JavaScript の実行] タブ」にある「JavaScript ポリフィル」のセクションを参照してください。

オブジェクトまたは **API**	説明	注意
Array.prototype.values Array.prototype.keys Array.prototype.includes Array.prototype.findIndex Array.prototype.find Array.prototype.fill Array.prototype.entries Array.prototype.copyWithin Array.prototype.contains Array.prototype.@@iterator Array.of Array.from	JavaScript Array は、配列の構築に使用されるグローバル オブジェクトです。配列は、高レベルのリスト構造オブジェクトです。	
Console	Console オブジェクトはブラウザのデバッグ コンソールへのアクセスを提供します。	ブラウザで使用できる Console オブジェクトの現在の機能を置き換えるものです。
DOMTokenList.prototype.@@iterator	DOMTokenList インターフェイスは、スペースで区切られたトークンのセットで表現されます。JavaScript Array オブジェクトと同様に 0 からインデックス付けされます。常に大文字と小文字が区別されます。	
Element.prototype.replaceWith Element.prototype.prepend Element.prototype.matches Element.prototype.closest Element.prototype.before Element.prototype.append Element.prototype.after	Element は Web ページの一部です。	
IntersectionObserverEntry IntersectionObserver	上位層要素または最上位ドキュメントのビューポイントを含むターゲット要素の交差部の変更を非同期で監視する方法を提供する Intersection Observer API のインターフェイス。上位層要素またはビューポイントはルートとして参照されます。	

オブジェクトまたは **API**	説明	注意
Intl	言語依存文字列の比較、数値の書式、日付や時刻の書式を提供する ECMAScript 国際化 API の名前空間。INTL オブジェクトは、複数のコンストラクタへのアクセス、および国際化コンストラクタに共通する機能とその他の言語依存機能を提供します。	オブジェクトの全セットおよび次の言語をサポートします。 `da, de , en, ja, ru`
Map	キーと値のペアを保有し、キーの元の挿入順序を記憶するオブジェクト。	オブジェクトの全セットをサポートします。
Math.trunc Math.tanh Math.sinh Math.sign Math.log2 Math.log1p Math.log10 Math.imul Math.hypot Math.fround Math.cosh Math.clz32 Math.cbrt Math.atanh Math.asinh Math.acosh	Math は、数学の定数および関数のプロパティおよびメソッドの組み込みオブジェクトです。関数オブジェクトではありません。	
NodeList.prototype.forEach NodeList.prototype.@@iterator	NodeList オブジェクトは、通常、Node.childNodes などのプロパティや document.querySelectorAll() などのメソッドから返されたノードのコレクションです。	
Number.parseInt Number.parseFloat Number.isSafeInteger Number.isInteger Number.MIN_SAFE_INTEGER Number.MAX_SAFE_INTEGER Number.Epsilon	Number は、倍精度 64 ビット浮動小数点形式 (IEEE 754) の数値データ型です。その他のプログラミング言語には、次のようなさまざまな数値タイプを含めることができます。Integer、Float、Double、または Bignum。	
Object.values Object.setPrototypeOf Object.keys Object.entries Object.assign	オブジェクト ラッパーを作成するコンストラクタ。	

オブジェクトまたは **API**	説明	注意
Performance	Performance インターフェイスは、現在のページのパフォーマンス関連情報へのアクセスを提供します。High Resolution Time API の一部ですが、Performance Timeline API、Navigation Timing API、User Timing API、および Resource Timing API によって強化されています。 このタイプのオブジェクトは、読み取り専用の属性である `window.performance` を呼び出すことで取得できます。	`PerformanceEntry` オブジェクトをサポートします。 `PerformanceEntry` オブジェクトは、*performance timeline* の一部である単一のパフォーマンス メトリックをカプセル化します。
Promise Promise.prototype.finally	Promise は、非同期処理のイベントの完了 (または失敗) および結果の値を表すオブジェクトです。	
RegExp.prototype.flags	現在の正規表現オブジェクトのフラグから成る文字列を返すプロパティ。	
セット	プリミティブ値かオブジェクト参照かに関わらず、任意のタイプの一意の値を保存できるようにするオブジェクト。	オブジェクトの全セットをサポートします。
String.prototype.contains String.prototype.codePointAt String.prototype.endsWith String.prototype.@@iterator String.prototype.includes String.prototype.padEnd String.prototype.padStart String.prototype.repeat String.prototype.startsWith String.prototype.trim	すべての String インスタンスは String.prototype から継承します。String プロトタイプ オブジェクトへの変更は、すべての String インスタンスに伝播されます。	
Symbol Symbol.hasInstance Symbol.isConcatSpreadable Symbol.iterator Symbol.match Symbol.search Symbol.species Symbol.split Symbol.toPrimitive Symbol.toStringTag Symbol.unscopables	Symbol は関数 Symbol を呼び出して作成されるプリミティブ値で、匿名の一意の値を動的に生成し、オブジェクト プロパティとして使用される場合があります。	
Url	オブジェクト URL の作成に使用する静的メソッドを提供するオブジェクトを表すインターフェイス。	オブジェクトの全セットをサポートします。
WeakSet	弱保持オブジェクトをコレクションに保存するために使用されるオブジェクト。	

オブジェクトまたは **API**	説明	注意
WeakMap	キーが弱参照されるキー/値のペアのコレクション。キーはオブジェクトである必要があり、値は任意です。	

Excel の補足ドキュメント

このセクションでは、Kofax RPA ロボットで Excel データとワークブックを操作するための補足ドキュメントを提供します。

このセクションのトピックでは、Kofax RPA コンポーネント内のユーザー インターフェイス オプションに固有ではない Excel の関数、エラー、設定、および機能について説明します。

- ロボットの構築 およびユーザー インターフェイス関連のトピックについては、『Kofax RPA のヘルプ』の「RPA のコンポーネント」セクションを参照してください。
- トレーニングと詳細な手順については、『Kofax RPA ロボット構築の開始ガイド』を参照してください。
- Microsoft Excel の使用に関する一般的な情報については、使用している Excel バージョンに応じて Microsoft が提供するドキュメントを参照してください。

Excel の互換性と制限事項

互換性

Kofax RPA は、ロボットとプラットフォームのタイプに応じて Microsoft Excel ワークブックのサポートを提供します。

- ベーシック エンジン ロボットの場合、組み込み Excel はサードパーティ ライブラリである Apache POI を使用して Excel 機能を実装します。これは、Windows および Linux のプラットフォーム上の RoboServer または Design Studio ホスト上で実行される Java のクロスプラットフォーム実装です。
- ロボットの場合、組み込み Excel ドライバーによって Excel 自体が自動化されます。ステップを実行するコンピューターには Microsoft Excel がインストールされている必要があります。
  - Windows インストールの場合のみ、ホストは RoboServer または Design Studio を実行しているホストになります。
  - Windows および Linux インストールの場合、Desktop Automation サービスのホストは RoboServer と Design Studio をサポートします。
- リモートの Desktop Automation サービスのホスト上で、ロボットは Microsoft Excel を自動化できます。Microsoft Excel はリモート ホスト上で実行されるため、このオプションは Windows または Linux 上の RoboServer または Design Studio で使用できます。Desktop Automation サービスがインストールされているコンピューターには、Microsoft Excel がインストールされている必要があります。

制限事項

Kofax RPA での Microsoft Excel のサポートには、次の既知の制限があります。

- ベーシック エンジン ロボットで組み込み Excel を使用する場合は、Microsoft Excel および Microsoft 365 ドライバーに制限があることに注意してください。これらの制限は、サードパーティの Apache POI ライブラリに関連するものです。これはサードパーティの実装であるため、機能が Microsoft Excel および Microsoft 365 アプリケーションと 100% の互換性を持つことは保証されません。
- ロボットで組み込み Excel ドライバーを使用する場合、2020 年に Microsoft Excel および Microsoft 365 に導入された動的な配列機能のサポートは、Microsoft Excel の Dynamic Array 以前の互換モードに制限されます。ロボットによってセルに挿入されたエクスプレッションは、暗黙的な交差評価モードで評価されるように Microsoft Excel によって設定されているため、結果は隣接するセルにはスピルされません。
- Microsoft Excel の異なるバージョンを使用した場合、あるバージョンから別のバージョンへの 100% の互換性は保証されません。

関連項目: Kofax RPA の制限。

既知の問題

このセクションでは、Kofax RPA で Microsoft Excel を使用した際に発生する既知の問題を解決するためのトピックを提供します。

ロボットで Excel を使用する際のエラー

このトピックでは、組み込み Excel ドライバーを使用してロボットを実行する際のエラーを解決および防止する方法について説明します。

SORT 関数によって **1** つのセルだけが並べ替えられることがある
Excel で SORT 関数を入力すると、通常はセルが並べ替えられ、結果が順番に並べ替えられて表示されます。組み込み Excel ドライバーでは、SORT 関数の結果は 1 つのセル (並べ替え順序の最初のセル) にのみ適用されます。

Desktop Automation サービスを使用して、デスクトップ上の Microsoft Excel を自動化します。

式に **@** 記号が使用される場合がある
ロボットによってワークブックに式が挿入された場合、新しいバージョンの Microsoft Excel では、挿入された式に @ 記号が追加されて表示されることがあります。これは Microsoft Excel API の制限です。

例: `=SORT(A2:A9)` が、Microsoft Excel では `=@SORT(A2:A9)` . と表示される場合があります。

ワークブックをロボットで再度開くと、@ 記号がない状態で式が正しく表示されます。

@ は、新しい Microsoft Excel バージョンで導入された暗黙的な交差演算子です。このバージョンの Microsoft Excel では、暗黙的な交差評価モードで計算される関数の互換性のヒントとして式内の演算子が表示されます。@ は、関数の結果が、隣接するセルにオーバーフローしないことを示します。

[フォーマットを設定]> [カスタム フォーマット] フィールドをローカライズする必要がある
Microsoft Excel の日付などの一部のフィールドのフォーマット パターンは、システムの Windows 地域設定に依存します。これらの設定は、要素の順序付けの方法および要素に使用する必要があるコードに影響します。たとえば、日付フォーマットと年フィールドのコードはロケールに応じて異なります。

[フォーマットを設定] ステップは、Kofax RPA インストールの言語に基づいて、事前定義された設定をローカライズします。この言語が Windows の地域設定と一致しない場合は、[カスタム フォーマット] オプションを使用して、インストールに適切な形式を指定します。

[カスタム フォーマット] フィールドはそのまま Microsoft Excel に渡されます。開発者は、ロボットを実行しているシステムに対応する Windows の地域設定に一致したフォーマットを使用する必要があります。

Microsoft Excel からのインストール エラー

このトピックでは、RoboServer で Windows サービスとしてロボットを実行する際のエラーを解決および防止する方法について説明します。この場合、組み込みの Excel ドライバーを Windows で正しく動作させるために追加の設定が必要になることがあります。

デバイスの問題。コマンド実行エラー: `Excel` インスタンスの作成エラー: 未知のエラー `0x800A03EC`

エラーを解決するには、次の手順を実行します。

1. 次の 2 つの空のフォルダを作成します (存在しない場合)。
   - `C:\Windows\system32\config\systemprofile\Desktop`
   - `C:\Windows\SysWow64\config\systemprofile\Desktop`
2. Windows サービスのユーザーにこれらのフォルダへのアクセス権があることを確認してください。これにより、サービスから Excel を起動できるようになります。

コマンド実行エラー: `Excel` インスタンスの作成エラー: サーバーの実行に失敗しました: コマンド実行エラーと `Excel` インスタンス作成エラー: アクセス拒否

これらのエラーは、Excel COM の起動が不十分な場合、および設定されたアクティブ化権限が不正な場合に発生することがあります。

両方のエラーを解決するには、次の手順を実行します。

1. [実行] ダイアログまたは Windows の [スタート] メニューで、`dcomcnfg.exe` アプリケーションを実行します。
2. ウィンドウが開いたら、**[コンポーネント サービス] > [コンピューター] > [マイ コンピューター] > [DCOM の構成] > [Microsoft Excel Application]** に移動します。

   > [**Microsoft Excel Application**] エントリがない場合は、別のアーキテクチャの `dcomcnfg.exe` アプリケーションを起動する必要があります。
   > そのためには、コマンドライン ウィンドウを開き、`C:\Windows\SysWOW64` に移動して、次のコマンドを実行します: `mmc comexp.msc /32`

3. [**Microsoft Excel Application**] エントリを右クリックし、[プロパティ] を選択します。
4. **[セキュリティ]** タブで、[起動とアクティブ化のアクセス許可] の [カスタマイズ] を選択し、[編集] をクリックします。
5. ユーザーまたはユーザー グループがリストにない場合は、[追加] をクリックして追加します。
6. [ローカルからの起動] と [ローカルからのアクティブ化] のチェック ボックスが選択されていることを確認します。
7. すべての変更を保存し、Windows サービスを再起動します。

ベーシック エンジン ロボットで Excel を使用する際のエラー

このトピックでは、Apache POI 実装である組み込み Excel を使用してベーシック エンジン ロボットを実行する際のエラーを解決および防止する方法について説明します。

[フォーマットを設定] するとセルの背景色が白になる

Design Studio レコーダー ビューの [フォーマットを設定] オプションを使用すると、Excel セルの背景色が白になります。ファイルを保存した後も、セルの背景色は白のままになります。セルの背景色を黒に設定すると、同様のエラーが発生します。

(ベーシック エンジン ロボットではなく) 組み込み Excel を使用してフォーマットを設定します。

シート名またはシート インデックスが無効となる

数値名 (1、2、3) が割り当てられたシートを含む Excel ファイルを編集し、`=SUM` などのアクションにこのシートを使用する式がファイル内にある場合、エラーが発生します。

シートの名前を変更して、数値以外の名前 (一、二、三 など) を使用します。また、最初の文字が数字ではない場合も正常に機能します。

ロボットでの Excel の使用

Kofax RPA ロボット ワークフローには、Excel ブックで操作を実行するための組み込み Excel ドライバーが含まれています。この操作を実行するコンピューターには Microsoft Excel がインストールされている必要があります。

組み込み Excel ドライバーは、次のオートメーション デバイスで使用できます。

- リモートの Desktop Automation サービス デバイスでは、リモート デバイスにインストールされている Microsoft Excel を使用します。
- ローカル オートメーション デバイスでは、Design Studio および RoboServer ホスト上の Microsoft Excel を使用します。

  ローカル オプションが選択されている場合、このオプションがサポートされるようにするには、Design Studio と RoboServer ホストに Microsoft Excel がインストールされている必要があります。

ワークブックを作成して開く

- 新しい Excel ワークブックを作成するには、Excel ステップを挿入し、[アクション] リストで [ファイルを作成] を選択します。
- Excel ワークブックを開くには、次の手順を実行します。
  1. スタンドアロン実行モードで実行している場合は、**[実行の準備]** をクリックします。
  2. Excel ステップを挿入します。
  3. **[アクション]** リストで **[ファイルを開く]** を選択します。
  4. ソース (直接アクセス、RFS、または変数) を選択します。
  5. 直接アクセスまたは RFS ソースの場合は、ワークブックへのフル パスを入力します。

     例: `C:/documents/myworkbook.xlsx`

6. 変数ソースの場合は、データを含むバイナリ タイプの変数を選択します。

ワークブック内を移動する

ワークシートを含むワークブックを開くときは、各ワークシートのボタンをクリックするか、[コンポーネント アクション] メニューと [アプリケーション アクション] メニューを使用して、ワークシート間を移動します。

ツールバー ボタン

組み込み Excel ドライバーのツールバー ボタンを使用して、次の操作を実行します。

ボタン	アクション	説明
	保存	直接アクセスおよび RFS ソースの場合は、ワークブックに変更を保存します。 • 以前に保存を実行している場合は、最後に保存した場所に保存されます。 • ファイルから読み取った場合は、同じ場所に保存されます。 • 新規の場合は、[名前を付けて保存] ダイアログボックスが開きます。 ワークブックを保存するには、[保存] ボタンまたは [名前を付けて保存] ボタンを使用します。
	名前を付けて保存	ワークブックをファイル システムまたは RFS の場所に保存します。 [名前を付けて保存] した後に [保存] アクションを実行すると、同じ場所に保存されます。
	マークしてコピー	現在の選択をコピー/ペースト アクションするためにマークします。選択可能なオプションについては、「セルの選択」を参照してください。
	貼り付け	直前にマークした選択を選択したセルにコピーします。「クリップボードの操作」を参照してください。
	テキスト色の設定:	選択したセルのテキストにアクティブな色を適用します。
	背景色の設定:	選択したセルにアクティブな色を適用します。
	カラー ピッカー:	色の変更操作に使用するアクティブな色を設定します。
	カスタム カラーの入力:	[カラー ピッカー] のカスタム カラーを 16 進数形式で指定します。 たとえば、純粋な白は `ffffff` で、純粋な黒は `000000` です。
		このテキスト フィールドには、現在選択されているセルのアドレスおよびその値が表示されます。

セルの選択

次のように、ワークシート内の 1 つまたは複数のセルを選択します。

- ワークシート内の 1 つのセルを選択するには、デスクトップ版の Microsoft Excel と同様にクリックします。
- 行を選択するには、行見出しをクリックします。
- 列を選択するには、列見出しをクリックします。
- ワークシート全体を選択するには、[すべて選択]  をクリックします。
- ワークシート内のセル範囲を選択するには、キープレス で Shift キーを押しながら矢印キーを使用します。

  選択は左から右へ、上から下へと行われます。選択範囲を下から上に移動したり、右から左に移動してセルを選択することはできません。

  たとえば、幅 5 セル × 高さ 3 セルの範囲を選択するには、次の手順を実行します。

  1. 範囲の左上隅にあるセルをクリックします。
  2. キー プレス ステップを挿入します。このステップでは、ファインダー内のアプリケーション名として `excel` を指定し、[標準キー] で [右矢印] を選択し、キー修飾子として [**Shift**] を選択して、[カウント] フィールドに 4 を入力します。
  3. ステップを実行します。
  4. キー プレス ステップを挿入します。このステップでは、ファインダー内のアプリケーション名として `excel` を指定し、[標準キー] で [下矢印] を選択し、キー修飾子として [**Shift**] を選択して、[カウント] フィールドに 2 を入力します。
  5. ステップを実行します。

  これにより、選択した範囲のセルがワークシート内に表示されます。

技術上の制限があるため、[キー プレス] オプション用に構成されたキー プレスの一部が機能しない場合があります。

色の変更

- 色の変更アクションを使用するには、[カラー ピッカー] でカラーを選択するか、[カスタム カラーの入力] でカスタム カラーを指定します。
- カスタム カラーを指定するには、[カスタム カラーの入力] でテキスト領域をクリックしてから必要なカラーを 16 進数形式で入力し、[設定] をクリックします。
- アクティブな色を設定してから、「テキスト色の設定」または「背景色の設定」を使用します。

クリップボードの操作

組み込み Excel ドライバーは、RoboServer インスタンスで実行されている口ボット用に設計されています。Windows クリップボードは実行中のすべての口ボット間で共有されているため、組み込み Excel ドライバーでは使用しないでください。

コピーと貼り付けは、同じ Excel ワークブック内のシート間でのみ実行できます。複数の Excel ワークブックを開いている場合に、あるワークブックからコピーして別のワークブックに貼り付けることはできません。

Excel ワークブックから情報を切り取りまたはコピーした場合、情報はクリップボードにコピーされず、「クリップボードから抽出」ステップは、Excel シートから切り取りまたはコピーしたコンテンツを提供しません。

Excel ワークブック内のコンテンツをコピーするには、ツールバーの「マークしてコピー」ボタンと「貼り付け」ボタン、またはコピー/貼り付けアプリケーション アクションとコンポーネント アクションを使用します。

ツリーの凍結操作

この操作によりセル ループがより高速に実行され、より効率的な口ボットを作成できます。

- ツリーの凍結 グループ ステップ内の Excel ドライバーを使用してワークブックにアクセスします。
- ツリーの凍結ステップ内で Excel ドライバーを使用するには、ワークフローに「ツリーの凍結」ステップを挿入し、Excel ドライバーを呼び出す「開く」ステップを挿入します。

アクション メニュー

- **[アプリケーション アクション]** メニューにアクセスするには、**[レコーダー ビュー]** の **[Excel]** タブを右クリックします。
- **[コンポーネント アクション]** メニューにアクセスするには、**[レコーダー ビュー]** またはツリー ビューでコンポーネントを右クリックします。

このメニューには、組み込み Excel ツールバーで利用可能なすべての項目および次のようなメニュー アクションが含まれています。

- 「AA」は、[アプリケーション アクション] メニューを表します。
- 「CA」は、[コンポーネント アクション] メニューを表します。

アクション	説明
[選択] AA および CA	現在のワークシートのセルを選択します。 [ステップを選択] の [選択] プロパティでセル アドレスを指定します。 たとえば、列 b の 4 番目のセルを選択する場合は `$b$4` を使用します。 セルの範囲を選択するには、[カーソルから展開] オプションを使用して、アクティブな選択範囲を、現在選択されているセルおよび新しい範囲が含まれる長方形の領域まで広げるか、A1:B2 という表記を使用します。
[シートを選択] AA	指定したワークシートを開きます。 [シートを選択] ステップの [シートを選択] プロパティにワークシート名を入力します。
指定タグまでスクロール AA および CA	スクロールして、選択したセルを画面上に表示します。 • AA の場合、入力したセルの位置までスクロールします。 • CA の場合、コンポーネント ファインダーの結果である場所までスクロールします。
オフセット AA	ポインターまたは選択範囲を、指定した行数および列数だけ移動させます。
値を取得 AA および CA	1 つ以上のセルの内容をテキスト変数に抽出します。 [次を抽出] を設定して、フォーマットされたセル、オリジナルのコンテンツ、または数式定義をセルから抽出できます。 複数のセルを含む [範囲] を指定している場合、結果内のセルのデータはタブと行で区切られます。 [ターゲット] を使用して検索範囲を指定します: [カーソル]、[選択]、または [カスタム]。 [結果] の変数には、選択されたセルの値が含まれます。

アクション	説明
数値を取得 AA および CA	セルの内容を数値変数に抽出します。このアクションはバイナリ データを抽出し、ロケール設定の影響を受けません。 [ターゲット] を使用してセルの範囲を指定します: [カスタム] または [カーソル]。 セルの内容を数値に変換できない場合、またはセルの内容が Excel の `#error` 値を含んでいる場合は、**[エラー値]** を返すように選択および指定します。それ以外の場合は 0.0 が返されます。 [結果] フィールドには、抽出されたセルの数値が含まれています。
ハイパーリンクを取得 AA および CA	指定したセルのハイパーリンクの URL を抽出します。セルにハイパーリンクが含まれない場合は空の値を抽出します。 [結果] フィールドには、抽出された URL を含む Text 型の変数が含まれています。 ハイパーリンクは、複数のセルを含む範囲に関連付けることができます。このアクションは、範囲内の任意のセルで機能します。
シート名を取得 AA	アクティブなワークシートの名前を抽出します。 [結果] フィールドには、抽出された Excel シートの名前を持つ Text 型の変数が含まれています。
コピー AA および CA	現在の選択範囲をコピーします。
貼り付け AA および CA	直前にコピーした選択を、選択したセルに貼り付けます。
クリア AA および CA	セルの範囲をクリアします。このアクションを使用して、フォーマット、コンテンツ、またはその両方をクリアします。
検索 AA	指定したスコープで新しい検索を開始します。 • [検索する文字列]: 検索するテキストを入力します。 • [大文字と小文字を区別する]: 大文字と小文字を区別して検索を実行する場合に選択します。 • [セル全体に一致]: セルの内容全体を検索対象のテキストと照合する場合に選択します。 • [検索方向]: 行または列で検索する場合に選択します。 • [検索対象]: 数式、結果、またはコメントを検索する場合に選択します。 • [指定したセルから開始]: 検索を開始するセルを指定します。 • [ターゲット] と [範囲]: 検索範囲を指定します。 • [結果を選択]: 検出されたセルを選択します。このオプションが有効で、セルがデバイス ツリーに含まれている場合は、ポインターもこのセルに設定されます。このオプションを使用して、選択範囲を適用し、検出されたセルに他のアクション操作を実行します。 変数 [結果] には、検出されたセルの場所が含まれます。アクションが失敗した場合、変数には空の文字列が含まれます。
次を検索 AA	直前に開始した検索アクションを続行します。 [結果を選択] を選択した場合は、検出されたセルが選択されます。このオプションが有効で、セルがデバイス ツリーに含まれている場合は、ポインターもこのセルに設定されます。このオプションを使用して、選択範囲を適用し、検出されたセルに他のアクション操作を実行します。 変数 [結果] には、検出されたセルの場所が含まれます。アクションが失敗した場合、変数には空の文字列が含まれます。

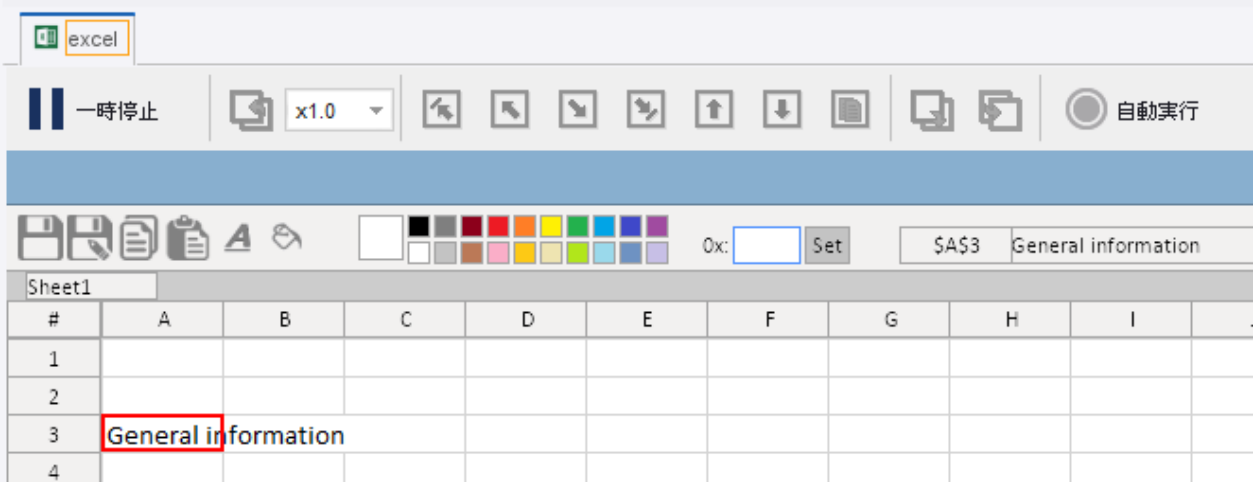
アクション	説明
値の設定 AA および CA	セルの範囲に値を設定し、オプションとしてセルにフォーマットを適用します。このアクションを使用して、データまたは式を設定します。 [フォーマット] で [カスタム] を選択すると、フォーマットは [カスタム フォーマット] フィールドの値に設定されます。
数値を設定 AA および CA	セルの範囲に数値を設定し、オプションとしてセルにフォーマットを適用します。コピーされる値はバイナリで、ロケール設定の影響を受けません。 [フォーマット] で [カスタム] を選択すると、フォーマットは [カスタム フォーマット] フィールドの値に設定されます。
フォーマットを設定 AA および CA	セルの内容を変更せずに、セルの範囲のフォーマットを設定します。 [フォーマット] で [カスタム] を選択すると、フォーマットは [カスタム フォーマット] フィールドの値に設定されます。 [カスタム フォーマット] のローカライズ要件については、「ロボットで Excel を使用する際のエラー」を参照してください。
ハイパーリンクを設定 AA および CA	セルの範囲にハイパーリンクを割り当てます。リンクは範囲の最初のセルに視覚的に表示されますが、範囲内のすべてのセルに適用されます。
シート名設定 AA	アクティブなワークシートの名前を変更します。
行の高さ設定 AA および CA	範囲内のセルを含むすべての行を特定の高さに設定するか、コンテンツに基づいてセルを自動調整する場合に選択します。 Excel では、列の実際の高さは指定した値の近似値となることに注意してください。
列の幅設定 AA および CA	範囲内のセルを含むすべての列を特定の幅に設定するか、コンテンツに基づいてセルを自動調整する場合に選択します。 Excel では、列の実際の幅は指定した値の近似値となることに注意してください。
表としてフォーマット AA	指定した範囲をテーブルに変換します。このアクションは、要求されたテーブル スタイルを使用して、選択したテーブル スタイル オプションを適用します。 [テーブルにはヘッダーがあります] オプションに対して [推測] オプションを使用すると、ヒューリスティックスを使用してテーブルにヘッダーがあるかどうかを判断するよう Excel に指示することができます。 テーブル スタイルに視覚的な手がかりがない場合、選択したテーブル スタイル オプションが表示されない場合があります。 このステップのオプションは、Excel のリボンの [テーブル ツール] タブと同じであり、同一のデフォルト設定を使用します。詳細については、Microsoft Excel のドキュメントを参照してください。
シート挿入 AA	ワークブックに、指定した名前の新しいワークシートを挿入します。このアクションによって、アクティブなワークブックが変更されることはありません。
行挿入 AA および CA	範囲の最初のセルを基準にして、1 つ以上の行を挿入します。
列挿入 AA および CA	範囲の最初のセルを基準にして、1 つ以上の列を挿入します。
シートを削除 AA	アクティブなワークシートを削除します。ワークブックの最後のワークシートを削除することはできません。

アクション	説明
行を削除 AA および CA	範囲内のセルを含むすべての行を削除します。
列を削除 AA および CA	範囲内のセルを含むすべての列を削除します。
コメントを追加 AA および CA	範囲の最初のセルにコメントを割り当てます。このコメントで既存のコメントが置き換えられます。
コメントの削除 AA および CA	範囲内のすべてのコメントを削除します。
セル タイプ判定 AA および CA	範囲内のすべてのセルに対してテストを実行します。 ロボットは、1 つまたは複数のセルをテストして、セルが空白であるか、テキスト、数値、論理関数、エラー、または式が含まれているかを検証できます。 セルが条件を満たす場合、[結果] 変数は `true` を返します。それ以外の場合は `false` を返します。変数はブール値である必要があります。 テストでは、Excel 関数と同等の関数を使用します。
名前付き範囲設定 AA および CA	名前付き範囲を、その範囲を参照するワークブックまたはワークシートに追加します。 ワークブックを保存した後に Excel のユーザー インターフェイスで範囲を非表示にするには、[表示可能] オプションをオフにします。
カラーを選択 AA	カラー変更アクションに使用できるカラーの中から 1 つを選択します。
カスタム カラーを選択 AA	RGB 16 進数形式で指定されたカラーを選択します。
テキストのカラーを設定 AA および CA	選択したセルのテキストにアクティブな色を適用します。
背景色を設定 AA および CA	選択したセルにアクティブな色を適用します。
保存 AA	新しいワークブックを保存し、ワークブックに次のような変更を加えます。 • 以前に保存を実行している場合は、最後に保存した場所に保存されます。 • ファイルから読み取った場合は、同じ場所に保存されます。 • 新規の場合、または変数から読み取った場合、システムはワークブックの保存場所を認識できないため、このアクションによって エラーが表示されます。 新しいワークブックにはファイル名が付けられていないため、[名前を付けて保存] ダイアログ ボックスを使用して保存する必要があります。変数から読み取ったソースは、[保存先] アクションを使用して保存する必要があります。 ワークブックを変数に保存するには、**[保存先]** アプリケーション アクションを使用します。 ワークブックを別の場所に保存するには、**[名前を付けて保存]** アプリケーション アクションを使用します。
名前を付けて保存 AA	ワークブックをファイル システムまたは RFS の場所に保存します。

アクション	説明
保存先 AA	ワークブックを変数に保存します。この変数はバイナリ タイプである必要があります。 後続のワークブックの保存は失敗します。[名前を付けて保存] または [保存先] を使用します。
[閉じる] AA	保存されていない変更を破棄して、Excel ウィンドウを閉じます。

グラフ シート

グラフ シートにはロボットで操作できる要素が含まれていないため、エディターのツリーには表示されません。ワークブックを保存すると、グラフ シートが含まれます。



ベーシック エンジン ロボットでの Excel の使用

Kofax RPA ベーシック エンジン ロボットのワークフローには、Microsoft ドキュメント用の Java API である Apache POI に基づいた組み込み Excel が含まれています。ベーシック エンジン ロボットで組み込み Excel を使用して、Excel ワークブックで操作を実行します。この操作を実行するコンピューターには Microsoft Excel がインストールされている必要があります。

各種機能

ベーシック エンジン ロボットについて、このトピックでは、Kofax RPA の組み込み Excel の Apache POI 実装でサポートされている機能およびサポートされていない機能を一覧で示します。

Design Studio では、Excel でスピルされた数式を作成することはできません。ただし、これらの Excel ドキュメントは Design Studio で開いたり、編集したり、保存したりすることができます。

機能

サポートされている機能とサポートされていない機能のリストを以下に示します。

詳細については、Apache POI の Web サイトを参照してください。

サポートされている機能
- 参照: 1 つのセルと領域、2D と 3D、相対と絶対
- リテラル: 数値、テキスト、ブール値、エラー、および配列
- 演算子：算術演算子と論理演算子、一部の領域演算子
- 組み込み関数: 350 以上の認識されている関数と 280 以上の評価ベース関数
- アドイン機能：分析ツールパックのうち 3 つ
- `[red]` などの書式文字列を使用するフォント色
- マイナスの数値が赤になっている、条件付きフォント カラー: `#.##0;[Red]-#.##0`
- 日付書式設定

サポート対象外の機能
- 配列とテーブルの式の操作 (Excel で、`=...` ではなく `{=...}` と表される式)
- 地域演算子の使用: union、intersection
- 前に未呼び出しのアドイン機能の解析
- 式の空白の保持 (POI での操作時)
- 太字サイズなどのフォントの変更。
- セルの背景色の設定
- 式からの外部ファイル参照の使用
- 非表示値の除外
- FLOOR、HOUR、MINUTE 関数を使用して文字列を日付に変換します。10 進値のみがサポートされています。
- SUBTOTAL 関数の使用

API 関数

サポートされている機能とサポートされていない機能のリストを以下に示します。

関数と制限事項の詳細については、Apache POI の Web サイトを参照してください。

サポートされている関数。

ABS、ACOS、ACOSH、ADDRESS、AND、AREAS、ASIN、ASINH、ATAN、ATAN2、ATANH、AVEDEV、AVERAGE、B

サポートされていない関数

ACCRINT、ACCRINTM、AMORDEGRC、AMORLINC、ASC、AVERAGEA、AVERAGEIF、AVERAGEIFS、BAHTTEXT、B

アクション、ステップ、およびデータ コンバータ

Kofax RPA で、Excel で使用できる次のようなアクション、ステップ、データ コンバータを使用します。これらのトピックについては、特定の RPA コンポーネントおよびユーザー インターフェイス要素に関連する他のセクションで説明されています。
- データ コンバータ
  - Excel 日付からの変換
  - Excel 日付への変換
- Excel 内ループ アクション (「Excel 内ループ」も参照)。

- コンテンツ設定 アクション:
  - セルのコンテンツ設定
  - 列のコンテンツ設定
  - 行のコンテンツ設定
- XML 形式表示 アクション
- 評価モードの設定 アクション

変数からの Excel ページの読み込み

Design Studio でページを読み込む最も一般的な方法はページ読み込みステップを使用することですが、ロボットが Excel ドキュメントをバイナリ属性の入力として受け取ることもできます。Excel ドキュメントをロボットにロードし、Excel ドキュメントからデータをループおよび抽出するには、以下の手順を実行します。

サイズの大きい Excel ファイルを操作している場合は、Design Studio が応答しなくなることがあります。この問題を回避するには、使用する Excel ファイルのサイズを小さくしてください。あるいは、マシンの物理メモリ (RAM) の容量を大きくするか、`memory.conf` (このファイルは、メモリ使用量を設定するために使用します) で割り当てるメモリ容量を増やします。

1. ロボットのワークフローで、[ページ生成] アクション ステップを挿入します。
2. [コンテンツ] リストでバイナリ変数を選択します。
   これは、ファイルをページ ビューにロードするために使用されます。
   変数を選択するには、[コンテンツ] リストの値を「値セレクター」に設定し、リストからバイナリ変数を選択します。
3. [詳細] を選択し、[オプション] ダイアログを開きます。
4. [ページ ローディング] タブの [ページ コンテンツ タイプ] で、[すべてのページで同じ] を選択します。
5. [コンテンツ タイプ] フィールドで、[**Excel**] を選択し、[**OK**] をクリックします。
   これで、ロボットが Excel ページからのバイナリ データを認識できます。

Excel のコンテンツの抽出

ベーシック エンジン ロボットの組み込み Excel 機能は、Excel ワークブックからデータを抽出し、抽出した結果を他のオートメーション タスクに使用するように設計されています。また、既存の Excel ドワークブックをデータを使用して更新することもできます。

新しいドキュメントを作成し、高度なスタイルや書式設定を適用する必要がある場合は、専用の Excel アプリケーションでテンプレート ドキュメントを作成し、ロボットを使用してデータを入力してください。スプレッドシートでは、組み込みの数値のフォーマットを使用できます。

サポートされているデータ形式を確認するには、`BuiltinFormats.getAll()` POI メソッドを使用するか、https://poi.apache.org/ に移動し、そのサイトで「組み込み形式」を検索します。

Design Studio には、スプレッドシートからコンテンツを抽出するためのステップが 3 つあります。

- セル値抽出ステップは、見つかった範囲からテキスト コンテンツを抽出するときに使用します。
- シート名抽出ステップは、見つかった範囲のシートのシート名を抽出するときに使用します。

- HTML として抽出ステップは、スプレッドシートの見つかった範囲を、その範囲のセルを持つテーブルが含まれる HTML ページとして変数に抽出するときに使用します。

セル値抽出ステップと HTML として抽出ステップの場合、セルから何を抽出するかを指定できます。これは、[次を抽出] オプションの値によって制御されます。ここでの選択は、スプレッドシート ビューのビュー モードの場合と同じです。可能なオプションについては、このトピックで説明しています。

書式設定された値

抽出した値は Excel に表示される値であり、日付と数値の値は書式設定されて抽出されます。つまり、数値は、セルの実際の値よりも桁数が少なくなる場合があります。

プレーン値

セルの値が書式設定されていなかった場合、抽出した値は、Excel に表示される実際の値です。たとえば、数値の小数は丸められません。

数式

セルに数式が含まれる場合、抽出されたものかどうかにかかわらず、プレーン値オプションで抽出される値と同じです。

スプレッドシート ビューを右クリックして、ステップを作成する場合、[次を抽出] の値は選択した [ビュー モード] の値に設定されます。[ビュー モード] を [数式] に設定してから、ページ ビューで右クリックして、コンテキスト メニューから **[抽出]** > **[テキスト]** (テキスト変数に) を選択すると、セル値抽出アクション ステップの [次を抽出] オプションが [数式] に設定されます。

抽出したコンテンツを再度書式設定 (または正規化) することが必要な場合があります。セル値抽出アクションでは、データ コンバータのリストを設定して、コンテンツの書式を再度設定できます。

このためには、スプレッドシート ビューで右クリックして、ステップを作成します。目的の抽出ステップを選択し、必要なパラメータを指定します。

Excel ファイルの共有数式

組み込み Excel ドライバーはドキュメントの共有数式をサポートしません。共有数式は 1 つのセルから別のセルに自動的にコピーされる数式があるセルのことです。共有数式が含まれる Excel ドキュメントの構造を変更する操作 (列の追加や削除など) で、ドキュメントのエラーが発生することがあります。

この制限が確認されているのは、Design Studio の外部で作成された Excel ドキュメントのみです。ロボットで作成された Excel ファイルに共有数式を含めることはできません。

回避策: Excel ドキュメントに共有数式が含まれていないことを確認します。1 つの数式セルを複数のセルにコピーする場合、同時に複数をコピーするのではなく、同時にコピーするのは 1 つのセルのみにします。

または、使用中の Kofax RPA インストールの Snippets フォルダに含まれる convertSharedFormulas.snippet ファイルを使用すると、Excel ドキュメントの共有数式を変換できます。このスニペットは次の手順を実行します:

1. 共有セルが含まれる Excel ドキュメントを特定します。
2. すべての数式セルをループします。
3. これらの各セルで数式を抽出し、再度セルに入力します。

セルの値の抽出

セル値抽出ステップを使用して、セルまたはセル範囲のコンテンツを変数に抽出します。

1. [アクション] タブで、[セル値抽出] を選択します。
2. [次を抽出] フィールドのオプションを選択します。
   見つかった範囲が単一のセルである場合、このセルの値が抽出されます。見つかった範囲に複数のセルが含まれる場合、セルの値がテキストとして抽出され、セルがタブで、行が改行でそれぞれ区切られます。両方の場合について、コンバータを抽出値に適用することにより、変数に格納される抽出値が作成されます。

   [次を抽出] オプションのこの値を考慮すると、本質的に、セルから抽出される値は Excel のセルのコンテンツです。空のセルの場合、値は空の文字列です。また、セルが C4:D6 などの結合されたセル (Excel でセルを結合して作成した) の一部である場合、結合されたセルの左上のセル C4 以外のセルでは、抽出された値は空になります。

シート名の抽出

シート名抽出ステップを使用して、シートの名前を抽出します。このステップは、値判定ステップと組み合わせて、すべてのシートをループしているときに指定された名前のシートをスキップするときに便利です。また、シート名をコンプレックス タイプの変数の属性に抽出し、シート名が返された値の一部になるようにするときも便利です。

1. シート名抽出ステップの [アクション] タブで、[シート名抽出] を選択します。
2. [変数] フィールドで、[テキスト] を選択します。
   このアクションにより、Excel ページの名前が変数に抽出されます。

HTML として抽出

HTML として抽出ステップを使用して、HTML タイプなど、構造化テキスト変数に格納される HTML ソース コードとしてスプレッドシート ドキュメントの一部を抽出します。抽出したコードには、Sheet1:A1:H17 など、抽出された範囲がヘッダー タグに含まれます。つまり、シートの名前はコードに含まれます。見つかった範囲のセルは、生成されたコードのテーブルに配置されます。このステップは主に、スプレッドシートの一部の HTML バージョンを口ボットに戻して、ブラウザで表示できるようにするために取得するためのものです。また、口ボットでこのステップを使用して、ページ生成ステップを使って抽出されたコードで HTML ページを作成することもできます。口ボットのパフォーマンスが低下する場合があるため、HTML として抽出ステップを使用して、スプレッドシートを HTML ページに変換し、そのコンテンツにアクセスすることはお勧めしません。

Excel 内ループ

Excel 内ループは多くの点で HTML 内ループに似ていますが、Excel の構造が単純であるため、HTML 内ループよりもはるかに単純です。基本的に、見つかったページの行、列、またはセルをループすることにより、ドキュメントのすべてのシートをループしたり、シートのセルをループしたりできます。Excel 内でループするには、Excel 内ループ ステップを使用します。このステップには、[最初のインデックス] や [増分] など、HTML 内でループするステップと共通の多くのオプションがあります。これらのオプションの詳細については、Excel 内ループ アクションのトピックを参照してください。

テーブルのすべての行をループするループ ステップを挿入できます。

1. [Excel ドキュメントからロードする口ボットを使用] で、Excel ビューの左上隅をクリックし、スプレッドシート全体を選択します。
2. 選択した領域内を右クリックします。

オプションのリストが表示されます。

3. **[ループ]** > **[選択中の行をループ]** > **[最初の行を除外]** を選択します。
   これにより、スプレッドシートのヘッダー行が検索から除外されます。ここで、Excel 内ループ ステップでループの最初のセルが名前付き範囲として設定されます。
   これで、名前付き範囲から抽出することが可能になり、ループのため、対応する値を他の行からも抽出できます。
4. ヘッダーのすぐ下にある列の最上部のセルを右クリックし、抽出する情報を選択します。たとえば、一連の定義を抽出するには、ID 列の最初のセルを右クリックし、[抽出]、[数値を抽出]、[ID] を選択します。
   書式パターンが正しく設定されたウィザードが表示されます。
5. **[OK]** をクリックします。
   ウィザードが終了します。
6. 名前付きの値に対して、手順 4 および 5 をそれぞれ繰り返して抽出します。
7. デバッグ モードに切り替えるには、[デバッグ] をクリックします。
8. ツールバーで、[実行] をクリックします。
   結果の値が表示されます。

シートと行のループ

テーブルが含まれた複数のシートおよび同じタイプのデータがある Excel ドキュメント内でループするように口ボットを作成することができます。たとえば、Excel スプレッドシートの各シートで年度の個別の月のアカウント情報が表示されるとします。この場合、口ボットが最初に複数のシートをループしてから、各シートの行をループするようにします。空のシートなど、他のシートと同じタイプのデータを含まないシートがドキュメントに含まれている状態への対処が必要になる場合もあります。次の図は、このような口ボットの構造を示しています。

この口ボットの最初のステップは、URL から Excel ドキュメントをロードするページ読み込みステップです。口ボットの次のステップは、ドキュメントのすべてのシートをループする Excel 内ループ ステップです。口ボットは、この最初のループ ステップの各イテレーションに対して、シートの各行をループする別の Excel 内ループ ステップを実行します。行をループするステップの「エラー処理」プロパティの 次の処理 が [次のイテレーションへ移動] に設定されます。つまり、ステップの範囲ファインダーがテーブル サイズを持つ範囲の照合に失敗すると、次のイテレーションに進みます。

この簡素化された工ラー処理は、シートが空になっている単純な状態に対応できますが、完全に異なるタイプのデータが含まれるテーブルがシートにある場合は対応できません。一般的には、シートの一部を抽出するステップを挿入し、その後に構造をテストするステップを挿入する必要があります。たとえば、列ヘッダーを抽出してから、指定された構造を持つ列ヘッダーであるかどうかをテストします。次の図は、口ボットに追加された工ラー処理を示しています。

この例では、ヘッダー抽出という名前のセル値抽出ステップがシートの最初の行を変数に抽出しており、値判定ステップには値を判定する条件があります。値が一致すると、ロボットは次のステップ (行のループ ステップ) を実行します。値が一致しない場合、ロボットは後続のステップ全てをスキップします。値判定ステップの Do プロパティは [後続のステップすべてをスキップ] します。

結合されたセルのループ

Excel の結合されたセルとは、隣接する 2 つ以上のセルが 1 つのセルに結合され、1 つのセルとして表示されるセルです。結合されたセルをループするようにロボットを設定することができます。結合されたセルのコンテンツはセルの左上のセルに格納され、他のセルは空になります。結合されたセルが含まれるテーブルをループすると、抽出の問題が発生する可能性があります。たとえば、受講者のテスト結果が表示された次のシートを見ると、一部の受講者はテストを受けておらず、結合されたセルを使用して 2 つのテストが示されている場合もあることがわかります。

	A	B	C	D	E
1	判定結果				
2		テスト1	テスト2	テスト3	
3	Alice	12	7	9	
4	Bob	到達不能		4	
5	Jane	11	8	7	
6	John	12	到達不能		
7	Zach	10	到達不能	7	
8					

受講者がテストを受けていない場合、テキスト "Missed" は数字ではないため、受講者のテスト結果を抽出するために行をループしても、結果が正しく抽出されないことがあります。これを修正するには、用語 "Missed" を検索するためのテストを挿入してから、失敗した結果に対して値 0 を格納します。このテストは、セルが結合されている状態では機能しません。前の例では、セル B4 に値 "Missed" が格納されているため、テストはセル B4 に対して適切に機能しますが、C4 のコンテンツは空の値になるため、C4 に対しては機能しません。

空のセルに対して別のテストを適用する代わりに、すべての範囲ファインダーで [**If Then**] データ コンバータを使用して、結合されたセル内の単一のセルを特定し、結合されたセルの左上のセルを返すことができます。

1. [ファインダー] タブの [説明] フィールドに [範囲ファインダー 1: 列 `at +2 (範囲名 "row")`] を入力します。
2. [範囲] フィールドで [行] を選択します。
3. [使用] フィールドで [指定位置の列] を選択します。
4. [列] フィールドで [インデックスで指定] を選択します。

5. [オフセット] フィールドに整数 [2] を入力します。
6. [高さ] フィールドで、**[範囲と同じ]** および **[高さが範囲の一番下までになります]** を選択します。
7. [結合セルの左上のセルを使用する] を選択します。
8. [アクション] タブの [次を抽出] フィールドで [書式設定された値] を選択します。
9. [コンバータ] フィールドに If Then ステートメントを入力します。例：`if contains "Missed" then "0" Else INPUT`
   セル値抽出では、テキスト "Missed" がテストされ、結果に対して 0 が使用されます。"Missed" が見つからない場合、抽出された値が使用されます。

Excel のセル タイプ判定

Excel ページ内のセルの内容をテストするには、まずセルの内容を抽出し、値判定ステップを使って実際にテストを実行します。これは基本的に、HTML などの他のページ タイプで行う場合と同様です。セルのセル タイプを決定するのは簡単ではなく、セルの内容を抽出してからテストを実行することができないこともあります。たとえば、セルが空白であるか、空のテキストが含まれているかを判断する方法はありません。しかし、Design Studio にはこのようなテストを実行するステップ (セル タイプ判定ステップ) が含まれています。

6 つの異なるセル タイプを判定することができます。ISTEXT や ISNUMBER などの関数を使用する Excel 上での判定に相当します。

空白
Excel 関数 ISBLANK に相当。

テキスト
Excel 関数 ISTEXT に相当。

数値
Excel 関数 ISNUMBER に相当。日付は Excel に数字として表されるため、このタイプには日付も含まれます。

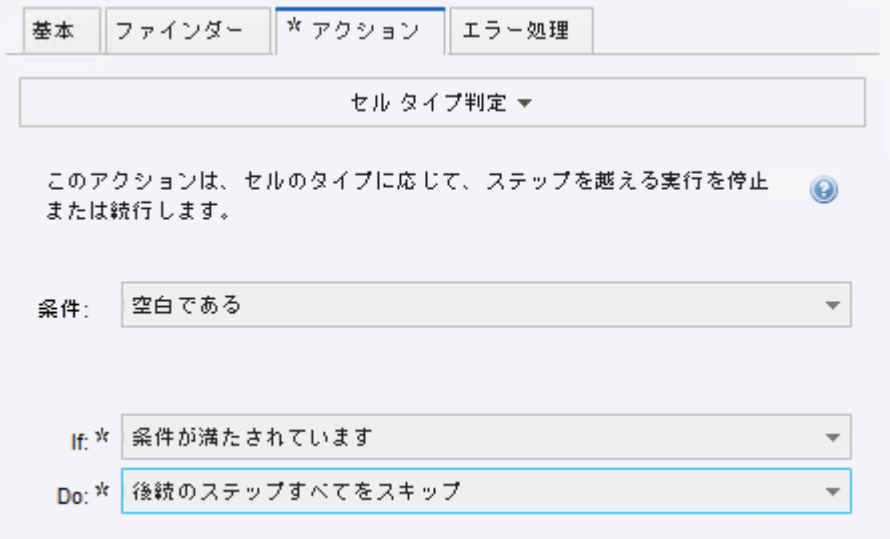
論理
Excel 関数 ISLOGICAL に相当します。Design Studio 内のブール型と関連しています。

エラー
Excel 関数 ISERROR に相当。

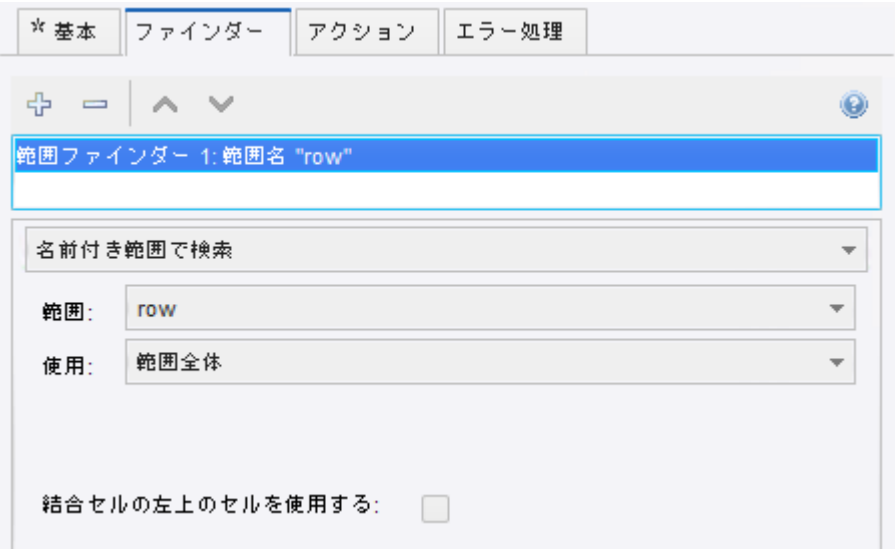
式
Excel 関数 ISFORMULA に相当。

セル タイプ判定は、他のテスト ステップと同様に機能します。見つかった範囲のセル タイプが指定されたタイプと一致するかを判定し、その結果に基づいて、分岐に沿って続行するか、次の手順をスキップするかを判断します。ステップについては、セル タイプ判定で詳細に説明しています。

セル タイプ判定ステップの重要な特徴は、多数のステップのタイプを同時にテストできることです。たとえば、全体が空の行の判定について考えます。空白行で区切られている同じ構造のテーブルを複数含む文書をループするとき、このテストは役に立ちます。次の図は、セル タイプ判定ステップの設定方法を示しています。この例では、見つかった範囲内のセルがすべて空白の場合、ステップに続く分岐はスキップされます。



次の図は、行全体を見つけるように範囲ファインダーを設定する方法を示しています。この例では、Excel 内ループ ステップが行をループし、セル タイプ判定ステップの前に実行されることで設定される「行」と呼ばれる名前付き範囲があります。使用プロパティに「範囲全体」を選択することで、結果が行全体となるように指定しました。



RoboServer

RoboServer は、クライアント向けサービスとしてロボットを実行する Kofax RPA のアプリケーションです。RoboServer を起動すると、ロボットの実行要求などのクライアントからの要求を受け付け、実行中のロボットが抽出したオブジェクトといった応答を返します。

RoboServer の詳細な説明については、『Kofax RPA 管理者ガイド』を参照してください。

RoboServer の開始

RoboServer はいくつかの異なる方法で開始できます:

- コマンド ラインから呼び出す。
- サービスとして実行する。サーバーを自動的に起動を参照してください。
- Docker コンテナを実行する。詳細については、『Kofax RPA 管理者ガイド』を参照してください。
- RoboServer プログラム アイコン (または、Management Console および RoboServer の両方を開始する Management Console プログラムの開始アイコン) をクリックする。デモおよびテストのみを目的としています。

コマンド ラインから RoboServer を呼び出すには、コマンド プロンプト ウィンドウを開き、以下の `Kofax RPA` インストール フォルダにある `bin` フォルダに移動し、次のコマンドを実行します:

```
RoboServer
```

必要なパラメータがすべて `C:Users\[ユーザー]\AppData\Local\[バージョン]\Kofax RPA \Configuration\roboserver.settings` 設定ファイルで指定されている場合、RoboServer が開始されます。

必要なパラメータのいずれかが欠落している場合、RoboServer はエラーを指定し、使用法のヘルプと使用可能なパラメータを表示します。

RoboServer パラメータ

RoboServer を開始するためのコマンド ラインは、次のパラメータを含めることができます。

```
RoboServer [-client] [-MC] [-mcUrl <arg>] [-cl <arg>] [-cpuThreads <arg>] [-h] [-
maxConcurrentRobots <arg>] [-maxQueuedRobots <arg>] [-ss <MC Shared Secret>] [-p <arg>]
 [-pauseAfterStartupError] [-s <arg>] [-sslPort <arg>] [-v] [-V]
```

RoboServer は、次の表のパラメータを受け入れます。RoboServer 設定アプリケーションですべてのパラメータを編集できることに注意してください。詳細については、RoboServer の設定 を参照してください。

パラメータ	説明
`-mcUrl <arg>`	この必須パラメータでは、次の形式で登録する Management Console を指定します。 `http[s]://[ホスト名]:[ポート番号]` 例：`-mcUrl http://myserver:8080/ ManagementConsole` **i** ロボットでコールバック オプションとともに Document Transformation ステップを使用する場合は、`-mcUrl` パラメータで RoboServer ホスト名または IP アドレスを使用します。Document Transformation サービスが Management Console にアクセスできなくなり、コールバック ロボットがキューに格納されなくなるため、「localhost」は使用しないでください。

パラメータ	説明
-ss --mcSharedSecret [MC 共有シークレット]	この必須パラメータにより、RoboServer を Management Console で認証するために使用される共有シークレットを指定します。共有シークレットは、Management Console の [サービス認証] セクションからコピーする必要があります。詳細については サービス認証 を参照してください。
-cl --cluster <arg>	この必須パラメータは Management Console で指定されたクラスタに RoboServer を自動的に登録します。以下の例では、RoboServer が *Production* クラスタにそれ自体を登録します。 例：-cl Production 例：-mcUrl http://myserver:8080/ManagementConsole -ss [MC 共有シークレット] -cl Production
-eh --externalHost <ポート番号>	RoboServer ホストの名前または IP アドレスを明示的に指定します。 このパラメータは、クラウド内で NAT を使用して実行しているとき、または Docker コンテナ内で RoboServer を実行しているときのように、RoboServer がローカルマシンで見つけたものとホスト アドレスが異なる場合に指定する必要があります。 例：-eh 10.10.0.123
-ep --externalPort <ポート番号>	RoboServer ホストのポート番号を明示的に指定します。 クラウド内で NAT を使用して実行しているとき、または Docker コンテナ内で RoboServer を実行しているときのように、RoboServer がローカル マシンで見つけたものとホスト ポートが異なる場合、このパラメータを指定する必要があります。
-jmxPass	JMX アプリケーションを使用して RoboServer を監視し、パスワードが必要な場合、JMX パスワードを設定します。
-v --verbose	このオプション パラメータにより、RoboServer がステータスおよびランタイム イベントを出力します。
-V --version	このオプションのパラメータによって、RoboServer でバージョン番号が出力され、終了します。
-h --help	ヘルプを表示します。
-pauseAfterStartupError	起動時にエラーが発生した場合に一時停止します。
-s --service <service-name:service-parameter>	このパラメータは、RoboServer が開始する必要のある RQL または JMX サービスを指定します。このパラメータは少なくとも一度指定する必要があり、同じ RoboServer で複数のサービスを開始するには複数回指定する場合があります。利用可能なサービスは、インストールによって異なります。 例：--service socket:50000 例：--service jmx:50100 詳細については、下の表で「利用可能なサービス」を確認してください。

パラメータ	説明
-p --port <ポート番号>	これは、-s socket:<port-number> を呼び出すための省略形です。 例：--port 50000
-sslPort <arg>	これは、-s ssl:<port number> を書き込むための省略形です。
-nd --NoDoc	このオプション パラメータにより、この RoboServer に対する 口ボットのドキュメント リクエストを禁止します。
-sn --serverName	これは、後で Kofax Analytics for RPA に表示される、RoboServer 統計を記録するためのサーバー名を設定するオプション パラメータです。サーバー名を指定しない場合、統計はサーバーの IP アドレスに基づいて収集されます。
-ll --licenseLimit <arg>	このパラメータは、RoboServer が受け取ることができるライセンス ユニットの最大数を定義します。
-client	RoboServer が Management Console クラスタ向けのクライアントとして実行されるように指定します。高可用性モードではサポートされません。
利用可能なサービス	
--service socket:<portNumber>	RoboServer が Management Console クラスタ向けのサービスとして実行されるように指定します。 <portNumber>: 監視対象のソケット サービスのポート番号。
--service ssl:<portNumber>	RoboServer を Management Console クラスタ向けのサービスとして実行し、セキュアな接続を作成することを指定します。 <portNumber>: 監視対象のソケット サービスのポート番号。
--service jmx:<jmx_port_Number>,<jmx_rmi_url>	<jmx_port_Number>: 監視対象の JMX サービスのポート番号。 <jmx_rmi_url>: JMX サービス向けのオプションの RMI ホストとポート。ファイアウォールを接続する必要がある場合に使用します。 例：example.com:51001

RoboServer と Management Console の間の接続タイプを設定するには、次のパラメータのいずれかを選択します。

- -client
- --service socket:<portNumber>
- --service ssl:<portNumber>

RoboServer に接続するコマンド ラインの例:

- Management Console クラスタ向けのクライアントとして:

```
-client -mcUrl http://localhost:8080/ManagementConsole -cluster <clientCluster> -ss
 <MC Shared Secret>
```

ここで、clientCluster は、Management Console で作成されたクライアント接続タイプのクラスタの名前です。

- Management Console クラスタ向けのサービスとして:

```
-service socket:50000 -mcUrl http://localhost:8080/ManagementConsole -cluster
 <serviceCluster> -ss <MC Shared Secret>
```

ここで、<serviceCluster> は、Management Console で作成されたサービス接続タイプのクラスタの名前です。

共有シークレットを設定する場合は、RoboServer 設定アプリケーションを使用できます。詳細については、RoboServer の設定 と サービス認証 を参照してください。

⚠ Kofax RPA バージョン 10 以降、すべての RoboServer は Management Console に自動登録する必要があります。したがって、Management Console を起動するときに、Management Console の URL と共有シークレット、およびクラスタ名を指定する必要があります (次の例のようにコマンド ラインで指定するか、[Management Console に登録] オプションで RoboServer 設定アプリケーションを使用します)。

```
RoboServer.exe -mcUrl http://myserver:8080/ManagementConsole -ss [MC 共有シークレット] -cluster Production -service socket:50000
```

サーバーを自動的に起動

インストールにサーバー機能が含まれている場合、サーバーが自動的に開始するように設定できます。

「サーバー機能」とは、RoboServer および Management Console (ライセンス サーバー) を意味します。実際には、この 2 つの機能は、このプログラムの起動時に指定した引数に基づいて、同じサーバー プログラム RoboServer から提供されます。

「RoboServer パラメータ」セクションには、RoboServer プログラム用のコマンドライン引数の詳細な説明が含まれています。RoboServer プログラムを有効にするには口ボットを実行し、 -service 引数を指定します。同様に、-MC 引数により Management Console 機能が有効になります (『インストール ガイド』の「Management Console」(ライセンス サーバー) を参照)。

RoboServer およびその他の RPA コンポーネントをサービスとして起動することについては、『Kofax RPA 管理者ガイド』の「RPA コンポーネントをサービスとして実行」を参照してください。

RoboServer のシャットダウン

RoboServer は、次のコマンド ライン ツールを使用してシャットダウンできます。引数なしで ShutDownRoboServer を実行して、サーバーをシャットダウンする方法、特に現在サーバーで実行中の口ボットを処理する方法のさまざまなオプションを確認します。

RoboServer の設定

RoboServer の設定は、RoboServer 設定アプリケーションで行います。RoboServer 設定は、Windows のスタート メニューから起動できます。RoboServer 設定の詳細については、『管理者ガイド』の「RoboServer 設定「」」を参照してください。

このアプリケーションを使用して以下の設定を行います。

- 一般: RoboServer 接続オプション、共有シークレットを含む Management Console 接続オプション、RoboServer ホスト設定、ライセンス ユニットの数、および詳細オプション。
- セキュリティ: 認証や権限などのセキュリティ設定。
- 証明書: 証明書の使用。
- プロジェクト: デフォルトのプロジェクトの場所。
- **JMX** サーバー: JMX サーバーの設定
- [**Management Console**]: 組み込み Management Console の設定。

設定を変更した後に、[**OK**] をクリックして新しい設定を保存し、実行中のいずれかの RoboServer を再起動して変更を有効にします。

⚠ 埋め込み Derby データベースを使用している場合は、[セキュリティ] タブの [ファイル システムとコマンド ラインのアクセスを許可] オプションが選択されていなくても、ロボットはコンピューター上でファイルの作成および編集を行うことができます。お使いのネットワーク環境では、MySQL または別のエンタープライズクラス データベースを使用することをお勧めします。

Kofax RPA バージョン 10 以降、すべての RoboServer は Management Console に自動登録する必要があります。そのため、クラスタ名と Management Console の URL および共有シークレットは、RoboServer の開始時に指定する必要があります (コマンド ライン、または **RoboServer** 設定アプリケーションを使用)。

[RoboServer ID] は、**Management Console** > **[管理]** > **[RoboServer]** > **[サーバー]** タブで RoboServer を識別するために使用される一意の識別子です。`roboserver-id` 変数値を編集することで、`roboserver.settings` 設定ファイルでこの ID を変更できます。

クラウド内で NAT を使用している場合、または Docker コンテナ内で RoboServer を実行している場合など、RoboServer がローカル コンピューターで見つけたものと RoboServer ホストの名前、IP アドレスとポート番号が異なっていれば後者を指定する必要があります。

RoboServer が使用できる RAM の最大容量を変更する必要がある場合は、『Kofax RPA 管理者ガイド』の「RAM 割り当ての変更「」」を参照してください。

プロキシ サービスの使用

一部の IP プロキシ プロバイダは、組み込みの IP ローテーション サービスの提供を開始しています。これは便利なサービスですが、IP プロキシを必要とする人々にとって、必ずしも使いやすいとは限りません。

IP アドレス変更後にブラウザ セッションが維持されない Web サイトや、セッションの維持が許可されていない Web サイトもあるため、Kofax RPA では、IP プロキシ プロバイダがランダムにローテーションを行う IP ローテーション モデル、または事前に設定した時点で IP のローテーションを行う IP ローテーション モデルの使用を推奨していません。

開かれたインターネットでは典型的なブラウザ ソケット接続の寿命は非常に短く、Web サイトはソースの IP アドレスを確認しないため、IP ローテーションは問題なく動作します。多くの Web サイトやショップは、最も基本的なセッション管理を実装しています。サイバー世界の脅威が増大し、セキュリティに注目が集まる中で、多くの Web サイトはセキュリティ レベルを引き上げて、IP アドレスの監視

を強化しています。Web サーバーでセッション内の IP アドレスの変化を検出して防止するようにすることは、第三者による攻撃を防ぐための適切な方法です。ユーザーがログインする必要のあるネット バンキング サイトやその他の多くの商用サービスや金融サービスのサイトが、セッション内の IP アドレス変更からの保護を実装しているのはこのためです。

IP アドレスを変更すると Web サーバーで進行中のセッションが中断される可能性があるため、ロボットの実行中に IP アドレスを自由にローテーションさせることはできませんプロキシを効果的に使用する最高の方法は、プロキシ切替ステップを使用したり、プロキシ サービスによって提供される Web サービスを使用したりして、ロボット内からプロキシを制御することです。

ローテーションがロボット内で行われ、リモート Web サイトを考慮して、ロボットが一つの IP アドレスセッションでトランザクションを行っていれば、このロボットは正常に動作します。

関連項目：

- 「プロキシ切替」ステップ
- Design Studio 設定のプロキシ サーバー
- Management Console でのプロキシ サーバーの構成

Kofax RPA の制限

以下のセクションでは、Kofax RPA 製品の制限について説明します。

一般

大量のデータ エレメントの収集時は、分割統治アプローチ向けにロボットを構築し、ロボットを実行するたびにデータ エレメントの一部のみが収集されるようにすることをお勧めします。

ブラウザ

- [スナップショット生成] を使用してダウンロードした Web ページには、Kofax RPA の内部ブラウザからダウンロードしたコンテンツとスタイルが表示されます。ダウンロードされたページのデスクトップ ブラウザでの表示は、そのブラウザでの元の Web ページの表示とは異なって表示されることがあります。

Excel

- Excel のグローバル変数をループする際には、特定のステップはループの内側、つまりループ ステップの後で許可されません。これは動的に強制されます。つまり、ステップが実行されるまでエラーは発生しません。次のステップは、ロボットがループ内部でループするグローバル変数での動作を常に拒否します。行挿入、列挿入、行除去、列除去、および シート名設定。
- Excel の修正はメモリ集約型で、大きな Excel 文書では動作しないことがあります。この制限は、設計プラットフォームまたはサーバー プラットフォームで使用可能なメモリ、ロボットが行う変更の数といった、多くの要因によって決まります。したがって、Kofax RPA で処理する Excel ドキュメントがどのような場合に大きすぎると判断されるかについては、正確な基準を指定することができません。
- 組み込み Excel で指定されている形式には Apache POI が使用されるため、Microsoft Excel と同じように表示されないことがあります。
- 式のサポート。Excel での数式のサポートの詳細については、Apache POI Web サイトを参照してください。
- サポートされている関数。サポートされている関数と関数の制限事項の詳細については、「各種機能」および Apache POI Web サイトを参照してください。

- サポート対象外の機能
  - 配列/テーブル数式の操作 (Excel で、"=..." とは異なり "{=...}" で表される数式)
  - 地域演算子：union、intersection
  - 前に未呼び出しのアドイン機能の解析
  - 数式の空白の保持 (POI での操作時)
  - 太字、サイズなどのフォント変更
  - セルの背景色
  - 数式からの外部ファイル参照
  - 非表示値の除外
  - FLOOR、HOUR、MINUTE 関数を使用して文字列を日付に変換します。10 進値のみがサポートされています。
- ロボットが非常に大きな Excel 文書で一部のアクションを実行するとき、処理が数百のイテレーションの後に遅くなることがあります。プロセスを高速にするには、Excel 処理についてロボットの設定を次のようにします。
  - グローバル変数の使用
  - 無視して続行エラー処理は使用しない
  - ロボットをデザイン モードで実行しない

    上記のすべての条件を設定して、Excel の操作時にエラーが発生した場合、変数値は空に設定されます。ロボットを調査し、サポート対象の Excel 機能を使用していることを確認し、変数の値が有効になるようにエラーを修正する必要があります。

詳細については、各種機能を参照してください。関連項目: Excel の互換性と制限事項。

実行モード

次のリストには、スマート再実行モードで利用できないステップが含まれています。

- Web ストレージ消去
- テキスト分割
- 画像抽出
- タグ非表示化
- タグ挿入
- スナップショット生成
- セル結合解除
- テーブル行除去
- タグ範囲除去
- タグ書き換え
- ページ再描画
- CSS 再描画
- テーブル行列入れ替え
- タグ表示化

Kofax RPAKapplets

次のリストには、Kofax RPA Kapplets で利用できない機能が含まれています。

- Kapplets への OAuth 口ボットの追加

大量のデータ出力を含む口ボットで Kapplets を使用することはお勧めしません。

日付コンバータ

- 年を 2 桁で入力している場合は、2029 年を境にした世紀の切り替わりが適用されます。
  例
  09/10/30 は 1930-09-10 00:00:00.0 になります。
  09/10/29 は 2029-09-10 00:00:00.0 になります。
- 日付内の数値は、値が 32 以上の場合のみ検証されます。値が 32 未満の場合、翌月の対応する日に日付が変換されます。
  例
  2020-02-30 は 2020-03-01 になります。
  2020-02-31 は 2020/03/02 になります。
  2020-02-32 はエラーになります。

口ボットのドキュメント

次のリストには、口ボットのドキュメントで利用できない機能が含まれています。

- 口ボットによって使用される Connector に関する情報は、口ボットのドキュメントには含まれません。

JavaScript

最新の JavaScript 構築には、ポリフィルを適用しても工ラーが解決されないものが多くあります。次のリストには、JavaScript の既知の問題が含まれています。

- let ステートメント
  CMAScript 2015 (6th Edition, ECMA-262)
- 定数
  CMAScript 2015 (6th Edition, ECMA-262)
- アロー関数式 () => {}
  CMAScript 2015 (6th Edition, ECMA-262)
- デフォルトの関数パラメータ
  CMAScript 2015 (6th Edition, ECMA-262)
- for...of ステートメント
  ECMAScript 2015 (6th Edition, ECMA-262)
- Rest パラメータ
  ECMAScript 2015 (6th Edition, ECMA-262)
- メソッド定義

```
var obj = {
          property( parameters… ) {},
          *generator( parameters… ) {},
          async property( parameters… ) {},
          async* generator( parameters… ) {},

          // with computed keys:
```

```
          [property]( parameters… ) {},
          *[generator]( parameters… ) {},
          async [property]( parameters… ) {},

          // compare getter/setter syntax:
          get property() {},
          set property(value) {}
        };
```

ECMAScript 2015 (6th Edition, ECMA-262)

ECMAScript 2016 (ECMA-262)

- Fetch API

```
fetch('http://example.com/movies.json')
          .then(function(response) {
            return response.json();
          })
          .then(function(myJson) {
            console.log(JSON.stringify(myJson));
          });
```

- リソース リクエストを表す Fetch API のリクエスト インターフェイス

```
var a = new Request(url);
```