Kofax RPA

ユーザー ガイド

バージョン: 11.0.0

日付: 2019-12-17

KOFAX

目次

はじめに 11
関連ドキュメント 11
トレーニング 12
Kofax 製品のヘルプの入手 12
第 1 章: 概要 14
第 2 章: チュートリアル 15
ビギナー チュートリアル 15
概要 15
ロボット ビギナー チュートリアル 15
Kapplet ビギナー チュートリアル 15
タイプ ビギナー チュートリアル 15
アドバンスド チュートリアル 15
分岐、ロボット状態、実行フロー 15
ループの基本 16
トライ ステップ 16
Excel 16
データの変換 16
パターン 16
スニペット 16
日付抽出 - シンプルなケース 16
日付抽出 - 複雑なケース 16
API 17
第 3 章: Design Studio 18
Design Studio について 19
ロボット 19
スニペット 28
変数とタイプ 29
ライブラリとロボット プロジェクト 30
命名規則 30
セキュア パスワードの処理 31
Design Studio ユーザー インターフェース 32
メニュー バー 32
ツールバー 33
マイ プロジェクト 35

エディター ビュー....38
ロボット エディター....38
タイプ エディター....44
テキスト エディター....44
Design Studio のウィンドウ....44
一般編集....45
タイプ....47
タイプ属性....48
ステップ アクションとデータ コンバータ....49
パターン....49
エクスプレッション....51
エクスプレッションの試行....52
エクスプレッションの編集....54
プロジェクトとライブラリ....55
ロボット プロジェクトの操作....56
ロボット ファイルの整理....56
シェア プロジェクトの使用....57
データベースの操作....58
データベースのマッピング....59
タイプとデータベース....60
データベース警告....60
データベース テーブルの生成....61
データベースへのデータ格納....62
ロボットの構造....65
適切に構造化されたロボットの記述....66
ページ タイプの判定....68
タグ ファインダーの使用....69
タグ パス....69
タグ ファインダーのプロパティ....70
タグ ファインダーの設定....71
フォーム送信....72
フォームの基本....73
ステップ アクションの決定....74
フォームでのループの使用....75
ファイルのアップロード....75
ページ ビューでのコンテキスト メニューの使用....76
ページのループ スルー タグ....77
class の同じタグのループ....77

class の異なるタグのループ....78
HTML ページのループ....79
最初のページにある他のすべてのページへのリンク....79
次ページにリンクしている各ページ....80
待機基準の使用....82
HTML からのコンテンツの抽出....88
テキスト抽出....88
バイナリ データの抽出....89
ページ ビューでコンテキスト メニューを使用....89
一般的なタスクの実行....89
HTML テーブルからコンテンツを抽出する....90
テーブル コンテンツの不規則性の処理....91
テーブル構造の不規則性の処理....91
ロボットでのローカル ファイルの使用....92
変数から Excel ページ読込....93
Excel のコンテンツの抽出....94
セルの値の抽出....95
シート名の抽出....95
HTML として抽出....95
Excel のセル タイプ判定....96
Excel 内ループ....97
シートと行のループ....98
結合されたセルのループ....99
アプリケーション ビューでの変数の使用....100
変数を開く....100
変数の変更....101
JSON の使用....102
JSON の用語....103
JSON MIME タイプ....103
JSON とステップ アクション....103
JavaScript オブジェクトとしての JSON....105
エラーの処理....106
別のエラー処理方法....107
一般的なケースにおけるショートカット....108
At ターゲット....110
ルーピング....111
Try-Catch....113
ロボット ビューでのエラー処理の識別....114

スニペットの作成と再利用 114
変数とスニペット 115
スニペットのベスト プラクティス 118
ロバストなロボットの作成 118
セッションの再利用 119
既存のタイプの修正 120
ロボット設定 121
デフォルトのロボット設定から変更を表示 121
他のブラウザ エンジンにロボットを切り替える 125
クラシック ブラウザにロボットを切り替え 125
デフォルト ブラウザにロボットを切り替え 126
変数の設定 127
ロボットのデバッグ 129
基本的なデバッグ 129
デザイン モードで現在のステップからデバッグ 131
デバッグ ロケーションからデザイン モードに戻る 131
ブレークポイントの使用 132
シングル ステップ 132
ステップ イントゥー 132
Design Studio の設定 133
一般 133
テキスト ファイル 134
ロボット エディター 134
Desktop Automation 134
ローカル データベース 135
プロキシ サーバー 136
証明書 137
バグ レポート 138
Management Console 138
第 4 章: Desktop Automation 140
Windows 10 ユーザーへの注意 140
Desktop Automation の概要 142
はじめに 144
古い Desktop Automation アクション ステップの変換 146
オートメーション デバイスの参照 146
スタティック リファレンス 146
ダイナミック リファレンス 146
トリガーで参照 147

オートメーション デバイス マッピング 147
Desktop Automation ワークフローを編集する 149
エディターの手順 153
Desktop Automation サービスの設定 153
Desktop Automation の前提条件 153
Desktop Automation エージェントの設定 153
ローカル Desktop Automation の使用 163
ファインダー 165
ファインダーのタイプ 166
セレクター構文 171
再利用可能なファインダー 172
9 グリッド イメージ ファインダー 174
ツリー モード 181
インテリジェント スクリーン オートメーション (ISA) 182
Windows オプション 186
Desktop Automation ステップ 188
割り当て 189
参照 189
バンドル 190
クリック 190
条件 191
デバイスに接続 191
コンポーネント セレクターのコピー 192
コピー 192
値の変換 193
カスタム アクション 195
デバイスからの切断 203
Document Transformation 203
テキストを入力 210
Excel 211
クリップボードを抽出 211
画像抽出 212
画像からテキスト抽出 213
ツリーを XML として抽出 214
値を抽出 216
ツリーの凍結 217
グループ 217
ガード チョイス 218

セッションの開始....221
KTA....221
繰り返しステップ....223
マウス移動....228
通知....229
開く....229
キー プレス....231
ファイルの読み込み....232
リモート デバイス アクション....233
リターン....234
スクロール....235
クリップボード設定....235
ターミナル....235
Throw....236
トリガー チョイス....237
Try-Catch....238
記録されないインスタント クリック....243
ウィンドウ....243
ファイル出力....243
ログ出力....244
ターミナルの自動化....244
基本 ターミナル チュートリアル....258
Web サイトのアクセス....260
Chrome Inspector を使用したデバッグ....262
Chromium 組み込みブラウザでの Cookie の管理....263
組み込み Excel ドライバー....265
Windows 環境のヒント....268
アテンデッド オートメーション....269
TLS コミュニケーションを使用....270
データの変換....271
エクスプレッション....272
エクスプレッション エディター....281
変数....286
数値の制限....287
RDP 接続の使用....288
リモート デスクトップの管理....289
Desktop Automation サービスの管理....289
第 5 章: Management Console....290

Management Console の構造の概要....290
命名規則....291
Management Console の起動....291
Management Console の設定およびユーザー インターフェイス....292
スケジュール....292
リポジトリ....300
ログ ビュー....320
管理....322
データベース タイプの追加....358
JMX....359
OAuth....359
サポートされているサービス プロバイダー....360
アプリケーションの追加....360
ユーザーの追加....362
ロボットの書き込み....364
資格情報を持つスケジュール ロボット....365
アウト オブ バンド アプリケーション....366
第 6 章: Kofax RPAProcess Discovery....367
Process Discovery Agent の構成....367
設定ウィンドウ....367
バージョン情報ウィンドウ....370
タスクの開始....370
タスクのキャンセル....371
ヘルプを開く....371
Process Discovery Analyzer の構成....371
Process Discovery Analyzer オプション....372
第 7 章: Kofax RPA Kapplet....374
Kapplet のユーザー インターフェイス....374
メイン ユーザー インターフェイスの要素....374
ツールバー....375
サイド メニュー....376
ユーザー メニュー....377
ユーザー ロール....378
Management Console バックアップからの Kofax RPA Kapplet の復元....378
Kapplet の作成とメンテナンス....379
ワークスペースの作成....379
テンプレートの作成....379
Kapplet の作成....382

Kapplet の実行 384
Kofax RPA Kapplet のインストール 384
第 8 章: リファレンス 385
Design Studio 385
ステップ アクション 385
データ コンバータ 508
タイプ エディター 549
テーブルの作成と削除 554
プロトコル 554
ロボット ライブラリ 555
Management Console へのアップロード 556
その他のトピック 556
RoboServer 612
RoboServer を開始 612
RoboServer 設定 614
RoboServer 設定 - ヘッドレス モード 615
Management Console 618
その他のトピック 618
Kofax RPA Classic Kapplet 621
Kapplets の作成とメンテナンス 622
Kapplets のインストールと使用 626
Kapplet ブランディングのカスタマイズ 629
Management Console での Kapplet のアクセス 629
プロキシ サービスの使用 630
Kofax RPA でのパスワード暗号化 631
Kofax RPA の制限 631

はじめに

このガイドは、Kofax RPA ヘルプの PDF 版です。

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/11.0.0_qrvv5i5e1a/RPA.htm

このガイドの他に、ドキュメント セットには次の項目が含まれています。

Kofax RPA Release Notes (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 Installation Guide (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 Administrator's Guide (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 Desktop Automation スタート ガイド
Kofax RPA Desktop Automation を使用してロ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。

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 Desktop Automation サービス設定ガイド
リモート コンピュータで Desktop Automation を使用するために必要な Desktop Automation サービスを設定する方法について説明します。

Kofax RPA Developer's Guide (Kofax RPA** 開発者ガイド**)
RoboServer 上でロボットを実行するために使用される API に関する情報が含まれています。

[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、『Installation Guide』(インストール ガイド) を参照してください。

Kofax RPA Integration API documentation (Kofax RPA* 統合 *API* 文書*)

Kofax RPA へのプログラムでのアクセスを提供する Kofax RPA Java API および Kofax RPA .NET API についての情報が含まれています。Java API 文書は、オンラインおよびオフラインの Kofax RPA 文書から入手できますが、.NET API 文書はオフラインのみとなります。

注 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 の名前は後方互換性を確保するために残されています。API 文書の中では、RoboSuite という用語は Kofax RPA と同じ意味で使われています。

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の両方を提供しています。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、www.kofax.com の Kofax Web サイトをご覧ください。

Kofax 製品のヘルプの入手

Kofax Knowledge Base (Kofax ナレッジ ベース) 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 Base (ナレッジ ベース) で情報を検索することをお勧めします。

Kofax Knowledge Base (Kofax ナレッジ ベース) を参照するには、Kofax Web サイトにアクセスして、ホームページでサポートを選択してください。

注 Kofax Knowledge Base (Kofax ナレッジ ベース) は Google Chrome、Mozilla Firefox または Microsoft Edge 向けに最適化されています。

Kofax Knowledge Base (Kofax ナレッジ ベース) で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。

  Kofax Knowledge Base (Kofax ナレッジ ベース) のホームページをスクロールして、製品ファミリーを見つけます。目的の製品ファミリー名をクリックして、関連記事の一覧を表示します。一部の製品ファミリーの場合は、関連記事を表示するために Kofax Portal (Kofax ポータル) の有効なログイン情報を入力する必要があります。
- Kofax Customer Portal (Kofax カスタマー ポータル) へのアクセス (資格のあるカスタマー向け)

  ページ上部にある **Customer Support** (カスタマー サポート) リンクをクリックしてから、**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 (資格のあるカスタマー向け)

  ページ上部にある **Partner Support** (パートナー サポート) リンクをクリックしてから、**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。

  General Support (一般サポート) セクションまでスクロールして、**Support Details** (サポートの詳細) をクリックし適切なタブを選択します。

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。

ビジュアル エディター Design Studio では、統合したいアプリケーションとデータ ソースをクリックして、自動化されたワークフローを作成します。

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。ロボットはデータベース、ファイル、API、Web サービスおよび他のロボットにアクセスすることもできます。データをアプリケーションからエクスポートし、別のアプリケーションにロードします。必要に応じて途中でデータを変換することもできます。

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端末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、Desktop Automation 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、ロボットは RoboServer で一括実行をスケジュールすることも、Java および C# API 経由のオンデマンド、または調整可能な REST サービスで実行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 Kapplets という特定目的のエンドユーザー 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。

Management Console を使用して、負荷分散、フェールオーバー、RoboServer の正常性の監視、ユーザー ロールと権限の管理を行うこともできます。

第 2 章

チュートリアル

このセクションのトピックには、Kofax RPA でさまざまなタスクを実行するのに役立つビデオ チュートリアルへのリンクが含まれています。各チュートリアル ページには、ビデオの書き起こしがあります。

注 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

このセクションに含まれているチュートリアルでは、Kofax RPA の概要と、本製品の最初のプロジェクトについてのガイドを提供します。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。ビデオを再生するには、ビデオのリンクをクリックします。

注 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。

概要

入門チュートリアル ビデオ

ロボット ビギナー チュートリアル

ロボット ビギナー チュートリアル ビデオ

Kapplet ビギナー チュートリアル

Kapplet ビギナー チュートリアル ビデオ

タイプ ビギナー チュートリアル

タイプ ビギナー チュートリアル ビデオ

アドバンスド チュートリアル

このセクションには、Kofax RPA の高度なトピックのチュートリアルが含まれます。

分岐、ロボット状態、実行フロー

分岐、ロボット状態、実行フローについてのチュートリアル ビデオ

ループの基本

ロボットの基本ループについてのチュートリアル ビデオ。

フォームにおけるループ

フォームに適用する繰り返しステップについてのチュートリアル ビデオ

次を繰り返し

繰り返しループを説明しているチュートリアル ビデオ

トライ ステップ

ロボットに条件および エラー処理を設定するためにトライ ステップを使用する方法についてのチュートリアル ビデオ

Excel

Excel ドキュメントからのデータ読み取り方法についてのチュートリアル ビデオ

Excel への書き込み

Excel ドキュメントへの書き込み方法についてのチュートリアル ビデオ

データの変換

ロボットでのデータ変換についてのチュートリアル ビデオ

パターン

パターンと、Design Studio での使用方法についてのチュートリアル ビデオ

スニペット

ロボット間でステップを共有するためにスニペットを使用する方法についてのチュートリアル ビデオ

日付抽出 - シンプルなケース

News Magazine ロボットを変更して記事の日付を抽出する方法についてのチュートリアル ビデオ

日付抽出 - 複雑なケース

日付情報が複数の場所に散らばっている場合に日付抽出する方法についてのチュートリアル ビデオ

API

JSON および REST 呼び出しを使用して LinkedIn API にアクセスするロボットを作成する方法についてのチュートリアル ビデオ

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。

構成

ヘルプの Design Studio セクションは、次のように構造化されています。まず、Design Studio の最も重要な概念が紹介されます。次に、ユーザー インターフェイスについての説明があり、ロボットのコアな構成要素の概要が提供されます。基本についてしっかりと学習してから、Design Studio を使用して具体的な作業を実行するロボットを作成する方法について、チュートリアルを開始します。チュートリアルは、自身が定義したタスクを実行するロボットを作成できるようになるまでに徐々に高度になっていきます。チュートリアルはヘルプのこのセクションの要であり、続行する前にこれらを習得することは重要です。

以下について読む前に

続行する前に、Kofax RPA の概要を読むことをお勧めします。ここでは Design Studio および使用されているコンテキストを紹介し、基本的なチュートリアルについて説明します。

> 注 チュートリアルにアクセスするには、インターネットへのアクセスが必要です。

Design Studio を使用するには、プログラミング、HTML、JavaScript に関する基本知識が必要です。

その他のリソース

Design Studio の追加情報については、参照ドキュメントを参照してください。

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。ロボットは、データソース (通常はウェブサイトやその他のロボットですが、Excel ドキュメントやデータベースも該当します) に関するタスクを実行するようにデザインされたプログラムです。一般的には、各データ ソースでタスクごとに 1 つのロボットを記述します。たとえばロボットは、http://cnn.com からニュースを抽出するために 1 つ、http://yahoo.com からニュースを抽出するために別のものを 1 つ、オンライン製品カタログから製品情報を抽出するために別のものをもう 1 つ、というように作成します。

基本的に、ロボットには、ブラウザで実行可能なあらゆる動作を (自動的に) 実行し、データベースや Excel 文書からデータを抽出して、データベースやファイルに格納されたデータと結合させる、といったプログラミングが可能です。

Kofax RPA では、Web オートメーション ロボット (通常は単に「ロボット」と表記) と Desktop Automation ロボットの 2 種類のロボットが区別されています。2 番目のロボットの詳細については、Desktop Automation の章を参照してください。

重要 ロボットの実行を許可するか、ロボット間で実行権限を切り替えるには、ロボットが表示されているタブを開き、ツールバーで [実行の準備] をクリックする必要があります。正しく実行されるようにするために、複数のロボットを同時に実行することはできません。

また、デザイン モードとデバッグ モードで同時に実行することもできません。

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 (フル)

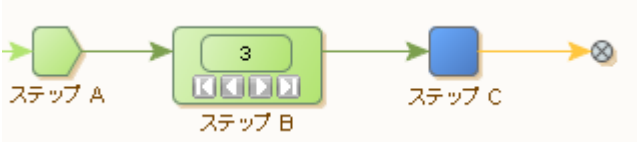
スマート再実行モードにおいて、デザイン モードでのロボットの実行方法は、実行時またはデバッグ モードで実行される方法と同様です。このモードは新規ロボットのデフォルトのモードです。

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。

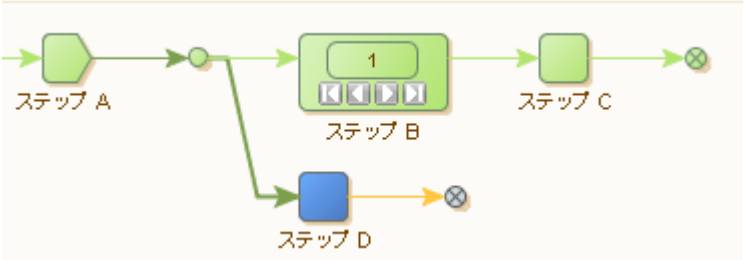
[値判定] ステップの青色の感嘆符アイコンは、[次の代替手段を試行] へのエラー処理がトリガーされたことを示します。

ループ内のステップをクリックすると、繰り返しステップによって、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、以下のロ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ロ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、ロ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ロボットの構築が容易となります。

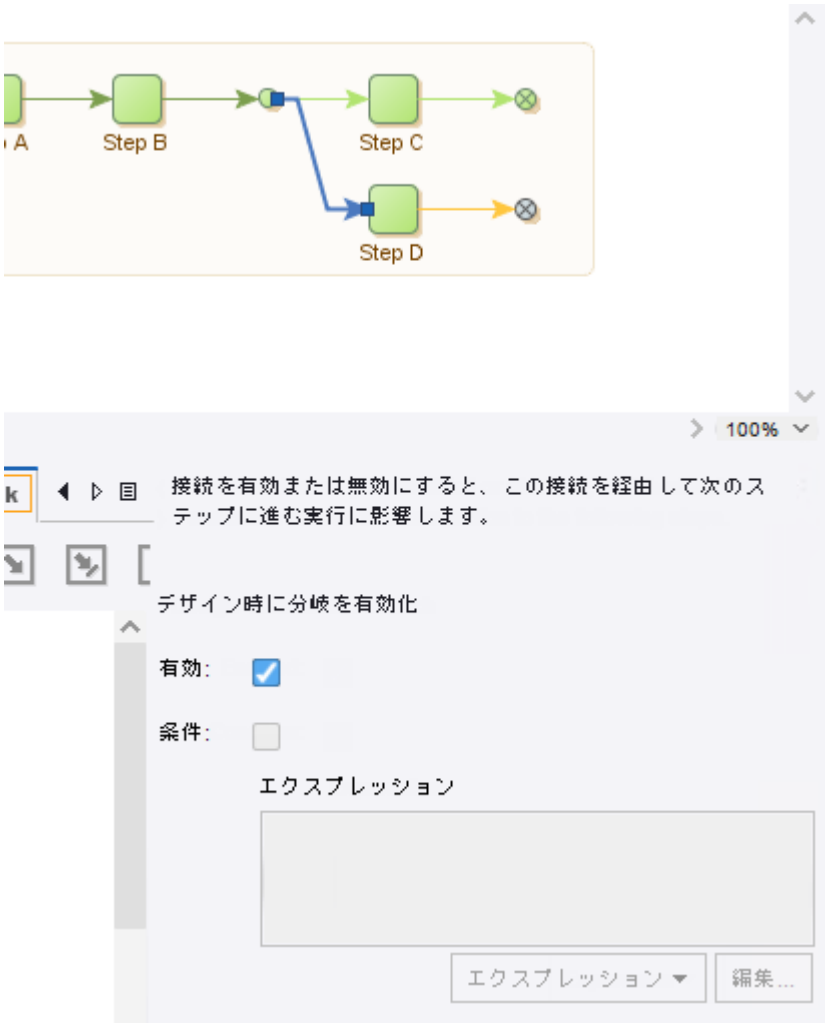
スマート再実行では、ウェブサイト、データベース、またはウェブ サービスによる外部とのやりとりがキャッシュされます。実行結果を格納するための前提条件 (ロードする URL を決定する変数など) が変更されるまで、キャッシングによってステップの再実行が回避されます。スマート再実行モードには、最小実行モードよりも高いメモリ フットプリントがあります。

スマート再実行モードは、Desktop Automation ワークフローをサポートするモードです。Desktop Automation 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。キャッ

シュされた変数は、Desktop Automation ワークフローが更新された場合も更新されません。Desktop Automation 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。Desktop Automation 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します。

外部と重要なやりとりを行う大型ロボットや長時間稼働するロボットには、スマート再実行モードはお勧めしません。こうしたロボットでは、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。



接続構成

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)。

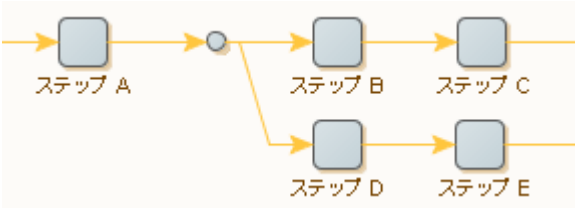
重要 スマート再実行モードでは、一部のステップ アクションが利用できません。使用できないステップの一覧については、Kofax RPA の制限トピックの「実行モード」の項をご覧ください。

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ロボットはすべてこの実行モードを使用します。

ロボット 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、ロ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ロ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ロボットには、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ロ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ロ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。

ロボット状態

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。このページには、HTML ページ、スプレッドシート、XML ページなどがあります。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て特定のページ 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。

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。

ステップには、以下の 5 種類があります。

- アクション
- トライ
- スニペット
- グループ化
- 終了

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。唯一の例外は、終了ステップです。終了ステップは、ロボット内の分岐の最後をマークしますが、ロボットの最後はマークしません。たとえば、終了ステップの後にロボットが必ずしも実行を停止するとは限りません。終了ステップは、ロボットで出力接続を持たない唯一のステップです。

ステップには、ステップ名、タグ ファインダーのリスト、ステップ アクション、エラー処理などのプロパティがあります。アクション ステップにはこれらのプロパティがすべてありますが、他のタイプのステップには一部のプロパティしかありません。

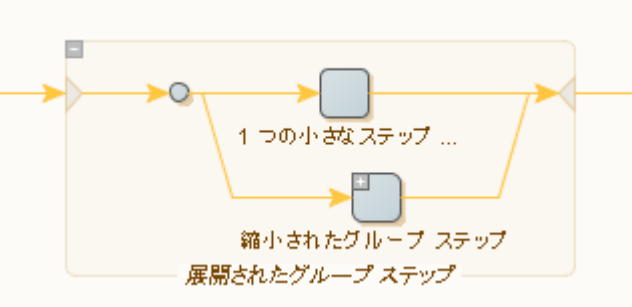
ステップ名は、たとえば「見出しの抽出 (Extract Headline)」や「検索ページの読込 (Load Search Page)」など、ステップの内容を表す名前にします。前述のロボットでは、ステップ名は "MyStep" です。

ファインダーは、ステップ アクションが動作するページ内の エレメント (HTML/ XML タグまたは Excel セル) を検出します。ステップ アクションには単一の 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 アクションには エレメントをまったく受け入れないものもあります。ファインダーには以下の 2 種類があります。HTML や XML ページでタグを検索するタグ ファインダー、および Excel ページのセルを検索する範囲ファインダー

ステップ アクションは、ステップが実行するアクションです。アクションはステップの「中枢部」で、ロボット記述で問題となるのは適切なステップ アクションを選択することです。たとえば、抽出アクションは HTML ページ内のタグからテキストを抽出し、変数に格納します。クリック アクションは、<a> タグにある URL をロードし、ロボット状態の現在のウィンドウのページを、新しくロードされた HTML ページに置き換えます。アクションは、通常、ロボット状態を変更します。たとえば、抽出アクションは変数を変更し、クリック アクションはページ/ウィンドウ、Cookie、および認証を変更します。

ステップを実行することができます。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 アクションを順番に適用し、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 アクションの中には "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 "loop steps" と呼びます。繰り返しステップは 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。下の図は、内部に縮小されたグループ ステップがある状態で、グループ ステップを展開した場合の例です。



ステップは、実行が行われるように適切に構成されている場合に有効となります。たとえば、ステップにアクションがない場合、実行ができないため無効になります。

また、ステップ定義によって エラー処理 を指定します。

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。

注 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ロボットがあるとします。

このロボットは、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 つの出力ロボット状態が生成される場合、ロボットは次のように実行されます。最初のロ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 により出力ロボット状態が生成されます。この出力ロボット状態はステップ B の入力ロボット状態です。ステップ B はステップ C の入力ロボット状態である ロボット状態を生成します。ステップ C が実行され、出力ロ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」。

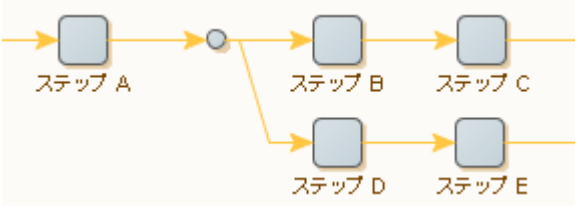
ステップは実行時に出力ロボット状態を生成しないことがあります。これは、エラーまたはテストステップが原因で実行がロボット内の別のステップで続行された際に発生します (条件とエラー処理を参照)。

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、そのたびに異なるロ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 ロ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 つのロボット状態を出力し、その他すべてのステップが 1 つのロ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 ロボット状態は別のロボット状態です。各イテレーションにより新しい ロ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ロボット状態を受け取ります。

詳細については、分岐、ロボット状態、実行フロー チュートリアルを参照してください。

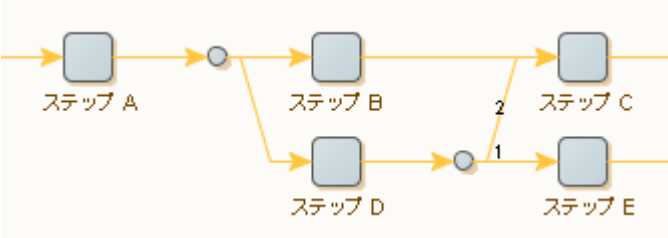
1 つのステップを複数のステップに接続できます。これは「分岐」と呼ばれます。次のような ロボットがあるとします。



このロボットで、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 つの出力ロボット状態が生成される場合、先行ロ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ロ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。次のようなロボットがあるとします。



このロボットは、接続が明示的に順序付けられる様子を示しています。このロボットでは、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 つの出力ロボット状態が生成される場合、ロ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ロ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ロ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) したい場合があります。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。

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。

注 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。

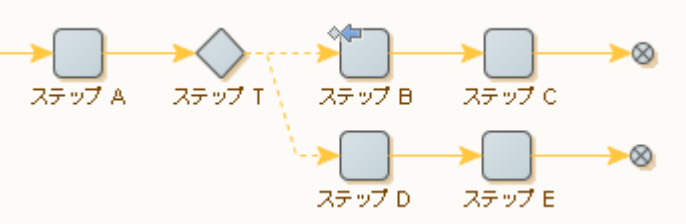
条件によって、入力ロボット状態 (HTML ページに特定のタグがあるなど) のコンテンツに基づき実行のフローが変更されます。エラー処理とは、特定のエラーの発生時に実行のフローを変えることです。たとえば、存在が想定されるアンカー タグが HTML ページに見つからず、クリックできない場合などです。多くの場合、状況は次の 2 通りになります：アンカー タグが見つかった場合 (条件) にはクリックする。または、ロボットがアンカー タグのクリックを試行してエラーを処理できる (アンカー タグが見つからなかった場合)。一部のケースでは、一般的に条件と考えられているものは非常に複雑でそれ自体を記述することはできません (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、ページ読込し、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サインです。たとえば、Web サイトがダウンし、ページ読込 エラーが発生した、または HTML ページの動的なページ レイアウト変更のためにタグ ファインダーで必要なタグを見つけられなかった場合です。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「次の代替手段を試行」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「次の代替手段を試行」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「次の代替手段を試行」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 "try-catch" 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、ロ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ロケーションとロケーション コードと共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ロケーションは、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。次のロ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、そのロ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」(このロケーションには、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。

ロケーション コードはロ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。前述のロケーションの例では、ロケーション コードは以下のようになります。`{a-i1-a}`Design Studio のロケーション 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 [指定したロケーションへ移動] を使用)。

重要 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}`

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。

その他の情報については、スニペットのチュートリアルを参照してください。

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。これは、スニペット エディターで編集され、ロボットにスニペットが実行されると表示されます。

変数とタイプ

Design Studio では、変数とタイプが重要な概念になります。

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。

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。
- 通常、ロボット内部で一時変数として使用されます。
- 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。各ニュース変数には、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 タイプが含まれます。出力される各ニュース値は、各名前付き属性の、一意な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、入力変数に割り当てられる値でロボットの入力の一部として入力変数を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。

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配布および展開するときは、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、いつでも 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配布または展開する手段になります。

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　のプロジェクトは、複数の Design Studio の間で共有可能です。シェア プロジェクト ビューは、シェア プロジェクト ファイルのステータスを視覚的に示すだけでなく、説明付きのヒントも提供します。

命名規則

Kofax RPA は、プロジェクト名、スケジュール名、フォルダ名 (パス)、および API を使用してアップロードされたロボット、タイプ、スニペット、リソースなどのフォルダ アイテムに適用される、以下の命名規則を採用しています。

- 不正なシステム文字を許可、ただし警告メッセージを生成。
- 空の名前を許可せず、エラー メッセージを生成。
- 243 文字を超える名前を許可せず、エラー メッセージを生成。
- 特殊な HTML 書式の名前を許可、ただし警告を生成。HTML フォーマットを除去。
- システム予約語は使用しないでください。システムで予約された名前を持つファイルを Windows システムのディスクに保存することはできません。次に、システム予約語のリストを示します。CON、PRN、AUX、NUL、COM1、COM2、COM3、COM4、COM5、COM6、COM7、COM8、COM9、LPT1、LPT2、LPT3、LPT4、LPT5、LPT6、LPT7、LPT8、LPT9。

- 名前 `local` は予約された名前であり、Desktop Automation で使用することはできません。

以下のような、ファイル名のピリオド "." の扱いに注意してください。

- ピリオド “.” で始まるファイルとフォルダは隠しファイルとして扱われ (隠しフォルダ内のすべてのファイルも非表示になります)、プロジェクト ツリーには表示されない。
- 隠しファイルは、プロジェクトの同期には含まれない。
- ロボット、スニペット、タイプ、テキスト、およびデータベース マッピング ファイルには、ピリオド "." で始まる名前をつけることができない。
- フォルダにはピリオド "." で始まる名前をつけることができない。

名前変換アルゴリズム

タイプ付きの名前	結果
'<t> aaa </tag>' 注 「」は文字列の開始と終了を示します。	'aaa' 注 HTML タグ <t> および </tag> は除去されます。実際のフォルダ/プロジェクト/スケジュール名の前後のスペースも除去されます。
'< t> hello '	'< t> hello' 注 タグ < t> は、有効な HTML タグではないため保持されます。実際の名前の後のスペースは除去されます。
'com1 /*** /& @'	'com1/***/& @' 注 "com1" は一部のオペレーティング システムでは予約語なので警告を生成しますが、フォルダ、プロジェクト、またはスケジュールには有効な名前です。アスタリスク "*" は不正な文字です。Kofax RPA は、有効な名前としてアスタリスクを受け入れますが、警告を発します。有効なファイル名として名前がオペレーティング システムでサポートされていないローカル コンピュータに他のユーザーがプロジェクトをダウンロードする場合、予約語と不正な文字を含む名前を入力すると、エラーが発生する可能性があります。 & のような HTML エンティティ番号は実際の文字に変換されます。

セキュア パスワードの処理

Design Studio では、ライセンス サーバーと Management Console のパスワードをディスクに格納しないように設定できます。

ライセンス サーバーについて [パスワードの記憶] オプションをオフにした場合、Design Studio ではライセンス サーバーのパスワードを保存しないため、Design Studio を起動するたびにライセンス サーバーでパスワードを入力する必要があります。ライセンス サーバー情報ウィンドウを開くには、**[ファイル] > [ライセンス サーバー設定]** に移動します。

[**Design Studio** 設定] ウィンドウの [**Management Console**] タブの Management Console に対する [パスワードの記憶] オプションをオフにすると、Design Studioでは Management Console のパスワードは保存されません。Design Studio から Management Console に接続するたびにパスワードを入力する必要があります。

注 デフォルトでは、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オフになっています。この機能を有効にして設定するには、ユーザーおよびグループの管理の「ログイン試行回数のチェック」を参照してください。

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 インターフェースについて説明し、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

Design Studio を開始してユーザー インターフェース ツアーへ進んでください。ただし、このツアーで扱う Design Studio ユーザー インターフェースは、ロボットの作成またはロードをしていない状態の、起動時における表示です。

Design Studio 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 Installation Guide』(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。
- [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 リファレンス ヘルプ、ドキュメンテーション、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すぐに、利用可能なメニューとして以下の編集メニューが追加されます。

- 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。

タイプまたはロボット ファイルを開くと、さらに以下のメニューが利用できるようになります。

- 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。

ロボット ファイルを開くと、さらに以下の 3 つのメニューが表示されます。

- 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。
- 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。このメニューは、ロボット エディターがデバッグ モードの場合にのみ使用できます。

ツールバー

ツールバーのボタンを使うと、メニューでも使用できる多くの操作を実行できます。

使用可能なボタンは、エディター ビューでアクティブなエディターに応じて変更されます。

アイコン	説明
	プロジェクトを開く
	すべてのファイルを保存
	すべてを同期
	ロボット設定
	終了 - エスケープ
	実行の準備
	更新

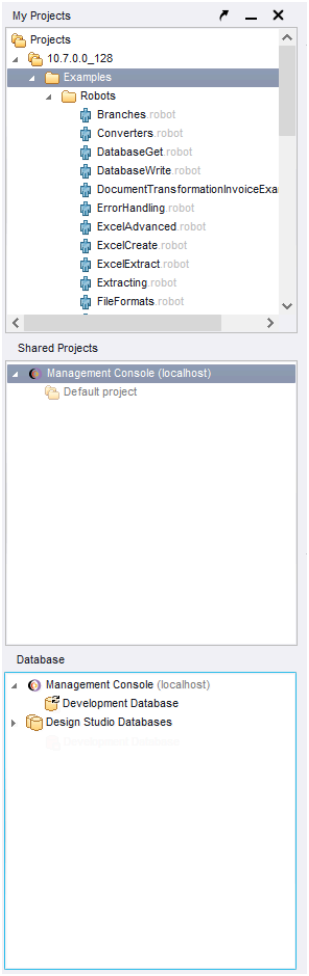
アイコン	説明
	DA ロボットにステップ イントゥー
	元に戻す
	やり直し
	切り取り
	コピー
	ステップの前に貼り付け
	削除
	選択したステップの前にステップを挿入する
	選択したステップの後にステップを挿入する
	選択したステップからブランチを追加する
	グループ化
	グループを解除
	選択した範囲からスニペットを作成
	スニペットをグループに変換
	ステップまたは接続を上に移動

アイコン	説明
	ステップまたは接続を下に移動
	すべて広げる
	すべて閉じる
	デバッグモード に切替
	現在のステップからデバッグ開始
	Management Console からロボットをダウンロード
	Management Console にロボットをアップロード

Desktop Automation に関連付けられたボタンのリストについては、Desktop Automation ワークフローを編集するを参照してください。

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配置されます。



[マイ プロジェクト] ペインは、[マイ プロジェクト] ビュー、[シェア プロジェクト] ビュー、および [データベース] ビューで構成されています。

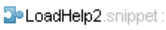
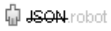
マイ プロジェクト ビュー

[マイ プロジェクト] ビューには、Design Studio で開いているプロジェクトを表す、展開または縮小された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。この表示には、開いている任意の数のプロジェクトが含まれます。[マイ プロジェクト] では、右クリックで開くコンテキスト メニューから、フォルダに新しいロボットを作成する、または以前に保存したロボットを開くといったさまざまなアクションを実行できます。

シェア プロジェクト ビュー

[シェア プロジェクト] ビューには、接続している Management Console に対するロボット プロジェクトを示す展開/縮小されたツリー構造が表示されます。[シェア プロジェクト] ビューのプロジェクトがコンピュータの Design Studio で共有されると、両方のプロジェクト ビューにそのプロジェクトが表示されます。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の表示は異なります。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。また、[シェア プロジェクト] ビューには、ヒントや同期に関する問題が表示されます。

アイコン	説明	意味
	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 に存在しますが、Design Studio にダウンロードされていません。
	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 と同期しています。
	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。
	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。
	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、まだ Management Console にアップロードされていません。
	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 ビュー

データベース ビューには、Design Studio のデータベースと、接続されている Management Console が表示されます。

Management Console への接続を構成するには、**[設定]** > **[Design Studio 設定]** > **[Management Console]** 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 [データベース] ビューに表示するには、Design Studio ユーザーと共有するクラスタ データベースに対しデータベース マッピングが存在する必要があります。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。

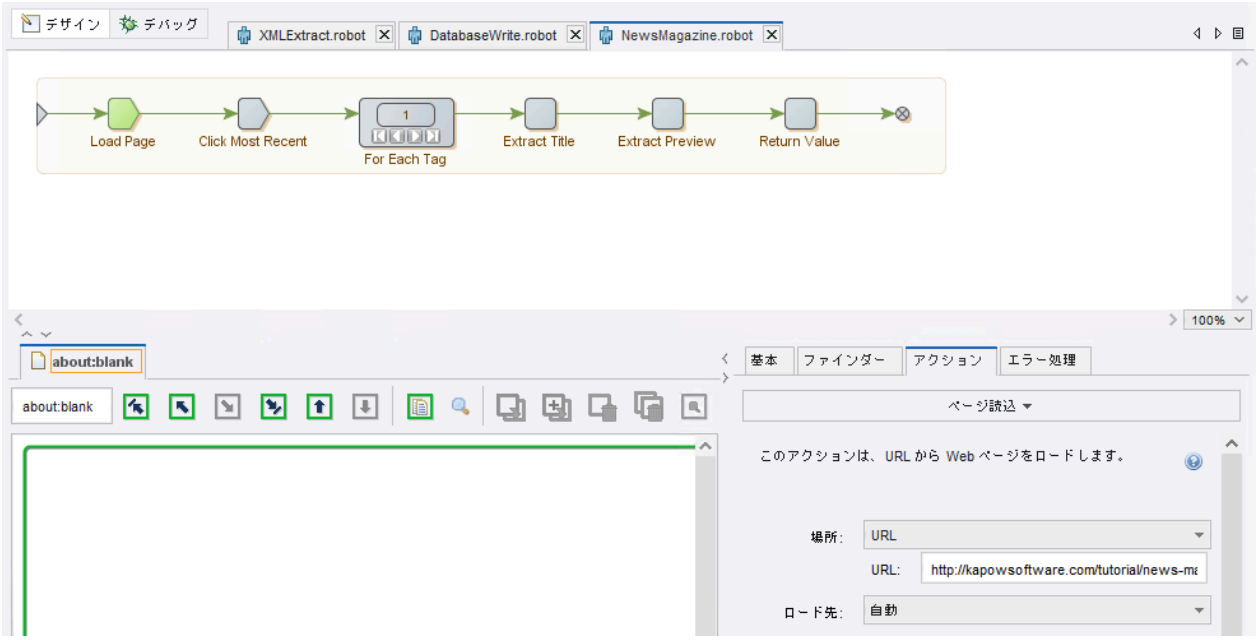
重要 データベース マッピング、タイプ、およびドライバは、Management Console と Design Studio のコピー間の接続が次のイベント中に確立または更新されるときにのみ Management Console からフェッチされます。

- 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 (更新するには、データベース ビュー 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

Web オートメーション ロボットおよび Desktop Automation 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、エディター ビューの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。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。

Desktop Automation ロボットの編集の詳細については、Desktop Automation ワークフローを編集するを参照してください。

各ビューには複数のサブ表示が含まれています。デザイン モードでは、サブ表示には以下のようなものがあります。

- ロボット ビュー
- アプリケーション ビュー
- ステップ ビュー
- 変数ビュー
- フレーム ビュー

ロボット ビュー

ロボット ビューは、タブの下にあるロボット エディターの上部に位置しています。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、または接続などを実行して、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。その基本的な考え方は、ユーザーが構築しているロボットの一部が、構築中に実際に実行されるということです。現在のステップはこの実行の位置をマークし、Studio のページ ビューと変数ビューで状態を表示します。

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。ロボットの実行中には、クリックしたステップが黄色で表示されます。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。現在のステップは、常にステップ ビューで設定します。

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。ロボットは、分岐ノードまたは末端ノードに達するまで、この経路上で実行を繰り返します。現在の実行パスは、接続がより暗い色でマークされます。現在の実行パスを変更するには接続をクリックします。この操作によって、その接続がパスに含まれます。

パスの制限なしの実行

パスの制限なしモードでは、現在のステップから必要なステップまで実行を続けることができない場合でも、ワークフローの必要なステップまで確実に到達できるようにします。この状況は、他のステップで制限が強制されている場合に多く発生します。

アイテムの選択

一連のステップまたは接続を選択するには、Ctrl キーを押したままアイテムをクリックします。また、マウスの左ボタンを押したまま、ステップをドラッグして選択することもできます。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。

ステップまたは接続が選択されている場合は、このステップまたは接続にアクションを適用できます。たとえば、新しいステップを挿入するには、ステップを選択してツールバーの をクリックします。また、ステップまたは接続を右クリックして、リストからアクションを選択することもできます。

注 ステップまたは接続を右クリックすると、アクションが自動的に選択されます。

アクションの編集

ロボット エディターを使用すると、長い範囲のアクションをステップと接続で実行することができます。こうしたアクションには、コピー、貼り付け、切り取りおよび削除などの標準の編集アクションや、ループのイテレーションを変更するなど、デザイン ビューでのロボットの実行に関するものがあります。現在のステップ (他のステップが選択されていない場合)、選択されたステップまたは選択された接続のいずれかで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。

別のステップを設定するには、そのステップを選択 (Ctrl キーを押しながらクリックするか、周囲の選択ボックスをドラッグ) し、F2 キーを押すか、コンテキスト メニューの [ステップを設定] を選択します。

詳細については、一般編集を参照してください。

アプリケーション ビュー

アプリケーション ビューは、ロボット エディターのロボット ビューの下にあります。アプリケーション ビューには、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 ビューには、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矢印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 アクションを適用できます。

各ウィンドウに対してクラシック ブラウザ エンジンを使用すると、ページのタイプに応じて、ページ ビューがいくつかのサブ ビューに分割されます。たとえば、ロードしたページが HTML ページの場合、ページ ビューにはサブ ビューが含まれます。ページには、HTML、XML、JSON、Excel、バイナリの 5 つのタイプがあります。HTML とバイナリでは同じビューが使用され、その他のページ タイプでは独自の特別なページ ビューが使用されます。

注 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**]** > **[**デフォルト オプション**:** 設定**]** > **[**レガシー タブ**]** > **[**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。

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。

ステップ ビュー

ステップ ビューには、現在のステップの構成が表示されます。タブをクリックすると、次のプロパティの表示と編集ができます。

- [基本]: ステップの名前と、関連付けられた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]: ステップのファインダーのリストを表示および設定します。通常は、ページ ビューでエレメントを右クリックして、ファインダーを構成します。タグ ファインダーの使用を参照。
- [アクション]: ステップのアクションを表示および設定します。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]: 現在のステップのエラー処理方法を確認します。エラー処理を参照してください。

変数ビュー

変数ビューには、変数のリストが含まれています。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ビューの右側に表示されます。このビューには、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。[変数ビュー] ウィンドウに類似した変数ビューが表示されます。このウィンドウには、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。

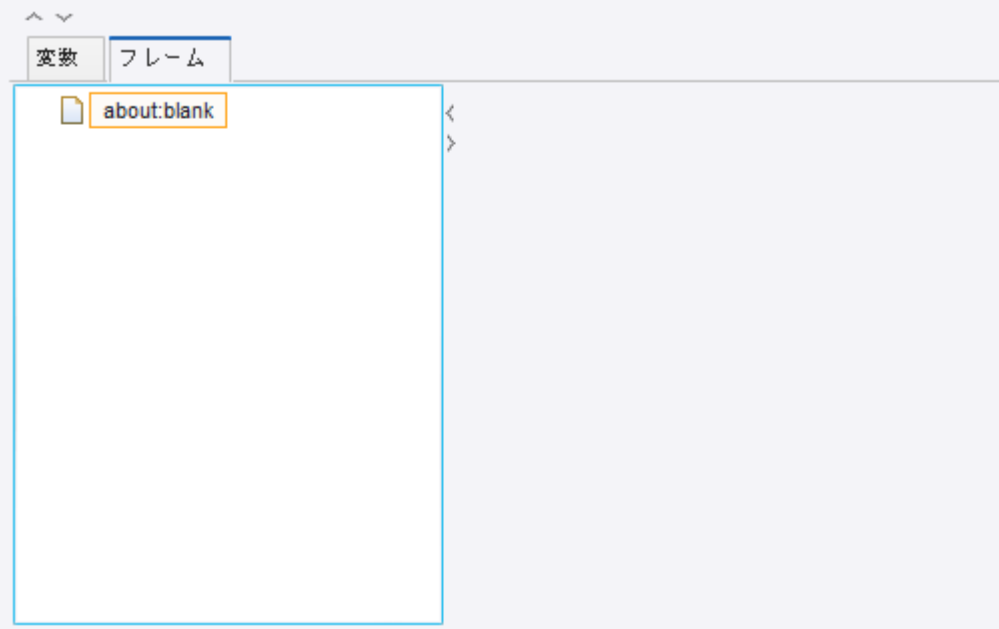
ロボットの記述とテストを行うときは、初期入力変数の値を使用します。実稼働環境でロボットを実行するときは、入力変数が、ロボットを実行しているアプリケーションによって決定された値に初期化されます。

注 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 ビュー

[フレーム] ビューは、Design Studio の右下隅にある [変数] タブの横にあります。



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。このビューには、選択したフレームの詳細を表示するプレビュー パネルも含まれてます。Design Studio バージョン 9.6 以降では、フレーム ビューでのみフレームの概要を見ることができます。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の両方で ⊘ が表示されます。

注 [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構成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、このパターンをロボットのブロックされた URL パターンのリストに追加します。
- [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。

注 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。

デバッグ モード

ロボット エディターには、ロボットをデバッグするための特殊なモードが含まれています。ツールバーで、[デバッグ] または [デザイン] をクリックして、デザイン モードとデバッグ モードを切り替えます。これは、Design Studio のメイン ウィンドウ ツールバーでも使用できます。または、Design Studio の現在のステップからデバッグするには、[デバッグ] をクリックします。

デバッグ モードのロボット エディターの上部には、デザイン モードのものに似たロボット ビューも含まれています。

注 デバッグ モードのロボット ビューには、ロボットを実際にデバッグしている場合にのみ現在のステップが含まれます。この現在のステップは、デザイン モードのロボット ビューの現在のステップと常に同じというわけではありません。

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。構成によっては、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ロ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、[Goto] ボタンをクリックして、エラーが生成されたステップに移動できます。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。

詳細については、ロボットのデバッグを参照してください。

タイプ エディター

メイン ウィンドウを使用して、現在編集されているタイプを設定することができます。特に、属性テーブルでタイプの属性を設定できます。属性テーブルの下にあるボタンを使用して、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。

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) を扱う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、 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。

一般編集

このトピックでは、Design Studio での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、ステップ ビューでロボット、タイプ エディターでタイプ、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有効なものです。

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

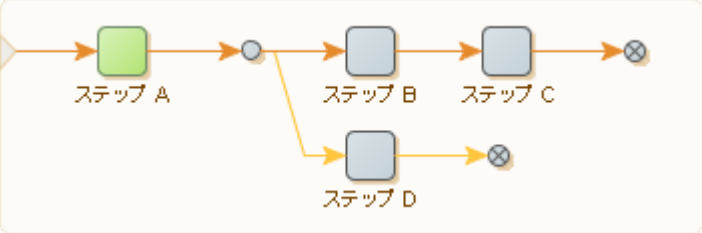
さらに、ステップのファインダー一覧などの多くのリストでは、**Ctrl-Shift-C** を使ってリスト内のすべてのアイテムをコピーすることができます。

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 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 つ以上の出力接続)
- ステップ B、C、D および 2 つのエンド ステップ (複数の入力接続)

接続の付近で (Ctrl キーを押しながら) クリックするか、またはドラッグの選択内に含めることによって、さらに多くのグループ ステップを選択できます。

ステップまたはステップの集合のグループを解除するには、グループを解除するアイテムを選択し、ツールバーまたはステップのコンテキスト メニューのグループを解除 をクリックします。

注 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 [閉じる] を使用します。

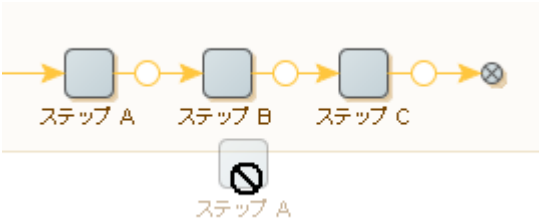
選択した 1 つ以上のグループに対してこのアクションを実行するには、コンテキスト メニューの [展開する] と [閉じる] を使用します。

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えますが、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。



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カーソルを置くと、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 [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。

タイプ

ロボットで使用するパラメータを定義するタイプを作成および維持します。

関連するすべてのプロパティを構成することが重要です。そうでない場合、タイプは想定どおりに実行されない場合や、無効になる場合があります。

タイプには、以下のものが必要です。

- 有効な名前。タイプ名は文字またはアンダースコアで始まる必要があり、文字、数字、アンダースコアのみを含むことができます。

  注 Design Studio では、名前に拡張子を含みません。例えば、ファイル名が ExampleType.type のタイプは、Design Studio では ExampleType として利用可能です。名前付けの詳細については、命名規則を参照してください。

- 一意の名前。同じプロジェクトにある 2 つのタイプを同じ名前にすることはできません。
- タイプの使用目的を示すタイプの種類。

属性テーブルの下にある「タイプの種類」ドロップダウン リストを使用して、タイプの種類を選択できます。レガシーのタイプの種類「データベース出力タイプ」が必要な場合以外は、タイプの種類「スタンダード タイプ」が常に使用されるため、通常は選択する必要がありません。詳細については、Design Studio のリファレンス ドキュメントを参照してください。

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構成する必要があります。各属性には名前とタイプの両方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
整数	12 のような整数。可能な範囲は、-9223372036854775808〜9223372036854775807 であり、両端を含む。
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。
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。
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。
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。
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。
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点を除いてロング テキストと同じです。
XML	XML ドキュメント。適格な XML 文書のみが許容される点を除いてロング テキストと同じです。
日付	"1992-04-25 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。
バイナリ	バイナリ データ。 任意のバイト配列。
イメージ	イメージ。イメージはプレビューできる点を除いてバイナリ データと同じです。
PDF	PDF 文書。PDF 文書はプレビューできる点を除いてバイナリ データと同じです。

属性タイプ	説明
セッション	Cookie、認証などを含むセッション。
通貨	ユーロの "EUR" など、ISO-4217 規格で定義されている通貨コード。
国	ドイツの "DE" など、ISO-3166 規格で定義されている国コード。
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。
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。

ステップ アクションとデータ コンバータ

Design Studio では、各アクションとデータ 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アクションまたはデータ 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 アクションまたはデータ 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。

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。例えば、[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。例えば、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" のときに出力は "ROBOMAKER" になります。

パターン

パターンに慣れていない方は、パターンに関する紹介ビデオをご覧ください。

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、テキストには 2 つの数字が含まれ、2 桁のみ (d は数字を表す記号) であることを示す規則的な方法、つまりパターン \d\d として表すことができます。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) を付けて工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するが、"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 では非常に重要な抽出手法です。

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、*? 演算子が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。

[シンボル] ボタンは、パターンに特殊記号を入力したいとき、非常に便利です。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。

使用できる特殊記号やパターンの詳細については、パターンの内容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、エクスプレッションの参照ドキュメント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。
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。

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。

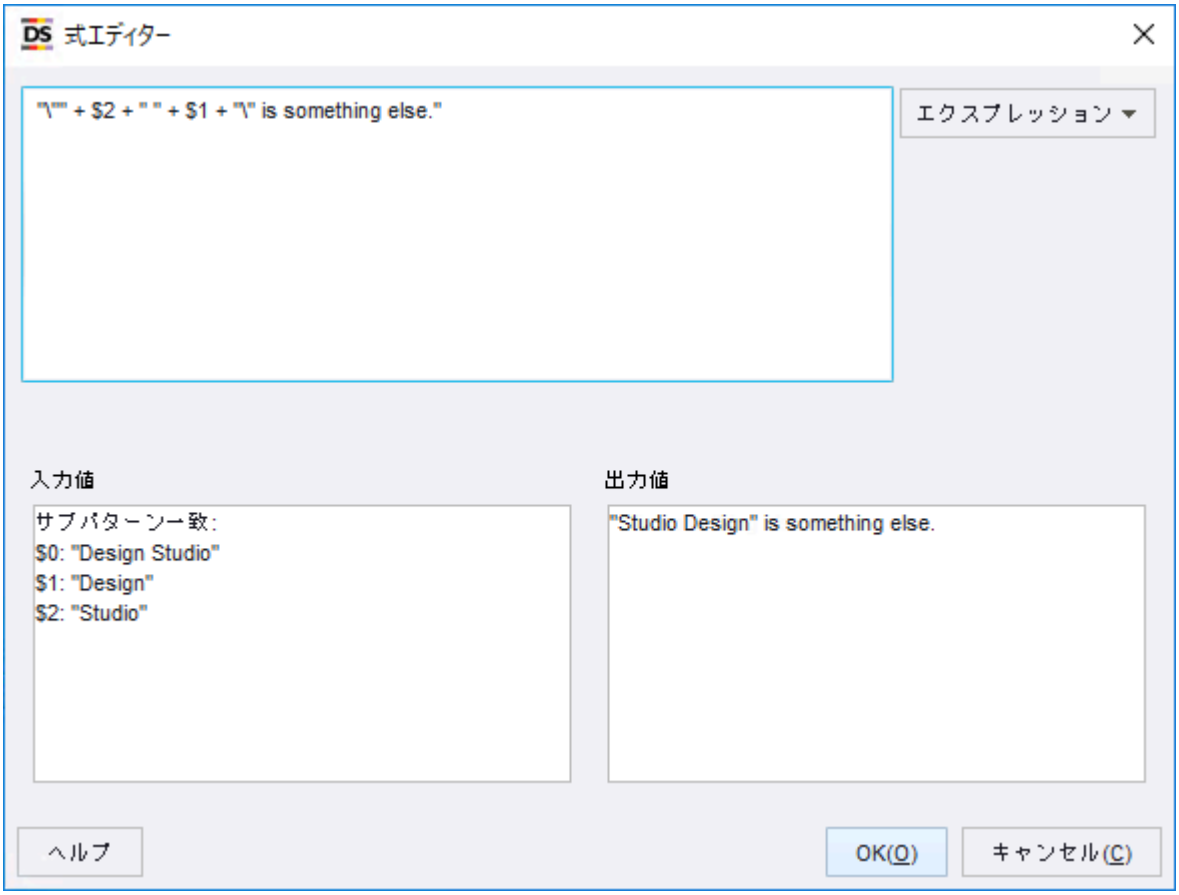
3. 設定 アイコンをクリックして、データ コンバータを設定します。
   [アドバンスド抽出設定] ウィンドウが表示されます。

   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。

   注 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。

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いつでも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、タイプの範囲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。

注 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した作業が行えますが、RoboServer などの Kofax RPA の他のアプリケーションは必ず、現在のプロジェクトとされる特定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 Developer's Guide』(Kofax RPA 開発者ガイド) をご覧ください。

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、複数のユーザーがプロジェクトを編集できます。ユーザーのシェア プロジェクトが、Management Console のプロジェクトと同期されない場合、シェア プロジェクト ビューで、プロジェクト内の各オブジェクトの状態を視覚化します。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 ライブラリを配布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 Developer's Guide』(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、シェア プロジェクト ビューに、アクセスできるすべての Management Console にデプロイされる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。

プロジェクトを **Management Console** にアップロード

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. 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、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 からプロジェクトをダウンロード

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. シェア プロジェクト ビュー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 [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 マイ プロジェクト ビューに表示され、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。

プロジェクトを同期

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 では、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提供されます。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(U)	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ています。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。	同期化	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 でロボットを設計しているときは、RoboServer から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。複数の異なる Management Console 間に同じ名前のデータベースが存在する場合があるため、Design Studio でデータベース マッピングを作成するときにはデータベースを区別する必要があることに注意してください。リストのデータベース名には、Management Console の名前が含まれます。

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. 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名前を入力します。この名前はマッピング ファイルの名前であり、データベースにアクセスするときの名前です。
   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。

Design Studio が適切なデータベース テーブルの設定方法をどのようにサポートしているかについては、データベース テーブルの生成と削除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構成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。

1. [ツール] メニューから [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。
2. データベースの名前を入力します。
3. データベース タイプと作成するテーブル タイプを定義します。
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. 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。

注 データベース テーブルが既に存在する場合、SQL を実行するときにデータベースからそのテーブルが破棄されます (DROP TABLE ステートメントが先頭にあるため)。

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 つのハウスホールド 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。

注 "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。例えば、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 3 つ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、データベース データ削除。検索と削除の操作はシンプルですが、データベース データ登録は、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。

ハウスホールド フィールド

値が挿入されると、ハウスホールド フィールドの 7 つ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
ExecutionId	この値を格納したロボット実行の実行 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

各ロボットの実行後 (ロボットがデータベースデータ登録を使用)、このロ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。

注 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、ロ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。

ロ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'

ロ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、実行中のロ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ロ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、ロ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。

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 と一緒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。

すべてのハウスホールド 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、次の点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、Oracle、Sybase を使う場合は、ここにある重要な 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。
- Sybase では、空の文字列は " "（スペースがひとつある文字列）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 3 つの事例はすべて、データがデータベースに格納された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。

口ボットの構造

口ボットの動作は人間の動作とよく似ています。つまり口ボットは、人がブラウザを使用してインターネット上のコンテンツを検索する場合とほぼ同じことを行います。ある人がコンテンツの検索を開始したとします。コンテンツを見つけたら、次にそれを読んで処理します。同様に、多くの口ボットの動作は、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、「コンテンツがどこにあるのか」ということに関係しています。ナビゲーションは、主にページ読込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入手する」ということに関係しています。抽出は、主に移動先のウェブ ページのコンテンツの選択、コピー、および正規化に関連する部分です。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。

一般的なロボットは、ページ読込またはクリック アクションが含まれる 1 つ以上のステップから開始され、ウェブ サイト上の意味のあるコンテンツに移動します。そして、各ステップに抽出アクションが含まれる 1 つ以上のステップの実行を続け、抽出された値を保存するステップまたは返すステップで終了します。

多くのロボットでは、抽出するコンテンツが複数のページに及ぶため、ナビゲーション部分と抽出部分が重複します。繰り返しとなりますが、これは人がコンテンツを検索するのとよく似ています。つまり多くの場合、必要なコンテンツを見つけるには複数のページにアクセスする必要があります。

大半のロボットには、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。ロボットによってさまざまなタスクがあるため、そのニーズも異なります。このため、Design Studio には、相当数のステップ アクションとデータ コンバータを盛り込みました。最も一般的に使用される基本のステップ アクションとデータ コンバータに慣れてから、さらに詳しい内容に触れてみてください。わずかなステップ アクションやデータ コンバータだけで、ほとんどのロボットを作成できることが経験的に証明されています。ですので、有効だと思うステップ アクションやデータ コンバータを見つけ出し、他のものを学ぶ必要性を感じるまではそれに磨きをかけてください。

適切に構造化されたロボットの記述

各ロボットはプログラムであるため、適切に構造化されたロボットの記述は重要です。構造化されていないロボットを記述することは、章や目次のない本を書くようなものです。次の理由から、適切に構造化されたロボットを記述することが重要になります。

- ロボットを文書化することが容易になる。
- ロボットのメンテナンスが容易になる。
- ロボットの操作が容易になる。

適切に構造化されたロボットを記述すると、副次的に Design Studio でロ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、通常はロボットをロボット ビューで編集すると、応答がより高速になります。

Web オートメーション ロボット

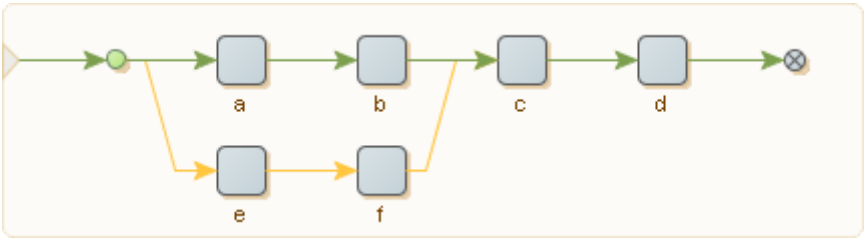
適切に構造化された Web オートメーション ロボットを記述するための主なツールに、スニペット ステップとグループ ステップの 2 つがあります。両ステップ タイプでは、ロボットの一部分を取り出し、説明的な名前を付けたり、単一のステップにまとめたりします。このため、ロボットの一部分の機能を詳細に知らなくても、ロボットの全体的な構造に注意を向けることが可能になります。この概念は、メソッド、関数、プロシージャなど、他のプログラミング言語の概念と似ています。

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 への口グイン」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

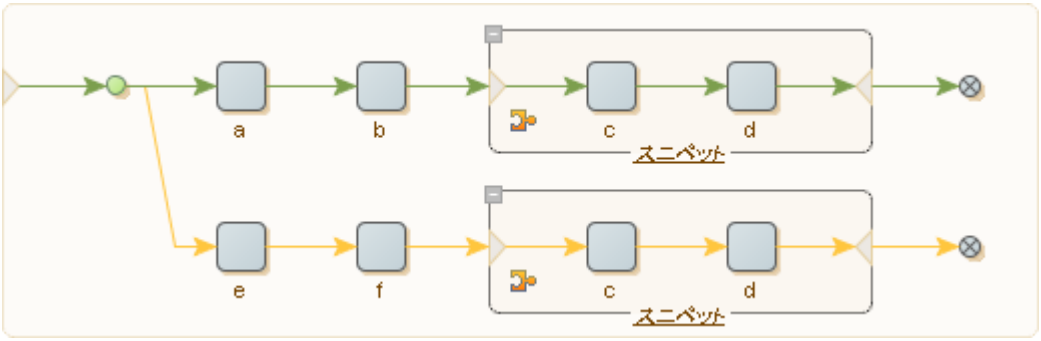
れたタスクをグループが実行できない原因になる可能性があります。グループ ステップを導入すると、名前でロボットのその部分の動作を表せるため、ロボットの文書化に役立ちます。

スニペットは主にロボット間の機能の共有に導入されていますが、単一のロボット内で使用してロ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、ロボットの各部分からの接続など、ロ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。

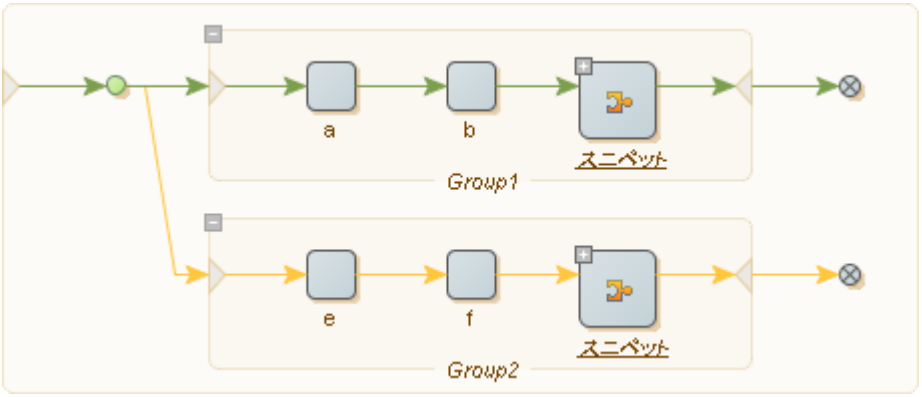
次のロボット構造では、接続を結合する代わりに、スニペットとグループが使用されています。



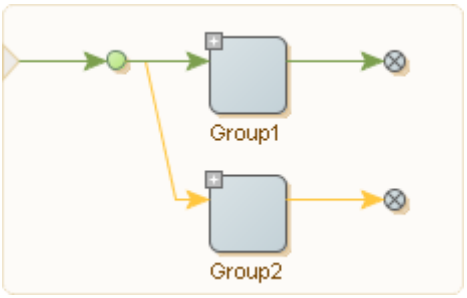
最後の 2 つのステップ c とステップ d は、ステップ a とステップ e で始まる 2 つの分岐で共有されます。実際には、はるかに大きなロボットを作成することがあり、このようにステップを共有する 3 つ以上の分岐が存在し、関連するステップがさらに離れている場合があります。その結果、ロボットの全体像を把握することが難しくなることがあります。ロボットの構造化を向上するために、まず次のようにステップ c とステップ d を含むスニペット ステップを導入できます。



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、次の簡単な口ボットを記述します。

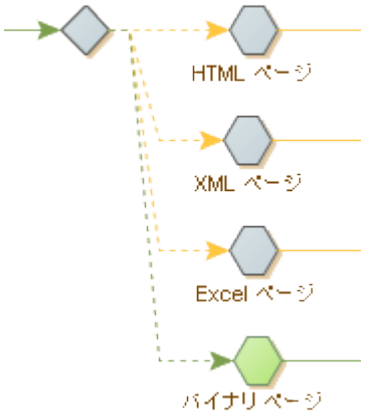


結果として、この口ボットは、2 つのタスクのうち 1 つを Group1、もう 1 つを Group2 として実行します。これらの 2 つのグループに説明的な名前を付けることにより、口ボットの構造が元の口ボットよりも論理的になります。

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。

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、ページ読込ステップの後、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、レイアウト関連のタグなど、時間の経過と共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与える可能性があるため、このプロパティの使用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

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。

例えば、タグ パスを "table" に、タグ属性プロパティを "align = center" に、 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を見つけます。

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、タグ ファインダー  をクリックし、タグ ファインダーで見つかったタグを確認します。

次のいずれかの方法でタグ ファインダーを設定します。

- タグ ファインダーを手動で設定するには、[ファインダー] タブで詳細を入力します。
- タグ ファインダーを自動的に設定するには、ページ ビューでタグを選択し、[ファインダーで選択ノードを設定]  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。
- 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。メニューから [タグを使用] を選択した場合は、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 アクションが選択され、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。
- 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 アクションを選択します。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。

1. Designer では、ノードを右クリックし、[ステップを挿入] > [アクション ステップ] を選択します。
2. リストからアクションを選択します。
3. [アクション] タブで、選択したアクションに基づいて属性を定義します。
   必須の属性は、警告 マークで示されます。

フォーム送信

フォーム送信は口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。

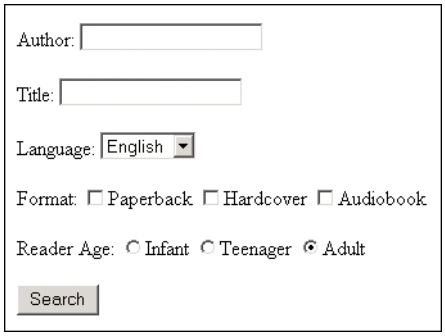
1. フォームの詳細を入力します。
   次のアクションを使用できます。
   - テキストを入力
   - パスワード入力 オプション選択
   - オプション選択
   - 複数オプション選択
   - チェックボックス設定
   - ラジオ ボタン選択

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。例えば、"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。例えば、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。例えば、"book_format" はチェック ボックス フィールドなので、"book_format" フィールドに値 "format_pb" と値 "format_hc" の両方を割り当て、ペーパーバックとハードカバーの両方が選択できます。

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。例えば、"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。例えば、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 マップを作成するのに使用しています。

JavaScript を使用するフォームもあります。例えば、<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、フォームをループ スルーことで必要な結果が得られます。

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、次のビデオをご覧になるか、以下のセクションをお読みください。

詳細については、フォームにおけるループ チュートリアルをご覧ください。

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。

   注 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。

1. 現在のステップで [フォーム送信] または [ループ フォーム] 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 [フォーム送信] 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

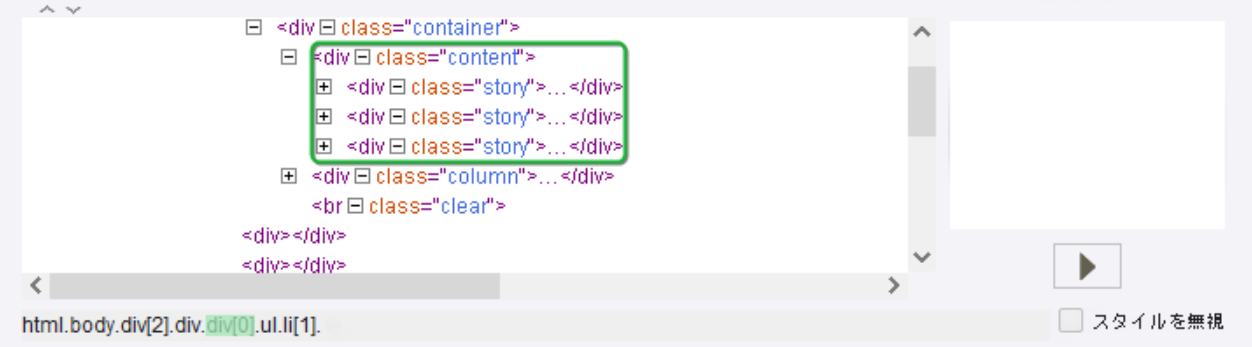
ロボットは、各エレメントで何らかのアクションを実行するために、ページ上のエレメントをループする必要が頻繁にあります。たとえば、検索の各結果から、あるいは表の各行から取得することができる特定のプロパティの抽出が必要な場合があります。以下のトピックでは、その方法について説明します。

- 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繰り返しステップでは、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。



注 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、繰り返し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。

見つかったタグは 3 つの div タグを含んでいるが、すべての div タグが同じクラスを持っている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

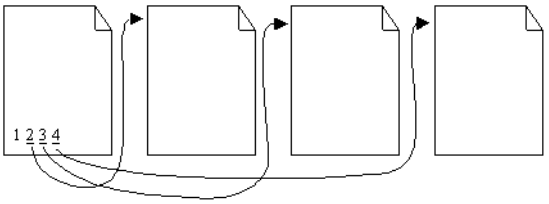
ロボットは頻繁にページをループする必要があります。たとえば、検索要求の結果を表示する多くのウェブサイトでは、各ページに 20 件といったように、複数のページにわたる検索結果を表示します。検索結果を取得するには、ページをループして、一度に 1 ページを処理する必要があります。以下のトピックでは、この方法について説明します。

- 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

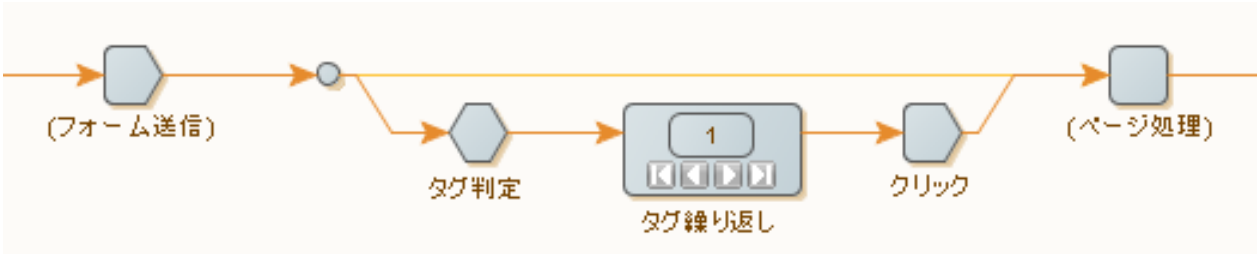
最初のページにある他のすべてのページへのリンク

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。

注 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。

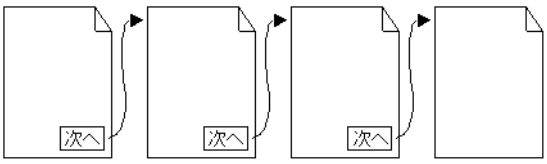
残りのページは、2 番目の分岐のタグ繰り返しアクションを使用してループします。

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読込します。
  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。



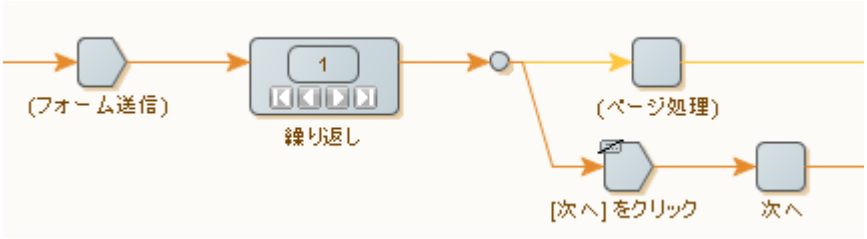
繰り返しアクションを使用して、こうしたページをループします。このアクションは、「次へ」という名前の別のアクションにより、指定されているページをループします。詳細については、次を繰り返しチュートリアルを参照してください。

「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。

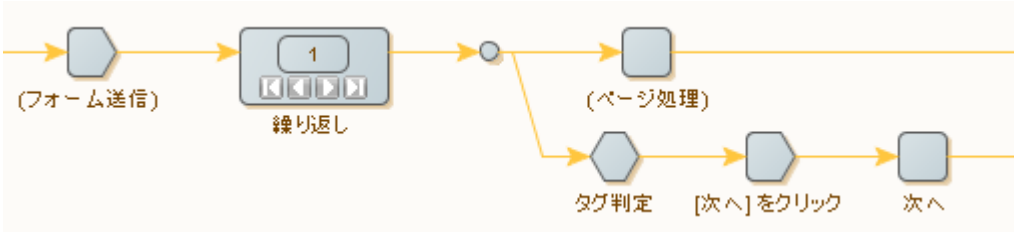
1. 最初のページで、「繰り返し」ステップを追加します。
2. 必要に応じて、追加のアクションを挿入します。

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。

   注 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、次のページ読込します。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、[**Then**] 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、エラーの処理をご覧ください。
   以下の口ボットの例は、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、このページ読込して、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)。

注 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を使用すると、9.6 よりも前のバージョンの場合よりも高速かつ信頼性の高いロボットを簡単に作成できます。

注 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。待機基準を使用するには、デフォルト オプション ダイアログ ボックスの [JavaScript 実行] タブにある [**JavaScript** の実行] を選択します。

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込) の処理がロボットを続行するために、十分に完了したタイミングを正確に特定できるようになりました。

ビルトイン 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、ページでエレメント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るようになりました。

次のステップ アクション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
- 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
- ページ読込
- 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
- HTTP 通信
- 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

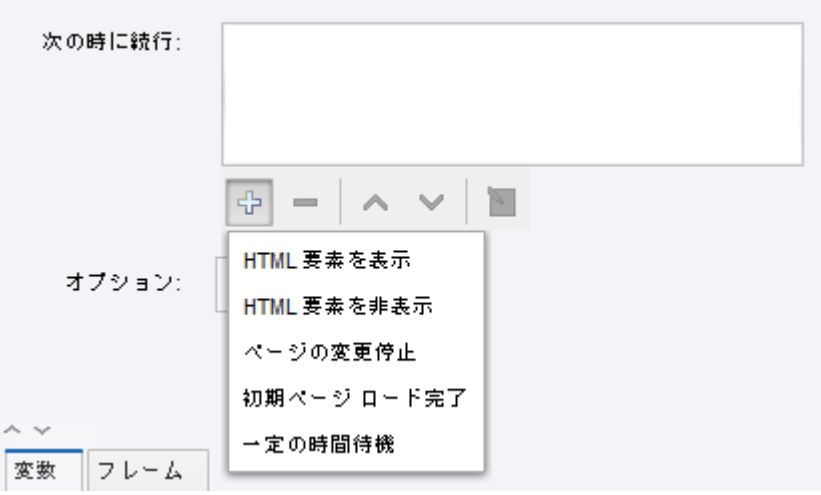
Kofax RPA バージョン 9.6 以降で作成されたすべての口ボットでは、デフォルト待機が [ページは 500 ミリ秒の間、変更を停止します] に設定されています。この設定は、[口ボットの設定] ウィンドウの [詳細] タブに表示されます。待機基準が有効化されたすべてのステップには、デフォルトの [ページを 500 ms 変更しない] 待機基準と有効化された [再開] ボタンがあり、[結果ビュー] に表示されます。この待機基準は [待機] ビューに常に灰色表示され、また常に満たされています。その他のすべてのブラウザ ステップには、デフォルトの [初期ページ読込完了] 待機基準と無効化された [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。

デフォルトのブラウザ エンジン ロボットをクラシック ブラウザ エンジン ロボットに切り替える場合は、次のルールが適用されます。

- デフォルトのブラウザ ロボットのステップに待機基準が指定されている場合、クラシック ブラウザ ロボットでは、このステップが [タイマー イベントをリアルタイムで待機=true] に設定されます。
- デフォルトのブラウザ ロボットのステップに、[タイマー イベントをリアルタイムで待機=true] に加えて、レガシー タイミングがある場合、クラシック ブラウザ ロボットでは、このステップの [タイマー イベントの最大待機時間] が、レガシー タイミングで指定されている時間に設定されます。
- 同じロボットをデフォルトのブラウザ ロボットに切り替える場合、待機基準は復元されません。

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、待機基準の構成ウィンドウの [ステップを再実行] 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。

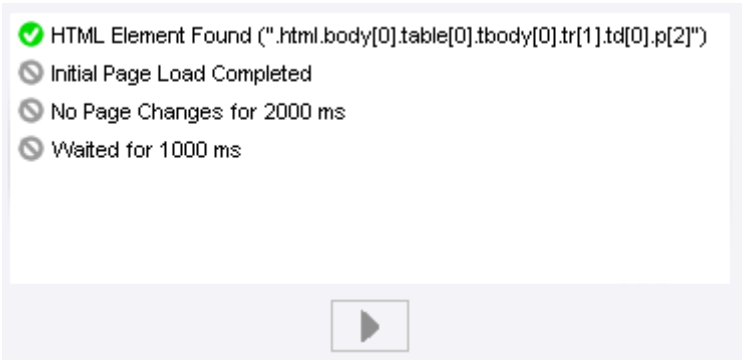
複数の待機基準をステップに追加することができます。待機基準がいくつかある場合、いずれかの待機基準が満たされたときに実行が停止します。同一のロードで表示される 2 つの HTML エレメントを待機しているときや、メイン フレームのエレメントを待機しているときなど、[初期ページ読込完了] が設定されている場合は、複数の満たされた待機基準がある可能性があります。

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、

ページ読込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 エ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。

注 [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

[初期ページ読込完了] 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

待機基準の構成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化されています。待機基準を無効化するには、[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。

注 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、待機基準が既に満たされているが、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。この基準の設定は、[検知された要素は次である必要がある] グループの 2 つのエレメントを除いて、「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。

- 有効：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、`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、2 つのオプションが含まれる追加プロパティ [最初の要素の検知] は異なります。

- [ページ内に要素が見つかりました] オプションを選択した場合、ロボットは、ページにエレメントが表示されるまで待機します。この動作は、DOM ツリーにエレメントが表示されなくなるのをロボットが待機した後にのみ発生します。
- [一定の時間待機] オプションを選択した場合、ロボットは、指定された時間待機し、その後、DOM にエレメントが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、ページ読込の最初から DOM にエレメントが表示されていなかったとしても、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。

初期ページ読込完了

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込が完了したときに満たされます。

注 この基準には [満たされたときすべてのロードを停止] 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 以降での古いロボット

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] 設定をご覧ください。

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際は、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 への既存のロ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。

ページ読込の修正

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、[ページ読込完了] 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、前のページ読込が中止された場合に発生することがあります。ページ読込を修正するには、[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 アクションがあります：

- 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。

さまざまなバイナリ データ フォーマット (PDF や Flash など) からデータを抽出する方法として、2 つのアクションがあります。この 2 つのアクションは、データを抽出し、ロボットがデータへ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 ページを生成するという点で、前述のアクションとは異なります。これらのアクションは、実際にデータを抽出する前の初期ステップで用いられ、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。

- **PDF** からのテキスト抽出アクションは、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ために使用します。
- **Flash** からの抽出アクションは、見つかったタグの Flash オブジェクトからデータを抽出するために使用します。

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 データを抽出します。

[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。画像、PDF、およびセッション タイプではデータをプレビューできることを除いて、これらのすべてのタイプは同等です。

ページ ビューでコンテキスト メニューを使用

1. 抽出元のテキストまたはタグ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 アクション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" への正規化など、国コードを自然言語の説明に正規化する場合です。

- 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。
   詳細については、簡単な日付抽出と複雑な日付抽出のチュートリアルを参照してください。

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。

HTML テーブルからコンテンツを抽出する

HTML テーブルのコンテンツと構造が不規則である場合があります。このトピックで説明するように、Design Studio はこうした不規則性に対処するように設計されています。

注 このトピックで説明するテクニックは、テーブル コンテンツと構造の不規則性への対応に限定されるものではありません。このテクニックを使用して、その他のタグの不規則性を処理することもできます。

1. ロボットに行のループ ステップを挿入します。
2. <table> タグの <tbody> タグにある <tr> タグをループするタグ繰り返しアクションで行のループ ステップを設定します。
3. ステップを追加して、テーブルの行のセルから列方向にコンテンツを抽出していきます。
   例

注 テーブルのコンテンツがコンテンツと構造の両方において完全に規則的である場合は、HTML からコンテンツを抽出する方法の説明どおりにコンテンツを抽出できます。

テーブル コンテンツの不規則性の処理

同じテーブル列のセル コンテンツの書式が異なる場合があります。たとえば、セルが空であったり、セルに "Bob" (名) や "Bob Smith" (名と姓) が含まれる場合があります。

1. コンテンツの不規則性に対処するには、抽出ステップで、[**If Then**] データ コンバータを追加します。
2. 各書式と照合させるために [**If**] および [**Else If**] プロパティを設定します。

   対応する [**Then**] プロパティにより、照合するサブパターンが抽出されます。

   注 抽出アクションでは 1 つの値のみを抽出できます。2 つの値 (名と姓) が含まれる "Bob Smith" の場合、2 つのステップ (1 つは名を、もう 1 つは姓を抽出する) を作成する必要があります。両方のステップには If Then データ コンバータを使用する抽出アクションが含まれるため、最初のステップで名 (ある場合) が、2 番目のステップで姓 (ある場合) が抽出されます。

テーブル構造の不規則性の処理

テーブル行では、含まれるセルの数が異なる場合があります。こうした不規則性に対処する一般的な方法は、各テーブル行の書式をテストすることです。たとえば、特定の数のセルが含まれる行または特定のテキストが含まれる行のみを考慮することが必要な場合があります。

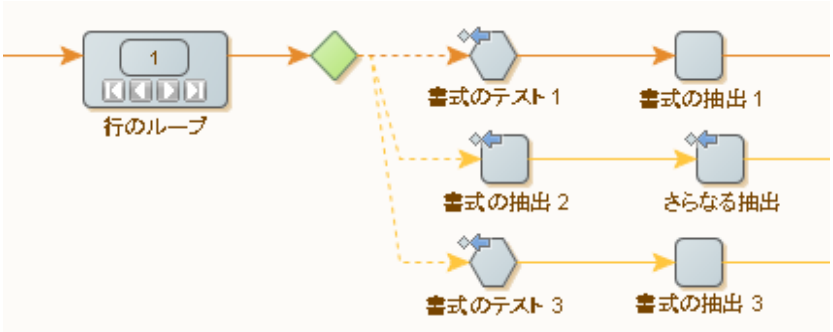
1. 各タグ繰り返しステップに、[トライ] ステップを続けます。
2. テーブル行をループするようにトライ ステップを設定します。

   トライ ステップの各分岐で 1 つの書式を処理します。この処理は、たとえば、パターンとして記述されたある書式に一致するすべての行を受け入れるタグ判定アクションなど、[次の代替手段を試行] エラー処理による条件ステップで各分岐を開始することによって実行されます。

3. 条件ステップに、条件アクションで受け入れられる書式を想定している 1 つ以上の抽出ステップを続けます。

   場合によっては、条件ステップと抽出を組み合わせることにより、書式の抽出を試行し、失敗した場合は次の抽出を実行するように設定することができます。

   次のロボットは両方のアプローチを使用しています。



   2 番目の書式の抽出は 2 ステップ プロセスであることに注意してください。[次の代替手段を試行] エラー処理は両方のステップに設定されているため、2 つのステップのいずれかが失敗すると、3 番目の分岐が試行されます。これは、2 番目の分岐のかなり複雑な条件を表しています。

   このロボットが実行されると、各分岐が、いずれかの分岐が成功するまで順番に実行されます。すなわち、後の分岐の条件では、前の分岐から条件を繰り返す必要がありません。失敗したことが明らかであるためです。

   注 条件ステップの後で分岐をする必要はありません。2 つ以上の分岐が抽出ステップを共有している場合、それぞれ異なるステップの後にある分岐は結合することをお勧めします。

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にある場合、ページ読込アクションでファイルの URL を指定することにより、またはクリック アクションでファイルへのリンクをクリックすることにより、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。

   注 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。

   注 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。

変数から Excel ページ読込

Design Studio でページ読込する最も一般的な方法はページ読込 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。

重要 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.config` 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

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**]** > **[**テキスト抽出**]** (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 バージョンをロボットに戻して、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、ロ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。ロ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 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 タイプで行うことと同じです。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

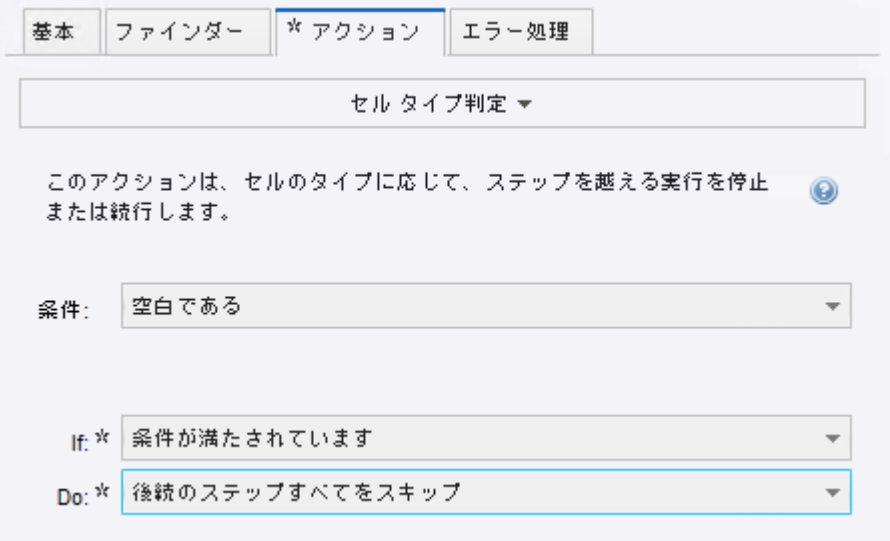
Excel 関数 ISERROR に相当。

式

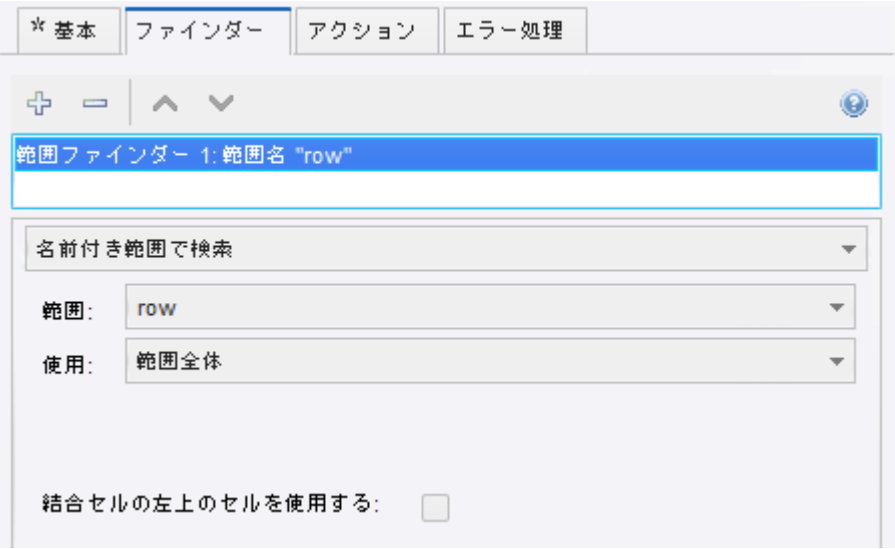
Excel 関数 ISFORMULA に相当。

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。



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、リファレンス ドキュメント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ロボットを使用] 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] > [テーブルの行をループ] > [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。
   詳細については、Excel チュートリアルを参照してください。

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ロ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、ロ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。

このロ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込 ステップです。ロ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。ロボットは、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 `Then` が [次のイテレーション] 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 エラー処理は、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、列ヘッダーが指定された構造を持っていることをテストします。次の図は、ロボットに追加された エ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	Test Results				
2		Test1	Test2	Test3	
3	Alice	12	7	9	
4	Bob	Missed		4	
5	Jane	11	8	7	
6	John	12	Missed		
7	Zach	10	Missed	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、[名前付き範囲と同じ] を選択し、[高さは名前付き範囲の一番下まで] という説明を入力します。
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。

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、ほとんどの場合、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 アクションで変更した 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 アクション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 アクションに別の 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 アクションを使用します。

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 アクションの動作は、ページ読込 ステップ アクションによく似ています。

   変数が既に開いている場合、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 アクションの動作は、ページ読込 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 アクションに似ています。

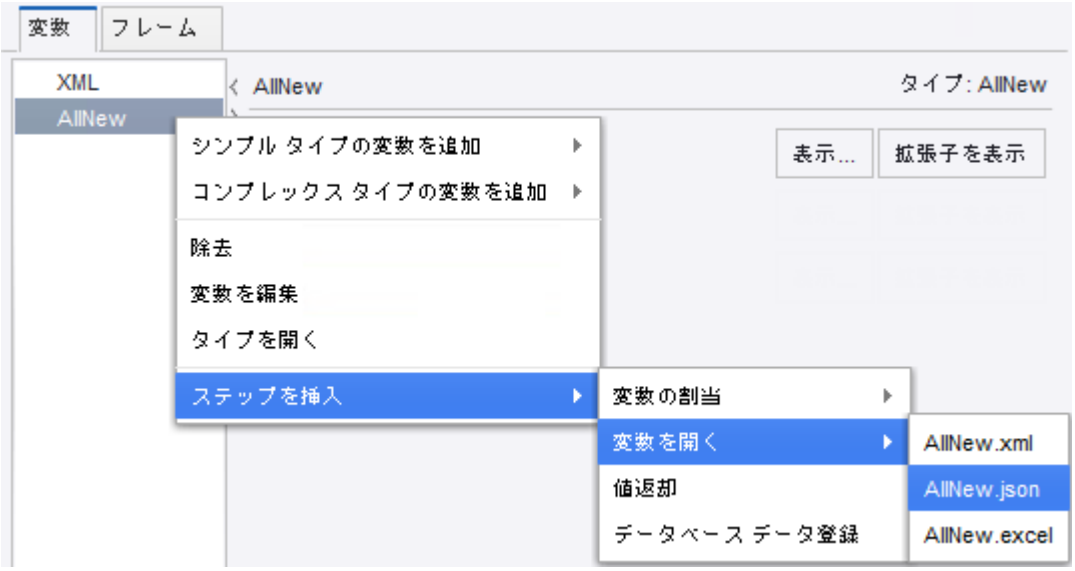
   このステップ アクションは変数を開くステップ アクションと呼ばれるものの、変数の属性もウィンドウで開かれることがあるタイプの場合には、変数の属性についても機能します。

   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 アクションを挿入できます。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、構成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 アクションを挿入します。

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 アクションがあります。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 アクションを使用して、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 アクションは次のとおりです。

- 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 アクションは次のとおりです。

- 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 アクション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 アクションでは、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の両方で開かれる 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 アクションを利用できます。

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 (") を付けなければならず、", \ 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 \ を用いて工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
```

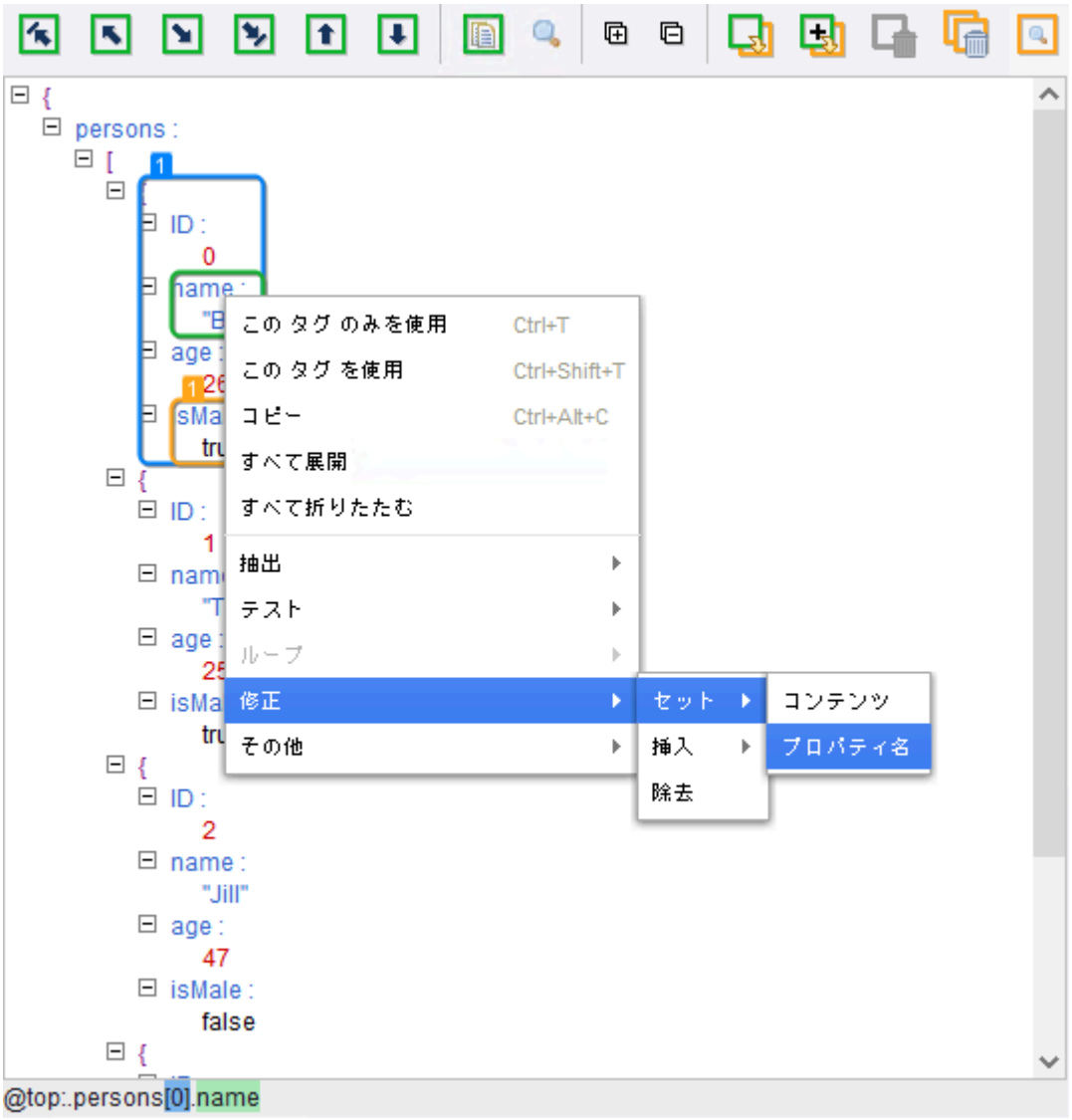
厳密に言えば、すべての JSON 値がこの MIME タイプに対して有効というわけではありません。JSON の受け入れや返却を行うサービスの開発者は、さらに自由な JSON 値の受け入れや返却を行う可能性もあります。Kofax RPA では、JSON に対して、こうしたより自由なアプローチを採るという選択をしました。そのために、JSON 変数には JSON 値が含まれ、JSON 表示にはその値が表示されます。

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。そうでない場合は、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、[ページ読込] 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 アクションなど)。

JSON とステップ アクション

多くのステップ アクションは 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 でなければなりません (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 アクションは、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 アクション 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 アクションが抽出できます。

- JSON 抽出。このステップ アクションは、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 アクションは、プロパティの名前を抽出します。

以下の 2 つのステップ アクションは、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 アクションは、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 アクションは、JSON 配列の各 JSON 値をループします。

両方のステップ アクションは、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)。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が失敗するような値に変わる可能性があるため、変数でイテレートした場合にはこの設定は全体に実行されません。これは、イテレーションされたリストからアイテムが除去された場合などに発生します。

以下の 4 つのステップ アクションで 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 アクション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 アクションが JSON で動作します。

- JSON 判定。このステップ アクションは、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 アクションは、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 アクション内のコンバータ 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
5	OUTPUT = eval(INPUT +「 + 3")	8
[1,2,3]	OUTPUT = INPUT[0]	[

INPUT 値	JavaScript (OUTPUT =)	結果 (OUTPUT 値)
[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、以下の点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。

エラーの処理

ロ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 アクション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。ロ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、ロ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ロボットを設定できます。

注 このヘルプ システムで説明しているエラー処理動作は、ロ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 モードでの実行では適用されません。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 例外とログ 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. ロ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 ロボットが 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 を使用している限り、呼び出し元側で例外が発生し

ます。ロボットがその他の方法で実行された場合の詳細については、リファレンスのエラー処理 を参照してください。

b. エラーをログに記録するには、[エラーとしてログ記録] を選択します。ロボットが 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。

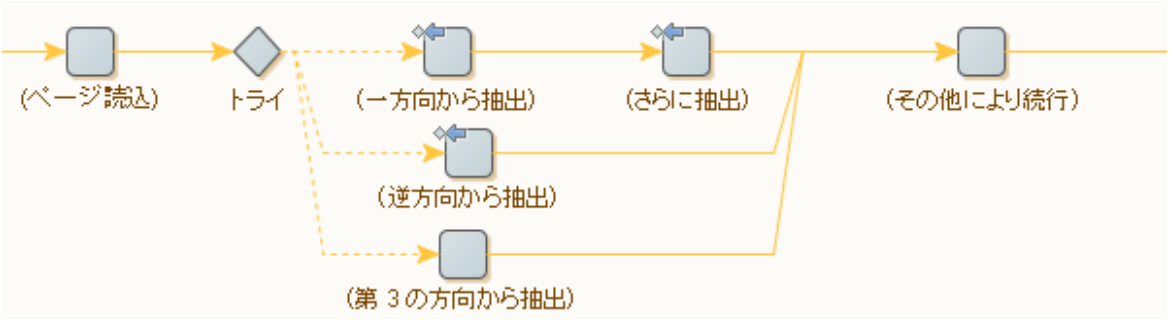
注 チェック ボックスのオン/オフで選択します。チェック ボックスがアスタリスク * でマークされている場合は、デフォルト以外の値に設定されていることを示します。詳細については、アスタリスクを除去して、デフォルト値に戻す方法を説明しているデフォルトからの変更を表示を参照してください。デフォルト値が適用された場合 (つまり、アスタリスクがない場合)、デフォルト値は、エラーを処理する方法によって異なることに注意してください。

2. **[Then]** 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、リファレンス ドキュメント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。

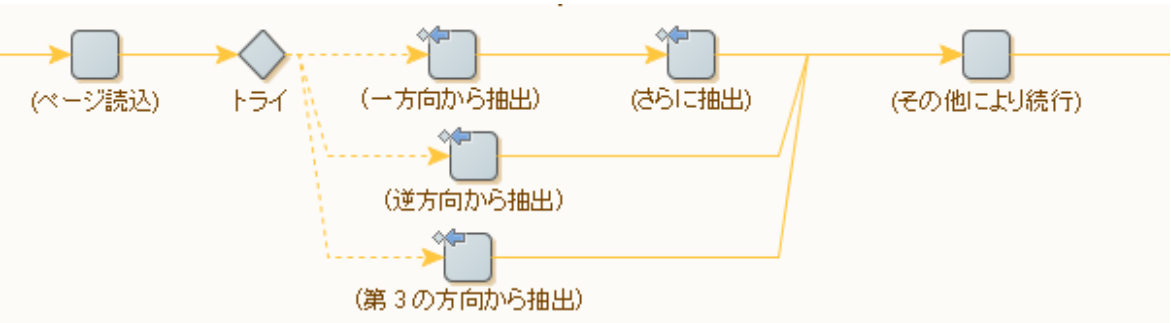
Web ページの一部にさまざまな構造とレイアウトがあるものの、その部分が 3 つのケースのいずれかに該当する場合を仮定します。各ケースには、抽出する情報があります。これは、1 度に 1 つのケースの抽出を試行することによって実行できます。失敗した場合には次のケースを試行し、3 番目のケースまで試行します。これで成功することが想定できます。



抽出ステップの (次の代替手段を試行) 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 (例えば、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。次回にロボットを実行したときには 3 つの戦略のどれも成功しないほど Web サイトが変更された可能性があるため、ロボットは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、Try-Catch 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「次の代替手段を試行」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

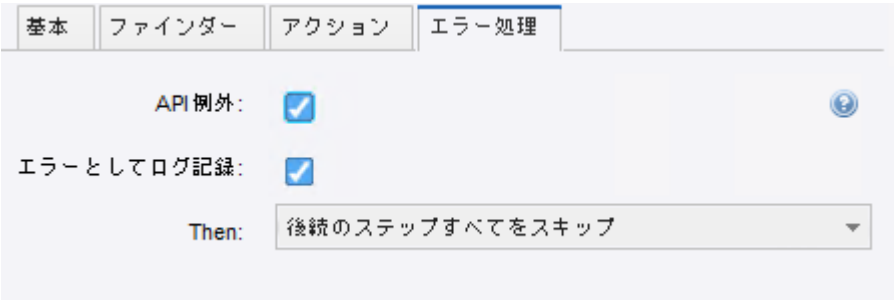
多くの場合、ロボットは Web ページで任意のエレメントの処理が必要になります。つまり、エレメントが存在する場合には処理が必要ですが (データの抽出など)、存在しない場合には、エレメントの処理をスキップできます。エレメントが存在しなくてもエラーではなく、想定内の状況です。これは、ロボットでは次のように表すことができます。ステップ A はエレメントが存在するかどうかテストします。ステップ B と C は続けて処理を実行しますが、これはステップ A の成功に依存します。

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)、[次の代替手段を試行] () 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 [エラーとして口グ記録] 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 [次の代替手段を試行] 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。

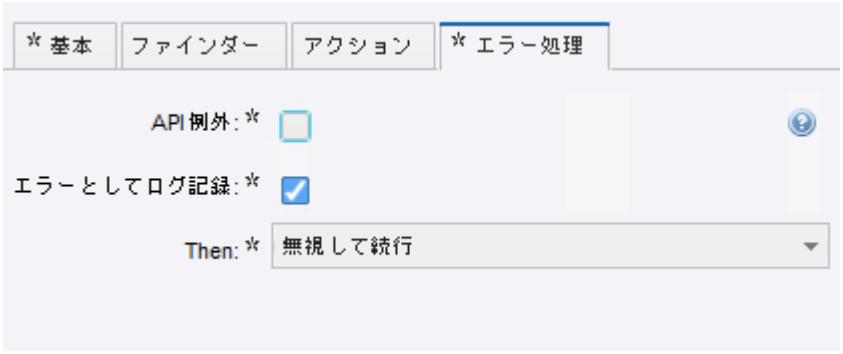
無視して続行

何らかの条件が満たされた場合、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したい場合があります。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、[次の代替手段を試行] ( ) 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。

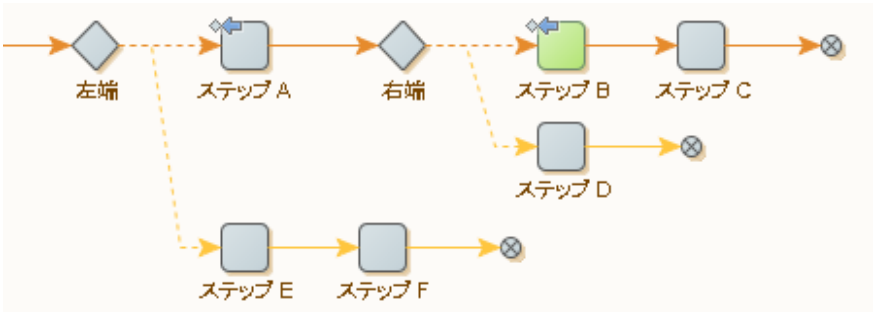
At ターゲット

「次の代替手段を試行」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。そうでない場合、次の代替手段を試行の「次の代替手段を試行」の部分は意味をなさなくなります。

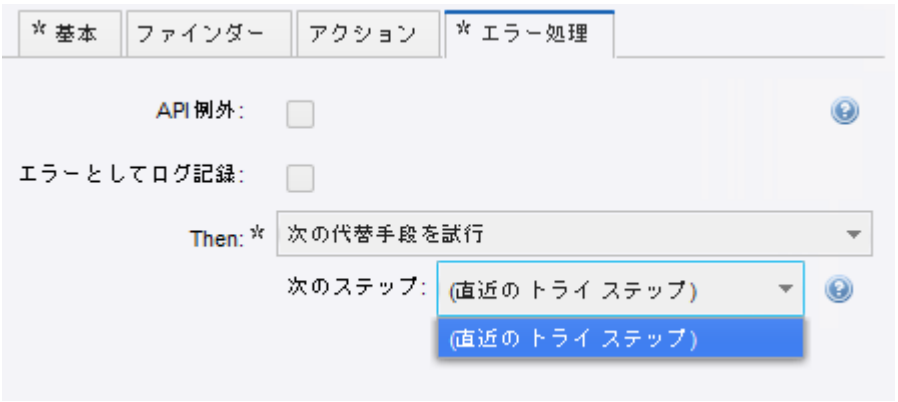
現在の実行パスに複数のトライ ステップがある口ボットについて考えます。この状況で、口ボットはこれらのトライ ステップのそれぞれに関連する「現在の分岐」の実行中で、トライ ステップごとに「次の分岐」は異なります。たとえば次の口ボットでは、ステップ B でエラーが発生した場合、「次の代替手段を試行」は次のいずれかのアプローチを使用して実行を続行します。

- 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、Try-Catch に記載されているように活用できます。

この例では「次の代替手段を試行」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」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。

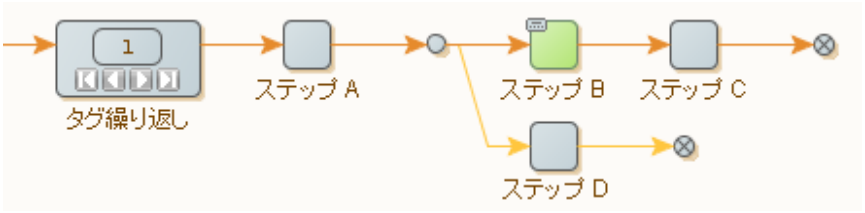
ルーピング

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。

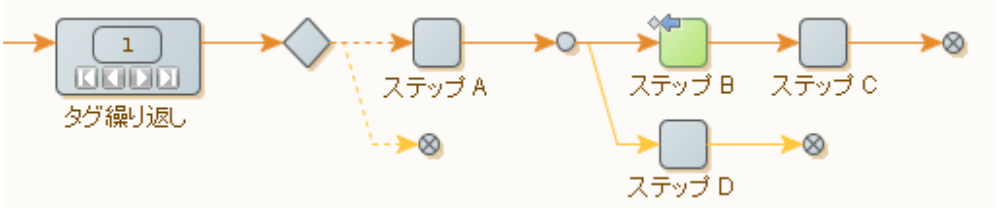
次のイテレーション

後続の口ボット、ステップ B には、[次のイテレーション] 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、[次の代替手段を試行] 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。



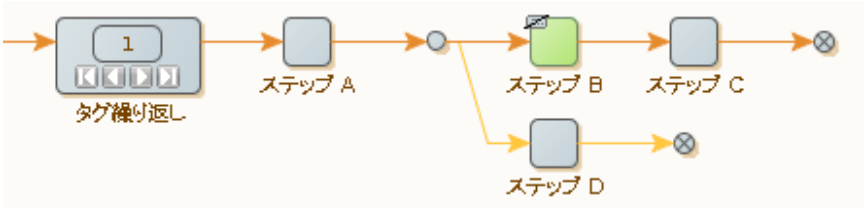
他のトライ ステップが干渉する恐れがあるため、この変換には一般的に At ターゲットの使用が必要であることに注意してください。

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。

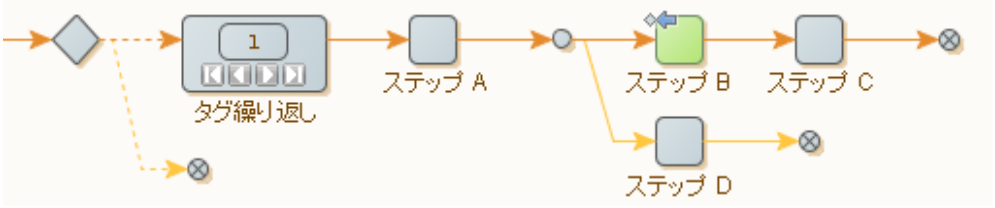
[次のイテレーション] は、[繰り返し - 次] のループでは機能しません。これら 2 つの場合において、ブラウザの状態については、「次」という言葉がまったく別の意味を持ちます。

ループ終了

[次のイテレーション] 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] とは異なり、[ループ終了] は、[繰り返し - 次] のループでは機能しません。

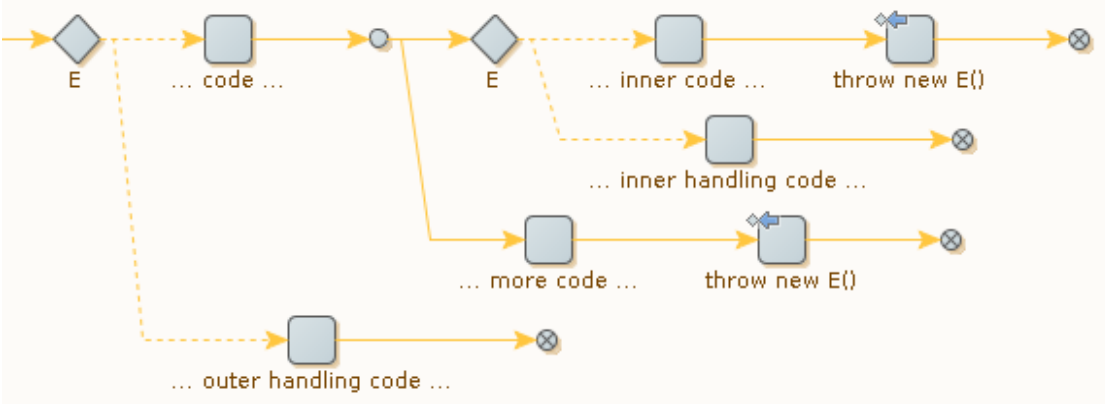
Try-Catch

[次の代替手段を試行] 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 ...
 }
```

ロボットでは、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 [次の代替手段を試行] 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。ステップでカスタム エラー処理をマークする必要がない場合、Design Studio の [オプション] メニューでこの機能を無効にできます。

詳細については、ロボット エディターをご覧ください。

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。
   d. 新しいスニペット ステップを挿入して選択したステップを置き換えます。
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、[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。
   d. 新しいスニペット ステップを挿入して選択したグループ ステップを置き換えます。

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。

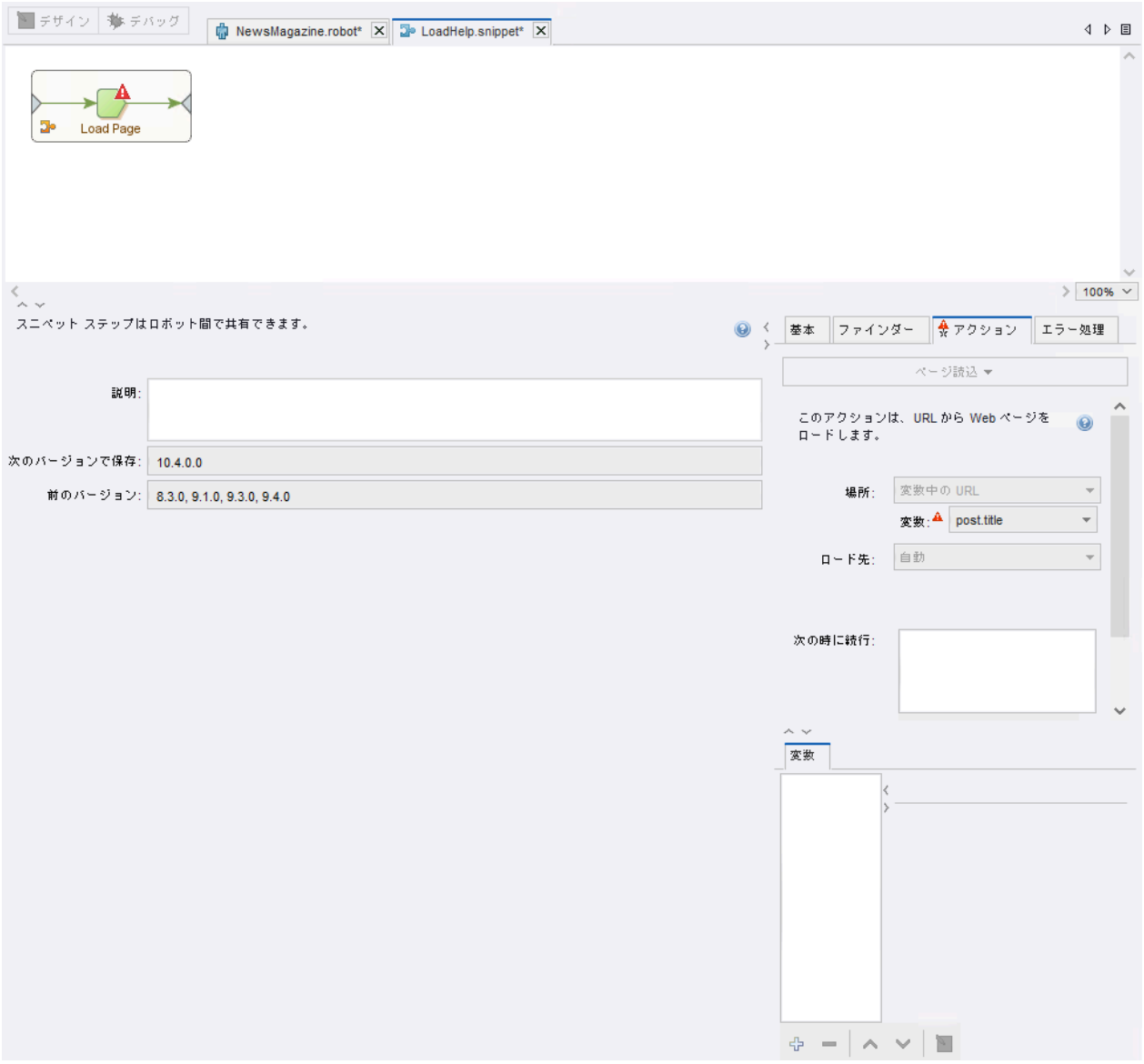
      注 このエディター内でスニペットのコンテンツ (スニペット内のステップ) を編集することはできません。

   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。

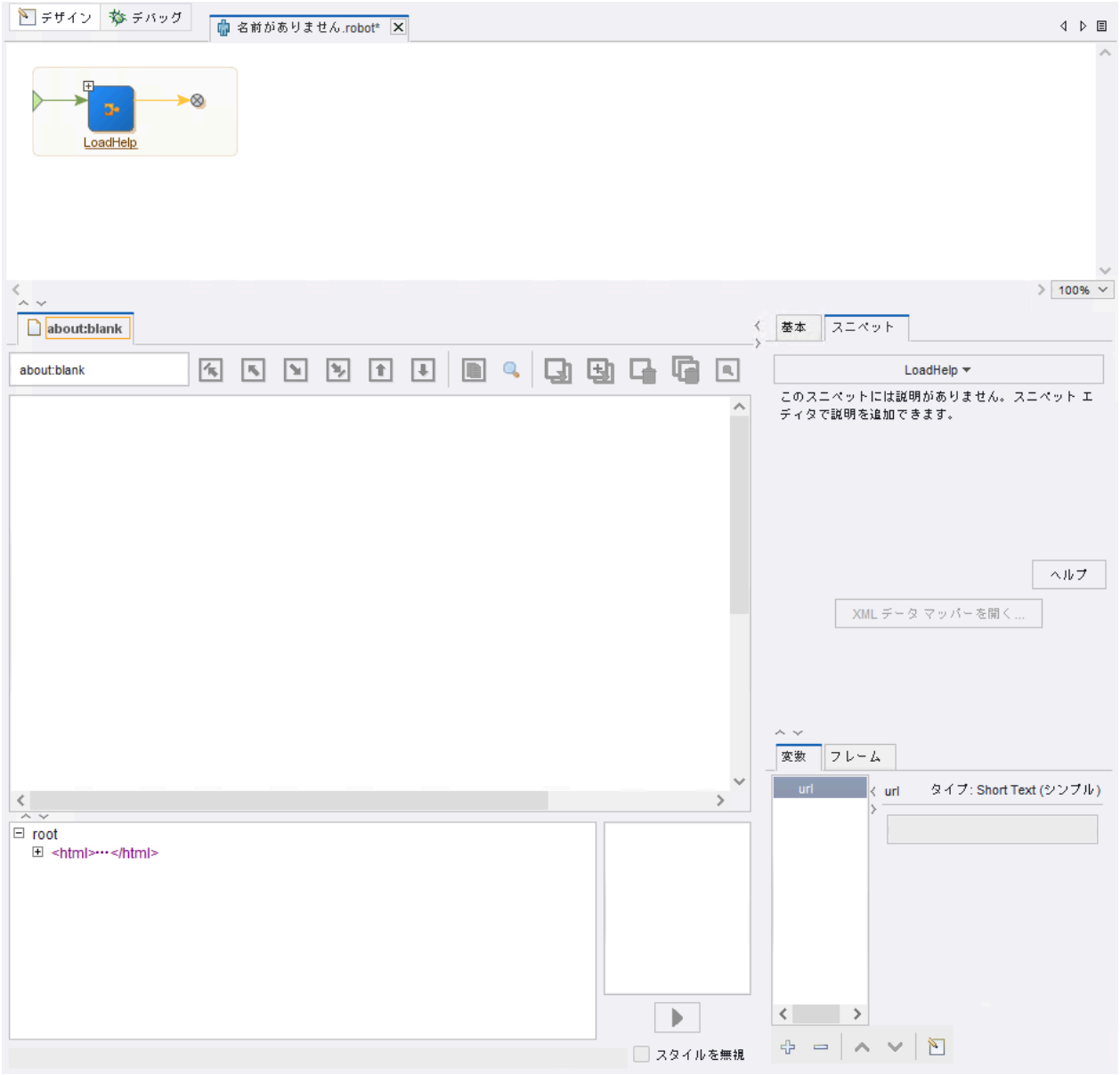
スニペットは自身の変数を定義することができます。スニペット自身のエディターでスニペットを開き、スニペットで変数を定義します。スニペットに既にステップが含まれる場合、スニペットが編集されたロボットに存在していた変数を使用すると、ステップは、赤のフラグでマークされます。



上記の図は、自身のエディター内のスニペットを示しています。このスニペットは、スニペットで定義されていない変数を使用しています。

右下のアクティブな変数エディターはロボットの場合と同じであることに注意してください。

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。



上記の図は、変数 "url" を定義するスニペットを使用するロボットを示しています。

スニペットからインポートされた変数が変数リストでマークされていることに注意してください。

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。

ロバストなロボットの作成

Web サイトは、予告なしに変更されることが頻繁にあります。このような変更を考慮しない場合、ロボットがタスクの実行に失敗する可能性があります。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。ロボットが対処可能で、正しく動作できる変更が多いほど、ロバストということになります。

しかし、ロボットをロバストにするにはコストもかかります。ロバストなロボットの記述は、脆弱なロボットの記述に比べ、手間と時間がかかります。これには、対象の Web サイトの分析や、登録フォームが間違って入力された場合などのさまざまな状況においてロボットがどのように対応するかといった内容の理解が必要となります。また、ロバストなロボットの作成には、Web サイトのロジックをリバースエンジニアリングすることも必要になります。通常、それには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。Design Studio 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ロボットが返すようにすることを意味します。入力変数を取るロボットでは、成功よりも頻繁に失敗が発生することあります。Design Studio 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。

ロ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。HTML からのコンテンツ抽出、HTML テーブルからのコンテンツ抽出、エラー処理、およびタグ ファインダーの使用。

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。

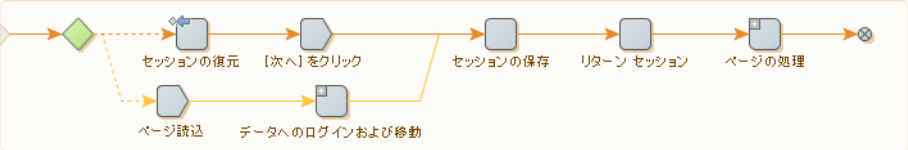
ロボットの実行が頻繁過ぎてその応答時間を大幅に短くする必要がある場合、ロボットの適切なセッションの作成には必要以上に多くの時間がかかります。しかし、セッションの取得が行われロボットおよびロボットの実行間で共有されると、時間が大幅に節約されることになります。

セッションの再利用には以下の 2 つのステップ アクションを使用します。

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

ウェブ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) を返すようにしたいとします。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したいと考え、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既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、いつでも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。

ロボット設定

1. Design Studio ツールバーで [ロボット設定] を選択します。[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。

   [ロボットの設定] ウィンドウが表示されます。
2. [基本] タブで [設定] をクリックします。
3. ロボットのすべてのステップ アクションに適用されるデフォルト オプションを設定します。

   必要に応じて、ステップ アクションでグローバル オプションを上書きできます。
4. [**OK**] をクリックします。
5. [ロボット コメント] フィールドにオプションのコメントを入力します。

   このコメントは、ロボットがどのように動作するか、あるいはロボットを編集するときに考慮する必要があることなどを文書化する場合に便利です。
6. [ロボット タグ] フィールドでロボットのタグを追加できます。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。
7. [手動の処理時間 (分)] フィールドで、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。
8. [詳細] タブで、ロボットが実行するすべてのページおよびデータのローディングに使用されるオプションのプロキシ サーバーを指定できます。

   このプロパティは頻繁に使用されません。通常、Design Studio 設定の下で 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。詳細については、プロキシ サーバーをご覧ください。特定のロボット用に指定したプロキシ サーバーは、その他の方法で指定したプロキシ サーバーを上書きします。また、プロキシ切替アクションを使用して、ロボットの実行中にプロキシ サービスを変更することができます。

デフォルトのロボット設定から変更を表示

ロ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、ページ読込 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。


デフォルト オプション
すべてのローディング
ページ ロード中
URL フィルタ
JavaScript 実行
プラグイン
レガシー
切り替え
クレデンシャル:
スタンダード
ユーザー名:
パスワード:
クライアント証明書:
自動
SSL/TLS:
SSL Hello を使用、SSL 3.0 のみ受け入れ (非セキュア)
SSL 証明書を検証:
認証方法:
NTLM
HTTP ユーザー エージェント:
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.134 Safar...
言語:
英語 (アメリカ合衆国) (en_US)
画面サイズ:
1280 x 1024
この URL から参照:
Cookie を有効化:
HTTP キャッシュ:
有効
最大試行回数:
1
試行間隔 (秒):
5.0
試行タイムアウト (秒):
120.0
送信する追加ヘッダー:
リストから
受信したステータス コードをここに保存:
(なし)
受信したヘッダーをここに保存:
(なし)
ヘルプ
OK(O)
キャンセル(C)

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限り、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、加えられた変更があればそれら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。

他のブラウザ エンジンにロボットを切り替える

Kofax RPA ではブラウザを使用して、Web サイトまたは Web アプリケーションと通信します。Kofax RPA には現在、2 つの異なるブラウザが搭載され、それぞれ別の目的向けに最適化されています。クラシック エンジンはレガシー Web アプリケーション向け、デフォルト (WebKit) エンジンは現在の標準的な Web アプリケーション向けです。ただし、これらのブラウザは、内部アプリケーションまたはインターネット Web サイトと互換性がない場合があります。いずれかのブラウザを使用しているときに問題が発生した場合、以下の手順を使用して、Design Studio で他のブラウザ タイプにロボットを切り替えることができます。

クラシック ブラウザにロボットを切り替え

1. ツリー ビューでロボットを右クリックし、[切り替える] を選択します。
2. ロボットやスニペットなどの切り替えるファイルを選択し、[次へ] をクリックします。

3. ファイルをバックアップするかどうかを指定して、バックアップ方法を選択し、[バックアップ] ステップで [終了] をクリックします。
   Kofax RPA がロボットを切り替えるすると、ロボット アイコンの色が変わります。

   注 レガシーの待機基準を使用するステップがある WebKit ロボットを切り替えるときは、この待機基準が [タイマー イベントをリアルタイムで待機] と [タイム アウトの最大待機時間] に変換されます。

デフォルト ブラウザにロボットを切り替え

1. ツリー ビューでロボットを右クリックし、[切り替える] を選択します。
2. ロボットやスニペットなどの切り替えるファイルを選択し、[次へ] をクリックします。
3. ファイルをバックアップするかどうかを指定して、バックアップ方法を選択し、[バックアップ] ステップで [終了] をクリックします。
   Kofax RPA がロボットを切り替えるすると、ロボット アイコンの色が変わります。

   注 クラシックからデフォルトのブラウザ エンジンにロボットを切り替えるときは、ロボットの設定の [タイマー イベントをリアルタイムで待機] および [タイム アウトの最大待機時間] 設定がチェックされます。これらの設定が見つかった場合、Design Studio では、切り替え中にこれらの構成設定が失われるという警告が表示されます。これは、ロボットの設定のみに適用されます。ステップに [タイマー イベントをリアルタイムで待機] および [タイム アウトの最大待機時間] 設定がある場合、これらの設定はレガシー待機基準に切り替えられます。

変数の設定

新しいロボットを作成するときは、通常、その変数を設定することから開始します。変数は、変数の初期値の変更など、ロボットのライフタイム中にいつでも再設定できます。

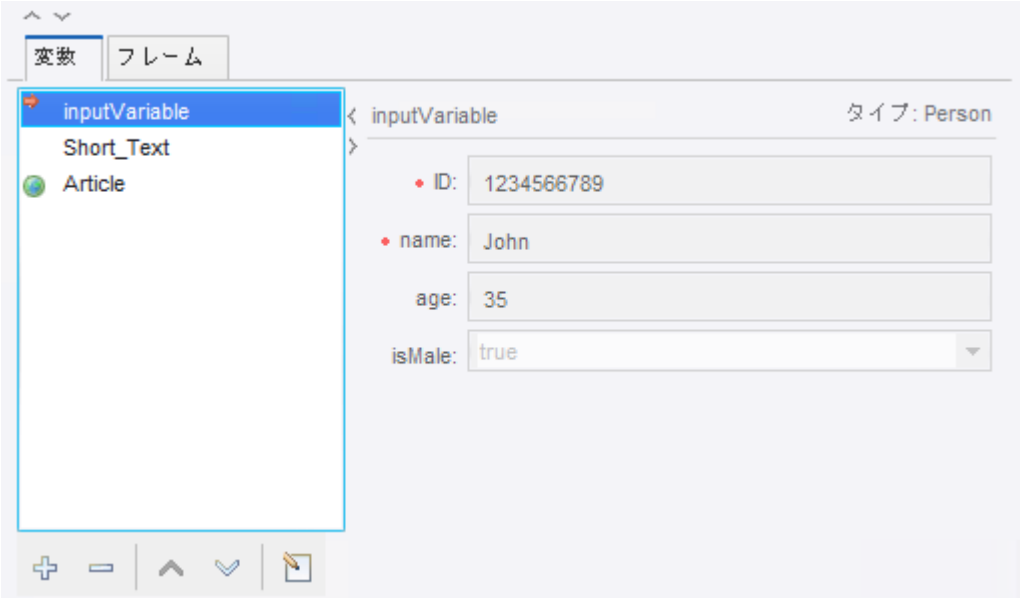
1. ロボット エディターで、ステップ ビューの下にある [変数ビュー] を選択します。

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

   変数ビューには、変数のリストに加えて、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。

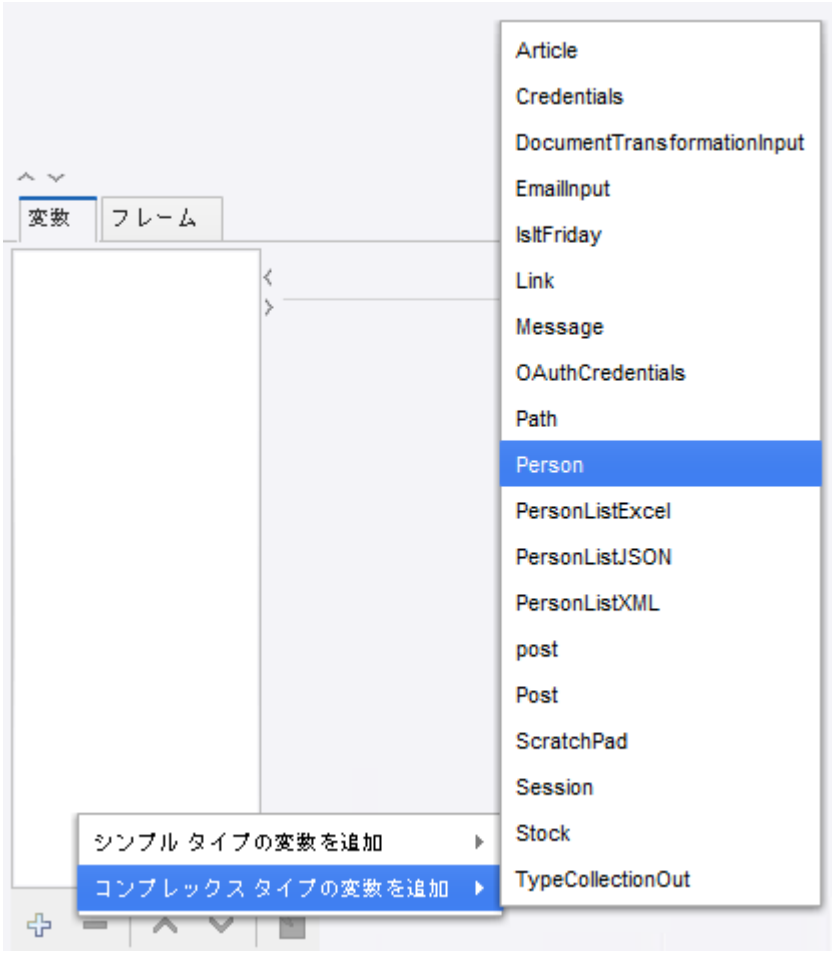
   - 入力変数
   - グローバル変数

   次の変数ビューには、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。



   この変数ビューには、現在のステップの変数の値が表示されます。これらの値はロボットを実行した結果であるため、値を直接変更することはできません。ただし、変数の追加または除去は可能です。

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] ウィンドウが表示されます。

注 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。

注 また、このウィンドウで、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] ウィンドウで、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。[OK] をクリックすると、変数名が不正の場合、通知されます。無効な名前を変更するか、[キャンセル] をクリックします。

注 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。
   タイプと変数への接続の詳細については、変数とタイプをご覧ください。
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。

   注 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ことに留意しない限り、値は、ロボットが実際に実行されるときの値とは異なります。

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。

ロボットのデバッグ

このセクションでは、Design Studio に組み込まれているデバッグ モードを使用してロボットをデバッグする方法について説明します。ロボットが予想どおりに動作することを確認するには、デバッグ モードを使用して、RoboServer で実行されるのと同じ方法でロボットを実行します。

重要 デザイン モードとデバッグ モードでロボットを同時に実行することはできません。

基本的なデバッグ

1. デバッグ モードに切り替えるには、Design Studio で [デバッグ モード] または [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、[再生] 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 パネルで結果を確認することもできます。
   [入力値/出力値] タブの表示内容は次のとおりです。
   - [入力値] パネルには、入力変数が表示されます。

     注 ロボットに入力変数がない場合、[入力] パネルは表示されません。

   - [出力値] パネルには、実行中に返されたすべての値が表示されます。
   - [API 例外] タブには、実行中に生成されたすべての API 例外が表示されます。
   - [ログ] タブには、実行中にログに出力されたログ記録が表示されます。
   - [状態] タブには、ロボット状態 (存在する場合) が表示されます。
   - [概要] パネル (メイン パネルの右側) には、実行の概要が表示されます。この概要には、返された値の数、生成された API 例外の数、HTTP リクエストの数の統計情報、送受信されたデータの量、実行された JavaScript 命令の数が含まれます。

   注 デバッグ モードでの実行は、Design Studio のデザイン モードでの実行とは別個に行われることを理解することが重要です。したがって、デバッグ モードには、デザイン モードの現在のステップおよび現在のロボット状態とは関係のない、独自の現在のステップと独自の現在のロボット状態があります。デバック モードでは、現在のステップは、デバッグ プロセスで実行しようとしているステップまたは実行中のステップであり、現在のロボット状態はそのステップへの入力です。

4. デバッグを終了するには、[終了] をクリックします。
   デバッグはいつでも終了できます。
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。
   当然ですが、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 バーに停止した理由が表示されます。
   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]、[ウィンドウ]、[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、[再生] をクリックして、デバッグを再開します。
   [再起動] 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。

   注 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、いつでも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。Enter を押して、新しい入力変数でデバッグを再起動します。

   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. Design Studio のデザイン モードで現在のステップからデバッグを開始するには、[指定ロケーションから戻る] 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. [再生] 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。

   [指定ロケーションから戻る] をクリックしたときに、Design Studio が既にロボットをデバッグ中である場合、デバッグを再開してから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、デザインに移動または GOTO を使用して、ロケーションに戻ることができます。以下の手順は、両方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. ロボットのどこかのロケーションでデバッグが停止した場合、デバッグ モードで、[デザイン ロケーションに移動] をクリックし、どこかの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を変更したりできます。

   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 [**Goto**] 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] タブに、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 [**Goto**] 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 [**Goto**] 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、問題を修正したい場合に便利です。
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、再生 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、[指定ロケーションからデバッグ] 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、[再生] 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全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。

注 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 [再生] をクリックします。

ステップ イントゥー

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 イントゥー アイコンを使用してグループを折り畳むと、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。

Design Studio の設定

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
- ロボット エディター
- ローカル データベース
- Desktop Automation
- プロキシ サーバー
- 証明書
- バグ レポート
- 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	Design Studio に切り替えたときに何が起こるかを示します。
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。
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。
ドキュメント取得情報の表示	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。

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。

次の表で、「テキストファイル」タブのオプションについて説明します。

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 ストップを指定する。

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。

以下の表で、[ロボット エディター] タブのオプションについて説明します。

オプション	説明
ステップのツールチップを表示	チェックボックスを選択すると、ツールチップがロボット ステップに表示されます。
ステップのエラー処理を表示	選択すると、カスタム エラー処理を含むロボット ステップがシンボルでマーク付けられます。
デフォルトのズーム比	ロボットがロボット エディターで開かれるときに適用されるズーム比をリストします。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。
開くロボットの最大数	ロボット エディターで開くロボットの最大数をリストします。最大を超えようとすると、ロボットを閉じるように求められます。
開くスニペットの最大数	エディターで開くスニペットの最大数。最大数のスニペットが開かれている場合、それ以上のスニペットを開こうとすると、閉じるスニペットを選択するように求められます。
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。

Desktop Automation

このタブを使用して、Desktop Automation 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、Desktop Automation ワークフローでの端末の自動化および Web サイトのブラウズにのみ適用されます。 コマンドはオートメーション デスクトップに送信される命令です。マウス ボタンのクリック、アプリケーションを開く、location found ガードの場合追加などです。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。 Location Found (ロケーションが見つかった場合) ガードの場合、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。	
デフォルトの TLS 構成を使用	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 とサーバーの両方で使用できるようにするには、データベースを Management Console で構成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構成されています。フィールド名、ホスト、タイプ、およびスキーマは必須で、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)。さまざまなタイプのデータベースが Management Console で構成され、Design Studio 起動時に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	いいえ	データベースのユーザー名。
パスワード	いいえ	データベースのパスワード。
最大アクティブ接続	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。
最大アイドル接続	はい	許可されるアイドル接続の最大数。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。

注 これにより、データベースへの接続のみがテストされます。データベースで適切な権限があることはテストされません。

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。

以下の表で、[プロキシ サーバー] タブのオプションについて説明します。

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)。
ポート番号	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 サーバーでログインが必要な場合に使用するパスワード。
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。行ごとに 1 つのホスト名を指定します。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構成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] タブを使用して、ロボットで HTTPS 経由でアクセスする Web サーバーの ID を確認する必要があるかどうかを指定します。そのような確認は、フィッシング攻撃を検出するために通常のブラウザで定期的に (そして目に見えないように) 行われます。ただしロボットで情報を収集する際、ほとんどの場合は確認が不要です。ロボットはアクセスする Web サイト向けに特別に記述され、それら Web サイトにのみアクセスするからです。そのため、確認はデフォルトで有効になっていません。

確認は、ブラウザでの確認の実行と同じ方法で行われます。

Web サーバーの証明書は、ブラウザで構成できるものに似た、インストール済みの信頼されている HTTPS 証明書のセットに基づいてチェックされます。HTTP 証明書の詳細については、『Kofax RPA Developer's Guide』(Kofax RPA 開発者ガイド) を参照してください。

以下の表で、[証明書] タブのオプションについて説明します。

オプション	説明
HTTPS 証明書の確認	選択すると、ロボットは Web サイトの証明書を HTTPS でアクセスしたときに確認します。確認は、2 つの信頼されている証明書のセット、つまりルート証明書のセットとサーバー証明書の追加セットに基づいて行われます。
HTTPS クライアント証明書	ロボットが使用できるクライアント証明書のリスト。リスト下のボタンを使用して、証明書を追加したり除去したりします。

ルート証明書はブラウザにインストールされているように、Design Studio にインストールされています。ルート証明書は、アプリケーション データ フォルダの Certificates/Root フォルダにあります。詳細については、『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) を参照してください。

一部の HTTPS サイトでは、デフォルトで含まれていない証明機関が使用されることがあります。この場合、Design Studio でこれらのサイトからロードするため適切な証明書をインストールする必要がありま

す。ほとんどの場合、これらはアプリケーション データ フォルダの Certificates/Server フォルダにインストールされます。

HTTPS サイトを処理する場合、証明書をルート証明書のセットまたはサーバー証明書のセットのどちらに追加するかは問題ではありません。

証明書をインストールするには、証明書を PKCS#7 証明書チェーン、Netscape 証明書チェーン、または DER エンコード済み証明書として取得する必要があります。証明書をインストールするには、証明書を上記のいずれかのフォルダにコピーします。証明書が含まれるファイルの名前は任意でかまいません。

バグ レポート

[バグ レポート] タブを使用して、内部 Design Studio エラーに関連するバグ レポートを電子メール送信するためのユーザー設定を行います。Design Studio で内部エラーが発生すると、ユーザーはバグ レポートをエラーに関する詳細とともに送信できます。ユーザーは、ヘルプ メニューでバグ報告アクションを選択して、バグ レポートを手動で送信することもできます。

通常、絶対的に必要な場合を除き、ユーザーがこのタブのデフォルトの設定を変更することは推奨されません。

オプション	説明
メール サーバー	バグ レポートの送信時に使用するメール サーバーをリストします。
電子メール送信先	バグ レポートを送信する必要がある電子メール アドレスをリストします。

Management Console

Management Console タブを使って、接続設定を Management Console に設定します。URL は一意でなければなりませんが、異なるプロトコル、ユーザー名、およびパスワードで同じ Management Console への複数の接続を構成することができます。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、そのサーバーは自動的に Management Console のリストに追加されます。

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。

ユーザー名

Management Console でユーザー管理を有効にした場合、Management Console にアクセスするユーザー名を指定します。以下の事前定義されたユーザーは Management Console に接続できます。

- 管理 (admin)
- 管理者 (Administrator)
- プロジェクト管理者 (Project Administrator)
- 開発者 (Developer)

パスワードの記憶
選択すると、Design Studio では、Management Console に入力されたパスワードが保存されます。このオプションをオフにすると、Design Studio から Management Console に接続するたびに必要なパスワードを入力する必要があります。

パスワード

Management Console でユーザー管理を有効にした場合、ユーザーのパスワードを指定します。

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 ストアまたはロボット ファイル システムにアクセスする際に接続する 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。

第 4 章

Desktop Automation

Desktop Automation は、Kofax RPA 10 で導入されました。この機能を使用すると、以下のようなコンピュータ アプリケーションに関わる作業プロセスを自動化できます。

- 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、Desktop Automation の概要を参照してください。

注 Desktop Automation サービスは 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 Desktop Automation Agent と同時に実行しないでください。

また、Desktop Automation を使用してロボットを構築するプロセスについては、Kofax RPAドキュメント セットの『Getting Started with Desktop Automation Guide』(Desktop Automation スタート ガイド) を参照してください。

注 Desktop Automation サービスでは、Windows で整合性レベルが「高」と定義されているアプリケーションに対し、フォーカスの除去や、キーボード入力またはマウス クリックなどの入力の生成を自動化することができません。Desktop Automation サービスは、整合性レベルが「中」のプロセスとして実行されるため、レベルがより高いアプリケーションには入力を生成できません。整合性レベルが「高」で実行されるアプリケーションには、タスク マネージャ、システム プロパティ、管理者権限で実行されるアプリケーションがあります。

また、Desktop Automation サービスは整合性レベルの高いアプリケーションに対してツリーを生成することもできません。

回避策: Desktop Automation サービスを管理者権限で実行し、整合性レベルを「高」に引き上げます。

Windows 10 ユーザーへの注意

Kofax Kapow バージョン 10.3.0.1 以降で Windows 10 をリモート デバイスとして実行している場合、Windows の [スタート] メニューは選択できないため、レコーダー ビューにタブとして表示されません。

症状
Desktop Automation サービス (DAS) を Windows 10 (または対応するバージョンのサーバー) で実行すると、Windows 7 の場合よりも口ボットの動作が著しく遅くなります。さらに、Windows 10 (または対応

するバージョンのサーバー) で実行される DAS を使用するロボットが、Windows 7 での実行時には認識できていた一部のエレメント (ウィンドウ、ポップアップなど) を認識できなくなる可能性があります。

詳細

Kofax Kapow バージョン 10.2.0.3 ~ 10.3.0.0 には、ユニバーサル Windows プラットフォーム (UWP) アプリケーション (Windows 8.1 および Windows 10 での名称は Metro スタイル アプリケーション) で発生する問題を修正するためのパッチが含まれています。Windows デスクトップ上に開いたアプリケーションのルーピング (列挙) (これは 1 秒間に 5 回実行されます) 時に、従来のメソッドではこれらのアプリケーションはスキップされます。

このパッチを使用することで、ロボットは Metro スタイル アプリケーションを含めた、Microsoft UI オートメーション API により提供されるメソッドを使用しますが、以下の 2 つの副作用が発生します。

- 一般的に、このメソッドは CPU の消費が大きく、時間がかかります。さらに Desktop Automation サービス (DAS) がシングル スレッドであるため、すべてのインストール環境でパフォーマンスの低下が発生します。
- 特定のアプリケーションについては、1 つ以上のインスタンスにで DAS (node.exe プロセス) およびアプリケーションの両方の CPU 使用率が 100% に到達することが観察されています。

解決方法

バージョン 10.2.0.3 ~ 10.3.0.0 には、以下の環境変数を設定することで元の列挙アルゴリズムに切り替えるメソッドが存在します。

```
KAPOWHUB_APPLIST_VERSION=1
```

以下の手順に従って、この環境変数を追加します。

1. DAS を停止する (DAS システムのトレイ アイコンを右クリックし、ショートカット メニューからオプションを選択)。
2. タスク マネージャを開き、node.exe プロセスが実行中でないことを確認する。実行されていた場合は停止する。
3. タスク マネージャで、すべての DesktopAutomationServiceControl.exe を停止する。このプロセスは DAS コントローラ (システム トレイ アイコンも含む) であるため、非常に重要です。DAS を開始すると、コントローラはシステムから環境変数を取得し、Desktop Automation サービスはこの変数をコントローラから取得します。そのため、新しい環境変数が DAS で使用される場合、コントローラを再起動する必要があります。
4. 環境変数を設定します。
5. Desktop Automation サービスを再起動します ([スタート] > [プログラム] から、システム トレイと node.exe の両方が再起動されます)。

この環境変数を追加すると、Windows 8.1 および 10 で実行される DAS のパフォーマンスが改善し、Metro スタイル アプリケーションが自動化されなくなります。

Kofax Kapow バージョン 10.3.0.1 では、前のバージョンにあった、いくつかのマイナーなパフォーマンスの問題が改善し、デフォルトで元の列挙メソッドを使用するように戻されています。そのため、Kofax RPA バージョン 10.3.0.1 以降では、Windows 8.1 または 10 と、Windows 7 との間のパフォーマンスの問題はありませんが、デフォルトで Metro スタイル アプリケーションが自動化されないため、レコーダー ビューで Windows のスタート メニュー タブが表示されなくなります。

Metro スタイル アプリケーションを自動化する場合は、以下の環境変数を設定します。

```
KAPOWHUB_APPLIST_VERSION=2
```

この変数を設定する手順は上記と同じです。この変数を設定する場合は、パフォーマンスの低下が発生することに注意してください。

Desktop Automation の概要

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 Desktop Automation ロボットを作成します。Desktop Automation は実質上、Web サイトおよびデータベースのオートメーションとは異なるため、Design Studio にはこの目的のための専用のワークフロー言語およびステップがあります。

Desktop Automation ロボット

Desktop Automation 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。ステップでは、ユーザーが自動化中のアプリケーションとどのように相互作用するかがモデル化されます。Desktop Automation ロボットを使用するには、「Desktop Automation ワークフローの呼び出し」という専用のアクション ステップを使って呼び出す必要があります。

ロボット (Web オートメーション ロボット) に、それぞれ独自のワークフローを持つ「Desktop Automation ワークフローの呼び出し」ステップを複数含むことができます。複数の Web オートメーション ロボットで 1 つの Desktop Automation 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場合は時間を大幅に短縮できます。「Desktop Automation ワークフローの呼び出し」ステップを使用するロボットは、スケジュールから、API 経由、Kapplet 経由、あるいは開発またはテスト中に手動でその他の Kofax RPA ロボットと同様に実行可能です。

Desktop Automation ワークフロー

Desktop Automation 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、Desktop Automation ワークフローを編集する を参照してください。

メニューの [ヘルプ] ボタンから、関連したドキュメントと、Desktop Automation を使用したロボットの構築プロセスを解説しているスタート ガイドへのリンクを利用できます。

ステップ

ステップは、Desktop Automation 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。Desktop Automation では、終了ポイントがないいくつかのステップを除く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、共に属している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されます。ステップの完全なリストについては、Desktop Automation ステップ を参照してください。

Desktop Automation のステップは通常粒度が細かく、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

Desktop Automation の目的は、アプリケーションの制御の自動化です。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 エージェントと接続することにより、Desktop Automation を実行します。ロボットから直接接続されている端末がデバイスで実行されている場合を除きます。デバイスの処理およびエージェントの設定については、Desktop Automation サービスの設定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 を使用している場合など、Kofax RPA で Internet Explorer の拡張サポートを有効にして、DOM (Document Object Model) ツリーを取得することにより、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。

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、「派生属性」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、派生属性として境界ボックス (x、y、幅、高さ) を追加します。派生属性は、「`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派生属性のリストとその説明を示します。

派生属性	説明
すべての要素を対象とする属性	
der_x	要素の左上隅の X 座標。
der_y	要素の左上隅の Y 座標。
der_width	要素 (要素の境界ボックス) の幅。
der_height	要素 (要素の境界ボックス) の高さ。
der_rendered	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	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。
フォーム コントロールの (入力) 要素を対象とする属性	
der_value	「email」、「text」、「number」、「range」、「tel」、「time」、「url」、「search」、「date」、「datetime-local」、「week」、「color」、「month」、および「textarea」の入力要素に使用されます。
der_checked	「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。

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。

Desktop Automation ワークフローと **Web** サイト 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) の状態をロボットで内部的に追跡できます。一方、Desktop Automation 機能は、状態がアプリケーションに存在するリモート 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、状態はロボットの外部になります。

Desktop Automation でのステップの実行は、前にのみ移動します。実行の状態はリモート デバイス上のものであり、ワークフローを戻って元に戻すことはできません。ただし、「値を抽出」ステップと「値

の変換」ステップのグループは除きます。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 Desktop Automation ワークフローを編集する で明示的に実行を選択するまで実行されません。

重要 分岐は、Kofax RPA Design Studio で設計されているため、Desktop Automation サービスには存在しません。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、分岐が順々に続けて実行される Web サイト ロボットとは異なり、状態は各分岐の開始時に戻ります。

Desktop Automation で、エラー処理はすべてのステップに対して指定されているわけではないために異なります。代わりに、try-catch ステップ ではその範囲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。

一般に、Desktop Automation ワークフローの設計時に、ユーザーがどのように自動化中のアプリケーションのユーザー インターフェイスと相互作用するかを考えます。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。

Desktop Automation には、ロボット デザイナーでオートメーションを設計し、アプリケーションの外部状態を判断して、適切に対応できるようにする機能があります。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。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 エレメントを見つけ、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。

リモート デバイスを自動化するステップについては、はじめにを参照してください。

はじめに

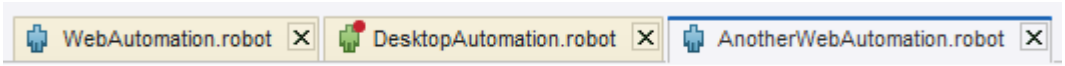
次の説明は、Kofax RPA をダウンロードしていずれかのコンピュータにインストールしていることを前提としています。Kofax RPA の使用を開始するには、『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) の「Quick Start Guide」(クイック スタート ガイド) を参照してください。

1. Kofax のダウンロード ポータルから Kofax_RPA_DesktopAutomation_11.0.0_x32.msi インストール ファイルをダウンロードします。デバイスが『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) の「Desktop Automation requirements and prerequisites」(Desktop Automation の要件および前提条件) に記載されている要件を満たしていることを確認します。
2. 自動化したいアプリケーションを実行するリモート コンピュータに Desktop Automation サービスのインストールおよび設定を行います。端末アプリケーションを自動化するだけの場合は、このステップをスキップします。
3. Design Studio を開きます。

4. Desktop Automation ロボットを作成します。
   a. **[ファイル]** > **[新しい Desktop Automation 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。青いアイコンで識別される Web オートメーション ロボットとは対照的に、Desktop Automation ロボットは緑色のアイコンで識別されます： 
      まず Web オートメーション 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 Desktop Automation ワークフローを編集できません。
5. 既存の Web オートメーション ロボットを開くか、**[ファイル]** > **[新しいロボット]** をクリックして新しい Web オートメーション ロボットを作成します。
6. 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、[**Desktop Automation** ワークフローの呼び出し] を選択します。
   b. [ワークフロー] ドロップダウン リストで、ステップ 4 で作成した Desktop Automation ロボットを選択します。
   c. 入力値、出力マッピング、および必要なデバイスを設定します。「オートメーション デバイス マッピング」およびオートメーション デバイスの参照を参照してください。ターミナル アプリケーションを自動化するだけの場合は、必要なデバイスの設定をスキップします。
7. Web オートメーション ロボットの実行を許可するには、ツールバーで [実行の準備]  をクリックします。

   重要 Web オートメーション ロボットの実行を許可するか、Web オートメーション ロボット間で実行権限を切り替えるには、ロボットが表示されているタブを開き、ツールバーで [実行の準備]  をクリックします。正しく実行されるようにするために、複数のロボットを同時に実行することはできません。また、デザイン モードとデバッグ モードで同時に実行することもできません。

   ロボットに実行権限がある場合、ロボット アイコンに赤い点が表示され、ロボット タブが強調表示されます。Desktop Automation ロボットに実行権限がある場合、Web オートメーションロボットが呼び出しているタブも強調表示されます。

8. 実行が許可されている場合は、Desktop Automation ワークフローを編集のために開きます。この操作を行うには、ワークフローを「Desktop Automation ワークフローの呼び出し」ステップまで実行してから、ツールバーの **[DA** ロボットにステップ イントゥー]  をクリックします。
   Desktop Automation 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。
9. これで、オートメーション ワークフローの設計を開始できるようになりました。ワークフローを実行して、動作を確認することもできます。
10. Desktop Automation ロボットの設計が完了したら、それを展開して実行し、デバイスを自動化することができます。
    - Desktop Automation ロボットからステップアウトして、Web オートメーション ロボットでの作業に切り替えるには、Desktop Automation 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。Web オートメーション ロボットで、「Desktop Automation ワークフローの呼び出し」ステップは実行されていると表示されます。

- 結果を保存しないでロボットを終了するには、ツールバーの [ロボットの終了] をクリックします。Web オートメーション ロボットで、「Desktop Automation ワークフローの呼び出し」ステップが「実行されていない」と表示されるようになりました。

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 以前では、Desktop Automation アクション ステップに含まれている Desktop Automation 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ました。Kofax RPA 11 で、バージョン 10.7 以前で作成された Desktop Automation アクション ステップを実行できますが、ワークフローを編集するには、アクション ステップを Desktop Automation ロボットに変換する必要があります。そのためには、Desktop Automation ステップを開き、[アクション] タブの [ワークフローにエクスポート] をクリックします。新しいロボットに名前を付けて、[終了] をクリックします。

Desktop Automation ステップが Desktop Automation ロボットにエクスポートされて、このセクションの上記の手順に従って使用できるようになりました。

オートメーション デバイスの参照

「Desktop Automation ワークフローの呼び出し」ステップでロボットを編集する前に、「Desktop Automation ワークフローの呼び出し」ステップの [アクション] タブの [必要なデバイス] フィールドにある をクリックして、デバイス リファレンスを入力します。[デバイスの追加] ウィンドウで、[スタティック リファレンス] または [ダイナミック リファレンス] を選択します。「ローカル」リファレンスは、その他のリファレンスの入力に関わらず、常にデフォルトで利用可能です。ローカル リファレンスは、組み込みブラウザで Web サイトにアクセスし、組み込み Excel ドライバーで Excel スプレッドシートを操作するために使用されます。

注 端末コンピュータを自動化する際は、Desktop Automation エージェントをインストールせず、オートメーション デバイス リファレンスを入力しないでください。

スタティック リファレンス

スタティック 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、組み込みブラウザと組み込み Excel ドライバーで操作する端末の自動化には不要です。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、スタティック リファレンスを使用します。

ダイナミック リファレンス

このタイプのリファレンスでは、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します。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。その他すべての接続パラメータは、Desktop Automation ワークフロー内で指定されます。ダイナミック リファレンス接続が「 Desktop Automation ワークフローの呼び出

し」ステップでロボットによって使用され、デバイスに接続すると、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「Desktop Automation ワークフローの呼び出し」ステップで使用できます。

重要 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。

トリガーで参照

トリガーで参照は、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。詳細については、オートメーション デバイスのマッピングを参照してください。

オートメーション デバイス マッピング

オートメーション デバイスのマッピングにより、Design Studio のプロジェクトからこのデバイスにアクセスできるようになります。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ことをお勧めします。この方法で、作成するロボットがネットワーク インフラストラクチャが変更された場合でも、オートメーション デバイスが常に指定された専用の Management Console で動作できるようになるためです。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する必要があります。Design Studio で使用するラベルは、設計時に使用するデバイスと一致していなければならず、Management Console でのラベルはプロダクション環境で使用するデバイスと一致する必要があります。

重要 オートメーション デバイス マッピング名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で構成される必要があります。また、'false' または 'true' にすることはできません。

デバイス マッピングの構成のデバイス マッピング オプションは、Design Studio におけるロボットの開発とデバッグに推奨されます。

注 端末デバイスを自動化するには、リモート コンピュータに Desktop Automation サービスをインストールしない、またはデバイス マッピングを Management Console で作成しないでください。

オートメーション デバイスのマッピング

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。あるいは、オートメーション デバイスの名前を入力します。[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。

3. 「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
   このオプションは、Management Console でマッピングを使用してオートメーション デバイスに接続するときに役立ちます。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
   このオプションは、オートメーション デバイスに直接接続するときに役立ちます。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。

デバイス マッピングの設定

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。[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
このオプションは、Management Console でマッピングを使用してオートメーション デバイスに接続するときに役立ちます。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
このオプションは、オートメーション デバイスに直接接続するときに役立ちます。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。

Desktop Automation ワークフローを編集する

Desktop Automation ロボットで作業しているときには、次のペインがエディター存在します。エディターのウィンドウを切り離して移動し、編集しやすくすることができます。

注 Desktop Automation ワークフローを編集するには、Web オートメーション ロボットの実行を許可するために [実行の準備] を最初にクリックし、次に [**DA** ロボットにステップ イントゥー] をクリックし、Desktop Automation ワークフローを開いて編集します。詳細については、はじめに を参照してください。

- エディター: ステップのワークフロー、および変数とエクスプレッション、ツリー モードの設定が含まれます。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役立ちます。[実行の開始]、[ステップ イントゥー]、[ステップ オーバー]、[ステップ アウト] ボタンを使用して、ワークフローを進めることができます。ロボットの実行を一時停止したりリセットすることもできます。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/削除します
  - Ctrl + Shift +クリック: 範囲の選択を追加します

ボタン	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	ワークフローの変更を保存します。
元に戻す	最後の変更を元に戻します。
やり直し	元に戻したアクションを繰り返します。
コピー	ワークフローの選択されたエレメントをクリップボードにコピーします。たとえば、複数のステップまたはファインダーです。
カット	選択したステップをカットします。
貼り付け	クリップボードの内容を選択されたエレメントのワークフローに貼り付けます。たとえば、フロー ポイントまたはファインダーです。
削除	選択したステップを削除します。
ロボットの終了	実行結果を保存しないで、Desktop Automation ロボットを停止して終了します。
実行の開始	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。
一時停止	オートメーション ワークフローの実行を一時停止します。
ステップイントゥー	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は、そのステップは展開されます。これは、本質的にはシングル ステップ ケースに関連しています。
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 ポイントに対して実行されます。

ボタン	説明
ステップ アウト	現在のフロー ポイントを含むステップの直後のフロー ポイントに対して実行されます。ワークフローが最後まで実行された場合、このボタンをクリックするときに Desktop Automation ロボットが終了します。
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できます。これ以上イテレーションがなくなると、実行は繰り返しステップ外のフロー ポイントで停止します。
リセット	Desktop Automation ワークフローの実行をリセットします。

ボタン	説明
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。
折りたたむ	[オートメーション ワークフロー] ペインのすべてのステップとその他のエレメントを折りたたみます。
展開	[オートメーション ワークフロー] ペインのすべてのステップとその他のエレメントを展開します。
選択したステップ以外すべて閉じる	選択されているものを除くすべてのステップとその他のエレメントを折りたたみます。

新しく挿入されたステップは、ツールバーの [ステップ イントゥー] 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。

ワークフロー ビューの拡大

ユーザビリティを向上させるため、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 キーと +、または Ctrl キーとマウス ホイールのスクロール アップ
- 縮小するには：Ctrl キーと -、または Ctrl キーとマウス ホイールのスクロール ダウン

- [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。
ビューでエレメントをウィンドウ座標とともに選択すると、選択したエレメントの左上隅を基準とした座標、および下部のバーの要素へのパスが表示されます。
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、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。
タグは、現在の状態に従って色の付いたボックスでマークされます。

- タグに緑のボックス: 現在選択しているタグ。
- タグにオレンジ色のボックス: 1 つ下のレベルのタグ。
- タグの青のボックス: 3 番目のタグ。

各種レベルの要素を切り替えるには、[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]、[前の兄弟ノードを選択]、[次の兄弟ノードを選択] ボタンを使用します。

注 アプリケーション ビューのテーブルのセル エレメントを選択できないことがあります。アプリケーション ツリーのセル エレメントを選択したり、ツリー ビューからステップ アクションを追加したりできます。

[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 + マウス ホイールを上にスクロール
- 縮小するには：Ctrl + マウス ホイールを下にスクロール

ボタン	説明
一時停止 再開	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になると、レコーダー ビューに以下が表示されます。ストリーム ライブ。 ストリーミングが一時停止すると、レコーダー ビューに以下が表示されます。ストリーム 一時停止中。
選択用ファインダーを作成	[オートメーション ワークフロー] ビュー ファインダーでの選択を [レコーダー ビュー] での選択に一致するファインダーと置き換えます。
見つかった次のロケーションを表示	ファインダーに一致する次のエレメントを表示します。ボタンのツールチップには、一致したエレメントの数も表示されます。
次のノード マッチング クリックを選択	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。
シンプル/9 グリッド イメージ ファインダーを切り替える	イメージ選択をシンプル/9 グリッド間で変更します。
x1 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	選択した子ノードの最初の子ノードを選択します。

ボタン	説明
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	選択した子ノードの最初の子ノードを選択します。
前の兄弟ノードを選択	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。
次の兄弟ノードを選択	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。
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。
自動実行	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。有効になると、ボタンの円が赤色になります。 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。

- [ワークフロー状態] ビュー：変数値など、ワークフロー実行の状態を表示します。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。
- コメント: オートメーション ワークフロー ステップのコメントを表示します。
  コメントを入力または変更するには、ステップまたはグループ ステップをクリックして、[コメント] ウィンドウでメモを追加/変更します。ここでは [元に戻す] および [やり直し] ボタンを使用できます。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。
  コメントを含むステップには、コメント記号が付きます。

エディターの手順

変更の保存/元に戻す
ツールバーで **[ファイル]** > **[保存]** をクリックすると、Desktop Automation ワークフローだけでなく、ロボット全体が保存されます。ワークフローの編集をキャンセルする場合は、[閉じる] をクリックしてから、ロボット エディターで [元に戻す] (Ctrl + Z) をクリックします。

ワークフローを **PDF** にエクスポート
ロボット ワークフローを現在の状態で PDF ファイルに保存できます。ツールバーで **[ファイル]** > **[ワークフローを PDF にエクスポート]** をクリックし、コンピュータ上の保存場所を選択します。

Desktop Automation サービスの設定

この章では、Desktop Automation サービスの設定について説明します。

Desktop Automation の前提条件

Desktop Automation の要件および前提条件については、すべて『Kofax RPA Installation Guide』(Kofax RPA インストールガイド) の「Dependencies and Prerequisites」(依存関係と要件) の章に記載されています。

注 Desktop Automation サービスは 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 Desktop Automation Agent と同時に実行しないでください。

Desktop Automation エージェントの設定

コンピュータが Desktop Automation に必要な要件をすべて満たすと、Desktop Automation エージェントをインストールおよび構成することができます。

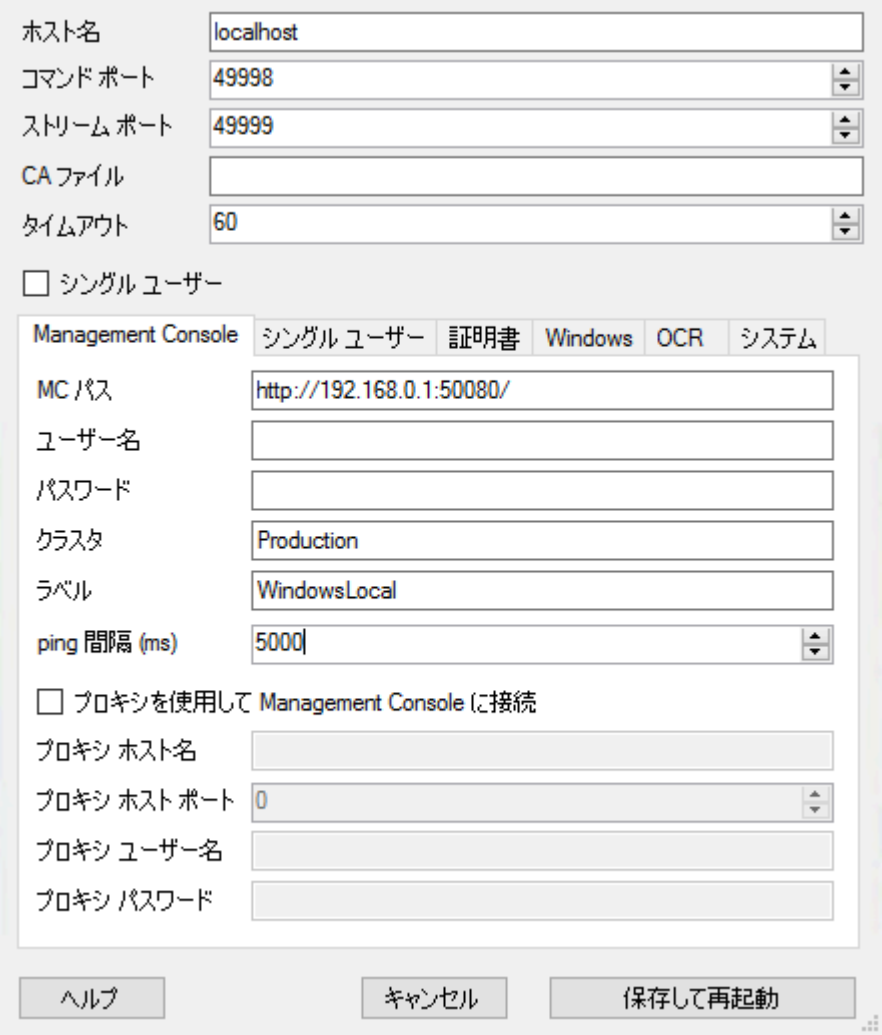
1. Java アプリケーションを自動化する必要がある場合、Java 32 ビット (JRE または JDK) をリモート デバイスにインストールし、Java Access Bridge がデバイスで有効になっていることを確認します。詳細については、Java Access Bridge の確認 を参照してください。
2. Kofax RPA Desktop Automation インストーラをデバイスにダウンロードし、実行します。
3. Desktop Automation サービスをスタート メニューから開始します。サービスが開始すると、そのステータスを通知領域のアイコンで確認できます。

アイコン	ステータス
	Desktop Automation サービスが開始し、設定されている Management Console に接続しようとしています。
	Desktop Automation サービスが実行中で、設定に応じて Management Console に接続されているか、またはシングル ユーザー モードで実行中です。

アイコン	ステータス
	Desktop Automation サービスが実行中で、RoboServer または Design Studio によって使用中です。
	Desktop Automation サービスは実行していません。
	Desktop Automation サービスはエラーのため実行していません。

4. Desktop Automation サービス オプションを編集するには、通知領域の Desktop Automation サービス アイコンを右クリックし、[設定] を選択します。これにより、Desktop Automation サービス ウィンドウが開きます。オプションを変更したら、[保存して再起動] をクリックします。
   オプションを手動で編集するには、オートメーション デバイスの `server.conf` ファイルを開きます。ファイルは [ユーザー] > [ユーザー名] > [AppData] > [ローカル] > [Kofax RPA 11.0.0] フォルダにあります。ユーザー名はサービスが実行されているユーザーの名前です。
   以下の Desktop Automation サービス オプションの表を参照してください。
5. **[管理]** > **[デバイス]** タブでデバイスが Management Console に登録されていることを確認します。

以下は、Desktop Automation サービス構成ウィンドウです。



次の表に、利用可能な Desktop Automation サービス オプションを一覧表示します。

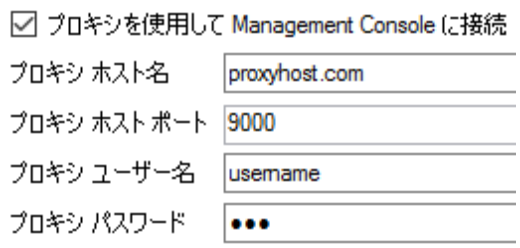
設定ウィンドウ オプション	**server.conf** オプション	値と説明
[シングル ユーザー] クリア (デフォルト) Design Studio からオートメーション デスクトップへの直接接続、または RDP 接続 の使用時に選択します。	"singleUser"	`false` (デフォルト) `true` false に設定すると、Desktop Automation サービスが指定された Management Console に自動的に登録されます。 オートメーション デスクトップへの直接接続の場合、`true` に設定し、トークンを指定します。*

設定ウィンドウ オプション	**server.conf** オプション	値と説明
[ホスト名]	"hostName"	Desktop Automation サービスを実行しているコンピュータの名前または IP アドレス。 コンピュータに複数の名前または IP アドレスがある場合、RoboServer と Design Studio がこの Desktop Automation エージェントと通信するものを指定します。つまり、ホスト名または IP アドレスは RoboServer と Design Studio から接続可能である必要があります。
[コマンド ポート]	"commandPort"	`49998` (デフォルト) 必要に応じて、このポートをオートメーション デスクトップに再割り当てします。
[ストリーム ポート]	"streamPort"	49999 (デフォルト) このポートは、Design Studio と Desktop Automation サービス間のデータ送信に使用されます。streamPort が "0" に設定されると、Desktop Automation サービスではランダムにポート番号が選択されます。 Design Studio とオートメーション デスクトップの間にファイアウォールがある場合、streamPort を再割り当てする必要があることがあります。
[**CA** ファイル]	"caFile"	空 (デフォルト) SSL を使用して Management Console と通信できます。`node.js` のデフォルトの証明書が使用されない場合、このパラメータを使用して別の証明書へのパスを指定できます。これを動作させるにはルート証明書が必要であります。Google Chrome ブラウザからルート証明書をファイルに保存するには、次のようにします。 1. アドレス バーのロック アイコンを右クリックし、[証明書 (有効)] をクリックします。 2. [証明のパス] タブで、一番上の (ルート) 証明書を選択し、[証明書の表示] をクリックします 。 3. [詳細] タブで [ファイルにコピー] をクリックし、ウィザードを完了してルート証明書を base-64 エンコード X.509 証明書としてエクスポートします。 これで、エクスポートされた証明書を含むファイルへのパスを指定できるようになります。

設定ウィンドウ オプション	**server.conf** オプション	値と説明
[タイムアウト]	"commandTimeout"	このオプションでは、コマンド実行のタイムアウトを秒単位で指定します。コマンドはオートメーション デスクトップに送信される命令です。マウス ボタンのクリック、アプリケーションを開く、location found ガードの場合追加などです。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。 Location Found (ロケーションが見つかった場合) ガードの場合、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。 Desktop Automation ワークフローでの端末の自動化、または Web サイトのブラウズに対するコマンド タイムアウトは、Design Studio でのワークフロー実行の場合は [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] タブの **Desktop Automation** セクションで設定します。
[シングル ユーザー] タブの [トークン]	"token"	空 (デフォルト) "singleUser" オプションが `false` に設定されている場合、このオプションは空のままにします。オートメーション デスクトップ (`"singleUser": true`) への直接接続を使用する場合、トークンを指定します。定義されたどのトークンでもかまいません。

設定ウィンドウ オプション	**server.conf** オプション	値と説明
[証明書] タブ  	"tlsServerConfig"	Kofax RPA では、オートメーション デスクトップと RoboServer、または Design Studio との間の TLS 通信を提供しています。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。以下は、`server.conf` ファイル コードの抽出です。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。 `"tlsServerConfig": { "key": "kapow.remote.das.pem", "cert": "kapow.remote.das.cert.pem", "ca": "./serverCa" },`
[Windows] タブ	"automationnative"	• `"useLegacy"` 状況によっては、Java Access Bridge が動作しないため、レガシー モードに切り替えることをお勧めします。デフォルトは `false` です。 • インストール済みパッケージ このコンピュータにインストールされている Desktop Automation サービスパッケージを表示します。バージョン 10.7 から、以下の [パッケージをロック] オプションが選択されていない場合は、新しいバージョン パッケージが自動的にインストールされます。ZIP ファイルのパッケージは、自動化されたコンピュータの `C:\ProgramData\Kofax RPA` にインストールされます。RoboServer のバージョンに応じて、適切なパッケージが自動的に選択されます。使用するバージョン パッケージを 1 つだけ指定する場合は、[パッケージをロック] を選択し、インストールされているパッケージの 1 つを選択します。 • パッケージをロック 選択すると、作業する唯一のバージョン パッケージを選択できます。異なるバージョンの RoboServer は、このサービスに接続できません。デフォルト: オプションはクリアまたは server.conf ファイルで `false` です。 トリガーを使用してロボットを実行する場合、このオプションを選択するか、server.conf ファイルで設定を `true` に変更します。 • RFS 共有をドライブ文字にマッピング ロボット ファイル システムのファイル共有が使用している Windows ドライブ。ファイル共有が Windows ドライブにマップされると、他の Windows アプリケーションもこのファイル共有にアクセスできます。

設定ウィンドウ オプション	**server.conf** オプション	値と説明
[**OCR**] タブ	"ocrConfig"	"defaultLanguage": "eng" OCR 操作を実行する言語を指定します。デフォルトで、Kofax RPA では英語がインストールされます。言語のインストール手順については、デフォルトの OCR 言語の変更を参照してください。
[システム] タブ		このタブでは、ログ ファイルを開いてエラーを調べることができ、サービス ファイルのバージョンと場所が表示されます。このタブを使用して、サービスが実行されているコンピュータに Java Access Bridge が適切にインストールされているかどうかを確認できます。
Management Console オプション		
[**MC** パス] 接続プロトコル、名前または IP アドレス、ポート番号、およびデバイス パスを登録する必要がある Management Console 。形式は次のとおりです。 `http://10.10.0.136:50080.`	"hostName"	デバイスを登録する必要がある Management Console の名前または IP アドレス。
	"port"	指定した Management Console の接続ポート。
	"schema"	指定した Management Console の接続プロトコル。
	"path"	空 (デフォルト) ポート番号の後のスタンドアロン Management Console へのパスの部分。たとえば、Management Console が Tomcat (http://computer.domain.com:8080/ManagementConsole/) で展開されている場合、このパラメータで `"/ManagementConsole/"` を指定します。埋め込み Management Console インストールの場合、このパラメータを空のままにします。
[ユーザー名]	"user"	空 (デフォルト) 指定された Management Console で認証するためのユーザー名。
[パスワード]	"password"	空 (デフォルト) 指定された Management Console で認証するためのパスワード。
[クラスタ]	"cluster"	プロダクション (デフォルト) 指定した Management Console のクラスタ名。
[ラベル]	"labels"	"label1,label2" (デフォルト) オートメーション デバイスを区別するためのラベル。
[**ping** 間隔 **(ms)**]	"pingInterval"	5000 (デフォルト) Desktop Automation サービスが Management Console に ping 送信する時間間隔。

設定ウィンドウ オプション	**server.conf** オプション	値と説明
[プロキシを使用して **Management Console** に接続]	"useProxy"	このオプションを Desktop Automation サービスに選択し、Management Console への接続時にプロキシを使用します。必要なパラメータはすべて次のフィールドで指定されます。  Linux の場合、`server.conf` ファイルの `managementConsole` セクションでプロキシパラメータを設定できます。 `"useProxy": true, "proxyHostName": "proxyhost.com", "proxyPort": 9000, "proxyUserName": "username", "proxyPassword": "pwd"`

* オートメーション デスクトップへの直接接続は、Design Studio でのロボットの作成とデバッグ、および RDP 接続との使用にお勧めします。

Desktop Automation サービスのロギング

Kofax RPA はサービス パフォーマンスを向上するために、特定の Desktop Automation サービス イベントの使用情報を収集します。

- Desktop Automation サービスが Management Console に接続している場合、イベントはRoboServer Log Databaseに保存されます 。Management Console の RoboServer ログデータベースに保存されます。イベントを表示するには、[ログ ビュー] タブで [**DAS**] を選択します。

  注 Management Console の接続パラメータが Desktop Automation サービスの設定ウィンドウで指定されている場合、[シングル ユーザー] モードが選択されていても、イベントは常に Management Console にログ記録されます。つまり、自動化デスクトップへの接続は Management Console がなくても直接確立されます。

- もし Desktop Automation サービスが Management Console に接続できない場合 (Management Console が設定されていないため)、次の場所にある **Desktop Automation Service Usage.csv** のログファイルにイベントが書き込まれます。`{path}\AppData\Local\Kofax RPA\<バージョン番号>\Logs\`
  ファイルの場所は、**log4net.xml** ファイルで設定できます。

各イベントの情報には次が含まれています。

- イベントが発生した時間 (UTC)。

- イベントの種類：開始、停止、接続、切断、中断、スクリーン ロック。
- Desktop Automation サービスの識別。`host:port` フォームの ID、サービスを実行しているユーザー アカウント、サービスに定義されたラベルを含みます。
- ロボットの名前と実行 ID (接続と切断の場合のみ)。
- 重要度の表示 (常に「情報」)。
- メッセージ (常に空)。

Desktop Automation でのプロキシ サーバーの設定

すべての Desktop Automation サービス ロボットは Kofax RPA グローバル プロキシ設定を使用できます。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 サーバーの設定を構成するには、2 つの方法があります。

重要 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限り、ロボットが Kofax RPA グローバル プロキシ設定を使用していることを確認してください。Desktop Automation の詳細については、Kofax RPA のオンライン ヘルプを参照してください。

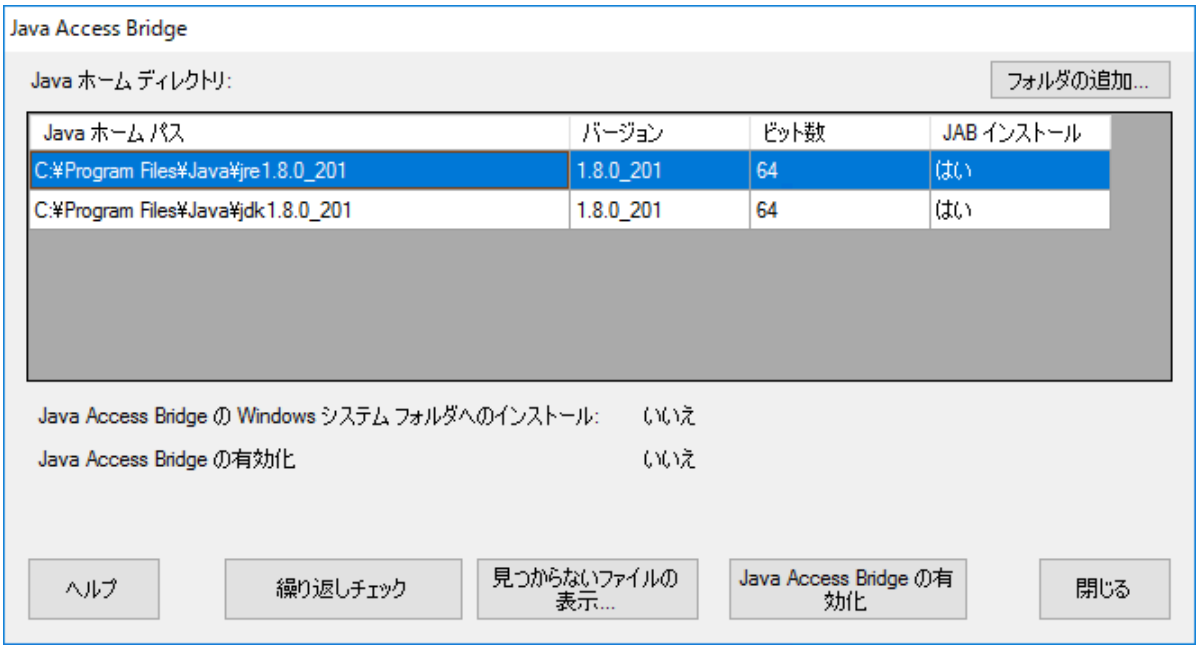
1. Desktop Automation サービスで実行しているすべてのロボットに対して、[Design Studio 設定] ダイアログ ボックスの [プロキシ サーバー] タブで、次のプロキシ サーバーの詳細を入力します。
   - ホスト
   - ポート番号
   - ユーザー名
   - パスワード
   - 除外ホスト名
2. 展開されたすべてのロボットに対して、[Management Console] > [クラスタ設定] > [プロキシ サーバー] タブで、[プロキシ サーバーを追加] を選択し、次のプロキシ サーバーの詳細を入力します。
   - ホスト名
   - ポート番号
   - ユーザー名
   - パスワード
   - 除外ホスト名

Java Access Bridge の確認

Java Access Bridge は、Java アプリケーションを自動化する際に不可欠なコンポーネントです。Java のバージョンによっては、必要なファイルの一部がシステム フォルダ内にないことがあるため、Java Access Bridge は Desktop Automation サービスがインストールされているコンピュータで無効になっている場合があります。Java Access Bridge のインストール状態を確認するには、以下の手順を実行します。

1. 通知領域で Desktop Automation アイコンを右クリックし、[設定] を選択します。
2. [システム] タブをクリックし、[**Java Access Bridge** ファイルの確認] をクリックします。
   [**Java Access Bridge**] ダイアログ ボックスが開き、インストールされている Java のバージョンと、各バージョンの Java Access Bridge のインストール ステータスが表示されます。[**JAB** イン

ストール] 列、[**Java Access Bridge** の **Windows** システム フォルダへのインストール]、[**Java Access Bridge** の有効化] に [はい] と表示されている場合、Java Access Bridge はコンピュータ上に適切にインストールされ、有効にされています。



3. [**Java** ホーム ディレクトリ] の下に Java の実装が表示されない場合、[フォルダの追加] をクリックし、Java ファイルがインストールされているホーム フォルダを指定します。
4. [**JAB** インストール] 列に [いいえ] が表示されるなど、見つからないファイルがある場合は、[見つからないファイルの表示] をクリックします。

   [**Java Access Bridge** の見つからないファイル] ダイアログ ボックスに指定したフォルダにコピーする必要のあるファイルが表示されます。[見つからないファイルのインストール] をクリックして、Kofax RPA から提供されている最新バージョンの Java Access Bridge ファイルを Desktop Automation サービスのインストール フォルダにインストールします。
5. [**Java Access Bridge** の有効化] に [いいえ] と表示されている場合は、[**Java Access Bridge** の有効化] をクリックします。

デフォルトの OCR 言語の変更

Kofax RPA では、Tesseract OCR エンジンを使用して、テキストをイメージからキャプチャします。デフォルトで、Kofax RPA には OCR の言語として英語がインストールされます。ロボットが 画像からテキスト抽出 でテキスト認識を実行するとき、Kofax RPA では Desktop Automation サービス ウィンドウの [**OCR**] タブで選択した言語が使用されます。OCR のデフォルトの言語を変更するには、次のステップを実行します。

1. 必要な言語の `.traineddata` ファイルを https://github.com/tesseract-ocr/tessdata からダウンロードします。たとえば、フランス語のファイルは `fra.traineddata` です。
2. ダウンロード済みのトレーニングしたデータ ファイルを **ProgramData** フォルダの `Kofax RPA\[バージョン]\lib\tessdata` にコピーします。例：

   `C:\ProgramData\Kofax RPA\11.0.0_110\lib\tessdata`

3. 通知領域で Desktop Automation アイコンを右クリックし、[設定] を選択します。
4. [**OCR**] タブをクリックし、[デフォルトの **OCR** 言語] リストで言語を選択します。[保存して再起動] をクリックします。

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することができます。詳細については、Tesseract のトレーニングを参照してください。

仮想入力ドライバーをアクティブにする

仮想入力ドライバーは、ハードウェア キーボードをシミュレートできる Windows デバイス ドライバーです。ドライバーのオペレーティング システムでサポートされているものについては、『Kofax RPA Technical Specification』(Kofax RPA 技術仕様書) を参照してください。ドライバーのインストールについては、『Kofax RPA Installation Guide』(Kofax RPAインストール ガイド)を参照してください。

ドライバーを使用するには、自動化されたデバイスで環境変数「`KAPOW_KEYBOARD_INPUT_METHOD`」を「`VIRTUAL_KEYBOARD`」に設定します。仮想入力ドライバーの使用をキャンセルするには、環境変数を削除します。

ローカル Desktop Automation の使用

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、Desktop Automation のプロセスが迅速で容易になります。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 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 Installation Guide』(Kofax RPA インストール ガイド) を参照してください。

   ヒント 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. Desktop Automation サービスの設定の説明に従って 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ラベル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 デバイスのマッピング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. Desktop Automation ロボットを作成します。
   a. **[**ファイル**]** > **[**新しい **Desktop Automation** 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。まず Web オートメーション 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 Desktop Automation ワークフローを編集できません。
4. 既存の Web オートメーション ロボットを開くか、**[**ファイル**]** > **[**新しいロボット**]** をクリックして新しい Web オートメーション 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、[**Desktop Automation** ワークフローの呼び出し] を選択します。
   b. [ワークフロー] ドロップダウン リストで、ステップ 4 で作成した Desktop Automation ロボットを選択します。
      同じタブで、入力値と出力マッピングを設定します。
      - [必要なデバイス] プロパティでプラス アイコンをクリックし、[スタティック 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 [実行の切り替え] 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。
   「**Desktop Automation** ワークフローの呼び出し」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。そのためには、ツールバーで [**DA** ロボットにステップ イントゥー] をクリックします。
   Desktop Automation 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。これで Desktop Automation ワークフローの設計を開始できるようになりました。ローカル Desktop Automation モードが有効であるという通知が表示されます。
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、[オートメーション ワークフロー] ビューで、消える要素の使用が含まれ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。

ヒント 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。

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、[オートメーション ワークフロー] で [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 Desktop Automation ワークフローに戻すには、そ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。

   注 Desktop Automation ステップの実行中、キーボードとマウスは自動的にブロックされ、誤って実行との相互作用が発生するのを防ぎます。ステップが完了するとブロックは解除されます。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。

ファインダー

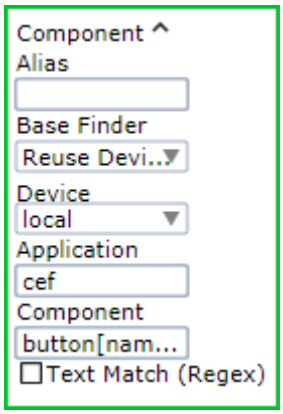
ファインダーは Desktop Automation の重要な概念です。ファインダーは、あるアクションの実行を行う エレメントを検索する方法を示します。例：

- 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を開くべきかがわかります。
- ボタンをクリックする場合に、ファインダーによってどこをクリックするべきかがわかります。
- また、テキストを抽出する場合に、ファインダーによってテキストを探す場所がわかります。

[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。

[オートメーション ワークフロー] ビューをクリックしてステップを直接挿入した場合は、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いつでも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

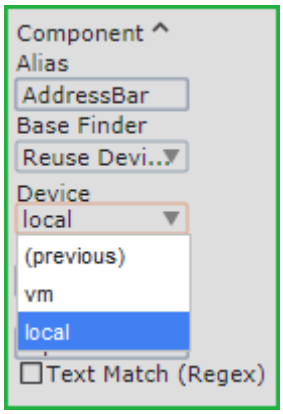
各ファインダーは下のものほど複雑で、より具体的な要素が得られます。

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、「Desktop Automation ワークフローの呼び出し」ステップの [アクション] タブにある [必要なデバイス] にリスト表示されます。

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。

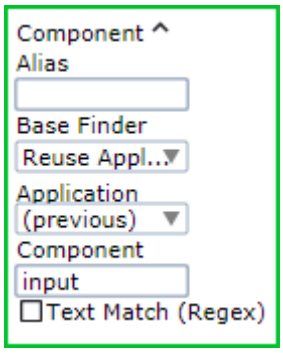


デバイス ファインダーは、開くなどでアプリケーションを開くデバイスを選択する際に使用します。

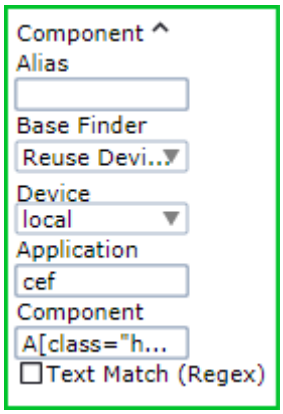
アプリケーション ファインダー

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。

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。

注 一部の構成では、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。

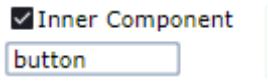
- デバイス
- 検出されたアプリケーション
- 検出されたコンポーネント

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。

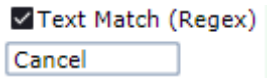
検出済みのコンポーネントを再利用する場合は、すべてのコンポーネントに検索を再実行するのではなく、このコンポーネントの内部ハンドルが使用されます。これにより、コンポーネントの位置が変更され、実行時間が短縮されている場合でも、コンポーネントのターゲット設定を正しく行うことができます。

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 [内側のコンポーネント] 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。



内側のコンポーネント 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 マッチ (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 マッチ (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 ボタンの両方が検出された場合、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。



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。

注 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。たとえば、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。

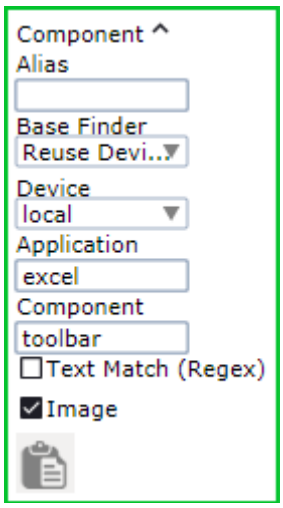
イメージ ファインダーはデフォルトでは使用しませんが、コンポーネント ファインダーを使用するあらゆるステップで、その代替となります。イメージ ファインダーの使い方：

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。

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。

イメージ セレクターには、次の 2 つの形式があります：シンプルなイメージ セレクター、および 9 グリッド セレクター (9 グリッド イメージ ファインダーで説明)。



重要 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
```

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、どのエレメントを選択すべき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、親子関係を示す不等号 (>) および祖先 - 子孫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。次の表は、サポートされている演算子とその動作の一覧を示します。

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
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 要素 (ドキュメントのルート)
E:nth-child(n)	E エレメント (親から n 番目の子)

パターン	意味
E:nth-last-child(n)	E エレメント (最後のエレメントから数えて、親の 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 番目の兄弟)
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 番目の兄弟エレメント)
E:first-child	E 要素 (親の最初の子)
E:last-child	E 要素 (親の最後の子)
E:first-of-type	E 要素 (同タイプの最初の兄弟)
E:last-of-type	E 要素 (同タイプの最後の兄弟)
E F	E エレメントの子孫 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

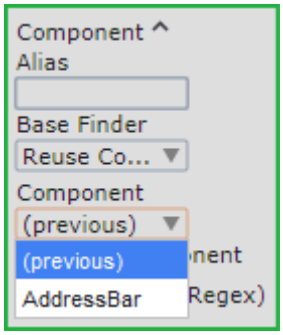
再利用において、ファインダーのコピーと貼り付けは最も脆弱な方法です。この方法は、空のファインダーでファインダーの作成を始めたくない場合には実用的です。

ファインダーをコピーするには、オートメーション ワークフロー表示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。



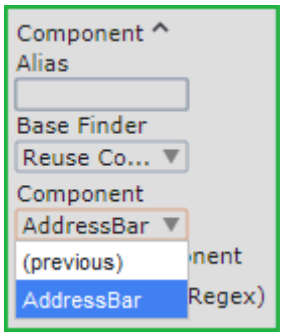
一連のファインダーで前のファインダーを使用している場合、その各ファインダーは、前のものではない、最初のファインダーの構成を共有していることになります。チェーン内のすべてのファインダーを編集するには、最初のファインダーを編集します。

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。

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。

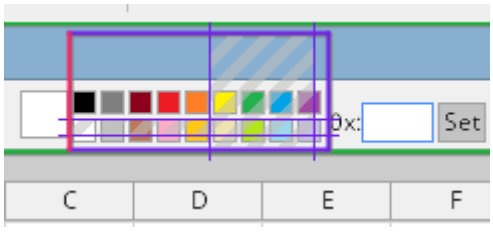


ファインダーが別のファインダーを再利用する場合は、そのファインダーの構成を共有します。名前付きファインダーを編集すると、そのファインダーを参照するすべてのファインダーに影響します。

9 グリッド イメージ ファインダー

9 グリッド イメージ ファインダーは廃止される予定です。このファインダーでロボットを使用し、9 グリッド イメージ ファインダーを新規作成できます。ただし、[オートメーション ワークフロー] ビューでファインダーを編集することはできません。Kofax では、このファインダーは今後のリリースで削除されるため、使用しないことを推奨しています。

9 グリッド イメージ ファインダーはイメージ ファインダーの一種で、レコーダー ビューでさまざまなサイズのエレメントを見つけるために使用することができます。このファインダーを使用して、主にボタンまたはテキスト フィールドを検索できます。このコンテキストにあるエレメントは、イメージ ファインダーを評価する際に検索されるサブ イメージになります。ファインダーを作成するときは、最大 9 個までのエレメントを定義して検索することができます。抽出に 9 グリッドのファインダーを使うと、中央のエレメントが抽出されます。ファインダーをマウス移動で使用する場合、ステップで定義されたオフセットに対して、ポインタが中央の要素に移動されます。



上の画像では、斜線がかかったセルはファインダー評価から除外されています。

前提条件

- 2 つの対角部、または上部および左側の工レメントといったように、中心の工レメントの大きさを計算することができるように十分な工レメントを定義します。
- 含まれる工レメントは、過不足なく照合され、ピクセル単位で検索されます。
- 各工レメントは、最低 1x1 ピクセルのサイズになります。

9 グリッド ファインダーの追加

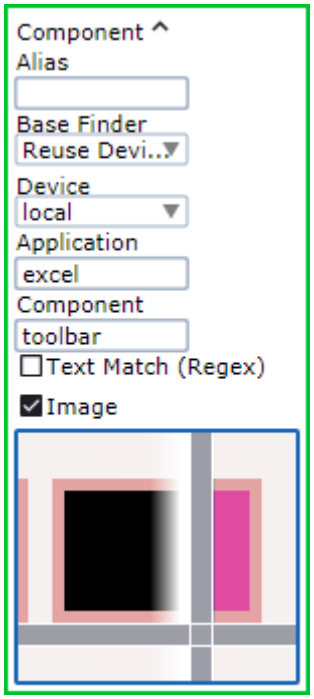
9 グリッド イメージ ファインダーを作成するには、レコーダー ビューのボックスをイメージ ファインダーに同様にドラッグし、イメージ ファインダーのトグル ボタン をクリックします。ファインダー内のバーの移動は、9 グリッド ファインダーの工レメントを変更するのに役立ちます。マウスを使ってクリックしたり、矢印キーを使ってバーをファインダーの内側に移動することができます。長方形を変更するには、辺をドラッグするか、新しい長方形を描きます。

ファインダー評価に工レメントを含めるには、工レメントを右クリックし、[検索時セルを含める] を選択します。工レメントを除外するには、セルを右クリックし、[検索時セルを含めない] を選択します。また、[Ctrl+Click] を使って、工レメントを含めたり除外したりすることもできます。含まれる工レメントは透明になり、除外された工レメントには灰色の斜線がかかります。デフォルトでは、9 グリッド ファインダーにはすべてのコーナー 工レメントが含まれることに注意してください。

シンプルなイメージ ファインダーに戻るには、イメージ ファインダーのトグル ボタンをもう一度クリックします。

ワークフロー ビューでの **9** グリッド ファインダーの操作

9 グリッド ファインダーを使ってステップを追加すると、オートメーション ワークフロー ビューでそのステップを確認できます。9 グリッド ファインダーは、ワークフロー ビューに含まれる工レメントのみを表示します。除外された工レメントは灰色です。ファインダー トグル内の工レメントをクリックすると、包含状態が切り替わります。



小さなエレメントの詳細がよく見えるように、また巨大なイメージが表示されないように、イメージは拡大および切り抜きされます。

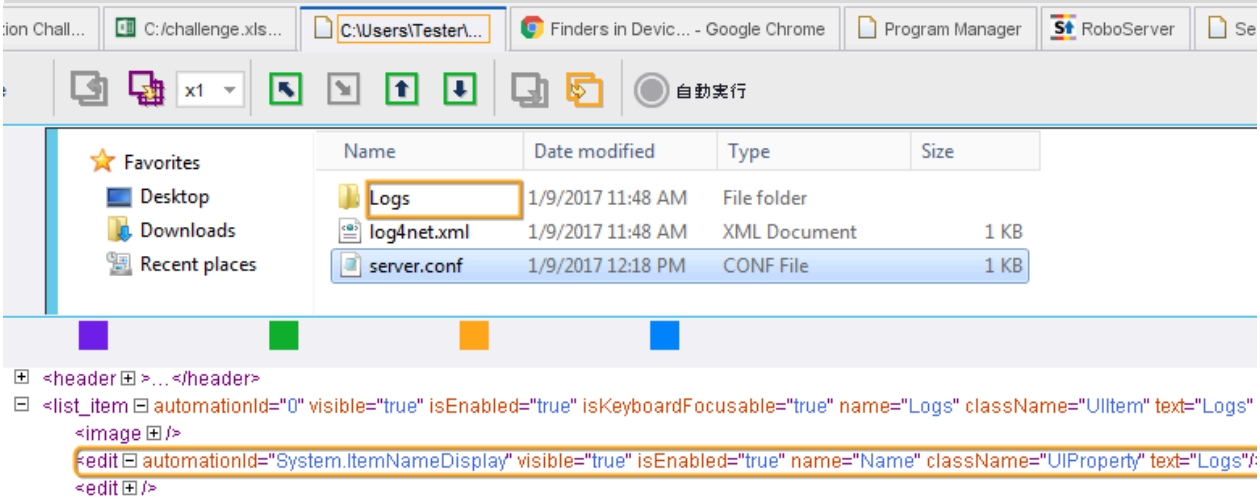
例: ファインダーの例

以下は、Windows エクスプローラのファイル リストで 1 つ以上のエレメントを見つけるためのファインダーの例です。

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

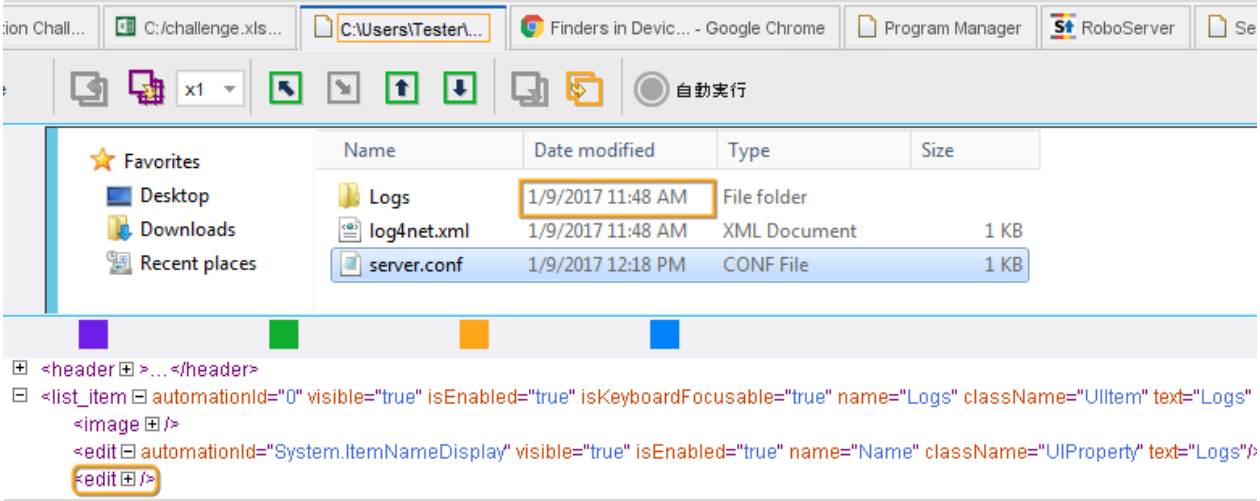
"list" の子である最初の **"edit"** エレメントの検索

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



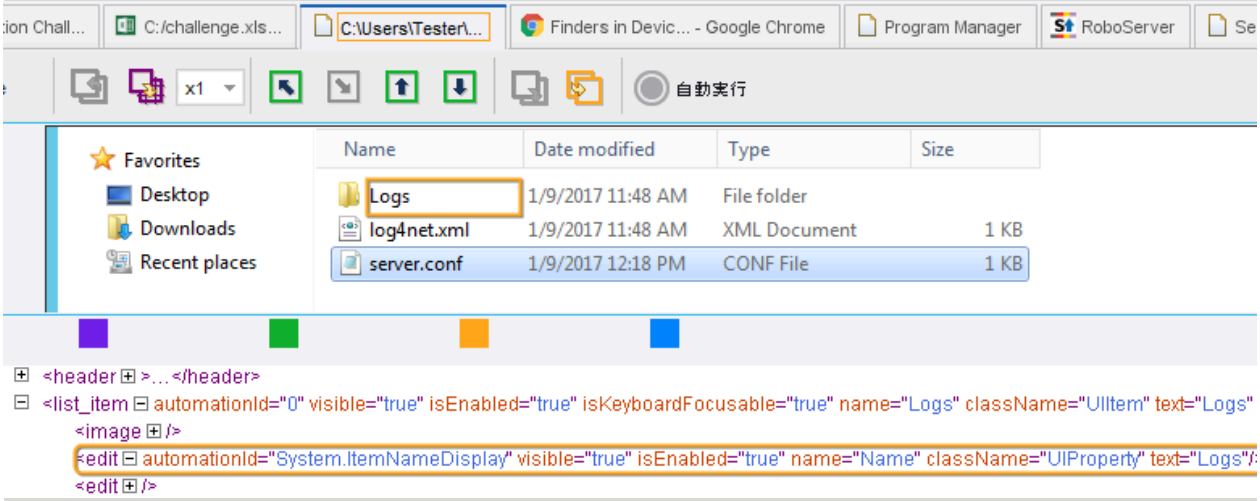
"list" の子である 2 番目の "edit" エレメントの検索

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



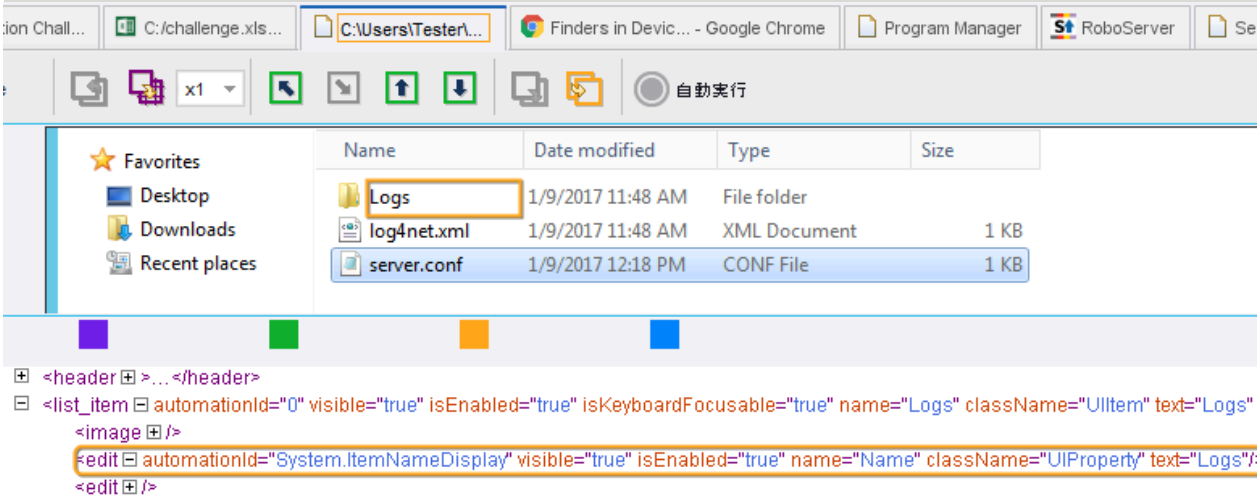
2 番目の子である "edit" エレメントの検索

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



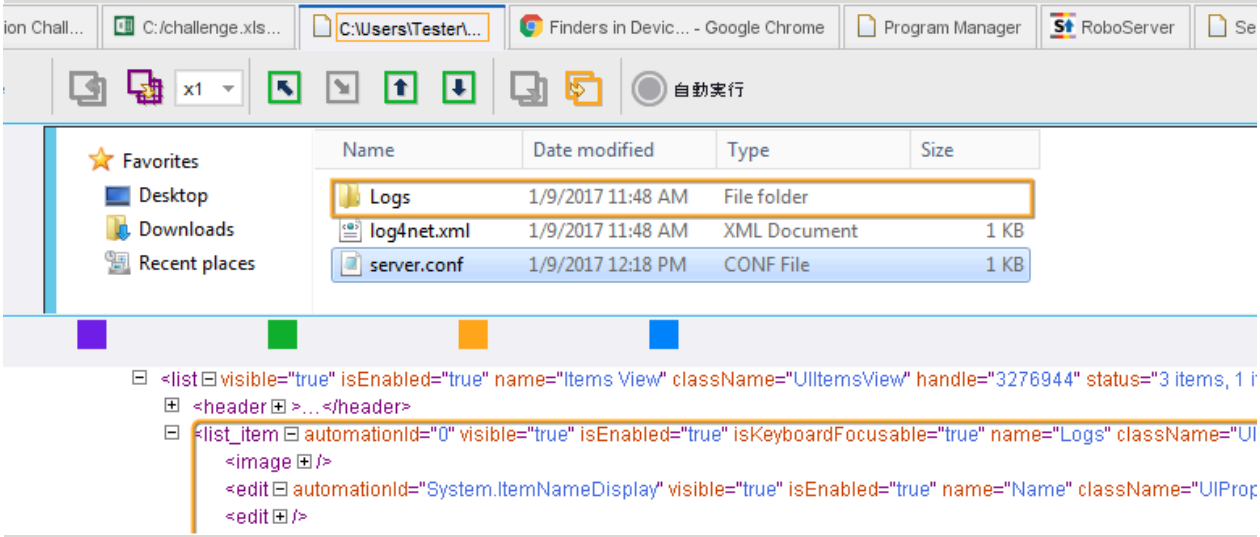
"list" の子孫である "edit" エレメントの検索

- セレクター：<space>
- ファインダー："list edit"



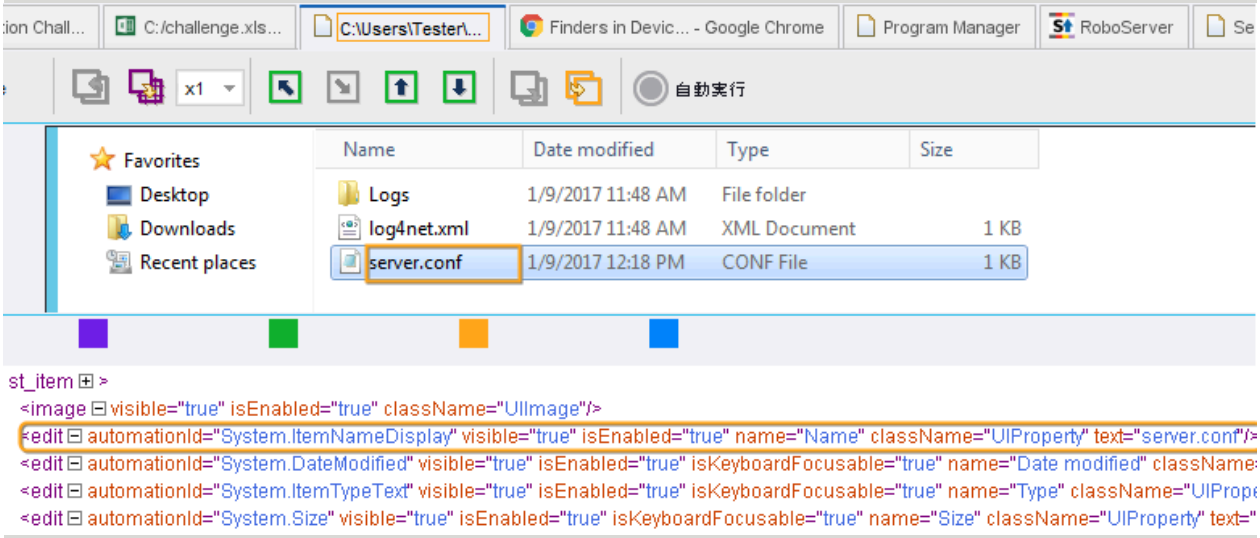
"list" の子である "list_item" エレメントの検索

- セレクター：>
- ファインダー："list > list_item"



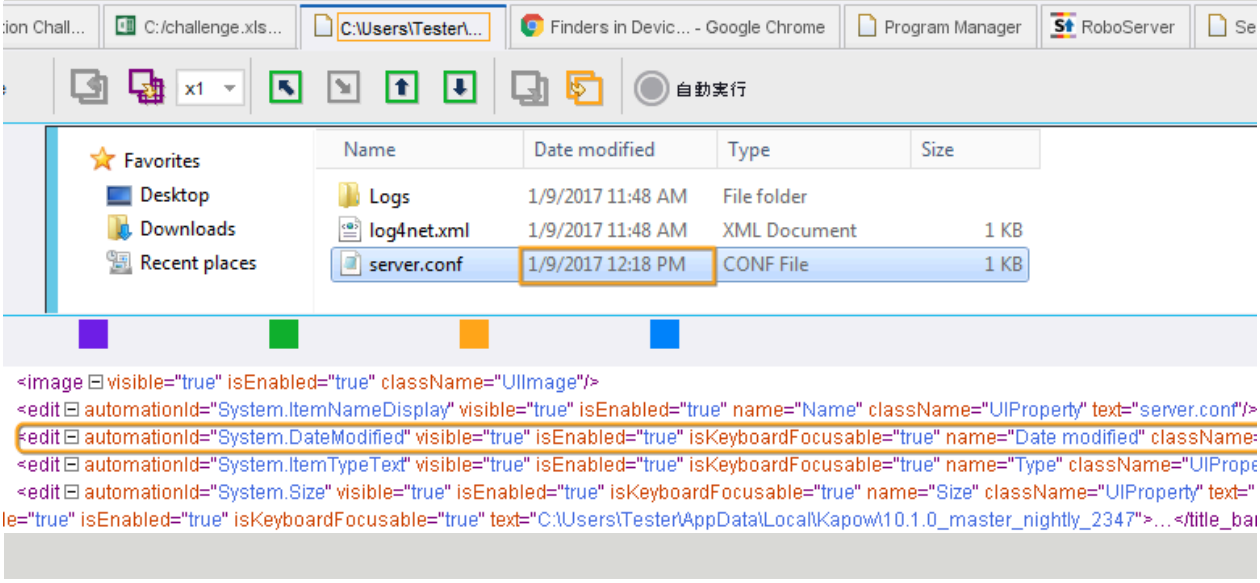
"server.conf" 値に "text" 属性を持つ"edit" エレメントの検索

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



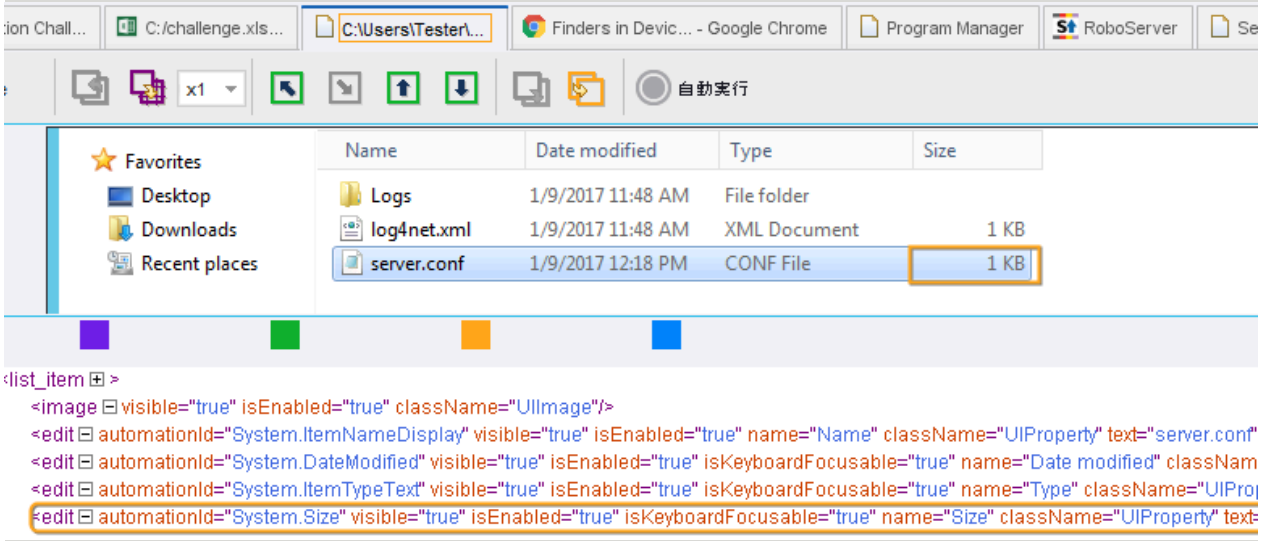
"server.conf" 値に "text" 属性を持つエレメントの直後にある "edit" エレメントの検索

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



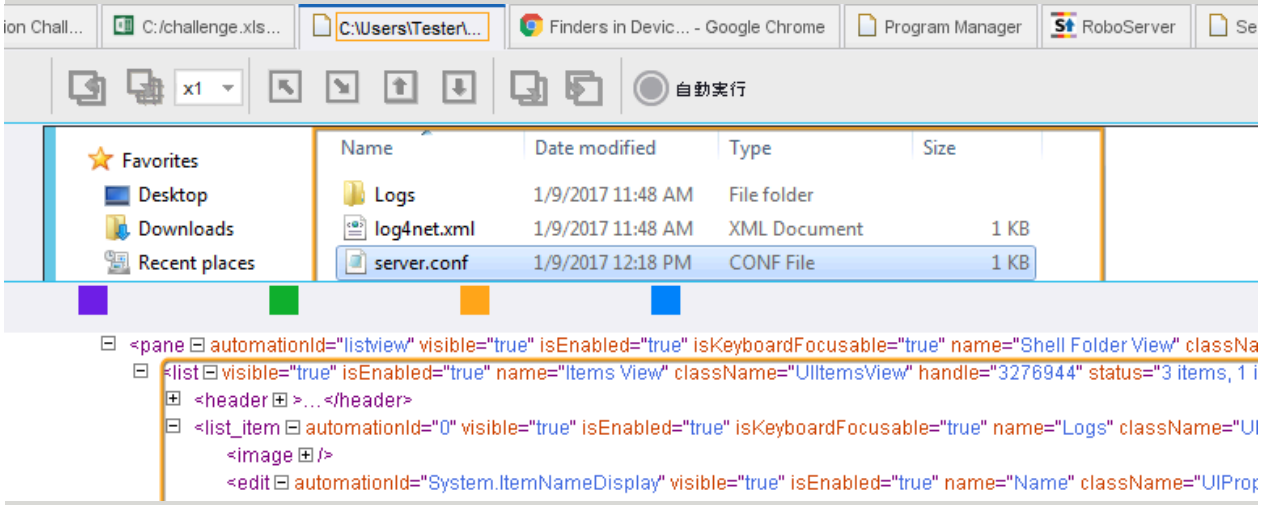
"server.conf" 値に **"text"** 属性を持つエレメントの後の任意の位置にある **"edit"** エレメントの検索

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



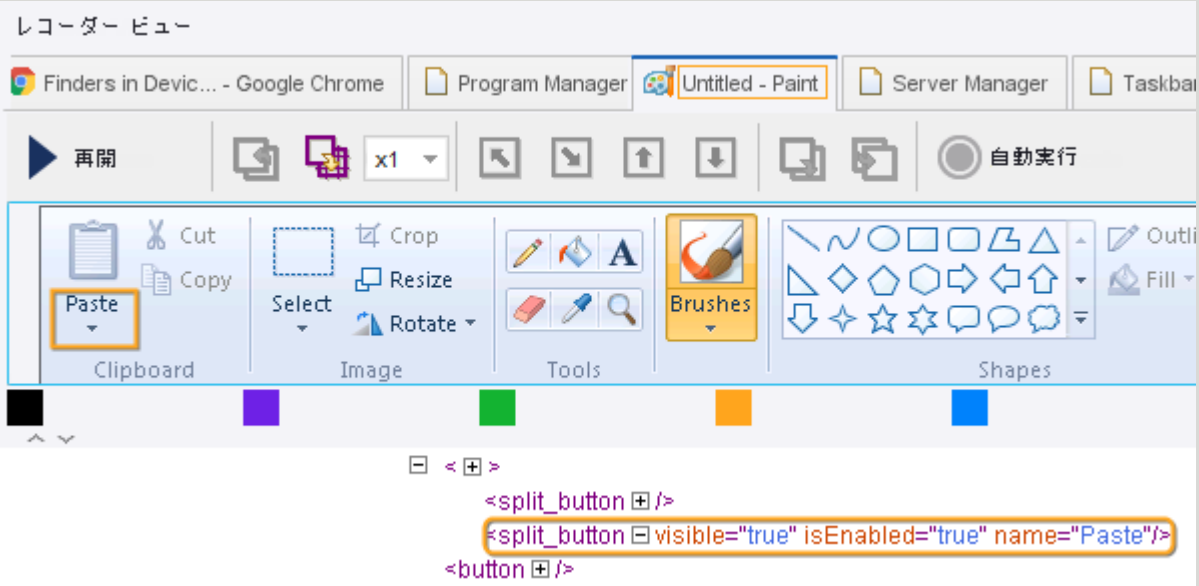
任意の位置にある最初の **"list"** エレメントの検索

- 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"]



ツリー モード

ツリー モードは、アプリケーションのツリー生成方法を定義するアプリケーション ベースの設定です。

[ツリー モード] オプションは、[オートメーション ワークフロー] ビューの上部にある [入力値]、[出力値]、[変数] のいずれかの設定、またはレコーダー ビューのアプリケーション タブのタイトルを右クリックして使用できます。

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 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特定のオプションを選択するのに役立ちます。

- [ツリーなし]: Desktop Automation は、アプリケーション ツリーにデータを追加しません。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。
- ISA: インテリジェント スクリーン オートメーションの略です。このオプションにより、プログラム インターフェースのグラフィカル表示からインターフェース エレメントを決定する、ユーザー イン

ターフェース (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 ツリーに複数のオプションを提供します。

インテリジェント スクリーン オートメーション (ISA)

ISA は、UI エレメント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。

注 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てい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できます。

認識された各 UI エレメントには、ファインダーで使用でき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が、選択を解除した際に座標が変化する場合などです。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	エレメント固有の属性	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。

UI エレメント名	ウィジェット名	エレメント固有の属性	注意
テーブル	`table`		テーブルは、アプリケーション ツリー内で別々のエレメントとして表されます。結合されたセルは認識されず、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 ボックスやテキスト ボックス、およびその他の 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`	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 ボックスのラベルやテキスト ボックスのラベルなどの UI 要素のラベル。 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場合は、Windows オートメーション API を使用する場合と同様に、値を抽出アクションを使用します。エレメントの値が ISA モードで認識されない場合は、画像からテキスト抽出アクションを実行してください。
- 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。
- 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することができます。詳細については、Tesseract のトレーニングを参照してください。

重要 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 は英語の UI を自動化できます。デフォルトの OCR 言語の変更と同様の手順を使用して、UI 認識言語を追加できます。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。

1. 必要な言語の `.traineddata` ファイルを https://github.com/tesseract-ocr/tessdata 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。
2. ダウンロード済みのトレーニングしたデータ ファイルを適切なフォルダにコピーします。
   - 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\tessdata`。例：

     `C:\Program Files (x86)\Kofax RPA DesktopAutomation 10.6.0 x32\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では、

     Kofax RPA のインストール ディレクトリの `nativelib\hub\windows-x32\[ビルド番号]\lib\tessdata*` にコピーします。例：

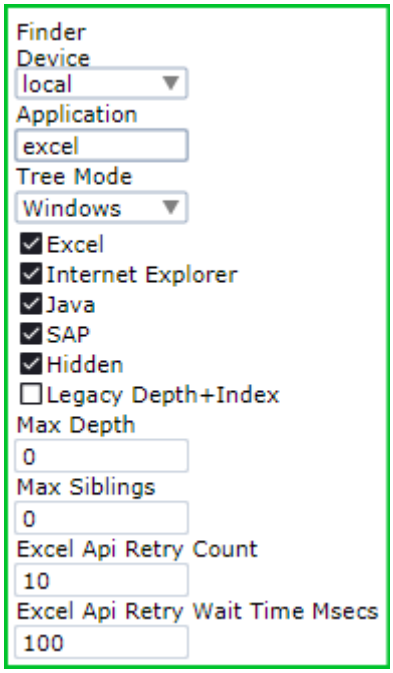
     `C:\Program Files\Kofax RPA 10.6.0.0 x64\nativelib\hub\windows-x32\622\lib\tessdata`
   - 組み込みブラウザを使用するローカルの Linux ベースのコンピュータでは、

     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_10.6.0.0_x64/nativelib/hub/linux-x64/533/lib/tessdata`

   * 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。
3. テキスト エディターで、`lib` ディレクトリにある `isa_v1.cfg` ファイルを開きます。
4. `ocr_language` パラメータで `eng` を `jpn` に置き換えるか、複数の言語を使用する場合は、`ocr_language=eng+jpn` などのプラス記号を使用して `jpn` を追加します。ファイルを保存して閉じます。
5. 変更を有効化するには、Desktop Automation サービスを再起動します。

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。また、Kofax RPA には、特定のアプリケーションに対する複数の拡張サポートが用意されています。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。Kofax RPA では、Excel のサポートについても組み込み Excel ドライバー 機能で拡張されています。

注 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。

Internet Explorer
Internet Explorer の拡張サポートを有効にして、DOM (Document Object Model) ツリーを取得することにより、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 Chromium ベースの組み込み Web ブラウザが用意されています。Internet Explorer の拡張サポートの使用を検討している場合、次のことに注意してください。

- 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:
  - ドキュメント オブジェクト モデル (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。
- Internet Explorer の拡張サポートが無効になっている場合、UI オートメーションを使用して Internet Explorer からオブジェクト ツリーを取得します (Microsoftの `Inspect.exe`を使用して取得するツリーと同じ)。

Java
状況によっては、Java Access Bridge が動作しないため、このオプションの選択を解除してレガシー モードに切り替えることをお勧めします。[Java] オプションは、Internet Explorer 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。

非表示
アプリケーションのツリー全体を抽出するかどうかを指定します。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、多くの工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工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」属性と「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 で作成された口ボットを使用する場合は、Kofax RPA 10.3 以降で動作する口ボットに [レガシーの深度とインデックス] 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、ネストされた工レメントの最大数を設定します。
- [兄弟の最大数]: 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、兄弟工レメント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 応答が遅い場合は、この回数を増やすことができます。

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。

Desktop Automation ステップ

このトピックでは、Desktop Automation ステップ アクションに関する情報を提供します。

- 割り当て
- 参照
- バンドル
- クリック
- 条件
- デバイスに接続
- コンポーネント セレクターのコピー
- コピー
- カスタム アクション
- デバイスからの切断
- Document Transformation
- テキストを入力
- Excel
- クリップボードを抽出
- 画像抽出
- 画像からテキスト抽出
- ツリーを XML として抽出
- 値を抽出
- ツリーの凍結
- グループ
- ガード チョイス
- セッションの開始
- KTA
- 繰り返しステップ
  - Break
  - 続行
  - ループ繰り返し
  - ループ
  - ループ中ステップ
- マウス移動
- 通知
- 開く
- キー プレス
- ファイルの読み込みステップ
- リモート デバイス アクション
- リターン

- スクロール
- クリップボード設定
- ターミナル
- Throw
- トリガー チョイス
- Try-Catch
- 記録されないインスタント クリック
- ウィンドウ
- ファイルの書き込みステップ

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。

プロパティ

- [名前]: ステップの名前。
- [変数]: 変数名。
- [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。

参照

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、「Desktop Automation の組み込みブラウザでの Cookie 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。

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.google.com
```

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、およびその他の情報に関する情報が含まれます。

受け入れ言語リスト
このオプションを選択すると、サーバーで使用可能な場合にブラウザが受け入れることができる言語を指定できます。デフォルト: en-US

読み込み状態を無視
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。

認証
Web ページ認証のユーザー資格情報を指定します。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。

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。

- [背景グラフィックス] : 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。

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、[オートメーション ワークフロー] ビューで、消える要素の使用が含まれ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。

注 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。

クリック

クリック ステップは、Desktop Automation で最も一般的に使用されるアクションです (端末を自動化していない場合)。クリック (および マウス移動) ステップを使用して、ロボットはプログラムの開始と終了、プログラム インターフェースの使用、ドラッグアンドドロップ操作の実行、テキストの選択、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。

プロパティ

アプリケーション
クリック ステップ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。

- [クリック]: インターフェースでクリックします。
- [押す]: マウス ボタンを解除せずに押します。
- [解除]: 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 [スタンダード ボタン] または [計算ボタン] を選択します。

- [スタンダード ボタン]: 左、中央、右
- [計算ボタン]: このオプションを選択する場合、仮想キー コードのマウス ボタン記号定数名を指定します。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。

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。

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
ブール条件。

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。

プロパティ

名前
ステップの名前。

デバイス
使用するダイナミック リファレンス名を選択します。このリファレンス名は、「 Desktop Automation ワークフローの呼び出し」ステップの [必要なデバイス] プロパティで指定されます。リストにはダイナミック 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する必要があります。

重要 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。Desktop Automation エージェントの設定を参照してください。

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。

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、オートメーション ワークフローのコンポーネント 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。

値の変換

[値の変換ステップ] を使用して、手動で指定した値を変換したり、抽出ステップと一緒に抽出した値を変換したりできます。

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の評価ステップ
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の評価ステップ] 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 ステップを保存
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。

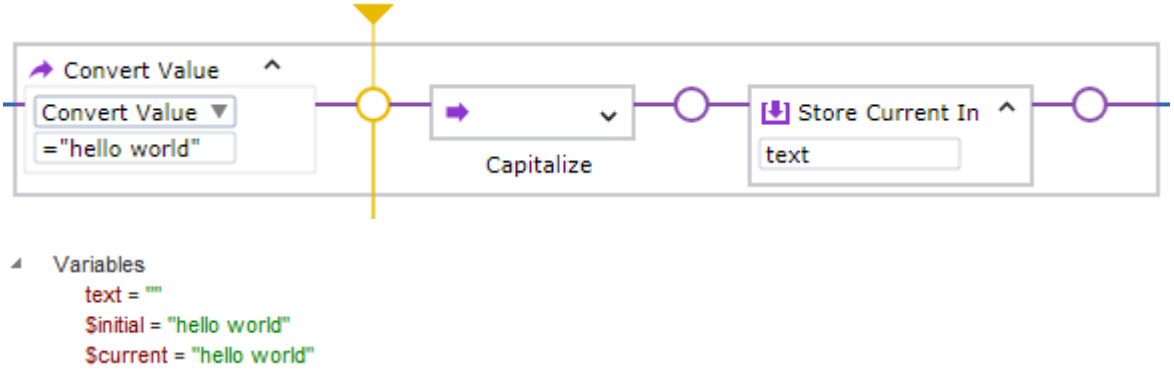
- 条件ステップ
- グループ ステップ
- Throw ステップ
- ログの書き込みステップ

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。

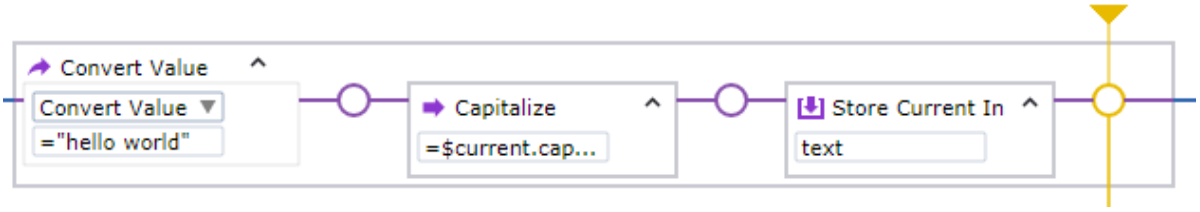
ヒント コンバータ グループ内でワークフロー状態の変更を元に戻す必要がある場合、または特定の先行フロー ポイントの状態を確認する必要がある場合は、必要な先行フローポイントを実行することにより、変換ステップのグループ内に戻ることができます。また、現在のフロー ポイントの前のワークフローを変更すると、ワークフローの状態が自動的に調整されるため、変換ステップを再実行する必要はありません。

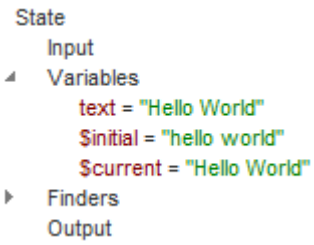
1. オートメーション ワークフローに値の変換ステップを追加し、次のように設定します。
   a. [値の変換] の下で**"hello world"**と入力します。
   b. エクスプレッションの後に、フロー ポイントを右クリックして、**[**エクスプレッションの評価ステップ**]** > **[Capitalize: (Text) -> Text]** をクリックします。
2. 最初の部分を実行すると、 **"hello world"**エクスプレッションの (指定された) 初期値が変数 **$initial** と **$current** に割り当てられます。



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 ステップ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。

カスタム アクション

[カスタム アクション] ステップは、Connector を使用して口ボット内からデータを処理するために、外部リソースの利用を可能にします。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。

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。

Connector の作成

Connector は、マニフェスト ファイルと追加リソースのディレクトリ ツリーを含む. zip ファイルとして配布されます。Kofax RPA は、.zipファイルを一時ディレクトリに抽出し、一時ディレクトリのコンテキストで口ボットの要求を実行します。口ボットの完了後、一時ディレクトリは削除されます。

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。パラメータは口ボット設計者に提示され、口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。

Connector を Management Console にアップロード

Design Studio の [カスタム アクション] ステップを使用してロボットの設計を完了すると、Connector の拡張子を `.zip` から `.connector` に変更して、プロジェクトに追加します。プロジェクトが Management Console にアップロードされると、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。 Connectors は次に RoboServer および必要に応じて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されます。

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、使用されない場合、リソースを必要としません。各 Connector は独自の python または node.js インスタンスを受け取り、このタイプの Connector を必要に応じて異なるバージョンで複数実行できます。

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。

Windows の場合、Kofax RPA は CMD シェルを使用します。Linux の場合、Kofax RPA は bash シェルを使用します。

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。

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の両方を受け取ります。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。

Python 2 と Python 3 の言語が異なるため、使用するメジャーな Python バージョン用にパッケージを構成する必要があります。Python 2.7 および Python 3 バージョン 3.5 以降のみがサポートされています。`6` つのパッケージが Python のインストールに存在している必要があります。

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`」
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。

フィールド	ステータス	説明
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。サポートされる値は、Python 2.7 では 2、Python 3.x では 3 です。 デフォルト: `3`

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。次のいずれかでなければなりません: 「`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。
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。
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。

フィールド	ステータス	説明
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`
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、次のいずれかでなければなりません。「`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、次のいずれかでなければなりません。「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。

レスポンスの定義

プログラム

- 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ロ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、ロ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

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ロ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、ロ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、結果をロ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、ロ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、ロ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、ロ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。

タイプ	パス	説明
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、ダイナミック 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 でドキュメントの区切り機能を使用すると、Kofax RPA は DT ブラウザ でナビゲートできる複数のドキュメントを受信できます。詳細については、Document Transformation ワークフロー を参照してください。

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 6.3.1 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。

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。以下、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 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。

ドキュメント ソース

ロ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。ロ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。
- ロボット 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[ロボット ファイル システム タブ]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へのパスを指定すると、DT 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。

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:8082
```

コールバック **URL**
ドキュメントの検証後に呼び出すシン クライアント サービスの REST ロボット URL 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、この URL は Management Console でロボットを起動するために使用されます。URL には、実行するロボットのパスとともに、Management Console アドレスが含まれている必要があります。URL は次のようになります。

```
http://localhost:8080/ManagementConsole/rest/run/Default project/
binaryInputAndWait.robot
```

指定のロボットに対して [**REST**] ボタンをクリックすると、Management Console で有効なコールバック URL を見つけることができます。このようなロボットには、バイナリ タイプの `doc` という属性を持つ `document` と呼ばれる入力変数があります。ロボットが呼び出されると、ドキュメントの `doc` 属性には、変換されて検証された文書が含まれます。Management Console でログインのためのクレデンシャルが必要な場合は、URL に次のように指定します。

```
http://user:password@localhost:8080/ManagementConsole/rest/run/Default
project/binaryInputAndWait.robot
```

詳細については、ロボットの実行のRESTを参照してください。

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 データを指定して構成するには、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**] をダブルクリックします。
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配布されます。

- 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。

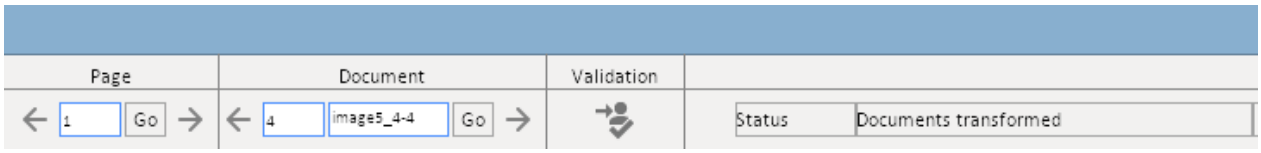
認識言語をデフォルト (英語) から変更するには、以下の手順を実行します。

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、[デフォルトのプロジェクト] 定義を選択します。
8. スクロール ダウンんして、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。

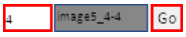
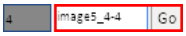
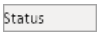
カスタマイズ済みのプロジェクト
このオプションを選択するときは、ドキュメントを処理するプロジェクトのパス (`c:\rpa\ocr` など) を [プロジェクト名] プロパティに指定します。プロジェクト リンクは、Design Studio を実行しているコンピュータではなく、Document Transformation ホスト上のローカルにアクセス可能なフォルダでなければなりません。

DT ブラウザ

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
[ページ] セクションでは、複数ページのドキュメント内を移動することができます。	

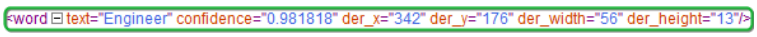
ボタン	説明
	複数ページのドキュメントで 1 つ前のページに戻ります。
	複数ページのドキュメントで 1 つ先のページに進みます。
	複数ページのドキュメントで指定のページに移動します。
[ドキュメント] セクションでは、複数または分割されたドキュメント内を移動できます。	
	前のドキュメントに移動します。
	次のドキュメントに移動します。
	番号で指定されたドキュメントに移動します。
	名前で指定されたドキュメントに移動します。 • 通常のドキュメントの場合は、ファイル名を拡張子なしで指定します。例：`mydocument.pdf` ファイルの場合、`mydocument` で指定。 • 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。
[検証] セクションは、ドキュメントの検証に役立ちます。	
	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)。

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 Desktop Automation ロボットに戻され、レコーダー ビューの Document Transformation Browser で開かれます。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、DT 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。



`der_x` などの派生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 URL で指定されたロボットの引数として使用されます。

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。

注 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 (Desktop Automation サービスの設定で「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テキストを入力ステップは、テキストの入力にこのドライバーを自動的に使用します。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。

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`)。

Excel

このステップを使用すると、組み込みの Excel ドライバーを使用して、Excel スプレッドシートでいくつかの操作を実行できます。詳細については、組み込み Excel ドライバーを参照してください。

プロパティ

ファイルを作成
このオプションを選択すると、新しい Excel ドキュメントを作成できます。

ファイルを開く
このオプションを選択すると、既存の Excel ドキュメントを開きます。[ファイルを開く] を選択した後、[ワークシート パス] でスプレッドシートへのフル パスを指定します。例：`c:/documents/myspreadsheet.xlsx`

クリップボードを抽出

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。

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の評価ステップ
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の評価ステップ] 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 ステップを保存
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
- Throw ステップ
- ログの書き込みステップ

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、Desktop Automation ワークフローでのブラウザの使用時に使用されます。

[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。

> 注 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。

イメージ ファインダーについて、詳しくは9 グリッド イメージ ファインダーを参照してください。

画像からテキスト抽出

このステップでは、テキストを画像から抽出します。Kofax RPA では、Tesseract OCR エンジンを使用して、テキストをイメージからキャプチャします。英語はインストールに含まれています。詳細については、デフォルトの OCR 言語の変更を参照してください。

注 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。

  閾値デルタ
  なし

  Positive
  - Small
  - Medium
  - Large

  Negative
  - Small
  - Medium
  - Large

テキスト認識の微調整

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

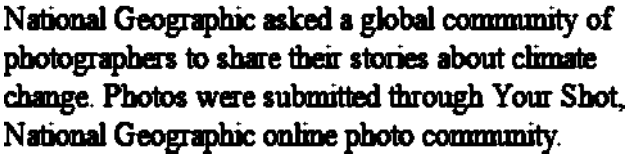
よくない場合、[画像の二値化] オプションで [カスタム] に変更し、許容可能な結果が生成されるように [閾値デルタ] オプションを調整できます。

以下は、認識のために画面からコピーされた画像です。

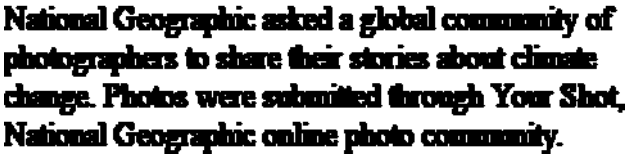
National Geographic asked a global community of
photographers to share their stories about climate
change. Photos were submitted through Your Shot,
National Geographic online photo community.

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。

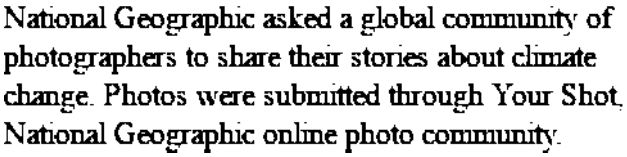
閾値デルタ：なし


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。

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。

プロパティ

派生属性を含める
派生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標である場合は、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 [派生属性を除外する] を選択するか、ワークフロー ビューのステップで [派生属性を含める] オ

プションをクリアします。派生属性の詳細については、Desktop Automation 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の評価ステップ
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の評価ステップ] 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 ステップを保存
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。

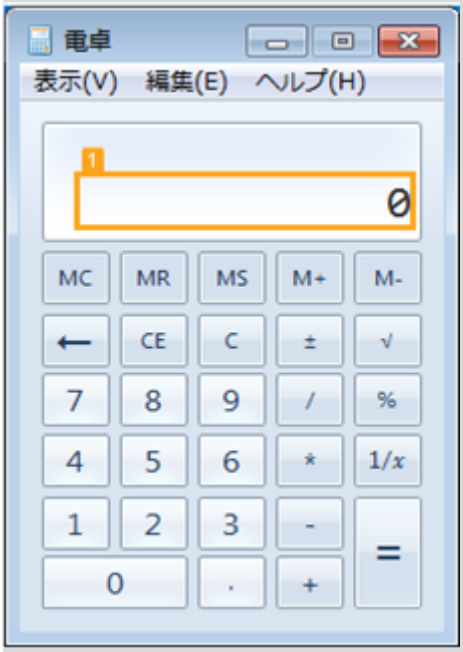
ツリーを XML として抽出ステップには、次の一連のアクション ステップを含めることもできます。

- 条件ステップ
- グループ ステップ
- Throw 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。

注 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。




```
<image ⊞ />
<text ⊞ />
<text ⊞ />
<text ⊟ automationId="150" visible="true" depth="3" isEnabled="true" name="Result" index="3" className="Static" handle="131474" text="0 "/>
<text ⊞ />
<button ⊞ />
<button ⊞ />
<button ⊞ />
```

以下は、変数にエクスポートされた XML 文字列です。

```
State
    Input
⊟   Variables
    ⊟  text
            "<text depth ="3" index ="3" handle ="131474" visible ="true" name ="Result" text ="0 " className ="Static" automationId ="150" isEnabled ="true" />"
⊞   Finders
⊞   Output
```

値を抽出

このステップでは、異なるエ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。
- [派生した属性]: 選択した派生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子エレメントからテキストを抽出します。すべての子エレメントからテキストを抽出するには、[すべての子を含める] を選択します。

エクスプレッションの評価ステップ
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の評価ステップ] 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 ステップを保存
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。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。

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
- Throw ステップ
- ログの書き込みステップ

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。

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ース 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。

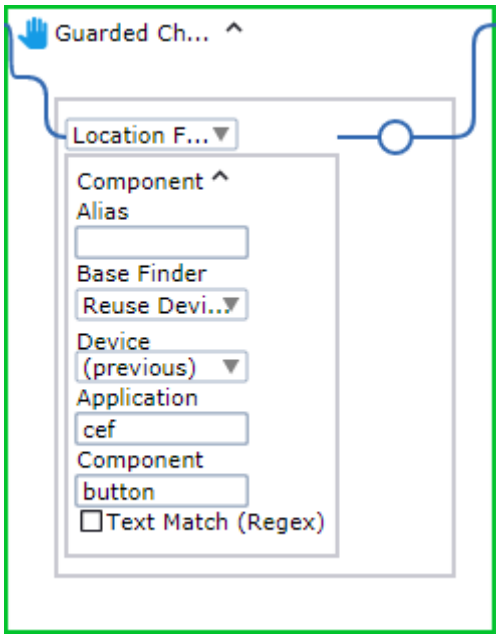
特定の秒数が経過したとき
これは、ロボットで次のステップを実行する前に指定された時間待機するタイムアウト ガードです。

注 デフォルトでは、以下のステップが [アプリケーション ビュー] で追加されると、ステップに 60 秒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、テキスト抽出、コンテンツ抽出、画像抽出、画像からテキスト抽出、およびキー プレス。

アプリケーションが見つかった場合
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。

アプリケーションが見つからなかった場合
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。

ロケーションが見つかった場合
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。



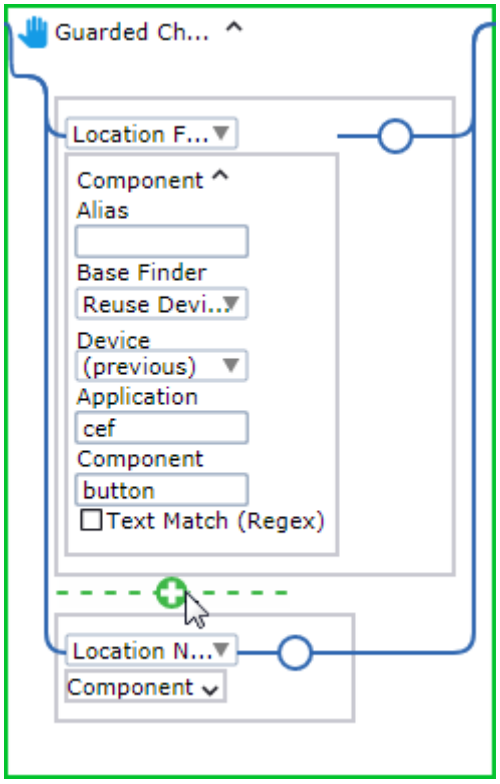
ロケーションが見つからなかった場合
エレメントが指定したファインダーで見つからないようにするロケーション ガード。

除去されたロケーション
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 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。

注 このヒントはパスワード フィールドに適用されません。

セッションの開始

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。

注 セッションの初期化ステップは、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。

注 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。

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。

注 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。

KTA

このステップでは、Kofax TotalAgility (KTA) サーバーに接続し、新しいジョブを開始したり、ジョブ ステータスを取得したり、新しいドキュメントを作成したりできます。

重要 RPA を使用するには、KTA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。ジョブの作成、ジョブステータスの取得、ドキュメントを作成。

KTA 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**]** タブに移動します。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、KTA 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、KTA 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 KTA カテゴリを選択します。該当するカテゴリの KTA プロセスのドロップダウン リストが表示されます。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。KTA 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使用したくない場合、こ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 は特定のジョブのステータス情報を提供します。ジョブは、ジョブの作成ステップから返されたステータスなど、ジョブ ID によって識別されます。ステータス情報は、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

ドキュメントを作成
このアクションは、KTA でドキュメントを作成します。ドキュメント ID が口ボットに返されます。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、KTA 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

ドキュメント検証の例

次の手順では、PDF ドキュメントを検証します。

1. Read File ステップを使用して、ドキュメントを `DocData` 変数にロードします。
2. KTA ステップで [ドキュメント作成アクション] を使用して、KTA データベースにドキュメントを保存します。以下のオプションを記入します。他のオプションはKTA 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 (.pdf ドキュメントの`application/pdf` など) を指定します。
   - ドキュメント**ID**: 参照できるように、KTA がドキュメントに割り当てた ID。
3. KTA ステップで [ジョブの作成] アクションを使用して、ドキュメントを検証できるプロセスを KTA で開始します。以下のオプションを記入します。
   - プロセス: KTA 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。
   - **NMDDOCUMENT**: KTA プロセスの単一のパラメータ (ドキュメントの ID を渡します)。
   - ジョブ **ID**: 参照可能できるように、KTA がドキュメントに割り当てた ID を指定します。
4. このステップは、[ジョブ ステータスの取得] ステップを追加して、ジョブが成功することを確かにします。以下のオプションを記入します。
   - ジョブ **ID**: KTA ステップの [ジョブの作成] アクションの ID。
   - 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 ステータスの取得] ステップは「評価の準備待ち」ステータスを返し、KTA 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 ステータスの取得] ステップを実行した後に「完了」ステータスが返されます。

繰り返しステップ

このセクションでは、Desktop Automation ワークフローの繰り返しステップについて説明します。

繰り返しステップには、繰り返しステップ、ループ中ステップ、ループ繰り返し ステップの 3 つがあります。

以下はすべての繰り返しステップに共通です。

1. すべての繰り返しステップには、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、繰り返しステップを終了できます。
3. 続行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
すべての繰り返しステップには、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。
- 変数が繰り返し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。

Break

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。

続行

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。

ループ繰り返し

繰り返しループ 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。[範囲ファインダー] 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。[範囲ファインダー] 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、繰り返しループ ステップの中で一意となる名前を持つ必要があります。

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、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。

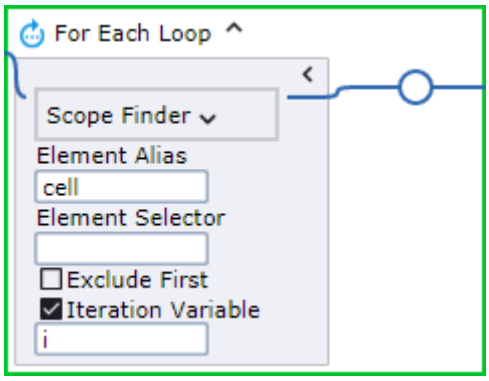
相対セレクターは、ループするエレメントを検出するために使用されます。このセレクターは、別のセレクターと結合されて実際の新しいセレクターを形成するようになっているため、相対セレクターと呼ばれます。スコープ ファインダーにセレクター `"DIV[class="someClass"]"` があり、要素ファインダーのセレクターが `"> DIV"` の場合、結合セレクターは `"DIV[class="someClass"] > DIV"` のようになります。反復するエレメントを検出するために使用される実際のファインダーはスコープ ファインダーと本質的に同じですが、セレクターがこの新しい結合セレクターに置き換えられます。

繰り返しループ 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、エレメント 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。

エレメント 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、繰り返しループ 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

ループ繰り返しステップは、他のステップと同様にオートメーション ワークフロー ビュー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便利です。[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。

- すべての兄弟
- 各 <タグ名> 兄弟
- 各 <タグ名> (<レベル>) 祖先
- 各テーブル行

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繰り返しループ ステップを作成するのに役立ちます。Design Studio では、繰り返しループ 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、繰り返し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。

すべての兄弟
このメニュー項目は、選択されたタグのすべての兄弟をループする繰り返しループ 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>** 兄弟
このメニュー項目は、選択されたタグのすべての兄弟を同じタグ名でループする繰り返しループ 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>)** 祖先
このメニュー項目は、Desktop Automation ワークフロー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祖先ノードを検索します。そうすることで、自

身に似た兄弟ノードが複数あるノード、または以下のタグ名の工レメントを持つ祖先ノードを探します。"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、選択された工レメントのいくつ上のレベルに実際のループ 工レメント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、実際のループ 工レメントのタグ名です。

各テーブル行
このメニュー項目は、Desktop Automation ワークフロー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繰り返しループ ステップを挿入します。

繰り返しループ ステップの使用

繰り返しループ ステップを使用する際には、考慮するべき事柄が何点か必要があります。

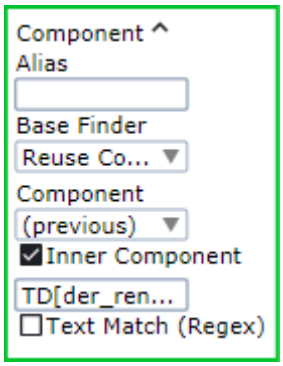
イテレーションのスキップ
一部の初期ノードや各 2 番目のノードなど、一部の工レメントをループ中にスキップしたい場合は、ループ内の最初のステップとして、テストが true である場合に継続される条件ステップを挿入します。たとえば、次の条件では、ループはすべての偶数回の繰り返しをスキップします。

```
=i % 2 == 0
```

繰り返しステップ内のファインダー

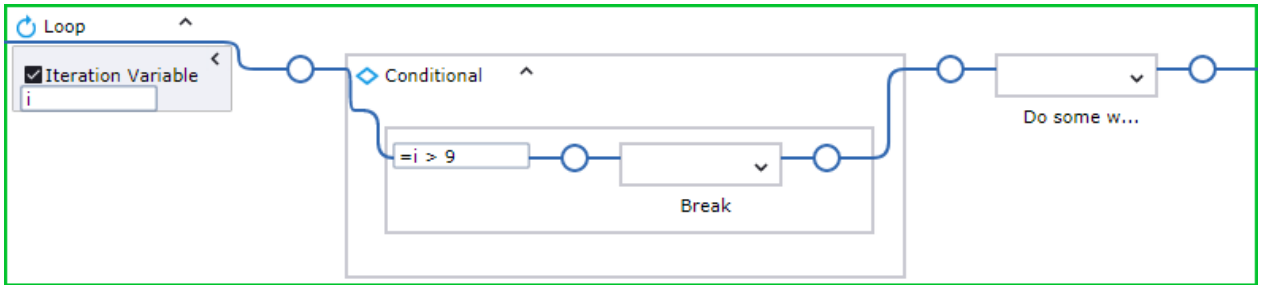
Kofax RPA は、検出されたエレメントに相対的な要素を自動的に検出します。レコーダー ビューで要素を右クリックしてアクションを挿入するときに、この要素が検出された指定エレメント内にある場合、生成されたファインダーは、次のように検出された要素に対して相対的になります。

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、エレメント ファインダーに付けた名前によって異なります。



ループ

繰り返しステップは、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続行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。

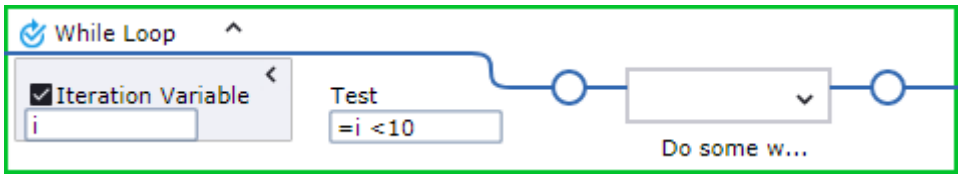


[次のイテレーションに移動] ボタンを押して、同じフロー ポイントに再度達するまで実行します。フロー ポイントが一部のイテレーションでスキップされた場合、ループは複数回実行できます。これ以上イテレーションがなくなると、実行は繰り返しステップ外のフロー ポイントで停止します。

ループ中ステップ

ループ中ステップは、追加のプロパティを持つ以下の繰り返し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ループ中ステップです。



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。レコーダー ビューからクリックを追加すると、マウス移動ステップがクリック ステップの前に自動的に追加されます。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。

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 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。

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

アプリケーションをオートメーション デバイスまたはローカルで開きます。たとえばヘッドレス端末は、そのドライバがローカル デバイスで有効になっている場合、ローカル デバイスで開かれます。

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、組み込み Excel ドライバー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`

  注 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。

注
- 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、Desktop Automation 組み込みブラウザで Cookie を使用する場合は [開く] ステップを使用します。詳細については、「Desktop Automation の組み込みブラウザでの Cookie 管理」を参照してください。

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。
- Desktop Automation 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 (Desktop Automation サービスの設定で「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 ステップは、テキストの入力にこのドライバーを自動的に使用します。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。

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
[スタンダード キー] または [計算キー] を選択します。

- [スタンダード 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。
- [計算キー]: 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` 計算キーは、キリル文字 Ӂ 用です。`0x04c1` コードは Ӂ 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。

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`)。

ファイルの読み込み

ファイルの読み込みステップは、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。

注 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 設定ウィンドウの [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。

使用中の 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、ローカル デバイス上で [ファイル アクセス] を [ダイレクト アクセス] に設定してこのステップを使用することはできません。ただし、[ファイル アクセス] が [**RFS** 経由] に設定されているローカル デバイス上では使用が可能です。リモート デバイス上では、[ダイレクト アクセス] と [RFS 経由] の両方が使用できます。

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、「Desktop Automation ワークステーションの呼び出し」ステップの [必要なデバイス] プロパティで指定されます。
- [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、[ダイレクト 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

- [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。
- [シャットダウン]: 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。
- [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。このアクションでは、パラメータとしてパスワードが必要になります。詳細については、スクリーン ロックの使用を参照してください。
- [マシンの再起動]: Desktop Automation サービスを実行中のコンピュータを再起動します。
- [マシンのシャットダウン]: 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。

スクリーン ロックの使用

オートメーション デバイスの操作中、コンピュータ スクリーンのロックが必要になる場合があります。Desktop Automation サービス メニューの [スクリーン ロック] コマンドを使用して、スクリーンをロックすることができます。デバイス スクリーンをロックする前に、サービスが実行中で、接続状態になっていることを確認します。スクリーンをロックするには、Desktop Automation サービス アイコンを右クリックして、[スクリーン ロック] を選択します。

ユーザーがログインしたときに別の画面を表示するように Windows が設定されている場合、スクリーン ロックは [OK] が押されたときにこの別の画面を検出して閉じようとします。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。別の画面が検出されると、スクリーン ロック機能はシステムとの接続が確立されてから 3 秒後に画面を閉じます。自動検出が失敗する場合、または 3 秒では十分でない場合は、自動化されたデバイスの環境変数に KAPOW_LEGALNOTICE_SECONDS システム変数を追加し、接続後にウィンドウを閉じるまで待機する秒数を [変数値] フィールドに設定します。変数を追加した後、Desktop Automation サービスを再起動します。

スクリーン ロックの使用前提条件
Desktop Automation でスクリーン ロック機能を使用するには、デバイスが次の要件を満たす必要があります。

- リモート デスクトップが有効になっていること。
- Desktop Automation サービスを実行しているユーザーは、リモート デスクトップ経由での接続が管理者グループまたはリモート デスクトップ グループのメンバーとして許可されており、パスワードを使用する必要があります。
- 有効なグループ ポリシーで **[**コンピュータの設定**]** > **[**管理用テンプレート**]** > **[Windows** コンポーネント**]** > **[**リモート デスクトップ サービス**]** > **[**リモート デスクトップ セッション ホスト**]** > **[**セキュリティ**]** [接続時に常にパスワードの入力を求める] がオフにされていること。
- ポート 3389 が開いていること。
- オートメーション デバイスをドメイン コントローラにすることはできません。

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 ウィンドウの [出力値] 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。このステップはロボットで必須です。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。

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、[出力値] 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。

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。

注 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。

クリップボード設定

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。

ターミナル

このカテゴリを使用して、ターミナルに接続して端末を自動化できます。詳細については、ターミナルの自動化 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、ターミナルの自動化 を参照してください。

Throw

このステップでは、例外をスローしてエラーを示し、それを Desktop Automation ワークフローで別の場所で処理します。

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、Try-Catchを参照してください。

スローされた例外は、[**Catch**] 分岐に指定された例外を含む最も近い Try-Catch によってキャッチされ、処理されます。そのような Try-Catch 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。そのような場合、ワークフローおよび含まれている「Desktop Automation ワークフローの呼び出し」ステップの実行が停止し、エラーは「 Desktop Automation ワークフローの呼び出し」ステップの [エラー処理] タブで指定されているように処理されます。

Throw ステップは通常、タイムアウト ガードと共に使用します。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、Throw 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、Try-Catch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 Throw 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 Try-Catch 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 ガードでいつアクションなしで 60 秒待機するかを知りたい場合、Throw ステップを「60 秒のタイムアウトが経過しました。」などのテキストと共にタイムアウト 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。

Throw ステップは Try-Catch ステップの Finally 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。

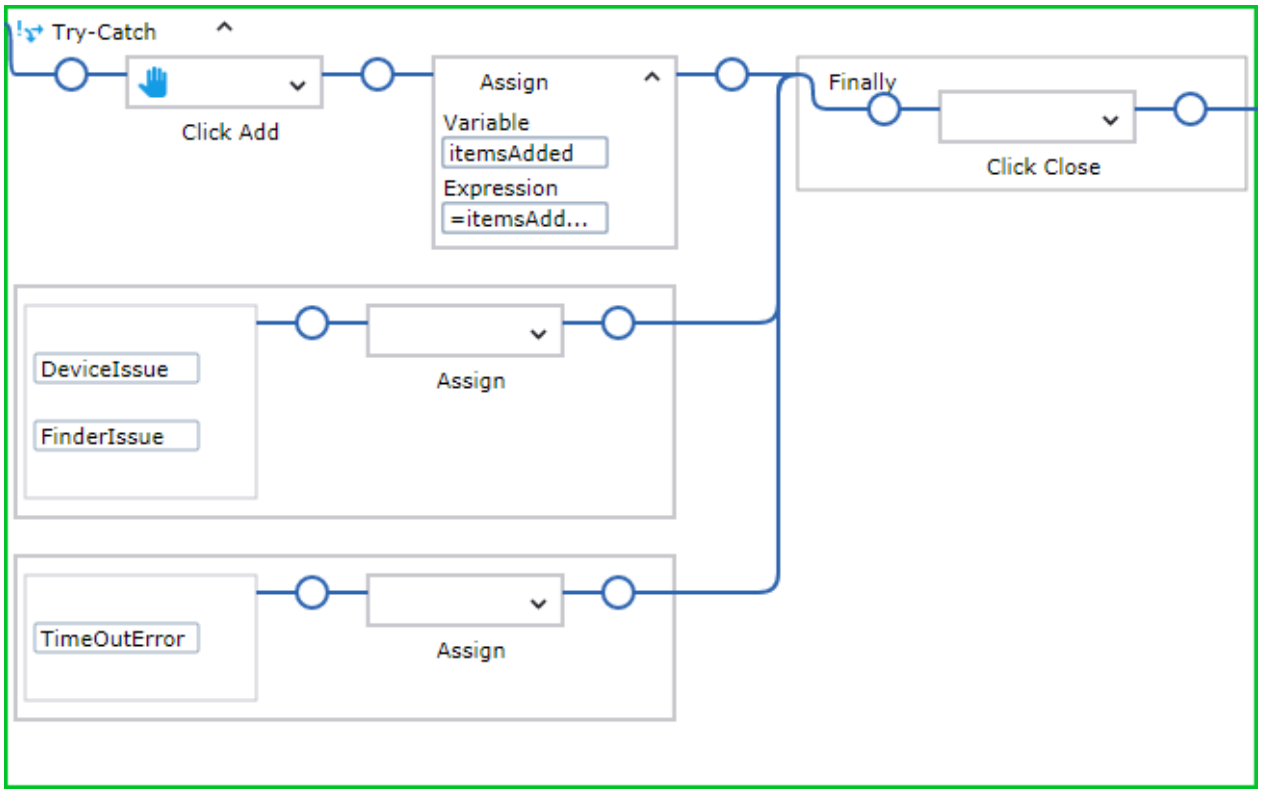
ホット キーを押す
指定されたキーまたはキーの組み合わせを押すと、ロボットにより、選択されたアクションが実行されます。トリガーの [名前] を指定し、リストから [キー] を選択します。また、Shift、Ctrl、Alt の 3 つのスタンダードなキー修飾子から選択します。

Try-Catch

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。

- Try 分岐：実行するアクションを指定する最上部分。
- Catch 分岐：Try 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 Catch 分岐を設定することができ、それぞれの Catch 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。
- Finally 分岐：実行するアクションを指定します。この分岐は、Try と Catch の実行結果に関係なく、常に最後に実行されます。

例外は、Throw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。



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、 Catch 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。

Catch 分岐はそれぞれ例外のリスト、およびその右側にある、Try 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、Throw 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。

Catch 分岐で例外が追加または編集されると、エディターでは Try 分岐内でスローされる可能性があり、かつ Catch 分岐にまだリストされていない例外が提案されます。

Finally
Try-Catch 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

Try-Catch 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 Finally 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、Try-Catch ステップの実行は、Try 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 : **Finally** 分岐が空
Try 分岐が正常に終了
実行は、Try-Catch ステップ全体の後のステップで続行されます。つまり、Catch 分岐はこの場合は実行されません。

Try 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 Catch 分岐の開始を直接続行します。

より複雑なケース : **Finally** 分岐が空
Try 分岐がスローされた例外で終了するが、**Catch** 分岐はその例外をリストしない
このケースは、Try-Catch 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。

> 注 この戦略 (「Try-Catch 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 Try-Catch ステップに空の Finally 分岐がある場合、ワークフロー ロジックで周囲の Try-Catch ステップの一致する Catch 分岐が検索され、どの Try 分岐にこの Try-Catch ステップが含まれているかが検索されます。そのような Catch 分岐が周囲の Try-Catch 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。そのような場合、ワークフローおよび含まれている「Desktop Automation ワークフローの呼び出し」ステップの実行が停止し、エラーは

「 Desktop Automation ワークフローの呼び出し」ステップの **[エラー処理]** タブで指定されているように処理されます。

Try-Catch ステップに Finally 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 "throw" が実行されます。

Try 分岐がスローされた例外で終了し、当該の **Catch** 分岐もスローされた例外で終了する
Catch 分岐でスローされた例外は同じ Try-Catch ステップの Catch 分岐では処理されません。代わりに、これは Try-Catch 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 **Try-Catch** ステップに関する注意
Try-Catch ステップによって処理される例外は、周囲の Try-Catch ステップによって処理されません。例外を処理できる Catch 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。Catch 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：**Finally** 分岐が空ではない
この場合、実行の状態に関係なく、Finally 分岐のステップは実行が Try-Catch 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。

Try 分岐が正常に終了
例外は Finally 分岐のステップで続行します。その後のことは、Finally 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。

- Finally 分岐の実行が正常に終了すると、実行は Try-Catch ステップ全体の後のステップで続行されます。
- 例外が Finally 分岐の実行中にスローされる場合、これは Try-Catch 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。

Try 分岐はスローされた例外で終了し、**Catch** 分岐は正常に終了する
Catch 分岐の実行後、ロジックは前のケースのとおりです。

Try 分岐がスローされた例外で終了するが、**Catch** 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 Finally 分岐のステップで続行されます。その後のことは、Finally 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。

- Finally 分岐の実行が正常に終了すると、実行は Try-Catch 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 Finally 分岐の実行時にスローされると、これは同じ Try-Catch ステップの Catch 分岐によって処理されません。代わりに、これは Try-Catch ステップ自体がその例外 (つまり、Finally 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。

Try 分岐はスローされた例外で終了し、当該の **Catch** 分岐もスローされた例外で終了する
これは、"remembered" の例外が Try 分岐ではなく、Catch 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、Try 分岐によってスローされた例外が Catch 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 Throw 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、例外がワークフローの Try-Catch ステップで処理されず、代わりに「Desktop Automation ワークフローの呼び出し」ステップの実行を終了する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 アクションの種類によっては、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`: ファインダーでエレメント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。

エクスプレッションがステップの一部であるときはいつでも、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
クリップボード設定	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
クリップボードを抽出	DeviceIssue
画像抽出	DeviceIssue、FinderIssue、エクスプレッションの問題、ExtractIssue

ワークフロー エレメント	例外
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
Try-Catch	なし
Break	なし
Throw	なし
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、Conversio
ガード	
数秒が経過したとき	エクスプレッションの問題、IncorrectValueIssue
アプリケーションが見つかった場合	エクスプレッションの問題、DeviceIssue、FinderIssue
アプリケーションが見つからなかった場合	エクスプレッションの問題、DeviceIssue、FinderIssue
ロケーションが見つかった場合	エクスプレッションの問題、DeviceIssue、FinderIssue
ロケーションが見つからなかった場合	エクスプレッションの問題、DeviceIssue、FinderIssue
除去されたロケーション	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
数秒が経過したとき	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

記録されないインスタント クリック

ワークフローへの記録をすることなく、要素に対してマウスのインスタント クリックを実行することができます。このアクションを実行するには、[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。

ウィンドウ

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。デバイスに接続したら、マウスとキーボードを使用してアプリケーションを実行するか、アプリケーションを開くためのパスを指定できます。

パス

- 開くアプリケーション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"`
- 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。

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。このステップは、ロボット ファイル システム上のファイルへの書き込みに利用できます。

注 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。

使用中の 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、ローカル デバイス上で [ファイル アクセス] を [ダイレクト アクセス] に設定してこのステップを使用することはできません。ただし、[ファイル アクセス] が [**RFS** 経由] に設定されているローカル デバイス上では使用が可能です。リモート デバイス上では、[ダイレクト アクセス] と [RFS 経由] の両方が使用できます。

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、 Desktop Automation ワークステーションの呼び出しステップの [必要なデバイス] プロパティで指定されます。
- [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なければなりません。
- [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、[ダイレクト 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、そのパスは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.txt` 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 タブ]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。

ターミナルの自動化

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。

ターミナル自動化のコマンド タイムアウトは、Design Studio 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 [**Desktop Automation**] タブ、または RoboServer 実行のための

[**RoboServer** 設定] ウィンドウの [セキュリティ] タブにある [**Desktop Automation**] セクションで設定します。『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Runtime」(ランタイム) >「Security」(セキュリティ) を参照してください。

注 ターミナル デバイスを自動化するときには、リモート コンピュータにDesktop Automation サービスをインストール、または Management Console でデバイス マッピングを作成する必要はありません。

マウス移動 ステップとクリック ステップを挿入することはできますが、ターミナルはマウス操作に対応していません。ターミナルはキーボードのキーを使用して操作する必要があります。

フォント

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に、複数のフォントをバンドルしています。フォントは以下のパスにあります。

```
C:\Program Files\Kofax RPA 11.0.0\nativelib\hub\windows-x32\XXX\fonts
```

ここでは、XXX は内部目的用の 3 桁の数字です。

バンドルされた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` ファイルの Truetype フォントにパスを追加することで修正ができます。例 :

```
C:\TerminalFonts\MyTerminalFont.ttf
```

TN6530 ターミナル

6530 ターミナルに接続するには、「ターミナル」ステップを挿入し、[ノンストップ **(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、必要なすべてのパラメータを指定します。

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。

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 ~ 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 も F11 ~ F16 にマッピングされます)

さらに以下の計算キーがサポートされます。

- 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。

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。

たとえば、`NetworkTrace=MyLogfile.log&LineBuffer=2048` のようになります。

`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/スクロールに使用できます。指定できる値は 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 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 `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>`

  SSH ユーザー。この設定は、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>`

  SSH パスワード。この設定は、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 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/スクロールに使用できます。指定できる値は 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るのに使用されます。`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。

`logfile` パラメータは、ファイルの作成時に以下の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 ($)。	$

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、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。

tn5250 ターミナル

5250 ターミナルに接続するには、「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。デフォルトと異なるポートを使用するには、[ホスト] フィールドで指定します (`TerminalHost:11625` など)。

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。

`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 タイプの値です。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)

Kofax RPA は 80x24 と 132x27 の両方のモードのターミナルをサポートしています。

5250 ターミナルは、現在の PC キーボードでは利用できない多数のキーを備えた特殊なキーボードを使用していました。このようなキーを入力するには、キー プレスで次の計算キーを使用できます。

- 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、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 には次のキーが追加されています。

- 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。このキーを使用すると、完全に入力されていないフィールドを終了することができますが、カーソルが次のフィールドにラップされている場合は、FieldExit が抑制されます。
- VK_DBCS_SPACE

  DBCS 空間を入力します。

5250 ターミナルに接続して情報を抽出する方法については、基本 ターミナル チュートリアルを参照してください。

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`

Kofax RPA は、tn5250 ターミナルの次の文字セットをサポートしています。

文字名	ホスト コードページ
US/Canada	cp037 (デフォルト)
Multinational	cp500
Thai	cp838
Japanese	cp930
Simplified Chinese	cp935
Korean	cp933

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。たとえば、iSeries ホストの DEVROBOT デバイスに接続するには:

```
env.DEVNAME=DEVROBOT
```

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、またはデバイスがオフあるいは使用中である場合は、接続が失敗します。

強化された **5250** インターフェース 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ー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。

注 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。

3270 ターミナル
Kofax RPA では、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。

3270 ターミナルは、現在の PC キーボードでは利用できない複数のキーを備えた特殊なキーボードを使用していました。このようなキーを入力するには、キー プレスで次の計算キーを使用できます。

- 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。

文字名	ホスト コードページ
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
italian	280
italian-euro	1144
japanese-kana	930
japanese-latin	939
norwegian	277
norwegian-euro	1142
russian	880
simplified-chinese	935

文字名	ホスト コードページ
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

SSH / Telnet の設定 (vt100 および ANSI)

SSH 接続を使用するには、「ターミナル」ステップでエミュレータに [接続 **(SSH)**] を選択します。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できますが、実際の PuTTY クライアントでセッションを定義をする必要はなく、[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できます。

ヘッドレス 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。このため、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、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のリストは、ICU (Unicode の国際コンポーネント) ライブラリのリストと同じです。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最も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。

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 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。

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	この設定は、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	この設定は、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。

SSH 認証およびセキュリティガイド

SSH を使用してアプリケーションを自動化するとき、ユーザーの認証方法には 4 つの方法を使用できます。

1. 基盤となる SSH クライアントでパスワードの入力を要求されます。「テキストを入力」ステップ、次に「キープレス」ステップ (リターン) を使用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 と RoboServer のファイル システムに置くことができます。「ターミナル」ステップの [オプション] フィールドに `PublicKeyFile=<path to key>` 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 秘密鍵を変換することができます。
3. 暗号化された秘密鍵は、Design Studio と RoboServer のファイル システムに置くことができます。[オプション] フィールドに `PublicKeyFile=<path to key>` を追加します。基盤となる

SSH クライアントで接続時にパスワードを要求されます。パスワードを入力するには、「テキストを入力」ステップ、次に「キープレス」ステップ (リターン) を使用します。

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、`SSH_AUTH_SOCK` と `SSH_AGENT_PID` 環境変数を RoboServer と Design Studio 環境にコピーします。以下のように、RoboServer.conf と DesignStudio.conf に行を追加することで実現できます。
     - `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。

上記の認証方法を、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. ロボットを読むことができる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ため、Management Console からロボットを取得するだけではログオンできない。
3. 攻撃者は、ロボットとプライベート キー ファイルの両方が必要になる。
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。

TLS/SSL を使用
Kofax RPA はターミナルでの TLS / SSL 通信をサポートしています。TLS/SSL を使用するには、「ターミナル」ステップで **stn3270** または **stn5250** ターミナルを使用し、ホスト名または IP アドレスと適切なポート番号を指定して、[セキュアな接続] を選択します。Kofax RPA は、サーバーによって提示された証明書を検証しないことに注意してください。

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、ログインし、コマンドを実行し、ターミナルから情報を抽出します。

1. 既存の Desktop Automation ロボットを開くか、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、「Desktop Automation ワークフローの呼び出し」ステップを使用して新しいロボットを追加します。このステップで開かれる Desktop Automation ロ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ロ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. 「Desktop Automation ワークフローの呼び出し」ステップを実行して、ツールバーの [**DA** ロボットにステップ イントゥー] ボタンをクリックします。
5. Desktop Automation 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。ターミナルの自動化の「サポートされている tn5250 ドライバ ターミナル」を参照してください。

注 「ターミナル」ステップを使用してターミナルに Desktop Automation ワークフローを再実行するには、Desktop Automation ロボットを終了し、Design Studio で「Desktop Automation ワークフローの呼び出し」ステップを使用して Web オートメーション ロボットを更新して、[**DA** ロボットにステップ イントゥー] をクリックします。ターミナルを閉じずにロ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、ロ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 "User ID" フィールドを右クリックし、**[**テキストを入力**]** > **[**変数から**]** > **[**ログイン**]** を選択し、ユーザー名を含む変数を選択します。[ステップ イントゥー] 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. Password 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、**[**スタンダード キー**]** > **[**タブ**]** を選択します。[ステップ イントゥー] 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 イントゥー] 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、**[**コンテキストを次へ抽出**]** > **[variableName]** を選択します。[ステップ イントゥー] 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、(バイナリ変数をリターン 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 ビューで画面エレメントを選択し、**[**画像を次へ抽出**]** > **[binaryVariableName]** をクリックします。後で、情報をバイナリ変数から、Web サイトのロボットで画像に変換することができます。[ステップ イントゥー] をクリックします。

ターミナルから情報を抽出すると、Web サイトのロボット エディター ウィンドウに戻って、抽出した情報を利用することができます。

Web サイトのアクセス

Desktop Automation ワークフローを作成する際に、組み込みブラウザで Web サイトを開き、アクションステップを使用して情報を抽出し、サイトに移動することができます。組み込みブラウザは、Chromium などの選択されたエンジンに基づいています。サイトに移動するには、Desktop Automation ステップ アクションを使用します。

注 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] ウィンドウの [セキュリティ] タブにある [**Desktop Automation**] セクションで設定します。『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Runtime」(ランタイム) >「Security」(セキュリティ) を参照してください。

Kofax RPA 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:`
- `ftp:`
- `file:`

Web サイトを開くには、参照 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、[**URL**] フィールドのアドレスなど、必要なパラメータをすべて指定します。ページの読み込みを含むステップの後、または Web ページ上で他の変更が発生した場合には、ツリー変更停止ガードを使用します。詳細については、ガード チョイス を参照してください。

ブラウザ インターフェース
Desktop Automation の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 つ前のページに戻ります。
	[進む]: 1 つ先のページに進みます。
	[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。
	[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	説明
	[**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。
	[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。このボタンはブラウザ ツール バーの右隅にあり、一部の画面では表示されない場合があります。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。

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。

重要 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限り、ロボットが Kofax RPA グローバル プロキシ設定を使用していることを確認してください。

Desktop Automation で組み込みブラウザのプロキシ設定を変更するには、Web 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、バイパス 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。

ページを **PDF** ドキュメントとして保存

Web ページを PDF 形式で保存するには、ブラウザ ツール バーの [**PDF** に印刷] をクリックし、開いているダイアログ ボックスにファイル名を含む完全なパスを入力します。次のように、パスに二重スラッシュを入力します。`C:\\AutomatedDevice\\PDFs\\MyPage.pdf`。[**OK**] をクリックしてファイルを保存します。

必要に応じて、[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 タブ]での指定に対応している必要があります。

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、[参照] 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。

Web ページの保存
ブラウザの [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。ページを保存するには:

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。ブラウザの左上隅にダイアログ ボックスが開きます。
2. ファイルを保存するためにパスを変更する場合は、パス テキスト 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からファイル名を含む新しいパスを挿入します。
3. [保存] ボタンをクリックしてページを保存します。

ファイルをダウンロードする
Web サイトからファイルをダウンロードし、[名前を付けて保存] ダイアログ ボックスでファイルを保存するパスを指定できます。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。

必要に応じて、[名前を付けて保存] 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 タブ]での指定に対応している必要があります。

Chrome Inspector を使用したデバッグ

Kofax RPA は、デバイス アクション ステップで Chrome Inspector を使用して、Chromium ブラウザ エンジンのデバッグをサポートします。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-x32\[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 (x86)\Kofax RPA 11.0.0.0 x32\nativelib\hub\windows-x32\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 [参照] 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。

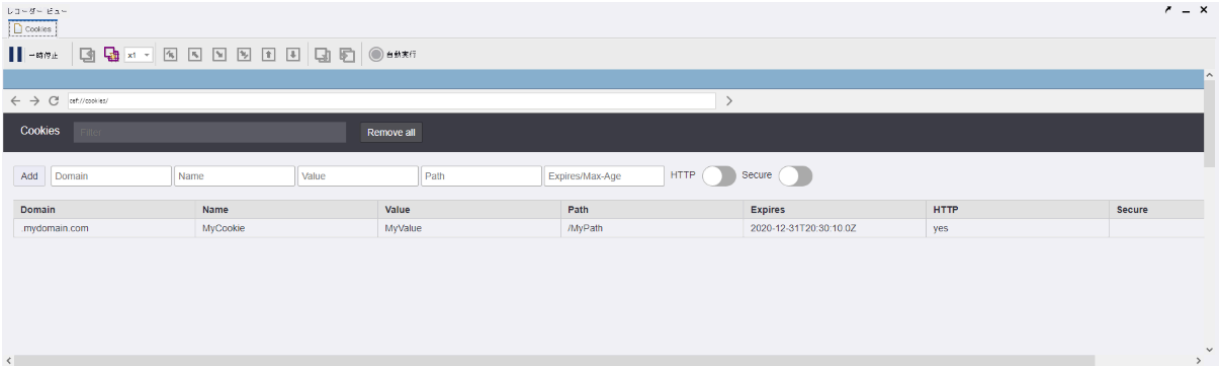
4. トレースを右クリックし、[コンテンツ付きの**HAR**としてすべて保存] を選択して、ネットワーク トレースをファイルに保存します。HTTP アーカイブ インスペクターでファイルを開くことができます。

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、Desktop Automation 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、「参照」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。

「参照」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、「参照」ステップを使用します。

- CookieStore に新しい Cookie を追加するには、「参照」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、「参照」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
  ```

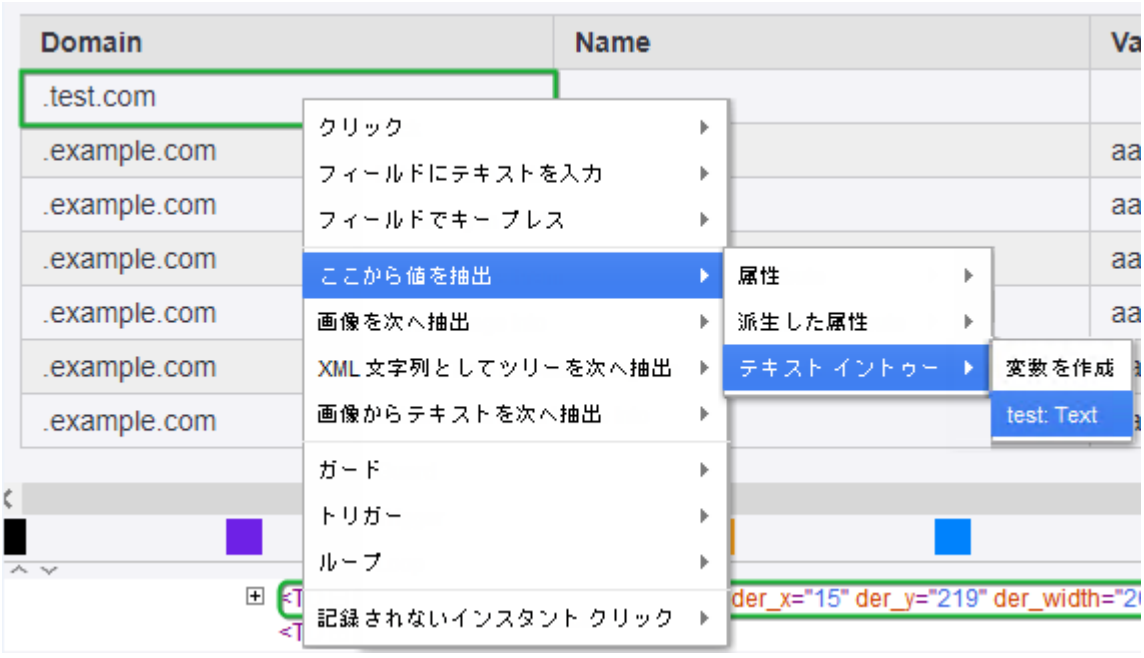
- すべての Cookie を CookieStore から削除するには、「参照」ステップの [URL] フィールドに次のテキストを入力します。

  ```
  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。

次のようなコンテキスト 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。



CookieStore から Cookie を抽出し、コンプレックス タイプの JSON 変数に格納することもできます

組み込み Excel ドライバー

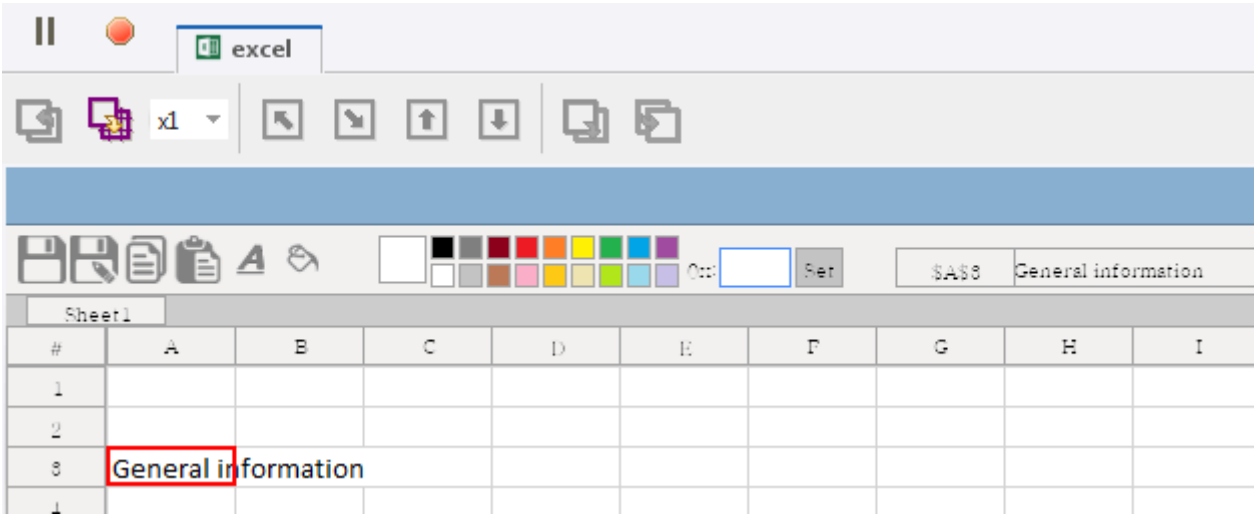
Desktop Automation ワークフローには、Excel スプレッドシートで一部の操作を実行できる組み込み Excel ドライバーが含まれています。

注 Desktop Automation で組み込み Excel ドライバーを使用するには、ロボットが実行されているコンピュータに Microsoft Excel をインストールする必要があります。

新しい Excel スプレッドシートを作成するには、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。

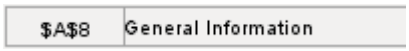
既存の Excel スプレッドシートを開くには、Excel ステップを実行し、[アクション] リストで [ファイルを開く] を選択して、スプレッドシートへのフル パスを入力します。たとえば、`excel://c:/documents/myspreadsheet.xlsx` のように入力します。

Kofax RPA では、1 つのワークシートのみのスプレッドシートを作成できます。複数のワークシートを含むスプレッドシートを開くときは、それぞれのワークシートのボタンをクリックしてワークシート間を移動できます。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表示されません。スプレッドシートを保存すると、グラフ シートが含まれます。



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	説明
	[保存]: スプレッドシートの変更を保存します。
	[名前を付けて保存]: [名前を付けて保存] ダイアログ ボックスを開き、現在のスプレッドシートを別の名前で保存します。

ボタン	説明
	[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。選択オプションについては、この表の下の「セルの選択」サブセクション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。詳細については、下記の「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`、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できます。

- 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。セルの選択範囲を上方向や、右から左に動か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、次のようにします。

  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、[スタンダード 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、[スタンダード 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。

色の変更
色の変更アクションを使用するには、最初に [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で使用されるよう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。Excel ドキュメント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] および [貼り付け] ボタンを使用してください。

重要 同じ Excel ドキュメント内のシート間でのみコピーと貼り付けができます。複数の Excel ドキュメントを開いている場合、ドキュメント間でコピーして貼り付けることはできません。

Excel ドキュメント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、「クリップボードを抽出」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。

ツリーの凍結操作
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スプレッドシートを操作できますこれにより、セル ループをより高速に実行し、より効率的な口ボットを作成できます。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、このトピックの説明に従って Excel ドライバーを呼び出す「ステップ画面を開く」を挿入します。

Windows 環境のヒント

このセクションでは、Windows サービス内でロボットを実行している場合のエラーを回避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。

「ステップ画面を開く」の実行中に次のエラーが発生することがあります。

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

このエラーは、Excel のヘッドレス モード動作の機能によって発生します。

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ます。これらのフォルダによって、Excel をサービスから開始することができます。

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、コマンド実行中にエラーが発生しました: Excel インスタンスの作成エラー: 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。これら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。

   注 [**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。

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新しい方法です。次のステップは、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するのに役立ちます。

重要 Kofax RPA バージョン 10.7 以降、トリガーを持つロボットを使用するには追加の設定が必要です。詳細については、 Desktop Automation サービスの設定の「Windowsタブ」セクションを参照してください。

1. トリガー ロボット用にシンプルなデバイス マッピングを作成します。
2. 作成したマッピングを「トリガーで参照」として、「Desktop Automation ワークフローの呼び出し」ステップの [デバイスの追加] ダイアログ ボックスで [必要なデバイス] に追加します。
3. トリガー チョイスを Desktop Automation ロボットに追加します。
4. ロボット開発中、リモート デバイスのDesktop automation service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。
5. 開発の完了後、リモート デバイスでDesktop automation service設定を開き、[シングル ユーザー] オプションをクリアして、ロボットが展開される Management Console を指定します。

トリガーを持つロボットを Management Console にアップロードしたら、ロボットを [トリガー マッピング] タブ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。

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。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] タブ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。

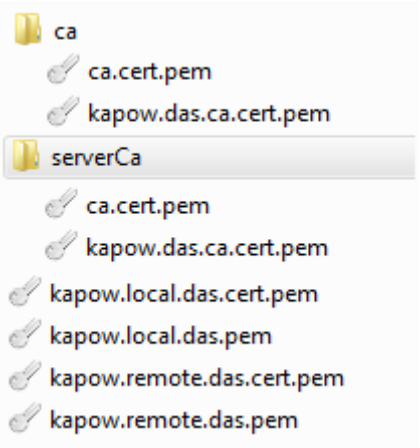
TLS コミュニケーションを使用

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。

注 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、6 つのファイルと 2 つの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。`kapow.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 `ca` と `serverCa` フォルダにあります。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。

`kapow.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。`kapow.local.das.cert.pem` は、`kapow.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。

kapow.remote.das.pem は、Desktop Automation サービスが使用する秘密鍵ファイルです。kapow.remote.das.cert.pem は、kapow.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、kapow.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、kapow.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れば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あれば十分です。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。Desktop Automation サービスの設定ウィンドウの [証明書] タブを使用して、証明書のパスを新しいものに変更します。
  2. 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。
  3. 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、kapow.local.das.pem、kapow.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。

データの変換

Desktop Automation の変換機能はデータ変換と口ボット開発を促進します。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

- クリップボードを抽出
- ツリーを XML として抽出

これらのステップを使用して、データを抽出および/または変換できます。これらのステップに、実際のデータ変換を実行するエクスプレッションの評価ステップを追加できます。

Desktop Automation 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。

- **$current** 変数には初期値が含まれています。この変数の値は実行中に変更されます。値のタイプも変更できます (数値をテキスト値に変換できるなど)。
- **$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、いつでも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、Desktop Automation 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。

Desktop Automation ステップのプロパティの多くは、プレーン値 (たとえば、数値) またはエクスプレッションとして指定することもできます。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` がステップの範囲で定義された変数となる、Desktop Automation ワークフローの他の場所に値が与えられる `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。

Desktop Automation のエクスプレッションは、Java、C＃、JavaScript などの一般的なプログラミング言語のエクスプレッションと非常によく似ています。これらのエクスプレッションは、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とものとなります。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て Desktop Automation ワークフローでのエラーとしてそのタイプ エラーが報告される前に、エクスプレッションのタイプの静的な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について説明します。

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 注 Desktop Automation の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、加算、減算、乗算、および除算演算で発生することがあります。結果が数値の限界を超えている場合は、数値が大きすぎてオーバーフローし、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なければなりません。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型は同じ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

演算子	説明
>=	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 タイプでなければならず、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、同じでなければなりません。

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

演算子	説明	例
==,!=	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`
!	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。

数値関数

関数	結果タイプ
abs(Integer)	整数
abs(Number)	数値
ceil(Number)	整数
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。
floor(Number)	整数
round(Number)	整数
trunc(Number)	整数

関数	結果タイプ
max(Integer, Integer)	整数
max(Number, Number)	数値
min(Integer, Integer)	整数
min(Number, Number)	数値
random()	数値 0.0 以上 1.0 未満の乱数を返します。
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。

例	以下の結果に評価
abs(-2)	2
1.5.round()	2
random(1,6)	1 と 6 の間の整数値

テキスト関数

関数	結果タイプ
length(Text)	整数
substring(Text, Integer)	テキスト
substring(Text, Integer, Integer)	テキスト
indexOf(Text, Text)	整数
contains(Text, Text)	ブール値
trim(Text)	テキスト
capitalize(Text)	テキスト
startsWith(Text, Text)	ブール値
endsWith(Text, Text)	ブール値
toLowerCase(Text)	テキスト
toUpperCase(Text)	テキスト
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。
removeNonPrintable(text: Text)	テキスト 印刷不可文字を除去します。
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。

関数	結果タイプ
unquote(text: Text)	テキスト テキストから引用符を除去します。たとえば、`"hello"` と `'hello'` は、引用符なしの `hello` となります。

注 * 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。

例	以下の結果に評価
"workflow".substring(5)	低

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。
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。
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。
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。
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。
integer(Number)	整数 数値は整数値である必要があります (例：1.0)。
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。
number(Integer)	数値
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

関数	結果タイプ
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。
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 注 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。Desktop Automation 口ボットで 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。これにより、Desktop Automation 口ボットから Web オートメーション 口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/>

例	
関数	以下の結果に評価
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=
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

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。Desktop Automation ワークフロー状態ビューまたは式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。

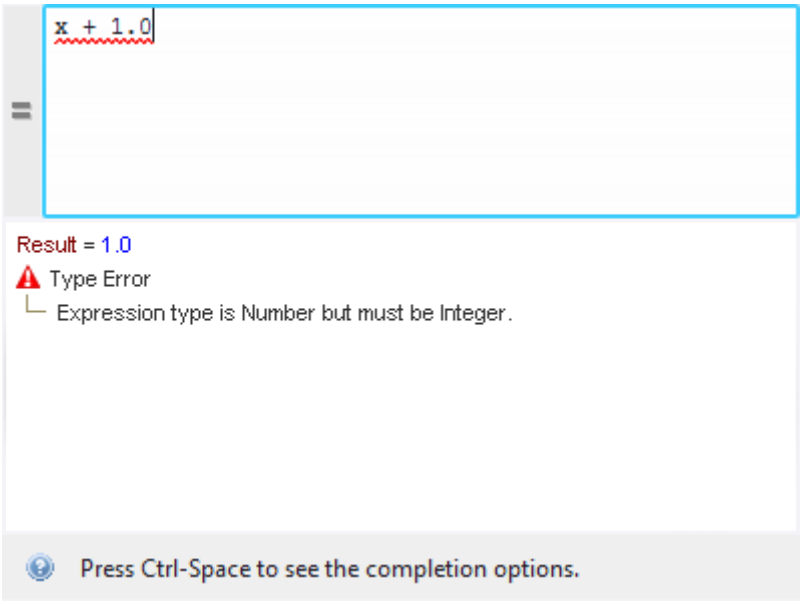
また、変換関数 `text(Number)` も上記のルールに従って記述された数値を返します。

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 Desktop Automation ロボットで入力フィールドをクリックして開く、指定され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の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。

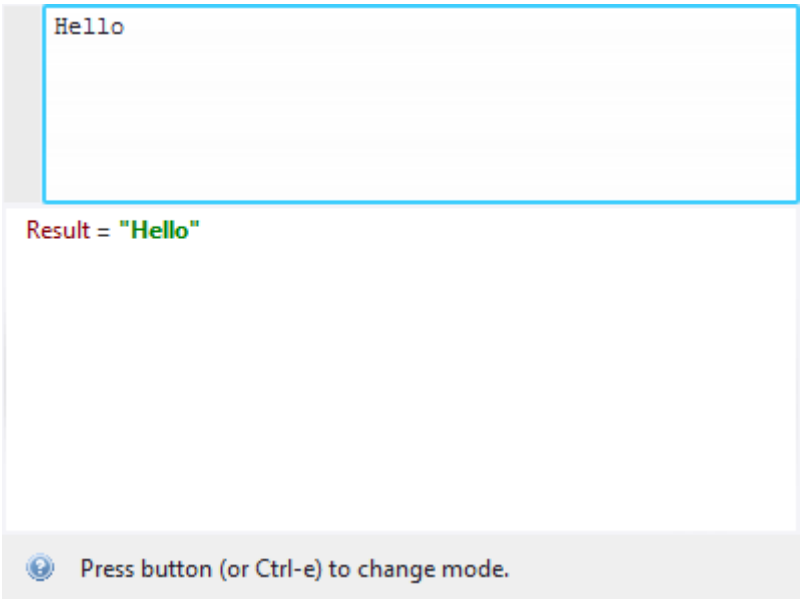


右クリックして [値のコピー] を選択すると、エディターの下部ペインからエクスプレッションの結果とエラー メッセージをコピーすることができます。値がレコード タイプの場合、結果はツリーとして表示され、各属性値は個別にコピーできます。パスワードとバイナリ値はコピーできません。

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。

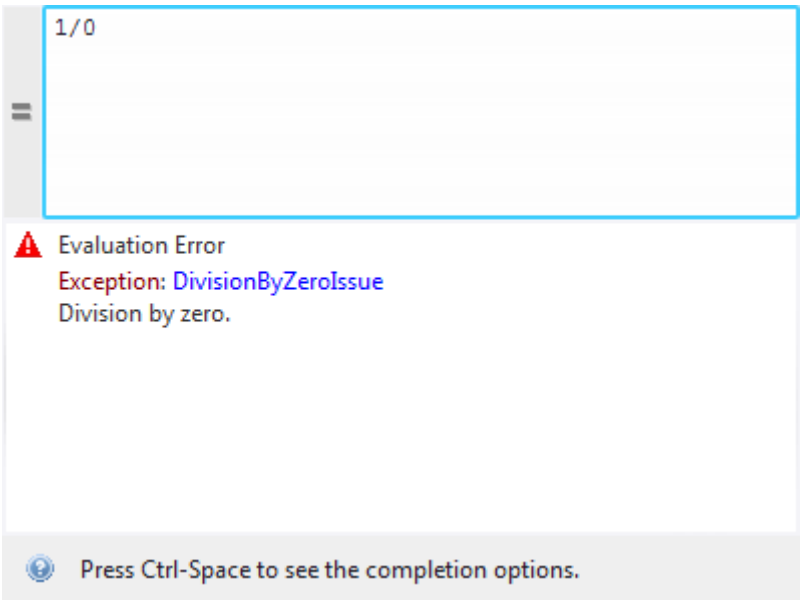
エディター モード
エディターには、上部ペインの左側にエクスプレッション モードと値モードを切り替えるモード ボタンがあります。上図のエディターは、エクスプレッション モードです。エディターが値モードになっているとき、モード ボタンは空白となります。エディターがエクスプレッション モードのとき、ボタンには等号 (=) が表示されます。キーボード ショートカット [Ctrl-E] を使って、2 つのモードを切り替えることができます。

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。



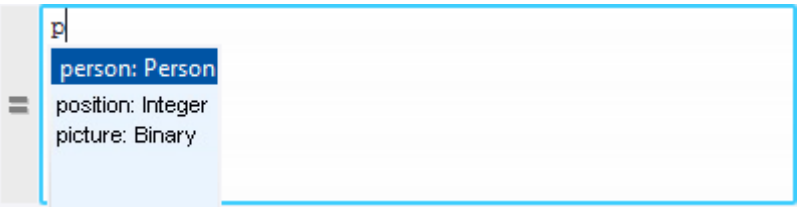
エクスプレッション モード
エクスプレッション モードでは、入力ペインに入力したすべての内容がエクスプレッションとして解釈され、構文とタイプのエラーについてチェックが行えます。現在のフロー ポイントでエクスプレッションを編集していてエラーがない場合は、評価され、その結果が表示されます。エクスプレッションの状態が常にわかるように、入力の途中で評価が行われます。現在のフロー ポイントにないエクスプ

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。

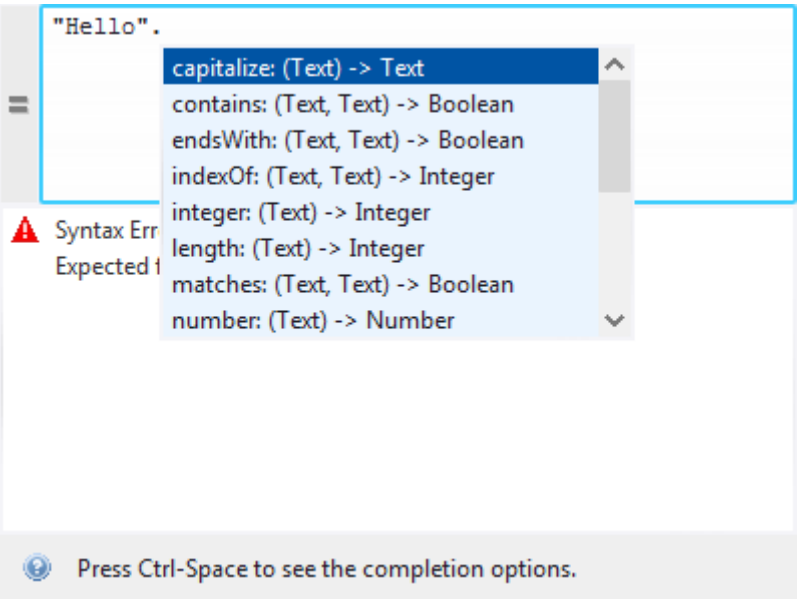


エクスプレッション モードでのテキスト補完は、変数名、フィールド名、関数名の入力に役立ちます。補完ヘルプがあるものを入力すると、テキスト補完ウィンドウが自動的に表示されます。たとえば、変数名の入力を開始し、すでに入力したものから始まる変数がある場合、補完ウィンドウが表示されます。レコード タイプの変数の後でドット (.) を押すと、補完ウィンドウには、レコード タイプのフィー

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。



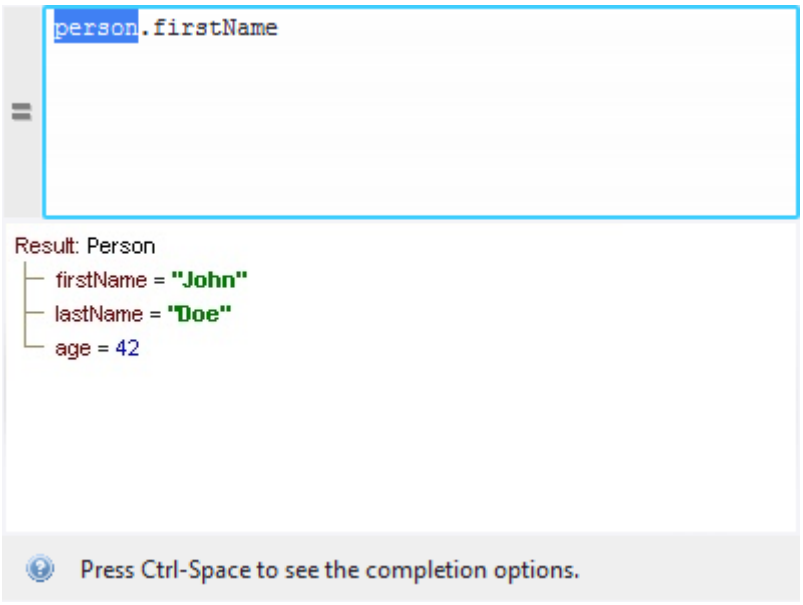
シンプル タイプのサブエクスプレッションの後にドット (.) を押すと、最初の引数が同じシンプル タイプを持つ関数に対応する補完オプションのリストが、補完ウィンドウに表示されます。



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、次のように、この機能の説明が結果ペインに表示されます。

変数

Desktop Automation ワークフローでは、Web サイト ロボットから入力を取得して、何らかの付加を伴う同一の変数タイプを使用できます。

- 複雑な変数とその属性を Desktop Automation ワークフローへの入力として使用できます。
- 日付、通貨、国、言語フィールドを含む変数を除く、プロジェクトのすべてのコンプレックス タイプを使用して、Desktop Automation ワークフローでレコード変数を作成できます。

  レコード変数は、ロボットが使用するレコード タイプ (複雑な変数) に基づいています。レコード タイプとは、使用可能な任意の値およびそのタイプの変数で表される、一連のフィールドとして定義可能なデータ タイプのことです。

  たとえば、クレデンシャル レコード タイプの値には «Credentials» という名前が付けられ、ユーザー名は「Joe」、パスワードは «Password» になります。この情報は状態ビューに表示されます。クレデンシャル タイプの変数 (userCred など) がある場合に、このタイプの値を変更するには、「割り当て」ステップで userCred.username に別の値 (「Tom」など) を割り当てます。こうすると、変数の値が «Credentials» に変更され、ユーザー名は「Tom」、パスワードは «Password» になります。
- 同じレコード タイプのレコード変数を互いに割り当てることができます。

- レコード変数の属性を同じタイプの値や同じタイプの別の属性または簡単な変数に割り当てることができます。

  注 レコーダー ビューの変数を使用するすべてのショートカット メニューからレコード タイプ変数のフィールドを選択できます。
  たとえば、テキストを入力はその値を複雑な変数のフィールドから取得でき、選択した変数のタイプがテキストではない場合、値をテキストに変換するために、変換関数が挿入されます。テキストに変換できないタイプの変数とフィールドは、リストに表示されません。
  抽出ステップの場合、複雑な変数のフィールドに抽出することもできますが、フィールドのタイプは、テキストや (画像の) バイナリなどの抽出されたデータのタイプに一致する必要があります。正しいタイプの変数のみがメニューに表示されます。

- ワークフローの実行状態をリセットせずに既存の変数を編集できます。
- 作業を中断して変数セクションにスクロールせずに、コンテキスト メニューから新しい変数を作成できます。
- Web サイト ロボットの日付タイプ変数は、Desktop Automation ロボットの入力として使用できません。
- ローカル変数を作成できますが、グループ ステップのみで使用できます。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。
- Desktop Automation のパスワード タイプ変数は、その値を Web サイト 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。Desktop Automation 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。

デフォルトの初期値で利用可能な変数タイプのリストは次のとおりです。

- 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："""

数値の制限

Desktop Automation ワークフローとウェブサイト ロボットの数値形式は異なります。ウェブサイト ロボットでは、変数に格納される値は倍精度です (IEEE754 規格で規定された binary64)。Desktop Automationでは、変数は IEEE 754R decimal128 形式で数値を格納しますが、34 桁の 10 進数と −2147483647 ~ +2147483648 (= 2^31) の指数範囲を使います。Desktop Automation ワークフローに値を格納するときは、四捨五入が発生し、精度の低下が予想できます。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、1234567890123456789012345678901234567890.0 は、123456789012345678901234567890123500000 0 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。

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ます。サービスがシングル ユーザー モードで動作するように構成し、トークンを指定します。Desktop Automation エージェントの設定を参照してください。
2. Desktop Automation ワークフローで、「セッションの開始」ステップを挿入します。
3. RDP 接続に必要なすべてのオプションを指定します。
   セッションの初期化ステップは、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。

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、「セッションの開始」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、長時間アイドル状態にある RDP セッションを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、「セッションの開始」ステップの [**Keep-awake** のキー プレス間隔] オプションで間隔を秒数で指定します。このオプションをオフにするには、ゼロ (0) を指定します。

注 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 接続はすべて閉じられます。

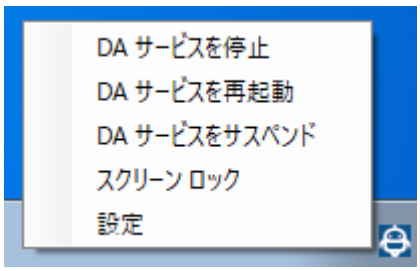
デバイスがスタティック リファレンスの場合は、RDP 接続が確立された後に、レコーダー ビューに使用可能なアプリケーションが表示されます。

デバイスがダイナミック 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。タイムアウト設定と再試行を複数回実行し、RDP 接続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。

注 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。

リモート デスクトップの管理

Desktop Automation サービスのショートカット メニューを使用して以下のアクションを実行できます。



Desktop Automation サービスの管理

以下のコマンドは、リモートのコンピュータ上で実行される Desktop Automation サービスを管理するのに役立ちます。

- [**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。
- [**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。
- [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: Desktop Automation サービスの設定ダイアログ ボックスを開きます。

第 5 章

Management Console

Management Console は、Kofax RPA プラットフォーム サーバーのポイント アンド クリック監視および管理に利用できる Web ベースのインターフェイスです。

Management Console では、以下を行うことができます。

- 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 ロールおよび権限の管理、およびソリューションの集中管理。
- リポジトリからのロボットのスケジュール設定。
- 抽出データのブラウジング、および MS Excel 形式へのエクスポート。
- プロダクション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。
- Design Studio からリポジトリへのロボットの展開。

続行する前に、Management Console ビギナーズ チュートリアルを参照してください。

Management Console の構造の概要

Management Console は 5 つの機能領域に分割されます。

スケジュール

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。スケジュールはロボットを実行するものではありません。単純にロボットを実行するタイミングについての計画を提供するものです。これは設定したサーバーにロボットを渡すことによって実行されます。

リポジトリ

ロボット、タイプの定義、リソースを Design Studio から Management Console リポジトリにアップロードしたり、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したりできます。アップロードしたロボットは、スケジュール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

データ ビューでは、ロボットがデータベース データ登録したデータを表示したり、このデータを Excel や XML にエクスポートしたりできます。

ログ

ログを使用して、スケジュールの実行履歴を確認できます。データベース ログが有効化されている場合、すべてのロボット実行の詳細が含まれている RoboServer ログを表示することもできます。

管理

管理セクションを使用して、Management Console の設定を構成します。また、このセクションでは、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。このセクションで、ライセンスを設定し、バックアップを作成/復元することもできます。

注 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。

命名規則

ファイル名に "." を使用するときの制限に注意してください。

- "." で始まるファイルをアップロードしたり、そのようなフォルダを作成したりすることはできません。
- "." で始まる名前を使って、(クラスタの下に) データベースを、および (プロジェクトの下に) データベース マッピングをそれぞれ作成することはできません。

Management Console の起動

Management Console のコンポーネントは、RoboServer のオプション部分です。RoboServer を起動し、Management Console として機能するように命令することで Management Console を起動できます。また、RoboServer の機能を実行することも可能です。RoboServer の起動と使用についての詳細は、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) を参照してください。

Management Console の機能を持つ RoboServer は、スタート メニュー (Windows) の Management Console の項目から起動できます。Linux および Unix 系 OS では、以下のコマンド ラインを使用します。

```
RoboServer -MC
```

この操作では、Management Console としてのみ RoboServer が起動するため、ロボットを実行することはできません。Management Console の Web インターフェイスは、[設定] で設定されているポートに接続することでアクセスできます。組み込み型 Management Console の設定についての詳細は、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「RoboServer Configuration」(RoboServer 設定) を参照してください。この方法で Management Console を起動した場合、Control Center または ShutDownRoboServer プログラムから再起動またはシャット ダウンすることはできません。

以下のコマンドを使用することでも、Management Console を起動することができます。

```
RoboServer -p 50000 -MC
```

これにより、ポート 50000 のソケットでリスニングしている RoboServer が起動し、設定したポート (ポートは [設定] で設定) の Web インターフェイスから Management Console 機能を利用できます。

Management Console の起動方法に関係なく、RoboServer および Design Studio インスタンスのライセンス サーバーとして機能させるには、Web インターフェイスにライセンス キーを入力する必要があります。

プロダクション シナリオについては、Management Console でひとつの RoboServer を起動し、クラスタに RoboServer を追加しないことをお勧めします。こうすることで、RoboServer によって使用されているリソースへのアクセスが拒否された場合に起こり得る Management Console スタベーションを防ぐことができます。Management Console を実行して RoboServer をセットアップしたら、パラメータ "-p 50000" を使用して、希望の数の RoboServer を起動することができます。これらは、クラスタ タブの RoboServer クラスタに追加される必要のある RoboServer のみです。

Management Console の設定およびユーザー インターフェイス

Management Console には、同じネットワーク上のマシンからのアクセスを容易にする Web ベースのインターフェイスがあります。Hostname という名前のマシンにインストールされている場合は、ブラウザを以下に指定する必要があります。

```
http://hostname:50080
```

ポート番号 50080 は、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Embedded Management Console Configuration」(埋め込み Management Console の設定) の説明に従って設定できます。

Management Console 内の移動は、ウィンドウ上部のタブを使用して行います。

Management Console のサブセクションの多くは、特定のタイプの項目を表に表示します。表示する項目が多すぎる場合は、複数のページに分割して表示されます。[ページごとのスケジュール] 設定は、同時に表示する項目数 (ここではスケジュール) を選択するのに使用できます。注意：この数値は、ブラウザ ウィンドウの高さに自動的に調整されません。そのため、表の下の空白は、必ずしも最後の項目が表示されていることを意味しているわけではありません。ウィンドウの高さに比べて、「ページごと」の設定が低すぎることを意味している場合もあります。

表の下部では、表示の個別のページ間を移動することができます。つまり、同じ表内での上下への移動が可能です。 ボタンは、表示を更新します。

表は、並び替え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。並び替えは、ページごとに個別ではなく、表全体で行われます。そのため、2「ページ」以上ある場合、並び替え順を変更すると、完全に異なる一連の行が表示されます。

Kofax RPA バージョン 10 以降では、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、クラスタ名とともに Management Console の URL および資格情報は、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)。

スケジュール

スケジュールのサブセクションでは、Management Console のスケジュールを管理することができます。スケジュールは、一連の口ボットを表示し、これらの実行を計画します。スケジュールを実行する

ということは、選択したロボットを実行し (同時または連続)、オプションで事前実行スクリプトおよび事後実行のスクリプトまたはロボットを実行することを意味します。

重要 スケジュールの実行間隔を選択する場合、スケジュールが実行されている際に実行時間となっても、このスケジュールは開始されません。

以下の情報は、スケジュールのリストにおける各スケジュールに表示されます。この情報は、列に示されます。一部の列はデフォルトで隠されています。また、列のヘッダーの下向きの矢印をクリックして、列を選択することで表示できます。

列	説明
アクティブ	アクティブの際には、スケジュールは計画通りに実行されます。以下のような場合には、いくつかの理由で、スケジュールを非アクティブの状態にする必要があります。 • スケジュールによって実行されている機能が現在不要な場合。 • ロボットにエラーが見つかり、これらのエラーを修正する前にスケジュールを実行したくない場合。 • 実行する度に手動でスケジュールをトリガしたい場合。これは、タスクの準備またはクリーンアップなど、一部のロボットおよびスケジュールに適切です。
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」のプロジェクトが選択されている際に便利)。
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、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には、2 (3) が一覧表示されます。
合計ジョブ数 (デフォルトでは非表示)	スケジュールにおけるジョブの合計数。(ジョブを表示するには、以下で説明されているようにスケジュールを編集します。)
アクティブなジョブ (デフォルトでは非表示)	スケジュールにおけるアクティブなジョブの合計数。(ジョブを表示するには、以下で説明されているようにスケジュールを編集します。)
次の実行	スケジュールの次の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	スケジュールの 2 回にわたる連続した実行における計画間隔。
合計実行数	スケジュールが実行された回数。
作成者 (デフォルトでは非表示)	このスケジュールを作成したユーザーの名前。
変更者 (デフォルトでは非表示)	この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 (デフォルトでは非表示)	スケジュールが実行されるクラスタ。
削除	クリックしてスケジュールを削除します。
編集	クリックして、スケジュールの編集、または特定のスケジュールの詳細の表示を行います。スケジュールの編集は、行をダブルクリックすることでも行えます。

列	説明
実行/停止	クリックして、スケジュールを手動で実行します。これは、非アクティブなスケジュールに特に役立ちます。 スケジュールが既に実行されている場合は、停止します。これは、すべての実行しているロボットが可能な限り早く停止することを意味します。スケジュールは、すべてのロボットが実行を停止するまで「実行中」として表示されます。
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。事前処理または事後処理で削除されたクラスタまたはエラーなど、スケジュール エラーが発生すると、スケジュールの実行が妨げられます。 注 エラー (およびロボットのエラー) を表示するには、ログ データベースが Management Console 設定 ([設定] -> [ログ データベース]) で設定され、データベースのロギングが ロギング クラスタ設定から RoboServer で有効化されていることを確認してください。詳細については、ログ ビューを参照してください。
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。

新規スケジュールを作成するには、左上隅の [追加] をクリックします。現在のプロジェクト選択が「すべて」である場合、[追加] ボタンをクリックする際、新規スケジュールを追加する前に実際のプロジェクトを選択する必要があります。

既存のスケジュールをコピーするには、スケジュールを右クリックし、[コピーの作成] を選択します。

[編集] の列で をクリックするか、既存のスケジュールの列の任意の場所をダブルクリックすると、[スケジュールの編集] ダイアログ ボックスが開きます。これは、スケジュールの変更や、すべての詳細の表示に便利です。

このダイアログ ボックスには、以下の 2 つのタブが含まれます。[基本] および [詳細]。[基本] タブには、通常のスケジュールの設定に必要なすべてが含まれます。[詳細] タブには、ランタイム制約を設定できます。

スケジュールに設定できるのは、以下の情報です。

フィールド	説明
名前	スケジュールの名前。
アクティブ	アクティブなスケジュールには、チェック マークが付きます。
シンプル/Cron	これは、スケジュールの時間計画を定義する 2 つの方法から選択できます。
毎回	(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。これは、単位を付けた正数として入力できます (例 : 1 分または 3 時間)
パターン	(Cron スケジュールにのみ利用できます。) スケジュールが実行されるべき時を定義するパターン。フォーマットの詳細については、Cron スケジュールを参照してください。

フィールド	説明
前処理	その他の口ボットが実行される前に、スケジュールの一部として実行されるスクリプトまたは口ボットの名前。RoboServer コンピュータでアプリケーションを実行する場合、または Management Console コンピュータで事前処理スクリプトを実行する場合は、次の手順を実行する必要があります。 • RoboServer コンピュータでアプリケーションを実行するには: 1. RoboServer の設定で [ファイル システムとコマンド ラインのアクセスを許可] オプションを有効にします。これらの設定は、インストール フォルダまたは Windows のスタート メニューからアクセスできます。 2. 実行するアプリケーションを開く「開く」ステップを口ボットに挿入します。 • スケジュールを開始する前に Management Console コンピュータで事前処理スクリプトを実行するには: 1. Management Console を閉じます。 2. `/WEB-INF/Configuration.xml` ファイルを開きます。 3. 行 `<property name="allowScriptExecution" value="false">` を検索して、value を `true` に変更します。 4. Web サーバーを再起動して、Management Console を起動します。 スクリプト ファイルは、Windows では .cmd ファイル、Linux では .sh ファイルです。フィールドには、ファイルの絶対パスが含まれる必要があります (例 : `c:\scripts\truncatedb.cmd`)。プロジェクトのインポートまたはバックアップの復元を行う際には、スクリプトのコピーを忘れずに行ってください。 口ボットの場合は、runrobot を使用できます: TutorialCase1 を使用して、リポジトリから TutorialCase1 口ボットを実行します。
後処理	その他すべての口ボットが実行された後、スケジュールの一部として実行するスクリプトまたは口ボットの名前。詳細については、上記の事後処理を参照してください。
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。
口ボット (右側)	ジョブのリストを含む以下の表を参照してください。
実行時間制限 ([詳細] タブ)	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り実行されると、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 ([詳細] タブ)	各口ボットが出力できる値の最大数を選択します。口ボットの出力値がこの値を超えると、サーバーが停止し、エラーがログに記録されます。デフォルトの値は `-1` で、これは数に制限がないことを示します。
ジョブを順次実行 ([詳細] タブ)	これをチェックすると、口ボットは [基本] タブに一覧されている順で実行されます。

フィールド	説明
電子メール通知を使用 ([詳細] タブ)	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 [オプション] タブで設定し、必要な電子メール アドレスを以下のフィールドに入力している場合にのみ動作します。
電子メール ([詳細] タブ)	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。

右側には、スケジュールがトリガされたときに実行するジョブのリストが表示されます。

列	説明
ジョブ名	ジョブの表示名。これは、ジョブが作成されたときに選択されます。
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ている機能が現在不要な場合。 • ロボットにエラーが見つかり、これらのエラーを修正する前にスケジュールを実行したくない場合。 • 実行する度に毎回手動でジョブをトリガしたい場合。
除去	クリックすると、このスケジュールからジョブが除去されます。この操作を行っても、ジョブによって参照されるロボットが削除されることはありません。
編集	クリックしてジョブを編集します。

スケジュールにジョブを追加するには、右上隅の [ジョブの追加] をクリックします。これにより、ジョブの作成手順をガイドするウィザードが開きます。

[スケジュール] タブの右上隅の [ヘルプ] ボタンを使用すると、ブラウザで [Management Console] ヘルプが開きます。

別のスケジュール作成方法

ロボット タブからスケジュールを作成することもできます。これは、ロボットの数を選択して、右クリックし、コンテキスト メニューから [スケジュールの作成] を選択することで行います。これにより、すでに追加されたロボットを含む [新しいスケジュール] が開きます。

ジョブの追加

1. [新規スケジュール] ウィンドウの [基本] タブで、[ジョブの追加] をクリックします。
   ジョブの作成をガイドするウィザードが表示されます。

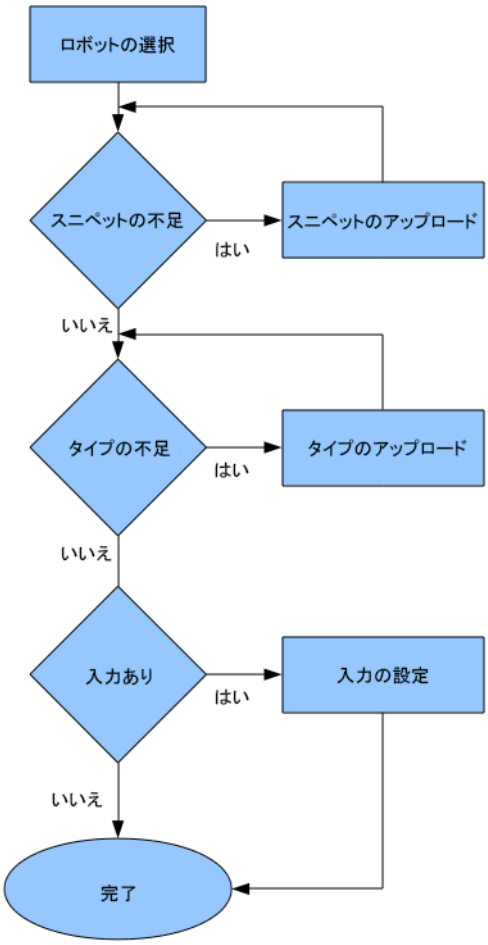
2. [ジョブ タイプの選択] ウィンドウで、オプションを選択します。利用可能なオプションには、[シングル ロボット] と [名前を基準としたロボットのグループ] が含まれます。

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。入力をロボットに渡す必要がある場合、必要なパラメータを指定します。
複数のロボット	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。この基準は、「次でロボット パスを開始」、「ロボット パスは次を含む」、「ロボット パスはパターンに一致」です。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規作成ロボットが自動的に実行されます。

3. [次へ] をクリックします。
選択したジョブ タイプに基づくシングル ロボットの追加またはロボットのグループの追加のトピックを参照してください。

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。このウィザードは、このトピックで概説されているフローに従います。4 つの長方形は、4 つの考えられるステップに対応しています。



1. [ロボットの選択] ステップのロボット リストにおいてロボットを選択します。
2. ロボットによって使用されるスニペットおよびタイプすべてが既にアップロードされており、ロボットにいかなる入力変数もない場合は、[終了] をクリックしてください。ロボットに入力変数がある場合は、[次へ] をクリックします。

   選択したロボットが Management Console リポジトリにアップロードされていないスニペットを使用する場合、これらを今すぐアップロードする必要があります。
3. [不足したスニペットのアップロード] ウィンドウにおいて、不足しているスニペットを参照して、[アップロード] をクリックします。フォルダは、[フォルダ選択] リストで指定することもできます。

   選択したロボットが、Management Console リポジトリにアップロードされていないタイプを使用する場合、これらを今すぐアップロードする必要があります。
4. [不足したタイプのアップロード] ウィンドウにおいて、不足しているタイプを参照して、[アップロード] をクリックします。フォルダは、[フォルダ選択] リストで指定することもできます。

   [ロボット] ウィンドウの設定入力が表示されます。
5. 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。

6. 属性がバイナリ 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既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。

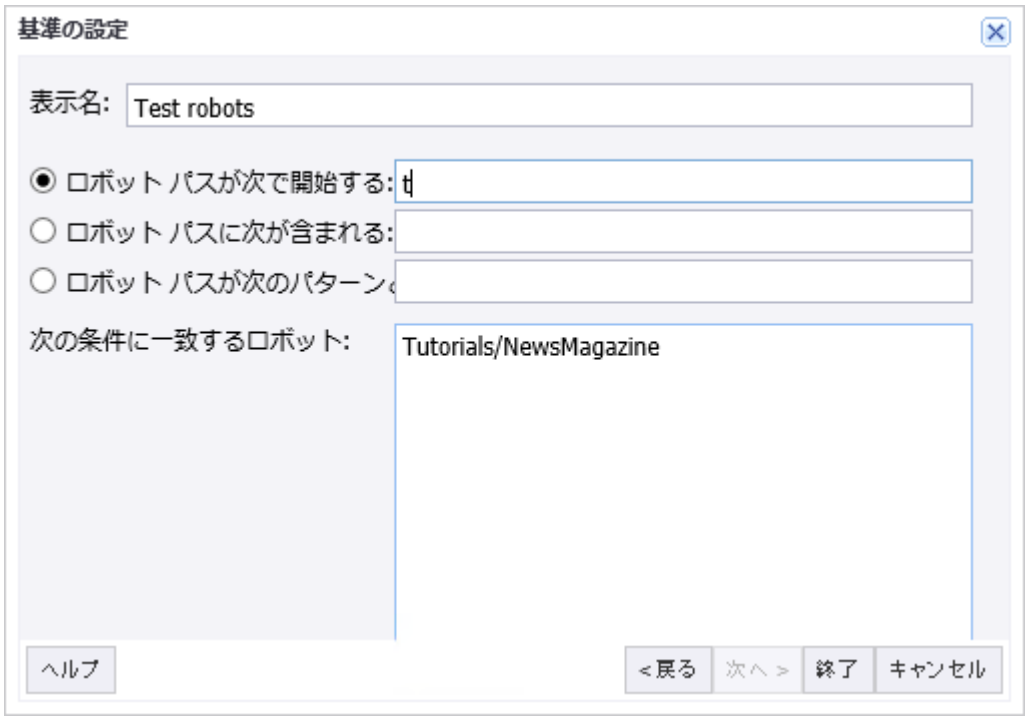
   属性が必要なときは、赤で下線が引かれます。
7. すべての必須のフィールドを記入して、[終了] をクリックします。

ロボットのグループの追加

これを選択すると、指定の条件に一致するパス名とともに、すべてのロボットを実行する単一のジョブを追加できます。この条件は、スケジュールが実行されるときに評価されます。したがって、ジョブの作成後にアップロードされたロボットは、設定された条件に一致する場合に含まれることになります。

この条件はランタイムで評価されるため、スニペット/タイプの不足などのエラーが、スケジュール実行ログにエラーとしてログが記録されます。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。

パス名の条件に基づいてグループを作成するためのウィザードには、単一のステップが含まれます。



1. [基準の設定] ウィンドウで、条件のタイプを選択します。利用できるオプションは次の通りです。「次でロボット パスを開始」、「ロボット パスは次を含む」、「ロボット パスはパターンに一致」です。

   下部のリストでは、選択した条件に一致する、1 つ以上のロボットが表示されます。[表示名] フィールドを編集しない限り、条件の選択によって変更されます。ただし、編集を開始すると、自動命名が無効化されます。

2. [終了] をクリックします。
   設定した条件に一致するロボットがない場合でも、[終了] をクリックできます。これは、一致するロボットを後でアップロードできるためです。
   複数のロボットがある場合は、選択した条件に一致するロボットのリストの更新にしばらく時間がかかることがあります。

   注 このジョブ タイプが、ジョブを連続的に実行するように設定されたスケジュールに追加されると、条件に一致するロボットのグループ内では、順番を制御できません (ただし、連続的に実行されます)。

リポジトリ

Management Console では、ロボット、タイプ、スニペット、リソース、および OAuth 資格情報のリポジトリを維持します。「リポジトリ」の章のトピックは、アセットの管理に役立ちます。

注 [フィルタ] テキスト ボックスのフィルタを適用して、テーブルのアイテム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。

ロボット

このページは、各プロジェクトに基づいて、リポジトリにおいてロボットを管理するのに役立ちます。ロボットがスケジュールで実行できるようにするためには、リポジトリ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ると、リポジトリにコピーされます。よって、Design Studio においてロボットに対して後で変更が行われた場合は、ロボットを再びアップロードする必要があります (これは Design Studio から簡単に行うことができます)。この操作を行っても、すでに関連付けられているスケジュールからロボットが除去されることはありません。その代わりに、これらのスケジュールは、次回実行する際に新しいバージョンのロボットを使用します。

各ロボットは、プロジェクトに属します。[ロボット] タブの上部では、ロボットを表示するプロジェクトを選択できます。

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。これらは同じロボットとして考慮され、最後にアップロードしたもので以前のものが上書きされます。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。同類のロボットにタグを割り当てると、リストから検索およびフィルタする際に使用できるようになります。

ロボットは、デフォルトでは各ページにつき 40 ロボットずつテーブルに表示されます。この情報は、以下のように列で構成されます。

注 ロボット名、タグ名を入力するか、プロジェクトを選択することで、テーブル内のアイテムのリストをフィルタできます。詳細については、フィルタリングを参照してください。

列	説明
ロボット ID	自動的に生成されたロボットの ID。

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。ファイルを保存する新しいフォルダを作成できます。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 タブの [ロボット実行] ビューのロボット名の一部として表示されます。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。ロボットがリポジトリに存在しないタイプまたはスニペットを使用する場合、その名前は赤くマークされます。
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。
削除	クリックすると、リポジトリからロボットが削除されます。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。
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、各タイプに関連した ".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 からロボットを実行する際、これらのタイプの値を返すことがあります。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	ロボットに関連付けられたトリガーの名前。
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。ロボットがスニペット A を使用し、スニペット A がスニペット B を使用する場合、スニペット A のみがここに一覧表示されます。
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。
作成者 (デフォルトでは非表示)	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。この機能は、スタンドアロンの Kofax RPA を実行している場合にのみ利用できます。
変更者 (デフォルトでは非表示)	ロボットを最後に変更したユーザーのユーザー名。この機能は、スタンドアロンの Kofax RPA を実行している場合にのみ利用できます。
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	ロボットのリビジョンの回数。
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。

列	説明
インポート日付	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付。
すぐに実行	このボタンをクリックすると、RoboServer のロボット実行がすぐに開始されます。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。
API	このボタンをクリックすると、RoboServer でロボットを実行するための Java または C# のコードが表示されます。
REST	これにより、REST サービスとしてロボットを呼び出すことができるウィンドウが開きます。
SOAP	これにより、SOAP からロボットを呼び出すことができるウィンドウが開きます。
ダウンロード	クリックすると、リポジトリからロボットのコピーがダウンロードされ、ファイルシステムに保存されます。

ロボットを右クリックすると、以下のメニューが開きます。

複数のロボットを選択すると、[削除]、[フォルダの設定]、[スケジュールの作成] オプションを利用できます。複数のロボットを選択して、スケジュールを作成すると、[新しいスケジュール] ダイアログ ボックスが、選択したすべてのロボットが追加された状態で開きます。この方法で追加されたロボットすべてが入力を必要とする場合は、後で追加する必要があります。

[ユーザーにマッピング]、[トリガーをサスペンド] および [トリガーのアクティブ化] は、Desktop Automation ワークフロー の トリガー を持つロボットで利用できます。

Management Console にロボットを追加するには、左上隅の [ロボットの追加] 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。新しいロボットをアップロードしても、スケジュールには自動的に追加されません。これは手動で行う必要があります。[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。

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 リポジトリに複数のプロジェクトが含まれる場合、アップロードするプロジェクトを選択するためのダイアログが表示されます。

ロボットの実行

ロボットが Management Console にアップロードされると、4 つの異なる方法でロボットを実行できます。多くの場合、ロボットはスケジュールの一部または Kapplet として実行されますが、Java/.Net API を介して、または RESTful サービスとしてプログラムで実行することもできます。

注 Java/.Net API を通して、または RESTful サービスとしてロボットを起動した場合に、Management Console にロボットを実行するために必要なスロットがなければ、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起動する場合は、スロットが十分にあるときに常に実行できるようにロボットをスケジュールする方法を確立します。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。

API

[ロボット] タブで、[API] 列をクリックして、Java または .NET のテンプレート コードを生成でき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 Developer's Guide』(Kofax RPA 開発者ガイド) の関連する章を読んで、API の仕組みを理解しておくことをお勧めします。

REST

ロボットは、RESTful サービスとして実行できます。こうすることで、あらゆるプログラミング言語や、JavaScript を使用してブラウザから直接ロボットを呼び出すことができます。

[リポジトリ**]** > **[**ロボット**]** タブには、REST という名前の列があります。この列をクリックすると、サービスとしてロボットをテストできるウィンドウが表示されます。

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。

[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およびレスポンスのフォーマットを設定できます。ただし、コードからサービスを呼び出す場合、フォーマットは `Accept` および`Content-Type` HTTP ヘッダーから制御されます。`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。

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。

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。

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。サービスがコードから呼び出される場合、開発者が URL をエンコーディングする必要があります。

注 サービスとして実行するロボットは、ロボットが API 例外を初めて生成する際に停止します。これは、スケジュール設定されたロボットとは異なります。スケジュール設定されたロボットは、ロボットによって生成された API 例外に関係なく、実行を継続します。REST サービスは、Google 検索や一文の翻訳など、短いものと考えてください。

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 の組み込み型として実行している場合は、最大 100 件のリクエスト スレッドがあります。これら 100 件のスレッドは、Management Console、Design Studio からのアップロード、リポジトリ API にアクセスするユーザーなど、すべての HTTP リクエストに使用されます。多数の同時 REST サービスを実行する必要がある場合には、スタンドアロン バージョンの Management Console を Tomcat にインストールする必要があります。そうすることで、リクエスト スレッドの数を制御できます。

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**]** タブで、[SOAP] の列をクリックして、SOAP リクエストの編集およびテストを行うためのウィンドウにアクセスします。

サービス ウィンドウの [リクエスト] ペインを使用して、リクエストを構成します。[テスト サービス] をクリックしてロボットを実行します。すると結果が [レスポンス] ペインに表示されます。

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、ロボットの myRobot は、入力変数 var1 および var2 両方のタイプに属性 attr1 および attr2 があると予測している場合、「正常」は、以下のような SOAP メッセージを予測します。

```
 <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。

SOAPAction

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/テスト値が設定されます。これは、テスト サービスをプレス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、プロジェクト名は、WSDL/SOAP メッセージでは異なることがあります。すなわち、"A-Z"、"0-9"、または "_" ではないすべての文字は、"_" で置き換えられます。
- 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。

タイプ

このサブセクションでは、Management Console リポジトリにアップロードされているタイプが一覧表示されます。[タイプ] タブの上部では、タイプを表示するプロジェクトを選択できます。

スケジュールが実行されるとき、そのスケジュールにリンクされているロボットが RoboServer 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、あるいは両方の定義としてタイプを必要とします。これらのタイプを、(ロボットと同じプロジェクト内の) リポジトリに追加する必要があります。タイプを追加しない場合、ロボットは失敗します。

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、Design Studio においてタイプに後で変更が行われた場合は、タイプを再びアップロードする必要があります。各タイプ名は (各プロジェクト内において) 一意である必要があるため、同じタイプを再びアップロードすると、以前のバージョンが上書きされます。

タイプからデータベース テーブルを作成できます。

同じ名前を持つ複数のタイプのアップロードは、別のプロジェクトに配置する場合にのみ可能です。

注 [フィルタ] テキスト ボックスのフィルタを適用して、テーブルのアイテム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。

以下の情報は、各タイプに表示されます。

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。ファイルを保存する新しいフォルダを作成できます。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。

列	説明
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)。
削除	クリックすると、リポジトリからタイプが削除されます。ファイル システムにタイプのコピーがない場合は、完全に失われます。
最終変更日	このタイプの最終変更のタイム スタンプ。
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。
インポート日付	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付。
作成者 (デフォルトでは表示なし)	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。この機能は、LDAP を使用してスタンドアロンの Kofax RPA を実行している場合にのみ利用できます。
変更者 (デフォルトでは表示なし)	タイプを最後に変更したユーザーのユーザー名。この機能は、LDAP を使用してスタンドアロンの Kofax RPA を実行している場合にのみ利用できます。
ダウンロード	クリックすると、リポジトリからタイプのコピーがダウンロードされ、ファイル システムに保存されます。

1. Management Console にタイプを追加するには、左上隅の [タイプの追加] をクリックします。
   「タイプのアップロード」ウィンドウが開きます。
   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、[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。
   [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。
   タイプは暗黙的に Design Studio によってアップロードされることがあります。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. タイプを除去するには、[削除] の列をクリックします。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クリプトによって使用されている場合、確認メッセージには、使用率カウントが含まれます。ロボットまたはスクリプ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、そのスクリプトを使用しているロボット) は実行できなくなります。空のフォルダも削除できます。

タイプからのデータベース テーブルの生成

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、以下の手順に従います。

1. [リポジトリ] タブ下の [タイプ] タブで 1 つ以上のタイプを選択します。
2. 選択したタイプを右クリックして、[データベース テーブルの生成] を選択します。
   [テーブル生成] ウィンドウが表示されます。

3. [テーブル生成] ウィンドウで、プロジェクトで定義された設定済みデータベース マッピングの 1 つを選択し、SQL コードを生成してテーブルをドロップ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] を選択した場合はドロップする) ための SQL コードを含む、SQL エディターが開きます。コードを編集して、[**SQL** 実行] をクリックします。

   SQL コードが実行されると、実行状態 (説明を含む、成功または失敗) とともにメッセージが発行されます。[**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

このサブセクションは、Management Console リポジトリにアップロードされているスニペットを一覧表示します。[スニペット] タブの上部では、スニペットを表示するプロジェクトを選択できます。

スケジュールが実行されるとき、そのスケジュールにリンクされているロボットが RoboServer で実行されます。一部のロボットはスニペットを使用します。これは、リポジトリで利用できる必要があります (ロボットと同じプロジェクト内)。そうでなければ、ロボットは失敗します。

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。よって、Design Studio においてスニペットに後で変更が行われた場合は、スニペットを再びアップロードする必要があります。各スニペット名は (各プロジェクト内において) 一意である必要があります。そのため、同じ名前のスニペットを再びアップロードすると、以前のバージョンが上書きされます。

注 [フィルタ] テキスト ボックスのフィルタを適用して、テーブルのアイテム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。

以下の情報は、各スニペットに表示されます。

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。ファイルを保存する新しいフォルダを作成できます。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。
返されたタイプ	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。
使用したスニペット	このスニペットによって使用されたスニペットの名前。スニペットがスニペット A を使用し、スニペット A がスニペット B を使用する場合、スニペット A のみがここに一覧表示されます。
サイズ	スニペットのサイズはバイトで表記されます。

名前	説明
削除	クリックすると、リポジトリからスニペットが削除されます。ファイル システムにスニペットのコピーがない場合は、完全に失われます。
作成者 (デフォルトでは非表示)	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。この機能は、LDAP を使用してスタンドアロンの Kofax RPA を実行している場合にのみ利用できます。
変更者 (デフォルトでは非表示)	スニペットを最後に変更したユーザーのユーザー名。この機能は、LDAP を使用してスタンドアロンの Kofax RPA を実行している場合にのみ利用できます。
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。
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付。
ダウンロード	クリックすると、リポジトリからスニペットのコピーが取得され、ファイル システムに保存されます。

1. スニペットを追加するには、左上隅の [スニペットの追加] をクリックします。

   「スニペットのアップロード」ウィンドウが開きます。

   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、[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。

   [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。

   スニペットは暗黙的に Design Studio によってアップロードされることがあります。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. スニペットを除去するには、[削除] の列をクリックします。

   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、使用率カウントが含まれます。スニペットが、その他のスニペットまたはロボットによって使用されている場合、確認メッセージには、使用率カウントが含まれます。ロボット (またはこのロボットによって使用されているスニペット) によって使用されているスニペットを削除する場合、そのロボットは実行できなくなります。必要なスニペットが欠如しているロボットは、[ロボット] タブにおいて赤いフォントの名前でリストされます。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。

リソース

このタブでは、Management Console 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、ロボットでローカル ファイルを使用する を参照してください。)

注 [フィルタ] テキスト ボックスのフィルタを適用して、テーブルのアイテム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。

以下の情報が、各リソースに表示されます。

列	説明
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。ファイルを保存する新しいフォルダを作成できます。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。
名前	リソースの名前。
サイズ	リソースのサイズはバイトで表記されます。
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)。
作成者 (デフォルトでは表示なし)	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。個別のユーザー ログインのあるスタンドアロン バージョンの Kofax RPA を実行している際は、この列には関連情報のみが含まれます。
変更者 (デフォルトでは表示なし)	リソースを最後に変更したユーザーのユーザー名。個別のユーザー ログインのあるスタンドアロン バージョンの Kofax RPA を実行している際は、この列には関連情報のみが含まれます。
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	リソースのリビジョンの回数。
最終変更日	リソースの最終変更のタイム スタンプ。
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。
インポート日付	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付。
削除	クリックすると、リポジトリからリソースが削除されます。ファイル システムにリソースのコピーがない場合は、完全に失われます。
ダウンロード	クリックすると、リポジトリからリソースのコピーがダウンロードされ、ファイル システムに保存されます。

1. リソースを追加するには、左上隅の [リソースの追加] をクリックします。

   [リソースのアップロード] ウィンドウが開きます。

   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。

   [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。

   スケジュール設定されたロボットの入力値を構成するためのダイアログボックスに追加されたリソース、または Design Studio から直接追加されたリソースがアップロードされます。
2. リソースを除去するには、[削除] の列をクリックします。

デバイス マッピング

このタブには、ロボットで使用できるマッピングされたオートメーション デバイスが表示されます。Design Studio のこのタブでデバイスへのマッピングを作成できます。

注 [フィルタ] テキスト ボックスのフィルタを適用して、テーブルのアイテム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。

各デバイス マッピングに対して、次の情報が表示されます。

列	説明
マッピング	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することができます。
編集	をクリックして、[デバイス マッピングの編集] ダイアログ ボックスでマッピング設定を変更します。
削除	をクリックして、Management Console からマッピングを削除します。

新しいオートメーション デバイス マッピングを作成するには、[デバイス マッピング] タブで [新しいデバイス マッピング] をクリックして、新しいマッピング名とラベルを指定し、[保存] をクリックします。

仕組み

Desktop Automation ロボットが起動したとき、[Management Console] > [管理] > [デバイス] にリストされているデバイスの IP アドレスまたはホスト名には直接アクセスすることができません。

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。同じラベルの複数のデバイスが利用可能な場合、ロボットは現在 [利用可能] としてリストされているデバイスの 1 つをランダムに選択します。

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、毎回同じデバイスを返す可能性があります (利用可能な場合)。Management Console もロボットも、順番を強制または制御することはありません。

ラベルを使用すると、ロボットは単一のデバイスに結び付けられていないため冗長性が提供されます (そのためデバイスが停止しているか使用中の場合はエラーが発生します)。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ている任意のデバイスを使用することができます。

トリガー マッピング

このタブには、トリガーとトリガー マッピングを持つロボットが表示されます。トリガーは、Desktop Automation ワークフローのアテンデッド オートメーション機能の一部です。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。

注 [フィルタ] テキスト ボックスのフィルタを適用して、テーブルのアイテム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。

タブは次の 2 つのペインに分割されます。ユーザーとラベル。ユーザー ペインには、割り当てられたユーザー マッピングを持つロボットが表示されます。ラベル ペインには、ラベルが割り当てられた口

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。各マッピングに対して、次の情報が表示されます。

列	説明
ロボット名	ロボットの名前。
トリガー	利用可能なトリガー名。
プロジェクト名	Kofax RPA プロジェクトの名前。
ユーザー名	ロボットがマッピングされているユーザーの名前。
ラベル	ロボットがマッピングされているラベルの名前。
編集	をクリックして、ウィザード内のマッピング設定を変更します。
削除	をクリックして、Management Console からマッピングを削除します。

トリガー マッピングの使用

- 新しいマッピングを作成するには、[トリガー マッピング] タブ [新しいトリガー ユーザー マッピング] または [新しいトリガー ラベル マッピング] をクリックし、ウィザードの手順に従って新しいマッピングを作成します。
- ウィザード内のマッピング設定を変更するには、ロボットを右クリックして [編集] を選択するか、または [編集] 列で をクリックします。
- マッピングを削除するには、ロボットを右クリックして [削除] を選択するか、または [削除] 列で をクリックします。
- ロボット内でトリガーを無効にするには、トリガーを持つロボットを 1 つ以上選択し、選択したロボットを右クリックして、[トリガーをサスペンド] をクリックします。
- ロボット内でトリガーを有効にするには、トリガーを持つロボットを 1 つ以上選択し、選択したロボットを右クリックして、[トリガーのアクティブ化] をクリックします。

データベース

このタブには、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、クラスタ内の異なるデータベースにロボットをリンクすることができ、このタブでは新しいマッピングを作成できます。データベース マッピングには、プロジェクト スコープがあります。1 つのプロジェクトに、同じ名前を持つデータベース マッピングを指定することはできません。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。

重要 Management Console のデータベース マッピングは、バージョン 9.6.0 で導入されました。以前のバージョンの RoboServer は使用できませんが、代わりにデータベースへの接続には 9.6 以前のバージョンのメカニズムを使用します。

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。

注 [フィルタ] テキスト ボックスのフィルタを適用して、テーブルのアイテム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。

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	オブジェクトがマッピングされているデータベースの名前。
クラスタ	オブジェクトがマッピングされているクラスタの名前。
編集	をクリックして、[データベース マッピングの編集] ダイアログ ボックスにおいてマッピング設定を変更します。
削除	をクリックして、Management Console からマッピングを削除します。 注 データベースの最終マッピングを削除すると、このデータベースは、クラスタが共有データベースを Design Studio に提供するように選択されていても Design Studio で利用できなくなります。

データベース マッピングについての詳細は、Design Studio セクションのデータベースのマッピングを参照してください。

OAuth

このタブには、Management Console を使用して認証された OAuth アプリケーションおよびユーザーが含まれます。ユーザーの資格情報は、スケジュールにおけるロボットに対する入力として使用できます。これにより、ユーザー名およびパスワードなしで、認証されているユーザーに代わって 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、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。

各アプリケーションに対して、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)。
編集	クリックして、追加されたアプリケーションを編集します。
削除	クリックすると、リポジトリからアプリケーションが削除されます。
ユーザーの追加	クリックすると、リポジトリにアプリケーション ユーザーが追加されます。
変更者 (デフォルトでは表示なし)	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。個別のユーザー ログインのあるスタンドアロン バージョンの Kofax RPA を実行している際は、この列には関連情報のみが含まれます。
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。

各ユーザーに対して、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。
削除	クリックすると、リポジトリからユーザーが削除されます。

パスワード ストア

パスワード ストアは、機密情報を開示せずに各種システムにアクセスを付与するために設計されています。このタブでは、利用でき、割り当てられているパスワード ストア エントリを表示します。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。新しいパスワード アクセス エントリを作成することで、Design Studio および Management Console ロボットへのアクセスを付与できます。

このタブには、2 つのペインが含まれます。

- パスワード
- パスワード アクセス

パスワード ストアの使用

以下は、 パスワード ストア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]** > **[パスワード ストア]** タブのパスワード ペインでパスワード エントリを作成します。
2. Design Studio の設定ウィンドウの Management Console タブで、パスワードを保存する Management Console を指定して、[パスワード ストアとして使用] を選択します。
3. ユーザー提供の Design Studio のアクセス トークンを使用して **[リポジトリ]** > **[パスワード ストア]** タブの [パスワード アクセス] ペインで新しいパスワード アクセス エントリを作成します。
   Design Studio で、アクセス トークンは [バージョン情報] ウィンドウからコピーできます。
4. ロボットの展開準備が整ったら、Management Console にアップロードします。ロボットをアップロードする際、セキュリティ チェックを実行し、アップロードしたロボット用の新しいパスワード アクセス エントリを作成します。

   重要 Management Console で、ロボットや、そのタイプ、スニペットなどのコンポーネントをアップロードする際には毎回、新しいパスワード アクセス エントリをロボットに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

このペインでは、利用できるパスワード エントリが一覧表示され、新規エントリの作成および既存のエントリの編集を支援します。

新しいパスワード入力の作成
新規パスワードを作成するには、[新しいパスワード エントリ] をクリックして、以下のフィールドを記入します。

- 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。

> 注 パスワード取得ステップを挿入する際には、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] の値と一致する必要があります。

- パスワード：パスワードを入力して、ターゲット システムにアクセスします。
- 再入力：パスワードを再入力します。

パスワード エントリの [編集] ウィンドウには、[新しいパスワード エントリ] と同じフィールドが含まれます。

プロジェクトを移動する
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、プロジェクト間においてパスワード エントリを移動することでプロジェクトのパスワード エントリを管理でき、ターゲット システムにアクセスを付与することができます。あるプロジェクトから別のプロジェクトへとパスワード エントリを移動するには、[プロジェクトを移動する] をクリックします。し、[移動] ウィンドウでターゲット プロジェクトを指定します。[移動] ウィンドウの [エントリの結合] オプションは、同一のユーザー名およびターゲット システムとエントリのマージを支援します。

パスワード アクセス

このペインでは、ロボットに割り当てられているパスワード エントリを一覧表示します。

Design Studio ロボットの新しいパスワード アクセス エントリを作成
新規アクセス エントリを作成するには、[新しいパスワード アクセス エントリ] をクリックして、以下のフィールドを記入します。

- アクセス トークン: Design Studio 内の [バージョン情報] ウィンドウにあるトークンを入力します。
- 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：[利用可能] リストの 1 つ以上のパスワード エントリを選択し、これらを [割り当て済み] リストに矢印を使用して追加します。

アップロードされたロボットに対するパスワード アクセス エントリの新規作成または編集
新しいパスワード アクセス エントリをアップロードされたロボットに作成する、または既存のものを編集するには、**[リポジトリ] > [ロボット]** タブのロボットを右クリックして、[ロボットのパスワード アクセスの追加/編集] を選択します。以下のフィールドを記入します。

- パスワード アクセス：トークンは Management Console によって自動的に生成されます。これは変更しないでください。
- 説明：リスト内のエントリを識別するための名前を指定します。
- パスワード エントリ：[利用可能] リストの 1 つ以上のパスワード エントリを選択し、これらを [割り当て済み] リストに矢印を使用して追加します。

CyberArk

CyberArk は、Kofax RPA でサポートされてい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。

注 CyberArk をインストールするには、『Kofax RPA Installation Guide』(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 の設定」および「CyberArk のキー ストア」を参照してください。

CyberArk はパスワード ストアと同様に、機密情報を開示せずに各種システムにアクセスを付与するために設計されています。このタブには、割り当てられている利用可能なパスワード ストア エントリが表示されます。新規エントリのマッピングおよび既存のエントリの編集を行うことができます。新しいパスワード アクセス エントリを作成することで、Design Studio および Management Console ロボットへのアクセスを付与できます。

このタブには、2 つのペインが含まれます。

- パスワード
- パスワード アクセス

CyberArk の使用

CyberArk 環境が設定されている場合は、次に示す一般的な CyberArk の使用手順に従うことができます。

1. Management Console で、**[リポジトリ]** > **[CyberArk]** タブのパスワード ペインでパスワード エントリを作成します。
2. Design Studio で **[設定]** > **[Management Console]** に移動し、パスワードを保存する Management Console を指定して、[プライマリとして使用] を選択します。
3. Management Console の **[リポジトリ]** > **[CyberArk]** タブで、ユーザーが指定した Design Studio のアクセス トークンを使用して新しいパスワード アクセスを使用します。
   Design Studio で、アクセス トークンは [バージョン情報] ウィンドウからコピーできます。
4. ロボットの展開準備が整ったら、Management Console にアップロードします。ロボットをアップロードする際、セキュリティ チェックを実行し、アップロードしたロボット用の新しいパスワード アクセス エントリを作成します。

   重要 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、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

このペインでは、利用できるパスワード エントリが一覧表示され、新規エントリの作成および既存のエントリの編集を支援します。

新しいパスワード入力の作成
新規パスワードを作成するには、[新しいパスワード エントリ] をクリックして、以下のフィールドを記入します。

- 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。

> 注 パスワード取得ステップを挿入する際には、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] の値と一致する必要があります。

- 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。

パスワード エントリの [編集] ウィンドウには、[新しいパスワード エントリ] と同じフィールドが含まれます。

プロジェクトを移動する
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、プロジェクト間においてパスワード エントリを移動することでプロジェクトのパスワード エントリを管理でき、ターゲット システムにアクセスを付与することができます。あるプロジェクトから別のプロジェクトへとパスワード エントリを移動するには、[プロジェクトを移動する] をクリックします。し、[移動] ウィンドウでターゲット プロジェクトを指定します。[移動] ウィンドウの [エントリの結合] オプションは、同一のユーザー名およびターゲット システムとエントリのマージを支援します。

パスワード アクセス

このペインでは、ロボットまたは Design Studio に割り当てられているパスワード エントリを一覧表示します。

Design Studio ロボットの新しいパスワード アクセス エントリを作成
新規アクセス エントリを作成するには、[新しいパスワード アクセス エントリ] をクリックして、以下のフィールドに入力します。

- アクセス トークン: Design Studio 内の [バージョン情報] ウィンドウにあるトークンを入力します。
- 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：[利用可能] リストの 1 つ以上のパスワード エントリを選択し、これらを [割り当て済み] リストに矢印を使用して追加します。

アップロードされたロボットに対するパスワード アクセス エントリの新規作成または編集
新しいパスワード アクセス エントリをアップロードされたロボットに作成する、または既存のものを編集するには、**[リポジトリ] > [ロボット]** タブのロボットを右クリックして、[ロボットのパスワード アクセスの追加/編集] を選択します。以下のフィールドを記入します。

- パスワード アクセス：トークンは Management Console によって自動的に生成されます。これは変更しないでください。
- 説明：リスト内のエントリを識別するための名前を指定します。
- パスワード エントリ：[利用可能] リストの 1 つ以上のパスワード エントリを選択し、これらを [割り当て済み] リストに矢印を使用して追加します。

ロボット ファイル システム

[ロボット ファイル システム] タブは、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ているファイル システムのリストを表示します。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。

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ついての詳細は、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の『Set up Robot File System server』 (ロボット ファイル システム サーバーを設定する) を参照してください。

重要 デフォルトでは、**admin**、**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の管理を参照してください。

各システムに対して、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。
削除	クリックして、Management Console からファイル システム設定を削除します。
編集	クリックしてファイル システム設定を編集します。

1. 新規ファイル システムを作成するには、左上隅の [追加] をクリックします。
2. プロジェクトを選択して [**OK**] をクリックします。
   設定ダイアログ ボックスが開きます。
3. 左側の [名前] フィールドに、新しいファイル システム設定の名前を入力します。
   例: `RFS1`

4. [パス] フィールドでファイル システムへのパスを指定します。
   - Windows では、パスは次のようになります。

     `Folder\Subfolder1\Subfolder2`
   - Windows ファイル共有では、パスは次のようになります。

     `\\WindowsServer\FileShareName\Folder`

     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 システムにマップされます。
   - SFTP、FTP、または FTPSサーバーの場合、パスはそれぞれ `sftp://`、`ftp://`、または `ftps://` で始まり 、次のようにする必要があります。

     `sftp://website.com:9551/guest`
5. [ユーザー名] および [パスワード] フィールドに、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。
6. ファイルシステムにアクセスできる口ボットを設定します。口ボットは Management Console と同期している必要があります。
   - プロジェクトのすべての口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、[プロジェクト範囲] を選択します。
   - プロジェクト内の特定の口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、口ボットへのアクセス トークンを追加します。
     - 口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口ボット名を使用します。この方法は口ボット名が変わらない限り機能します。
       a. [口ボット マッピング] タブで口ボットを選択します。

          選択した口ボット名の左にチェック マークが表示されます。
       b. 変更を保存します。
     - 現在のバージョン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し、口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、口ボット 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口ボットに対応するため、口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. 設定ダイアログ ボックスを閉じます。
       b. [口ボット] タブのManagement Console メイン ウィンドウで、口ボットのパスワード アクセスの追加/編集.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ます。
       c. アクセス トークンの値をコピーし、ダイアログ ボックスを閉じます。
       d. 作業中のファイル システムに対する設定ダイアログ ボックスを再度開きます。
       e. [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。
       f. 変更を保存します。

7. ロボットのファイル システムに対して読み取り/書き込みを実行できるように、ファイル システムに許可された Design Studio ユーザーを設定することができます。

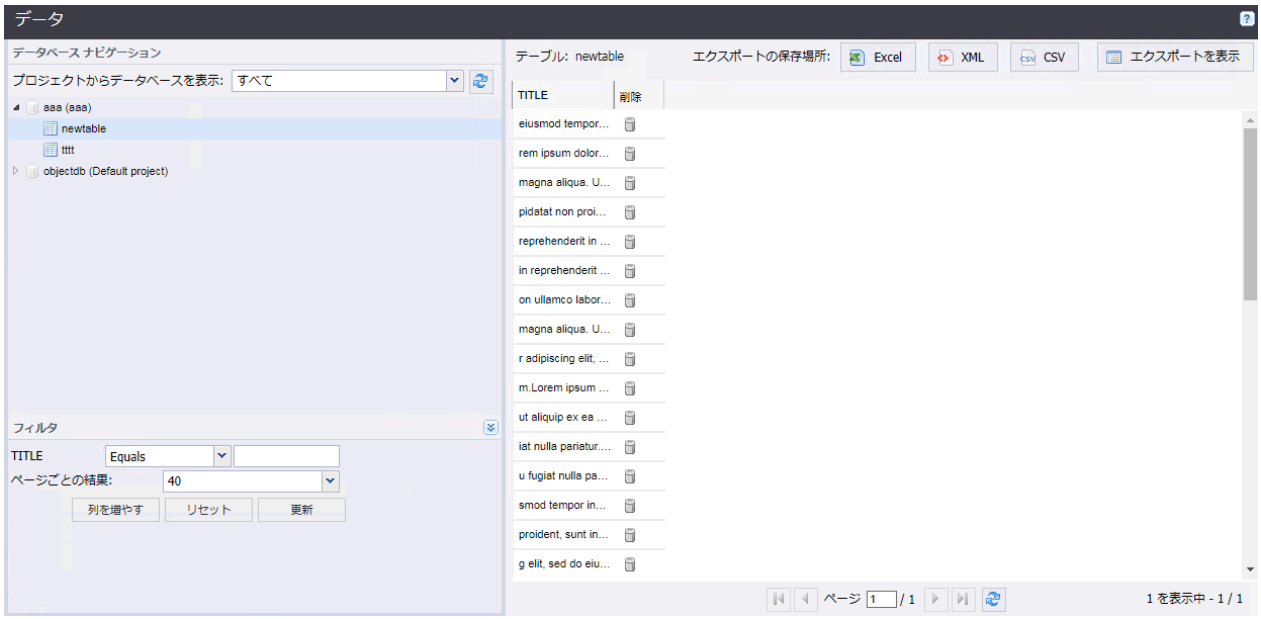
   ファイル システムに特定の Design Studio ユーザーがアクセスできるようにするには、[認証されたアクセス トークン] タブでユーザーのアクセス トークンを追加します。

   トークンは、ユーザーが Design Studio 内の **[ヘルプ]** > **[バージョン情報]** ウィンドウからコピーし、管理者と共有する必要があります。

8. 変更を保存します。

データ ビュー

データ ビューでは、ロボットが抽出したデータを表示したり、エクスポートしたりできます。これは、Design Studio または Management Console のいずれかを使用して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場合にのみ使用できます。



データ ビューの左上には、[データベース ナビゲーション] ツリーとプロジェクト セレクタが表示されます。各プロジェクトのデータベースからのデータを表示することができます。任意のプロジェクトからのデータを表示するには、プロジェクトを選択します。そのプロジェクトで定義されているデータベース マッピングが [データベース ナビゲーション] ツリーに表示されます。プロジェクト セレクタの横には、データベースが変更されたときに使用する更新ボタンがあります。データベースが変更された場合、更新ボタンにより [データベース ナビゲーション] ツリーが再構築され、新しい情報が表示されます。

データベース マッピング名をクリックすると、ツリーが開き、データベースのさまざまなスキーマが表示されます。スキーマをクリックすると、そのスキーマの Kofax RPA テーブルが表示されます。テーブルをクリックすると、そのテーブルのコンテンツが左ペインのデータ グリッドにロードされます。データがロードされると、[フィルタ] ウィンドウで多くのフィルタが表示されます。これらのフィルタは、[ログ] タブのフィルタとまったく同じように機能します。`binary` および `longtext` 属性をフィルタすることはできません。

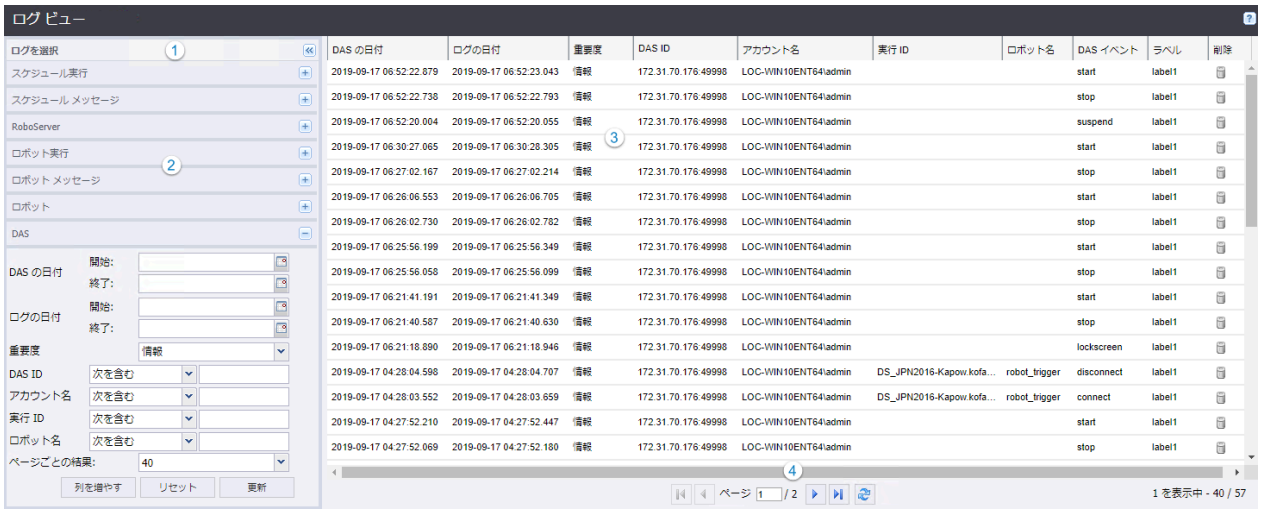
[削除] 列をクリックすると、ウィンドウが開き、テーブルの 1 つ以上の行を削除できます。行をダブルクリックすると、その行のコンテンツが含まれたウィンドウが開き、データをコピーできるようになります。

データ グリッドの上には、4 つのボタンが含まれたエクスポート バーがあります。左側の 3 つのボタンを使用して、テーブル データを Excel、XML、または CSV 形式でそれぞれエクスポートできます。右側のボタンを使用して、前回のエクスポートを表示し、これらのデータを再度ダウンロードすることができます。Management Console を次回起動すると、エクスポートが自動的に削除され、エクスポートの数が 100 を超えると、エクスポートが古い順に削除されます。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

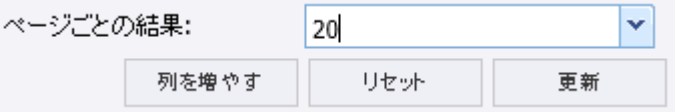
このタブは、Management Console および RoboServer のデータベース ロギングによって作成されたログの表示に使用されます。スケジュール実行およびスケジュール メッセージは、スケジュールに基づいて情報を報告する Management Console のログです。残りのログは、RoboServers のステータス情報およびロボットならびにロボット実行の情報を含む RoboServer ログです。Management Console ログおよび RoboServer ログは両者ともロギング データベースに書き込まれます。そのため、ログインデータベースを Management Console 設定 (ログ データベースを参照) でセットアップし、データベース ロギングをロギング クラスタ設定から RoboServer で有効化する必要があります。



1	ログ ペインを選択
2	ログ フィルタ
3	ホバーして詳細表示
4	ページャー

ログの隣にある アイコンをクリックして表示するには、ログ ビューを 1 つ選択します。

各ログのページ レイアウトは同一です。 をクリックすると、左ペインに複数のフィルタが表示され、ログ記録されたデータが右のグリッドにロードされます。[ページごとの結果] フィールドは、フィルタ下で利用できます。



- [列を増やす] は、右のグリッドへの列の追加と除去を行います。
- [リセット] は、入力されているフィルタ設定を消去します。
- [更新] は、フィルタの設定に基づいてデータをグリッドにロードします。
- リスト ボックスは、ページごとの結果数を制御します (次の更新)。

  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。任意のフィルタ テキスト フィールドで Enter を押すと、グリッドの更新がトリガされます。

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することで、RoboServer ログデータベースのログの保持ポリシーをセットアップできます。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 メッセージや、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。

[削除] の列をクリックすると、実行とメッセージを削除するためのウィンドウが開きます。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。

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。これにより、スケジュールの実行に失敗した理由を見ることができます。

注 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 [ロボット ビュー エラーの表示] を選択します。

[削除] の列をクリックすることで、1 つ以上の記録を削除することができます。こうすることで、このメッセージのみの削除、現在のフィルタに一致するメッセージすべての削除、またはすべての RoboServer ログメッセージの削除を選択できるウィンドウが開きます (500 件を超える一致結果がある場合には、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化されます。これはパフォーマンスが原因です)。

注 スケジュール実行およびスケジュール メッセージを表示するには、ログ データベースをセットアップし (ログ データベースを参照)、ロギング クラスタ設定から RoboServer でのロギングを有効化します。

RoboServer

RoboServer または Management Console から一般的なメッセージを表示します。ダブルクリックまたはコンテキスト メニューを使用して、別のウィンドウでメッセージを開きます。これにより、エラー メッセージのコピーが容易になります。

[削除] の列をクリックすることで、1 つ以上の記録を削除することができます。こうすることで、このメッセージのみの削除、現在のフィルタに一致するメッセージすべての削除、またはすべての RoboServer ログメッセージの削除を選択できるウィンドウが開きます (500 件を超える一致結果がある場合には、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化されます。これはパフォーマンスが原因です)。

ロボット実行

各ロボット実行の情報を表示します。このログには、デフォルトでは表示されない、その他のフィールドが含まれています。ただし、フィルタの下の [列の追加] をクリックすることで追加できます。行をダブルクリックすると、この実行中にログの記録されたメッセージすべてに移動します。これは、コンテキスト メニューからも利用できます。コンテキスト メニューでは、同じロボットの実行すべてを表示することや、この実行が実行されたときに、このロボットに指定された入力を表示できます。

[削除] の列をクリックすると、ウィンドウが開いて、実行とメッセージを削除できるようになります。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。行をダブルクリックすると、エラー メッセージを簡単にコピーできるウィンドウが開きます。コンテキスト メニューを使用することで、このメッセージが属する実行に移動できます。

ロボット エラーについては、データ グリッドの [ロケーション コード] の列にリンクが含まれています。リンクをクリックすると、小さな .robotDebug ファイルがダウンロードされます。このファイルを Design Studio で開くと、ロボットがロードされ、実行の入力が設定されます。また、ロボットはエラーの場所へと移動し、エラーを素早くデバッグできるようになります。

ロボット

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。行をダブルクリックするか、コンテキストメニューを使用すると、指定のロボットの実行すべてに移動します。

DAS

Desktop Automation サービスから受信したイベントを表示します。このログに表示された情報でリストをフィルタできます。詳細については、Desktop Automation サービスのロギングを参照してください。

管理

このタブを使用して、Management Console を管理します。

タスク ビュー

Management Console によって実行され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。

デバイス

Management Console で登録された利用可能なオートメーション デバイスを表示します。

プロジェクト

プロジェクトを作成および削除します。

ユーザー

Management Console ユーザーに関する情報を表示します。

設定

Management Console 設定を構成します。

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

タスク表示では、スケジュール設定されたロボットと同様、Management Console のスケジューラーによって起動された事前処理のタスクおよび事後処理のタスクが表示されます。タスク表示では、すべてのサーバーおよびロボットにわたる現在のアクティビティの概要が表示されます。以前のロボット実行の結果について知りたい場合は、ログを調べる必要があります。

更新の遅延により、短期実行ロボットは、このタブには表示されないことがあります (ほぼされることはありません)。

以下の情報は、現在実行しているすべてのロボットに表示されます。

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	タスクが属す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。
最後のステータス変更	最終ステータス変更の時間。
停止	クリックすると、タスクの実行が停止していなくてもタスクを停止します。
入力値	この列では、実行している口ボットの入力値概要が表示されます。

注 列をポイントすると、ポップアップ テキストが、利用できる情報すべてを表示します。また、実際の列のテキストは通常、スペースの不足により省略されて表示されます。

RoboServer

このセクションでは、Management Console で認識されているクラスタと RoboServer を管理する方法を説明します。デフォルトでは、リストは、1 つの RoboServer を含む 1 つのクラスタが含まれています。これは、Management Console 機能を実行している RoboServer です。複数の RoboServer とクラスタが含まれる大きな設定では、スタンドアロンの Web コンテナ (ライセンスで許可される場合) または口ボットの実行に使用されていない RoboServer に Management Console をデプロイ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 Administrator's Guide』(Kofax RPA 管理者ガイド) を参照してください。

各サーバーに次の情報が表示されます。バージョン 9.4 よりも前の RoboServer では、一部の情報が利用できないことに注意してください。

列	説明
クラスタ/サーバー	クラスタの場合 クラスタの名前。クラスタが SSL を使用している場合には末尾に SSL が付きます。クラスタに未適用の設定がある場合、これらの設定は青で表示されます。クラスタに無効な設定がある場合、その名前は赤で表示されます。 **RoboServer** の場合 RoboServer の名前とポート。RoboServer が正しく設定されていない場合、その名前は赤で表示されます。赤の名前にカーソルを合わせると、ツールチップでエラーが示されます。
バージョン	実行中の RoboServer 上のソフトウェアのバージョン。
CRE/KCU	表示される列名は、使用されるライセンス タイプによって異なります。CRE および KCU ライセンスの詳細については、『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) の「Concurrent Robot Execution License」(同時ロボット実行ライセンス) および「Kofax RPA Compute Units」(Kofax RPA 計算単位) セクションを参照してください。 **CRE** 静的ライセンス配布のモードでは、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 間で均等に配布されます。CRE は整数単位であるため、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、それぞれの RoboServer は 1 つの CRE を取得します。したがって 1 つの CRE は未使用のままになります。 注 クラスタ内の CRE の数は RoboServer の CRE 以上でなければなりません。クラスタに存在する RoboServer の数よりも少ない CRE をクラスタに割り当てると、クラスタは無効になります。 CRE の数を調整するには、[**CRE** の割り当て] をクリックします。 動的ライセンスの配布モードでは、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。RoboServer は、要求された数のライセンスを取得できます (使用可能な場合)。このモードでは、RoboServers は Management Console とのみ通信し、API コールなどの他のリクエストをブロックします。 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 Administrator's Guide』(Kofax RPA 管理者ガイド) の「Production Configuration」(プロダクションの設定) セクションを参照してください。 **KCU** 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 間で均等に配布されます。クラスタで KCU を調整するには、[**KCU** の割り当て] をクリックします。
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。
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。サーバーが起動または再起動された時刻を確認できます。

列	説明
コマンド ライン (非表示)	RoboServer を起動したコマンド ライン。
CPU 数 (非表示)	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。
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。
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。
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。
最大キュー数 (非表示)	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。
ライセンス タイプ	RoboServer 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。
クラスタの状態/サーバーの状態	クラスタの場合、クラスタの状態が表示されます。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を示します。
一時プロファイリング	特定のサーバーに対してプロファイルが一時的に有効化されているかどうかを示します。サーバーが再起動されると、この設定はクリアされます。
最終更新	RoboServer から Management Console が最後にステータスの更新を受信した時刻を表示します。
設定	をクリックして、[クラスタ設定] ダイアログ ボックスでクラスタ設定を変更します。
削除	削除 をクリックして、Management Console からクラスタ/サーバーを削除します。

ロボット ランタイム ビューには、実行中のロボットに関する詳細情報が含まれています。ロボット ランタイム ビューの上部のバーには、次の情報が含まれています。

- ロボットは次で実行：- 現在、ビューが表示しているロボットのクラスタまたはサーバーを示します。
- 次でフィルタ - ロボット、プロジェクト、実行 ID でリストをフィルタしたり、空の行を選択して、フィルタを無効にしたりします。フィルタの一致では大文字と小文字が区別され、フィルタでは「ロボット」、「プロジェクト」、または「実行 ID」フィールドにサブ文字列として入力されたテキストを含むロボットが選択されます。
- ページあたりのロボット - ページに表示される ロボットの最大数を制限します。
- 更新間隔 - ビューの更新間隔 (デフォルトは 1 秒) を設定します。

列ヘッダーをクリックして、任意の列を昇順または降順で並べ替えます。デフォルトでは、クライアントでの開始時刻に基づいて並べ替えられます。

テーブルの行をダブルクリックして、ウィンドウでロボットの情報を開きます。このウィンドウには、ロボット ランタイム ビューで利用できる情報と同じ情報が表示されます。つまり、ウィンドウを開いたときの情報のスナップショットです。この情報は、ロボットが実行を停止した場合でも更新されません。

更新 をクリックして、テーブルの情報を更新します。

実行中または直近の完了したロボットに対して、次の情報がそれぞれ表示されます。次のテーブルは、1 つの RoboServer の、またはクラスタが選択されている場合はすべての RoboServer のロボットを示しています。

列	説明
ロボット	ロボットの名前
サーバー	ロボットを実行しているサーバーの名前
プロジェクト	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]** タブ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 の実行リクエストを作成するときに、file://URL または Library:/ を指定することができます。これにより、ファイル システムとライブラリのどちらからロボットをロードするかが指定されます。 • ファイル システム URL - file://C:/Kofax RPA/Robots/Library/Input.robot • ライブラリ URL - Library:/Input.robot ロボットの実行リクエストは、次のようなリクエストです。 `Request request = new Request("Library:/Input.robot")`
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 • 埋め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 Developer's Guide』(Kofax RPA 開発者ガイド) を参照してください。また、このヘルプ システムのリファレンス セクションにあるロボット ライブラリも参照してください。
クライアントでの開始時刻	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。
実行パス	ロボットが実行したステップのシーケンス。
ロケーション コード	Design Studio に表示できるステップに割り当てられているコード。
ステップの実行時間	現在のステップの実行時間 (秒単位)
実行されるステップの制限	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。

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 での実行を完了。このステータスは、次の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
	完了したステータスのロボットは、完了してから 1 分後にテーブルから除去されます。

KCU ポイント コスト	ロボットを実行するために消費した KCU ポイント。KCU ポイント コストは、Design Studio からの KCU ポイント使用量に等しくなります。
KCU 待機	KCU ポイント (その時点での) が既に消費されているためロボットが実行できなかった時間。
ロードされたバイト数	ロボットの実行中にロードされたバイト数。
抽出値の制限	オブジェクト抽出数の上限。ロボットがこのプロパティで示されている上限よりも多くのオブジェクトを抽出すると、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い場合、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。
最後の出力時刻	最後の抽出が実行された時刻。
送信した電子メール数	ロボットが送信した電子メールの数。
終了	ロ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。
出力数	ロボットが生成したオブジェクトの数。
実行されたステップ	ロボットが実行したステップの数。
接続が失われると停止	このフラグを設定した場合、Management Console への接続が失われると、ロボットが停止します。
API 例外で終了	このフラグを設定した場合、API 例外が生成されると、ロボットが停止します。
ログ	をクリックして、[ログ] タブを開きます。
終了	をクリックして、ロボット実行を終了します。
スケジュール	をクリックして、[スケジュールの編集] ダイアログ ボックスを開きます。

クラスタの作成

クラスタを作成するときに、クラスタの名前とタイプを指定します。非プロダクション クラスタを作成する場合は、非プロダクション 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、プロダクション クラスタを作成する場合は、プロダクション 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。SSL オプションを選択した場合、クラスタ内の RoboServer は SSL RQL サービスを使用する必要があります。

クラスタを作成した後、RoboServer をクラスタに追加できます。

クラスタには、さまざまな Kofax RPA 製品バージョンの RoboServer を含めることができ、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。

たとえば、異なるバージョンのロボットとともにクラスタ内で実行されている複数の RoboServer がある場合、ロボットの配布は次のようになります。

	10.2.1 RoboServer	10.2.4 RoboServer	10.3.0 RoboServer	10.4.1 RoboServer
9.7.0 Robot	一致	-	-	-
10.2.5 Robot	-	-	一致	-
10.3.0 Robot	-	-	一致	-
10.3.1 Robot	-	-	-	一致
10.5.0 Robot	-	-	-	-

上記の表に見られるように、各ロボットは、10.5 Robot を除き、同等またはそれ以上の製品バージョンの RoboServer と正常に一致します。クラスタには、一致する関連の RoboServer バージョンはありません。

コンテキスト メニュー

グリッドには、下部のメニューから利用できない、ほとんど使用されない機能があるコンテキスト メニューが含まれています。コンテキスト メニューの項目は次のとおりです。

クラスタ設定

[クラスタ設定] ダイアログ ボックスが表示されます。

RoboServer の終了

選択した RoboServer を終了/再起動できるダイアログ ボックスが表示されます。

スレッドのダンプ

選択した RoboServer にリクエストを送信し、フル スレッド ダンプを実行します。このスレッド ダンプは別個のウィンドウで開かれます。また、Management Console のスレッド ダンプを取得することもできます。詳細については、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Tomcat Management Console」(Tomcat 管理コンソール) を参照してください。

ロードの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 で既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 (クラスタ設定で指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布されます。

クラスタの状態の変更

クラスタには、以下の 2 種類の状態を設定できます。実行中または一時停止一時停止の状態は、クラスタを何も実行しない状態に設定します。これにより、特定のスケジュール内のすべてのロボットで同様の設定を確実に使用し、その設定がスケジュールの実行中に変更されないようにクラスタ設定を更新できるようにします。各状態について、以下の表で説明します。

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、新しい実行リクエストを受け入れることはできません。また、RoboServer は現在のすべての実行を終了します。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。

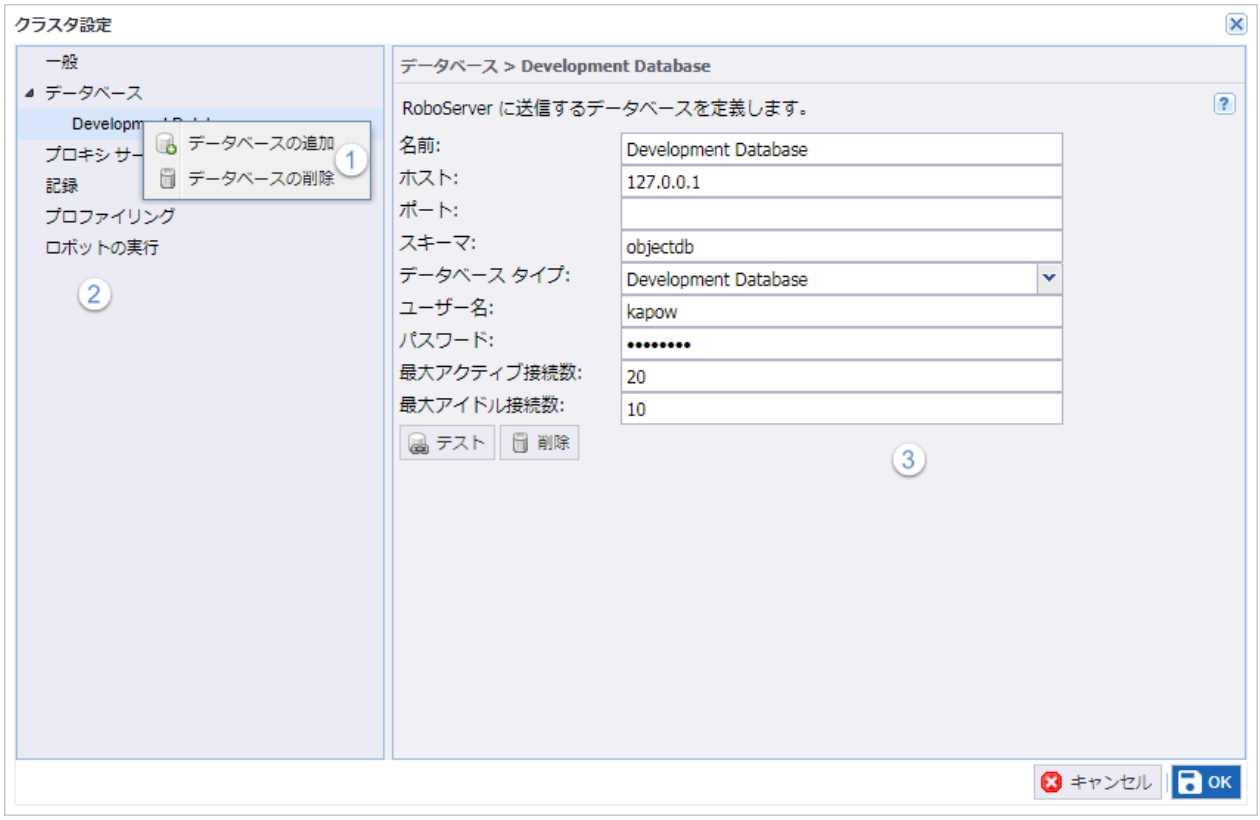
1. クラスタ リストの上の [クラスタの状態の変更] をクリックします。
   [クラスタの状態の変更] ウィンドウが表示されます。

   注 また、リストのクラスタを右クリックして、適切なサブメニュー アイテムを選択できます。

2. [クラスタ] フィールドで、変更するクラスタを選択します。
3. [クラスタの状態] フィールドで、一時停止状態に移行するためのに一時停止オプションを選択します。
   クラスタの状態は、その名前が示すように、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 口ボットが実行している際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るような場合、口ボットはその実行が開始された時点でに設定されていたデータベースを使用します。そして、口ボットは次の実行時に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、クラスタの各 RoboServer に送信されます。クラスタ設定により、RoboServer で実行する口ボットに利用可能なデータベースなどを構成できます。

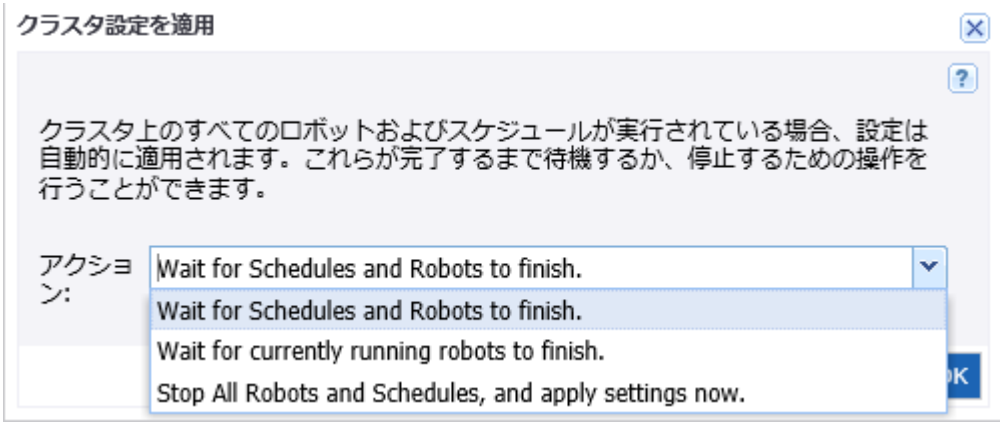


1	コンテキスト メニュー
2	利用可能な設定
3	設定ビュー

1. クラスタを右クリックして、[クラスタ設定] を選択します。
   オプションとして、[設定] の列で、 をクリックします。
   [クラスタ設定] ウィンドウが表示されます。
2. 必要に応じてオプションを構成します。

   注 クラスタ設定を変更した場合には変更が太字で表示されるため、[OK] をクリックして決定する前に、行った変更の内容を確認できます。

3. [**OK**] をクリックします。
   設定が RoboServer に送信されるようにするには、クラスタ設定を最初に適用する必要があります。クラスタを一時停止状態に設定して、設定の適用を行います (詳細については、クラスタの状態の変更を参照)。変更を行った場合は、次のウィンドウが開きます。



4. 設定の適用を行うためには、[クラスタ設定を適用] ウィンドウの「アクション」フィールドでクラスタの一時停止モードへの切り替えを選択します。
   設定が適用されたら、クラスタを実行状態に戻します。

一般

このビューを使用して、CRE ライセンスの配布と閾値の　RoboServer バージョンを定義します。

重要 [ライセンスの配布] と [閾値の RoboServer 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、または Management Console が JMS モードで実行されている場合に無効になります。JMS モードでは、ライセンスの配布は静的で、閾値の RoboServer バージョンは 11.0.0 に設定されます。

ライセンスの配布

CRE ライセンスの配布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 間で均等に配布されます。CRE は整数単位であるため、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、それぞれの RoboServer は 1 つの CRE を取得します。したがって 1 つの CRE は未使用のままになります。 注 クラスタ内の CRE の数は RoboServer の CRE 以上でなければなりません。クラスタに存在する RoboServer の数よりも少ない CRE をクラスタに割り当てると、クラスタは無効になります。 CRE の数を調整するには、[**CRE** の割り当て] をクリックします。

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。RoboServer は、要求された数のライセンスを取得できます (使用可能な場合)。このモードでは、RoboServers は Management Console とのみ通信し、API コールなどの他のリクエストをブロックします。

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。ロボットの円滑なアップグレードを実行し、ロボットがテストされていない RoboServer で実行されないようブロック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、最も近い新しい RoboServer のバージョンに転送されます。閾値を上回るロボットはすべて厳密なバージョン一致に従って、RoboServer 間で配布されます。つまり、製品バージョンが閾値より高いロボットは、同じ製品バージョンの RoboServer でのみ実行する必要があります。

たとえば、閾値が 10.2 に設定されている場合、クラスタにアップロードされた複数のランダム RoboServer 間での複数のランダム ロボット配布は次のようになります。

	10.2.0 RoboServer	10.3.0 RoboServer	10.5.0 RoboServer
9.7.0 Robot	一致	-	-
10.1.0 Robot	一致	-	-
10.3.0 Robot	-	厳密なバージョン一致	-
10.4.0 Robot	-	-	-

製品バージョンが閾値を下回っているすべてのロボットは、同等以上のバージョンの RoboServer と正常に一致されます。

10.3.0 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、厳密なバージョン一致があります。

10.4 ロボットには一致がなく、実行できません。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 がありません。

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。RoboServer のバージョン一致の詳細については、「RoboServer」トピックの「クラスタの作成」を参照してください。

クラスタ データベース

クラスタ設定 ウィンドウの [データベース] メニューで、データベースの追加または削除を行うことができます。

クラスタで定義されたデータベースは、データベース マッピングからクラスタ RoboServer で実行しているロボットに対して利用できます。データベースは、データベース タイプを使用します。データベース タイプは、Management Console 設定で定義されます。

クラスタが、リポジトリに存在するデータベースに指定されているシェア データベースおよびデータベース マッピングを Design Studio に提供するように選択されている場合は、クラスタで定義されてい

るデータベースも Design Studio で利用できます。**[リポジトリ]** > **[データベース]** タブでデータベースマッピングを確認してください。

9.2.0 以前のバージョンからアップグレードしたユーザーについては、データベースを定義しているレガシー db.properties ファイルからデータベースをインポートすることができます。このインポートを行うには、データベース ノードをツリーで選択し、[ファイルからデータベースをインポート] をクリックします。この操作により、ファイルのコンテンツを貼り付けるウィンドウが開きます。

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。

新しいデータベースを追加するには、データベース カテゴリを選択して [データベースの追加] をクリックするか、データベース サーバーのカテゴリを右クリックして [追加] をクリックします。

注 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 ドライバが必要となります。このドライバは、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。使用する Kofax RPA のドライバを Management Console に導入するには、**[管理]** > **[設定]** > **[データベース ドライバ]** > **[ドライバ Jar ファイルのアップロード]** にアップロードします。

デフォルトでは、localhost から管理者として接続している場合、ドライバ Jar ファイルを Management Console にアップロードできます (管理者として接続していない場合は、[ドライバ Jar ファイルのアップロード] ボタンが無効になります)。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 プロセスを実行しているユーザーとして RoboServer 設定を開始します。RoboServer 設定において変更を行った後は、変更を有効化するために Management Console を再起動します。

[開発用データベースの追加] 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。

1. **[クラスタ設定]** > **[データベース]** ウィンドウで、[データベースの追加] を選択します。
   オプションとして、データベースのカテゴリを右クリックして、[追加] をクリックすることができます。

   注 レガシー プロパティ ファイルからデータベースをインポートするには、データベース カテゴリのカテゴリを選択して、[ファイルからデータベースをインポート] をクリックします。
   ウィンドウが表示されたら、インポートするファイルのコンテンツを貼り付けます。

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
   - スキーマ

     注 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。

   - データベース タイプ
   - ユーザー名
   - パスワード
   - 最大アクティブ接続数
   - 最大アイドル接続数
3. [テスト] をクリックします。
4. [**OK**] をクリックします。

プロキシ サーバー

RoboServer 上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。その後の接続については、ラウンドロビンの順番でプロキシ サーバーが使用されます。

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。

新しいプロキシ サーバーを追加するには、プロキシ サーバー カテゴリを選択して、[プロキシ サーバーを追加] をクリックするか、プロキシ サーバーのカテゴリを右クリックして、[追加] をクリックします。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 サーバー設定をインポート] 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. プロキシ サーバーの [クラスタ設定] ウィンドウで、[プロキシ サーバーを追加] を選択します。

   オプションとして、プロキシ サーバーのカテゴリを右クリックして、[追加] をクリックすることができます。

   注 レガシー プロパティ ファイルからプロキシ サーバー設定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、[レガシー プロパティ ファイルからプロキシ サーバー設定をインポート] をクリックします。
   インポートするファイルのコンテンツを貼り付けるウィンドウが開きます。

2. 以下のプロキシ サーバーの詳細をすべて入力します。
   - ホスト名
   - ポート番号
   - ユーザー名
   - パスワード
   - 除外ホスト名
3. [**OK**] をクリックします。

クラスタ ログ

クラスタ RoboServer のログ オプションを有効または無効にすることができます。

データベース ログ
データベース ログが有効になっている場合、RoboServer は、Design Studio 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。[ロボット入力をデータベースにログ記録] が選択されている場合、ロボット入力はデータベースにログ記録されます。

注 データベース ログを機能させるには、RoboServer ログ データベースを設定して有効化する必要があります。データベースが設定されていない場合、クラスタ設定が無効になります (無効になったことが通知されるため、この問題は修正できます)。

log4J

log4j では、通常の log4j.properties ファイルを使用して、ログが記録される場所を制御することができます。log4j.properties ファイルは、`C:\Users\name.lastname\AppData\Local\Kofax RPA\11.0.0\Configuration` など、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。詳細については、『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) を参照してください。

デフォルトの log4j.properties ファイルの場合、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。詳細については、log4j のマニュアルを参照してください。

このログを有効にすると、RoboServer は 3 つの異なるロガーを使用してログ記録します。

ロガー	説明
log4j.logger.kapow.servermessagelog	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。
log4j.logger.kapow.robotrunlog	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。
log4j.logger.kapow.robotmessagelog	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ルが設定されている場合は、プロファイルも含まれます。

電子メール ログ

このログの場合、ロボットがエラー メッセージをログに記録したとき、またはサーバーにエラー メッセージがあるときに、電子メールが常に送信されます。

プロパティ	説明
宛先アドレス	電子メールを受け取るユーザー。"," で区切られた複数のアドレスを追加できます。
送信元アドレス	電子メールで使用される送信元アドレス。

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見ることができます。これは、速度の遅いロボットを使用していて、そのパフォーマンスを向上したい場合に非常に便利です。

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、ステップの実行時間が [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。Desktop Automation ステップでは、ステートメントの前に **DA** が付きます。 • ステップの実行時間 (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	これは、プロファイリング情報の送信先 (コンソール、ファイル、または口グ) 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。
待機メッセージでページ URL を記録	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。

Robot 実行

このセクションを使用して、RoboServer ロボット実行のプロパティを定義します。(個々の) RoboServer で同時に実行できるロボット数およびキュー登録可能数を指定します。CRE タイプの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	各 RoboServer で同時に実行できるロボットの最大数。ロボットが CPU またはメモリを著しく消耗する場合や、頻繁にメモリのしきい値に達する場合には、最大同時ロボット数を下げてください。
最大キュー格納可能ロボット数	各 RoboServer でキュー登録に存在できるロボットの最大数。 ロボットが Management Console 上のスケジュールによって開始された場合、このプロパティは適用されません。Management Console は独自の共通キューを維持します (また、利用可能なスロットがある RoboServer にロボットを割り当てます)。 ロボットが API 呼び出しで開始される場合、ロボットが各 RoboServer でキュー登録できる数が、この値で決まります。

デバイス

このタブには、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。Management Console でオートメーション デバイスを登録するには、Management Console の詳細を指定し、Desktop Automation エージェント構成ファイルで `"singleUser"` オプションを `false` に設定します。詳細については、オートメーション デバイス マッピングを参照してください。

各デバイスに対して、次の情報が表示されます。

列	説明
クラスタ/デバイス	クラスタの場合 クラスタの名前。 デバイスの場合 オートメーション デバイスの IP アドレスまたは名前のいずれかと、デバイスへの接続に使用するポートです。
ステータス	クラスタの場合 現在のクラスタの状態。状態は、次のとおりです。 • 実行中 • 一時停止中 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 エージェントが実行されていて、デバイスへの接続はありません。 • 使用中：Desktop Automation エージェントが実行されていて、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 エージェントが起動していません。

列	説明
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することができます。
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	指定のオートメーション デスクトップ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。

プロジェクト

[管理] タブの [プロジェクト] セクションを使用して、Management Console で利用できるプロジェクトを設定します。

プロジェクトは、ロボット、タイプ、スニペット、リソース、およびスケジュールを分割するための手段です。ロボットには、属するプロジェクトに含まれるタイプ、スニペット、およびリソースへのアクセス権のみがあります。ロボット、タイプなどの名前は、プロジェクト内でのみ区別できる必要があります。

注意 : プロジェクトを削除すると、関連付けされているロボット、タイプ、スニペット、リソース、およびスケジュールも削除されます。

デフォルトでは、Management Console には単一のプロジェクトが含まれます。

プロジェクトを編集するには、[詳細] ( ) をクリックするか、プロジェクトをダブルクリックします。[プロジェクトの編集] ダイアログ ボックスが開き、以下のタブが表示されます。

- 基本: プロジェクトの名前と説明を入力します。
- 権限: Management Console でのユーザー管理が有効になっている場合、プロジェクトの権限を設定します。

  スタンドアロン Web コンテナーで Management Console を展開する際は、ユーザーのグループ メンバーシップ (LDAP など) に基づいて、ユーザーのプロジェクト アクセス権限を設定することもできます。詳細については、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Project Permissions」(プロジェクト権限) の「Tomcat Management Console」(Tomcat 管理コンソール) セクションを参照してください。

  デバイス マッピングとデータベースを設定するオプションは、すべてのユーザー グループに対してデフォルトで提供されることに注目してください。
- サービス: ロボットが RESTful サービスとして呼び出される場合、クラスタを選択してこのプロジェクトでロボットを除外します。詳細については、REST および SOAP サービスを参照してください。
- リポジトリ: プロジェクトの Git バージョン コントロール システムでリポジトリの使用を有効にし、使用するリポジトリを指定します。詳細については、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Configure Robot Lifecycle Management」(ロボットのライフサイクル管理の設定) セクションを参照してください。

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。これは、プロジェクトの名前を変更した場合、このプロジェクトに属する既存のロボット実行およびロボット メッセージすべてが新しい名前を反映するために更新される必要があることを意味します。そうでなければ、ログがプロジェクトによってフィルタ処理される際に、名前が表示されなくなります。プロジェクトの名前を変更する際に、Management Console がロギング データベースに接続されている場合、Management Console はログ データベースのロボット実行/メッセージ入

力の名前を自動的に変更します。ただし、接続されていない場合や、接続が更新中に失われた場合、管理者はログ テーブルを更新するために以下の SQL を手動で実行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。

REST および SOAP サービス

ロボットは、REST または SOAP サービスとして公開することができます。**[管理]** > **[プロジェクト]** から [プロジェクトの編集] ダイアログ ボックスの [サービス] タブを開くと、これらのサービスのランタイムを設定できます。

[サービス クラスタ] は、現在のプロジェクトでの REST サービスの実行に使用されるクラスタです。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。[プロジェクトでサービス クラスタのみを使用する] を選択すると、Management Console はコードを生成する際、そしてロボット メニューからロボットを実行する際に、その他すべてのクラスタを非表示にし、すべてのスケジュールにサービス クラスタを使用します。注意 : [プロジェクトでサービス クラスタのみを使用する] を選択する際は、サービス クラスタを選択する必要があります。

デフォルトでは、REST/SOAP サービスは基本的な認証で保護されています。ブラウザから直接サービスを呼び出す際には (XMLHttpRequest を使用)、認証を無効化する必要があります。無効化しなければ、JavaScript ソース ファイルのロ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際には、認証によって保護されるサービスを使用することをお勧めします (安全な方法で資格情報を保持できることを仮定)。

サービスが呼び出される Web ページと Web サーバーが同じ場所にない限り、REST/SOAP サービスをブラウザから呼び出す場合には、特定の制限があります。別のドメインから REST/SOAP サービスを呼び出す際には (Cross-Origin Resource Sharing (CORS))、別のドメインからリソースを処理できるようにするために、クライアントに特定のヘッダーを含める必要があります。[アクセス-制御-許可オリジン] は、その一例です。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。http://example.com のページが、http://kofax.com にあるサービスにリクエストを生成する JavaScript が組み込まれたページを含む場合、http://kofax.com からのサービス レスポンスには、"Access-Control-Allow-Origin" のヘッダーが含まれる必要があります。http://example.com またはブラウザのビルトイン セキュリティ システムは、CORS (Cross Origin Response) が処理されないようにします。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。

ユーザーおよびグループの管理

このタブを使用して、Management Console およびプロジェクトにアクセスを付与できるユーザーおよび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。ユーザーを作成した後、1 つ以上のグループにユーザーを追加する必要があります。このグループは、1 つ以上のプロジェクトにおける特定のロールに関連しています。最初にグループを作成することをお勧めします。これは、ユーザーは最低でも 1 つのプロジェクト内の表示ロールが付与されているグループに追加されるまで、ログインできないためです。

注 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。名前などには、次の文字は使用できません。

" / \ [ ] : ; | = , + * ? < >

詳細については、technet.microsoft.com でユーザー アカウントおよびグループ アカウントの作成を参照してください。

Management Console では、ユーザーに割り当てることができる複数のビルトイン ロールが用意されています。ロールはセキュリティ グループのユーザーに割り当てられます。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。ビルトイン 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。

ビルトイン ロールを以下に示します。

- **RPA** 管理者: RPA 管理者グループのユーザーは、プロジェクトの新しい管理者とユーザーの作成を含むすべての権限 (特殊な管理者権限を除く) があります。

  重要 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。

- プロジェクト管理者: プロジェクト管理者ロールを持つユーザーは、所有するプロジェクトのグループにロールを割り当てる権限があります。また、RoboServer およびクラスタを変更なし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。
- 開発者: 開発者には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。このロールを持つユーザーは、スケジュールの作成、編集、削除、ロボットの実行、実行ログとクラスタの表示が可能になります。
- 閲覧者: 閲覧者には、開発者から Kapplets へのアクセス、および変更と実行の権限を除いた権利があります。
- **API** (このユーザーは API 経由でサービス認証としてのみログインします): このロールを持つユーザーは、リポジトリ API を使用して、リポジトリに対して読み書きを行うことができます。
- **DAS** クライアント ユーザー (このユーザーは API 経由でサービス認証としてのみログインします): これはリモートの Desktop Automation Service (DAS) クライアント用に作成されたユーザーで、DAS API へのアクセス権限のみを持ちます。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。
- パスワード ストア ユーザー: このアドオン ロールを持つユーザーは、パスワード ストアにアクセスできます。このロールは、開発者ロールなど、他のロールの上に提供されます。このロールにはパスワード ストア タブへのアクセス権限のみが付与されます。
- **VCS** サービス ユーザー (このユーザーは API 経由でサービス認証としてのみログインします): バージョン コントロール サービス (Version control service) ユーザーには、Synchronizer の権限の特別なセットが付与されます。このロールにはリソースを追加、編集、削除する権利があります。他のユーザーの代理で展開を実行してバージョン コントロール サービスで「deployer」機能を使用できるのはこのロールのみです。
- **RoboServer** (このユーザーは API 経由でサービス認証としてのみログインします): リポジトリからのみ読み取ることができる制限付きのユーザー。このロールは、RoboServer がクラスタにアクセスしたり、リポジトリ アイテムを取得したりする場合や、パスワードをパスワード ストアにリクエストする際に使用します。
- **Kapplet** 管理者: Kapplet を作成、表示、実行、編集できるユーザー。

- **Kapplet** ユーザー: Kapplet を表示、実行できるユーザー。このロールを持つユーザーは、他の権限がないかぎり、Management Console にアクセスすることはできません。
- **Process Discovery** クライアント (このユーザーは API 経由でサービス認証としてのみログインします): このロールは Process Discovery コンポーネントと Management Console の通信を許可します。

admin ユーザー

admin は RPA 管理者グループのメンバーではありませんが、常に任意のアクセス権限を持ちます。

LDAP 統合設定では、**admin** グループは LDAP 構成の一部として定義されます。**admin** は、ログインおよび開発者、管理者、RoboServer などのロールにマッピングする LDAP グループの定義が可能です。

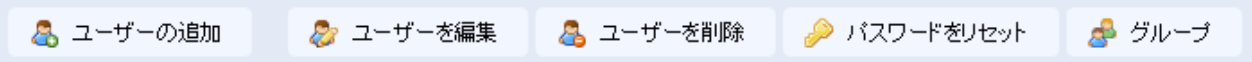
内部ユーザー設定では、**admin** ユーザーは初期開始時に作成され、ログインおよび管理者、開発者などのユーザーの作成が可能です。

admin の特別権限

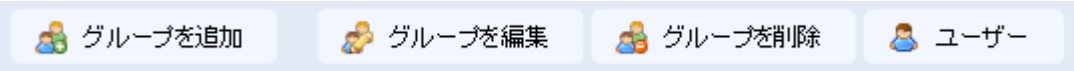
初期ユーザーの他に、**admin** には以下の特別権限があります。

- [RoboServer] tab で、**admin** は RoboServer ノードをクリックして対応する 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。

[ユーザー] タブでは、新しいユーザーの作成、選択したユーザーのパスワードの編集、除去、およびリセットを行うことができます。グループの管理については、[グループ] をクリックします。[ユーザー] タブに戻るには、[ユーザー] をクリックします。



[グループ] タブでは、グループの作成、除去、および編集を行うことができます。



以下の情報は、Kofax RPA のユーザー管理に関する原則の一部を理解するのに役立ちます。

Management Console の実行には以下の 2 つの方法があります。任意のライセンス付きで RoboServer に組み込まれたもの、およびスタンドアロン Tomcat サーバー (エンタープライズ ライセンスが必要) に組み込まれたもの。Tomcat の Management Console の詳細は、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Tomcat Management Console」(Tomcat 管理コンソール) を参照してください。

組み込み型モードで Management Console を実行している場合は、シングル ユーザーおよびマルチ ユーザーという、2 つのタイプのユーザー管理があります。これは、Kofax RPA インストール フォルダの `/bin` サブフォルダに位置する RoboServer 設定 (RoboServerSettings.exe) アプリケーションを使用して設定します。

マルチ ユーザー モードを有効化するには、以下を実行します。

1. Windows のスタート メニューの [RoboServer 設定] をクリックするか、Kofax RPA インストール フォルダの `/bin` サブフォルダにある `RoboServerSettings.exe` をダブルクリックして、RoboServer 設定アプリケーションを起動します。

2. Management Console タブで、[ユーザー管理の有効化] を選択します。
   また、管理者の名前とパスワードを指定することができます。
3. [一般] タブで、[**Management Console** に登録] を選択し、マルチ ユーザー モードを有効化する Management Console の URL、名前、パスワード、およびクラスタ名を指定します。
4. 変更を有効化するには、Management Console を再起動します。
   ライセンスや、Management Console の実行方法にもよりますが、以下のようにユーザー アクセスを管理できます。
   - シングル ユーザー - 組み込み型モードでのみ利用可能
   - 内部ユーザー管理 - 組み込み型およびスタンドアロン モードの両方で利用可能
   - 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 ライセンスによるスタンドアロン モードでのみ利用可能

   注 内部ユーザー管理を使用する場合、管理者グループは表示されず、RoboServer 設定で定義された管理者のみが表示されます。
   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、Management Console で (組み込み型モード) ユーザーを管理するか、コーポレート LDAP サーバーからユーザーの資格情報を取得するかを選択できます。

   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オフになっています。この機能を有効にするには、`<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。

```
   <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>
```

   最初の値を **true** に設定します。2 番目と 3 番目の value はそれぞれログイン試行回数 (この例では 3) および次の試行までの待ち時間 (この例では 10) です。

設定

このタブを使用して、Management Console を設定します。

注 構成設定を変更して保存したら、変更を適用するために Management Console を必ず再起動してください。

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 Administrator' s s Guide』(Kofax RPA 管理者ガイド) の「Embedded Management Console Configuration」(埋め込み Management Console の設定) のトピック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されます。

RoboServer 認証

このセクションでは、設定されたクラスタに属する RoboServer 上でロボットを実行する際に Management Console が使用する資格情報 (ユーザー名およびパスワード) を設定します。Management Console は、すべての RoboServer に同じセットの資格情報を使用します。これらの資格情報は、設定された RoboServer すべてに対して『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Request Authentication」(リクエスト認証) セクションで説明されているように、設定と一致する必要があります。

RoboServer パージ

このオプションが選択されていると、Management Console は **[管理]** > **[Roboserver]** タブの RoboServer のリストからオフラインの RoboServer を自動的に除去します。RoboServer がオフラインになると、Management Console で自動的に登録され、RoboServer のリストに表示されます。

RoboServer ログ データベース

これは、Management Console がスケジュールの実行、スケジュール メッセージ、ロボットの実行、ロボットのメッセージ、ロボットのタグ、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 ロギングが有効化されているクラスタに属する RoboServer すべてを保存するために使用されるロギング データベースです。このデータベースは、クラスタ データベースと同じように設定します。

RoboServer ログ データベースには、以下のクリーンアップのしきい値を設定できます。

プロパティ	説明
ロボットおよびスケジュール統計を (日) 維持	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件数	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。

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。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。

Management Console および RoboServer は両者とも、開始されたときに自動的にログ テーブルを作成します (テーブルがまだ存在しない場合)。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 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 (x86)\Kofax RPA 11.0.0 x32\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。

注 SQL スクリプトは、Design Studio をインストールした場合、Kofax RPA ドキュメンテーションとともにインストールされています。

データベース テーブルの **SQL** スクリプト

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。

- altosoftsession: Altosoft とのシングル サインオンのテーブルの作成と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ドロップ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 でのレポート作成に必要な統計テーブルを作成およびドロップするためのスクリプト。

重要 IBM DB2 データベースには、すべてのテーブルを作成するために最低 8192 KB のページ サイズを持つテーブル スペースが必要です。

Management Console は、Quartz というサードパーティのスケジューリング コンポーネントを使用します。Quartz にも、その他のプラットフォーム テーブルに存在する必要のある多数のテーブルが必要です。これらのテーブルは、Management Console が開始したときに自動的に作成されます。または、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

このタブは、Kofax RPA ダッシュボードが使用するデータベースへの接続を設定するときに役立ちます。データベースは、RoboServer ログ データベースの場合と同じように設定されます。

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。

Kofax Insight で HTTP リクエスト認証を使用する場合は、シングル サインオンデータベースを作成します。

注 Analytics 機能の可用性は、ライセンスによって異なります。Analytics 向けのライセンスがある場合、Management Console の **[管理]** > **[設定]** タブに [Analytics データベース] 設定パネルが表示されます。

SMTP サーバー

このセクションでは、Kapplet ユーザーへの通知の送信 (通知が、Kapplet を開始する前に選択されている場合)、および電子メール ロギングがクラスタ データベースで有効化されている場合の、電子メール ログ メッセージの送信に使用されるメールサーバーを設定できます。

注意 : Kapplet 通知電子メールの動作には、Kapplet 結果で送信者アドレスが指定されている必要があります。

SMTP サーバー、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	SMTP サーバーが認証を必要とする場合は、ここにユーザー名を入力します。
パスワード	SMTP サーバーが認証を必要とする場合は、ここに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。これは、SMTP サーバーに信頼できる証明書がある場合にのみ動作します (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 のトラストストアへのエクスポートおよびインポートが必要です)。STARTTLS SMTP 拡張を使用します。 • **SMTPS**: SMTP over SSL電子メールが送信される資格情報の通信に安全なチャンネルが確立されます。これは、**SMTP** サーバーでサポートされていることはあまりありません。ただし、サーバーが自己署名証明書を使用している場合でも動作します。

ベース URL

アプリケーション ベース URL を使用して、Classic Kapplet で結果のリンクを生成し、Management Console サーバーに存在する結果が電子メールに含まれるようにします。

注 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/ManagementConsoleHelp/`

[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 はインターネット接続がアクティブであって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われます。

変更を有効にするには、Management Console を更新する必要があります。

オフライン ドキュメントについての詳細は、『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) を参照してください。

DAS 設定

このセクションは、Desktop Automation サービスが Management Console に ping を送信する ping 間隔を設定するために使用されます。最初の ping で、Management Console は Desktop Automation サービスに対して管理コンソールへの ping の送信頻度を指定します。

プロキシ サーバー

このセクションでは、プロキシ サーバーを指定することができます。Management Console は、たとえば、外部 API にアクセスするために OAuth セキュリティ トークンを生成するときに、このプロキシ サーバーを通じて外部サーバーに接続します。

プロキシ 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
プロキシ サーバーを使用	プロキシ サーバーを使用する必要がある場合は true、直接接続を使用する場合は false。
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	プロキシ サーバーで認証が必要な場合、ここに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、ここに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

このタブは、ロボット ファイル システム サーバーの接続設定に役立ちます。このサーバーは 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担当します。ロボット ファイル システム サーバーは、以下のプロパティで設定します。

プロパティ	説明
ロボット ファイル システム サーバーの使用	ロボット ファイル システム サーバーを使用できるようにします。
URL	ロボット ファイル システム サーバーの URL を入力します。
ユーザー	ロボット ファイル システム サーバーにアクセスできるユーザー名を指定します。
パスワード	ロボット ファイル システム サーバーにアクセスするためのパスワードを指定します。

Kofax Insight ダッシュボード

Classic Kapplet で Kofax Insight ダッシュボードを有効にして、接続先のサーバー URL を指定することができます。

[ダッシュボードを許可] を選択して、[サーバー URL] で URL を指定した後、Classic Kapplet を再起動します。メイン ナビゲーション メニューに、[新規ダッシュボード] メニュー アイテムが含まれています。

Kofax Insight ダッシュボードに接続できる新規 Kapplet を作成するには、[新しいダッシュボード] をクリックして、必要なすべてのパラメータを指定します。

シングル サインオン パラメータを指定することもできます。

シングル サインオン

Kofax RPA ダッシュボードによる HTTP リクエスト認証に使用されるシングル サインオン (SSO) データベースへのデータベース接続を設定できます。SSO データベースは、Kofax Insight で使用するその他のデータベースと同じように事前に作成します。すべての必要な表は、Kofax RPA によって作成されます。詳細については、『Kofax Analytics for RPA Administrator's guide』(Kofax Analytics for RPA 管理者ガイド)を参照してください。

Kapplet に全パラメータを指定して、Kofax RPA ダッシュボードに使用されるデータベースに接続します。

- ホスト: Kofax RPA ダッシュボードによって使用されるデータベースを実行するサーバー名を指定します。
- スキーマ: スキーマ名を指定します。
- データベース タイプ：リストからデータベース タイプを選択します。
- ユーザー名：選択したデータベースに接続するユーザー名を入力します。
- パスワード：選択したデータベースに接続するパスワードを入力します。

シェア データベース

シェア データベースは、Management Console (からライセンスを取得して) によって有効化されている、すべての Design Studio クライアントに送信されます。

ユーザー インターフェイスを簡素化するためにも、これらのデータベースはクラスで設定されます。

- 特定のクラスタおよび Design Studio クライアントでデータベースを利用できるようにするには、[シェア データベース] ウィンドウでクラスタを選択し、このデータベースを指定している 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。**[リポジトリ]** > **[データベース]** タブでデータベース マッピングを確認してください。[すべてのクラスタ] を選択して、すべてのクラスタからのデータベースを Design Studio クライアントで利用できるようにします。
- Design Studio が利用できる特種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。

"Production" という名前のデフォルトのクラスタが、シェア データベースに自動的に作成されます。

Kapplet 結果

このセクションを使用して、Kapplet 結果設定を設定します。

プロパティ	説明
送信元アドレス	結果電子メールの送信に使用する電子メール アドレスを設定します。注意：結果電子メールを使用するには、SMTP サーバーを適切に設定する必要があります。

データベース タイプ

MySQL などのデータベース タイプを定義することができます。データベース タイプは、次のプロパティで構成されます。

プロパティ	説明
名前	データベースをタイプを識別する名前。
JDBC ドライバ	ドライバの JDBC ドライバ クラス (ドライバは、データベース ドライバ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が作成されたときに提供される値です (「クラス データベース」セクションを参照してください)。
データベース タイプ	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については、『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) の「System Requirements」(システム要件) を参照してください。

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。

新しいデータベース タイプを追加するには、データベース タイプ カテゴリを選択して、データベース タイプの追加をクリックします。または、データベース タイプ カテゴリを右クリックして、データベース タイプの追加を選択します。

注 Kofax RPA は、新しいデータベース タイプの追加をサポートしていません。ただし、非標準ポートで動作するデータベース サーバー、または異なる認証方法が必要なデータベース サーバーを定義するタイプを追加するために、データベース タイプを変更できます。

データベース ドライバ

データベース ドライバ セクションには、アップロードされたデータベース JDBC ドライバが含まれています。これらのドライバは、さまざまなデータベース タイプに必要です。データベース タイプと同じように、アップロードされたデータベース ドライバは Design Studio クライアントに送信されます。たとえば、MySQL データベースで Management Console を実行する場合、Tomcat に MySQL ドライバを提供する必要があります。つまり、MySQL データベースは Management Console での使用時 ([ログ] タブ上の操作、または接続のテスト時) に動作します。ただし、MySQL データベースをすべての RoboServer

で動作させるには、ここで MySQL ドライバ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。

注 エンタープライズ Management Console では、Derby JDBC ドライバは配布されません。Derby JDBC ドライバ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、特定のサイズよりも大きいファイルを格納するために、パラメータまたは設定の微調整が必要になる場合があります。たとえば、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 ドライバを格納することはできません)。ドライバ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。問題の特定を容易にするために、ユーザーにエラー メッセージが送信されます。また、Management Console ログには、追加情報が含まれています。

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 ドライバをアップロードできるのは管理者ユーザーだけです。埋め込みモードで Management Console を実行している場合、RoboServer 設定アプリケーションでこの動作を変更できます。詳細については、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Embedded Management Console Configuration」(埋め込み 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 Administrator's Guide』(Kofax RPA 管理者ガイド) の「Tomcat Management Console」(Tomcat 管理コンソール) にある「Security」(セキュリティ) セクションを参照してください。

Process Discovery Analyzer

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 パラメータを指定し、データベースのプロビジョニングを実行するためのクレデンシャルを設定します (Process Discovery Analyzer 分岐の [設定] ページ)。

プロパティ	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 ツールを使用する場合、これが、Process Discovery が Docker コンテナーで実行するコンピューターとなります。 • [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。 • [データベース 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。

プロパティ	説明
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。 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。
詳細設定	Analyzer 実行設定を指定します。 • [スケジュール モード]: Analyzer の実行方法を選択します。 • [特定時に実行]: Analyzer を実行する `hh:mm` 形式の時間値のカンマ区切りリストを設定します。[特定時に実行] スケジュール モードを選択すると、このプロパティが表示されます。時間は、以下のように指定できます: 1:00、15:30 • [実行間隔 (分)]: Analyzer の実行間隔を設定します。[定期的に実行] スケジュール モードを選択すると、このプロパティが表示されます。 • [プロセス インスタンスの非アクティブ状態の最大間隔 (秒)]: 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。ユーザー非アクティビティ期間が指定された値を超えると、プロセスを検出できなくなります。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。 • [シーケンシャル 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。 • [マウス スクロールの使用イベント]: 分析でマウスのスクロールを使用する場合に選択します。

Process Discovery グループ

Kofax RPAProcess Discovery は、ユーザーのアクティビティを監視し、ボトルネックを処理するために必要なデータを収集するのに役立ちます。詳細については、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Kofax RPA Process Discovery」を参照してください。

このウィンドウで指定されたグループは、コンピュータを `finacial_dept`、`hr_dept`、`sales_dept` などのインストールされている Process Discovery Agent と論理的に組み合わせます。Process Discovery Agent を展開する場合は、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、左ペインで [**Process Discovery** グループ] をクリックし、[**Process Discovery** グループ] ペインで [**Process Discovery** グループの追加] をクリックします。

プロパティ	説明

記録設定	このグループの記録設定を指定します。
	• 記録モード: エージェントの記録モードを設定
	  • 無視リストを除く、すべてのアプリケーションを記録: [アプリケーション無視リスト] フィールドのコンマ区切りのリストで指定されたアプリケーションを除く、すべてのアプリケーションのアクションを記録します。例：`Outlook`、`Word`、`Excel`。デフォルトでは、Skype と Lync はアプリケーションの無視リストに含まれています。
	  • 記録リストのアプリケーションのみを記録: [アプリケーション記録リスト] フィールドのコンマ区切りのリストで指定されたアプリケーションのアクションを記録します。例：`Onenote`、`Photoshop`、`Publisher`。
	  • 記録を無効化: アクションは記録されません。
	• アプリケーション ツリー モード: ユーザー アクションの記録中、エージェントはアプリケーション ツリーのアクティブな要素へのパスを記録します。アプリケーション ツリーは、このプロパティで選択されたモードを使用して取得されます。詳細については、ツリー モードを参照してください。
	  • **ISA** および **Windows**: これはアプリケーション ツリーを記録する場合にデフォルトの、推奨されるモードです。アプリケーションによっては、インテリジェント スクリーン オートメーション (ISA) または Windows (以下を参照) を使用して最適な結果を得ることができます。Process Discovery Analyzer は解析に最適です。
	  • **ISA**: インテリジェント スクリーン オートメーションは、Desktop Automation で使用される Kofax RPA テクノロジで、アプリケーションの UI 要素のツリーを画面のイメージを基にして構築することができます。画面認識は CPU に大きな負荷がかかる操作です。Process Discovery Agent は夜間など、CPU の負荷が低いときに記録されたスクリーンショットの認識を実行します。
	  • **Windows**: このプロパティは、Microsoft UI オートメーション フレームワークを使用して、アプリケーション ツリーを構築します。Web ブラウザ (UI オートメーション API では DOM 構造は利用できない)、Citrix、RDP、SAP、Java など、一部のアプリケーション タイプでは、UI オートメーション API は限定されて提供されるか、または提供されません。このようなアプリケーションの場合、エージェントは要素の情報へのパスを限定して記録するか、また記録しません。
	• [記録の自動開始]: 選択すると、このグループのエージェントは自動的に記録を開始します。
	• [スクリーンショットを記録]: アクティブなアプリケーションの画像をキャプチャーし、保存します。記録されたスクリーンショットは Kofax RPA ダッシュボードの [発見されたプロセス] ビューでユーザー アクションを視覚的に表示するために使用されます。
	• スクリーンショットをぼかす: [スクリーンショットを記録] オプションが選択されている場合、このオプションを選択することでスクリーンショットのテキストを認識できないようにできます。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。
	• [スクリーンショットの圧縮率 **(%)**]: 記録されたスクリーンショットの圧縮レベルをパーセントで明示的に設定します。値が大きいほど画質は良くなり、画像を保存するために必要な容量が増加します。たとえば、100% の画質を選択すると、60% の画質と比べて必要な容量が 3 倍になります。
	• [スクリーンショットのサイズ]: スクリーンショットの寸法の閾値をピクセル単位で設定します。スクリーンショットのサイズが大きい場合は、指定した値に合わせて、画像の高さと幅のサイズが比例しながら変更されます。

プロパティ	説明
[データベースの設定]	エージェント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。 • 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 ツールを使用する場合、これが、Process Discovery が Docker コンテナーで実行するコンピューターとなります。 • [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。 • [データベース 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]: このグループのエージェントによって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]: このグループのエージェント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。 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。

KTA 設定

このタブを使用して、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。

Kofax TotalAgility インストールを新規追加するには、以下の手順を実行します。

1. 左ペインの [**KTA** 設定] をクリックします。
2. [**KTA** 設定を追加] ボタンをクリックします。
3. 以下のプロパティを使用して資格情報を設定します。

プロパティ	説明
KTA インストール名	新規 Kofax TotalAgility インストール名を指定します。
URL	Kofax TotalAgility を実行しているコンピュータの URL を指定します。
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 Kofax 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。
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 Kofax 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。

4. [テスト] をクリックして接続をテストします。
   テスト接続が実行されると、Management Console は実行状態 (成功、または説明付きの失敗) を記載したメッセージを表示します。
5. 変更を有効化するには、[保存] をクリックします。

CyberArk の設定

CyberArk にアクセスするときに使用する Management Console のクレデンシャルを設定する場合は、このタブを使用します。

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 の 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 のポート	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	Internet Information Services (IIS) で指定された CyberArk の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入力します。
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。 • 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。

CyberArk のキー ストア

このタブは、CyberArk キー ストアをアップロードする場合に使用します。

新しい CyberArk キー ストアをアップロードするには、次の操作を実行します。

1. [**CyberArk** キー ストアのアップロード] ボタンをクリックします。
2. 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
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	証明書と秘密鍵のペアを含むキー ストアをアップロードして、CyberArk でアプリケーションを認証します。
ストアのパスワード	キー ストアのパスワードを入力します。 重要 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。

プロパティ	説明
キーのパスワード	秘密鍵のパスワードがある場合は、キー ストアに入力します。

3. [アップロード] をクリックして、CyberArk キー ストアにアップロードします。
4. [保存] をクリックして変更を有効にします。

Management Console のバックアップ

一部またはすべての Management Console 設定を、バックアップまたはエクスポートのためファイルにコピーするには、[バックアップ] タブを使用します。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。

注 復元できるのは、Kofax RPA バージョン 9.2 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。

バックアップには、リポジトリにおけるすべてのスケジュールおよびプロジェクトを含む、すべての Management Console 設定が含まれます。また、バックアップは全体の復元のみが行えます。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。8.1 リリース以前の Management Console のインスタンスからの復元に関する詳細は、バックアップの復元を参照してください。

注 8.1 リリース以前は、「バックアップの作成」は「エクスポート」と呼ばれていました。この「エクスポート」という用語は現在では、「プロジェクトのエクスポート」にのみ使用されています。

[プロジェクトのエクスポート] は、単一のプロジェクトに関連する情報を持つファイルの作成に使用します。これは、プロジェクト内のスケジュール、ロボット、タイプ、リソースから構成されます。このようなファイルは、テスト環境からプロダクションへと移行するといった場合に、スケジュールやロボットなどを特定の Kofax RPA システムから別のシステムにコピーするために使用することができます。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。
   [バックアップの作成] ウィンドウが表示されます。
2. [バックアップの作成] をクリックします。
3. バックアップが完了したら、[クリックしてダウンロード] を選択します。
   バックアップ ファイルを開いたり、[保存] をクリックすることができます。

プロジェクトのエクスポート

プロジェクトをエクスポートする際に、必要なプロジェクトをこのウィンドウで選択する必要があります。[作成] をクリックして、必要なバックアップ/エクスポート ファイルを作成します。この作業にはしばらく時間がかかることがあります。ファイルの準備が整うと、ダウンロード リンクが表示されます。このリンクは、ウィンドウが開いている際にのみアクティブとなります。リンクをコピーしたり、ウィンドウを閉じてから使用することはできません。

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。

   [エクスポートの作成] ウィンドウが表示されます。
2. エクスポートするプロジェクトを選択します。
3. [エクスポートの作成] をクリックします。
4. プロジェクトのエクスポートが完了したら、[クリックしてダウンロード] を選択します。

   エクスポートしたファイルを開いたり、[保存] をクリックすることができます。

プロジェクトの名前別のエクスポート

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

8.1 リリースより前の Management Console からエクスポートを復元する場合：上記のように、以前の Management Console で作成したバックアップから復元を行うことができます。以下の 3 つの注意事項を頭に入れておいてください。初めに、8.1 リリースでは、用語が変更されています。以前は「エクスポート」という用語は、現在の「バックアップ」を意味していました。つまり、8.1 リリース以前の Management Console からコンテンツを切り替える場合、その「エクスポート」機能を使用してファイルをエクスポートする必要があります。またこのファイルは、Design Studio の [ツール] メニューの「8.1 以前の Management Console エクスポートのアップグレード」を使用して新しいバックアップ ファイルのフォーマットに変換する必要があります。3 つ目の注意事項として、8.2 で新たに導入されたプロジェクト ベースのアクセス許可では、以前のバージョンの権限が自動的にアップグレードされないことが挙げられます。以前のバックアップを復元した後には、プロジェクトのアクセス許可を手動で割り当てる必要があります。

注 Kofax Kapow 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ます。そのため、50000 以上の記録数を持つ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。
   [バックアップの復元] ウィンドウが表示されます。
2. 復元するバックアップ ファイルを参照します。
3. ドロップダウン リストから [結合] または [リセット] のいずれかを選択します。
   - 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は、カスタム値を保持します。
   - 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。
4. [復元] をクリックします。
5. 復元が完了したら、[**OK**] をクリックして、[閉じる] をクリックします。

プロジェクトのインポート

1. バックアップ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。
   [プロジェクトのインポート] ウィンドウが開きます。
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
   - マスター Kapplet のインポート (Classic Kapplet)
   - 権限のインポート
3. インポートするバックアップ ファイルを参照します。
4. [インポート] をクリックします。
5. インポートが完了したら、[**OK**] をクリックして、[閉じる] をクリックします。

ライセンス

[ライセンス] タブを使用して、新しいライセンスを入力したり、現在のラインセスを表示したりします。Management Console には、以下の 2 つのライセンスキーを入力できます：プロダクション キーおよび非プロダクション キー。プロダクション環境が開発/QA 環境と完全に分かれている場合は、2 つの Management Console をインストールする必要があります。そのうちのひとつには、プロダクション ライセンス キーが含まれ、もうひとつには非プロダクション ライセンス キーが含まれます。混合環境では、単一の Management Console に両方のキーが含まれます。

また、このタブには、現在使用中の Design Studio の情報も表示されます (この Management Console からライセンスを受信し、現在アクティブな状態の Design Studio クライアント)。

注 リストの一部の項目は、適切なライセンスを所有している場合にのみ表示されます。たとえば、Analytics のライセンスを所有している場合は、**Kofax RPA Analytics:** [はい] の項目が [現在のライセンスについて] の表に表示されます。

データベース タイプの追加

変更したデータベース タイプを Management Console で追加するには、以下の手順を実行します。例として、Windows 統合セキュリティを必要とする Microsoft SQL Server を追加します。

1. Management Consoleで、**[管理]** > **[設定]** の順に選択し、[データベース タイプ] をクリックします。
2. [データベース タイプの追加] をクリックします。
3. 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：com.microsoft.sqlserver.jdbc.SQLServerDriver
   - 接続 URL テンプレート：jdbc:sqlserver://${ServerName};databaseName=${Schema};integratedSecurity=true
   - [データベース タイプ] リストで [**Microsoft SQL Server**] を選択します。

   [保存] をクリックして、データベース タイプをリストに追加します。
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、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。**[管理]** > **[設定]** タブで [データベース ドライバ] を選択し、[ドライバ **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]** > **[クラスタ]** タブで、新規作成されたデータベース タイプを使用するデータベースに接続できます。

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、64 ビットではなく、auth .dll の 32 ビット バージョンを使用します。また、Management Console を再起動しない場合、「データベースへの接続エラー：このドライバ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
- シェア データベース
- データベースの操作

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 API だとみなすことはできません。

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構成に関する情報。
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 Management Console インスタンス間に配布されたタスク 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。これにより、アプリケーション (Kofax RPA の場合はロボットも) は、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、ユーザーに代わってデータにアクセスすることや、アクションを実行することが可能になります。たとえば、ロボットは Twitter の API を使用することで、@Kofax といった、ユーザーの最近のメンションを抽出することができます。これは、@Kofax の Twitter パスワードに明示的なアクセスを行うことなく、ユーザーによって提供されるアクセス トークンを使用して行われます。

Management Console は、アクセス トークンを生成するために使用されます。これは、キーストアに安全に保存されます。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 プロバイダー

OAuth 用語では、サービス プロバイダーとは、Web サービスのプロバイダーを意味します。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、関連の Web サイトをご覧ください。

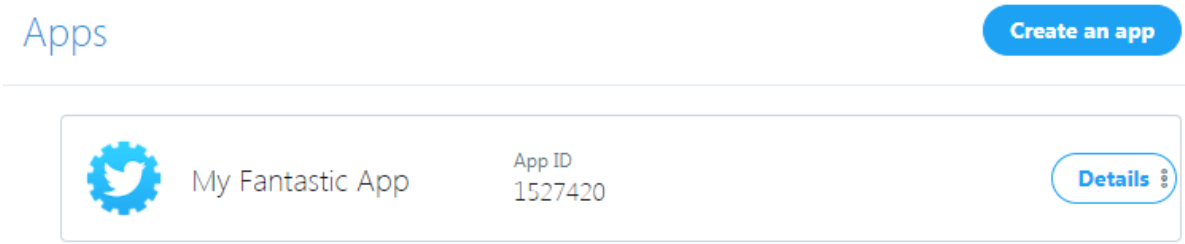
- Dropbox
  https://www.dropbox.com/developers
- Facebook
  https://developers.facebook.com
- Google
  https://console.developers.google.com/apis/library
- LinkedIn
  https://developer.linkedin.com
- Microsoft Azure AD 2.0
  https://docs.microsoft.com
- Salesforce
  https://developer.salesforce.com
- 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、コンシューマー キー (API キーまたはアプリケーション キーとしても知られる) およびコンシューマー シークレット (API シークレットまたはアプリケーション シークレットとしても知られる) が生成されます。

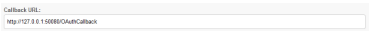
アプリケーションを作成するには、サービス プロバイダーの開発者コミュニティにログインし、[新しいアプリケーションの作成] またはそれと同等のオプションを選択して必要な情報を入力します。開発者のサービス プロバイダーのドキュメンテーションへのリンクは、サポートされているサービス プロバイダーを参照してください。このトピックでは、Twitter で行う方法を説明します。

1. まず、https://developer.twitter.com/en/apps/ にログインします (必要に応じてアカウントを作成します)。そして、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といった必要な情報を入力し、承認する前に Twitter の利用規約を読みます。

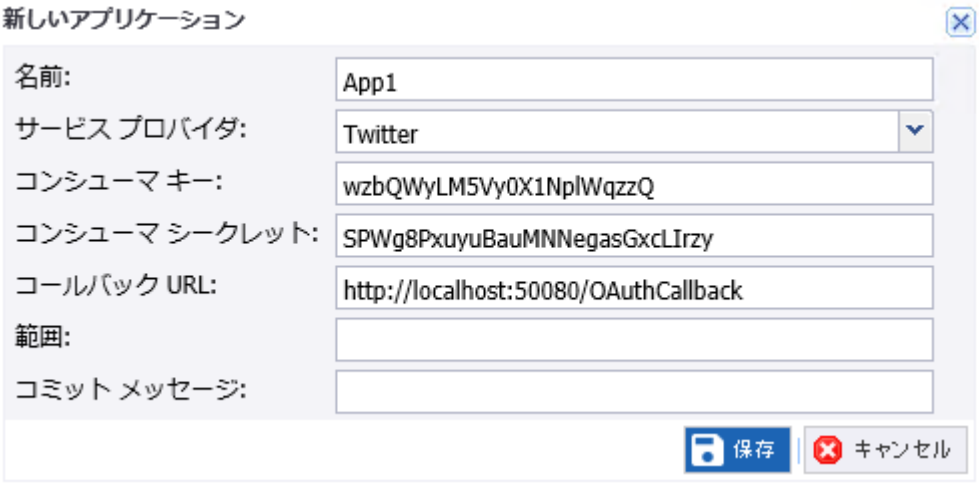
   フォームのフィールドの 1 つに、[コールバック **URL**] があります。これは、ユーザーが、ユーザーの代わりに Twitter アカウントとアプリケーションを対話させることを承認した後に Twitter がブラウザをリダイレクトする URL です。このフィールドは、Management Console が展開されているフォルダ下のパス OAuthCallback に設定される必要があります。たとえば、組み込み型の Management Console で実行している場合、これは `http://localhost:50080/` で実行します。この場合、コールバック URL は、`http://localhost:50080/OAuthCallback` に指定されます。ただし、一部のサービス プロバイダーは、`localhost` を含む コールバック URL を許可し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ーの 1 つです。そのため、代わりに `http://127.0.0.1:50080/OAuthCallback` を使用します。

   または (これは、一部のサービス プロバイダーによって必要とされます)、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。このページは、認証している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 アドレスである必要はありません。

   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 として入力されたものと同じ 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、[新しいアプリケーション] をクリックします。
5. アプリケーションの名前 (サービス プロバイダーでアプリケーションを作成したときに使用したものと同じ名前である必要はありません) を選択し、サービス プロバイダー (この場合は Twitter) を選択します。

   コンシューマー キーおよびコンシューマー シークレットは、サービス プロバイダーによって示されたアプリケーションの概要ページからコピーされる必要があります。
6. ステップ 3 で入力したものと同じ コールバック URL を入力して [保存] をクリックします。

   さらに一部のサービス プロバイダーでは、ユーザーがアプリケーションにアクセスを承認する 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、そのスコー

プ `https://www.google.com/analytics/feeds/` が指定される必要があります。Twitter はスコープ フィールドを使用しないため、この例では空欄にしています。

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。

注 アプリケーションを後で編集すると、セキュリティ上の理由からコンシューマー シークレットは「 (encrypted) 」として表示されます。

ユーザーの追加

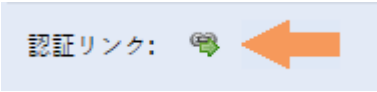
1. ユーザーの追加の列において、アプリケーションの追加で作成したアプリケーションの隣にある [ユーザーの追加] をクリックします。

ユーザーの追加ウィザードが表示されます。

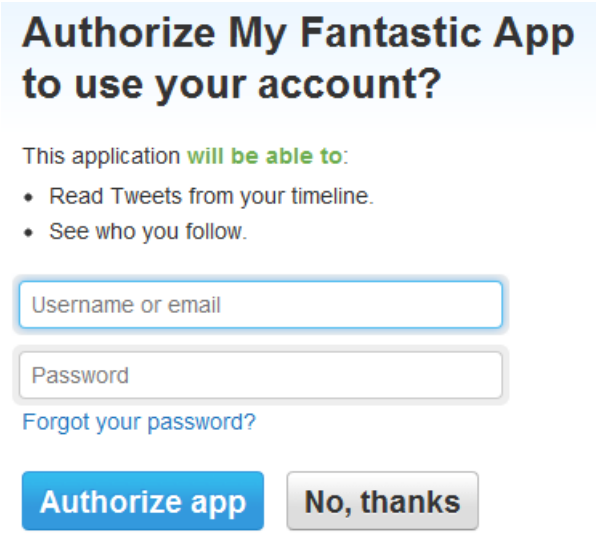
2. 「ユーザー名」フィールドで、ユーザーの名前を入力します。

このユーザー名は、サービス プロバイダーによって使用されるユーザー名にマッピングする必要はありません。これは、Management Console 内でのみ使用されます。

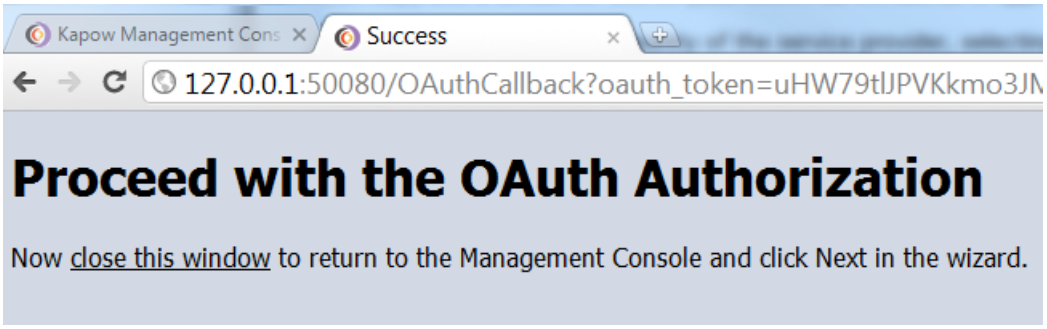
3. [次へ] をクリックします。
4. 承認リンクをクリックします。



これにより、サービス プロバイダーの 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、[アプリの認証] 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。
7. ウィザードで、[次へ] をクリックします。
   ユーザーの代わりにサービス プロバイダーにアクセスするために使用できるアクセス トークンが表示されます。これらは、Management Console のキー ストアに保存されています。また、スケジュールに対する入力として使用できます。

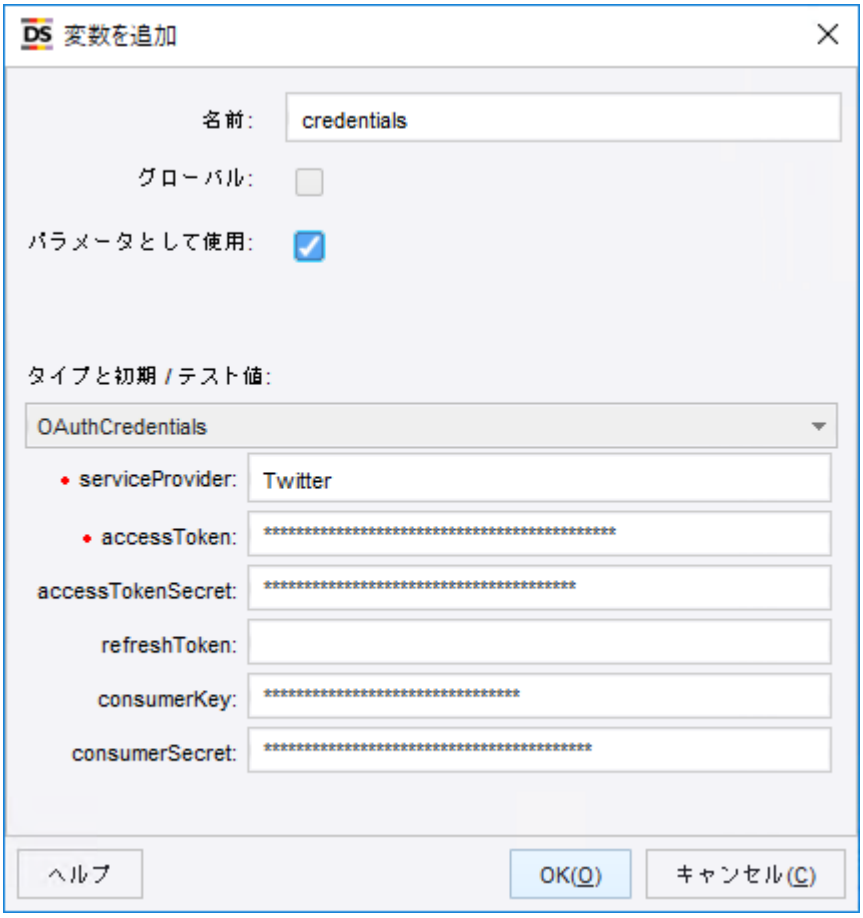
   注 後のステップで構築するロボットのテスト入力値として、サンプル アクセス トークンが必要になります。Notepad などのテキスト エディターに値をコピーします。セキュリティ上の理由から、[終了] をクリックした後で非暗号化フォームのキー ストアから値を取得することはできません。

   Twitter では、アクセス トークンおよびアクセス トークン シークレットの両方を取得します。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。すると、ロボットは更新トークンを使用することで、Management Console からユーザーの再認証なしで、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。
8. [終了] をクリックします。
   [OAuth] タブに、[ユーザー] セクションが表示されます。

ロボットの書き込み

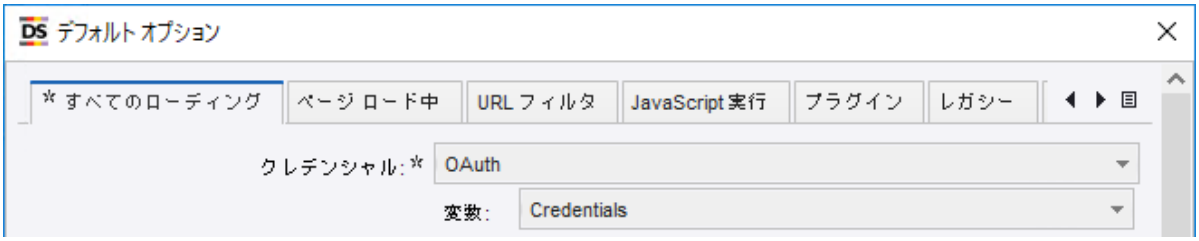
以下の手順では、認証メカニズムとして OAuth を使用する REST API にアクセスするロボットを書き込みます。たとえば、Twitter REST API を使用して、ユーザーおよびユーザーがフォローしているユーザーを認証することで最近のステータスを取得します。

1. Design Studio を開始し、ロボットを新規作成します。
   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. タイプ OAuthCredentials の新しい入力変数を追加します。
3. [**serviceProvider**] フィールドに、[**Twitter**] を入力します。
4. アクセス トークンおよび Management Console ウィザードにおけるユーザー承認プロセスを一通り終えてから、取得されたアクセス 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ー キーおよびコンシューマー 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、上記のような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。

これでロボットは、OAuth を使用するように設定され、Design Studio で実行する際に指定された資格情報を使用します。これで、Twitter の API へのアクセスを開始できます。たとえば、ユーザーが投稿した最新のツイートのコレクションを表示するには、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

1. 書き込みロボットで作成したロボットを保存して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. スケジュールにロボットを追加するには、ウィザードで [ロボットの追加] をクリックします。
4. [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。

5. 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 で実行される際にサービス プロバイダー、アクセス トークン、コンシューマー キー、コンシューマー フィールドが自動入力されます。
6. [終了] をクリックして、スケジュールを保存します。

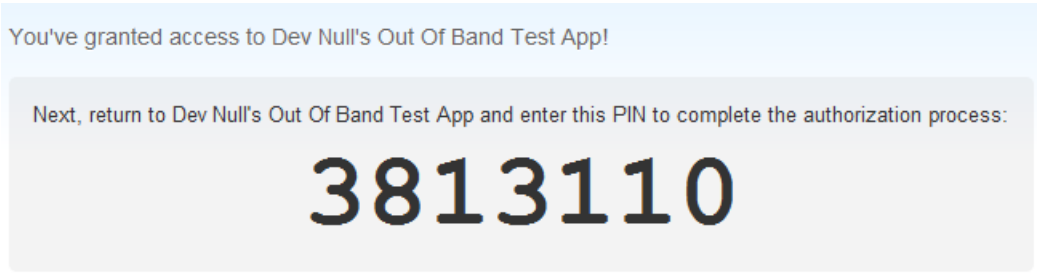
   これでロボットを実行する準備が整いました。

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 プロバイダーで作成したアプリケーションは、アウトオブバンド アプリケーションとして登録される必要があります。Twitter では、これはコールバック URL フィールドを空にすることで簡単に行えます。その他のサービス プロバイダは、アウトオブバンド メカニズムを使用しません。

1. Twitter アプリケーションを作成するときは、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 フィールドを空欄にします。

   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ません。代わりに、サービス プロバイダーは、検証機能または PIN を表示します。



3. [検証機能] フィールドで、サービス プロバイダーが提供する PIN を入力します。
4. [次へ] をクリックします。

第 6 章

Kofax RPAProcess Discovery

Process 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、Kofax RPA の使用に関する意思決定を行うために役立ちます。

Kofax Analytics for RPA ダッシュボードの Process Discovery ビューは、Process Discovery Agentが収集したデータに基づいたレポートを提供します。ビューには、グラフとテーブルを使用したさまざまな視覚的および分析的なデータの表現が含まれています。

詳細については、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) の「Kofax RPA Process Discovery」および「Kofax Analytics for RPA」を参照してください。

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。

注 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、エージェントの実行を阻止することがあります。エージェント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。
- バージョン情報: 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**]: 以下に接続する Management Console の URL とポート:
  `http://localhost:50080`
- [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。

> 注 ユーザー管理が Management Console で有効になっておらず、ログイン クレデンシャルが要求されない場合は、[ユーザー] フィールドと [パスワード] フィールドは空白のままでもかまいません。

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 は実行していますが、Management Console またはデータベースへの接続が確立できていません。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。 誤った設定: 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エージェント用です。スキーマを再作成するか、Management Console の設定を変更します。 エージェント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。

注 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 [バージョン情報] を選択するか、設定ウィンドウの [バージョン情報] 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

エージェントのショートカット 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 Analytics for Kofax RPA ダッシュボードに表示されます。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。

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、エージェントのショートカット 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、設定ウィンドウで [ヘルプ] をクリックするか、[**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。

Process Discovery Analyzer の構成

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。

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。

> 注 Analyzer と Management Console は同じバージョンを使用する必要があります。バージョンが一致しない場合は、Analyzer がエラーを表示することがあります。

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 グループのエージェント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。

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。

> 注 Analyzer が Management Console または Analyzer データベースに接続できない場合、解析は開始されません。

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。

> 注 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。

ログ ファイル
Analyzer ログ ファイル (analyzer.log) の場所は、オペレーティング システムや実行モードによって異なります。

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

Linux で **Docker** を使用した **Process Discovery Analyzer** の展開
Analyzer が Docker を使用して展開される場合、Docker イメージの作成前に docker-compose.yml ファイルの environment セクションでオプションが指定されます。イメージ ファイルの作成後に設定を変更する場合、docker-compose.yml を開いてオプションを編集し、ターミナルで次のコマンドを実行してイメージを再度作成します。

```
docker-compose -p process_discovery -f compose-examples/docker-compose.yml up --build
```

問題を回避するためには、Analyzer イメージの作成前に次のコマンドを実行してキャッシュをクリアします。

```
docker-compose -p process_discovery -f compose-examples/docker-compose.yml rm -v
```

詳細については、『Administrators Guide』(管理者ガイド) の「Deploy Analyzer using Docker on Linux」(Linux 上での Docker を使用した Analyzer の展開) を参照してください。

docker-compose.yml には、次のセクションが含まれています:

サービス セクション

- MC_URL: Management Console の URL
- MC_USER: Management Console へのログイン名
- MC_PASSWORD: Management Console へのログイン 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。

mysql-server-service セクション
MYSQL_ROOT_PASSWORD: 出たベースにアクセスできるルート ユーザーのパスワードを指定します。

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`	Management Console の URL。
`--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。

注 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ることに注意してください。

Analyzer のデフォルトの設定

```
--mc-url=localhost:50080
--log=info
--locale=<system settings>
```

第 7 章

Kofax RPA Kapplet

Kofax RPA Kapplet は、使いやすいユーザー インターフェイスをロボットに提供して、ユーザーが操作しやすいようにサポートします。ロボットやその構築方法に関する知識がなくても、付属の Kapplet を使用してロボットを操作することができます。

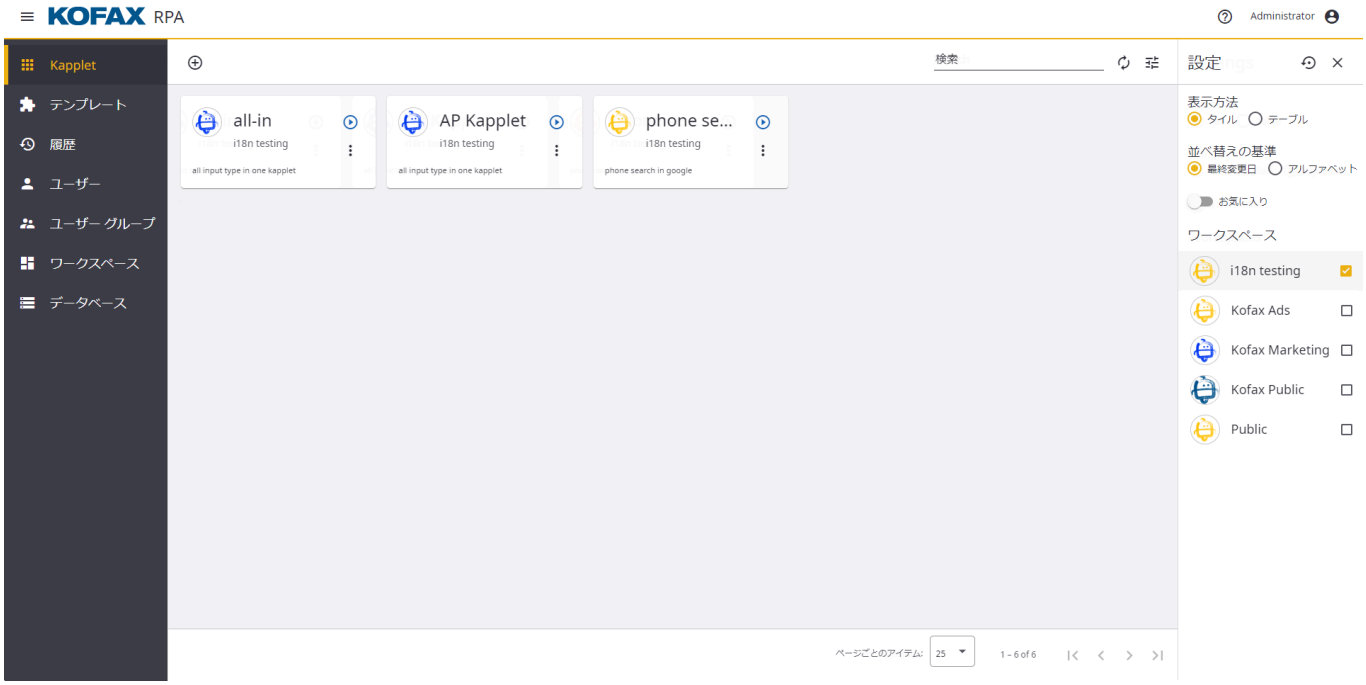
Kapplet のユーザー インターフェイス

このトピックでは、Kofax RPA Kapplet のユーザー インターフェイスの概要とその要素について説明します。

Kofax RPA Kapplet 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 Installation Guide』(Kofax RPA インストールガイド) を参照してください。

メイン ユーザー インターフェイスの要素

Kofax RPA Kapplet 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、コンピュータからだけでなく、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 Kapplet を操作することもできます。



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。

- ツールバー
- サイド メニュー
- ユーザー メニュー

ツールバー

ツールバーは、各タブ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、選択したタブに応じて異なる場合があります。

選択したタブに新しいアイテムを作成するには、⊕ [追加] 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。

選択したタブのアイテムを検索するには、検索ボックスを使用します。

Kofax RPA Kapplet のほとんどのタブでは、特定のタイプのアイテムが表形式で表示されます。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] ペイン 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、選択したタブに応じて異なる場合があります。たとえば、[**Kapplet**] および [テンプレート] タブには、Kapplet およびテンプレートを基準としてフィルタできるワークスペースのリストも表示されます。

サイド メニュー

サイド メニューは、Kofax RPA Kapplet のメイン ウィンドウの左側に配置されています。

サイド 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 ≡ サイド メニュー ボタンをクリックします。

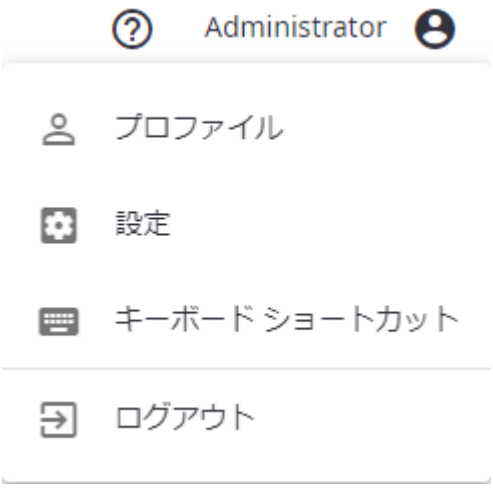
サイド メニューのタブを使用して、Kofax RPA Kapplet 間を移動します。

タブの名前	説明
Kapplet	このタブは、すべてのユーザー グループが使用できます。 Kapplet を作成、編集、および実行するには、このタブを使用します。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	このタブは、開発者、管理者、およびグローバル管理者のみが使用できます。 テンプレートを作成および編集するには、このタブを使用します。Kapplet は、テンプレートに完全に基づいています。
履歴	このタブは、すべてのユーザー グループが使用できます。 このタブは、Kapplet の実行情報を表示する場合に使用します。テーブル内の Kapplet の名前をクリックすると、[結果履歴] タブを開いて実行情報を表示することができます。 [設定] ペインの実行状態を基準として実行情報をフィルタします。
ユーザー	このタブは、グローバル管理者のみが使用できます。 このタブは、次のように Kofax RPA Kapplet 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	このタブは、グローバル管理者のみが使用できます。 このタブは、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

タブの名前	説明
ワークスペース	このタブは、管理者およびグローバル管理者のみが使用できます。 • ワークスペースを作成、管理、および除去する • 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
データベース	このタブは、グローバル管理者のみが使用できます。 このタブは、Kofax RPA Management Console バックアップ ファイルをインポートする場合に使用します。[ファイル選択] をクリックしてファイルを選択し、アップロードします。

ユーザー メニュー

ユーザー メニュー ボタンは、メイン ウィンドウの右上隅に配置されています。このボタンをクリックすると、ドロップダウン メニューが開きます。



[プロファイル] をクリックすると、ユーザーに関する一般情報が表示されます。

[設定] をクリックすると、[設定] タブが開きます。このタブは、ユーザー インターフェイスの言語を変更する場合に使用します (ユーザー インターフェイスの言語はデフォルトで英語に設定されています)。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。

設定をデフォルトの設定にリセットするには、[設定] タブの [設定をリセット] をクリックします。

[キーボード ショートカット] をクリックすると、Kofax RPA Kapplet で使用できるキーボード ショートカットが表示されます。

ユーザー ロール

Kapplet を使用する準備ができたら、Kapplet ユーザーまたはユーザー グループは専用ワークスペースを使用して Kapplet へのアクセス権を取得できます。ユーザー グループが使用できるのは、管理者が定義した一連の Kapplet のみです。Kapplet ユーザーのワークスペースには、インストールされた Kapplet のリストが保持されます。

Kapplet はユーザーの評価権限に応じて、ユーザーが構築および管理します。次に示す権限は、グローバル管理者が変更できます。

- ユーザー: Kapplet を実行します。
- 開発者: Kapplet およびテンプレートを作成して、Kapplet を実行します。
- 管理者: Kapplet、テンプレート、およびワークスペースを作成し、Kapplet を実行して、権限を割り当てます。

相関関係は次のようになります。

Management Console グループのロール	**Kofax RPA Kapplet** のグループ権限
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	グローバル管理者

注 Management Console の他のすべてのロールは Kofax RPA Kapplet 権限と相関関係がありません。

これらの権限には、ローカル権限とグローバル権限の 2 種類があります。

- グローバル権限はシステム全体で管理され、ユーザー グループまたは特定のユーザーに割り当てることができます。ユーザー グループ内のユーザーは権限を継承します。
- ローカル権限は、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。

Management Console バックアップからの Kofax RPA Kapplet の復元

新しい Kapplet を作成する以外にも、[結合] オプションを指定して Management Console のバックアップによる Classic Kapplet の復元を実行することができます。

ユーザー グループ内のユーザーも Kofax RPA Kapplet 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 バックアップ内のオブジェクトは、次の方法で Kofax RPA Kapplet に移動されます。

Management Console バックアップ オブジェクト	**Kofax RPA Kapplet** 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	Kofax RPA Kapplet 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

Kapplet の作成とメンテナンス

このセクションでは、ワークスペースの作成、テンプレートの作成、Kapplet の作成、および Kapplet の実行に関する情報を示します。

ワークスペースの作成

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。

ワークスペースとは、Kapplet とテンプレートを、意味や各部署のニーズに基づいて 1 か所に統合した論理的なグループのことです。

ワークスペースを作成するには、サイド メニューの [ワークスペース] タブに移動して、次の手順を実行します。

1. ツールバーの ⊕ [新しいワークスペースの作成] ボタンをクリックします。
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。

     重要 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. 変更を有効にするには、ウィンドウの右下隅にある [保存] をクリックします。

ユーザーに表示されたタブの [設定] ペインには、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、ローカル権限を持つユーザーには、[**Kapplet**] タブの [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] タブと [**Kapplet**] タブのワークスペースのみです。

テンプレートの作成

すべての Kapplet は、Kapplet の主要な基本オブジェクトとして機能するテンプレートに基づいています。Kapplet はこのテンプレートの「コピー」と見なされます。テンプレートを変更しても、Kapplet は変更されません。

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。

テンプレートを作成するには、サイド メニューの [テンプレート] タブに移動して、次の手順を実行します。

1. ツールバーの ⊕ [追加] ボタンをクリックして、[一般] ステップ ウィンドウを開きます。
2. [一般] ステップ ウィンドウ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]: 使用可能なワークスペースのドロップダウン 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。
   - [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。[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。

     重要 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. ウィンドウの右下隅にある [次へ] をクリックして、次のステップに進みます。
4. [ロボット] ステップ ウィンドウで、テンプレートにロボットを追加します。

   選択したロボットが、ウィンドウの左側にある [選択したロボット] ペインに表示されます。このペインを開くには、ツールバーの [選択したロボット] ペインをクリックします。選択したロボットの番号がボタンに反映されます。[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。

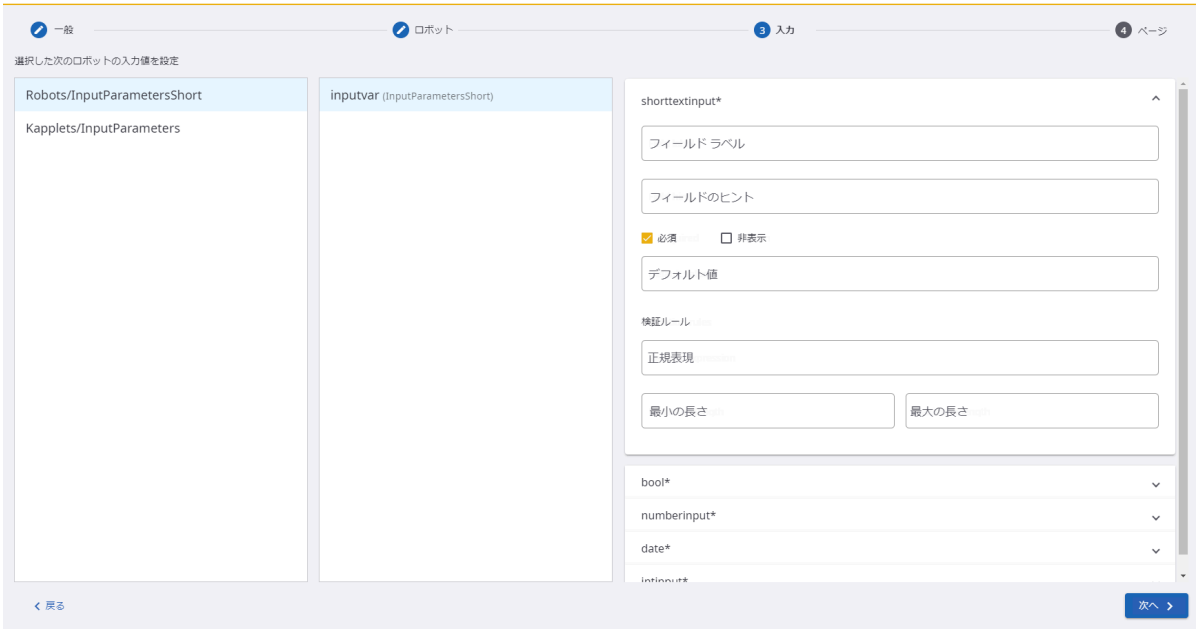
   [設定] ペインで、Management Console プロジェクトを基準としてロボットをフィルタすることもできます。

   [次へ] をクリックして、次のステップに進みます。

5. [入力] ステップ ウィンドウで、選択したロボットの入力パラメータを設定します。

   注 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「入力」ステップが追加されます。

   - ロボットをクリックして、変数 (変数のタイプ) 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。



   属性ペインで、[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。

   [必須] および [非表示] チェック ボックスを使用して、[開始ページ] にフィールドを反映させる方法を設定します。

   属性のパラメータを個別に設定します。パラメータの組み合わせは各属性に依存するため、図とは異なる場合があります。

   [次へ] をクリックして、次のステップに進みます。

6. [ページ] ステップ ウィンドウには、次の設定が含まれています。
   - [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、ツールバーの対応するアイコンをクリックして、[テーブル ページ] に追加します。[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。ロボットの下にすべてのパラメータがリストされます。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。
   - 実行結果をグラフに表示するには、ツールバーの対応するアイコンをクリックして、[グラフ ページ] に追加します。[グラフ 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。

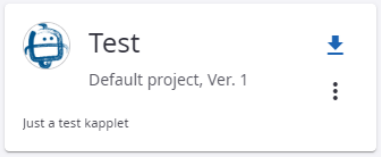
- グラフのタイプ
- 配色
- ラベル軸
- データ軸

他のグラフ パラメータも定義します。

[プレビュー] ボタンをクリックして、グラフの画像をプレビューします。

7. [保存] をクリックしてテンプレートを保存します。

テンプレートのエンティティは次のようになります。



テンプレートを編集または除去するには、⋮ [コンテキスト メニューを開く] ボタンをクリックします。

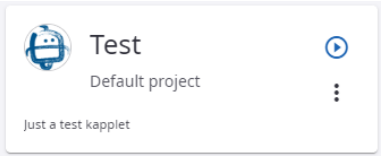
Kapplet の作成

Kapplet 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。

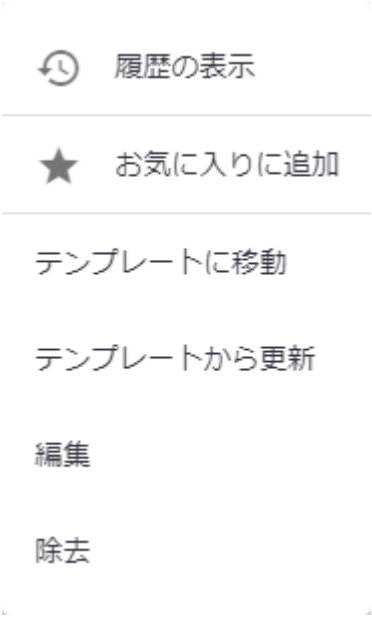
テンプレートから Kapplet を作成するには、次の手順を実行します。

1. 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。
2. [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。値を編集するか、Kapplet を保存して先へ進みます。Kapplet を保存するには、ウィンドウの右下隅にある [保存] をクリックします。

Kapplet のエンティティは次のようになります。



Kapplet にその他のアクションを実行するには、⋮ [コンテキスト メニューを開く] ボタンをクリックします。



また、1 つ以上のテンプレートがすでにある場合は、[**Kapplet**] タブで Kapplet を作成できます。

[**Kapplet**] タブで新しい Kapplet を作成するには、次の手順を実行します。

1. ツールバーの ⊕ [追加] ボタンをクリックして、[新しい **Kapplet** の作成] ウィンドウを開きます。
2. [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]: 使用可能なテンプレートのドロップダウン リストからテンプレートを選択します。
   - [ワークスペース]: 選択したテンプレートのワークスペース。
   - [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。[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。

     重要 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. 変更を有効にするには、ウィンドウの右下隅にある [保存] をクリックします。

重要 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。次の通知がスローされます。

Kapplet を有効にするには、ロボットが変更または除去されたテンプレートを更新してから、Kapplet を更新します。

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。
- Kapplet が実行されると、[履歴] タブが開きます。[結果履歴] タブの実行情報を確認します。ここで、[ステータス ページ]、[テーブル ページ]、および [グラフ ページ] の情報も確認できます。

Kofax RPA Kapplet のインストール

Kofax RPA Kapplet をインストールするには、次の手順を実行します。詳細については、『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) を参照してください。

1. Web サーバーに Kofax RPA Kapplet を展開します。
2. グローバル管理者が Kofax RPA Kapplet にアクセスする場合は、通常、Web ブラウザから直接行います。Kapplet には、`http://[IP アドレス]:[ポート]/kapplets` から直接アクセスできます。Kofax RPA Kapplet での認証には、Management Console クレデンシャルを使用します。

Kofax RPA Kapplet の制限事項については、Kofax RPA の制限セクションを参照してください。

第 8 章

リファレンス

コンテンツ

- Design Studio
- RoboServer
- Management Console
- プロキシ サービスの使用
- Kofax RPA の制限

Design Studio

Design Studio はロボットを作成し、デバッグするための Kofax RPA アプリケーションです。

Design Studio はロボット用の統合開発環境 (IDE) です。Design Studio では、理解しやすいビジュアル プログラミング言語でロボットを作成することができます。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 の詳細情報については、Design Studio を参照してください。

ステップ アクション

このトピックでは、利用可能なステップ アクション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 アクションが含まれます。

アクション	説明
変数の割当	値を変数に割り当てます。
Desktop Automation ワークフローの呼び出し	ネットワーク コンピュータ上で Windows アプリケーションおよび Java アプリケーションを自動化する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。「Desktop Automation ワークフローの呼び出し」を参照してください。

アクション	説明
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。
ページ読込	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 アクションが含まれます。

アクション	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 アクションが含まれます。

アクション	説明
セッションの保存	後で別のロ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、別のロ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	名前、ドメイン、パスのパターンと一致する Cookie の値を抽出します。
Cookie 作成	指定されたドメイン、パス、名前および (オプションで) 値を持つ Cookie を作成します。
Cookie 除去	名前、ドメイン、パス、値のパターンと一致する 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 ストレージ読込	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込ます。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 アクションが含まれます。

アクション	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。

アクション	説明
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 アクションが含まれます。

アクション	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 アクションが含まれます。

アクション	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 アクションが含まれます。

アクション	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。
HBase テーブル データ登録	値を HBase テーブルに保存します。

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 アクションが含まれます。

アクション	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。

アクション	説明
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：このアクションはリスト 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 アクションが含まれます。データは、Web サイトあるいは PDF、CSV、Excel、Flash 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。

アクション	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。
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。
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	Flash オブジェクトからコンテンツを抽出します。
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。
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。

アクション	説明
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 アクションが含まれます。ファイルとディレクトリの読込、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。

アクション	説明
ファイル読込	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込ます。
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、このアクションはエラーを生成します。
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、このアクション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 アクションが含まれます。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 アクションの方が単純です。このステップ アクションは検出されたタグの直下の子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 アクションと [次のステップ] ステップ アクションを使用する必要があります。

注 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 [タグ繰り返し] 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。

アクション	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。

アクション	説明
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 つでなければなりません。
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] アクション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] アクション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。
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。

ページ読込

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 アクションまたは既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 アクションが含まれます。必要に応じて、基本的な HTTP レベルでページ読込リクエストを指定することもできます。

アクション	説明
ページ読込	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 アクションが含まれます。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。

アクション	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することです。

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 アクションが含まれます。

アクション	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。
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 アクションが含まれます。

アクション	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点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。

アクション	説明
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れば現在の分岐以降の実行を停止するなど、判定用の条件付きアクションが含まれます。この条件は、検出されたタグのコンテンツ、変数または指定されたウィンドウの有無に左右されることがあります。

アクション	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。
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アクションが含まれます。

アクション	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。

アクション	説明
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。

JSON
このカテゴリには JSON 値を管理するためのステップ アクションが含まれます。

アクション	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 アクションが含まれます。

アクション	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。
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 アクションが含まれます。

アクション	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。

アクション	説明
ブラウザ再開	指定の待機条件が満たされるか、ブラウザがアイドルになった場合に (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 がこの操作を行うのに十分な特権を持ってい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。
Desktop Automation	ネットワーク コンピュータ上で Windows アプリケーションおよび Java アプリケーションを自動化するステップを作成します。
パスワード取得	パスワード ストアからユーザー 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。

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 アクション 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。

分岐をまたがって存続するグローバル変数を除き、各分岐は同じ状態 (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。そのような変更とは、ファイルへの書き込み、データベースへの保存、ウェブサイトでのフォーム送信など、実世界に影響を与えるものすべてです (例えば Amazon での本の購入)。

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。

動画
分岐ポイントの理解に役立つ分岐、ロボット状態、実行パスに関する短い動画を見ることができます。

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なければなりません。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。値のキーを知ることは、値を別の値に (例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、標準タイプの種類を持っている必要があります。

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。

Desktop Automation ワークフローの呼び出し

このアクションを行うと、Web オートメーション ロボットから Desktop Automation ロボットを呼び出すために必要な「Desktop Automation ワークフローの呼び出し」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。

Desktop Automation 機能を使用する前に、オートメーション デバイスを設定して、オートメーション デバイスにリファレンスを指定する必要があります (ターミナルを自動化する際には不要)。

プロパティ

以下のプロパティを使用して「Desktop Automation ワークフローの呼び出し」ステップを設定します。

入力値
Desktop Automation ワークフローに入力値を指定します。

出力マッピング
「Desktop Automation ワークフローの呼び出し」ステップからの出力値を保持する変数を割り当てます。

必要なデバイス
「Desktop Automation ワークフローの呼び出し」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。静的リファレンスまたは動的リファレンスを選択できます。静的リファレンスを選択した場合、使用するオートメーション デバイス マッピングを指定します。動的リ

ファレンスを選択する場合は、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。詳細については、オートメーション デバイスの参照を参照してください。動的リファレンス接続がロボットによって使用されていて、デバイスが接続されている場合、接続は維持され、ロボットの次の「Desktop Automation ワークフローの呼び出し」ステップで使用できるようになります。

REST Web サービス呼出

REST Web サービス呼出アクションは REST Web サービスにリクエストを送信し、例えば XML、JSON または HTML などの Web サービスの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 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。

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 4 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 リクエストでは、複数のパラメータを name/value 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
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したい場合は、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。

重要

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。

削除

データを削除するために使用されます。DELETE 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。

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、[ページ読込] 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。

ブラウザへの読込

結果がページ読込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。

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 ダイアログでアステ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。その他すべてのオプションは口ボットに対して指定されているものと同じになります。

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 Request プロパティにペーストします。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。

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じになります。

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。それを行う方法については、「プロキシ サーバーの指定」を参照してください。

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
これをチェックすると、ステップは (ラウンドロビン方式で) 次のプロキシを「プロキシ サーバーの指定」で説明しているプロパティ ファイルから選択します。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。個々のホストが別々の行に記載されていなければなりません。

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 アクションはローカル 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 ストレージ読込ステップ アクションを使用して、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 アクションは、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、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
特定のキーを持っている保存項目のみを消去したい場合は、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する必要がある点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したい場合は、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点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込、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。

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よりも優先します。その他すべてのオプションは口ボットに対して指定されているものと同じになります。

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)。

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 ダイアログでアステ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。その他すべてのオプションは口ボットに対して指定されているものと同じになります。

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、Web サイトから抽出した値を、返す変数に必要な値に変換します。また、入力変数の値を、特定の Web サイトで使用する値に変換するために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。このリストは空の場合があります。例えば、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です。

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、異なる場合もあります。

動画
データ コンバータ リストの紹介と、具体的な変換方法を短い動画で見ることができます。

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ますので注意してください。

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。パスは絶対でなければなりません。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、例えば file:/C:/temp/myFile などのファイル URL です。この場合はエンコード URL でなければなりません。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。パスは絶対でなければなりません。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、例えば file:/C:/temp/myFile などのファイル URL です。この場合はエンコード URL でなければなりません。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複数の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複数の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いくつかの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、ページ読込のステップ アクションの動作と同様です。これに伴って、JavaScript の実行がオプションで無効になっていない限り、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、例えば 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いくつかの方法で指定できます。例えば、変数からコンテンツを取得することが可能です。変数がテキストやバイナリ コンテンツを伴う場合もです。コンテンツのタイプ (HTML など) は自動的に検出されます。自動検出が不十分だったり、コンテンツを別の方法で読み込む場合 (例えば、HTML ドキュメントをプレーンテキストとして読み込むなど) 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 リファレンスの解釈のために使用します。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。

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じになります。

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ますので注意してください。

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。パスは絶対でなければなりません。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も、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て、代わりに「Desktop Automation ワークフローの呼び出し」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 以前では、Desktop Automation アクション ステップに含まれている Desktop Automation 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ました。Kofax RPA 11 で、バージョン 10.7 以前で作成された Desktop Automation アクション ステップを実行できますが、ワークフローを編集するには、アクション ステップを Desktop Automation ロボットに変換する必要があります。そのためには、Desktop Automation ステップを開き、[アクション] タブの [ワークフローにエクスポート] をクリックします。新しいロボットに名前を付けて、[終了] をクリックします。

Desktop Automation ステップが Desktop Automation ロボットにエクスポートされて、このセクションの上記の手順に従って使用できるようになりました。

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・小文字を無視
このプロパティにチェックが入っていると、パターンの照合では大文字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。

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。

マーカーをクリックすると分岐内のすべてのステップの実行が可能になるため、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、コメント アクションと共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。

注 [ステップ終了] はロボットの実行を停止させません。ロ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] アクションと共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 および保存されたロボット 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 フィールドでなければなりません。

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点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ロ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
パスワードを入力する前に既存のパスワードを選択する必要があるかどうかを示します。

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じになります。

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なければなりません。

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点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。

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、テキストの読込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。例えば、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。

注 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。

注 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。

大半のステップ アクションは該当する 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性がない限り、通常、[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。
- タグに特殊属性として出現する可能性があるイベント ハンドラー。onClick や onMouseOver のように必ず "on" から始まります。JavaScript からイベント ハンドラーをアタッチして HTML ソース上で見えなくすることもできます。
- <a href="javascript:open()"> など、URL を含めることができるタグ属性の値と共に JavaScript: Protocol を指定する JavaScript URL。

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

このプロパティでは実行したい 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 ハンドラーをドロップダウン 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。

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じになります。

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。

SQL は Design Studio 内のデザインモードでの実行中には実行されませんので注意してください。

重要 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 (同一のロボット内など) 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。

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。例えば、"<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 (例えば、<?xml version="1.0" encoding="UTF-8"?>) が有る場合は、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

[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。

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。例えば、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。例えば、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とで差がないセル、例えば、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なければなりません。

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。例えば、URL に基づくファイルの名前。この名前は例えば、ロボットの次のステップで、ファイルへの保存がある場合に必要です。

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。例えば、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。例えば、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とで差がないセル、例えば、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。

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 は正規表現を使用してドメイン、パス、および/または名前で選択します。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。

Flash から抽出

このアクションでは、検知タグ内の Flash オブジェクトからコンテンツを抽出します。

静的 テキスト、HTML フラグメントおよび画像は、Flash オブジェクトから抽出して HTML ページとして表示します。含まれる URL、他の Flash オブジェクトの参照、および設定ファイルは、ページの一番下にあるリスト内の各リンクに変換されます。

プロパティ

[Flash から抽出] 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。

画像を含める
画像を Flash オブジェクトから抽出するかを指定します。

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じになります。

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。

ただし、PDF ドキュメントには、テーブルやパラグラフなどの構造上の情報は含まれません。テキストやグラフィックの位置のみが、テーブルやパラグラフのように見えるように配置されている場合と、されていない場合とがあります。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。

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るとはかぎりません。元のドキュメントへの埋め込み方法によって異なります。

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なければなりません。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

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なければならず、存在しない場合は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。例えば、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。例えば、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。その他すべてのオプションは口ボットに対して指定されているものと同じになります。

JSON 抽出

[JSON 抽出] ステップ アクションでは、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。例えば、名前 / 値のペア "answer" : 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 アクションは、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 Sheet1!A5:B7 などの範囲になります。JSON では、@top:.a[5].b などの、JSON ファインダーで使用するパスです。抽出したパスは常に絶対パスであり、名前付きタグ、範囲などには関連しません。ワイルドカード (*) シンボルはありません。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の、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 アクションでは、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) でなければなりません。たとえば、"answer" ：42 のようになります。

プロパティ

「JSON 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複数の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。

画像の読込
すべての画像をページに読込ます。ただし、「ページ読込」ステップなどで行っていない場合です。

タイムアウト **(ms)**
画像の読込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なければなりません。

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 アクション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 を読み取るように指定します。

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、「ページ読込」ステップ アクション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。

次の場所に保存

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、次のいずれかでなければなりません。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。

注 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 モードの両方で実行されます。

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

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なければならず、存在しないときは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既にあ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。ファイルが既に存在する場合は、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既に存在する場合は、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じになります。

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がオンになっている場合、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、実際の読込は行いません。このプロパティのチェックがオフになっている場合、JavaScript URL 自体を抽出します。

絶対 **URL** に変換

このチェックがオンになっている場合、相対 URL を絶対 URL に変換します。

読み込まずにクリック

URL を、タグをクリックしたときのように抽出できます。ただし、読込は行いません。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に効果的です。

読み込まずにフォーム送信

URL を、フォームを送信したときのように抽出できます。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

抽出済み URL を保存するための変数。

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じになります。

Web ストレージ抽出

「Web ストレージ抽出」ステップ アクションでは、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。抽出したデータをロ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

チェックがオンになっている場合、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

チェックがオンになっている場合、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
検知した値の読込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 タイプでなければなりません。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なければなりません。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、ファイル読込ステップを使用して、CSV ファイルを読込ます。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。

注 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 [タグ繰り返し] 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いくつかの方法で指定できます。名前は絶対ディレクトリ名でなければなりません。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

[自動]、[名前付き] という 2 つのオプションがあります。「自動」は項目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項目が (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」は項目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。

- 適切な名前を付けると、その項目の内容を思い出しやすくなります。
- 明確に名前付けされた項目は、名前付きの別の項目がその前に挿入されても影響を受けません。
- 「名前付き JSON 設定」で同じ名前を使用すると、その名前は、新しい項目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)

既存の名前付きアイテムを保持

このチェックがオンの場合、既存の名前付き項目を保持します。オフの場合は、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> タグでなければなりません。

<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点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。

詳細については、『フォームのループ』チュートリアルを参照してください。

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。

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じになります。

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

[自動]、[名前付き] という 2 つのオプションがあります。「自動」は項目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項目が (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」は項目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。

- 適切な名前を付けると、その項目の内容を思い出しやすくなります。
- 明確に名前付けされた項目は、名前付きの別の項目がその前に挿入されても影響を受けません。
- 「名前付き JSON 設定」で同じ名前を使用すると、その名前は、新しい項目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)

既存の名前付きアイテムを保持

このチェックがオンの場合、既存の名前付き項目を保持します。オフの場合は、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 つでなければなりません。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点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。

詳細については、『フォームのループ』チュートリアルを参照してください。

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。

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じになります。

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。主な違いは、「タグ繰り返し」アクションでは検出されたタグの直下の子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点です。

詳細については、『ループの基礎入門』チュートリアルを参照してください。

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。例えば、class1 class2 は、class1 と class2 の両方であるノードを指定し、class1 | class2 | class3 は、class1、class2、または class3 のいずれかのノードを指定しますが、class1 class2 | class3 class4 は、class1 と class2 の両方、または class3 と class4 の両方であるノードを指定します。

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、論理和 (OR) は | で表されます。class が明示的に 存在しないことは $ で表されます。論理積は論理和に優先します。例えば、class1 class2 は、class1 と class2 の両方であるノードを指定し、class1 | $ は、class1 かまったく class のないノードのいずれかを指定します。 class1 | class2 | class3 は、class1、class2、または class3 のいずれかのノードを指定しますが、class1 class2 | class3 class4 は、class1 と class2 の両方、または class3 と class4 の両方であるノードを指定します。

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のいずれ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のいずれ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。例えば、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。あたかも順方向に逆順でループしているかのように、まったく同じタグを通過することに注意してください。つまり、「最初のタグ番号」については、ループするときに最初にアクセスしたタグではなく、ループするタグの選択における最初のタグを参照するということです (実際には最後のタグになります)。

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。主な違いは、「タグ繰り返し」アクションでは検出されたタグの直下の子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点です。

詳細については、『ループの基礎入門』チュートリアルを参照してください。

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されます。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。例えば、class1 class2 は、class1 と class2 の両方であるタグを指定し、class1 | class2 | class3 は、class1、class2、または class3 のいずれかのタグを指定しますが、class1 class2 | class3 class4 は、class1 と class2 の両方、または class3 と class4 の両方であるタグを指定します。

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、論理和 (OR) は | で表されます。class が明示的に 存在しないことは $ で表されます。論理積は論理和に優先します。例えば、class1 class2 は、class1 と class2 の両方であるタグを指定し、class1 | $ は、class1 かまったく class のないタグのいずれかを指定します。 class1 | class2 | class3 は、class1、class2、または class3 のいずれかのタグを指定しますが、class1 class2 | class3 class4 は、class1 と class2 の両方、または class3 と class4 の両方であるタグを指定します。

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のいずれ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のいずれ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。例えば、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。あたかも順方向に逆順でループしているかのように、まったく同じタグを通過することに注意してください。つまり、「最初のタグ番号」については、ループするときに最初にアクセスしたタグではなく、ループするタグの選択における最初のタグを参照するということです (実際には最後のタグになります)。

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 アクションは、次のプロパティを使用して設定できます。

入力

値セレクターを使用して複数の方法で分割する文字列を指定することができます。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 アクションを使用してタグ テキストを変数に抽出しなくてはなりません。

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。例えば、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。

詳細については、『ループの基礎入門』チュートリアルを参照してください。

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のいずれ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のいずれ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。あたかも順方向に逆順でループしているかのように、まったく同じタグを通過することに注意してください。つまり、「最初のタグ番号」については、ループするときに最初にアクセスしたタグではなく、ループするタグの選択における最初のタグを参照するということです (実際には最後のタグになります)。

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 ページが受信された場合など、エラーとみ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複数の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いくつかの方法で指定できます。名前は絶対パス名でなければなりません。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 ステップ

グループ ステップは別のステップを含むように設計されていますが、グループ ステップを 1 つのステップにまとめることにより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のに便利な方法です。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。これをクリックすると、グループ ステップが広がったり、閉じたりします。グループ ステップを閉じると、見た目は通常のステップと同様になり、内部のすべてのステップが非表示になります。グループ ステップを広げると、含まれているステップが表示され、グループ化されていない場合に表示されるレイアウトに似ています。グループ ステップは、ツールバーの [すべて広げる]/[すべて閉じる] ボタンを使用して広げたり閉じた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 ステップに対して実行されます。また、[グループを展開]/[グループを閉じる] 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実行されます。

ステップをグループ化する (ステップを含む新しいグループ ステップを作成する) には、グループ化するステップを選択し、ツールバー、編集メニューまたはステップのポップアップ メニューにある「グループ化」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。

グループを解除する (グループ化されたステップを保持してグループ ステップを除去する) には、解除するグループ ステップを選択し、ツールバー、編集メニューまたはステップのポップアップ メニューにある「グループを解除」アクションを使用します。

(グループを広げている場合、) 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 ステップのエントリとイグジットをマークしており、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ポップアップ メニューを使用してグループ ステップ上でアクションの選択を実行できます。たとえば、グループの最初に分岐を挿入できます。

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。

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

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 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。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 アクションは 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

「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タグとして
- 検出されたタグの最後の子タグとして
- 検出されたタグの前
- 検出されたタグの後ろ

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 アクションは 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なければならず、そうでないときは 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する場合、検出された JSON 値はオブジェクトでなければならず、そうでないときは 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。これはもちろん、有効なプロパティ名でなければならず、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 値でなければなりません。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 値は配列でなければならず、そうでないときは 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。

値

これは項目の値ですが、有効な JSON 値でなければなりません。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。

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

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
「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。
- 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)。

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

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限り、新しいタグの HTML 内にある JavaScript が実行されます。

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、以下に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タグとして
- 検出されたタグの最後の子タグとして
- 検出されたタグの前
- 検出されたタグの後ろ

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。そうでない場合、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。

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 ダイアログでアステ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。その他すべてのオプションは口ボットに対して指定されているものと同じになります。

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 **HTML** を作成
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なければならず、一致してい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・小文字を無視
このチェックがオンになっている場合、パターンが入力と照合されるときに大文字小文字の区別は無視されます。たとえば、"KoFaX" は "kOfax" と同じと見なされます。

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。

ファイル読込

このアクションは、ファイルをブラウザまたは変数に読込ます。

プロパティ

「ファイル読込」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。

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じになります。

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 からページを読込ます。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

「ページ読込」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。

読込先

このプロパティで、ページの読込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、ブラウザの読込先と同じウィンドウにページを読込ます。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)。

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じになります。

Web ストレージ読込

「Web ストレージ読込」ステップ アクションは、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込ます。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 アクションは、Web ストレージ抽出ステップ アクションを使用して抽出されたストレージ データの読込にも使用できます。

プロパティ

「Web ストレージ読込」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りますが、特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しなければなりません。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」のいずれかでなければなりません。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、文字列タイプでなければなりません。

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 ストレージ読込」ステップ アクションに入力します。

```
    [
            { "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。

詳細は、「フォームでのループ」チュートリアルを参照してください。

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。

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 ダイアログでアステ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。その他すべてのオプションは口ボットに対して指定されているものと同じになります。

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エ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エレメントの数字。数字は、最初のエレメントから数えるか、最後のエレメントから数えるか、いずれかによって指定できます。

最終インデックス

ループに含める最後のエレメントの数字。数字は、最初のエレメントから数えるか、最後のエレメントから数えるか、いずれかによって指定できます。

増数

ループでエレメントをスキップさせます。例えば、増数が 2 に指定されれば、ループは 1 つおきにエレメントをスキップします。

逆方向にループ

逆順で一致するエレメントをループすることを選択します。ループは、単に逆順で、前からループするのとまったく同じエレメントを通過することに注意してください。これは、開始インデックスが、ループするエレメント選択の最初のエレメントを指し、ループ時に最初に通過するエ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エ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

このアクションは、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 アクションは、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、Desktop Automation 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 管理者はユーザーのアクセス 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。トークンを取得するには、Design Studio 内で **[ヘルプ]** > **[バージョン情報]** に進み、**Design Studio** アクセス トークン セクションからトークンをコピーします。次に、コピーしたトークンを管理者と共有します。

重要 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、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。

注 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っている場合、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。

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ますので注意してください。

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。パスは絶対でなければなりません。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 アクションは、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。

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 アクションの使用を検討してください。「ページ再描画」ステップ アクション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 アクションは、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 アクション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じになります。

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。

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じになります。

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。

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じになります。

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される反復ループで、別のイテレーションをリクエストします。

「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。

詳細は、「繰り返し - 次へ」チュートリアルを参照してください。

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。

このアクションはページを修正するため、JavaScript またはフォーム送信が動作を停止する原因となる可能性があります。ページ構造に依存してる場合があるためです。ただし、このアクションは、通常、そこからナビゲーションがさらに発生しないデータ テーブルに適用されるため、問題が発生することはめったにありません。

プロパティ

「データベース データ削除」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。対応するバージョンの Kofax RPA 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、『Release Note』(リリース ノート) ドキュメントおよび『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) の「Kofax RPA Upgrade Guidelines」(Kofax RPA アップグレード ガイドライン) の章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、例えば、Web サービスのパラメータに必要な内容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 アクションは、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 プレスをエミュレートします。

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 に使用されるキーの名前については、Qt::Key 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。

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 ダイアログでアステ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。その他すべてのオプションは口ボットに対して指定されているものと同じになります。

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。

注 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。

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。

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なければならず、変数名は既存の変数のリストから選択されます。列のタイプは選択された変数のタイプと一致していなければならない点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 (例えばデータベース データ登録ステップ) 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。例えば、

- 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込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複数の方法で指定できます。

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したい場合は、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、そのレスポンスの一部としてデータを返さないため (返されるのはステータス コードとレスポンス ヘッダーのみ)、そのデータにアクセスするために、ページ読込オプションの一部として定義された変数を使用します。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、そのレスポンスの一部としてデータを返さないため (返されるのはステータス コードとレスポンス ヘッダーのみ)、そのデータにアクセスするために、ページ読込オプションの一部として定義された変数を使用します。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。

注 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。

オブジェクトを再検出

このステップは、「データベースへのデータの保存」で説明している新しいストレージ メカニズムで使用できないため、廃止される予定です。

このアクションは、ストレージ、すなわちデータベース内のオブジェクトの再検出を試みます。オブジェクトを再検出できる場合、現在の分岐に沿った実行は停止します。再検出できない場合は、「オブジェクトを再検出」アクションが含まれたステップを超えて抽出が続行されます。その結果、「オブジェクトを再検出」アクションはロボットの実行時間を最小化する手段になります。

プロパティ

以下のプロパティを使用して「オブジェクトを再検出」アクションを設定できます。

変数

[オブジェクト] ドロップダウン ボックスからオブジェクトを選択することによって、影響を受けるオブジェクトを選択することができます。ドロップダウン ボックスに表示されるオブジェクトは設定済み出力オブジェクトです。

再検出に使用する属性

このボックスは、ストレージ内のオブジェクトを一意に識別するオブジェクト属性のリストです。「オブジェクトを再検出」アクションは、ロボットの実行の現在の時点で、それまでに見つかったものと一致するストレージ内のオブジェクトの検出を試みます。そのようなオブジェクトが存在する場合は、ストレージ内のその他のフィールドも正しい可能性が高いため、継続することにほとんど意味はありません。

オブジェクトの有効期間全体にわたるオブジェクト属性のグローバルな一意性を保ちながら、選択するオブジェクト属性の数をなるべく少なくするよう試みる必要があります。

いずれのオブジェクトにも存在しないオブジェクト属性またはオブジェクトの有効期間中に変化する可能性のあるオブジェクト属性を含めないでください。

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 アクションは 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。

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 アクションは 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

「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。このアクションはオブジェクトからプロパティ (名前/値ペア) を除去するか、配列から項目を除去するか、JSON 値全体を除去します (その場合、結果は空の JSON 値になります)。

このステップ アクションは 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。

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。

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
テーブル行に存在してもよい列の最大数を入力します。

範囲 [列の最小数; 列の最大数] に収まる数の列を持っていない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> タグを持っている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 アクションは 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。

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。

注 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。

タグ照合規則の構成

リストの下の '+' または [編集] をクリックして開くタグ マッチャー規則工ディターでは、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

「子を含む」オプション

「子を含む」チェックボックスをオン (デフォルト) にすると、その規則に子が含まれます。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 (その子を含めて) 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。

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ますので注意してください。

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。パスは絶対でなければなりません。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。

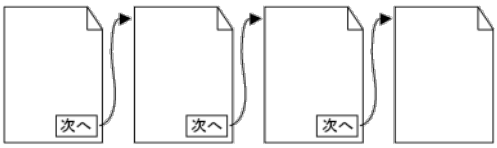
繰り返し

このアクションでは、「次へ」アクションと共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。

「次へ」ステップ アクションを実行したときに、ループ対象となる結果を提供するデータベース クエリがあれば、ロボットは実行を続行します。この場合、ロボットは「繰り返し」ステップに戻る前に、データベース クエリの結果内でループします。

「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ときに特に役立ちます。



「繰り返し」アクションの使い方を示す例については、「各ページが次のページにリンクしているページ (Design Studio セクション)」を参照してください。

詳細については、Repeat-Next のチュートリアルも参照してください。

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限り、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。このトピックの後半で説明するように、さまざまな方法で HTML を指定することができます。

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じになります。

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 **HTML** を作成
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なければならず、一致してい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。

大文字・小文字を無視
このチェックがオンになっている場合、パターンが入力と照合されるときに大文字小文字の区別は無視されます。たとえば、"KoFaX" は "kOfax" と同じと見なされます。

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。例えば、$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。

デフォルト オプションを使用

これにチェックが入っていると、(ロボットの設定の一部として設定されている) ロボットのデフォルト オプションが使用されます。チェックが入っていないと、以下の [オプション] プロパティで設定された特定のオプションが使用されます。

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、以前別のロ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。

注 [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。

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、ブラウザがアイドルになった場合に (どちから早い方を優先) ブラウザを再開し、実行します。

このステップの待機条件を指定します。詳細については、待機基準の使用を参照してください。

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 アクションは、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。

Kofax RPA 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。Kofax RPA 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。

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。

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やるべきことを選択します。

ページ再描画

「ページ再描画」ステップ アクションは、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、既にページとページのフレームを読込、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 アクション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 アクション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 アクションは、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 アクションを使用できます。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 アクションと異なり、「スナップショット生成」ステップ アクションでは、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。

注 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 アクションを設定できます。

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。これはページの読込に使用された 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 アクションが元のページ 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 アクション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 の両方が含まれた 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 アクションを使用します。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かもしれません。

URL

抽出された URL の保存先となる変数。ステップ アクションは、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点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 の両方が提供されます。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。例えば、<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点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

例えばフレーム内の PDF または Flash オブジェクトなどのバイナリ リソース。

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 アクションを使用します。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かもしれません。

CSS 再描画

「CSS 再描画」ステップ アクションは、「ページ再描画」ステップ アクション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 アクションは、「ページ再描画」ステップ アクション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 アクションの 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。

注 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 アクション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 アクションが後に元のページ 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 アクションと同じ方法で 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 アクションは、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点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 の両方が提供されます。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。例えば、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 アクションを使用します。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かもしれません。

オプション
ステップの読込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 ウィンドウでアステ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。その他すべてのオプションは口ボットに対して指定されているものと同じになります。

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。口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、他の口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、セッションを口ボットからの出力値として保持して、その後、それを別の口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、別の口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。

注 [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。

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 アクションの使用目的は、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複数の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。

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じになります。

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。

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じになります。

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。

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じになります。

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。例えば、GIF 画像については "image/gif" というテキストが変数に含まれます。

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。例えば、テキスト コンテンツについては "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> タグでなければなりません。

<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点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、選択したいオプションだけを指定します。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。

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じになります。

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・小文字を無視

チェックをオンにすると、大文字と小文字を区別しないで、オプション値との照合を実行します。

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じになります。

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。

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じになります。

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。

注 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。

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。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> x64\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。そうでない場合は、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 アクションは 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。

チェックボックス設定

このアクションで、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 ハンドラが有る場合は、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
設定するべきチェックボックスの状態を指定します。例えば、オンにする、オフにするなどです。状態の指定には値セレクターを使用します。結果の値は、チェックボックスをオンにする場合は

"true"、"on"、"1"、または "checked" です。チェックボックスをオフにする場合は "false"、"off"、"0"、または "unchecked" です。

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じになります。

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 アクションは 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

「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

このアクションでは、スプレッドシートのセルに指定したコンテンツを挿入します。

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。

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 を現在のウィンドウ、つまり次以降のステップが操作の対象とするウィンドウとして設定します。

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。

プロパティ

以下のプロパティを使用して [カレント ウィンドウ設定] アクションを設定できます。

現在のウィンドウとして設定するウィンドウ
現在のウィンドウとして選択する必要があるウィンドウ (ウィンドウを識別する方法については、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することができます。
- 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を検索します。
- 見つかったタグのウィンドウ: このオプションは検出されたタグによって作成されるウィンドウを現在のウィンドウにします。検出されたタグは FRAME 要素、IFRAME 要素、OBJECT 要素または EMBED 要素でなければなりません。

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 アクションは 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 値でなければなりません。

このステップ アクションは 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。
値セレクターの拡張バージョンを使用して複数の方法で値を指定することができます。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

このアクションは、検出された JSON を名前付き JSON としてマークし、次以降のステップで JSON 値の他の部分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。

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
[自動]、[名前付き] という 2 つのオプションがあります。「自動」は項目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項目が (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」は項目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。

- 適切な名前を付けると、その項目の内容を思い出しやすくなります。
- 明確に名前付けされた項目は、名前付きの別の項目がその前に挿入されても影響を受けません。
- 「名前付き JSON 設定」で同じ名前を使用すると、その名前は、新しい項目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)

既存の名前付きアイテムを保持
このチェックがオンの場合、既存の名前付き項目を保持します。オフの場合は、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。

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 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)

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。

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」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 アクションは 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。これはもちろん、有効なプロパティ名でなければならず、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。制御はスライダーで実行できます。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。

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

「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。

高さ
このプロパティは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 アクション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 アクションは [コンテンツ設定] ステップ アクションと似ていますが、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 アクションではタグ自体を置き換えます。

このステップ アクションは 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 アクションは 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。

コンテンツ変更モード
このプロパティはソースを変更するモードを指定します。

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なければなりません。

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 アクション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 アクションは [コンテンツ設定] ステップ アクションと似ていますが、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 アクション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 アクションは 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。

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。

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 - 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っていなければなりません。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が変更されると、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すべてのスニペット ステップがスニペット ステップに含まれているステップを変更します。

詳細については、スニペットのチュートリアルの概要を参照してください。

プロパティ

以下のプロパティを利用してスニペット ステップを設定します。

ステップ名
これは省略可能なステップの名前です。このプロパティの値を入力すると、その値がステップの下のロボット ビューに表示されます。ステップ名が入力されていない場合は、その代わりにスニペットの名前が表示されます。その場合は、名前に下線が引かれます。

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。これは、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。選択ボックスから別の名前を選択すると、スニペット ステップの場所に新規作成スニペットが挿入されます。スニペットにコメントが設定されている場合は、そのコメントが選択ボックスの下に表示されます。

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、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。

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、正規変数でなければなりません。

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既に存在する場合、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。

HBase テーブル データ登録

重要 このステップは廃止されています。

このステップは変数を HBase テーブルに保存する操作を行います。接続設定はステップで設定されます。

以下の説明は、ユーザーが HBase クラスタの一般的なアーキテクチャを理解していることを前提としています。データがどのように HBase テーブルに挿入されるかについては、以下の記載内容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 [HBase テーブル データ登録] を設定できます。

ZooKeeper Quorum ホスト名

HBase へのアクセス管理を行う、基礎的な ZooKeeper Quorum への接続に使用されるホスト名のリスト。Quorum は、変数が保存されるテーブルを持つ HBase マスターへのアクセスを与える必要があります。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リスト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。

HBase テーブル

どの HBase テーブル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テーブル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。

列ファミリー

どの列ファミリー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列ファミリー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、正規変数でなければなりません。

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既に存在する場合、このステップは既存のレコードを更新します。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。キーに関する重要な情報について、以下の記載内容を参照してください。

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。

注

HBase によって保存されるすべてのデータ (キーおよび列名を含む) はバイトの配列であるため、Design Studio は、データをテーブルに挿入する前に UTF-8 エンコーディングを使用してすべてのデータをバイト配列に変換します。

変数データは属性キーで指定されたキーと共に行に書き込まれます。キーが「データベース キーの一部」としてマークされた属性から構成されるように設定されている場合、キーは、下線で区切られた (かつタイプ エディター上と同様に順序付けされた) 属性の値を連結したものになります。たとえば、タイプが Company であり、「データベース キーの一部」としてマークされた 2 つの順序付けされた属性を持つ変数を書き込むとします。

- Name = Kofax
- Year = 1999

結果として生成されるキーは Kofax_1999 です。バイナリ属性をキーの一部として選択する場合は、実際の値の代わりに値のハッシュが使用される点に注意してください。さらに、「データベース キーの一部」として使用される属性値に含まれている下線は、二重下線に置き換えられます。

HBase テーブルへのデータの書き込みは、個々の属性の値を、属性の名前と同じ修飾子を持つ列に挿入することによって行われます。列は指定された列ファミリー内に存在します。「データベース キーの一部」として選択されている属性を含む、変数内のすべてのデータは、各行に書き込まれます。「データベースへのデータの保存」セクションに記載された、2 つのハウスホールド フィールド「RobotName」と「ExecutionId」については、ステップが実行されるたびにテーブル内で更新されます。このページに記載される残り 5 つのハウスホールド フィールドはテーブルに保存されません。書き込みに固有のタイムスタンプは選択されません。妥当な値を選択する役割は HBase が担います。上記の例および列ファミリー = cf を使用すると、挿入されるデータは以下のようになります (タイムスタンプは無視します)。Kofax_1999 => [cf:Name = Kofax, cf:Year = 1999, cf:RobotName = XXXX, cf:ExecutionId = YYYY]

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なければなりません。

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
これには満たされる必要のある条件が含まれます。この条件には次の 6 つがあります。

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いくつかの方法で指定できます。名前は絶対パス名でなければなりません。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、次の 2 つの一般的なタイプを判定することもできます: 基本タイプと数値タイプ。

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

これには満たされる必要のある条件が含まれます。この条件には次の 9 つがあります。

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、同じように配列のオブジェクトを返す可能性がある JavaScript typeof 演算子と等価ではありません。

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 (\) で工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。

注 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" の両方が 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 4 つのページ タイプがあります: HTML、XML、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
これには判定するページ タイプが含まれます。これには 4 つの可能性があります。

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。

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> タグでなければなりません。[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・小文字を無視
このオプションにチェックが入っていると、パターンの照合では大文字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。

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。
テキストが指定された長さを持っているかを判定します。

```
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。

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複数の方法で値を指定することができます。

大文字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、比較は大文字小文字の区別を無視して行われます。

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。
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。 注 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。

注 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なければなりません。

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
存在の有無を確認するウィンドウ (ウィンドウを特定する方法の説明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。

注 XML の変換ステップ アクションは、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。出力値がテキスト変数に格納される場合、XML、HTML、およびテキストはすべてテキストとして格納できるため、出力のメソッドは何でも構いません。

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。

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

注 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代替手段を試行 (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。詳細な説明は、Design Studio ヘルプの「複数の分岐の試行」に記載されています。

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。

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 アクションは、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 アクションは、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

現在、プロパティはありません。

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 アクションは、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。

このステップのアクションは、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

現在、プロパティはありません。

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、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)、ファイル出力を末端で使用して (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 コンテンツを保持するグローバル変数を変更するには、変数自体を指定している「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いくつかの方法で指定できます。名前は絶対パス名でなければなりません。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複数の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、上記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、ファイルが既に存在する場合に、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端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複数の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。

データの変換のチュートリアルを表示することもできます。

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。抽出したい部分を 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。抽出したい部分を 1 組の括弧でマークする必要があります。

アクション	説明
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。既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。

アクション	説明
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。既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 日付へ	日付を標準の日付書式から Excel 書式へ変換します。
Excel 日付から	日付を Excel 書式から標準の日付書式へ変換します。

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・ テキスト変換	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
XML に追加	このデータ コンバータは 1 つの XML を別の XML で拡張し、オプションで、最初に変数から XML を作成します。

エンコーディングとデコーディング

このカテゴリーには、エンコーディングとデコーディング用のデータ コンバータが含まれます。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· テキスト変換	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· バイナリ変換	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。

アクション	説明
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フェッチします。
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

入力テキストの末端に追加する前に、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・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、

```
    &alt; &gt; &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。

```
    &alt; &gt; &amp; &quot; &aacute; &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
```

重要 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] リストの右のドロップダウン 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] の中から選択することもできます。

大文字・小文字を無視

このオプションが選択されると、アクションはパターンの内容の大文字小文字の区別を無視します。

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。出力では大文字小文字が区別される点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：

キャピタライズしてはいけない語
キャピタライズしてはいけない語を含めます。語はカンマで区切られます。例えば、"in" および "the" という語をキャピタライズしてはいけない場合は、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。例えば、このプロパティが 3 に設定されていると、3 文字未満のすべての語は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しか適用できないため、MD5 を適用する前にあらゆる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。例えば、"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・テキスト変換データ 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なければなりません。

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・テキスト変換

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・テキスト変換データ コンバータを設定できます：

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・バイナリ変換データ 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 コンバータは例えば、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。

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。これは文字通りに、または値セレクターを使用していくつかの方法で指定できます。

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。

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。ビルトインの 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。そうでない場合は、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**

金額のコンマ区切りリスト、例えば "$10.50,$13,$21.75,$7" 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。ビルトインの 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)。

メソッド	説明
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) で指定するグローバル属性がない限り、最初の一致だけを返します。グローバル属性がある場合は、一致の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。
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。

プロパティ / メソッド	説明
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。例えば、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。例えば、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"
...
```

この変換リストによれば、例えば入力テキスト "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、カウントしたいタグの名前を入力します。例えば、"tr" です。

終了タグが必要
これを選択すると、開始タグと終了タグの両方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。例えば、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)。

さらに高度な抽出オプション、例えば複数のテキストを抽出する場合や、操作オプションがある場合は、アドバンスド抽出データ コンバータを代用してください。

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・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。

日付抽出の手法についての詳細は、アドバンスド チュートリアル セクション内の日付抽出チュートリアルを参照してください。

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。これは例えば、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、これはエクスプレッション now() で、現在の日付と時間を得られます。

時間軸の方向

抽出する日付が過去か未来かを指定します。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。例えば、「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。例えば、「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
G	紀元の表記 (AD または BC)
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 (例えば、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 (例えば、"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。例えば、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一緒に使用する場合 (おそらく月および年の場合も)、曜日を破棄します。例えば、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 (例えば 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。例えば、入力が次のテキストの場合

```
Thus, let us meet on Wednesday
```

'EEEE' のパターンを使用します。 'EEE' のパターンでは 'Thu' に一致するため、日付抽出で次の木曜日を検知してしまうからです。

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
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。

例えば、「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、一致があればそれぞれ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させる必要はないので注意してください。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・小文字を無視
このチェックがオンになっている場合、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：既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がオンになっている場合、抽出した番号を整数に変換します。

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) と一緒に示します。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、番号 2.0 が抽出されます。double kilo という名前の定数がないからです。

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。

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。

注 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。

注 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、番号 6.023E23)、通常、「整数に変換」のチェックをオンにしません。このような大きな数を整数に変換すると、誤った結果を招くことがあるからです。

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」のチェックがオンになっている場合、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」のチェックをオン、定数を mill.=1000000.0 および bill.=1000000000.0、と設定すると、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
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")

パターン	説明
*	任意の文字数をスキップします (これを使用するのは、英字 a ~ z および A ~ Z をスキップするときです)
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。例えば、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一緒に使用する場合 (おそらく月および年の場合も)、曜日を破棄します。例えば、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。

注 'EEE' パターンは、曜日を示す短い名前に一致します (例えば 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。例えば、入力が次のテキストの場合

それでは、水曜日 (Wednesday) に会いましょう

'EEEE' のパターンを使用します。 'EEE' のパターンでは 'Thu' にマッチするため、「年抽出」データ コンバータで次の木曜日の年を検知してしまうからです。

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。入力するテキストを、標準的な日付書式、例えば "2001-02-25 14:32:49.0" のような形式にする必要があります。

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
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

特殊文字	説明	タイプ	例
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。例えば、':'、'.'、' '、'#'、'@' 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 :

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 アクション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、例えば "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ください。

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。

注 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フェッチします。

この変数は、プロパティ変数タイプでなければなりません。

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、例えば "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いくつかの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" のような書式にしなければなりません。

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。

Then
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。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 (コンバータは使用しません)。

大文字・小文字を無視
これがオンになっていると、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。例えば、"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 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。

Then
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。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" のとき出力テキストを "Porsche 911" にしたい、と仮定します。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。

入力するテキストを、標準的な日付書式、例えば "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。

量
日付に加算したり減算したりする量。量は値セレクターを使用して指定されます。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点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] の両方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点に注意してください。

大文字・小文字を無視
このプロパティにチェックが入っていると、パターン照合で大文字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
このプロパティにチェックが入っていると、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。チェックが入っていないと、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・小文字を無視
このプロパティにチェックが入っていると、テキスト照合で大文字小文字の区別が無視されます。

すべてを置き換え
このプロパティにチェックが入っていると、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。チェックが入っていないと、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
このプロパティにチェックが入っていると、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
```

タイプ エディター

タイプ エディターは、Design Studio で使用される、タイプの書き込みおよび維持を行うために使用されます。タイプ エディターを使用することで、新しいタイプの作成および設定を簡単に行うことができます。

このセクションでは、タイプと属性の設定についての詳細な説明を含め、Design Studio のタイプ エディターのユーザー インターフェイス機能についてのヘルプを説明します。

タイプ エディータについての詳細情報は、Design Studio の『ユーザー ガイド』をご覧ください。

タイプ設定

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、タイプの種類、タイプに付いているコメント (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。また、プロジェクトにおいて一意である必要があります。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。新規属性を追加するには、テーブルの下の [新規属性の追加および設定] をクリックします。属性を除去するには、属性の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、[属性の設定] 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。

タイプの種類

タイプの種類の選択は、下位互換性のために用意されています。通常、「標準タイプ」のタイプの種類が使用されます。つまり、このプロパティの設定は不要です。ドロップ ダウンでは、以下の選択肢を利用できます。

- 標準タイプは、タイプが標準であることを表します。
- データベース出力タイプ (レガシ) は、タイプがデータベース出力タイプであることを表します。これは、7.2 以前のバージョンの Kofax RPA で作成されたデータベースにデータを保存する際に必要です。このタイプの種類は、下位互換性目的のみを対象とします。データベース出力タイプには、最低でも示されている属性タイプを持つ、以下の属性が必要です。

属性名	属性タイプ
robotId	整数
robotRunId	整数
refindKey	再検索キー
firstExtractionDate	日付
latestExtractionDate	日付

属性名	属性タイプ
extractedInLatestRun	ブール値

これらの属性が必須となる理由は、データベース出力タイプの変数から抽出された値がストレージに保持される場合、Kofax RPA がデータベースにおいて同じ 'refindKey' を持つ値の検索を試行するためです。このような値が存在する場合は更新されます。存在しない場合は、抽出された値が挿入されます。'firstExtractedDate' は、初めて値が抽出されたときの日付と時刻です。また、'latestExtractedDate' は、抽出された値が最後に挿入または更新されたときの日付と時刻です。'extractedInLatestRun' は、値が最新の口ボット実行で抽出された場合に 'true' となります。これら必須の属性の一部がデータベース出力値タイプから不足していると、そのタイプは無効となります。タイプ エディターは、この場合に警告を表示し、不足している属性の追加についての可能性を表示します。

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。

属性の構成

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 エディタでのみ表示されます。

[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、タイプ内で一意である必要があります。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。

属性タイプ

属性タイプのリストから属性の属性タイプを選びます。

デフォルト値

属性のデフォルト値を設定します。

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) は、(ファイルまたはデータベースに) 該当するタイプの変数を保存しない。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 Robot の実行が開始されない。

コメント

このフィールドでは、属性の詳細な説明を入力する、オプションのコメントを属性に追加できます。

[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

オプションとしての別の名前です。これは属性を保存する際に使用し、例としてデータ ベースの行名、CSV ファイルの列のヘッダー、または XML のタグ名などを入力します。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 Robot 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。

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、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、良い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、Robot 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 Robot に適切な設定が行われていても、1 つの Robot に別のキー計算が必要となる場合は、Robot 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。

先に区切り記号を表示

タイプを Design Studio の Robot 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。

属性タイプ

以下の属性タイプを利用できます。

整数
整数 (例：12)

数値
数値 (例：12.345)

ブール値
ブール値 (例： "true" または "false"

ショート テキスト
ショート テキスト。Kofax RPA は、1 行テキスト フィールドにおいてショート テキスト属性を表示します。

文字
単一の文字 (例： "A")

ロング テキスト
ロング テキストKofax RPA は、複数行テキスト ボックスにおいてロング テキスト属性を表示します。

パスワード
パスワード テキストKofax RPA は、パスワード フィールドにパスワード属性を表示します。このフィールドでは、パスワードの文字の代わりにアスタリスクが表示されます。

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点を除いてロング テキストと同じです。

XML
XML ドキュメント。適格な XML 文書のみが許容される点を除いてロング テキストと同じです。

日付
日付日付は、yyyy-mm-dd hh:mm:ss.n の形式である必要があります (例：1992-04-25 10:33:06.0)。

バイナリ
バイナリ データ (例、任意のバイトの連続)

イメージ
イメージ。これは、画像をプレビューできることを除き、バイナリと同じです。

PDF
PDF 文書。これは、PDF ドキュメントをプレビューできることを除き、バイナリと同じです。

プロパティ
名前/値ペア プロパティのリスト。これは、各プロパティが名前/値ペアであるプロパティのリストを示すテキストを除き、ロング テキストと同じです。この属性タイプの詳細については、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
ページ、ページ URL、参照元 URL、Cookie、認証、およびタイム スタンプから構成されるセッション。

通貨
ISO-4217 基準によって定義される通貨コード (ユーロは、"EUR")。

国
ISO-3166 基準によって定義される国コード (ドイツは、"DE")。

言語
ISO-639 基準によって定義される言語コード (ドイツ語は、"de")。

再検索キー
値を再検索するのに使用する特殊再検索キー。

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。

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、引用符を付けて指定しても付けずに指定してもかまいません。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。

テーブルの作成と削除

抽出した変数値をデータベース データ登録したい場合は、マッチング テーブルを作成する必要があります。Design Studioがこれらのテーブル生成を支援します。

ツール メニューで「データベース テーブルの生成」を選択するとウィンドウが開きます。テーブルを作成したいデータベースの名前とタイプを選択できます。「SQL を生成」をクリックすると、テーブル生成用の SQL ステートメントの候補が表示されます。デフォルトで 'Add Drop Table' のチェックボックスが選択されています。これにより、それぞれの 'CREATE TABLE' ステートメントの前に 'DROP TABLE' ステートメントが来るようになり、該当するテーブルがすでにある場合は除去されるようになります。テーブルが無い場合は、このステートメントは無効です。

生成された SQL は、ニーズに合わせて実行前に変更することができます。例えば、テキスト属性の列タイプを 'VARCHAR(255)' から 'VARCHAR(50)' に変更したり、オートインクリメントのプライマリキーを追加したりできます。ただし、通常は、テーブルの名前や列の名前を変更したり、列を除去したりしないでください。

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。リストで指定される 1 つ以上の RoboServer が利用できる場合、これによって簡単なフェイルオーバーが提供できます。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 へ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、その口ボットが「冪等」、つまり口ボットの起動が繰り返されても一台の口ボットと同じ効果を持っていなければなりません。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口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。

口ボット ライブラリ

口ボット ライブラリは、Kofax RPA の口ボットおよびタイプのコレクションです。口ボットを実行する要求を受け付けると、RoboServer は口ボット ライブラリの中で、その口ボットと関連するタイプを検索します。

口ボット ライブラリ	説明
デフォルト 口ボット ライブラリ	RoboServer は、現在のプロジェクト 口ボット ライブラリを使用して口ボットおよびタイプを検索します。 そのため、開発中の口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 の口ボット ライブラリ ファイル	RoboServer は、URL からライブラリを 1 度読み込んで、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口ボット	このオプションで、RoboServer に送るすべての要求の中に口ボット ライブラリを埋め込みます。その後、RoboServer はこのライブラリを使用して、要求を満たすために必要な口ボットおよびタイプを抽出します。
URL の口ボット ライブラリ フォルダ	指定の URL に関連する口ボットを検索するように、RoboServer に指示します。 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 の口ボット ライブラリ ファイル

URL
口ボット ライブラリ パッケージの場所。

キャッシングを許可

オプションを有効にすると、RoboServer がロボット ライブラリをキャッシュできるようになります。URL コンテンツをキャッシュできない場合は、RoboServer は要求を受けるごとに URL のコンテンツを引き出します。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。

Management Console へのアップロード

Management Console に使用するロボットおよびタイプやスニペットをパブリッシュします。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。Management Console の接続設定については、Management Console を参照してください。

Management Console

リストから Management Console の 1 つを選択します。このリストには、Design Studio が接続されている 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。

その他のトピック

このセクションでは、Design Studio で使用されるその他の重要な概念の一部に関するリファレンス ヘルプを説明します。

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。

[基本] タブ

デフォルト オプション

ロボットのステップ アクション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、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 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。

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

HTTP クライアント **(**クラシック ブラウザのみ**)**
リモート サイトへの HTTP 要求を作るために使用するクライアントです。

NTLM 認証 **(**クラシック ブラウザのみ**)**
Kofax RPA は、(プロキシ システムおよび対象システムの両方について) HTTP 経由の NTLM 認証スキームに対するビルトイン サポートを備えます。

JCIFS 認証 **(**クラシック ブラウザ限定**)**
クラシック ブラウザ ロボットについては、Kofax RPA がお使いのシステムで認証できない場合には、JCIFS という名前の、代わりの NTLM 認証エンジンを利用できます。JCIFS を使用するには、http://jcifs.samba.org から JCIFS Library バージョン 1.3.16 JAR ファイルをダウンロードして、それを Kofax RPA インストール ディレクトリの lib フォルダに置きます。 "JCIFS" を NTLM 認証として選択して、ロボットの構成の中で使用できるようにします。

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) 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、再実行を回避必要がある場合に限られます。例えば、これによってパートナーのシステムで、データの間違いや重複が起きるような場合です。

バージョン

保存したロボットのバージョンや、ロボットを最後に編集した Design Studio のバージョンを示します。

デフォルト オプション

ここでは、アクションのデフォルト 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、一般的な読込のプロパティが含まれ、ページ読込とその他のタイプの読込の両方に使用されます。

クレデンシャル

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。標準のクレデンシャルを選択する場合、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複数の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複数の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

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」としてマークされるインストールされた証明書の 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]: インストールされた時に定義された ID を提供することで、インストールされた証明書の中の 1 つを選択します。
- [変数からの証明書]: 証明書はバイナリ変数の値として与えられます。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。
- [変数からの **ID**]: 変数の値として ID を与えて、インストールされた証明書の中の 1 つを選択します。

SSL/TLS

このプロパティは、HTTPS URL から読み込まれる時に使用する SSL/TLS のバージョンを指定します。これは、TLS では動作しないとか、SSL の弱いセキュリティを受け入れないといった、使用される SSL/TLS バージョンによって異なる結果を生成するサイトがあるため、設定可能になっています。サイトとのネゴシエーションで、SSLv3 またはその後継の TLS から選択するか、両方を受け入れることができます。いずれの場合も、プロトコルのネゴシエーションが SSL の Hello または TLS の Hello で開始されることを指定できます。

SSL 証明書を検証 **(**デフォルト ブラウザのみ**)**

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。

エミュレートするブラウザ **(**クラシック ブラウザのみ**)**

このプロパティは、何かを読み込む時に、どのブラウザとしてアクションを表示するかを指定します。古いブラウザとして表示されると、単純なページが提示される場合があります。ただし、一般的にはデフォルトを使用し続けることが推奨されます。通常はリモートの Web サーバーが Kofax RPA の組み込みブラウザと互換性のある JavaScript などを実行することになるためです。

むしろ、匿名性の観点から、以下に説明される "HTTP User Agent" プロパティを変更するべきです。

認証方法

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

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。デフォルトでは、User-Agent ヘッダ値は「エミュレートするブラウザ」で指定されます。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

このプロパティは、JavaScript による問い合わせ時および何かを読み込む時の両方で、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

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。

Flash バージョン **(**クラシック ブラウザのみ**)**

このプロパティは、JavaScript によって問い合わせられる場合にサポートして表示される Flash のバージョンを指定します。

この **URL** から参照

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

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

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。

デフォルトのブラウザ エンジン

デフォルト設定の [有効] では 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] オプションは、HTTP キャッシュを無効化します。[アグレッシブ] オプションは、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。

クラシック ブラウザ エンジン

デフォルト設定は [スタンダード] です。スタンダード HTTP キャッシュ モードは、HTTP キャッシュを有効化して、HTTP キャッシュ ルールに基づき透過的に HTTP レスポンスをキャッシュします。[**JS** および **CSS** のキャッシングを強制] に設定すると、キャッシュ ルールが上書きされ、ロボットは強制的に JavaScript とスタイルシートをキャッシュします。これによって、レイテンシが大きいサイトのパフォーマンスが向上する場合があります。[無効] を選択すると、すべての HTTP キャッシュが無効化されます。

最大試行回数

このプロパティは、読込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**)**

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**)**

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

このプロパティは、送信する追加の HTTP ヘッダを含む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

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

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

このプロパティは、ページやリソースの読込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、ページの読込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

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。

ページ コンテンツ エンコーディング (クラシック ブラウザのみ)

このプロパティは、読み込まれるページの文字エンコーディングを指定します。「自動」設定はほとんどの状況に対応しますが、URL に応じて、アクションで読み込まれるすべてのページとテキスト リソース (外部 JavaScript ファイルなど)、またはその一部に対してのみ、特にページのエンコーディングを設定する必要がある場合があります。

フォーム パラメータ エンコード (クラシック ブラウザのみ)

このプロパティは、フォームを送信する時にエンコーディング フィールド値に使用する文字エンコーディングを指定します。通常、「自動」設定で十分ですが、送信されるデータに不正な文字による問題が発生する場合、ここで特定のエンコーディング設定を試すことができます。

メタ リダイレクトに従う

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。

XSL スタイル シートを適用 (クラシック ブラウザのみ)

このプロパティは、XML を含むページの読込時に、参照される XSL スタイルシートを適用するかどうか指定します。例えば、ブラウザに表示されることを意図した XML ドキュメントは、XML ドキュメントを HTML に変換する XSL スタイルシートへの参照を含む可能性があります。

プリロードを使用 (クラシック ブラウザのみ)

HTML ドキュメントに JavaScript やスタイルシートをプリロードします。つまり、Web サーバーから HTML レスポンスを受信すると、すぐにリソースの読込を開始します。このオプションを有効にすると、リソース読込完了のために各ステップが待機する時間が削減されます。リソースの準備が完了するまでブロックしなければならない状態にロボットが到達する前に、読込が開始されるためです。

フレームをロード

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。

非サポート形式を読み込む (クラシック ブラウザのみ)

このプロパティは、サポートされない形式のコンテンツを読み込むかどうか指定します。サポートされない形式とは、Design Studio が解析できず、動画形式などのページ ビューに提示できない形式のことで

す。そのような形式のリソースは、読込に時間がかかり、ロボットがコンテンツにアクセスできない場合が多いため、リソース コンテンツの読込によってロボットの実行が遅くなる可能性があります。コンテンツの読込がオフになっている場合も、ヘッダとステータス コードは、レスポンスから必ず取得されます。この機能によって、読み込むことなくリソースのヘッダ情報を取得することもできます (リソースがサポートされない形式である場合)。これは、単なる HEAD リクエストの使用とは異なります。HEAD リクエストは、初期リクエストのヘッダ情報のみ取得し、META または JavaScript リダイレクトを通じて取得されるリソースを取得しません。

ロードするイメージ

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、画像の読込にページ 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、ページの画像の読込を選択できます。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。

ウィンドウごとの最大読込数 (クラシック ブラウザのみ)

このプロパティは、アクションで許容されるウィンドウごとのページ読込の最大数を指定します。これは、リダイレクションや再ロードの無限ループが発生する場合に、ページ読込を停止するために使用できます。そのような無限ループによって、最終的にはアクションはタイムアウトしますが、これを早めに検出することで、アクセスしている Web サーバーに過剰な負荷がかからなくなります。アクションが停止すると、ページ読込の最大数に到達しているため、エラーが生成されます。

ウィンドウの最大ネスト数 (クラシック ブラウザのみ)

このプロパティは、ウィンドウが、内部で互いにネストできる最大数を指定します。この場合のウィンドウは、さまざまな意味を持つ可能性があります。フレームセットの中の各フレームはウィンドウであり、Max. Window Nesting プロパティは、読み込まれたページが内部で互いにいくつフレームを持てるか指定します。アクションが停止すると、ネストされるウィンドウの最大数に到達しているため、エラーが生成されます。[ロード エラーを無視] オプションをチェックすると、このステップ アクションは正しく完了して、ネストの最大数を超えるウィンドウを含まないページが出力されます。このフィールドを空で残す場合、ウィンドウのネスト レベルに制限はありません。

ページの変更

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。このデータ コンバータを設定する時は、アンパサンド エンコーディングをデコードする前に、ページの未処理の元のテキストに適用してください。このため、標準ブラウザの「ソースを表示」機能を使用して、このテキストを取得することを推奨します。データ コンバータ設定ウィンドウで、左下の入力エリアにテキストを貼り付けることができます。[テスト] ボタンをクリックすると、コンバータによるテキストへの希望するアクションの実行をテストできます。

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てください。

ページ エラー テスト (クラシック ブラウザのみ)

このプロパティは、ページのコンテンツに基づいて、Web サイトのエラーに対するカスタム テストを指定します。通常、何か不具合があった時、Web サイトはエラー コードを送信しますが、これがエラーの検出に十分ではない場合、このプロパティを使用することができます。

[全てのページで同じ] が選択されている場合、すべてのページでテストが実行されます。[URL による] を選択すると、特定の URL (のグループ) に対して個別のテストをセットアップできます。

([拒否されたページにパターンが一致] を選択して) エラー ページに一致するパターンを指定するか、([許可されたページにパターンが一致] を選択して) その他すべてのページに一致するパターンを指定できます。

エラー時にページを出力 (クラシック ブラウザのみ)

このプロパティは、Web サイトがエラー コードを送信する場合もページを出力するかどうか指定します。無効になっている場合、Web サイトのエラーによってアクションが失敗します。有効になっている場合、一定の Web サイト エラーは受け入れられてページが出力されますが、その他すべてのサーバー エラーは、アクションが失敗する原因となります。受け入れられる Web サイトのエラーは 403 Forbidden、404 Not Found、500 Internal Server Error です。

タイムアウトの場合はページを出力

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、そこまで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。

- このプロパティに対して [**FALSE**] がデフォルトとして設定される古いデフォルト ブラウザ (WebKit) のロボットが Kofax RPA の新しいバージョンで開かれるときは、ロボットのブラウザ設定の「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。
- 新しいクラシック 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 false に設定されます。

[URL フィルタリング] タブ

このタブは、広告フレームの読込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。

URL をフィルタリング

このプロパティは、特定の URL の読込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

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 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、読込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。ページ表示のそのウィンドウのタブの 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 **URL** のパターン

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。[ロギング] タブのオプションを使用すると、Design Studio の [ログ] ウィンドウで、ロボットの実行中に実行される JavaScript に関する情報を確認できます。このウィンドウを使用すると、実行される JavaScript や発生するエラーなどを理解できます。

注 このタブのプロパティの一部は、デフォルトとクラシック ブラウザ エンジンで異なります。

JavaScript の実行

このプロパティは、JavaScript を実行するべきかどうか指定します。

JavaScript エラーを無視 **(**クラシック ブラウザのみ**)**

このプロパティは、JavaScript 実行中に発生するエラーを無視するべきかどうかを指定します。多くの場合、実行の結果が希望通りであれば、そのようなエラーは安全に無視することができます。

アラート メッセージを無視

チェックされると、JavaScript メソッドの alert() によって生成されるアラート メッセージが無視され、チェックされない場合はエラーが生成されます。アラート メッセージは、通常、JavaScript によって作成され、正しく入力されていないフォームを送信しようとしているといった無効なアクションを、ブラウザのユーザーに警告します。

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

このプロパティは、無視されるアラート 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。

タイマー イベントを有効化 **(**クラシック ブラウザのみ**)**

このプロパティは、タイマー イベントを実行するかどうか指定します。タイマー イベントは、指定された期間後に発生するイベントであり、setTimeout() または setInterval() を使用する JavaScript によって設定するか、<meta> リダイレクションによって一定の秒数後のページのリダイレクトが指定されている時に設定できます。

タイマー イベントの最大待機時間 **(ms) (**クラシック ブラウザのみ**)**

このプロパティは、アクション実行の開始から実行されるタイマー イベントを待機する最大ミリ秒数を指定します。例えば、ページが 3000 ms で読み込まれ、いくつかタイマー イベントを設定し、このプロパティが 30000 ms に設定される場合、ページ読込実行後 27000 ms 以内に期限が切れるタイマー イベントのみ実行されます。以下の「タイマー イベントをリアルタイムで待機」プロパティに応じて、タイマー イベントの待機がリアルタイムで実行されるか、単に エミュレートされることに注意してください。

タイマー イベントをリアルタイムで待機 **(**クラシック ブラウザのみ**)**

このプロパティは、上記の「タイマー イベントの最大待機時間」プロパティによって指定される時間待機するか、単に待機を エミュレートしてトリガされるタイマー イベントのいずれかをすぐに実行するかを指定します。多くのタイマー イベントの場合、実際は、指定された期間、待機する必要はありません。ロボットはすぐに続行することができます。ただし、タイマー イベントの理由が Web サーバーが結果を処理するのを待機する必要があるといったものであれば、リアルタイムで待機する必要がある可能性があります。

キー プレス間の遅延 **(ms)**

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 アクションにのみ関係します。

CSS スタイル シートを使用 **(**クラシック ブラウザのみ**)**

このプロパティは、ロボットの実行中に CSS スタイルシートを読み込んで解析するかどうかを指定します。これは、ページの JavaScript が正しく動作するのに必要になる場合があります。一方で、スタイルシートの使用を無効にすると、ページ読込の実行速度を向上させることができます。このオプションが無効になっていても、ページは表示目的でスタイルシートを読み込むことができますが、サーバーでロボットが実行される時にこの読込は発生しません。

JavaScript 変更

「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の両方に対して適用されます。このデータ コンバータは、JavaScript を変更したり修正したりするのに便利です。例えば、これらは、JavaScript が VBScript によって定義されたと認識する変数を定義するのに使用することができます。この目的のために使用する最も一般的なデータ コンバータは、テキスト置き換えとパターン置き換えです。

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、左下の入力エリアにテキストを貼り付けることができます。[テスト] ボタンをクリックすると、コンバータによるテキストへの希望するアクションの実行をテストできます。

重要 このオプションは、読み込むページで JavaScript が実行される方法に影響し、「[ページ読込]」と「[ページ生成]」ステップに適用されます。

JavaScript ポリフィル

デフォルトの Kofax RPA ブラウザ (WebKit) では、ES5 および ES6 の JavaScript 機能の一部がサポートされていません。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。たとえば、ポリフィルは Silverlight プラグインを使用して Microsoft Internet Explorer 7 の HTML Canvas 要素の機能を複製したり、CSS の rem 単位のサポートを提供したりすることができます。

エラーの場合、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工ラーが解決されない場合が多くあります。例として以下の JavaScript の既知の問題リストを参照してください。

- 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);
```

プラグイン

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

- [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。

[Javascript イベント ハンドラ] タブ

このタブには、どの JavaScript イベント ハンドラが実行されるか決定するプロパティが含まれます。[ロギング] タブのオプションを使用して、Design Studio の [ログ] ウィンドウで、ロボットの実行中に実行されるイベント ハンドラに関する情報を取得できます。

クリック イベント ハンドラを有効化

このプロパティは、タグをクリックする時に、存在する場合は onclick イベント ハンドラが実行されるかどうかを指定します。

変更イベント ハンドラを有効化

このプロパティは、フォームの値を変更する時に、存在する場合は onchange イベント ハンドラが実行されるかどうかを指定します。

フォーム イベント ハンドラを有効化

このプロパティは、フォームを送信または再設定する時に、存在する場合は onsubmit および onreset イベント ハンドラをそれぞれ実行するかどうかを指定します。

ロード イベント ハンドラを有効化
このプロパティは、ページを読み込んだりアンロードしたり、あるいは画像を読み込む時に、存在する場合は onload および onunload イベント ハンドラを実行するかどうかを指定します。

マウス イベント ハンドラを有効化
このプロパティは、タグにマウス カーソルを合わせたり、クリックしたりする時に、存在する場合は onmouseover、onmouseenter、onmouseout、onmouseleave、onmousedown、onmouseup イベント ハンドラを実行するかどうかを指定します。

ドラッグ イベント ハンドラを有効化
このプロパティは、マウスがタグにドラッグされる時に、存在する場合は ondrag、ondragstart、ondragenter、ondragleave、ondragend、ondragover イベント ハンドラを実行するかどうかを指定します。

キー イベント ハンドラを有効化
このプロパティは、テキストを入力する時に、存在する場合は onkeydown、onkeypress、onkeyup イベント ハンドラを実行するかどうかを指定します。

フォーカス イベント ハンドラを有効化
このプロパティは、タグまたはドキュメントがフォーカスを取得したり失ったりする時に、存在する場合は onfocus、onfocusin、onfocusout、onblur、onactivate、onbeforeactivate、ondeactivate イベント ハンドラを実行するかどうか指定します。

キャプチャ イベント ハンドラを有効化
このプロパティは、タグまたはドキュメントがマウス キャプチャを失う時に、存在する場合は onlosecapture イベント ハンドラを実行するかどうか指定します。

状態変更イベント ハンドラを有効化
このプロパティは、タグまたは ActiveX オブジェクトの状態が変更された時に、存在する場合は onreadystatechange イベント ハンドラを実行するかどうか指定します。

エラー イベント ハンドラを有効化
このプロパティは、エラーが発生した時に、存在する場合は onerror イベント ハンドラを実行するかどうか指定します。

[ロギング] タブ

このタブには、ロボット実行中に実行される JavaScript のロギング レベルを決定するために使用されるプロパティが含まれます。ロギング情報は、Design Studio の [ログ] ウィンドウで取得することができます。どの JavaScript が実行されたか、どのエラーが発生したか、などを理解するために使用できます。

すべての **JavaScript** ソースを記録
このプロパティは、実行されるすべての JavaScript のソースをログに記録するかどうか指定します。JavaScript は、イベント ハンドラから呼び出されたりするまで、自身が実行されない関数を宣言する可能性があります。「JavaScript 変更」プロパティを使用すると、JavaScript で変更や修正を行うことができます。

JavaScript 実行トレースを記録
このプロパティは、JavaScript 実行の詳細なトレース内容をログに記録するかどうか指定します。このトレースには、呼び出されるすべての関数と、設定または取得されるすべてのプロパティが含まれます。
例えば

```
location.href =
"http://www.kofax.com"
```

が実行される時、トレースは、

```
GET location = [location]
```

の後に
SET [location].href = "http://www.kofax.com" を読み取ります。

トレースに関数ソースを含める
このプロパティは、JavaScript 実行のトレースに、実行される関数のソース コードを含めるかどうか指定します。

JavaScript イベント ハンドラを記録
このプロパティは、実行された JavaScript イベント ハンドラをログに記録するかどうか指定します。

タイマー イベントを記録
このプロパティは、実行されたタイマー イベントをログに記録するかどうか指定します。タイマー イベントは、指定された期間後に発生するイベントであり、setTimeout() または setInterval() を使用する JavaScript によって設定するか、<meta> リダイレクションによって一定の秒数後のページのリダイレクトが指定されている時に設定できます。

ロードを記録
このプロパティは、すべてのページとリソースの読込をログに記録するかどうか指定します。

XML HTTP 要求を記録
このプロパティは、送信される XML HTTP リクエストをログに記録するかどうか指定します。

絶対位置を記録
このプロパティは、JavaScript を使用して配置されるメニューなど、視覚要素の絶対位置をログに記録するかどうか指定します。

最大ログ エントリ数
このプロパティは、許容されるログ エントリの最大数を指定します。最小値は "1" です。許容されるより多くのログ エントリが存在する場合、最初のログ エントリが破棄され、[ログ] ウィンドウに表示されなくなります。

[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

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 **(**デフォルト**)**
Kofax RPA は、使用するすべてのサポートされる非 HTML コンテンツを読込ます。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 アクションを適用できます。[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

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。デフォルトでは、JSON は XML に変換されます。これによって、Design Studio の標準的な方法での処理が簡単になります。さらに、HTML に変換することもできます。人間が読みやすいのは XML より HTML ですが、自動抽出は少し難しくなります。

XML から **HTML** へ変換

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口ボットで主に使用されます。新しい口ボットは、通常、XML 構造で直接動作させる方が便利です。

注 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、**[**口ボット設定**]** > **[**デフォルト オプション: 設定**]** > **[**レガシー タブ**]** > **[**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

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口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 インターフェースを提供するため、新しい口ボットは、通常、Excel ドキュメントで直接動作させる方が便利です。

CSV

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口ボットで主に使用されます。新しい口ボットは、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 アクション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。

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>
```

ステップの設定

さまざまなプロパティを使用してステップを設定できます。以下にそのプロパティを示します。

[基本] タブ

このタブにはさまざまな基本ステップ プロパティが含まれています。

ステップ名
ここでは、ステップの名前を入力できます。

ステップのコメント
ここでは、オプションで、このステップに関するコメントを入力します。

[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。詳細については、タグ ファインダーまたは範囲ファインダーの説明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選択し、設定することができます。

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。

ウィンドウ

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矢印でマークされます。

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、別のページで作業したいときは常に、現在のウィンドウを変更する必要があります。

適切なステップ アクションを使用することで、以下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。

他のページ読込するページ読込する際 (例：<frameset>-tag を含むページ)、各ページは自動的に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 アクション (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ても、(フレーム セットの構造を維持しながら) フレームに毎回異なる名前を付けます。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 アクション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 要素でなければなりません。Design Studio では、フレームのリストは、フレーム ビューの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において [見つかったタグのウィンドウ] 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。

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、処理する工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。例えば、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 から始まる点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複数の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) タグが一致しなければならない「.*.***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多大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ときは注意する必要があります。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、一致するタグどうし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

例えば、タグ パスを "table" に、属性名プロパティを "align"、タグ属性プロパティを "align" に、 属性値プロパティをテキストを "center" にした「固定テキスト」、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を見つけます。

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点に注意してください。

範囲は、値セレクターを使用して複数の方法で指定することができます。

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、例えば名前付き範囲設定ステップまたはExcel 内ループステップ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っていると見なし、「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、現在表示されているシートを参照したい場合は、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」(例えば、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複数の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 要素から情報セットを抽出したいときは、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` と同様のものが得られます。

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)。
\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。

特殊記号	説明
[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) を付けてエ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" に一致するが "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" に一致するが "goodbye" には一致しません。

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。

特殊記号	説明
+	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。
(.+?)(.*)	"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" と一致し、"bc" をエクスプレッションでは $1 として参照できます。 "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。 "abc=abc" と一致しますが、"abc=ab" とは一致しません。

パターン	一致内容
a(?:bc)	
(.*)=\1	

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。例えば、エクスプレッション "The author of the book " + Book.title + " is " + Book.author + "." からは、Book 変数の各属性 title および author に、テキスト "Gone with the Wind" および "Margaret Mitchell" がそれぞれ含まれる場合は、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。例えば、属性 Book.price に本の価格が含まれる場合、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

エクスプレッションの場所に応じて、入力テキストがエクスプレッション内で使用できるようになります。この入力テキストをエクスプレッション内で参照するには、INPUT 表記を使用します。例えば、入力テキストが「The time is」で、現在の時間を追加したい場合、次のエクスプレッションを使用できます。

INPUT + time(now())

エクスプレッション タイプ

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"
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.<<

エクスプレッションタイプ	表記	説明	例
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
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、expr1 の値を使用します。そうでない場合は、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
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。	Robot.executionErrors

注 テキスト定数に対する >>text<< 表記は、HTML など、引用符文字を多く含む長いテキストを指定するのに便利です。>>text<< 表記を使用すると、前後に >> および << のシンボルを配置するだけで、テキストをそのまま使えます。"text" 表記を使用する場合は、テキスト内のすべての引用符文字を、2 つの連続する引用符文字に置き換える必要があります。

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、そうでない場合は false を返します。
excelColStringToNumberFunction	Excel スプレッドシートの文字列を数に変換します。
excelNumberToColStringFunction	数を Excel スプレッドシート内の文字列に変換します。
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。
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。

関数	説明
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、そうでない場合は 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)	テキストの両端からすべてのスペースを除去します。
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。

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 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。
- ロボットがデバッグ モードで実行される場合、レポートは [エラー レポート] タブに追加されます。
- 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によって実行され、かつ RoboServer で実行される場合、レポートは、以下で説明するように、API を介して呼び出し元に送信されます。

  ```
  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) 。

注 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 から、または API を介して) RoboServer で実行される場合、インシデントはその RoboServer のクラスター設定で定義されている場所にログ記録されます。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。

注 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。

Then

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。

次の代替手段を試行

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、[次の代替手段を試行] 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

ループの内部で、つまり (Repeat-Next ループを除く) 何らかの繰り返しステップの後で、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) 何らかの繰り返しステップの後で、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。

- 次の代替手段を試行
- 次のイテレーション

- 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] 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点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] 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しなければなりません。そうでない場合、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 try-catch 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、Try-Catch 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 [次の代替手段を試行] 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「次の代替手段を試行」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「次の代替手段を試行」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」は不正にはなりませんが、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、遅延の影響で使用しづらいため「次の代替手段を試行」の使用を推奨します。8.0 よりも前のバージョンの Design Studio で作られたロボットを開くと、「後方に送る」の代わりに「次の代替手段を試行」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できる場合には必ず) 。

URL としての POST リクエスト

標準的な URL は、GET リクエストを表すことはできますが、HTTP POST リクエストを表すことはできません。POST リクエストを URL として表すため、Kofax RPA は、次の文法による独自の特別な URL 形式を使用します。

post://<HTTP プロトコルを除外した URL>??<エンコードされた POST パラメータ>

HTTPS プロトコルの場合は以下の通りです。

posts://<HTTPS プロトコルを除外した URL>??<エンコードされた POST パラメータ>

プロトコルのない相対 URL の場合は以下の通りです。

postx://<プロトコルのない URL>??<エンコードされた POST パラメータ>

例
URL に対する POST リクエスト
http://www.abc.com
これにエンコードされた POST パラメータを付けます。
x=y&v=z
これにより次の URL として表されます。
post://www.abc.com??x=y&v=z

ライブラリ プロトコル

Kofax RPA の非標準ライブラリ プロトコルを使用して、ロボットがロボット ライブラリ内で所属する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、値を指定する以下の方法の 1 つを選択します (これらすべての方法がどこでも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。)

URL セレクター

URL セレクターを使用して、必要に応じて異なる方法で URL を指定します。

URL セレクターの上部のドロップダウン メニューを使用して、URL を指定する以下の方法の 1 つを選択します。

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 を読み取るように指定します。

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。

キーボード ヘルプ

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
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

説明	キーボード ショートカット
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

ブラウザ トレーサ

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 トラフィックおよび 9.3 以前の口ボットでデフォルトだったクラシック ブラウザで書かれた口ボット用の JavaScript 実行をトレースすることができます。ブラウザ トレーサは、口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 記録を利用した記録中は大量のデータが収集されるため、動作が通常よりずっと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。

注 保留中の読込は青で表示されます。

URL のブロック

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

JavaScript トレーシングは、9.3 以前の口ボットでデフォルトだったクラシック ブラウザを使用している口ボットに対してのみ利用できます。デフォルト ブラウザを使用している口ボットの場合は、[JavaScript] タブを利用できません。

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 コードが表示されます。トレース項目が選択されると、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。そうするには、[トレース] メニューで [JavaScript トレーシングの有効化] 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読込

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 トレースの両方および JavaScript トレースと HTTP トレースの両方が含まれます。トレース セッションが大きいため、後で詳しく調べたい場合、または誰かにトレース セッションをメールしたい場合は、トレース セッションを保存すると便利です。

注 Design Studio から送信されるバグ レポートには、存在すれば、現在のトレース セッションが自動的に含まれます。

注 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。

ツールバーの ボタンをクリックしてファイル ダイアログを開き、すべての本文レスポンスを保存するディレクトリを選択します。

1 つの本文レスポンスのみを保存するには、レスポンスを右クリックし、コンテキスト メニューの [レスポンス データに名前を付けて保存] をクリックします。

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
```

変数の検証エラー

検証エラーは、変更、名前変更、または削除されるタイプを使用する変数で発生します。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。

変数によって使用されているタイプが不足している場合、「タイプが見つかりませんでした」を通知する検証エラーが表示されます。不足しているタイプの名前でタイプを作成するか、[変数の設定] ウィンドウを開いて値に別のタイプを選択して エラーを解決します。

変数によって使用されるタイプによる設定の問題は、「無効なタイプ」が存在することを通知する変数上の検証エラーに発展します。この問題は、タイプによる問題を修正するか、[変数の設定] ウィンドウの変数に別のタイプを選択することで解決します。

変数の初期/値判定の一部が「そのタイプと互換性がない」ことを通知する検証エラーは、捉えにくいもので、前述のエラーほどすぐにわかるものではありません。このエラーは、初期/値判定に割り当てられている初期/判定値が変数にいくつかあり、1 つ以上の値が属性に割り当てられないような変数のタイプに変更された場合に起こります。これは、属性がタイプから削除されたり、属性タイプが変更されたりすることで、以前の割り当て値が現在互換性を持たなくなった場合に発生します。たとえば、初期値が Short Text の属性タイプを持つ属性に割り当てられても、変数タイプが変更されることで、属性の属性タイプが "Boolean" になる場合は、以前の値が割り当てできなくなります。この問題を修正するには、[変数の設定] ウィンドウを開きます。すると、割り当てられていない値が表示され、他の属性にコピーできるようになります。割り当てられていない値を消去して、検証エラーを解決するには、[OK] をクリックして [変数の設定] ウィンドウを終了します。値が破棄されたことを通知するダイアログが表示されます。再び [OK] をクリックすると、検証エラーが解決されます。

サポートされている Excel の機能

このトピックでは、Excel の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両方を説明します。

数式のサポート

サポートされる数式の詳細については、https://poi.apache.org を参照してください。

機能

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、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、280 以上の評価可能な関数
- 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例に示すような条件付きフォント色：`#.##0;[Red]-#.##0`
- 日付書式設定

サポートされていない機能

- 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

POI の関数

サポートされている関数

ABS、ACOS、ACOSH、ADDRESS、AND、ASIN、ASINH、ATAN、ATAN2、ATANH、AVEDEV、AVERAGE、BIN2DEC、CEILING、CHAR、CHOOSE、CLEAN、CODE、COLUMN、COLUMNS、COMBIN、COMPRATE、REPLACE、REPT、RIGHT、ROMAN、ROUND、ROUNDDOWN、ROUNDUP、ROW、ROWS、SEARCH、

サポートされていない関数

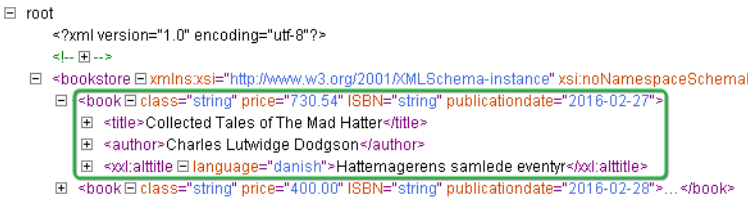
ACCRINT、ACCRINTM、AMORDEGRC、AMORLINC、AREAS、ASC、AVERAGEA、AVERAGEIF、AVERAGEIFS、

XML データ マッパー

Kofax RPA バージョン 9.1 およびそれ以降には、XML データ マッパーが含まれています。これを使用することで、XML ドキュメントによって指定されたデータ記録を適切な構造の変数に便利にマッピングすることが可能になります。

データ マッピングの作成

1. ソース ビューで、関連の 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の両方が < および > で閉じらている XML のような形式で表示されます。

注 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、マッピングしたい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。

   注 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。
   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。

   注 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、ロ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。

   注 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既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。これは、グループ ステップは実際のデータ マッピングでなければならないことが特に重要です。

   - 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、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済み URL 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。

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 Web サイトのロボットは、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 は、基本、ダイジェスト、NTLM、ネゴシエート プロトコルをサポートします。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。

注 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 Installation Guide』(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、[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] オプションで実行するか、または spn.txt ファイルを使用して実行できます。

[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。[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。

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

Kofax RPA インストール ディレクトリの bin ディレクトリに spn.txt を作成します。例: C:\Program Files\Kofax RPA 11.0.0 x64\bin。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

パラメータ	説明	例
<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、さまざまなポート番号を使用して同一のホスト名で実行され、さまざまなアプリケーション プール ID を使用して複数の Web サイトがある環境で使用されている場合は、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 log4j.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：log4j.logger.webkit=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、クラスタ ログトピックの「log4J」セクション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 (x86)\Kofax RPA DesktopAutomation 11.0.0
  x32\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-x32\[ビルド番号]\lib`*。
  例：
  `C:\Program Files\Kofax RPA 11.0.0 x64\nativelib\hub\windows-x32\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_11.0.0_x64/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。

画像の前処理

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。

注 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。デフォルトで、Kofax RPA には OCR の言語として英語がインストールされます。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。

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

`sudo apt-get install` コマンドを使用して以下のライブラリをインストールします。

- 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。

注 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。

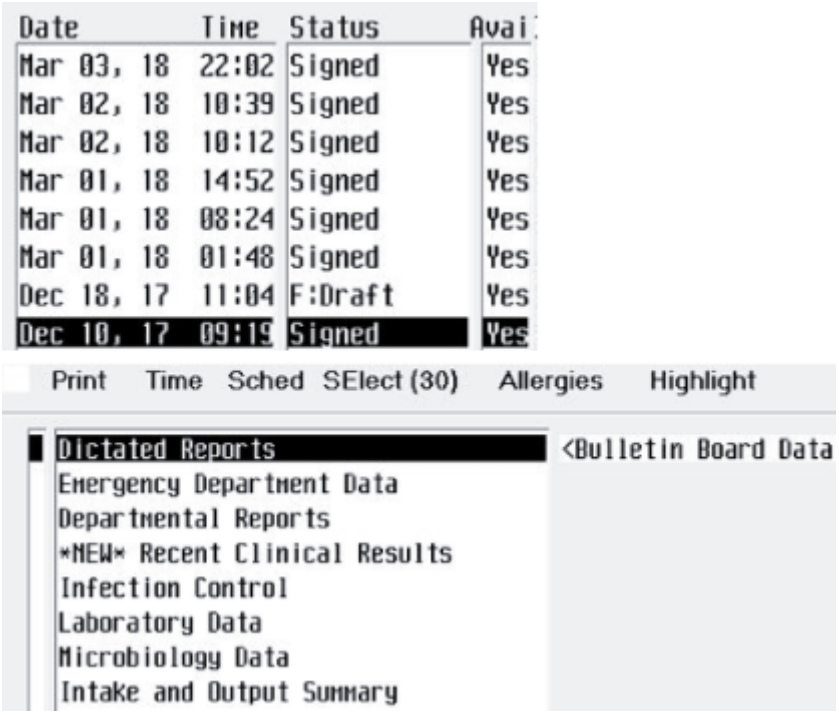
```
Moving /tmp/tmp.OtEqYbS3qV/eng/eng.traineddata to ../output
Completed training for language 'eng'
```

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。「オートメーション デバイスの設定」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

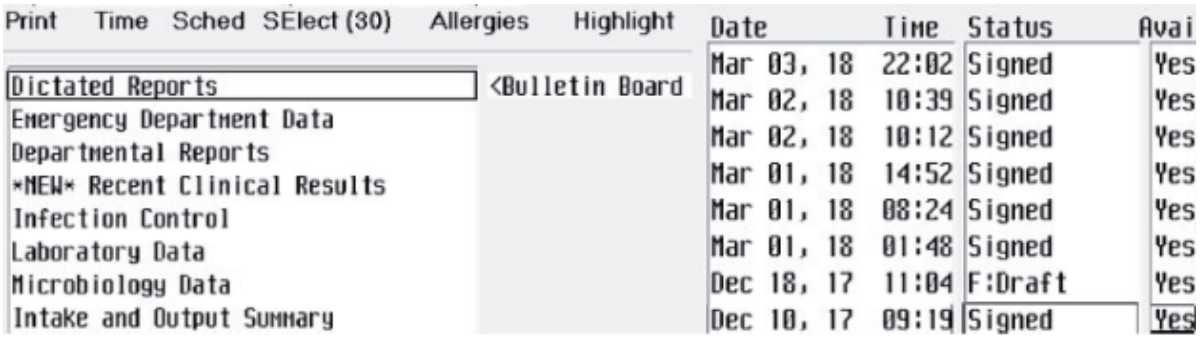
このモードは、UI で TTF フォント ファイルが使用されていない場合 (つまり自動モードを適用できない場合) で、ロボットの認識対象となる文字がすべてアルファベットの UI スクリーン ショットが多数ある場合に選択します。トレーニング画像ファイルがスクリプトによって自動的に作成される自動モードとは異なり、トレーニング画像は手動で作成する必要があります。このファイルの作成には、時間と労力が必要になります。

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。

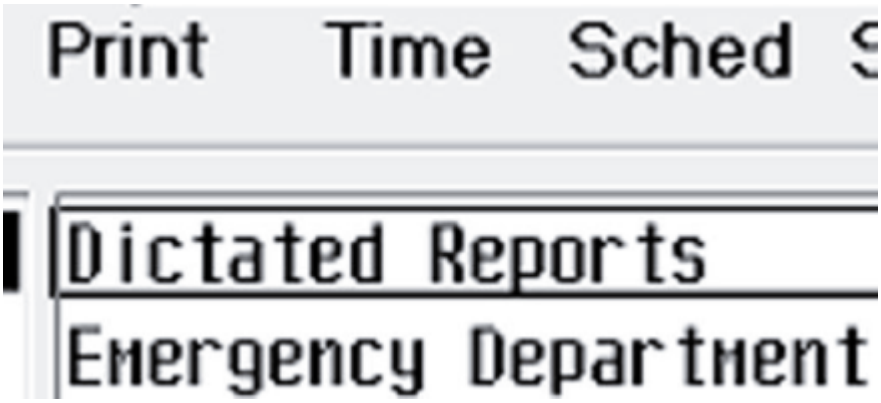


1. 使用する完全な文字セットを決定します。トレーニング ファイルの作成時には、各文字に対するサンプルの最小数は 5 であることに注意してください。最も頻繁に使用される文字の場合、追加でサンプルを含めます。
2. 単一の TIFF ファイルに、トレーニングで使用する UI スクリーン ショットの全部分を貼り付けます。この操作では使用する画像エディターは任意です。この例では、ターゲットのアルファベットを 10 ~ 15 の英字に限定します。実際の操作では、すべての文字のサンプルを用意します。

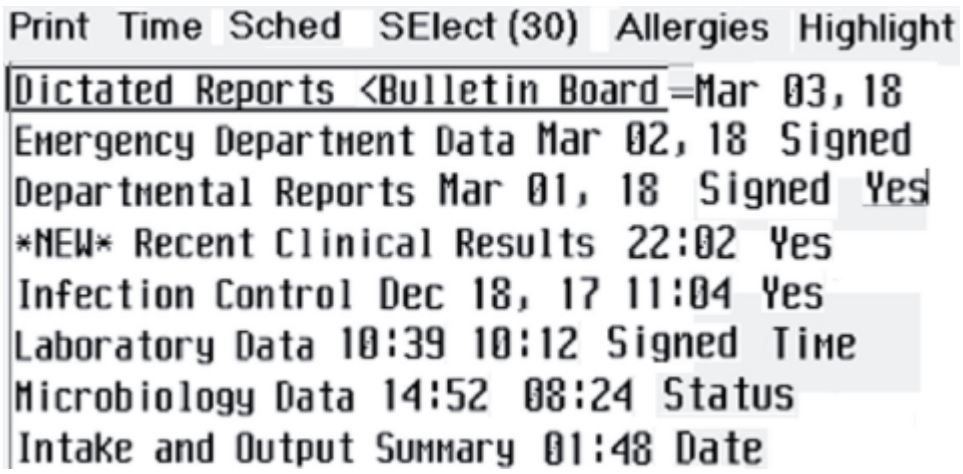
3. 反転色の領域を選択して、色を反転させて通常色にします。



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)。

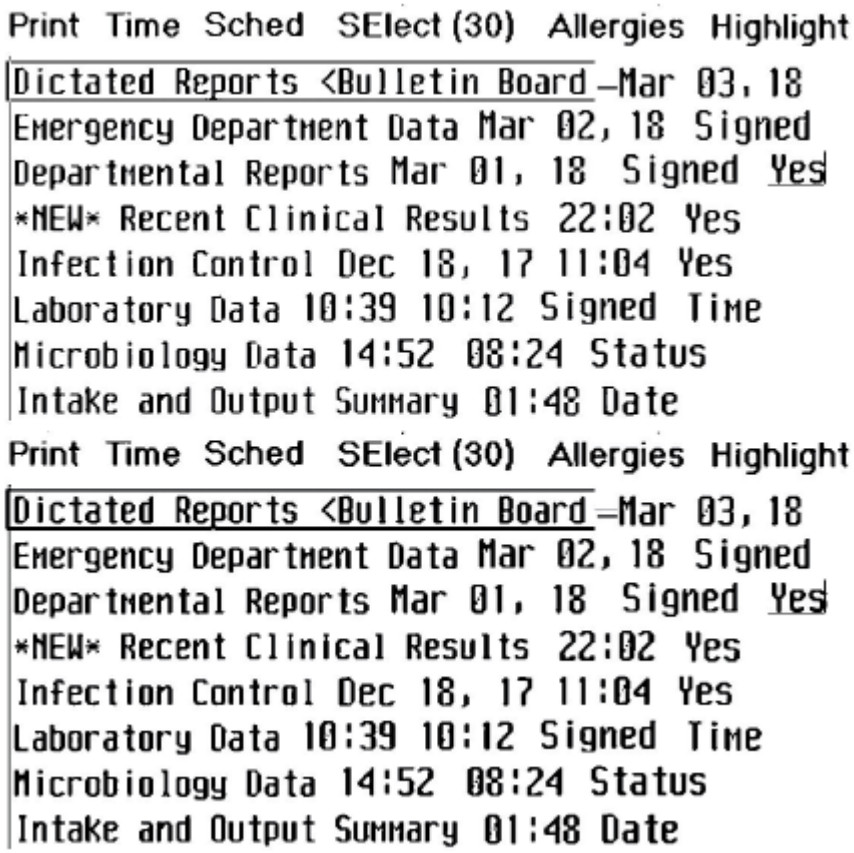


5. 単語を調節して、テキスト領域間に大きな空白がなくてもテキスト行を容易に検出できるようにします。以下の例のように、余分と思われるテキストを除去します (ページに合わせて縮小しています)。

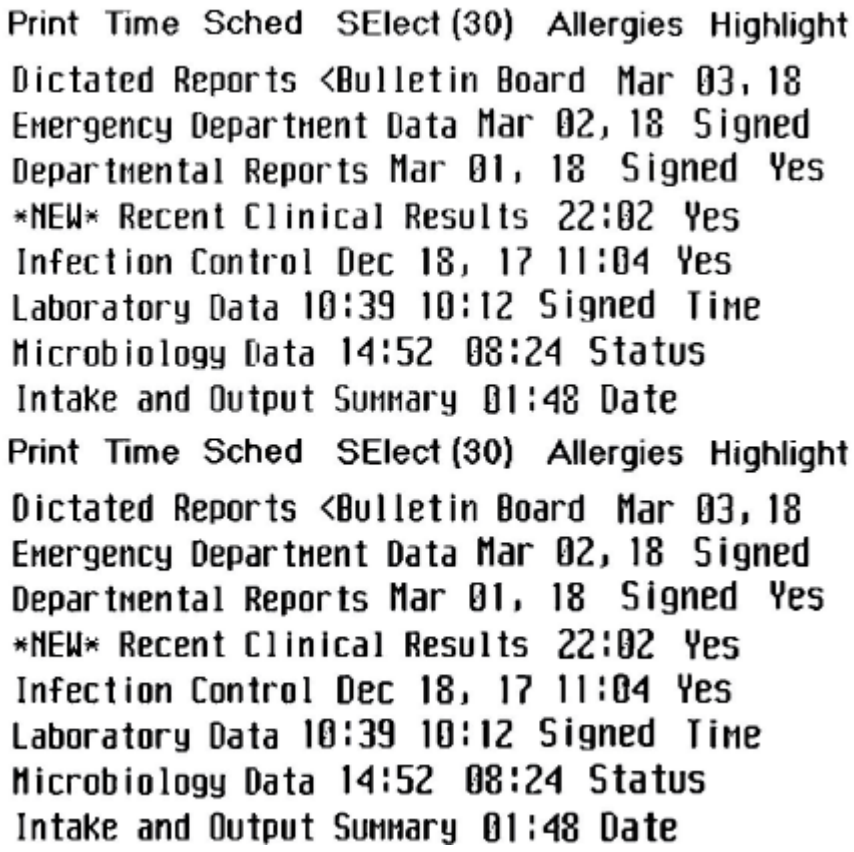


6. 画像をグレースケールに変換し、テキストの質が最も良くなるしきい値の色効果を適用します。適切なしきい値を選択することは難しい可能性があります。複数の異なるしきい値を適用することを

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` フォルダーに移動し、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。「オートメーション デバイスの設定」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。

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。

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、「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。	
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。	
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。	
IntersectionObserverEntry IntersectionObserver	祖先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。祖先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なければならず、値は任意の値でもかまいません。	

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などの応答を返します。

RoboServer の詳細説明は、『Kofax RPA Administrator's Guide』(Kofax RPA 管理者ガイド) を参照してください。

RoboServer を開始

RoboServer を開始するには、次のアクションを使用します。

- RoboServer プログラム アイコン (または、Management Console と RoboServer の両方を開始する Management Console の開始プログラム アイコン) をクリックします。
- RoboServer をコマンド ラインから開始します。
- サーバーをサービスとして実行します。RoboServer をサービスとして実行する際の詳細については、「サーバーの自動開始」を参照してください。

RoboServer をコマンド ラインから呼び出すには、コマンド プロンプト ウィンドウを開き、`Kofax RPA` インストール フォルダの `bin` フォルダに移動し、次のように入力します。

```
RoboServer
```

必要なパラメータがすべて roboserver.settings 構成ファイルで指定されると、RoboServer が開始します。

必要なパラメータのいずれかがない場合、RoboServer ではエラーが指定され、使用ヘルプと利用可能なパラメータが表示されます。

RoboServer パラメータ

RoboServer を開始するためのコマンド ラインには、次のパラメータが含まれています。

```
RoboServer [-cl <arg>] [-cpuThreads <arg>] [-h]
       [-maxConcurrentRobots <arg>] [-maxQueuedRobots <arg>] [-MC] [-mcUrl
       <arg>] [-p <arg>] [-pauseAfterStartupError] [-s <arg>] [-sslPort
       <arg>] [-v] [-V]
```

RoboServer を開始する方法に関係なく、次の表のパラメータが受け入れられます。すべてのパラメータは RoboServer 設定ユーティリティで編集できます。詳細については、RoboServer 設定を参照してください。

パラメータ	説明
`-cpuThreads <arg>`	使用する CPU スレッドの数を指定します。

パラメータ	説明
-h --help	ヘルプを表示します。
-maxConcurrentRobots <arg>	同時に実行される口ボットの最大数を指定します。
-maxQueuedRobots <arg>	キューの口ボットの最大数を指定します。
-mcUrl <arg>	次の形式で登録する Management Console を指定します。 http[s]://<username>:<password>@<hostname>:<port number> 例：-mcUrl http://admin:password@localhost:8080/ManagementConsole
-pauseAfterStartupError	
-v | -verbose	このオプションのパラメータによって RoboServer でステータスとランタイム イベントが出力されます。
-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。
-MC --scheduler	このオプションのパラメータによって、Management Console が RoboServer の一部として開始されます。Management Console は、設定アプリケーションで構成された埋め込み Web サーバーで実行されます。
-cl --cluster <arg>	このオプションのパラメータによって、RoboServer が Management Console 上の指定されたクラスタに自動的に登録されます。次の例では、RoboServer が *Production* クラスタに自己登録します。 例：-cl Production 例：-mcUrl http://admin:password@localhost:8080/ManagementConsole -cl Production
-s <service-name:service-parameter> | -service <service-name:service-parameter>	このパラメータによって、RoboServer で開始する必要がある RQL または JMX サービスが指定されます。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
-p <port-number> | -port <port-number>	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。
利用可能なサービス	

パラメータ	説明
`-service socket:<portNumber>`	`<portNumber>`: リッスンするソケット サービスのポート番号。
`-service ssl:<portNumber>`	`<portNumber>`: リッスンするソケット サービスのポート番号。
`-service jmx:<jmx_port_Number>,<jmx_rmi_url>`	`<jmx_port_Number>`: リッスンする 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`

パスワードを設定するには、RoboServer 設定ユーティリティ、または ConfigureRS ツールを使用する必要があります。詳細については、RoboServer 設定および RoboServer 設定 - ヘッドレス モードを参照してください。

サーバーの自動開始

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 と Management Console (ライセンス サーバー) を指します。実際には、この 2 つのコンポーネントは RoboServer の開始時に指定される引数に基づいて RoboServer から提供されます。

RoboServer パラメータのトピックには、RoboServer プログラム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 プログラムで口ボットを実行できるようにするには、`-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)。

次のトピックでは、RoboServer を Windows と Linux で自動的に開始する方法について詳述します。

RoboServer のシャットダウン

RoboServer をシャットダウンするには、コマンド ライン ツール ShutDownRoboServer を使用します。ShutDownRoboServer を引数なしで実行して、サーバーをシャットダウンする方法、特にサーバー上で現在実行中の口ボットの処理方法についてさまざまなオプションを確認します。

RoboServer 設定

RoboServer の設定は、RoboServer 設定アプリケーションで行えます。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、『Administrator's Guide』(管理者ガイド)の「RoboServer Configuration」(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すると、次の各項目を設定できます。

- 全般：ソケット サービス オプション。JMS サービス オプション、Management Console 接続オプション、および詳細オプションを有効にして設定します。
- 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 v プロジェクトの場所。
- JMX Server: JMX サーバー。
- Management Console内蔵の Management Console の設定。

設定を変更した場合は、OK をクリックして新しい設定を保存します。その後、実行中の RoboServer があれば再起動して、変更を反映させます。

重要 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 以降では、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、Management Console の URL およびクレデンシャルとクラスタの名前を、(コマンド ラインで、または RoboServer 設定アプリケーションを使用して) RoboServer を起動するとき指定する必要があります。

Kofax RPA にはいくつかのコマンド行ツールが含まれ、バッチ モードでの設定の変更に役立ちます。例えば、指定した権限を持つ複数のユーザーを作成できます。ヘッドレス モード内の「RoboServer の設定」を参照してください。

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、管理者ガイドの「Change the RAM Allocation」(RAM 割り当ての変更) を参照してください。

RoboServer 設定 - ヘッドレス モード

Kofax RPA には RoboServer をコマンド ラインで設定するためのユーティリティがいくつか同梱されています。ユーティリティの場所は、Kofax RPA インストール フォルダの `bin` サブフォルダの中です。設定ファイルはユーザー依存で、ユーザー フォルダに保存されます。詳細については、『Kofax RPA Installation Guide』(Kofax RPA インストール ガイド) の「Important Folders」(重要なフォルダ) を参照してください。

- ConfigureRS: JMX パスワードと Management Console パスワードを RoboServer 設定ファイル (roboserver.settings) に設定します。
- ConfigureMC: Management Console 管理者と、証明書パスワードを mc.settings ファイルに設定します。
- ConfigureRSUser: ユーザーの追加や除去を行い、rsusers.xml ファイル内のユーザー クレデンシャルをアップデートします。このファイル内の情報を使用して、API 要求を認証します。

使用上のヘルプについては、 -h オプションでユーティリティを実行します。

Management Console (RoboServer の登録先) に接続を設定するには、次のコマンドをタイプします。

```
ConfigureRS -mcUrl http://admin:password@localhost:8080/ManagementConsole
```

すべての権限と Password1 というパスワードを持つ user1 というユーザーを作成するには、次のコマンドをタイプします。

```
ConfigureRSUser user1 Password1 -a
```

API 要求の認証を有効にするには、rsusers.xml を開いて、有効になっている属性を true にする必要があります。以下に例を示します。

Sample rsusers.xml configuration file

```
<?xml version="1.0" encoding="UTF-8"?>

 <userConfiguration enabled="true">
   <users>
     <user username="user1"
 password_hash="20c7628c31534b8718a1da00435505e4262e3f4dc305">
       <startRobot/>
       <stopRobot/>
       <shutdownRoboServer/>
     </user>
   </users>
 </userConfiguration>
```

Sample roboserver.settings configuration file

```
# Settings file for RoboServer. Some configurations contains encrypted passwords and
 should not be edited by hand,
           these should be modified using dedicated commandline tools.

# The directory of use on RoboServer when the API used the DefaultRoboLibrary. On
 windows \ must be escaped like this
c:\\\\users\\AppData\\Local\\Kofax RPA\\...
defaultProject = /home/TestUser/Kofax RPA/trunk

# Should RoboServer be allowed to access the fileSystem, or call commands/scripts.
 Values: true/false
sec_allow_file_system_access = false

# Will RoboServer accept JDBC drivers sent from the Management Console. Values: true/
false
sec_accept_jdbc_drivers = true

# Should RoboServer log all loaded URLs to the log4j audit log. Values true/false
sec_log_http_traffic = false

# If enabled RoboServer will check credentials for API requests. Values: true/false
sec_authenticate_api_requests = false

# If enabled RoboServer generate an error when accessing a https site without a valid
 certificate. Values: true/false
cert_verify_https_certificates = false

# If enabled, RoboServer will only allow SSL connections from trusted client. Values
 true/false
cert_verify_api_certificates = false

# Configures if the the JMX service should be enabled
enable_jmx = false

# The port number for the JMX service to listen on.
jmx_port_Number = 50100

# If enabled, input for robots is exposed through JMX. Values: true/false
jmx_show_inputs = true

# Heartbeat notification interval, in seconds
jmx_heartbeat_interval = 0

# Configure if JMX should use RMI
enable_jmx_rmi = false
```

```
# Optional RMI host and port for the JMX service. Use if you need to connect through a
 firewall. Example: example.com:51001
jmx_rmi_url =

# Enables authentication for JMX requests. Values: true/false
jmx_enable_authentication = true

# The user-name used for JMX authentication
jmx_username =

# The password used for JMX authentication. This should be created using the
 ConfigureRS command line tool.
jmx_password =

# Configures if the socket service should be enabled
enable_socket_service = false

# Configures which port the RoboServer should be listening on
port = 50000

# Configures if the ssl socket service should be enabled
enable_ssl_socket_service = false

# Configures which ssl port the RoboServer should be listening on
ssl_port = 50001

# Configures if the JMS service should be enabled
enable_jms_service = false

# Configures which id the RoboServer should have when running JMS
jms_id = 1

# Configures the URL of the message broker when running JMS
broker_url =

# Configures if the RoboServer should register to a Management Console
enable_mc_registration = false

# Specify which Management Console to register to formatted as: http[s]://
<hostname>:<port number>
mc_URL =

# The user name to use for authentication to the Management Console
mc_username =

# The password to use for authentication to the Management Console
mc_password =

# Specifies which cluster the RoboServer should be registered to
cluster =

# Causes RoboServer to output status and runtime events
verbose = false
```

Sample mc.settings configuration file

```
# Settings file for Management Console. Passwords should not be edited by hand, but
 using the 'ConfigureMC' command line utility.


 # Should the MC web-server start a HTTP listener. Values true/false
 mc_http = true

 # Configures the port of the http listener.
```

```
mc_http_port = 50080

# Should the MC web-server start a HTTPs listener. Values true/false
mc_https = false

# Configures the port of the HTTPS listener.
mc_https_port = 50443

# Password for the certificate used by the HTTPs listener. This should be created
using the ConfigureMC command line tool.
mc_https_cert_password = 3W2MTrL/b2k=

# Enables MC internal user management, to support multi user scenarios. Values: true/
false
mc_enable_usermanagement = true

# The user-name of the MC super user.
mc_admin_user =admin

# The passwordHash of the MC super user. This is a salted SHA-256 hash.
mc_admin_password
=7800451255702ef8ae5f5fa0337833059d80b81d5af5872bdeafed230bab479896b6df4f63b25a24

# Configures which hosts are allowed to upload JDBC jar files to MC. Values: NONE,
LOCALHOST, ANY_HOST
mc_allow_jdbc_upload = LOCALHOST
```

Management Console

Management Console は Kofax RPA インストールを監視して管理する Web ベースのアプリケーションであり、Management Console を使用して次の内容を実行できます。

- 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 ロールおよび権限の管理、およびソリューションの集中管理。
- リポジトリからのロボットの高度なスケジューリング。
- 抽出データのブラウジング、および MS Excel 形式へのエクスポート。
- プロダクション結果およびエラーの詳細ログへのアクセス。
- ロボットを Design Studio からリポジトリにワンクリックで展開します。

詳細は、Management Console ドキュメントか Management Console チュートリアルを参照してください。

その他のトピック

このセクションでは、Management Console で使用されるその他の重要な概念の一部に関するリファレンス ヘルプを説明します。

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、ロボットをいつ実行すべきかを記述します。サブフィールドは 1 または複数のスペースで区切られ、以下を表現します。

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。例えば、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" などにも置き換えできます。

ワイルドカード ("*" 文字) も使用できます。その意味は、「このフィールドで可能なすべての値」です。例えば、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 は当然、「毎日」の意味になります。

各フィールドに対する一連の有効値は以下のとりです。

- 秒：0 ~ 59 の数。
- 分：0 ~ 59 の数。
- 時：0 ~ 23 の数。
- 日：1 ~ 31 の数 (ただし、その月の日数に注意してください)。
- 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。例えば、"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。"3/20" を分フィールドに入力すると、「1 時間のうちで 3 分から開始して 20 分毎」の意味になります - 言い換えれば、"3,23,43" と指定するのと同じです。
?	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。「指定値なし」の意味です。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。 例えば日フィールドでは、"L" の意味は「月の最終日」で、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。 これを曜日フィールドで使用すると、単純に "7" または "SAT" の意味ですが、これを曜日フィールドで、別の値の後に使用すると、その意味は「月の最終 x 曜日」になります。例えば、"6L" または "FRIL" の意味はどちらも「月の最終金曜日」になります。 "L" 文字を使用するときは、リストや値の範囲を指定しないでください。混乱しやすい結果になります。
#	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。例えば、"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。一例を挙げると、日フィールドの値に "15W" を指定すると、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

フィルタリング

ほとんどのタブで [リポジトリ] の下にある [フィルタ] フィールドを使用することで、アイテムのリストをフィルタして、名前とパスで表示できます。[ロボット] タブと [トリガー マッピング] タブには、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ています。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。

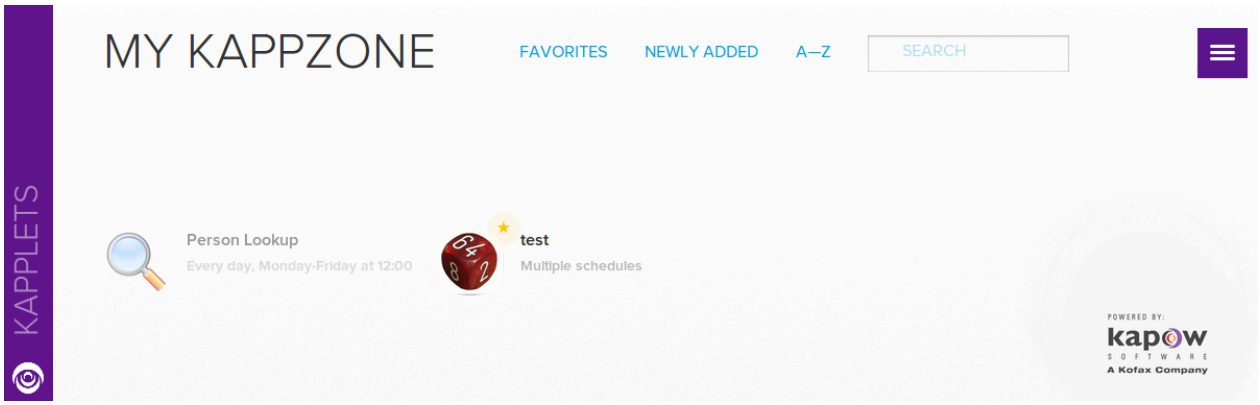
[フィルタ] フィールドを使用してリストをフィルタリングするときは、次のルールに注意してください。

- ? 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、項目の完全なパス、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` の両方にマッチします。

Kofax RPA Classic Kapplet

Kofax RPA Classic Kapplet は、使いやすいユーザー インターフェイスをロボットに提供します。Kapplet 管理者は、ユーザーが 1 つ以上のロボットを実行できるようにすることが可能です。これらのユーザーは、提供された Kapplet を通じてロボットを操作するときにロボットについて知る必要は一切ありません。Kapplet をカスタマイズして、用語をエンドユーザーが使用しているものに合わせることができます。アイコンや説明を追加することもできます。

Kapplet は、Kapplet 管理者が Kapplet Studio で作成およびメンテナンスします。Kapplet を使用する準備ができると、Kapplets を各ユーザーの My KappZone にインストールできる KappZone から、すべての Kapplet ユーザーが Kapplet を使用できるようになります。Kapplet ユーザーはインストールした Kapplets の独自のリストを My KappZone に保持したり、ブラウザで Kapplets をブックマークしたりします。



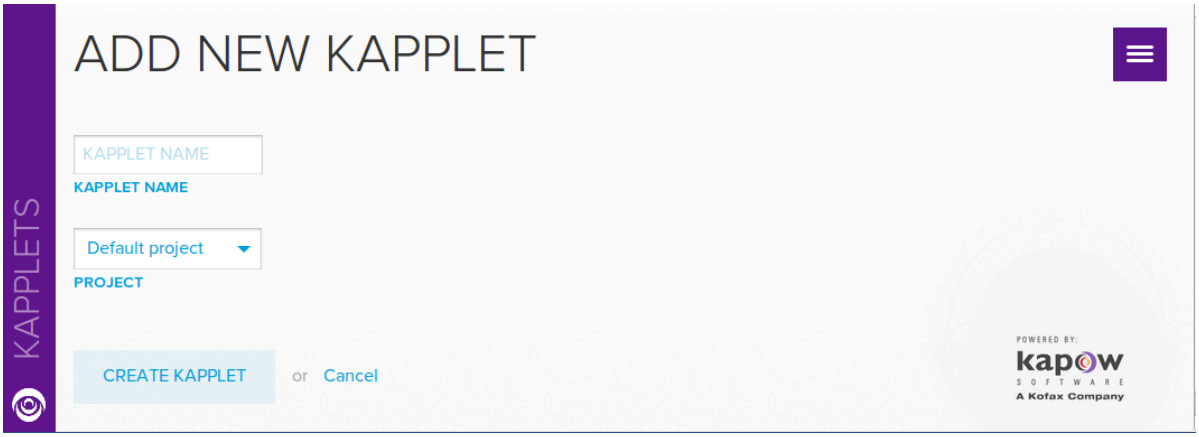
Kapplets は、Management Console のユーザー/ロール管理に基づいています。スタンドアロンのアプリケーション サーバーにデプロイされた Management Console で、Kapplet 管理者は LDAP を使用し、どのユーザーが、どのプロジェクトで Kapplet で公開されている ロボットにアクセスできるかを管理します。

Kapplets の作成とメンテナンス

このセクションのトピックでは、Kapplet の作成と Kapplet Studio の使用に関する情報を提供します。

Kapplets の作成

1. KappZone などの Kapplet ユーザー領域で、[新しい **Kapplet** を追加] をクリックします。
   または、メイン ナビゲーション メニューで、[新しい **Kapplet**] をクリックします。
   [新規 Kapplet を追加] ウィンドウが表示されます。
2. Kapplet の名前を入力します。
3. Kapplet とプロジェクトを関連付けます。



Kapplet Studio の使用

Kapplet Studio を使用して、名前、コメント、アイコンなどの一般的な Kapplet 情報を入力します。

注 ID セクションは、ロボットが Kapplet に追加されていない場合にのみ表示されます。ロボットが追加されている場合は、ページ セクションが表示されます。

1. 新規作成された Kapplet を開いて、ID セクションを表示します。
2. ID セクションで、Kapplet の名前を入力します。
   必要に応じて、追加情報を入力します。
3. ページをクリックします。
   ページ セクションが表示されます。
4. Kapplet を設定します。
   左側に Kapplet ページ定義の現在のコレクションが、クリック可能なタイトル付きのページ アイコンの形式で列挙されます。これらの各アイコンは、作成中の Kapplet の動作するページ/部分を表し

ています。一部のページは動作するすべての Kapplet に対して必須ですが、実際の機能によって異なるページもあります。

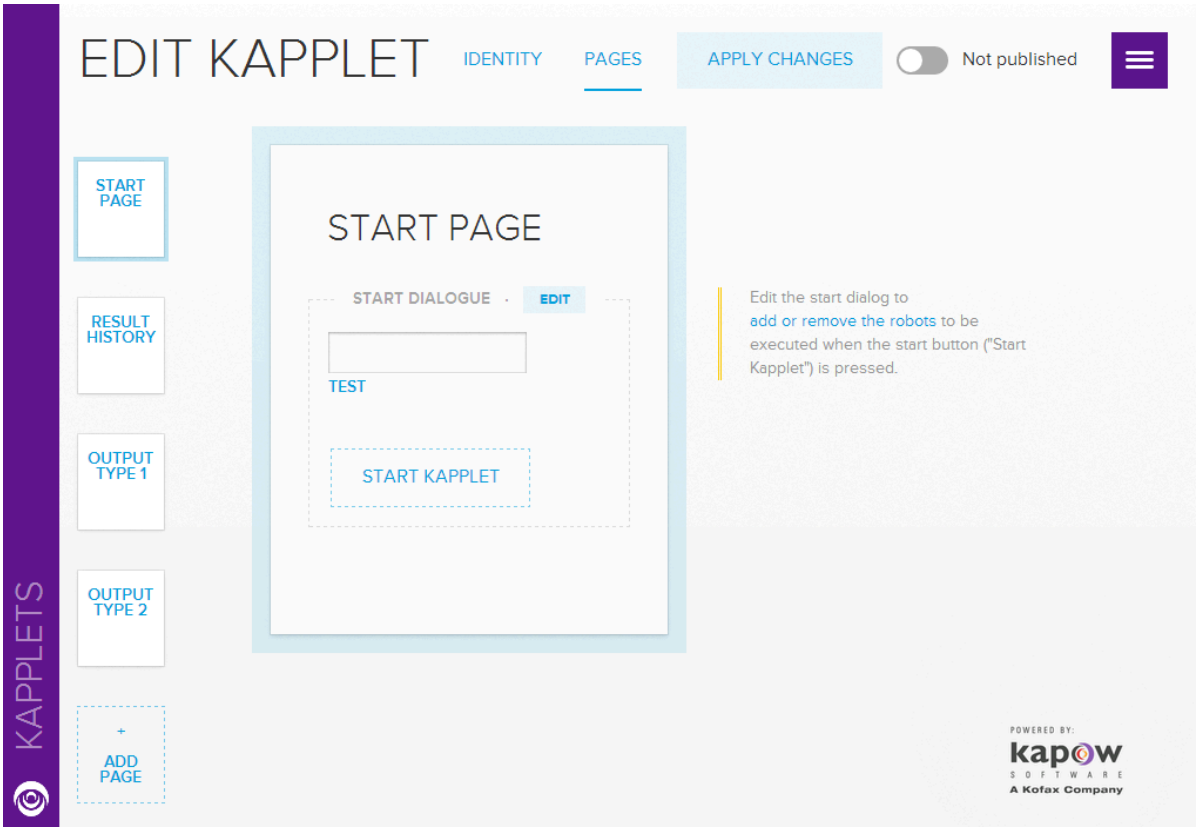
注 開始ページと結果履歴ページは必須です。これらのページは、開始アクションのために入力が収集される方法と結果履歴が表示される方法をそれぞれ管理します。

a. [開始ページ] を設定します。

b. [結果履歴ページ] を設定します。

c. 必要に応じて、残りのページを設定します。
残りのページは、開始アクションで生成された結果が表示される方法とその操作方法を管理します。

5. Kapplet Studio の上部で、[変更の適用] をクリックします。



重要 変更は適用しない限り失われるため、必ず適用する必要があります。

開始ページの設定

1. ページ セクションを開きます。
このページのコンテンツは、開始アクションに関連付けられているロボットによって異なります。ロボットが関連付けられていない場合、このページは空になります。[ロボットの追加] 領域をクリックするだけでロボットを追加できます。

2. ロボットを追加するには、[ロボットの追加] 領域をクリックします。
   開始アクションの追加ページが表示されます。
3. ロボット開始の追加を使用して、アクションにロボットを選択します。
4. [新しいロボットの追加] をクリックします。
   選択ページに利用可能なロボットのリストが表示されます。
5. ロボットを選択するには、ロボット アイコンをクリックしてから、[ロボットの選択] をクリックします。
   追加されたロボットによって定義された入力変数の属性のリストが表示されます。次に、各属性に対して、Kapplet 管理者は、固定値を関連付ける必要があるかどうか (および、ユーザーに対してフィールドを非表示にするかどうか)、または入力フィールドを表示する必要があるかどうかを決定できます。たとえば、追加された 'test' ロボットは、2 つの属性 (1 つはユーザー入力で、もう 1 つは固定属性) を持つ単一の変数 'input' を想定します。

   注 ロボット名をポイントすると、ロボット ファイルのパスが表示されます。

6. [**OK**] をクリックします。
   Kapplet 管理者はページ セクションに戻ります。このセクションでは、Kapplet の実行時に Kapplet ユーザーに提供されるページのアウトラインが開始ページに表示されています。
7. Kapplet 管理者は、ページ タイトル、フィールドの順序やラベル、開始ボタンのラベルなど、アウトラインを操作して、ユーザー エクスペリエンスを変更することもできます。

   注 また、ページ セクションには、左側に出力タイプ (ある場合) のリストが表示されます。

Kapplet での OAuth の使用

Kapplet に追加したロボットにサービス プロバイダに接続するための OAuth 入力が含まれる場合、Kapplet の編集画面には、OAuth クレデンシャルの設定セクションが含まれます。

1. サービス プロバイダに接続するには、[**OAuth** クレデンシャルの設定] を選択します。
2. リストからアプリケーションを選択します。
   アプリケーションの開始ページが表示されます。
3. [接続] をクリックします。
   承認プロセスが開始され、ユーザーはアクセスを許可するよう求められます。このフローが完了すると、接続ボタンが非表示になります。
   ユーザー用に格納されている認証を表示するには、メイン メニューで [設定] を使用します。

Kofax Analytics for RPA ダッシュボードへの接続

Kofax Insight を使用すると、組織は、複雑な大量のデータをすばやく収集、分析、視覚化して、意思決定、業務効率、および収益性を向上させることができます。Kofax RPA Kapplet は、シングル サインオン メカニズムを使用して Kofax Insight ダッシュボードに接続する手段を提供します。次の手順を実行して、ダッシュボード Kapplet を作成します。

1. Kofax RPA Management Console で、**[管理]** > **[設定]** > **[Kofax Insight** ダッシュボード**]** を選択します。
2. [ダッシュボードを許可] をクリックします。
3. Insight ダッシュボードをホストするサーバーの [サーバー **URL**] を指定します。
4. シングル サインオン パラメータを設定するには、**[管理]** > **[設定]** > **[**シングル サインオン**]** を選択し、Kofax Insight が使用するデータベースに接続するために必要なパラメータを指定します。

5. Kofax RPA Management Console で、Kapplet に移動し、KappZone を開きます。

   注 KappZone が既に開いている場合は、KappZone を再起動します。

6. メイン メニューで、[新しいダッシュボード] をクリックします。
   [新しいダッシュボードの追加] ウィンドウが表示されます。
7. Kofax Insight に接続するためのパラメータを指定します。
   a. [ダッシュボード名] に名前を指定します。この名前は、Kapplet の表示名です。
   b. 他のフィールドは空のままにします。

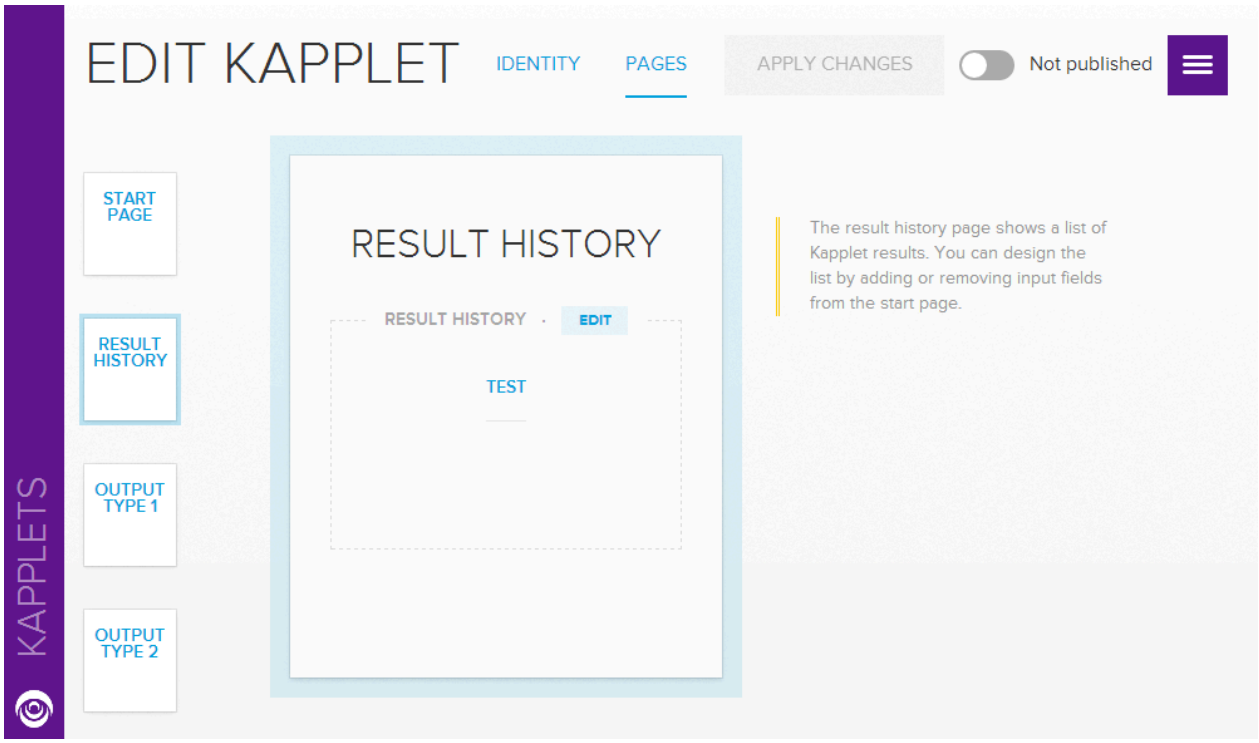
   注 何も指定しない場合、Kofax Insight は、そのデフォルト パラメータを使用します。または、シングル サインオンが有効化されている場合、ユーザーのデフォルト パラメータを使用します。

8. [ダッシュボードの作成] をクリックします。

結果履歴ページの設定

すべての Kapplet には、Kapplet ユーザーが過去および現在の Kapplet 実行の結果にアクセスできる結果履歴ページがあります。このページには、結果が項目のリストとして表示されます。各結果は過去の特定の実行に対応し、タイムスタンプと、当該の実行の開始アクションに指定された入力パラメータの列挙で構成されます。

Kapplet Studio では、結果履歴ページのアウトラインをレビューしたり、一部の要素 (現時点ではページ タイトルのみ) をカスタマイズしたりできます。次に示すように、'test' サンプル ロボットが、タイムスタンプ (非表示) と、'iterations' パラメータに対応する単一の入力値で構成される エレメントのリストに表示されます。



コミット アクションの追加

開始アクションで生成されたデータセットを表形式ビューで表示するように結果ページを設定すると、コミット アクションを Kapplet に追加できるようになります。このウィンドウを使用すると、ロボットを選択して、データベース、Web サービス、または実際の Web サイトに選択したテーブル行を容易にコミットできます。ロボットは、選択テーブルに表示されているデータと同じタイプの入力データを取得し、コミットの結果を入力タイプとは異なるタイプの単一オブジェクトの形式で返す必要があります。

コミット アクションが完了すると、返された情報を元の情報と一緒に表示できるように、現在の選択テーブルが返されたタイプで変更されます。

1. 結果ページ テーブル セクションで、[ロボットの追加] をクリックします。
   [新しいコミット アクションの追加] ウィンドウが表示されます。
2. [**OK**] をクリックします。
   コミット アクションを追加すると、Kapplet が (再度) 正しく設定されるため、Kapplet を使用してデータを収集、選択、コミットできるようになります。

   重要 Kapplet ユーザーが Kapplet を使用できるようにするには、Kapplet 管理者が Kapplet を公開する必要があります。

3. ウィンドウ上部にある適切なトグルを選択します。

Kapplets のインストールと使用

現在、Kapplet 管理者になっている Kapplet ユーザーの場合、通常、KappZone を Management Console URL に追加すると、Web ブラウザから Kofax RPA Classic Kapplet に直接アクセスできます。たとえば、Management Console を http://localhost:50080/ にデプロイしている場合は、http://localhost:50080/kappzone で Kapplet に直接アクセスできます。

注 Kapplet がアンインストールされている場合にはインストールの操作、Kapplet がインストールされている場合には開く操作 (実行) が可能です。

ユーザーは、KappZone タブで Kapplets にアクセスできます。ユーザーが Kapplet をインストールすると、ユーザーの My KappZone で Kapplet のインスタンスを使用できるようになります。特定の Kapplet ユーザーは任意の Kapplet を一度だけインストールできます。インストールされた Kapplet は、My KappZone (My KappZone をクリック) と KappZone ([開く] をクリック) の両方から実行することができます。

ユーザーはブラウザの My KappZone からインストール済みの Kapplets をブックマークしたり、インストール済みの Kapplet に星印を付けて、お気に入りの表示に追加したりできます。このようにして、ユーザーは最も使用頻度の高い Kapplet にスムーズに移動できます。

Kapplet をブックマークするには、ブラウザに表示される Kapplet URL を見つけます。単純に現在のページをブックマークできます。

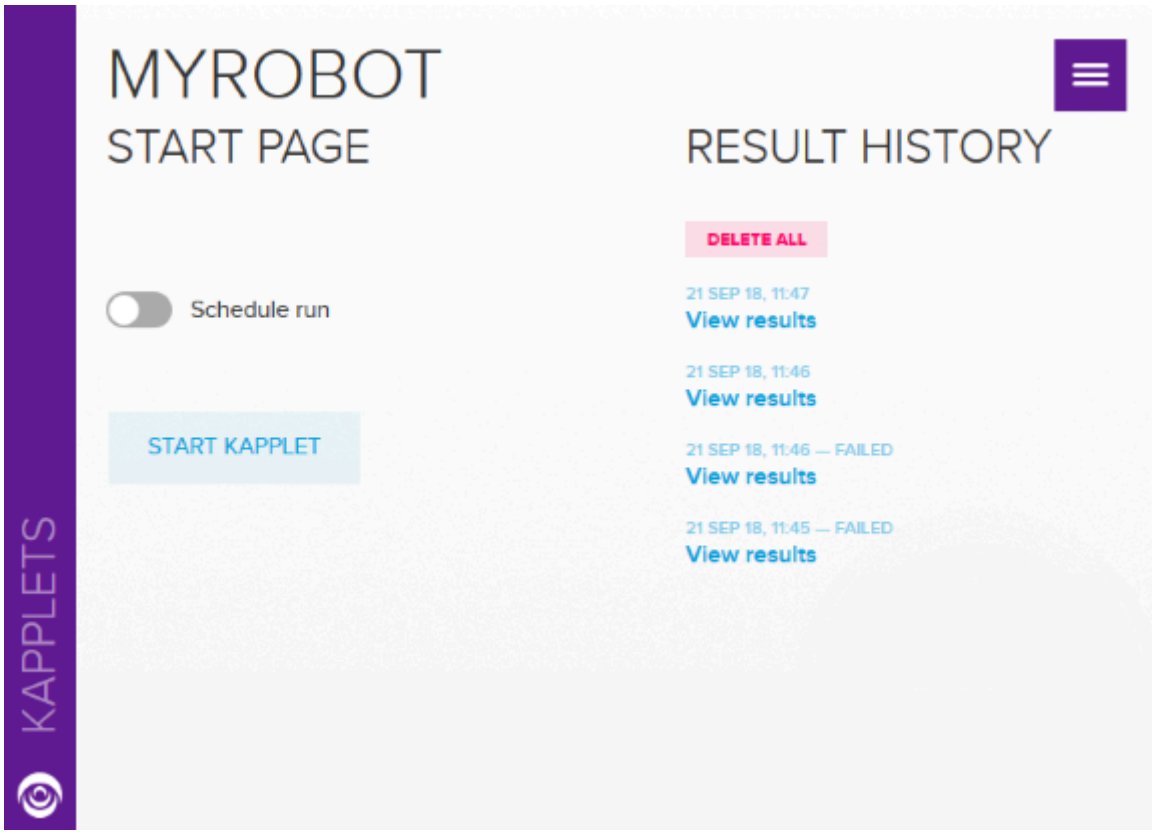
KappZone を開き、[お気に入り] をクリックして、現在のユーザーのインストール済みの Kapplets を表示します。

Kapplets の呼び出し

インストールした Kapplet を実行するには、My KappZone またはお気に入りで Kapplet をクリックし、必要なすべてのフィールドに入力し、開始ボタンをクリックします。以下の例の 'test' Kapplet では、'iterations' フィールドのみに入力する必要があります。

通常、Kapplet ユーザーには、Kapplet を使用するときにビューが表示されます。このビューでは、アクティブなすべての Kapplet ページが左から開始ページ、結果履歴ページ、結果ページ (関連する場合) の順に同時に表示されます。Kapplet を開始するまで、または結果履歴ページからアクティブな Kapplet 実行を選択するまでは、開始ページと結果履歴ページのみが表示されることに注意してください。

Kapplet を開始すると、対応する新規 Kapplet 実行が結果履歴ページにリストされ、対応する結果ページが結果履歴ページの右側に表示されます。アクティブな (まだ削除されていない) Kapplet 実行は常に保存され、対応するインストール済みの Kapplet の結果履歴ページで見つけることができます。関連する結果履歴ページのエントリをクリックして、対応する結果ページを開くことができます。次の例は、「Myrobot」Kapplet の結果を示します。

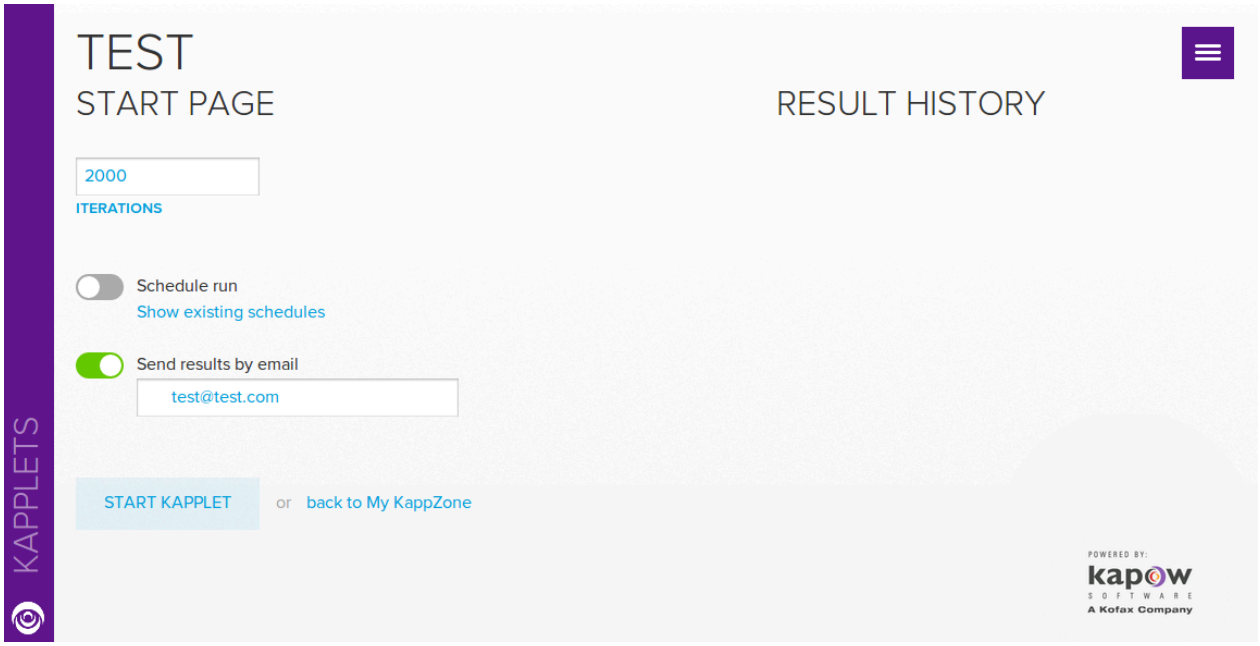


サード パーティのプログラムで Kapplet のテーブル形式の結果をさらに操作する場合は、Kapplet の結果を Excel 形式でダウンロードできます。Kapplet の結果履歴全体を削除するには、[結果履歴] の下にある [すべて削除] をクリックします。

Kapplet からの電子メール通知の作成

長時間実行される Kapplet の場合、結果ページの生成を待機する時間を短縮したい場合があります。Kapplet の実行を開始するときに、電子メール アドレスを入力すると、結果が利用できるようになったときに電子メールを受け取ることができます。

SMTP サーバーが設定されていて、Kapplet ユーザーの電子メール アドレスが入力されているインストール済みの Kapplet を次に示します。



1. [電子メールで結果を送付] を有効にします。
   このオプションは、Management Console で SMTP サーバーが設定されていて、Kapplet に対して 'from' アドレスが設定されている場合にのみ表示されます。
2. [電子メールで結果を送付] フィールドに有効な電子メール アドレスを入力します。
3. [**KAPPLET** を開始] をクリックします。
   開始アクションが完了すると、入力した電子メール アドレスに電子メールが送信されます。

Kapplets のスケジュール

毎日正午、毎週金曜日の午後 4 時 50 分など、あらかじめ定義した間隔で Kapplets を実行することができます。

1. Kapplet をスケジュールするには、スケジュール実行セクションで、リストから実行オプションを選択し、適切なスケジュール情報を入力します。

   時刻指定は、Management Console を実行しているサーバーのタイム ゾーンの時刻として解釈されることに注意してください。また、[スケジュール実行] オプションを切り替えると、開始ボタンがスケジュール ボタンに変更されることに注意してください。

   Kapplet をスケジュールすると、Kapplet は、Kapplet ユーザーが指定したとおりに定期的に実行されます。[既存のスケジュールを表示] リンクをクリックして、インストールした Kapplet のスケジュール設定を画面の左側にリストすることができます。

   インストールした Kapplets を Kapplet ユーザーがリストすると、Kapplet にスケジュールが表示されます。
2. スケジュールを削除するには、スケジュールを選択し、左側にあるスケジュール リストで削除アイコンをクリックします。

   Kapplets に対して、スケジュールおよび電子メール通知を組み合わせることができます。

Kapplet ブランディングのカスタマイズ

KappZone と My KappZone のブランディングをカスタマイズして、企業のカラー スキームに適合させたり、Kofax RPA 口ゴを独自の口ゴに置き換えたりできます。

1. Management Console の [Branding, Kapplets, Branding] セクションで、[カスタム] を選択します。

   注 右側に口ゴと色のプレビューが表示されます。
2. [ここに口ゴをドロップ] または [クリックしてアップロード] で、使用する口ゴを参照します。ブラウザで機能がサポートされている場合、新しい口ゴを領域にドラッグ アンド ドロップすることもできます。

   My KapZone および KapZone ウィンドウの上部に新しい口ゴが表示されます。一部のサブページには、小さなバージョンの口ゴも表示されます。
3. ブランド カラー領域で、背景色コードを指定します。
4. コントラスト セクションで、コントラスト オプションを選択します。

   この色は、My KapZone テキストに加えて、選択した背景色の上部に配置される一部のアイコンに使用されます。暗い背景には明るいコントラスト色を指定し、明るい背景色を指定した場合は暗いコントラスト色を使用することをお勧めします。

Management Console での Kapplet のアクセス

Management Console では、Kapplets は Kapplets タブで管理できる別のエンティティです。

このタブを選択して、サブセクションを表示します。

KappZone
現在使用が許可されている、Kapplet すべてを表示します。ここから Kapplets をインストールし、マイ KappZone で利用できるようにします。Kapplets の呼び出しを参照してください。

リポジトリ
現在定義されている Kapplet のリストを表示します。ここでは、Kapplet の定義の表示、削除、編集を行うことができます。新しい Kapplets を作成するには、Kapplets ユーザー エリアを開きます。Kapplets の作成を参照してください。

ブランディング
Kapplet の管理者は [ブランディング] タブを使用して、Kapplets の基本的な外観をカスタマイズします。Kapplet ブランディングのカスタマイズを参照してください。

注 すべての Management Console ユーザーが Kapplets を使用したり、管理したりできるわけではありません。権限は、プロジェクトの [アクセス許可] タブで割り当てられています。注意：セキュリティ オプションは、標準 (Standard) 版のものよりも厳格ではりません。すべてのユーザーに、Kapplets の表示・編集の権限を付与します。

Kapplet の構築および使用についての情報は、「Kofax RPA Kapow Kapplets」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。

Kofax RPA は、IP プロキシ プロバイダがランダムまたは事前に設定された時に IP を変更する IP ローテーション モデルの使用を推奨しません。すべての Web サイトが、IP アドレスの変更を通じてブラウザ セッションを維持できる、または維持を許可しているわけではないからです。

開かれたインターネットでは、典型的なブラウザ ソケット接続は寿命が非常に短く、Web サイトがソースの IP アドレスを確認しないため、IP ローテーションはうまく動作することができます。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てセキュリティに注目が集まり、多くの Web サイトはセキュリティ レベルを引き上げ、IP アドレス監視を追加しています。Web サーバーが、セッション内の IP アドレスの変化を検出して防止するのは中間者攻撃を防ぐ良い方法です。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 アドレスが変化すると、進行中の Web サーバーとのセッションが終了する可能性が高くなるため、ロボットを動作させながら、IP アドレスを自由にローテーションす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
- プロキシ サーバー Design Studio 設定の
- Management Console でのプロキシ サーバーの構成

Kofax RPA でのパスワード暗号化

Kofax RPA は、「bcrypt」パスワード ハッシュ関数を実装して、プログラムでパスワードの使用と保存を行います。

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。

ブラウザ
- Kofax RPA ではブラウザを使用して、Web サイトまたは Web アプリケーションと通信します。実際には、Kofax RPA には現在、2 つの異なるブラウザが搭載され、それぞれ別の目的向けに最適化されています。クラシック エンジンはレガシー Web アプリケーション向け、デフォルト エンジンは現在の標準的な Web アプリケーション向けです。ただし、これらのブラウザは市場のその他のブラウザと同様に、すべての内部アプリケーションまたはインターネット Web サイトと互換性があるわけではありません。
- [スナップショット生成] を使用してダウンロードされた Web ページは、Kofax RPA の内部ブラウザからダウンロードされたコンテンツとスタイルを表したものです。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、特定のステップのアクションはループの内側、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 アクションは、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。制限は多くの要因によって異なります。たとえば、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利用可能

なメモリ、ロボットが行う修正の数などです。そのため、Excel 文書の大きさが原因で Kofax RPA が処理を実行できない場合についての正確な基準を提供することはできません。

- 数式のサポート

  Excel での数式サポートの詳細については、https://poi.apache.org のサイトを検索してください。
- 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
- 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

  注 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、サポートされている Excel の機能を参照してください。

パスワードの長さ

たとえば、クラスタ データベース、Logdb、Analytics、プロキシ、SMTP のパスワードなど、Kofax RPA で使用されるすべてのパスワードは、117 バイトを超えてはなりません。

データベース

IBM DB2 データベースの次の制限に注意してください。

- すべてのテーブルを作成するためにはデータベースに少なくともページ サイズが 8192 KB の「テーブル領域」がある必要があります。
- DB2 データベースでのストアド プロシージャの呼び出しはサポートされていません。

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。
- REST Web サービス呼出 (旧バージョン)
- Web サービスの呼び出し (旧バージョン)
- Web ストレージ消去
- タグの分割 (旧バージョン)
- テキスト分割
- テキスト分割 (旧バージョン)
- SQL 実行 (旧バージョン)
- Excel から抽出 (旧バージョン)
- PDF から抽出 (旧バージョン)
- 画像抽出
- ファイル繰り返し (旧バージョン)
- タグ非表示化
- タグ挿入
- データのロード (旧バージョン)
- スナップショット生成
- セル結合解除
- データベース照会 (旧バージョン)
- テーブル行除去
- タグ範囲除去
- タグ除去
- タグ除去 (旧バージョン)
- タグ書き換え
- タグの復元
- ページ再描画
- CSS 再描画
- ファイル選択
- トップ タグの設定 (旧バージョン)
- フォーム送信
- テーブル行列入れ替え
- タグ表示化

Kofax RPA Kapplet

次のリストには、Kofax RPA Kapplet で利用できない機能が含まれています。

- Kofax RPA Kapplet バックアップのインポートとエクスポート
- 結果を Excel ファイルにエクスポート
- WebSphere アプリケーション サーバーのサポート
- DB2 データベースのサポート
- Kapplet のスケジューリング
- Oauth 口ボットを Kapplet に追加する

大量のデータ出力を含む口ボットで Kapplet を使用することはお勧めしません。

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口ボットを構築し、口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。