遅延実行フレームワーク

ユーザーガイド

版 1.0

目次

概要

なぜ、遅延実行フレームワークが必要か

Blue Prism をスケジュール実行していると、処理が思ったよりも長引いて、後に予定していたスケジュールがリソースを獲得できず、エラー終了してしまうことがある。

このようなケースを指して、「Blue Prism の製品機能に瑕疵がある」と主張する意見があるが、どうだろうか。少なくとも、私は同意しない。そもそも、スケジュール実行というものはそういう（実行が重複するとリソースを獲得できなくなることがある）ものであり、それを避けたいのであれば、Blue Prism は実現するための機能を提供しているからだ。

「スケジュールの重複を避けたい」という主張を裏返すと、「リソースが空いたタイミングで、順次実行したい」という要件だと理解できる。このような要件を実現する方式を、ここでは「遅延実行」と呼ぶ。実行要求をリアルタイムに処理するのではなく、多少の遅延を許容してでも実行したい、という意味合いを込めている[1]。

遅延実行の仕組みを提供する「遅延実行フレームワーク」は、Blue Prism の AutomateC.exe、ワークキューなどを組み合わせることで実現が可能であり、その実現方法はひとつに限定されない。ここでは、ひとつのやり方をオープンソース（OSS）として公開する。

このアセットは、既に公開されているアセットである

Blue Prism Dynamic Scheduler Examples（英語）

https://portal.blueprism.com/documents/blue-prism-dynamic-scheduler-examples

を実装のベースとしている。

遅延実行フレームワークのメリット

Blue Prism ユーザーは、このアセットを活用し、またユーザー固有の要件に基づいて拡張をすることで、以下のようなメリットを享受できると考えている。

1. Blue Prism のプロセスを、リソースが空いたタイミングで、順次実行できる
2. シンプルな実行方法で遅延実行が行える
3. リアルタイム起動フレームワークと組み合わせることで、リアルタイムに実行要求を受付、リソースが空いたタイミングで、順次実行できる

[1] これはいろいろな解釈が可能で、「リソースが空いたタイミングで実行する」ことから、As Soon As Possible（ASAP）と呼ぶこともできる。これは OSS の仕組みなので、好きに呼べばいいだろう。

遅延実行フレームワークを構成する

前提条件

1. Blue Prism 6.7 で構成された、Blue Prism サーバと最低 2 つ以上（推奨は 3 つ以上）のパブリックなランタイムリソースが稼働している環境（リアルタイム起動フレームワークと組み合わせる場合は、最低 3 つ以上、推奨は 4 つ以上）
2. .NET Framework 4.7.2 がインストール済みの端末（クライアント・アプリを動かすため）[2]
3. リアルタイム起動フレームワークが構成されていること

アセットに含まれる成果物

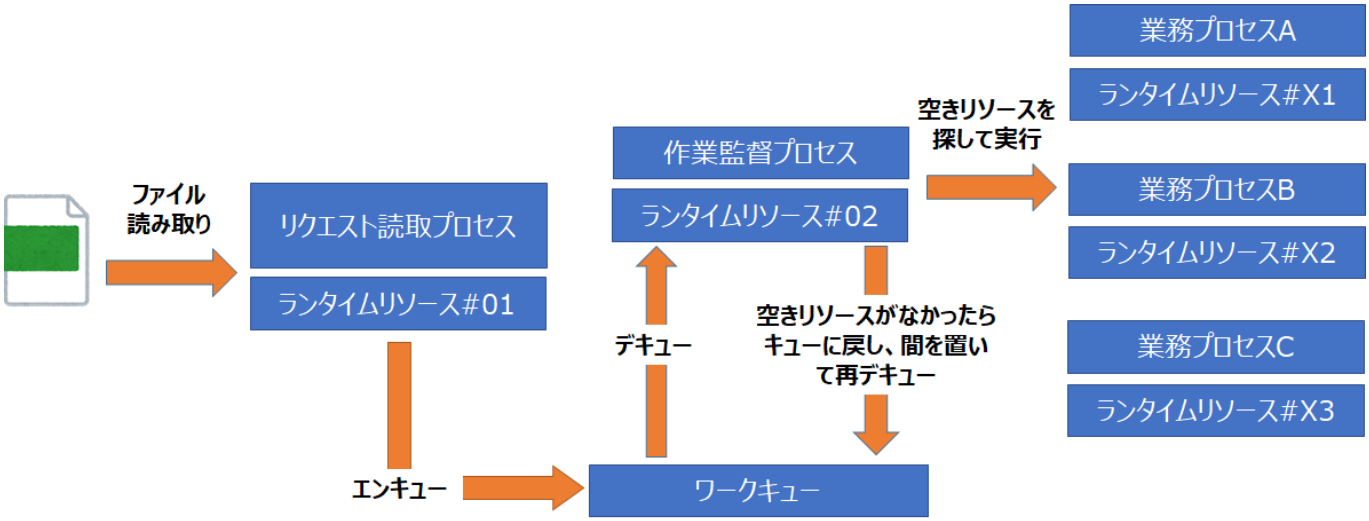
このアセット（遅延実行フレームワーク）には、以下の成果物が含まれる。

場所	ファイル名	概要
ドキュメント	遅延実行フレームワーク ユーザーガイド 1.0.pdf	このドキュメント
YouTube	Blue Prism で遅延実行フレームワーク（試製一型）	https://www.youtube.com/watch?v=YzhQtZjiGPo 遅延実行フレームワークのデモ動画
YouTube	Blue Prism で遅延実行フレームワーク（完成版）	https://www.youtube.com/watch?v=lqeYM0DpN1k&t=1s リアルタイム起動フレームワークと組み合わせた場合の、遅延実行フレームワークのデモ動画
YouTube	Blue Prism のリアルタイム起動と遅延実行	https://www.youtube.com/watch?v=f0aC1thHO_k リアルタイム起動と遅延実行の考え方と、その違いについて説明した動画
プログラム	遅延実行フレームワーク ver 1.0.bprelease	遅延実行フレームワークを構成するプロセスやオブジェクト（VBO）が含まれる
プログラム	遅延実行フレームワークのクライアント（ソースコード）.zip	Microsoft Visual Studio で開くことが可能なプロジェクト

[2]クライアントアプリを作る際に .NET Framework 4.7.2 を選択しただけであり、4.7.2 固有の機能が使われている可能性は低い。もし、より古いバージョンの.NET Framework で動作させたい場合は、提供されているクライアントアプリのソースからビルドを試みることができる

各コンポーネントの関係図

遅延実行フレームワークは、以下のような関係で各コンポーネントが動作する。



各コンポーネントの配置手順

bprelease のインポート

遅延実行フレームワーク ver 1.0.bprelease を Blue Prism 6.7.1 にインポートする。

特定のリソースで、作業監督プロセスを起動する

作業監督プロセスは、一度、実行した場合、停止を要求するまで半永久的に動き続ける。

（オプション）返却係の設定

リアルタイム起動フレームワークの当該箇所を参照。

（オプション）返却係のための認証情報の設定

リアルタイム起動フレームワークの当該箇所を参照。

（オプション：リアルタイム起動フレームワークと組み合わせる場合）オブジェクトの公開

オブジェクト「Reception - JP」および「Reception – Delayed Run」を公開する

（オプション：リアルタイム起動フレームワークと組み合わせる場合）クライアントアプリのインストール

遅延実行フレームワークのクライアント（ソースコード）.zip を解凍し、Visual Studio でインストーラーを作成する（ビルド > リアルタイム起動フレームワークのクライアントの発行 など）。Visual Studio でのインストーラーの作成方法は、Visual Studio のマニュアルや関連ドキュメントを参考にしてほしい。

なお、クライアントアプリは Reception – JP および Reception – Delayed Run が公開されたランタイムリソースを参照するようになっている。デフォルトでは両方とも localhost:8182 を参照しており、これを変更したい場合は、App.config の以下の箇所を修正し、インストーラーを作成する。

```
    <client>
        <endpoint address="http://localhost:8182/ws/ReceptionJP" binding="basicHttpBinding"
          bindingConfiguration="ReceptionJPSoapBinding"
contract="ReceptionJP.ReceptionJPPortType"
          name="ReceptionJPSoap" />
        <endpoint address="http://localhost:8182/ws/ReceptionDelayedRun"
          binding="basicHttpBinding" bindingConfiguration="ReceptionDelayedRunSoapBinding"
          contract="ReceptionDelayedRun.ReceptionDelayedRunPortType"
          name="ReceptionDelayedRunSoap" />
    </client>
```

インストーラーを作成したら、クライアント端末に配布し、クライアントアプリのインストールを行う。

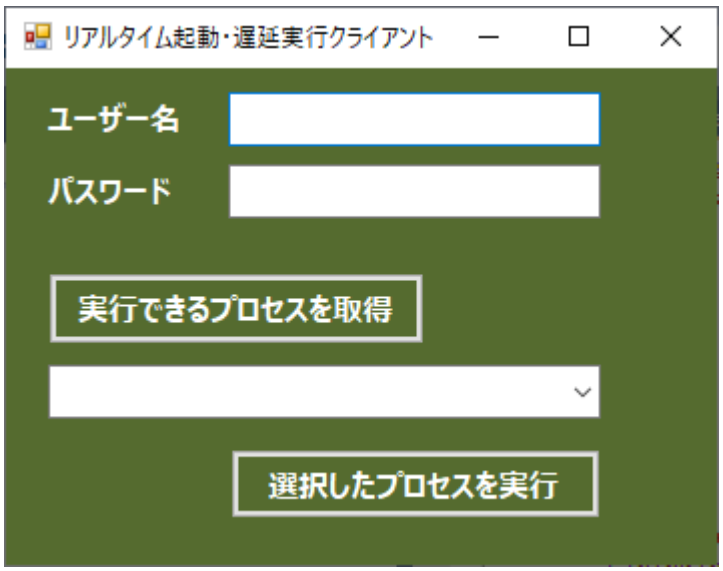
稼働確認

リクエスト読取プロセスを使ったエンキュー

1. 事前に Excel ファイルを用意し、Sheet1 シートの A1 セルから縦に順番に、実行したいプロセス名を記述し、保存する（デフォルトのファイル名とパスの組合せは"C:\temp\requests.xlsx"となっている）
2. いずれかのリソースでリクエスト読取プロセスを 1 回だけ動かし、Excel の内容をエンキューする
3. 稼働中の作業監督プロセスが、キューの中身を読み取り、空いているリソースに割り当てていく

（オプション：リアルタイム起動フレームワークと組み合わせる場合）クライアントアプリの操作

クライアントアプリの各項目の説明は以下の通り。



1. ユーザー名：Blue Prism のユーザー名を入力
2. パスワード：Blue Prism のパスワードを入力
3. 実行できるプロセスを取得：入力したユーザーが実行できるプロセス群を取得してプロセス名の候補に設定
4. リストボックス：実行したいプロセス名を候補から選択
5. 選択したプロセスを実行：選択したプロセス名を、任意のランタイムリソースで実行

全体の動き

遅延実行フレームワークの全体の動きは、以下の動画を参照。

https://www.youtube.com/watch?v=YzhQtZjiGPo

リアルタイム起動フレームワークと組み合わせた場合の全体の動きは、以下の動画を参照。

https://www.youtube.com/watch?v=lqeYM0DpN1k&t=1s

拡張について

拡張の考え方

リアルタイム起動と同様に、遅延実行に対する Blue Prism ユーザーの要件はさまざまである。したがって、遅延実行フレームワークはあくまでサンプル実装とし、オープンソース（OSS）で提供している。

要件が様々である以上、拡張もまた様々であると考えられる。以下にこのユーザーガイドを執筆している時点で想定可能な主な拡張のポイントを列挙しておく。

拡張のポイント

主な拡張のポイントは以下である。

1. SSO（シングルサインオン）への対応
2. ログインエージェントへの対応
3. 通信プロトコルに応じたクライアントアプリの改修（現在は、HTTP/認証ありを前提としている）
4. ランタイムリソースの選択アルゴリズムを増やす
5. 入出力パラメータの追加
6. AutomateC.exe のロケールを en-US に固定する（今後のバージョンアップで AutomateC.exe の出力文字列が変わった場合の対処）
7. 公開したオブジェクトに Web API アクセスが行われるたびに、監査ログが出力されるので、その抑制
8. いくつかのデータアイテムを環境変数にする

以上