目次

改訂情報

変更年月日	変更内容
2020-09-01	初版
2020-12-01	第2版 以下を追加・変更しました。 ▪ 「intra-mart製品との連携」を追加 ▪ 「*セットアップ*」に 「*WinActor連携 設定編集画面*」の利用について追加 ▪ 「*WinActor連携 設定編集画面*」を追加 ▪ 「*WinActor連携チュートリアル*」を追加 ▪ 「*BizRobo!連携チュートリアル*」を追加 ▪ 「*UiPath連携チュートリアル*」を追加 ▪ 「*ダッシュボード作成チュートリアル*」を追加

はじめに

本書の目的

本書は、IM-RPA連携を利用するユーザのみなさまの支援を目的としたドキュメントです。

対象読者

本書では以下のユーザを対象としています。

- WinActor連携を利用し、シナリオをリモートから実行したい
- BizRobo!連携を利用し、BizRobo!のサービスを呼び出したい
- UiPath連携を利用し、UiPathをリモートから実行したい

本書の構成

本書は以下のように構成されています。

- *IM-RPAとは*

  本書、および、IM-RPA連携の概要について説明します。

- *WinActor連携*

  WinActor連携のセットアップ手順、およびIM-LogicDesignerタスクの仕様を説明します。

- *BizRobo!連携*

  BizRobo!連携のセットアップ手順、およびIM-LogicDesignerタスクの仕様を説明します。

- *UiPath連携*

  UiPath連携のセットアップ手順、およびIM-LogicDesignerタスクの仕様を説明します。

- *著作権および特記事項*

  著作権および特記事項について記載します。

IM-RPAとは

項目

概要

IM-RPAは、WinActor, BizRobo!, UiPathとの接続を容易に行うための IM-LogicDesignerタスクを提供します。
IM-LogicDesignerを介す事により、IM-Workflowや、IM-BPMといった機能との連携を容易にします。
IM-RPAを利用することにより、業務プロセス中に任意のロボットを組み込み業務の自動化を推進します。

WinActor

WinActorは、NTTグループにより開発・利用されてきた長い歴史と豊富な導入実績に裏打ちされた機能を備えた純国産「RPA」ソリューションです。
WinActor連携タスクでは、複数の WinActorクライアントをグループ化し、シナリオの実行を行うことが可能です。

BizRobo!

BizRobo!は、ホワイトカラーの生産性を革新する、ソフトウェアロボット（Digital Labor）の導入・運用を支援するデジタルレイバープラットフォームです。
「ロボット」と「IT」によって、ホワイトカラーをルーティンワークから解放し、企業を始め社会全体の生産性向上を図り、未来の働き方を変えていきます。
BizRobo!連携タスクでは BizRobo!に対して、シナリオのアップロード、シナリオの実行等を行うことが可能です。
これにより、 IM-Workflowにより承認されたシナリオを BizRobo!へアップロードすると言った使い方により、ロボットの統制を行うことも可能です。

UiPath

UiPathは、気軽にRPAロボットを作成・実行できるUiPath Studioと、作成したロボットを組織全体で管理・運用するUiPath Orchestratorを備えるRPA製品です。
UiPath連携タスクでは UiPath Orchestratorに対して、ジョブの実行等を行うことが可能です。
UiPath Orchestratorを介することにより、ロボット実行の非同期化が可能となり、これにより大規模なロボット運用をより柔軟に行うことが可能です。

intra-mart Accel Platformとの連携

項目

intra-mart Accel Platformの製品群と各RPA製品の連携について

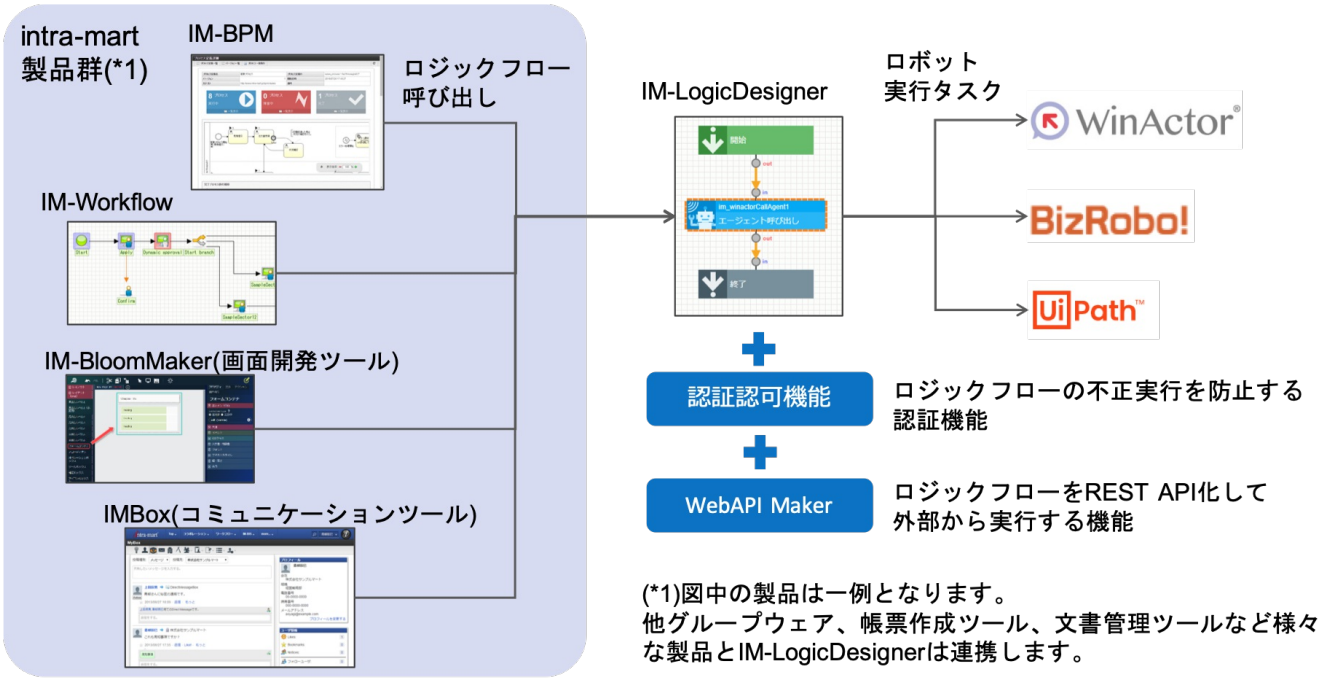
IM-RPA では、IM-LogicDesignerの「ロボット実行用タスク」を使用して各RPA製品のロボットを実行することが可能です。

intra-mart Accel Platformの製品群には、BPMやワークフローを簡単に作成・実行できるIM-BPM、IM-Workflow、ローコード画面開発ツールIM-BloomMaker、コミュニケーションツールのIMBox、その他グループウェア、帳票作成、文書管理、ポータルなど様々なラインナップがあり、そのほぼすべてがIM-LogicDesignerのロジックフローを呼び出すことが可能です。
すなわち、 intra-mart Accel Platformの製品群は全て、IM-RPAを利用することで、各RPA製品のロボットと連携できるということです。

さらに、 IM-LogicDesignerは柔軟で堅牢な権限管理機能「認証認可機能」を持ち、これにより適切な権限でロボットを実行可能です。

全体図

以下は、intra-mart Accel Platform製品群から各RPA製品への連携の全体図です。



図：連携の全体図

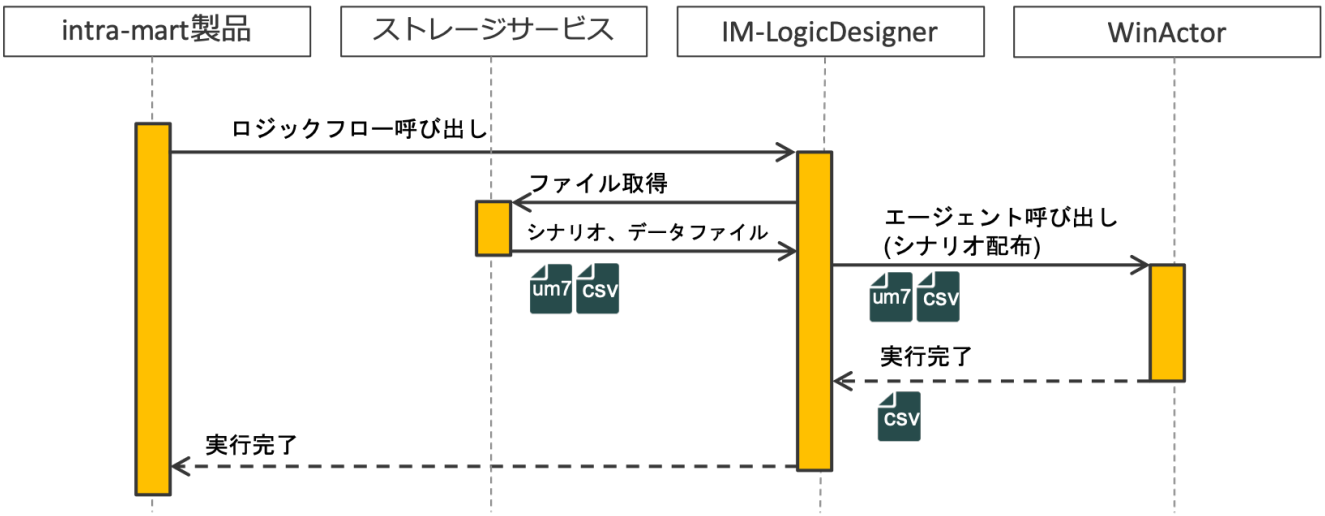
連携フロー概要

IM-LogicDesignerの「ロボット実行用タスク」は、連携するRPA製品によって、呼び出し方や条件に違いがあります。
ここでは、RPA製品ごとの呼び出し方の違いを説明します。
実際の設定方法については、後述する各RPA製品との連携章を参照してください。

WinActorとの連携フロー

WinActorの実行はIM-LogicDesignerの「エージェント呼び出し」タスクにて行います。

WinActor連携では、エージェント呼び出し時にロボットのシナリオ、およびロボットとの入出力パラメータを格納したデータファイルを配布します。

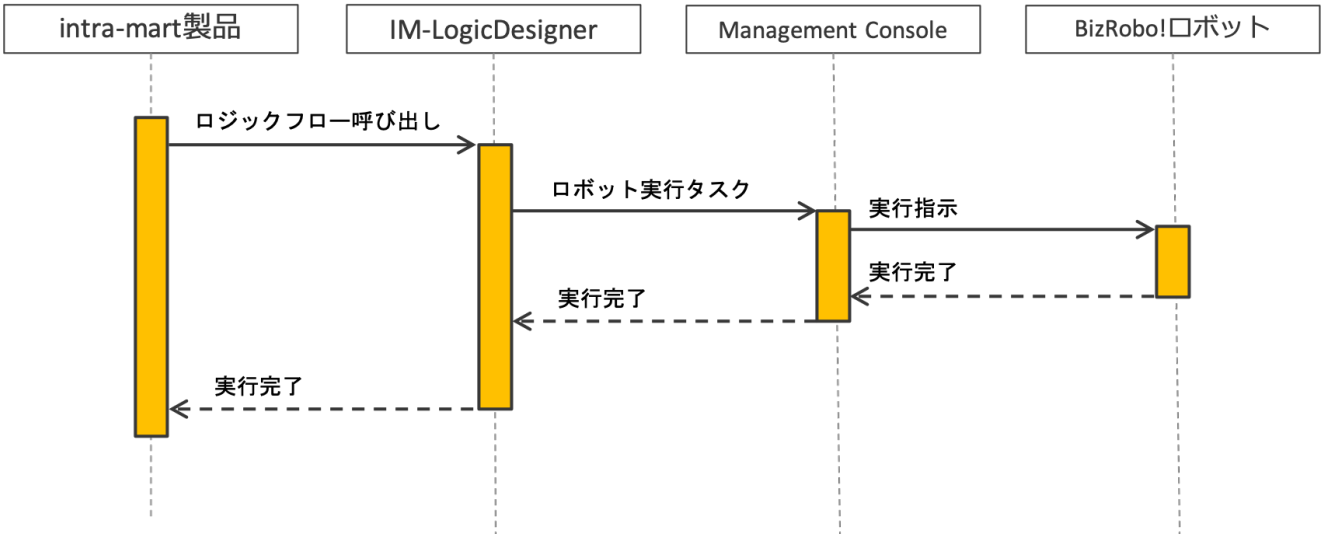


図：WinActorの実行シーケンス

> コラム
>
> 「エージェント呼び出し」タスクは、同期処理として実行されます。
> すなわち、ロボットの実行が終わるまで、タスクの処理は継続します。

BizRobo!との連携フロー

BizRobo!の実行はIM-LogicDesignerの「BizRobo!ユーザ定義」タスクにて、Management Consoleを介して行います。

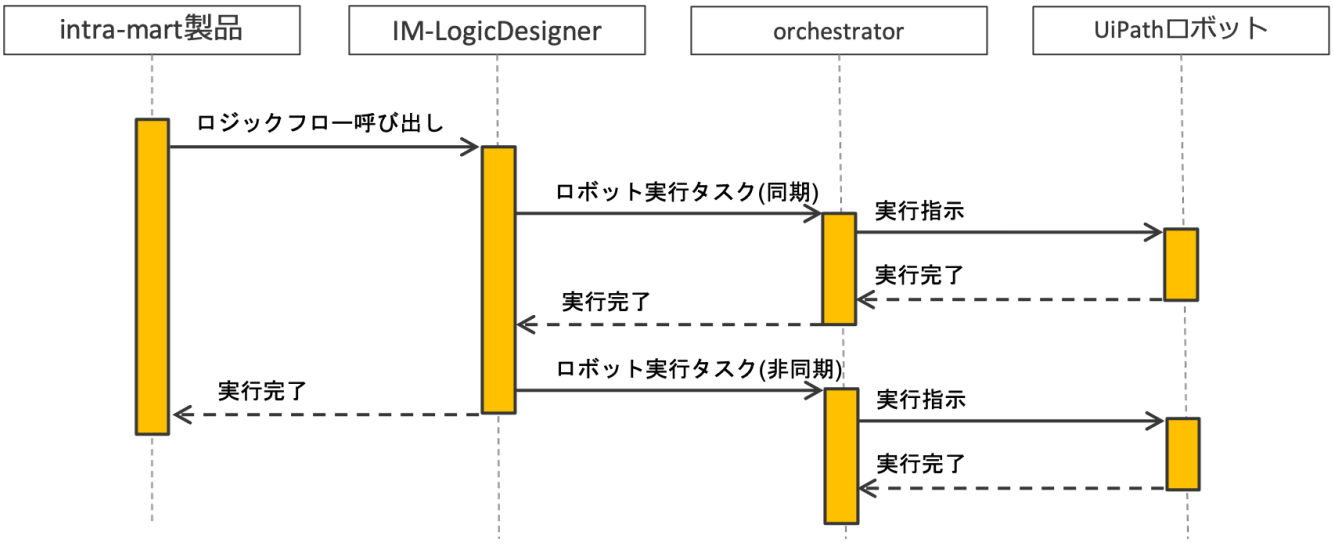


図：BizRobo!の実行シーケンス

> コラム
>
> 「BizRobo!ユーザ定義」タスクは、同期処理として実行されます。

UiPathとの連携フロー

UiPathの実行はIM-LogicDesignerの「UiPathユーザ定義」タスクにて、UiPath Orchestratorを介して行います。

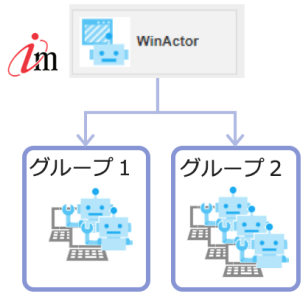
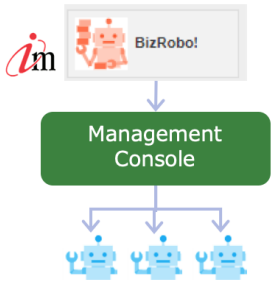
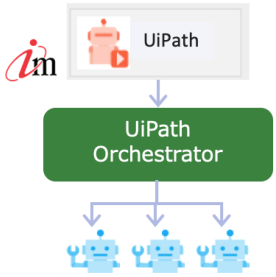


図：UiPathの実行シーケンス

> **コラム**
>
> 「UiPathユーザ定義」タスクは、同期/非同期処理で実行可能です。

各RPA連携方法のまとめ

各RPA製品と連携する際の構成と、連携方法をまとめたものが以下の通りです。

	WinActor	BizRobo!	UiPath
構成	 実行するロボットは iAPでグループ化して管理	 実行するロボットは Management Consoleで管理	 実行するロボットは Orchestratorで管理
シナリオ配布	可能	可能	不可
同期非同期	同期のみ	同期のみ	同期/非同期

図：RPAごとの連携方法まとめ

パラメータ等の連携について

いずれのRPA製品も、ロボットの呼び出しとIN/OUTパラメータの連携は可能です。
具体的な方法については、RPA製品ごとに違いがあります。詳細は、各RPA製品のチュートリアルを参照してください。

	WinActor	BizRobo!	UiPath
ロボット実行方法	WinActorエージェント呼び出しタスク	ユーザタスク定義 (BizRoboユーザ定義)	ユーザタスク定義 (UiPathユーザ定義)
パラメータのINPUT	可能 データファイル(CSV)経由	可能 データマッピング経由	可能 データマッピング経由
パラメータのOUTPUT	可能 データファイル(CSV)経由	可能 データマッピング経由	可能 データマッピング経由
ファイルの受け渡し	不可	可能 (リソース配置タスク)	不可

図：パラメータ等の連携についてまとめ

WinActor連携

ここでは、WinActor連携のセットアップ手順、およびIM-LogicDesignerタスクの仕様を説明します。

セットアップ

項目

IM-Juggling プロジェクトの編集

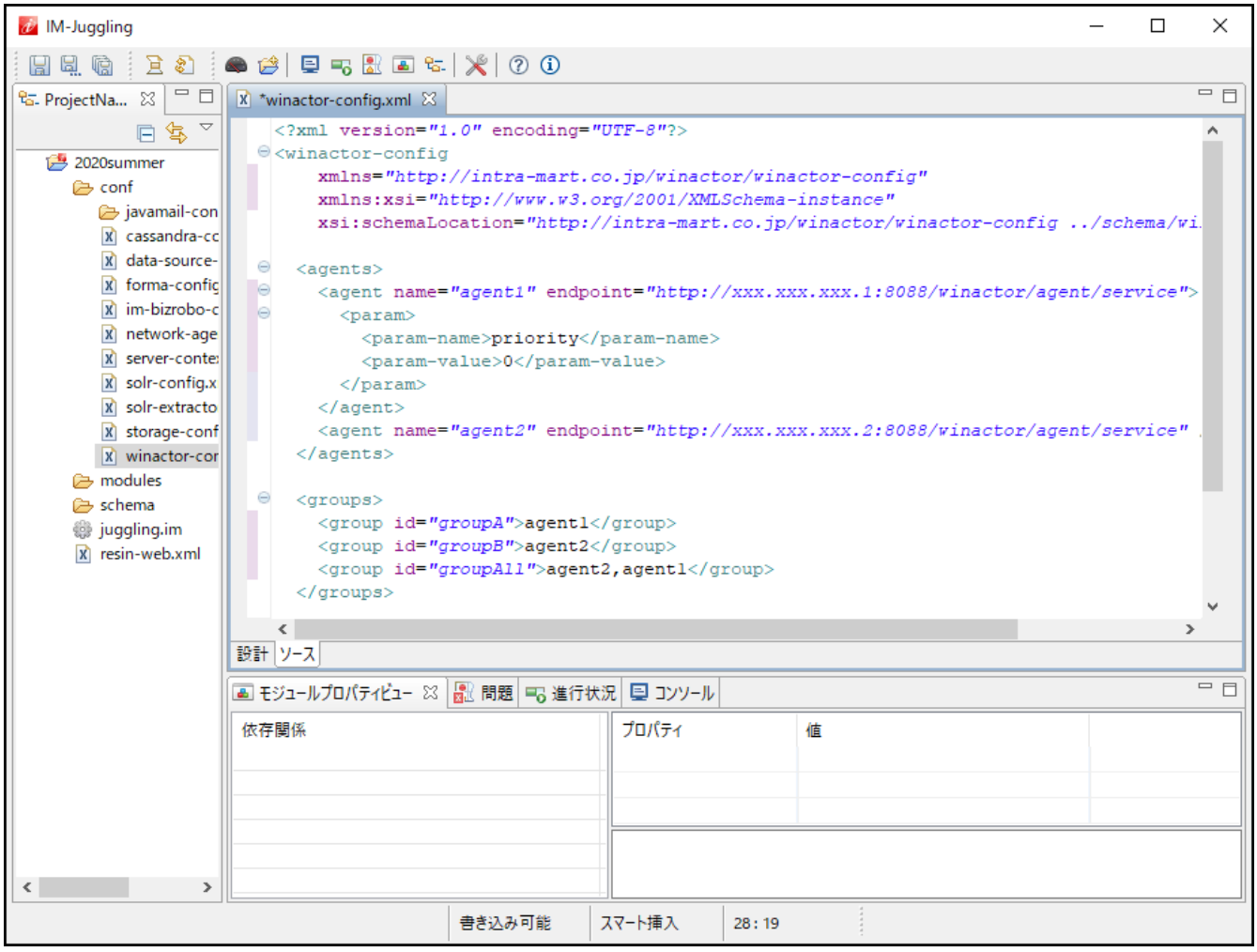
WinActor連携は、IM-RPAモジュールを使用します。
以下の手順で設定を行ってください。

1. IM-Jugglingにて、プロジェクトにIM-RPAモジュールを追加してください。
   以下のドキュメントを参照してください。
   - 「 intra-mart Accel Platform セットアップガイド 」-「 プロジェクトの作成とモジュールの選択 」

   > **注意**
   >
   > IM-RPAモジュールのご利用には、エンタープライズ版の構成、およびライセンスが必要です。

2. 「設定ファイルが存在しません」という赤字のメッセージをクリックします。
   表示されるダイアログにて「OK」をクリックし、このプロジェクトに「WinActorクライアント設定ファイル」(winactor-config.xml)を追加します。

3. 「ProjectNavigator」内の「 <（プロジェクト名）/conf/winactor-config.xml> ファイル」をダブルクリックで開き、「ソース」を選択します。



4. <winactor-config>/<agents>/<agent> タグのendpoint属性に対して、接続先エージェントのエンドポイントを記述します。
   エージェントが複数存在する場合は、<agent>タグを繰り返し設定してください。
   エンドポイントは以下の形のURLです。ポート番号は標準では8088を利用します。

   ```
   http://<エージェントのアドレス>:8088/winactor/agent/service
   ```

5. <winactor-config>/<retry>/<max-count> および <wait-seconds> タグに、エージェントへの接続試行回数と待ち時間を設定します。

6. 「WinActorクライアント設定ファイル」を保存します。

7. IM-JugglingでWARファイルを出力し、アプリケーションサーバへデプロイを行ってください。

> **コラム**
>
> エージェントを選出するエージェントセレクタに対して、各エージェント別にパラメータを与えることができます。
> ただし、エージェントセレクタの標準実装では、このパラメータは利用しておりません。
> エージェントセレクタを独自実装で差し替えた場合、必要に応じてパラメータを名前と値の組み合わせで記述可能です。
> <agent>タグ内に<param>タグ、および<param-name>、<param-value>タグの組み合わせで設定してください。

> **コラム**
>
> 2020 Winter(Azalea) 以降、上記エージェントの設定は、テナント環境セットアップ後、テナント管理者が画面から行うことも可能です。
> エージェントの追加・削除などが頻繁に発生する場合は、そちらを利用することを推奨します。
> 詳細については 「*WinActor連携 設定編集画面*」 を参照してください。
> 上記エージェントの設定手順は、 IM-Jugglingプロジェクト内に設定内容を保持する必要がある場合にご利用ください。

ライセンスコードの登録

IM-RPAを利用するためには、テナント環境セットアップ後、製品ライセンスコードの登録が必要です。

テナント環境セットアップについては、「intra-mart Accel Platform セットアップガイド」-「テナント環境セットアップ」を参照してください。

1. IM-RPAのライセンス登録を行います。
   詳細は「ライセンスの登録」を参照してください。

エージェントのダウンロード

WinActorエージェントは、Product File Download (https://product.intra-mart.jp/srcdownload/)よりダウンロードしてください。
ダウンロードには、IM-RPAのライセンスキーが必要です。
RPASDという文字列から始まるライセンスキーを入力することでWinActorエージェントのダウンロードリンクが表示されます。

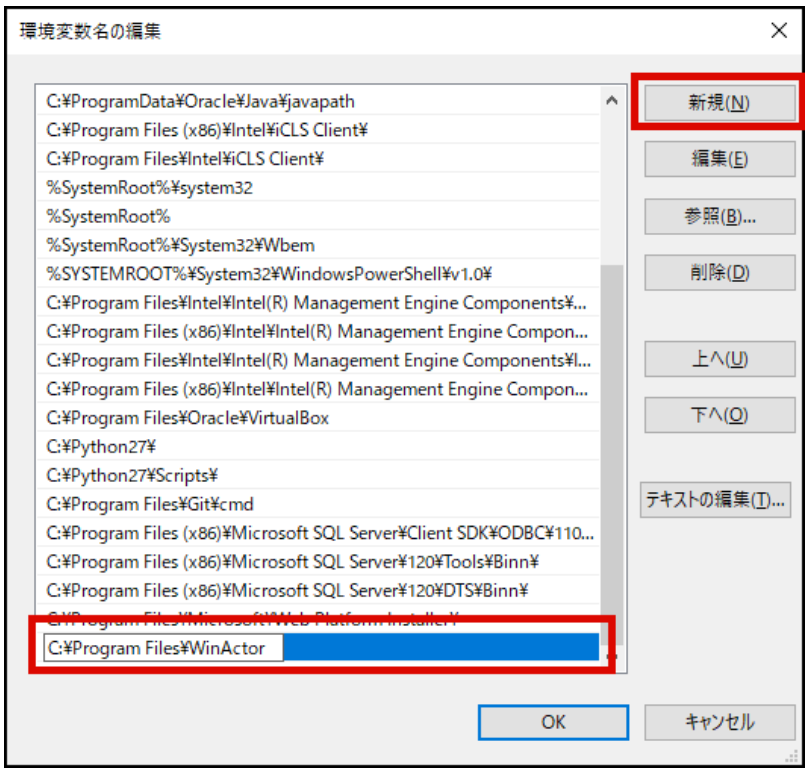
エージェントの起動

WinActorエージェントは、実行可能jar形式で同梱しています。
WinActorがインストールされている各端末にて、以下の手順を実施し、エージェントを起動してください。

1. Java 8 JDKをインストールします。

2. WinActorの実行ファイルが存在しているフォルダのパスを、システム環境変数 `PATH` に設定します。

   例） WinActorを `C:\Program Files\WinActor` にインストールした場合
   　　システム環境変数 `PATH` の編集画面にて、上記パスを新規追加します。



3. 「im_winactor_agent-8.x.x.jar」を任意のフォルダに配置します。

4. jarファイルを配置したフォルダにて、以下のコマンドを実行し、エージェントを起動します。

```
java -jar im_winactor_agent-8.x.x.jar
```

5. エージェントが起動すると、リクエスト受付の待機状態が始まります。
   アプリケーションサーバからのリクエストを受信すると、WinActorが自動的に起動され、シナリオが実行されます。

コラム

エージェントの起動時に、以下の表のシステムプロパティをJavaの引数として -Dxxx=xxx 形式で指定可能です。
次のコマンドのように、引数は -jar よりも前に指定してください。

```
java -Dserver.port=18088 -jar im_winactor_agent-8.x.x.jar
```

プロパティ名	デフォルト値	意味
server.port	8088	エージェントのポート番号
process.timeout.default	1800	シナリオ実行のタイムアウト時間のデフォルト値（秒）
path.exec.winactor	WinActor7.exe	WinActorの実行ファイル名
logging.file	winactor_agent.log	出力ログファイル名
logging.level.jp.co.intra_mart.foundation.winactor.AgentRestController	INFO	出力ログレベル
spring.servlet.multipart.max-file-size	200MB	シナリオファイル/データー覧ファイルの許容最大サイズ
spring.servlet.multipart.max-request-size	200MB	シナリオファイルなどを含む、エージェントへの通信の許容最大サイズ

設定ファイル

項目

概要

WinActor連携 に関する設定です。

モジュール	WinActor連携
フォーマットファイル(xsd)	WEB-INF/schema/winactor-config.xsd
設定場所	WEB-INF/conf/winactor-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<winactor-config
   xmlns="http://intra-mart.co.jp/winactor/winactor-config"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://intra-mart.co.jp/winactor/winactor-config ../schema/winactor-config.xsd">

 <agents>
  <agent name="agent1" endpoint="http://xxx.xxx.xxx.1:8088/winactor/agent/service">
   <param>
    <param-name>priority</param-name>
    <param-value>0</param-value>
   </param>
  </agent>
  <agent name="agent2" endpoint="http://xxx.xxx.xxx.2:8088/winactor/agent/service" />
 </agents>

 <groups>
  <group id="groupA">agent1</group>
  <group id="groupB">agent2</group>
  <group id="groupAll">agent2,agent1</group>
 </groups>

 <retry>
  <max-count>1</max-count>
  <wait-seconds>5</wait-seconds>
 </retry>

</winactor-config>
```

リファレンス

エージェント設定

タグ名　agents

接続先エージェント全てに関する設定を定義します。

【設定項目】

```
<winactor-config>
 <agents>
   :
 </agents>
</winactor-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	agents タグを親とするタグ
単位・型	なし
省略時のデフォルト値	なし
親タグ	winactor-config

エージェント個別設定

タグ名　agent

接続先エージェント一つに関する設定を定義します。

【設定項目】

```
<winactor-config>
 <agents>
  <agent name="agent1" endpoint="http://xxx.xxx.xxx.1:8088/winactor/agent/service">
   :
  </agent>
  <agent name="agent2" endpoint="http://xxx.xxx.xxx.2:8088/winactor/agent/service" />
 </agents>
</winactor-config>
```

必須項目	○
複数設定	○
設定値・設定する内容	agent タグを親とするタグ
単位・型	なし
省略時のデフォルト値	なし
親タグ	agents

【属性】

属性名	説明	必須	デフォルト値
name	エージェント名	○	なし
endpoint	接続先エンドポイントのURL	○	なし

パラメータ設定

タグ名　param

このエージェントに関するパラメータを一つ設定します。

【設定項目】

```
<winactor-config>
 <agents>
  <agent name="agent1" endpoint="http://xxx.xxx.xxx.1:8088/winactor/agent/service">
   <param>
   :
   </param>
  </agent>
 </agents>
</winactor-config>
```

必須項目	×
複数設定	○
設定値・設定する内容	param タグを親とするタグ
単位・型	なし
省略時のデフォルト値	なし
親タグ	agent

パラメータ名前設定

タグ名　param-name

このパラメータの名前を設定します。

【設定項目】

```
<winactor-config>
 <agents>
  <agent name="agent1" endpoint="http://xxx.xxx.xxx.1:8088/winactor/agent/service">
   <param>
    <param-name>priority</param-name>
   :
   </param>
  </agent>
 </agents>
</winactor-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	このパラメータの名前
単位・型	文字列
省略時のデフォルト値	なし
親タグ	param

パラメータ値設定

タグ名	param-value

このパラメータの値を設定します。

【設定項目】

```
<winactor-config>
 <agents>
  <agent name="agent1" endpoint="http://xxx.xxx.xxx.1:8088/winactor/agent/service">
   <param>
   :
    <param-value>0</param-value>
   </param>
  </agent>
 </agents>
</winactor-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	このパラメータの値
単位・型	文字列
省略時のデフォルト値	なし
親タグ	param

グループ設定

タグ名	groups

エージェントのグループ分けを設定します。

【設定項目】

```
<winactor-config>
 <groups>
   :
 </groups>
</winactor-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	groups タグを親とするタグ
単位・型	なし
省略時のデフォルト値	なし
親タグ	winactor-config

グループ個別設定

タグ名　group

グループの個別設定をします。

【設定項目】

```
<winactor-config>
 <groups>
   <group id="groupA">agent1</group>
   <group id="groupB">agent2</group>
   <group id="groupAll">agent2,agent1</group>
   :
 </groups>
</winactor-config>
```

必須項目	○
複数設定	◯
設定値・設定する内容	対象のエージェント名（agentで指定した名前） 複数指定する場合はカンマ区切りで指定。
単位・型	文字列
省略時のデフォルト値	なし
親タグ	groups

【属性】

属性名	説明	必須	デフォルト値
id	グループID	○	なし

リトライ設定

タグ名　retry

エージェントへの接続の際、設定したエージェント全てが実行中であったときのリトライに関する設定を定義します。

【設定項目】

```
<winactor-config>
 <retry>
   :
 </retry>
</winactor-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	retry タグを親とするタグ
単位・型	なし
省略時のデフォルト値	なし
親タグ	winactor-config

リトライ最大回数設定

タグ名　max-count

リトライの最大回数を設定します。
0を設定した場合、リトライを行いません。

【設定項目】

```
<winactor-config>
 <retry>
   <max-count>3</max-count>
   :
 </retry>
</winactor-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	リトライの最大回数
単位・型	整数値（0-）
省略時のデフォルト値	なし
親タグ	retry

リトライ待ち秒数設定

タグ名　wait-seconds

次にリトライするまでの待ち時間の秒数を設定します。

【設定項目】

```
<winactor-config>
 <retry>
   :
   <wait-seconds>10</wait-seconds>
 </retry>
</winactor-config>
```

必須項目	○

複数設定	×
設定値・設定する内容	リトライするまでの待ち時間の秒数
単位・型	整数値（0-）
省略時のデフォルト値	なし
親タグ	retry

IM-LogicDesigner タスク説明

項目

エージェント呼び出し

WinActorエージェントを呼び出し、シナリオを実行するタスクです。

エージェントセレクタの標準実装では、エージェントは登録されている中からランダムな順序で選択されます。
選択されたエージェントが現在シナリオ実行中であるならば、次のエージェントが選択されます。
すべてのエージェントがシナリオ実行中の場合は、設定ファイルにて指定した時間だけ待った後、最初からリトライを行います。
リトライ回数が設定された最大回数を超えた場合、エラーが発生します。

シナリオの実行結果として、更新されたデーター覧ファイルや、特定のパスのファイルを取得することが可能です。

> **コラム**
>
> ロジックフロー定義編集画面にて、WinActorのカテゴリおよびタスクのアイコンが表示されない場合、アイコンのリカバリを実行することで表示されるようになる場合があります。
> 詳細は、「IM-LogicDesigner ユーザ操作ガイド」 -「ユーザアイコンをリカバリする」を参照してください。

入力値

```
im_winactorCallAgent <object>
  ├─ dataFile <storage>
  ├─ groupId <string>
  ├─ resultFilePath <string>
  ├─ scenario <storage> *
  ├─ timeout <integer>
  └─ vmargs <string>
```

項目名	必須/任意	型	配列/リスト	説明
im_winactorCallAgent	任意	object	なし	-
dataFile	任意	storage	なし	シナリオにて利用するデーター覧ファイルを指定してください。
groupId	任意	string	なし	実行するロボットのグループIDを指定してください。
resultFilePath	任意	string	なし	シナリオによって出力された結果ファイルを取得したい場合に指定します。 エージェントのファイルシステム上の絶対パスを指定してください。
scenario	必須	storage	なし	実行するシナリオファイルを指定してください。
timeout	任意	integer	なし	シナリオ実行のタイムアウト時間を指定します。 （単位：秒） エージェントとの通信のタイムアウト時間ではありません。

項目名	必須/任意	型	配列/リスト	説明
vmargs	任意	string	なし	WinActorに与えるVM引数（-VM）の内容を指定します。 詳細については、WinActorのドキュメントに含まれる 「WinActor簡易マニュアル」を参照してください。

> **注意**
>
> エージェントを呼び出してシナリオを実行している間、アプリケーションサーバとエージェントとの間の通信のセッションは維持され続けている必要があります。
> 適切に `timeout` を設定し、通信のタイムアウトが発生する前にエージェントがレスポンスを返すようにしてください。

出力値

```
im_winactorCallAgent <object>
 ├─ dataFile <binary>
 ├─ resultFile <binary>
 ├─ stderrLog <binary>
 └─ stdoutLog <binary>
```

項目名	型	配列/リスト	説明
im_winactorCallAgent	object	なし	-
dataFile	binary	なし	シナリオによって更新されたデーター覧ファイル
resultFile	binary	なし	入力値 `resultFilePath` によって指定された結果ファイル
stderrLog	binary	なし	WinActorが標準エラーに出力した内容 （通常、何も出力されません）
stdoutLog	binary	なし	WinActorが標準出力に出力した内容 （通常、何も出力されません）

WinActor連携 設定編集画面

項目

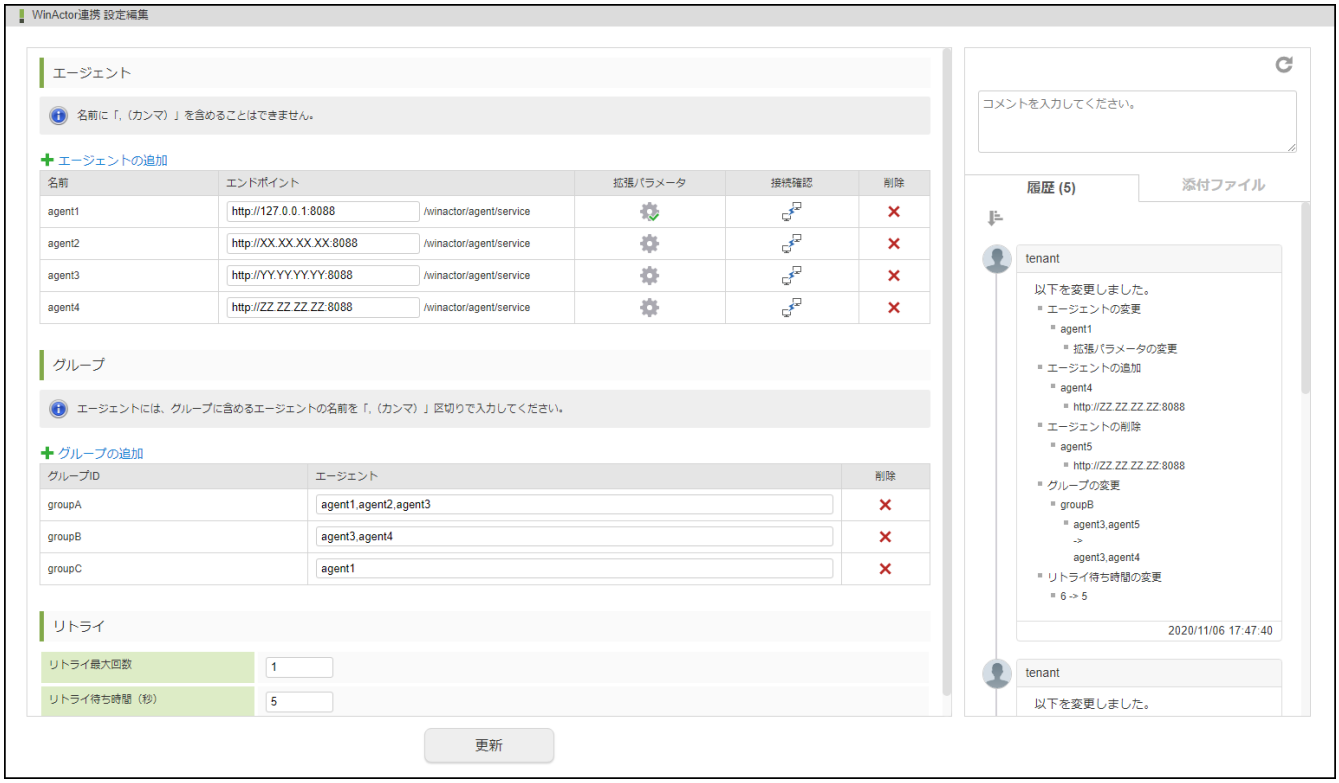
概要

WinActor連携に関する設定の編集を、テナント管理者がiAP画面から行うことができます。
これを利用することで、エージェントの構成に変更があった際に、設定ファイルを手作業で編集する必要がなくなります。
また、WARファイルの再デプロイを行わずとも、全アプリケーションサーバに対して即座に設定内容を反映させることができます。

> **コラム**
>
> WinActor連携 設定編集画面は、2020 Winter(Azalea)以降に利用が可能です。

設定編集画面

1. テナント管理者でログインします。
2. 「サイトマップ」→「RPA」→「WinActor連携設定」をクリックして「WinActor連携 設定編集」画面を開きます。



図：「WinActor連携 設定編集」画面

3. 当画面の入力項目を編集し、「更新」ボタンをクリックすることで、設定を保存できます。
   入力項目の詳細については、次節以降を参照してください。

> **コラム**
>
> 保存した設定内容は、設定ファイルとしてシステムストレージ内の以下のパスに出力され、全アプリケーションサーバ間で共有されます。
> `<ストレージルート>/system/storage/conf/winactor-config.xml`
> システムストレージに出力された設定ファイルは、`WEB-INF/conf/winactor-config.xml` よりも優先して利用されます。

エージェント

エージェントに関する設定を行います。

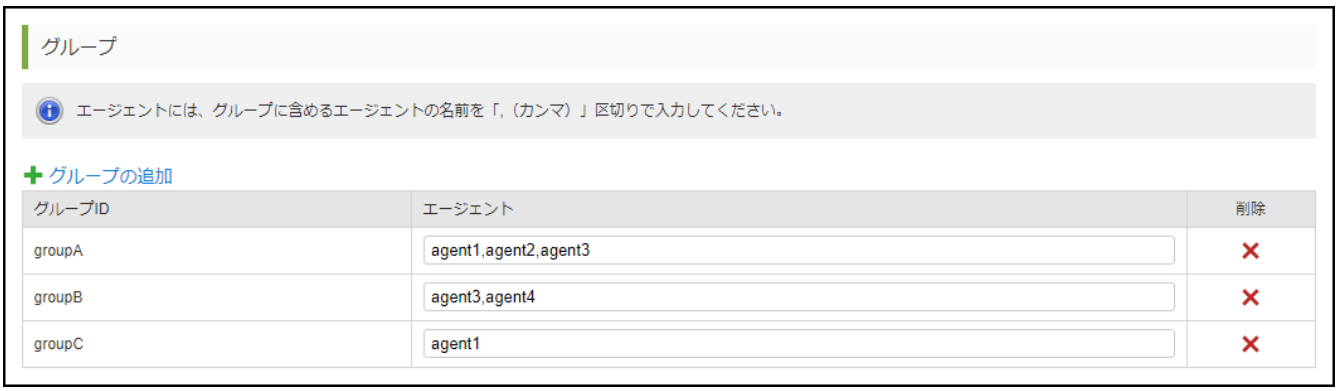
図：「WinActor連携 設定編集」画面 - 「エージェント」

項目名	説明
エージェントの追加	エージェントを1つ追加します。
名前	エージェントの名前です。新しく追加したエージェントに入力可能です。 空白ではない、ユニークなものを入力してください。 「,（カンマ）」を含めることはできません。

項目名	説明
エンドポイント	接続先エンドポイントのURLです。 パス /winactor/agent/service より前のホスト名/ポート番号を入力してください。
拡張パラメータ	クリックすることで拡張パラメータ編集ダイアログを開きます。 拡張パラメータが設定されているエージェントには設定済みのアイコンが表示されます。
接続確認	クリックすると入力済みのエンドポイントへ接続テストを行います。 アプリケーションサーバからエンドポイントのホストに通信が可能で、かつ、エンドポイントのURLでエージェントが起動している場合、 接続に成功しました。 と表示されます。 エンドポイントのホストに通信が行えない、または、エージェントが起動していない場合は、 接続に失敗しました。 と表示されます。
削除	このエージェントを削除します。

グループ

グループに関する設定を行います。
「エージェント呼び出し」タスクを実行する際、この設定に記述したグループIDを指定することで、呼び出し先のエージェントをグループ内のエージェントに限定できます。

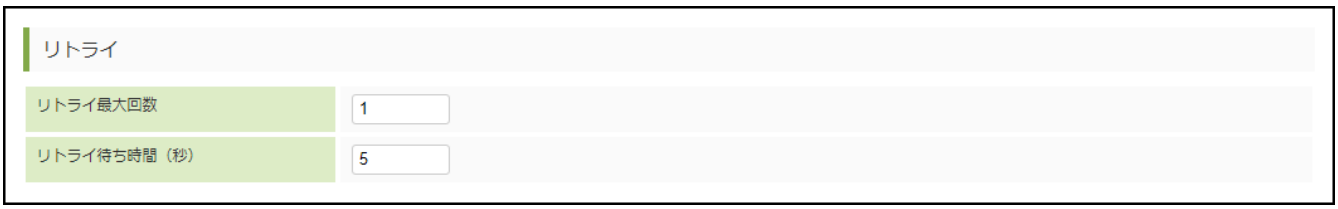


図：「WinActor連携 設定編集」画面 - 「グループ」

項目名	説明
グループの追加	グループを1つ追加します。
グループID	グループのIDです。新しく追加したグループに入力可能です。 空白ではない、ユニークなものを入力してください。
エージェント	このグループに含めるエージェントです。 複数のエージェントの名前を、 「,（カンマ）」区切りで入力してください。
削除	このグループを削除します。

リトライ

エージェントへの接続の際、設定したエージェント全てが実行中であったときのリトライに関する設定を行います。



図：「WinActor連携 設定編集」画面 - 「リトライ」

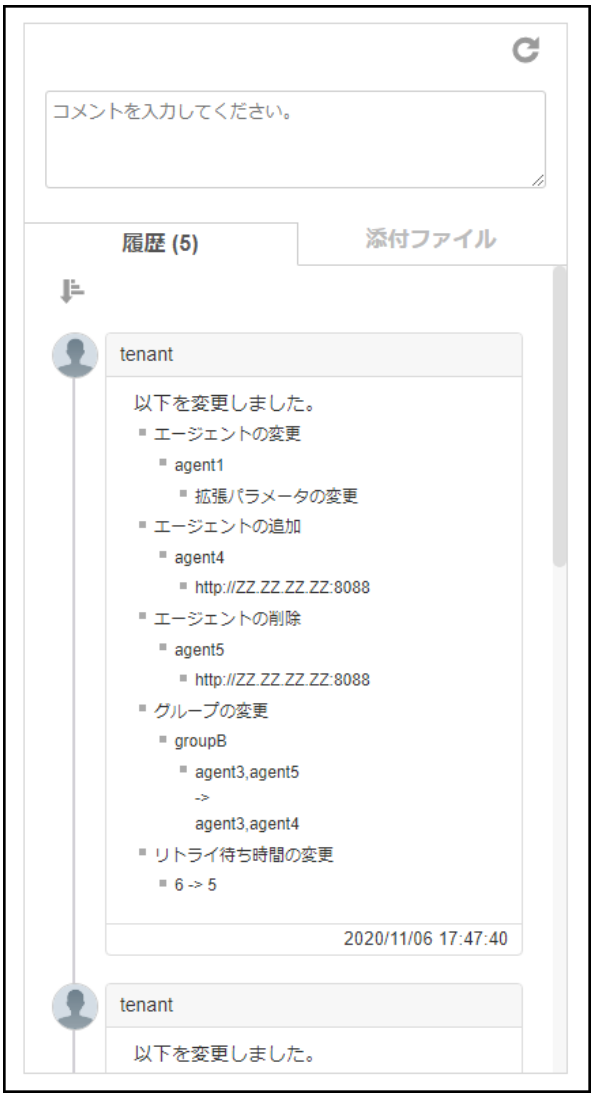
項目名	説明
リトライ最大回数	リトライの最大回数です。 0を設定した場合、リトライを行いません。

項目名	説明
リトライ待ち時間（秒）	リトライの待ち時間です。 次にリトライするまでの待ち時間の秒数を設定します。

履歴・コメント

設定内容の変更履歴、および、コメントが表示されます。

更新した設定内容に応じて、変更履歴は自動的に投稿されます。
任意のコメントを手動で書き込むことも可能です。



図：「WinActor連携 設定編集」画面 - 「履歴・コメント」

> **コラム**
>
> 履歴・コメント機能の詳細な操作方法については、「履歴・コメントモジュールユーザ操作ガイド」 を参照してください。

WinActor連携チュートリアル

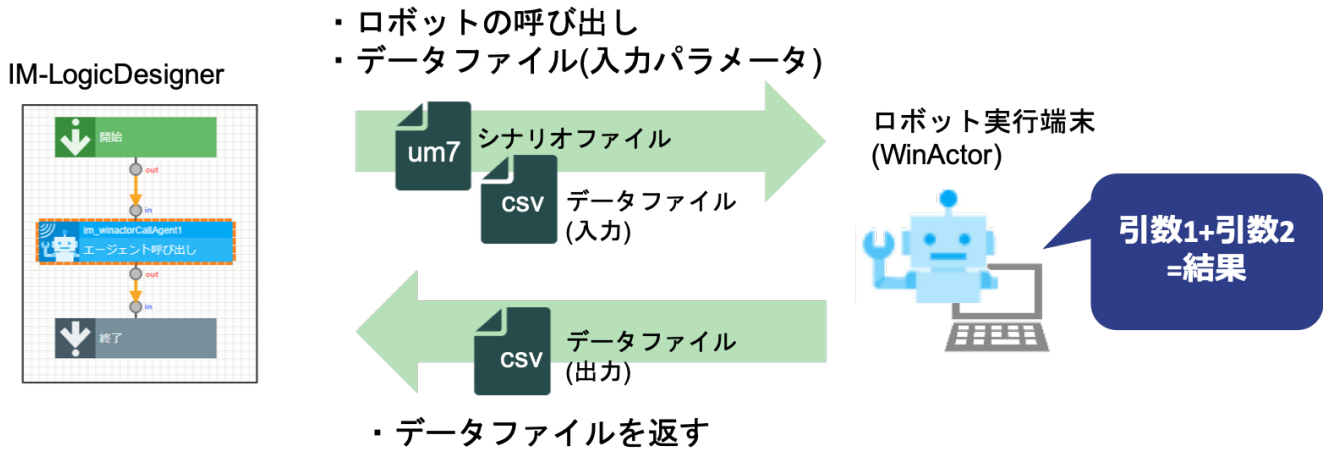
項目

チュートリアルの概要

本章では、IM-RPAのWinActor連携機能を使用して、 WinActorのロボットを実行する方法をチュートリアル形式でご説明します。
このチュートリアルに沿って設定を行うことで、以下のような機能を実現できます。

- intra-mart Accel PlatformからIM-LogicDesignerを介してWinActorのロボットを実行します。
- WinActorのロボットに対してパラメータを渡し、計算した結果を受け取ります。

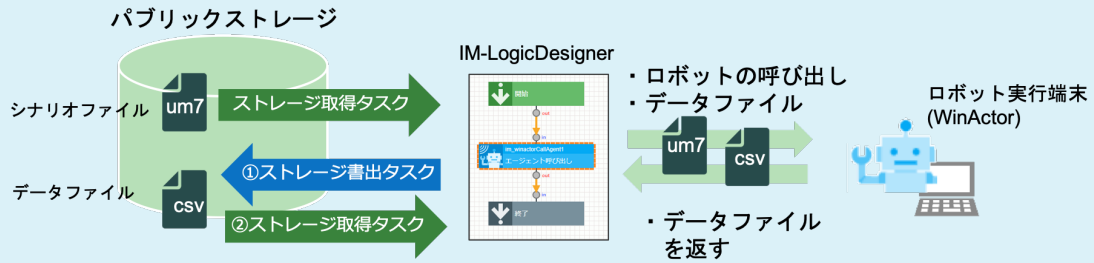
本章では、説明を簡単にするために、ロボットのシナリオはシンプルなものにしております。
WinActor連携においては、ロボット実行時に、intra-mart Accel Platformからロボット実行端末へシナリオファイルを送付します。
また、ロボットとのパラメータのやりとりには「データファイル」というCSVファイルを使用します。
シナリオの実行イメージは以下の通りです。



図：本チュートリアルシナリオ実行イメージ

コラム

シナリオ、およびデータファイルの扱いについて
WinActor連携に使用するシナリオファイル、およびデータファイルは、intra-mart Accel Platformのパブリックストレージ上から「ストレージ取得」タスク使用して取得する必要があります。
また、本チュートリアルでは、ロボットに動的なパラメータを連携するために、データファイルの書出を読込前に実行しています。

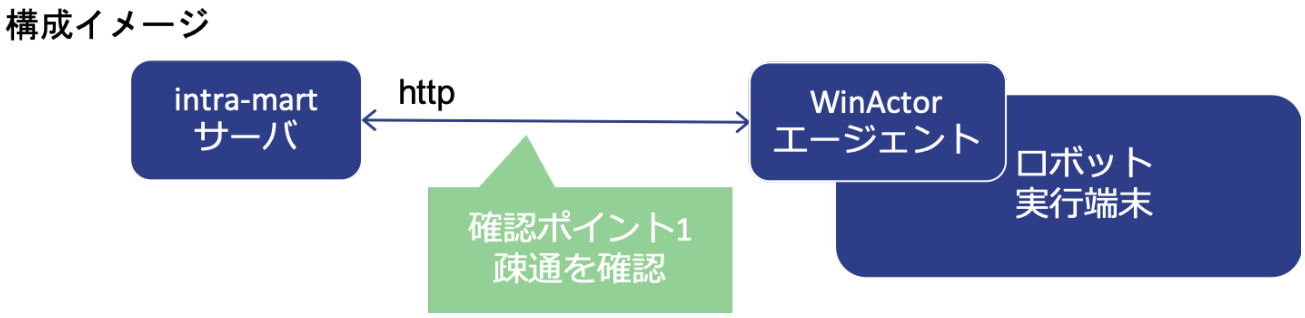


図：ストレージサービスへの書き込みについて

準備・環境設定

WinActor環境の確認

WinActorのロボット実行端末上にIM-RPAの提供しているWinActorエージェントをセットアップした環境を用意してください。
あらかじめ、 WinActorのロボット実行端末上で単体のロボットが実行できることをご確認ください。
WinActorのロボット実行端末はintra-martサーバとhttpポートで通信できる必要があります。



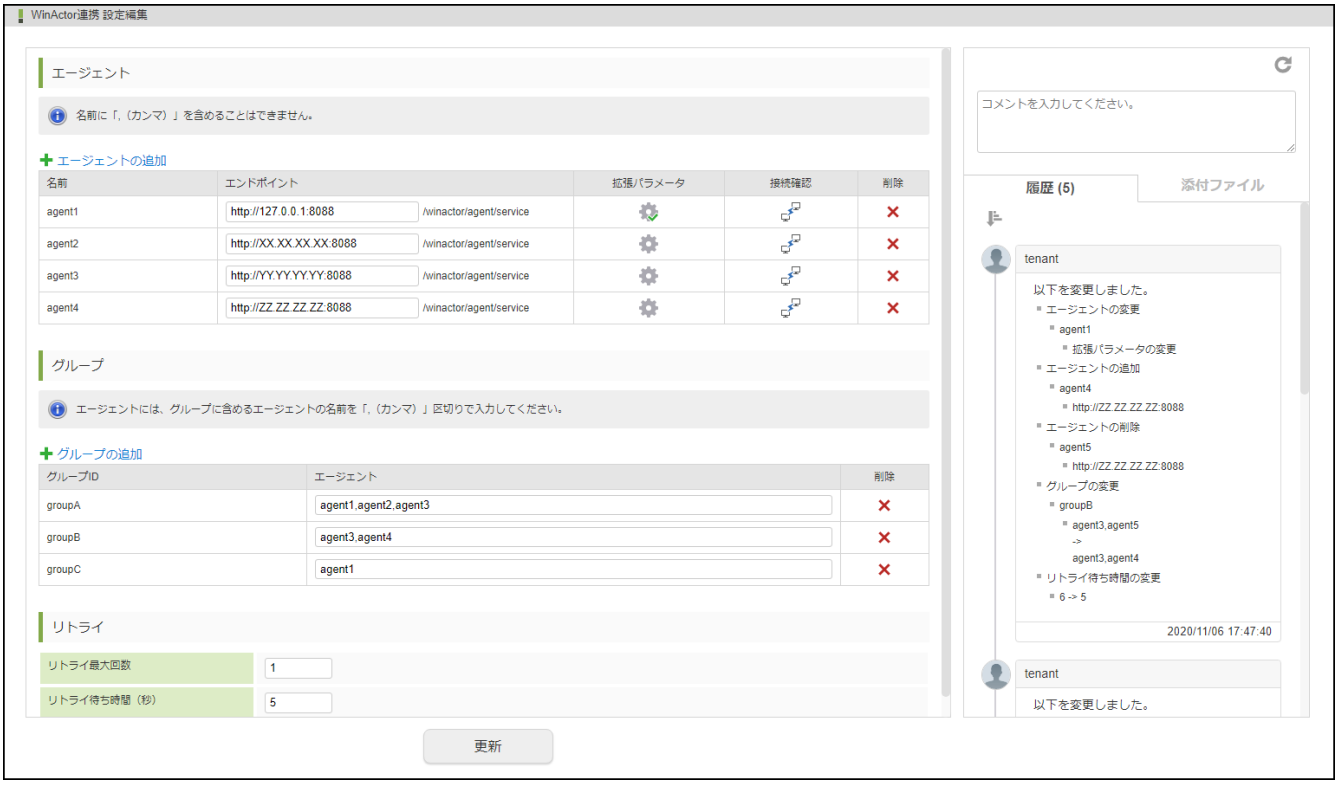
図：構成イメージ

コラム

WinActor連携のセットアップでは、intra-mart Accel Platformのセットアップ、 WinActorエージェント、および設定ファイルを記載します。
詳細は、「*セットアップ*」を参照してください。

WinActor設定の確認

「サイトマップ」→「RPA」→「WinActor連携設定」をクリックして「WinActor連携 設定編集」画面を開きます。
「接続確認」アイコンをクリックし、 接続に成功しました。 というメッセージが表示されることを確認します。



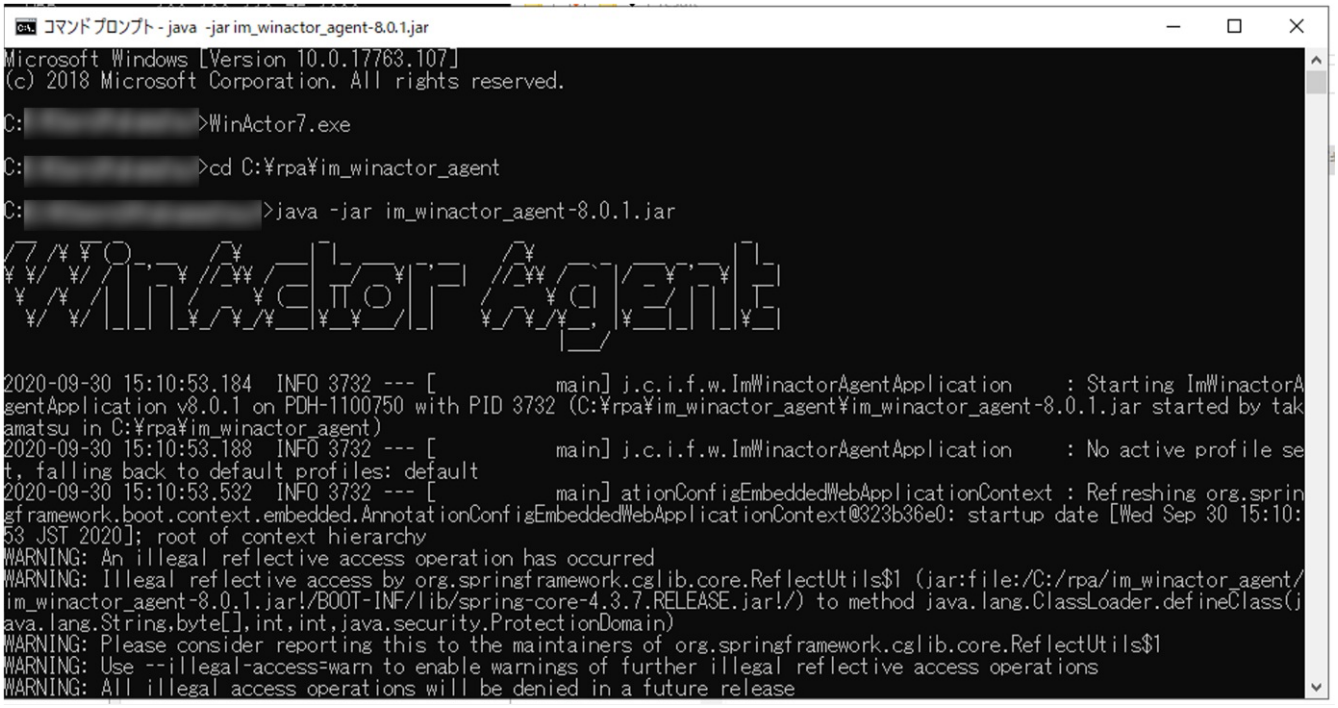
図：「WinActor連携 設定編集」画面

確認ポイント

よくある接続ミスを事前に確認するポイントを記載します。
確認ポイントのチェックが確認できない場合、WinActor連携の実行時にエラーが発生します。
「セットアップ」を再度確認し、セットアップを行ってください。

確認ポイント1

WinActorのロボット実行端末上でWinActorエージェントが起動していることを確認してください。



図：確認ポイント1

確認ポイント2

WinActorエージェントが指定したポート（デフォルト値は `8088` ）でLISTENING状態になっていることを確認してください。

図：確認ポイント2

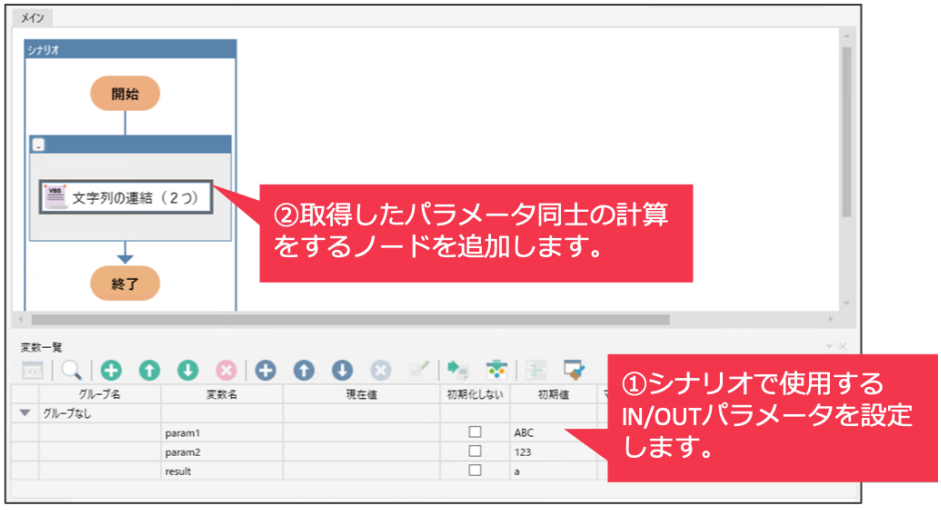
確認ポイント3

WinActorエージェントが指定したプログラム名（デフォルト値は WinActor7.exe ）で、パスが通っていることを確認してください。

図：確認ポイント3

シナリオの作成

WinActorにてロボットのシナリオを作成します。
以下は、ロボットのシナリオ例です。



図：WinActorシナリオ設定例

> **コラム**
>
> WinActorを使用したロボットの作成方法については、 詳細は、「WinActorマニュアルページ」を参照してください。

パラメータの設定

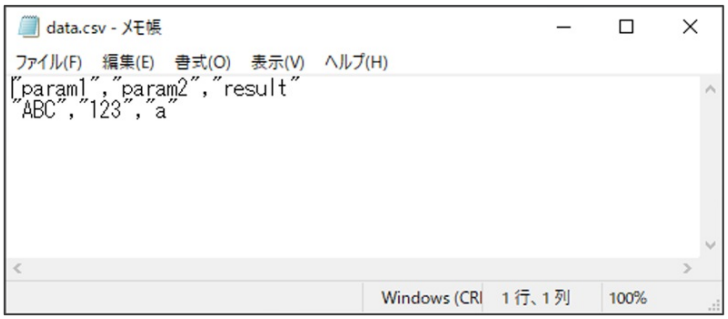
上記「①シナリオで使用するIN/OUTパラメータを設定します。」部分でパラメータ設定を行います。
変数にIN/OUTの区別はありません。

図：WinActorパラメータの設定

> **コラム**
>
> WinActorでは変数は全て文字列型として扱われます。
> なお、IM-LogicDesigner側でパラメータを受け取る際に別のデータ型を指定した場合、自動的に型変換をします。
> 具体的な対応表は、「IM-LogicDesigner仕様書」 - 「IM-LogicDesigner データ型変換 仕様書」を参照してください。

データファイルの用意

前項で設定した変数について、下記のように相対するデータファイルを用意します。



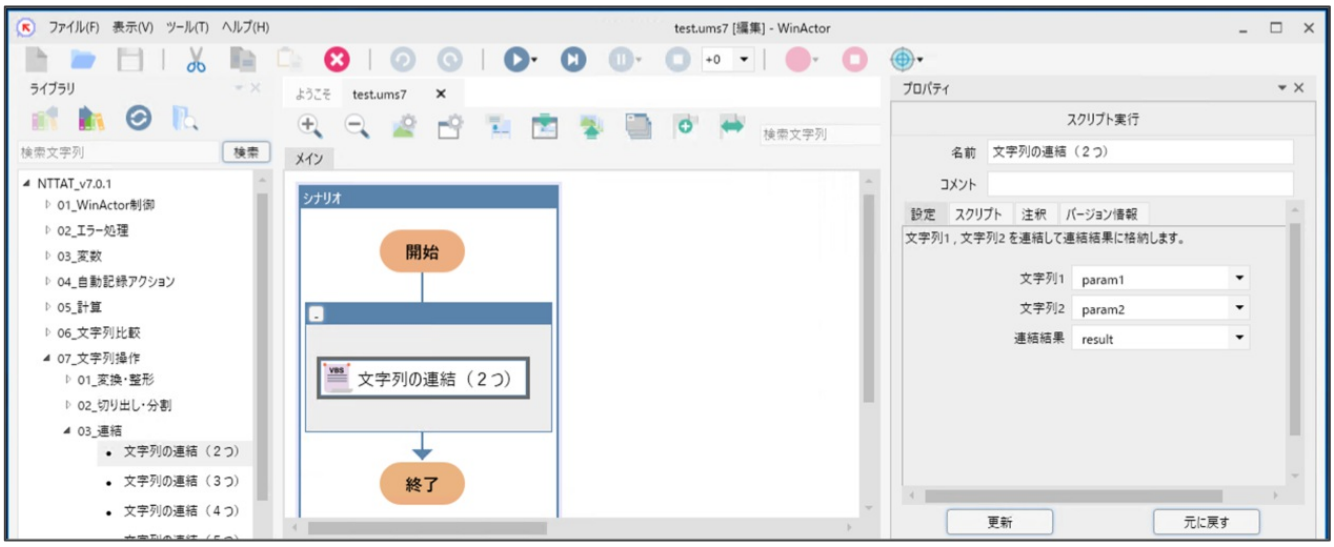
図：WinActorデータファイルの用意

作成したデータファイルについて、「変数名インポート」をして同じ結果となることを確認してください。

図：WinActor「変数名インポート」

ノードの設定

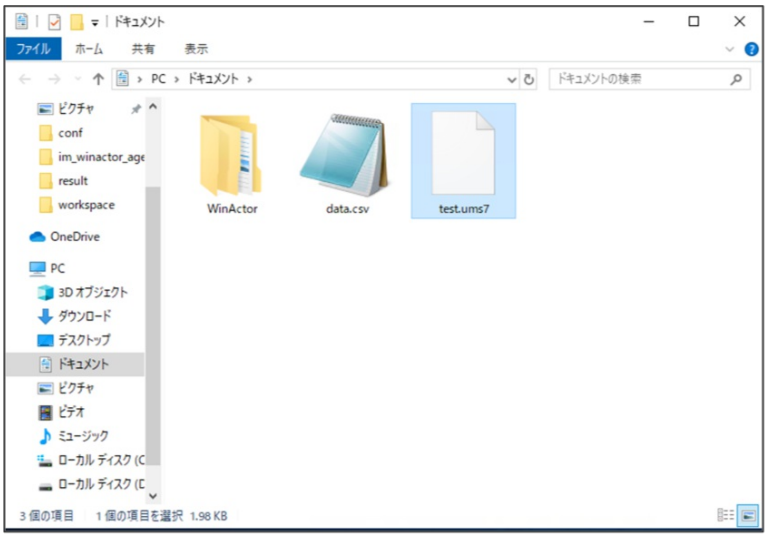
上記「②取得したパラメータ同士の計算をするノードを追加します。」部分でノードの設定を行います。
「ライブラリ」より「文字列の連携（2つ）」を選択し、ノードに追加します。



図：WinActor「文字列の連携（2つ）」をノードに追加

シナリオの保存

作成したシナリオを保存します。
シナリオファイルとデータファイルは後ほど使用するため、保存場所をあらかじめ確認してください。



図：シナリオファイルとデータファイルを保存

WinActorシナリオの作成から保存までの具体的な操作の流れについては、動画にてご覧いただけます。

シナリオのアップロード

WinActorにて作成したシナリオファイル、およびデータファイルをIM-RPAのパブリックストレージにアップロードします。
ここでは、テナント管理者機能の「ファイル操作」を利用した方法を説明します。
ファイル操作の利用方法については、「テナント管理者操作ガイド」 - 「ファイル操作を使用する」を参照してください。

ファイル操作にて、「default」ディレクトリの配下に任意のディレクトリを作成します。
ここではディレクト名を「RPA」とします。
作成したディレクトリ配下に、シナリオファイル、およびデータファイルをアップロードします。

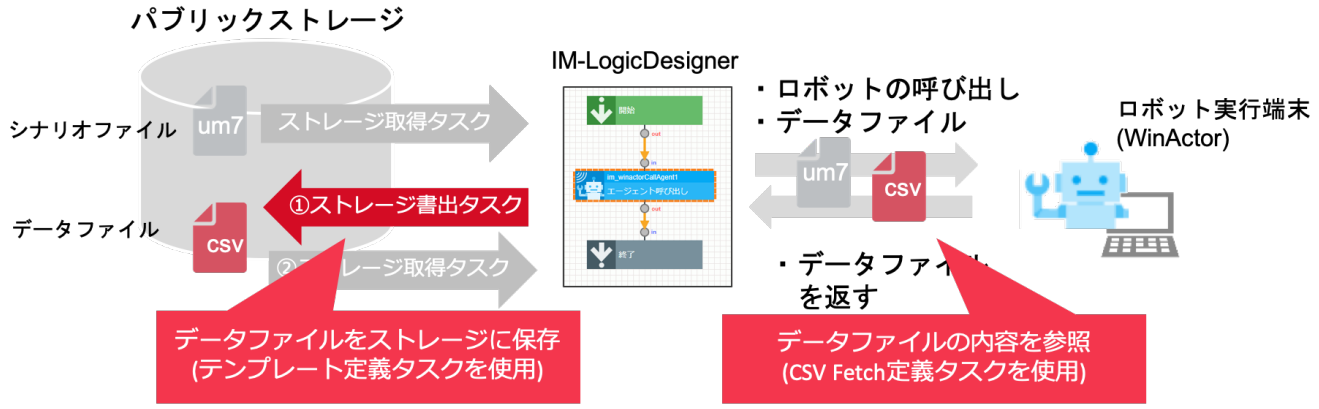
図：「ファイル操作」を使用してシナリオをアップロード

アップロードの具体的な操作の流れについては、動画にてご覧いただけます。

IM-LogicDesignerユーザ定義タスクの準備

IM-LogicDesignerユーザ定義タスクとは、IM-LogicDesignerにおいて作成可能な、各種機能を持つ独自タスクです。

本チュートリアルにおいては、IM-LogicDesignerで作成したWinActor用のデータファイルをintra-mart Accel Platformのストレージに保存する時と、ロボットから返却されたデータファイルを参照する際に、この独自タスクを使用します。



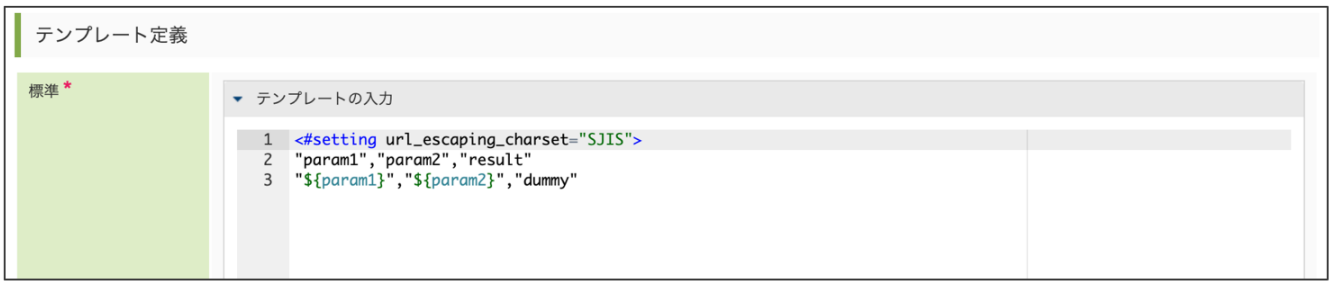
図：ユーザ定義タスクの使用箇所

ユーザ定義タスクの具体的な説明については、「IM-LogicDesigner仕様書」 - 「ユーザ定義タスク」を参照してください。
本チュートリアルでは、 2つのユーザ定義タスクを用意いたします。

ユーザタスク種別	内容
テンプレート定義	IM-LogicDesignerへの入力値を使って、データファイルの内容を作成します。 データファイルは、WinActor連携においてロボットとパラメータの入出力を行うためのCSVファイルです。
CSV Fetch定義	WinActorのロボット実行後、返却されたデータファイルの内容をIM-LogicDesignerに読み込みます。

ユーザ定義作成（テンプレート定義）

「サイトマップ」→「LogicDesigner」→「ユーザ定義」→「テンプレート定義新規作成」をクリックします。
テンプレート定義を下記のように編集します。
入力パラメータで動的にWinActorに渡すデータファイルのテキストを作成します。



図：「テンプレート定義新規作成」 - 「標準」

テンプレート入力例は以下の通りです。

```
<#setting url_escaping_charset="SJIS">
"param1","param2","result"
"${param1}","${param2}",”dummy”
```

ユーザ定義タスク（テンプレート定義）の具体的な説明については、「IM-LogicDesigner仕様書」 - 「ユーザ定義タスク-テンプレート」を参照してください。

返却値について、今回使用するデータファイルに合わせてキー項目を設定してください。

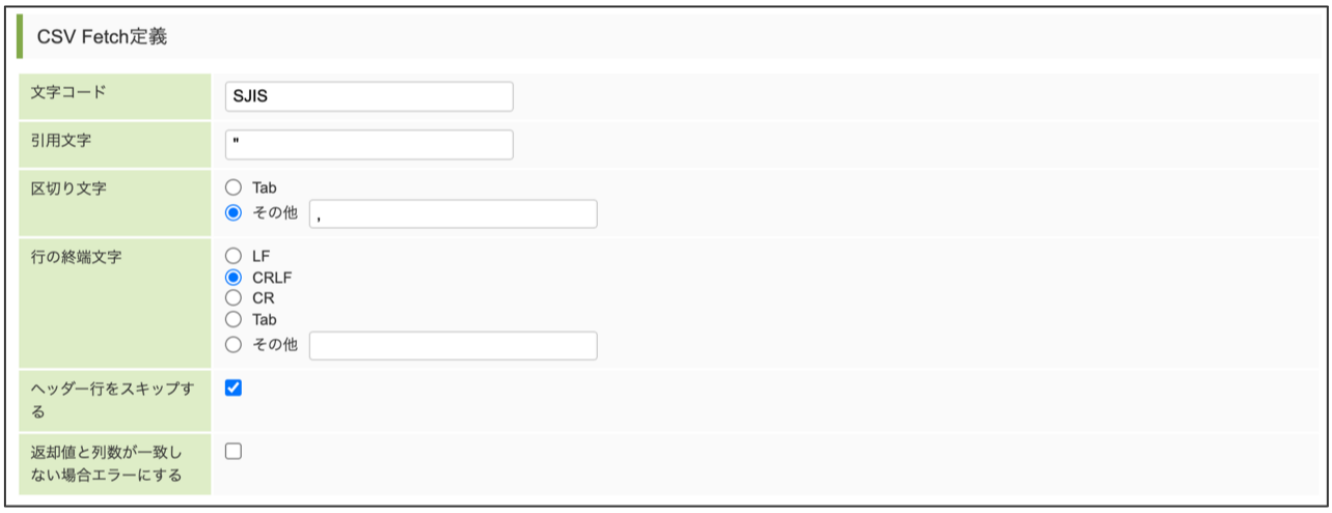
図：「テンプレート定義新規作成」 - 「入力値/出力値」

「ユーザ定義名」など適宜設定し、テンプレート定義を完了します。

> **注意**
>
> 記載のコードはサンプルです。
> 本チュートリアルのシナリオ上では動作しますが、最低限の記載となっており、文字列内の改行などに対応しておりません。
> 実際に運用する際には、要件に合わせて必要な表現を追加してください。

ユーザ定義作成（CSV Fetch定義）

「サイトマップ」→「LogicDesigner」→「ユーザ定義」→「CSV Fetch定義新規作成」をクリックします。
今回使用するデータファイルの形式に合わせて「CSV Fetch定義」カテゴリを設定します。



図：「CSV Fetch定義新規作成」 - 「CSV Fetch定義」

ユーザ定義タスク（テンプレート定義）の具体的な説明については、「IM-LogicDesigner仕様書」 - 「ユーザ定義タスク-CSV Fetch」を参照してください。

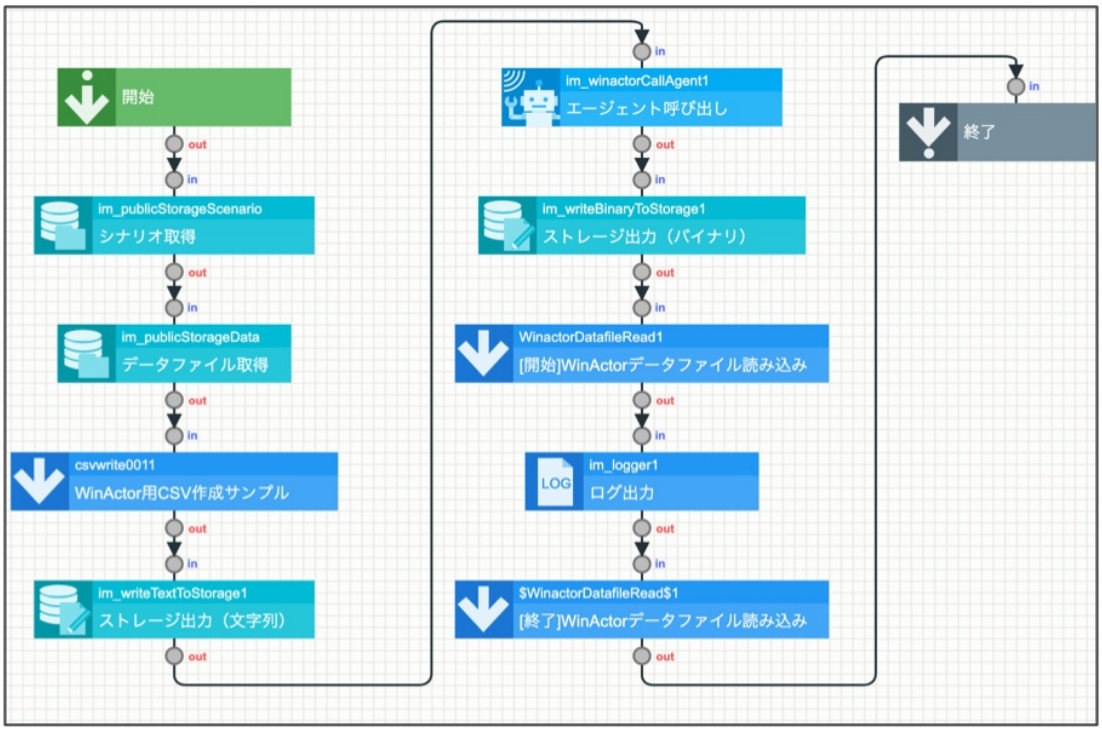
返却値について、今回使用するデータファイルに合わせてキー項目を設定してください。

図：「CSV Fetch定義新規作成」 - 「入力値/出力値」

「ユーザ定義名」など適宜設定し、CSV Fetch定義を完了します。

LDフローの呼び出し

「サイトマップ」→「LogicDesigner」→「フロー定義一覧」→「新規作成」をクリックします。
以下は、WinActor連携（パラメータ使用）を行うための最もシンプルなLDフローです。



図：完成イメージ（ロジックフロー）

具体的な手順は以下の通りです。

項目

- タスクとフローを設定します
- ロボットに連携するパラメータを設定します
- データマッピングをします
- デバッグ実行で動作を確認します

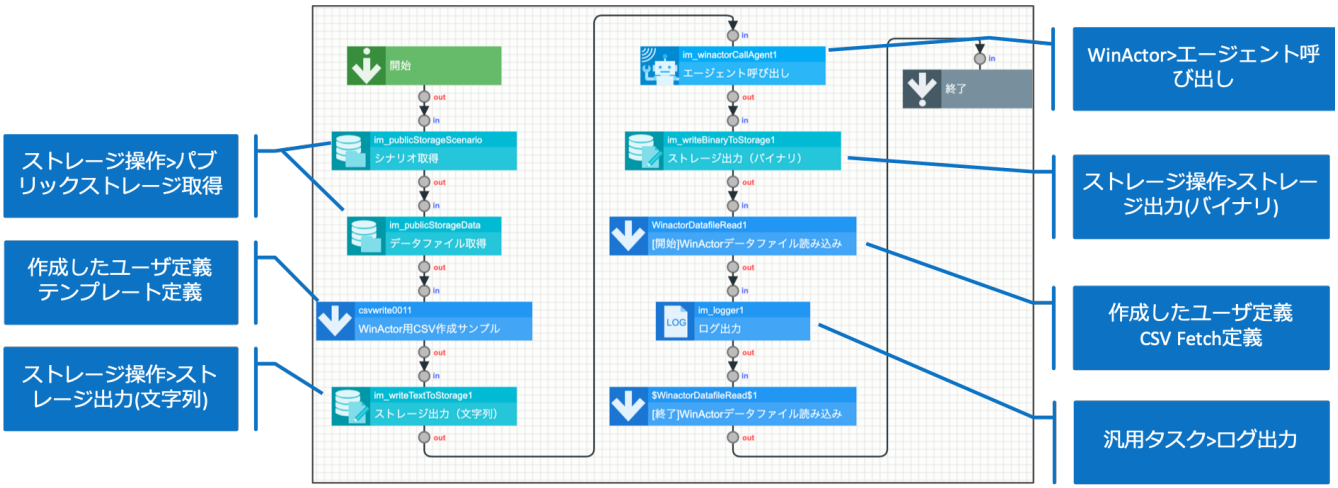
タスクとフローを設定します

パレットから各タスクを選択し、今回使用するタスクをキャンパスに配置します。

前項で作成したユーザ定義は、パレット内の「前項で設定したユーザカテゴリ>作成したユーザ定義名」にあります。

図：IM-LogicDesignerタスク選択

配置する各タスクは、以下の通りです。



図：IM-LogicDesigner配置タスク

次に、配置したタスク同士をフローで繋ぎます。

ロボットに連携するパラメータを設定します

IM-LogicDesignerの「入出力設定」を行います。

設定値の説明（入出力設定）

カテゴリ	設定値	説明
入力	param1	ロボットに渡す引数1
入力	param2	ロボットに渡す引数2
出力	result	ロボットからの返り値

図：IM-LogicDesigner- 「入出力設定」

次に、IM-LogicDesignerの「定数設定」を行います。

設定値の説明（定数設定）

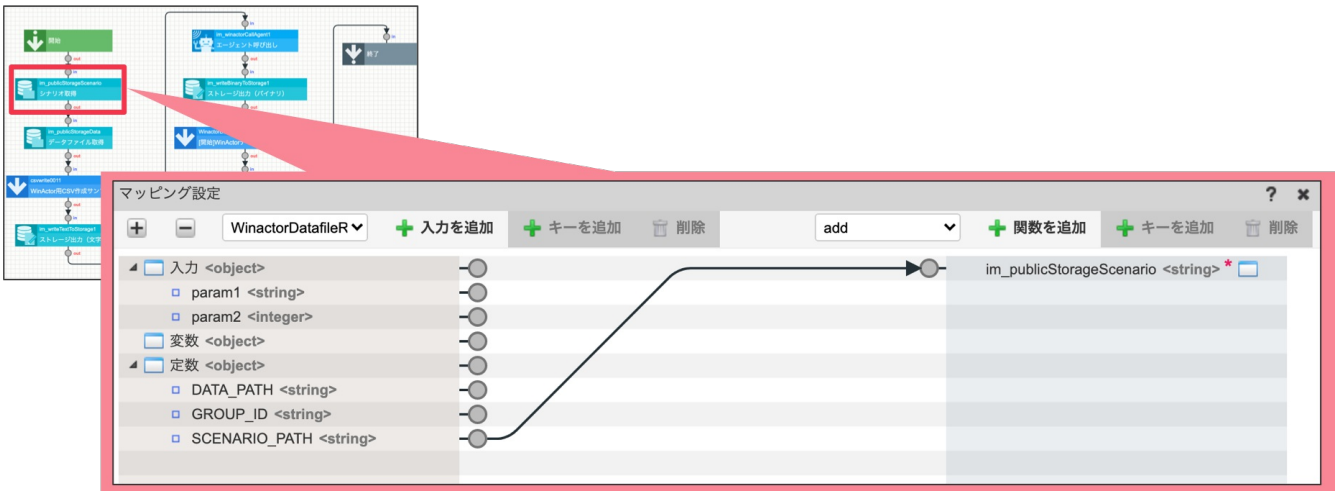
カテゴリ	設定値	説明
定数	DATA_PATH	パブリックストレージ上のデータファイルのパス
定数	GROUP_ID	呼び出すエージェントのグループID
定数	SCENARIO_PATH	パブリックストレージ上のシナリオファイルのパス

図：IM-LogicDesigner- 「定数設定」

データマッピングをします

各タスクをダブルクリックして、「データマッピング」の設定を行います。
各図を参考にマッピングをしてください。

シナリオ取得タスク（「ストレージ操作」-「パブリックストレージ取得」）のデータマッピングを行います。



図：データマッピングの設定 - シナリオ取得タスク

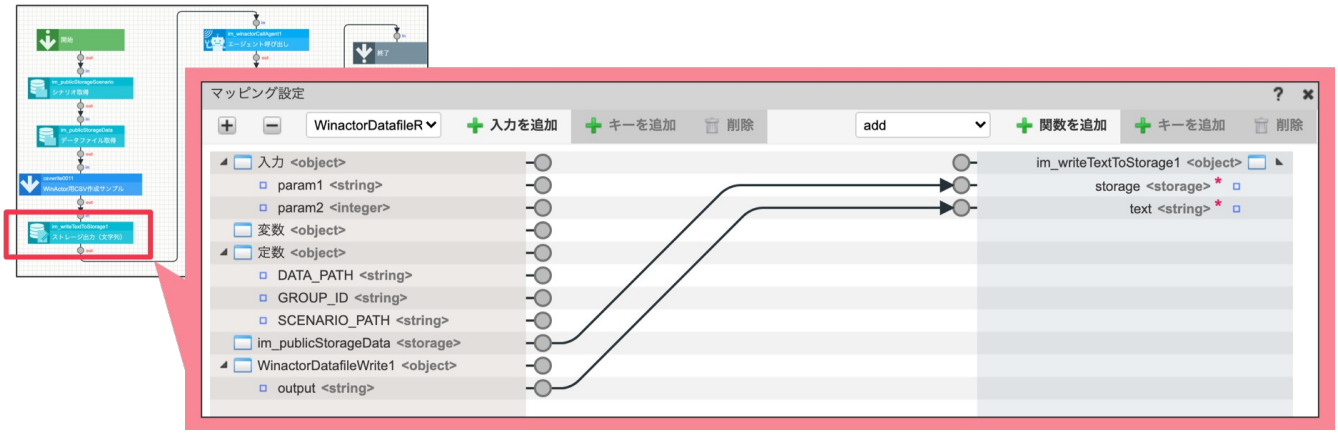
データファイル取得タスク（「ストレージ操作」-「パブリックストレージ取得」）のデータマッピングを行います。

図：データマッピングの設定 - データファイル取得タスク

データファイル作成タスク（作成したユーザ定義テンプレート）のデータマッピングを行います。

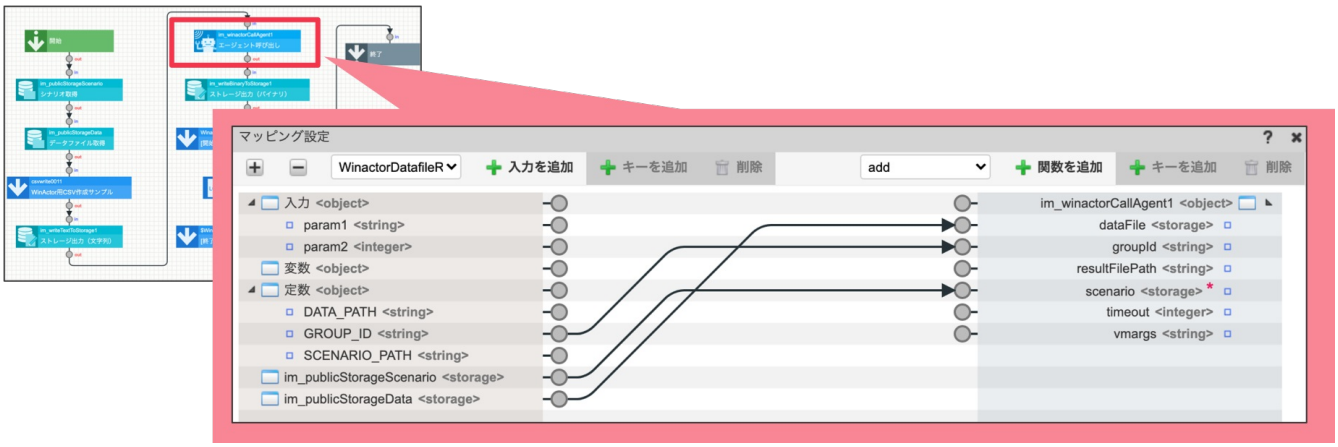
図：データマッピングの設定 - データファイル作成タスク

ストレージ出力（「ストレージ操作」 - 「ストレージ出力（文字列）」）のデータマッピングを行います。



図：データマッピングの設定 - ストレージ出力

エージェント呼び出し（「WinActor」 - 「エージェント呼び出し」）のデータマッピングを行います。

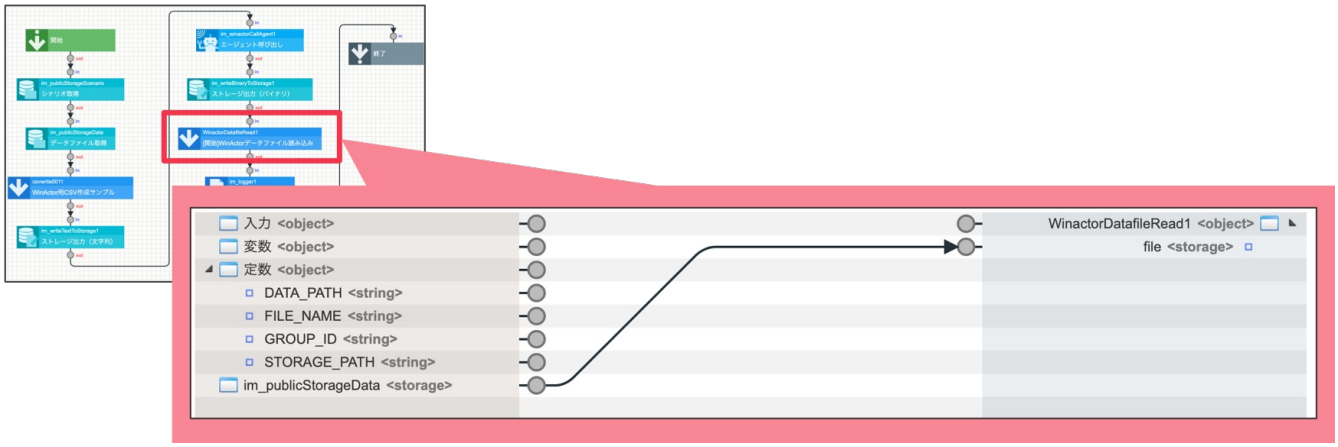


図：データマッピングの設定 - エージェント呼び出し

ストレージ出力（「ストレージ操作」 - 「ストレージ出力（バイナリ）」）のデータマッピングを行います。

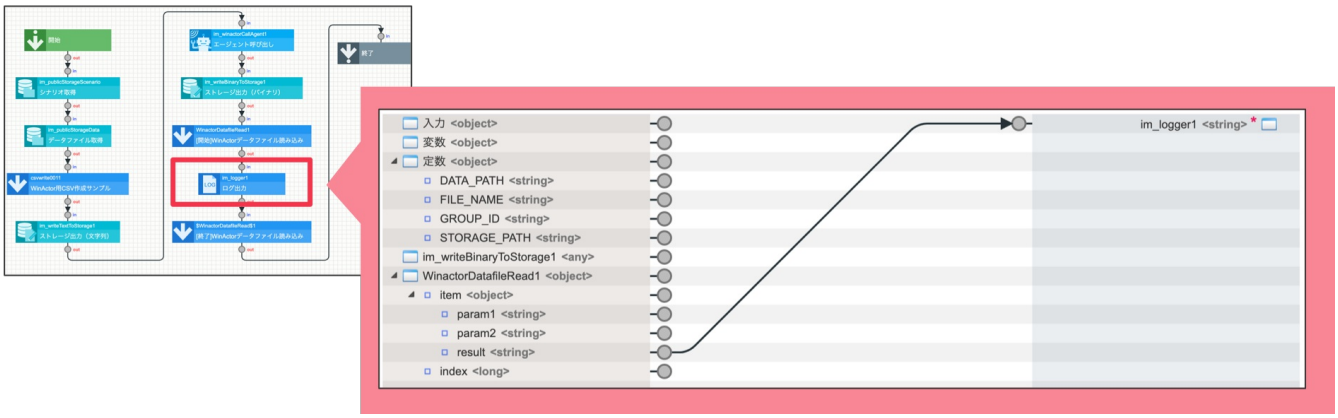
図：データマッピングの設定 - ストレージ出力

データファイル読込タスク「作成したユーザ定義 CSV Fetch定義」のデータマッピングを行います。



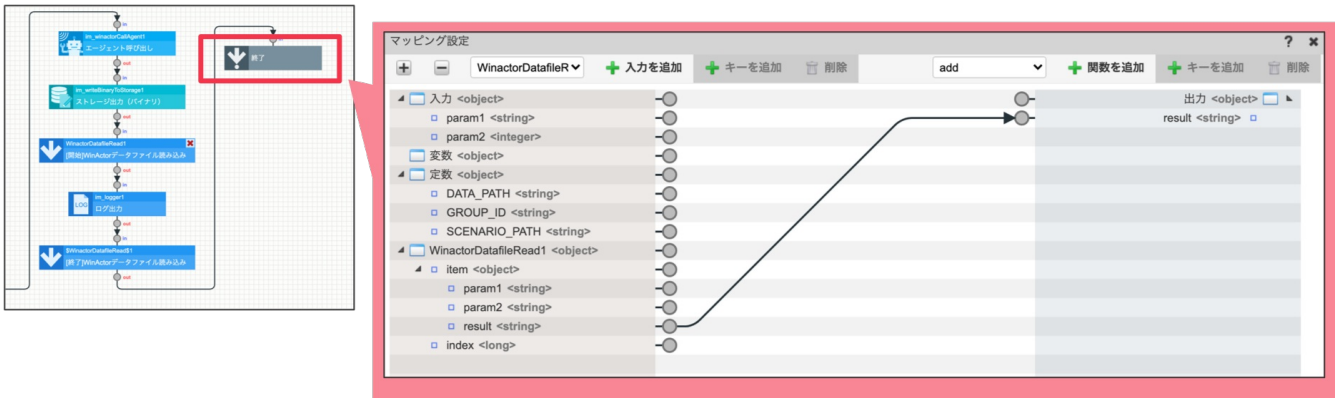
図：データマッピングの設定 - データファイル読込タスク

ログ出力（「汎用タスク」 - 「ログ出力」）のデータマッピングを行います。



図：データマッピングの設定 - ログ出力

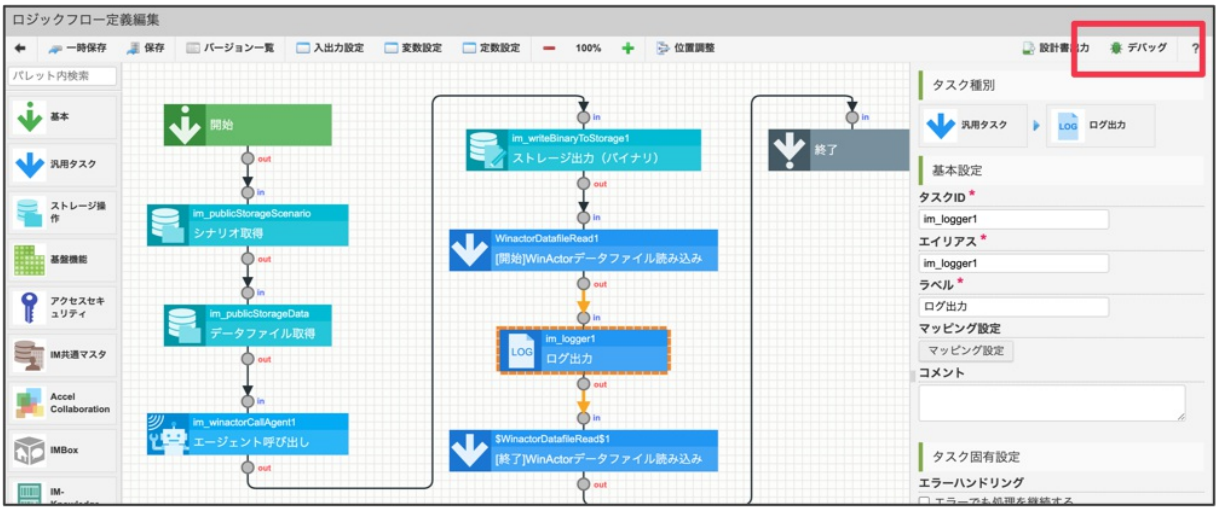
ロボットの実行結果を、フローの出力用変数にマッピングします。



図：データマッピングの設定 - 終了タスク

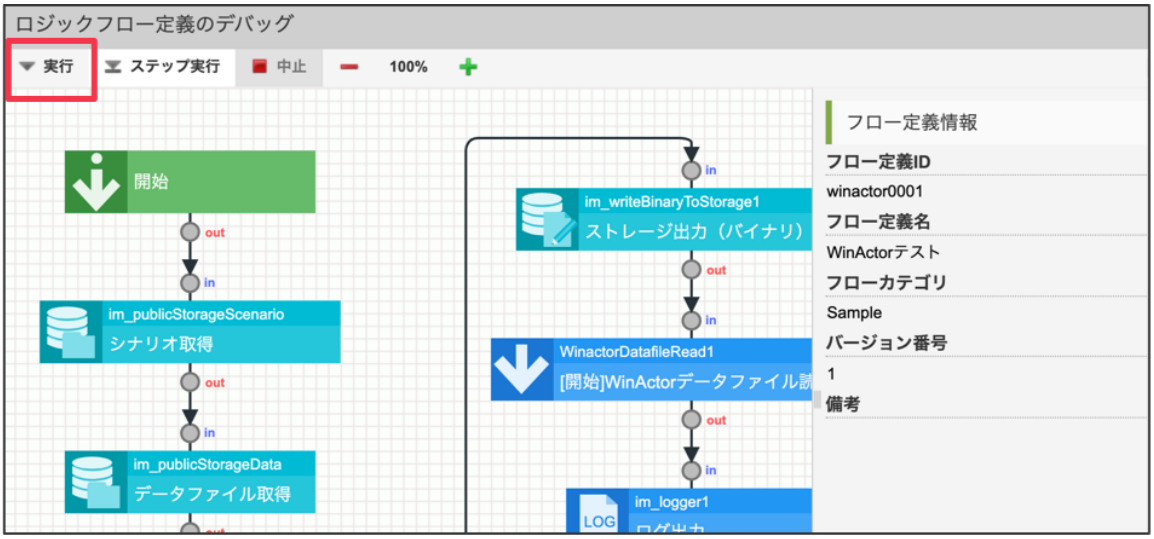
デバッグ実行で動作を確認します

ロジックフロー定義編集画面の「デバッグ」をクリックし、作成しているロジックフローをデバッグ実行します。



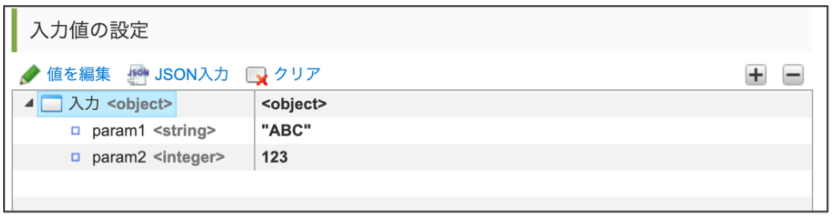
図：IM-LogicDesigner- 「デバッグ実行」

デバッグ画面にて、「実行」をクリックします。



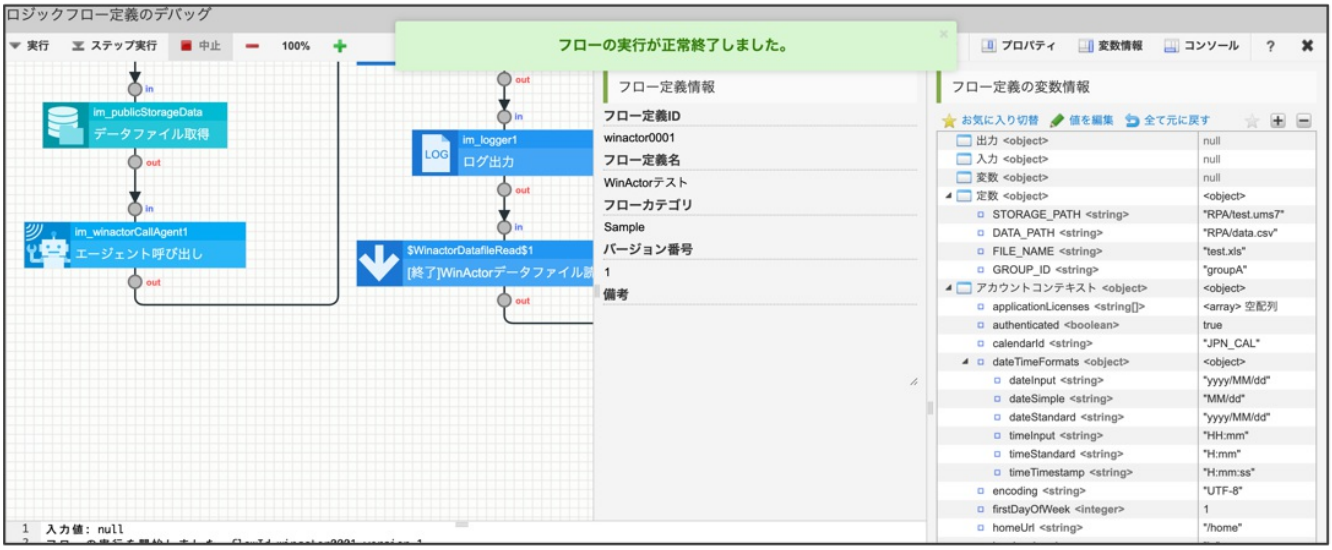
図：IM-LogicDesigner- 「デバッグ実行」 - 「実行」

デバッグ実行時の入力値の設定を行います。ロボットに引き渡すパラメータとなる、param1とparam2を設定してください。



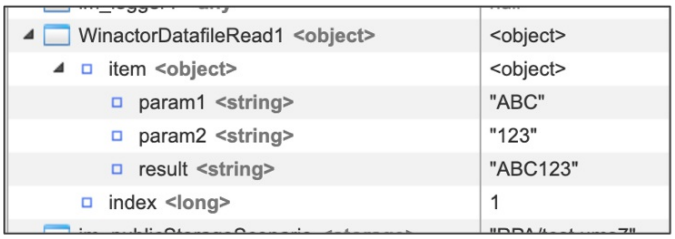
図：IM-LogicDesigner - 「デバッグ実行」 - 「実行」- 「入出力値の設定」

デバッグ実行をし、 WinActor連携の入出力が正しく行えていることを確認します。



図：IM-LogicDesigner- デバッグ実行実行の成功

ロボットからの返却値となる、resultが正しく表示されていることを確認します。

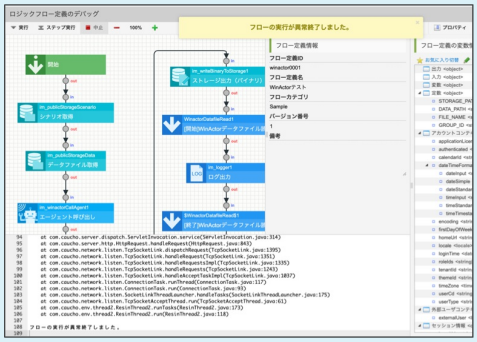


図：IM-LogicDesigner- 「フローの変数情報（デバッグ実行後）」

> **コラム**
>
> デバッグ実行時にエラーが発生した場合、デバッグ実行を「ステップ実行」にすることで、フローのどこでエラーとなったか確認することができます。

コラム

エラーが発生した場合、WinActorエージェント、WinActor連携設定を確認します。
[よくあるエラー]
・WinActor7.exeのパスは通っていますか？
・WinActor端末で、すでにWinActorが起動していませんか？

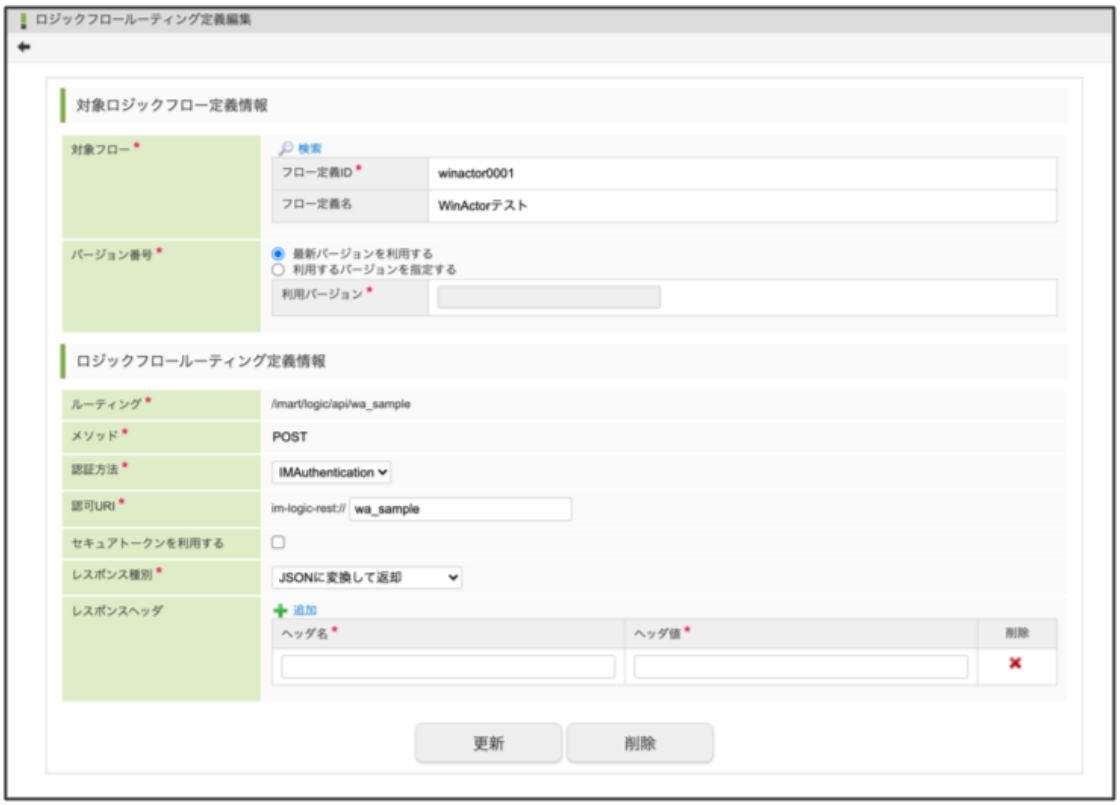


図：IM-LogicDesigner-「デバッグ実行」 - エラー例

入力値: null
フローの実行を開始しました。flowId=winactor0001,version=1
パブリックストレージ取得 タスクを実行します。
executeId=im_publicStorageScenario,taskId=ApplicationElementKey（elementId=im_publicStorage）
パブリックストレージ取得 タスクを実行します。
executeId=im_publicStorageData,taskId=ApplicationElementKey（elementId=im_publicStorage）
エージェント呼び出し タスクを実行します。
executeId=im_winactorCallAgent1,taskId=ApplicationElementKey（elementId=im_winactorCallAgent）
jp.co.intra_mart.foundation.logic.exception.FlowExecutionException: [E.IWP.WINACTOR.TASK.00008] エージェントがエラーレスポンスを返しました。 title="Abnormal termination", detail="Process terminated with code 1056"

フロールーティングの設定

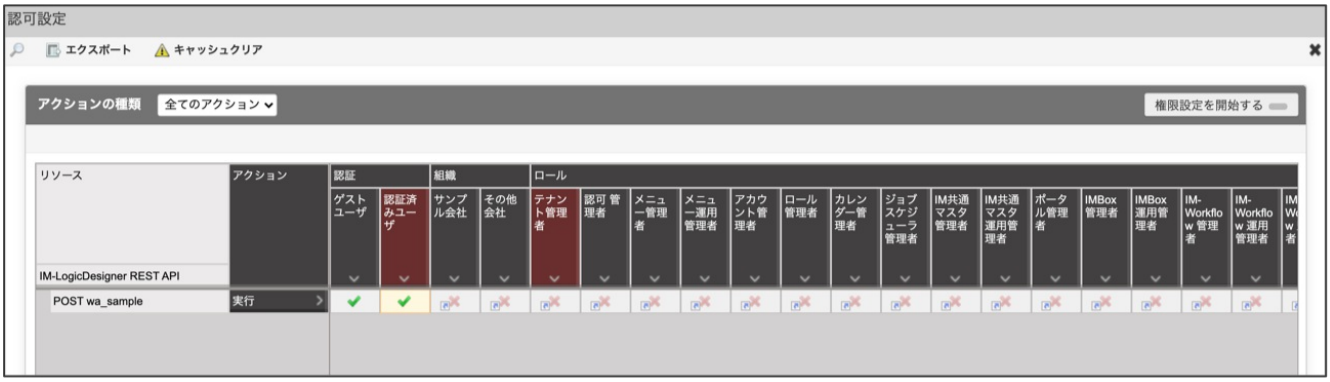
作成したIM-LogicDesignerのフローを、intra-mart Accel Platform他機能やREST経由で実行するため、フロールーティングの設定を行います。
「サイトマップ」→「LogicDesigner」→「ルーティング定義一覧」→「新規作成」をクリックします。
フロールーティングの設定については、「IM-LogicDesigner仕様書」 - 「フロールーティングの認可設定」を参照してください。



図：フロールーティングの設定

> コラム
>
> メソッドには「POST」を選択してください。

ロジックフロールーティング定義一覧より「認可」をクリックして、認可設定を行います。
「認証済みユーザ」へ実行権限を付与します。



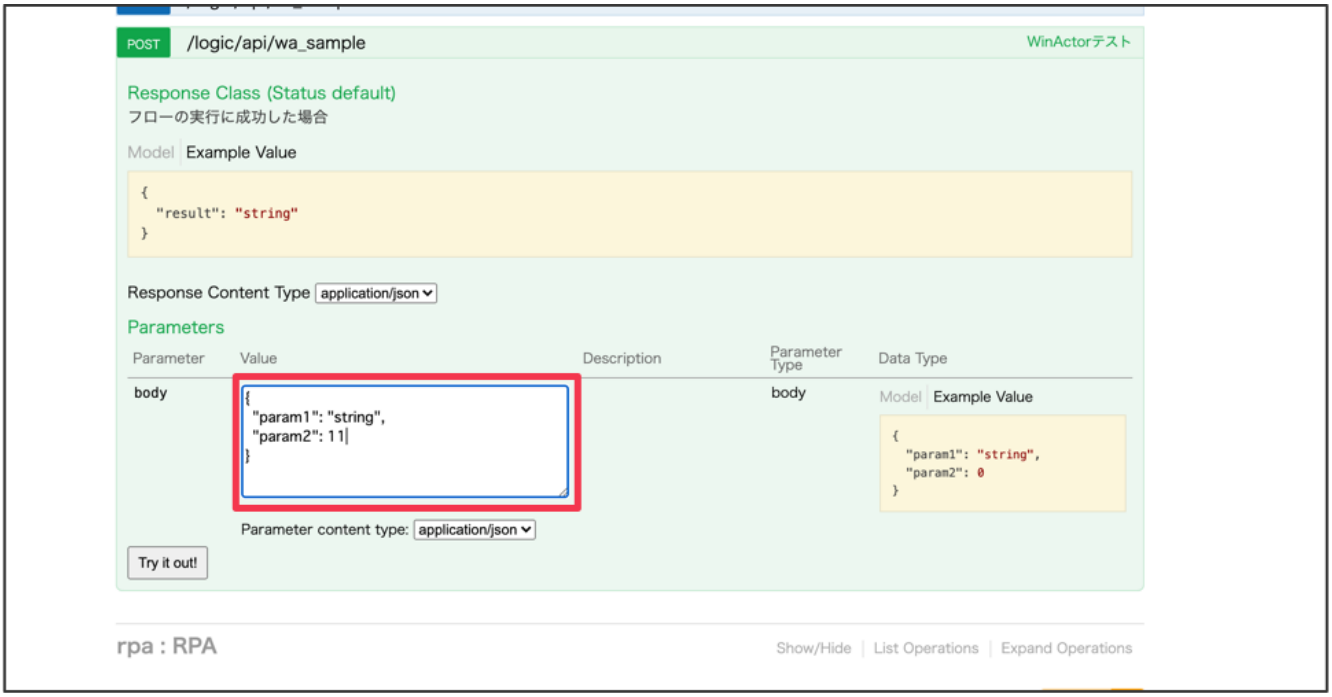
図：認可設定

> コラム
>
> 実運用をする際は適切な権限付与を検討してください。

swaggerで動作確認

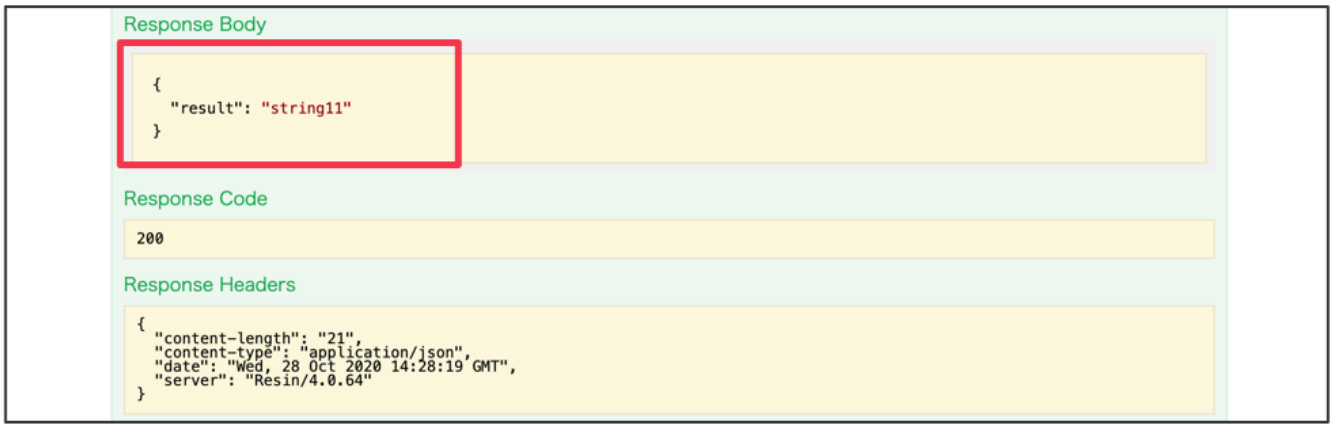
ロジックフロールーティング定義一覧より「SPEC」をクリックしてswaggerを表示します。
swaggerの設定については、「IM-LogicDesigner仕様書」 - 「Swaggerの利用」を参照してください。

リクエストbodyを適宜設定し、「Try it out!」をクリックします。



図：swaggerでリクエストを設定

リクエストの実行後、レスポンスコードが200で返ること、およびレスポンス内のパラメータresultが想定通りであることを確認してください。



図：swaggerでレスポンスを確認

フロールーティングの設定からswaggerでの動作確認の具体的な操作の流れについては、動画にてご覧いただけます。

BizRobo!連携

ここでは、BizRobo!連携のセットアップ手順、およびIM-LogicDesignerタスクの仕様を説明します。

セットアップ

項目

IM-Juggling プロジェクトの編集

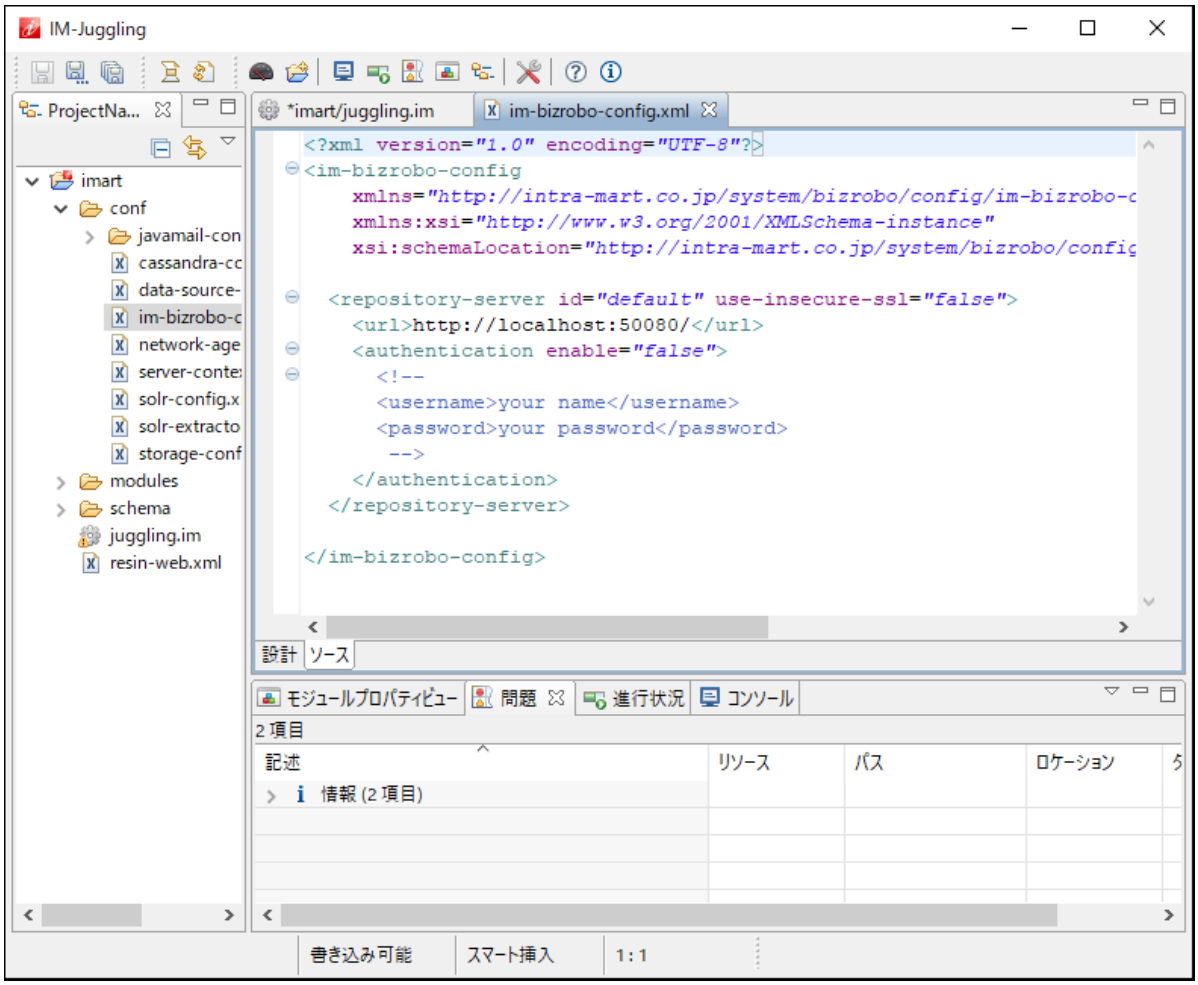
BizRobo!連携は、IM-RPAモジュールを使用します。
以下の手順で設定を行ってください。

1. IM-Jugglingにて、プロジェクトにIM-RPAモジュールを追加してください。
   以下のドキュメントを参照してください。
   - 「intra-mart Accel Platform セットアップガイド」-「プロジェクトの作成とモジュールの選択」

   > **注意**
   > IM-RPAモジュールのご利用には、エンタープライズ版の構成、およびライセンスが必要です。

2. 「設定ファイルが存在しません」という赤字のメッセージをクリックします。
   表示されるダイアログにて「OK」をクリックし、このプロジェクトに「BizRobo!クライアント設定ファイル」（im-bizrobo-config.xml）を追加します。

3. 「ProjectNavigator」内の「＜（プロジェクト名）/conf/im-bizrobo-config.xml> ファイル」をダブルクリックで開き、「ソース」を選択します。



4. *設定ファイルの説明* に従って、設定を記述してください。

5. 「BizRobo!クライアント設定ファイル」を保存します。
6. IM-JugglingでWARファイルを出力し、アプリケーションサーバへデプロイを行ってください。

ライセンスコードの登録

IM-RPAを利用するためには、テナント環境セットアップ後、製品ライセンスコードの登録が必要です。
テナント環境セットアップについては、「intra-mart Accel Platform セットアップガイド」-「テナント環境セットアップ」を参照してください。

1. IM-RPAのライセンス登録を行います。
   詳細は「ライセンスの登録」を参照してください。

設定ファイル

概要

BizRobo!連携 に関する設定です。

モジュール	BizRobo!連携
フォーマットファイル(xsd)	WEB-INF/schema/im-bizrobo-config.xsd
設定場所	WEB-INF/conf/im-bizrobo-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<im-bizrobo-config
   xmlns="http://intra-mart.co.jp/system/bizrobo/config/im-bizrobo-config"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://intra-mart.co.jp/system/bizrobo/config/im-bizrobo-config ../schema/im-bizrobo-config.xsd ">

  <repository-server id="default" use-insecure-ssl="false">
    <url>http://localhost:50080/</url>
    <authentication enable="false">
      <!--
      <username>your name</username>
      <password>your password</password>
      -->
    </authentication>
  </repository-server>

</im-bizrobo-config>
```

リファレンス

テナント別設定

タグ名	repository-server

テナント毎の BizRobo!接続に関する設定を定義します。

【設定項目】

```
<im-bizrobo-config>
 <repository-server id="default" use-insecure-ssl="false">
   :
 </server>
</im-bizrobo-config>
```

必須項目	×
複数設定	○
設定値・設定する内容	repository-server タグを親とするタグ
単位・型	なし
省略時のデフォルト値	なし
親タグ	im-bizrobo-config

【属性】

属性名	説明	必須	デフォルト値
id	テナントID このタグの設定の対象となるテナントIDを指定してください。	×	なし
use-insecure-ssl	セキュアでないSSLを利用するか否か BizRobo! Management Consoleへの接続にhttpsプロトコルを利用する場合は `true` を設定してください。	×	なし

URL設定

タグ名　url

BizRobo! Management ConsoleのURLを設定します。

【設定項目】

```
<im-bizrobo-config>
 <repository-server id="default" use-insecure-ssl="false">
   <url>http://localhost:50080/</url>
   :
 </server>
</im-bizrobo-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	BizRobo! Management ConsoleへのURL
単位・型	URL
省略時のデフォルト値	なし
親タグ	repository-server

認証設定

タグ名　authentication

BizRobo! Management Consoleへの接続時の認証設定です。

【設定項目】

```
<im-bizrobo-config>
 <repository-server id="default" use-insecure-ssl="false">
   :
  <authentication enable="true">
   :
  </authentication>
 </server>
</im-bizrobo-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	authentication タグを親とするタグ
単位・型	なし
省略時のデフォルト値	なし
親タグ	repository-server

【属性】

属性名	説明	必須	デフォルト値
enable	認証を必要とするか否か BizRobo! Management Consoleへの接続に認証が必要な場合 `true` を設定してください。	×	なし

ユーザ名設定

タグ名　username

BizRobo! Management Consoleへの接続時のユーザ名設定です。

【設定項目】

```
<im-bizrobo-config>
 <repository-server id="default" use-insecure-ssl="false">
   :
  <authentication enable="true">
   <username>your name</username>
   :
  </authentication>
 </server>
</im-bizrobo-config>
```

必須項目	×
複数設定	×
設定値・設定する内容	BizRobo! Management Consoleへ接続する際のユーザ名
単位・型	文字列
省略時のデフォルト値	なし
親タグ	authentication

パスワード設定

タグ名　password

BizRobo! Management Consoleへの接続時のパスワード設定です。

【設定項目】

```
<im-bizrobo-config>
 <repository-server id="default" use-insecure-ssl="false">
   :
  <authentication enable="true">
   :
    <password>your password</password>
  </authentication>
 </server>
</im-bizrobo-config>
```

必須項目	×
複数設定	×
設定値・設定する内容	BizRobo! Management Consoleへ接続する際のパスワード
単位・型	文字列
省略時のデフォルト値	なし
親タグ	authentication

IM-LogicDesigner タスク説明

項目

ロボットの配置

リポジトリにロボットを配置します。

入力値

```
im_bizRoboDeployRobot <object>
 ├─ fileData <binary> *
 ├─ fileName <string> *
 └─ projectName <string> *
```

項目名	必須/任意	型	配列/リスト	説明
im_bizRoboDeployRobot	任意	any	なし	-
fileData	必須	binary	なし	ロボットのデータストリーム
fileName	必須	string	なし	ロボットのファイル名
projectName	必須	string	なし	配置先のプロジェクト名

出力値

```
im_bizRoboDeployRobot <any>
```

項目名	型	配列/リスト	説明
im_bizRoboDeployRobot	any	なし	出力値として利用可能な値はありません。

ロボットの削除

リポジトリからロボットを削除します。

入力値

```
im_bizRoboDeleteRobot <object>
  ├─ fileName <string> *
  ├─ projectName <string> *
  └─ silent <boolean>
```

項目名	必須/任意	型	配列/リスト	説明
im_bizRoboDeleteRobot	任意	any	なし	-
fileName	必須	string	なし	ロボットのファイル名
projectName	必須	string	なし	配置先のプロジェクト名
silent	任意	boolean	なし	エラーを無視するか `false` を指定するとロボットが存在しない場合にエラーが発生します。

出力値

```
im_bizRoboDeleteRobot <any>
```

項目名	型	配列/リスト	説明
im_bizRoboDeleteRobot	any	なし	出力値として利用可能な値はありません。

タイプの配置

リポジトリにタイプを配置します。

入力値

```
im_bizRoboDeployType <object>
  ├─ fileData <binary> *
  ├─ fileName <string> *
  └─ projectName <string> *
```

項目名	必須/任意	型	配列/リスト	説明
im_bizRoboDeployType	任意	any	なし	-
fileData	必須	binary	なし	タイプのデータストリーム
fileName	必須	string	なし	タイプのファイル名
projectName	必須	string	なし	配置先のプロジェクト名

出力値

```
im_bizRoboDeployType <any>
```

項目名	型	配列/リスト	説明

項目名	型	配列/リスト	説明
im_bizRoboDeployType	any	なし	出力値として利用可能な値はありません。

タイプの削除

リポジトリからタイプを削除します。

入力値

```
im_bizRoboDeleteType <object>
  ├─ fileName <string> *
  ├─ projectName <string> *
  └─ silent <boolean>
```

項目名	必須/任意	型	配列/リスト	説明
im_bizRoboDeleteType	任意	any	なし	-
fileName	必須	string	なし	タイプのファイル名
projectName	必須	string	なし	配置先のプロジェクト名
silent	任意	boolean	なし	エラーを無視するか `false` を指定するとタイプが存在しない場合にエラーが発生します。

出力値

```
im_bizRoboDeleteType <any>
```

項目名	型	配列/リスト	説明
im_bizRoboDeleteType	any	なし	出力値として利用可能な値はありません。

スニペットの配置

リポジトリにスニペットを配置します。

入力値

```
im_bizRoboDeploySnippet <object>
  ├─ fileData <binary> *
  ├─ fileName <string> *
  └─ projectName <string> *
```

項目名	必須/任意	型	配列/リスト	説明
im_bizRoboDeploySnippet	任意	any	なし	-
fileData	必須	binary	なし	スニペットのデータストリーム
fileName	必須	string	なし	スニペットのファイル名
projectName	必須	string	なし	配置先のプロジェクト名

出力値

```
im_bizRoboDeploySnippet <any>
```

項目名	型	配列/リスト	説明
im_bizRoboDeploySnippet	any	なし	出力値として利用可能な値はありません。

スニペットの削除

リポジトリからスニペットを削除します。

入力値

```
im_bizRoboDeleteSnippet <object>
  ├─ fileName <string> *
  ├─ projectName <string> *
  └─ silent <boolean>
```

項目名	必須/任意	型	配列/リスト	説明
im_bizRoboDeleteSnippet	任意	any	なし	-
fileName	必須	string	なし	スニペットのファイル名
projectName	必須	string	なし	配置先のプロジェクト名
silent	任意	boolean	なし	エラーを無視するか `false` を指定するとスニペットが存在しない場合にエラーが発生します。

出力値

```
im_bizRoboDeleteSnippet <any>
```

項目名	型	配列/リスト	説明
im_bizRoboDeleteSnippet	any	なし	出力値として利用可能な値はありません。

リソースの配置

リポジトリにリソースを配置します。

入力値

```
im_bizRoboDeployResource <object>
  ├─ fileData <binary> *
  ├─ fileName <string> *
  └─ projectName <string> *
```

項目名	必須/任意	型	配列/リスト	説明
im_bizRoboDeployResource	任意	any	なし	-
fileData	必須	binary	なし	リソースのデータストリーム
fileName	必須	string	なし	リソースのファイル名
projectName	必須	string	なし	配置先のプロジェクト名

出力値

```
im_bizRoboDeployResource <any>
```

項目名	型	配列/リスト	説明

項目名	型	配列/リスト	説明
im_bizRoboDeployResource	any	なし	出力値として利用可能な値はありません。

リソースの削除

リポジトリからリソースを削除します。

入力値

```
im_bizRoboDeleteResource <object>
  ├─ fileName <string> *
  ├─ projectName <string> *
  └─ silent <boolean>
```

項目名	必須/任意	型	配列/リスト	説明
im_bizRoboDeleteResource	任意	any	なし	-
fileName	必須	string	なし	リソースのファイル名
projectName	必須	string	なし	配置先のプロジェクト名
silent	任意	boolean	なし	エラーを無視するか `false` を指定するとリソースが存在しない場合にエラーが発生します。

出力値

```
im_bizRoboDeleteResource <any>
```

項目名	型	配列/リスト	説明
im_bizRoboDeleteResource	any	なし	出力値として利用可能な値はありません。

ライブラリの展開

リポジトリにライブラリを展開します。

入力値

```
im_bizRoboDeployLibrary <object>
  ├─ library <binary> *
  └─ projectName <string> *
```

項目名	必須/任意	型	配列/リスト	説明
im_bizRoboDeployLibrary	任意	any	なし	-
library	必須	binary	なし	ライブラリのデータストリーム
projectName	必須	string	なし	配置先のプロジェクト名

出力値

```
im_bizRoboDeployLibrary <any>
```

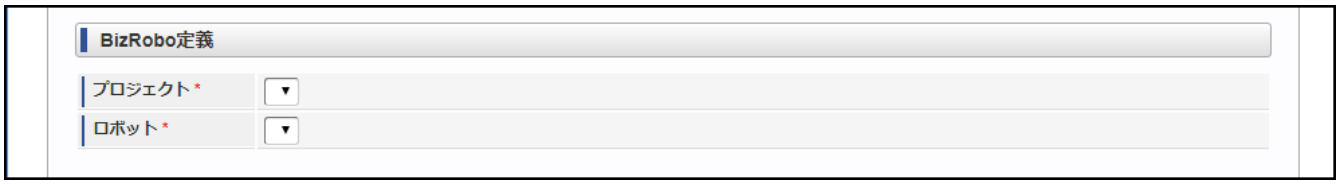
項目名	型	配列/リスト	説明
im_bizRoboDeployLibrary	any	なし	出力値として利用可能な値はありません。

IM-LogicDesigner ユーザ定義説明

BizRobo!定義の新規作成

IM-LogicDesignerのユーザ定義として、BizRobo!ロボット実行タスクが作成できます。
以下の手順で、IM-LogicDesignerにユーザ定義を追加してください。

1. ユーザ定義一覧より、BizRobo!定義を新規作成します。
   詳細な手順は、「IM-LogicDesigner ユーザ操作ガイド」 - 「ユーザ定義を新規登録する」を参照してください。
2. 共通設定を定義します。
   「入力値」および「返却値」については、次項のプロジェクトおよびロボットを選択することで自動設定します。
3. BizRobo! 定義の詳細情報を定義します。



<画面項目>

項目	説明
プロジェクト	対象のプロジェクトを選択します。
ロボット	実行対象のロボットを選択します。

注意

ユーザ定義の新規作成時に次のようなエラーが表示された場合、BizRobo! Management Consoleへの接続に失敗しています。
接続ファイルの記述内容やネットワーク環境に問題がないかご確認ください。

BizRobo!連携チュートリアル

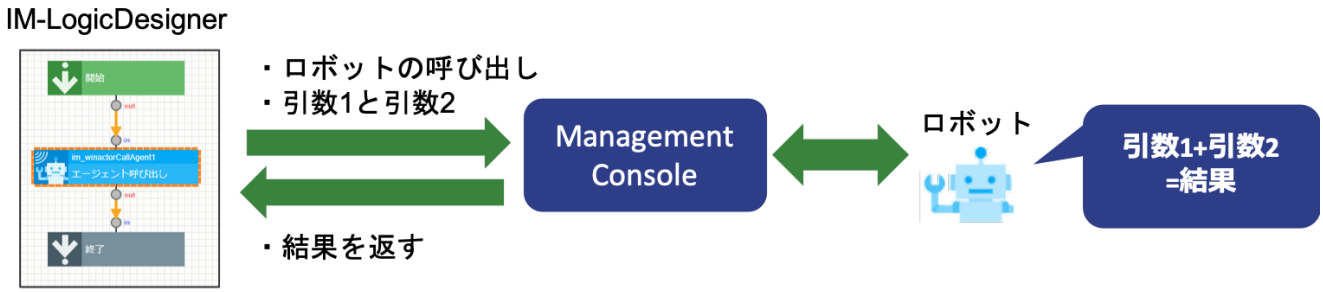
項目

チュートリアルの概要

本章では、IM-RPAのBizRobo!連携機能を使用して、BizRobo!のロボットを実行する方法をチュートリアル形式でご説明します。
このチュートリアルに沿って設定を行うことで、以下のような機能を実現できます。

- intra-mart Accel PlatformからIM-LogicDesignerを介してBizRobo!のロボットを実行します。
- BizRobo!のロボットに対してパラメータを渡し、計算した結果を受け取ります。

本章では、説明を簡単にするために、ロボットのシナリオはシンプルなものにしております。
シナリオの実行イメージは以下の通りです。



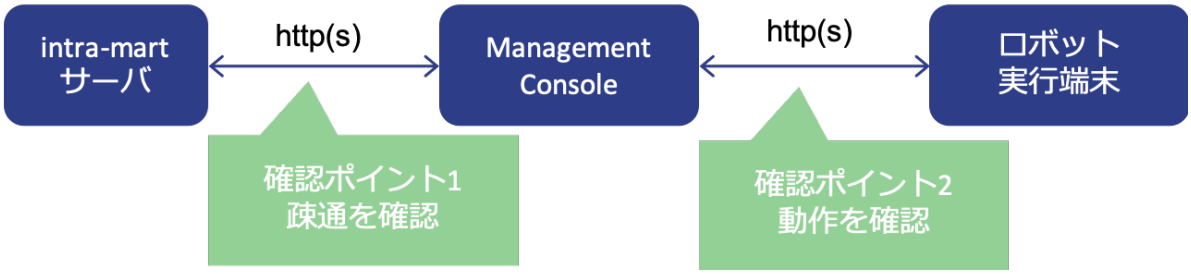
図：本チュートリアルシナリオ実行イメージ

準備・環境設定

BizRobo!環境の確認

BizRobo!のManagement ConsoleとDesign Studioが利用できる環境を準備します。
あらかじめ、 Management Consoleからロボットの実行ができることをご確認ください。
Management Consoleはintra-mart Accel Platformサーバとhttpポートで通信できる必要があります。

構成イメージ



図：構成イメージ

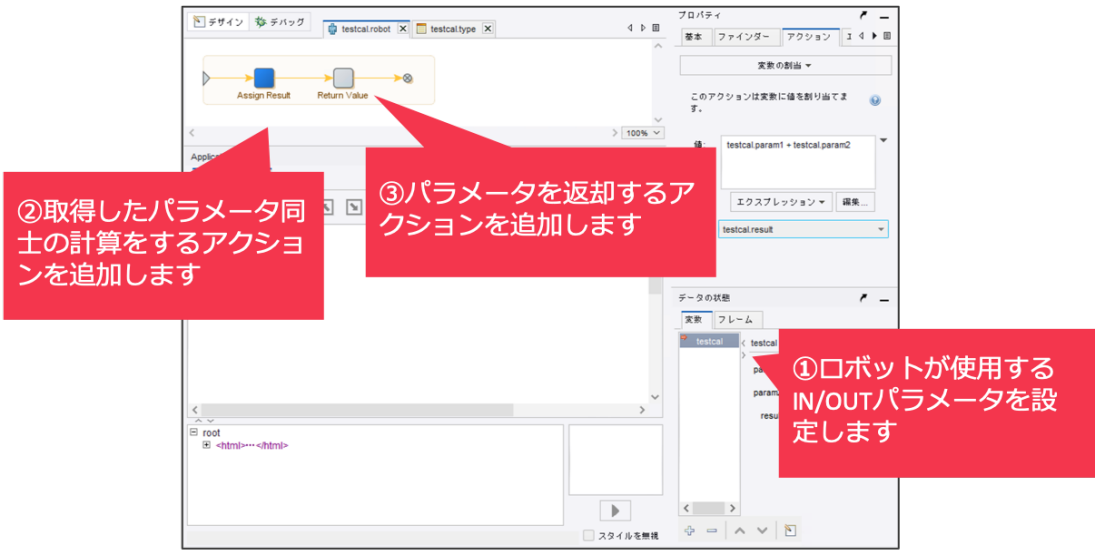
BizRobo!連携のセットアップ

IM-RPA利用ガイドの「BizRobo!連携」に従って、 intra-mart Accel Platformのセットアップおよび設定ファイルを記載します。

詳細は、「セットアップ」を参照してください。

ロボットの作成

Design Studioにてロボットを作成します。
以下は、ロボットの例です。



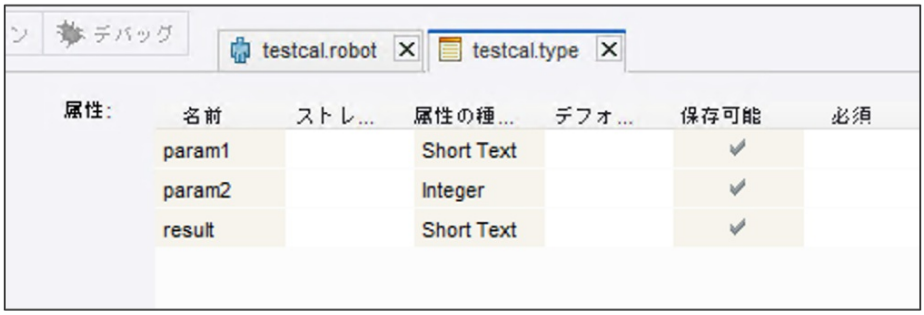
図：Design Studio作成イメージ

> **コラム**
>
> Design Studioを使用したロボットの作成方法については、 「BizRobo!ナレッジベース」を参照してください。

パラメータの設定

上記「①ロボットが使用するIN/OUTパラメータを設定します」部分でパラメータ設定を行います。

先に、ロボットが使用する「タイプ」を作成します。ここでは、作成するタイプを testcal とします。



図：Design Studioタイプの設定イメージ

> **コラム**
>
> 「タイプ」は、BizRobo!のロボットが使用する、複数の変数をまとめたものです。

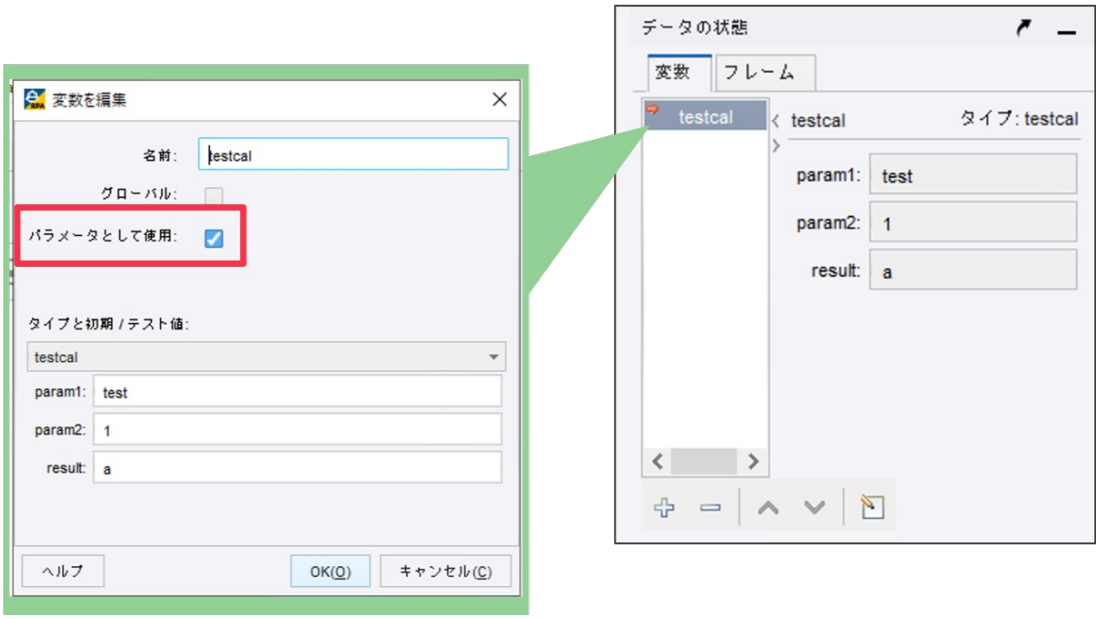
コラム

BizRobo!連携における、双方のパラメータの型は以下のようにマッピングされます。
記載以外のパラメータの型については、対応しておりません。

BizRobo!		intra-mart Accel Platform　備考
boolean	<->	Boolean
character	<->	Character
date	<->	IMDateTime
Long Text	<->	String
Short Text	<->	String

なお、IM-LogicDesigner側でパラメータを受け取る際に別のデータ型を指定した場合、自動的に型変換をします。
具体的な対応表は、「IM-LogicDesigner仕様書」 - 「IM-LogicDesigner データ型変換 仕様書」を参照してください。

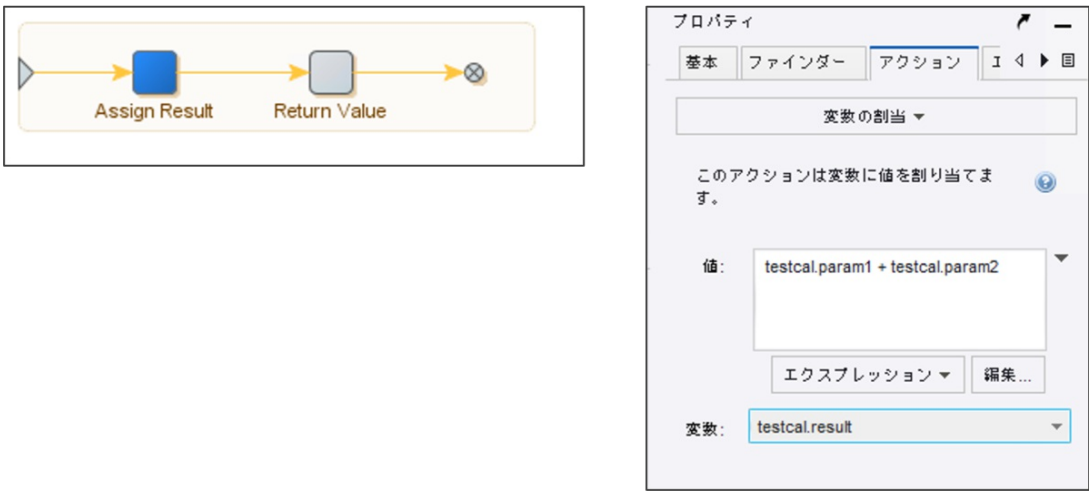
次に、ロボットの「データの状態」より、 testcal タイプを追加します。
変数を追加する際、「パラメータとして使用」にチェックをすることで、パラメータの入出力が可能です。



図：Design Studioパラメータの設定イメージ

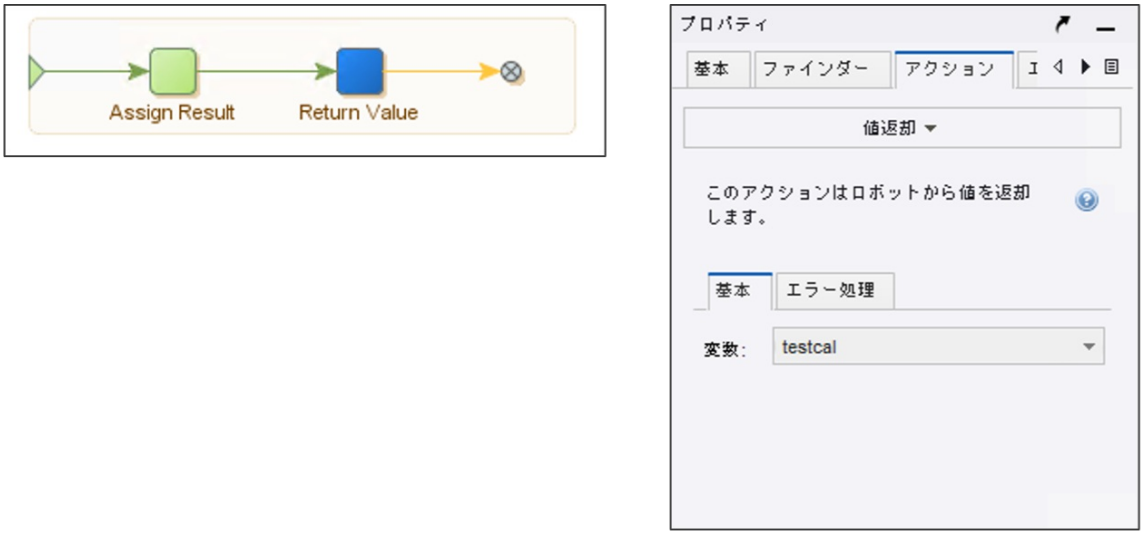
アクションの設定

上記「②取得したパラメータ同士の計算をするアクションを追加します」部分でロボットのアクションを設定します。
「変数の割当」アクションを以下のように設定します。



図：Design Studioアクションの設定 - 「変数の割当」

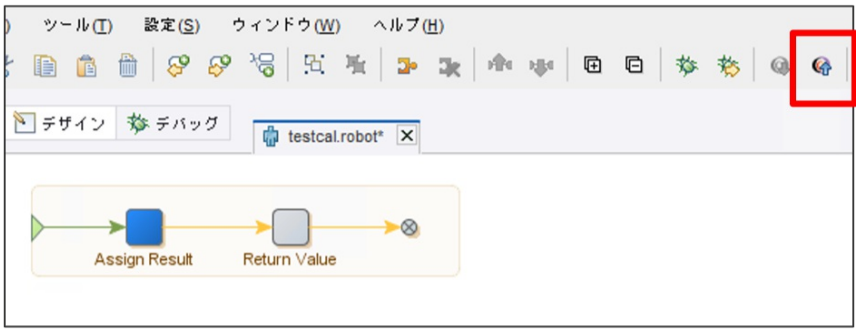
「③パラメータを返却するアクションを追加します」のアクションを設定します。
「値返却」アクションを以下のように設定します。



図：Design Studioアクションの設定 - 「値返却」

ロボットのデプロイ

Design Studioの「アップロード」ボタンを押下し、Management Consoleにロボットをアップロードしてください。



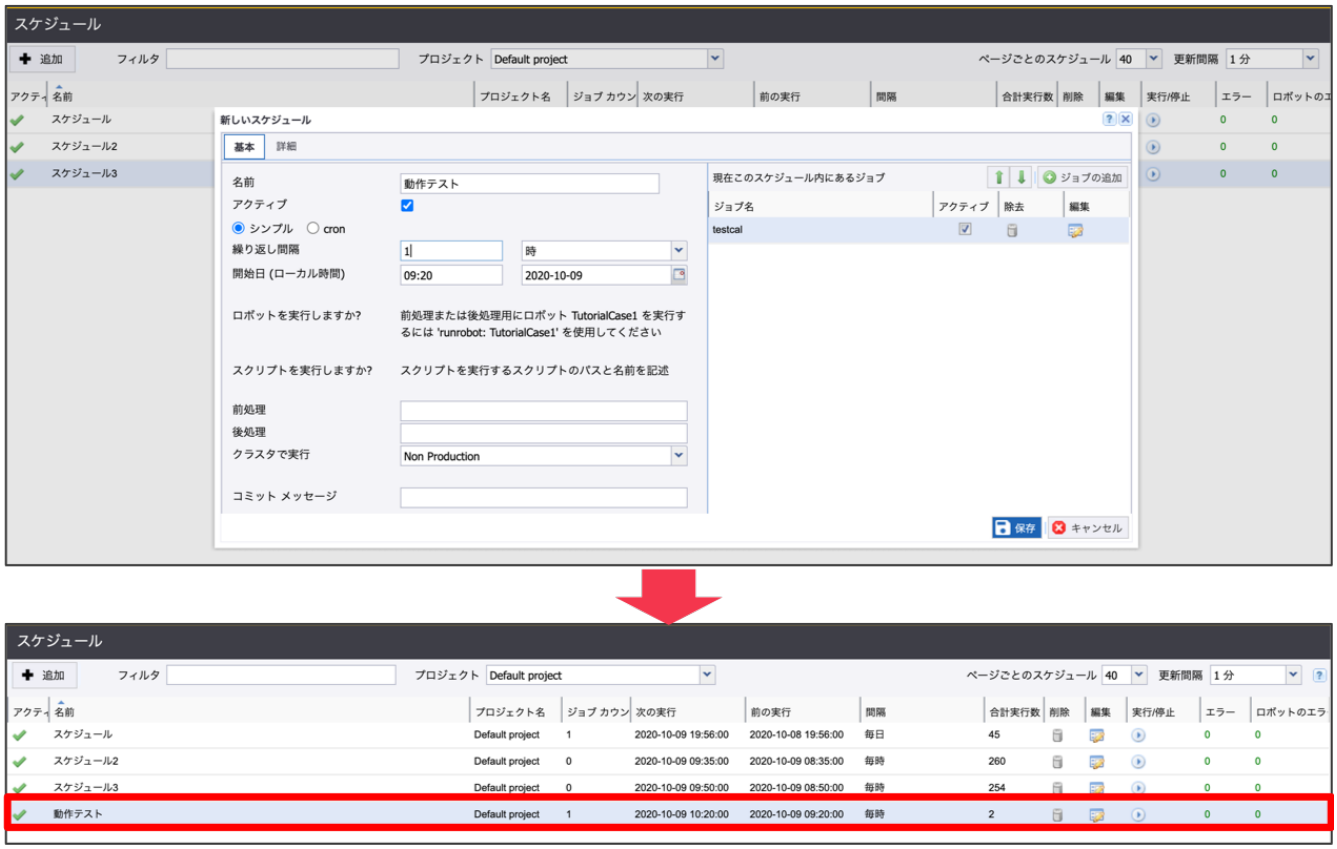
図：Design Studioロボットのデプロイ

> **コラム**
>
> ロボットやタイプは、Management Console上からアップロードすることも可能です。

Management Consoleにて、連携対象のプロジェクトに、ロボット、および変数タイプが登録されていることを確認してください。

図：Management Console「リポジトリ」 - 「ロボット」

デプロイしたロボットが、Management Console上で正常に動作することを確認してください。
引数を含むロボットは、スケジュール実行で動作を確認します。



図：Management Console「スケジュール」

> **コラム**
>
> 引数を含まないロボットは、リポジトリのロボットを直接実行することも可能です。
> Management Consoleの詳しい操作方法については、「BizRobo!ナレッジベース」を参照してください。

BizRobo!ロボットの作成からデプロイまでの具体的な操作の流れについては、動画にてご覧いただけます。

IM-LogicDesignerユーザ定義タスクの準備

「ユーザ定義タスク」とは、IM-LogicDesignerにおいて作成可能な、各種機能を持つ独自タスクです。
ユーザ定義タスクの具体的な説明については、「IM-LogicDesigner仕様書」 - 「ユーザ定義タスク」を参照してください。
本チュートリアルでは、 1つのユーザ定義タスクを用意いたします。

ユーザタスク種別	内容
BizRobo定義	Management Consoleを介してBizRobo!のロボットを実行します。 実行時に任意のパラメータの送受信を行います。

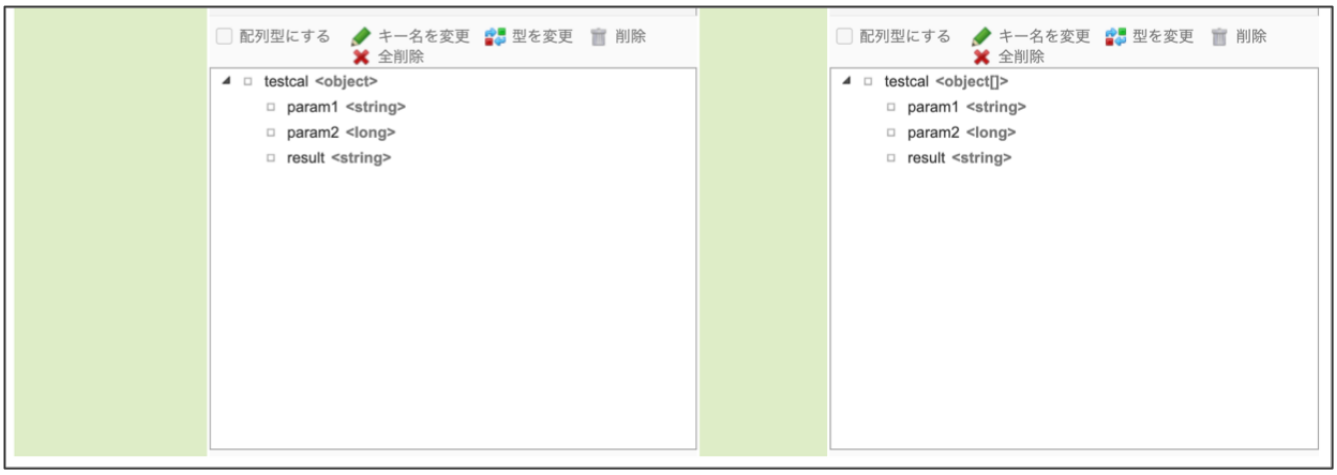
ユーザ定義作成（BizRobo定義）

「サイトマップ」→「LogicDesigner」→「ユーザ定義」→「BizRobo!定義新規作成」をクリックします。
BizRobo定義を下記のように編集します。
BizRobo定義カテゴリより、対象のプロジェクト、およびロボットを選択してください。

図：「BizRobo定義新規作成」 - 「BizRobo定義」

ユーザ定義タスク（BizRobo定義）の具体的な説明については、「*IM-LogicDesigner ユーザ定義説明*」を参照してください。

ロボットを選択すると、Management Consoleを介してロボットに必要な入出力パラメータを自動的に取得し、「入力値」および「返却値」に表示されます。
ここで入出力のパラメータを修正する必要はありません。

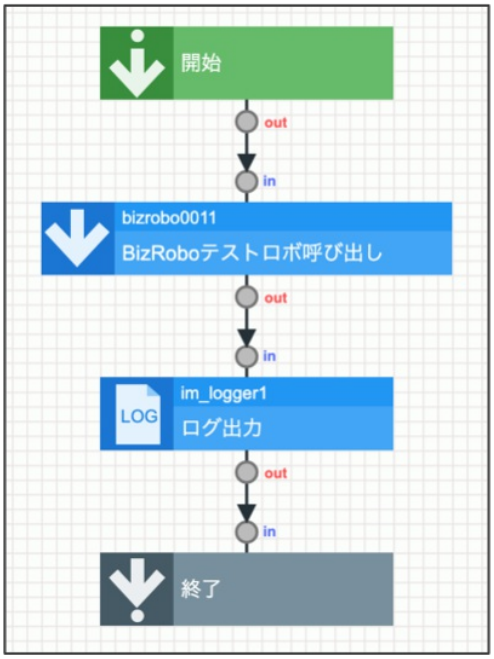


図：「BizRobo定義新規作成」 - 「入力値/出力値」

「ユーザ定義名」など適宜設定し、BizRobo定義を登録します。

IM-LogicDesignerフローの呼び出し

「サイトマップ」→「LogicDesigner」→「フロー定義一覧」→「新規作成」をクリックします。
以下は、BizRobo!連携（パラメータ使用）を行うための最もシンプルなIM-LogicDesignerフローです。



図：完成イメージ（ロジックフロー）

具体的な手順は以下の通りです。

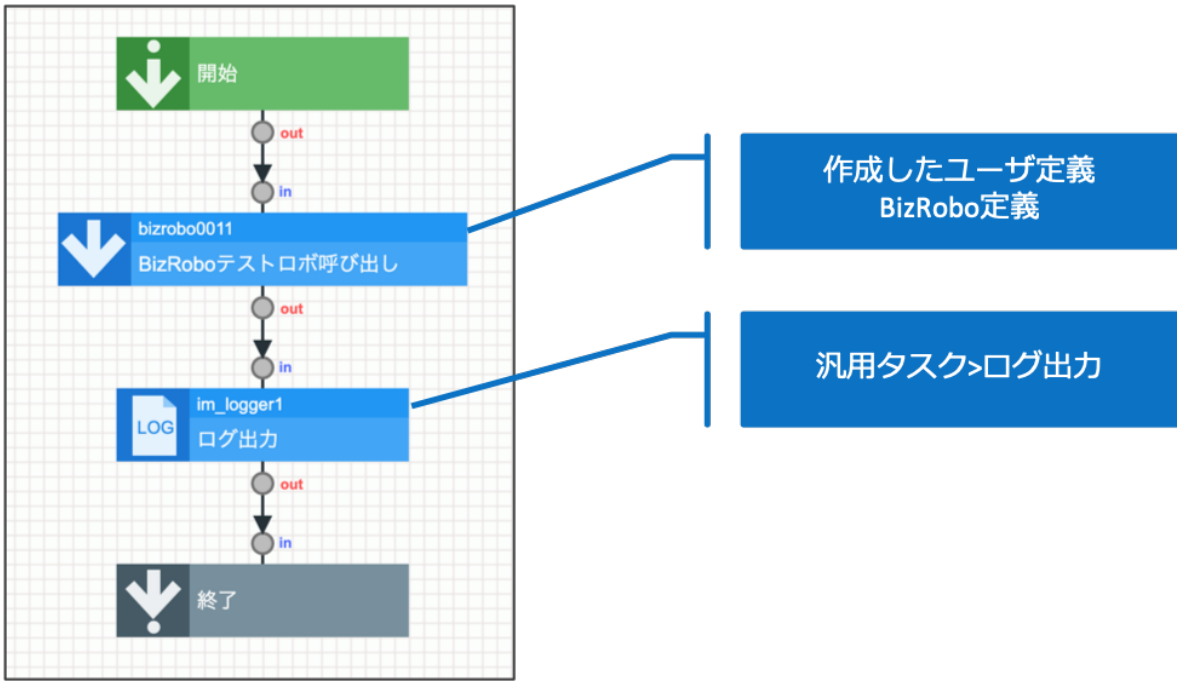
項目

タスクとフローを設定します

パレットから各タスクを選択し、今回使用するタスクをキャンパスに配置します。
前項で作成したユーザ定義は、パレット内の「前項で設定したユーザカテゴリ>作成したユーザ定義名」にあります。

図：IM-LogicDesignerタスク選択

配置する各タスクは、以下の通りです。



図：IM-LogicDesigner配置タスク

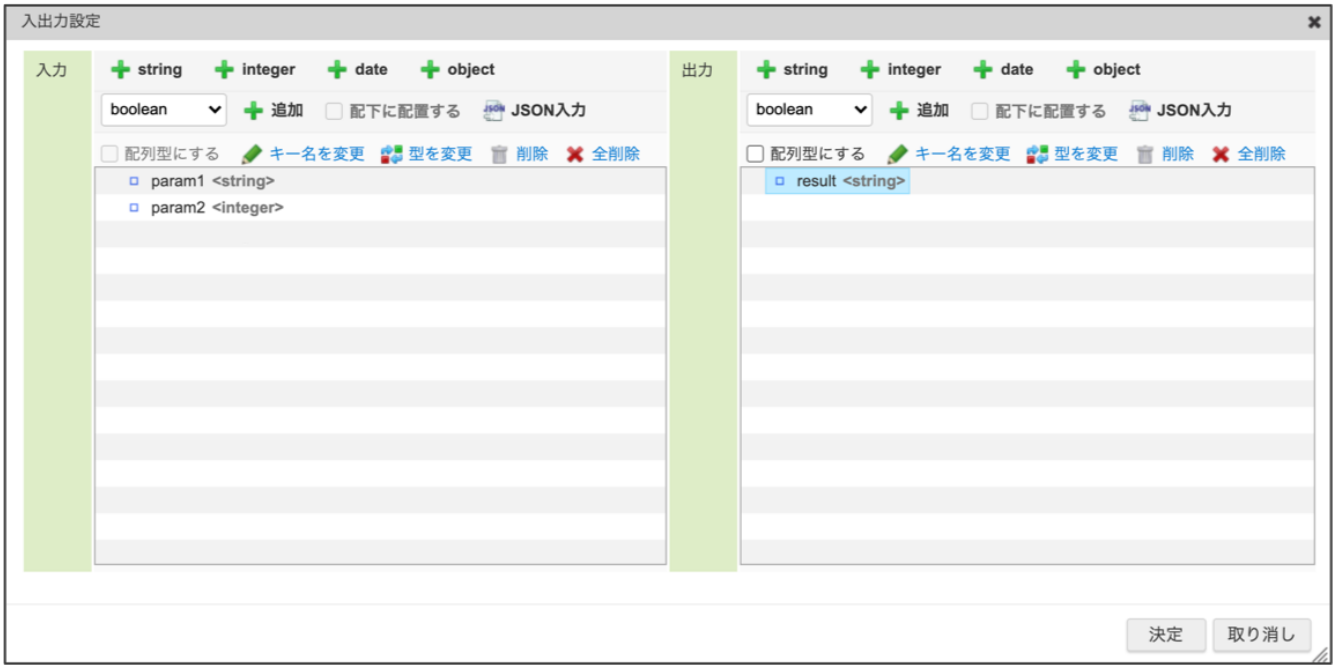
次に、配置したタスク同士をフローで繋ぎます。

ロボットに連携するパラメータを設定します

IM-LogicDesignerの「入出力設定」を行います。

設定値の説明（入出力設定）

カテゴリ	設定値	説明
入力	param1	ロボットに渡す引数1
入力	param2	ロボットに渡す引数2
出力	result	ロボットからの返り値



図：IM-LogicDesigner- 「入出力設定」

データマッピングをします

各タスクをダブルクリックして、「データマッピング」の設定を行います。
各図を参考にマッピングをしてください。

ロボット実行タスク（作成したユーザ定義 BizRobo定義）のデータマッピングを行います。

図：データマッピングの設定 - ロボット実行タスク

ログ出力（「汎用タスク」- 「ログ出力」）のデータマッピングを行います。

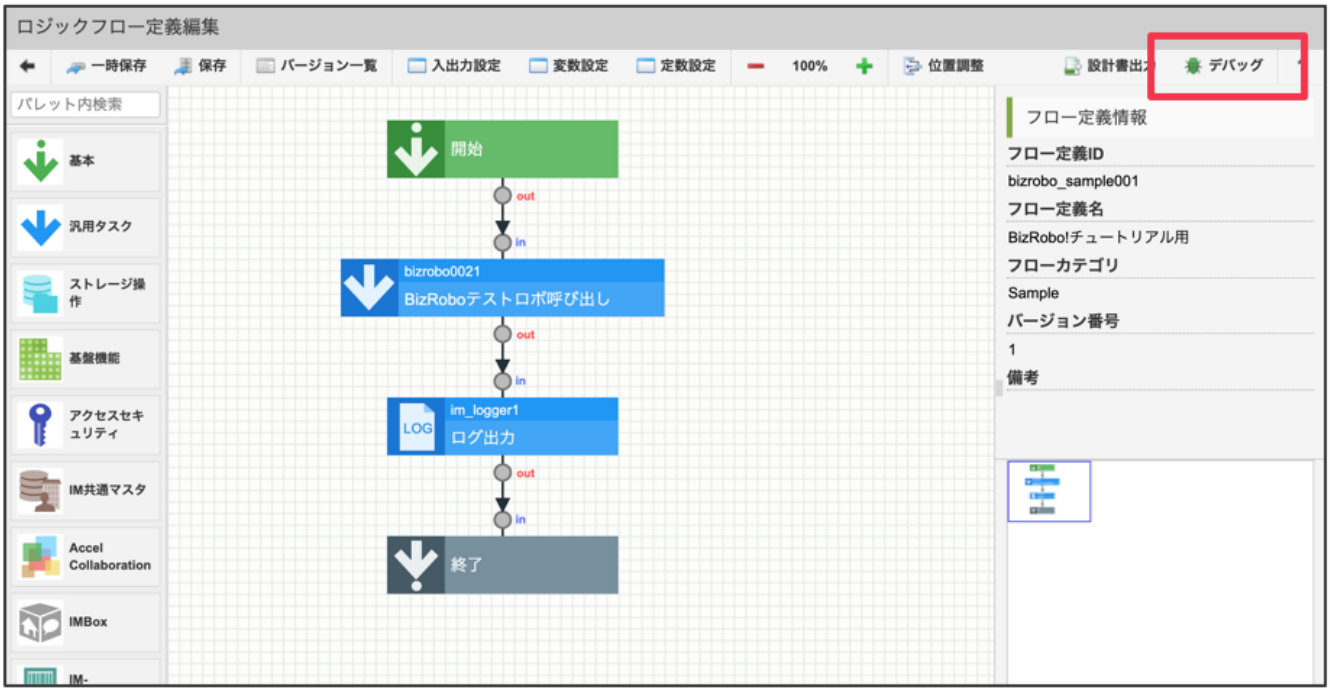
図：データマッピングの設定 - ログ出力

ロボットの実行結果resultを、フローの出力用変数にマッピングします。

図：データマッピングの設定 - 終了タスク

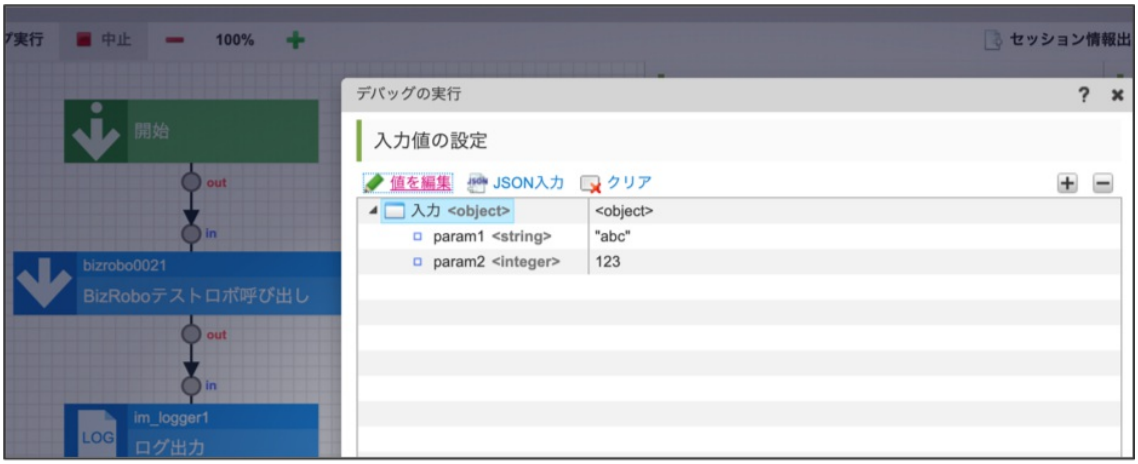
デバッグ実行で動作を確認します

ロジックフロー定義編集画面の「デバッグ」をクリックし、作成しているロジックフローをデバッグ実行します。



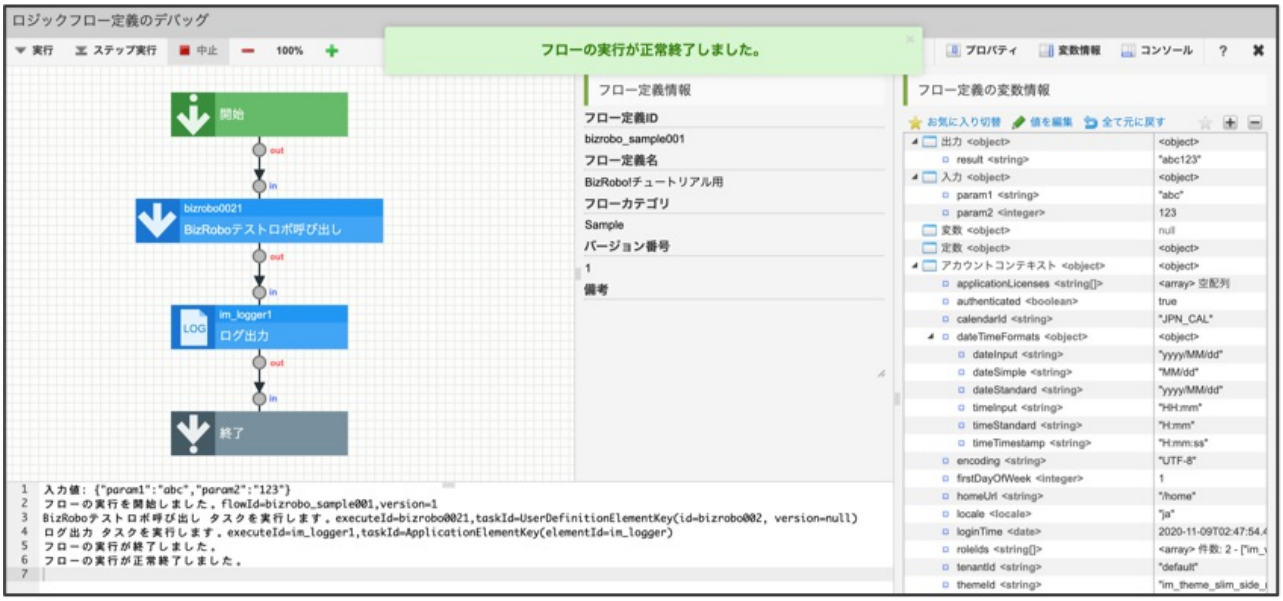
図：IM-LogicDesigner- 「デバッグ実行」

デバッグ画面にて、「実行」をクリックします。
デバッグ実行時の入力値の設定を行います。ロボットに引き渡すパラメータとなる、param1とparam2を設定してください。



図：IM-LogicDesigner- 「実行」- 「入出力値の設定」

デバッグ実行が正常に完了することを確認します。



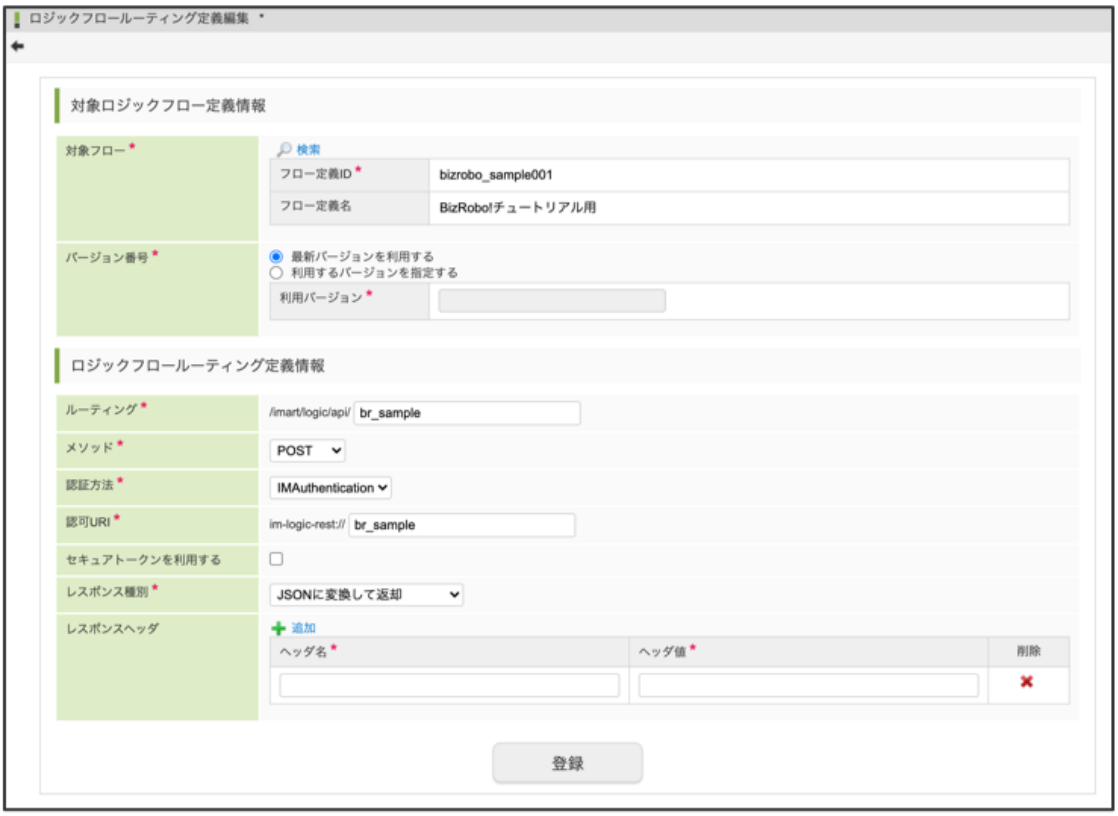
図：IM-LogicDesigner- デバッグ実行実行の成功

ロボットからの返却値となる、resultが正しく表示されていることを確認します。

図：IM-LogicDesigner- 「フローの変数情報（デバッグ実行後）」

フロールーティングの設定

作成したIM-LogicDesignerのフローを、intra-mart Accel Platform他機能やREST経由で実行するため、フロールーティングの設定を行います。
「サイトマップ」→「LogicDesigner」→「ルーティング定義一覧」→「新規作成」をクリックします。
フロールーティングの設定については、「IM-LogicDesigner仕様書」 - 「フロールーティングの認可設定」を参照してください。



図：フロールーティングの設定

> **コラム**
>
> メソッドには「POST」を選択してください。

ロジックフロールーティング定義一覧より「認可」をクリックして、認可設定を行います。
「認証済みユーザ」へ実行権限を付与します。

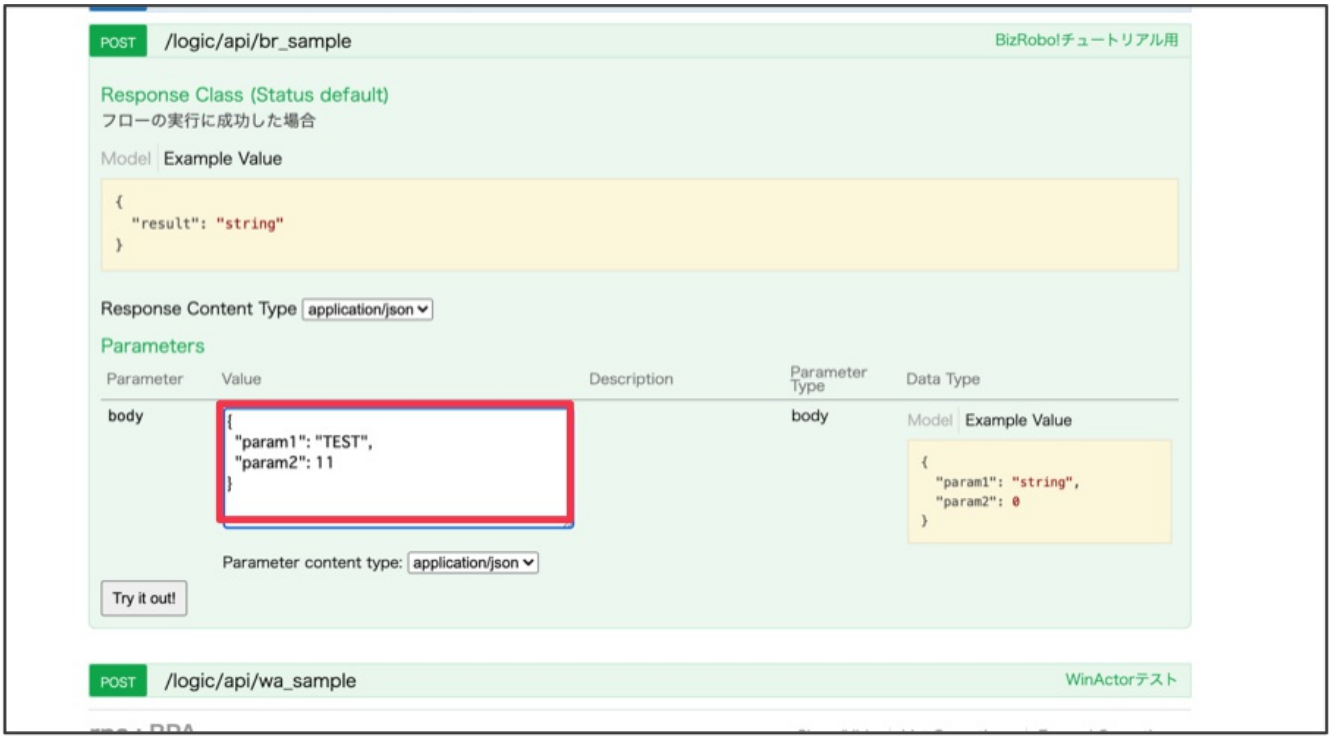
図：認可設定

> **コラム**
>
> 実運用をする際は適切な権限付与を検討してください。

swaggerで動作確認

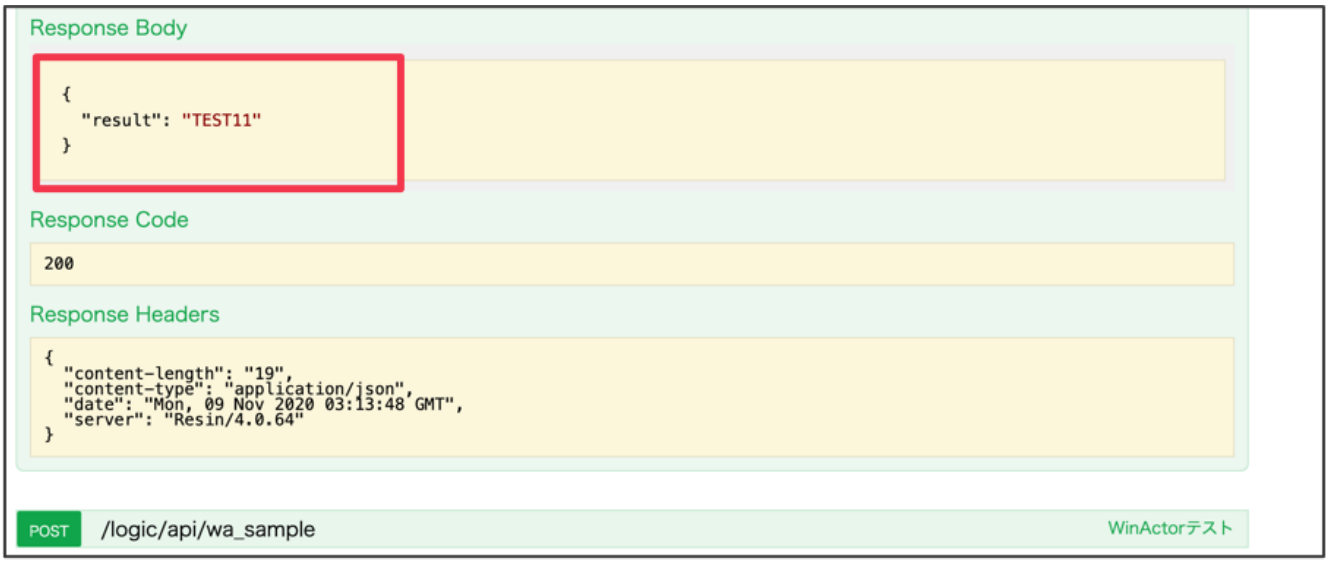
ロジックフロールーティング定義一覧より「SPEC」をクリックしてswaggerを表示します。
swaggerの設定については、「IM-LogicDesigner仕様書」 - 「Swaggerの利用」を参照してください。

リクエストbodyを適宜設定し、「Try it out!」をクリックします。



図：swaggerでリクエストを設定

リクエストの実行後、レスポンスコードが200で返ること、およびレスポンス内のパラメータresultが想定通りであることを確認してください。



図：swaggerでレスポンスを確認

フロールーティングの設定からswaggerでの動作確認の具体的な操作の流れについては、動画にてご覧いただけます。

BizRobo!連携チュートリアル（その他BizRobo!タスクの活用）

項目

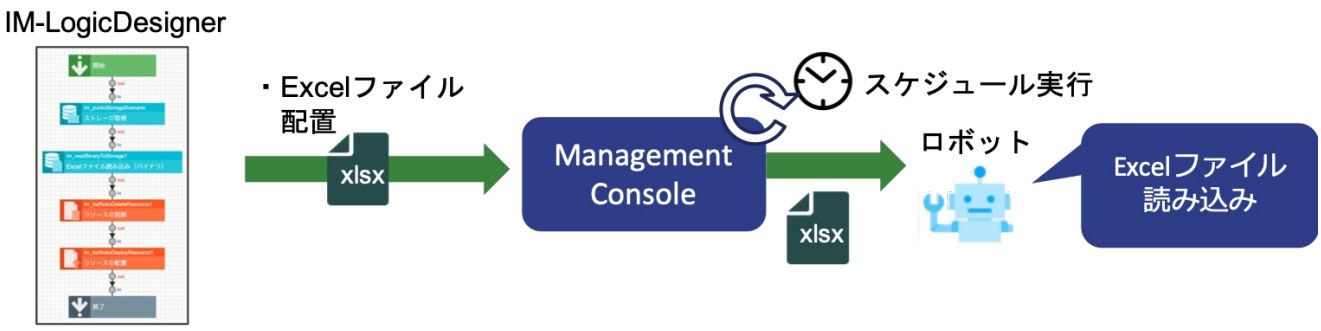
チュートリアルの概要

本章では、IM-RPAのBizRobo!連携機能を使用して、BizRobo!のロボットが使用するリソースファイルを連携する方法をチュートリアル形式でご説明します。
このチュートリアルに沿って設定を行うことで、以下のような機能を実現できます。

- intra-mart Accel PlatformからIM-LogicDesignerを介してBizRobo!のロボットが使用するExcelファイルを連携します。

本章では、説明を簡単にするために、ロボットのシナリオはシンプルなものにしております。
またここでは、「*BizRobo!連携チュートリアル*」と説明が重複する部分について、詳細な手順を省略して記載しております。

シナリオの実行イメージは以下の通りです。
Excelファイルを Management Consoleに保存し、別途Management Consoleのスケジュール実行により当該Excelファイルを読み込みます。



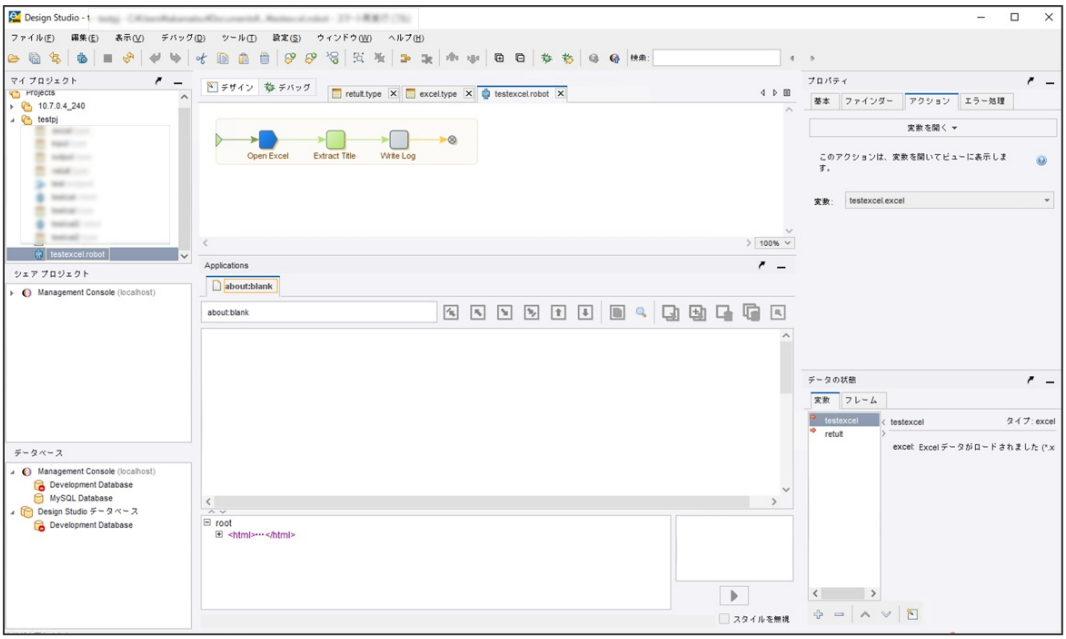
図：本チュートリアルシナリオ実行イメージ

> **コラム**
>
> ロボットに、Excelなどのファイルを連携するには、Management Consoleの「リソース」を使用します。
> ロボットがリソースを利用するには、Management Consoleのスケジュール機能でロボットを実行する必要があります。
> このため、 本章ではIM-RPAをExcelファイルの連携にのみ使用し、ロボットの実行はManagement Consoleのスケジュール機能で行います。

ロボットの作成

Design Studioにてロボットを作成します。
以下は、ロボットの例です。

最初のアクションは、Excelタイプの変数を開くアクションです。



図：Design Studio作成イメージ

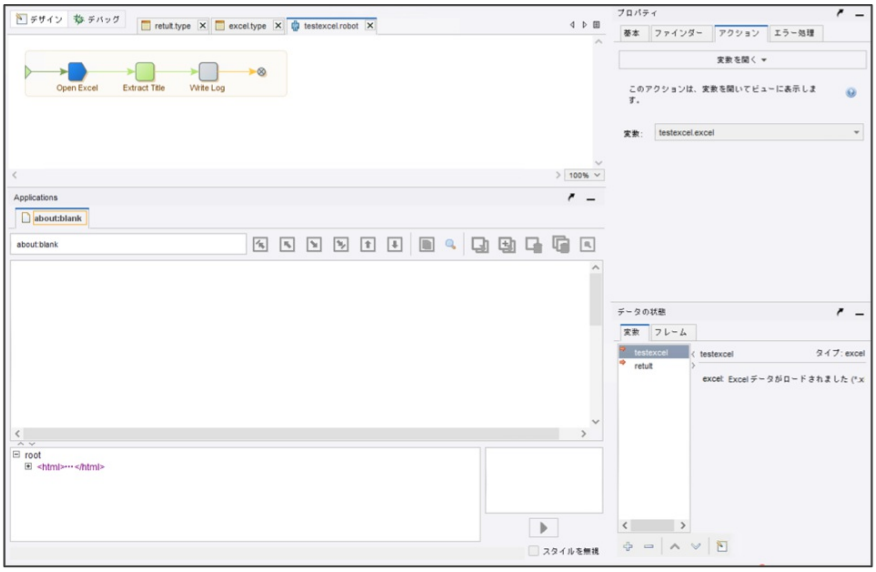
> **コラム**
>
> Design Studioを使用したロボットの作成方法については、 「BizRobo!ナレッジベース」を参照してください。

パラメータの設定

パラメータ設定を行います。ここでは、Excel読み込み用の変数タイプと、読み取り結果格納用の変数タイプを適宜作成します。

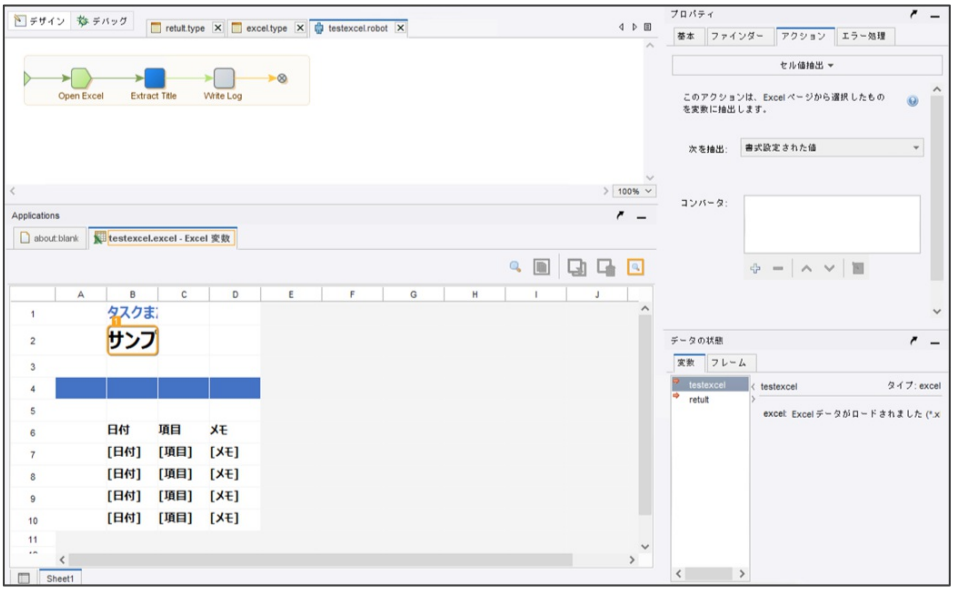
アクションの設定

Excelタイプの変数を開くアクションを設定します。



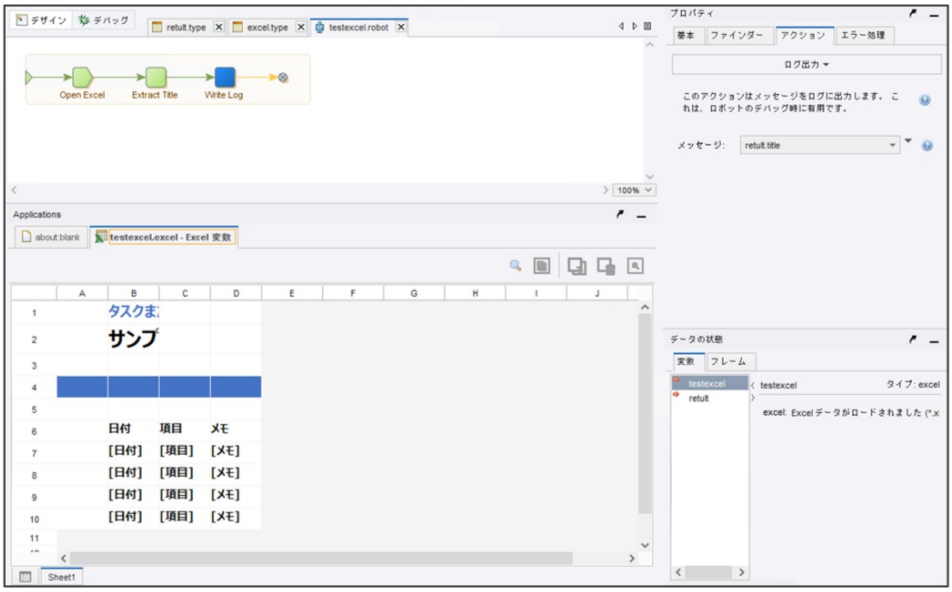
図：Design Studioアクションの設定 -「変数を開く」

サンプルのExcel帳票から、特定のセルを抽出して result.title に保存するアクションを設定します。



図：Design Studioアクションの設定 -「値の抽出」

result.title をログに保存するアクションを設定します。



図：Design Studioアクションの設定 -「ログ出力」

ロボットのデプロイ

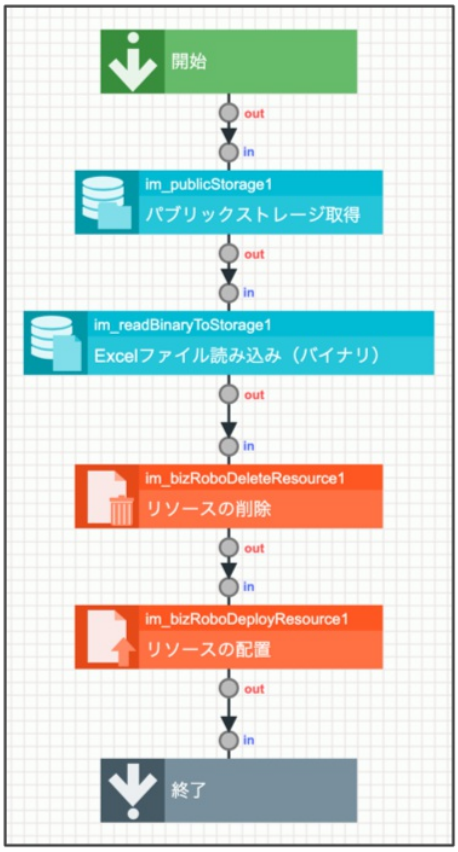
Management Consoleにログインし、ロボットとタイプをアップロードしてください。

図：Design Studioロボットとタイプのデプロイ

> **コラム**
>
> ロボットやタイプは、Design Studioから直接アップロードすることも可能です。

IM-LogicDesignerフローの呼び出し

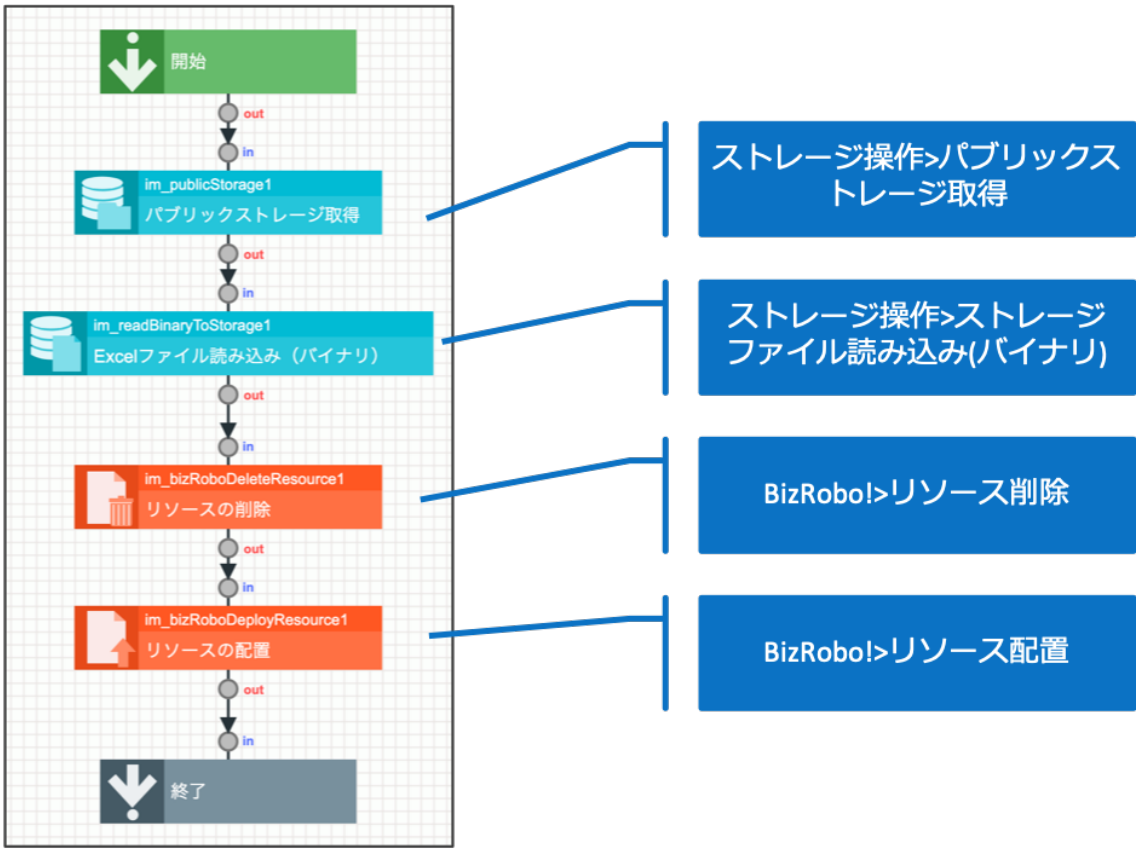
「サイトマップ」→「LogicDesigner」→「フロー定義一覧」→「新規作成」をクリックします。
以下は、BizRobo!連携（リソースファイル連携）を行うための最もシンプルなIM-LogicDesignerフローです。



図：完成イメージ（ロジックフロー）

タスクとフローを設定します

パレットから各タスクを選択し、今回使用するタスクをキャンパスに配置します。
配置する各タスクは、以下の通りです。



図：IM-LogicDesigner配置タスク

ロボットに連携するパラメータを設定します

IM-LogicDesignerの「定数設定」を行います。

設定値の説明（定数設定）

定数ID	定数値	説明
FILE_NAME	test.xlsx	ロボットに渡すExcelファイル名
PROJECT_NAME	Default project	プロジェクト名
SILENT	true	ファイル削除操作時に渡す引数（削除対象ファイルがなくてもエラーにしない）
STORAGE_PATH	RPA/test.xlsx	Excelファイルのストレージファイルパス

図：IM-LogicDesigner- 「入出力設定」

データマッピングをします

各タスクをダブルクリックして、「データマッピング」の設定を行います。
各図を参考にマッピングをしてください。

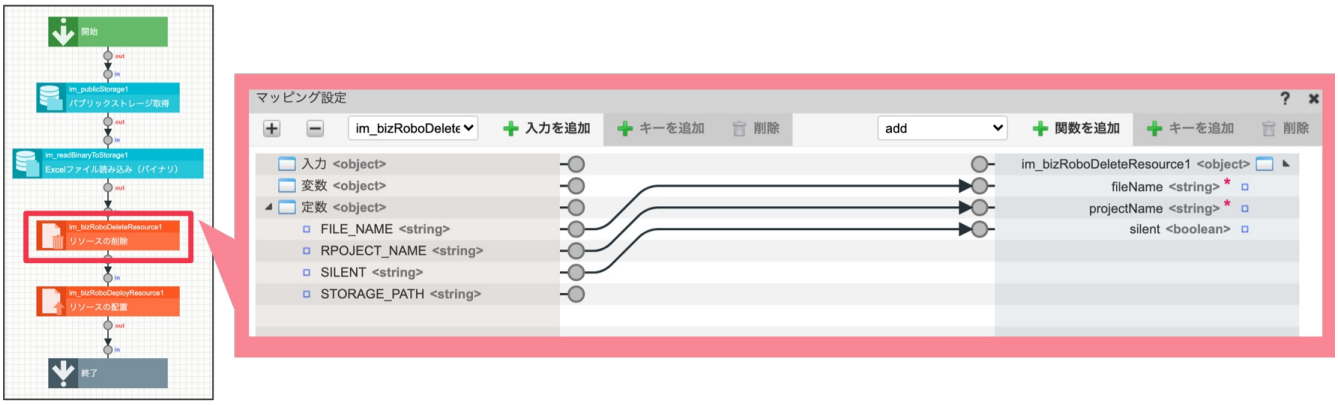
Excel取得タスク（「ストレージ操作 」- 「パブリックストレージ取得」）のデータマッピングを行います。

図：データマッピングの設定 - 「ストレージ操作 」- 「パブリックストレージ取得」タスク

Excel取得タスク（「ストレージ操作 」- 「ストレージファイル読み込み（バイナリ）取得」）のデータマッピングを行います。

図：データマッピングの設定 - 「ストレージ操作 」- 「ストレージファイル読み込み（バイナリ）取得」タスク

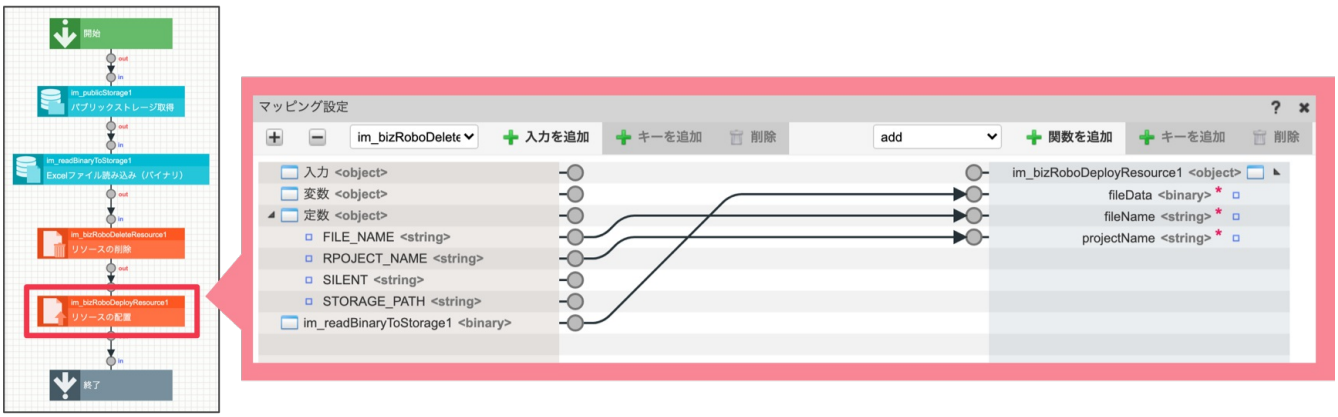
リソース削除タスク（「BizRobo!」- 「リソースの削除」）のデータマッピングを行います。



図：データマッピングの設定 - 「BizRobo!」- 「リソースの削除」タスク

> **コラム**
>
> 「リソースの配置」タスクの前に「リソースの削除」タスクを実行する理由は、リソース配置時にすでに同名のリソースが存在した場合にエラーとなるのを防ぐためです。

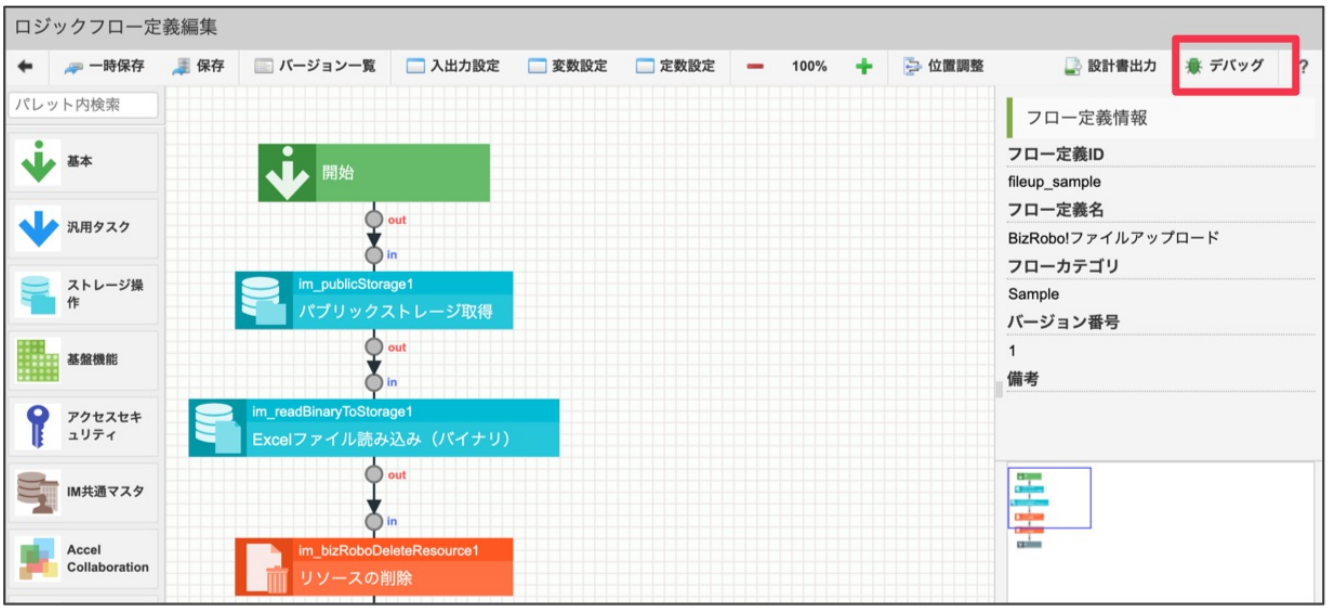
リソース配置タスク（「BizRobo!」- 「リソースの配置」）のデータマッピングを行います。



図：データマッピングの設定 - 「BizRobo!」- 「リソースの配置」タスク

デバッグ実行で動作を確認します

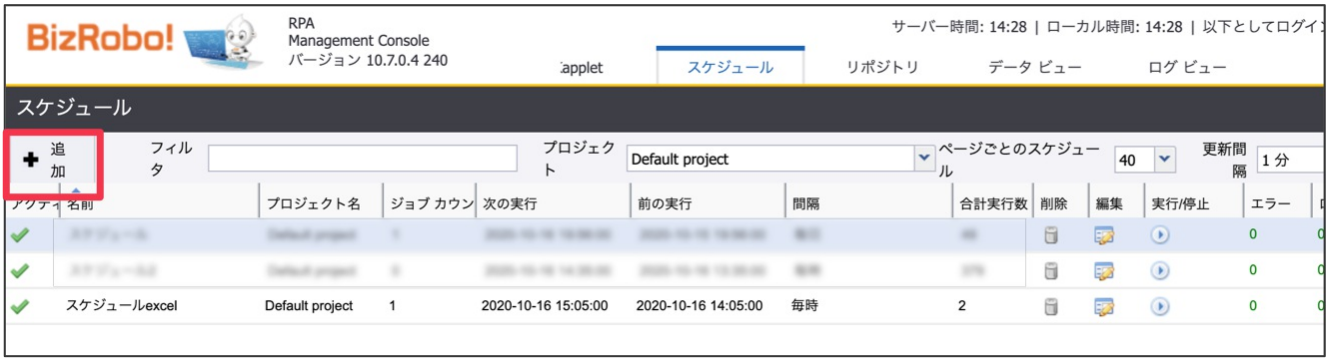
デバッグ実行を行い、ロジックフローの動作を確認してください。
デバッグ実行が成功すること、およびMC上のリソースにExcelファイルがアップロードされていることを確認してください。



図：IM-LogicDesigner- 「デバッグ実行」

Management Consoleのスケジュール設定・動作確認

1. Management Console上で、ロボットのスケジュール実行の設定、および動作確認を行います。
   Management Consoleにログイン後、スケジュールの追加をクリックします。



図：Management Consoleスケジュール追加

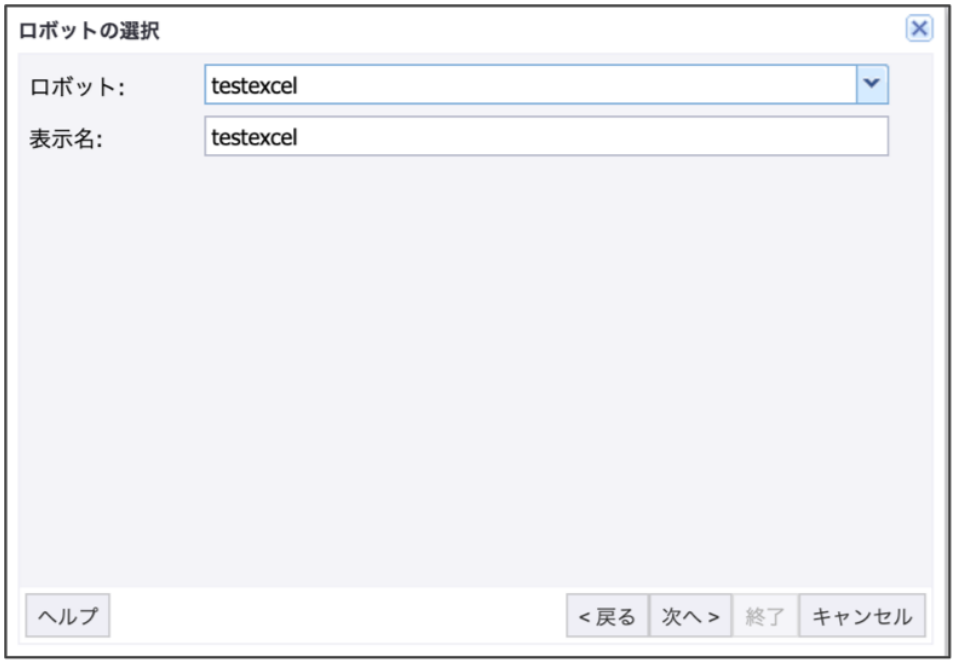
2. 適宜スケジュール設定をしてください。
   「ジョブの追加」をクリックします。

図：Management Consoleスケジュール設定

3. 「ジョブ　タイプの選択」で「シングルロボット」を選択します。
   「次へ」をクリックします。

図：Management Consoleジョブタイプの設定

4. 先ほどアップロードしたロボットを選択します。



図：Management Consoleロボットの選択

5. ロボットに設定したExcelタイプの変数を選択し、「excel:」横の選択メニューより、配置済みのExcelリソースを選択します。
「終了」をクリックします。

図：Management Console変数の入力値を設定

6. 設定したスケジュールが実行されるのを待ちます。
スケジュールが実行されたのち、「ログビュー」より、ロボットの実行メッセージを確認してください。
intra-mart Accel Platformから配置したリソースが正しく読み込まれていることを確認してください。

図：Management Consoleログの確認

UiPath連携

ここでは、UiPath連携のセットアップ手順、およびIM-LogicDesignerタスクの仕様を説明します。

セットアップ

項目

IM-Juggling プロジェクトの編集

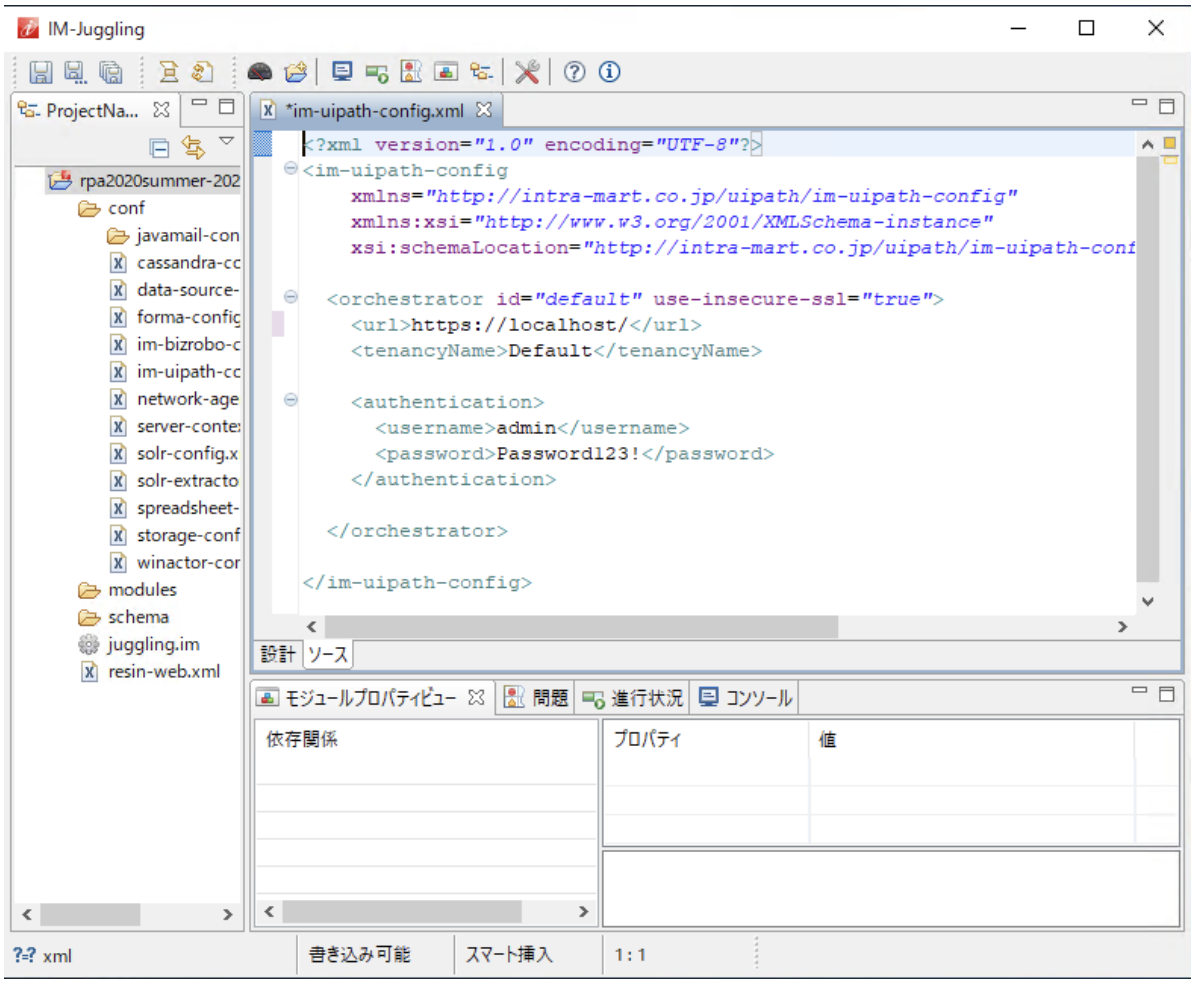
UiPath連携は、IM-RPAモジュールを使用します。
以下の手順で設定を行ってください。

1. IM-Jugglingにて、プロジェクトにIM-RPAモジュールを追加してください。
   以下のドキュメントを参照してください。
   - 「intra-mart Accel Platform セットアップガイド」-「プロジェクトの作成とモジュールの選択」

   > **注意**
   > IM-RPAモジュールのご利用には、エンタープライズ版の構成、およびライセンスが必要です。

2. 「設定ファイルが存在しません」という赤字のメッセージをクリックします。
   表示されるダイアログにて「OK」をクリックし、このプロジェクトに「UiPathクライアント設定ファイル」(im-uipath-config.xml)を追加します。

3. 「ProjectNavigator」内の「＜（プロジェクト名）/conf/im-uipath-config.xml> ファイル」をダブルクリックで開き、「ソース」を選択します。



4. *設定ファイルの説明* に従って、設定を記述してください。

5. 「UiPathクライアント設定ファイル」を保存します。
6. IM-JugglingでWARファイルを出力し、アプリケーションサーバへデプロイを行ってください。

ライセンスコードの登録

IM-RPAを利用するためには、テナント環境セットアップ後、製品ライセンスコードの登録が必要です。
テナント環境セットアップについては、「intra-mart Accel Platform セットアップガイド」-「テナント環境セットアップ」を参照してください。

1. IM-RPAのライセンス登録を行います。
   詳細は「ライセンスの登録」を参照してください。

設定ファイル

概要

UiPath連携 に関する設定です。

モジュール	UiPath連携
フォーマットファイル(xsd)	WEB-INF/schema/im-uipath-config.xsd
設定場所	WEB-INF/conf/im-uipath-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<im-uipath-config
   xmlns="http://intra-mart.co.jp/uipath/im-uipath-config"
   mlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://intra-mart.co.jp/uipath/im-uipath-config ../schema/im-uipath-config.xsd ">

 <orchestrator id="default" use-insecure-ssl="true">
   <url>https://xxx.xxx.xxx.xxx/</url>
   <tenancyName>Default</tenancyName>

   <authentication>
     <username>username</username>
     <password>password</password>
   </authentication>
 </orchestrator>
</im-uipath-config>
```

リファレンス

テナント別設定

タグ名	orchestrator

テナント毎の　UiPath Orchestrator接続に関する設定を定義します。

【設定項目】

```
<im-uipath-config>
 <orchestrator id="default" use-insecure-ssl="true">
   :
 </orchestrator>
</im-uipath-config>
```

必須項目	×
複数設定	○
設定値・設定する内容	orchestrator タグを親とするタグ
単位・型	なし
省略時のデフォルト値	なし
親タグ	im-uipath-config

【属性】

属性名	説明	必須	デフォルト値
id	テナントID このタグの設定の対象となるテナントIDを指定してください。	×	なし
use-insecure-ssl	セキュアでないSSLを利用するか否か UiPath Orchestratorへの接続にhttpsプロトコルを利用する場合は `true` を設定してください。	×	なし

URL設定

タグ名　url

UiPath OrchestratorのURLを設定します。

【設定項目】

```
<im-uipath-config>
 <orchestrator id="default" use-insecure-ssl="true">
   <url>http://xxx.xxx.xxx.10/</url>
   :
 </orchestrator>
</im-uipath-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	UiPath OrchestratorへのURL
単位・型	URL
省略時のデフォルト値	なし
親タグ	orchestrator

テナント名設定

タグ名　tenancyName

UiPath Orchestratorのテナント名を設定します。

【設定項目】

```
<im-uipath-config>
 <orchestrator id="default" use-insecure-ssl="true">
  <tenancyName>Default</tenancyName>
   :
 </orchestrator>
</im-uipath-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	UiPath Orchestrator上で管理するテナント名
単位・型	文字列
省略時のデフォルト値	なし
親タグ	orchestrator

認証設定

タグ名	authentication

UiPath Orchestratorへの接続時の認証設定です。

【設定項目】

```
<im-uipath-config>
 <orchestrator id="default" use-insecure-ssl="true">
   :
  <authentication>
   :
  </authentication>
 </orchestrator>
</im-uipath-config>
```

必須項目	○
複数設定	×
設定値・設定する内容	authentication タグを親とするタグ
単位・型	なし
省略時のデフォルト値	なし
親タグ	orchestrator

ユーザ名設定

タグ名	username

UiPath Orchestratorへの接続時のユーザ名設定です。

【設定項目】

```
<im-uipath-config>
 <orchestrator id="default" use-insecure-ssl="true">
   :
  <authentication>
   <username>username</username>
   :
  </authentication>
 </orchestrator>
</im-uipath-config>
```

必須項目	×
複数設定	×
設定値・設定する内容	UiPath Orchestratorへ接続する際のユーザ名
単位・型	文字列
省略時のデフォルト値	なし
親タグ	authentication

パスワード設定

タグ名	password

UiPath Orchestratorへの接続時のパスワード設定です。

【設定項目】

```
<im-uipath-config>
 <orchestrator id="default" use-insecure-ssl="true">
   :
  <authentication>
   :
    <password>your password</password>
  </authentication>
 </orchestrator>
</im-uipath-config>
```

必須項目	×
複数設定	×
設定値・設定する内容	UiPath Orchestratorへ接続する際のパスワード
単位・型	文字列
省略時のデフォルト値	なし
親タグ	authentication

IM-LogicDesigner タスク説明

項目

ジョブステータス取得

UiPathジョブステータスを取得するタスクです。

> **コラム**
>
> ロジックフロー定義編集画面にて、UiPathのカテゴリおよびタスクのアイコンが表示されない場合、アイコンのリカバリを実行することで表示されるようになる場合があります。
> 詳細は、「IM-LogicDesigner ユーザ操作ガイド」 -「ユーザアイコンをリカバリする」を参照してください。

入力値

```
im_uipathGetJobStatus <object>
 └ id <integer>
```

項目名	必須/任意	型	配列/リスト	説明
im_uipathGetJobStatus	任意	object	なし	-
id	必須	integer	なし	ジョブID

出力値

```
im_uipathGetJobStatus <object>
  ├ batchExecutionKey <string>
  ├ creationTime <string>
  ├ endTime <string>
  ├ hostMachineName <string>
  ├ id <integer>
  ├ info <string>
  ├ inputArguments <map>
  ├ key <string>
  ├ outputArguments <map>
  ├ releaseName <string>
  ├ source <string>
  ├ sourceType <string>
  ├ startTime <string>
  ├ startingScheduleId <integer>
  ├ state <string>
  └ type <string>
```

項目名	型	配列/リスト	説明
im_uipathGetJobStatus	object	なし	-
batchExecutionKey	string	なし	バッチの実行ID
creationTime	string	なし	ジョブが作成された日時
endTime	string	なし	ジョブの完了日時
hostMachineName	string	なし	プロセスが実行されたマシン名
id	string	なし	ジョブID
info	string	なし	ジョブの実行状態に関する補足説明
inputArguments	map	なし	ジョブ実行時の入力値
key	string	なし	ジョブの実行ごとに一意となる文字列
outputArguments	string	なし	ジョブ実行結果の出力値
releaseName	string	なし	プロセス名
source	string	なし	ジョブのソースコード
sourceType	string	なし	ジョブのソースタイプ
startTime	string	なし	ジョブの開始日時
startingScheduleId	integer	なし	スケジュールID
state	string	なし	ステータス
type	string	なし	タイプ

コラム

ジョブのステータスについてはUiPath Orchestratorのガイドを参照してください。
「UiPath Orchestrator ジョブステータス」を参照してください。

IM-LogicDesigner ユーザ定義説明

項目

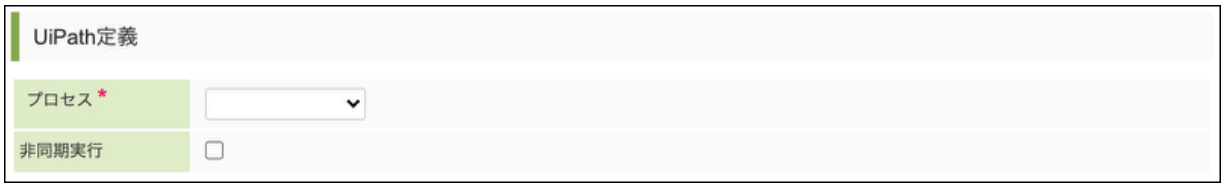
UiPath定義の新規作成

IM-LogicDesignerのユーザ定義として、UiPathプロセス実行タスクが作成できます。
以下の手順で、IM-LogicDesignerにユーザ定義を追加してください。

1. ユーザ定義一覧より、UiPath定義を新規作成します。
   詳細な手順は、「IM-LogicDesigner ユーザ操作ガイド」 - 「ユーザ定義を新規登録する」を参照してください。

2. 共通設定を定義します。
   「入力値」および「返却値」については、次項のプロセスを選択することで自動設定します。

3. UiPath 定義の詳細情報を定義します。



<画面項目>

項目	説明
プロセス	対象のプロセスを選択します。
非同期実行	プロセス実行を非同期で実行する場合に選択します。

コラム

利用可能なデータの型は以下の5種類です。（括弧内はIM-LogicDesignerでの取り扱いです。）
多次元配列の取り扱いは未対応です。
String(String)
Int32(Integer)
Boolean(Boolean)
DateTime(IMDateTim)
Double(Double)

UiPath連携チュートリアル

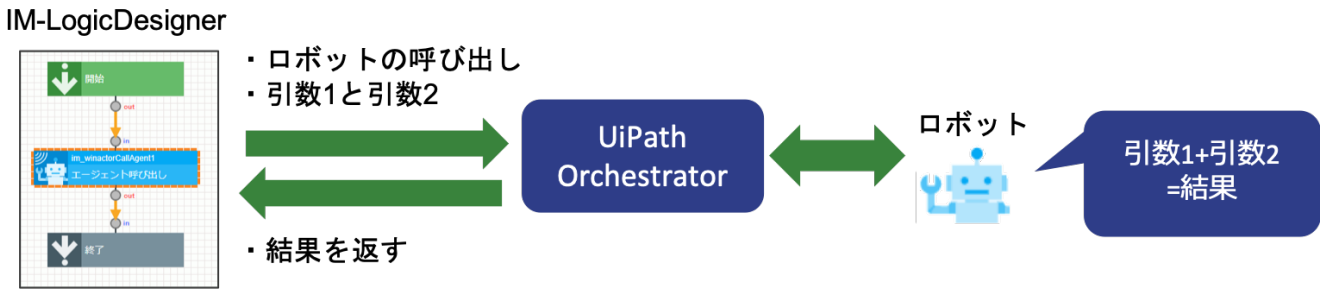
項目

チュートリアルの概要

本章では、IM-RPAのUiPath連携機能を使用して、UiPathのプロセスを実行する方法をチュートリアル形式でご説明します。
このチュートリアルに沿って設定を行うことで、以下のような機能を実現できます。

- intra-mart Accel PlatformからIM-LogicDesignerを介してUiPathのプロセスを実行します。
- UiPathのプロセスに対してパラメータを渡し、計算した結果を受け取ります。

本章では、説明を簡単にするために、ロボットのシナリオはシンプルなものにしております。
シナリオの実行イメージは以下の通りです。



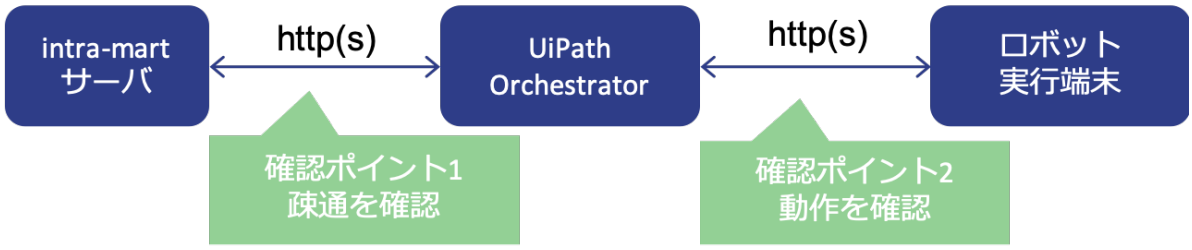
図：本チュートリアルシナリオ実行イメージ

準備・環境設定

UiPath環境の確認

UiPathのUiPath OrchestratorとUiPath Studioが利用できる環境を準備します。
あらかじめ、 UiPath Orchestratorからプロセスの実行ができることをご確認ください。
UiPath Orchestratorはintra-mart Accel Platformサーバとhttpポートで通信できる必要があります。

構成イメージ



図：構成イメージ

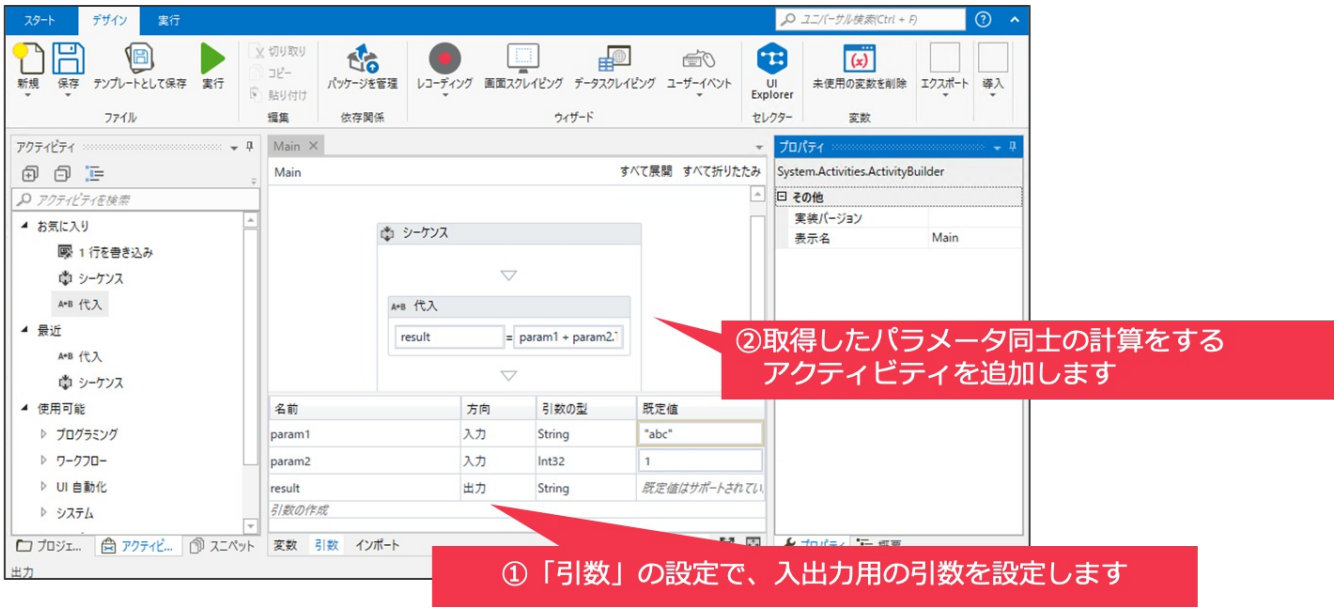
UiPath連携のセットアップ

IM-RPA利用ガイドの「UiPath連携」に従って、 intra-mart Accel Platformのセットアップおよび設定ファイルを記載します。

詳細は、「セットアップ」を参照してください。

プロセスの作成

UiPath Studioにてプロセスを作成します。
以下は、プロセスの例です。



図：UiPath Studio作成イメージ

> **コラム**
>
> UiPath Studioを使用したプロセスの作成方法については、 「UiPath Studio ガイド」を参照してください。

パラメータの設定

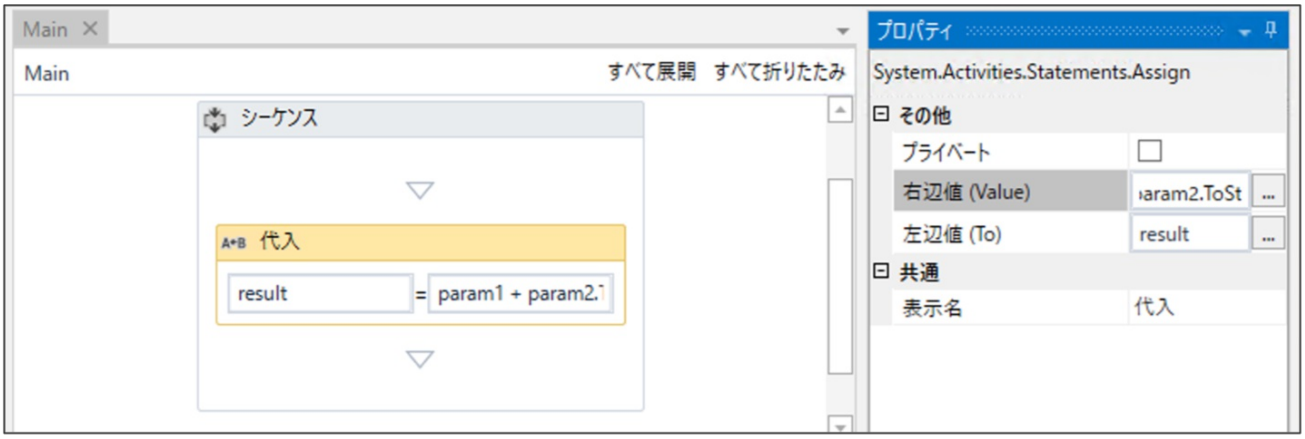
上記「①『引数』の設定で、入出力用の引数を設定します」部分でパラメータ設定を行います。

図：UiPath Studio引数の設定イメージ

> **コラム**
>
> UiPath連携における、双方のパラメータの型は以下のようにマッピングされます。
> 記載以外のパラメータの型については、対応しておりません。
>
> | UiPath | | intra-mart Accel Platform　備考 |
> |---|---|---|
> | String | <-> | String |
> | Int32 | <-> | Integer |
> | Boolean | <-> | Boolean |
> | DateTime | <-> | IMDateTime |
> | Double | <-> | Double |
>
> なお、IM-LogicDesigner側でパラメータを受け取る際に別のデータ型を指定した場合、自動的に型変換をします。
> 具体的な対応表は、「IM-LogicDesigner仕様書」 - 「IM-LogicDesigner データ型変換 仕様書」を参照してください。

アクションの設定

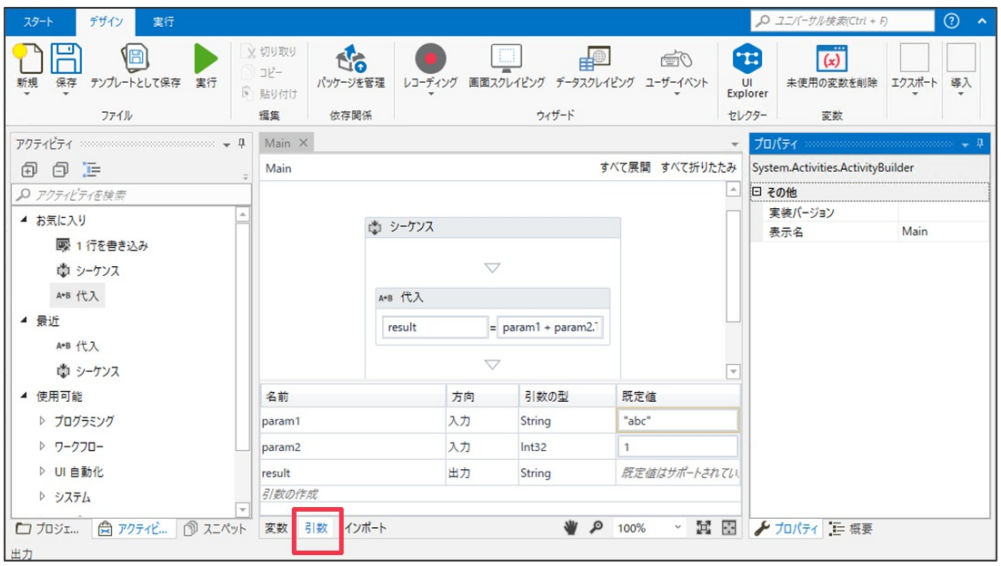
上記「②取得したパラメータ同士の計算をするアクティビティを追加します」部分でプロセスのアクションを設定します。
「代入」アクティビティを以下のように設定します。



図：UiPath Studioアクティビティの設定 - 「代入アクティビティ」

パラメータの設定

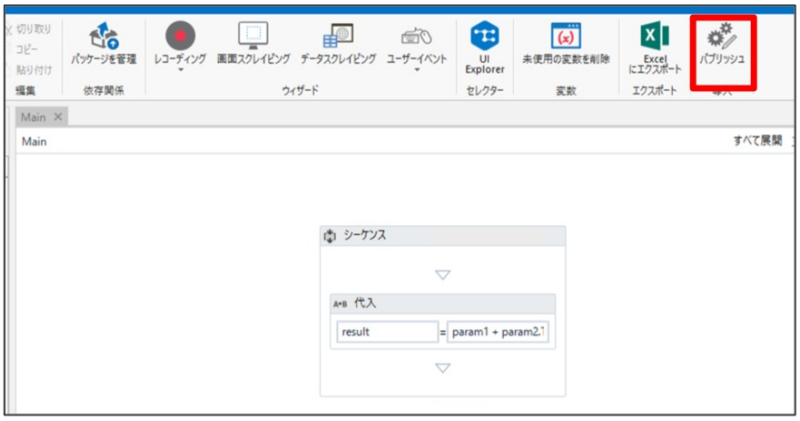
プロセスで使用する変数のうち、IN/OUTで使用するパラメータについては、 UiPath Studioにて、プロセスの使用する「引数」として設定してください。
「変数」に設定した場合、パラメータの連携が行えません。



図：UiPath Studio変数と引数

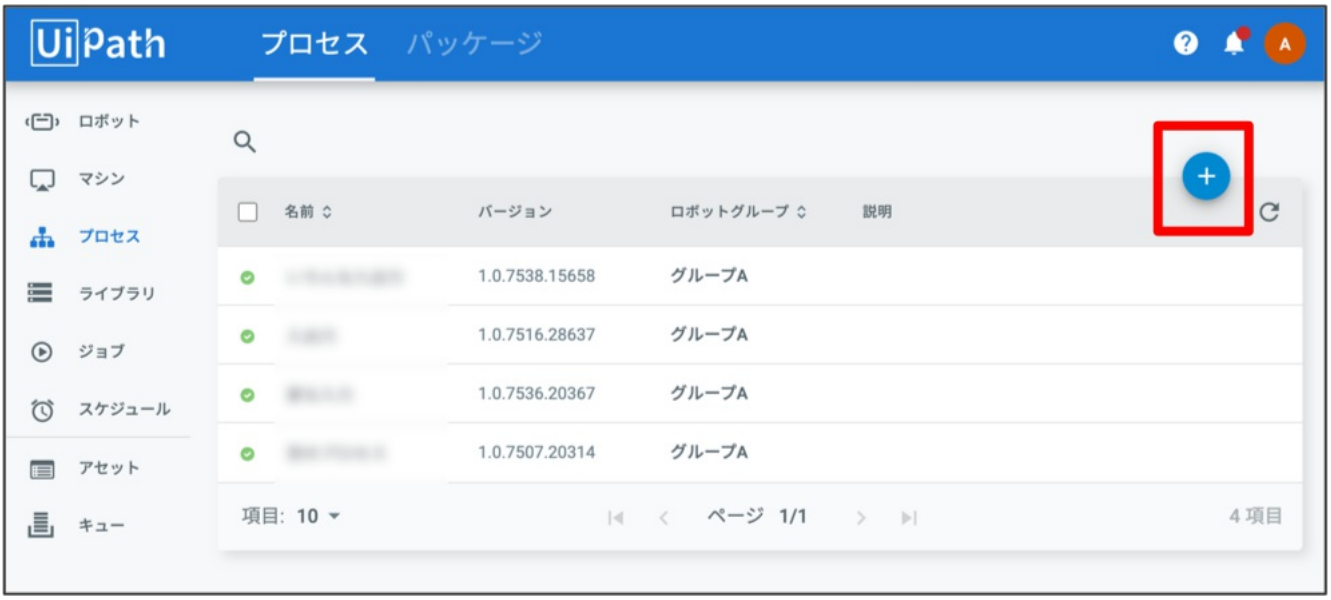
プロセスのデプロイ

UiPath Studioの「パブリッシュ」ボタンを押下し、UiPath Orchestratorにプロセスをアップロードしてください。



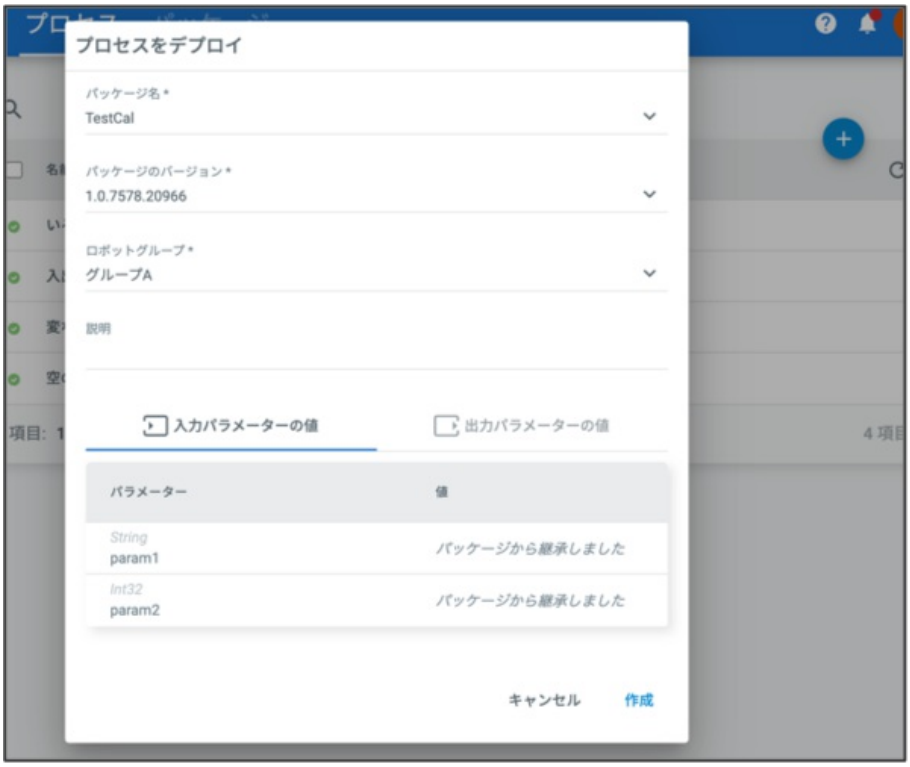
図：UiPath Studioプロセスのパブリッシュ

UiPath Orchestratorにログインし、プロセス画面より先ほどパブリッシュしたプロセスをデプロイしてください。



図：UiPath Orchestratorプロセスをデプロイ

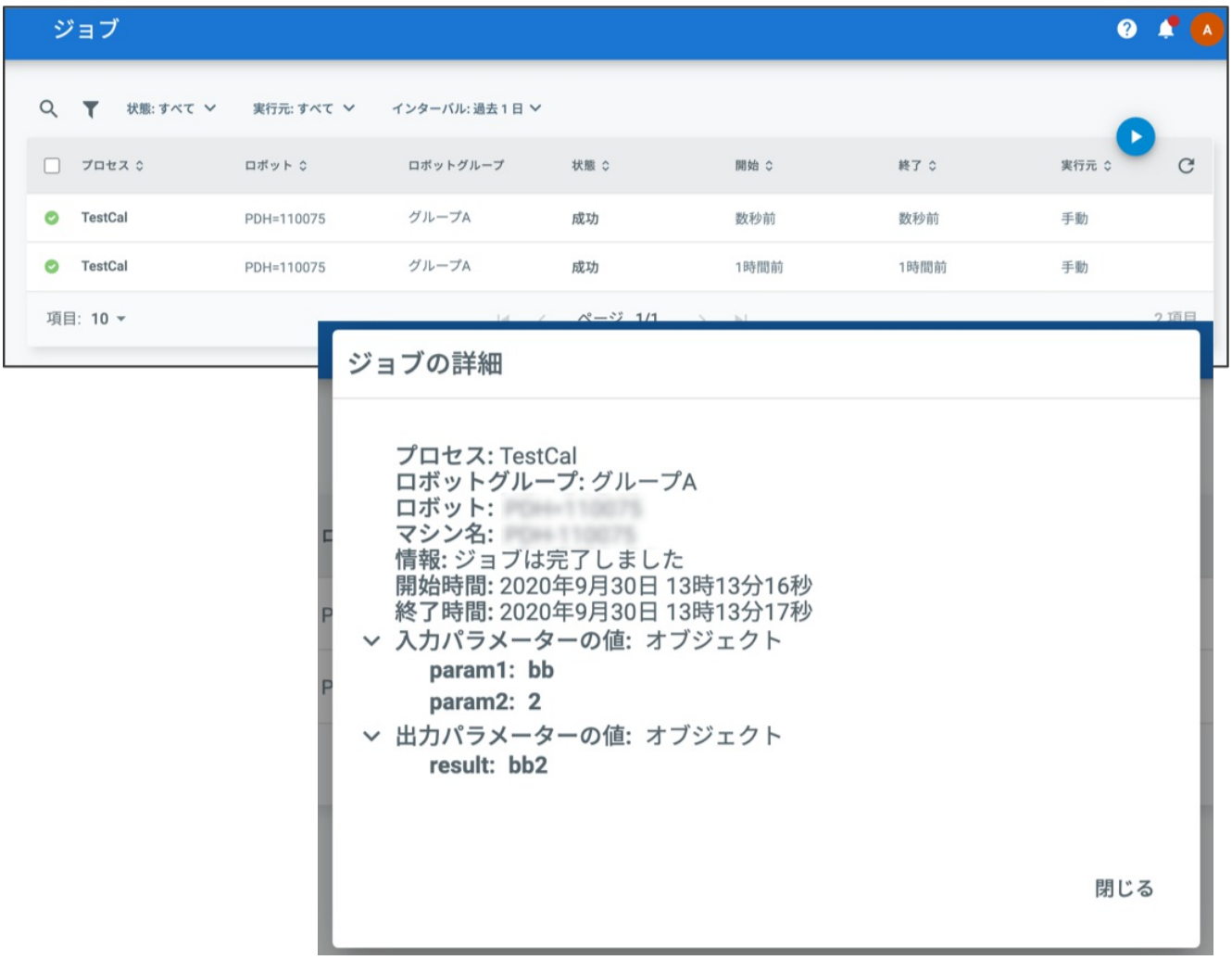
この際、入出力パラメータが反映されていることを確認してください。



図：UiPath Orchestrator入力パラメータの確認

UiPath Orchestrator上での動作確認

UiPath Orchestrator上のジョブで、対象のプロセスを実行し、正常に動作することを確認してください。



図：UiPath Orchestratorジョブの実行

UiPathプロセスの作成から動作確認までの具体的な操作の流れについては、動画にてご覧いただけます。

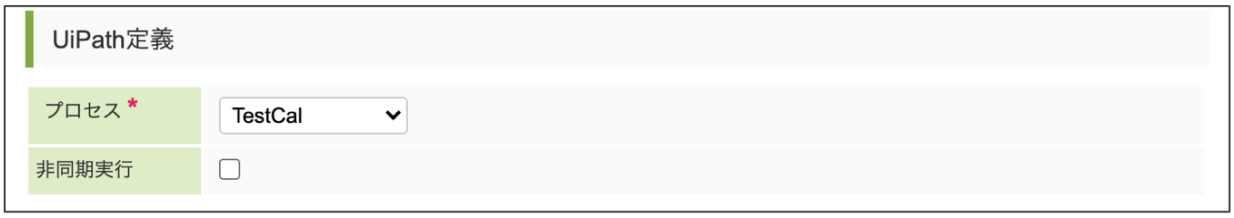
IM-LogicDesignerユーザ定義タスクの準備

「ユーザ定義タスク」とは、IM-LogicDesignerにおいて作成可能な、各種機能を持つ独自タスクです。
ユーザ定義タスクの具体的な説明については、「IM-LogicDesigner仕様書」 - 「ユーザ定義タスク」を参照してください。
本チュートリアルでは、 1つのユーザ定義タスクを用意いたします。

ユーザタスク種別	内容
UiPath定義	UiPath Orchestratorを介してUiPathのプロセスを実行します。 実行時に任意のパラメータの送受信を行います。

ユーザ定義作成（UiPath定義）

「サイトマップ」→「LogicDesigner」→「ユーザ定義」→「UiPath定義新規作成」をクリックします。
UiPath定義を下記のように編集します。
UiPath定義カテゴリより、対象のプロジェクト、およびプロセスを選択してください。



図：「UiPath定義新規作成」 - 「UiPath定義」

ユーザ定義タスク（UiPath定義）の具体的な説明については、「*IM-LogicDesigner ユーザ定義説明*」を参照してください。

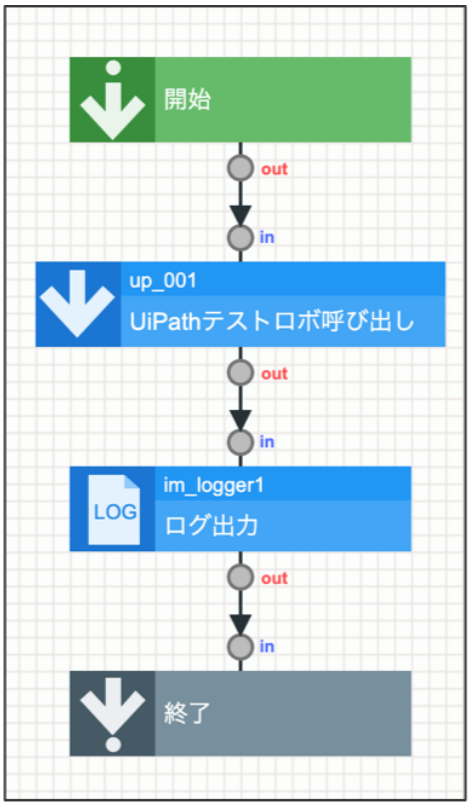
プロセスを選択すると、UiPath Orchestratorを介してプロセスに必要な入出力パラメータを自動的に取得し、「入力値」および「返却値」に表示されます。
ここで入出力のパラメータを修正する必要はありません。

図：「UiPath定義新規作成」 -「入力値/出力値」

「ユーザ定義名」など適宜設定し、UiPath定義を登録します。

IM-LogicDesignerフローの呼び出し

「サイトマップ」→「LogicDesigner」→「フロー定義一覧」→「新規作成」をクリックします。
以下は、UiPath連携（パラメータ使用）を行うための最もシンプルなIM-LogicDesignerフローです。



図：完成イメージ（ロジックフロー）

具体的な手順は以下の通りです。

項目

- タスクとフローを設定します
- プロセスに連携するパラメータを設定します
- データマッピングをします
- デバッグ実行で動作を確認します

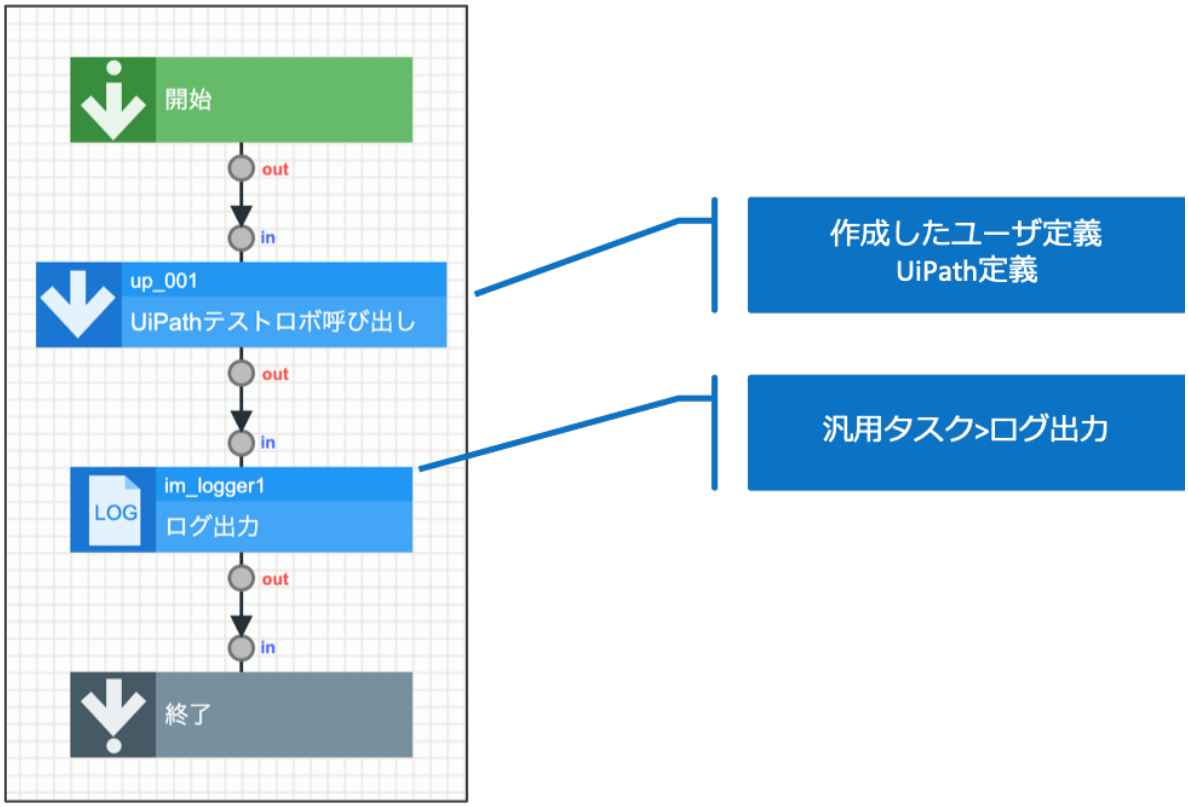
タスクとフローを設定します

パレットから各タスクを選択し、今回使用するタスクをキャンパスに配置します。

前項で作成したユーザ定義は、パレット内の「前項で設定したユーザカテゴリ>作成したユーザ定義名」にあります。

図：IM-LogicDesignerタスク選択

配置する各タスクは、以下の通りです。



図：IM-LogicDesigner配置タスク

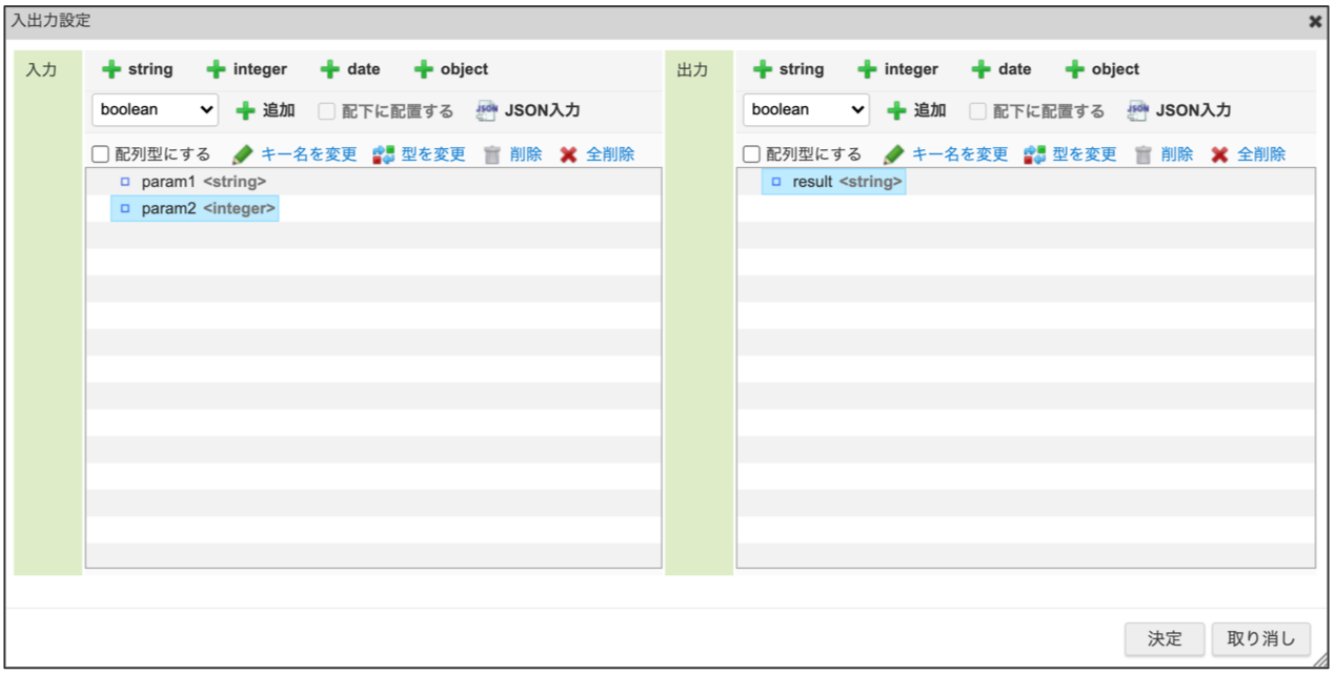
次に、配置したタスク同士をフローで繋ぎます。

プロセスに連携するパラメータを設定します

IM-LogicDesignerの「入出力設定」を行います。

設定値の説明（入出力設定）

カテゴリ	設定値	説明
入力	param1	プロセスに渡す引数1
入力	param2	プロセスに渡す引数2
出力	result	プロセスからの返り値

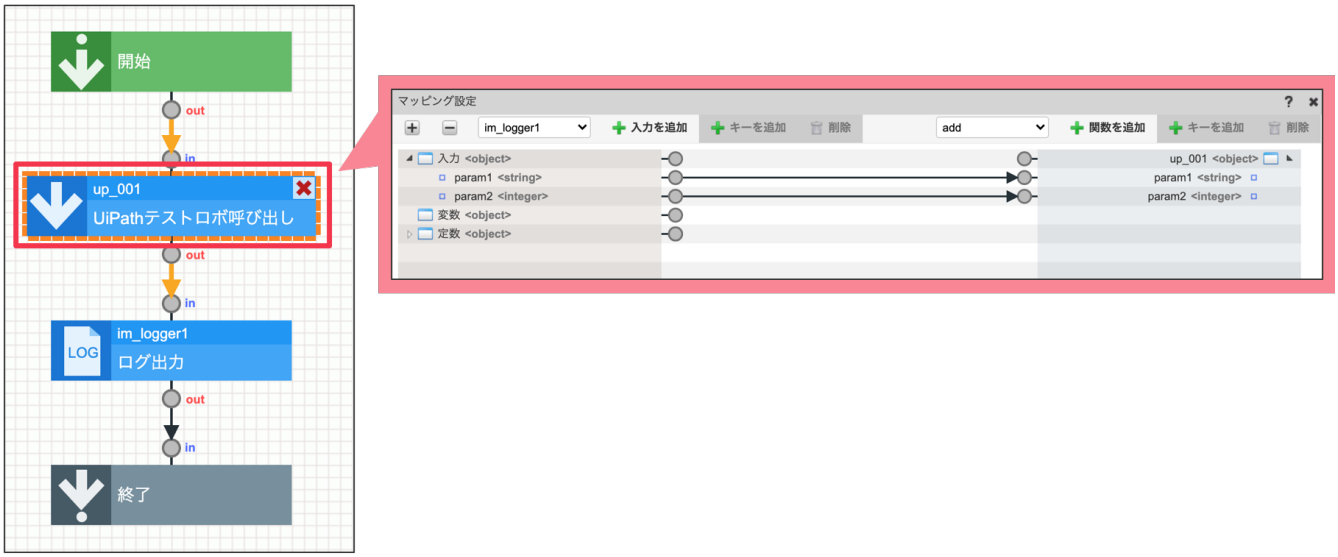


図：IM-LogicDesigner- 「入出力設定」

データマッピングをします

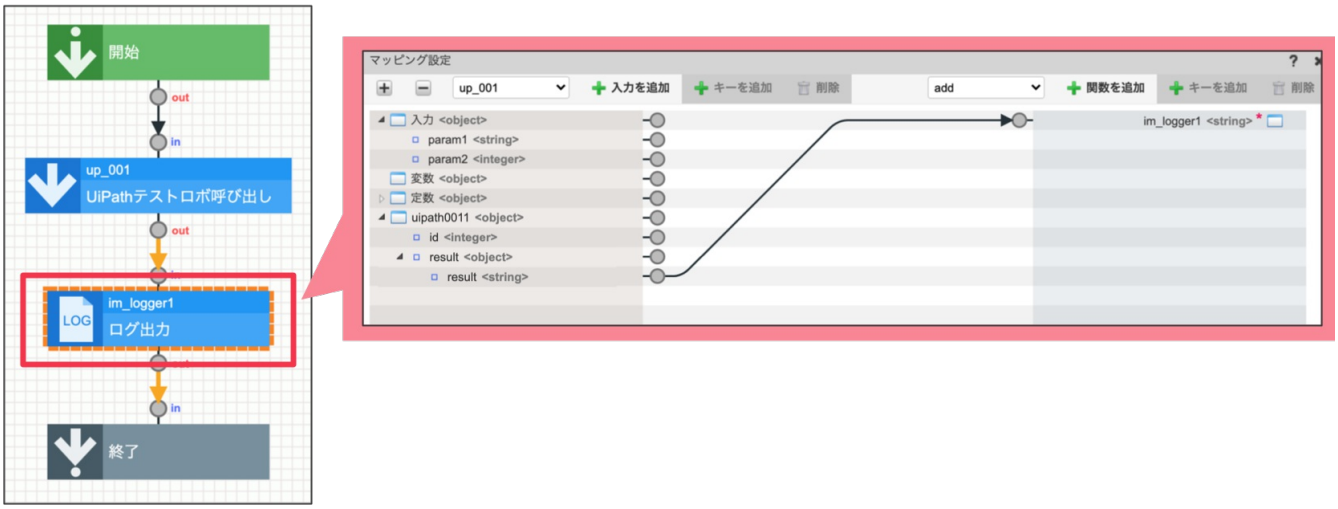
各タスクをダブルクリックして、「データマッピング」の設定を行います。
各図を参考にマッピングをしてください。

プロセス実行タスク（作成したユーザ定義 UiPath定義）のデータマッピングを行います。



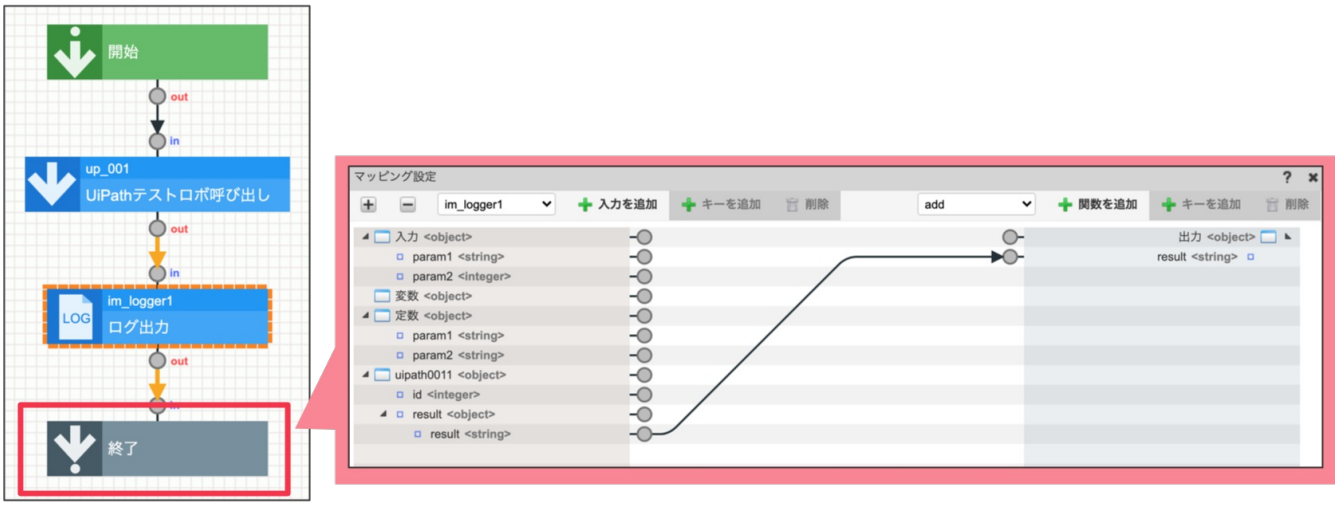
図：データマッピングの設定 - プロセス実行タスク

ログ出力（「汎用タスク」　-「ログ出力」）のデータマッピングを行います。



図：データマッピングの設定 - ログ出力

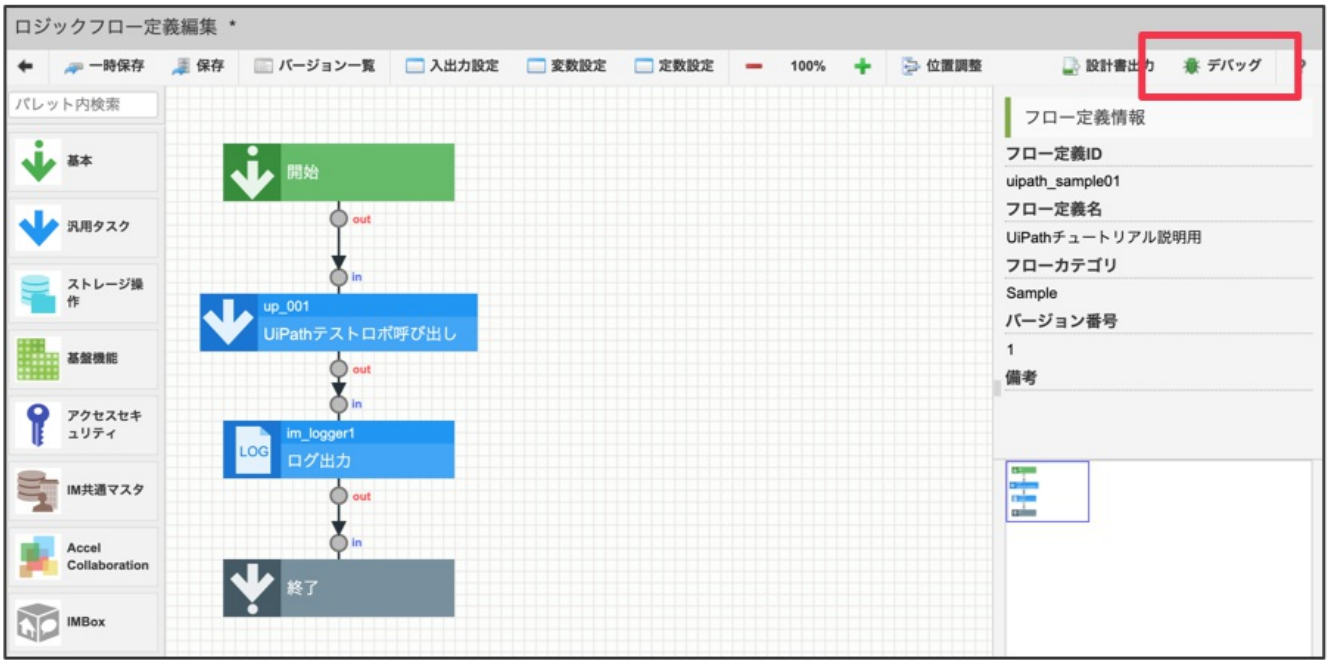
プロセスの実行結果resultを、フローの出力用変数にマッピングします。



図：データマッピングの設定 - 終了タスク

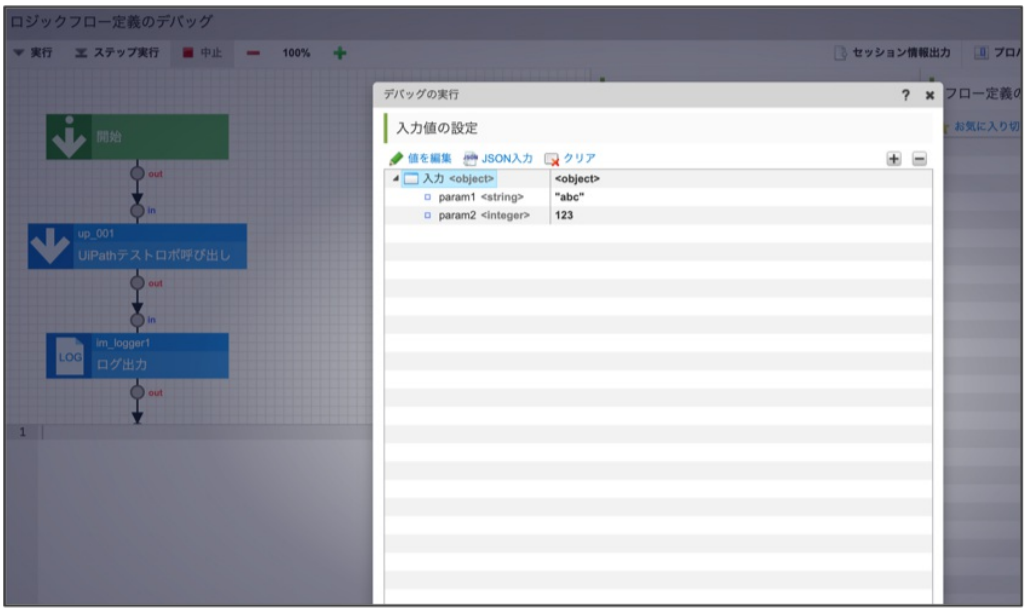
デバッグ実行で動作を確認します

ロジックフロー定義編集画面の「デバッグ」をクリックし、作成しているロジックフローをデバッグ実行します。



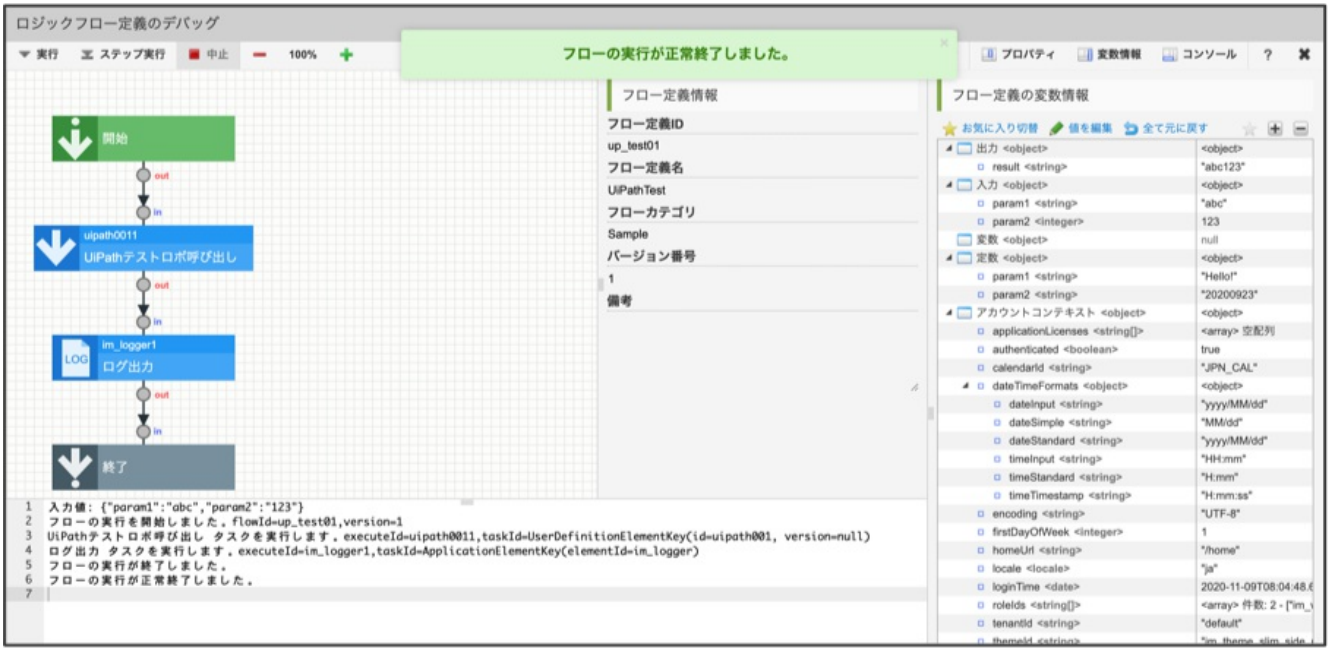
図：IM-LogicDesigner- 「デバッグ実行」

デバッグ画面にて、「実行」をクリックします。
デバッグ実行時の入力値の設定を行います。プロセスに引き渡すパラメータとなる、param1とparam2を設定してください。



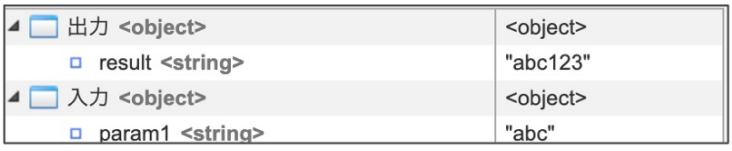
図：IM-LogicDesigner- 「実行」- 「入出力値の設定」

デバッグ実行が正常に完了することを確認します。



図：IM-LogicDesigner- デバッグ実行実行の成功

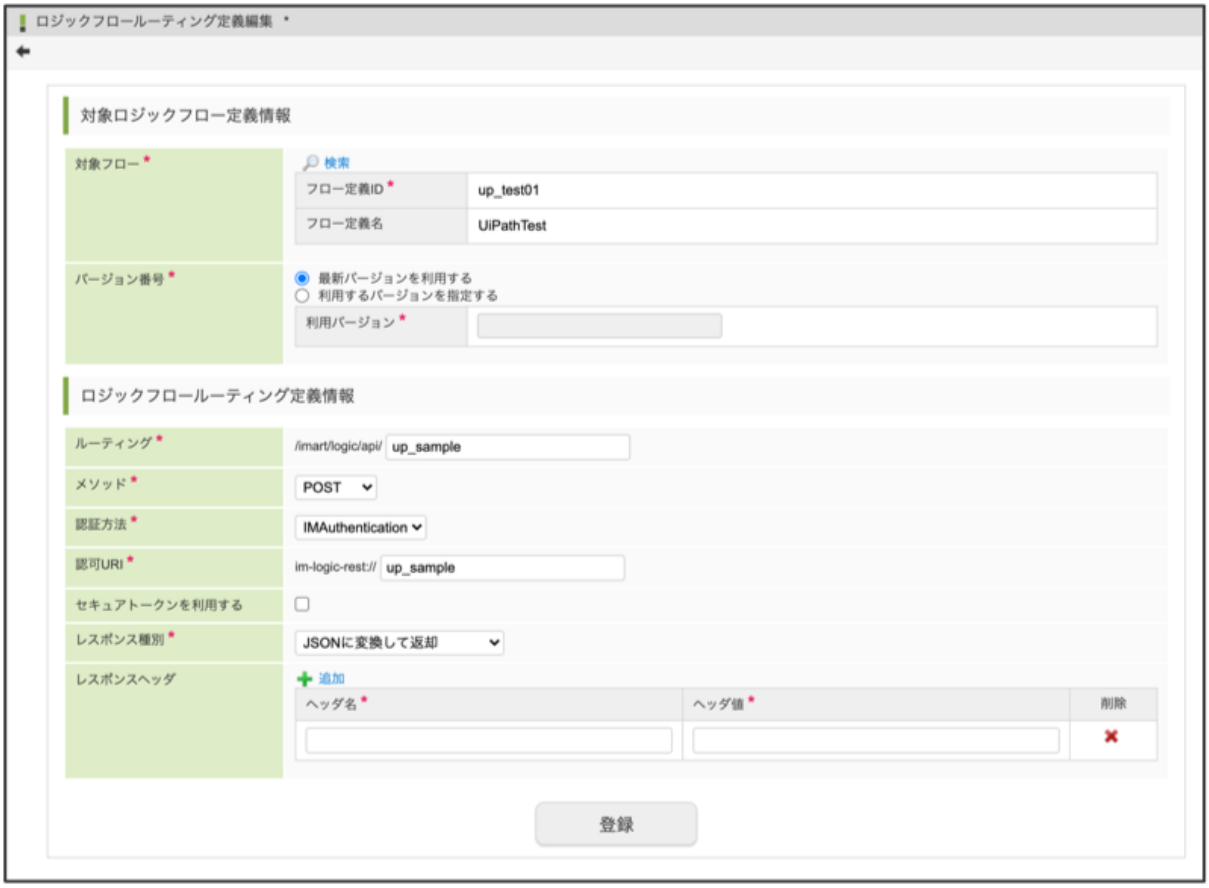
プロセスからの返却値となる、resultが正しく表示されていることを確認します。



図：IM-LogicDesigner- 「フローの変数情報（デバッグ実行後）」

フロールーティングの設定

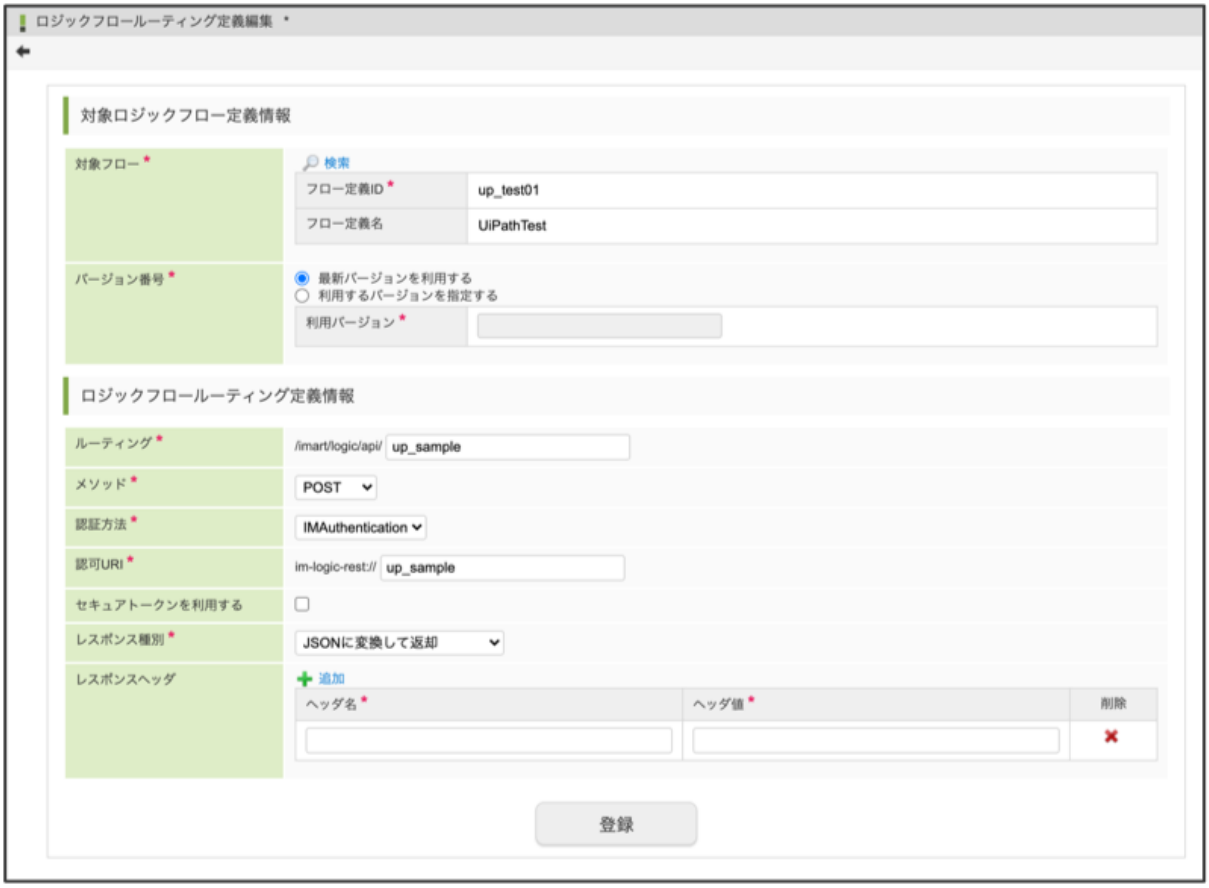
作成したIM-LogicDesignerのフローを、intra-mart Accel Platform他機能やREST経由で実行するため、フロールーティングの設定を行います。
ルーティングに対する認可設定も合わせて行ってください。
フロールーティングの設定については、「IM-LogicDesigner仕様書」 - 「フロールーティングの認可設定」を参照してください。



図：フロールーティングの設定

フロールーティングの設定

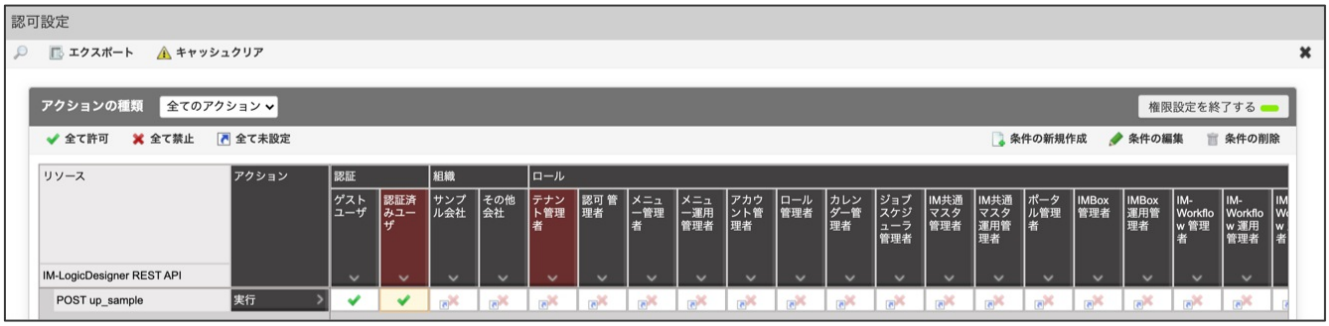
作成したIM-LogicDesignerのフローを、intra-mart Accel Platform他機能やREST経由で実行するため、フロールーティングの設定を行います。
「サイトマップ」→「LogicDesigner」→「ルーティング定義一覧」→「新規作成」をクリックします。
フロールーティングの設定については、「IM-LogicDesigner仕様書」 - 「フロールーティングの認可設定」を参照してください。



図：フロールーティングの設定

> **コラム**
>
> メソッドには「POST」を選択してください。

ロジックフロールーティング定義一覧より「認可」をクリックして、認可設定を行います。
「認証済みユーザ」へ実行権限を付与します。



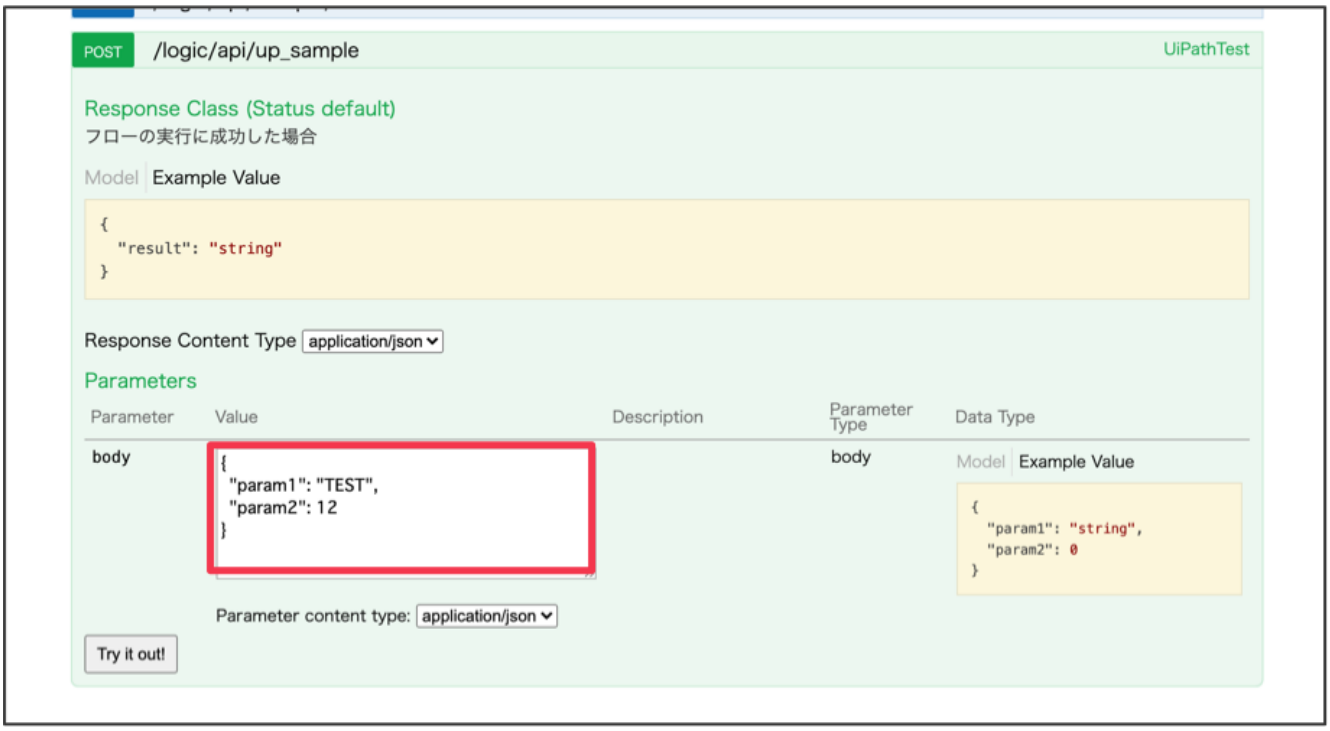
図：認可設定

> **コラム**
>
> 実運用をする際は適切な権限付与を検討してください。

swaggerで動作確認

ロジックフロールーティング定義一覧より「SPEC」をクリックしてswaggerを表示します。
swaggerの設定については、「IM-LogicDesigner仕様書」 - 「Swaggerの利用」を参照してください。

リクエストbodyを適宜設定し、「Try it out!」をクリックします。



図：swaggerでリクエストを設定

リクエストの実行後、レスポンスコードが200で返ること、およびレスポンス内のパラメータresultが想定通りであることを確認してください。

図：swaggerでレスポンスを確認

フロールーティングの設定からswaggerでの動作確認の具体的な操作の流れについては、動画にてご覧いただけます。

ダッシュボード作成チュートリアル

チュートリアル

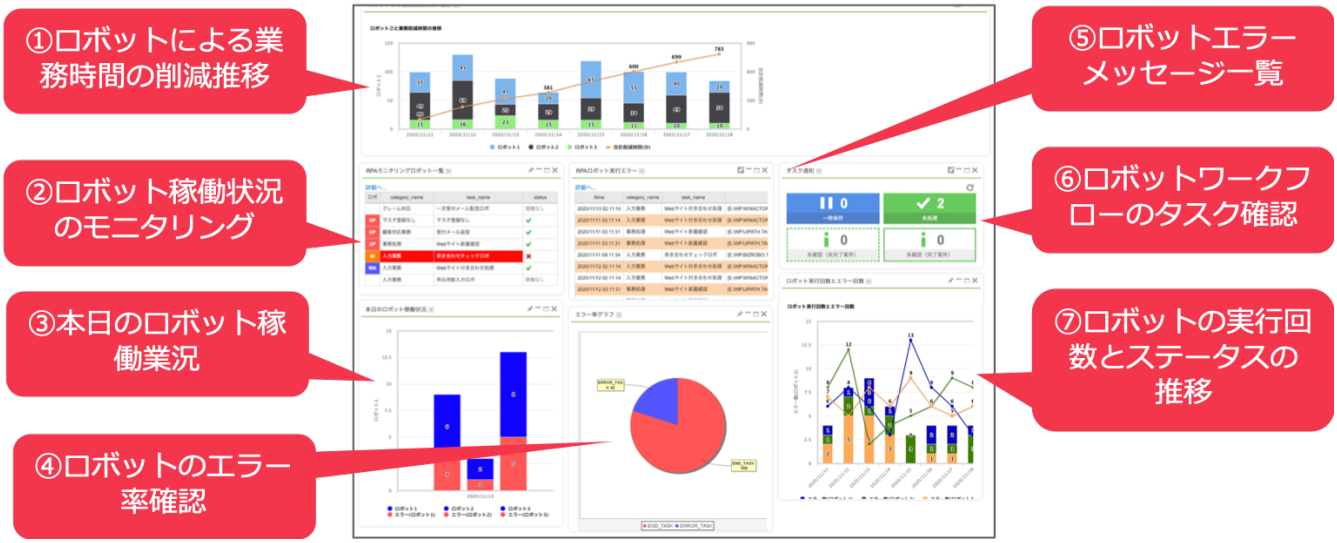
項目

チュートリアルの概要

本章では、IM-RPAのログを利用して、下図のようなロボット管理用のダッシュボードを作成する方法をチュートリアル形式でご説明します。
ダッシュボードに表示するグラフは、自由に設定することが可能です。

図：本チュートリアルで作成可能なダッシュボード

各グラフの内容は下図の通りです。
必要なグラフを自由に選択することが可能です。
また、グラフの表示は適宜変更可能です。



図：ダッシュボード各グラフの説明

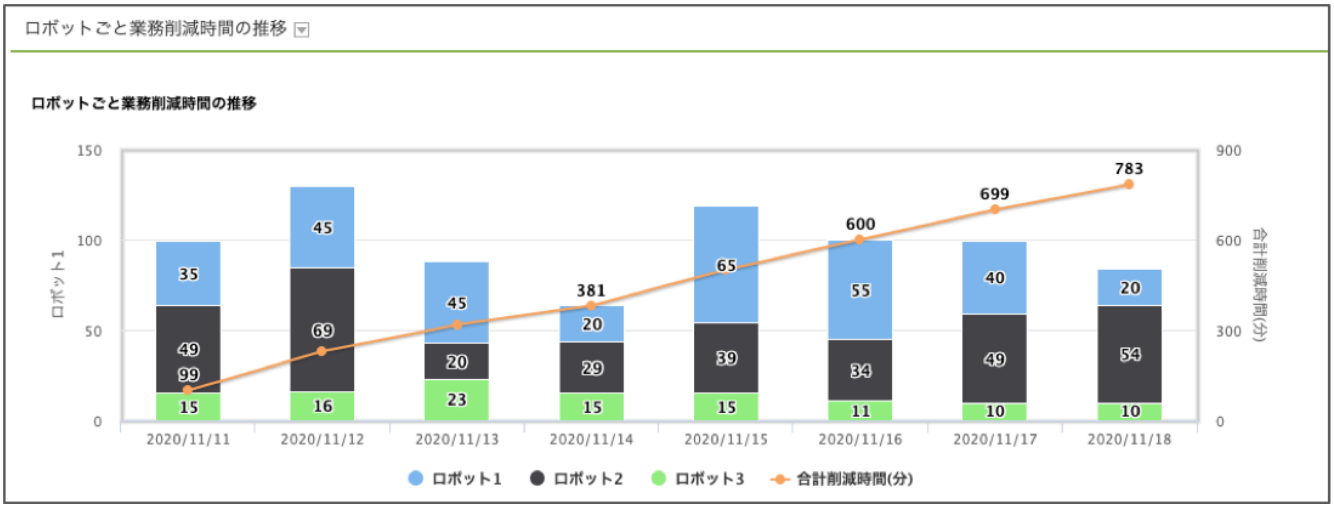
本チュートリアルの利用方法について

本章の記載内容は、RPAダッシュボードを作成するためのサンプルです。
利用者が、本章を参考に業務で必要となるグラフを適宜選定、および、変更を行うことを想定しております。
IM-RPAのログには、ロボット実行タスクの状態や結果、経過時間を格納しており、本章に紹介するグラフ以外にも、様々な表現が可能です。

各グラフの詳細

①ロボットによる業務時間の削減推移

ロボットが業務を代行することにより実現した、業務時間の削減効果を日ごとの推移でグラフ化したものです。
ロボットごとの業務削減時間（分）を分けて積み上げ棒グラフで見ることで、RPA導入のROIを視覚的に確認できます。
折れ線グラフによりトータルの削減時間推移を確認することも可能です。



図：①ロボットによる業務時間の削減推移 グラフ

②ロボット稼働状況のモニタリング

直近のロボット実行状況をモニタリングする一覧です。
異なるRPA製品をまたがって、全てのロボットの状況を確認することが可能です。

ロボ	category_name	task_name	status
	クレーム対応	一次受付メール配信ロボ	稼働なし
UP	マスタ登録なし	マスタ登録なし	✔
UP	顧客対応業務	受付メール返信	✔
UP	事務処理	Webサイト新着確認	✔
B!	入力業務	突き合わせチェックロボ	✖
WA	入力業務	Webサイト付き合わせ処理	✔
	入力業務	申込用紙入力ロボ	稼働なし

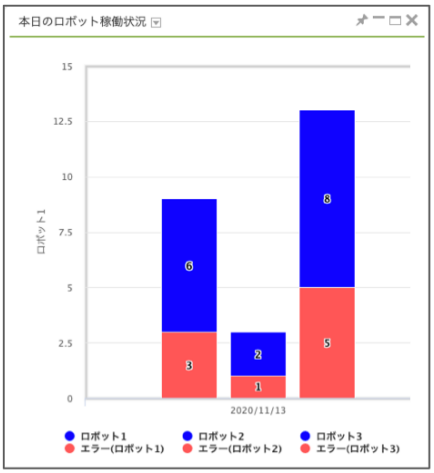
図：②ロボット稼働状況のモニタリング

各列の説明

項目名	内容
ロボ	RPA製品の略称 ▪ WA：winactor ▪ B!：BizRobo! ▪ UP：UiPath
category_name	業務名
task_name	タスク名
status	ロボットの状態 ▪ 正常（チェック） ▪ エラー（×） ▪ 直近の稼働なし ▪ 遅延（実行時間が10分以上）

③本日のロボット稼働業況

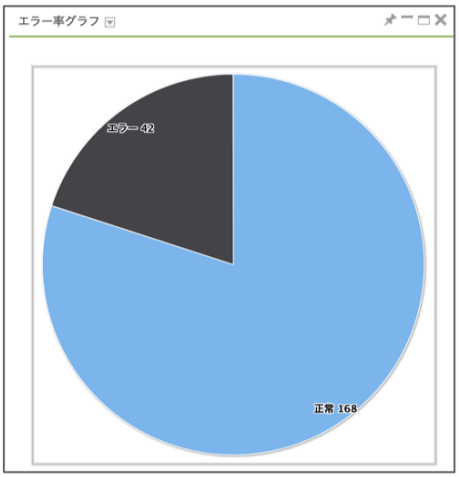
当日のロボットの実行回数、エラー回数をまとめて確認できるグラフです。
ロボットごとの実行回数を棒グラフで確認可能です。
それぞれのロボットごとに、正常/エラーの内訳を確認することが可能です。



図：③本日のロボット稼働業況

④ロボットのエラー率確認

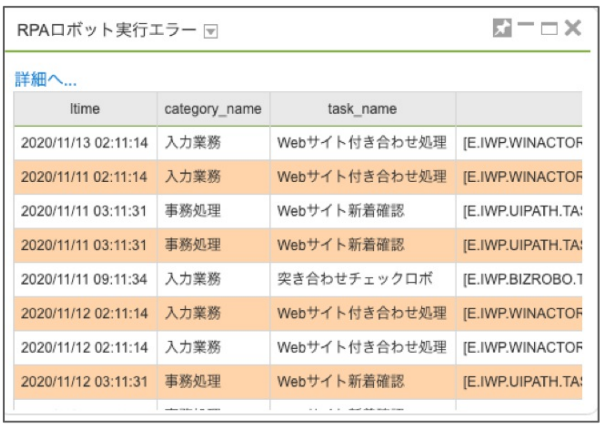
直近数日間のロボット実行状況を確認するグラフです。



図：④ロボットのエラー率確認

⑤ロボットエラーメッセージ一覧

直近の、IM-RPAで実行したロボットで発生したエラーメッセージを表示します。

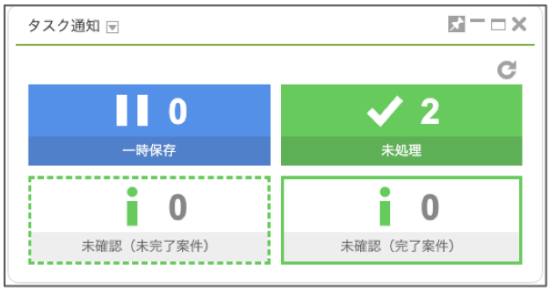


ltime	category_name	task_name	
2020/11/13 02:11:14	入力業務	Webサイト付き合わせ処理	[E.IWP.WINACTOR
2020/11/11 02:11:14	入力業務	Webサイト付き合わせ処理	[E.IWP.WINACTOR
2020/11/11 03:11:31	事務処理	Webサイト新着確認	[E.IWP.UIPATH.TA
2020/11/11 03:11:31	事務処理	Webサイト新着確認	[E.IWP.UIPATH.TA
2020/11/11 09:11:34	入力業務	突き合わせチェックロボ	[E.IWP.BIZROBO.T
2020/11/12 02:11:14	入力業務	Webサイト付き合わせ処理	[E.IWP.WINACTOR
2020/11/12 02:11:14	入力業務	Webサイト付き合わせ処理	[E.IWP.WINACTOR
2020/11/12 03:11:31	事務処理	Webサイト新着確認	[E.IWP.UIPATH.TA

図：⑤ロボットエラーメッセージ一覧

⑥ロボットワークフローのタスク確認

IM-RPA連携により、ワークフローの起票をロボットが行ったり、ワークフロー中の手続きをロボットが代行することが可能です。
ワークフローのタスク通知機能を使うことで、ロボットが関連するワークフローのタスク状況をカンバン形式で確認することが可能です。

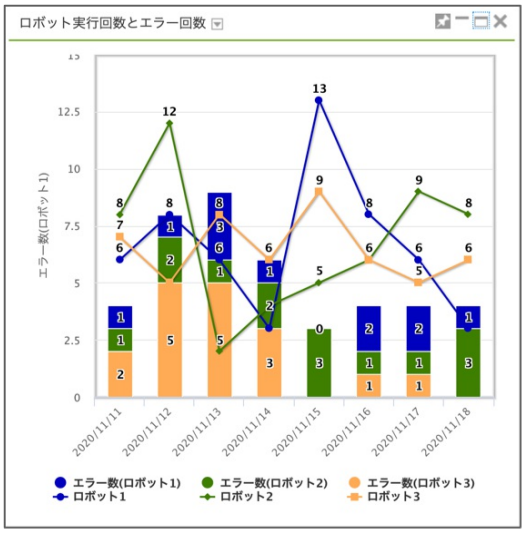


図：⑥ロボットワークフローのタスク確認

> **コラム**
>
> ワークフロータスク通知はIM-Workflowの機能です。
> タスク通知ポートレットの利用方法については、「IM-Workflow 管理者操作ガイド」 - 「タスク通知ポートレット」を参照してください。

⑦ロボットの実行回数とステータスの推移

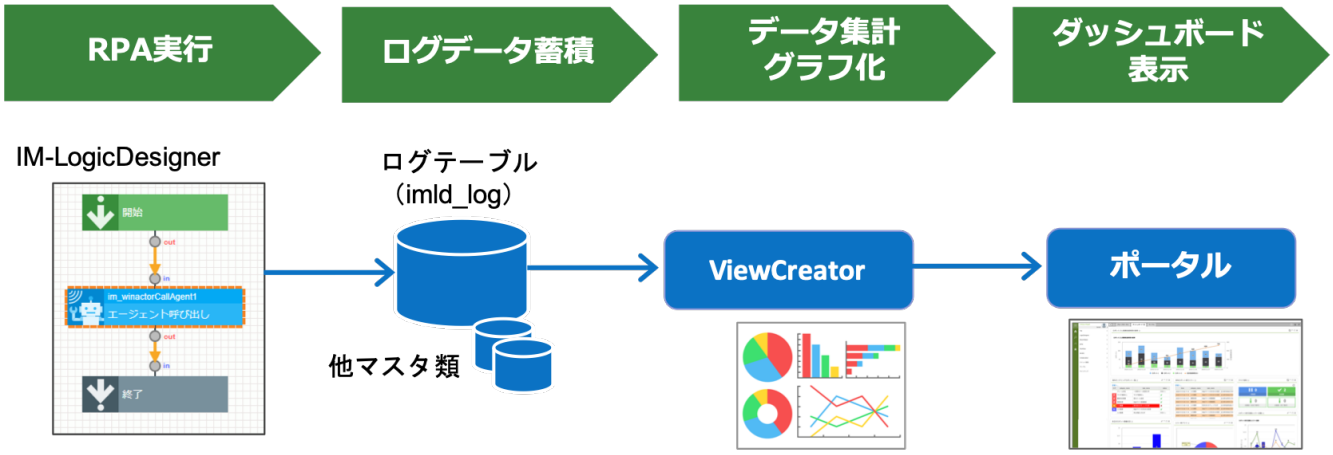
複数のロボットをまとめて、日々の実行回数とエラー状況を俯瞰して確認するグラフです。
折れ線グラフでロボットごとの実行回数の推移を表示します。
積み上げ棒グラフで、ロボットごとのエラー回数の推移を表示します。
各ロボットの状況を俯瞰することで、日々のロボット稼働状況に異常がないか確認することが可能です。



図：⑦ロボットの実行回数とステータスの推移

ダッシュボード実現方式

本チュートリアルのダッシュボード実現には、IM-LogicDesigner、 ViewCreator 、および、ポータルを使用します。
IM-LogicDesignerによって蓄積したログデータを、 ViewCreatorでグラフ化し、ポータルにダッシュボードとして表示します。

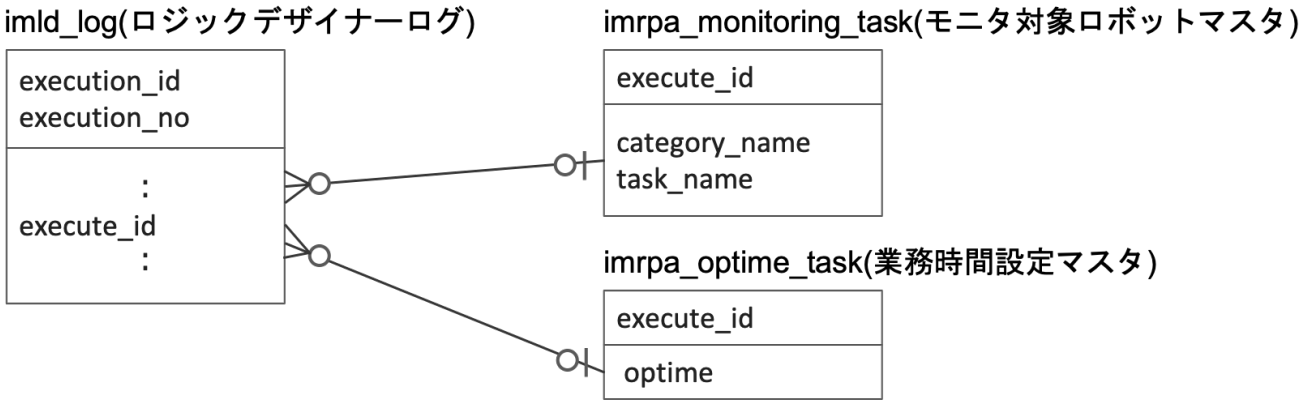


図：ダッシュボード実現方式イメージ

追加するテーブルのリレーション

IM-LogicDesignerのタスク実行ログは、IM-RPAに限らず全てタスク単位で保存されます。
タスク実行ログの詳細について「IM-LogicDesigner仕様書」 - 「グループポータルを管理する」 を参照してください。

本チュートリアルでは、追加で下図の右側の2テーブルを用意します。



図：追加するテーブルとタスク実行ログの関連

> **コラム**
>
> IM-LogicDesignerのタスク実行ログは、2020 Winter(Azalea)以降に利用が可能です。

> **コラム**
>
> タスク実行ログは、運用の環境によっては大量の実行ログが保存されます。
> 著しくログの量が増えた場合、ダッシュボードの表示速度が低下します。
> 本番運用前に性能面の検討をして頂き、必要に応じてタスク実行ログの定期的な削除や、ダッシュボード用ログデータの別保管を検討してください。

追加するテーブルの詳細

各テーブルの仕様は以下の通りです。

imrpa_optime_task

ロボットの業務時間削減に使用するテーブルです。
元々業務に掛かっていた時間の目安を秒単位で指定します。

カラム名	データ型	内容
execute_id	VARCHAR(100)	対象となるロボット実行タスクのタスクID
coptime	DECIMAL(16)	元々業務に掛かっていた時間の目安（秒単位）

imrpa_monitoring_task

モニタリング対象となるロボットの一覧を設定するテーブルです。
グラフ表示用の業務名、ロボット名を設定します。

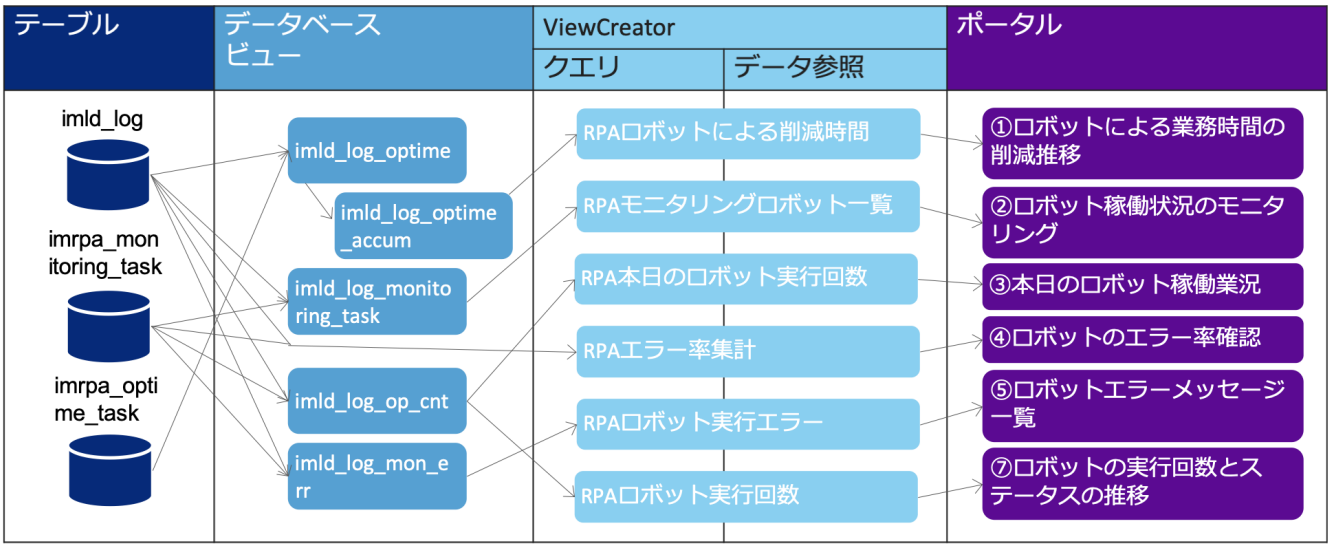
カラム名	データ型	内容
execute_id	VARCHAR(100)	対象となるロボット実行タスクのタスクID
category_name	VARCHAR(300)	ダッシュボードに表示する業務名
task_name	VARCHAR(300)	ダッシュボードに表示するロボット名

データの集計方法

テーブルのグラフ化には、以下のようにデータベースのビューとViewCreator機能を使用します。

利用したいグラフに応じて、必要なテーブル、データベースビュー、ViewCreator機能のみを作成することでも、実現が可能です。
例えば、「*⑤ロボットエラーメッセージ一覧*」のみ利用したい場合、必要な追加テーブルは imrpa_monitoring_task のみです。
他、データベースビューは imld_log_mon_err のみ、ViewCreatorのクエリ・データ参照は「RPAロボット実行エラー」のみ、必要です。

具体的な作成手順は、後述します。



図：データの集計方法

> **コラム**
>
> 「*⑥ロボットワークフローのタスク確認*」についてはIM-Workflowの「タスク通知ポートレット」を利用しているため、説明を割愛します。

ダッシュボード作成手順

ダッシュボードの作成手順は以下の通りです。

手順１　テーブルを作成する

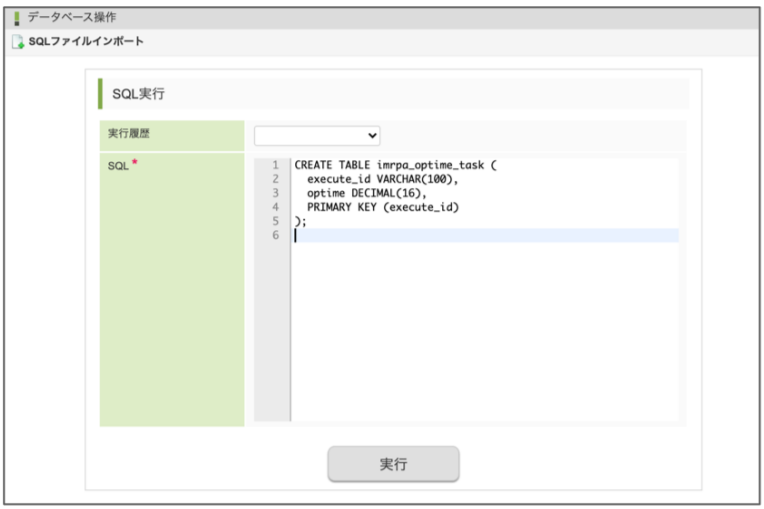
ダッシュボード用のテーブルを2つ作成します。
テナントDB内に追加してください。

```
CREATE TABLE imrpa_optime_task (
  execute_id VARCHAR(100),
  optime DECIMAL(16),
  PRIMARY KEY (execute_id)
);
```

```
CREATE TABLE imrpa_monitoring_task (
  execute_id VARCHAR(100),
  category_name VARCHAR(300),
  task_name VARCHAR(300),
  PRIMARY KEY (execute_id)
);
```

方法は任意ですが、ここではテナント管理者機能である「データベース操作」を使用した方法を紹介します。

1. 「サイトマップ」→「 テナント管理」→「データベース操作」をクリックします。
   テーブル作成用のSQLをセットし、「実行」をクリックします。



図：「データベース操作」画面

2. 「データベース処理が完了しました。 」のメッセージを確認します。

図：処理完了メッセージ

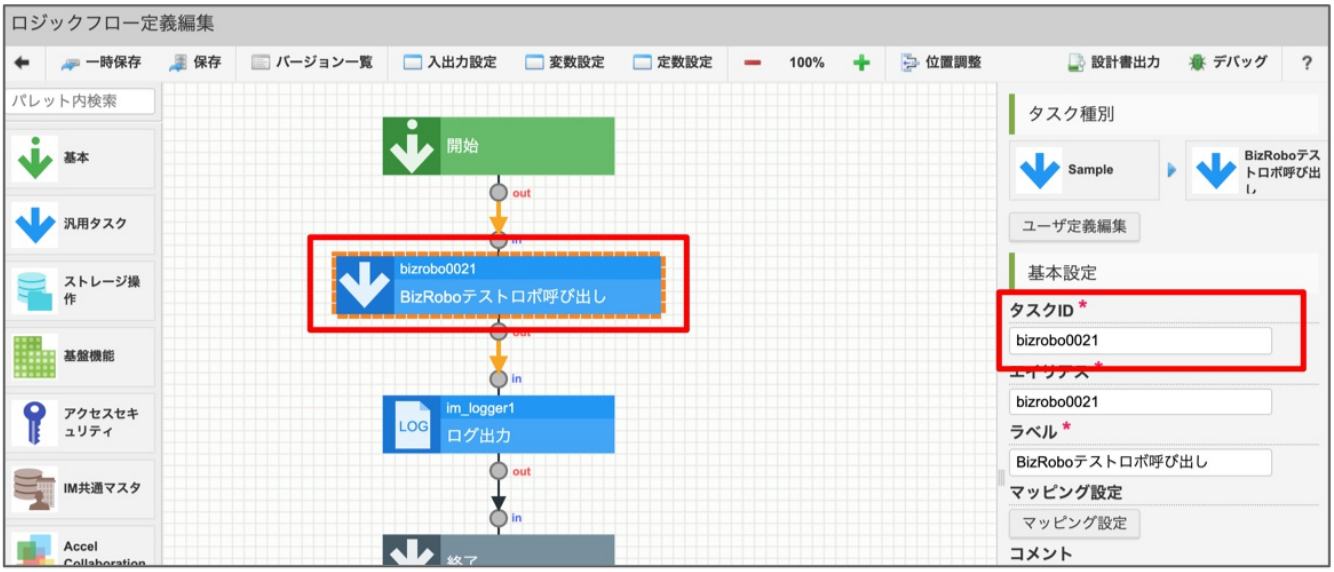
3. 同様の手順で、もう1つのテーブルを作成します。

手順２　モニタリング対象のロボットを確認する

マスタ設定を行うために、IM-RPAで設定しているロボットタスクのIDを確認します。

1. 「サイトマップ」→「LogicDesigner」→「フロー定義一覧」をクリックします。
   ダッシュボードでモニタリングしたいロボット呼び出しのあるロジックフローをクリックします。

2. モニタリングしたいロボット呼び出しタスクをクリックします。
   「基本設定」の「タスクID」に記載されている文字列を控えておきます。



図：タスクIDの確認方法

3. 同様の手順で、モニタリングしたいすべてのロボット呼び出しタスクのタスクIDを控えておきます。

> **コラム**
>
> 上記タスクIDが、当該タスクの実行時にログとして、 imld_log テーブルの execute_id 項目に保存されます。
> execute_id は本チュートリアルで追加するテーブルの主キーです。

手順３　マスタ設定をする

追加したテーブルに対して、モニタリング対象のロボット呼び出しタスクをマスタデータとして登録します。

1. 「サイトマップ」→「 TableMaintenance」→「テーブル一覧」をクリックします。
   imrpa で検索を行い、今回作成したテーブルに絞り込みます。
   imrpa_monitoring_task をクリックします。

図：「テーブル一覧」画面

2. imrpa_monitoring_task の内容を設定します。
   日々状態を監視したいロボットタスクを登録してください。
   execute_id には、先ほど確認したモニタリング対象となるロボット呼び出しのタスクIDを設定します。
   category_name に業務名を、 task_name にロボットタスク名を記載します。

図： imrpa_monitoring_task テーブルの設定例

3. imrpa_optime_task の内容を設定します。
   ロボットの導入効果を測定したいタスクを登録してください。
   execute_id には、先ほど確認したモニタリング対象となるロボット呼び出しのタスクIDを設定します。
   optime には、対象の業務にロボットを導入する前に掛かっていた処理時間を秒単位で設定してください。

図： imrpa_optime_task テーブルの設定例

> **コラム**
>
> imrpa_monitoring_task と imrpa_optime_task に記載する execute_id は異なっても構いません。
> モニタリングしたいロボットと、ROIを計測したいロボットは分けることが可能です。

手順４　テーブルビューを作成する

「*手順１　テーブルを作成する*」と同様の手順で、各テーブルを参照するテーブルビューを作成します。

1. 各テーブルビューを作成するスクリプトのサンプルコードについては、下記リンクより参照してください。
   imld_log_optime ビュー
   imld_log_optime_accum ビュー
   imld_log_monitoring_task ビュー
   imld_log_op_cnt ビュー
   imld_log_mon_err ビュー

   必要なビューの分だけ、スクリプトを実行してください。

図：「データベース操作」画面

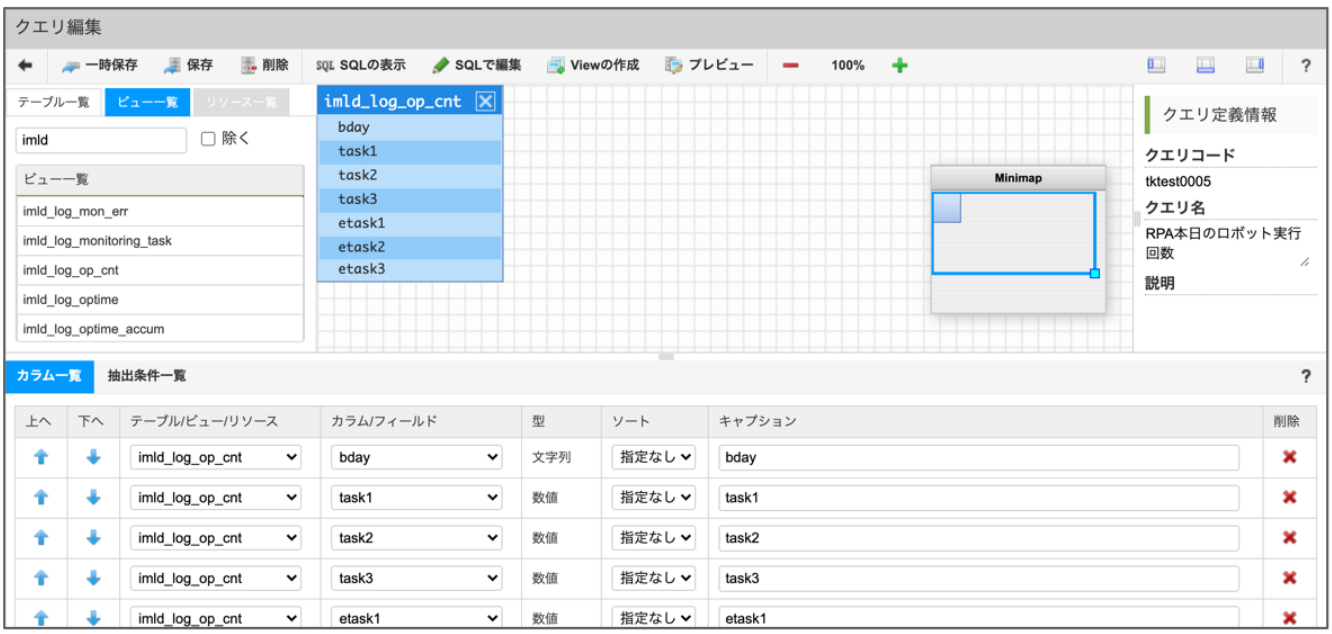
> **コラム**
>
> imld_log_optime 、および、 imld_log_op_cnt のサンプルコード内には execute_id として以下の3つを記述しています。
>
> - bizrobo0021
> - im_winactorCallAgent1
> - uipath0012
>
> 必要に応じて、これらの execute_id をモニタリング対象のタスクIDに差し替えてご利用ください。
> 対象のタスクを増減する場合は、これらの execute_id を含む該当ブロックごと、記述の追加・削除をしてください。

> **コラム**
>
> サンプルコードに含まれるSQLはPostgreSQL用です。テナントDBに他のDBMSを利用している場合は、適宜記述内容の変更をしてください。

手順５　ViewCreatorのクエリを作成する

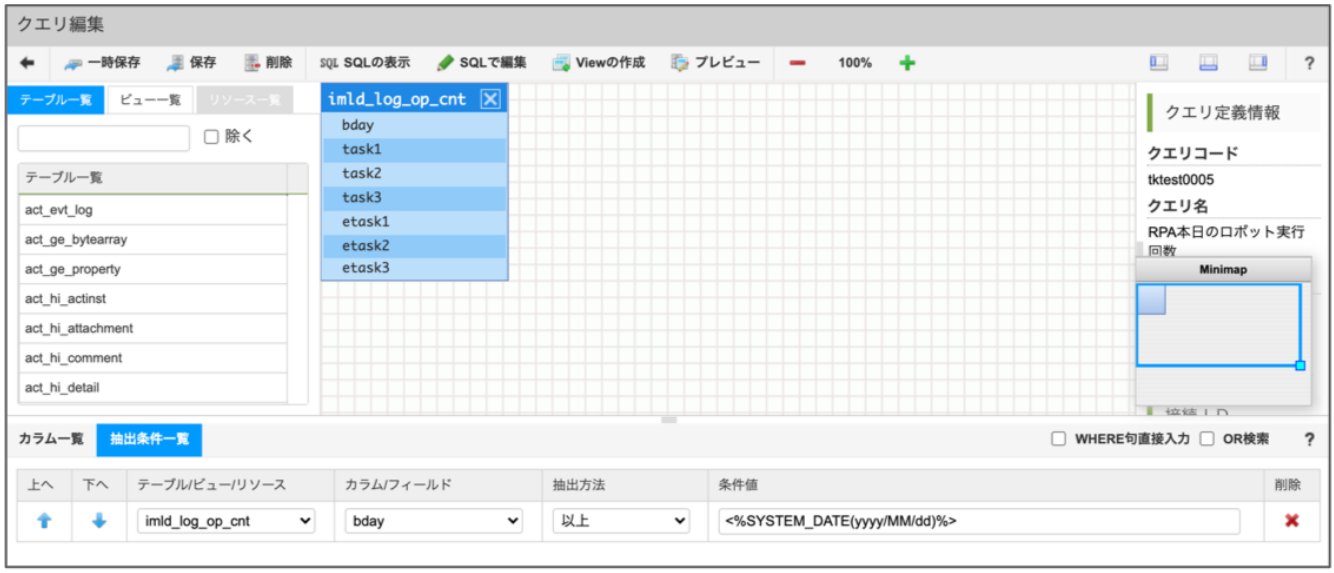
ViewCreatorのクエリを作成します。
以下では、「*③本日のロボット稼働業況*」に用いる「RPA本日のロボット実行回数」クエリを例にとって説明します。

1. 「サイトマップ」→「ViewCreator」→「クエリー覧」 →「新規」をクリックします。
   imld_log_op_cnt を選択します。
   表示された imld_log_op_cnt のカラムを全てダブルクリックして、「カラム一覧」に表示させます。



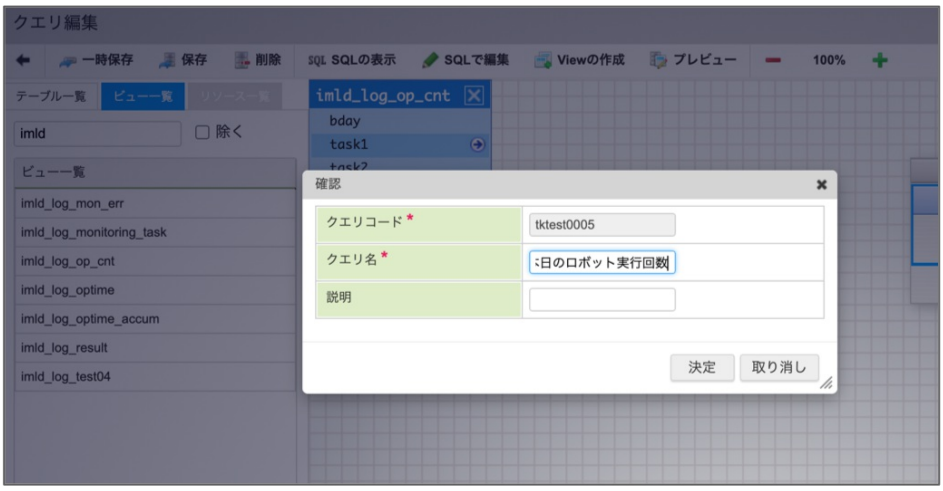
図：「クエリ編集」画面 - カラム一覧

2. 「抽出条件一覧」を選択します。
   カラム「 bday 」をダブルクリックし、抽出方法を「直接入力」、条件値を <%SYSTEM_DATE(yyyy/MM/dd)%> と設定します。



図：「クエリ編集」画面 - 抽出条件一覧

3. 「保存」をクリックします。
   クエリ名に「RPA本日のロボット実行回数」と設定します。
   クエリコードは任意で設定してください。



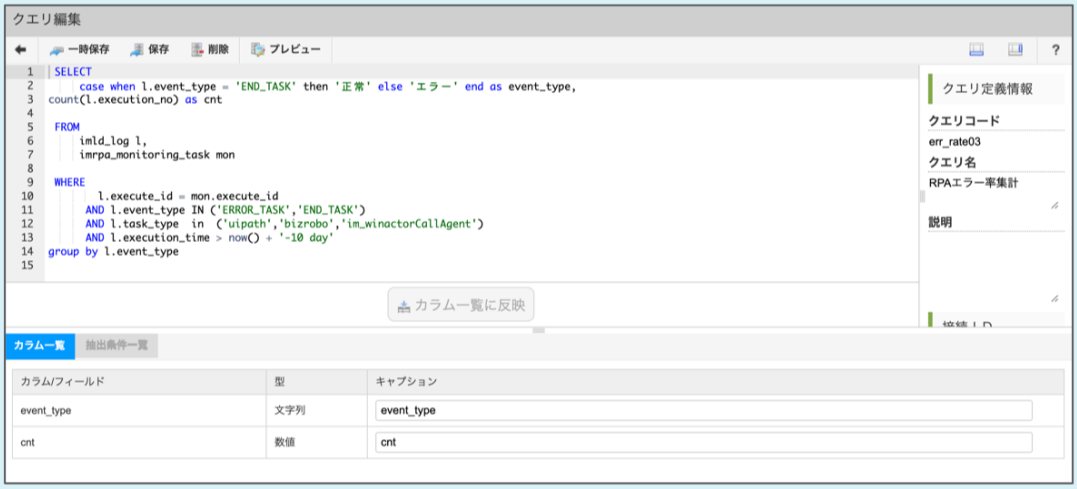
図：「クエリ編集」画面 - 保存

4. 以上の手順を必要なクエリの分だけ、実施します。

クエリ名	参照ビュー/テーブル	カラム一覧	抽出条件一覧	備考
RPAロボットによる削減時間	imld_log_optime_accum	全て Bdayのソートを昇順にします	なし	
RPAモニタリングロボット一覧	mld_log_monitoring_task	全て category_nameのソートを昇順にします	なし	
RPA本日のロボット実行回数	imld_log_op_cnt	全て	<%SYSTEM_DATE(yyyy/MM/dd)%>	
RPAエラー率集計	SQLを記載	event_type、cnt	なし	SQLで編集で直接スクリプトを記載します。（後述）
RPAロボット実行エラー	imld_log_mon_err	全て	なし	
RPAロボット実行回数	imld_log_op_cnt	全て	なし	

コラム

「RPAエラー率集計」のクエリでは、ビューを選択せず、直接SQLを設定してください。
直接SQLを設定するには、「クエリ編集」画面で「SQLで編集」ボタンをクリックしてください。



図：「クエリ編集」画面 - SQLで編集

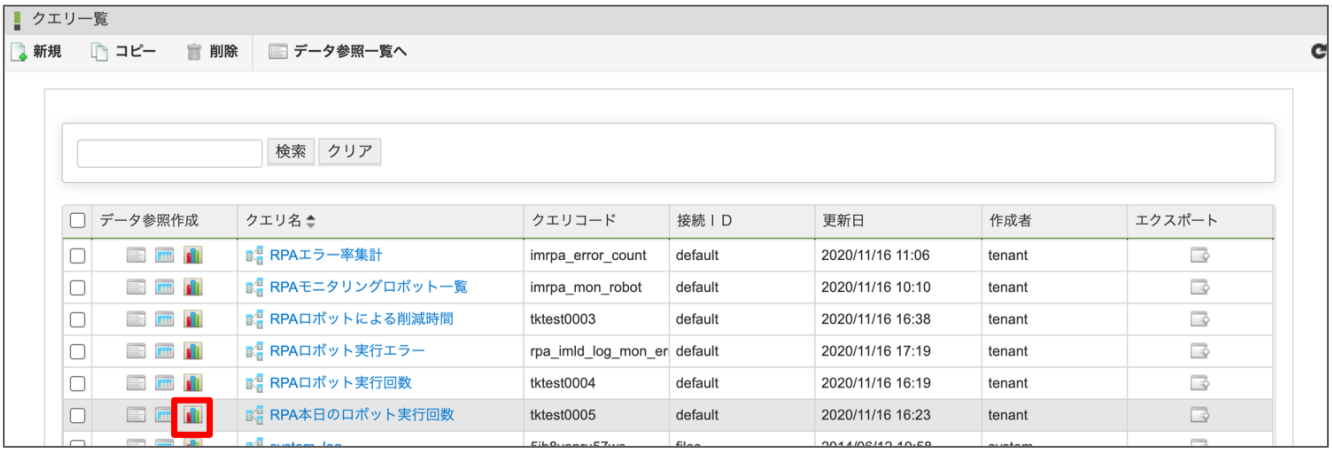
SQLのサンプルコードは以下の通りです。

```
SELECT
    case when l.event_type = 'END_TASK' then '正常' else 'エラー' end as event_type,
count(l.execution_no) as cnt

 FROM
    imld_log l,
    imrpa_monitoring_task mon

 WHERE
     l.execute_id = mon.execute_id
    AND l.event_type IN ('ERROR_TASK','END_TASK')
    AND l.task_type  in  ('uipath','bizrobo','im_winactorCallAgent')
    AND l.execution_time > now() + '-5 day'
group by l.event_type
```

手順６　ViewCreatorのデータ参照を作成する

ViewCreatorのデータ参照を作成します。
以下では、「*③本日のロボット稼働業況*」に用いる「RPA本日のロボット実行回数」データ参照を例にとって説明します。

1. 「サイトマップ」→「ViewCreator」→「クエリー覧」 をクリックします。
   クエリ名「RPA本日のロボット実行回数」の左にある「グラフ」アイコンをクリックします。



図：「クエリー覧」画面

2. 「データ参照・編集 グラフ集計」画面の「カラム設定」カテゴリを設定します。
   `taskX` と `etaskX` を同じグループに設定します。

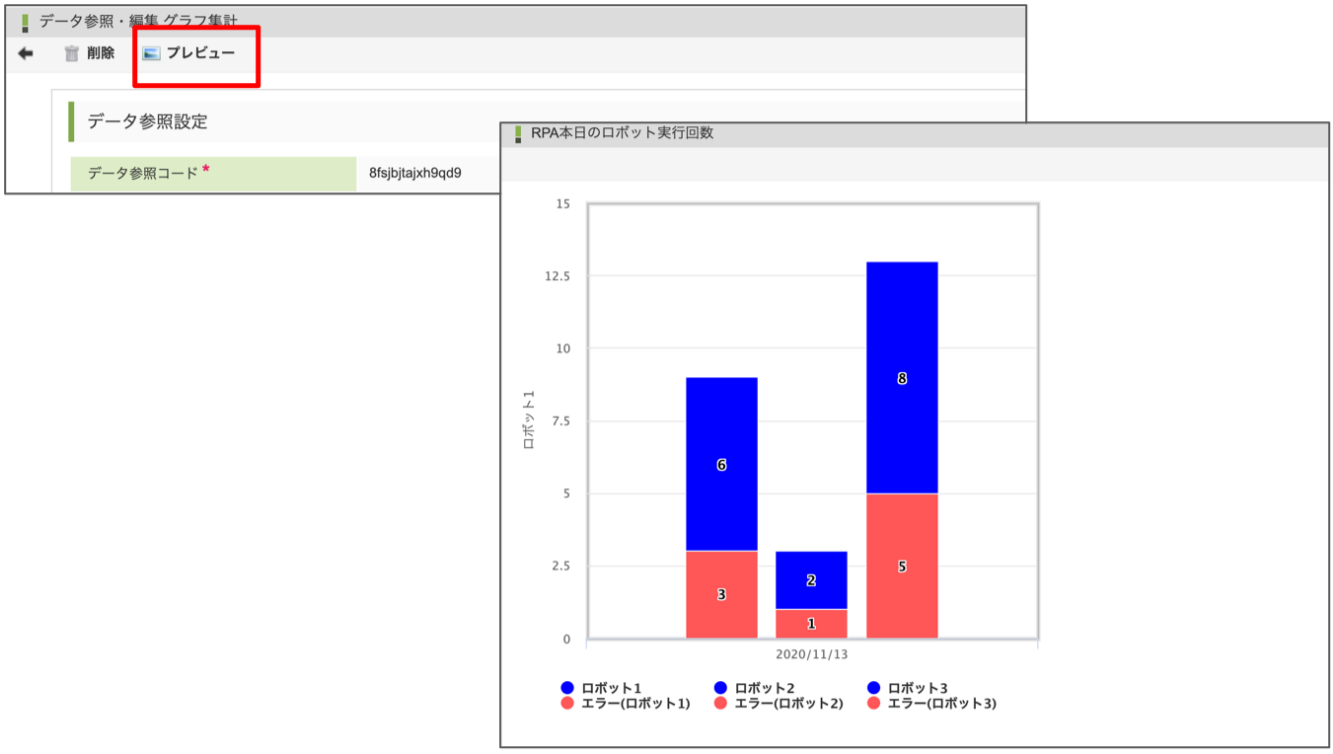
グラフタイプは全て棒グラフとします。
カラーは任意ですが、エラー表示となる `etask` を警告色とすると視認性が高まります。

図：「データ参照・編集 グラフ集計」画面 - カラム設定

3. 「権限設定」カテゴリを設定します。
認証済みユーザを選択します。
ゲストユーザにも参照可能なグラフとしたい場合、ゲストユーザも選択します。

図：「データ参照・編集 グラフ集計」画面 - 権限設定

4. 「プレビュー」をクリックします。
設定したグラフが表示されることを確認してください。



図：「データ参照・編集 グラフ集計」画面 - プレビュー

5. その他「データ参照設定」「グラフ設定」「ページ設定」各カテゴリについては、任意で設定し、「更新して一覧へ戻る」をクリックします。

6. 以上の手順を必要なクエリの数分実施をします。

クエリ名/データ参照名	集計パターン	グラフ描画形式	カラム設定
RPAロボットによる削減時間	グラフ	HighCharts	下記「RPAロボットによる削減時間カラム設定」参照
RPAモニタリングロボット一覧	リスト	-	下記「RPAモニタリングロボット一覧カラム設定」参照
RPA本日のロボット実行回数	グラフ	HighCharts	上記参照
RPAエラー率集計	グラフ	HighCharts	下記「RPAエラー率集計カラム設定」参照
RPAロボット実行エラー	リスト	-	下記「RPAロボット実行エラーカラム設定」参照
RPAロボット実行回数	グラフ	HighCharts	下記「RPAロボット実行回数カラム設定」参照

コラム

「RPAロボットによる削減時間」カラム設定

図：「RPAロボットによる削減時間」カラム設定

カラムの国際化項目の編集

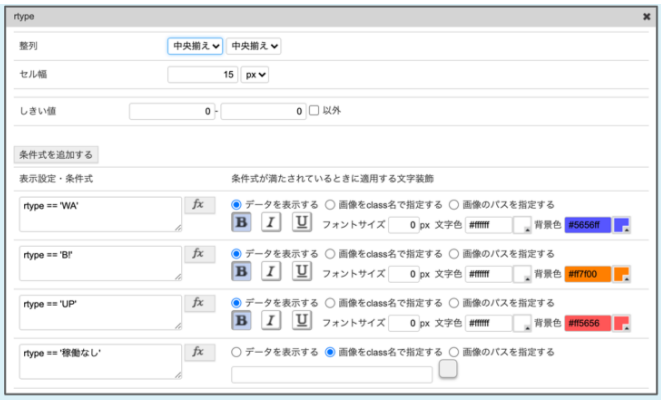
図：「RPAロボットによる削減時間カラム設定」 - カラムの国際化項目の編集

コラム

「RPAモニタリングロボット一覧」カラム設定

図：「RPAモニタリングロボット一覧」カラム設定

rtype カラムの表示設定



図：「RPAモニタリングロボット一覧」カラム設定 - 表示設定 - rtype

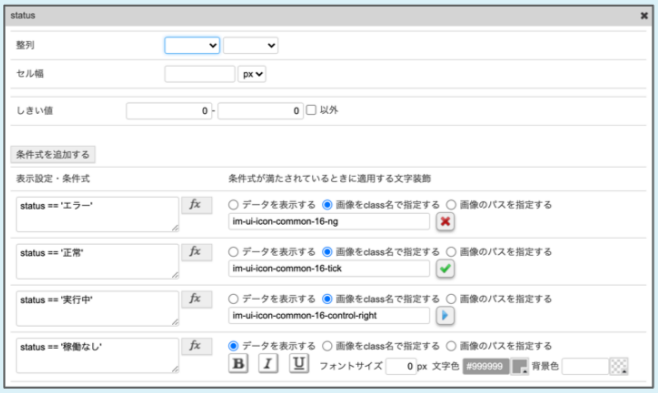
category_name カラムの表示設定

図：「RPAモニタリングロボット一覧」カラム設定 - 表示設定 - category_name

task_name カラムの表示設定

図：「RPAモニタリングロボット一覧」カラム設定 - 表示設定 - task_name

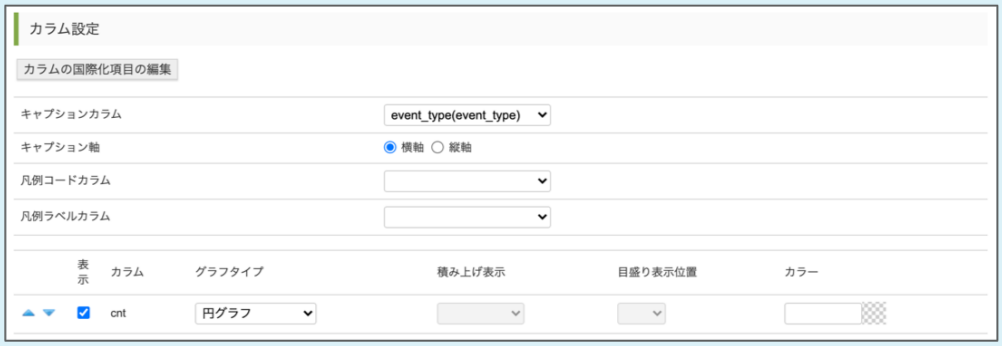
statusカラム の表示設定



図：「RPAモニタリングロボット一覧」カラム設定 - 表示設定 - status

コラム

「RPAエラー率集計」カラム設定



図：「RPAエラー率集計」カラム設定

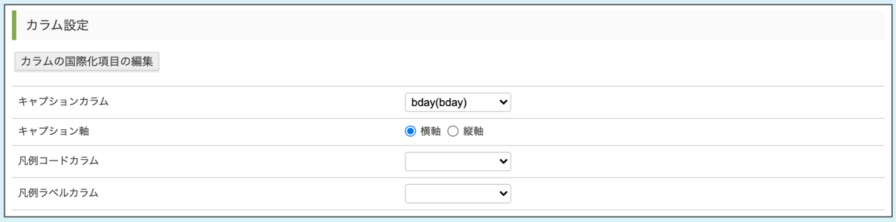
コラム

「RPAロボット実行エラー」カラム設定

図：「RPAロボット実行エラー」カラム設定

コラム

「RPAロボット実行回数」カラム設定

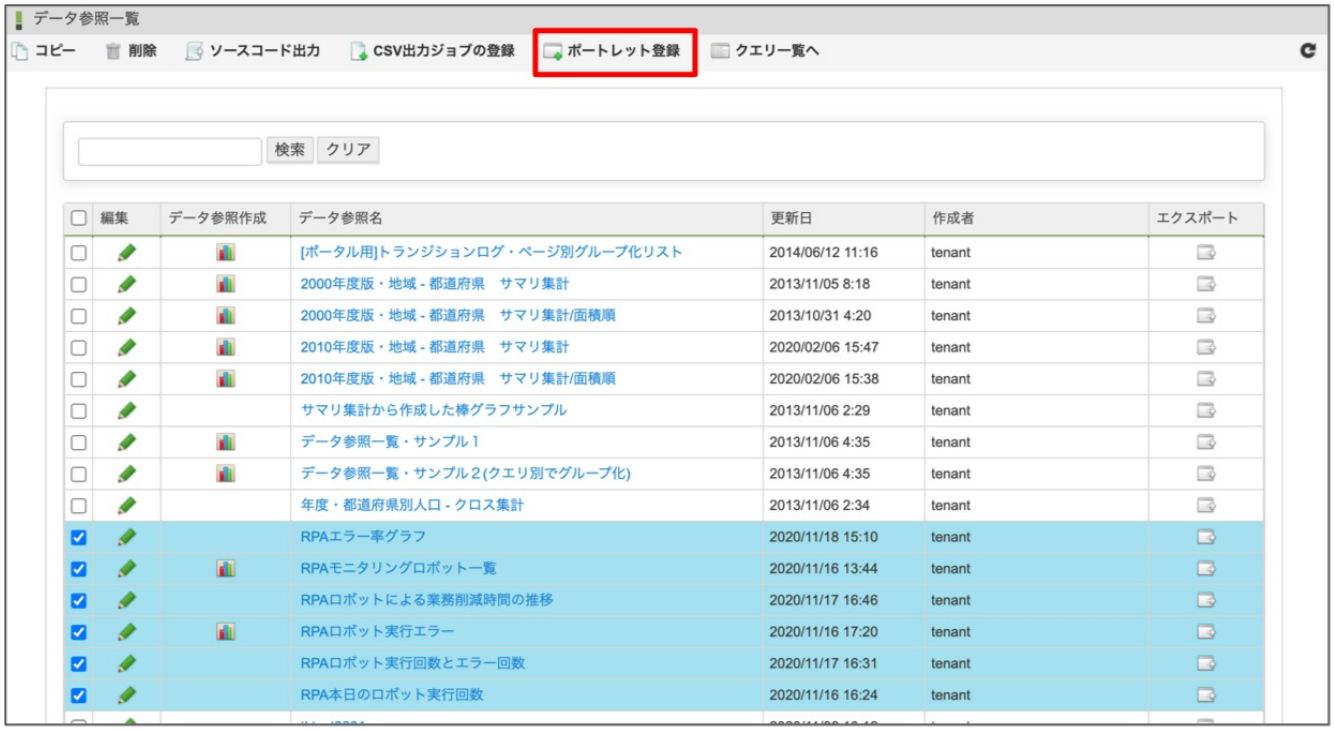


図：「RPAロボット実行回数」カラム設定

手順７　グラフおよび一覧をポートレット登録する

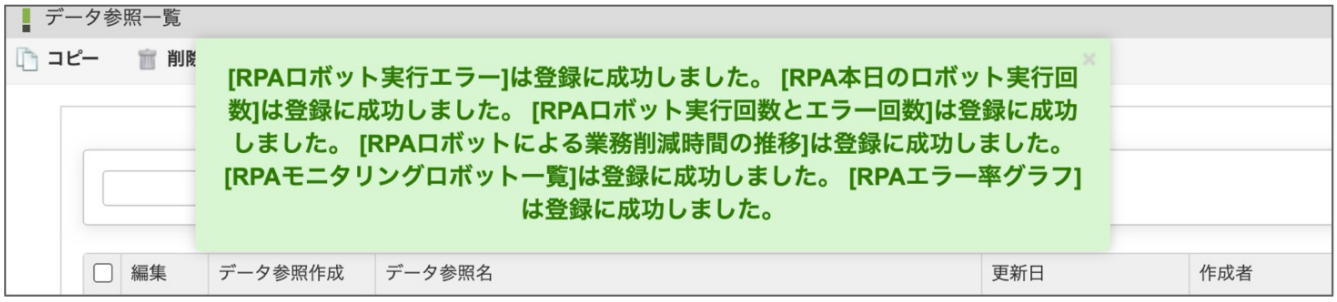
作成したグラフをポートレットに登録します。

1. 「サイトマップ」→「ViewCreator」→「データ参照一覧」 をクリックします。
   作成したグラフにチェックをつけ、「ポートレット登録」をクリックします。



図：「データ参照一覧」画面 - ポートレット登録

2. 「登録に成功しました。」というメッセージを確認します。

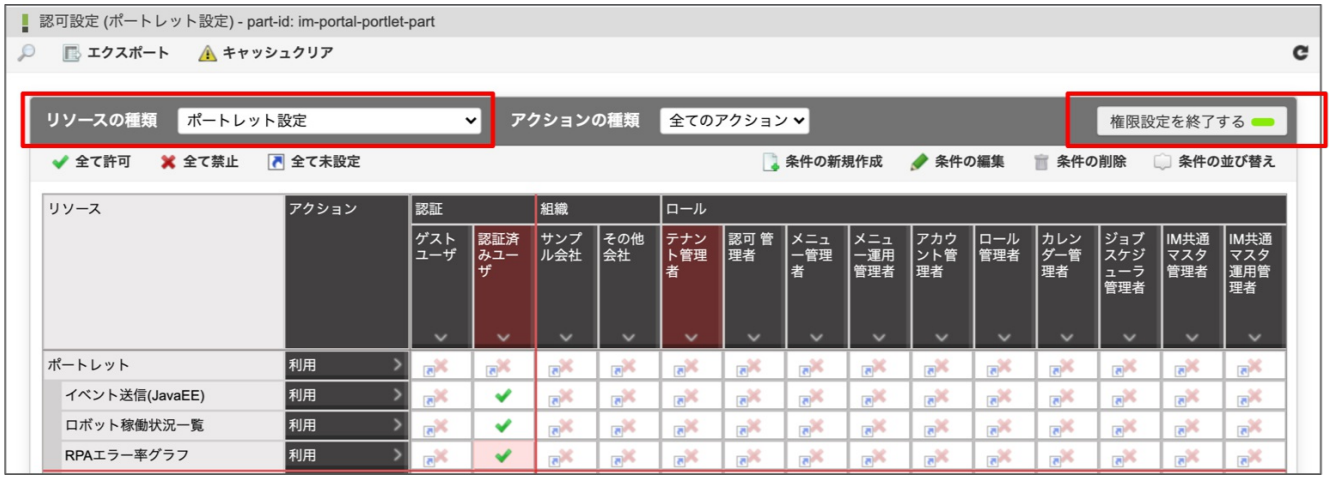


図：ポートレット登録成功メッセージ

手順８　ポートレットの権限設定

登録したポートレットの権限設定をします。

1. 「サイトマップ」→「 テナント管理」→「認可」をクリックします。
「リソースの種類」で「ポートレット設定」を選択します。
「権限設定を開始する」ボタンをクリックし、インジケーターがオンの状態にします。
先ほど登録したポートレットに対して、「認証済みユーザ」の権限にチェックをつけ、設定を完了します。



図：「認可設定」画面

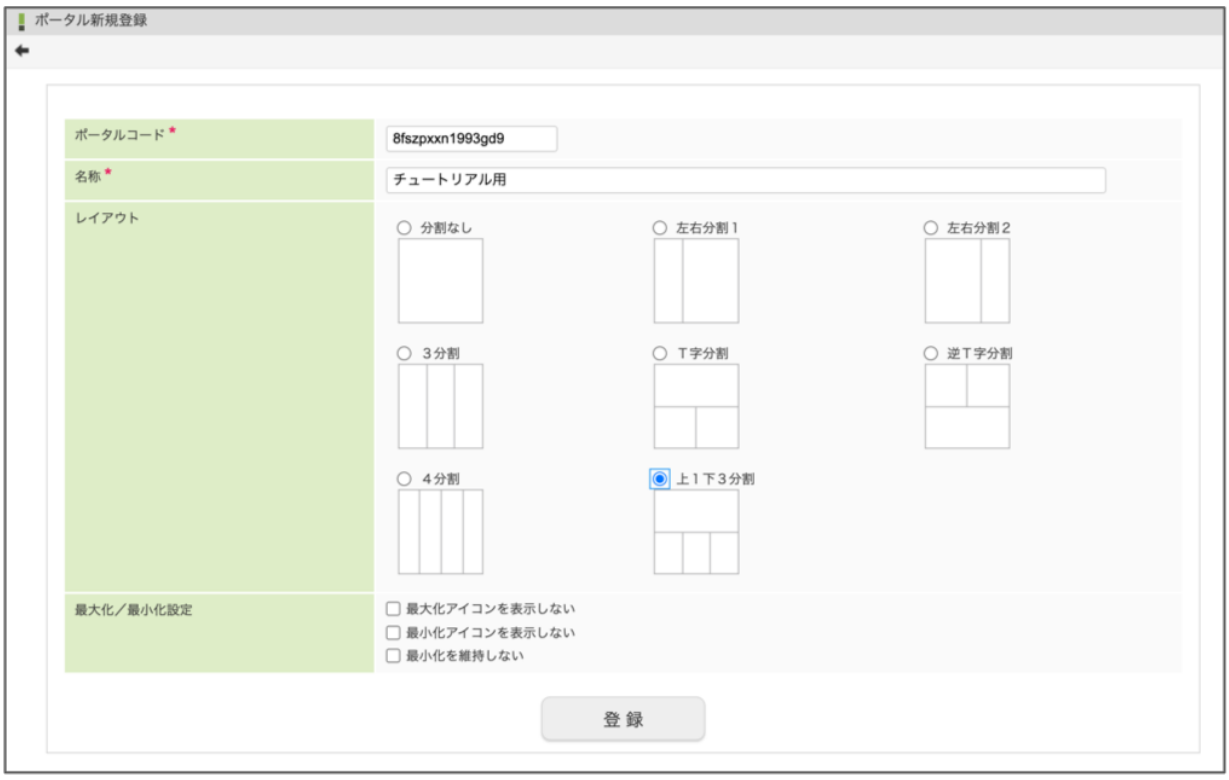
手順９　ポートレット配置

登録したポートレットをポータル上に配置して、ダッシュボードにします。

1. ポータルを表示します。
   ポータルタブの左にある「＋」ボタンをクリックします。

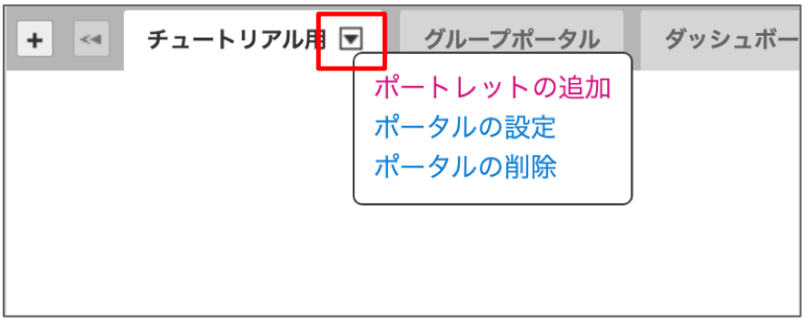
図：「ポータル」画面

2. 「ポータルコード」、「名称」に任意のIDと名前を設定します。
   「レイアウト」を任意に設定します。
   「登録」ボタンをクリックします。



図：「ポータル新規登録」画面

3. 登録されたポータルタブの右にある「▼」をクリックします。
   「ポートレットの追加」をクリックします。



図：ポートレットのタブ

4. 「アプリケーション」で「ViewCreator」を選択します。
   配置したいポートレットを選択します。
   「追加」ボタンをクリックします。

図：ポートレットの追加

5. ポートレットが配置されたことを確認します。
   配置したポートレットは、タイトル部分をドラッグ&ドロップすることで、任意の場所に配置可能です。
   画面の解像度などを考慮し、見やすいレイアウトに再配置してください。

図：ポートレットの配置

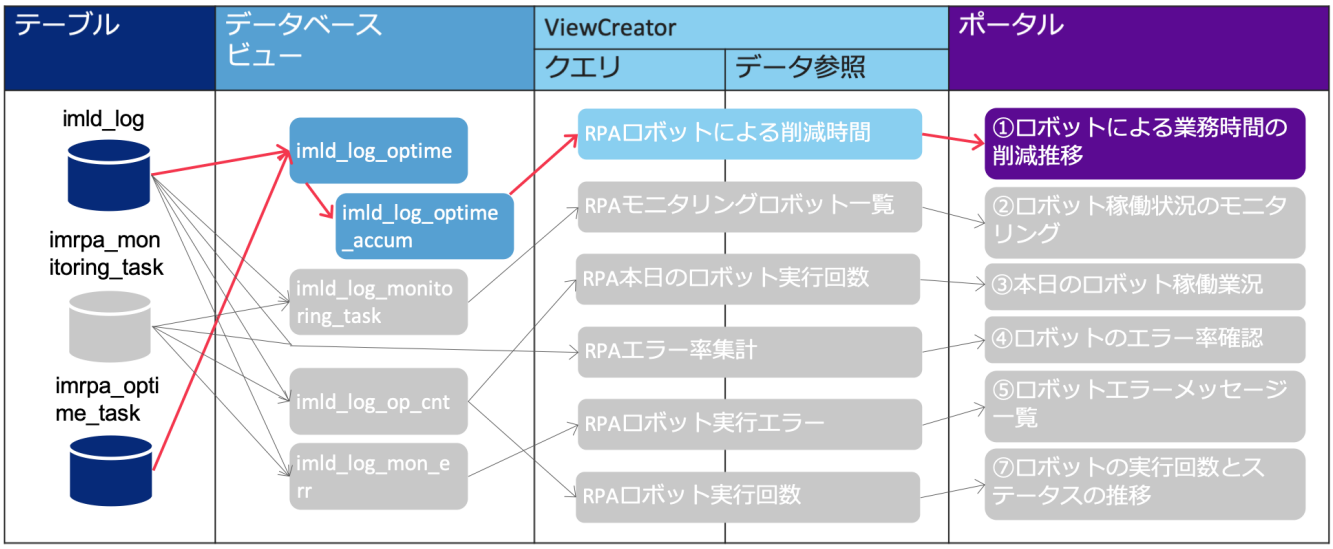
コラム

紹介した手順はユーザごとのポートレットを設定する「ユーザポータル」の設定例です。
intra-mart Accel Platformでは、グループで共通したポートレットを表示する「グループポータル」も利用可能です。

利用方法は下記を参照してください。
「ポータル 管理者操作ガイド」 - 「グループポータルを管理する」

ダッシュボード作成の一連手順について

ダッシュボード作成について、一連の流れを動画としてまとめました。
紹介する内容は、「*①ロボットによる業務時間の削減推移*」を作成するための一連の手順です。



図：「*①ロボットによる業務時間の削減推移*」に必要な設定一覧

動画では、「*①ロボットによる業務時間の削減推移*」ポートレットに必要な追加テーブル、テーブルビュー、ViewCreator設定のみを実施しています。
他のポートレットを追加する際も、本動画と本チュートリアルを参考にしてください。

テーブルビュー作成スクリプト サンプルコード

項目

imld_log_optime ビュー

```
create or replace view imld_log_optime as
   select
      bday,
      COALESCE(
         round(sum(task1) / 60),
         0
      ) as task1,
      COALESCE(
         round(sum(task2) / 60),
         0
      ) as task2,
      COALESCE(
         round(sum(task3) / 60),
         0
      ) as task3,
      count(*),
      round(
         COALESCE(round(sum(task1) / 60), 0) + COALESCE(round(sum(task2) / 60), 0) + COALESCE(round(sum(task3) / 60), 0)
      ) as total
   from
      (
         select
            TO_CHAR(i.execution_time, 'YYYY/MM/DD') as bday,
            (
               select
                  (
                     case
                        when t.optime - (i2.duration / 1000) < 0 then 0
                        else t.optime - (i2.duration / 1000)
                     end
                  )
               from
                  imld_log i2
               where
                  i.execution_id = i2.execution_id
               and i.execution_no = i2.execution_no
               and i2.execute_id = 'bizrobo0021'
            ) as task1,
            (
               select
                  (
                     case
                        when t.optime - (i2.duration / 1000) < 0 then 0
                        else t.optime - (i2.duration / 1000)
                     end
                  )
               from
                  imld_log i2
               where
                  i.execution_id = i2.execution_id
               and i.execution_no = i2.execution_no
               and i2.execute_id = 'im_winactorCallAgent1'
            ) as task2,
            (
               select
                  (
                     case
                        when t.optime - (i2.duration / 1000) < 0 then 0
                        else t.optime - (i2.duration / 1000)
                     end
                  )
               from
                  imld_log i2
               where
                  i.execution_id = i2.execution_id
               and i.execution_no = i2.execution_no
               and i2.execute_id = 'uipath0012'
            ) as task3
         from
            imld_log i,
```

```
            imrpa_optime_task t
        where
            i.execute_id = t.execute_id
        and i.execution_time >= current_date - 30
    ) as A
group by
    bday
order by
    bday
```

imld_log_optime_accum ビュー

```
create or replace view imld_log_optime_accum as
    select
        i.bday,
        i.task1,
        i.task2,
        i.task3,
        i.total,
        sum(
            r.total
        ) as accum
    from
        imld_log_optime as i,
        imld_log_optime as r
    where
        i.bday >= r.bday
    group by
        i.bday,
        i.task1,
        i.task2,
        i.task3,
        i.total
    order by
        i.bday
```

imld_log_monitoring_task ビュー

```
create or replace view imld_log_monitoring_task as
 select
  COALESCE(
     category_name,
     'マスタ登録なし'
  ) as category_name,
  COALESCE(
     task_name,
     'マスタ登録なし'
  ) as task_name,
  COALESCE(
     status,
     '稼働なし'
  ) as status,
  COALESCE(
     rtype,
     '稼働なし'
  ) as rtype
 from
  (
     select
        m.flow_id,
        m.execute_id,
        m.event_type,
        m.duration,
        m.error_message,
        CASE
           WHEN m.event_type = 'END_TASK'
        and m.duration > 100000 THEN '遅い'
           WHEN m.event_type = 'END_TASK' THEN '正常'
           WHEN m.event_type = 'BEGIN_TASK' THEN '実行中'
           WHEN m.event_type = 'ERROR_TASK' THEN 'エラー'
           ELSE '稼働なし'
        END as status,
        CASE
           WHEN m.task_type = 'im_winactorCallAgent' THEN 'WA'
           WHEN m.task_type = 'bizrobo' THEN 'B!'
           WHEN m.task_type = 'uipath' THEN 'UP'
           ELSE '稼働なし'
        END as rtype
     from
        (
           select
              i.flow_id,
              i.execute_id,
              max(i.execution_time) as execution_time
           from
              imld_log i,
              imrpa_monitoring_task mon
           where
              i.execute_id = mon.execute_id
           and i.execution_time >= current_date - 5
           group by
              i.flow_id,
              i.execute_id
        ) as s,
        imld_log as m
     where
        s.flow_id = m.flow_id
     and s.execute_id = m.execute_id
     and s.execution_time = m.execution_time
  ) z
  FULL OUTER JOIN
     imrpa_monitoring_task mon
  on  z.execute_id = mon.execute_id
```

imld_log_op_cnt ビュー

```
create or replace view imld_log_op_cnt as
   select
```

```
    bday,
    count(
        task1
    ) as task1,
    count(
        task2
    ) as task2,
    count(
        task3
    ) as task3,
    count(
        etask1
    ) as etask1,
    count(
        etask2
    ) as etask2,
    count(
        etask3
    ) as etask3
from
    (
        select
            TO_CHAR(i.execution_time, 'YYYY/MM/DD') as bday,
            (
                select
                    i2.execution_id
                from
                    imld_log i2
                where
                    i.execution_id = i2.execution_id
                and i.execution_no = i2.execution_no
                and i2.execute_id = 'bizrobo0021'
                and i2.event_type = 'END_TASK'
            ) as task1,
            (
                select
                    i2.execution_id
                from
                    imld_log i2
                where
                    i.execution_id = i2.execution_id
                and i.execution_no = i2.execution_no
                and i2.execute_id = 'im_winactorCallAgent1'
                and i2.event_type = 'END_TASK'
            ) as task2,
            (
                select
                    i2.execution_id
                from
                    imld_log i2
                where
                    i.execution_id = i2.execution_id
                and i.execution_no = i2.execution_no
                and i2.execute_id = 'uipath0012'
                and i2.event_type = 'END_TASK'
            ) as task3,
            (
                select
                    i2.execution_id
                from
                    imld_log i2
                where
                    i.execution_id = i2.execution_id
                and i.execution_no = i2.execution_no
                and i2.execute_id = 'bizrobo0021'
                and i2.event_type = 'ERROR_TASK'
            ) as etask1,
            (
                select
                    i2.execution_id
                from
                    imld_log i2
                where
```

```
                i.execution_id = i2.execution_id
            and i.execution_no = i2.execution_no
            and i2.execute_id = 'im_winactorCallAgent1'
            and i2.event_type = 'ERROR_TASK'
        ) as etask2,
        (
            select
                i2.execution_id
            from
                imld_log i2
            where
                i.execution_id = i2.execution_id
            and i.execution_no = i2.execution_no
            and i2.execute_id = 'uipath0012'
            and i2.event_type = 'ERROR_TASK'
        ) as etask3
    from
        imld_log i,
        imrpa_monitoring_task as mon
    where
        i.execute_id = mon.execute_id
    and i.execution_time >= current_date - 30
) as A
group by
    bday
order by
    bday
```

imld_log_mon_err ビュー

```
create or replace view imld_log_mon_err as
    select
        TO_CHAR(
            i.execution_time,
            'YYYY/MM/DD HH:mm:ss'
        ) as ltime,
        mon.category_name,
        mon.task_name,
        i.error_message
    from
        imld_log as i,
        imrpa_monitoring_task as mon
    where
        i.execute_id = mon.execute_id
    and i.execution_time >= current_date - 30
    and i.error_message <> ''
```

著作権および特記事項