目次

- 1. 改訂情報
- 2. はじめに
  - 2.1. 本書の目的
  - 2.2. 対象読者
  - 2.3. 本書の構成
- 3. 概要
  - 3.1. IM-BPMとは
  - 3.2. 本チュートリアルガイドの説明範囲
  - 3.3. 事前準備
    - 3.3.1. 環境セットアップ
    - 3.3.2. チュートリアル実行ユーザ
- 4. プロセスの作成
  - 4.1. 基礎編
    - 4.1.1. チュートリアルの概要（作成物のイメージ）
    - 4.1.2. プロセス定義を新規に作成する
    - 4.1.3. プロセス定義を IM-BPM Runtimeへデプロイする
    - 4.1.4. プロセスを開始する
  - 4.2. 実用編
    - 4.2.1. データプロパティ
      - 4.2.1.1. データプロパティを定義する
    - 4.2.2. マルチインスタンス
      - 4.2.2.1. マルチインスタンス を使用する
    - 4.2.3. リスナ
      - 4.2.3.1. IM-LogicDesignerのリスナを利用する
      - 4.2.3.2. タスクリスナを利用してタスクの処理依頼メールを送信する
      - 4.2.3.3. タスクリスナを利用して特定の組織に所属するユーザを処理対象ユーザに設定する
    - 4.2.4. 関連ドキュメント
      - 4.2.4.1. IM-Wikiを関連ドキュメントとして設定する
    - 4.2.5. プール、レーン
      - 4.2.5.1. プールとレーンを利用して作業の関係者を明確にする
    - 4.2.6. コールアクティビティ
      - 4.2.6.1. コールアクティビティを使用する
      - 4.2.6.2. コールアクティビティで呼び出すプロセス定義に業務キーを設定する
    - 4.2.7. パラレルゲートウェイ
      - 4.2.7.1. パラレルゲートウェイを使用する
    - 4.2.8. 排他ゲートウェイ
      - 4.2.8.1. 排他ゲートウェイを使用する
    - 4.2.9. 包括ゲートウェイ
      - 4.2.9.1. 包括ゲートウェイを使用する
    - 4.2.10. イベント
      - 4.2.10.1. シグナルイベントを使用する
      - 4.2.10.2. メッセージイベントを使用する
      - 4.2.10.3. タイマイベントを使用する
    - 4.2.11. IM-LogicDesignerタスク
      - 4.2.11.1. IM-LogicDesignerタスクを利用してユーザ情報を取得する
      - 4.2.11.2. IM-LogicDesignerタスクで実行するロジックフローを動的に設定する
      - 4.2.11.3. IM-LogicDesignerタスクを利用してOpenRulesを使用する
    - 4.2.12. 申請タスク
      - 4.2.12.1. 申請タスクを使用してワークフローと連携する
    - 4.2.13. 起票タスク
      - 4.2.13.1. 起票タスクを使用してワークフローと連携する
      - 4.2.13.2. 起票タスクを使用してワークフローの申請ノード処理対象者を動的に設定する
    - 4.2.14. IM-FormaDesigner for Accel Platform をユーザ入力フォームとして利用する
      - 4.2.14.1. 開始イベントのフォームにFormaアプリケーションを利用する
      - 4.2.14.2. ユーザタスクのフォームにFormaアプリケーションを利用する
  - 4.3. 発展編
    - 4.3.1. ワークフローとの連携
      - 4.3.1.1. 承認ノードの処理対象者をロジックフローで指定する

- 5. プロセスの管理
  - 5.1. 本番環境への適用
    - 5.1.1. プロセス定義のデプロイをスケジューリングする
      - 5.1.1.1. プロセスデザイナのプロジェクトをエクスポートする
      - 5.1.1.2. プロセスデザイナのプロジェクトをパブリックストレージに配置する
      - 5.1.1.3. プロセスデザイナのプロジェクトのインポートとプロセス定義のデプロイを行うロジックフロー定義を作成する
      - 5.1.1.4. ロジックフローを実行するジョブネットを作成する
      - 5.1.1.5. 実行結果を確認する
  - 5.2. アドオン画面の作成
    - 5.2.1. アドオンのプロセス一覧画面の作成
      - 5.2.1.1. プロセスインスタンスのデータを取得するIM-LogicDesignerのロジックフローを作成する
      - 5.2.1.2. IM-LogicDesignerのロジックフローと連携したViewCreatorのクエリ・データ参照を作成する
      - 5.2.1.3. 作成したプロセスインスタンス一覧画面を確認する

改訂情報

変更年月日	変更内容
2017-12-01	初版
2017-12-22	第2版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*コールアクティビティを使用する*」 　■ 「*パラレルゲートウェイを使用する*」 　■ 「*排他ゲートウェイを使用する*」
2018-04-01	第3版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*起票タスクを使用してワークフローの申請ノード処理対象者を動的に設定する*」 　■ 「*IM-LogicDesignerのリスナを利用する*」 　■ 「*コールアクティビティで呼び出すプロセス定義に業務キーを設定する*」
2018-04-27	第4版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*シグナルイベントを使用する*」
2018-05-31	第5版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*メッセージイベントを使用する*」
2018-06-29	第6版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*タスクリスナを利用してタスクの処理依頼メールを送信する*」
2018-08-01	第7版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*タイマイベントを使用する*」 　■ 「*タスクリスナを利用して特定の組織に所属するユーザを処理対象ユーザに設定する*」 ■ 「*プロセスの作成*」 - 「*発展編*」を追加しました。 ■ 「*プロセスの作成*」 - 「*発展編*」に下記のページを追加しました。 　■ 「*承認ノードの処理対象者をロジックフローで指定する*」 ■ 「*プロセスの管理*」を追加しました。 ■ 「*プロセスの管理*」 - 「*本番環境への適用*」を追加しました。 ■ 「*プロセスの管理*」 - 「*本番環境への適用*」に下記のページを追加しました。 　■ 「*プロセス定義のデプロイをスケジューリングする*」 ■ 「*プロセスの管理*」 - 「*アドオン画面の作成*」を追加しました。 ■ 「*プロセスの管理*」 - 「*アドオン画面の作成*」に下記のページを追加しました。 　■ 「*アドオンのプロセス一覧画面の作成*」
2018-09-28	第8版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*IM-Wikiを関連ドキュメントとして設定する*」
2018-12-01	第9版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*IM-LogicDesignerタスクを利用してOpenRulesを使用する*」

変更年月日	変更内容
2018-12-27	第10版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 ■■ 「*包括ゲートウェイを使用する*」 ■ 各エレメントの詳細を追記しました。 ■ 「IM-BPM プロセスデザイナ 操作ガイド」に説明を対応しました。

はじめに

- 本書の目的
- 対象読者
- 本書の構成

本書の目的

本書は、IM-BPM for Accel Platform（以下、IM-BPM）を利用してプロセス定義の作成を行う開発者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

本書では次の開発者を対象としています。

- IM-BPMによる開発の一連の流れを知りたい
- IM-BPMを利用してビジネスロジックを開発したい

なお、本書を読み進めるにあたり、次の内容を理解していることが必須条件です。

- intra-mart Accel Platform の概要、および操作方法
- Business Process Management（BPM）の基本概念

> **コラム**
>
> BPM について知りたい方は、以下のページを参照してください。
> BPMとは（一般社団法人　日本ビジネスプロセス・マネジメント協会）

また、次のドキュメントを読了していると、より理解が深まります。

- IM-BPM 仕様書
- IM-BPM ユーザ操作ガイド
- IM-BPM プロセスデザイナ 操作ガイド

IM-BPM は intra-mart Accel Platform 上の他のアプリケーションと連携します。
必要に応じて、以下のドキュメントも併せて参照してください。

- IM-LogicDesigner仕様書
- IM-LogicDesigner ユーザ操作ガイド
- IM-Workflow 仕様書
- IM-Workflow 管理者操作ガイド
- IM-Workflow ユーザ操作ガイド
- IM-FormaDesigner 仕様書
- IM-FormaDesigner 作成者操作ガイド
- IM-BIS 仕様書
- IM-BIS 業務管理者操作ガイド
- IM-BIS ユーザ 操作ガイド

本書の構成

- *概要*

  本書、および、IM-BPMの概要について説明します。

- *プロセスの作成*

  プロセスを作成する機能について説明します。

  - *基礎編*

    基礎編として、シンプルなプロセスの作成を行うチュートリアルです。
    このチュートリアルを通していただくことで、プロセスを作成する上で必要な作業の流れや、機能についてご理解いただけます。

  - *実用編*

    実用編として、基礎編の作業を元に、実用的なプロセスを作成する方法を説明します。

- *プロセスの管理*

  プロセスを管理する方法について説明します。

概要

- IM-BPMとは
- 本チュートリアルガイドの説明範囲
- 事前準備
  - 環境セットアップ
  - チュートリアル実行ユーザ

IM-BPMとは

IM-BPMについて、ここでは「IM-BPM 仕様書」から一部引用して説明します。

> IM-BPM 仕様書 - 3.1 IM-BPM for Accel Platformとは
>
> IM-BPMとは、intra-mart Accel Platform上で業務プロセスを実行することが可能なアプリケーションです。
> IM-BPMの特徴は以下の通りです。
>
> - BPMN2.0形式に対応した業務プロセスを実行できます。
> - IM-FormaDesignerと連携し、システムに必要となる画面と簡単に連携できます。
> - IM-LogicDesignerと連携し、システム間連携等の業務ロジックを簡単に呼び出すことができます。
> - IM-BPMでは、IM-Workflow/IM-BISと連携し、複数のワークフローを含めた業務プロセスを管理できます。
> - 作成した業務プロセスは、その業務プロセスの実行方法をモニタリングできます。
>   これにより業務プロセスの見直しを図り改善につなげることが可能です。
> - 作成した業務プロセスはREST APIにより外部システムから呼び出すことができます。

本チュートリアルガイドは上記概要をベースとして、簡単なプロセス定義の作成方法から、一歩先の利用方法まで、実際に開発者の皆様へ紹介します。

本チュートリアルガイドの説明範囲

本チュートリアルガイドでは、IM-BPM プロセスデザイナを用いて、IM-BPM Runtime上で動作するプロセス資材を作成し、IM-BPM Runtimeへプロセス資材をデプロイし、プロセスがどのように動作するか説明します。
また、作成したプロセスを管理する方法についても説明します。

事前準備

本チュートリアルを進めるにあたり、以下の事前準備が行われていることが前提です。

環境セットアップ

本チュートリアルを進める上で必要なintra-mart Accel Platformの環境セットアップ情報は以下の通りです。

1. 以下のモジュールを含んだ形で、intra-mart Accel Platformのwarファイルを作成していること
   - IM-BPM モジュール
   - IM-FormaDesigner モジュール （プロセスの作成の基礎編には必要ありません。）
   - IM-BIS モジュール （プロセスの作成の基礎編には必要ありません。）
2. intra-mart Accel Platformのテナント環境セットアップが完了していること
3. サンプルデータのインポートが完了していること

> **コラム**
>
> セットアップについては「intra-mart Accel Platform セットアップガイド」を参照してください。

チュートリアル実行ユーザ

本チュートリアルで行う操作は全て、「IM-BPM 管理者（im_bpm_manager）」ロールを持つユーザであることを前提としています。
必要に応じて、ロールの付与を実施してください。
また、本チュートリアルで作成するプロセスを実行する場合は、「IM-BPM 管理者（im_bpm_manager）」ロールを付与したユーザ「青柳辰巳（aoyagi）」でログインしてください。

コラム

ユーザの設定については、「IM-共通マスタ 管理者操作ガイド」-「ユーザ」を参照してください。

プロセスの作成

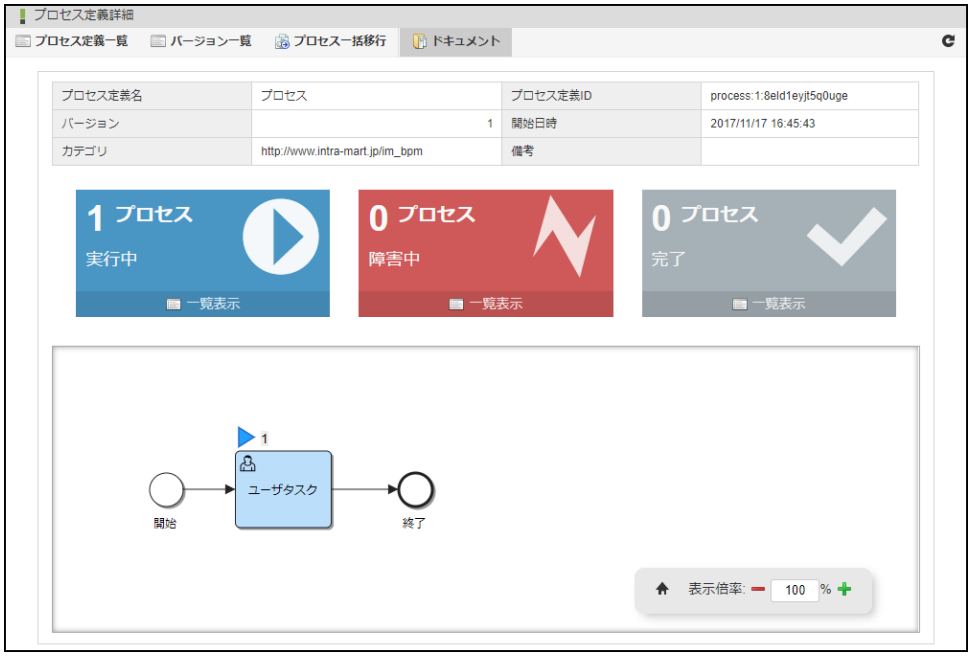
基礎編

基礎編のチュートリアルは「*事前準備*」が完了していることを前提としています。

- チュートリアルの概要（作成物のイメージ）
- プロセス定義を新規に作成する
- プロセス定義を IM-BPM Runtimeへデプロイする
- プロセスを開始する

チュートリアルの概要（作成物のイメージ）

基礎編のチュートリアルでは、IM-BPM プロセスデザイナを用いて、IM-BPM Runtime上で動作する簡単なプロセス定義を作成し、IM-BPM Runtimeへプロセス資材をデプロイし、プロセスを開始するまでを説明します。

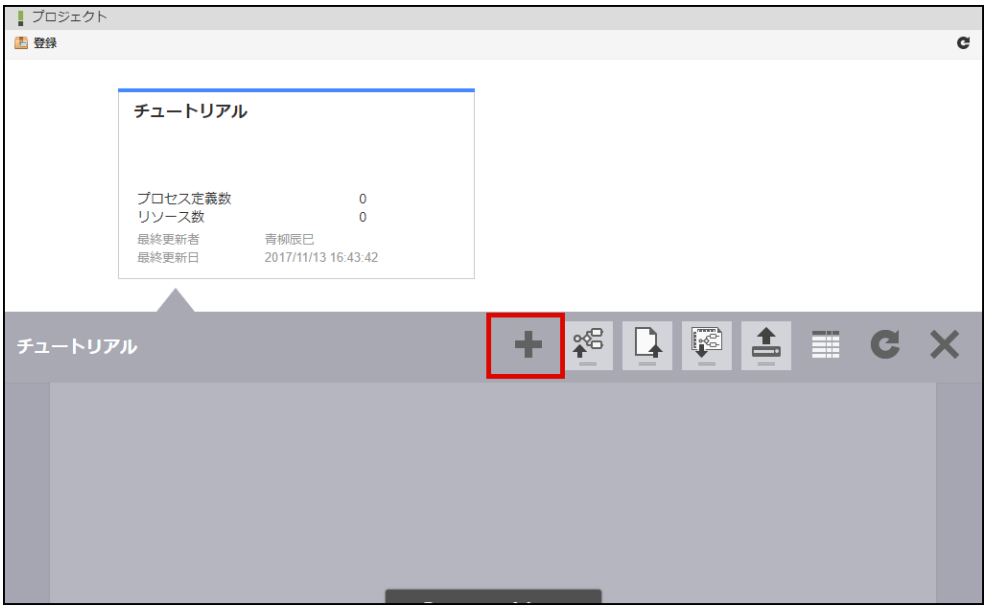


図：プロセス定義

プロセス定義を新規に作成する

プロセス定義を作成します。

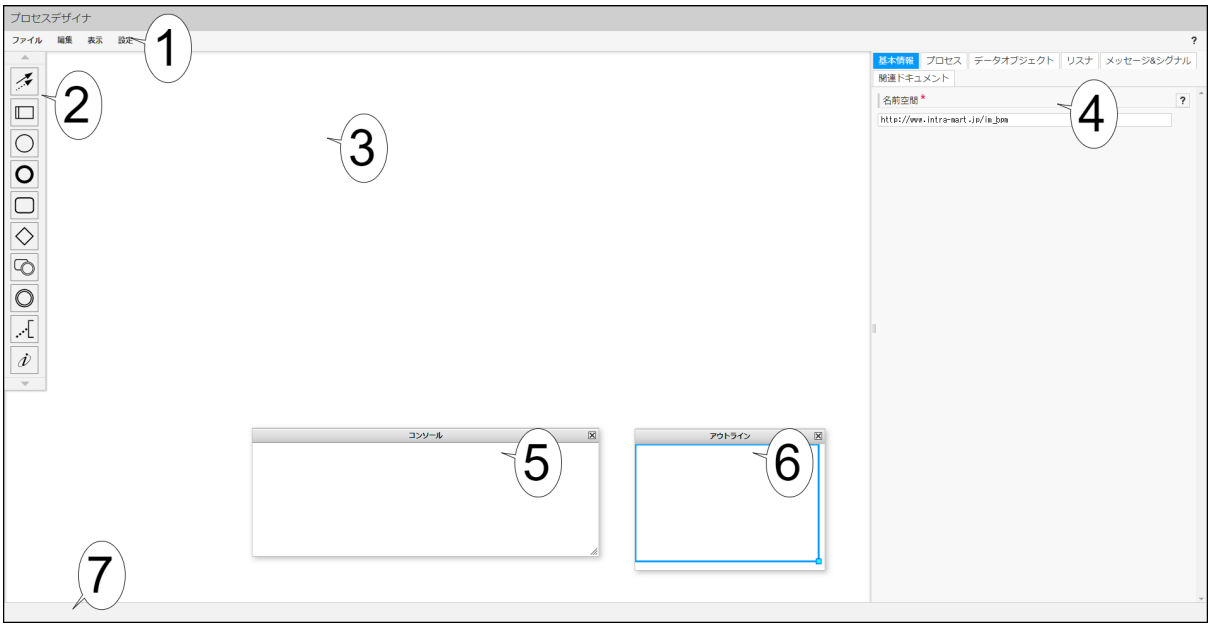
1. 「サイトマップ」→「BPM」→「プロセスデザイナ」をクリックします。
2. 新規プロジェクトを作成し、  をクリックします。



図：プロジェクト詳細エリア - 新規登録

> **コラム**
>
> プロジェクトの作成や認可設定については「IM-BPM プロセスデザイナ 操作ガイド」 - 「プロジェクトの管理」を参照してください。

3. プロセスエディタが開きます。



図：プロセスエディタ

1. ヘッダメニュー

   プロセスエディタの操作や設定を行うメニューです。

2. パレット

   キャンバスに配置するエレメントやコネクタの一覧です。
   この一覧からエレメントをドラッグ＆ドロップ操作によりキャンバスに配置します。

3. キャンバス

   エレメントを配置してプロセスを作成する領域です。

4. プロパティエリア

   エレメントのプロパティを設定する領域です。

5. コンソール

   プロセス定義のチェック時にエラーや警告が表示されます。

6. アウトライン

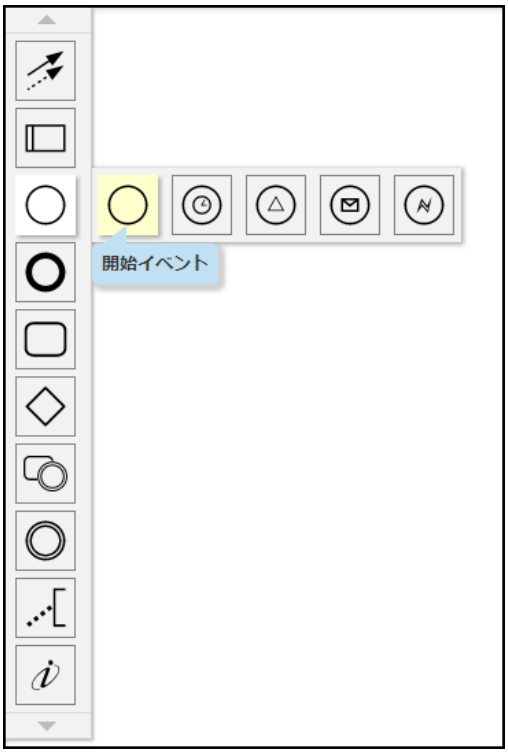
プロセス全体を俯瞰表示する領域です。
表示領域を移動したり、表示の拡大縮小を行ったりすることができます。

7. フッタ

   プロセスエディタの動作に関するメッセージを表示します。

4. 「パレット」にて ◯ にカーソルを合わせます。

5. 「パレット」の右側に現れる一覧から、 ◯ をドラッグし、「キャンバス」上の任意の位置でドロップして配置します。



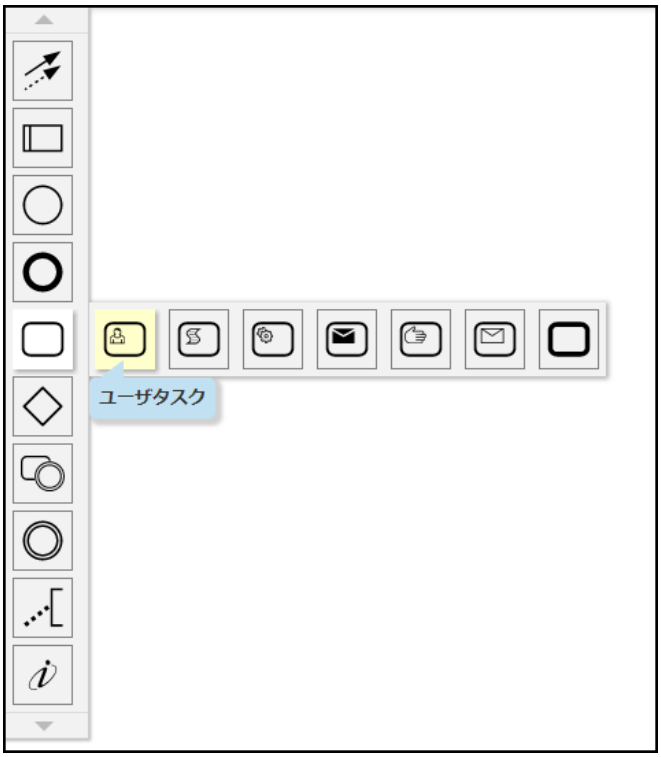
図：パレット - 開始イベント - 開始イベント

6. 「キャンバス」の余白をクリックし「プロパティエリア」にて、「プロセス」の「処理対象ユーザ」を設定します。

   プロセスを開始するユーザを設定してください。

図：プロセス - プロパティ - 処理対象ユーザ

7. 「パレット」から「キャンバス」に「ユーザタスク」を配置します。



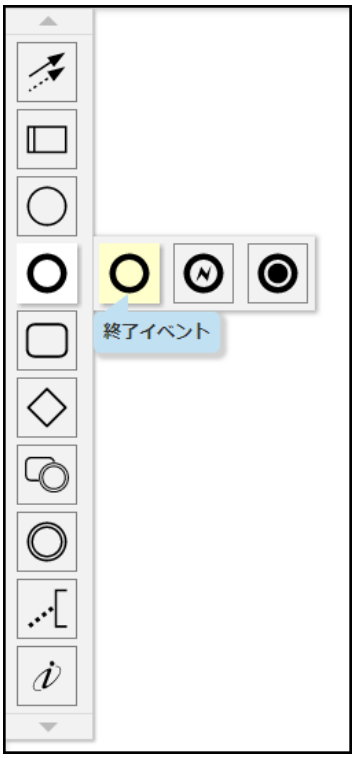
図：パレット - タスク - ユーザタスク

8. 「プロパティエリア」にて、「ユーザタスク」の「処理対象ユーザ」を設定します。

   タスクを処理するユーザを設定してください。

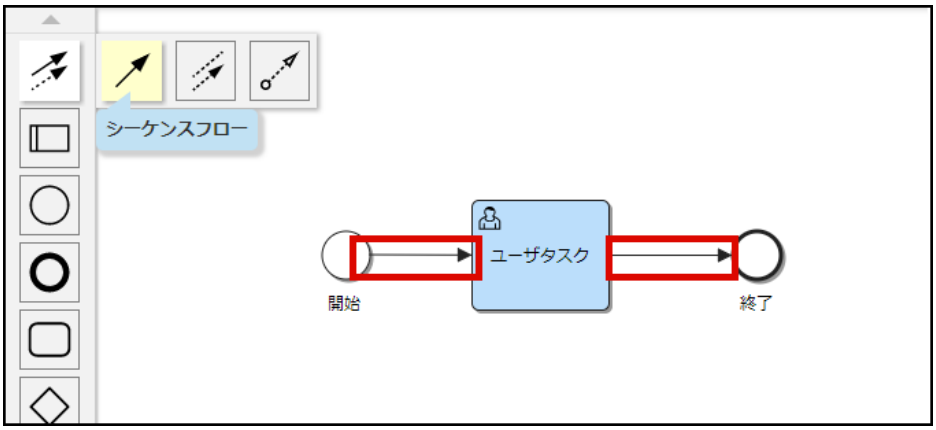
図：ユーザタスク - プロパティ - メインコンフィグ - 処理対象ユーザ

9. 「パレット」から「キャンバス」に「終了イベント」を配置します。



図：パレット - 終了イベント - 終了イベント

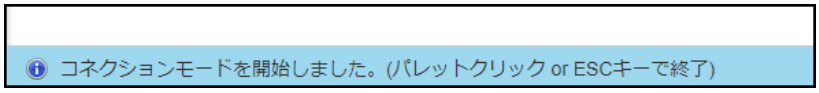
10. 「シーケンスフロー」で各エレメントを接続します。



図：パレット - コネクション - シーケンスフロー

1. 「パレット」にて、「  」にカーソルを合わせます。
2. 「パレット」の右側に現れる一覧から「シーケンスフロー」をクリックし、コネクションモードを開始します。

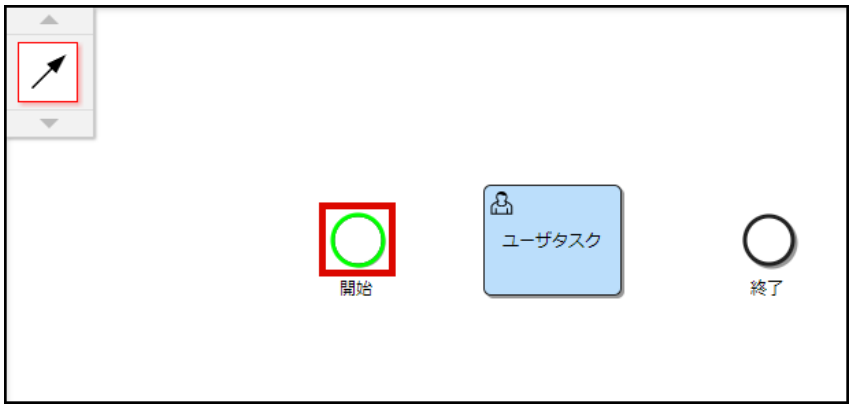
   コネクションモードが開始すると、フッタにメッセージが表示されます。



   図：フッタ

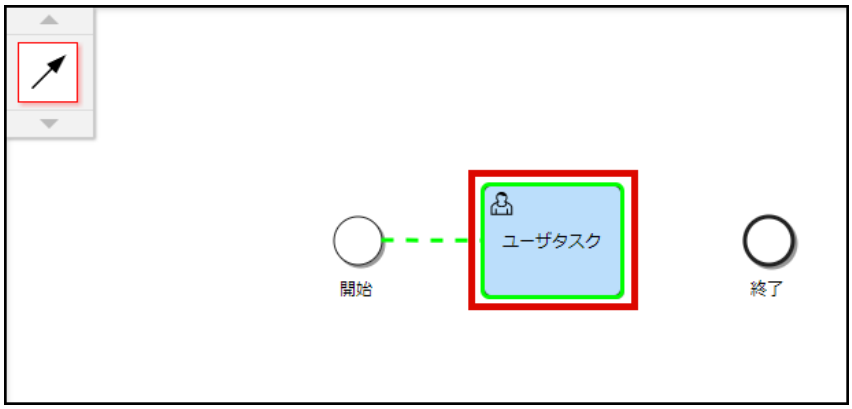
3. 「キャンバス」に配置された「開始イベント」をクリックまたはドラッグし、接続を開始します。

   コネクタの接続は、接続元と接続先の2クリック操作、または接続元からドラッグし接続先にドロップする操作どちらでも行えます。



図：接続を開始する

4. 「キャンバス」に配置された「ユーザタスク」上でクリックまたはドロップし、「シーケンスフロー」を接続します。

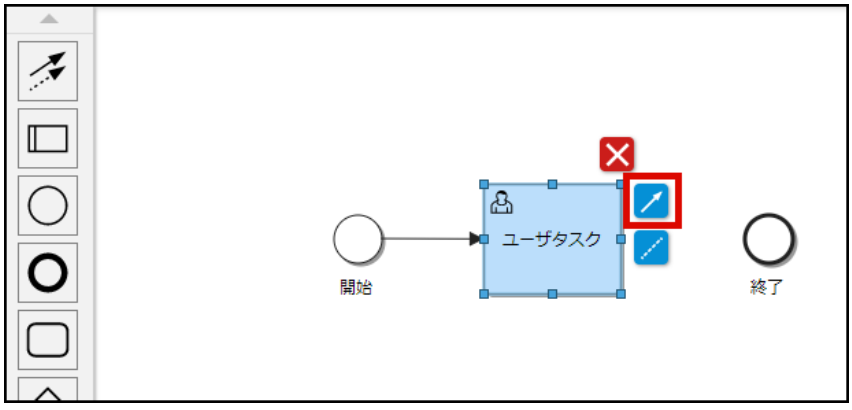


図：接続する

5. キーボードの「ESC」キーを押下するか、「パレット」の  をクリックし、コネクションモードを終了します。

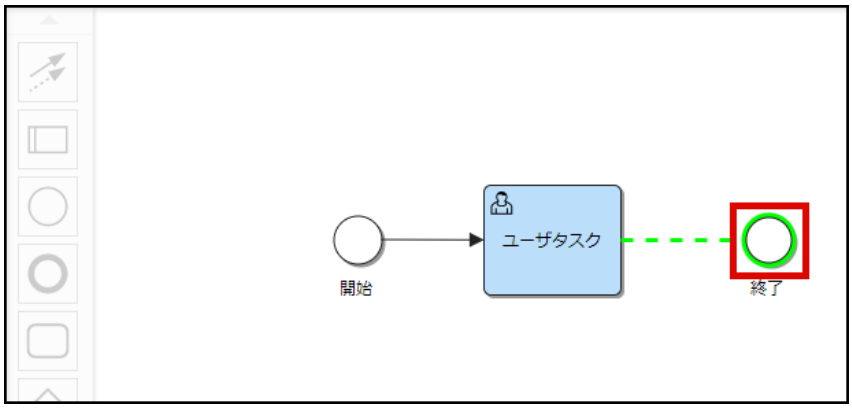
6. キャンバス上の「ユーザタスク」を選択します。

7. 「ユーザタスク」の右上に表示されるツールアイコン「シーケンスフローを引く」をクリックまたはドラッグし、接続を開始します。



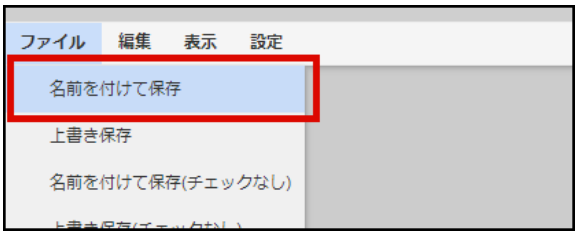
図：接続を開始する

8. 「キャンバス」に配置された「終了イベント」上でクリックまたはドロップし、「シーケンスフロー」を接続します。



図：接続する

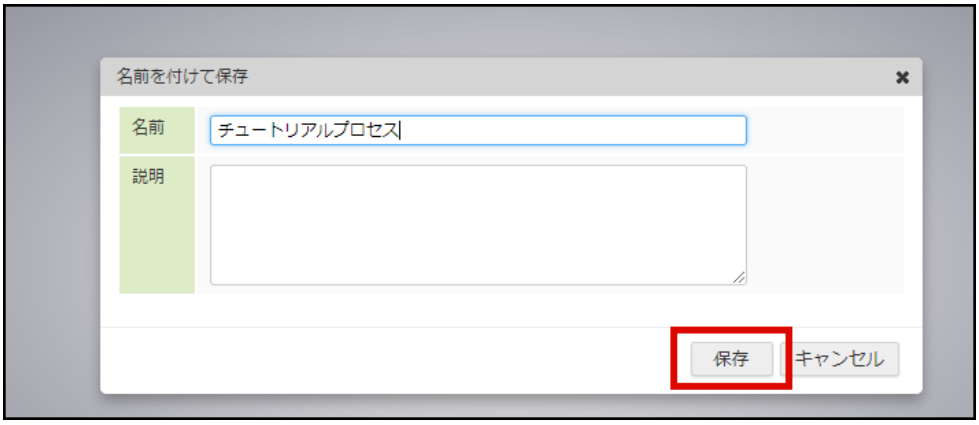
11. 「ヘッダメニュー」から、「ファイル」 - 「名前を付けて保存」をクリックします。



図：ヘッダメニュー - ファイル - 名前を付けて保存

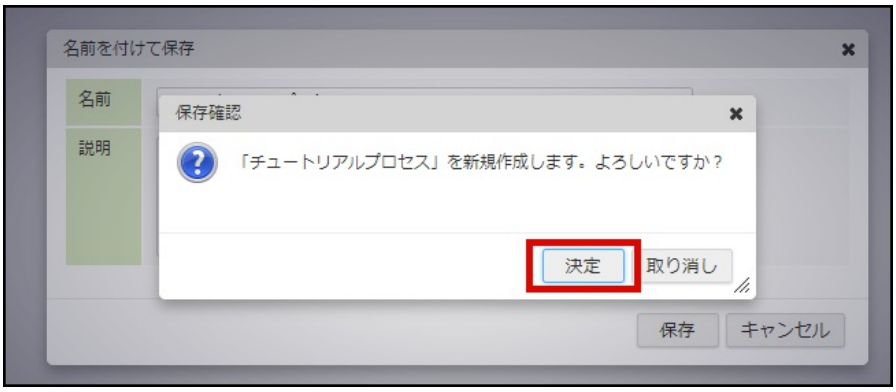
> **コラム**
>
> 「名前を付けて保存」と「名前を付けて保存（チェックなし）」の違い
>
> IM-BPM プロセスデザイナは、作成したプロセス定義がデプロイ可能かどうかをチェックを行う機能を有しています。
> 「名前を付けて保存」を実行した場合は上記チェックが行われ、デプロイ可能と判定されるまで保存できません。
> 「名前を付けて保存（チェックなし）」を実行した場合は上記チェックを行わず、保存が可能となる最低限のチェックのみを行って保存することができます。

12. 「名前」を入力して「保存」をクリックします。



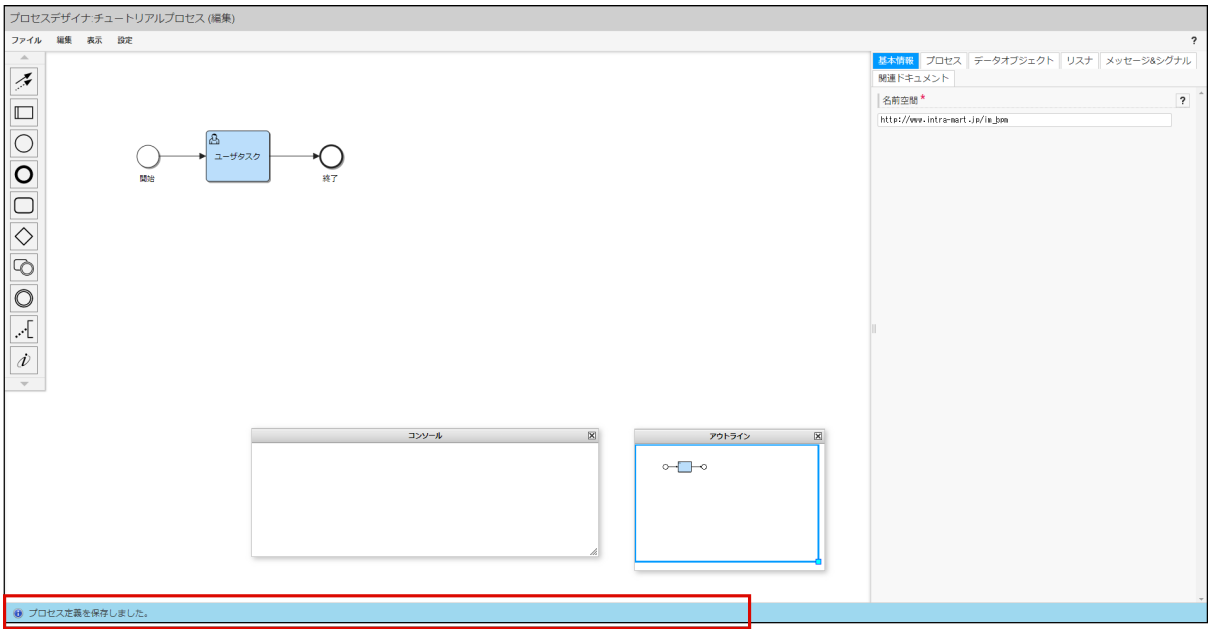
図：名前を付けて保存

13. 保存確認ダイアログで「決定」をクリックします。



図：名前を付けて保存（保存確認）

14. 保存されました。

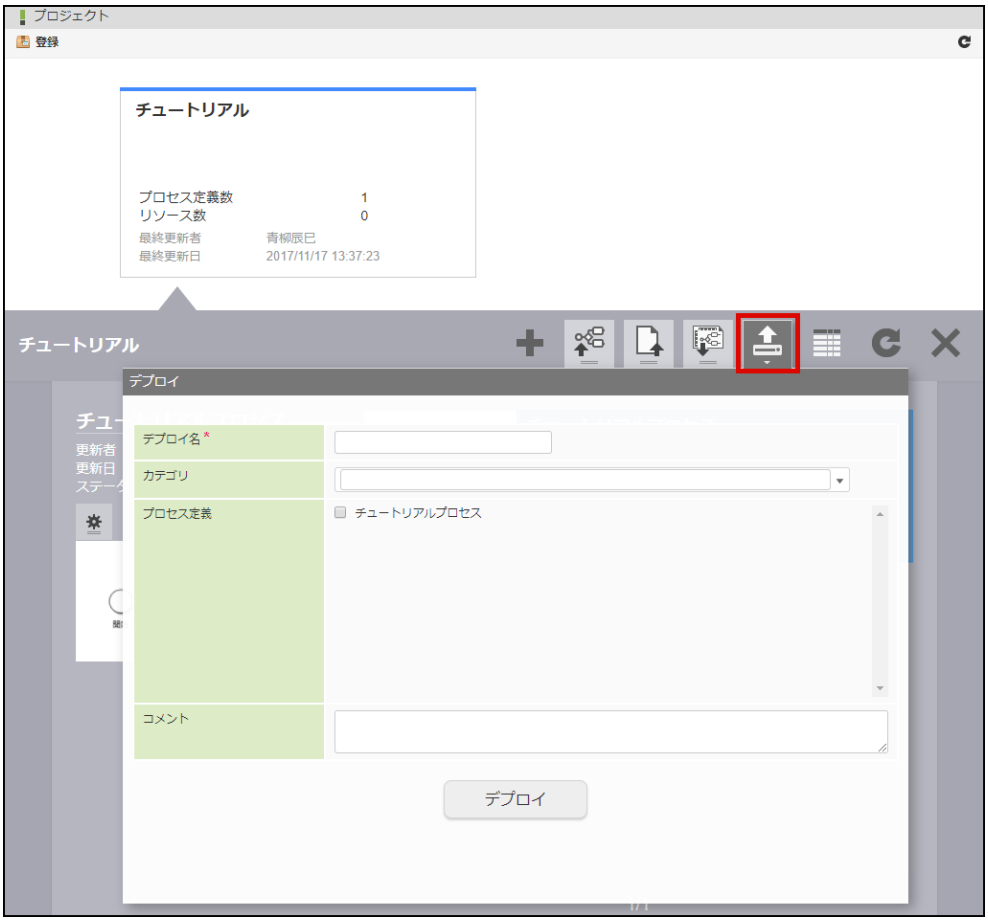


図：プロセス保存

プロセス定義を IM-BPM Runtimeへデプロイする

プロセス定義を IM-BPM Runtimeへデプロイします。

1. 「サイトマップ」→「BPM」→「プロセスデザイナ」をクリックします。
2. プロジェクトを選択し、  をクリックしデプロイエリアを開きます。

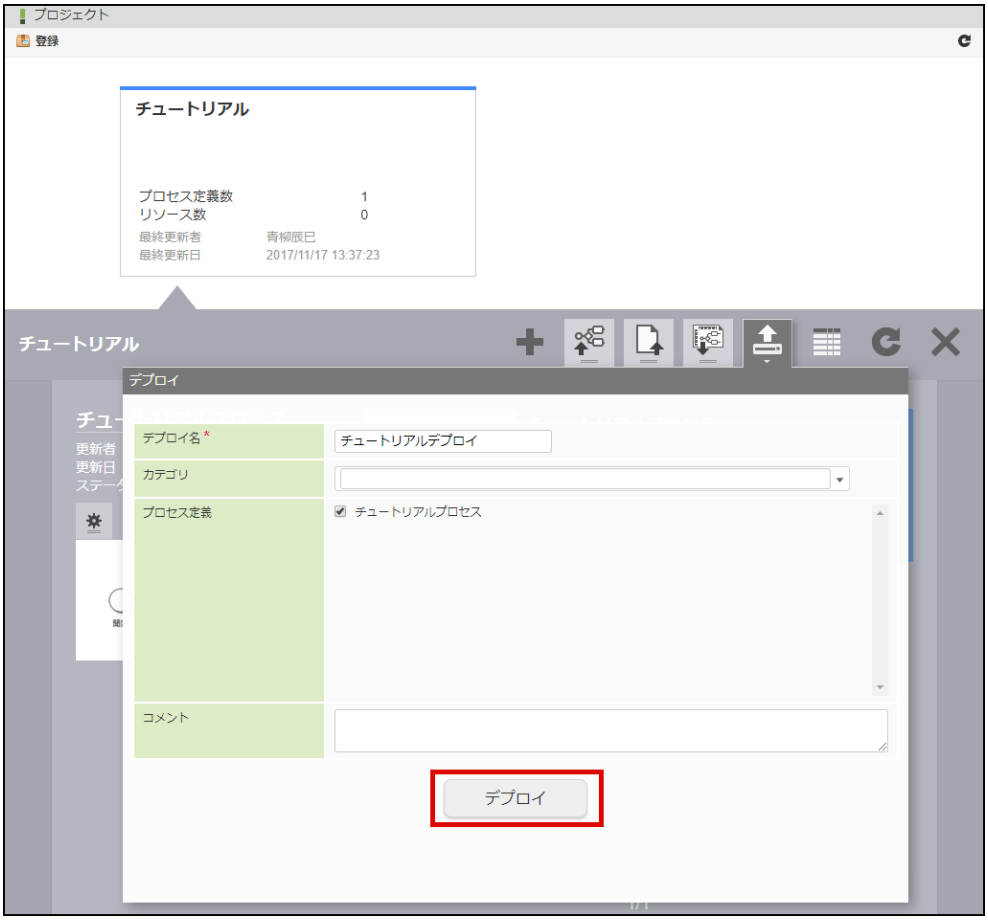


図：デプロイエリア

3. 「デプロイ名」の入力と「プロセス定義」の選択をします。

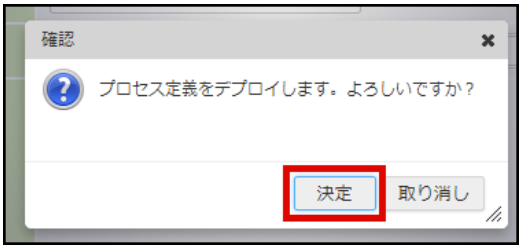
4. 「デプロイ」をクリックします。

※ デプロイについては「IM-BPM プロセスデザイナ 操作ガイド」 - 「デプロイ情報の入力」を参照してください。



図：デプロイ

5. 確認ダイアログにて「決定」をクリックします。



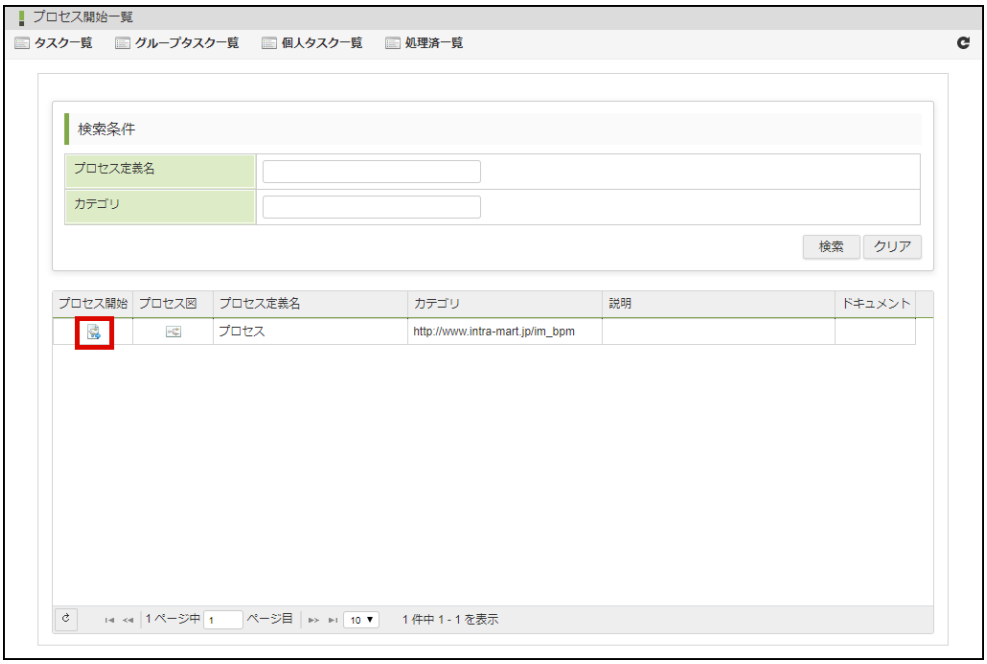
図：デプロイ確認

6. デプロイされました。

プロセスを開始する

IM-BPM Runtimeにデプロイしたプロセス定義のプロセスを開始します。

1. 「サイトマップ」→「BPM」→「プロセス開始一覧」をクリックします。
2. デプロイしたプロセス定義の  をクリックします。



図：プロセス開始一覧

3. プロセスが開始されました。

図：プロセス開始一覧

4. 「サイトマップ」→「BPM」→「タスク一覧」をクリックし、ユーザタスクが開始していることを確認します。

図：タスク一覧

> **コラム**
>
> ユーザタスクの操作については、「IM-BPM ユーザ操作ガイド」 を確認してください。

実用編

各種エレメントやプロパティの実用的な設定方法について説明します。
実用編の各チュートリアルは「*事前準備*」が完了していることを前提としています。

- データプロパティ
- マルチインスタンス
- リスナ
- 関連ドキュメント
- プール、レーン
- コールアクティビティ
- パラレルゲートウェイ
- 排他ゲートウェイ
- 包括ゲートウェイ
- イベント
- IM-LogicDesignerタスク
- 申請タスク
- 起票タスク
- IM-FormaDesigner for Accel Platform をユーザ入力フォームとして利用する

データプロパティ

データプロパティを定義する

「データプロパティ」は、以下のエレメントに定義できる変数宣言です。

- プロセス
- プール
- サブプロセス
- イベントサブプロセス

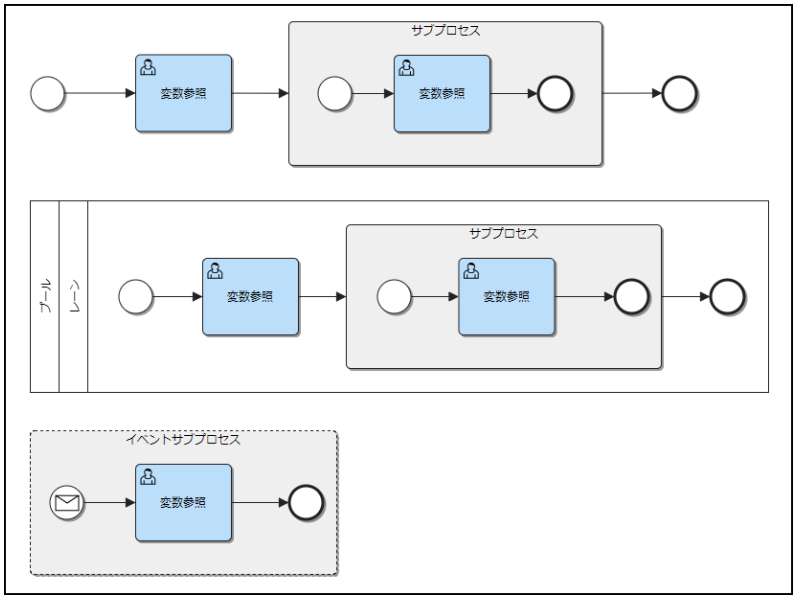
「データプロパティ」で定義した変数をエレメントから参照／更新できます。

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **data_property_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- エレメントの配置
- プロセスにデータプロパティを設定する
- サブプロセスにデータプロパティを設定する
- プールにデータプロパティを定義する
- プール内のサブプロセスにデータプロパティを定義する
- イベントサブプロセスにデータプロパティを定義する
- データプロパティの参照可能範囲を確認する

エレメントの配置

配置イメージを参考に、各エレメントを配置してシーケンスフローで接続します。



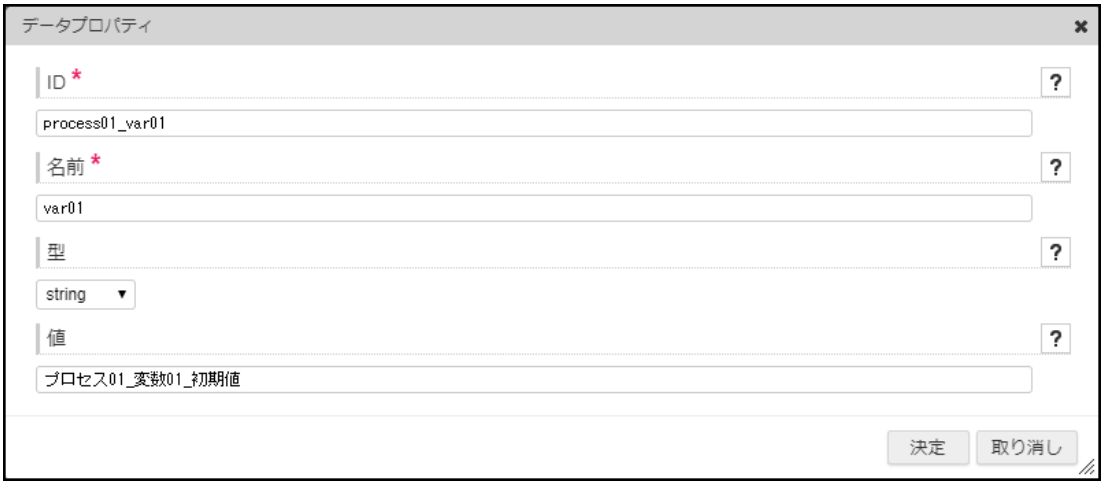
図：配置イメージ

1. 「開始イベント」、「ユーザタスク」、「サブプロセス」、「終了イベント」を配置
2. 上記で配置した「サブプロセス」内に「開始イベント」、「ユーザタスク」、「終了イベント」を配置
3. 「プール」を配置（このとき、「レーン」も同時に配置されます）
4. 上記で配置した「プール」内に「開始イベント」、「ユーザタスク」、「サブプロセス」、「終了イベント」を配置
5. 上記で配置した「プール」内の「サブプロセス」内に、「開始イベント」、「ユーザタスク」、「終了イベント」を配置
6. 「イベントサブプロセス」を配置
7. 上記で配置した「イベントサブプロセス」内に「メッセージ開始イベント」、「ユーザタスク」、「終了イベント」を配置
8. 配置イメージを参考に、配置したエレメント間を「シーケンスフロー」で接続します。

プロセスにデータプロパティを設定する

キャンバスに直接エレメントを配置した場合、キャンバスそのものが一つのプロセス定義として扱われます。
これを「プロセス」と呼びます。
この「プロセス」にデータプロパティを定義します。

1. キャンバス内のエレメントが配置されていない場所をクリックするか、「ヘッダメニュー > 編集 > 選択を解除」を実行することで、プロパティエリアに「プロセス」のプロパティが表示されます。
2. 「データオブジェクト」タブを開きます。
3. 「追加」をクリックし、ダイアログを開きます。
4. 以下のようにプロパティを入力します。



図：プロセス：プロパティ - データオブジェクト

項目名	入力値	説明
ID	process01_var01	このファイル内で一意となるID値を入力します。 データプロパティ内だけでなく、各エレメントに設定した「基本情報」タブの「ID」と重複することもできません。 「ID」の値は、システムが変数を一意に特定するためのものであり、変数名ではありません。
名前	var01	変数名を入力します。 同一のデータプロパティ内で同じ名前の変数を複数個定義しても、一つの変数として扱われます。
型	string	以下の型を選択できます。 **型名** / **説明** string / 文字列 boolean / 真偽値 datetime / 日時（ISO8601拡張形式） double / 8バイト倍精度浮動小数点数 int / 4バイト整数（-2147483648～2147483647） long / 8バイト整数（-9223372036854775808～9223372036854775807）
値	プロセス01_変数01_初期値	選択した「型」に合わせて、変数の初期値を入力します。 「型」と「値」の入力値に矛盾があった場合、ファイルの保存時にエラーが発生します。

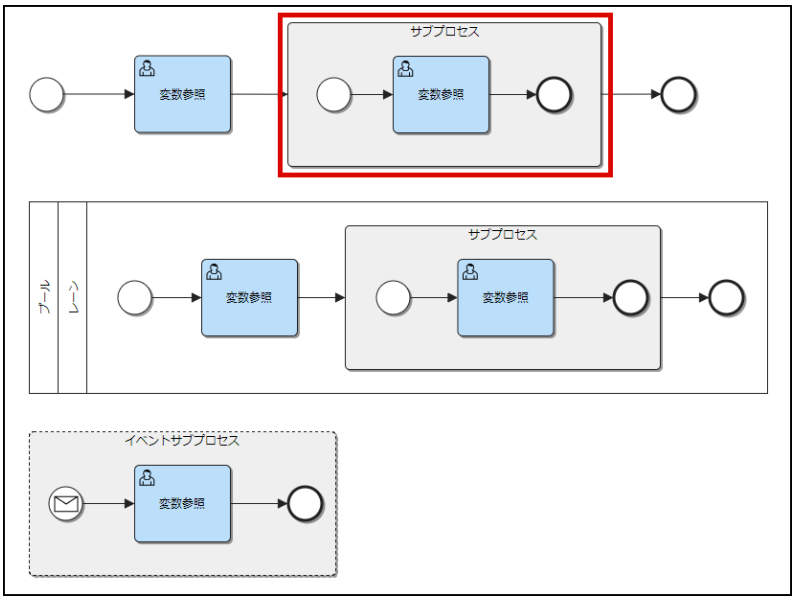
5. 「決定」をクリックして設定値を表に反映します。
6. 同様の手順で、以下の変数も定義します。

項目名	入力値
ID	process01_var02
名前	var02
型	datetime
値	2017-12-01T08:00:00+09:00

サブプロセスにデータプロパティを設定する

キャンバスに直接配置されている「サブプロセス」にデータプロパティを定義します。

1. 図の赤枠で示した「サブプロセス」を選択します。



図：サブプロセス

2. プロパティエリアで「データオブジェクト」タブを開きます。

3. プロセスのデータプロパティを設定した手順と同様の手順で、以下のデータプロパティを定義します。

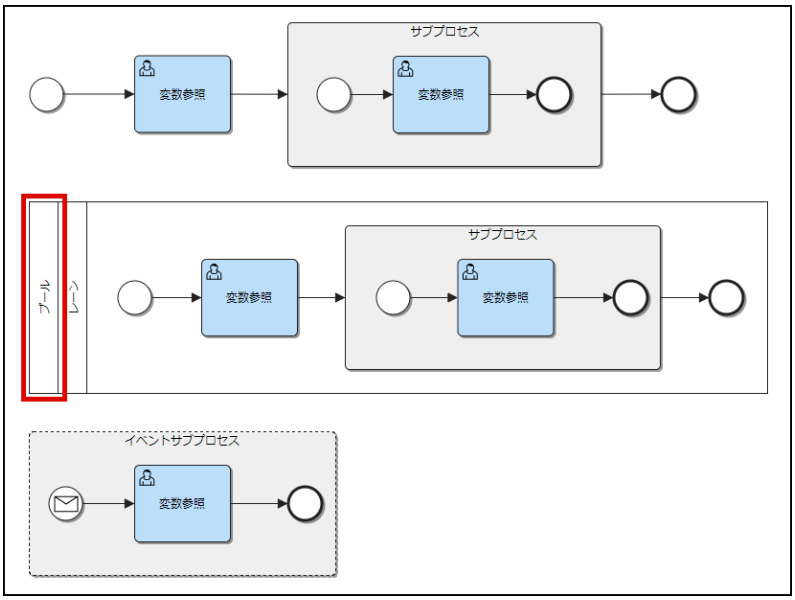
図：サブプロセス：プロパティ - データオブジェクト

ID	名前	型	値
subProcess01_var01	var01	int	1
subProcess01_var03	var03	boolean	true

プールにデータプロパティを定義する

「プール」にデータプロパティを定義します。
「プール」は一つのプロセス定義として扱われます。
「プロセス」と「プール」はそれぞれが一つのプロセス定義であり、親子関係はありません。

1. 図の赤枠で示した「プール」を選択します。



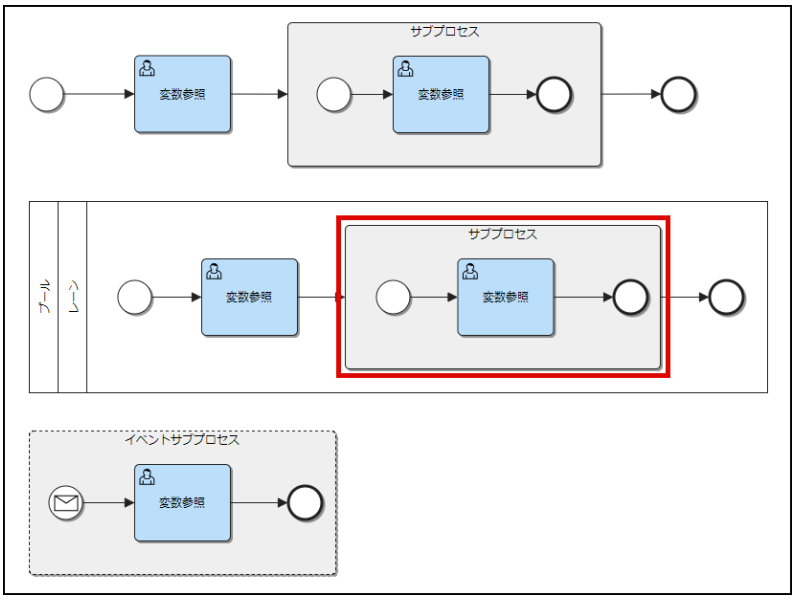
図：プール：プロパティ - データオブジェクト

2. 以下の通り、データプロパティを定義します。

ID	名前	型	値
process02_var04	var04	double	1.41421356
process02_var05	var05	long	1

プール内のサブプロセスにデータプロパティを定義する

1. 図の赤枠で示した「サブプロセス」を選択します。



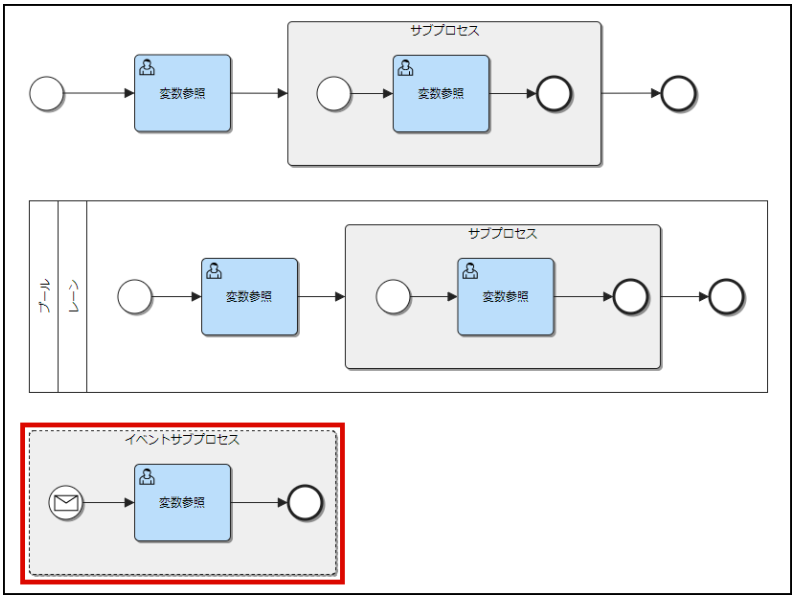
図：サブプロセス：プロパティ - データオブジェクト

2. 以下の通り、データプロパティを定義します。

ID	名前	型	値
subProcess02_var03	var03	boolean	false
subProcess02_var06	var06	double	2.2360679

イベントサブプロセスにデータプロパティを定義する

1. 図の赤枠で示した「イベントサブプロセス」を選択します。



図：イベントサブプロセス：プロパティ - データオブジェクト

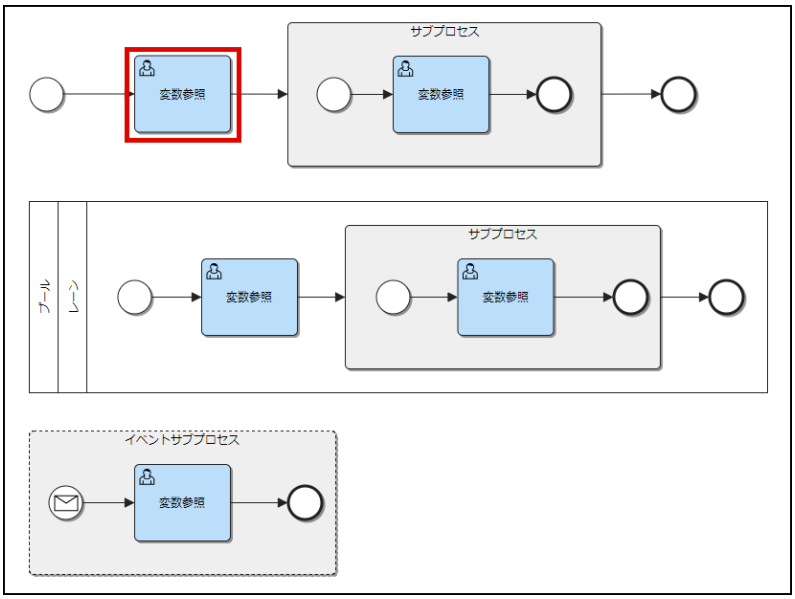
2. 以下の通り、データプロパティを定義します。

ID	名前	型	値
subProcess03_var07	var07	string	（入力なし）

データプロパティの参照可能範囲を確認する

各部に配置した「ユーザタスク」を利用して、データプロパティの参照可能範囲を確認します。

1. 図の赤枠で示した「ユーザタスク」を選択します。

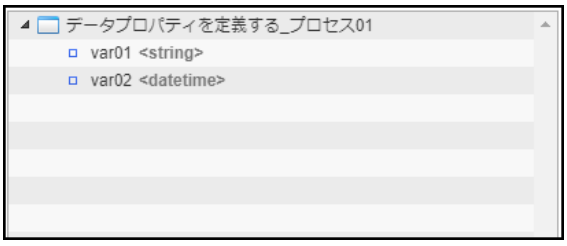


図：ユーザタスクの選択

2. プロパティの任意の項目で、 ${} をクリックします。

3. 開いたEL式エディタの右側にある「データプロパティツリー」を確認します。

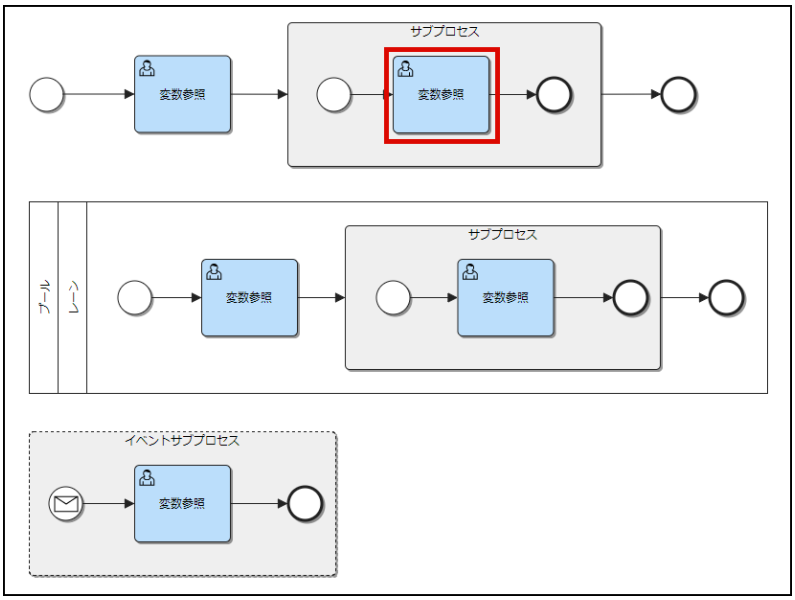
4. プロセス「データプロパティを定義する_プロセス01」のデータプロパティ「var01 <string>」と「var02 <datetime>」が参照できることがわかります。



図：EL式エディタ - データプロパティツリー

> **コラム**
>
> ここでは、プール、サブプロセス、イベントサブプロセスに定義したデータプロパティは参照できません。

5. 図の赤枠で示した「ユーザタスク」を選択します。



図：ユーザタスクの選択

6. EL式エディタを開き、「データプロパティツリー」を確認します。

7. プロセス「データプロパティを定義する_プロセス01」のデータプロパティに加え、サブプロセスで定義したデータプロパティ「var01 <int>」と

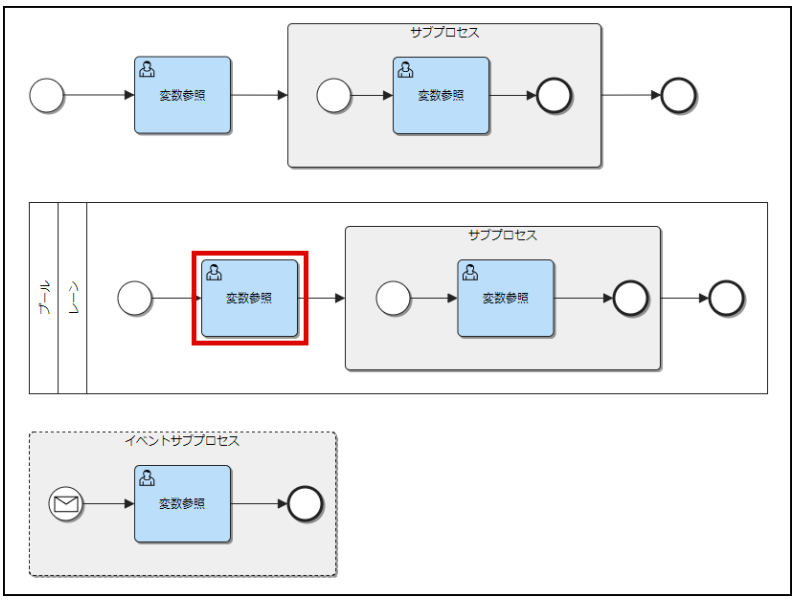
「var03 <boolean>」が参照できることがわかります。

図：EL式エディタ - データプロパティツリー

> **コラム**
>
> 親となるコンテナをさかのぼって、プロセスグローバルの変数も参照できます。
> 例えばサブプロセス内にさらにサブプロセスがあった場合、最も深い階層のサブプロセスは自身で定義されたデータプロパティに加え 、親のサブプロセスとプロセスのデータプロパティを参照できます。

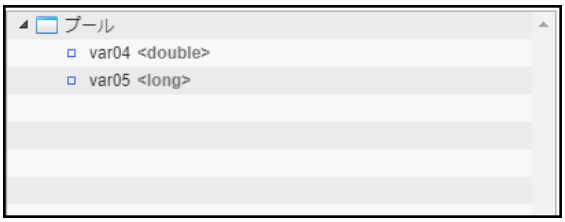
> **コラム**
>
> プロセスグローバルの定義とサブプロセスの定義で重複して命名されている変数「var01」を参照する場合、サブプロセスで定義された変数が参照されます。
> つまり、直近の階層で定義された変数が優先的に参照されます。

8. 図の赤枠で示した「ユーザタスク」を選択します。



図：ユーザタスクの選択

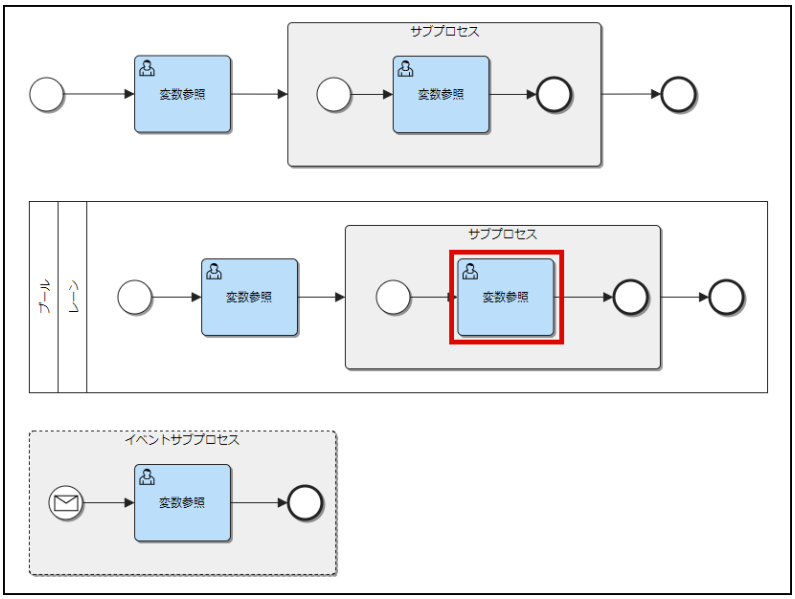
9. EL式エディタを開き、「データプロパティツリー」を確認します。
10. 「プール」のデータプロパティ「var04 <double>」と「var05 <long>」が参照できることがわかります。



図：EL式エディタ - データプロパティツリー

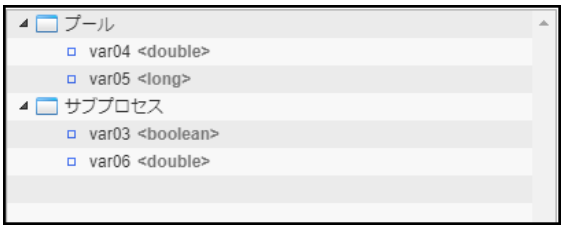
> **コラム**
>
> プールは一見、プロセスの中に配置されているように見えますが、そうではありません。
> プールは一つのプロセス定義として扱われるため、プロセスの子エレメントになることはありません。
> プロセス定義を跨って変数を参照することはできないため、プロセスで定義されたデータプロパティは参照できません。

11. 図の赤枠で示した「ユーザタスク」を選択します。



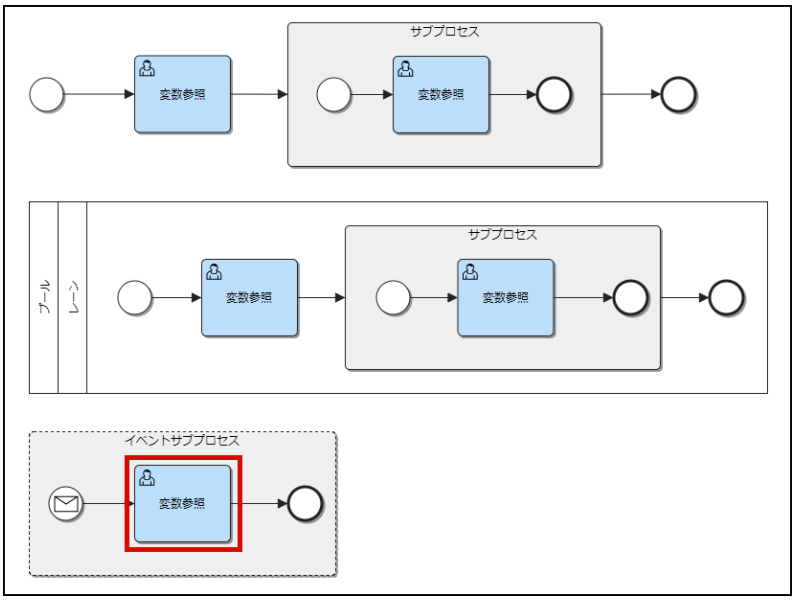
図：ユーザタスクの選択

12. EL式エディタを開き、「データプロパティツリー」を確認します。
13. 「プール」のデータプロパティに加え、プール内サブプロセスのデータプロパティ「var03 <boolean>」と「var06 <double>」が参照できることがわかります。



図：EL式エディタ - データプロパティツリー

14. 図の赤枠で示した「ユーザタスク」を選択します。



図：ユーザタスクの選択

15. EL式エディタを開き、「データプロパティツリー」を確認します。
16. プロセス「データプロパティを定義する_プロセス01」のデータプロパティに加え、イベントサブプロセスで定義したデータプロパティ「var07 <string>」が参照できることがわかります。

図：EL式エディタ - データプロパティツリー

マルチインスタンス

マルチインスタンス を使用する

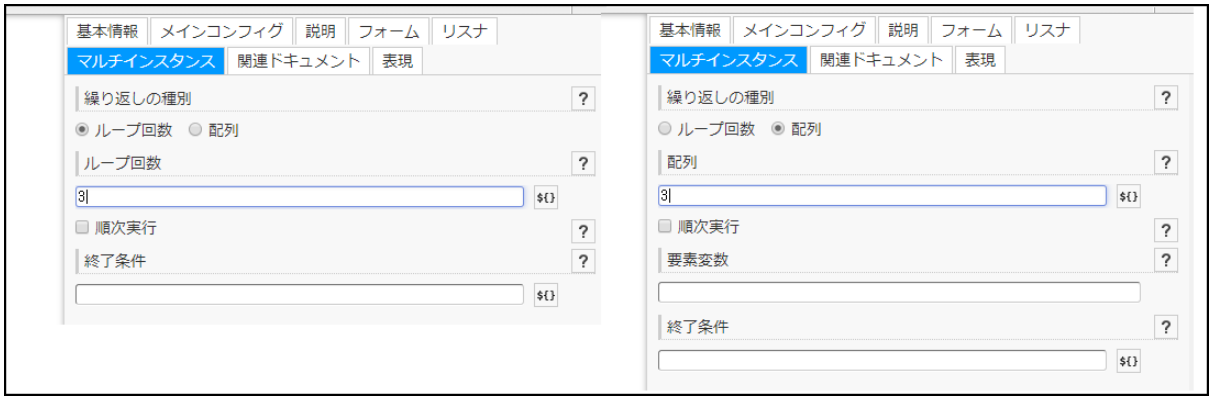
このチュートリアルでは、マルチインスタンスの定義を使用して複数のインスタンスを並列および、順次に処理する方法を解説します。
マルチインスタンス の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「マルチインスタンス」も合わせて参照してください。

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **multi_instance_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- プロパティを確認する
- ループ回数を指定して使用する
- 配列を指定して使用する
- 並列または、順次で実行する
- 要素変数を利用する
- 終了条件を利用する

プロパティを確認する

プロパティを確認します。



図：プロパティ

- 繰り返しの種別

  ループ回数か配列を選択します。

- ループ回数

  繰り返しの種別が「ループ回数」の場合、表示されます。
  回数（数字）または、回数が保存されているオブジェクトを指定します。

- 配列

  繰り返しの種別が「配列」の場合、表示されます。
  繰り返しに使用する配列（コレクションのオブジェクト）を指定します。

- 順次実行

  順次で実行するか並列で実行するか設定します。
  有効の場合、順次で実行されます。

無効の場合、並列で実行されます。

- 要素変数

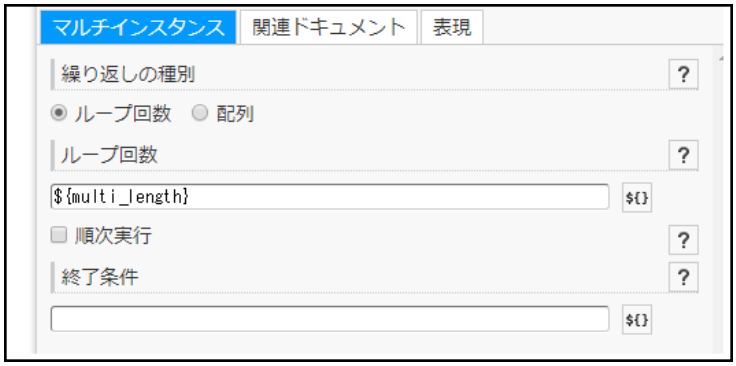
  繰り返しの種別が「配列」の場合、表示されます。
  「配列」の要素が「要素変数」に保存されます。

- 終了条件

  終了条件を設定します。

ループ回数を指定して使用する

「ループ回数」を指定して使用します。
※ 並列で実行します。



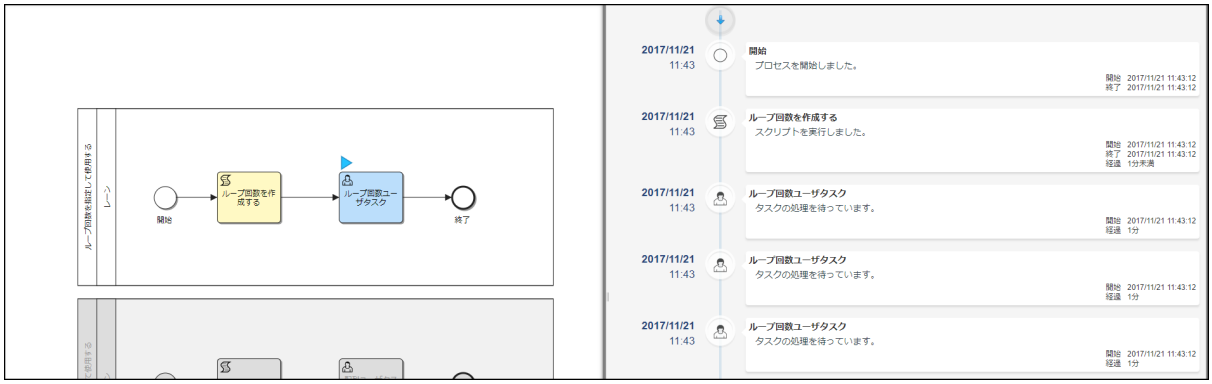
図：プロパティの設定

1. 任意の方法で、multi_length の変数に回数を設定します。

   ※ 本チュートリアルのサンプルでは、「スクリプトタスク」を使用して、multi_length の変数に回数 3 を設定しています。

2. マルチインスタンスの「繰り返しの種別」を「ループ回数」に設定します（初期設定は「ループ回数」）。

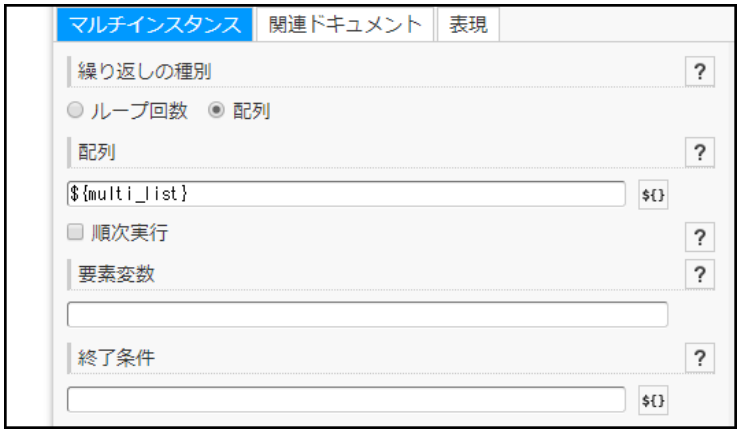
3. 「ループ回数」にEL式を使用して、multi_length の変数を指定します。



図：実行時のイメージ

配列を指定して使用する

「配列」を指定して使用します。
※ 並列で実行します。



図：プロパティの設定

1. 任意の方法で、multi_list の変数にコレクションを設定します。

   ※ 本チュートリアルのサンプルでは、「スクリプトタスク」を使用して、multi_list の変数にコレクションを設定しています。
   ['data1', 'data2', 'data3']

2. マルチインスタンスの「繰り返しの種別」を「配列」に設定します（初期設定は「ループ回数」）。

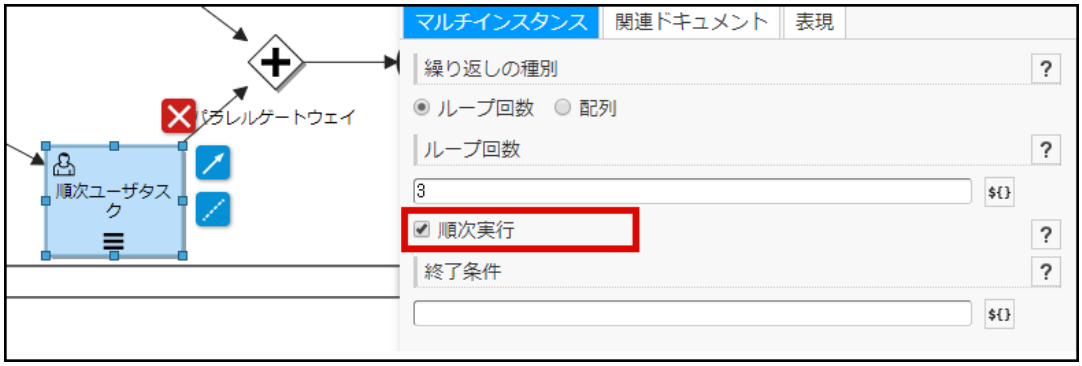
3. 「配列」にEL式を使用して、multi_list の変数を指定します。

図：実行時のイメージ

並列または、順次で実行する

並列または、順次で実行します。

図：並列



図：順次

1. 並列で実行する場合、「順次実行」のチェックボックスを無効にします。

2. 順次で実行する場合、「順次実行」のチェックボックスを有効にします。

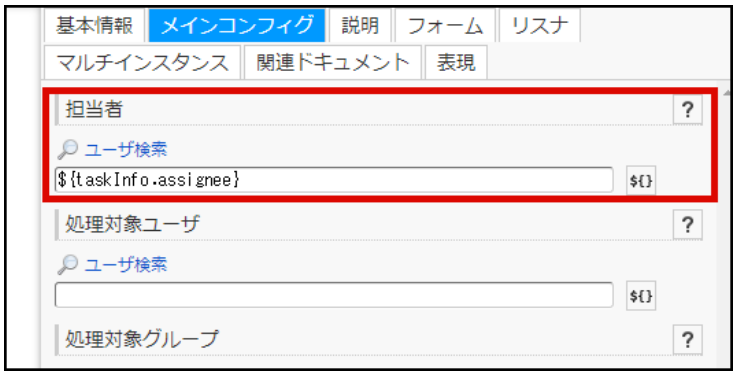
図：ユーザタスク到達時イメージ

要素変数を利用する

マルチインスタンスの要素変数を利用し、ユーザタスクの「タスク名」と「担当者」を設定します。
※ 並列で実行します。

図：プロパティの設定 - マルチインスタンス

図：プロパティの設定 - 基本情報



図：プロパティの設定 - メインコンフィグ

1. 任意の方法で、multi_list の変数にコレクションを設定します。

   ※ 本チュートリアルのサンプルでは、「スクリプトタスク」を使用して、multi_list の変数にコレクションを設定しています。
   [{'taskName' : 'タスク名１', 'assignee', 'aoyagi'}, {'taskName' : 'タスク名２', 'assignee', 'ueda'}, {'taskName' : 'タスク名３', 'assignee', 'ikuta'}]

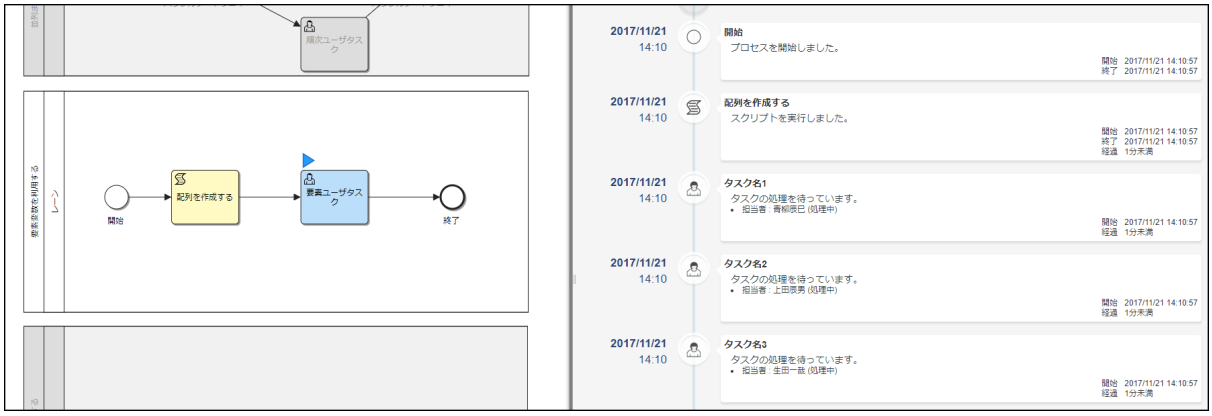
2. マルチインスタンスの「繰り返しの種別」を「配列」に設定します（初期設定は「ループ回数」）。

3. 「配列」にEL式を使用して、multi_list の変数を指定します。

4. 「配列変数」に変数名 taskInfo を設定します。

5. プロパティの「基本情報」 - 「別名」にEL式を使用して、taskInfo.taskName を指定します。

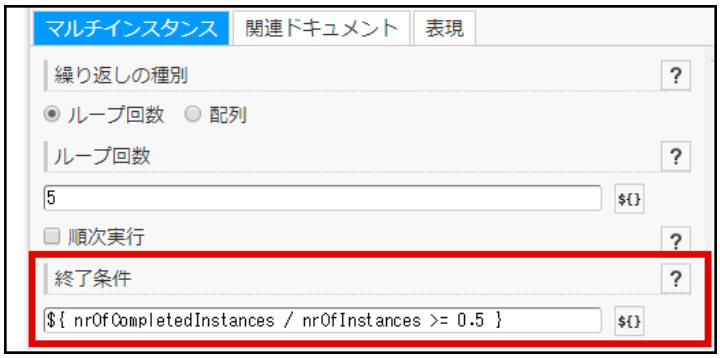
6. プロパティの「メインコンフィグ」 - 「担当者」にEL式を使用して、taskInfo.assignee を指定します。



図：実行時のイメージ

終了条件を利用する

「終了条件」を利用します。
※ 並列で実行します。



図：プロパティの設定

マルチインスタンスの実行時にスコープの変数として以下の3つの変数が作成されます。

- nrOfInstances

  マルチインスタンスの総件数が設定されます。

- nrOfCompletedInstances

  マルチインスタンスの完了した件数が設定されます。

- nrOfActiveInstances

  マルチインスタンスの完了していない件数が設定されます。

終了条件に設定する例です。

- マルチインスタンスの半分以上が完了したら終了させたい場合

  ${ nrOfCompletedInstances / nrOfInstances >= 0.5 }

- マルチインスタンスのうち、2件が完了したら終了させたい場合

  ${ nrOfCompletedInstances == 2 }

- マルチインスタンスのうち、残り1件になったら終了させたい場合

${ nrOfActiveInstances == 1 }

- 任意のフラグがtrueになったら終了させたい場合

  ${ %任意のフラグ% }

「終了条件」が満たされない場合は、全てのインスタンスが完了した場合に次のフローエレメントに遷移します。

リスナ

IM-LogicDesignerのリスナを利用する

このチュートリアルでは、「IM-LogicDesigner」のリスナを利用する方法を解説します。
「IM-LogicDesigner」とは、intra-mart Accel Platform上でビジネスロジックを簡単に作成できるアプリケーションです。
「IM-LogicDesigner」の詳細については、「IM-LogicDesigner仕様書」を参照してください。

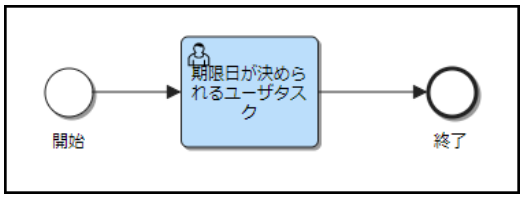
> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **listener_logic_designer.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- IM-LogicDesignerのリスナを利用したプロセス定義を作成する
- 実行結果を確認する

IM-LogicDesignerのリスナを利用したプロセス定義を作成する

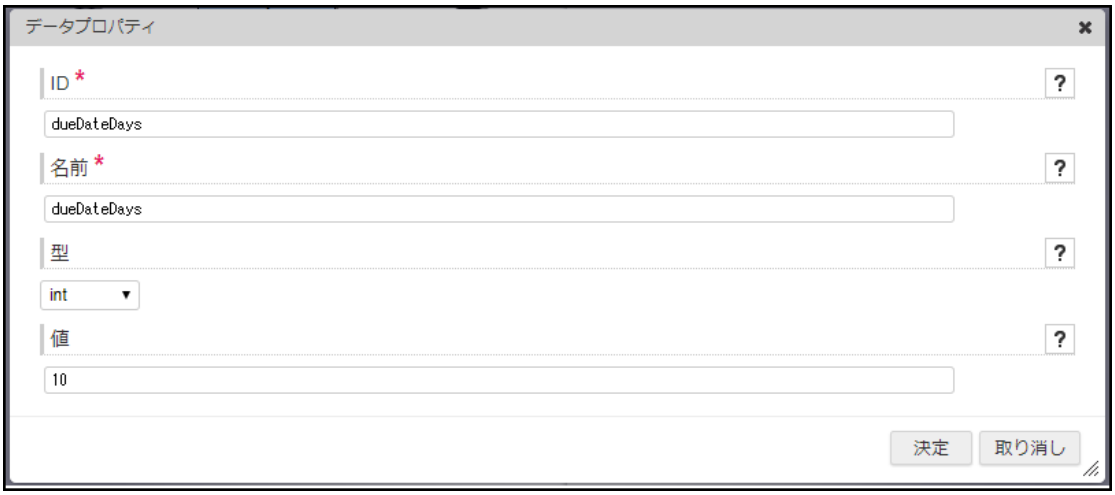
IM-LogicDesignerのリスナを利用して、基準日から数日後の営業日を取得し、ユーザタスクの期限日に設定します。
このチュートリアルでは、基準日を現在日に設定し、10日後の営業日を期限日に設定するようにします。
このチュートリアルを開始する前に以下リンクから「ロジックフロー定義」をダウンロードし、インポートしてください。
im_logicdesigner-data-listener.zip

> **コラム**
>
> 「ロジックフロー定義」のインポートについての詳細は「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」を参照してください。



図：完成イメージ

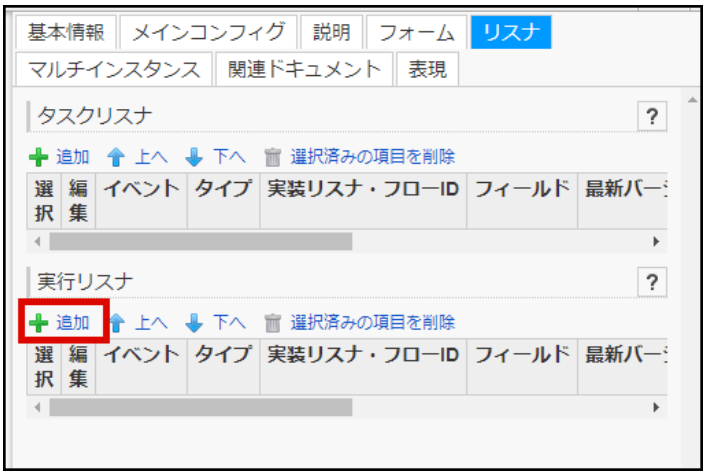
1. プロセスのプロパティ「データオブジェクト」に dueDateDays = 10 のデータプロパティを定義します。



図：プロセス：プロパティ - データオブジェクト

> **コラム**
>
> プロセスのデータオブジェクト設定方法については「*プロセスにデータプロパティを設定する*」を参照してください。

2. ユーザタスクのリスナを追加します。
   実行リスナの「追加」リンクをクリックします。



図：ユーザタスク - リスナ

3. タイプを IM-LogicDesigner に設定します。

図：リスナ

4. フローIDを設定します。

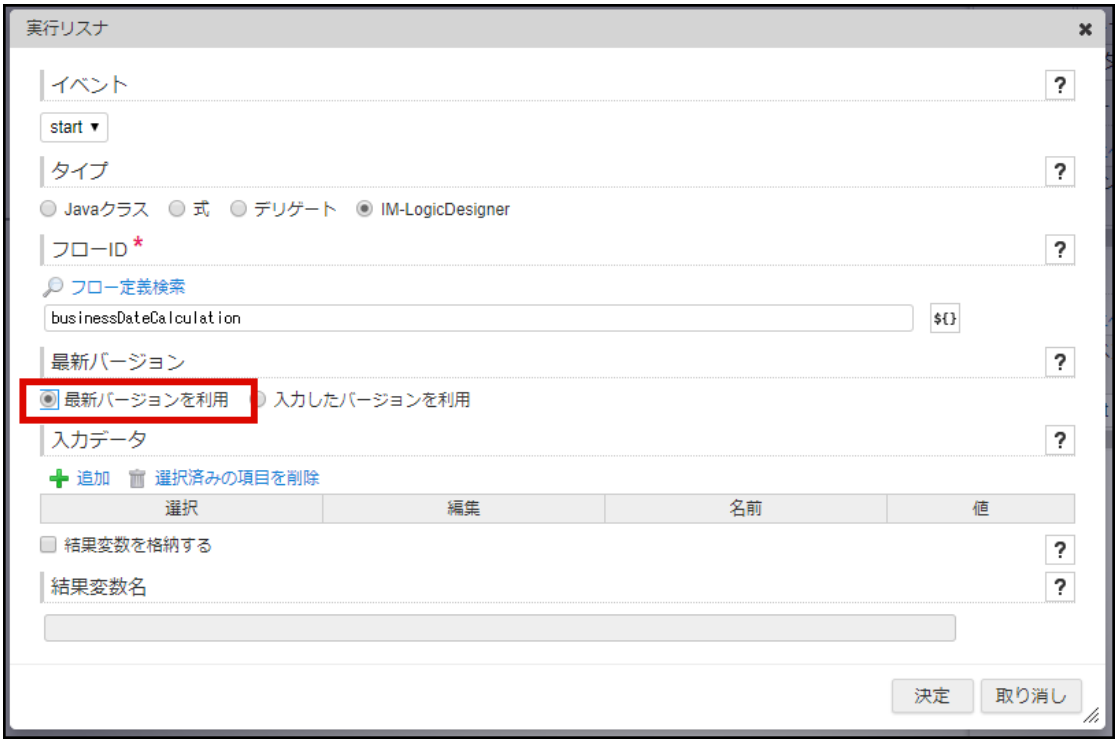
   フローIDを設定するには、以下3つの方法があります。

   - **「フロー定義検索」により、ロジックフロー定義を選択する**
     1. 「フロー定義検索」をクリックし、「ロジックフロー定義検索」ウィンドウを開きます。
     2. フロー定義ID businessDateCalculation のロジックフローを選択し、「決定」ボタンをクリックします。
   - **フローIDを文字列で入力する**
     1. フローIDの入力フォームに直接 businessDateCalculation を入力します。
   - **EL式で動的にフローIDを設定する**

     EL式による動的なフローIDの設定方法については、別途「*IM-LogicDesignerタスクで実行するロジックフローを動的に設定する*」で説明してい

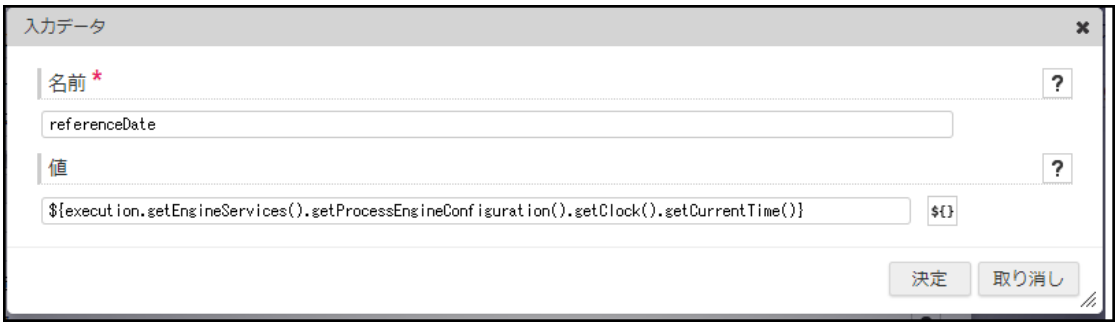
ます。

5. 最新バージョンを利用するように設定します。

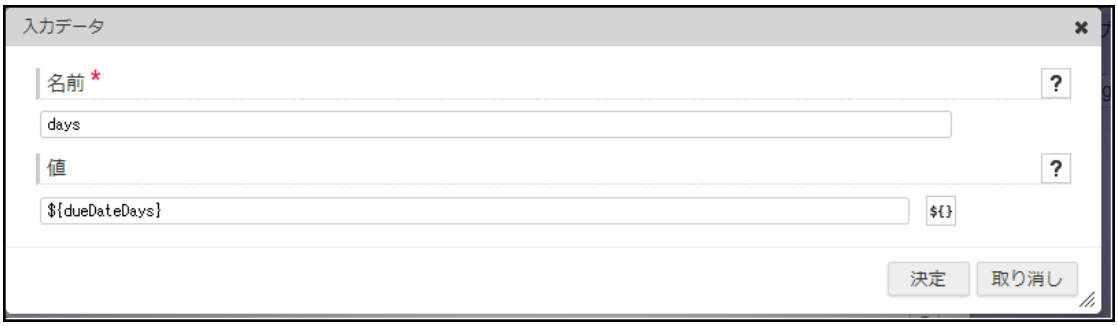


図：リスナ

6. 入力データを以下のように設定します。
   - 名前: referenceDate , 値: ${execution.getEngineServices().getProcessEngineConfiguration().getClock().getCurrentTime()}
   - 名前: days , 値: ${dueDateDays}



図：基準日の設定

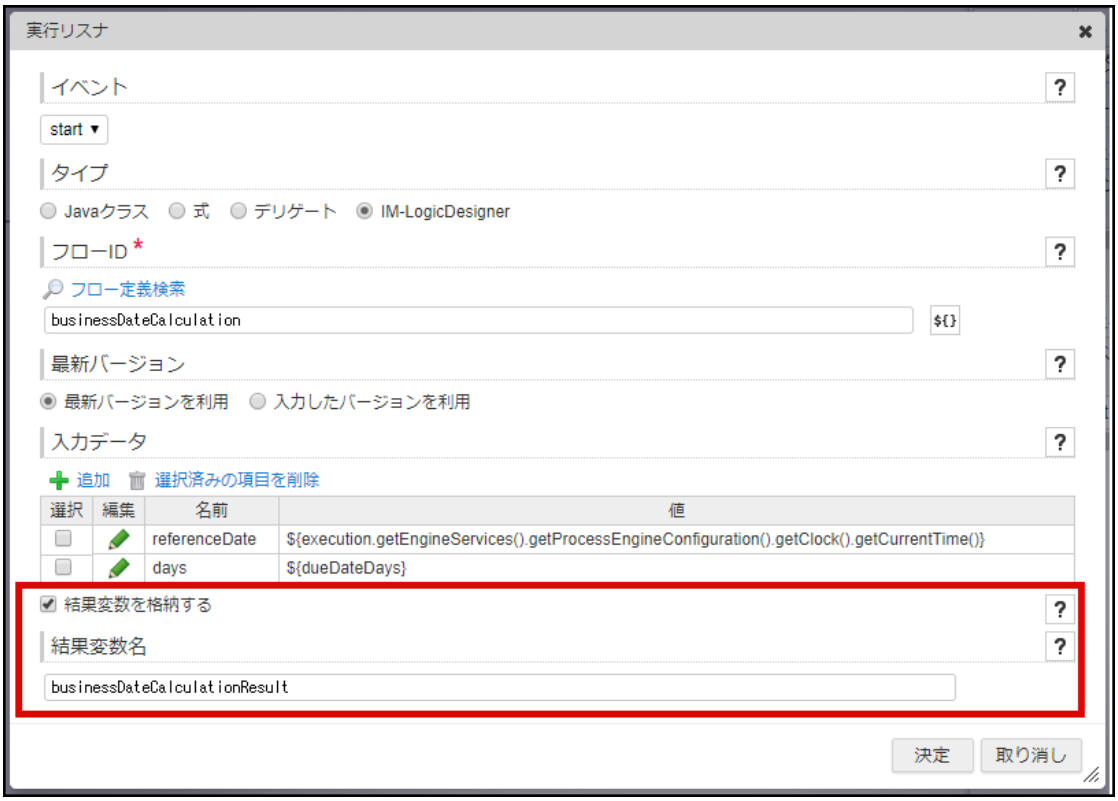


図：日数の設定

図：ロジックフロー側の対応する入力設定

7. 「結果変数を格納する」を有効にします。

8. 結果変数名に businessDateCalculationResult を設定します。



図：リスナ

9. ユーザタスクの期限に ${businessDateCalculationResult.addedBusinessDate} を追加します。

図：ユーザタスク - プロパティ - メインコンフィグ

図：ロジックフロー側の対応する出力設定

実行結果を確認する

このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

1. プロセスを開始します。
2. タスク一覧を確認します

図：タスク一覧

タスクリスナを利用してタスクの処理依頼メールを送信する

このチュートリアルでは、タスクリスナを利用する方法を解説します。
「タスクリスナ」を設定することで、任意の処理をタスクのイベント発生時に実行できます。
「タスクリスナ」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「タスクリスナ」もあわせて参照してください。

タスクリスナを使って処理依頼メールを送信する際に、「IM-LogicDesigner」と「IM-FormaDesigner」を使用します。
チュートリアルを開始する前に「ロジックフロー定義」と「IM-FormaDesigner」で作成したアプリケーションをインポートしてください。
im_logicdesigner-data-listener_send_mail.zip
select_usercd.zip
conduct_button.zip

また、タスク処理依頼メールの受信者としてサンプルユーザ「上田辰男」（ユーザコード：ueda）を使用します。
「サイトマップ」→「共通マスタ」→「ユーザ」から「上田辰男」を検索し、以下の設定を行ってください。

- 「ロール」タブ - 「IM-BPMユーザ」ロールの追加
- 「プロファイル」タブ - 「メールアドレス」の設定

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **listener_send_mail.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」 - 「プロセス定義のアップロード」を参照してください。

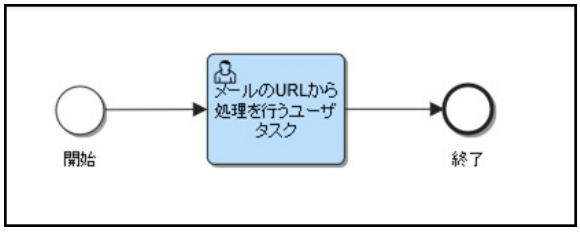
> **コラム**
>
> 各種設定・詳細については、以下のリンクを参照してください。
>
> ロール設定：「IM-共通マスタ 管理者操作ガイド」 - 「ユーザ」
> メールサーバの設定：「設定ファイルリファレンス」 - 「 メール設定|

> **コラム**
>
> 各種インポート方法については、以下のリンクを参照してください。
>
> ロジックフロー定義：「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」
> 「IM-FormaDesigner」で作成したアプリケーション：「IM-FormaDesigner 作成者操作ガイド」-「インポート・エクスポートを利用した IM-FormaDesigner のアプリケーションやデータソース定義の移行」

- プロセス定義を作成する
- 実行結果を確認する

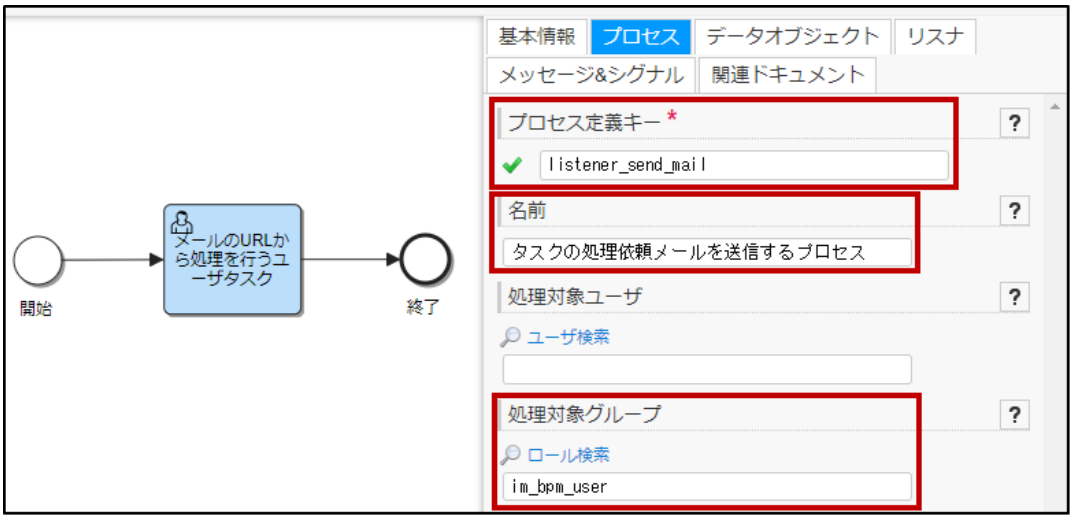
プロセス定義を作成する

タスクリスナを利用してタスクの処理依頼メールを送信するプロセスを作成します。
このチュートリアルでは、画面から選択されたタスク処理者に対して、タスク処理画面のURLをメール送信します。
タスク処理者を選択する際に、「IM-FormaDesigner」で作成したアプリケーションを入力フォーム画面として使用します。



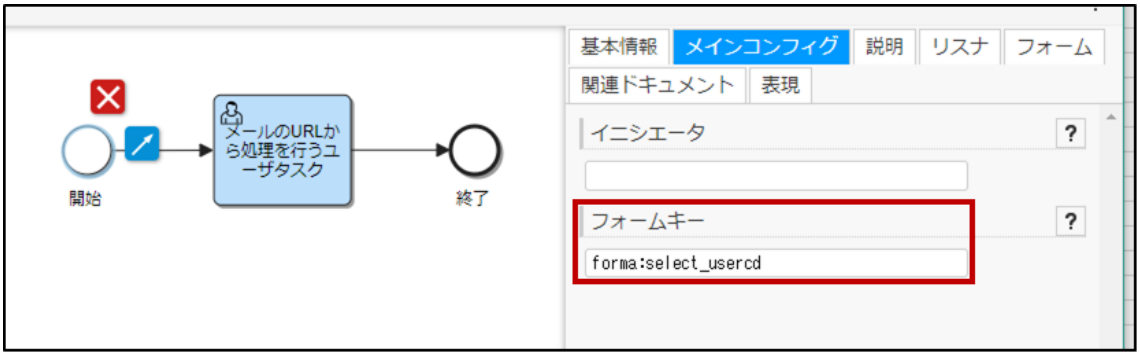
図：完成イメージ

1. 「開始イベント」を配置します。
2. 「プロセス」の「プロセス定義キー」と「名前」と「処理対象グループ」を設定します。
   - プロセス定義キー: listener_send_mail
   - 名前：タスクの処理依頼メールを送信するプロセス
   - 処理対象グループ：im_bpm_user



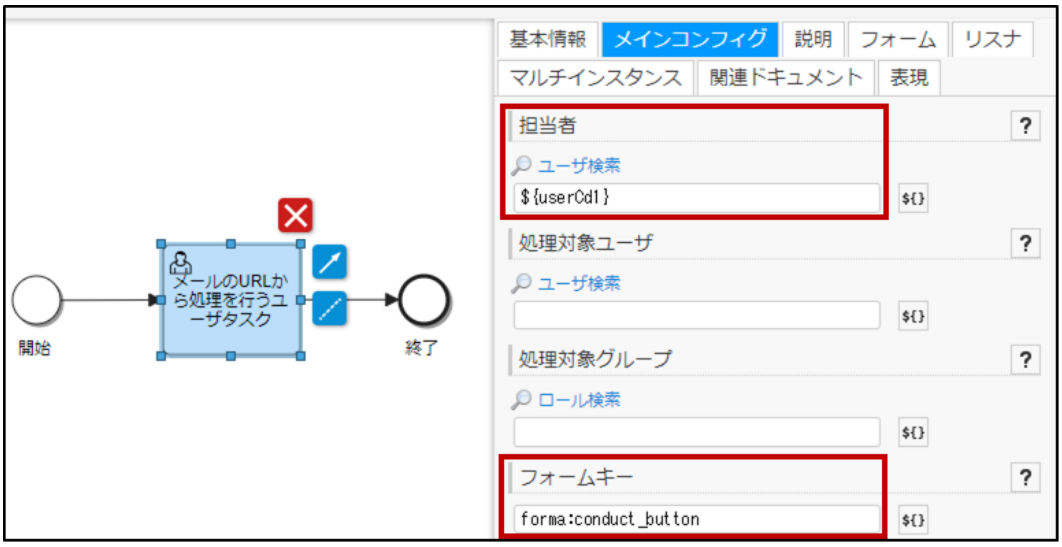
図：「プロパティ」 - 「プロセス」

3. 「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」の「フォームキー」を設定します。
   - フォームキー：forma:select_usercd



図：「開始ベント」 - 「プロパティ」 - 「メインコンフィグ」

4. 「ユーザタスク」を配置し、「メインコンフィグ」の「担当者」と「フォームキー」を設定します。
   - 担当者：${userCd1}
   - フォームキー：forma:conduct_button



図：「ユーザタスク」 -「プロパティ」 -「メインコンフィグ」

5. 「リスナ」 -「タスクリスナ」 の 「追加」をクリックします。

図：「ユーザタスク」-「プロパティ」 -「リスナ」

6. 「タスクリスナ」を以下のように設定します。
   - イベント：create
   - タイプ：IM-LogicDesigner
   - フローID：listener_send_mail
   - 最新バージョン：最新バージョンを利用

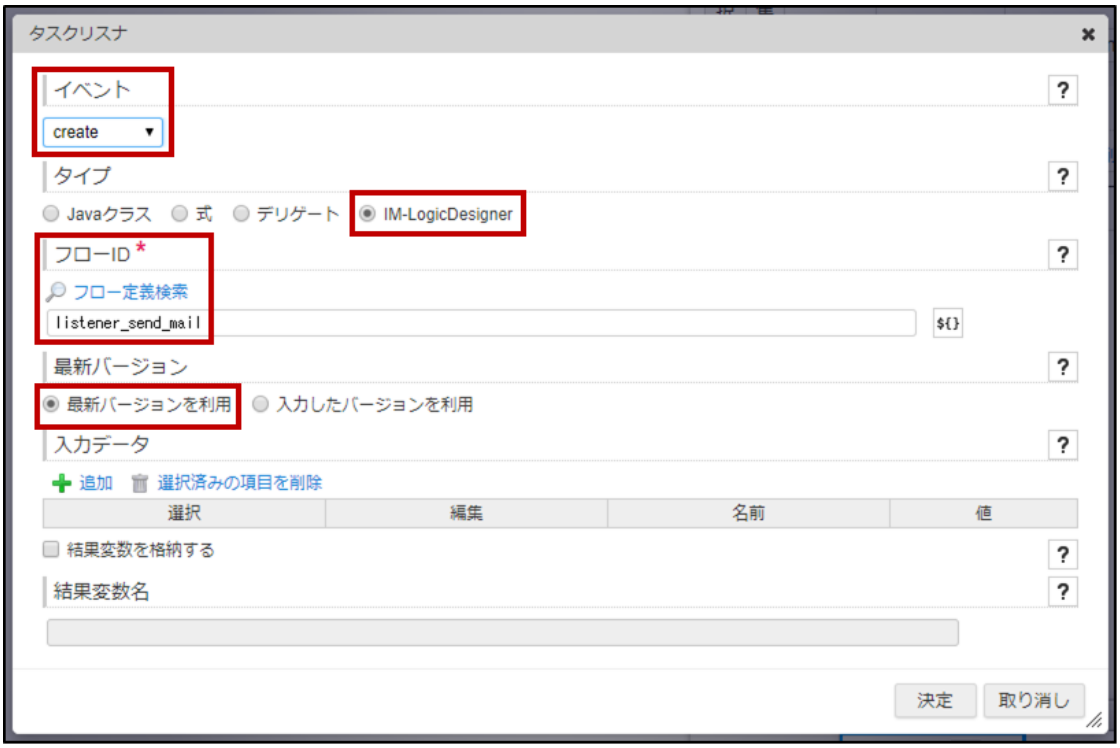


図 ：「タスクリスナ」

コラム

本来ロジックフローと変数を連携するには、「IM-LogicDesigner」リスナのプロパティ「入力データ」を設定する必要があります。
今回は、ロジックフローの「入出力設定」で、入力データに対し「暗黙オブジェクト」を設定してあるため、必要ありません。

「暗黙オブジェクト」についての詳細は「IM-BPM 仕様書」-「IM-LogicDesignerリスナ」を参照してください。

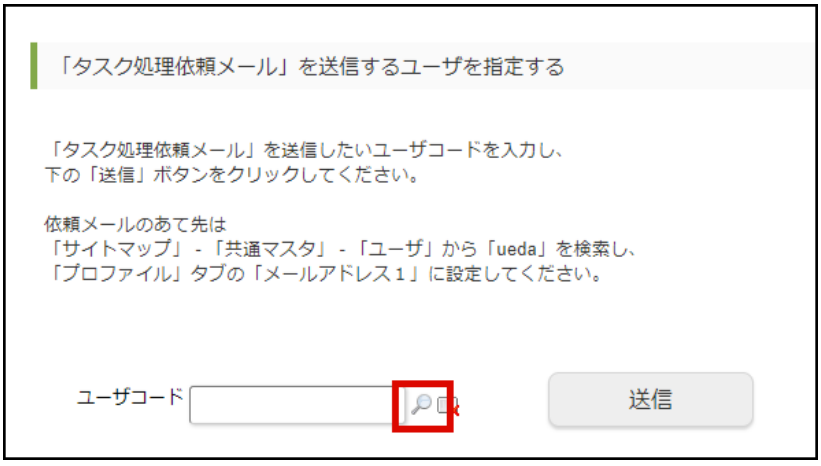
実行結果を確認する

このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行した結果の確認を行います。

1. 「プロセス開始一覧」 から、「タスクの処理依頼メールを送信するプロセス」を開始します。

図：「プロセス開始一覧」

2. 「IM-FormaDesigner」で作成したアプリケーション、「select_usercd」に遷移します。
虫眼鏡のアイコンをクリックし、検索画面を開いてください。



図：アプリケーション画面「select_usercd」

3. 「上田辰男」を選択します。
「決定」ボタンをクリックすると検索画面が閉じるので、「select_usercd」の「送信」ボタンをクリックします。

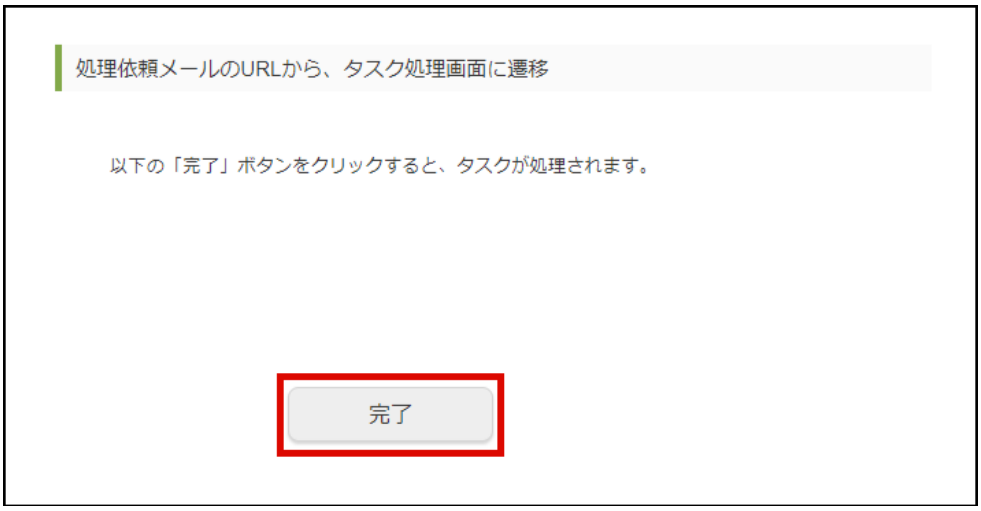
図：「ユーザ検索」

4. 「select_usercd」画面で設定した「上田辰男」に届いたメールを確認します。
メールに記載されたURLをクリックすると、「IM-FormaDesigner」で作成したアプリケーション画面「conduct_button」が開きます。

図：処理依頼先の「上田辰男」に届いたメール

> **注意**
>
> 依頼URLへアクセスした際、既に「青柳辰巳」（ユーザコード：aoyagi）でログインしている場合は実行権限がないためエラー画面へ遷移します。
> ログイン画面に戻り、作業依頼先の「上田辰男」でログインして、再度依頼URLへアクセスしてください。

5. 「完了」ボタンをクリックすることにより、タスクが完了します。



図：アプリケーション画面「conduct_button」

タスクリスナを利用して特定の組織に所属するユーザを処理対象ユーザに設定する

このチュートリアルでは「タスクリスナ」を利用して、ユーザタスクの処理対象ユーザに特定の組織に所属するユーザを設定します。
「タスクリスナ」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「タスクリスナ」もあわせて参照してください。

コラム

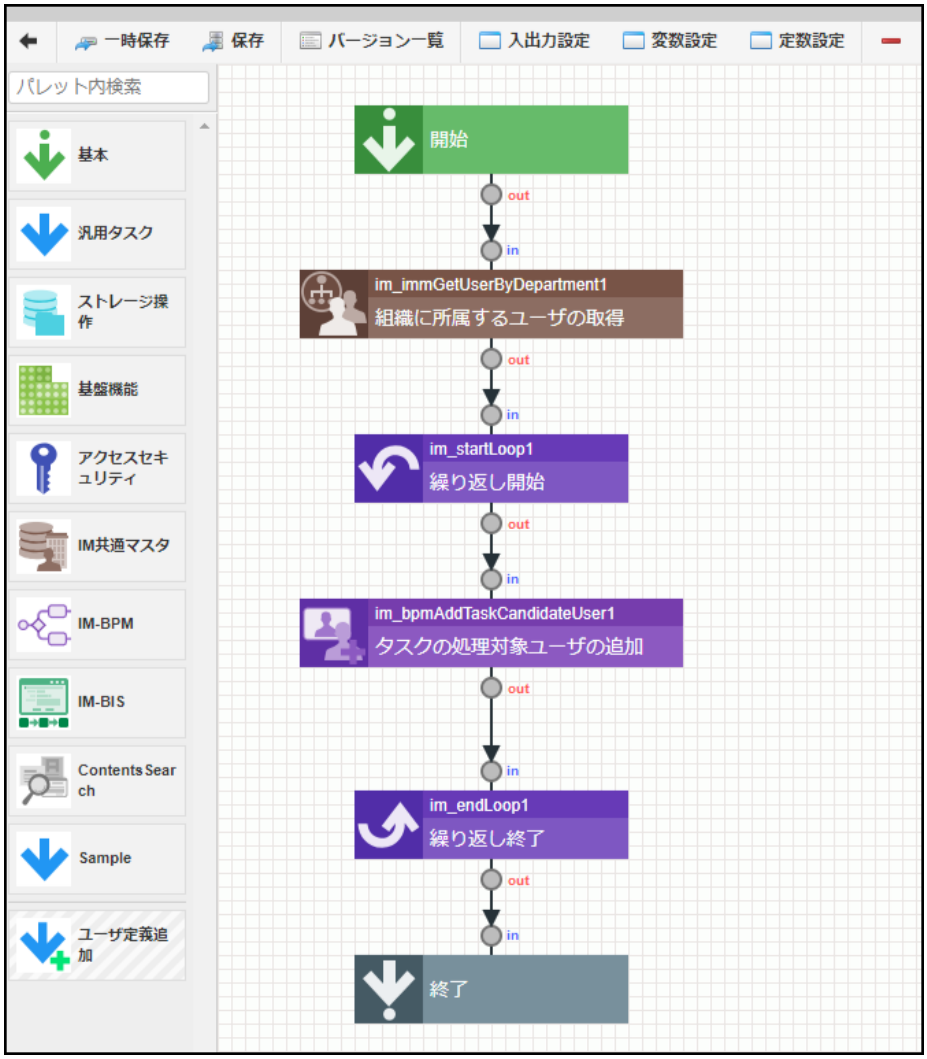
このチュートリアルで作成する「プロセス定義」と「ロジックフロー定義」のサンプルを以下のリンクからダウンロードできます。
listener_add_task_candidate_user.bpmn
im_logicdesigner-data-listener_add_task_candidate_user.zip

これらのサンプルは、各アプリケーションの機能でアップロードまたはインポートして利用可能です。
詳細な手順については、以下のリンク先を参照してください。
プロセス定義：「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」
ロジックフロー定義：「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」

- ロジックフローを作成する
- プロセス定義を作成する
- 実行結果を確認する

ロジックフローを作成する

特定の組織に所属するユーザを、処理対象ユーザに設定するロジックフローを作成します。
会社コード、組織セットコードおよび組織コードを使用して「組織に所属するユーザの取得」タスクから、組織に所属するユーザの一覧を取得します。
特定の組織に所属するユーザをユーザタスクの処理対象ユーザに設定するロジックフローを作成します。



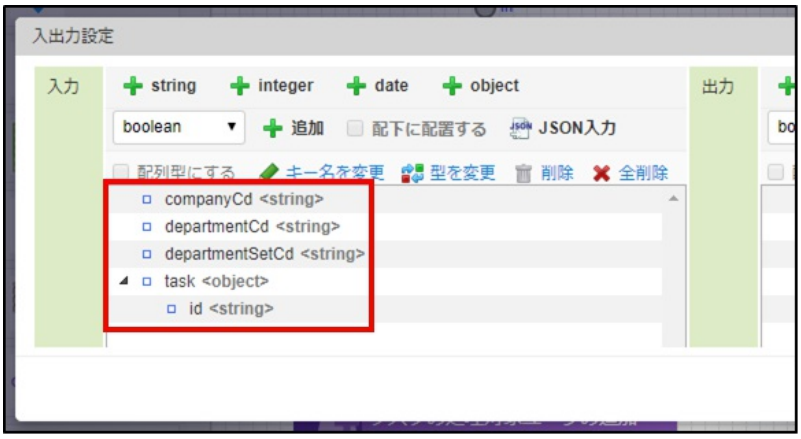
図：完成イメージ

1. ツールバーの「入出力設定」をクリックします。

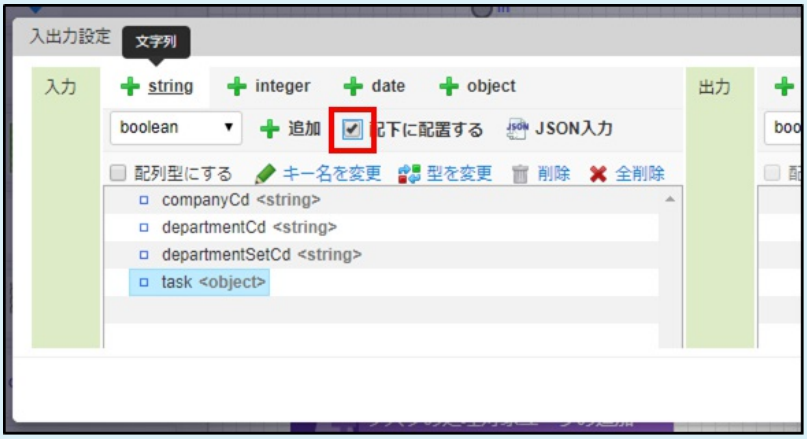
図：「ツールバー」 - 「入出力設定」

2. 入力値を以下のように設定します。

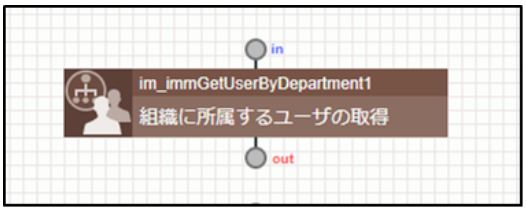
キー名	型
companyCd	<string>
departmentSetCd	<string>
departmentCd	<string>
task	<object>
task - id	<string>



図：「入出力設定」 - 「入力」

> **コラム**
>
> パラメータに子要素を追加する場合、親要素を選択した状態でメニューの「配下に設置する」にチェックを入れ、追加したい各データ型を選択してください。
>
> 
>
>
> 図：「入出力設定」 - 「入力」

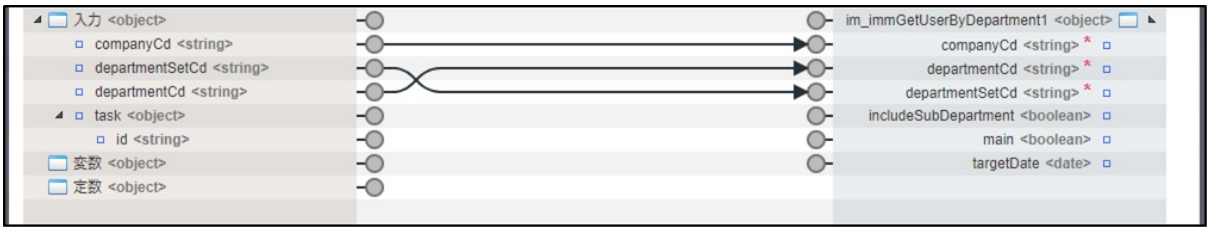
3. パレットの種別「IM共通マスタ」から、「組織に所属するユーザの取得」を設置します。



図：「組織に所属するユーザの取得」

4. 「組織に所属するユーザの取得」のマッピングを、以下のとおりに設定します。

入力（始点）	出力（終点）
入力<Object> - companyCd<string>	im_immGetUserByDepartment1<object> - companyCd<string>
入力<Object> - departmentSetCd<string>	im_immGetUserByDepartment1<object> - departmentSetCd<string>
入力<Object> - departmentCd<string>	im_immGetUserByDepartment1<object> - departmentCd<string>

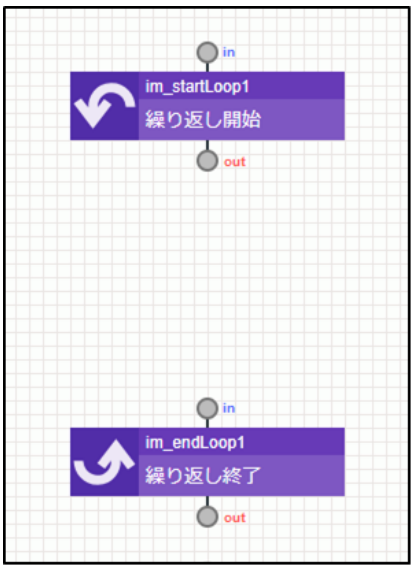


図：「マッピング設定」 - 「組織に所属するユーザの取得」

コラム

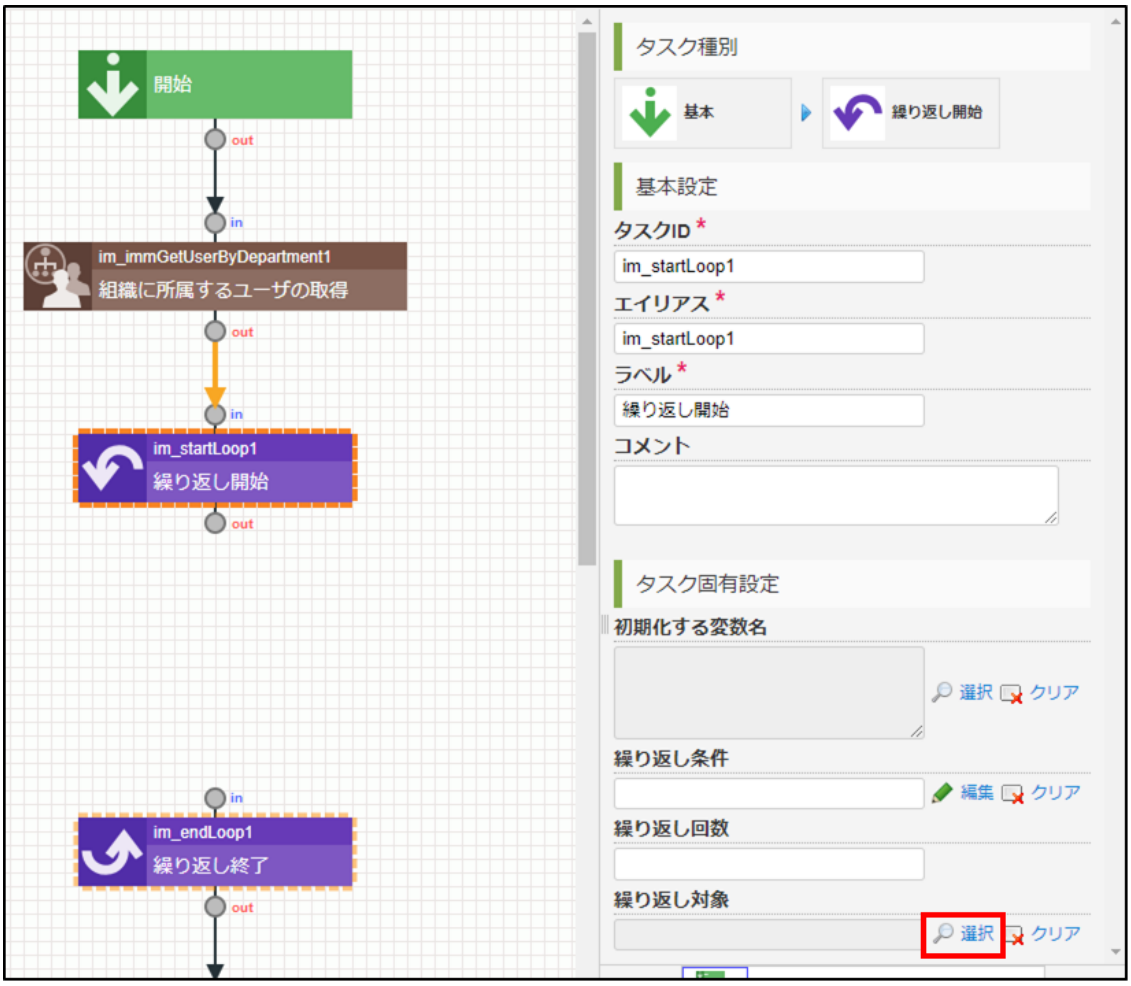
「組織に所属するユーザの取得」に関するタスクの詳細は「IM-LogicDesigner仕様書」-「組織に所属するユーザの取得」を参照してください。

5. パレットの種別「基本」から、「繰り返し」を設置します。
「繰り返し開始」を選択すると、自動で「繰り返し終了」も配置されます。



図：「繰り返し開始」と「繰り返し終了」

6. 「繰り返し開始」を選択した状態で、「プロパティ」 - 「タスク固有設定」 - 「繰り返し対象」の選択をクリックします。

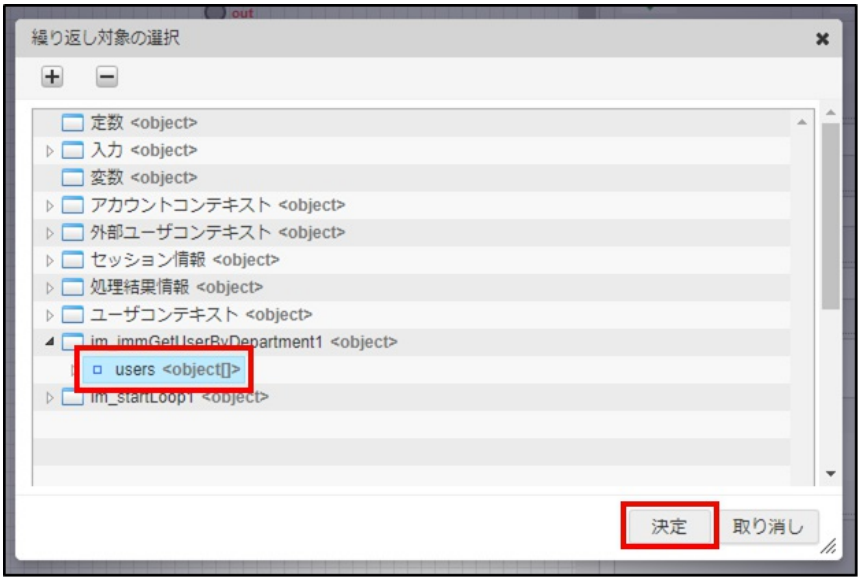


図：「プロパティ」 - 「タスク固有設定」 - 「繰り返し対象」

コラム

「繰り返し」に関する制御要素の詳細は「IM-LogicDesigner チュートリアルガイド」-「繰り返し処理を利用したフロー」を参照してください。

7. 「繰り返し対象の選択」から、`im_immGetUserByDepartment1<object>` - `users<object[]>` を選択し、決定をクリックします。



図：「タスク固有設定」 - 「繰り返し対象」

8. パレットの種別「IM-BPM」から「タスク処理対象のユーザの追加」を選択し、上記で配置した「繰り返し開始」と「繰り返し終了」の間に配置します。

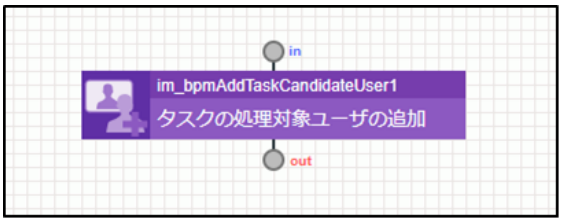


図 ：「タスク処理対象のユーザの追加」

9. 「タスク処理対象のユーザの追加」のマッピングを、以下のように設定します。

入力（始点）	出力（終点）
入力<Object> - task<Object> - id <string>	im_bpmAddTaskCandidateUser1<object> - taskId <string>
im_startLoop1<object> - item<object> - userCd<string>	im_bpmAddTaskCandidateUser1<object> - userIds<string[]>

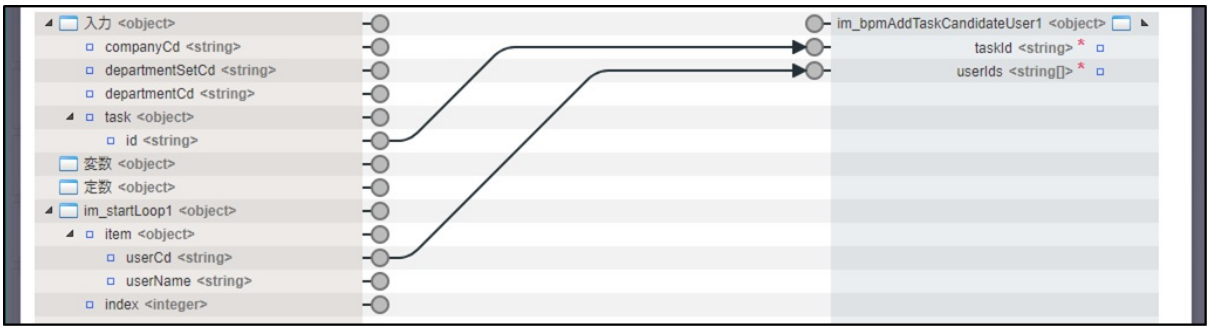
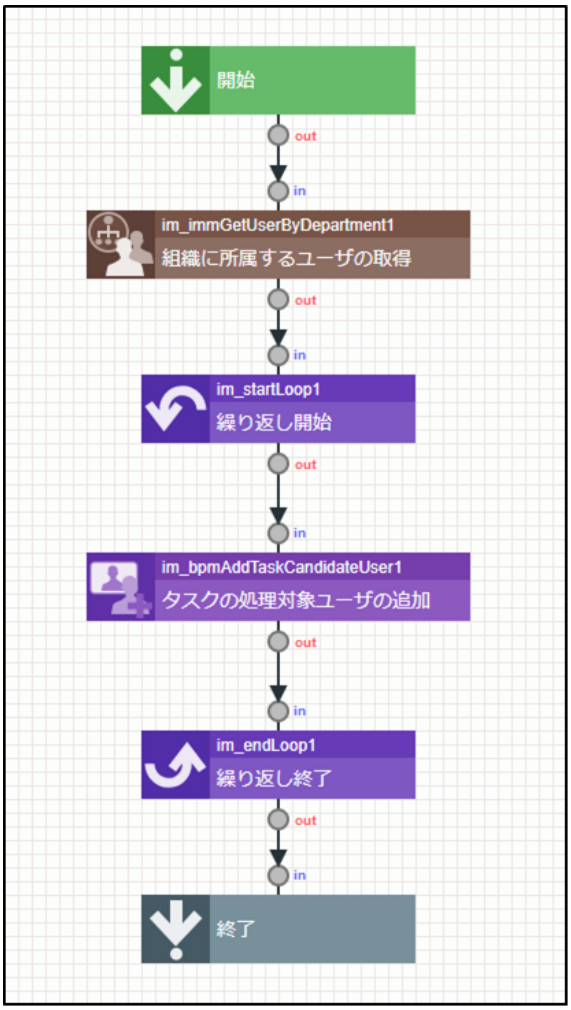


図 ：「タスク処理対象のユーザの追加」 - 「マッピング設定」

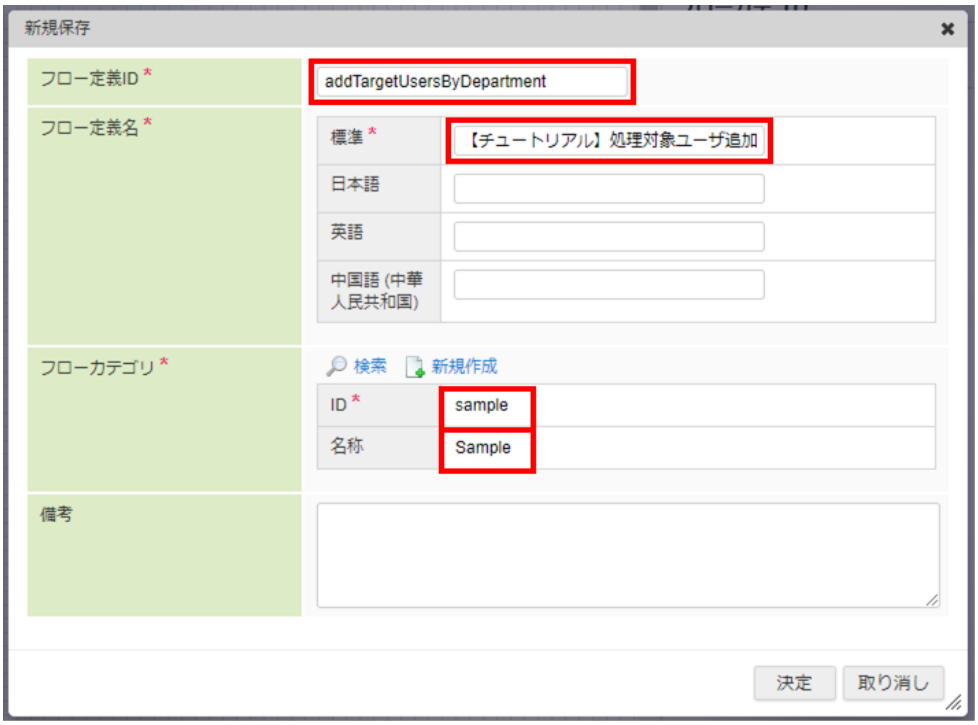
> **コラム**
>
> 「マッピング設定」に関するタスクの詳細は「IM-LogicDesigner仕様書」 - 「タスクの処理対象ユーザの追加」を参照してください。

10. シーケンスでつなぎます。



図：「ロジックフロー定義編集画面」

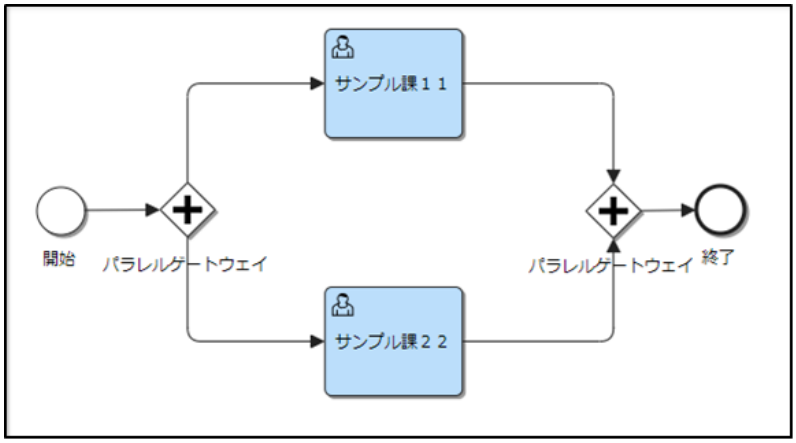
11. 「ロジックフロー定義編集」画面上部、ヘッダ内の「新規保存」をクリックします。
以下のように設定し、ロジックフロー定義を新規保存します。
    - フロー定義ID：addTargetUsersByDepartment
    - フロー定義名：【チュートリアル】処理対象ユーザ追加
    - フローカテゴリ：
        - ID：sample
        - 名称：Sample



図：「新規保存」

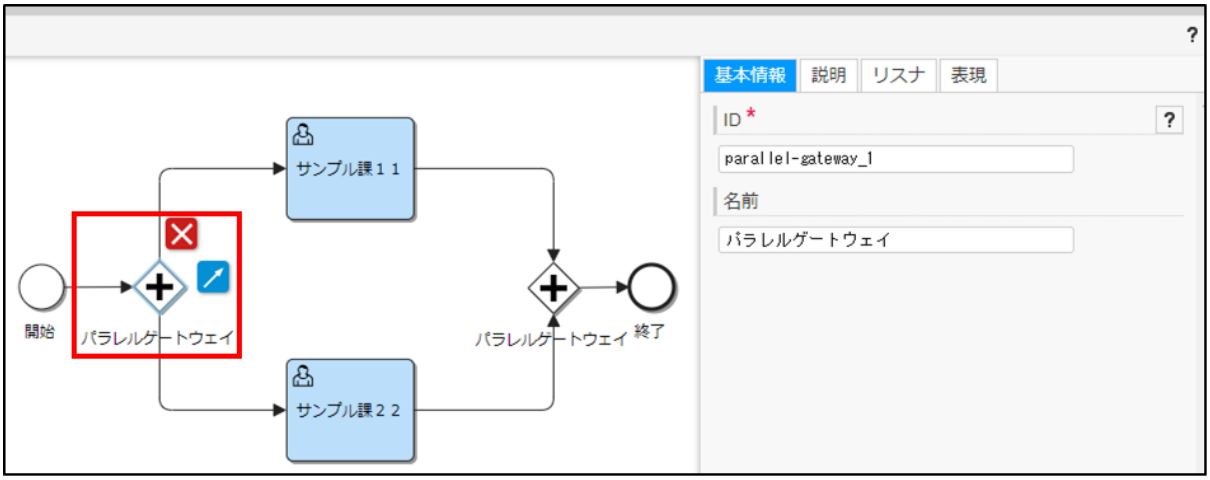
プロセス定義を作成する

上記で作成したロジックフローを呼び出すことで、 特定の組織に所属するユーザを処理対象ユーザに設定するプロセスを作成します。
ユーザタスクの「サンプル課１１」と「サンプル課２２」の2つを配置します。
「サンプル課１１」のユーザタスクには「サンプル課１１」の組織に所属するユーザを処理対象ユーザに設定します。
「サンプル課２２」のユーザタスクには「サンプル課２２」の組織に所属するユーザを処理対象ユーザに設定します。



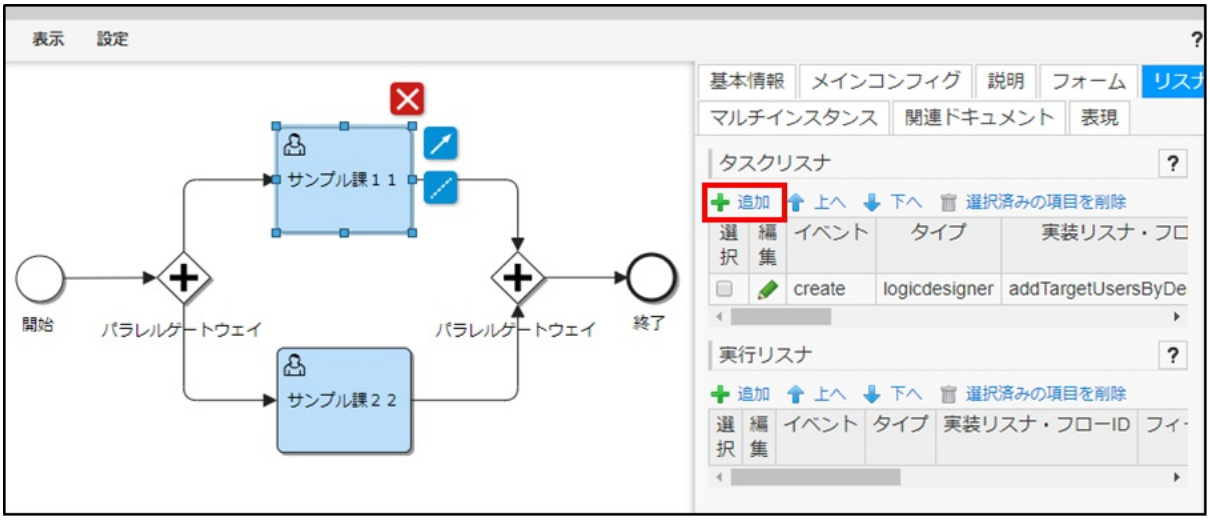
図：完成イメージ

1. 「開始イベント」を配置します。
2. 複数のユーザタスクに分岐するため、「パラレルゲートウェイ」を設置します。



図：「パラレルゲートウェイ」

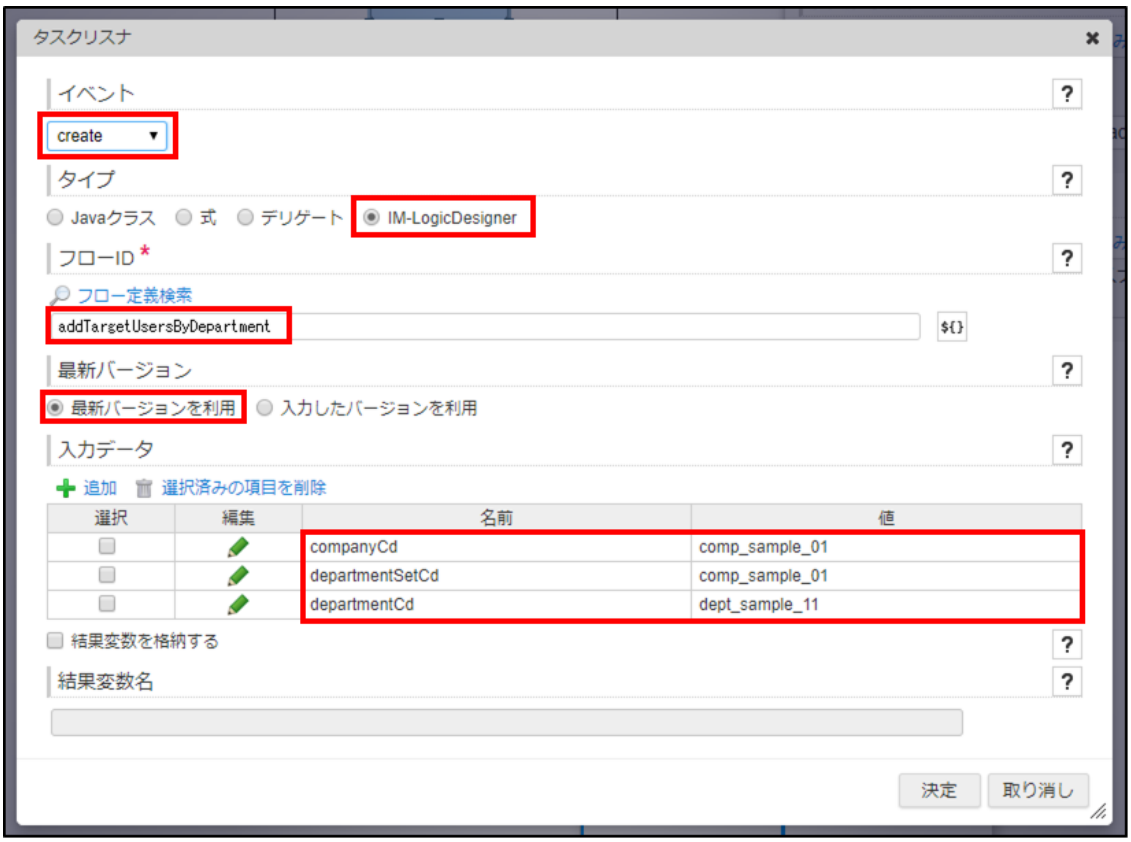
3. 「ユーザタスク」を配置し、「リスナ」 - 「タスクリスナ」 の 「追加」をクリックします。



図：「ユーザタスク」 - 「プロパティ」 - 「リスナ」

4. 「サンプル課１１」に所属しているユーザを処理対象ユーザに設定するため、「タスクリスナ」を以下のように設定します。
   - イベント：create
   - タイプ：IM-LogicDesigner
   - フローID：addTargetUsersByDepartment

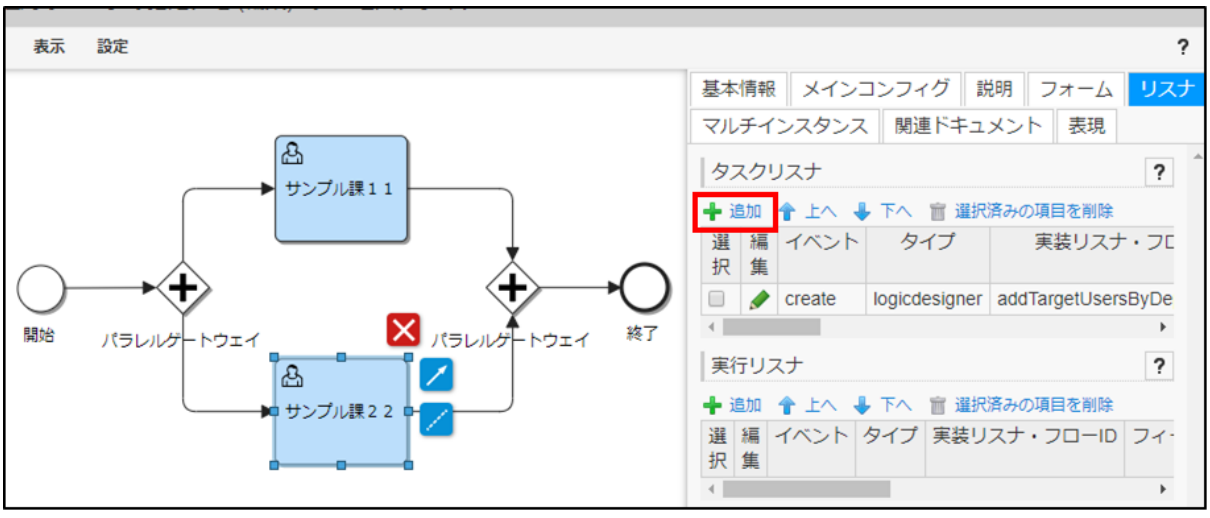
- 最新バージョン：最新バージョンを利用
- 入力データ1（会社コード）：
  - 名前：companyCd
  - 値：comp_sample_01
- 入力データ2（組織セットコード）：
  - 名前：departmentSetCd
  - 値：comp_sample_01
- 入力データ3（組織コード）：
  - 名前：departmentCd
  - 値：dept_sample_11



図：「ユーザタスク」 - 「プロパティ」 - 「リスナ」

> **コラム**
>
> ロジックフローと変数を連携するには、「IM-LogicDesigner」リスナのプロパティ「入力データ」で設定する必要があります。
> task<object>とid<string>は「暗黙オブジェクト」のため、設定は行いません。
> 「暗黙オブジェクト」についての詳細は「IM-BPM 仕様書」-「IM-LogicDesignerリスナ」を参照してください。

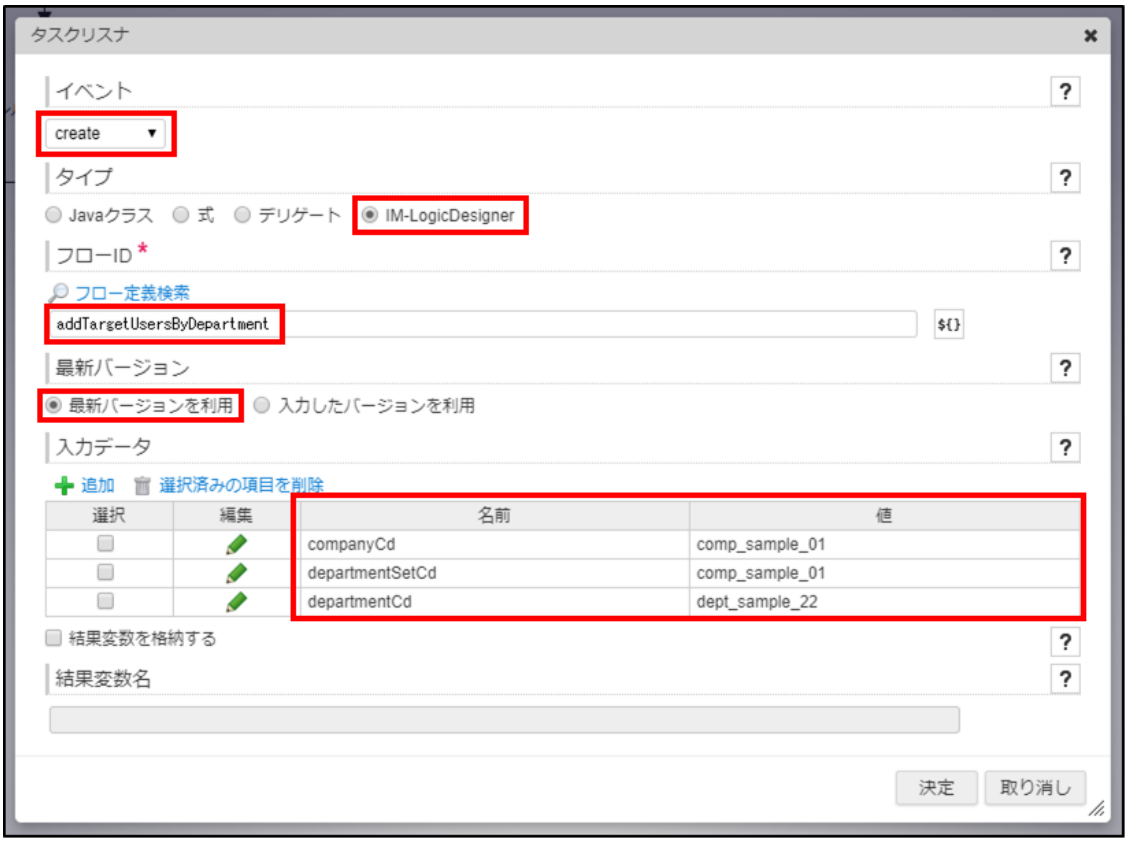
5. 「ユーザタスク」を配置し、「リスナ」 - 「タスクリスナ」 の 「追加」をクリックします。



図：「ユーザタスク」 - 「プロパティ」 - 「リスナ」

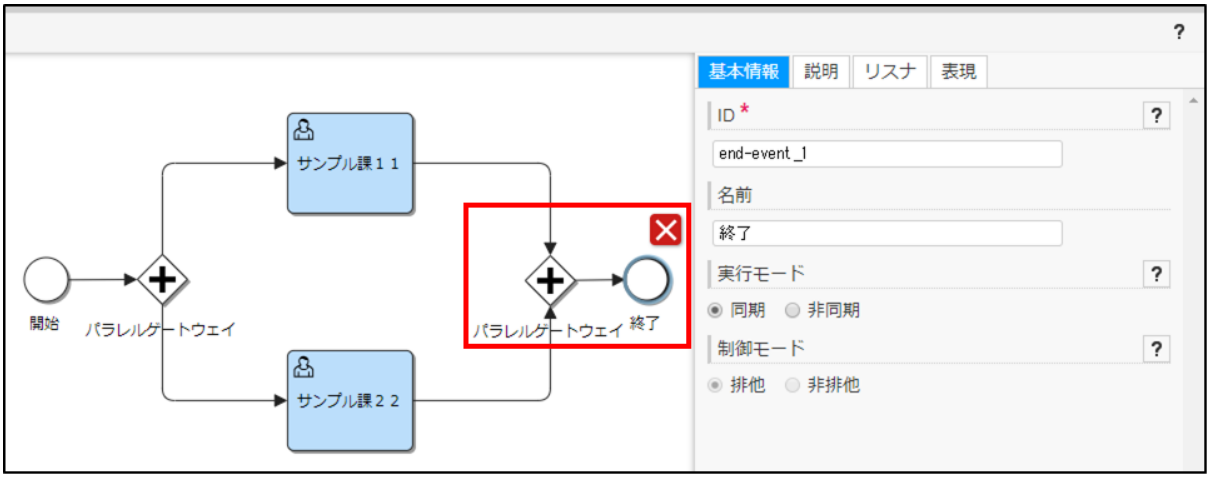
6. 「サンプル課２２」に所属しているユーザを処理対象ユーザに設定するため、「タスクリスナ」を以下のように設定します。

- プロセス定義キー: create
- タイプ : IM-LogicDesigner
- フローID : addTargetUsersByDepartment
- 最新バージョン : 最新バージョンを利用
- 入力データ1（会社コード）:
- 名前 : *companyCd*
  - 値 : comp_sample_01
- 入力データ2（組織セットコード）:
  - 名前 : departmentSetCd
  - 値 : comp_sample_01
- 入力データ3（組織コード）:
  - 名前 : departmentCd
  - 値 : dept_sample_22



図：「タスクリスナ」

7. 「パラレルゲートウェイ」と「終了イベント」を設置します。



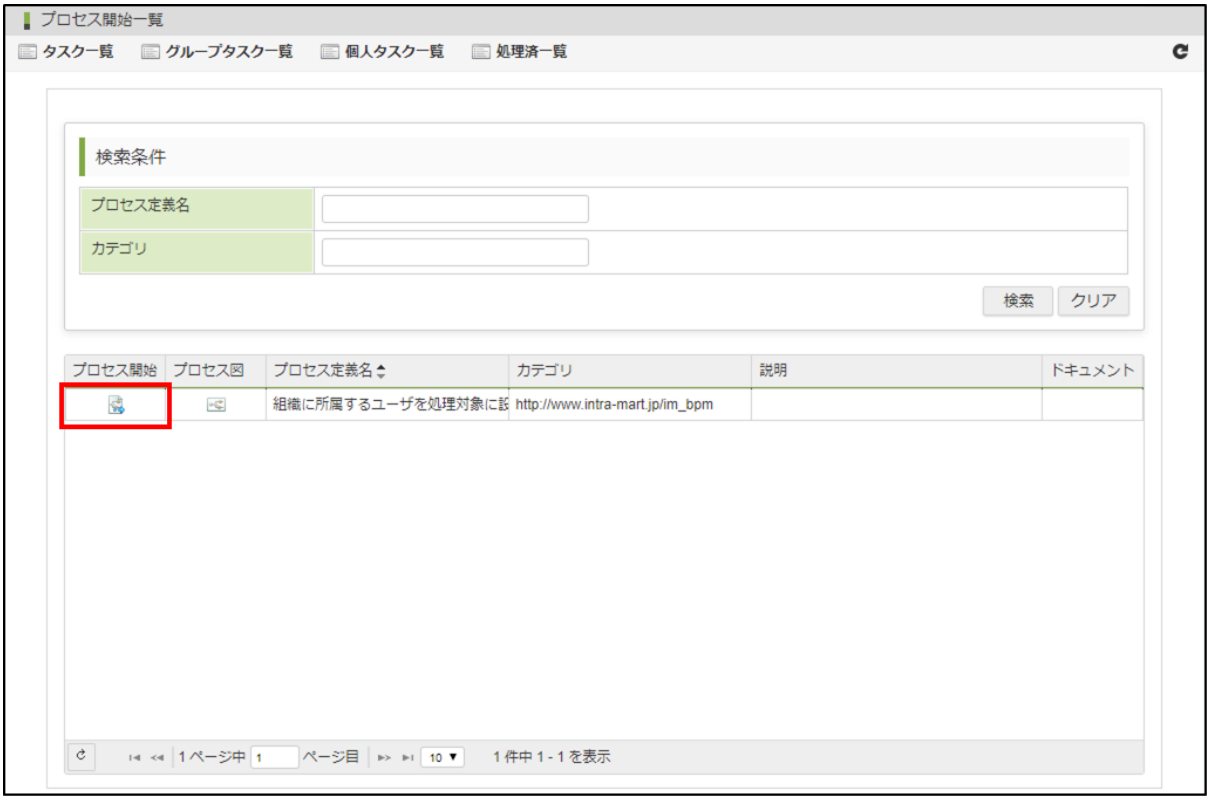
図：「パラレルゲートウェイ」と「終了イベント」

8. 名前を付けて保存します。

実行結果を確認する

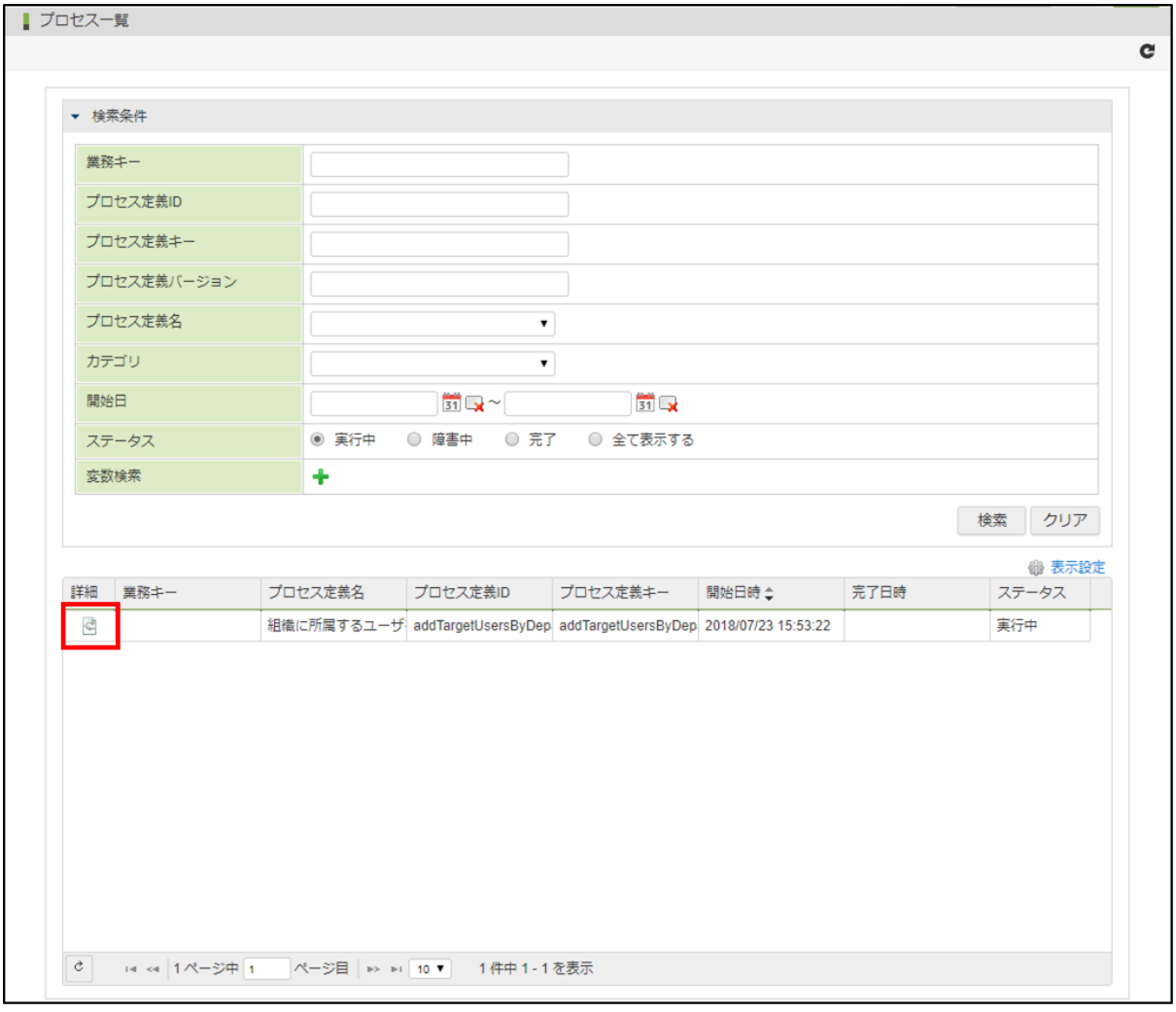
このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行した結果の確認を行います。

1. 「サイトマップ」→「BPM」→「プロセス開始一覧」を開きます。
   プロセス開始をクリックし、プロセスを開始します。



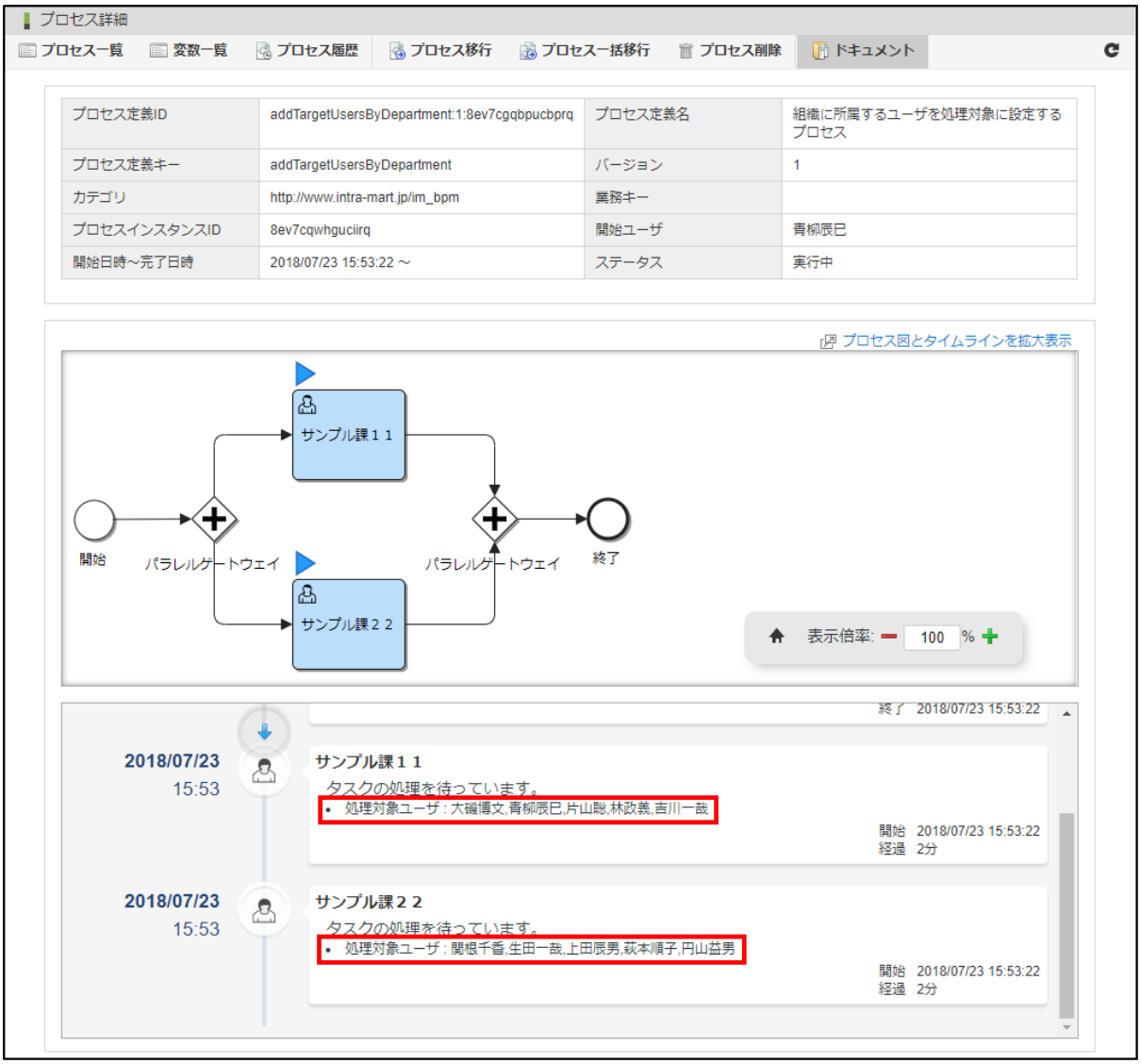
図：「プロセス開始一覧」

2. 「プロセス一覧」から、「プロセス詳細」を開きます。



図：「プロセス一覧」

3. 「プロセス詳細」から、ユーザタスクにそれぞれの組織の処理対象ユーザが設定されていることを確認します。



図：「プロセス一覧」 -「プロセス詳細」

4. テナント管理画面を開くため、テナント管理者でログインします。
サイトマップから「共通マスタ」 -「マスタメンテナンス」-「組織」をクリックします。
所属しているユーザが、上記で確認した「プロセス詳細」のタスク「サンプル課１１」の処理対象ユーザであることを確認します。

図：「共通マスタ」 -「組織」

5. さらに、「サンプル課２２」をクリックします。
所属しているユーザが、上記で確認した「プロセス詳細」のタスク「サンプル課２２」の処理対象ユーザであることを確認します。

図：「共通マスタ」 - 「組織」

関連ドキュメント

IM-Wikiを関連ドキュメントとして設定する

このチュートリアルでは、プロセスやタスクに対し、「IM-Wiki」を「関連ドキュメント」として設定する方法を解説します。
「IM-Wiki」は、「intra-mart Accel Platform」上でWikiページを作成、編集、管理ができる機能です。

プロセスを進めていく中で、「IM-FormaDesigner」を使用します。
チュートリアルを開始する前に「IM-FormaDesigner」で作成したアプリケーションをインポートしてください。
forma_designer-relation_document_wiki-select_user.zip

また、「IM-Knowledge」を閲覧するには、「Knowledge コンテンツ利用者」ロールが必要です。
「サイトマップ」→「共通マスタ」→「ユーザ」から「青柳辰巳（ユーザコード：aoyagi）」を検索し、以下の設定を行ってください。

- 「ロール」タブ - 「Knowledge コンテンツ利用者」ロールの追加

> **コラム**
>
> このチュートリアルで作成する資材のサンプルを以下のリンクからダウンロードできます。
> 「プロセス定義」：**relation_document_wiki-set_relation_document.bpmn**
> 「IM-Wiki」：**im_knowledge-relation_document_wiki.zip**
> 「IM-Knowledgeナレッジグループの認可（ポリシー）」：**authz_policy-relation_document_wiki.xml**
> IM-Knowledgeナレッジグループの認可（ポリシー）をインポートする際は、ジョブネットパラメータの file を authz_policy-relation_document_wiki.xml に変更してください。
>
> これらのサンプルは、各アプリケーションの機能でアップロード、または、インポートして利用可能です。
> 詳細な手順については、以下のリンク先を参照してください。
> プロセス定義：「IM-BPM プロセスデザイナ 操作ガイド」 - 「プロセス定義のアップロード」
> IM-Wiki：「IM-Knowledge管理者操作ガイド」 - 「インポート」
> IM-Knowledge ナレッジグループの認可（ポリシー）：「IM-Authz（認可）インポート・エクスポート仕様書」 - 「IM-Knowledge ナレッジグループの認可（ポリシー）」、「ジョブ インポート・エクスポート仕様書」 - 「インポート実行オプション」

> **コラム**
>
> 「IM-FormaDesigner」で作成したアプリケーションのインポート方法は以下を参照してください。
>
> 「IM-FormaDesigner」で作成したアプリケーション：「IM-FormaDesigner 作成者操作ガイド」-「インポート・エクスポートを利用した IM-FormaDesigner のアプリケーションやデータソース定義の移行」

- IM-Wikiで業務マニュアルを作成する
- プロセス定義を作成する
- 実行結果を確認する

IM-Wikiで業務マニュアルを作成する

関連ドキュメントとして紐づけるWikiを「IM-Knowledge」で作成します。
「IM-Knowledge」のグループを作成し、そのグループにWikiを作成します。
「プロセス開始時に参照するWiki」と「タスク処理時に参照するWiki」を作成します。

注意

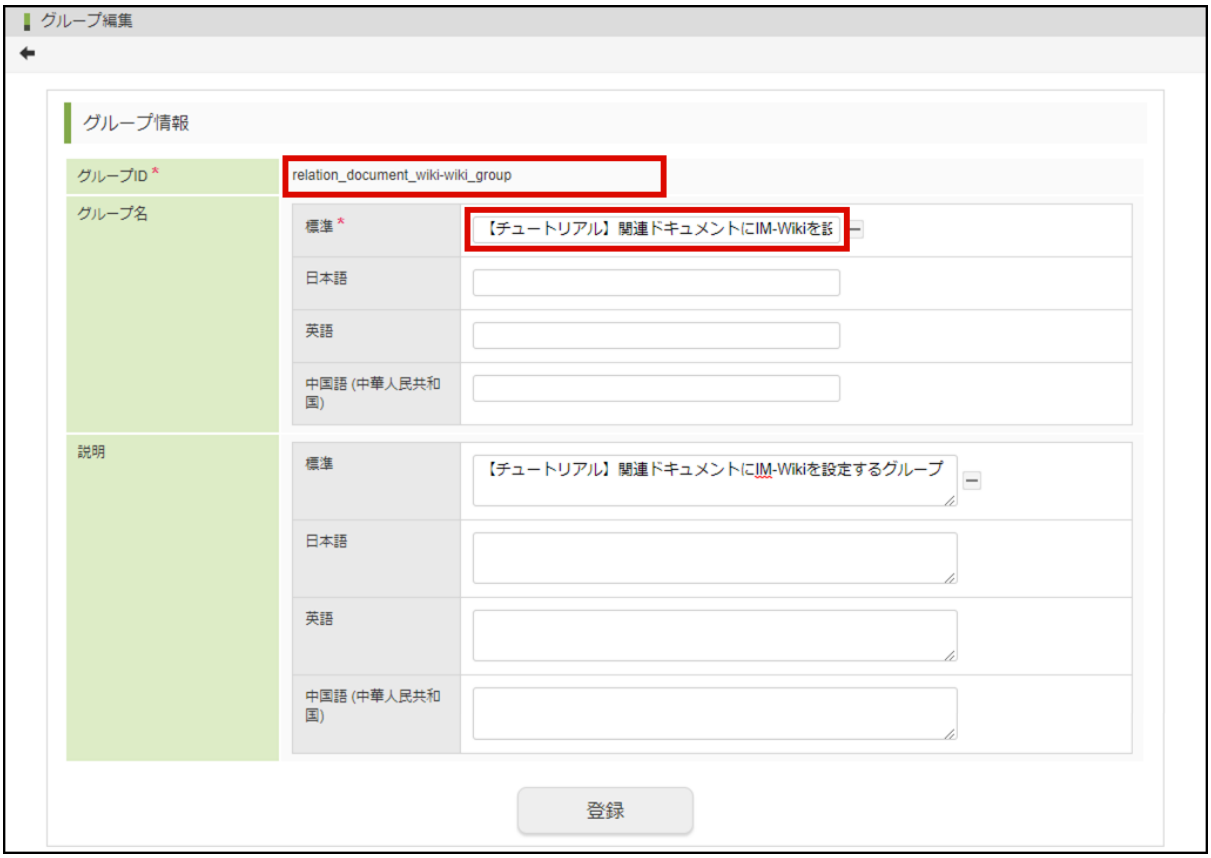
IM-Knowledgeグループを作成するためには、以下のいずれかに当てはまるユーザでアクセスする必要があります。

- IM-Knowledgeグループ管理者の認可と、「認可設定（ポップアップ）」の認可を持つユーザ
- 「テナント管理者」のロールを持つユーザ

1. 「IM-Knowledge」のグループを作成します。
   「テナント管理者（ユーザコード：tenant）」でログインします。
2. 「サイトマップ」→「Knowledge」→「管理」→「グループ一覧」画面を表示します。
3. 「グループ一覧」の「新規登録」をクリックし、「グループ登録」画面を表示します。

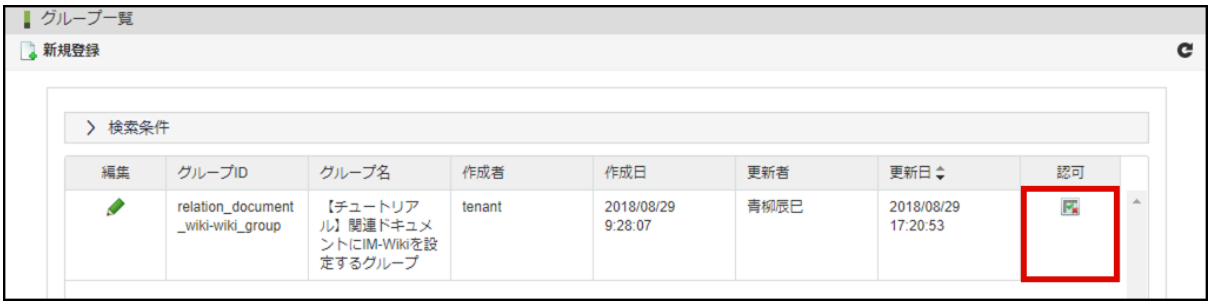
図：「グループ一覧」

4. 「グループ登録」画面で、項目を以下のように設定します。
   - グループID：relation_document_wiki-wiki_group
   - グループ名（標準）：任意



図：「グループ登録」

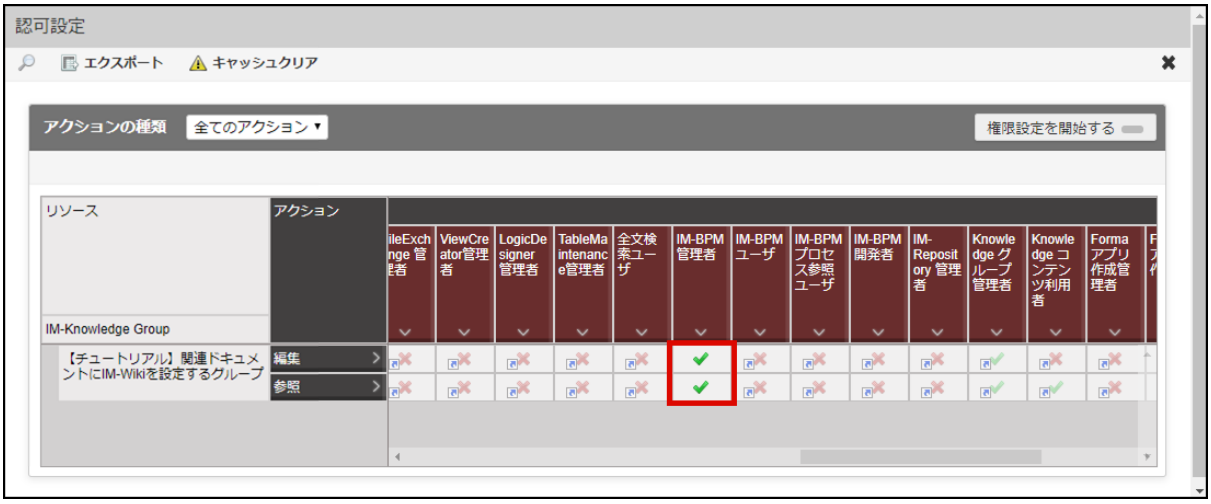
5. 入力後、「登録」をクリックします。
6. グループの「参照・編集」の権限を設定します。
   「グループ一覧画面」で「  」アイコンをクリックし、「認可設定」ダイアログを表示します。



図：「グループ一覧」

7. 画面の右上の「権限設定を開始する」ボタンをクリックします。

8. 「IM-BPM管理者」の「  」をクリックし、「  」に切り替えます。



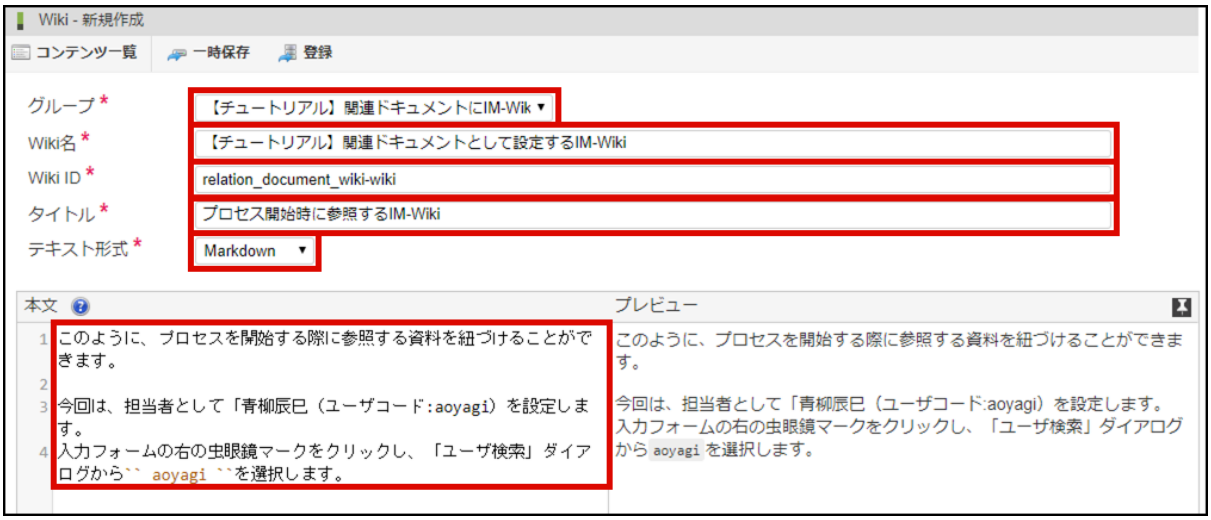
図：「認可設定」 ダイアログ

9. 上記で作成したグループの「コンテンツ」にWikiを作成します。
「IM-BPM 管理者（im_bpm_manager）」のロールが付与されている aoyagi でログインします。

10. 「サイトマップ」→「Knowledge」→「コンテンツ」→「Wiki新規作成」をクリックし、「Wiki新規作成」画面を表示します。

11. 「プロセス開始一覧」画面で、プロセスを開始する際に参照できるWikiを作成します。
項目を以下のように設定します。
    - グループ :上記で作成したグループを選択
    - Wiki名：任意
    - Wiki ID : relation_document_wiki-wiki
    - タイトル：プロセス開始時に参照するIM-Wiki
    - テキスト形式：任意
    - 本文：以下の文章をコピー&ペーストします。

> このように、プロセスを開始する際に参照する資料を紐づけることが可能です。
> 今回は、担当者として「青柳辰巳（ユーザコード:aoyagi）を設定します。
> 入力フォームの右の虫眼鏡マークをクリックし、「ユーザ検索」ダイアログから``aoyagi``を選択します。



図：「IM-Wiki」 - 「新規作成」

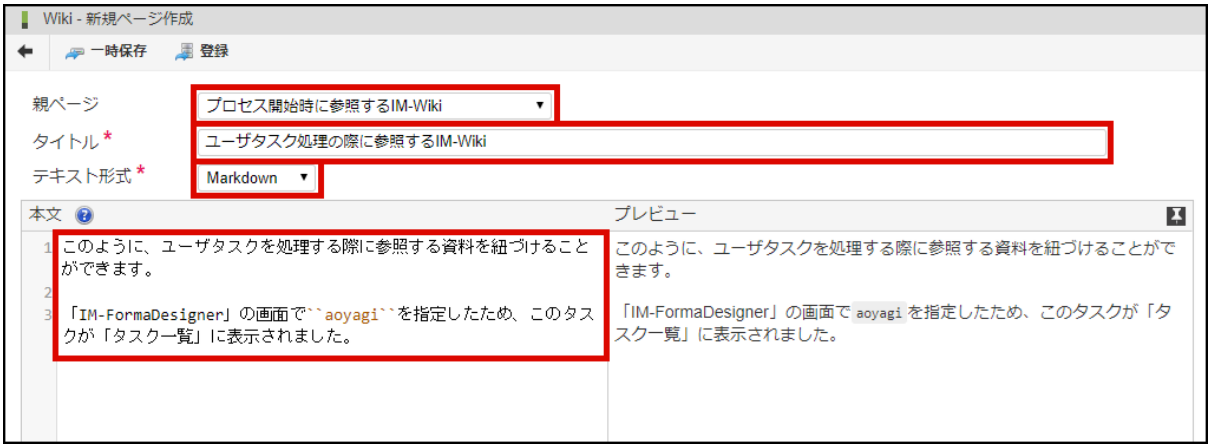
12. 入力後、「登録」をクリックします。

13. 「タスク一覧」画面で、タスクを処理する際に参照できるWikiを作成します。
「新規ページ作成」をクリックします。

図：「IM-Wiki」 - 「【チュートリアル】関連ドキュメントとして設定するIM-Wiki」

14. 項目を以下のように設定します。
    - 親ページ：上記で作成したページ
    - タイトル：ユーザタスク処理の際に参照するIM-Wiki
    - テキスト形式：任意
    - 本文：以下の文章をコピー&ペーストします。

```
このように、ユーザタスクを処理する際に参照する資料を紐づけることが可能です。
「IM-FormaDesigner」の画面で``aoyagi``を指定したため、このタスクが「タスク一覧」に表示されました。
```

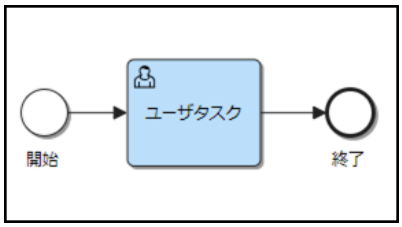


図：「IM-Wiki」 - 「新規ページ作成」

15. 入力後、「登録」をクリックします。

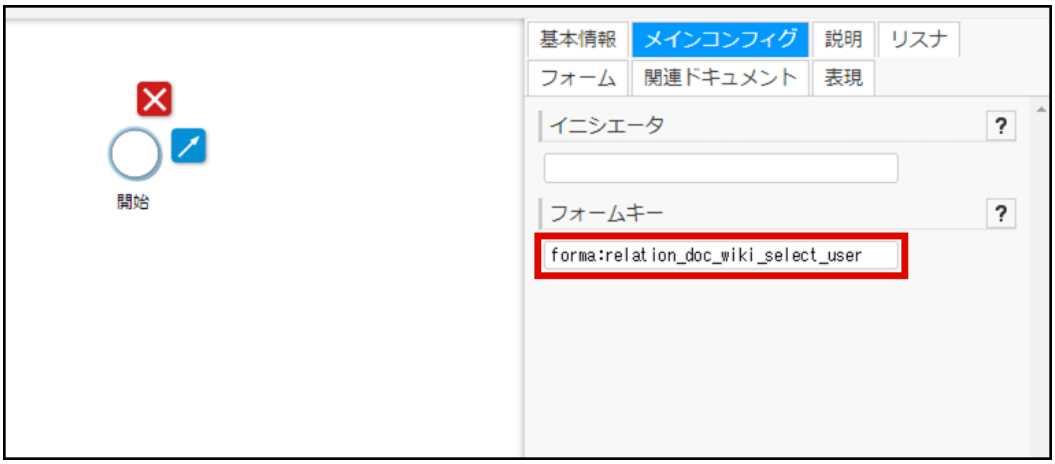
プロセス定義を作成する

このプロセスは、「関連ドキュメント」にWikiを設定することで、プロセス開始・タスク処理の際に参照できるようにします。
プロセスを開始する際に、インポートした「IM-FormaDesigner」のアプリケーションを使用します。



図：完成イメージ

1. 「開始イベント」を配置します。
2. 「開始イベント」の「プロパティ」 - 「 メインコンフィグ」で項目を以下のように設定します。
   - フォームキー : forma:relation_doc_wiki_select_user



図：「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

3. プロセス全体に対する設定を行います。
   キャンバスの余白をクリックし、「プロパティ」をプロセス全体に切り替えます。
4. 「プロセス」で項目を以下のように設定します。
   - 処理対象グループ : im_bpm_manager

図：「プロパティ」 - 「プロセス」

5. 「関連ドキュメント」の設定を行います。
   「プロパティ」 - 「関連ドキュメント」の「追加」リンクをクリックします。

図：「プロパティ」 -「関連ドキュメント」

6. 「関連ドキュメント」ダイアログの項目を以下のように設定します。
   - ドキュメントタイプ：外部URL
   - URL：http://localhost:8080/imart/knowledge/contents/wiki_view/relation_document_wiki-wiki/
   - 表示名：任意

図：「関連ドキュメント」ダイアログ

> **注意**
>
> URLは、チュートリアルを行っている環境に合わせて指定してください。
> 今回は 「 http://localhost:8080/imart/ 」 を使用していると仮定し、行っています。
>
> ベースURLを省略することで、相対パス形式でURLを指定できます。
> 例：knowledge/contents/wiki_view/relation_document_wiki-wiki/
>
> URLは、絶対パスまたは相対パスで指定してください。
> 相対パスの指定は、IM-WikiとIM-Wikiを開こうとしている環境のベースURLが同一の場合のみ利用可能です。

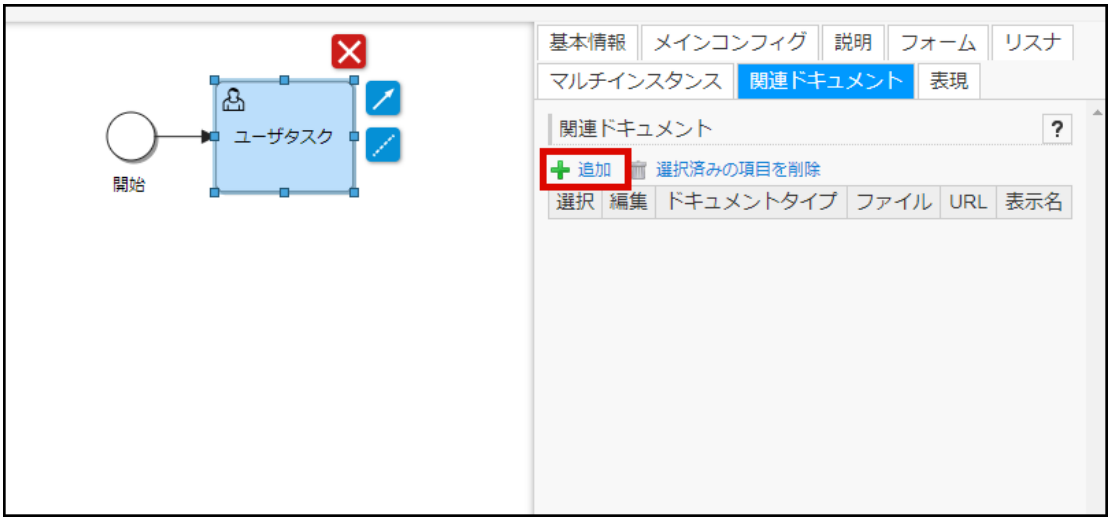
> **コラム**
>
> URLの「/wiki/」を「/wiki_view/」に置き換えることで、「編集」ボタンのないWikiを表示させることが可能です。
> 例：../imart/knowledge/contents/wiki_view/Wiki名

7. 「決定」をクリックします。
8. 「ユーザタスク」を配置します。
9. 「ユーザタスク」の「メインコンフィグ」で項目を以下のように設定します。
   - 担当者：${userCd1}

図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

> **コラム**
>
> `${userCd1}` は、「IM-FormaDesigner」で作成した入力フォームのフィールド識別IDです。
> 「担当者」にEL式を入力することにより、フォームで指定したユーザをタスクの担当者に設定できます。

10. 「関連ドキュメント」の設定を行います。
「ユーザタスク」の「関連ドキュメント」の「追加」リンクをクリックします。



図：「ユーザタスク」 - 「プロパティ」 - 「関連ドキュメント」

11. 「関連ドキュメント」ダイアログの項目を以下のように設定します。
    - ドキュメントタイプ：外部URL
    - URL：http://localhost:8080/imart/knowledge/contents/wiki_view/relation_document_wiki-wiki/ユーザタスク処理の際に参照するIM-Wiki?sidebar=false
    - 表示名：任意

図：「関連ドキュメント」ダイアログ

> **コラム**
>
> URLの最後に「 ?sidebar=false 」を書き加えることで、目次一覧が非表示のWikiを表示させることが可能です。
> 例：../imart/knowledge/contents/wiki_view/Wiki名 ?sidebar=false

12. 「決定」をクリックします。

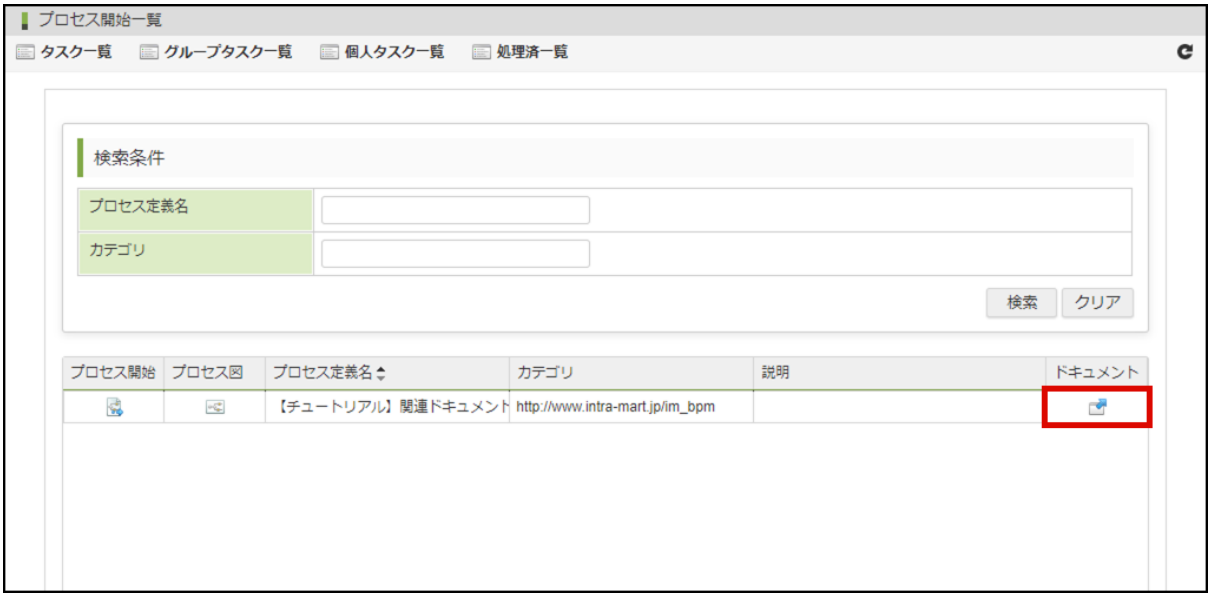
13. 「終了イベント」を配置します。

実行結果を確認する

このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行結果の確認を行います。
「Knowledge コンテンツ利用者」のロールを付与した aoyagi でログインしてください。

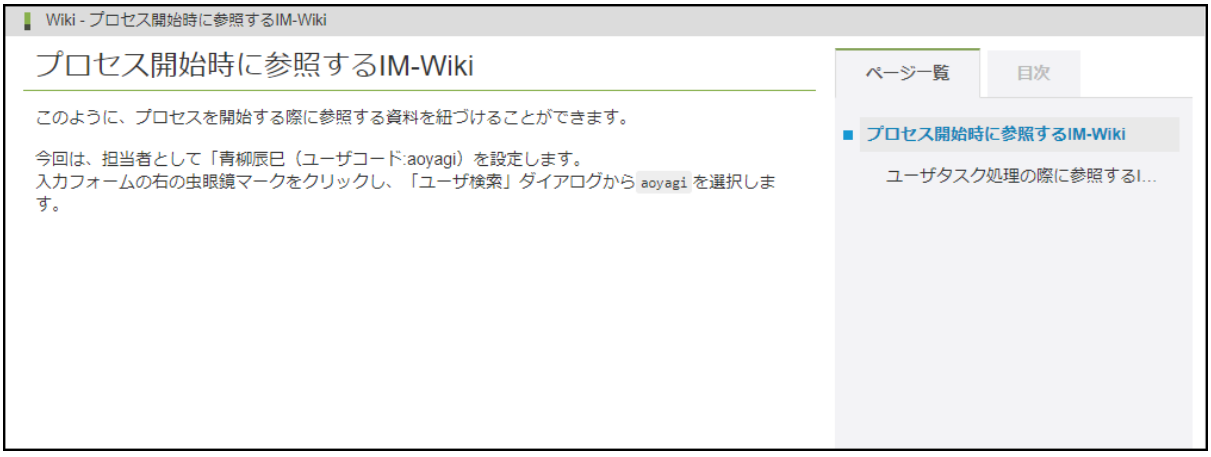
1. 「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示します。

2. 「 」アイコンをクリックします。



図：「プロセス開始一覧」

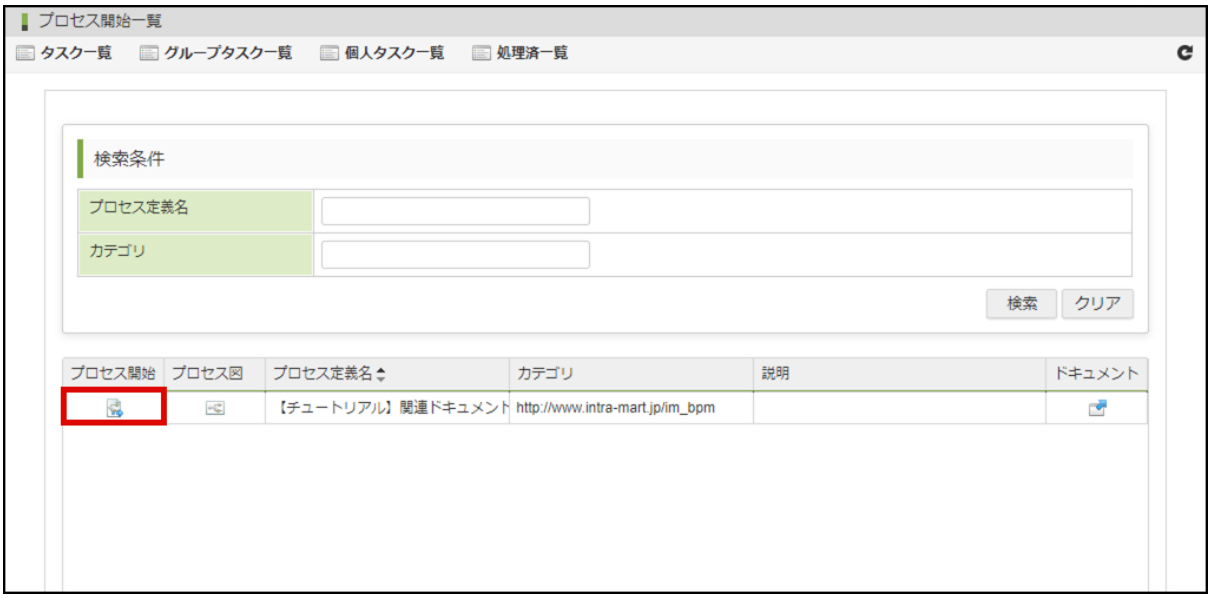
3. Wikiが新しいタブ、または、ウィンドウで表示されます。
参照用のURL形式（「 /wiki_view/ 」）を利用しているため、ここからWikiの編集は行えません。



図：「IM-Wiki」 - 「プロセス開始時に参照するIM-Wiki」

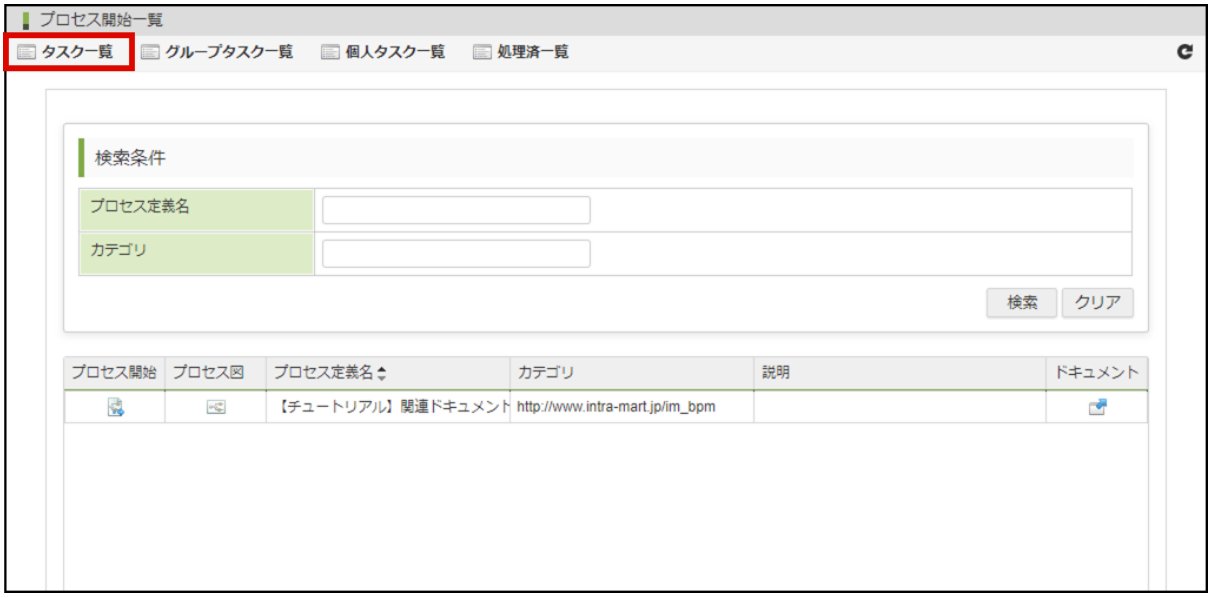
4. Wikiに従って処理を行います。

プロセス開始一覧から、「 」アイコンをクリックします。



図：「プロセス開始一覧」

5. 「プロセス開始一覧」画面の左上、「タスク一覧」をクリックします。



図：「プロセス開始一覧」

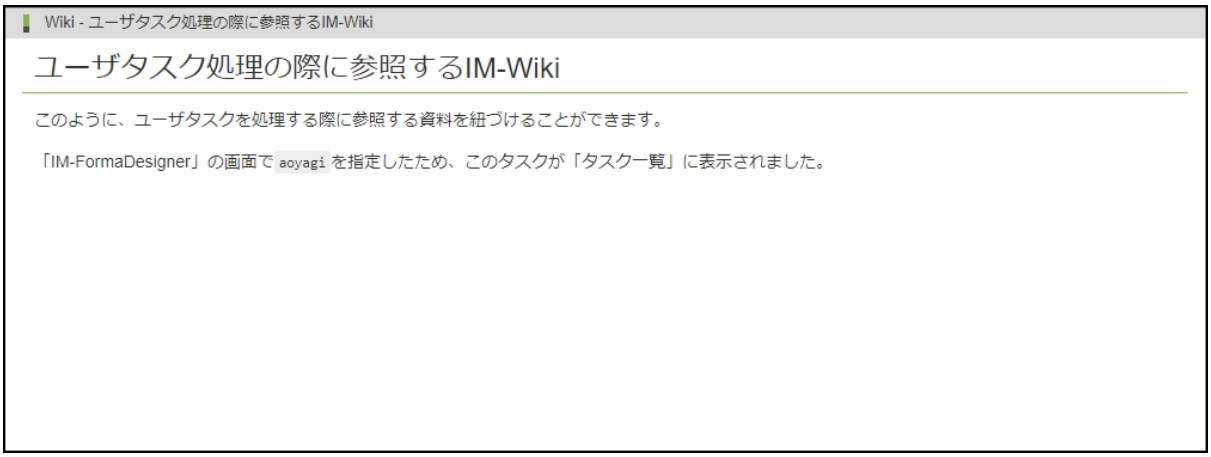
6. 個人タスクに、上記で実行したプロセスのタスクがあるので、「 」アイコンをクリックします。

図：「タスク一覧」

7. 「ドキュメント」ダイアログから、「ユーザタスク処理の際に参照するIM-Wiki」を選択します。

図：「ドキュメント」ダイアログ

8. Wikiが新しいタブ、または、ウィンドウで表示されます。
参照用のURL形式（「/Wiki名 ?sidebar=false 」）を利用しているため、Wikiの目次一覧は表示されません。



図：「IM-Wiki」 - 「ユーザタスク処理の際に参照するIM-Wiki」

プール、レーン

プールとレーンを利用して作業の関係者を明確にする

このチュートリアルでは、「プール」と「レーン」の基本的な使用方法について解説します。

- 「プール」とは、プロセス定義における関係者の境界線を表現したフローエレメントです。
- 「レーン」は、プールの中でさらに役割を区分したい場合に使用します。

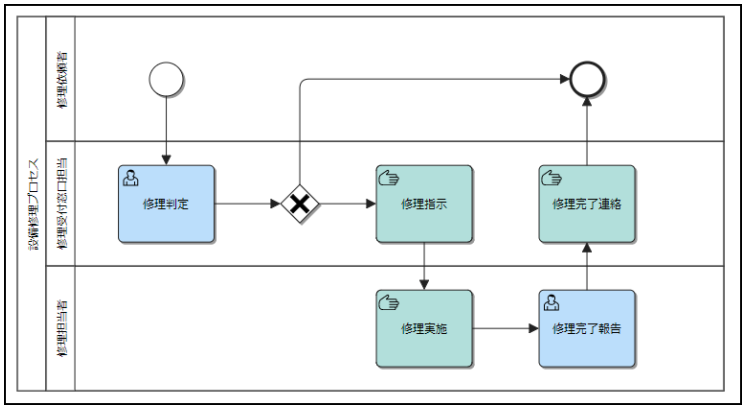
「プール」と「レーン」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「プール・レーン」もあわせて参照してください。

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **swimlane_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- 完成イメージ
- プールを配置し、プロパティを設定する
- レーンを追加し、プロパティを設定する
- エレメントを配置する
- 補足

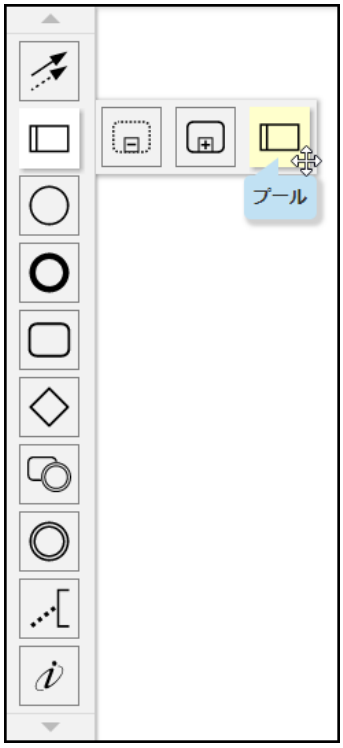
完成イメージ

このチュートリアルでは、以下のようなプロセスを作成します。



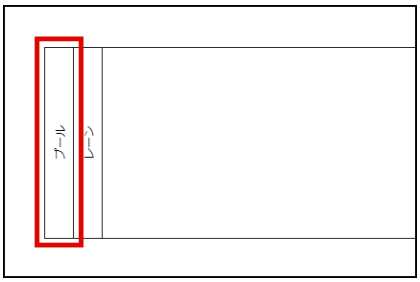
図：完成イメージ

プールを配置し、プロパティを設定する



図：パレット - コンテナ - プール

1. パレットの  にカーソルを合わせ、パレット右側に現れるコンテナの一覧から  をドラッグ＆ドロップして配置します。
2. レーンは、プールが配置された時点で一つだけプールの中に自動的に配置されます。
3. 配置されたプールは選択された状態になっているため、プロパティエリアで「基本情報」タブの「名前」に `設備修理プロセス` を入力します。
   プールの選択を解除してしまった場合は、下図の赤枠内をクリックすることで選択できます。



図：プールの選択

レーンを追加し、プロパティを設定する

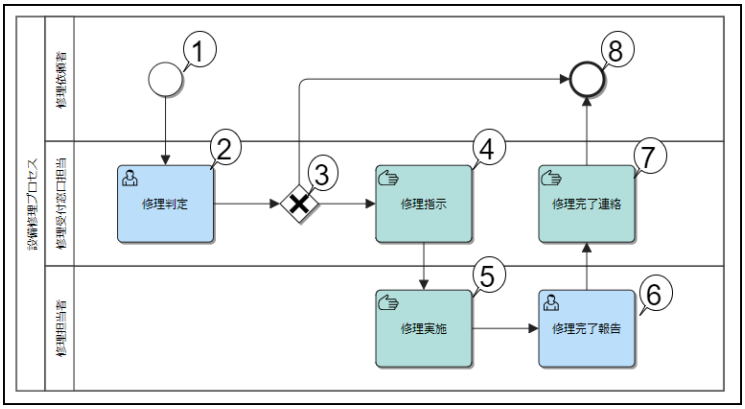
1. プールまたはレーン選択時に表示される  をクリックし、レーンを二つ追加します。
   - プールの**上部**にあるアイコンをクリックした場合、追加されるレーンはプール内の**最上部**に挿入されます。
   - プールの**下部**にあるアイコンをクリックした場合、追加されるレーンはプール内の**最下部**に挿入されます。

- レーンの**上部**にあるアイコンをクリックした場合、追加されるレーンは**直上**に挿入されます。
- レーンの**下部**にあるアイコンをクリックした場合、追加されるレーンは**直下**に挿入されます。

2. レーンのプロパティにて、「基本情報」タブの「名前」を上から順に以下のように設定します。
   - 修理依頼者
   - 修理受付窓口担当
   - 修理担当者

エレメントを配置する

1. 完成イメージを参考に、下表のとおり各レーンにエレメントを配置し、シーケンスフローで接続します。



図：完成イメージ

項番	レーン	エレメントタイプ	ラベル	接続元	接続先
1	修理依頼者	開始イベント	-	-	「修理判定」タスク
2	修理受付窓口担当	ユーザタスク	修理判定	開始イベント	排他ゲートウェイ
3	修理受付窓口担当	排他ゲートウェイ	-	「修理判定」タスク	「修理指示」タスク、終了イベント
4	修理受付窓口担当	マニュアルタスク	修理指示	排他ゲートウェイ	「修理実施」タスク
5	修理担当者	マニュアルタスク	修理実施	「修理指示」タスク	「修理完了報告」タスク
6	修理担当者	ユーザタスク	修理完了報告	「修理実施」タスク	「修理完了連絡」タスク
7	修理受付担当窓口	マニュアルタスク	修理完了連絡	「修理完了報告」タスク	終了イベント
8	修理依頼者	終了イベント	-	「修理完了報告」タスク、排他ゲートウェイ	-

補足

- 一つのファイル内に複数のプールを配置することもできます。
  デプロイ時には、一つのプールが一つのプロセス定義としてデプロイされます。
- レーンには実行権限を制御する機能はありません。
  レーンは業務の流れを視覚的に整理するエレメントです。

コールアクティビティ

コールアクティビティを使用する

このチュートリアルでは、「コールアクティビティ」を使用して他のプロセス定義を呼び出す方法を解説します。
「サブプロセス」との違いは、異なるプロセス定義ファイルに定義されたプロセス定義を呼び出すことが可能な点です。
「コールアクティビティ」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「コールアクティビティ」もあわせて参照してください。

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **call_activity_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

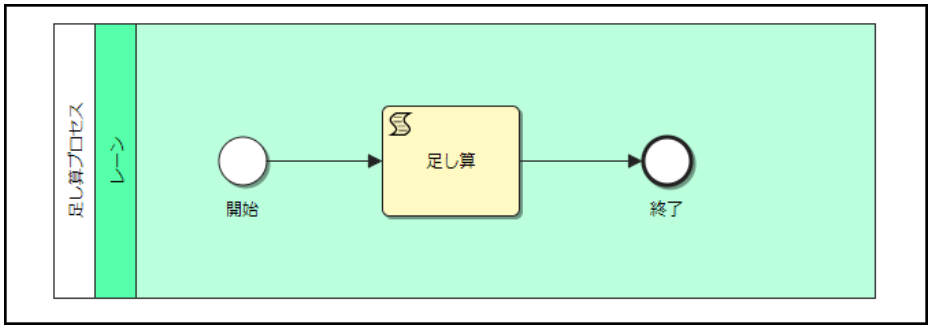
コラム

このチュートリアルのサンプルでは、同一ファイル内に「呼び出されるプロセス定義」と「呼び出し元のプロセス定義」が定義されていますが、それぞれファイルを分けて定義することも可能です。

- 呼び出されるプロセス定義を作成する
- 入力パラメータを設定してプロセス定義を呼び出す
- 参照可能な変数を継承してプロセス定義を呼び出す

呼び出されるプロセス定義を作成する

コールアクティビティから呼び出されるプロセス定義を作成します。



図：呼び出されるプロセス定義

このチュートリアルでは、足し算を「スクリプトタスク」で行い、結果を変数 result に設定する「呼び出されるプロセス定義」を作成します。
変数 result は後述する「呼び出し元のプロセス定義」にて「出力パラメータ」の設定を行うことにより、「呼び出し元のプロセス定義」の変数へ設定できます。
足し算の対象の数字は、「呼び出し元のプロセス定義」より変数を介して受け取ります。この際、数字の受け渡しに使用する変数の名前は連番（1, 2, 3, ...）を付与した規則的な変数名とします。

val + 数字 ： val1 , val2 , val3 , ・・・

スクリプトタスク「足し算」のスクリプトです。

```
function run(variables, execution, entity) {
  var result = 0;
  var i = 1;
  while (true) {
    var num = entity.getVariable('val' + i);
    if (num != null) {
      result += parseInt(num);
      i++;
    } else {
      break;
    }
  }

  entity.setVariable('result', result);
}
```

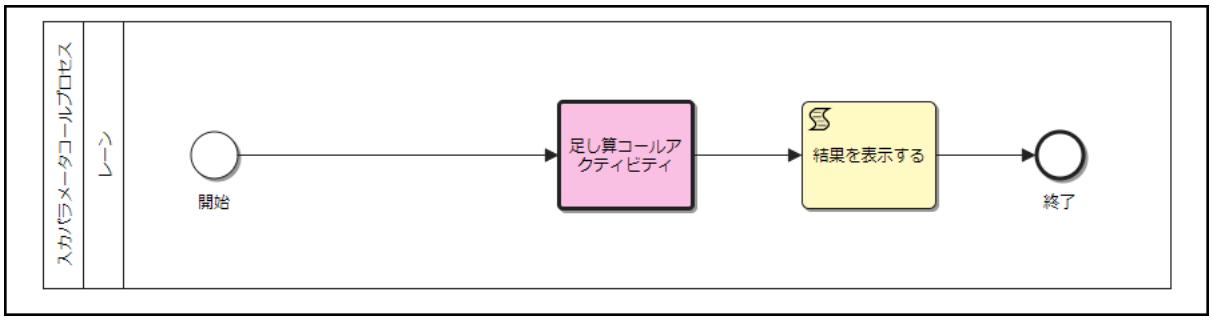
「プロセス定義キー」に addition_process を設定します。

図：プロセス定義キー

開始イベント、および、終了イベントには、特に設定が必要なプロパティはありません。

入力パラメータを設定してプロセス定義を呼び出す

コールアクティビティで入力パラメータを設定して、プロセス定義を呼び出します。

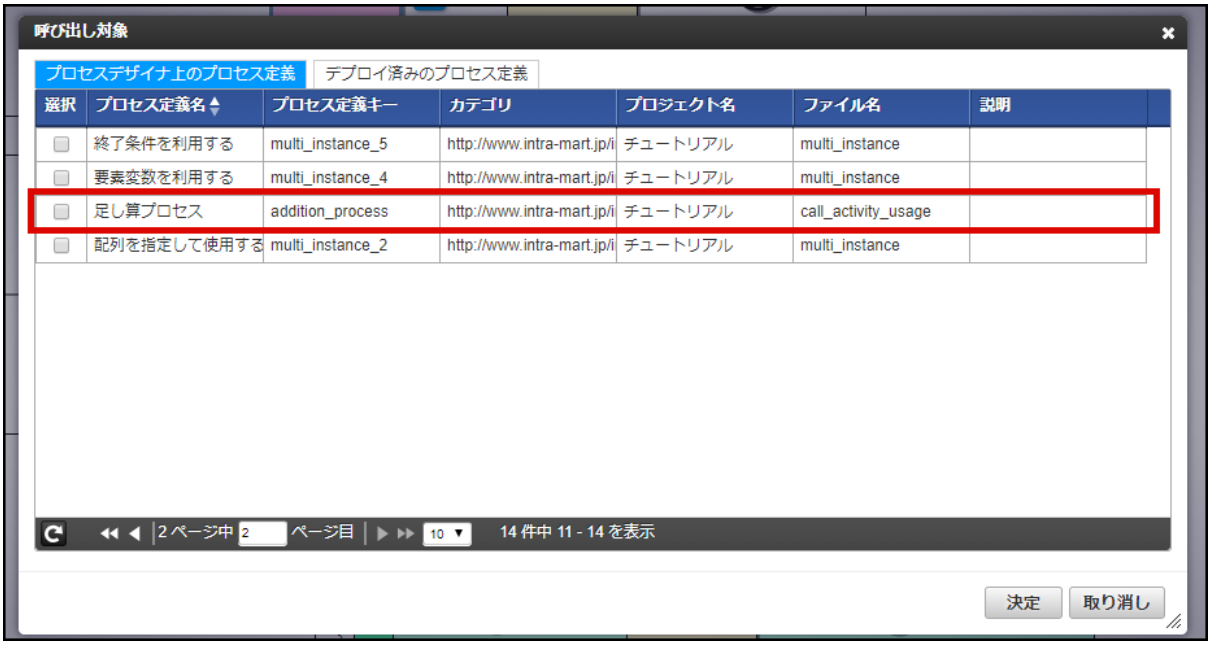


図：完成イメージ

図：コールアクティビティ - メインコンフィグ

1. 「呼び出し対象」に addition_process 指定します。

直接入力するか、プロセス定義検索（）を使用して指定します。



図：プロセス定義検索

コラム

プロセス定義検索

「プロセス定義検索」子画面で、参照可能なプロセス定義を検索し、「呼び出し対象」の項目へ設定できます。

- **「プロセスデザイナ上のプロセス定義」タブ**
  「プロセスデザイナ上のプロセス定義」タブでは、プロセスデザイナ上のプロセス定義が表示されます。
  「参照」アクションが許可されているプロジェクトのプロセス定義が表示されます。
- **「デプロイ済みのプロセス定義」タブ**
  「デプロイ済みのプロセス定義」タブでは、既にデプロイされているプロセス定義が表示さます。
  プロセス定義の検索の認可がある場合のみ表示されます。

プロジェクトの認可設定については「IM-BPM プロセスデザイナ 操作ガイド」-「プロジェクトの管理」-「プロジェクトの認可設定」を参照してください。

注意

「呼び出されるプロセス定義」はプロセス定義の実行前までに、必ずデプロイされている必要があります。
デプロイされていないプロセス定義が呼び出された場合、実行時にエラーが発生します。

2. 「入力パラメータ」を設定します。

上記「呼び出されるプロセス定義」の足し算の対象となる数字を格納する変数の名前は、連番（1, 2, 3, ...）を付与した規則的な変数名とします。
リンクをクリックします。

図：コールアクティビティ - メインコンフィグ - 入力パラメータ

1. 「入力パラメータ」の「ソース種別」に「式」を選択します。

2. 「ソース式」にEL式で数字を設定します。

   例：`${123}`

3. 「ターゲット変数名」に `val1` （val + 連番（1, 2, 3, ...））を設定します。

4. 上記の手順を繰り返し、足しあわせたい数字を全て「入力パラメータ」に設定します。

図：コールアクティビティ - メインコンフィグ

3. 「出力パラメータ」を設定します。

   上記「呼び出されるプロセス定義」で設定される変数 `result` の値を「呼び出し元プロセス定義」の変数 `output_result` へ格納する設定を行います。

   ✚リンクをクリックします。

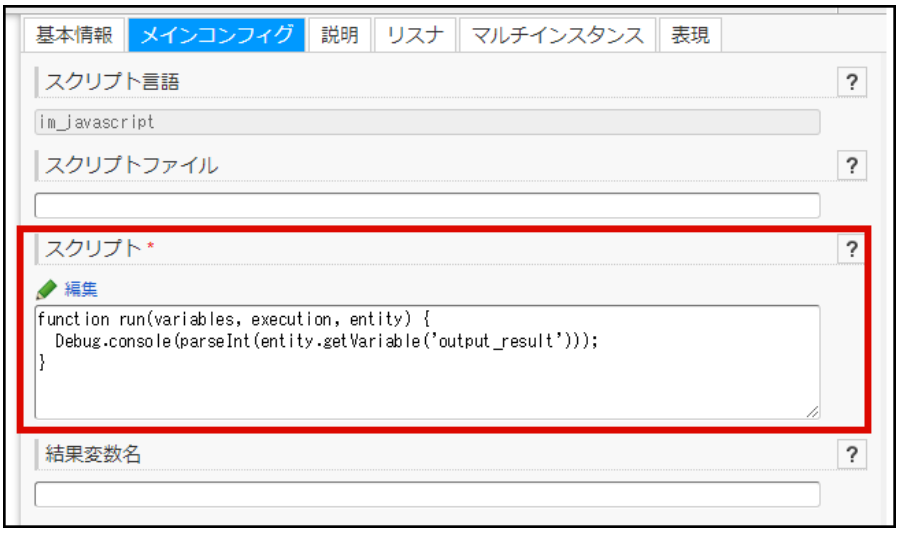
図：コールアクティビティ - メインコンフィグ - 出力パラメータ

   1. 「出力パラメータ」の「ソース種別」に「変数」を選択します。
   2. 「ソース変数名」に変数名 `result` を設定します。
   3. 「ターゲット変数名」に `output_result` を設定します。

4. 結果を表示する「スクリプトタスク」を作成します。

   「スクリプトタスク」のスクリプトです。

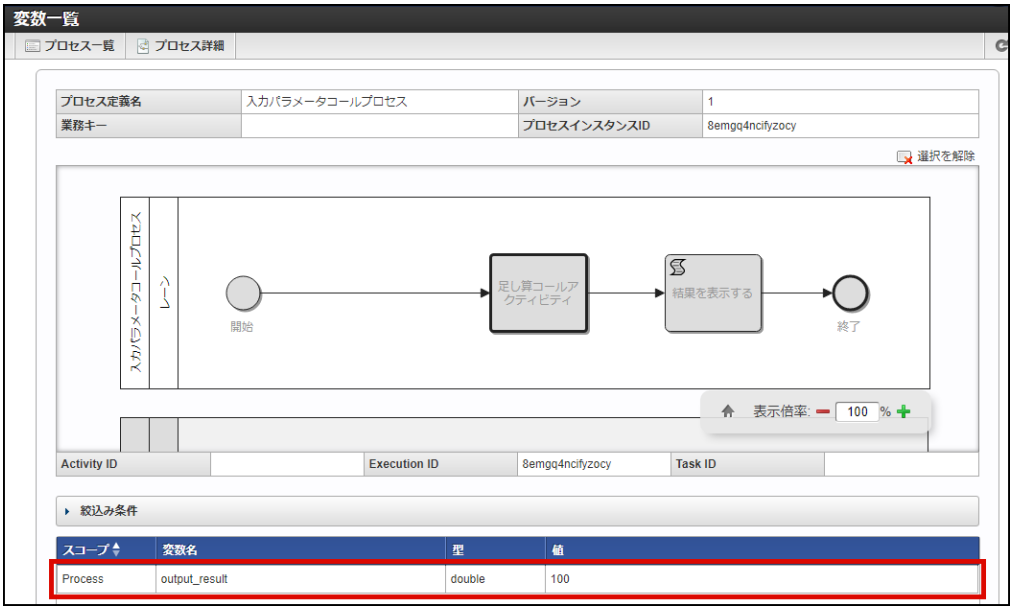
```
function run(variables, execution, entity) {
  Debug.console(parseInt(entity.getVariable('output_result')));
}
```



図：スクリプトタスク - メインコンフィグ - スクリプト

5. 実行結果を確認します。
   上記で作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

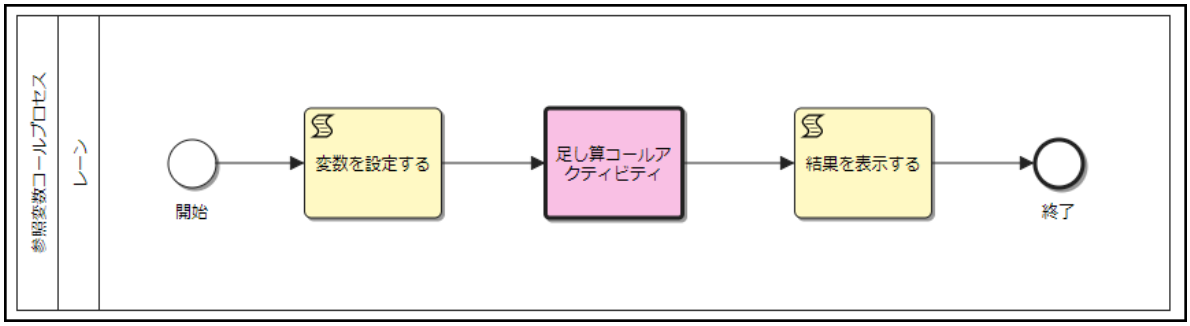
図：結果表示



図：変数一覧

参照可能な変数を継承してプロセス定義を呼び出す

コールアクティビティで参照可能な変数を継承してプロセス定義を呼びます。



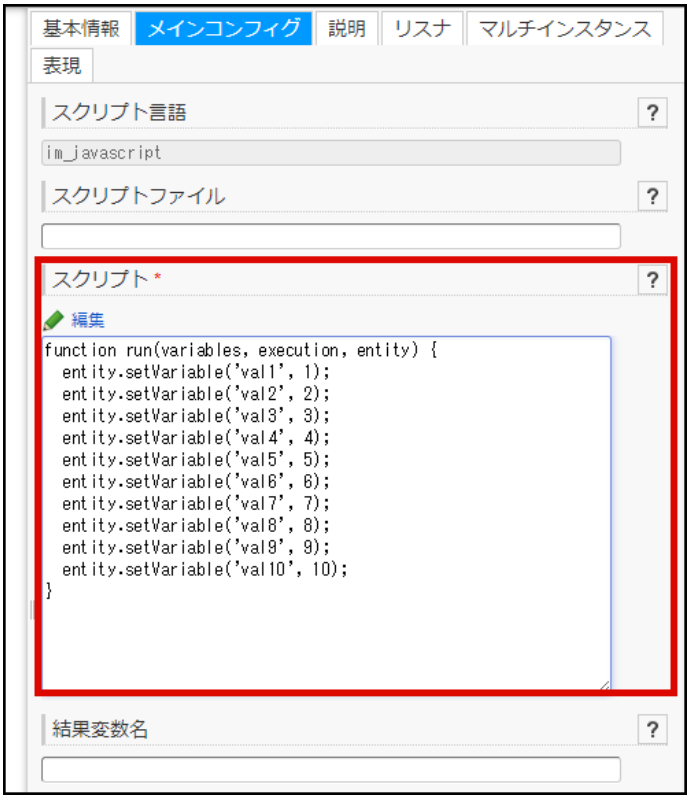
図：完成イメージ

図：コールアクティビティ - メインコンフィグ

1. 変数を設定するスクリプトタスクを作成します。

   「スクリプトタスク」のスクリプトです。

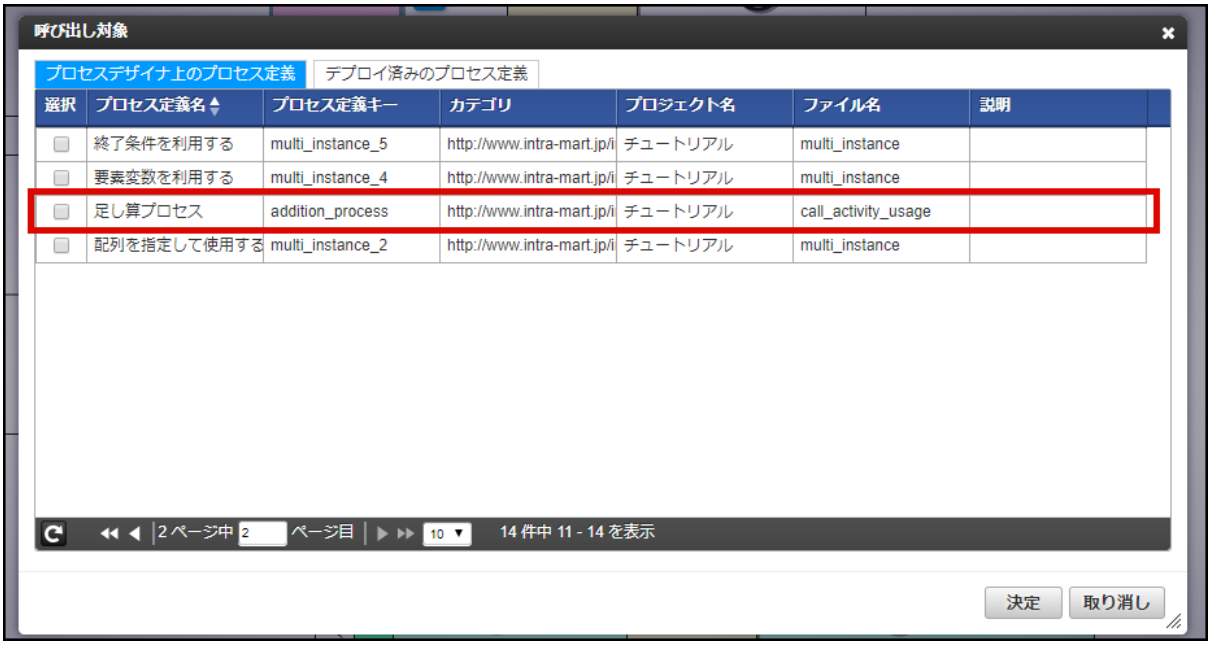
```
function run(variables, execution, entity) {
  entity.setVariable('val1', 1);
  entity.setVariable('val2', 2);
  entity.setVariable('val3', 3);
  entity.setVariable('val4', 4);
  entity.setVariable('val5', 5);
  entity.setVariable('val6', 6);
  entity.setVariable('val7', 7);
  entity.setVariable('val8', 8);
  entity.setVariable('val9', 9);
  entity.setVariable('val10', 10);
}
```



図：スクリプトタスク - メインコンフィグ - スクリプト

2. 「呼び出し対象」に `addition_process` 指定します。

   直接入力するか、プロセス定義検索を使用して指定します。



図：プロセス定義検索

3. 「参照可能な変数を継承する」を有効にします。

図：コールアクティビティ - メインコンフィグ - 参照可能な変数を継承する

4. 「出力パラメータ」を設定します。

上記「呼び出される プロセス定義」の変数 result の値を「呼び出し元プロセス定義」の変数 output_result へ格納する設定を行います。

リンクをクリックします。

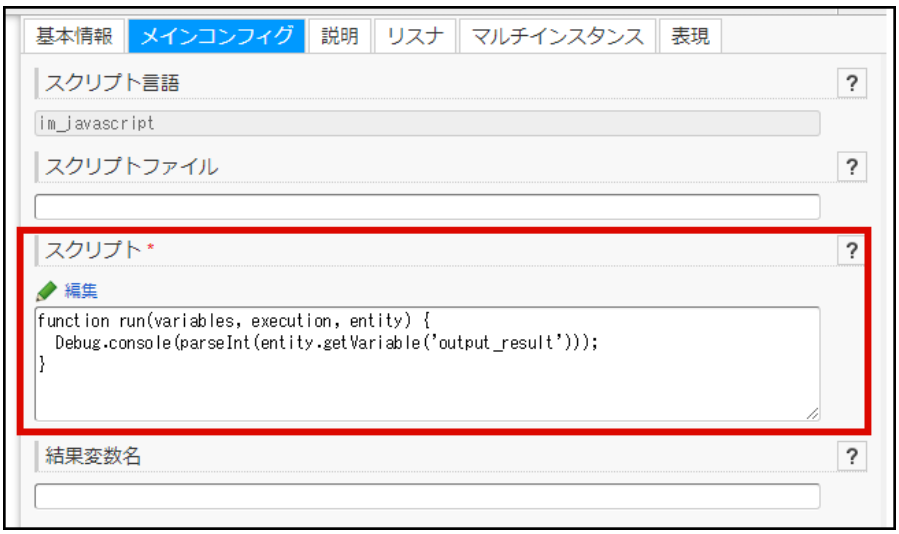
図：コールアクティビティ - メインコンフィグ - 出力パラメータ

1. 「出力パラメータ」の「ソース種別」に「変数」を選択します。
2. 「ソース変数名」に変数名 result を設定します。
3. 「ターゲット変数名」に output_result を設定します。

5. 結果を表示する「スクリプトタスク」を作成します。

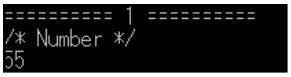
「スクリプトタスク」のスクリプトです。

```
function run(variables, execution, entity) {
  Debug.console(parseInt(entity.getVariable('output_result')));
}
```

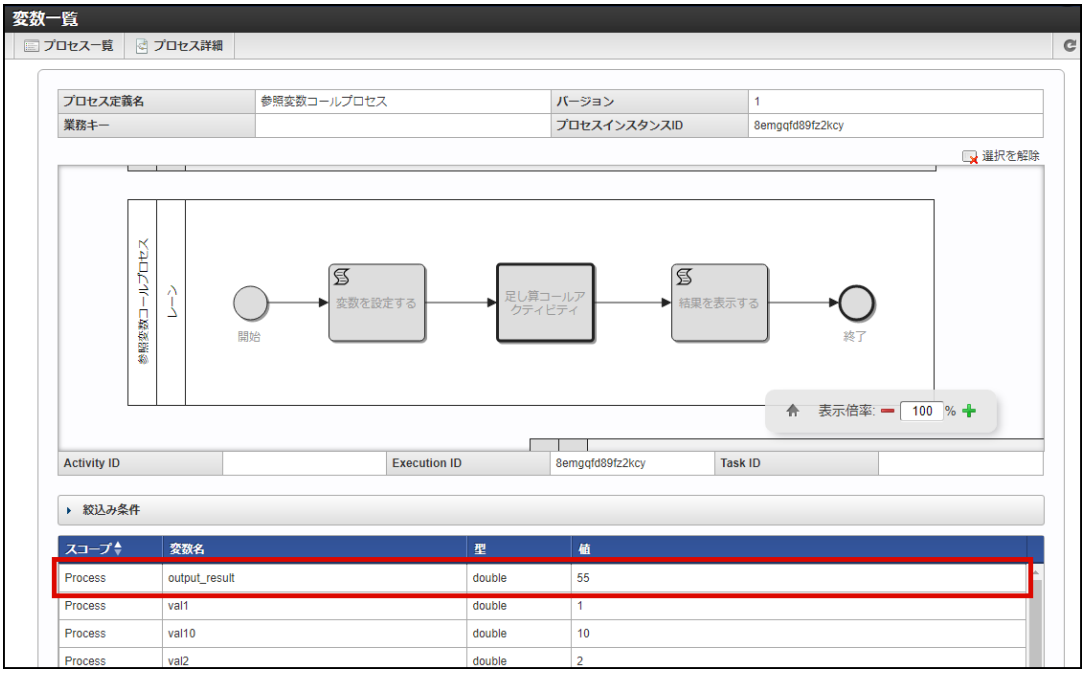


図：スクリプトタスク - メインコンフィグ - スクリプト

6. 実行結果を確認します。
   上記で作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。



図：結果表示



図：変数一覧

コールアクティビティで呼び出すプロセス定義に業務キーを設定する

このチュートリアルでは、「コールアクティビティ」を使用して他のプロセス定義を呼び出す際に「業務キー」を設定する方法を解説します。
「コールアクティビティ」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「コールアクティビティ」もあわせて参照してください。

コラム

このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
call_activity_business_key.bpmn

このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

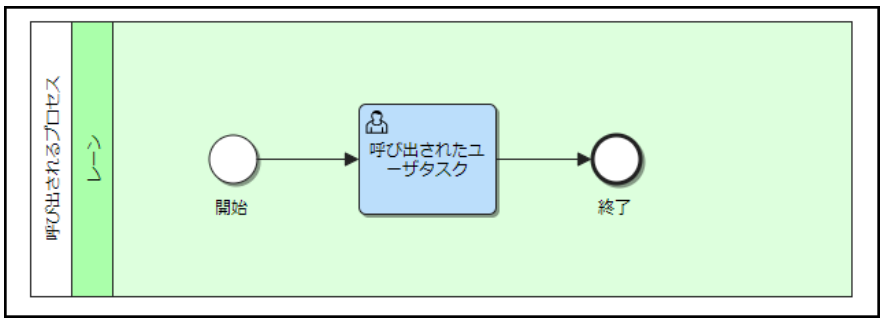
コラム

このチュートリアルのサンプルでは、同一ファイル内に「呼び出されるプロセス定義」と「呼び出し元のプロセス定義」が定義されていますが、それぞれファイルを分けて定義することも可能です。

- 呼び出されるプロセス定義を作成する
- 呼び出し元のプロセス定義から業務キーを継承する
- 任意に業務キーを設定する

呼び出されるプロセス定義を作成する

コールアクティビティから呼び出されるプロセス定義を作成します。

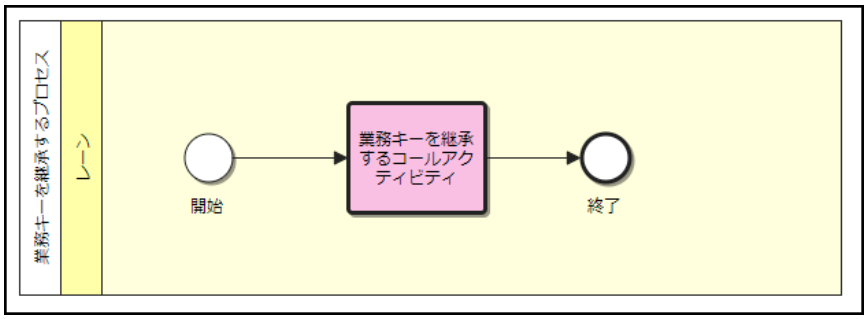


図：呼び出されるプロセス定義

「プロセス定義キー」に receive_business_key_process を設定します。

呼び出し元のプロセス定義から業務キーを継承する

コールアクティビティで「業務キーを継承する」を有効にして、プロセス定義を呼び出します。



図：完成イメージ

1. 「呼び出し対象」に receive_business_key_process 指定します。

   直接入力するか、プロセス定義検索（🔎）を使用して指定します。

   コラム

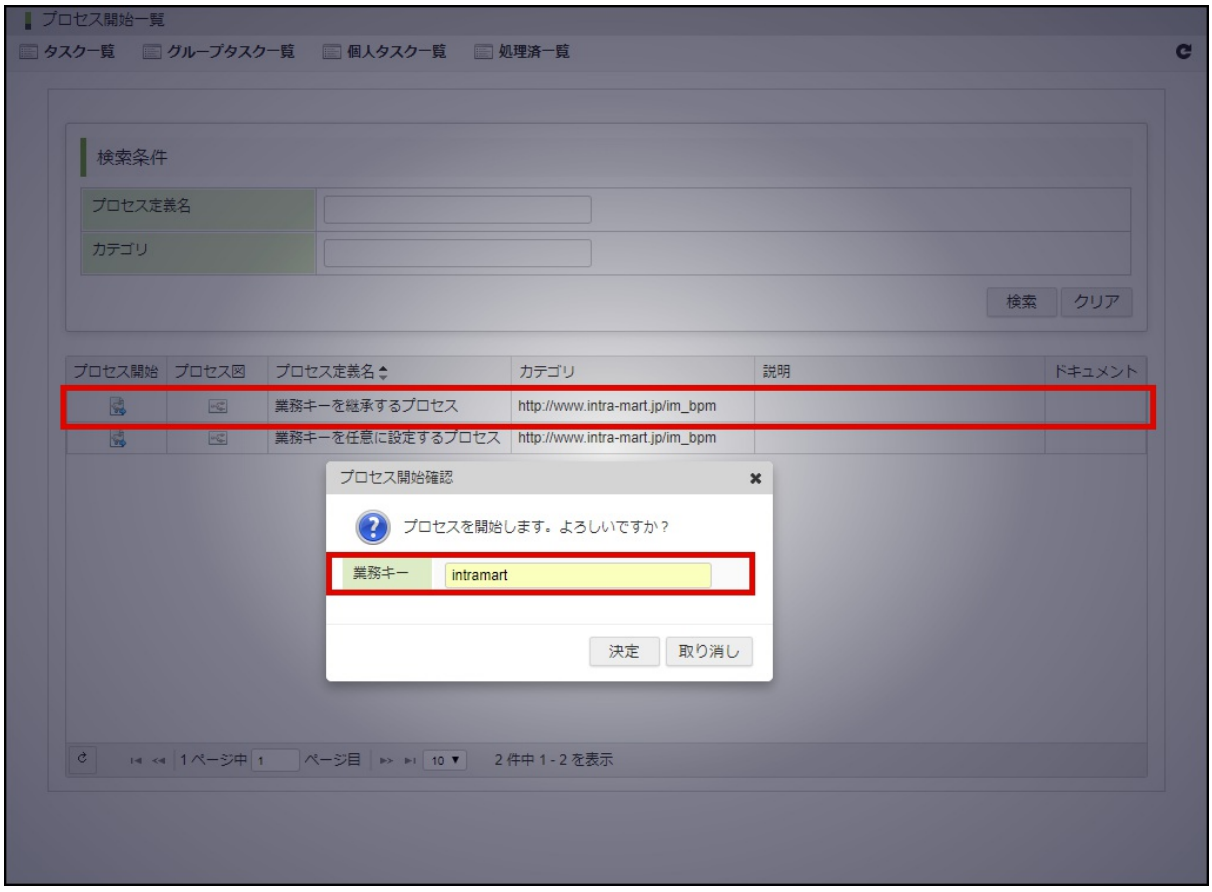
   「呼び出し対象」の指定は、「*コールアクティビティを使用する*」を参考にしてください。

2. 「業務キーを継承する」を有効にします。

図：コールアクティビティ - メインコンフィグ

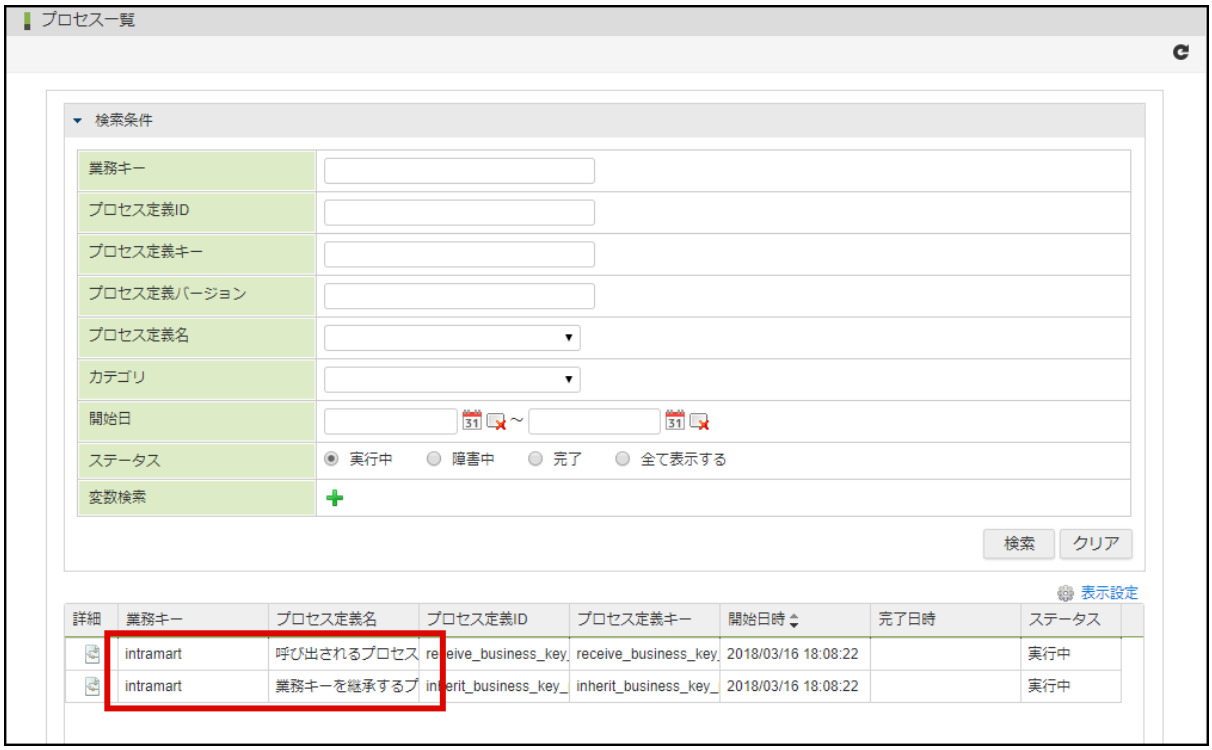
実行結果を確認します。

1. 上記で作成したプロセス定義を実行環境にデプロイします。
2. 任意の業務キーを設定しプロセスを開始します。



図：プロセス開始一覧

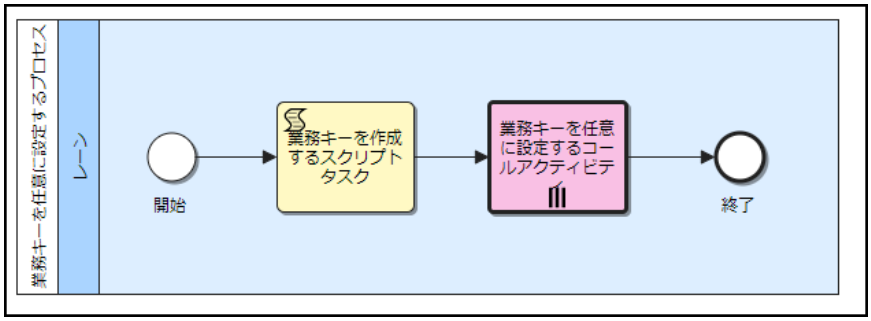
3. プロセス一覧で確認します。



図：プロセス一覧

任意に業務キーを設定する

コールアクティビティで任意に業務キーを設定して、プロセス定義を呼び出します。



図：完成イメージ

1. スクリプトタスクで、業務キーの配列を作成します。

```
function run(variables, execution, entity) {
  var businessKeys = new java.util.ArrayList();

  for (var i = 1; i <= 3; i++) {
    businessKeys.add(entity.getProcessBusinessKey() + '_' + i);
  }

  entity.setVariable('businessKeys', businessKeys);
}
```

2. 「呼び出し対象」に `receive_business_key_process` 指定します。

   直接入力するか、プロセス定義検索（🔎）を使用して指定します。

   > **コラム**
   >
   > 「呼び出し対象」の指定は、「*コールアクティビティを使用する*」を参考にしてください。

3. 「業務キー」に `${businessKey}` 指定します。

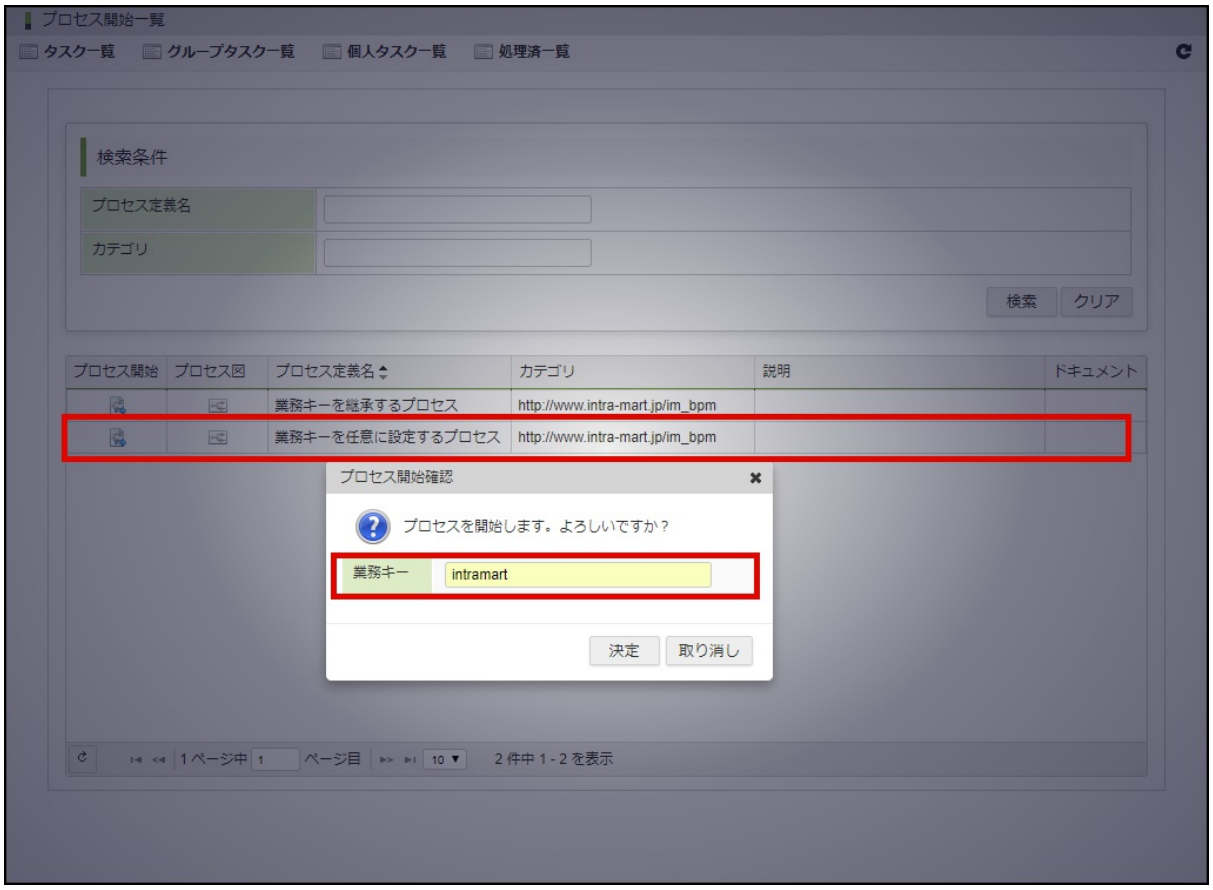
図：コールアクティビティ - メインコンフィグ

1. マルチインスタンスを設定します。
    - 繰り返しの種別：`配列`
    - 配列：`${businessKeys}`
    - 要素変数：`businessKey`

図： マルチインスタンス

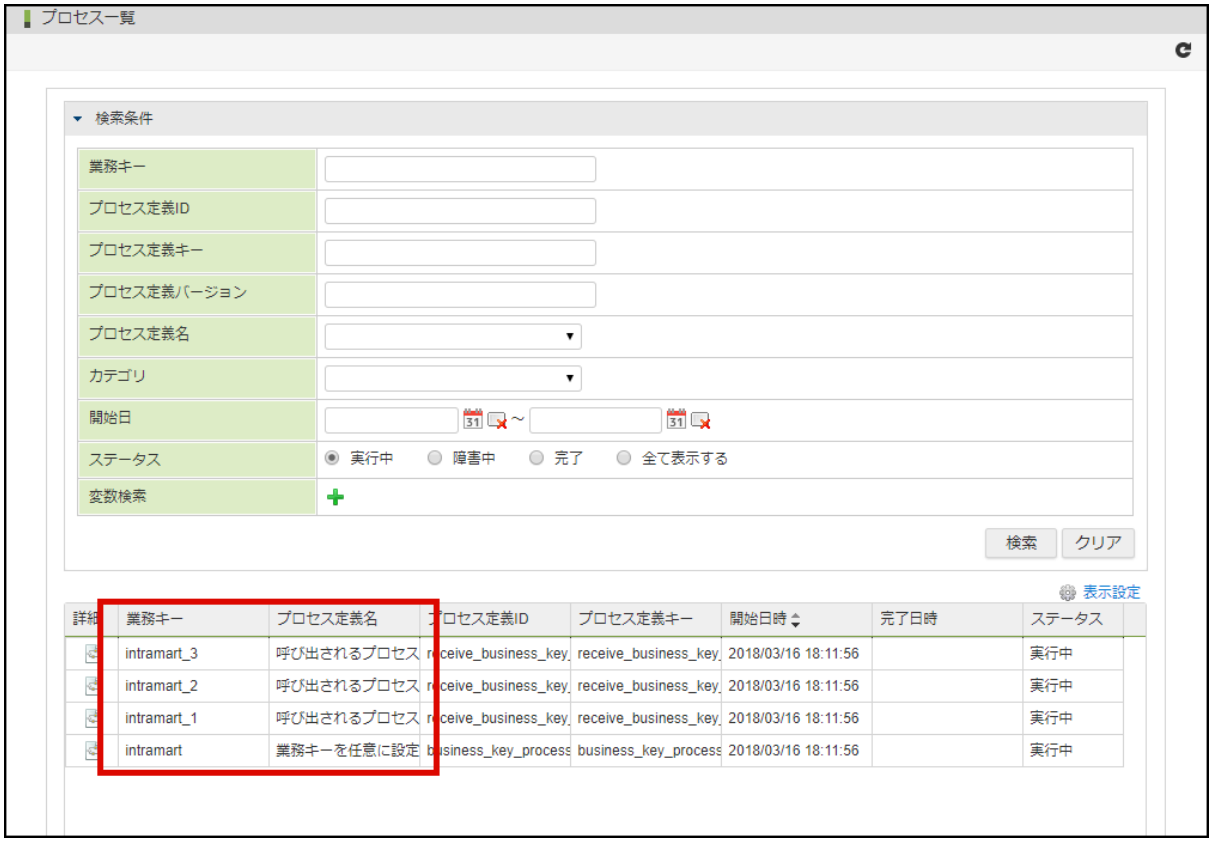
実行結果を確認します。

1. 上記で作成したプロセス定義を実行環境にデプロイします。
2. 任意の業務キーを設定しプロセスを開始します。



図：プロセス開始一覧

3. プロセス一覧で確認します。



図：プロセス一覧

パラレルゲートウェイ

パラレルゲートウェイを使用する

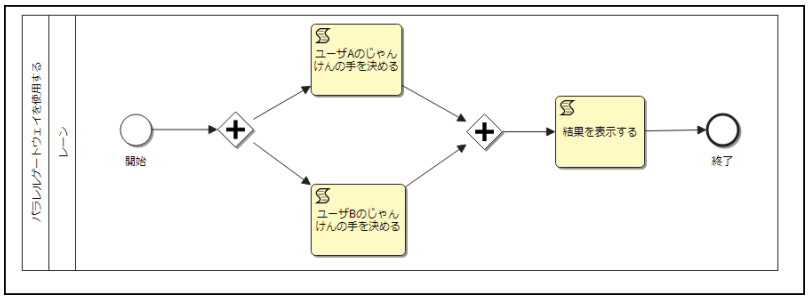
このチュートリアルでは、「パラレルゲートウェイ」を使用してプロセスの並列処理の開始や合流の定義方法を解説します。
「パラレルゲートウェイ」を使用することで、無条件で並列処理が行えます。
「パラレルゲートウェイ」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「ゲートウェイ」 - 「パラレルゲートウェイ」もあわせて参照してください。

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **parallel_gateway_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- パラレルゲートウェイを使用し、並列処理を行う

パラレルゲートウェイを使用し、並列処理を行う

以下の図は、ユーザAとユーザBでじゃんけんを行うプロセスです。
ユーザA、および、ユーザBの手を選択するスクリプトタスクを並列に実行します。
2つのスクリプトタスクがどちらも完了したら結果を判定するスクリプトタスクを呼び出します。

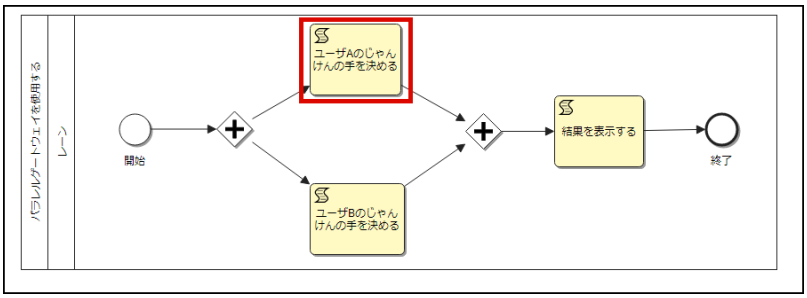


図：完成イメージ

1. パラレルゲートウェイのプロパティを設定します。
   パラレルゲートウェイで分岐と結合を行うにあたり、特別なプロパティの設定を行う必要はありません。

2. シーケンスフローのプロパティを設定します。
   パラレルゲートウェイから出力されているシーケンスフローは全て無条件で同時並列に分岐対象となるため、特別なプロパティの設定を行う必要はありません。

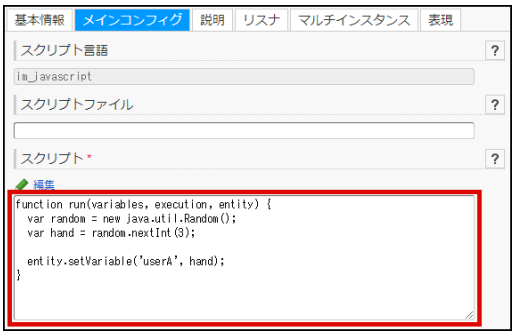
3. スクリプトタスク「ユーザAのじゃんけんの手を決める」 のプロパティを設定します。



図：スクリプトタスク「ユーザAのじゃんけんの手を決める」

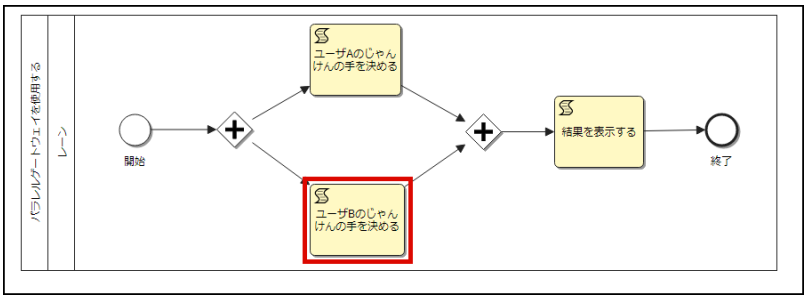
スクリプトタスク「ユーザAのじゃんけんの手を決める」にユーザAの手をランダムに選択し、変数 `userA` に格納するスクリプトを設定します。

```
function run(variables, execution, entity) {
  var random = new java.util.Random();
  var hand = random.nextInt(3);

  entity.setVariable('userA', hand);
}
```



図：スクリプトタスク「ユーザAのじゃんけんの手を決める」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

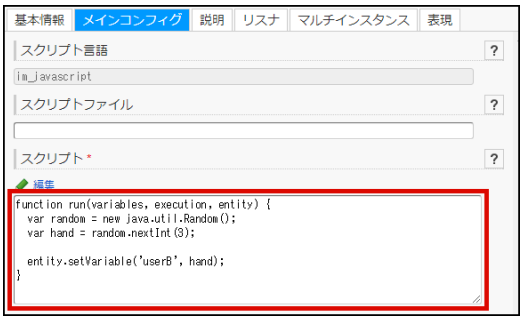
4. スクリプトタスク「ユーザBのじゃんけんの手を決める」 のプロパティを設定します。



図：スクリプトタスク「ユーザBのじゃんけんの手を決める」

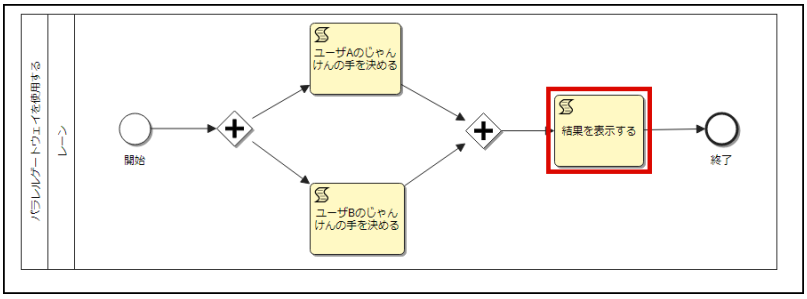
スクリプトタスク「ユーザBのじゃんけんの手を決める」にユーザBの手をランダムに選択し、変数 `userB` に格納するスクリプトを設定します。

```
function run(variables, execution, entity) {
  var random = new java.util.Random();
  var hand = random.nextInt(3);

  entity.setVariable('userB', hand);
}
```



図：スクリプトタスク「ユーザBのじゃんけんの手を決める」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

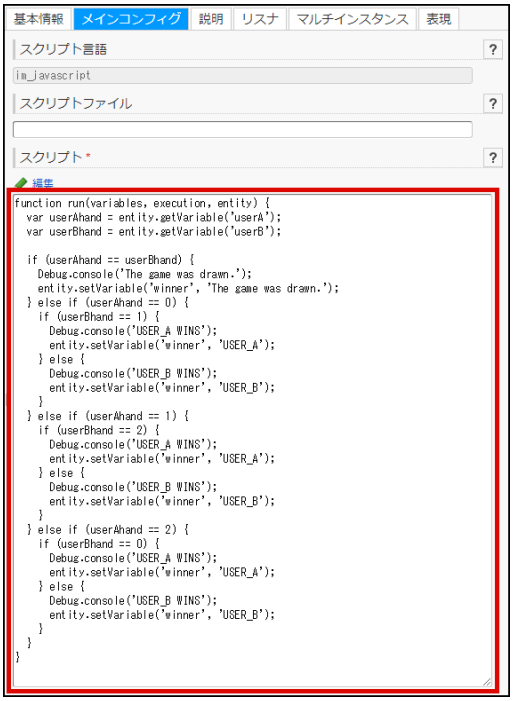
5. スクリプトタスク「結果を表示する」のプロパティを設定します。



図：スクリプトタスク「結果を表示する」

スクリプトタスク「結果を表示する」にユーザAとユーザBのじゃんけんの勝者を変数 `winner` に格納するスクリプトを設定します。

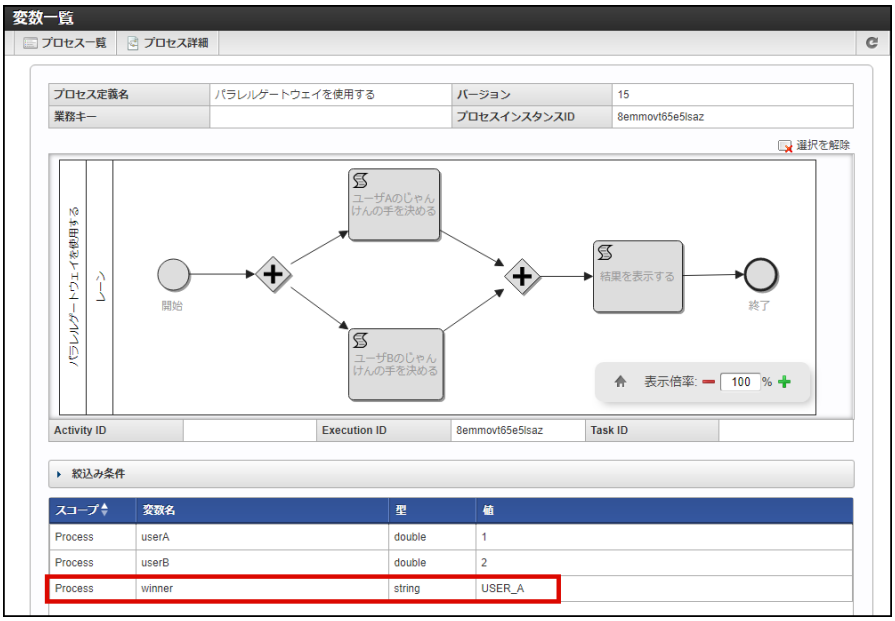
```
function run(variables, execution, entity) {
  var userAhand = entity.getVariable('userA');
  var userBhand = entity.getVariable('userB');

  if (userAhand == userBhand) {
    Debug.console('The game was drawn.');
    entity.setVariable('winner', 'The game was drawn.');
  } else if (userAhand == 0) {
    if (userBhand == 1) {
      Debug.console('USER_A WINS');
      entity.setVariable('winner', 'USER_A');
    } else {
      Debug.console('USER_B WINS');
      entity.setVariable('winner', 'USER_B');
    }
  } else if (userAhand == 1) {
    if (userBhand == 2) {
      Debug.console('USER_A WINS');
      entity.setVariable('winner', 'USER_A');
    } else {
      Debug.console('USER_B WINS');
      entity.setVariable('winner', 'USER_B');
    }
  } else if (userAhand == 2) {
    if (userBhand == 0) {
      Debug.console('USER_A WINS');
      entity.setVariable('winner', 'USER_A');
    } else {
      Debug.console('USER_B WINS');
      entity.setVariable('winner', 'USER_B');
    }
  }
}
```



図：スクリプトタスク「結果を表示する」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

6. 実行結果を確認します。
   上記で作成したプロセスを実行環境にデプロイし、実行した結果の確認を行います。

図：結果表示



図：変数一覧

排他ゲートウェイ

排他ゲートウェイを使用する

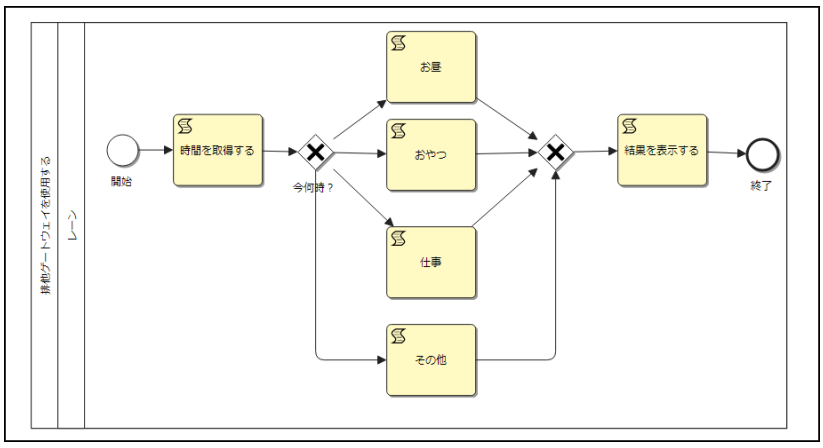
このチュートリアルでは、「排他ゲートウェイ」を使用してプロセスの遷移先の選択や合流の定義方法を解説します。
「排他ゲートウェイ」を使用することで、最初にtrueと評価された1つのシーケンスフローだけが継続して次の処理に進むフローを作成できます。
「排他ゲートウェイ」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「ゲートウェイ」 - 「排他ゲートウェイ」もあわせて参照してください。

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **exclusive_gateway_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- 排他ゲートウェイを使用し、複数の分岐先から1つの遷移先を選択する

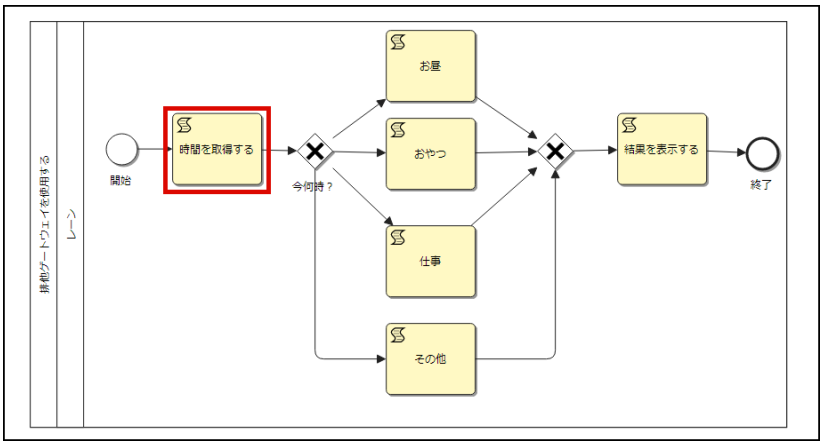
排他ゲートウェイを使用し、複数の分岐先から1つの遷移先を選択する

以下の図は、現在時と作業名をログ出力するプロセスです。
スクリプトタスクで取得した現在時刻によって、排他ゲートウェイで分岐します。
分岐した先では、現在時に対応する作業名を取得するスクリプトタスクを実行します。



図：完成イメージ

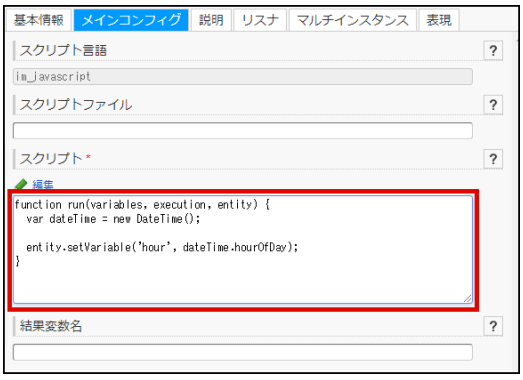
1. スクリプトタスク「時間を取得する」 のプロパティを設定します。



図：スクリプトタスク「時間を取得する」

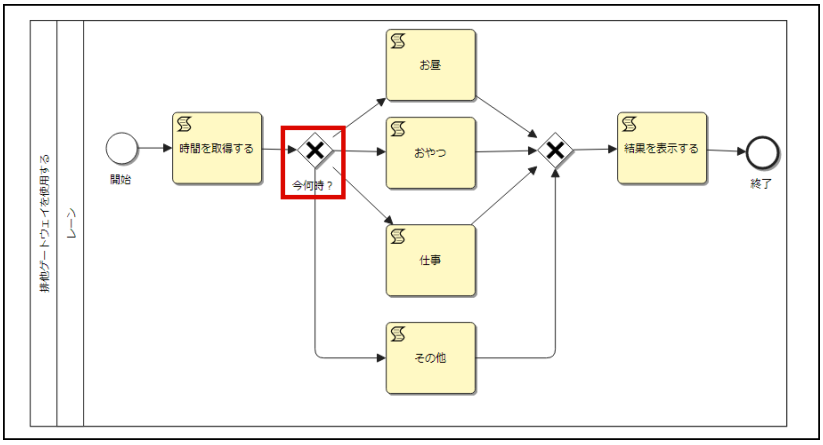
スクリプトタスク「時間を取得する」に、現在の時間を取得し変数 hour に格納するスクリプトを設定します。

```
function run(variables, execution, entity) {
  var dateTime = new DateTime();

  entity.setVariable('hour', dateTime.hourOfDay);
}
```



図：スクリプトタスク「時間を取得する」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

2. 排他ゲートウェイのプロパティを設定します。

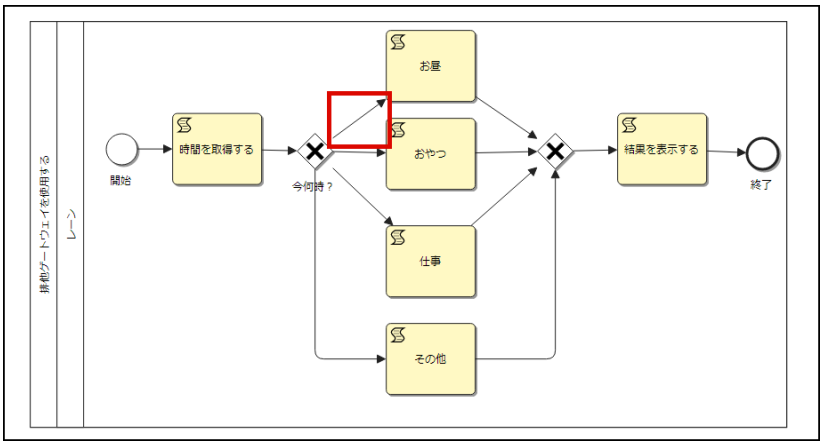


図：排他ゲートウェイ「今何時？」

分岐の開始となる排他ゲートウェイ「今何時？」に「デフォルトフロー」として sequence-flow_6 （スクリプトタスク「その他」に接続されているシーケンスフロー）を設定します。
結合側の排他ゲートウェイでは特別なプロパティの設定を行う必要はありません。

図：排他ゲートウェイ「今何時？」の「プロパティエリア」-「基本情報」-「デフォルトフロー」

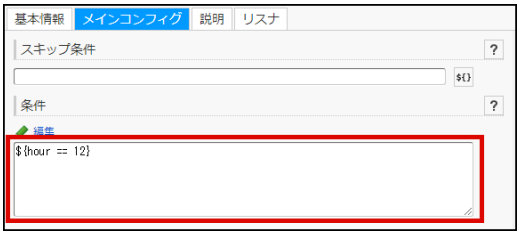
3. シーケンスフロー「sequence-flow_3」の条件を設定します。



図：シーケンスフロー「sequence-flow_3」

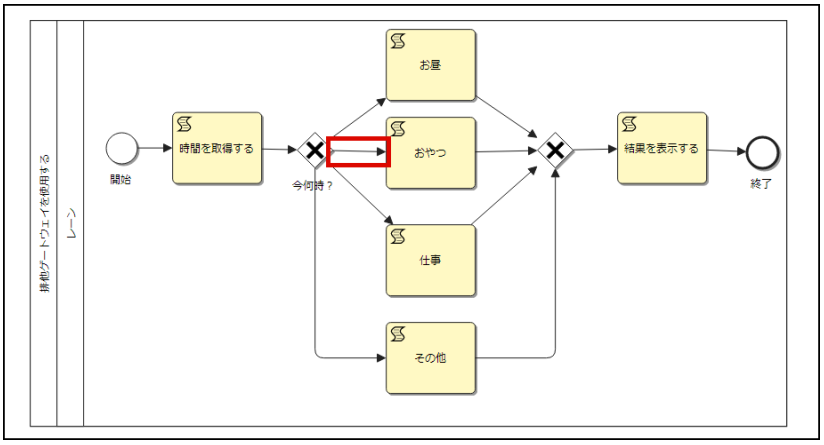
排他ゲートウェイ「今何時？」とスクリプトタスク「お昼」を接続しているシーケンスフロー「sequence-flow_3」 に、変数 hour が 12 の場合に真となる条件を設定します。

```
${hour == 12}
```



図：シーケンスフロー「sequence-flow_3」の「プロパティエリア」-「メインコンフィグ」-「条件」

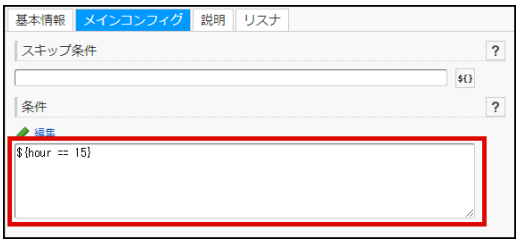
4. シーケンスフロー「sequence-flow_4」の条件を設定します。



図：シーケンスフロー「sequence-flow_4」

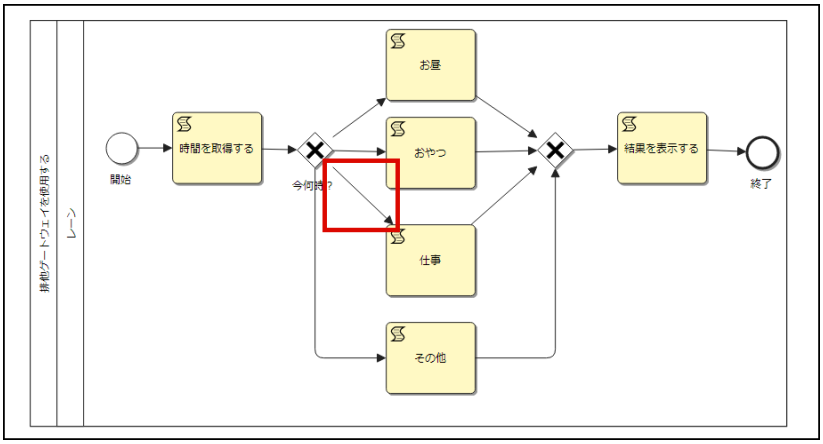
排他ゲートウェイ「今何時？」とスクリプトタスク「おやつ」を接続しているシーケンスフロー「sequence-flow_4」 に、変数 hour が 15 の場合に真となる条件を設定します。

```
${hour == 15}
```



図：シーケンスフロー「sequence-flow_4」の「プロパティエリア」-「メインコンフィグ」-「条件」

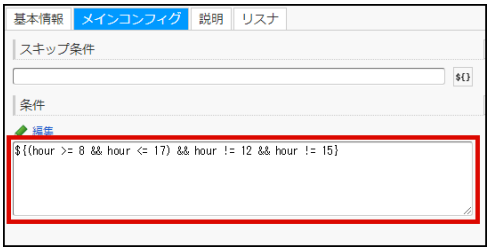
5. シーケンスフロー「sequence-flow_5」の条件を設定します。



図：シーケンスフロー「sequence-flow_5」

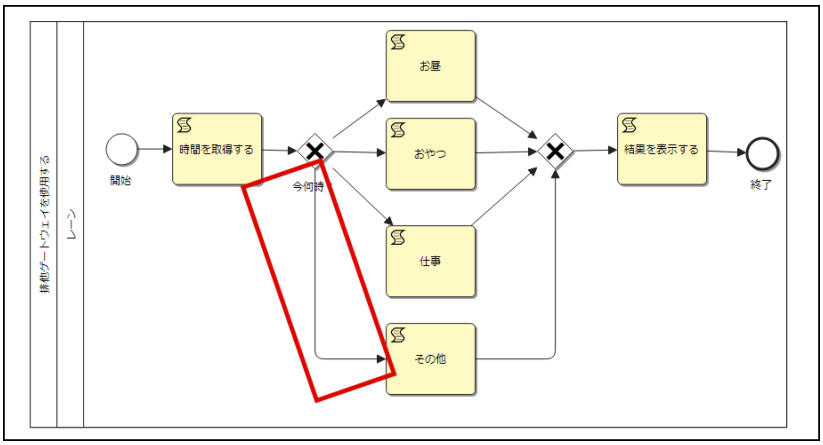
排他ゲートウェイ「今何時？」とスクリプトタスク「仕事」を接続しているシーケンスフロー「sequence-flow_5」 に、変数 `hour` が `8` 以上かつ `17` 以下で、 `12` ではなく、 `15` でもない場合に真となる条件を設定します。

```
${(hour >= 8 && hour <= 17) && hour != 12 && hour != 15}
```



図：シーケンスフロー「sequence-flow_5」の「プロパティエリア」-「メインコンフィグ」-「条件」

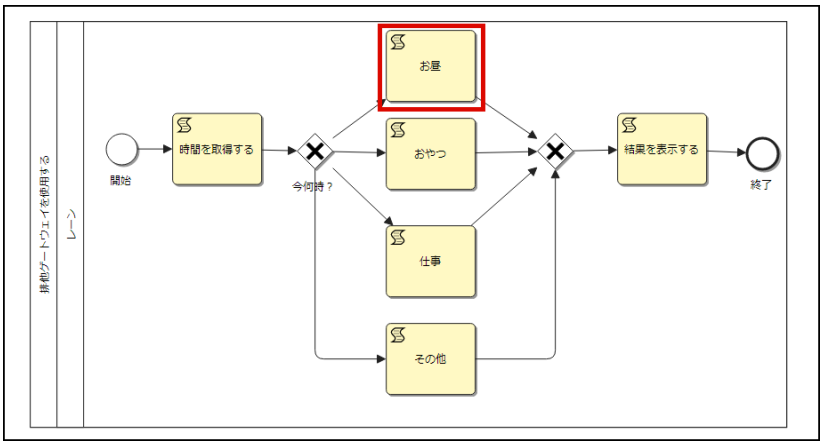
6. シーケンスフロー「sequence-flow_6」の条件を設定します。



図：スクリプトタスク「sequence-flow_6」

排他ゲートウェイ「今何時？」とスクリプトタスク「その他」を接続しているシーケンスフロー「sequence-flow_6」は、排他ゲートウェイの「デフォルトフロー」であるため条件の設定はできません。

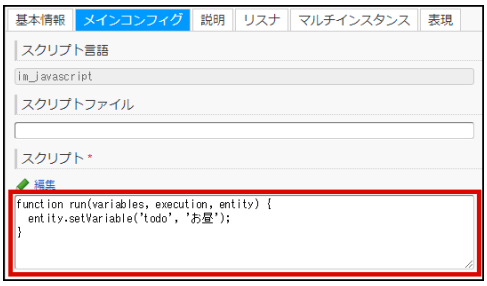
7. スクリプトタスク「お昼」 のプロパティを設定します。



図：スクリプトタスク「お昼」

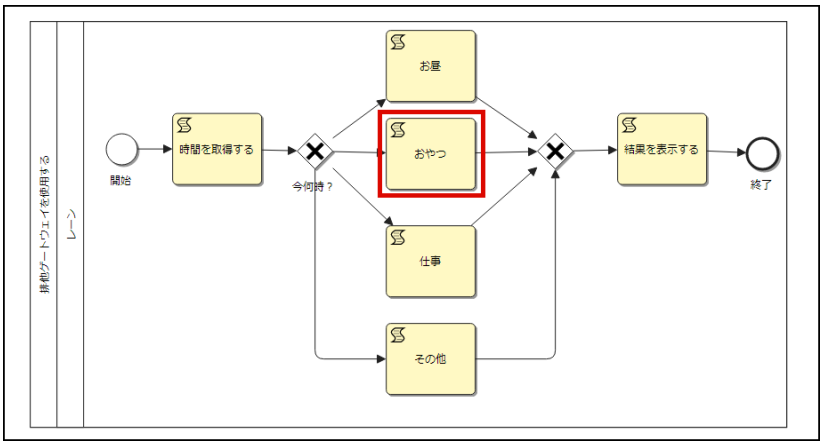
スクリプトタスク「お昼」のプロパティにて、作業名 `お昼` を変数 `todo` に格納するスクリプトを設定します。

```
function run(variables, execution, entity) {
  entity.setVariable('todo', 'お昼');
}
```



図：スクリプトタスク「お昼」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

8. スクリプトタスク「おやつ」 のプロパティを設定します。



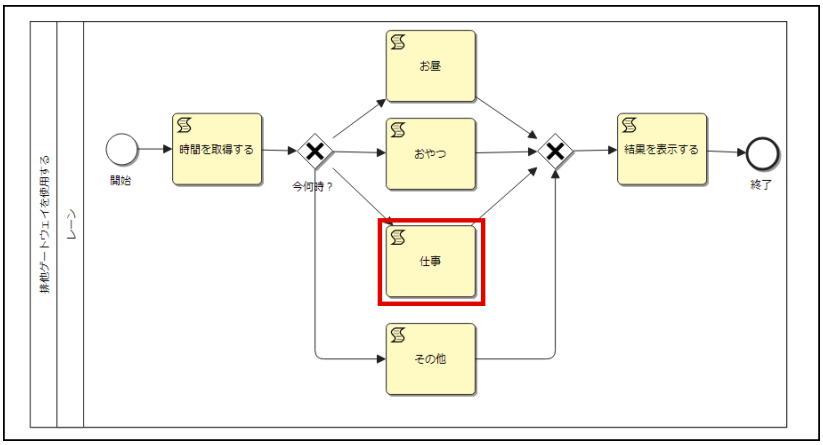
図：スクリプトタスク「おやつ」

スクリプトタスク「おやつ」のプロパティにて、作業名 `おやつ` を変数 `todo` に格納するスクリプトを設定します。

```
function run(variables, execution, entity) {
  entity.setVariable('todo', 'おやつ');
}
```

図：スクリプトタスク「おやつ」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

9. スクリプトタスク「仕事」 のプロパティを設定します。



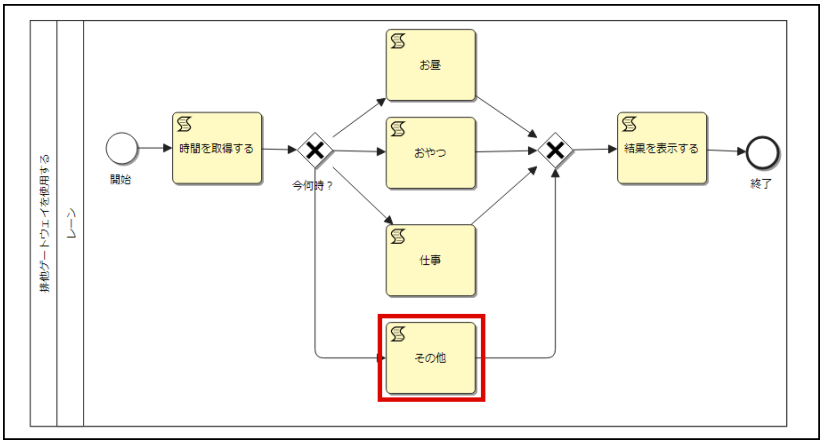
図：スクリプトタスク「仕事」

スクリプトタスク「仕事」のプロパティにて、作業名 `仕事` を変数 `todo` に格納するスクリプトを設定します。

```
function run(variables, execution, entity) {
  entity.setVariable('todo', '仕事');
}
```

図：スクリプトタスク「おやつ」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

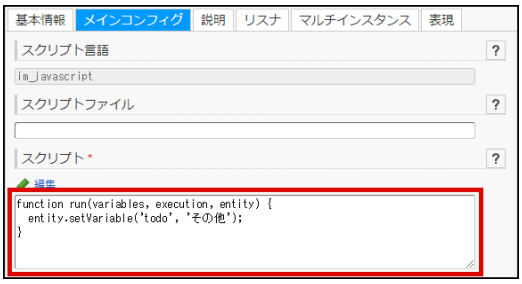
10. スクリプトタスク「その他」 のプロパティを設定します。



図：スクリプトタスク「その他」

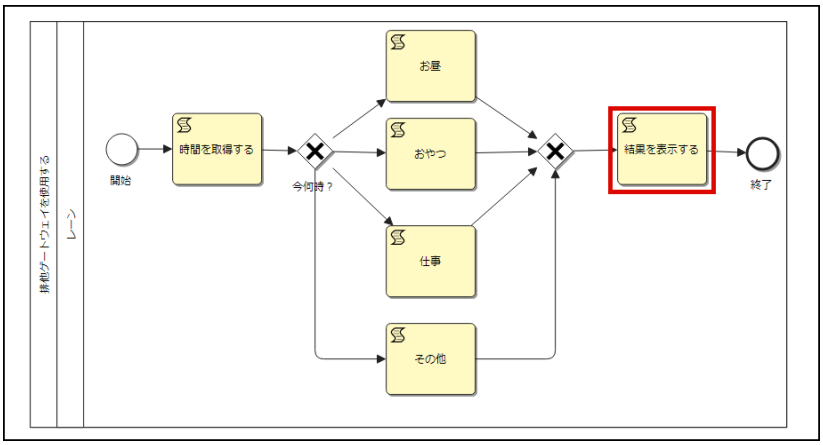
スクリプトタスク「その他」のプロパティにて、作業名 `その他` を変数 `todo` に格納するスクリプトを設定します。

```
function run(variables, execution, entity) {
  entity.setVariable('todo', 'その他');
}
```



図：スクリプトタスク「その他」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

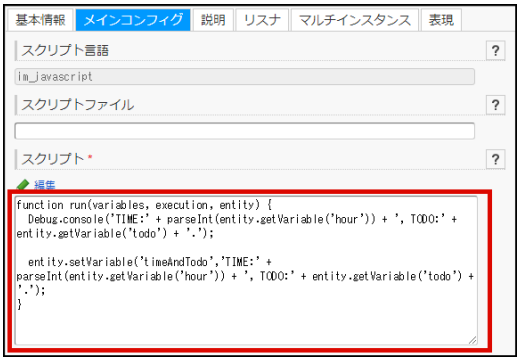
11. スクリプトタスク「結果を表示する」のプロパティを設定します。



図：スクリプトタスク「結果を表示する」

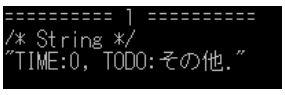
スクリプトタスク「結果を表示する」に、変数 hour と変数 todo を組み合わせ、変数 timeAndTodo に格納するスクリプトを設定します。

```
function run(variables, execution, entity) {
  Debug.console('TIME:' + parseInt(entity.getVariable('hour')) + ', TODO:' + entity.getVariable('todo') + '.');

  entity.setVariable('timeAndTodo','TIME:' + parseInt(entity.getVariable('hour')) + ', TODO:' + entity.getVariable('todo') + '.');
}
```

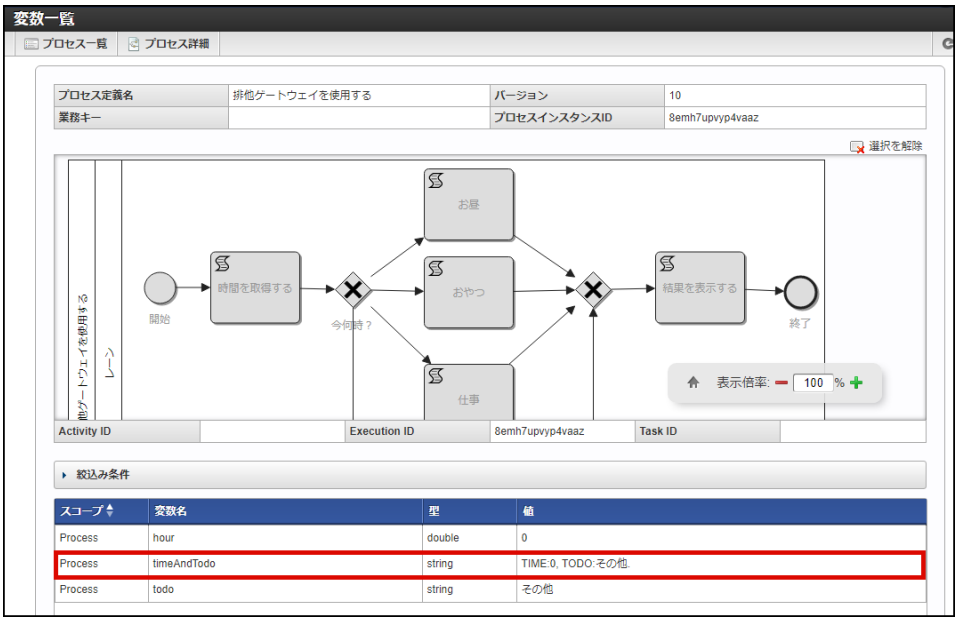


図：スクリプトタスク「結果を表示する」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

12. 実行結果を確認します。
    上記で作成したプロセスを実行環境にデプロイし、実行した結果の確認を行います。



図：結果表示



図：変数一覧

> **コラム**
>
> 排他ゲートウェイ
>
> 排他ゲートウェイでは複数のシーケンスフローの条件が真となる場合、最初に条件が真となるシーケンスフローへ遷移を行います。

> **コラム**
>
> デフォルトフロー
>
> 排他ゲートウェイでは、シーケンスフローの条件が真となるものが存在しない場合、実行の際にエラーが発生します。そのため、シーケンスフローの条件が真となるものが存在しない可能性がある場合、デフォルトフローを設定するようにしてください。
> デフォルトフローの詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「ゲートウェイ」 - 「デフォルトフロー」もあわせて参照してください。

包括ゲートウェイ

包括ゲートウェイを使用する

このチュートリアルでは、包括ゲートウェイを使用して、プロセスの遷移先の選択や合流の定義方法を解説します。
排他ゲートウェイとの違いは、trueと評価されるシーケンスフローに対して1つの分岐先に決定せず、同時並行で次の処理に進めることが可能な点です。
包括ゲートウェイの詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「ゲートウェイ」 - 「包括ゲートウェイ」もあわせて参照してください。

プロセスを進めていく中で、「IM-FormaDesigner」を使用します。
チュートリアルを開始する前に「IM-FormaDesigner」で作成したアプリケーションをインポートしてください。
im_forma_designer-inclusive_gateway-bus_trip_expenses_form.zip
im_forma_designer-inclusive_gateway-bus_trip_expenses_approval.zip
im_forma_designer-inclusive_gateway-bus_trip_expenses_confirm.zip

また、以下のサンプルユーザに対し、「IM-BPMユーザ」ロールの付与を行ってください。
「サイトマップ」→「共通マスタ」→「ユーザ」から以下のユーザを検索してください。
「ロール」タブ にて、「IM-BPMユーザ」ロールの追加を行ってください。

- 「生田一哉」（ユーザコード：ikuta）
- 「上田辰男」（ユーザコード：ueda）
- 「 萩本順子」（ユーザコード：hagimoto）

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **inclusive_gateway_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

コラム

「IM-FormaDesigner」で作成したアプリケーションのインポート方法は以下を参照してください。

「IM-FormaDesigner」で作成したアプリケーション：「IM-FormaDesigner 作成者操作ガイド」-「インポート・エクスポートを利用した IM-FormaDesigner のアプリケーションやデータソース定義の移行」

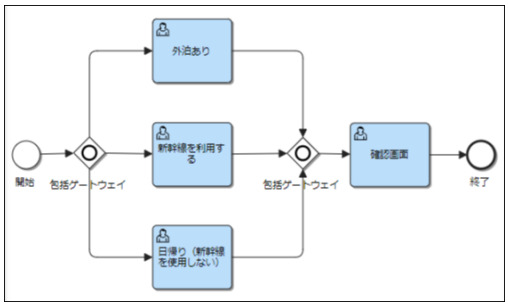
- プロセス定義を作成する
- 結果を確認する

プロセス定義を作成する

下記の図は、出張経費の承認申請に対し、該当するタスクへフローが分岐するプロセスです。
以下のそれぞれの条件に当てはまるフローへ進行します。

- 出張経費に「外泊費」を含めて申請する
- 出張経費に「新幹線の交通費」を含めて申請する
- 上記二つが当てはまらない申請をする

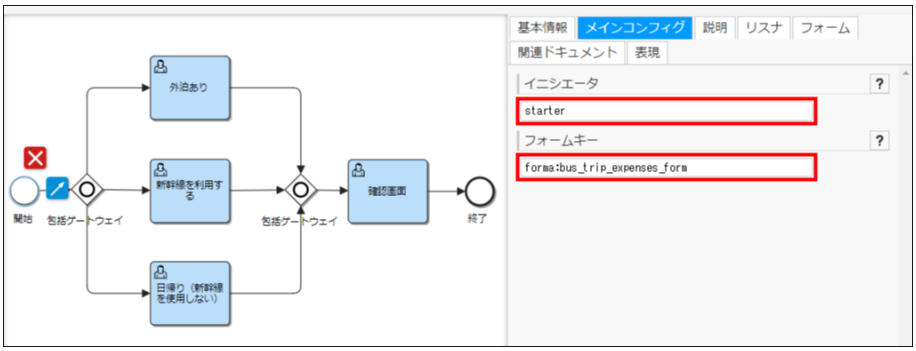
「ユーザタスク」の処理画面は、インポートした「IM-FormaDesigner」のアプリケーションを使用します。
プロセス開始時の「申請画面」で選択した情報と、シーケンスフローに設定した条件で「包括ゲートウェイ」から分岐します。
条件に当てはまらない場合、包括ゲートウェイの「デフォルトフロー」の設定により、一番下のタスクへ進みます。
必要な承認結果が揃った後で、プロセスを開始したユーザが「確認画面」のタスクで確認します。



図：完成イメージ

1. 「開始イベント」 を設置します。

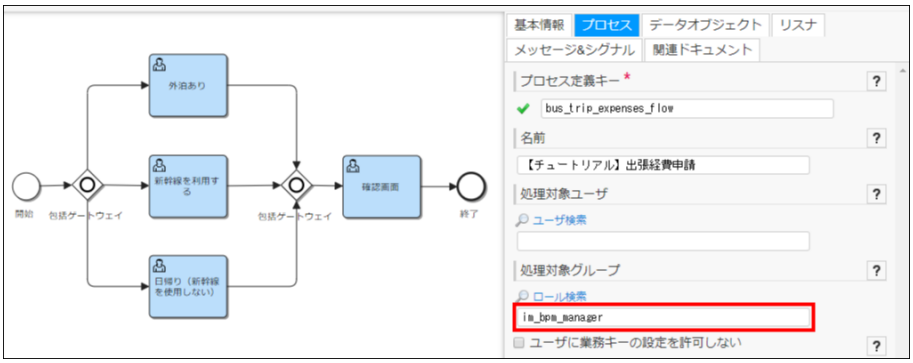
2. プロセス開始時に表示する「申請画面」のアプリーケーション画面を設定します。
   申請したユーザを「確認画面」のタスクの担当者に設定するために、「イニシエータ」を設定しておきます。
   「開始イベント」の「メインコンフィグ」から、以下のとおりに項目を設定します。
   - イニシエータ ： starter
   - フォームキー ： forma:bus_trip_expenses_form



図：「開始イベント」 -「プロパティ」 -「メインコンフィグ」

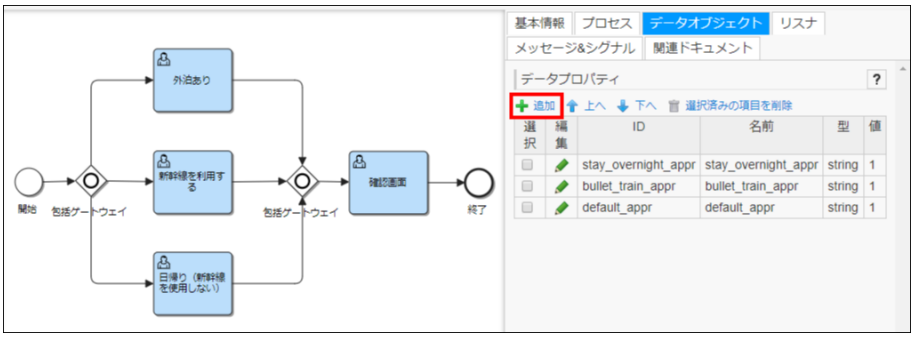
3. プロセス全体に対する設定を行います。
   「開始イベント」ではない場所をクリックし、「プロパティ」をプロセス全体に切り替えます。

4. 「プロセス」タブの「処理対象グループ」に「 im_bpm_manager 」を設定します。



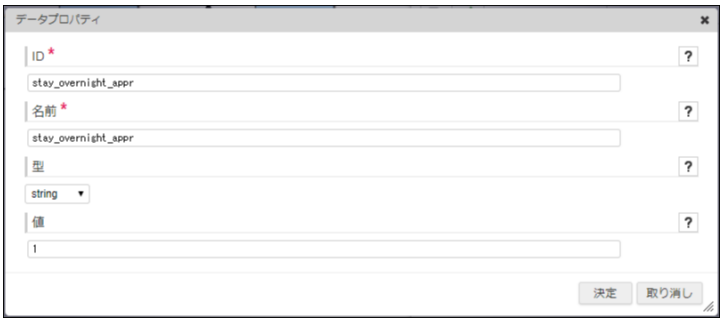
図：プロセス全体 - 「プロパティ」 - 「プロセス」

5. 承認結果の初期値を設定します。
「データオブジェクト」タブから、「データプロパティ」の「追加」をクリックします。



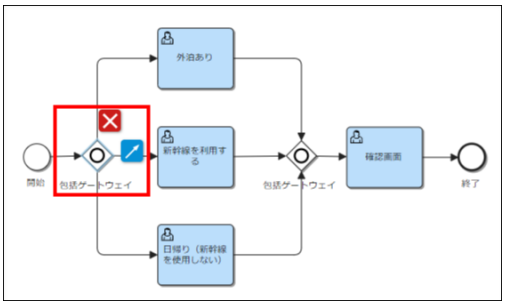
図：プロセス全体 - 「プロパティ」- 「データオブジェクト」

6. 「データプロパティ」ダイアログから、以下のとおりに3つデータプロパティを作成します。
   - 「外泊承認結果」
     - ID：stay_overnight_appr
     - 名前：stay_overnight_appr
     - 型： string
     - 値：1
   - 「新幹線使用承認結果」
     - ID：bullet_train_appr
     - 名前：bullet_train_appr
     - 型： string
     - 値：1
   - 「デフォルト承認結果」
     - ID：default_appr
     - 名前：default_appr
     - 型： string
     - 値：1



図：「データプロパティ」

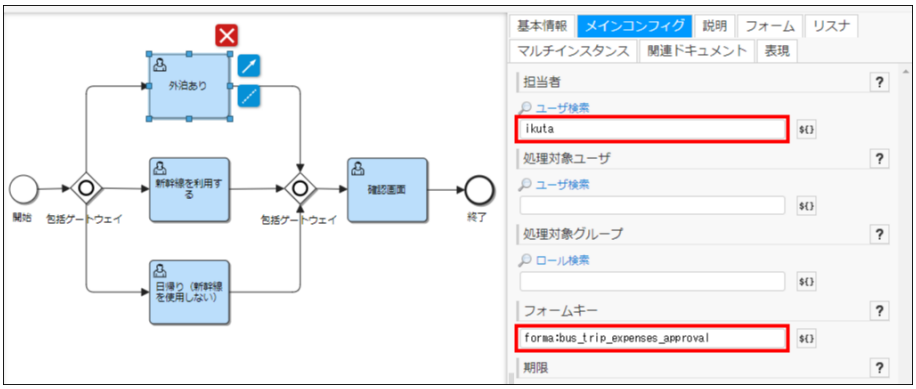
7. 「包括ゲートウェイ」を設置します。



図：「包括ゲートウェイ」

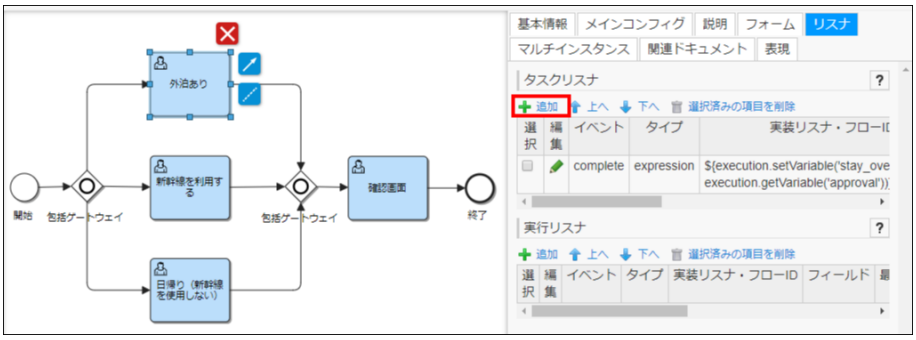
8. 「外泊あり」の出張経費申請の承認を「生田」に対して依頼するタスクを作成します。
「承認画面」の「ユーザタスク」を設置します。

9. 「ユーザタスク」を選択し、「メインコンフィグ」タブで以下のとおりに項目を設定します。
   - 担当者：ikuta
   - フォームキー：forma:bus_trip_expenses_approval



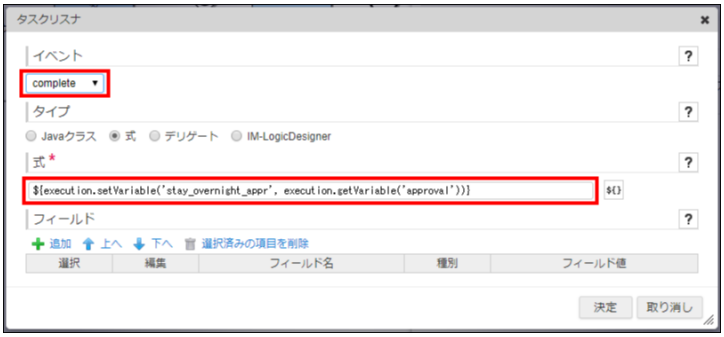
図：「ユーザタスク」 - 「プロパティ」- 「メインコンフィグ」

10. 「承認画面」で選択された承認結果を「確認画面」で判定するため、「データオブジェクト」で用意した変数「stay_overnight_appr」を上書きします。
「ユーザタスク」の「リスナ」タブから、「タスクリスナ」の追加をクリックします。



図：「ユーザタスク」 - 「プロパティ」- 「リスナ」

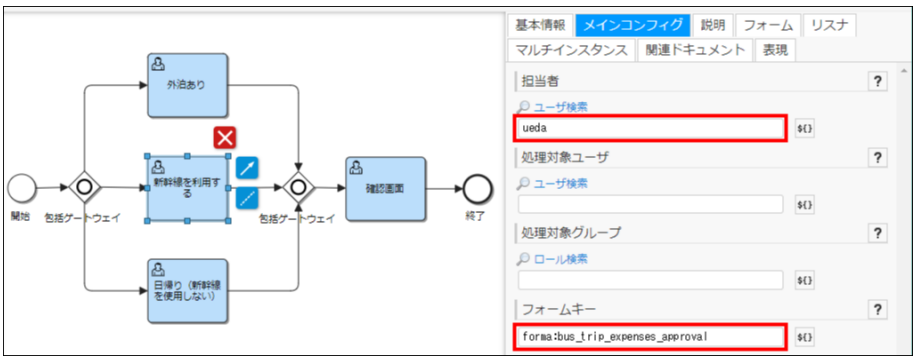
11. 「タスクリスナ」ダイアログで、以下のとおりに項目を設定します。
   - イベント ： complete
   - タイプ ： 式
   - 式 ： ${execution.setVariable('stay_overnight_appr', execution.getVariable('approval'))}



図：「タスクリスナ」

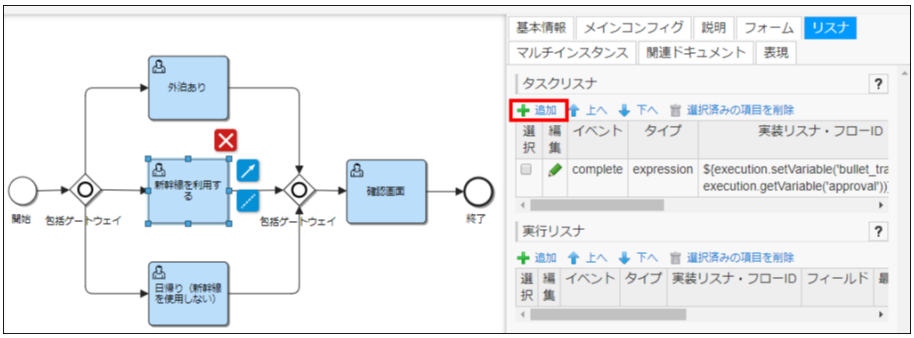
12. 上記の「外泊あり」と同じ手順で、「新幹線を利用する」が選択された場合に「上田」に対して承認を依頼するタスクを作成します。
「承認画面」の「ユーザタスク」を設置します。

13. 「ユーザタスク」を選択し、「メインコンフィグ」タブで以下のとおりに項目を設定します。
    - 担当者：ueda
    - フォームキー：forma:bus_trip_expenses_approval



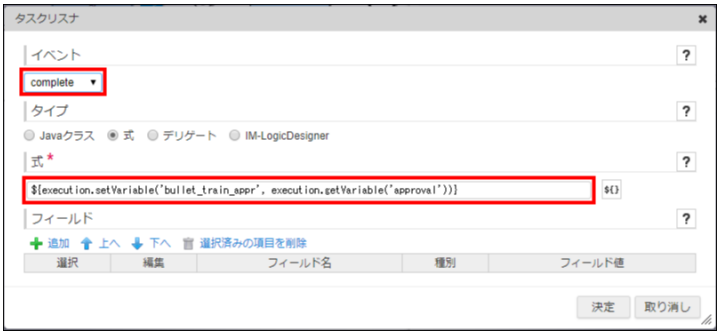
図：「ユーザタスク」 -「プロパティ」-「メインコンフィグ」

14. 「承認画面」で選択された評価結果を「確認画面」で判定するため、「データオブジェクト」で用意した変数「bullet_train_appr」に上書きします。
    「ユーザタスク」の「リスナ」タブから、「タスクリスナ」の「追加」をクリックします。



図：「ユーザタスク」 -「プロパティ」-「リスナ」

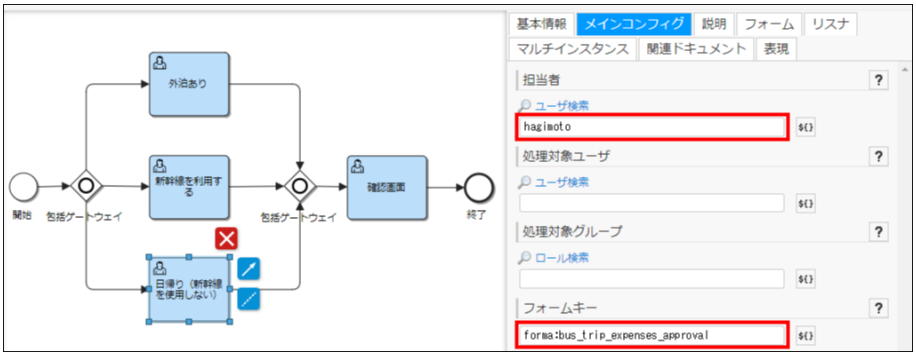
15. 「タスクリスナ」ダイアログで、以下のとおりに項目を設定します。
    - イベント ： complete
    - タイプ ： 式
    - 式 ： ${execution.setVariable('bullet_train_appr', execution.getVariable('approval'))}



図：「タスクリスナ」

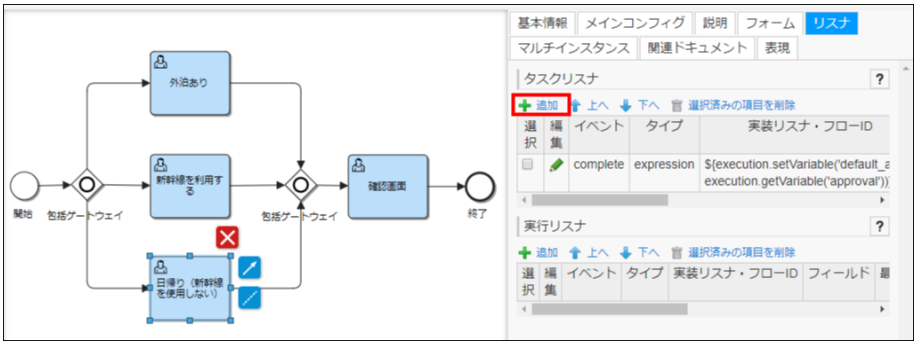
16. 「外泊あり」と「新幹線を利用する」が選択されなかった申請の場合、「萩本」に対して承認を依頼するタスクを配置します。
    「承認画面」の「ユーザタスク」を設置します。

17. 「ユーザタスク」を選択し、「メインコンフィグ」タブで以下のとおりに項目を設定します。
    - 担当者：hagimoto
    - フォームキー：forma:bus_trip_expenses_approval



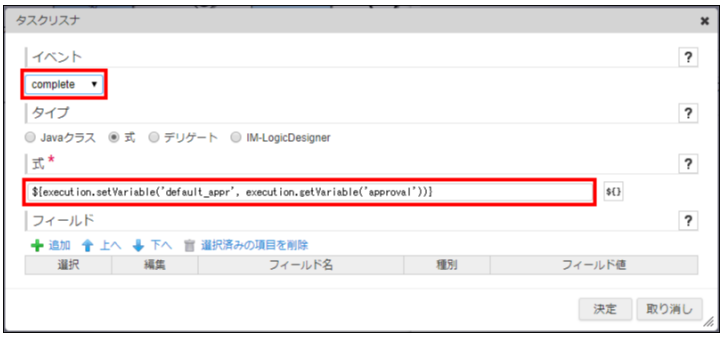
図：「ユーザタスク」 -「プロパティ」-「メインコンフィグ」

18. 「承認画面」で選択された評価結果を「確認画面」で判定するため、「データオブジェクト」で用意した変数「default_appr」を上書きします。
「ユーザタスク」の「リスナ」タブから、「タスクリスナ」の「追加」をクリックします。



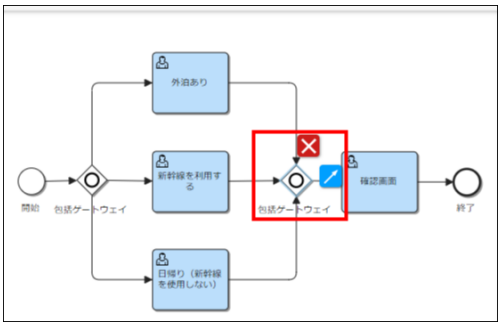
図：「ユーザタスク」 -「プロパティ」-「リスナ」

19. 「タスクリスナ」ダイアログの項目を、以下のとおりに設定します。
    - イベント ： complete
    - タイプ ： 式
    - 式 ：${execution.setVariable('default_appr', execution.getVariable('approval'))}



図：「タスクリスナ」

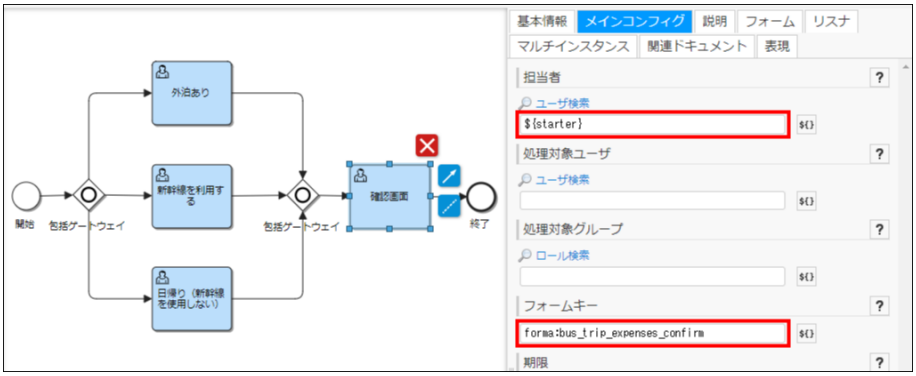
20. 他の承認の判定を待つため「包括ゲートウェイ」を設置します。



図：「包括ゲートウェイ」

21. 出張経費申請の結果を表示する「確認画面」のタスクを作成します。
「ユーザタスク」を設置します。

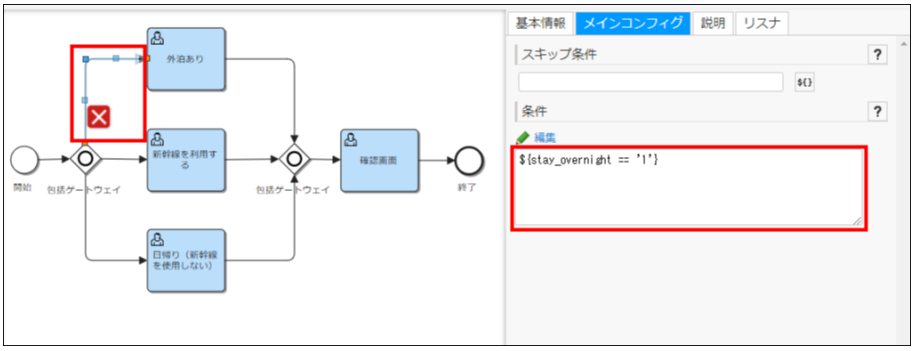
22. 開始イベントに設定した「イニシエータ」の値を使用し、申請者に対して「確認画面」を送信します。
「ユーザタスク」を選択し、「プロパティ」の「メインコンフィグ」から以下のとおりに項目を設定します。
    - 担当者 ： ${starter}
    - フォームキー ： forma:bus_trip_expenses_confirm



図：「ユーザタスク」 -「プロパティ」 -「メインコンフィグ」

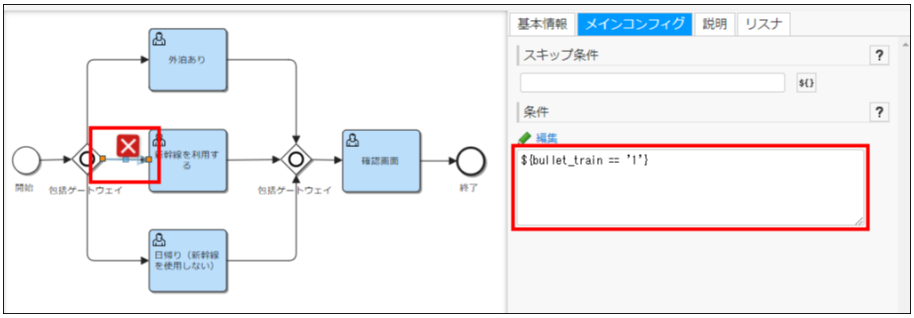
23. 終了イベントを設置します。

24. 「申請画面」で選択された情報をもとに、フローを進める条件を設定します。
「外泊あり」のタスクに繋がる「シーケンスフロー」を選択し、「メインコンフィグ」の「条件」に「`${stay_overnight == '1'}`」を設定します。



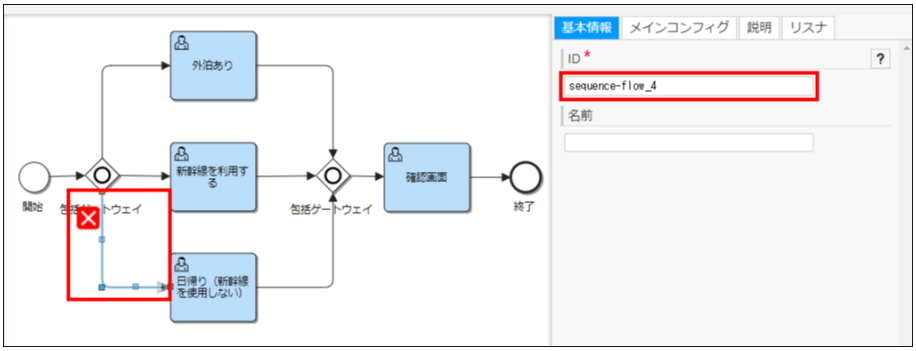
図：「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」

25. 「新幹線を使用する」のタスクに繋がる「シーケンスフロー」を選択し、「メインコンフィグ」の「条件」に「`${bullet_train == '1'}`」を設定します。



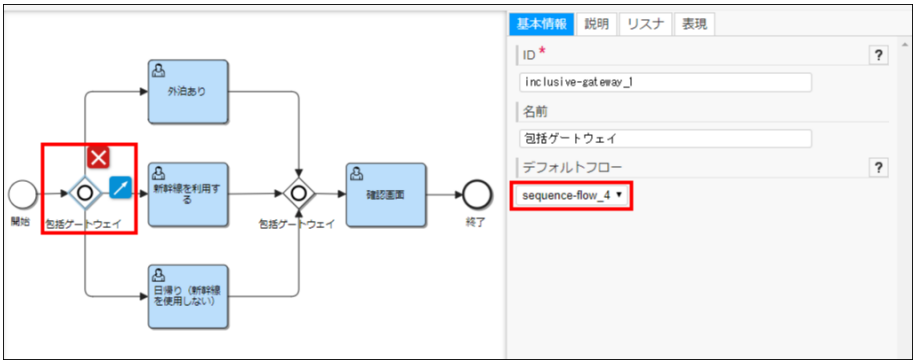
図：「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」

26. 「外泊あり」と「新幹線を利用」が選択されなかった場合、「日帰り（新幹線を使用しない）」へ進行するよう設定します。
「日帰り（新幹線を使用しない）」へ続くシーケンスフローをクリックし、「基本情報」タブからIDを確認します。



図：「シーケンスフロー」 - 「プロパティ」 - 「基本情報」

27. 「包括ゲートウェイ」を選択し、「基本情報」の「デフォルトフロー」から、上記で確認したIDを選択します。



図：「包括ゲートウェイ」 - 「プロパティ」 - 「基本情報」

結果を確認する

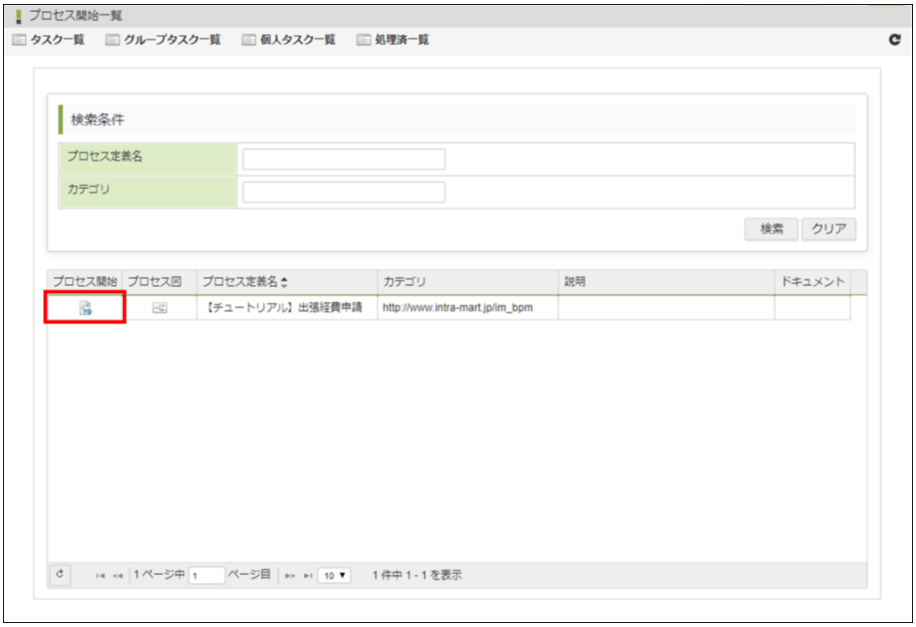
このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行結果の確認を行います。

デフォルトフローの結果を確認する

「包括ゲートウェイ」から、デフォルトフローへ進行する場合を確認します。

1. 「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示します。

2. プロセス開始一覧から、「  」アイコンをクリックします。



図：「プロセス開始一覧」

3. 「IM-FormaDesigner」で作成したアプリケーション画面、「申請画面」が表示されます。

4. 今回は包括ゲートウェイの「デフォルトフロー」へ進行したいので、「外泊あり」と「新幹線を利用する」のチェックボックスは入れません。
適当な値を入力し、「申請」ボタンをクリックします。

図：申請画面（IM-FormaDesigner）

5. 「日帰り（新幹線を利用しない）」の申請を承認します。
「日帰り（新幹線を利用しない）」の担当者である「萩本」でログインします。

6. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。

個人タスクに振り分けられている「日帰り（新幹線を利用しない）」の「  」をクリックします。

図：「タスク一覧」 -「個人タスク」

7. 「IM-FormaDesigner」で作成したアプリケーション画面、「承認画面」が表示されます。

8. ラジオボタンの「承認する」を選択し、「送信」ボタンをクリックします。

図：承認画面（IM-FormaDesigner）

9. 承認結果を確認します。
   プロセスを開始したユーザでログインします。

10. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
    個人タスクに振り分けられている「確認画面」の「  」をクリックします。

図：「タスク一覧」 - 「個人タスク」

11. 「IM-FormaDesigner」で作成したアプリケーション画面、「確認画面」が表示されます。
    「萩本」の承認が下りたため、出張経費申請が承認されました。

図：確認画面（IM-FormaDesigner）

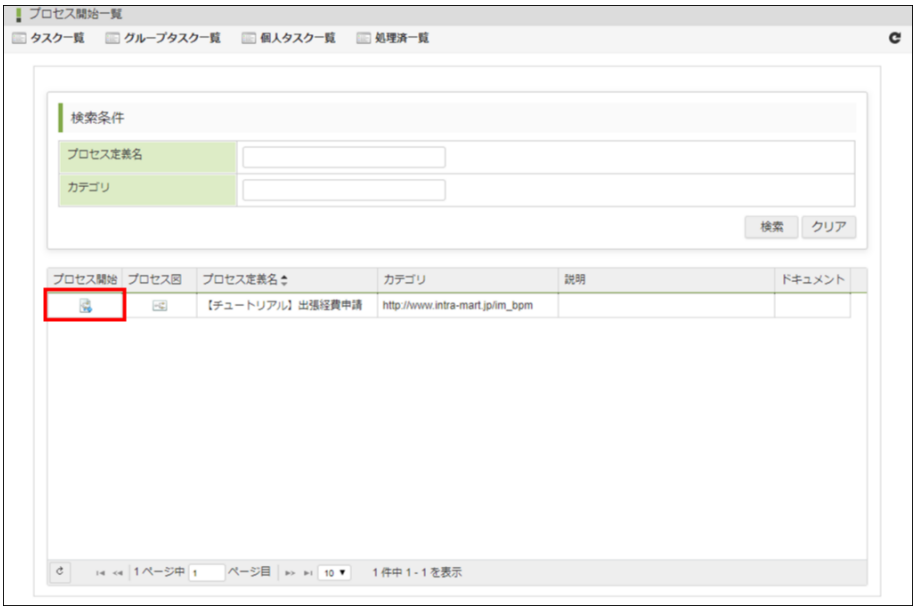
12. デフォルトフローに進行した場合、ほかのタスクは処理されません。
    「外泊あり」の担当者である「生田」、「新幹線を利用する」の担当者である「上田」の個人タスク一覧に、依頼がないことを確認します。

条件分岐による複数の承認・否認の結果を確認する

シーケンスフローの条件に該当し、「包括ゲートウェイ」から複数分岐する場合を確認します。

1. 「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示します。

2. プロセス開始一覧から、「  」アイコンをクリックします。



図：「プロセス開始一覧」

3. 「IM-FormaDesigner」で作成したアプリケーション画面、「申請画面」が表示されます。

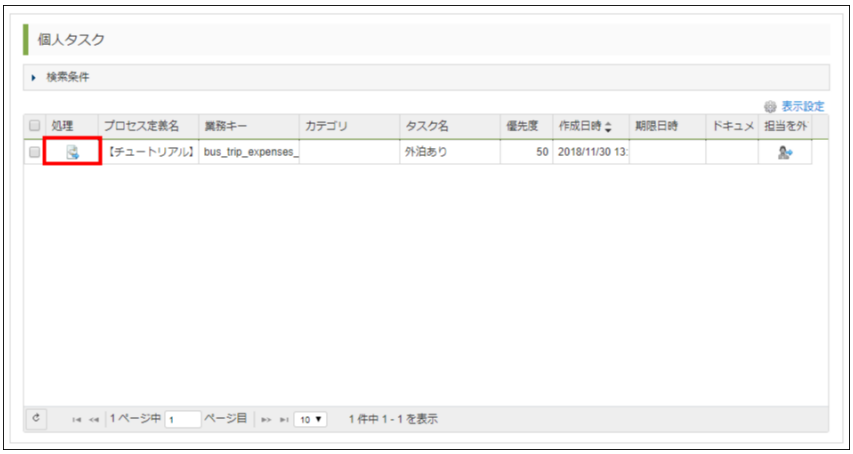
4. 適当な値を入力し、「申請」ボタンをクリックします。
   今回は「外泊あり」と「新幹線を利用する」を選択した状態で申請します。

図：申請画面（IM-FormaDesigner）

5. 「外泊あり」の申請を否認します。
   「外泊あり」の承認依頼先である「生田」でログインします。

6. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。

7. 個人タスクに振り分けられている「外泊あり」の「  」をクリックします。



図：「タスク一覧」 - 「個人タスク」

8. 「IM-FormaDesigner」で作成したアプリケーション画面、「承認画面」が表示されます。

9. ラジオボタンの「否認する」を選択し、「送信」ボタンをクリックします。

図：承認画面（IM-FormaDesigner）

10. 「新幹線を利用する」の申請を承認します。
    「新幹線を利用する」の承認依頼先である「上田」でログインします。

11. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。

12. 個人タスクに振り分けられている「新幹線を使用する」の「  」をクリックします。

図：「タスク一覧」 - 「個人タスク」

13. 「IM-FormaDesigner」で作成したアプリケーション画面、「承認画面」が表示されます。

14. ラジオボタンの「承認する」を選択し、「送信」ボタンをクリックします。

図：承認画面（IM-FormaDesigner）

15. 承認結果を確認します。
    プロセスを開始したユーザでログインします。

16. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。

    個人タスクに振り分けられている「確認画面」の「  」をクリックします。

図：「タスク一覧」 - 「個人タスク」

17. 「IM-FormaDesigner」で作成したアプリケーション画面、「確認画面」が表示されます。
「新幹線を利用する」タスクの担当者である「上田」は承認しましたが、「外泊あり」タスクの担当者である「生田」が否認したため、結果が否認になりました。

図：確認画面（IM-FormaDesigner）

18. 他のシーケンスフローの条件に一致したため、「デフォルトフロー」である「日帰り（新幹線を利用しない）」のタスクは実行されません。
「日帰り（新幹線を利用しない）」の担当者である「萩本」でログインし、個人タスク一覧に承認依頼のタスクがないことを確認します。

イベント

シグナルイベントを使用する

このチュートリアルでは、シグナルイベントを使用して、シグナルの送信とシグナルの受信の定義方法を解説します。
シグナルは、シグナル名ごとに受信を待機しているプロセスに一斉に送信（ブロードキャスト）し、シグナル名での受信を待機している全てのプロセスを進めます。

シグナルを送受信するイベントは、以下のとおりです。

シグナルを送信するイベント

- シグナルスローイベント

シグナルを受信するイベント

- シグナル開始イベント
- シグナル境界イベント
- シグナルキャッチイベント

> **コラム**
>
> 各イベントの詳細については、以下を参照してください。
> シグナルスローイベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「シグナルスローイベント」
> シグナル開始イベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「シグナル開始イベント」
> シグナル境界イベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「シグナル境界イベント」
> シグナルキャッチイベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「シグナルキャッチイベント」

コラム

このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
event_signal_usage.bpmn

このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

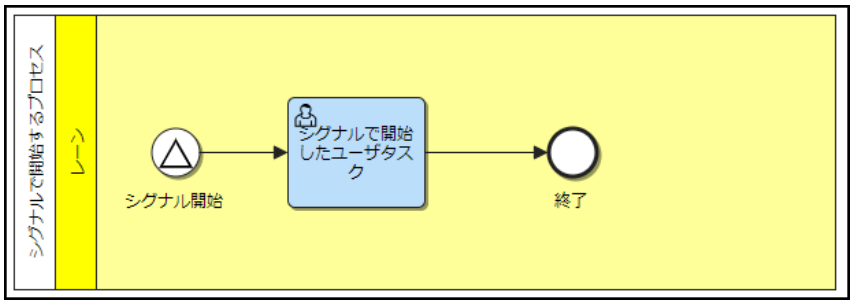
コラム

このチュートリアルのサンプルでは、同一ファイル内に「シグナルを受信するプロセス定義」と「シグナルを送信するプロセス定義」が定義されていますが、それぞれファイルを分けて定義することも可能です。

- シグナルを受信することにより開始するプロセス定義を作成する
- シグナルを受信することによりユーザタスクを中断するプロセス定義を作成する
- シグナルを受信することにより先に進むプロセス定義を作成する
- シグナルを送信するプロセス定義を作成する
- 実行結果を確認する

シグナルを受信することにより開始するプロセス定義を作成する

シグナルを受信することにより開始するプロセス定義を作成します。

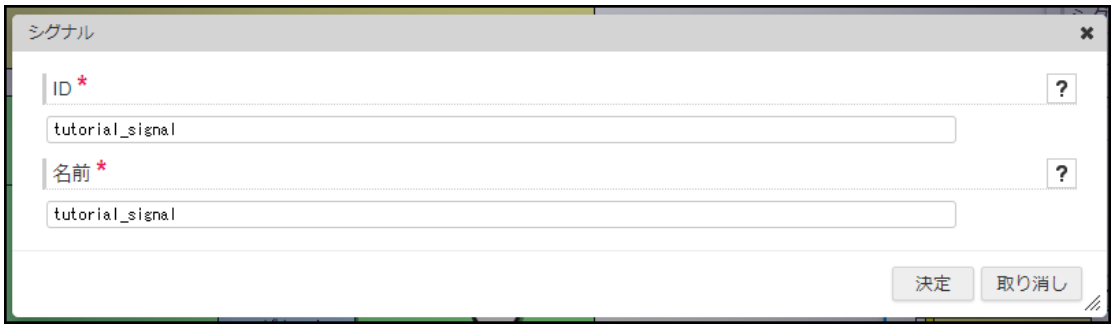


図：完成イメージ

1. 「シグナル」を登録します。
    1. 「プロセス」-「プロパティ」-「メッセージ＆シグナル」-「シグナル」の「追加」リンクをクリックします。

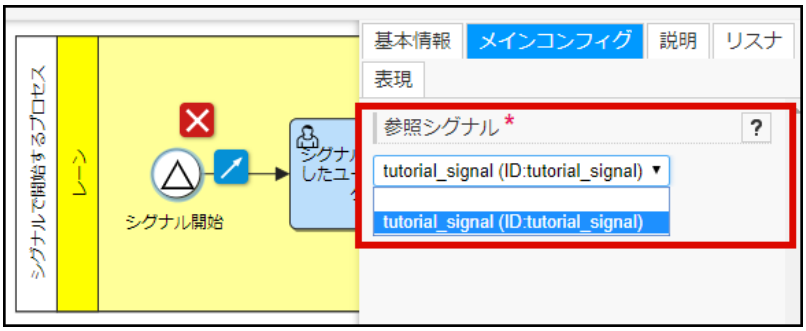
図：「プロセス」-「プロパティ」-「メッセージ＆シグナル」-「シグナル」

    2. 「シグナル」の「ID」と「名前」を設定します。
        - ID：tutorial_signal
        - 名前：tutorial_signal



図：「シグナル」

2. 「シグナル開始イベント」を配置します。
3. 「シグナル開始イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照シグナル」のプルダウンリストのtutorial_signal を選択します。



図：「シグナル開始イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照シグナル」

4. 「プロセス」の「プロセス定義キー」を設定します。
   - プロセス定義キー：event_signal_usage_start_process

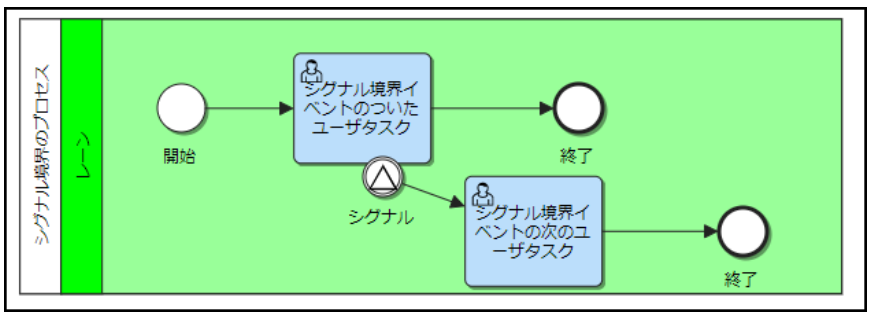
図：「プロパティ」 - 「プロセス」

5. 「シグナル開始イベント」で開始されたことを確認するために「ユーザタスク」を配置します。
   - 担当者：aoyagi

図：「ユーザタスク」

シグナルを受信することによりユーザタスクを中断するプロセス定義を作成する

シグナルを受信することによりユーザタスクを中断するプロセス定義を作成します。



図：完成イメージ

1. 「シグナル」を登録します。前項と同一ファイル内でプロセス定義を作成している場合は行う必要はありません。

   登録方法は「*シグナルを受信することにより開始するプロセス定義を作成する*」を参照してください。

2. 「開始イベント」を配置します。

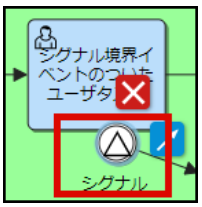
3. 「プロセス」の「プロセス定義キー」と「処理対象ユーザ」を設定します。
   - プロセス定義キー：event_signal_usage_boundary_process
   - 処理対象ユーザ：aoyagi

図：「プロパティ」 - 「プロセス」

4. 「ユーザタスク」を配置します。
   - 担当者：aoyagi

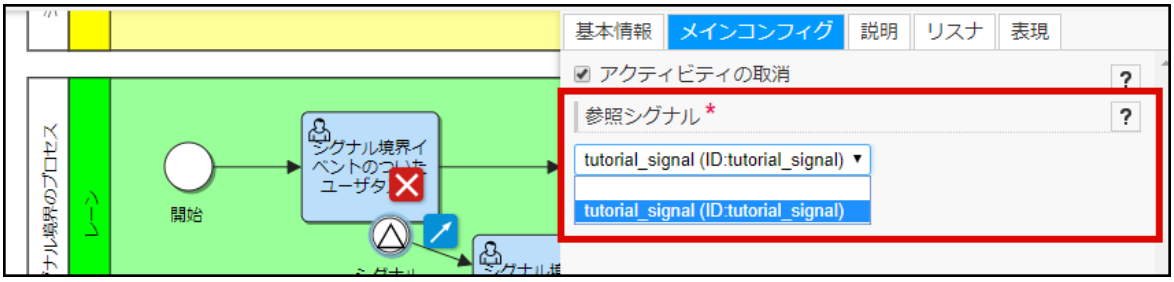
図：「ユーザタスク」

5. 「シグナル境界イベント」を「ユーザタスク」に配置します。



図：「シグナル境界イベント」

6. 「シグナル境界イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照シグナル」のプルダウンリストのtutorial_signal を選択します。



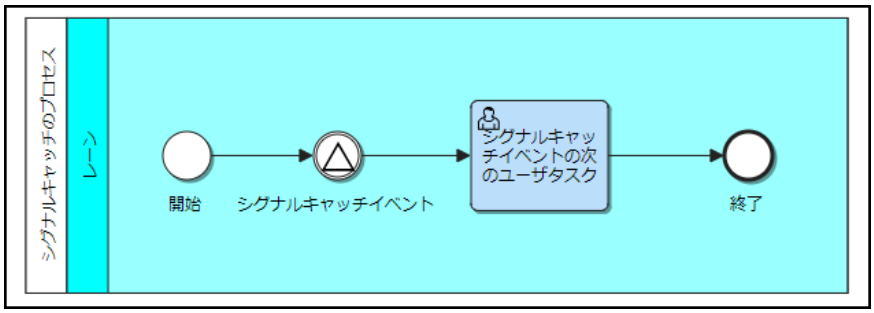
図：「シグナル境界イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照シグナル」

7. 「シグナル境界イベント」で「ユーザタスク」が中断された確認するために別の「ユーザタスク」を配置します。
   - 担当者：aoyagi

図：「シグナル境界イベント」で中断された後に遷移する「ユーザタスク」

シグナルを受信することにより先に進むプロセス定義を作成する

シグナルを受信することにより先に進むプロセス定義を作成します。



図：完成イメージ

1. 「シグナル」を登録します。前項と同一ファイル内でプロセス定義を作成している場合は行う必要はありません。

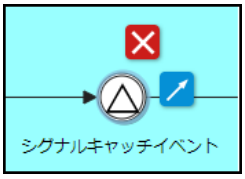
   登録方法は「*シグナルを受信することにより開始するプロセス定義を作成する*」を参照してください。

2. 「開始イベント」を配置します。

3. 「プロセス」の「プロセス定義キー」と「処理対象ユーザ」を設定します。
   - プロセス定義キー：event_signal_usage_intermediate_process
   - 処理対象ユーザ：aoyagi

図：「プロパティ」 - 「プロセス」

4. 「シグナルキャッチイベント」を配置します。



図：「シグナルキャッチイベント」

5. 「シグナルキャッチイベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照シグナル」のプルダウンリストのtutorial_signal を選択します。

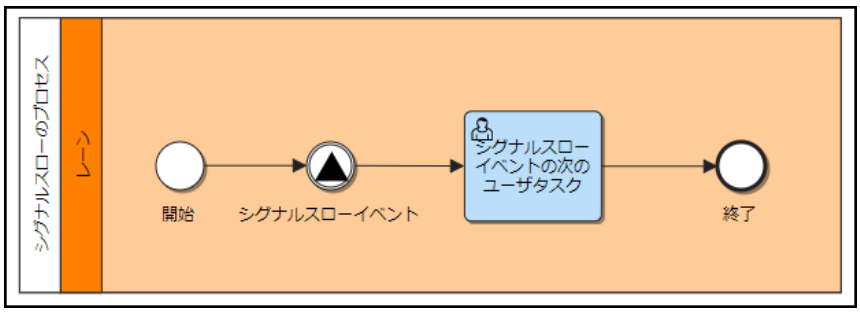
図：「シグナルキャッチイベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照シグナル」

6. 「シグナルキャッチイベント」から次に進んだことを確認するために「ユーザタスク」を配置します。
   - 担当者：aoyagi

図：「ユーザタスク」

シグナルを送信するプロセス定義を作成する

シグナルを送信するプロセス定義を作成します。



図：完成イメージ

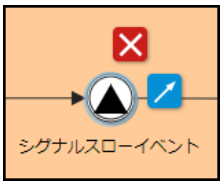
1. 「シグナル」を登録します。前項と同一ファイル内でプロセス定義を作成している場合は行う必要はありません。

   登録方法は「*シグナルを受信することにより開始するプロセス定義を作成する*」を参照してください。

2. 「開始イベント」を配置します。
3. 「プロセス」の「プロセス定義キー」と「処理対象ユーザ」を設定します。
   - プロセス定義キー：event_signal_usage_throw_process
   - 処理対象ユーザ：aoyagi

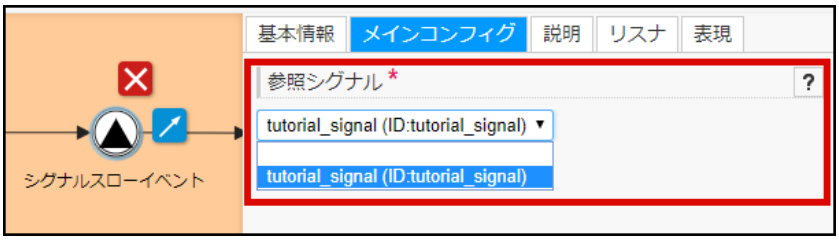
図：「プロパティ」 - 「プロセス」

4. 「シグナルスローイベント」を配置します。



図：「シグナルスローイベント」

5. 「シグナルスローイベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照シグナル」のプルダウンリストのtutorial_signal を選択します。



図：「シグナルスローイベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照シグナル」

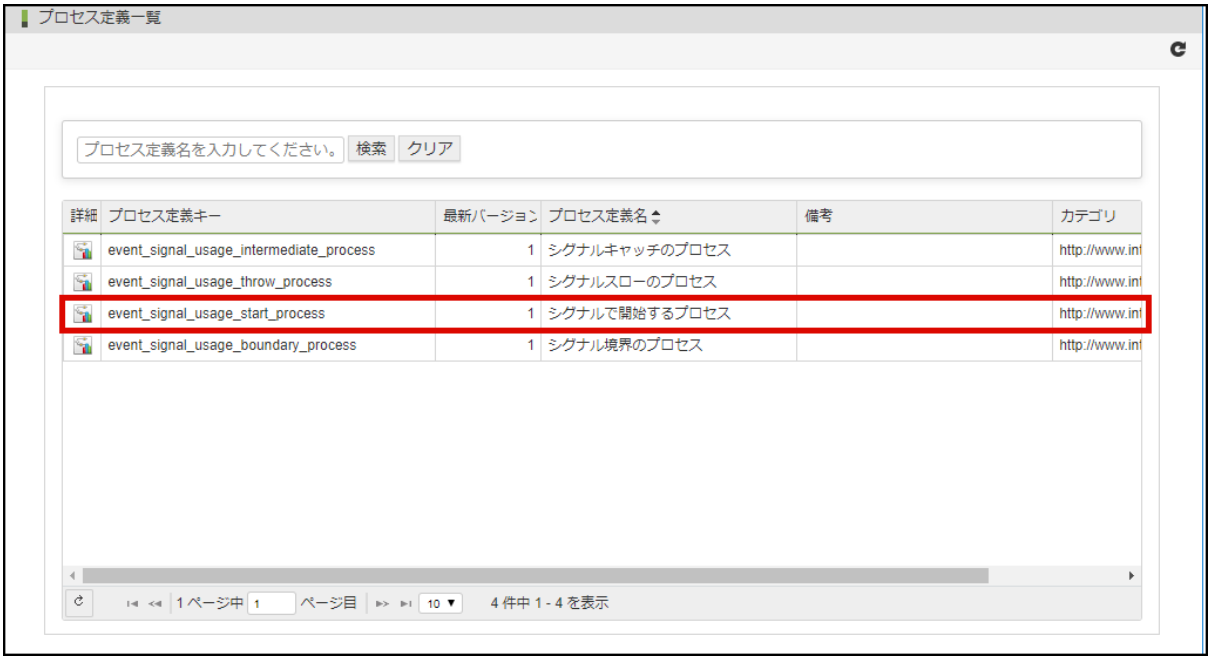
6. 「シグナルスローイベント」から次に進んだことを確認するために「ユーザタスク」を配置します。
   - 担当者 : aoyagi

図：「ユーザタスク」

実行結果を確認する

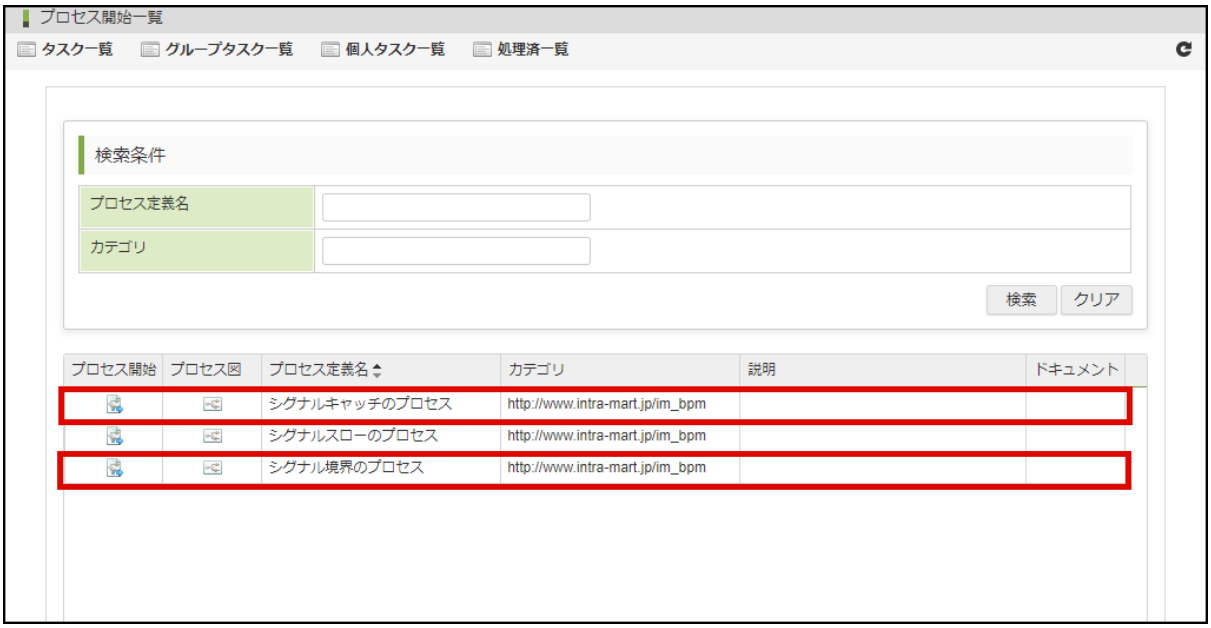
このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

1. 「プロセス定義キー」 event_signal_usage_start_process がプロセス定義として、デプロイされていることを確認します。



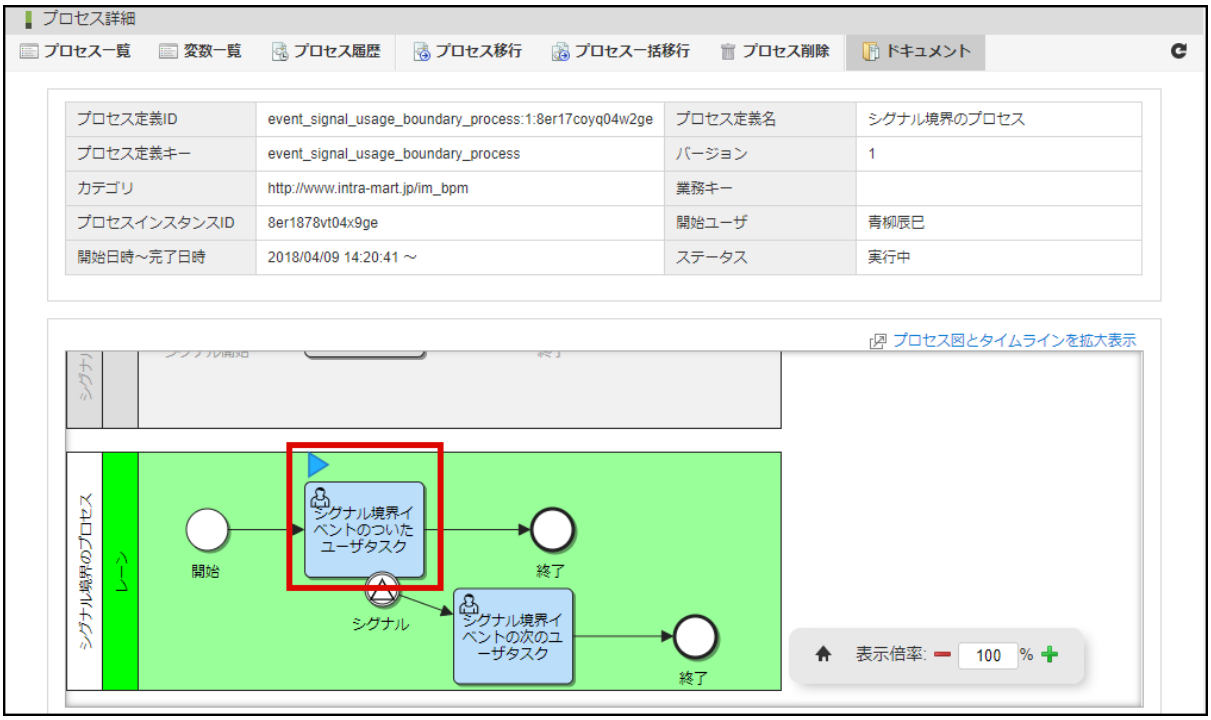
図：プロセス定義一覧 event_signal_usage_start_process

2. 「プロセス定義キー」 event_signal_usage_boundary_process と event_signal_usage_intermediate_process のプロセスを開始します。



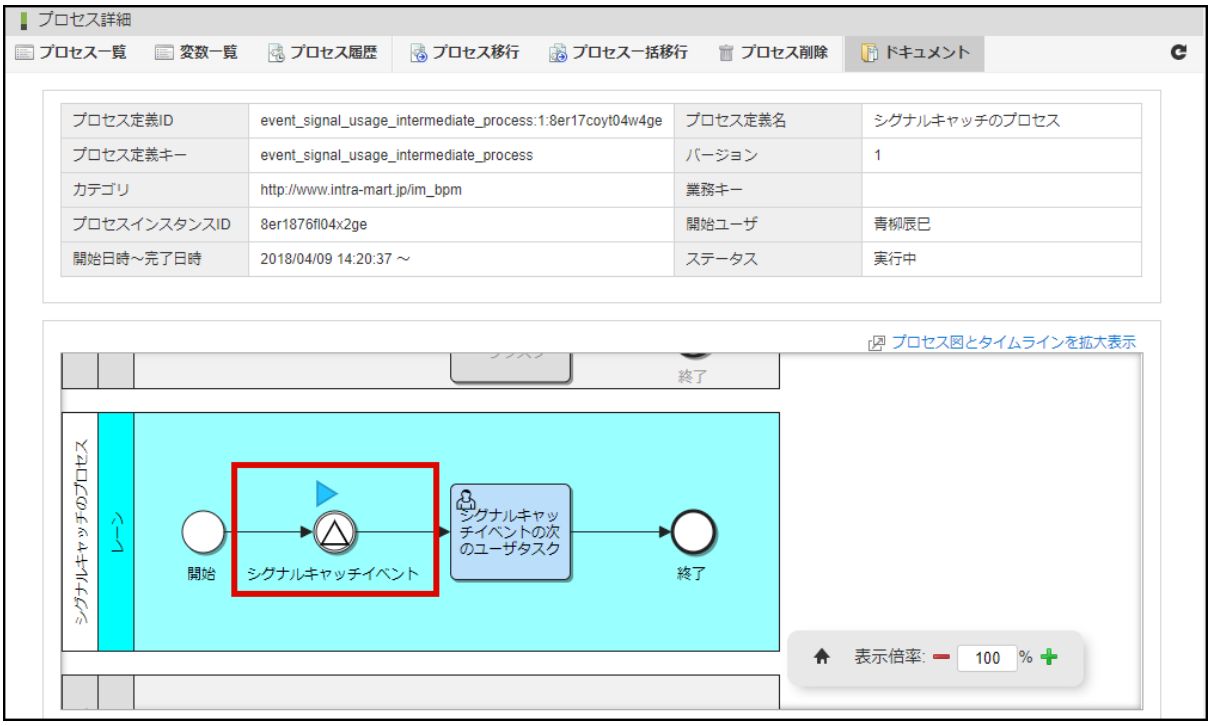
図：プロセス開始

3. 「プロセス定義キー」 event_signal_usage_boundary_process が開始され、「シグナル境界イベント」がついた「ユーザタスク」が開始されていることを確認します。



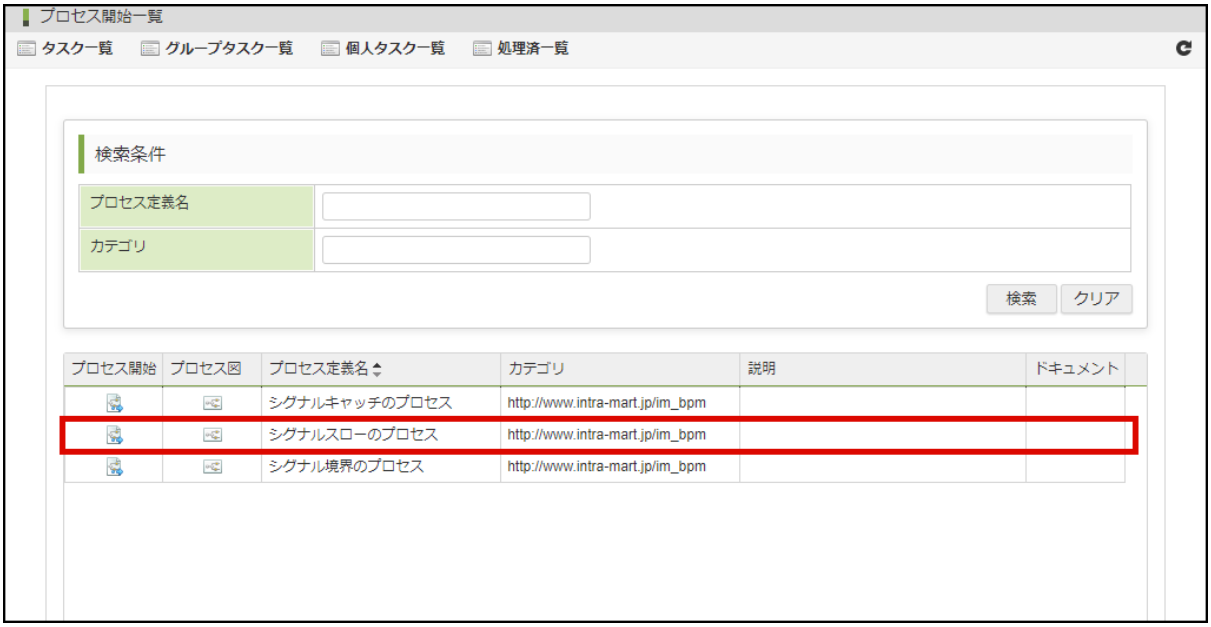
図：プロセス詳細 event_signal_usage_boundary_process

4. 「プロセス定義キー」 event_signal_usage_intermediate_process が開始され、「シグナルキャッチイベント」に滞在していることを確認します。



図：プロセス詳細 event_signal_usage_intermediate_process

5. 「プロセス定義キー」 event_signal_usage_throw_process のプロセスを開始し、シグナルを送信（ブロードキャスト）します。



図：プロセス開始 event_signal_usage_throw_process

6. それぞれのプロセスが、シグナルを受信して次のユーザタスクに進んでいることを確認します。

図：タスク一覧

メッセージイベントを使用する

このチュートリアルでは、メッセージイベントを使用して、メッセージの受信の定義方法を解説します。
メッセージは、特定の1プロセスに送信することで、そのプロセスを進めることが可能です。

メッセージを受信するイベントは、以下のとおりです。

- メッセージ開始イベント
- メッセージ境界イベント
- メッセージキャッチイベント

このチュートリアルでは、メッセージの送信に「IM-LogicDesignerタスク」を使用します。
開始する前に以下のリンクから「ロジックフロー定義」をダウンロードし、インポートしてください。
im_logicdesigner-data-message-event.zip

> **注意**
>
> このチュートリアルの「IM-LogicDesignerタスク」で使用する「ロジックフロー定義」は、IM-BPM for Accel Platform 2018 Spring(Skylark) 以降のバージョンで動作します。

> **コラム**
>
> 「ロジックフロー定義」のインポートについての詳細は「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」を参照してください。

> **コラム**
>
> 各イベントの詳細については、以下を参照してください。
>
> メッセージ開始イベント：「IM-BPM プロセスデザイナ 操作ガイド」 -「メッセージ開始イベント」
> メッセージ境界イベント：「IM-BPM プロセスデザイナ 操作ガイド」 -「メッセージ境界イベント」
> メッセージキャッチイベント：「IM-BPM プロセスデザイナ 操作ガイド」 -「メッセージキャッチイベント」
>
> IM-LogicDesignerの詳細については、「IM-LogicDesigner仕様書」を参照してください。

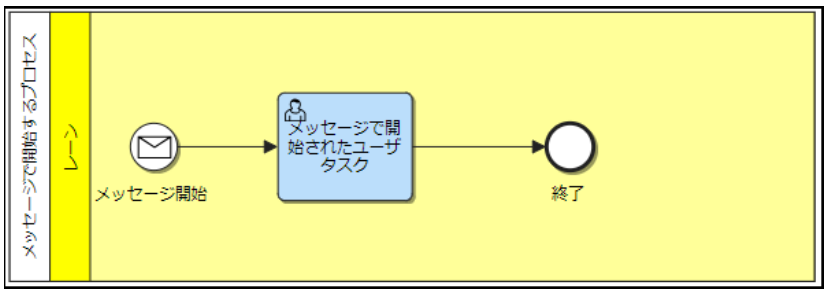
> **コラム**
>
> このチュートリアルで作成する「プロセス定義」のサンプルを、以下のリンクからダウンロードできます。
> **event_message_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

> **コラム**
>
> このチュートリアルのサンプルでは、同一ファイル内に「メッセージを送信するプロセス定義」と「メッセージを受信するプロセス定義」が定義されていますが、それぞれファイルを分けて定義することも可能です。

- メッセージを受信することにより開始するプロセス定義を作成する
- メッセージを受信することによりユーザタスクを中断するプロセス定義を作成する
- メッセージを受信することにより先に進むプロセス定義を作成する
- メッセージを送信するプロセス定義を作成する
- 実行結果を確認する

メッセージを受信することにより開始するプロセス定義を作成する

メッセージを受信することにより開始するプロセス定義を作成します。



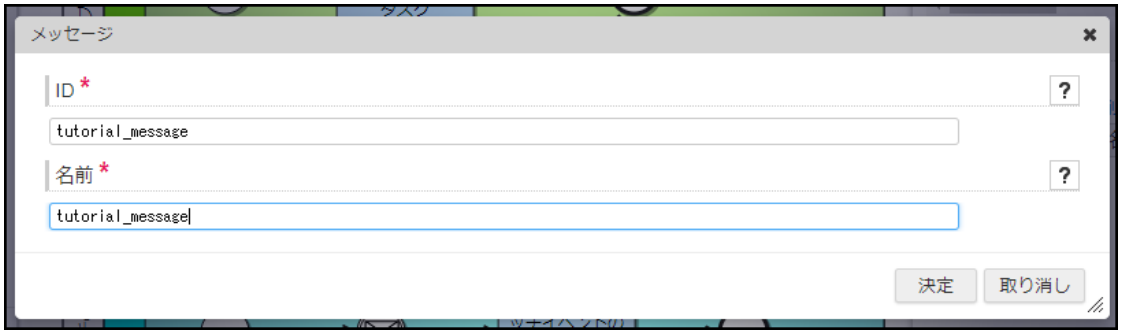
図：完成イメージ

1. 「メッセージ」を登録します。

1. 「プロセス」 - 「プロパティ」 - 「メッセージ＆シグナル」 - 「メッセージ」の「追加」リンクをクリックします。

図：「プロセス」 - 「プロパティ」 - 「メッセージ＆シグナル」 - 「メッセージ」

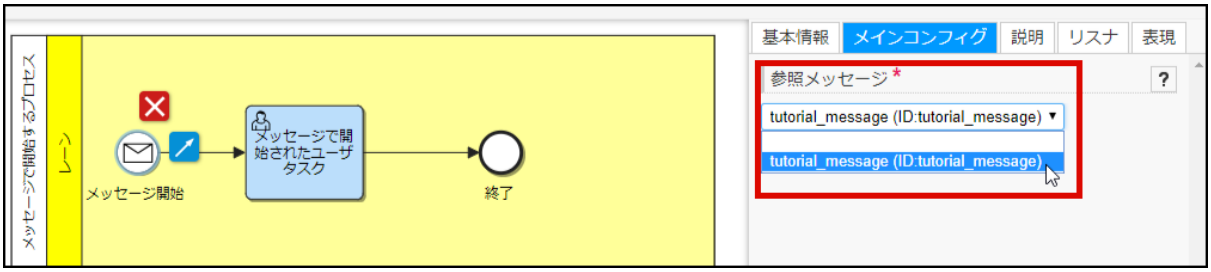
2. 「メッセージ」の「ID」と「名前」を設定します。
   - ID : tutorial_message
   - 名前 : tutorial_message



図：「メッセージ」

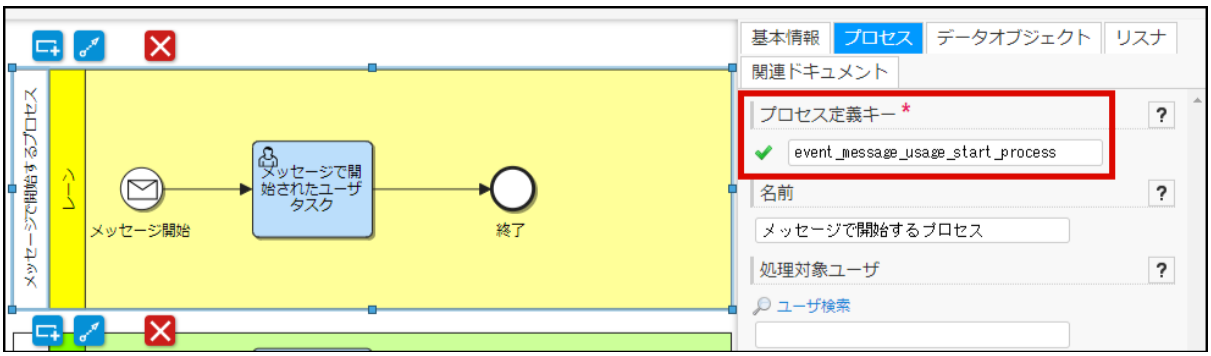
2. 「メッセージ開始イベント」を配置します。

3. 「メッセージ開始イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照メッセージ」のプルダウンリストのtutorial_message を選択します。



図：「メッセージ開始イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照メッセージ」

4. 「プロセス」の「プロセス定義キー」を設定します。
   - プロセス定義キー : event_message_usage_start_process



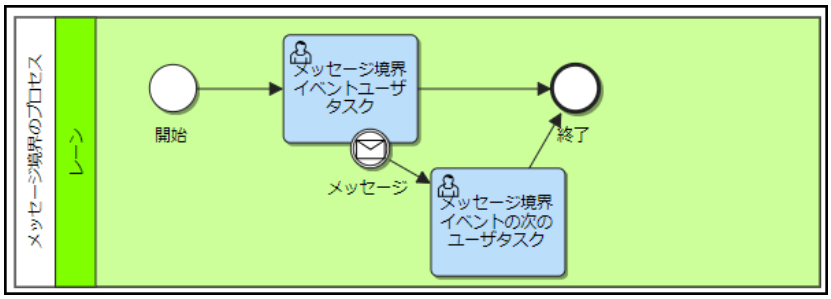
図：「プロパティ」 - 「プロセス」

5. 「メッセージ開始イベント」がメッセージを受信して「プロセス」が開始したことを確認するため、「ユーザタスク」を配置します。
   - 担当者 : aoyagi

図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

メッセージを受信することによりユーザタスクを中断するプロセス定義を作成する

メッセージを受信することにより、ユーザタスクを中断するプロセス定義を作成します。



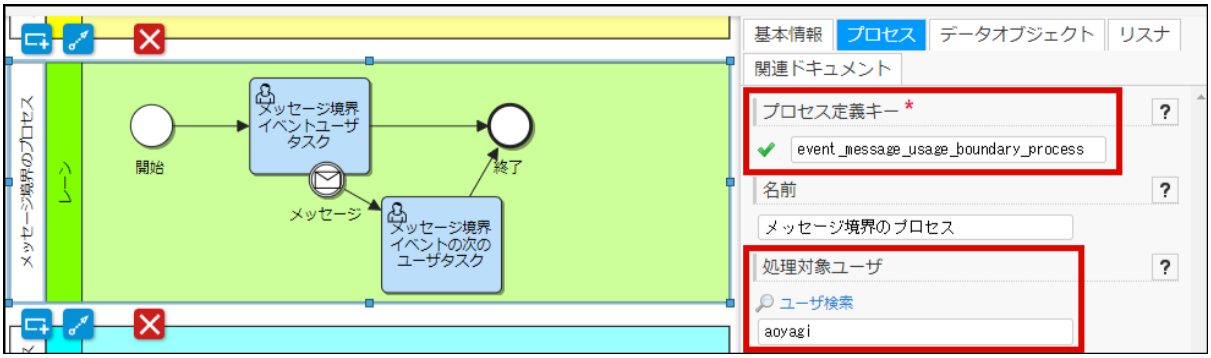
図：完成イメージ

1. 「メッセージ」を登録します。前項と同一ファイル内でプロセス定義を作成している場合は行う必要はありません。

   登録方法は「*メッセージを受信することにより開始するプロセス定義を作成する*」を参照してください。

2. 「開始イベント」を配置します。

3. 「プロセス」の「プロセス定義キー」と「処理対象ユーザ」を設定します。
   - プロセス定義キー：event_message_usage_boundary_process
   - 処理対象ユーザ：aoyagi

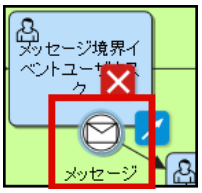


図：「プロパティ」 - 「プロセス」

4. 「ユーザタスク」を配置します。
   - 担当者：aoyagi

図：「ユーザタスク」

5. 「メッセージ境界イベント」を「ユーザタスク」に配置します。



図：「メッセージ境界イベント」

6. 「メッセージ境界イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照メッセージ」のプルダウンリストのtutorial_message を選択します。

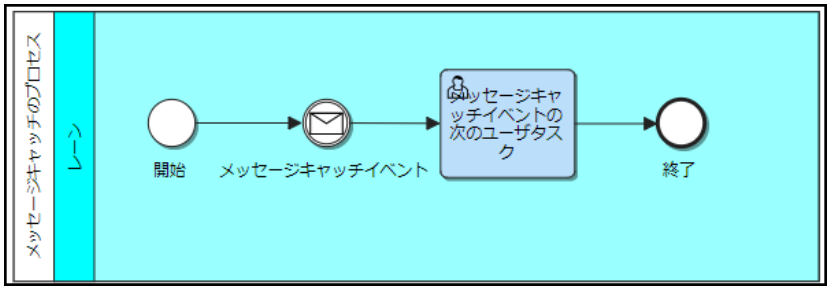
図：「メッセージ境界イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照メッセージ」

7. 「メッセージ境界イベント」で「ユーザタスク」が中断されたことを確認するために、別の「ユーザタスク」を配置します。
   - 担当者：aoyagi

図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

メッセージを受信することにより先に進むプロセス定義を作成する

メッセージを受信することにより先に進むプロセス定義を作成します。

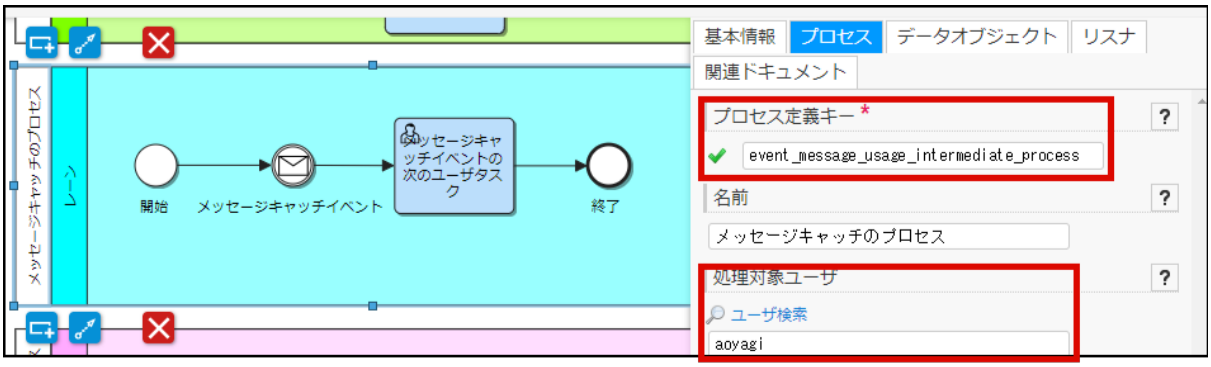


図：完成イメージ

1. 「メッセージ」を登録します。前項と同一ファイル内でプロセス定義を作成している場合は、行う必要はありません。

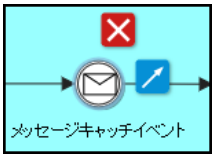
   登録方法は「*メッセージを受信することにより開始するプロセス定義を作成する*」を参照してください。

2. 「開始イベント」を配置します。
3. 「プロセス」の「プロセス定義キー」と「処理対象ユーザ」を設定します。
   - プロセス定義キー：event_message_usage_intermediate_process
   - 処理対象ユーザ：aoyagi



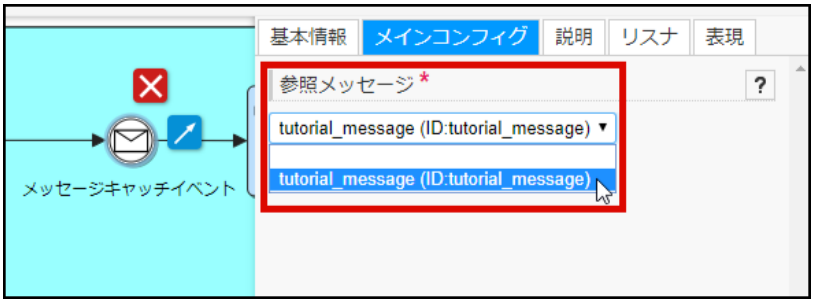
図：「プロパティ」 - 「プロセス」

4. 「メッセージキャッチイベント」を配置します。



図：「メッセージキャッチイベント」

5. 「メッセージキャッチイベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照メッセージ」のプルダウンリストのtutorial_message を選択します。



図：「メッセージキャッチイベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「参照メッセージ」

6. 「メッセージキャッチイベント」から次に進んだことを確認するために「ユーザタスク」を配置します。
   - 担当者：aoyagi

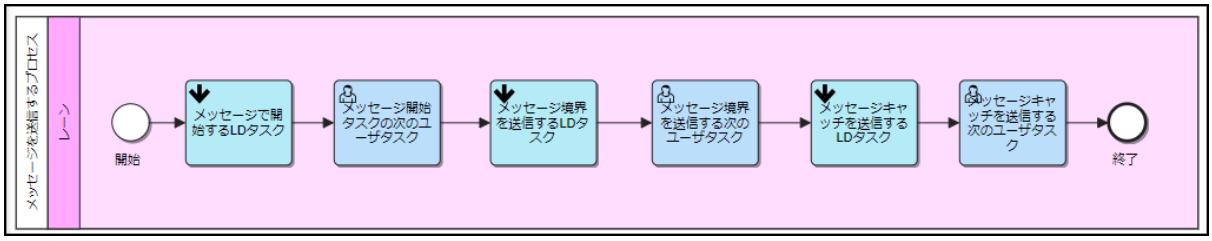
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

メッセージを送信するプロセス定義を作成する

メッセージを送信するプロセス定義を作成します。

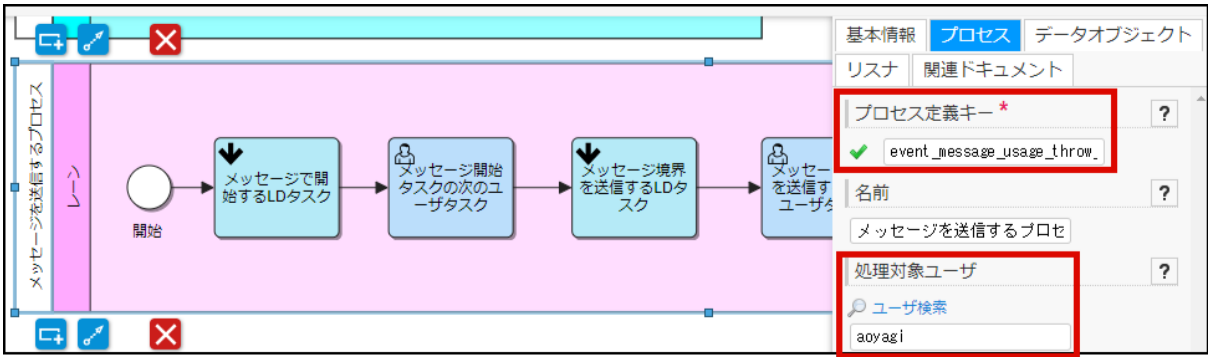
「IM-LogicDesignerタスク」を使用し、メッセージを送信するプロセスです。
「入力データ」として内容を与えたメッセージが、「IM-LogicDesignerタスク」に設定したロジックフローによって送信されます。
送信されたメッセージは、「メッセージで開始するプロセス」「メッセージ境界のプロセス」「メッセージキャッチのプロセス」がそれぞれ受信し、結果としてプロセスが進行します。

また、プロセスの進行状況を確認するため、「IM-LogicDesignerタスク」の他に「ユーザタスク」を配置しています。



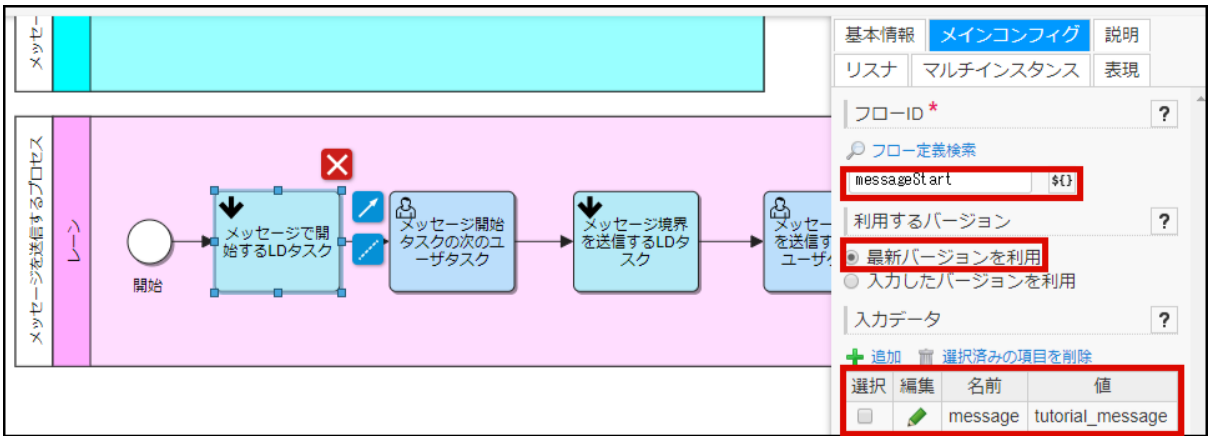
図：完成イメージ

1. 「開始イベント」を配置します。
2. 「プロセス」の「プロセス定義キー」と「処理対象ユーザ」を設定します。
   - プロセス定義キー：event_message_usage_throw_process
   - 処理対象ユーザ：aoyagi



図：「プロパティ」 - 「プロセス」

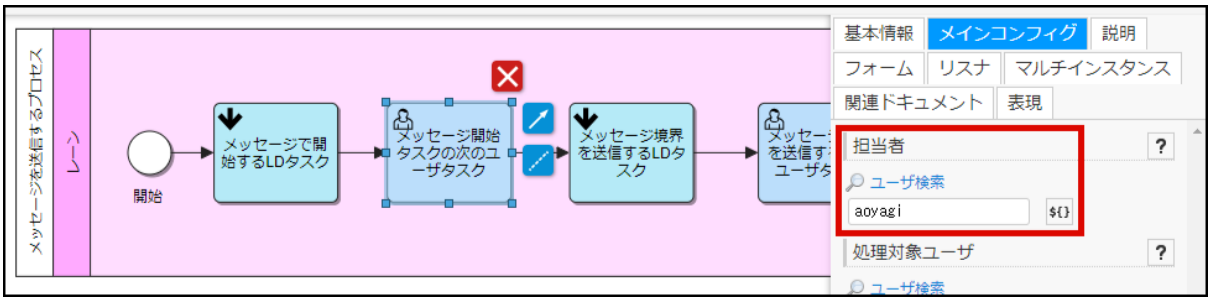
3. 「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」 で以下の項目を設定してください。
   - フローID：messageStart
   - 利用するバージョン：最新バージョンを利用
   - 入力データ：
     - 名前 message
     - 値 tutorial_message



図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

> **コラム**
>
> 「フローIDの設定」、「入力データ」の設定方法は、「*IM-LogicDesignerのリスナを利用する*」を参照してください。

4. 「IM-LogicDesignerタスク」から次に進んだことを確認するために「ユーザタスク」を配置します。
   - 担当者：aoyagi

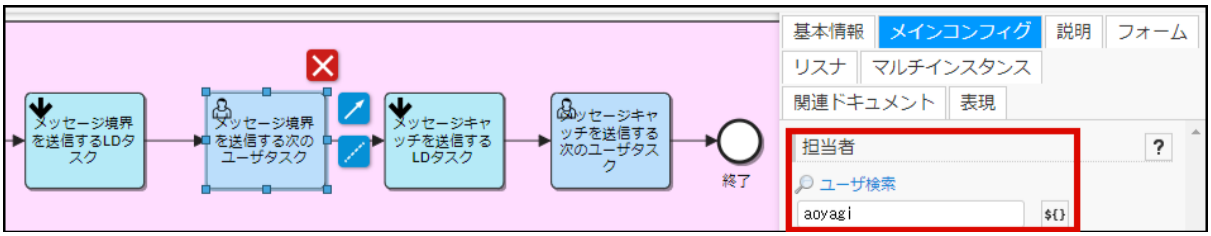


図：「ユーザタスク」

5. 「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」 で以下の項目を設定してください。
   - フローID : messageSend
   - 利用するバージョン : 最新バージョンを利用
   - 入力データ1:
     - 名前 message
     - 値 tutorial_message
   - 入力データ2:
     - 名前 processDefinitionKey
     - 値 event_message_usage_boundary_process

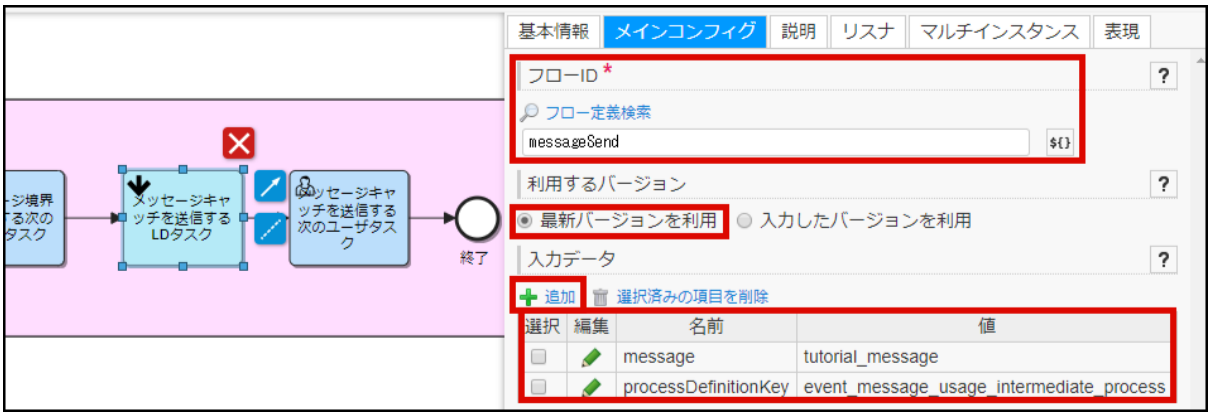
図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

6. 「IM-LogicDesignerタスク」から次に進んだことを確認するために「ユーザタスク」を配置します。
   - 担当者 : aoyagi



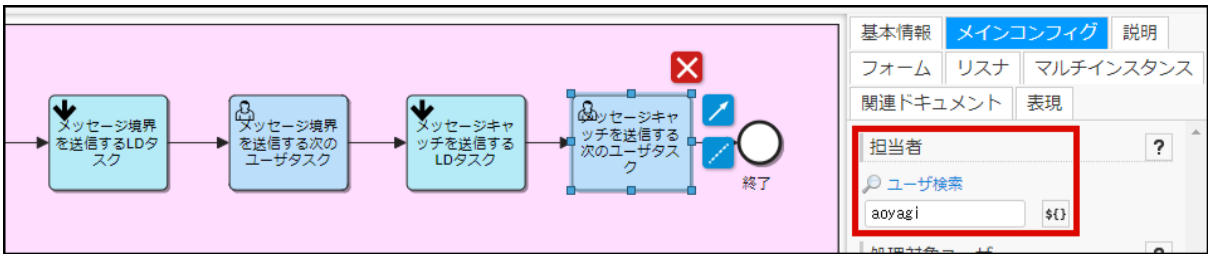
図：「ユーザタスク」

7. 「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」 で以下の項目を設定してください。
   - フローID : messageSend
   - 利用するバージョン : 最新バージョンを利用
   - 入力データ1:
     - 名前 message
     - 値 tutorial_message
   - 入力データ2:
     - 名前 processDefinitionKey
     - 値 event_message_usage_intermediate_process



図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

8. 「IM-LogicDesignerタスク」から次に進んだことを確認するために「ユーザタスク」を配置します。
   - 担当者 : aoyagi



図：「ユーザタスク」

実行結果を確認する

このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行した結果の確認を行います。

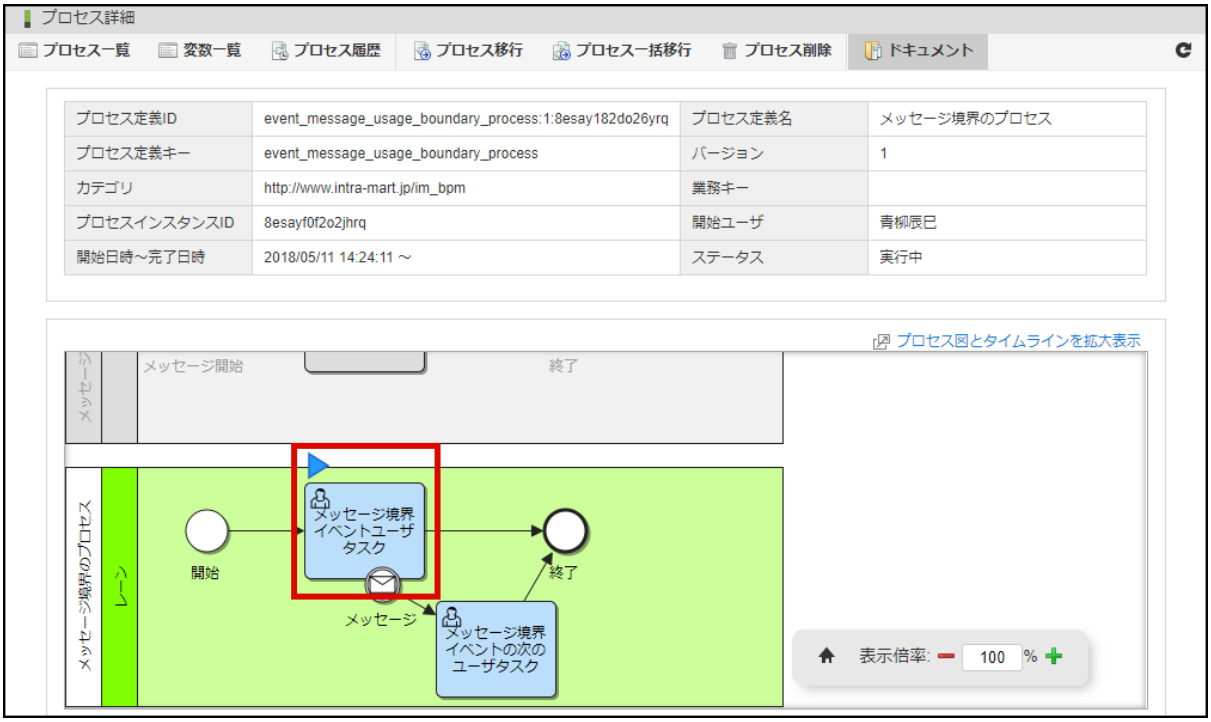
1. 「プロセス定義キー」 event_message_usage_start_process がプロセス定義として、デプロイされていることを確認します。

図：プロセス定義一覧 event_message_usage_start_process

2. 「プロセス定義キー」 event_message_usage_boundary_process と event_message_usage_intermediate_process のプロセスを開始します。

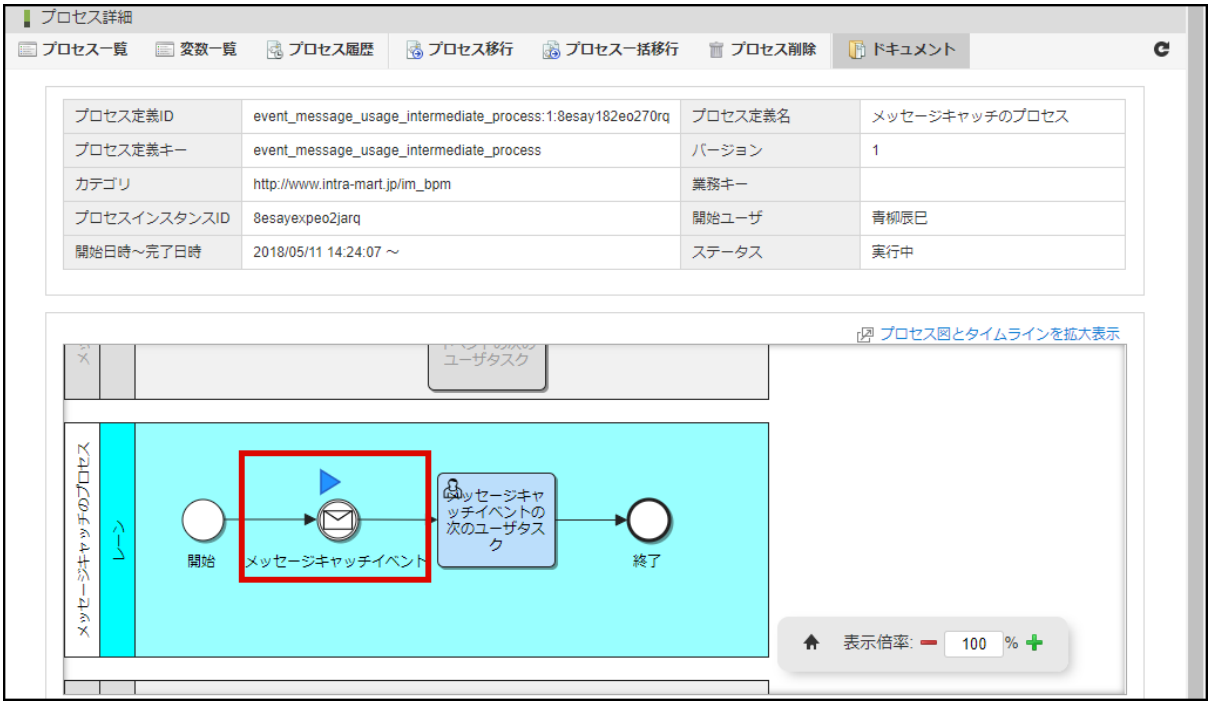
図：プロセス開始

3. 「プロセス定義キー」 event_message_usage_boundary_process が開始され、「メッセージ境界イベント」がついた「ユーザタスク」が開始されていることを確認します。



図：プロセス詳細 event_message_usage_boundary_process

4. 「プロセス定義キー」 event_message_usage_intermediate_process が開始され、「メッセージキャッチイベント」に滞在していることを確認します。

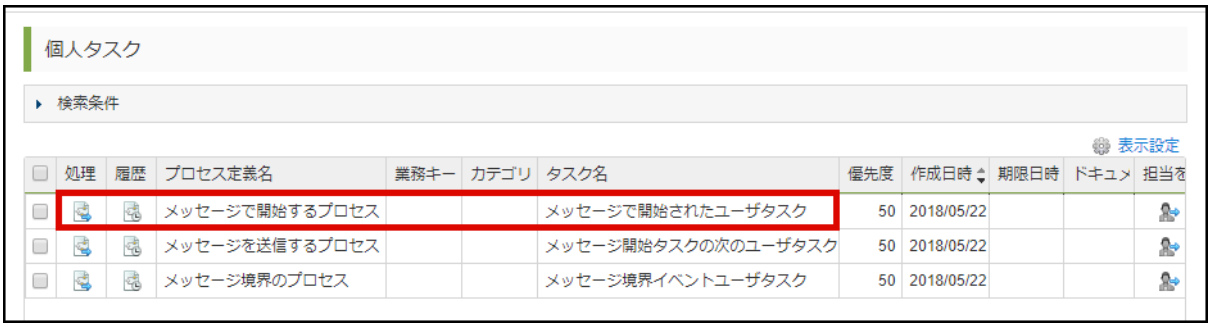


図：プロセス詳細 event_message_usage_intermediate_process

5. 「プロセス定義キー」 event_message_usage_throw_process のプロセスを開始します。

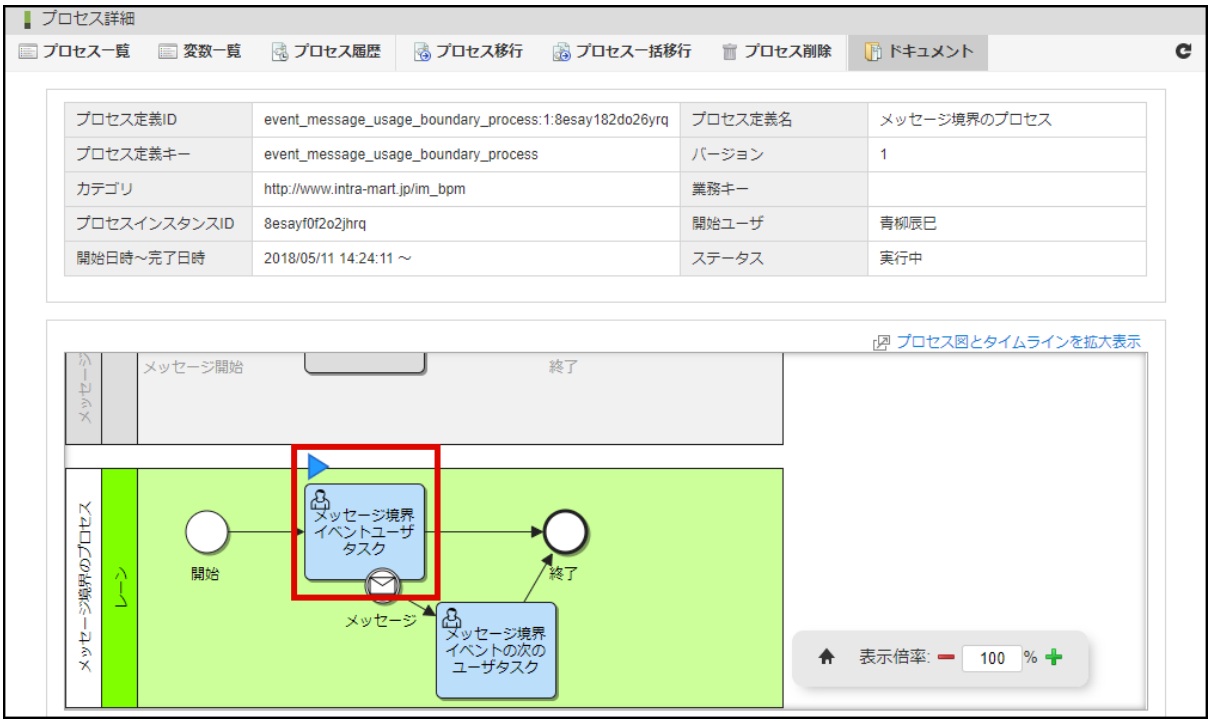
図：プロセス開始 event_message_usage_throw_process

6. 「プロセス定義キー」 event_message_usage_start_process が開始されていることを確認します。

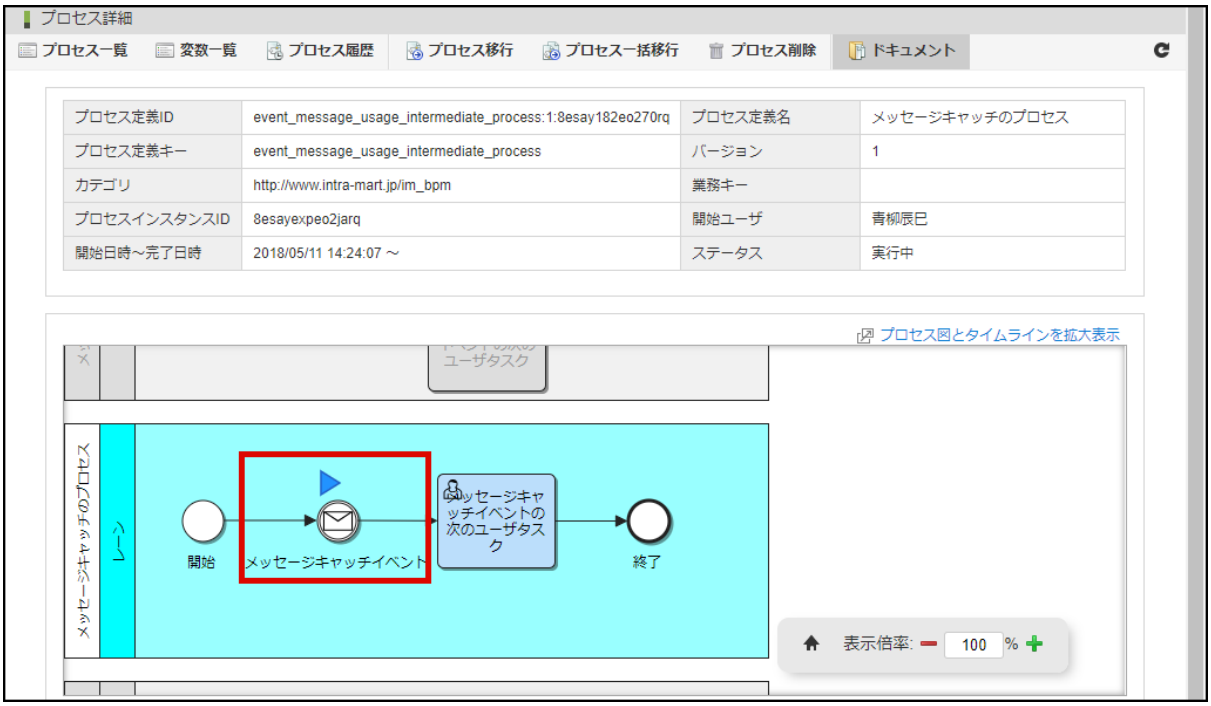


図：タスク一覧

7. 「メッセージ境界イベント」と「メッセージキャッチイベント」がメッセージを受信していないことを確認します。



図：プロセス詳細 event_message_usage_boundary_process



図：プロセス詳細 event_message_usage_intermediate_process

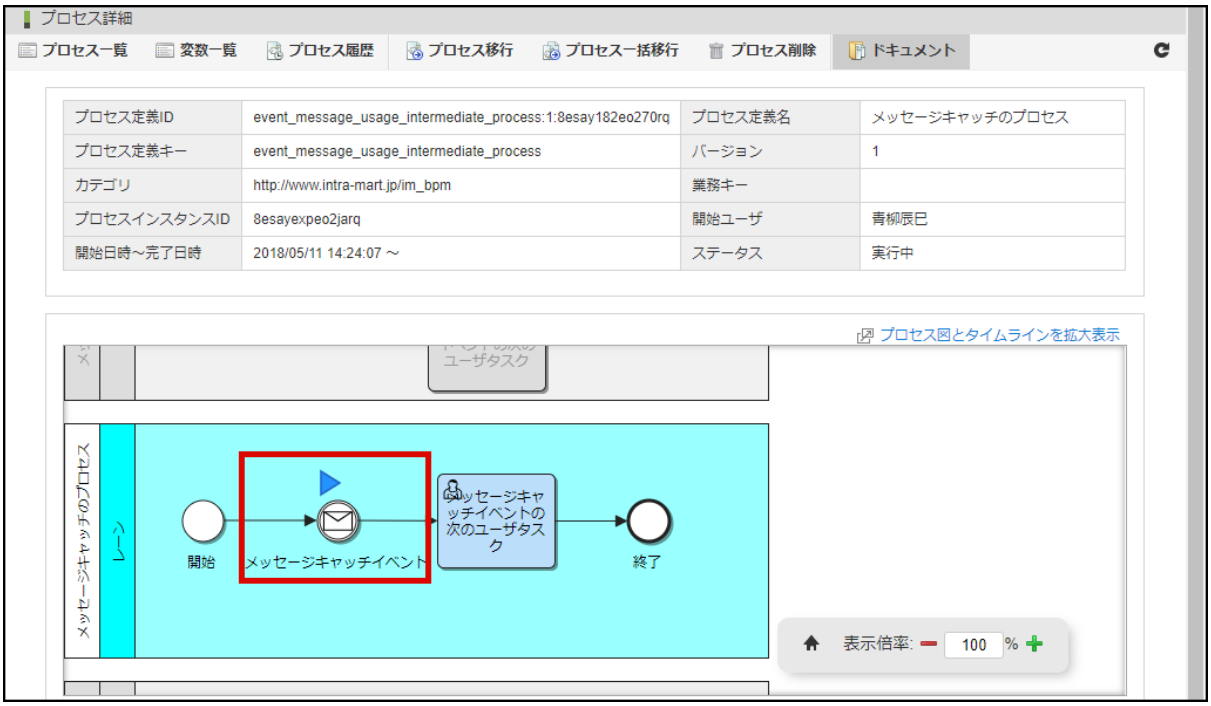
8. 「メッセージを送信するプロセス」の「ユーザタスク」を処理します。

図：タスク一覧

9. 「メッセージ境界イベント」に対してメッセージが送信されていることを確認します。

図：タスク一覧

10. 「メッセージキャッチイベント」に対してメッセージが送信されていないことを確認します。

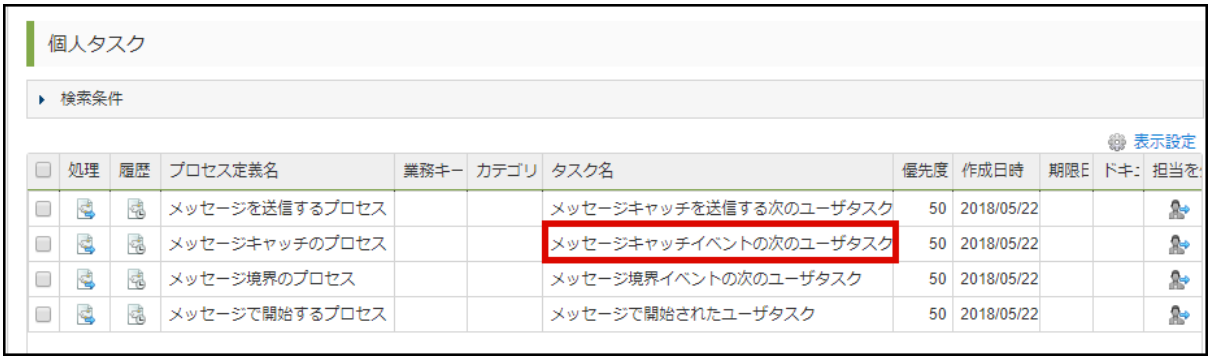


図：プロセス詳細 event_message_usage_intermediate_process

11. 「メッセージを送信するプロセス」の「ユーザタスク」を処理します。

図：タスク一覧

12. 「メッセージキャッチイベント」に対してメッセージが送信されていることを確認します。



図：タスク一覧

タイマイベントを使用する

このチュートリアルでは、タイマイベントを使用して、時間をトリガにし、プロセスの開始やプロセスを進める方法を解説します。
タイマをトリガとするイベントは、以下のとおりです。

- タイマ開始イベント
- タイマ境界イベント
- タイマキャッチイベント

コラム

各イベントの詳細については、以下を参照してください。

タイマ開始イベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「タイマ開始イベント」
タイマ境界イベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「タイマ境界イベント」
タイマキャッチイベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「タイマキャッチイベント」

コラム

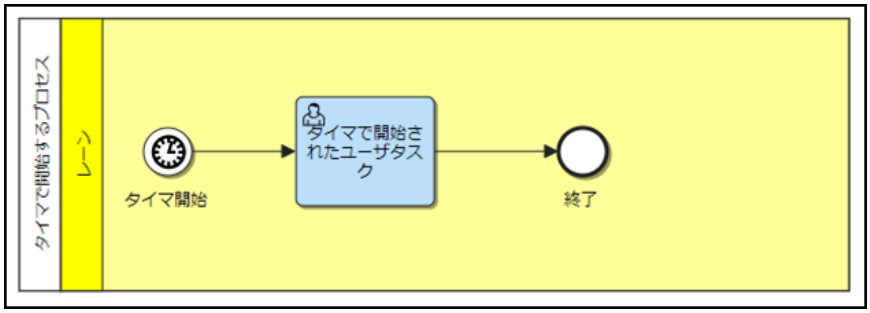
このチュートリアルで作成する「プロセス定義」のサンプルを、以下のリンクからダウンロードできます。
event_timer_usage.bpmn

このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- 指定した時間間隔で複数回プロセスを開始するプロセス定義を作成する
- 指定した時間経過によりユーザタスクを中断するプロセス定義を作成する
- 指定した時間にユーザタスクを開始するプロセス定義を作成する
- 実行結果を確認する

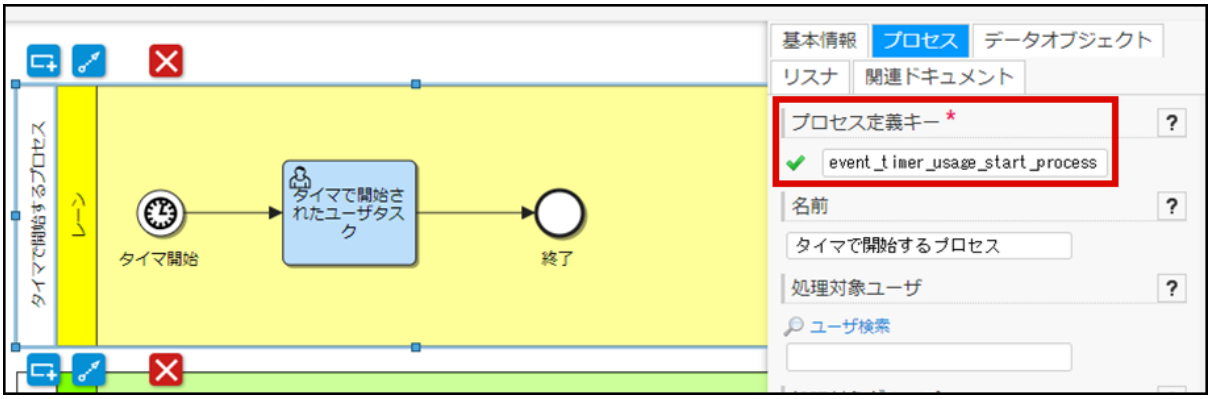
指定した時間間隔で複数回プロセスを開始するプロセス定義を作成する

指定した時間間隔で複数回プロセスを開始するプロセス定義を作成します。



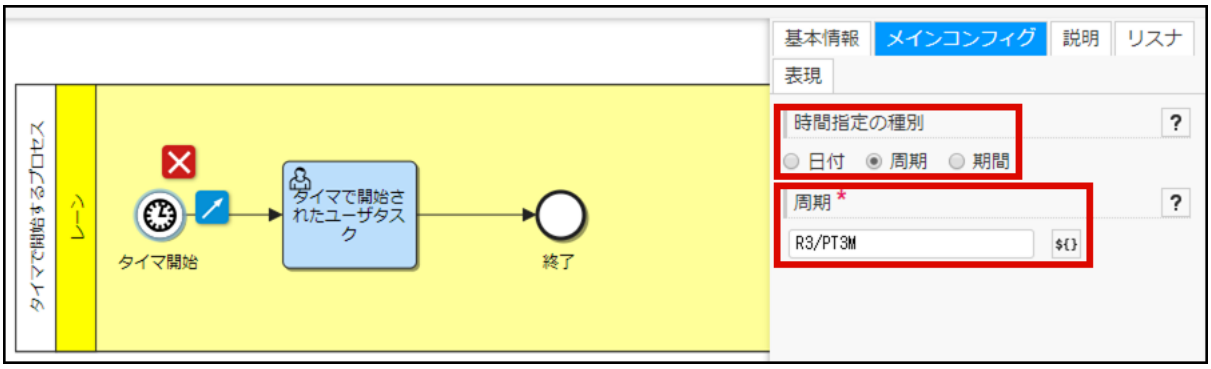
図：完成イメージ

1. 「プロセス」の「プロセス定義キー」を設定します。
   プロセス定義キー：event_timer_usage_start_process



図：「タイマで開始するプロセス」 - 「プロパティ」 - 「プロセス」

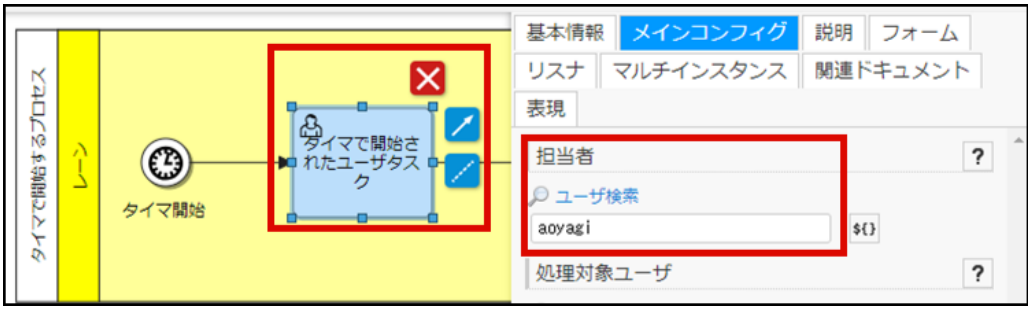
2. 「タイマ開始イベント」 - 「メインコンフィグ」 の 「周期」を設定します。
   - 時間指定の種別：周期
   - 周期：R3/PT3M



図：「タイマ開始イベント 」 - 「プロパティ」 - 「メインコンフィグ」

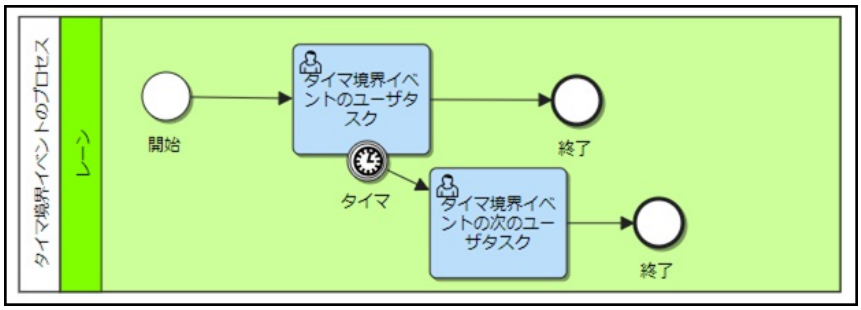
> **コラム**
>
> R3/PT3Mは「3分間隔の繰り返しを3回行う」ことを、ISO 8601の形式に則って記述したものです。
>
> 例:10時間毎に5回開始→R5/PT10H
> 1日毎に10回開始→R10/P1D

3. 「タイマ開始イベント」が指定した時間の後「プロセス」を開始したことを確認するため、「ユーザタスク」を配置します。



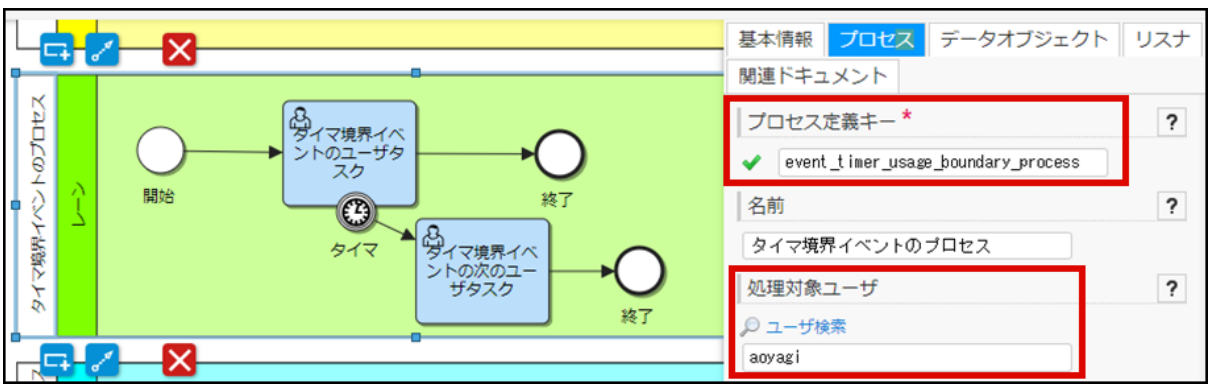
指定した時間経過によりユーザタスクを中断するプロセス定義を作成する

指定した時間経過により、ユーザタスクを中断するプロセス定義を作成します。



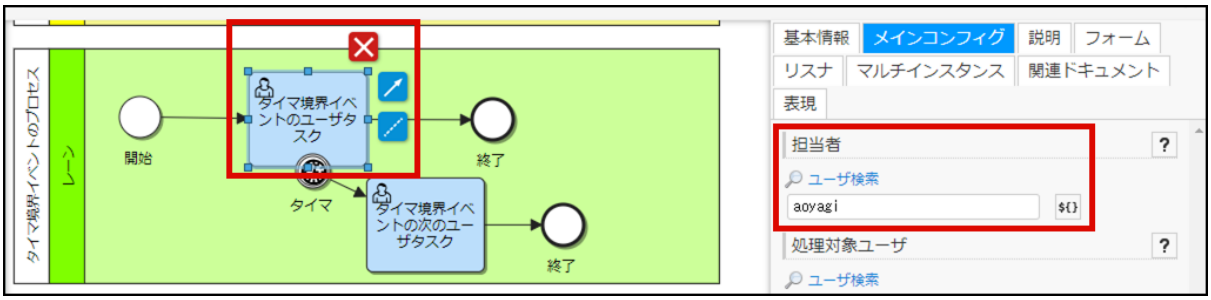
図：完成イメージ

1. 「開始イベント」を配置します。
2. 「プロセス」の「プロセス定義キー」と「処理対象ユーザ」を設定します。
   - プロセス定義キー：event_timer_usage_boundary_process
   - 処理対象ユーザ：aoyagi



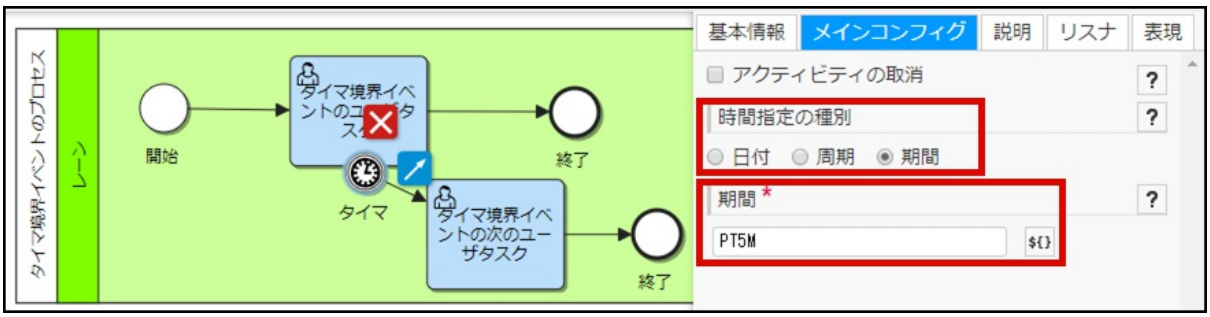
図：「タイマ境界イベントのプロセス」 - 「プロパティ」 - 「プロセス」

3. 指定した時間経過により、中断される「ユーザタスク」を配置します。
   - 処理対象ユーザ：aoyagi



図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

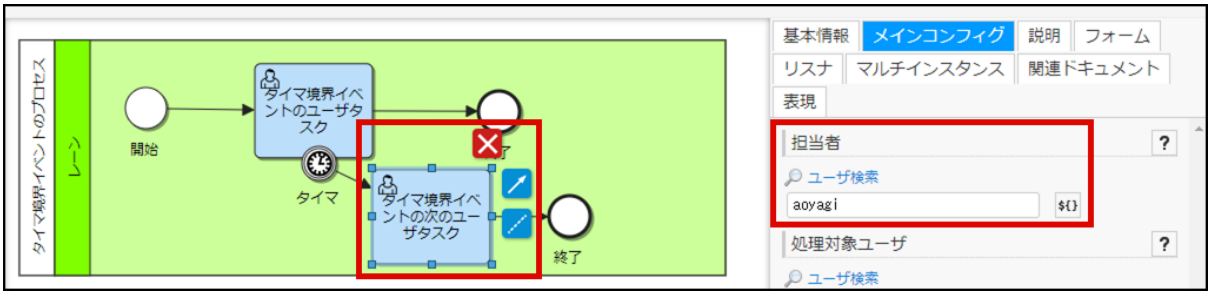
4. 5分経過することで、ユーザタスクを中断する「タイマ境界イベント」を配置します。
   タイマ境界イベントの「プロパティ」で以下の項目を設定します。
   - 時間指定の種別：期間
   - 期間：PT5M



図：「メッセージ境界イベント」

> **コラム**
>
> PT5Mは「5分」を、ISO 8601の形式に則って記述したものです。
>
> 例:10分→PT10M
> 2日→P2D

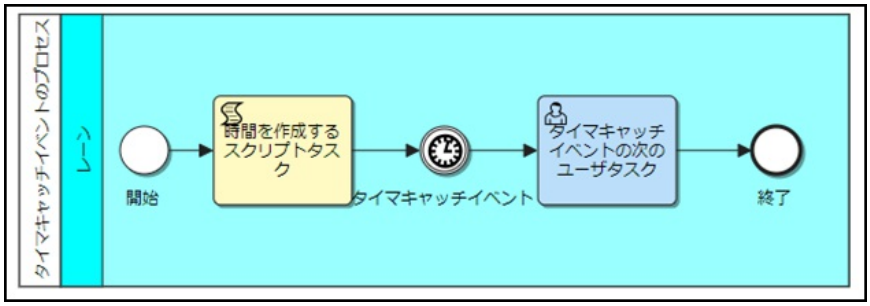
5. 指定した時間経過により中断されたことを確認するため、別のユーザタスクを配置します。
   - 処理対象ユーザ：aoyagi



図：「ユーザタスク」

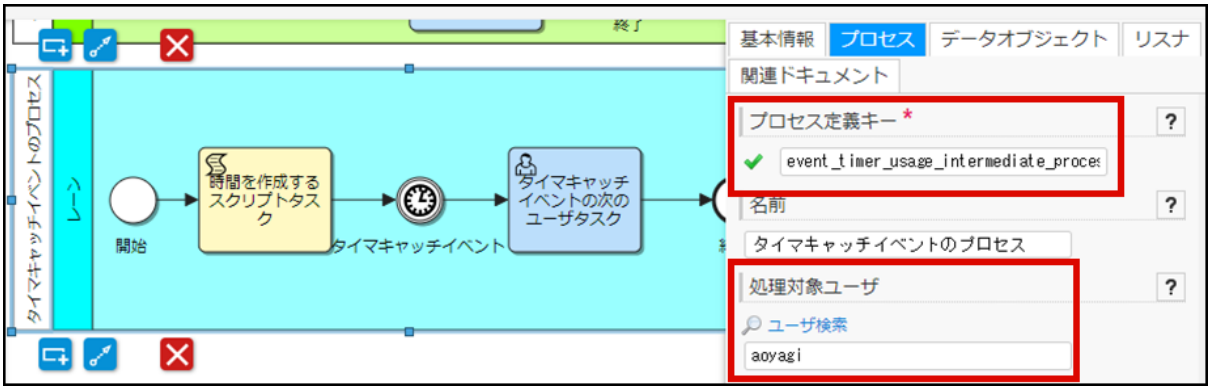
指定した時間にユーザタスクを開始するプロセス定義を作成する

指定した時間にユーザタスクを開始するプロセス定義を作成します。



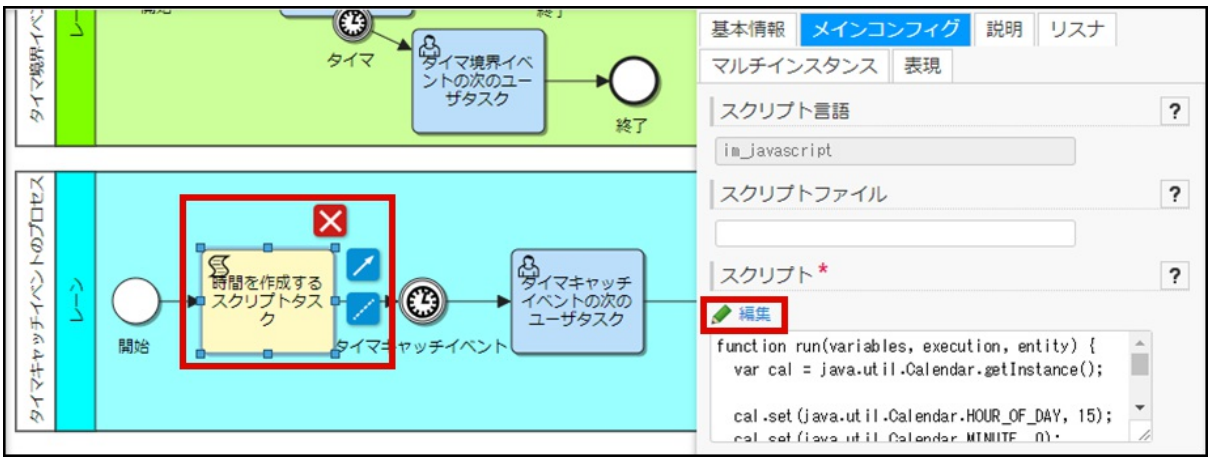
図：完成イメージ

1. 「開始イベント」を設置します。
2. 「プロセス」の「プロセス定義キー」と「処理対象ユーザ」を設定します。
   - プロセス定義キー：event_timer_usage_intermediate_process
   - 処理対象ユーザ：aoyagi



図：「タイマキャッチイベントのプロセス」 - 「プロパティ」 - 「プロセス」

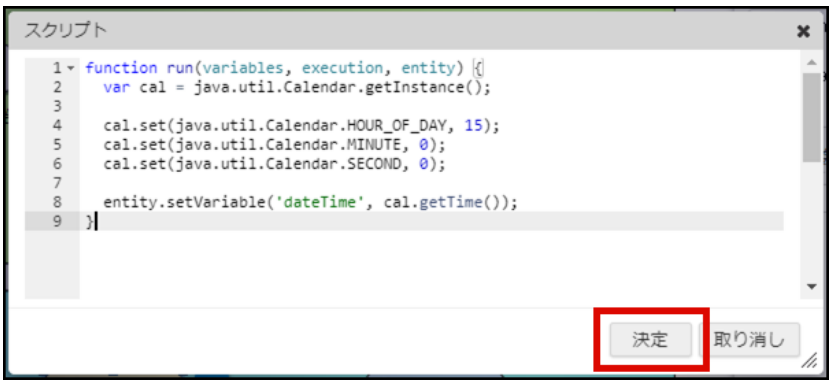
3. 「スクリプトタスク」を作成します。
   スクリプトタスクの「メインコンフィグ」の、スクリプトの編集リンクをクリックします。



図：「スクリプトタスク」 - 「プロパティ」 - 「メインコンフィグ」

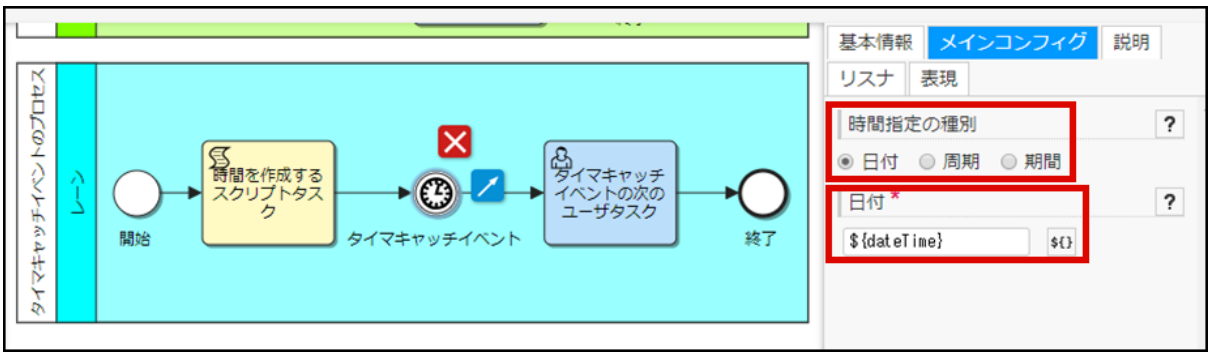
4. 当日の15時という情報を変数 dateTime に設定するため、以下のスクリプトを設定します。

```
function run(variables, execution, entity) {
  var cal = java.util.Calendar.getInstance();

  cal.set(java.util.Calendar.HOUR_OF_DAY, 15);
  cal.set(java.util.Calendar.MINUTE, 0);
  cal.set(java.util.Calendar.SECOND, 0);

 entity.setVariable('dateTime', cal.getTime());
}
```



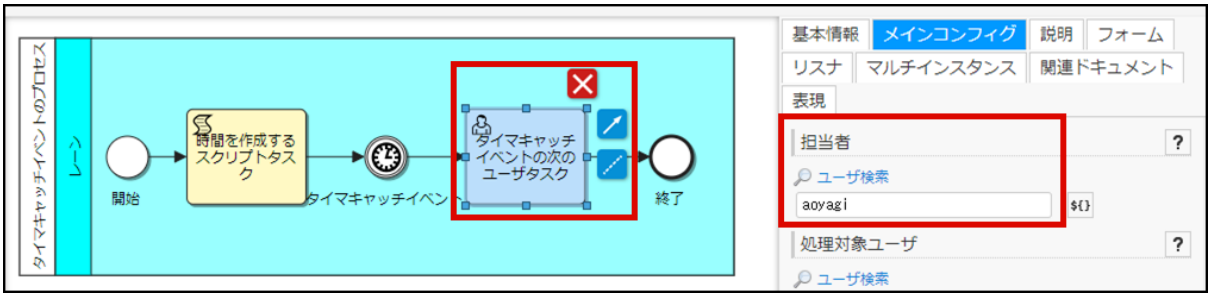
図：スクリプト編集のポップアップ

5. タイマキャッチイベントの「メインコンフィグ」で、「時間指定の種別」と「日付」を設定してください。
   - 時間指定の種別：日付
   - 日付：${dateTime}



図：「タイマキャッチイベント」 - 「プロパティ」 - 「メインコンフィグ」

6. タイマキャッチイベントから次に進んだことを確認するために「ユーザタスク」を配置します
   - 担当者：aoyagi



図：「ユーザタスク」

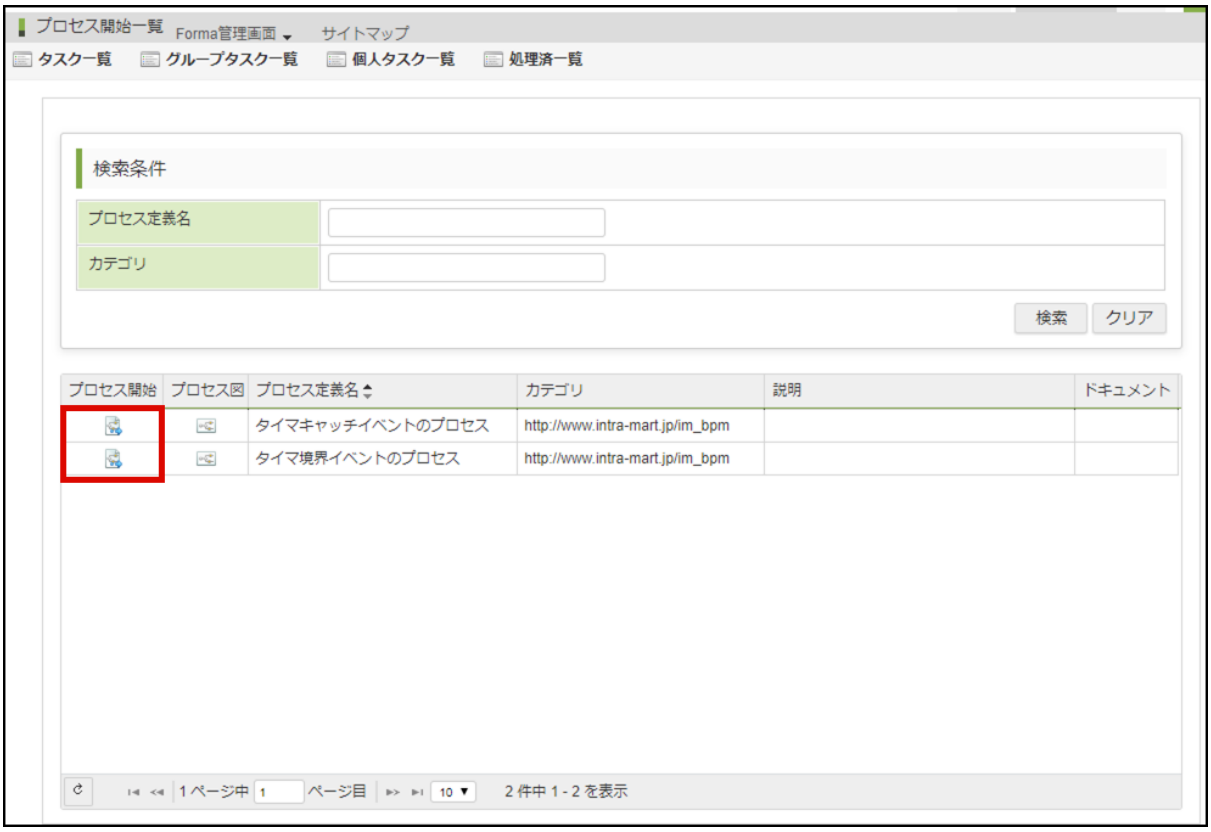
実行結果を確認する

このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行した結果の確認を行います。

1. 「タイマで開始するプロセス」はデプロイ完了後にタイマ開始イベントが始動します。
   設定した周期 R3/PT3M どおり、3分おきにタスクが起動し、合計３つ揃うのを確認します。

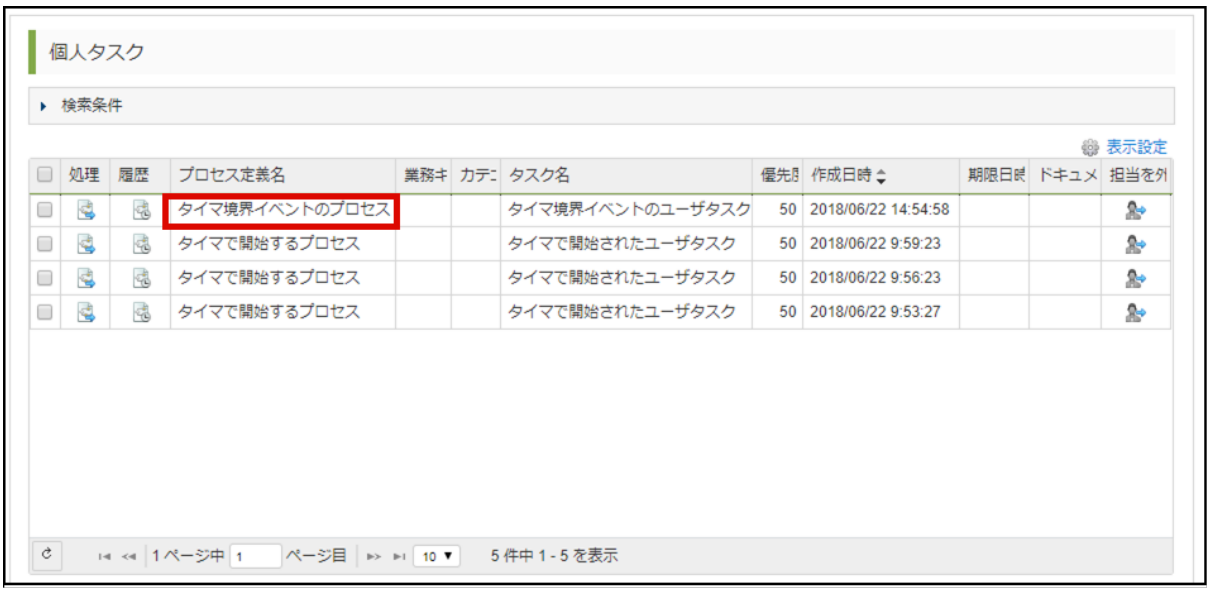
図：「タスク一覧」

2. 「プロセス開始一覧」から、「プロセス定義名」 タイマキャッチイベントのプロセス と タイマ境界イベントのプロセス を実行します。



図：「プロセス開始一覧」

3. 「タスク一覧」の「個人タスク」で、 タイマ境界イベントのプロセス のタスクが開始されていることを確認します。

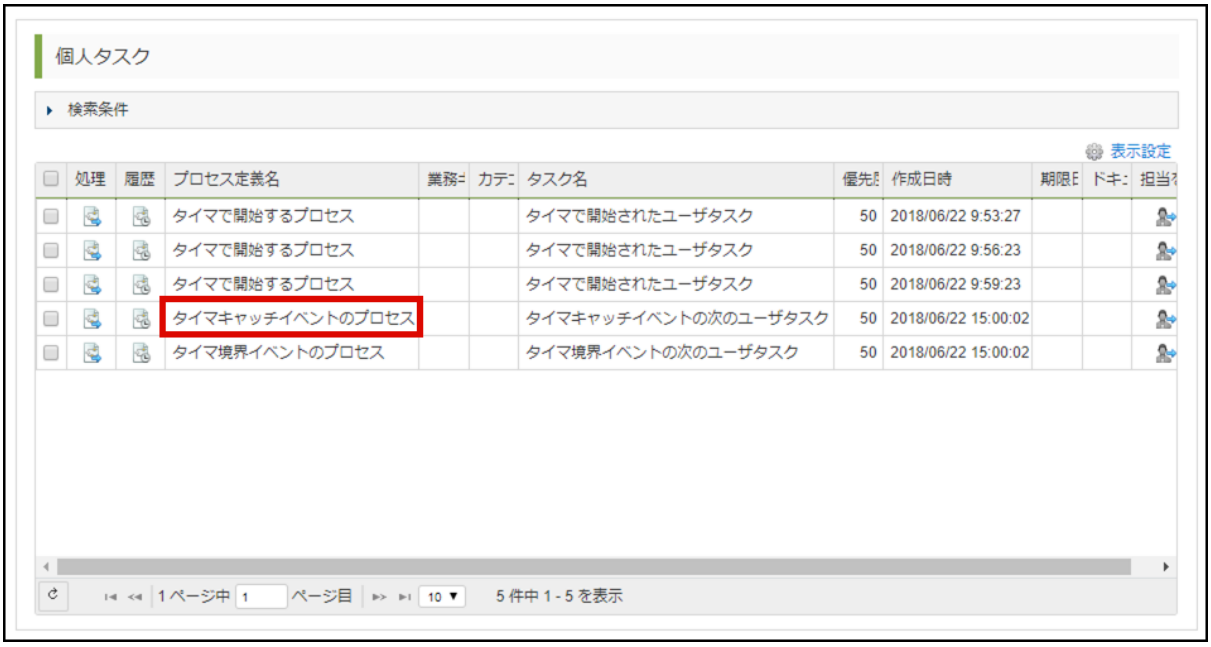


図：「タスク一覧」

4. PT5M の設定どおり5分後にタスクが中止され、「タイマ境界イベントの次のユーザタスク」に移っていることを確認します。

図：「プロセス一覧」 - 「プロセス詳細」

5. 「プロセス一覧」 タイマキャッチイベントのプロセス が「15時」に起動することを確認します。



図：「タスク一覧」

> **コラム**
>
> タスクを開始した時間が15時以降の場合、即時実行されます。

IM-LogicDesignerタスクを利用してユーザ情報を取得する

このチュートリアルでは、「IM-LogicDesignerタスク」を利用してユーザ情報取得のロジックフローを実行する方法を解説します。

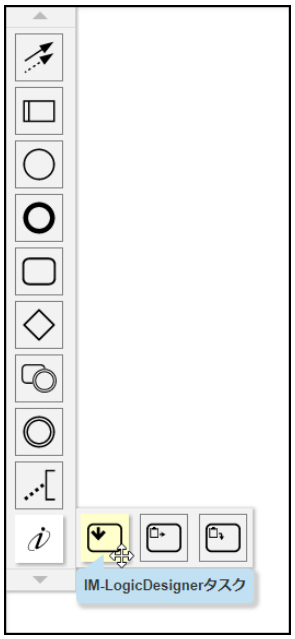
- 「IM-LogicDesigner」とは、intra-mart Accel Platform上でビジネスロジックを簡単に作成できるアプリケーションです。
- 「IM-LogicDesignerタスク」は、タスク中に設定された情報からIM-LogicDesignerで定義されたロジックフローの処理を行います。

「IM-LogicDesigner」の詳細については、「IM-LogicDesigner仕様書」を参照してください。
「IM-LogicDesignerタスク 」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「IM-LogicDesignerタスク」もあわせて参照してください。

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **logicdesigner_task_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- IM-LogicDesignerタスクを配置する
- プロセスグローバルのデータオブジェクトを定義する
- 実行するロジックフローを設定する
- 入力データを設定する
- 実行結果の受け取りについて設定する
- 実行結果を確認する

IM-LogicDesignerタスクを配置する



図：パレット - intra-mart - IM-LogicDesignerタスク

1. パレットの  にカーソルを合わせ、パレット右側に現れるintra-martタスクの一覧から  をドラッグ＆ドロップして配置します。
2. 配置されたエレメントをクリックして選択することにより、プロパティエリアに選択したエレメントのプロパティが表示されます。

プロセスグローバルのデータオブジェクトを定義する

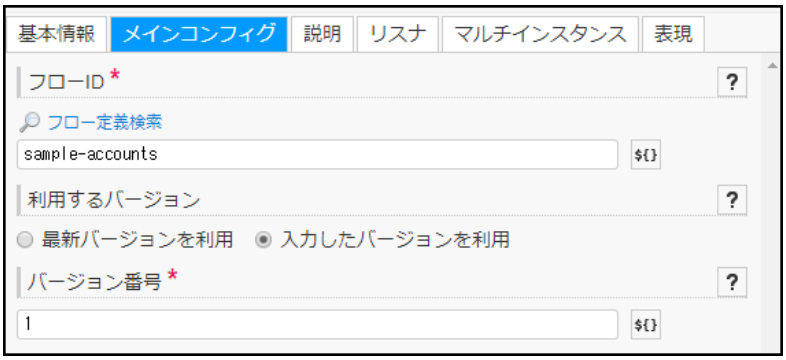
1. プロセスのプロパティ「データオブジェクト」に `user_cd = "aoyagi"` のデータプロパティを定義します。

図：プロセス：プロパティ - データオブジェクト

> **コラム**
>
> プロセスグローバルのデータオブジェクト設定方法については「*プロセスにデータプロパティを設定する*」を参照してください。

実行するロジックフローを設定する

1. プロパティエリアにて「メインコンフィグ」タブを開きます。



図：IM-LogicDesignerタスク：プロパティ - メインコンフィグ - 1

2. フローIDを設定します。

   フローIDを設定するには、以下3つの方法があります。

   - **「フロー定義検索」により、ロジックフロー定義を選択する**
     1. 「フロー定義検索」をクリックし、「ロジックフロー定義検索」ウィンドウを開きます。
     2. フロー定義ID `sample-accounts` のロジックフローを選択し、「決定」ボタンをクリックします。
   - **フローIDを文字列で入力する**
     1. フローIDの入力フォームに直接 `sample-accounts` を入力します。
   - **EL式で動的にフローIDを設定する**

     EL式による動的なフローIDの設定方法については、別途「*IM-LogicDesignerタスクで実行するロジックフローを動的に設定する*」で説明しています。

3. 「利用するバージョン」で、実行するロジックフローのバージョン指定方法を選択します。

   実行するロジックフローのバージョンを指定できます。
   以下2パターンの設定が可能です。
   このチュートリアルでは、どちらの設定でも構いません。

   - **最新バージョンを利用**

     指定されたロジックフローは、IM-LogicDesignerタスクに到達した時点の最新バージョンが実行されます。

   - **入力したバージョンを利用**

     「入力したバージョンを利用」の選択とあわせて「バージョン番号」を入力し、実行するロジックフローのバージョンを固定できます。
     このチュートリアルでは `1` を入力します。

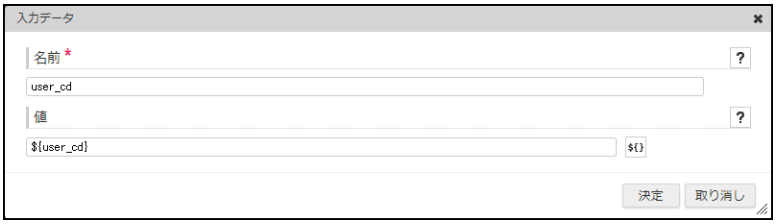
     > **コラム**
     >
     > バージョン番号は EL式 で入力することもできます。

入力データを設定する

実行するロジックフローに受け渡す入力データを設定します。
ここでは、ユーザコードを設定しています。

図：IM-LogicDesignerタスク：プロパティ - メインコンフィグ - 2

1. 「追加」をクリックし、ダイアログを開きます。
2. 「名前」に `user_cd` を入力し、「値」に `${user_cd}` を入力します。



図：入力データダイアログ

3. 「決定」ボタンをクリックして、入力データの表に反映します。

実行結果の受け取りについて設定する

実行したロジックフローから受け渡される実行結果データに関する設定を行います。

1. 「結果変数を格納する」チェックボックスをチェックONにします。
2. 「結果変数名」に `result_data` を入力します。
   ロジックフローの実行後、プロセスグローバルの変数 `result_data` が作成され、ロジックフローの出力値 `records` が格納されます。

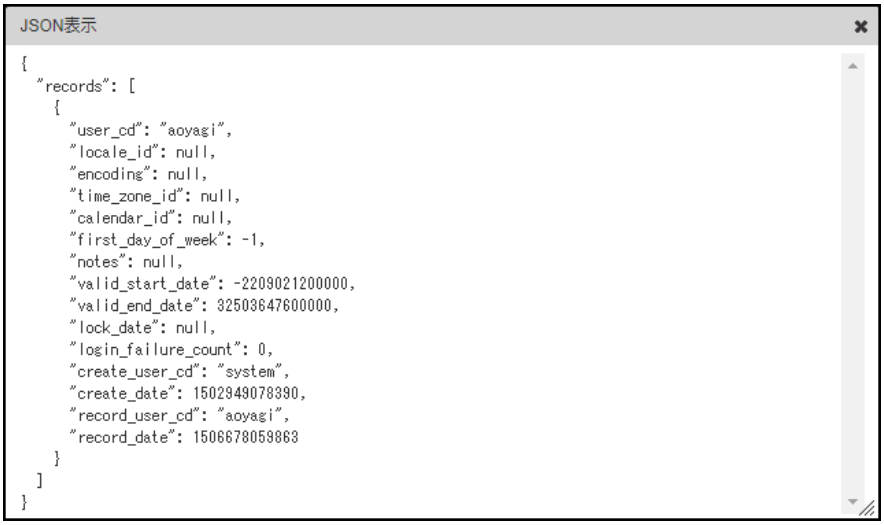
図：IM-LogicDesignerタスク：プロパティ - メインコンフィグ - 3

> **注意**
> 「結果変数を格納する」チェックボックスをオフにした場合、実行したロジックフローの出力値はすべて破棄されます。

実行結果を確認する

このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

1. 「プロセス一覧」画面にて完了したプロセスを検索し、実行したプロセスの「プロセス詳細」画面へ遷移します。
2. 「変数一覧」画面にて、変数 `result_data` の内容を確認します。



図：「変数一覧」画面 - JSON表示

コラム

- プロセスのデプロイ方法については「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のデプロイ」を参照してください。
- デプロイしたプロセスの実行方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスの開始」を参照してください。
- 「プロセス詳細」画面への遷移方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスを確認する」を参照してください。
- 「変数一覧」画面の操作方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスの変数を確認する」を参照してください。

IM-LogicDesignerタスクで実行するロジックフローを動的に設定する

このチュートリアルでは、「IM-LogicDesignerタスク」で実行するロジックフローを実行時に決定する方法を解説します。
「IM-LogicDesignerタスク」の基本的な使用方法については「*IM-LogicDesignerタスクを利用してユーザ情報を取得する*」を参照してください。

コラム

このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
logicdesigner_task_dynamic.bpmn

このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- 事前準備
- エレメントを配置する
- IM-LogicDesignerタスクのプロパティを設定する
- 開始イベントにFormaアプリケーションを設定する
- 実行結果を確認する

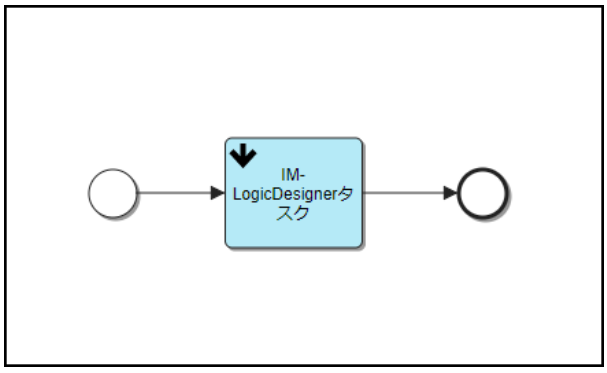
事前準備

1. Formaアプリケーション「**select_logic_flow**」をインポートします。

コラム

Formaアプリケーションのインポート手順は「IM-FormaDesigner 作成者操作ガイド」-「インポート・エクスポートを利用した IM-FormaDesigner のアプリケーションやデータソース定義の移行」を参照してください。

エレメントを配置する



図：配置イメージ

1. 開始イベント、IM-LogicDesignerタスク、終了イベントを配置して、順にシーケンスフローで接続します。

IM-LogicDesignerタスクのプロパティを設定する

図：プロパティ入力イメージ

1. IM-LogicDesignerタスクのプロパティで「メインコンフィグ」タブを開きます。
2. 「フローID」に ${dynamic_flow_id} を入力します。
3. 「利用するバージョン」で「入力したバージョンを利用」を選択します。
4. 「バージョン番号」に ${dynamic_flow_version} を入力します。
5. 「入力データ」には、ここでは何も設定しません。
6. 「結果変数を格納する」チェックボックスをオンにします。
7. 「結果変数名」に result_data を入力します。

開始イベントにFormaアプリケーションを設定する

1. 開始イベントのプロパティで「メインコンフィグ」タブを開きます。
2. 「フォームキー」に forma:select_logic_flow を入力します。

実行結果を確認する

このチュートリアルで作成したプロセスを開始すると、Formaアプリケーションが開きます。
開いたFormaアプリケーションで実行するフローを選択し、登録ボタンをクリックすることでタスクが進み、選択したロジックフローが実行されます。
バージョン番号は 1 が設定されており、変更不可の設定をしてあります。

1. 「プロセス一覧」画面にて完了したプロセスを検索し、実行したプロセスの「プロセス詳細」画面へ遷移します。
2. 「変数一覧」画面にて、以下の変数の値を確認します。

変数名	値
dynamic_flow_id	「List of Accounts」を実行した場合：sample-accounts 「Read intra-mart atom feed」を実行した場合：sample-im-topics-to-log
dynamic_flow_version	1
result_data	「List of Accounts」を実行した場合： ユーザ情報を持つ records 配列 「Read intra-mart atom feed」を実行した場合： ニューストピックス情報を持つ topics 配列

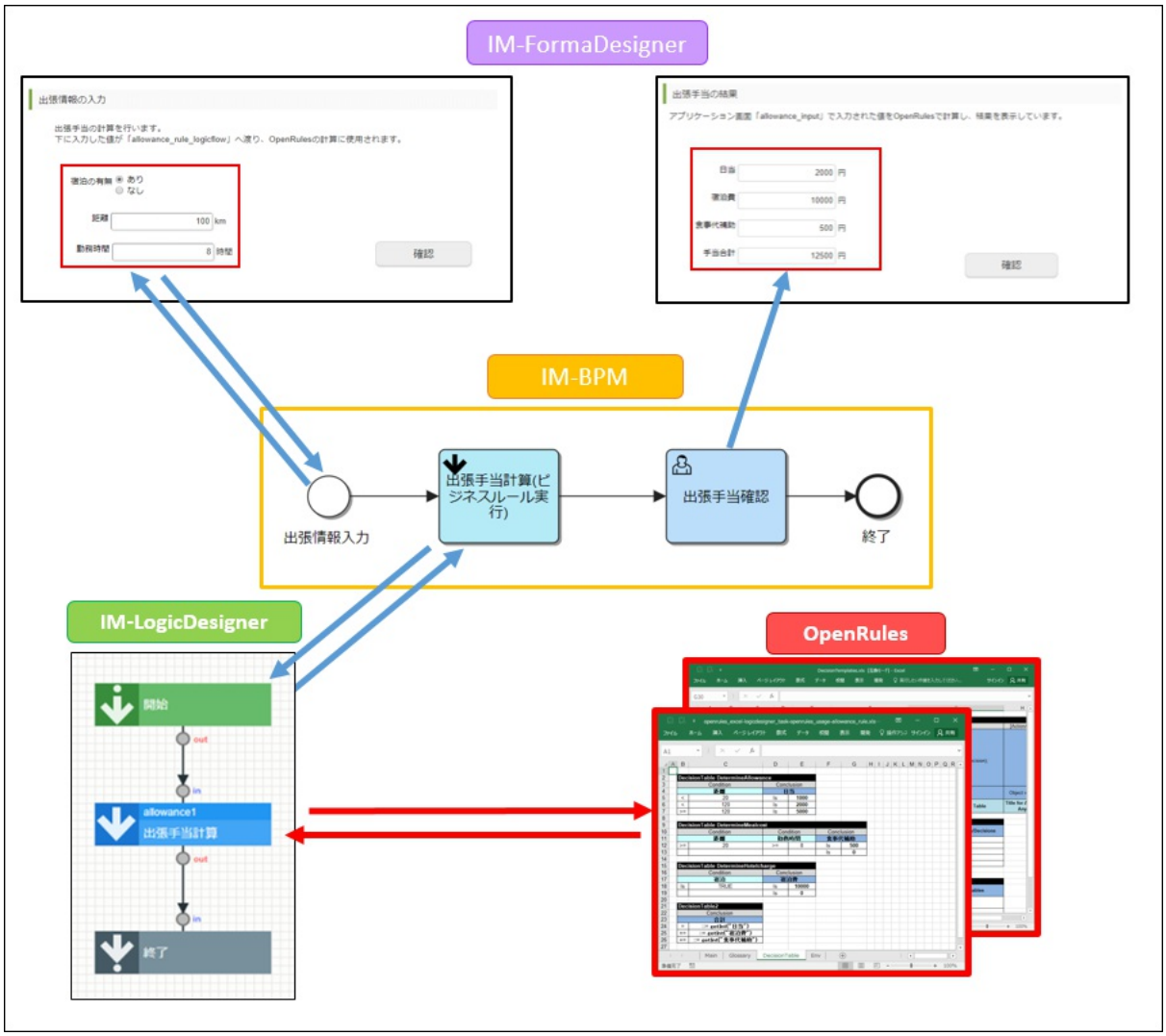
コラム
- プロセスのデプロイ方法については「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のデプロイ」を参照してください。
- デプロイしたプロセスの実行方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスの開始」を参照してください。
- 「プロセス詳細」画面への遷移方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスを確認する」を参照してください。
- 「変数一覧」画面の操作方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスの変数を確認する」を参照してください。

IM-LogicDesignerタスクを利用してOpenRulesを使用する

このチュートリアルでは、「OpenRules」と連携した「IM-LogicDesigner」のタスクを使用するプロセス定義を解説します。

- 「OpenRules」を使用することにより、プログラムに詳しくない人でも処理の分岐やビジネスロジックを設定・修正できます。
- 「IM-LogicDesignerタスク」は、タスク中に設定された情報からIM-LogicDesignerで定義されたロジックフローの処理を行います。

「OpenRules」の詳細については、「Product File Downloadサイト」の「OPENRULES ユーザ゛・マニュアル」参照してください。
「IM-LogicDesignerタスク」の基本的な使用方法については「*IM-LogicDesignerタスクを利用してユーザ情報を取得する*」を参照してください。



図：概要

「OpenRules」には、「ルールを定義するEXCELファイル」と「処理を実行するEXCELファイル」の2種類が必要です。
「ルールを定義するEXCELファイル」はユーザモジュールに内包しているものを使用します。
このチュートリアルでは、使用するルールの定義に沿った「処理を実行するEXCELファイル」の説明を行いますが、作成手順の解説は行いません。
「処理を実行するEXCELファイル」は以下をダウンロードして使用してください。

openrules_excel-logicdesigner_task_openrules_usage-allowance_rule.xls

プロセスを進めていく中で、「IM-LogicDesigner」のロジックフローと「IM-FormaDesigner for Accel Platform」で作成したアプリケーション画面を使用します。
チュートリアルを開始する前に以下の資材をインポートしてください。

im_logic_designer-logicdesigner_task_openrules_usage-allowance_rule_logic_flow.zip
im_forma_designer-logicdesigner_task_openrules_usage-allowance_input.zip
im_forma_designer-logicdesigner_task_openrules_usage-allowance_confirm.zip

> コラム
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
>
> **logicdesigner_task_openrules_usage-allowance.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」 - 「プロセス定義のアップロード」を参照してください。

コラム

各種インポート方法については、以下のリンクを参照してください。

ロジックフロー定義：「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」
「IM-FormaDesigner」で作成したアプリケーション：「IM-FormaDesigner 作成者操作ガイド」-「インポート・エクスポートを利用した IM-FormaDesigner のアプリケーションやデータソース定義の移行」

- 環境構築を行う
- EXCELファイルを確認・配置する
- ロジックフローを確認する
- プロセス定義を作成する
- 実行結果を確認する

環境構築を行う

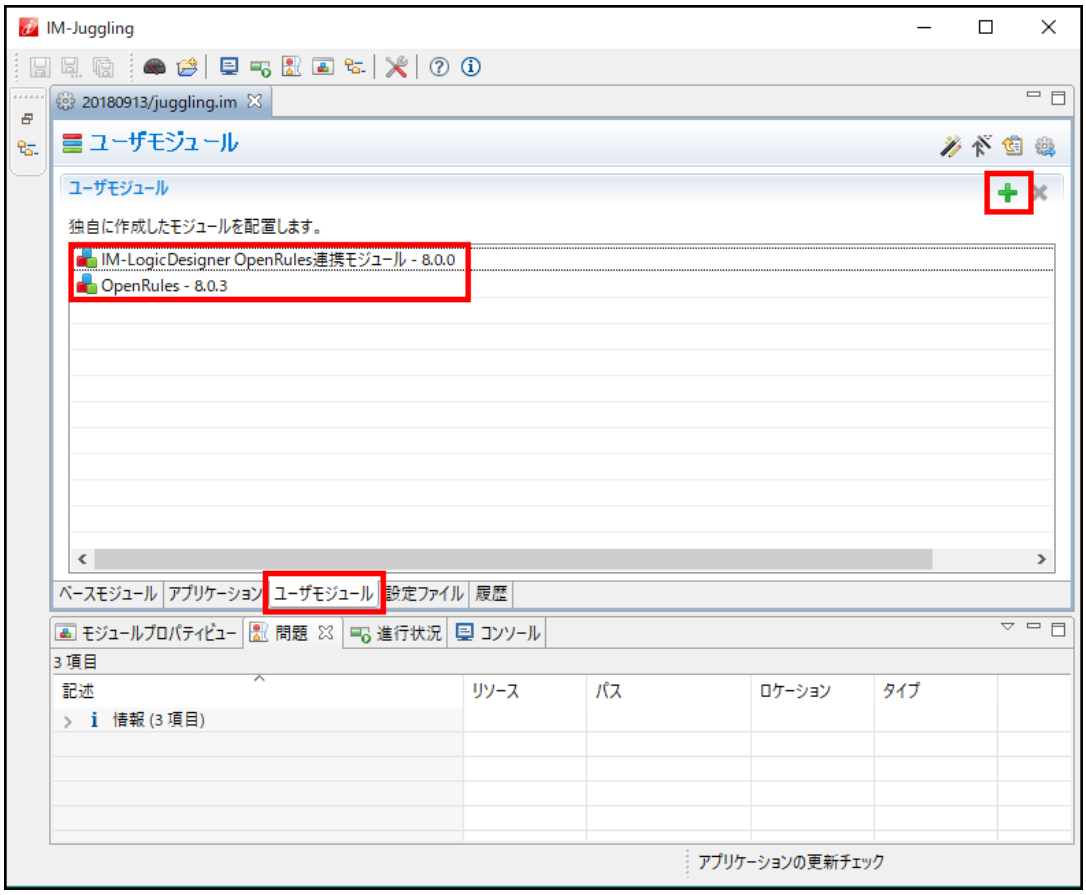
「IM-LogicDesigner」で「OpenRules」を使用するためには、専用の「ユーザモジュール」を追加する必要があります。
以下の手順のとおりに、環境構築をおこなってください。

1. モジュールを取得します。
   「Product File Downloadサイト」から、プロダクトファイルダウンロード画面を開きます。
2. OpenRules用のライセンスキーを入力し、ファイル一覧取得 （Get FileList）をクリックします。
3. 以下のユーザモジュールをダウンロードしてください。
   - openrules_modules_6.4.2.zip
     openrules_modules_6.4.2.zipを解凍し、 jp.co.intra_mart.oss_linkage.openrules-8.0.3.imm を取得します。
   - im_logic_openrules-8.0.0.imm
     上記のバージョンを選択してください。
4. 「IM-Juggling」を起動し、プロジェクトの新規作成を行います。
   「モジュールの選択」を行うところまで作業を進めてください。
   プロジェクトの作成とモジュールの選択の方法については、「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」を参照してください。

   注意

   ご利用になるバージョンによっては、対応していないことがあります。
   このチュートリアルでは「intra-mart Accel Platform 2018 Summer(Tiffany) 」を使用しています。

5. 「ユーザモジュール」タブの右上の「  」をクリックします。
   上記でダウンロードした、以下のモジュールを追加してください。
   - jp.co.intra_mart.oss_linkage.openrules-8.0.3.imm
   - im_logic_openrules-8.0.0.imm



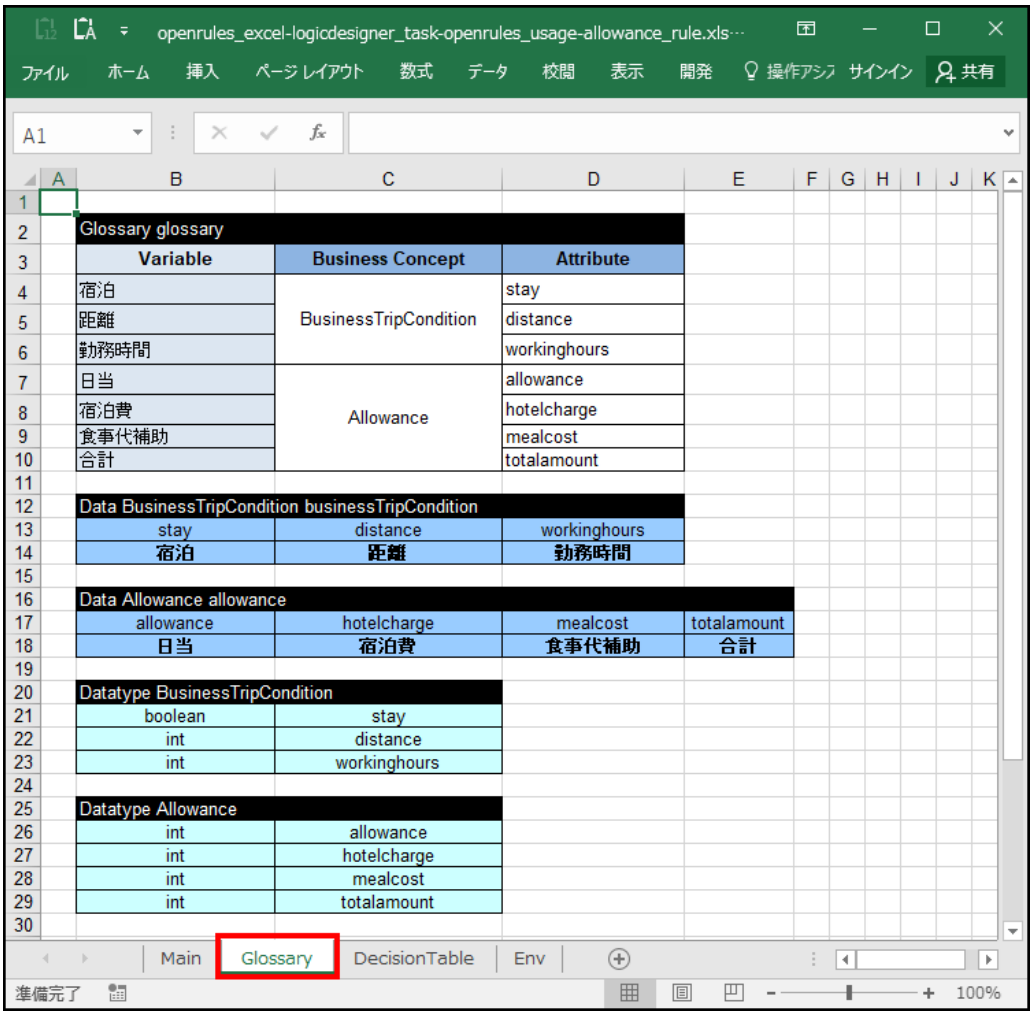
図：「IM-Juggling」 - 「ユーザモジュール」

6. 通常と同じ手順でセットアップを行ってください。
   手順については「intra-mart Accel Platform セットアップガイド」を参照してください。

EXCELファイルを確認・配置する

「OpenRules」で使用する「処理を実行するEXCELファイル」を確認・配置します。
このチュートリアルでは、「IM-FormaDesigner」のアプリケーション画面で入力された値をもとに、「出張手当」を算出するビジネスロジックが作成されています。

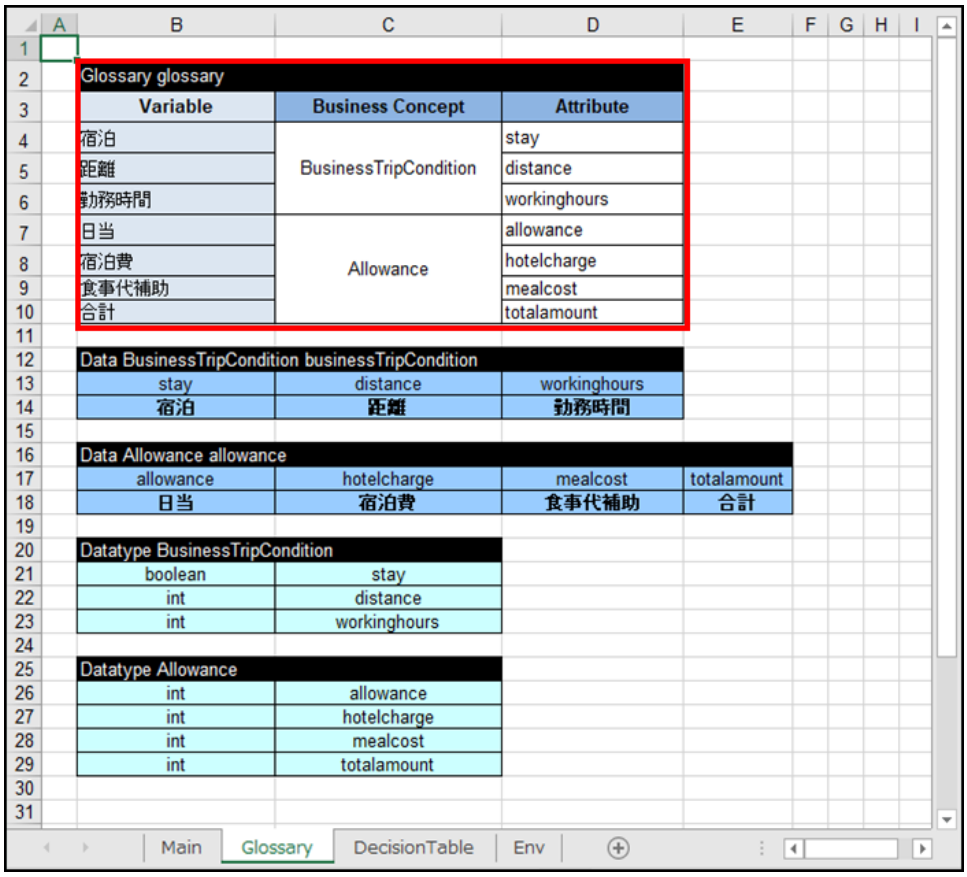
1. 2枚目のシートの「Glossary」を確認します。



	A	B	C	D	E
2		Glossary glossary			
3		Variable	Business Concept	Attribute	
4		宿泊	BusinessTripCondition	stay	
5		距離		distance	
6		勤務時間		workinghours	
7		日当	Allowance	allowance	
8		宿泊費		hotelcharge	
9		食事代補助		mealcost	
10		合計		totalamount	
12		Data BusinessTripCondition businessTripCondition			
13		stay	distance	workinghours	
14		宿泊	距離	勤務時間	
16		Data Allowance allowance			
17		allowance	hotelcharge	mealcost	totalamount
18		日当	宿泊費	食事代補助	合計
20		Datatype BusinessTripCondition			
21		boolean	stay		
22		int	distance		
23		int	workinghours		
25		Datatype Allowance			
26		int	allowance		
27		int	hotelcharge		
28		int	mealcost		
29		int	totalamount		

図：「openrules_excel-logicdesigner_task_openrules_usage-allowance_rule.xls」 - 「Glossary」シート

- Glossaryテーブル
  「ルールを定義するEXCELファイル」で利用している「項目の名称（論理名と物理名）」をマッピングしています。
  「IM-BPM」の「開始イベント」で呼び出した「IM-FormaDesigner」のアプリケーション画面で入力された値を「OpenRules」で計算し、結果を返します。
  - Glossary glossary
    「OpenRules」 で利用する全ての項目をここで定義しています。



Glossary glossary

Variable	Business Concept	Attribute
宿泊	BusinessTripCondition	stay
距離		distance
勤務時間		workinghours
日当	Allowance	allowance
宿泊費		hotelcharge
食事代補助		mealcost
合計		totalamount

Data BusinessTripCondition businessTripCondition

stay	distance	workinghours
宿泊	距離	勤務時間

Data Allowance allowance

allowance	hotelcharge	mealcost	totalamount
日当	宿泊費	食事代補助	合計

Datatype BusinessTripCondition

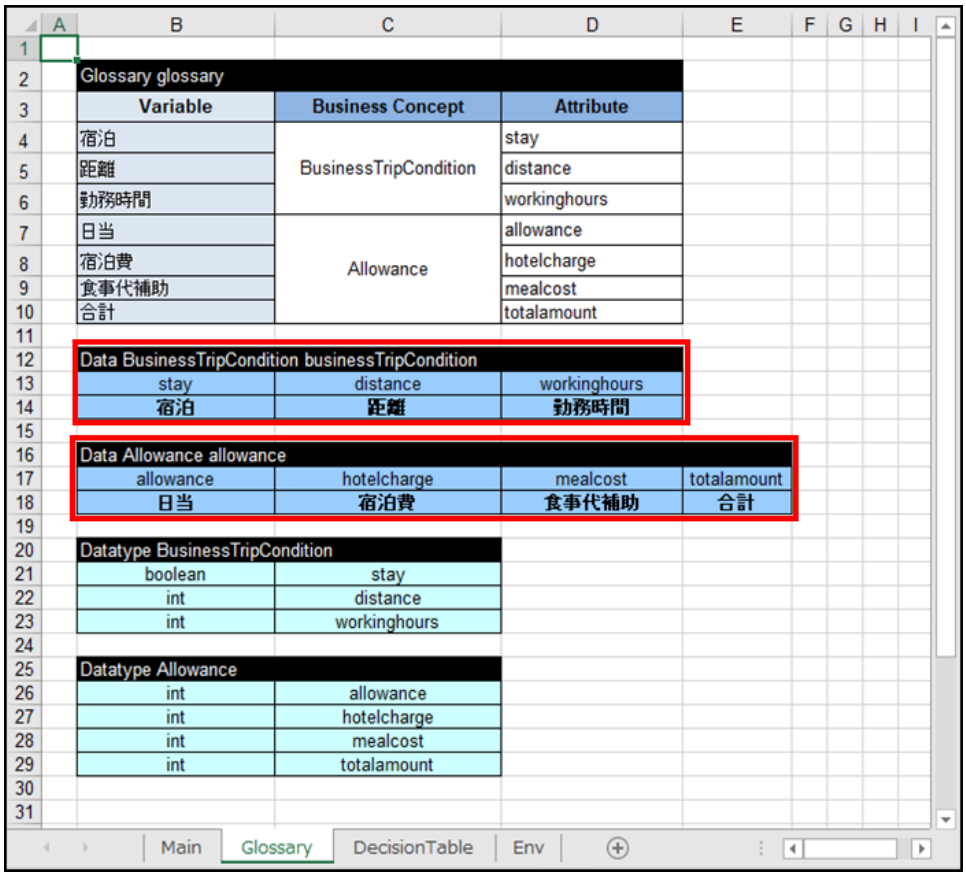
boolean	stay
int	distance
int	workinghours

Datatype Allowance

int	allowance
int	hotelcharge
int	mealcost
int	totalamount

図：「Glossary」シート - 「Glossary glossary」

- Data テーブル
  データを保持しているテーブルです。
  使用する「グループ（Business Concept）」と、「物理名（Attribute）」を設定します。
  - Data BusinessTripCondition businessTripCondition
    「IM-FormaDesigner」のアプリケーション画面で入力され、「IM-LogicDesigner」から渡される値です。
  - Data Allowance allowance
    「DecisionTable」シートで算出し、「IM-LogicDesigner」を通して「IM-FormaDesigner」のアプリケーション画面に返却される表示結果の値です。



Glossary glossary

Variable	Business Concept	Attribute
宿泊	BusinessTripCondition	stay
距離		distance
勤務時間		workinghours
日当	Allowance	allowance
宿泊費		hotelcharge
食事代補助		mealcost
合計		totalamount

Data BusinessTripCondition businessTripCondition

stay	distance	workinghours
宿泊	距離	勤務時間

Data Allowance allowance

allowance	hotelcharge	mealcost	totalamount
日当	宿泊費	食事代補助	合計

Datatype BusinessTripCondition

boolean	stay
int	distance
int	workinghours

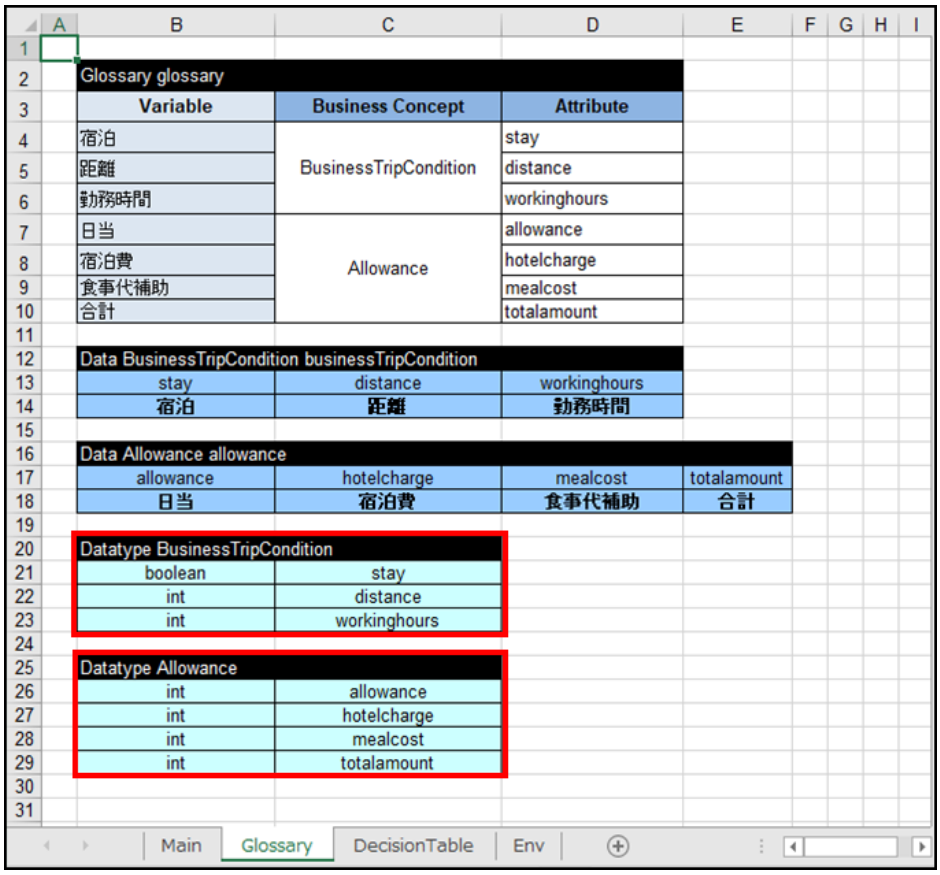
Datatype Allowance

int	allowance
int	hotelcharge
int	mealcost
int	totalamount

Main | Glossary | DecisionTable | Env

図：「Glossary」シート - 「Data」テーブル

- Datatypeテーブル
  「Glossary」テーブルで設定した「グループ（Business Concept）」のデータ型を指定しています。
  このテーブルは、グループ（Business Concept）単位に作成されています。
  - Datatype BusinessTripCondition
    「IM-LogicDesigner」から受け取る値のデータ型を指定しています。
  - Datatype Allowance
    「IM-LogicDesigner」に返却する値のデータ型を指定しています。



Glossary glossary		
Variable	Business Concept	Attribute
宿泊	BusinessTripCondition	stay
距離		distance
勤務時間		workinghours
日当	Allowance	allowance
宿泊費		hotelcharge
食事代補助		mealcost
合計		totalamount

Data BusinessTripCondition businessTripCondition		
stay	distance	workinghours
宿泊	距離	勤務時間

Data Allowance allowance			
allowance	hotelcharge	mealcost	totalamount
日当	宿泊費	食事代補助	合計

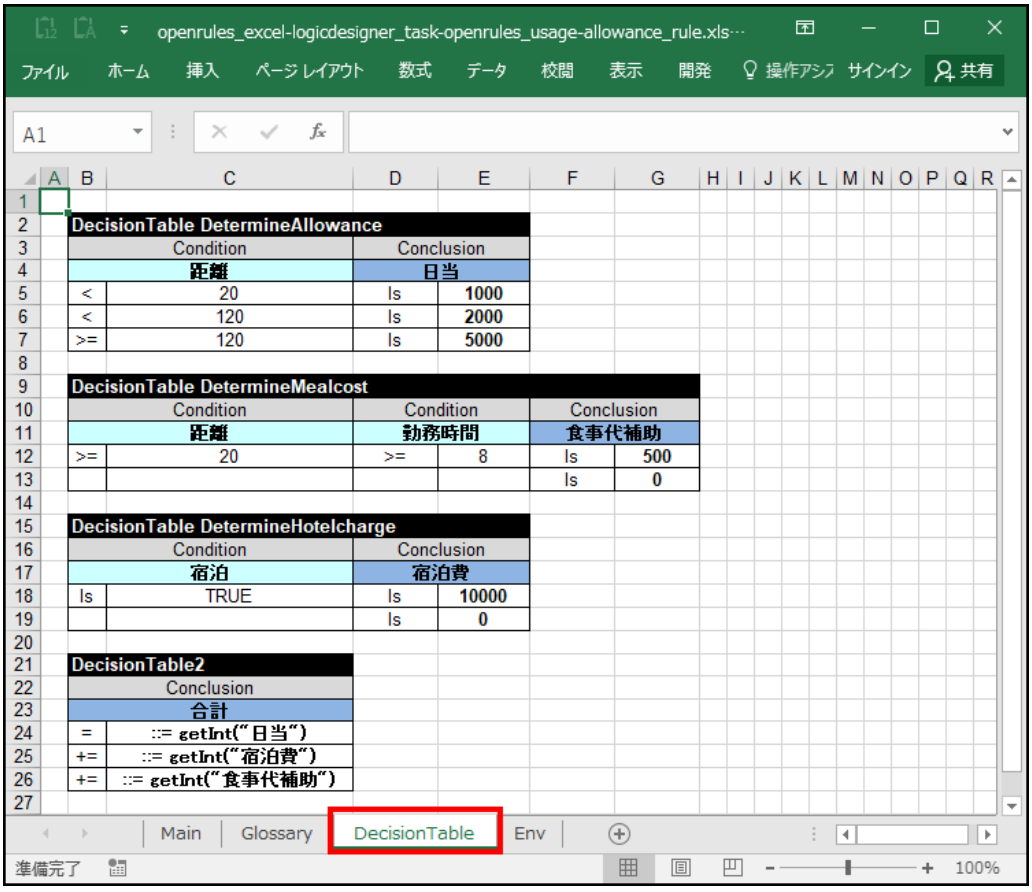
Datatype BusinessTripCondition	
boolean	stay
int	distance
int	workinghours

Datatype Allowance	
int	allowance
int	hotelcharge
int	mealcost
int	totalamount

Main | Glossary | DecisionTable | Env

図：「Glossary」シート - 「Datatype」テーブル

2. 3枚目のシートの「DecisionTable」を確認します。



openrules_excel-logicdesigner_task-openrules_usage-allowance_rule.xls…

ファイル　ホーム　挿入　ページ レイアウト　数式　データ　校閲　表示　開発　操作アシス　サインイン　共有

A1

DecisionTable DetermineAllowance

Condition		Conclusion	
距離		日当	
<	20	Is	1000
<	120	Is	2000
>=	120	Is	5000

DecisionTable DetermineMealcost

Condition		Condition		Conclusion	
距離		勤務時間		食事代補助	
>=	20	>=	8	Is	500
				Is	0

DecisionTable DetermineHotelcharge

Condition		Conclusion	
宿泊		宿泊費	
Is	TRUE	Is	10000
		Is	0

DecisionTable2

Conclusion
合計
= ::= getInt("日当")
+= ::= getInt("宿泊費")
+= ::= getInt("食事代補助")

Main | Glossary | DecisionTable | Env

準備完了　100%

図：「openrules_excel-logicdesigner_task_openrules_usage-allowance_rule.xls」 - 「DecisionTable」シート

- DecisionTableテーブル
  実行の条件と設定する値が定義されているテーブルです。

- DecisionTable DetermineAllowance
  入力された「距離」を評価し、「日当」を決定します。
  - 入力された値が「20Km」未満（<）なら、日当は「1000」
  - 入力された値が「120Km」未満（<）なら、日当は「2000」
  - 入力された値が「120Km」以上（>=）なら、日当は「5000」

DecisionTable DetermineAllowance			
Condition		Conclusion	
距離		日当	
<	20	ls	1000
<	120	ls	2000
>=	120	ls	5000

図：「DecisionTable」テーブル - 「DetermineAllowance」

- DecisionTable DetermineMealcost
  入力された勤務時間を評価し、「食事代補助」を決定します。
  - 入力された値が「20km」以上（>=）なら、食事代補助は「500」
  - 入力された値が上記の条件に当てはまらないなら、食事代補助は「0」

DecisionTable DetermineMealcost					
Condition		Condition		Conclusion	
距離		勤務時間		食事代補助	
>=	20	>=	8	ls	500
				ls	0

図：「DecisionTable」テーブル - 「DetermineMealcost」

- DecisionTable DetermineHotelcharge
  ラジオボタンで設定された「宿泊の有無」を判定し、宿泊費を決定します。
  - ラジオボタンが宿泊あり（TRUE）になっていた場合、宿泊費は「1000」
  - 上記の条件に当てはまらなかった場合、宿泊費は「0」

DecisionTable DetermineHotelcharge			
Condition		Conclusion	
宿泊		宿泊費	
ls	TRUE	ls	10000
		ls	0

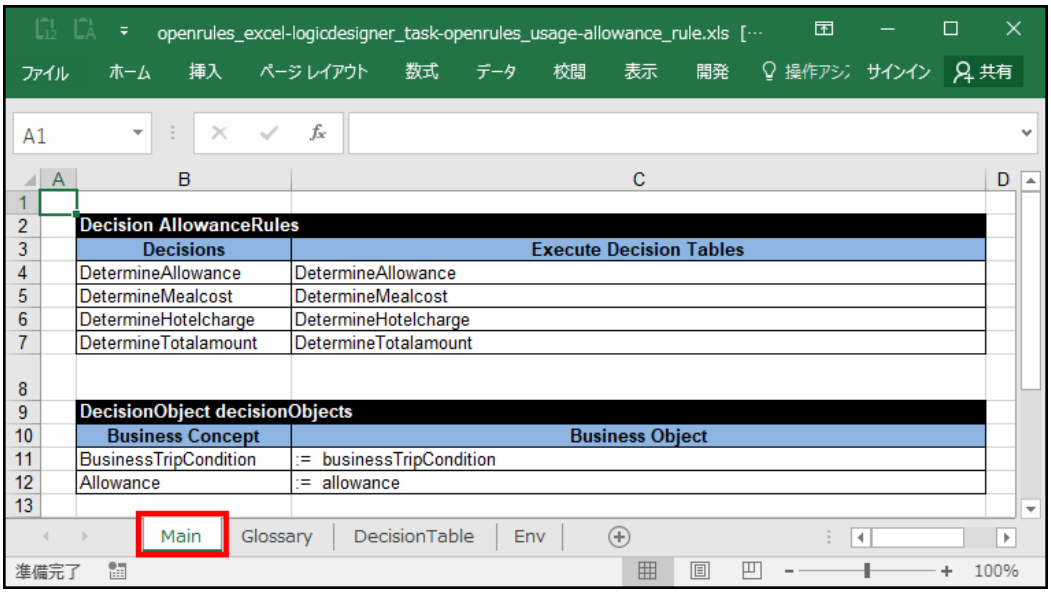
図：「DecisionTable」テーブル - 「DetermineHotelcharge」

- DecisionTable2テーブル
  実行の条件と設定する値が定義されているテーブルです。
  - DecisionTable2 DetermineTotalamount
    このチュートリアルでは、列「Condition」が作成されていないため、列「Conclusion」の式すべてが実行されます。
    - 列「Conclusion」
      上記3つの「DecisionTable」テーブルで決定した値を使用して「合計」を算出しています。

DecisionTable2 DetermineTotalamount	
Conclusion	
合計	
=	::= getInt("日当")
+=	::= getInt("宿泊費")
+=	::= getInt("食事代補助")

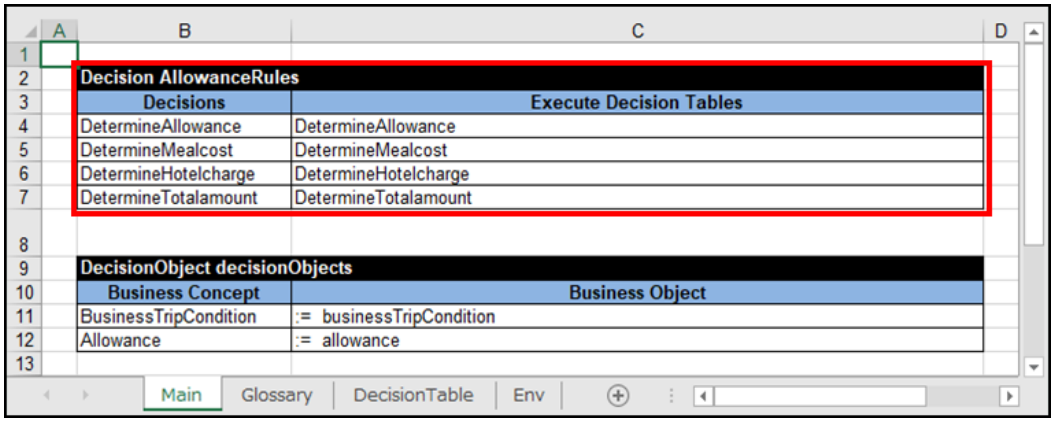
図：「DecisionTable2」テーブル - 「DetermineTotalamount」

3. 1枚目のシートの「Main」を確認します。



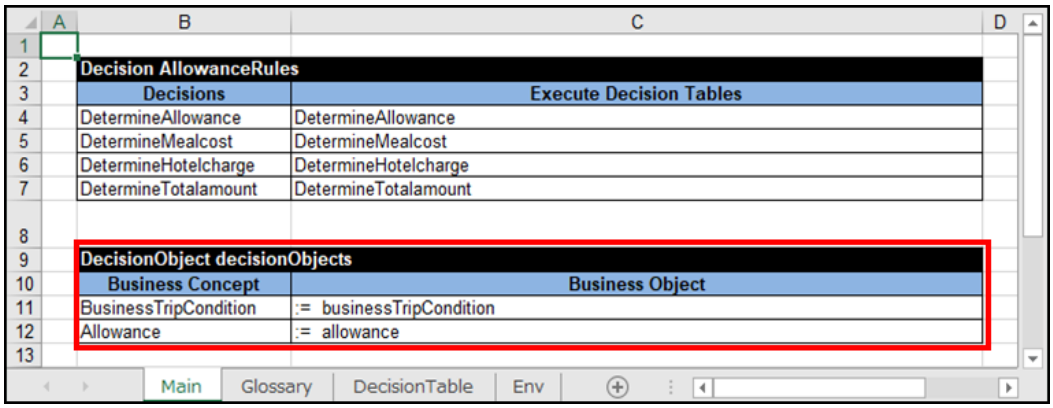
図：「openrules_excel-logicdesigner_task_openrules_usage-allowance_rule.xls」 - 「Main」シート

- Decision テーブル
  「DecisionTable」シートに定義されているテーブルの実行条件や順番を設定します。
  実行結果を出力（返却）項目のグループ（オブジェクト）に設定します。
  - Decision AllowanceRules
    ここで、評価に使用する「DecisionTable」を設定します。
    ここで設定した順番で「DecisionTable」が実行されます。
    - 列「Decisions」
      「DecisionTable」テーブルを使用するため、名前を付けます。自由に命名できます。
    - 列「Execute Decision Tables」
      「DecisionTable」シートで定義した「DecisionTable」テーブルの名前です。
      - DetermineAllowance：距離に対応した「日当」の判定
      - DetermineMealcost：勤務時間に対応した、「食事代補助」の判定
      - DetermineHotelcharge：宿泊の有無による「宿泊費」の判定
      - DetermineTotalamount：上記の「合計」の算出



図：「Decision」テーブル - 「AllowanceRules」

- DecisionObject テーブル
  ルールで利用する項目を、一定のグループ（オブジェクト）単位に受け渡しを行います。
  - DecisionObject decisionObjects
    「IM-LogicDesigner」から渡される「BusinessTripCondition<object> 」と、計算結果を返却する「Allowance<object>」を、Glossary で定義しているオブジェクト（Business Concept）とマッピングします。
    - 列「Business Concept」
      「DecisionTable」シートで定義した「DecisionTable」テーブルの名前です。
    - 列「Business Object」
      「Business Concept」に対する、値の入出力の式を定義するための列です。

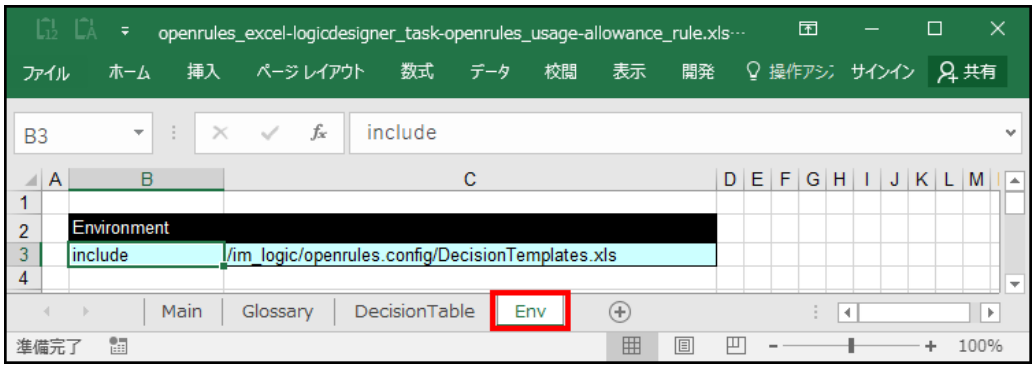


Decision AllowanceRules	
Decisions	Execute Decision Tables
DetermineAllowance	DetermineAllowance
DetermineMealcost	DetermineMealcost
DetermineHotelcharge	DetermineHotelcharge
DetermineTotalamount	DetermineTotalamount

DecisionObject decisionObjects	
Business Concept	Business Object
BusinessTripCondition	:= businessTripCondition
Allowance	:= allowance

図：「DecisionObject」テーブル - 「decisionObjects」

4. 4枚目のシート「Env」を確認します。
   - Environmentテーブル
     ルールの実行に必要なEXCELファイルやJavaのパッケージ等の情報を管理します。
     - Environment
       「ルールを定義するEXCELファイル」が置いてあるフォルダ階層とファイル名が設定されています。



図：「openrules_excel-logicdesigner_task_openrules_usage-allowance_rule.xls」 - 「Env」シート

> **コラム**
> 本テーブルの詳細な設定方法は、「OPENRULES ユーザ゛・マニュアル」参照してください。

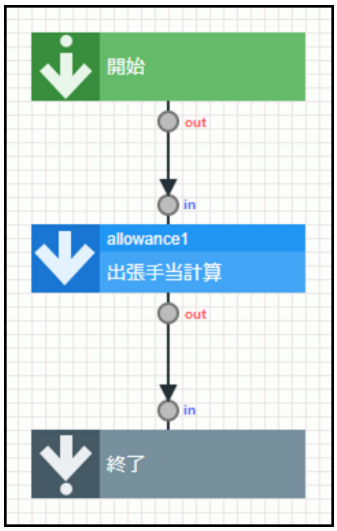
5. EXCELファイルの確認が終わったら、ファイルを「パブリックストレージ直下」に配置します。

図：「パブリックストレージ」直下

> **コラム**
>
> ストレージに対しての操作はテナント管理者で行えます。
> 詳細については、以下のリンク先を参照してください。
> 「システム管理者操作ガイド」-「ファイル操作」

ロジックフローを確認する

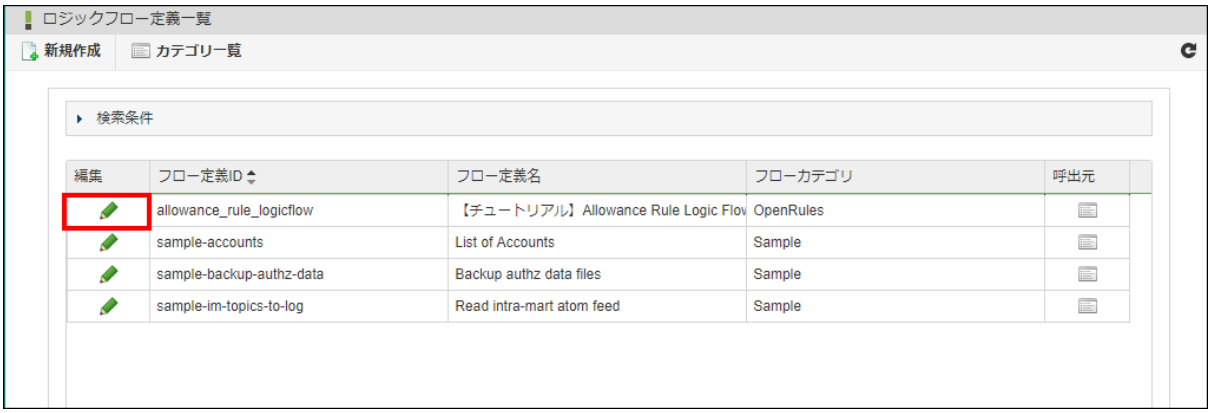
「IM-LogicDesigner」のロジックフローを確認します。
このチュートリアルでは、「IM-FormaDesigner」のアプリケーション画面で入力された値を「OpenRules」で計算し、結果を返します。



図：ロジックフロー

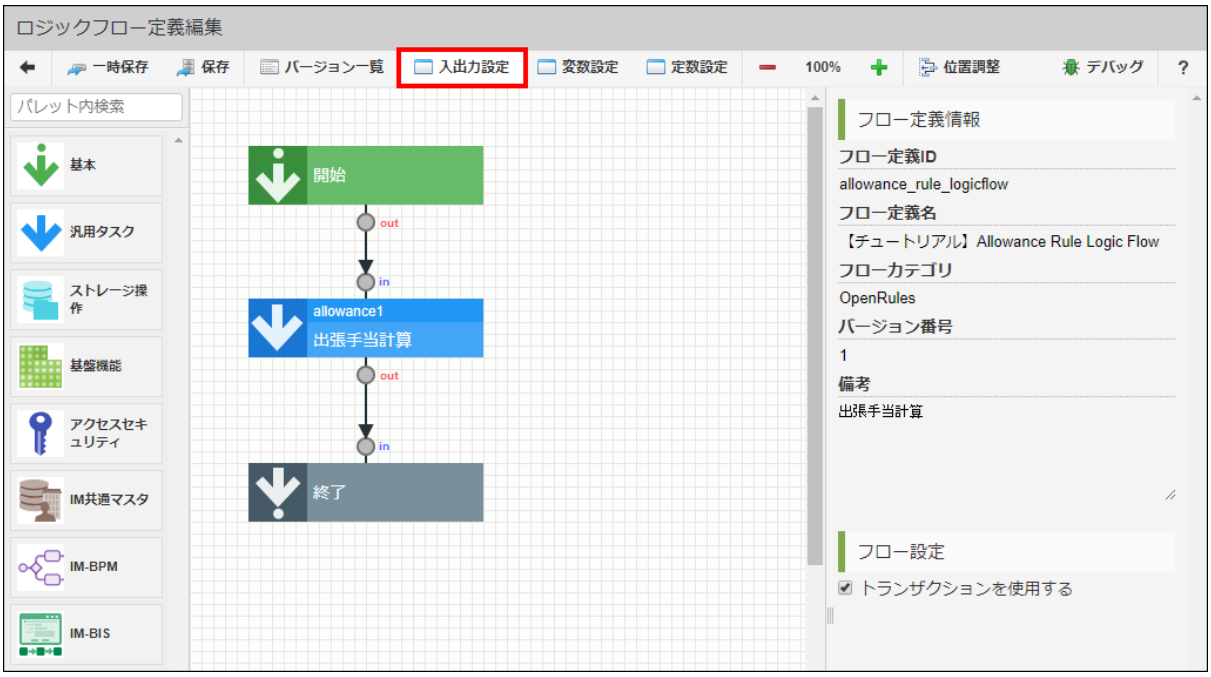
1. チュートリアル開始前にインポートしたロジックフローを確認します。
   「サイトマップ」→「IM-LogicDesigner」→「フロー定義一覧」→「ロジックフロー定義一覧」画面を表示します。

2. フロー定義ID「allowance_rule_logicflow」の「」をクリックします。



図：「ロジックフロー定義一覧」

3. 「allowance_rule_logicflow」の「ロジックフロー定義編集」画面が表示されます。
   メニューバーの「入出力設定」をクリックし、「入力設定」を表示します。



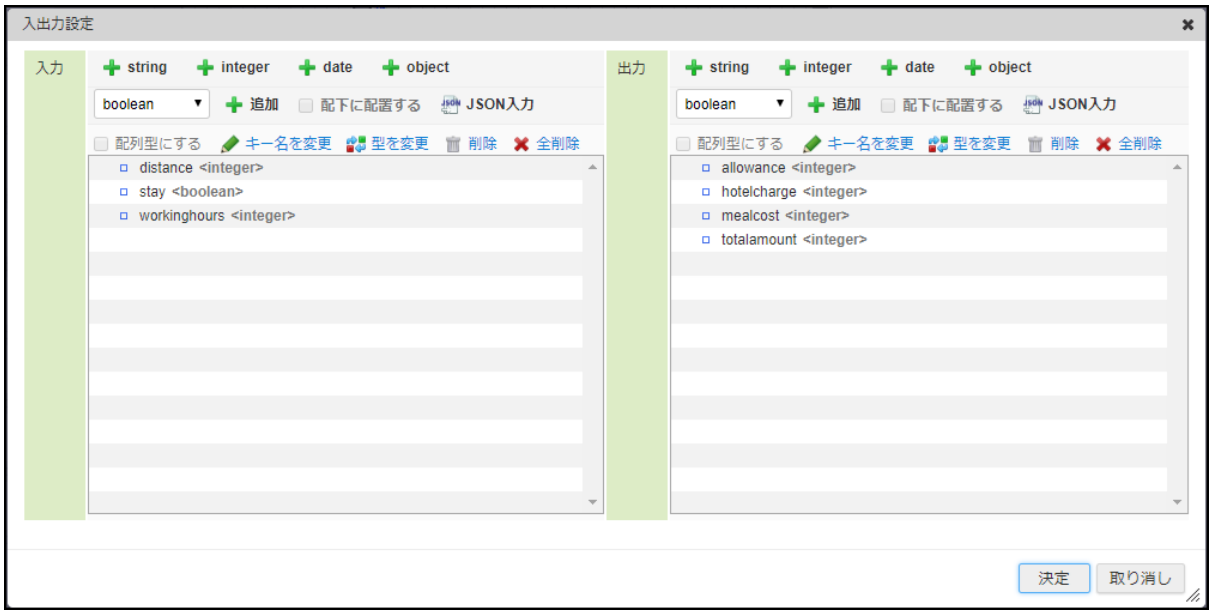
図：「ロジックフロー定義編集」

4. 設定されている「入出力値」を確認します。
   以下の値が登録されているのを確認します。
   - 入力
     「IM-FormaDesigner」のアプリケーション画面「allowance_input」で入力され、「IM-BPM」から渡ってきた値

キー名	型
distance	<integer>
stay	<boolean>
workinghours	<integer>

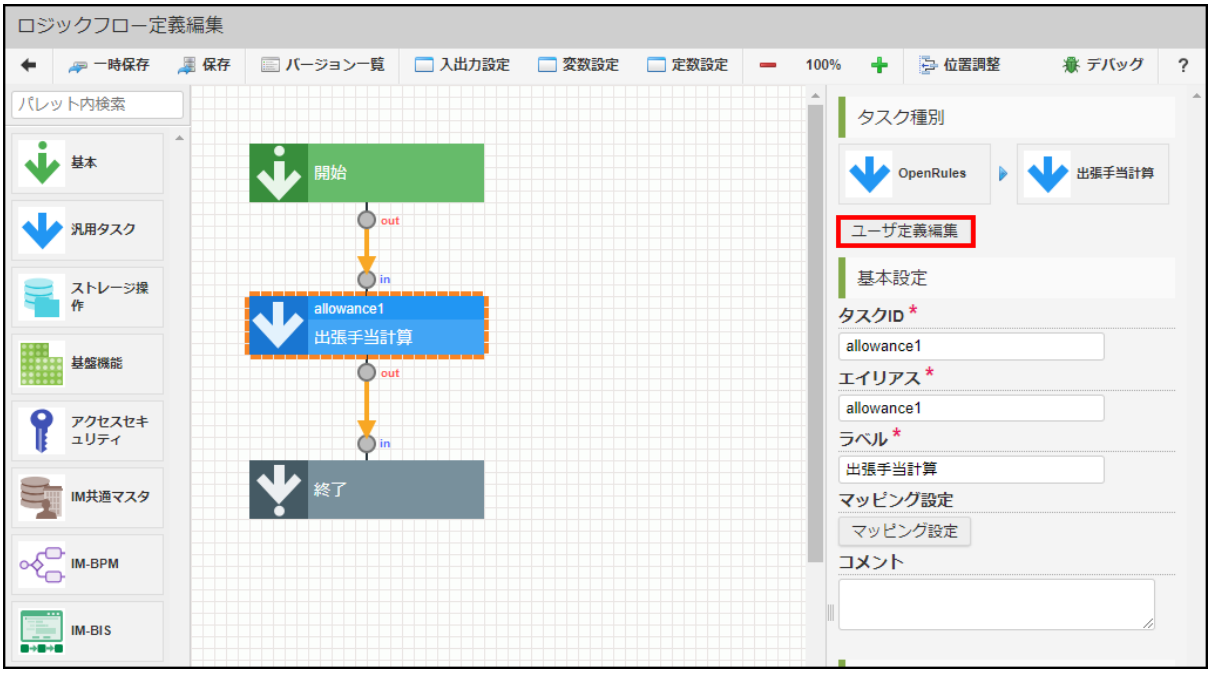
   - 出力
     「IM-FormaDesigner」のアプリケーション画面「allowance_confirm」で表示するため、「IM-BPM」に返却する値

キー名	型
allowance	<integer>
hotelcharge	<integer>
mealcost	<integer>
totalamount	<integer>



図：「入出力設定」

5. 「OpenRules」に対する設定の確認をします。
「出張手当計算（allowance1）」タスクの「ユーザ定義編集」をクリックし、「OpenRules ルール定義編集」ダイアログを表示します。



図：「ロジックフロー定義編集」

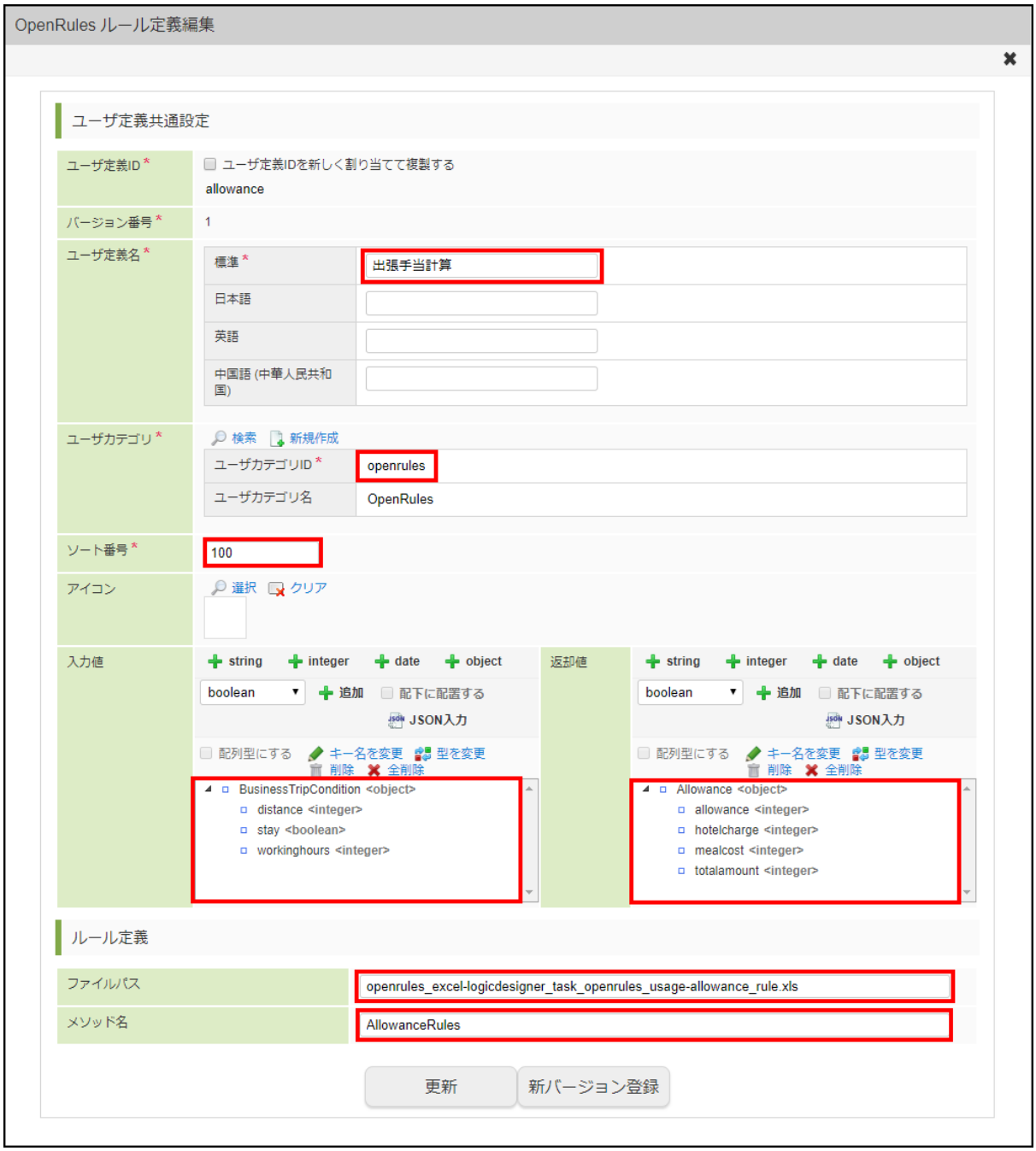
6. 「OpenRulesルール定義編集」が表示されます。
以下のとおりに項目が設定されていることを確認してください。
   - ユーザ定義共通設定
     - ユーザ定義名：出張手当計算
     - ユーザカテゴリ：任意
     - ソート番号：任意
     - 入力値

キー名	型
BusinessTripCondition	<object>
distance	<integer>
stay	<boolean>
workinghours	<integer>

     - 返却値

キー名	型
Allowance	<object>
allowance	<integer>
hotelcharge	<integer>
mealcost	<integer>
totalamount	<integer>

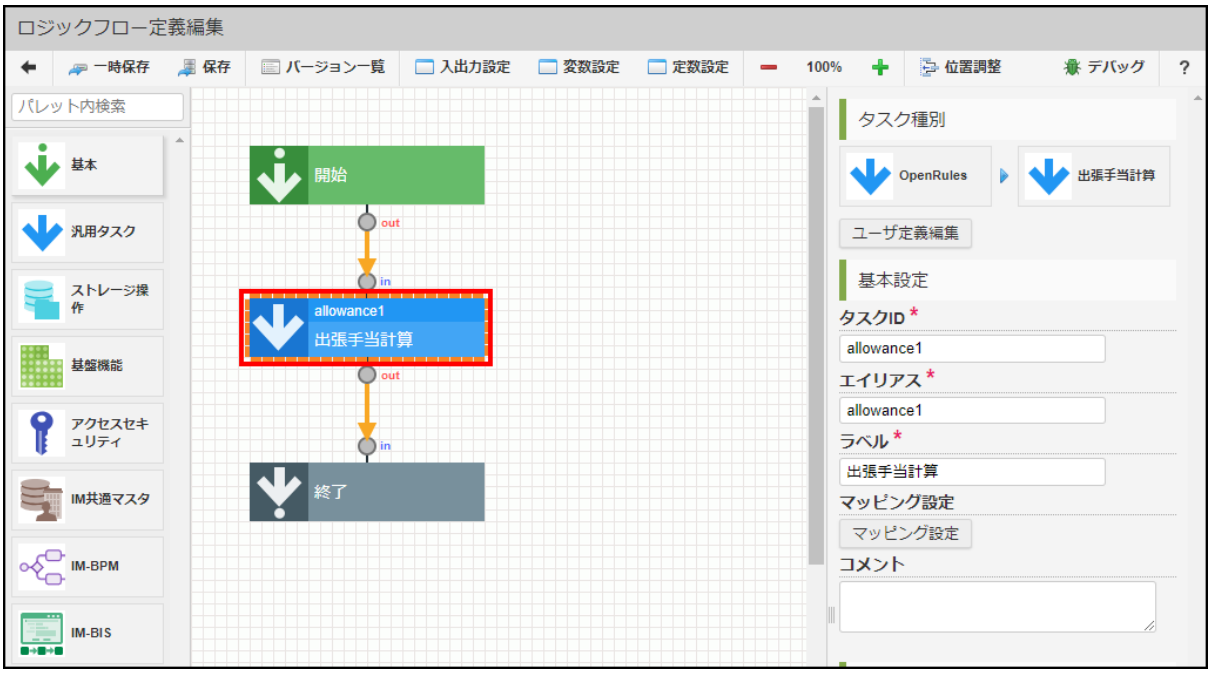
     - ルール定義
       - ファイルパス： openrules_excel-logicdesigner_task_openrules_usage-allowance_rule.xls
       - メソッド名：AllowanceRules



図：「OpenRules ルール定義編集」

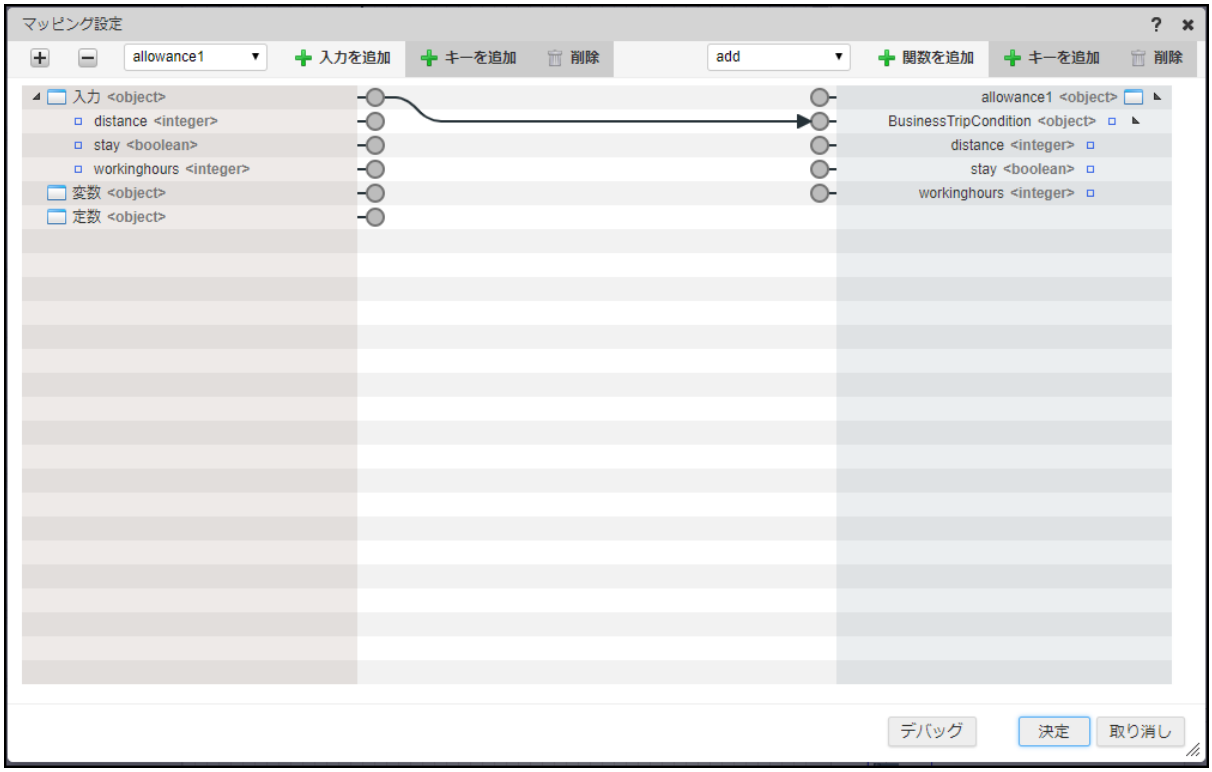
> **コラム**
>
> 「ルール定義」の「メソッド名」は、「処理を実行するEXCELファイル」の「Main」シート - 「Decision」テーブルに紐づいています。

7. 「OpenRules」を使用するタスク「出張手当計算（allowance1）」のマッピング設定を確認します。
「出張手当計算（allowance1）」タスクをクリックしてください。



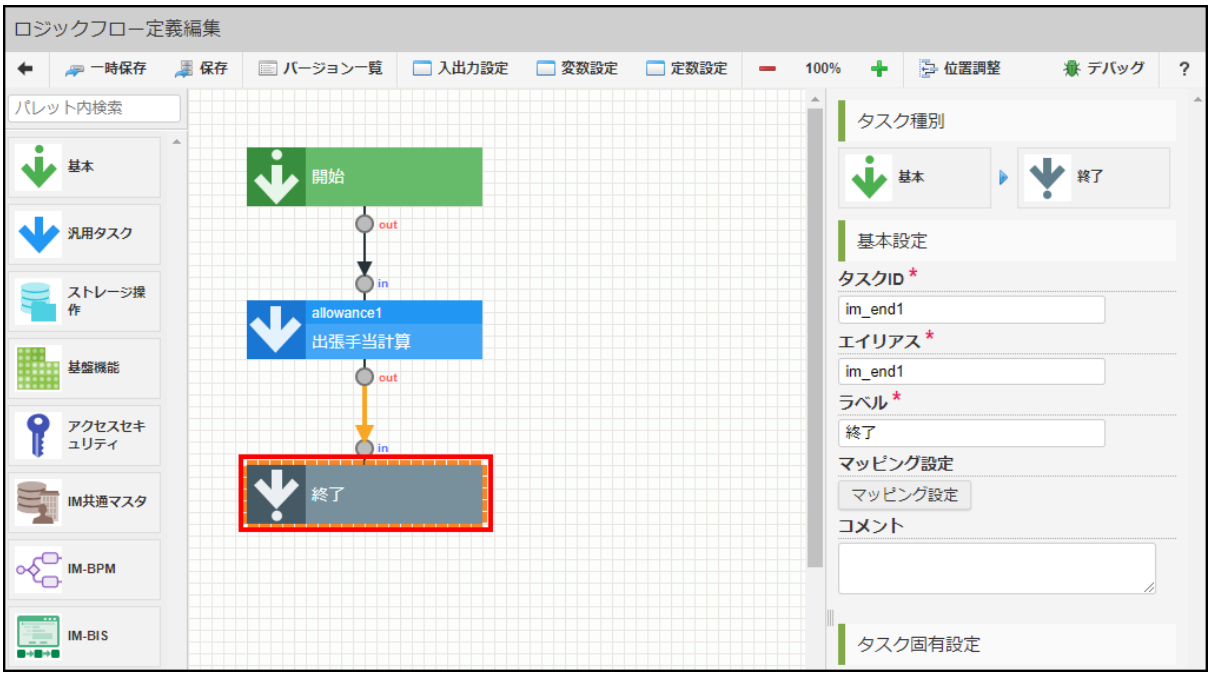
図：「ロジックフロー定義編集」

8. 左右で以下の値が登録・マッピングされていることを確認します。
   - 左側：入力<object>
     「IM-BPM」から受け取った値を、「入力<Object>」に入れています。
   - 右側：BusinesTripCondition<object>
     「Main」シートの「DecisionObject」テーブルで設定したオブジェクトに紐づけています。



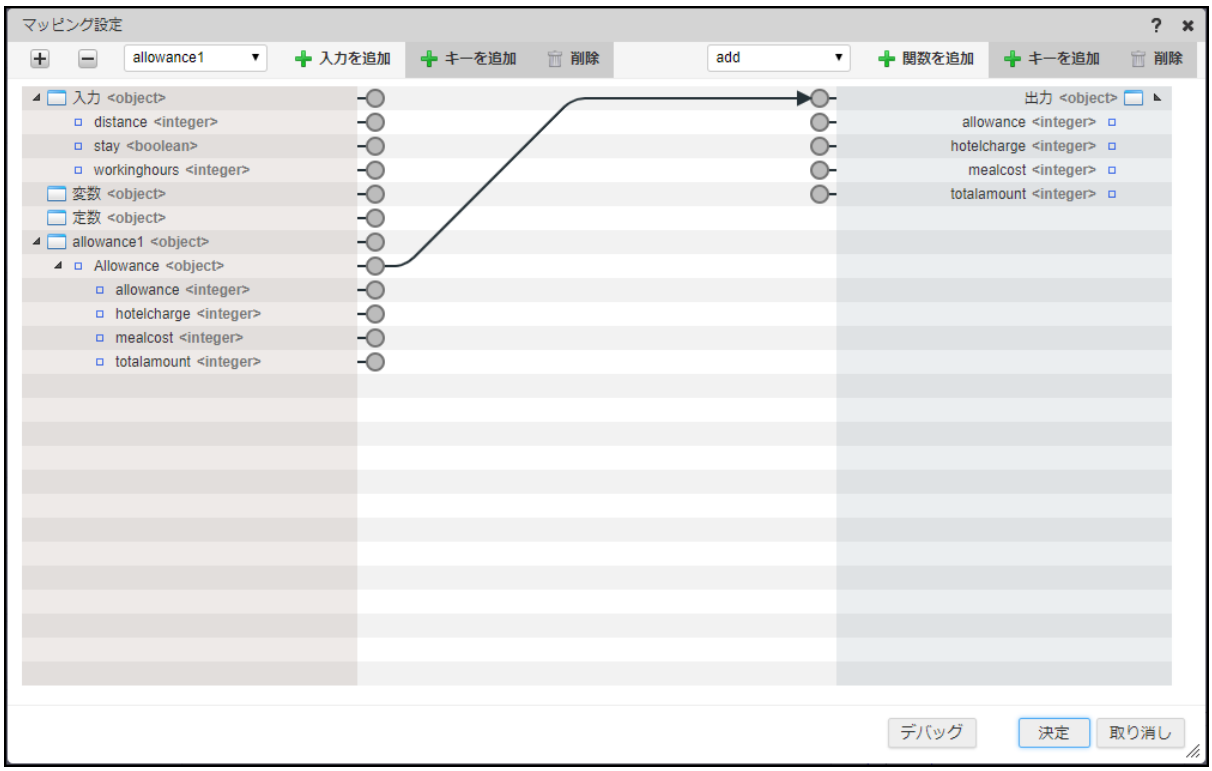
図：「マッピング設定」

9. 「終了」タスクの「マッピング設定」画面を表示します。
   「終了」タスクをクリックしてください。



図：「ロジックフロー定義編集」

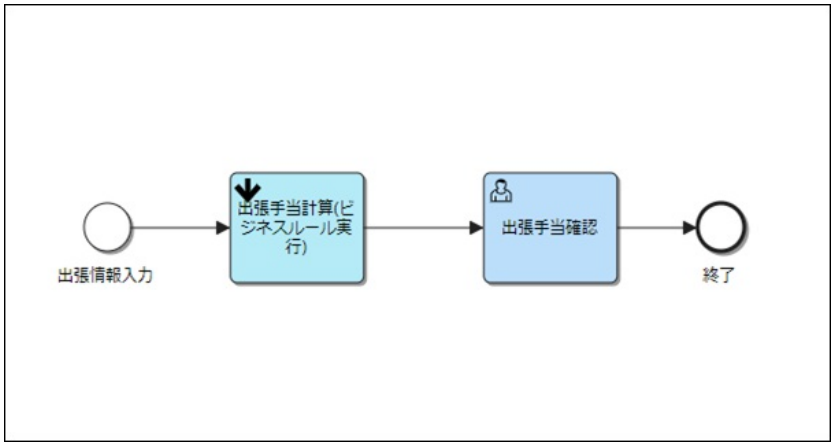
10. 左右で以下の値が登録・マッピングされていることを確認します。
    - 左側：Allowance<object>
      「OpenRules」が算出した値を「DecisionObject」テーブルで設定したオブジェクトに紐づけています。
    - 右側：出力<object>
      「IM-BPM」に返却する値を、「出力<object>」に紐づけています。



図：「マッピング設定」

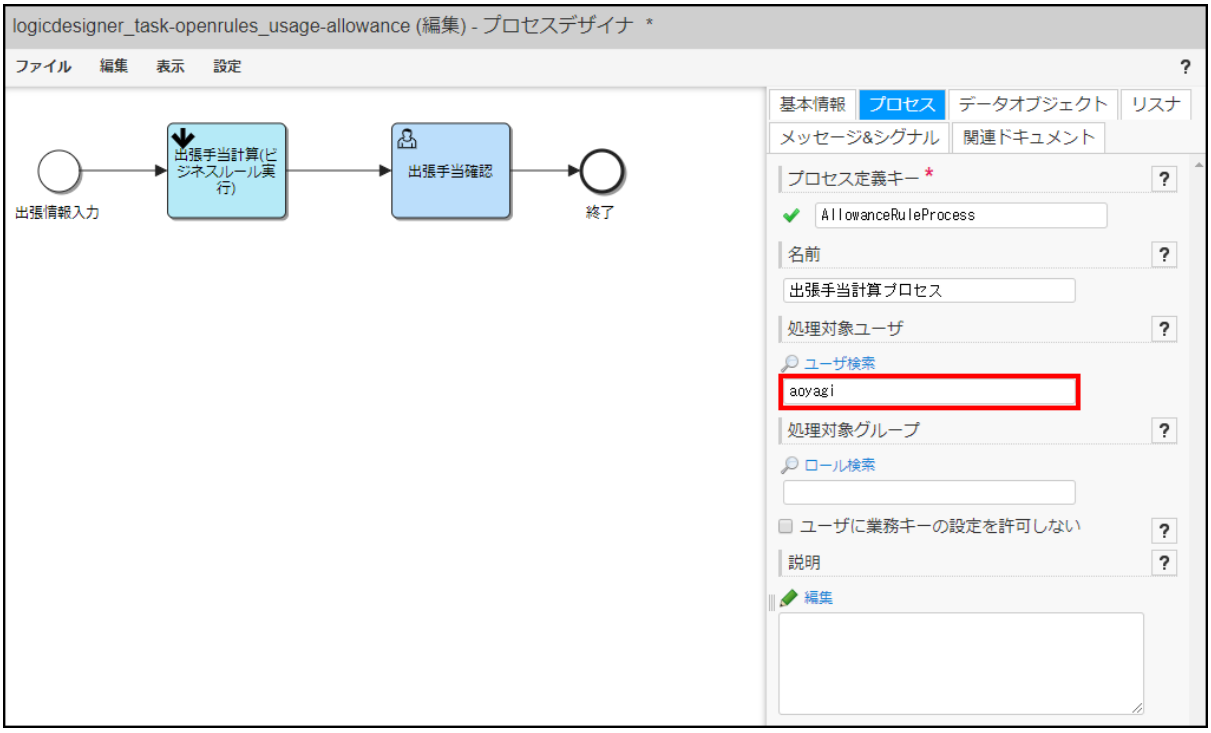
プロセス定義を作成する

上記の「IM-FormaDesigner」や「IM-LogicDesigner」を組み込んだプロセスを作成します。
このプロセスは、「IM-FormaDesigner」のアプリケーション画面に「出張情報」を入力することで、出張手当の明細を確認できます。



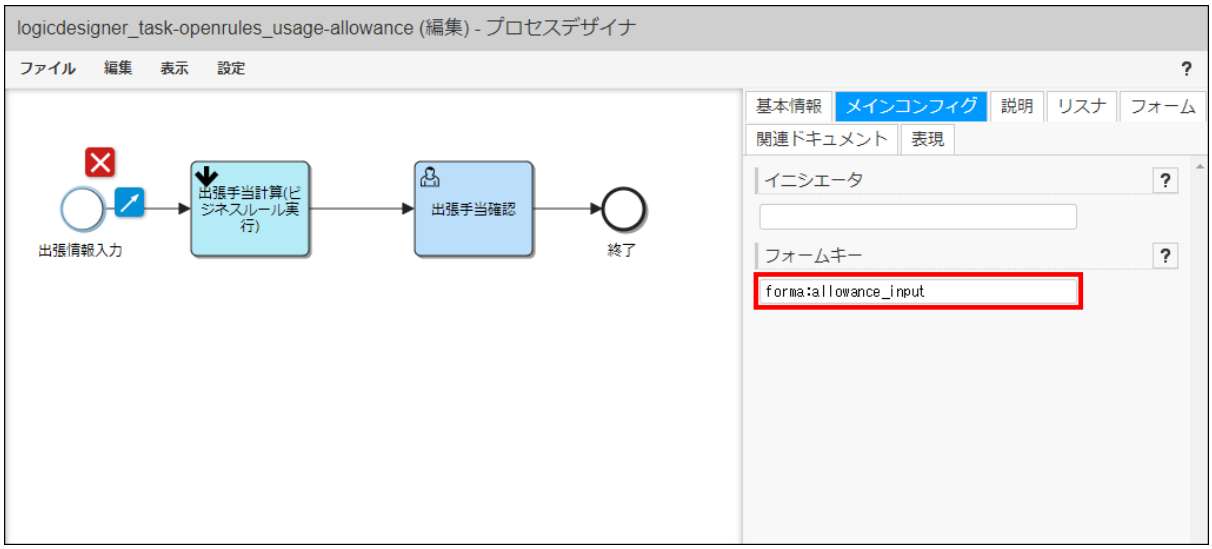
図：完成イメージ

1. 「開始イベント」を設置します。
2. プロセス全体に対する設定を行います。
   「開始イベント」ではない場所をクリックし、「プロパティ」をプロセス全体に切り替えます。
3. 「プロセス」タブから、処理対象ユーザを「青柳辰巳（ユーザコード: aoyagi ）に設定します。



図：「プロパティ」 - 「プロセス」

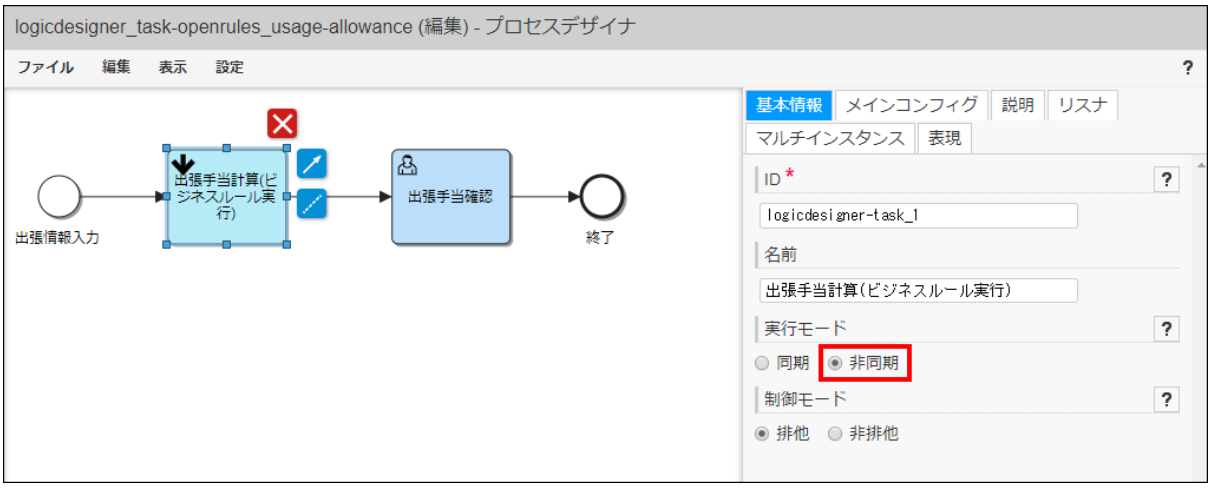
4. 「開始イベント」に、インポートした「IM-FormaDesigner」のアプリケーションを設定します。
   「開始イベント」を選択し、「メインコンフィグ」タブの「フォームキー」に以下の値を設定します。
   - フォームキー : forma:allowance_input



図：「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

5. 「OpenRules」を利用するために、「IM-LogicDesigner」を呼び出します。
   パレットの「Intra-mart」から「IM-LogicDesignerタスク」を選択し、配置します。

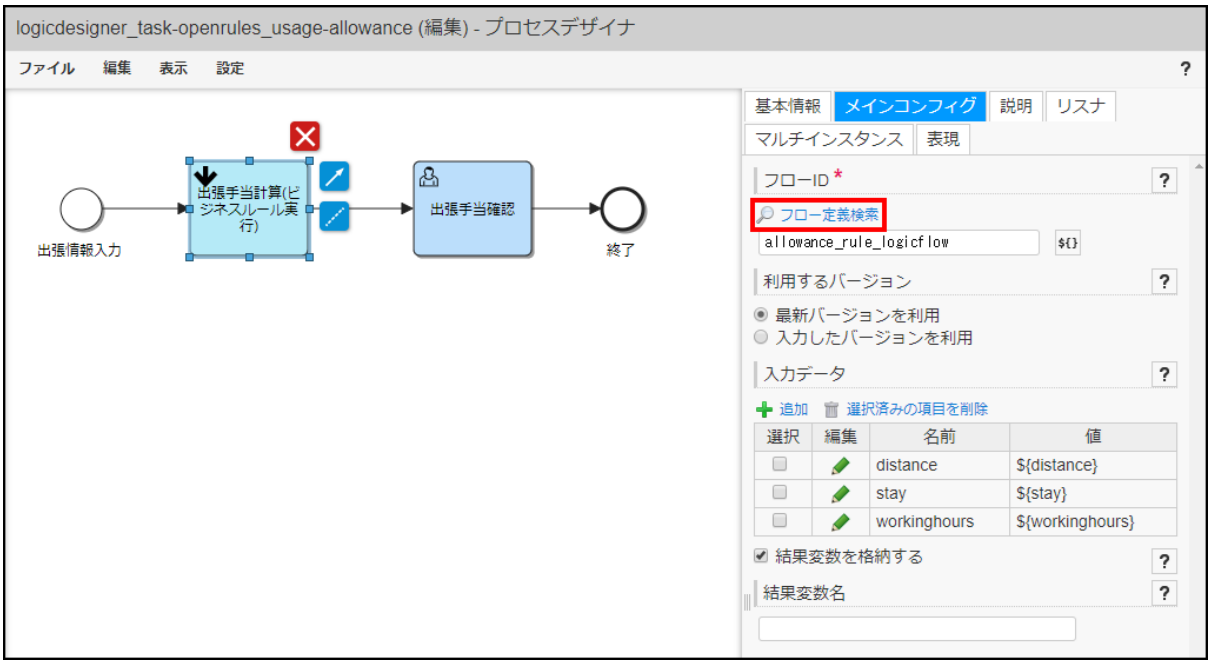
6. 「実行モード」の設定を行います。
   「基本情報」タブの「 実行モード」で、「非同期」を選択してください。



図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「基本情報」

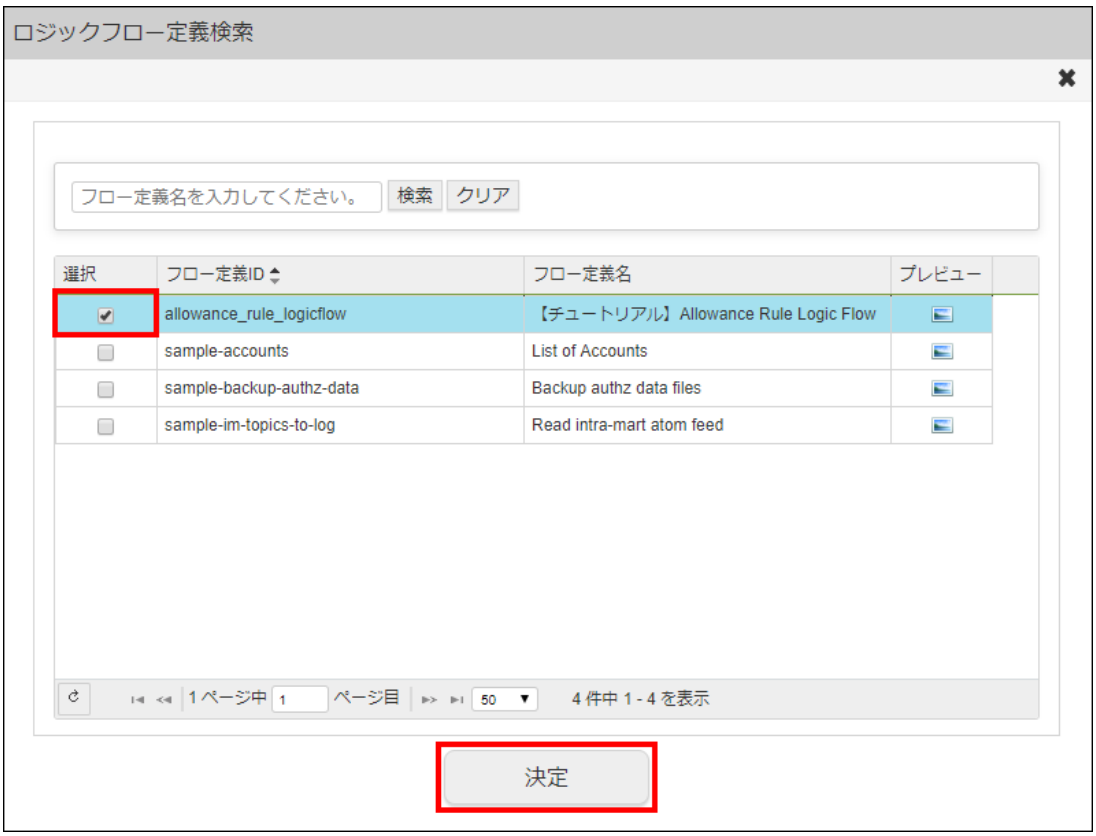
> **コラム**
>
> 「OpenRules」の実行は、時間がかかる可能性があります。
> 「IM-LogicDesignerタスク」を同期で実行すると、「OpenRules」の処理が終了するまで画面は応答しません。
> 「実行モード」を「非同期」にすることで、画面は即応答され「OpenRules」は非同期で実行されます。

7. 「フローID」を設定します。
   「メインコンフィグ」タブの「フローID」にある、「フロー定義検索」をクリックしてください。



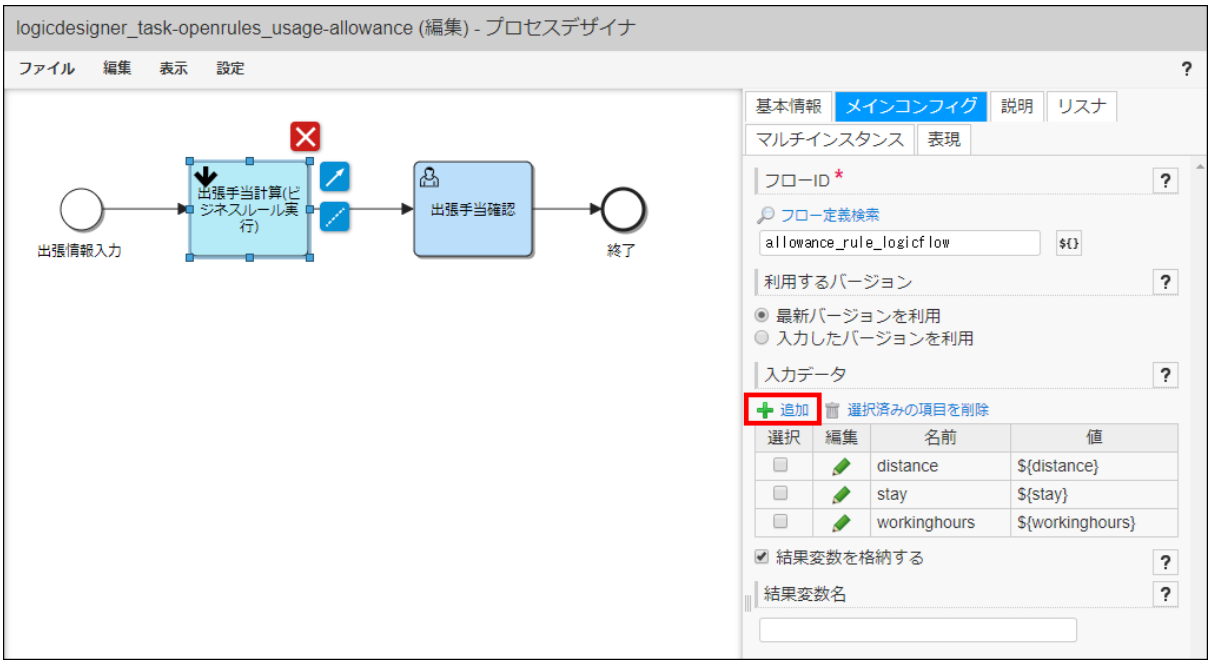
図：「IM-LogicDesignerタスク」-「プロパティ」 -「メインコンフィグ」

8. チュートリアル開始前にインポートしたものを使用します。
「ロジックフロー定義検索」画面でフロー定義ID「allowance_rule_logicflow」を選択し、「決定」をクリックします。



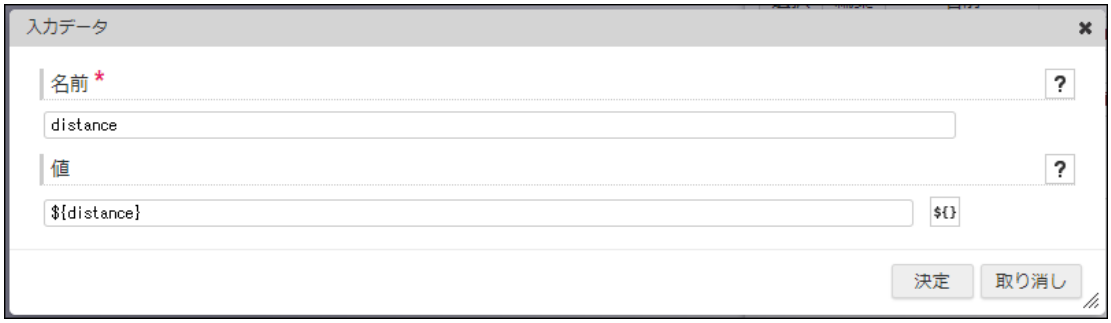
図：「ロジックフロー定義検索」

9. 「IM-FormaDesigner」のアプリケーション画面で入力した値を「IM-LogicDesigner」の入力値に紐づける際に使用するデータを設定します。
「メインコンフィグ」タブから、「入力データ」の「追加」をクリックします。



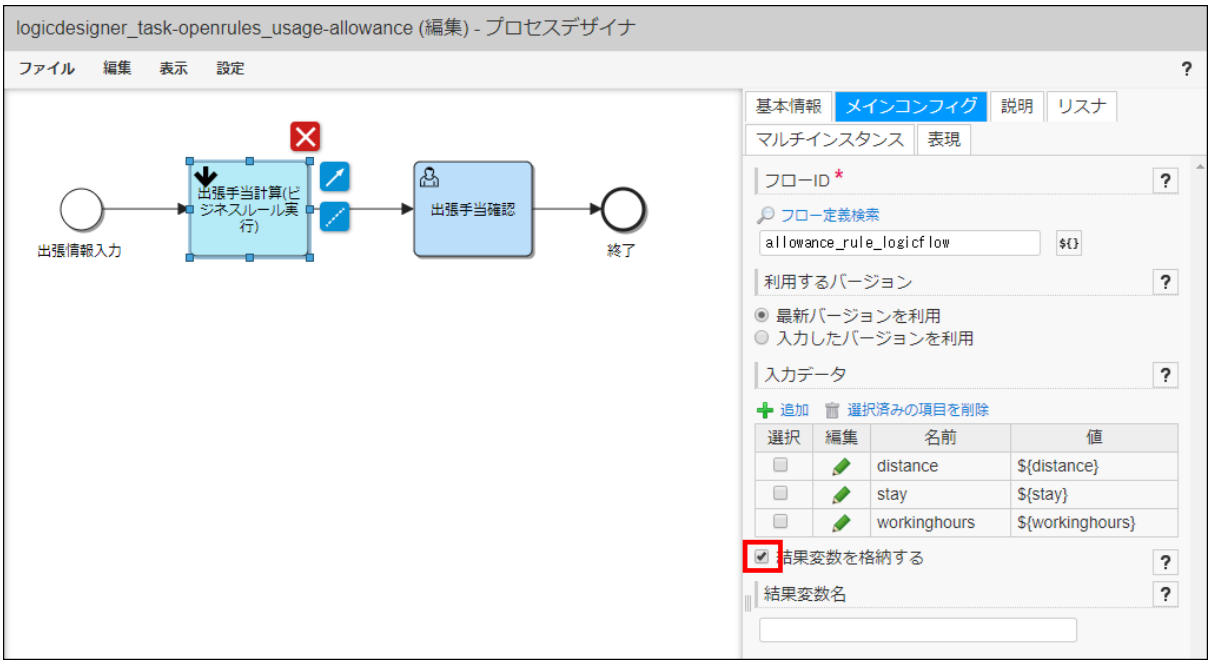
図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

10. 以下のとおりに項目の値を設定してください。
    - 入力データ1
      - 名前：distance
      - 値：${distance}
    - 入力データ2
      - 名前：stay
      - 値：${stay}
    - 入力データ3
      - 名前：workinghours
      - 値：${workinghours}



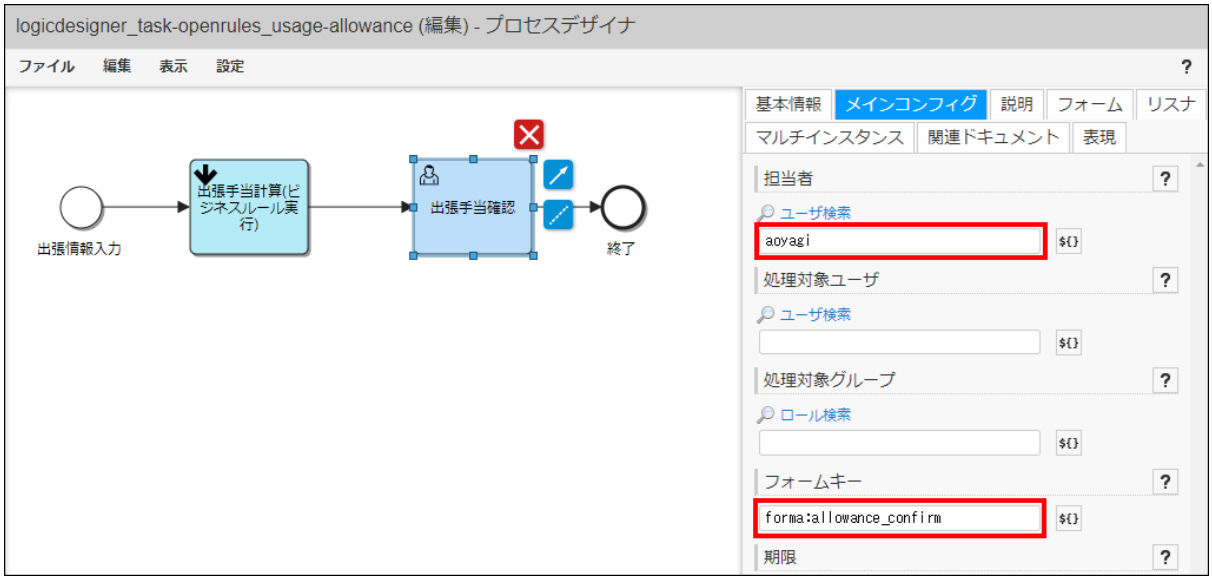
図：「入力データ」

11. 出力値を変数に格納する設定を行います。
    「メインコンフィグ」タブから、「結果変数を格納する」を有効にします。



図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

12. ユーザタスク「出張手当確認」を配置します。
13. ユーザタスク「出張手当確認」に、インポートした「IM-FormaDesigner」のアプリケーションを設定します。
    「メインコンフィグ」タブで、以下のように項目を設定してください。
    - 担当者：aoyagi
    - フォームキー：forma:allowance_confirm



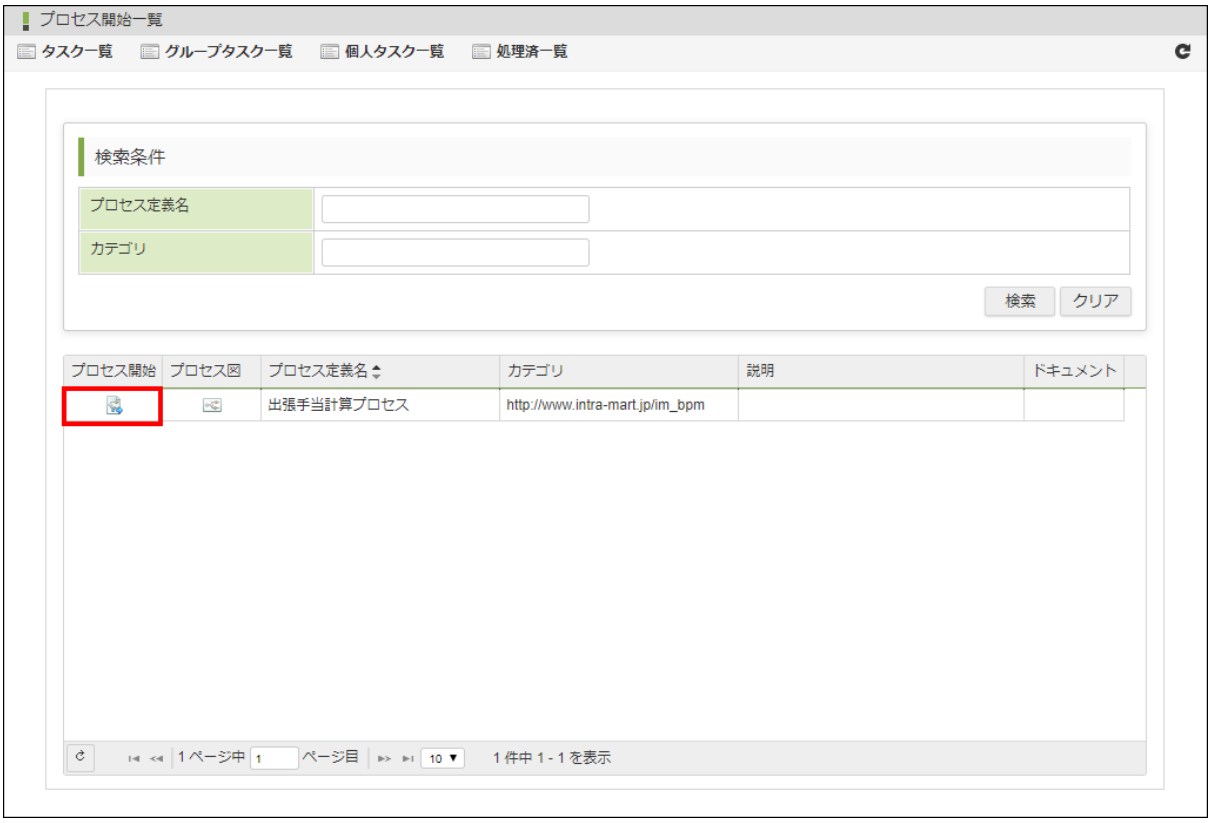
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

14. 終了イベントを設置します。
15. メニューバー左上、「ファイル」 から「名前を付けて保存」を選択し、保存します。

実行結果を確認する

このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行結果の確認を行います。

1. 「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示します。
2. 「出張手当計算プロセス」の「 」をクリックし、プロセスを開始します。



図：「プロセス開始一覧」

3. 開始イベントに設定したアプリケーション画面「allowance_input」に遷移します。
   適当に入力し、「確認」をクリックします。

図：アプリケーション画面「allowance_input」

4. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
5. 「タスク一覧」画面の「個人タスク」を確認します。
   タスク名「出張手当確認」の「  」をクリックします。

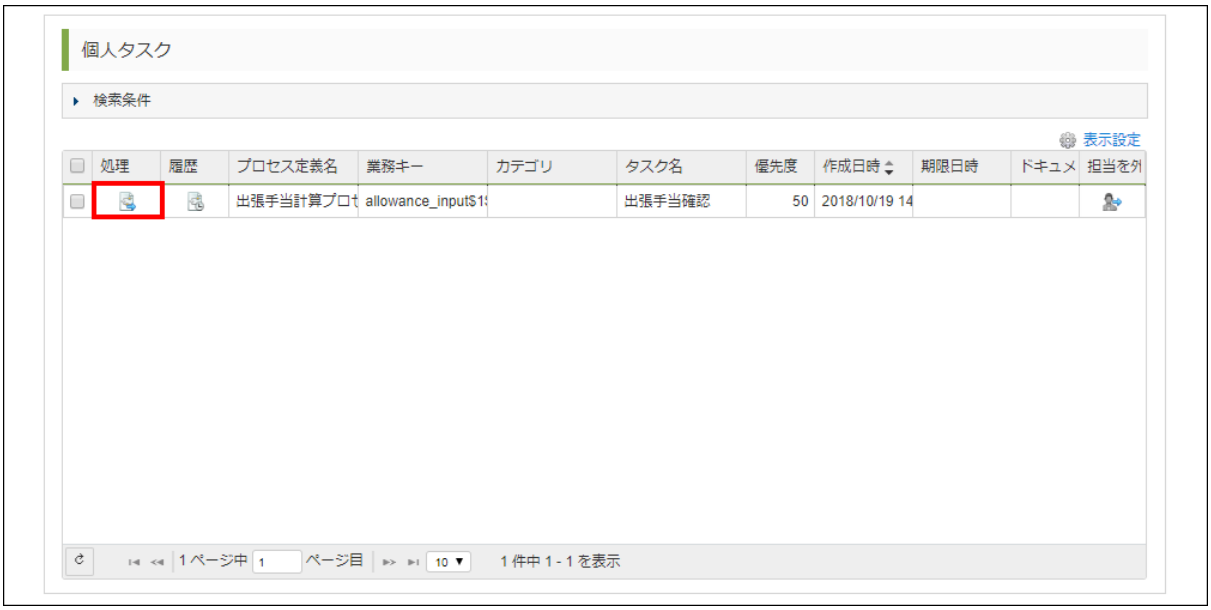


図 ：「タスク一覧」 - 「個人タスク」

6. 結果が表示されます。

図 ：アプリケーション画面「allowance_confirm」

申請タスク

申請タスクを使用してワークフローと連携する

このチュートリアルでは、「申請タスク」を使用してワークフロー「直線ルート[JavaEE開発モデル]」を申請する方法を解説します。
「IM-Workflow」の詳細については、「IM-Workflow 仕様書」を参照してください。
「申請タスク」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「申請タスク」もあわせて参照してください。

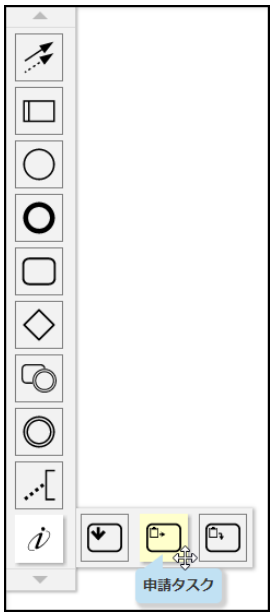
コラム

このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
apply_task_usage.bpmn

このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- 申請タスクを配置する
- 実行するワークフローを設定する
- 実行結果を確認する

申請タスクを配置する



図：パレット - intra-mart - 申請タスク

1. パレットの  にカーソルを合わせ、パレット右側に現れるintra-martタスクの一覧から  をドラッグ＆ドロップして配置します。
2. 配置されたエレメントをクリックして選択することにより、プロパティエリアに選択したエレメントのプロパティが表示されます。

実行するワークフローを設定する

1. プロパティエリアにて「メインコンフィグ」タブを開きます。

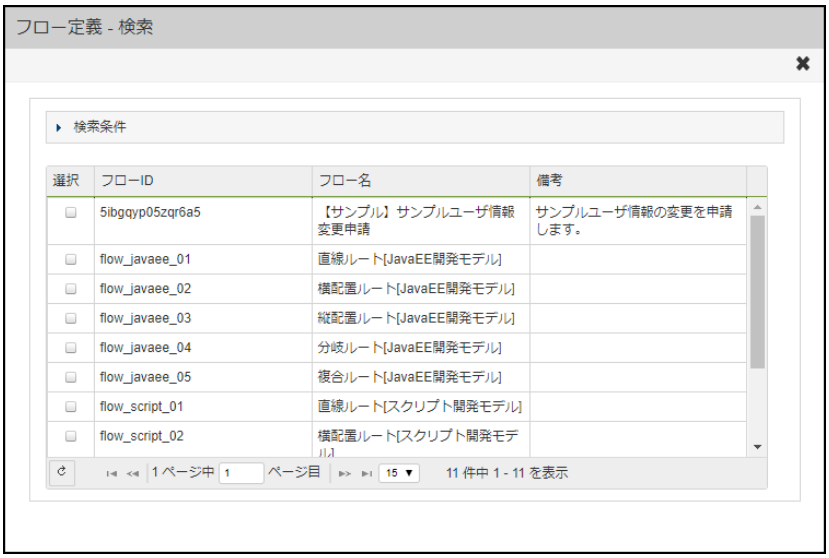
図：申請タスク：プロパティ - メインコンフィグ

2. 「アプリケーション」を設定します。

   このチュートリアルでは、「IM-Workflow」の「直線ルート[JavaEE開発モデル]」を申請するため「IM-Workflow」を設定します。

3. 「フローID」を設定します。

   「フロー定義検索」リンクをクリックして、フロー定義検索ウィンドウから「直線ルート[JavaEE開発モデル]」を検索して選択するとフローID `flow_javaee_01` が自動入力されます。
   ※フロー定義検索ウィンドウから選択せずにフローIDの直接入力や、EL式により動的にフローIDを設定する事もできます。



図：フロー定義検索ウィンドウ

4. 「案件名」を設定します。

   申請タスクチュートリアル を設定します。
   ※EL式で動的に案件名を設定する事も可能です。

5. 「申請実行者コード」を設定します。

   「ユーザ検索」リンクをクリックして、申請実行者コードダイアログから「青柳辰巳」を検索して選択すると、ユーザコード aoyagi が自動入力されます。
   ※申請実行者コードダイアログから選択せずにユーザコードの直接入力や、EL式により動的にユーザコードを設定する事もできます。

図：申請実行者コードダイアログ

6. 「申請権限者コード」を設定します。

   「ユーザ検索」リンクをクリックして、申請権限者コードダイアログから「青柳辰巳」を検索して選択すると、ユーザコード aoyagi が自動入力されます。
   ※申請者コードダイアログから選択せずにユーザコードの直接入力や、EL式により動的にユーザコードを設定する事もできます。

図：申請権限者コードダイアログ

7. 「入力データ」を設定します。

   利用する「直線ルート[JavaEE開発モデル]」の申請に必要な以下の入力データを登録します。

名前	値
item_name	サンプル商品
item_amount	2
item_price	10000
item_comment	コメント

8. 「案件の処理結果を格納する変数名」を設定します。

   workflowResult を設定します。

9. 「案件のユーザデータを格納する」を設定します。

   チェックボックスを有効にします。

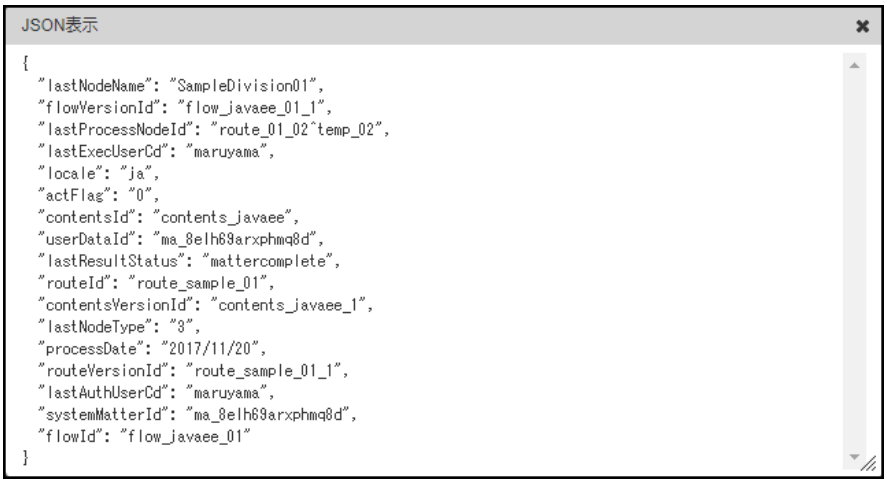
10. 「案件のユーザデータを格納する変数名」を設定します。

    workflowData を設定します。

実行結果を確認する

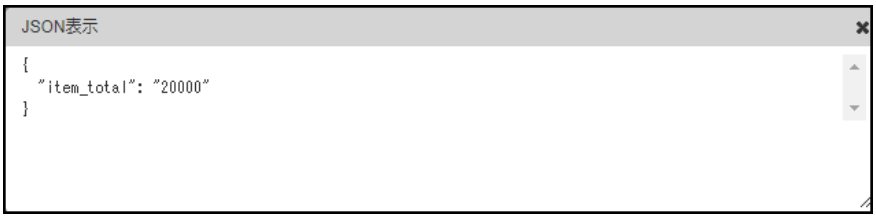
このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

1. プロセスを実行すると、ワークフローに案件名「申請タスクチュートリアル」が申請されるので案件を完了させます。
2. 「プロセス一覧」画面にて完了したプロセスを検索し、実行したプロセスの「プロセス詳細」画面へ遷移します。
3. 「変数一覧」画面にて、ワークフローの処理結果が格納されている変数 workflowResult の内容を確認します。



図：「変数一覧」画面 - JSON表示

4. 「変数一覧」画面にて、ワークフローのユーザデータが格納されている変数 workflowData の内容を確認します。



図：「変数一覧」画面 - JSON表示

起票タスク

起票タスクを使用してワークフローと連携する

このチュートリアルでは、「起票タスク」を使用してワークフロー「直線ルート[JavaEE開発モデル]」を起票する方法を解説します。
「IM-Workflow」の詳細については、「IM-Workflow 仕様書」を参照してください。
「起票タスク」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「起票タスク」もあわせて参照してください。

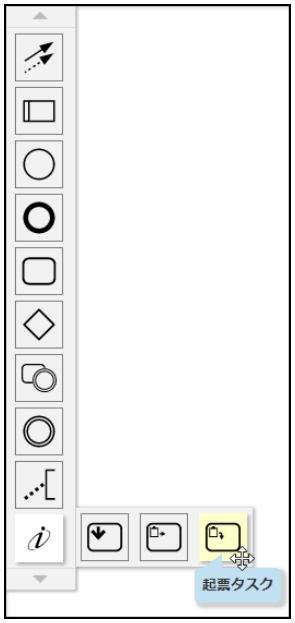
コラム

このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
draft_task_usage.bpmn

このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- 起票タスクを配置する
- 実行するワークフローを設定する
- 実行結果を確認する

起票タスクを配置する



図：パレット - intra-mart - 起票タスク

1. パレットの  にカーソルを合わせ、パレット右側に現れるintra-martタスクの一覧から  をドラッグ＆ドロップして配置します。
2. 配置されたエレメントをクリックして選択することにより、プロパティエリアに選択したエレメントのプロパティが表示されます。

実行するワークフローを設定する

1. プロパティエリアにて「メインコンフィグ」タブを開きます。

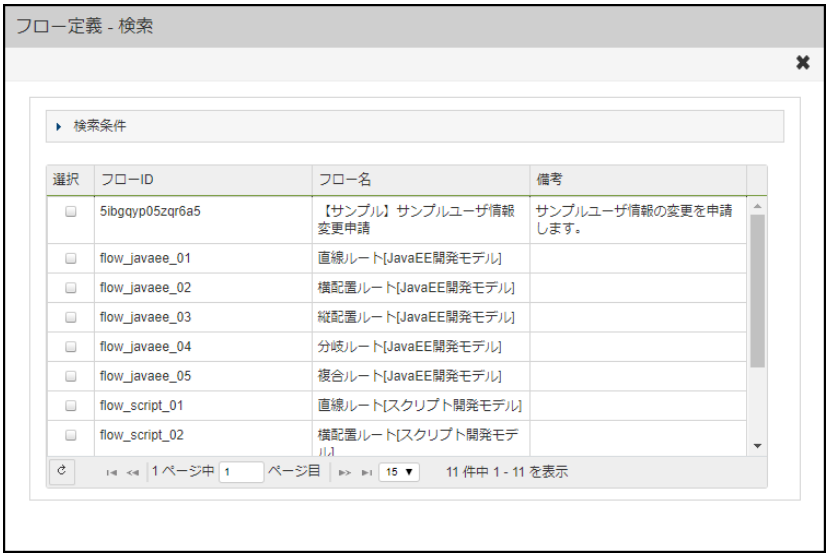
図：起票タスク：プロパティ - メインコンフィグ

2. 「アプリケーション」を設定します。

   このチュートリアルでは、「IM-Workflow」の「直線ルート[JavaEE開発モデル]」を起票するため「IM-Workflow」を設定します。

3. 「フローID」を設定します。

   「フロー定義検索」リンクをクリックして、フロー定義検索ウィンドウから「直線ルート[JavaEE開発モデル]」を検索して選択するとフローID `flow_javaee_01` が自動入力されます。
   ※フロー定義検索ウィンドウから選択せずにフローIDの直接入力や、EL式により動的にフローIDを設定する事もできます。



図：フロー定義検索ウィンドウ

4. 「案件名」を設定します。

   `起票タスクチュートリアル` を設定します。
   ※EL式により動的に案件名を設定する事もできます。

5. 「起票者コード」を設定します。

   「ユーザ検索」リンクをクリックして、起票者コードダイアログから「青柳辰巳」を検索して選択すると、ユーザコード `aoyagi` が自動入力されます。
   ※起票者コードダイアログから選択せずにユーザコードの直接入力や、EL式により動的にユーザコードを設定する事もできます。

図：起票者コードダイアログ

6. 「案件の処理結果を格納する変数名」を設定します。

   `workflowResult` を設定します。

7. 「案件のユーザデータを格納する」を設定します。

チェックボックスを有効にします。

8. 「案件のユーザデータを格納する変数名」を設定します。

   workflowData を設定します。

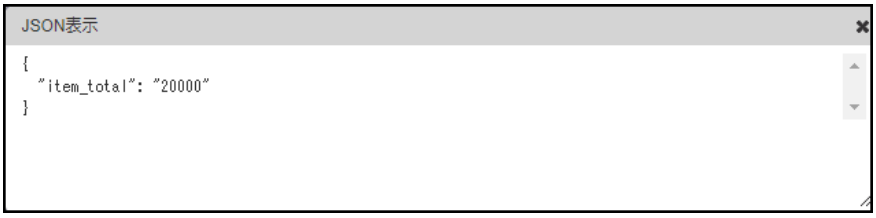
実行結果を確認する

このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

1. プロセスを実行すると、ワークフローに案件名「起票タスクチュートリアル」が起票されるので案件を完了させます。
2. 「プロセス一覧」画面にて完了したプロセスを検索し、実行したプロセスの「プロセス詳細」画面へ遷移します。
3. 「変数一覧」画面にて、ワークフローの処理結果が格納されている変数 workflowResult の内容を確認します。

図：「変数一覧」画面 - JSON表示

4. 「変数一覧」画面にて、ワークフローのユーザデータが格納されている変数 workflowData の内容を確認します。



図：「変数一覧」画面 - JSON表示

起票タスクを使用してワークフローの申請ノード処理対象者を動的に設定する

このチュートリアルでは、「起票タスク」を使用してワークフロー「直線ルート[JavaEE開発モデル]」を起票する際に「申請ノード処理対象者」を動的に設定する方法を解説します。
IM-Workflowの詳細については、「IM-Workflow 仕様書」を参照してください。
ワークフローを起票する方法については、前章の「*起票タスクを使用してワークフローと連携する*」を参照してください。

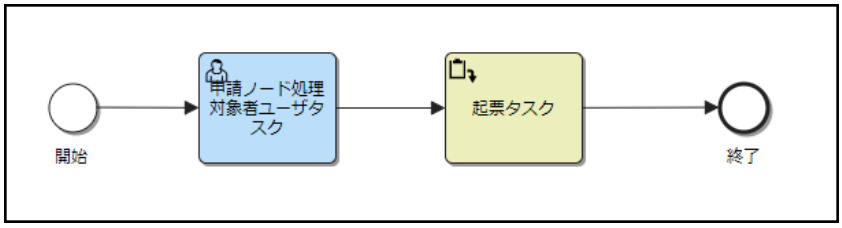
> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **draft_task_apply_user_plugin.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- ワークフローの申請ノード処理対象者を動的に設定するプロセス定義を作成する
- 実行結果を確認する

ワークフローの申請ノード処理対象者を動的に設定するプロセス定義を作成する

以下のユーザを「申請ノード処理対象者」に設定します。

- プロセスを開始したユーザ
- 起票タスクの手前のユーザタスクを処理したユーザ



図：完成イメージ

1. 開始イベントの「イニシエータ」に starter を設定します。

図：開始イベント：プロパティ - メインコンフィグ

2. ユーザタスクの「ID」に draft_task_before_user_task を設定します。
3. ユーザタスクの「名前」に 申請ノード処理対象者ユーザタスク を設定します。

図：ユーザタスク：プロパティ - 基本情報

4. ユーザタスクの「処理対象グループ」に im_bpm_user を設定します。

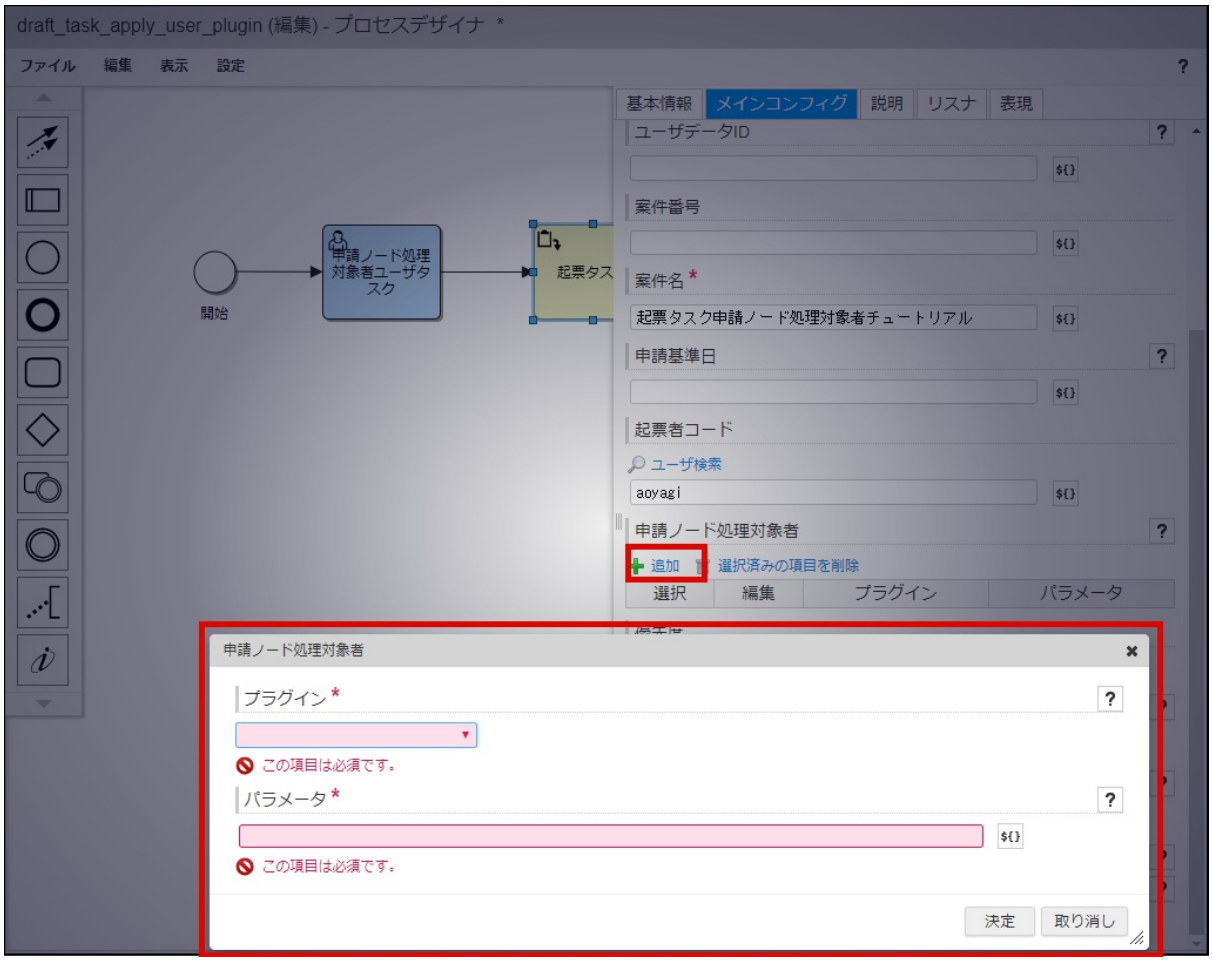
図：ユーザタスク：プロパティ - メインコンフィグ

5. 起票タスクの各プロパティを設定します。
   - アプリケーション: IM-Workflow
   - フローID: flow_javaee_01

- 案件名: 起票タスク申請ノード処理対象者チュートリアル
- 起票者コード: aoyagi

6. 起票タスクの「申請ノード処理対象者」を設定します。

    1. 「申請ノード処理対象者」の「追加」リンクをクリックします。



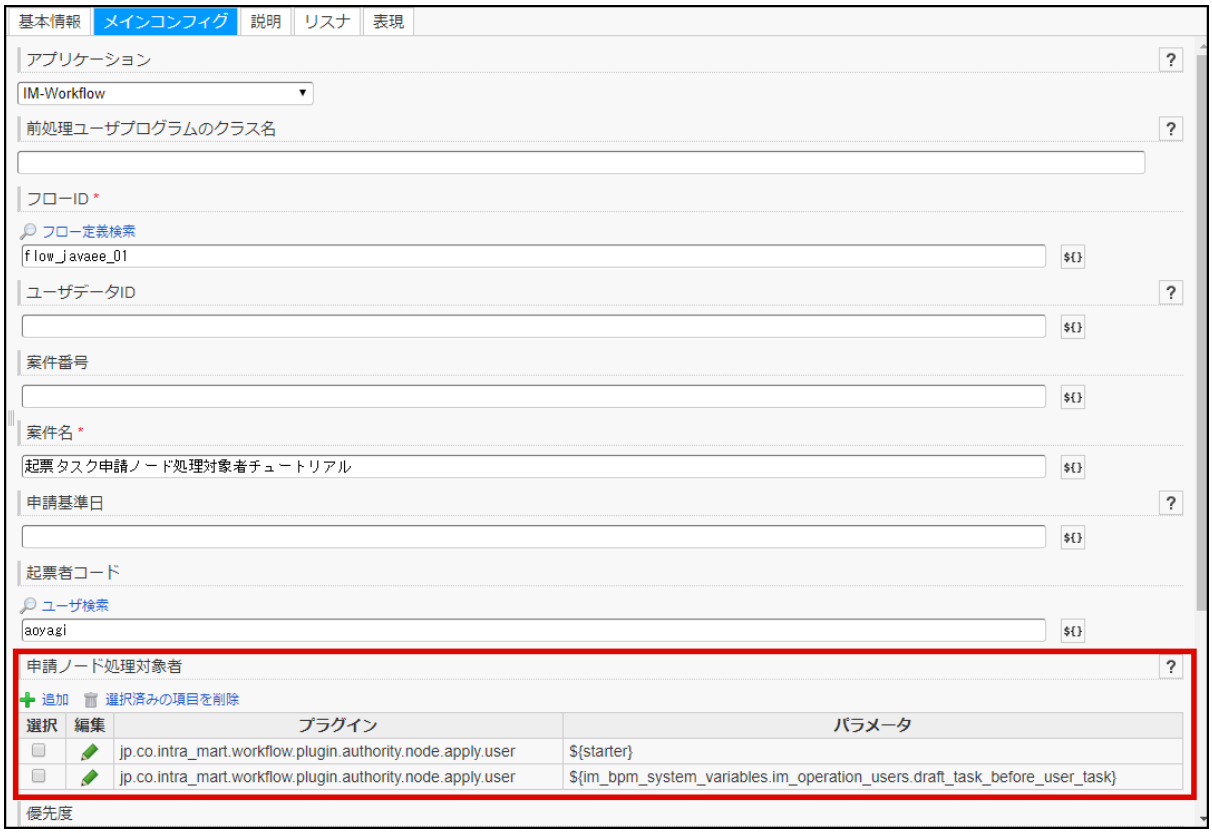
図：申請ノード処理対象者ダイアログ

    2. 「プラグイン」と「パラメータ」を設定します。

図：申請ノード処理対象者ダイアログ

以下を設定します。

- プラグイン: ユーザ , パラメータ: ${starter}
- プラグイン: ユーザ , パラメータ: ${im_bpm_system_variables.im_operation_users.draft_task_before_user_task}



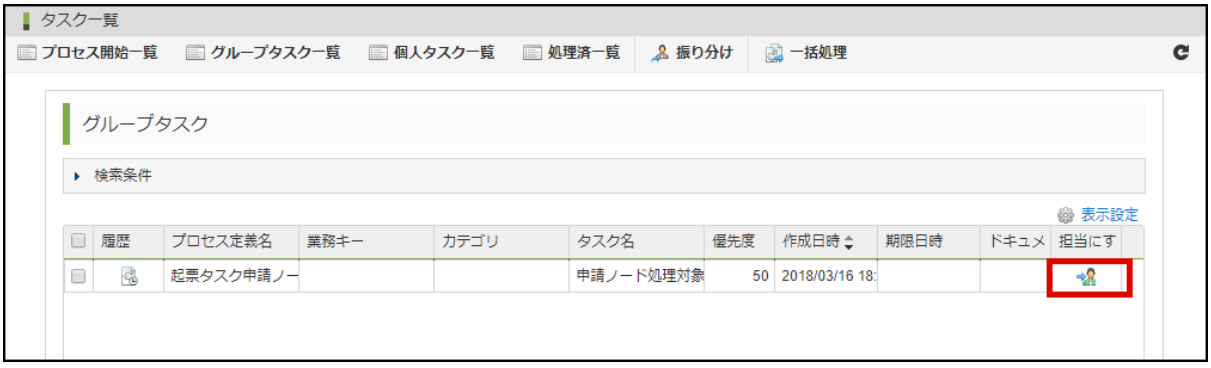
図：起票タスク：プロパティ - メインコンフィグ

> **コラム**
>
> 申請ノード処理対象者ダイアログのプラグイン選択プルダウンリストは、サーバ通信により取得しています。
> 通信の状況によっては、プルダウンリストが表示されない場合があります。

実行結果を確認する

このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

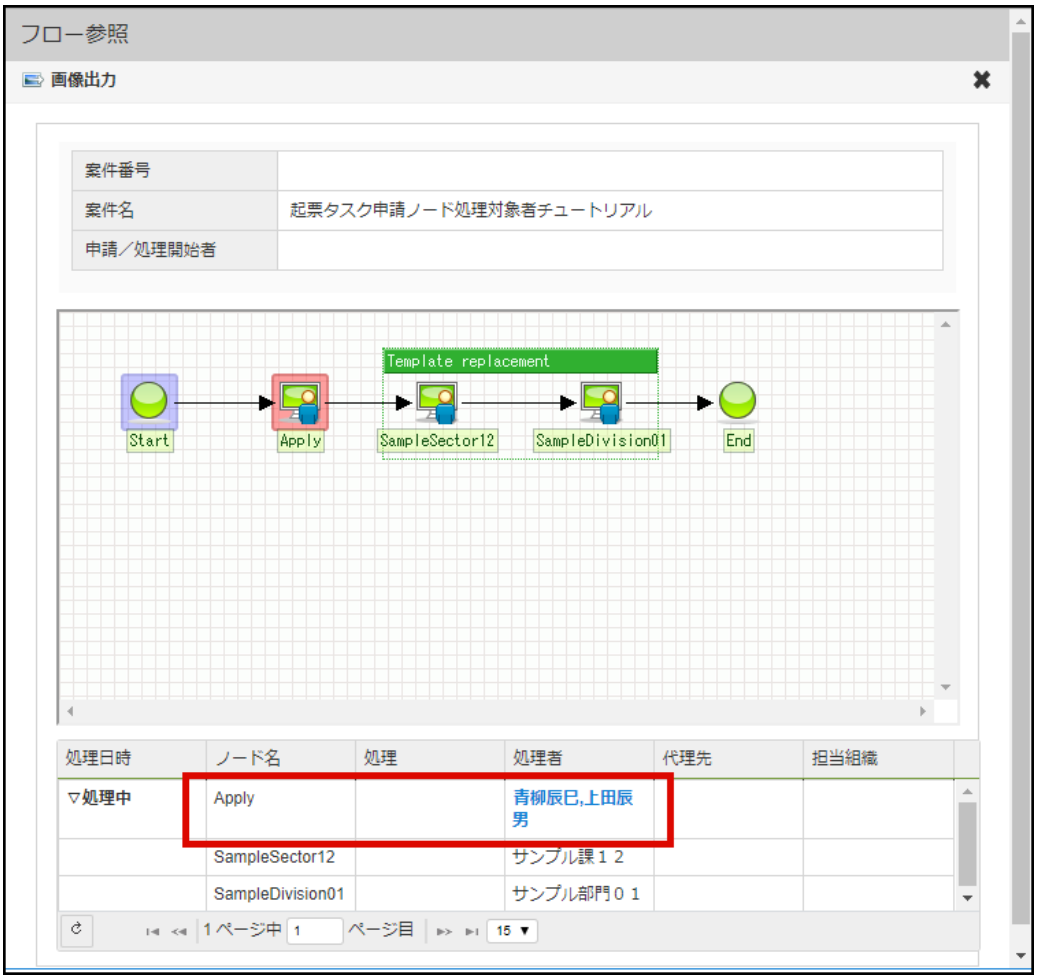
1. プロセスを開始すると、「申請ノード処理対象者ユーザタスク」がグループタスクに作成されます。
2. 「申請ノード処理対象者ユーザタスク」を担当にし、続けて処理を行います。
   プロセスを開始したユーザと異なるユーザで行います。「IM-BPM ユーザ」のロールが付与されたユーザが対象です。



図：担当にする

図：処理する

3. ワークフローの未処理一覧からフローを確認します。



図：フロー参照

IM-FormaDesigner for Accel Platform をユーザ入力フォームとして利用する

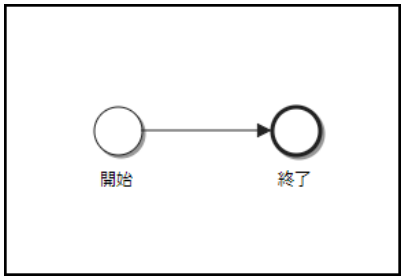
開始イベントのフォームにFormaアプリケーションを利用する

このチュートリアルでは、開始イベントのフォームにFormaアプリケーション「【サンプル】備品管理(v8)」を設定し、プロセスの開始時にFormaアプリケーションで登録する画面を表示する方法を解説します。
IM-FormaDesigner for Accel Platform の詳細については、「IM-FormaDesigner 仕様書」を参照してください。

> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **start_event_with_forma.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- エレメントを配置する
- 開始イベントのプロパティを設定する
- 実行結果を確認する

エレメントを配置する



図：配置イメージ

1. 開始イベント、終了イベントを配置してシーケンスフローで接続します。

開始イベントのプロパティを設定する

図：プロパティ入力イメージ

1. 開始イベントのプロパティで「メインコンフィグ」タブを開きます。
2. 「フォームキー」に `forma:sample_app_equipment` を入力します。

> **コラム**
>
> 「フォームキー」に `forma:` と、Formaアプリケーションの「 `アプリケーションID` 」を入力すると、プロセスの開始時にFormaアプリケーションの画面を開くことが可能です。

> **コラム**
>
> 開始イベントから実行されるFormaアプリケーションに対して、IM-BPMの機能で初期値を渡すことはできません。

実行結果を確認する

このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

1. 「プロセス一覧」画面にて完了したプロセスを検索し、実行したプロセスの「プロセス詳細」画面へ遷移します。
2. 「変数一覧」画面にて、変数の内容を確認します。
   Formaアプリケーションのフォームに定義された「フィールド識別ID」と同名の変数が作成され、フォームで入力した値が格納されていることが確認できます。

スコープ	変数名	型	値
Process	additional_description	string	今回から発注先が変更になりましたので、ご留意願います。
Process	color	string	白
Process	company	string	intra-mart文具
Process	equipment_name	string	コピー用紙
Process	locale	string	ja
Process	maker	string	intra-mart製紙
Process	minimum_quantity	long	10
Process	price	long	1450
Process	purchase_unit	string	2
Process	size	string	A4

図：「変数一覧」画面

コラム

- プロセスのデプロイ方法については「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のデプロイ」を参照してください。
- デプロイしたプロセスの実行方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスの開始」を参照してください。
- 「プロセス詳細」画面への遷移方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスを確認する」を参照してください。
- 「変数一覧」画面の操作方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスの変数を確認する」を参照してください。

ユーザタスクのフォームにFormaアプリケーションを利用する

このチュートリアルでは、ユーザタスクのフォームにFormaアプリケーション「【サンプル】備品管理(v8)」を設定し、ユーザタスクの処理時にFormaアプリケーションで登録する画面を表示する方法を解説します。
IM-FormaDesigner for Accel Platform の詳細については、「IM-FormaDesigner 仕様書」を参照してください。

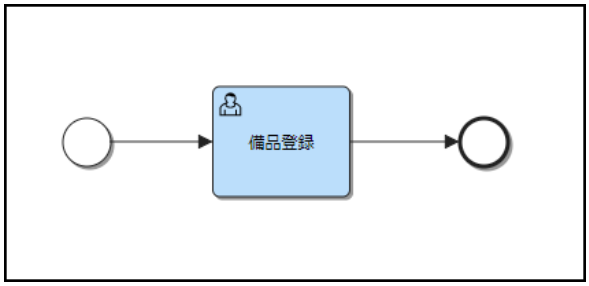
コラム

このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
user_task_with_forma.bpmn

このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- エレメントを配置する
- ユーザタスクのプロパティを設定する
- Formaアプリケーションに受け渡す変数を定義する
- 実行結果を確認する

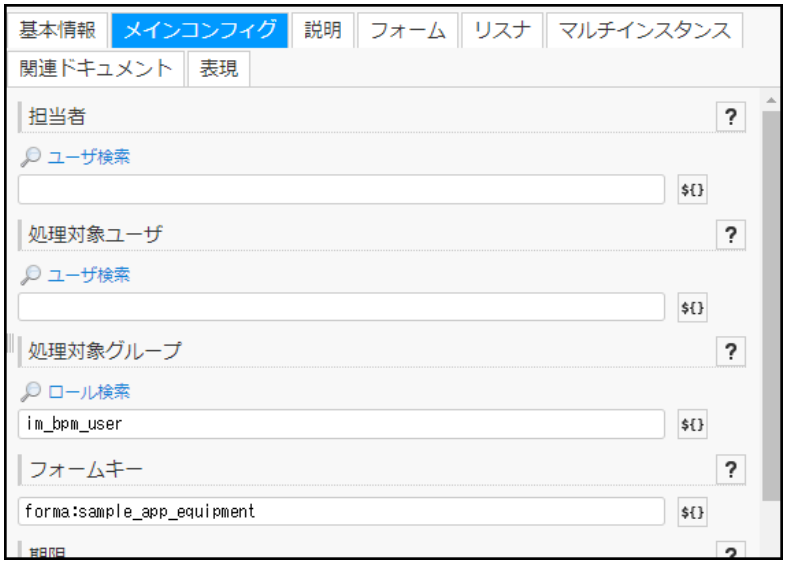
エレメントを配置する



図：配置イメージ

1. 開始イベント、ユーザタスク、終了イベントを配置して、順にシーケンスフローで接続します。

ユーザタスクのプロパティを設定する



図：プロパティ入力イメージ

1. ユーザタスクのプロパティで「メインコンフィグ」タブを開きます。

2. 「処理対象グループ」に im_bpm_user を入力します。

3. 「フォームキー」に forma:sample_app_equipment を入力します。

> **コラム**
>
> 「フォームキー」に forma: 、続けてFormaアプリケーションの「 アプリケーションID 」を入力すると、プロセスの開始時にFormaアプリケーションの画面を開くことが可能です。

Formaアプリケーションに受け渡す変数を定義する

ユーザタスク実行時に参照できるスコープに存在する変数のうち、Formaアプリケーションの「フィールド識別ID」に一致する「名前」の変数が、Formaアプリケーションの初期値として受け渡されます。
このチュートリアルでは、プロセスグローバルの変数として定義します。

図：プロセス：プロパティ - データオブジェクト

1. プロセスグローバルのプロパティで、「データオブジェクト」タブを開きます。

2. 「データプロパティ」に equipment_name="コピー用紙"、maker="intra-mart製紙" を定義します。

> **コラム**
>
> プロセスグローバルのデータオブジェクト設定方法については「*プロセスにデータプロパティを設定する*」を参照してください。

実行結果を確認する

このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

1. 「プロセス開始一覧」画面にて当該プロセスを開始します。

2. 「タスク一覧」画面にて当該ユーザタスクを引き受け、「処理」をクリックするとFormaアプリケーションが表示されます。
「備品名」と「メーカー」の項目に、受け渡した変数値が初期表示されていることが確認できます。

図：「備品登録」画面

3. 「備品登録」画面にて、各入力フォームに値を設定して登録します。

4. 「プロセス一覧」画面にて完了したプロセスを検索し、実行したプロセスの「プロセス詳細」画面へ遷移します。

5. 「変数一覧」画面にて、変数の内容を確認します。
Formaアプリケーションのフォームに定義された「フィールド識別ID」と同名の変数が作成され、フォームで入力した値が格納されていることが確認できます。

スコープ	変数名	型	値
Process	additional_description	string	今回から発注先が変更になりましたので、ご留意願います。
Process	color	string	白
Process	company	string	intra-mart文具
Process	equipment_name	string	コピー用紙
Process	locale	string	ja
Process	maker	string	intra-mart製紙
Process	minimum_quantity	long	10
Process	price	long	1450
Process	purchase_unit	string	2
Process	size	string	A4

図：「変数一覧」画面

コラム
- プロセスのデプロイ方法については「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のデプロイ」を参照してください。
- デプロイしたプロセスの実行方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスの開始」を参照してください。
- 「タスク一覧」画面の操作方法については「IM-BPM ユーザ操作ガイド」-「タスクについて」を参照してください。
- 「プロセス詳細」画面への遷移方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスを確認する」を参照してください。
- 「変数一覧」画面の操作方法については「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスの変数を確認する」を参照してください。

発展編

各種エレメントと他アプリケーションを組み合わせて、発展的な機能を実現する手順を説明します。
発展編の各チュートリアルは「*事前準備*」が完了していることを前提としています。

- ワークフローとの連携

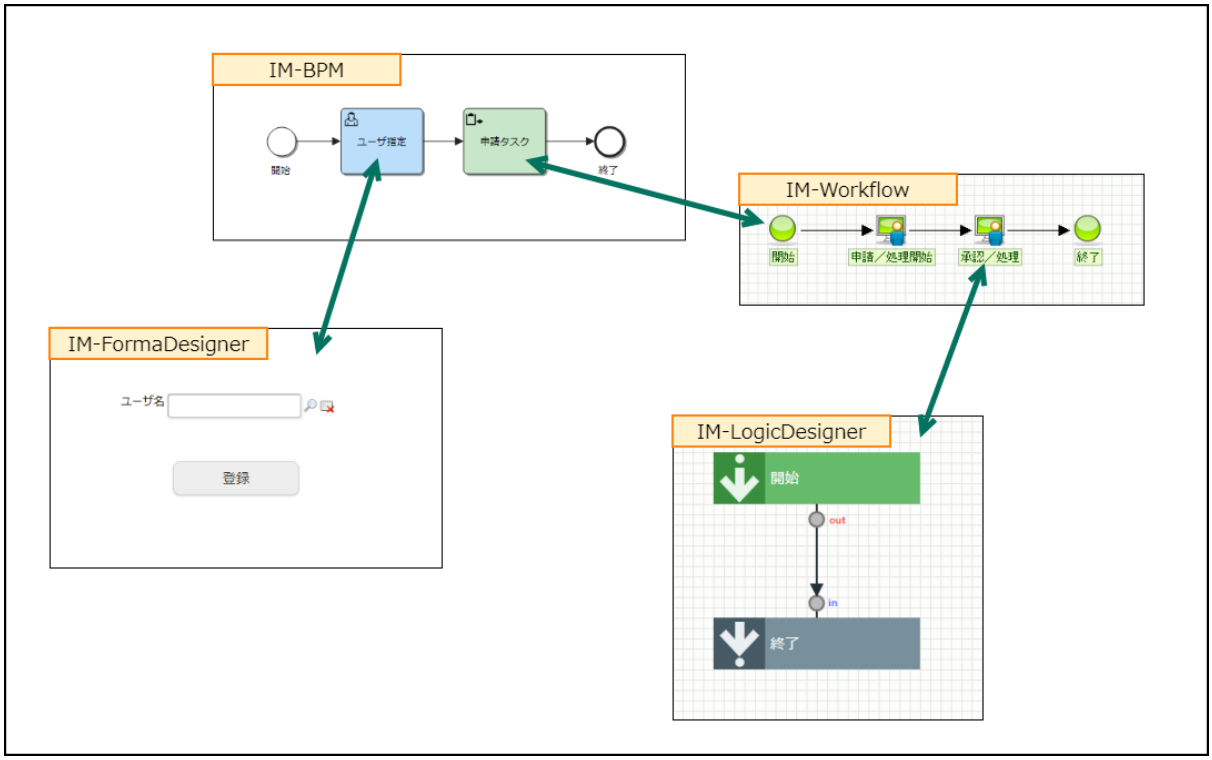
ワークフローとの連携

承認ノードの処理対象者をロジックフローで指定する

このチュートリアルでは、申請タスクで連携したワークフローに対して、承認ノードの処理対象者を動的に指定する方法を解説します。
処理対象者の動的な指定については、処理対象者プラグインをロジックフローを利用して記述することで行います。
指定する対象のユーザについては、ユーザタスク処理時に画面から入力したユーザとユーザタスクを処理したユーザ自身の2人とします。

チュートリアルで作成するものは以下の4つです。

- プロセス定義 （IM-BPM）
  - ユーザタスクと申請タスクを配置し、ワークフローと連携します。
  - 画面から入力された処理対象者情報をプロセスの変数情報に保持します。
- Formaアプリケーション （IM-FormaDesigner）
  - ユーザタスクの処理画面として使用します。
  - 処理対象者情報を画面から入力させます。
- ロジックフロー （IM-LogicDesigner）
  - ワークフローから処理対象者プラグインとして利用します。
  - プロセスの変数情報から処理対象者情報を取得して返却します。
- ワークフローのルート定義・フロー定義 （IM-Workflow）
  - 申請タスクによって申請されます。
  - 承認ノード到達時、処理対象者プラグインとしてロジックフローと連携します。



図：概要

> **注意**
>
> このチュートリアルで利用している、プロセス/ワークフロー/ロジックフロー間の連携機能は、IM-BPM for Accel Platform 2018 Summer(Tiffany) 以降のバージョンで動作します。

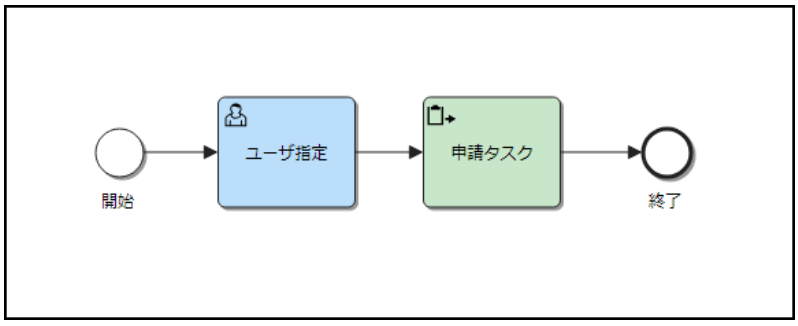
> **コラム**
>
> このチュートリアルで作成する各種定義のサンプルを以下のリンクからダウンロードできます。
>
> - プロセス定義
>   **approve_target_logicdesigner.bpmn**
> - IM-FormaDesignerアプリケーション情報
>   **app_select_target.zip**
> - IM-LogicDesigner ロジックフロー定義
>   **im_logicdesigner-data-workflow_target_plugin.zip**
> - IM-Workflow ロジックフロー管理テーブルデータ
>   **imw_m_logic_flow.csv**
> - IM-Workflow ルート定義およびフロー定義
>   **route_flow_target_ld.xml**
>
> これらのサンプルは、各アプリケーションの機能でインポートまたはアップロードして利用可能です。
> 詳細な手順については、以下のリンク先を参照してください。
>
> - 「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」
> - 「IM-FormaDesigner 作成者操作ガイド」-「インポート・エクスポートを利用した IM-FormaDesigner のアプリケーションやデータソース定義の移行」
> - 「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」
> - 「TableMaintenance 管理者操作ガイド」-「テーブル・インポート」
> - 「IM-Workflow 管理者操作ガイド」-「インポート/エクスポートを行う」
>
> インポートまたはアップロードは、上記の順番で実行してください。
> ロジックフロー定義のインポートよりも前に、ロジックフロー管理テーブルデータやルート定義のインポートを行った場合、正しくデータが連携しません。

- プロセス定義を作成する
  - エレメントを配置する
  - ユーザタスクのプロパティを設定する
  - 申請タスクのプロパティを設定する
- Formaアプリケーションを作成する
  - アプリケーションの基本情報を入力する
  - フォームを編集する
  - テーブルを登録する
  - 権限を設定する
- ロジックフローを作成する
  - 入出力設定を行う
  - 定数設定を行う
  - 変数設定を行う
  - エレメントのマッピングを行う
  - 新規保存を行う
- ワークフローのルート定義・フロー定義を作成する
  - ロジックフローを管理する
  - ルート定義を作成する
  - フロー定義を作成する
- 実行結果を確認する
  - プロセスを開始し、ユーザタスクを処理する
  - ワークフローの処理対象者を確認する

プロセス定義を作成する

プロセスエディタを利用して、ユーザタスクと申請タスクを持ちワークフローと連携するプロセス定義を作成します。
ユーザタスクの処理画面として、Formaアプリケーションを使用します。

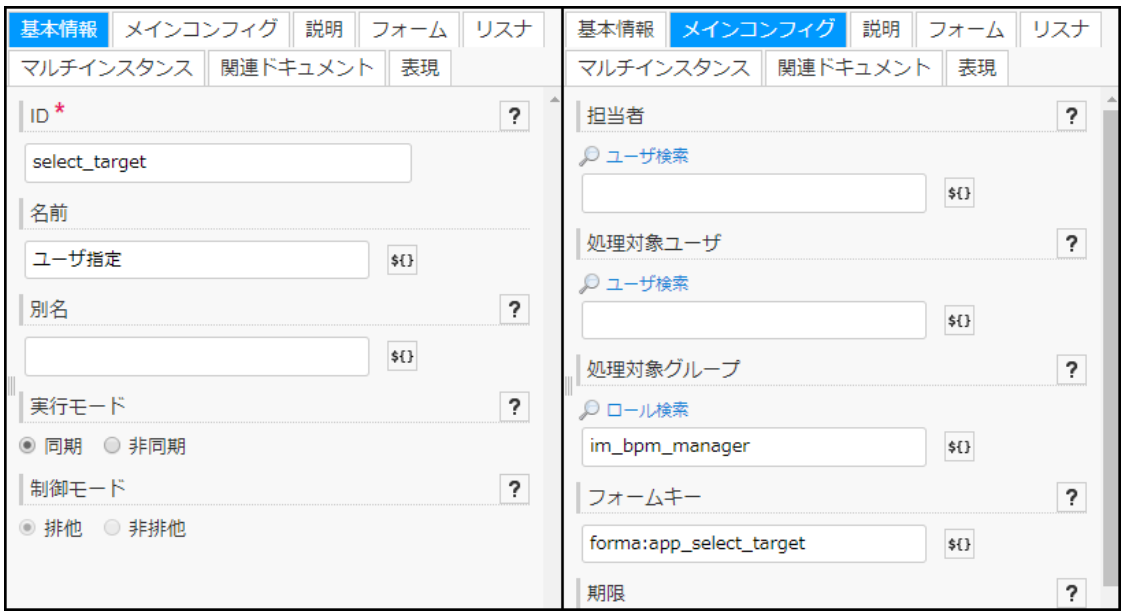
エレメントを配置する



図：完成イメージ

1. 開始イベント、ユーザタスク、申請タスク、終了イベントを配置してシーケンスフローで接続します。

ユーザタスクのプロパティを設定する



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」/「メインコンフィグ」

1. ユーザタスクのプロパティエリアにて「基本情報」タブを表示します。
2. 「ID」に select_target を入力します。
   このチュートリアルでは、ここで入力したIDに紐づく変数から、このユーザタスクを処理したユーザを取得します。
3. 「メインコンフィグ」タブを表示します。
4. 「処理対象グループ」に im_bpm_manager を入力します。
5. 「フォームキー」に forma:app_select_target を入力します。
   このチュートリアルでは、後ほど app_select_target というアプリケーションIDを持つFormaアプリケーションを作成します。

申請タスクのプロパティを設定する

図：「申請タスク」 - 「プロパティ」 - 「メインコンフィグ」

1. 申請タスクのプロパティエリアにて「メインコンフィグ」タブを表示します。
2. 「アプリケーション」にて「IM-Workflow」を選択します。
3. 「フローID」に flow_target_id を入力します。
   このチュートリアルでは、後ほど flow_target_id というIDを持つフロー定義を作成します。
4. 「案件名」に 承認ノード処理対象者チュートリアル を入力します。
5. 「申請実行者コード」に aoyagi を入力します。
6. 「申請権限者コード」に aoyagi を入力します。

Formaアプリケーションを作成する

画面から処理対象者を指定するため、Formaアプリケーションを作成します。
「サイトマップ」→「Forma管理画面」→「アプリ一覧」から「アプリケーション一覧」画面を表示し、「登録」をクリックしてアプリケーションの作成を開始します。

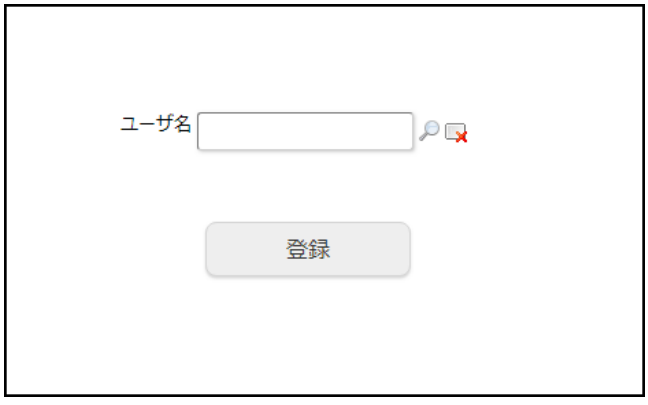
アプリケーションの基本情報を入力する

図：「アプリケーション登録」

1. 「アプリケーションID」に app_select_target を入力します。
   プロセス定義の作成時、ユーザタスクのプロパティ「フォームキー」に指定したものです。
2. 「アプリケーション種別」にて「標準」を選択します。
3. 「有効日付（開始）」に 2018/01/01 を入力します。
4. 「アプリケーション名」に 【チュートリアル】処理対象者選択 を入力します。
5. 「登録」をクリックして先に進みます。

> **コラム**
>
> Formaアプリケーションの基本情報については、「IM-FormaDesigner 作成者操作ガイド」-「IM-FormaDesigner のアプリケーションの基本情報を設定する」を参照してください。

フォームを編集する



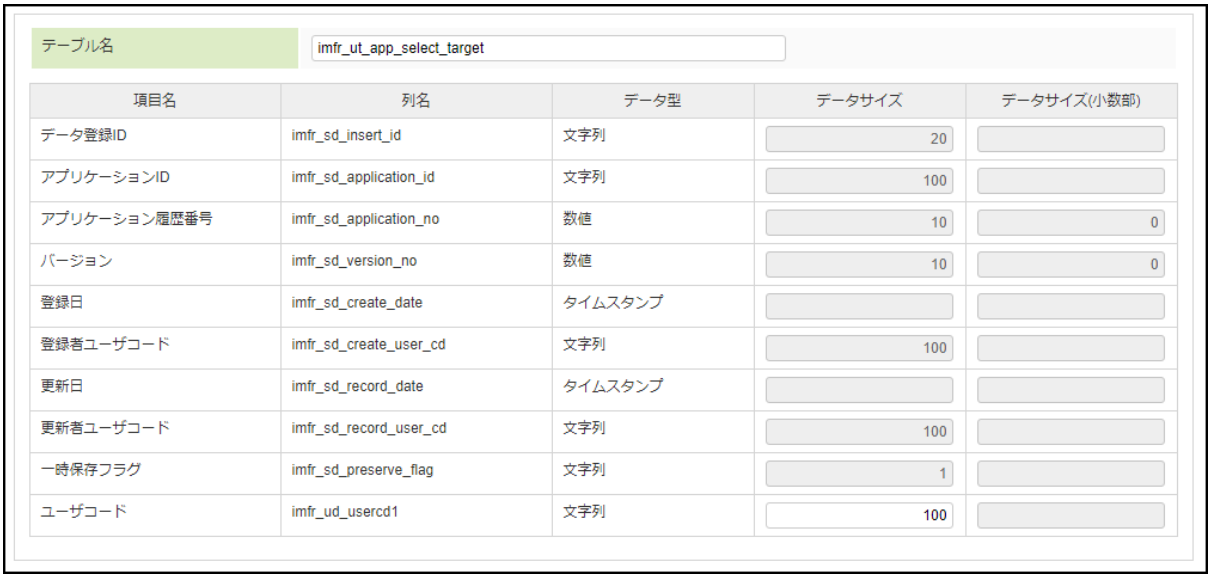
図：完成イメージ（フォーム）

1. 「ツールキット」を開き、「共通マスタアイテム」 -「ユーザ選択」アイテムをドラッグし、フォームに配置します。
   アイテムのプロパティは特に変更する必要はありませんが、「フィールド識別ID」が userCd1 であることを確認してください。
2. 「ツールキット」 -「ボタンアイテム」 -「ボタン（登録）」アイテムをドラッグし、フォームに配置します。
3. 「更新」をクリックしてフォームを更新します。

> **コラム**
>
> フォーム・デザイナの詳細については、「IM-FormaDesigner 作成者操作ガイド」-「「フォーム・デザイナ」画面の各部の名称と機能」を参照してください。

テーブルを登録する

1. 以下のいずれかの方法で「フォーム設定」画面を表示します。
   - 「フォーム編集」画面から「戻る」リンクをクリックして「フォーム一覧」画面に移動し、もう一度 「戻る」リンクをクリックします。
   - 「サイトマップ」→「Forma管理画面」→「アプリ一覧」をクリックし、アプリケーションID「app_select_target」の編集アイコンをクリックします。
2. 「テーブル設定」タブをクリックします。
3. 「登録」をクリックします。



図：「テーブル設定」

4. 「ユーザコード」項目のデータサイズに `100` を入力します。
5. 「登録」をクリックして、テーブルを登録します。

> **コラム**
>
> テーブルの詳細については、「IM-FormaDesigner 作成者操作ガイド」-「テーブルを設定する」を参照してください。

権限を設定する

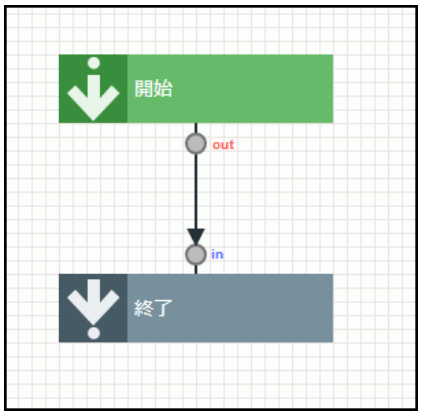
1. 「サイトマップ」→「Forma管理画面」→「アプリ一覧」から「アプリケーション一覧」画面を表示します。
2. アプリケーションID「app_select_target」の編集アイコンをクリックします。
3. 「権限設定」タブをクリックします。
4. 「+ロール」をクリックします。
5. 「追加」をクリックします。
6. 「ロール検索」ダイアログにて、「IM-BPM 管理者（im_bpm_manager）」ロールを検索して選択し、「決定」をクリックします。
7. 追加された「IM-BPM 管理者（im_bpm_manager）」ロールの「権限」セレクトボックスにて、「登録・更新・削除可能」を選択します。

図：「権限設定」

ロジックフローを作成する

ワークフローの処理対象者プラグインを、ロジックフローにて記述します。
入力としてプロセスの変数情報が渡されるので、処理対象者情報を取り出して出力に設定するのが、このロジックフローの役割です。

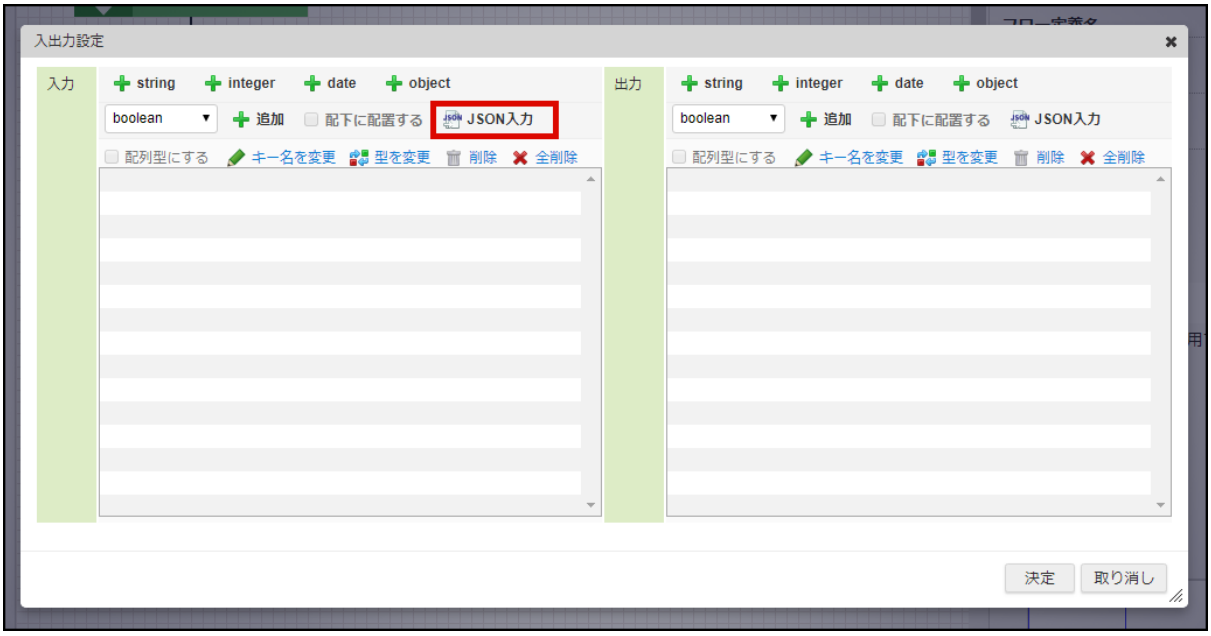
「サイトマップ」→「LogicDesigner」→「フロー定義一覧」から「ロジックフロー定義一覧」を表示し、「新規作成」をクリックしてロジックフローの作成を開始します。



図：完成イメージ（ロジックフロー）

入出力設定を行う

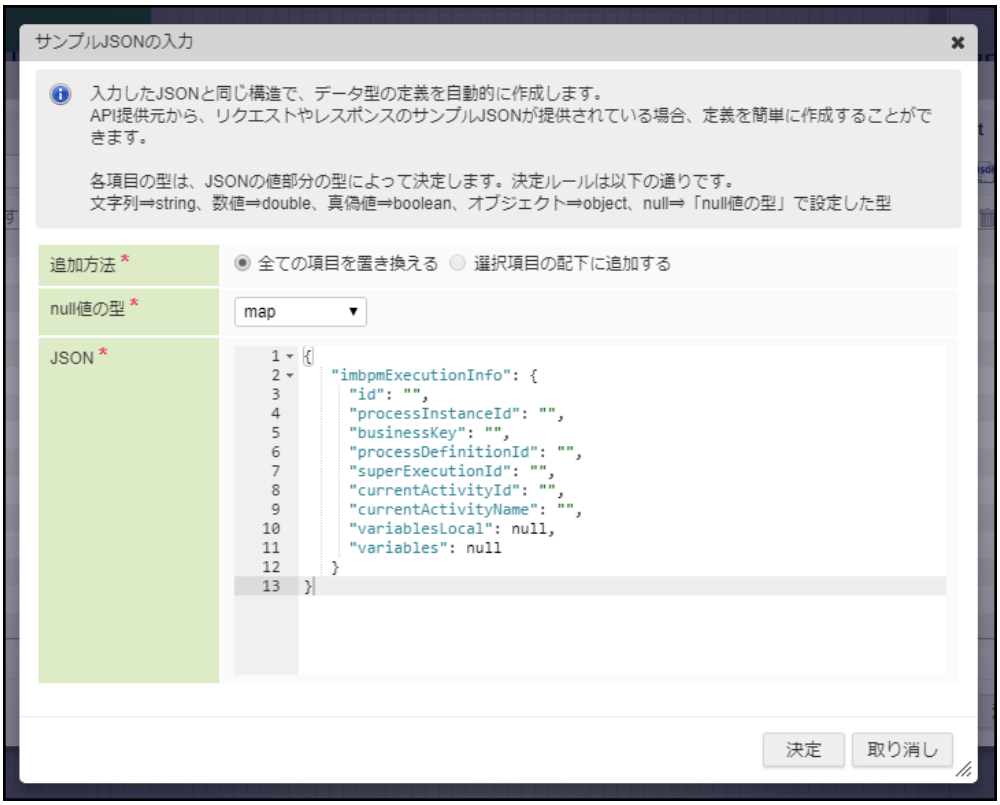
1. 「入出力設定」をクリックします。
2. 「入力」ペインにて、「JSON入力」をクリックします。



図：「入出力設定」 - 「JSON入力」

3. 「null値の型」にて `map` を選択します。
4. 「JSON」の内容を、以下の文字列で置き換えます。

```
{
  "imbpmExecutionInfo": {
    "id": "",
    "processInstanceId": "",
    "businessKey": "",
    "processDefinitionId": "",
    "superExecutionId": "",
    "currentActivityId": "",
    "currentActivityName": "",
    "variablesLocal": null,
    "variables": null
  }
}
```



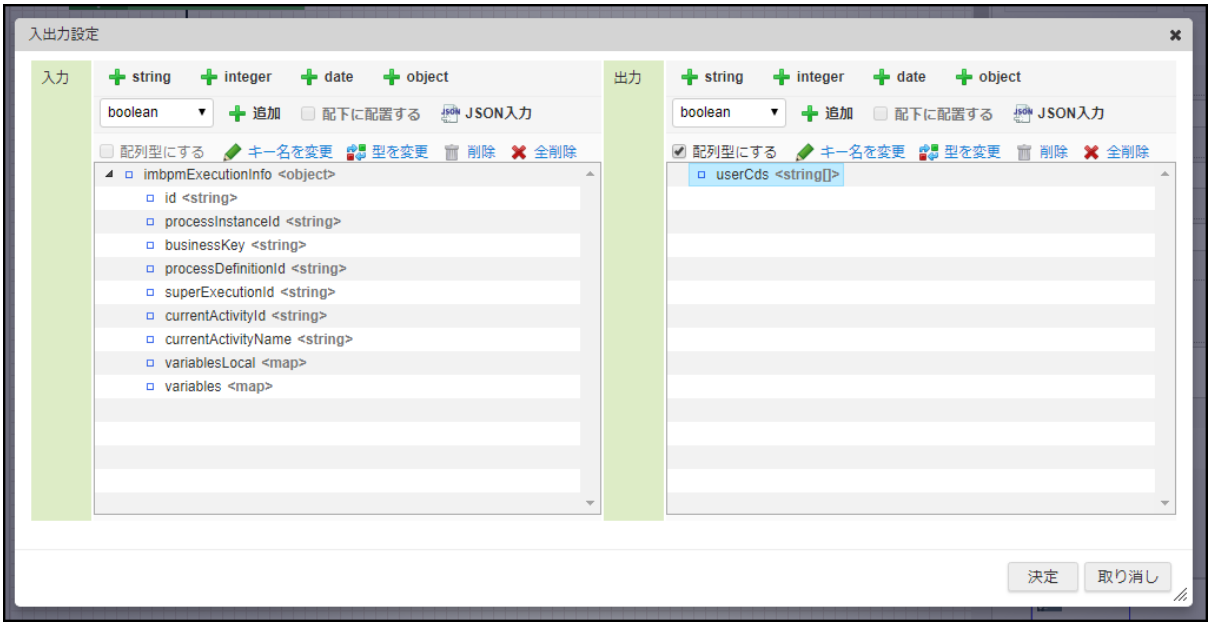
図：「サンプルJSONの入力」

5. 「決定」をクリックします。

6. 「出力」ペイン上部にある、「+string」をクリックします。

7. 「出力」ペイン下部に新しく値が設定されたのを確認し、名称に userCds を入力します。

8. 追加した userCds を選択している状態で、「配列型にする」チェックボックスをオンにします。



図：「入出力設定」（設定後）

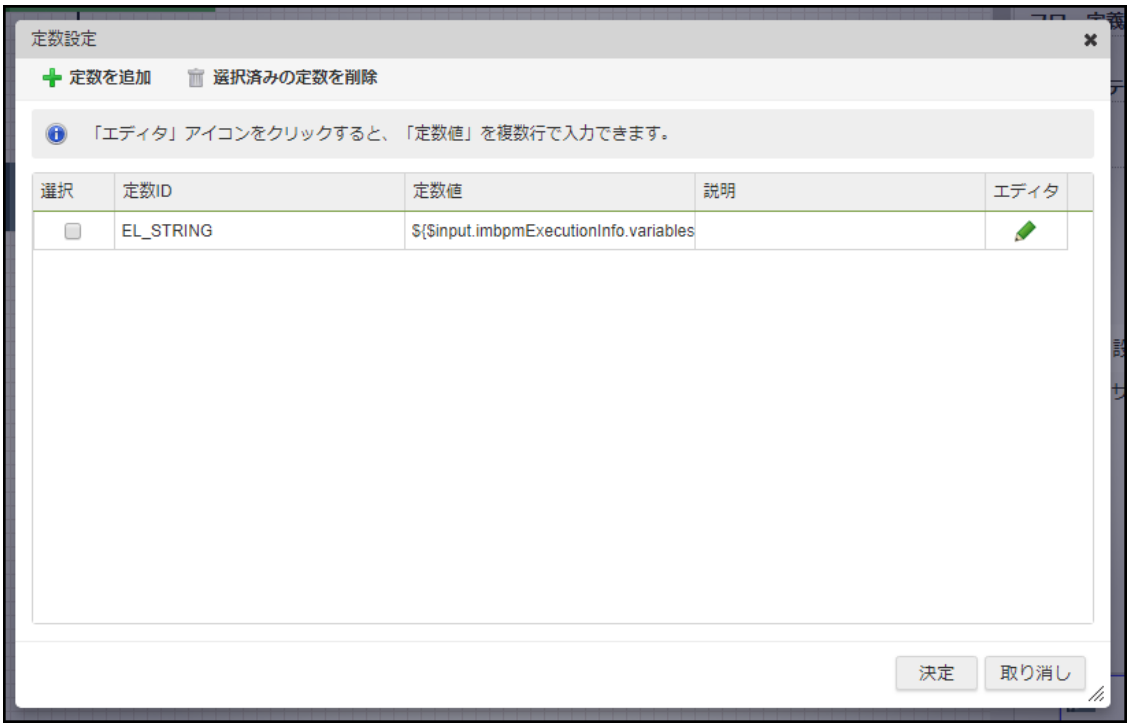
9. 「決定」をクリックします。

> **コラム**
>
> ロジックフローの入出力設定については、「IM-LogicDesigner ユーザ操作ガイド」-「入出力/変数/定数を設定する」を参照してください。
>
> また、処理対象者プラグインの入出力設定については、「IM-Workflow 管理者操作ガイド」-「ロジックフローの入出力設定」を参照してください。
> このチュートリアルではIM-Workflowの案件情報を利用しないため、IM-BPMが提供する拡張入力値のみを定義しています。

定数設定を行う

1. 「定数設定」をクリックします。
2. 「+定数を追加」をクリックします。
3. 「定数ID」に EL_STRING を入力します。
4. 「定数値」に以下の値を入力します。

```
${$input.imbpmExecutionInfo.variables.im_bpm_system_variables.im_operation_users["select_target"]}
```



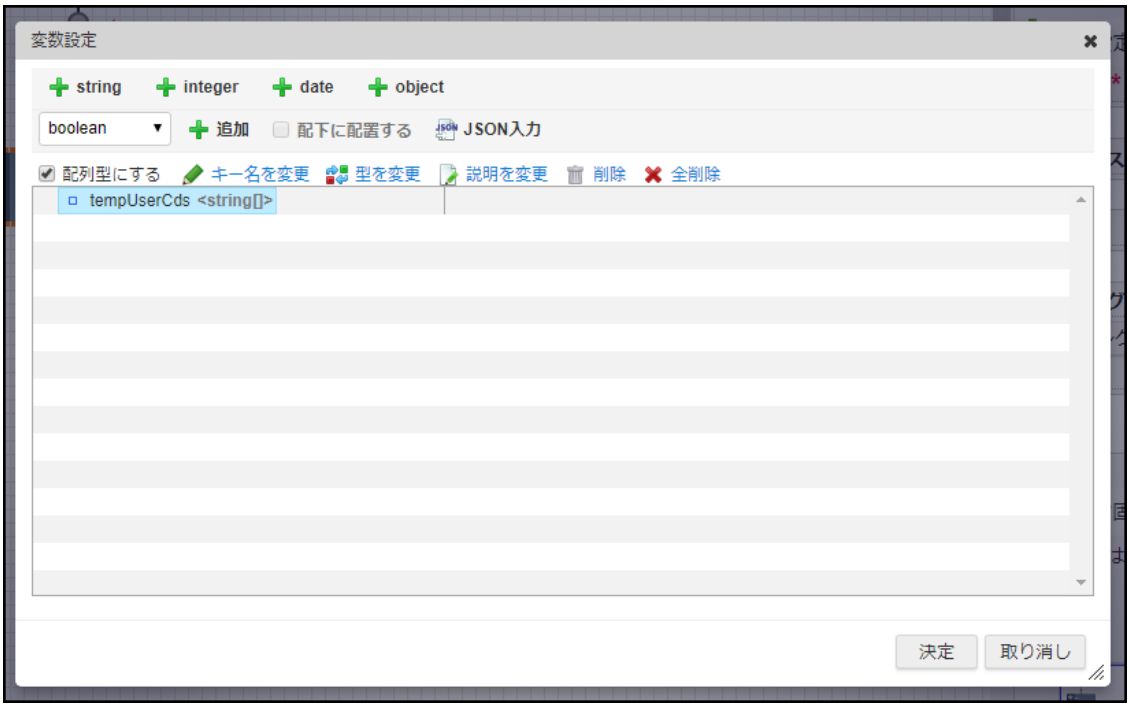
図：「定数設定」（設定後）

5. 「決定」をクリックします。

> **コラム**
>
> ロジックフローの定数設定については、「IM-LogicDesigner ユーザ操作ガイド」-「入出力/変数/定数を設定する」を参照してください。

変数設定を行う

1. 「変数設定」をクリックします。
2. 「+string」をクリックします。
3. 下部に新しく値が設定されたのを確認し、名称に tempUserCds を入力します。
4. 追加した tempUserCds を選択している状態で、「配列型にする」チェックボックスをオンにします。



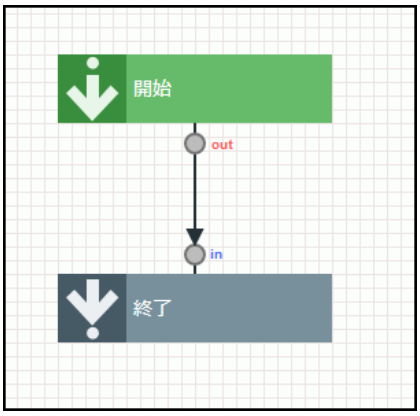
図：「変数設定」（設定後）

5. 「決定」をクリックします。

> **コラム**
>
> ロジックフローの変数設定については、「IM-LogicDesigner ユーザ操作ガイド」-「入出力/変数/定数を設定する」を参照してください。

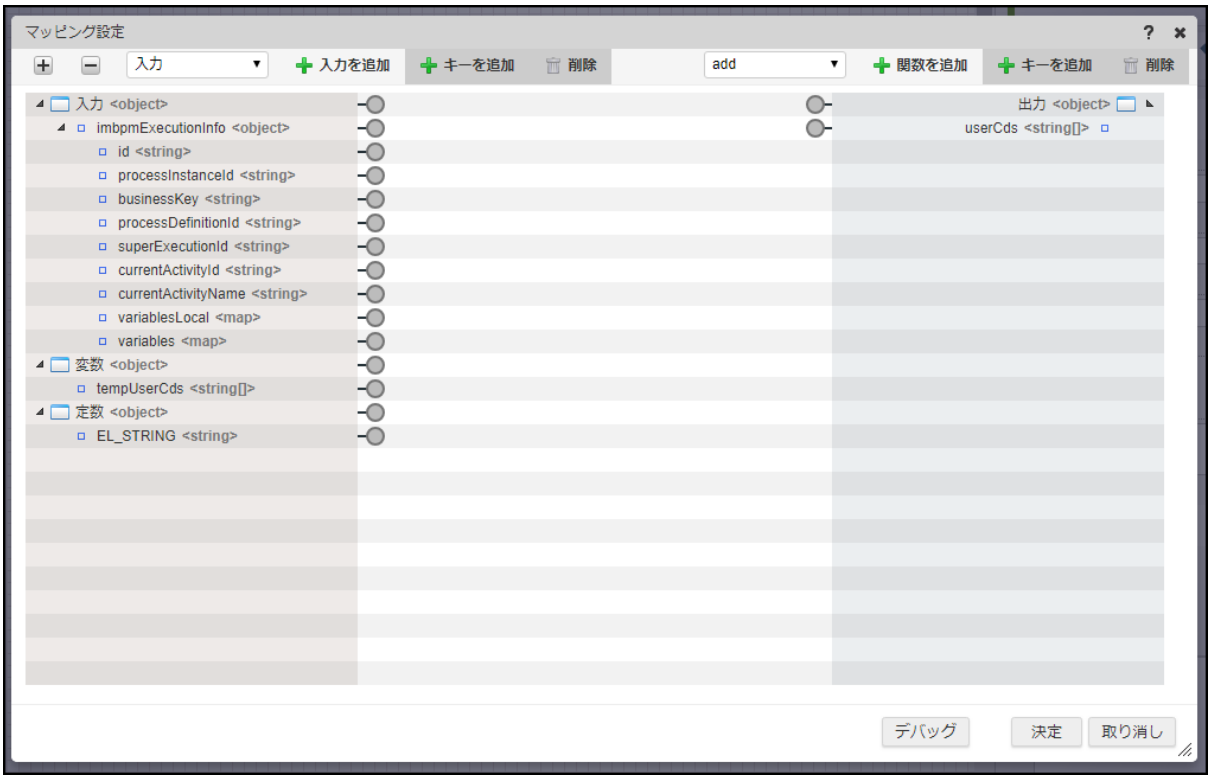
エレメントのマッピングを行う

1. 「開始」エレメントから「終了」エレメントに向けて、線を接続します。



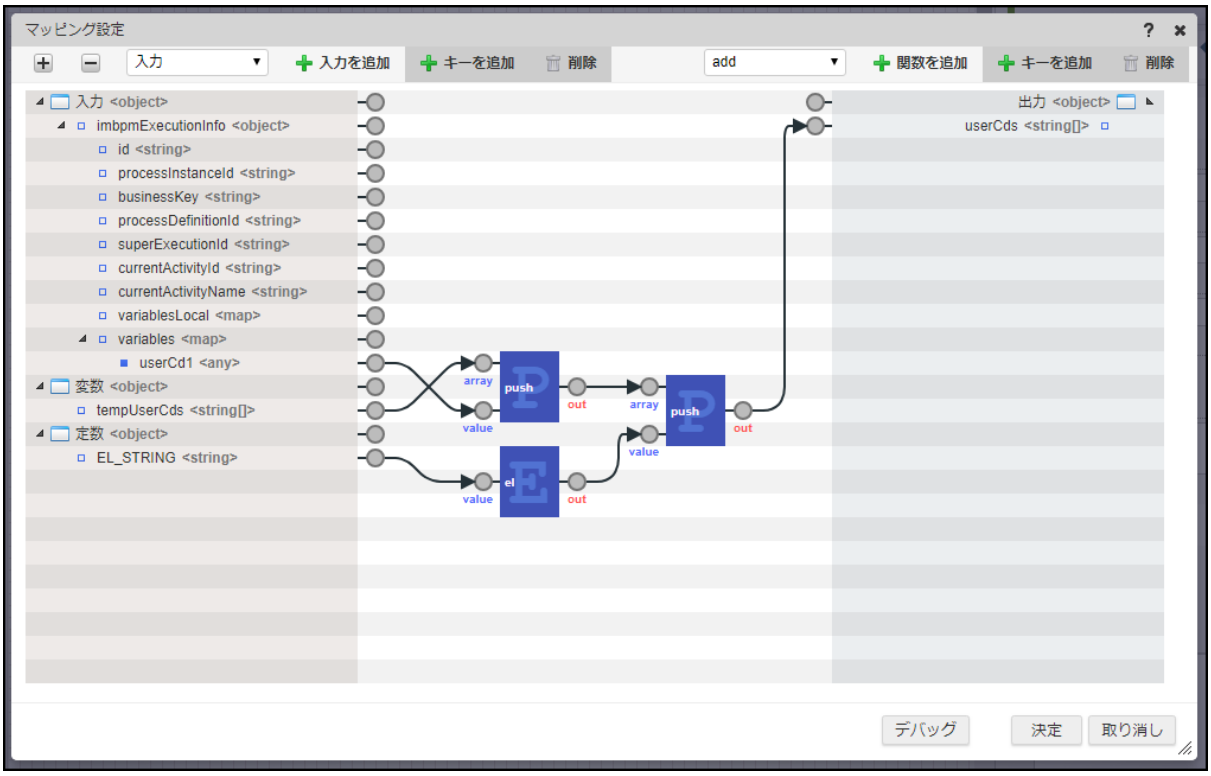
図：エレメントの接続

2. 「終了」エレメントをダブルクリックし、「マッピング設定」を表示します。



図：「終了」エレメント - 「マッピング設定」

3. 入力 variables<map> をクリックして選択し、「+キーを追加」をクリックします。
4. variables<map> 配下にキーが追加されたのを確認し、名称に userCd1 を入力します。
5. 「関数選択」セレクトボックスにて、push を選択し、「+関数を追加」をクリックします。
6. 変数 tempUserCds<string[]> から、追加したpush関数の入力「array」に対して線を引きます。
7. 入力 userCd1 から、追加したpush関数の入力「value」に対して線を引きます。
8. 「関数選択」セレクトボックスにて、el を選択し、「+関数を追加」をクリックします。
9. 定数 EL_STRING から、追加したel関数の入力「value」に対して線を引きます。
10. 「関数選択」セレクトボックスにて、push を選択し、「+関数を追加」をクリックします。
11. 1つ目のpush関数の出力「out」から、2つ目のpush関数の入力「array」に対して線を引きます。
12. el関数の出力「out」から、2つ目のpush関数の入力「value」に対して線を引きます。
13. 2つ目のpush関数の出力「out」から、出力 userCds<string[]> に対して線を引きます。



図：「終了」エレメント - 「マッピング設定」（設定後）

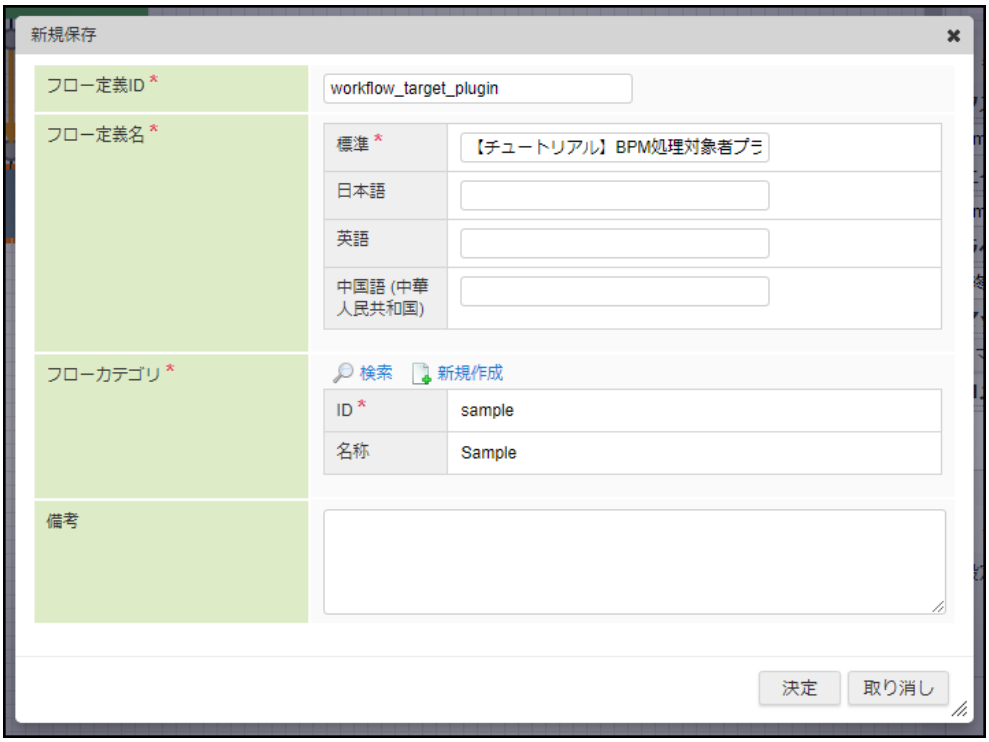
14. 「決定」をクリックします。

> **コラム**
>
> エレメントのマッピングについては、「IM-LogicDesigner ユーザ操作ガイド」-「エレメントのマッピングを設定する」を参照してください。
> push関数については、「IM-LogicDesigner仕様書」-「push」を参照してください。
> el関数については、「IM-LogicDesigner仕様書」-「el」を参照してください。
> IM-LogicDesignerにおけるEL式については、「IM-LogicDesigner仕様書」-「EL式」を参照してください。

> **コラム**
>
> ここでは、入力に与えられたオブジェクトから以下の2つの値を取り出して、出力にマッピングしています。
>
> - 変数「userCd1」の値
>   Formaアプリケーションの識別ID「userCd1」のフィールドの値が、連携先プロセスの変数に格納されたものです。
> - 暗黙オブジェクト「im_operation_users」の値
>   システム変数「im_bpm_system_variables」内に、Map型の暗黙オブジェクトとして存在しています。
>   ユーザタスクのプロパティに設定したIDをキーとして与えることで、タスクを処理したユーザコードを取得できます。
>   このチュートリアルでは、IDが「select_target」のユーザタスクを処理したユーザコードを、el関数を利用したEL式で取得しています。
>
> 暗黙オブジェクトについては、「IM-BPM 仕様書」-「EL式」を参照してください。

新規保存を行う

1. 「新規保存」をクリックします。
2. 「フロー定義ID」に `workflow_target_plugin` を入力します。
3. 「フロー定義名」 - 「標準」に `【チュートリアル】BPM処理対象者プラグイン` を入力します。
4. 「フローカテゴリ」を検索し、 `Sample` を選択します。



図：「新規保存」

5. 「決定」をクリックします。

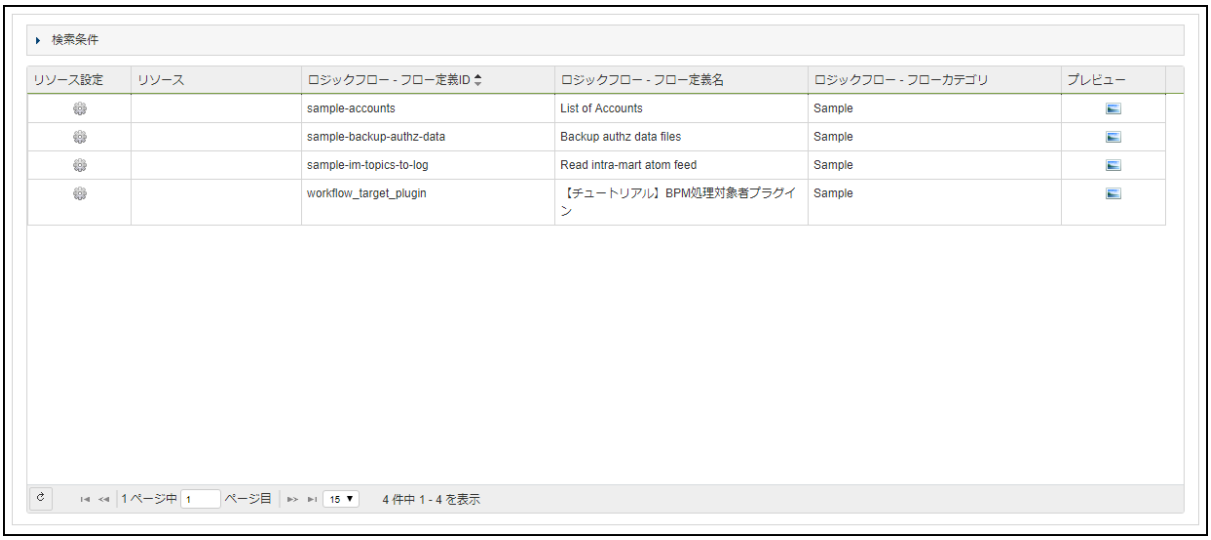
> **コラム**
>
> ここで設定した「フロー定義名」は、ワークフローの「履歴参照」画面などに処理者として表示されます。

ワークフローのルート定義・フロー定義を作成する

作成したロジックフローをワークフロー「ロジックフロー管理」に登録して、処理対象者プラグインとして指定できるようにします。
その後、ワークフローのルート定義・フロー定義を作成します。
ルート定義の作成時、承認ノードの処理対象者プラグインとして作成したロジックフローを指定します。

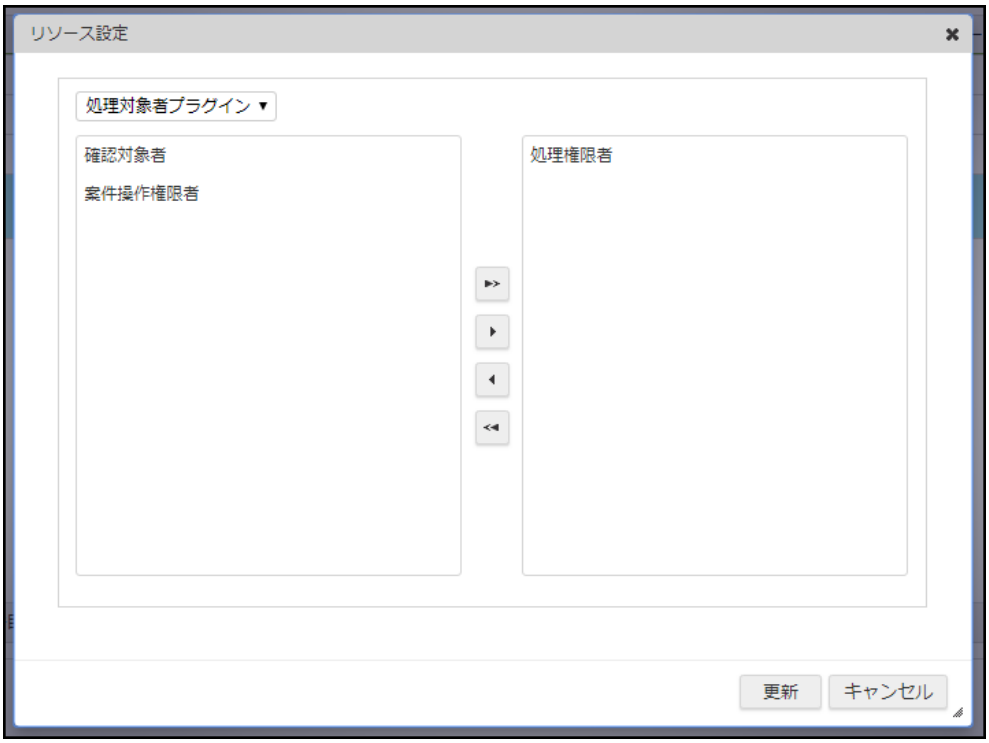
ロジックフローを管理する

1. 「サイトマップ」→「ワークフロー」→「ワークフロー管理者」→「ロジックフロー管理」から、「ロジックフロー管理」画面を表示します。



図：「ロジックフロー管理」

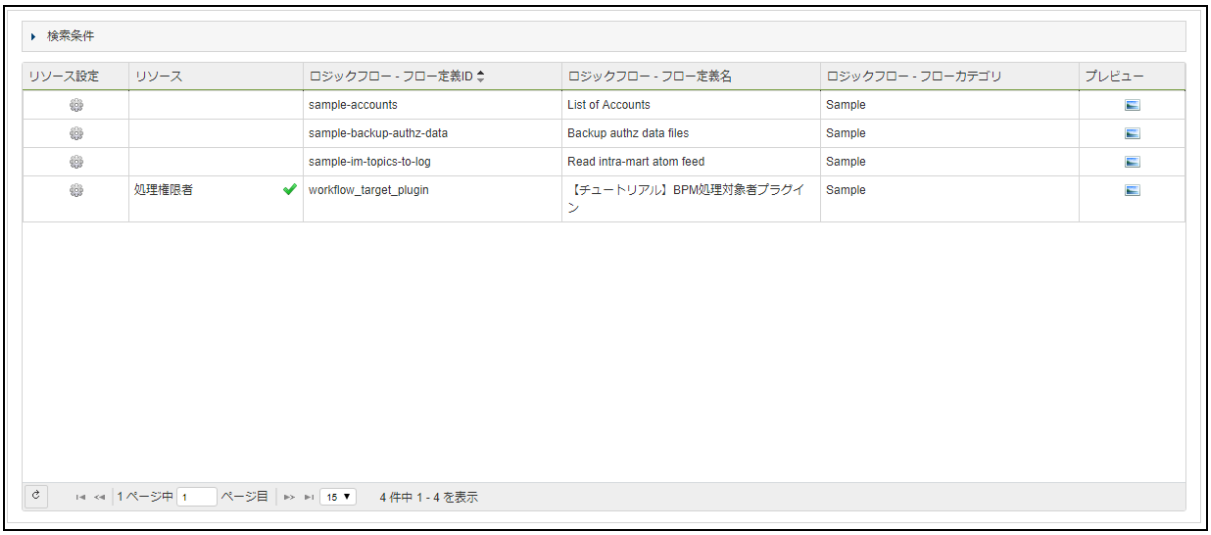
2. 作成した「【チュートリアル】BPM処理対象者プラグイン」について、「リソース設定」アイコンをクリックします。
3. 「リソース設定」ダイアログにて、セレクトボックスで「処理対象者プラグイン」を選択します。
4. 左ペインに表示されている「処理権限者」をクリックして選択します。
5. 中央に表示されている ▸ をクリックし、「処理権限者」を右ペインに移動します。



図：「リソース設定」

6. 「更新」をクリックします。

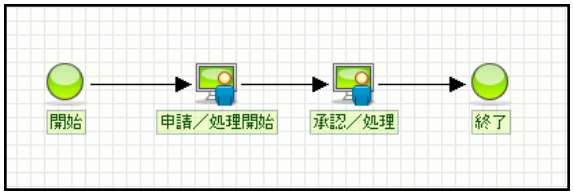
7. 「【チュートリアル】BPM処理対象者プラグイン」が、処理対象者プラグイン（処理権限者）として利用可能になりました。



図：「ロジックフロー管理」（設定後）

> **コラム**
>
> ロジックフローの管理については、「IM-Workflow 管理者操作ガイド」-「各機能で利用するロジックフローを管理する」を参照してください。

ルート定義を作成する



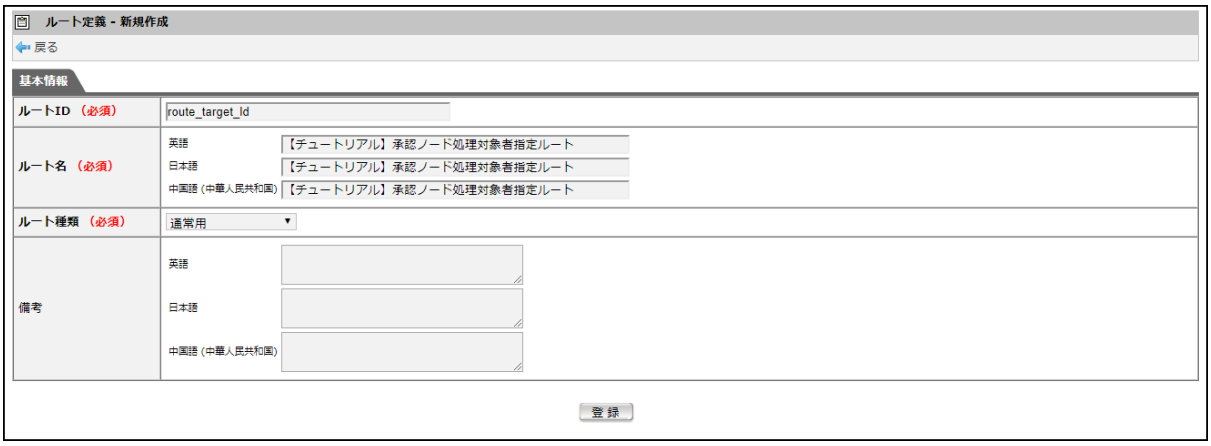
図：完成イメージ（ルート定義）

1. 「サイトマップ」→「ワークフロー」→「ワークフロー管理者」→「マスタ定義」→「ルート定義」から、「ルート定義」画面を表示します。

2. 「新規作成」をクリックします。

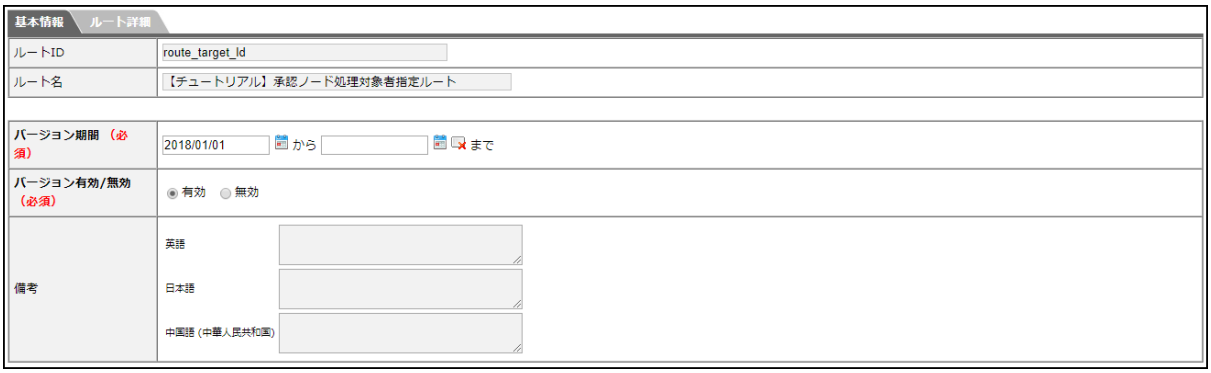
3. 「ルートID」に、 `route_target_Id` を入力します。

4. 「ルート名」に表示されているロケール分すべてに、 `【チュートリアル】承認ノード処理対象者指定ルート` を入力します。



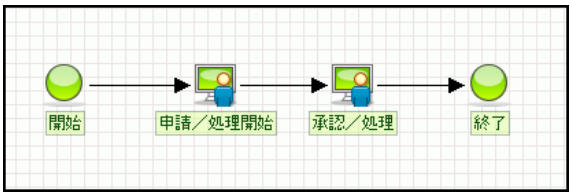
図：「ルート定義 - 新規作成」

5. 「登録」をクリックします。
6. 表示されている「バージョン」タブ画面にて、「新規作成」をクリックします。
7. 「基本情報」タブ画面にて、「バージョン期間」 の範囲開始日に 2018/01/01 を入力します。
8. 「バージョン有効/無効」にて、「有効」を選択します。



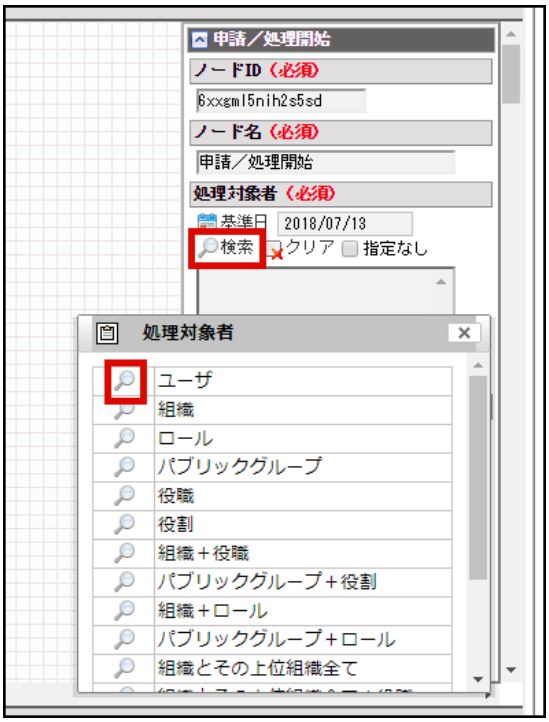
図：「ルート定義 - バージョン - 新規作成」 - 「基本情報」

9. 「ルート詳細」タブをクリックします。
10. 「承認／処理」ノードをドラッグし、ドロップして配置します。
11. 「申請／処理開始」ノードから「承認／処理」ノードに線を引きます。
12. 同様に、「承認／処理」ノードから「終了」ノードに線を引きます。



図：「ルート定義 - バージョン - 新規作成」 - 「ルート詳細」

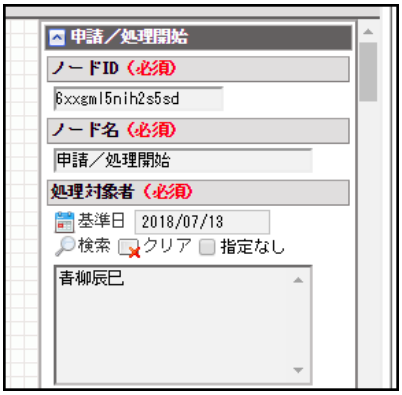
13. 「申請／処理開始」ノードをクリックし、選択します。
14. 右側に表示されたノード詳細エリアにて、「処理対象者」の「検索」をクリックします。
15. 表示された「処理対象者」一覧にて、「ユーザ」を選択し、虫眼鏡アイコンをクリックします。



図：「ルート定義 - バージョン - 新規作成」 - 「申請／処理開始」ノード詳細

16. 表示された「ユーザ検索」ダイアログにて、「青柳辰巳」を検索して選択し、「決定」をクリックします。

17. 右側に表示されたノード詳細エリアにて、「処理対象者」に「青柳辰巳」が設定されているのを確認します。

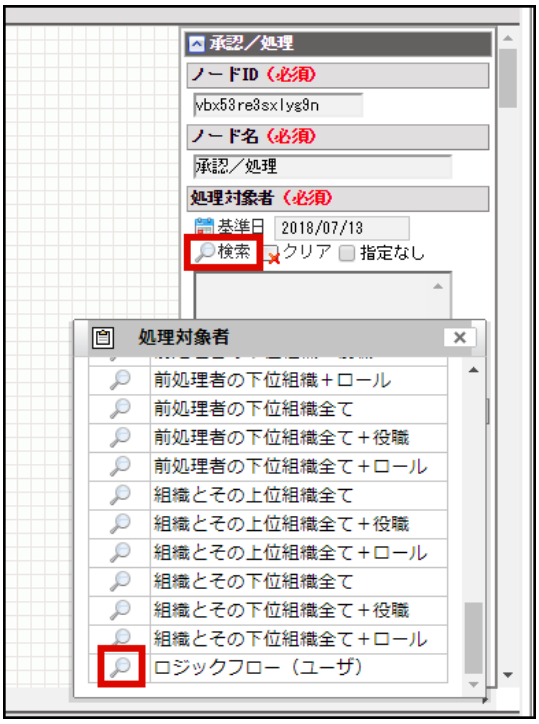


図：「ルート定義 - バージョン - 新規作成」 - 「申請／処理開始」ノード詳細（設定後）

18. 「承認／処理」ノードをクリックし、選択します。

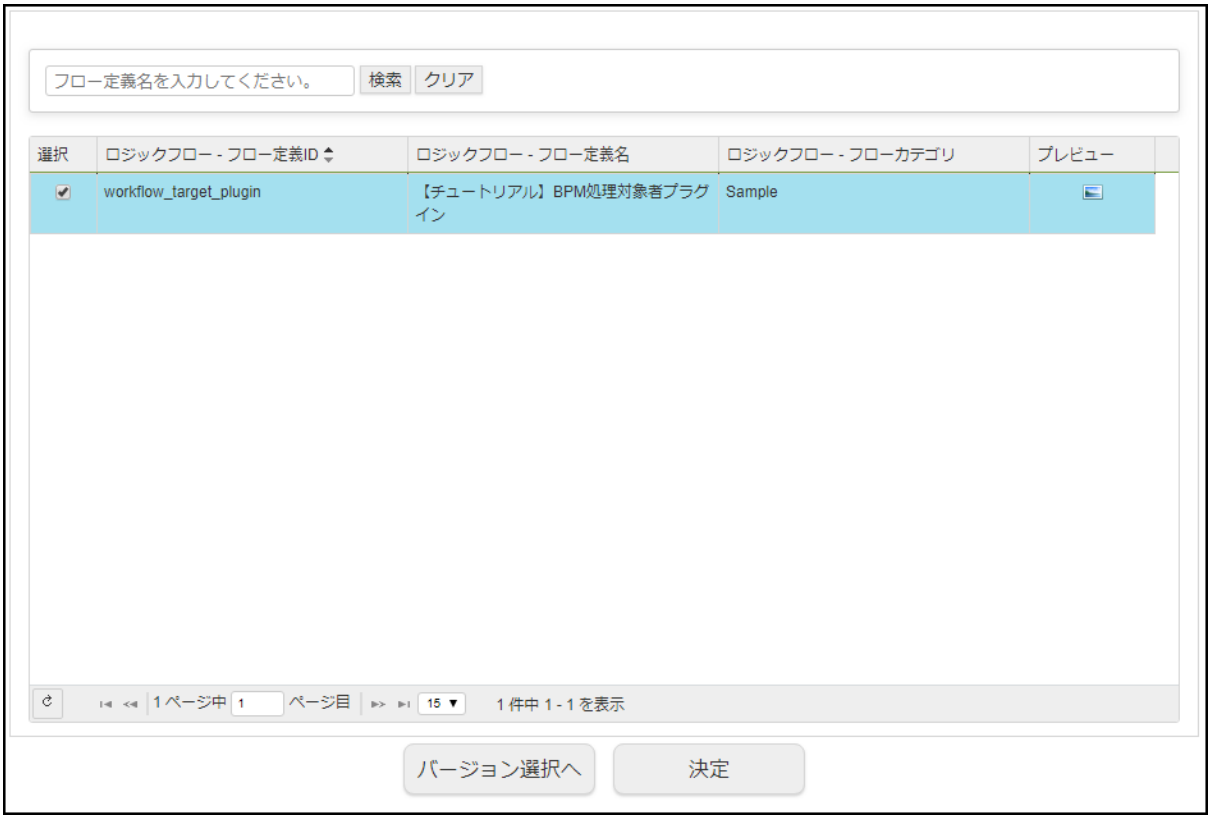
19. 右側に表示されたノード詳細エリアにて、「処理対象者」の「検索」をクリックします。

20. 表示された「処理対象者」一覧にて、「ロジックフロー（ユーザ）」を選択し、虫眼鏡アイコンをクリックします。



図：「ルート定義 - バージョン - 新規作成」 - 「承認／処理」ノード詳細

21. 表示された「ロジックフロー検索」ダイアログにて、 「【チュートリアル】BPM処理対象者プラグイン」の「選択」チェックボックスをオンにします。



図：「ロジックフロー検索」選択イメージ

22. 「決定」をクリックします。

23. 右側に表示されたノード詳細エリアにて、「処理対象者」に「【チュートリアル】BPM処理対象者プラグイン」が設定されているのを確認します。

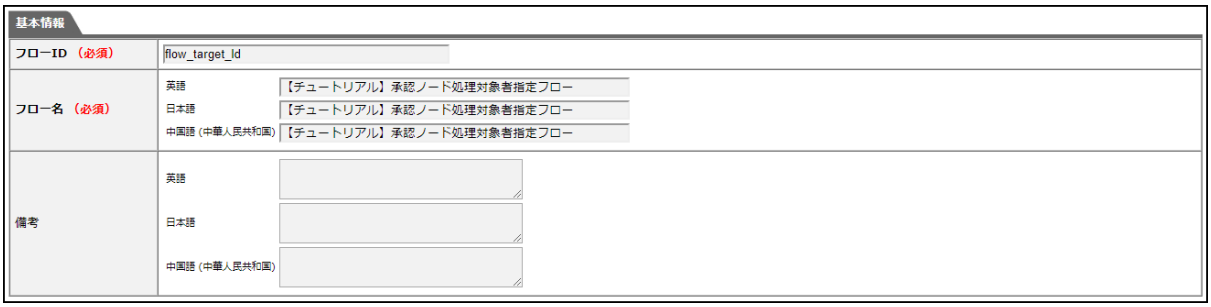
図：「ルート定義 - バージョン - 新規作成」 - 「承認／処理」ノード詳細（設定後）

24. 「登録」をクリックします。

> **コラム**
>
> ルート定義の作成については、「IM-Workflow 管理者操作ガイド」-「ルート定義を登録・設定する」を参照してください。
> ロジックフローの選択については、「IM-Workflow 管理者操作ガイド」-「利用するロジックフローを選択する」を参照してください。

フロー定義を作成する

1. 「サイトマップ」→「ワークフロー」→「ワークフロー管理者」→「マスタ定義」→「フロー定義」から、「フロー定義」画面を表示します。
2. 「新規作成」をクリックします。
3. 「フローID」に、 `flow_target_id` を入力します。
   プロセス定義の作成時、申請タスクのプロパティ「フローID」に指定したものです。
4. 「フロー名」に表示されているロケール分すべてに、 `【チュートリアル】承認ノード処理対象者指定フロー` を入力します。



図：「フロー定義 - 新規作成」

5. 「登録」をクリックします。
6. 表示されている「バージョン」タブ画面にて、「新規作成」をクリックします。
7. 「基本情報」タブ画面にて、「バージョン期間」 の範囲開始日に `2018/01/01` を入力します。
8. 「バージョン有効/無効」にて、「有効」を選択します。
9. 「コンテンツ」にて「検索」をクリックし、「スクリプト開発モデル」を選択します。
10. 「ルート」にて「検索」をクリックし、「【チュートリアル】承認ノード処理対象者指定ルート」を選択します。

図：「フロー定義 - バージョン - 新規作成」 - 「基本情報」

11. 「登録」をクリックします。

> **コラム**
>
> フロー定義の作成については、「IM-Workflow 管理者操作ガイド」-「フロー定義を登録・設定する」を参照してください。
> また、このチュートリアルではルート定義とフロー定義は作成していますが、コンテンツ定義についてはIM-Workflowサンプルの「スクリプト開発モデル」を利用しています。

実行結果を確認する

作成したプロセスを開始し、ユーザタスクを処理してユーザを指定します。
申請タスクで申請されたワークフローの処理対象者が、ユーザタスクを処理したユーザと指定したユーザの2人であることを確認します。

プロセスを開始し、ユーザタスクを処理する

1. 作成したプロセスを実行環境にデプロイします。
2. 「青柳辰巳」でログインします。
3. 「プロセス開始一覧」画面を表示し、デプロイしたプロセスを開始します。
4. 「タスク一覧」画面を表示し、「グループタスク」に存在する「ユーザ指定」タスクを担当にします。
5. 「個人タスク」に移動した「ユーザ指定」タスクを処理します。

図：「個人タスク」 - 「ユーザ指定」タスク - 「処理」

6. 作成したFormaアプリケーション「【チュートリアル】処理対象者選択」のフォームが表示されます。
   「ユーザ名」にて「検索」を行い、「上田辰男」を選択します。

図：「【チュートリアル】処理対象者選択」入力画面

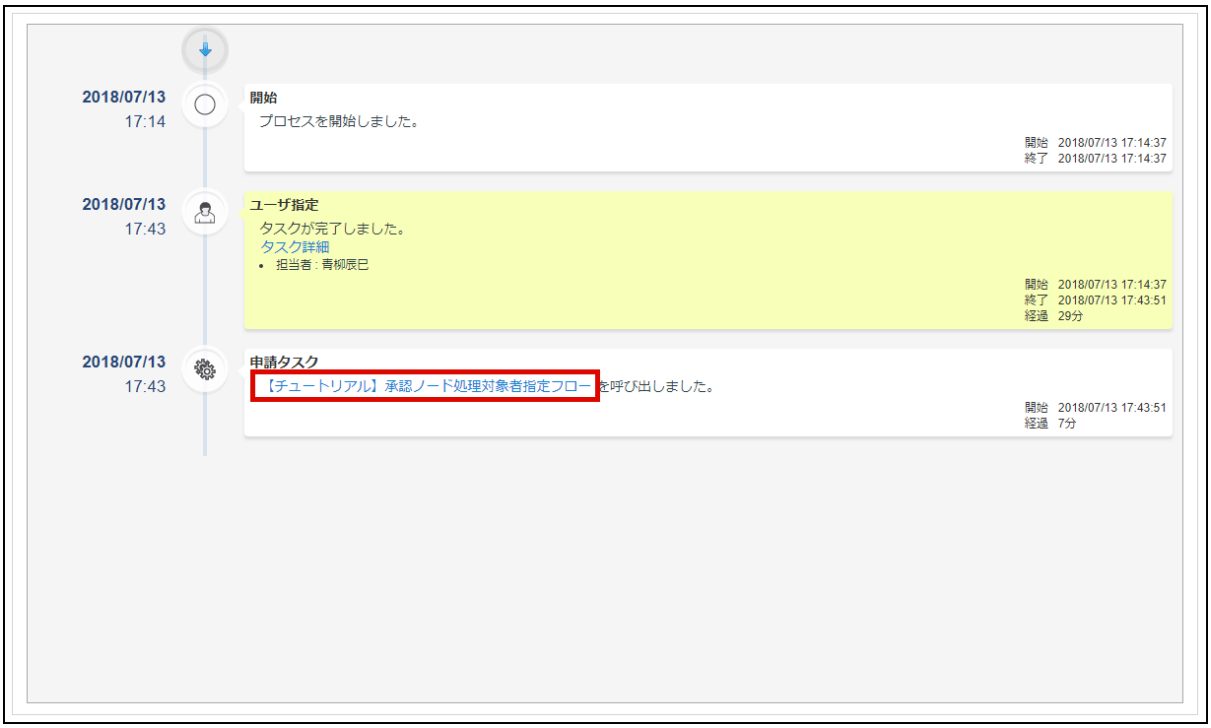
7. 「登録」をクリックします。

ワークフローの処理対象者を確認する

1. 「処理済一覧」画面を表示し、「処理済タスク」に存在する「ユーザ指定」タスクの「履歴」アイコンをクリックします。

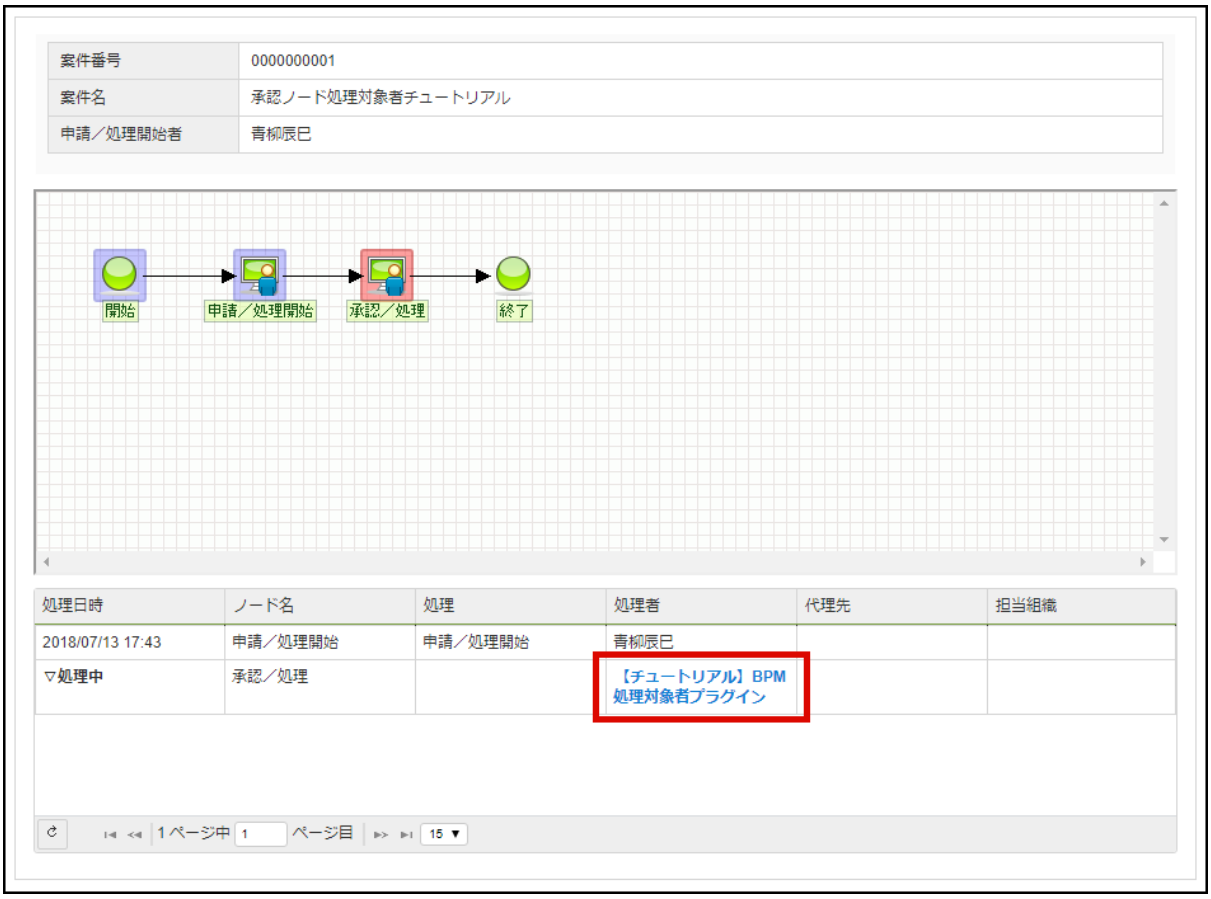
図：「処理済タスク」 - 「ユーザ指定」タスク - 「履歴」

2. 「履歴」画面が表示されます。
「申請タスク」に記載されている、「【チュートリアル】承認ノード処理対象者指定フロー」リンクをクリックします。



図：「履歴」 - 「申請」タスク - ワークフロー「履歴参照」へのリンク

3. ワークフローの「履歴参照」ダイアログが表示されます。
「承認／処理」ノードの「処理者」列に表示されている「【チュートリアル】BPM処理対象者プラグイン」リンクをクリックします。



図：ワークフロー「履歴参照」 - 「承認／処理」ノード - 「処理対象者状況確認」へのリンク

4. 「処理対象者状況確認」ダイアログが表示されます。
   以下の2人のユーザが表示されていることを確認します。
   - 上田辰男
   - 青柳辰巳

図：「処理対象者状況確認」

本番環境への適用

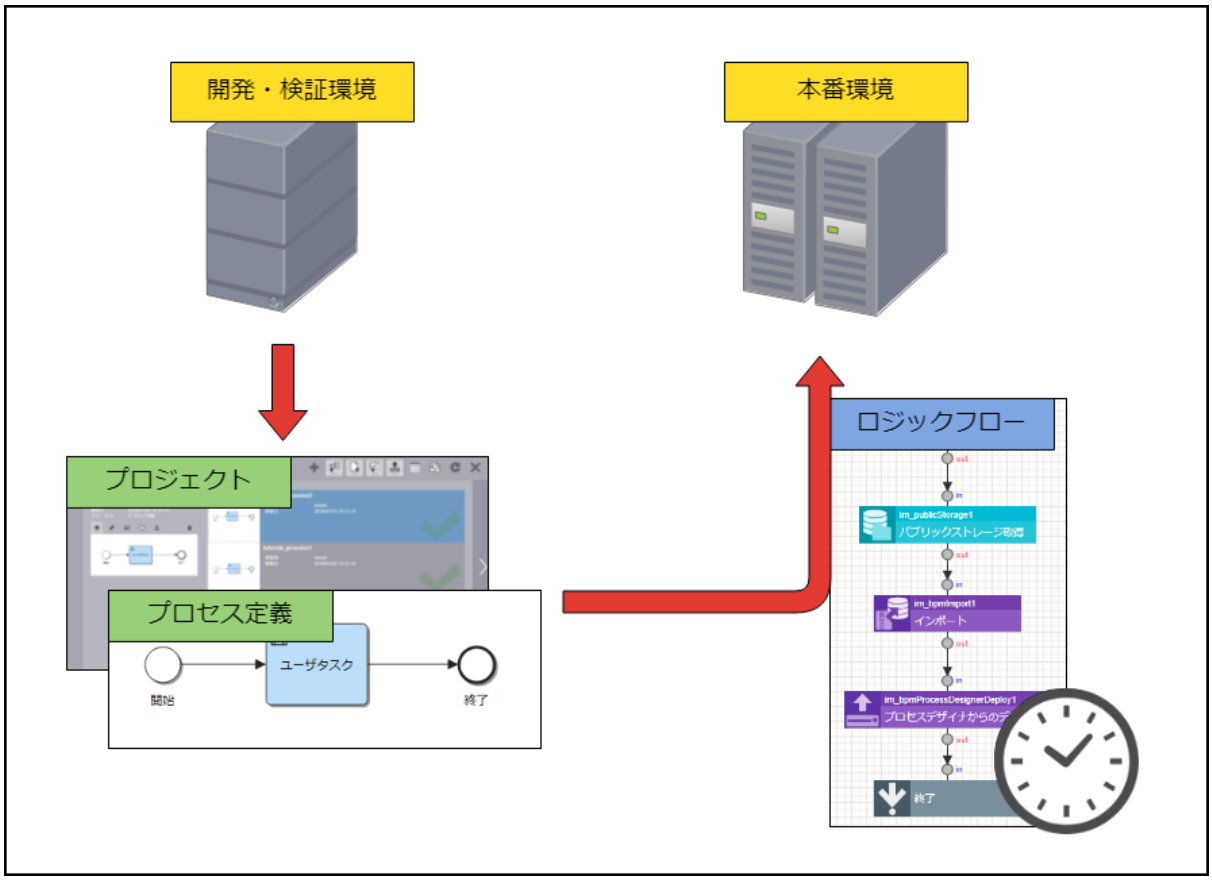
プロセス定義を本番環境へ適用するプロセスの管理方法について説明します。
このチュートリアルは「*事前準備*」が完了していることを前提としています。

プロセス定義のデプロイをスケジューリングする

このチュートリアルでは、検証環境から本番環境へのプロセス定義のデプロイをスケジューリングする方法を解説します。

IM-LogicDesigner の IM-BPMタスク を利用してプロセスデザイナのプロジェクトのインポート、プロセス定義のデプロイを行います。
IM-BPMタスク についての詳細は「IM-LogicDesigner仕様書」-「IM-BPM」を参照してください。
プロセスデザイナのプロジェクトをインポートする際に、パブリックストレージに配置したプロジェクトを使用します。

また、ジョブを利用したロジックフローの実行によって、プロセス定義のデプロイをスケジューリングします。



図：概要

> **注意**
>
> このチュートリアルの「ロジックフロー定義」は、IM-BPM for Accel Platform 2018 Summer(Tiffany) 以降のバージョンで動作します。

> **コラム**
>
> ストレージに対しての操作、およびジョブスケジューラの利用はテナント管理者が行えます。
> 詳細については、以下のリンク先を参照してください。
>
> - 「システム管理者操作ガイド」-「ファイル操作」
> - 「ジョブスケジューラ仕様書」

コラム

このチュートリアルで作成するロジックフロー定義のサンプルを以下のリンクからダウンロードできます。
im_logicdesigner-data-process_auto_deploy.zip

このサンプルはロジックフロー定義の「インポート」機能でインポートできます。
「ロジックフロー定義」のインポートについての詳細は「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」を参照してください。

- プロセスデザイナのプロジェクトをエクスポートする
- プロセスデザイナのプロジェクトをパブリックストレージに配置する
- プロセスデザイナのプロジェクトのインポートとプロセス定義のデプロイを行うロジックフロー定義を作成する
- ロジックフローを実行するジョブネットを作成する
- 実行結果を確認する

プロセスデザイナのプロジェクトをエクスポートする

検証環境からリリース対象のプロセスデザイナのプロジェクトをエクスポートします。
プロセスデザイナのプロジェクトのエクスポートの詳細については「IM-BPM ユーザ操作ガイド」-「エクスポート」を参照してください。

コラム

このチュートリアルで使用するプロセスデザイナのプロジェクトのサンプルを、以下のリンクからダウンロードできます。
process_auto_deploy.zip

プロセスデザイナのプロジェクトをパブリックストレージに配置する

このチュートリアルでエクスポートしたプロセスデザイナのプロジェクトを、パブリックストレージに配置します。

1. 「サイトマップ」→「テナント管理」→「ファイル操作」から、「ファイル操作」画面を表示します。
2. 「ファイル操作」画面、左側のツリーからパブリックストレージの直下を選択します。

図：「ファイル操作」

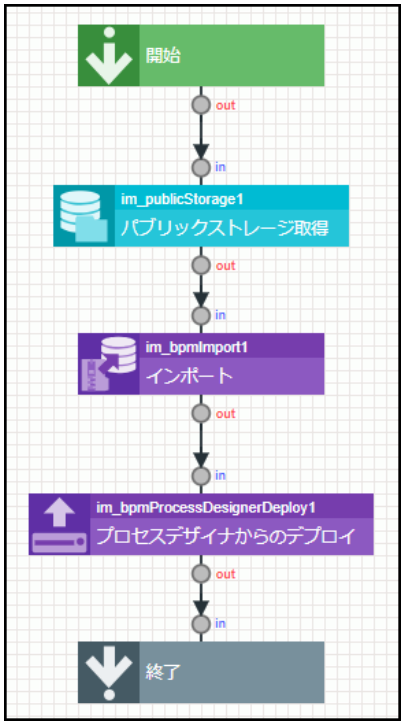
3. 「詳細」 -「ファイルアップロード」 -「ファイル追加」をクリックします。

このチュートリアルでエクスポートしたプロセスデザイナのプロジェクトを選択します。
サンプルデータを使用する場合は、 process_auto_deploy.zip を選択します。

図：「ファイル操作」 process_auto_deploy.zip

プロセスデザイナのプロジェクトのインポートとプロセス定義のデプロイを行うロジックフロー定義を作成する

プロセスデザイナのプロジェクトをプロセスデザイナへインポートし、プロセスデザイナから IM-BPM Runtime へプロセス定義のデプロイを行うロジックフロー定義を作成します。



図：完成イメージ（ロジックフロー定義）

1. 「サイトマップ」→「LogicDesigner」→「フロー定義一覧」から、「ロジックフロー定義一覧」画面を表示します。
2. 「フロー定義一覧」画面、ツールバー内の「新規作成」をクリックし、「ロジックフロー定義編集」画面を表示します。
3. 「ロジックフロー定義編集」画面上部、ヘッダ内の「入出力設定」をクリックします。

   入力値を以下のように設定します。

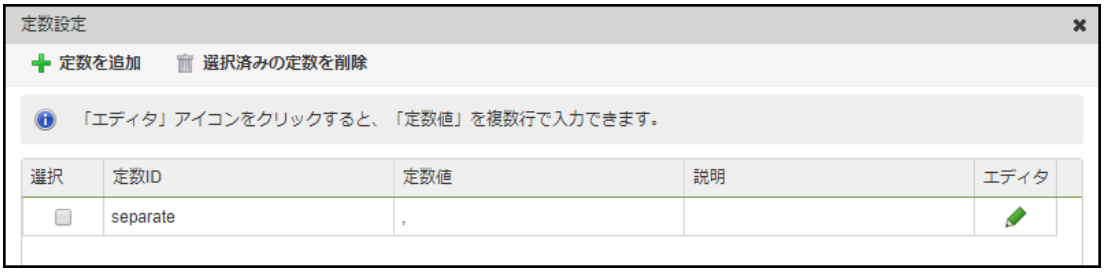
キー名	型
deploymentName	<string>
processResourceNameList	<string>
projectId	<string>
zipFilePath	<string>

図：「入出力設定」

4. 「ロジックフロー定義編集」画面上部、ヘッダ内の「定数設定」をクリックします。

定数値を以下のように設定します。

定数ID	定数値
separate	,



図：「定数設定」

5. 「パブリックストレージ取得」を配置します。

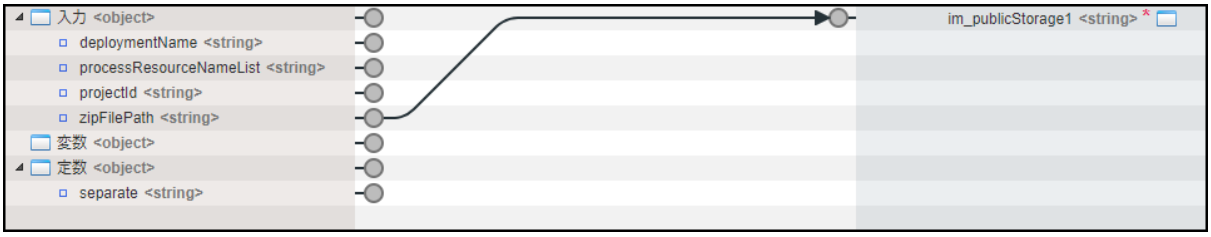
パレット内の「ストレージ操作」の一覧から「パブリックストレージ取得」を選び、フロー編集画面上に追加します。

図：「パブリックストレージ取得」

6. 「パブリックストレージ取得」のマッピング設定をします。

以下のように線を引きます。

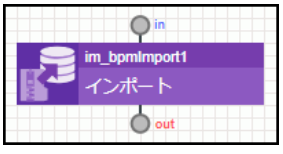
入力（始点）	出力（終点）
入力<object> - zipFilePath<string>	im_publicStorage1<string>



図：「マッピング設定」 - 「パブリックストレージ取得」

7. 「インポート」を配置します。

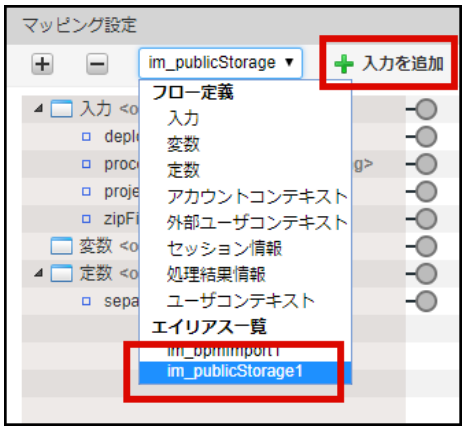
パレット内の「IM-BPM」の一覧から「インポート」を選び、フロー編集画面上に追加します。



図：「インポート」

8. 「インポート」のマッピング設定をします。

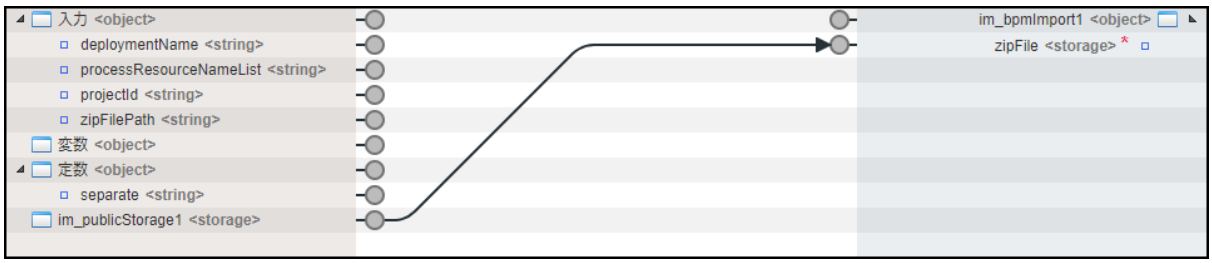
    1. 「マッピング設定」画面上部、ヘッダ内の左側に位置するセレクトボックスから「im_publicStorage1」を選択し、「入力を追加」をクリックします。



図：「マッピング設定」 - 「インポート」 - 「入力を追加」

    2. 以下のように線を引きます。

入力（始点）	出力（終点）
im_publicStorage1<storage>	im_bpmImport1<object> - zipFile<storage>

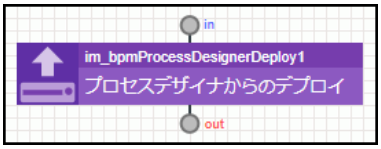


図：「マッピング設定」 - 「インポート」

> **コラム**
>
> 「インポート」についての詳細は、「IM-LogicDesigner仕様書」-「インポート」を参照してください。

9. 「プロセスデザイナからのデプロイ」を配置します。

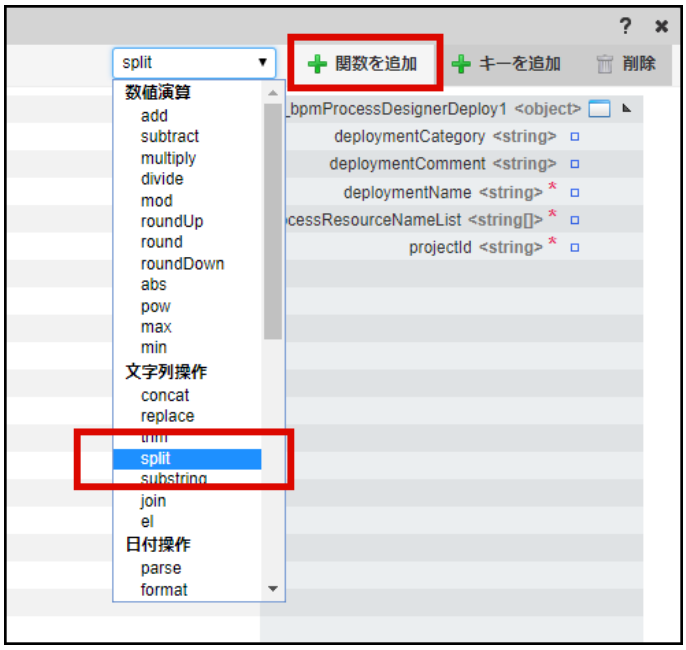
パレット内の「IM-BPM」の一覧から「プロセスデザイナからのデプロイ」を選び、フロー編集画面上に追加します。



図：「プロセスデザイナからのデプロイ」

10. 「プロセスデザイナからのデプロイ」のマッピング設定をします。

    1. 「マッピング設定」画面上部、ヘッダ内の右側に位置するセレクトボックスから「split」を選択し、「関数を追加」をクリックします。

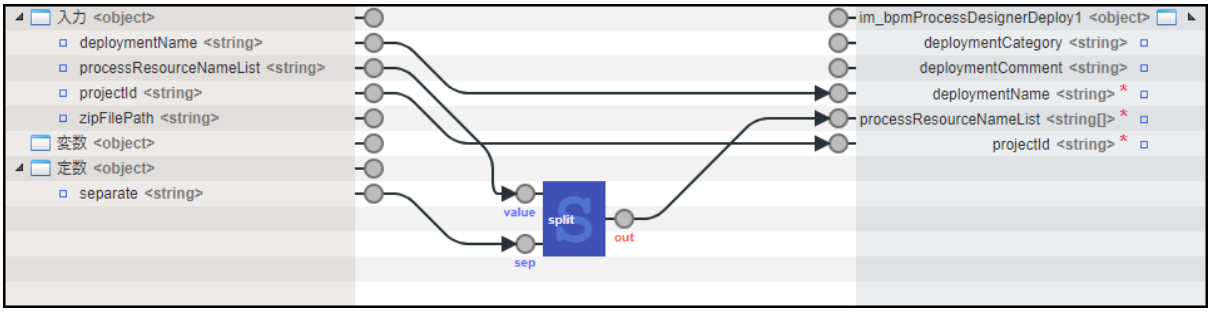


図：「マッピング設定」 - 「プロセスデザイナからのデプロイ」 - 「関数を追加」

> **コラム**
>
> マッピング関数については、「IM-LogicDesigner仕様書」-「マッピング関数一覧」を参照してください。

2. 以下のように線を引きます。

入力（始点）	出力（終点）
入力<object> - deploymentName<string>	im_bpmProcessDesignerDeploy1<object> - deploymentName<string>
入力<object> - processResourceNameList<string>	split : value
入力<object> - projectId<string>	im_bpmProcessDesignerDeploy1<object> - projectId<string>
定数<object> - separate<string>	split : sep
split : out	im_bpmProcessDesignerDeploy1<object> - processResourceNameList<string[]>



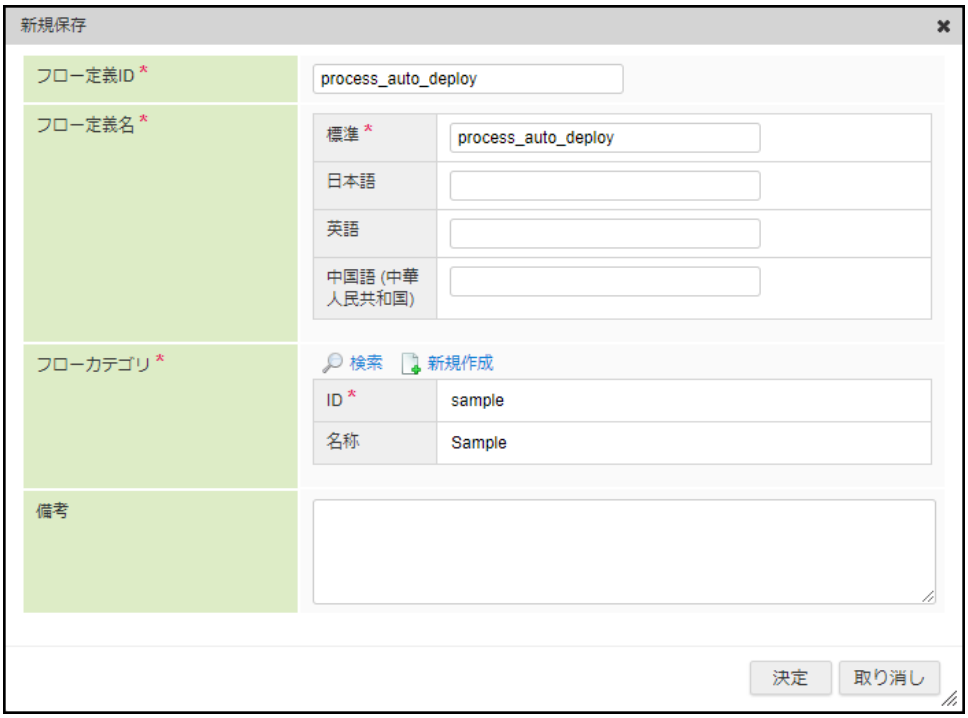
図：「マッピング設定」 - 「プロセスデザイナからのデプロイ」

> **コラム**
>
> 「プロセスデザイナからのデプロイ」についての詳細は、「IM-LogicDesigner仕様書」-「プロセスデザイナからのデプロイ」を参照してください。

11. 「ロジックフロー定義編集」画面上部、ヘッダ内の「新規保存」をクリックします。

以下のように設定し、ロジックフロー定義を新規保存します。

- フロー定義ID：process_auto_deploy
- フロー定義名：process_auto_deploy
- フローカテゴリ：
  - ID：sample
  - 名称：Sample



図：「新規保存」

ロジックフローを実行するジョブネットを作成する

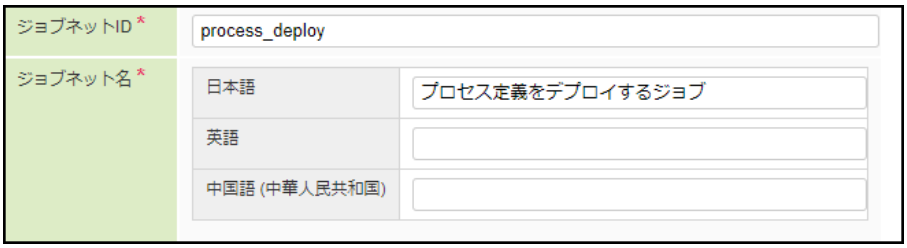
ロジックフローを実行するジョブネットを作成します。

> **コラム**
>
> このチュートリアルで作成するジョブネットのサンプルを、以下のリンクからダウンロードできます。
> **job-scheduler.xml**
>
> このサンプルはジョブネットの「ジョブインポート」でインポートできます。
> サンプル「job-scheduler.xml」 をパブリックストレージの直下に配置し、「ジョブネット管理」画面から、「テナントマスタ」 - 「インポート」 - 「ジョブインポート」を選択し、ジョブネットの「即時実行」をクリックすることでインポートできます。

1. 「サイトマップ」→「テナント管理」→「ジョブ管理」→「ジョブネット設定」から、「ジョブネット管理」画面を表示します。
2. 「ジョブネット管理」画面、ツールバー内の「ジョブネット新規作成」をクリックします。
3. 「基本情報」を設定します。

   以下のように設定します。

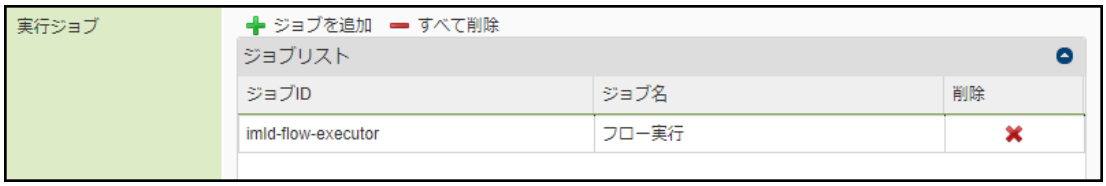
   - ジョブネットID：process_deploy
   - ジョブネット名：プロセス定義をデプロイするジョブ



図：「ジョブネット作成」 - 「基本情報」

4. 「実行時の情報」を設定します。
   1. 「実行ジョブ」 - 「ジョブを追加」をクリックし、「IM-LogicDesigner」 - 「フロー実行」を選択します。

      フロー実行ジョブについての詳細は「ジョブ・ジョブネットリファレンス」-「フロー実行」を参照してください。



図：「ジョブネット作成」 - 「実行時の情報」 - 「実行ジョブ」

> **コラム**
>
> フロー実行ジョブについての詳細は「ジョブ・ジョブネットリファレンス」-「フロー実行」を参照してください。

2. 「実行パラメータ」 - 「パラメータを追加」をクリックし、以下のように設定します。

キー	値
deploymentName	process_auto_deploy
flow_id	process_auto_deploy
processResourceNameList	tutorial_process1,tutorial_process2,tutorial_process3
projectId	processAutoDeploy
zipFilePath	process_auto_deploy.zip

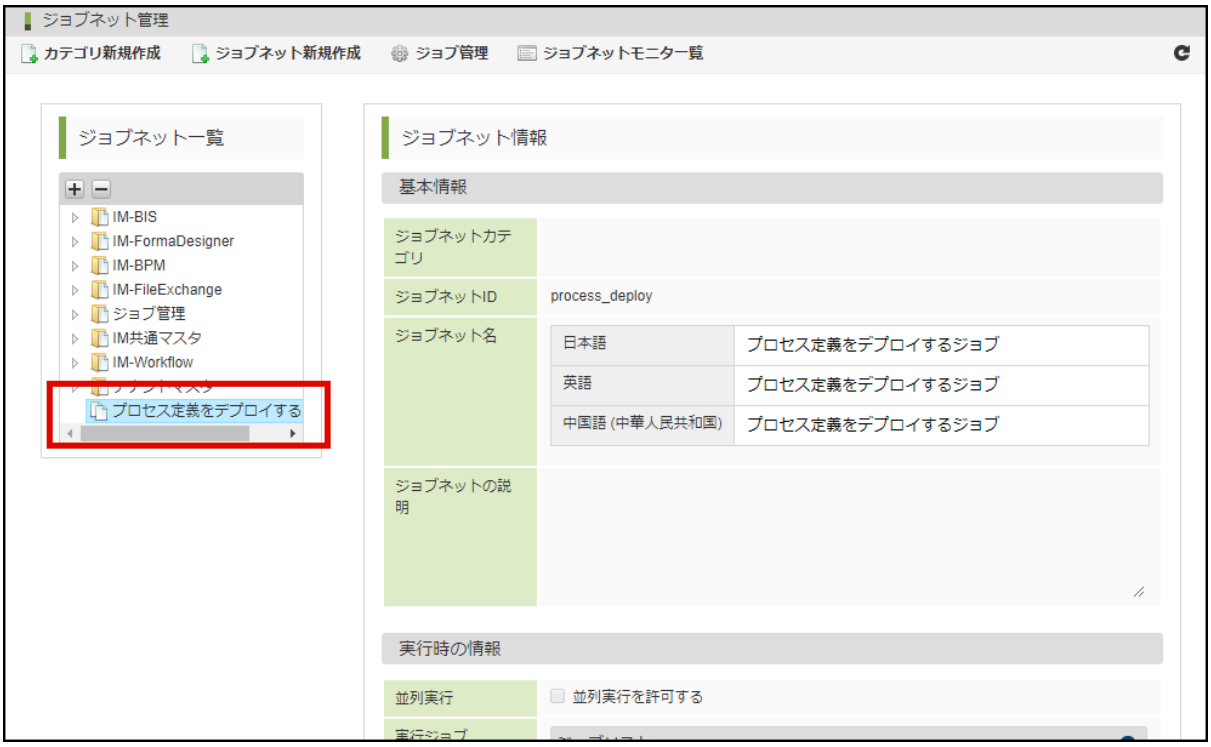
図：「ジョブネット作成」 - 「実行時の情報」 - 「実行パラメータ」

5. 「トリガ設定」を設定します。
    1. 「トリガ設定」のセレクトボックスから「日時指定」を選択し、「新規登録」を選択します。
    2. ジョブを実行する日時を指定します。
    3. 設定したトリガの「有効」のチェックボックスをオンにします。

実行結果を確認する

このチュートリアルで作成したジョブネットを実行し、実行結果の確認を行います。
ここでは、実行結果を確認するためにジョブネットの即時実行を行います。

1. 「サイトマップ」→「テナント管理」→「ジョブ管理」→「ジョブネット設定」から、「ジョブネット管理」画面を表示します。
2. 「ジョブネット管理」画面、左側のツリーから「プロセス定義をデプロイするジョブ」を選択します。



図：「ジョブネット管理」 - 「プロセス定義をデプロイするジョブ」

3. 「即時実行」をクリックします。

図：「ジョブネット管理」 - 「プロセス定義をデプロイするジョブ」- 「即時実行」

4. 「ジョブネット管理」画面、ツールバー内の「ジョブネットモニター覧」をクリックします。

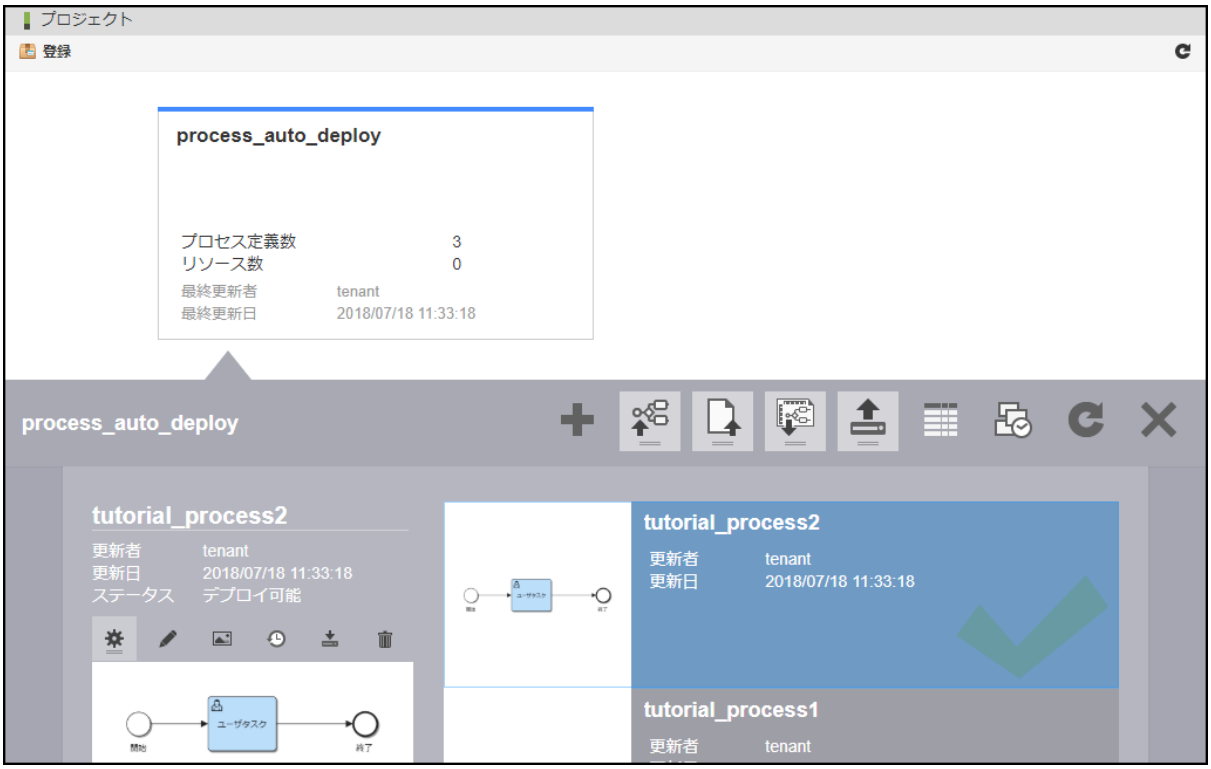
プロセス定義をデプロイするジョブ が正常に実行されたことを確認します。

図：「ジョブネットモニター覧」 プロセス定義をデプロイするジョブ

5. 「サイトマップ」→「BPM」→「プロセスデザイナ」から、「プロセスデザイナ」画面を表示します。

サンプルデータを使用した場合は、プロセスデザイナのプロジェクト画面に以下のプロジェクトがインポートされていることを確認します。

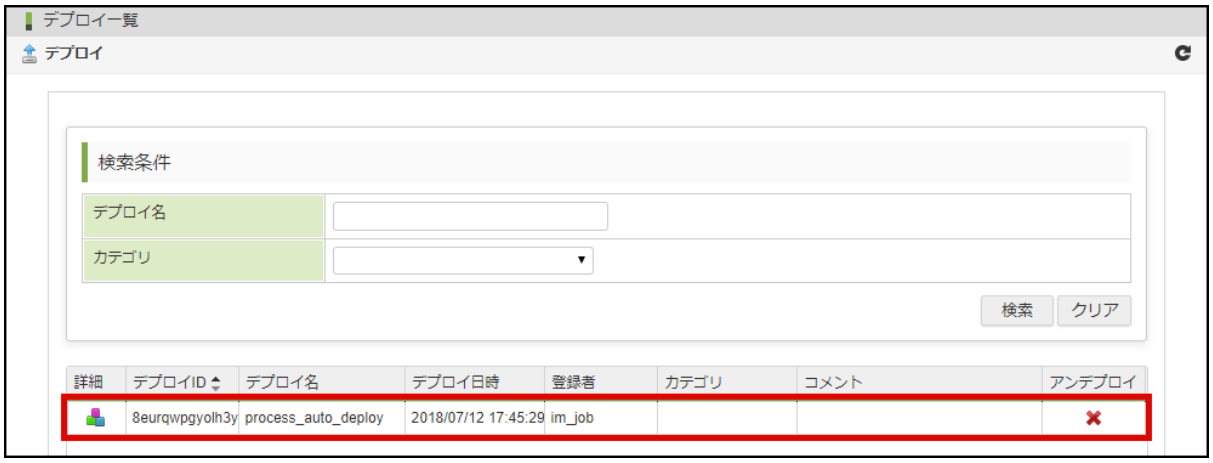
- プロジェクト名 : process_auto_deploy
- プロセス定義名 :
  - tutorial_process1
  - tutorial_process2
  - tutorial_process3



図：「プロセスデザイナ」 process_auto_deploy

6. 「サイトマップ」→「BPM」→「デプロイ一覧」から、「デプロイ一覧」画面を表示します。

   1. 「デプロイ名」 process_auto_deploy がプロセス定義として、デプロイされていることを確認します。

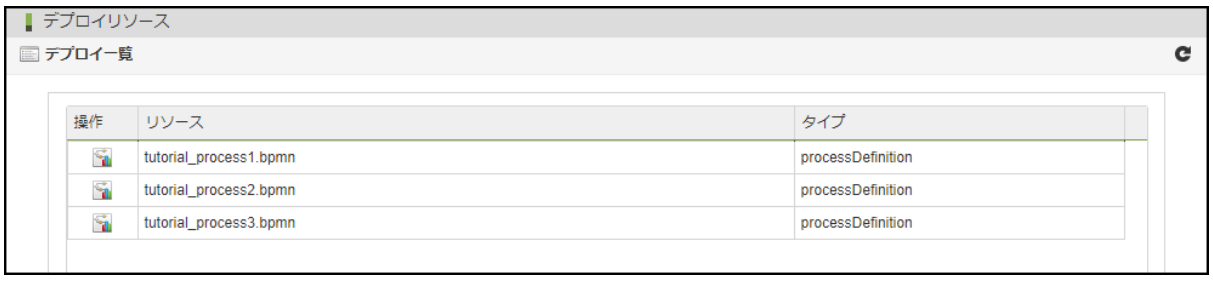


図：「デプロイ一覧」 process_auto_deploy

   2. 「デプロイ一覧」画面、一覧から「デプロイ名」 process_auto_deploy の「詳細」をクリックします。

      以下の「リソース」のファイルがデプロイ資材として展開されていることを確認します。

      - tutorial_process1.bpmn
      - tutorial_process2.bpmn
      - tutorial_process3.bpmn



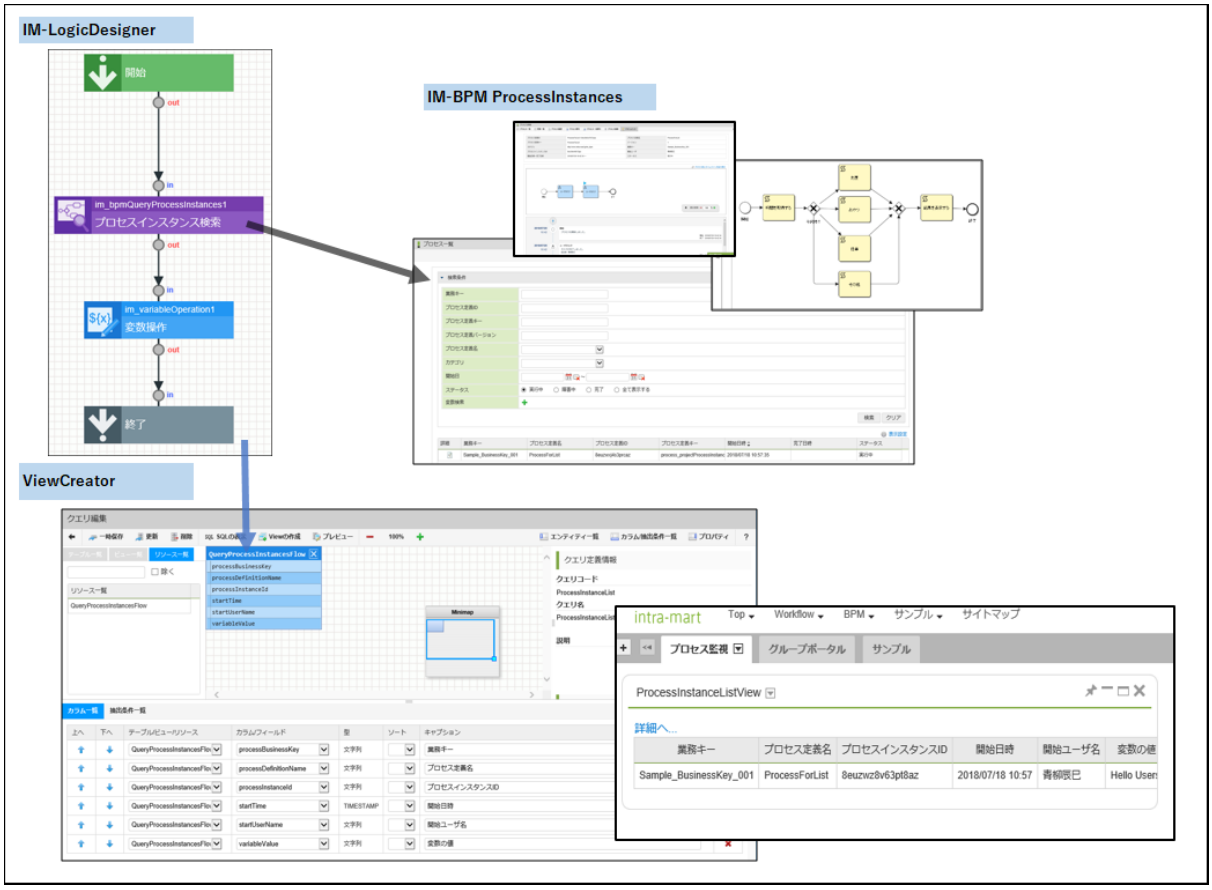
図：「デプロイリソース」

アドオン画面の作成

アドオン画面を作成する方法について説明します。
このチュートリアルは「*事前準備*」が完了していることを前提としています。

アドオンのプロセス一覧画面の作成

このチュートリアルでは、IM-LogicDesignerのロジックフローとViewCreatorを使用し、カスタマイズしたプロセスインスタンスの一覧画面を作成する方法を解説します。



図：作成物の構成

> **コラム**
>
> IM-LogicDesignerとは、intra-mart Accel Platform上でビジネスロジックを簡単に作成できるアプリケーションです。
> IM-LogicDesignerの詳細については、「IM-LogicDesigner仕様書」を参照してください。

> **コラム**
>
> ViewCreatorはintra-martのWeb画面上から、データベースなどのデータを使用して、表やグラフを簡単に作成できるツールです。
> ViewCreatorの詳細については、「ViewCreator 管理者操作ガイド」を参照してください。

コラム

このチュートリアルで使用する「デプロイメント情報」および、作成する「ロジックフロー定義」、「クエリ」、「データ参照」を以下のリンクからダウンロードできます。

- 「デプロイメント情報」
  im_bpm_process_instance_list.zip
- IM-LogicDesigner「ロジックフロー定義」
  im_logicdesigner-data-process-instance-list-view.zip
- ViewCreator「クエリ」
  ProcessInstanceList.xml
- ViewCreator「データ参照」
  im_bpm_process_instance_list_view.xml

各種インポート方法については、以下のリンクを参照してください。

- 「デプロイメント情報」：「IM-BPM ユーザ操作ガイド」-「インポート」
- IM-LogicDesignerの「ロジックフロー定義」：「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」
- ViewCreatorの「クエリ」：「ViewCreator 管理者操作ガイド」-「クエリ一覧画面」
- ViewCreatorの「データ参照」：「ViewCreator 管理者操作ガイド」-「データ参照の種類の選択」

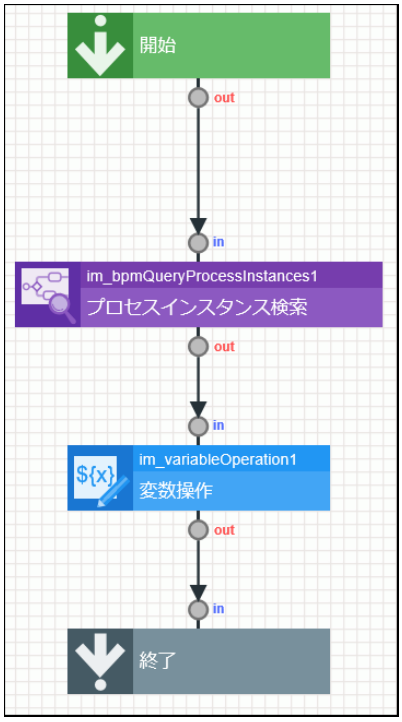
注意

このチュートリアルの「ロジックフロー定義」は IM-BPM for Accel Platform 2018 Summer(Tiffany) 以降のバージョンで動作します。

- プロセスインスタンスのデータを取得するIM-LogicDesignerのロジックフローを作成する
- IM-LogicDesignerのロジックフローと連携したViewCreatorのクエリ・データ参照を作成する
- 作成したプロセスインスタンス一覧画面を確認する

プロセスインスタンスのデータを取得するIM-LogicDesignerのロジックフローを作成する

プロセスインスタンスの検索を行い、検索結果を出力項目`result<object[]>`として返却するIM-LogicDesignerのロジックフローを作成します。



図：完成イメージ

1. 「サイトマップ」→「LogicDesigner」→「フロー定義一覧」から、「ロジックフロー定義一覧」画面を表示します。
2. 「フロー定義一覧」画面、ツールバー内の「新規作成」をクリックします。
3. 「ロジックフロー定義編集」画面上部、ヘッダ内の「定数設定」をクリックします。

   定数値を以下のように設定します。

定数ID	定数値
TRUE	true

定数ID	定数値
processBusinessKeyLike	%BusinessKey%

図：定数設定

4. 「ロジックフロー定義編集」画面上部、ヘッダ内の「変数設定」をクリックします。

   変数に以下のパラメータを追加します。

   1. 「object」をクリックし、キー名をeditedQueryResultとします。

   2. 「配列型にする」にチェックを入れます。

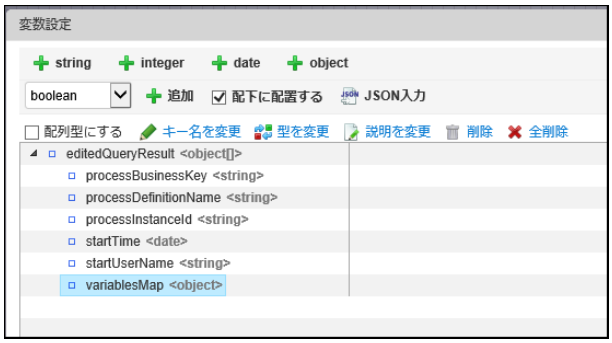
キー名	型
editedQueryResult	object[]

      図：editedQueryResult<object[]>の設定

   3. 「配下に配置する」にチェックを入れ、editedQueryResult<object[]>パラメータの配下に以下の項目を設定します。

キー名	型
processBusinessKey	string
processDefinitionName	string
processInstanceId	string
startTime	date
startUserName	string
variablesMap	object

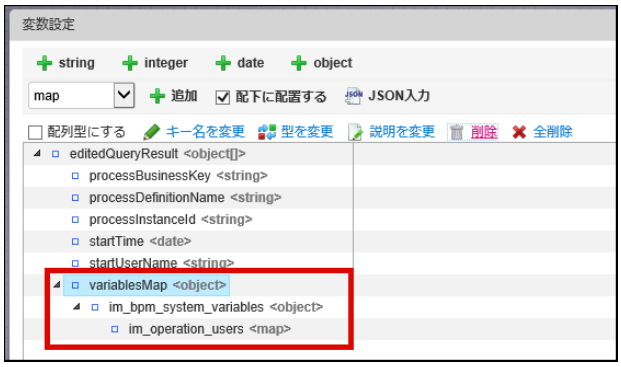
      図：editedQueryResult<object[]>の配下の項目設定

   4. editedQueryResult<object[]>配下のvariablesMap<object>を選択し、「配下に配置する」にチェックを入れ、以下の項目を設定します。

キー名	型
im_bpm_system_variables	object

   5. variablesMap<object>配下のim_bpm_system_variables<object>を選択し、「配下に配置する」にチェックを入れ、以下の項目を設定します。

キー名	型
im_operation_users	map



図：variablesMap<object>の配下の項目設定

5. 「ロジックフロー定義編集」画面上部、ヘッダ内の「入出力設定」をクリックします。

「出力」に以下のパラメータを追加します。

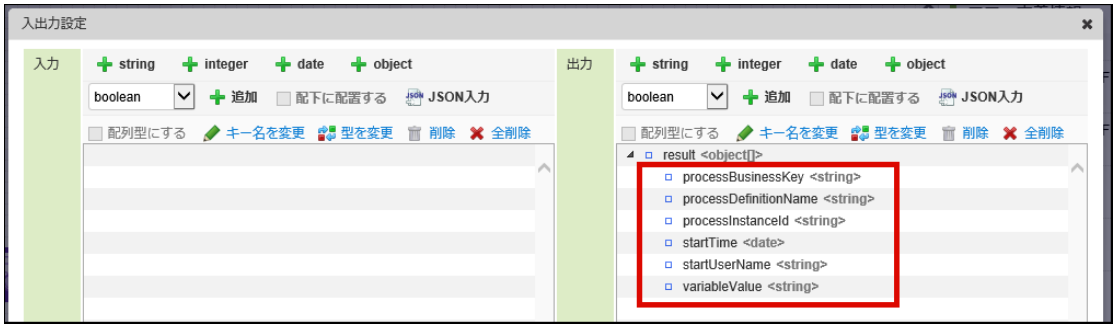
1. 「object」をクリックし、キー名をresultとします。
2. 「配列型にする」にチェックを入れます。

キー名	型
result	object[]

図：result<object[]>の設定

3. 「配下に配置する」にチェックを入れ、result<object[]>パラメータの配下に以下のキーを設定します。

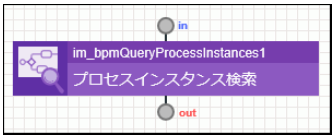
キー名	型
processBusinessKey	string
processDefinitionName	string
processInstanceId	string
startTime	date
startUserName	string
variableValue	string



図：result<object[]>の配下のキー設定

> **コラム**
>
> キーresult<object[]>は、ViewCreatorと連携する際には必須のパラメータです。
>
> ロジックフローの出力結果をリソースとして利用する際の詳細は「ViewCreator 管理者操作ガイド」-「ViewCreatorで利用可能なロジックフローの出力設定」を参照してください。

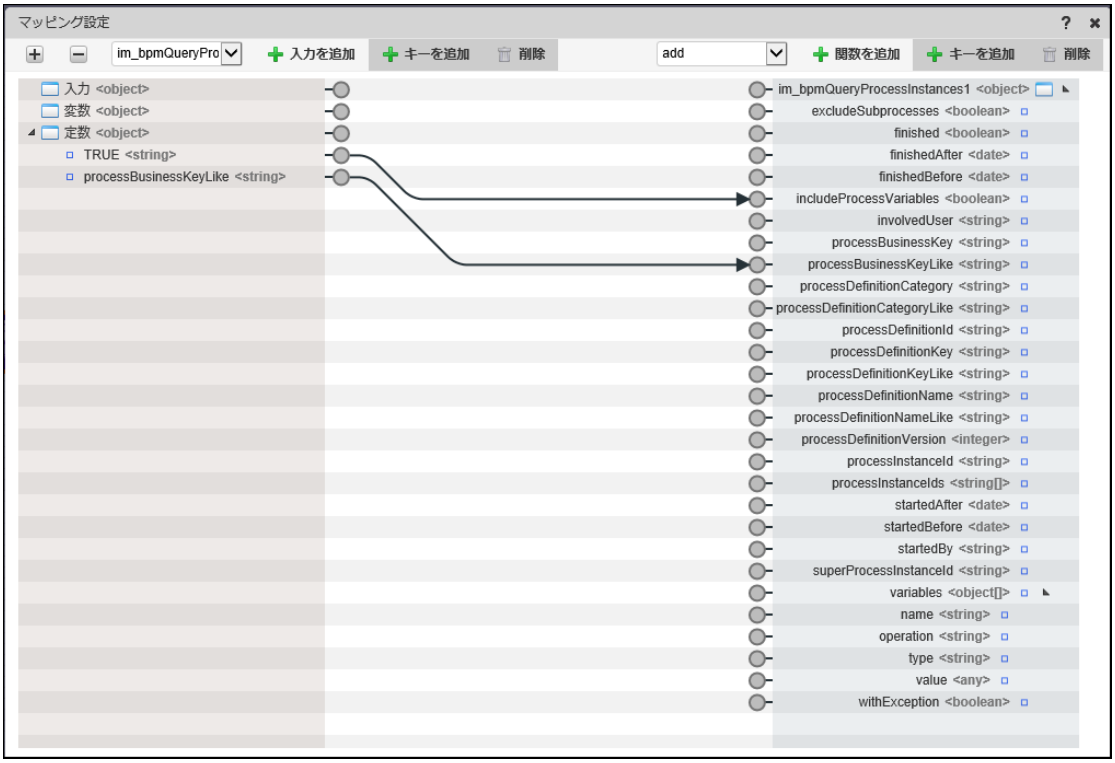
6. パレット内の「IM-BPM」の一覧から「プロセスインスタンス検索」を選び、フロー編集画面上に追加します。



図：「プロセスインスタンス検索」タスク

7. 「プロセスインスタンス検索」のプロパティ項目「マッピング設定」を開き、定数項目から「出力ペイン」項目のim_bpmQueryProcessInstances1<object>配下の各キーに対し、以下のようにマッピングを行います。

入力（始点）	出力（終点）
定数<object> - TRUE<string>	im_bpmQueryProcessInstances1<object> - includeProcessVariables<boolean>
定数<object> - processBusinessKeyLike<string>	im_bpmQueryProcessInstances1<object> - processBusinessKeyLike<string>



図：「プロセスインスタンス検索」タスクへのマッピング設定

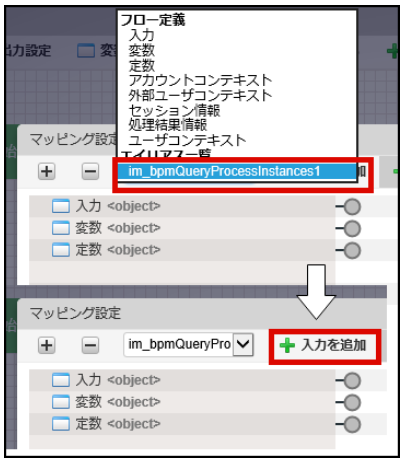
> **コラム**
>
> 上記の設定では、プロセスインスタンスを「業務キー」で部分一致検索し、返却される項目へ「プロセス変数を含む」オプションを設定しています。
>
> 「プロセスインスタンス検索」に関するタスクの詳細は「IM-LogicDesigner仕様書」-「プロセスインスタンス検索」を参照してください。

8. パレット内の「基本」の一覧から「変数操作」を選び、フロー編集画面上に追加します。

図：「変数操作」タスク

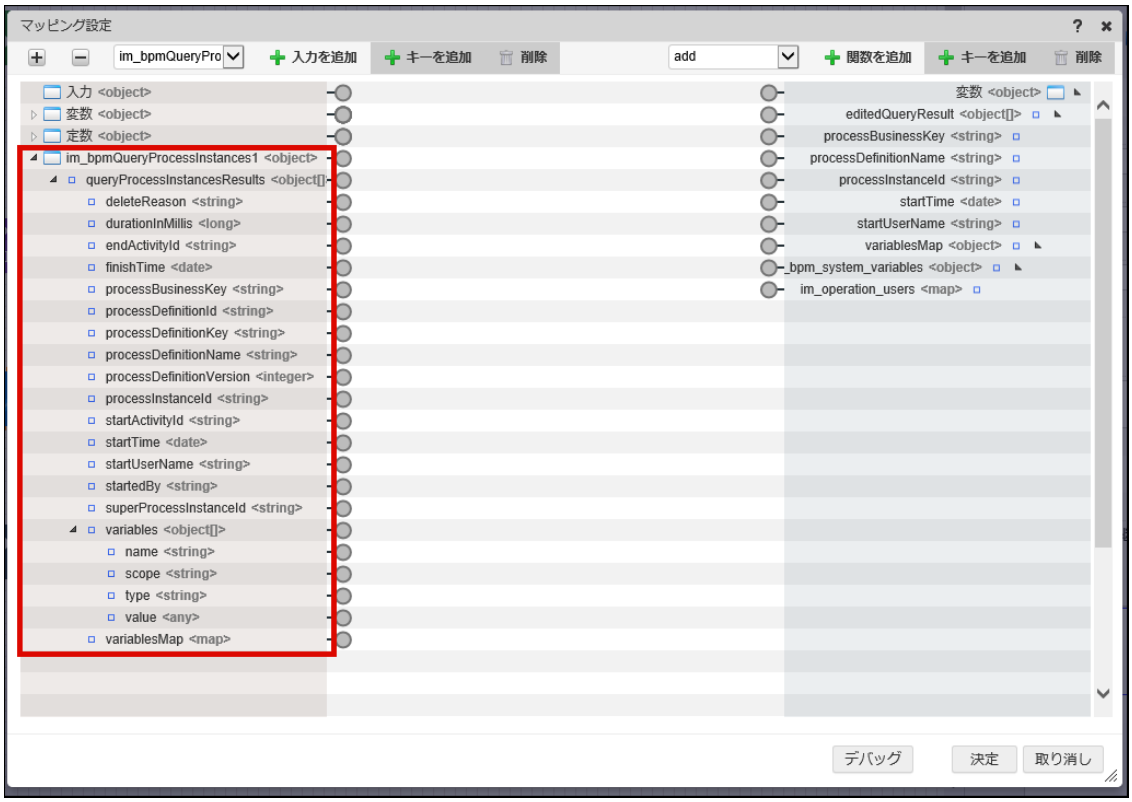
9. 「変数操作」のプロパティ項目「マッピング設定」を開き、以下のようにマッピングを行います。

1. 「マッピング設定」画面上部、ヘッダ内の左上に位置するセレクトボックスからim_bpmQueryProcessInstances1を選択し、「入力を追加」をクリックします。



図：im_bpmQueryProcessInstances1を選択

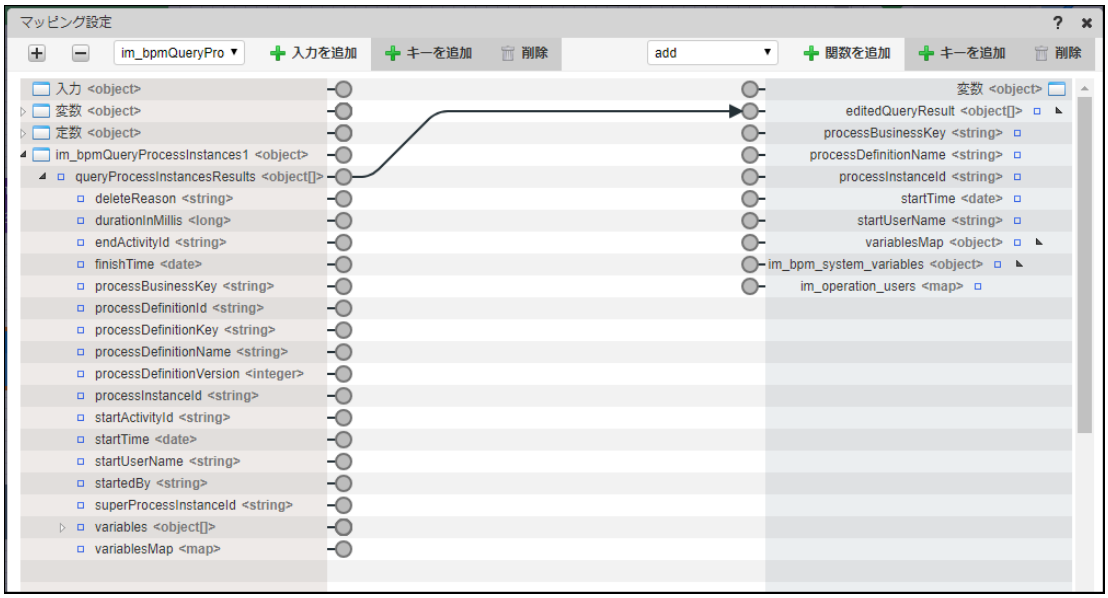
以下のように、「入力ペイン」項目にim_bpmQueryProcessInstances1<object>が追加されます。



図：im_bpmQueryProcessInstances1<object>の追加

2. 「入力ペイン」項目のim_bpmQueryProcessInstances1<object>配下のqueryProcessInstancesResults<object[]>配下の各キーより変数editedQueryResult<object[]>の各キーに対して、以下のようにマッピングを行います。

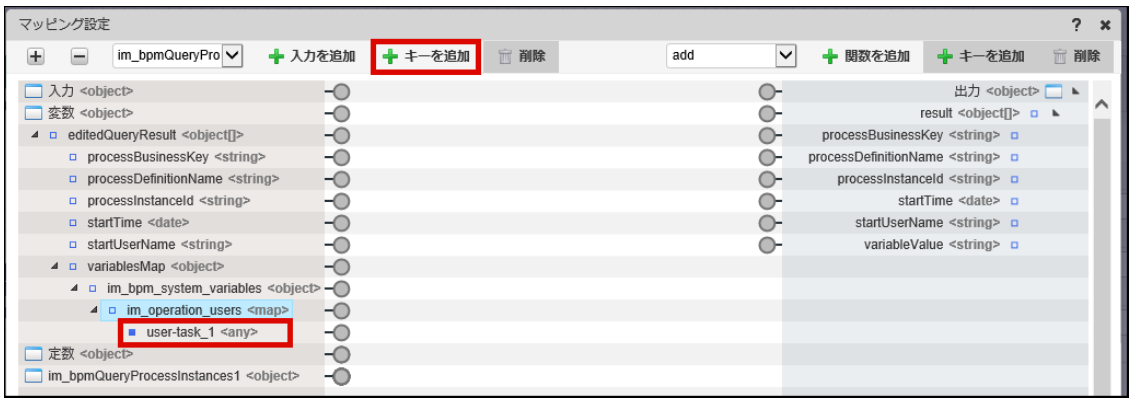
入力（始点）	出力（終点）
im_bpmQueryProcessInstances1<object> - queryProcessInstancesResults<object[]>	変数<object> - editedQueryResult<object[]>



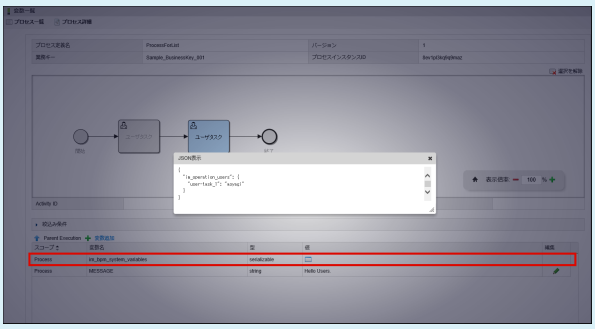
図：editedQueryResult<object[]>へのマッピング設定

10. 「終了」のプロパティ項目「マッピング設定」を開き、以下のようにマッピングを行います。

    1. 「入力ペイン」項目の「変数」内のeditedQueryResult<object[]> - variablesMap<object> - im_bpm_system_variables<object> - im_operation_users<map>を選択し、「マッピング設定」画面上部、ヘッダ内の「キーを追加」をクリックしてキーの追加を行い、追加されたキーの名称にuser-task_1と入力します。

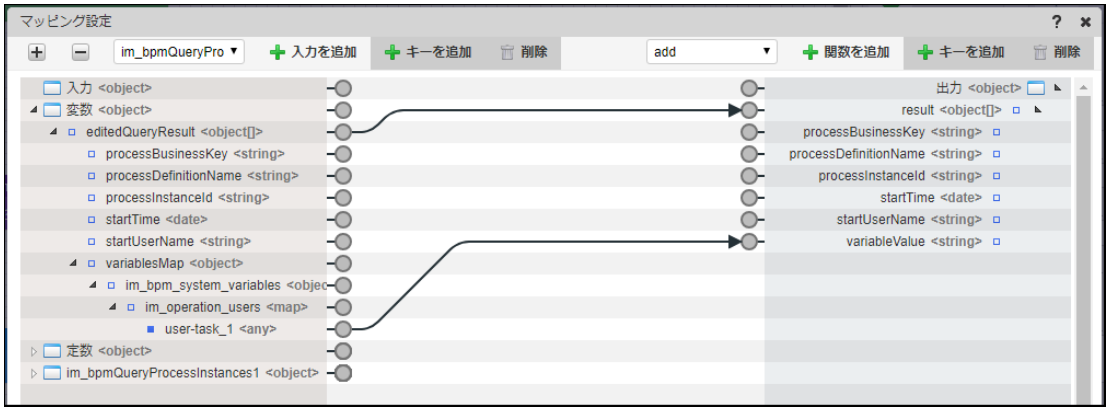


図：im_operation_users<map>へキーuser-task_1を追加

> **コラム**
>
> 上記の設定は、プロセスインスタンスが内部的に保持しているMap型の変数im_operation_usersから、タスク「user-task_1」の処理を行ったユーザのユーザコードを取得するための設定です。
>
> プロセスインスタンスが保持している変数については、以下のように標準機能の「変数一覧」画面でも確認できます。
>
> 
>
> 図：「変数一覧」画面
>
> 変数一覧画面についての詳細は「IM-BPM ユーザ操作ガイド」 - 「プロセスインスタンスの変数を確認する」を参照してください。

    2. 「入力ペイン」項目の「変数」内のeditedQueryResult<object[]>配下の各キーより、出力項目result<object[]>の配下の各キーに対して以下のようにマッピングを行います。

入力（始点）	出力（終点）
変数<object> - editedQueryResult<object[]>	出力<object> - result<object[]>
... - im_operation_users<map> - user-task_1<any>	出力<object> - result<object[]> - variableValue<string>

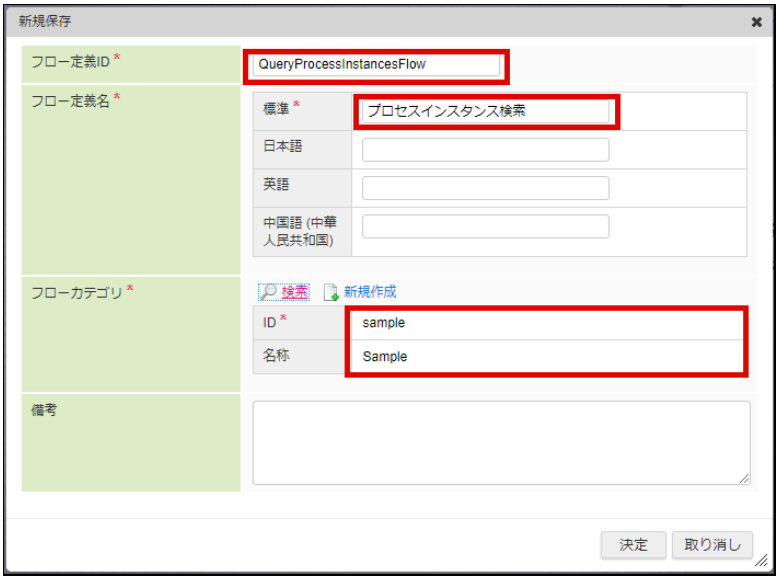


図：result<object[]>へのマッピング設定

11. 「開始」→「プロセスインスタンス検索」→「変数操作」→「終了」の順にタスクを接続します。
12. 「新規保存」をクリックし、以下を入力し、「決定」をクリックします。

<画面項目>

項目	設定値
フロー定義ID	QueryProcessInstancesFlow
フロー定義名（標準）	プロセスインスタンス検索
フローカテゴリ	Sample



図：新規保存ダイアログ

> **コラム**
>
> マッピング元とマッピング先で配下に同名の値を持っている場合、IM-LogicDesignerは自動でその値をマッピングします。
>
> 階層化された値のマッピングの詳細については、「IM-LogicDesigner チュートリアルガイド」-「階層化された入力値・出力値の定義（Object型の定義）」を参照してください。

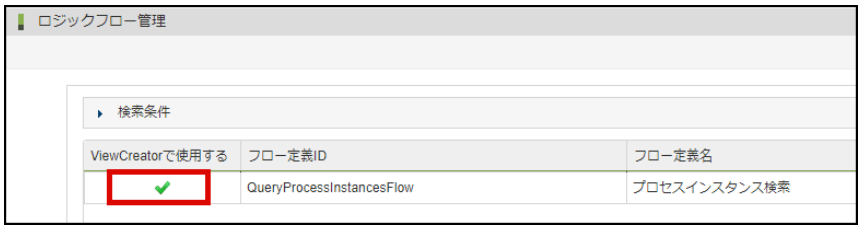
IM-LogicDesignerのロジックフローと連携したViewCreatorのクエリ・データ参照を作成する

IM-LogicDesignerのロジックフローからデータを取得し、ViewCreatorで表示するためのクエリ・データ参照の設定を作成します。

1. 外部データソース定義
    1. 「サイトマップ」→「ViewCreator」→「外部データソース連携」→「ロジックフロー管理」をクリックし「ロジックフロー管理」画面を表示し

ます。

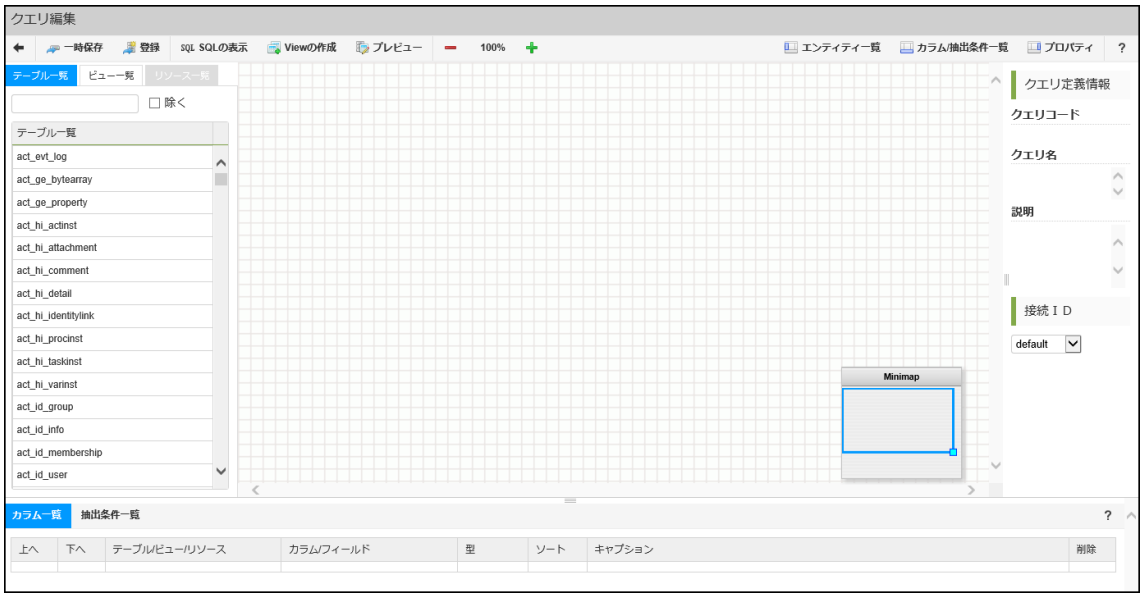
2. 「ロジックフロー管理」画面のフロー定義ID一覧に表示されているフロー定義IDQueryProcessInstancesFlowの「ViewCreatorで使用する」にチェックを入れます。



図：「ViewCreatorで使用する」にチェック

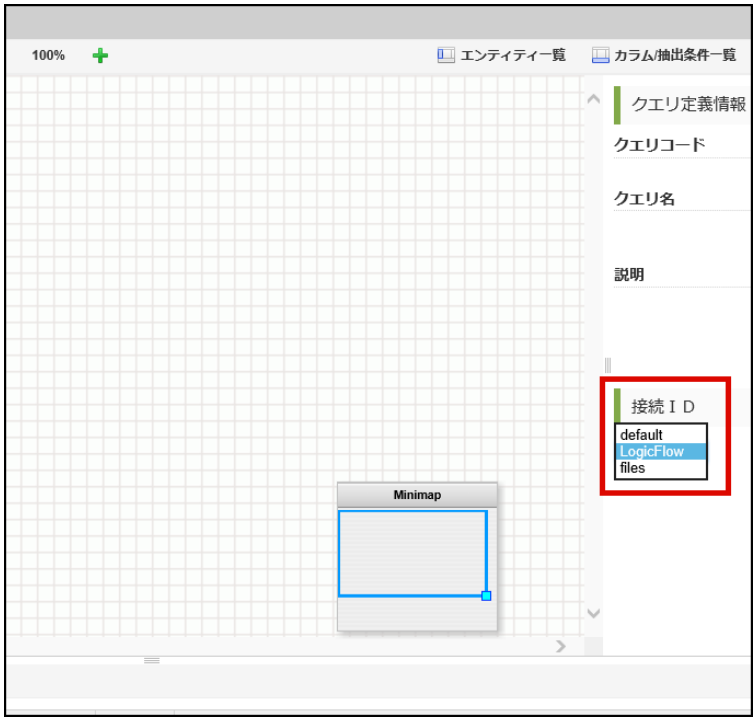
2. クエリの作成

1. 「サイトマップ」→「ViewCreator」→「クエリー覧」から、ViewCreatorの「クエリー覧」画面を表示します。
2. クエリー覧画面の「新規」をクリックして「クエリ編集」画面を表示します。



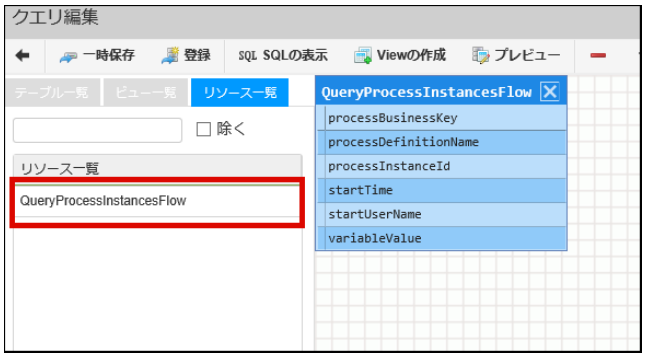
図：クエリの新規作成

3. 「接続ID」のプルダウンよりLogicFlowを選択します。



図：接続IDの選択

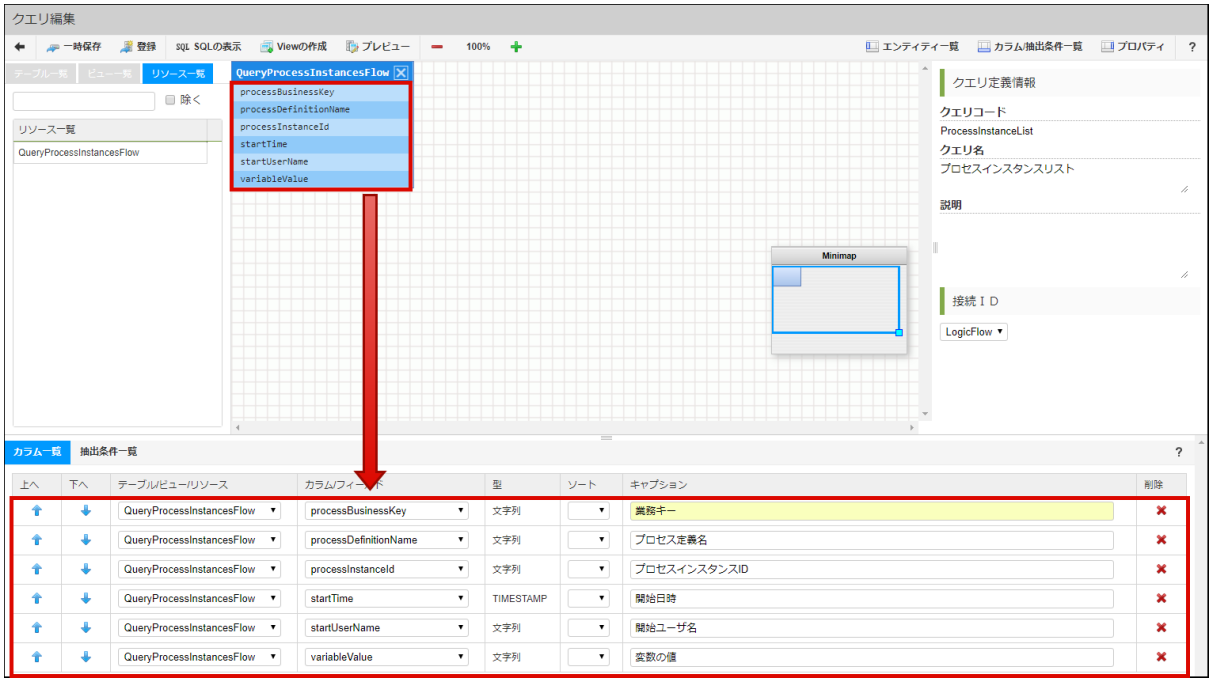
4. 「リソース一覧」よりQueryProcessInstancesFlowを選択します。



図：リソース一覧

5. デザイナ部分に表示されたリソースQueryProcessInstancesFlowより、以下の「フィールド」をダブルクリックし、クエリの「カラム」として選択し、選択した「カラム」に「キャプション」を設定します。

フィールド名	キャプション
processBusinessKey	業務キー
processDefinitionName	プロセス定義名
processInstanceId	プロセスインスタンスID
startTime	開始日時
startUserName	開始ユーザ名
variableValue	変数の値

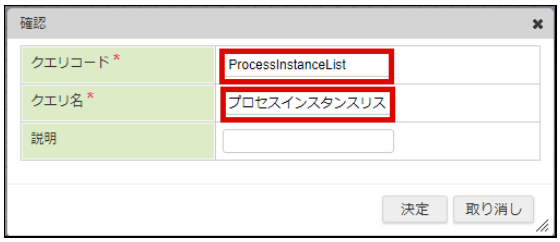


図：カラムの選択とキャプションの設定

6. 「クエリ編集」画面上部の「登録」ボタンをクリックし、以下の値を入力し、作成したクエリを登録します。

<画面項目>

項目	設定値
クエリコード	ProcessInstanceList
クエリ名	プロセスインスタンスリスト



図：登録確認ダイアログ

3. データ参照の作成

    1. 「サイトマップ」→「ViewCreator」→「クエリー覧」から、ViewCreatorの「クエリー覧」画面を表示します。

    2. プロセスインスタンスリストの「データ参照作成」欄のリスト集計作成（）をクリックし、「リスト集計の作成」画面を表示します。

図：リスト集計作成

    3. 「リスト集計の作成」画面の入力項目に以下の値を入力します。

    <画面項目>

項目	設定値
データ参照名	プロセスインスタンスリスト参照
参照権 - 認証済みユーザ	チェック

図：データ参照名

図：参照権 - 認証済みユーザ

    4. カラム一覧のカラムプロセス定義名（processDefinitionName）の「ソート順」プルダウンより「昇順」を、カラム開始日時（startTime）の「ソート順」プルダウンより「降順」をそれぞれ選択します。

図：ソートの設定

5. カラム一覧のカラム開始日時（startTime）の「フォーマット」にyyyy/MM/dd HH:mm:ssを設定します。

図：日付フォーマットの設定

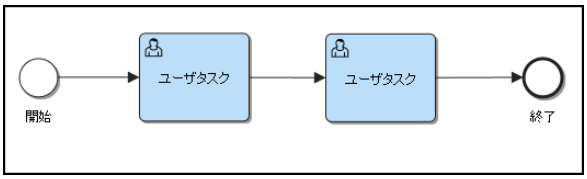
> **コラム**
>
> 上記の設定では、プロセス定義名（processDefinitionName）カラム（昇順）が第１ソートキー、開始日時（startTime）カラム（降順）が第２ソートキーです。
>
> 日付フォーマットや、「リスト集計の作成」についてのその他の設定の詳細は「ViewCreator 管理者操作ガイド」-「ViewCreator 管理者操作ガイド - リスト集計の作成」を参照してください。

6. 「登録して一覧へ戻る」をクリックします。

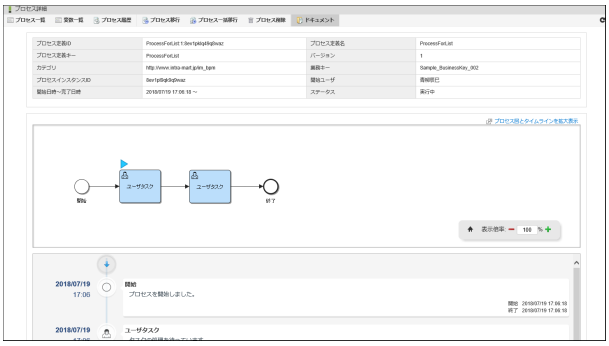
作成したプロセスインスタンス一覧画面を確認する

検索対象となるプロセスインスタンスを作成し、プロセスインスタンス一覧画面を確認します。

1. 検索対象となるプロセスインスタンスを作成するために、デプロイメント情報をインポートします。



図：インポートされるプロセス定義



図：実行中のプロセスインスタンスの詳細画面

コラム

デプロイされたプロセス定義を開始することにより、プロセスインスタンスが作成されます。

プロセスインスタンスの確認については、「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスを確認する」を参照してください。

1. 「サイトマップ」→「BPM」→「インポート」画面を表示し、「インポートファイル」項目に、デプロイメント情報im_bpm_process_instance_list.zipを設定し、「実行」ボタンをクリックします。

図：インポート画面

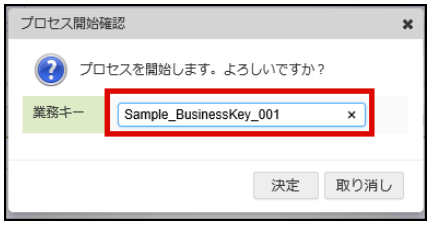
2. 「サイトマップ」→「BPM」→「デプロイ一覧」画面を表示し、デプロイ名DeployProcessForListが存在することを確認します。

図：デプロイ一覧画面

3. 「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示し、プロセス定義名ProcessForListのプロセス開始（  ）をクリックし、「プロセス開始確認」ダイアログの「業務キー」に以下の値を入力しプロセスを開始してください。

合計で3件のプロセスを開始します。

業務キー
Sample_BusinessKey_001
Sample_BusinessKey_002
Sample_ManagementProcessKey_001



図：プロセス開始確認ダイアログ

4. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示し、グループタスクにプロセス定義名ProcessForListのプロセスのユーザタスクが3件存在することを確認します。

図：タスク一覧画面

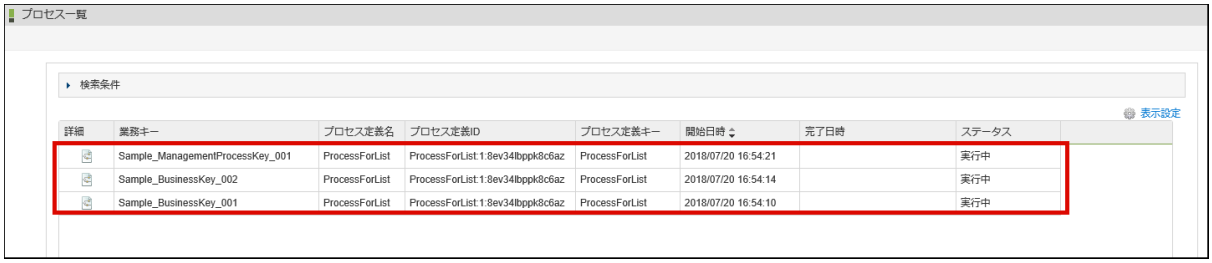
5. グループタスクの「業務キー」に Sample_BusinessKey_001 が設定されているユーザタスクの担当にする（  ）をクリックし、タスクの割り当てを行います。

図：グループタスク一覧

6. 「業務キー」に Sample_BusinessKey_001 が設定されているユーザタスクが個人タスクに移動したことを確認し、個人タスクの処理（  ）をクリックし、処理確認ダイアログで「決定」をクリックし、タスクを処理します。

図：個人タスク一覧

7. 「サイトマップ」→「BPM」→「プロセス一覧」画面を表示し、プロセス定義名ProcessForListのプロセスが3件存在することを確認します。



図：プロセス一覧画面

> **注意**
>
> 本チュートリアルではプロセスの開始を行うユーザは、「IM-BPM 管理者（im_bpm_manager）」ロールを持つユーザであることを前提としています。
> プロセス定義の開始権限の詳細については「IM-BPM 仕様書」 - 「プロセス定義の開始権限」を参照してください。

2. ViewCreatorで作成したプロセスインスタンスの一覧画面を確認します。

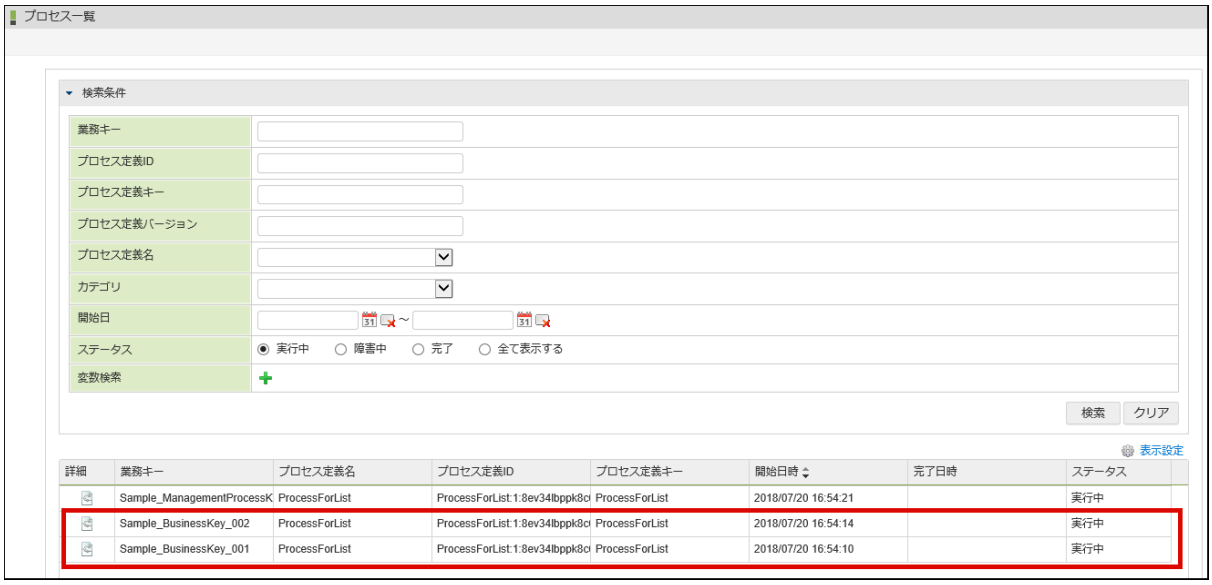
1. 「サイトマップ」→「ViewCreator」→「データ参照一覧」から、ViewCreatorの「データ参照一覧」画面を表示します。
2. 一覧に表示されているプロセスインスタンスリスト参照のリンクをクリックします。

図：データ参照へのリンク

3. 「業務キー」に文字列BusinessKeyを含むプロセスインスタンスのリストが表示されることを確認します。

図：データ参照-プロセスインスタンス一覧

4. 「サイトマップ」→「BPM」→「プロセス一覧」画面を表示し、プロセス定義名「ProcessForList」のプロセスが存在することを確認します。



図：標準機能のプロセス一覧画面

5. 「プロセス一覧」画面に表示されている「業務キー」、「プロセス定義名」、「開始日時」が作成したデータ参照のプロセスインスタンス一覧の「業務キー」、「プロセス定義名」、「開始日時」と等しいこと、「プロセス一覧」画面には存在しない「プロセスインスタンスID」、「開始ユーザ名」、「変数の値」が表示されていることを確認します。

> **コラム**
>
> 作成したデータ参照はポートレットとして登録することで、ポータル画面へ表示することが可能です。
>
> ポートレットとして使用する場合の詳細については「ViewCreator 管理者操作ガイド」-「ポータルとの連携」を参照してください。