Copyright © 2017 NTT DATA INTRAMART CORPORATION

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. プール、レーン
      - 4.2.3.1. プールとレーンを利用して作業の関係者を明確にする
    - 4.2.4. コールアクティビティ
      - 4.2.4.1. コールアクティビティを使用する
    - 4.2.5. パラレルゲートウェイ
      - 4.2.5.1. パラレルゲートウェイを使用する
    - 4.2.6. 排他ゲートウェイ
      - 4.2.6.1. 排他ゲートウェイを使用する
    - 4.2.7. IM-LogicDesignerタスク
      - 4.2.7.1. IM-LogicDesignerタスクを利用してユーザ情報を取得する
      - 4.2.7.2. 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
    - 4.2.8. 申請タスク
      - 4.2.8.1. 申請タスクを使用してワークフローと連携する
    - 4.2.9. 起票タスク
      - 4.2.9.1. 起票タスクを使用してワークフローと連携する
    - 4.2.10. IM-FormaDesigner for Accel Platform をユーザ入力フォームとして利用する
      - 4.2.10.1. 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
      - 4.2.10.2. 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

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*」

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 Designer 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。

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することができます。
> - IM-FormaDesignerと連携し、システムに必要となる画面と簡単に連携することができます。
> - 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することができます。
> - 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することができます。
>   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。

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 （基礎編には必要ありません。）
   - IM-BIS モジュール （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

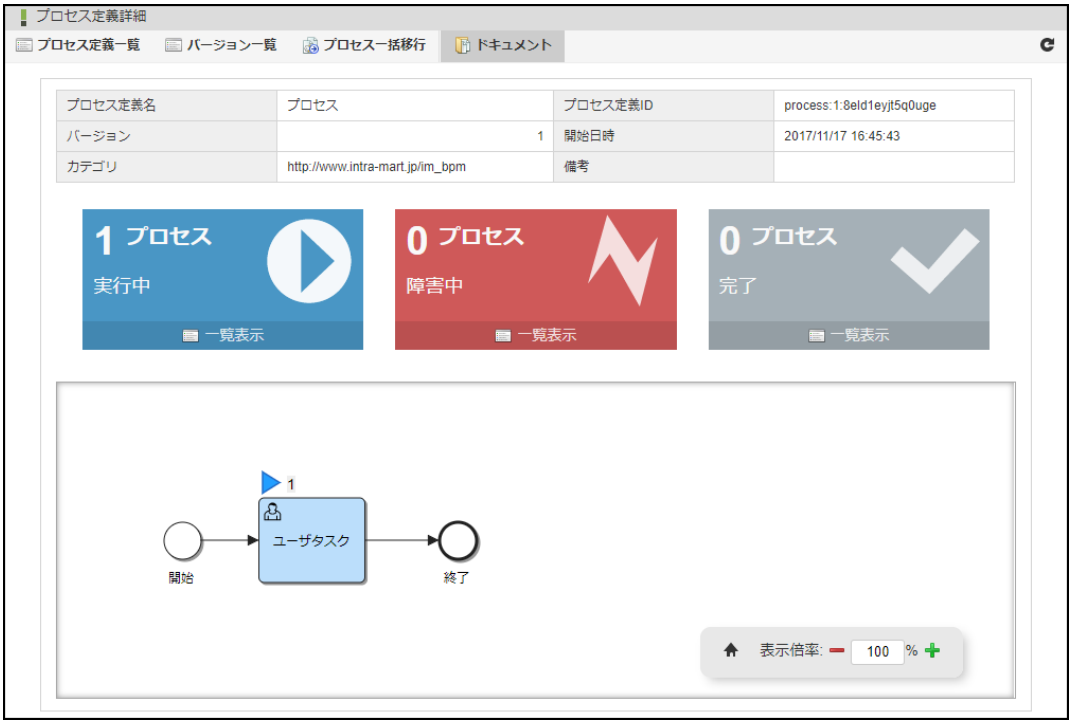
基礎編

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。

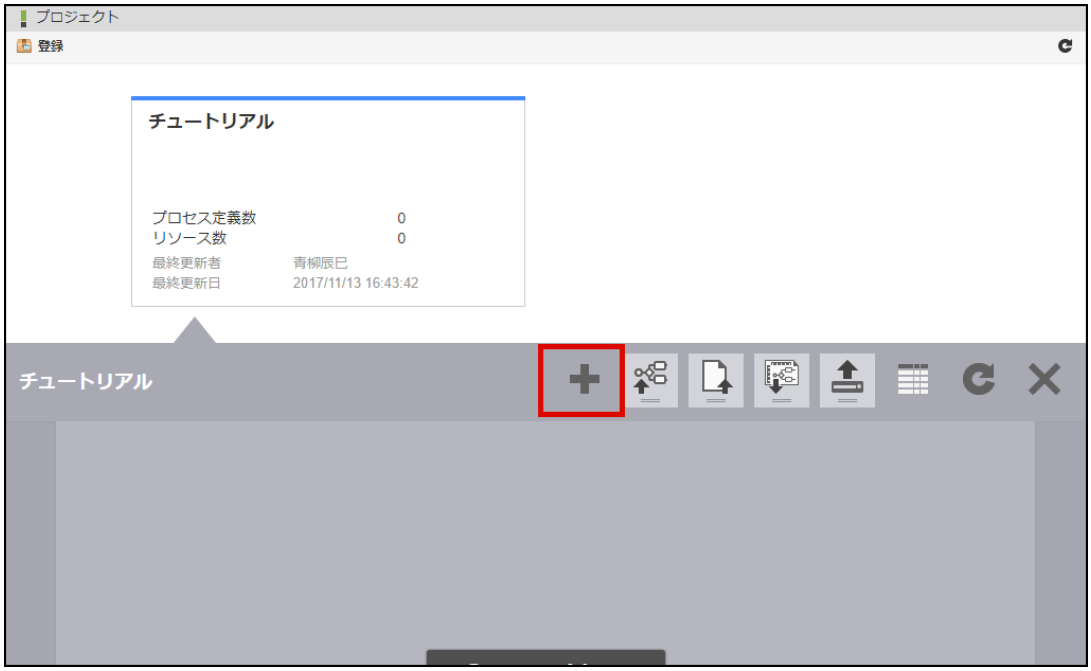


図：プロセス定義

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。

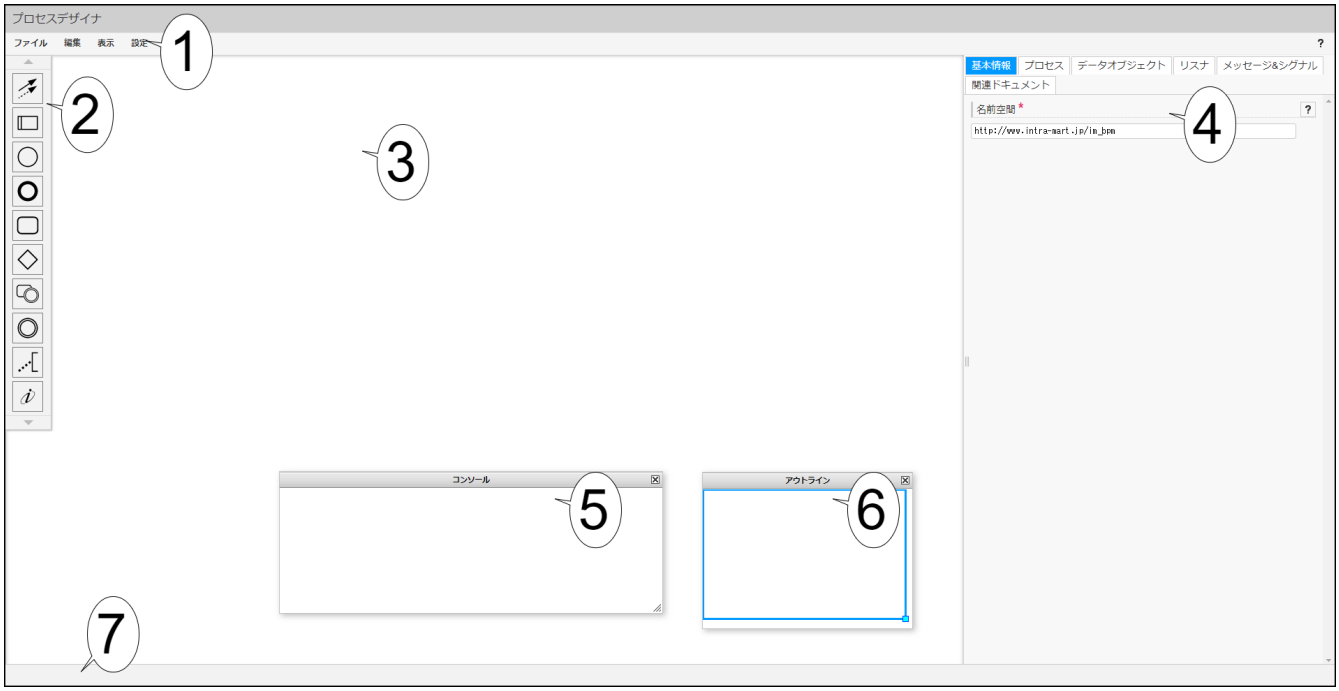
1. 「サイトマップ」→「BPM」→「プロセスデザイナ」をクリックします。
2. 新規プロジェクトを作成し、 をクリックします。



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 Designer 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

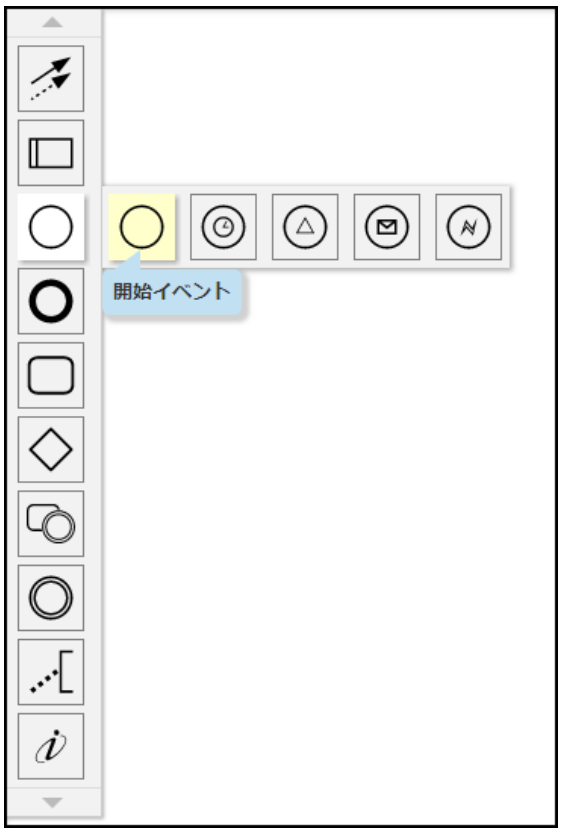
   プロセス全体を俯瞰表示する領域です。
   表示領域を移動したり、表示の拡大縮小を行ったりすることができます。

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。



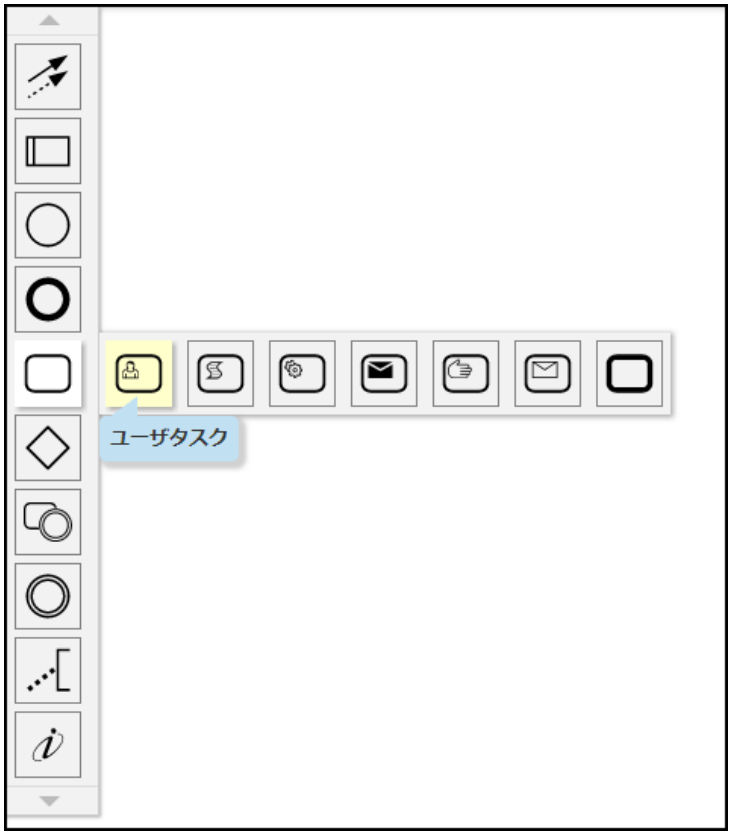
図：パレット - 開始イベント - 開始イベント

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。



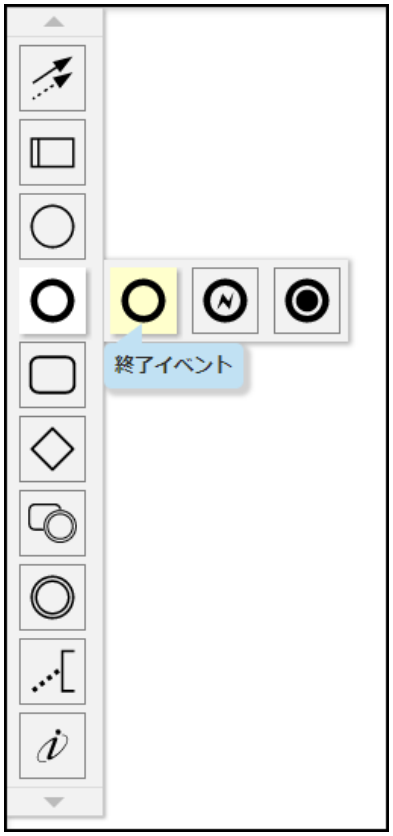
図：パレット - タスク - ユーザタスク

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。

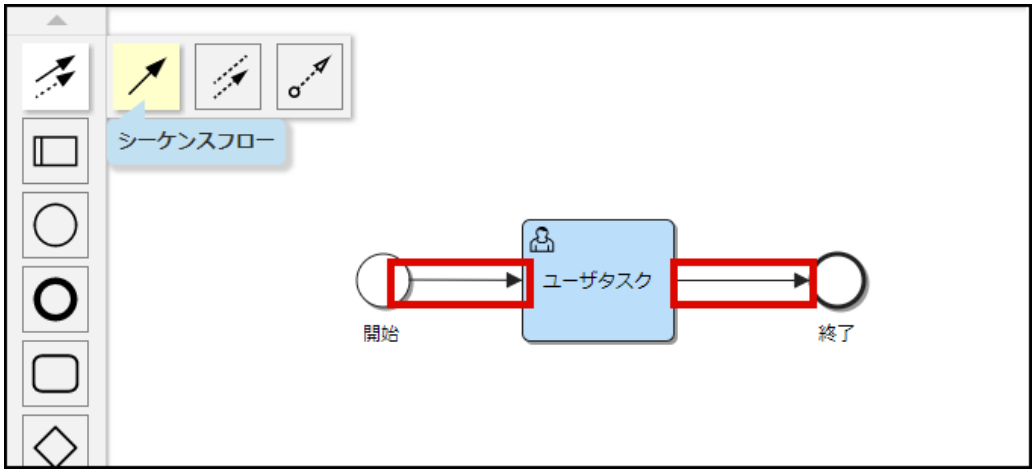
図：ユーザタスク - プロパティ - メインコンフィグ - 処理対象ユーザ

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。

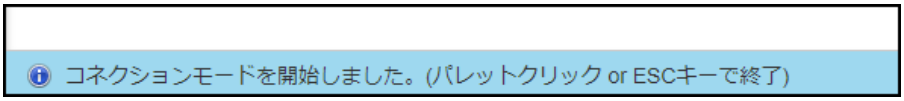


図：パレット - コネクション - シーケンスフロー

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。

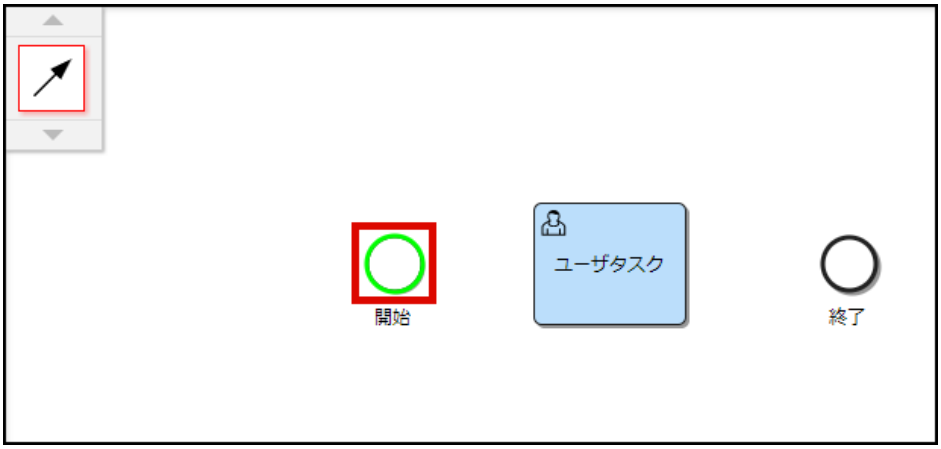
   コネクションモードが開始すると、フッタにメッセージが表示されます。



   図：フッタ

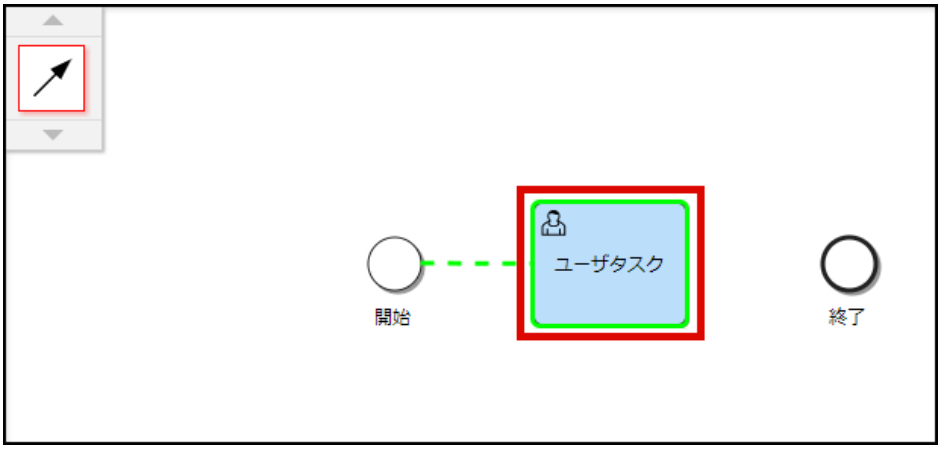
3. 「キャンバス」に配置された「開始イベント」をクリックまたはドラッグし、接続を開始します。

   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。

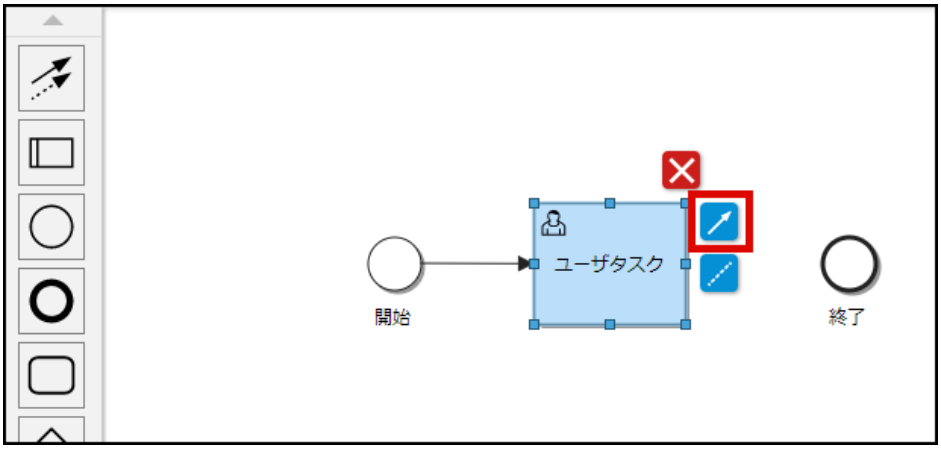


図：接続する

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。

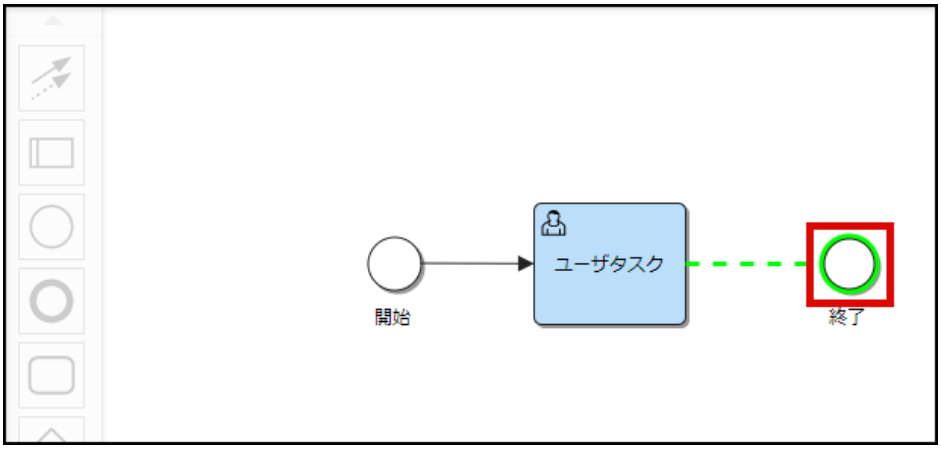
6. キャンバス上の「ユーザタスク」を選択します。

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。



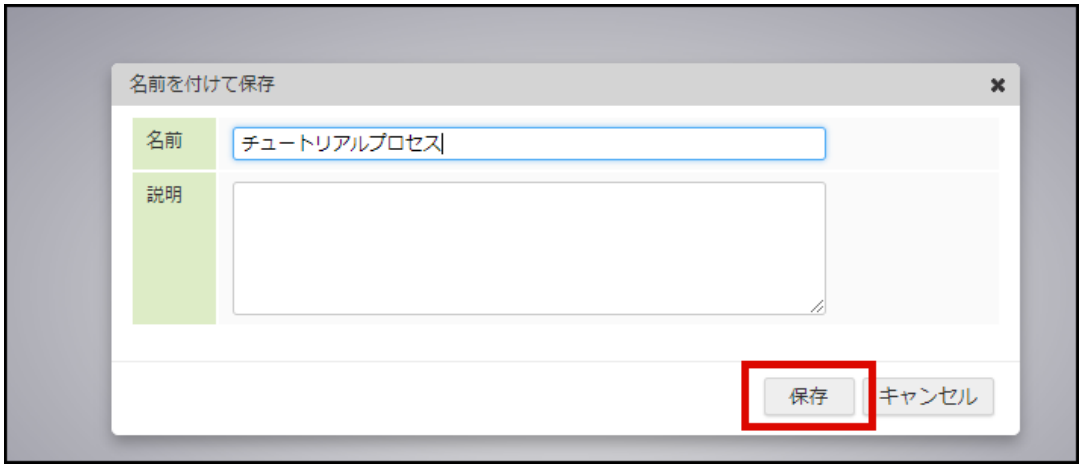
図：接続する

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

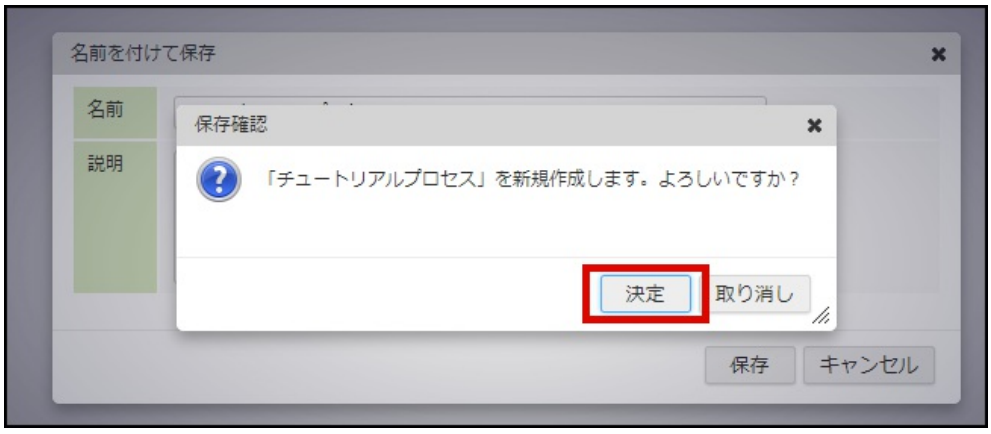
> **コラム**
>
> 「名前を付けて保存」と「名前を付けて保存（チェックなし）」の違い
>
> IM-BPM プロセスデザイナは、作成したプロセス定義がデプロイ可能かどうかをチェックを行う機能を有しています。
> 「名前を付けて保存」を実行した場合は上記チェックが行われ、デプロイ可能と判定されるまで保存できません。
> 「名前を付けて保存（チェックなし）」を実行した場合は上記チェックを行わず、保存が可能となる最低限のチェックのみを行って保存することができます。

12. 「名前」を入力して「保存」をクリックします。



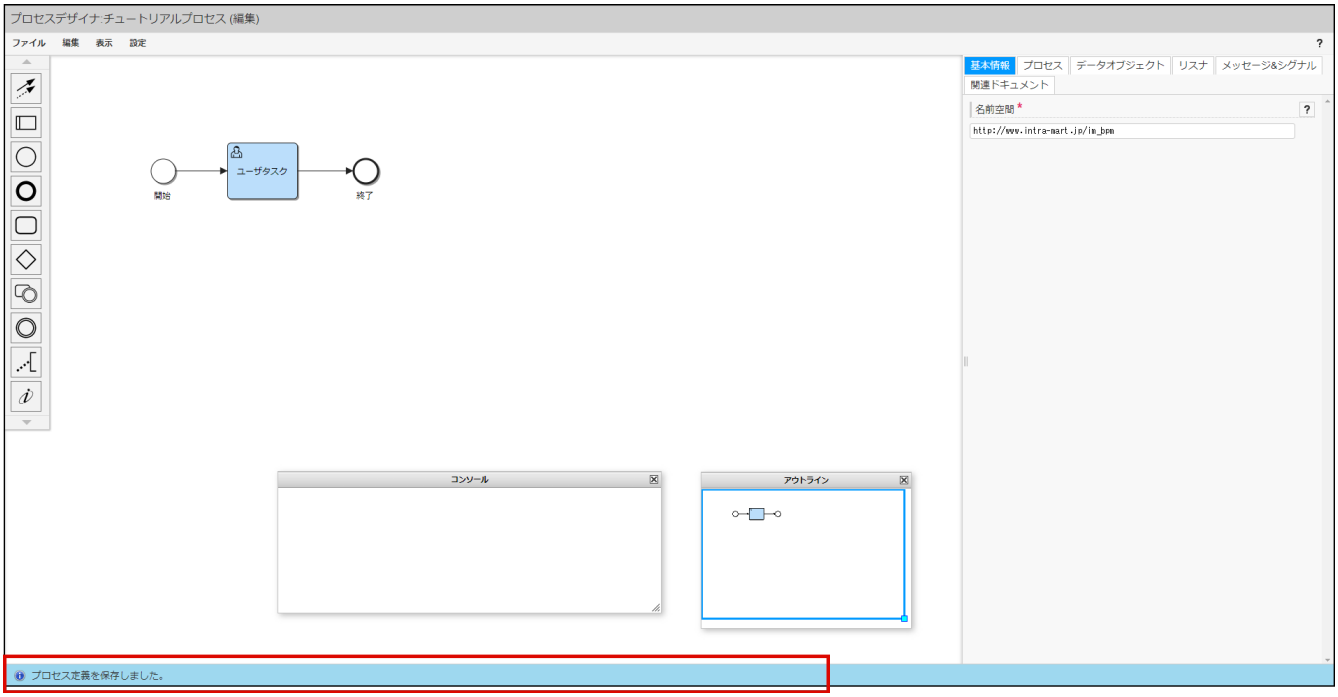
図：名前を付けて保存

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。

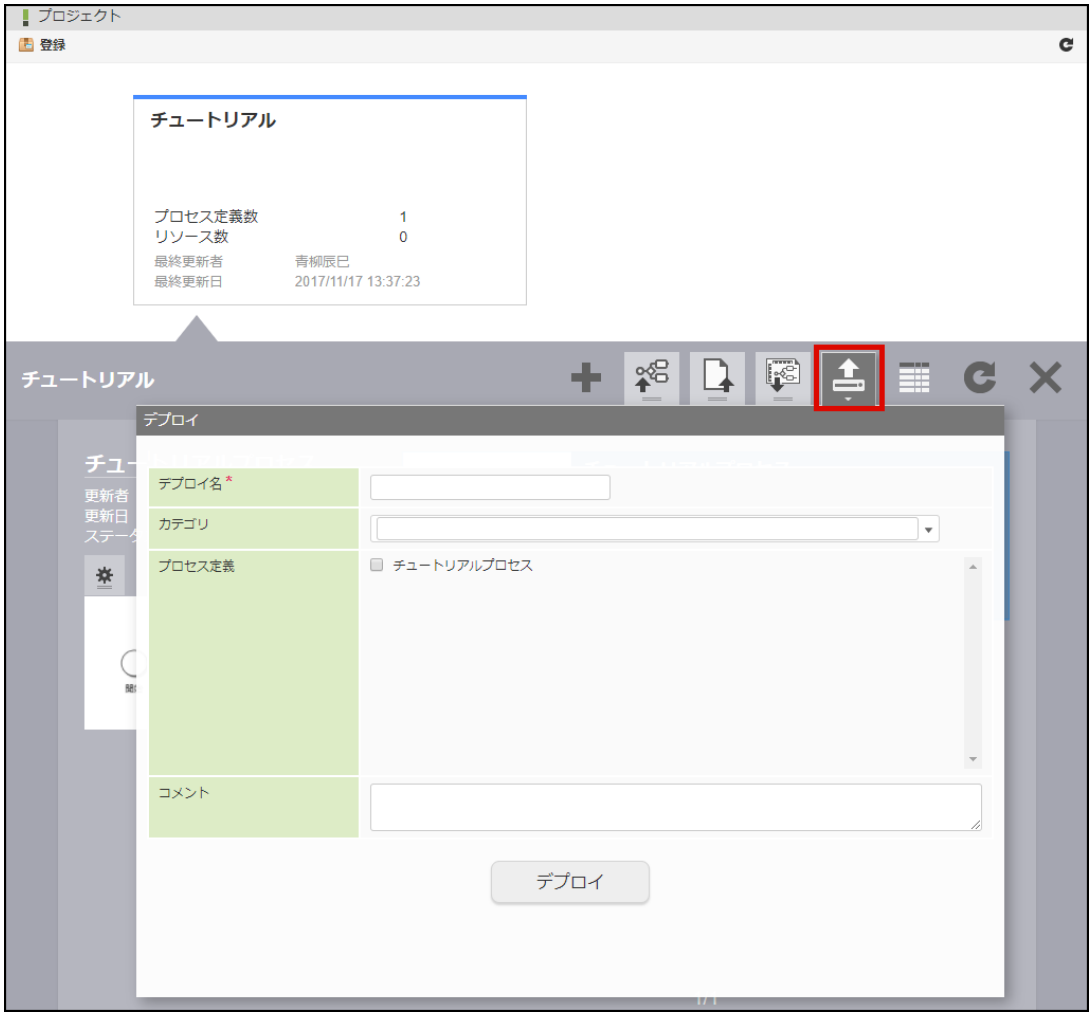


図：プロセス保存

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。

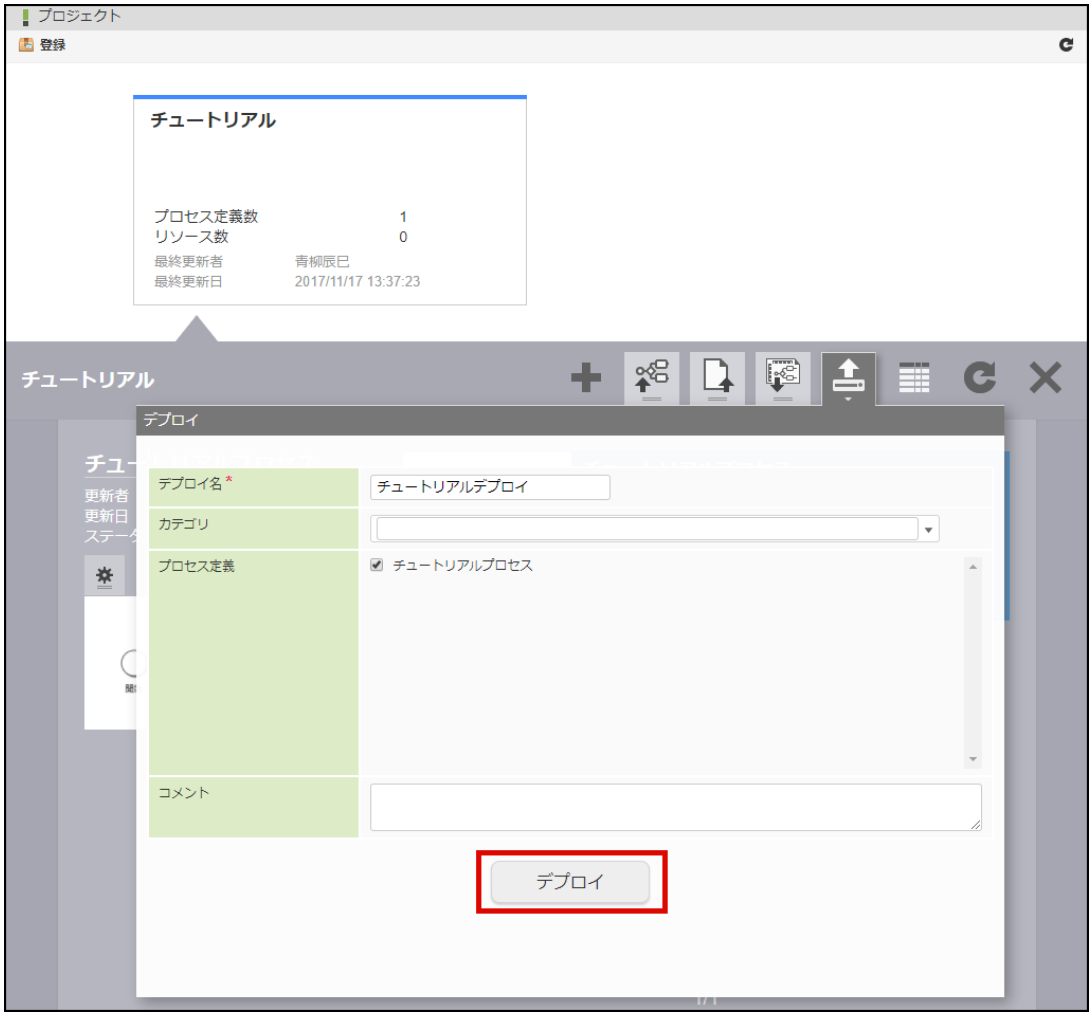
1. 「サイトマップ」→「BPM」→「プロセスデザイナ」をクリックします。
2. プロジェクトを選択し、 をクリックしデプロイエリアを開きます。



図：デプロイエリア

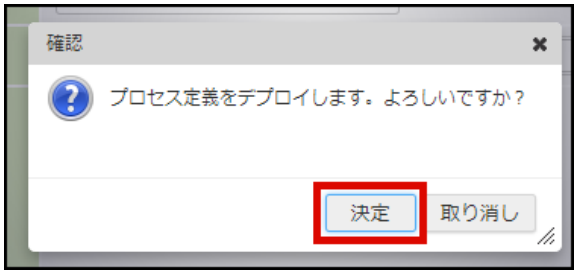
3. 「デプロイ名」の入力と「プロセス定義」の選択をします。
4. 「デプロイ」をクリックします。

※ デプロイについては「IM-BPM Designer 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。



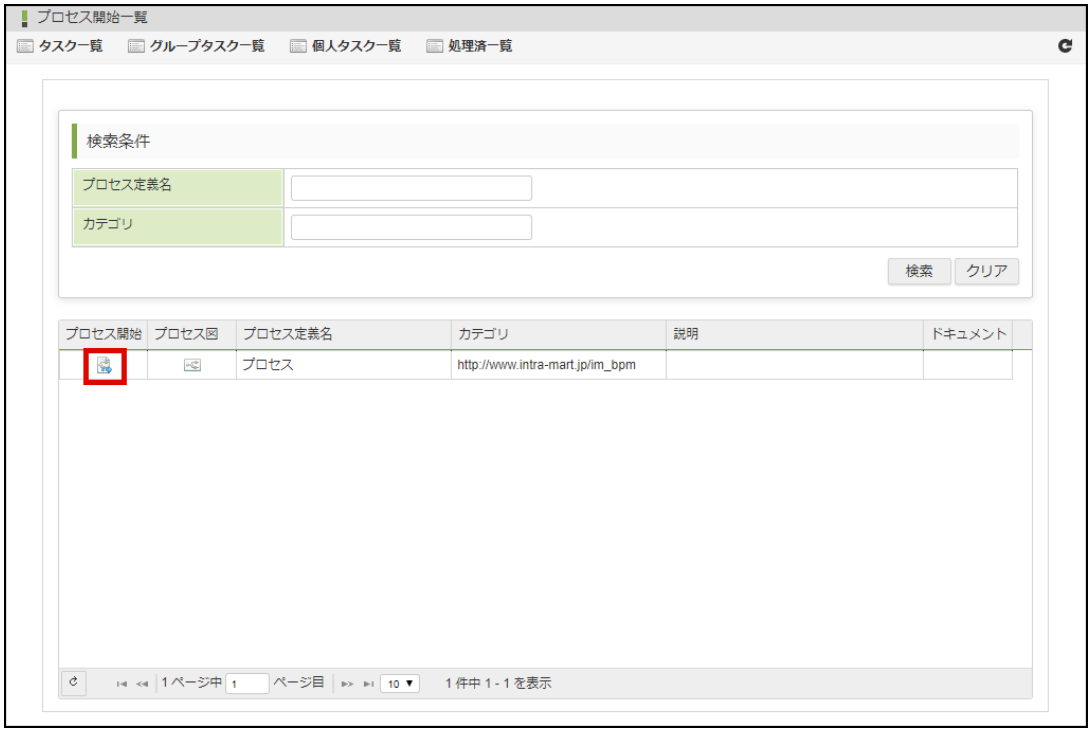
図：デプロイ確認

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
- 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
- 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

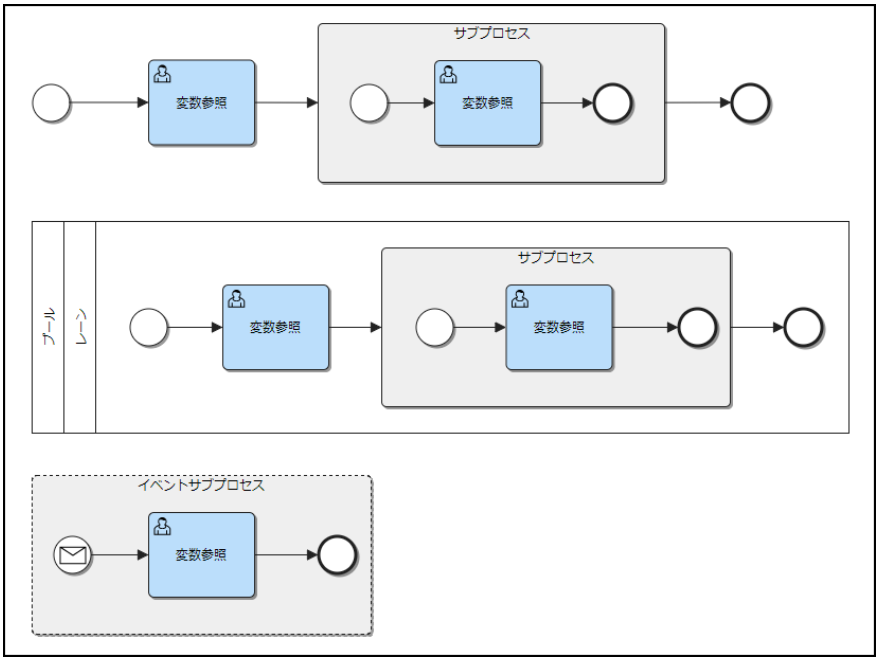
「データプロパティ」で定義した変数をエレメントから参照／更新することができます。

> **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することができます。
> アップロード手順は「IM-BPM Designer 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。



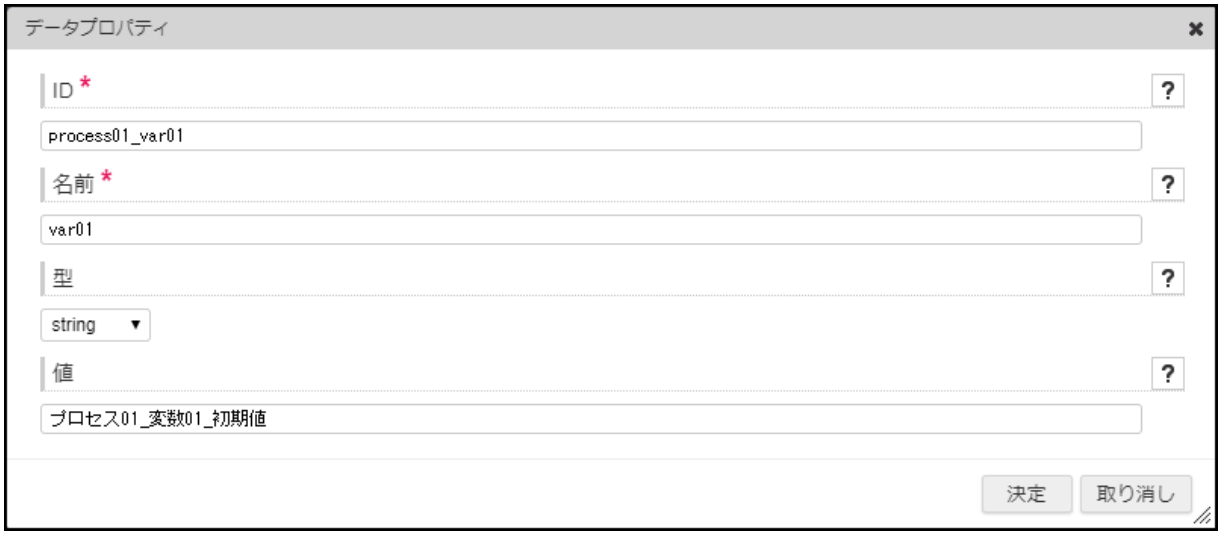
図：配置イメージ

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。

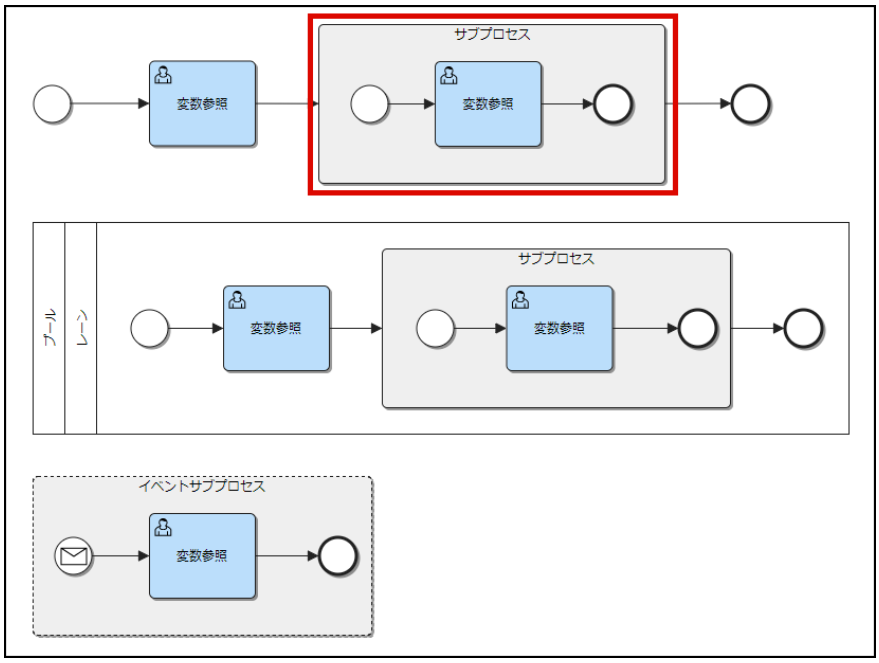
6. 同様の手順で、以下の変数も定義します。

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。

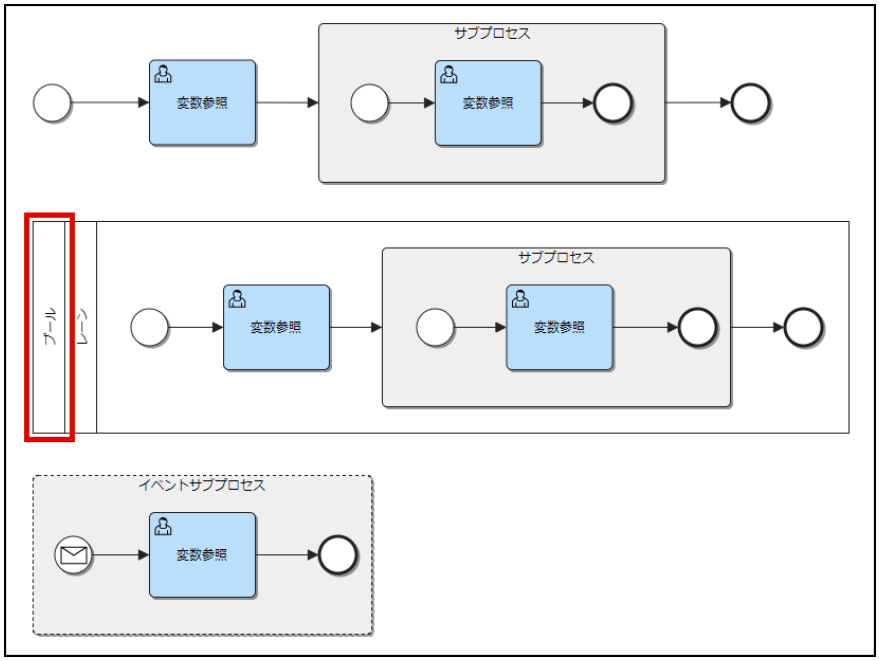
図：サブプロセス：プロパティ - データオブジェクト

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。



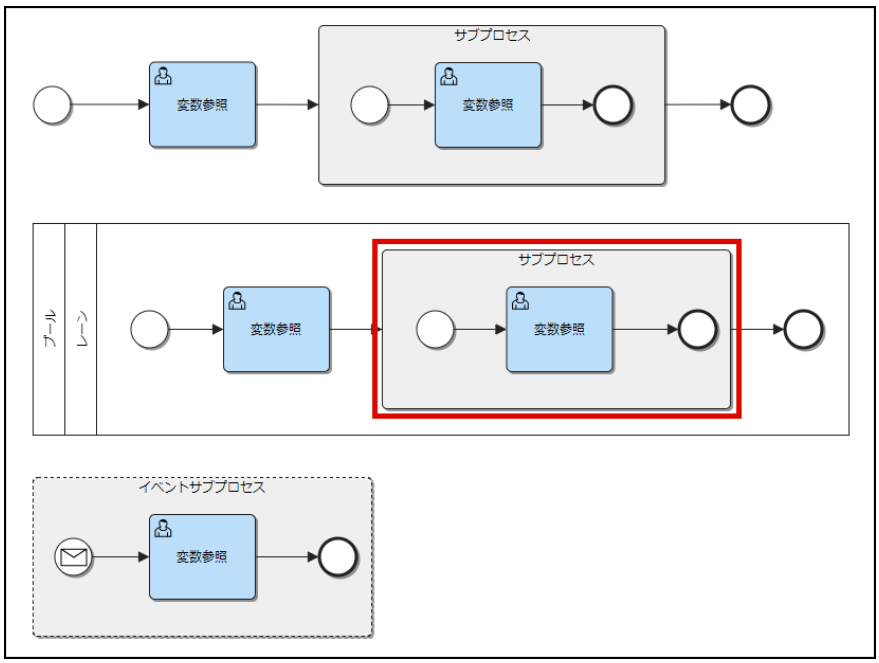
図：プール：プロパティ - データオブジェクト

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。



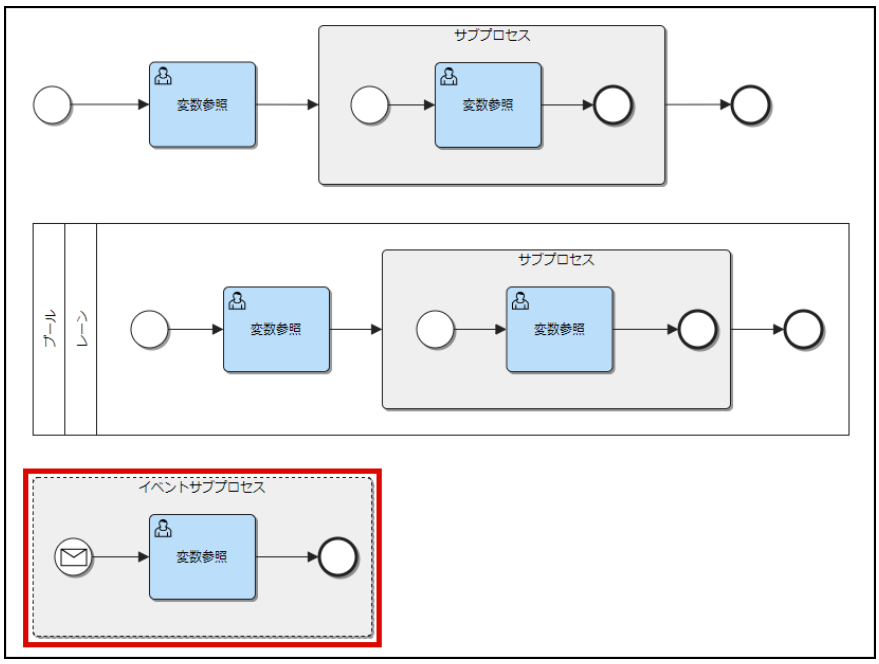
図：サブプロセス：プロパティ - データオブジェクト

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

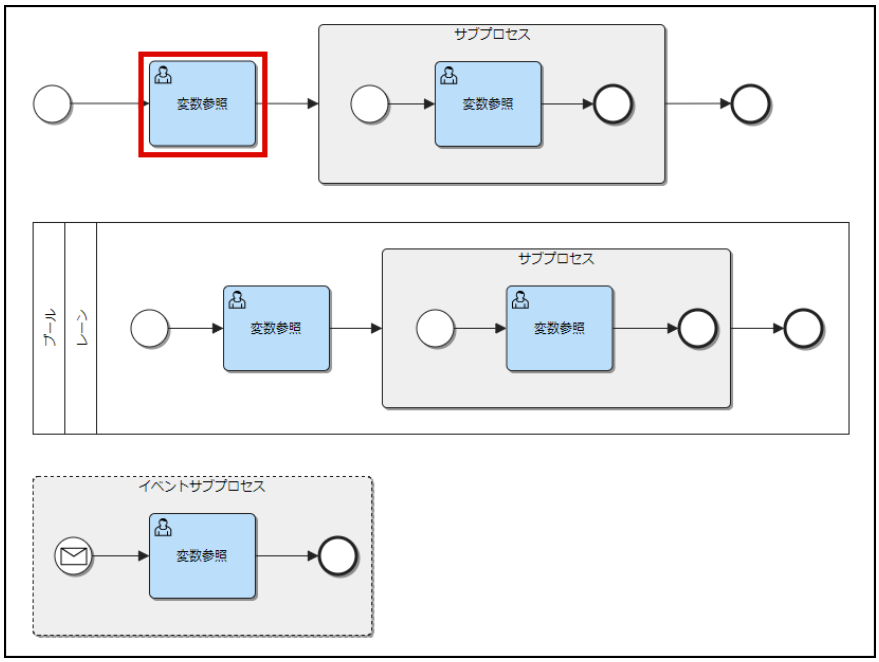
2. 以下の通り、データプロパティを定義します。

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。

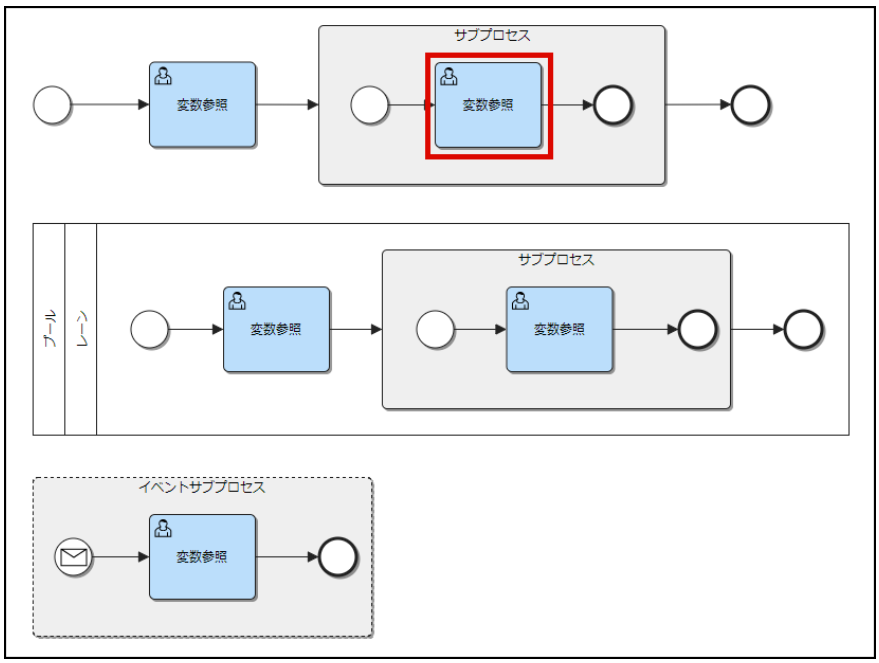
4. プロセス「データプロパティを定義する_プロセス01」のデータプロパティ「var01 <string>」と「var02 <string>」が参照できる

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

> **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。

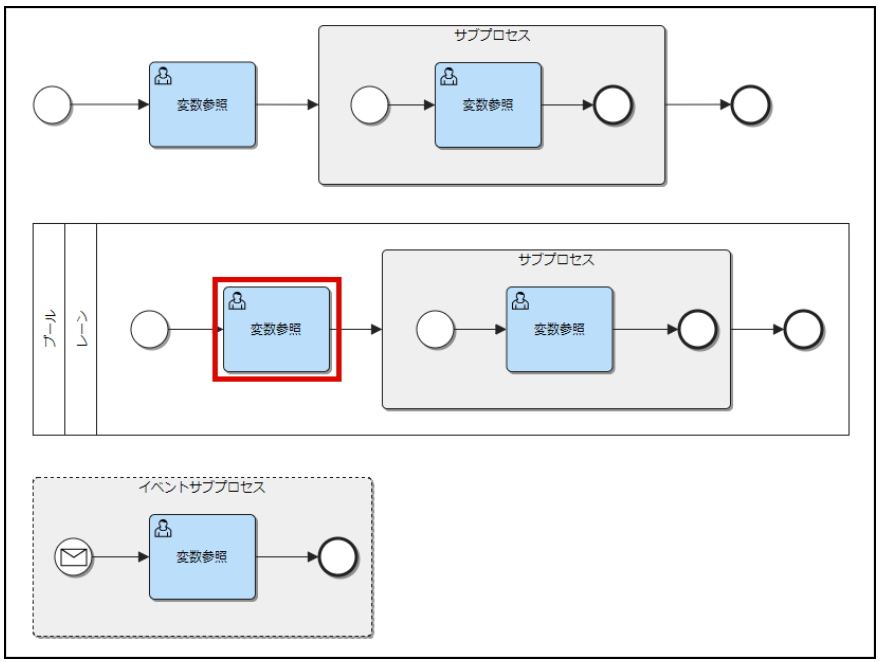
7. プロセス「データプロパティを定義する_プロセス01」のデータプロパティに加え、サブプロセスで定義したデータプロパティ「var01 <int>」と「var03 <boolean>」が参照できることがわかります。

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することができます。
> 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することができます。

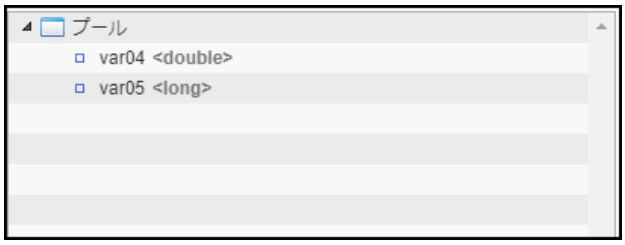
> **コラム**
>
> プロセスグローバルの定義とサブプロセスの定義で重複して命名されている変数「var01」を参照する場合、サブプロセスで定義された変数が参照されます。
> つまり、直近の階層で定義された変数が優先的に参照されます。

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

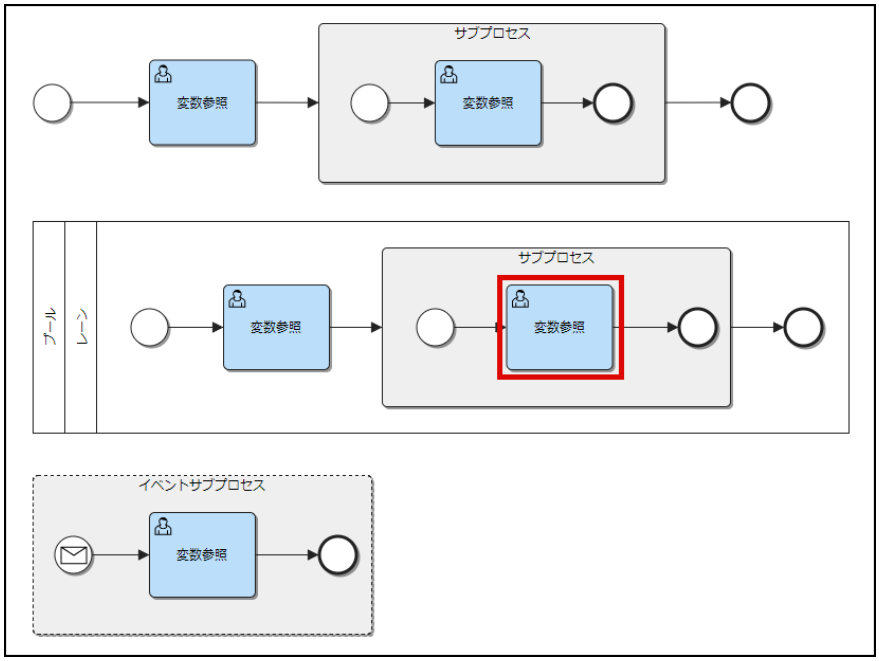
9. EL式エディタを開き、「データプロパティツリー」を確認します。
10. 「プール」のデータプロパティ「var04 <double>」と「var05 <long>」が参照できることがわかります。



図：EL式エディタ - データプロパティツリー

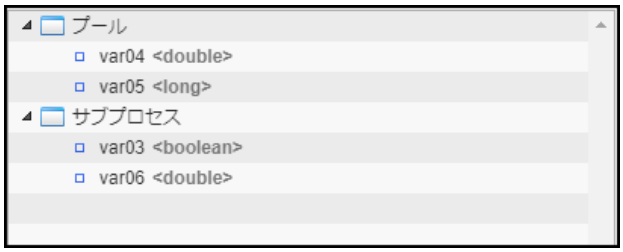
> **コラム**
>
> プールは一見、プロセスの中に配置されているように見えますが、そうではありません。
> プールは一つのプロセス定義として扱われるため、プロセスの子エレメントになることはありません。
> プロセス定義を跨って変数を参照することはできないため、プロセスで定義されたデータプロパティは参照できません。

11. 図の赤枠で示した「ユーザタスク」を選択します。



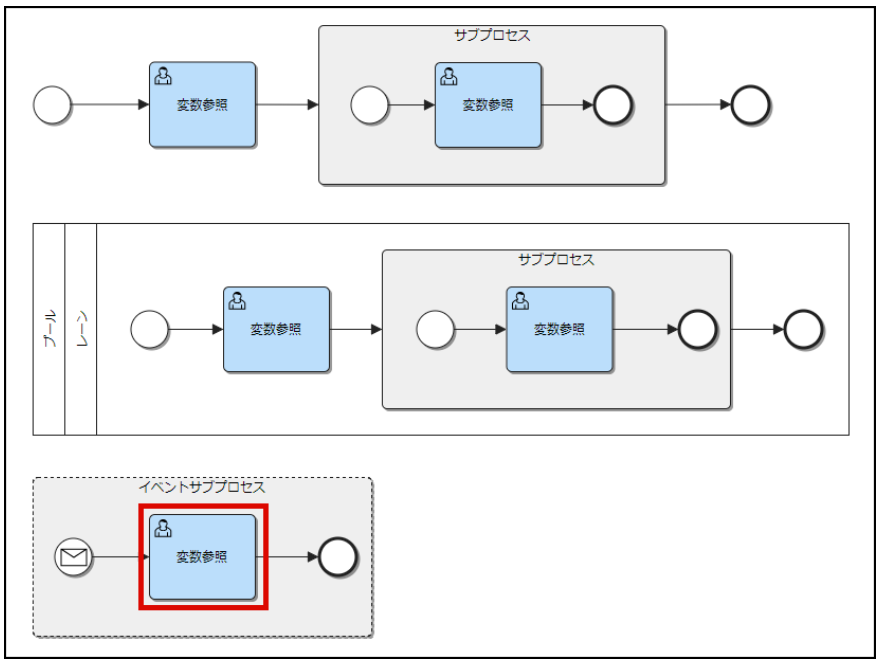
図：ユーザタスクの選択

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 Designer 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することができます。
> アップロード手順は「IM-BPM Designer 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。

- マルチインスタンスのプロパティを確認する
- マルチインスタンスをループ回数を指定して使用する
- マルチインスタンスを配列を指定して使用する
- マルチインスタンスを並列または、順次で実行する
- マルチインスタンスの要素変数を利用する
- マルチインスタンスの終了条件を利用する

マルチインスタンスのプロパティを確認する

マルチインスタンスの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

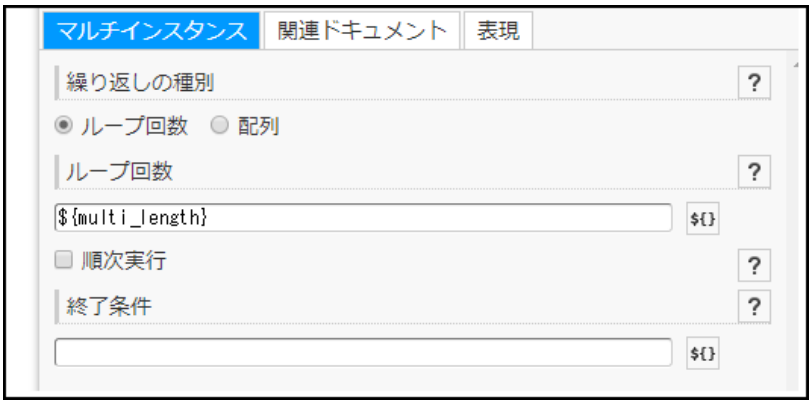
  繰り返しの種別が「配列」の場合、表示されます。
  「配列」の要素が「要素変数」に保存されます。

- 終了条件

  終了条件を設定します。

マルチインスタンスをループ回数を指定して使用する

マルチインスタンスを「ループ回数」を指定して使用します。
※ 並列で実行します。



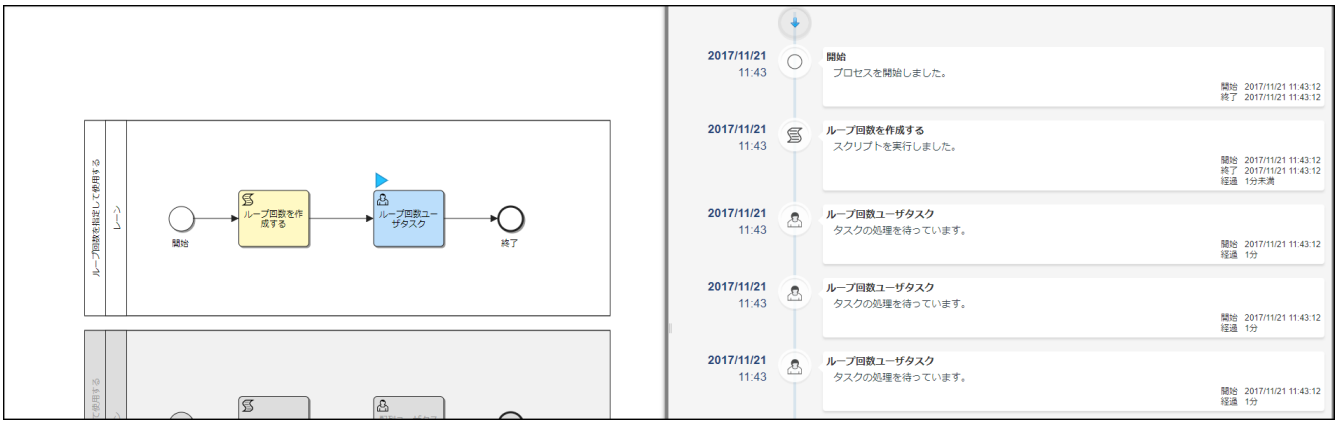
図：プロパティの設定

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」）。

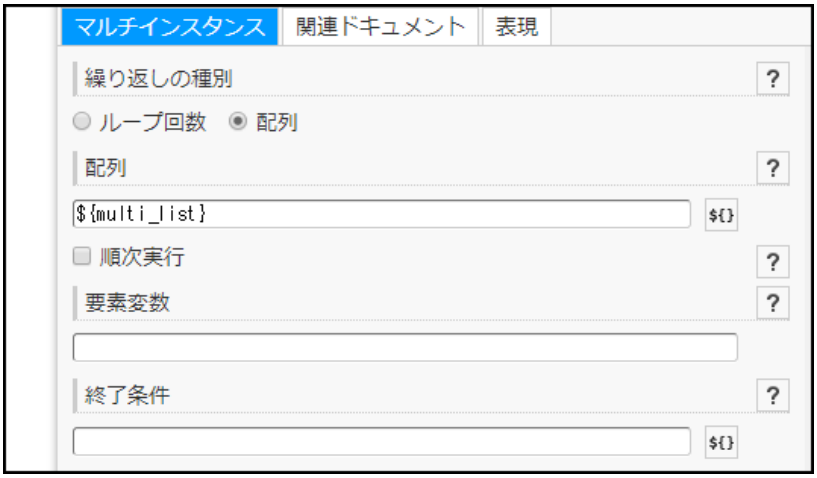
3. 「ループ回数」にEL式を使用して、multi_length の変数を指定します。



図：実行時のイメージ

マルチインスタンスを配列を指定して使用する

マルチインスタンスを「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」）。

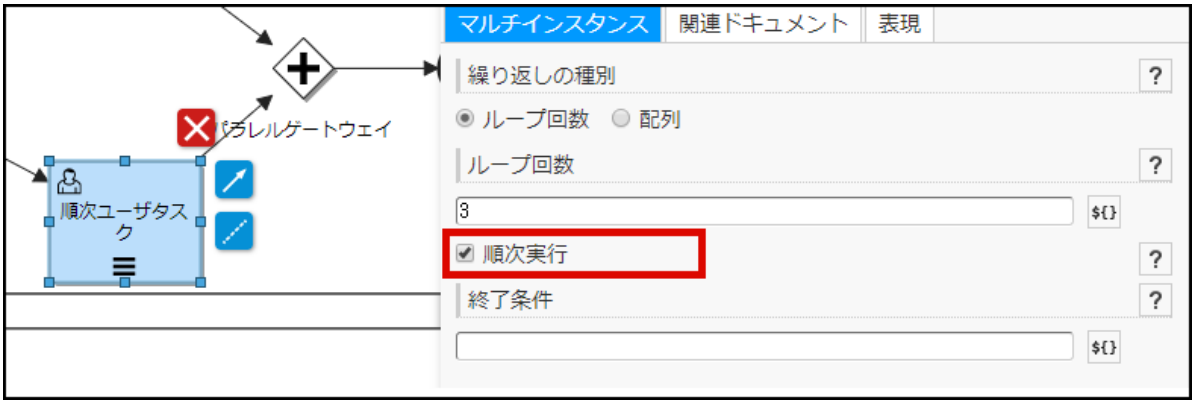
3. 「配列」にEL式を使用して、multi_list の変数を指定します。

図：実行時のイメージ

マルチインスタンスを並列または、順次で実行する

マルチインスタンスを並列または、順次で実行します。

図：並列



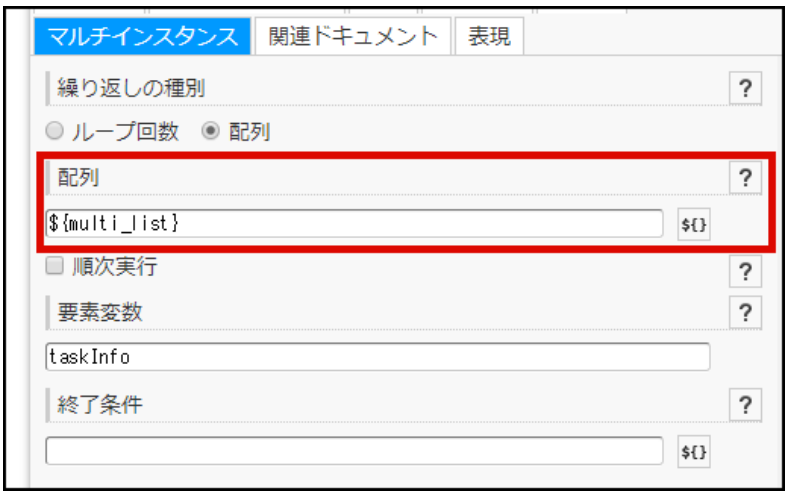
図：順次

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

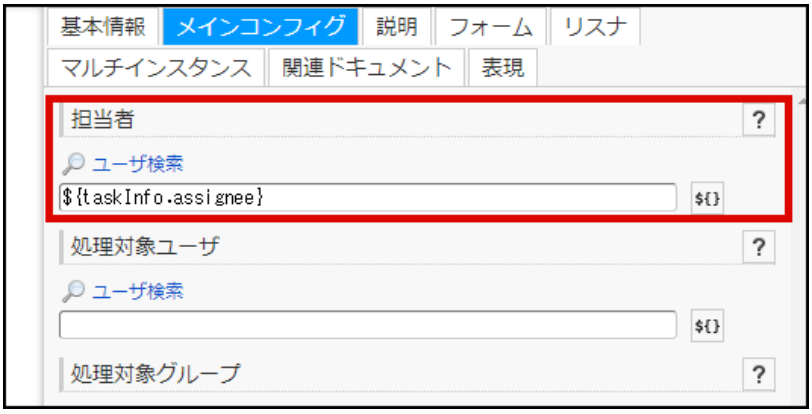
マルチインスタンスの要素変数を利用する

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'}]

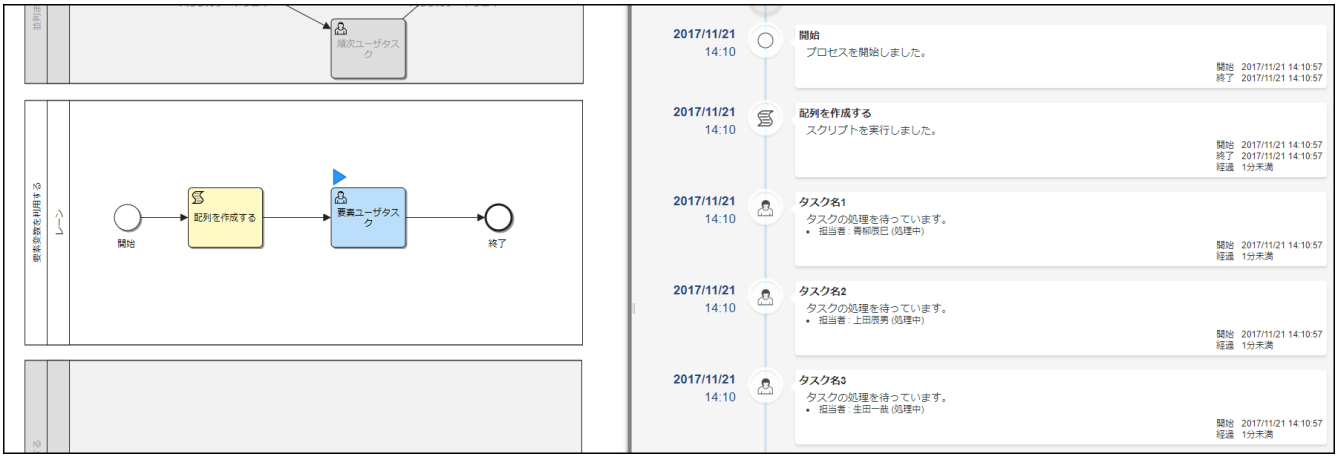
2. マルチインスタンスの「繰り返しの種別」を「配列」に設定します（初期設定は「ループ回数」）。

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。

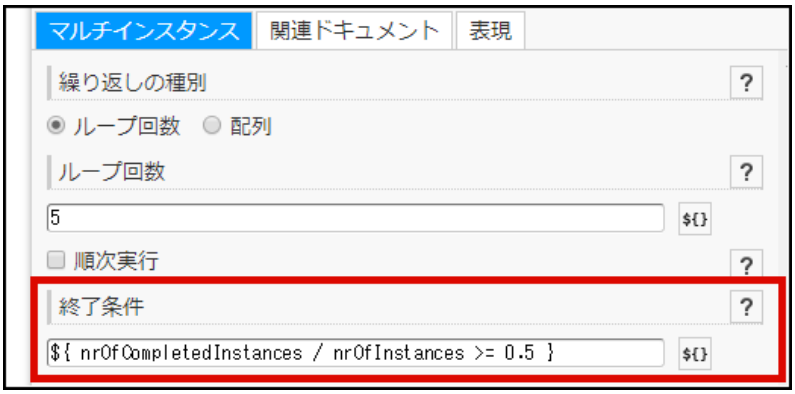
6. プロパティの「メインコンフィグ」 - 「担当者」にEL式を使用して、taskInfo.assignee を指定します。



図：実行時のイメージ

マルチインスタンスの終了条件を利用する

マルチインスタンスの「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。

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

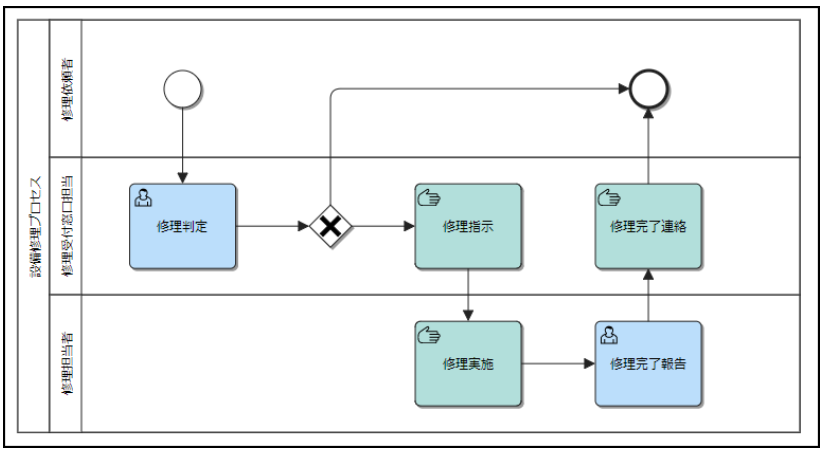
このチュートリアルでは、プールとレーンの基本的な使用方法について解説します。

> **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することができます。
> アップロード手順は「IM-BPM Designer 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

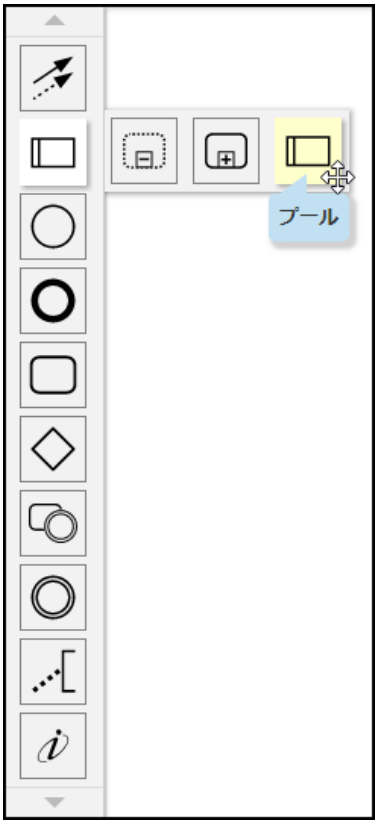
完成イメージ

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。



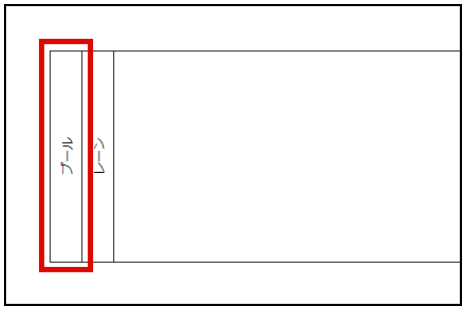
図：完成イメージ

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することができます。



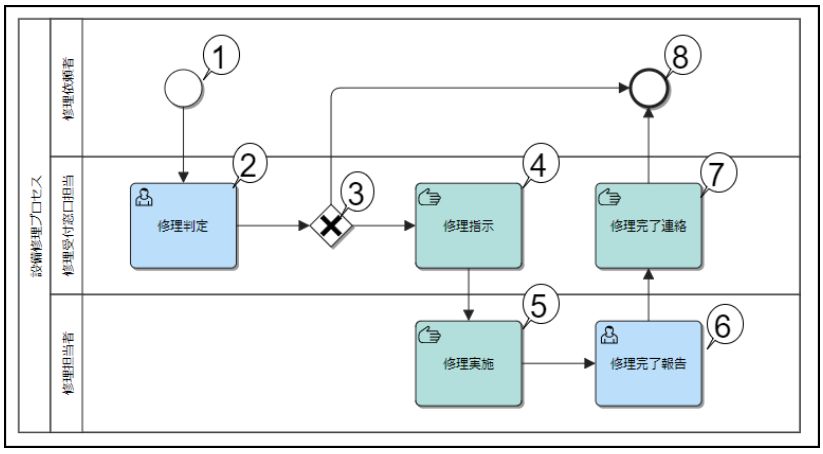
図：プールの選択

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

項番	レーン	エレメントタイプ	ラベル	接続元	接続先
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。
コールアクティビティの詳細については、「IM-BPM Designer 操作ガイド」 - 「コールアクティビティ」も併せて参照してください。

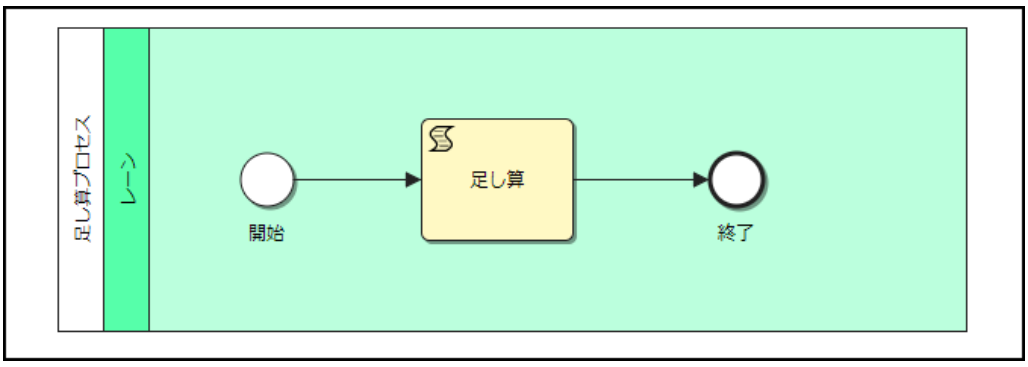
> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **call_activity_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードすることができます。
> アップロード手順は「IM-BPM Designer 操作ガイド」-「プロセス定義のアップロード」を参照してください。

> **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);
}
```

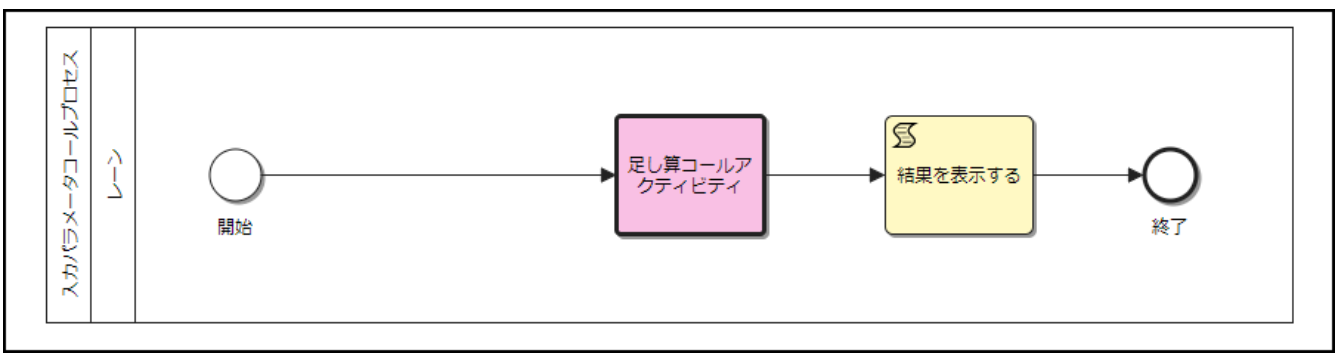
「プロセス定義キー」に addition_process を設定します。

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。

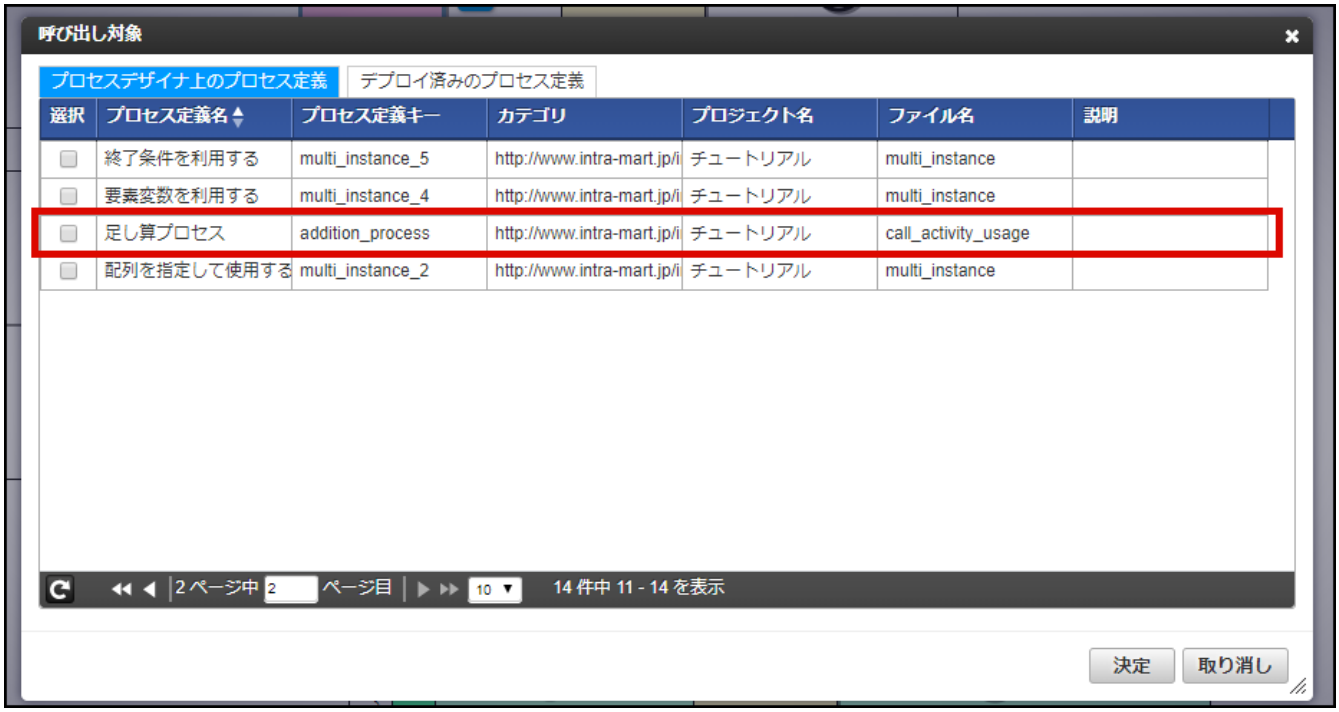


図：完成イメージ

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」の項目へ設定することができます。

- **「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 Designer 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。

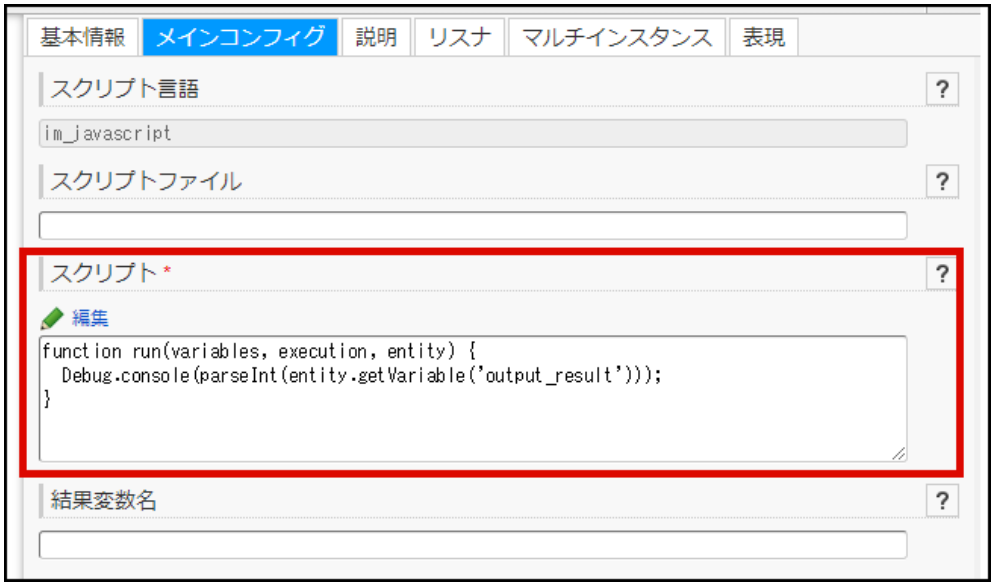
図：コールアクティビティ - メインコンフィグ - 出力パラメータ

   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。

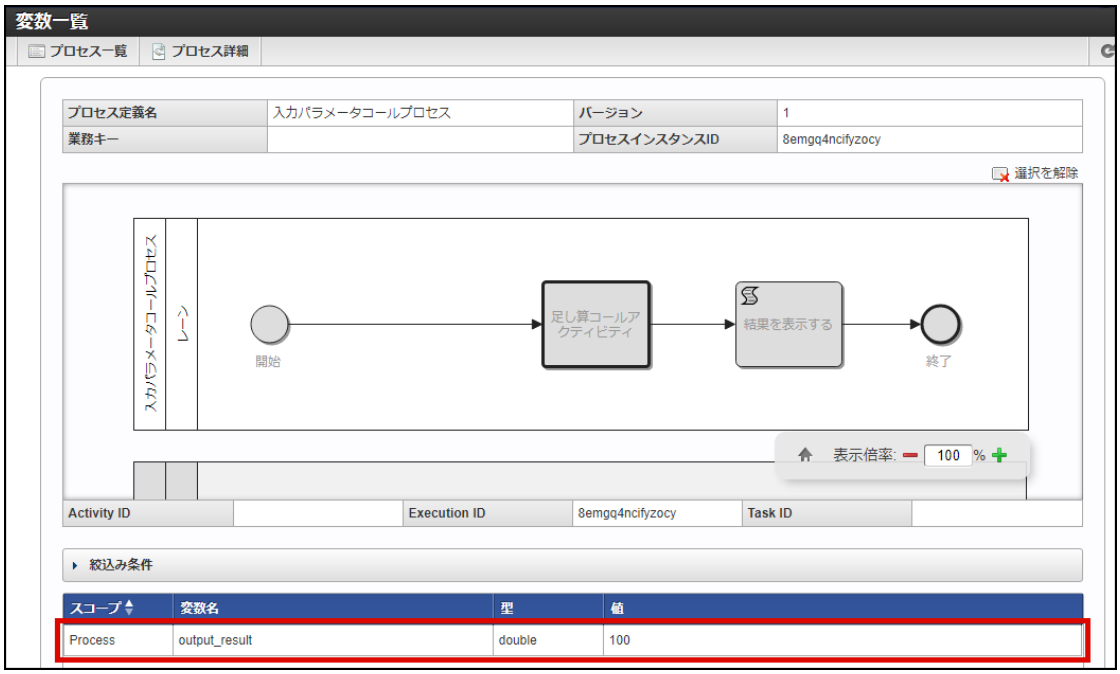
```
function run(variables, execution, entity) {
  Debug.console(parseInt(entity.getVariable('output_result')));
}
```



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。

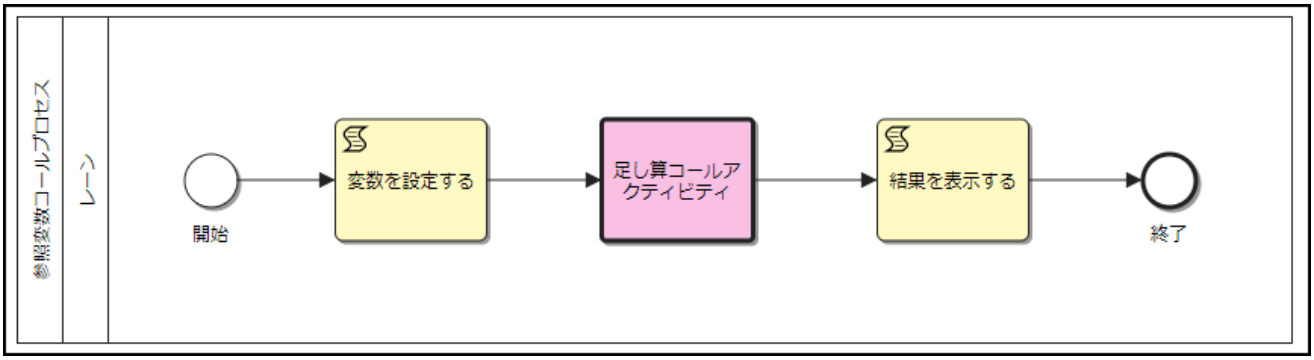
図：結果表示



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。



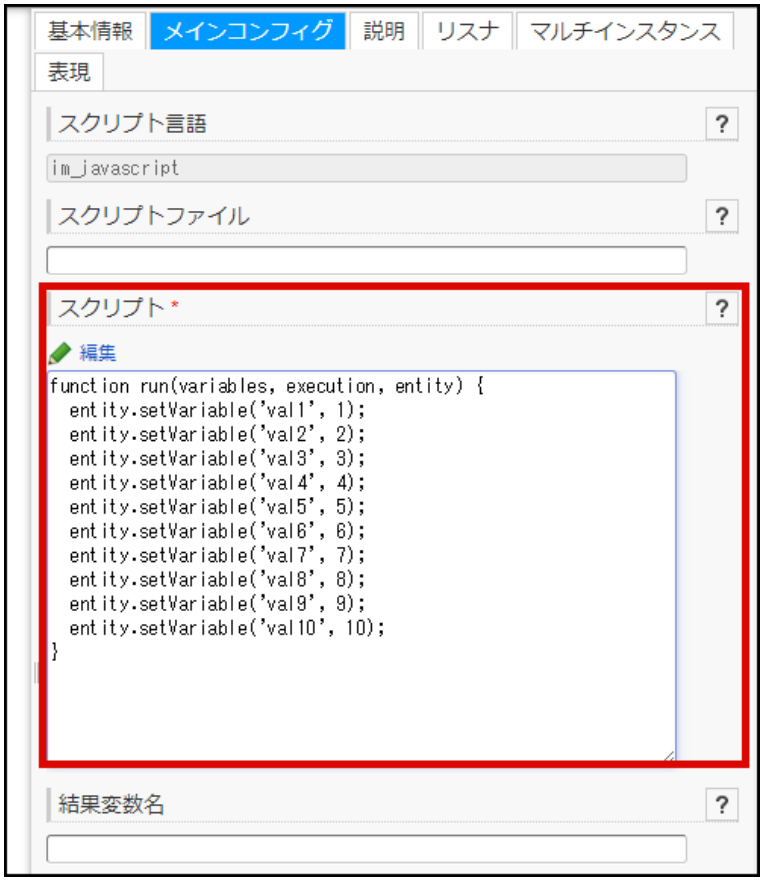
図：完成イメージ

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。

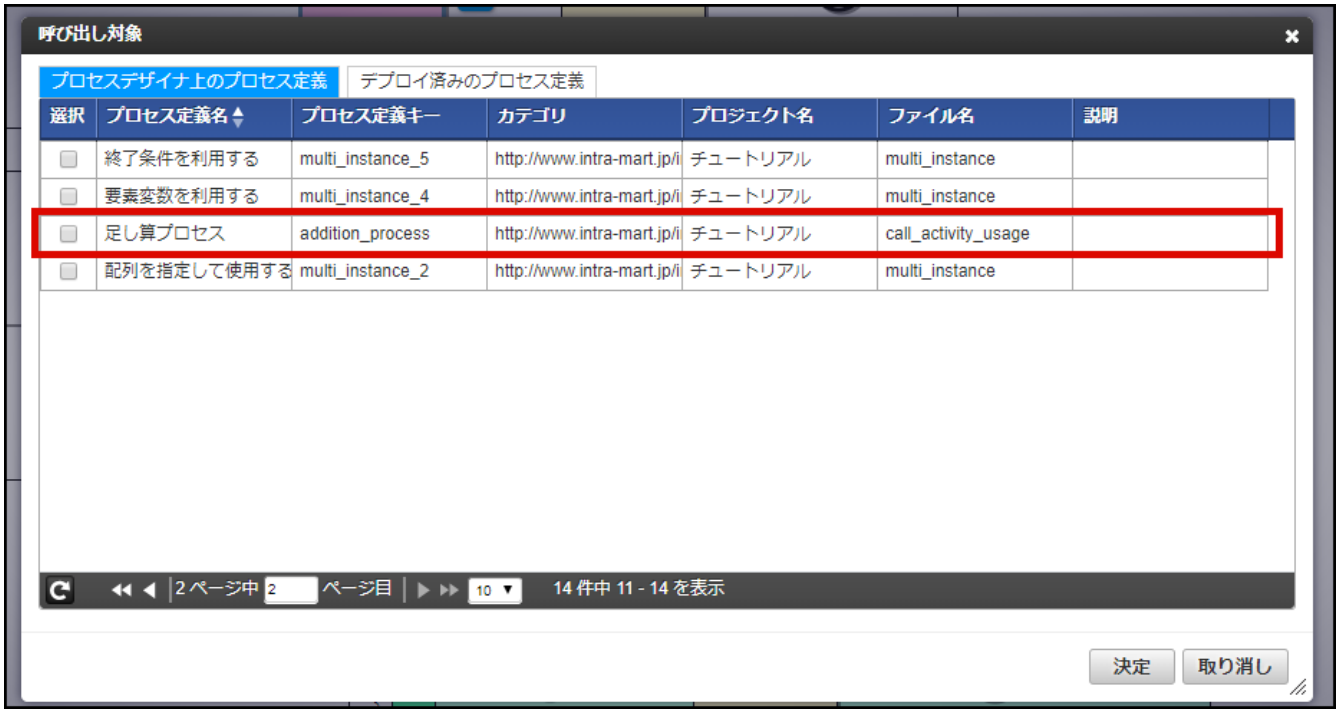
```
function run(variables, execution, entity) {
  entity.setVariable('val1', 1);
  entity.setVariable('val2', 2);
  entity.setVariable('val3', 3);
  entity.setVariable('val4', 4);
  entity.setVariable('val5', 5);
  entity.setVariable('val6', 6);
  entity.setVariable('val7', 7);
  entity.setVariable('val8',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

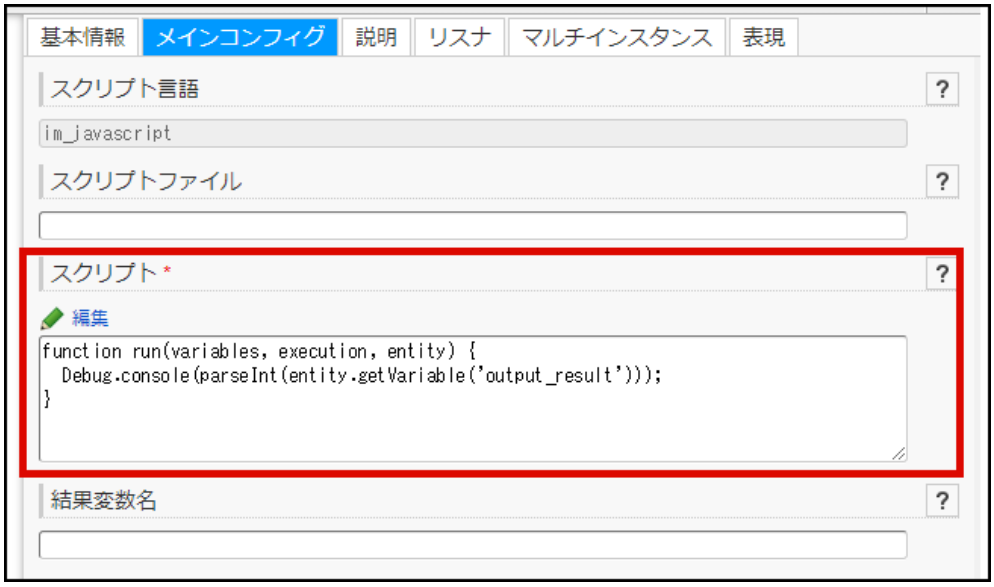
図：コールアクティビティ - メインコンフィグ - 出力パラメータ

   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。

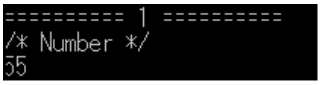
   「スクリプトタスク」のスクリプトです。

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

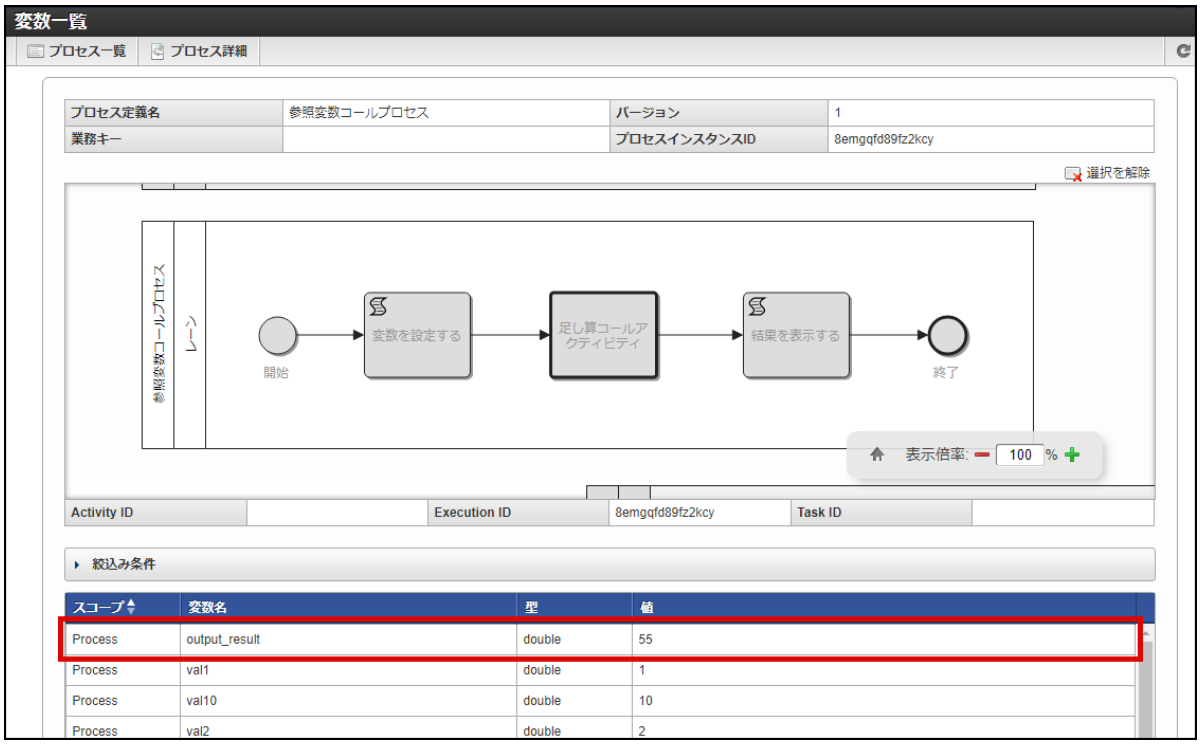


図：スクリプトタスク - メインコンフィグ - スクリプト

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

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の詳細については、「IM-BPM Designer 操作ガイド」 - 「ゲートウェイ」 - 「パラレルゲートウェイ」も併せて参照してください。

コラム

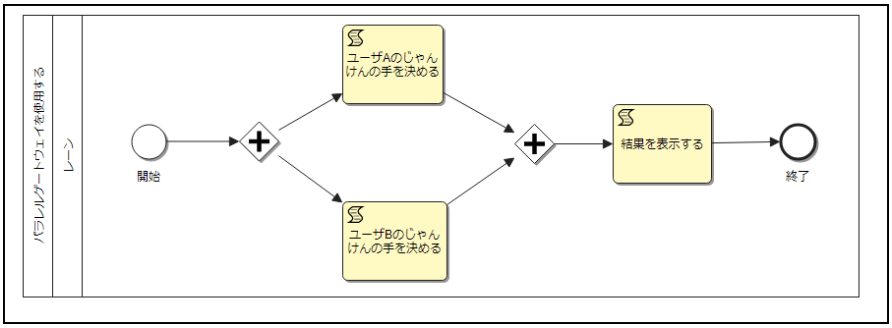
このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
parallel_gateway_usage.bpmn

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することができます。
アップロード手順は「IM-BPM Designer 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。

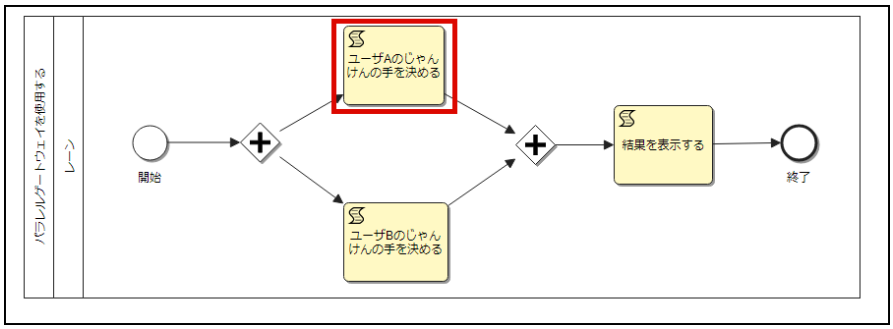


図：完成イメージ

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。

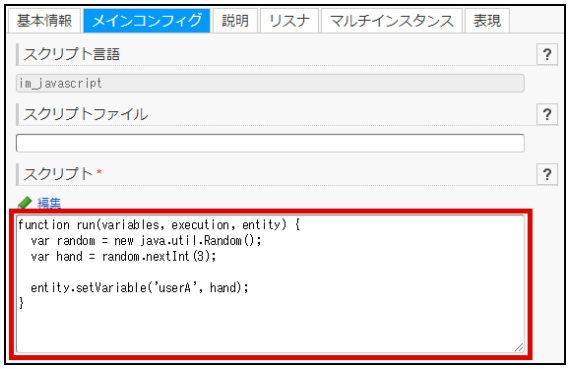


図：スクリプトタスク「ユーザAのじゃんけんの手を決める」

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。

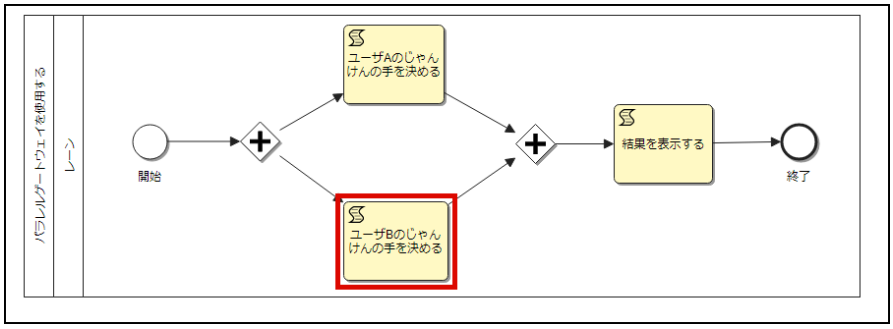
```
function run(variables, execution, entity) {
  var random = new java.util.Random();
  var hand = random.nextInt(3);

  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。

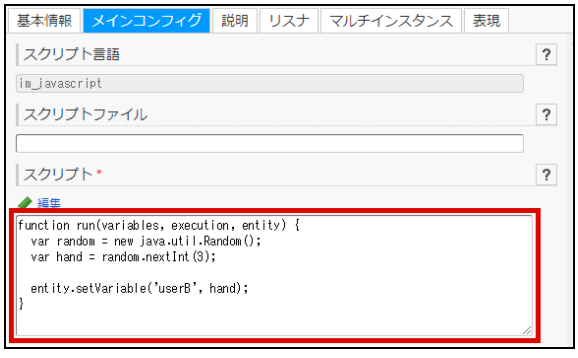


図：スクリプトタスク「ユーザBのじゃんけんの手を決める」

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。

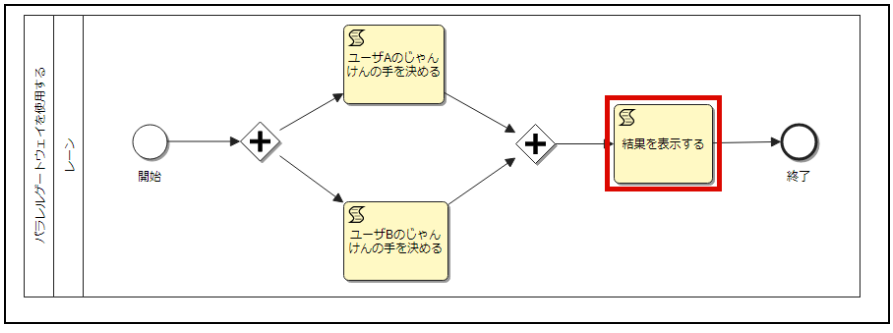
```
function run(variables, execution, entity) {
  var random = new java.util.Random();
  var hand = random.nextInt(3);

  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。

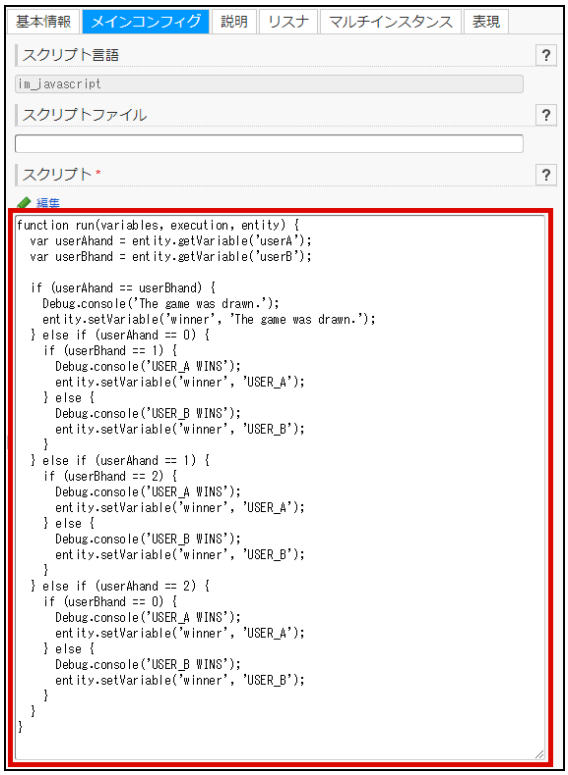


図：スクリプトタスク「結果を表示する」

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');
    }
  }
}
```

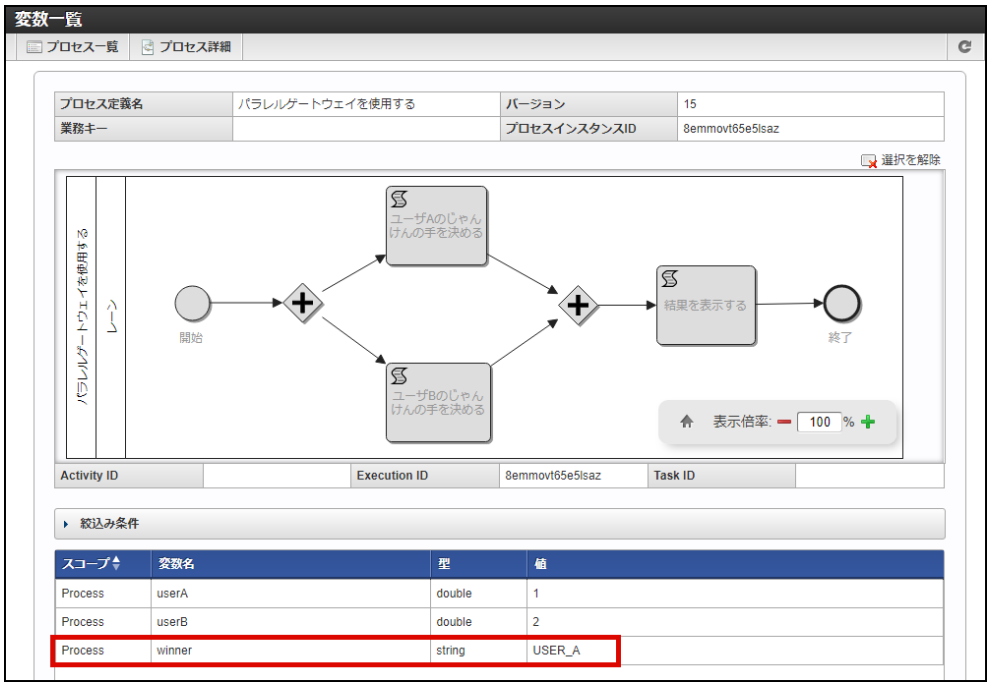


図：スクリプトタスク「結果を表示する」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

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。

```
/* String */
"USER_A WINS"
```

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

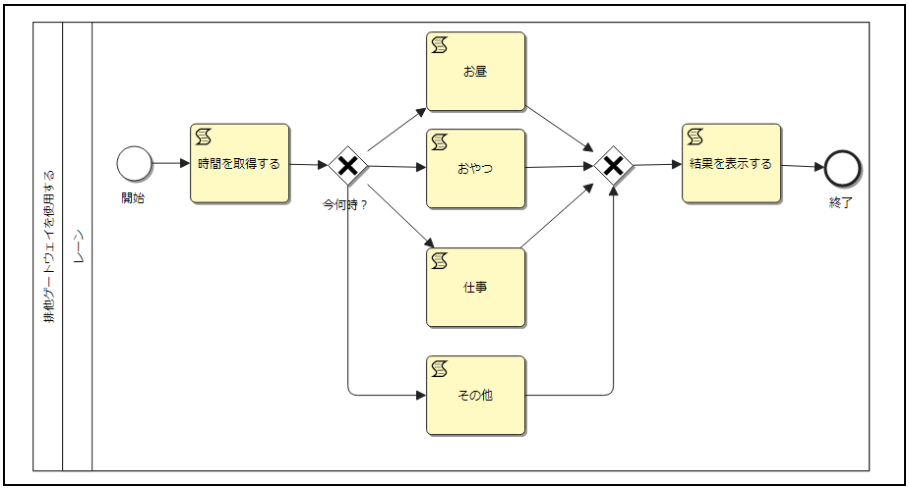
このチュートリアルでは、排他ゲートウェイを使用してプロセスの遷移先の選択や合流の定義方法を解説します。
排他ゲートウェイの詳細については、「IM-BPM Designer 操作ガイド」 - 「ゲートウェイ」 - 「排他ゲートウェイ」も併せて参照してください。

> **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することができます。
> アップロード手順は「IM-BPM Designer 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

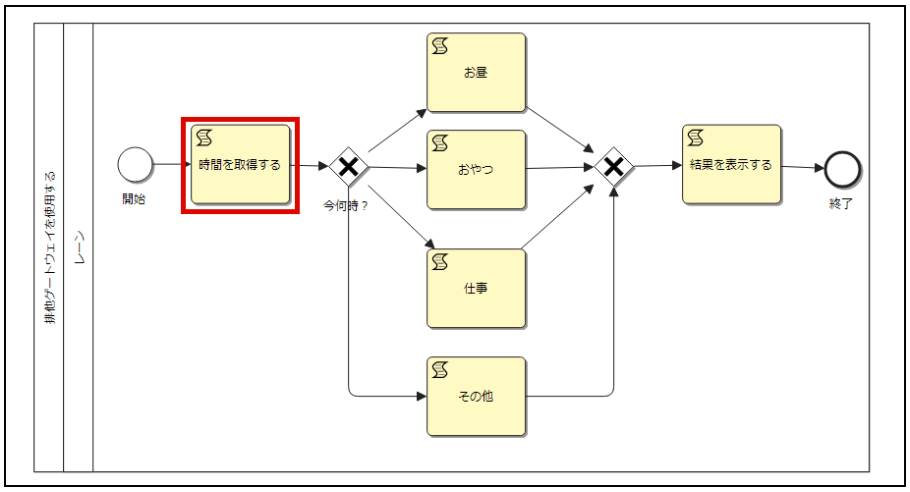
排他ゲートウェイを使用し、複数の分岐先から1つの遷移先を選択する

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。

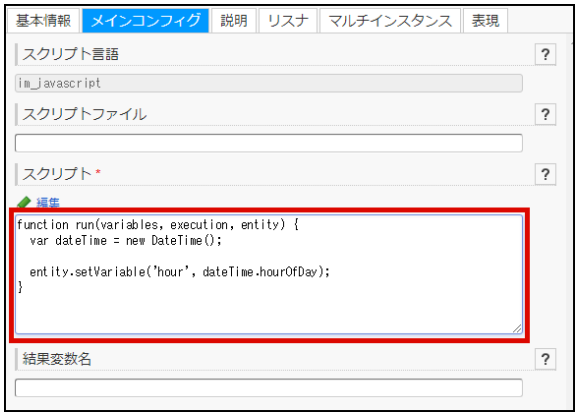


図：スクリプトタスク「時間を取得する」

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。

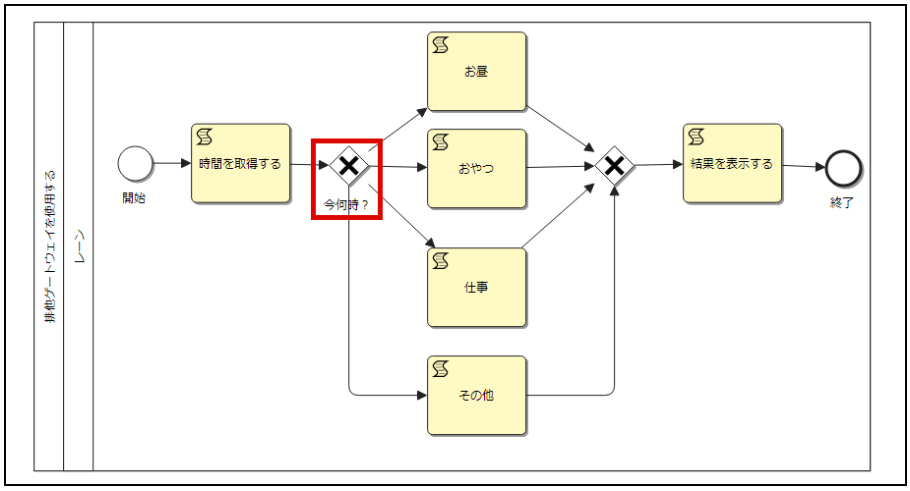
```
function run(variables, execution, entity) {
  var dateTime = new DateTime();

  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。

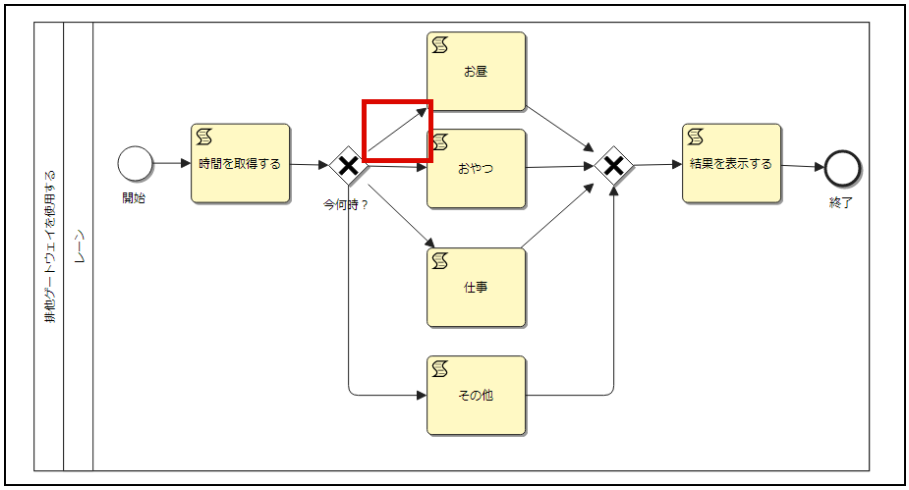


図：排他ゲートウェイ「今何時？」

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」

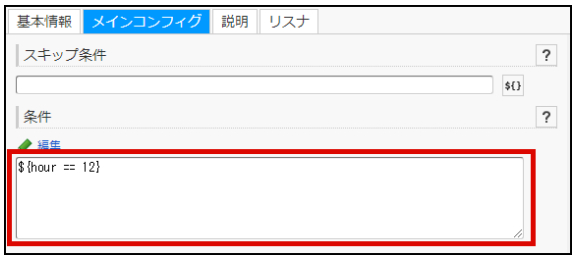
3. シーケンスフロー「sequence-flow_3」の条件を設定します。



図：シーケンスフロー「sequence-flow_3」

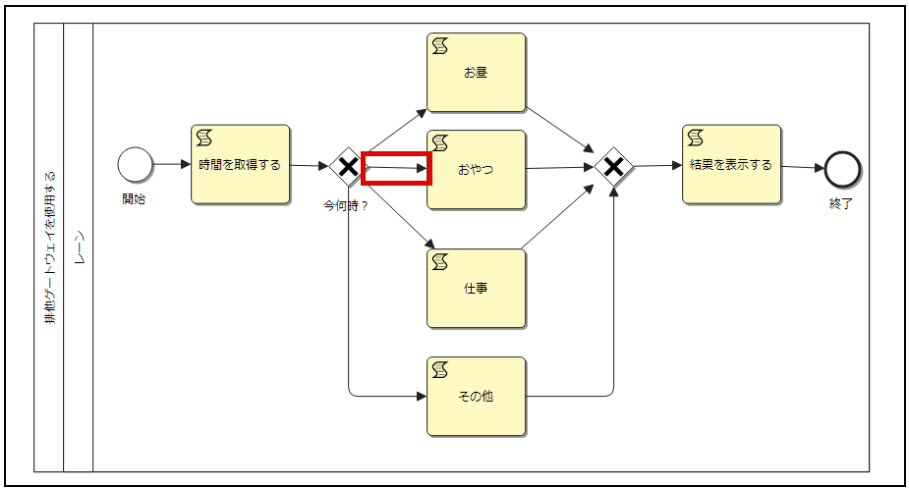
排他ゲートウェイ「今何時？」とスクリプトタスク「お昼」を接続しているシーケンスフロー「sequence-flow_3」 に、変数 hour が 12 の場合に真となる条件を設定します。

```
${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」

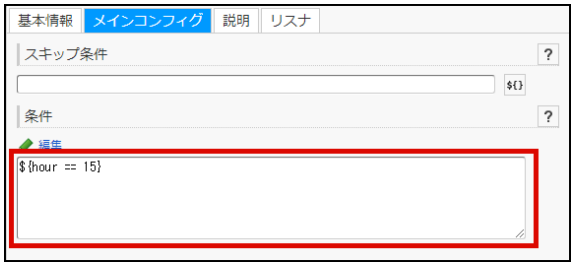
4. シーケンスフロー「sequence-flow_4」の条件を設定します。



図：シーケンスフロー「sequence-flow_4」

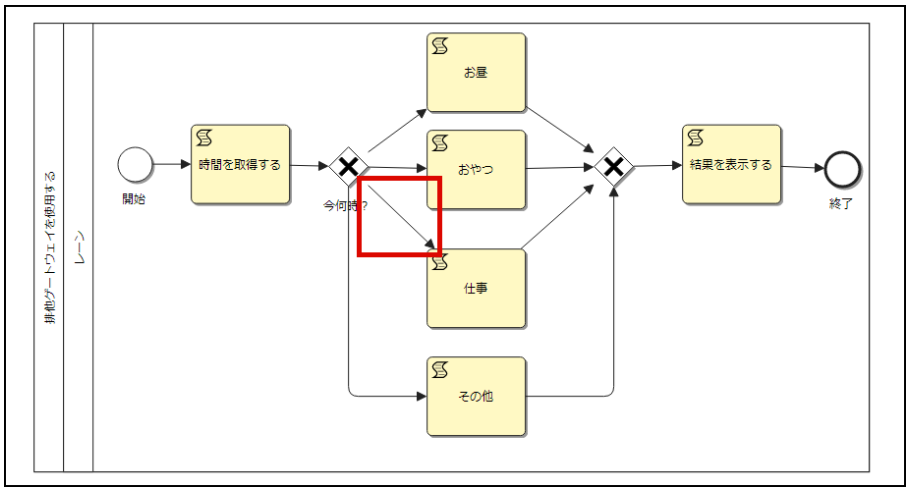
排他ゲートウェイ「今何時？」とスクリプトタスク「おやつ」を接続しているシーケンスフロー「sequence-flow_4」 に、変数 `hour` が `15` の場合に真となる条件を設定します。

```
${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」

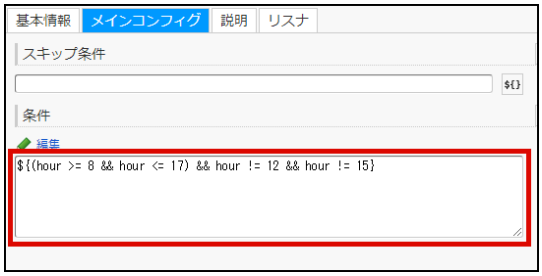
5. シーケンスフロー「sequence-flow_5」の条件を設定します。



図：シーケンスフロー「sequence-flow_5」

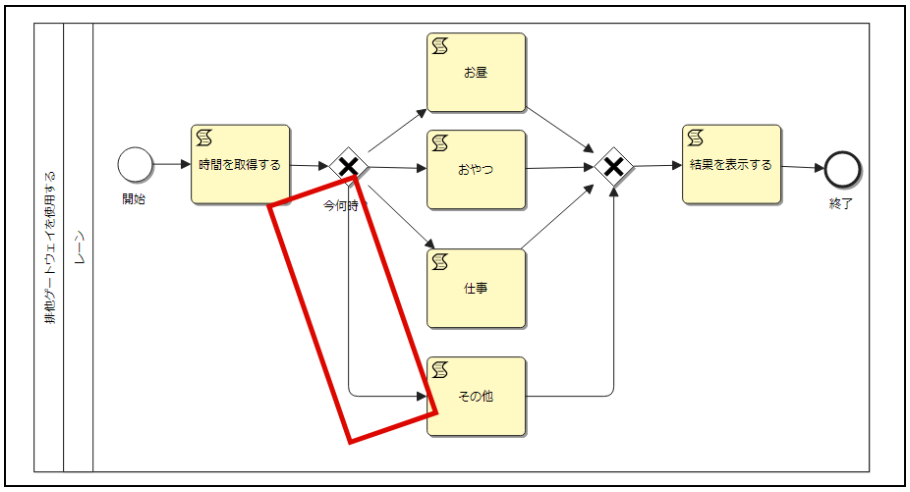
排他ゲートウェイ「今何時？」とスクリプトタスク「仕事」を接続しているシーケンスフロー「sequence-flow_5」 に、変数 `hour` が `8` 以上かつ `17` 以下で、 `12` ではなく、 `15` でもない場合に真となる条件を設定します。

```
${(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」

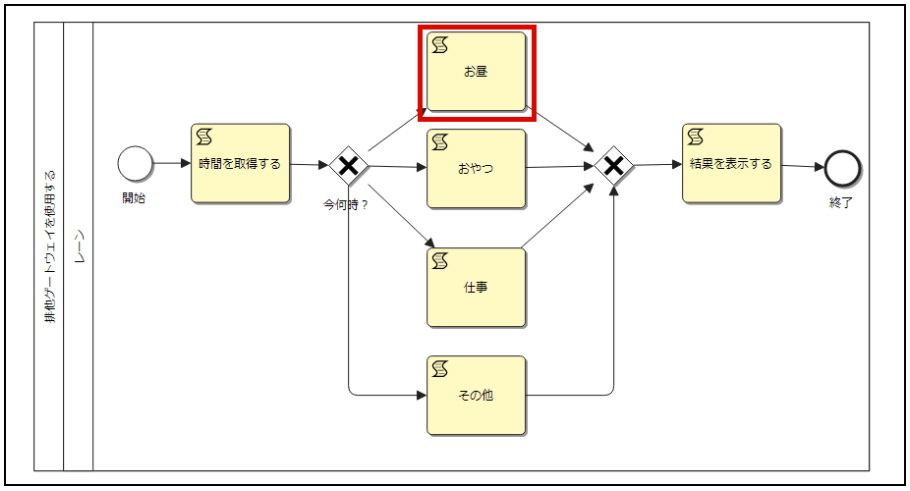
6. シーケンスフロー「sequence-flow_6」の条件を設定します。



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。

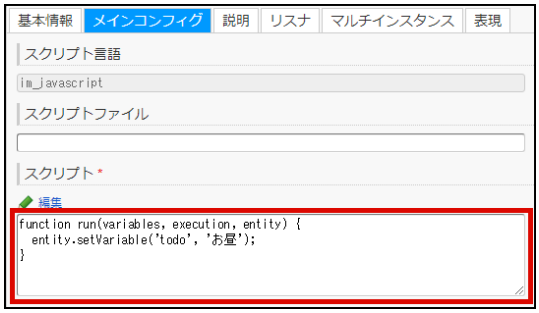
7. スクリプトタスク「お昼」 のプロパティを設定します。



図：スクリプトタスク「お昼」

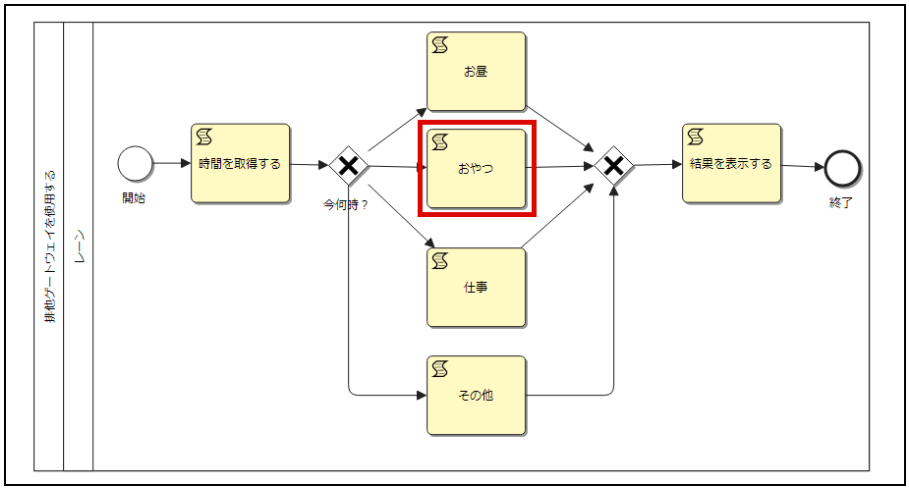
スクリプトタスク「お昼」のプロパティにて、作業名 `お昼` を変数 `todo` に格納するスクリプトを設定します。

```
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」

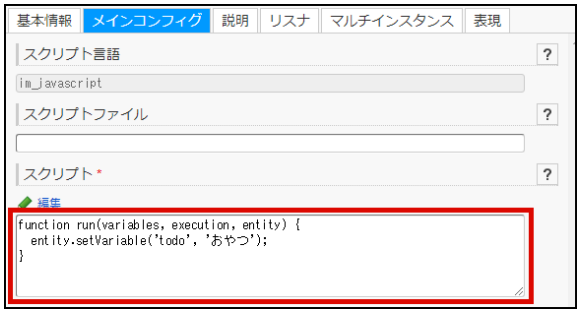
8. スクリプトタスク「おやつ」 のプロパティを設定します。



図：スクリプトタスク「おやつ」

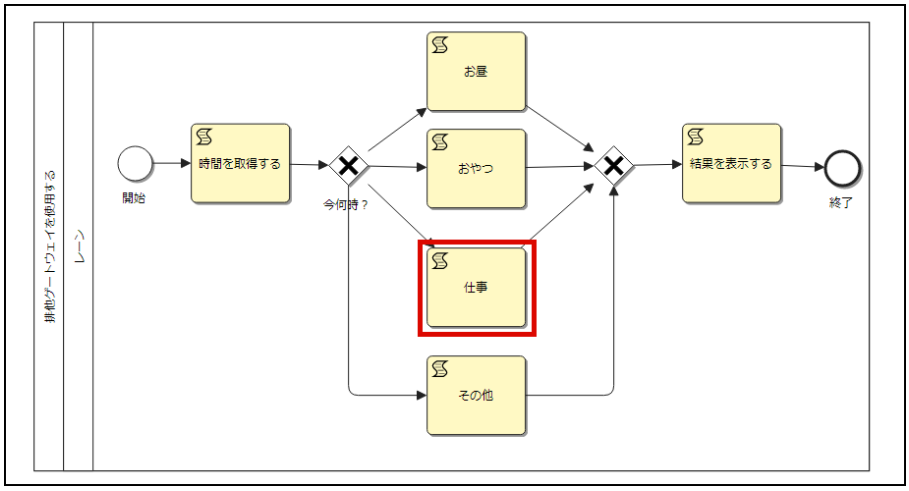
スクリプトタスク「おやつ」のプロパティにて、作業名 おやつ を変数 todo に格納するスクリプトを設定します。

```
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」

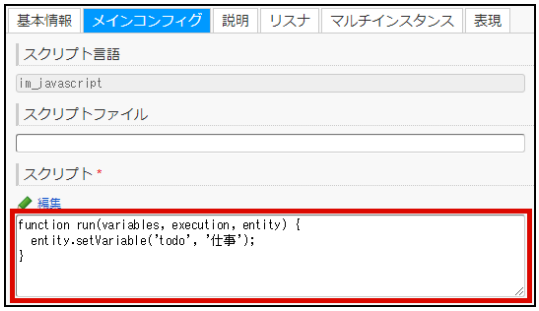
9. スクリプトタスク「仕事」 のプロパティを設定します。



図：スクリプトタスク「仕事」

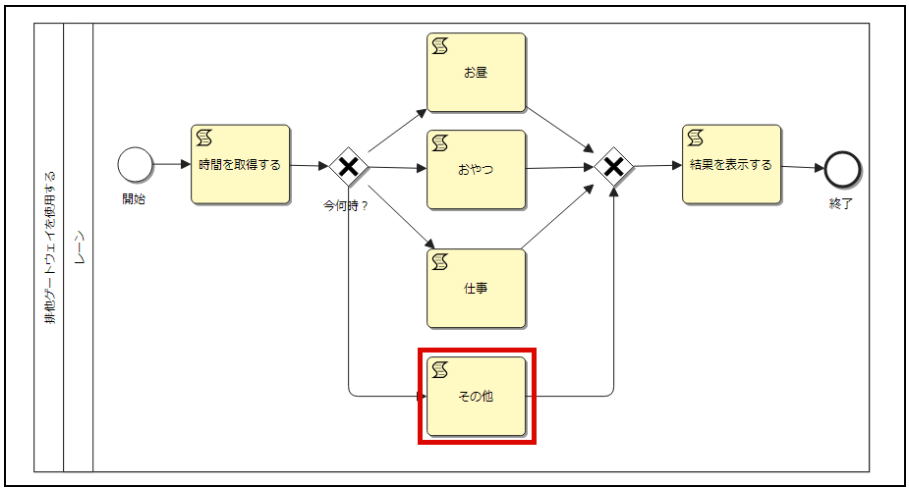
スクリプトタスク「仕事」のプロパティにて、作業名 仕事 を変数 todo に格納するスクリプトを設定します。

```
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」

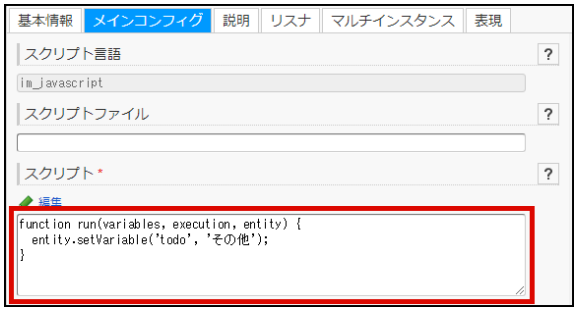
10. スクリプトタスク「その他」 のプロパティを設定します。



図：スクリプトタスク「その他」

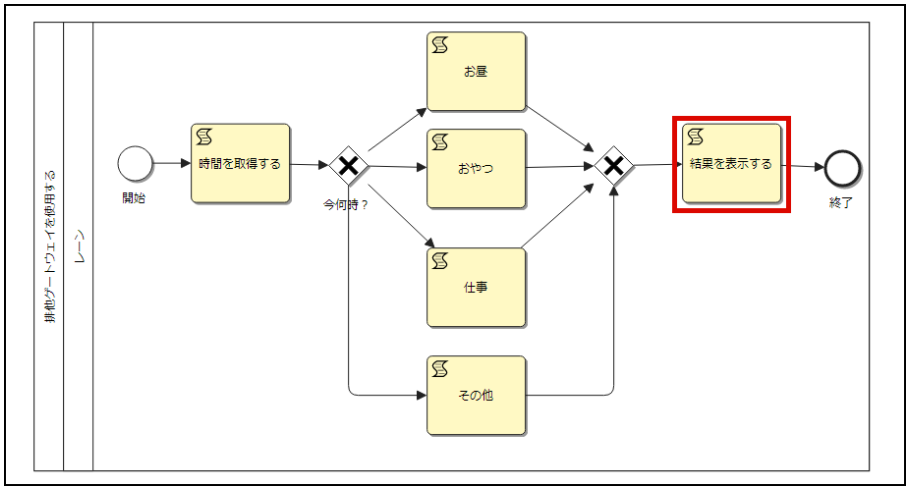
スクリプトタスク「その他」のプロパティにて、作業名 その他 を変数 todo に格納するスクリプトを設定します。

```
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。

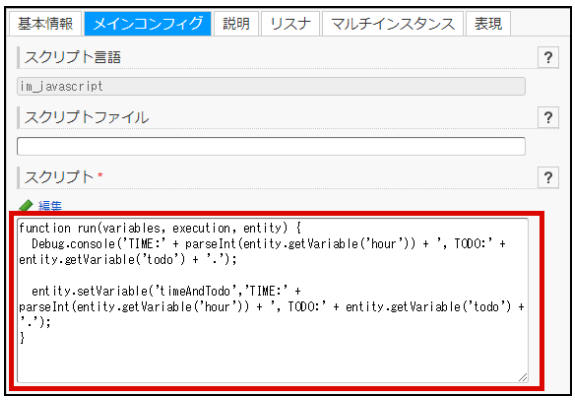


図：スクリプトタスク「結果を表示する」

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。

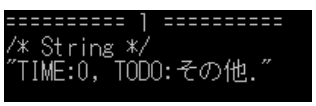
```
function run(variables, execution, entity) {
  Debug.console('TIME:' + parseInt(entity.getVariable('hour')) + ', TODO:' + entity.getVariable('todo') + '.');

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

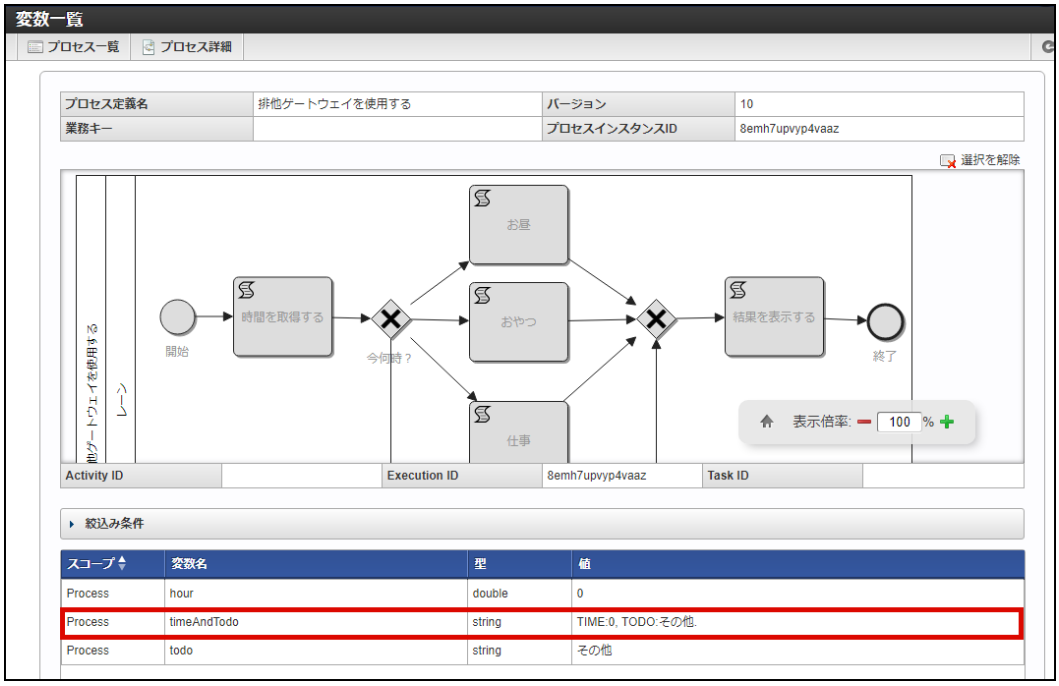


図：スクリプトタスク「結果を表示する」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

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 Designer 操作ガイド」 - 「ゲートウェイ」 - 「デフォルトフロー」も併せて参照してください。

IM-LogicDesignerタスク

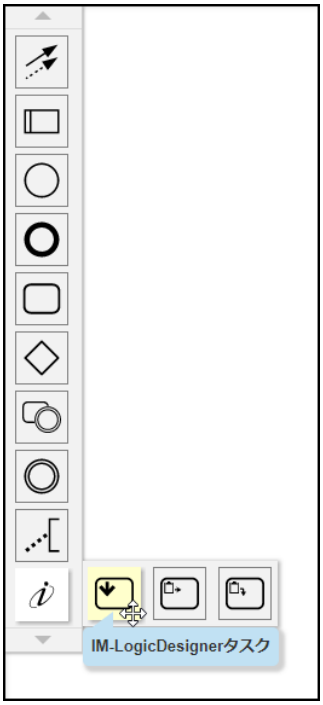
IM-LogicDesignerタスクを利用してユーザ情報を取得する

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することが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。
> **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することができます。
> アップロード手順は「IM-BPM Designer 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

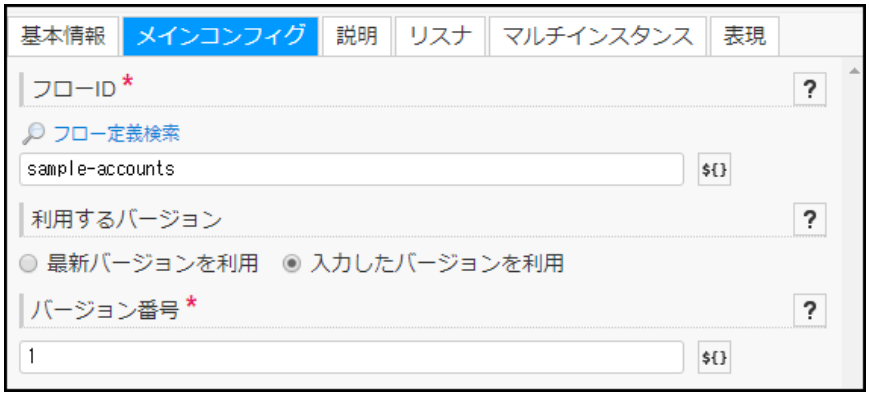
1. プロセスのプロパティ「データオブジェクト」に `user_cd = "aoyagi"` のデータプロパティを定義します。

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することができます。
   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」の選択と併せて「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することができます。
     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。

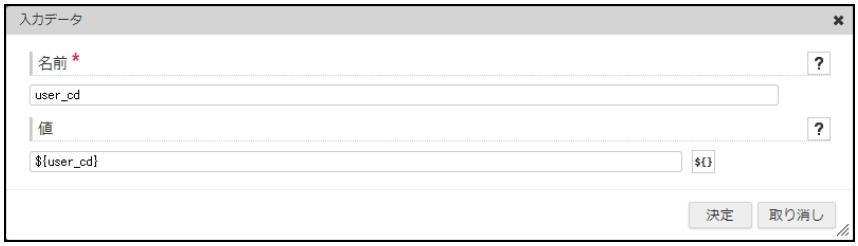
入力データを設定する

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。

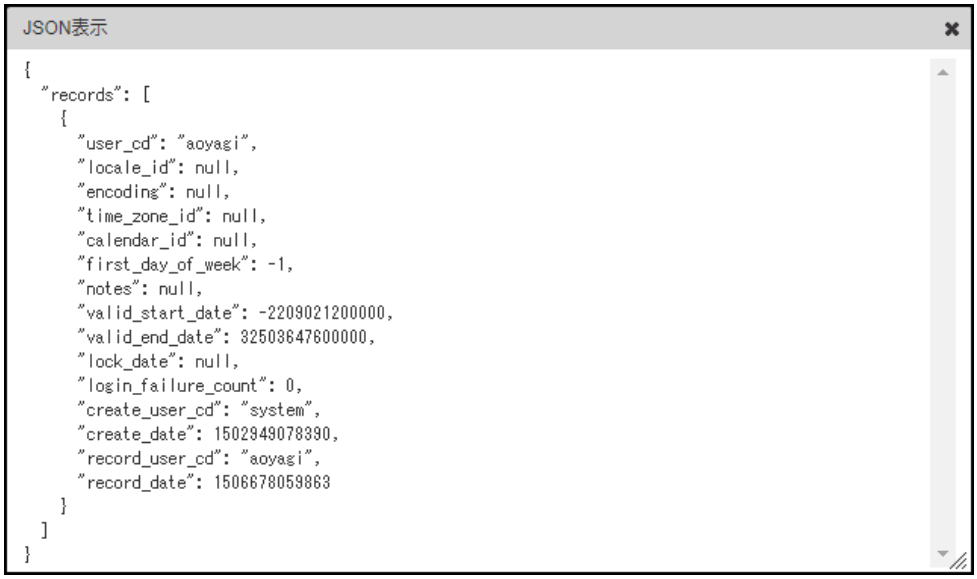
図：IM-LogicDesignerタスク：プロパティ - メインコンフィグ - 3

> **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 Designer 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することができます。
アップロード手順は「IM-BPM Designer 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

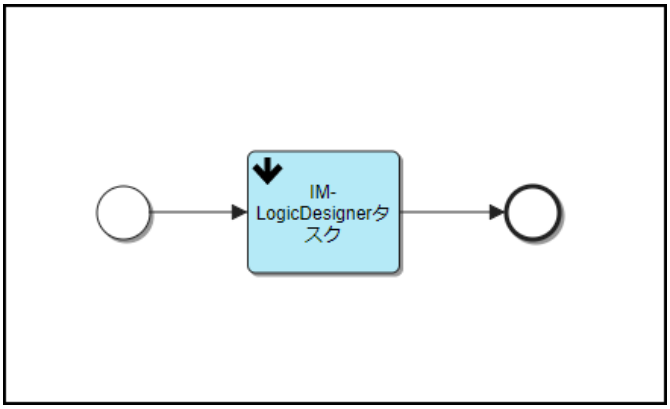
事前準備

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 Designer 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。

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。

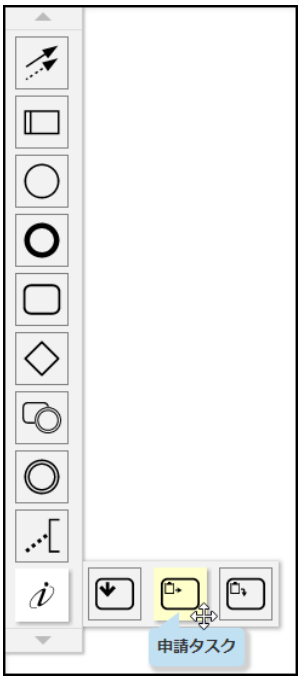
コラム

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することができます。
アップロード手順は「IM-BPM Designer 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。

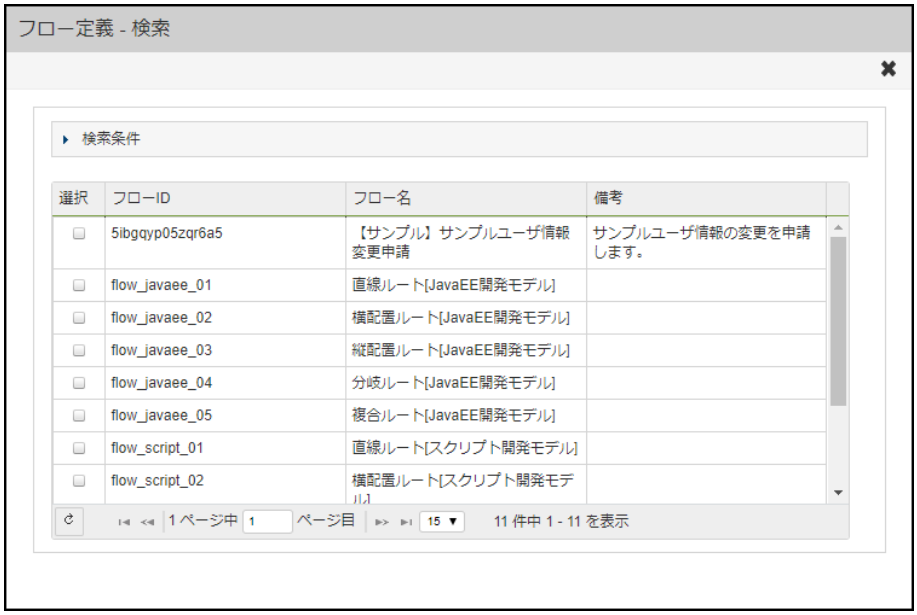
図：申請タスク：プロパティ - メインコンフィグ

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。

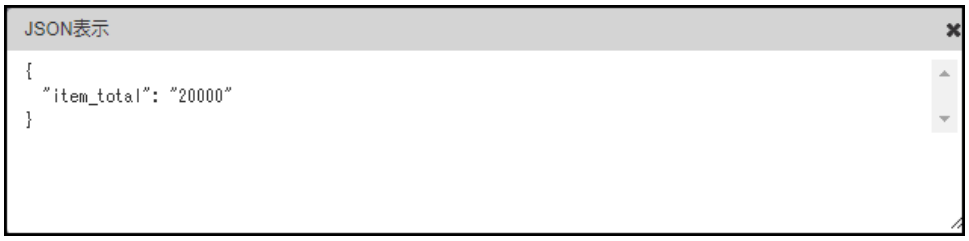
実行結果を確認する

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

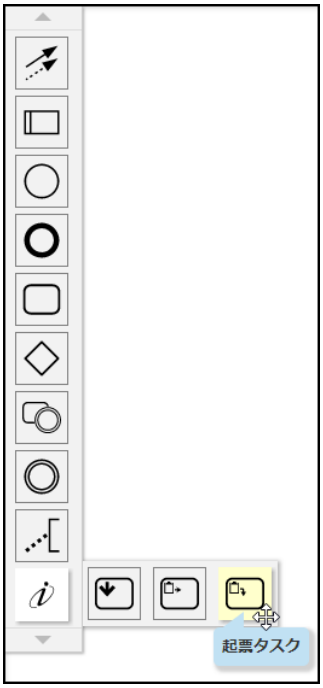
起票タスクを使用してワークフローと連携する

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。

> **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することができます。
> アップロード手順は「IM-BPM Designer 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。

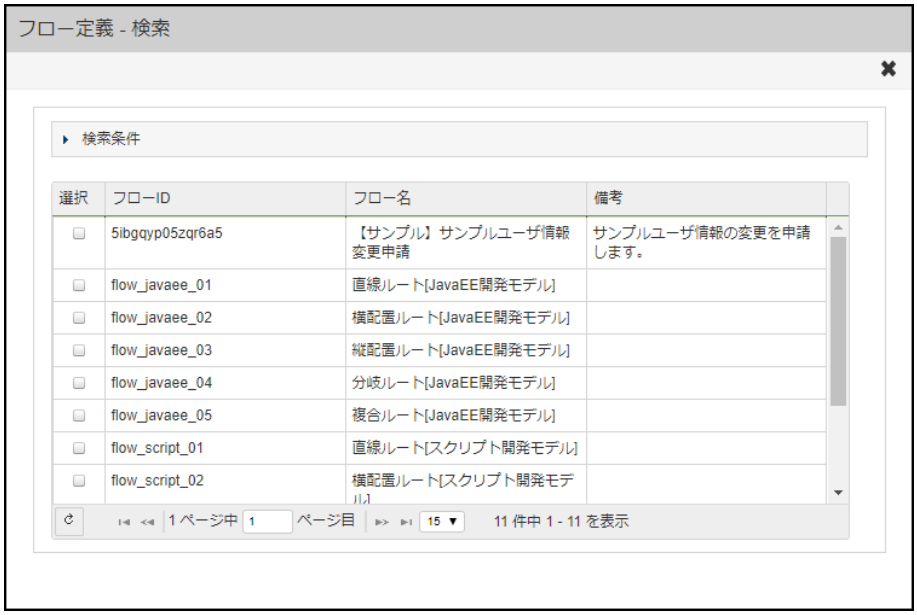
図：起票タスク：プロパティ - メインコンフィグ

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。

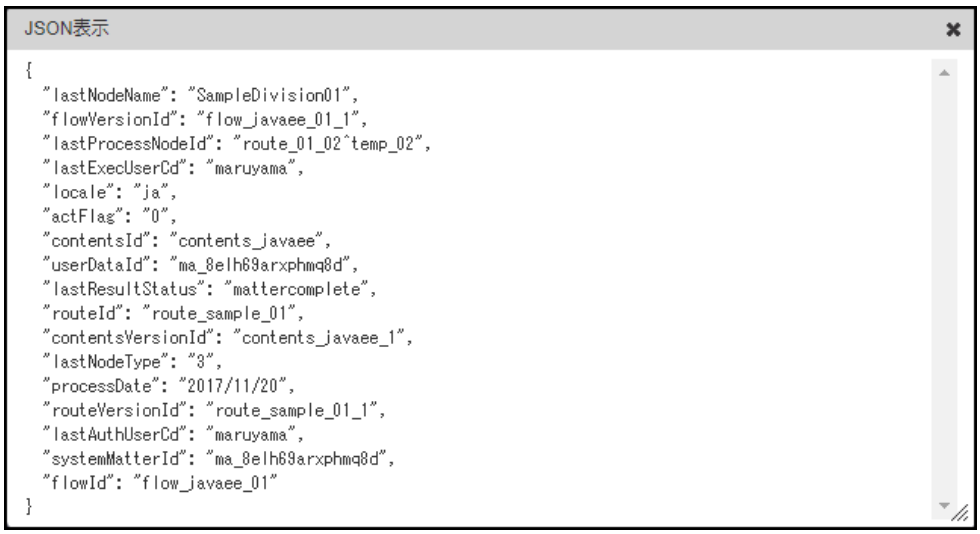
8. 「案件のユーザデータを格納する変数名」を設定します。

workflowData を設定します。

実行結果を確認する

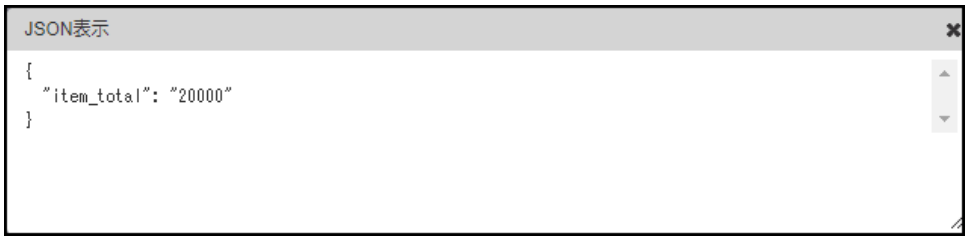
このチュートリアルで作成したプロセスを実行環境にデプロイし、実行した結果の確認を行います。

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

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

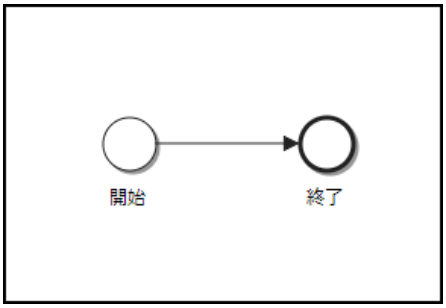
開始イベントのフォームにFormaアプリケーションを利用する

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することができます。
> アップロード手順は「IM-BPM Designer 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:` と入力し、その後に続けてFormaアプリケーションの「アプリケーションID」を入力することで、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できます。

> **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 Designer 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。

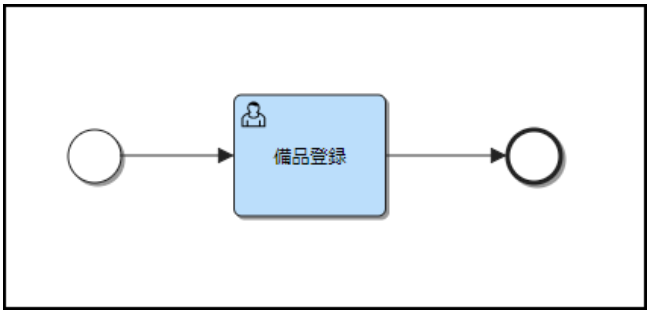
コラム

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することができます。
アップロード手順は「IM-BPM Designer 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

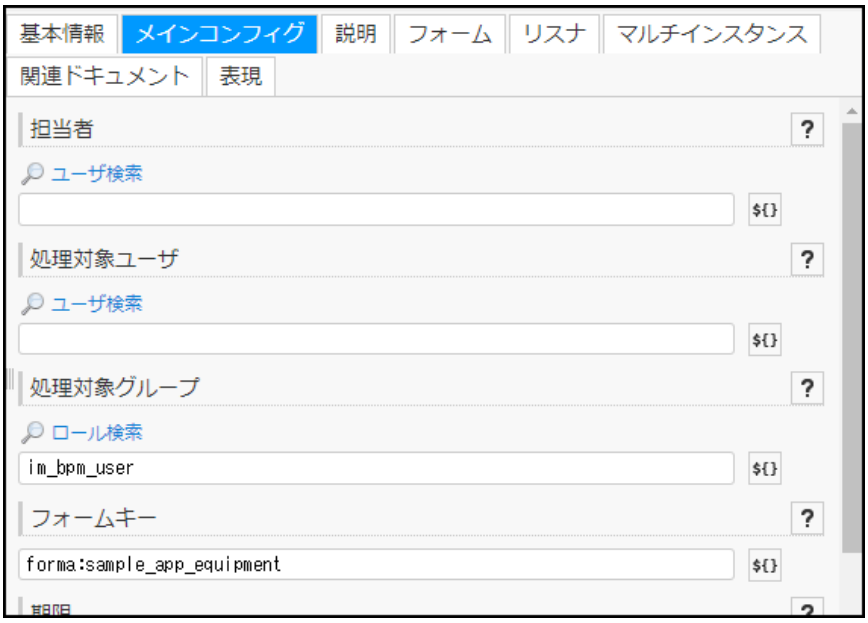
エレメントを配置する



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:` と入力し、その後に続けてFormaアプリケーションの「アプリケーションID」を入力することで、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できます。

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 Designer 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。