

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. オptionalタスク
 - 4.2.5.1. オptionalタスクを追加する
 - 4.2.6. プール、レーン
 - 4.2.6.1. プールとレーンを利用して作業の関係者を明確にする
 - 4.2.7. 強制終了イベント
 - 4.2.7.1. 強制終了イベントを利用して、プロセスを強制終了する
 - 4.2.8. イベントサブプロセス
 - 4.2.8.1. イベントサブプロセスを利用して実行中のプロセスを変更する
 - 4.2.9. サービスタスク
 - 4.2.9.1. e Builderで作成したJAVAクラスをサービスタスクで使用する
 - 4.2.10. コールアクティビティ
 - 4.2.10.1. コールアクティビティを使用する
 - 4.2.10.2. 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
 - 4.2.11. イベントゲートウェイ
 - 4.2.11.1. イベントゲートウェイを使用する
 - 4.2.12. パラレルゲートウェイ
 - 4.2.12.1. パラレルゲートウェイを使用する
 - 4.2.13. 排他ゲートウェイ
 - 4.2.13.1. 排他ゲートウェイを使用する
 - 4.2.14. 包括ゲートウェイ
 - 4.2.14.1. 包括ゲートウェイを使用する
 - 4.2.15. イベント
 - 4.2.15.1. シグナルイベントを使用する
 - 4.2.15.2. メッセージイベントを使用する
 - 4.2.15.3. タイマイベントを使用する
 - 4.2.15.4. エラーイベントを使用する
 - 4.2.16. IM-LogicDesignerタスク
 - 4.2.16.1. IM-LogicDesignerタスクを利用してユーザ情報を取得する
 - 4.2.16.2. 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

- 4.2.16.3. IM-LogicDesignerタスクを利用してOpenRulesを使用する
 - 4.2.17. 申請タスク
 - 4.2.17.1. 申請タスクを使用してワークフローと連携する
 - 4.2.18. 起票タスク
 - 4.2.18.1. 起票タスクを使用してワークフローと連携する
 - 4.2.18.2. 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
 - 4.2.19. IM-FormaDesigner for Accel Platform をユーザ入力フォームとして利用する
 - 4.2.19.1. 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
 - 4.2.19.2. 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
 - 5.3. プロセスの開始
 - 5.3.1. ジョブネットから営業日を指定してプロセスの開始を繰り返す
 - 5.3.1.1. プロセス定義を作成する
 - 5.3.1.2. IM-LogicDesignerを確認する
 - 5.3.1.3. ジョブネットの設定を確認する
 - 5.3.1.4. 結果を確認する
- 6. プロセスの分析
 - 6.1. Elasticsearch/ Kibana連携機能
 - 6.1.1. IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する
 - 6.1.1.1. Kibanaの導入
 - 6.1.1.2. プロセス実行とElasticsearchおよびKibanaの概要
 - 6.1.1.3. 本チュートリアルで使用する時系列データを作成する
 - 6.1.1.4. Elasticsearch に変数用のテンプレートを登録する
 - 6.1.1.5. データ作成用プロセスを実行する
 - 6.1.1.6. Elasticsearch のインデックスをKibanaから確認する
 - 6.1.1.7. Kibana上にインデックスパターンを作成する
 - 6.1.1.8. タスクの変数を使用して部署ごとの統計情報のグラフを作成する
 - 6.1.1.9. Kibana上にダッシュボードを作成する
 - 6.1.2. 円グラフを使用して情報の内訳を表示する
 - 6.1.2.1. 「Discover」でデータの検出を保存する
 - 6.1.2.2. 「完了タスク数のタスクごとの内訳」を円グラフで可視化する
 - 6.1.2.3. 「完了タスク数のタスク処理ユーザの内訳」を円グラフで可視化する
 - 6.1.2.4. 「完了タスク数のタスクとユーザの内訳」を2階層の円グラフで可視化する
 - 6.1.2.5. 「完了タスク数のタスクごとの内訳」、「完了タスク数のタスク処理ユーザの内訳」、「完了タスク数のタスクとユーザの内訳」をダッシュボードに追加する
 - 6.1.3. タスクの処理件数を地図上で可視化する
 - 6.1.3.1. 「Discover」でデータの検出を保存する
 - 6.1.3.2. 情報を地図上で可視化する
 - 6.1.3.3. 既存の可視化情報を修正して新しい可視化情報を作成する
 - 6.1.3.4. 可視化された情報をダッシュボードに追加する
 - 6.1.4. グラフ上にKPIを表示する
 - 6.1.4.1. Kibanaの「Timelion」機能を使用してグラフ上にKPIを表示する
 - 6.1.4.2. 「業務A_新規契約タスク完了数とKPI」をダッシュボードに追加する

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。
2019-04-01	第11版 下記を追加・変更しました。 「プロセスの分析」を追加しました。 「プロセスの分析」 - 「Elasticsearch/ Kibana連携機能」を追加しました。 「プロセスの分析」 - 「Elasticsearch/ Kibana連携機能」に下記のページを追加しました。 「IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する」 「はじめに」に「プロセスの分析」追加にともなう説明を追加しました。 「概要」に「プロセスの分析」追加にともなう説明を追加しました。 「タイマイイベントを使用する」 - 「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」に、「cron式で周期を指定する」コラムを追記しました。 「プロセスの作成」 - 「実用編」に下記のページを追加しました。 「e Builderで作成したJAVAクラスをサービスタスクで使用する」 「イベントゲートウェイを使用する」
2019-05-01	第12版 下記を追加・変更しました。 「プロセスの管理」 - 「プロセスの開始」を追加しました。 「ジョブネットから営業日を指定してプロセスの開始を繰り返す」を追加しました。
2019-08-01	第13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。 「イベントサブプロセスを利用して実行中のプロセスを変更する」 「エラーイベントを使用する」 「強制終了イベントを利用して、プロセスを強制終了する」 「オプションタスクを追加する」 「プロセスの分析」 - 「Elasticsearch/ Kibana連携機能」 - 「IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する」に追記・修正しました。 「プロセスの分析」 - 「Elasticsearch/ Kibana連携機能」に下記のページを追加しました。 「円グラフを使用して情報の内訳を表示する」 「タスクの処理件数を地図上で可視化する」 「グラフ上にKPIを表示する」

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 プロセスデザイナー 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あわ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](#)
- [Elasticsearch連携機能](#)

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。

- [プロセスの分析](#)

プロセスの分析を行う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、システムに必要な画面と簡単に連携できます。
- IM-FormaDesigner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せます。
- 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せます。

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や、プロセスの分析の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。）
 - IM-BPM/Elasticsearch コネクタモジュール（プロセスの分析編を行う場合必要です。）
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。

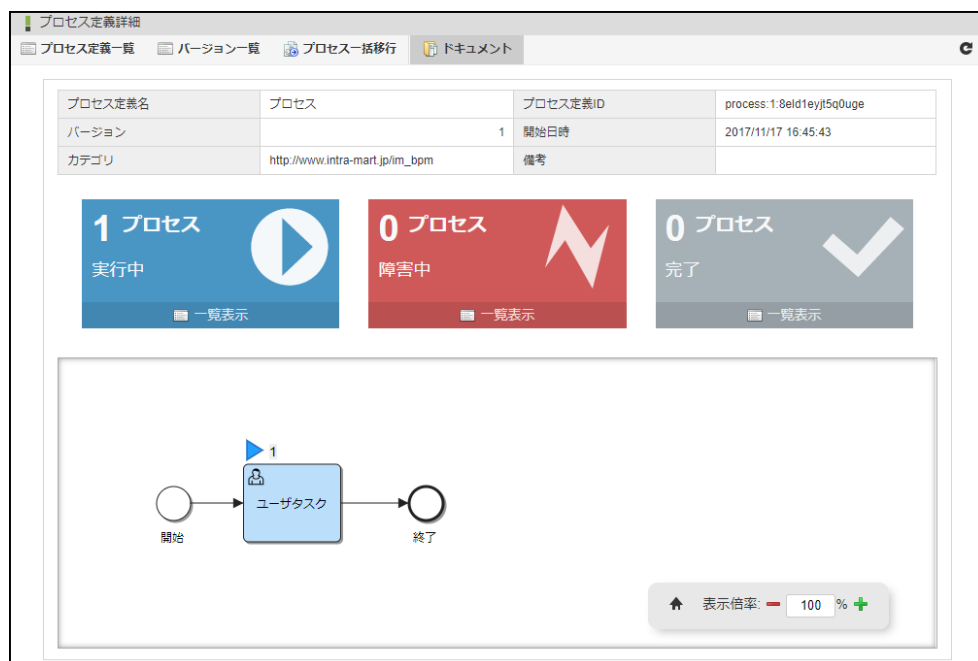
基礎編

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。



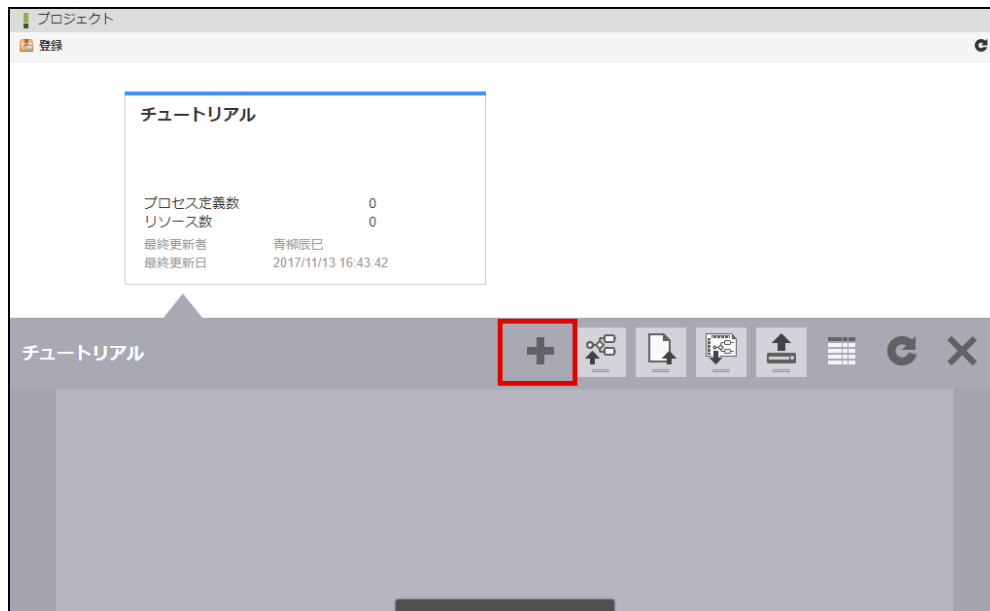
図：プロセス定義詳細

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。



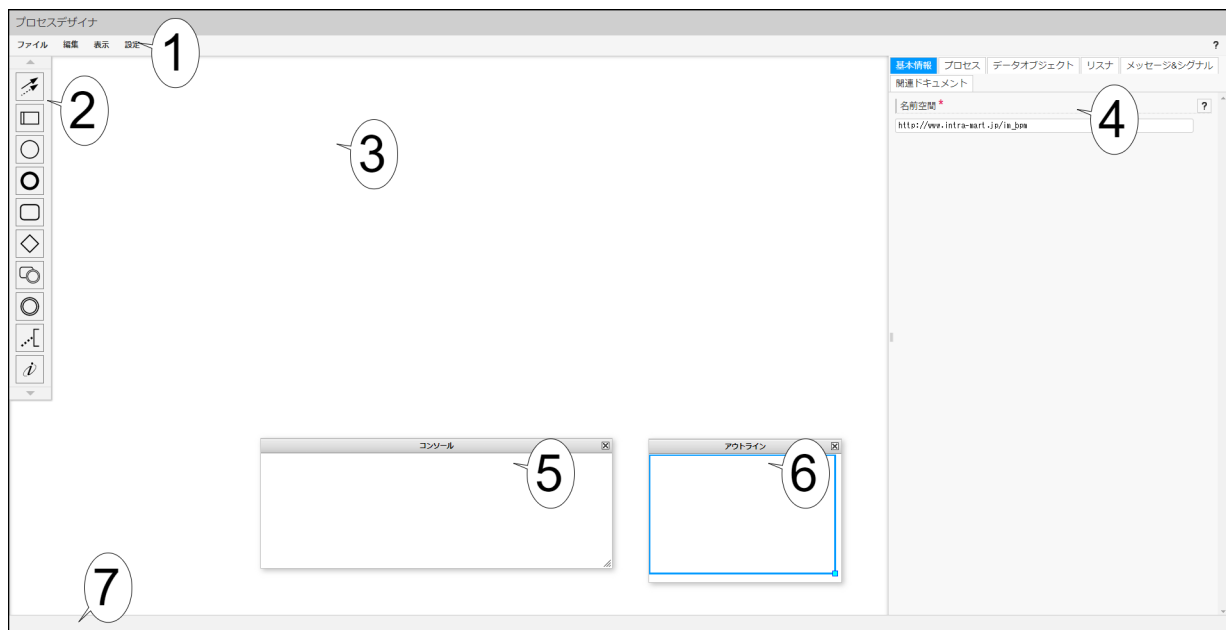
図：プロジェクト詳細エリア - 新規登録



コラム

プロジェクトの作成や認可設定については「IM-BPM プロセスデザイナー 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

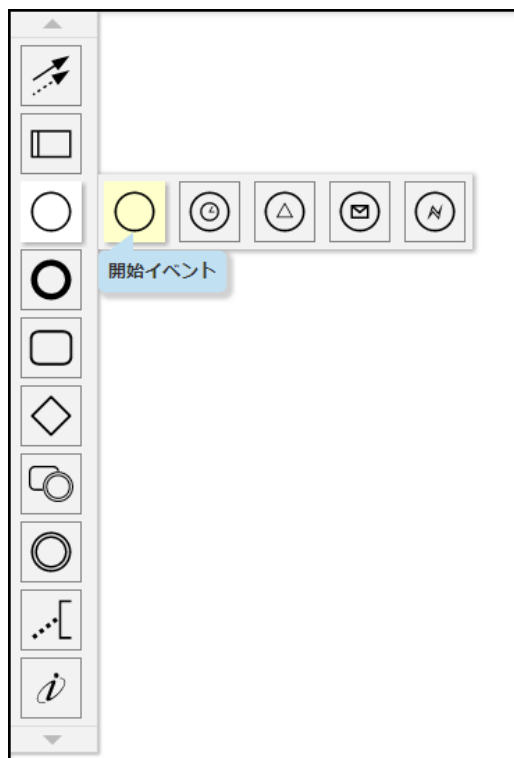
プロセス全体を俯瞰表示する領域です。
表示領域の移動や、表示の拡大縮小ができます。

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。

基本情報 プロセス データオブジェクト リスナ メッセージ&シグナル

関連ドキュメント

プロセス定義キー * ?

process_201711011645_20171113173122570

名前 ?

プロセス_20171113173122570

処理対象ユーザ ?

ユーザ検索

aoyagi

処理対象グループ ?

ロール検索

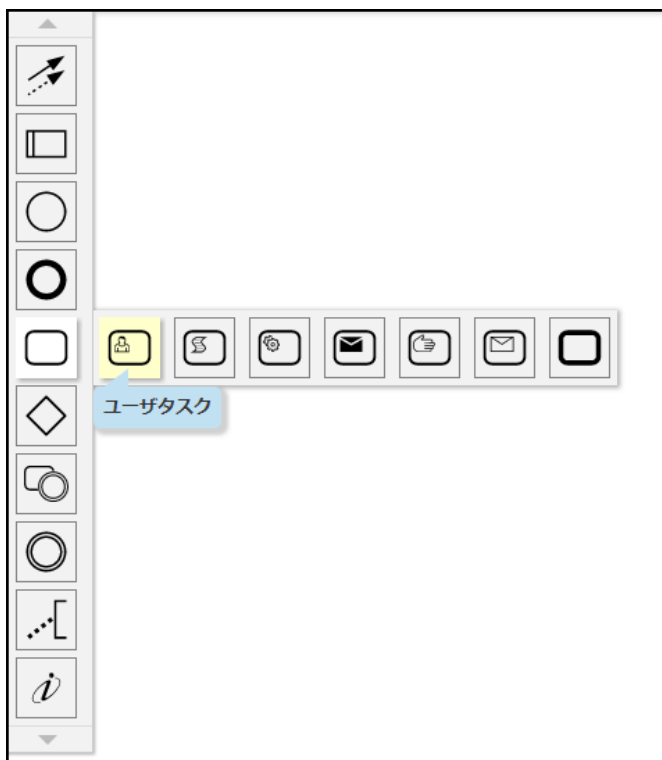
☐ ユーザに業務キーの設定を許可しない ?

説明 ?

編集

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

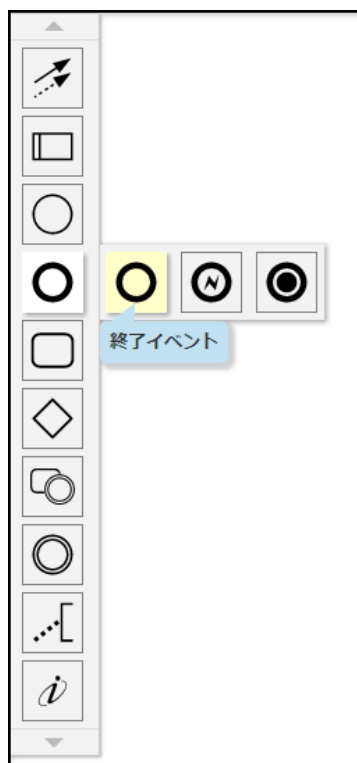
8. 「プロパティエリア」にて、「ユーザタスク」の「処理対象ユーザ」を設定します。

タスクを処理するユーザを設定してください。

The screenshot shows the 'Main Configuration' tab of the 'User Task Properties' dialog. The 'Target User' field is highlighted with a red box and contains the value 'aoyagi'. Other fields include 'Assignee', 'Target Group', 'Form Key', 'Period', 'Priority', 'Category', and 'Skip Condition', each with a search icon and a placeholder text '\$()'. The 'Target User' field also has a search icon and a placeholder text '\$()'.

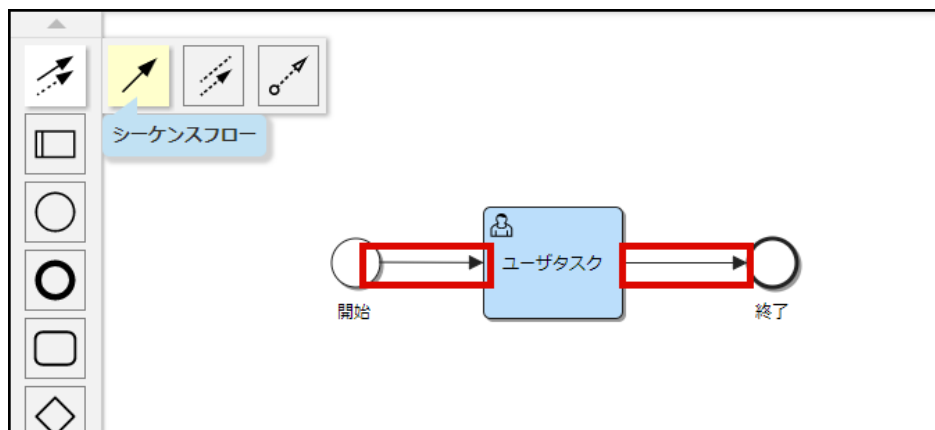
図：ユーザタスク - プロパティ - メインコンフィグ - 処理対象ユーザ

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。



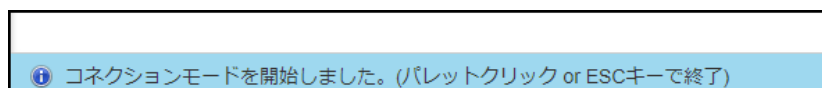
図：パレット - 終了イベント - 終了イベント

10. 「シーケンスフロー」で各エレメントを接続します。



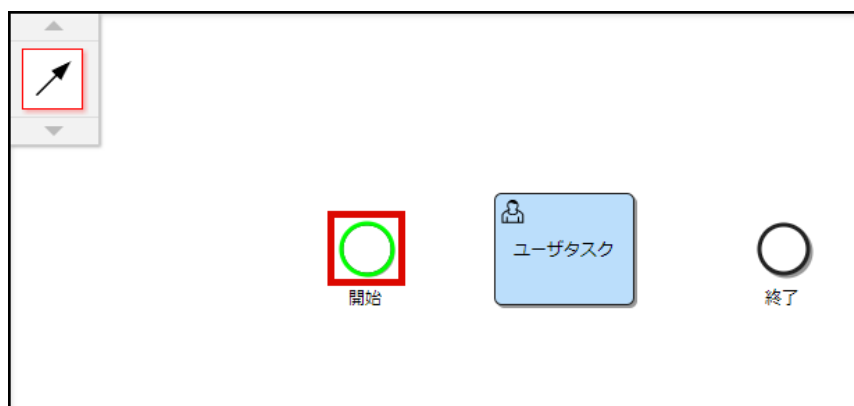
図：パレット - コネクション - シーケンスフロー

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。



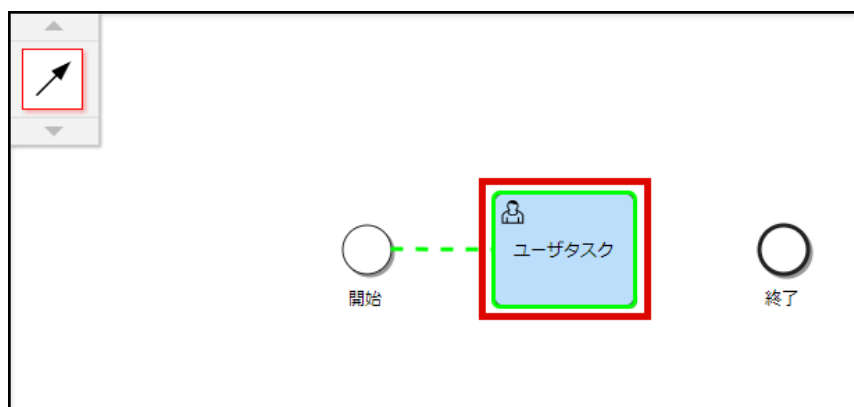
図：フッタ

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、ドラッグして接続を開始します。
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。

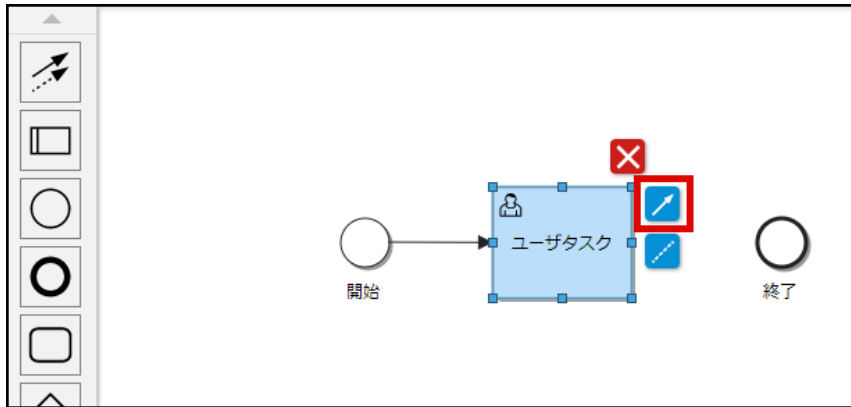


図：接続する

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。

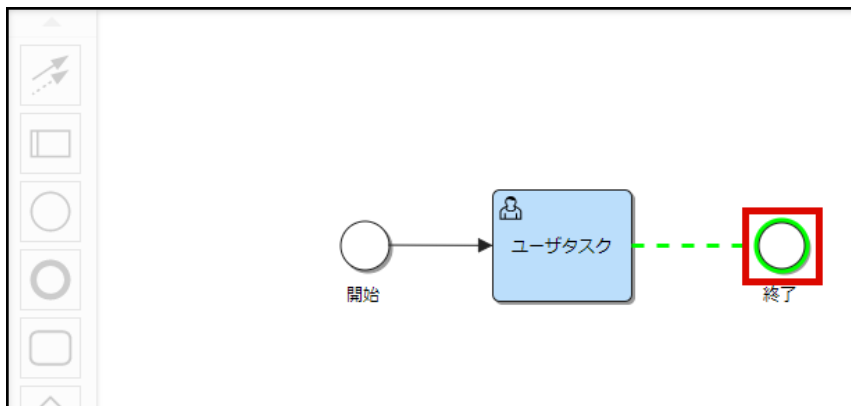
6. キャンバス上の「ユーザタスク」を選択します。

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。



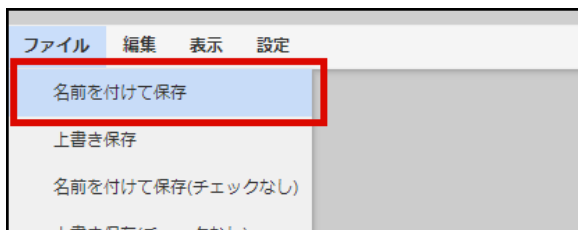
図：接続を開始する

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

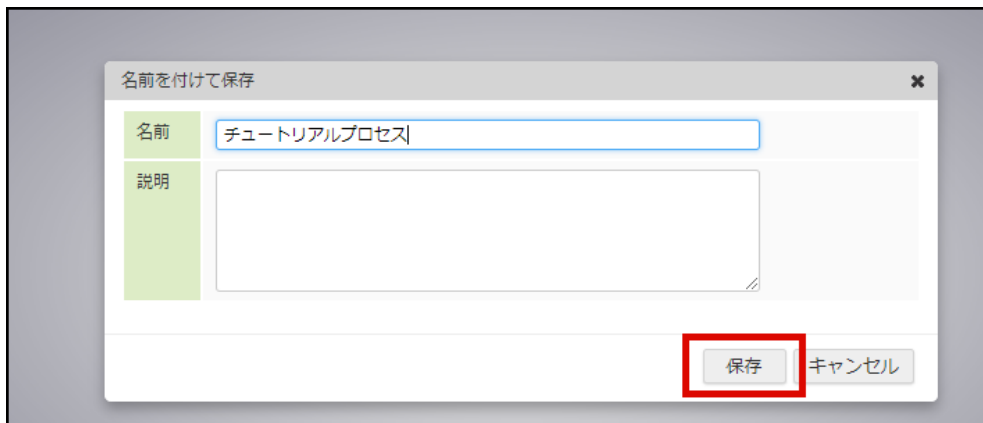
「名前を付けて保存」と「名前を付けて保存（チェックなし）」の違い

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。

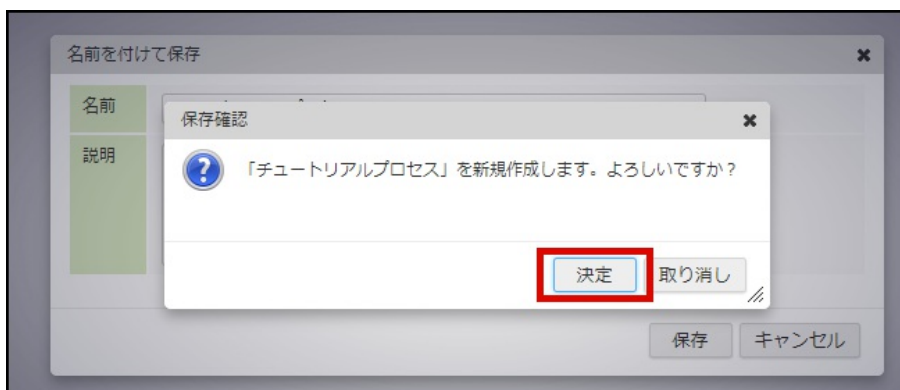
「名前を付けて保存（チェックなし）」を実行した場合は上記チェックを行わず、保存が可能となる最低限のチェックのみを行って保存できます。

12. 「名前」を入力して「保存」をクリックします。



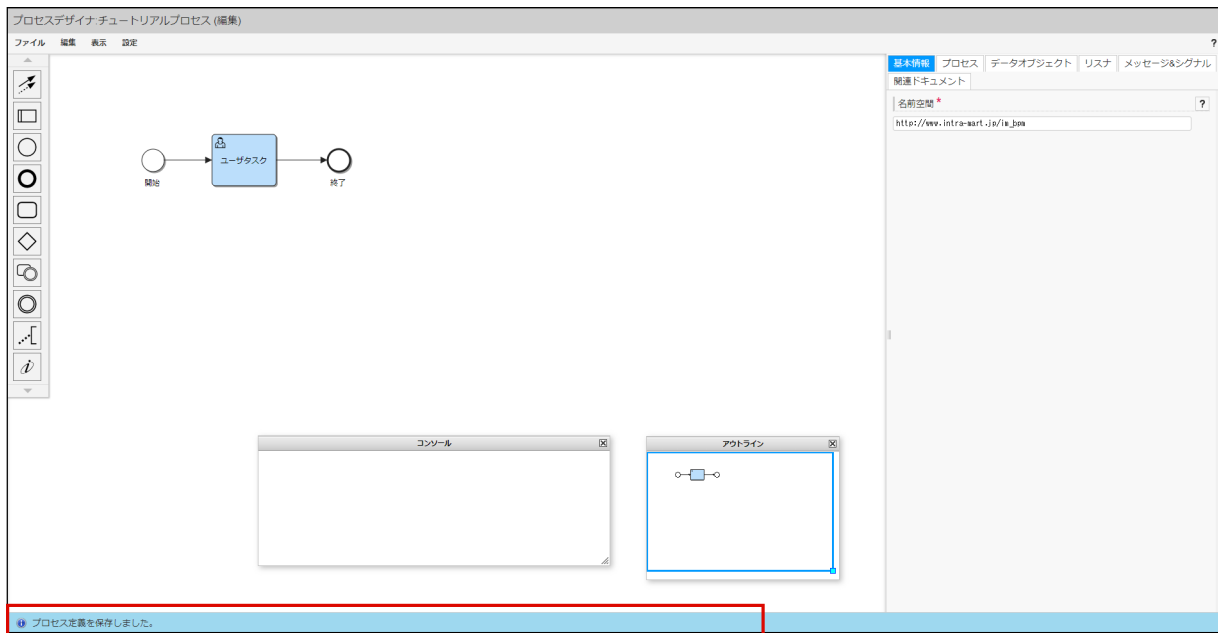
図：名前を付けて保存

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。



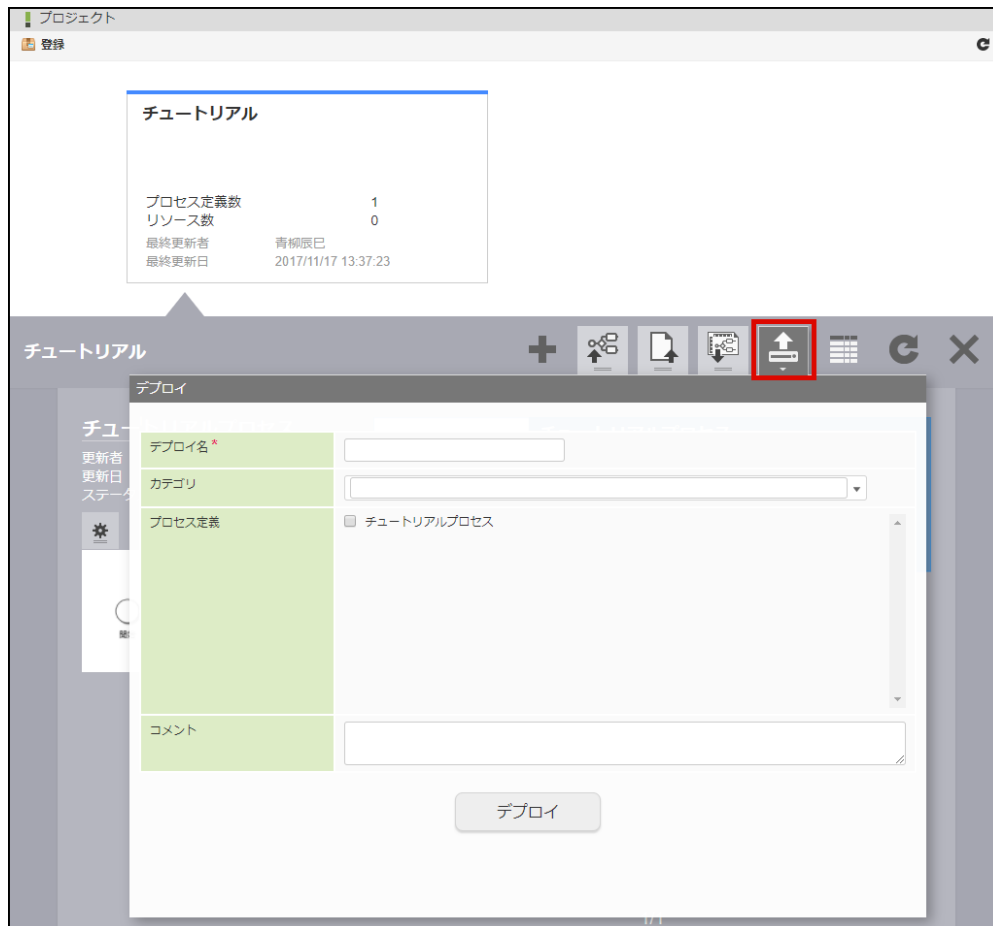
図：プロセス保存

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」→「プロセスデザイナー」をクリックします。

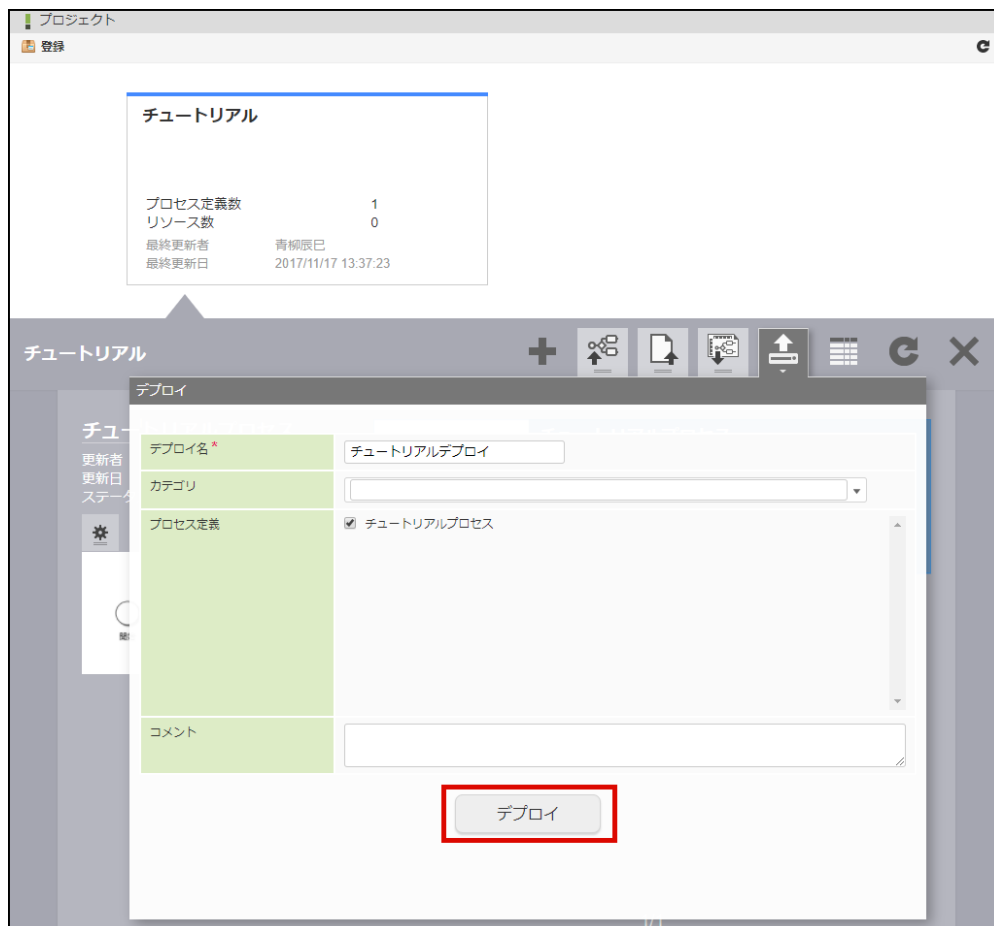
2. プロジェクトを選択し、 をクリックしデプロイエリアを開きます。



図：デプロイエリア

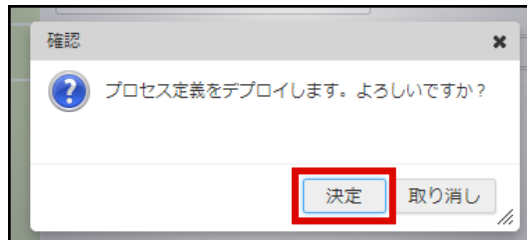
3. 「デプロイ名」の入力と「プロセス定義」の選択をします。
4. 「デプロイ」をクリックします。

※ デプロイについては「[IM-BPM プロセスデザイナー 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。



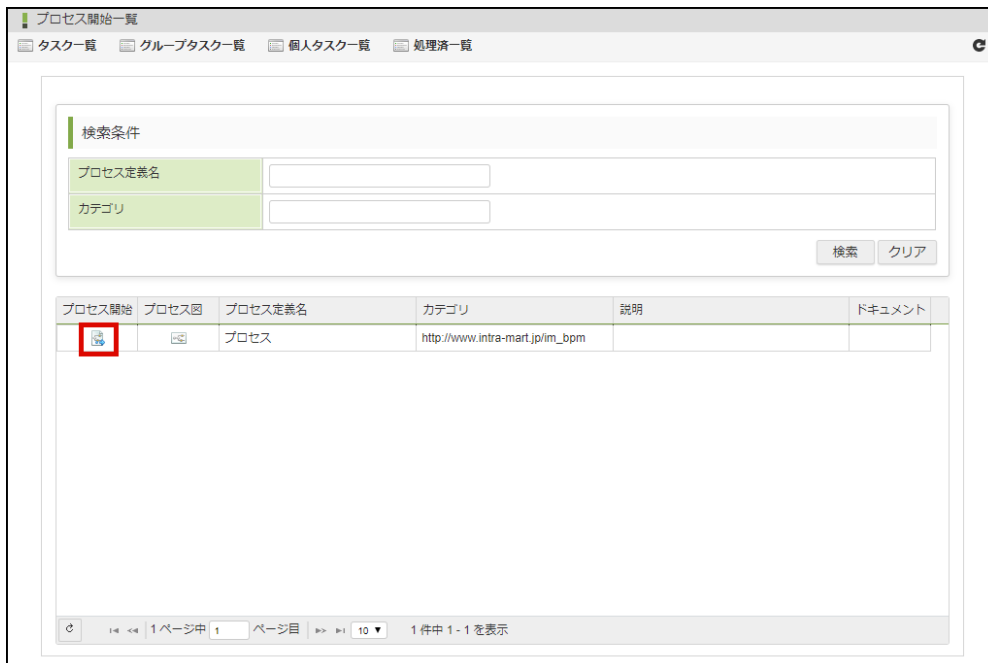
図：デプロイ確認

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
- オptionalタスク
- プール、レーン
- 強制終了イベント
- イベントサブプロセス
- サービスタスク
- コールアクティビティ
- イベントゲートウェイ
- 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](#)

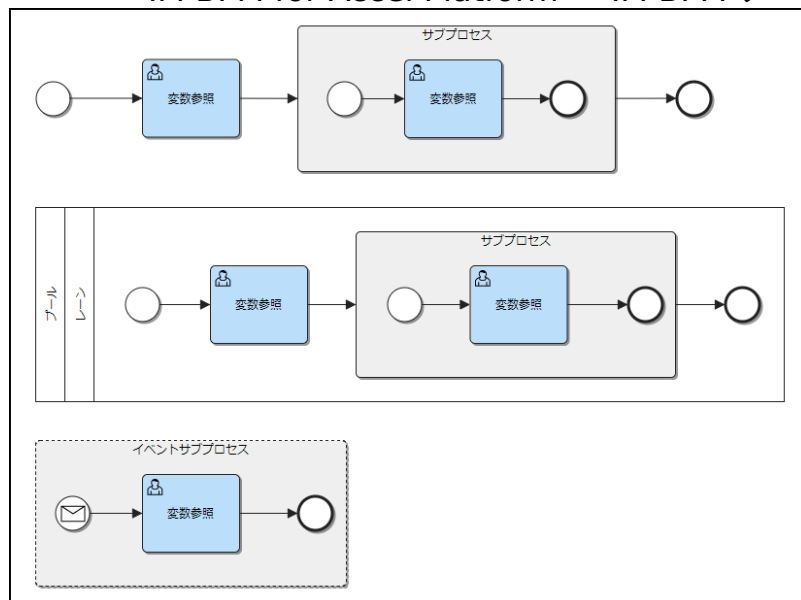
このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。

アップロード手順は「[IM-BPM プロセスデザイナー 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。

The screenshot shows the 'Data Property' dialog box. It has four main sections: ID, Name, Type, and Value. Each section has a text input field and a help icon (?). The ID field contains 'process01_var01'. The Name field contains 'var01'. The Type field is a dropdown menu currently showing 'string'. The Value field contains 'プロセス01_変数01_初期値'. At the bottom right, there are two buttons: '決定' (OK) and '取り消し' (Cancel).

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。 <table><tr><th>型名</th><th>説明</th></tr><tr><td>string</td><td>文字列</td></tr><tr><td>boolean</td><td>真偽値</td></tr><tr><td>datetime</td><td>日時（ISO8601拡張形式）</td></tr><tr><td>double</td><td>8バイト倍精度浮動小数点数</td></tr><tr><td>int</td><td>4バイト整数（-2147483648～2147483647）</td></tr><tr><td>long</td><td>8バイト整数（-9223372036854775808～9223372036854775807）</td></tr></table>	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）
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。

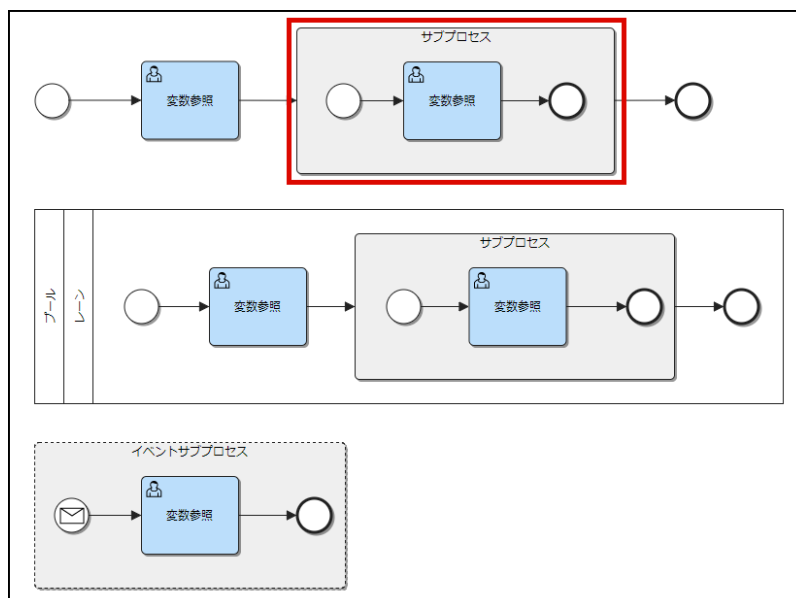
6. 同様の手順で、以下の変数も定義します。

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。

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

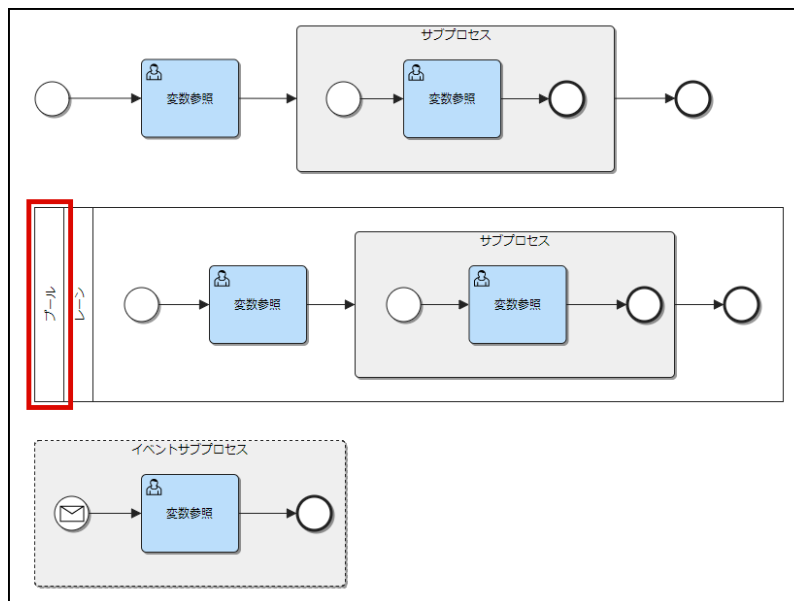
図：サブプロセス：プロパティ - データオブジェクト

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。



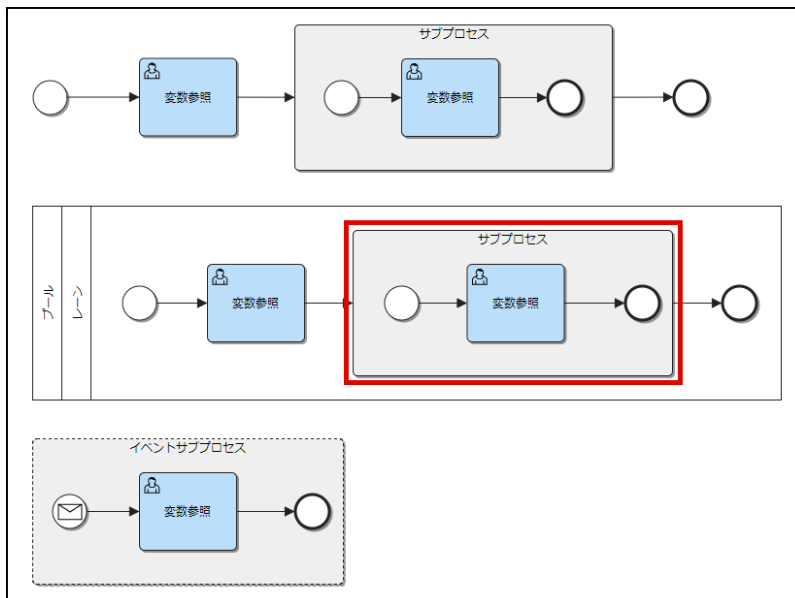
図：プール：プロパティ - データオブジェクト

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。



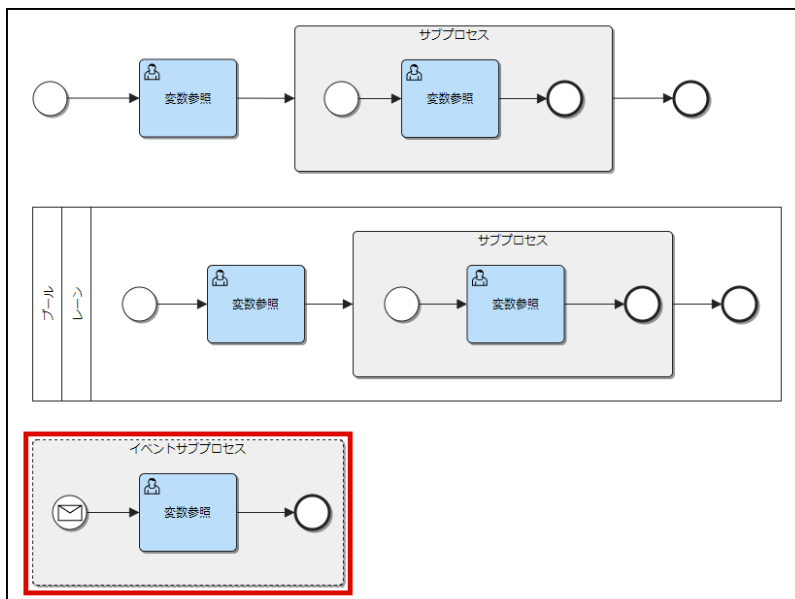
図：サブプロセス：プロパティ - データオブジェクト

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

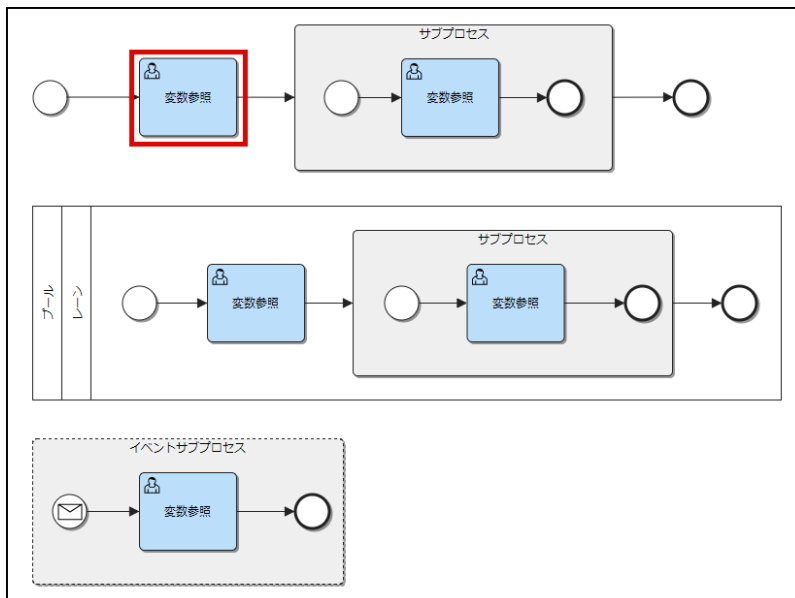
2. 以下の通り、データプロパティを定義します。

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。



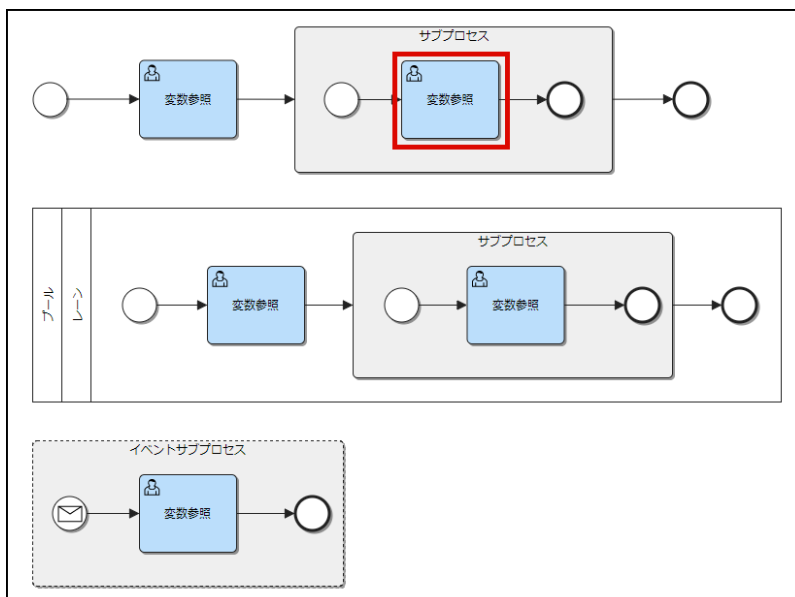
図：EL式エディタ - データプロパティツリー



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

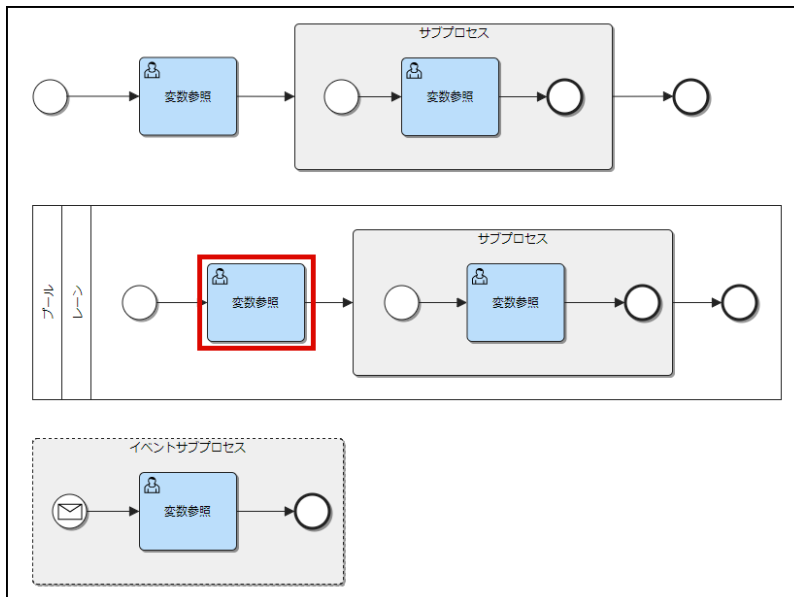
i コラム

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。

i コラム

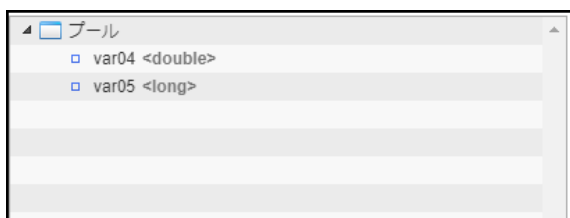
プロセスグローバルの定義とサブプロセスの定義で重複して命名されている変数「var01」を参照する場合、サブプロセスで定義された変数が参照されます。
つまり、直近の階層で定義された変数が優先的に参照されます。

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。

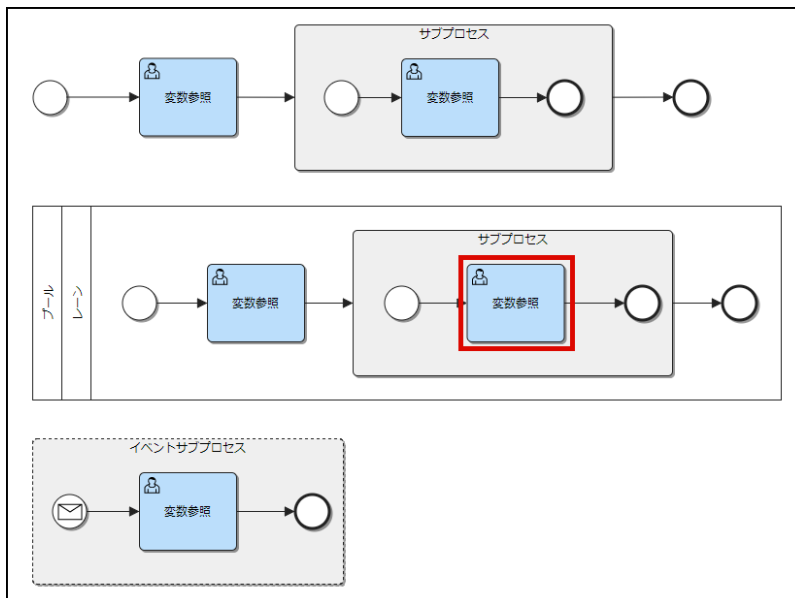


図：EL式エディタ - データプロパティツリー

i コラム

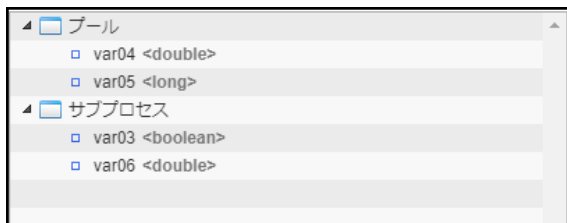
プールは一見、プロセスの中に配置されているように見えますが、そうではありません。
プールは一つのプロセス定義として扱われるため、プロセスの子要素になることはありません。
プロセス定義を跨って変数を参照することはできないため、プロセスで定義されたデータプロパティは参照できません。

11. 図の赤枠で示した「ユーザタスク」を選択します。



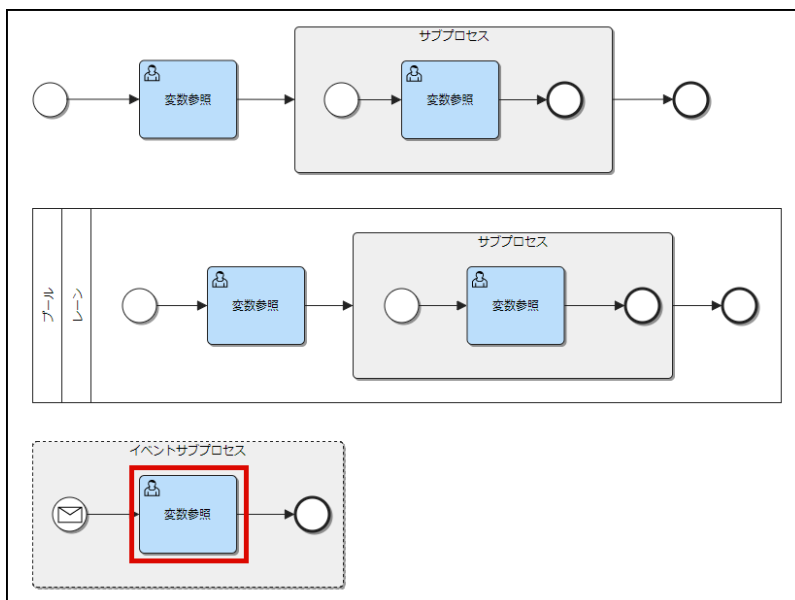
図：ユーザタスクの選択

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 プロセスデザイナー 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。

i 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 プロセスデザイナー 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
ループ回数が配列を選択します。
- 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：実行時のイメージ

1. 任意の方法で、`multi_length` の変数に回数を設定します。



コラム

本チュートリアルサンプルでは、「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：プロパティの設定

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：実行時のイメージ

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。

コラム

本チュートリアルサンプルでは、「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。

The screenshot shows the 'マルチインスタンス' (Multi-Instance) configuration dialog. The '繰り返しの種別' (Iteration Type) is set to '配列' (Array). The '配列' (Array) section shows the EL expression '\$multi_list'. Other options like 'ループ回数' (Loop Count), '順次実行' (Sequential Execution), '要素変数' (Element Variable), and '終了条件' (End Condition) are also visible.

図：プロパティの設定

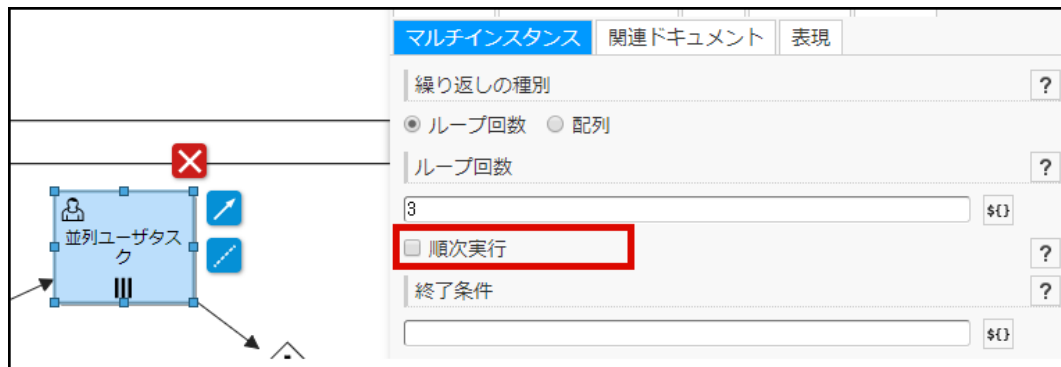
並列、または、順次で実行する

並列、または、順次で実行します。



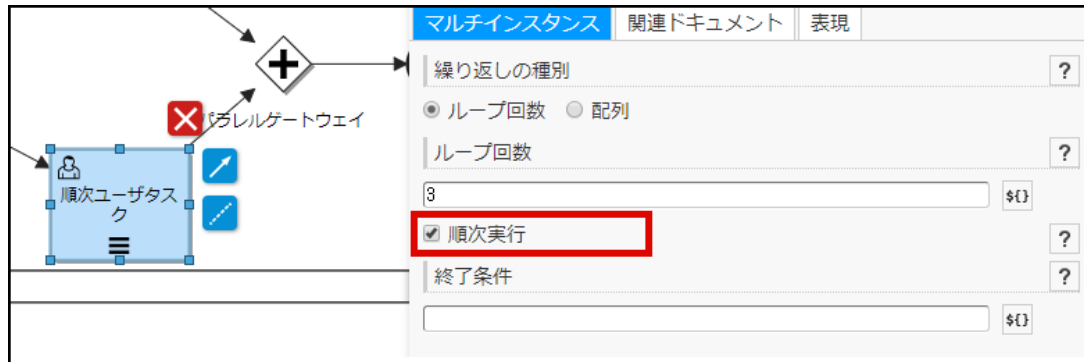
図：ユーザタスク到達時イメージ

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。



図：並列

2. 順次で実行する場合、「順序実行」のチェックボックスを有効にします。



図：順次

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：実行時のイメージ

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。

i コラム

※ 本チュートリアルサンプルでは、「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。

マルチインスタンス 関連ドキュメント 表現

繰り返しの種別 ?

☐ ループ回数 ☒ 配列

配列 ?

☐ 順次実行 ?

要素変数 ?

終了条件 ?

図：プロパティの設定 - マルチインスタンス

- 「配列変数」に変数名 `taskInfo` を設定します。
- プロパティの「基本情報」タブから「別名」にEL式を使用して、`taskInfo.taskName` を指定します。

基本情報 メインコンフィグ 説明 フォーム リスナ

マルチインスタンス 関連ドキュメント 表現

ID * ?

名前

要素ユーザタスク

別名 ?

実行モード ?

☒ 同期 ☐ 非同期

制御モード ?

☒ 排他 ☐ 非排他

図：プロパティの設定 - 基本情報

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

基本情報 メインコンフィグ 説明 フォーム リスナ

マルチインスタンス 関連ドキュメント 表現

担当者 ?

処理対象ユーザ ?

処理対象グループ ?

図：プロパティの設定 - メインコンフィグ

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移移します。

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 プロセスデザイナー 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。

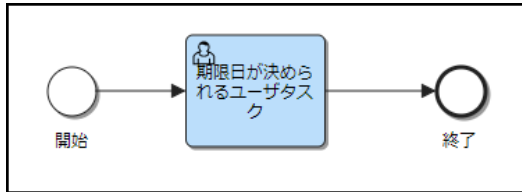
このチュートリアルでは、基準日を現在日に設定し、10日後の営業日を期限日に設定するようにします。

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. 開始イベントを設置します。
2. 「データプロパティ」を設定します。
キャンパスの空白部分をクリックし、「プロパティ」をプロセス全体に切り替えます。
「データオブジェクト」タブから、「データプロパティ」の「追加」リンクをクリックします。
3. 「データプロパティ」ダイアログから、以下のとおりに項目を設定します。
 - ID : dueDateDays
 - 名前 : dueDateDays
 - 型 : int
 - 値 : 10

図：データプロパティ

コラム

プロセスのデータプロパティ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。

4. 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」 - 「プロパティ」 - 「リスナ」

5. 以下の手順で、実行リスナを設定します。

1. タイプを **IM-LogicDesigner** に設定します。

実行リスナ

イベント ?

start ▼

タイプ ?

☐ Javaクラス
 ☐ 式
 ☐ デリゲート
 ☒ **IM-LogicDesigner**

フローID * ?

フロー定義検索

❌ この項目は必須です。

最新バージョン ?

☐ 最新バージョンを利用
 ☒ 入力したバージョンを利用

バージョン番号 * ?

❌ この項目は必須です。

入力データ ?

+ 追加
 選択済みの項目を削除

選択	編集	名前	値
☐			

☐ 結果変数を格納する ?

結果変数名 ?

決定 取り消し

図：「実行リスナ」

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 `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。

3. 最新バージョンを利用するように設定します。

実行リスナ

イベント ?

start ▼

タイプ ?

Javaクラス 式 デリゲート **IM-LogicDesigner**

フローID * ?

フロー定義検索

businessDateCalculation \${}

最新バージョン ?

☒ 最新バージョンを利用 ☐ 入力したバージョンを利用

入力データ ?

+ 追加 選択済みの項目を削除

選択	編集	名前	値
☐		結果変数を格納する	?

結果変数名 ?

決定 取り消し

図：「実行リスナ」

4. 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}`

入力データ

名前 * ?

referenceDate

値 ?

`${execution.getEngineServices().getProcessEngineConfiguration().getClock().getcurrentTime()}` \${}

決定 取り消し

図：基準日の設定

入力データ

名前 * ?

days

値 ?

`${dueDateDays}` \${}

決定 取り消し

図：日数の設定



コラム

上記で設定したEL式は、ロジックフローの入力設定に対応しています。

図：ロジックフロー側の対応する入力設定

5. 「結果変数を格納する」を有効にします。
6. 結果変数名に `businessDateCalculationResult` を設定します。

図：「実行リスナ」

6. 「ユーザタスク」に期限を設定します。
「ユーザタスク」の「メインコンフィグ」タブで、「期限」に「`${businessDateCalculationResult.addedBusinessDate}`」を入力します。

基本情報 | **メインコンフィグ** | 説明 | フォーム | リスナ

マルチインスタンス | 関連ドキュメント | 表現

担当者 ?

ユーザ検索
aoyagi \${{}}

処理対象ユーザ ?

ユーザ検索
\${{}}

処理対象グループ ?

ロール検索
\${{}}

フォームキー ?

\${{}}

期限 ?

\${businessDateCalculationResult.addedBusinessDate} \${{}}

優先度 ?

\${{}}

カテゴリ ?

\${{}}

スキップ条件 ?

\${{}}

図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

コラム

上記の期限で設定したEL式は、ロジックフローの出力設定に対応しています。

出力 + string + integer + date + object

boolean + 追加 配下に配置する JSON入力

配列型にする キー名を変更 型を変更 削除 全削除

addedBusinessDate <date>

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

個人タスク

検索条件

表示設定

処理	履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当を外
☐		IM-LogicDesigner!			期限日が決められ	50	2018/03/16 16:00	2018/03/29 16:00		

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 プロセスデザイナー 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

i 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 プロセスデザイナー 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。

i 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](#)」

i 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](#)」

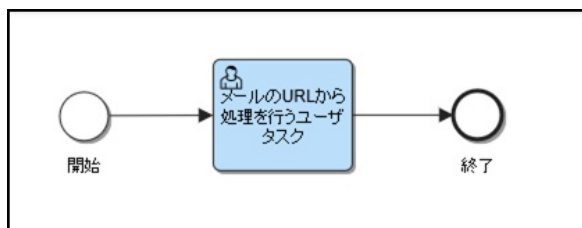
- プロセス定義を作成する
- 実行結果を確認する

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。

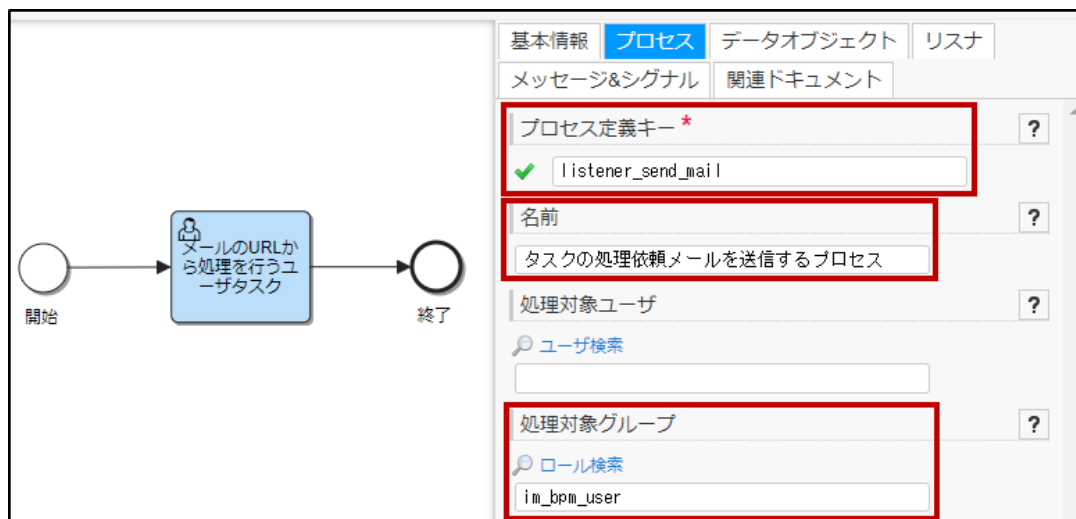
このチュートリアルでは、画面から選択されたタスク処理者に対して、タスク処理画面のURLをメール送信します。

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、「プロセス」タブの項目を以下のとおりに設定します。
 - 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：プロパティ全体 - 「プロパティ」 - 「プロセス」

3. 「開始イベント」の「メインコンフィグ」タブから、「フォームキー」を設定します。

- フォームキー: `forma:select_usercd`



図：「開始イベント」 - 「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」

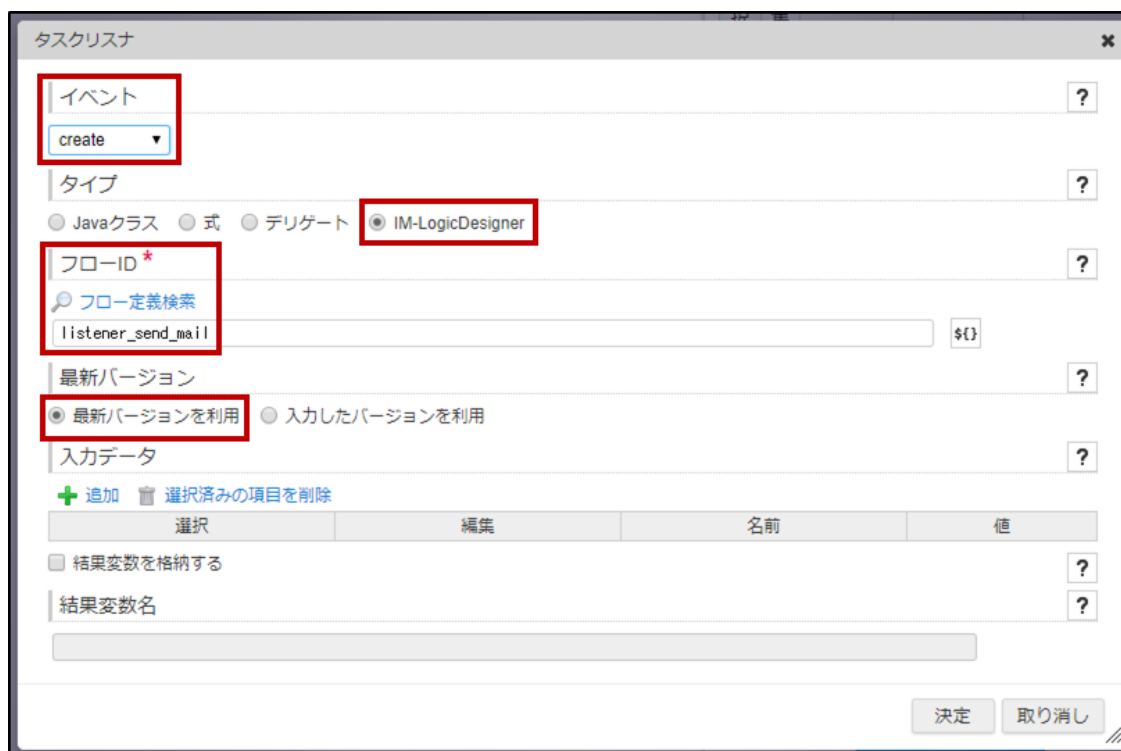
5. 「リスナ」タブから、「タスクリスナ」の「追加」リンクをクリックします。



図：「ユーザタスク」 - 「プロパティ」 - 「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」

i 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。

プロセス開始	プロセス図	プロセス定義名	カテゴリ	説明	ドキュメント
		タスクの処理依頼メールを送信するプロセス	http://www.intra-mart.jp/im_bpm		

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。

「タスク処理依頼メール」を送信するユーザを指定する

「タスク処理依頼メール」を送信したいユーザコードを入力し、下の「送信」ボタンをクリックしてください。

依頼メールのあて先は
「サイトマップ」 - 「共通マスタ」 - 「ユーザ」から「ueda」を検索し、
「プロフィール」タブの「メールアドレス1」に設定してください。

ユーザコード 

図：Formaアプリケーション「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。

ユーザー検索

検索基準日: 2018/06/06 ロケール: 日本語

キーワード

検索キーワードを入力してください。

☒ 名前 ☒ コード ☒ フリガナ

☒ 前方一致 ☐ 完全一致 ☐ 部分一致

検索

あ行	あ	い	う	え	お
か行	か	き	く	け	こ
さ行	さ	し	す	せ	そ
た行	た	ち	つ	て	と
な行	な	に	ぬ	ね	の
は行	は	ひ	ふ	へ	ほ
ま行	ま	み	む	め	も
や行	や		ゆ		よ
ら行	ら	り	る	れ	ろ
わ行	わ				

上田辰男

決定

図：「ユーザー検索」

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Formaアプリケーション「conduct_button」が開きます。

タスク処理依頼 2018年05月30日 午後 5:47

差出人: requestConduct@imart.jp

宛先: ueda@imart.jp

下記のURLにアクセスし、処理を完了させてください。

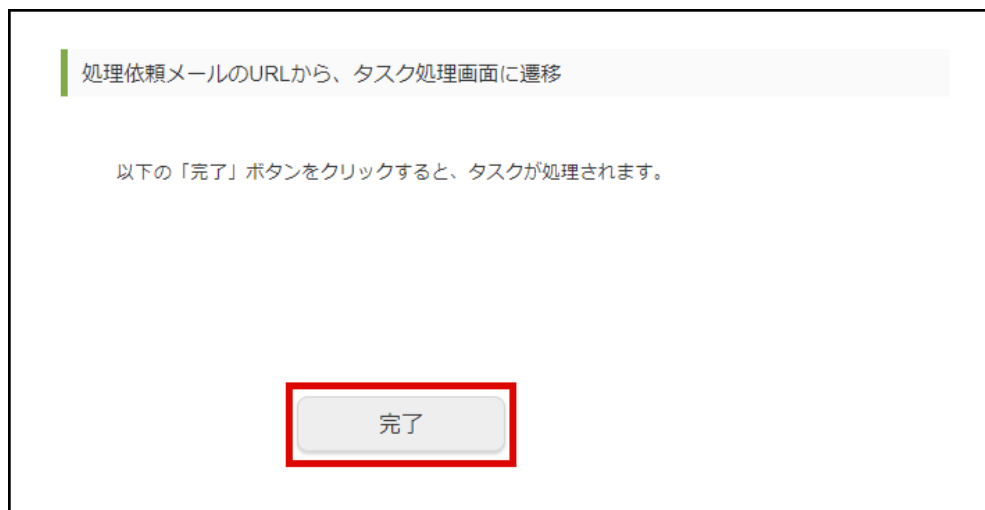
http://localhost:8080/imart/forma/normal/view/regist_application_view/conduct_button?imfr_callback_path=bpm%2ftask%2flist&imfr_not_save_item_data=true&app_taskId=8etcccc3&pfra

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：Formaアプリケーション「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 プロセスデザイナー 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 プロセスデザイナー 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](#)」

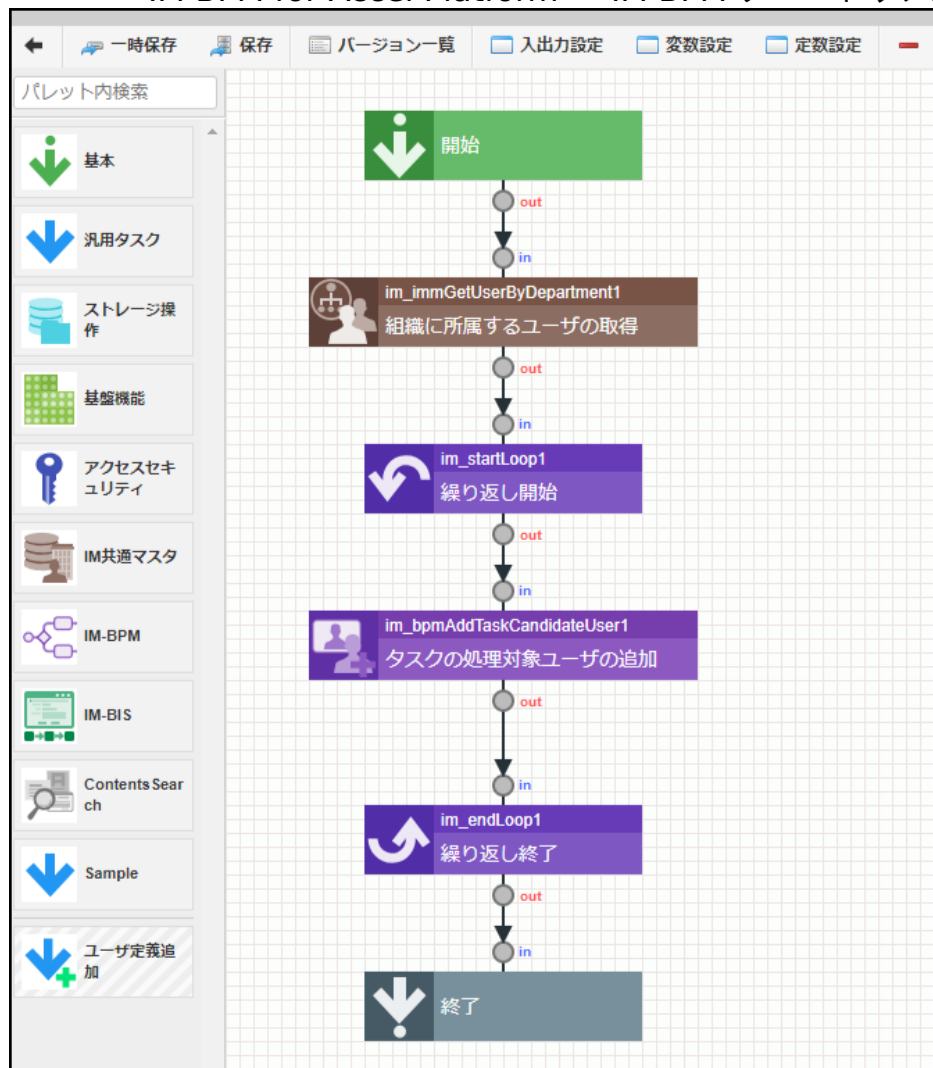
- [ロジックフローを作成する](#)
- [プロセス定義を作成する](#)
- [実行結果を確認する](#)

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

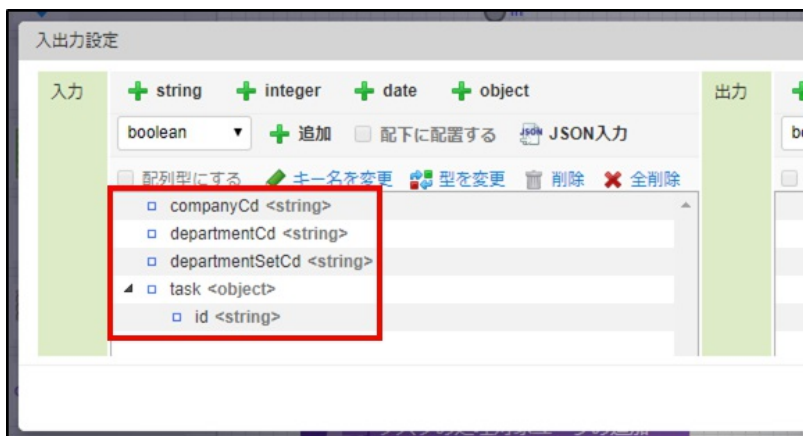
1. ツールバーの「入出力設定」をクリックします。



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。

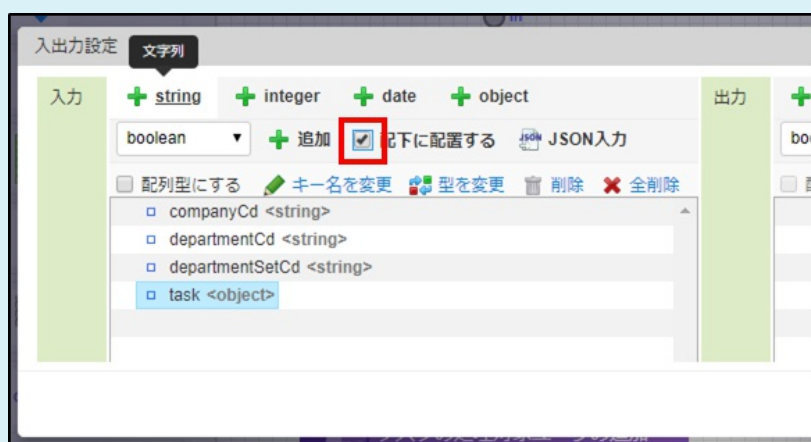
キー名	型
companyCd	<string>
departmentSetCd	<string>
departmentCd	<string>
task	<object>
task - id	<string>



図：「入出力設定」 - 「入力」

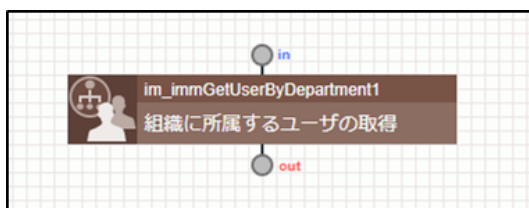
コラム

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」

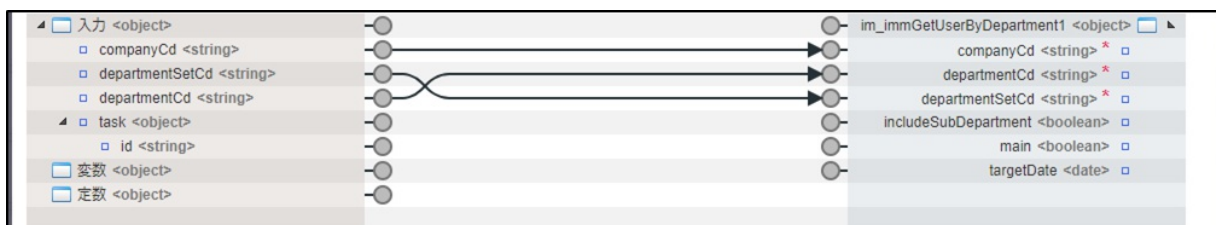
3. パレットの種別「IM共通マスタ」から、「組織に所属するユーザの取得」を設置します。



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>



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。

- 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」

- 「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。

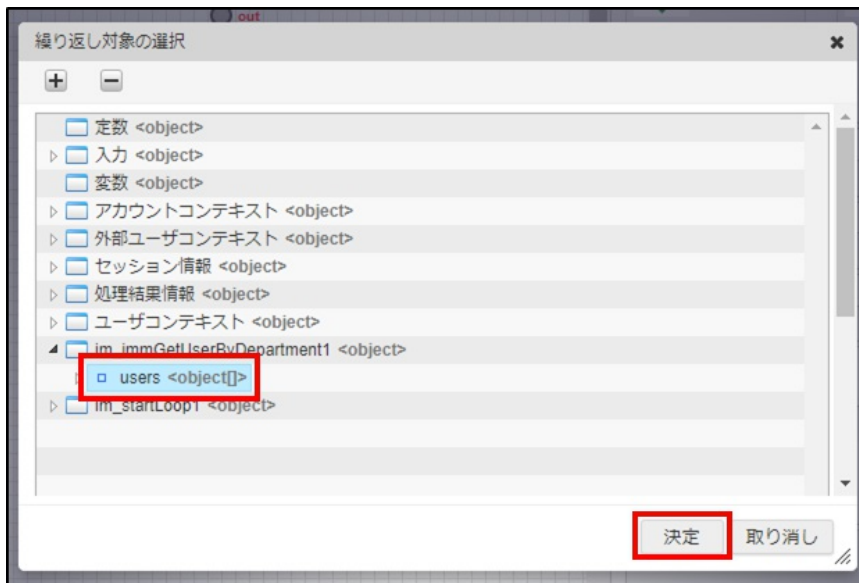
図：「プロパティ」 - 「タスク固有設定」 - 「繰り返し対象」



コラム

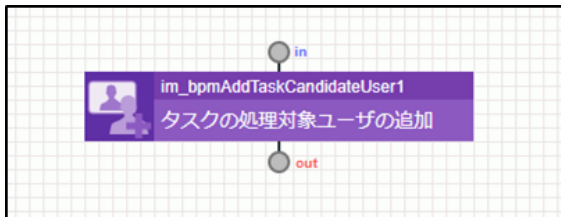
「繰り返し」に関する制御要素の詳細は「IM-LogicDesigner チュートリアルガイド」-「繰り返し処理を利用したフロー」を参照してください。

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」

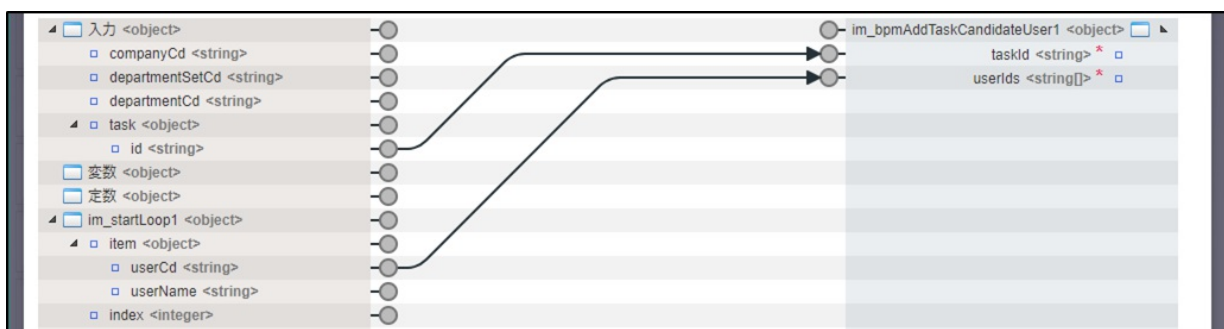
8. パレットの種別「IM-BPM」から「タスク処理対象のユーザの追加」を選択し、上記で配置した「繰り返し開始」と「繰り返し終了」の間に配置します。



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[]>



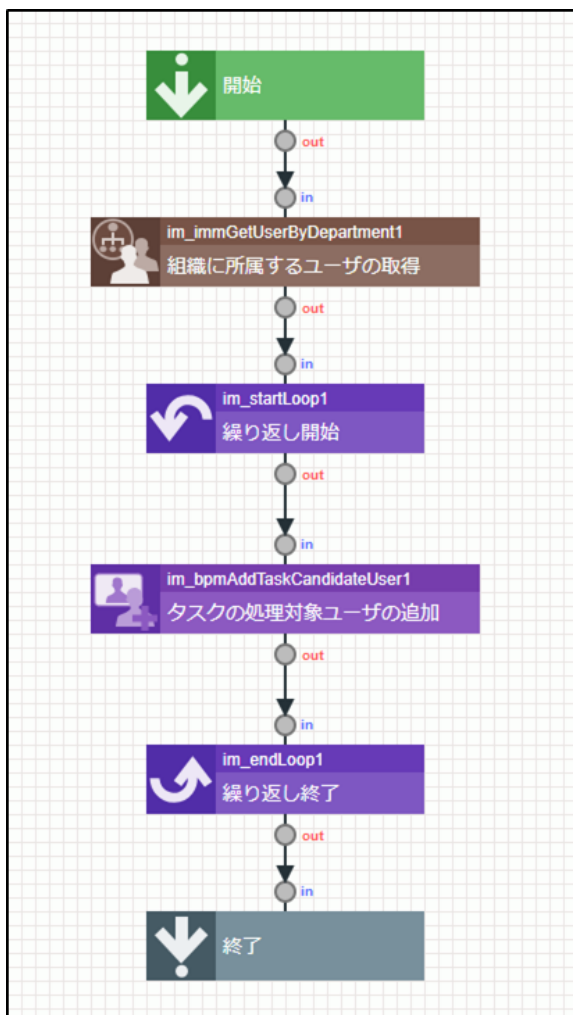
図：「タスク処理対象のユーザの追加」 - 「マッピング設定」



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」

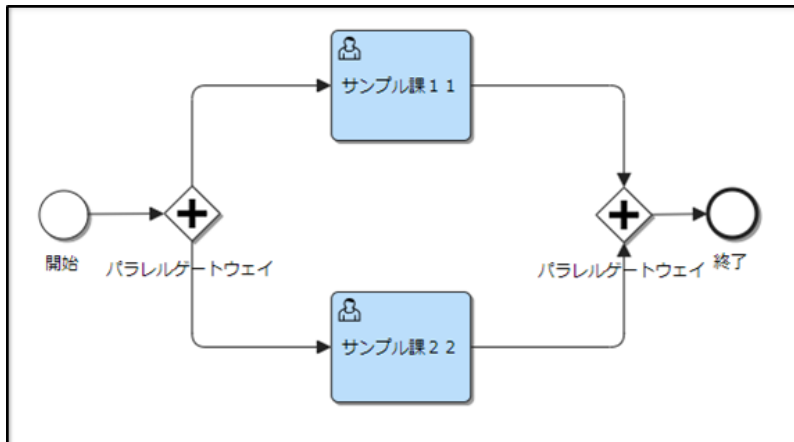
11. 「ロジックフロー定義編集」画面上部、ヘッダ内の「新規保存」をクリックします。
以下のように設定し、ロジックフロー定義を新規保存します。

- 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」

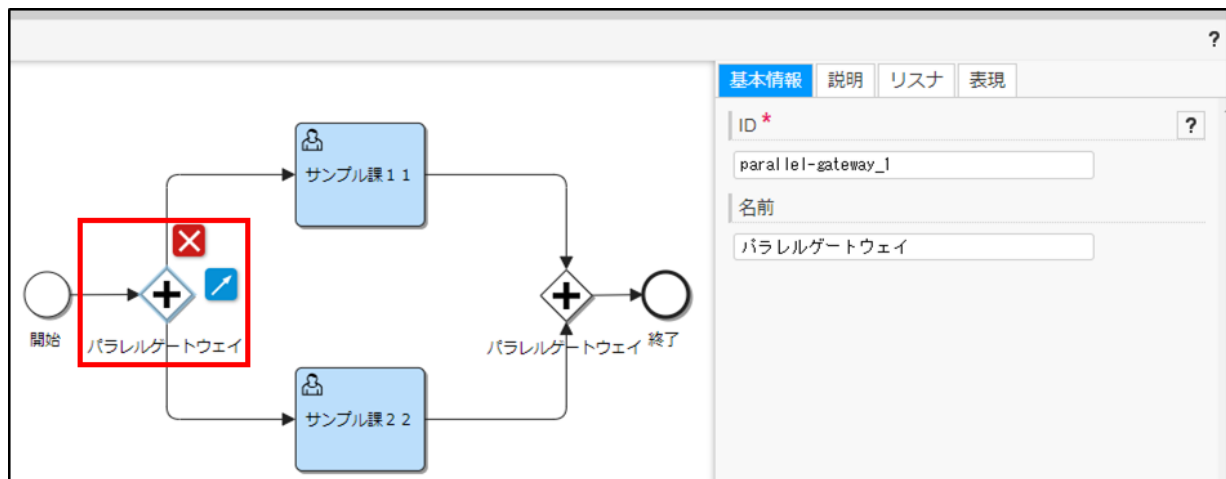
プロセス定義を作成する

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 1 1」と「サンプル課 2 2」の2つを配置します。
 「サンプル課 1 1」のユーザタスクには「サンプル課 1 1」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 2 2」のユーザタスクには「サンプル課 2 2」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。



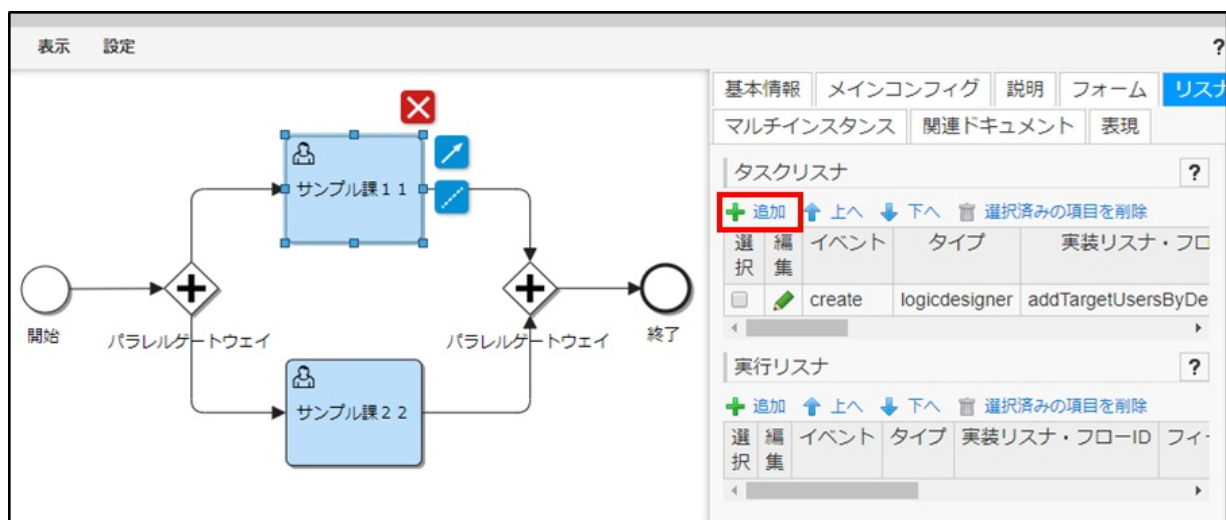
図：完成イメージ

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」タブから、「タスクリスナ」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」 - 「リスナ」

4. 「サンプル課 1 1」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

タスクリスナ

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

選択	編集	名前	値
☐		companyCd	comp_sample_01
☐		departmentSetCd	comp_sample_01
☐		departmentCd	dept_sample_11

☐ 結果変数を格納する

結果変数名

決定 取り消し

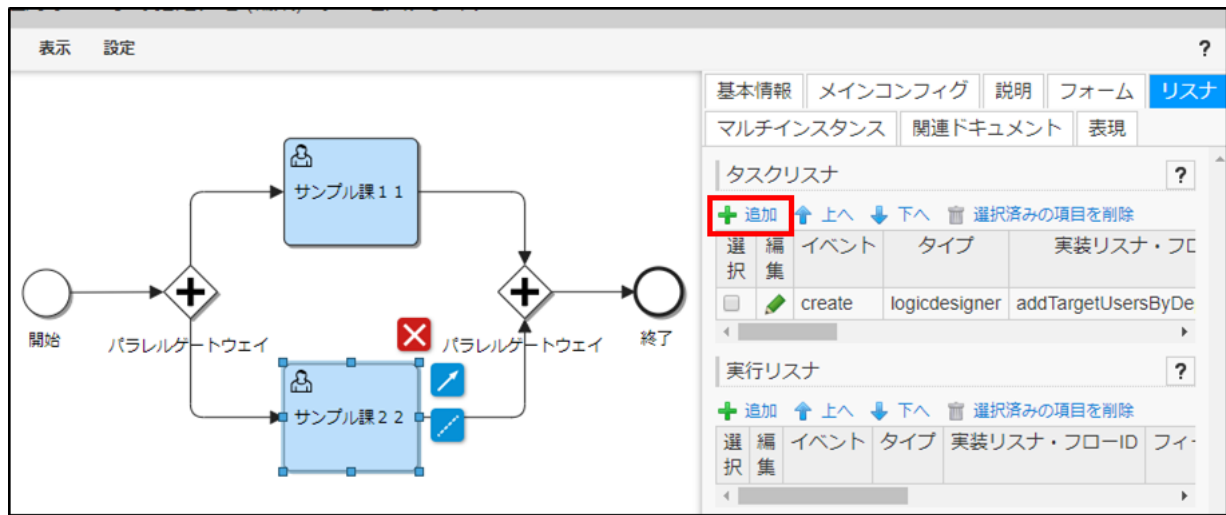
図: 「ユーザタスク」 - 「プロパティ」 - 「リスナ」



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」タブから、「タスクリスナ」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」 - 「リスナ」

6. 「サンプル課 2 2」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_22

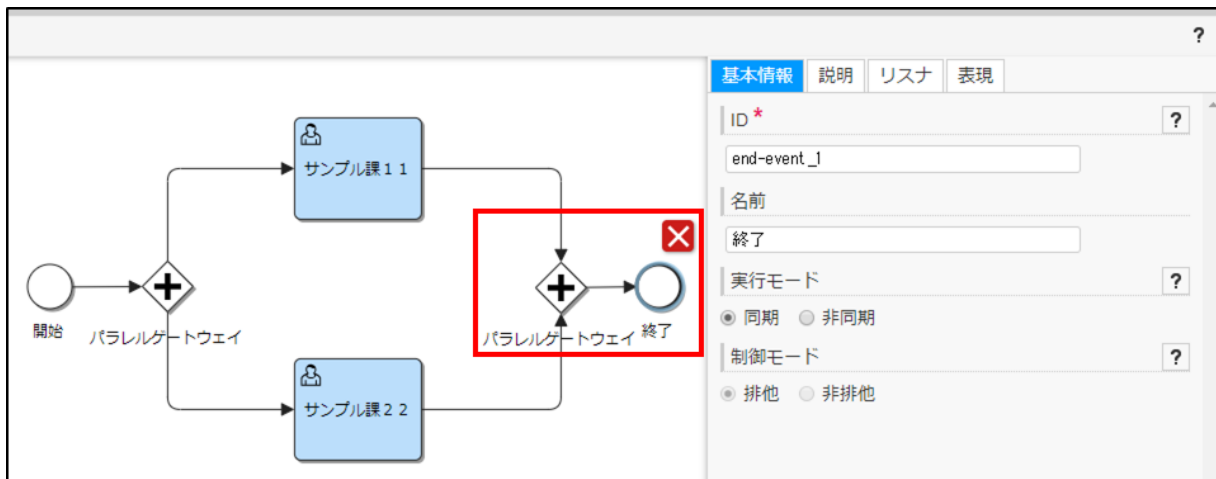
The screenshot shows the 'タスクリスナ' (Task Listener) configuration dialog box. The 'イベント' (Event) dropdown is set to 'create'. The 'タイプ' (Type) is set to 'IM-LogicDesigner'. The 'フローID' (Flow ID) is 'addTargetUsersByDepartment'. The '最新バージョン' (Latest Version) section has '最新バージョンを利用' (Use latest version) selected. The '入力データ' (Input Data) table is as follows:

選択	編集	名前	値
☐		companyCd	comp_sample_01
☐		departmentSetCd	comp_sample_01
☐		departmentCd	dept_sample_22

The '決定' (OK) and '取り消し' (Cancel) buttons are at the bottom right.

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。



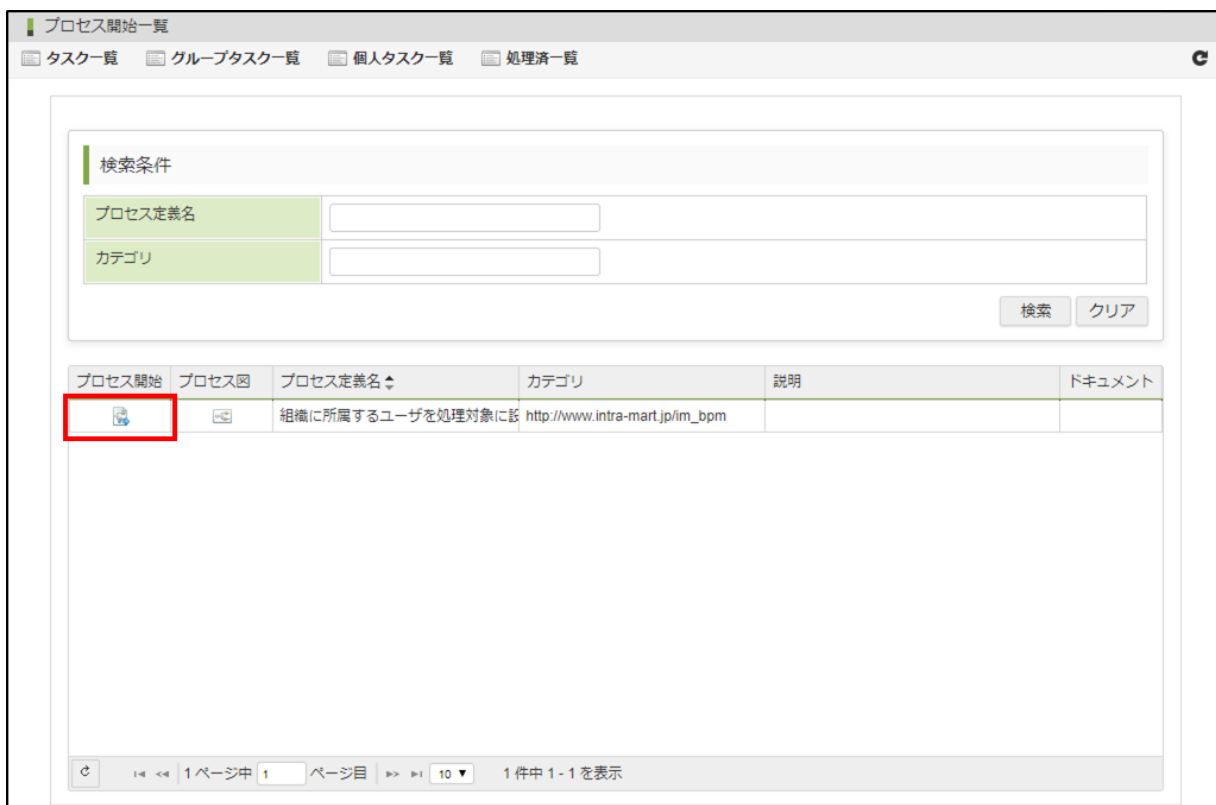
図：「パラレルゲートウェイ」と「終了イベント」

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。

プロセス一覧

検索条件

業務キー

プロセス定義ID

プロセス定義キー

プロセス定義バージョン

プロセス定義名

カテゴリ

開始日

ステータス

変数検索

検索

クリア

詳細	業務キー	プロセス定義名	プロセス定義ID	プロセス定義キー	開始日時	完了日時	ステータス
		組織に所属するユーザ	addTargetUsersByDep	addTargetUsersByDep	2018/07/23 15:53:22		実行中

1件中 1 - 1 を表示

図：「プロセス一覧」

- 「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。

プロセス詳細

プロセス一覧 変数一覧 プロセス履歴 プロセス移行 プロセス一括移行 プロセス削除 ドキュメント

プロセス定義ID	addTargetUsersByDepartment:1:8ev7cqbpucbprq	プロセス定義名	組織に所属するユーザを処理対象に設定するプロセス
プロセス定義キー	addTargetUsersByDepartment	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8ev7cqwhgucirq	開始ユーザ	青柳辰巳
開始日時～完了日時	2018/07/23 15:53:22 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

表示倍率: 100 %

タイムライン

2018/07/23 15:53

サンプル課 1 1

タスクの処理を待っています。

処理対象ユーザ: 大碓博文, 青柳辰巳, 片山聡, 林政義, 吉川一哉

開始 2018/07/23 15:53:22
経過 2分

2018/07/23 15:53

サンプル課 2 2

タスクの処理を待っています。

処理対象ユーザ: 関根千香, 生田一哉, 上田辰男, 萩本順子, 円山益男

開始 2018/07/23 15:53:22
経過 2分

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 1 1」の処理対象ユーザであることを確認します。

組織

組織セットの新規登録 検索 役職表示 会社ソート順変更 最新情報

基準日 2018/07/17 ロケール 日本語 表示 会社名 サンプル会社 会社選択 有効なもののみ表示

1999/12/13 1年間表示 2000/12/12

2000/01/01 - 2999/12/31

サンプル会社

サンプル部門 0 1

サンプル課 1 1

サンプル課 1 2

サンプル部門 0 2

サンプル課 2 1

サンプル課 2 2

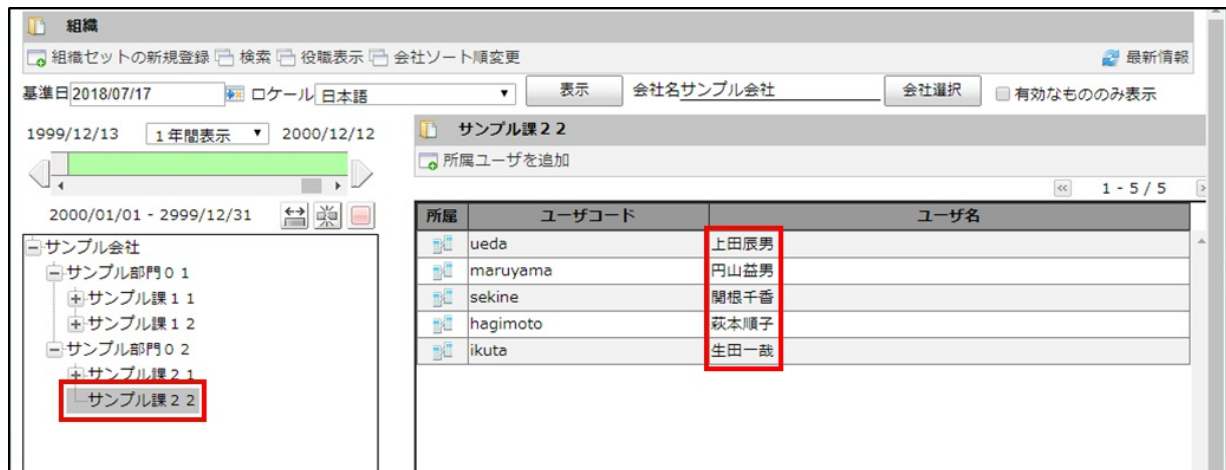
サンプル課 1 1

所属ユーザを追加

所属	ユーザコード	ユーザ名
	aoiyagi	青柳辰巳
	hayashi	林政義
	yoshikawa	吉川一哉
	ohiso	大碓博文
	katayama	片山聡

図: 「共通マスタ」 - 「組織」

5. さらに、「サンプル課 2 2」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 2 2」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

i コラム

このチュートリアルで作成する資料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 プロセスデザイナー 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」

i 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 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。

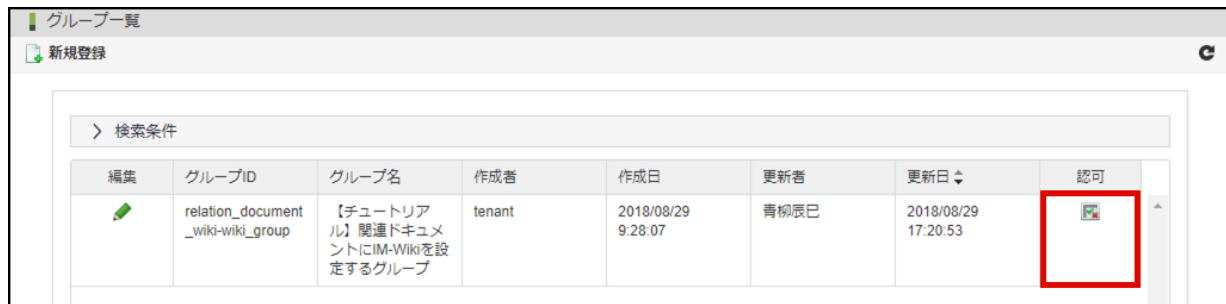


図：「グループ一覧」

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

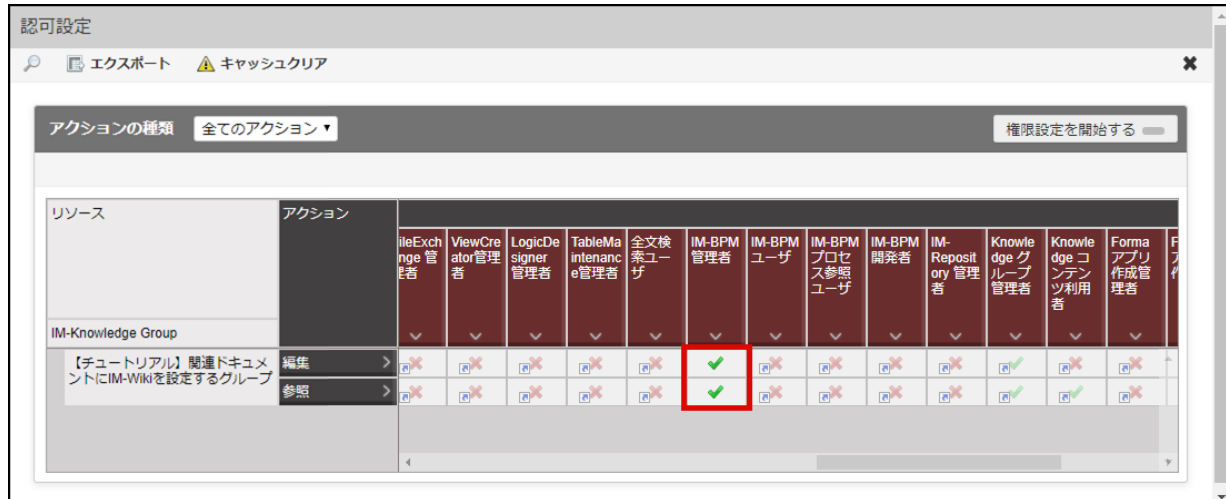
図：「グループ登録」

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」

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。

Wiki - 新規作成

コンテンツ一覧 一時保存 登録

グループ * [チュートリアル] 関連ドキュメントにIM-Wiki ▼

Wiki名 * [チュートリアル] 関連ドキュメントとして設定するIM-Wiki

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 * Markdown ▼

本文 ?

1 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できます。

2

3 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。

4 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。

プレビュー

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できます。

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。

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

コンテンツ一覧 新規ページ作成

プロセス開始時に参照するIM-Wiki

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できます。

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。

ページ一覧 目次

■ プロセス開始時に参照するIM-Wiki

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。

Wiki - 新規ページ作成

← 一時保存 登録

親ページ [プロセス開始時に参照するIM-Wiki ▼]

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 * Markdown ▼

本文 ?

1 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できます。

2

3 「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。

プレビュー

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できます。

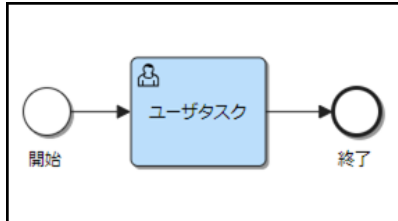
「IM-FormaDesigner」の画面で aoyagi を指定したため、このタスクが「タスク一覧」に表示されました。

図：「IM-Wiki」 - 「新規ページ作成」

15. 入力後、「登録」をクリックします。

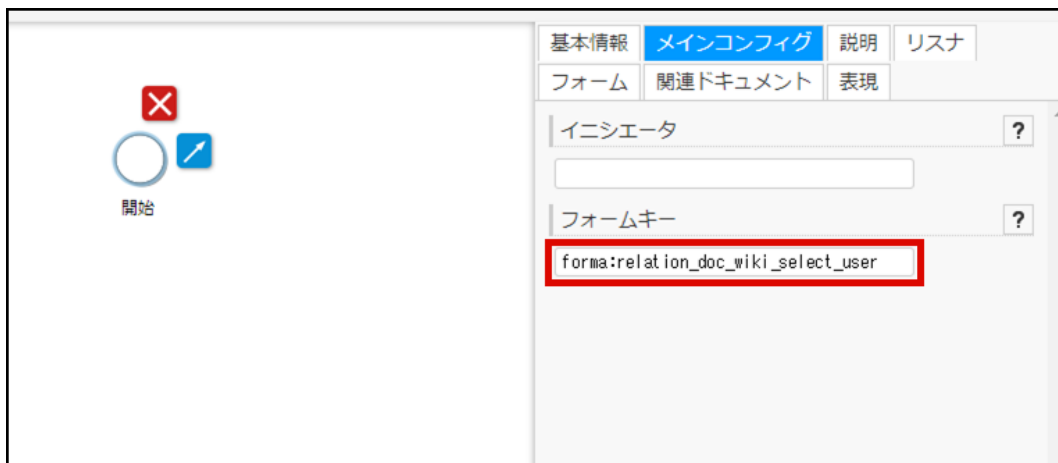
プロセス定義を作成する

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。



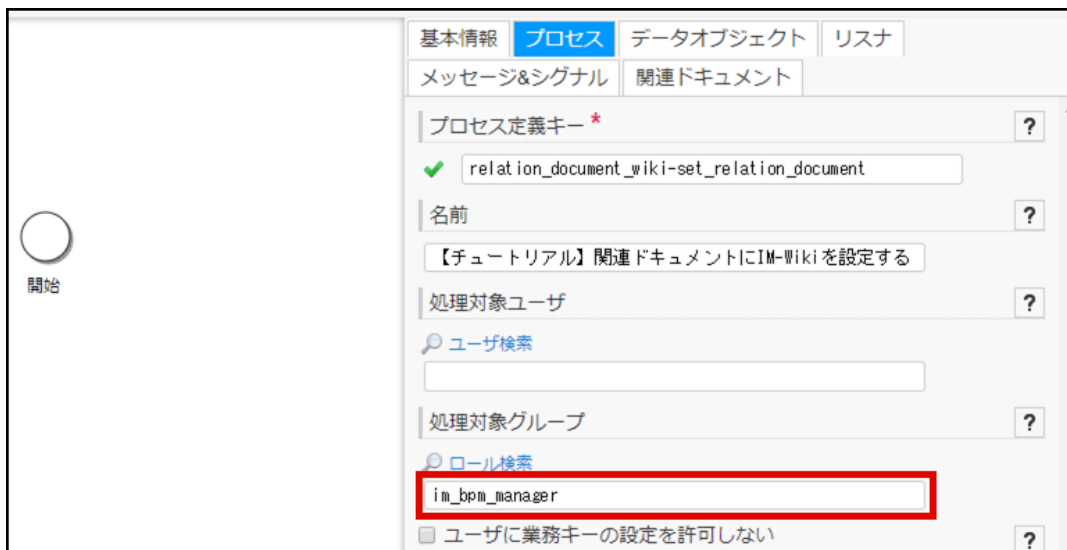
図：完成イメージ

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」



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/`

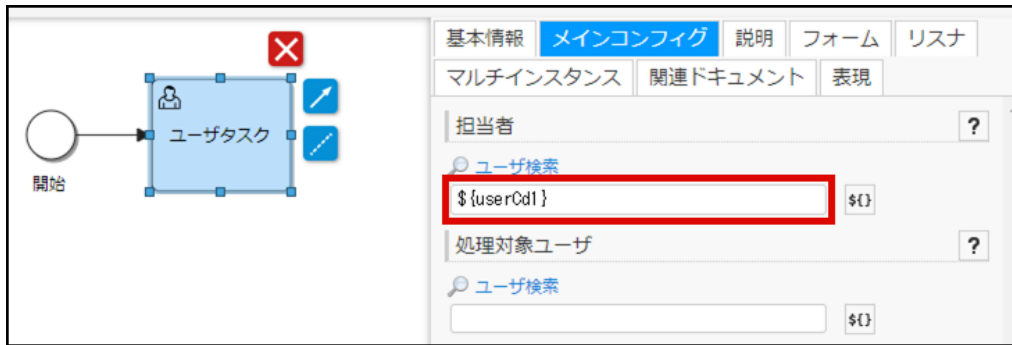
URLは、絶対パス、または、相対パスで指定してください。
相対パスの指定は、IM-WikiとIM-Wikiを開こうとしている環境のベースURLが同一の場合のみ利用可能です。



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`

- 「決定」をクリックします。
- 「ユーザタスク」を配置します。
- 「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

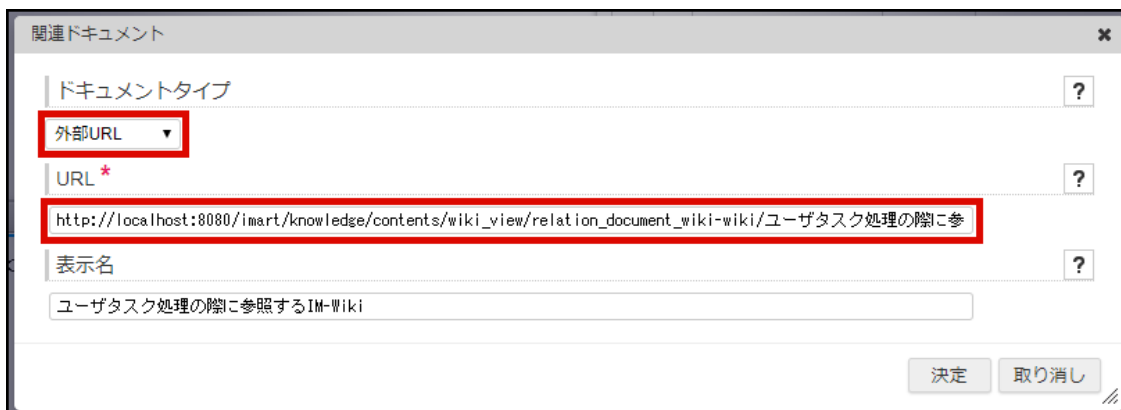
`${userCd1}` は、「IM-FormaDesigner」で作成した入力フォームのフィールド識別IDです。
「担当者」にEL式を入力することにより、フォームで指定したユーザをタスクの担当者に設定できます。

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」

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

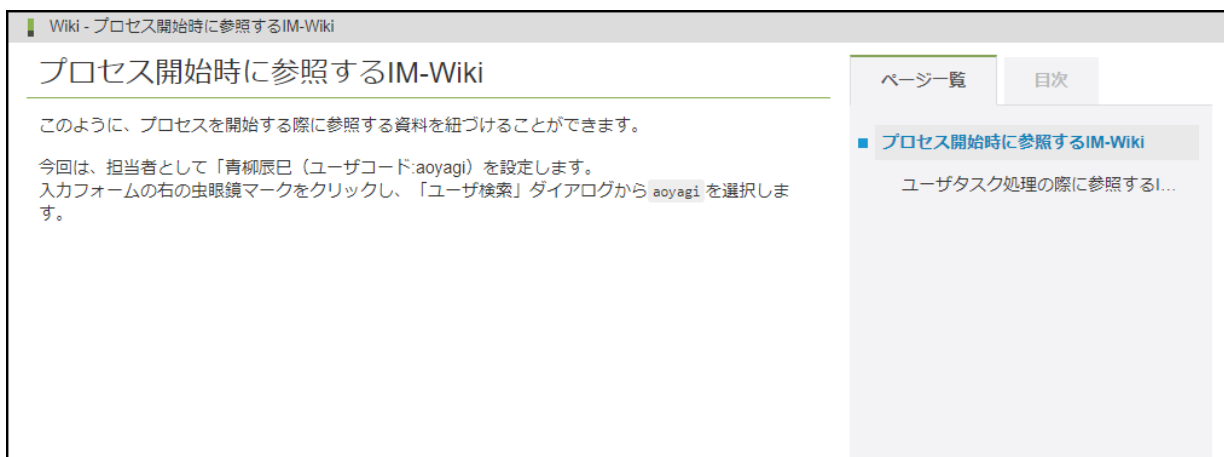
このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行結果の確認を行います。
「Knowledge コンテンツ利用者」のロールを付与した **aoyagi** でログインしてください。

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：プロセス開始時に参照するIM-Wiki（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。



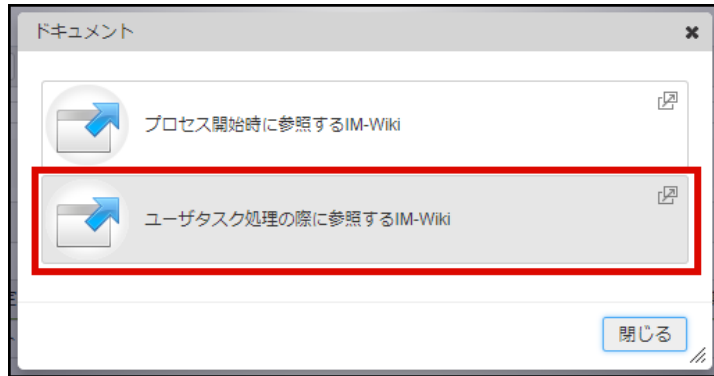
図：「プロセス開始一覧」

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」

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：ユーザタスク処理の際に参照するIM-Wiki (IM-Wiki)

オプションタスク

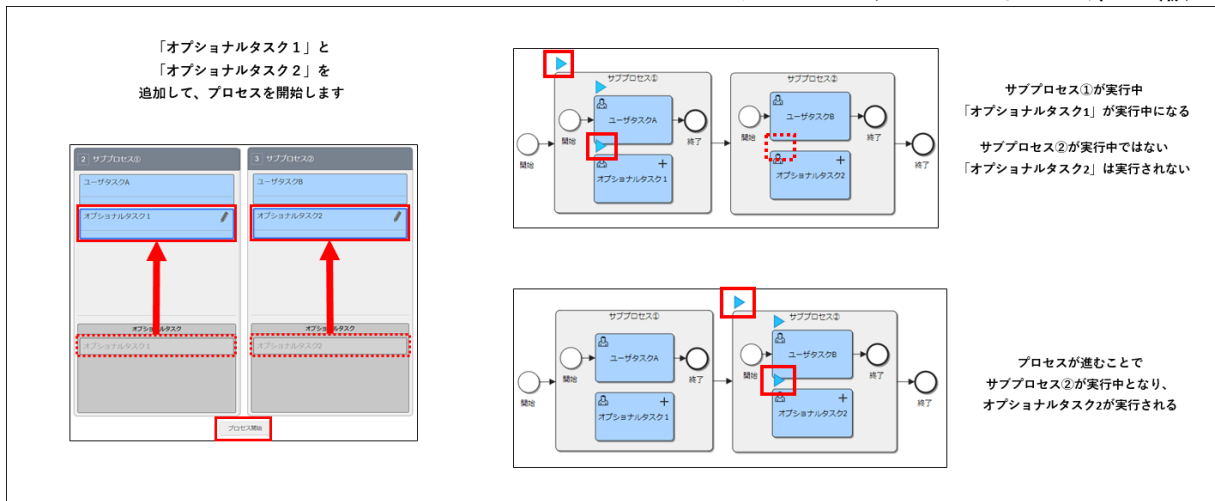
オプションタスクを追加する

このチュートリアルでは、オプションタスクを使用して実行中のプロセスを変更する方法を解説します。
オプションタスクは、プロセスの開始時、または、実行中に追加できるタスクです。

下記のフローエレメントをオプションタスクにできます。

- タスク
- ユーザタスク
- スクリプトタスク
- サービスタスク
- メールタスク
- マニュアルタスク
- 受信タスク
- コールアクティビティ
- IM-LogicDesignerタスク
- 申請タスク
- 起票タスク

オプションタスクの詳細については、「IM-BPM 仕様書」 - 「オプションタスク」 もあわせて参照してください。



図：概要図

コラム

このチュートリアルで作成するプロセス定義は、以下のリンクからダウンロードできます。

[optional_task_usage.bpmn](#)

プロセス定義のアップロード方法については、以下のリンクを参照してください。

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

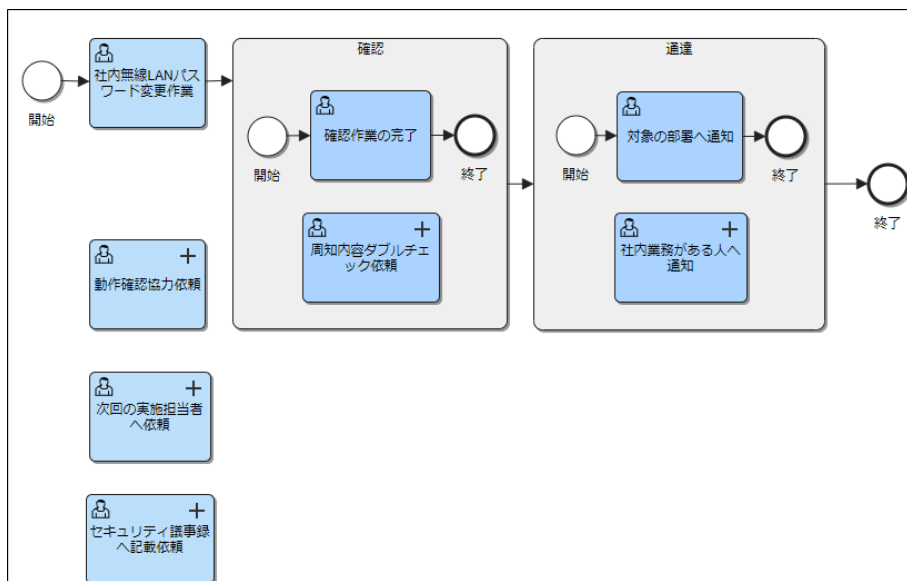
プロセス定義を作成する

下記の図は、プロセス・サブプロセスにオプションタスクを用意したプロセス定義です。

オプションタスクは、タスクの右上に「+」のマークが出ます。

オプションタスクはプロセスのどの段階でも追加・編集できますが、実行中になると編集できません。

- プロセス上のオプションタスク
「動作確認協力依頼」タスク、「次回の実施担当者へ依頼」タスク、「セキュリティ議事録へ記載依頼」タスクがオプションタスクです。
- 「確認」サブプロセス内のオプションタスク
「周知内容ダブルチェック依頼」タスクがオプションタスクです。
「確認」サブプロセスが実行中の場合、追加はできますが編集はできません。
- 「通達」サブプロセス内のオプションタスク
「社内業務がある人へ通知」タスクがオプションタスクです。
「通達」サブプロセスが実行中の場合、追加はできますが編集はできません。



図：プロセス定義図

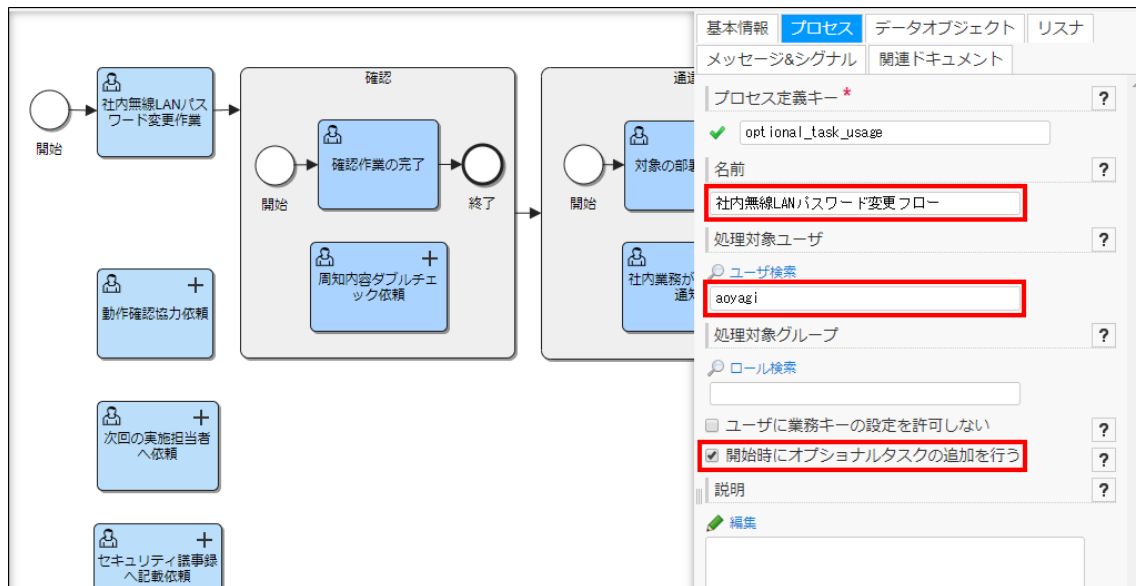
1. 「開始イベント」を設置します。

2. プロセスの処理対象ユーザと、オプションタスクを追加する画面を表示するよう設定をします。

キャンパスの空白部分をクリックし、「プロパティ」をプロセス全体に切り替えます。

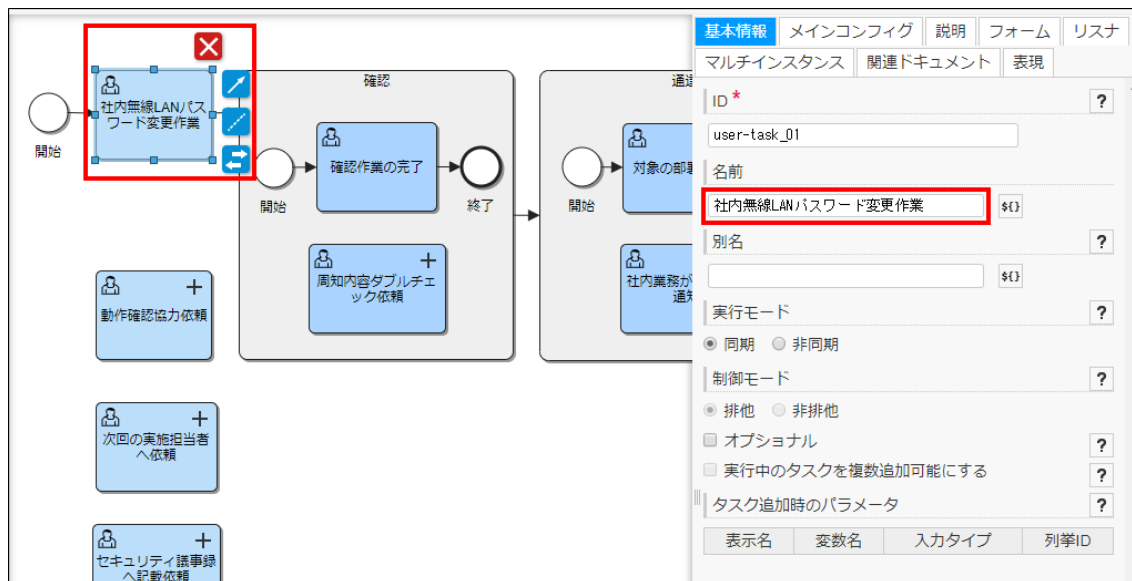
プロセスタブから、以下のように項目を設定してください。

- 名前：社内無線LANパスワード変更フロー
- 処理対象ユーザ：aoyagi
- 開始時にオプションタスクの追加を行う：有効



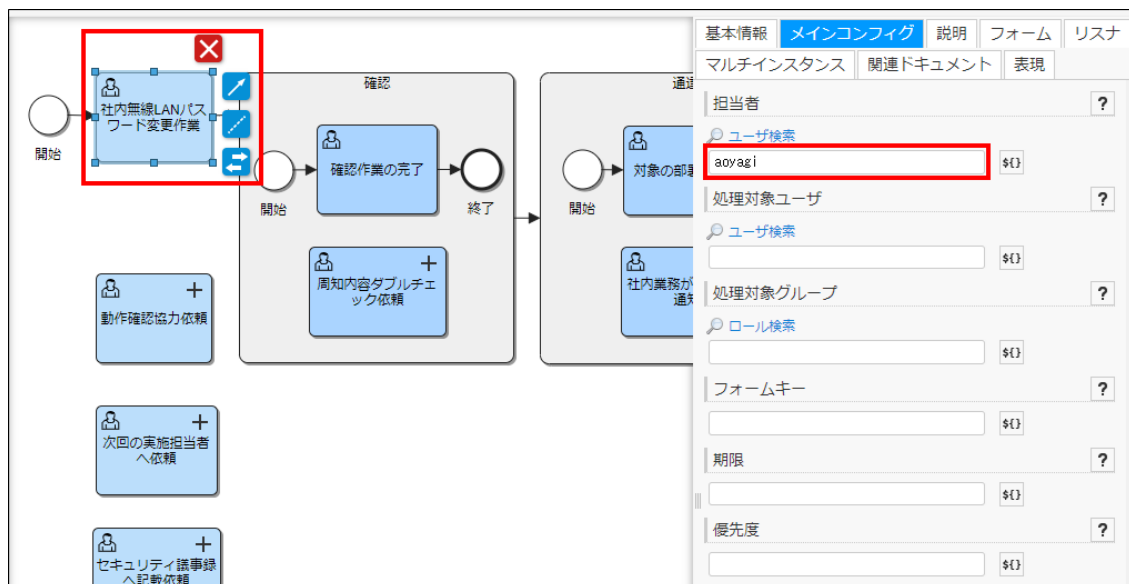
図：プロセス全体 - 「プロパティ」 - 「プロセス」

3. サブプロセス外からオプションタスクを追加できることを確認するため、ユーザタスクを配置します。
「ユーザタスク」を設置します。
4. 動作確認の際に「タスク一覧」画面でわかりやすくするため、ユーザタスクに名前を付けます。
「ユーザタスク」を選択し、「基本情報」タブから「名前」に 社内無線LANパスワード変更作業 と設定します。



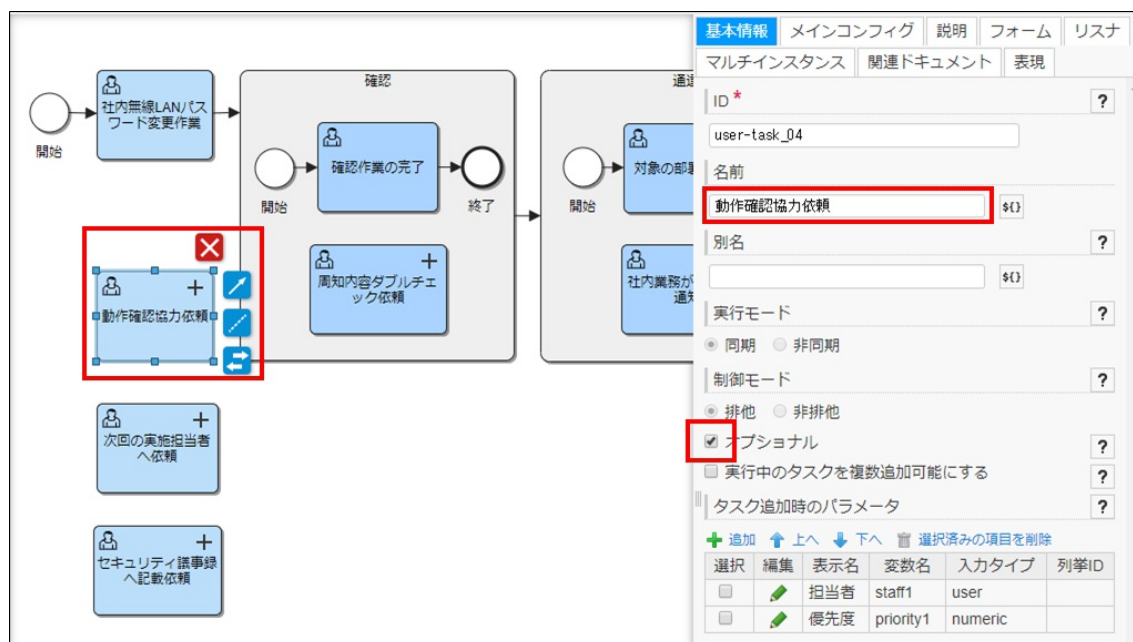
図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

5. 「社内無線LANパスワード変更作業」タスクに対して、担当者を設定をします。
「メインコンフィグ」タブから、「担当者」に aoyagi と設定してください。



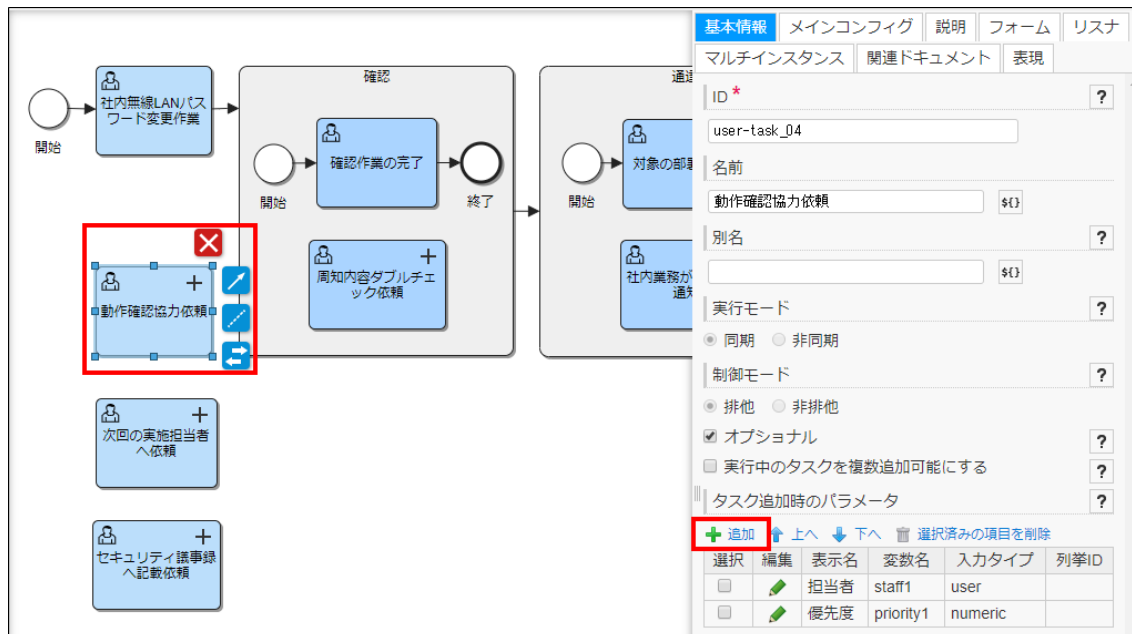
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

6. プロセス直下に、1つ目のオプションタスクを追加します。
「ユーザタスク」を設置します。
7. 設置したユーザタスクをオプションタスクにします。
「ユーザタスク」をクリックし、「基本情報」タブから以下のように項目を設定してください。
 - 名前：動作確認協力依頼
 - オプション：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

8. 「動作確認協力依頼」タスクを追加する際に、指定できるパラメータの設定をします。
「基本情報」タブから「タスク追加時のパラメータ」の「追加」リンクをクリックしてください。

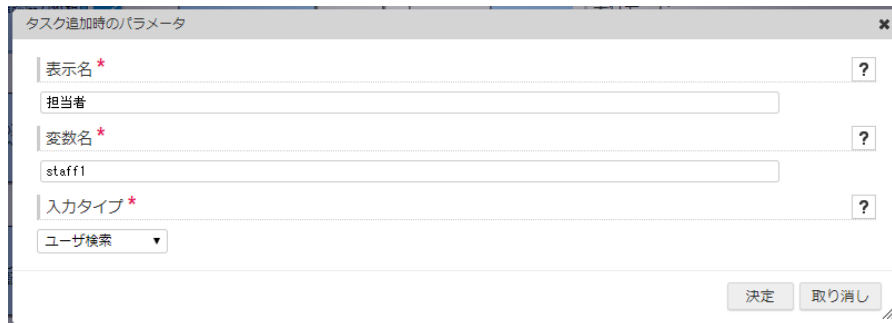


図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

9. 「タスク追加時のパラメータ」ダイアログが表示されます。

以下のように、2つのパラメータを設定してください。

- 担当者
表示名：担当者
変数名：staff1
入力タイプ：ユーザ検索
- 優先度
表示名：優先度
変数名：priority1
入力タイプ：数字



図：「タスク追加時のパラメータ」

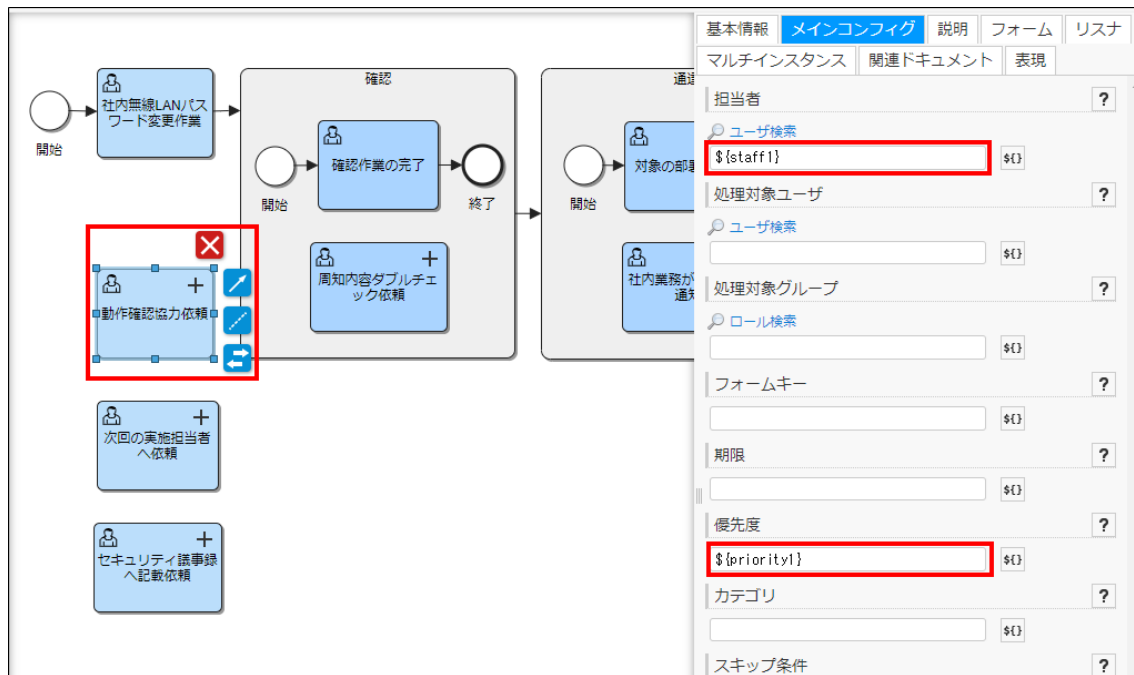
コラム

変数名はオプションタスクごとに設定します。
異なるオプションタスクであれば、同一の変数名を使用できます。
変数を参照できる範囲は、設定したオプションタスク内のみです。

10. EL式を使用し、指定されたパラメータが各項目に設定されるようにします。

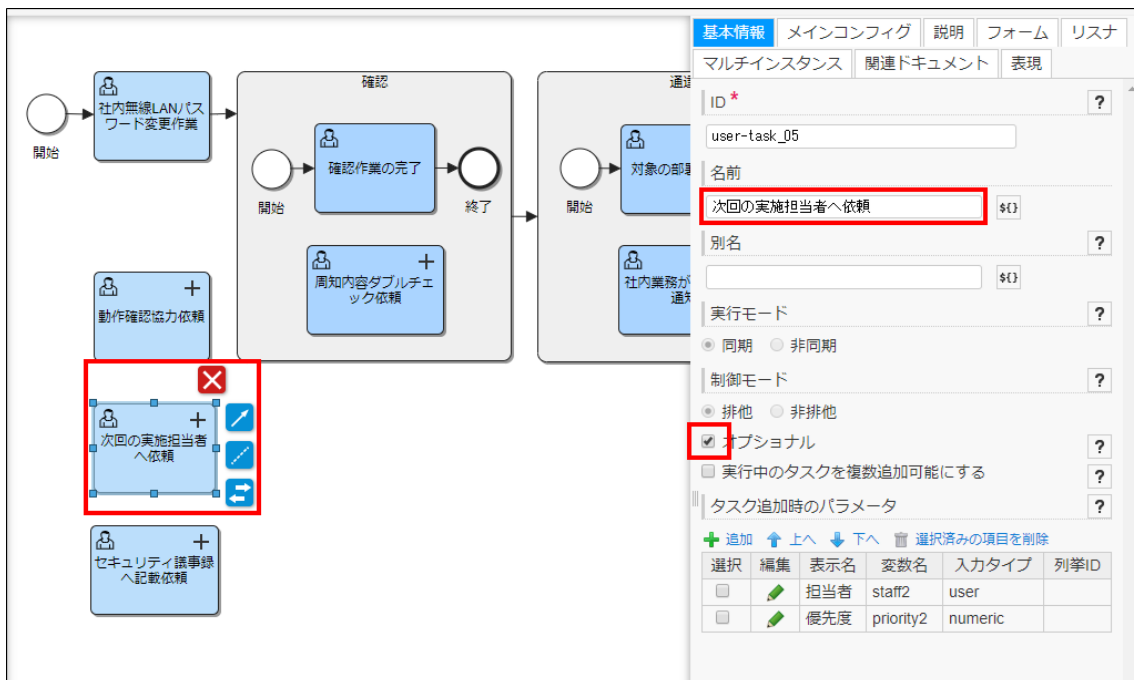
「ユーザタスク」をクリックし、「メインコンフィグ」タブから以下の項目を設定します。

- 担当者：\${staff1}
- 優先度：\${priority1}



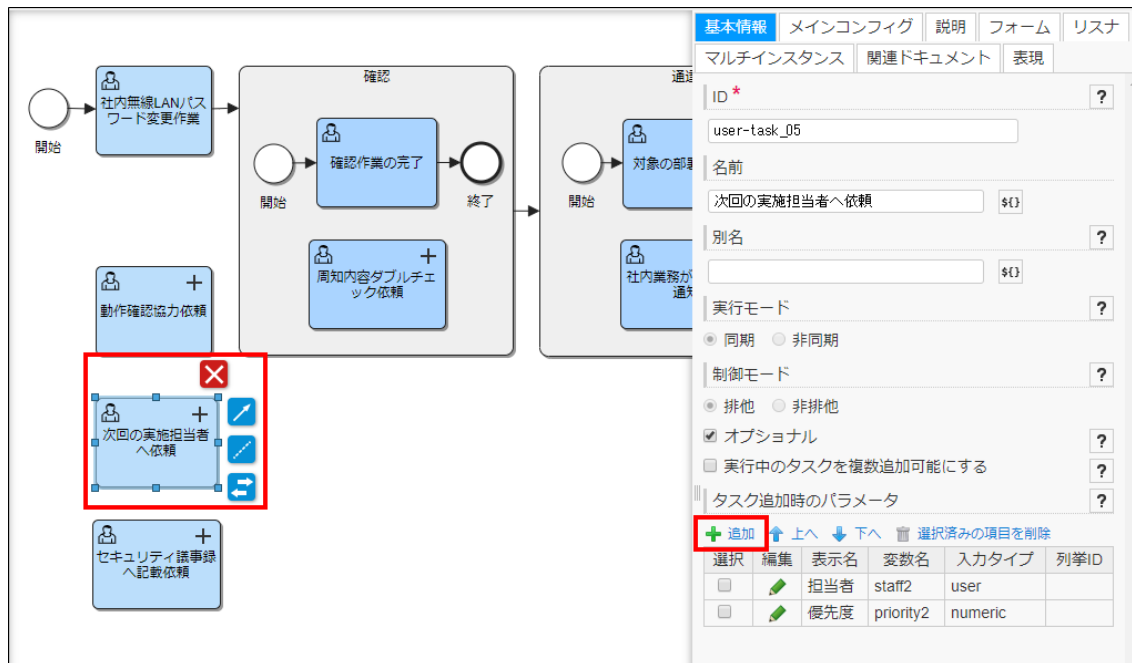
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

11. 上記と同じ手順で、2つ目のユーザタスクを追加します。
「ユーザタスク」を設置します。
12. 設置したユーザタスクをオプションタスクにします。
「ユーザタスク」をクリックし、「基本情報」タブから以下のように項目を設定してください。
 - 名前：次回の実施担当者へ依頼
 - オptional：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

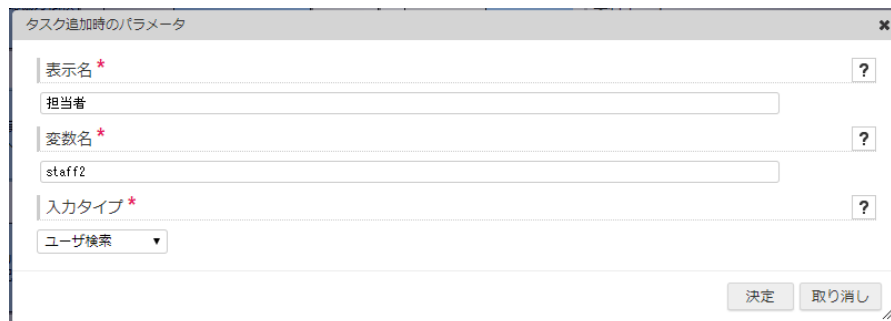
13. オプションタスクを追加する際に、指定できるパラメータの設定をします。
「基本情報」タブから「タスク追加時のパラメータ」の「追加」リンクをクリックしてください。



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

14. 「タスク追加時のパラメータ」ダイアログが表示されます。
以下のように、2つのパラメータを設定してください。

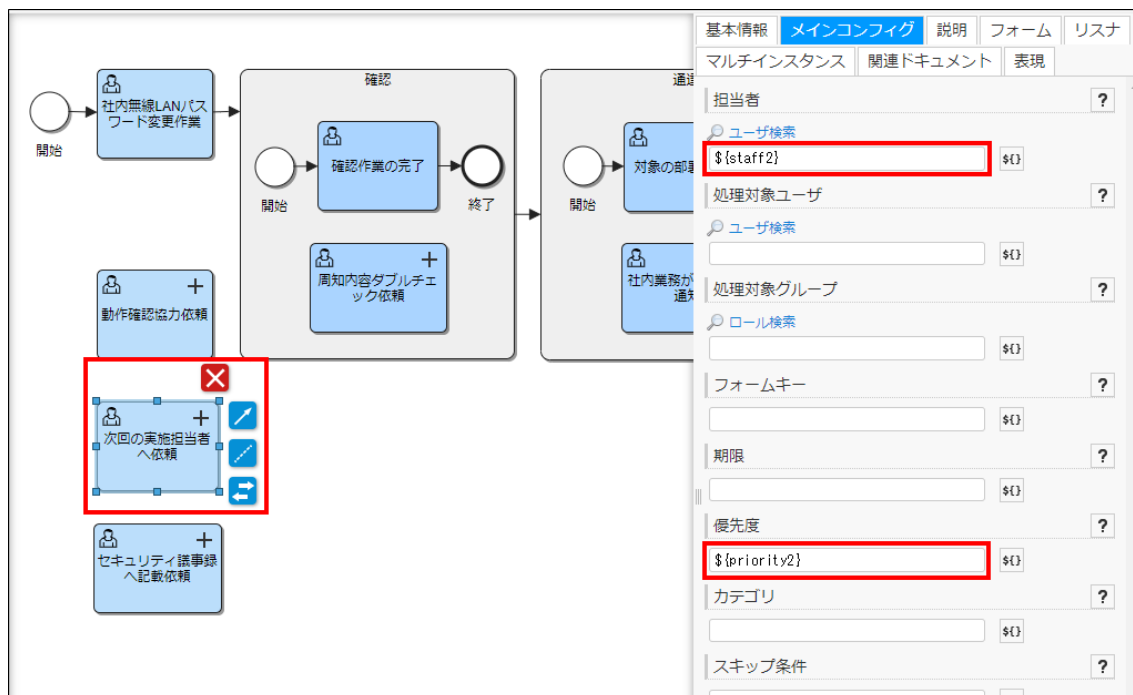
- 担当者
表示名：担当者
変数名：staff2
入力タイプ：ユーザ検索
- 優先度
表示名：優先度
変数名：priority2
入力タイプ：数字



図：「タスク追加時のパラメータ」

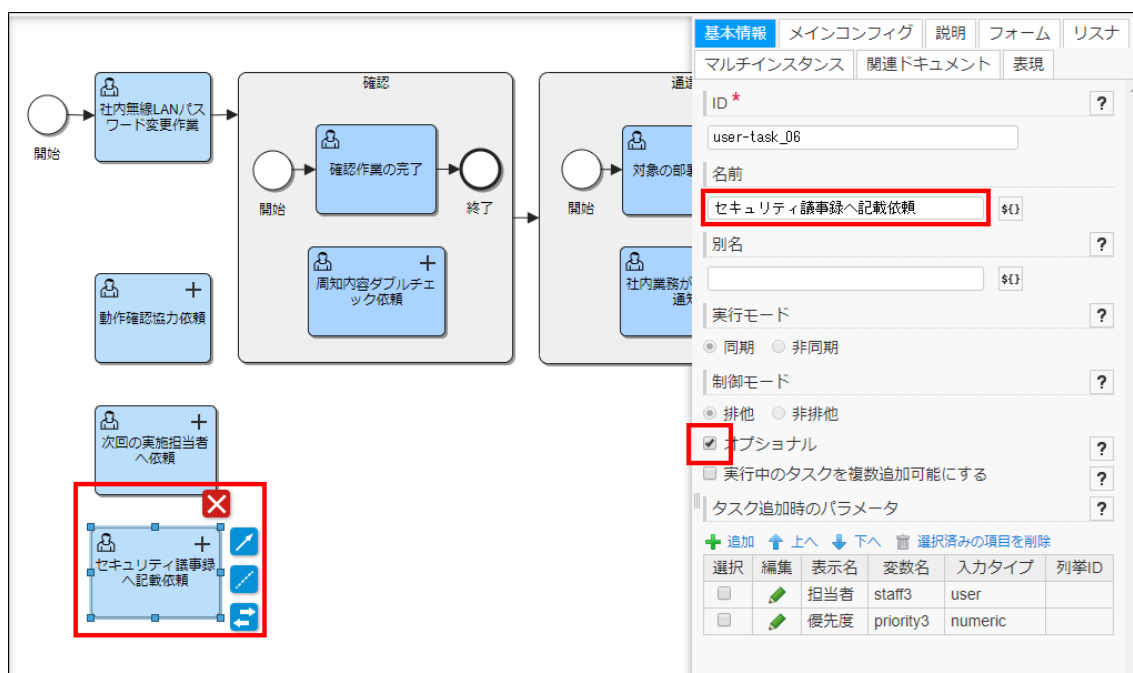
15. EL式を使用し、指定されたパラメータが各項目に設定されるようにします。
「ユーザタスク」をクリックし、「メインコンフィグ」タブから以下の項目を設定します。

- 担当者：\${staff2}
- 優先度：\${priority2}



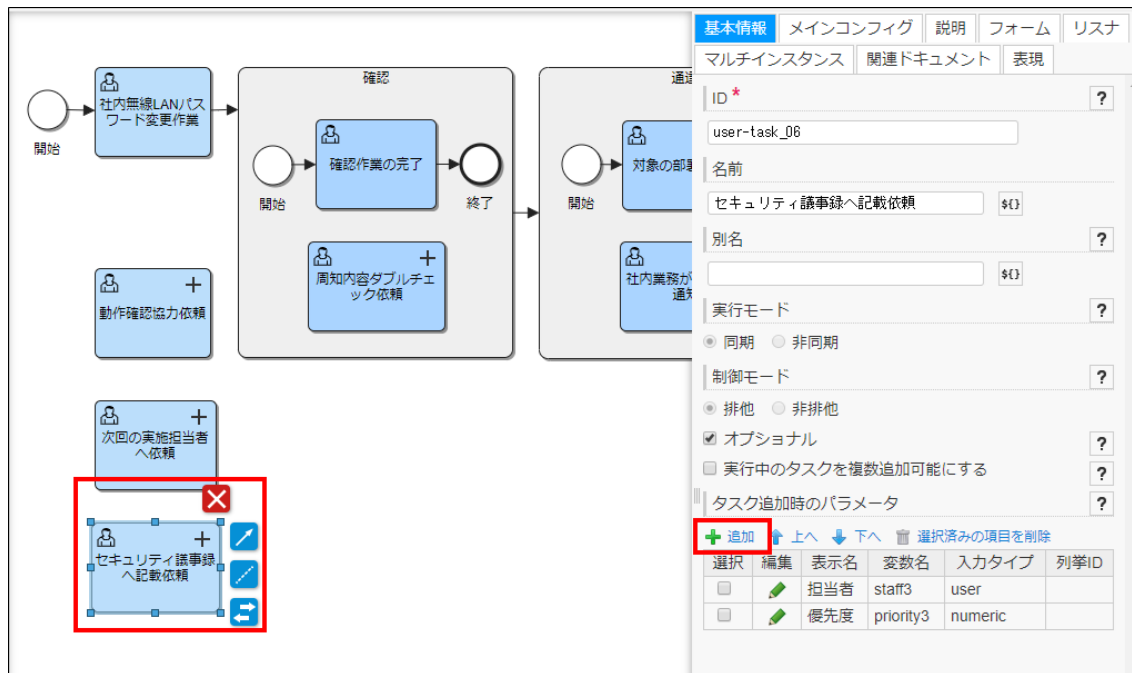
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

16. 上記と同じ手順で、3つ目のユーザタスクを追加します。
「ユーザタスク」を設置します。
17. 設置したユーザタスクをオプションタスクにします。
「ユーザタスク」をクリックし、「基本情報」タブから以下のように項目を設定してください。
 - 名前：セキュリティ議事録へ記載依頼
 - オptional：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

18. 「セキュリティ議事録へ記載依頼」タスクを追加する際に、指定できるパラメータの設定をします。
「基本情報」タブから「タスク追加時のパラメータ」の「追加」リンクをクリックしてください。

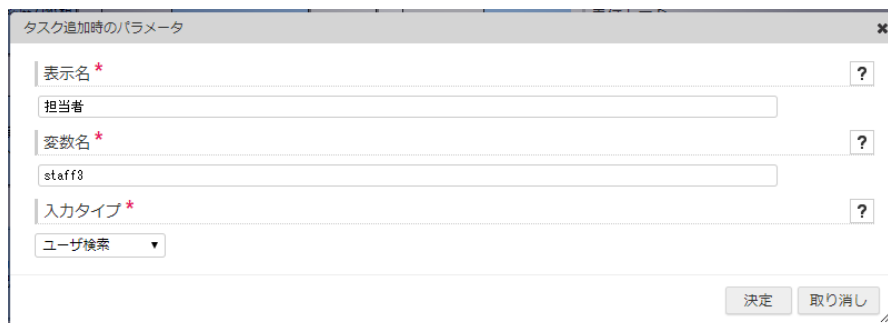


図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

19. 「タスク追加時のパラメータ」ダイアログが表示されます。

以下のように、2つのパラメータを設定してください。

- 担当者
表示名：担当者
変数名：staff3
入カタイプ：ユーザ検索
- 優先度
表示名：優先度
変数名：priority3
入カタイプ：数字

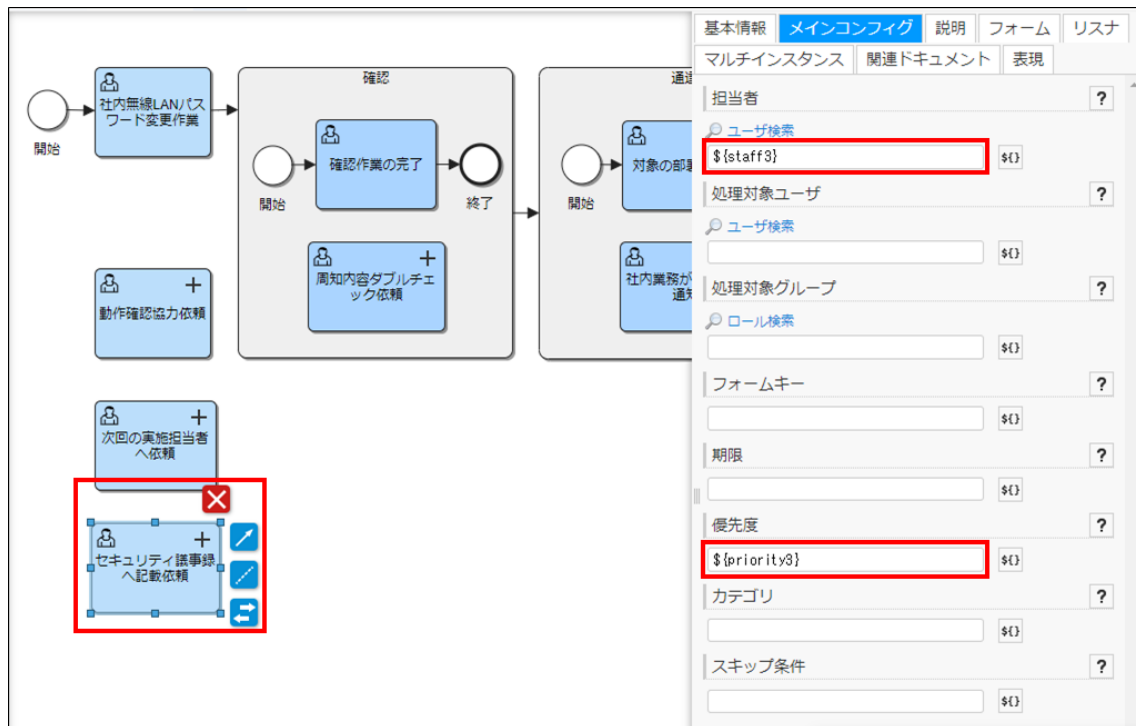


図：「タスク追加時のパラメータ」

20. EL式を使用し、指定されたパラメータが各項目に設定されるようにします。

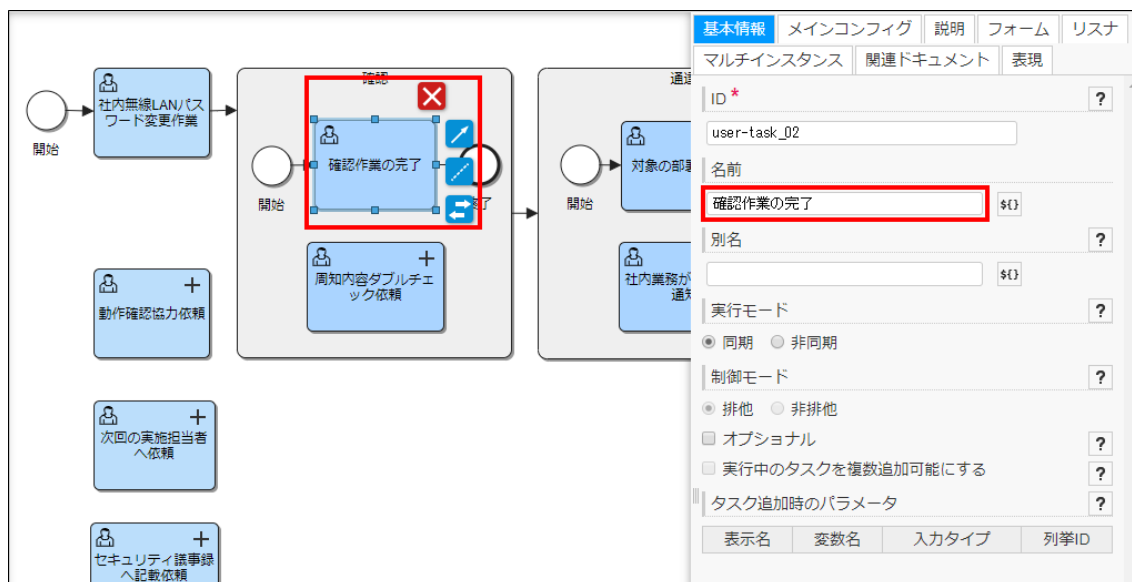
「ユーザタスク」をクリックし、「メインコンフィグ」タブから以下の項目を設定します。

- 担当者：\${staff3}
- 優先度：\${priority3}



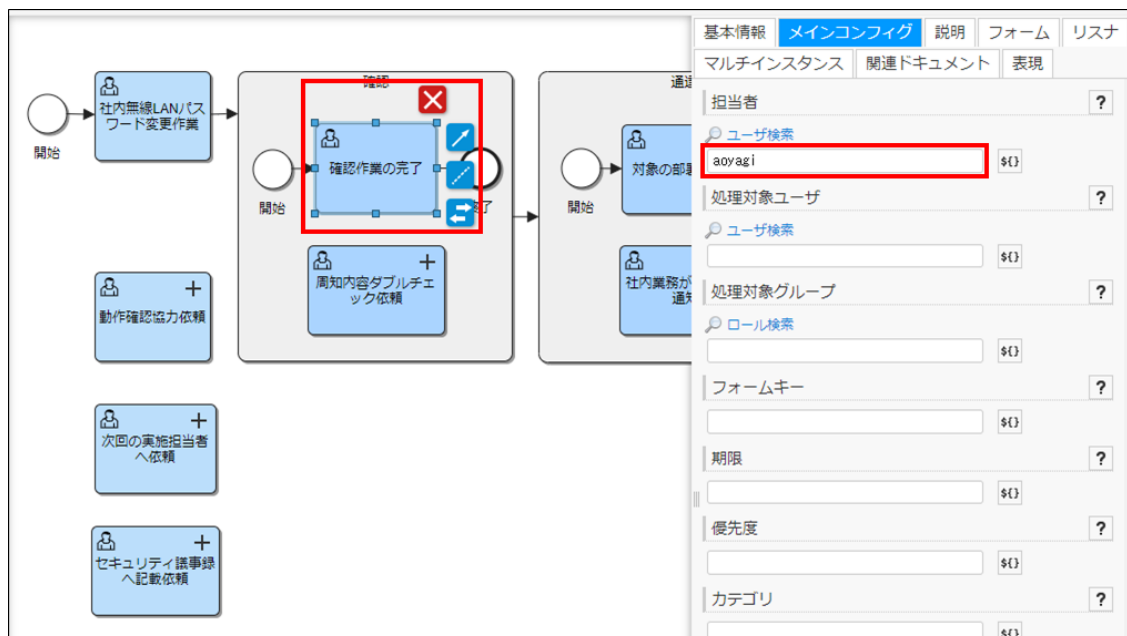
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

21. 実行状態によりタスクを編集できなくなることを確認するため、1つ目のサブプロセスを設置します。
「サブプロセス」を追加します。
22. サブプロセスに「開始イベント」を設置します。
23. サブプロセスに「ユーザタスク」を設置します。
24. 動作確認の際に「タスク一覧」画面でわかりやすくするため、設置したユーザタスクに名前を付けます。
「ユーザタスク」を選択し、「基本情報」タブから「名前」に 確認作業の完了 と設定します。



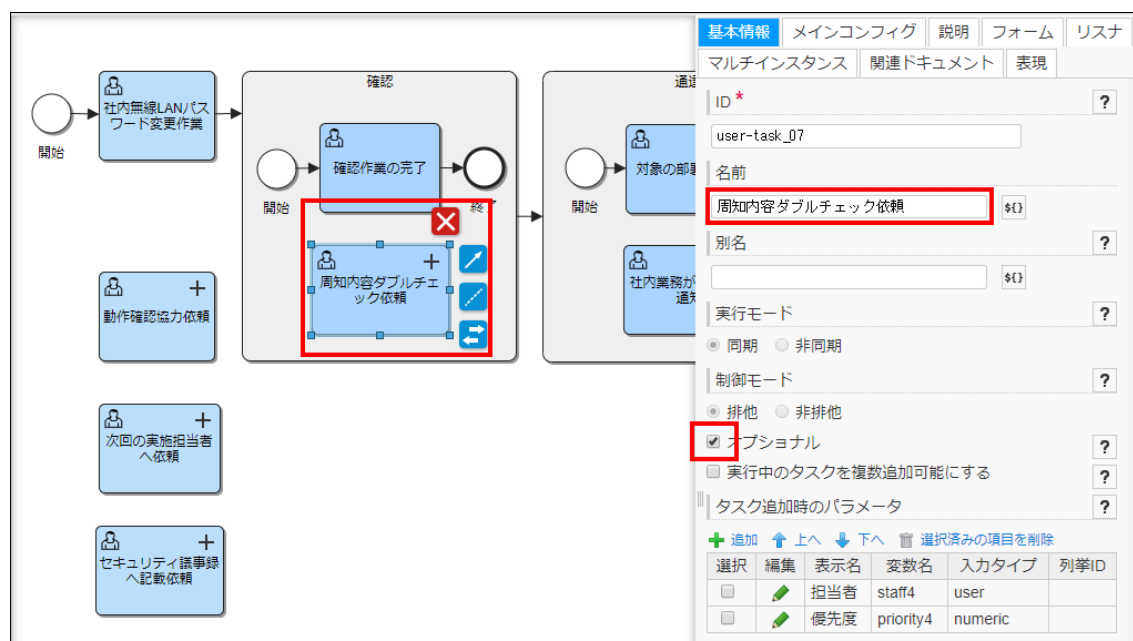
図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

25. 「確認作業の完了」タスクの担当者を設定をします。
「ユーザタスク」をクリックし、「メインコンフィグ」タブから、担当者に aoyagi と設定します。



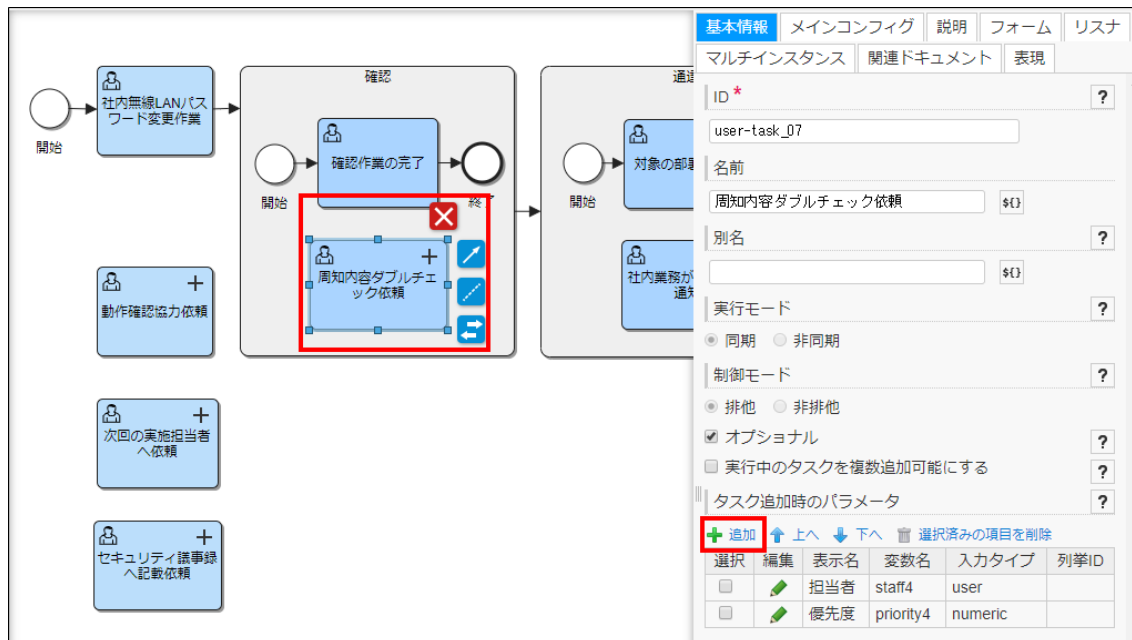
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

26. 「終了イベント」を設置します。
27. サブプロセス内にオプションタスクを設置します。
「ユーザタスク」を設置します。
28. 設置したユーザタスクをオプションタスクにします。
「ユーザタスク」をクリックし、「基本情報」タブから以下のように項目を設定してください。
 - 名前：周知内容ダブルチェック依頼
 - オptional：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

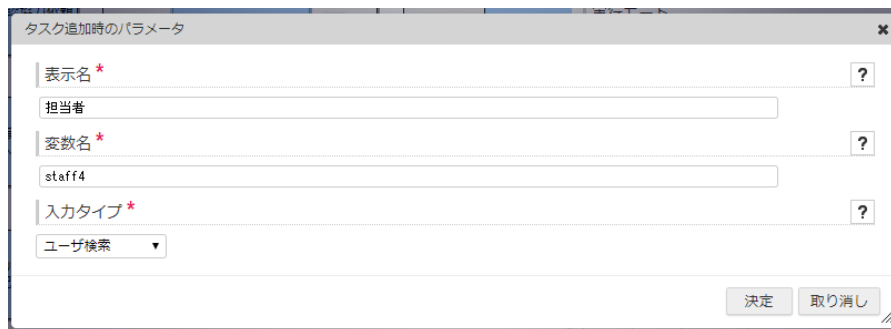
29. 「周知内容ダブルチェック依頼」タスクを追加する際に、指定できるパラメータの設定をします。
「ユーザタスク」をクリックし、「基本情報」タブから「タスク追加時のパラメータ」の「追加」リンクをクリックします。



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

30. 「タスク追加時のパラメータ」ダイアログが表示されます。
以下のように、2つのパラメータを設定してください。

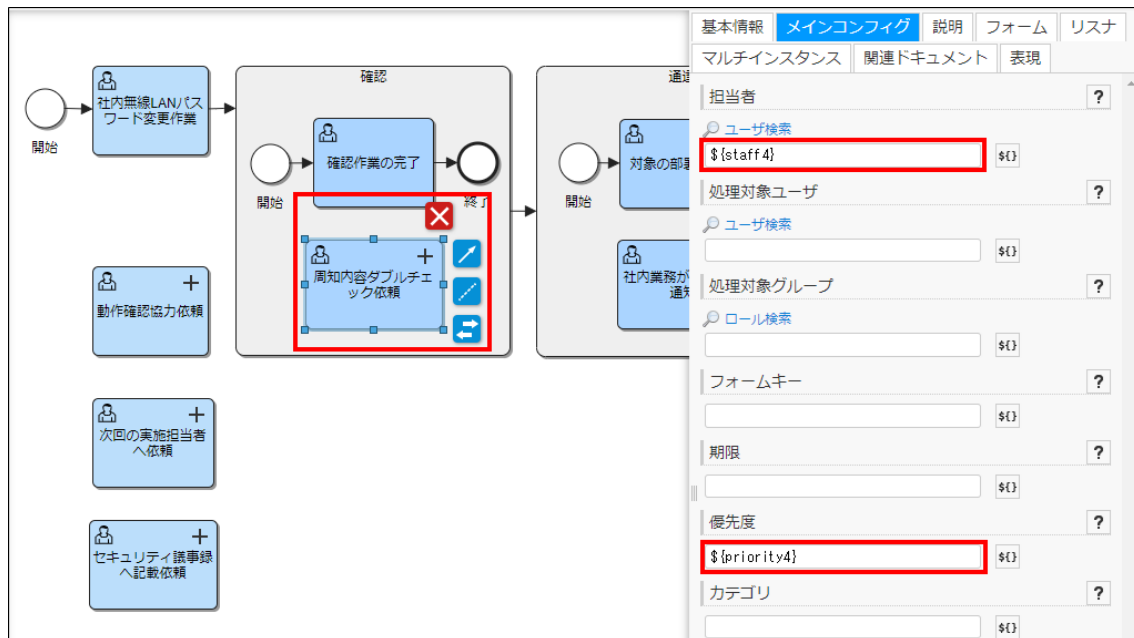
- 担当者
表示名：担当者
変数名：staff4
入力タイプ：ユーザ検索
- 優先度
表示名：優先度
変数名：priority4
入力タイプ：数字



図：「タスク追加時のパラメータ」

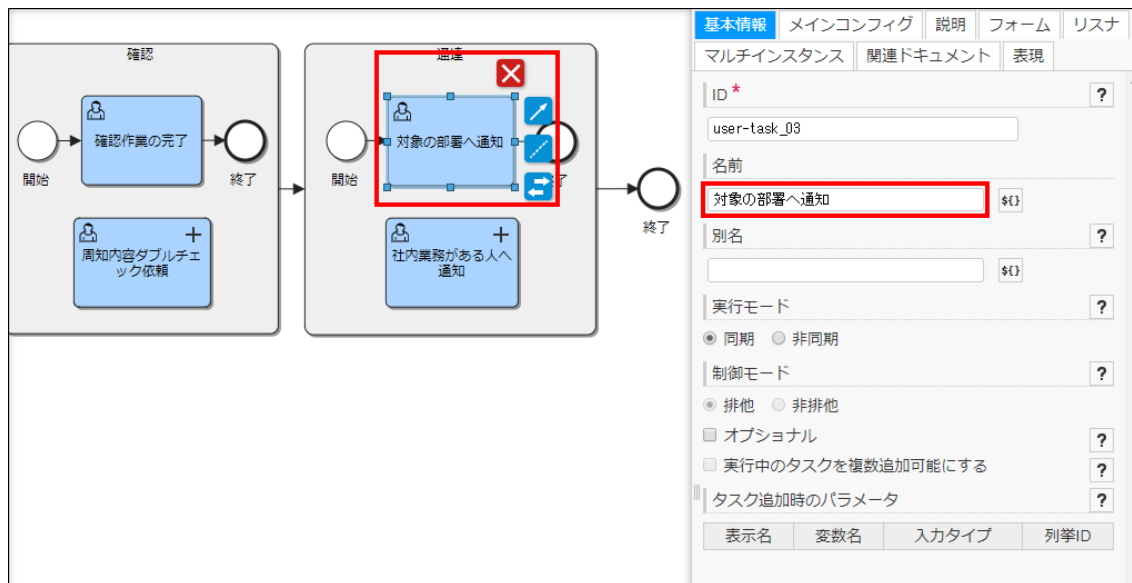
31. EL式を使用し、指定されたパラメータが各項目に設定されるようにします。
「ユーザタスク」をクリックし、「メインコンフィグ」タブから以下の項目を設定します。

- 担当者：\${staff4}
- 優先度：\${priority4}



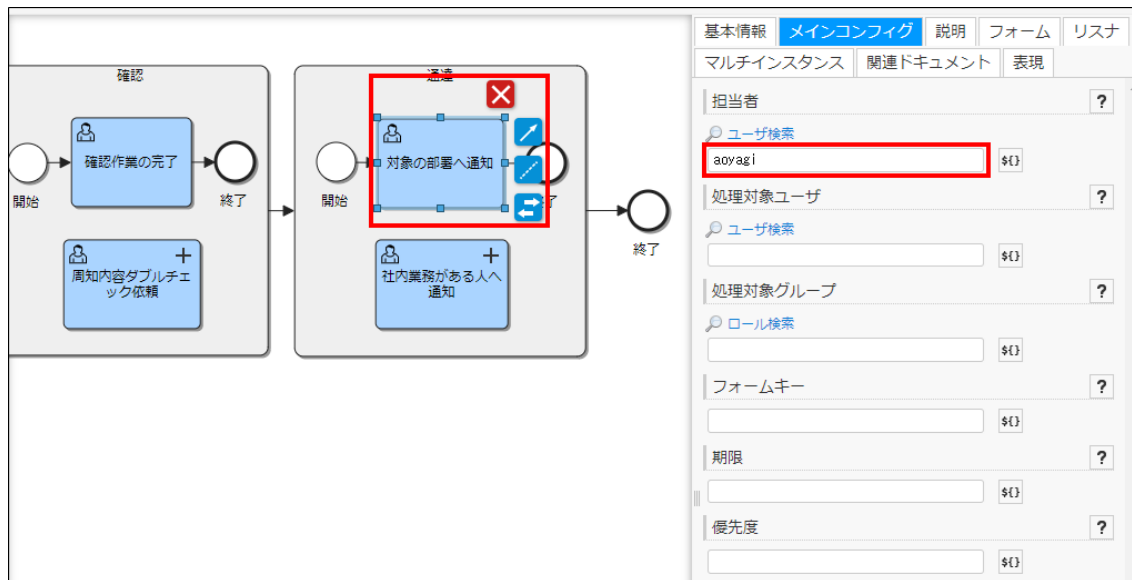
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

32. 実行状態によりタスクを編集できなくなることを確認するため、2つ目のサブプロセスを設置します。
「サブプロセス」を追加します。
33. サブプロセスに「開始イベント」を設置します。
34. サブプロセスに「ユーザタスク」を設置します。
35. 動作確認の際に「タスク一覧」画面でわかりやすくするため、設置したユーザタスクに名前を付けます。
「ユーザタスク」を選択し、「基本情報」タブから「名前」に 対象の部署へ通知 と設定します。



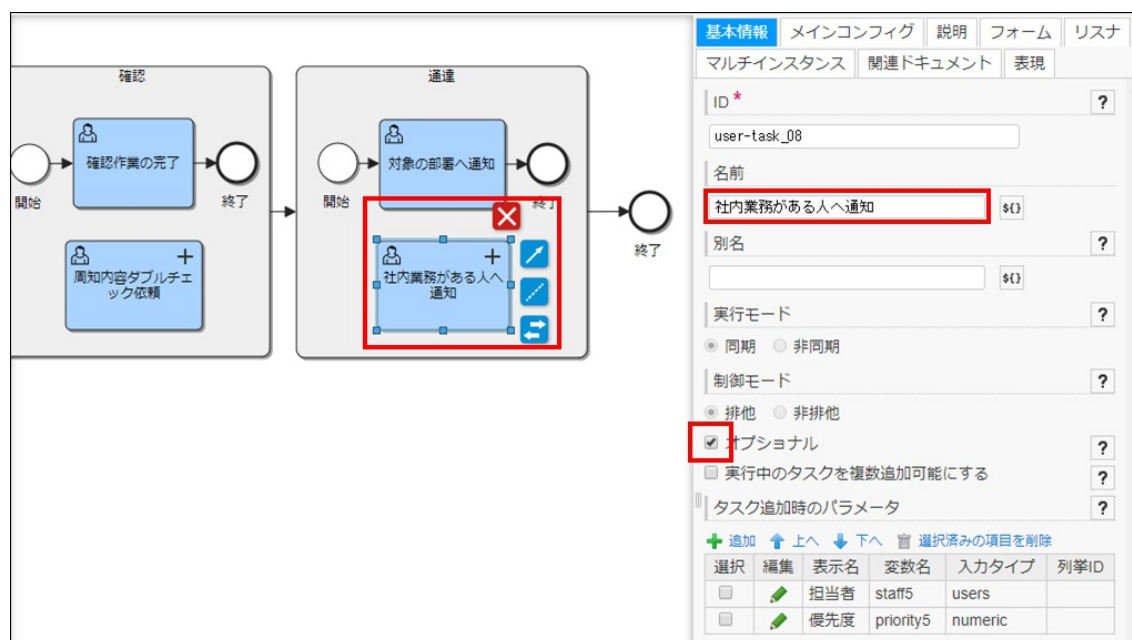
図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

36. 「対象の部署へ通知」タスクの担当者を設定をします。
「ユーザタスク」をクリックし、「メインコンフィグ」タブから、担当者に aoyagi と設定します。



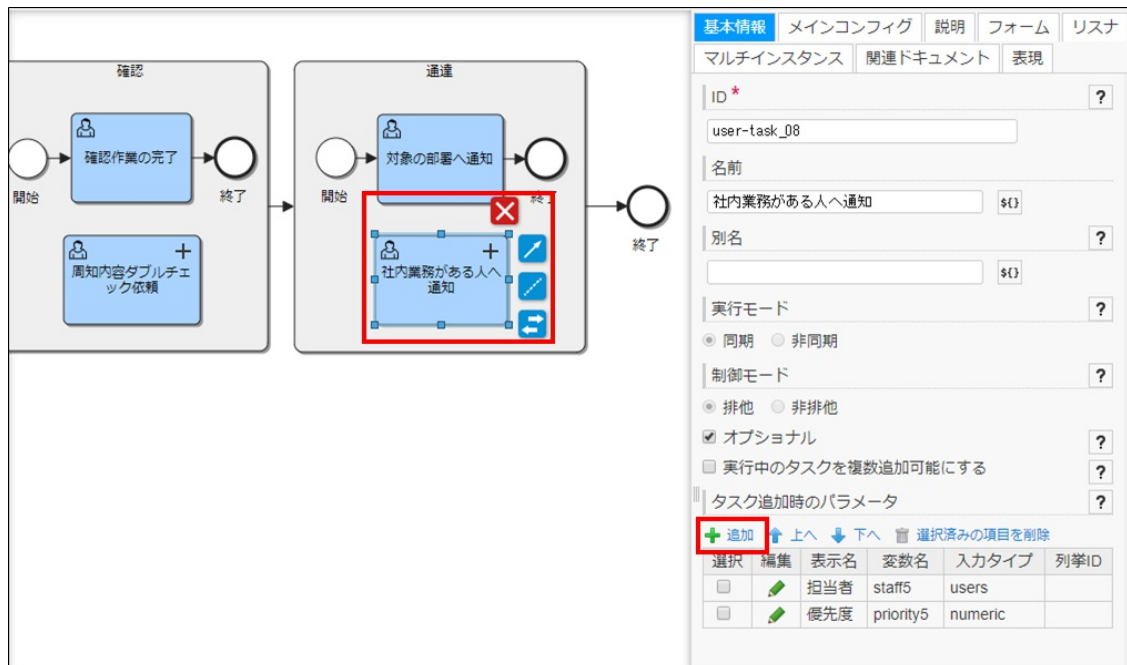
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

37. 「終了イベント」を設置します。
38. サブプロセス内にオプションタスクを設置します。
「ユーザタスク」を設置します。
39. 設置したユーザタスクをオプションタスクにします。
「ユーザタスク」をクリックし、「基本情報」タブから以下のように項目を設定してください。
 - 名前：社内業務がある人へ通知
 - オプション：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

40. 「社内業務がある人へ通知」タスクを追加する際に、指定できるパラメータの設定をします。
「ユーザタスク」をクリックし、「基本情報」タブから「タスク追加時のパラメータ」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」 - 「基本情報」

41. 「タスク追加時のパラメータ」ダイアログが表示されます。

以下のように、2つのパラメータを設定してください。

- 担当者
表示名：担当者
変数名：staff5
入力タイプ：ユーザ検索
- 優先度
表示名：優先度
変数名：priority5
入力タイプ：数字

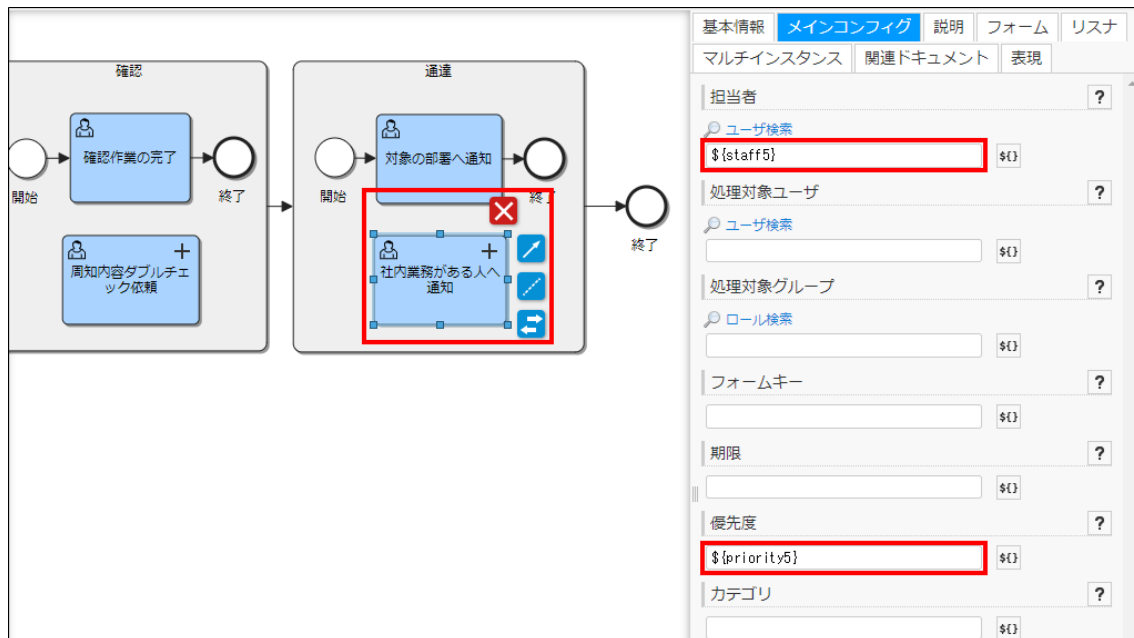


図：「タスク追加時のパラメータ」

42. EL式を使用し、指定されたパラメータが各項目に設定されるようにします。

「ユーザタスク」をクリックし、「メインコンフィグ」タブから以下の項目を設定します。

- 担当者：\${staff5}
- 優先度：\${priority5}



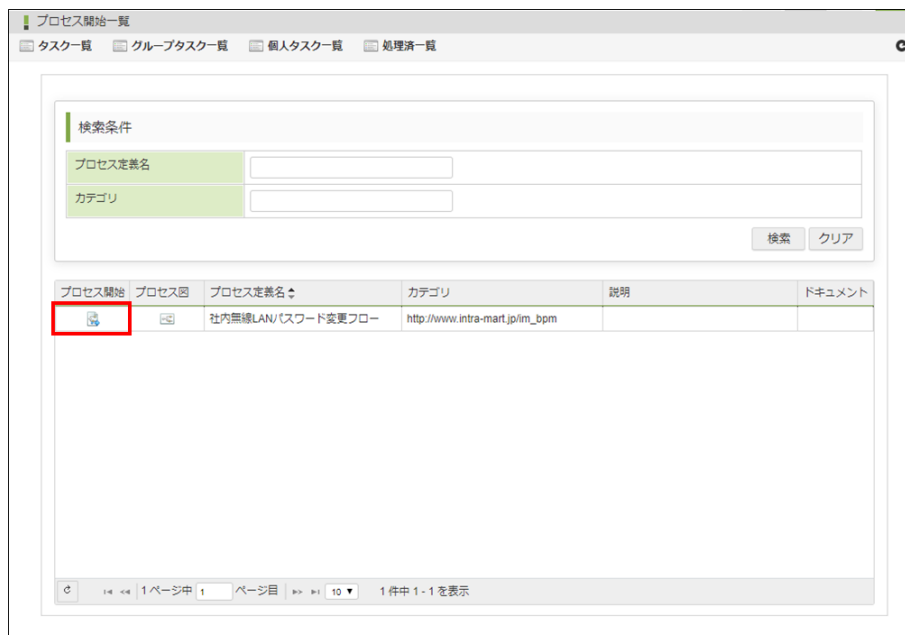
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

43. 「終了イベント」を設置します。

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. プロセスを開始します。
「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」画面から、対象となるプロセスの「」アイコンをクリックします。



図：「プロセス開始一覧」

3. プロセス定義で「プロセス開始時にタスク追加画面を表示する」と設定したため、「タスク追加」画面が表示されます。



図：「タスク追加」

コラム

メニューバーの「プロセス図表示」をクリックすると、プロセス図を見ながら動作確認ができます。

4. オプショナルタスクを追加します。
プロセス直下のオプショナルタスク一覧から、「動作確認協力依頼」タスクをクリックします。



図：「タスク追加」

5. 「タスクの追加」ダイアログが表示されます。
今回のチュートリアルでは結果確認動作を簡易化するため、全ての担当者に **aoyagi** を指定します。
パラメータを以下のように設定し、「決定」ボタンをクリックします。

- 担当者：aoyagi
- 優先度：1



図：「タスクの追加」

コラム

担当者の「検索」ダイアログから設定できるユーザは、「組織所属」、または「パブリックグループ」が設定されているユーザです。
タスクを処理するには「BPMユーザ」が設定されている必要があります。

6. 「動作確認協力依頼」のタスクが追加されたことを確認します。
また、オプショナルタスク一覧にある「動作確認協力依頼」のタスクがグレーアウトしていることを確認します。



図：「タスク追加」

コラム

グレーアウトしているオプションタスクは、タスク一覧に追加したタスクが処理されることで再度追加できます。同時に複数タスクを追加したい場合、プロセスエディタで対象のタスクを選択し、「基本情報」タブから「実行中のタスクを複数追加可能にする」にチェックを入れます。ただし、実行されていないタスクは、実行中のタスクを複数追加可能であっても複数追加することはできません。

- 「確認」サブプロセスに対して、オプションタスクを追加します。
「確認」サブプロセスのオプションタスク一覧から、「周知内容ダブルチェック依頼」をクリックします。



図：「タスク追加」

- 「タスクの追加」ダイアログが表示されます。
パラメータを以下のように設定し、「決定」ボタンをクリックします。
 - 担当者：aoyagi
 - 優先度：3



図：「タスクの追加」

- 「周知内容ダブルチェック依頼」のタスクが追加されたことを確認します。
また、オプションタスク一覧にある「周知内容ダブルチェック依頼」タスクがグレーアウトしていることを確認します。



図：「タスク追加」

10. 「通達」サブプロセスに対して、オプションタスクを追加します。
 「通達」サブプロセスのオプションタスク一覧から、「社内業務がある人への通知」をクリックします。



図：「タスク追加」

11. 「タスクの追加」ダイアログが表示されます。
 パラメータを以下のように設定し、「決定」ボタンをクリックします。
- 担当者：aoyagi
 - 優先度：3

タスクの追加(社内業務がある人への通知)

担当者

aoyagi

優先度

3

決定

取り消し

図：「タスクの追加」

12. 「社内業務がある人への通知」のタスクが追加されたことを確認します。
 また、オプションタスク一覧にある「社内業務がある人への通知」タスクがグレーアウトしていることを確認します。



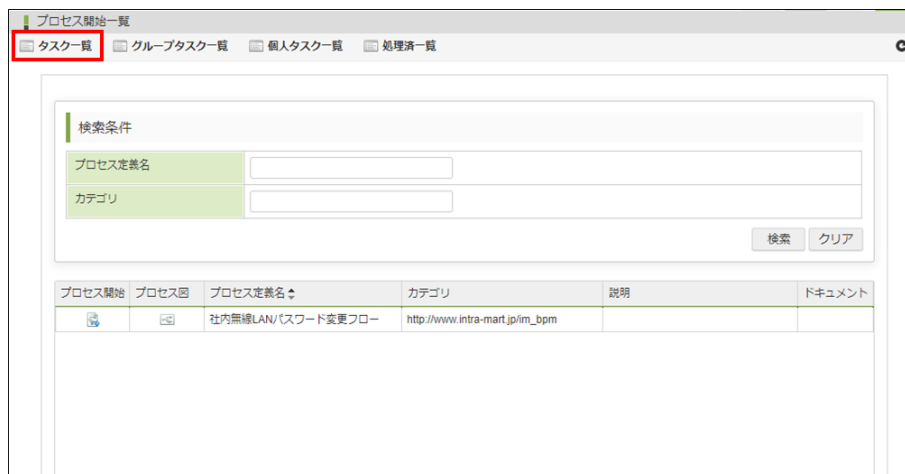
図：「タスク追加」

13. 「プロセス開始」ボタンをクリックして、プロセスを開始します。



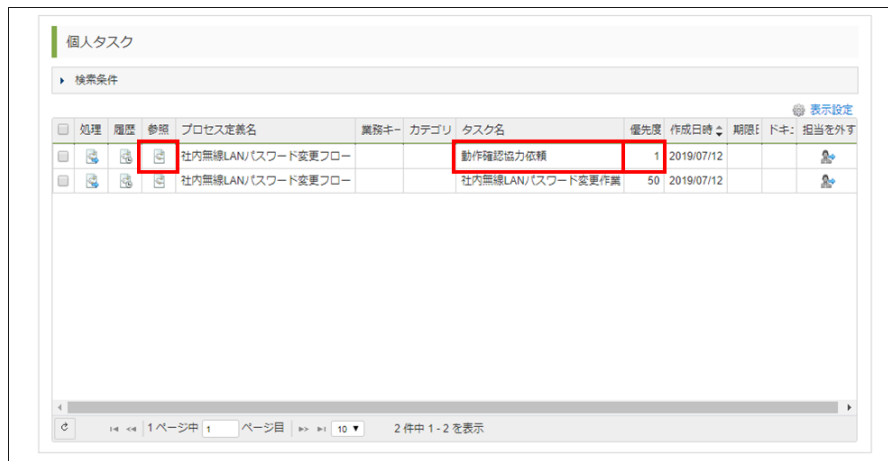
図：「タスク追加」

14. 開始されたプロセスを確認します。
「プロセス開始一覧」画面のツールバーにある「タスク一覧」をクリックし、「タスク一覧」画面を表示します。



図：「プロセス開始一覧」

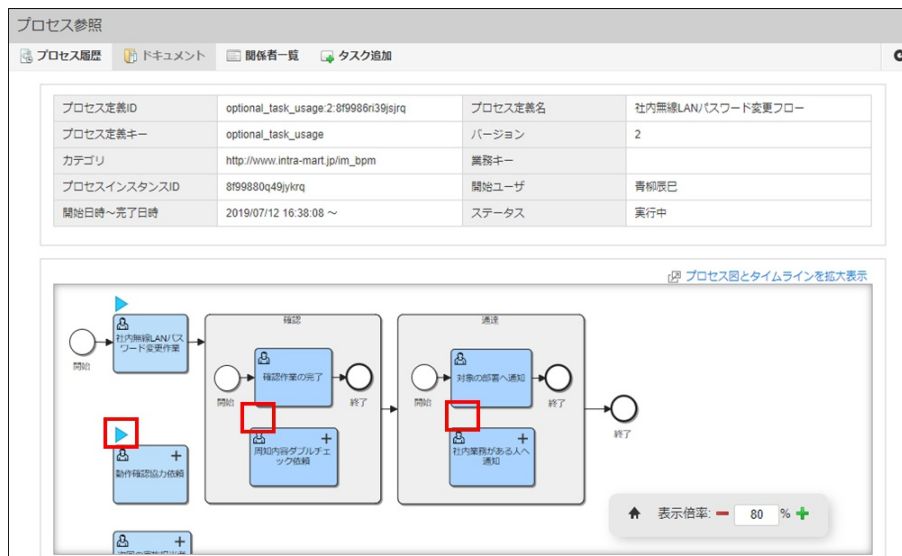
15. 対象のプロセス定義から、「動作確認協力依頼」タスクが優先度「1」で追加されていることを確認し、「」をクリックします。



図：「タスク一覧」 - 「個人タスク」

16. 表示される「プロセス参照」画面から、以下のことを確認します。

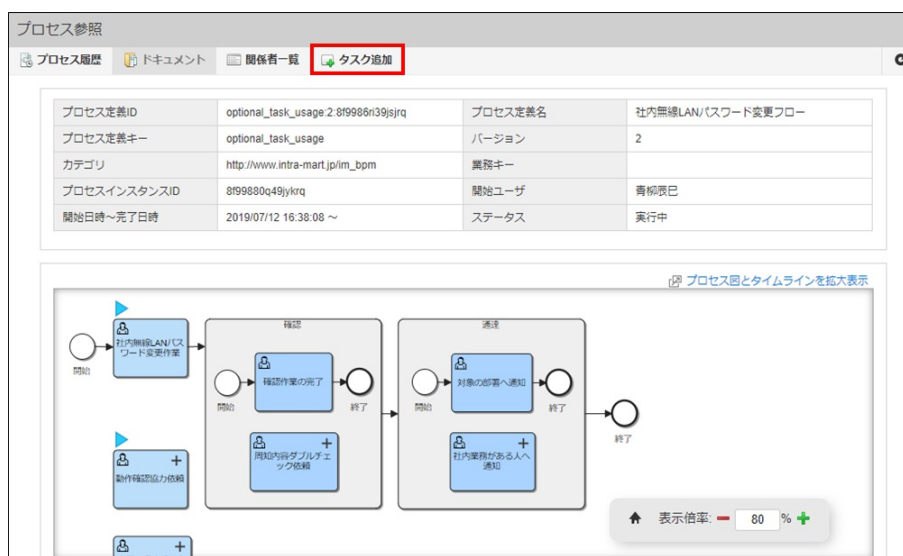
- 「タスク追加」画面で追加した、「動作確認協力依頼」タスクに「▶」マークがついていること
- 「タスク追加」画面で追加した「内容ダブルチェック依頼」タスクはまだ実行範囲ではないため、「▶」マークがついていないこと
- 「タスク追加」画面で追加した「社内業務がある人へ通知」タスクはまだ実行範囲ではないため、「▶」マークがついていないこと



図：「プロセス参照」

17. 「タスク追加」画面からタスクを追加・編集します。

ツールバーの「タスク追加」ボタンをクリックし、「タスク追加」画面を表示します。



図：「プロセス参照」

18. 「タスク追加」画面が表示されます。

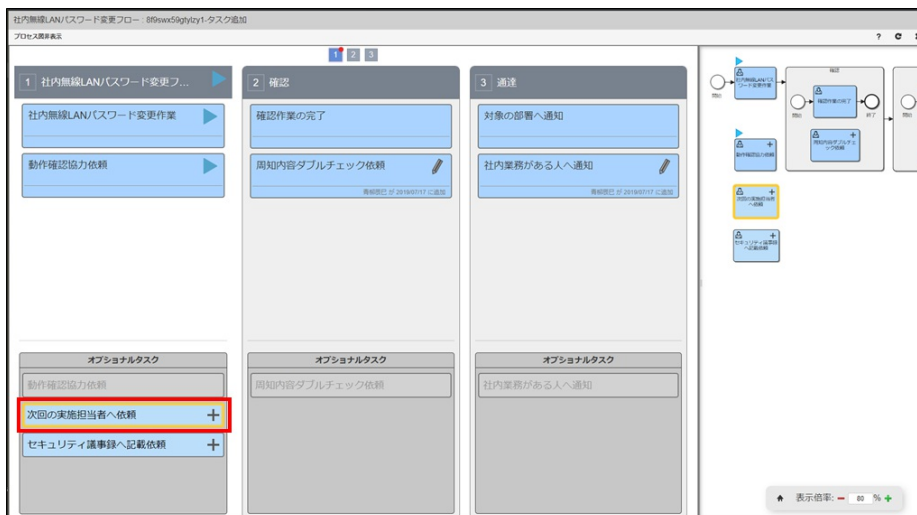
以下のことを確認します。

- 「動作確認協力依頼」タスク
実行中のため、「▶」マークがついていること
既に実行中のため、「✎」マークがないこと
- 「周知内容ダブルチェック依頼」タスク
プロセスが「確認」サブプロセスに到達していないため、「▶」マークがついていないこと
まだ実行できる状態でなく編集できるため、「✎」マークがついていること
- 「社内業務がある人へ通知」タスク
プロセスが「通達」サブプロセスに到達していないため、「▶」マークがついていないこと
まだ実行できる状態でなく編集できるため、「✎」マークがついていること
- 既に追加されたオプションタスクは、グレーアウトしていること



図：「タスク追加」

19. 実行中のプロセスに対して、オプションタスクを追加します。
プロセスのオプションタスク一覧から、「次回実行担当者へ依頼」タスクをクリックします。



図：「タスク追加」

20. 「タスクの追加」ダイアログが表示されます。
パラメータを以下のように設定し、「決定」ボタンをクリックします。
- 担当者：aoyagi
 - 優先度：4



図：「タスクの追加」

21. 「次回の実施担当者へ依頼」タスクが追加されたことを確認します。

プロセスに追加されたタスクはすぐに実行されるため、即時「▶」マークがつきます。

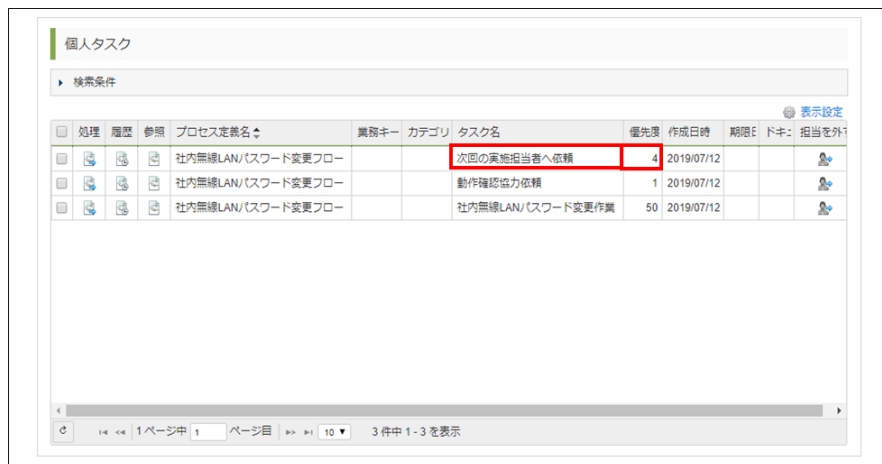
また、オプションタスク一覧にある「次回の実施担当者へ依頼」タスクがグレーアウトしていることを確認します。



図：「タスク追加」

22. 「タスク一覧」画面から、オプションタスクが追加されたことを確認します。

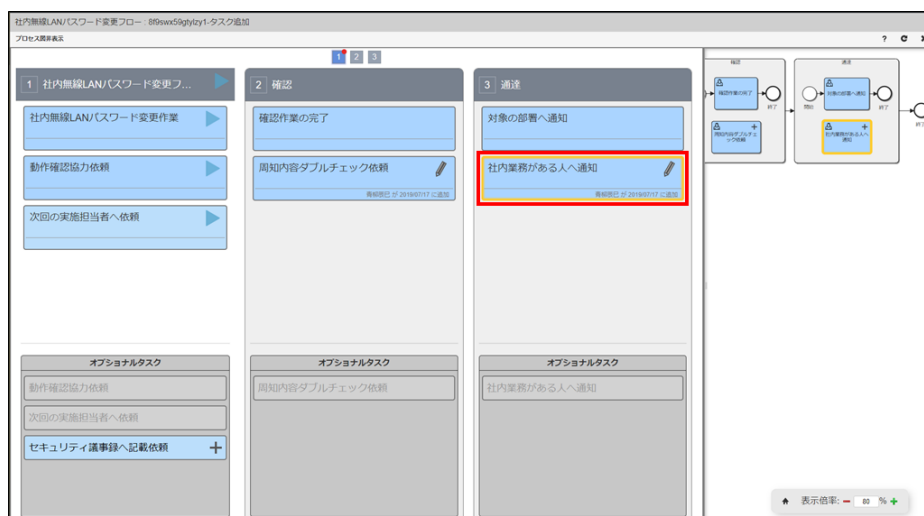
「個人タスク」に「次回の実施担当者へ依頼」タスクが優先度「4」で追加されていることを確認します。



図：「タスク一覧」- 「個人タスク」

23. 追加したオプションタスクを削除するため、「タスク追加」画面へ戻ります。

「通達」サブプロセス内の「社内業務がある人へ通知」タスクに「✎」マークがついているので、クリックします。



図：「タスク追加」

24. 「タスクの編集」ダイアログが表示されます。

ツールバーの「削除」ボタンをクリックします。

図：「タスクの編集」



コラム

実行範囲にないオプションタスクは、設定したパラメータを編集できます。
「タスクの編集」ダイアログから、パラメータの値を編集し、「決定」ボタンをクリックして保存します。

25. 「社内業務がある人へ通知」タスクがタスク一覧から消え、オプションタスク一覧に戻ります。

図：「タスク追加」

26. タスクを処理して、プロセスを進行させます。
「動作確認協力依頼」タスクと、「社内無線LAN/パスワード変更作業」タスクを処理します。

処理	履歴	参照	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限	ドキュメント	担当を外す
			社内無線LAN/パスワード変更フロー			次回の実施担当者へ依頼	4	2019/07/12			
			社内無線LAN/パスワード変更フロー			動作確認協力依頼	1	2019/07/12			
			社内無線LAN/パスワード変更フロー			社内無線LAN/パスワード変更作業	50	2019/07/12			

図：「タスク一覧」- 「個人タスク」

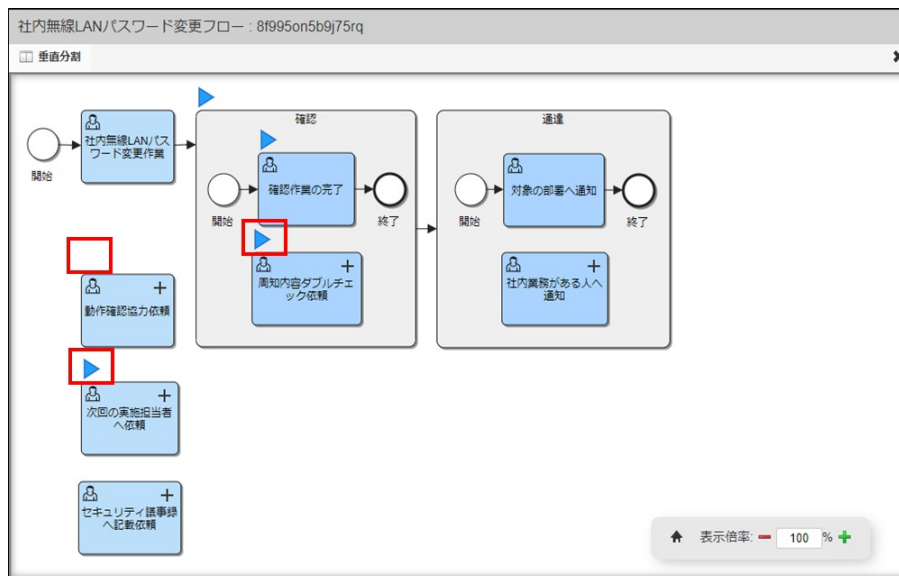
27. プロセスの状態を確認します。
タスク開始時に追加した「周知内容ダブルチェック依頼」タスクが優先度「2」で実行されているのを確認し、「」をクリックします。



図：「タスク一覧」- 「個人タスク」

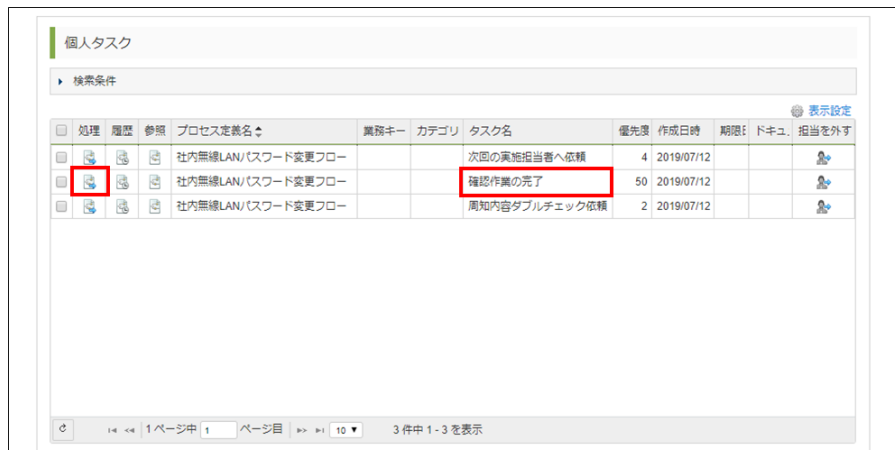
28. 「プロセス詳細」画面、または「プロセス図とタイムラインを拡大表示」から、プロセスの状態を確認します。
以下の事項を確認します。

- 処理した「動作確認協力依頼」に「▶」マークはついていないこと
プロセスの「次回の実施担当者へ依頼」は実行中であり、「▶」マークがついていること
- 「確認」サブプロセスが実行中のため、プロセス開始時に追加した「周知内容ダブルチェック」タスクに「▶」マークがついていること



図：「プロセス図とタイムラインを拡大表示」

29. 追加したオプションタスクが完了しないとサブプロセスが終了されないことを確認します。
「タスク一覧」画面から、「確認作業の完了」タスクを処理します。



図：「タスク一覧」- 「個人タスク」

30. プロセスの状態を確認します。

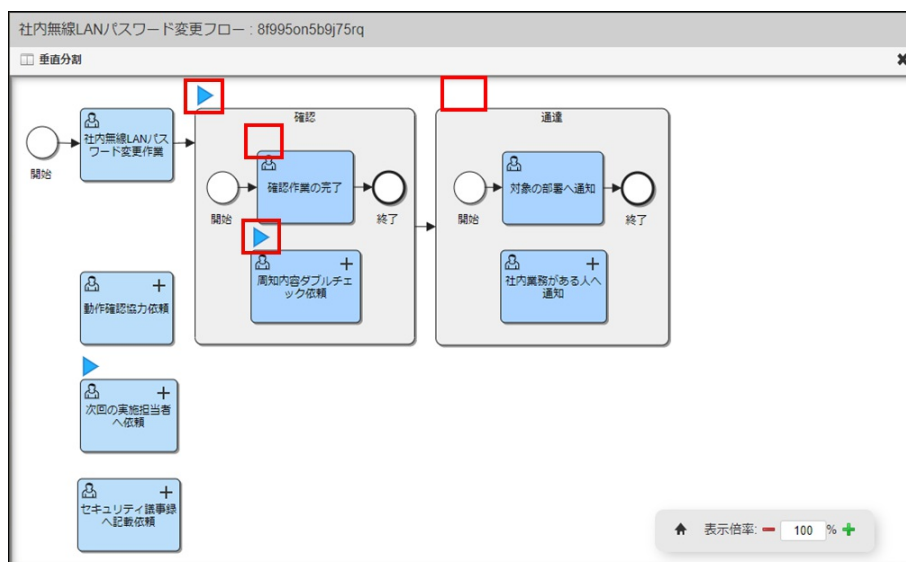
「タスク一覧」画面から、対象のプロセス定義の「」をクリックします。



図：「タスク一覧」-「個人タスク」

31. 「プロセス詳細」画面、または「プロセス図とタイムラインを拡大表示」から、プロセスの状態を確認します。
以下の事項を確認します。

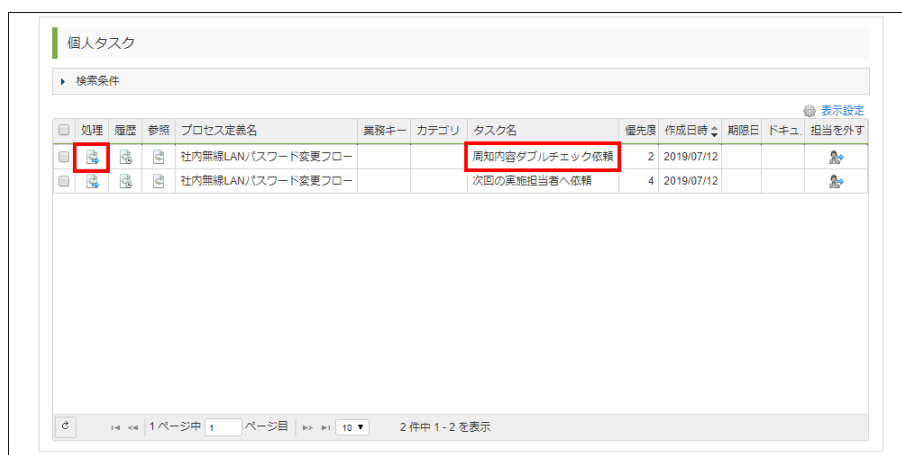
- 「確認」サブプロセスに「▶」マークがついていること
「確認」サブプロセスの「確認作業の完了」タスクは処理したため、「▶」マークがついていないこと
「確認」サブプロセスの「周知内容のダブルチェック」タスクは処理していないため、「▶」マークがついていること
- 「通達」サブプロセスは実行中ではないため、「▶」マークがついていないこと



図：「プロセス図とタイムラインを拡大表示」

32. オptionalタスクを処理し、「確認」サブプロセスを終了させます。

「周知内容ダブルチェック」タスクの「」をクリックし、タスクを処理します。



図：「タスク一覧」-「個人タスク」

33. プロセスの状態を確認します。

「タスク一覧」画面から、対象のプロセス定義の「」をクリックします。

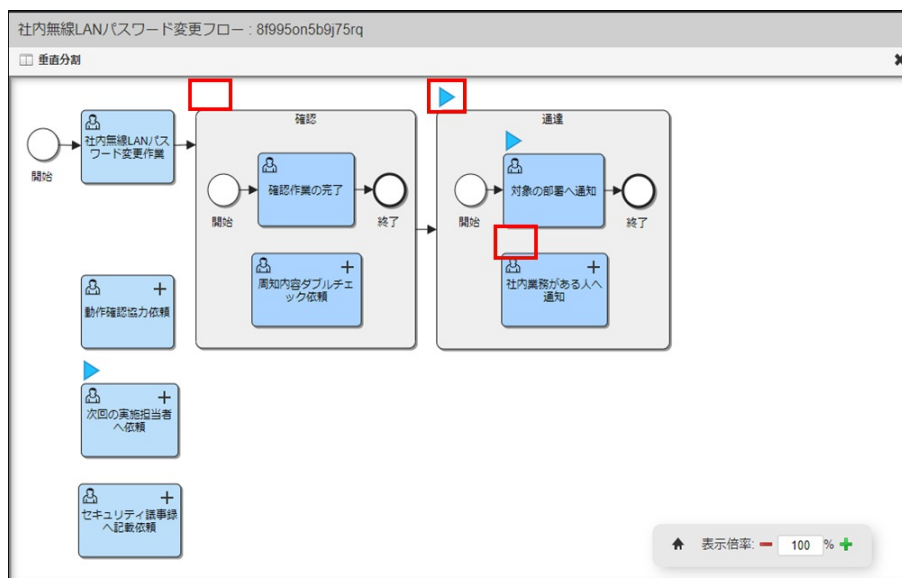


図：「タスク一覧」-「個人タスク」

34. 「プロセス詳細」画面、または「プロセス図とタイムラインを拡大表示」から、プロセスの状態を確認します。
以下の事項を確認します。

- 「確認」サブプロセスが終了し、「」マークがついていないこと
- 「通達」サブプロセスが実行中となり、「」マークがついていること

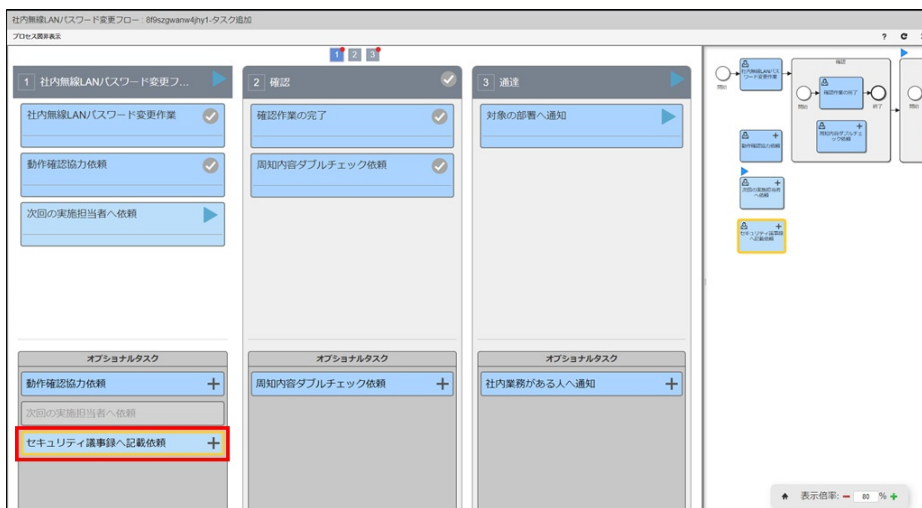
プロセス開始時に追加し、開始後に削除した「社内業務がある人へ通知」タスクに「」マークがついていないこと



図：「プロセス図とタイムラインを拡大表示」

35. プロセスに対してのオプションタスクを追加します。

「タスク追加」画面から、プロセスのオプションタスク一覧にある「セキュリティ議事録へ記載依頼」をクリックします。



図：「タスク追加」

36. 「タスクの追加」ダイアログが表示されます。

パラメータを以下のように設定し、「決定」ボタンをクリックします。

- 担当者: aoyagi
- 優先度: 5

図: 「タスクの追加」

37. 「セキュリティ議事録へ記載依頼」タスクが追加されたことを確認します。

プロセスに追加されたタスクはすぐに実行されるため、即時「▶」マークがつきます。

また、オプションタスク一覧にある「セキュリティ議事録へ記載依頼」タスクがグレイアウトしていることを確認します。

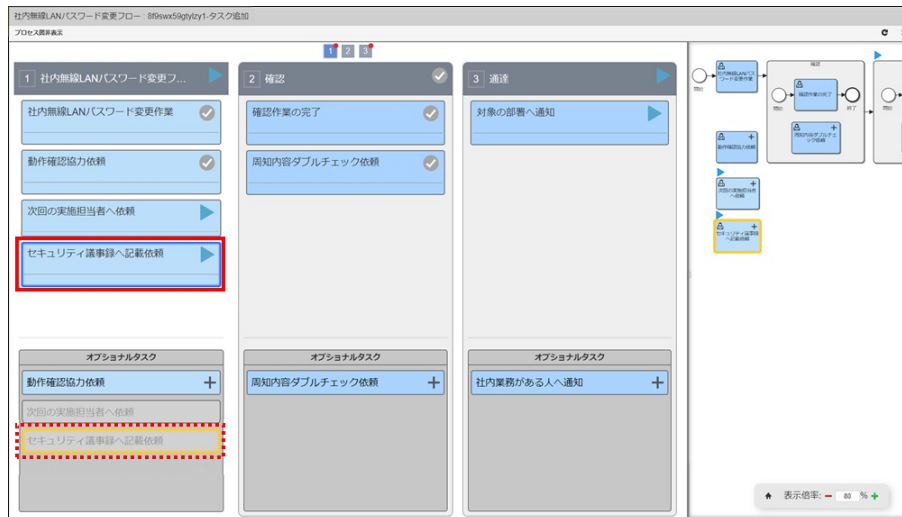


図: 「タスク追加」

38. 「通達」サブプロセスを終了し、プロセスを進行させます。

「タスク一覧」画面から、「対象の部署への通知」タスクと、優先度が「5」の「セキュリティ議事録」タスクを処理します。

処理	履歴	参照	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限E	ドキュメント	担当を外す
			社内無線LAN/パスワード変更フロー			セキュリティ議事録へ記載依頼	5	2019/07/18			
			社内無線LAN/パスワード変更フロー			対象の部署へ通知	50	2019/07/18			
			社内無線LAN/パスワード変更フロー			次回の実施担当者へ依頼	4	2019/07/18			

図: 「タスク一覧」 - 「個人タスク」

39. タスクの履歴を確認します。

タスク一覧に残っている「次回の実施担当者へ依頼」タスクの「」をクリックします。

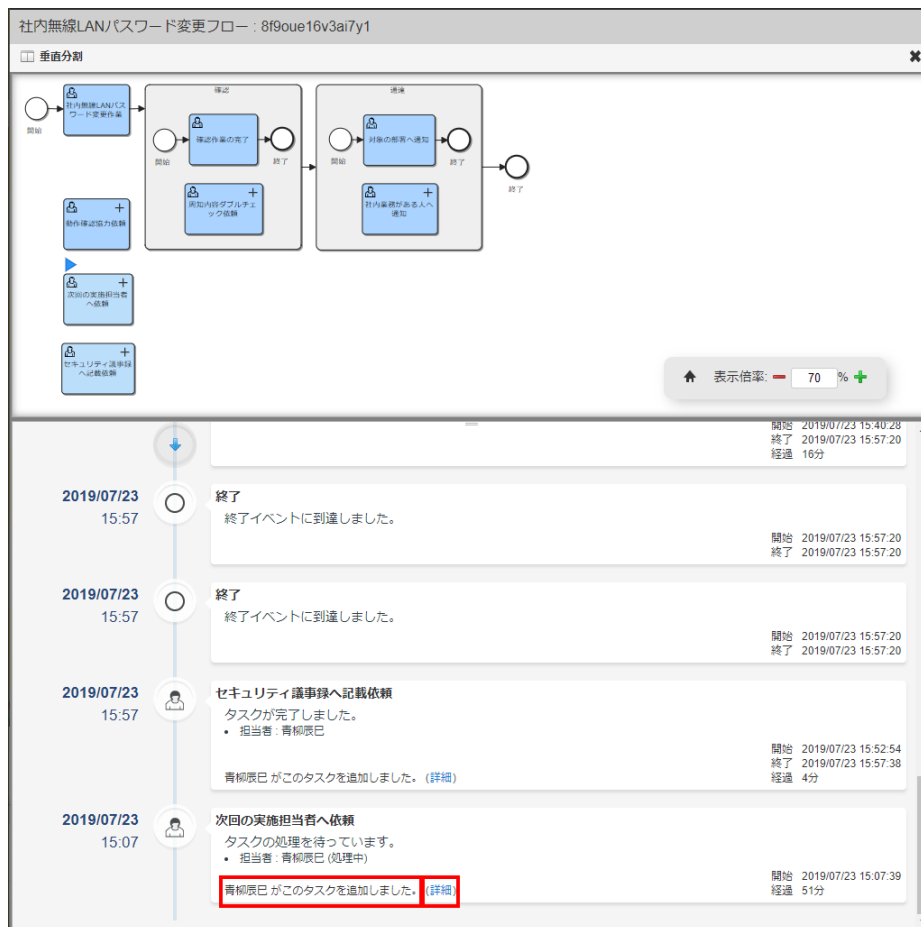


図：「タスクの追加」

40. 「プロセス詳細」画面、または、「プロセス図とタイムラインを拡大表示」画面を表示します。

プロセス履歴から、オプションタスクを追加したユーザが確認できます。

詳細リンクをクリックすると、「オプションタスク詳細」ダイアログが表示され、入力されたパラメータが確認できます。



図：「プロセス図とタイムラインを拡大表示」



図：「オプションタスク詳細」

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](#)

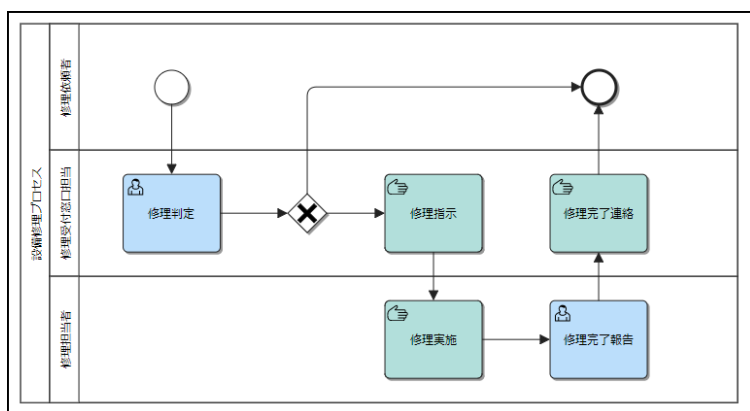
このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。

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

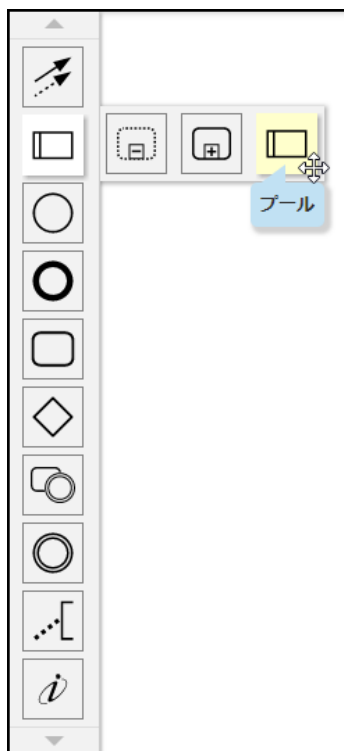
完成イメージ

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。



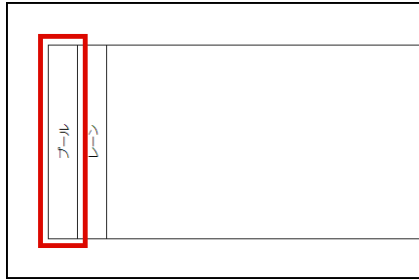
図：完成イメージ

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

- 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。
- 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。
- 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。



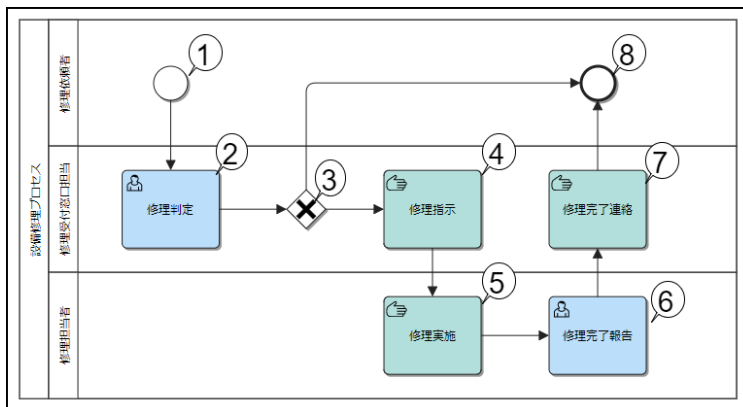
図：プールの選択

レーンを追加し、プロパティを設定する

- 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。
- 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

- 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	修理受付窓口担当	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

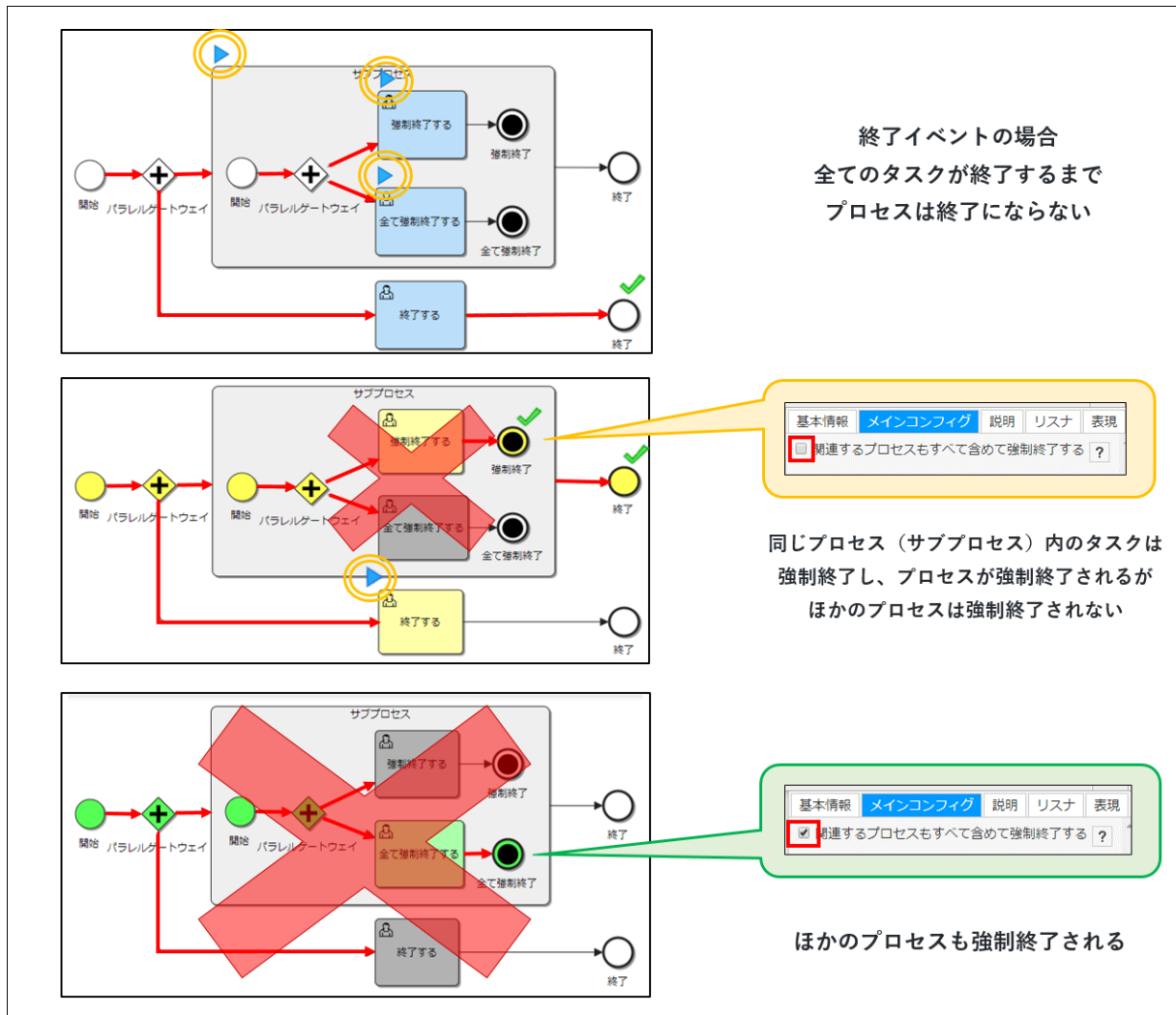
- 一つのファイル内に複数のプールを配置することもできます。
デプロイ時には、一つのプールが一つのプロセス定義としてデプロイされます。
- レーンには実行権限を制御する機能はありません。
レーンは業務の流れを視覚的に整理するエレメントです。

強制終了イベント

強制終了イベントを利用して、プロセスを強制終了する

このチュートリアルでは、強制終了イベントを使用して、プロセスを途中で終了できる定義の作成方法を解説します。
一部のプロセスを終了したい場合、または、すべてのプロセスを終了したい場合に使用します。

強制終了イベントの詳細については、「IM-BPM プロセスデザイナー 操作ガイド」 - 「終了イベント」 - 「強制終了イベント」もあわせて参照してください。



図：概要図

i 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。

[terminate_end_event_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 プロセスデザイナー 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。

i コラム

このチュートリアルのサンプルでは、1つのBPMNファイルに3つのプロセス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。

- 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「設定が異なる強制終了イベント」と「終了イベント」に分岐しているプロセス定義図です。

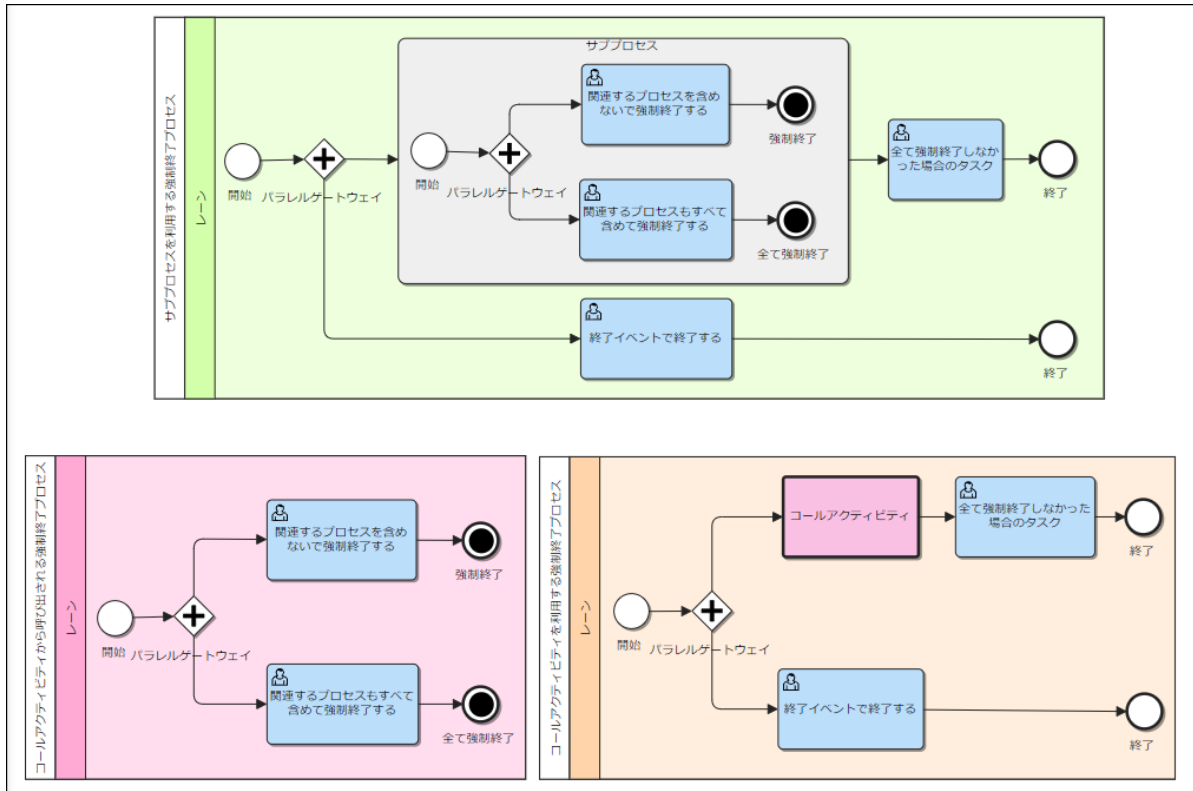
「強制終了イベント」の設定を変えることと「終了イベント」で、以下の3パターンの終了状態を確認できます。

- 実行中のプロセス（サブプロセス）のみを終了して、次のフローエレメントへ進行する
- 「関連するプロセスもすべて含めて強制終了する」設定を有効にすることにより、実行中のプロセス以外のプロセスも終了する
- 通常の終了イベントで終了する

このチュートリアルでは、「サブプロセス」を利用するプロセスと、「コールアクティビティ」を利用するプロセスを作成します。

サブプロセスの詳細については、「IM-BPM プロセスデザイナー 操作ガイド」 - 「コンテナ」 - 「サブプロセス」を参照してください。

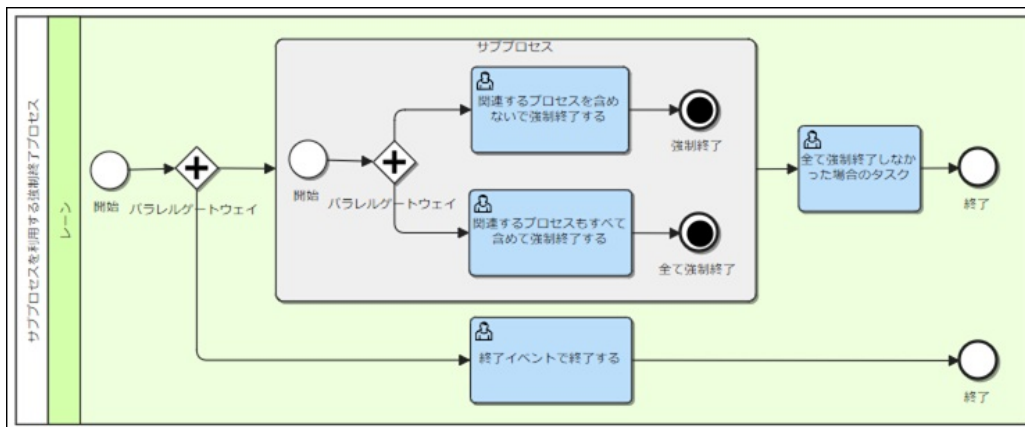
コールアクティビティの詳細については、「IM-BPM プロセスデザイナー 操作ガイド」 - 「タスク」 - 「コールアクティビティ」を参照してください。



図：「サブプロセスを利用する強制終了プロセス」、「コールアクティビティから呼び出される強制終了するプロセス」、「コールアクティビティを利用する強制終了プロセス」

サブプロセスを利用して、強制終了するプロセス定義を作成する

「サブプロセス」を利用して、異なる範囲を終了できるプロセスを作成します。

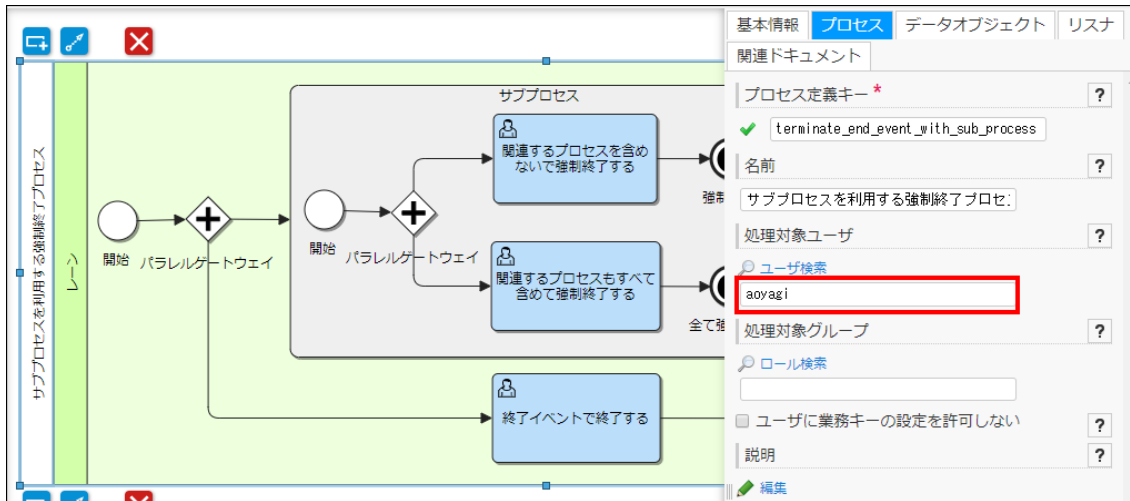


図：サブプロセスを利用する強制終了プロセス

1. 「プール」を設置します。
2. 「開始イベント」を設置します。

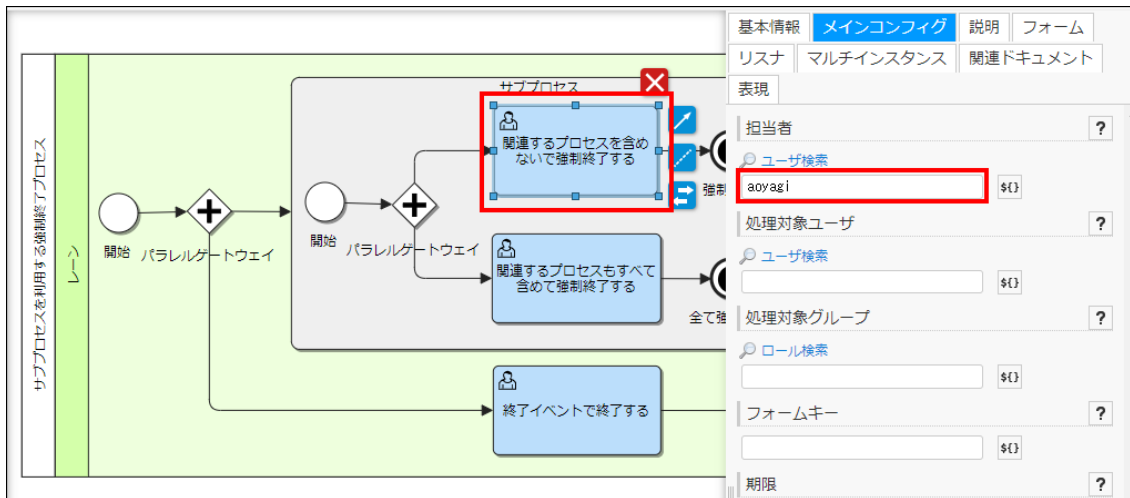
3. 「サブプロセスを利用する強制終了プロセス」の処理対象ユーザを設定します。

プールを選択した状態で、「プロセス」タブから処理対象ユーザに「aoyagi」と設定します。



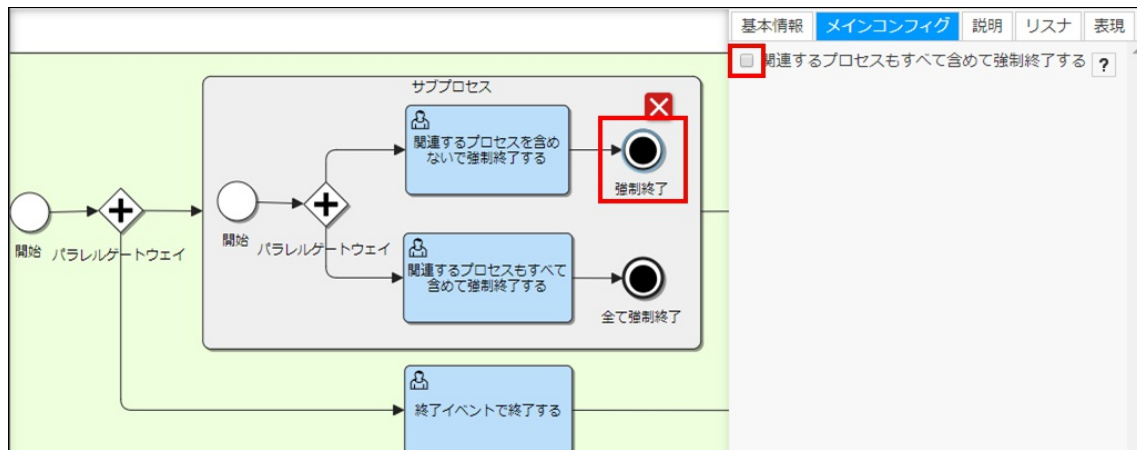
図：「プール」 - 「プロパティ」 - 「プロセス」 - 「処理対象ユーザ」

4. 「パラレルゲートウェイ」を設置します。
5. 「サブプロセス」を設置します。
6. サブプロセス内に「開始イベント」を設置します。
7. 「パラレルゲートウェイ」を設置します。
8. 関連するプロセスを含めないで強制終了するルートを作成します。
「ユーザタスク」を設置します。
9. 「関連するプロセスを含めないで強制終了する」の担当者を設定します。
「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「aoyagi」と設定します。



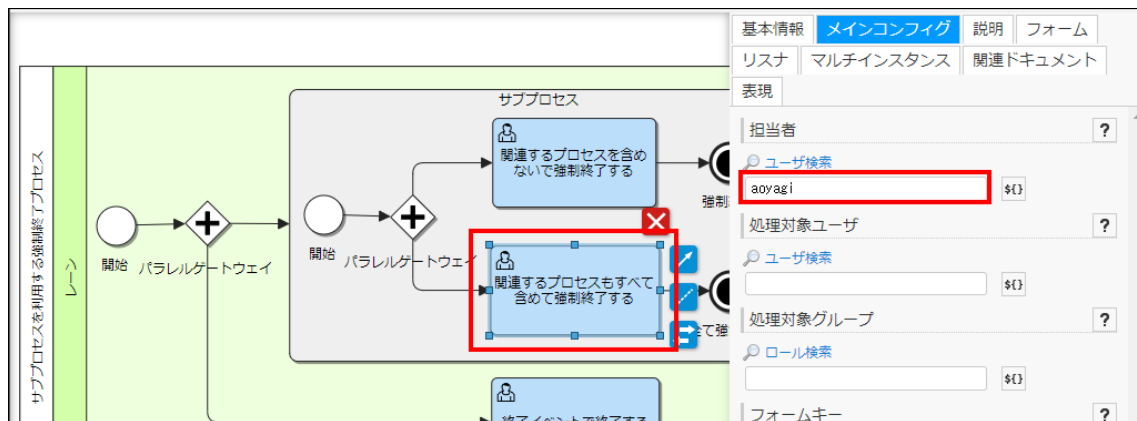
図：「サブプロセス」 - 「ユーザタスク」 - 「メインコンフィグ」 - 「担当者」

10. 「強制終了イベント」を設置します。
11. 「関連するプロセスも含めて強制終了する」の設定を確認します。
「メインコンフィグ」タブの「関連するプロセスも含めて強制終了する」はデフォルトでチェックが入っていません。
現在作成しているルートは設定不要です。



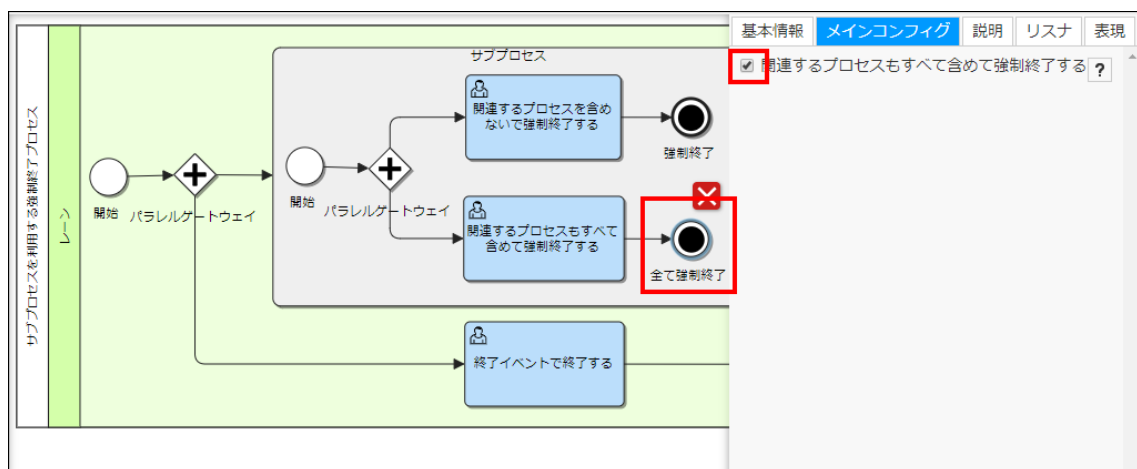
図：「コールアクティビティから呼び出される強制終了プロセス」 - 「強制終了」 - 「プロパティ」 - 「メインコンフィグ」 - 「関連するプロセスを含めないで強制終了する」

12. 関連するプロセスもすべて含めて強制終了するルートを作成します。
「ユーザタスク」を設置します。
13. 「関連するプロセスもすべて含めて強制終了する」の担当者を設定します。
「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「aoyagi」と設定します。



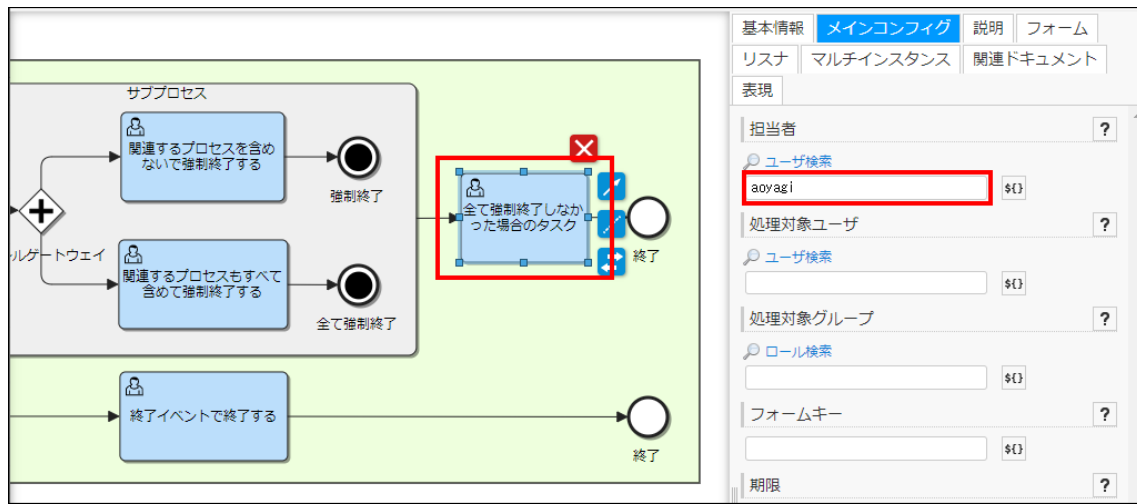
図：「サブプロセス」 - 「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

14. 「強制終了イベント」を設置します。
15. 「強制終了イベント」に到達した際に、関連するプロセスもすべて含めて強制終了させる設定を行います。
「強制終了イベント」を選択した状態で、「メインコンフィグ」タブにある「関連するプロセスもすべて含めて強制終了する」チェックボックスにチェックを入れます。



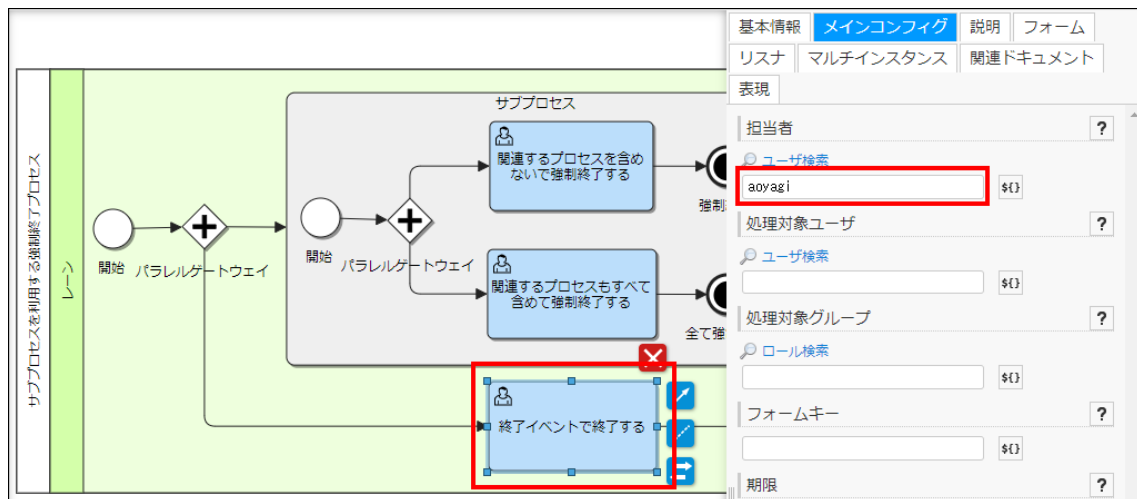
図：「サブプロセス」 - 「強制終了イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「関連するプロセスもすべて含めて強制終了する」

16. 「強制終了イベント」でサブプロセス内が終了した際に進行する「全て強制終了しなかった場合のタスク」を設置します。
「ユーザタスク」を設置します。
17. 「全て強制終了しなかった場合のタスク」の担当者を設定します。
「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「aoyagi」と設定します。



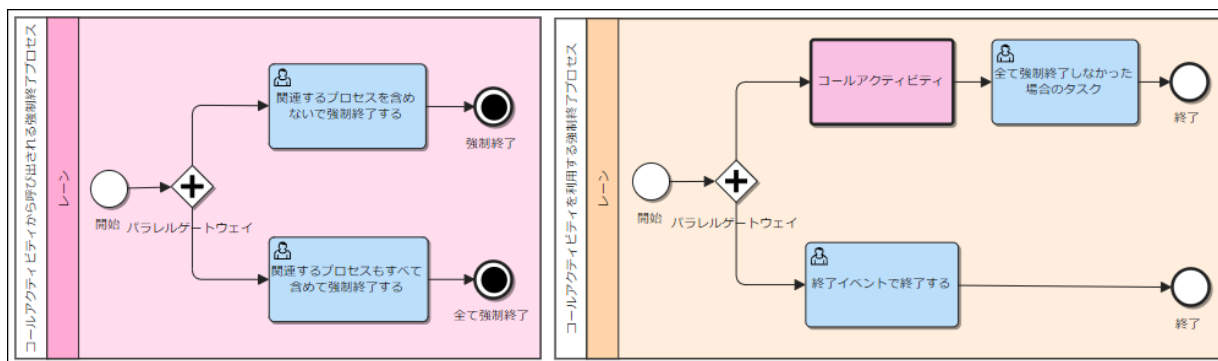
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

18. 「終了イベント」を設置します。
19. 終了イベントを使用した場合の動作を確認するルートを作成します。
「ユーザタスク」を設置します。
20. 「終了イベントで終了する」タスクの担当者を設定します。
「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「aoyagi」と設定します。



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

21. 「終了イベント」を設置します。
- コールアクティビティを利用して、強制終了するプロセス定義を作成する
- 「コールアクティビティ」を利用して、異なる範囲を終了するプロセスを作成します。
関連する別のプロセスに対しても、強制終了イベントの設定によりプロセスの終了範囲を変更できます。

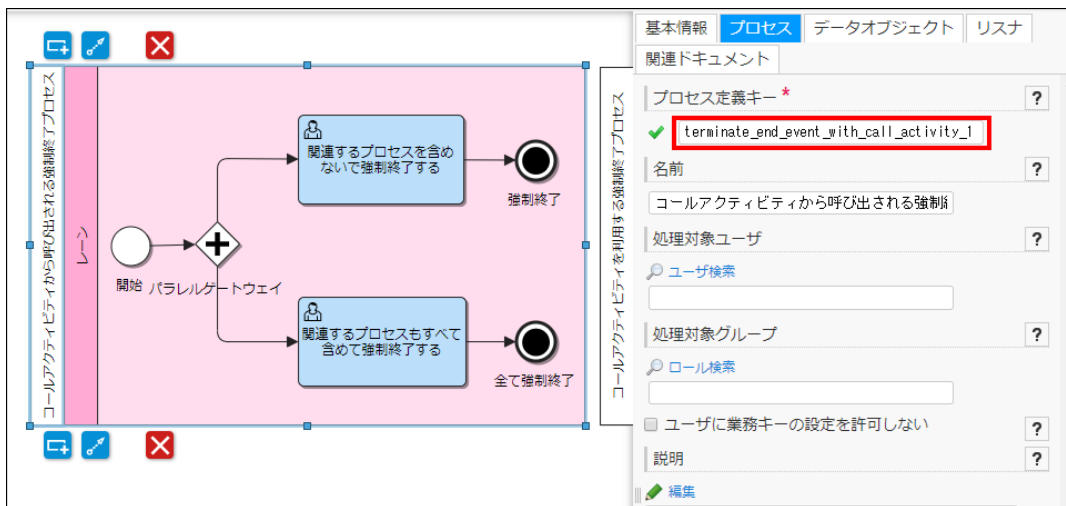


図：「コールアクティビティから呼び出される強制終了プロセス」と「コールアクティビティを利用する強制終了プロセス」

1. 「コールアクティビティから呼び出される強制終了プロセス」を作成します。
プールを設置します。
2. 「開始イベント」を設置します。

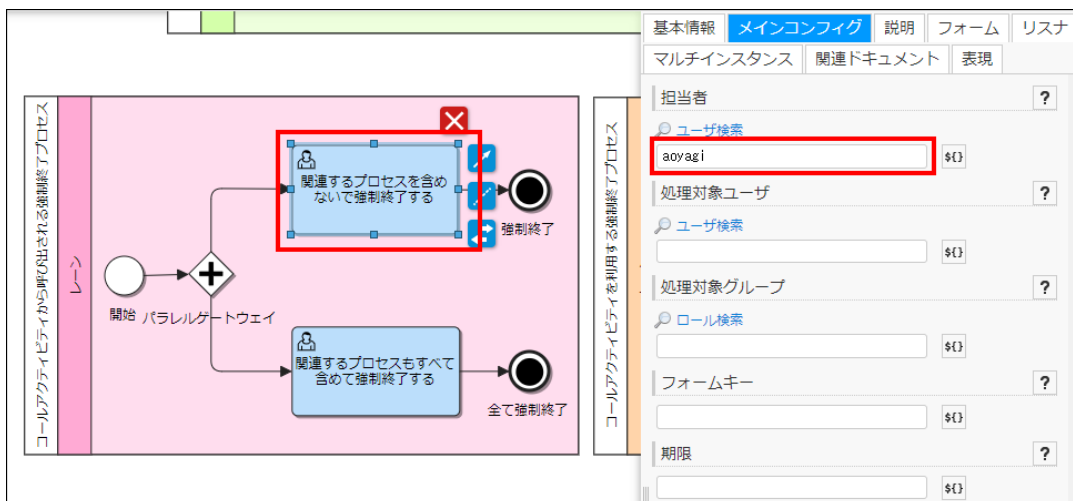
3. 「コールアクティビティ」から呼び出される際に使用するため、プロセス定義キーを設定します。

プールを選択した状態で、「プロセス」タブの「プロセス定義キー」に「`terminate_end_event_with_call_activity_1`」と設定します。



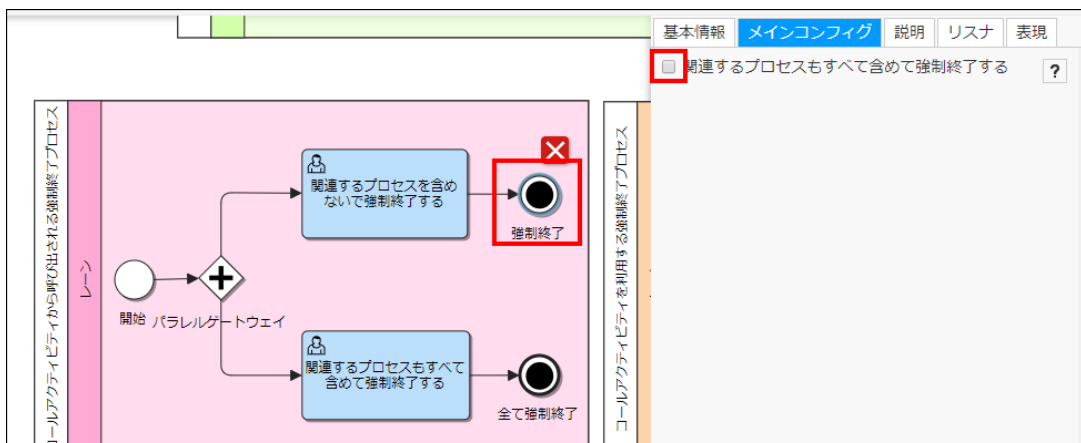
図：「プール」 - 「プロパティ」 - 「プロセス」 - 「プロセス定義キー」

4. 「パラレルゲートウェイ」を設置します。
5. 関連するプロセスを含めないで強制終了するルートを作成します。
「ユーザタスク」を設置します。
6. 「関連するプロセスを含めないで強制終了する」タスクの担当者を設定します。
「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「`aoyagi`」と設定します。



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

7. 「強制終了イベント」を設置します。
8. 「関連するプロセスもすべて含めて強制終了する」の設定を確認します。
「メインコンフィグ」タブの「関連するプロセスもすべて含めて強制終了する」はデフォルトでチェックが入っていません。
現在作成しているのは関連するプロセスを含めないで強制終了するルートのため、設定は不要です。

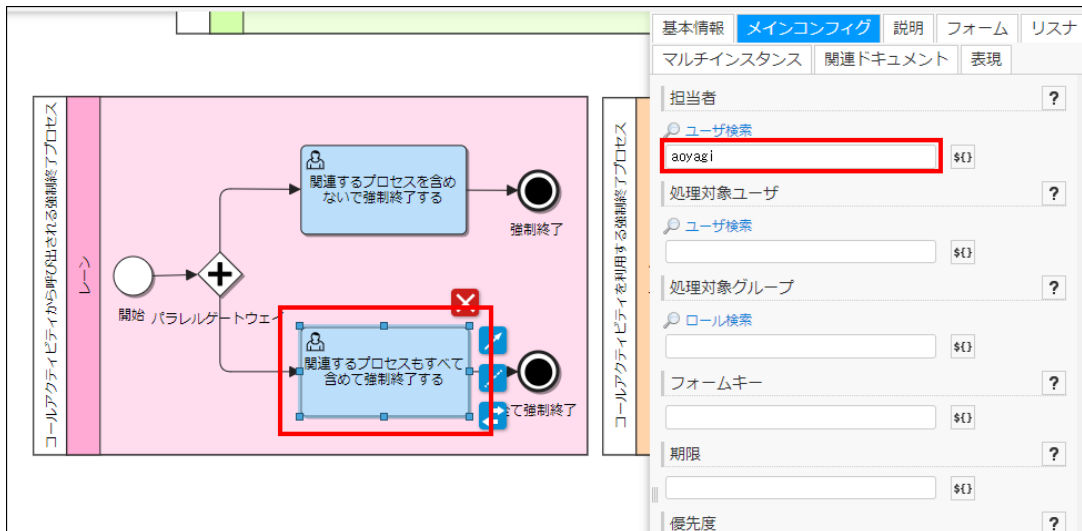


図：「強制終了イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「関連するプロセスもすべて含めて強制終了する」

9. 関連するプロセスもすべて含めて強制終了するルートを作成します。

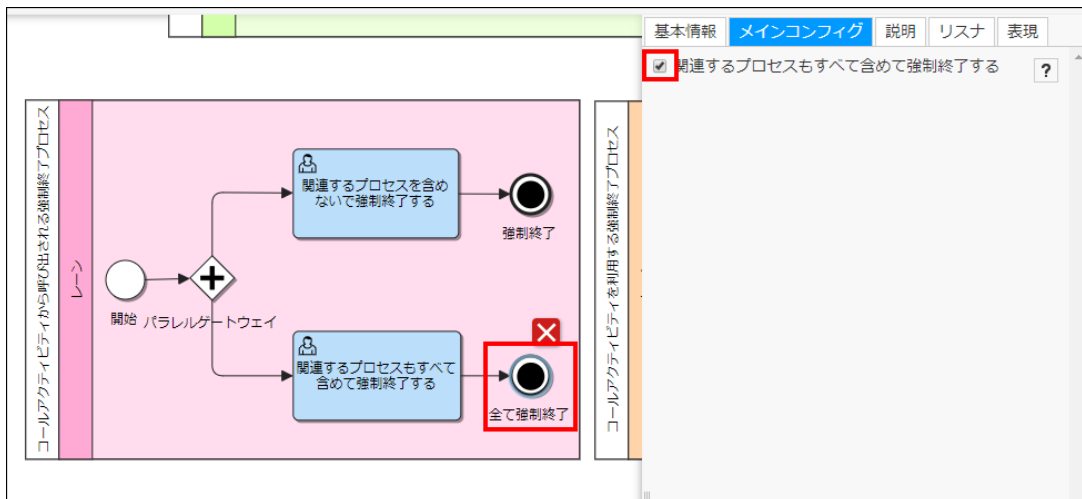
「ユーザタスク」を設置します。

10. 「関連するプロセスもすべて含めて強制終了する」タスクの担当者を設定します。
「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「aoyagi」と設定します。



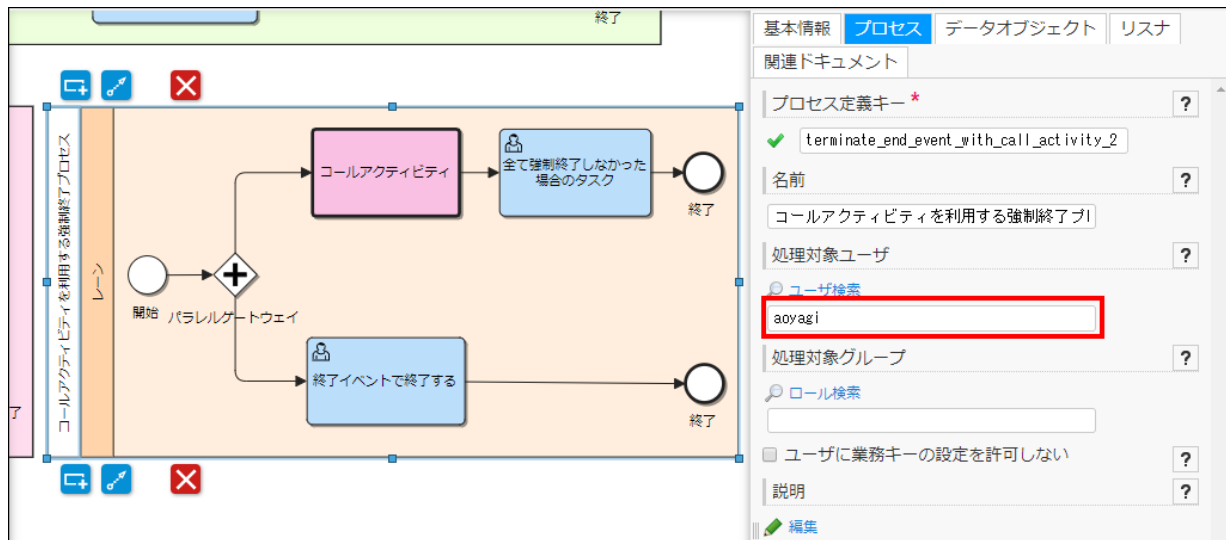
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

11. 関連するプロセスもすべて含めて強制終了するように設定します。
「強制終了イベント」を選択した状態で、「メインコンフィグ」タブにある「関連するプロセスもすべて含めて強制終了する」チェックボックスにチェックを入れます。



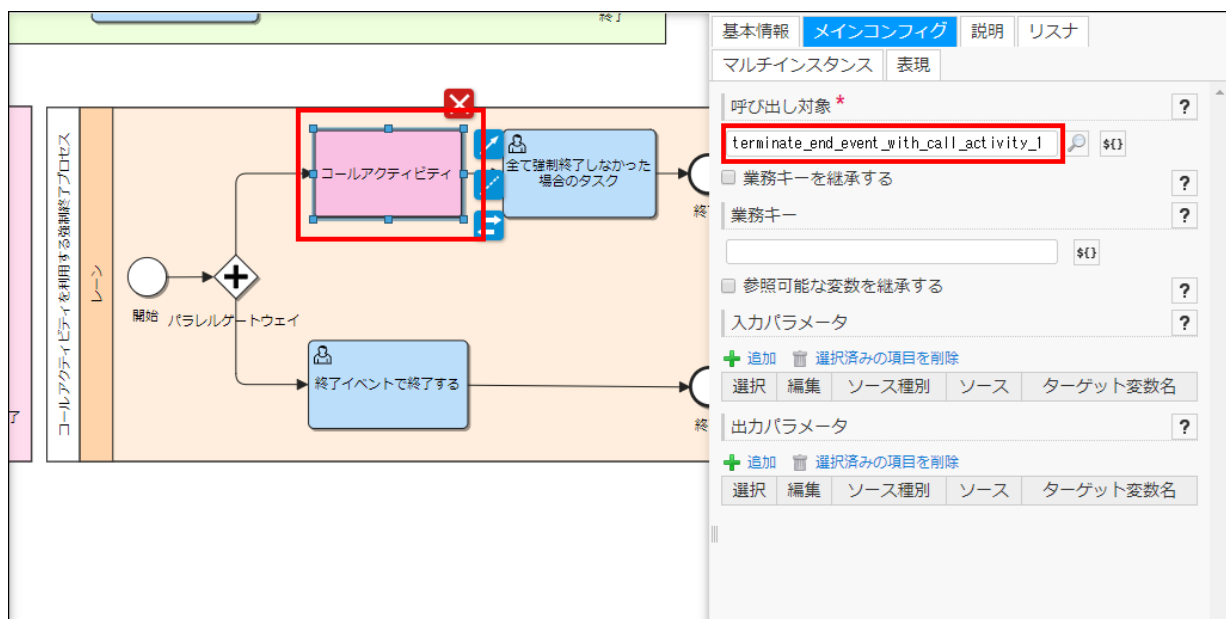
図：「強制終了イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「関連するプロセスもすべて含めて強制終了する」

12. 「コールアクティビティを利用する強制終了プロセス」を作成します。
プールを設置します。
13. 「レーン」の中に「開始イベント」を設置します。
14. 「コールアクティビティを使用する強制終了プロセス」の処理対象ユーザを設定します。
プールを選択した状態で、「プロセス」タブから「処理対象ユーザ」に「aoyagi」と設定します。



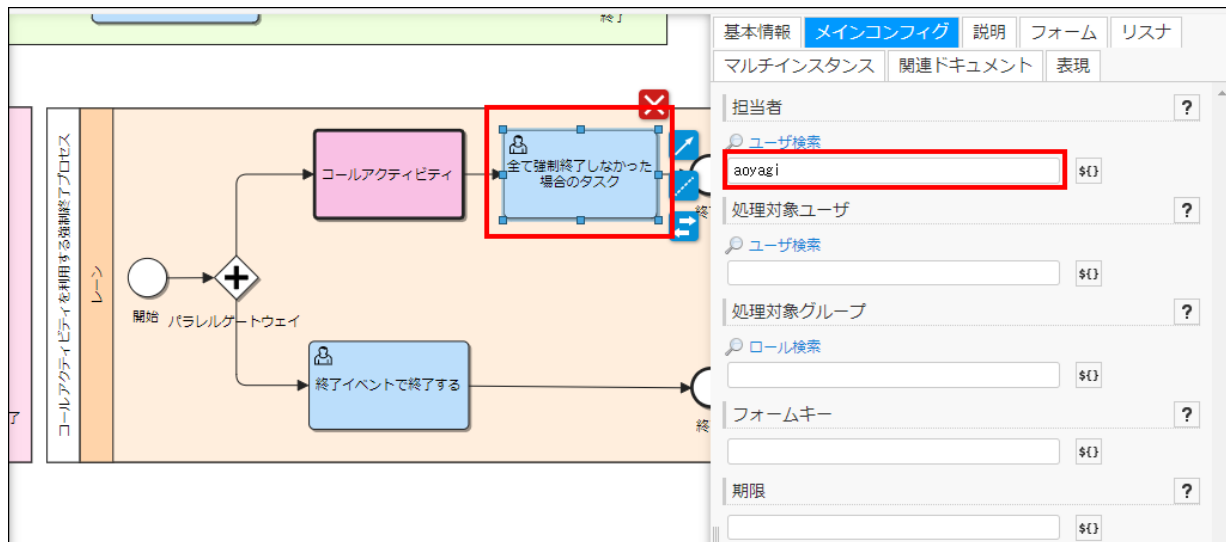
図：「プール」 - 「プロパティ」 - 「プロセス」 - 「処理対象ユーザ」

15. 「パラレルゲートウェイ」を設置します。
16. コールアクティビティを使用して、強制終了するプロセスを呼び出すルートを作成します。
「コールアクティビティ」を設置します。
17. コールアクティビティで呼び出すプロセスを設定します。
「メインコンフィグ」タブから「呼び出し対象」に、「コールアクティビティから呼び出される強制終了プロセス」のプロセス定義ID「`terminate_end_event_with_call_activity_1`」を入力します。



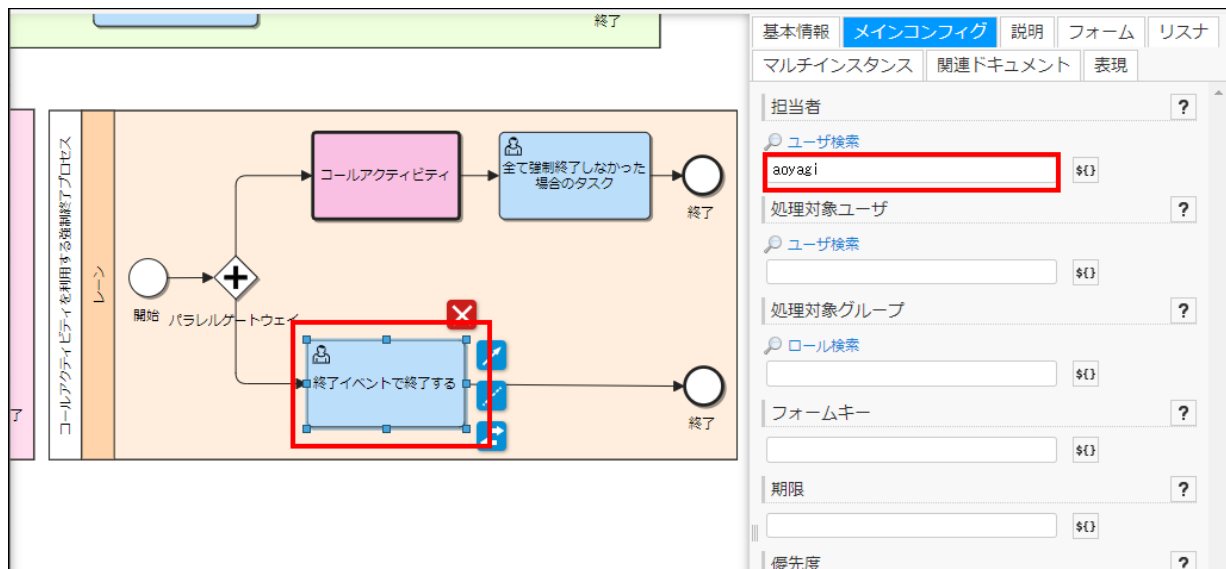
図：「コールアクティビティ」 - 「プロパティ」 - 「メインコンフィグ」 - 「呼び出し対象」

18. 「関連するプロセスもすべて含めて強制終了する」設定がされていない終了イベントへ進んだ場合のタスクを設置します。
「ユーザタスク」を設置します
19. 「全て強制終了しなかった場合のタスク」の担当者を設定します。
「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「`aoyagi`」と設定します。



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

20. 「終了イベント」を設置します。
21. 終了イベントで終了するルートを作成します。
「ユーザタスク」を設置します。
22. 「終了イベントで終了する」タスクの担当者を設定します。
「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「aoyagi」と設定します。



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

23. 「終了イベント」を設置します。

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。

「サブプロセスを使用する強制終了イベント」の動作確認

サブプロセスを使用して強制終了するプロセスの動作確認をします。

各動作確認の前に、「プロセス開始一覧」から「サブプロセスを利用する強制終了プロセス」を開始してください。



図: 「プロセス開始一覧」

終了イベントの動作確認

「終了イベント」に到達した場合のプロセスの状態を確認します。

1. タスクを確認します。
「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
2. 「終了イベントで終了する」タスクを実行します。
「個人タスク」から、「終了イベントで終了する」タスクの「」ボタンをクリックします。



図: 「タスク一覧」 - 「個人タスク」

3. 「終了イベント」に到達しても、他のタスクが終了していないことを確認します。
プロセス定義名が「サブプロセスを利用する強制終了プロセス」であるタスクの「」ボタンをクリックします。

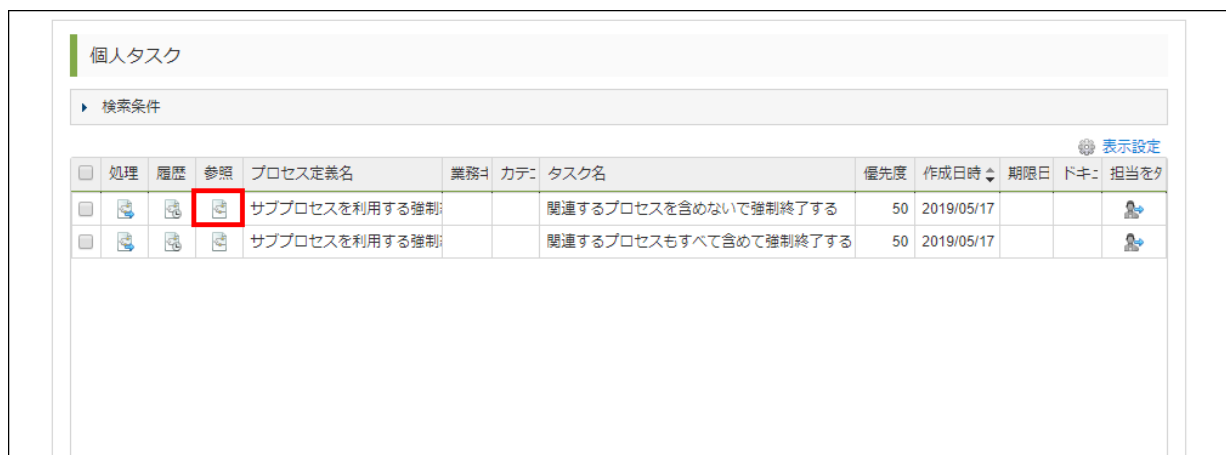


図: 「タスク一覧」 - 「個人タスク」

4. 「プロセス参照」画面、または、「拡大表示」画面から、タスクが下記の状態であることを確認します。

- ステータスが「実行中」であること
- プロセス図で、サブプロセスが終了しておらず、実行中の「▶」がついていること
- プロセス履歴で「終了イベントで終了する」タスクが完了していること
「関連するプロセスもすべて含めて強制終了する」と「関連するプロセスを含めないで強制終了する」が処理を待っていること

プロセス参照

ドキュメント 関係者一覧

プロセス定義ID	terminate_end_event_with_sub_process:1:8f713yp497md9rq	プロセス定義名	サブプロセスを利用する強制終了プロセス
プロセス定義キー	terminate_end_event_with_sub_process	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8f713z15e7mfsrq	開始ユーザ	青柳辰巳
開始日時~完了日時	2019/05/17 14:57:36 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

表示倍率: 90 %

終了 2019/05/17 16:09:31

2019/05/17 14:57 関連するプロセスもすべて含めて強制終了する
タスクの処理を待っています。
担当者: 青柳辰巳 (処理中)
開始 2019/05/17 14:57:36
経過 1時間 13分

2019/05/17 14:57 関連するプロセスを含めないで強制終了する
タスクの処理を待っています。
担当者: 青柳辰巳 (処理中)
開始 2019/05/17 14:57:36
経過 1時間 13分

図: 「タスク一覧」 - 「個人タスク」

「関連するプロセスもすべて含めて強制終了する」設定をしなかった強制終了イベントの動作確認

「関連するプロセスもすべて含めて強制終了する」設定をしなかった強制終了イベントに到達した場合のプロセスの状態を確認します。
上記の「終了イベント」で使用しなかったタスクを完了させたのち、再度新しくプロセスインスタンスを開始してください。

1. タスクを確認します。
「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
2. 「関連するプロセスを含めないで強制終了する」で終了した場合を確認します。
「個人タスク」から、「関連するプロセスを含めないで強制終了する」タスクの「▶」ボタンをクリックし、実行します。

個人タスク												
検索条件												
	処理	履歴	参照	プロセス定義名	業務	カテ	タスク名	優先度	作成日時	期限	ドキュ	担当を
☐				サブプロセスを利用する強制終了			終了イベントで終了する	50	2019/05/17			
☐				サブプロセスを利用する強制終了			関連するプロセスもすべて含めて強制終了する	50	2019/05/17			
☐				サブプロセスを利用する強制終了			関連するプロセスを含めないで強制終了する	50	2019/05/17			

図: 「タスク一覧」 - 「個人タスク」

3. プロセスインスタンスの状態を確認します。

「個人タスク」にある「サブプロセスを利用する強制終了プロセス」の「」ボタンをクリックします、

個人タスク												
検索条件												
	処理	履歴	参照	プロセス定義名	業務	カテ	タスク名	優先度	作成日時	期限	ドキュ	担当を外
☐				サブプロセスを利用する強制終了			全て強制終了しなかった場合のタスク	50	2019/05/17			
☐				サブプロセスを利用する強制終了			終了イベントで終了する	50	2019/05/17			

図: 「タスク一覧」 - 「個人タスク」

4. 「プロセス参照」画面、または、「拡大表示」画面から、タスクが下記の状態であることを確認します。

- ステータスが「実行中」であること
- プロセス図で「全て強制終了しなかった場合のタスク」と「終了イベントで終了する」タスクに「」がついていること
- プロセス履歴で「関連するプロセスもすべて含めて終了する」タスクが中止していること

プロセス参照

ドキュメント 関係者一覧

プロセス定義ID	terminate_end_event_with_sub_process:1:8f713yp497md9rq	プロセス定義名	サブプロセスを利用する強制終了プロセス
プロセス定義キー	terminate_end_event_with_sub_process	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8f71783vt7mmjrj	開始ユーザ	青柳辰巳
開始日時～完了日時	2019/05/17 16:28:38 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

2019/05/17 16:33

関連するプロセスを含めないで強制終了する
タスクが完了しました。
・ 担当者: 青柳辰巳

開始 2019/05/17 16:28:38
終了 2019/05/17 16:33:55
経過 5分

2019/05/17 16:33

関連するプロセスもすべて含めて強制終了する
以下の理由によりタスクを中止しました。
・ 関連するプロセスの強制終了イベントによりタスクが削除されました。

開始 2019/05/17 16:28:38
終了 2019/05/17 16:33:55
経過 5分

図: 「プロセス参照」

「関連するプロセスもすべて含めて強制終了する」設定をした強制終了イベントの動作確認

「関連するプロセスもすべて含めて強制終了する」に到達した場合のプロセスの状態を確認します。

上記の「関連するプロセスを含めないで強制終了する」で使用しなかったタスクを完了させたのち、再度新しくプロセスインスタンスを開始してください。

1. タスクを確認します。

「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。

2. 「関連するプロセスもすべて含めて強制終了する」を実行します。

「個人タスク」から、「関連するプロセスもすべて含めて強制終了する」タスクの「」ボタンをクリックします。

個人タスク

検索条件

表示設定

処理	履歴	参照	プロセス定義名	業務	カテ	タスク名	優先度	作成日時	期限	ドキ	担当を
			サブプロセスを利用する強制終了			終了イベントで終了する	50	2019/05/17			
			サブプロセスを利用する強制終了			関連するプロセスもすべて含めて強制終了する	50	2019/05/17			
			サブプロセスを利用する強制終了			関連するプロセスを含めないで強制終了する	50	2019/05/17			

図: 「タスク一覧」 - 「個人タスク」

3. 「関連するプロセスもすべて含めて強制終了する」タスクを処理したことで、「タスク一覧」にあったタスクが消えることを確認します。



図: 「タスク一覧」 - 「個人タスク」

4. プロセスの状態を確認します。
「サイトマップ」→「BPM」→「プロセス一覧」画面を表示します。
5. 「プロセス一覧」画面で対象のタスクを検索し、「」ボタンをクリックします。

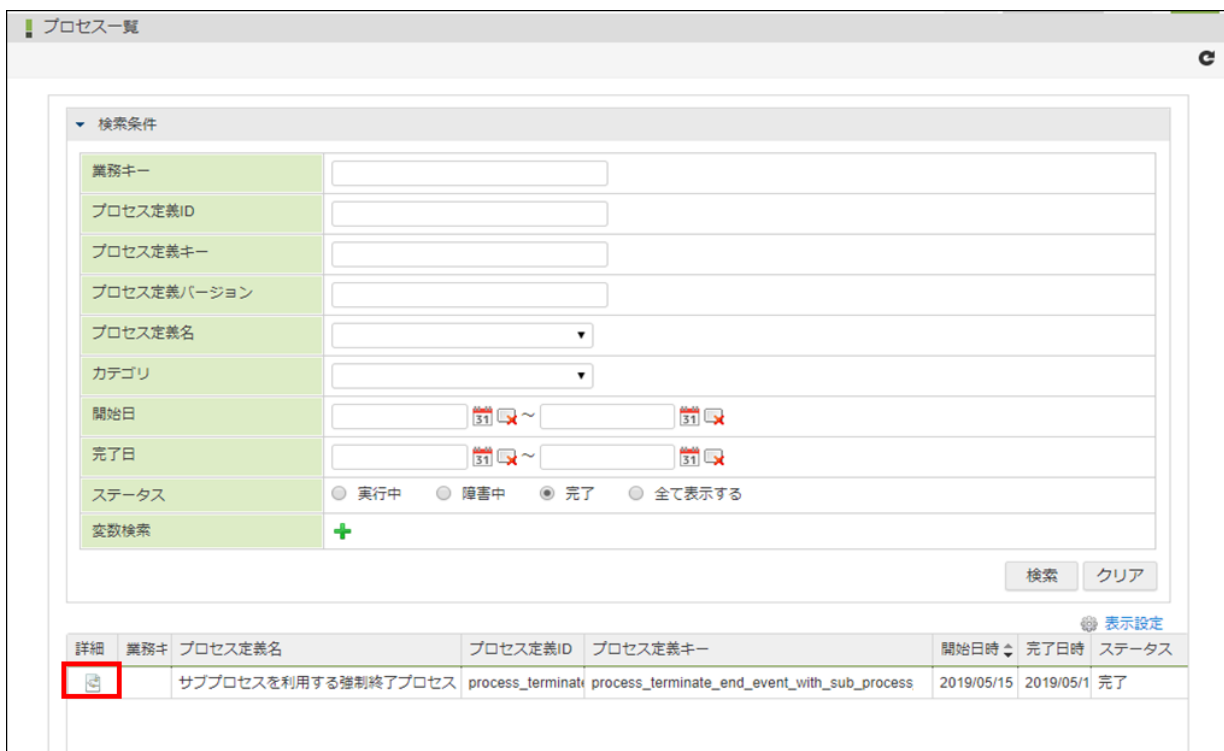


図: 「プロセス一覧」

6. 「プロセス参照」画面、または、「拡大表示」画面から、タスクが下記の状態であることを確認します。
- ステータスが「完了」であること
 - プロセス履歴で「関連するプロセスを含めないで強制終了する」タスクと「終了イベントで終了する」タスクが中止されていること

プロセス詳細

プロセス一覧 変数一覧 プロセス履歴 ドキュメント 関係者一覧

プロセス定義ID	terminate_end_event_with_sub_process:1:8f717m99z7mq6rq	プロセス定義名	サブプロセスを利用する強制終了プロセス
プロセス定義キー	terminate_end_event_with_sub_process	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8f717mxs7musrq	開始ユーザ	青柳辰巳
開始日時～完了日時	2019/05/17 16:40:11 ~ 2019/05/17 16:42:55	ステータス	完了

プロセス図とタイムラインを拡大表示

2019/05/17 16:42 全て強制終了
強制終了イベントに到達しました。プロセスが強制終了しました。
開始 2019/05/17 16:42:55
終了 2019/05/17 16:42:55

2019/05/17 16:42 終了イベントで終了する
以下の理由によりタスクを中止しました。
・ 関連するプロセスの強制終了イベントによりタスクが削除されました。
開始 2019/05/17 16:40:11
終了 2019/05/17 16:42:55
経過 2分

2019/05/17 16:42 関連するプロセスを含めないで強制終了する
以下の理由によりタスクを中止しました。

図：「プロセス詳細」

「コールアクティビティを使用する強制終了プロセス」の動作確認

「コールアクティビティを使用した強制終了イベント」の結果を確認します。

各動作確認の前に、「プロセス開始一覧」から「コールアクティビティを利用する強制終了プロセス」を開始してください。

プロセス開始一覧

タスク一覧 グループタスク一覧 個人タスク一覧 処理済一覧

検索条件

プロセス定義名

カテゴリ

検索 クリア

プロセス開始	プロセス図	プロセス定義名	カテゴリ	説明	ドキュメント
		コールアクティビティを利用する強制終了プロセス	http://www.intra-mart.jp/im_bpm		
		サブプロセスを利用する強制終了プロセス	http://www.intra-mart.jp/im_bpm		

図：「プロセス開始一覧」

終了ペントルートの動作確認

実行手順、確認箇所については「[終了イベントの動作確認](#)」と同じです。

「関連するプロセスもすべて含めて強制終了する」設定をしなかった強制終了イベントの動作確認

「関連するプロセスもすべて含めて強制終了する」設定をしなかった強制終了イベントに到達した場合のプロセスの状態を確認します。

上記の「終了イベント」で使用しなかったタスクを完了させたのち、再度新しくプロセスインスタンスを開始してください。

1. タスクを確認します。

「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。

2. 「強制終了イベント」で終了した場合を確認します。

「個人タスク」から、「関連するプロセスを含めないで終了する」の「」ボタンをクリックします。

個人タスク											
検索条件											
表示設定											
☐	処理	履歴	参照	プロセス定義名	業務キ	カテゴリ	タスク名	優先度	作成日時	期限	ドキュメント
☐				コールアクティビティから呼			関連するプロセスもすべて含めて強制終了する	50	2019/05/17		
☐				コールアクティビティから呼			関連するプロセスを含めないで強制終了する	50	2019/05/17		
☐				コールアクティビティを利用			終了イベントで終了する	50	2019/05/17		

図: 「タスク一覧」 - 「個人タスク」

3. 「関連するプロセスを含めないで終了する」タスクを処理することで、「全て強制終了しなかった場合のタスク」がきます。

プロセスの状態を確認するため、「全て強制終了しなかった場合のタスク」の「」ボタンをクリックします。

個人タスク											
検索条件											
表示設定											
☐	処理	履歴	参照	プロセス定義名	業務キ	カテゴリ	タスク名	優先度	作成日時	期限	ドキュメント
☐				コールアクティビティから呼び出さ			全て強制終了しなかった場合のタスク	50	2019/05/17		
☐				コールアクティビティを利用する強			終了イベントで終了する	50	2019/05/17		

図: 「タスク一覧」 - 「個人タスク」

4. 「プロセス参照」画面、または、「拡大表示」画面から、タスクが下記の状態であることを確認します。

- ステータスが「実行中」であること
- プロセス図で「全て強制終了しなかった場合のタスク」と「終了イベントで終了する」タスクに「」がついていること
- プロセス履歴でも「全て強制終了しなかった場合のタスク」と「終了イベントで終了する」タスクが処理を待機していること

プロセス参照

ドキュメント 関係者一覧

プロセス定義ID	terminate_end_event_with_call_activity_2:1.8f76jy8ikxvbbq	プロセス定義名	コールアクティビティを利用する強制終了プロセス
プロセス定義キー	terminate_end_event_with_call_activity_2	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8f76jyivuxvcmrq	開始ユーザ	青柳 辰巳
開始日時～完了日時	2019/05/21 10:23:57 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

表示倍率: 80 %

タイムライン:

- 2019/05/21 10:23:57: 終了イベントで終了する (タスクの処理を待っています。担当者: 青柳 辰巳 (処理中))
- 2019/05/21 10:24:57: 全て強制終了しなかった場合のタスク (タスクの処理を待っています。担当者: 青柳 辰巳 (処理中))

図: 「プロセス詳細」

「関連するプロセスもすべて含めて強制終了する」設定をした強制終了イベントの動作確認

「関連するプロセスもすべて含めて強制終了する」設定をした強制終了イベントに到達した場合のプロセスの状態を確認します。

上記の「関連するプロセスを含めないで終了する」で使用しなかったタスクを完了させたのち、再度新しくプロセスインスタンスを開始してください。

1. タスクを確認します。
「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
2. 「関連するプロセスもすべて含めて強制終了する」を実行します。
「個人タスク」から、「関連するプロセスもすべて含めて強制終了する」タスクの「」ボタンをクリックします。

個人タスク

検索条件

表示設定

処理	履歴	参照	プロセス定義名	業務キ	カテコ	タスク名	優先	作成日時	期限日	ドキコ	担当を外
			コールアクティビティから呼			関連するプロセスもすべて含めて強制終了する	50	2019/05/17			
			コールアクティビティから呼			関連するプロセスを含めないで強制終了する	50	2019/05/17			
			コールアクティビティを利用			終了イベントで終了する	50	2019/05/17			

図: 「タスク一覧」 - 「個人タスク」

3. 「関連するプロセスもすべて含めて強制終了する」タスクを処理したことで、「タスク一覧」にあったタスクが消えることを確認します。

個人タスク

検索条件

表示設定

☐	処理	履歴	参照	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当を
表示するレコードがありません												

0 ページ中 1 ページ目 10

図: 「タスク一覧」 - 「個人タスク」

4. プロセスの状態を確認します。
「サイトマップ」→「BPM」→「プロセス一覧」画面を表示します。
5. 「プロセス一覧」画面で対象のタスクを検索します。
プロセス定義キー「`terminate_end_event_with_call_activity_1`」と「`terminate_end_event_with_call_activity_2`」のステータスが「完了」になっていることを確認します。

プロセス一覧

検索条件

業務キー

プロセス定義ID

プロセス定義キー

プロセス定義バージョン

プロセス定義名

カテゴリ

開始日

完了日

ステータス

変数検索

検索

クリア

表示設定

詳細	業務	プロセス定義名	プロセス定義ID	プロセス定義キー	開始日時	完了日時	ステータス
		コールアクティビティから呼び出される強制終了	terminate_end_event	terminate_end_event_with_call_activity_1	2019/05/21	2019/05/21	完了
		コールアクティビティを利用する強制終了プロセス	terminate_end_event	terminate_end_event_with_call_activity_2	2019/05/21	2019/05/21	完了

図: 「プロセス一覧」

6. 正確な「強制終了イベント」の動きも、「プロセス詳細画面」のプロセス履歴から確認できます。

プロセス詳細

プロセス一覧 変数一覧 プロセス履歴 ドキュメント 関係者一覧

プロセス定義ID	terminate_end_event_with_call_activity_2:1:8f76lowetxvmurq	プロセス定義名	コールアクティビティを利用する強制終了プロセス
プロセス定義キー	terminate_end_event_with_call_activity_2	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8f76pp7kxvqarq	開始ユーザ	青柳辰巳
開始日時～完了日時	2019/05/21 11:13:05 ~ 2019/05/21 11:13:32	ステータス	完了

プロセス図とタイムラインを拡大表示

表示倍率: 90 %

2019/05/21 11:13

終了イベントで終了する

以下の理由によりタスクを中止しました。

- 関連するプロセスの強制終了イベントによりタスクが削除されました。

強制終了

関連するプロセスの強制終了イベントにより強制終了しました。

開始 2019/05/21 11:13:05
終了 2019/05/21 11:13:32
経過 1分未満

図: 「プロセス詳細」

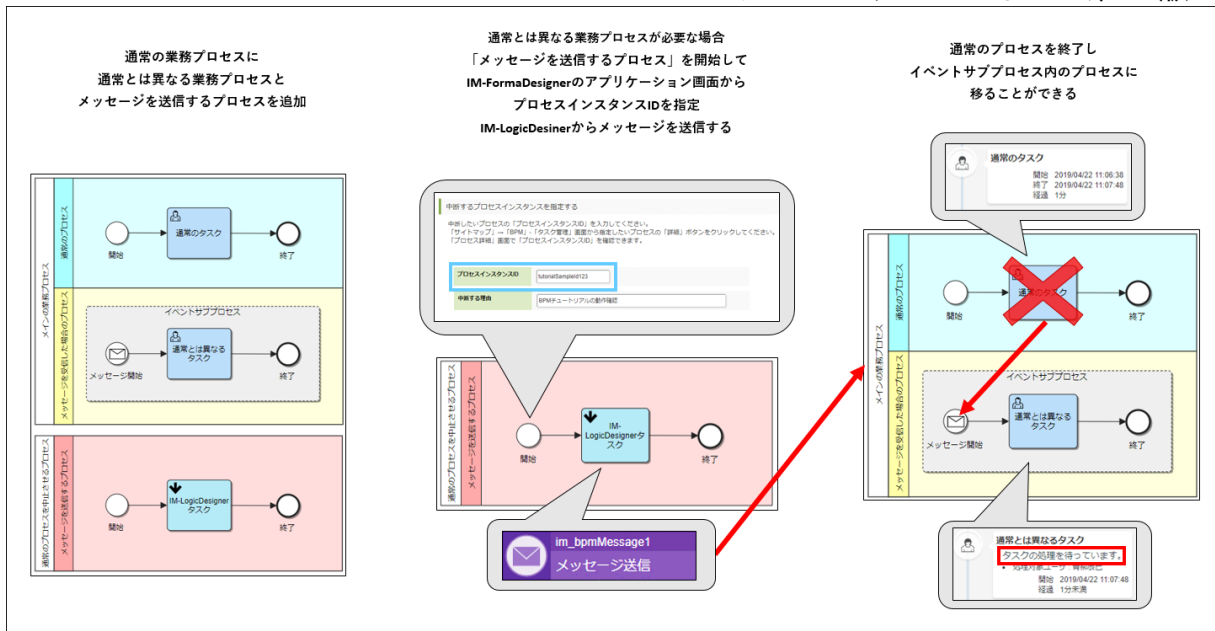
イベントサブプロセス

イベントサブプロセスを利用して実行中のプロセスを変更する

このチュートリアルでは、イベントサブプロセスを使用して実行中のプロセスを変更する方法を解説します。

イベントサブプロセスはイベントを受信した場合にのみ実行され、実行中のアクティビティは終了します。今回のチュートリアルではその機能を利用して、業務内容の変更に対応できるプロセス定義を作成します。

イベントサブプロセスの詳細については、「IM-BPM プロセスデザイナー 操作ガイド」 - 「コンテナ」 - 「イベントサブプロセス」もあわせて参照してください。



図：概要図

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Formaアプリケーションを使用します。

チュートリアルを開始する前に以下の資料をインポートしてください。

[im_forma_designer-event_sub_process_usage.zip](#)
[im_logic_designer-event_sub_process_usage.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。

[event_sub_process_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 プロセスデザイナー 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」

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。

- プロセス定義を作成する
- IM-FormaDesigner画面を確認する
- 実行結果を確認する

プロセス定義を作成する

下記の図は、メッセージを受信することによって実行中のプロセスを変更するプロセス定義です。

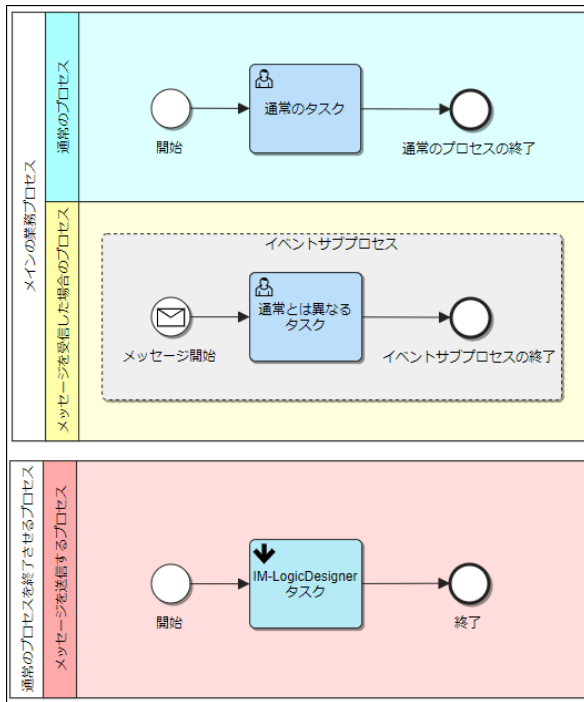
「通常のプロセス」を開始しても、「通常のプロセスを終了させるプロセス」を開始することで、「メッセージを受信した場合のプロセス」に移行できます。

業務の処理画面は、インポートした「IM-FormaDesigner」のアプリケーションを使用します。

- メインの業務プロセス
 - ユーザタスクを処理する通常のプロセス
 - メッセージを受信することで通常のプロセスを終了し、通常とは異なるユーザタスクを処理するイベントサブプロセス

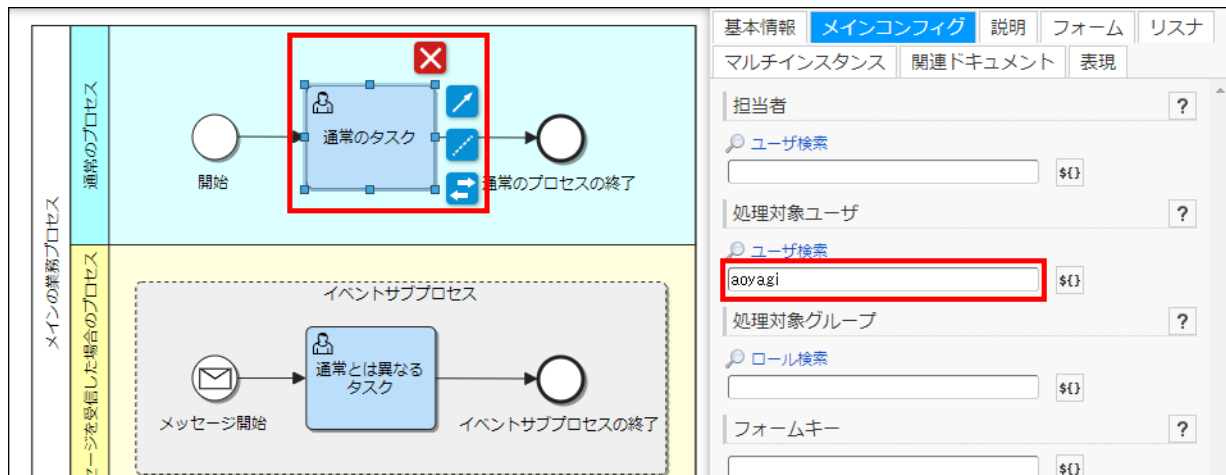
- イベントサブプロセスを起動するプロセス

- 中止したい「プロセスインスタンスID」をFormaアプリケーションから入力し、「LogicDesignerタスク」を使用してメッセージを送信するプロセス



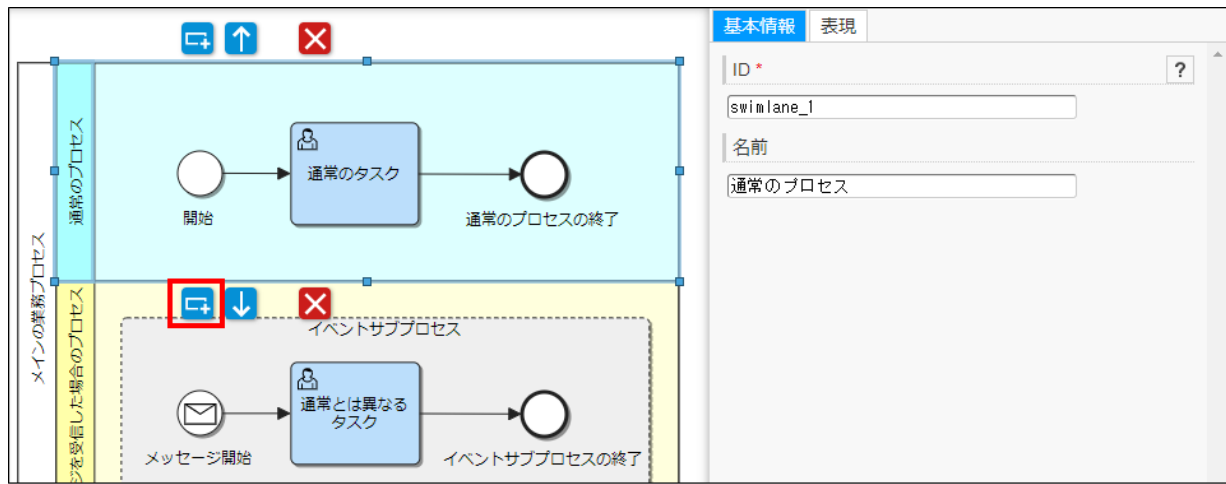
図：完成イメージ

- 「メインの業務プロセス」を作成します。
「プール」を設置します。
- 「通常のプロセス」を作成します。
「開始イベント」を設置します。
- 通常のプロセスで処理をする「ユーザタスク」を設置します。
- 「ユーザタスク」の処理対象ユーザを設定します。
「メインコンフィグ」タブから、「ユーザタスク」の処理対象ユーザを「aoyagi」にします。



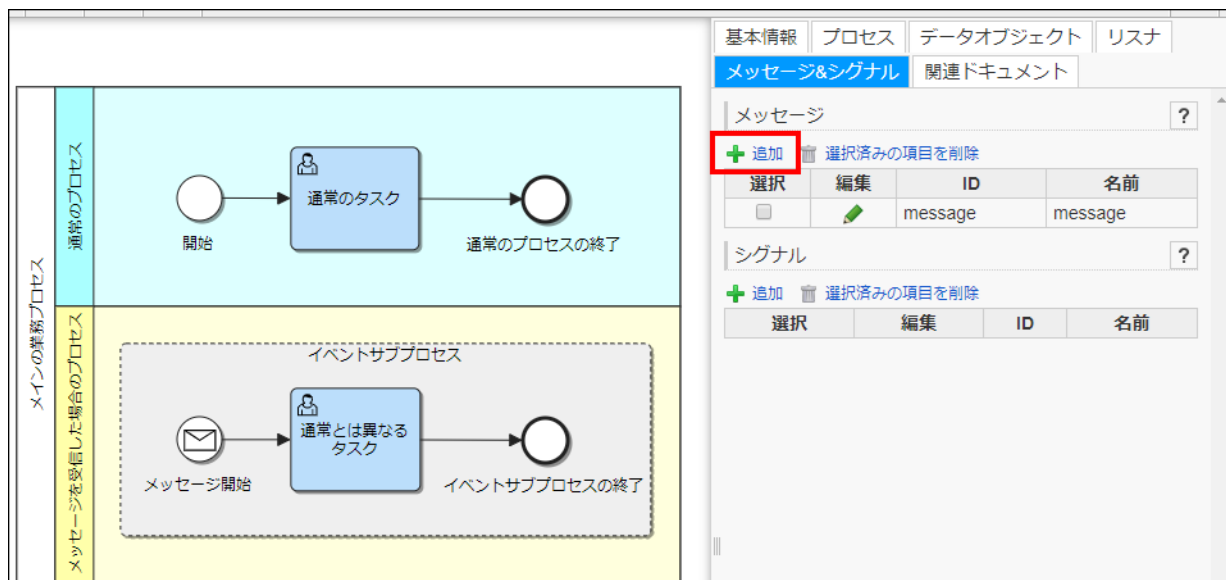
図：「メインの業務プロセス」 - 「通常のプロセス」 - 「プロパティ」 - 「メインコンフィグ」

- 終了イベントを設置します。
- 「メインの業務プロセス」に「メッセージを受信した場合のプロセス」レーンを追加します。
「通常のプロセス」レーンをクリックした状態でポップアップされる「」をクリックしてください。



図：「メインの業務プロセス」 - 「通常のプロセス」

7. 新しく作成されたレーンに「イベントサブプロセス」を設置します。
8. 「イベントサブプロセス」を実行する「メッセージ開始イベント」を設置します。
9. 「メッセージ開始イベント」に対して送信する「メッセージ」を設定します。
 エレメントが設置されていないキャンパスの空白部分をクリックし、「プロパティ」をプロセス全体に切り替えます。
 「メッセージ&シグナル」タブの「メッセージ」の「追加」リンクをクリックします。



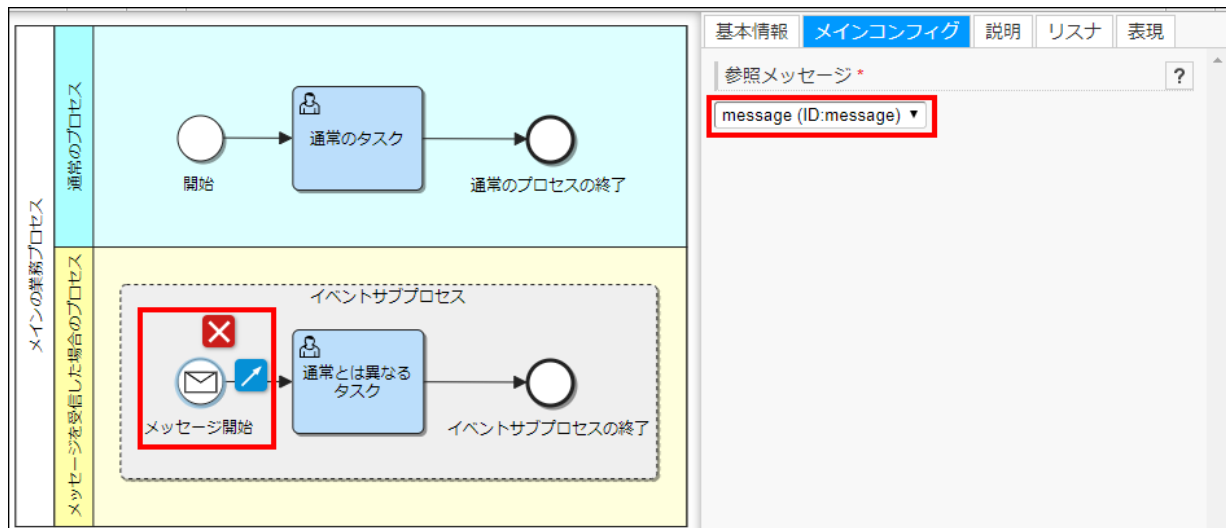
図：プロセス全体 - 「プロパティ」 - 「メッセージ&シグナル」

10. 「メッセージ」ダイアログから以下のように登録してください。

- ID : message
- 名前 : message

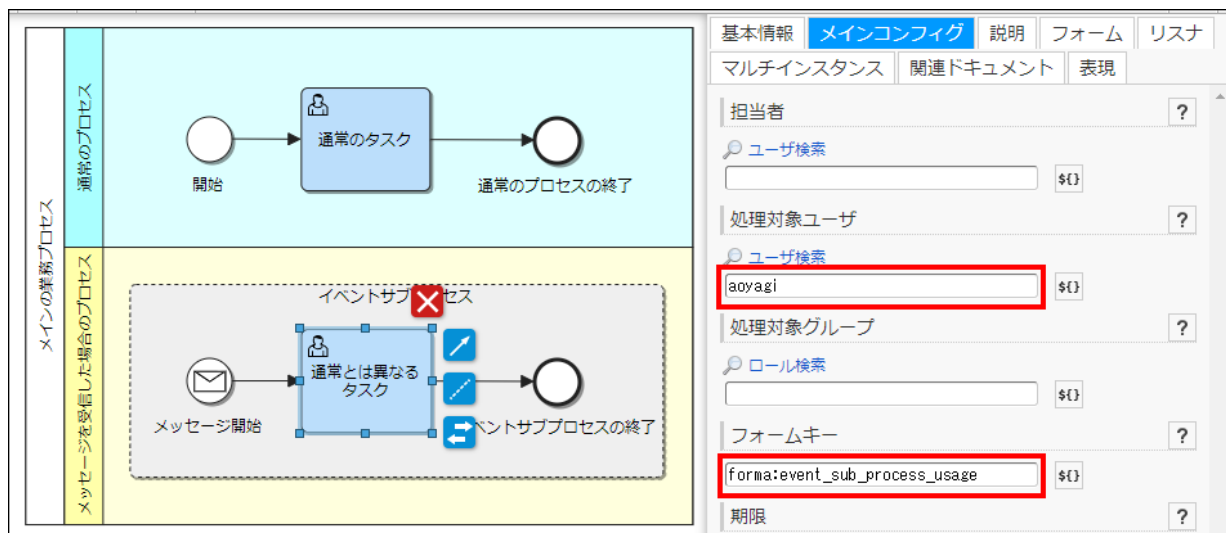
図：「メッセージ」

11. プロセス全体に登録した「メッセージ」を「メッセージ開始イベント」に紐づけます。
 「メッセージ開始イベント」をクリックし、「メインコンフィグ」タブの「参照メッセージ」から行います。
 ドロップダウンリストで「message(ID:message)」を選択します。



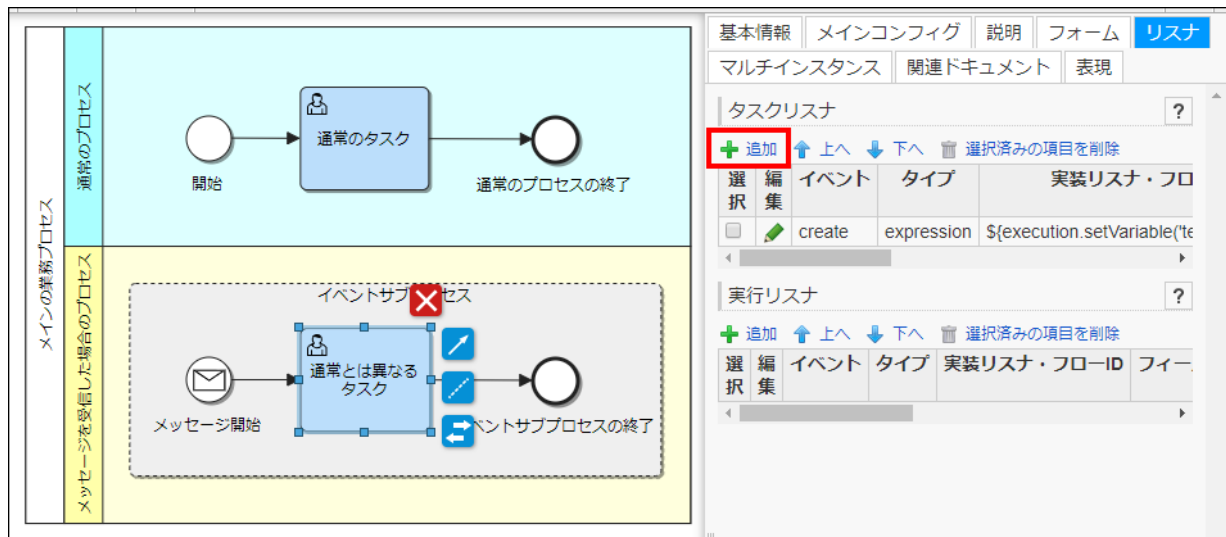
図：「メインの業務プロセス」 - 「メッセージを受信した場合のプロセス」 - 「イベントサブプロセス」 - 「メッセージ開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

12. メッセージを受信した場合に処理するユーザタスク「通常とは異なるタスク」を設置します。
13. 処理対象ユーザと、表示するFormアプリケーションを設定をします。
ユーザタスクをクリックし、「メインコンフィグ」タブから以下のように項目を設定してください。
 - 処理対象ユーザ：aoyagi
 - フォームキー：forma:event_sub_process_usage



図：「メインの業務プロセス」 - 「メッセージを受信した場合のプロセス」 - 「イベントサブプロセス」 - 「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

14. 表示するFormアプリケーションを「プロセスを中断しました」に切り替える値を設定します。
ユーザタスクをクリックし、「リスナ」タブから「タスクリスナ」の「追加」ボタンをクリックします。



図：「メインの業務プロセス」 - 「メッセージを受信した場合のプロセス」 - 「イベントサブプロセス」 - 「ユーザタスク」 - 「プロパティ」 - 「リスナ」

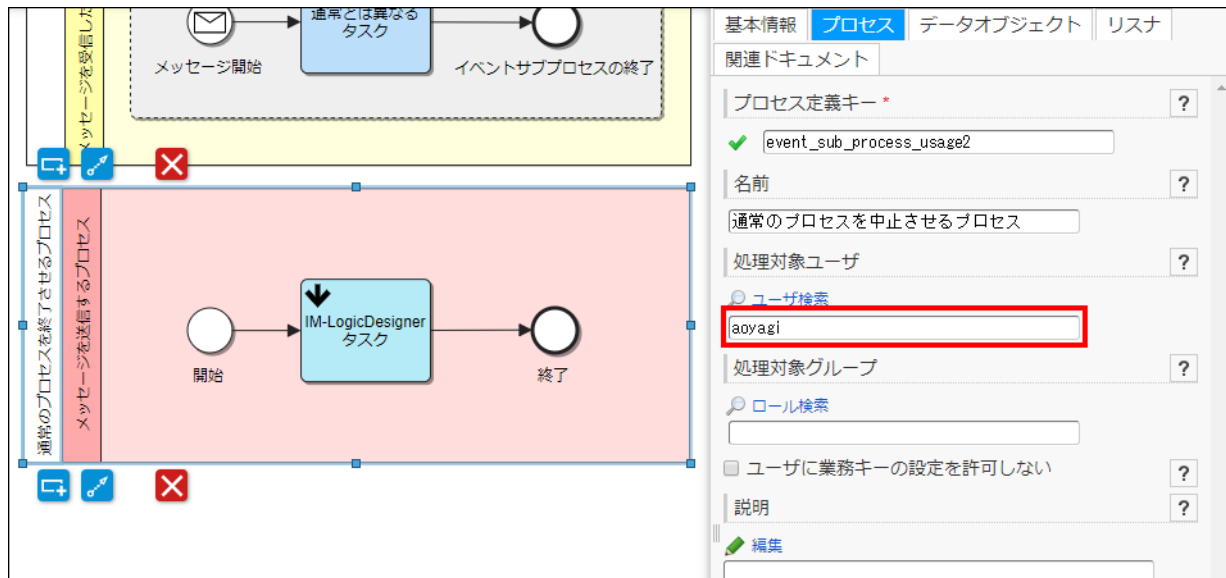
15. 「タスクリスナ」ダイアログから、以下のように項目を設定してください。

- 処理対象ユーザ：create
- タイプ：式
- 式：`${task.setVariableLocal('terminated','1')}`

The screenshot shows the 'タスクリスナ' (Task Listener) dialog box. The 'イベント' (Event) is set to 'create'. The 'タイプ' (Type) is set to '式' (Expression). The '式' (Expression) field contains the code: `${task.setVariableLocal('terminated','1')}`. The 'フィールド' (Field) section is empty. The '決定' (OK) and '取り消し' (Cancel) buttons are at the bottom right.

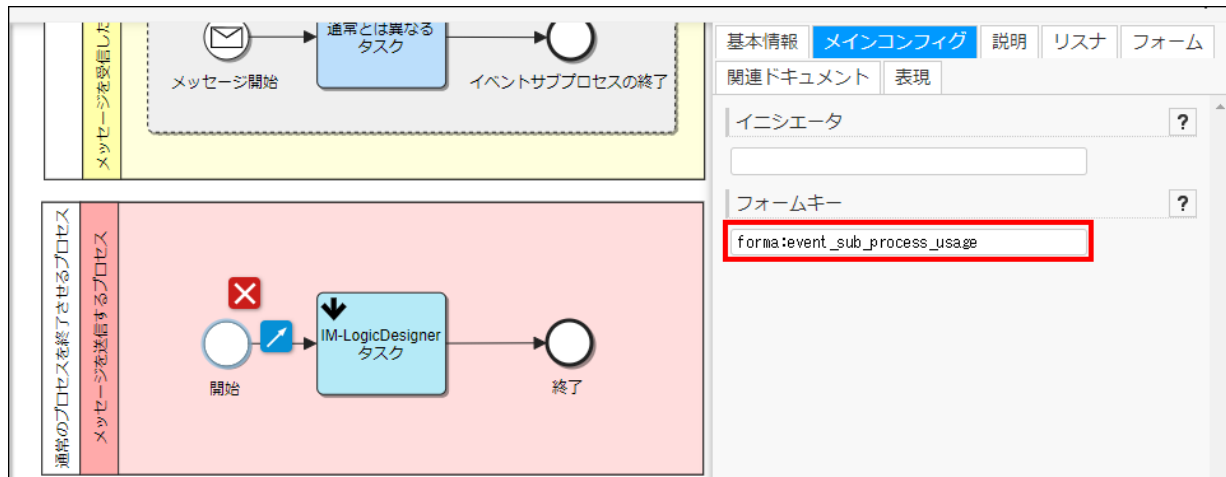
図：「タスクリスナ」

16. 「終了イベント」を設置します。
17. 「通常のプロセスを終了させるプロセス」を作成します。
「メインの業務プロセス」とは別の場所に、プールを設置します。
18. 「通常のプロセスを終了させるプロセス」の処理対象ユーザを設定します。
「プロセス」タブから、処理対象ユーザに「aoyagi」を設定します。



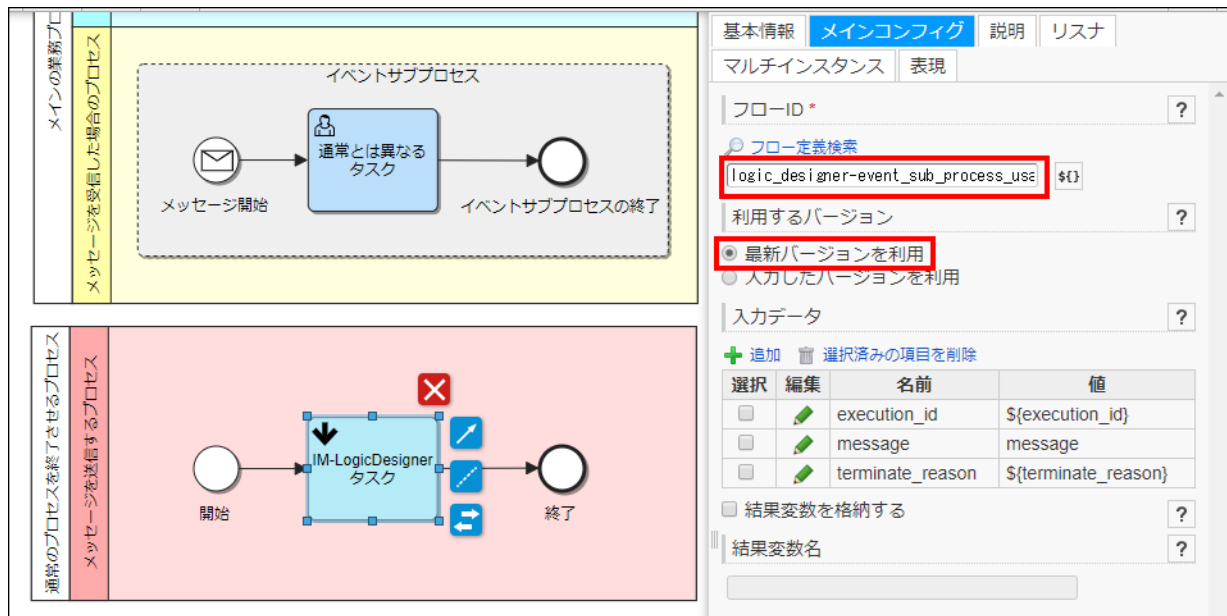
図：「通常のプロセスを終了させるプロセス」 - 「プロパティ」 - 「プロセス」

19. 「通常のプロセスを終了させるプロセス」の開始イベントを設置します。
20. 「通常のプロセスを終了させるプロセス」を開始した際に表示するFormaアプリケーションを設定します。
開始イベントをクリックし、「メインコンフィグ」タブから「フォームキー」に「form:event_sub_process_usage」と設定します。



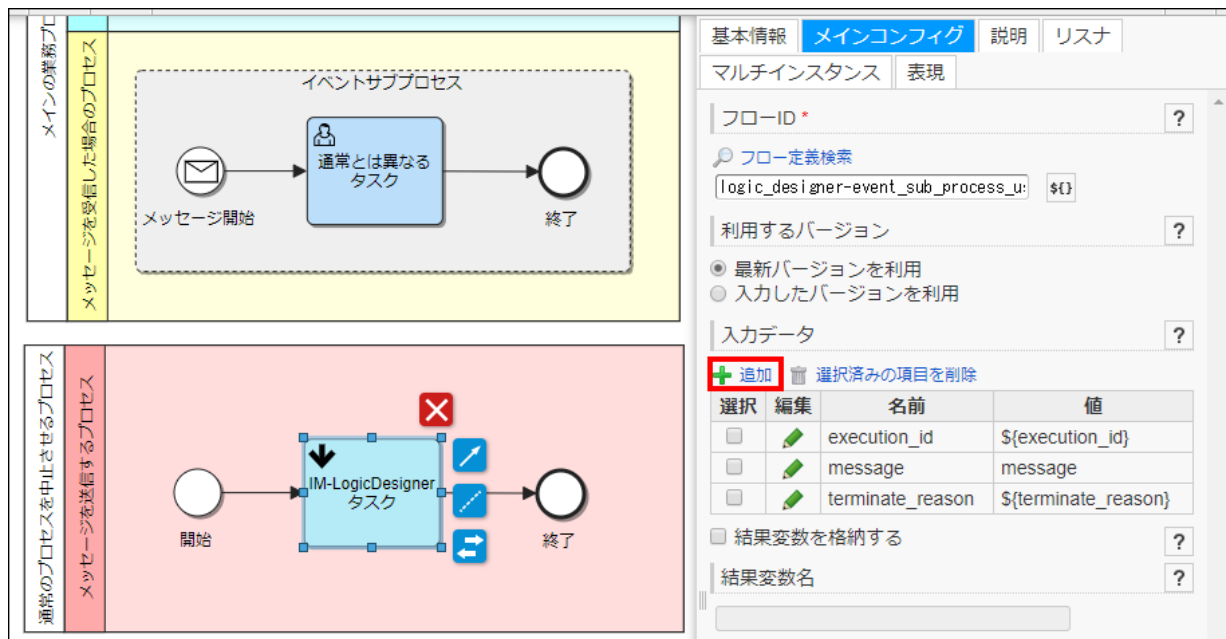
図：「通常のプロセスを終了させるプロセス」 - 「メッセージを送信するプロセス」 - 「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

21. 「メッセージ開始イベント」に対してメッセージを送信する「IM-LogicDesignerタスク」を設置します。
22. 実行するロジックフローの設定をします。
「メインコンフィグ」タブから、項目を以下のように設定します。
 - フローID：logic_designer-event_sub_process_usage
 - 利用するバージョン：最新バージョンを利用



図：「通常のプロセスを中止させるプロセス」 - 「メッセージを送信するプロセス」 - 「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

23. 「IM-LogicDesigner」に対する「入力データ」を設定します。
入力データの「追加」リンクをクリックします。



図：「通常のプロセスを中止させるプロセス」 - 「メッセージを送信するプロセス」 - 「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

24. 「入力データ」ダイアログから、3つのデータを登録します
以下のように項目を設定してください。

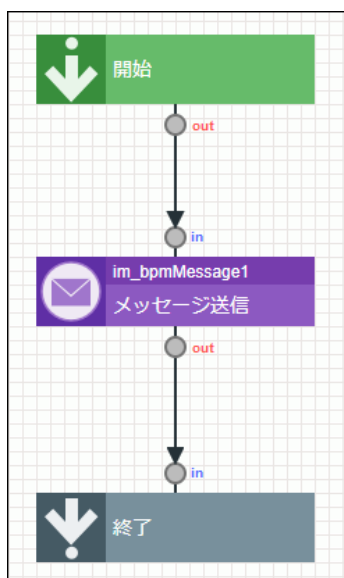
- execution_id
名前: execution_id
値: \${execution_id}
- message
名前: message
値: message
- terminate_reason
名前: terminate_reason
値: \${terminate_reason}

図：「入力データ」

25. 「終了イベント」を設置します。

IM-FormaDesigner画面を確認する

チュートリアル開始前にインポートした「IM-LogicDesigner」のロジックフローを確認します。
上記BPMの「LogicDesignerタスク」が実行されることで、メッセージを送信します。



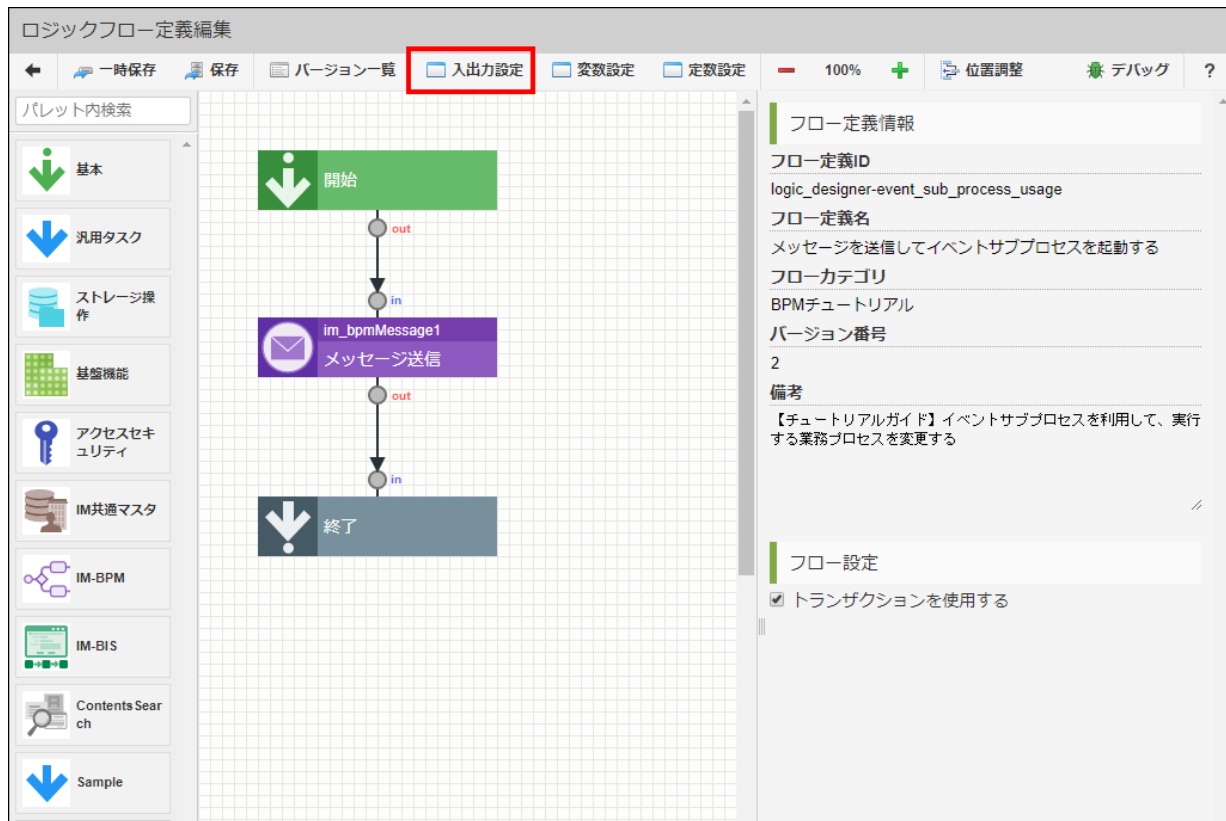
完成イメージ（IM-LogicDesigner）

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「logic_designer-event_sub_process_usage」の「」をクリックします。

編集	フロー定義ID	フロー定義名	フローカテゴリ	呼出元
	logic_designer-event_sub_process_usage	メッセージを送信してイベントサブプロセスを起動する	BPMチュートリアル	

図：「ロジックフロー定義一覧」

3. プロセス定義から渡ってくる入力データに対する設定を確認します。
メニューバーの「入出力設定」をクリックし、「入出力設定」ダイアログを表示します。



図：「ロジックフロー定義編集」

4. 「入出力設定」ダイアログで、「入力」に以下のように設定されていることを確認します。

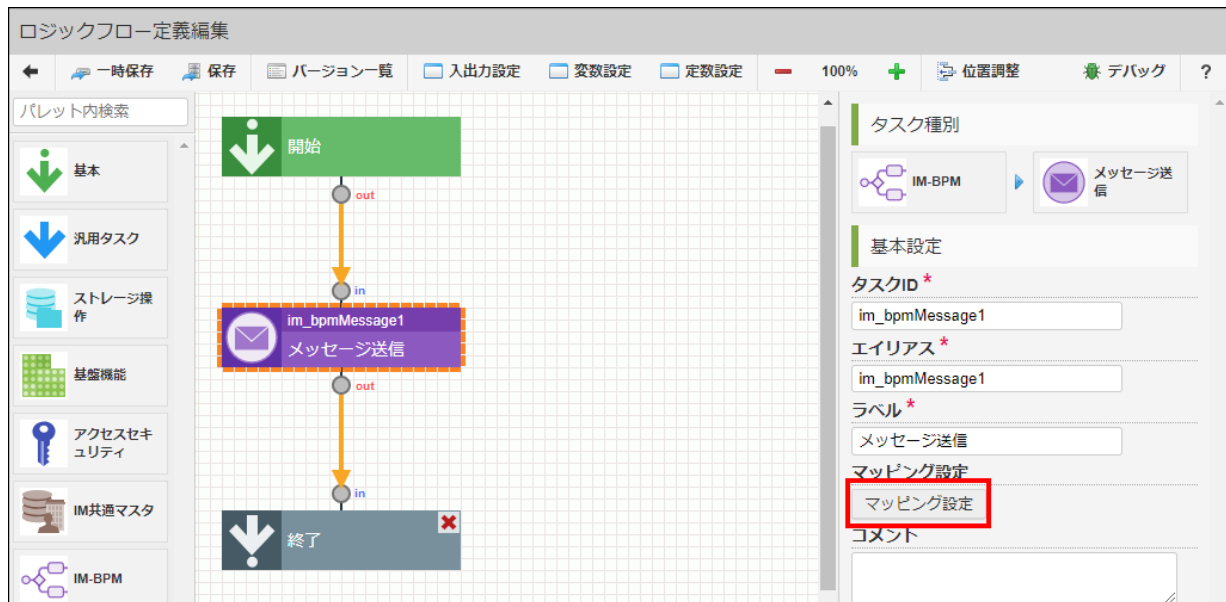
- execution_id<string>
- message<string>
- terminate_reason<string>



図：「入出力設定」

5. 「メッセージ送信」タスクのマッピングを確認します。

「メッセージ送信」タスクを選択し、基本設定の「マッピング設定」から「マッピング設定」ダイアログを表示します。



図：「ロジックフロー定義編集」 - 「メッセージ送信」

6. 「マッピング設定」ダイアログで、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。

入力 (始点)	出力 (終点)
入力<Object> - execution_id<string>	im_bpmMessage1<object> - executionId<string>
入力<Object> - message<string>	im_bpmMessage1<object> - message<string>
入力<Object> - terminate_reason<string>	im_bpmMessage1<object> - variables<map> - terminated_reason<any>



図：「マッピング設定」

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. プロセス開始一覧から、「メイン業務のプロセス」の「」アイコンをクリックします。



図：「プロセス開始一覧」

3. 実行したプロセスの状態と、プロセスインスタンスIDを確認します。
画面左上の「タスク一覧」をクリックしてください。



図：「プロセス開始一覧」

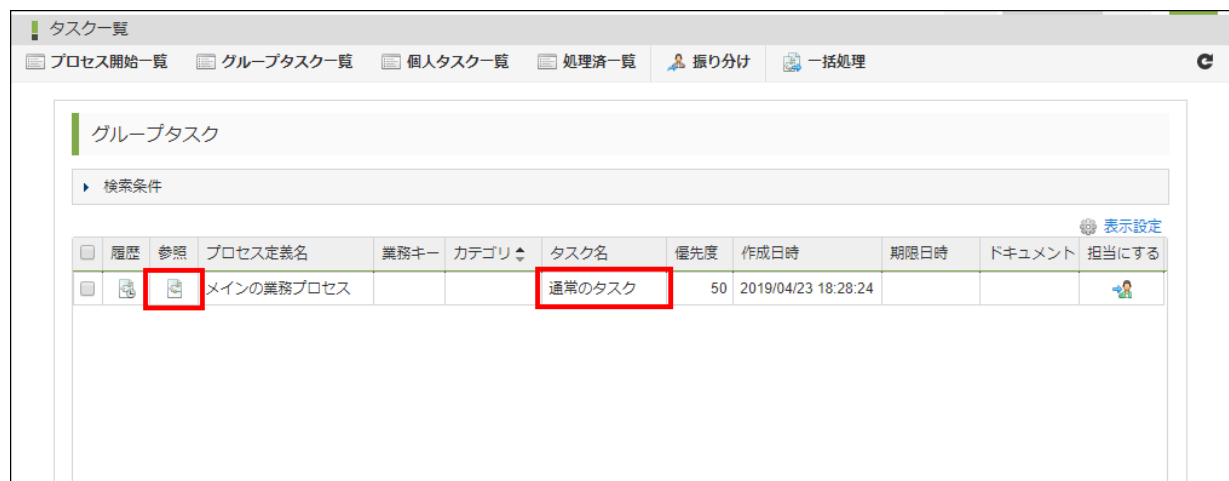
**注意**

「プロセス参照」画面は、「2018 Winter(Urara)」以降のバージョンで参照できます。

それ以前のバージョンの場合、「IM-BPM管理者」のロールをもつユーザでログインし、「サイトマップ」→「BPM」→「タスク管理」画面から「プロセス詳細」を表示することで確認できます。

4. 「タスク一覧」画面から、下記の条件を満たすタスクの「」ボタンをクリックし、「プロセス参照」画面を表示します。

- プロセス定義名：メインの業務プロセス
- タスク名：通常のタスク



図：「タスク一覧」

5. 「プロセス参照」画面から、実行中のタスクが「通常のタスク」であることを確認し、プロセスインスタンスIDを控えます。

プロセス参照

ドキュメント 関係者一覧

プロセス定義ID	event_sub_process_usage1:1-8f630xe6tmarorq	プロセス定義名	メインの業務プロセス
プロセス定義キー	event_sub_process_usage1	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8f630xo62masjrq	開始ユーザ	青柳辰巳
開始日時～完了日時	2019/04/23 18:28:24 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

表示倍率: 100 %

図：「プロセス参照」

6. 実行中のプロセスインスタンスを「通常のタスク」から、「通常とは異なるタスク」へ進行させます。

「プロセス開始一覧」画面へ戻り、「通常のプロセスを中止させるプロセス」の「」ボタンをクリックします。

プロセス開始一覧

タスク一覧 グループタスク一覧 個人タスク一覧 処理済一覧

検索条件

プロセス定義名

カテゴリ

検索 クリア

プロセス開始	プロセス図	プロセス定義名	カテゴリ	説明	ドキュメント
		メインの業務プロセス	http://www.intra-mart.jp/im_bpm		
		通常のプロセスを中止させるプロセス	http://www.intra-mart.jp/im_bpm		

図：「プロセス開始一覧」

7. Fromaアプリケーション「中断するプロセスインスタンスを指定する」が表示されます。
先ほど控えたプロセスインスタンスIDと任意の理由を記入し、「中断」ボタンをクリックします。

中断するプロセスインスタンスを指定する

中断したいプロセスの「プロセスインスタンスID」を入力してください。
「サイトマップ」→「BPM」-「タスク管理」画面から指定したいプロセスの「詳細」ボタンをクリックしてください。
「プロセス詳細」画面で「プロセスインスタンスID」を確認できます。

プロセスインスタンスID

8f630x062masjrq

中断する理由

イベントサブプロセスの動作確認

中断

図：「中断するプロセスインスタンスを指定する」(IM-FormaDesigner)

8. タスクの進行状態を確認します。

「タスク一覧」画面から、下記の状態になっているタスクの「」ボタンをクリックします。

- プロセス定義名：メインの業務プロセス
- タスク名：通常とは異なるタスク

タスク一覧

プロセス開始一覧 グループタスク一覧 個人タスク一覧 処理済一覧 振り分け 一括処理

グループタスク

検索条件

履歴	参照	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当にする
		メインの業務プロセス			通常とは異なるタスク	50	2019/04/23 18:34:32			

図：「タスク一覧」

9. プロセス図で、イベントサブプロセスの「通常とは異なるタスク」に進行していることを確認します。
また、Formaアプリケーションで指定したプロセスインスタンスと同一のIDであることを確認します。

プロセス参照

ドキュメント 関係者一覧

プロセス定義ID	event_sub_process_usage1:1:8f630xe6tmarorq	プロセス定義名	メインの業務プロセス
プロセス定義キー	event_sub_process_usage1	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8f630xe62masjrq	開始ユーザ	青柳辰巳
開始日時～完了日時	2019/04/23 18:28:24 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

図：「プロセス参照」

10. 「メインの業務プロセス」を完了させます。

「タスク一覧」画面の「グルーptask」に振り分けられているタスクを、「」ボタンをクリックします。

タスク一覧

プロセス開始一覧 グループタスク一覧 個人タスク一覧 処理済一覧 振り分け 一括処理

グループタスク

検索条件

履歴	参照	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当にする
		メインの業務プロセス			通常とは異なるタスク	50	2019/04/23 18:34:32			

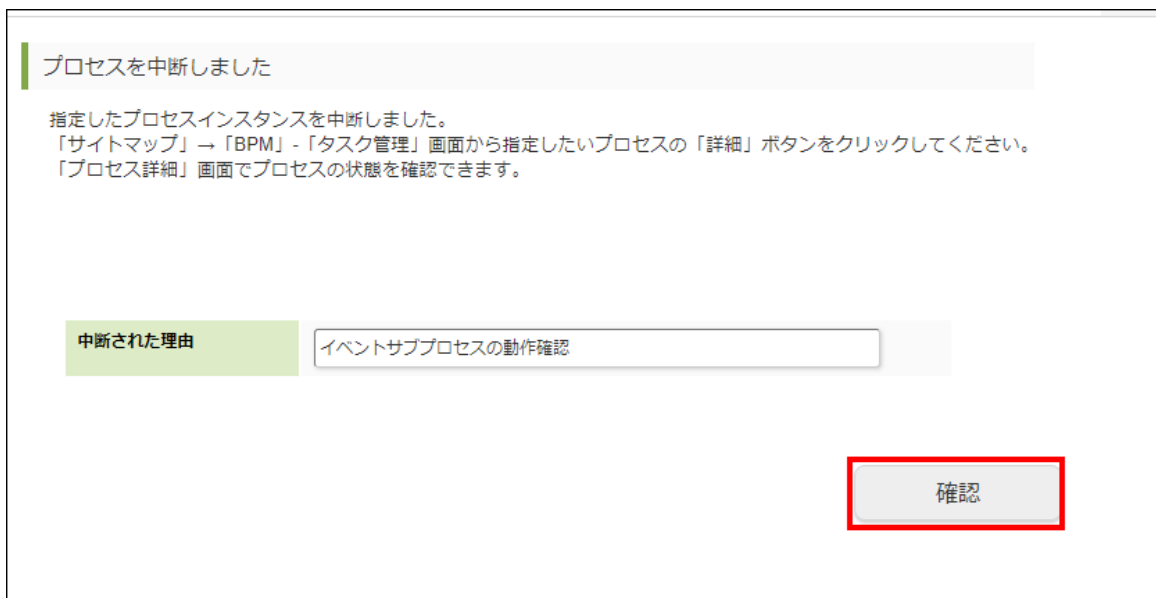
図：「タスク一覧」

11. 「個人タスク」に振り分けられたことを確認し、「」ボタンをクリックします。



図：「プロセス詳細」

12. Fromaアプリケーション「プロセスを中断しました」が表示されるので、「確認ボタン」をクリックすることでタスクが完了します。



図：「プロセスを中断しました」画面（IM-FormaDesigner）

13. 「プロセス一覧」画面から、検索条件の「ステータス」を「すべて表示する」に指定し、対象のプロセスを検索します。
対象の「メインの業務プロセス」の「」ボタンをクリックし、「プロセス詳細」画面を開きます。

プロセス一覧

検索条件

業務キー	
プロセス定義ID	
プロセス定義キー	
プロセス定義バージョン	
プロセス定義名	
カテゴリ	
開始日	31 ~ 31
完了日	31 ~ 31
ステータス	☐ 実行中 ☐ 障害中 ☐ 完了 ☒ 全て表示する
変数検索	+

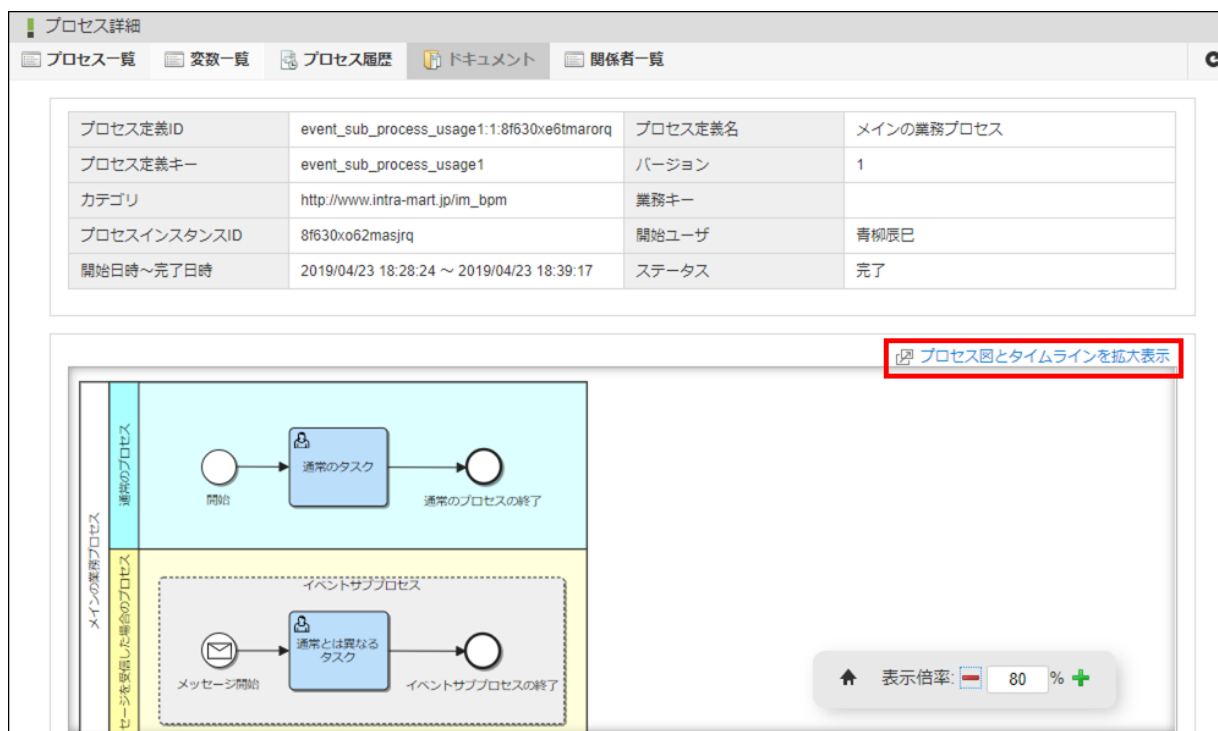
検索 クリア

表示設定

詳細	業務キー	プロセス定義名	プロセス定義ID	プロセス定義キー	開始日時	完了日時	ステータス
		メインの業務プロセス	event_sub_proce	event_sub_process_usage1	2019/04/23 18:28:24	2019/04/23 18:39:17	完了
	event_sut	通常のプロセスを中止させるプロセス	event_sub_proce	event_sub_process_usage2	2019/04/23 18:34:32	2019/04/23 18:34:32	完了

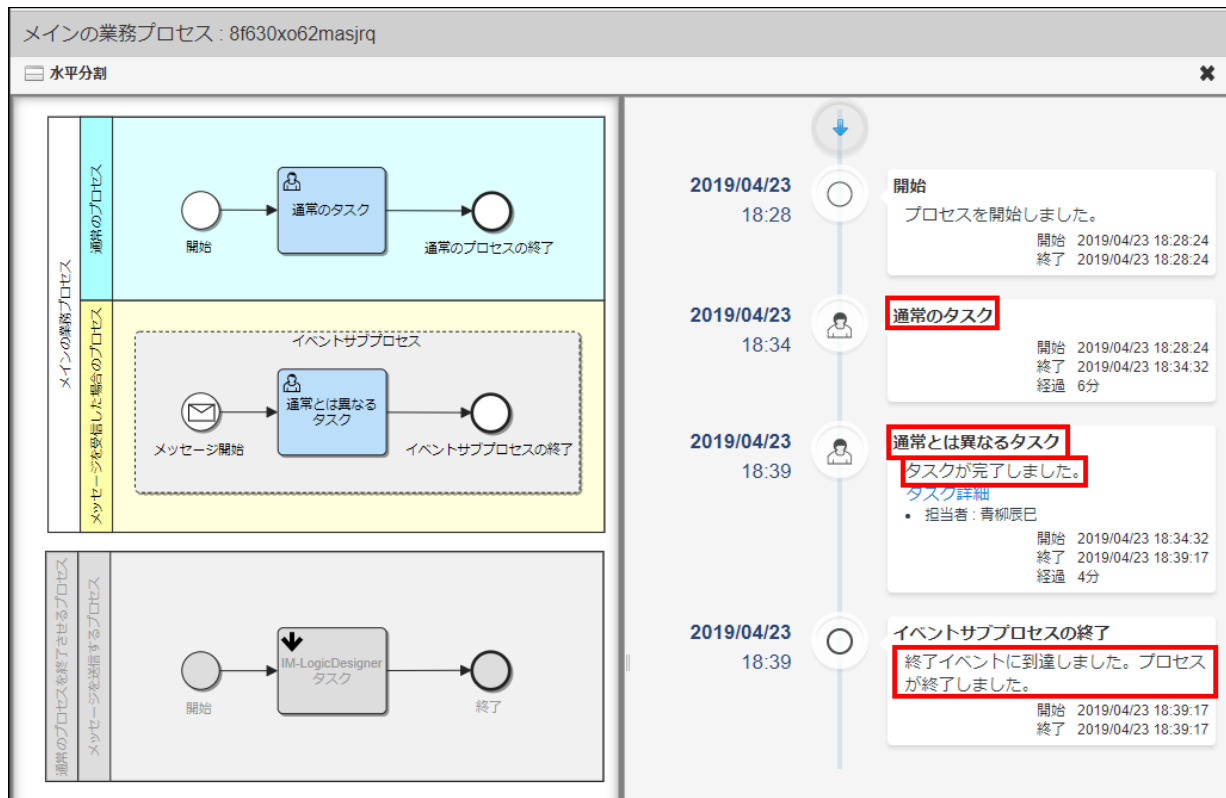
図：「プロセス詳細」

14. 「プロセス詳細」画面から、「プロセス図とタイムラインを拡大表示」をクリックしてください。



図：「プロセス詳細」画面

15. 「プロセス図とタイムライン」が拡大表示されたポップアップが表示されます。
「通常のタスク」へ進んでいたプロセスが、「通常とは異なるタスク」へ移行して終了していることを確認します。



図：「メイン業務のプロセス」

サービスタスク

e Builderで作成したJAVAクラスをサービスタスクで使用する

このチュートリアルでは、自分で作成したJAVAクラスを「サービスタスク」に設定する方法を解説します。

「intra-mart e Builder for Accel Platform」は、intra-mart Accel Platform上で動く業務アプリケーションの開発を支援するためのツールです。

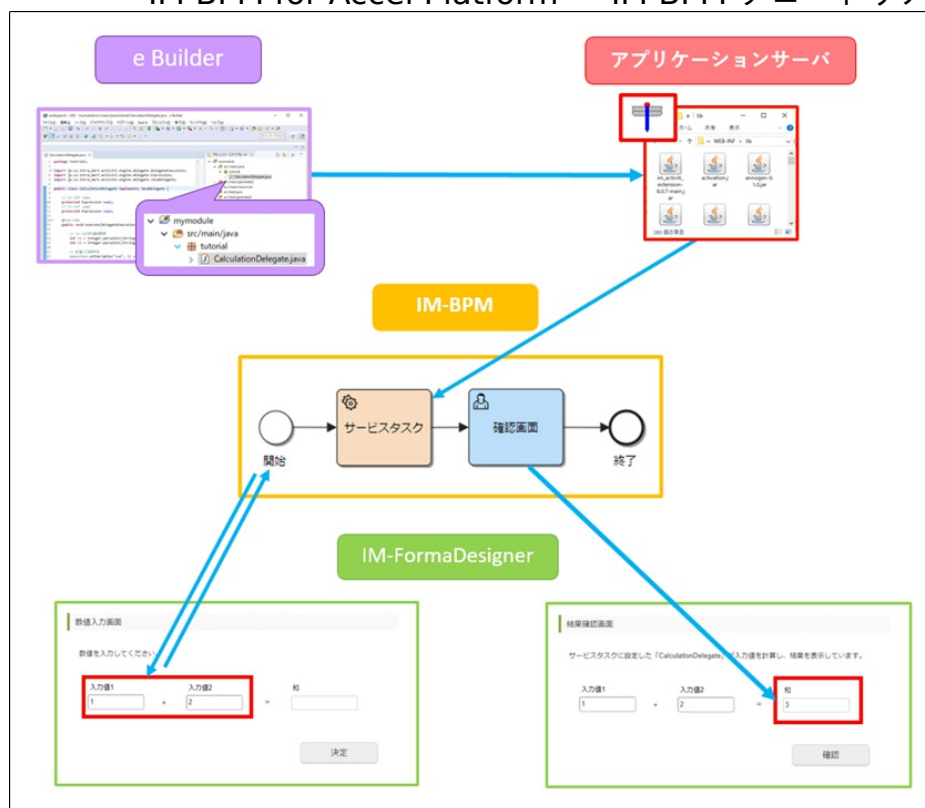
「e Builder」のモジュール・プロジェクトで作成したJAVAクラスはすぐに開発環境へ反映されるため、「サービスタスク」から利用できます。

サービスタスクの詳細については、「IM-BPM プロセスデザイナー 操作ガイド」 - 「タスク」 - 「サービスタスク」もあわせて参照してください。

このチュートリアルは「e Builder」の「モジュール・プロジェクト作成」と「プロジェクトの設定」が完了している必要があります。

それぞれの手順については、以下を参照してください。

- モジュール・プロジェクト作成：「intra-mart e Builder for Accel Platform アプリケーション開発ガイド」 - 「モジュール・プロジェクト作成」
- プロジェクトの設定：「intra-mart e Builder for Accel Platform アプリケーション開発ガイド」 - 「プロジェクトの設定」



図：概要図

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-service_task_usage_calculation.zip](#)



コラム

このチュートリアルで作成するプロセス定義は、以下のリンクからダウンロードできます。

[service_task_usage.bpmn](#)

プロセス定義のアップロード方法については、以下のリンクを参照してください。

- プロセス定義：「IM-BPM プロセスデザイナー 操作ガイド」 - 「プロセス定義のアップロード」



コラム

このチュートリアルで作成するモジュール・プロジェクトは、以下のリンクからダウンロードできます。

[mymodule-1.0.0-service_task_usage.zip](#)

上記のモジュール・プロジェクトをエクスポートしたimmファイルは、以下のリンクからダウンロードできます。

[mymodule-1.0.0-service_task_usage.imm](#)

immファイルの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。

- immファイル：「intra-mart e Builder for Accel Platform アプリケーション開発ガイド」 「e Builder での開発の流れ」

- e BuilderでJAVAクラスを作成する
- プロセス定義を作成する
- 結果を確認する

e BuilderでJAVAクラスを作成する

「サービスタスク」で実行したいJAVAクラスを「e Builder」で作成します。

今回は、「受け取った値を計算して設定する」という簡単なロジックを作成します。

「モジュール・プロジェクト」内で作成することにより、即時ローカル環境で実行可能です。

1. 「モジュール・プロジェクト」にクラスを作成します。

使用する「モジュールプロジェクト」を右クリックし、「新規 (N)」→「クラス」を選択します。

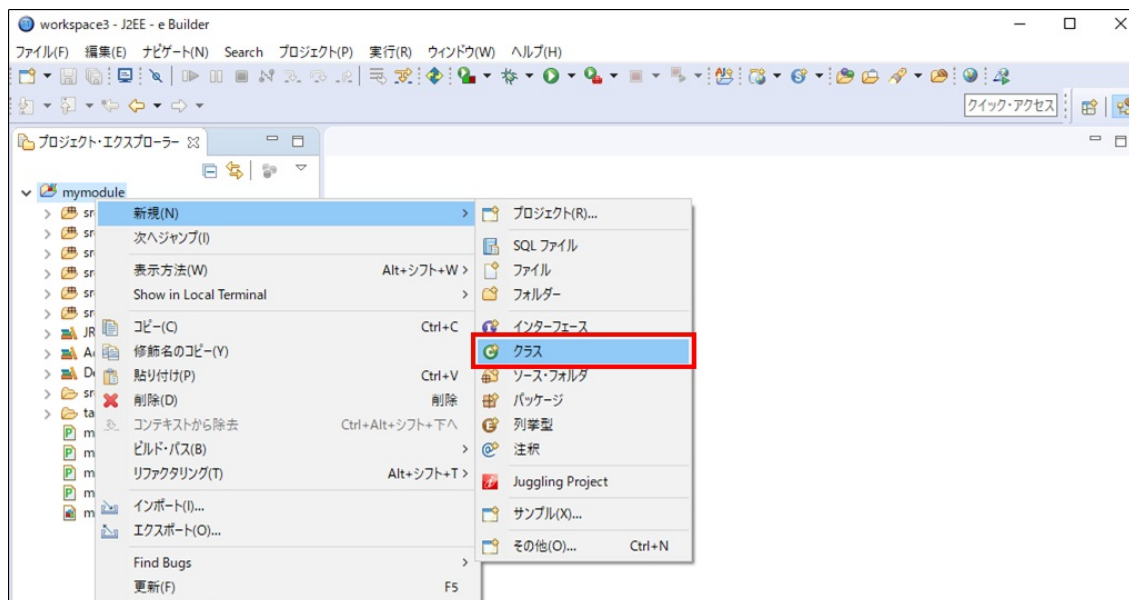


図 : e Builder

2. 「新規Javaクラス」ダイアログが表示されます。

以下のように項目を設定します。

- パッケージ : `tutorial`
- 名前 : `CalculationDelegate`

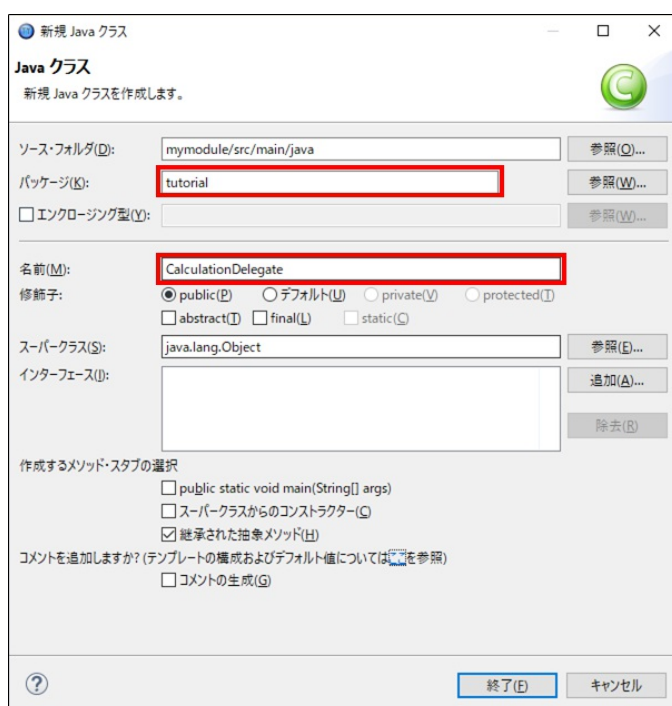
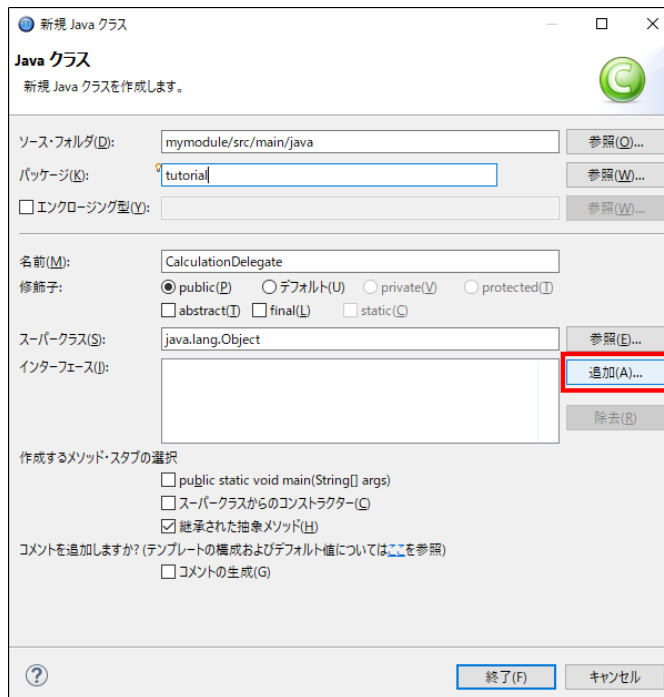


図 : 「新規Javaクラス」 (e Builder)

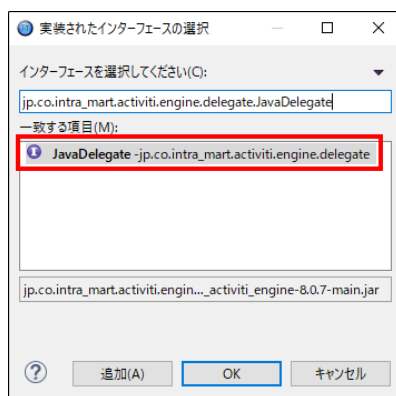
3. インタフェースを設定します。

「新規Javaクラス」ダイアログの中央「インタフェース」の「追加」をクリックします。



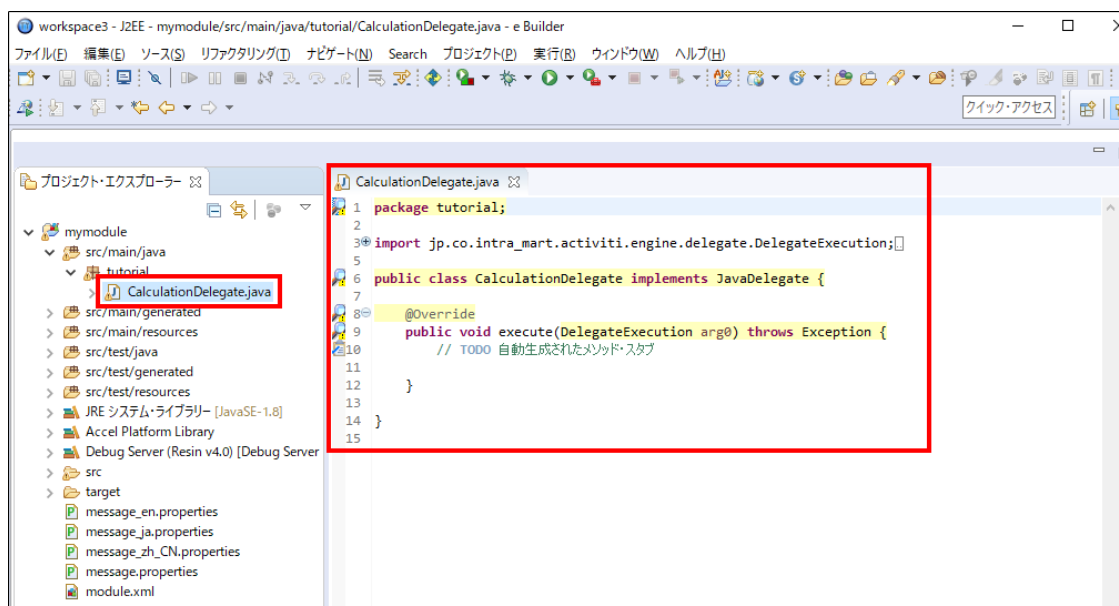
図：「新規Javaクラス」（e Builder）

4. 「実装されたインターフェースの選択」ダイアログで「jp.co.intra_mart.activiti.engine.delegate.JavaDelegate」を入力します。「一致する項目 (M)」に出てくる「JavaDelegate」を選択し、「OK」ボタンをクリックします。



図：「実装されたインターフェースの選択」（e Builder）

5. 「JavaDelegate」が実装された「CalculationDelegate.java」が作成されます。



図：「CalculationDelegate」クラス（e Builder）

6. 作成されたJAVAクラスに、以下のようにコーディングし、保存します。

```

package tutorial;

import jp.co.intra_mart.activiti.engine.delegate.DelegateExecution;
import jp.co.intra_mart.activiti.engine.delegate.Expression;
import jp.co.intra_mart.activiti.engine.delegate.JavaDelegate;

public class CalculationDelegate implements JavaDelegate {

    // フィールド num1
    protected Expression num1;
    // フィールド num2
    protected Expression num2;

    @Override
    public void execute(DelegateExecution execution) throws Exception {

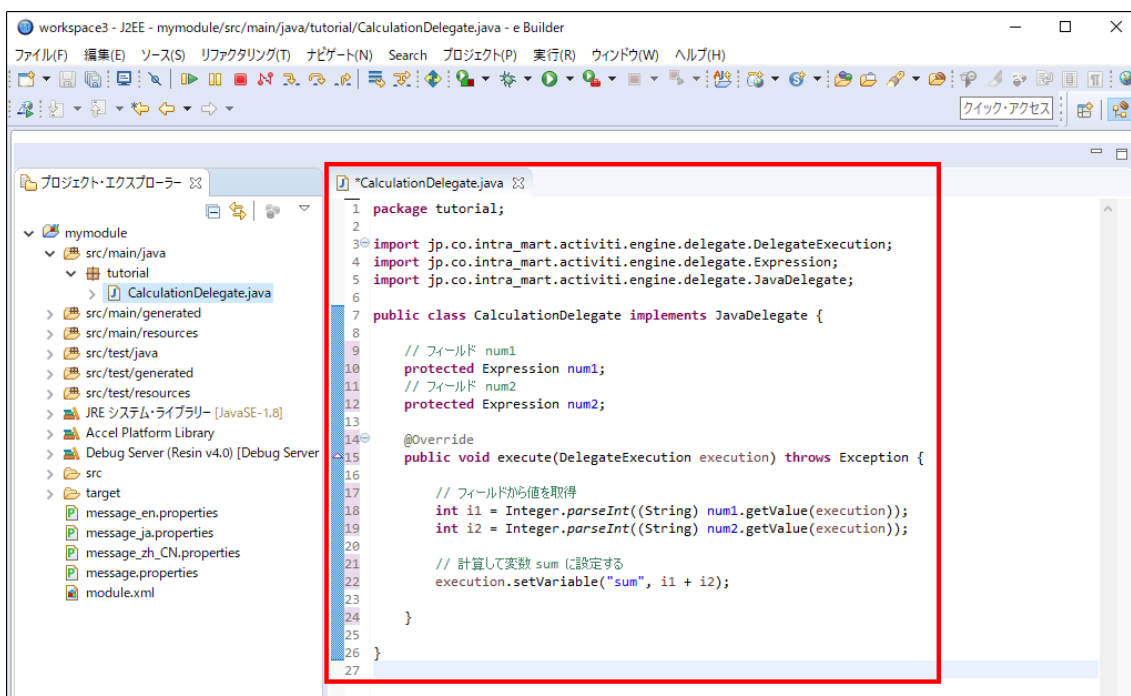
        // フィールドから値を取得
        int i1 = Integer.parseInt((String) num1.getValue(execution));
        int i2 = Integer.parseInt((String) num2.getValue(execution));

        // 計算して変数 sum に設定する
        execution.setVariable("sum", i1 + i2);

    }

}

```



図：「CalculationDelegate」クラス（e Builder）

コラム

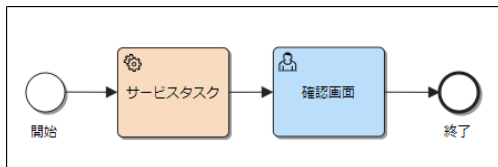
このチュートリアルでは、クラスをアプリケーションサーバに直接デプロイし実行します。実際に配布・運用するには「immファイル」としてエクスポートしてください。環境構築で「IM-Juggling」からWARファイルを生成する際に、その「immファイル」を組み込むことで使用できます。各種インポート方法については、以下のリンクを参照してください。

- immファイルのエクスポート：「[Intra-mart e Builder for Accel Platform アプリケーション開発ガイド / immファイルのエクスポート](#)」
- e Builder での開発の流れ：「[Intra-mart e Builder for Accel Platform アプリケーション開発ガイド](#)」 - 「[e Builder での開発の流れ](#)」

プロセス定義を作成する

以下の図は、入力された値を計算して結果を表示するプロセスです。

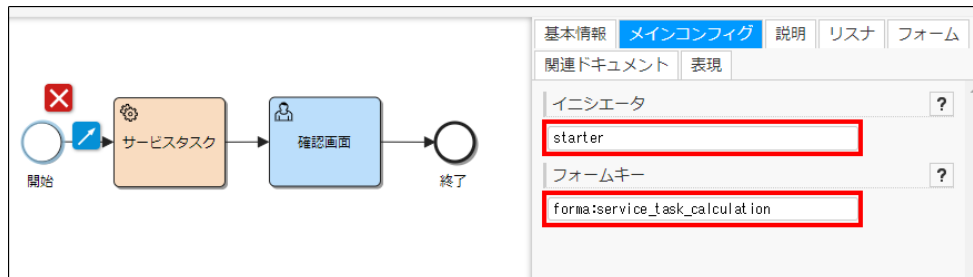
「開始イベント」に設定した「IM-FormaDesigner」のアプリケーション画面で入力された値を「サービスタスク」で計算し、アプリケーション画面で表示します。



図：完成イメージ

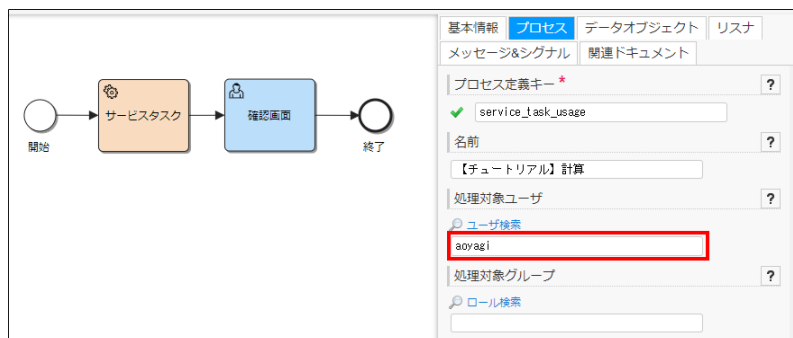
1. 「開始イベント」 を設置します。
2. プロセス開始時に、値を入力するアプリケーション画面を設定します。
実行したユーザを「確認画面」タスクの担当者に設定するために、「イニシエータ」を設定しておきます。
「開始イベント」の「メインコンフィグ」から、以下のように項目を設定します。

- イニシエータ：starter
- フォームキー：forma:service_task_calculation



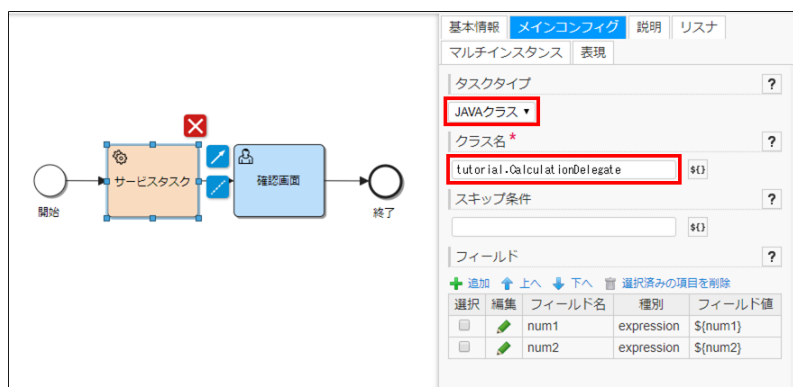
図：「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

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「処理対象ユーザ」に「青柳辰巳（ユーザコード：aoyagi）」を設定します。



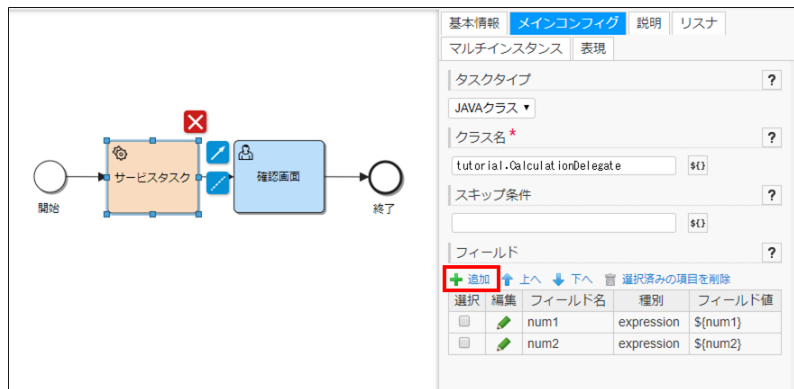
図：プロセス全体 - 「プロパティ」 - 「プロセス」

5. 「サービスタスク」にJAVAクラスを設定します。
「サービスタスク」を選択し、「メインコンフィグ」タブで以下のように項目を設定します。
- タスクタイプ：JAVAクラス
 - クラス名：tutorial.CalculationDelegate



図：「サービスタスク」 - 「プロパティ」 - 「メインコンフィグ」

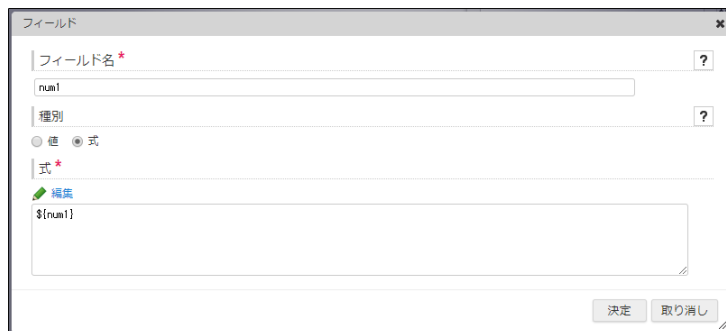
6. 「サービスタスク」のフィールドの設定をします。
「メインコンフィグ」タブの「フィールド」にある「追加」をクリックします。



図：「サービスタスク」 - 「プロパティ」 - 「メインコンフィグ」

7. 以下のように項目を設定します。

- 入力値1
 - フィールド名：num1
 - 種別：式
 - 式：\${num1}
- 入力値2
 - フィールド名：num2
 - 種別：式
 - 式：\${num2}



図：「フィールド」

コラム

上記で設定している値の詳細は以下のとおりです。

- フィールド名
呼び出すクラスのクラス変数（jp.co.intra_mart.activiti.engine.delegate.Expression）の変数名を指定します。
指定されたクラス変数に式が設定されます。
- 式
「IM-FormaDesigner」で作成したアプリケーション画面で、入力フォームに対して設定された「フィールド識別ID」を使用したEL式です。
EL式で設定することにより、画面で入力された値を取得できます。

8. 確認画面の「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」タブから、以下のように項目を設定します。

- 担当者：\${starter}
- フォームキー：forma:service_task_calculation



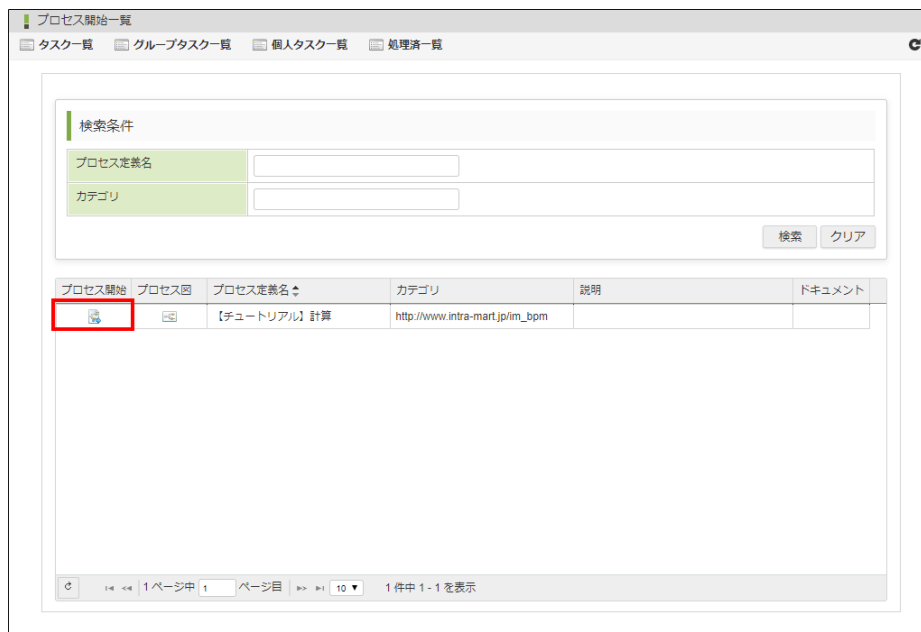
図：「確認画面」-「プロパティ」-「メインコンフィグ」

10. 「終了イベント」を設置します。

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. 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、「数値入力」が表示されます。
4. 適当な数値を入力し、「決定」ボタンをクリックします。



図：数値入力 (IM-FormaDesigner)

5. 「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。

処理	履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当を外
		【チュートリアル】 service_task_calcu			確認画面	50	2018/12/26 17			

図：「タスク一覧」 - 「個人タスク」

6. 「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。
「和」フォームに結果が入っていることを確認します。

サービスタスクに設定した「CalculationDelegate」が入力値を計算し、結果を表示しています。

入力値1: 1 + 入力値2: 2 = 和: 3

確認

図：結果確認 (IM-FormaDesigner)

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 プロセスデザイナー 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。

i 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 プロセスデザイナー 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。

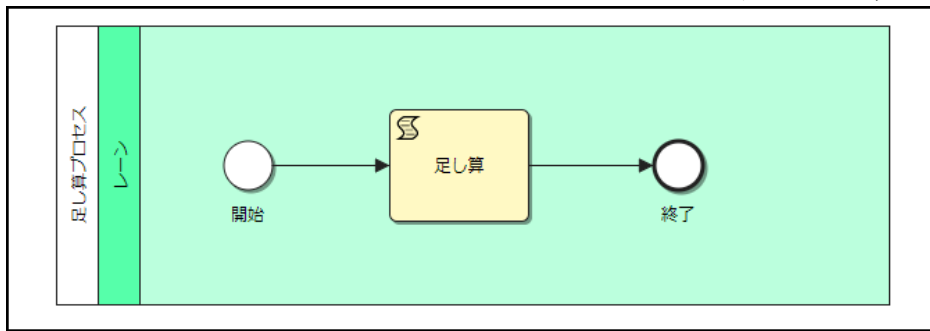
i コラム

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);
}
  
```

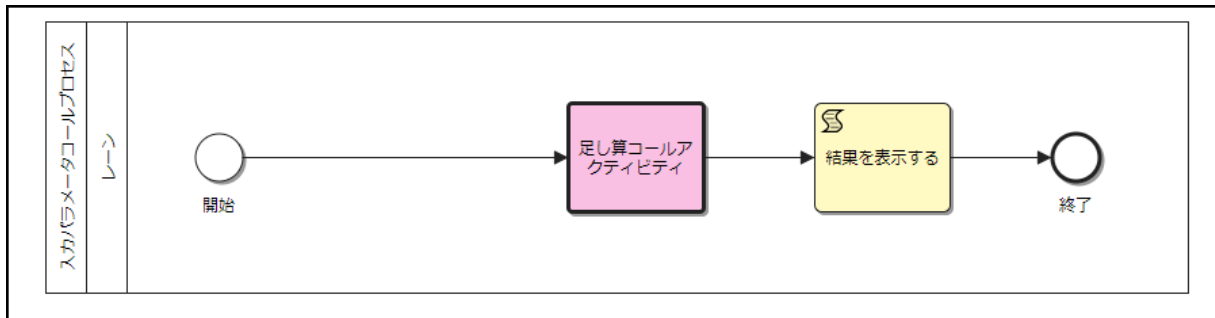
「プロセス定義キー」に `addition_process` を設定します。

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

選択	編集	ソース種別	ソース	ターゲット変数名
☐		sourceExpression	\${123}	val1
☐		sourceExpression	\${4}	val2
☐		sourceExpression	\${-5}	val3
☐		sourceExpression	\${67}	val4
☐		sourceExpression	\${-89}	val5

選択	編集	ソース種別	ソース	ターゲット変数名
☐		source	result	output_result

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。

選択	プロセス定義名	プロセス定義キー	カテゴリ	プロジェクト名	ファイル名	説明
☐	終了条件を利用する	multi_instance_5	http://www.intra-mart.jp/	チュートリアル	multi_instance	
☐	要素変数を利用する	multi_instance_4	http://www.intra-mart.jp/	チュートリアル	multi_instance	
☐	足し算プロセス	addition_process	http://www.intra-mart.jp/	チュートリアル	call_activity_usage	
☐	配列を指定して使用する	multi_instance_2	http://www.intra-mart.jp/	チュートリアル	multi_instance	

決定 取り消し

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。

- 「プロセスデザイナー上のプロセス定義」タブ
「プロセスデザイナー上のプロセス定義」タブでは、プロセスデザイナー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れます。
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 プロセスデザイナー 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。

選択	編集	ソース種別	ソース	ターゲット変数名
☐		sourceExpression	\${123}	val1
☐		sourceExpression	\${4}	val2
☐		sourceExpression	\${-5}	val3
☐		sourceExpression	\${67}	val4
☐		sourceExpression	\${-89}	val5

選択	編集	ソース種別	ソース	ターゲット変数名
☐		source	result	output_result

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')));
}
```

基本情報 **メインコンフィグ** 説明 リスナ マルチインスタンス 表現

スクリプト言語 ?
im_javascript

スクリプトファイル ?

スクリプト * ?

編集

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

結果変数名 ?

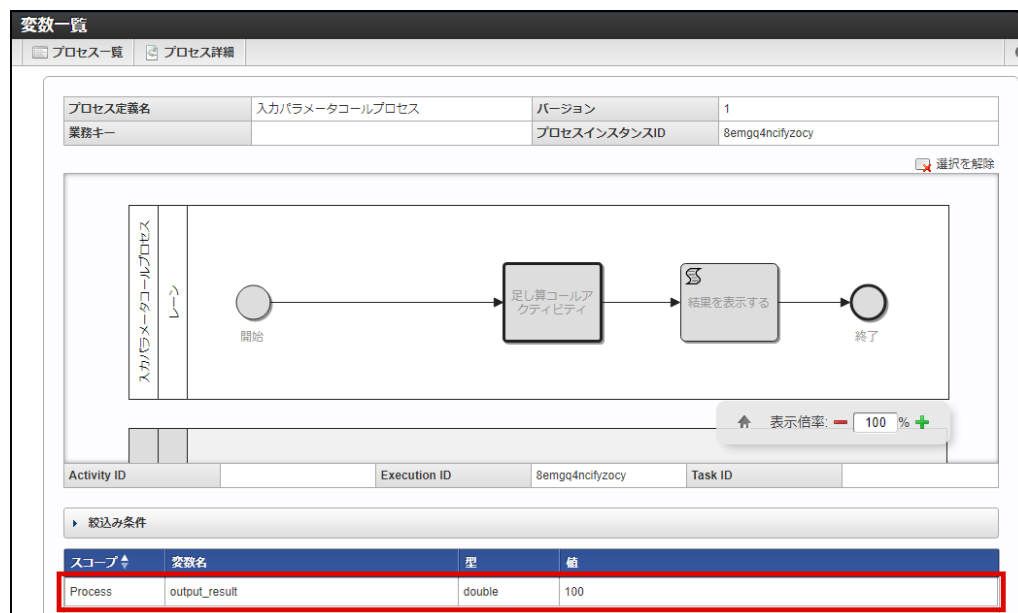
図：スクリプトタスク - メインコンフィグ - スクリプト

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。

```
===== 1 =====
/* Number */
100
```

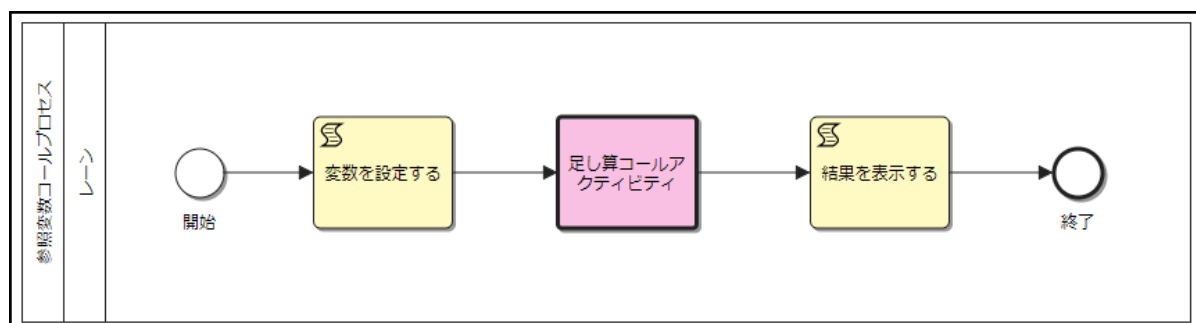
図：結果表示



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

選択	編集	ソース種別	ソース	ターゲット変数名
☐		source	result	output_result

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。

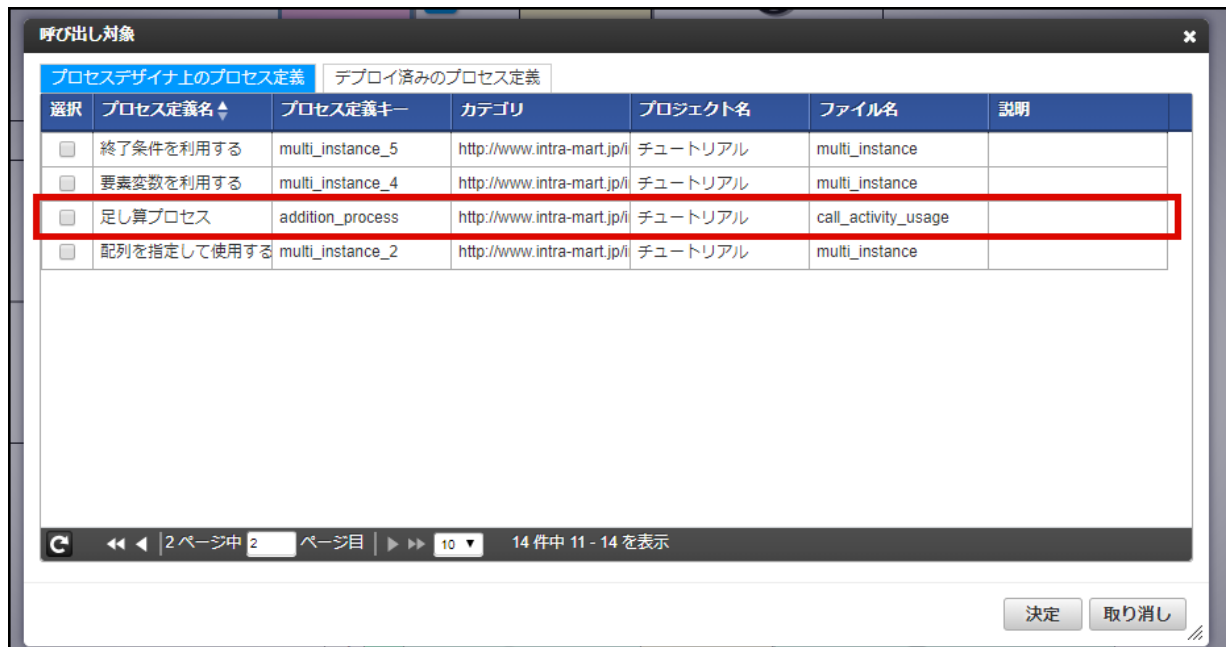
```
function run(variables, execution, entity) {
  entity.setVariable('val1', 1);
  entity.setVariable('val2', 2);
  entity.setVariable('val3', 3);
  entity.setVariable('val4', 4);
  entity.setVariable('val5', 5);
  entity.setVariable('val6', 6);
  entity.setVariable('val7', 7);
  entity.setVariable('val8',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

```
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

- 「出力パラメータ」の「ソース種別」に「変数」を選択します。
 - 「ソース変数名」に変数名 `result` を設定します。
 - 「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립トです。

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

基本情報 メインコンフィグ 説明 リスナ マルチインスタンス 表現

スクリプト言語 ?

im_javascript

スクリプトファイル ?

スクリプト * ?

編集

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

結果変数名 ?

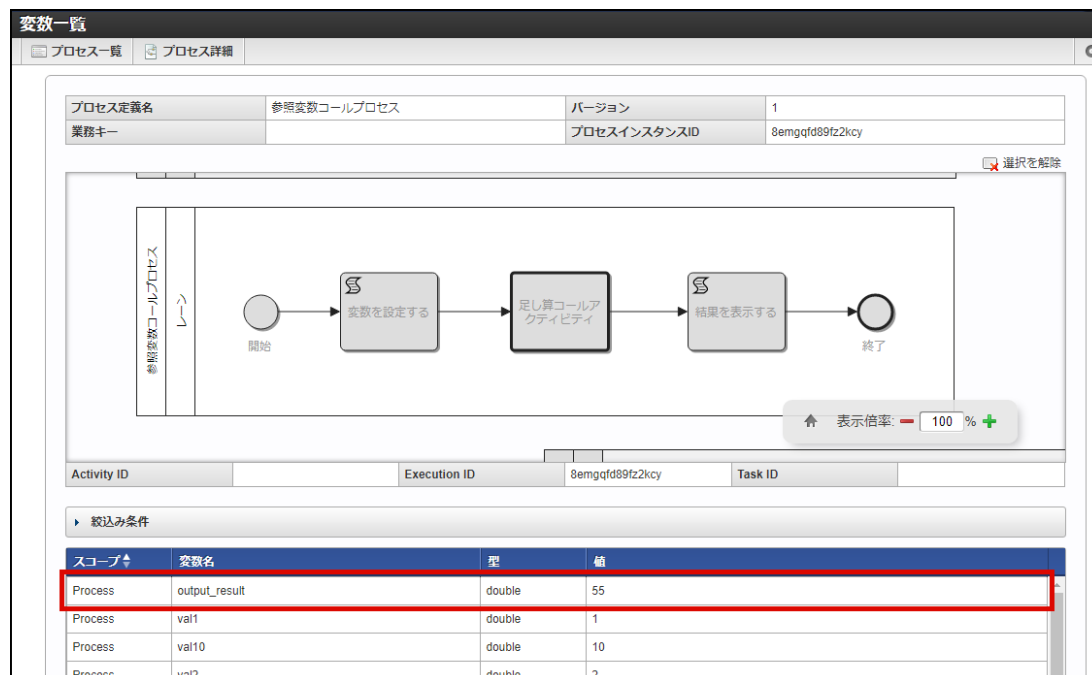
図：スクリプトタスク - メインコンフィグ - スクリプト

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。

```
===== 1 =====
/* Number */
35
```

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。

i 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。

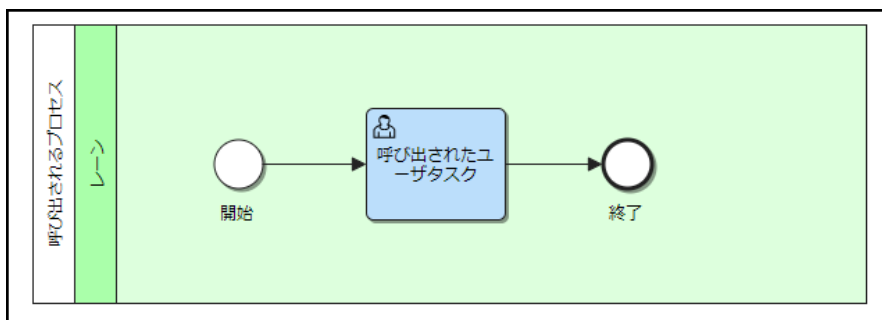
i コラム

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。

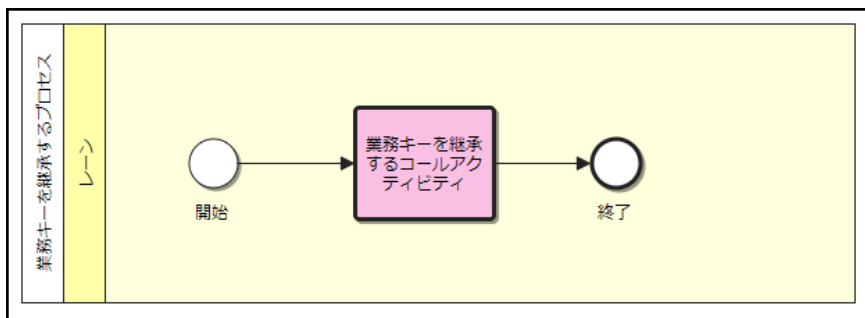


図：呼び出されるプロセス定義

「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。

i 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。

基本情報 | **メインコンフィグ** | 説明 | リスナ | マルチインスタンス

表現

呼び出し対象 *

receive_business_key_process

☒ 業務キーを継承する

☐ 参照可能な変数を継承する

入力パラメータ

+ 追加 - 選択済みの項目を削除

選択	編集	ソース種別	ソース	ターゲット変数名

出力パラメータ

+ 追加 - 選択済みの項目を削除

選択	編集	ソース種別	ソース	ターゲット変数名

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。

プロセス開始一覧

タスク一覧 | グループタスク一覧 | 個人タスク一覧 | 処理済一覧

検索条件

プロセス定義名

カテゴリ

検索 クリア

プロセス開始	プロセス図	プロセス定義名	カテゴリ	説明	ドキュメント
		業務キーを継承するプロセス	http://www.intra-mart.jp/im_bpm		
		業務キーを任意に設定するプロセス	http://www.intra-mart.jp/im_bpm		

プロセス開始確認

? プロセスを開始します。よろしいですか?

業務キー intramart

決定 取り消し

1 ページ中 1 ページ目 10 2 件中 1 - 2 を表示

図：プロセス開始一覧

3. プロセス一覧で確認します。

プロセス一覧

検索条件

業務キー	
プロセス定義ID	
プロセス定義キー	
プロセス定義バージョン	
プロセス定義名	
カテゴリ	
開始日	31 ~ 31
ステータス	☒ 実行中 ☐ 障害中 ☐ 完了 ☐ 全て表示する
変数検索	+

検索 クリア

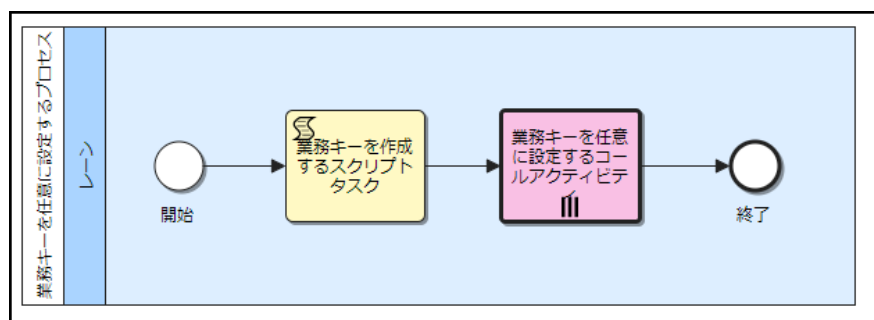
表示設定

詳細	業務キー	プロセス定義名	プロセス定義ID	プロセス定義キー	開始日時	完了日時	ステータス
	intramart	呼び出されるプロセス	receive_business_key_	receive_business_key_	2018/03/16 18:08:22		実行中
	intramart	業務キーを継承するプロセス	inherit_business_key_	inherit_business_key_	2018/03/16 18:08:22		実行中

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。

基本情報 | **メインコンフィグ** | 説明 | リスナ | マルチインスタンス

表現

呼び出し対象 * ?

receive_business_key_process ?

☐ 業務キーを継承する ?

業務キー ?

\${businessKey} ?

☐ 参照可能な変数を継承する ?

入力パラメータ ?

+ 追加 選択済みの項目を削除

選択 編集 ソース種別 ソース ターゲット変数名

出力パラメータ ?

+ 追加 選択済みの項目を削除

選択 編集 ソース種別 ソース ターゲット変数名

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

基本情報 | メインコンフィグ | 説明 | リスナ | **マルチインスタンス**

表現

繰り返しの種別 ?

☐ ループ回数 ☒ 配列

配列 ?

\${businessKeys} ?

☐ 順次実行 ?

要素変数 ?

businessKey

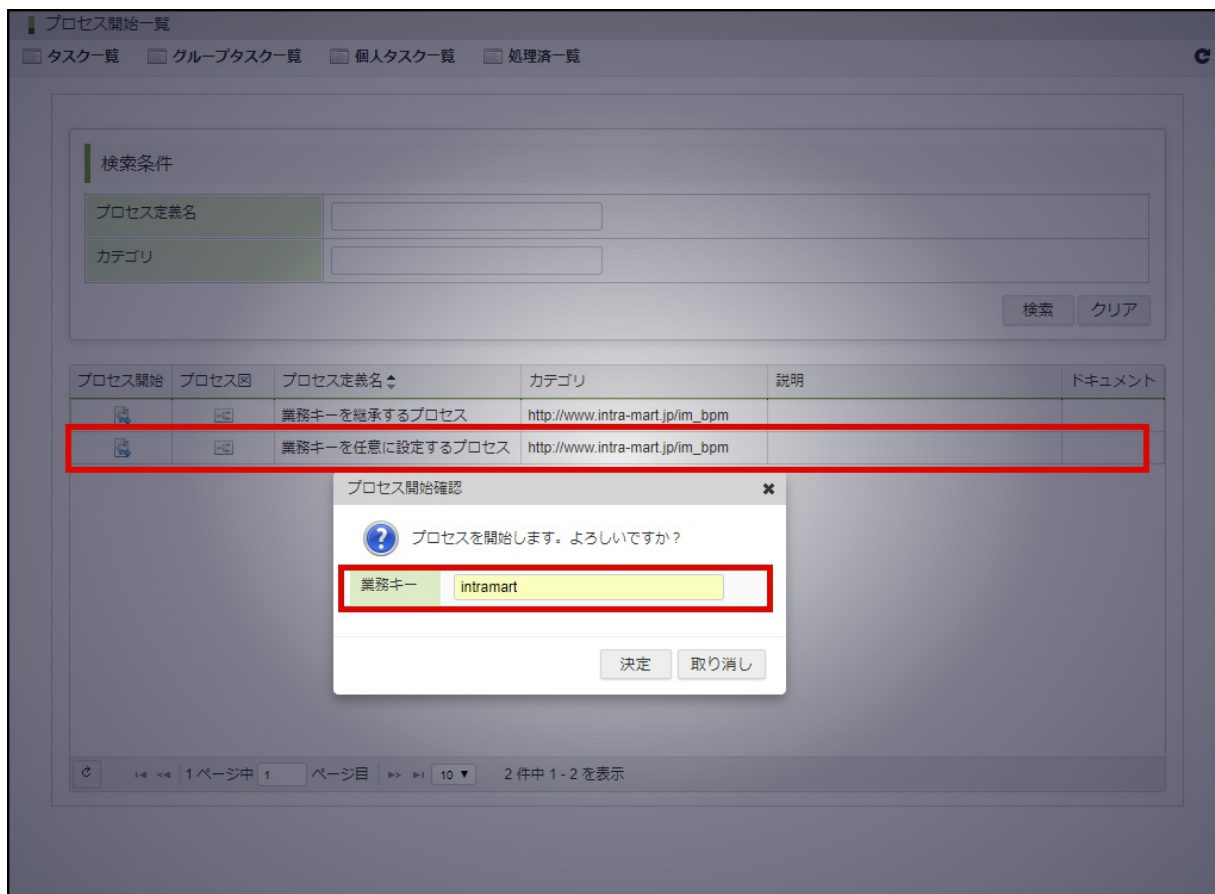
終了条件 ?

? ?

図：マルチインスタンス

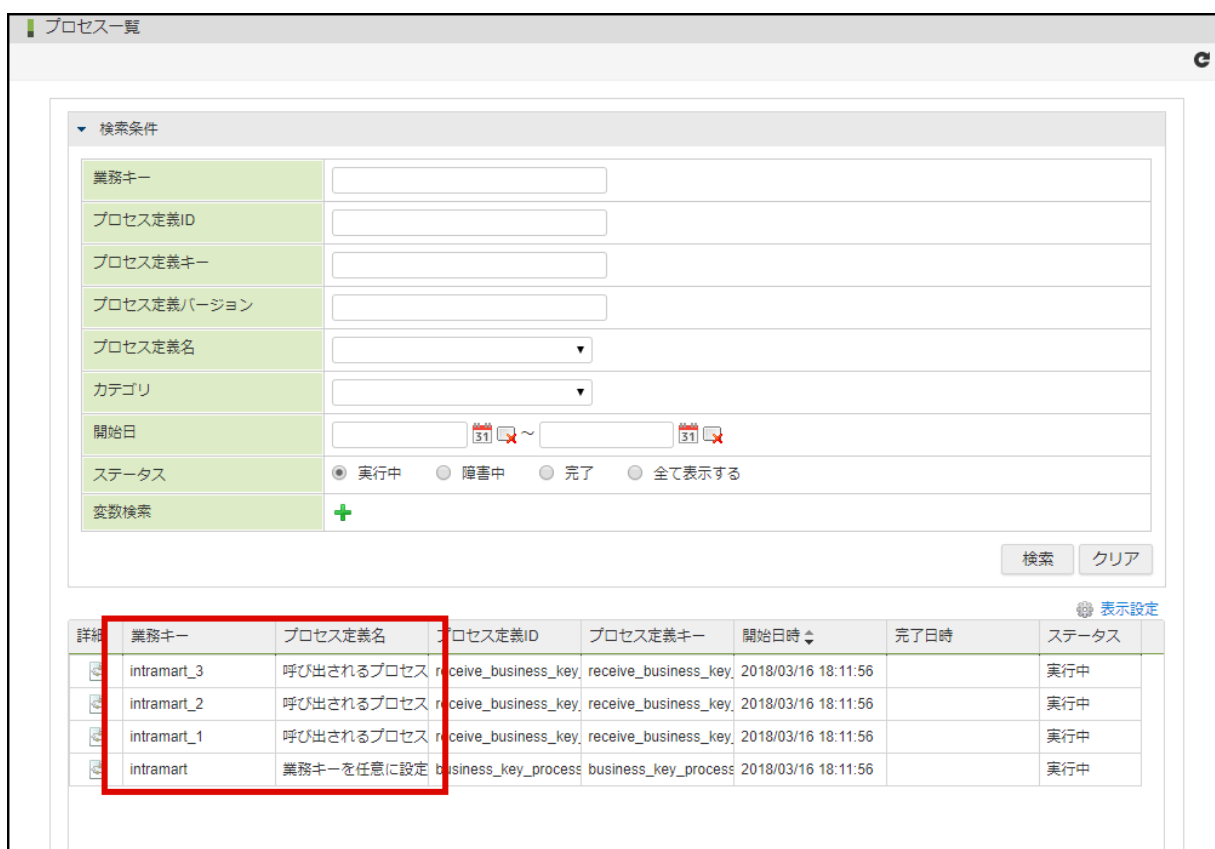
実行結果を確認します。

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

イベントゲートウェイ

イベントゲートウェイを使用する

このチュートリアルでは、イベントゲートウェイを使用した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。

イベントゲートウェイは、接続先のキャッチイベントの状態を評価して遷移先を決定します。

接続先のキャッチイベントで、最初の実行状態になった遷移先のみに遷移します。

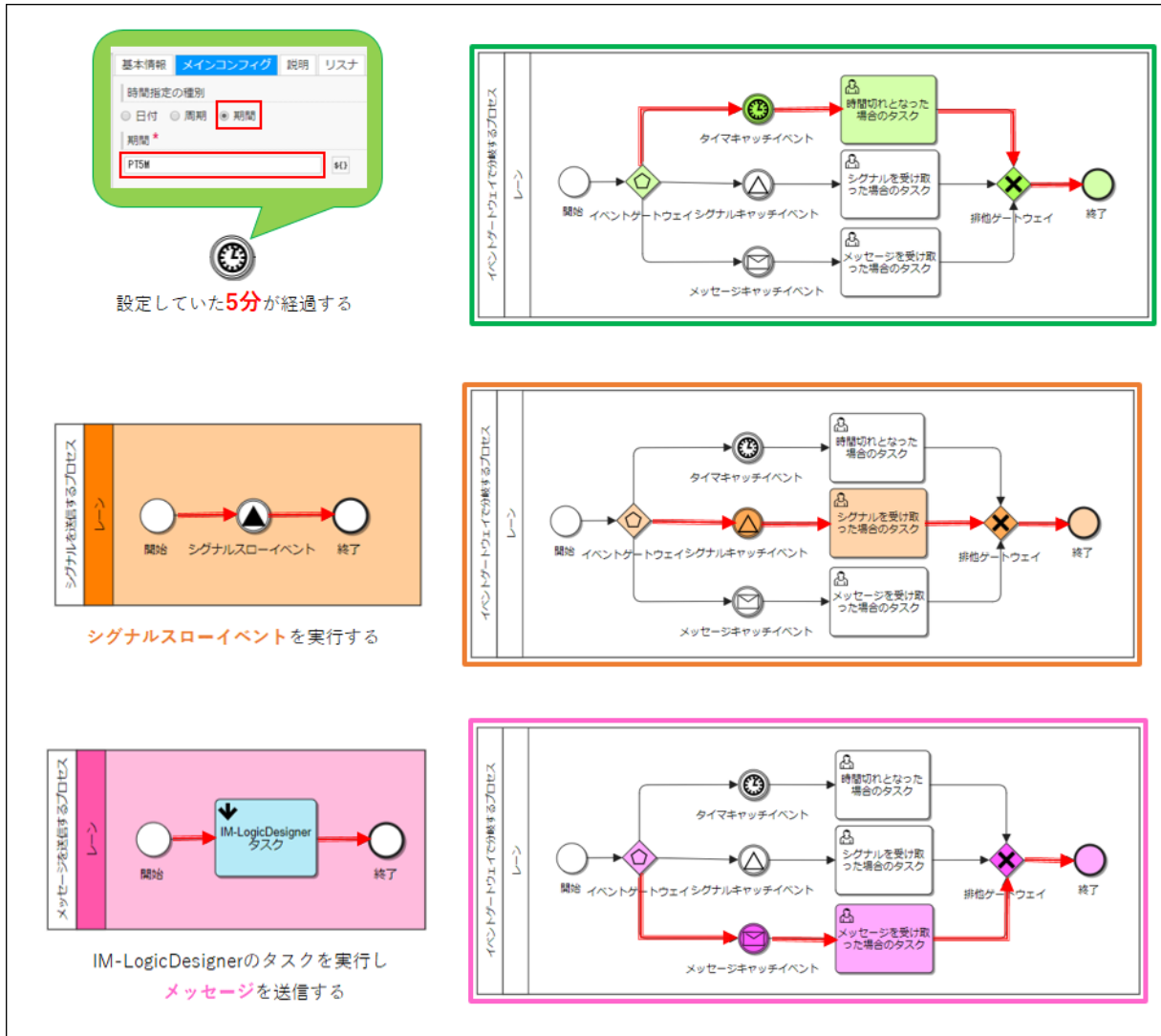
イベントゲートウェイの仕様上1方向のみに遷移するため、合流は排他ゲートウェイで定義します。

イベントゲートウェイの詳細については、「IM-BPM プロセスデザイナー 操作ガイド」 - 「ゲートウェイ」 - 「イベントゲートウェイ」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-LogicDesigner」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」のインポートをしてください。

[im_logicdesigner-data-event_gateway_usage.zip](#)



図：概要図

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

各イベントの詳細については、以下を参照してください。

- タイマキャッチイベント：「IM-BPM プロセスデザイナー 操作ガイド」 - 「タイマキャッチイベント」
- シグナルキャッチイベント：「IM-BPM プロセスデザイナー 操作ガイド」 - 「シグナルキャッチイベント」
- メッセージキャッチイベント：「IM-BPM プロセスデザイナー 操作ガイド」 - 「メッセージキャッチイベント」

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。

IM-LogicDesignerの詳細については、「IM-LogicDesigner仕様書」を参照してください。

i 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_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 プロセスデザイン 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。

i コラム

このチュートリアルのサンプルでは、同一ファイル内に複数のプロセス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。

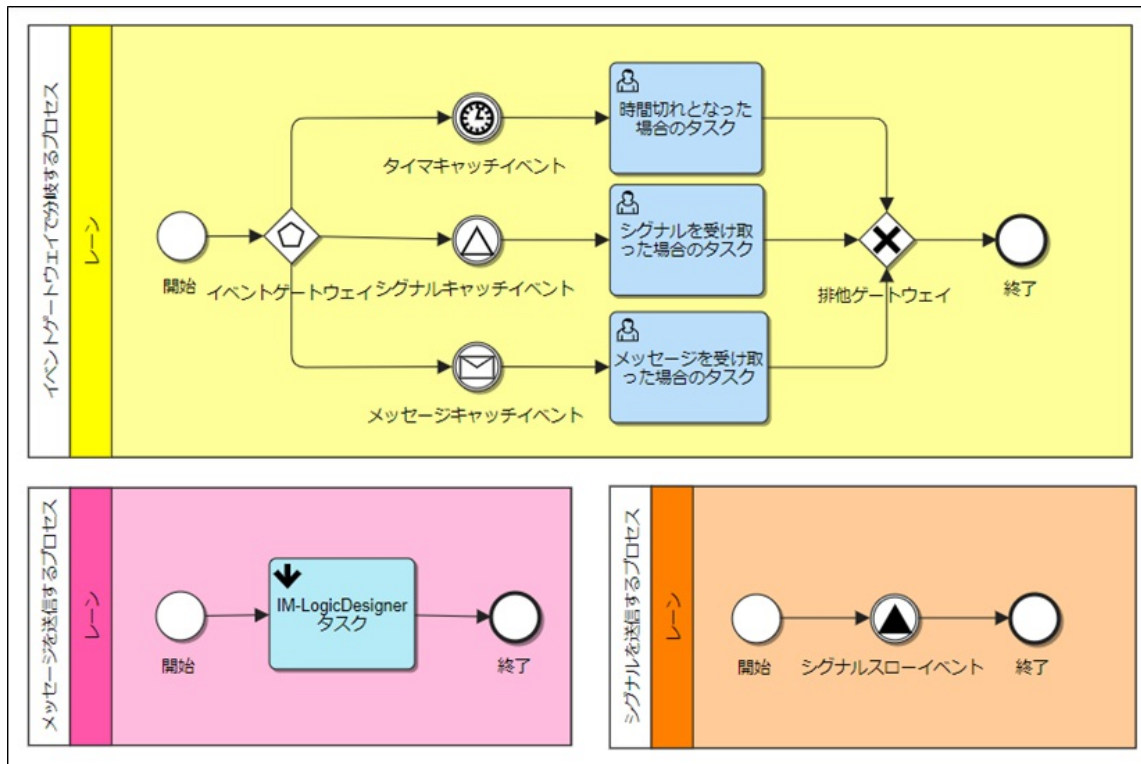
- プロセス定義を作成する
- 結果を確認する

プロセス定義を作成する

このチュートリアルでは、「プール」を使用することで、3つのプロセスを1つのキャンパスに作成しています。

- イベントゲートウェイで分岐するプロセス
- メッセージを送信するプロセス
- シグナルを送信するプロセス

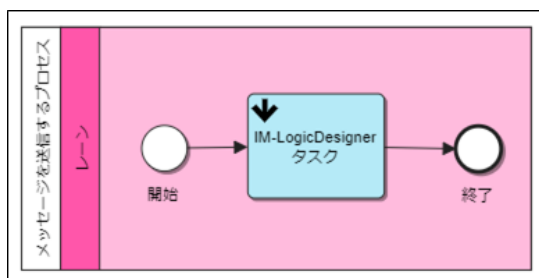
プロセスが進行する条件は「メッセージイベント・シグナルイベントのいずれかを受信する」か「タイマイイベントで設定した時刻に到達する」です。



図：完成イメージ

メッセージを送信するプロセス定義を作成する

「IM-LogicDesignerタスク」を実行することで、「メッセージキャッチイベント」メッセージを送信するプロセスを作成します。

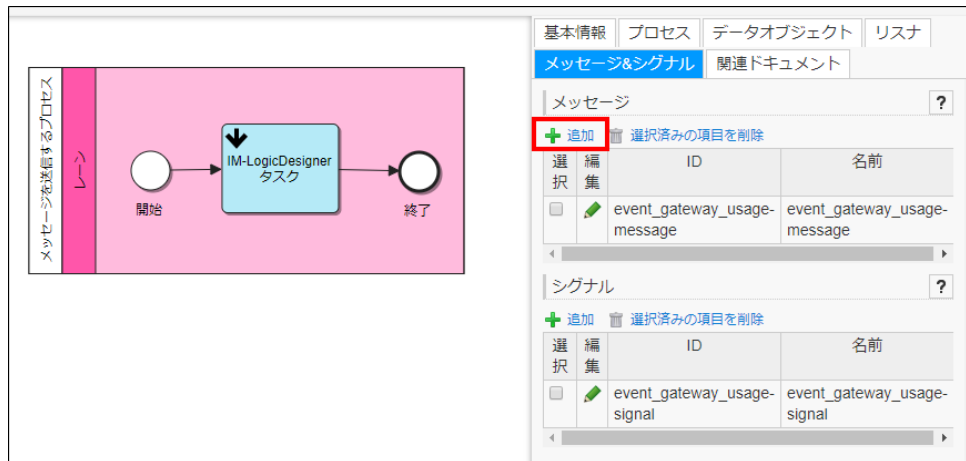


図：完成イメージ

1. 「メッセージ」を登録します。

キャンパスの空白部分をクリックし、プロパティをプロセス全体に切り替えます。

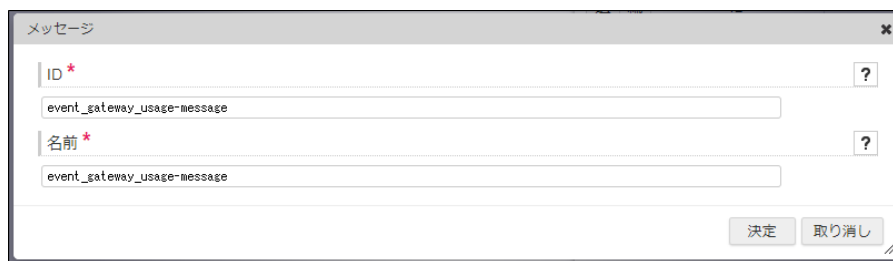
「メッセージ&シグナル」タブから、メッセージの「追加」リンクをクリックします。



図：プロセス全体 - 「プロパティ」 - 「メッセージ&シグナル」

2. 以下のとおりに「メッセージ」を登録します。

- ID : event_gateway_usage-message
- 名前 : event_gateway_usage-message

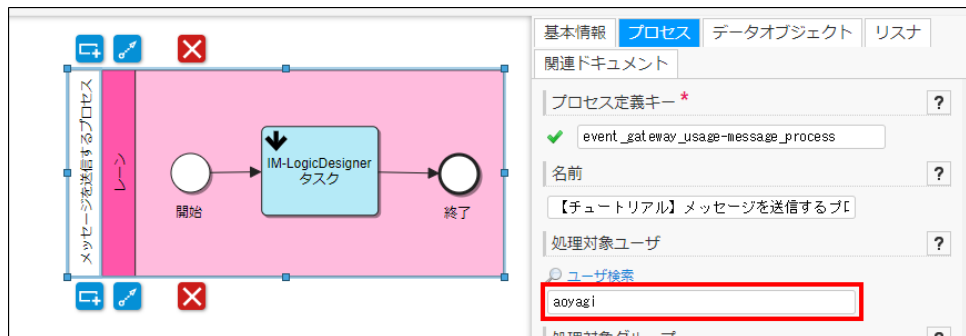


図：「メッセージ」

3. 「開始イベント」を設置します。

4. 「処理対象ユーザ」を設定します。

プールをクリックし、「プロセス」タブから担当者を aoyagi に設定します。



図：プロセス全体 - 「プロパティ」 - 「プロセス」

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ます。

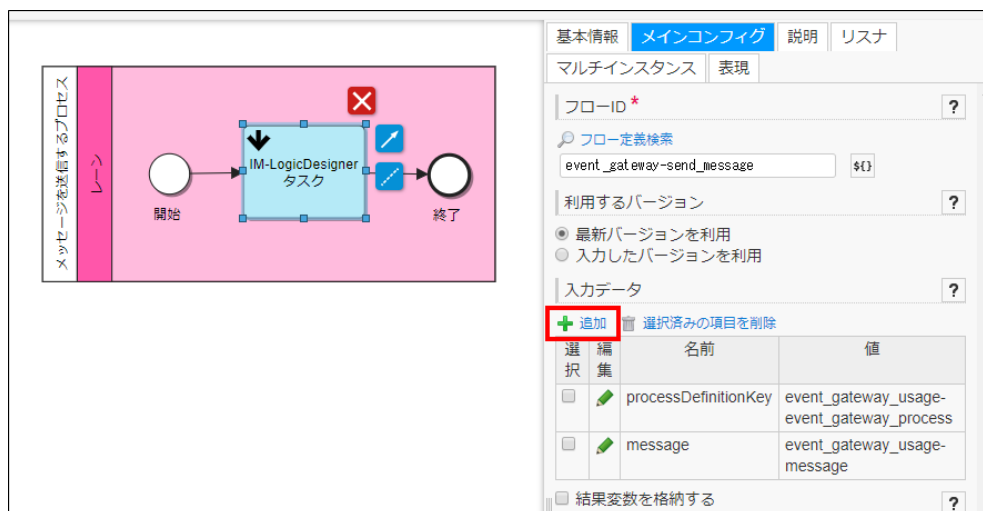
- フローID : event_gateway-send_message
- 利用するバージョン : 最新バージョンを利用



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

6. 入力データを設定します。

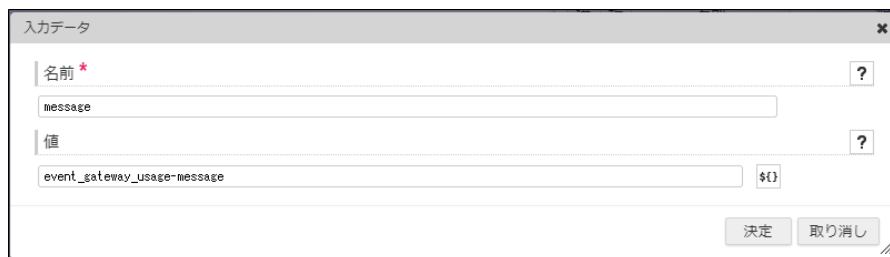
「IM-LogicDesignerタスク」を選択し、「メインコンフィグ」タブから、入力データの「追加」リンクをクリックします



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

7. 以下のとおりに値を登録し、保存します。

- 入力データ1
 - 名前: message
 - 値: event_gateway_usage-message
- 入力データ2
 - 名前: processDefinitionKey
 - 値: event_gateway_usage-event_gateway_process

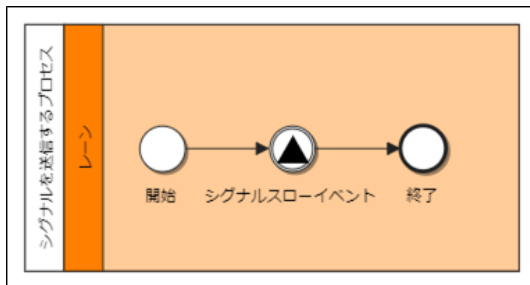


図：「入力データ」

8. 「終了イベント」を設置し、保存します。

シグナルを送信するプロセス定義を作成する

「シグナルスローイベント」を実行することで、シグナルを送信するプロセスを作成します。



図：完成イメージ

1. 「シグナル」を登録します。

キャンパスの空白部分をクリックし、プロパティをプロセス全体に切り替えます。

「メッセージ&シグナル」タブから、「シグナル」の「追加」リンクをクリックします。



図：プロセス全体 - 「プロパティ」 - 「メッセージ&シグナル」

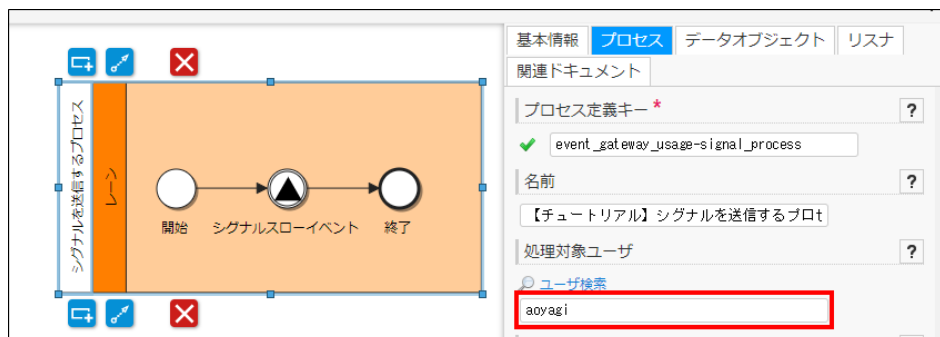
2. 以下のとおりに値を登録し、保存します。

- ID : event_gateway_usage-signal
- 名前 : event_gateway_usage-signal

図：「シグナル」

3. 「処理対象ユーザ」を設定します。

ツールをクリックし、「プロセス」タブから担当者を aoyagi に設定します。



図：プロセス全体 - 「プロパティ」 - 「プロセス」

4. 「開始イベント」を設置します。

5. 参照シグナルを設定します。

「シグナルスローイベント」を選択し、「メインコンフィグ」タブから、「参照シグナル」のプルダウンで event_gateway_usage-signal (ID: event_gateway_usage-signal) を選択します。



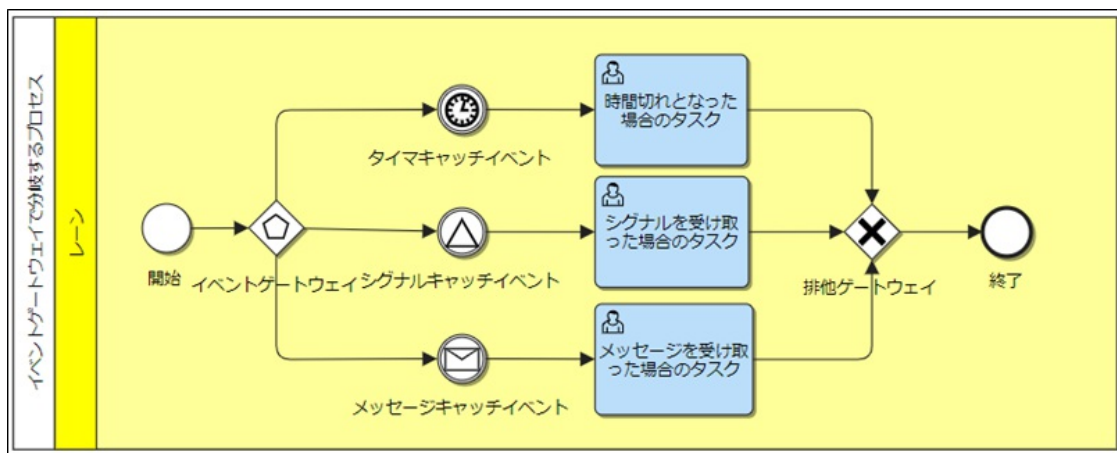
図：「シグナルスローイベント」 - 「プロパティ」 - 「メインコンフィグ」

6. 「終了イベント」を設置し、保存します。

イベントゲートウェイで分岐するプロセス定義を作成する

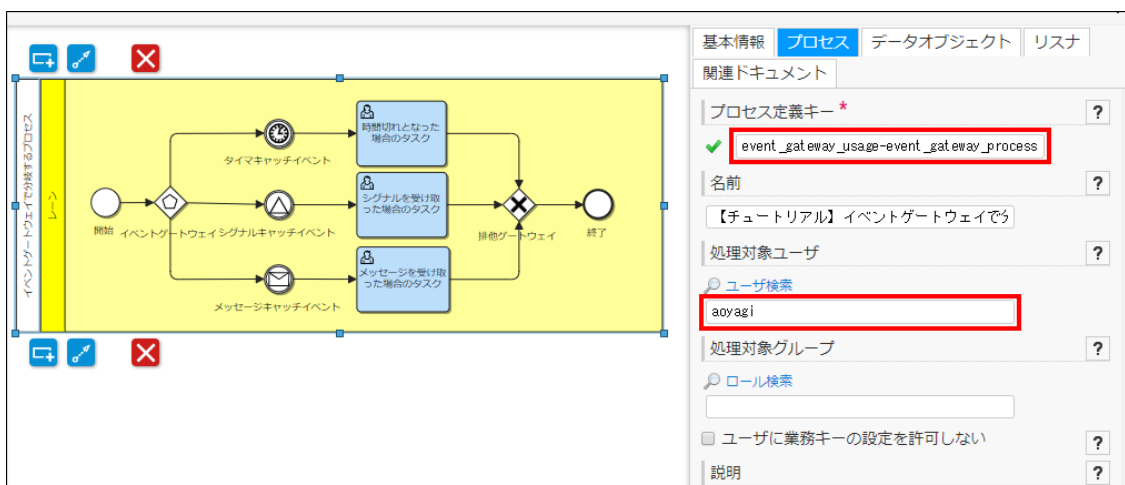
イベントゲートウェイで分岐するプロセスを作成します。

プロセスが進行する条件は「メッセージイベント・シグナルイベントのいずれかを受信する」か「タイマイベントで設定した時刻に到達する」です。どのイベントが発生したか「プロセス詳細」画面から確認できるように、各イベントの後にタスクを設置します。



図：完成イメージ

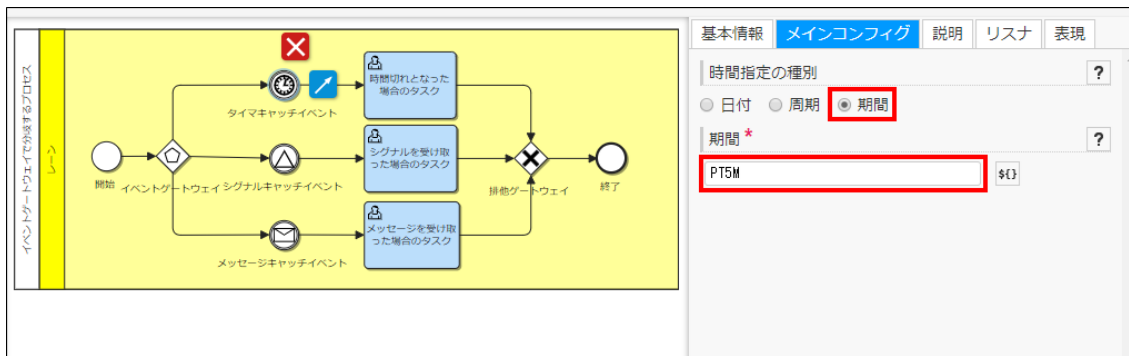
1. プロセスの設定を行います。
プールをクリックし、「プロセス」タブから、以下のように項目を設定します。
 - プロセス定義キー：event_gateway_usage-event_gateway_process
 - 処理対象ユーザ：aoyagi



図：プロセス全体 - 「プロパティ」 - 「プロセス」

2. 「開始イベント」を設置します。
3. 「イベントゲートウェイ」を設置します。
4. 1つ目の分岐である「タイマキャッチイベント」を設置します。
5. 「タイマキャッチイベント」の起動時間を設定します。
「タイマキャッチイベント」を選択し、「メインコンフィグ」タブから、以下のように項目を設定します。

- 時間指定の種別：期間
- 期間：PT5M



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」

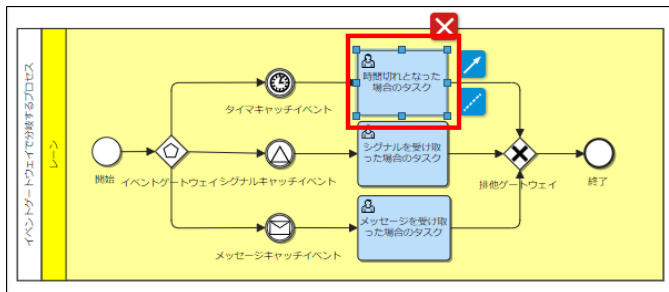
コラム

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

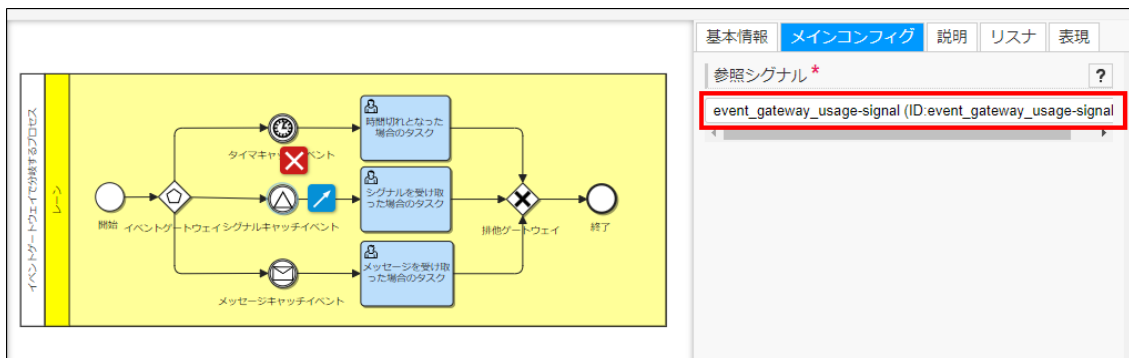
2日→P2D

6. 「タイマキャッチイベント」が起動したことを確認するための「ユーザタスク」を設置します。



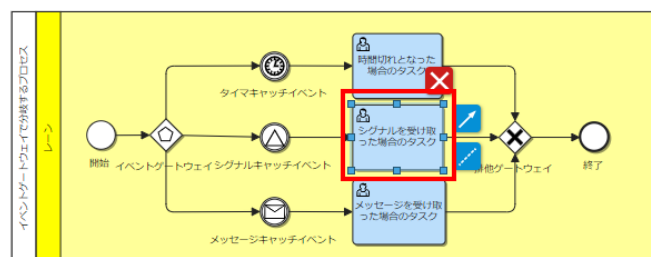
図：「ユーザタスク」

7. 2つ目の分岐である「シグナルキャッチイベント」を設置します。
8. 参照シグナルを設定します。
「シグナルキャッチイベント」を選択し、「メインコンフィグ」タブから、「参照シグナル」のプルダウンで `event_gateway_usage-signal` (ID: `event_gateway_usage-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」

9. 「シグナルキャッチイベント」が起動したことを確認するための「ユーザタスク」を設置します。

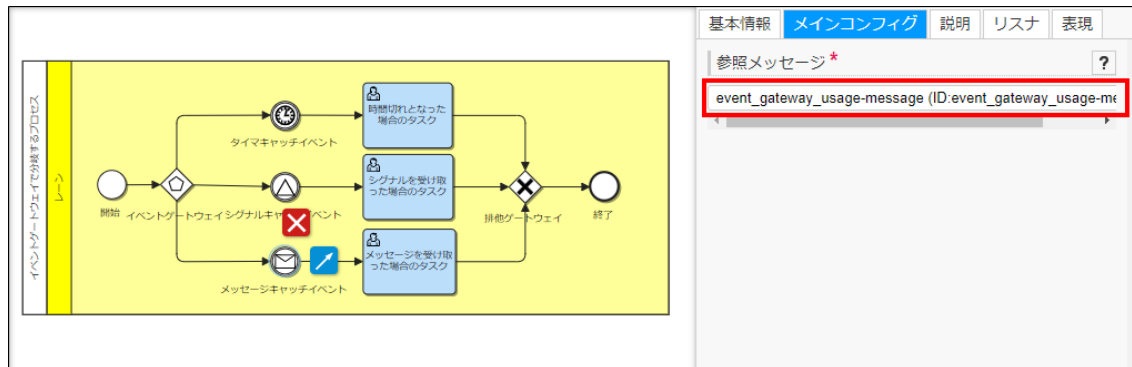


図：「ユーザタスク」

10. 3つ目の分岐である「メッセージキャッチイベント」を設置します。

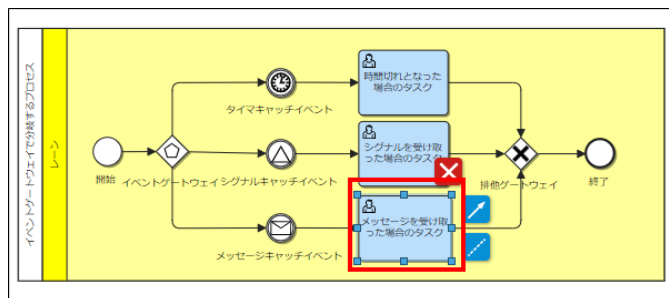
11. 参照メッセージを設定します。

「メッセージキャッチイベント」を選択し、「メインコンフィグ」タブから、「参照メッセージ」のプルダウンで `event_gateway_usage-message` (ID: `event_gateway_usage-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」

12. 「メッセージキャッチイベント」が起動したことを確認するための「ユーザタスク」を設置します。



図：「ユーザタスク」

13. 分岐の合流先として「排他ゲートウェイ」を設置します。

14. 「終了イベント」を設置して保存します。

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。

- 「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示します。
- プロセス開始一覧から、「イベントゲートウェイで分岐するプロセス」の「」のアイコンをクリックします。

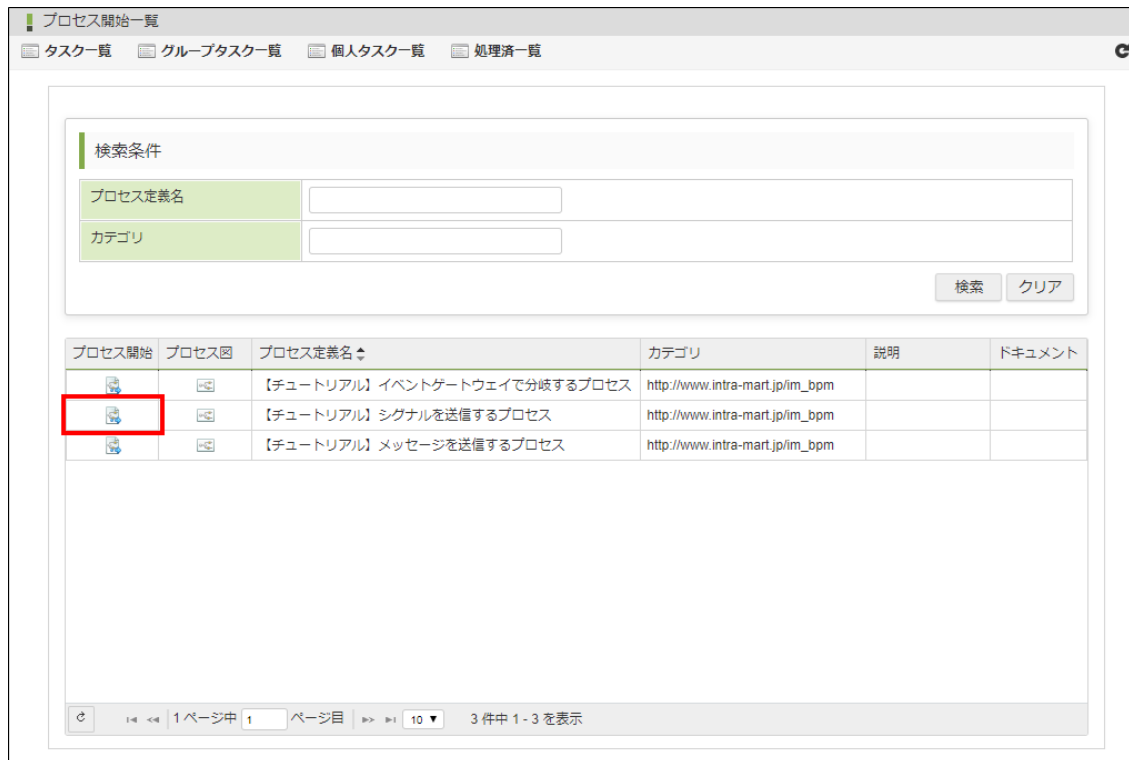
プロセス開始	プロセス図	プロセス定義名	カテゴリ	説明	ドキュメント
		【チュートリアル】 イベントゲートウェイで分岐するプロセス	http://www.intra-mart.jp/im_bpm		
		【チュートリアル】 シグナルを送信するプロセス	http://www.intra-mart.jp/im_bpm		
		【チュートリアル】 メッセージを送信するプロセス	http://www.intra-mart.jp/im_bpm		

図：「プロセス開始一覧」

シグナルキャッチイベントへ分岐させる場合

「シグナルキャッチイベント」を受けて進行するタスクを確認します。

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、「シグナルを送信するプロセス」の「」アイコンをクリックします。



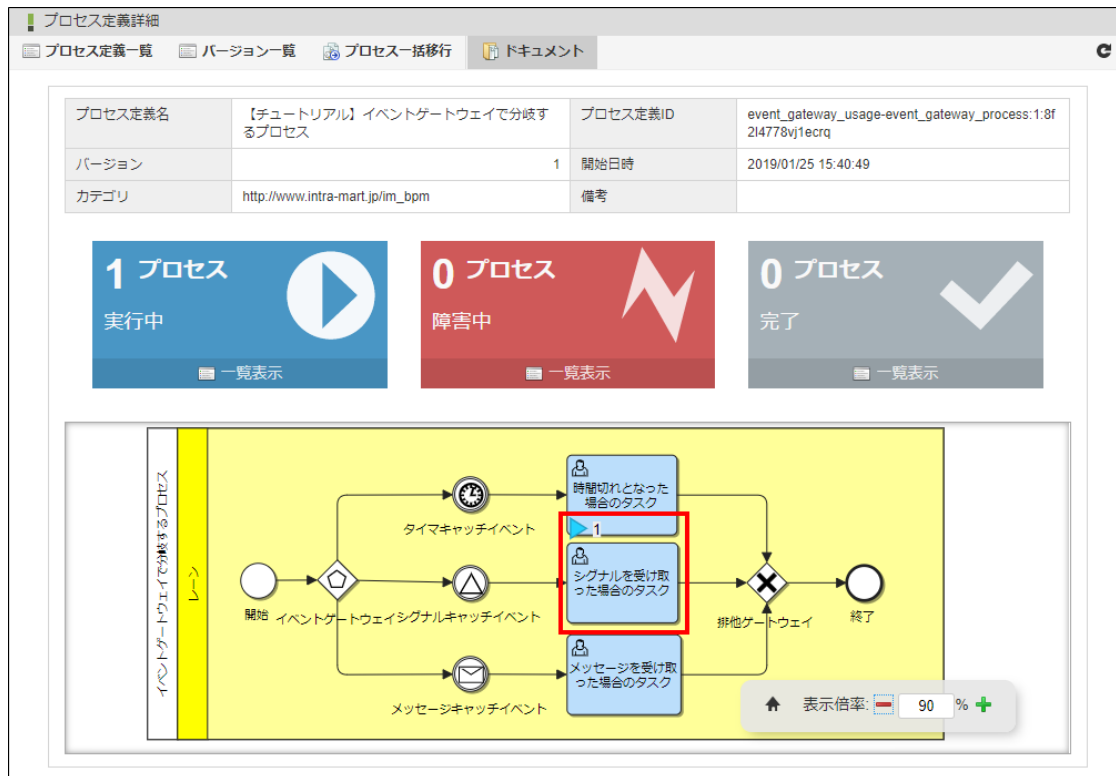
図：「プロセス開始一覧」

3. 「プロセス定義詳細」画面を表示します。
- 「サイトマップ」→「BPM」→「プロセス定義一覧」から、「イベントゲートウェイで分岐するプロセス」の「」をクリックします。



図：「プロセス定義一覧」

4. 「プロセス詳細」画面を表示します。
- 「シグナルキャッチイベント」の先にある「シグナルを受け取った場合のタスク」に進行していることを確認します。

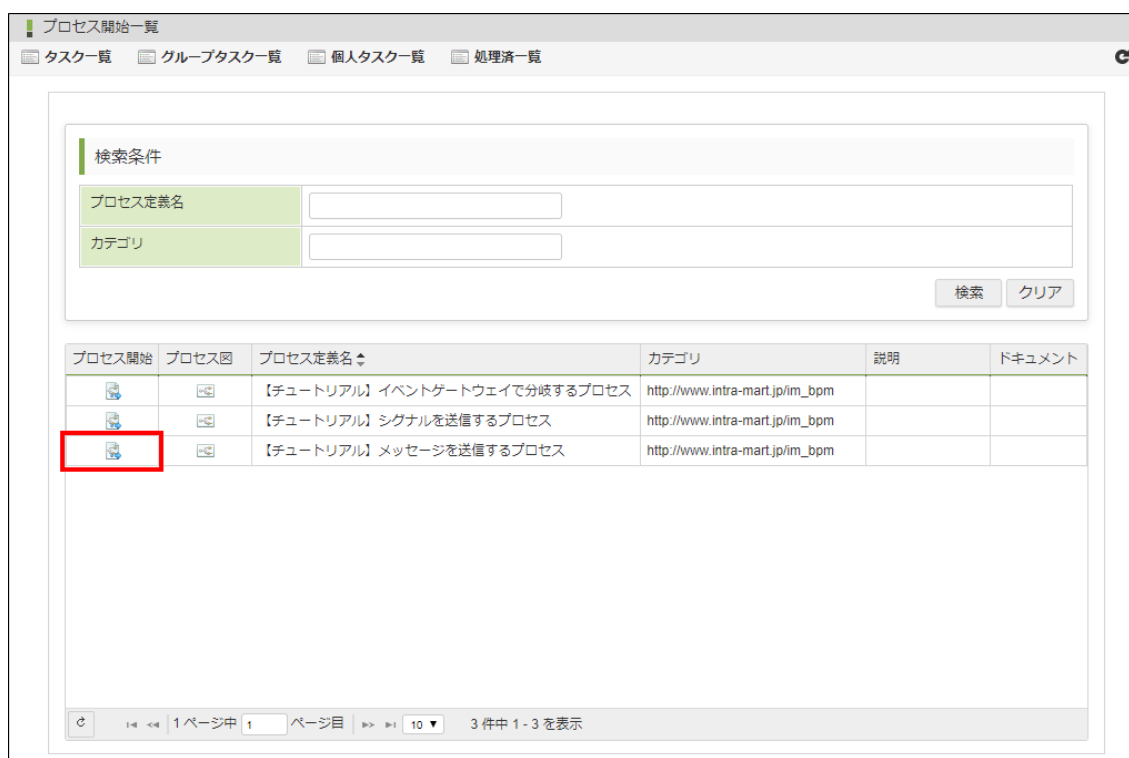


図：「プロセス定義詳細」

メッセージキャッチイベントへ分岐させる場合

「メッセージキャッチイベント」を受けて進行するタスクを確認します。

1. プロセス開始一覧から、「メッセージを送信するプロセス」の「」アイコンをクリックします。



図：「プロセス開始一覧」

2. 「プロセス定義詳細」画面を表示します。

「サイトマップ」→「BPM」→「プロセス定義一覧」から、「イベントゲートウェイで分岐するプロセス」の「」をクリックします。



図：「プロセス定義一覧」

3. 「メッセージキャッチイベント」の先にある「メッセージを受け取った場合のタスク」に進行していることを確認します。



図：「プロセス定義詳細」

タイムキャッチイベントへ分岐させる場合

「タイムキャッチイベント」を受けて進行するタスクを確認します。
今回のタイムキャッチイベントは「PT5M」と設定したため、5分後にイベントゲートウェイから進行していることを確認します。

1. 「プロセス定義詳細」画面を表示します。

「サイトマップ」→「BPM」→「プロセス定義一覧」から、「イベントゲートウェイで分岐するプロセス」の「」をクリックします。



図：「プロセス定義一覧」

2. 「タイマキャッチイベント」の先にある「時間切れとなった場合のタスク」に進行していることを確認します。



図：「プロセス定義詳細」

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 プロセスデザイナー 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 プロセスデザイナー 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。

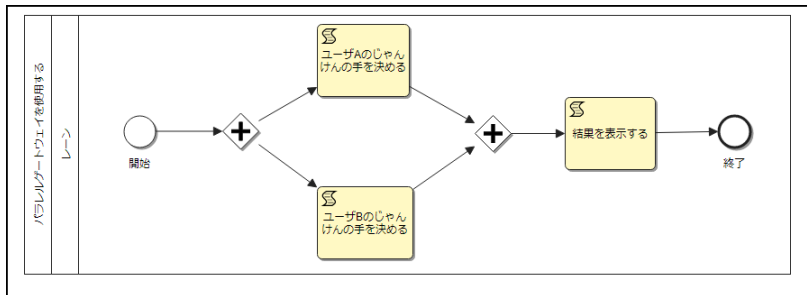
■ パラレルゲートウェイを使用し、並列処理を行う

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。

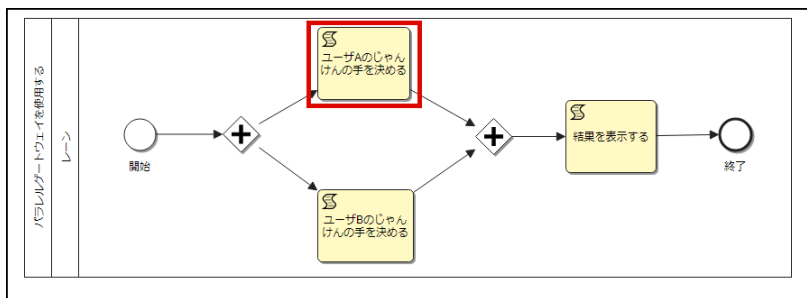
ユーザA、および、ユーザBの手を選択するスクリプトタスクを並列に実行します。

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。

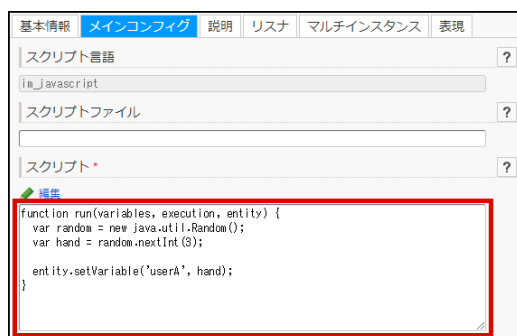


図：スクリプトタスク「ユーザAのじゃんけんの手を決める」

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。

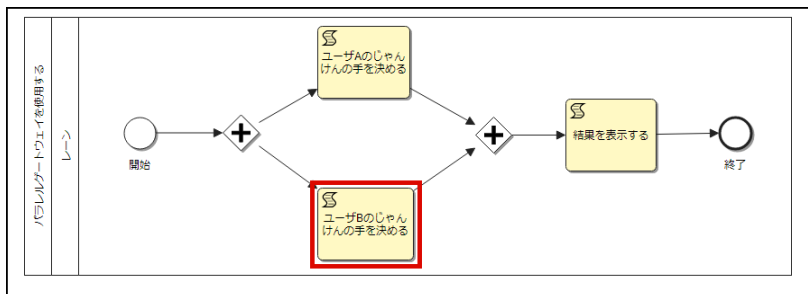
```
function run(variables, execution, entity) {
  var random = new java.util.Random();
  var hand = random.nextInt(3);

  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。

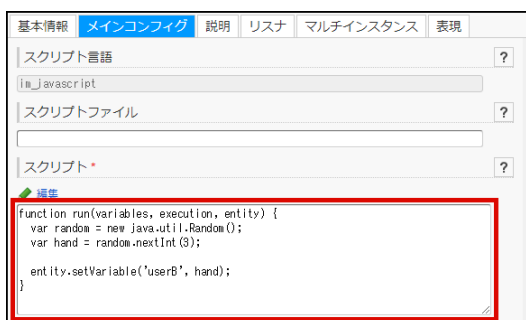


図：スクリプトタスク「ユーザーBのじゃんけんの手を決める」

スクリプトタスク「ユーザーBのじゃんけんの手を決める」にユーザー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。

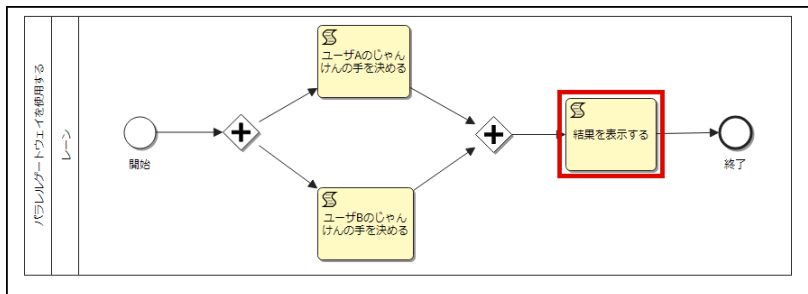
```
function run(variables, execution, entity) {
  var random = new java.util.Random();
  var hand = random.nextInt(3);

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



図：スクリプトタスク「ユーザー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」にユーザーAとユーザー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');
```

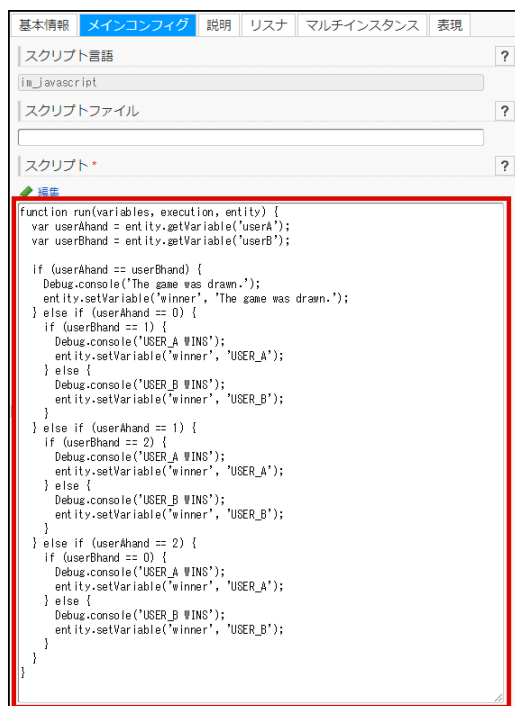
```
    }
  } else if (userAhand == 2) {
    if (userBhand == 0) {
      Debug.console('USER_A WINS');
```

```
      entity.setVariable('winner', 'USER_A');
```

```
    } else {
      Debug.console('USER_B WINS');
```

```
      entity.setVariable('winner', 'USER_B');
```

```
    }
  }
}
```



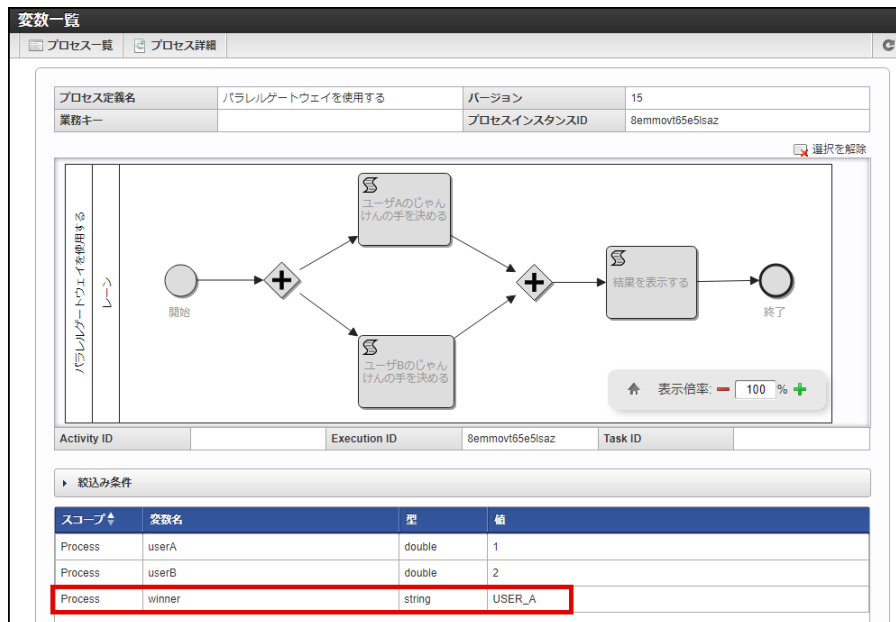
図：スクリプトタスク「結果を表示する」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

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」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 プロセスデザイナー 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 プロセスデザイナー 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。

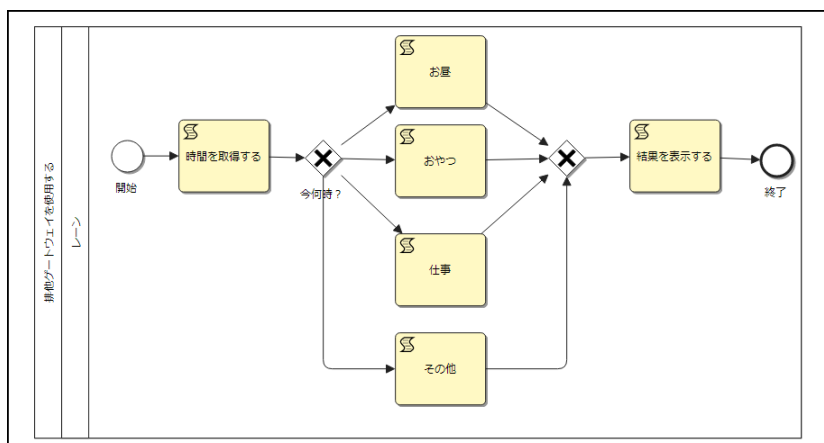
- 排他ゲートウェイを使用し、複数の分岐先から1つの遷移先を選択する

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。

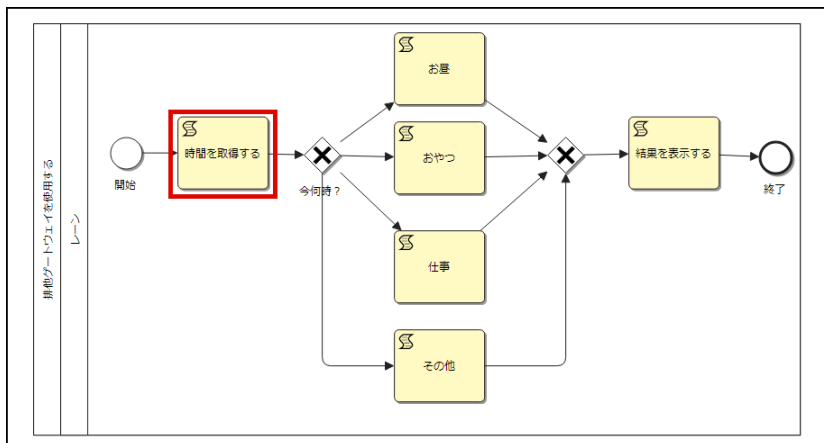
スクリプトタスクで取得した現在時刻によって、排他ゲートウェイで分岐します。

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」

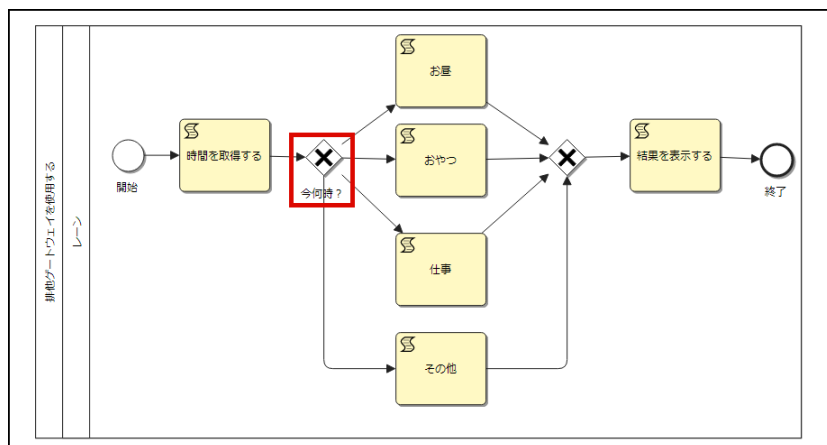
スクリプトタスク「時間を取得する」に、現在の時間を取得し変数 `hour` に格納するスクリプトを設定します。

```
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。

基本情報 | 説明 | リスナ | 表現

ID *

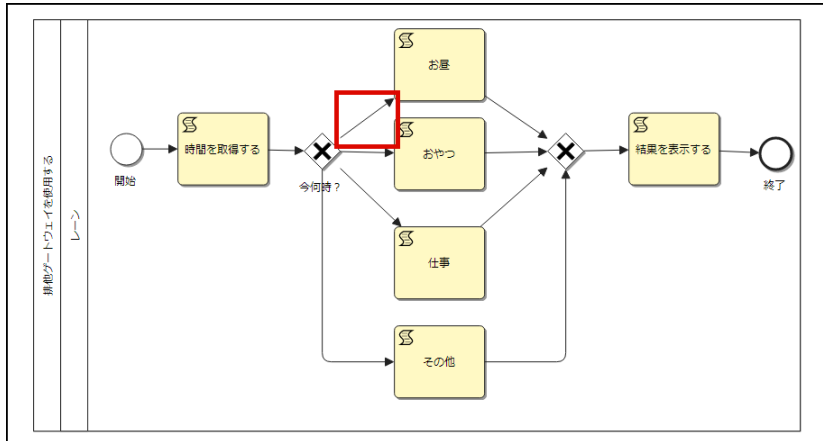
名前

今何時?

デフォルトフロー

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}
```

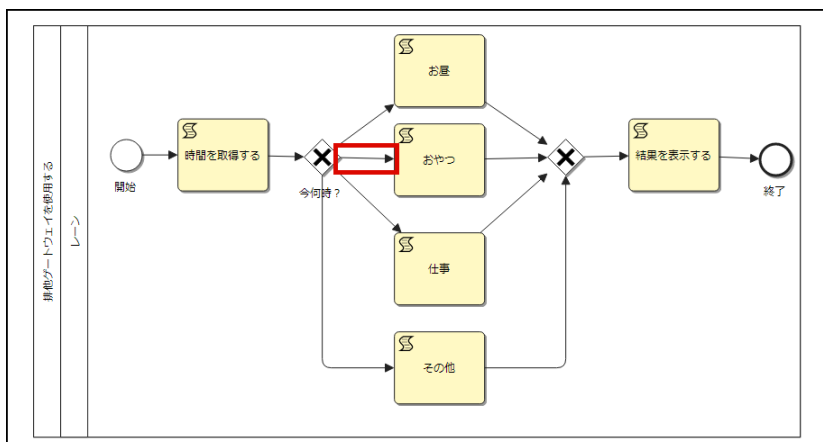
基本情報 | メインコンフィグ | 説明 | リスナ

スキップ条件

条件

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}
```

基本情報 メインコンフィグ 説明 リスナ

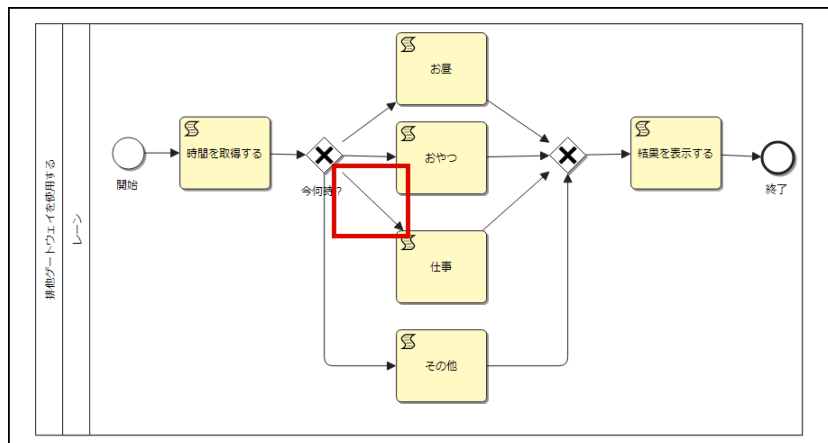
スキップ条件

条件

`$ {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}
```

基本情報 メインコンフィグ 説明 リスナ

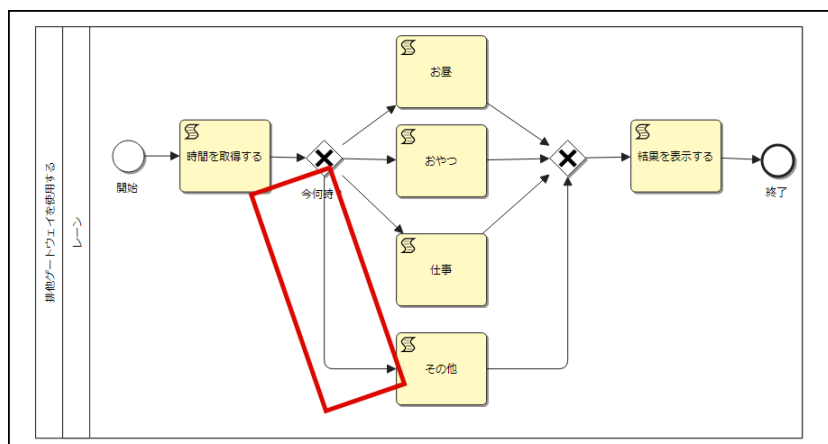
スキップ条件

条件

`${(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」

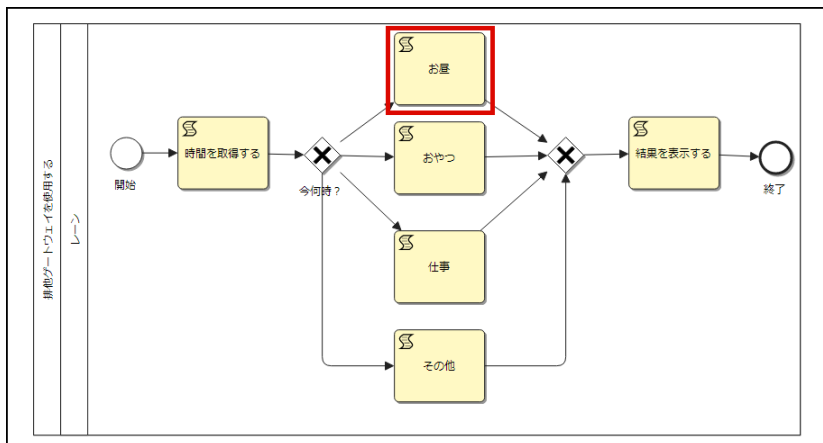
6. シーケンスフロー「sequence-flow_6」の条件を設定します。



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。

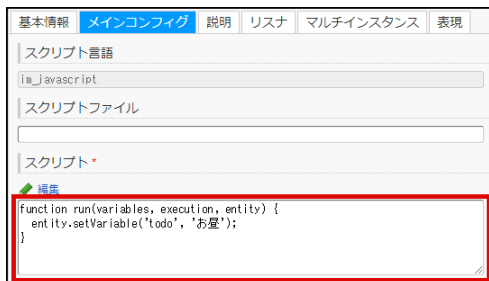
7. スクリプトタスク「お昼」のプロパティを設定します。



図：スクリプトタスク「お昼」

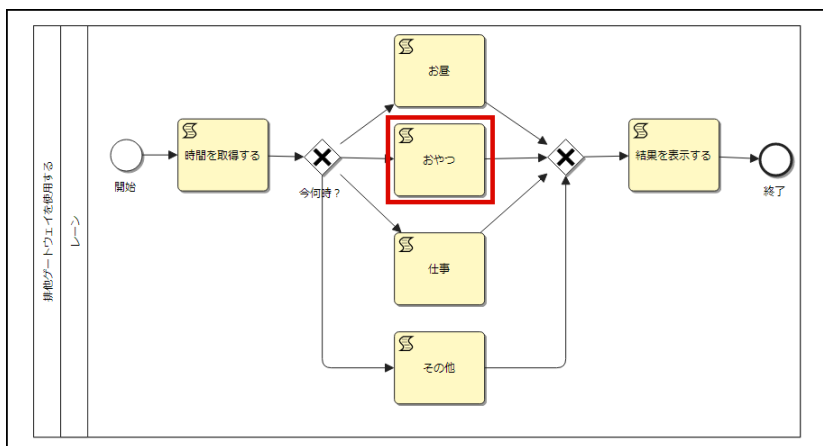
スクリプトタスク「お昼」のプロパティにて、作業名 **お昼** を変数 **todo** に格納するスクリプトを設定します。

```
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');
}
```

基本情報 | **メインコンフィグ** | 説明 | リスナ | マルチインスタンス | 表現

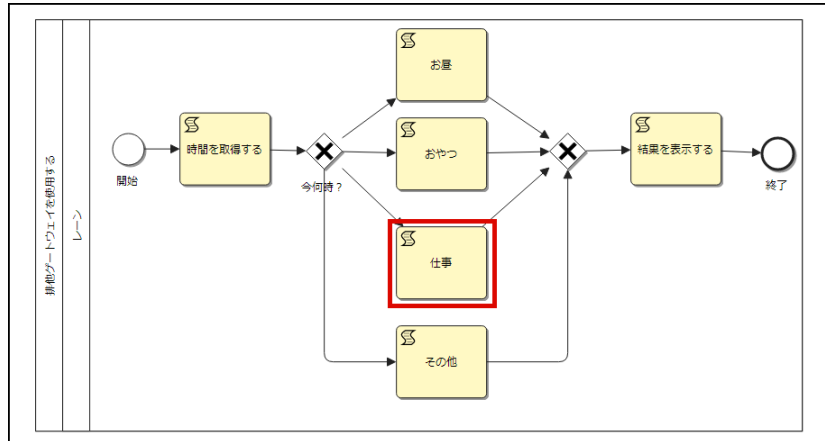
スクリプト言語:

スクリプトファイル:

スクリプト:

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');
}
```

基本情報 | **メインコンフィグ** | 説明 | リスナ | マルチインスタンス | 表現

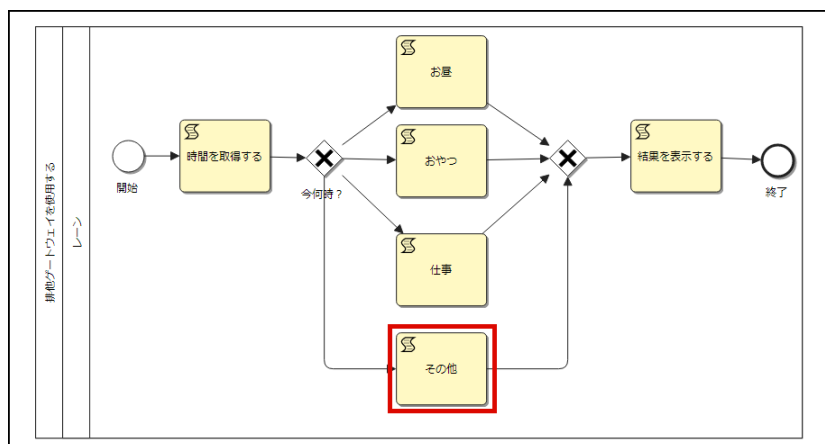
スクリプト言語:

スクリプトファイル:

スクリプト:

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」

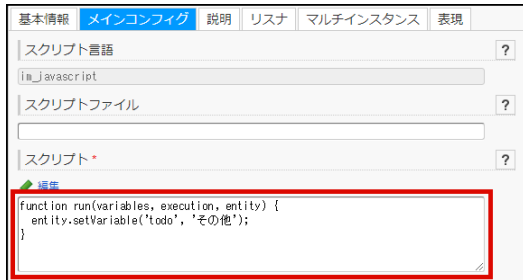
10. スクリプトタスク「その他」のプロパティを設定します。



図：スクリプトタスク「その他」

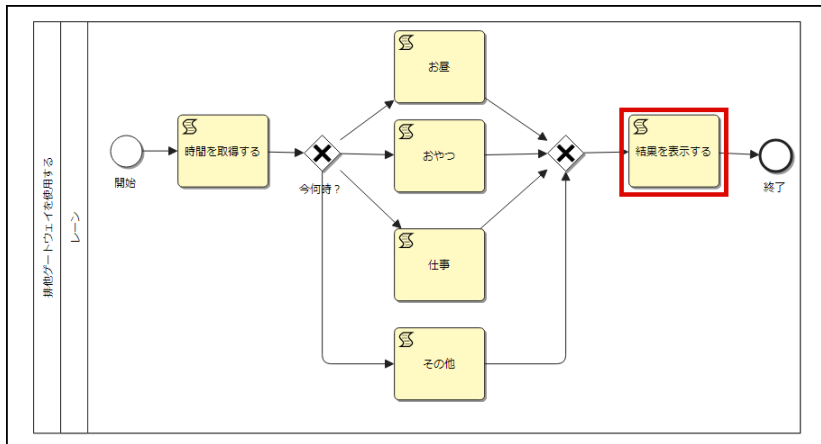
スクリプトタスク「その他」のプロパティにて、作業名 **その他** を変数 **todo** に格納するスクリプトを設定します。

```
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。

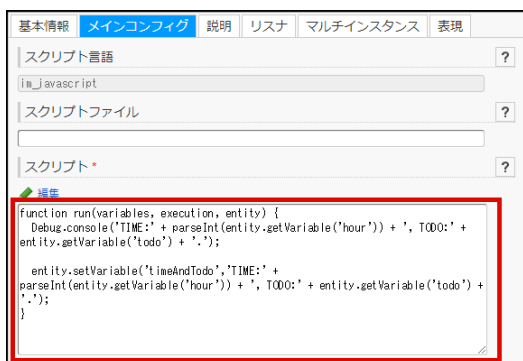


図：スクリプトタスク「結果を表示する」

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') + '.');
}
```



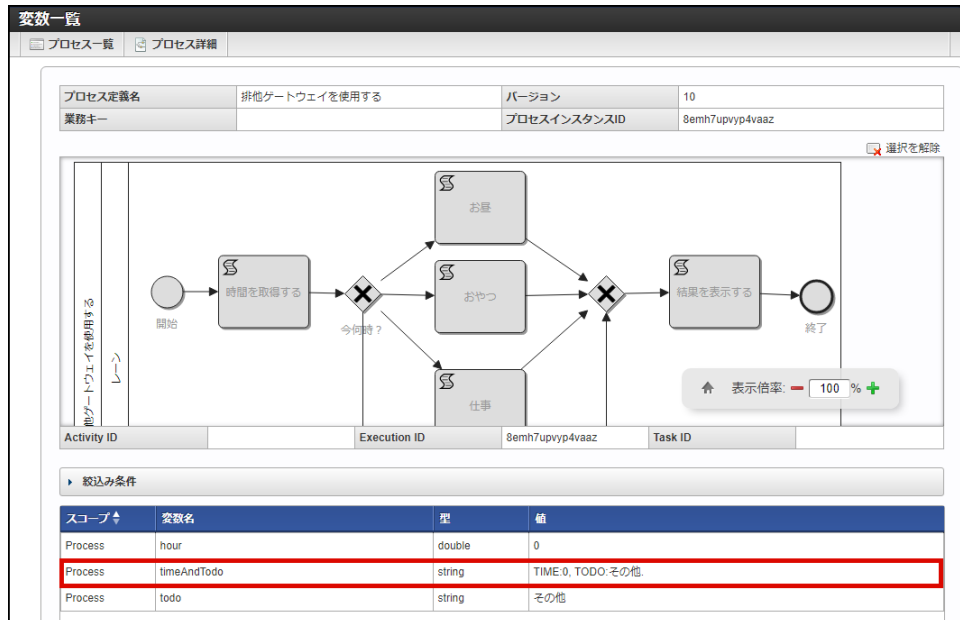
図：スクリプトタスク「結果を表示する」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

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。

```
===== } =====
/* String */
"TIME:0, TODO:その他."
```

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 プロセスデザイナー 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 プロセスデザイナー 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 プロセスデザイナー 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](#)」

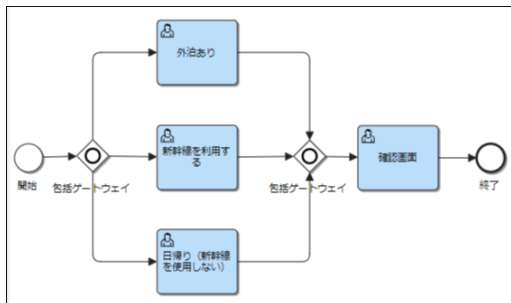
- プロセス定義を作成する
- 結果を確認する

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。

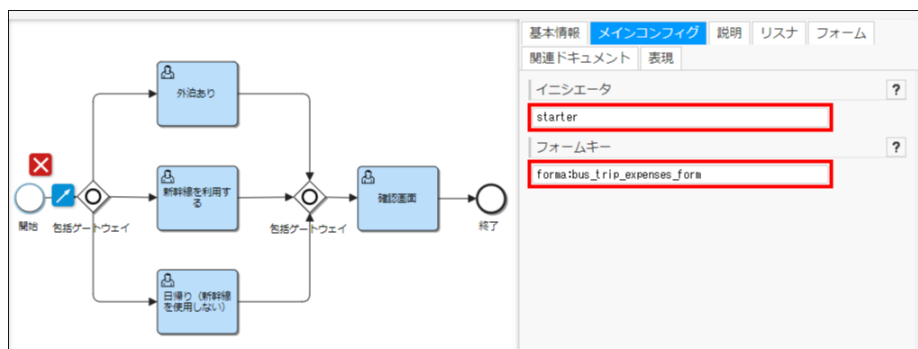
- 出張経費に「外泊費」を含めて申請する
- 出張経費に「新幹線の交通費」を含めて申請する
- 上記二つが当てはまらない申請をする

「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ーが「確認画面」のタスクで確認します。



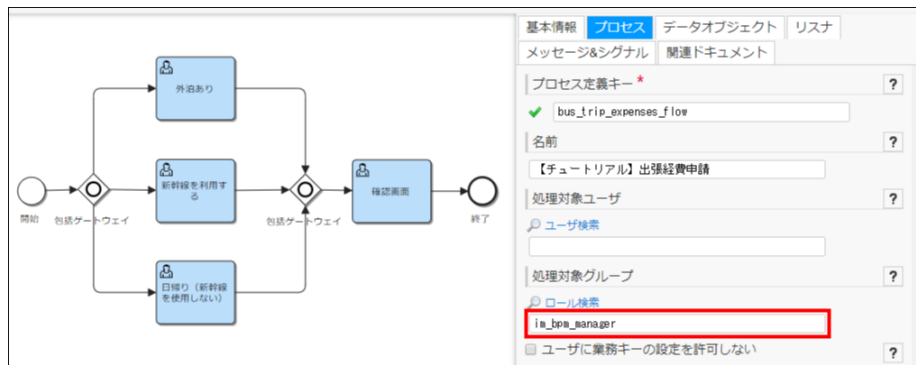
図：完成イメージ

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」のアプリケーション画面を設定します。
申請したユーザーを「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：formabus_trip_expenses_form



図：「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

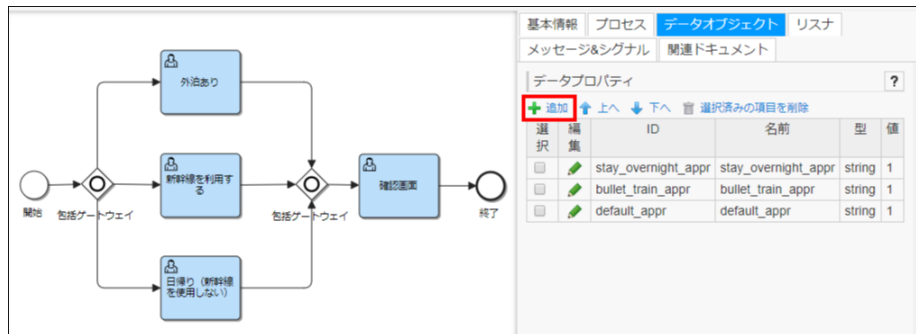
3. プロセス全体に対する設定を行います。
「開始イベント」ではない場所をクリックし、「プロパティ」をプロセス全体に切り替えます。
4. 「プロセス」タブの「処理対象グループ」に「im_bpm_manager」を設定します。



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。



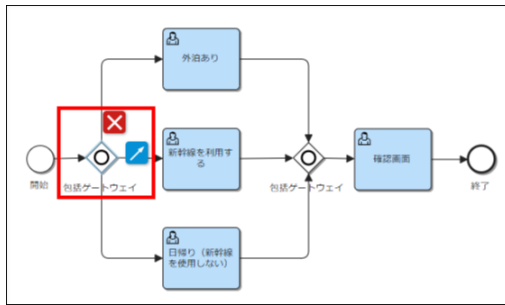
図：プロセス全体 - 「プロパティ」 - 「データオブジェクト」

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'))}`

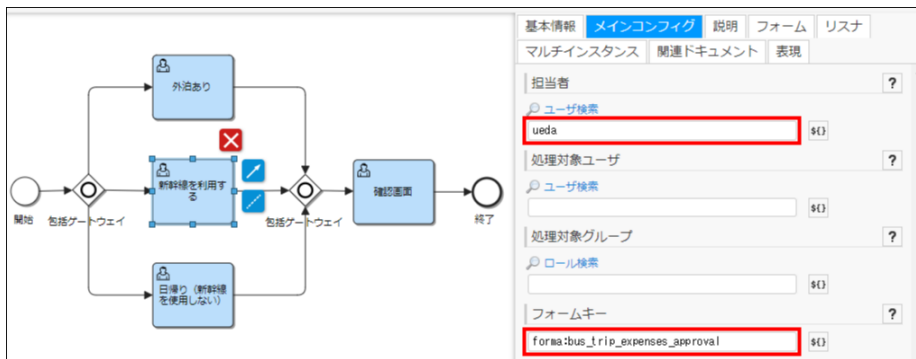
図：「タスクリスナ」

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。

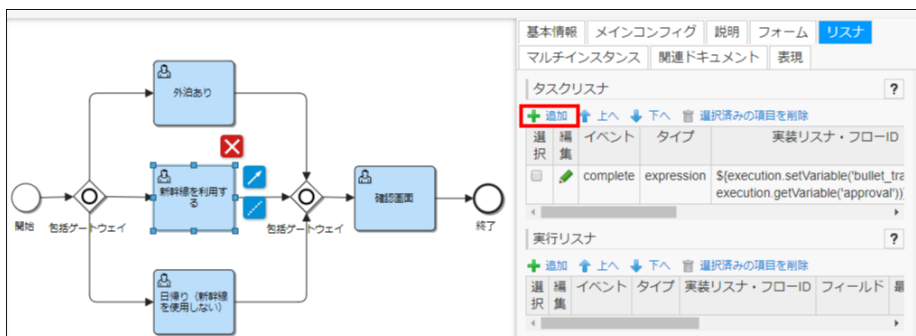
- 担当者 : ueda
- フォームキー : form:bus_trip_expenses_approval



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」に書き込みます。

「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')))`

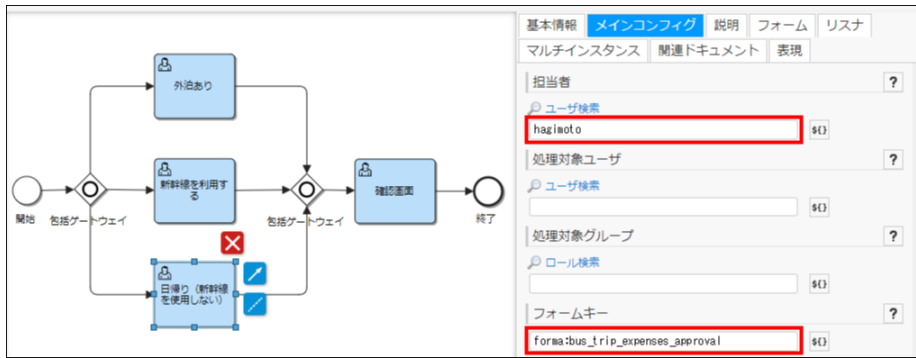


図：「タスクリスナ」

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。

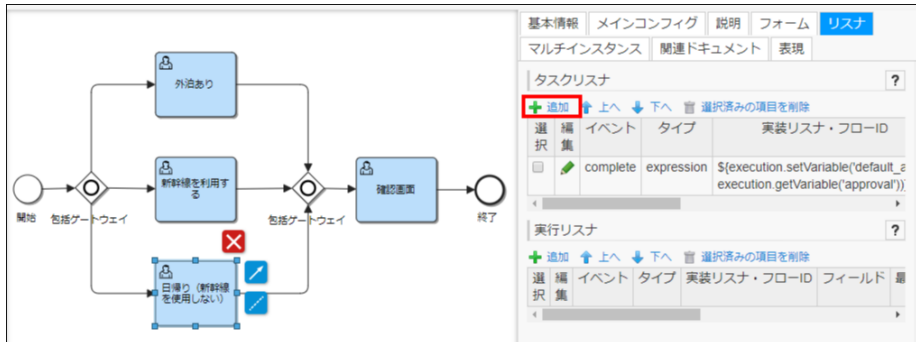
17. 「ユーザタスク」を選択し、「メインコンフィグ」タブで以下のとおりに項目を設定します。

- 担当者 : hagimoto
- フォームキー : form:bus_trip_expenses_approval



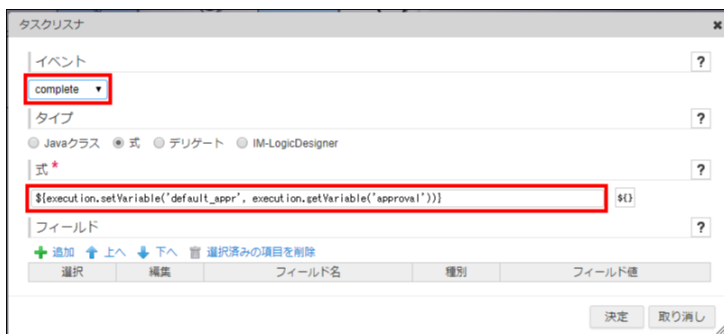
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

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。



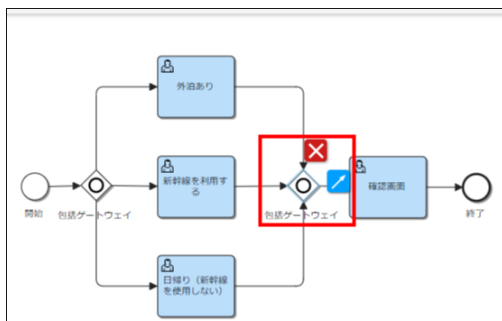
図：「ユーザタスク」 - 「プロパティ」 - 「リスナ」

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'))}`



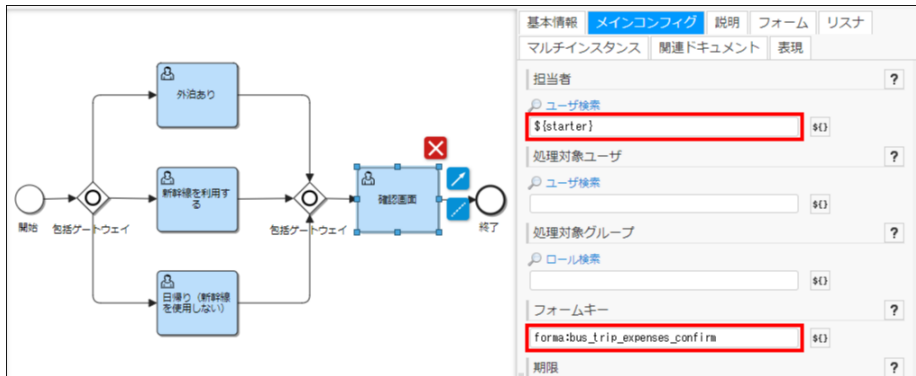
図：「タスクリスナ」

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

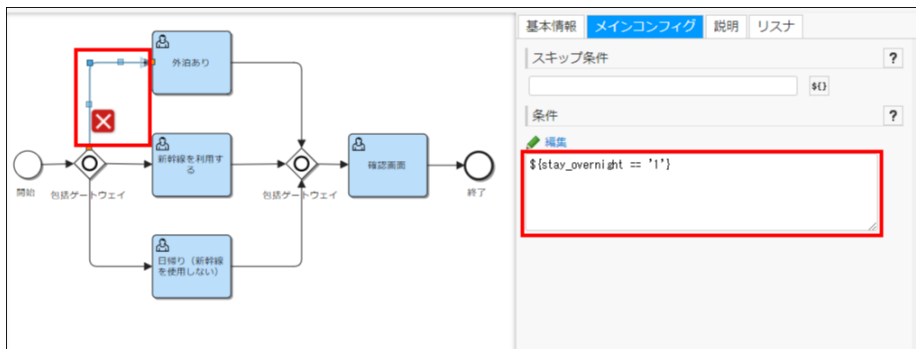


図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

23. 終了イベントを設置します。

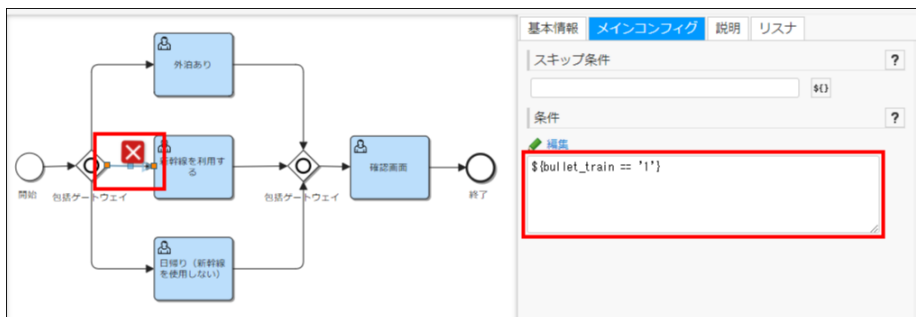
24. 「申請画面」で選択された情報をもとに、フローを進める条件を設定します。

「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。



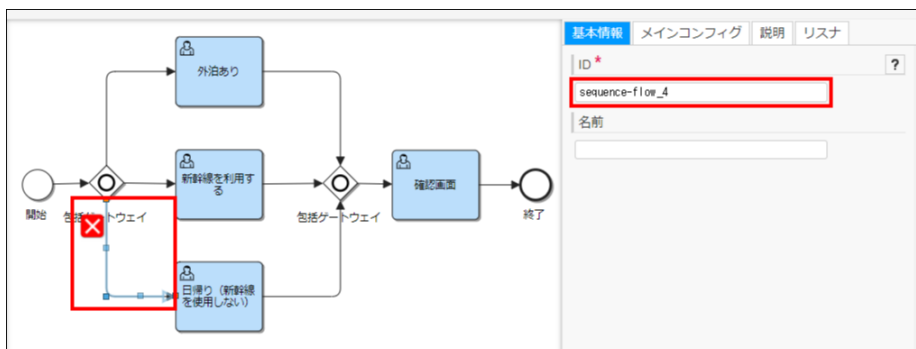
図：「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」

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。



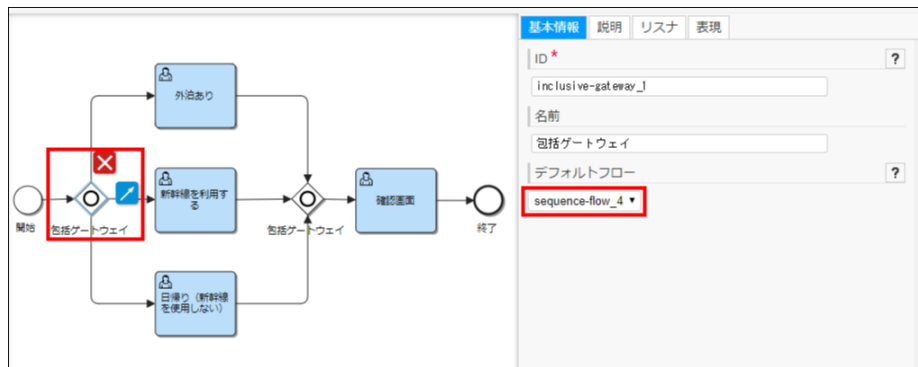
図：「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」

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」

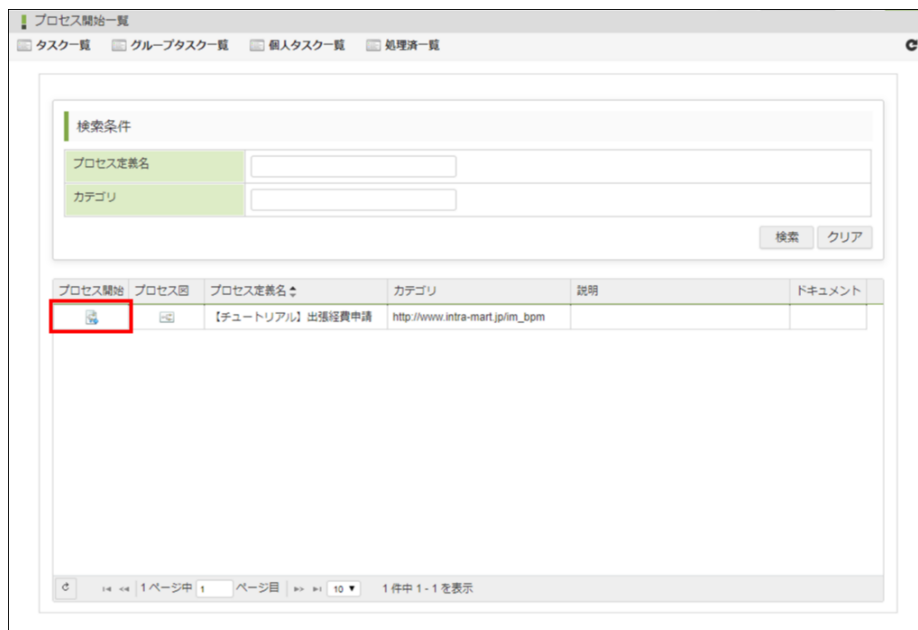
結果を確認する

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」

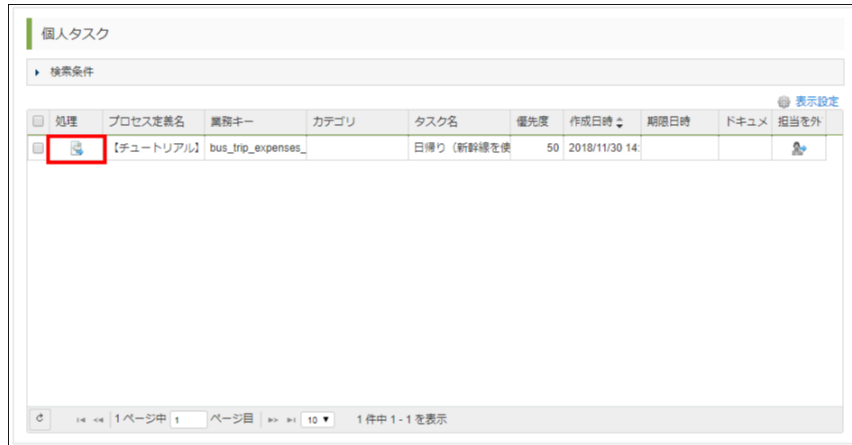
3. 「IM-FormaDesigner」で作成したFormaアプリケーション、「申請画面」が表示されます。
4. 今回は包括ゲートウェイの「デフォルトフロー」へ進行したいので、「外泊あり」と「新幹線を利用する」のチェックボックスは入れません。適当な値を入力し、「申請」ボタンをクリックします。

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「



図：「タスク一覧」 - 「個人タスク」

7. 「IM-FormaDesigner」で作成したFormaアプリケーション、「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。

出張経費承認画面

以下の出張経費精算の申請がありました。
承認しますか？

☐ 外泊あり

☐ 新幹線を利用する

目的地

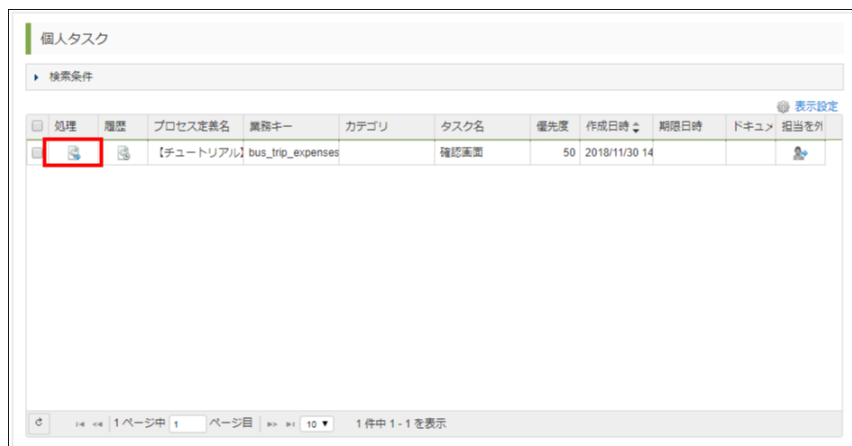
使用金額

上記の申請を ☒ 承認する ☐ 否認する

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「



図：「タスク一覧」 - 「個人タスク」

11. 「IM-FormaDesigner」で作成したFormaアプリケーション、「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。

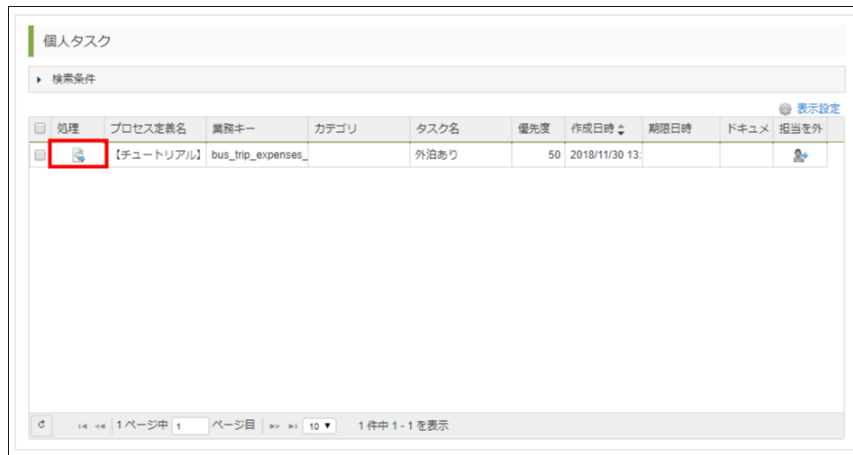
図：「プロセス開始一覧」

3. 「IM-FormaDesigner」で作成したFormaアプリケーション、「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「



図：「タスク一覧」 - 「個人タスク」

8. 「IM-FormaDesigner」で作成したFormaアプリケーション、「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。

出張経費承認画面

以下の出張経費精算の申請がありました。
承認しますか？

☒ 外泊あり

☒ 新幹線を利用する

目的地

使用金額

上記の申請を ☒ 承認する ☒ 否認する

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「新幹線を利用する」の「



図：「タスク一覧」 - 「個人タスク」

13. 「IM-FormaDesigner」で作成したFormaアプリケーション、「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。

出張経費承認画面

以下の出張経費精算の申請がありました。
承認しますか？

☒ 外泊あり
☒ 新幹線を利用する

目的地

使用金額

上記の申請を ☒ 承認する ☐ 否認する

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。

個人タスク

検索条件

処理	種類	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当を外
		【チュートリアル】bus_trip_expenses			確認画面	50	2018/11/30 14			

1件中 1 - 1 を表示

図：「タスク一覧」 - 「個人タスク」

17. 「IM-FormaDesigner」で作成したFormaアプリケーション、「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。

出張経費申請が否認されました

以下の内容の申請が否認されました。

☒ 外泊あり
☒ 新幹線を利用する

目的地

使用金額

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

i 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 プロセスデザイナー 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 プロセスデザイナー 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 プロセスデザイナー 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 プロセスデザイナー 操作ガイド」 - 「シグナルキャッチイベント」

i 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 プロセスデザイナー 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。

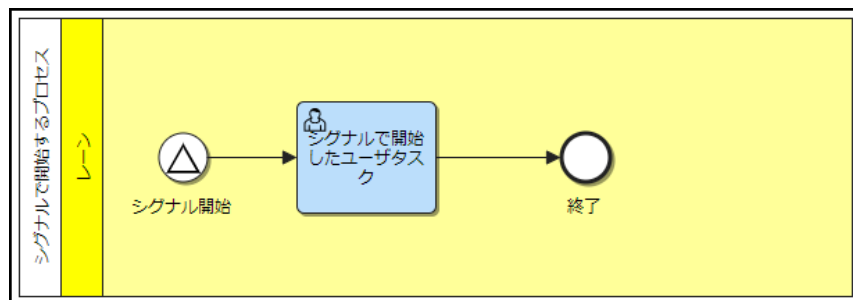
i コラム

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

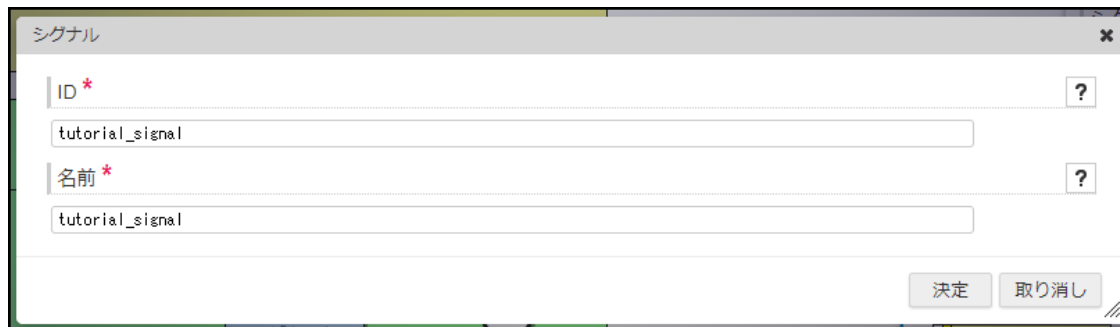
1. 「シグナル」を登録します。
 1. キャンパスの空白部分をクリックし、「メッセージ&シグナル」タブにある「シグナル」の「追加」リンクをクリックします。



図：「プロセス」 - 「プロパティ」 - 「メッセージ&シグナル」

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」

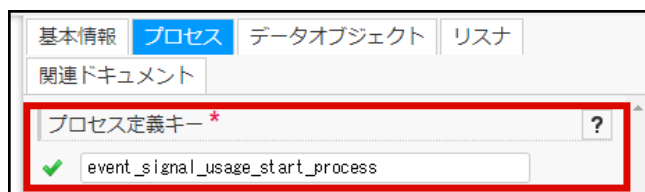
2. 「シグナル開始イベント」を配置します。
3. 「シグナル開始イベント」の「メインコンフィグ」タブから、「参照シグナル」のプルダウンリストで「tutorial_signal」を選択します。



図：「シグナル開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

4. キャンパスの空白部分、または、プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロパティ」の「プロセス」タブから、項目を以下のように設定してください。

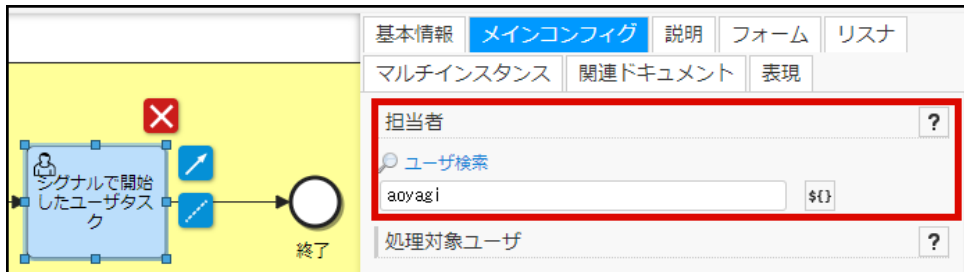
- プロセス定義キー : event_signal_usage_start_process



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。

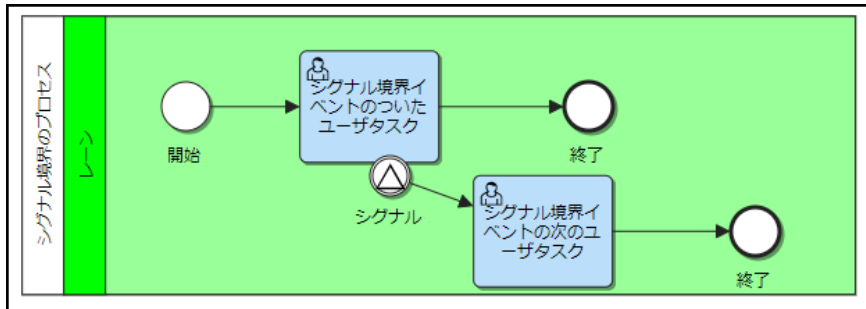
- 担当者 : aoyagi



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. キャンパスの空白部分、または、プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロパティ」の「プロセス」タブから、項目を以下のように設定してください。
 - 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」

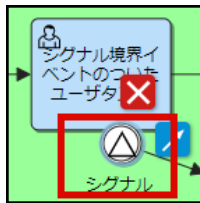
4. 「ユーザタスク」を配置します。

- 担当者：aoyagi



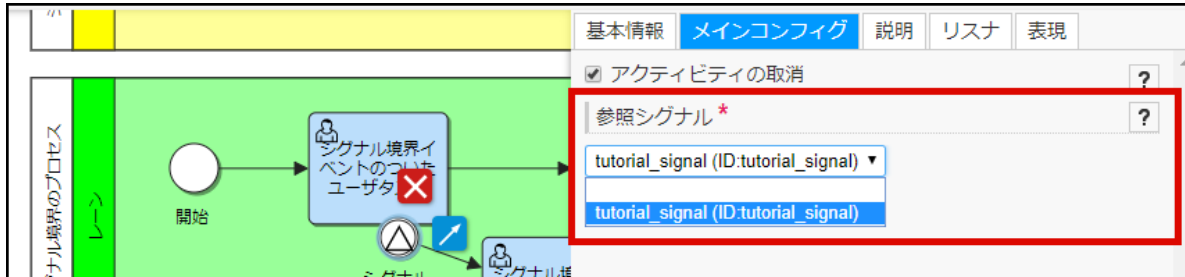
図：「ユーザタスク」

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」の「メインコンフィグ」タブから、「参照シグナル」のプルダウンリストで「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」

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。

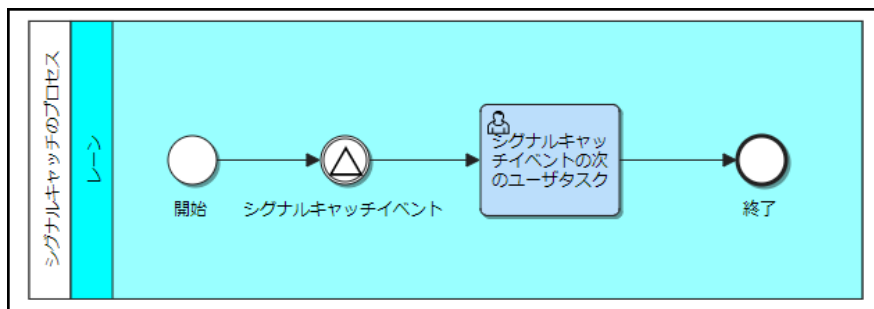
- 担当者：aoyagi



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. キャンパスの空白部分、または、プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロパティ」の「プロセス」タブから、項目を以下のように設定してください。

- プロセス定義キー：event_signal_usage_intermediate_process
- 処理対象ユーザ：aoyagi

基本情報 プロセス データオブジェクト リスナ

関連ドキュメント

プロセス定義キー * ?

✓ event_signal_usage_intermediate_process

名前 ?

シグナルキャッチのプロセス

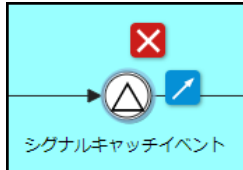
処理対象ユーザ ?

ユーザ検索

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」の「メインコンフィグ」タブから、「参照シグナル」のプルダウンリストで「tutorial_signal」を選択します。

基本情報 メインコンフィグ 説明 リスナ 表現

参照シグナル * ?

tutorial_signal (ID:tutorial_signal) ▼

tutorial_signal (ID:tutorial_signal)

シグナルキャッチイベント

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」

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

基本情報 メインコンフィグ 説明 フォーム リスナ

マルチインスタンス 関連ドキュメント 表現

担当者 ?

ユーザ検索

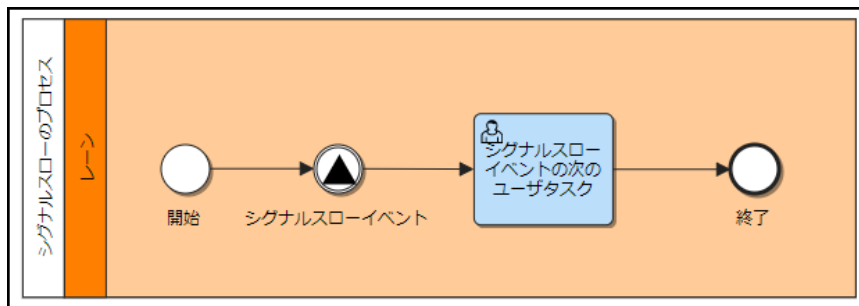
aoyagi

シグナルキャッチイベントの次のユーザタスク

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. キャンパスの空白部分、または、プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロパティ」の「プロセス」タブから、項目を以下のように設定してください。
 - プロセス定義キー：event_signal_usage_throw_process
 - 処理対象ユーザ：aoyagi

基本情報 プロセス データオブジェクト リスナ

関連ドキュメント

プロセス定義キー * ?

✓ event_signal_usage_throw_process

名前 ?

シグナルスローのプロセス

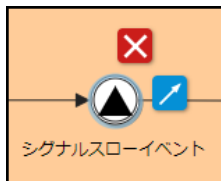
処理対象ユーザ ?

ユーザ検索

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」の「メインコンフィグ」タブから、「参照シグナル」のプルダウンリストで「tutorial_signal」を選択します。

基本情報 メインコンフィグ 説明 リスナ 表現

シグナルスローイベント

参照シグナル * ?

tutorial_signal (ID:tutorial_signal) ▼

tutorial_signal (ID:tutorial_signal)

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」

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

基本情報 メインコンフィグ 説明 フォーム リスナ

マルチインスタンス 関連ドキュメント 表現

シグナルスローイベントの次のユーザタスク

担当者 ?

ユーザ検索

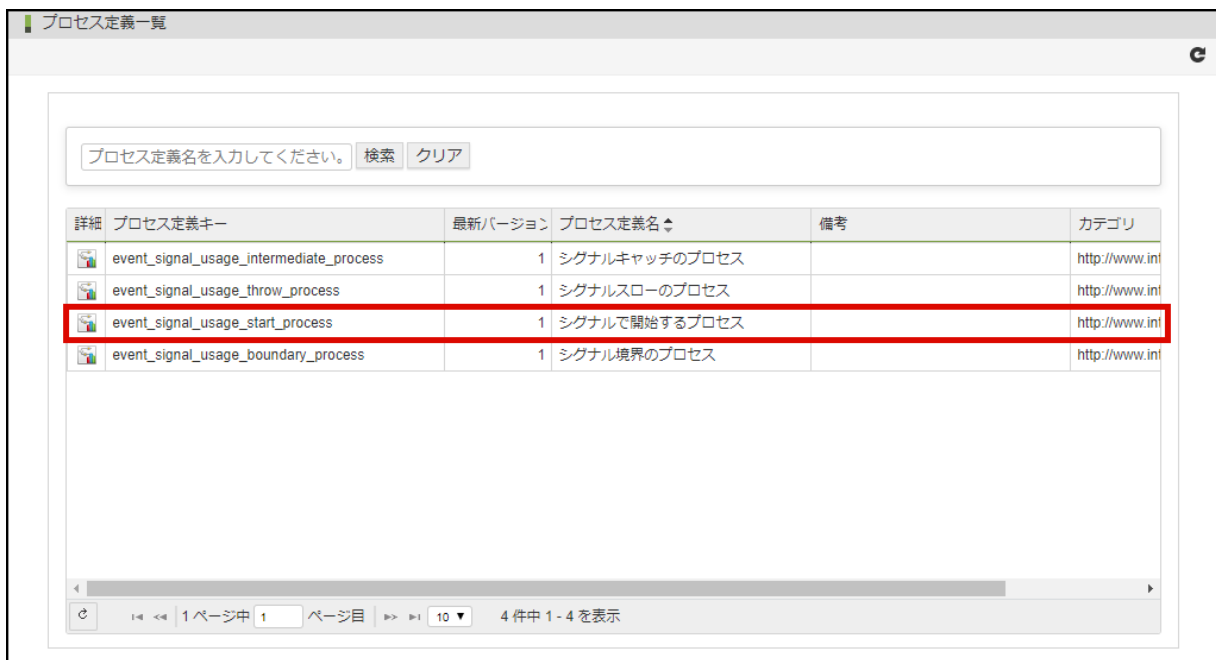
aoyagi \${{}}

図：「ユーザタスク」

実行結果を確認する

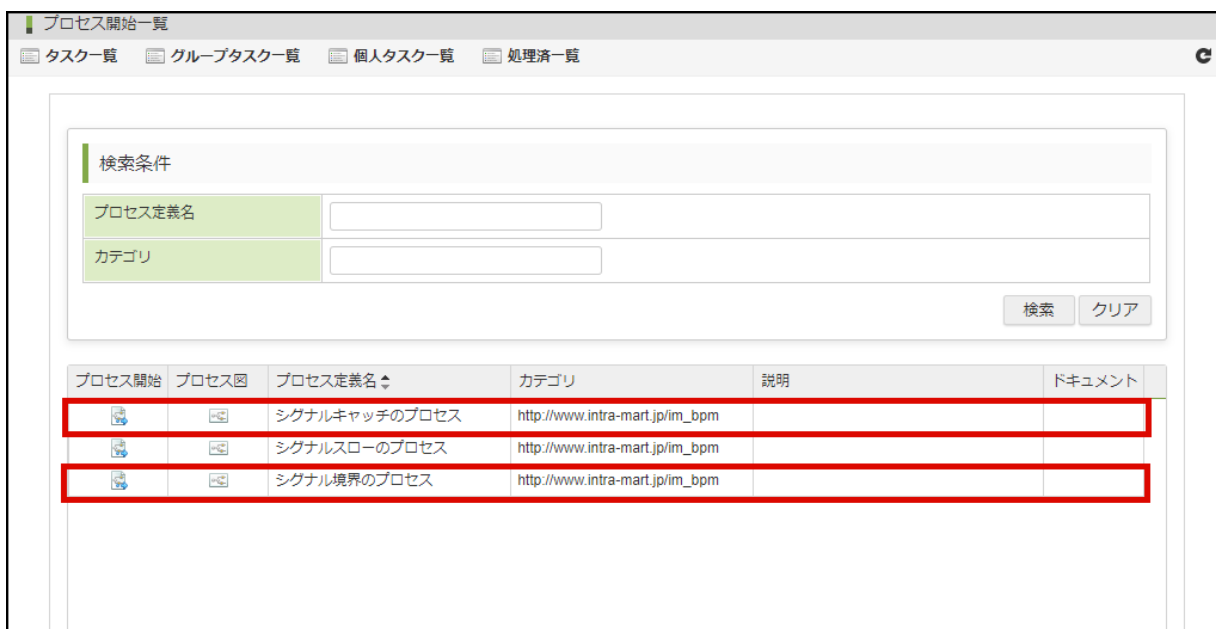
このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

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。



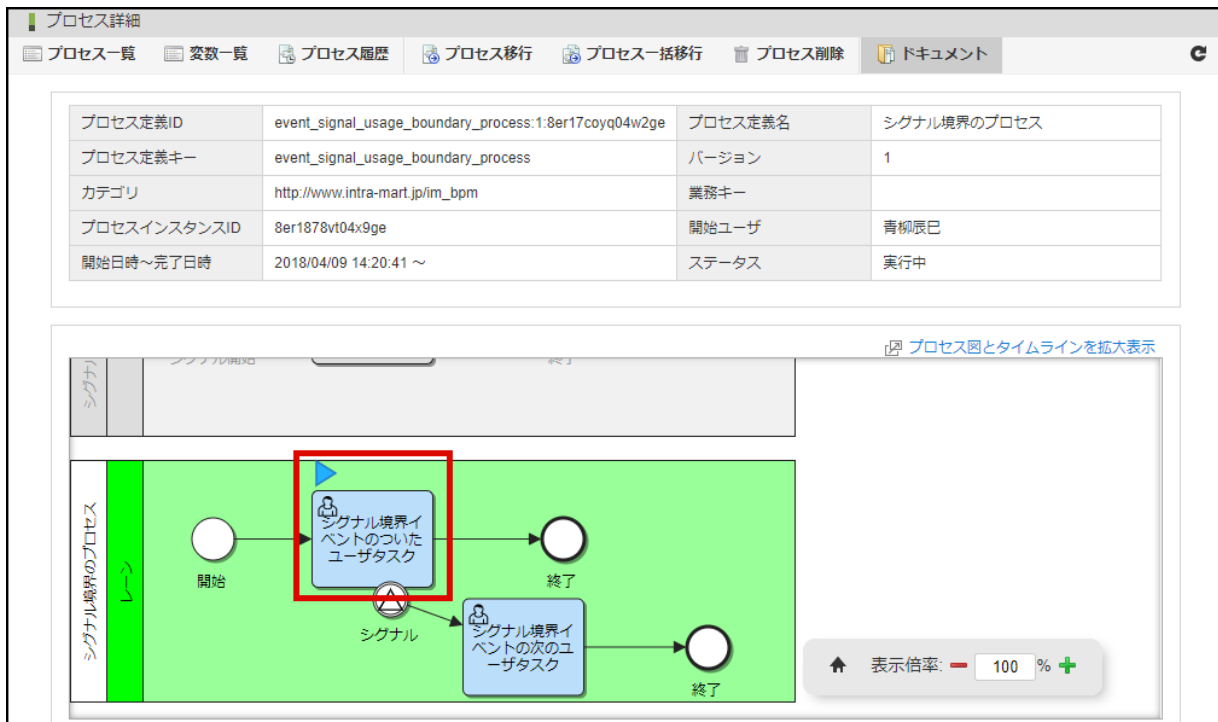
図：プロセス定義一覧 event_signal_usage_start_process

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。



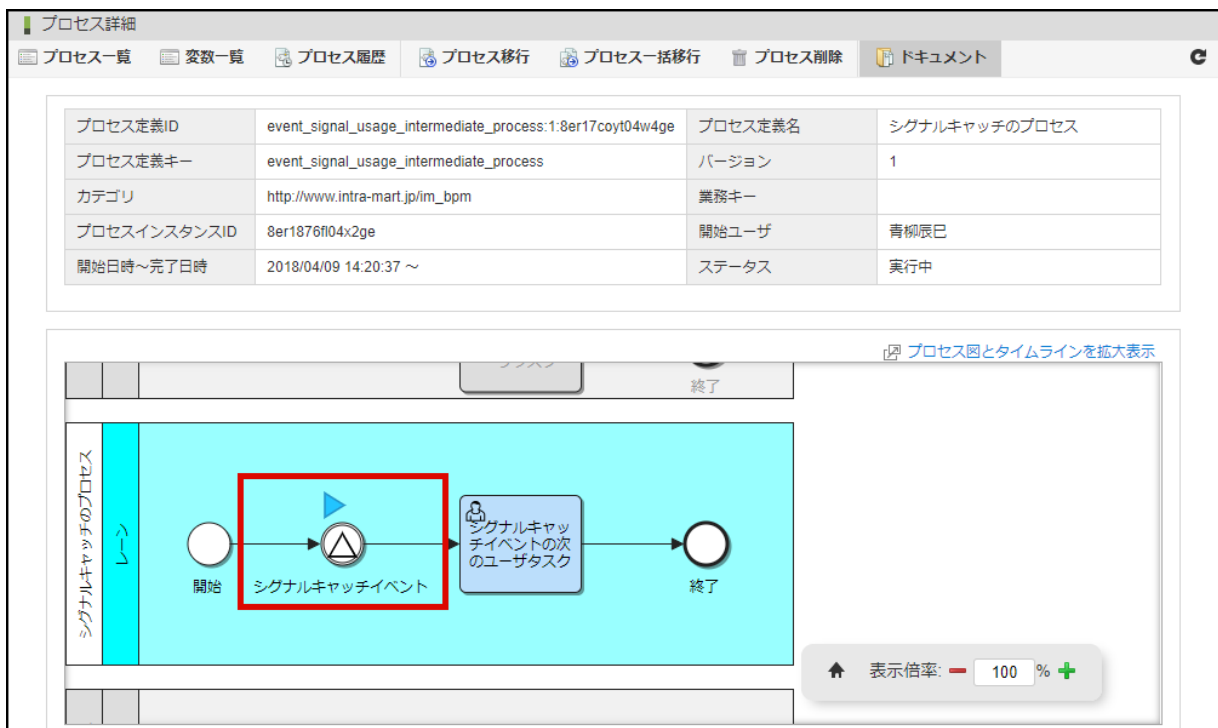
図：プロセス開始一覧

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。



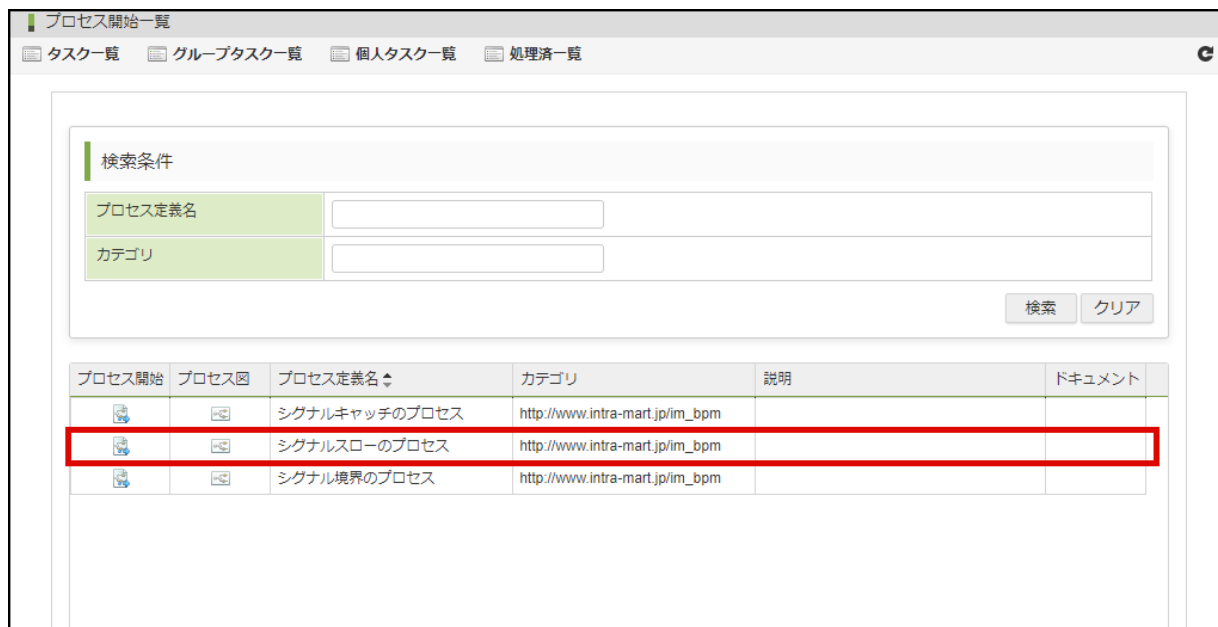
図：プロセス詳細 event_signal_usage_boundary_process

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：プロセス開始一覧 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。

i 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。

i 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。

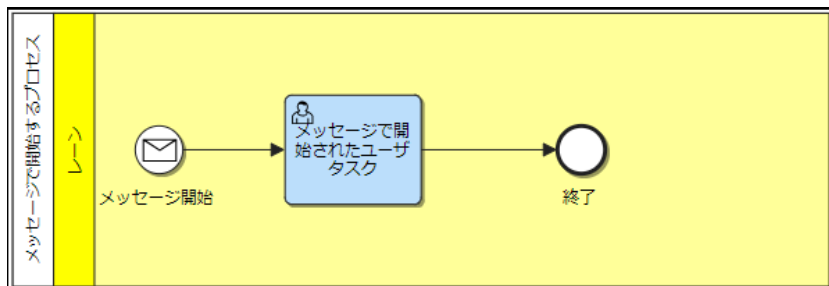
i コラム

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. キャンパスの空白部分をクリックし、「プロパティ」をプロセス全体切り替えます。
「プロパティ」の「メッセージ&シグナル」タブから、「メッセージ」の「追加」リンクをクリックします。

図：プロセス全体 - 「プロパティ」 - 「メッセージ&シグナル」

2. 「メッセージ」ダイアログで、項目を以下のとおりに設定します。
 - 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. 参照メッセージを設定します。
「メッセージ開始イベント」の「メインコンフィグ」タブから、「参照メッセージ」のプルダウンリストで「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」

4. プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロセス」タブから、「プロセス定義キー」に「event_message_usage_start_process」と入力します。

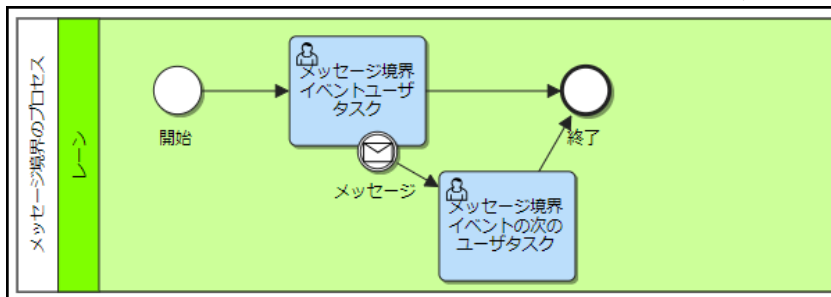
図：プロセス全体 - 「プロパティ」 - 「プロセス」

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」

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。

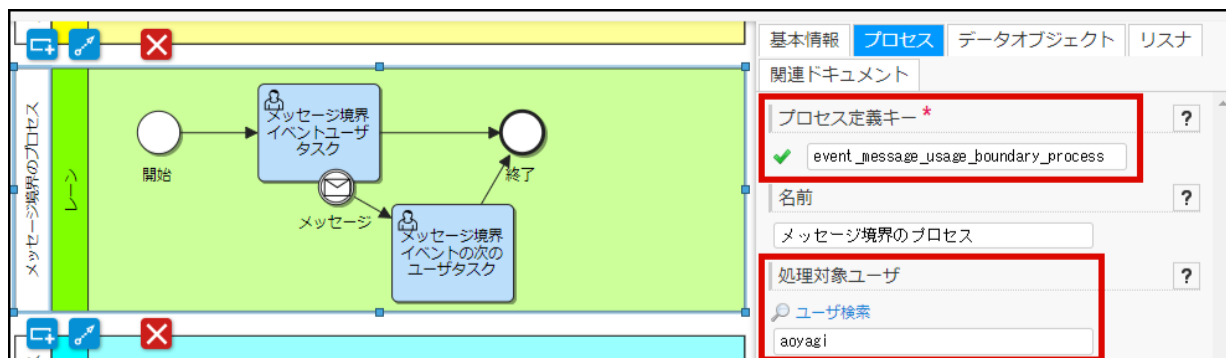


図：完成イメージ

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。

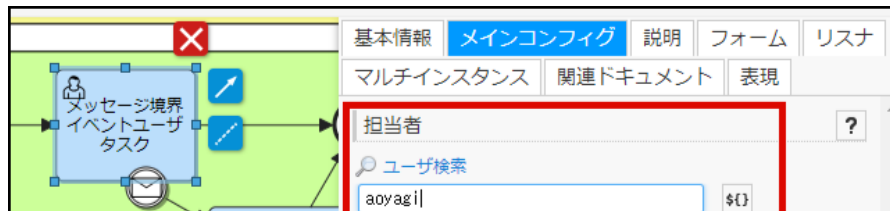
登録方法は「[メッセージを受信することにより開始するプロセス定義を作成する](#)」を参照してください。

2. 「開始イベント」を配置します。
3. プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロセス」タブから、以下のように項目を設定します。
 - プロセス定義キー：event_message_usage_boundary_process
 - 処理対象ユーザ：aoyagi



図：「プール」 - 「プロパティ」 - 「プロセス」

4. 「ユーザータスク」を配置します。
 - 担当者：aoyagi



図：「ユーザータスク」

5. 「メッセージ境界イベント」を「ユーザータスク」に配置します。



図：「メッセージ境界イベント」

6. 参照メッセージを設定します。
「メッセージ境界イベント」の「メインコンフィグ」タブから、「参照メッセージ」のプルダウンリストで「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」

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。

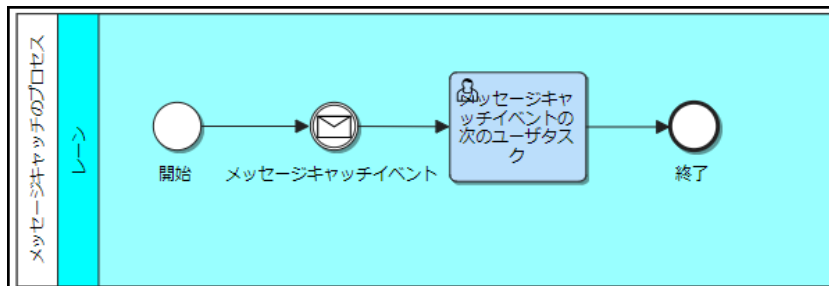
- 担当者：aoyagi



図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

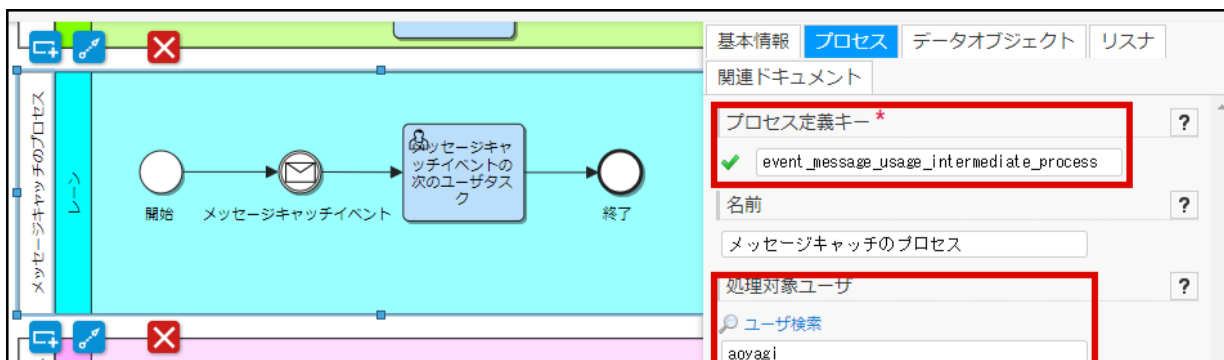
メッセージを受信することにより先に進むプロセス定義を作成する

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。



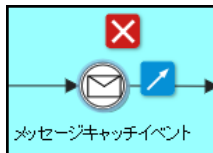
図：完成イメージ

- 「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。
- 「開始イベント」を配置します。
- プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロセス」タブから、以下のように項目を設定します。
 - プロセス定義キー：event_message_usage_intermediate_process
 - 処理対象ユーザ：aoyagi



図：「プール」 - 「プロパティ」 - 「プロセス」

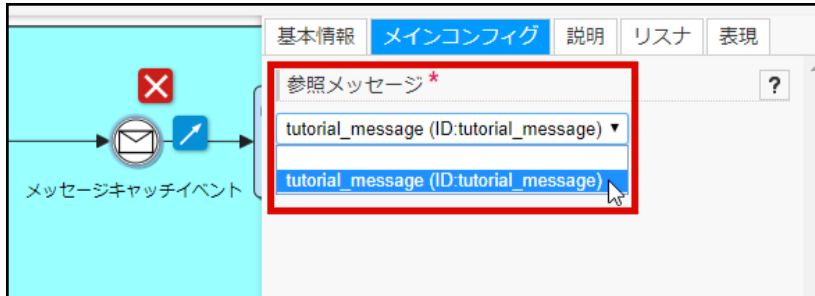
- 「メッセージキャッチイベント」を配置します。



図：「メッセージキャッチイベント」

5. 参照メッセージを設定します。

「メッセージキャッチイベント」の「メインコンフィグ」タブから、「参照メッセージ」のプルダウンリストで「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」

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」

メッセージを送信するプロセス定義を作成する

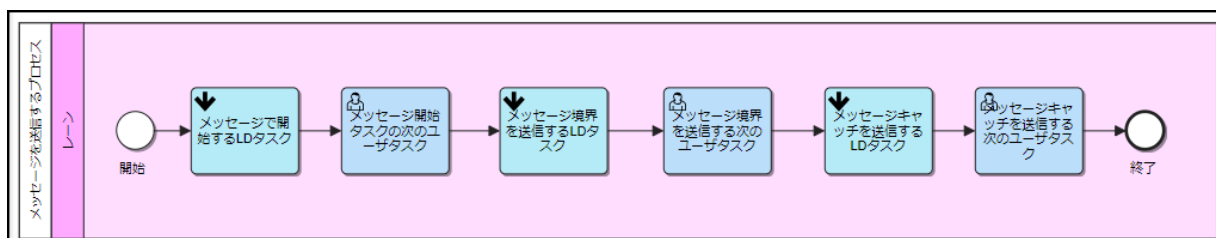
メッセージを送信するプロセス定義を作成します。

「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。

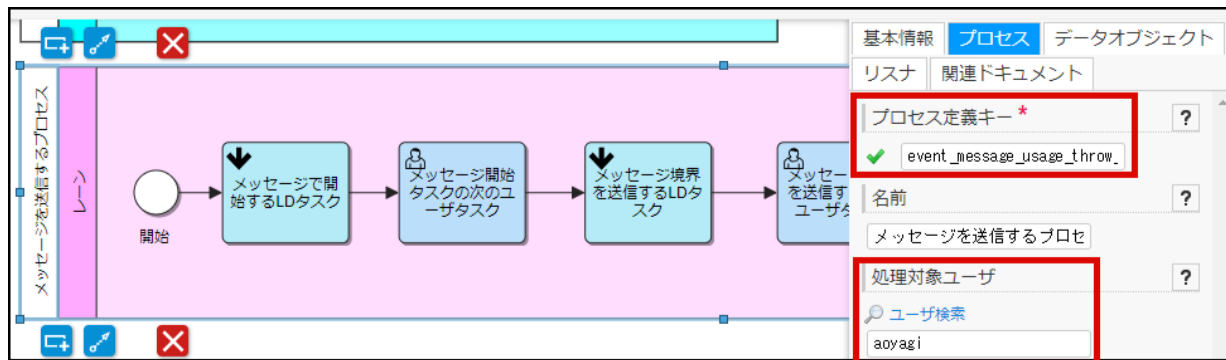
送信されたメッセージは、「メッセージで開始するプロセス」「メッセージ境界のプロセス」「メッセージキャッチのプロセス」がそれぞれ受信し、結果としてプロセスが進行します。

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

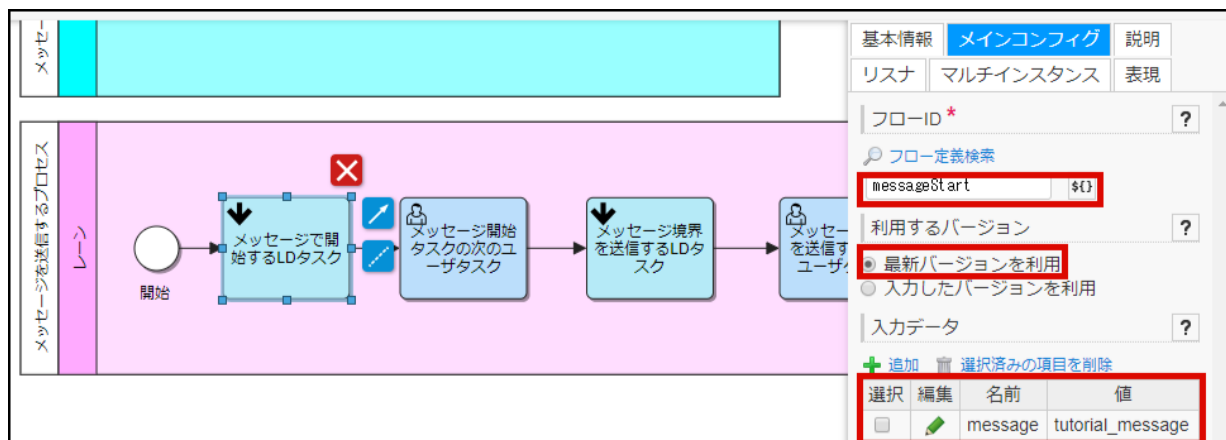
- 「開始イベント」を配置します。
- プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロセス」タブから、以下のように項目を設定します。
 - プロセス定義キー：event_message_usage_throw_process
 - 処理対象ユーザ：aoyagi



図：プロセス全体 - 「プロパティ」 - 「プロセス」

3. 「IM-LogicDesignerタスク」の「メインコンフィグ」で、以下のように項目を設定します。

- 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



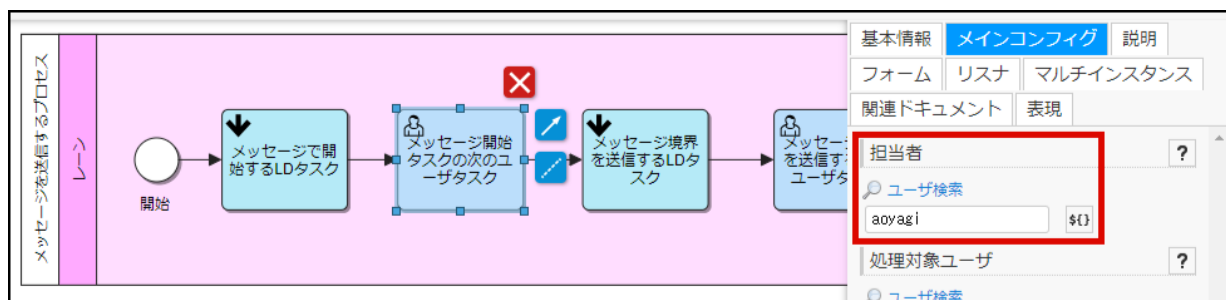
図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

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」の「メインコンフィグ」タブで、以下のように項目を設定します。

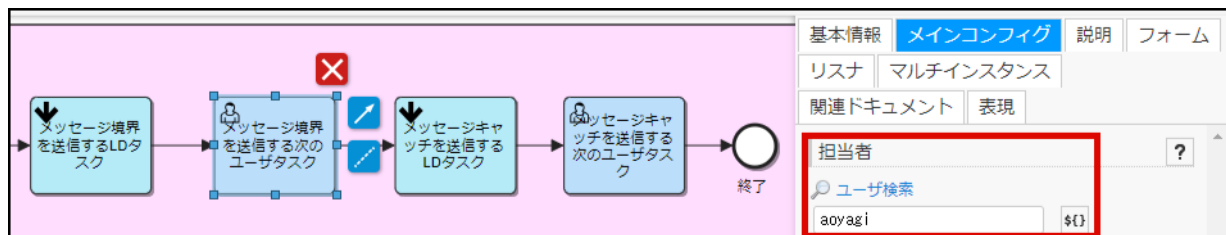
- フローID : messageSend
- 利用するバージョン : 最新バージョンを利用
- 入力データ1
 - 名前 : message
 - 値 : tutorial_message
- 入力データ2
 - 名前 : processDefinitionKey
 - 値 : event_message_usage_boundary_process



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」の「メインコンフィグ」タブで、以下のように項目を設定します。

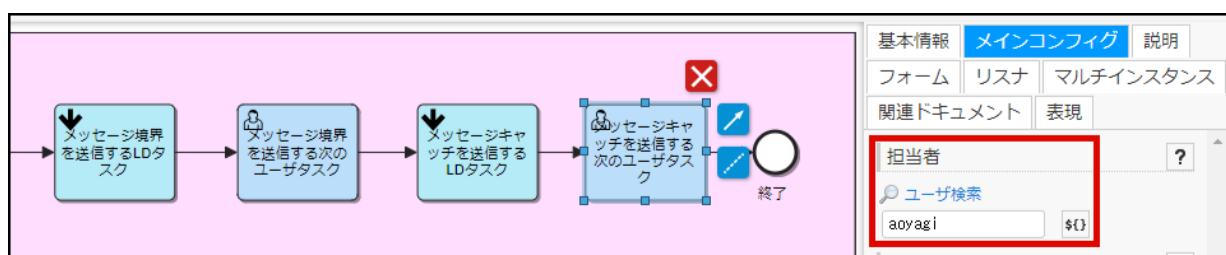
- フローID：messageSend
- 利用するバージョン：最新バージョンを利用
- 入力データ1
 - 名前：message
 - 値：tutorial_message
- 入力データ2:
 - 名前：processDefinitionKey
 - 値：event_message_usage_intermediate_process



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。



詳細	プロセス定義キー	最新バージョン	プロセス定義名	備考	カテゴリ	アクティブ
	event_message_usage_boundary_process	1	メッセージ境界のプロセス		http://www.intra-mart.jp/im_bpm	
	event_message_usage_intermediate_process	1	メッセージキャッチのプロセス		http://www.intra-mart.jp/im_bpm	
	event_message_usage_throw_process	1	メッセージを送信するプロセス		http://www.intra-mart.jp/im_bpm	
	event_message_usage_start_process	1	メッセージで開始するプロセス		http://www.intra-mart.jp/im_bpm	

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。



プロセス開始	プロセス図	プロセス定義名	カテゴリ	説明	ドキュメント
		メッセージを送信するプロセス	http://www.intra-mart.jp/im_bpm		
		メッセージキャッチのプロセス	http://www.intra-mart.jp/im_bpm		
		メッセージ境界のプロセス	http://www.intra-mart.jp/im_bpm		

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。

プロセス詳細

プロセス一覧 変数一覧 プロセス履歴 プロセス移行 プロセス一括移行 プロセス削除 ドキュメント

プロセス定義ID	event_message_usage_boundary_process:1:8esay182do26yrq	プロセス定義名	メッセージ境界のプロセス
プロセス定義キー	event_message_usage_boundary_process	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8esayf0f2o2jhrq	開始ユーザ	青柳辰巳
開始日時～完了日時	2018/05/11 14:24:11 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

表示倍率: 100 %

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。

プロセス詳細

プロセス一覧 変数一覧 プロセス履歴 プロセス移行 プロセス一括移行 プロセス削除 ドキュメント

プロセス定義ID	event_message_usage_intermediate_process:1:8esay182eo270rq	プロセス定義名	メッセージキャッチのプロセス
プロセス定義キー	event_message_usage_intermediate_process	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8esayexpeo2jarq	開始ユーザ	青柳辰巳
開始日時～完了日時	2018/05/11 14:24:07 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

表示倍率: 100 %

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。



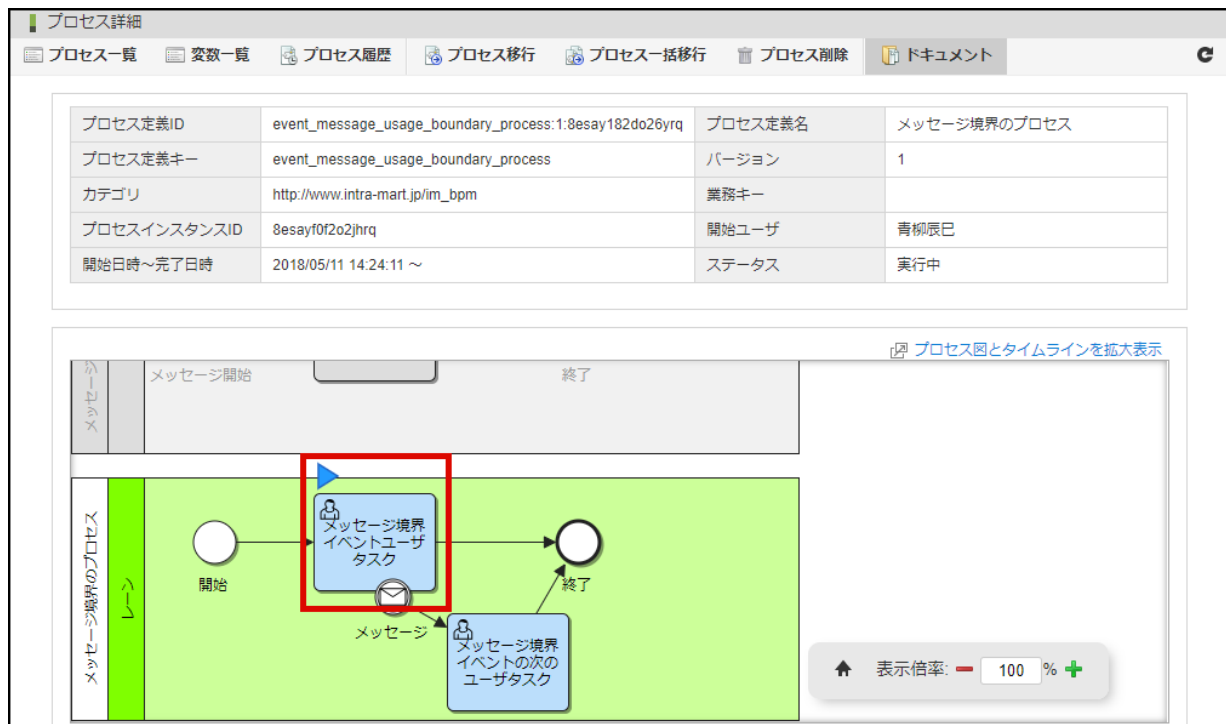
図：プロセス開始 event_message_usage_throw_process

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

プロセス詳細

プロセス一覧 変数一覧 プロセス履歴 プロセス移行 プロセス一括移行 プロセス削除 ドキュメント

プロセス定義ID	event_message_usage_intermediate_process:1:8esay182eo270rq	プロセス定義名	メッセージキャッチのプロセス
プロセス定義キー	event_message_usage_intermediate_process	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8esayexpeo2jarq	開始ユーザ	青柳辰巳
開始日時～完了日時	2018/05/11 14:24:07 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

表示倍率: 100 %

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。

個人タスク

検索条件

処理	履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュ	担当を外
☐		メッセージで開始するプロセス			メッセージで開始されたユーザタスク	50	2018/05/22			
☒		メッセージを送信するプロセス			メッセージ開始タスクの次のユーザタスク	50	2018/05/22			
☐		メッセージ境界のプロセス			メッセージ境界イベントユーザタスク	50	2018/05/22			

表示設定

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。

個人タスク

検索条件

処理	履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュ	担当を外
☐		メッセージを送信するプロセス			メッセージ境界を送信する次のユーザタスク	50	2018/05/22			
☐		メッセージ境界のプロセス			メッセージ境界イベントの次のユーザタスク	50	2018/05/22			
☐		メッセージで開始するプロセス			メッセージで開始されたユーザタスク	50	2018/05/22			

表示設定

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。

プロセス詳細

プロセス定義ID	event_message_usage_intermediate_process:1:8esay182eo270rq	プロセス定義名	メッセージキャッチのプロセス
プロセス定義キー	event_message_usage_intermediate_process	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8esayexpo2jarq	開始ユーザ	青柳辰巳
開始日時～完了日時	2018/05/11 14:24:07 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

開始 → メッセージキャッチイベント → メッセージキャッチイベントの次のユーザタスク → 終了

表示倍率: 100%

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。

個人タスク

検索条件

処理	履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当者
☒		メッセージを送信するプロセス			メッセージ境界を送信する次のユーザタスク	50	2018/05/22			
☐		メッセージ境界のプロセス			メッセージ境界イベントの次のユーザタスク	50	2018/05/22			
☐		メッセージで開始するプロセス			メッセージで開始されたユーザタスク	50	2018/05/22			

表示設定

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。

個人タスク

検索条件

処理	履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当者
☐		メッセージを送信するプロセス			メッセージキャッチを送信する次のユーザタスク	50	2018/05/22			
☐		メッセージキャッチのプロセス			メッセージキャッチイベントの次のユーザタスク	50	2018/05/22			
☐		メッセージ境界のプロセス			メッセージ境界イベントの次のユーザタスク	50	2018/05/22			
☐		メッセージで開始するプロセス			メッセージで開始されたユーザタスク	50	2018/05/22			

表示設定

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

i 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」

i 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](#)

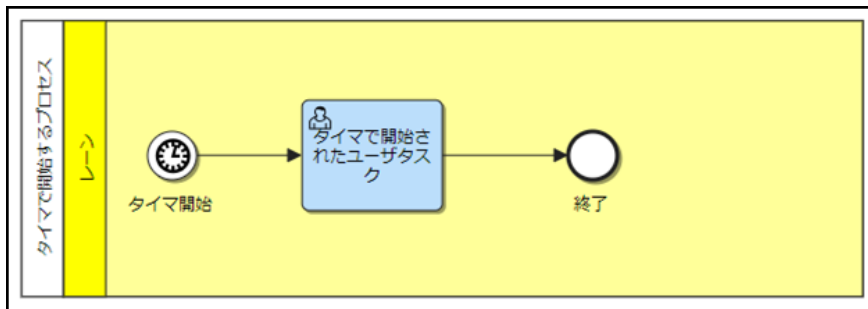
このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。

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

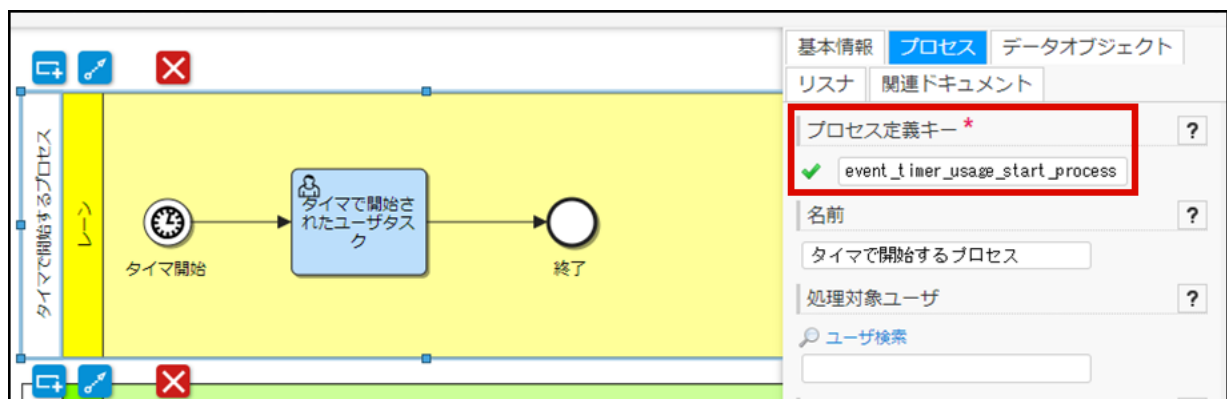
指定した時間間隔で複数回プロセスを開始するプロセス定義を作成します。



図：完成イメージ

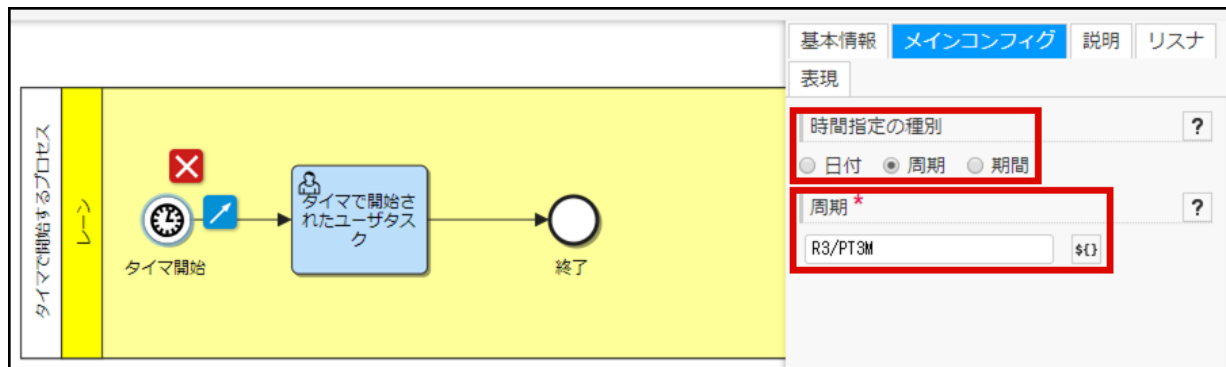
1. キャンパスの空白部分、または、プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロパティ」の「プロセス」タブから、項目を以下のように設定してください。

- 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」の「メインコンフィグ」タブから、以下のように項目を設定します。
- 時間指定の種類：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」

i 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

i コラム

cron式で周期を指定する

「タイマイイベント」の「周期」は、cronでも記述できます。
cronは、左から[秒][分][時間][日][月][曜日][年]で設定します。

例：5分毎→0 0/5 * * * ?
毎年4月3日2時1分→0 1 2 3 4 ? *
2019年、毎日1時2分 →0 2 1 * * ? 2019

cron式の各フィールドと詳細

フィールド名	設定可能範囲	使用可能な特殊記号
秒	0～59	, - * /
分	0～59	, - * /
時間	0～23	, - * /
日	1～31	, - * ? / L W
月	1～12	, - * /
曜日	1～7、SUN～SAT	, - * ? / L #
年	空, 1970～2199	, - * /

! 注意

タイマイイベントは、一定間隔毎に指定された時間を経過しているか判定して実行されます。
そのため、特に[秒]フィールドを使用する時間指定は、数秒遅れる場合があります。

- 「, (カンマ)」
「,」は、複数の数字を設定する際に使用します。
例：毎日10時と18時 → 0 0 10,18 * * ?
- 「- (ダッシュ)」
「-」は連続した範囲を指定できます。
[日]フィールドを「1-4」に設定すると、指定した月の1日から4日が設定されます。
例：毎月1日から4日の、3時2分→0 2 3 1-4 * * ?
毎日1時2分に開始し、1時5分までの1分毎→0 2-5 1 * * ?
- 「* (アスタリスク)」
「*」はフィールドに設定できるすべての値を対象にできます。
例：2019年毎月毎日毎時5分→0 5 * * * ? 2019
- 「? (クエスチョン マーク)」
「?」は他の条件による設定を許容できます。
曜日を問わず「1日」を指定したい場合、[日]のフィールドを「1」、[曜日]フィールドは「?」にします。
[日]フィールドと[曜日]フィールドを同時に設定できないため、いずれかを「?」で指定する必要があります。

例：2019年毎月1日の10時→0 0 10 1 * ? 2019

毎年毎月金曜日の18時→0 0 18 ? * 6 *

■ 「/（スラッシュ）」

「/」で繰り返し期間を指定できます。

左側の数字から開始し、右側の数字毎に繰り返します。

例：[分]の部分に「0/15」を設定→0分から15分毎に繰り返す

[日]の部分に「1/6」を設定→1日から6日毎に繰り返す



注意

「/」による繰り返しは、各フィールドの範囲内でのみ行われます。

[月]のフィールドに対して「7/6（7月から開始して6ヶ月毎に繰り返す）」を指定しても、来年の1月には及ばず、当年の7月だけが設定されます。

[秒]のフィールドに対して「5/30（5秒から開始して30秒毎に繰り返す）」を指定しても、同じ分の5秒と35秒のみ設定されます。

■ 曜日の英語表記

[曜日]フィールドに対しては英語の省略形を使用できます。

例：毎週水曜日の18時30分→0 30 18 ? * WED

■ 「L」

[曜日]フィールドに対する「L」は、月の最終週を指定できます。

例：毎月最終金曜日の2時1分→0 1 2 ? * 6L

[日]フィールドに対する「L」は、月の最終日を指定できます。

例：2019年4月30日の2時1分0秒→0 1 2 L 4 2019

■ 「W」

[日]フィールドに対する「W」は、指定した日付に最も近い平日を設定できます。

例：設定されたのが「15W」だった場合

「15日が土曜日」である月の場合、イベントが開始するのは「14日の金曜日」です。

「15日が日曜日」である月の場合、イベントが開始するのは「16日の月曜日」です。



注意

「W」による選定は、[月]フィールドの範囲内で行われます。

月を跨ぐ日にちが設定されることはありません。

例1. **0 0 0 1W 6 ? 2019** [月]のフィールドが「6」、[日]フィールドが[W1]で設定されている場合
2019年の6月1日は土曜日であり、最も近い平日は金曜日の5月31日です。
しかし、[月]フィールドに設定した[6]が優先されるのため、6月3日（月曜日）に指定されます。

例2. **0 0 0 30W 6 ? 2019** [月]のフィールドが「6」、[日]フィールドが[W30]で設定されている場合
2019年の6月30日は日曜日であり、最も近い平日は月曜日の7月1日です。
しかし、[月]フィールドに設定した[6]が優先されるのため、6月28日（金曜日）に指定されます。

■ 「LW(LastWeekday)」

[日]フィールドに対して「LW」と設定することで、月の最後の平日を指定できます。

例：0 0 10 ? * LW * → 毎月最後の平日の10時

■ 「#（シャープ）」

「#」は[曜日]フィールドに使用することで、週と曜日を設定できます。

左側の数字が曜日を表し、右側の数字が第何週目かを表します。

例：「4#2」→月の第2水曜日

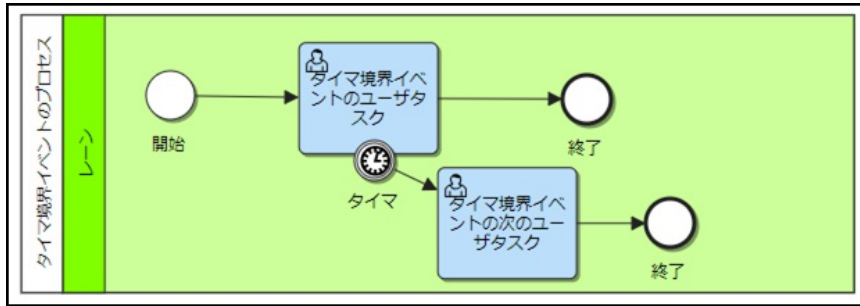
3. 「タイマ開始イベント」が指定した時間の後「プロセス」を開始したことを確認するため、「ユーザタスク」を配置します。

4. 「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。



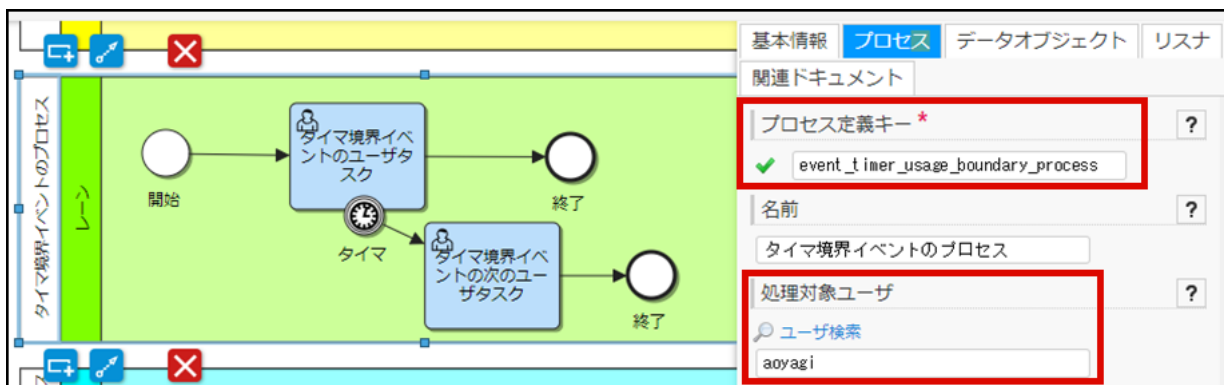
指定した時間経過によりユーザタスクを中断するプロセス定義を作成する

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。



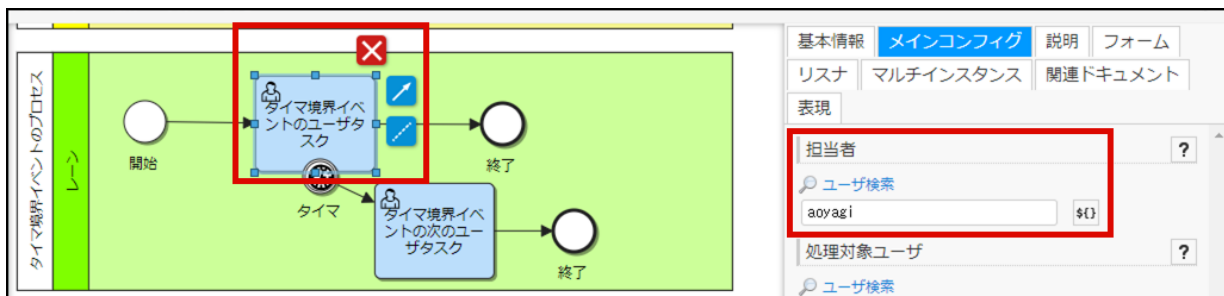
図：完成イメージ

1. 「開始イベント」を配置します。
2. キャンパスの空白部分、または、プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロパティ」の「プロセス」タブから、項目を以下のように設定してください。
 - プロセス定義キー：event_timer_usage_boundary_process
 - 処理対象ユーザ：aoyagi



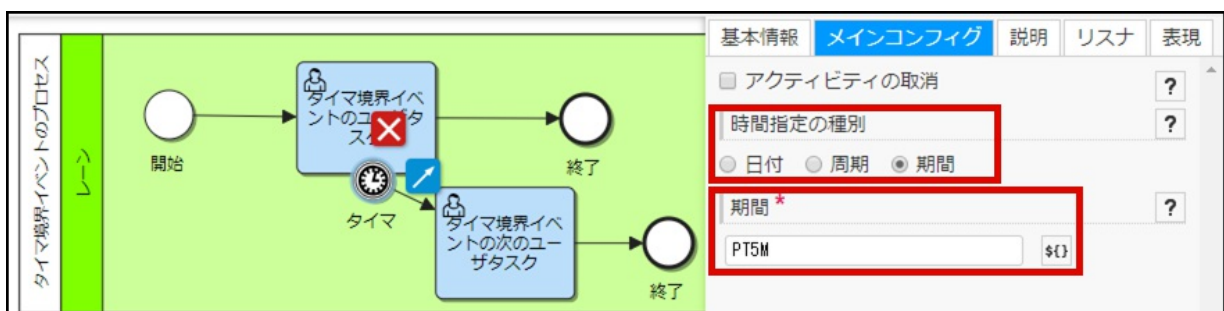
図：「タイマ境界イベントのプロセス」 - 「プロパティ」 - 「プロセス」

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

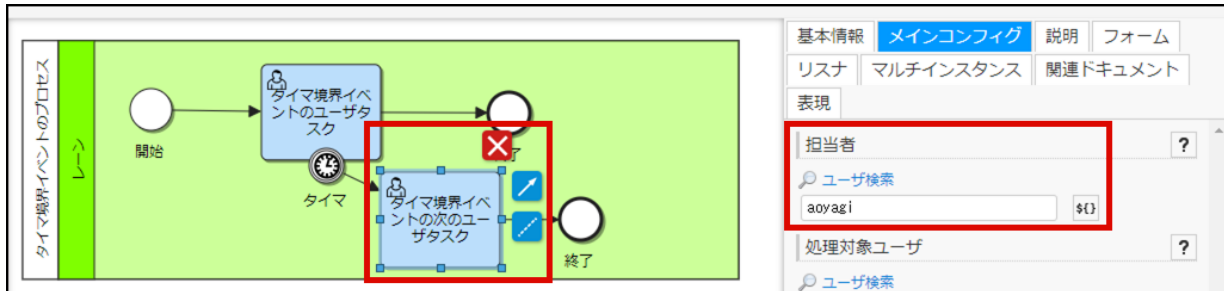


図: 「ユーザタスク」

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。

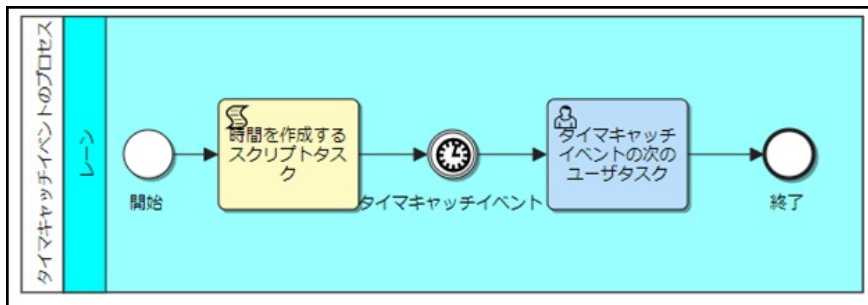


図: 完成イメージ

1. 「開始イベント」を設置します。
2. キャンパスの空白部分、または、プールをクリックし、「プロパティ」をプロセス全体切り替えます。
「プロパティ」の「プロセス」タブから、項目を以下のように設定してください。
 - プロセス定義キー: event_timer_usage_intermediate_process
 - 処理対象ユーザ: aoyagi

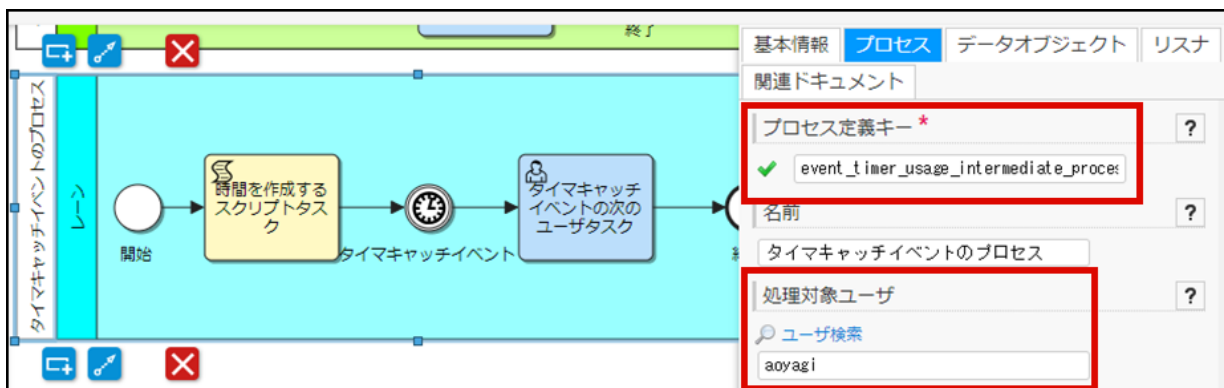
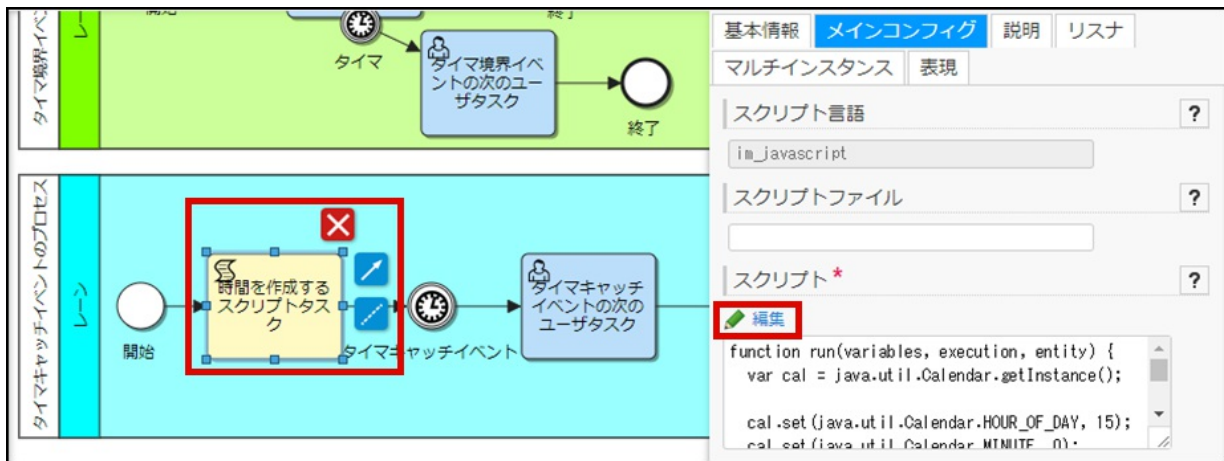


図: 「タイマキャッチイベントのプロセス」 - 「プロパティ」 - 「プロセス」

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」タブから、「スクリプト」の「編集」リンクをクリックします。



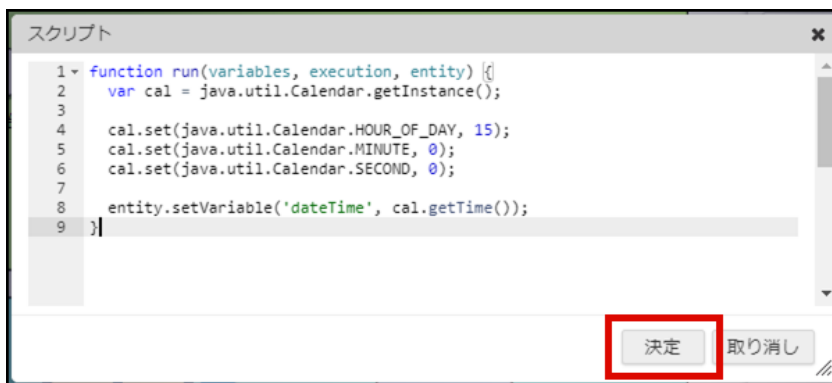
図：「スクリプトタスク」 - 「プロパティ」 - 「メインコンフィグ」

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();

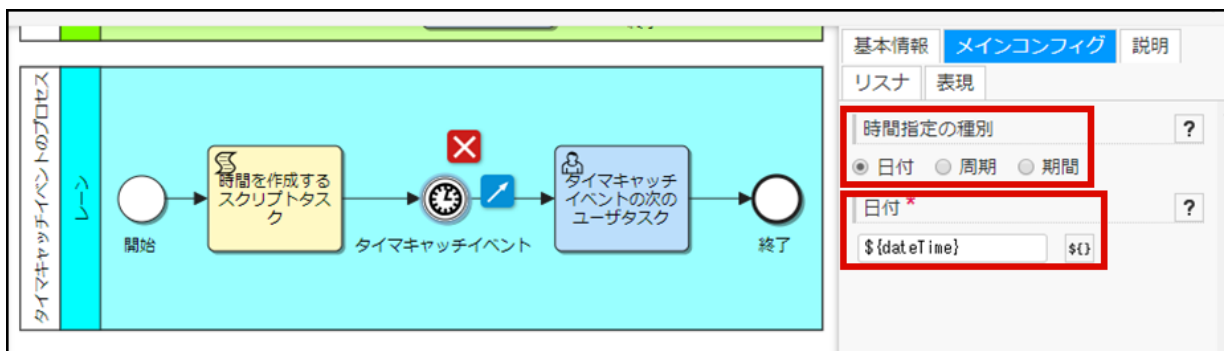
  cal.set(java.util.Calendar.HOUR_OF_DAY, 15);
  cal.set(java.util.Calendar.MINUTE, 0);
  cal.set(java.util.Calendar.SECOND, 0);

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



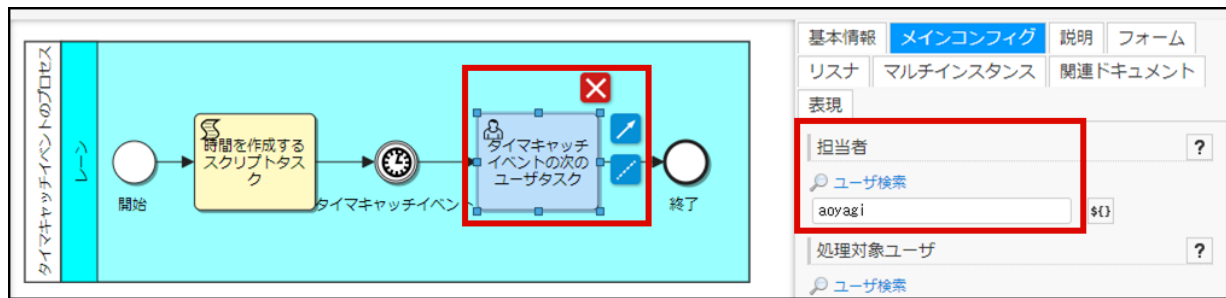
図：スクリプト編集のポップアップ

5. 「タイムキャッチイベント」の「メインコンフィグ」で、「時間指定の種別」と「日付」を設定します。
- 時間指定の種別：日付
 - 日付：`${dateTime}`



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

6. タイム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** により、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。

個人タスク										
検索条件										
処理	履歴	プロセス定義名	業務キ	カテ	タスク名	優先	作成日時	期限日時	ドキュメ	担当を外
☐		タイマで開始するプロセス			タイマで開始されたユーザタスク	50	2018/06/22 9:59:23			
☐		タイマで開始するプロセス			タイマで開始されたユーザタスク	50	2018/06/22 9:56:23			
☐		タイマで開始するプロセス			タイマで開始されたユーザタスク	50	2018/06/22 9:53:27			

図：「タスク一覧」

2. 「プロセス開始一覧」から、「プロセス定義名」 `\timer_catching_event\` のプロセス と `\timer_boundary\` のプロセス を実行します。

プロセス開始一覧 Forma管理画面 サイトマップ

タスク一覧 グループタスク一覧 個人タスク一覧 処理済一覧

検索条件

プロセス定義名

カテゴリ

検索 クリア

プロセス開始	プロセス図	プロセス定義名	カテゴリ	説明	ドキュメント
		タイマキャッチイベントのプロセス	http://www.intra-mart.jp/im_bpm		
		タイマ境界イベントのプロセス	http://www.intra-mart.jp/im_bpm		

1件中 1 - 2 を表示

図：「プロセス開始一覧」

3. 「タスク一覧」の「個人タスク」で、`\timer_boundary\` 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。

個人タスク

検索条件

表示設定

処理	履歴	プロセス定義名	業務キ	カテ	タスク名	優先	作成日時	期限日時	ドキュメ	担当を列
		タイマ境界イベントのプロセス			タイマ境界イベントのユーザタスク	50	2018/06/22 14:54:58			
		タイマで開始するプロセス			タイマで開始されたユーザタスク	50	2018/06/22 9:59:23			
		タイマで開始するプロセス			タイマで開始されたユーザタスク	50	2018/06/22 9:56:23			
		タイマで開始するプロセス			タイマで開始されたユーザタスク	50	2018/06/22 9:53:27			

1件中 1 - 5 を表示

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。

プロセス詳細 Forma管理画面 サイトマップ

プロセス一覧 変数一覧 プロセス履歴 プロセス移行 プロセス一括移行 プロセス削除 ドキュメント

プロセス定義ID	event_timer_usage_boundary_process:1:8etyp4ounvo36rq	プロセス定義名	タイマ境界イベントのプロセス
プロセス定義キー	event_timer_usage_boundary_process	バージョン	1
カテゴリ	http://www.intra-mart.jp/im_bpm	業務キー	
プロセスインスタンスID	8etz00dkmvooxrq	開始ユーザ	青柳辰巳
開始日時～完了日時	2018/06/22 14:54:58 ~	ステータス	実行中

プロセス図とタイムラインを拡大表示

表示倍率: 100 %

2018/06/22 15:00

タイマ境界イベントのユーザタスク

以下の理由によりタスクを中止しました。

- タイマのタイマが経過しました。

開始 2018/06/22 14:54:58
終了 2018/06/22 15:00:02
経過 5分

2018/06/22 15:00

タイマ境界イベントの次のユーザタスク

タスクの処理を待っています。

- 担当者: 青柳辰巳 (処理中)

開始 2018/06/22 15:00:02
経過 1時間 57分

図: 「プロセス一覧」 - 「プロセス詳細」

5. 「プロセス一覧」 \timer_catching_event\ のプロセス が「15時」に起動することを確認します。

個人タスク

検索条件

表示設定

処理	履歴	プロセス定義名	業務	カテ	タスク名	優先	作成日時	期限	ドキ	担当
☐		タイマで開始するプロセス			タイマで開始されたユーザタスク	50	2018/06/22 9:53:27			
☐		タイマで開始するプロセス			タイマで開始されたユーザタスク	50	2018/06/22 9:56:23			
☐		タイマで開始するプロセス			タイマで開始されたユーザタスク	50	2018/06/22 9:59:23			
☐		タイマキャッチイベントのプロセス			タイマキャッチイベントの次のユーザタスク	50	2018/06/22 15:00:02			
☐		タイマ境界イベントのプロセス			タイマ境界イベントの次のユーザタスク	50	2018/06/22 15:00:02			

1 ページ中 1 ページ目 5 件中 1 - 5 を表示

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。

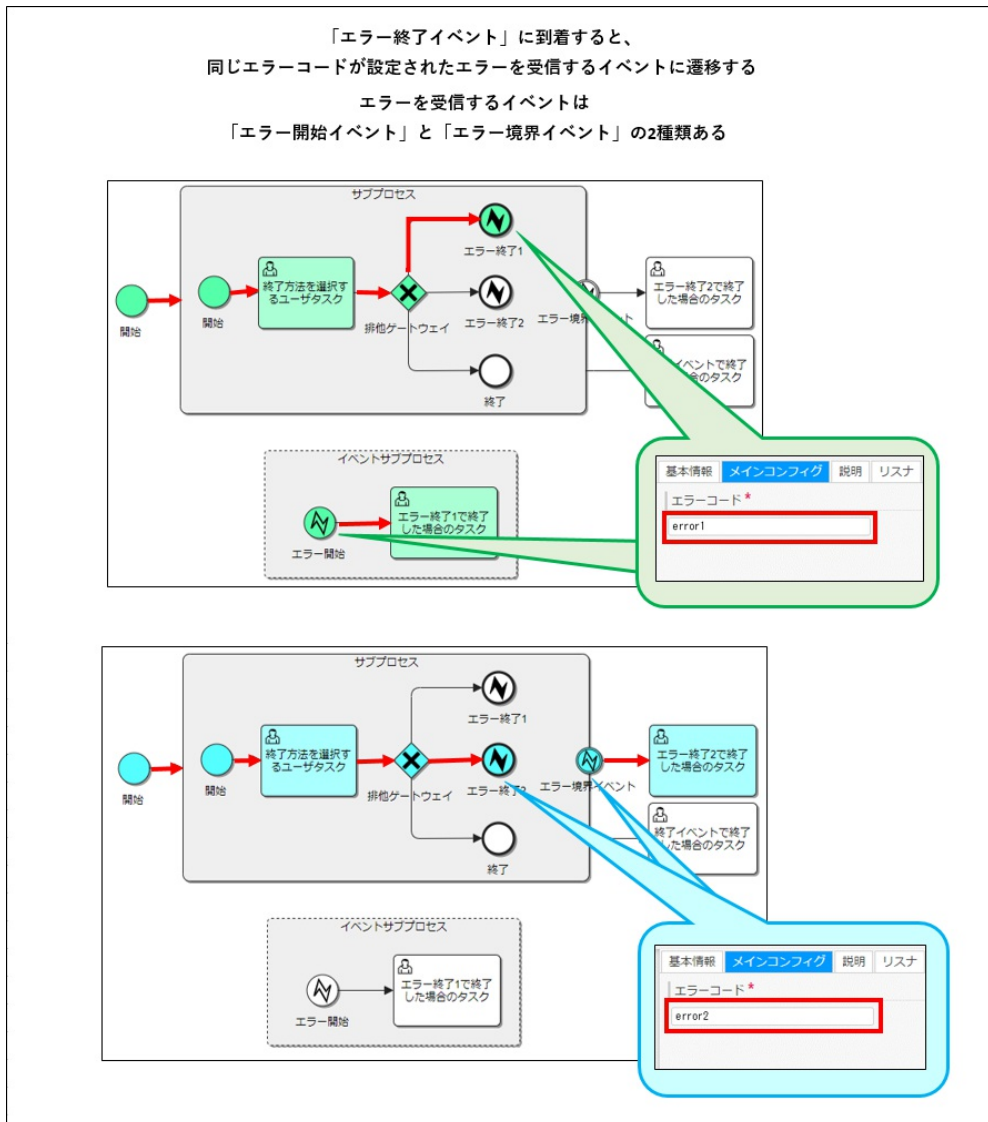
エラーイベントを使用する

このチュートリアルでは、エラーイベントを使用して例外処理を行う方法を解説します。

エラーイベントは、「設定されているエラーコード」と「エラー終了イベントなどで発生したエラーコード」を比較して合致した場合、フローを進行させます。

エラーをトリガとするイベントは、以下のとおりです。

- エラー開始イベント
- エラー境界イベント
- エラー終了イベント



図：概要図

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-event_error_usage.zip](#)

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。

- エラー開始イベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「エラー開始イベント」
- エラー境界イベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「エラー境界イベント」
- エラー終了イベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「エラー終了イベント」

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_error_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

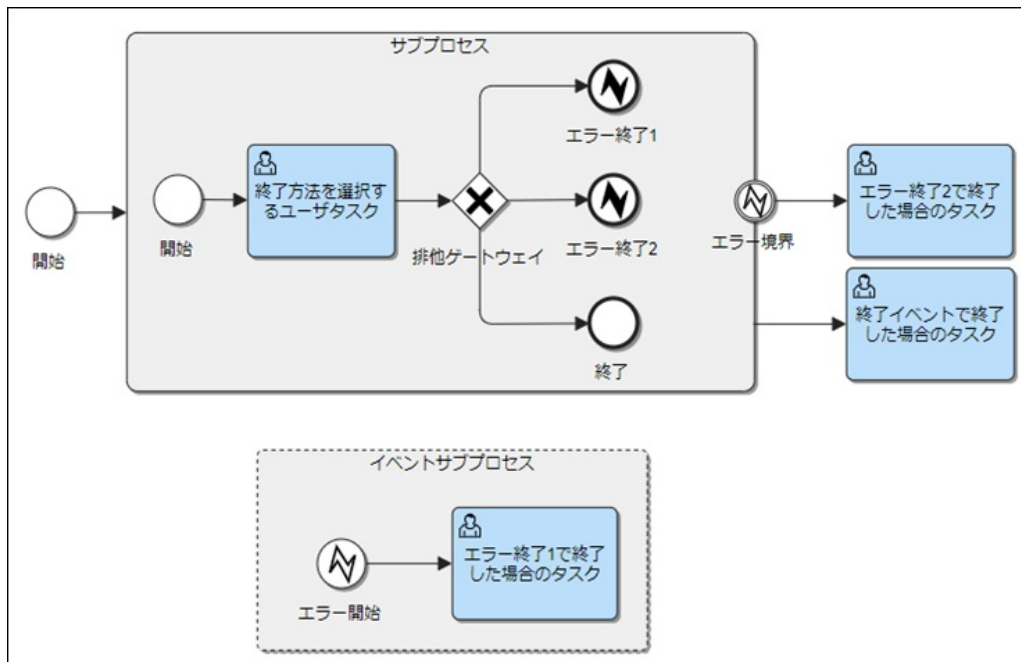
プロセス定義を作成する

下記の図は「設定が異なるエラー終了イベント」と「終了イベント」に分岐しているプロセス定義です。

- 「エラー終了1」から「エラー開始」イベントが設置された「イベントサブプロセス」へ移行し、「エラー終了1で終了した場合のタスク」へ進むルート
- 「エラー終了2」から「エラー境界」イベントに移行し、「エラー終了2で終了した場合のタスク」へ進むルート
- 「終了」イベントでサブプロセスを終了し、「終了イベントで終了した場合のタスク」へ進むルート

「終了方法を選択するユーザタスク」に到達すると、チュートリアル開始前にインポートしたFormaアプリケーション「動作確認したい終了イベントを選択する」画面が表示されます。

画面で選択した終了方法にむけて、排他ゲートウェイからプロセスが分岐します。

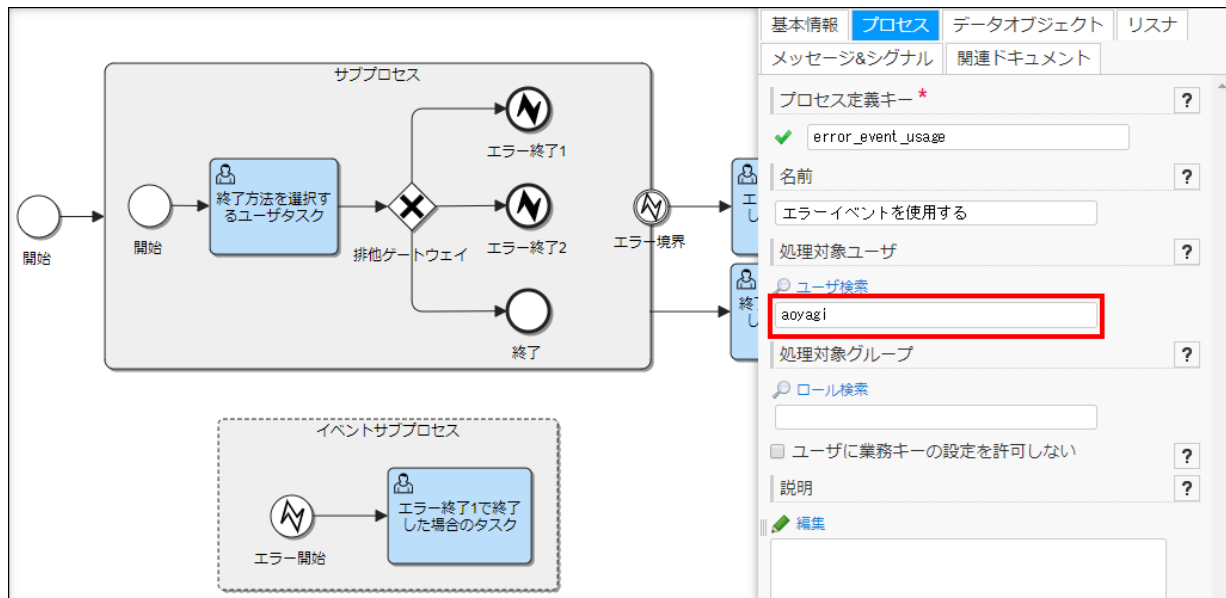


図：プロセス定義図「エラーイベントを使用する」

注意

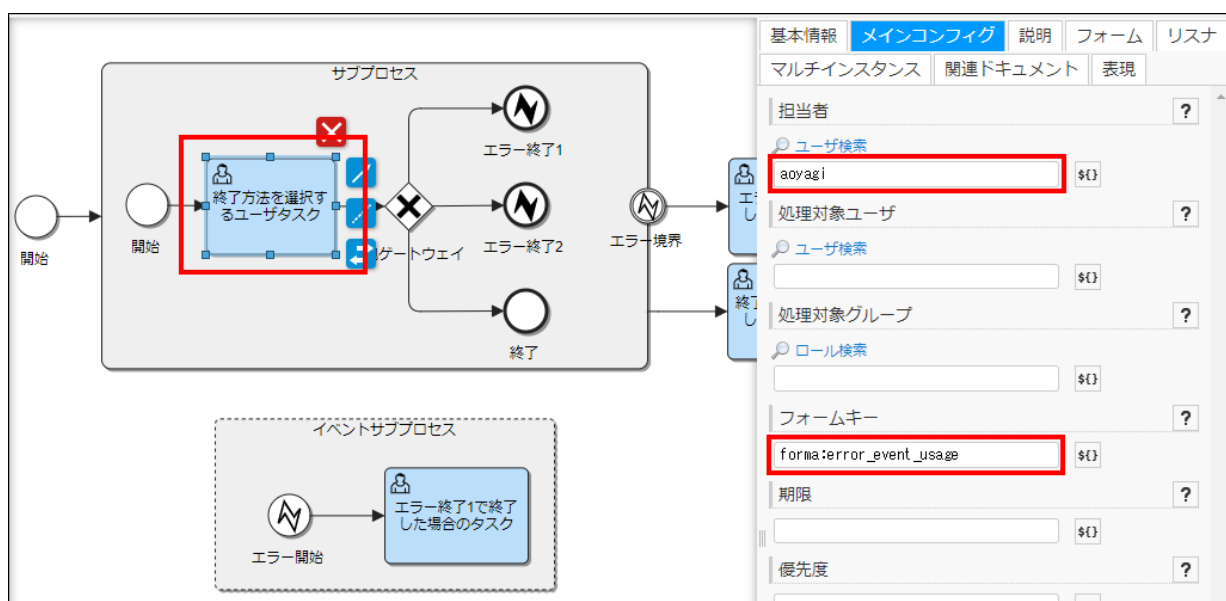
エラーを受信できる範囲は、エラーが発生した箇所から参照できるプロセス内のみです。

1. 「開始イベント」を設置します。
2. プロセスの処理担当ユーザを設定します。
キャンパスの空白部分をクリックし、「プロパティ」をプロセス全体切り替えます。
「プロセス」タブから、処理対象ユーザを「[aoyagi](#)」に設定します。



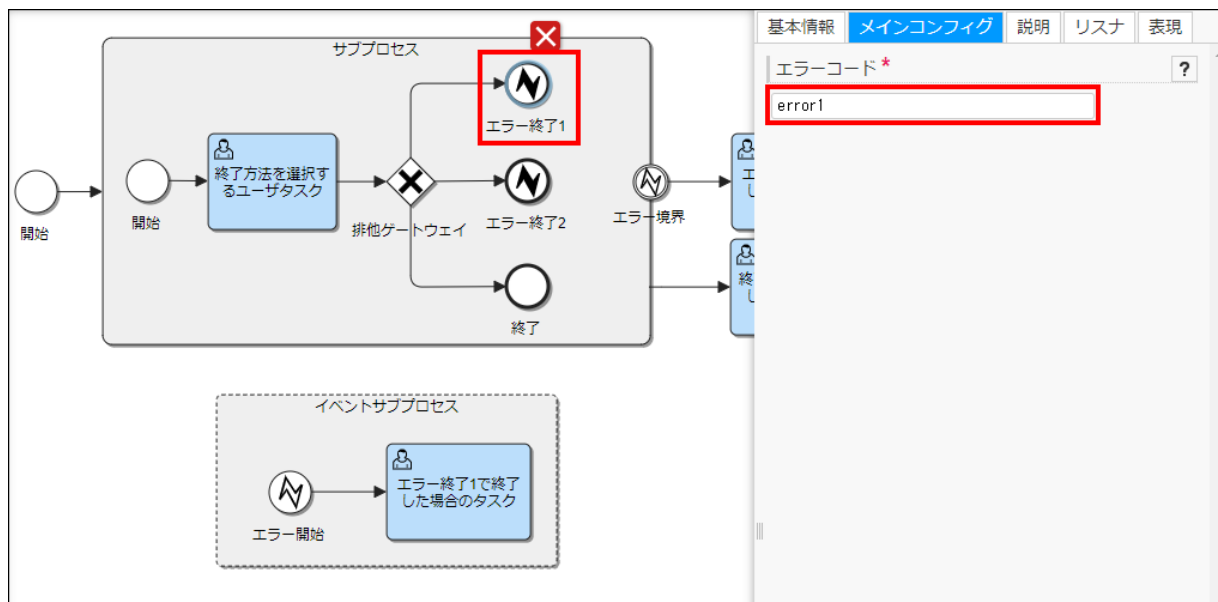
図：プロセス全体 - 「プロパティ」 - 「プロセス」 - 「処理対象ユーザ」

3. サブプロセスを設置します。
4. サブプロセス内に「開始イベント」を設置します。
5. Forma画面から終了方法を選択させるユーザタスクを設置します。
「ユーザタスク」を設置します。
6. 処理対象ユーザと、終了方法を選択するForma画面を設定をします。
「ユーザタスク」をクリックし、「メインコンフィグ」タブから、以下のように項目を設定してください。
 - 処理対象ユーザ: aoyagi
 - フォームキー: forma:error_event_usage



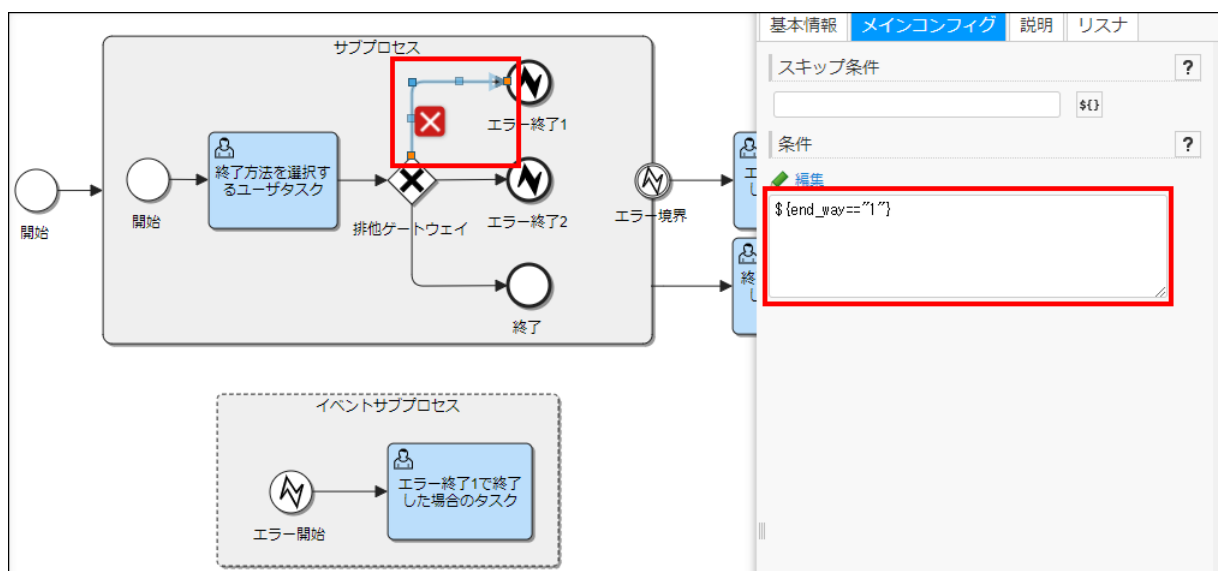
図：「サブプロセス」 - 「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

7. Forma画面で選択された情報をもとに、プロセスの進行先が変更されるようにします。
「排他ゲートウェイ」を設置します。
8. 「エラー終了1」からイベントサブプロセス内の「エラー開始」へ進行するルートを作成します。
「エラー終了イベント」を設置します。
9. 「エラー終了1」に到達した際に送信するエラーコードを設定します。
「エラー終了イベント」を選択した状態で、「メインコンフィグ」タブから「エラーコード」に「error1」と設定します。



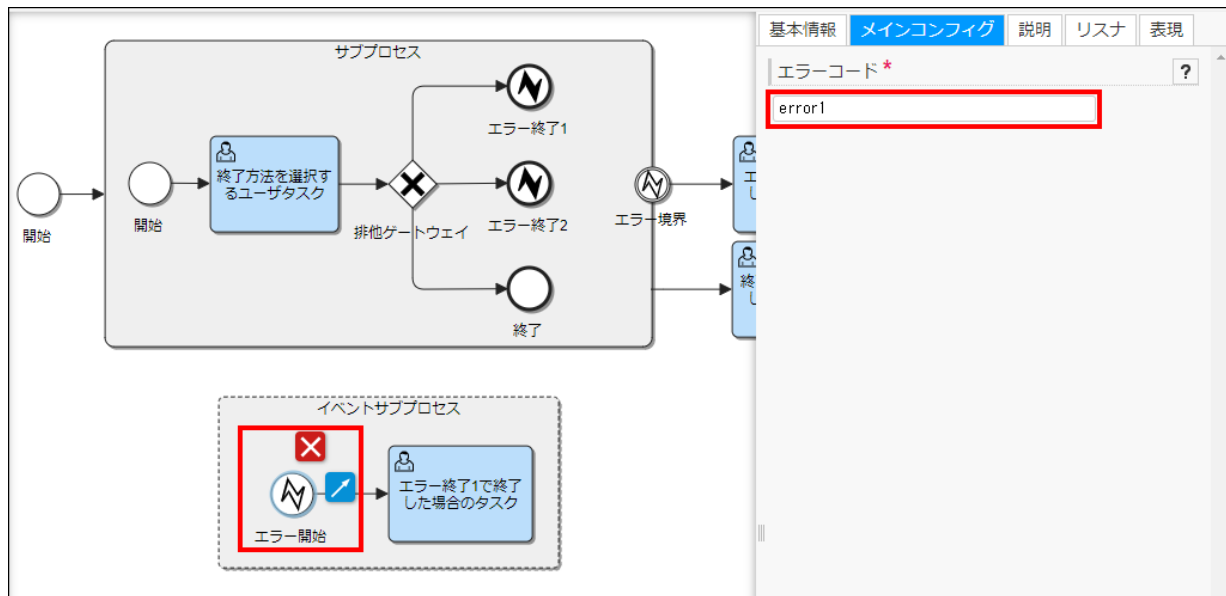
図：「サブプロセス」 - 「エラー終了イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「エラーコード」

10. Forma画面で「エラー終了1」が指定された場合に「エラー終了1」へ進行するように条件付けをします。
 「排他ゲートウェイ」から「エラー終了1」へ続く「シーケンスフロー」を選択します。
 「メインコンフィグ」タブから、「条件」に `${end_way=="1"}` と設定します。



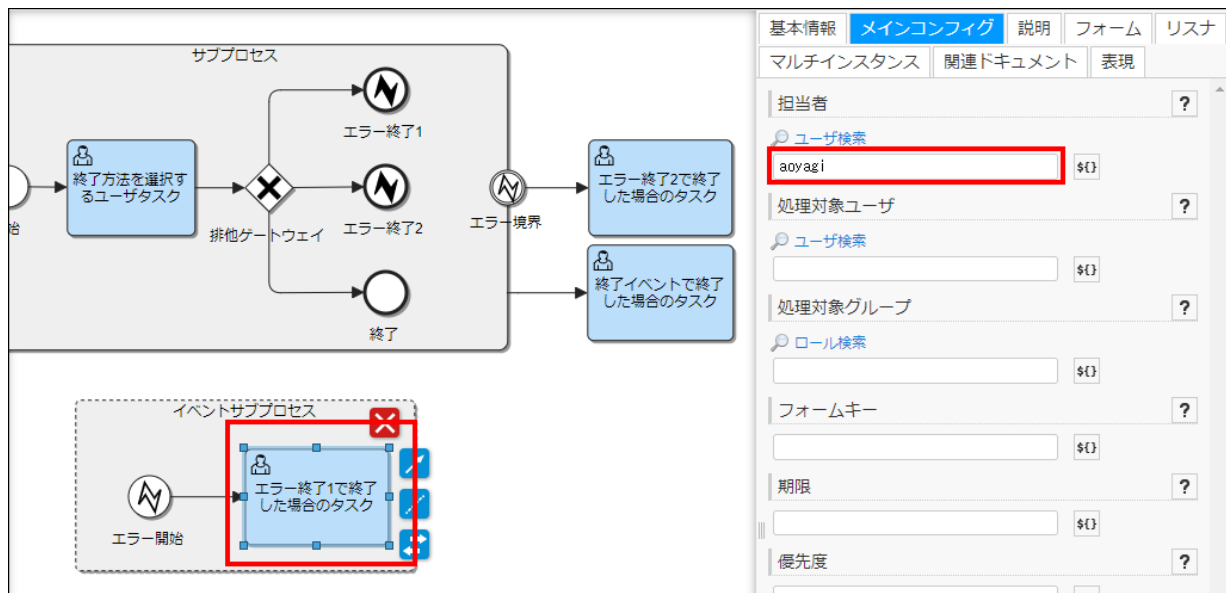
図：「サブプロセス」 - 「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」 - 「条件」

11. エラー終了イベントの遷移先としてイベントサブプロセスを用意します。
 「イベントサブプロセス」を設置します。
12. 「エラー終了1」に紐づくエラー開始イベントを設置します。
 「エラー開始イベント」を設置します。
13. 「エラー終了1」に設定したエラーコードで紐づけます。
 「エラー開始イベント」を選択した状態で、「メインコンフィグ」タブから「エラーコード」に「error1」と設定します。



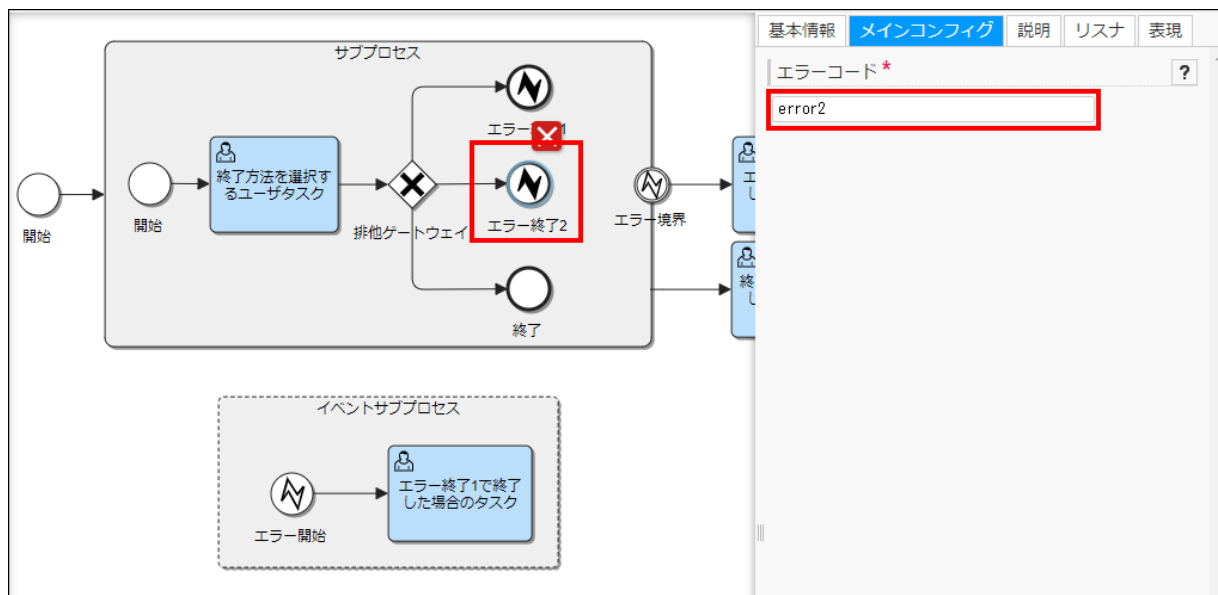
図：「イベントサブプロセス」 - 「エラー開始イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「エラーコード」

14. エラー開始イベントへ移行できたことを確認するためのタスクを設置します。
「ユーザタスク」を設置します。
15. 「エラー終了1で終了した場合のタスク」の担当者を設定します。
「ユーザタスク」を選択した状態で「メインコンフィグ」タブから、担当者に「aoyagi」と設定します。



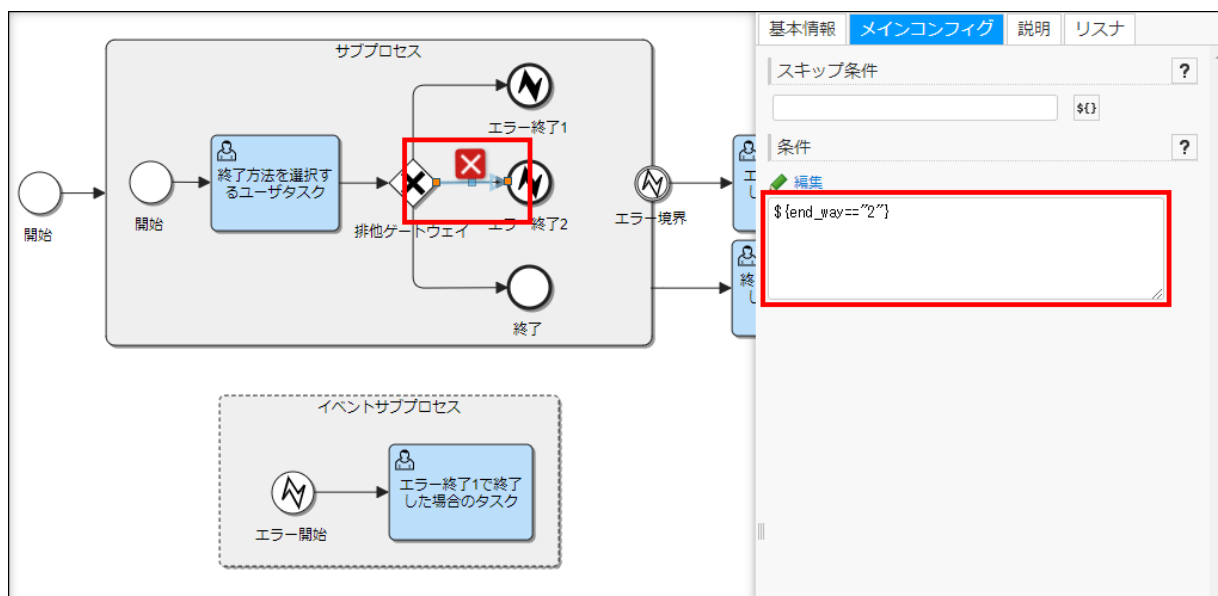
図：「イベントサブプロセス」 - 「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

16. エラー終了イベントから、イベントサブプロセス内のエラー開始イベントへ進行するルートを作成します。
「エラー終了イベント」を設置します。
17. 「エラー終了1」に到達したのち、エラー境界イベントへ移行するように紐づけます。
「エラー終了イベント」を選択した状態で、「メインコンフィグ」タブから「エラーコード」に「error2」と設定します。



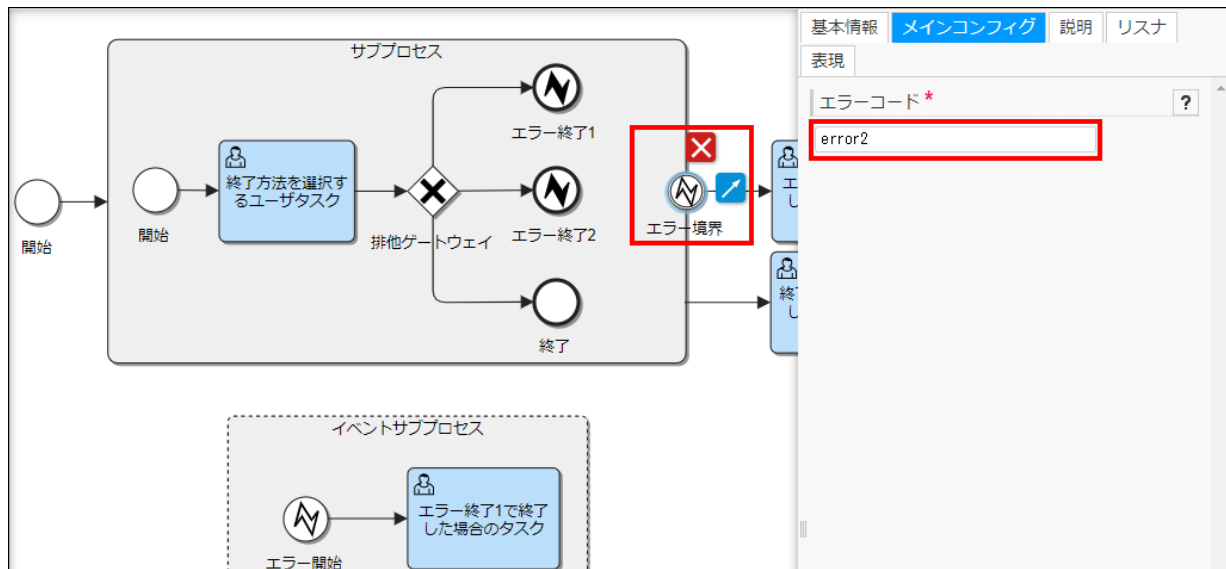
図：「サブプロセス」 - 「エラー終了イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「エラーコード」

18. Forma画面で「エラー終了2」が指定された場合に「エラー終了2」へ進行するように条件付けをします。
 「排他ゲートウェイ」から「エラー終了イベント」へ続く「シーケンスフロー」を選択します。
 「メインコンフィグ」タブから、「条件」に `${end_way=="2"}` と設定します。



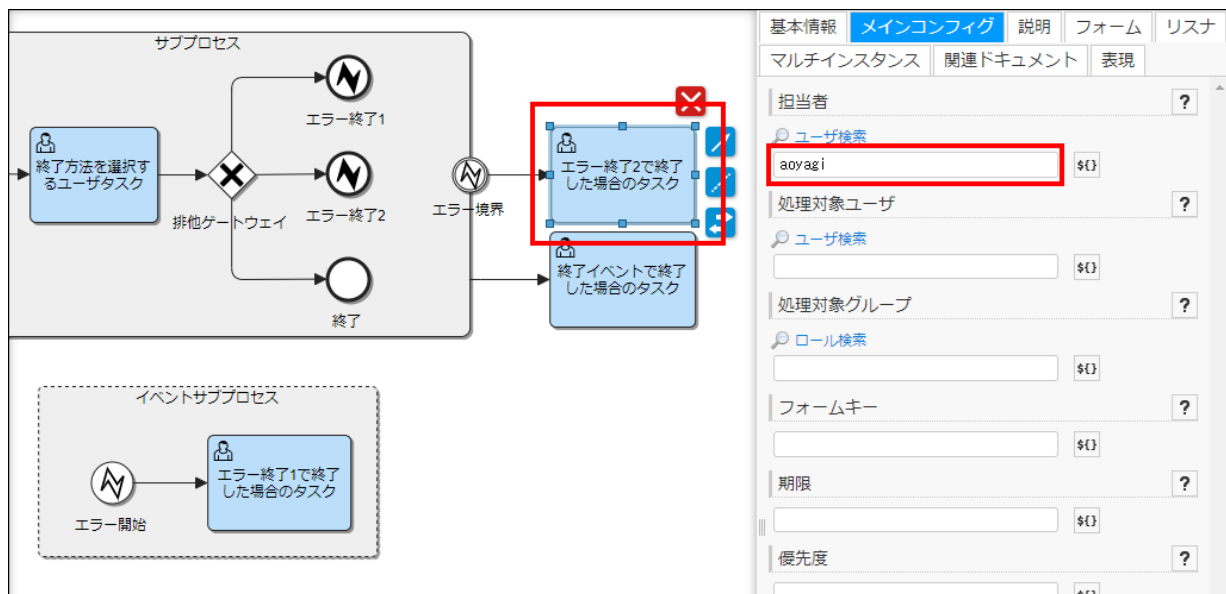
図：「サブプロセス」 - 「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」 - 「条件」

19. エラー終了イベントに到達したのち、エラー境界イベントを介してサブプロセス外のユーザタスクへ移行させます。
 「エラー境界イベント」を設置します。
20. エラー終了イベントが送信するエラーコードを設定します。
 「エラー境界イベント」を選択した状態で、「メインコンフィグ」タブから「エラーコード」に「error2」と設定します。



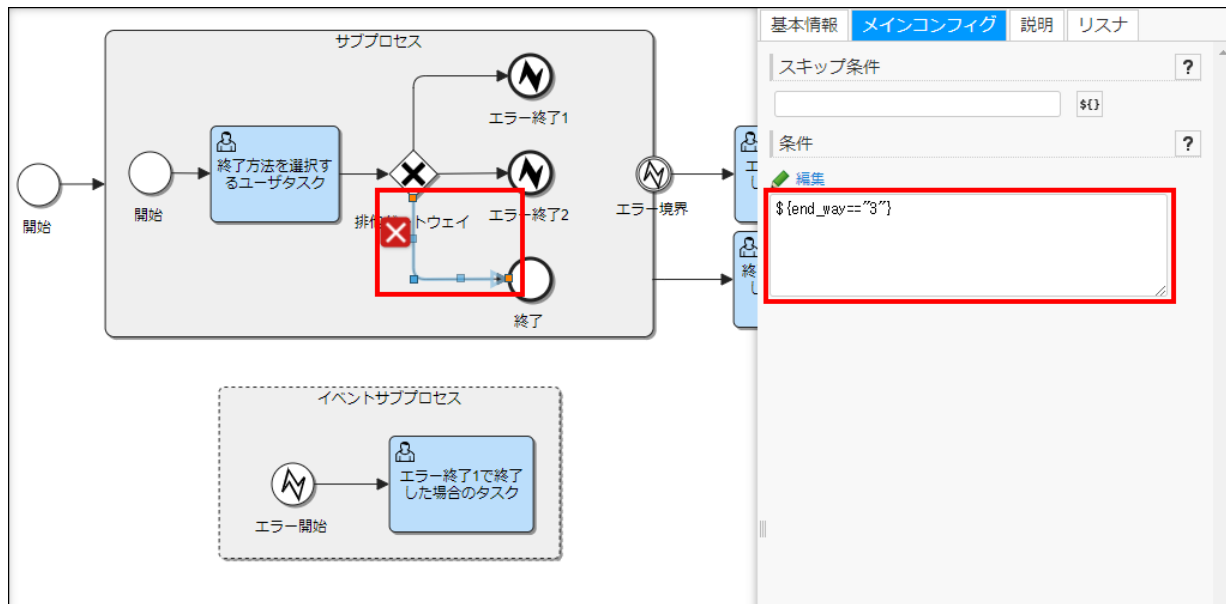
図：「サブプロセス」 - 「エラー境界イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「エラーコード」

21. エラー境界イベントへ移動できたことを確認するためのユーザタスクを設置します。
「ユーザタスク」を設置します。
22. 「エラー終了2で終了した場合のタスク」の担当者を設定します。
「ユーザタスク」を選択した状態で「メインコンフィグ」タブから、担当者に「aoyagi」と設定します。



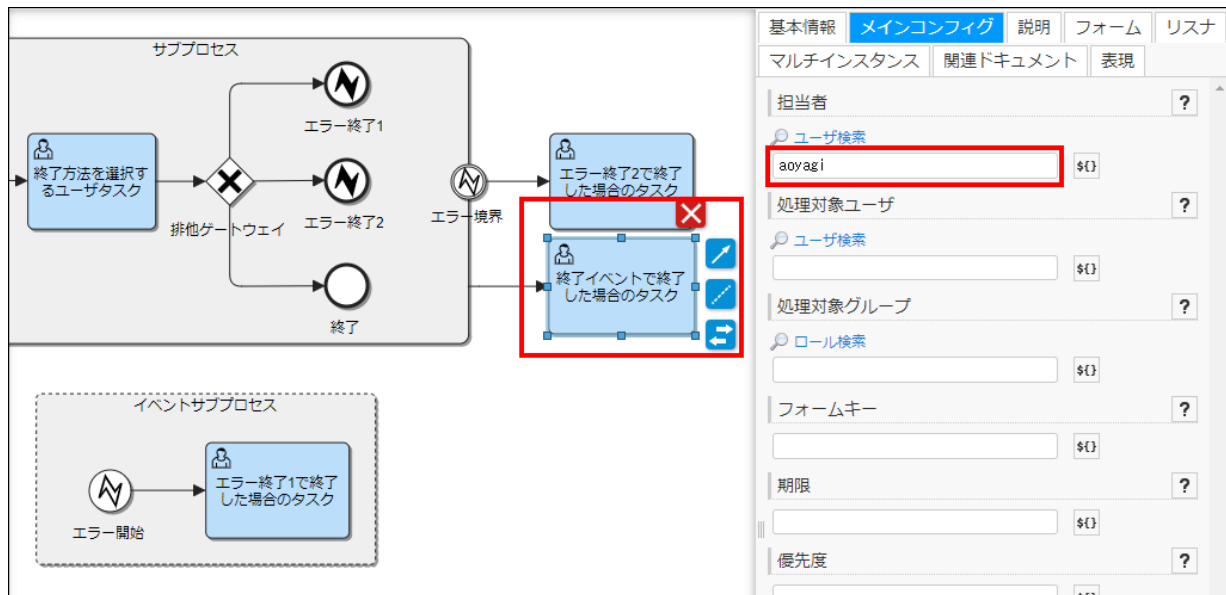
図：「ユーザタスク」 - 「エラー終了イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

23. 通常の終了イベントを使用したルートを作成します。
「終了イベント」を設置します。
24. Forma画面で終了イベントが指定された場合に、終了イベントへ進行するように条件付けをします。
「排他ゲートウェイ」から「終了イベント」へ続く「シーケンスフロー」を選択します。
「メインコンフィグ」タブから、「条件」に `${end_way=="3"}` と設定します。



図：「サブプロセス」 - 「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」 - 「条件」

25. 終了イベントに到達し、サブプロセスが終了したことを確認するためのタスクを設置します。
「ユーザタスク」を設置します。
26. 「終了イベントで終了した場合のタスク」の担当者を設定します。
ユーザタスクを選択した状態で「メインコンフィグ」タブから、担当者に「aoyagi」と設定します。



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

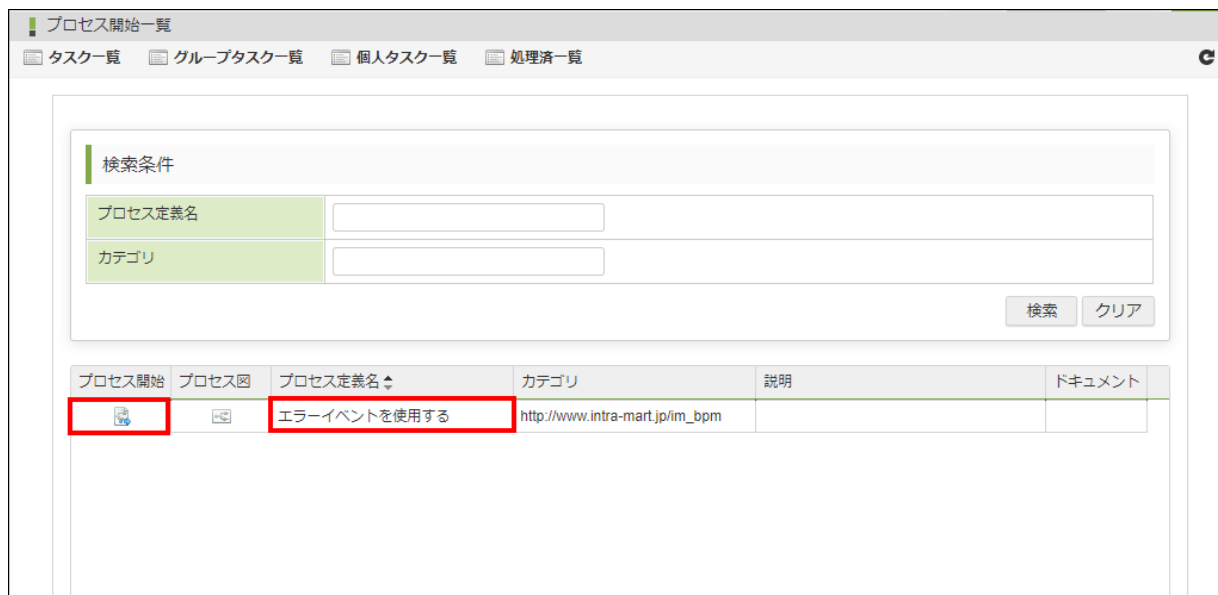
結果を確認する

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。

結果確認の共通動作

プロセスの開始からForma画面までの動作は共通しているため、適宜実施してください。

1. プロセスを開始します。
「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示し、「エラーイベントを使用する」の「」ボタンをクリックします。



図：「プロセス開始一覧」

2. タスクを確認します。
「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
3. 「個人タスク」から、「終了方法を選択するユーザタスク」の「

図：「プロセス開始一覧」

4. 「動作確認したい終了イベントを選択する」画面が表示されます。
以降の動作は、各イベントごとの説明に従って実行してください。

動作確認したい終了イベントを選択する

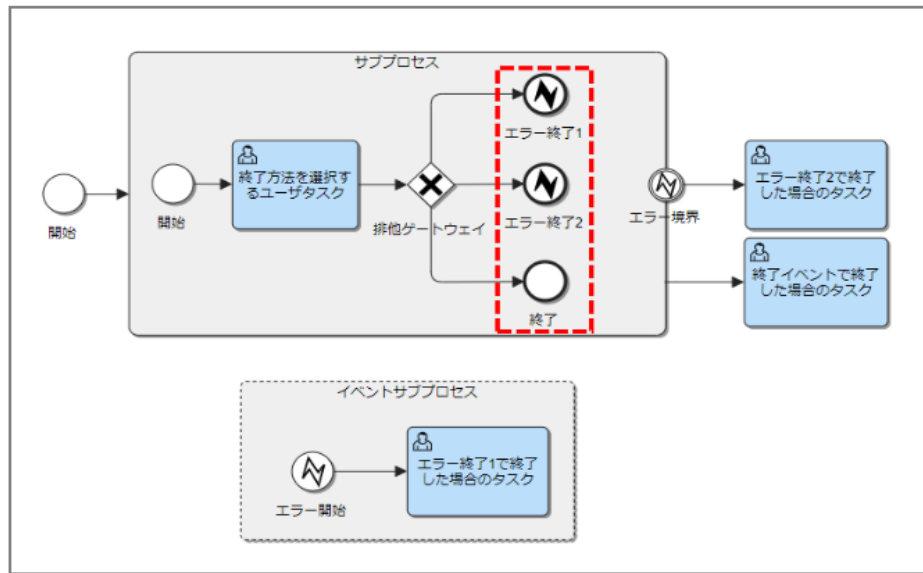
排他ゲートウェイから進行する終了イベントを選択します。

エラー開始イベントの動作を確認する場合、「エラー終了1」を選択します。
エラー境界イベントの動作を確認する場合、「エラー終了2」を選択します。
終了イベントの動作を確認する場合、「終了」を選択します。

終了方法の選択

- ☐ エラー終了1
☐ エラー終了2
☐ 終了

決定



図：動作確認したい終了イベントを選択する(IM-FormaDesigner)

「エラー開始イベント」の動作確認

「エラー終了イベント」から「エラー開始イベント」へ進行するプロセスの状態を確認します。

1. 「動作確認したい終了イベントを選択する」画面から「エラー終了1」を選択し、「決定」ボタンをクリックします。

動作確認したい終了イベントを選択する

排他ゲートウェイから進行する終了イベントを選択します。

エラー開始イベントの動作を確認する場合、「エラー終了1」を選択します。
エラー境界イベントの動作を確認する場合、「エラー終了2」を選択します。
終了イベントの動作を確認する場合、「終了」を選択します。

終了方法の選択

- ☒ エラー終了1
☐ エラー終了2
☐ 終了

決定

図：動作確認したい終了イベントを選択する(IM-FormaDesigner)

2. プロセスの状態を確認します。
「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
3. 「個人タスク」に対象のタスクがあることを確認します。
「」アイコンをクリックし、「プロセス詳細」画面を表示します。

個人タスク										
検索条件										
表示設定										
処理	履歴	参照	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント
☐	☐	☐	☒			エラー終了1で終了した場合のタスク	50	2019/05/30		担当をタ

図: 「タスク一覧」 - 「個人タスク」

4. 「プロセス参照」画面、または、「拡大表示」画面から、タスクが下記の状態であることを確認します。

- ステータスが「実行中」であること
- プロセス図で「エラー終了1で終了した場合のタスク」に「▶」がついていること
- タイムラインで、「サブプロセス」が終了していること
「エラー終了1」から「エラー開始」へ進行していること

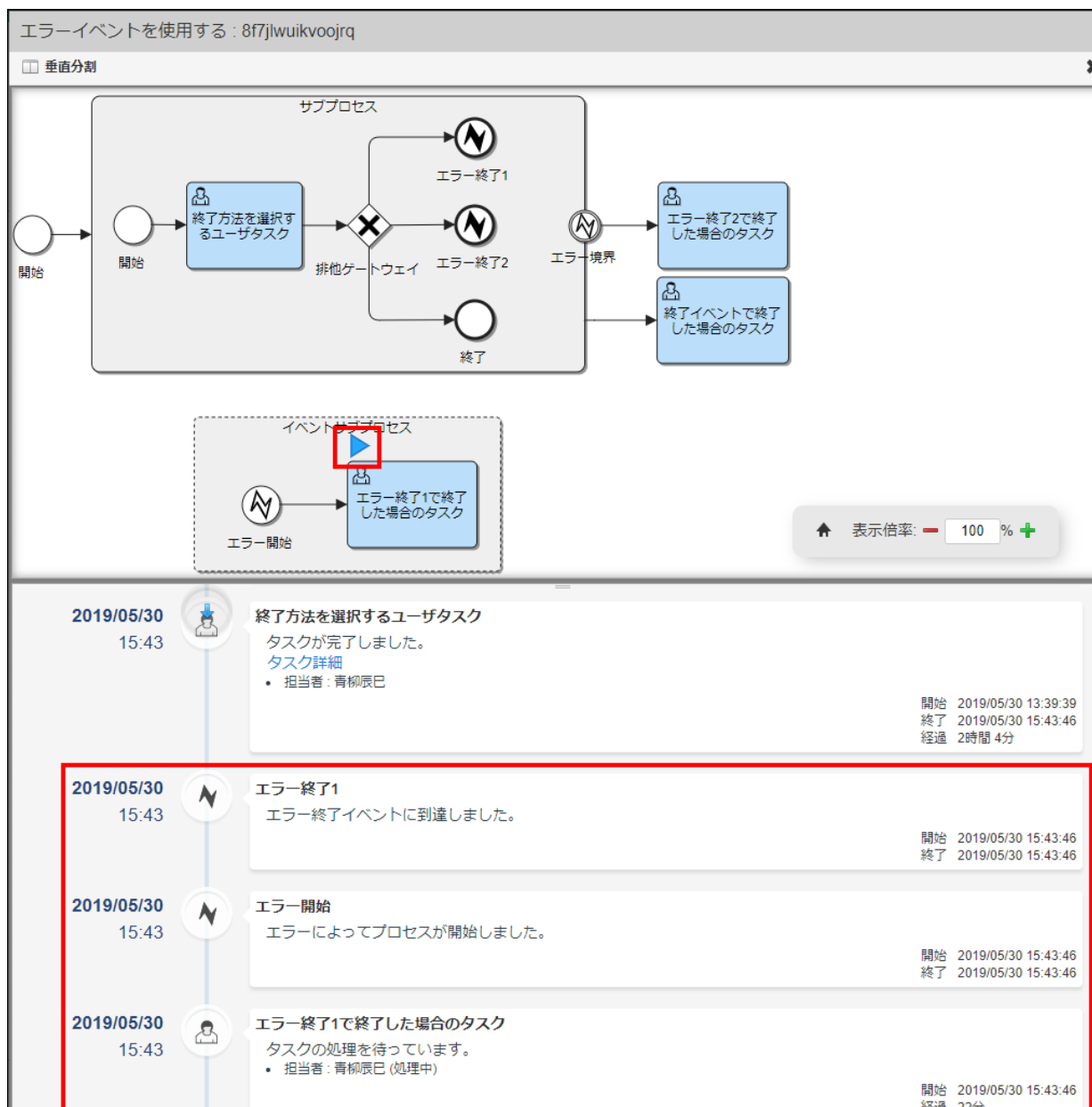


図: 「プロセス詳細画面」

「エラー境界イベント」の動作確認

「エラー終了イベント」から「エラー境界イベント」へ進行するプロセスの状況を確認します。

1. 「動作確認したい終了イベントを選択する」画面から「エラー終了2」を選択し、「決定」ボタンをクリックします。

動作確認したい終了イベントを選択する

排他ゲートウェイから進行する終了イベントを選択します。

エラー開始イベントの動作を確認する場合、「エラー終了1」を選択します。
 エラー境界イベントの動作を確認する場合、「エラー終了2」を選択します。
 終了イベントの動作を確認する場合、「終了」を選択します。

終了方法の選択

☐ エラー終了1
☒ **エラー終了2**
☐ 終了

決定

サブプロセス

エラー終了1

図：動作確認したい終了イベントを選択する(IM-FormaDesigner)

2. プロセスの状態を確認します。
「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
3. 「個人タスク」に対象のタスクがあることを確認します。
「」アイコンをクリックし、「プロセス詳細」画面を表示します。

個人タスク

検索条件

処理	履歴	参照	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当
				エラーイベントを使用する		エラー終了2で終了した場合のタスク	50	2019/05/30			

図：「タスク一覧」 - 「個人タスク」

4. プロセスの状態を確認します。
「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
5. 「プロセス参照」画面、または、「拡大表示」画面から、タスクが下記の状態であることを確認します。
 - ステータスが「実行中」であること
 - プロセス図で「エラー終了2で終了した場合のタスク」に「」がついていること
 - タイムラインで、「サブプロセス」が終了していること
「エラー終了2」から「エラー終了2で終了した場合のタスク」へ進行していること

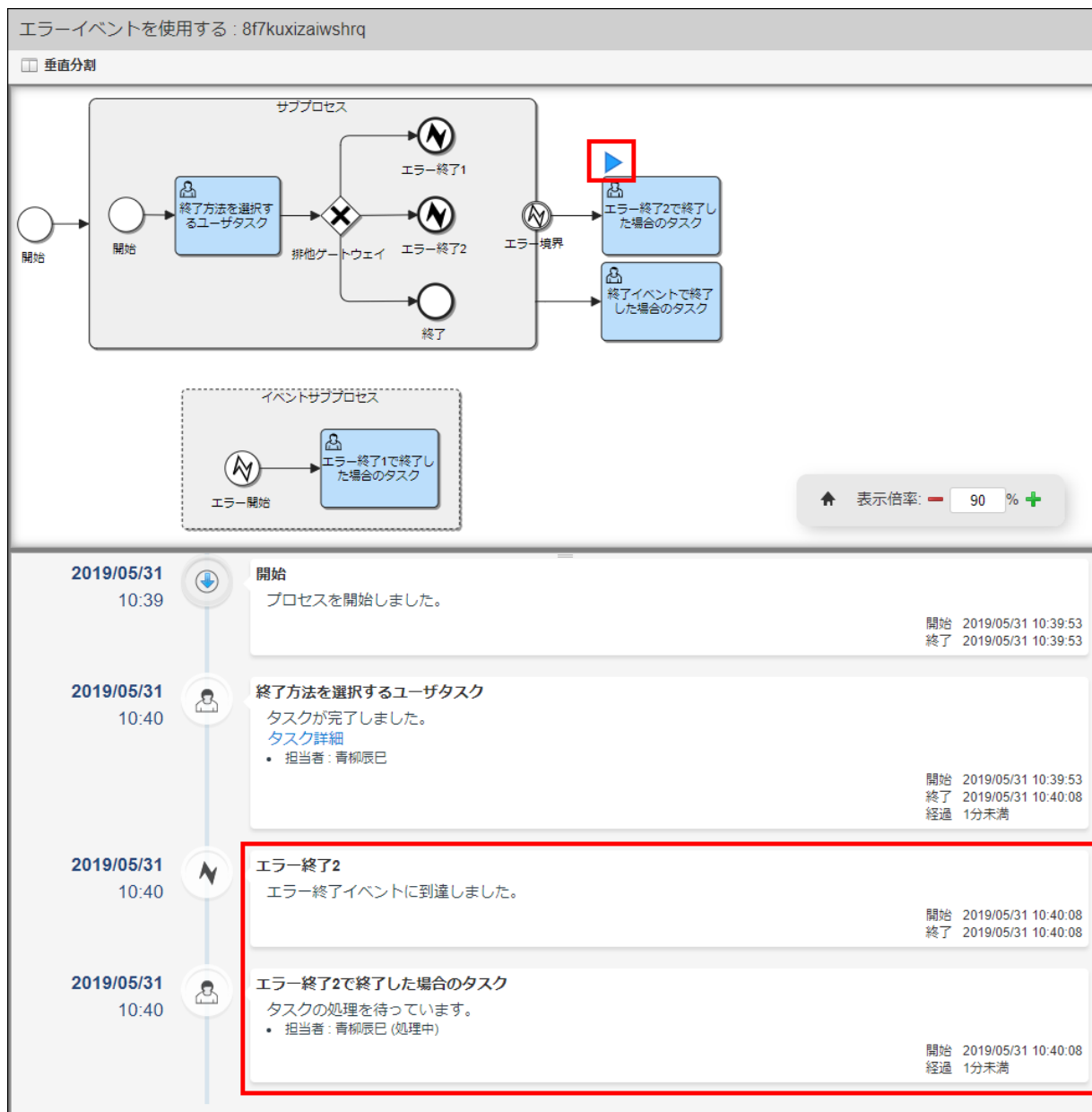


図: 「プロセス詳細画面」

「終了イベント」の動作確認

「終了イベント」に到達した場合、通常のプロセス同様サブプロセスが終了し、「サブプロセスで終了した場合のタスク」へ進行します。

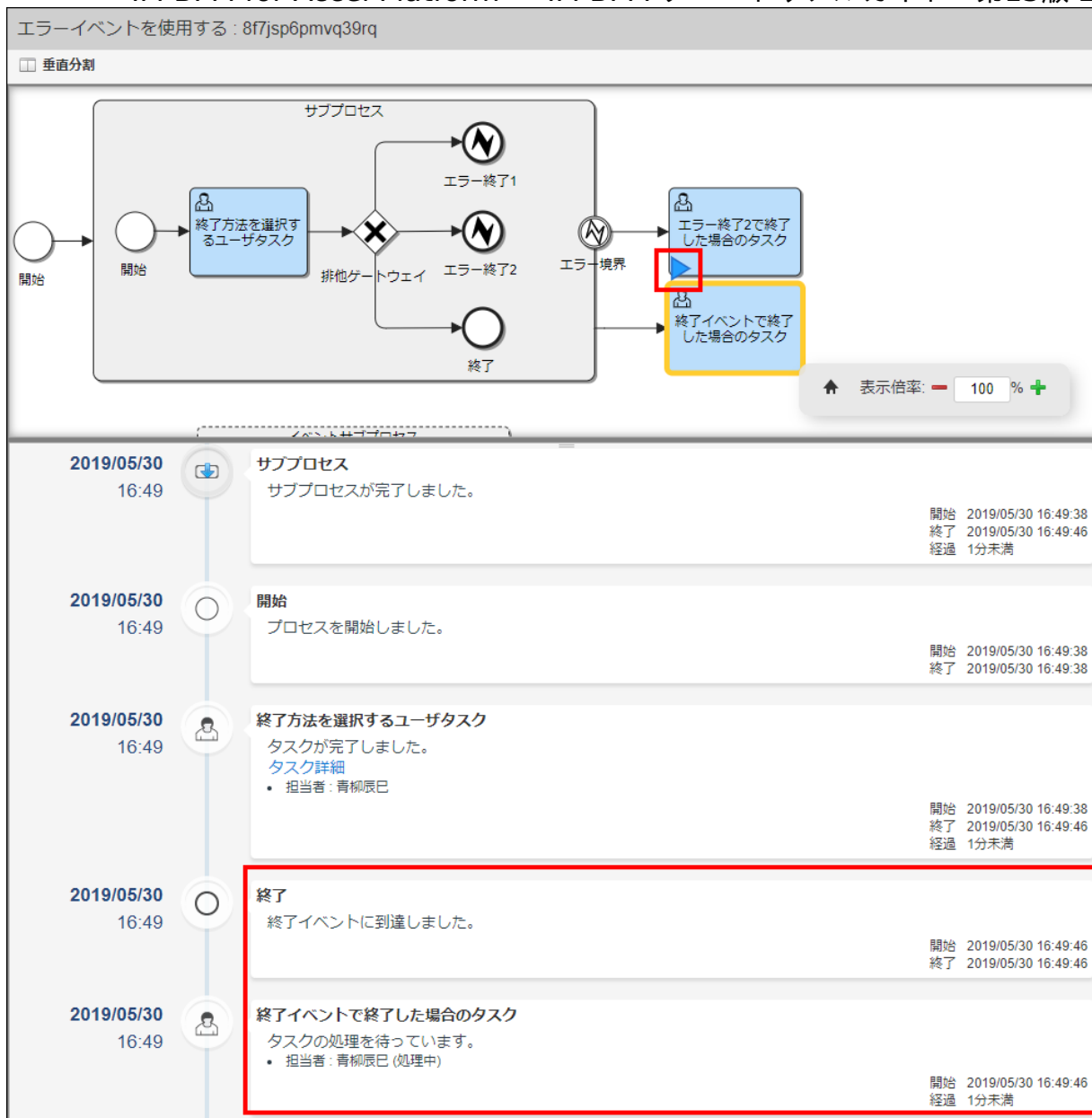


図: 「プロセス詳細画面」

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 プロセスデザイナー 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。

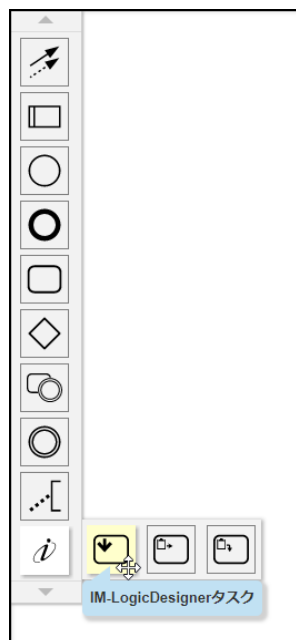
[logicdesigner_task_usage.bpmn](#)

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 プロセスデザイナー 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プロパティを定義する
- 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プロパティを定義する

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プロパティ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。

基本情報 **メインコンフィグ** 説明 リスナ マルチインスタンス 表現

フローID * ?

フロー定義検索
sample-accounts \${}

利用するバージョン ?

☐ 最新バージョンを利用 ☒ 入力したバージョンを利用

バージョン番号 * ?

1 \${}

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し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。

入力データ ?

+ 追加 選択済みの項目を削除

選択	編集	名前	値
☐		user_cd	\${user_cd}

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。

入力データ

名前 * ?

user_cd

値 ?

\${user_cd} \${}

決定 取り消し

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



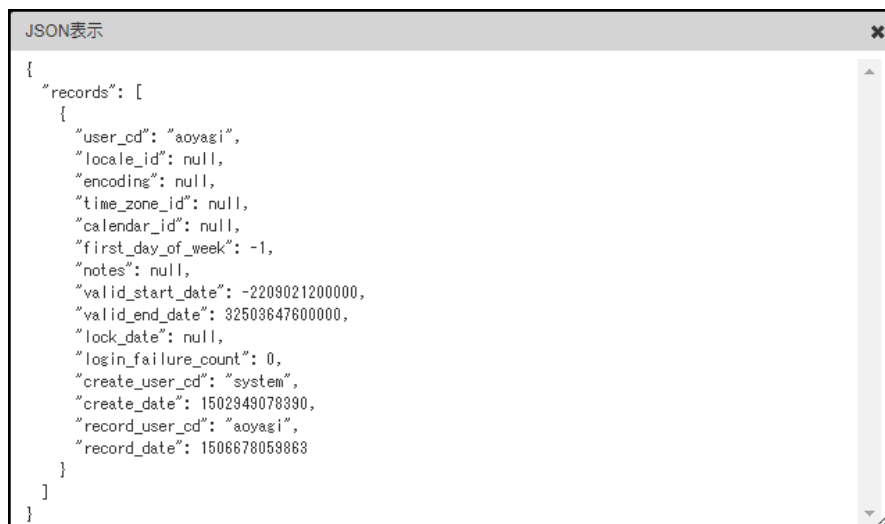
注意

「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 プロセスデザイナー 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。

i 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 プロセスデザイナー 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lement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

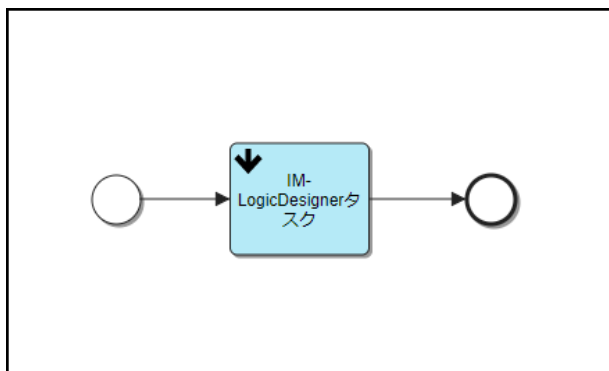
事前準備

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

i 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。

Element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

基本情報	メインコンフィグ	説明	リスナ	マルチインスタンス	表現								
フローID *													
フロー定義検索 \${dynamic_flow_id} \$()													
利用するバージョン													
☐ 最新バージョンを利用 ☒ 入力したバージョンを利用													
バージョン番号 *													
\${dynamic_flow_version} \$()													
入力データ													
+ 追加 選択済みの項目を削除 <table border="1"> <thead> <tr> <th>選択</th> <th>編集</th> <th>名前</th> <th>値</th> </tr> </thead> <tbody> <tr> <td>☒</td> <td></td> <td>結果変数を格納する</td> <td></td> </tr> </tbody> </table>						選択	編集	名前	値	☒		結果変数を格納する	
選択	編集	名前	値										
☒		結果変数を格納する											
結果変数名													
result_data													

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、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： <code>sample-accounts</code> 「Read intra-mart atom feed」を実行した場合： <code>sample-im-topics-to-log</code>
dynamic_flow_version	<code>1</code>
result_data	「List of Accounts」を実行した場合： ユーザ情報を持つ <code>records</code> 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 <code>topics</code> 配列



コラム

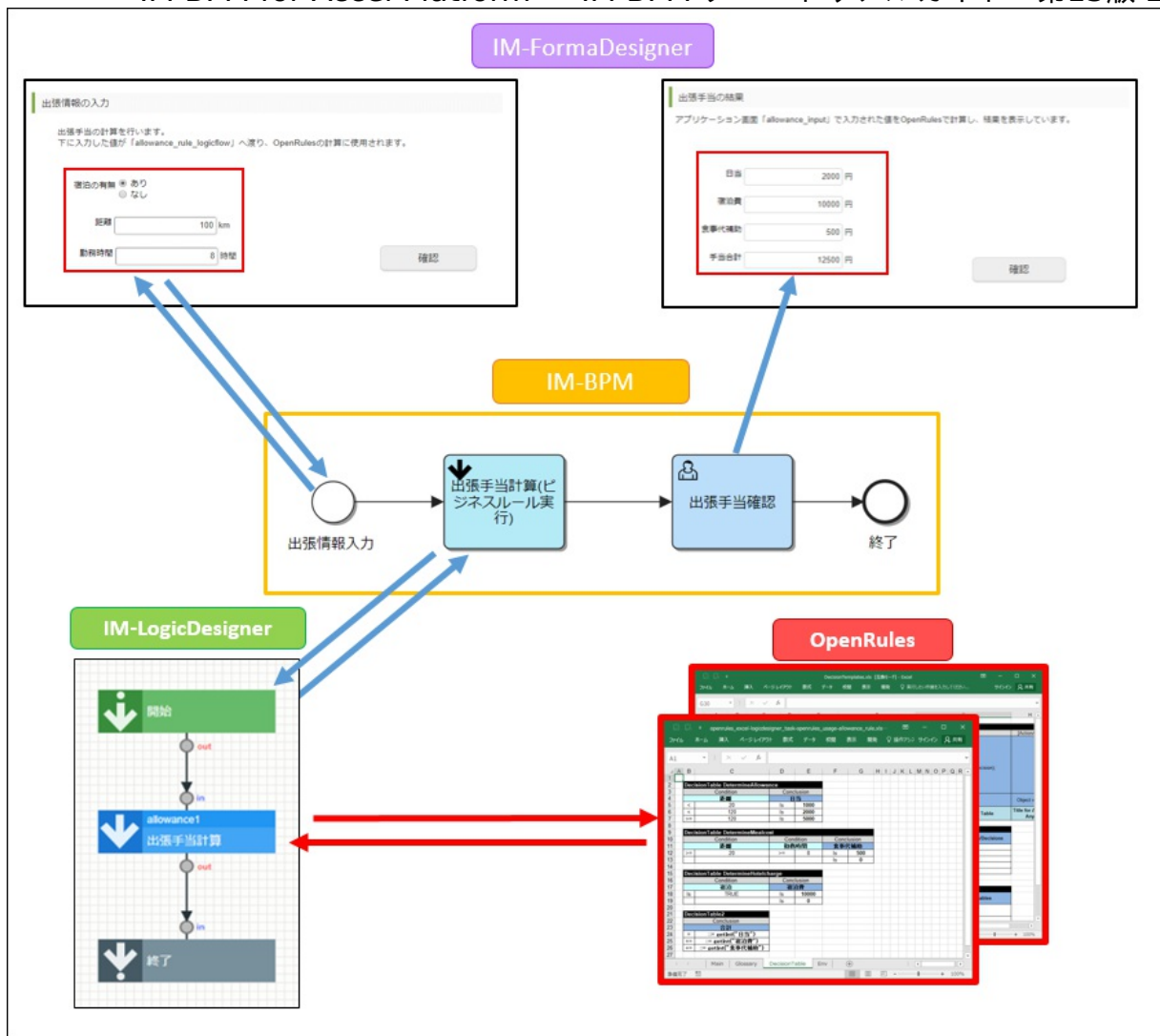
- プロセスのデプロイ方法については「[IM-BPM プロセスデザイナー 操作ガイド](#)」-「[プロセス定義のデプロイ](#)」を参照してください。
- デプロイしたプロセスの実行方法については「[IM-BPM ユーザ操作ガイド](#)」-「[プロセスインスタンスの開始](#)」を参照してください。
- 「プロセス詳細」画面への遷移方法については「[IM-BPM ユーザ操作ガイド](#)」-「[プロセスインスタンスを確認する](#)」を参照してください。
- 「変数一覧」画面の操作方法については「[IM-BPM ユーザ操作ガイド](#)」-「[プロセスインスタンスの変数を確認する](#)」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 ユーザーマニュアル](#)」を参照してください。
「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ー情報を取得する](#)」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。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Formaアプリケーションを使用します。

チュートリアルを開始する前に以下の資料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 プロセスデザイナー 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、環境構築をおこなってください。

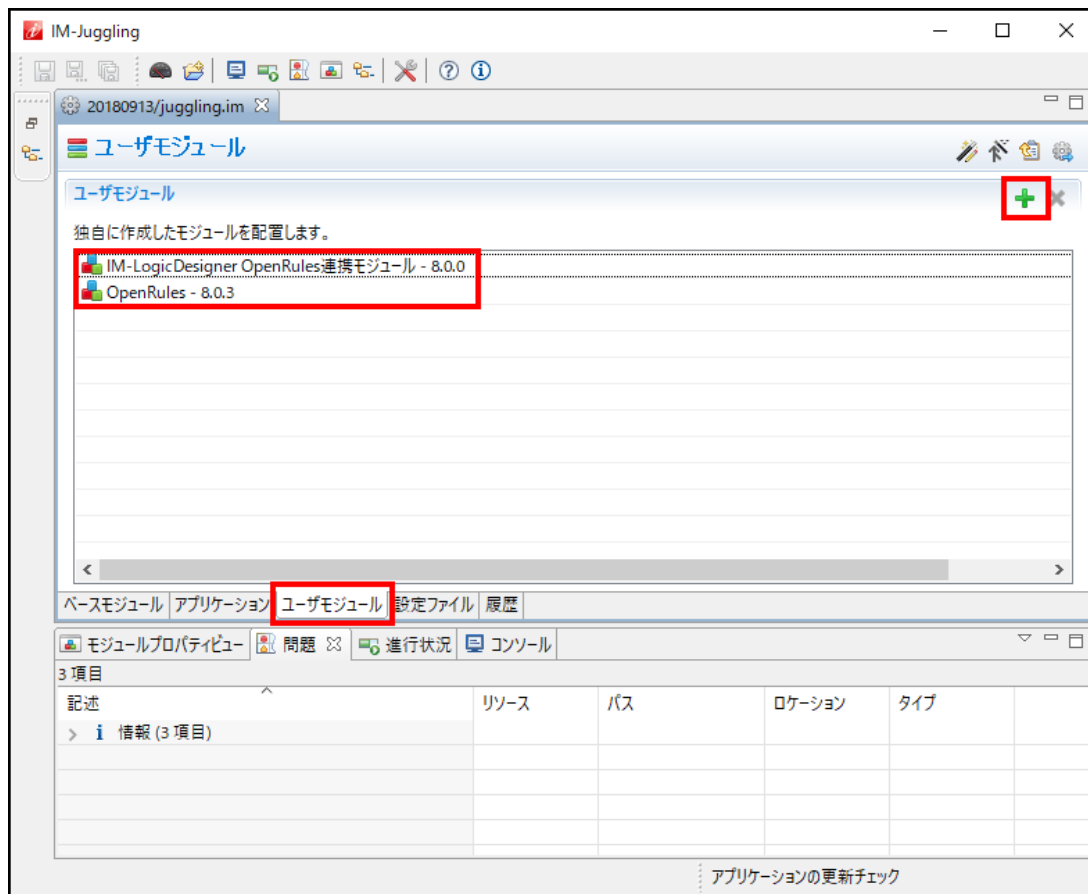
- モジュールを取得します。
「Product File Downloadサイト」から、プロダクトファイルダウンロード画面を開きます。
- OpenRules用のライセンスキーを入力し、ファイル一覧取得（Get FileList）をクリックします。
- 以下のユーザモジュールをダウンロードしてください。
 - openrules_modules_6.4.2.zip
openrules_modules_6.4.2.zipを解凍し、jp.co.intra_mart.oss_linkage.openrules-8.0.3.imm を取得します。
 - im_logic_openrules-8.0.0.imm
上記のバージョンを選択してください。
- 「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。

- 「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Formaアプリケーションに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。

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：「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Formaアプリケーションに入力された値を「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。

	A	B	C	D	E	F	G	H	I
1									
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					
11									
12		Data BusinessTripCondition businessTripCondition							
13		stay	distance	workinghours					
14		宿泊	距離	勤務時間					
15									
16		Data Allowance allowance							
17		allowance	hotelcharge	mealcost	totalamount				
18		日当	宿泊費	食事代補助	合計				
19									
20		Datatype BusinessTripCondition							
21		boolean	stay						
22		int	distance						
23		int	workinghours						
24									
25		Datatype Allowance							
26		int	allowance						
27		int	hotelcharge						
28		int	mealcost						
29		int	totalamount						
30									
31									

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Formaアプリケーションに入力され、「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Formaアプリケーションへ返却される表示結果の値です。

	A	B	C	D	E	F	G	H	I
1									
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					
11									
12		Data BusinessTripCondition businessTripCondition							
13		stay	distance	workinghours					
14		宿泊	距離	勤務時間					
15									
16		Data Allowance allowance							
17		allowance	hotelcharge	mealcost	totalamount				
18		日当	宿泊費	食事代補助	合計				
19									
20		Datatype BusinessTripCondition							
21		boolean	stay						
22		int	distance						
23		int	workinghours						
24									
25		Datatype Allowance							
26		int	allowance						
27		int	hotelcharge						
28		int	mealcost						
29		int	totalamount						
30									
31									

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。

	A	B	C	D	E	F	G	H	I
1									
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					
11									
12		Data BusinessTripCondition businessTripCondition							
13		stay	distance	workinghours					
14		宿泊	距離	勤務時間					
15									
16		Data Allowance allowance							
17		allowance	hotelcharge	mealcost	totalamount				
18		日当	宿泊費	食事代補助	合計				
19									
20		Datatype BusinessTripCondition							
21		boolean	stay						
22		int	distance						
23		int	workinghours						
24									
25		Datatype Allowance							
26		int	allowance						
27		int	hotelcharge						
28		int	mealcost						
29		int	totalamount						
30									
31									

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-logidesigner_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 操作アシスト サインイン 共有																		
A1																		
1																		
2		DecisionTable DetermineAllowance																
3		Condition		Conclusion														
4		距離		日当														
5		<	20	Is	1000													
6		<	120	Is	2000													
7		>=	120	Is	5000													
8																		
9		DecisionTable DetermineMealcost																
10		Condition		Condition	Conclusion													
11		距離		勤務時間	食事代補助													
12		>=	20	>=	8	Is	500											
13						Is	0											
14																		
15		DecisionTable DetermineHotelcharge																
16		Condition		Conclusion														
17		宿泊		宿泊費														
18		Is	TRUE	Is	10000													
19				Is	0													
20																		
21		DecisionTable2																
22		Conclusion																
23		合計																
24		=	:= getInt("日当")															
25		+=	:= getInt("宿泊費")															
26		+=	:= getInt("食事代補助")															
27																		

図：「openrules_excel-logidesigner_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	Is	1000
<	120	Is	2000
>=	120	Is	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			Conclusion		
勤務時間			食事代補助		
>=	20	>=	8	Is	500
				Is	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	
Is	TRUE	Is	1000
		Is	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。

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：「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

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

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。

A	B	C	D	E	F	G	H	I	J	K	L	M
1												
2	Environment											
3	include	/im_logic/openrules.config/DecisionTemplates.xls										
4												

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 ユーザーマニュアル」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



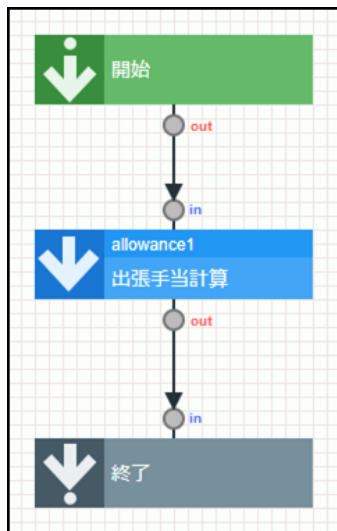
コラム

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Formaアプリケーションに入力された値を「OpenRules」で計算し、結果を返します。



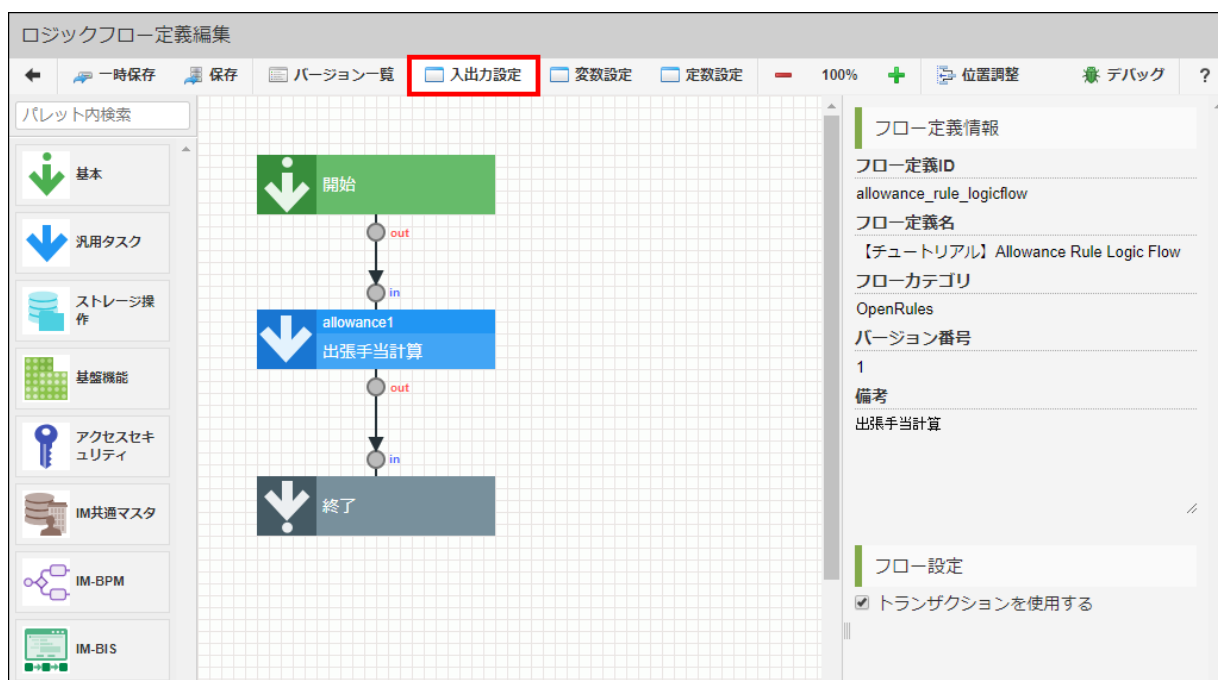
図：ロジックフロー

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

ロジックフロー定義一覧				
新規作成 カテゴリー一覧				
検索条件				
編集	フロー定義ID	フロー定義名	フローカテゴリ	呼出元
	allowance_rule_logicflow	【チュートリアル】 Allowance Rule Logic Flow	OpenRules	
	sample-accounts	List of Accounts	Sample	
	sample-backup-authz-data	Backup authz data files	Sample	
	sample-im-topics-to-log	Read intra-mart atom feed	Sample	

図：「ロジックフロー定義一覧」

- 「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

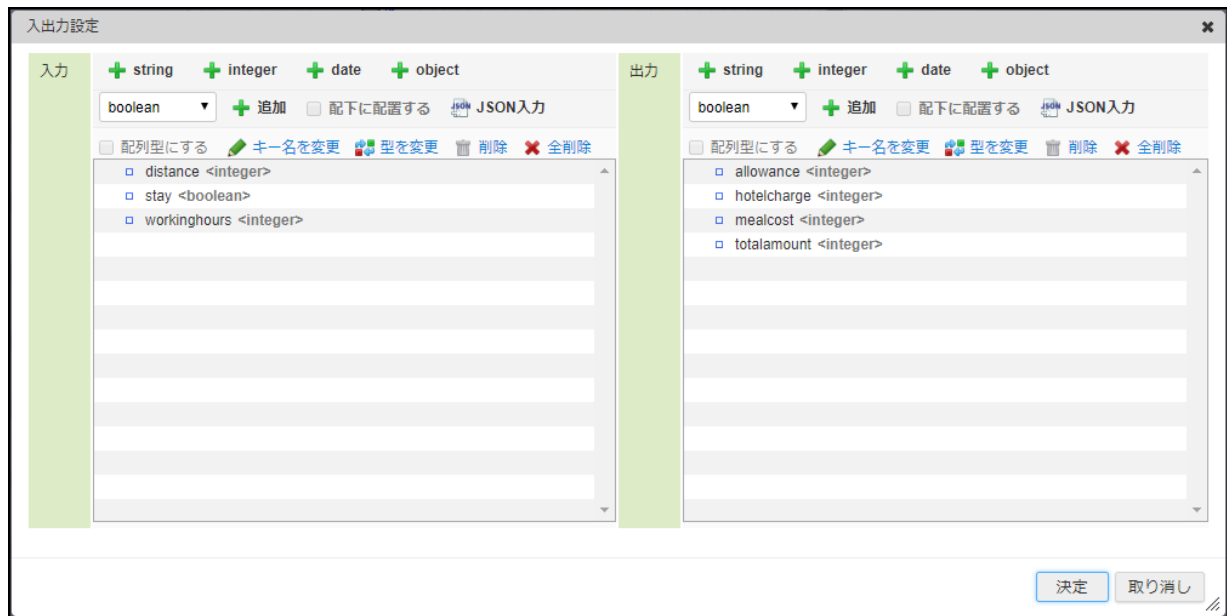
「IM-FormaDesigner」のFormaアプリケーション「allowance_input」で入力され、「IM-BPM」から渡ってきた値

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

- 出力

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

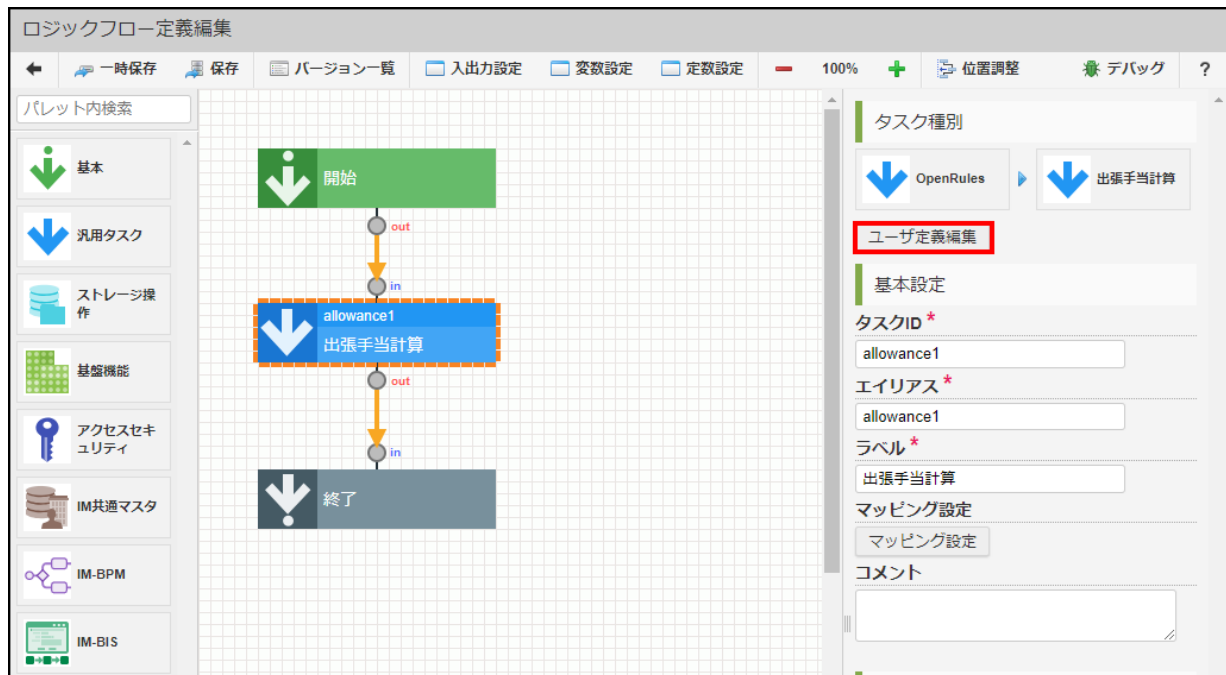
キー名	型
allowance	<integer>
hotelcharge	<integer>
mealcost	<integer>
totalamount	<integer>



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

OpenRules ルール定義編集

ユーザ定義共通設定

ユーザ定義ID * ☐ ユーザ定義IDを新しく割り当てて複製する
allowance

バージョン番号 * 1

ユーザ定義名 *
標準 * 出張手当計算
日本語
英語
中国語 (中華人民共和國)

ユーザカテゴリ *
検索 新規作成
ユーザカテゴリID * openrules
ユーザカテゴリ名 OpenRules

ソート番号 * 100

アイコン

入力値
+ string + integer + date + object
boolean 追加 配下に配置する
JSON入力
配列型にする キー名を変更 型を変更
削除 全削除
BusinessTripCondition <object>
distance <integer>
stay <boolean>
workinghours <integer>

返却値
+ string + integer + date + object
boolean 追加 配下に配置する
JSON入力
配列型にする キー名を変更 型を変更
削除 全削除
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

更新 新バージョン登録

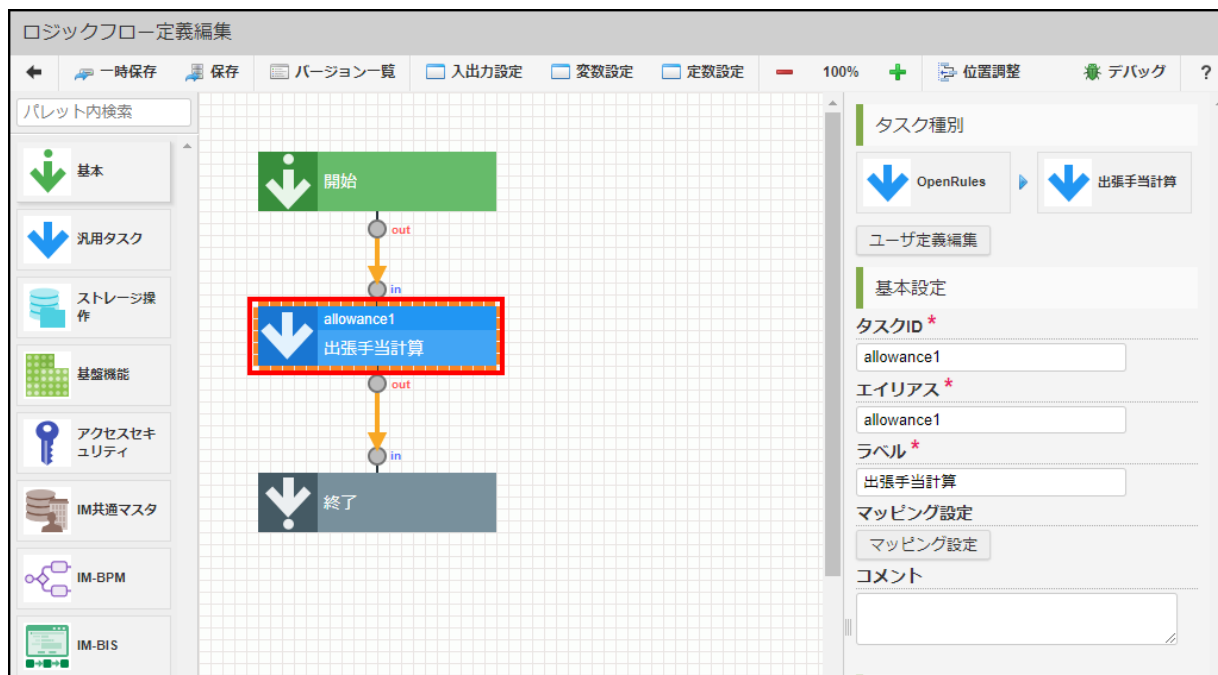
図：「OpenRulesルール定義編集」



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。

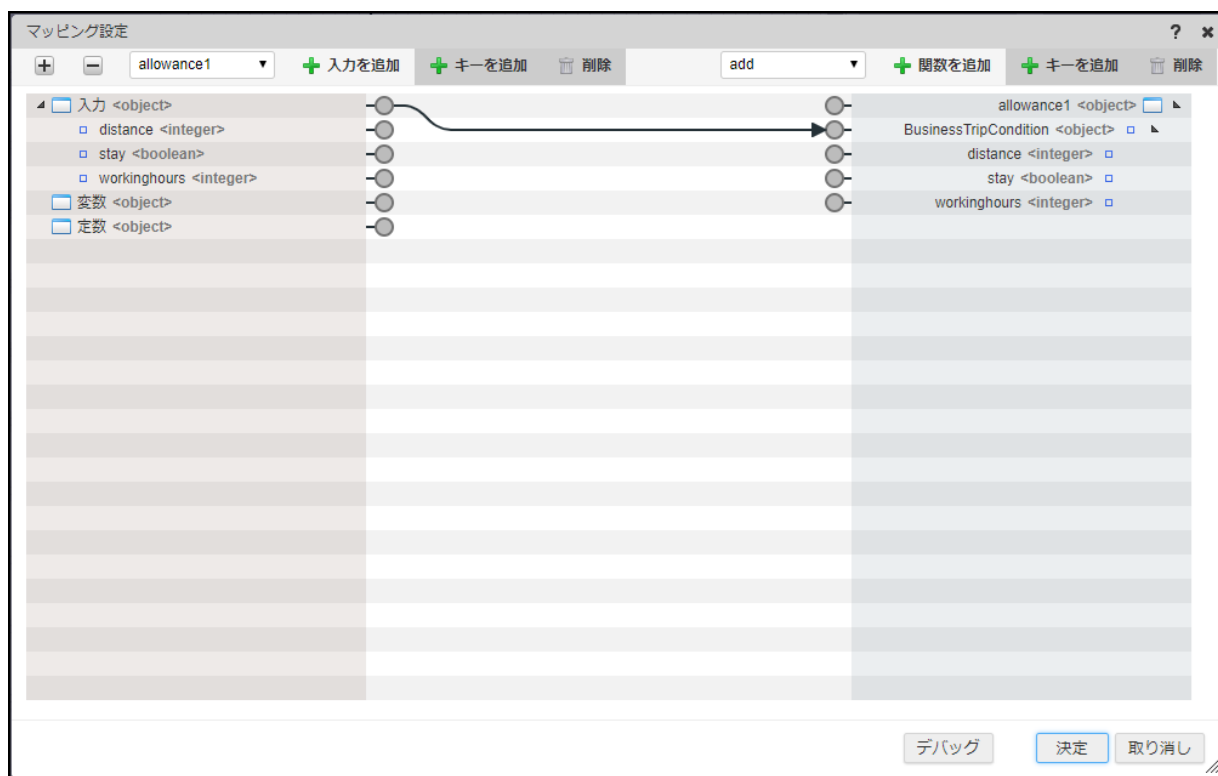
7. 「OpenRules」を使用するタスク「出張手当計算（allowance1）」のマッピング設定を確認します。
「出張手当計算（allowance1）」タスクをクリックしてください。



図：「ロジックフロー定義編集」

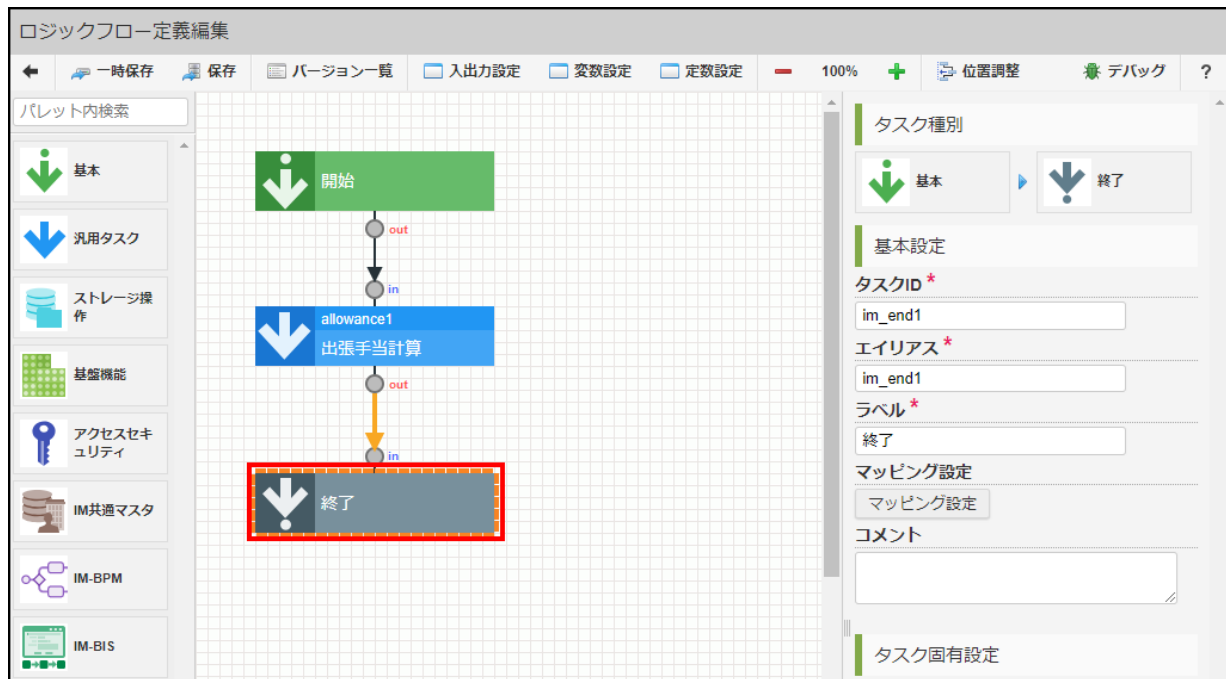
8. 左右で以下の値が登録・マッピングされていることを確認します。

- 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：BusinessTripCondition<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」

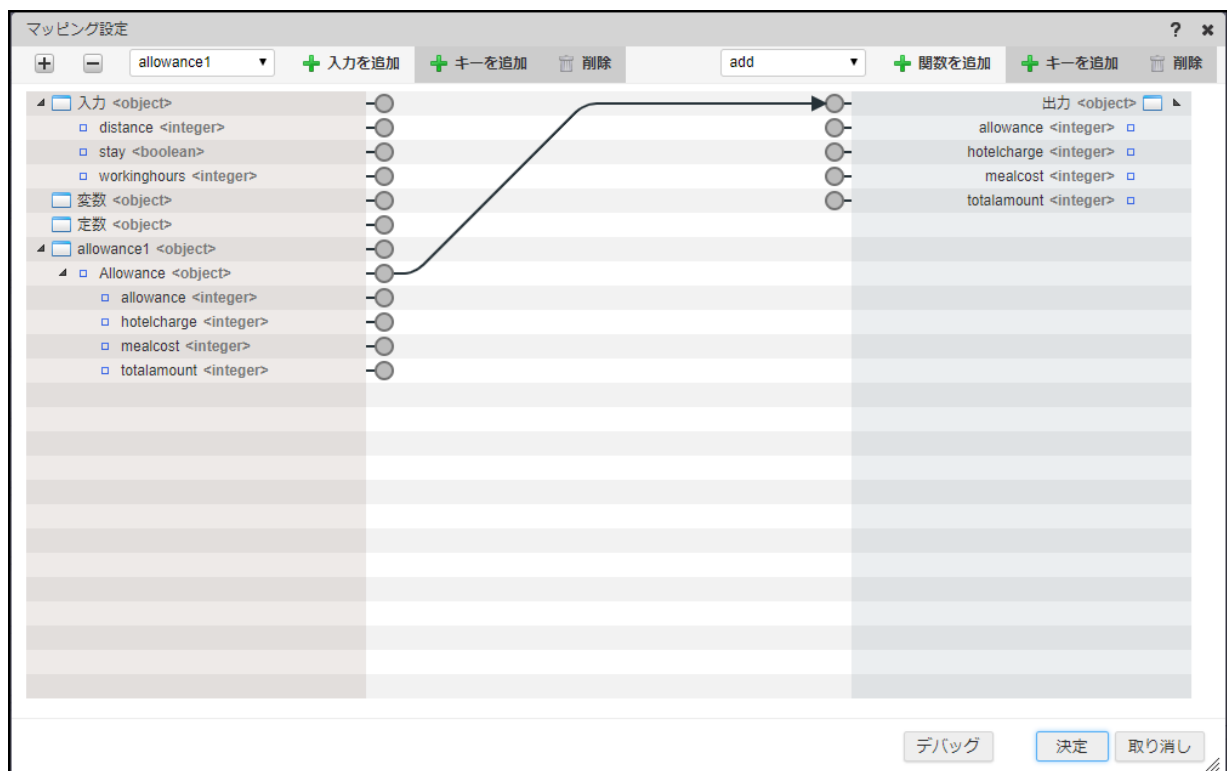
9. 「終了」タスクの「マッピング設定」画面を表示します。
「終了」タスクをクリックしてください。



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。

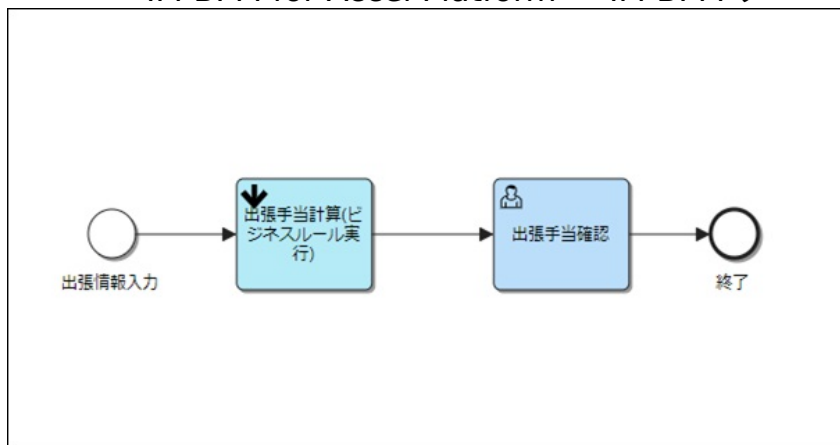


図：「マッピング設定」

プロセス定義を作成する

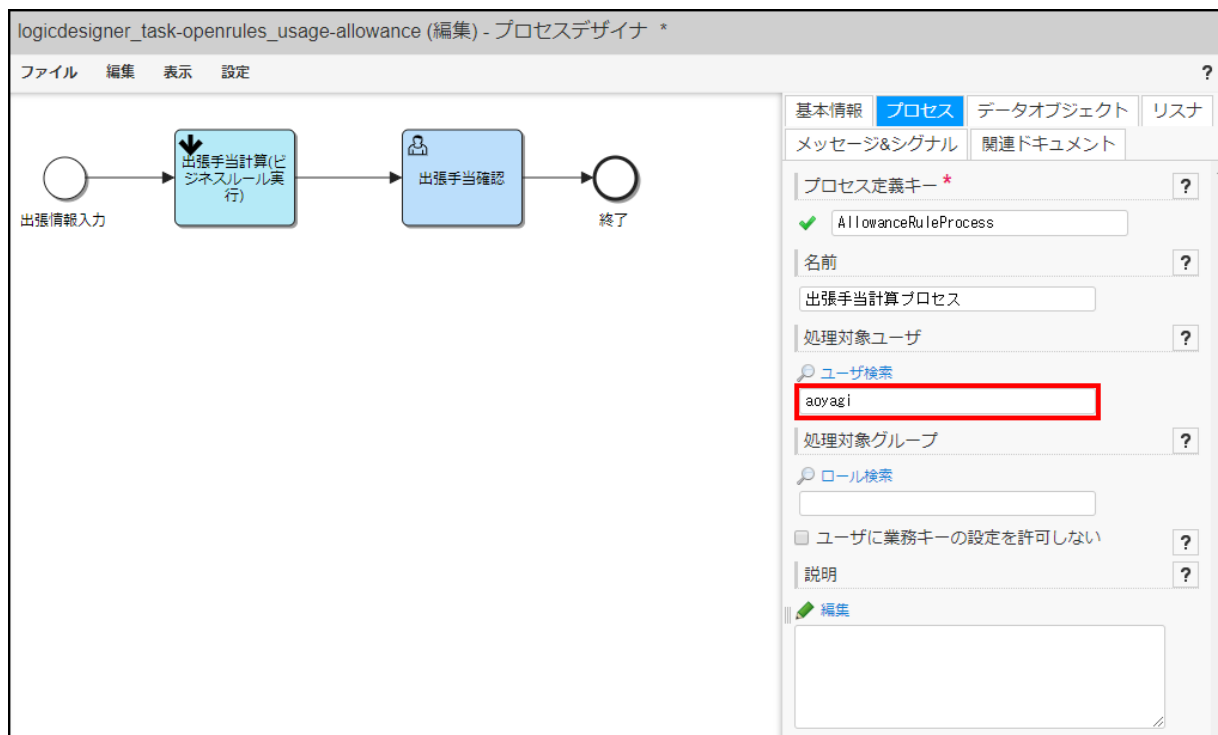
上記の「IM-FormaDesigner」や「IM-LogicDesigner」を組み込んだプロセスを作成します。

このプロセスは、「IM-FormaDesigner」のFormaアプリケーションに「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。



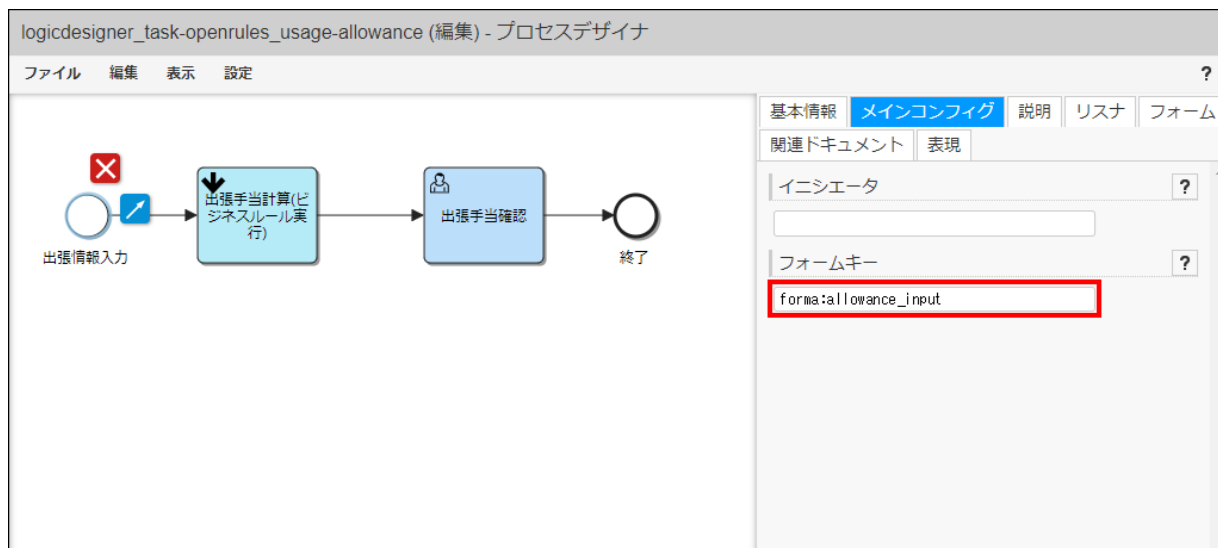
図：完成イメージ

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、処理対象ユーザーを「青柳辰巳（ユーザコード: aoyagi）」に設定します。



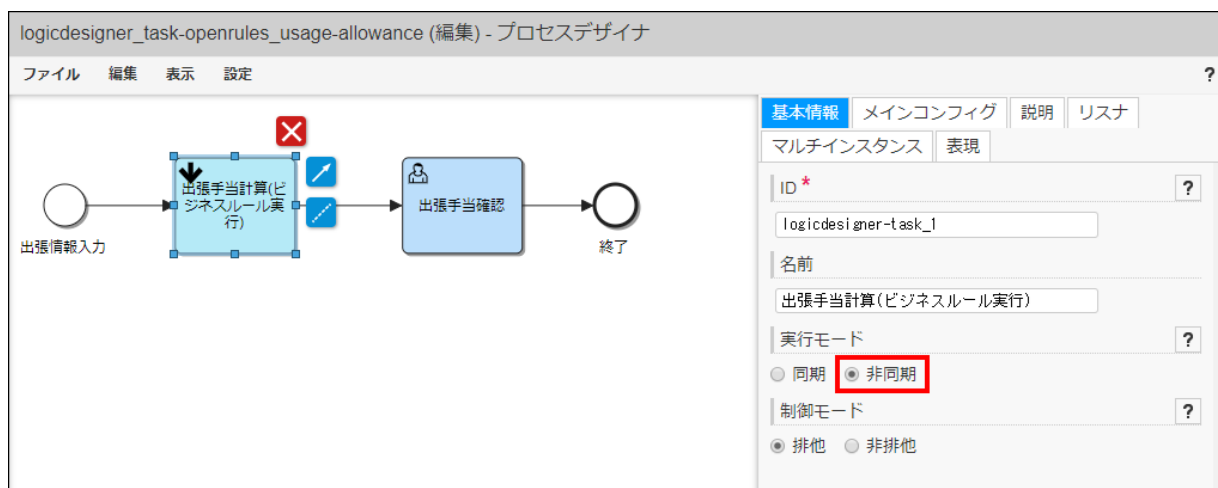
図：「プロパティ」 - 「プロセス」

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。

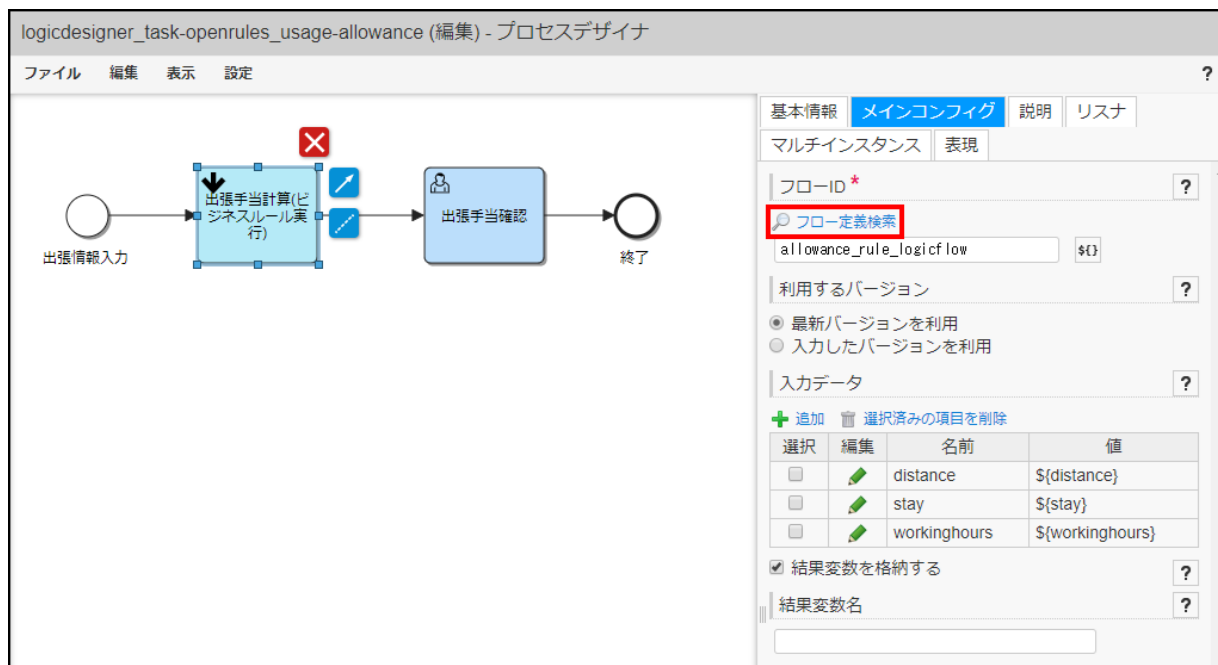


図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「基本情報」

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。

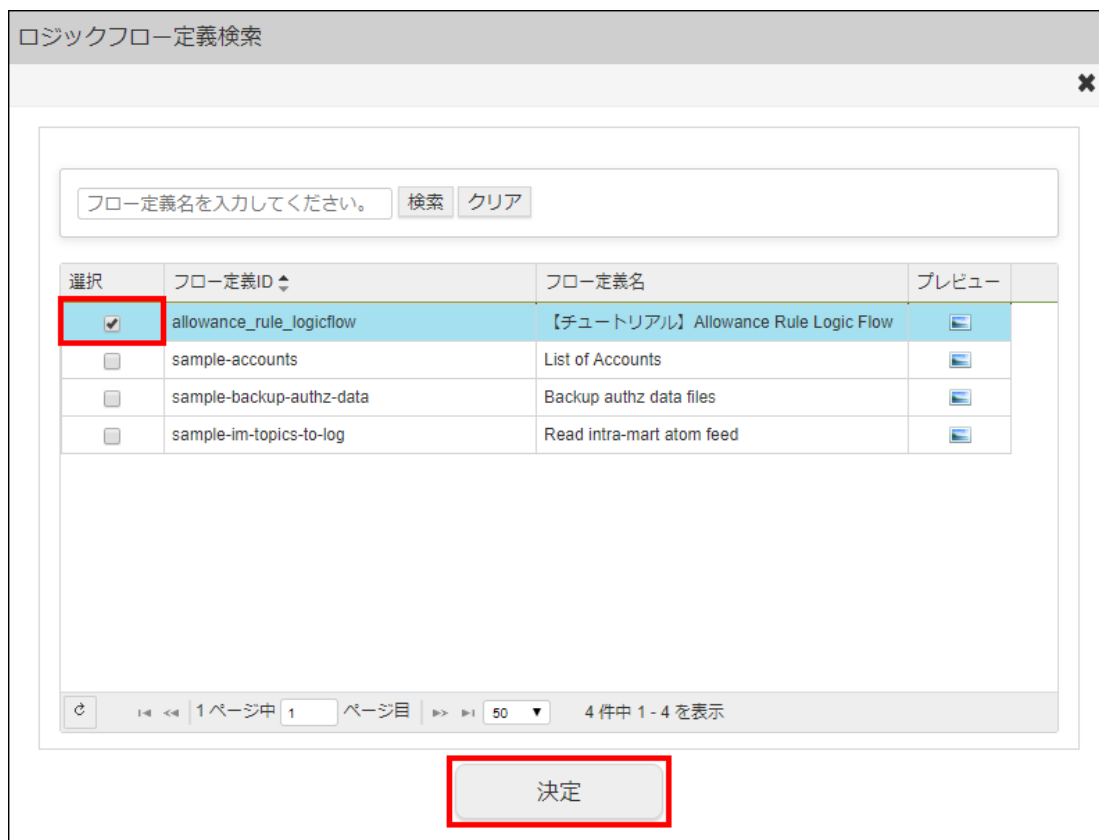
7. 「フローID」を設定します。
「メインコンフィグ」タブの「フローID」から、「フロー定義検索」をクリックしてください。



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

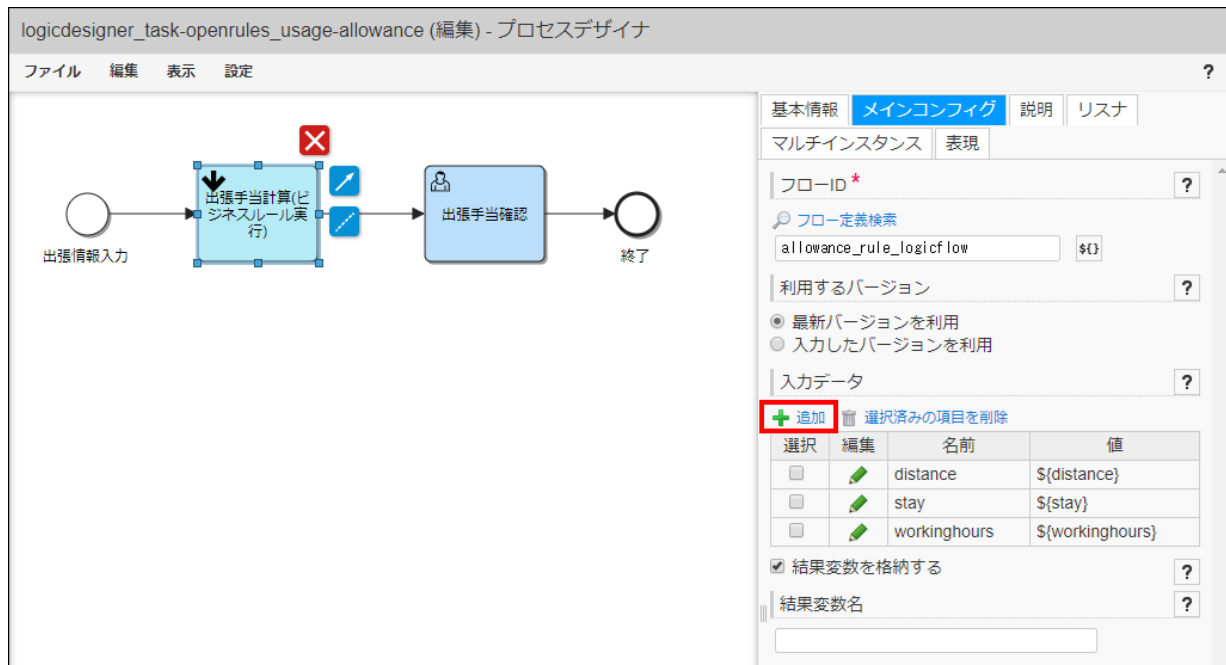
8. チュートリアル開始前にインポートしたものを使用します。

「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」

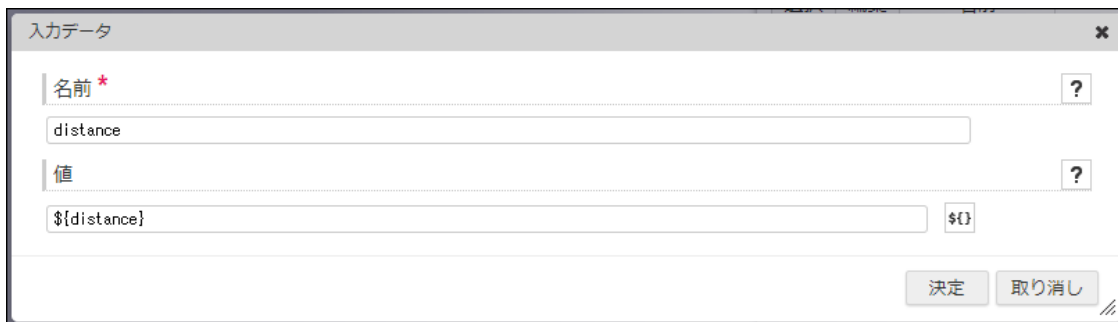
9. 「IM-FormaDesigner」のFormaアプリケーションに入力した値を「IM-LogicDesigner」の入力値に紐づける際に使用するデータを設定します。
「メインコンフィグ」タブから、「入力データ」の「追加」をクリックします。



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

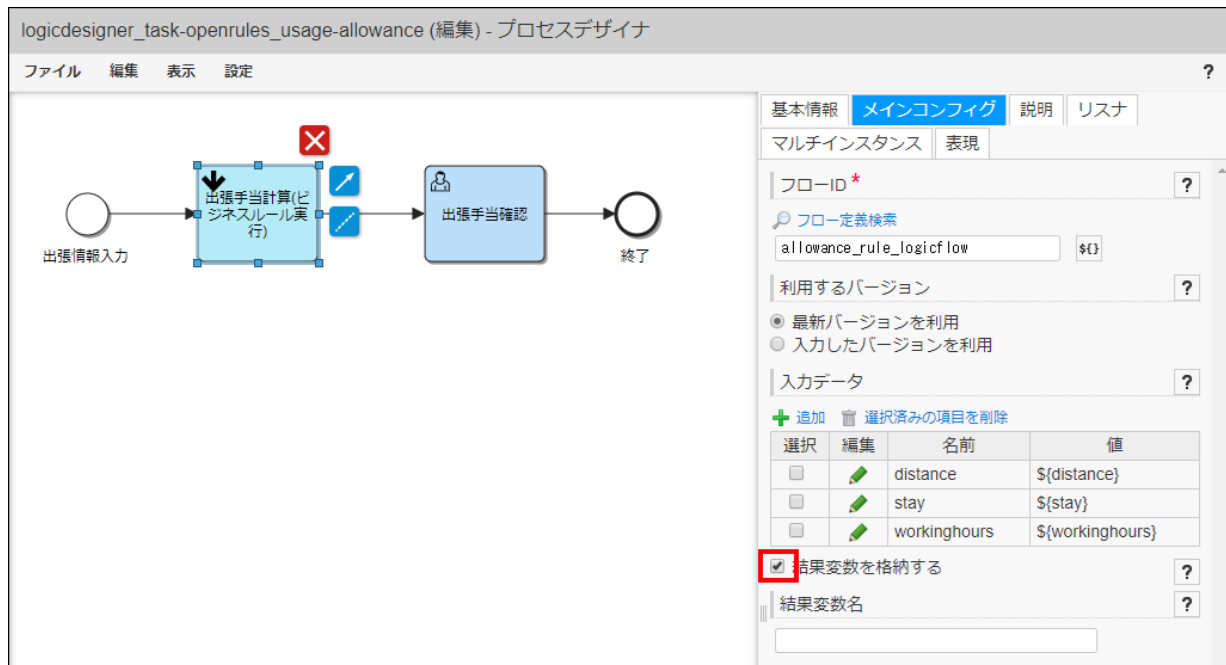
10. 以下のとおりに項目の値を設定してください。

- 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。

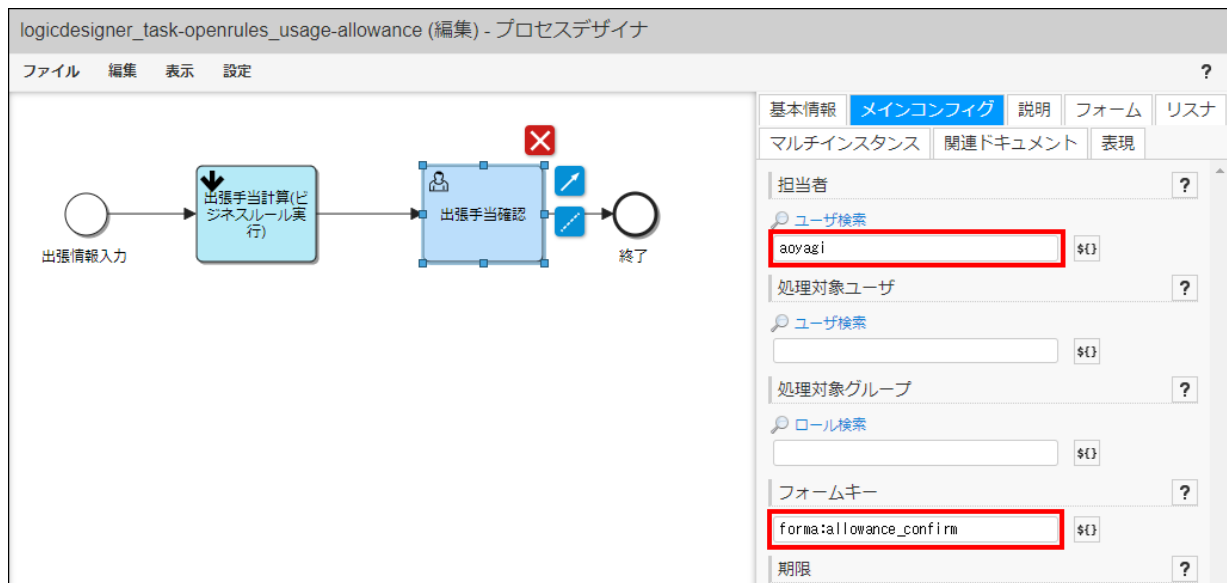


図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

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。

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

- 開始イベントに設定したFormaアプリケーション「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：Formaアプリケーション「allowance_input」

- 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。
- 「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：Formaアプリケーション「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 プロセスデザイナー 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。

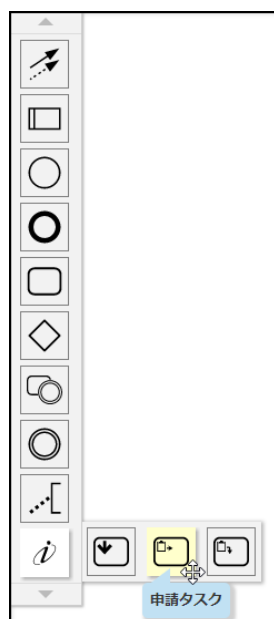
[apply_task_usage.bpmn](#)

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 プロセスデザイナー 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。

基本情報 メインコンフィグ 説明 リスナ 表現

アプリケーション ?

IM-Workflow

前処理ユーザプログラムのクラス名 ?

フローID *

フロー定義検索

flow_javaee_01

案件名 *

申請タスクチュートリアル

申請基準日 ?

申請実行者コード *

ユーザ検索

aoyagi

申請権限者コード *

ユーザ検索

aoyagi

タスク終了時の起票案件の操作 ?

terminate

☐ 参照可能な変数を継承する ?

入力データ ?

+ 追加 - 選択済みの項目を削除

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

案件の処理結果を格納する変数名 ?

workflowResult

☒ 案件のユーザデータを格納する ?

案件のユーザデータを格納する変数名 ?

workflowData

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
<code>item_name</code>	サンプル商品
<code>item_amount</code>	2
<code>item_price</code>	10000
<code>item_comment</code>	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。



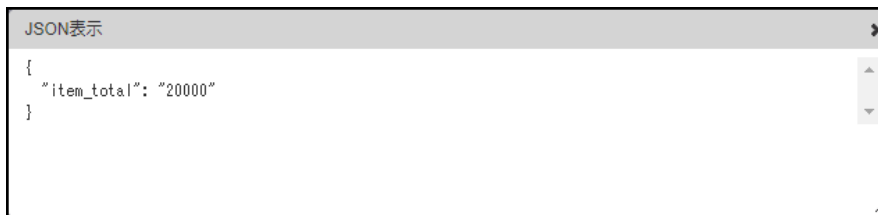
```

{
  "lastNodeName": "SampleDivision01",
  "flowVersionId": "flow_javaee_01_1",
  "lastProcessNodeId": "route_01_02^temp_02",
  "lastExecUserCd": "maruyama",
  "locale": "ja",
  "actFlag": "0",
  "contentsId": "contents_javaee",
  "userDataId": "ma_8elh69arxphmq8d",
  "lastResultStatus": "mattercomplete",
  "routeId": "route_sample_01",
  "contentsVersionId": "contents_javaee_1",
  "lastNodeType": "g",
  "processDate": "2017/11/20",
  "routeVersionId": "route_sample_01_1",
  "lastAuthUserCd": "maruyama",
  "systemMatterId": "ma_8elh69arxphmq8d",
  "flowId": "flow_javaee_01"
}

```

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ーデータが格納されている変数 `workflowData` の内容を確認します。



```

{
  "item_total": "20000"
}

```

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 プロセスデザイナー 操作ガイド](#)」 - 「[起票タスク](#)」もあわせて参照してください。

i 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。

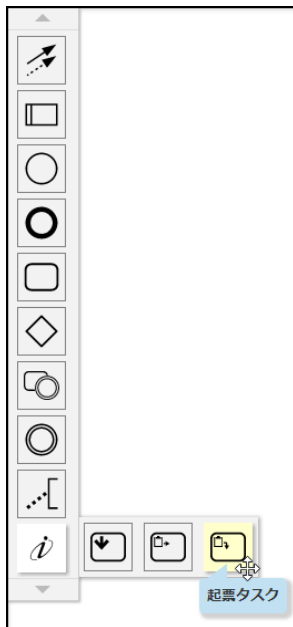
[draft_task_usage.bpmn](#)

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 プロセスデザイナー 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。

基本情報	メインコンフィグ	説明	リスナ	表現
アプリケーション				
IM-Workflow				
前処理ユーザプログラムのクラス名				
フローID *				
flow_javaee_01				
ユーザデータID				
案件番号				
案件名 *				
起票タスクチュートリアル				
申請基準日				
起票者コード				
aoyagi				
優先度				
タスク終了時の起票案件の操作				
terminate				
案件の処理結果を格納する変数名				
workflowResult				
☒ 案件のユーザデータを格納する				
案件のユーザデータを格納する変数名				
workflowData				

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。

フロー定義 - 検索			
検索条件			
選択	フローID	フロー名	備考
☐	5lbgqyp05zqr6a5	【サンプル】 サンプルユーザ情報変更申請	サンプルユーザ情報の変更を申請します。
☐	flow_javaee_01	直線ルート[JavaEE開発モデル]	
☐	flow_javaee_02	横配置ルート[JavaEE開発モデル]	
☐	flow_javaee_03	縦配置ルート[JavaEE開発モデル]	
☐	flow_javaee_04	分岐ルート[JavaEE開発モデル]	
☐	flow_javaee_05	複合ルート[JavaEE開発モデル]	
☐	flow_script_01	直線ルート[スクリプト開発モデル]	
☐	flow_script_02	横配置ルート[スクリプト開発モデル]	
検索 戻る リセット 1 ページ中 1 ページ目 15 11 件中 1 - 11 を表示			

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。


```

JSON表示
{
  "lastNodeName": "SampleDivision01",
  "flowVersionId": "flow_javaee_01_1",
  "lastProcessNodeId": "route_01_02^temp_02",
  "lastExecUserCd": "maruyama",
  "locale": "ja",
  "actFlag": "0",
  "contentsId": "contents_javaee",
  "userDataId": "ma_8elh69arxphmq8d",
  "lastResultStatus": "mattercomplete",
  "routeId": "route_sample_01",
  "contentsVersionId": "contents_javaee_1",
  "lastNodeType": "g",
  "processDate": "2017/11/20",
  "routeVersionId": "route_sample_01_1",
  "lastAuthUserCd": "maruyama",
  "systemMatterId": "ma_8elh69arxphmq8d",
  "flowId": "flow_javaee_01"
}

```

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ーデータが格納されている変数 `workflowData` の内容を確認します。

```

JSON表示
{
  "item_total": "20000"
}

```

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。

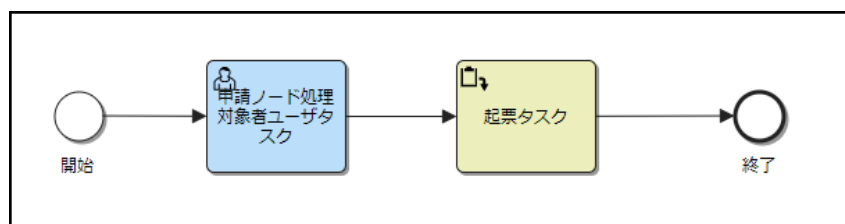
アップロード手順は「[IM-BPM プロセスデザイナー 操作ガイド](#)」-「[プロセス定義のアップロード](#)」を参照してください。

- 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。

基本情報 | **メインコンフィグ** | 説明 | リスナ | フォーム | 関連ドキュメント | 表現

イニシエータ ?

starter

フォームキー ?

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。

基本情報 | **メインコンフィグ** | 説明 | フォーム | リスナ | マルチインスタンス

関連ドキュメント | 表現

ID * ?

draft_task_before_user_task

名前

申請ノード処理対象者ユーザタスク \${}

別名 ?

{}

実行モード ?

☒ 同期 ☐ 非同期

制御モード ?

☒ 排他 ☐ 非排他

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。

基本情報 | **メインコンフィグ** | 説明 | フォーム | リスナ | マルチインスタンス

関連ドキュメント | 表現

担当者 ?

ユーザ検索

{}

処理対象ユーザ ?

ユーザ検索

{}

処理対象グループ ?

ロール検索

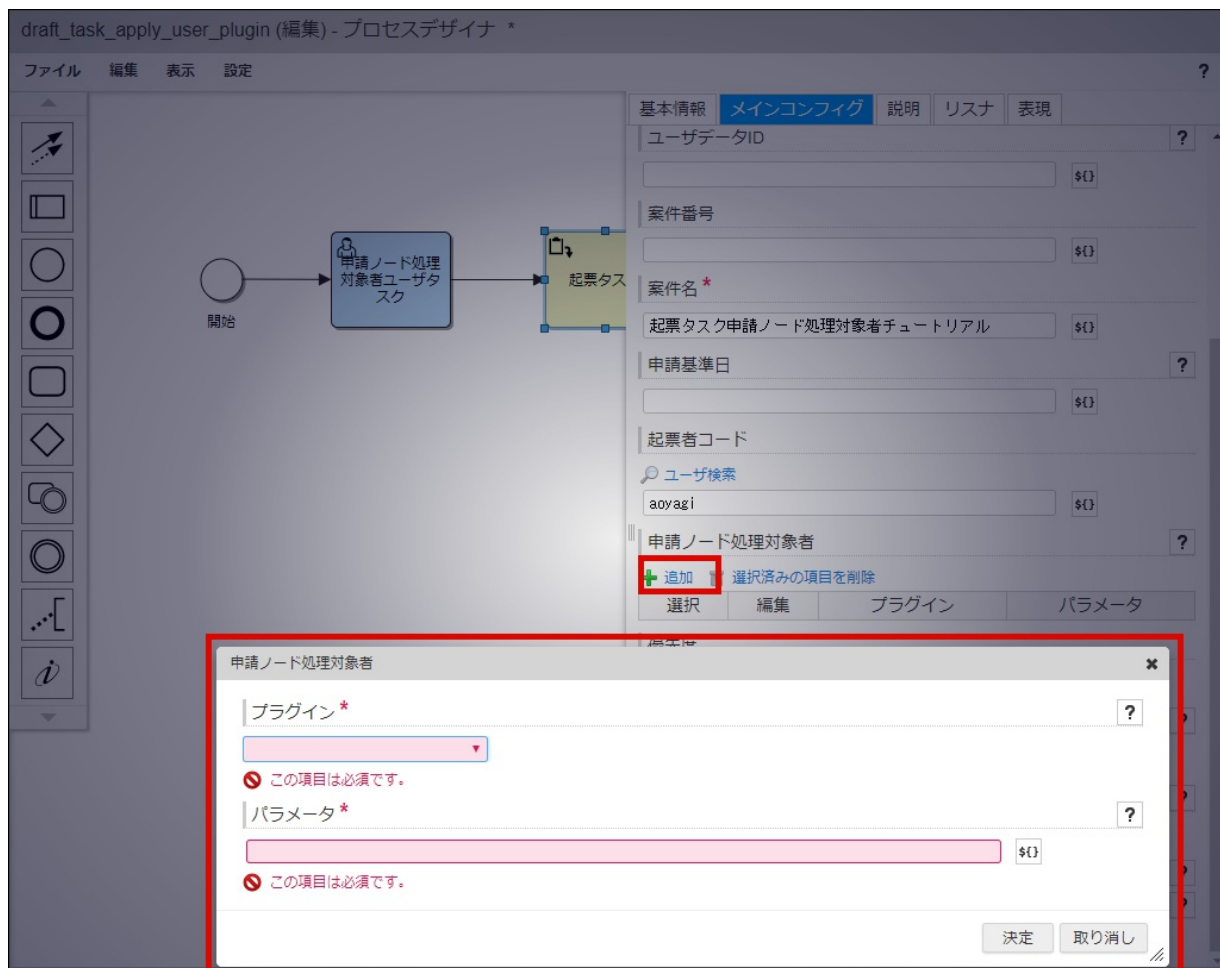
im_bpm_user \${}

フォームキー ?

{}

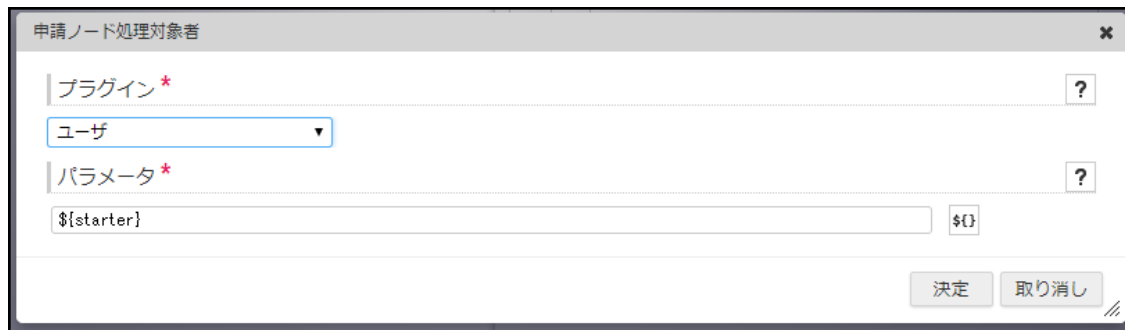
図：ユーザタスク：プロパティ - メインコンフィグ

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}`

基本情報 メインコンフィグ 説明 リスナ 表現

アプリケーション ?

IM-Workflow

前処理ユーザプログラムのクラス名 ?

フローID *

フロー定義検索

flow_javaee_01

ユーザデータID ?

案件番号

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

申請基準日 ?

起票者コード

ユーザ検索

aoyagi

申請ノード処理対象者 ?

+ 追加 選択済みの項目を削除

選択	編集	プラグイン	パラメータ
☐		jp.co.intra_mart.workflow.plugin.authority.node.apply.user	\${starter}
☐		jp.co.intra_mart.workflow.plugin.authority.node.apply.user	\${im_bpm_system_variables.im_operation_users.draft_task_before_user_task}

優先度

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。

タスク一覧

プロセス開始一覧 グループタスク一覧 個人タスク一覧 処理済一覧 振り分け 一括処理

グループタスク

検索条件

履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	表示設定
☐	起票タスク申請ノ			申請ノード処理対象	50	2018/03/16 18:			

図：担当にする

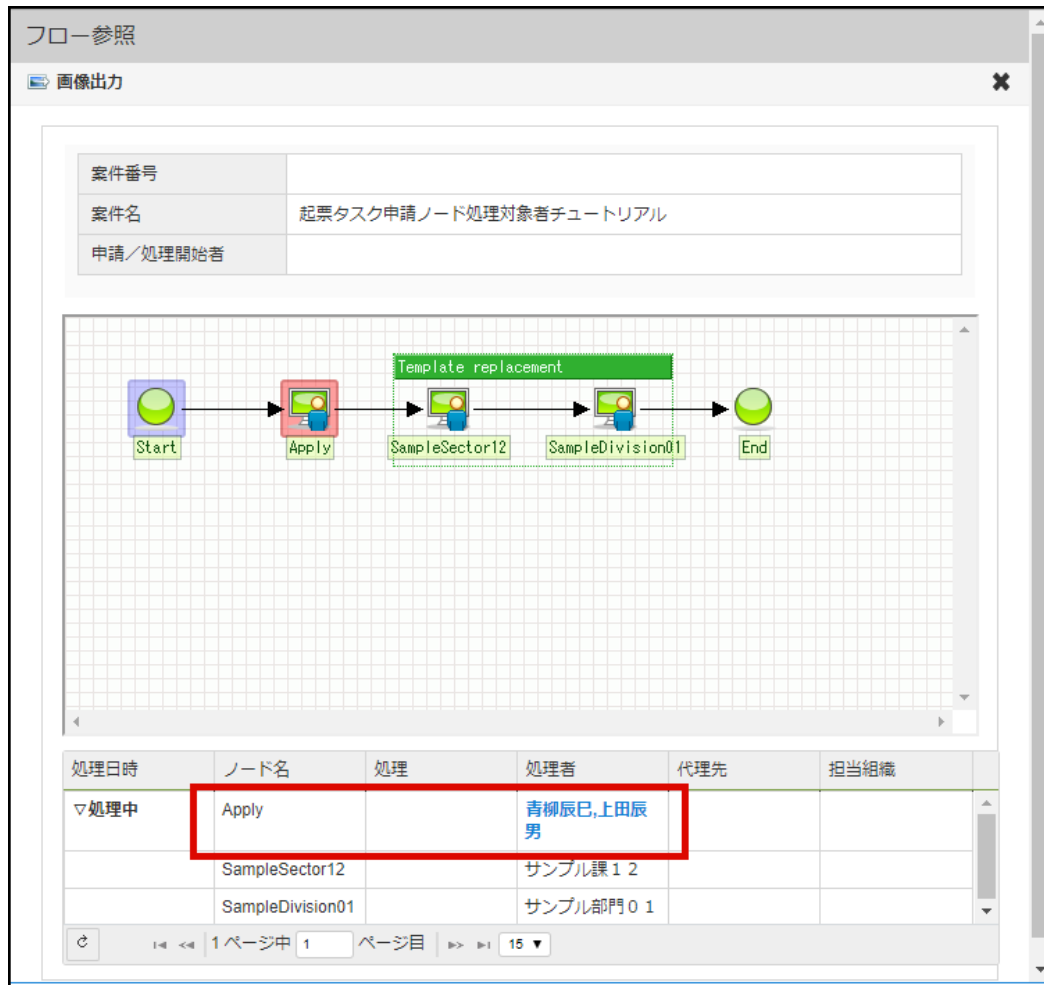
個人タスク

検索条件

処理	履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	表示設定
☐		起票タスク申請ノ			申請ノード処理対象	50	2018/03/16 18:			

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。

i 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。

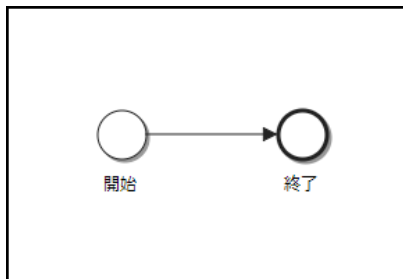
[start_event_with_forma.bpmn](#)

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 プロセスデザイナー 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lement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

Element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

基本情報 | **メインコンフィグ** | 説明 | リスナ | フォーム | 関連ドキュメント

表現

イニシエータ ?

フォームキー ?

forma:sample_app_equipment

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。

i 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。

i 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 プロセスデザイナー 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。

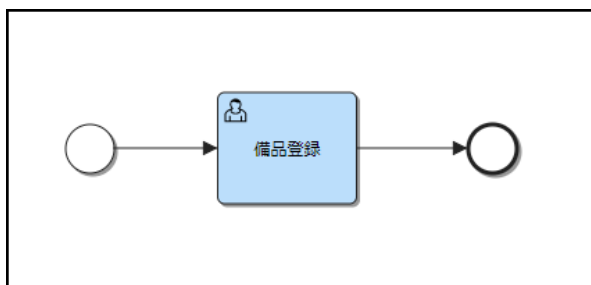
[user_task_with_forma.bpmn](#)

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 プロセスデザイナー 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lement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

Element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

基本情報	メインコンフィグ	説明	フォーム	リスナ	マルチインスタンス
関連ドキュメント 表現					
担当者		?			
ユーザ検索					
処理対象ユーザ		?			
ユーザ検索					
処理対象グループ		?			
ロール検索					
im_bpm_user					
フォームキー		?			
forma:sample_app_equipment					

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。

i 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。

選択	編集	ID	名前	型	値
☐		equipment_name	equipment_name	string	コピー用紙
☐		maker	maker	string	intra-mart製紙

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。

i コラム

プロセスグローバルのデータプロパティ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。

登録・更新する備品の情報を入力してください。*の項目は入力必須です。

備品名 *	コピー用紙
ロケール *	
メーカー	intra-mart製紙
購入単位 *	個
最低数量 *	1

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。

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

i コラム

- プロセスのデプロイ方法については「[IM-BPM プロセスデザイナー 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、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](#)

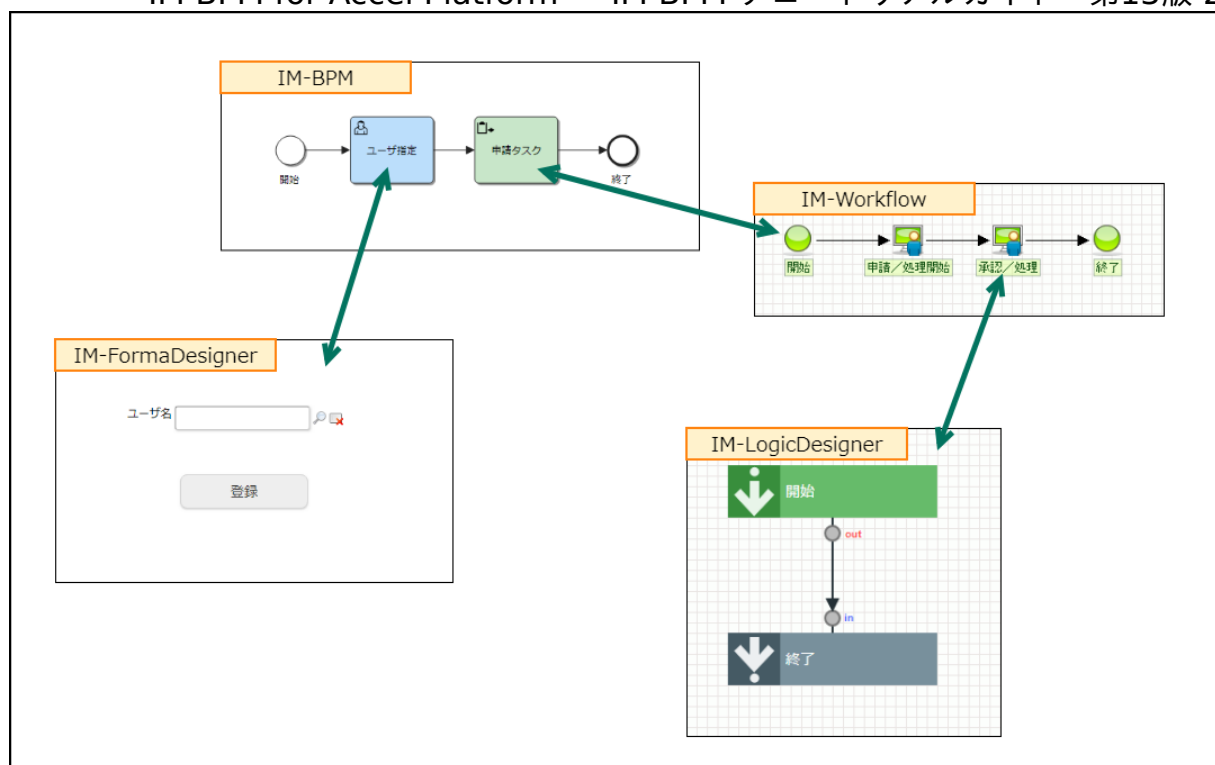
ワークフローとの連携

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_id.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 プロセスデザイナー 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」

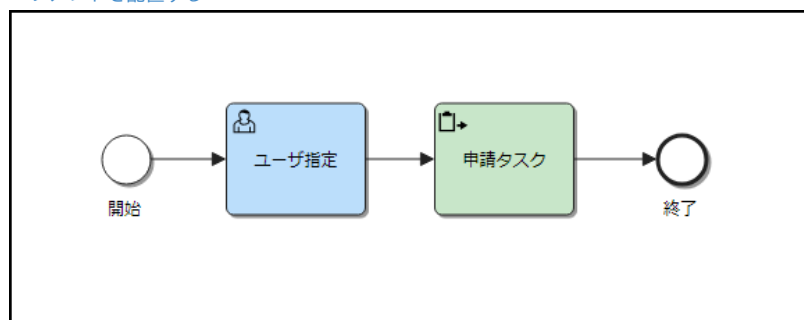
インポート、または、アップロードは、上記の順番で実行してください。
ロジックフロー定義のインポートよりも前に、ロジックフロー管理テーブルデータやルート定義のインポートを行った場合、正しくデータが連携しません。

- プロセス定義を作成する
 - エlement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
 - Element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。

Element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

基本情報	メインコンフィグ	説明	フォーム	リスナ
マルチインスタンス 関連ドキュメント 表現				
ID * select_target ?				
名前 ユーザ指定 #{}				
別名 #{} ?				
実行モード ☒ 同期 ☐ 非同期 ?				
制御モード ☒ 排他 ☐ 非排他 ?				
担当者 ? ユーザ検索 #{}				
処理対象ユーザ ? ユーザ検索 #{}				
処理対象グループ ? ロール検索 im_bpm_manager #{}				
フォームキー ? form:app_select_target #{}				
期限 ?				

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

The screenshot shows the 'Main Configuration' tab of the 'Properties' window for a 'User Task'. The form is organized into sections with expandable/collapsible headers. The 'Application' section is expanded, showing a dropdown menu with 'IM-Workflow' selected. Below it, the 'Pre-processing user program class name' field is empty. The 'Flow ID' section is expanded, showing a search icon and the text 'フロー定義検索', with the field containing 'flow_target_Id'. The 'User data ID' section is expanded, showing an empty field. The 'Case number' section is expanded, showing an empty field. The 'Case name' section is expanded, showing the text '承認ノード処理対象者チュートリアル'. The 'Application start date' section is expanded, showing an empty field. The 'Application executor code' section is expanded, showing a search icon and the text 'ユーザ検索', with the field containing 'aoyagi'. The 'Application authority code' section is expanded, showing a search icon and the text 'ユーザ検索', with the field containing 'aoyagi'. The 'Authority organization' section is partially visible at the bottom.

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

アプリケーション登録

アプリケーションID *

アプリケーション種別 *

有効日付(開始) *

有効日付(終了)

対象ロケール 日本語
対象ロケールの追加

一覧表示タイプ ViewCreatorの一覧を利用する ▼

少なくとも一つのロケール情報を設定してください。

日本語

アプリケーション名 *

備考

登録

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

ユーザ名

登録

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・デザイナーの詳細については、「[IM-FormaDesigner 作成者操作ガイド](#)」-「[「フォーム・デザイナー」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

- 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。
- 「テーブル設定」タブをクリックします。
- 「登録」をクリックします。

項目名	列名	データ型	データサイズ	データサイズ(小数部)
データ登録ID	imfr_sd_insert_id	文字列	20	
アプリケーションID	imfr_sd_application_id	文字列	100	
アプリケーション履歴番号	imfr_sd_application_no	数値	10	0
バージョン	imfr_sd_version_no	数値	10	0
登録日	imfr_sd_create_date	タイムスタンプ		
登録者ユーザコード	imfr_sd_create_user_cd	文字列	100	
更新日	imfr_sd_record_date	タイムスタンプ		
更新者ユーザコード	imfr_sd_record_user_cd	文字列	100	
一時保存フラグ	imfr_sd_preserve_flag	文字列	1	
ユーザコード	imfr_ud_usercd1	文字列	100	

図：「テーブル設定」

- 「ユーザコード」項目のデータサイズに `100` を入力します。
- 「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

- 「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。
- 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。
- 「権限設定」タブをクリックします。
- 「+ロール」をクリックします。
- 「追加」をクリックします。
- 「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。
- 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。



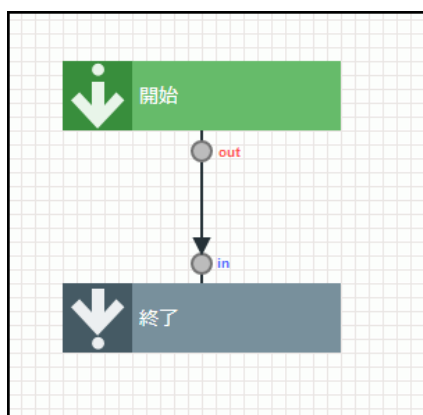
図：「権限設定」

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。

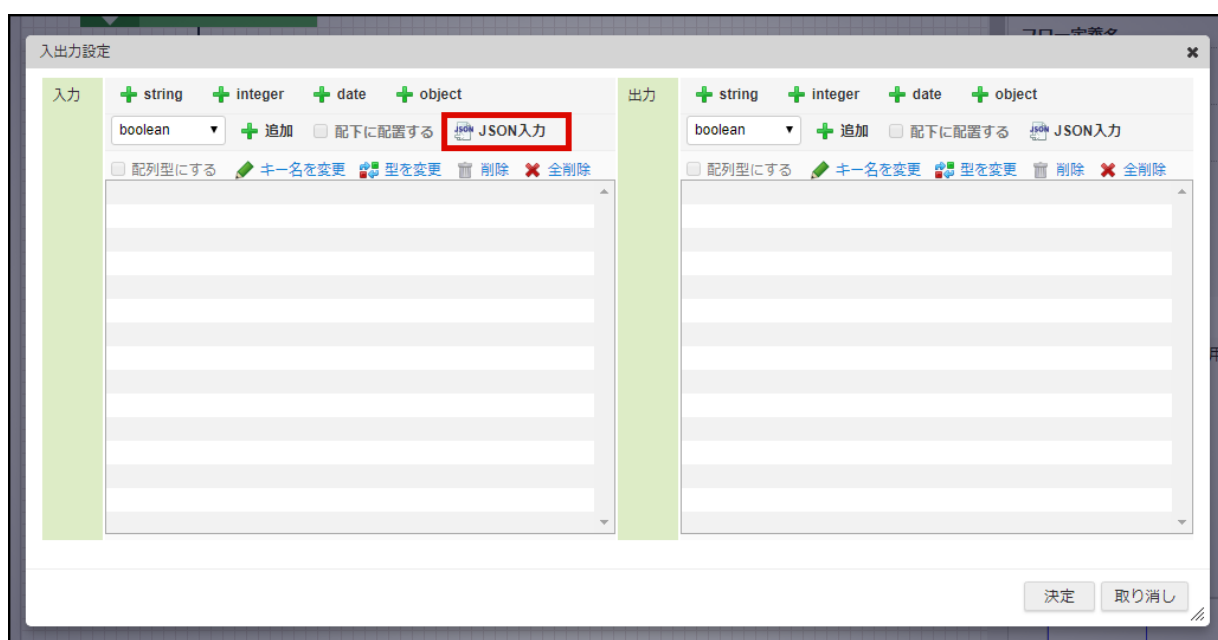
「サイトマップ」→「LogicDesigner」→「フロー定義一覧」から「ロジックフロー定義一覧」を表示し、「新規作成」をクリックしてロジックフローの作成を開始します。



図：完成イメージ（ロジックフロー）

入出力設定を行う

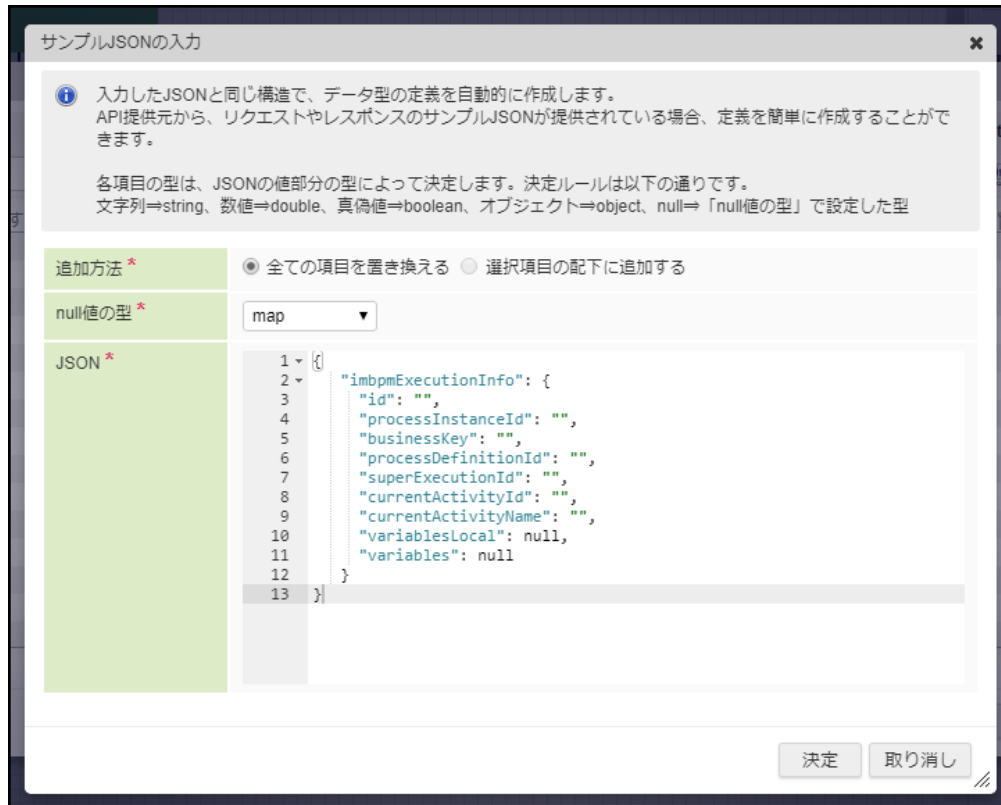
1. 「入出力設定」をクリックします。
2. 「入力」ペインにて、「JSON入力」をクリックします。



図：「入出力設定」 - 「JSON入力」

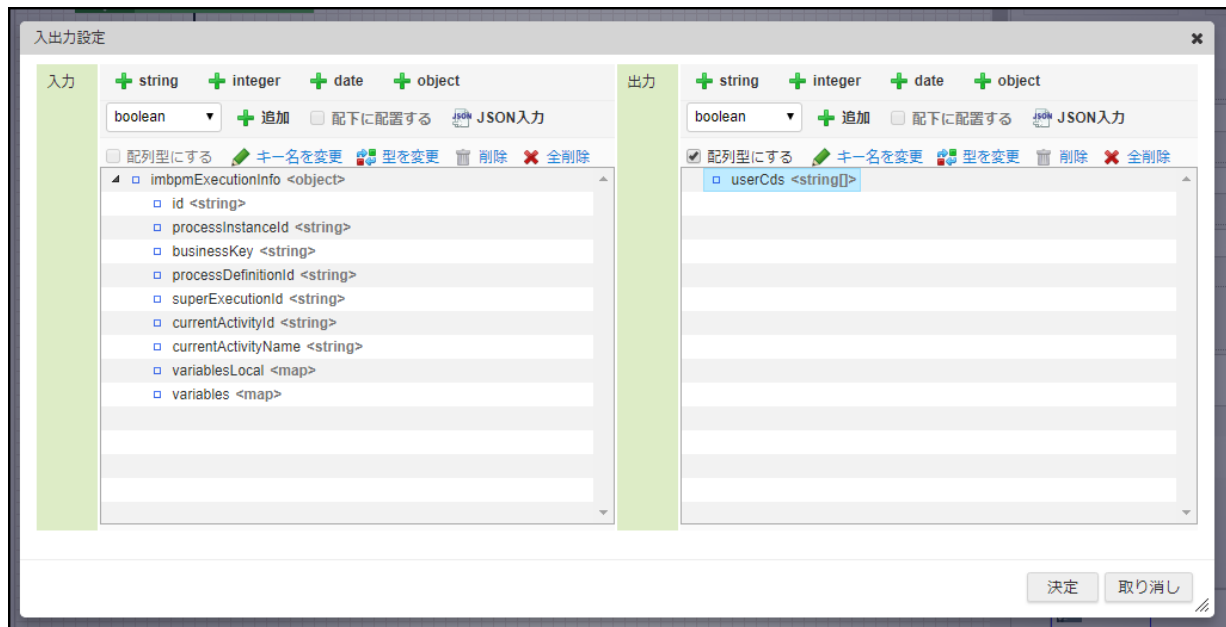
3. 「null値の型」にて `map` を選択します。
4. 「JSON」の内容を、以下の文字列で置き換えます。

```
{
  "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。

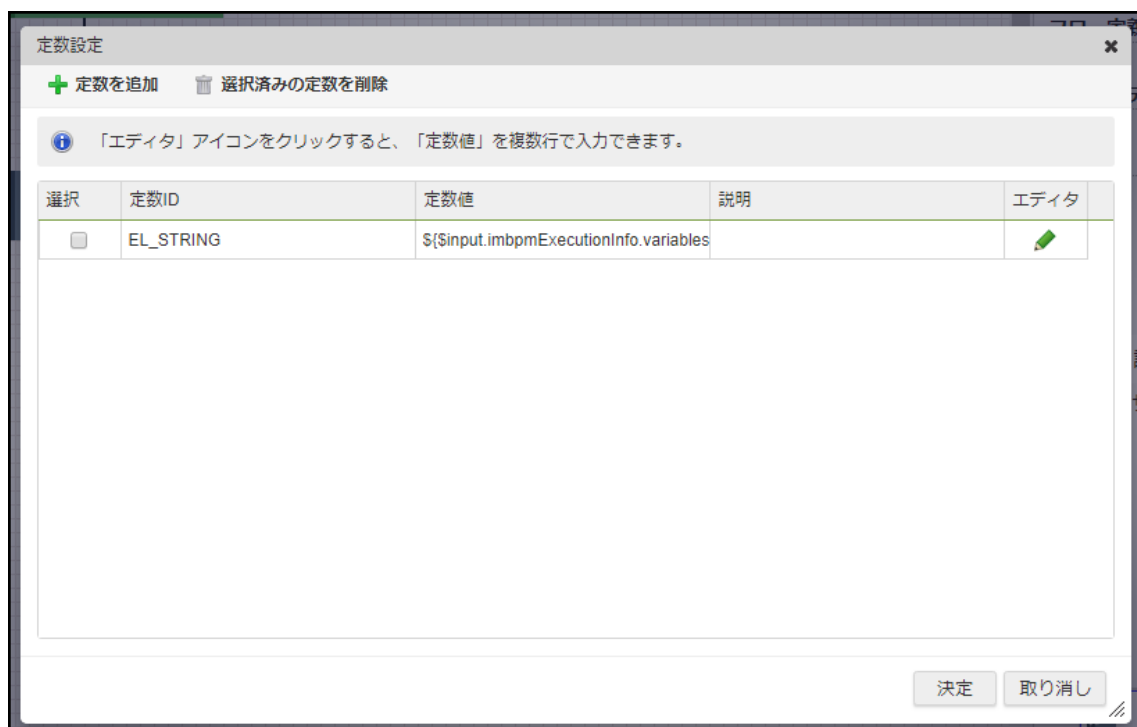
また、処理対象者プラグインの入出力設定については、「IM-Workflow 管理者操作ガイド」-「ロジックフローの入出力設定」を参照してください。

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）

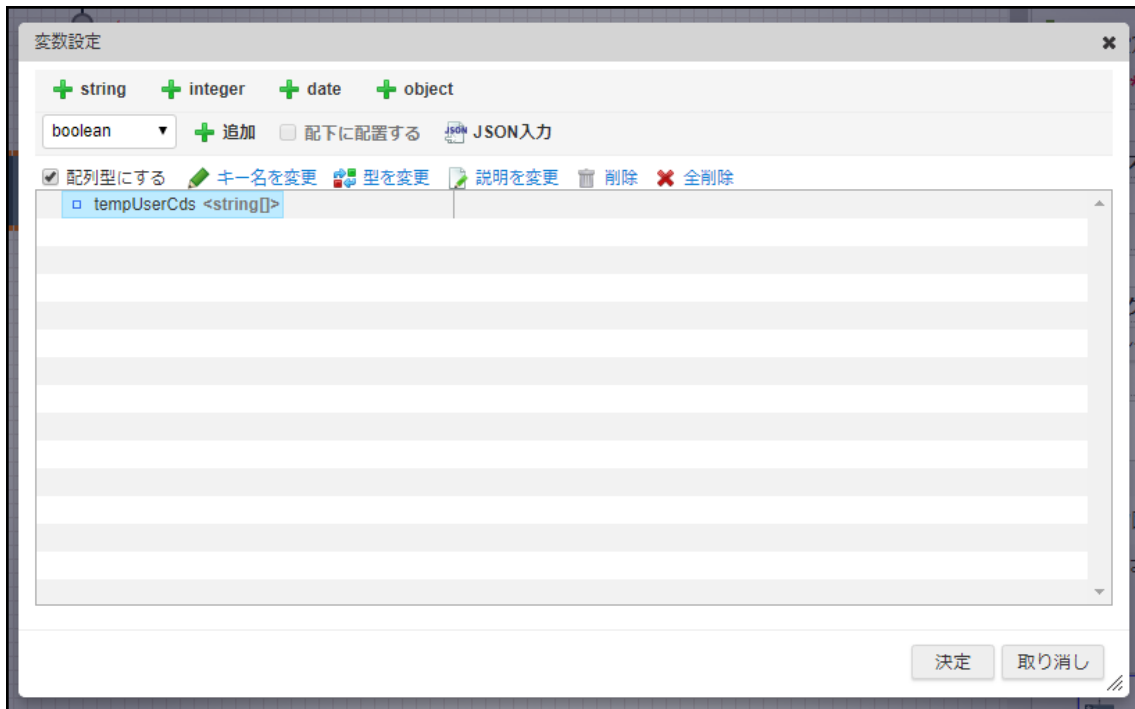
5. 「決定」をクリックします。

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）

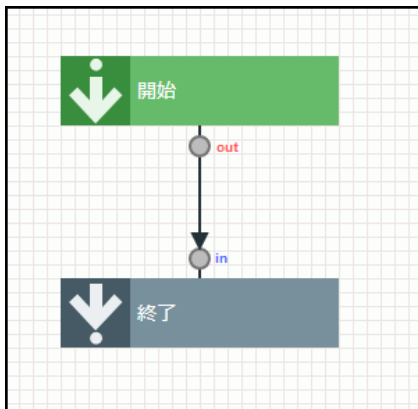
5. 「決定」をクリックします。

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。

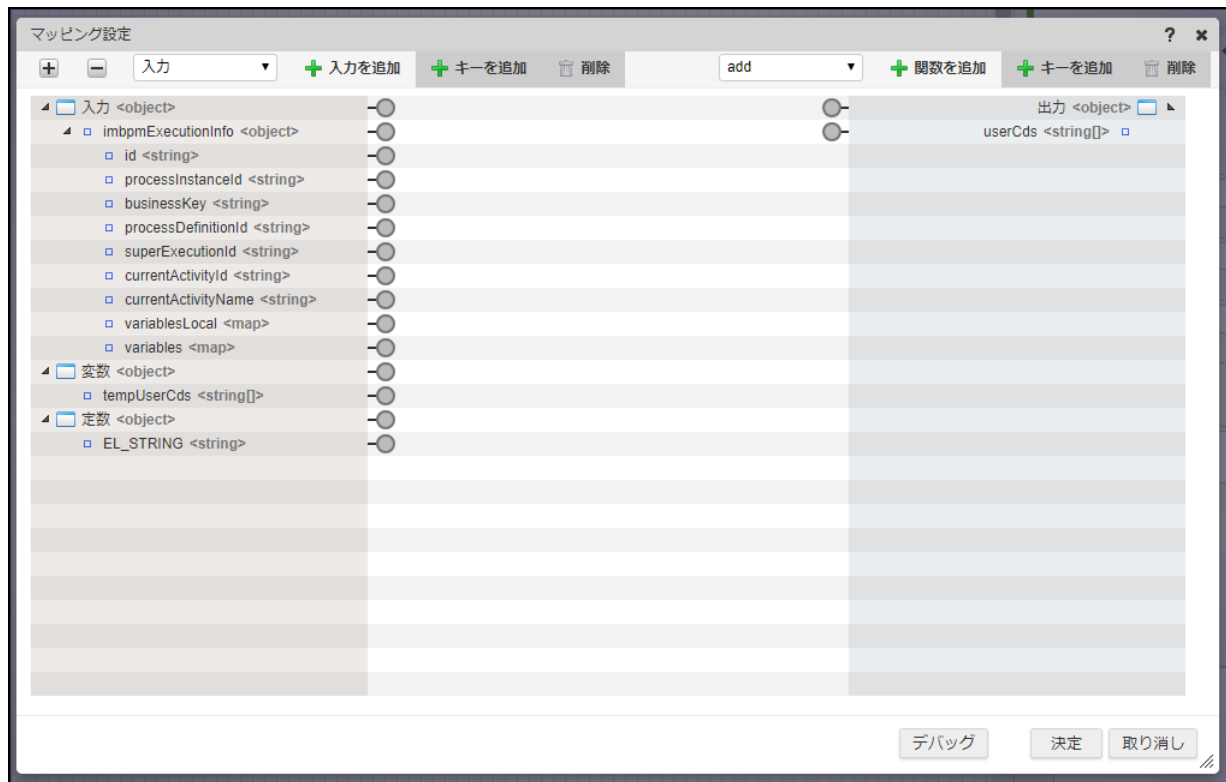
エレメントのマッピングを行う

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。



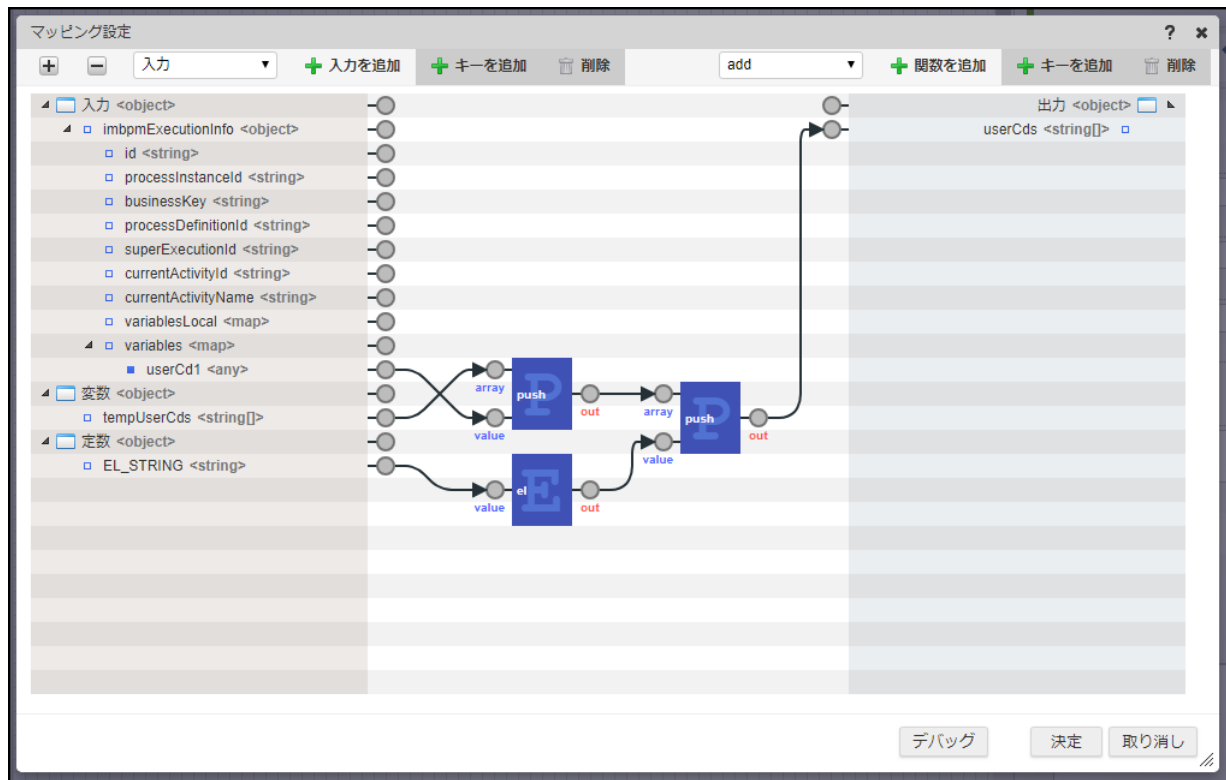
図：エレメントの接続

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。

i 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。

i 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。

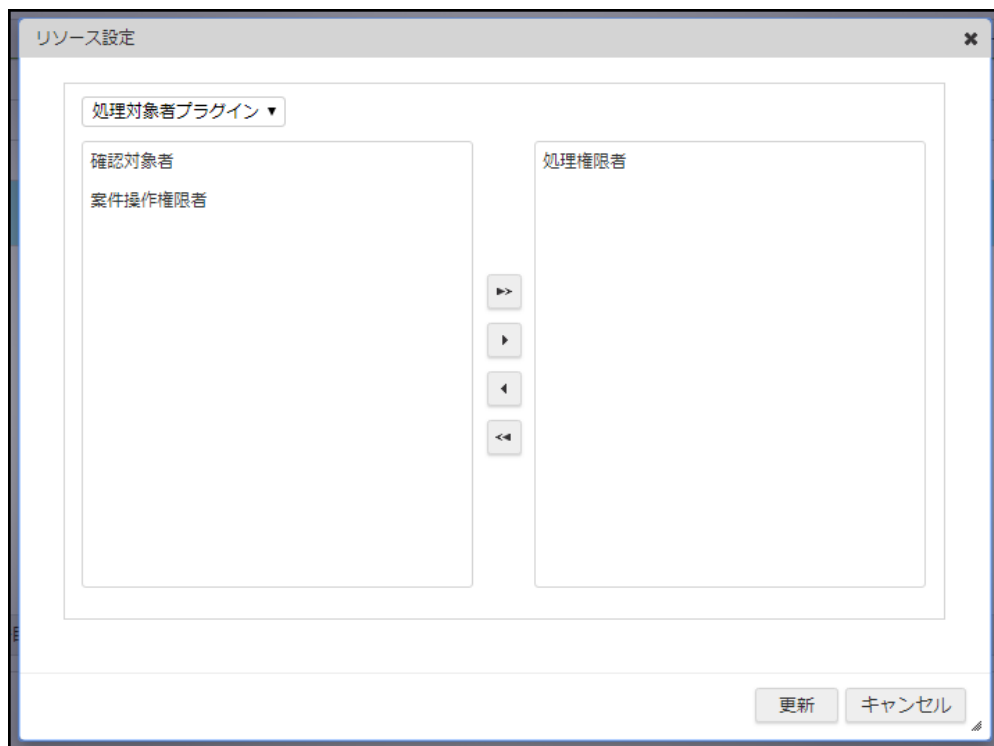
ロジックフローを管理する

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。

検索条件					
リソース設定	リソース	ロジックフロー - フロー定義ID	ロジックフロー - フロー定義名	ロジックフロー - フローカテゴリ	プレビュー
		sample-accounts	List of Accounts	Sample	
		sample-backup-authz-data	Backup authz data files	Sample	
		sample-im-topics-to-log	Read intra-mart atom feed	Sample	
		workflow_target_plugin	【チュートリアル】BPM処理対象者プラグイン	Sample	

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、「」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。

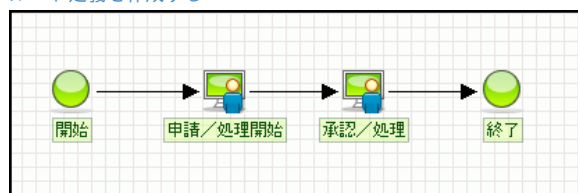
検索条件					
リソース設定	リソース	ロジックフロー - フロー定義ID	ロジックフロー - フロー定義名	ロジックフロー - フローカテゴリ	プレビュー
		sample-accounts	List of Accounts	Sample	
		sample-backup-authz-data	Backup authz data files	Sample	
		sample-im-topics-to-log	Read intra-mart atom feed	Sample	
	処理権限者	✓ workflow_target_plugin	【チュートリアル】BPM処理対象者プラグイン	Sample	

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。

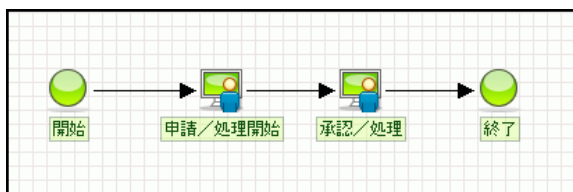
4. 「ルート名」に表示されているロケール分すべてに、【チュートリアル】承認ノード処理対象者指定ルートを入力します。

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。

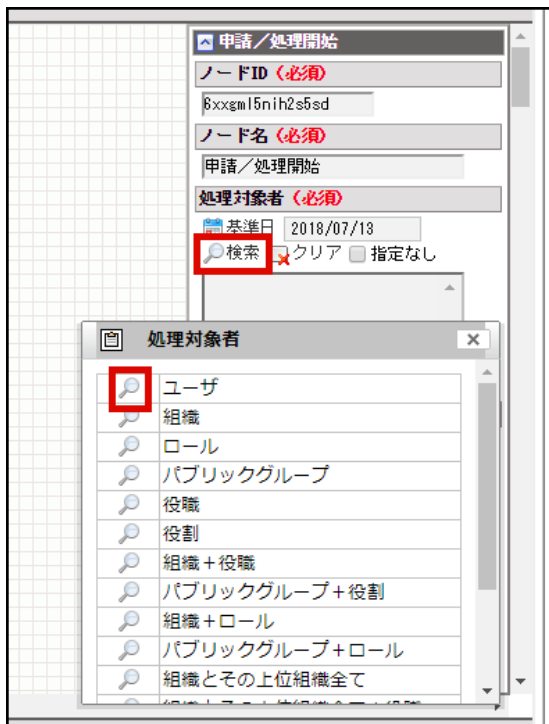
図：「ルート定義 - バージョン - 新規作成」 - 「基本情報」

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。



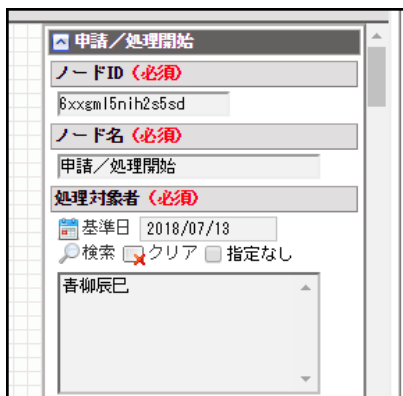
図：「ルート定義 - バージョン - 新規作成」 - 「ルート詳細」

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。



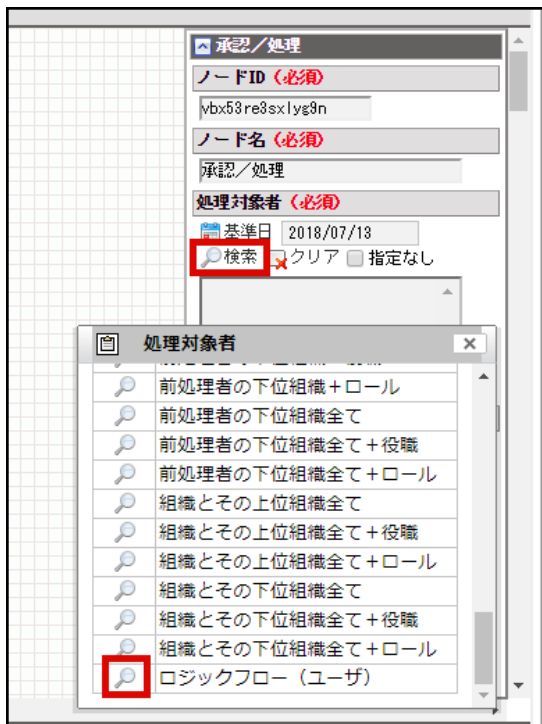
図：「ルート定義 - バージョン - 新規作成」 - 「申請/処理開始」ノード詳細

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。



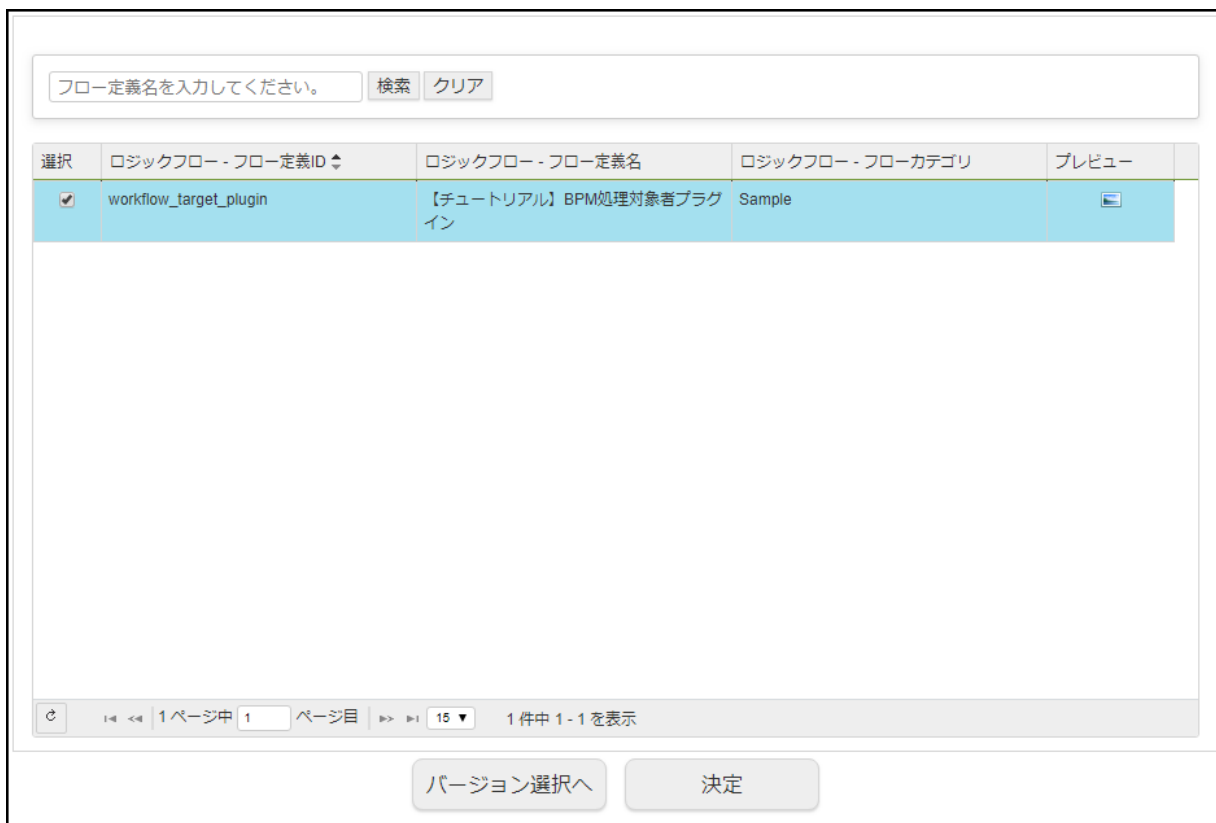
図：「ルート定義 - バージョン - 新規作成」 - 「申請/処理開始」ノード詳細（設定後）

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。

i 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。

基本情報	
フローID (必須)	<code>flow_target_id</code>
フロー名 (必須)	英語 <code>【チュートリアル】承認ノード処理対象者指定フロー</code>
	日本語 <code>【チュートリアル】承認ノード処理対象者指定フロー</code>
	中国語 (中華人民共和国) <code>【チュートリアル】承認ノード処理対象者指定フロー</code>
備考	英語
	日本語
	中国語 (中華人民共和国)

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。

基本情報	
バージョン期間 (必須)	2018/01/01 から まで
バージョン有効/無効 (必須)	☒ 有効 ☐ 無効
備考	英語
	日本語
	中国語 (中華人民共和国)
コンテンツ (必須)	スクリプト開発モデル 検索
ルート (必須)	【チュートリアル】承認ノード処理対象者指定ルート 検索
カレンダー	検索 クリア
機能設定	ファイルの添付 ☒ 有効 ☐ 無効
	一括処理 ☒ 有効 ☐ 無効
	一括確認 ☒ 有効 ☐ 無効
	完了した案件の確認 ☒ 有効 ☐ 無効
	自動処理 ☒ 有効 ☐ 無効
	自動催促 ☒ 有効 ☐ 無効
	対象者を展開する日 ☒ 申請/開始基準日 ☐ システム日

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

- 作成したプロセスを実行環境にデプロイします。
- 「青柳辰巳」でログインします。
- 「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。
- 「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。
- 「個人タスク」に移動した「ユーザ指定」タスクを処理します。

個人タスク										
検索条件										
	処理	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当を外す
☒		【チュートリアル】処理対	tutorial1		ユーザ指定	50	2018/07/13 17:14:3			

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。

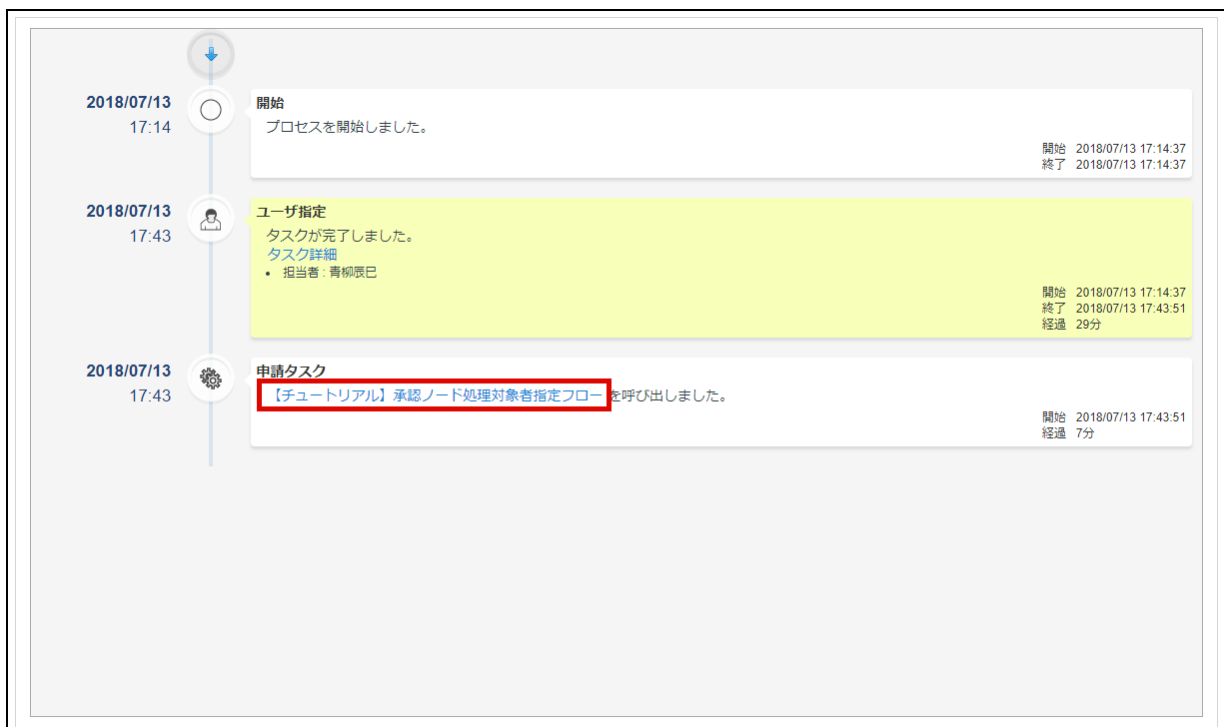
ワークフローの処理対象者を確認する

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「」アイコンをクリックします。

履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	開始日時	担当日時	期限日時	終了日時
	【チュートリアル】処理済	tutorial1		ユーザ指定	50	2018/07/13 17:14:3	2018/07/13 17:15:4		2018/07/13 17:43:5

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。

案件番号	0000000001
案件名	承認ノード処理対象者チュートリアル
申請/処理開始者	青柳辰巳

処理日時	ノード名	処理	処理者	代理先	担当組織
2018/07/13 17:43	申請/処理開始	申請/処理開始	青柳辰巳		
▽処理中	承認/処理		【チュートリアル】BPM 処理対象者プラグイン		

1 ページ中 1 ページ目 15

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

ノード名	承認/処理
------	-------

氏名
上田辰男
青柳辰巳

1 ページ中 1 ページ目 15 2件中 1-2 を表示

図：「処理対象者状況確認」

本番環境への適用

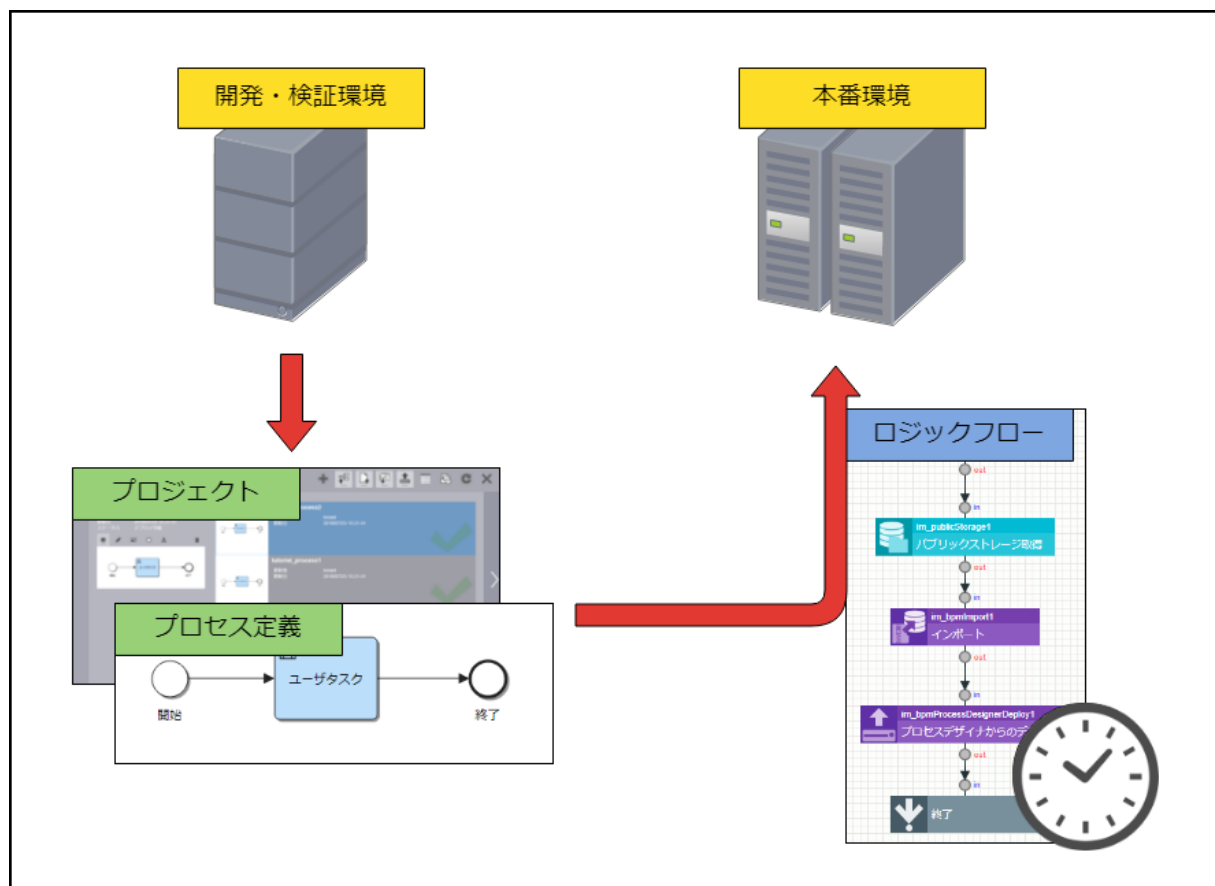
プロセス定義を本番環境へ適用するプロセスの管理方法について説明します。
このチュートリアルは「事前準備」が完了していることを前提としています。

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ー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。
プロセスデザイナー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

ストレージに対する操作、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」

i 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。

i 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。

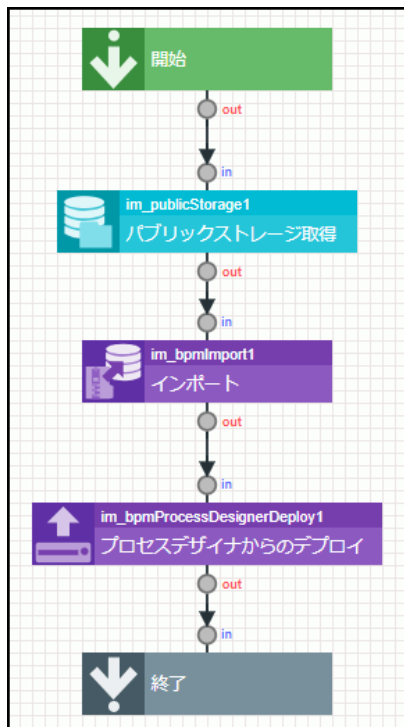
サンプルデータを使用する場合は、[process_auto_deploy.zip](#) を選択します。



図：「ファイル操作」 process_auto_deploy.zip

プロセスデザイナー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

プロセスデザイナーのプロジェクトをプロセスデザイナーへインポートし、プロセスデザイナーから 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>

入出力設定

入力 + string + integer + date + object

boolean + 追加 ☐ 配下に配置する JSON入力

☐ 配列型にする キー名を変更 型を変更 削除 全削除

- 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	,

定数設定

+ 定数を追加 選択済みの定数を削除

「エディタ」アイコンをクリックすると、「定数値」を複数行で入力できます。

選択	定数ID	定数値	説明	エディタ
☐	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。

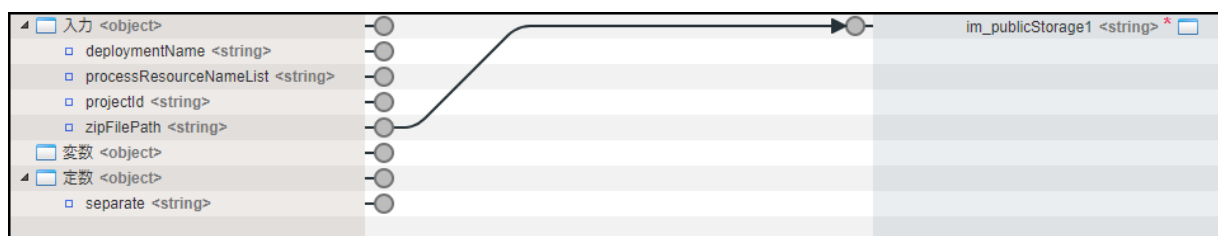


図：「パブリックストレージ取得」

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。

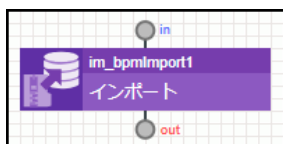
入力（始点）	出力（終点）
入力<object> - zipFilePath<string>	im_publicStorage1<string>



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。

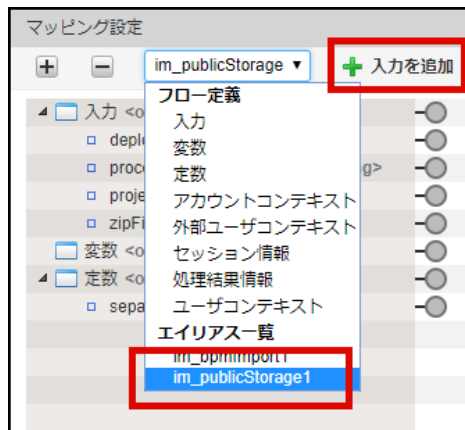
パレット内の「IM-BPM」の一覧から「インポート」を選び、フロー編集画面上に追加します。



図：「インポート」

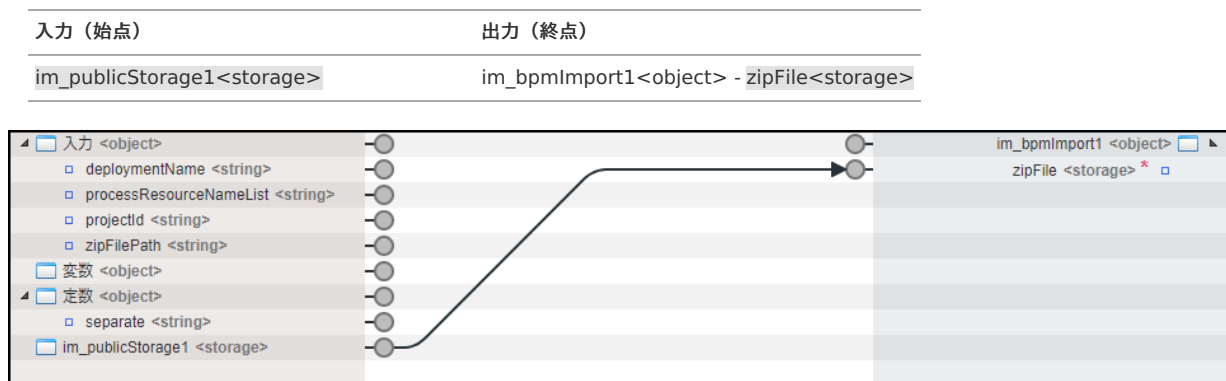
8. 「インポート」のマッピング設定をします。

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。



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. 「プロセスデザイナーからのデプロイ」を配置します。

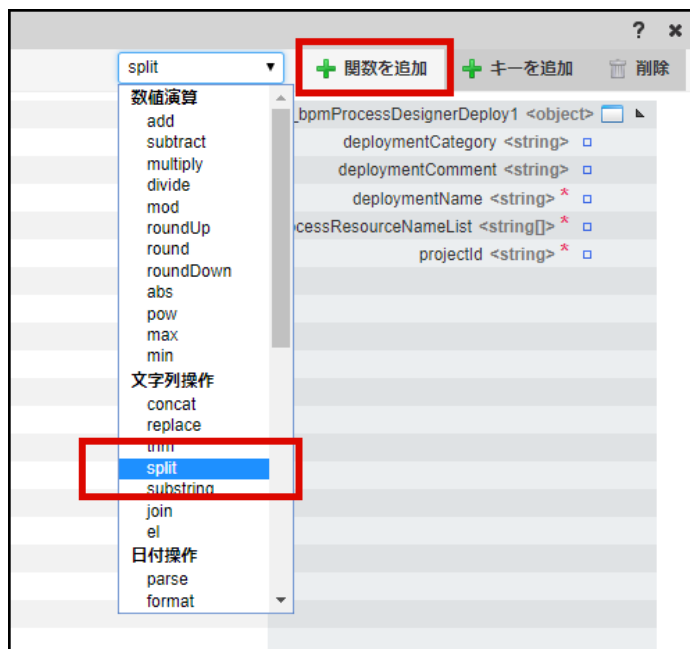
パレット内の「IM-BPM」の一覧から「プロセスデザイナーからのデプロイ」を選び、フロー編集画面上に追加します。



図：「プロセスデザイナーからのデプロイ」

10. 「プロセスデザイナーからのデプロイ」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」 - 「プロセスデザイナーからのデプロイ」 - 「関数を追加」

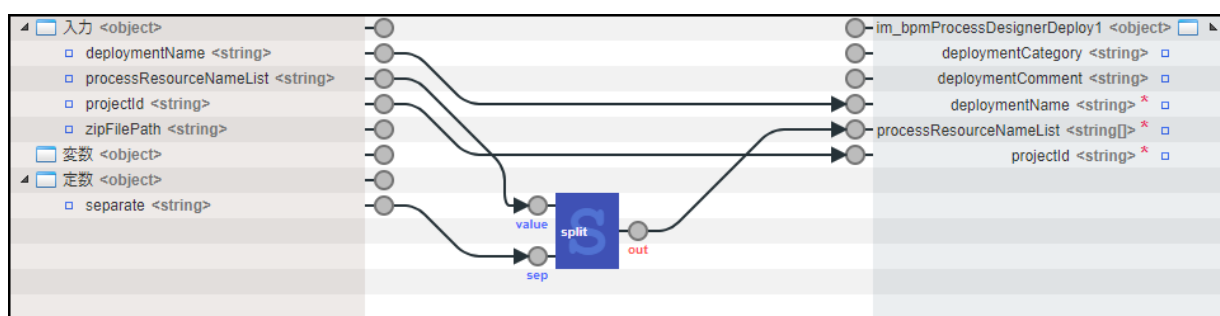


コラム

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」 - 「プロセスデザイナーからのデプロイ」



コラム

「プロセスデザイナーからのデプロイ」についての詳細は、「IM-LogicDesigner仕様書」-「プロセスデザイナーからのデプロイ」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

新規保存

フロー定義ID ^{*} process_auto_deploy

フロー定義名 ^{*}

標準 [*]	process_auto_deploy
日本語	
英語	
中国語 (中華人民共和国)	

フローカテゴリ ^{*}

ID [*]	sample
名称	Sample

備考

決定 取り消し

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。

i 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

ジョブネットID ^{*} process_deploy

ジョブネット名 ^{*}

日本語	プロセス定義をデプロイするジョブ
英語	
中国語 (中華人民共和国)	

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。

実行ジョブ	+ ジョブを追加 - すべて削除		
	ジョブリスト		
	ジョブID	ジョブ名	削除
	imld-flow-executor	フロー実行	✕

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

実行パラメータ	+ パラメータ追加 - すべて削除		
	パラメータリスト (追加後にクリックして入力してください)		
	キー	値	削除
	deploymentName	process_auto_deploy	✕
	flow_id	process_auto_deploy	✕
	processResourceNameList	tutorial_process1,tutorial_process2,tutor	✕
	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」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。



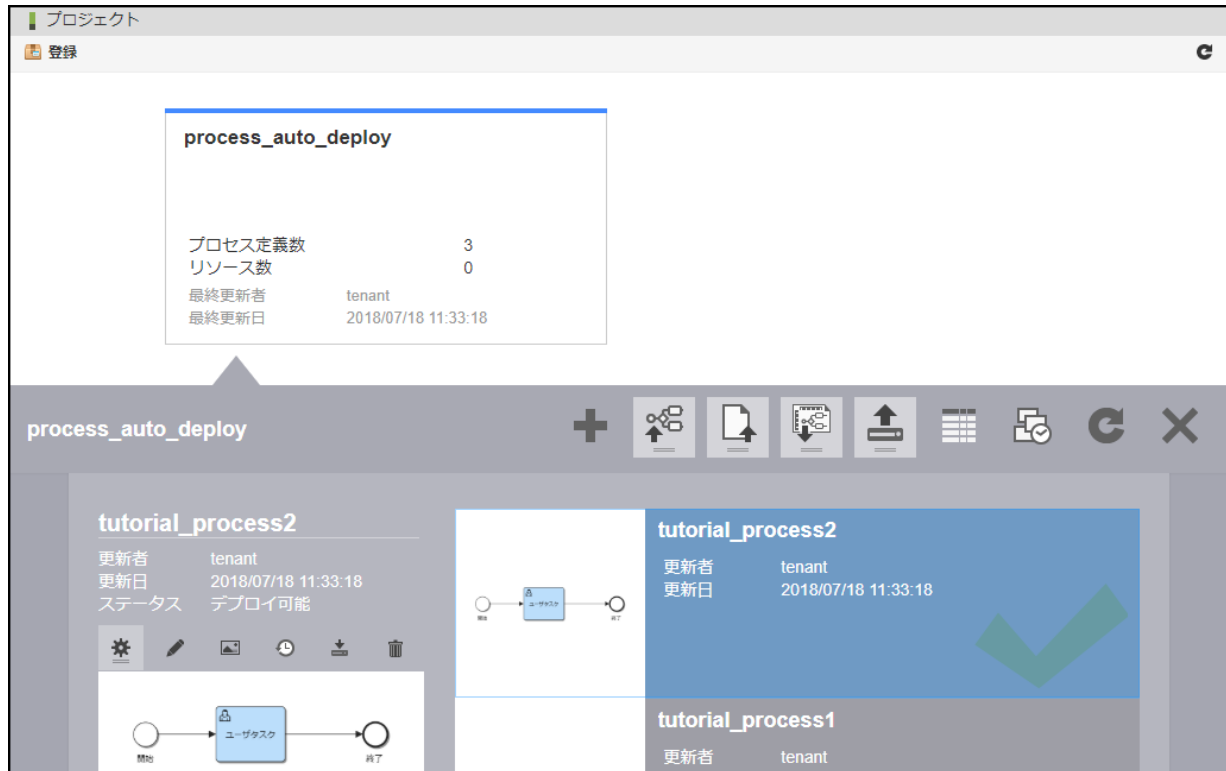
図：「ジョブネットモニター」 プロセス定義をデプロイするジョブ

5. 「サイトマップ」→「BPM」→「プロセスデザイナー」から、「プロセスデザイナー」画面を表示します。

サンプルデータを使用した場合は、プロセスデザイナー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

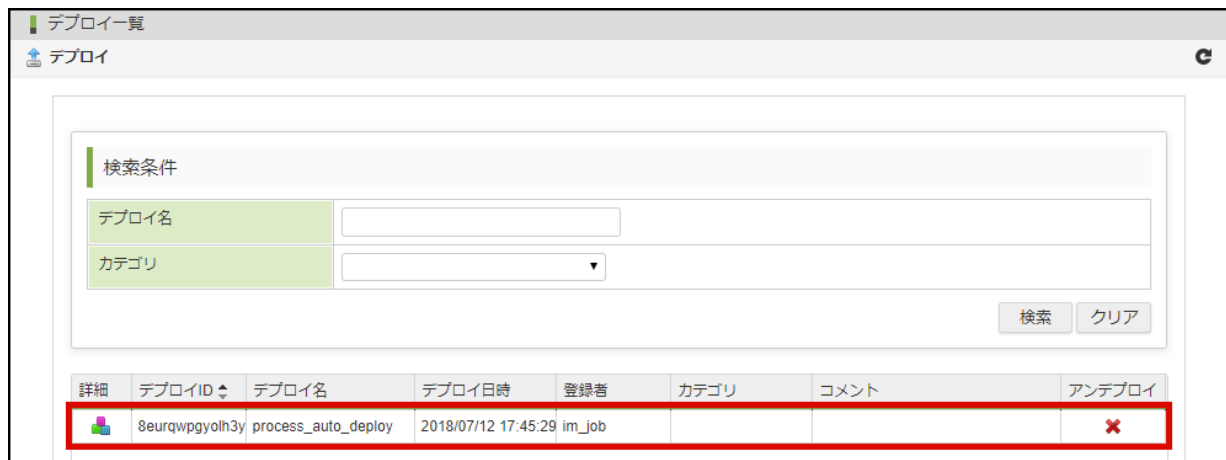
- tutorial_process3



図：「プロセスデザイナー」 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。

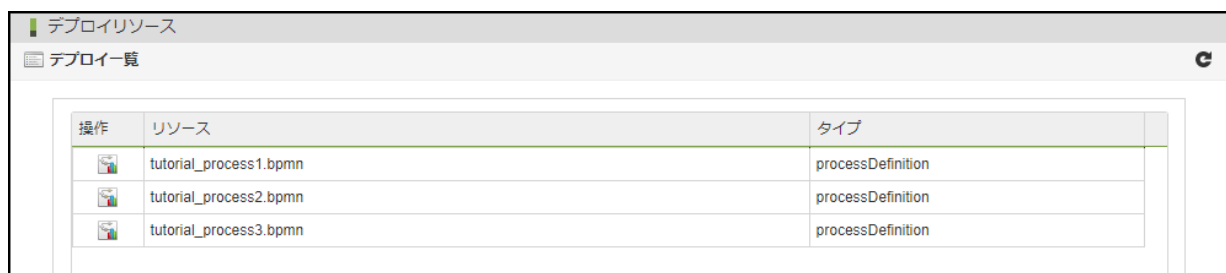


図：「デプロイ一覧」 process_auto_deploy

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

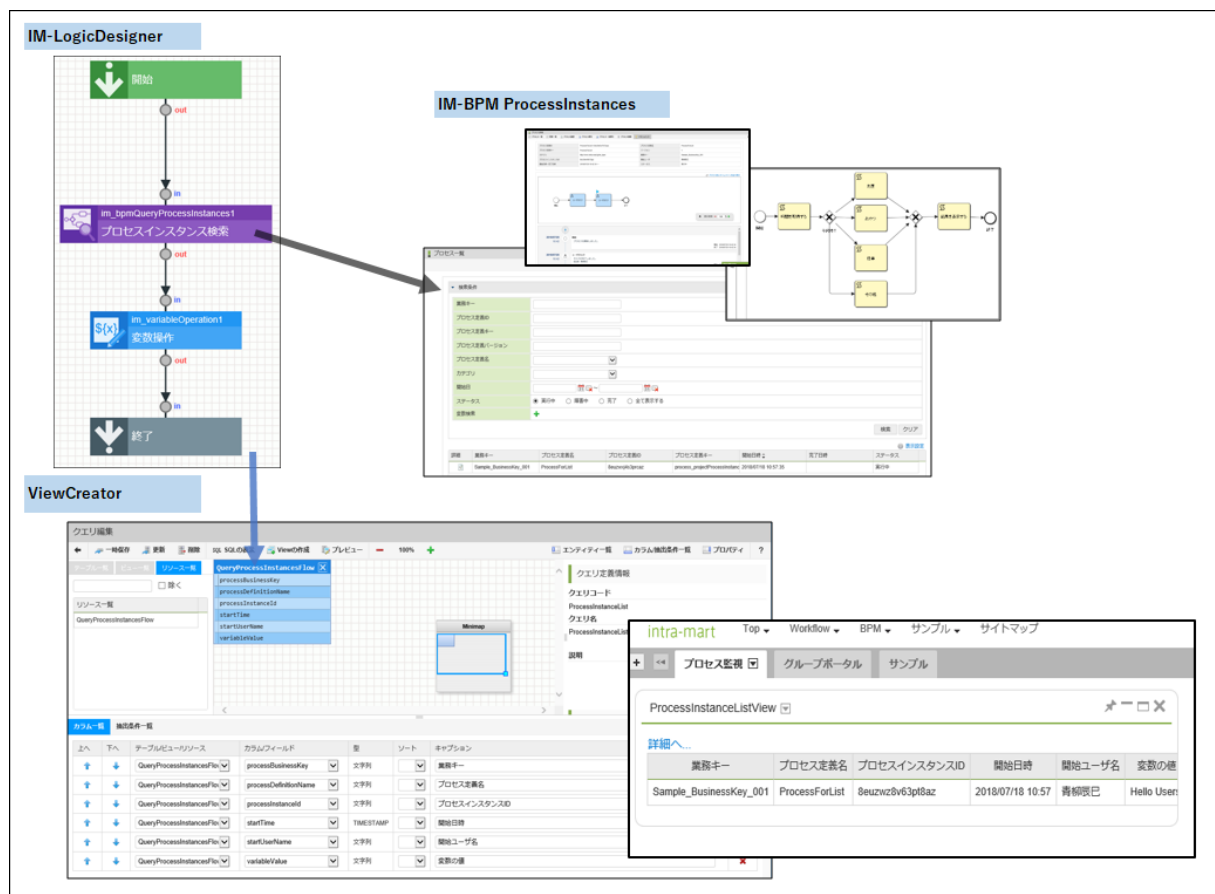


図：「デプロイリソース」

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

i 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。

i 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」

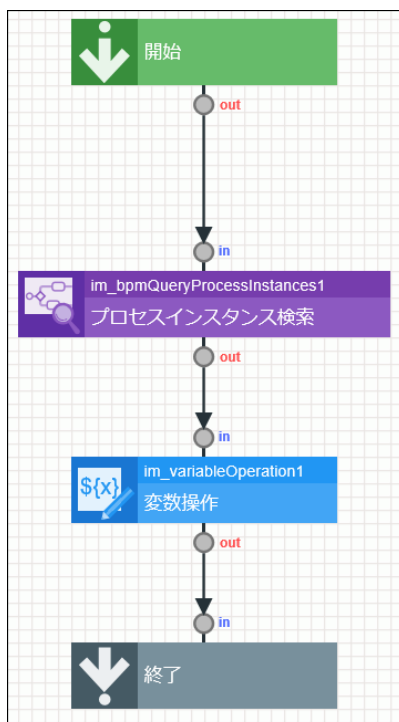
注意

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

- 「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。
- 「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。
- 「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
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。

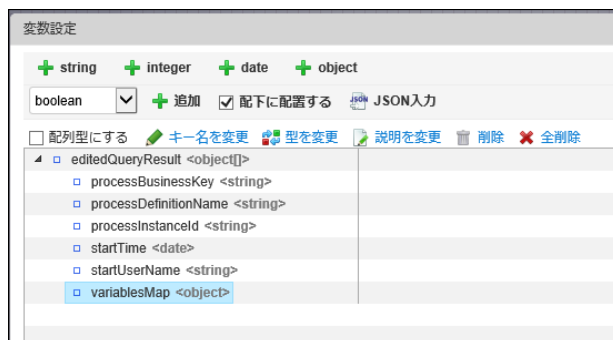
1. 「 object」をクリックし、キー名を `editedQueryResult` とします。
2. 「配列型にする」にチェックを入れます。

キー名	型
<code>editedQueryResult</code>	<code>object[]</code>

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
<code>processBusinessKey</code>	<code>string</code>
<code>processDefinitionName</code>	<code>string</code>
<code>processInstancelId</code>	<code>string</code>
<code>startTime</code>	<code>date</code>
<code>startUserName</code>	<code>string</code>
<code>variablesMap</code>	<code>object</code>

図： `editedQueryResult<object[]>` の配下の項目設定

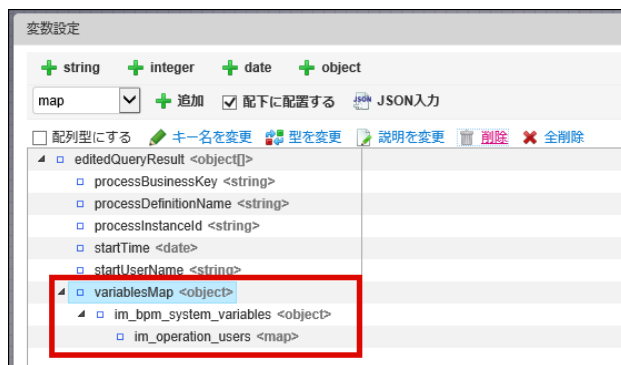
4. `editedQueryResult<object[]>` 配下の `variablesMap<object>` を選択し、「配下に配置する」にチェックを入れ、以下の項目を設定します。

キー名	型
<code>im_bpm_system_variables</code>	<code>object</code>

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。

- 「+ object」をクリックし、キー名をresultとします。
- 「配列型にする」にチェックを入れます。

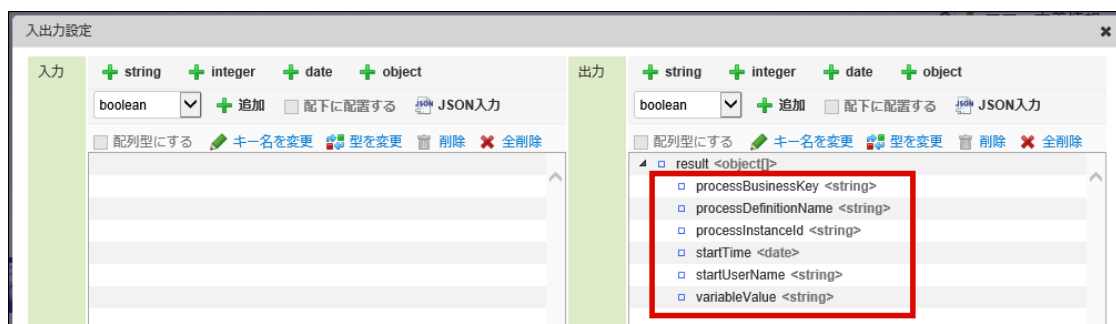
キー名	型
result	object[]



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
processInstancelId	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。

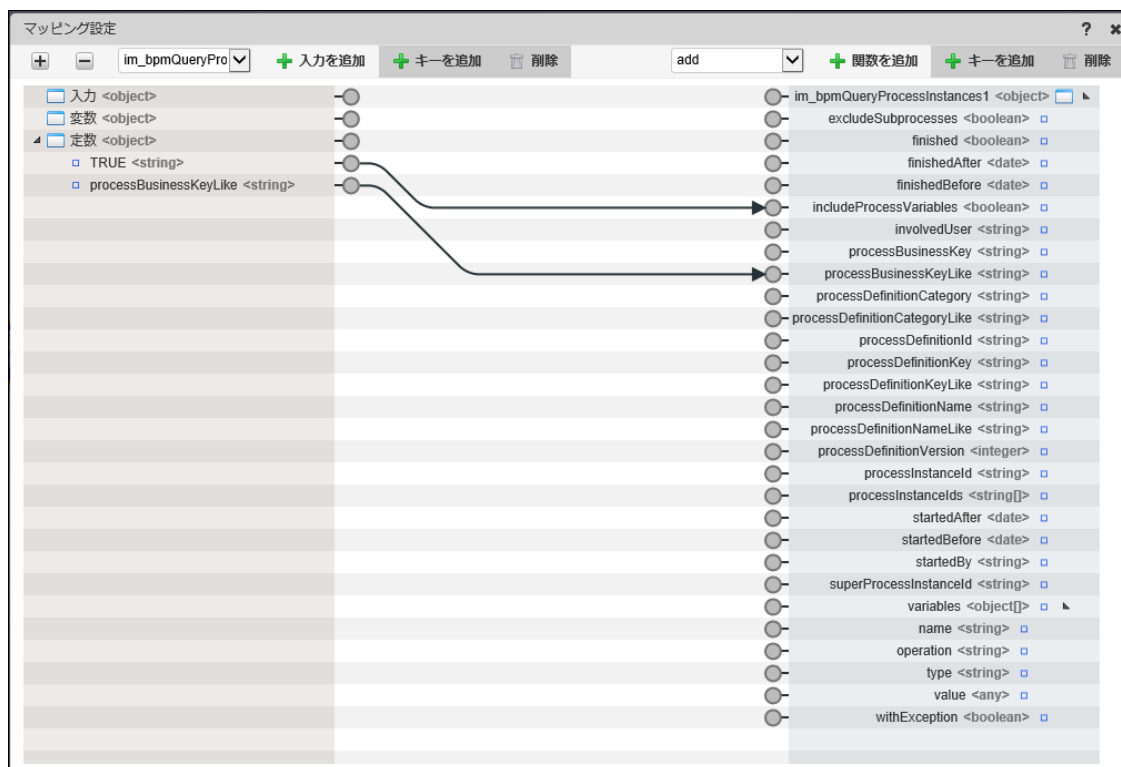
6. パレット内の「IM-BPM」の一覧から「プロセスインスタンス検索」を選び、フロー編集画面上に追加します。



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。

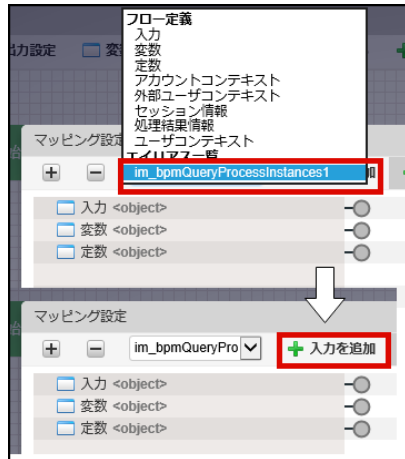
8. パレット内の「基本」の一覧から「変数操作」を選び、フロー編集画面上に追加します。



図：「変数操作」タスク

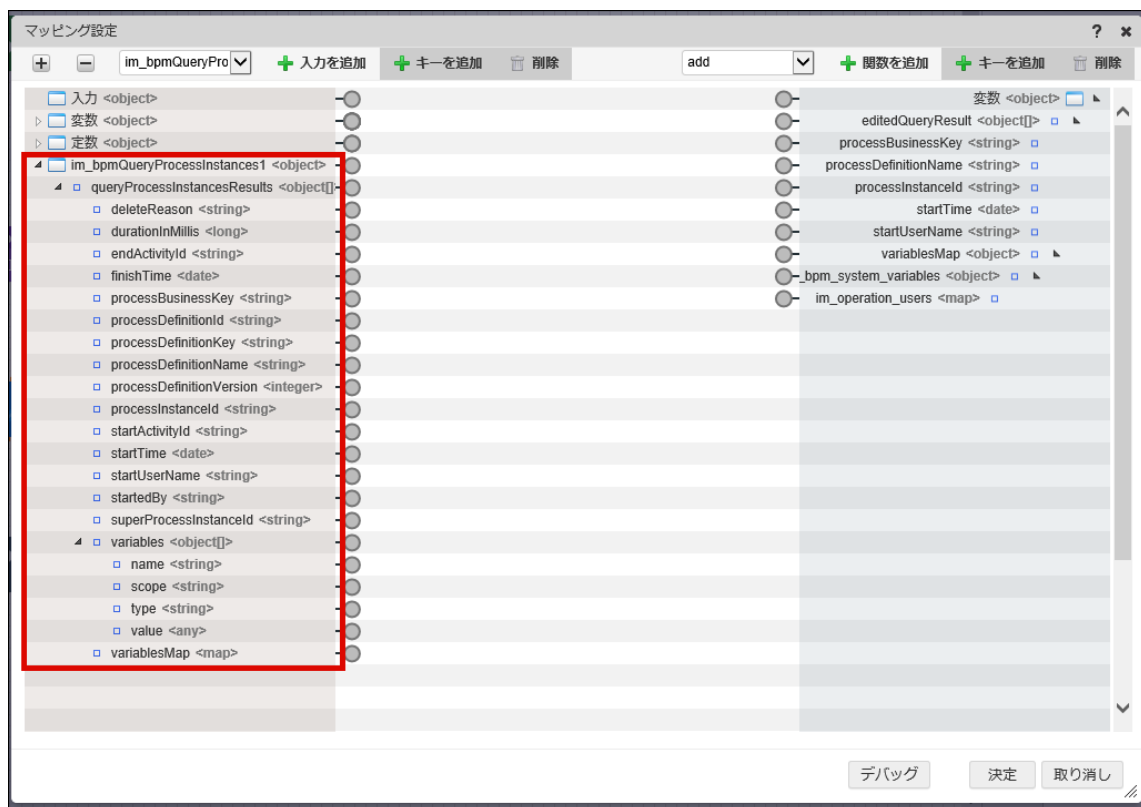
9. 「変数操作」のプロパティ項目「マッピング設定」を開き、以下のようにマッピングを行います。

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

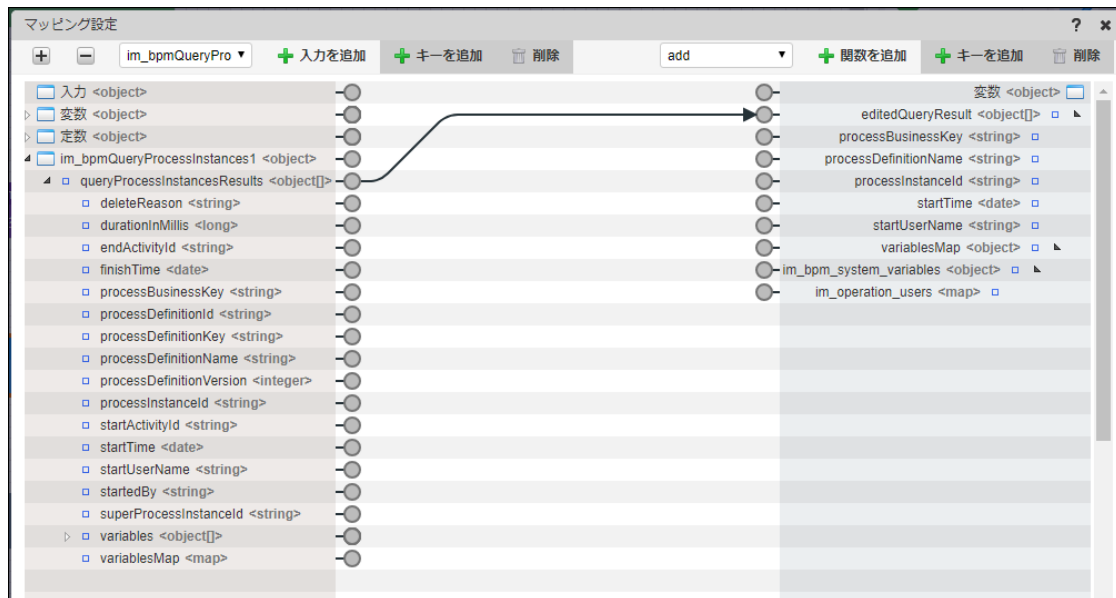
以下のように、「入力ペイン」項目に`im_bpmQueryProcessInstances1<object>`が追加されます。



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）
<code>im_bpmQueryProcessInstances1<object></code> - <code>queryProcessInstancesResults<object[]></code>	変数<object> - <code>editedQueryResult<object[]></code>



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。

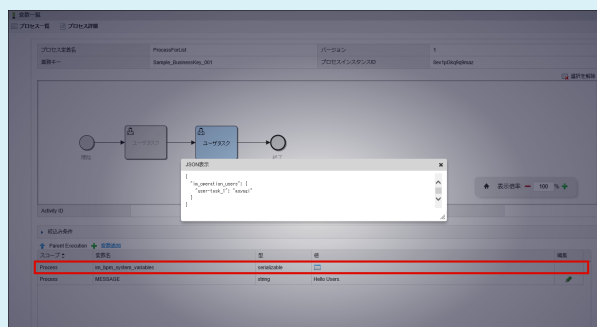


図：im_operation_users<map>へキーuser-task_1を追加

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。

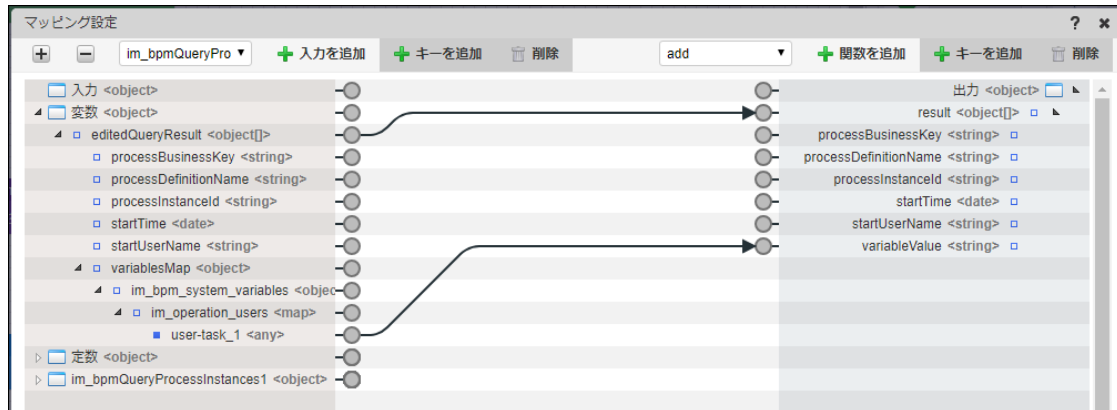


図：「変数一覧」画面

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

i 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

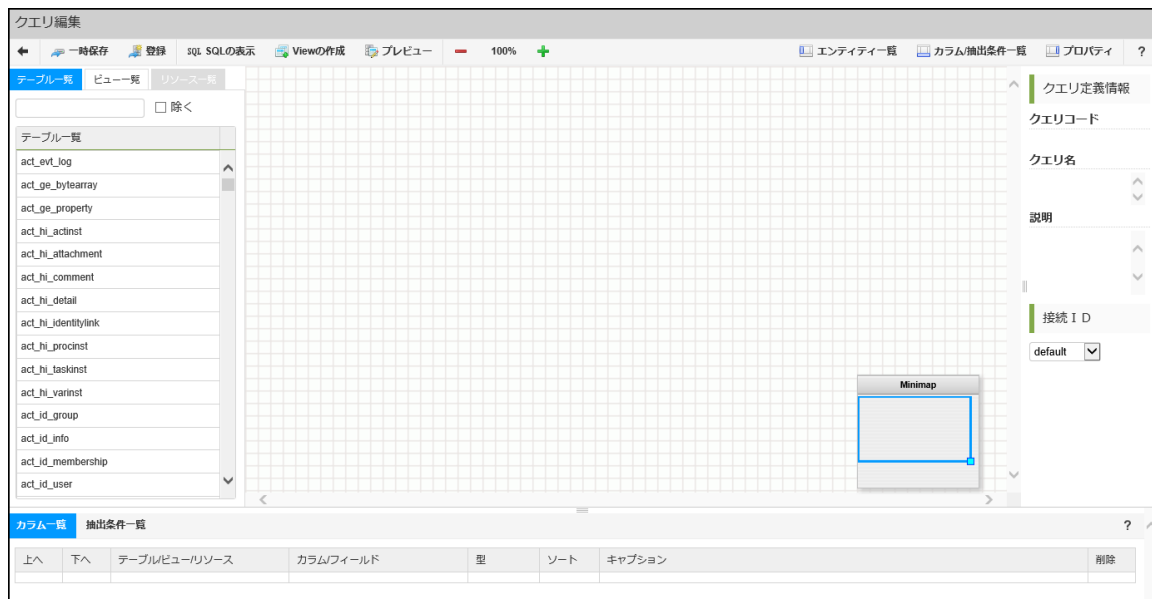
1. 「サイトマップ」→「ViewCreator」→「外部データソース連携」→「ロジックフロー管理」をクリックし「ロジックフロー管理」画面を表示します。
2. 「ロジックフロー管理」画面のフロー定義ID一覧に表示されているフロー定義ID `QueryProcessInstancesFlow` の「ViewCreatorで使用する」にチェックを入れます。



図：「ViewCreatorで使用する」にチェック

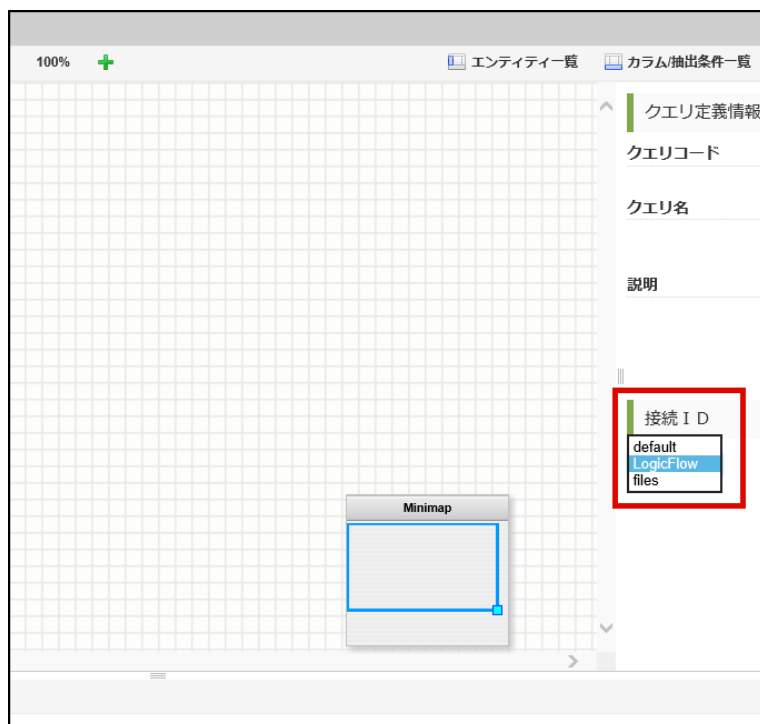
2. クエリの作成

1. 「サイトマップ」→「ViewCreator」→「クエリー一覧」から、ViewCreatorの「クエリー一覧」画面を表示します。
2. クエリー一覧画面の「新規」をクリックして「クエリー編集」画面を表示します。



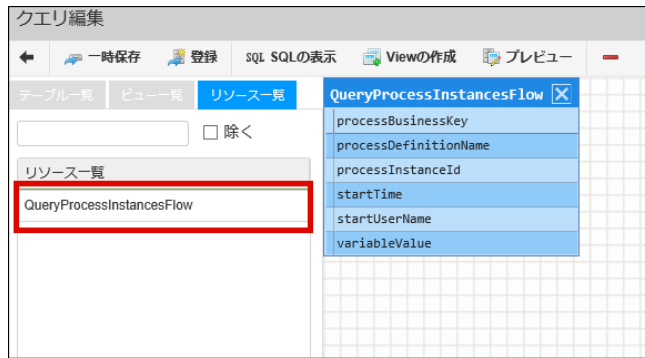
図：クエリの新規作成

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

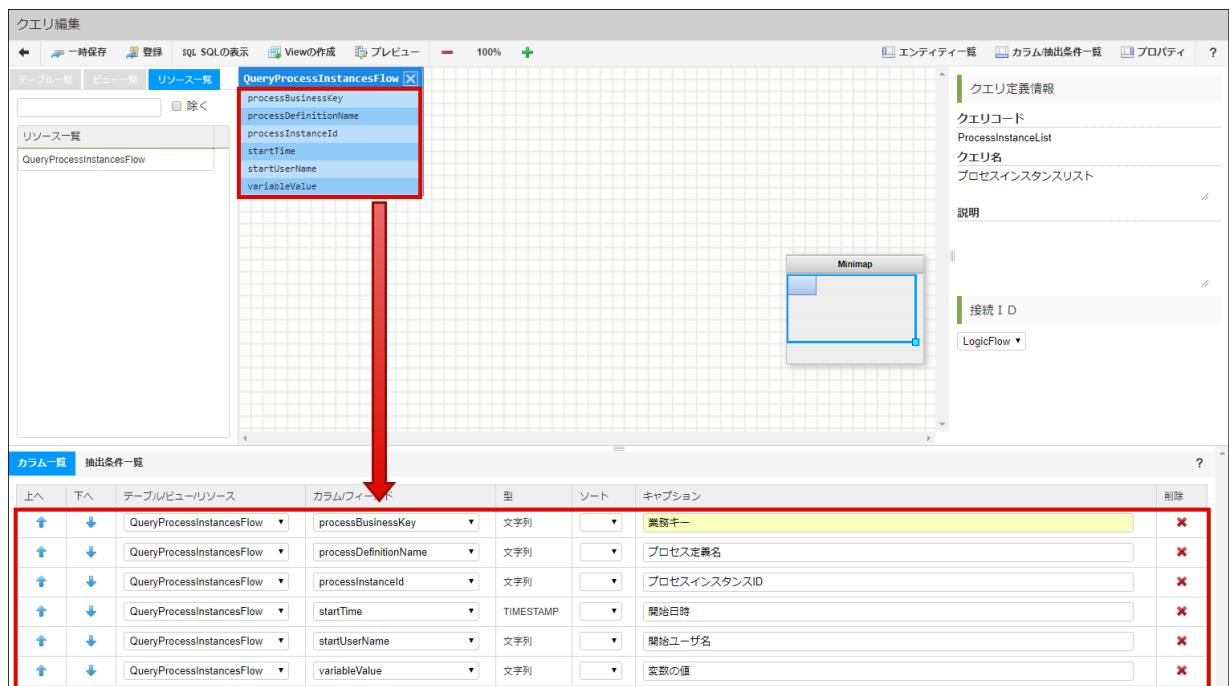
4. 「リソース一覧」よりQueryProcessInstancesFlowを選択します。



図：リソース一覧

5. デザイン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

登録確認ダイアログのスクリーンショット。クエリコードとクエリ名に値が入力されている。クエリコードは「ProcessInstanceList」、クエリ名は「プロセスインスタンスリスト」。

クエリコード *	ProcessInstanceList
クエリ名 *	プロセスインスタンスリスト
説明	

決定 取り消し

図：登録確認ダイアログ

3. データ参照の作成

- 「サイトマップ」→「ViewCreator」→「クエリー一覧」から、ViewCreatorの「クエリー一覧」画面を表示します。
- 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。

☐ データ参照作成	クエリ名	クエリコード	接続ID
☒	プロセスインスタンスリスト	ProcessInstanceList	LogicFlow

図：リスト集計作成

- 「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

データ参照設定画面のスクリーンショット。データ参照コード、データ参照名、クエリ名、集計パターン、説明、エクスポート機能の利用が設定されている。データ参照名は「プロセスインスタンスリスト参照」。

データ参照コード *	8ev7i3xmv6uetaz
データ参照名 *	プロセスインスタンスリスト参照
クエリ名	プロセスインスタンスリスト
集計パターン	リスト
説明	
エクスポート機能の利用	☒ 利用する

図：データ参照名

権限設定画面のスクリーンショット。参照権追加のセクションで、認証済みユーザが選択されている。

参照権追加	☐ ゲストユーザ ☒ 認証済みユーザ
-------	---

登録して一覧へ戻る

図：参照権 - 認証済みユーザ

- 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。

カラム一覧

計算式を追加 カラムの国際化項目の編集

カラム	タイプ	表示	フォーマット	ソート順	パラメータ名	表示設定
▲▼ 業務キー(processBusinessKey)		☒				
▲▼ プロセス定義名(processDefinition)		☒		昇順		
▲▼ プロセスインスタンスID(proc)		☒				
▲▼ 開始日時(startTime)		☒		降順		
▲▼ 開始ユーザ名(startUserName)		☒				
▲▼ 変数の値(variableValue)		☒				

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。

カラム一覧

計算式を追加 カラムの国際化項目の編集

カラム	タイプ	表示	フォーマット	ソート順	パラメータ名	表示設定
▲▼ 業務キー(processBusinessKey)		☒				
▲▼ プロセス定義名(processDefinition)		☒		昇順		
▲▼ プロセスインスタンスID(proc)		☒				
▲▼ 開始日時(startTime)		☒	yyyy/MM/dd HH:n	降順		
▲▼ 開始ユーザ名(startUserName)		☒				
▲▼ 変数の値(variableValue)		☒				

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。

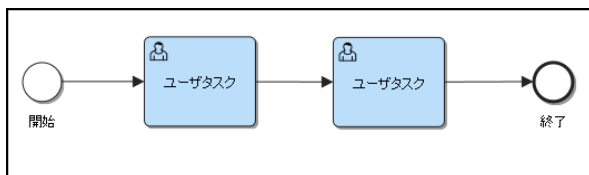
日付フォーマットや、「リスト集計の作成」についてのその他の設定の詳細は「ViewCreator 管理者操作ガイド」-「ViewCreator 管理者操作ガイド - リスト集計の作成」を参照してください。

6. 「登録して一覧へ戻る」をクリックします。

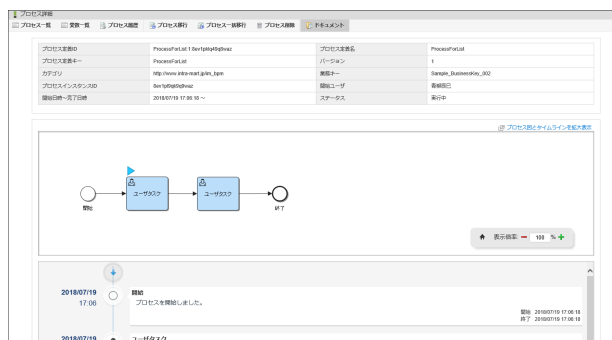
作成したプロセスインスタンス一覧画面を確認する

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。

詳細	デプロイID	デプロイ名	デプロイ日時	登録者	カテゴリ	コメント	アンデプロイ
	8euzvqj43praaz	DeployProcessForList	2018/07/18 10:56:15	青柳 辰巳		ViewCreator連携確認用プロセスのデプロイ	

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ptask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。

	履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント	担当にする
☐		ProcessForList	Sample_ManagementProcessKey_001		ユーザタスク	50	2018/07/20 16:54:21			
☐		ProcessForList	Sample_BusinessKey_002		ユーザタスク	50	2018/07/20 16:54:14			
☐		ProcessForList	Sample_BusinessKey_001		ユーザタスク	50	2018/07/20 16:54:10			

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。

グループタスク									
検索条件									
☐ 履歴		プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント 担当にする
☐		ProcessForList	Sample_ManagementProcessKey_001		ユーザタスク	50	2018/07/20 16:54:21		
☐		ProcessForList	Sample_BusinessKey_002		ユーザタスク	50	2018/07/20 16:54:14		
☐		ProcessForList	Sample_BusinessKey_001		ユーザタスク	50	2018/07/20 16:54:10		

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。

個人タスク									
検索条件									
☐ 処理	履歴	プロセス定義名	業務キー	カテゴリ	タスク名	優先度	作成日時	期限日時	ドキュメント 担当を外す
☐		ProcessForList	Sample_BusinessKey_001		ユーザタスク	50	2018/07/20 16:54:10		

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。

プロセス一覧							
検索条件							
詳細	業務キー	プロセス定義名	プロセス定義ID	プロセス定義キー	開始日時	完了日時	ステータス
	Sample_ManagementProcessKey_001	ProcessForList	ProcessForList.1.8ev34lbpk8c6az	ProcessForList	2018/07/20 16:54:21		実行中
	Sample_BusinessKey_002	ProcessForList	ProcessForList.1.8ev34lbpk8c6az	ProcessForList	2018/07/20 16:54:14		実行中
	Sample_BusinessKey_001	ProcessForList	ProcessForList.1.8ev34lbpk8c6az	ProcessForList	2018/07/20 16:54:10		実行中

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。

☐ 編集	データ参照作成	データ参照名	更新日
☐		プロセスインスタンスリスト参照	2018/0

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。

プロセスインスタンスリスト参照					
CSV出力					
業務キー	プロセス定義名	プロセスインスタンスID	開始日時	開始ユーザ名	実行の経
Sample_BusinessKey_002	ProcessForList	8ev34lmfg8d3az	2018/07/20 16:54:14	青柳 京己	
Sample_BusinessKey_001	ProcessForList	8ev34l8x8cva2	2018/07/20 16:54:10	青柳 京己	aoyagi

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。

詳細	業務キー	プロセス定義名	プロセス定義ID	プロセス定義キー	開始日時	完了日時	ステータス
	Sample_ManagementProcessK	ProcessForList	ProcessForList.1.8ev34lbpk8ci	ProcessForList	2018/07/20 16:54:21		実行中
	Sample_BusinessKey_002	ProcessForList	ProcessForList.1.8ev34lbpk8ci	ProcessForList	2018/07/20 16:54:14		実行中
	Sample_BusinessKey_001	ProcessForList	ProcessForList.1.8ev34lbpk8ci	ProcessForList	2018/07/20 16:54:10		実行中

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。

プロセスの開始

プロセスを開始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。

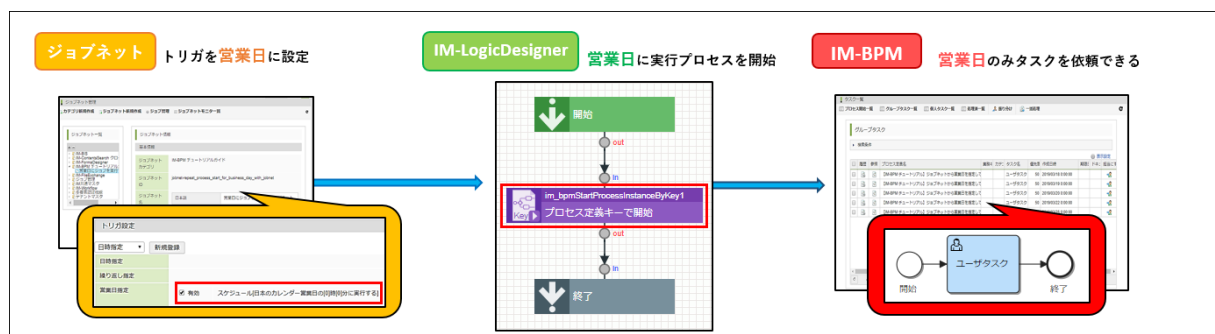
ジョブネットから営業日を指定してプロセスの開始を繰り返す

このチュートリアルでは「ジョブネット」の「トリガ」に営業日を指定して、自動でプロセスの開始を繰り返す方法を解説します。

プロセスを進めていく中で「IM-LogicDesigner」と「ジョブネット」を使用します。

チュートリアルを開始する前に、以下の資料をインポートしてください。

[im_logic_designer-repeat_process_start_for_business_day_with_jobnet.zip](#)
[job-scheduler-repeat_process_start_for_business_day_with_jobnet.xml](#)



図：概要図

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。

IM-LogicDesignerの詳細については、「[IM-LogicDesigner仕様書](#)」を参照してください。

i コラム

「ジョブネット」のインポートは、ジョブネットの「ジョブインポート」から可能です。

「`job-scheduler-repeat_process_start_for_business_day_with_jobnet.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。

インポートする前に、ジョブネットの実行パラメータに以下を追加してください。

- キー : file
- 値 : `job-scheduler-repeat_process_start_for_business_day_with_jobnet.xml`

i 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」

i 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。

[repeat_process_start_for_business_day_with_jobnet.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。

- [プロセス定義を作成する](#)
- [IM-LogicDesignerを確認する](#)
- [ジョブネットの設定を確認する](#)
- [結果を確認する](#)

プロセス定義を作成する

営業日に実行されるプロセスを作成します。

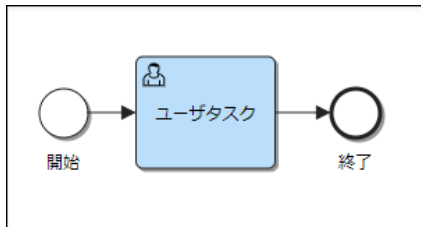
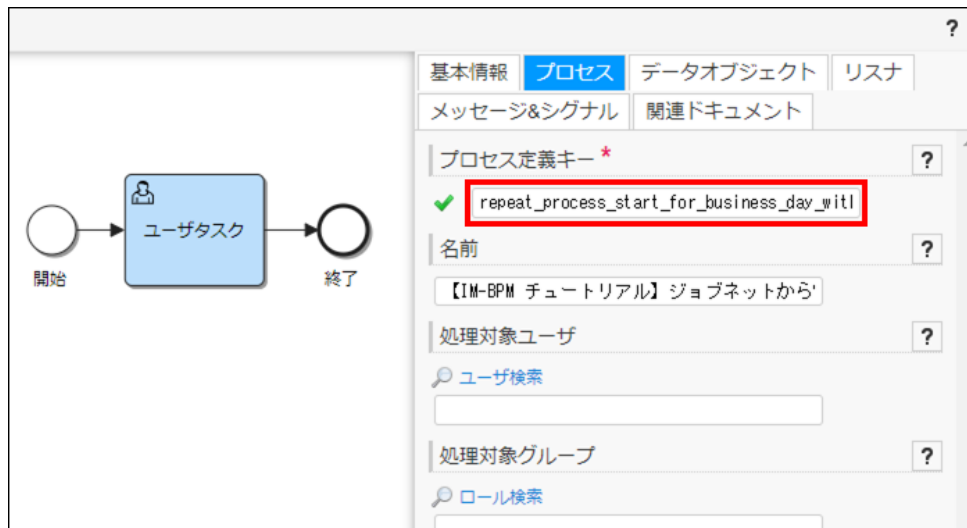


図 : 完成イメージ

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」タブの「プロセス定義キー」に「`repeat_process_start_for_business_day_with_jobnet`」を設定します。



図：プロセス全体 - 「プロパティ」 - 「プロセス」

4. 「ユーザタスク」を設置します。
5. 「ユーザタスク」の処理対象ユーザを設定します。
「ユーザタスク」の「メインコンフィグ」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。

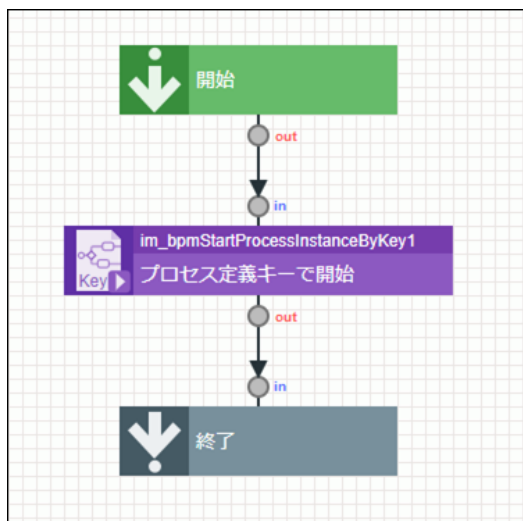


図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

6. 「終了イベント」を設置し、保存します。

IM-LogicDesignerを確認する

チュートリアル開始前にインポートしたIM-LogicDesignerのロジックフローを確認します。
ジョブネットから渡される「実行パラメータ」の値を使用することで、上記で作成したプロセスを実行します。



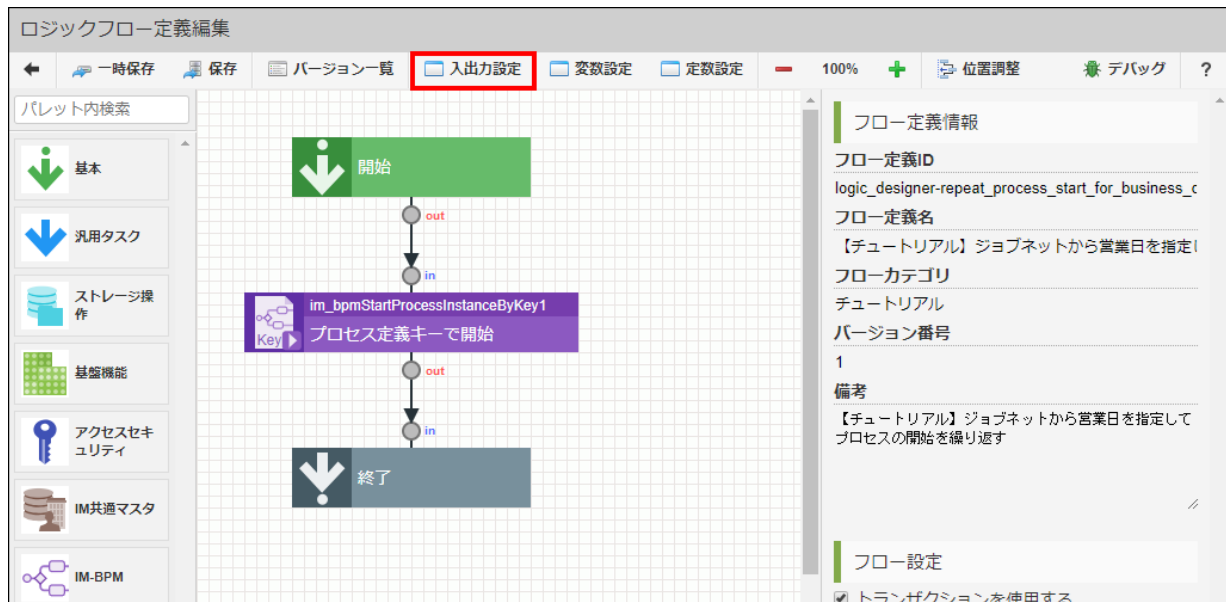
図：ロジックフロー（IM-LogicDesigner）

- 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。
- フロー定義ID「logic_designer-repeat_process_start_for_business_day_with_jobnet」の「アイコンをクリックします。



図：「ロジックフロー定義一覧」

- このロジックフローがプロセスを呼び出す際に使用する入力値を確認します。
メニューバーの「入出力設定」をクリックし、「入出力設定」ダイアログを表示します。



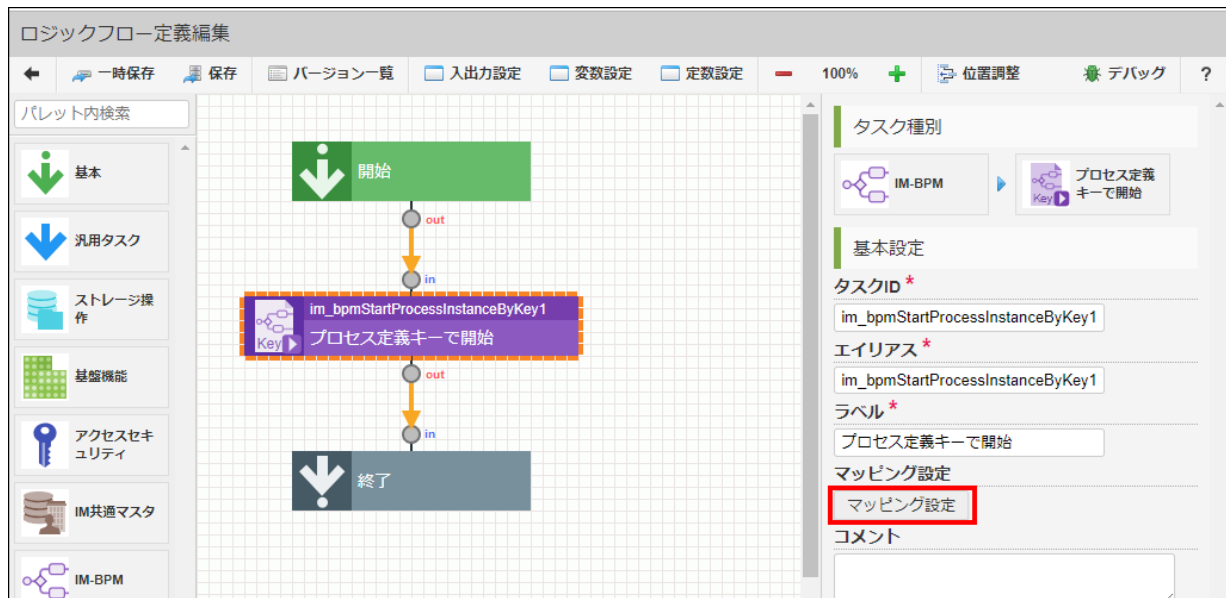
図：「ロジックフロー定義編集」

- 「入出力設定」ダイアログで、入力に「processDefinitionKey<string>」が設定されていることを確認します。
ジョブネットの「実行パラメータ」（後述）に設定してある「processDefinitionKey」をここで紐づけています。



図：「入出力設定」

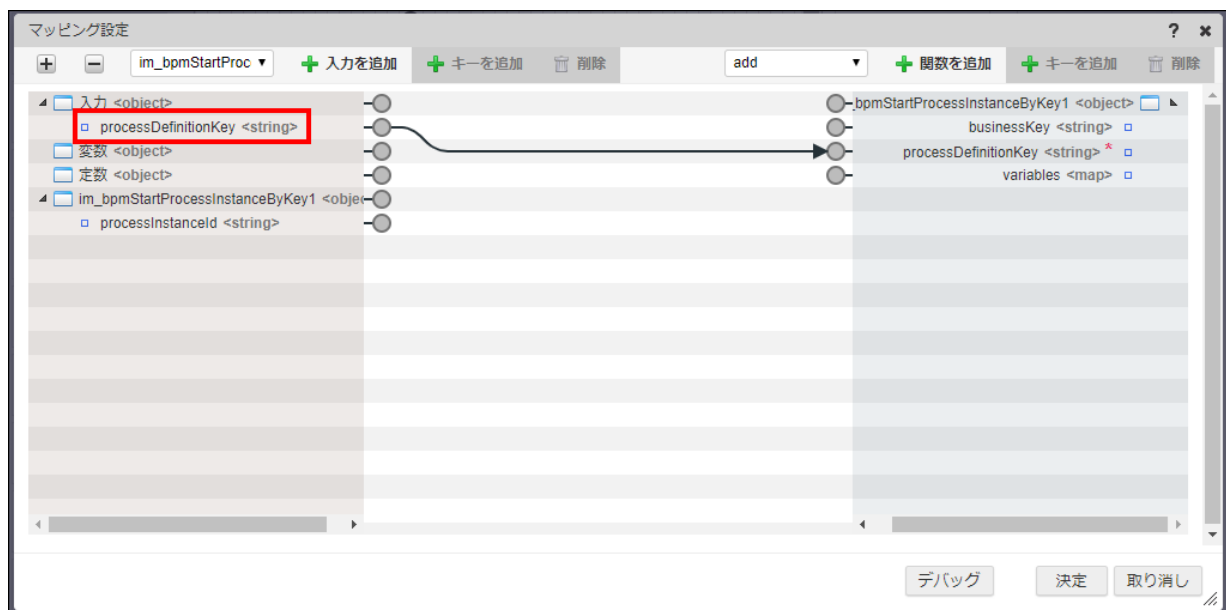
- プロセスを開始するタスクのマッピングを確認します。
タスク「プロセス定義キーで開始」の「マッピング設定」ダイアログを開きます。



図：「ロジックフロー定義編集」

6. 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。

入力（始点）	出力（終点）
入力<object> - processDefinitionKey<string>	processDefinitionKey<string>



図：「マッピング設定」

ジョブネットの設定を確認する

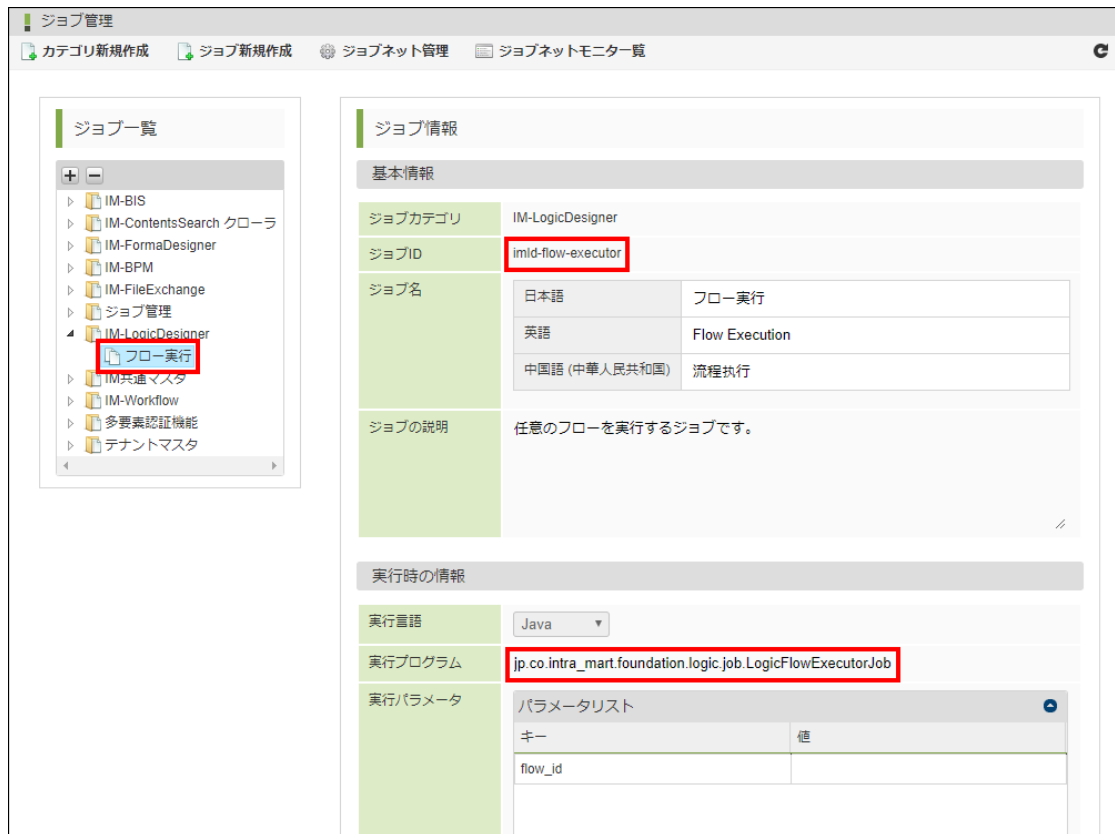
チュートリアル開始前にインポートしたジョブネットと、ジョブを確認します。

ロジックフローを実行するジョブは「IM-LogicDesigner」から提供されているものを利用します。

ジョブネットで「実行パラメータ」と「トリガ」を設定することで、作成したロジックフローを営業日に実行できます。

ジョブ・ジョブネットを実行するには「テナント管理者メニュー」を使用する権限のあるユーザーでログインしてください。

1. ロジックフローを実行するジョブを確認します。
「サイトマップ」→「テナント管理」→「ジョブ管理」→「ジョブ設定」画面を開きます。
2. ジョブ一覧の「IM-LogicDesigner」にある、「フロー実行」を確認します。
このジョブに対して「フローID」を渡すことで、実行するフローを指定できます。



図：「ジョブ管理」

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。

- チュートリアルを開始する前にインポートした「ジョブネット」を確認します。
メニューバーの「ジョブネット管理」をクリックし、「ジョブネット管理」画面を表示します。



図：「ジョブ管理」

- 「ジョブネット一覧」から、「IM-BPMチュートリアルガイド」の「営業日にジョブを実行するジョブネット」をクリックします。
実行したいロジックフローとプロセス定義を「実行パラメータ」に設定し、「トリガ」で営業日に実行されるように指定しています。
このチュートリアルでは、営業日の0時0分に1度起動するようにトリガが設定されています。

- 基本情報
 - ジョブネットカテゴリ：IM-BPM チュートリアルガイド
 - ジョブネットID：jobnet-repeat_process_start_for_business_day_with_jobnet
 - ジョブネット名：営業日にジョブを実行するジョブネット



図：「ジョブネット管理」 - 「基本情報」

- 実行時の情報
 - 実行ジョブ
 - ジョブID : imld-flow-executor
 - ジョブ名 : フロー実行
 - 実行パラメータ
 - フローID
 - キー : flow_id
 - 値 : logic_designer-repeat_process_start_for_business_day_with_jobnet
 - プロセス定義キー
 - キー : processDefinitionKey
 - 値 : repeat_process_start_for_business_day_with_jobnet



図：「ジョブネット管理」 - 「ジョブネット情報」 - 「実行時の情報」

- トリガ設定
 - 営業日指定 : 有効

図：「ジョブネット管理」 - 「ジョブネット情報」 - 「トリガ設定」



コラム

トリガに設定できるスケジュールについては「[ジョブスケジュール仕様書](#)」 - 「[スケジュールの定義方法](#)」を参照してください。

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。

ジョブネットは「トリガ設定」が有効となっているため、インポートした日から実行を待機しています。

今回は、2019年3月18日から3月25日までの8日間のうち営業日にだけ、プロセスを開始した場合を例に説明します。

2019年03月						
日	月	火	水	木	金	土
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

図：2019年3月のカレンダー

実行結果

営業日ではない21日（祝日）、23日（土曜日）、および24日（日曜日）はプロセスが始まらないため、それらの日付ではユーザタスクが作成されていないことを確認します。

タスク一覧

プロセス開始一覧

グループタスク一覧

個人タスク一覧

処理済一覧

振り分け

一括処理

グループタスク

検索条件

表示設定

	履歴	参照	プロセス定義名	業務主	カテゴリ	タスク名	優先度	作成日時	期限	ドキュメント	担当
☐			【IM-BPM チュートリアル】ジョブネットから営業日を指定して			ユーザタスク	50	2019/03/18 0:00:00			
☐			【IM-BPM チュートリアル】ジョブネットから営業日を指定して			ユーザタスク	50	2019/03/19 0:00:00			
☐			【IM-BPM チュートリアル】ジョブネットから営業日を指定して			ユーザタスク	50	2019/03/20 0:00:00			
☐			【IM-BPM チュートリアル】ジョブネットから営業日を指定して			ユーザタスク	50	2019/03/22 0:00:00			
☐			【IM-BPM チュートリアル】ジョブネットから営業日を指定して			ユーザタスク	50	2019/03/25 0:00:00			

1 ページ中

1

ページ目

10

5 件中 1 - 5 を表示

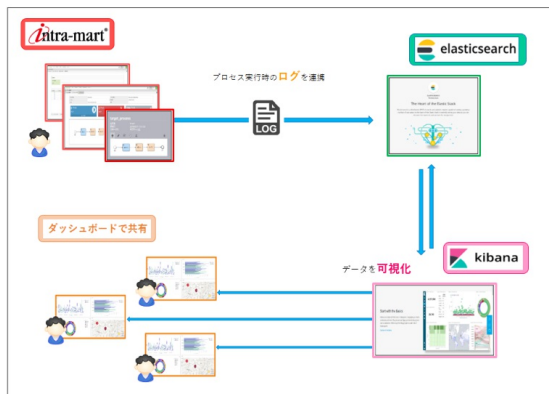
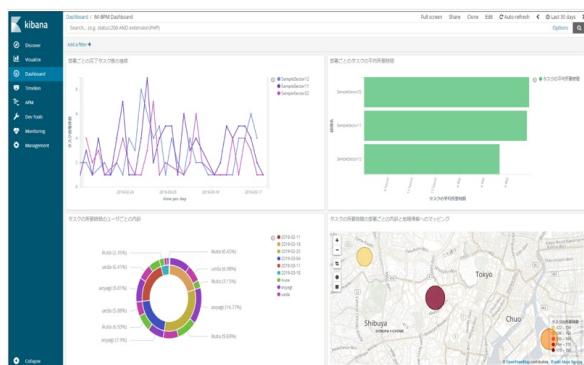
Elasticsearch/ Kibana連携機能

IM-BPM for Accel Platformのプロセスの実行時のログをElasticsearchおよびKibanaへ連携する機能や、Kibana上で、連携されたログを可視化し、ダッシュボードとして共有する方法について説明します。

このチュートリアルはintra-mart Accel Platformに「IM-BPM/Elasticsearch コネクタ」モジュールが組み込まれている環境上で「事前準備」が完了していること、ElasticsearchおよびKibanaの環境構築が完了していることを前提としています。

Elasticsearchの環境構築については、「IM-BPM for Accel Platform セットアップガイド」 - 「Elasticsearch のセットアップ」を参照してください。

また、Kibanaの導入については、「Kibanaの導入」を参照してください。



図：概要図

IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する

このチュートリアルでは、「IM-BPM/Elasticsearch コネクタ」モジュールによってElasticsearchへ登録されたプロセスの実行時のログを、Kibana上で可視化し、可視化した情報を表示するためのダッシュボードを作成する方法を解説します。



注意

本チュートリアルは、intra-mart Accel Platformに「IM-BPM/Elasticsearch コネクタ」モジュールが組み込まれている環境上で行うことを前提としています。

また、各プロダクトのバージョンは下記を想定しています。

- intra-mart Accel Platform : 2019 Spring(Violette)
- IM-BPM/Elasticsearch コネクタ : 8.0.3
- Elasticsearch : 6.4.0
- Kibana : 6.4.0



コラム

Elasticsearch連携機能の詳細については、「IM-BPM 仕様書」 - 「Elasticsearch連携機能」を参照してください。

IM-BPM/Elasticsearch コネクタの設定については、「IM-BPM 設定ファイルリファレンス」 - 「IM-BPM Elasticsearch コネクタ設定」を参照してください。

Elasticsearchの環境構築については、「IM-BPM for Accel Platform セットアップガイド」 - 「Elasticsearch のセットアップ」を参照してください。

- Kibanaの導入
 - Kibanaの取得
 - Kibanaの設定
 - Kibanaの起動
- プロセス実行とElasticsearchおよびKibanaの概要
- 本チュートリアルで使用する時系列データを作成する
 - 統計情報取得の対象となるプロセスのデプロイを行う
 - barファイルのデプロイ
- Elasticsearch に変数用のテンプレートを登録する
- データ作成用プロセスを実行する
- Elasticsearch のインデックスをKibanaから確認する
- Kibana上にインデックスパターンを作成する
- タスクの変数を使用して部署ごとの統計情報のグラフを作成する
 - 部署ごとの完了タスク数の推移
 - 部署ごとのタスクの平均所要時間
- Kibana上にダッシュボードを作成する

Kibanaの導入

Windows環境上へKibanaを導入する際の例を説明します。

Kibanaの取得

Kibanaのホームページより、version:6.4.0のKibanaをダウンロードし、任意の場所へ解凍します。
以降、Kibanaを展開した場所を %KIBANA_HOME% と略します。



コラム

URL(version:6.4.0)

ダウンロードページ : <https://www.elastic.co/downloads/past-releases/kibana-oss-6-4-0> (English)

Kibanaの設定

<%KIBANA_HOME%/config/kibana.yml> ファイルをエディタなどで開き、下記の設定を行います。

```
server.host: Kibanaのホスト名
elasticsearch.url: ElasticsearchのURL
```



注意

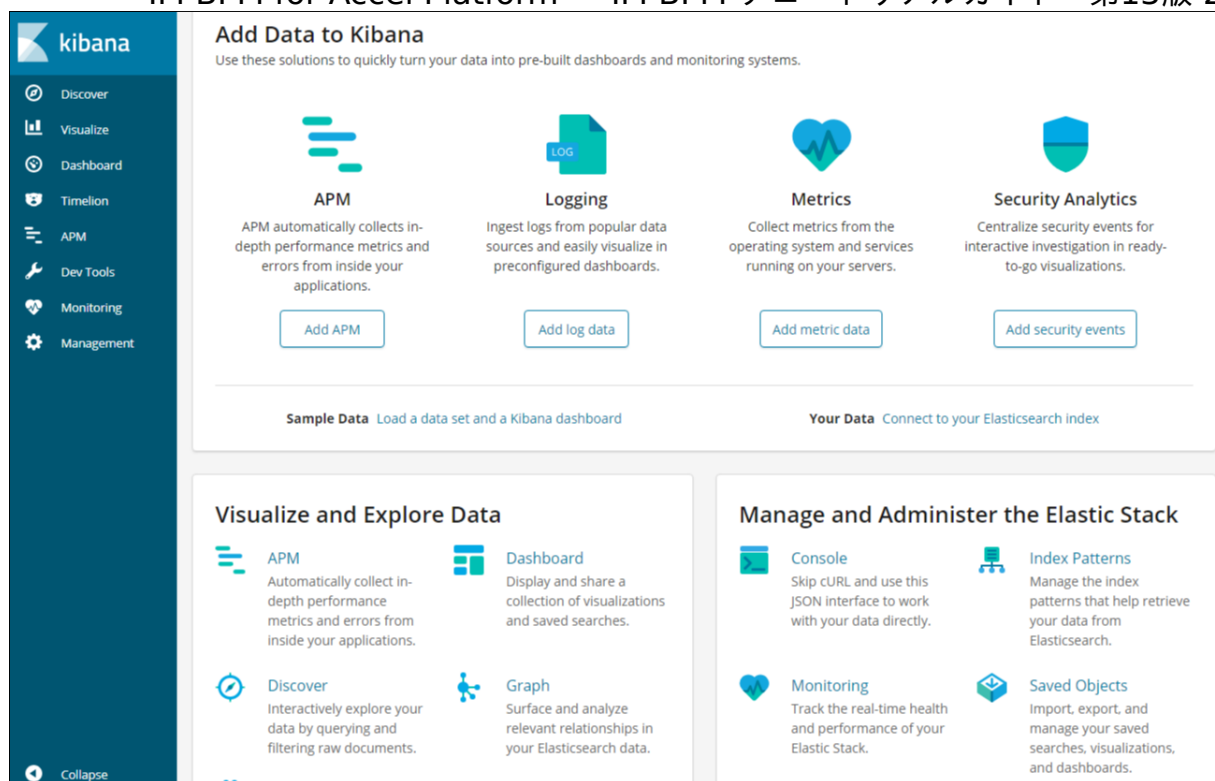
Kibanaは標準では通信の暗号化や認証機能が使用できません。Kibanaは、セキュアなネットワーク上にて運用されることを推奨します。

Kibanaの起動

<%KIBANA_HOME%/bin/kibana.bat> ファイルをダブルクリックし起動します。

起動完了後、ブラウザより、下記のURLにアクセスし、Kibanaの画面が表示されることを確認します。

<http://Kibanaのホスト名:5601/>

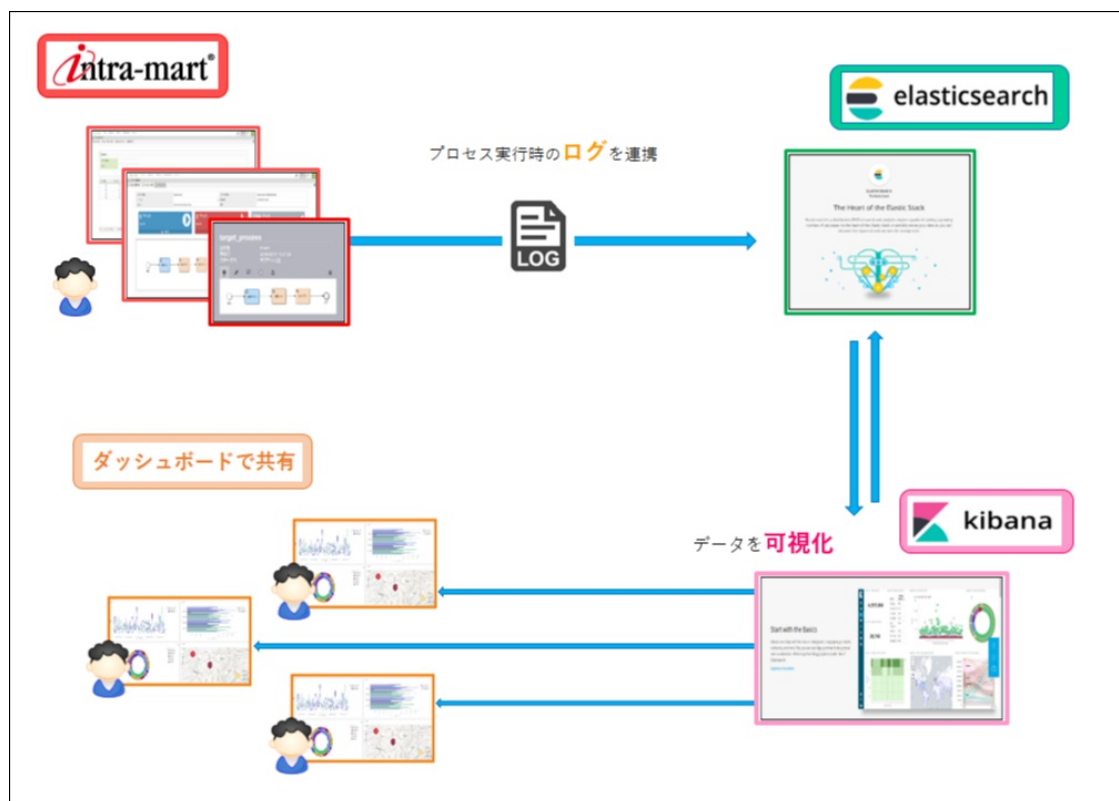


図：「Kibana」初期表示

プロセス実行とElasticsearchおよびKibanaの概要

「IM-BPM/Elasticsearch コネクタ」モジュールが含まれるintra-mart Accel Platform環境でプロセスの実行を行うと、「プロセスの開始」や、「タスクの完了」、「アクティビティの完了」、「プロセスの完了」など、特定のイベントが発生した際に、タスク名や処理を行ったユーザなどのイベントのログが自動でElasticsearchへ登録されます。

Elasticsearchへ登録された情報を、Kibanaなどのツールを用いて可視化し、ダッシュボードなどを用いて共有することで、プロセスのイベントデータをより有効に活用できます。



図：プロセスの実行とElasticsearchへの登録、Kibanaでの可視化

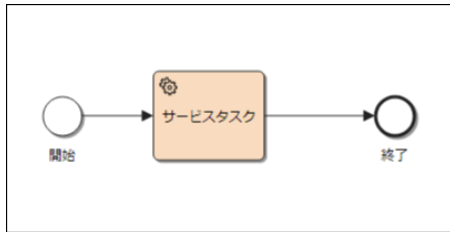
本チュートリアルで使用する時系列データを作成する

Kibana上で統計情報を可視化するにあたり、サンプルとなる時系列データを作成します。

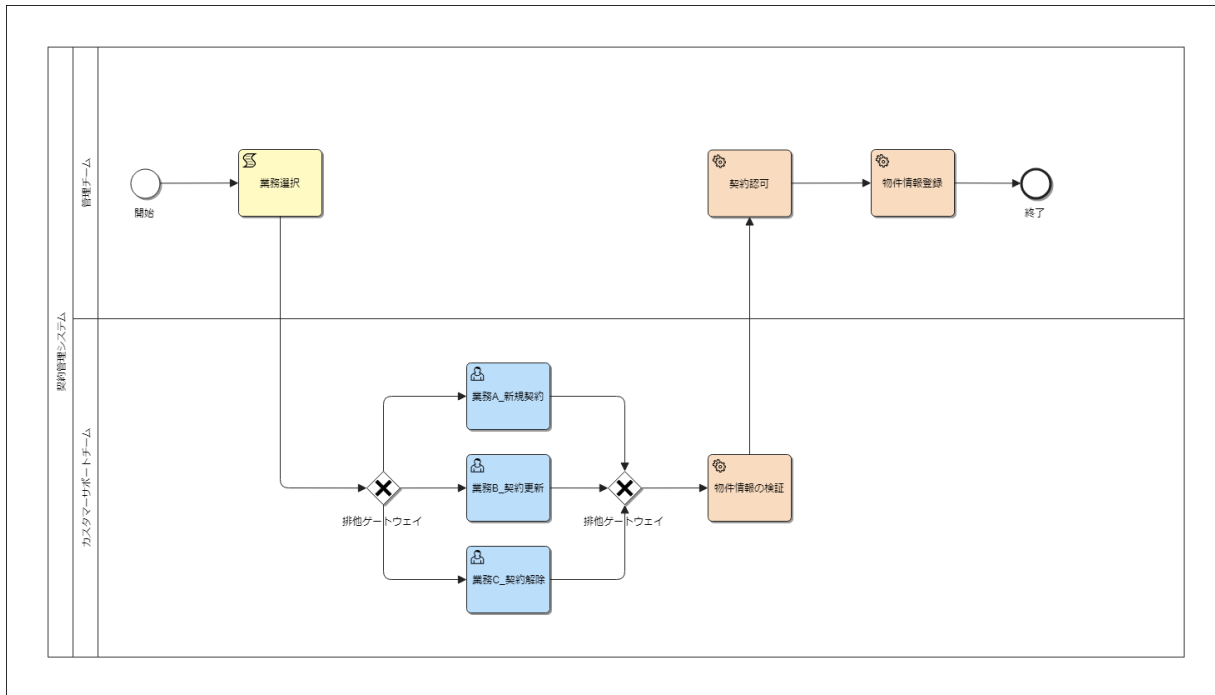
統計情報取得の対象となるプロセスのデプロイを行う

タスクの統計情報の取得対象となるプロセスをデプロイします。

本デプロイ情報には、下記の2つのプロセスと、各サービスタスクで実行するjavaプログラムが含まれます。



図：データ作成用プロセス（process_create_elasticsearch_data）



図：統計情報の取得対象となるプロセス（target_process）

- データ作成用プロセス（**process_create_elasticsearch_data**）：上記の統計情報の取得対象となるプロセスを、時刻や、タスクの担当者などの情報を変更しながら連続で実行するサービスタスクを含むプロセスです。
- 統計情報の取得対象となるプロセス（**target_process**）：ユーザタスクなど、本チュートリアルで統計情報の対象とする項目を含むプロセスです。
 - プロセス実行時の流れ
 1. 「業務選択」タスクにより、下記の3業務より実施する業務を選択します。
 - 「業務A_新規契約」
 - 「業務B_契約更新」
 - 「業務C_契約解除」
 2. 実施する業務が決まったら、「データ作成用プロセス」により下記の3ユーザより一人がタスクに割り当てられ、後述する「タスクを処理したユーザの氏名」や「タスクを処理したユーザの所属組織」などの変数情報をプロセス変数に設定したのち、30分～3日の範囲でランダムな処理時間（task_duration）を設定し、完了されます。
 - aoyagi（青柳辰巳）
 - ikuta（生田一哉）
 - ueda（上田辰男）
 3. 「物件情報の検証」タスクを自動で実行します。（50msec～400msecの時間待機を行った後、自動的に完了されます）
 4. 「契約認可」タスクを自動で実行します。（50msec～400msecの時間待機を行った後、自動的に完了されます）
 5. 「物件情報登録」タスクを自動実行します。（ランダムに選択された緯度・経度情報をプロセス変数に登録します）

i コラム

緯度・経度情報

出典：[街区レベル位置参照情報 国土交通省]

<http://nlftp.mlit.go.jp/index.html>

i コラム

変数について

各業務で設定される変数の詳細は後続の「[Elasticsearch に変数用のテンプレートを登録する](#)」を参照してください。

barファイルのデプロイ

barファイル「**CreateElasticsearchSampleData.bar**」をダウンロードし任意の場所に保存し、以下の設定にて「デプロイ」画面にてデプロイを行います。

i コラム

デプロイ画面からデプロイする方法については、「[IM-BPM ユーザ操作ガイド](#)」 - 「[デプロイする](#)」を参照してください。

項目名	設定値
デプロイ名	Elasticsearch/Kibana連携情報サンプル作成プロセスデプロイ
カテゴリ	- (入力不要)
ファイル	上記でダウンロードしたbarファイル 「 CreateElasticsearchSampleData.bar 」の保存先のパス

図：「デプロイ」画面

i コラム

デプロイに付随するJavaクラスについて

上記のbarファイルによるデプロイを行うと、各プロセスのサービスタスクに設定されているJavaクラスも同時にデプロイされます。これらのクラスはアンデプロイを行うことにより、デプロイされたプロセス定義と一緒に削除されます。

Elasticsearch に変数用のテンプレートを登録する

プロセスの実行時に、プロセスが保持している変数用のマッピング情報を含むインデックスのテンプレートをElasticsearchへ登録します。本チュートリアルでは統計処理のため、変数として以下の5つの項目をElasticsearchへ連携します。

- **selectedTaskName** :選択された業務名
- **taskCompletedUserName** :タスクを処理したユーザの氏名
- **taskCompletedUserDept** :タスクを処理したユーザの所属組織
- **taskCompletedUserDeptAddrGeo** :タスクを処理したユーザの所属組織の住所に紐づく緯度・経度情報
- **contractedAddrGeo** :業務で扱った物件情報に紐づく緯度・経度情報

! 注意

インデックスのテンプレートに関して

本項ではElasticsearchへ下記のインデックステンプレート例が登録されていることを前提としています。

インデックスのテンプレートの例に関しては、「[IM-BPM 仕様書](#)」 - 「[IM-BPM Elasticsearch インデックステンプレート例](#)」を参照してください。

Elasticsearchへのインデックスのテンプレート登録方法に関しては、「[IM-BPM for Accel Platform セットアップガイド](#)」 - 「[インデックステンプレートの登録](#)」を参照してください。

以下、本チュートリアルで追加が必要となる変数情報のマッピング情報のみを含むインデックステンプレートをElasticsearchへ、Kibanaの「Dev Tools」を使用して登録する方法を説明します。

Kibanaの「メインメニュー」→「Dev Tools」→「Console」のテキストエリアに下記のコマンドを貼り付け、実行します。

```
PUT _template/im_bpm-index_template_for_tutorial
```

```
{
  "index_patterns": "im_bpm-default-*",
  "mappings": {
    "im_bpm": {
      "properties": {
        "variables": {
          "properties": {
            "selectedTaskName": {
              "type": "keyword"
            },
            "taskCompletedUserName": {
              "type": "keyword"
            },
            "taskCompletedUserDept": {
              "type": "keyword"
            },
            "taskCompletedUserDeptAddrGeo": {
              "type": "geo_point"
            },
            "contractedAddrGeo": {
              "type": "geo_point"
            }
          }
        }
      }
    }
  }
}
```



図：「Dev Tools」→「Console」



注意

テンプレート内のインデックス名の指定について

上記のテンプレートの例はテナントIDがdefaultの場合の例です。

上記コマンド中の下記の部分の記載をご使用の環境に合わせて変更してください。

```
{
  "index_patterns": "im_bpm-default-*",
  . . .
}
```

(例) テナントIDがtestの場合、"template": "im_bpm-test-*"としてください。

以降、本チュートリアルでは、テナントIDはdefaultが設定されていると仮定して進めます。

データ作成用プロセスを実行する

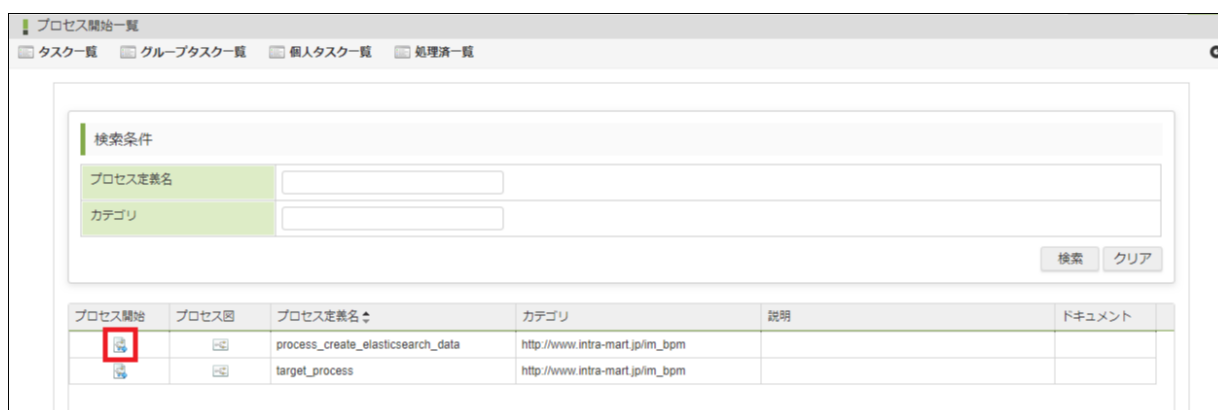
データ作成用プロセスを実行し、Elasticsearchへ統計情報の取得対象となるプロセスの実行ログをまとめて登録します。

1. 「サイトマップ」→「BPM」→「プロセス開始一覧」をクリックします。



図：「サイトマップ」→「BPM」→「プロセス開始一覧」

2. 「プロセス開始一覧」にてプロセス定義名「process_create_elasticsearch_data」のプロセスの「プロセス開始」をクリックします。



図：「プロセス開始一覧」画面



注意

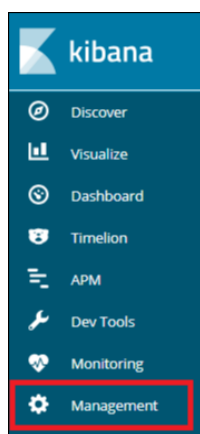
データ作成用プロセスの実行について

データ作成用プロセスは、時系列データを作成するために、実行時にプロセスエンジンの時刻情報を変更しながら実行されます。そのため、本チュートリアルを実施している環境において、データ作成用プロセスの実行中は他のプロセスの実行は行わないでください。

Elasticsearch のインデックスをKibanaから確認する

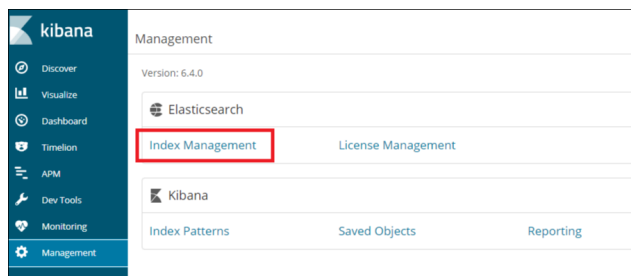
Kibana上でElasticsearchに登録されたデータを確認します。

1. Kibanaの「メインメニュー」→「Management」をクリックします。

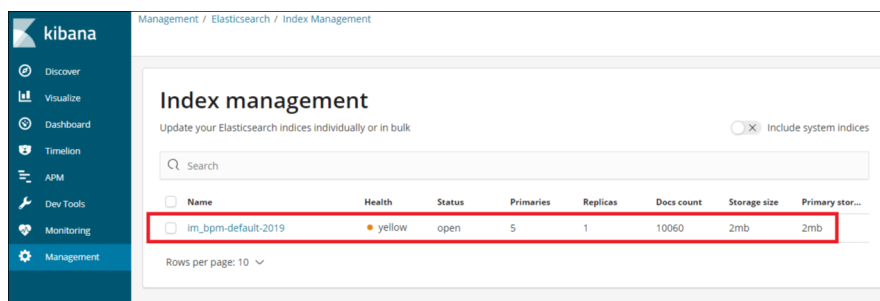


図：「Management」

2. 「Elasticsearch」→「Index Management」をクリックし、インデックス「im_bpm-default-YYYY」が存在することを確認します。



図：「Index Management」

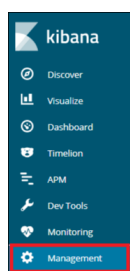


図：インデックス「im_bpm-default-YYYY」

Kibana上にインデックスパターンを作成する

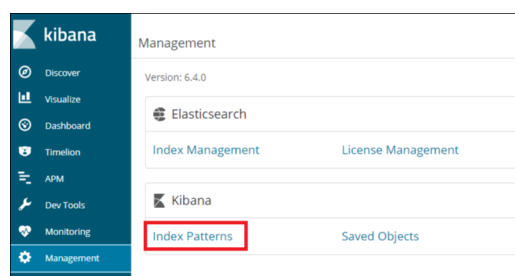
Kibana上にインデックスパターンを作成し、Elasticsearchのインデックスのフィールドを扱う際のルールの設定を行います。

1. Kibanaのメインメニューより「Management」をクリックし、「Management」画面を開きます。

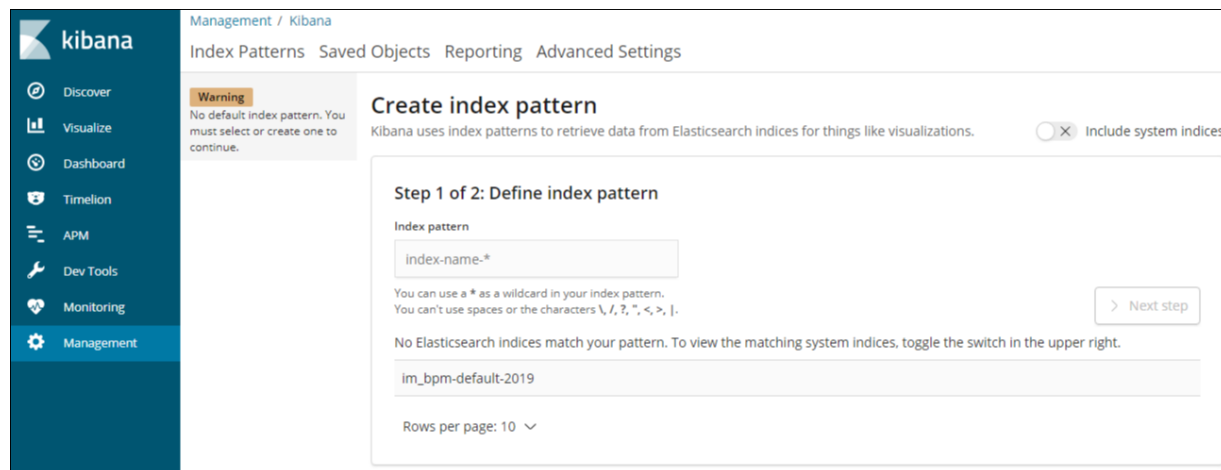


図：「Management」

2. 「Management」画面の「Kibana」→「Index Patterns」をクリックし、「Create index pattern」画面を開きます。



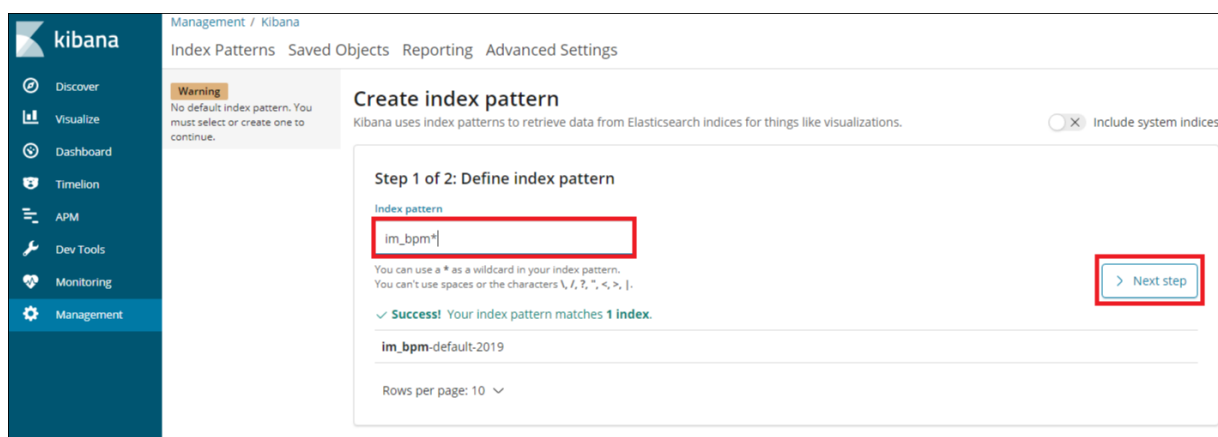
図：「Kibana」→「Index Patterns」



図：「Create index pattern」画面

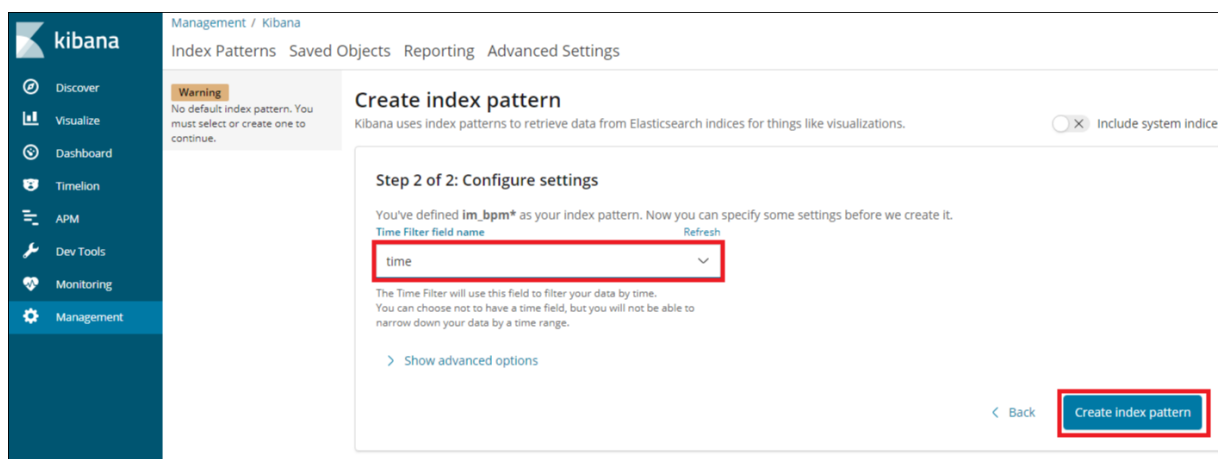
3. 「Create index pattern」画面の「Index pattern」に `im_bpm*` を指定し、「Next step」をクリックします。

項目名	設定値
Index pattern	<code>im_bpm*</code>



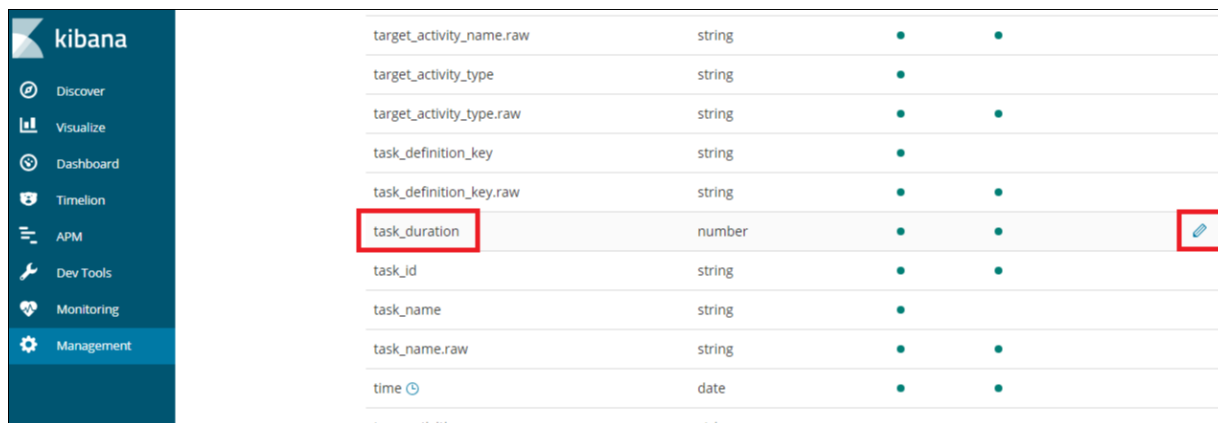
図：「Next step」をクリック

4. 「Time Filter field name」のプルダウン項目よりフィールド「time」を選択し、「Create index pattern」をクリックします。



図：「Create index pattern」をクリック

5. インデックスパターン「`im_bpm*`」の作成が完了し、フィールドの一覧が表示されたら、フィールド「`task_duration`」（タスクの所要時間）の「edit」アイコンをクリックします。

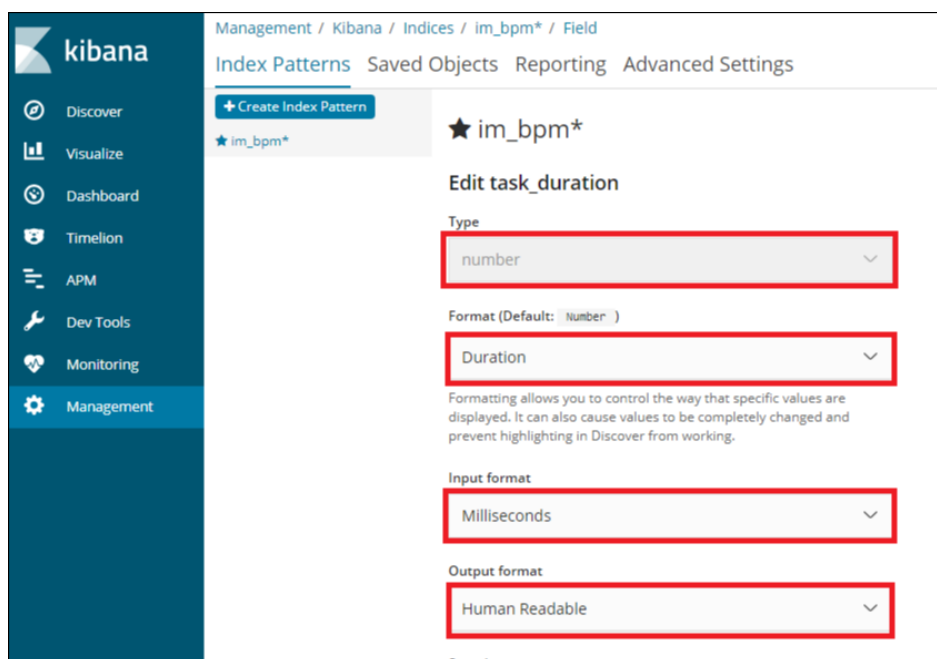


Field Name	Type	Format	Input Format	Output Format
target_activity_name.raw	string			
target_activity_type	string			
target_activity_type.raw	string			
task_definition_key	string			
task_definition_key.raw	string			
task_duration	number			
task_id	string			
task_name	string			
task_name.raw	string			
time	date			

図：「task_duration」 - 「edit」

6. 「Edit task_duration」画面の各項目に下記の値を設定します。

項目名	設定値
Type	number（変更不可）
Format (Default: Number)	Duration
Input format	Milliseconds
Output format	Human Readable



Management / Kibana / Indices / im_bpm* / Field

Index Patterns Saved Objects Reporting Advanced Settings

+ Create Index Pattern

★ im_bpm*

Edit task_duration

Type: number

Format (Default: Number): Duration

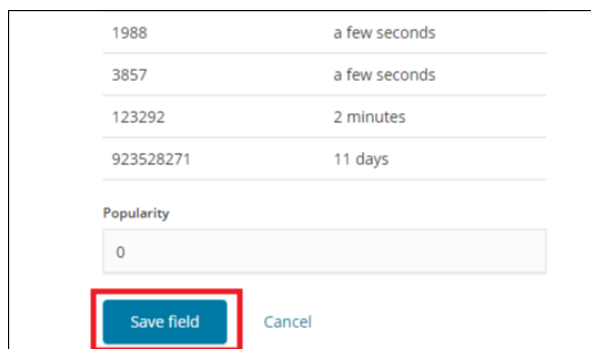
Input format: Milliseconds

Output format: Human Readable

Formatting allows you to control the way that specific values are displayed. It can also cause values to be completely changed and prevent highlighting in Discover from working.

図：「Edit task_duration」画面

7. 「Save field」をクリックし、変更を保存します。



1988	a few seconds
3857	a few seconds
123292	2 minutes
923528271	11 days

Popularity

0

Save field Cancel

図：「Save field」をクリック

8. 上記と同様の設定をフィールド「activity_duration」（アクティビティの所要時間）およびフィールド「process_duration」（プロセスの所要時間）

間) に対して行います。

★ im_bpm*

⌚ Time Filter field name: time

This page lists every field in the **im_bpm*** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the [Elasticsearch Mapping API](#).

Fields (175) Scripted fields (0) Source filters (0)

Q Filter All field types ▾

Name	Type	Format	Searchable	Aggregatable	Excluded
activity_cause_job_entity.tenantId	string		●		
activity_cause_job_entity.tenantId.raw	string		●	●	
activity_duration	number		●	●	
activity_id	string		●	●	
activity_name	string		●		
activity_name.raw	string		●	●	

図：フィールド「activity_duration」

process_cause_job_entity.retries	number		●	●	
process_cause_job_entity.revision	number		●	●	
process_cause_job_entity.tenantId	string		●		
process_cause_job_entity.tenantId.raw	string		●	●	
process_definition_id	string		●		
process_definition_id.raw	string		●	●	
process_duration	number		●	●	
process_instance_id	string		●	●	
sequence_flow_id	string		●		

図：フィールド「process_duration」

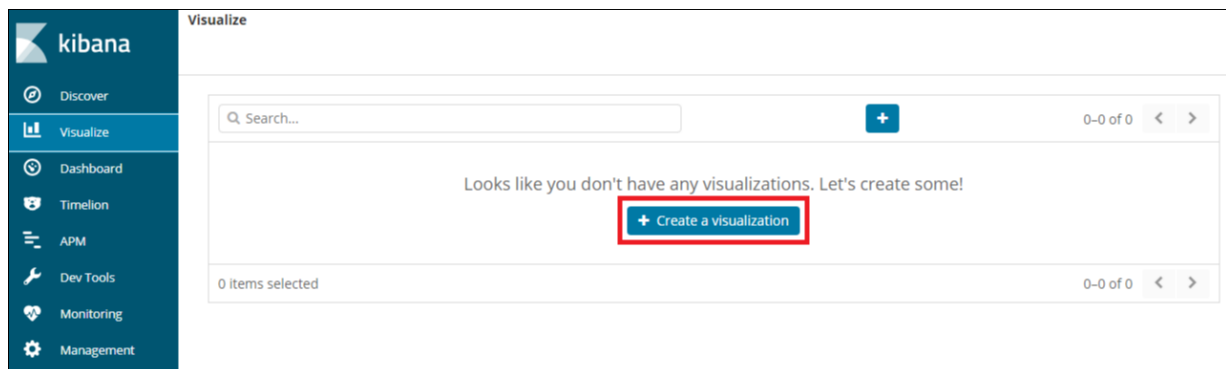
タスクの変数を使用して部署ごとの統計情報のグラフを作成する

Kibana上で、Elasticsearchへ連携されたプロセスのタスク完了イベント情報の変数情報を使用し、「部署ごとの完了タスク数の推移」と「部署ごとのタスクの平均処理時間」のグラフを作成します。

部署ごとの完了タスク数の推移

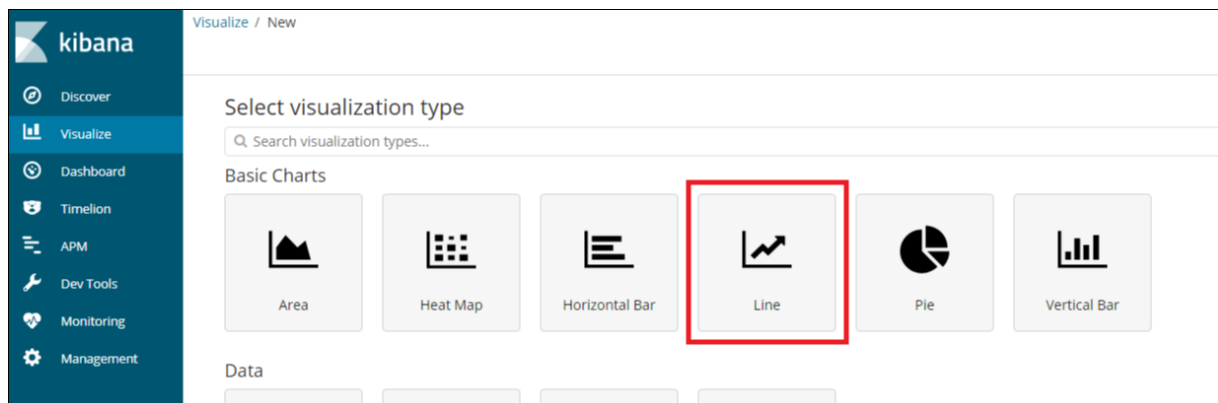
部署（処理したユーザの所属組織）ごとの、過去30日間の完了タスク数の推移のグラフを作成します。

1. Kibanaのメインメニューより「Visualize」をクリックし「Visualize」画面を開き、「Create a visualization」をクリックします。



図：「Visualize」画面

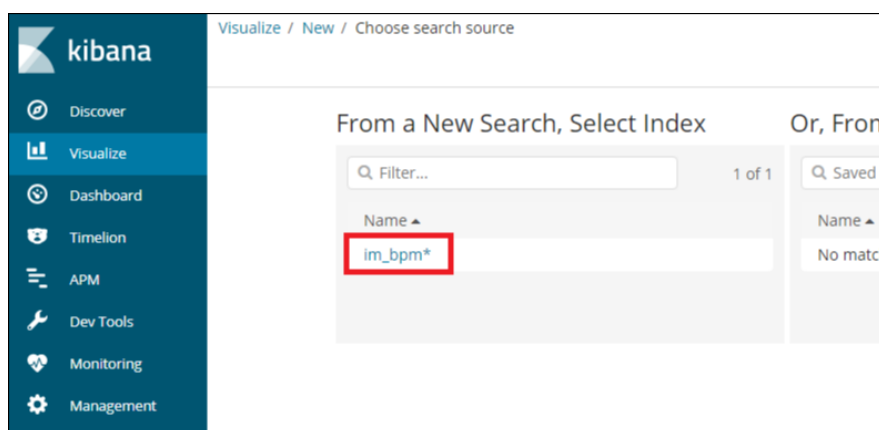
2. グラフの種類を指定します。
「Select visualization type」画面にて「Line」を選択します。



図：「Line」

3. 対象のインデックスパターンを指定します。

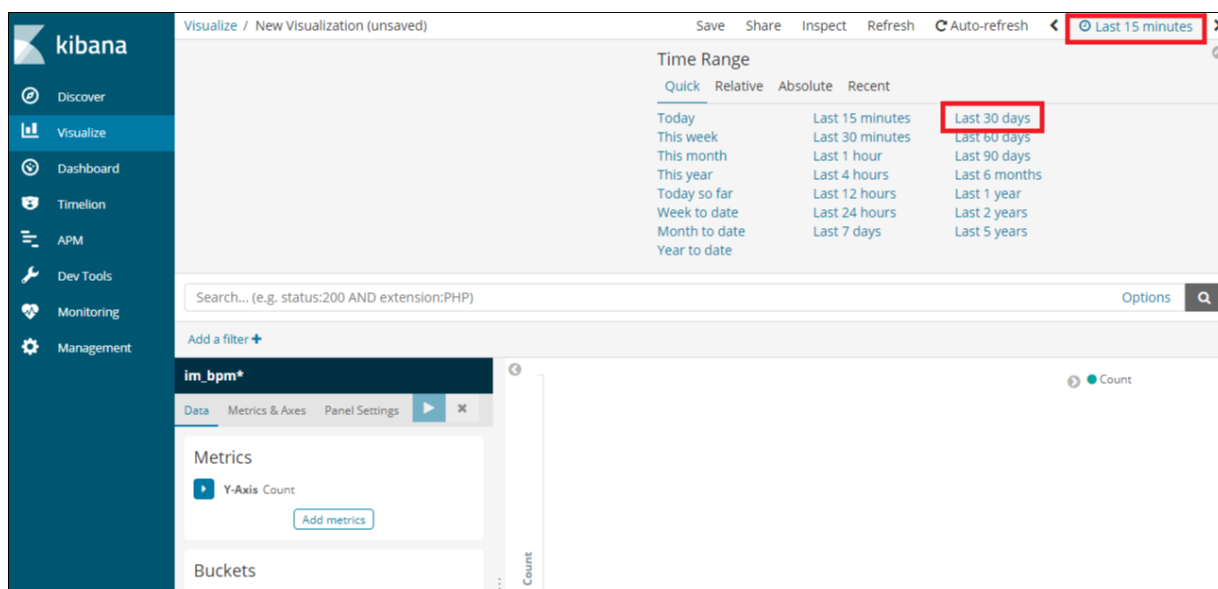
「From a New Search, Select Index」にてインデックスパターン「im_bpm*」を選択します。



図：インデックスパターン選択

4. 集計対象の期間を指定します。

「Time Range」欄より「Last 30 days」を選択します。デフォルトは「Last 15 minutes」が表示されています。

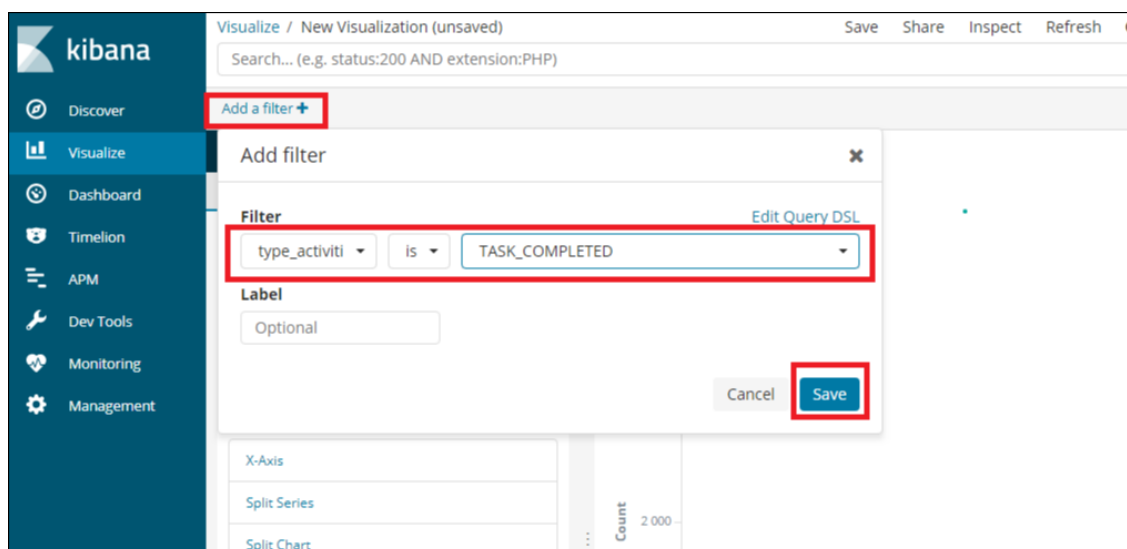


図：「Time Range」の選択

5. 集計対象のイベントとして「TASK_COMPLETED」（タスク完了イベント）を指定します。

「Add a filter」をクリックし、下記の値を設定し、「Save」をクリックします。

項目名	設定値
Fields...	type_activiti
Operators...	is
Value...	TASK_COMPLETED



図：「Add a filter」

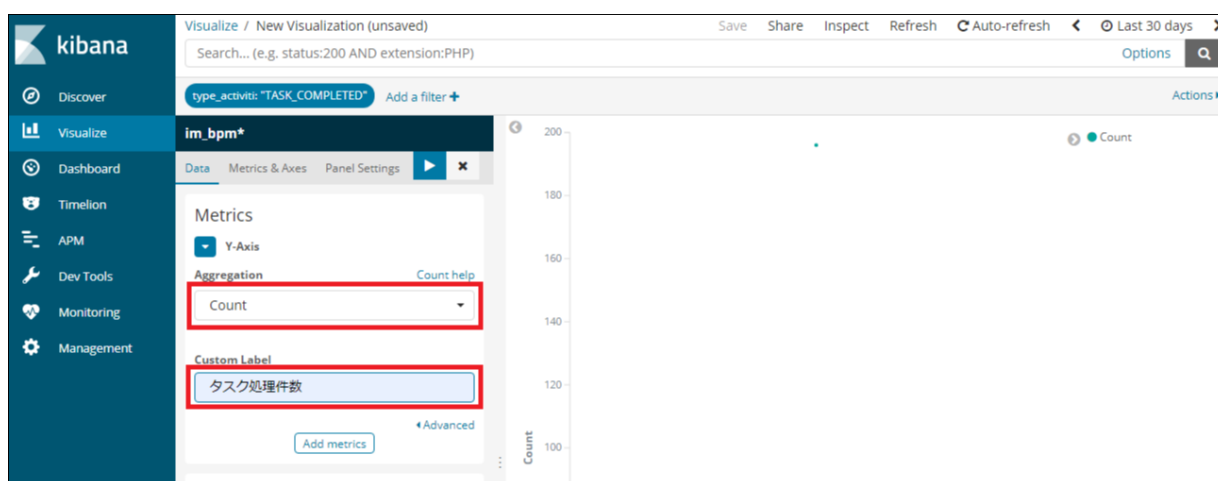
**注意**

フィールド「type_activiti」のほか、項目名が類似するフィールド「activity_type」が存在します。ここでは「type_activiti」を指定してください。

6. グラフの縦軸を指定します。

「Metrics」の「Y-Axis」をクリックし、下記の値を設定します。

項目名	設定値
Aggregation	Count
Custom Label	タスク処理件数

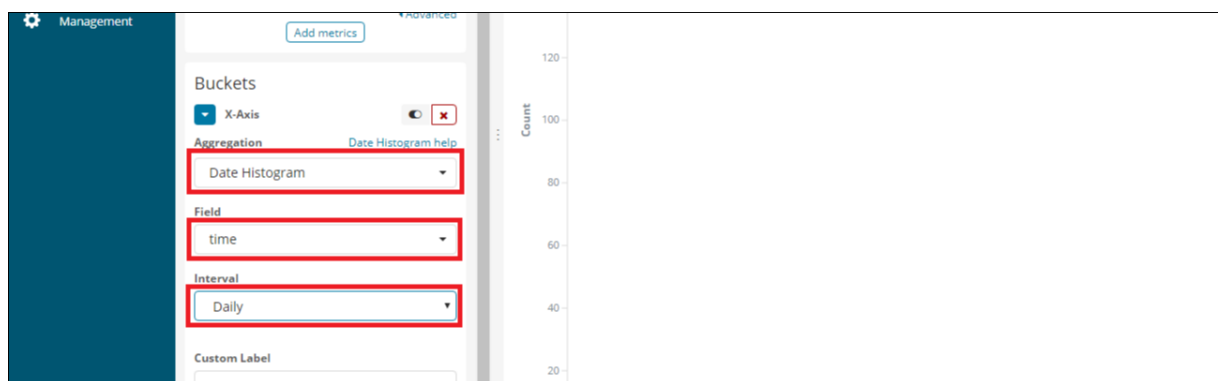


図：「Y-Axis」

7. グラフの横軸を指定します。

「Buckets」の「X-Axis」をクリックし、下記の項目を指定します。

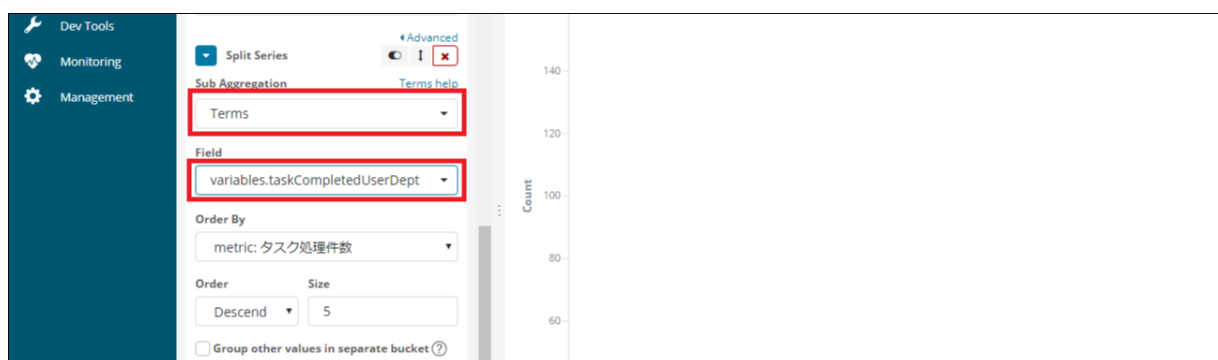
項目名	設定値
Aggregation	Date Histogram
Field	time
Interval	Daily



図：「X-Axis」

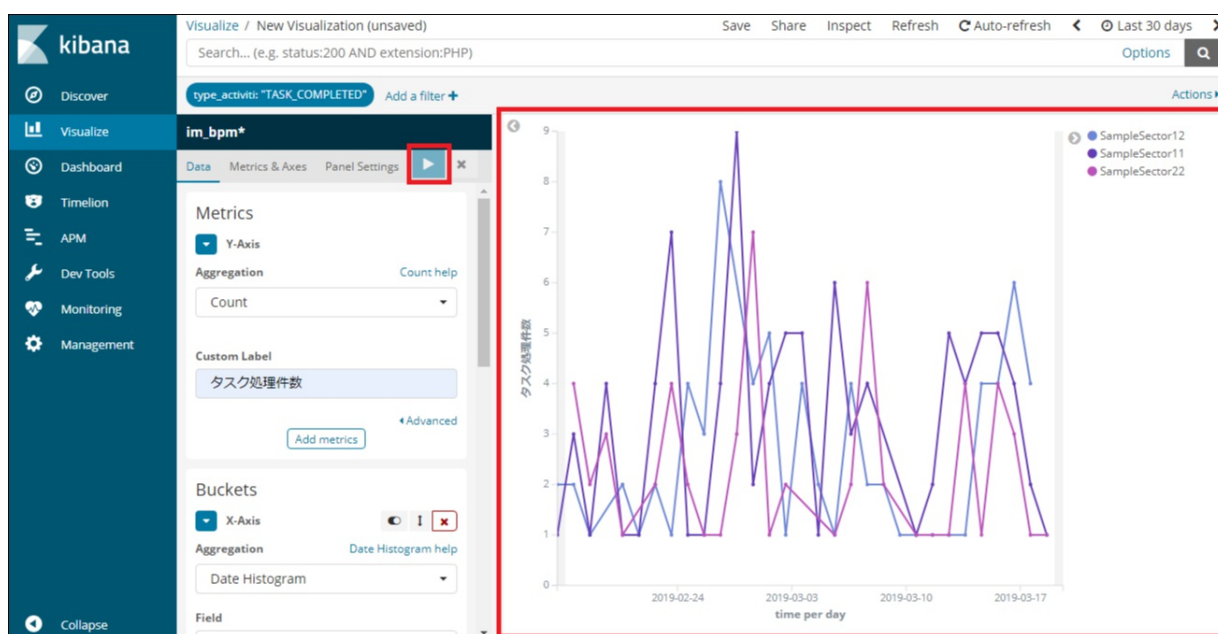
8. 部署ごとに集計結果を分割します。
「Add sub-buckets」の「Split Series」をクリックし、下記の値を設定します。

項目名	設定値
Sub Aggregation	Terms
Field	variables.taskCompletedUserDept



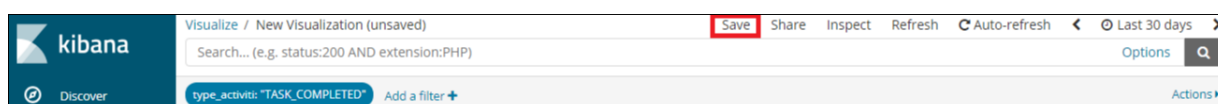
図：「Split Series」

9. 「Apply Changes」をクリックし、組織名「SampleSector11」「SampleSector12」「SampleSector22」のタスク処理件数の推移が表示されることを確認します。



図：「Apply Changes」

10. 「ヘッダメニュー」の「Save」をクリックします。



図：「Save」

11. 「Save Visualize」欄に「部署ごとの完了タスク数の推移」を設定し、「Save」をクリックして保存します。

項目名	設定値
Save Visualize	部署ごとの完了タスク数の推移

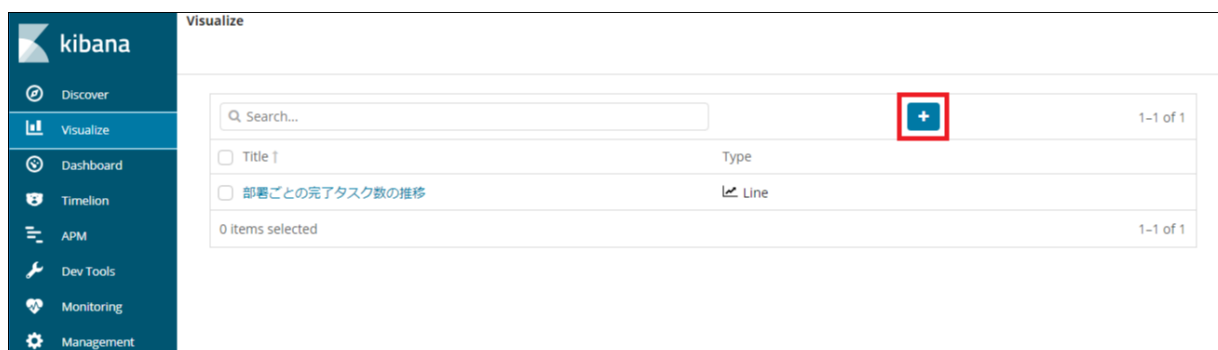


図：「Save Visualize」

部署ごとのタスクの平均所要時間

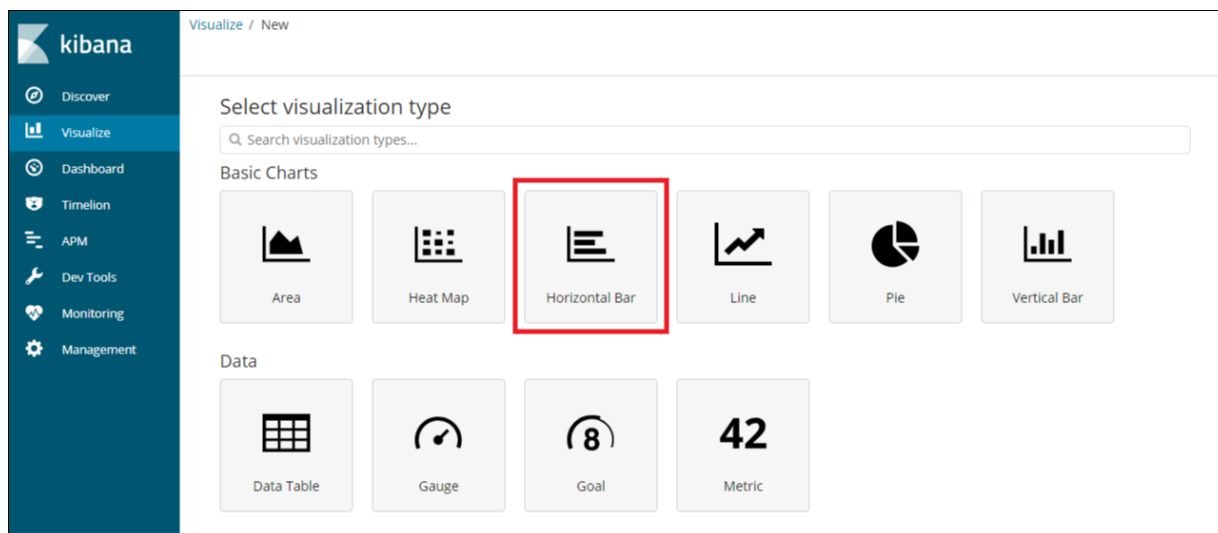
部署（処理したユーザの所属組織）ごとの過去30日間のタスクの平均所要時間のグラフを作成します。

1. Kibanaのメインメニューより「Visualize」をクリックし「Visualize」画面を開き、「+」をクリックします。



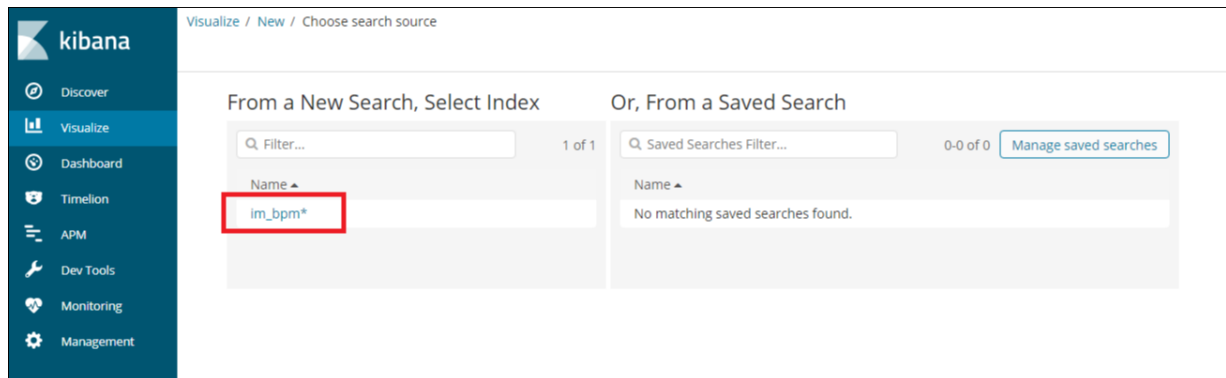
図：「Visualize」画面

2. グラフの種類を指定します。
「Select visualization type」画面にて「Horizontal Bar」を選択します。



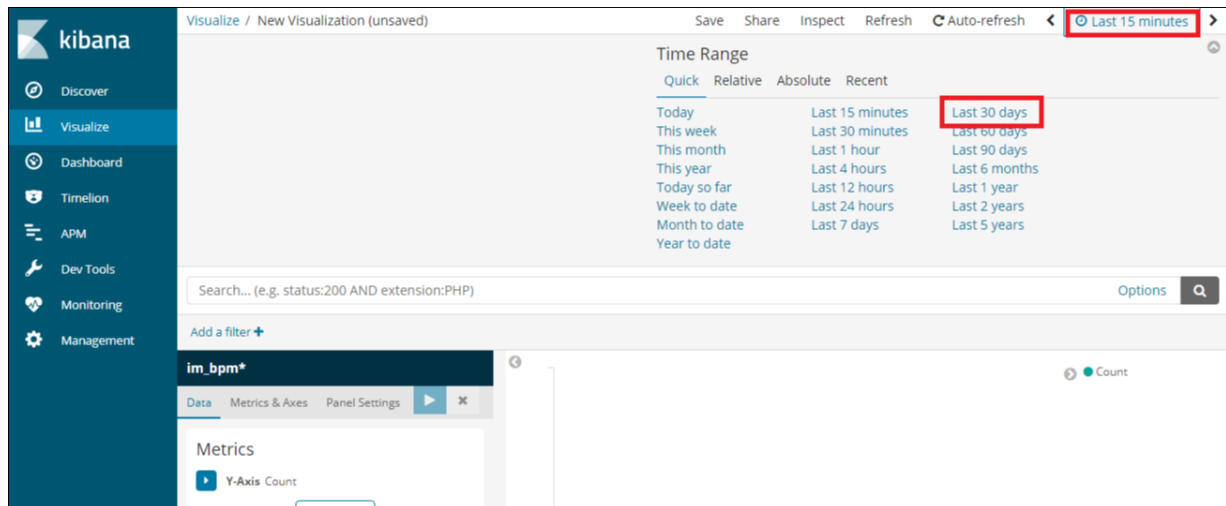
図：「Horizontal Bar」

3. 対象のインデックスパターンを指定します。
「From a New Search, Select Index」にてインデックスパターン「im_bpm*」を選択します。



図：インデックスペターン選択

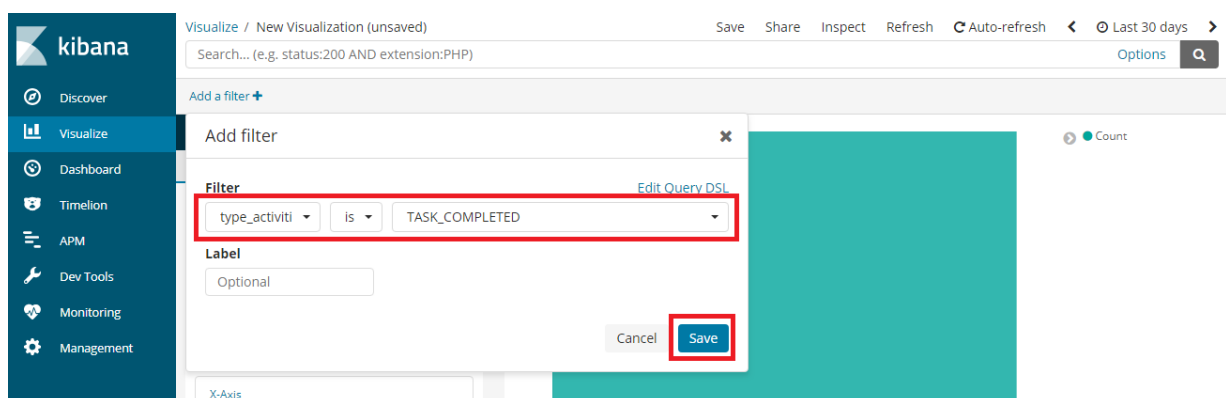
4. 集計対象の期間を指定します。
「Time Range」欄より「Last 30 days」を選択します。



図：「Time Range」の選択

5. 集計対象のイベントとして「TASK_COMPLETED」（タスク完了イベント）を指定します。
「Add a filter」をクリックし、下記の値を設定し、「Save」をクリックします。

項目名	設定値
Fields...	type_activiti
Operators...	is
Value...	TASK_COMPLETED



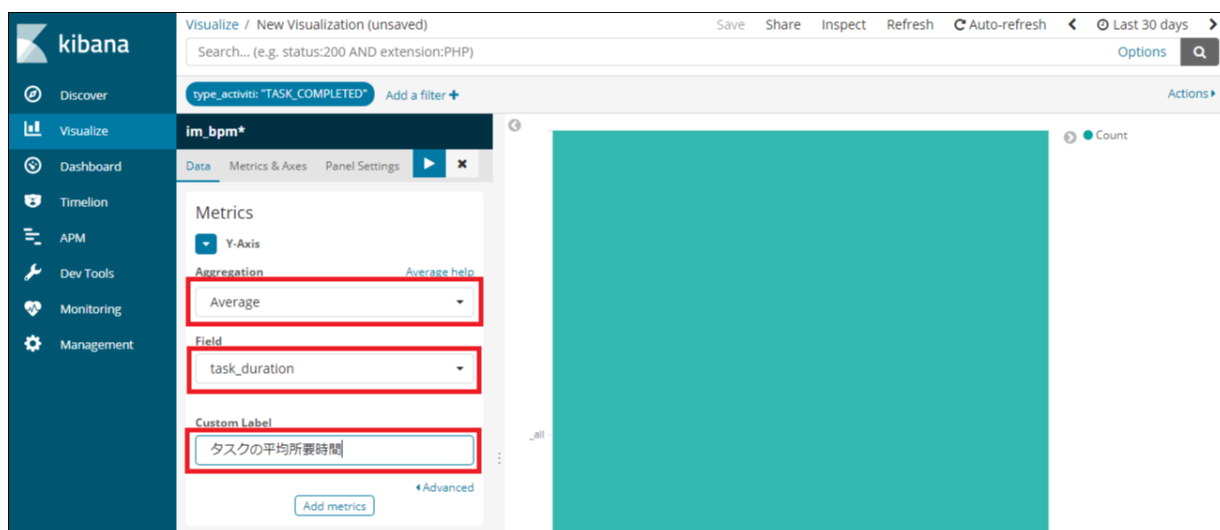
図：「Add a filter」

**注意**

フィールド「type_activiti」のほか、項目名が類似するフィールド「activity_type」が存在します。ここでは「type_activiti」を指定してください。

6. グラフの横軸を指定します。
「Metrics」の「Y-Axis」をクリックし、「Aggregation」に「Average」を、「Field」に「task_duration」を、「Custom Label」に「タスクの

項目名	設定値
Aggregation	Average
Field	task_duration
Custom Label	タスクの平均所要時間

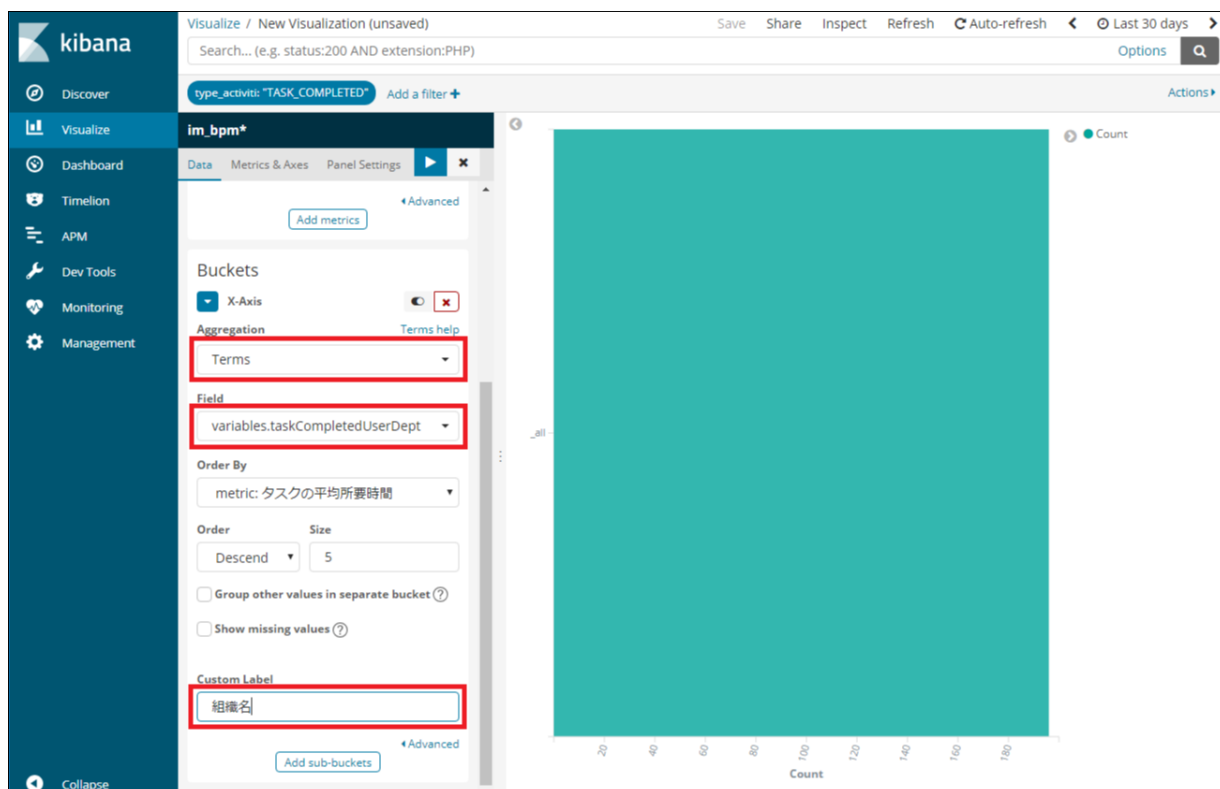


図：「Y-Axis」

7. グラフの縦軸を指定します。

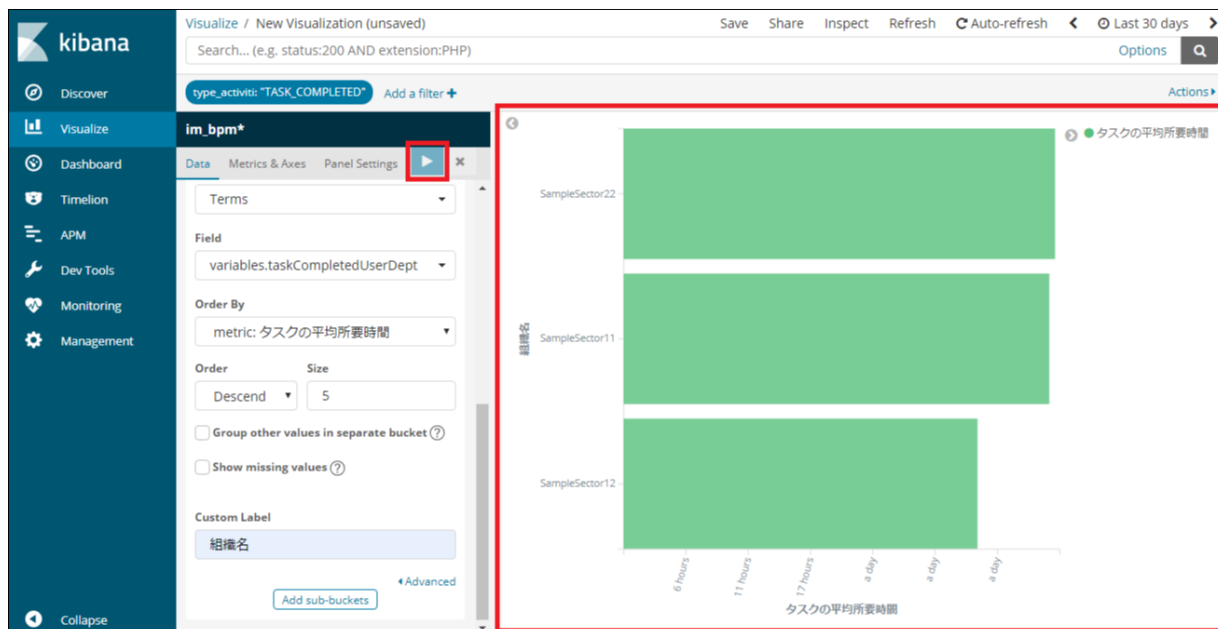
「Buckets」の「X-Axis」をクリックし、下記の値を設定します。

項目名	設定値
Aggregation	Terms
Field	variables.taskCompletedUserDept
Custom Label	組織名



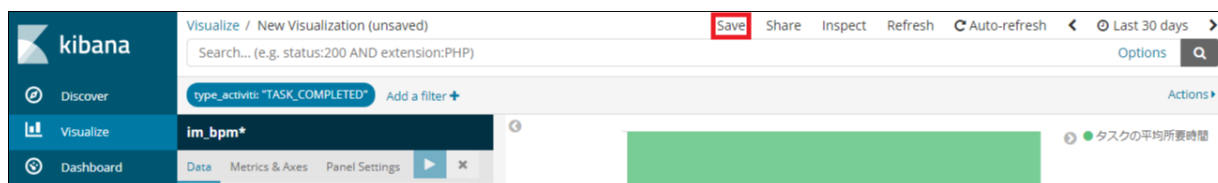
図：「X-Axis」

8. 「Apply Changes」をクリックし、組織名「SampleSector11」「SampleSector12」「SampleSector22」のタスクの平均所要時間が表示されることを確認します。



図：「Apply Changes」

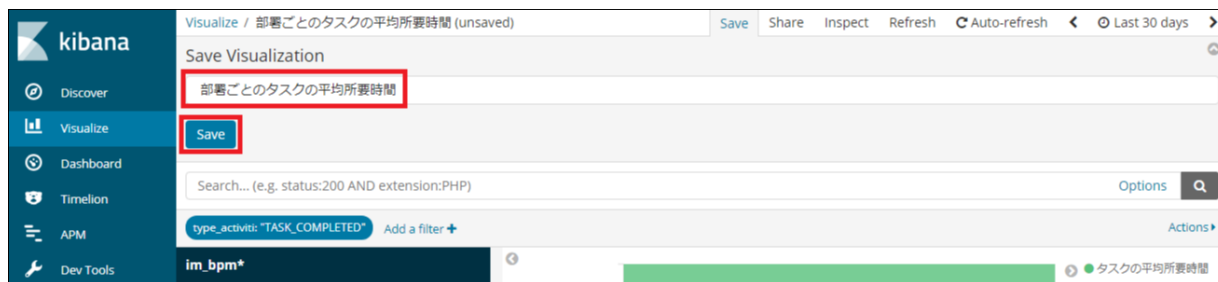
9. ヘッドメニューの「Save」をクリックします。



図：「Save」

10. 「Save Visualize」欄に「部署ごとのタスクの平均所要時間」を指定し「Save」をクリックし保存します。

項目名	設定値
Save Visualize	部署ごとのタスクの平均所要時間

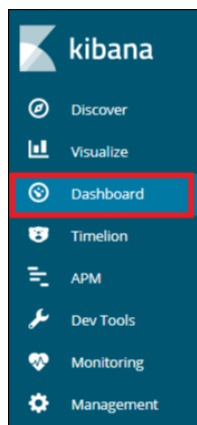


図：「Save Visualize」

Kibana上にダッシュボードを作成する

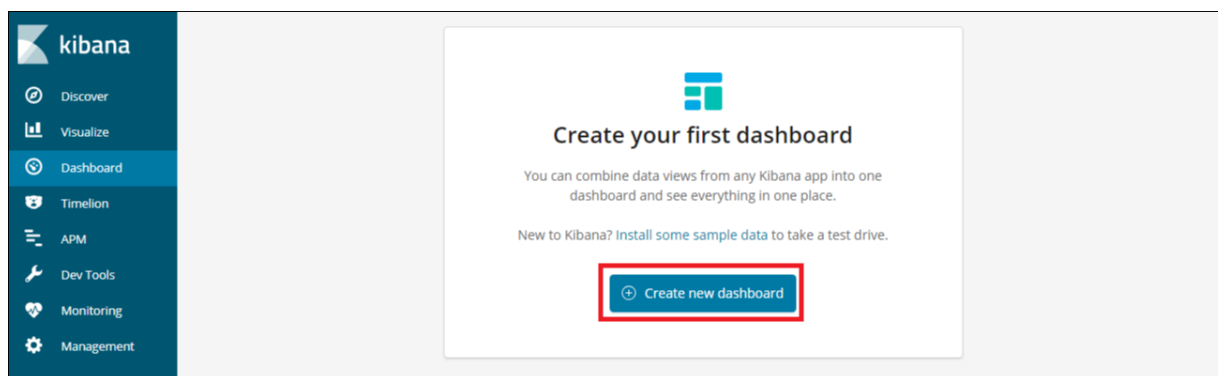
Kibana上に上記で作成した統計情報を表示するダッシュボードを作成します。

1. Kibanaのメインメニューより「Dashboard」をクリックし、「Dashboards」画面を開きます。

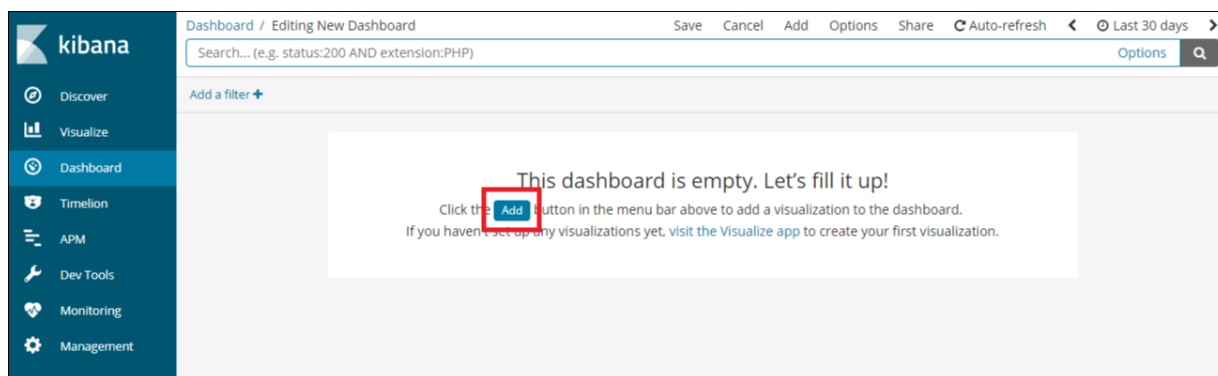


図：「Dashboard」

2. 「Create new dashboard」をクリックし、「Editing New Dashboard」画面を開き、「Add」をクリックし、「Add Panels」を開きます。

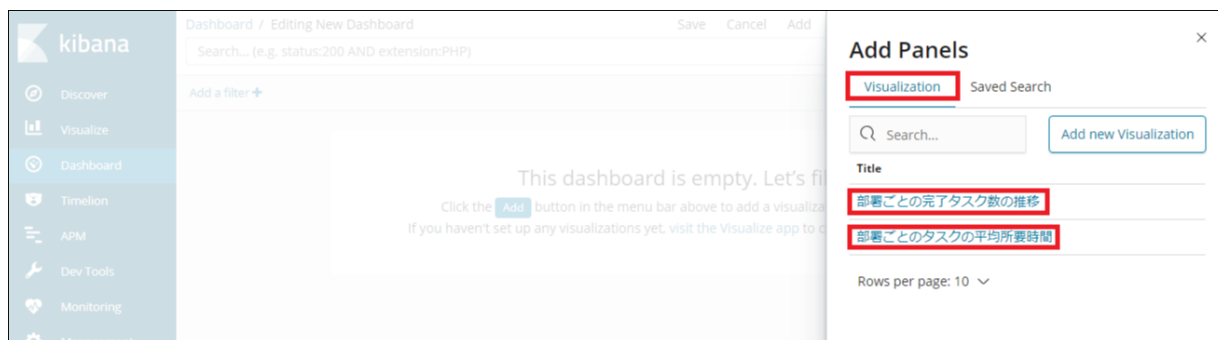


図：「Create new dashboard」



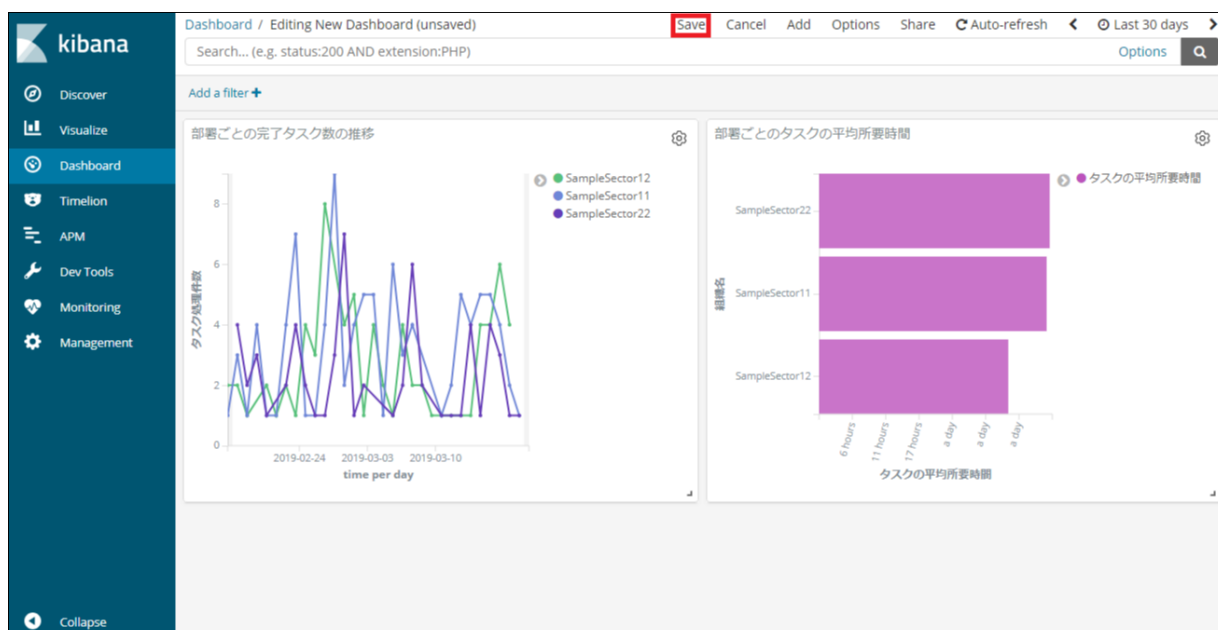
図：「Editing New Dashboard」→「Add」

3. 「Add Panels」→「Visualization」タブより「部署ごとのタスクの平均所要時間」および「部署ごとの完了タスク数の推移」をクリックします。



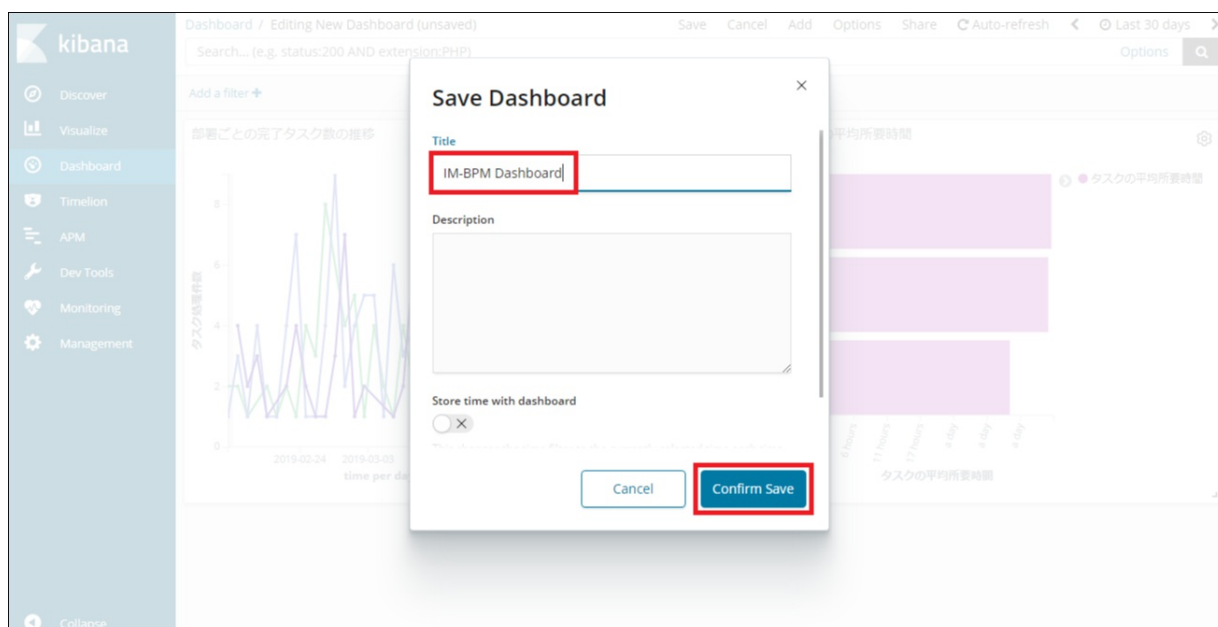
図：「Add Panels」→「Visualization」

4. ダッシュボードに「部署ごとのタスクの平均所要時間」および「部署ごとの完了タスク数の推移」が追加されたことを確認し、ヘッダメニューの「Save」をクリックします。



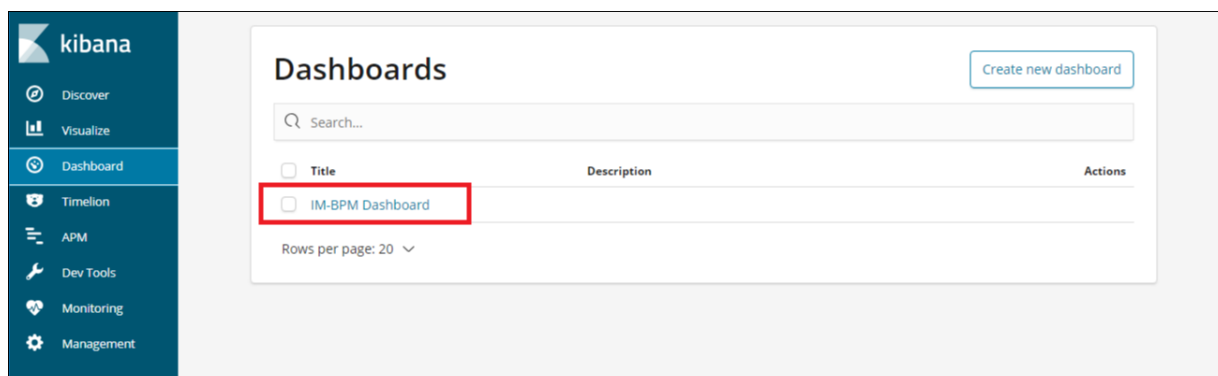
図：「Save」

5. 「Save Dashboard」ダイアログの「Title」に「IM-BPM Dashboard」を指定し、「Confirm Save」をクリックします。



図：「Save Dashboard」

6. Kibanaのメインメニューの「Dashboard」をクリックし、「Dashboards」画面を開き、一覧に「IM-BPM Dashboard」が存在することを確認します。



図：「Dashboards」画面

コラム

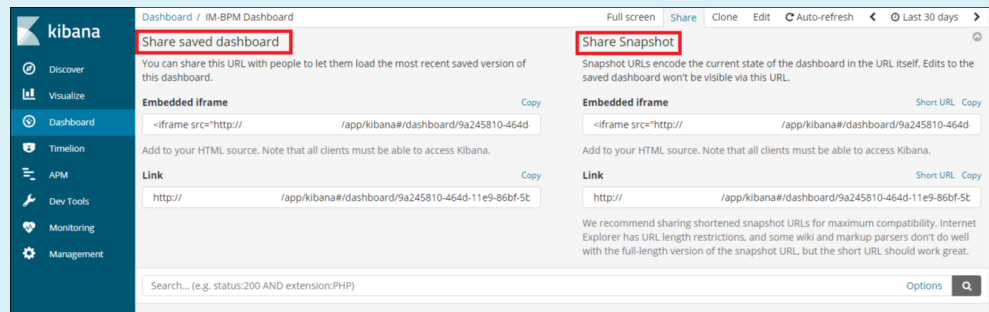
ダッシュボードの共有について

Kibanaがローカル環境以外からもアクセス可能な設定となっている場合、ダッシュボードをほかの環境から参照できます。

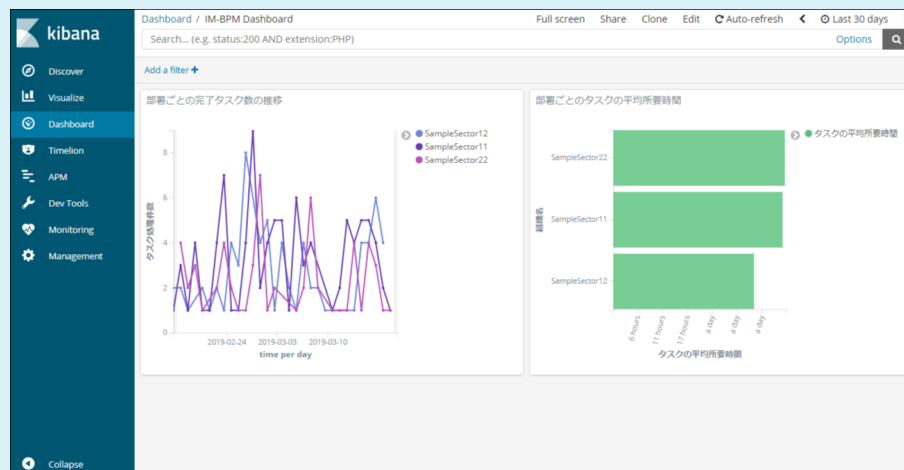
ダッシュボードを開いた状態で、右上の「Share」をクリックし、表示される各種の情報を共有してください。

「Share」にはダッシュボードの最新情報へのリンクのほか、ダッシュボードを表示している時点の断面の情報（スナップショット）へアクセスするための情報があり、それぞれに対して下記の2つの情報が表示されます。

- iframeへの表示を想定し、メニュー項目のない画面へのリンクを含むHTMLタグ
- Kibanaの「ダッシュボード」画面へのリンクのURL



図：「Share」

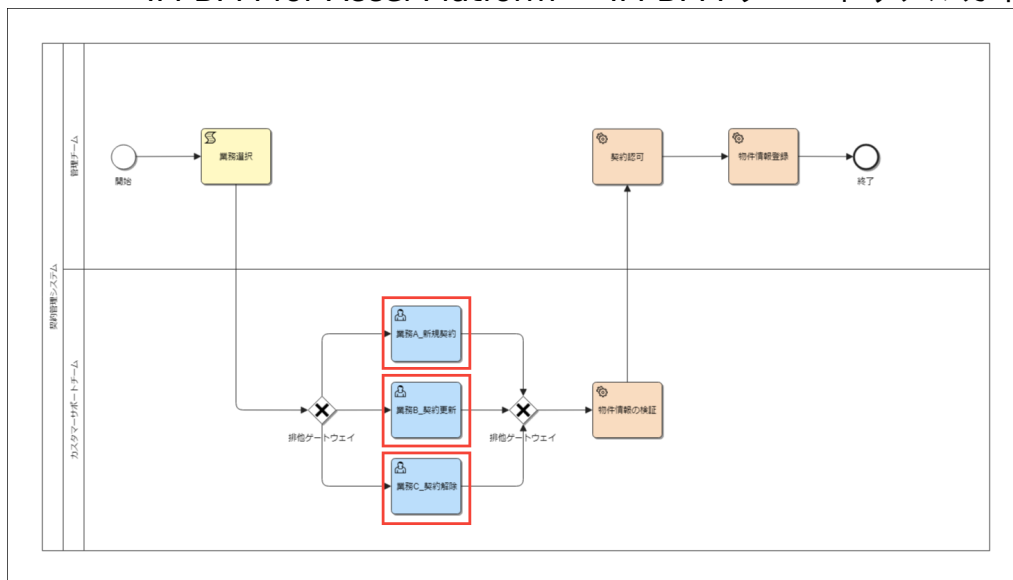


図：Kibanaの「ダッシュボード」画面

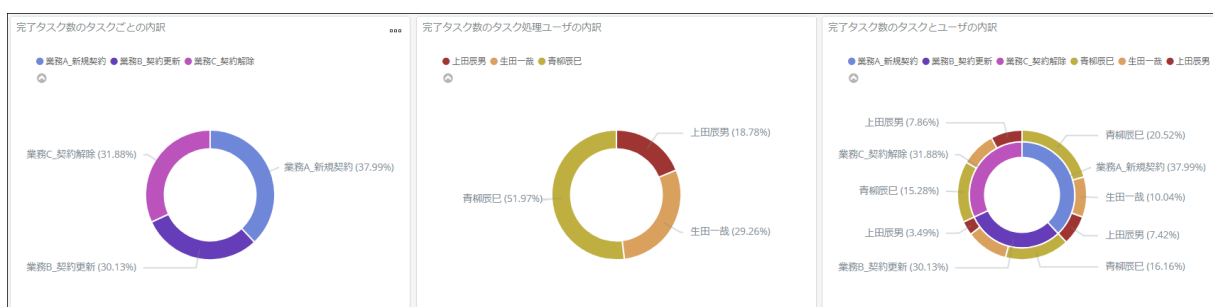
円グラフを使用して情報の内訳を表示する

Kibanaでは、Elasticsearchへ登録されたタスクやアクティビティ、変数などの情報の様々な観点における内訳を円グラフを用いて可視化できます。

このチュートリアルでは「IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する」の「統計情報取得の対象となるプロセスのデプロイを行う」でデプロイした、「統計情報の取得対象となるプロセス（target_process）」の「業務A_新規契約」「業務B_契約更新」「業務C_契約解除」タスクを対象として、「完了タスク数のタスクごとの内訳」、「完了タスク数のタスク処理ユーザの内訳」、「完了タスク数のタスクとユーザの内訳」を円グラフで可視化し、Kibanaのダッシュボードに追加する方法を解説します。



図：「業務A_新規契約」、「業務B_契約更新」、「業務C_契約解除」



図：「完了タスク数のタスクごとの内訳」、「完了タスク数のタスク処理ユーザの内訳」、「完了タスク数のタスクとユーザの内訳」

**注意**

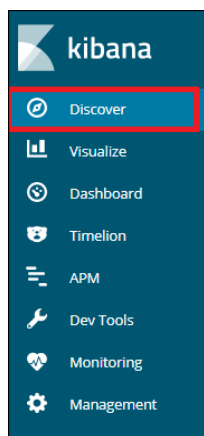
本チュートリアルは、事前に「IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する」の実施が完了していることを前提とします。

- 「Discover」でデータの検出を保存する
- 「完了タスク数のタスクごとの内訳」を円グラフで可視化する
- 「完了タスク数のタスク処理ユーザの内訳」を円グラフで可視化する
- 「完了タスク数のタスクとユーザの内訳」を2階層の円グラフで可視化する
- 「完了タスク数のタスクごとの内訳」、「完了タスク数のタスク処理ユーザの内訳」、「完了タスク数のタスクとユーザの内訳」をダッシュボードに追加する

「Discover」でデータの検出を保存する

あらかじめフィルタ条件が設定されたデータセットを作成します。

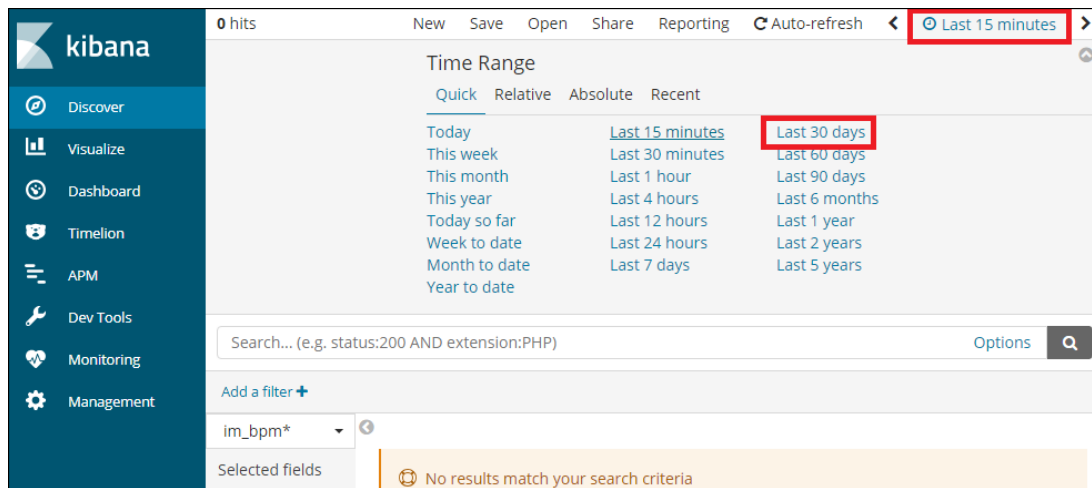
- 「Kibana」のメインメニューより「Discover」をクリックし、「Discover」画面を開きます。



図：「Discover」

2. 集計対象の期間を指定します。

「Time Range」欄より「Last 30 days」を選択します。

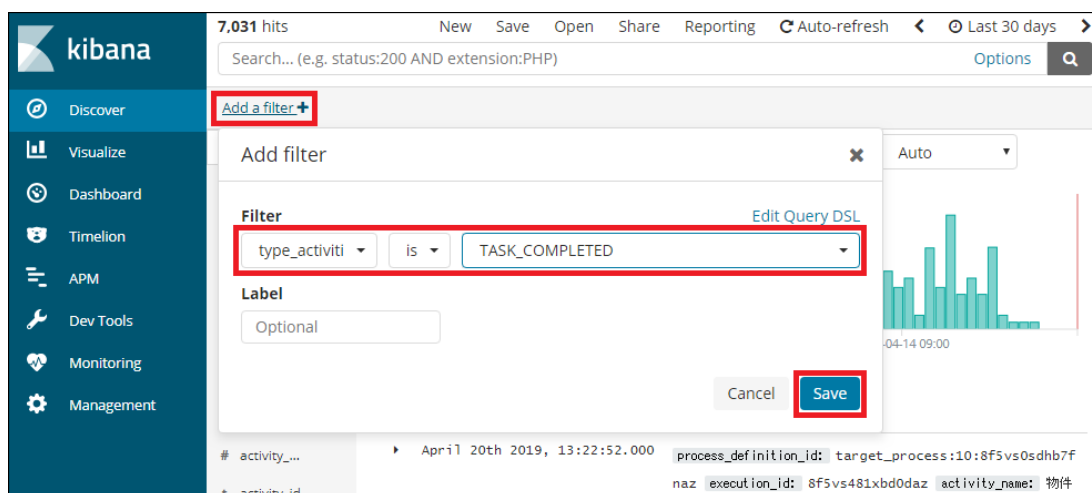


図：「Time Range」の選択

3. 集計対象のイベントとして「TASK_COMPLETED」（タスク完了イベント）を指定します。

「Add a filter」をクリックし、下記の値を設定し、「Save」をクリックします。

項目名	設定値
Fields...	type_activiti
Operators...	is
Values...	TASK_COMPLETED



図：「Add a filter」

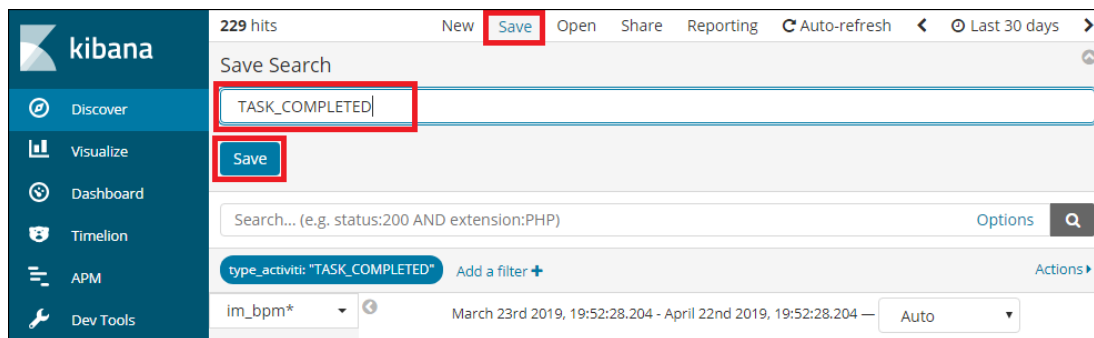


注意

フィールド「type_activiti」のほか、項目名が類似するフィールド「activity_type」が存在します。ここでは「type_activiti」を指定してください。

4. データセットに名前を付け保存します。ここではフィルタ条件とした「TASK_COMPLETED」をそのままデータセットの名前として指定します。「ヘッダメニュー」の「Save」をクリックし、「Save Search」欄に「TASK_COMPLETED」を設定し、「Save」をクリックして保存します。

項目名	設定値
Save Search	TASK_COMPLETED

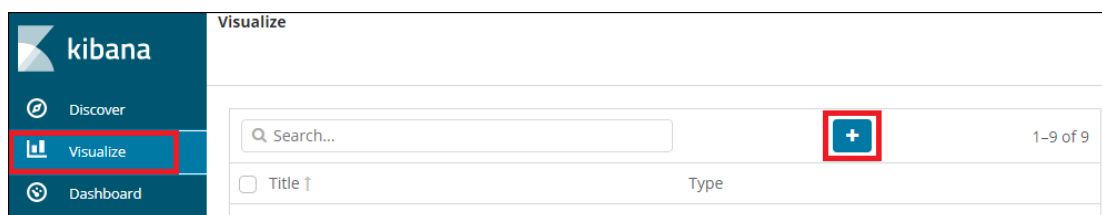


図：「Save Search」

「完了タスク数のタスクごとの内訳」を円グラフで可視化する

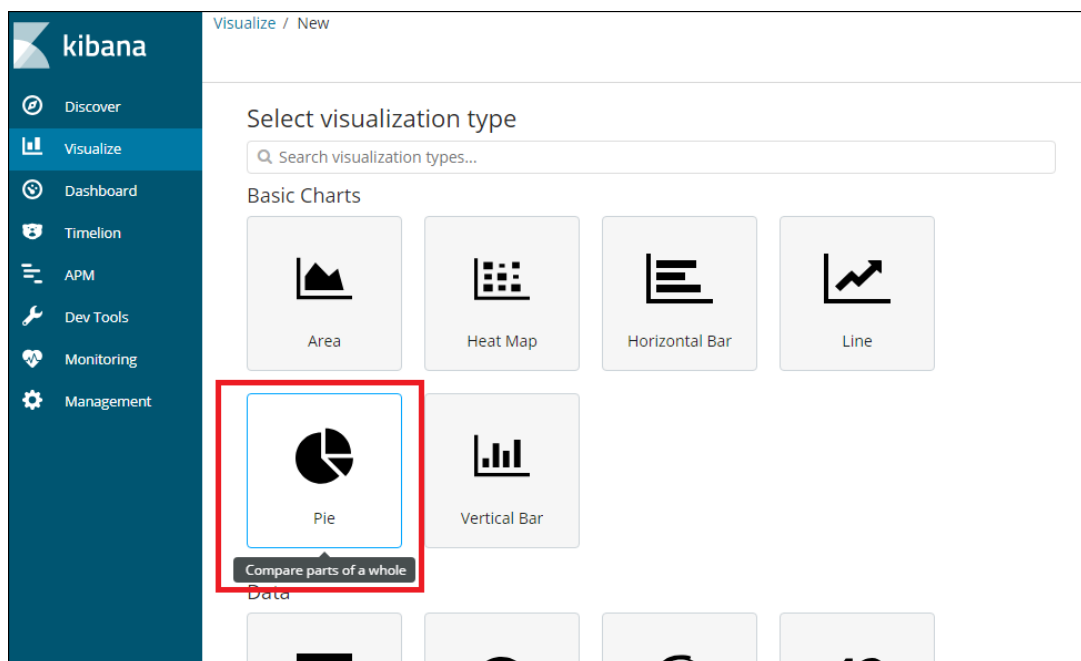
完了したタスク数のタスクごとの内訳を、円グラフを使用して可視化する方法を説明します。

1. 「Kibana」のメインメニューより「Visualize」をクリックし「Visualize」画面を開き、「+」をクリックします。



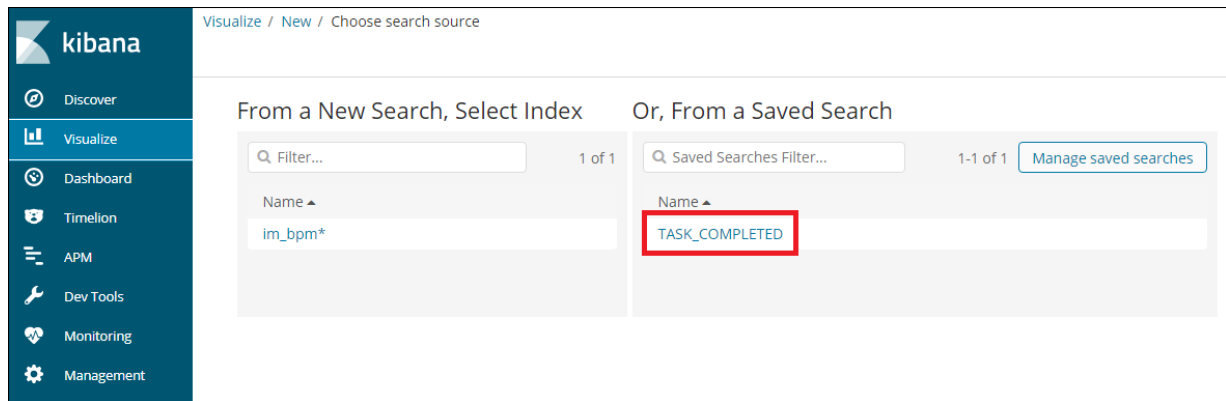
図：「Visualize」

2. グラフの種類を指定します。
「Select visualization type」画面にて「Pie」を選択します。



図：「Pie」

3. 可視化の対象を選択します。
上記で作成したフィルタ条件を設定済みのデータセット「TASK_COMPLETED」を可視化の対象として指定します。
「Or, From a Saved Search」より「TASK_COMPLETED」を選択します。

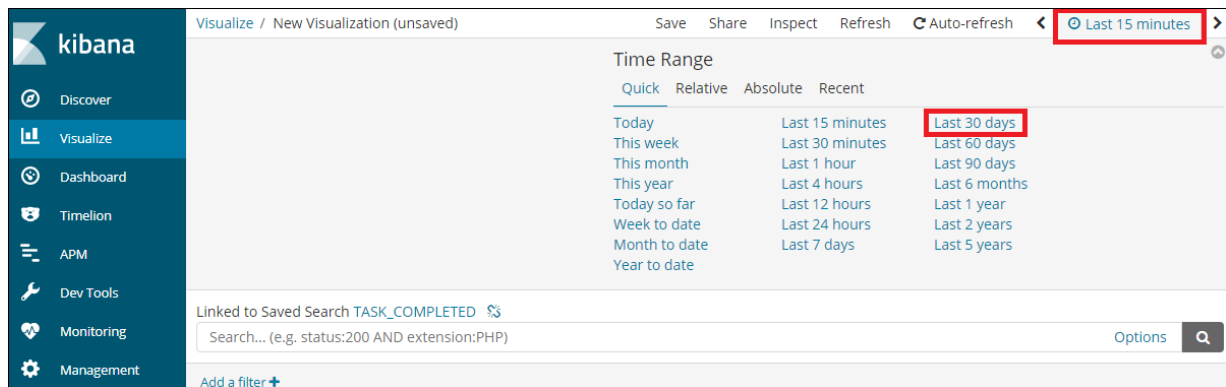


図：「Or, From a Saved Search」

4. 集計対象の期間を指定します。

Kibanaでは、データセットのフィルタ条件に期間が設定されている場合でも、現在のセッション中の「Time Range」の設定が優先される場合があるため、再度、期間を指定します。

「Time Range」欄より「Last 30 days」を選択します。

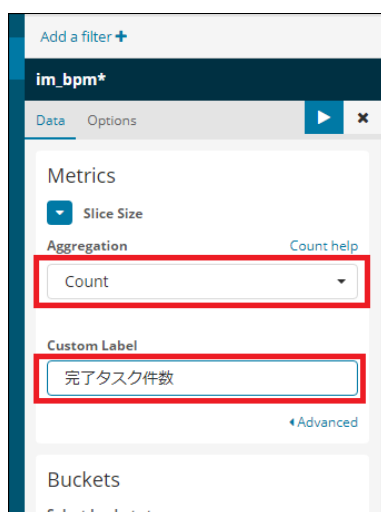


図：「Time Range」の選択

5. 完了したタスクの件数を集計対象に指定します。

「Metrics」→「Slice Size」をクリックし、下記の値を設定します。

項目名	設定値
Aggregation	Count
Custom Label	完了タスク件数



図：「Metrics」→「Slice Size」

6. 「業務A_新規契約」、「業務B_契約更新」、「業務C_契約解除」の各タスクごとに件数の合計を分割します。

「Buckets」→「Split Slices」をクリックし、下記の値を設定します。

項目名	設定値
Aggregation	Terms

項目名	設定値
Field	task_name.raw
Order By	Alphabetical
Order	Ascending
Size	5
Custom Label	タスク名

図：「Buckets」→「Split Slices」

コラム

「Order By」の「Size」指定について

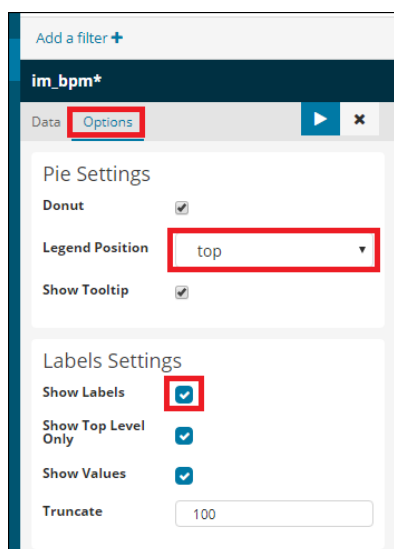
「Order By」で指定した方法にてソートを行った後、上位何件を表示するかを「Size」で指定します。

ここではアルファベット昇順でソートした結果の上位5件を表示します。

上記のSizeの指定で、ソート結果の上位X件を表示する設定（上記例では5件）を行った際、上位X件から漏れたデータは、「Group other values in separate bucket」チェックボックスにチェックを入れると、集計結果を合算して表示できます。ラベル名は「Label for other bucket」で指定したラベル名（デフォルトはOther）で表示されます。

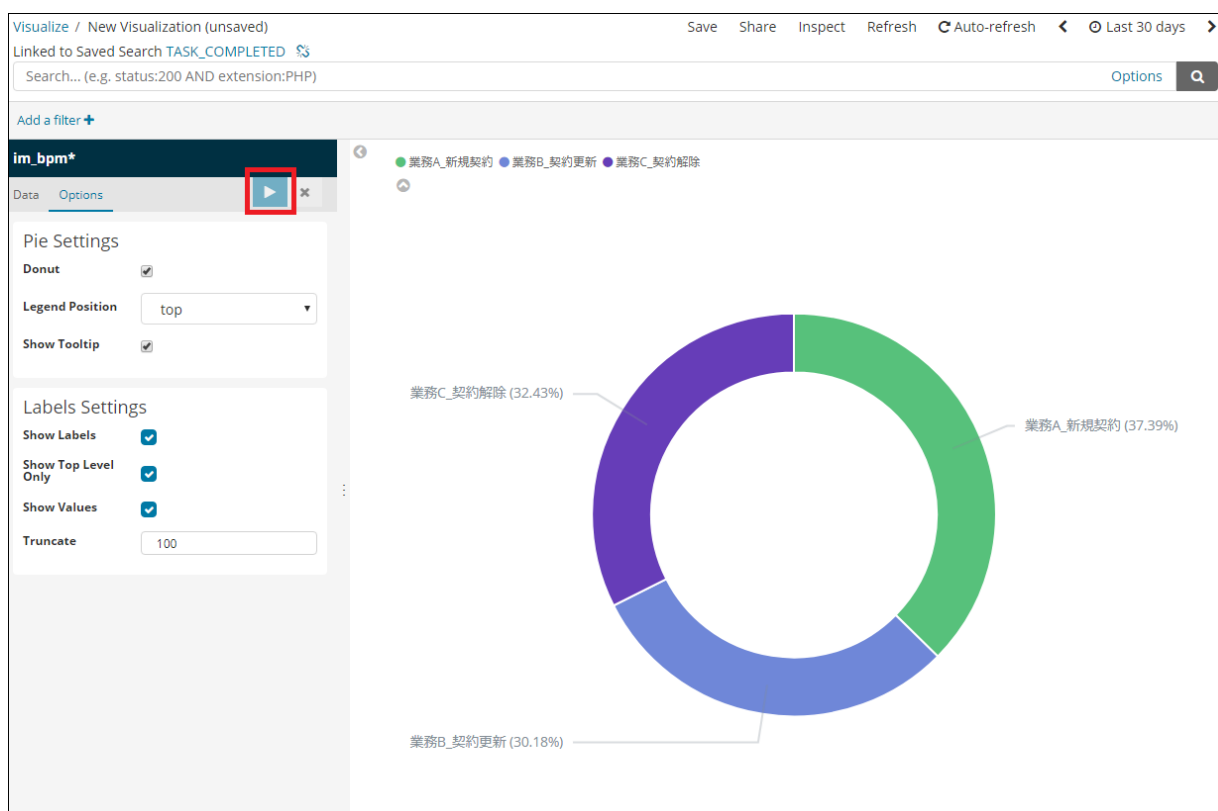
7. グラフの凡例（Legend）の表示位置をグラフ上部に指定します。

「Options」タブをクリックし、「Pie Settings」の「Legend Position」に「top」を指定し、「Labels Settings」の「Show Labels」にチェックを入れます。



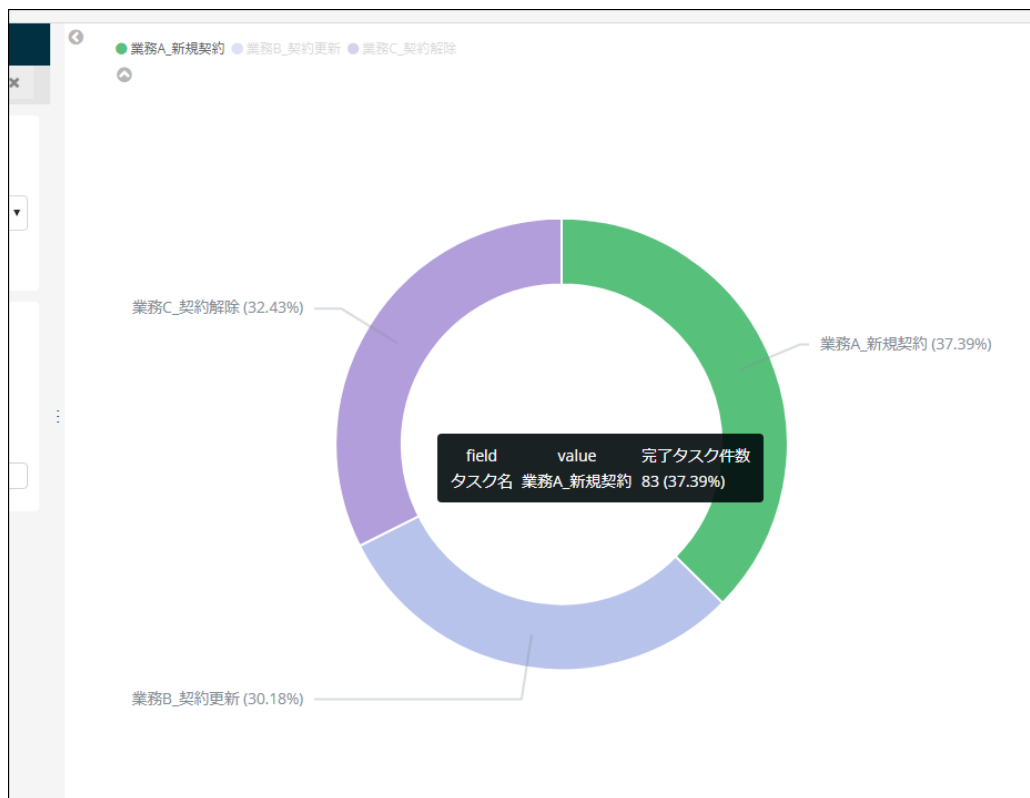
図：「Options」

8. 「Apply Changes」をクリックし、円グラフが表示されることを確認します。



図：「Apply Changes」

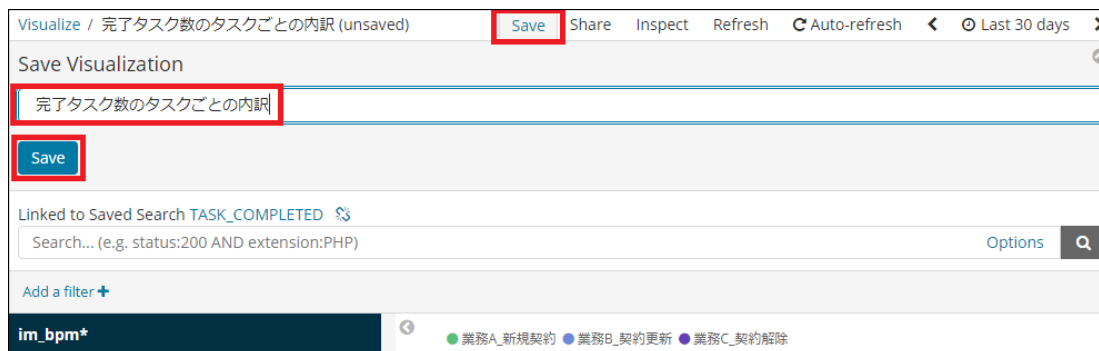
9. 円グラフにマウスカーソルを移動し、「field」「value」「タスク処理件数」がポップアップで表示されることを確認します。



図：円グラフの詳細情報のポップアップ

10. 「ヘッダメニュー」の「Save」をクリックし、「Save Visualization」欄に「完了タスク数のタスクごとの内訳」を設定し、「Save」をクリックして保存します。

項目名	設定値
Save Visualization	完了タスク数のタスクごとの内訳



図：「Save Visualization」

「完了タスク数のタスク処理ユーザーの内訳」を円グラフで可視化する

「[「完了タスク数のタスクごとの内訳」を円グラフで可視化する](#)」で作成した「完了タスク数のタスクごとの内訳」を修正し、タスクを処理したユーザーの内訳を円グラフを使用して可視化する方法を説明します。

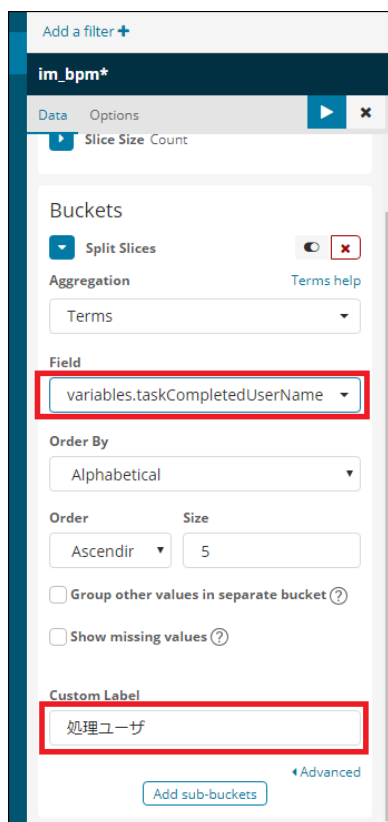
1. 「Kibana」のメインメニューより「Visualize」をクリックし「Visualize」画面を開き、「完了タスク数のタスクごとの内訳」をクリックします。



図：「Visualize」

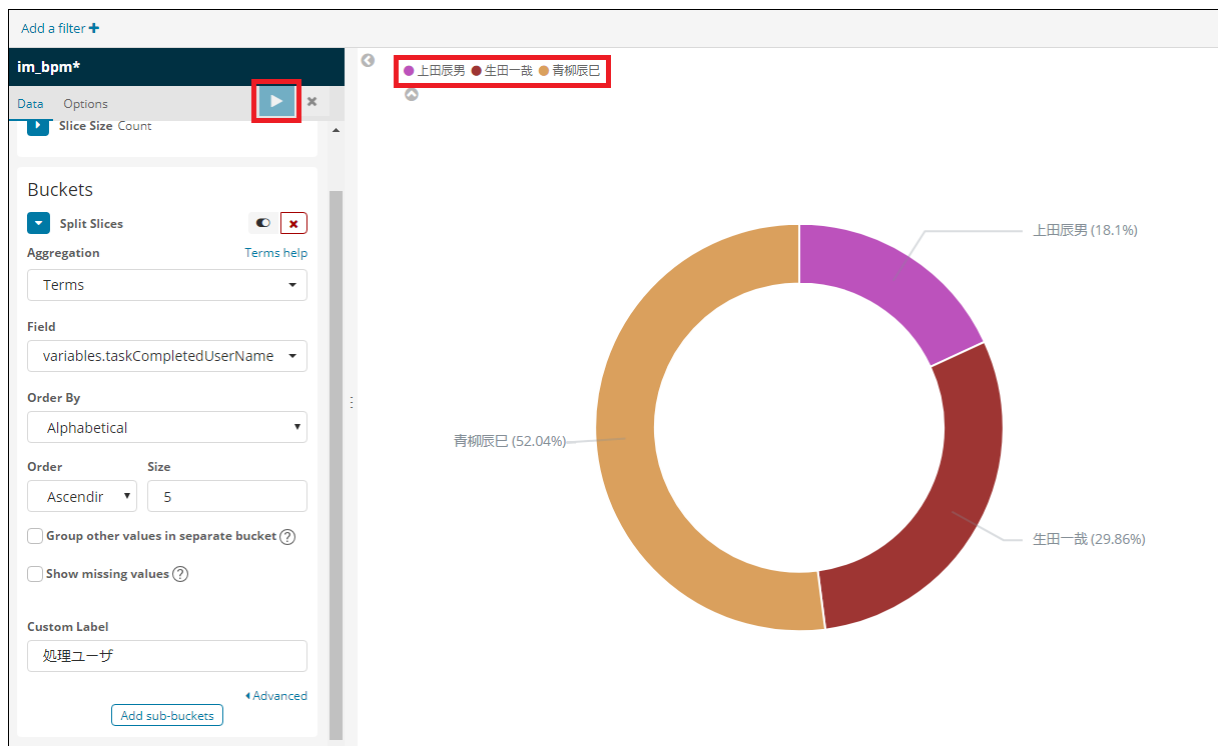
2. タスクの件数の分割単位を「処理ユーザ」ごとの分割に変更します。
「Buckets」→「Split Slices」をクリックし、各項目の値を下記の値に変更します。

項目名	設定値
Aggregation	Terms
Field	variables.taskCompletedUserName
Custom Label	処理ユーザ



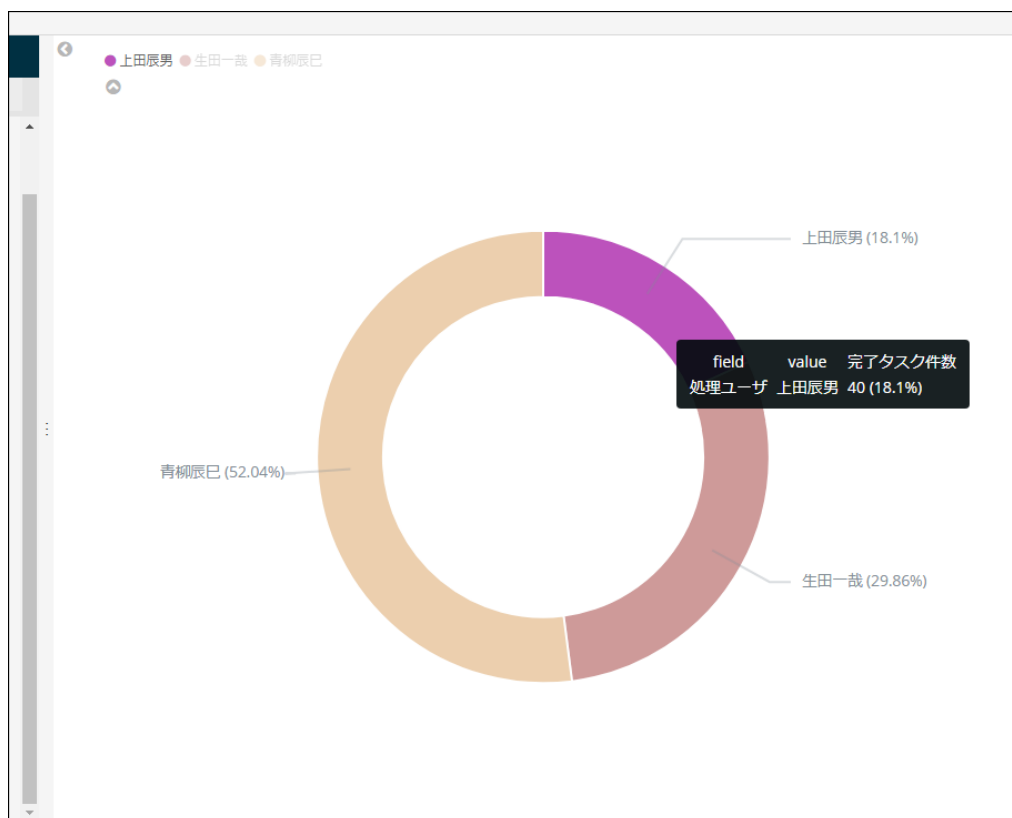
図：「Buckets」→「Split Slices」

3. 「Apply Changes」をクリックし、円グラフが処理ユーザごとの分割に変更されることを確認します。



図：「Apply Changes」

4. 円グラフにマウスカーソルを移動し、「field」「value」「完了タスク件数」がポップアップで表示されることを確認します。



図：円グラフの詳細情報のポップアップ

5. 「ヘッダメニュー」の「Save」をクリックし、「Save Visualization」欄を「完了タスク数のタスク処理ユーザの内訳」に変更し、「Save as a new visualization」にチェックを付け、「Save」をクリックして保存します。

項目名	設定値
Save Visualization	完了タスク数のタスク処理ユーザの内訳
Save as a new visualization	チェック



図：「Save Visualization」

「完了タスク数のタスクとユーザの内訳」を2階層の円グラフで可視化する

「「完了タスク数のタスクごとの内訳」を円グラフで可視化する」で作成した「完了タスク数のタスクごとの内訳」を修正し、完了タスク数のタスクごとの内訳とタスクを処理したユーザの内訳を、2階層の円グラフを使用して可視化する方法を説明します。

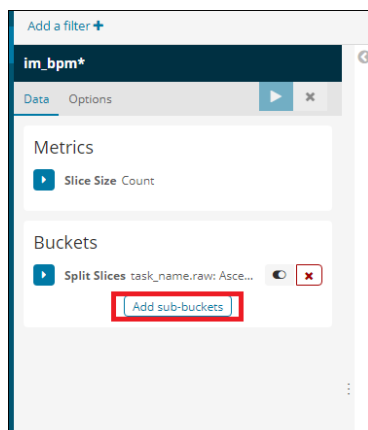
1. 「Kibana」のメインメニューより「Visualize」をクリックし「Visualize」画面を開き、「完了タスク数のタスクごとの内訳」をクリックします。



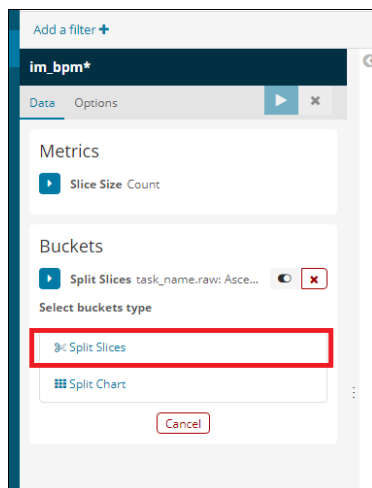
図：「Visualize」

2. タスクごとに分割された円グラフをさらに「処理ユーザ」ごとに分割します。
「Buckets」→「Split Slices」→「Add sub-buckets」から「Split Slices」をクリックし、下記の値を設定します。

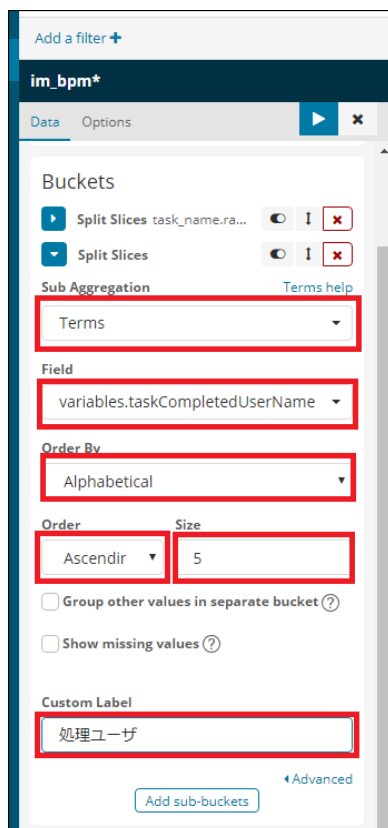
項目名	設定値
Aggregation	Terms
Field	variables.taskCompletedUserName
Order By	Alphabetical
Order	Ascending
Size	5
Custom Label	処理ユーザ



図：「Buckets」→「Add sub-buckets」

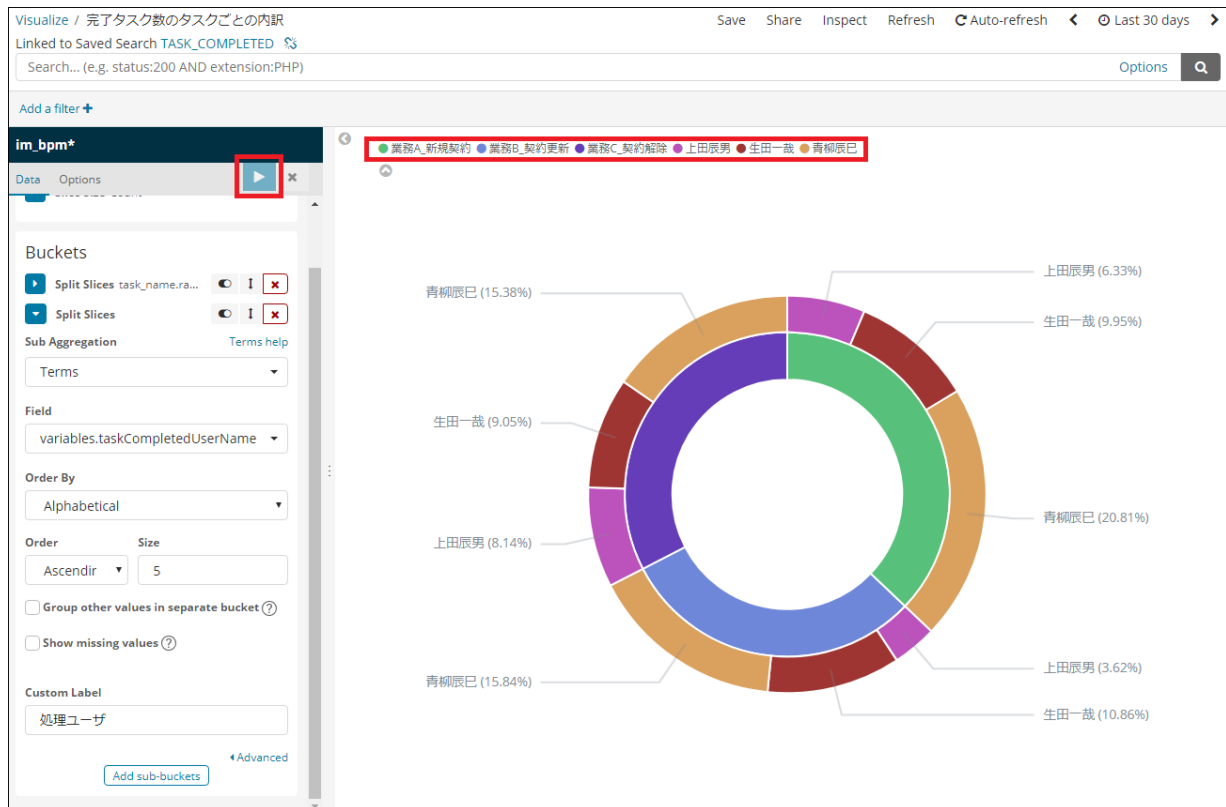


図：「Split Slices」



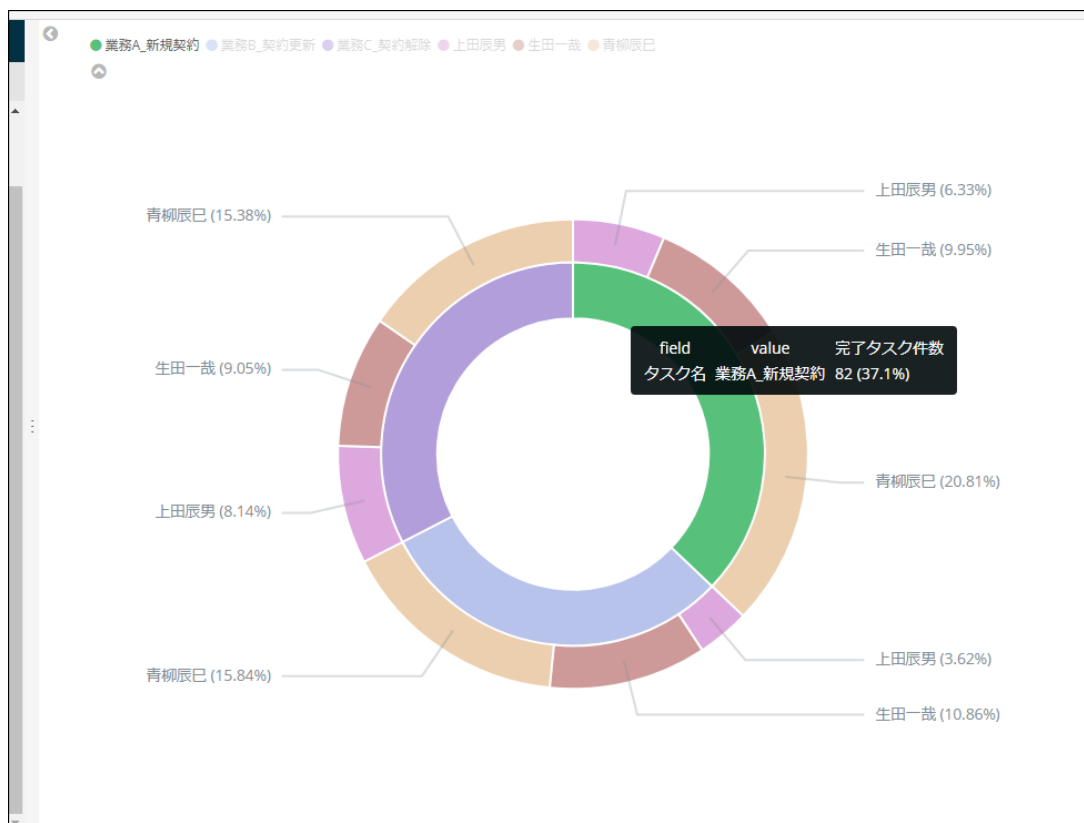
図：「Split Slices」の入力

3. 「Apply Changes」をクリックし、各タスクごとに分割された円グラフがさらに処理ユーザごとに分割されることを確認します。



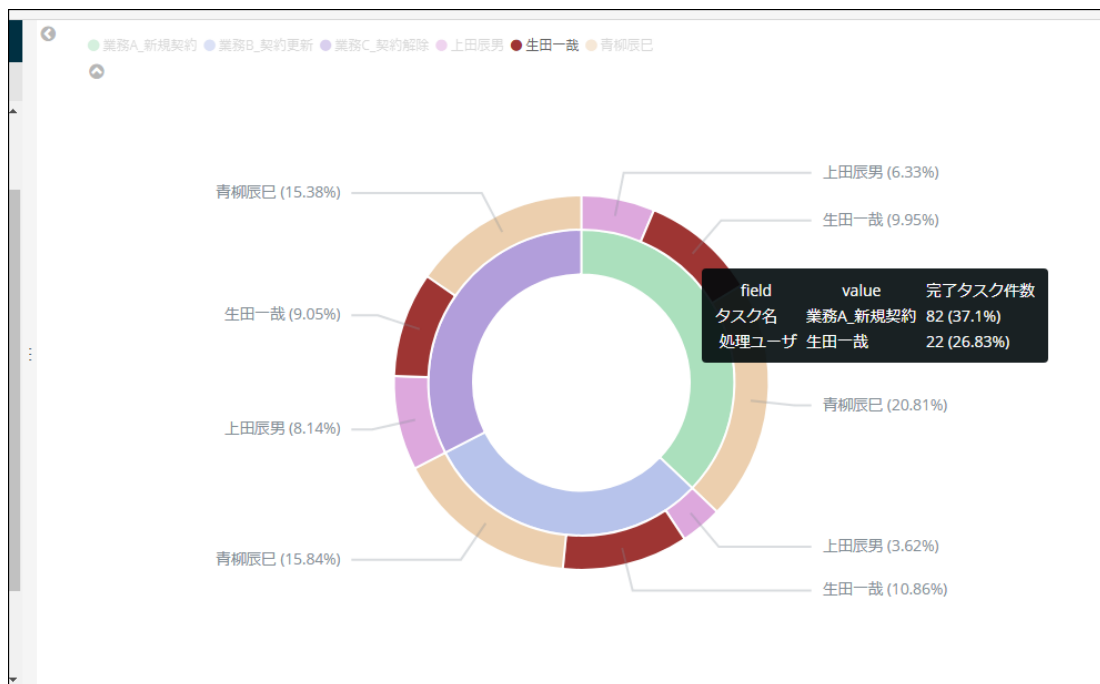
図：「Apply Changes」

4. 内側の円グラフにマウスカーソルを移動し、「field」「value」「完了タスク件数」がポップアップで表示されることを確認します。



図：内側の円グラフの詳細情報のポップアップ

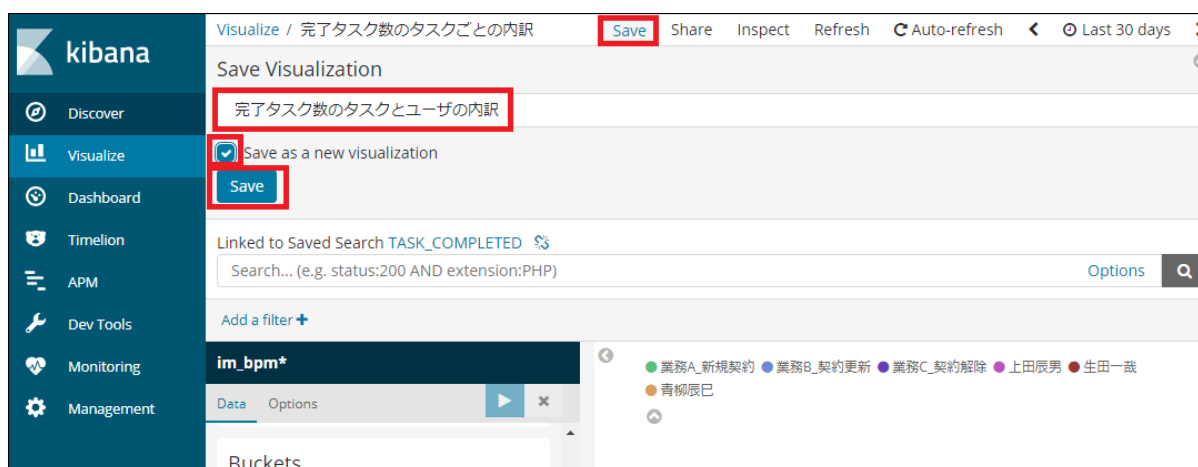
5. 外側の円グラフにマウスカーソルを移動し、「field」「value」「所要時間の内訳」がポップアップで表示されることを確認します。（「field」の詳細項目として「処理ユーザ」が表示されます。）



図：外側の円グラフの詳細情報のポップアップ

6. 「ヘッダメニュー」の「Save」をクリックし、「Save Visualization」欄を「完了タスク数のタスクとユーザの内訳」に変更し、「Save as a new visualization」にチェックを付け、「Save」をクリックして保存します。

項目名	設定値
Save Visualization	完了タスク数のタスクとユーザの内訳
Save as a new visualization	チェック

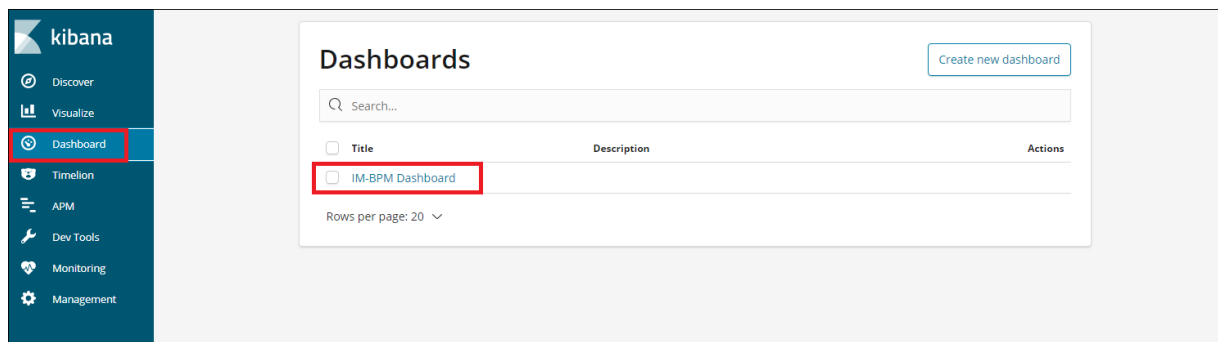


図：「Save Visualization」

「完了タスク数のタスクごとの内訳」、「完了タスク数のタスク処理ユーザの内訳」、「完了タスク数のタスクとユーザの内訳」をダッシュボードに追加する

Kibana上に上記で作成した「完了タスク数のタスクごとの内訳」、「完了タスク数のタスク処理ユーザの内訳」、「完了タスク数のタスクとユーザの内訳」を「IM-BPM Dashboard」ダッシュボードに追加します。

1. Kibanaのメインメニューより「Dashboard」をクリックし、「Dashboards」画面の「IM-BPM Dashboard」をクリックします。



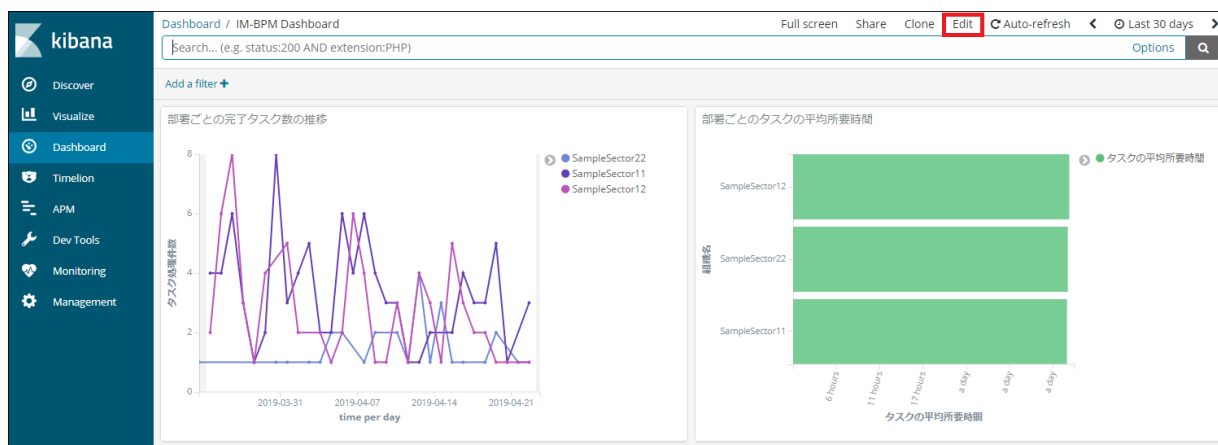
図：「Dashboards」画面

コラム

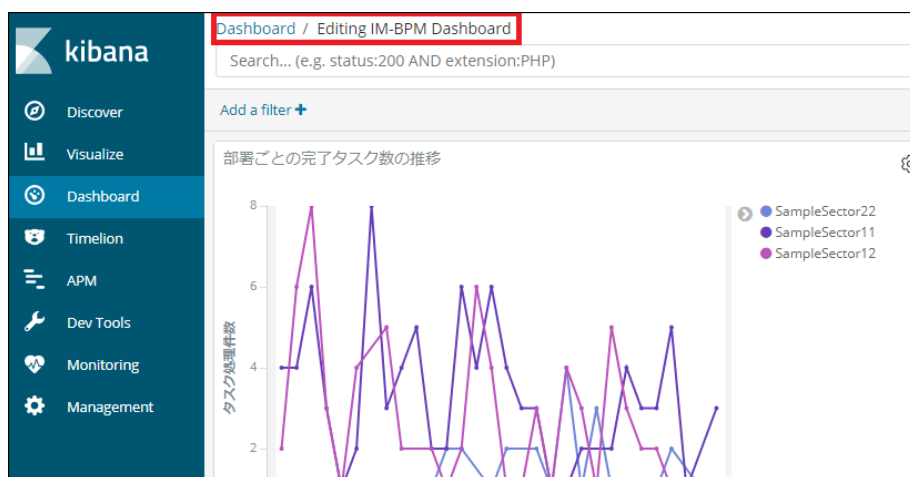
下記のような画面が表示され、データが表示されていない場合、「Time Range」（画面右上の「Last 15 minutes」と表示されている項目）より、データの表示期間を「Last 30 days」へ変更してください。

図：データが表示されていない場合

2. 「Dashboard / IM-BPM Dashboard」画面より、ヘッダメニューの「Edit」をクリックし、「Dashboard / Editing IM-BPM Dashboard」画面を開きます。

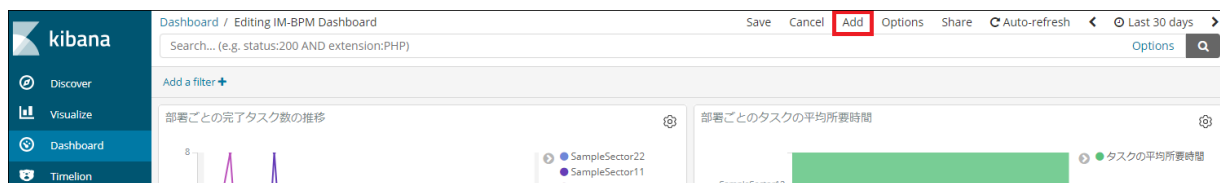


図：「IM-BPM Dashboard」



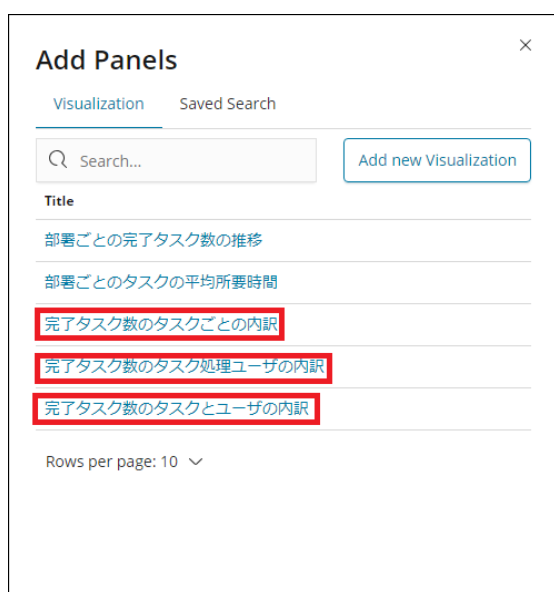
図：「Dashboard / Editing IM-BPM Dashboard」画面

3. ヘッダメニューの「Add」をクリックし、「Add Panels」を開きます。



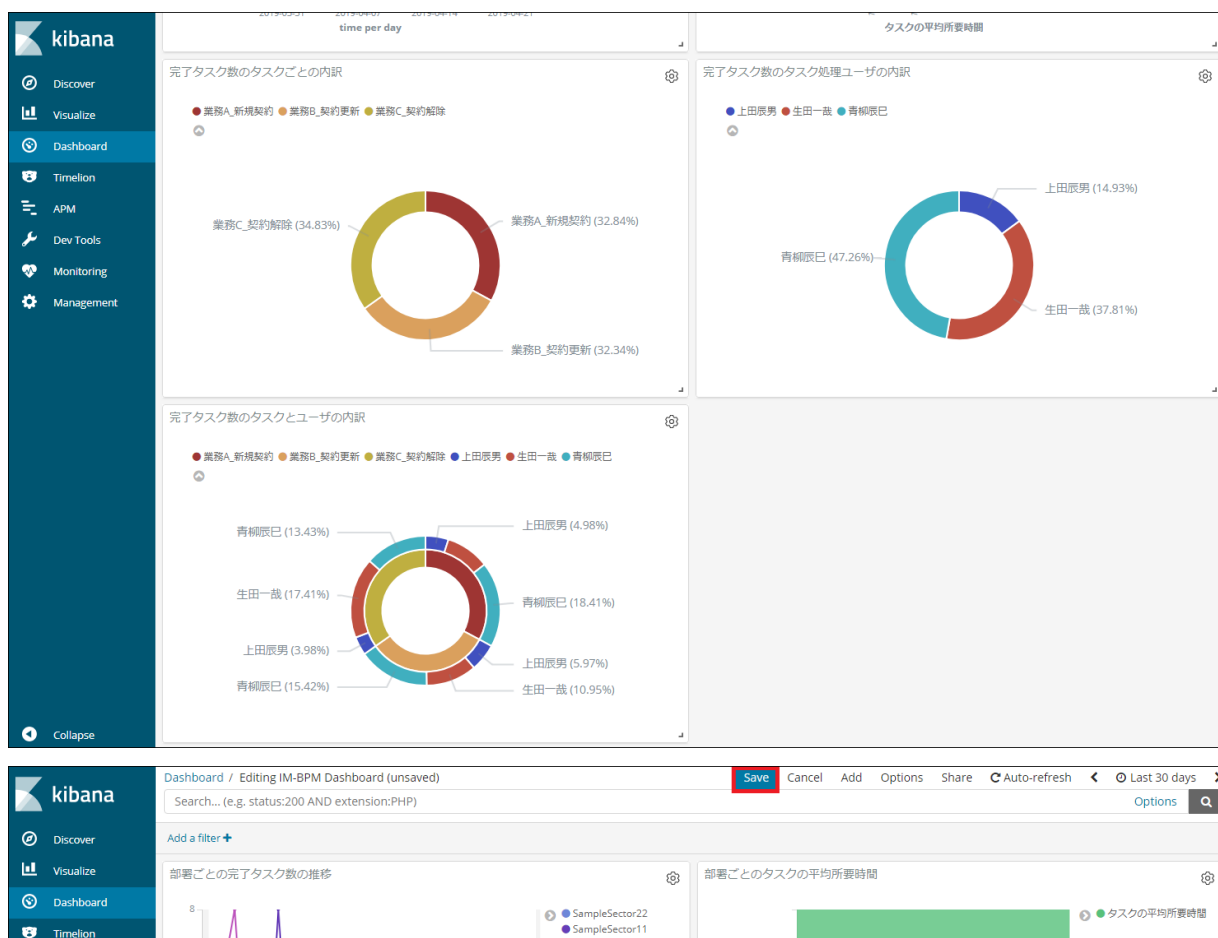
図：「Editing New Dashboard」→「Add」

4. 「Add Panels」→「Visualization」タブより「完了タスク数のタスクごとの内訳」、「完了タスク数のタスク処理ユーザの内訳」、「完了タスク数のタスクとユーザの内訳」をクリックします。



図：「Add Panels」→「Visualization」

5. ダッシュボードに「完了タスク数のタスクごとの内訳」、「完了タスク数のタスク処理ユーザの内訳」、「完了タスク数のタスクとユーザの内訳」が追加されたことを確認し、ヘッダメニューの「Save」をクリックします。



図：「Save」

6. 「Confirm Save」をクリックします。

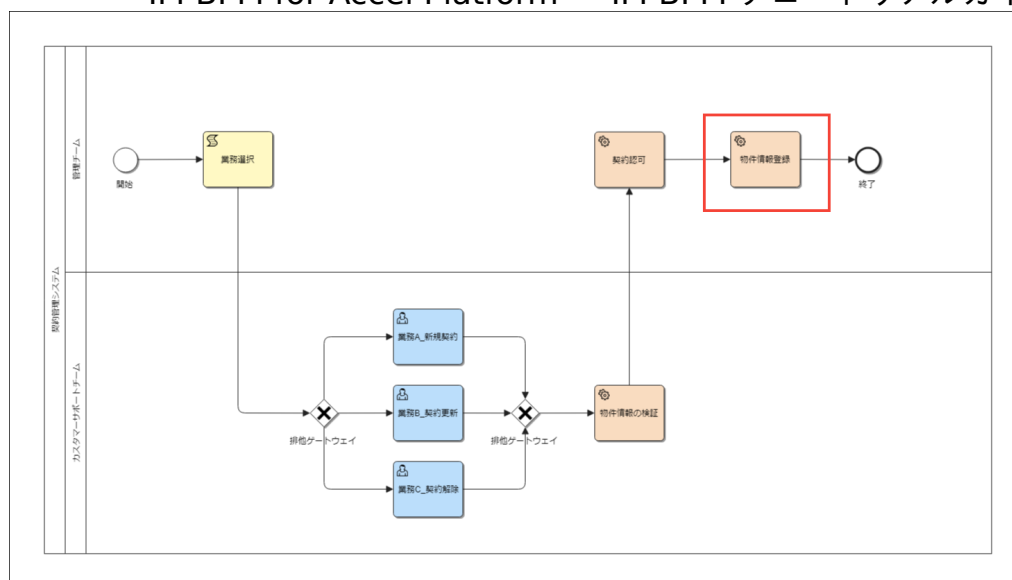
The 'Save Dashboard' dialog box is shown. It has a title 'Save Dashboard' and a close button. Below the title, there are two radio buttons: 'Save as a new dashboard' (selected) and 'Save as an existing dashboard'. The 'Title' field contains 'IM-BPM Dashboard'. The 'Description' field is empty. Below the description, there are two radio buttons: 'Store time with dashboard' (selected) and 'Do not store time with dashboard'. Below these radio buttons, there is a text box that says 'This changes the time filter to the currently selected time each time this dashboard is loaded.' At the bottom, there are two buttons: 'Cancel' and 'Confirm Save' (highlighted with a red border).

図：「Save Dashboard」

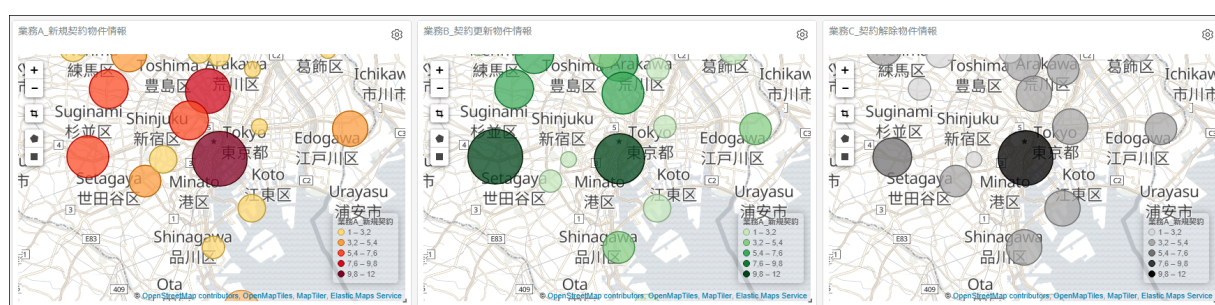
タスクの処理件数を地図上で可視化する

このチュートリアルでは各タスクの処理件数を地図上で可視化し、ダッシュボードに追加する方法について説明します。

「IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する」の「統計情報取得の対象となるプロセスのデプロイを行う」でデプロイした、「統計情報の取得対象となるプロセス (target_process)」の「物件情報登録」業務で登録された、契約物件の緯度・経度情報を「Coordinate Map」(地図上に表示)を使用して可視化します。



図：「物件情報登録」業務



図：「業務A_新規契約物件情報」、「業務B_契約更新物件情報」、「業務C_契約解除物件情報」

**注意**

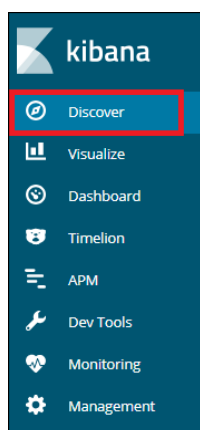
本チュートリアルは、事前に「[IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する](#)」の実施が完了していることを前提とします。

- 「Discover」でデータの検出を保存する
- 情報を地図上で可視化する
- 既存の可視化情報を修正して新しい可視化情報を作成する
- 可視化された情報をダッシュボードに追加する

「Discover」でデータの検出を保存する

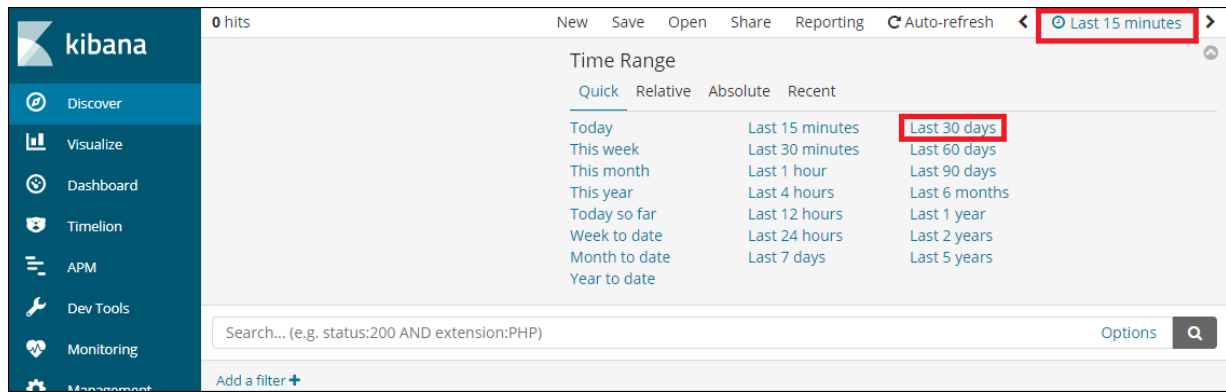
あらかじめフィルタ条件が設定されたデータセットを作成します。

1. 「Kibana」のメインメニューより「Discover」をクリックし、「Discover」画面を開きます。



図：「Discover」

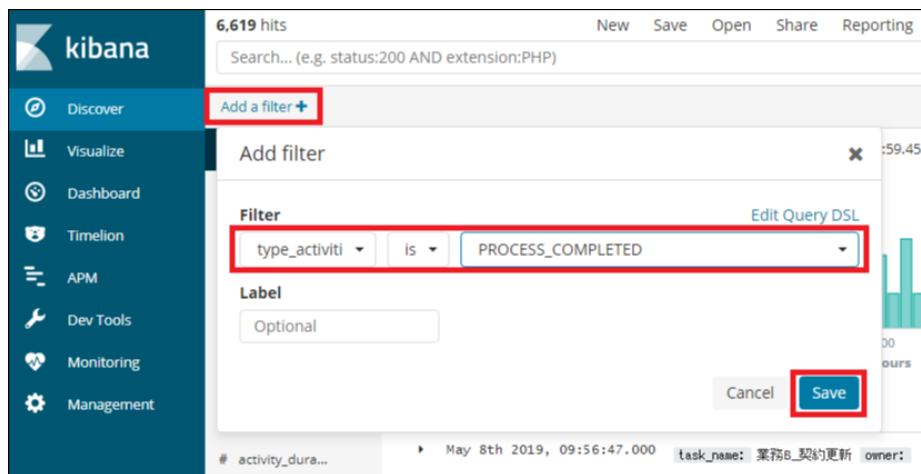
2. 集計対象の期間を指定します。
「Time Range」欄より「Last 30 days」を選択します。



図：「Time Range」の選択

3. 集計対象のイベントとして「PROCESS_COMPLETED」（プロセス完了イベント）を指定します。
「Add a filter」をクリックし、下記の値を設定し、「Save」をクリックします。

項目名	設定値
Fields...	type_activiti
Operators...	is
Values...	PROCESS_COMPLETED



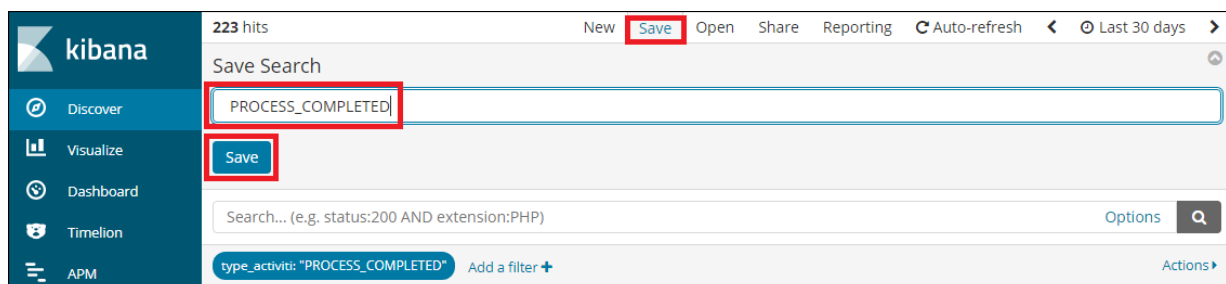
図：「Add a filter」

**注意**

フィールド「type_activiti」のほか、項目名が類似するフィールド「activity_type」が存在します。ここでは「type_activiti」を指定してください。

4. データセットに名前を付け保存します。ここではフィルタ条件とした「PROCESS_COMPLETED」をそのままデータセットの名前として指定します。
「ヘッダメニュー」の「Save」をクリックし、「Save Search」欄に「PROCESS_COMPLETED」を設定し、「Save」をクリックして保存します。

項目名	設定値
Save Search	PROCESS_COMPLETED



図：「Save Search」

情報を地図上で可視化する

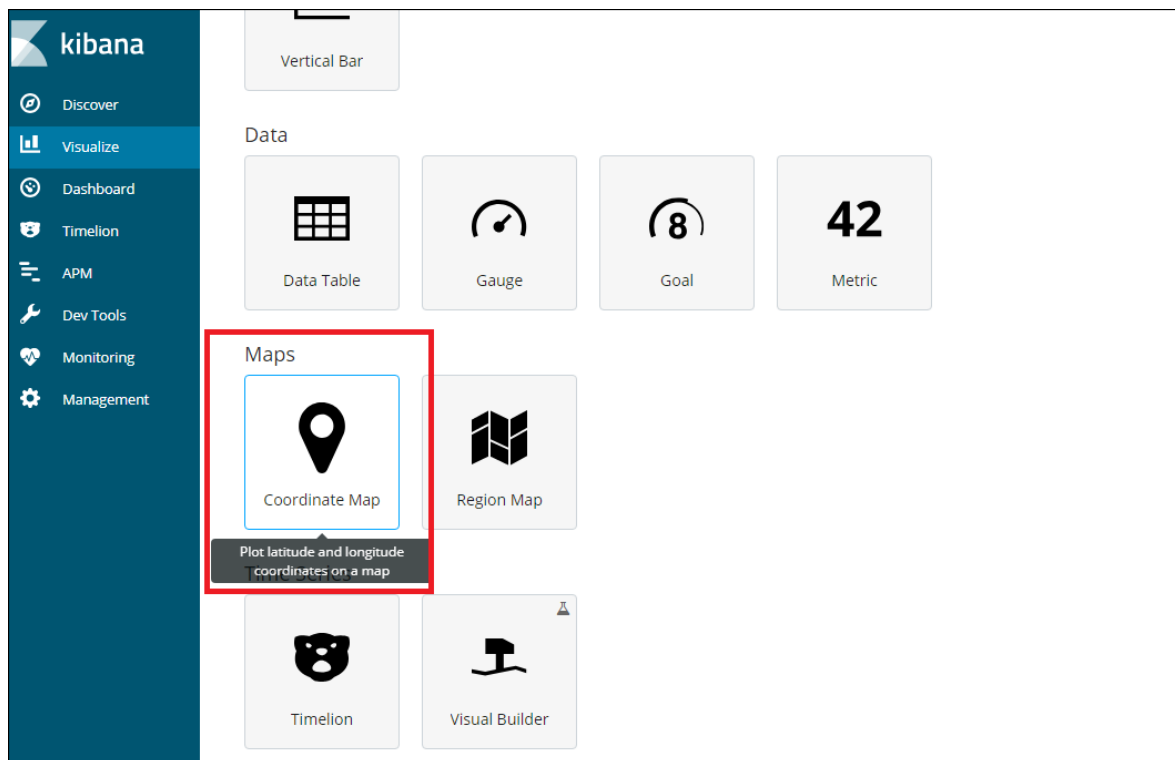
最初に「業務A_新規契約」業務で扱われた物件情報に絞り、可視化を行います。

1. 「Kibana」のメインメニューより「Visualize」をクリックし「Visualize」画面を開き、「+」をクリックします。



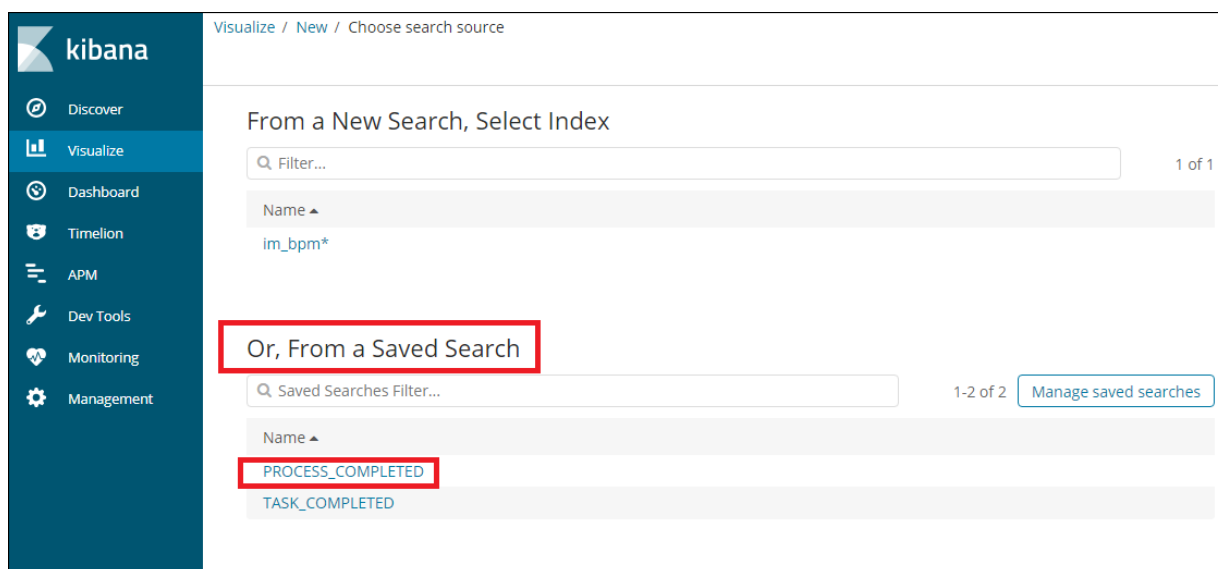
図：「Visualize」

2. グラフの種類を指定します。
「Select visualization type」画面にて「Coordinate Map」を選択します。



図：「Coordinate Map」

3. 可視化の対象を選択します。
上記で作成したフィルタ条件を設定済みのデータセット「PROCESS_COMPLETED」を可視化の対象として指定します。
「Or, From a Saved Search」より「PROCESS_COMPLETED」を選択します。

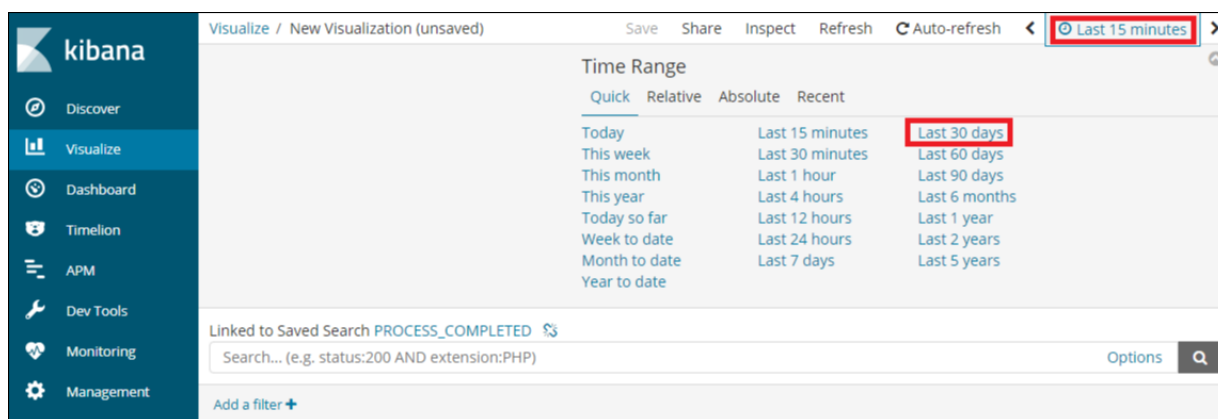


図：「Or, From a Saved Search」

4. 集計対象の期間を指定します。

Kibanaでは、データセットのフィルタ条件に期間が設定されている場合でも、現在のセッション中の「Time Range」の設定が優先される場合があるため、再度、期間を指定します。

「Time Range」欄より「Last 30 days」を選択します。

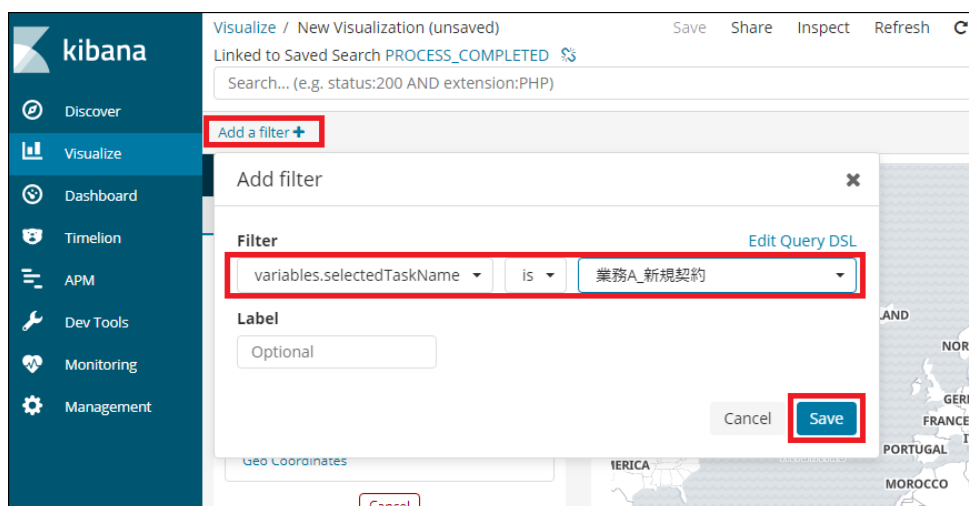


図：「Time Range」の選択

5. 「業務A_新規契約」業務を行ったプロセスのみを集計対象として指定します。

「Add a filter」をクリックし、下記の値を設定し、「Save」をクリックします。

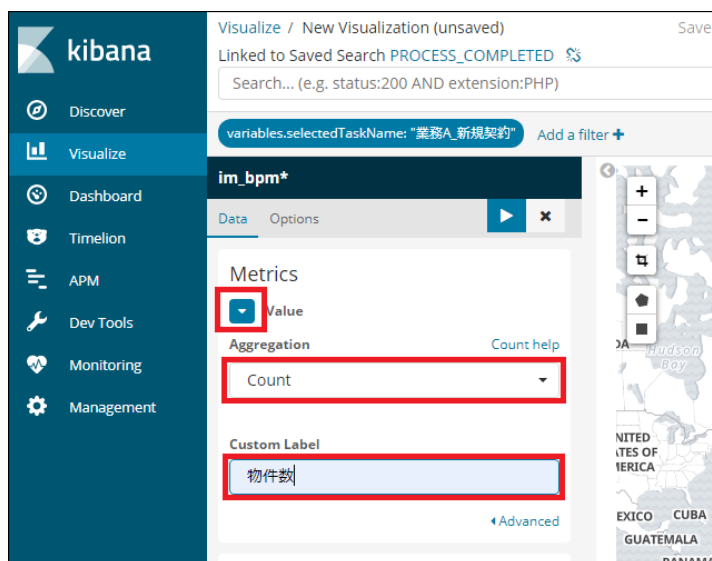
項目名	設定値
Fields...	variables.selectedTaskName
Operators...	is
Values...	業務A_新規契約



図：「Add a filter」

6. 完了したタスクの件数を集計対象に指定します。
「Data」タブ→「Metrics」の「Value」をクリックし、下記の値を指定します。

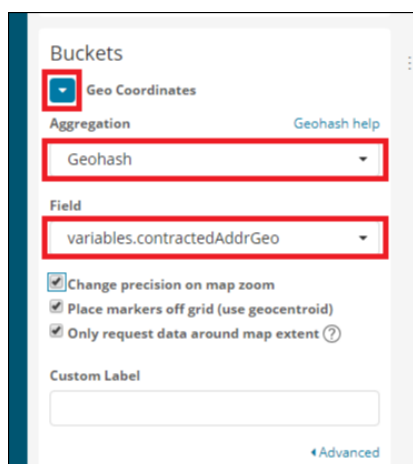
項目名	設定値
Aggregation	Count
Custom Label	物件数



図：「Data」→「Metrics」

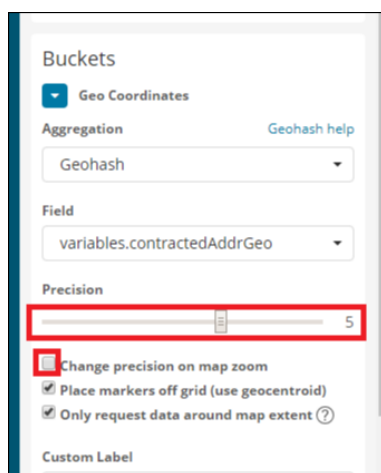
7. 緯度・経度情報を持つフィールド `variables.contractedAddrGeo` を可視化の対象として設定します。
「Data」タブ→「Buckets」→「Geo Coordinates」をクリックし、下記の値を指定してください。

項目名	設定値
Aggregation	Geohash
Custom Label	<code>variables.contractedAddrGeo</code>



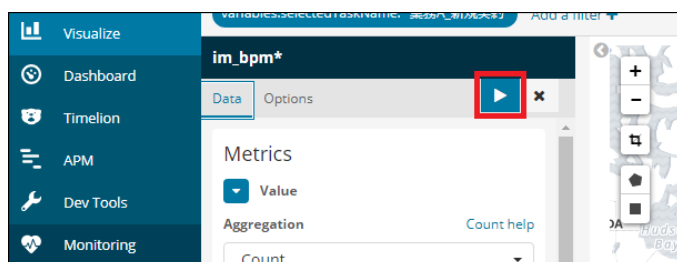
図：「Buckets」→「Geo Coordinates」

8. マップの拡大・縮小に追従し、precision（精度）の変更を行うかどうかの設定をします。
ここでは、固定の精度（マップの拡大・縮小に追従せず、精度一定）を選択します。
「Change precision on map zoom」のチェックを外し、スライダーでprecisionに「5」を設定します。



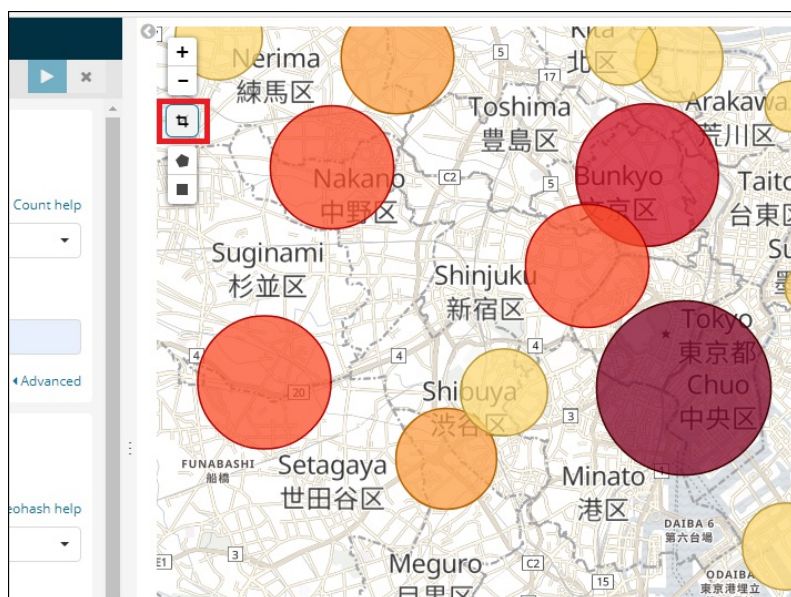
図：「Change precision on map zoom」

9. 「Apply changes」をクリックし設定を反映します。



図：「Apply changes」

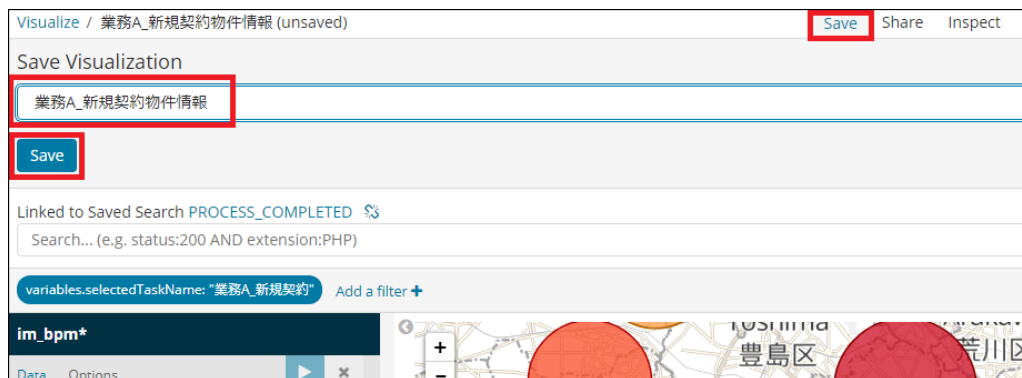
10. 「Fit Data Bounds」をクリックし地図の縮尺を調整し、地図上に物件情報が可視化されることを確認します。



図：「Fit Data Bounds」

11. 「ヘッダメニュー」の「Save」をクリックし、「Save Visualization」欄に「業務A_新規契約物件情報」を設定し、「Save」をクリックして保存します。

項目名	設定値
Save Visualization	業務A_新規契約物件情報



図：「Save Visualization」

既存の可視化情報を修正して新しい可視化情報を作成する

ここでは、「[情報を地図上で可視化する](#)」で作成した「業務A_新規契約物件情報」を修正し、「業務B_契約更新」業務で扱われた物件情報の可視化を行います。

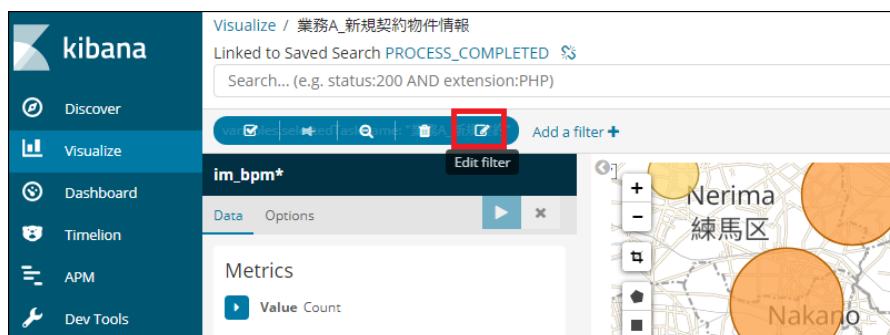
1. 「Kibana」のメインメニューより「Visualize」をクリックし、「Visualize」画面を開き、「業務A_新規契約物件情報」をクリックします。



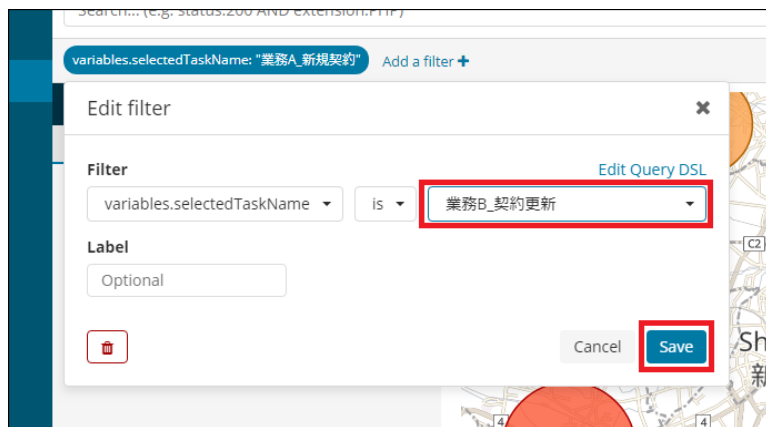
図：「Visualize」

2. 可視化対象の業務を「業務B_契約更新」業務へ変更します。
フィルタ「variables.selectedTaskName:業務A_新規契約」の「Edit Filter」をクリックし、フィルタ編集画面を開き、「Values...」の値を「業務B_契約更新」に変更し「Save」をクリックします。

項目名	設定値
Values...	業務B_契約更新

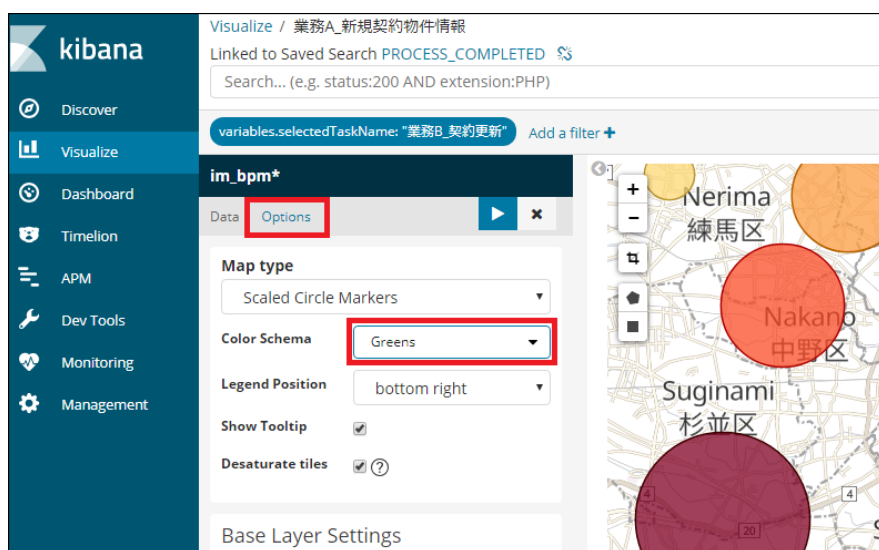


図：「Edit Filter」



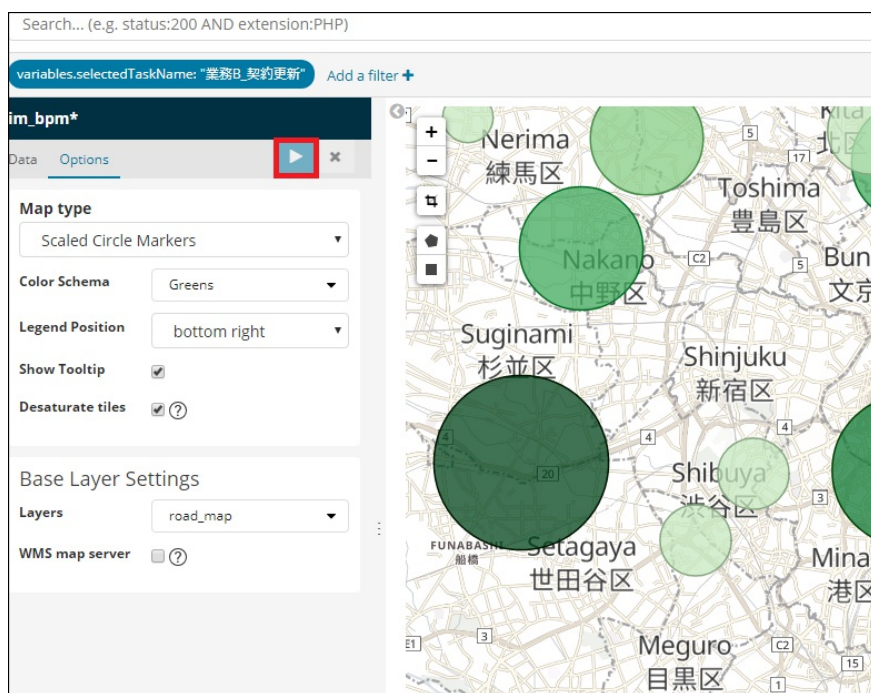
図：フィルタ「variables.selectedTaskName:「業務A_新規契約」」

3. 地図上で物件の位置を表す円マーカの色を変更します。
「Options」タブの「Color Schema」を「Greens」に変更します。



図：「Options」→「Color Schema」

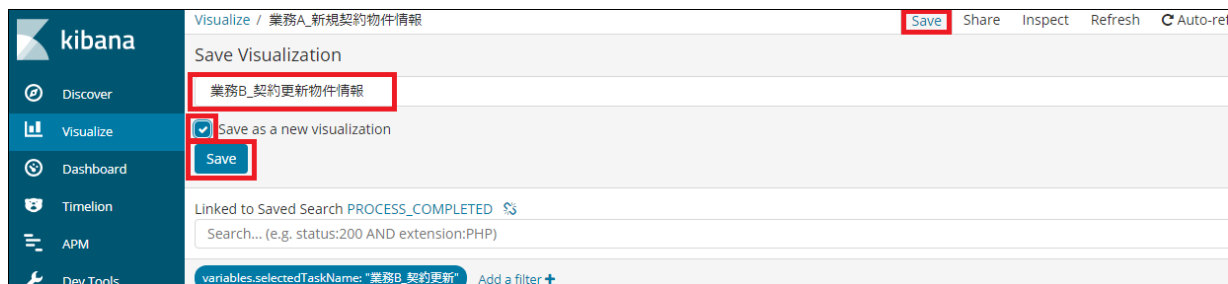
4. 「Apply Changes」をクリックし、修正内容が反映されることを確認します。



図：「Apply Changes」

5. 「ヘッダメニュー」の「Save」をクリックし、「Save Visualization」欄を「業務B_契約更新物件情報」に変更し、「Save as a new visualization」にチェックを付け「Save」をクリックして保存します。

項目名	設定値
Save Visualization	業務B_契約更新物件情報
Save as a new visualization	チェック



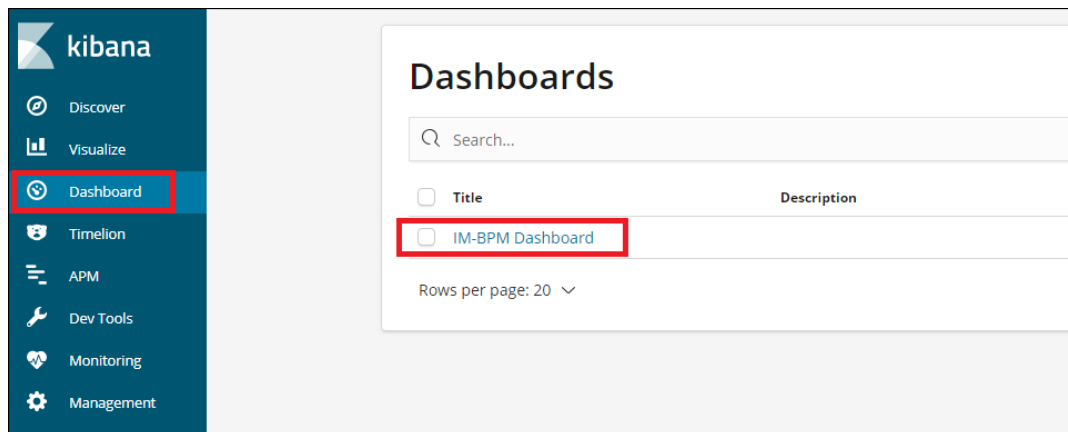
図：「Save Visualization」

6. 上記と同様の手順を「業務C_契約解除」タスクに対しても行い、「業務C_更新解除物件情報」として保存してください。

可視化された情報をダッシュボードに追加する

Kibana上に上記で作成した「業務A_新規契約物件情報」、「業務B_契約更新物件情報」、「業務C_契約解除物件情報」を「IM-BPM Dashboard」ダッシュボードに追加します。

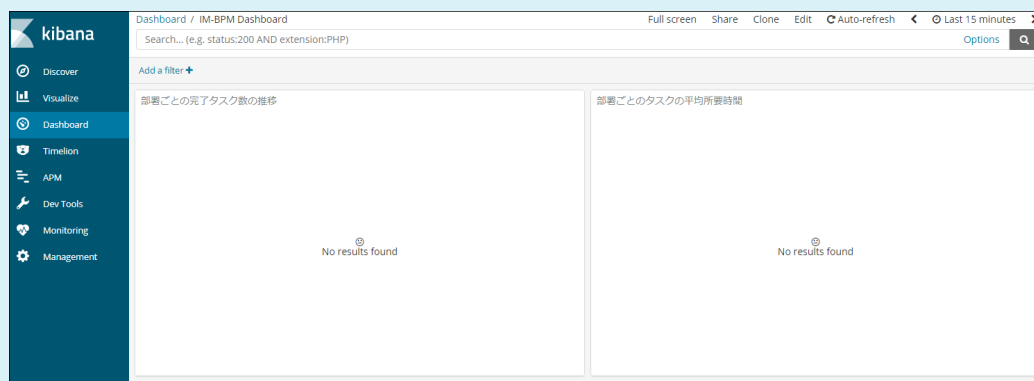
1. Kibanaのメインメニューより「Dashboard」をクリックし、「Dashboards」画面の「IM-BPM Dashboard」をクリックします。



図：「Dashboards」画面

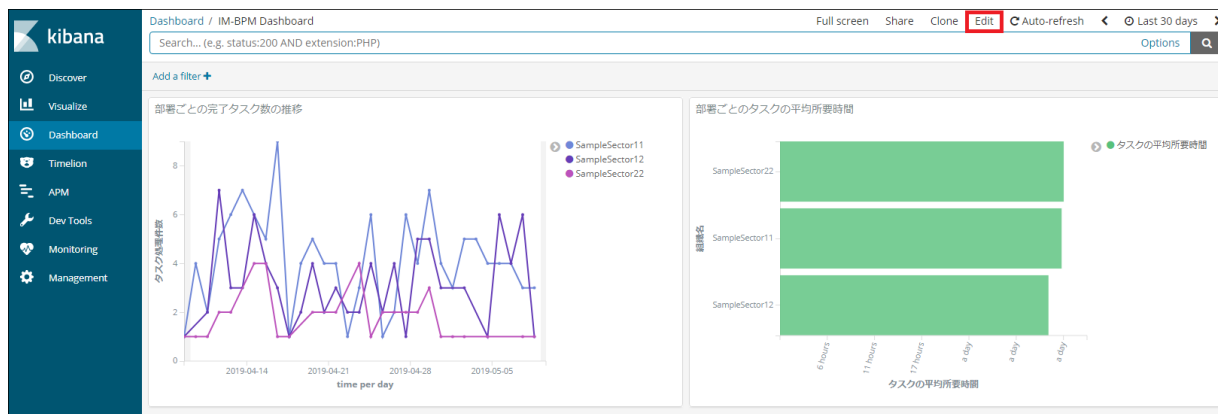
コラム

下記のような画面が表示され、データが表示されていない場合、「Time Range」（画面右上の「Last 15 minutes」と表示されている項目）より、データの表示期間を「Last 30 days」へ変更してください。

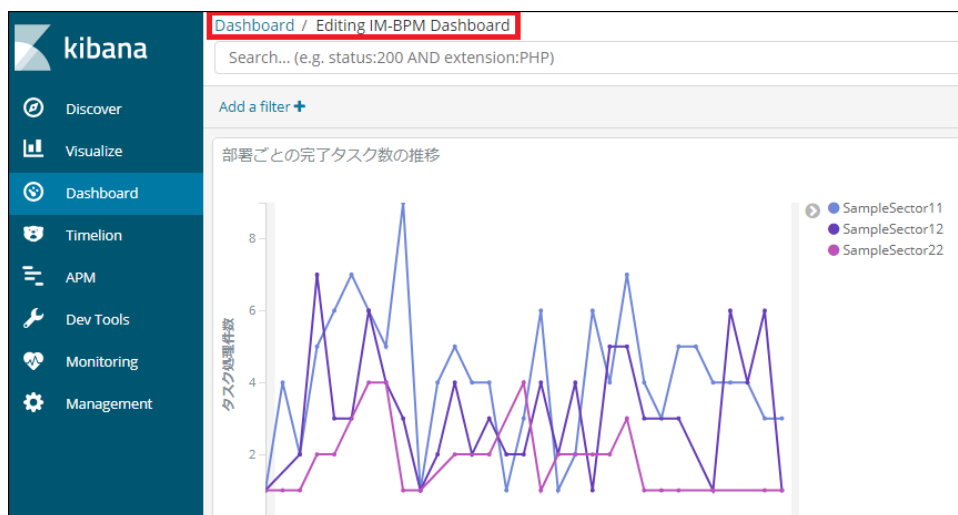


図：データが表示されていない場合

2. 「Dashboard / IM-BPM Dashboard」画面より、ヘッダメニューの「Edit」をクリックし、「Dashboard / Editing IM-BPM Dashboard」画面を開きます。

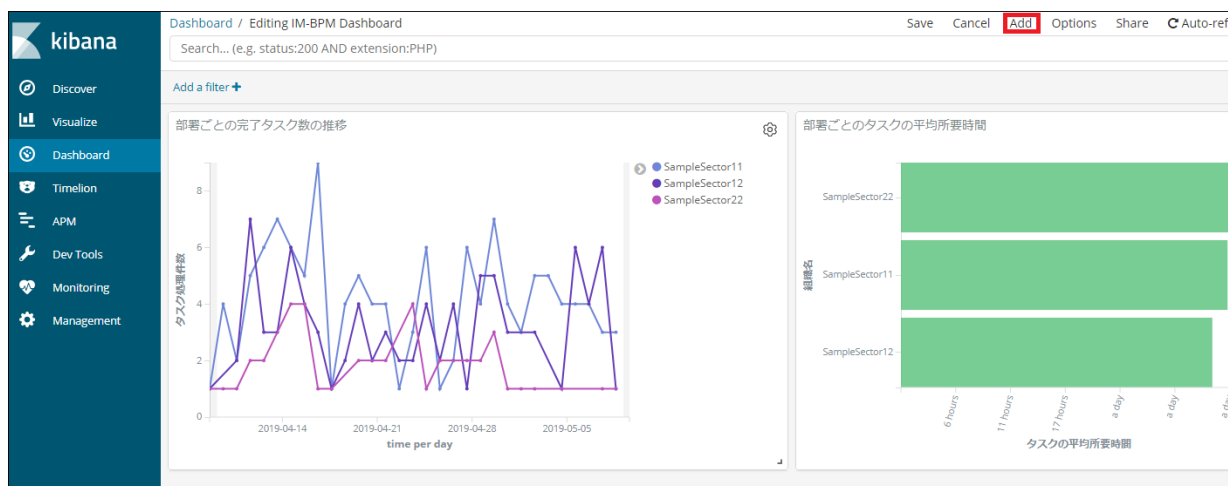


図：「IM-BPM Dashboard」



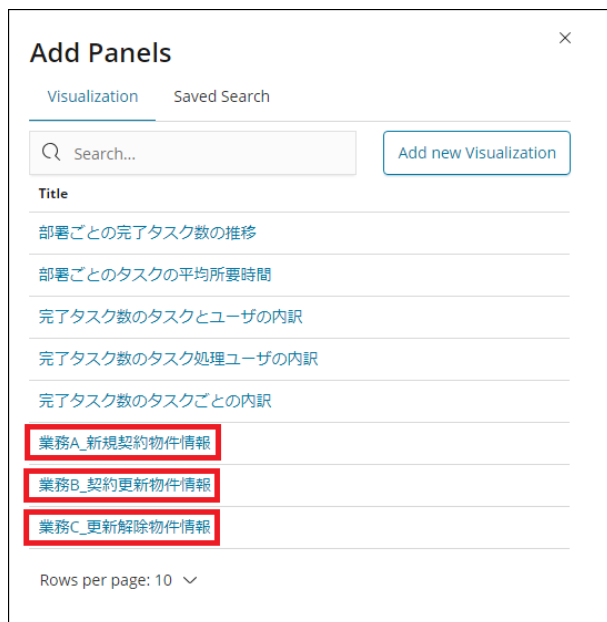
図：「Dashboard / Editing IM-BPM Dashboard」画面

3. ヘッドメニューの「Add」をクリックし、「Add Panels」を開きます。



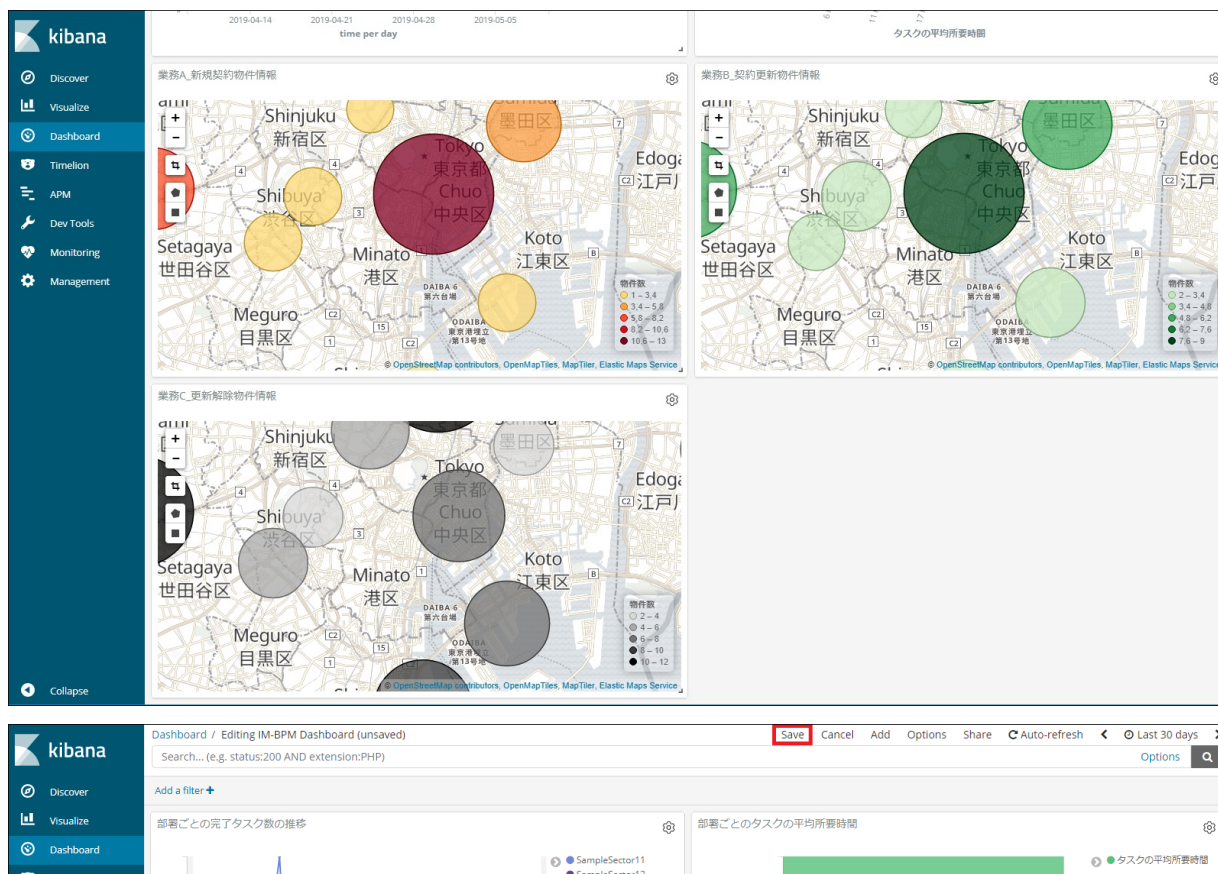
図：「Editing New Dashboard」→「Add」

4. 「Add Panels」→「Visualization」タブより「業務A_新規契約物件情報」、「業務B_契約更新物件情報」、「業務C_契約解除物件情報」をクリックします。



図：「Add Panels」→「Visualization」

5. ダッシュボードに「業務A_新規契約物件情報」、「業務B_契約更新物件情報」、「業務C_契約解除物件情報」が追加されたことを確認し、ヘッダメニューの「Save」をクリックします。



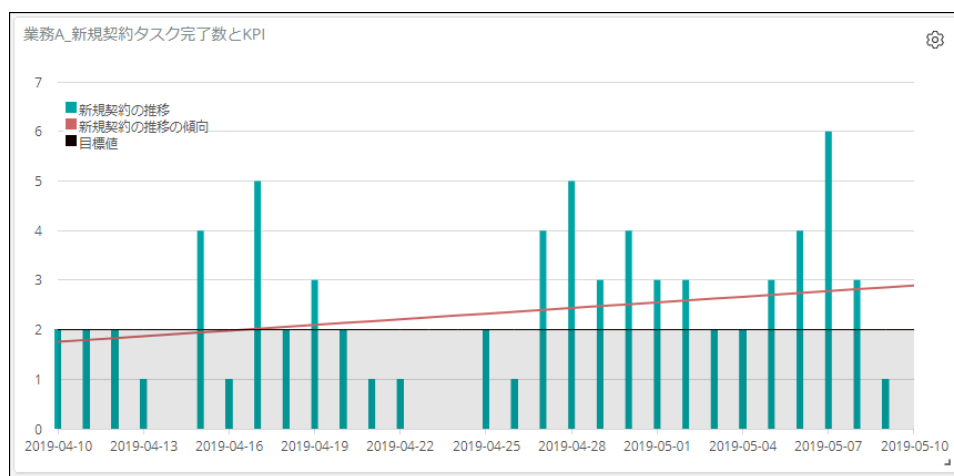
図：「Save」

6. 「Confirm Save」をクリックします。

図：「Save Dashboard」

グラフ上にKPIを表示する

このチュートリアルでは、Kibanaのグラフ上にKPIとして静的な基準線と、グラフの傾向（増加傾向／減少傾向）を表示し、KPIが付与されたグラフをダッシュボードに追加する方法を解説します。



図：「業務A_新規契約タスク完了数とKPI」



注意

本チュートリアルは、事前に「[IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する](#)」の実施が完了していることを前提とします。

- Kibanaの「[Timelion](#)」機能を使用してグラフ上にKPIを表示する
- 「[業務A_新規契約タスク完了数とKPI](#)」をダッシュボードに追加する

Kibanaの「Timelion」機能を使用してグラフ上にKPIを表示する

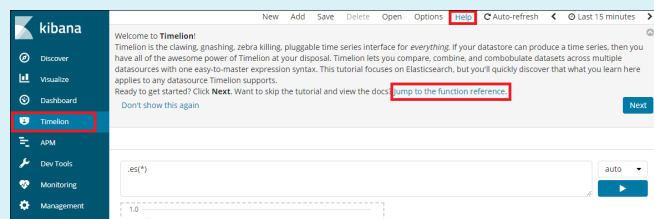
Kibanaの「Timelion」（タイムライン）機能を使用して、グラフ上にKPIとして静的な基準線と、グラフの傾向（増加傾向／減少傾向）を表示します。

コラム

「Timelion」画面と「Timelion Expression」について

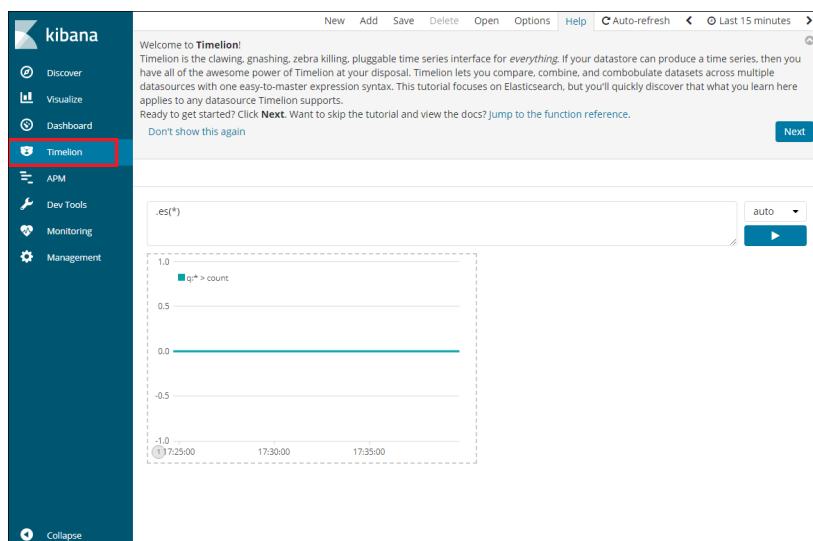
「Timelion」画面では、「Timelion Expression」という記法にもとづき、複雑な形式のグラフを表示したり、また、それらを組み合わせ、一つの図中に複数のグラフを重ねて表示するなど、様々な表現を行えます。

「Timelion Expression」で使用可能な関数についての詳細は「Help」の「Function reference」より確認できます。



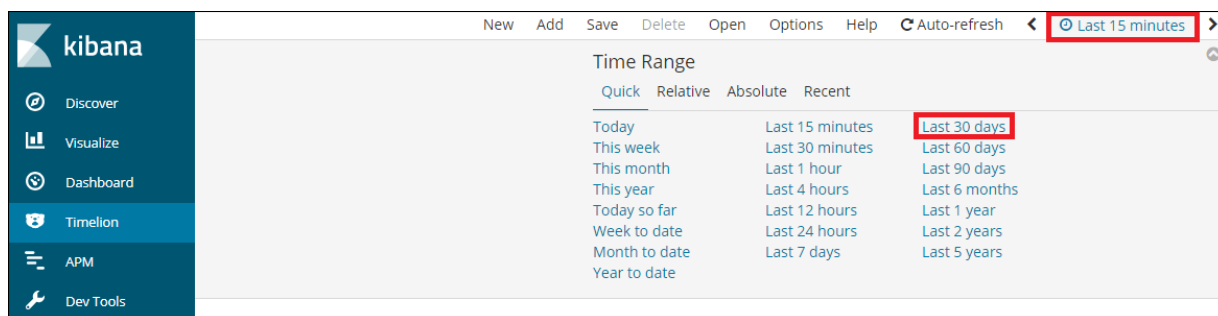
図：「Help」→「Function reference」

1. 「Kibana」のメインメニューより「Timelion」をクリックし、「Timelion」画面を開きます。



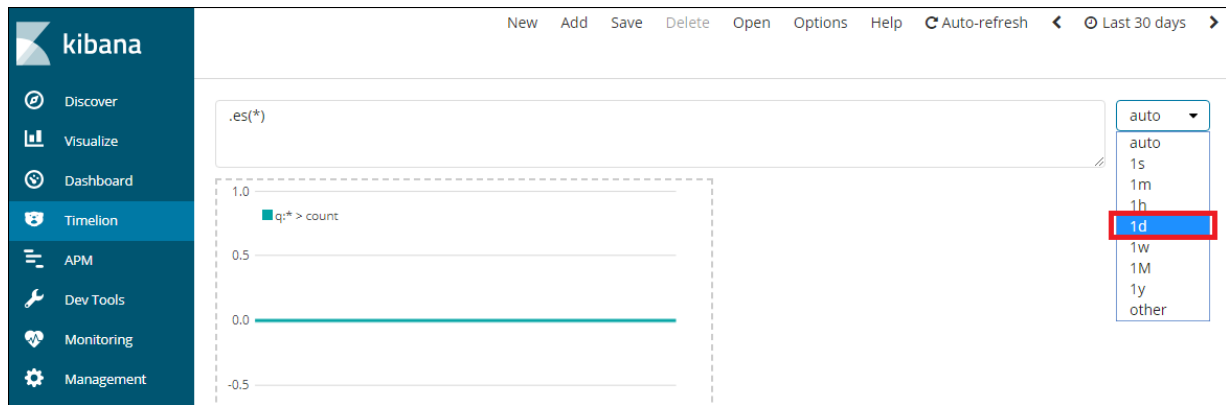
図：「Timelion」画面

2. 集計対象の期間を指定します。
「Time Range」欄より「Last 30 days」を選択します。



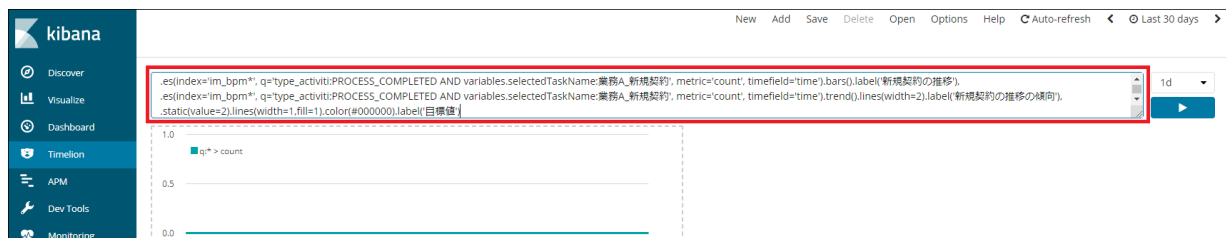
図：「Time Range」の選択

3. 集計単位を指定します。
画面右側のプルダウンリストより「1d」（1日毎）を選択します。



図：集計単位

4. グラフを表示するための「Timelion Expression」を指定します。
画面中央のテキストボックスに下記の文字列を設定します。



図：「Timelion Expression」

```
.es(index='im_bpm*', q='type_activiti:PROCESS_COMPLETED AND variables.selectedTaskName:業務A_新規契約', metric='count',
timefield='time').bars().label('新規契約の推移'),
.es(index='im_bpm*', q='type_activiti:PROCESS_COMPLETED AND variables.selectedTaskName:業務A_新規契約', metric='count',
timefield='time').trend().lines(width=2).label('新規契約の推移の傾向'),
.static(value=2).lines(width=1,fill=1).color(#000000).label('目標値')
```

各行で指定している関数および各パラメータの説明

- 1行目は、「業務A_新規契約」の完了件数の推移を表示するための記述です。

`.es()` 関数（Elasticsearchへ問い合わせを行うための関数）へ、インデックスパターンや、フィルタ条件、時系列フィールドなどの条件を引数として渡し、メソッドチェーン形式でグラフの見た目や、凡例上の表示などを指定します。

```
.es(index='im_bpm*', q='type_activiti:PROCESS_COMPLETED AND variables.selectedTaskName:業務A_新規契約', metric='count',
timefield='time').bars().label('新規契約の推移'),
```

コラム

ここでは、インデックスパターン「im_bpm*」に含まれるドキュメントのうち、「type_activiti」フィールドの値が `PROCESS_COMPLETED` かつ、「variables.selectedTaskName」フィールドの値が `業務A_新規契約` のドキュメントを対象とします。

- `.es(*)`関数のパラメータ

パラメータ	説明
<code>index='im_bpm*'</code>	可視化対象のインデックスパターンを指定します。
<code>q='type_activiti:PROCESS_COMPLETED AND variables.selectedTaskName:業務A_新規契約'</code>	抽出条件を表すクエリを指定します。
<code>metric='count'</code>	メトリクスとしてドキュメントの件数を指定します。
<code>timefield='time'</code>	時系列を表すフィールドを指定します。

- `.bars()`関数をメソッドチェーン形式で呼び出し、グラフの種類を「棒グラフ」に指定します。
- `.label()`関数をメソッドチェーン形式で呼び出し、凡例に表示されるラベルを設定します。

パラメータ	説明
'新規契約の推移'	凡例に表示されるラベルを設定します。

- 2行目は、「業務A_新規契約」の完了件数の推移の傾向（増加傾向／現象傾向）を表示するための記述です。
`.es()` 関数への引数は1行目と同様ですが、メソッドチェーン形式で呼び出す関数が異なります。

```
.es(index='im_bpm*', q='type_activiti:PROCESS_COMPLETED AND variables.selectedTaskName:業務A_新規契約', metric='count', timefield='time').trend().lines(width=2).label('新規契約の推移の傾向'),
```

- `.trend()`関数をメソッドチェーン形式で呼び出し、グラフの傾向（増加傾向／現象傾向）を表示することを指定します。
- `.label()`関数をメソッドチェーン形式で呼び出し、凡例に表示されるラベルを設定します。

パラメータ	説明
'新規契約の推移の傾向'	凡例に表示されるラベルを設定します。

- 3行目は、KPIとして静的な基準線を表示するための記述です。
ここでは、一日の目標件数を 2 件として設定しています。
`.static()` 関数を使用し、静的な基準線を表示します。

```
.static(value=2).lines(width=1, fill=1).color(#000000).label('目標値')
```

- `.static()`関数のパラメータ

パラメータ	説明
value=2	基準線の値を指定します。

- `.lines()`関数をメソッドチェーン形式で呼び出し、グラフの種類を「折れ線グラフ」に指定します。

パラメータ	説明
width=1	線の幅を指定します。
fill=1	塗りつぶしの濃さを指定します。

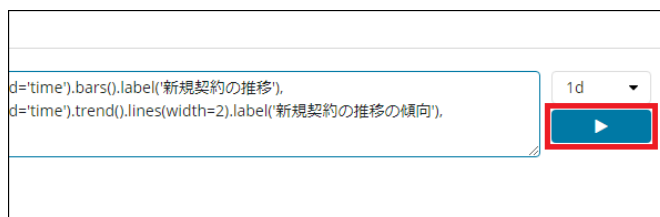
- `.color()`関数をメソッドチェーン形式で呼び出し、グラフの色を指定します。

パラメータ	説明
#000000	16進数カラーコードを指定します。

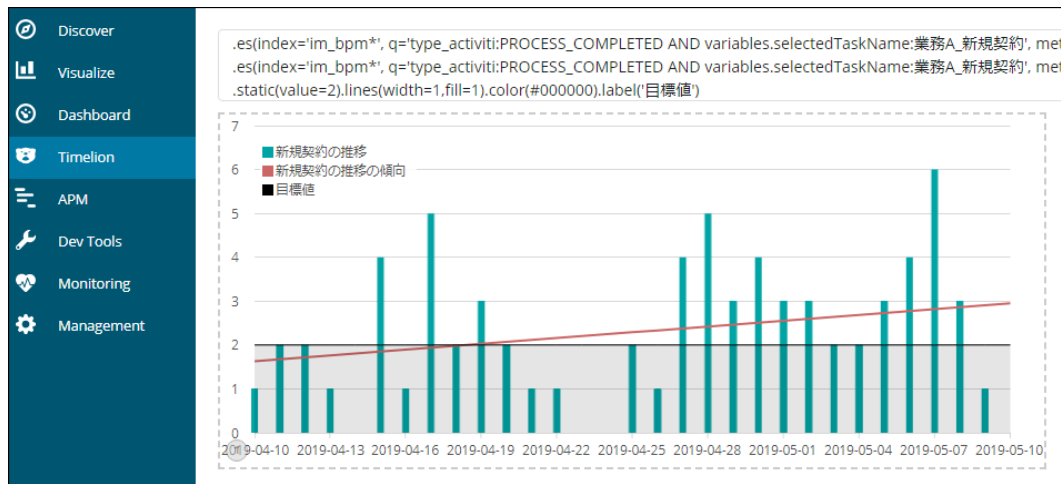
- `.label()`関数をメソッドチェーン形式で呼び出し、凡例に表示されるラベルを設定します。

パラメータ	説明
'目標値'	凡例に表示されるラベルを設定します。

- 「Apply Changes」をクリックし、「新規契約の推移」の棒グラフと、「目標値」KPI、「新規契約の推移の傾向」が表示されることを確認します。



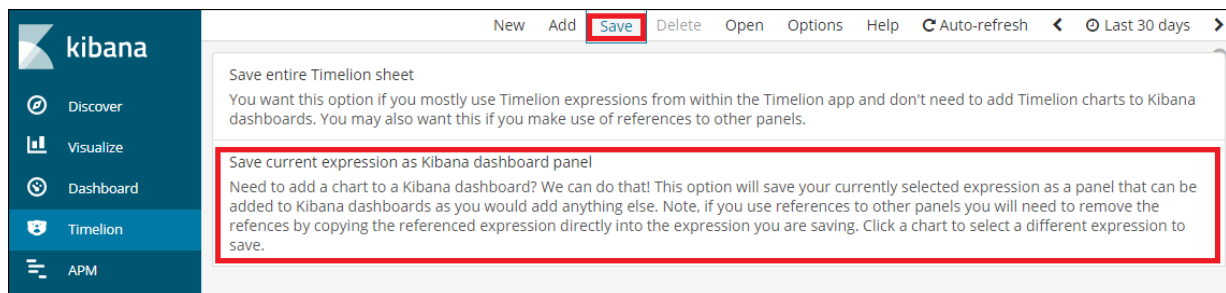
図：「Apply Changes」



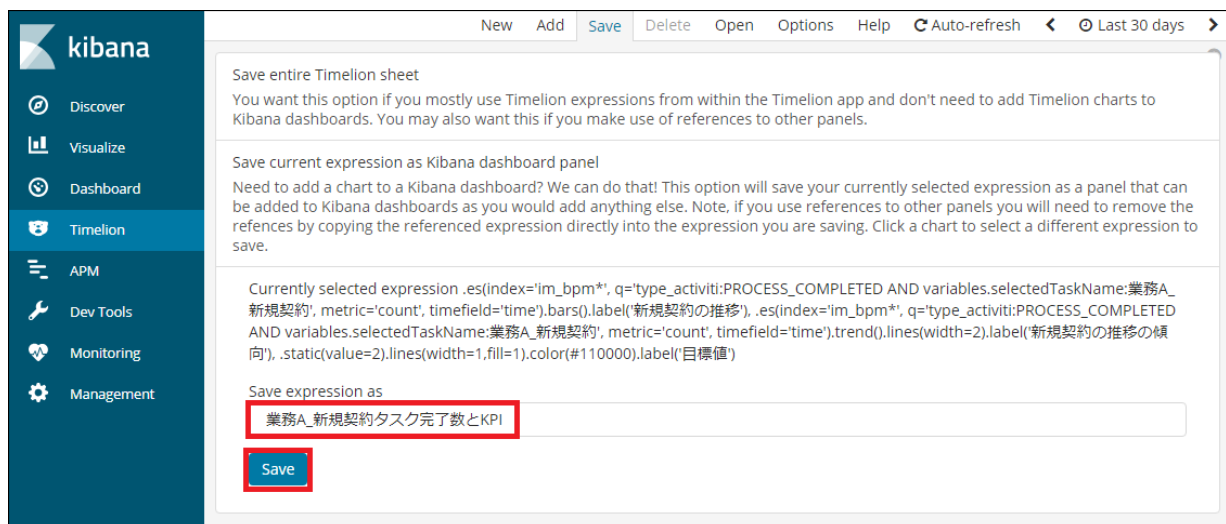
図：「新規契約の推移」の棒グラフと、「目標値」KPI、「新規契約の推移の傾向」

6. グラフに名前を付けダッシュボードに表示できる形式で保存します。
「ヘッダメニュー」の「Save」をクリックし、表示されたメニューより「Save current expression as Kibana dashboard panel」をクリックし、「Save expression as」に下記の値を設定し、「Save」をクリックします。

項目名	設定値
Save expression as	業務A_新規契約タスク完了数とKPI



図：「Save current expression as Kibana dashboard panel」

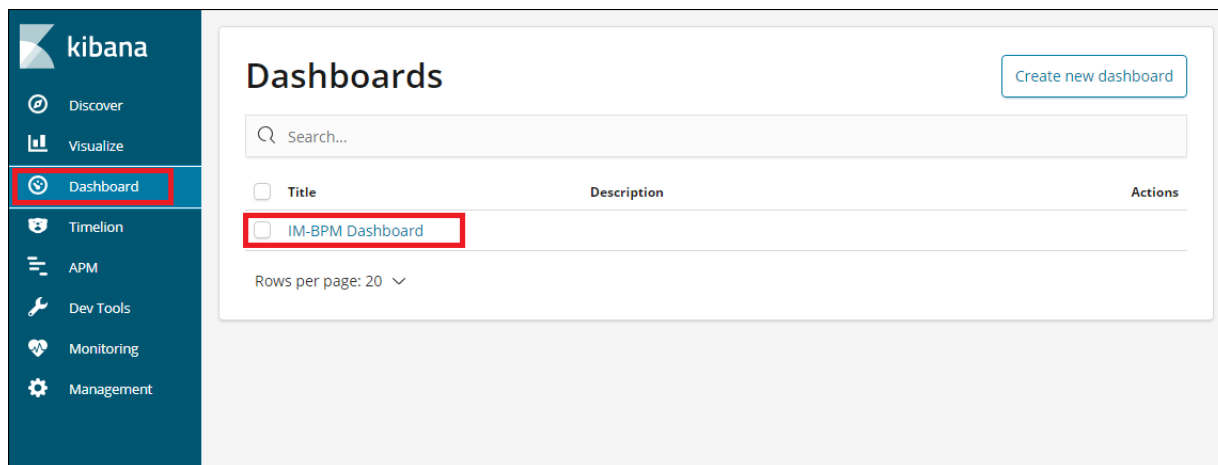


図：「Save expression as」

「業務A_新規契約タスク完了数とKPI」をダッシュボードに追加する

Kibana上に上記で作成した「業務A_新規契約タスク完了数とKPI」を「IM-BPM Dashboard」ダッシュボードに追加します。

1. Kibanaのメインメニューより「Dashboard」をクリックし、「Dashboards」画面の「IM-BPM Dashboard」をクリックします。



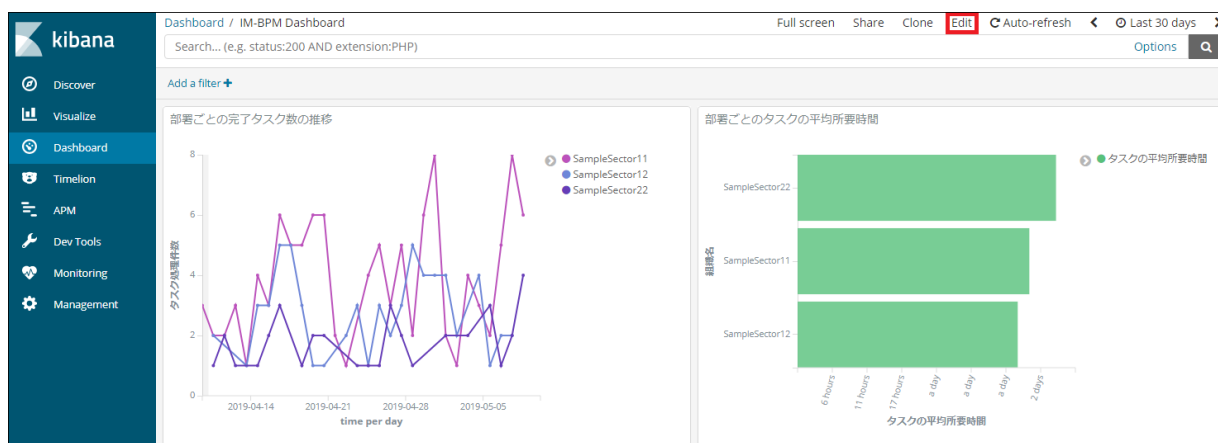
図：「Dashboards」画面

コラム

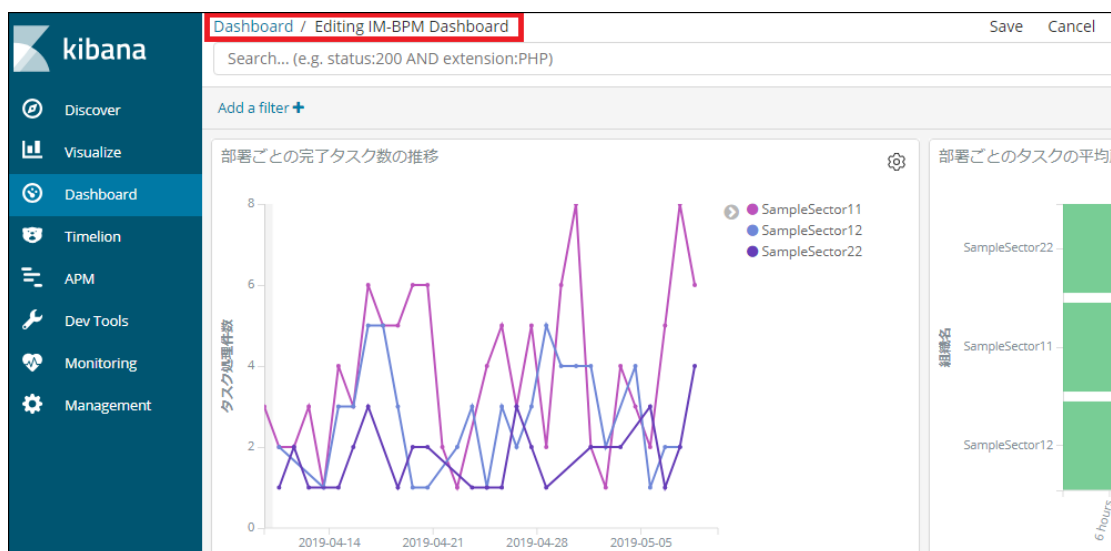
下記のような画面が表示され、データが表示されていない場合、「Time Range」（画面右上の「Last 15 minutes」と表示されている項目）より、データの表示期間を「Last 30 days」へ変更してください。

図：データが表示されていない場合

2. 「Dashboard / IM-BPM Dashboard」画面より、ヘッダメニューの「Edit」をクリックし、「Dashboard / Editing IM-BPM Dashboard」画面を開きます。

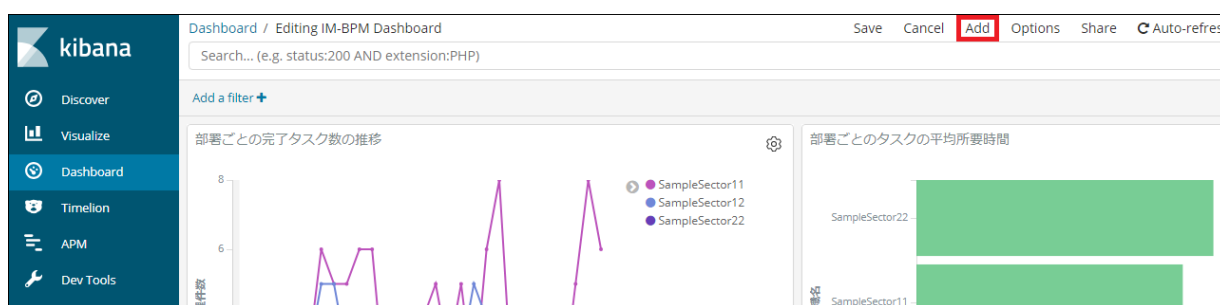


図：「IM-BPM Dashboard」



図：「Dashboard / Editing IM-BPM Dashboard」画面

3. ヘッダメニューの「Add」をクリックし、「Add Panels」を開きます。



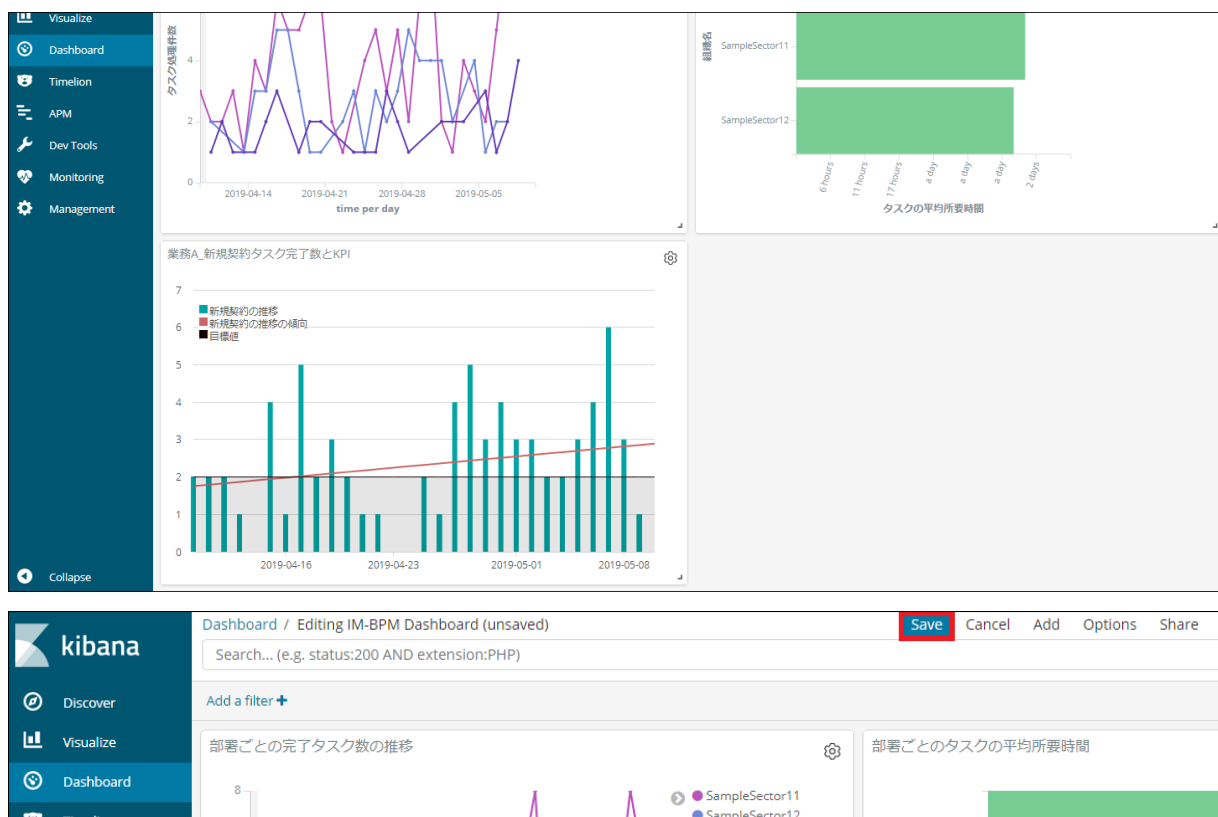
図：「Editing New Dashboard」→「Add」

4. 「Add Panels」→「Visualization」タブより「業務A_新規契約タスク完了数とKPI」をクリックします。



図：「Add Panels」→「Visualization」

5. ダッシュボードに「業務A_新規契約タスク完了数とKPI」が追加されたことを確認し、ヘッダメニューの「Save」をクリックします。



図：「Save」

6. 「Confirm Save」をクリックします。

The screenshot shows the 'Save Dashboard' dialog box in Kibana. The dialog has a title bar with a close button (X). The main content area contains the following fields and options:

- Save as a new dashboard:** A toggle switch that is currently turned off.
- Title:** A text input field containing 'IM-BPM Dashboard'.
- Description:** A large text area for adding a description.
- Store time with dashboard:** A toggle switch that is currently turned off.
- Buttons:** At the bottom, there are two buttons: 'Cancel' and 'Confirm Save'. The 'Confirm Save' button is highlighted with a red border.

図：「Save Dashboard」

