Copyright © 2017 NTT DATA INTRAMART CORPORATION

目次

- 1. 改訂情報
- 2. はじめに
  - 2.1. 本書の目的
  - 2.2. 対象読者
  - 2.3. 本書の構成
- 3. 概要
  - 3.1. IM-BPMとは
  - 3.2. 本チュートリアルガイドの説明範囲
  - 3.3. 事前準備
    - 3.3.1. 環境セットアップ
    - 3.3.2. チュートリアル実行ユーザ
- 4. プロセスの作成
  - 4.1. 基礎編
    - 4.1.1. チュートリアルの概要（作成物のイメージ）
    - 4.1.2. プロセス定義を新規に作成する
    - 4.1.3. プロセス定義を IM-BPM Runtimeへデプロイする
    - 4.1.4. プロセスを開始する
  - 4.2. 基礎編（ケース定義）
    - 4.2.1. チュートリアルの概要（作成物のイメージ）
    - 4.2.2. ケース定義を新規に作成する
    - 4.2.3. ケース定義を IM-BPM Runtimeへデプロイする
    - 4.2.4. ケースを開始する
    - 4.2.5. ケースインスタンスの詳細を確認する
    - 4.2.6. ケース定義の詳細を確認する
  - 4.3. 実用編
    - 4.3.1. データプロパティ
      - 4.3.1.1. データプロパティを定義する
    - 4.3.2. マルチインスタンス
      - 4.3.2.1. マルチインスタンス を使用する
    - 4.3.3. リスナ
      - 4.3.3.1. IM-LogicDesignerのリスナを利用する
      - 4.3.3.2. 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
      - 4.3.3.3. 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
      - 4.3.3.4. タスクリスナを利用して、ユーザタスクの期限日時を直前ユーザタスクの期限日時から3日後に設定する
    - 4.3.4. 関連ドキュメント
      - 4.3.4.1. IM-Wikiを関連ドキュメントとして設定する
    - 4.3.5. オプショナルタスク
      - 4.3.5.1. オプショナルタスクを追加する
      - 4.3.5.2. オプショナルタスクのパラメータをFormaアプリケーションと連携する
      - 4.3.5.3. IM-Repositoryの列挙型をオプショナルタスクのパラメータで使用する
    - 4.3.6. アドホックタスク
      - 4.3.6.1. アドホックタスクを利用する
    - 4.3.7. プール、レーン
      - 4.3.7.1. プールとレーンを利用して作業の関係者を明確にする
    - 4.3.8. 強制終了イベント
      - 4.3.8.1. 強制終了イベントを利用して、プロセスを強制終了する
    - 4.3.9. イベントサブプロセス
      - 4.3.9.1. 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
    - 4.3.10. サービスタスク
      - 4.3.10.1. e Builderで作成したJAVAクラスをサービスタスクで使用する
    - 4.3.11. 受信タスク
      - 4.3.11.1. 外部システムで実行されるタスクを受信タスクを用いて表現する
    - 4.3.12. コールアクティビティ
      - 4.3.12.1. コールアクティビティを使用する
      - 4.3.12.2. 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
    - 4.3.13. イベントゲートウェイ
      - 4.3.13.1. イベントゲートウェイを使用する

- 4.3.14. パラレルゲートウェイ
  - 4.3.14.1. パラレルゲートウェイを使用する
- 4.3.15. 排他ゲートウェイ
  - 4.3.15.1. 排他ゲートウェイを使用する
- 4.3.16. 包括ゲートウェイ
  - 4.3.16.1. 包括ゲートウェイを使用する
- 4.3.17. イベント
  - 4.3.17.1. シグナルイベントを使用する
  - 4.3.17.2. メッセージイベントを使用する
  - 4.3.17.3. タイマイベントを使用する
  - 4.3.17.4. エラーイベントを使用する
- 4.3.18. IM-LogicDesignerタスク
  - 4.3.18.1. IM-LogicDesignerタスクを利用してユーザ情報を取得する
  - 4.3.18.2. 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
  - 4.3.18.3. IM-LogicDesignerタスクを利用してOpenRulesを使用する
  - 4.3.18.4. IM-LogicDesignerタスクを利用してメッセージを送信する際に指定するエグゼキューションを取得する
  - 4.3.18.5. IM-LogicDesignerタスクを用いて、IM-FormaDesignerで入力したテーブルアイテムの値を扱う
- 4.3.19. 申請タスク
  - 4.3.19.1. 申請タスクを使用してワークフローと連携する
- 4.3.20. 起票タスク
  - 4.3.20.1. 起票タスクを使用してワークフローと連携する
  - 4.3.20.2. 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
  - 4.3.20.3. 起票タスクと変数を連携する
- 4.3.21. IM-FormaDesigner for Accel Platform をユーザ入力フォームとして利用する
  - 4.3.21.1. 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
  - 4.3.21.2. 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

- 4.4. 発展編
  - 4.4.1. ワークフローとの連携
    - 4.4.1.1. 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
  - 5.4. 差配管理
    - 5.4.1. 差配管理画面
      - 5.4.1.1. プロセスのタスク状況を確認する
      - 5.4.1.2. タスクの担当者を変更する

- 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

- 7. サンプル集
  - 7.1. プロセスの作成
    - 7.1.1. IM-BloomMaker
      - 7.1.1.1. IM-BloomMakerのコンテンツをユーザ入力フォームとして利用する
      - 7.1.1.2. IM-BloomMakerで作成した画面の入力値をもとにタスクの期限日時を設定する
  - 7.2. プロセスの管理
    - 7.2.1. IM-BloomMakerを利用してアドオンのプロセス一覧画面を作成する（1）
      - 7.2.1.1. 本サンプルの確認方法
      - 7.2.1.2. 本サンプルの構成資材
      - 7.2.1.3. IM-LogicDesigner 定義情報（**im_logicdesigner-data_timezone.zip**）の詳細
      - 7.2.1.4. IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_list.zip**）の詳細
      - 7.2.1.5. IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list-BM.xml**）の詳細
    - 7.2.2. IM-BloomMakerを利用してアドオンのプロセス一覧画面を作成する（2）
      - 7.2.2.1. 本サンプルの確認方法
      - 7.2.2.2. 本サンプルの構成資材
      - 7.2.2.3. IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_addon_process_list_user.zip**）の詳細
      - 7.2.2.4. IM-LogicDesigner 定義情報（**im_logicdesigner-data_timezone.zip**）の詳細
      - 7.2.2.5. IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_list_user.zip**）の詳細
      - 7.2.2.6. IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_user-BM.xml**）の詳細
      - 7.2.2.7. IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_user-LD.xml**）の詳細
    - 7.2.3. IM-BloomMakerを利用してアドオンのプロセス一覧画面を作成する（3）
      - 7.2.3.1. 本サンプルの確認方法
      - 7.2.3.2. 本サンプルの構成資材
      - 7.2.3.3. IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_addon_process_list_join_table.zip**）の詳細
      - 7.2.3.4. IM-BPM定義情報（**im_bpm-bloommaker_addon_process_list_join_table.zip**）の詳細
      - 7.2.3.5. IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_list_join_table.zip**）の詳細
      - 7.2.3.6. IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_join_table-BM.xml**）の詳細
      - 7.2.3.7. IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_join_table-LD.xml**）の詳細
    - 7.2.4. プロセス図を表示するIM-BloomMakerエレメントを利用したアドオン画面の作成（1）
      - 7.2.4.1. 本サンプルの構成資材
      - 7.2.4.2. 本サンプルの使用方法
      - 7.2.4.3. IM-BloomMaker定義情報（**im_bloommaker-data-process_diagram_bloommaker_element-show_activity_history.zip**）の詳細
    - 7.2.5. プロセス図を表示するIM-BloomMakerエレメントを利用したアドオン画面の作成（2）
      - 7.2.5.1. 本サンプルの構成資材
      - 7.2.5.2. 本サンプルの使用方法
      - 7.2.5.3. IM-BloomMaker定義情報（**im_bloommaker-data-process_diagram_bloommaker_element-**

show_task_assignee.zip）の詳細
- 7.2.6. IM-BloomMakerを利用してアドオンのプロセス詳細画面を作成する
  - 7.2.6.1. 本サンプルの確認方法
  - 7.2.6.2. 本サンプルの構成資材
  - 7.2.6.3. IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_addon_process_detail.zip**）の詳細
  - 7.2.6.4. IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_detail.zip**）の詳細
  - 7.2.6.5. IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_detail-BM.xml**）の詳細
- 7.3. プロセスの分析
  - 7.3.1. IM-BloomMakerを利用してプロセス定義のグラフを作成する
    - 7.3.1.1. 本サンプルの構成資材
    - 7.3.1.2. 本サンプルの使用方法
    - 7.3.1.3. IM-LogicDesigner定義情報（**im_logicdesigner-data-bloommaker_addon_process_definition_graph.zip**）の詳細
    - 7.3.1.4. IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_definition_graph.zip**）の詳細

改訂情報

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

変更年月日	変更内容
2018-12-27	第10版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*包括ゲートウェイを使用する*」 ■ 各エレメントの詳細を追記しました。 ■ 「IM-BPM プロセスデザイナ 操作ガイド」に説明を対応しました。
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*」 　■ 「*オプショナルタスクを追加する*」 ■ 「*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*」
2019-12-01	第14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。 　■ 「*オプショナルタスクのパラメータをFormaアプリケーションと連携する*」 　■ 「*IM-Repositoryの列挙型をオプショナルタスクのパラメータで使用する*」 ■ 「*プロセスの作成*」 - 「*基礎編（ケース定義）*」を追加しました。
2020-04-01	第15版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*基礎編（ケース定義）*」に追記しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*アドホックタスクを利用する*」

変更年月日	変更内容
2020-08-01	第16版 下記を追加・変更しました。 ■ 「*はじめに*」に関連するドキュメントを追記しました。 ■ 「*はじめに*」の「*本書の構成*」に追記しました。 ■ 「*概要*」 - 「*事前準備*」に追記・修正しました。 ■ 「*プロセスの作成*」 - 「*実用編*」を修正しました。 ■ 「*プロセスの作成*」 - 「*発展編*」を修正しました。 ■ 「*プロセスの作成*」 - 「*実用編*」に下記のページを追加しました。 　■ 「*外部システムで実行されるタスクを受信タスクを用いて表現する*」 ■ 「*サンプル集*」を追加しました。
2020-12-01	第17版 下記を追加・変更しました。 ■ 「*サンプル集*」 - 「*プロセスの管理*」に下記のページを追加しました。 　■ 「*プロセス図を表示するIM-BloomMakerエレメントを利用したアドオン画面の作成（1）*」 　■ 「*プロセス図を表示するIM-BloomMakerエレメントを利用したアドオン画面の作成（2）*」 　■ 「*IM-BloomMakerを利用してアドオンのプロセス詳細画面を作成する*」 ■ 「*プロセスの作成*」-「*実用編*」-「*IM-LogicDesignerタスク*」に下記のページを追加しました。 　■ 「*IM-LogicDesignerタスクを利用してメッセージを送信する際に指定するエグゼキューションを取得する*」 ■ 「*サンプル集*」 - 「*プロセスの分析*」 - 「*IM-BloomMakerを利用してプロセス定義のグラフを作成する*」を修正しました。
2021-04-01	第18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。 　■ 「*起票タスクと変数を連携する*」 ■ IM-LogicDesignerの「ロジックフロー定義一覧」画面の項目名変更に伴いドキュメント内の文言、および、キャプチャを修正しました。
2024-04-01	第19版 下記を追加・変更しました。 ■ 「*プロセスの作成*」 - 「*基礎編*」にプロセス管理ユーザの設定手順を追加しました。 ■ 「*プロセスの作成*」-「*実用編*」 - 「*IM-LogicDesignerのリスナを利用する*」にフロー定義編集の注釈を追加しました。 ■ 「*プロセスの作成*」-「*実用編*」 - 「*IM-LogicDesignerタスクを利用してユーザ情報を取得する*」にフロー定義編集の注釈を追加しました。 ■ 「*プロセスの管理*」 - 「*差配管理*」のページを追加しました。 ■ 「*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タスクを用いて、IM-FormaDesignerで入力したテーブルアイテムの値を扱う*」
2024-04-26	第20版 下記を追加・変更しました。 ■ 「*プロセスの管理*」 - 「*差配管理*」のページにプロセス管理ユーザ・グループ設定の前提条件を追加しました。
2024-06-28	第21版 下記を追加しました。 ■ 「*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、ユーザタスクの期限日時を直前ユーザタスクの期限日時から3日後に設定する*」 ■ 「*サンプル集*」 - 「*プロセスの作成*」に下記のページを追加しました。 　■ 「*IM-BloomMakerで作成した画面の入力値をもとにタスクの期限日時を設定する*」

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 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
- IM-Knowledge管理者操作ガイド
- ViewCreator 管理者操作ガイド
- IM-Repository ユーザ操作ガイド
- IM-Repository拡張プログラミングガイド
- チケットモジュール管理者操作ガイド
- IM-BloomMaker for Accel Platform ユーザ操作ガイド
- IM-BloomMaker for Accel Platform プログラミングガイド
- IM-BloomMaker for Accel Platform チュートリアルガイド

本書の構成

- *概要*

  本書、および、IM-BPMの概要について説明します。

- *プロセスの作成*

  プロセスを作成する機能について説明します。

  - *基礎編*

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

  - *基礎編（ケース定義）*

    基礎編として、シンプルなケース定義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、ケース定義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。

  - *発展編*

    発展編として、他アプリケーションを組み合わせて、発展的な機能を実現する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。

- *サンプル集*

  独自に作成した画面との連携処理などのサンプル集です。

概要

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

IM-BPMとは

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

> IM-BPM 仕様書 - 3.1 IM-BPM for Accel Platformとは
>
> IM-BPMとは、intra-mart Accel Platform上で業務プロセスを実行することが可能なアプリケーションです。
> IM-BPMの特徴は以下の通りです。
>
> - BPMN2.0形式に対応した業務プロセスを実行できます。
> - IM-FormaDesignerと連携し、システムに必要となる画面と簡単に連携できます。
> - IM-LogicDesignerと連携し、システム間連携等の業務ロジックを簡単に呼び出せます。
> - 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。）
   - IM-Knowledgeモジュール（プロセスの作成の実用編を行う場合に必要です。）
   - ViewCreatorモジュール（プロセスの管理編を行う場合に必要です。）
   - IM-BloomMaker for Accel Platformモジュール（IM-BloomMakerを使用するサンプル、またはチュートリアルを行う場合に必要です。）
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

本チュートリアルのプロセスやロジックフローの作成は 「テナント管理者」で行ってください。
また、プロセスの作成以外の操作は全て以下の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。

- 「IM-BPM管理者（im_bpm_manager）」
- 「IM-BPMユーザ（im_bpm_user）」

なお、本チュートリアルで作成するプロセスの実行は、特に記載がない場合は上記のロールを付与したユーザ「青柳辰巳（aoyagi）」で行ってください。

コラム

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

プロセスの作成

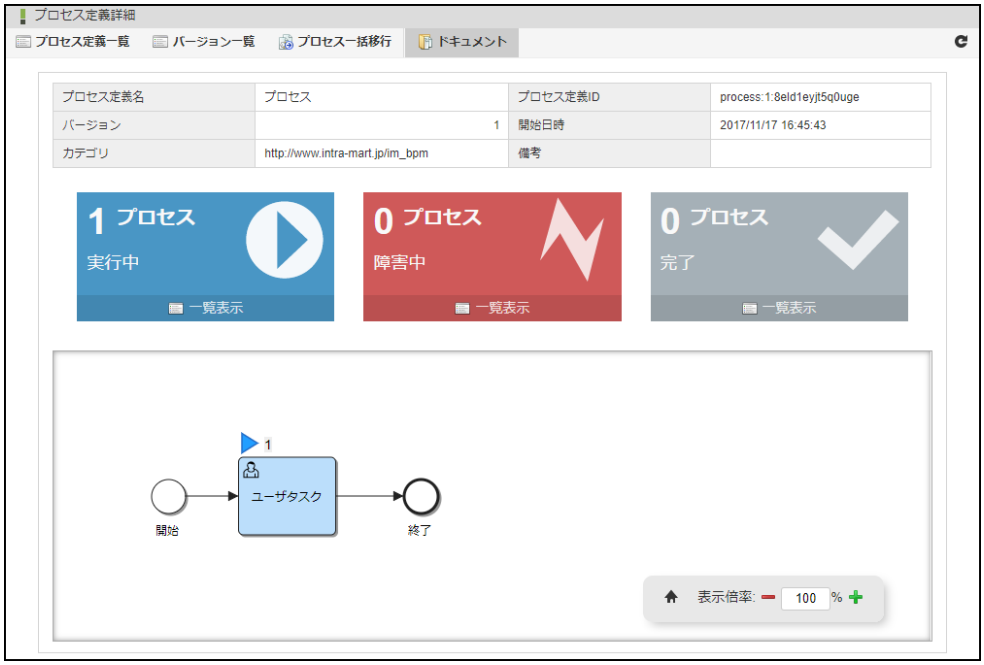
基礎編

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

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

チュートリアルの概要（作成物のイメージ）

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

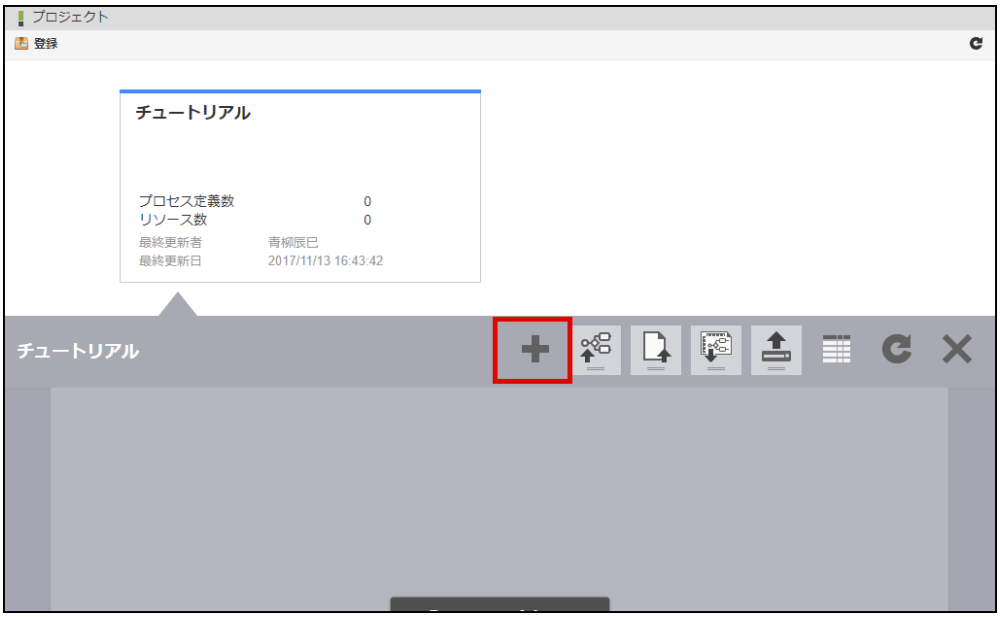


図：プロセス定義詳細

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。

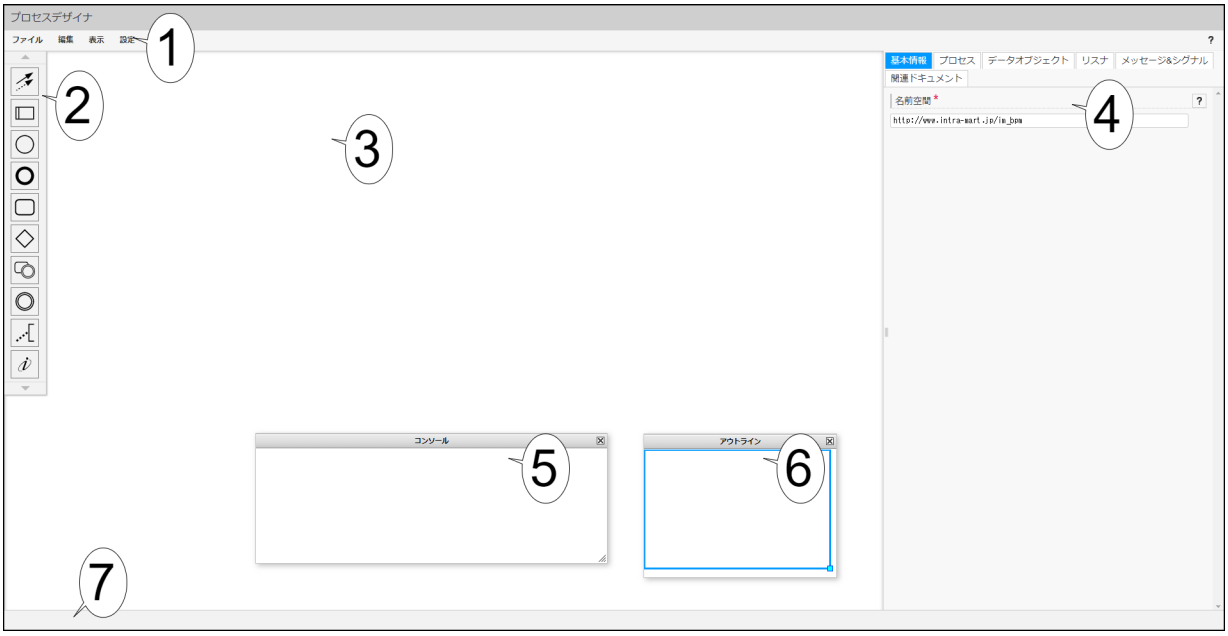
1. 「サイトマップ」→「BPM」→「プロセスデザイナ」をクリックします。
2. 新規プロジェクトを作成し、 をクリックします。



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 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

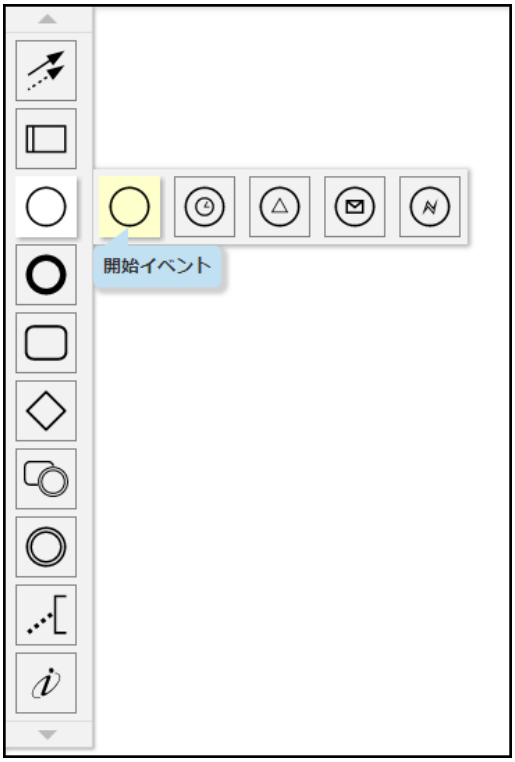
   プロセス全体を俯瞰表示する領域です。
   表示領域の移動や、表示の拡大縮小ができます。

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。

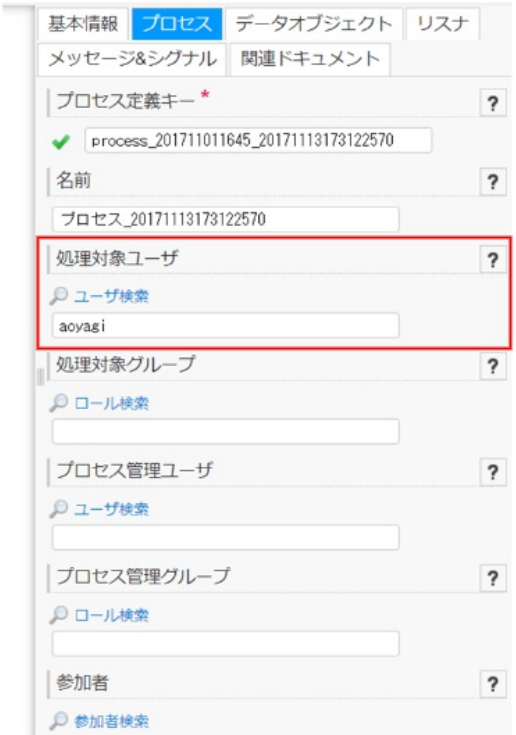
5. 「パレット」の右側に現れる一覧から、 をドラッグし、「キャンバス」上の任意の位置でドロップして配置します。



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。

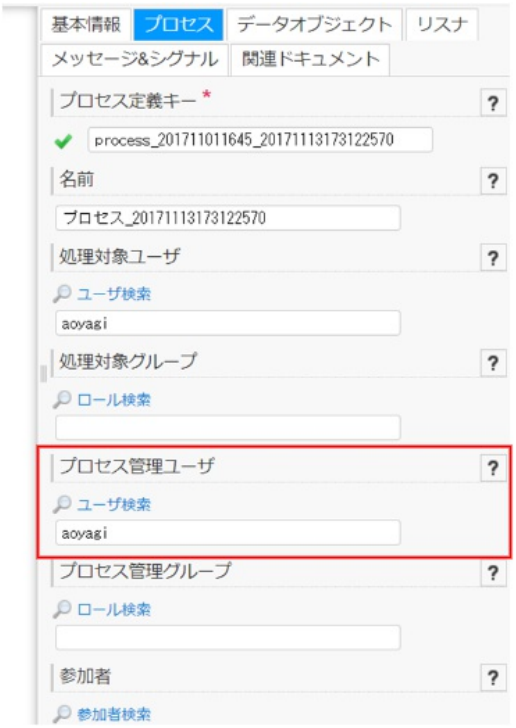
   プロセスを開始するユーザを設定してください。



図：プロセス - プロパティ - 処理対象ユーザ

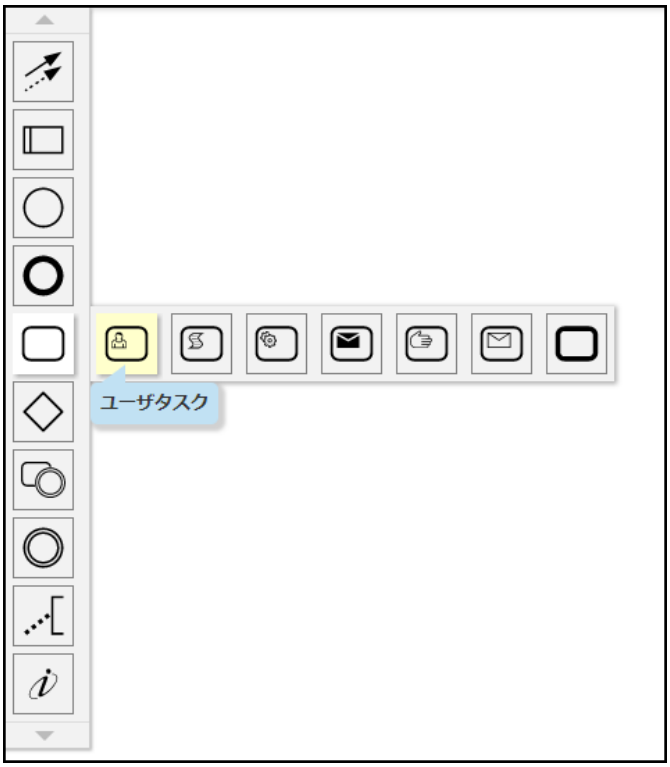
7. 「プロパティエリア」にて、「プロセス」の「プロセス管理ユーザ」を設定します。

   プロセスを管理するユーザを設定してください。



図：プロセス - プロパティ - プロセス管理ユーザ

8. 「パレット」から「キャンバス」に「ユーザタスク」を配置します。



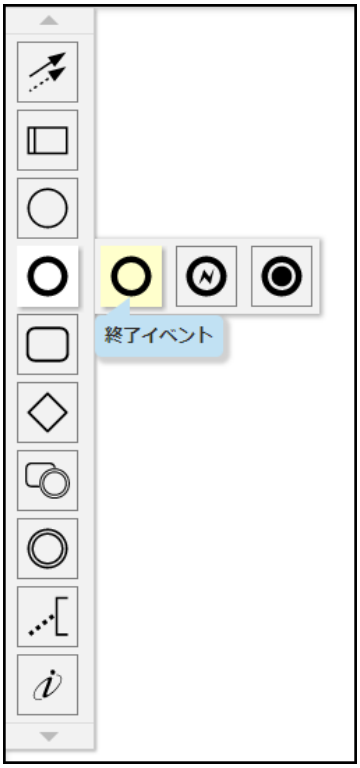
図：パレット - タスク - ユーザタスク

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

   タスクを処理するユーザを設定してください。

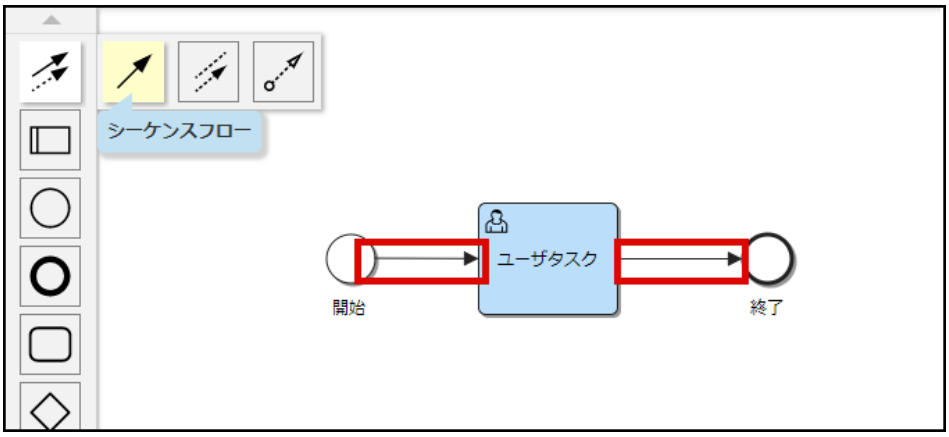
図：ユーザタスク - プロパティ - メインコンフィグ - 処理対象ユーザ

10. 「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

11. 「シーケンスフロー」で各エレメントを接続します。

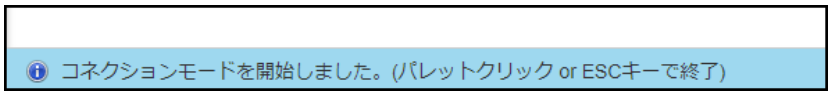


図：パレット - コネクション - シーケンスフロー

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。

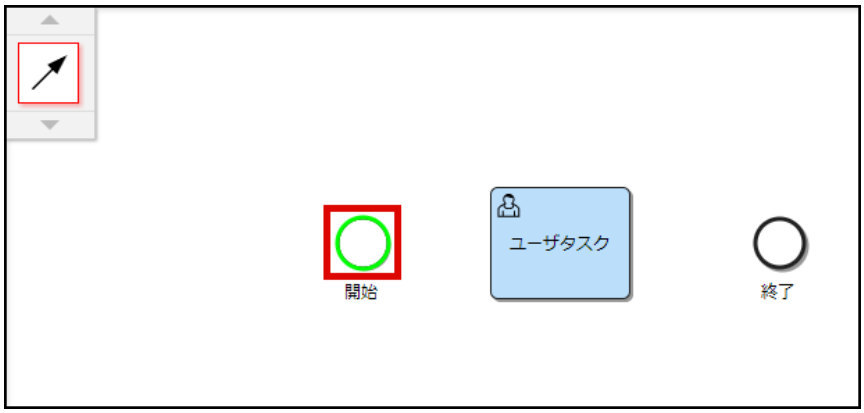
   コネクションモードが開始すると、フッタにメッセージが表示されます。



   図：フッタ

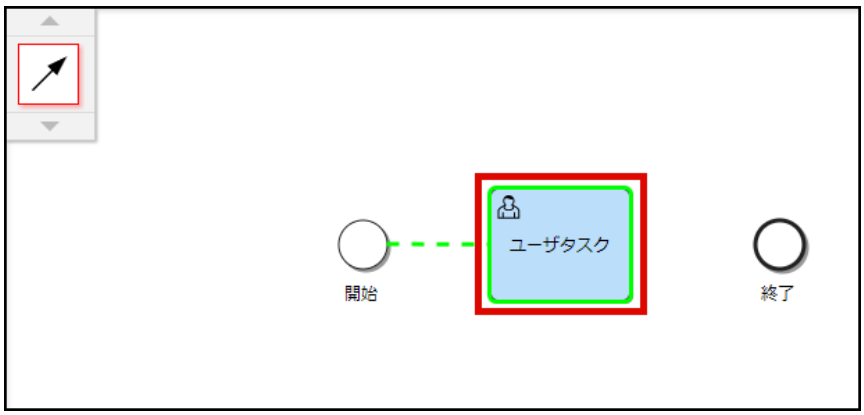
3. 「キャンバス」に配置された「開始イベント」をクリック、または、ドラッグして接続を開始します。

   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。

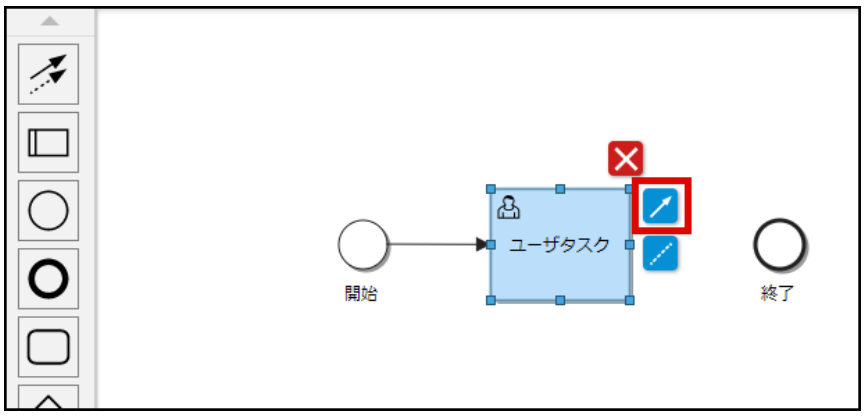


   図：接続する

5. キーボードの「ESC」キーを押すか、「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。

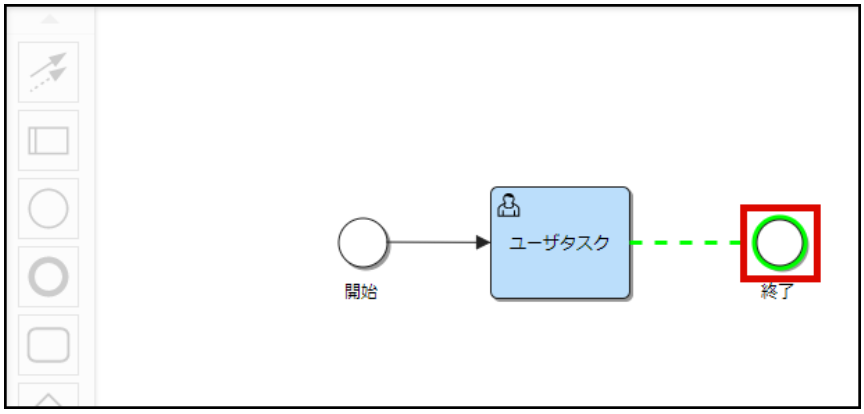
6. キャンバス上の「ユーザタスク」を選択します。

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。



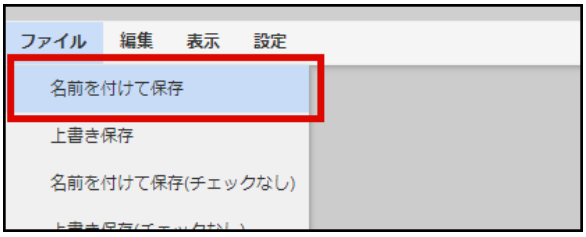
図：接続を開始する

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

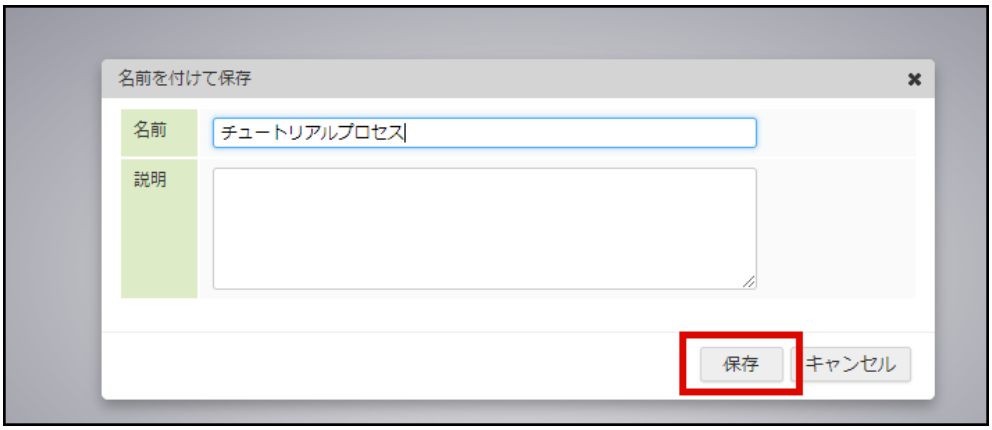
12. 「ヘッダメニュー」から、「ファイル」 - 「名前を付けて保存」をクリックします。



図：ヘッダメニュー - ファイル - 名前を付けて保存

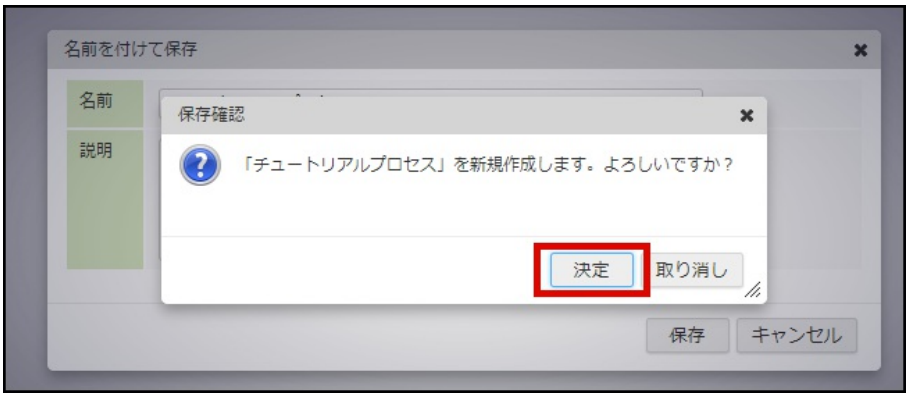
> **コラム**
>
> 「名前を付けて保存」と「名前を付けて保存（チェックなし）」の違い
>
> IM-BPM プロセスデザイナは、作成したプロセス定義がデプロイ可能かどうかをチェックを行う機能を有しています。
> 「名前を付けて保存」を実行した場合は上記チェックが行われ、デプロイ可能と判定されるまで保存できません。
> 「名前を付けて保存（チェックなし）」を実行した場合は上記チェックを行わず、保存が可能となる最低限のチェックのみを行って保存できます。

13. 「名前」を入力して「保存」をクリックします。



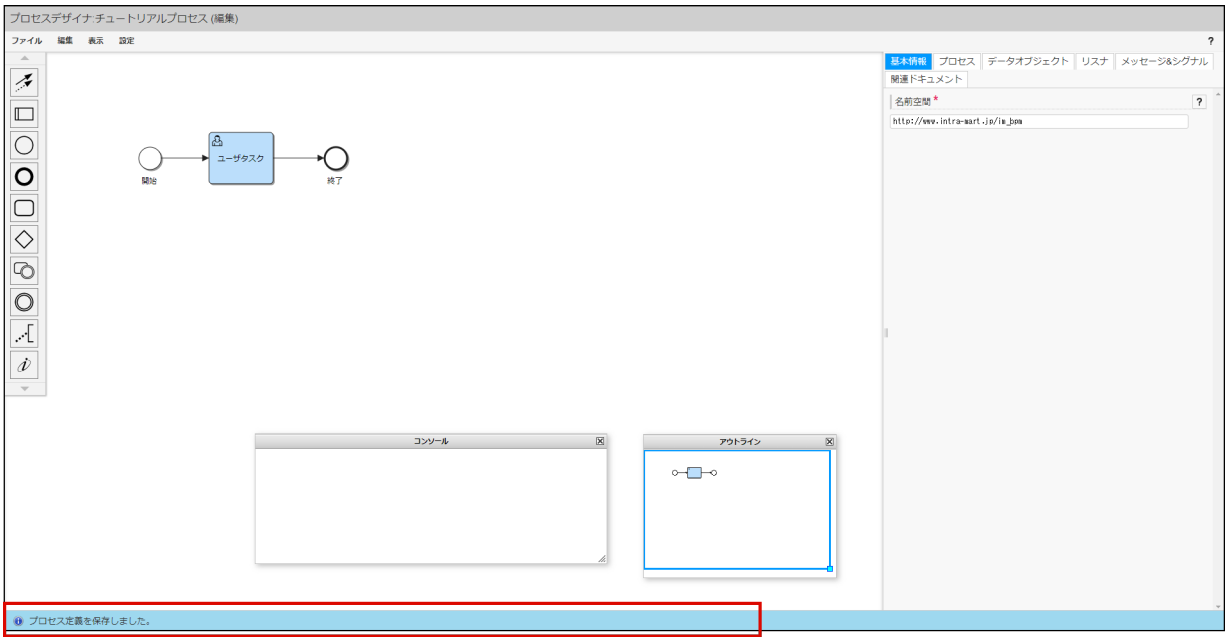
図：名前を付けて保存

14. 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）

15. 保存されました。

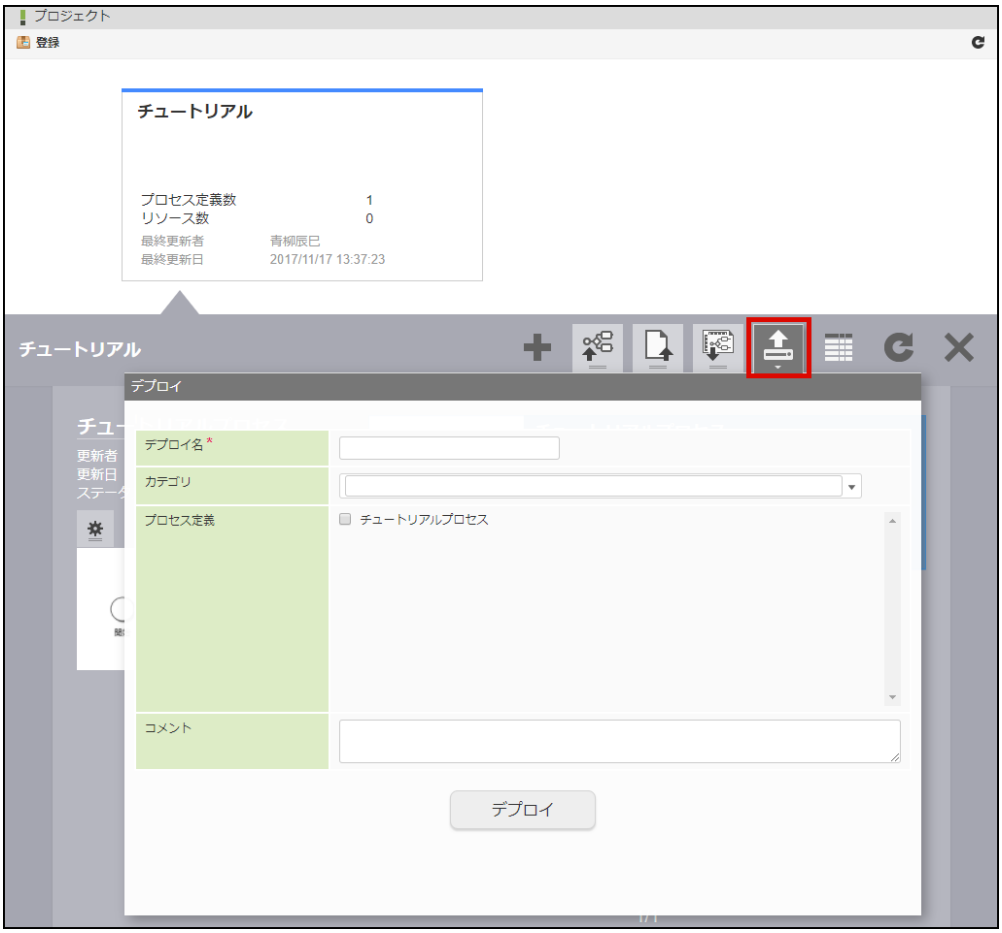


図：プロセス保存

プロセス定義を IM-BPM Runtimeへデプロイする

プロセス定義を IM-BPM Runtimeへデプロイします。

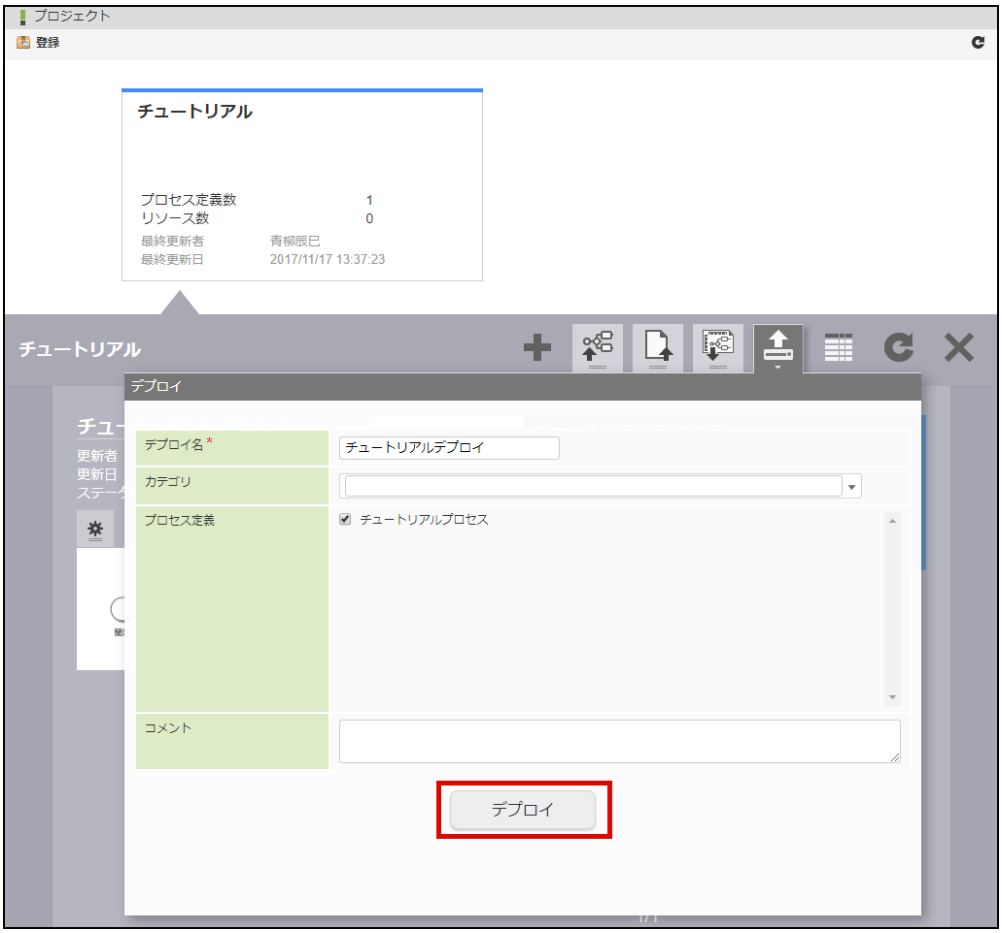
1. 「サイトマップ」→「BPM」→「プロセスデザイナ」をクリックします。
2. プロジェクトを選択し、 をクリックしデプロイエリアを開きます。



図：デプロイエリア

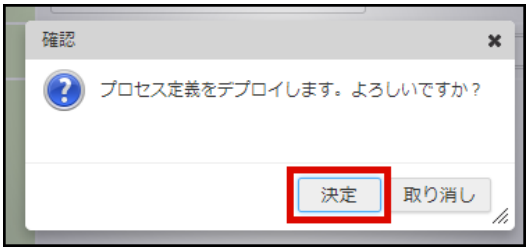
3. 「デプロイ名」の入力と「プロセス定義」の選択をします。
4. 「デプロイ」をクリックします。

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



図：デプロイ

5. 確認ダイアログにて「決定」をクリックします。



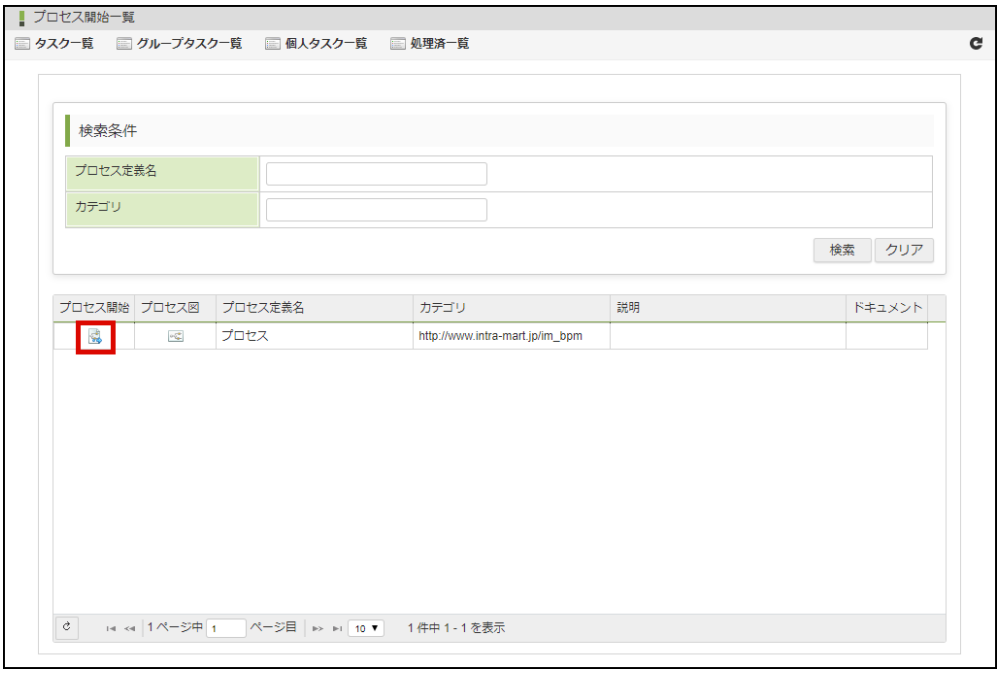
図：デプロイ確認

6. デプロイされました。

プロセスを開始する

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

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



図：プロセス開始一覧

3. プロセスが開始されました。

図：プロセス開始一覧

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

図：タスク一覧

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

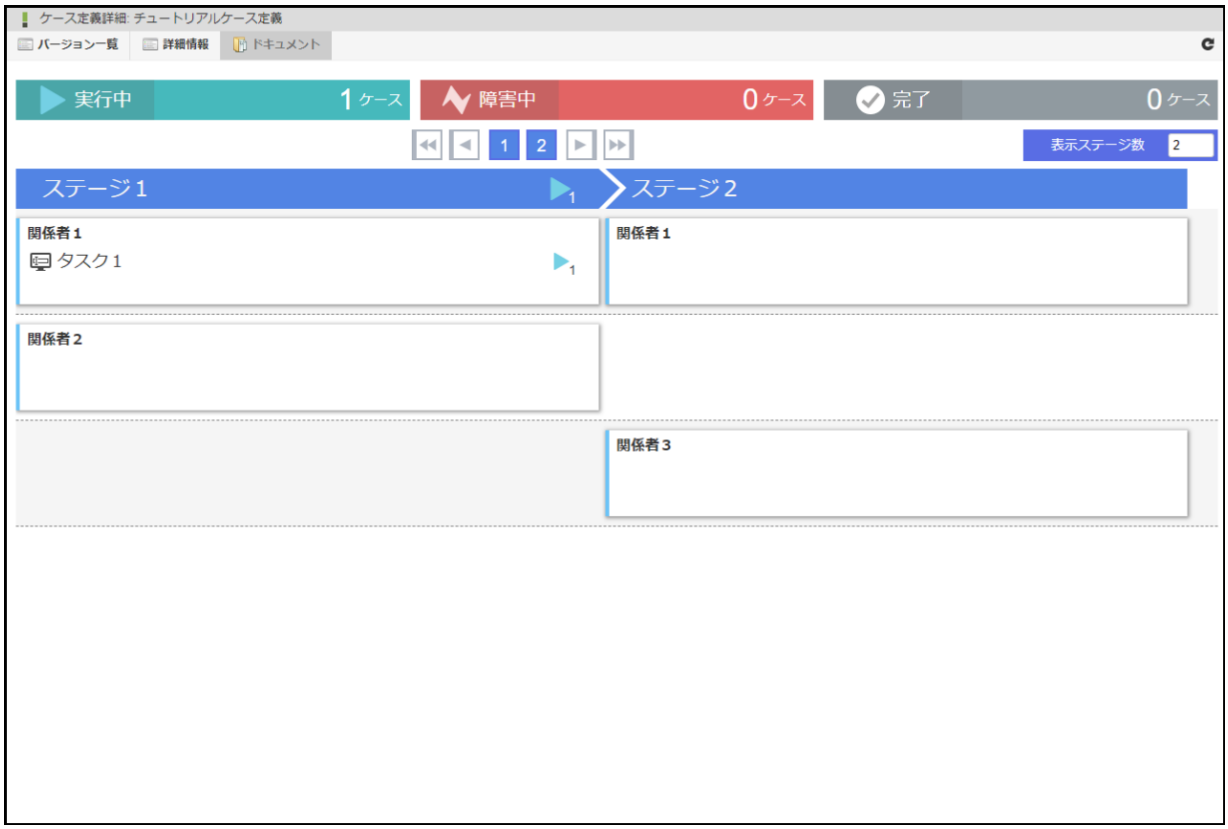
基礎編（ケース定義）

基礎編（ケース定義）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）
- ケース定義を新規に作成する
- ケース定義を IM-BPM Runtimeへデプロイする
- ケースを開始する
- ケースインスタンスの詳細を確認する
- ケース定義の詳細を確認する

チュートリアルの概要（作成物のイメージ）

基礎編（ケース定義）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ケース定義を作成し、IM-BPM Runtimeへケース資材をデプロイし、ケースを開始するまでを説明します。

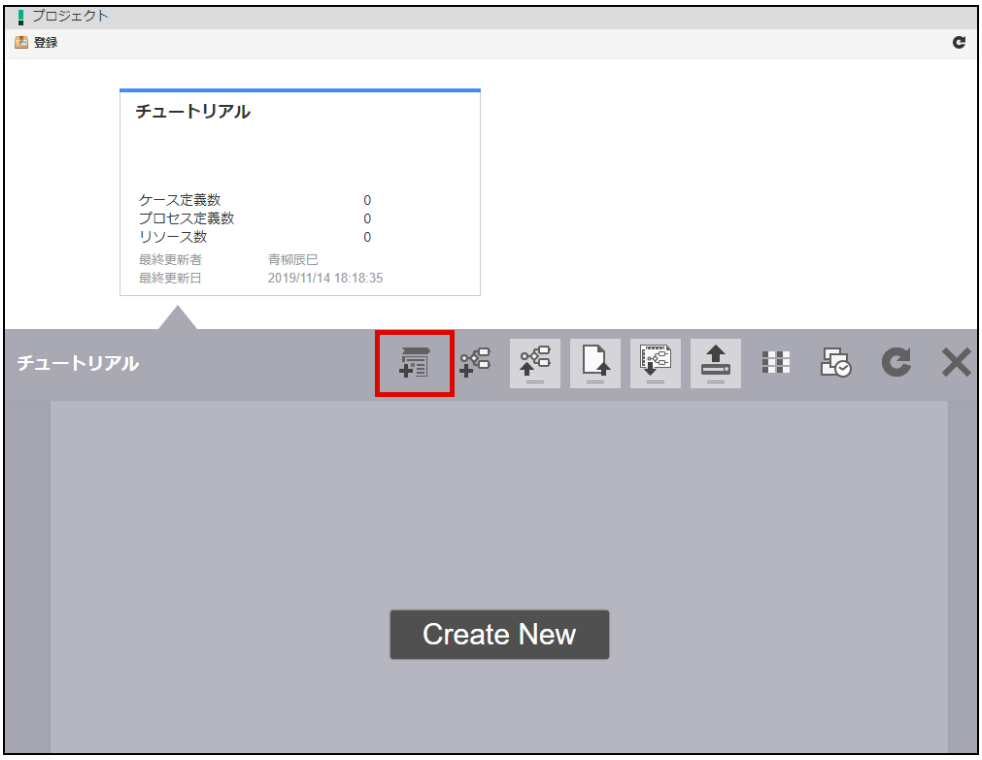


図：「ケース定義詳細」

ケース定義を新規に作成する

ケー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」 - 「ケース定義新規登録」

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

3. ケースエディタが開きます。

図：「ケースエディタ」

1. ヘッダメニュー

   ケースエディタの操作や設定を行うメニューです。

2. ステージ一覧

   ステージの一覧です。

3. 例外ステージ一覧

   例外ステージの一覧です。

4. ステージ詳細

ステージ、および、例外ステージの詳細です。

5. プロパティエリア

   各要素のプロパティを編集します。

4. ケースエディタを開いた際に、ケース定義の初期設定ができるウィザードが開きますが、本チュートリアルでは使用しませんので、ウィザードを終了してください。

5. 「ヘッダメニュー」から、「設定」 - 「ケース定義プロパティ」をクリックします。

図：「ヘッダメニュー」 - 「設定」 - 「ケース定義プロパティ」

6. 「処理対象ユーザ」を設定します。
   プロセスを開始するユーザを設定してください。

図：「ヘッダメニュー」 - 「設定」 - 「ケース定義プロパティ」

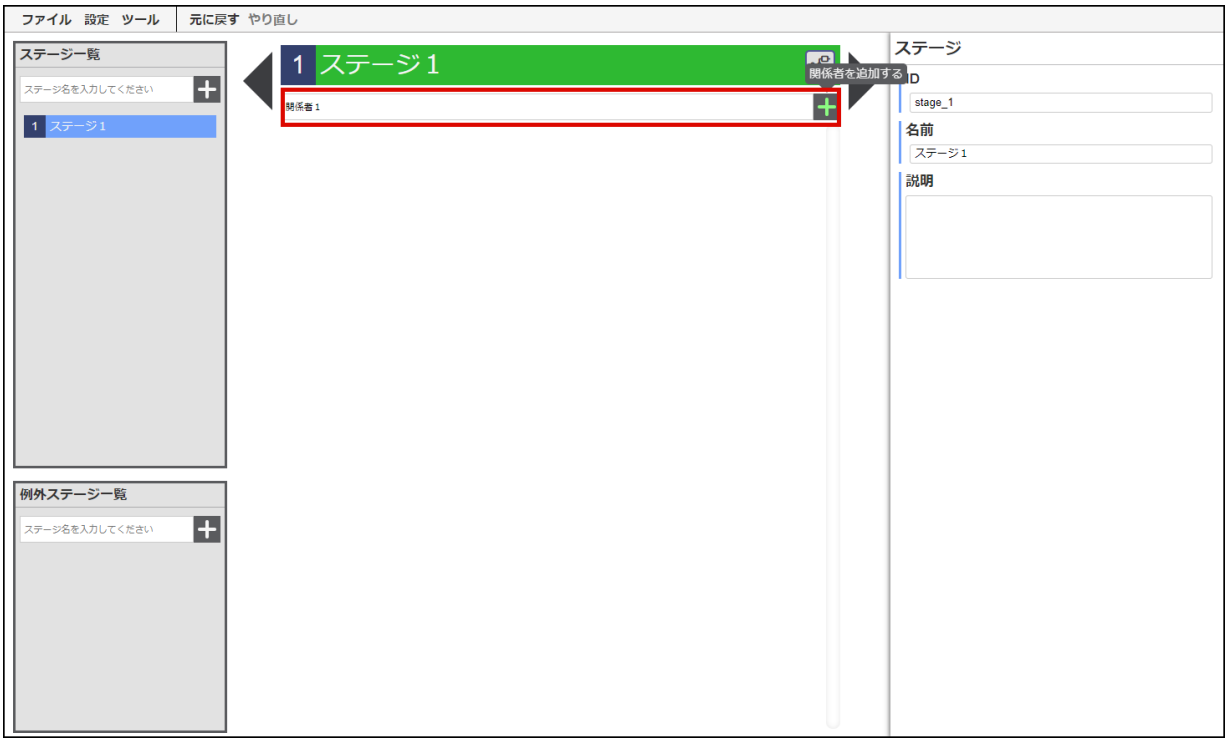
7. 「ステージ一覧」にて ステージ名を入力します。

8. をクリックし、「ステージ」を追加します。

図：「ステージ一覧」

9. 「ステージ詳細」にて関係者名を入力します。

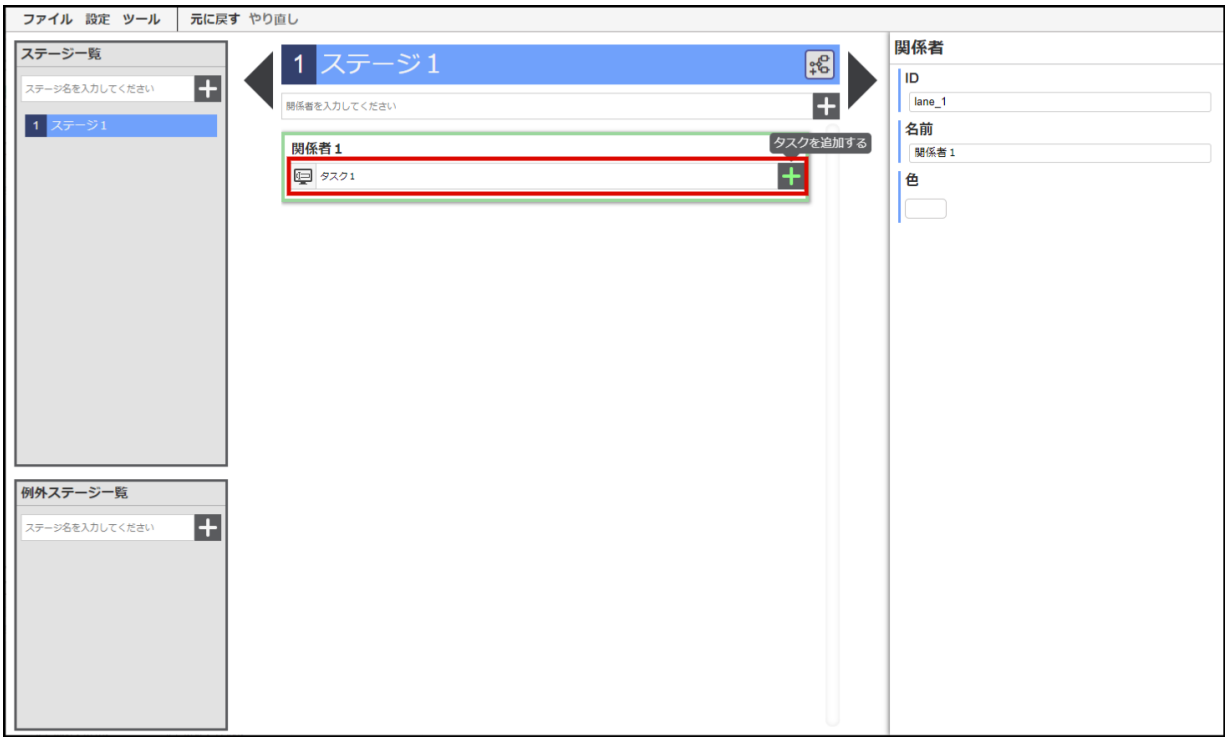
10. をクリックし、「関係者」を追加します。



図：「ステージ詳細」

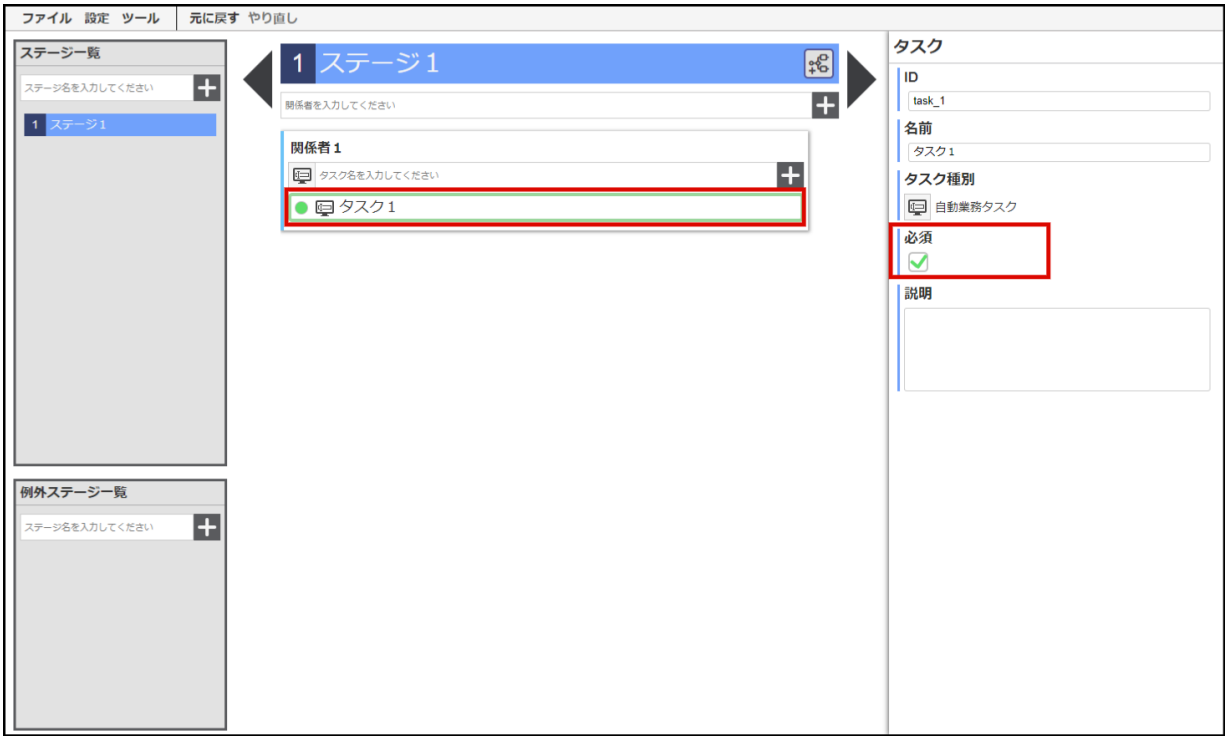
11. 「ステージ詳細」にてタスク名を入力します。

12. をクリックし、「タスク」を追加します。



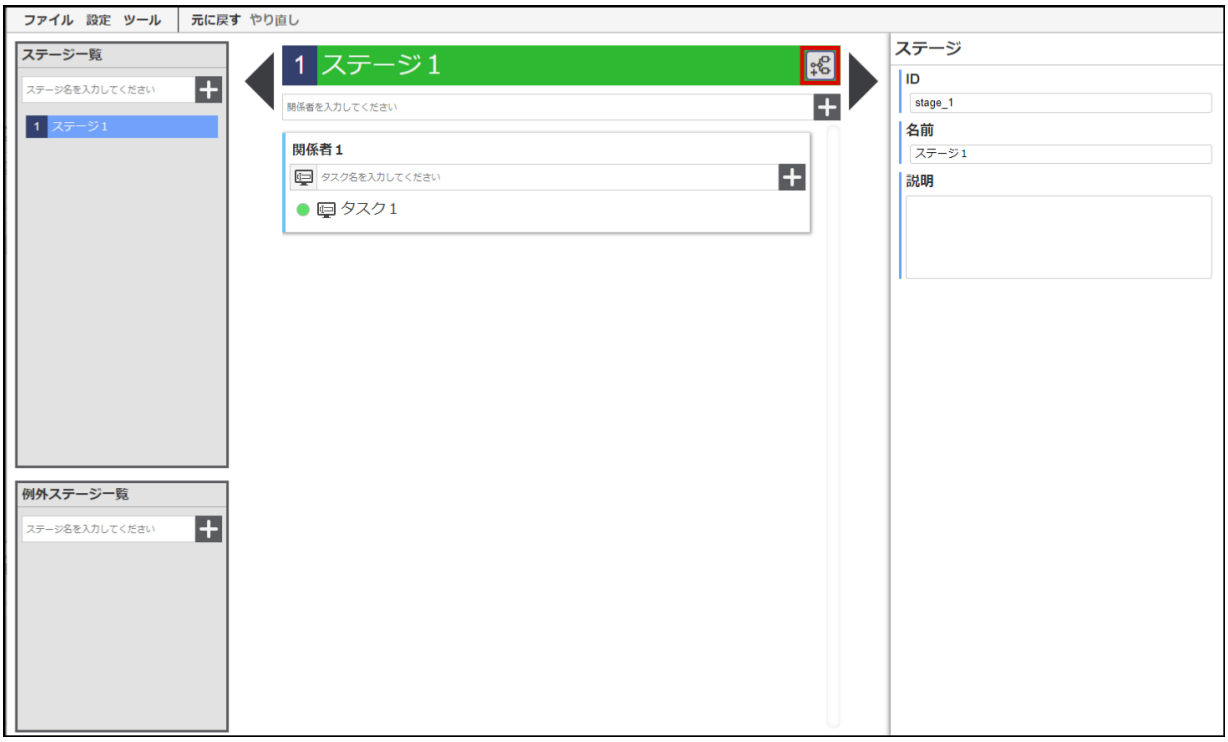
図：「ステージ詳細」

13. 追加された「タスク」を選択し、プロパティの「必須」を有効にします。



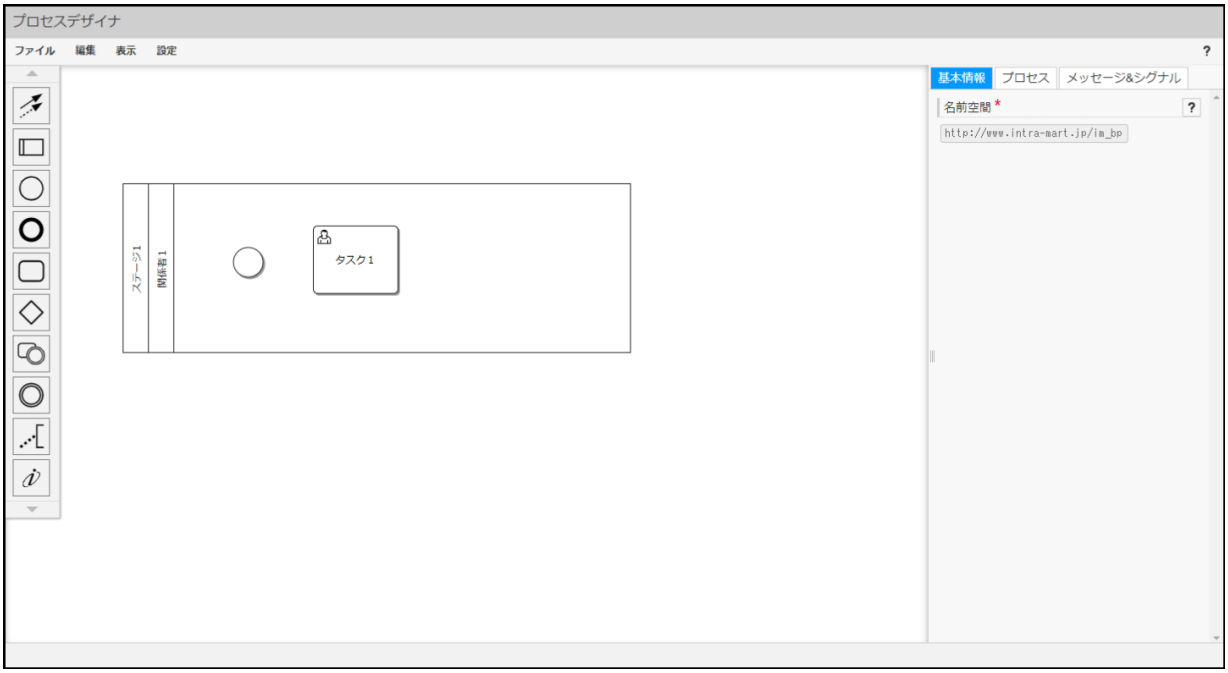
図：「プロパティエリア」

14. ステージの をクリックし、プロセスエディタを開きます。



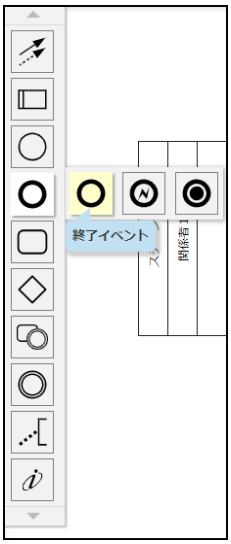
図：「ステージ詳細」

15. プロセスエディタが開かれました。



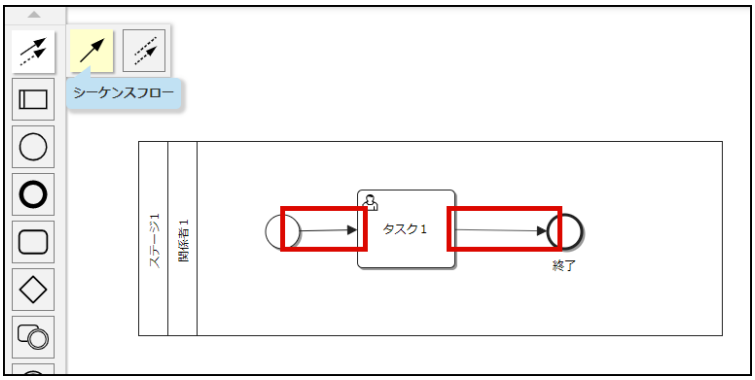
図：「プロセスエディタ」

16. 「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」

17. 「シーケンスフロー」で各エレメントを接続します。

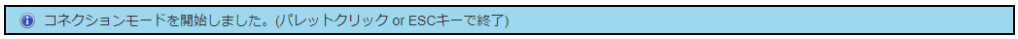


図：「パレット」 - 「コネクション」 - 「シーケンスフロー」

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。

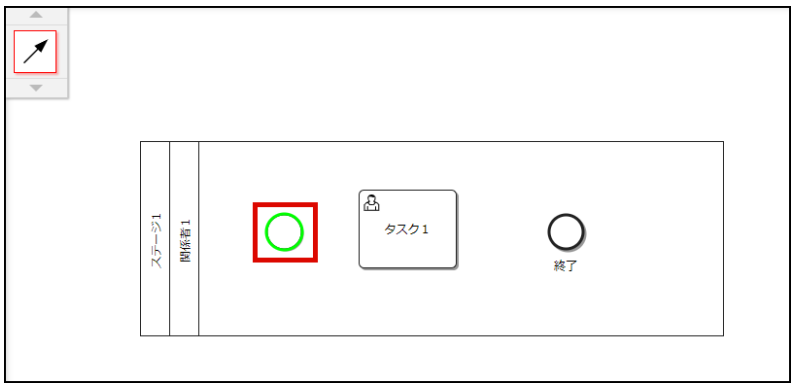
   コネクションモードが開始すると、フッタにメッセージが表示されます。



   図：「フッタ」

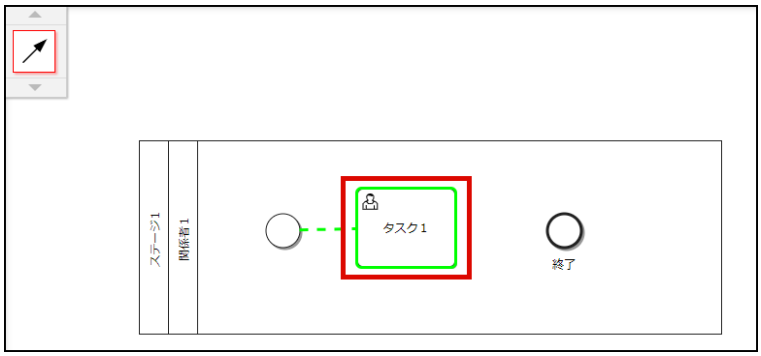
3. 「キャンバス」に配置された「開始イベント」をクリック、または、ドラッグして接続を開始します。

   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。

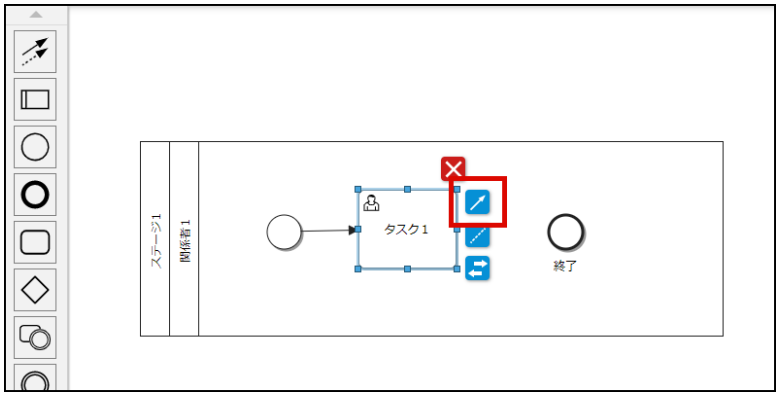


図：接続する

5. キーボードの「ESC」キーを押すか、「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。

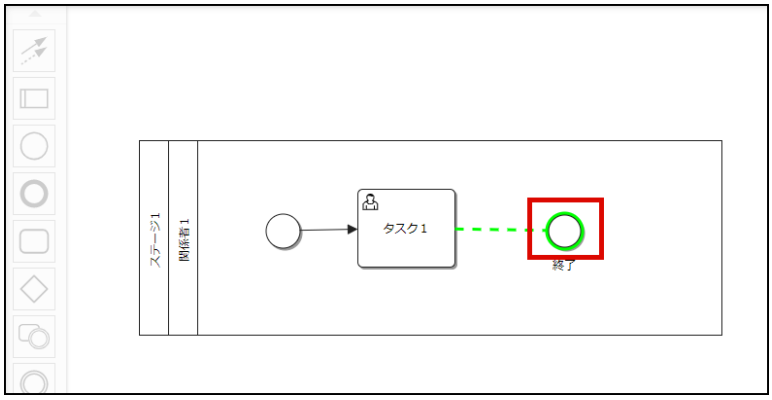
6. キャンバス上の「ユーザタスク」を選択します。

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。



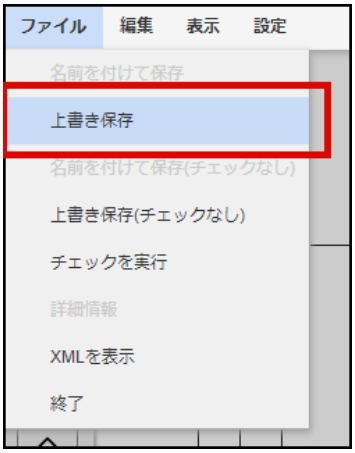
図：接続を開始する

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

18. 「ヘッダメニュー」から、「ファイル」 - 「上書き保存」をクリックします。



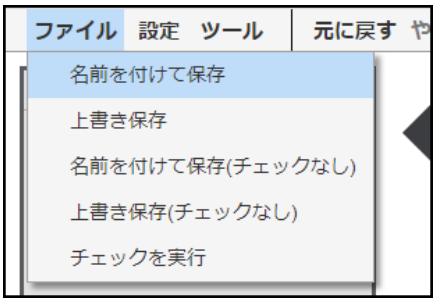
図：「ヘッダメニュー」 - 「ファイル」 - 「上書き保存」

> **コラム**
>
> 「上書き保存」と「上書き保存（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。
> 「上書き保存」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。
> 「上書き保存（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。

19. 上記「ステージ」の作成からプロセス定義の作成までの手順を再度繰り返します。

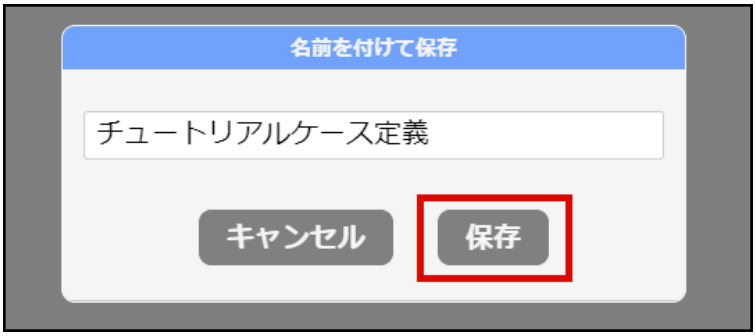
図：「ケースエディタ」

20. 「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」

21. 「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」

22. 保存されました。

図：ケース定義保存

ケース定義を IM-BPM Runtimeへデプロイする

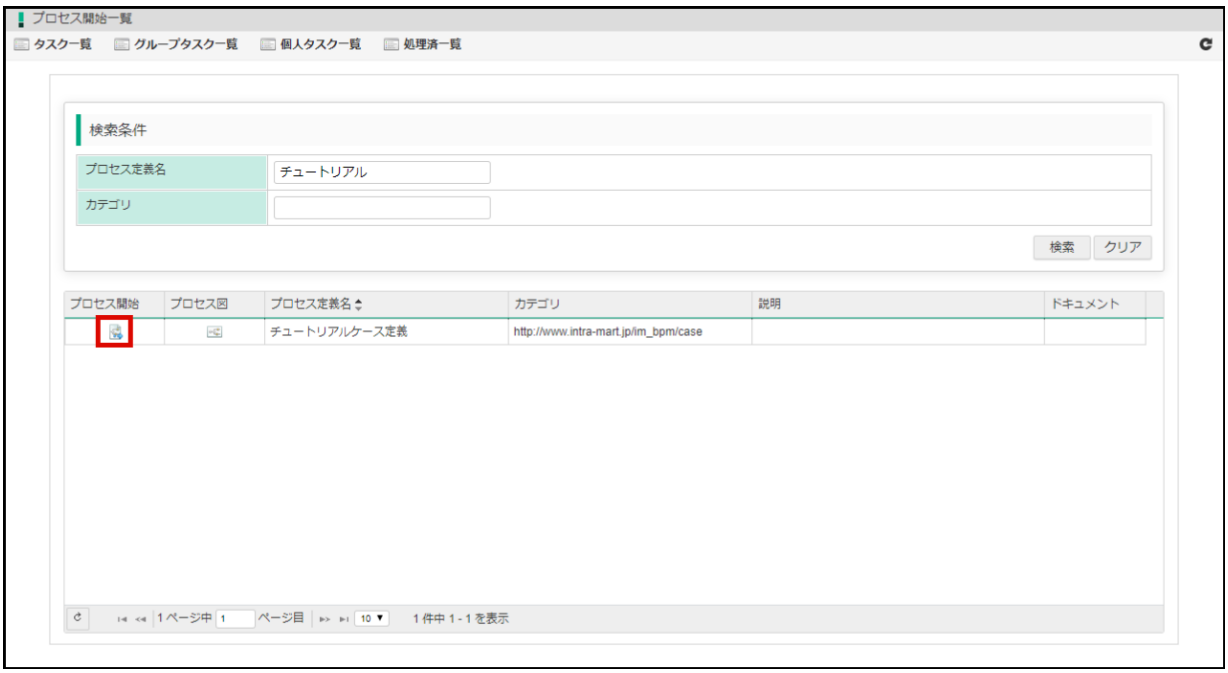
ケース定義を IM-BPM Runtimeへデプロイします。

手順はプロセス定義と同様なので「*プロセス定義を IM-BPM Runtimeへデプロイする*」を参照してください。

ケースを開始する

IM-BPM Runtimeにデプロイしたケー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. デプロイしたケース定義の をクリックします。



図：プロセス開始一覧

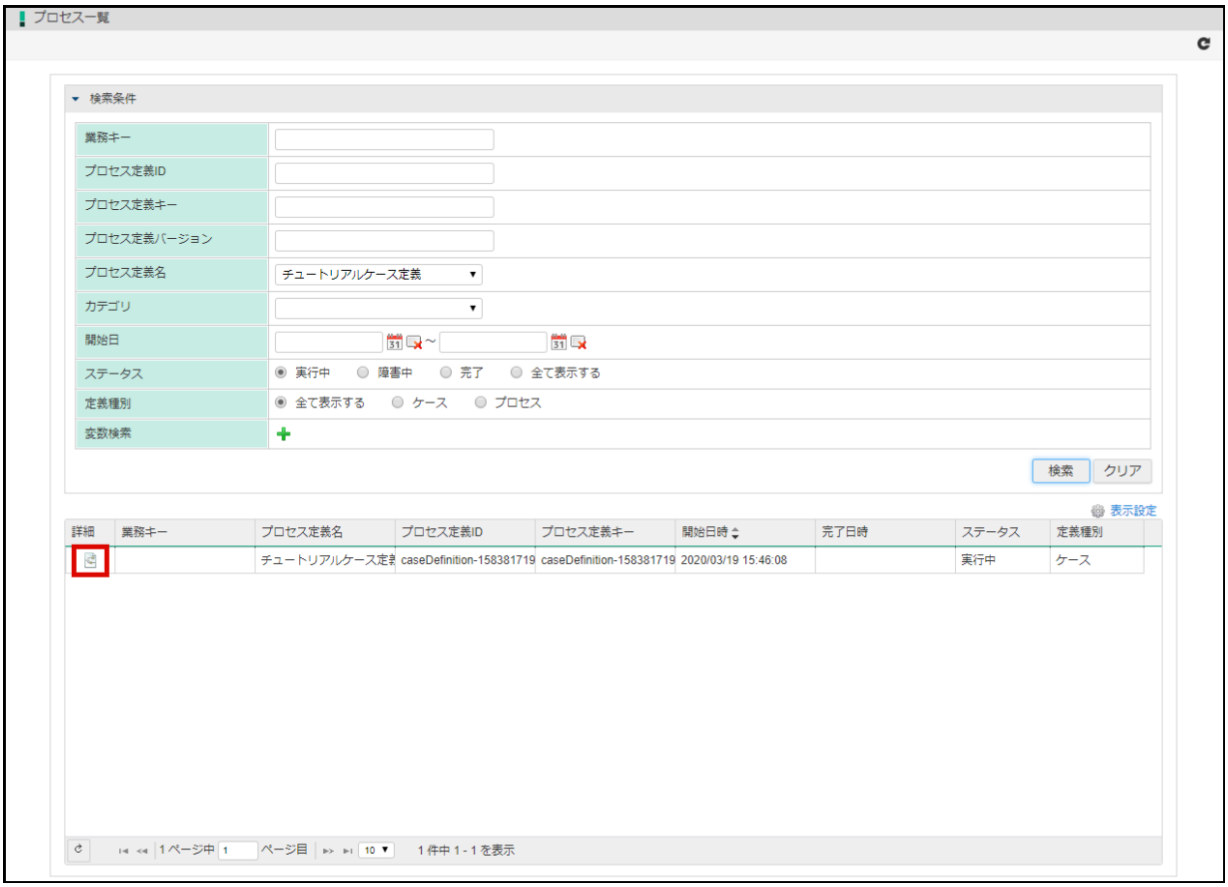
3. ケースが開始されました。

図：プロセス開始一覧

ケースインスタンスの詳細を確認する

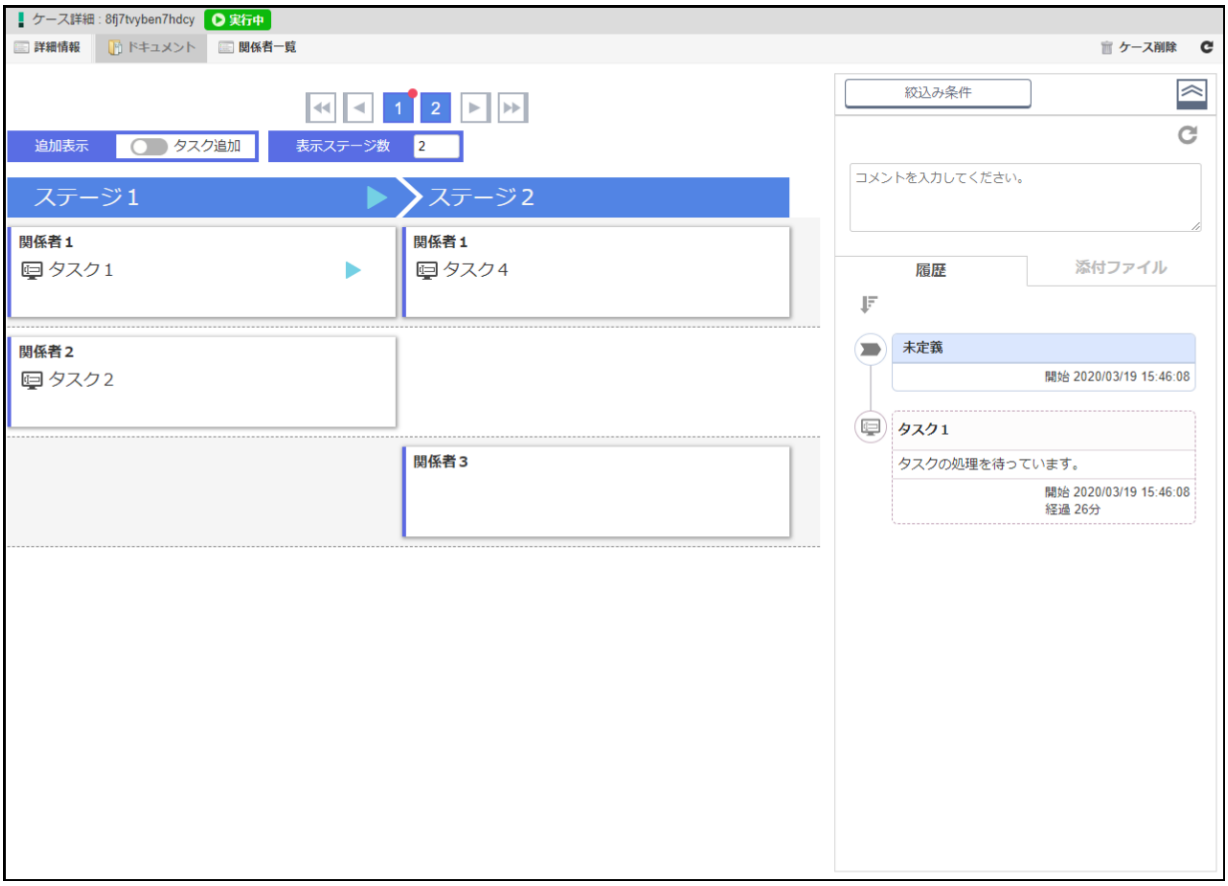
IM-BPM Runtimeで開始したケースのインスタンスの詳細を確認します。

1. 「サイトマップ」→「BPM」→「プロセス一覧」をクリックします。
2. 開始したケースの をクリックします。



図：プロセス一覧

3. ケースのインスタンスの詳細を確認します。



図：ケース詳細

> **コラム**
>
> ケース詳細
>
> 画面機能の詳細は「IM-BPM ユーザ操作ガイド」-「ケースインスタンスを確認する」を参照してください。

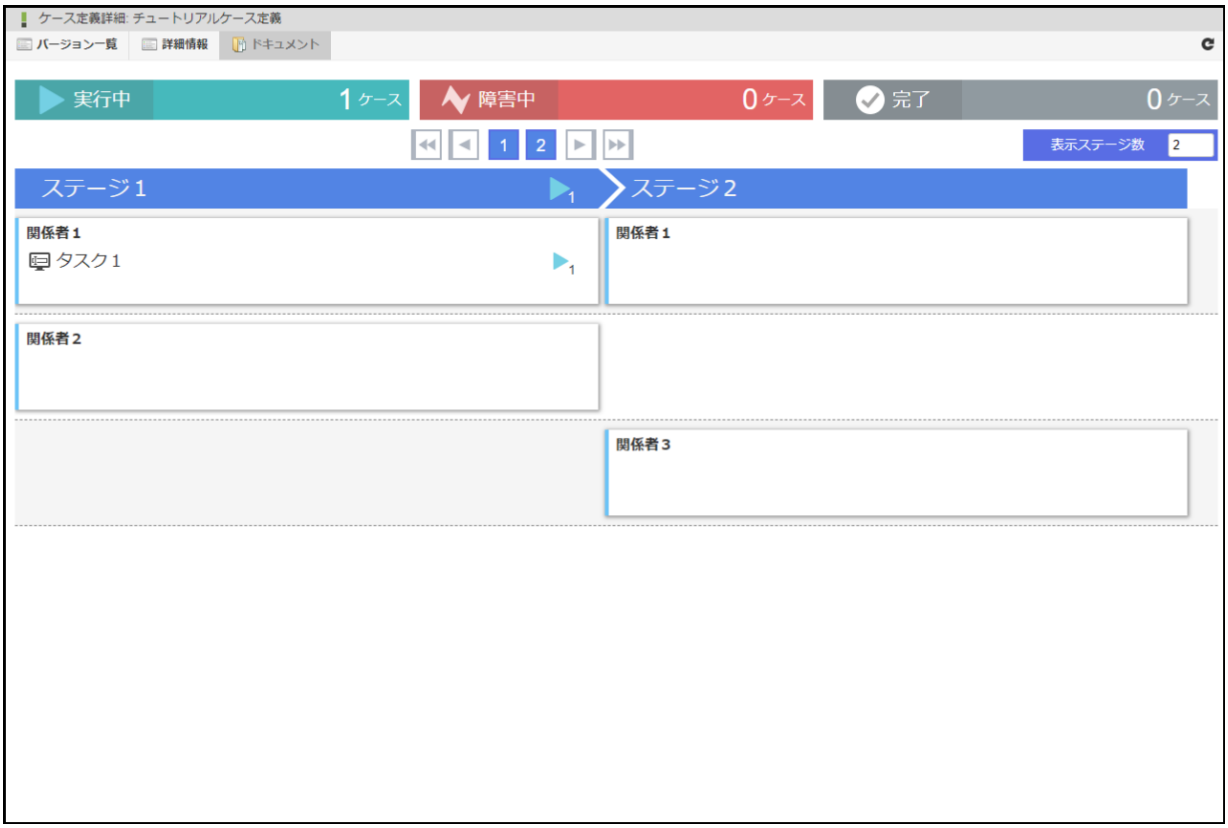
ケース定義の詳細を確認する

IM-BPM Runtimeにデプロイしたケース定義の詳細を確認します。

1. 「サイトマップ」→「BPM」→「プロセス定義一覧」をクリックします。
2. デプロイしたケースの をクリックします。

図：プロセス定義一覧

3. ケース定義の詳細を確認します。



図：ケース定義詳細

> **コラム**
>
> ケース定義詳細
>
> 画面機能の詳細は「IM-BPM ユーザ操作ガイド」-「ケース定義を確認する」を参照してください。

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
- オプショナルタスク
- アドホックタスク
- 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
- 受信タスク
- 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

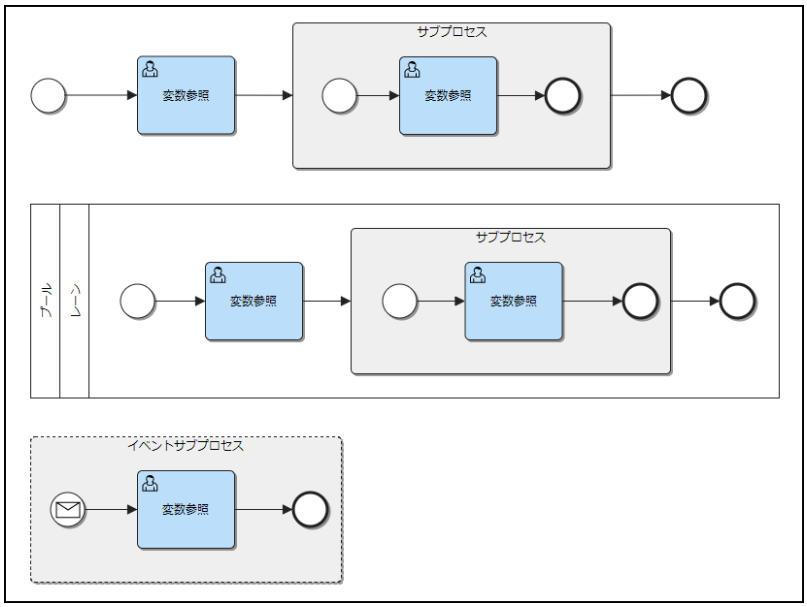
「データプロパティ」で定義した変数をエレメントから参照／更新できます。

> **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 操作ガイド」-「プロセス定義のアップロード」を参照してください。

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

エレメントの配置

配置イメージを参考に、各エレメントを配置してシーケンスフローで接続します。



図：配置イメージ

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

プロセスにデータプロパティを設定する

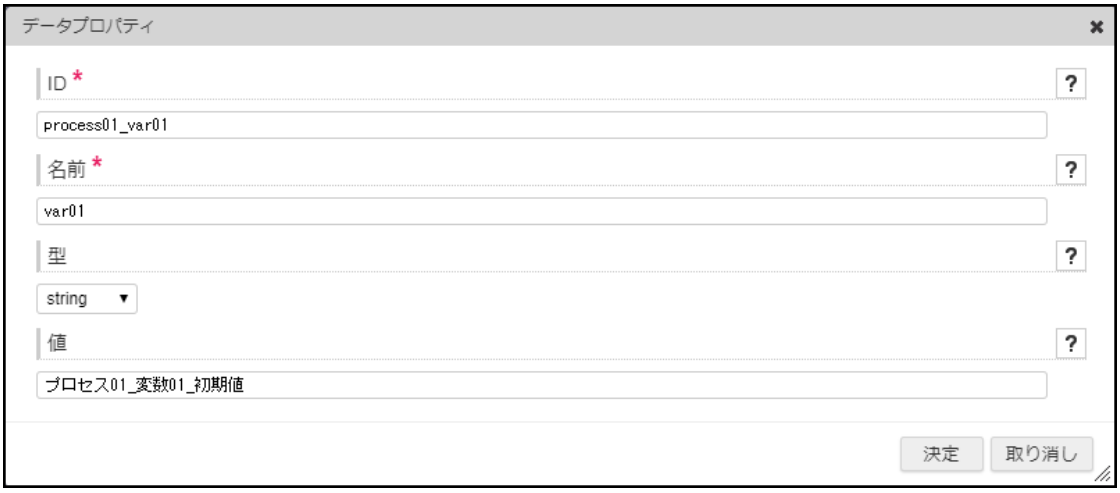
キャンバスに直接エレメントを配置した場合、キャンバスそのものが一つのプロセス定義として扱われます。
これを「プロセス」と呼びます。
この「プロセス」にデータプロパティを定義します。

1. キャンバス内のエレメントが配置されていない場所をクリックするか、「ヘッダメニュー > 編集 > 選択を解除」を実行することで、プロパティエリアに「プロセス」のプロパティが表示されます。

2. 「データオブジェクト」タブを開きます。

3. 「追加」をクリックし、ダイアログを開きます。

4. 以下のようにプロパティを入力します。



図：プロセス：プロパティ - データオブジェクト

項目名	入力値	説明
ID	process01_var01	このファイル内で一意となるID値を入力します。 データプロパティ内だけでなく、各エレメントに設定した「基本情報」タブの「ID」と重複することもできません。 「ID」の値は、システムが変数を一意に特定するためのものであり、変数名ではありません。
名前	var01	変数名を入力します。 同一のデータプロパティ内で同じ名前の変数を複数個定義しても、一つの変数として扱われます。
型	string	以下の型を選択できます。

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

値	プロセス01_変数01_初期値	選択した「型」に合わせて、変数の初期値を入力します。 「型」と「値」の入力値に矛盾があった場合、ファイルの保存時にエラーが発生します。

5. 「決定」をクリックして設定値を表に反映します。

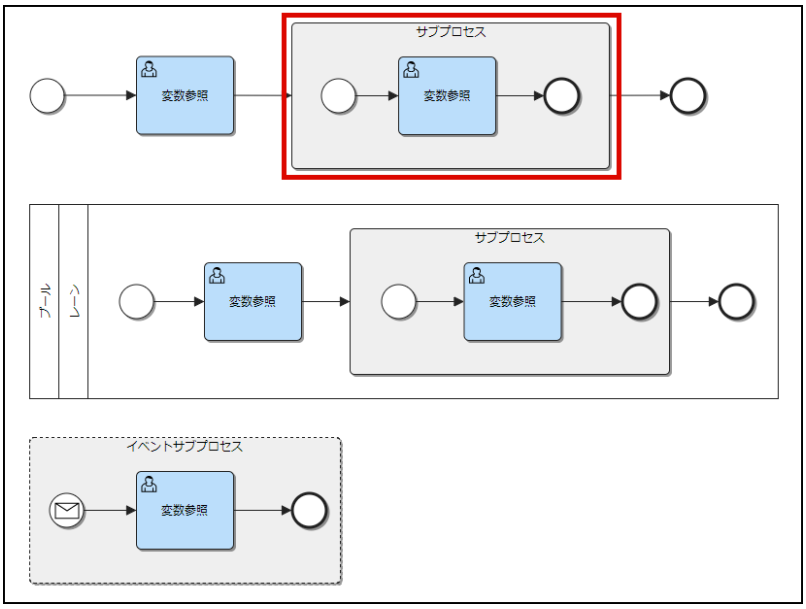
6. 同様の手順で、以下の変数も定義します。

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

サブプロセスにデータプロパティを設定する

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

1. 図の赤枠で示した「サブプロセス」を選択します。



図：サブプロセス

2. プロパティエリアで「データオブジェクト」タブを開きます。
3. プロセスのデータプロパティを設定した手順と同様の手順で、以下のデータプロパティを定義します。

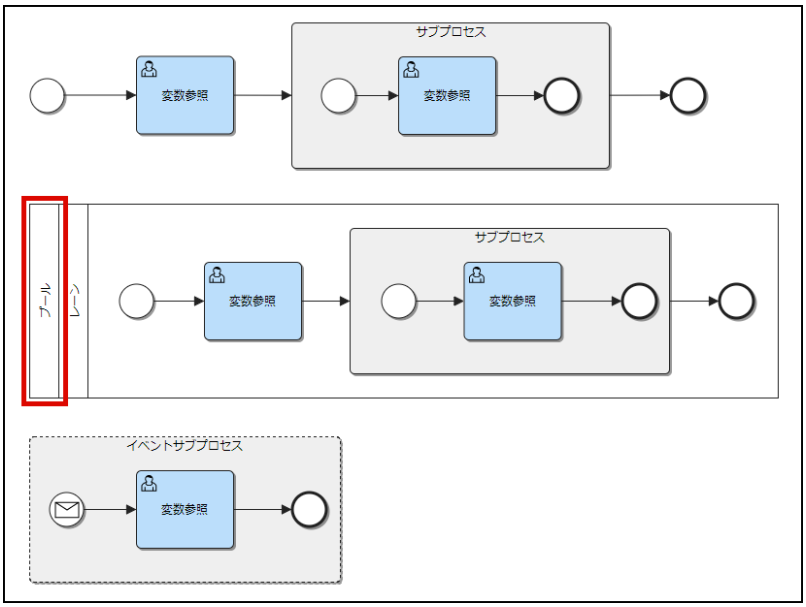
図：サブプロセス：プロパティ - データオブジェクト

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

プールにデータプロパティを定義する

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

1. 図の赤枠で示した「プール」を選択します。



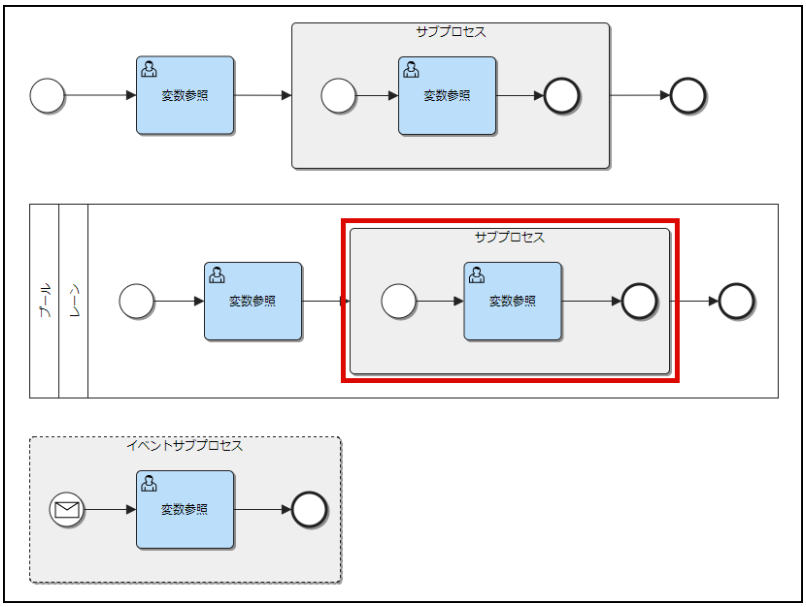
図：プール：プロパティ - データオブジェクト

2. 以下の通り、データプロパティを定義します。

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

プール内のサブプロセスにデータプロパティを定義する

1. 図の赤枠で示した「サブプロセス」を選択します。



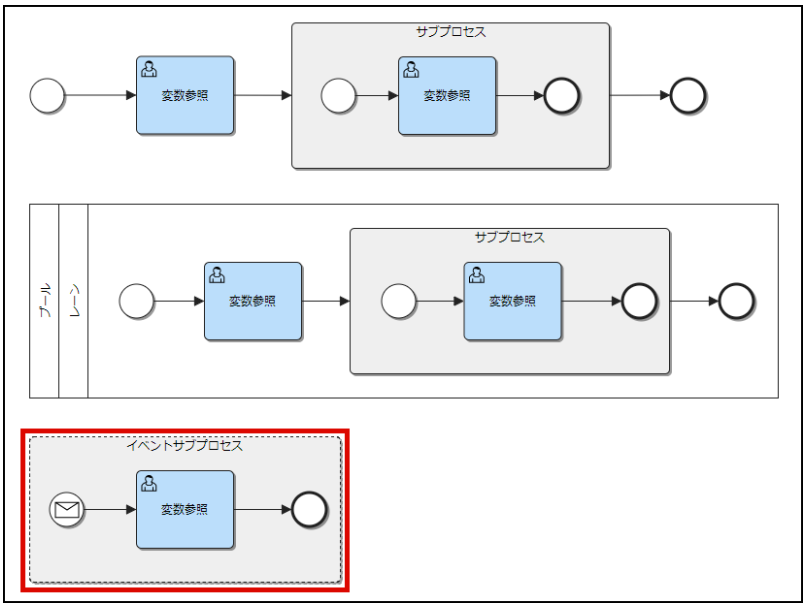
図：サブプロセス：プロパティ - データオブジェクト

2. 以下の通り、データプロパティを定義します。

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

イベントサブプロセスにデータプロパティを定義する

1. 図の赤枠で示した「イベントサブプロセス」を選択します。



図：イベントサブプロセス：プロパティ - データオブジェクト

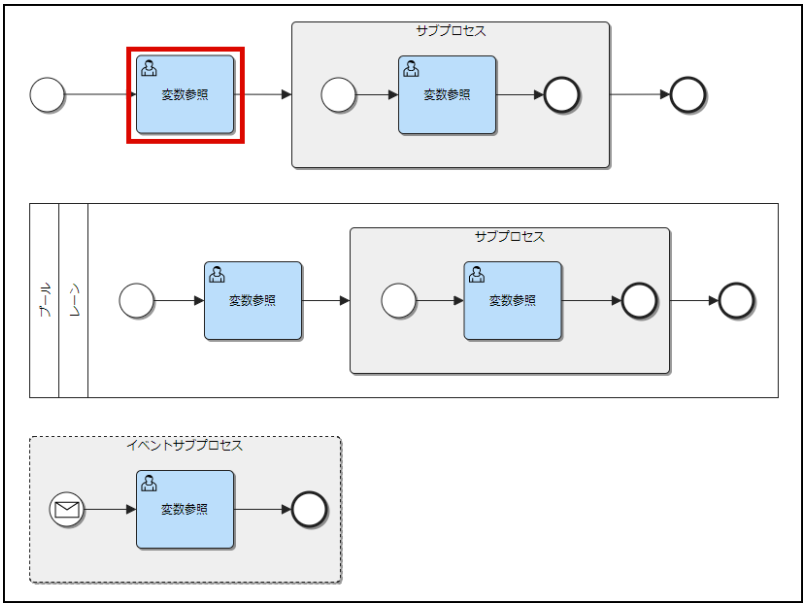
2. 以下の通り、データプロパティを定義します。

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

データプロパティの参照可能範囲を確認する

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

1. 図の赤枠で示した「ユーザタスク」を選択します。



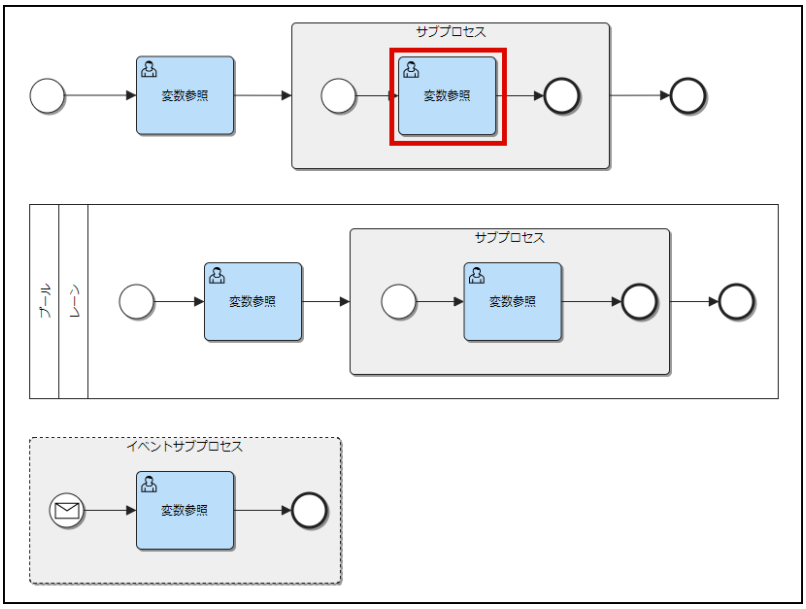
図：ユーザタスクの選択

2. プロパティの任意の項目で、 ${} をクリックします。
3. 開いたEL式エディタの右側にある「データプロパティツリー」を確認します。
4. プロセス「データプロパティを定義する_プロセス01」のデータプロパティ「var01 <string>」と「var02 <datetime>」が参照できることがわかります。

図：EL式エディタ - データプロパティツリー

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

5. 図の赤枠で示した「ユーザタスク」を選択します。



図：ユーザタスクの選択

6. EL式エディタを開き、「データプロパティツリー」を確認します。

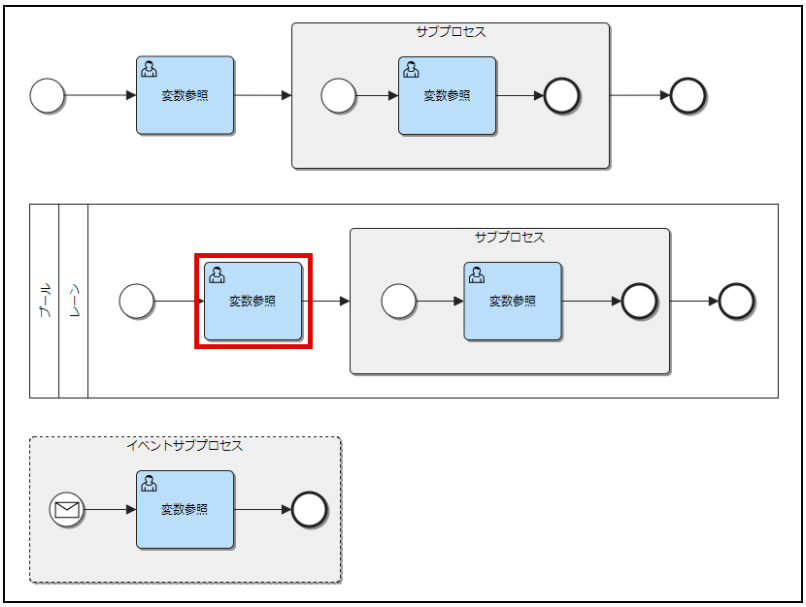
7. プロセス「データプロパティを定義する_プロセス01」のデータプロパティに加え、サブプロセスで定義したデータプロパティ「var01 <int>」と「var03 <boolean>」が参照できることがわかります。

図：EL式エディタ - データプロパティツリー

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

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

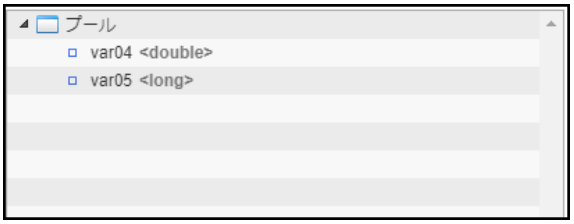
8. 図の赤枠で示した「ユーザタスク」を選択します。



図：ユーザタスクの選択

9. EL式エディタを開き、「データプロパティツリー」を確認します。

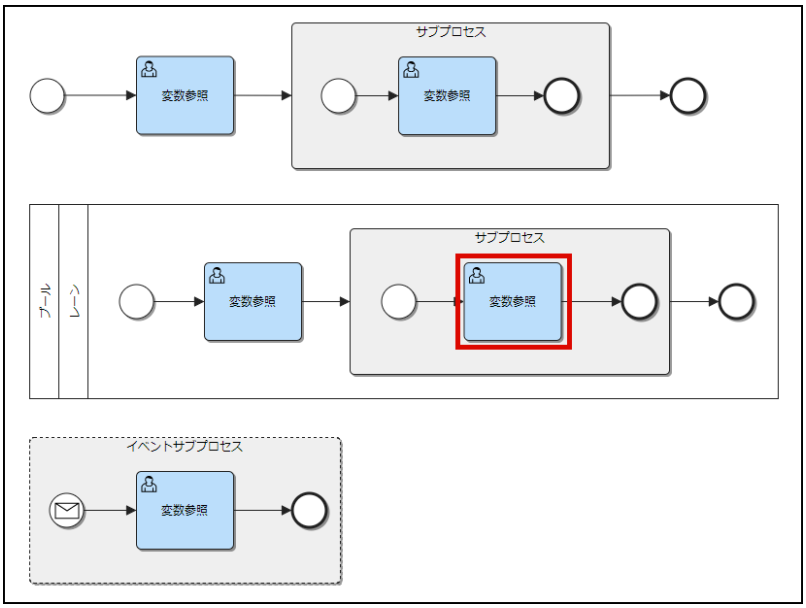
10. 「プール」のデータプロパティ「var04 <double>」と「var05 <long>」が参照できることがわかります。



図：EL式エディタ - データプロパティツリー

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

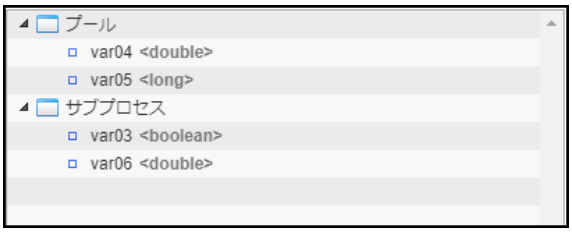
11. 図の赤枠で示した「ユーザタスク」を選択します。



図：ユーザタスクの選択

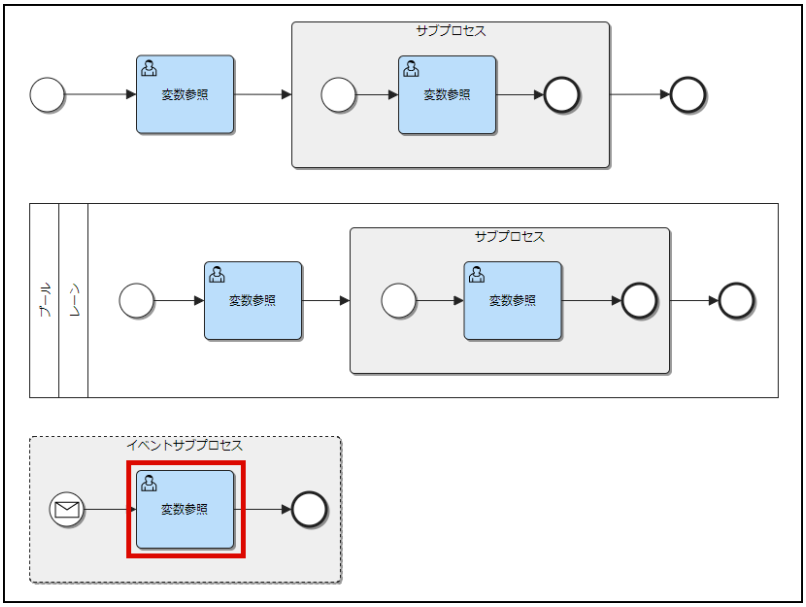
12. EL式エディタを開き、「データプロパティツリー」を確認します。

13. 「プール」のデータプロパティに加え、プール内サブプロセスのデータプロパティ「var03 <boolean>」と「var06 <double>」が参照できることがわかります。



図：EL式エディタ - データプロパティツリー

14. 図の赤枠で示した「ユーザタスク」を選択します。



図：ユーザタスクの選択

15. EL式エディタを開き、「データプロパティツリー」を確認します。

16. プロセス「データプロパティを定義する_プロセス01」のデータプロパティに加え、イベントサブプロセスで定義したデータプロパティ「var07 <string>」が参照できることがわかります。

図：EL式エディタ - データプロパティツリー

マルチインスタンス

マルチインスタンス を使用する

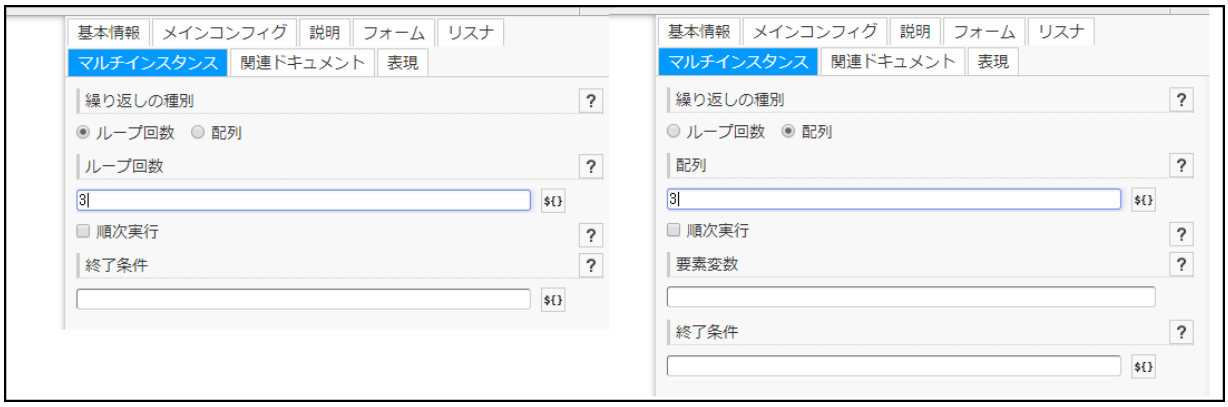
このチュートリアルでは、マルチインスタンスの定義を使用して複数のインスタンスを並列および、順次に処理する方法を解説します。
マルチインスタンス の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「マルチインスタンス」も合わせて参照してください。

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

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

プロパティを確認する

プロパティを確認します。



図：プロパティ

- 繰り返しの種別

  ループ回数か配列を選択します。

- ループ回数

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

- 配列

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

- 順次実行

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

- 要素変数

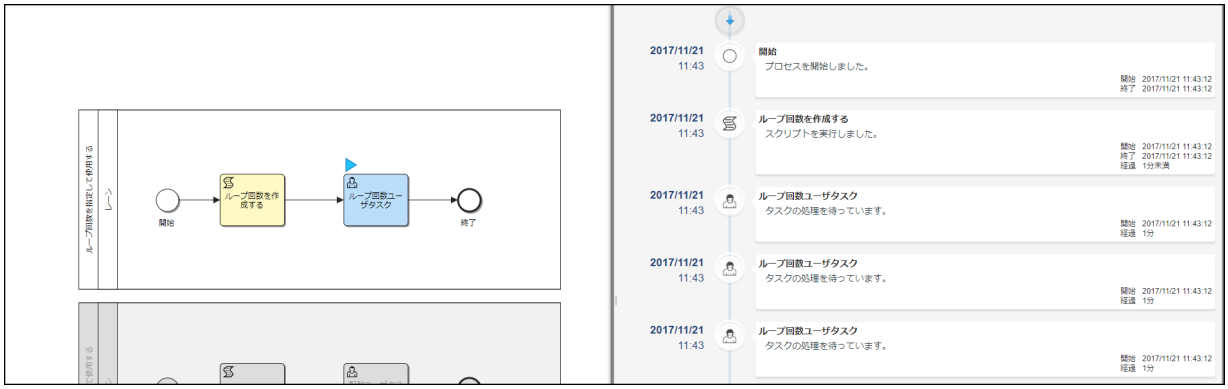
  繰り返しの種別が「配列」の場合、表示されます。
  「配列」の要素が「要素変数」に保存されます。

- 終了条件

  終了条件を設定します。

ループ回数を指定して使用する

「ループ回数」を指定して使用します。
※ 並列で実行します。



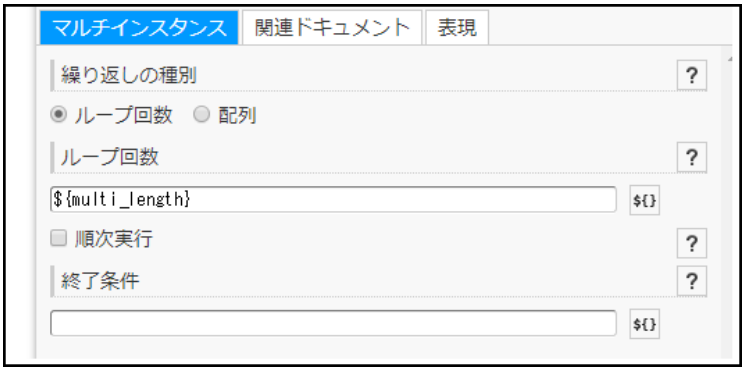
図：実行時のイメージ

1. 任意の方法で、multi_length の変数に回数を設定します。

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

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

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



図：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。

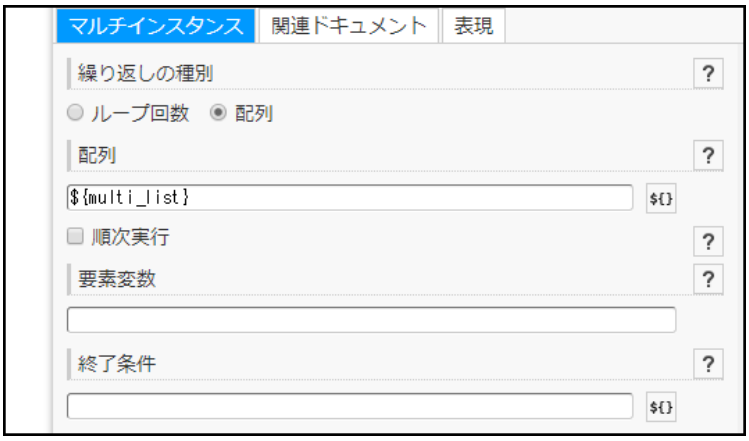
図：実行時のイメージ

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

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

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



図：プロパティの設定

並列、または、順次で実行する

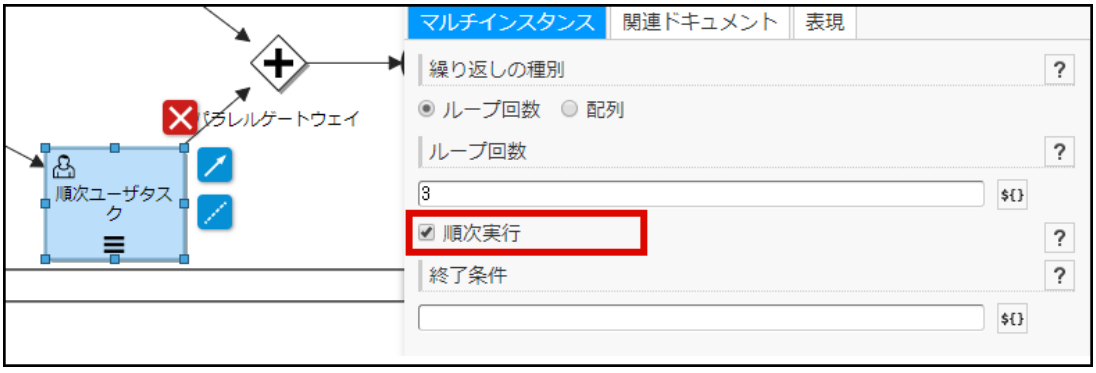
並列、または、順次で実行します。

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

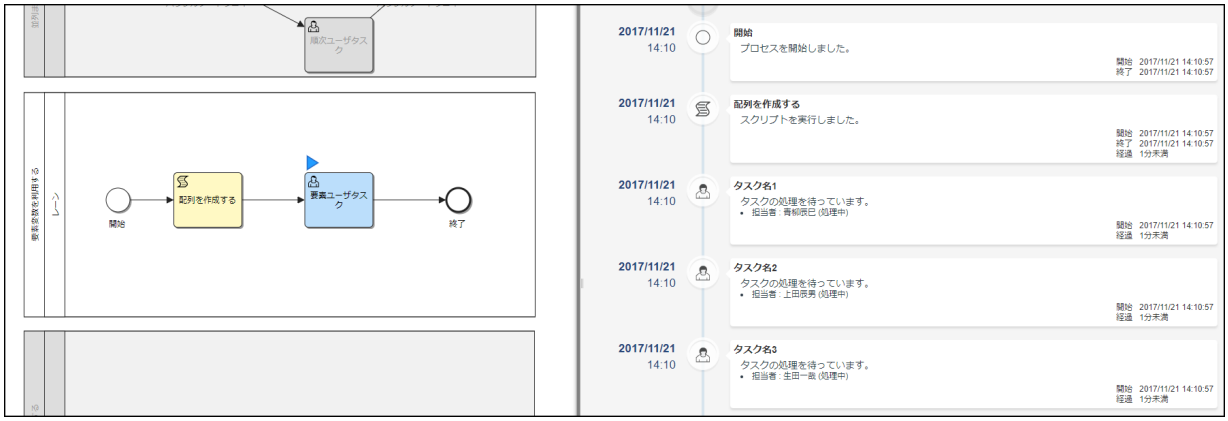
2. 順次で実行する場合、「順次実行」のチェックボックスを有効にします。



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。

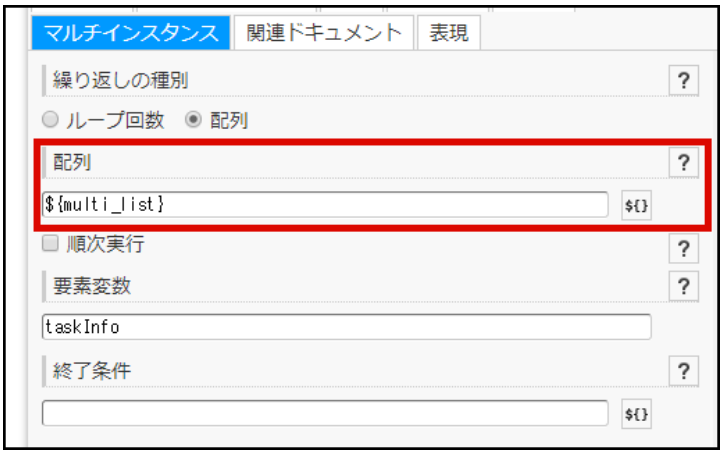


図：実行時のイメージ

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。

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

2. マルチインスタンスの「繰り返しの種別」を「配列」に設定します。（初期設定は「ループ回数」）
3. 「配列」にEL式を使用して、multi_list の変数を指定します。

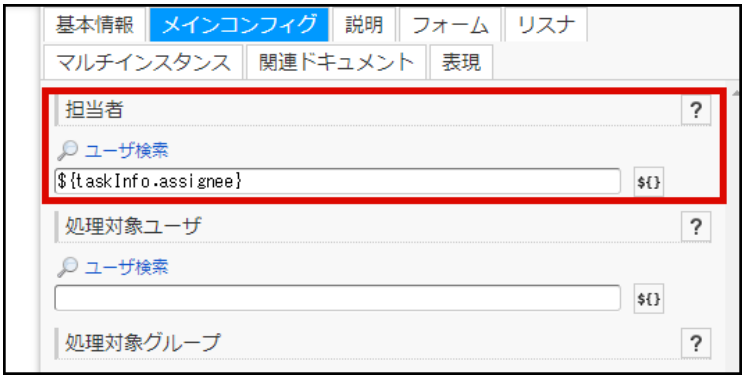


図：プロパティの設定 - マルチインスタンス

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

図：プロパティの設定 - 基本情報

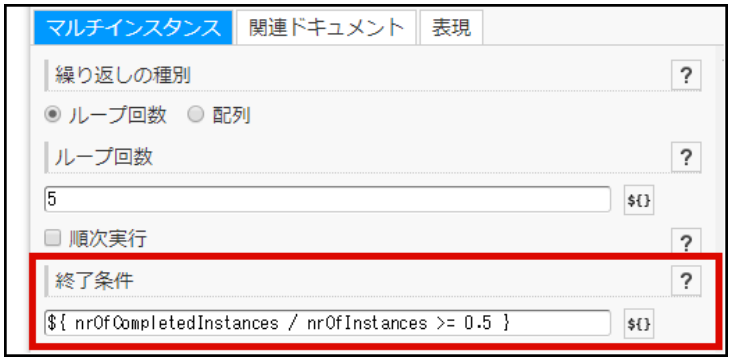
6. プロパティの「メインコンフィグ」タブから、「担当者」にEL式を使用して、taskInfo.assignee を指定します。



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遷移します。

リスナ

IM-LogicDesignerのリスナを利用する

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

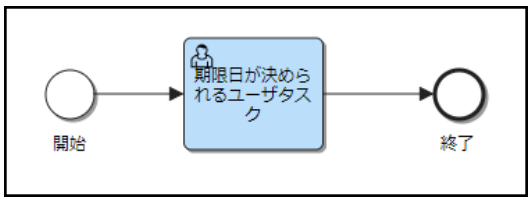
> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **listener_logic_designer.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

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

IM-LogicDesignerのリスナを利用したプロセス定義を作成する

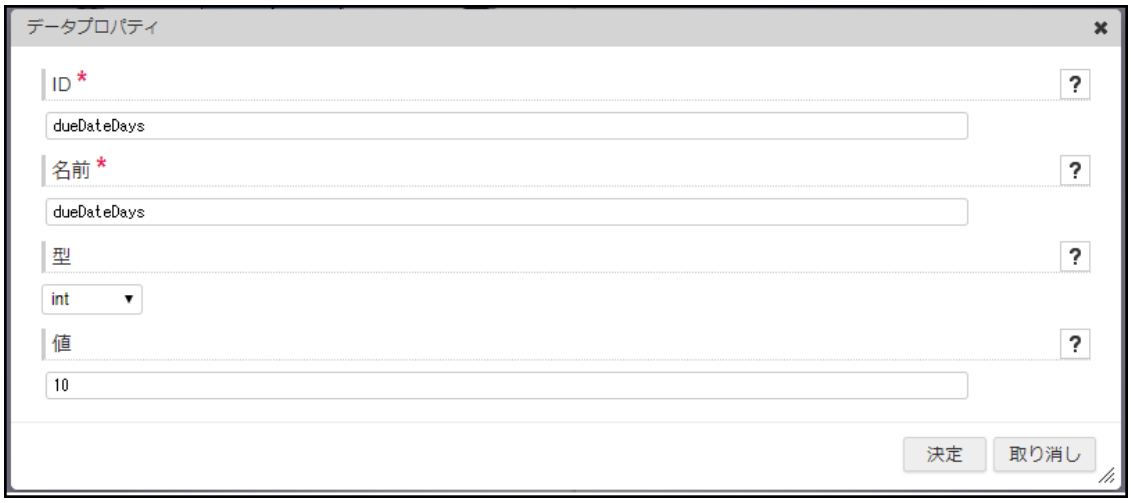
IM-LogicDesignerのリスナを利用して、基準日から数日後の営業日を取得し、ユーザタスクの期限日に設定します。
このチュートリアルでは、基準日を現在日に設定し、10日後の営業日を期限日に設定するようにします。
このチュートリアルを開始する前に以下リンクから「ロジックフロー」をダウンロードし、インポートしてください。
im_logicdesigner-data-listener.zip

> **コラム**
>
> 「ロジックフロー」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。
   キャンバ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

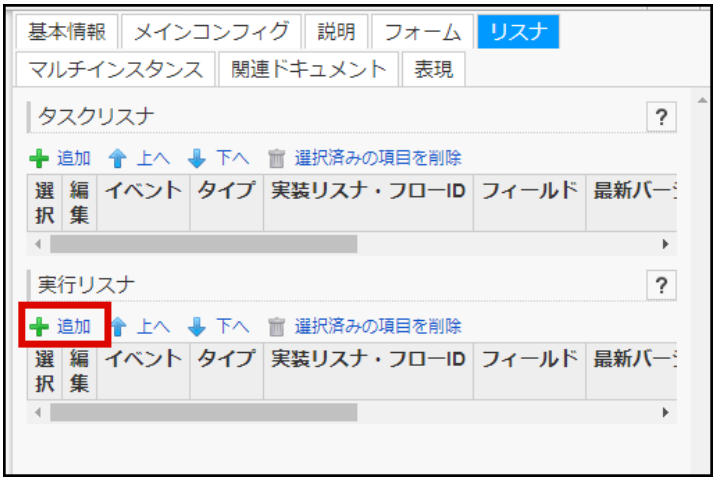


図：データプロパティ

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。

図：「実行リスナ」

   2. フローIDを設定します。

図：「実行リスナ」

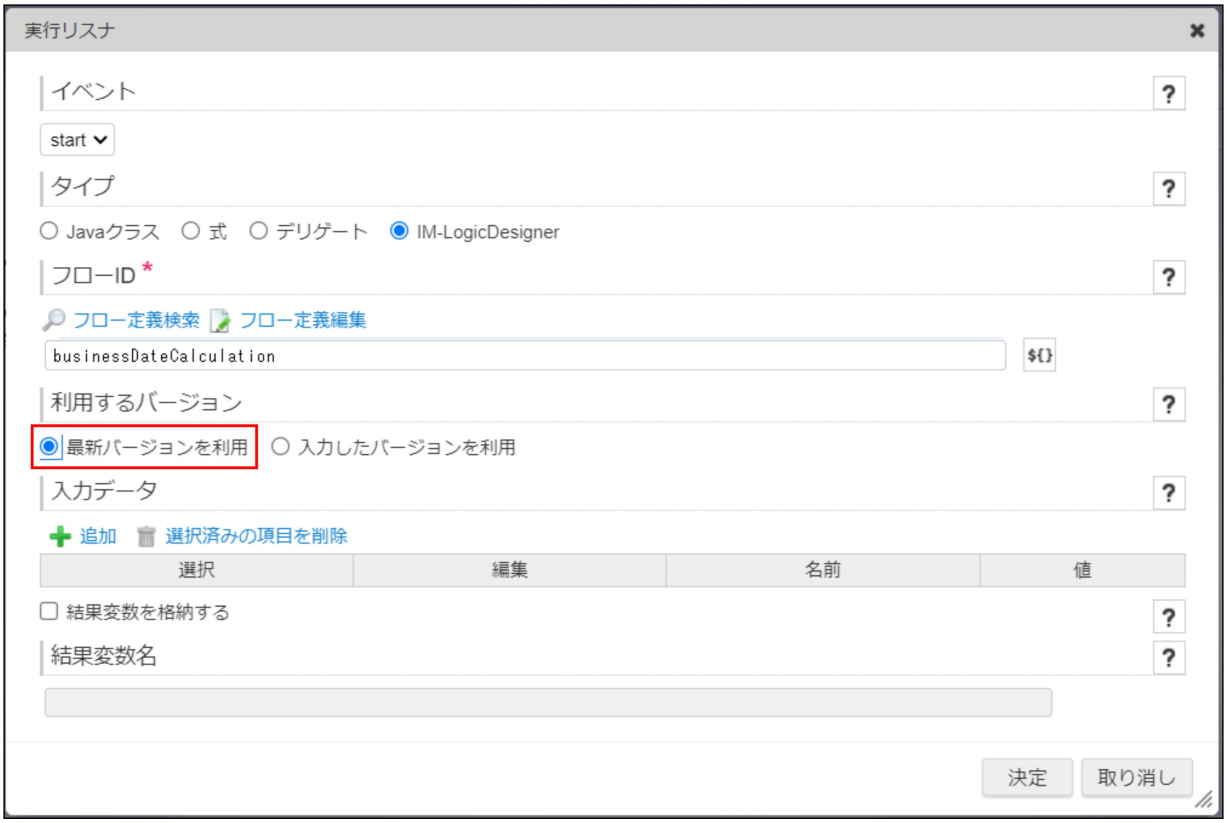
フローIDを設定するには、以下3つの方法があります。

- **「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。

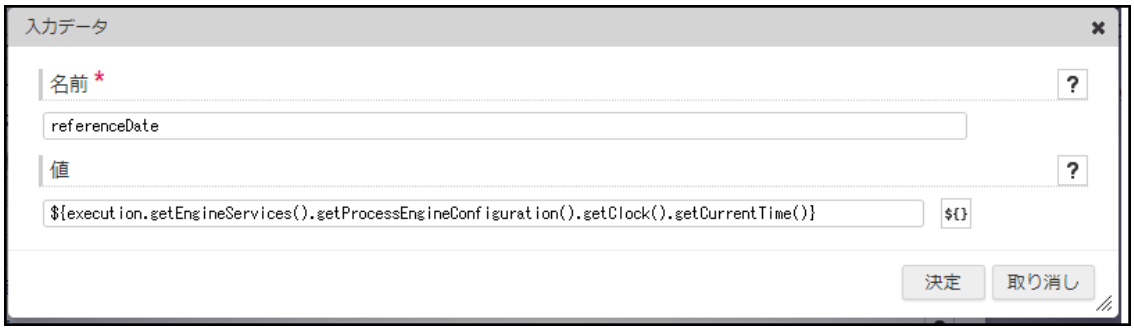
> **コラム**
>
> フローIDを直接指定している場合は、「フロー定義編集」より指定のフローを編集できます。

3. 最新バージョンを利用するように設定します。

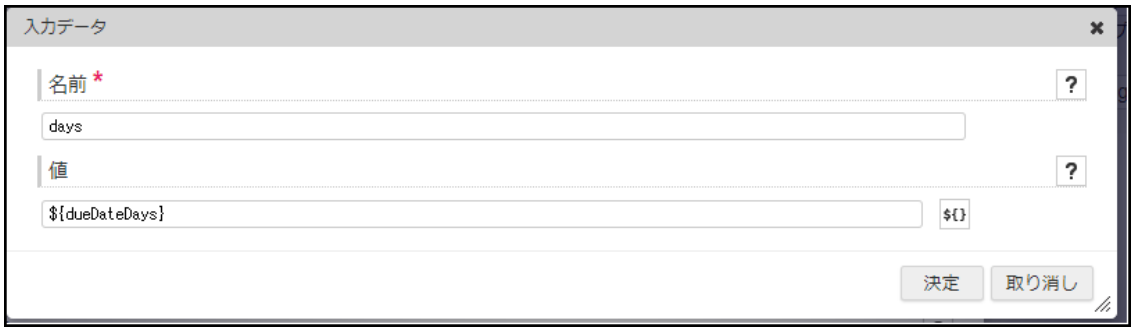


図：「実行リスナ」

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}



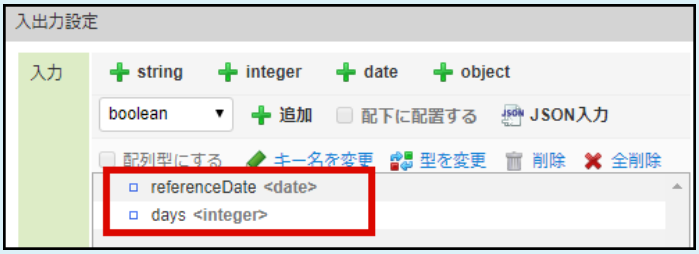
図：基準日の設定



図：日数の設定

コラム

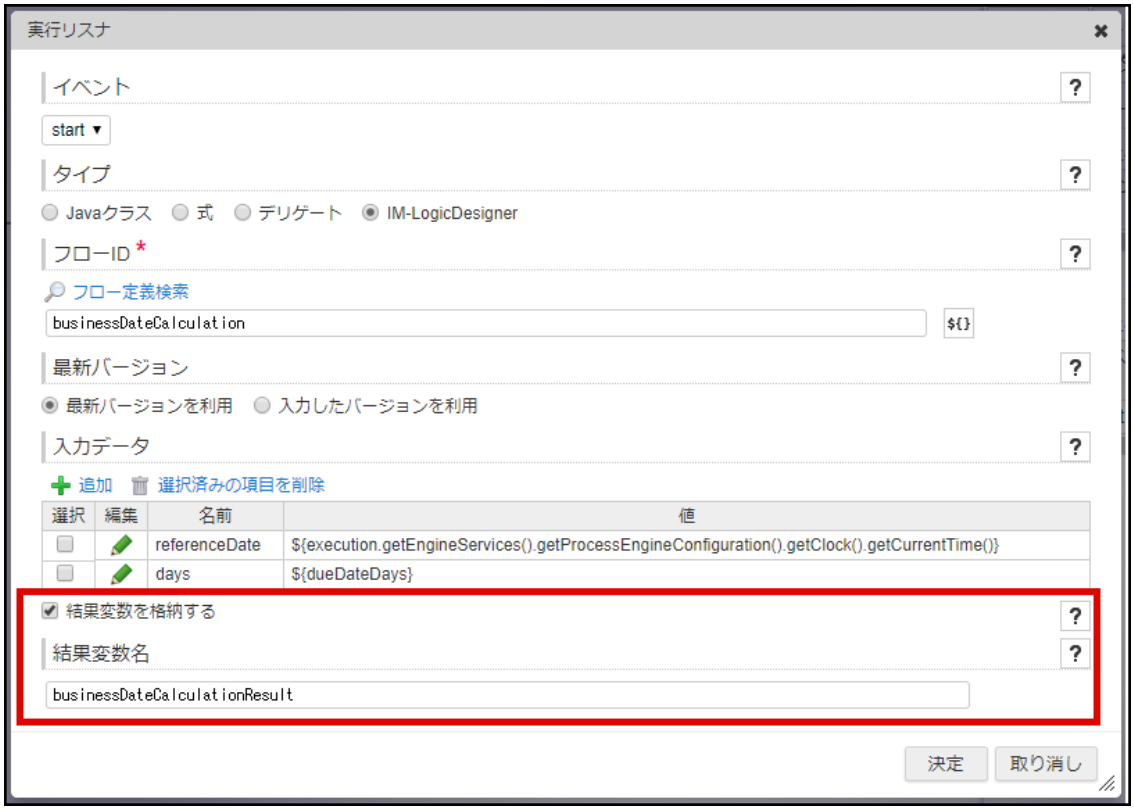
上記で設定したEL式は、ロジックフローの入力設定に対応しています。



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。



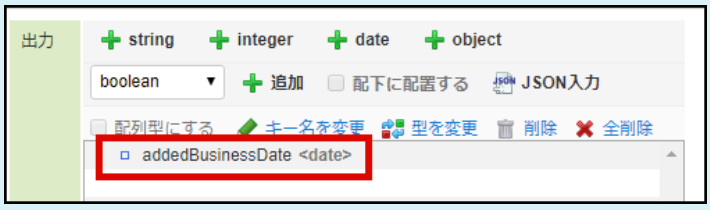
図：「実行リスナ」

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。

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。



図：ロジックフロー側の対応する出力設定

実行結果を確認する

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

1. プロセスを開始します。
2. タスク一覧を確認します

図：タスク一覧

タスクリスナを利用してタスクの処理依頼メールを送信する

このチュートリアルでは、タスクリスナを利用する方法を解説します。
「タスクリスナ」を設定することで、任意の処理をタスクのイベント発生時に実行できます。

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

タスクリスナを使って処理依頼メールを送信する際に、「IM-LogicDesigner」のロジックフローと「IM-FormaDesigner」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材をインポートしてください。

- ロジックフロー

  im_logicdesigner-data-listener_send_mail.zip

- Formaアプリケーション

  select_usercd.zip
  conduct_button.zip

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

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

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

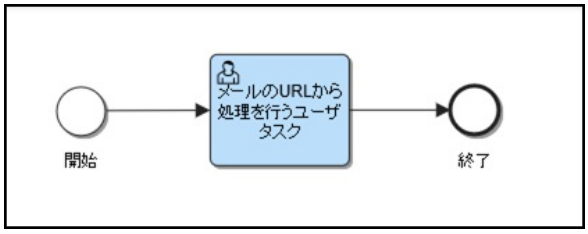
> **コラム**
>
> 各種設定・詳細については、以下のリンクを参照してください。
>
> - ロール設定：「IM-共通マスタ 管理者操作ガイド」 - 「ユーザ」
> - メールサーバの設定：「設定ファイルリファレンス」 - 「メール設定」

> **コラム**
>
> 各種インポート方法については、以下のリンクを参照してください。
>
> - ロジックフロー: 「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Forma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

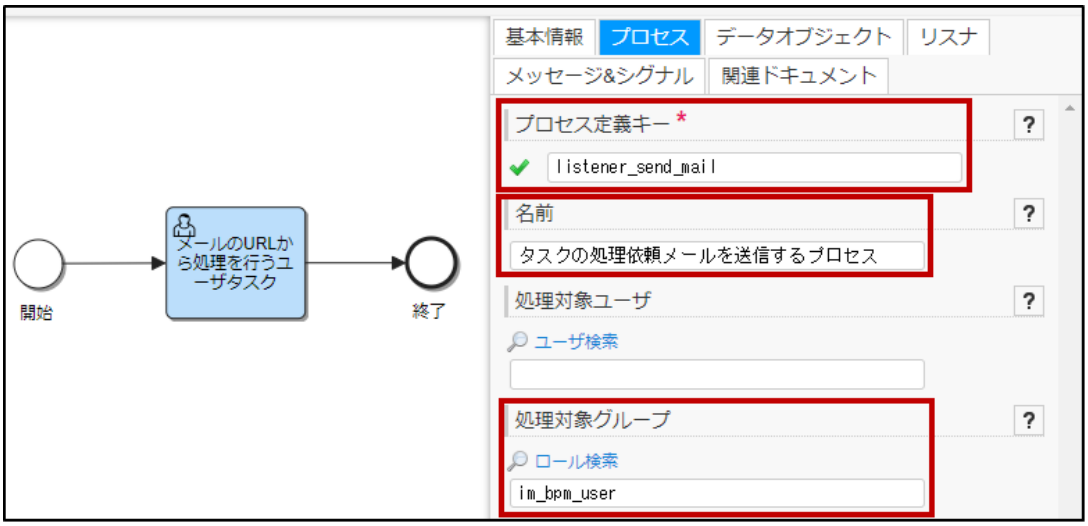
プロセス定義を作成する

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。



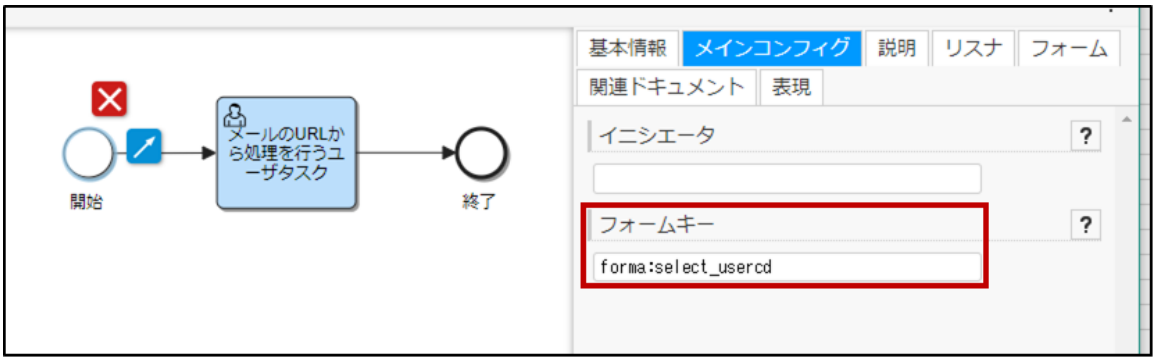
図：完成イメージ

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`



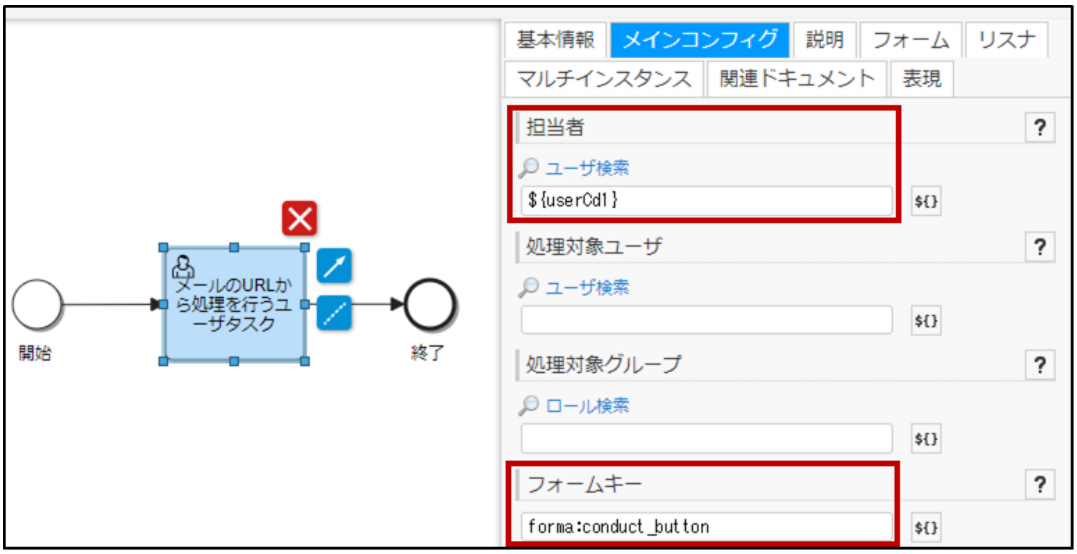
図：プロパティ全体 - 「プロパティ」 - 「プロセス」

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`

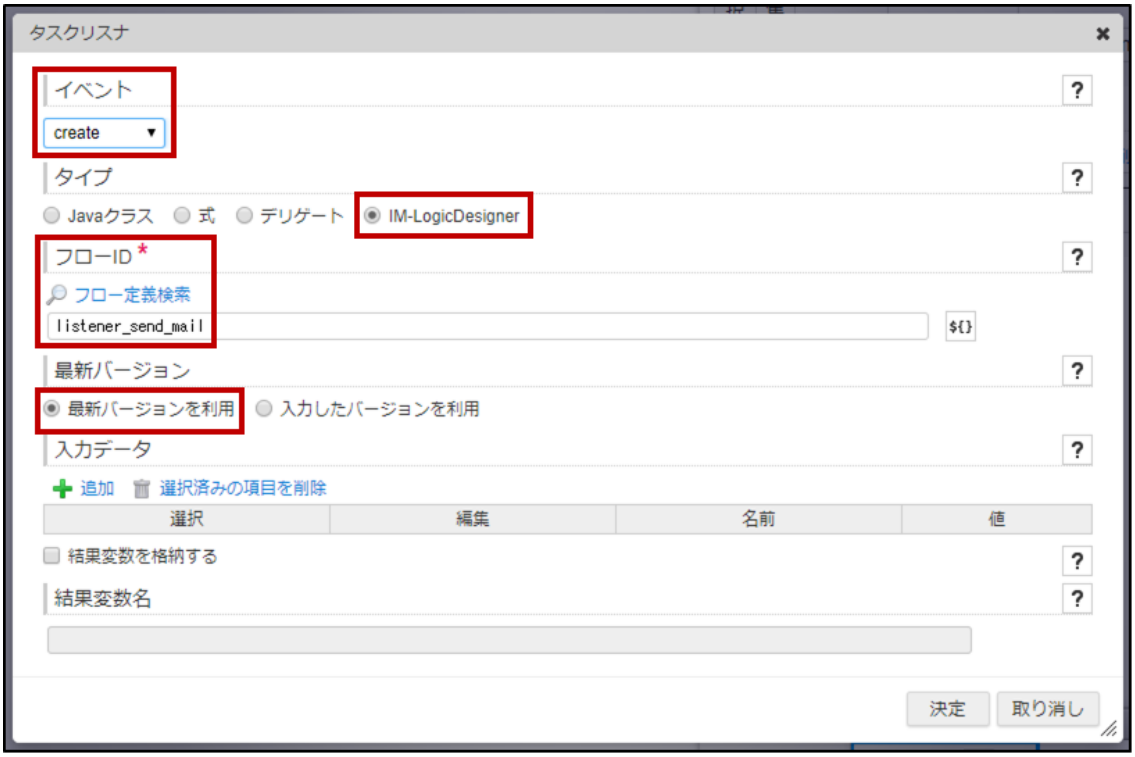


図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

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」

> **コラム**
>
> 本来ロジックフローと変数を連携するには、「IM-LogicDesigner」リスナのプロパティ「入力データ」を設定する必要があります。
> 今回は、ロジックフローの「入出力設定」で、入力データに対し「暗黙オブジェクト」を設定してあるため、必要ありません。
>
> 「暗黙オブジェクト」についての詳細は「IM-BPM 仕様書」-「IM-LogicDesignerリスナ」を参照してください。

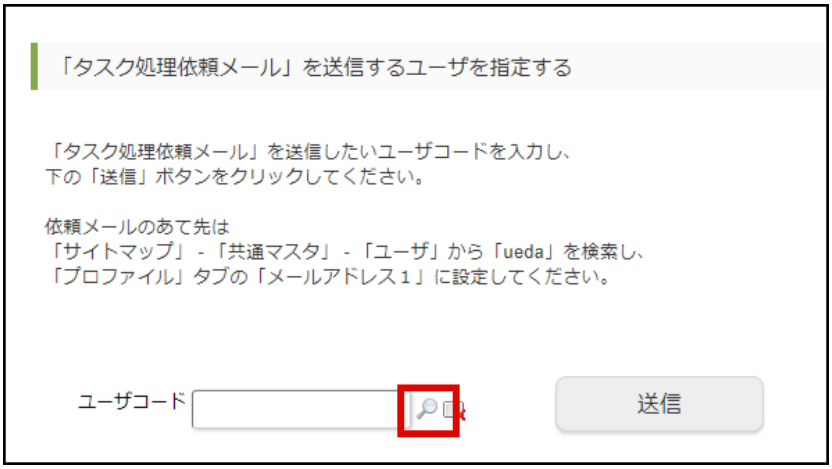
実行結果を確認する

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

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

図：「プロセス開始一覧」

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



図：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。

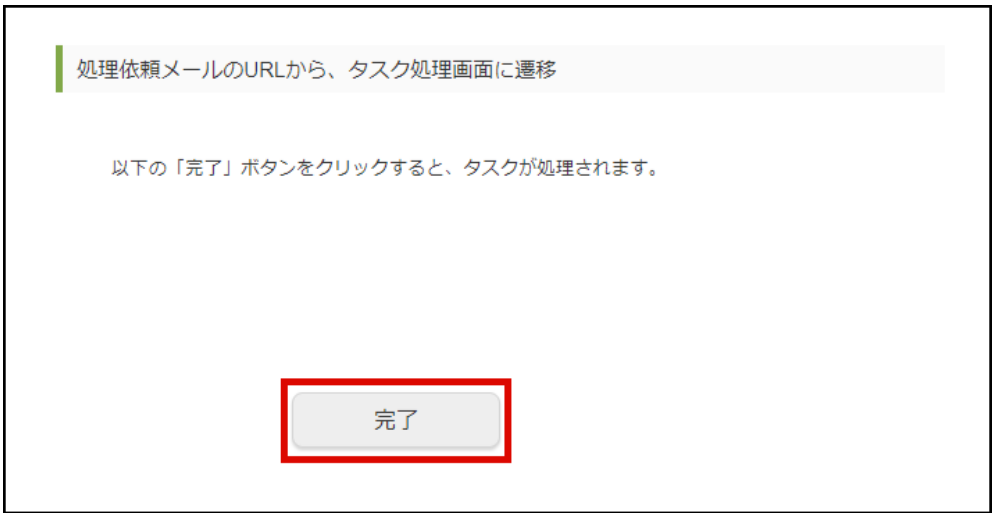
図：「ユーザ検索」

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

図：処理依頼先の「上田辰男」に届いたメール

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

5. 「完了」ボタンをクリックすることにより、タスクが完了します。



図：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 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」のサンプルを以下のリンクからダウンロードできます。
listener_add_task_candidate_user.bpmn
im_logicdesigner-data-listener_add_task_candidate_user.zip

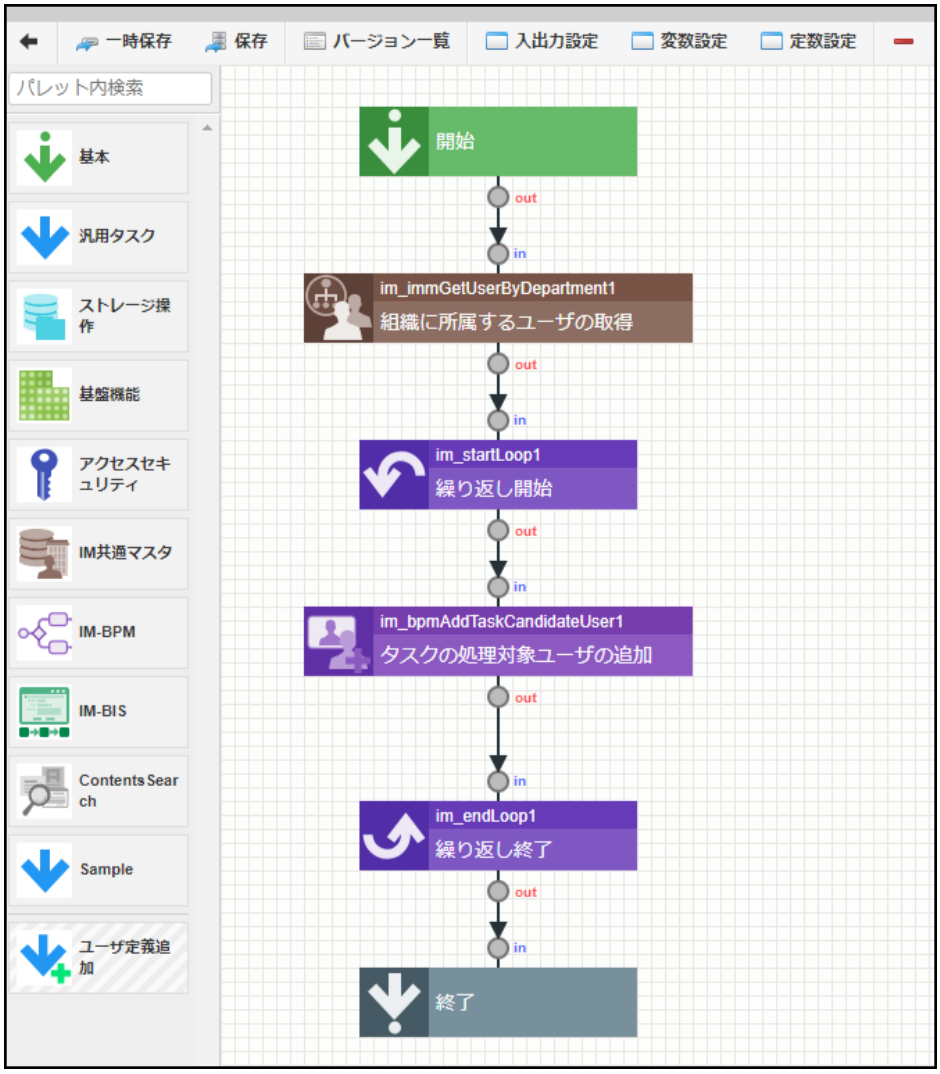
これらのサンプルは、各アプリケーションの機能でアップロード、または、インポートして利用可能です。
詳細な手順については、以下のリンク先を参照してください。

- プロセス定義：「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」
- ロジックフロー:「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」

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

ロジックフローを作成する

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



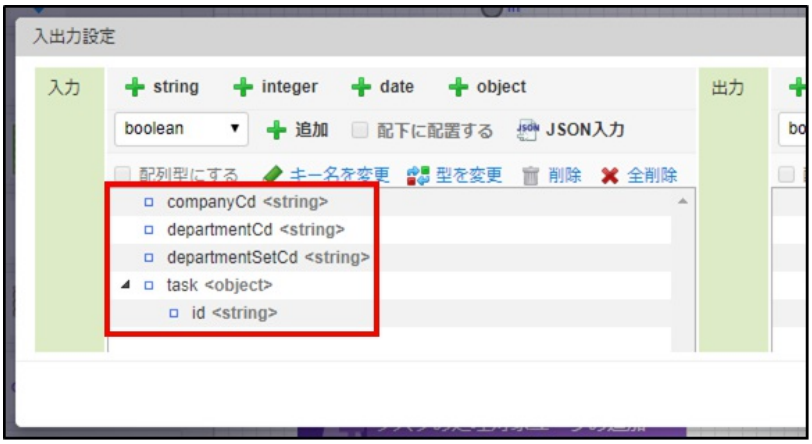
図：完成イメージ

1. ツールバーの「入出力設定」をクリックします。

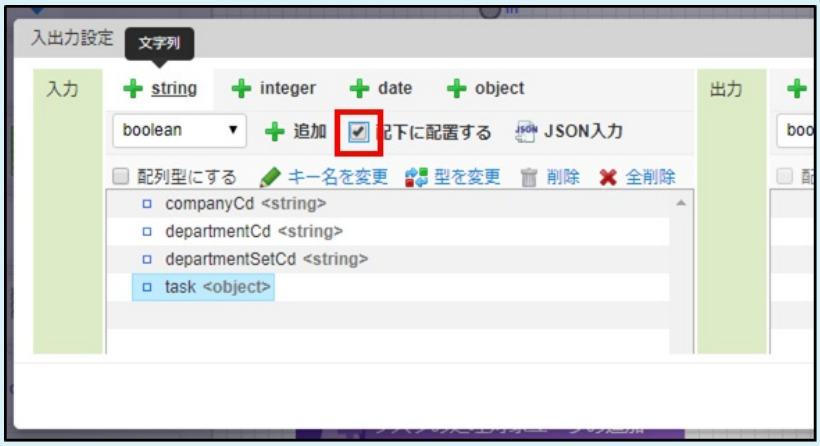
図：「ツールバー」 -「入出力設定」

2. 入力値を以下のように設定します。

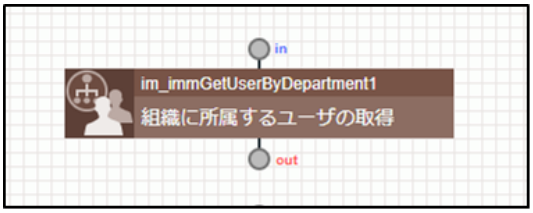
キー名	型
companyCd	<string>
departmentSetCd	<string>
departmentCd	<string>
task	<object>
task - id	<string>



図：「入出力設定」 - 「入力」

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

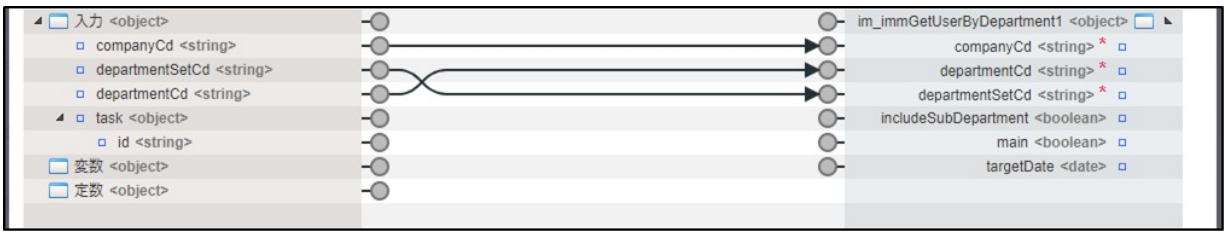
3. パレットの種別「IM共通マスタ」から、「組織に所属するユーザの取得」を設置します。



図：「組織に所属するユーザの取得」

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

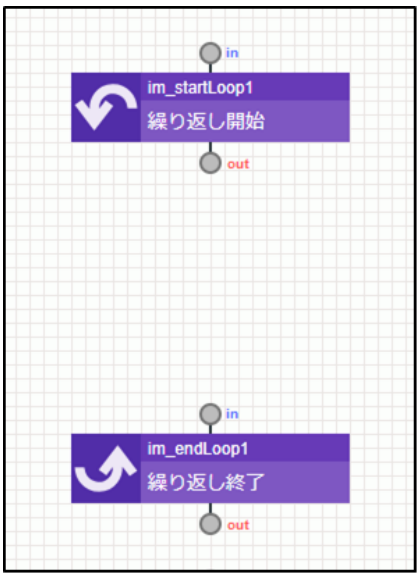
入力（始点）	出力（終点）
入力<Object> - companyCd<string>	im_immGetUserByDepartment1<object> - companyCd<string>
入力<Object> - departmentSetCd<string>	im_immGetUserByDepartment1<object> - departmentSetCd<string>
入力<Object> - departmentCd<string>	im_immGetUserByDepartment1<object> - departmentCd<string>



図：「マッピング設定」 - 「組織に所属するユーザの取得」

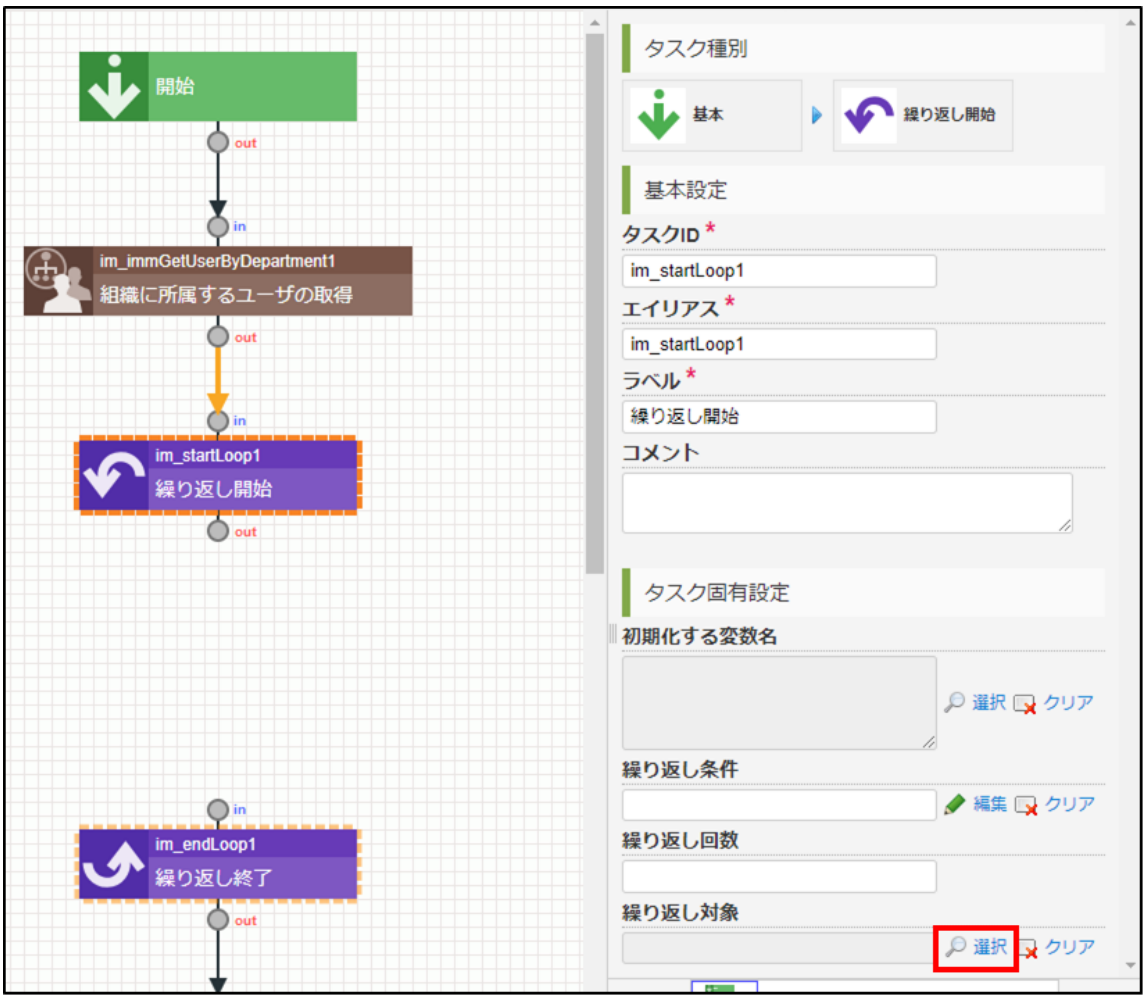
> **コラム**
>
> 「組織に所属するユーザの取得」に関するタスクの詳細は「IM-LogicDesigner仕様書」-「組織に所属するユーザの取得」を参照してください。

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



図：「繰り返し開始」と「繰り返し終了」

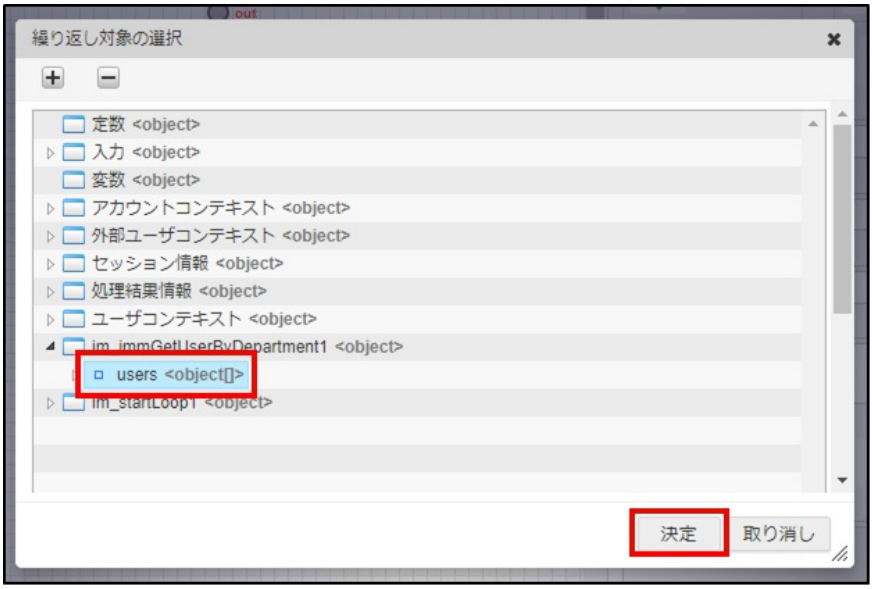
6. 「繰り返し開始」を選択した状態で、「プロパティ」 - 「タスク固有設定」 - 「繰り返し対象」の選択をクリックします。



図：「プロパティ」 - 「タスク固有設定」 - 「繰り返し対象」

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

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



図：「タスク固有設定」 -「繰り返し対象」

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

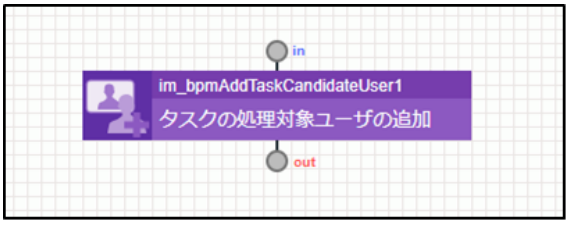


図 ：「タスク処理対象のユーザの追加」

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

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

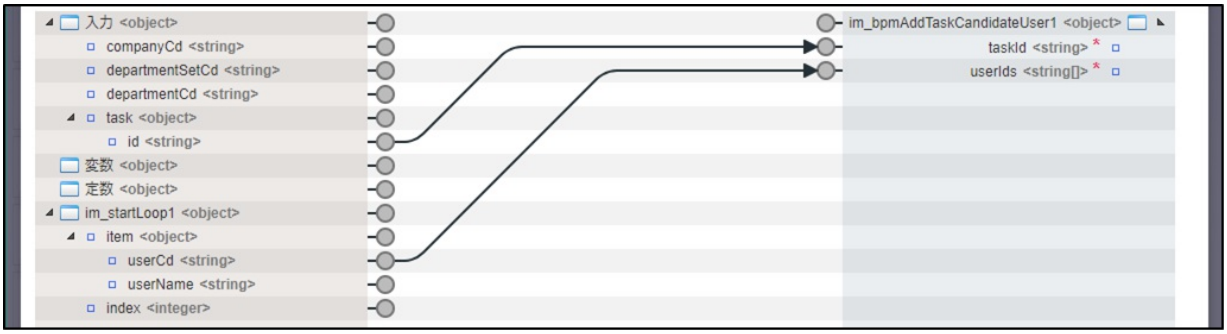
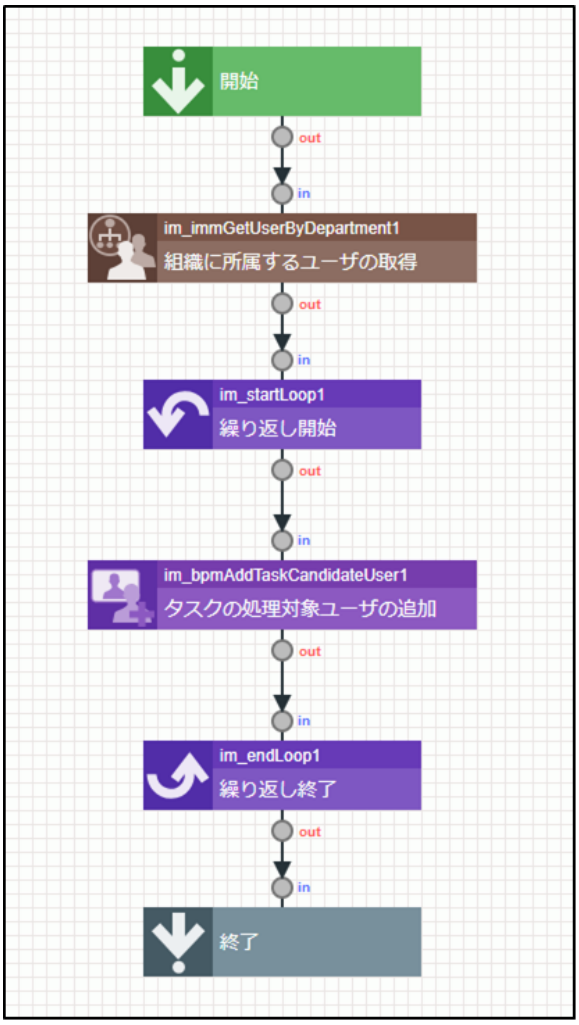


図 ：「タスク処理対象のユーザの追加」 -「マッピング設定」

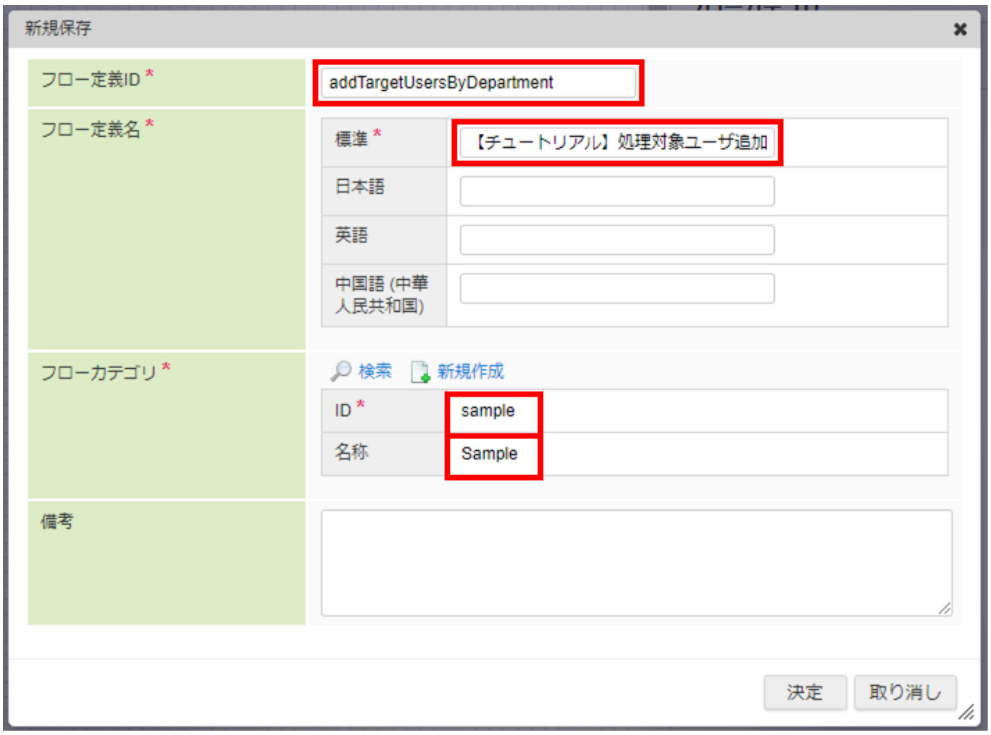
> コラム
>
> 「マッピング設定」に関するタスクの詳細は「IM-LogicDesigner仕様書」 -「タスクの処理対象ユーザの追加」を参照してください。

10. シーケンスでつなぎます。



図：「ロジックフロー定義編集画面」

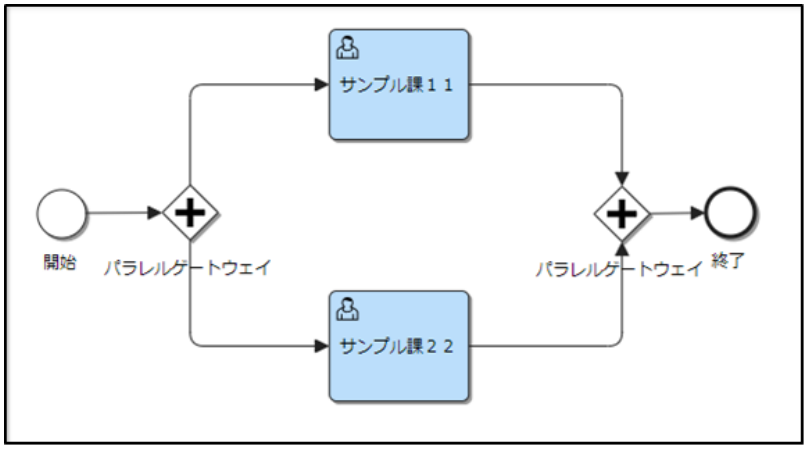
11. 「ロジックフロー定義編集」画面上部、ヘッダ内の「新規保存」をクリックします。
以下のように設定し、ロジックフロー定義を新規保存します。
    - フロー定義ID：addTargetUsersByDepartment
    - フロー定義名：【チュートリアル】処理対象ユーザ追加
    - フローカテゴリ：
        - ID：sample
        - 名称：Sample



図：「新規保存」

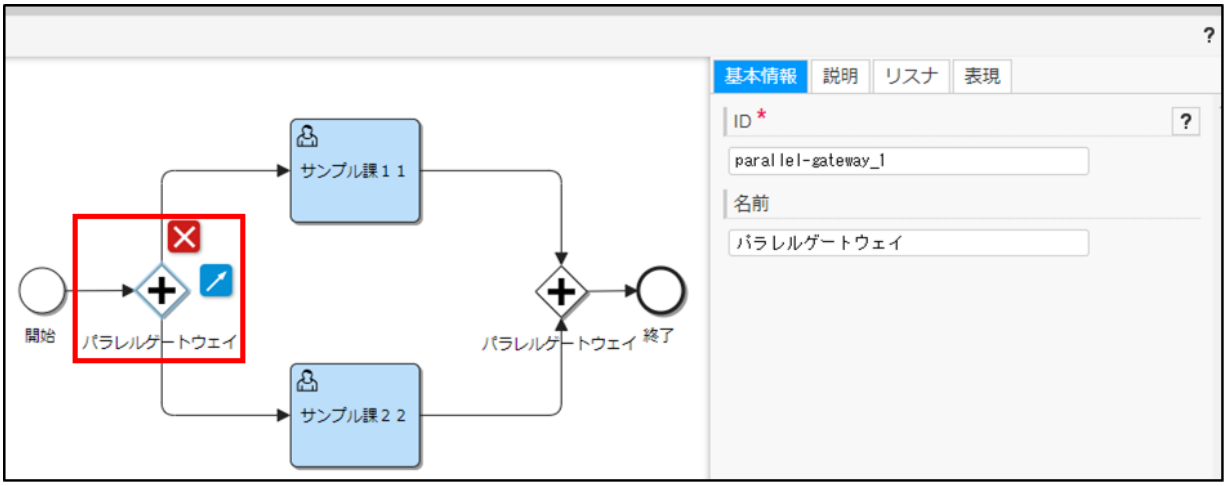
プロセス定義を作成する

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



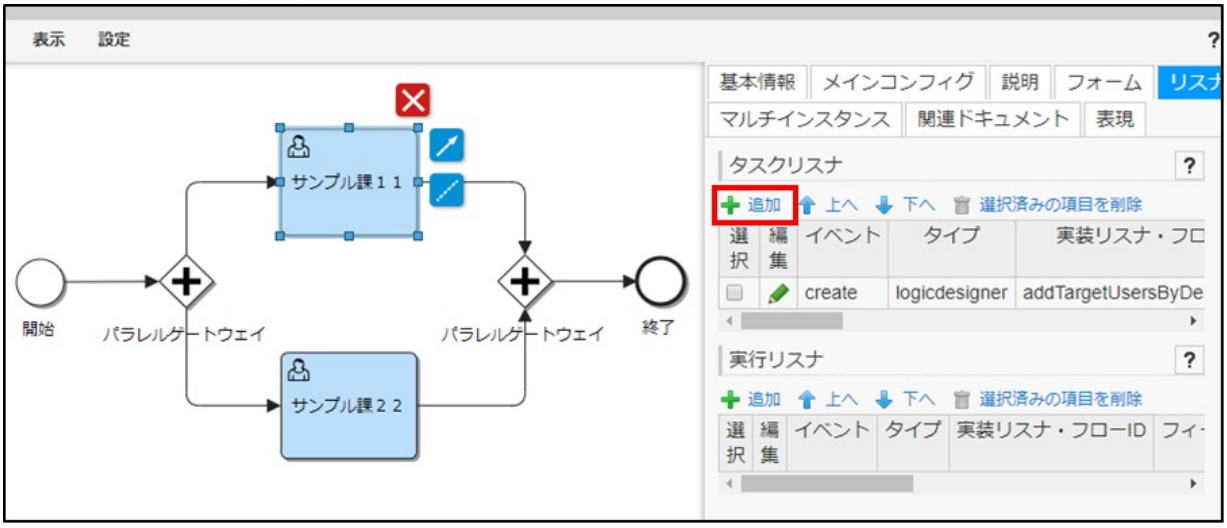
図：完成イメージ

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



図：「パラレルゲートウェイ」

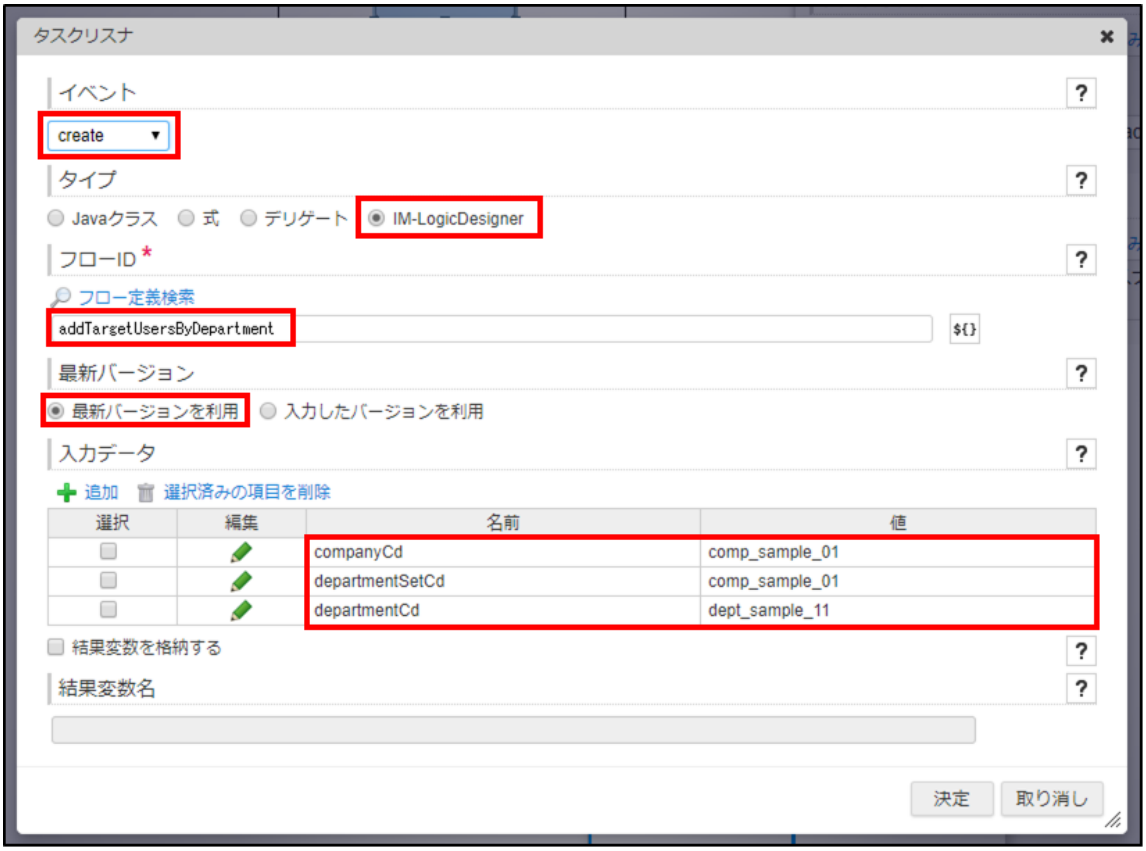
3. 「ユーザタスク」を配置し、「リスナ」タブから、「タスクリスナ」 の 「追加」リンクをクリックします。



図：「ユーザタスク」 - 「プロパティ」 - 「リスナ」

4. 「サンプル課１１」に所属しているユーザを処理対象ユーザに設定するため、「タスクリスナ」を以下のように設定します。

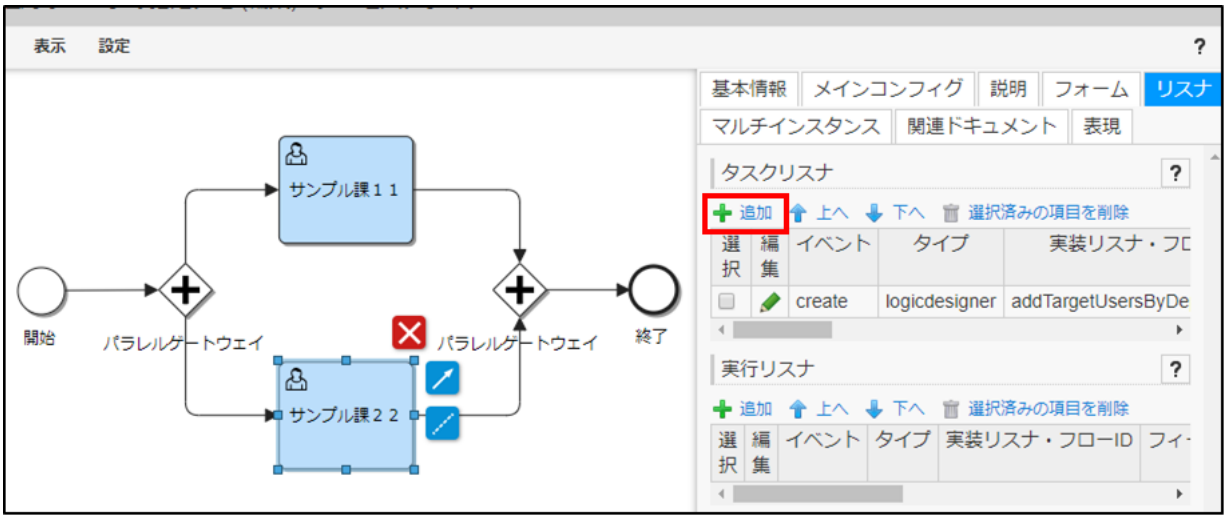
- イベント：create
- タイプ：IM-LogicDesigner
- フローID：addTargetUsersByDepartment
- 最新バージョン：最新バージョンを利用
- 入力データ1（会社コード）：
  - 名前：companyCd
  - 値：comp_sample_01
- 入力データ2（組織セットコード）：
  - 名前：departmentSetCd
  - 値：comp_sample_01
- 入力データ3（組織コード）：
  - 名前：departmentCd
  - 値：dept_sample_11



図：「ユーザタスク」 - 「プロパティ」 - 「リスナ」

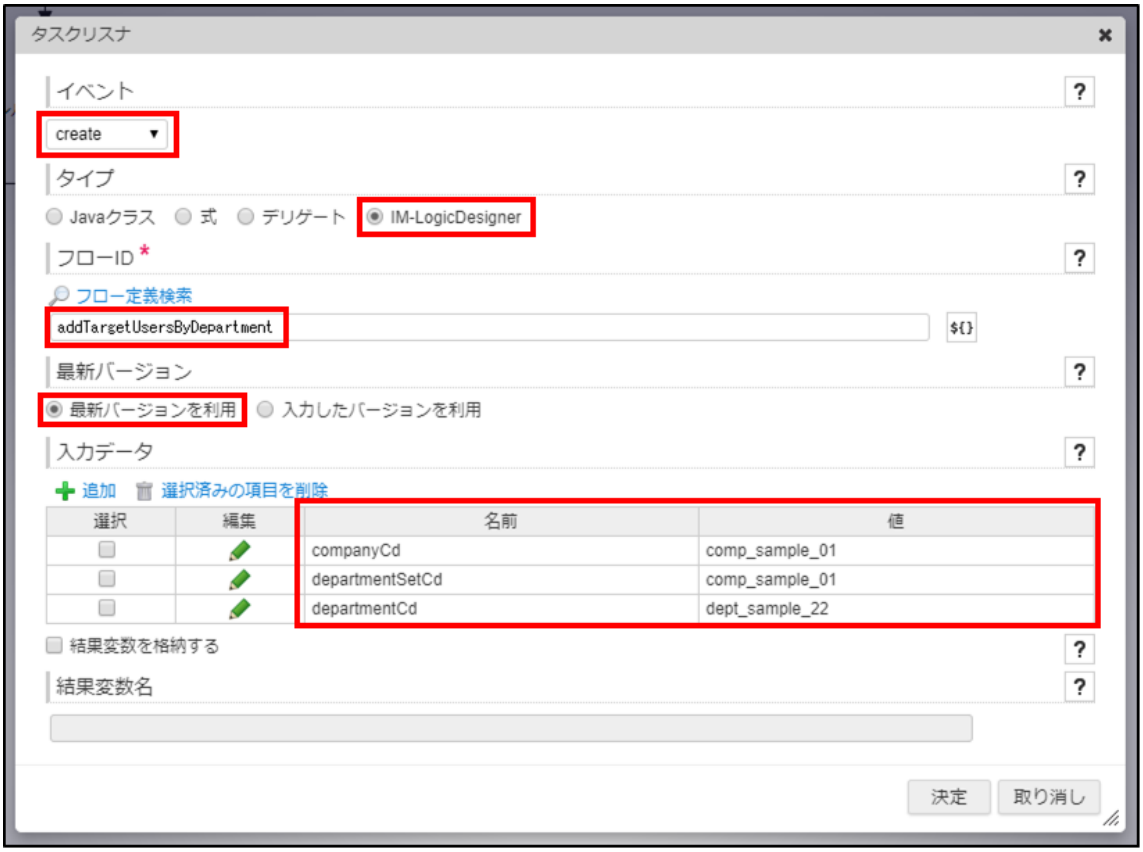
> **コラム**
>
> ロジックフローと変数を連携するには、「IM-LogicDesigner」リスナのプロパティ「入力データ」で設定する必要があります。
> task<object>とid<string>は「暗黙オブジェクト」のため、設定は行いません。
> 「暗黙オブジェクト」についての詳細は「IM-BPM 仕様書」-「IM-LogicDesignerリスナ」を参照してください。

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



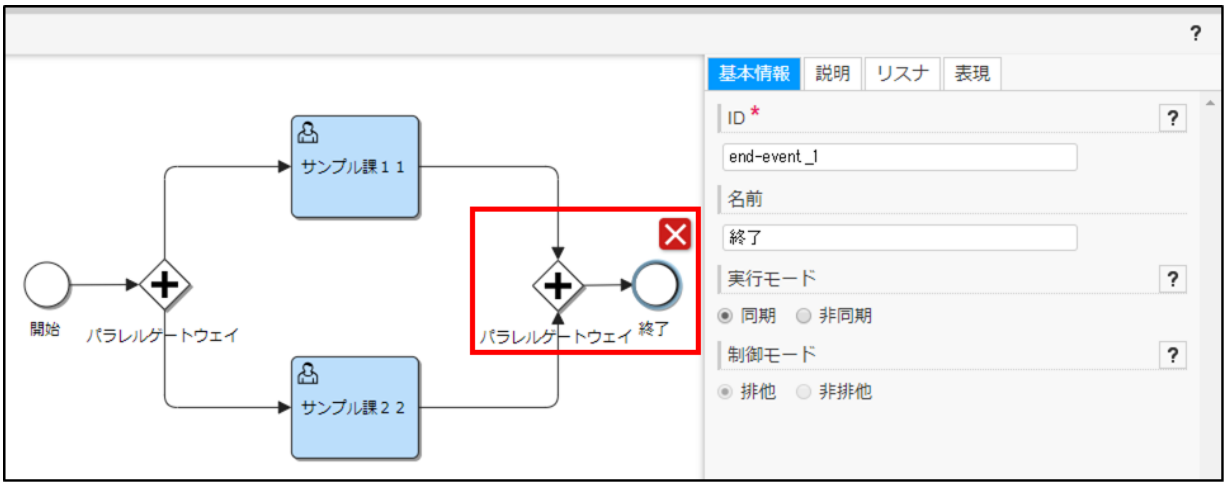
図：「ユーザタスク」 - 「プロパティ」 - 「リスナ」

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



図：「タスクリスナ」

7. 「パラレルゲートウェイ」と「終了イベント」を設置します。



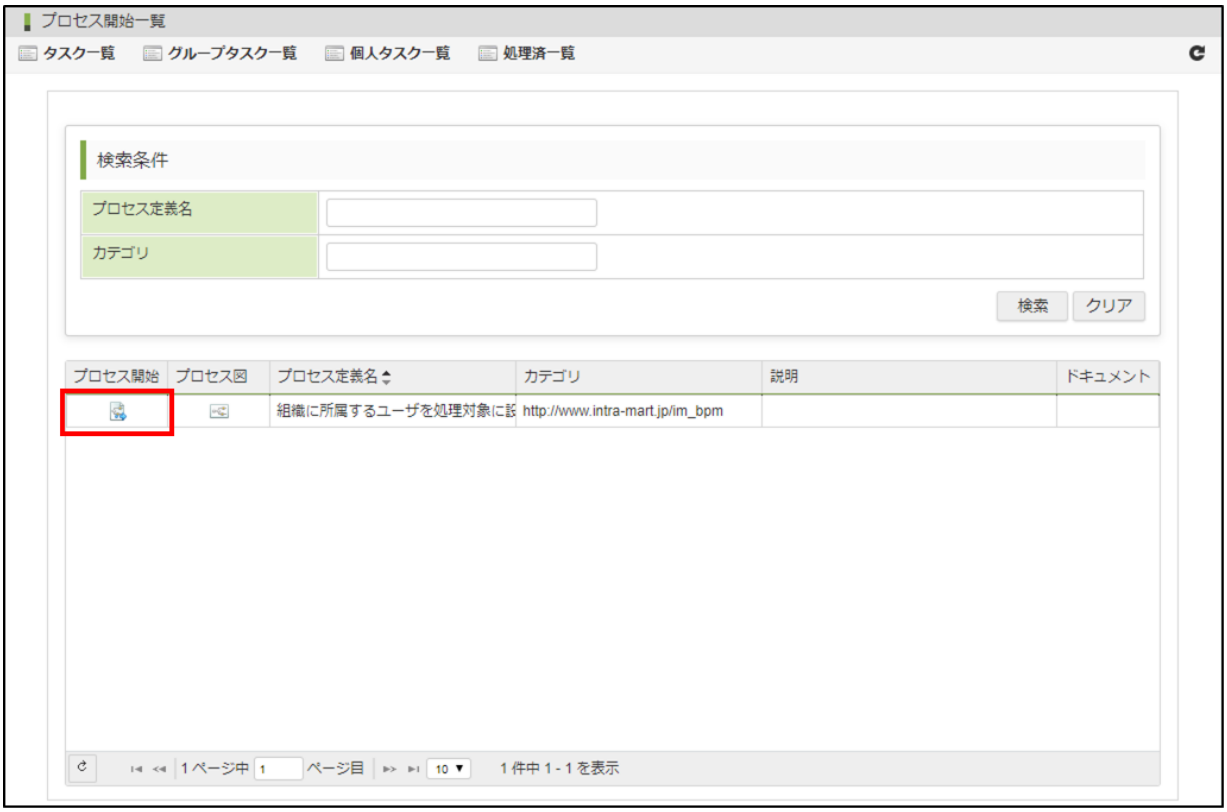
図：「パラレルゲートウェイ」と「終了イベント」

8. 名前を付けて保存します。

実行結果を確認する

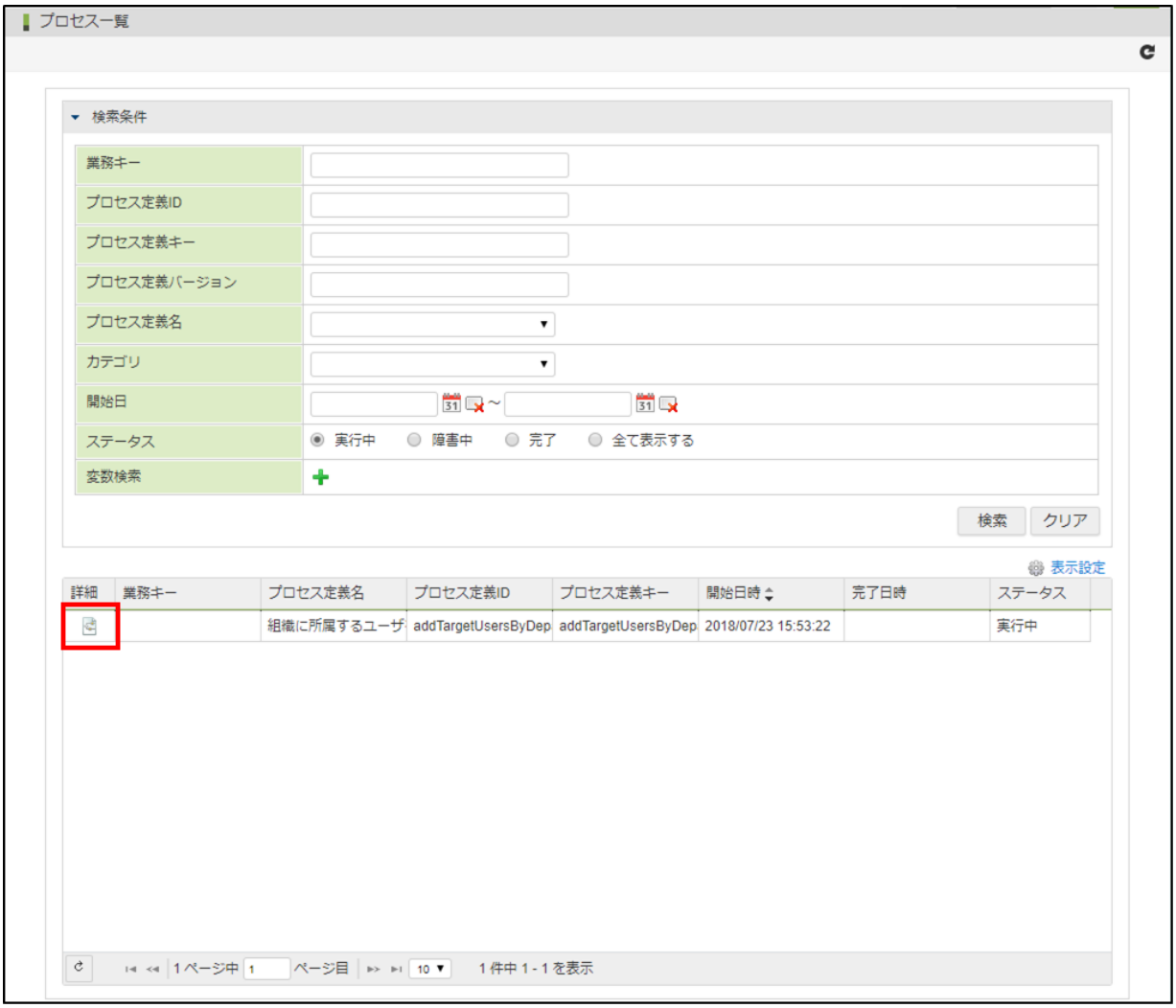
このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行した結果の確認を行います。

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



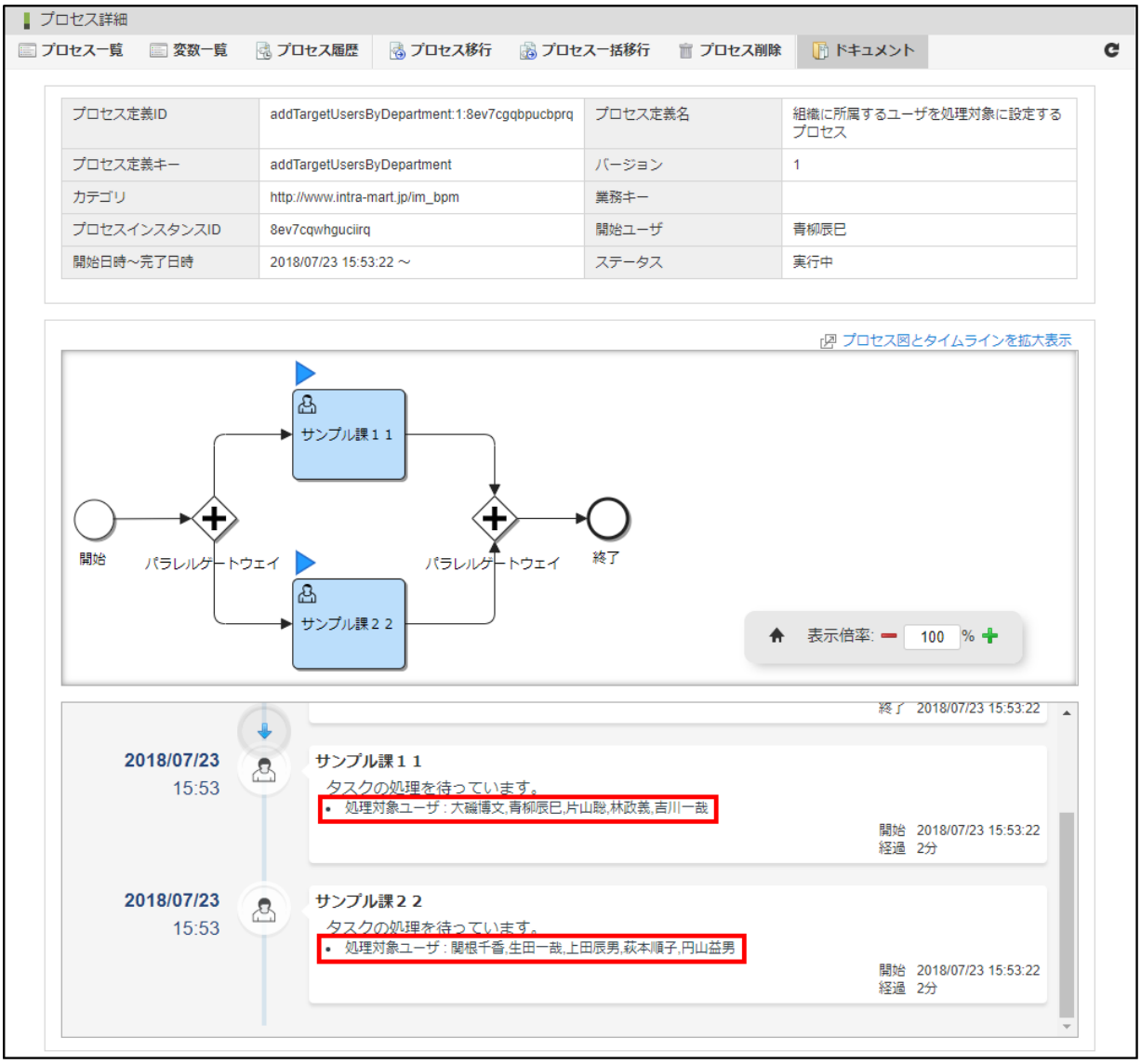
図：「プロセス開始一覧」

2. 「プロセス一覧」から、「プロセス詳細」を開きます。



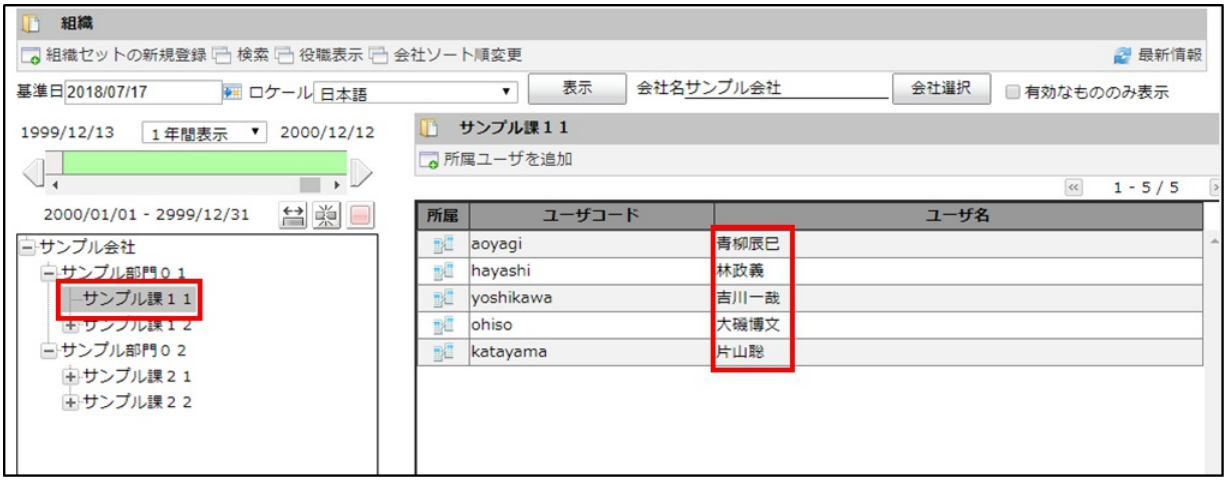
図：「プロセス一覧」

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



図：「プロセス一覧」 - 「プロセス詳細」

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



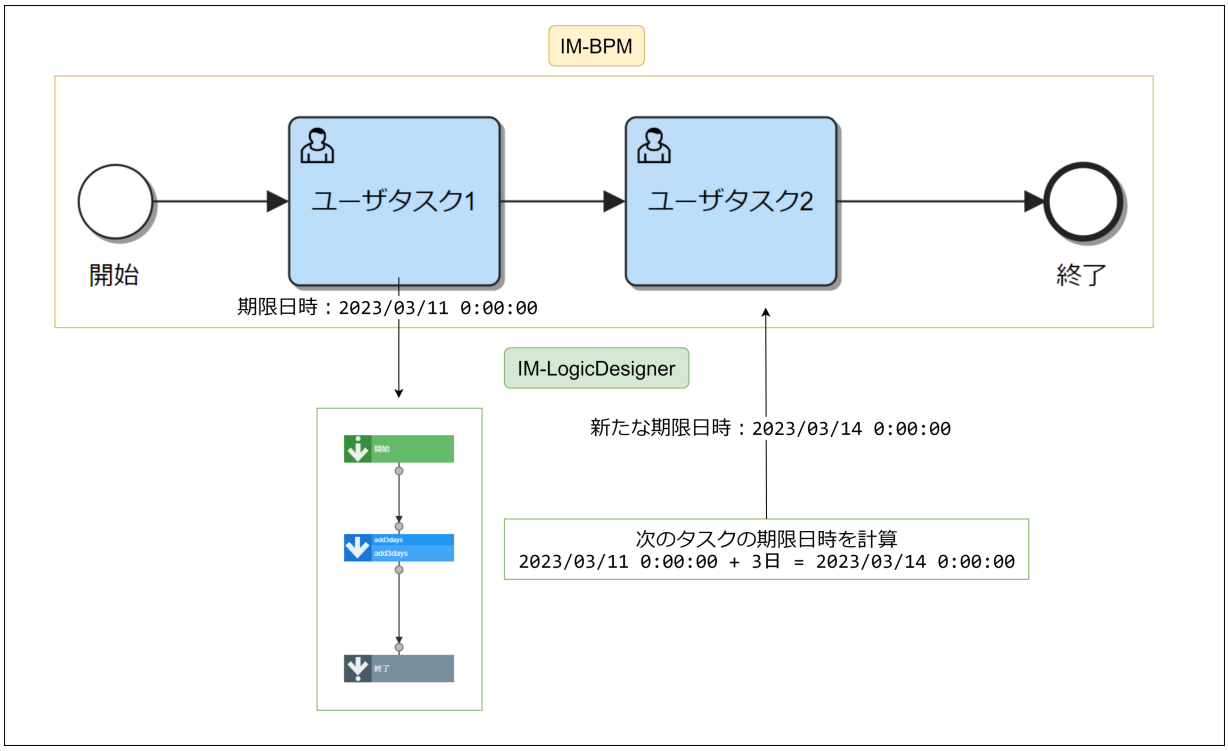
図：「共通マスタ」 - 「組織」

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

図：「共通マスタ」 - 「組織」

タスクリスナを利用して、ユーザタスクの期限日時を直前ユーザタスクの期限日時から3日後に設定する

このチュートリアルでは、タスクリスナにIM-LogicDesignerのロジックフローを設定して直前ユーザタスクの期限日時をもとに次ユーザタスクの期限日時を計算します。



図：概要

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

> **コラム**
>
> このチュートリアル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-listener_set_next_duedate.zip**
> **im_logicdesigner-data-listener_set_next_duedate_en.zip**
>
> インポート手順は「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。

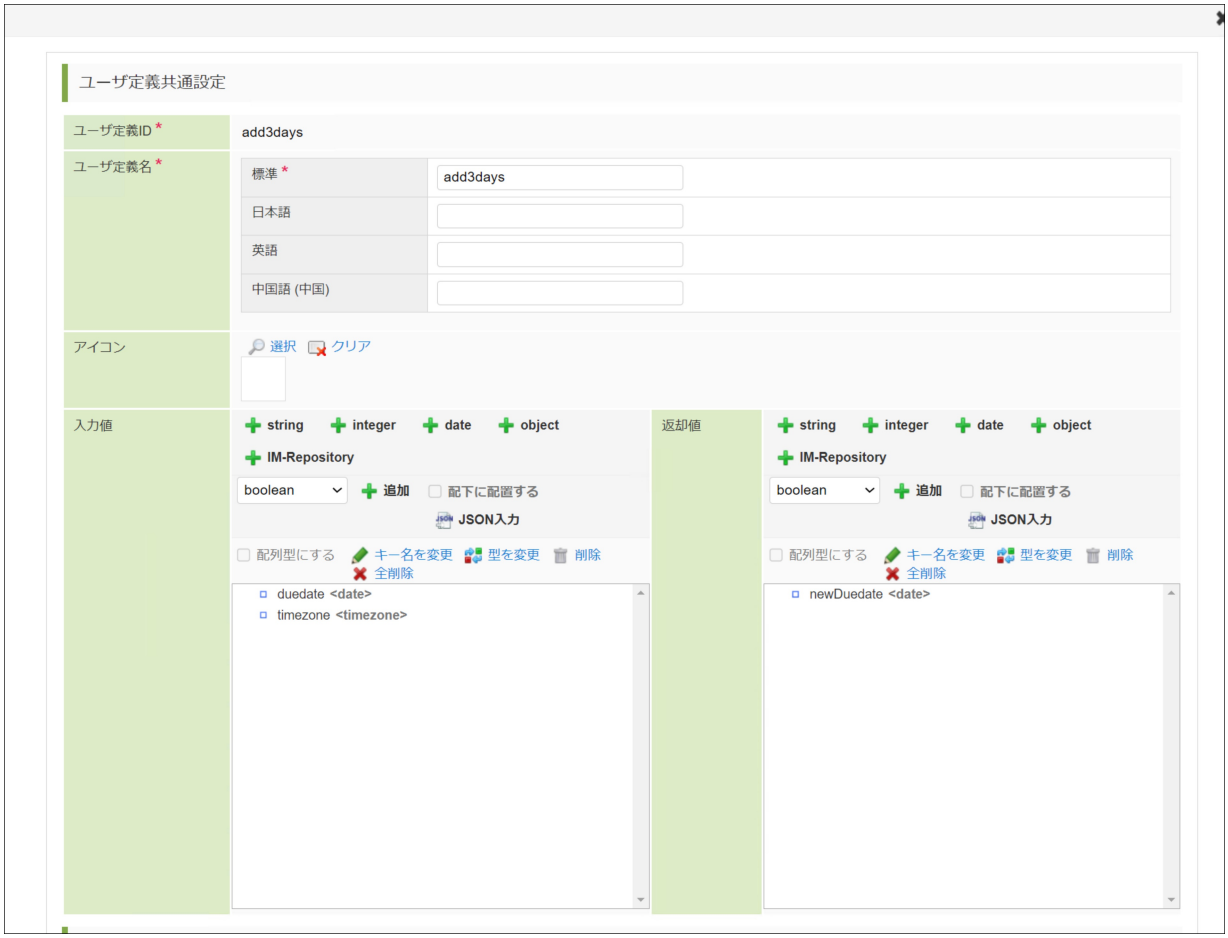
> **注意**
>
> 本サンプルは英語版と日本語版にて設定されている期限日時とタイムゾーンが異なります。
> 日本語版の期限日時は「2023/03/11 0:00:00（GMT+09:00）」、タイムゾーンは「Asia/Tokyo」です。
> 英語版の期限日時は「2023/03/11 0:00:00（GMT-05:00）」、タイムゾーンは「America/New_York」です。

- 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

直前タスクの期限日時に3日加算した日時を計算する ロジックフローを作成します。

1. 3日加算を行うJavaScriptユーザ定義を作成します。
   javascriptユーザ定義を新規作成し以下のような入力値、返却値、スクリプトを記述します。
   javascriptユーザ定義の作成方法については「IM-LogicDesigner ユーザ操作ガイド」 - 「ユーザ定義を新規登録する」を参照してください。

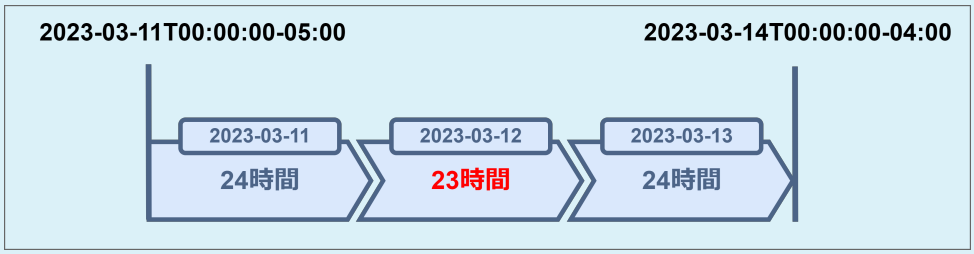


図：「add3days」 - 「JavaScript定義編集」

```
/**
* run.
*
* @param input {Object} - task input data.
* @return {Object} task result.
*/
function run(input) {
    const datetime = new DateTime(input.duedate, input.timezone);
    const addDaysNum = 3;
    const newDuedate = datetime.plusDays(addDaysNum).data.getDate().data;

    return {
        newDuedate:newDuedate
    };
}
```

コラム

スクリプト開発向けAPIであるDateTimeオブジェクトを利用することで、夏時間を考慮した日付の加算が可能です。
具体的には、日付加算の前後で夏時間が発生する場合一日を24時間ではなく23時間または25時間として加算します。
例：指定したタイムゾーンがAmerica/New_Yorkの場合→2023/03/11 0:00:00(GMT-05:00) + 3日 = 2023/03/14 0:00:00(GMT-04:00)



図：日付加算の例

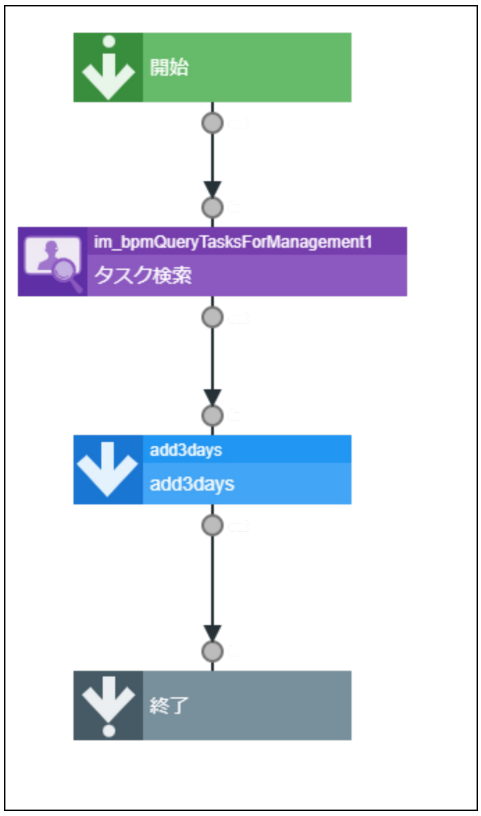
intra-mart Accel Platform のタイムゾーンや夏時間、日付と時刻の形式に関する考え方や仕様、制限事項については「intra-mart Accel Platform タイムゾーン仕様書」を参照してください。
スクリプト開発向けAPIであるDateTimeオブジェクトについては「intra-mart Accel Platform スクリプト開発向けim-BizAPI」-「DateTimeオブジェクト」を参照してください。

注意

IM-BPMの夏時間への対応について

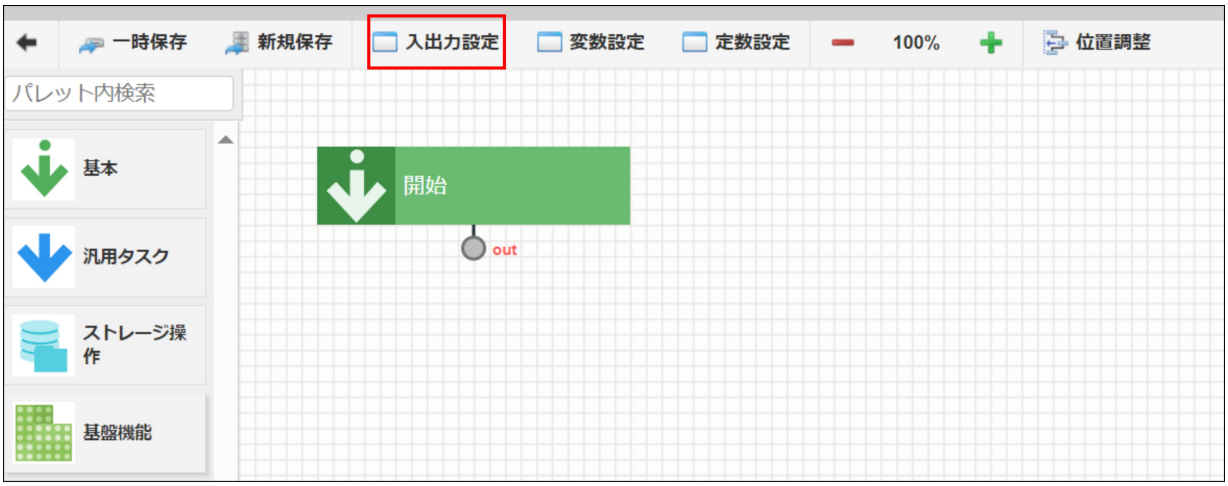
IM-BPMの夏時間への対応は2024 Spring(Iris)以降のバージョンでのみサポートされています。
2023 Autumn(Hollyhock)以前のバージョンでは、IM-BPMでの日時の表示や入力において夏時間の考慮が行われていません。

2. 「開始タスク」、「タスク検索タスク」、「add3daysタスク（直前で作成したJavaScriptユーザ定義）」「終了タスク」を配置し、順に接続します。



図：ロジックフロー

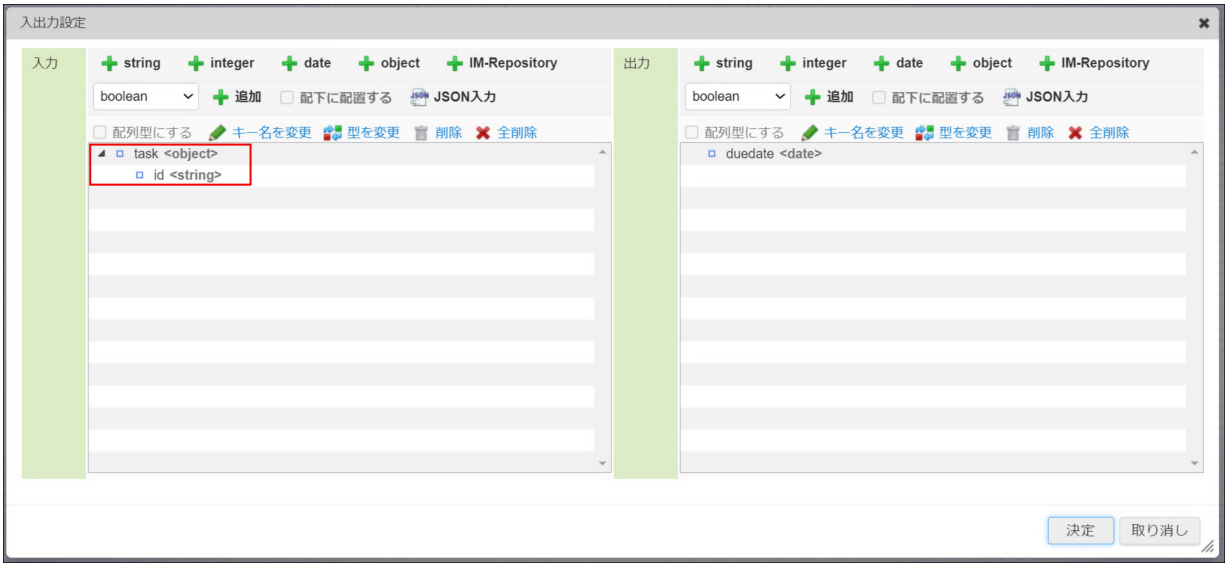
3. ツールバーの「入出力設定」をクリックします。



図：「ツールバー」 - 「入出力設定」

4. 入力値を以下のように設定します。

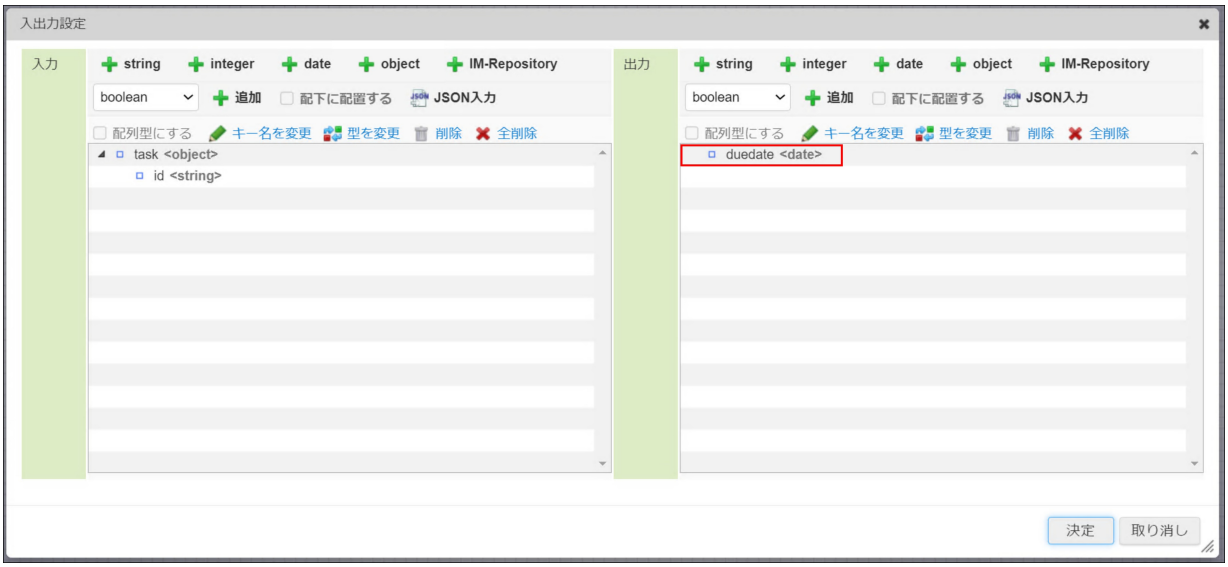
キー名	型
task	<object>
id	<string>



図：「入出力設定」 - 「入力」

5. 出力値を以下のように設定します。

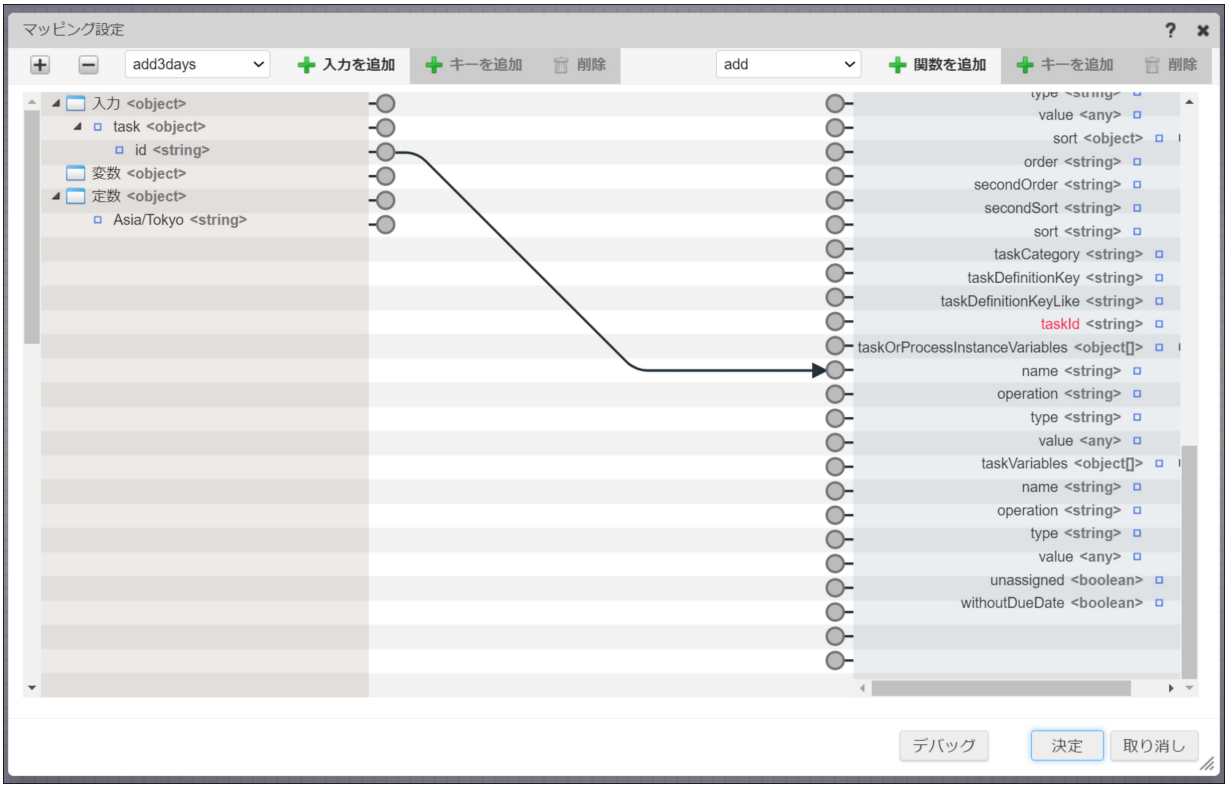
キー名	型
duedate	<date>



図：「入出力設定」 - 「出力」

6. タスクIDから、ユーザタスクの期限日時を取得します。
「タスク検索タスク」のマッピング設定を開き以下をマッピングします。

入力（始点）	出力（終点）
入力<object> - task<object> - id<string>	im_bpmQueryTasksForManagement1<object> - taskId<string>



図：「タスク検索タスク」 - 「マッピング設定」

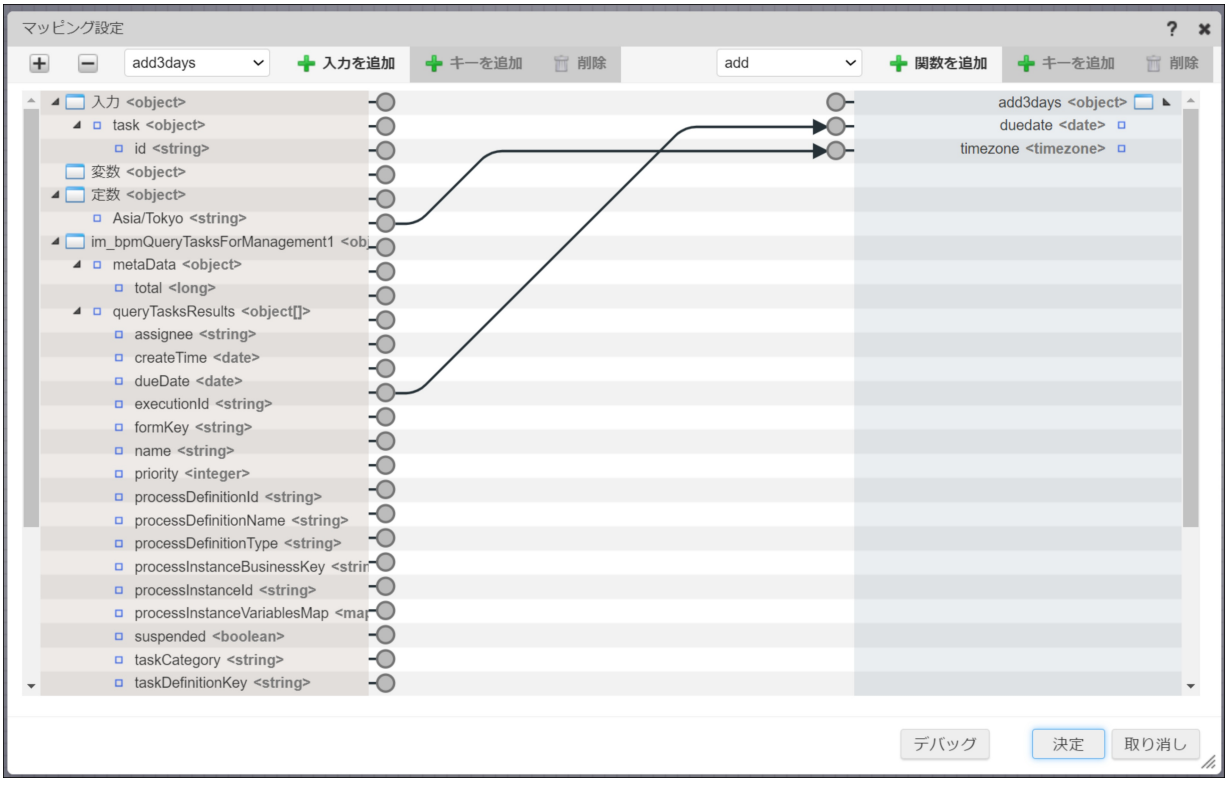
7. 期限日時の計算時に考慮したいタイムゾーンIDを定数に作成します。
メニューバーの「定数設定」をクリックし「定数設定」を表示します。
定数値を以下のように定義します。

定数ID	定数値	説明
Asia/Tokyo（英語版資材は America/New_York）	Asia/Tokyo（英語版資材は America/New_York）	期限日時の計算時に使用するタイムゾーンID

図：「定数設定」

8. 「タスク検索タスク」で取得した期限日時をもとに3日後の日時を計算します。
「add3daysタスク」のマッピング設定を開き以下をマッピングします。

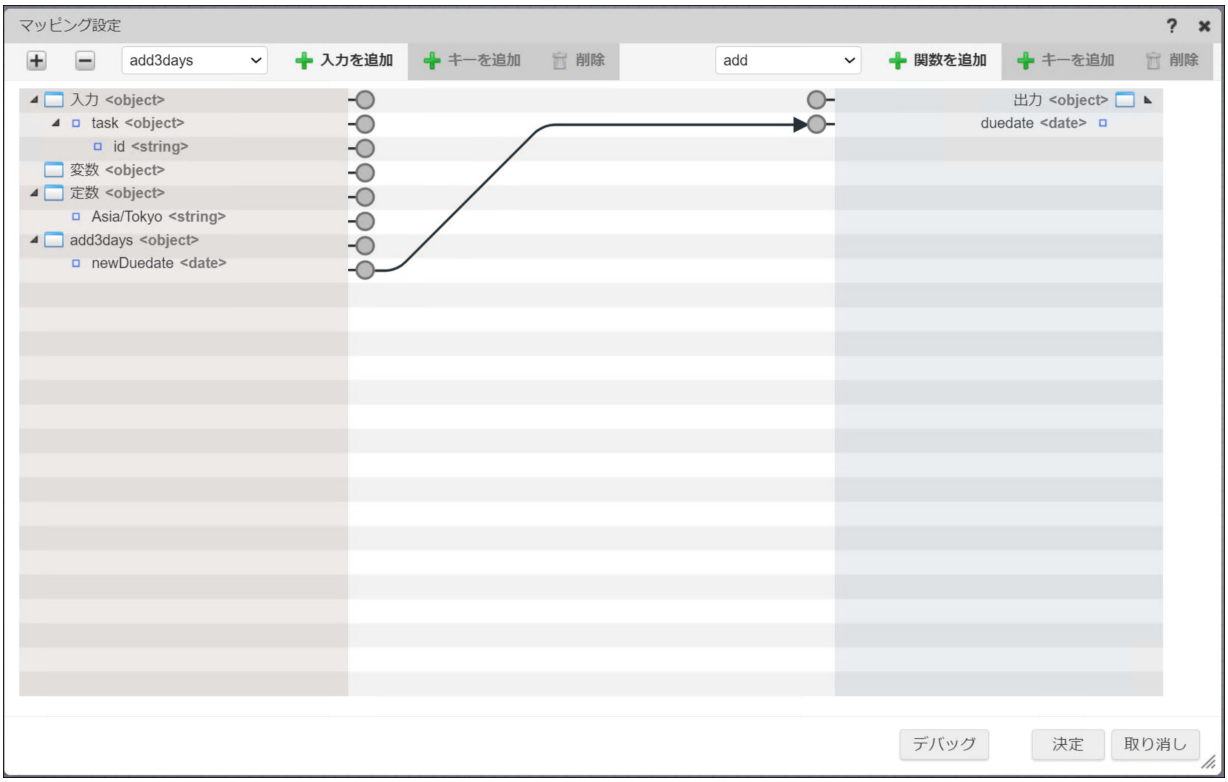
入力（始点）	出力（終点）
im_bpmQueryTasksForManagement1<object> - queryTasksResults<object> - duedate<date>	add3days<object> - duedate<date>
定数<object> - Asia/Tokyo（英語版資材は America/New_York）<string>	add3days<object> - timezone<timezone>



図：「add3daysタスク」 - 「マッピング設定」

9. 「終了タスク」のマッピング設定を開き以下をマッピングします。

入力（始点）	出力（終点）
add3days<object> - newDuedate<date>	出力<object> - variablesMap<map>



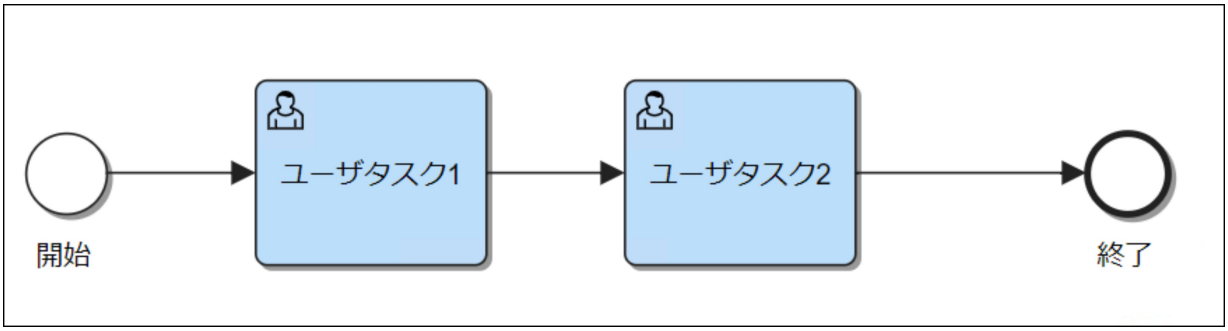
図：「終了タスク」 -「マッピング設定」

10. ツールバーの「新規保存」をクリックし、以下を設定しロジックフローを保存します。
    - フロー定義ID：listener_set_next_duedate
    - フロー定義名：【チュートリアル】listener_set_next_duedate
    - フローカテゴリ：
        - ID：im_logicdesigner-data
        - 名称：BPMチュートリアル

> **コラム**
> - ロジックフロー定義の作成方法については「IM-LogicDesigner ユーザ操作ガイド」を参照してください。

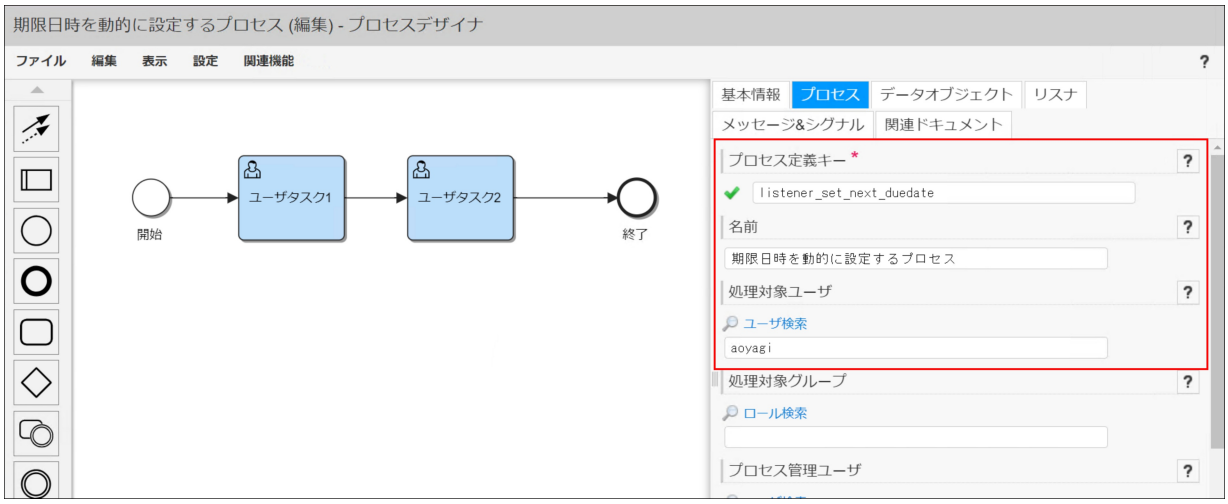
プロセス定義を作成する

1. 開始イベント、ユーザタスク、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。



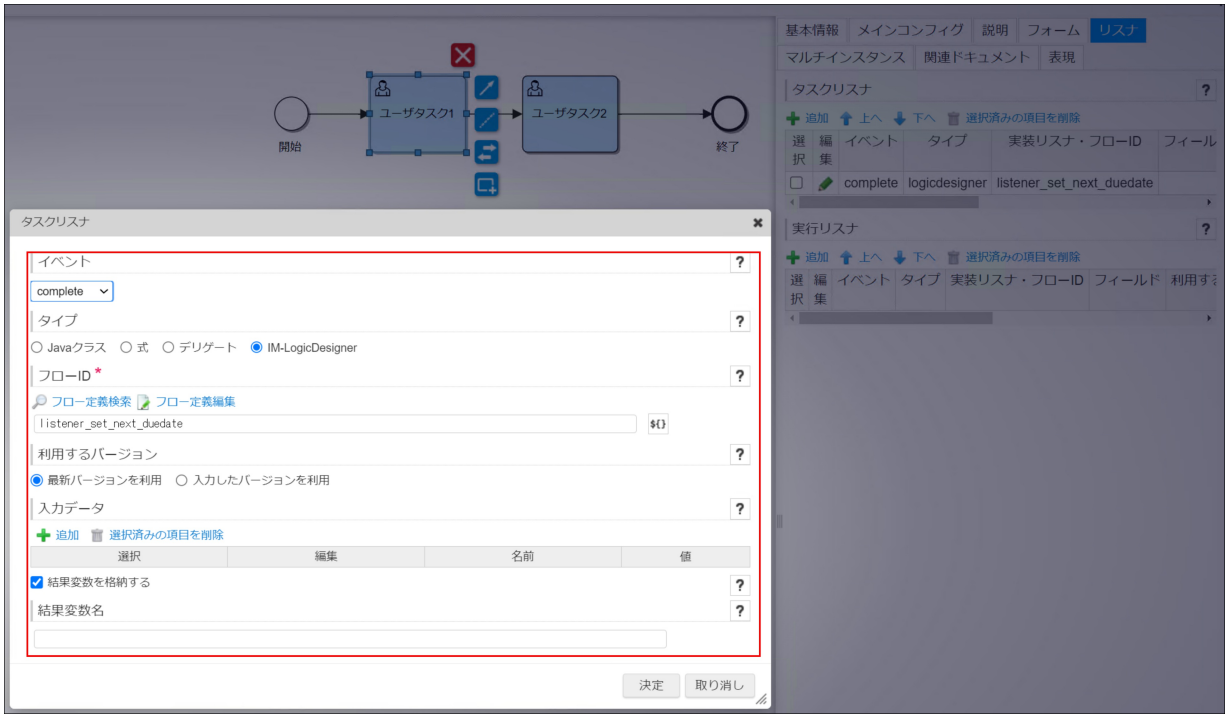
図：完成イメージ

2. 「プロセス」に、プロセス定義キーとプロセス名、処理対象ユーザを設定します。
余白を選択し、「プロセス」タブの「プロセス定義キー」、「名前」、「処理対象ユーザ」に以下の値を設定します。
    - プロセス定義キー：listener_set_next_duedate
    - 名前：期限日時を動的に設定するプロセス
    - 処理対象ユーザ：aoyagi



図：「プロパティ」 - 「プロセス」

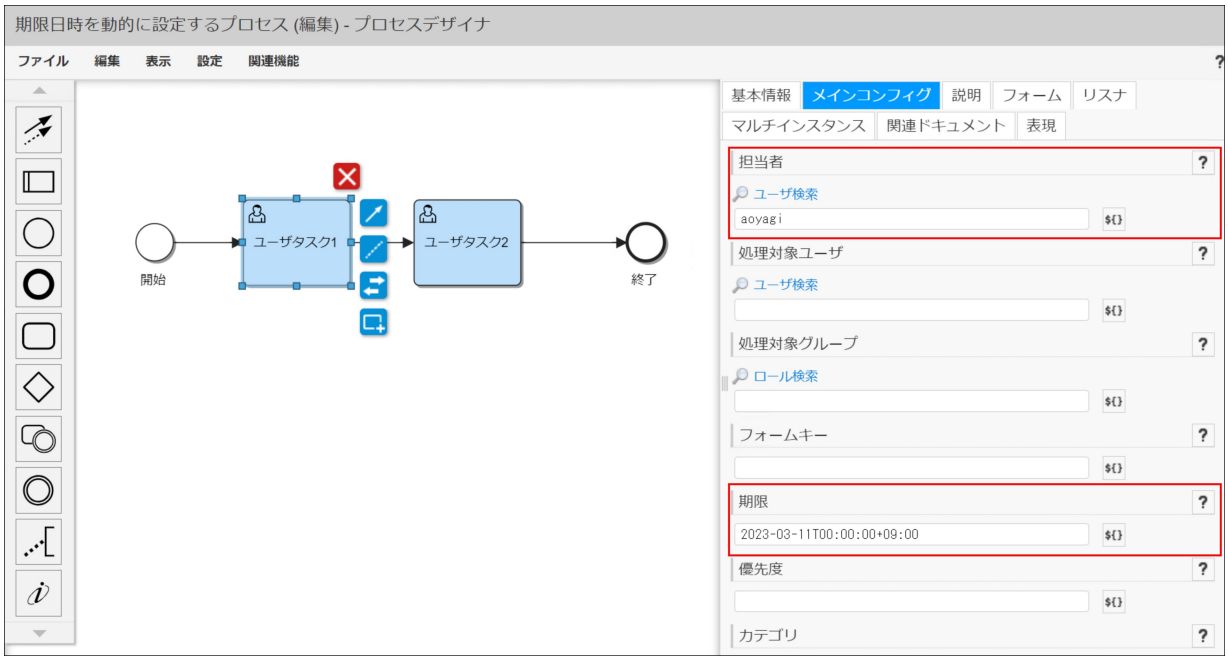
3. 「ユーザタスク1」の「タスクリスナ」に作成したロジックフローを設定します。
   「ユーザタスク1」を選択し、「リスナ」タブから「タスクリスナ」の「追加」をクリックします。
   以下のとおりに項目の値を設定してください。
   - イベント：complete
   - タイプ：IM-LogicDesigner
   - フローID：listener_set_next_duedate
   - 最新バージョン：最新バージョンを利用
   - 入力データ：結果変数を格納する
   - 結果変数名：（空欄）



図：「ユーザタスク1」 - 「プロパティ」 - 「リスナ」 - 「タスクリスナ」

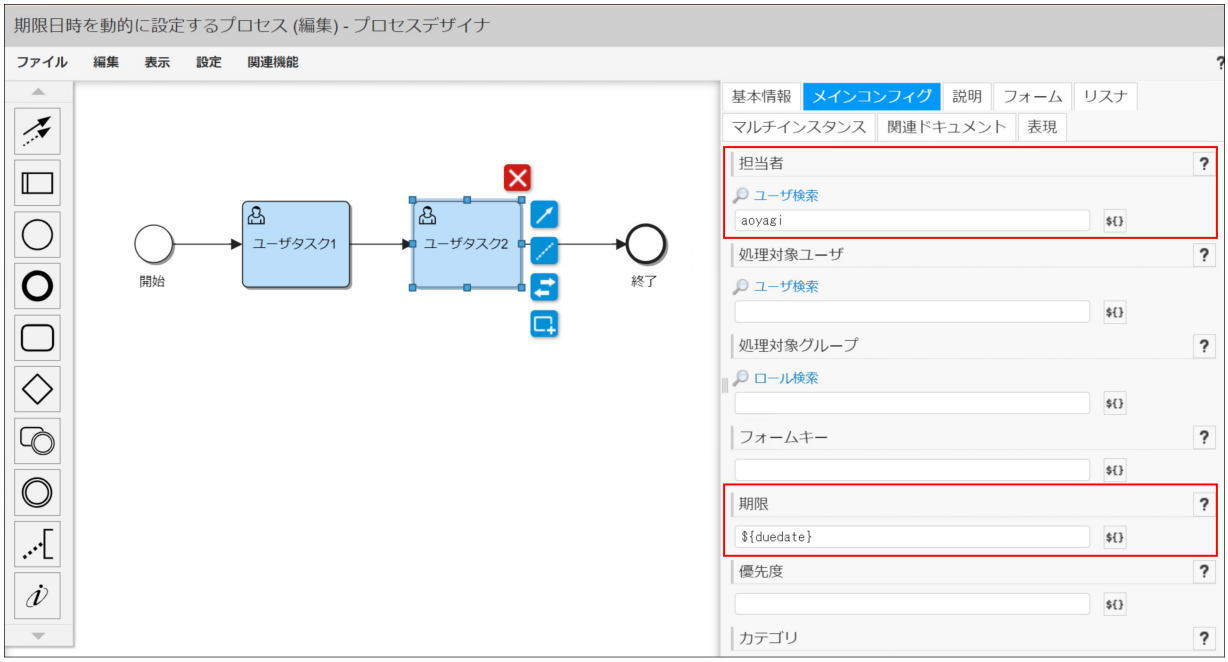
> **コラム**
>
> 本来ロジックフローと変数を連携するには、「IM-LogicDesigner」リスナのプロパティ「入力データ」を設定する必要があります。
> 今回は、ロジックフローの「入出力設定」で、入力データに対し「暗黙オブジェクト」を設定してあるため、必要ありません。
>
> 「暗黙オブジェクト」についての詳細は「IM-BPM 仕様書」-「IM-LogicDesignerリスナ」を参照してください。
> また結果変数名を空欄にした場合、ロジックフローの出力値の変数がそのままプロセスインスタンス変数として登録されます。

4. 「ユーザタスク1」に担当者と期限日時を設定します。
   「ユーザタスク1」を選択し、「メインコンフィグ」タブの「担当者」と「期限」に以下の値を設定します。
   - 担当者：aoyagi
   - 期限：2023-03-11T00:00:00+09:00（英語版資材は2023-03-11T00:00:00-05:00）



図：「ユーザタスク1」 -「プロパティ」 -「メインコンフィグ」

5. 「ユーザタスク2」に担当者と期限日時を設定します。
   期限日時にはEL式を使用して、ロジックフロー【チュートリアル】listener_set_next_duedate``の出力値``duedate を指定します。
   「ユーザタスク2」を選択し、「メインコンフィグ」タブの「担当者」と「期限」に以下の値を設定します。
   - 担当者：aoyagi
   - 期限：${duedate}



図：「ユーザタスク2」 -「プロパティ」 -「メインコンフィグ」

6. 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します。また、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、プロセス定義名 期限日時を動的に設定するプロセス を開始します。

2. 「タスク一覧」 画面の個人タスクを確認します。
   タスク名「ユーザタスク1」の期限日時が「2023/03/11 0:00:00」であることを確認します。
   タスク名「ユーザタスク1」の「処理」をクリックし、タスクを処理します。

3. 「タスク一覧」 画面の個人タスクを確認します。
   タスク名「ユーザタスク2」の期限日時が「2023/03/14 0:00:00」であることを確認します。

> **注意**
>
> 英語版と日本語版で期限日時とタイムゾーンの設定が異なります。以下の条件で確認を行うと上記のような結果を確認できます。
> 日本語版：aoyagiユーザのタイムゾーン設定は「Asia/Tokyo」
> 英語版：aoyagiユーザのタイムゾーン設定は「America/New_York」

関連ドキュメント

IM-Wikiを関連ドキュメントとして設定する

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

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

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

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

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

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

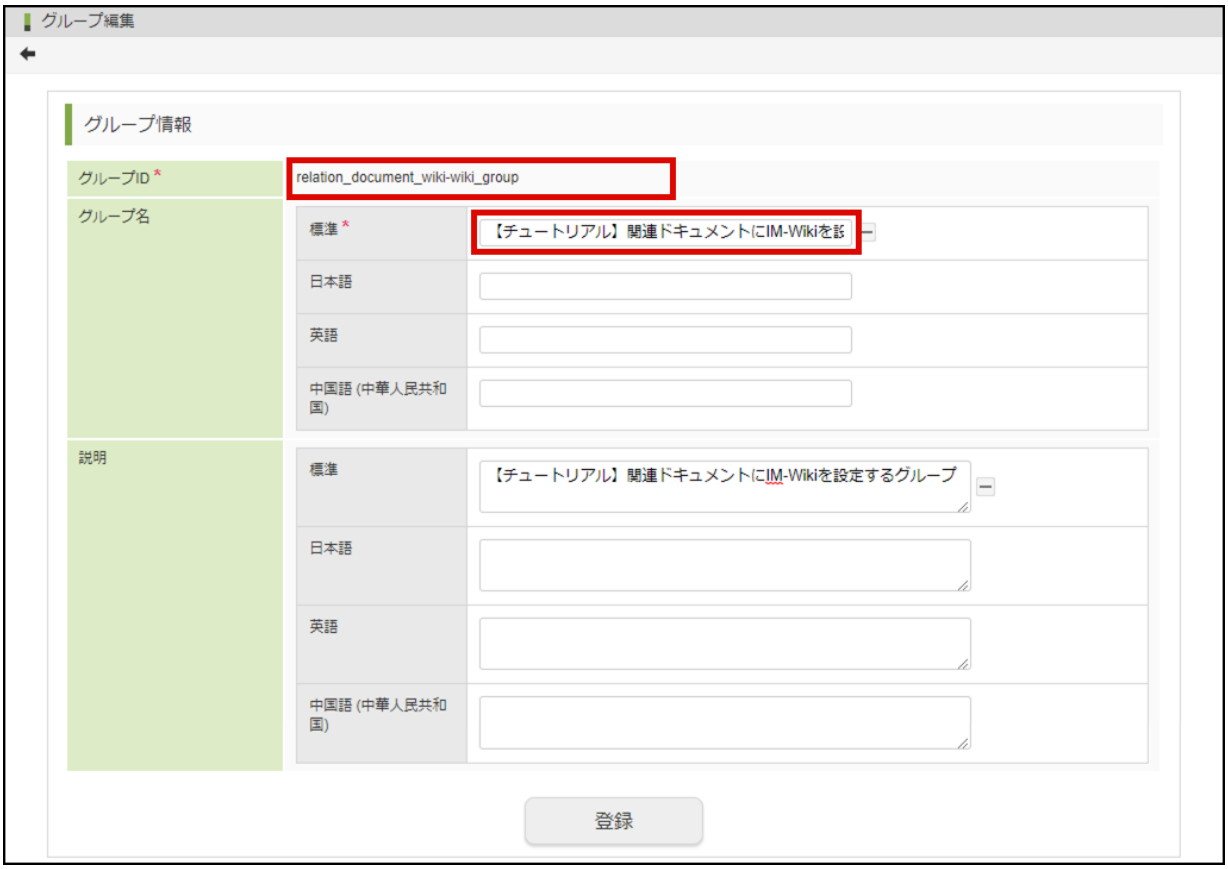
1. 「IM-Knowledge」のグループを作成します。
   「テナント管理者」でログインします。

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

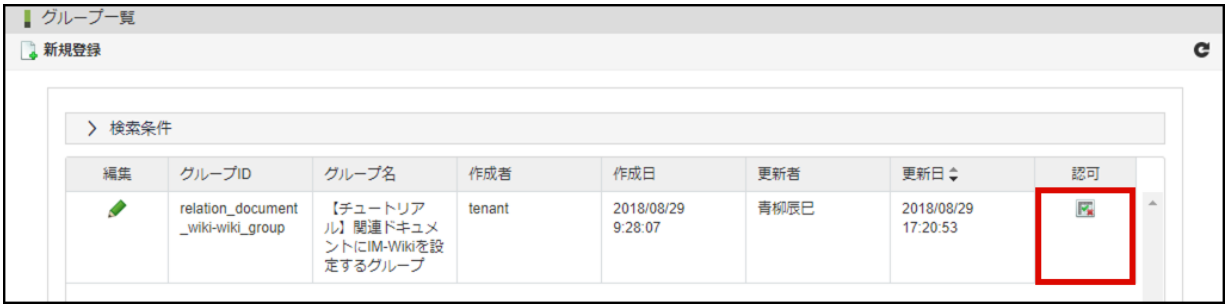


図：「グループ登録」

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。

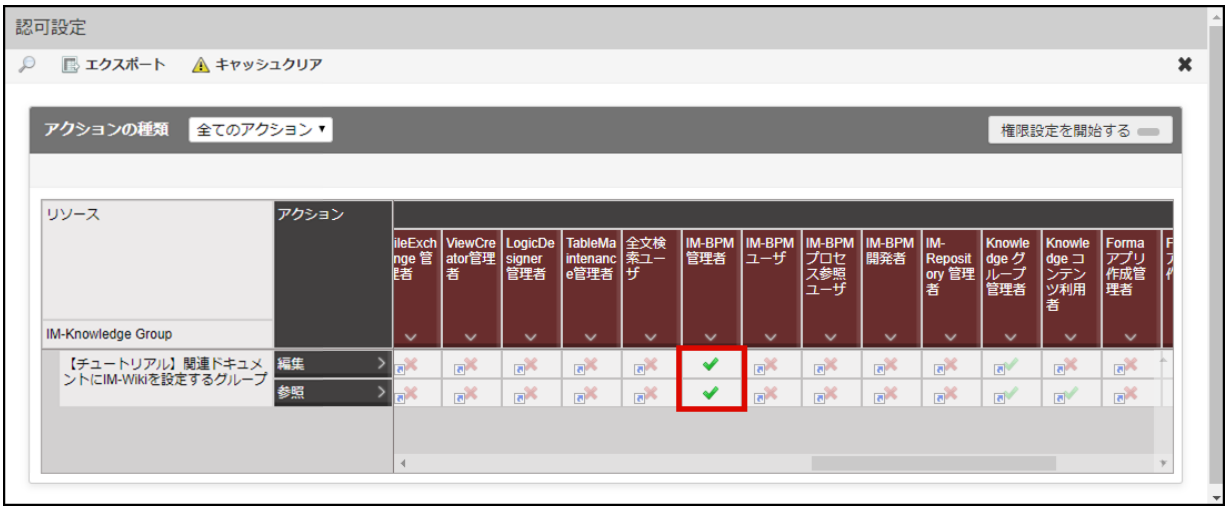
   「グループ一覧画面」で「 」アイコンをクリックし、「認可設定」ダイアログを表示します。



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。



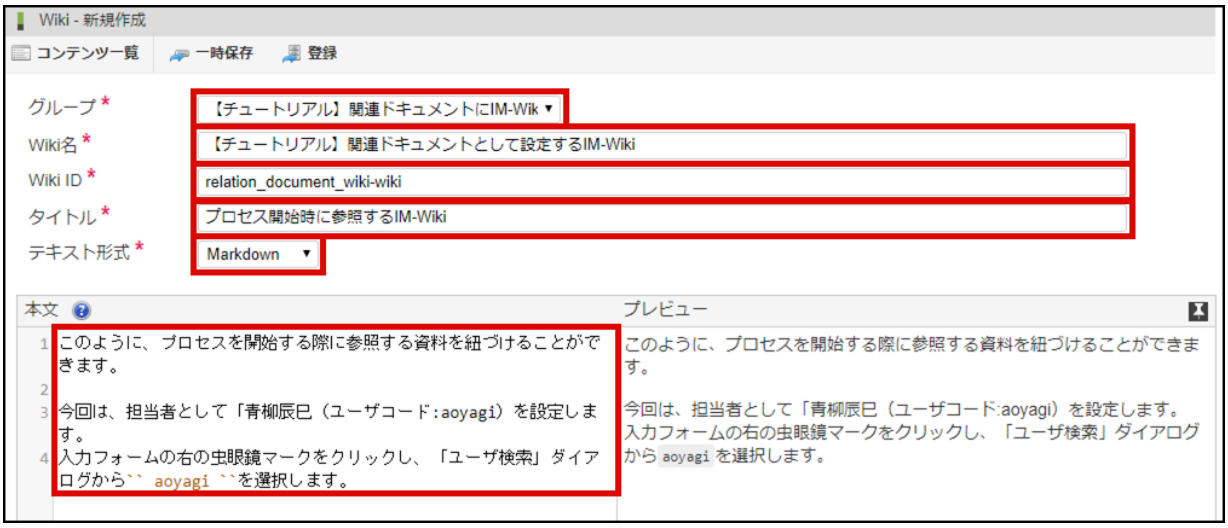
図：「認可設定」

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。
```



図：「IM-Wiki」 - 「新規作成」

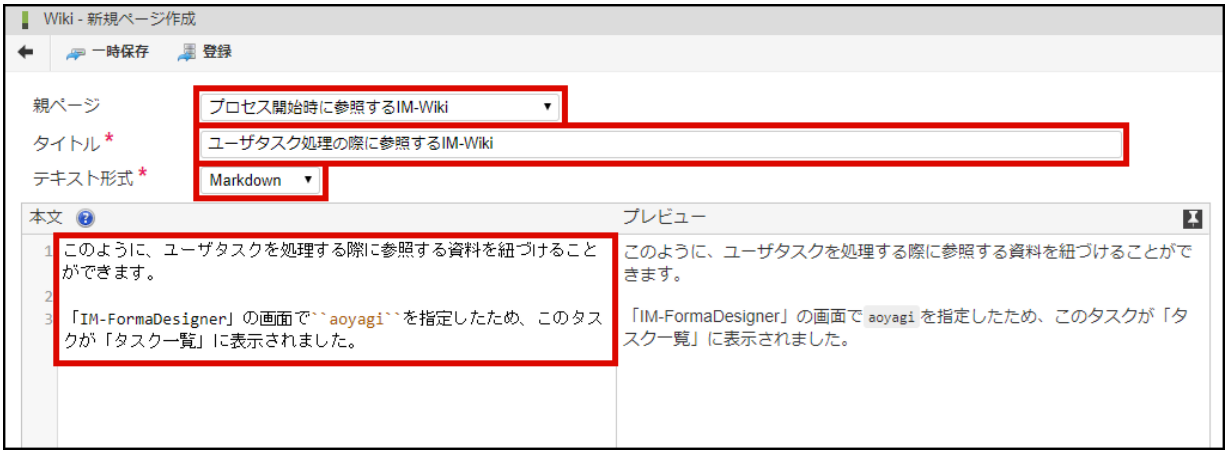
12. 入力後、「登録」をクリックします。

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

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

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

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

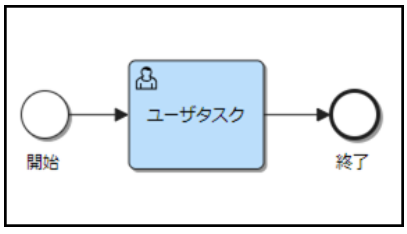


図：「IM-Wiki」 - 「新規ページ作成」

15. 入力後、「登録」をクリックします。

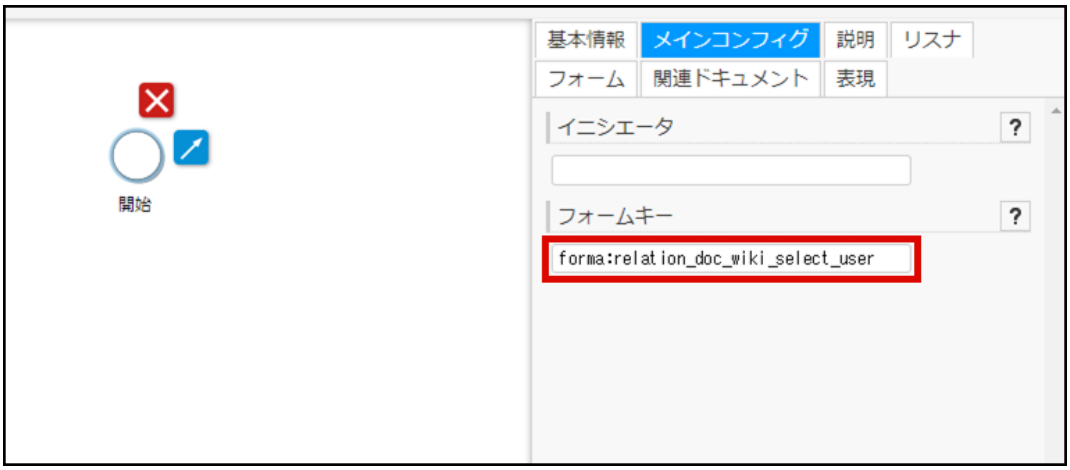
プロセス定義を作成する

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



図：完成イメージ

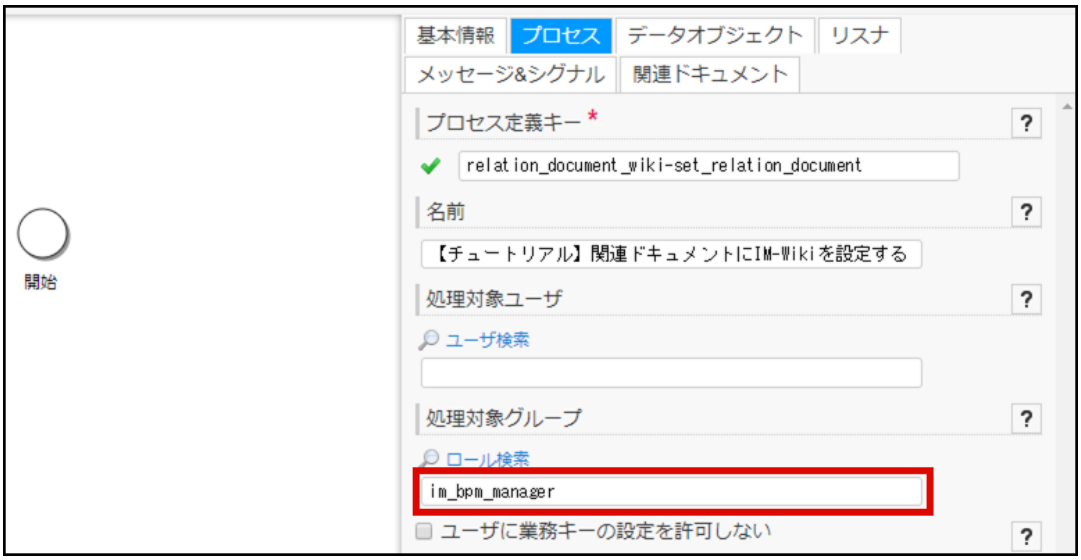
1. 「開始イベント」を配置します。
2. 「開始イベント」の「プロパティ」 - 「 メインコンフィグ」で項目を以下のように設定します。
    - フォームキー：forma:relation_doc_wiki_select_user



図：「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

3. プロセス全体に対する設定を行います。
   キャンバスの余白をクリックし、「プロパティ」をプロセス全体に切り替えます。

4. 「プロセス」で項目を以下のように設定します。
   - 処理対象グループ：im_bpm_manager



図：「プロパティ」 - 「プロセス」

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

図：「プロパティ」 - 「関連ドキュメント」

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

- 表示名：任意

図：「関連ドキュメント」

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

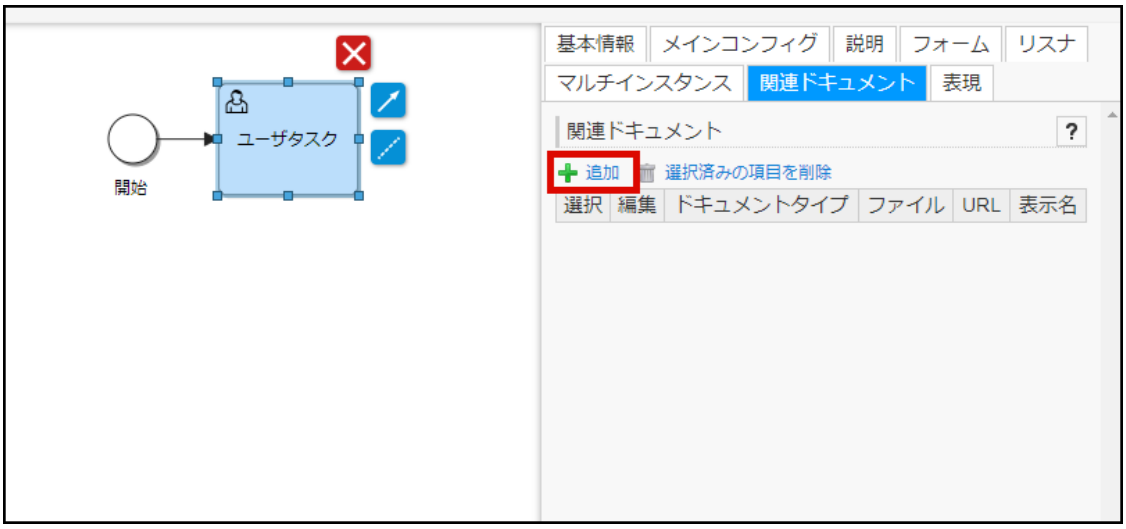
> **コラム**
>
> URLの「/wiki/」を「/wiki_view/」に置き換えることで、「編集」ボタンのないWikiを表示させることが可能です。
> 例：../imart/knowledge/contents/wiki_view/Wiki名

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

図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

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

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



図：「ユーザタスク」 - 「プロパティ」 - 「関連ドキュメント」

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

図：「関連ドキュメント」

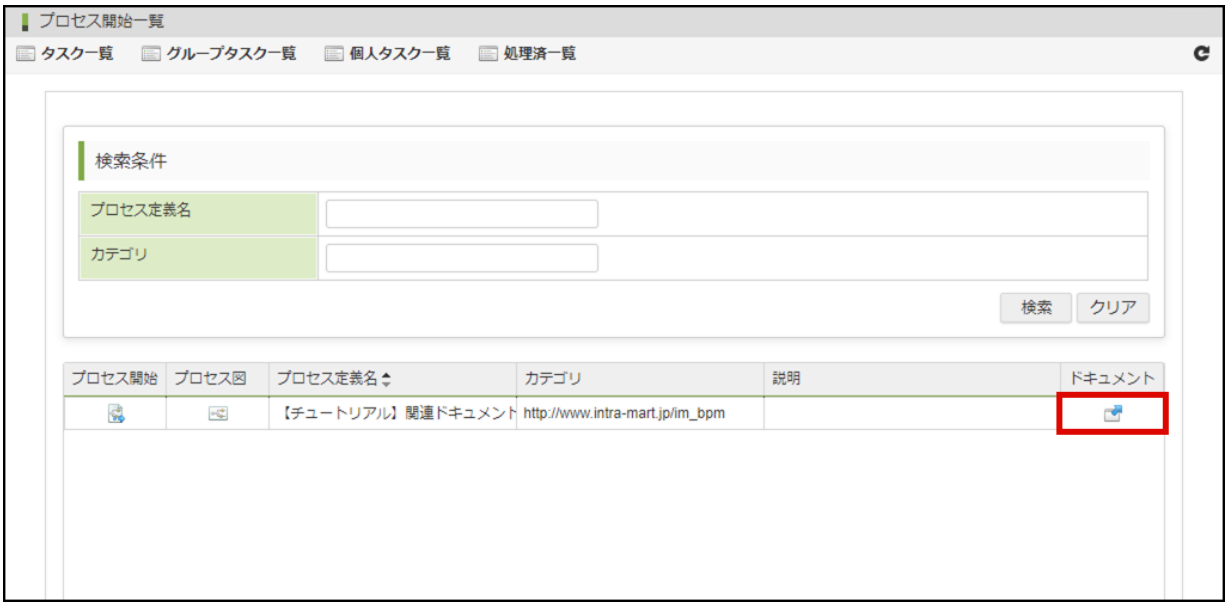
> **コラム**
>
> URLの最後に「 ?sidebar=false 」を書き加えることで、目次一覧が非表示のWikiを表示させることが可能です。
> 例：../imart/knowledge/contents/wiki_view/Wiki名 ?sidebar=false

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

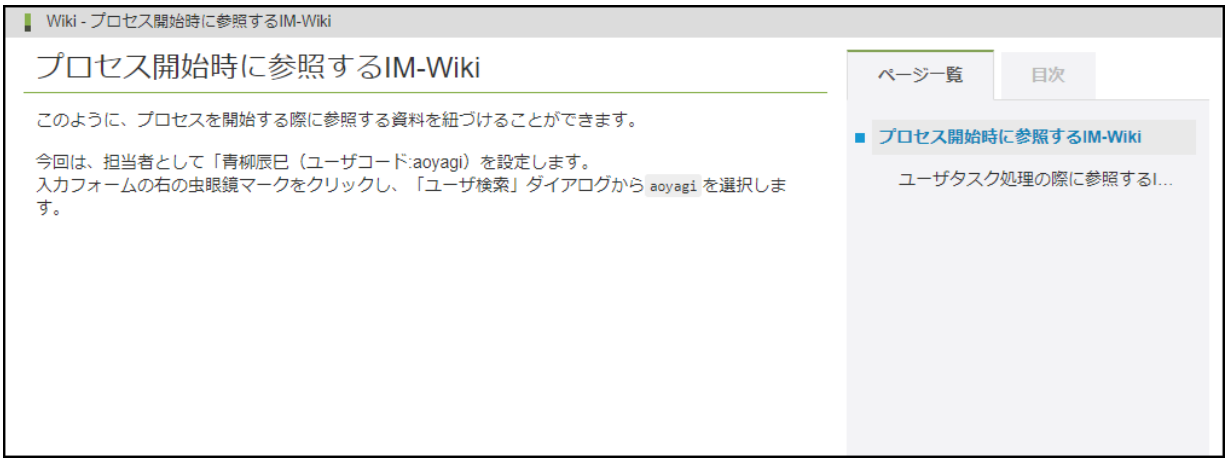
このチュートリアルで作成した「プロセス定義」を実行環境にデプロイし、実行結果の確認を行います。
「Knowledge コンテンツ利用者」のロールを付与した aoyagi でログインしてください。

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」

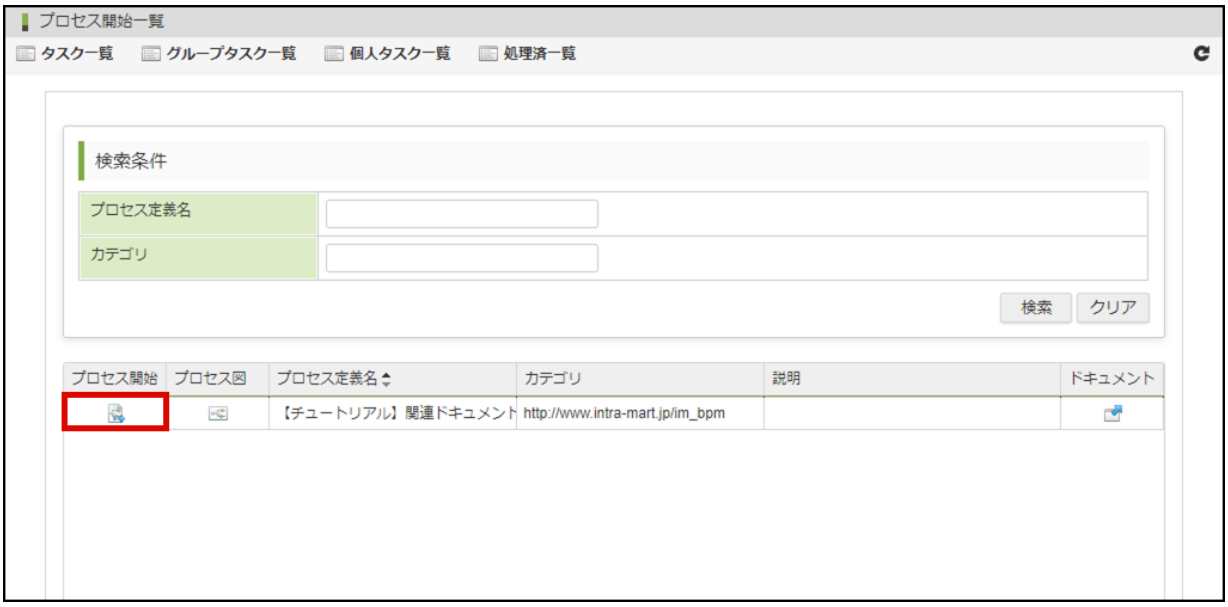
3. Wikiが新しいタブ、または、ウィンドウで表示されます。
   参照用のURL形式（「 /wiki_view/ 」）を利用しているため、ここからWikiの編集は行えません。



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」

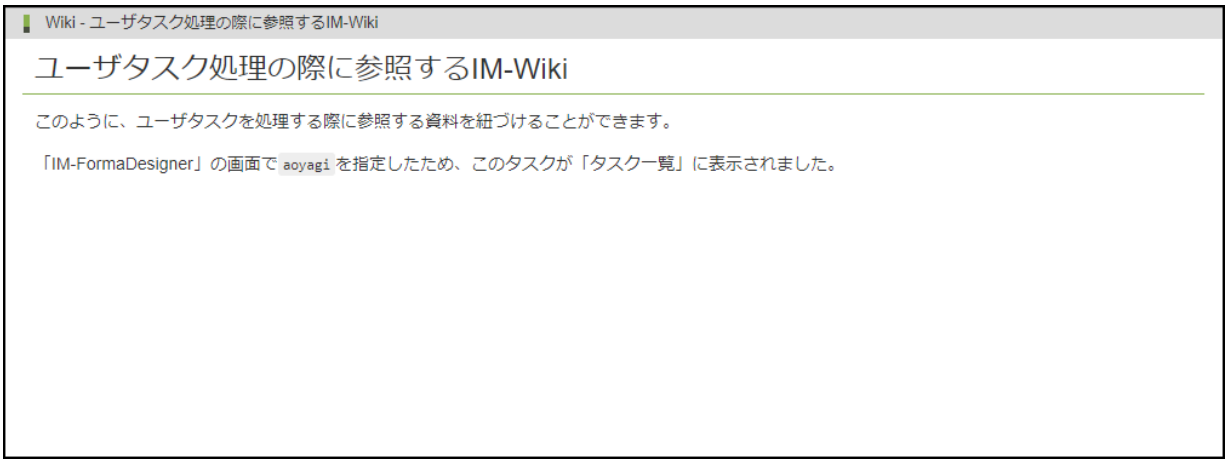
6. 個人タスクに、上記で実行したプロセスのタスクがあるので、「 」アイコンをクリックします。

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）

オプショナルタスク

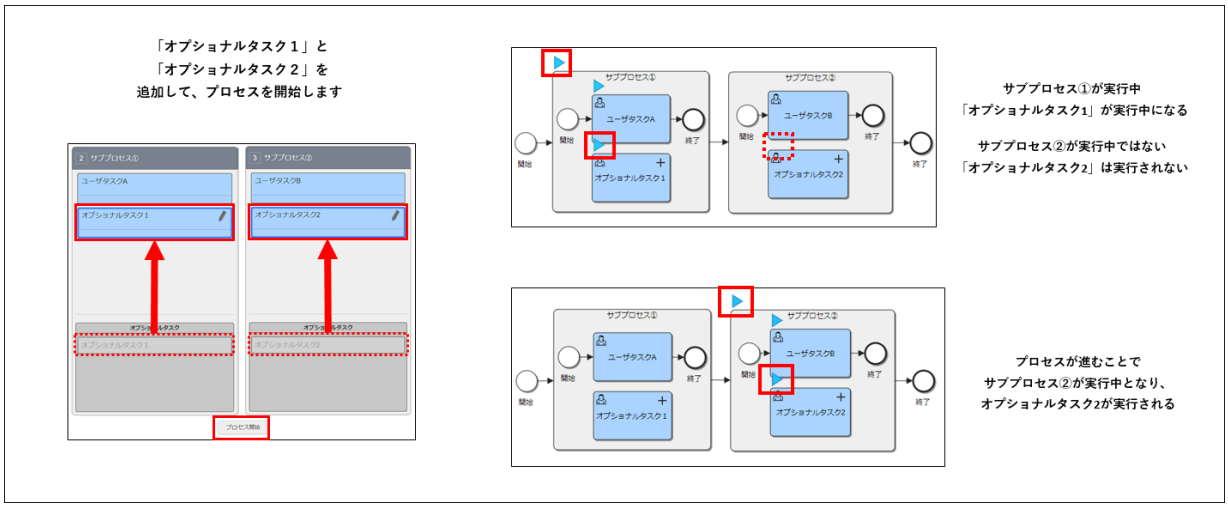
オプショナルタスクを追加する

このチュートリアルでは、オプショナル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。
オプショナルタスクは、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ナルタスクにできます。

- 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

オプショナルタスクの詳細については、「IM-BPM 仕様書」 - 「オプショナルタスク」 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。

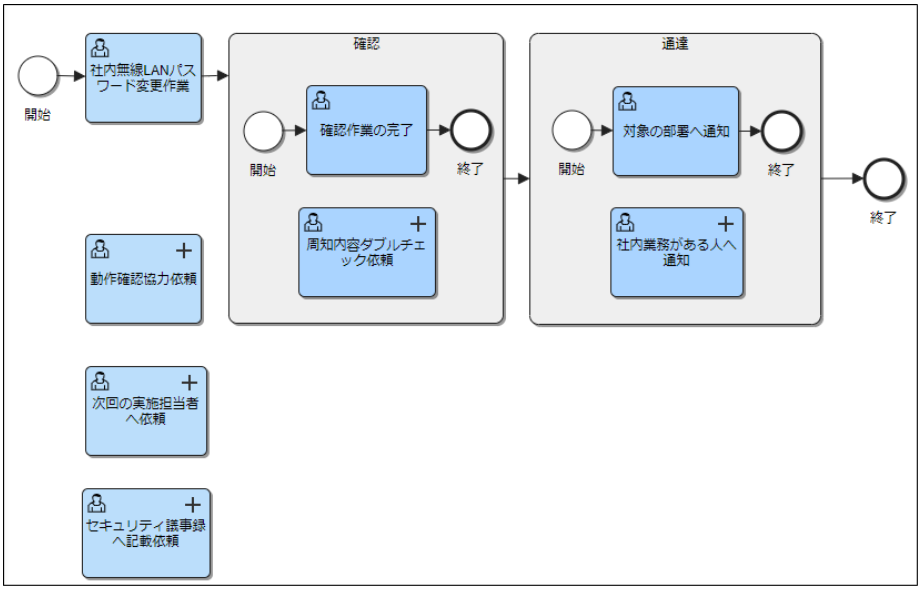
- プロセス定義を作成する
- 結果を確認する

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。

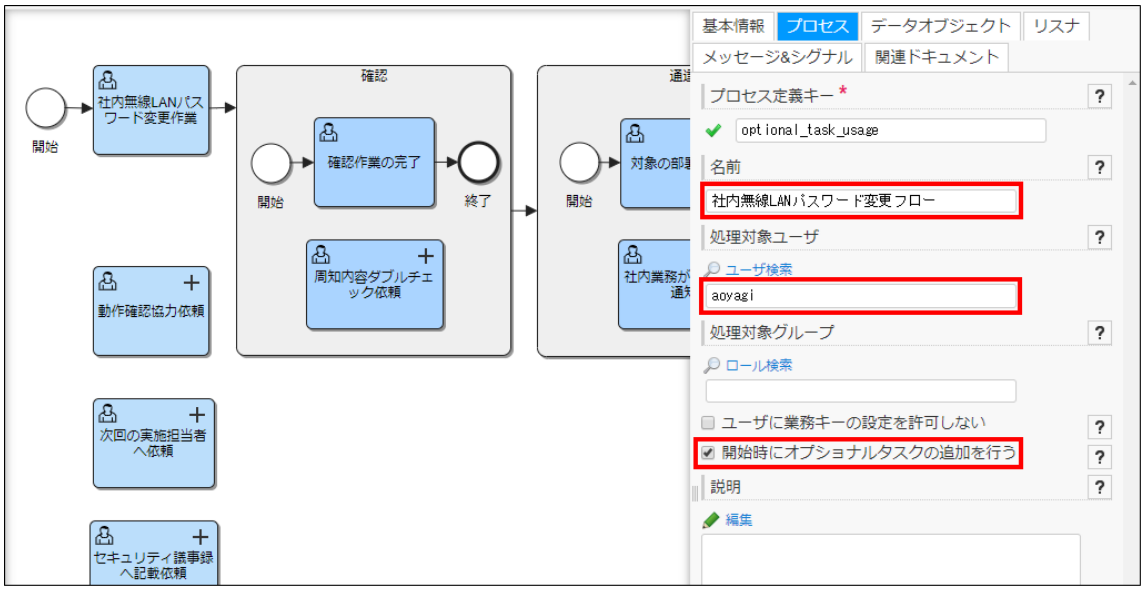
オプショナルタスクは、タスクの右上に「+」のマークが出ます。
オプショナルタスクはプロセスのどの段階でも追加・編集できますが、実行中になると編集できません。

- プロセス上のオプショナルタスク
  「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」タスクがオプショナルタスクです。
- 「確認」サブプロセス内のオプショナルタスク
  「周知内容ダブルチェック依頼」タスクがオプショナルタスクです。
  「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ナルタスク
  「社内業務がある人へ通知」タスクがオプショナルタスクです。
  「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

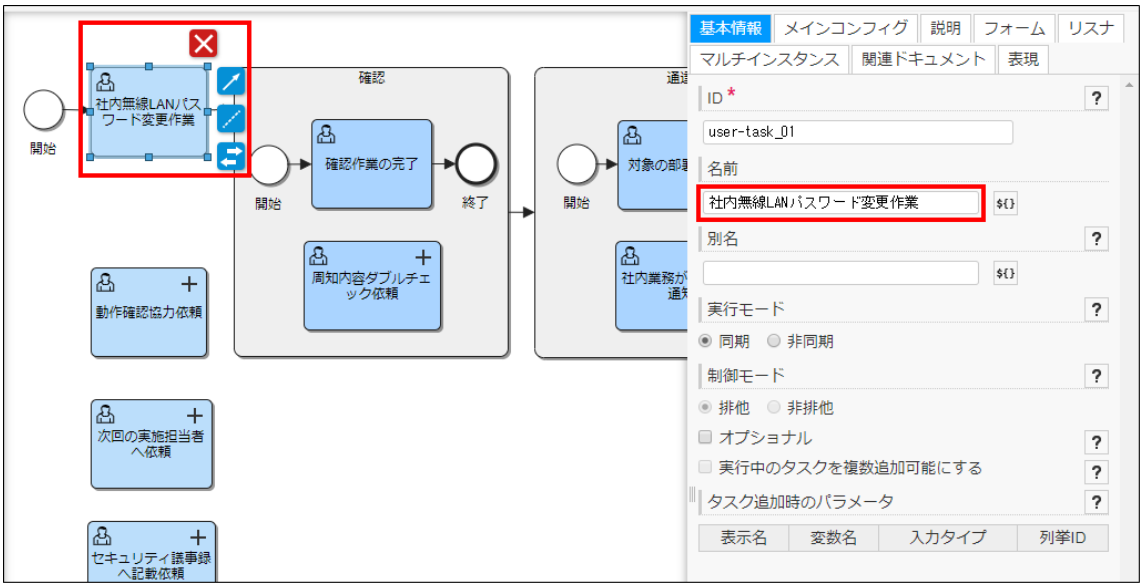
1. 「開始イベント」を設置します。
2. プロセスの処理対象ユーザと、オプショナルタスクを追加する画面を表示するよう設定をします。
   キャンパスの空白部分をクリックし、「プロパティ」をプロセス全体に切り替えます。
   プロセスタブから、以下のように項目を設定してください。
   - 名前：社内無線LANパスワード変更フロー
   - 処理対象ユーザ：aoyagi
   - 開始時にオプショナルタスクの追加を行う：有効



図：プロセス全体 - 「プロパティ」 - 「プロセス」

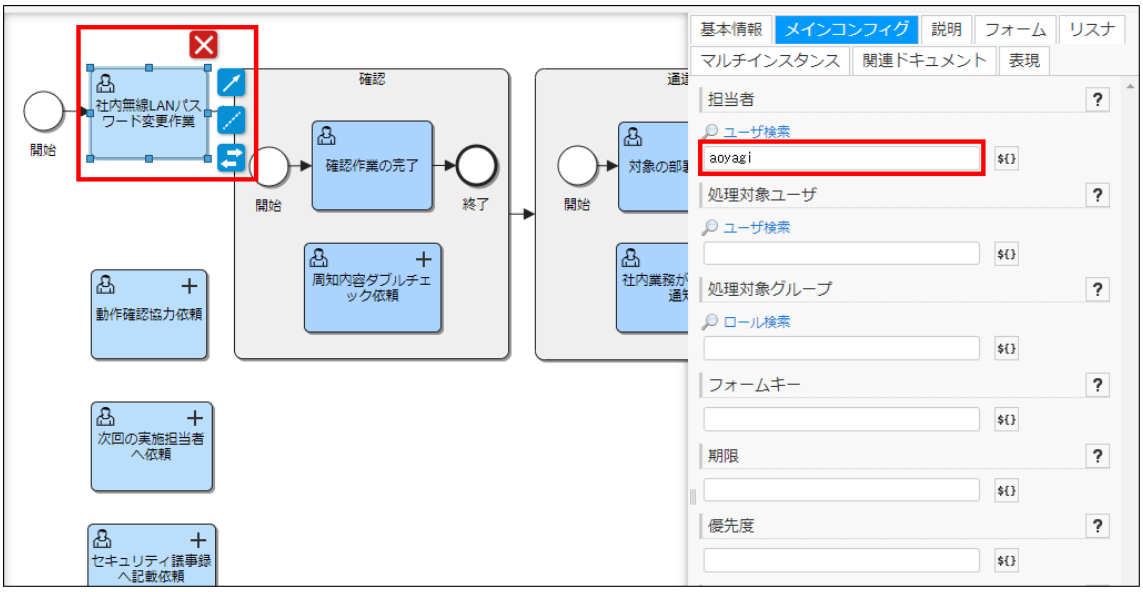
3. サブプロセス外からオプショナルタスクを追加できることを確認するため、ユーザタスクを配置します。
   「ユーザタスク」を設置します。

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」

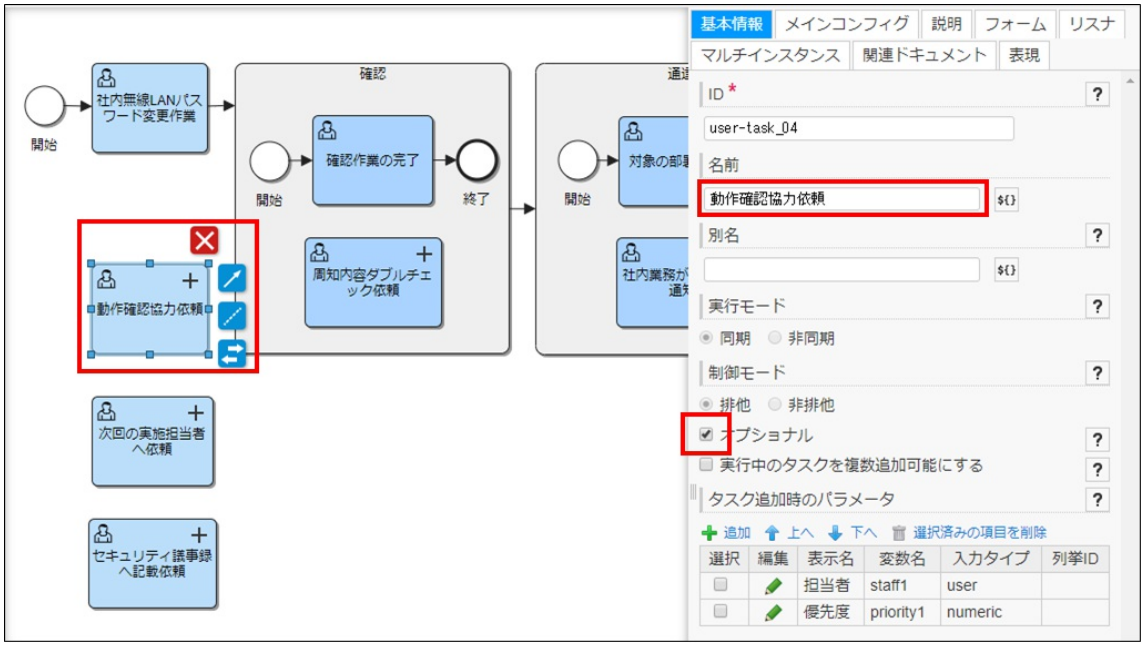
5. 「社内無線LANパスワード変更作業」タスクに対して、担当者を設定をします。
   「メインコンフィグ」タブから、「担当者」に `aoyagi` と設定してください。



図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

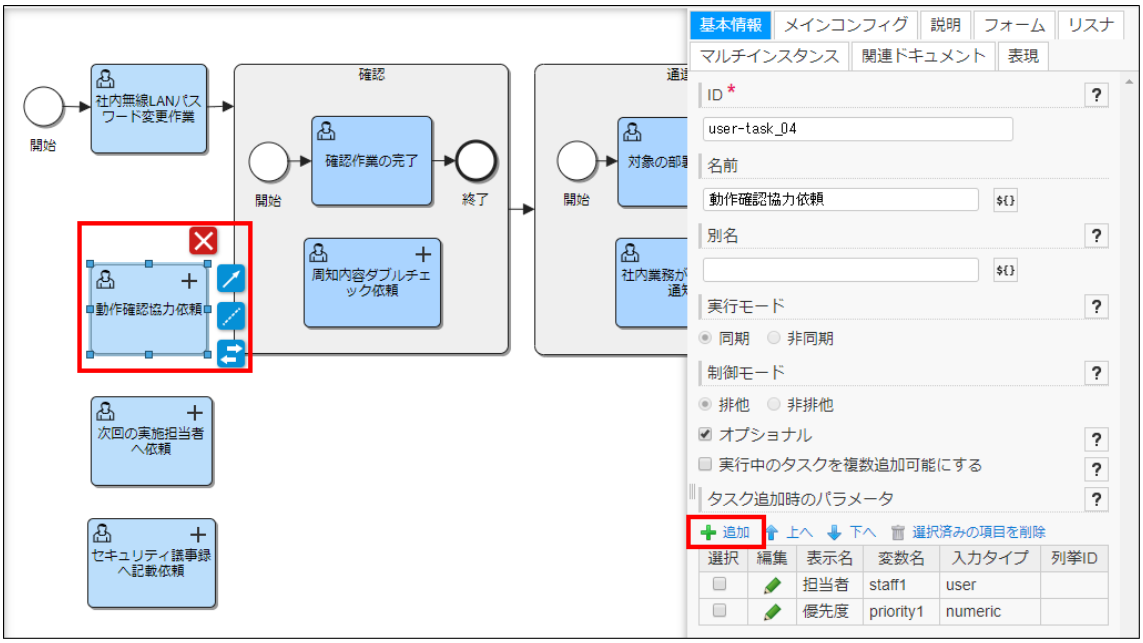
6. プロセス直下に、1つ目のオプショナルタスクを追加します。
   「ユーザタスク」を設置します。

7. 設置したユーザタスクをオプショナルタスクにします。
   「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
   - オプショナル：有効



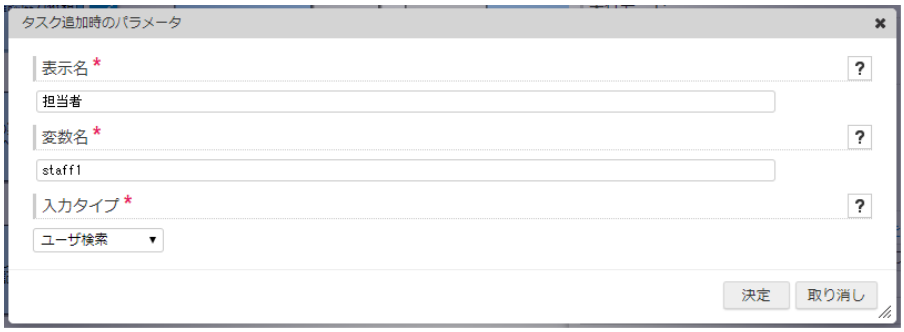
図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

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」

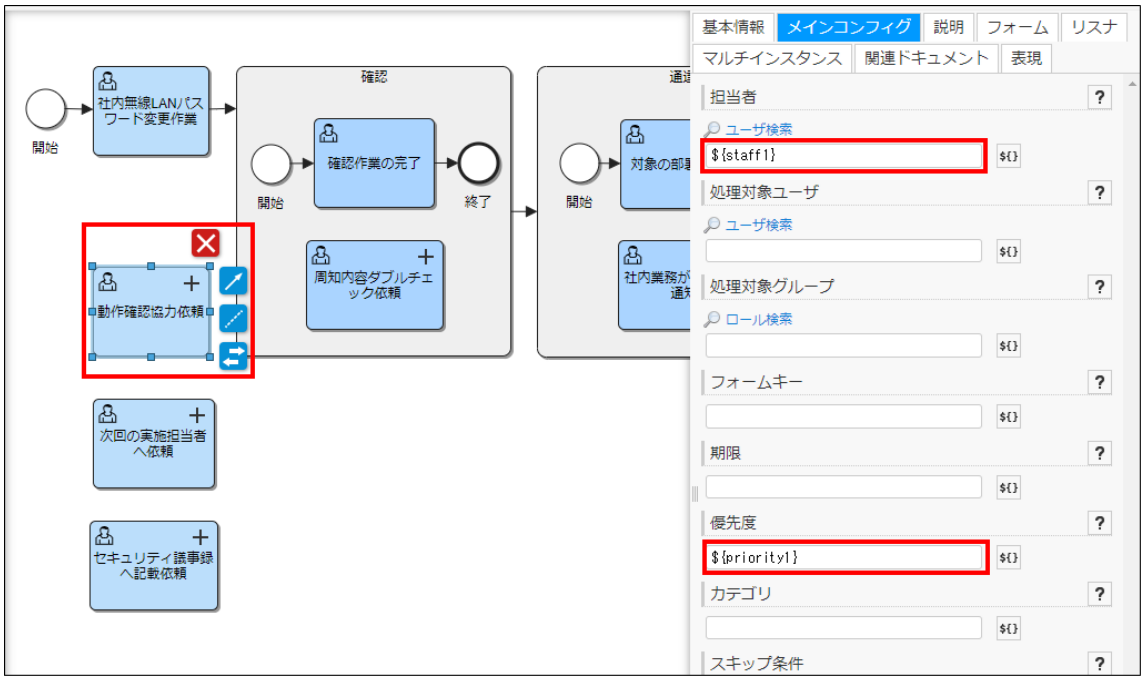
9. 「タスク追加時のパラメータ」ダイアログが表示されます。
以下のように、2つのパラメータを設定してください。
   - 担当者
     表示名：担当者
     変数名：staff1
     入力タイプ：ユーザ検索
   - 優先度
     表示名：優先度
     変数名：priority1
     入力タイプ：数字



図：「タスク追加時のパラメータ」

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

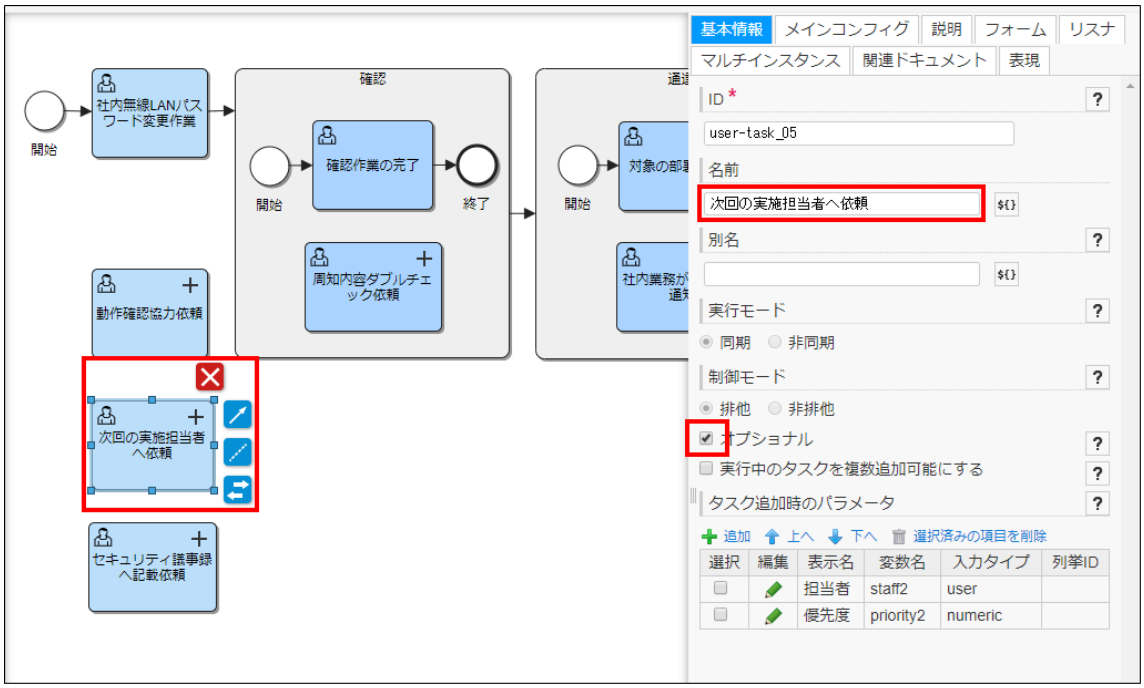
10. EL式を使用し、指定されたパラメータが各項目に設定されるようにします。
「ユーザタスク」をクリックし、「メインコンフィグ」タブから以下の項目を設定します。
   - 担当者：`${staff1}`
   - 優先度：`${priority1}`



図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

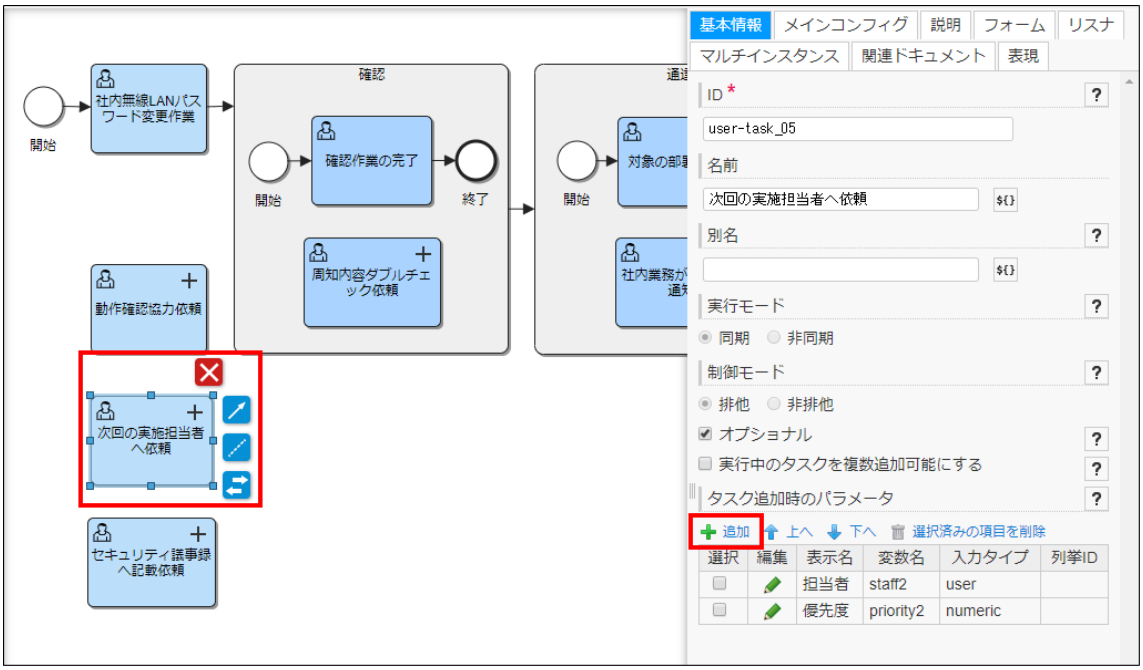
11. 上記と同じ手順で、2つ目のユーザタスクを追加します。
「ユーザタスク」を設置します。

12. 設置したユーザタスクをオプショナルタスクにします。
「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
    - オプショナル：有効



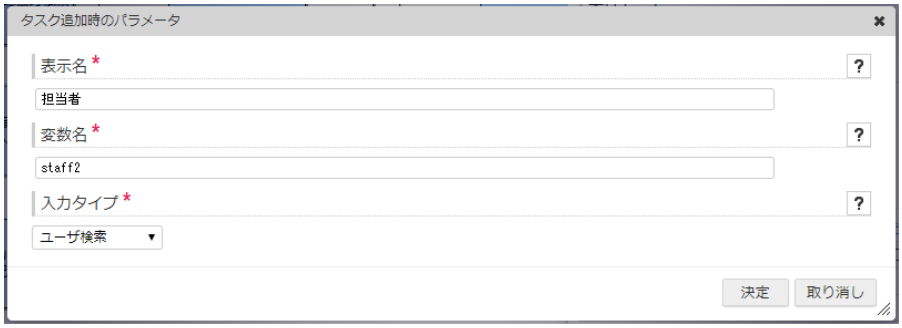
図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

13. オプショナル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。



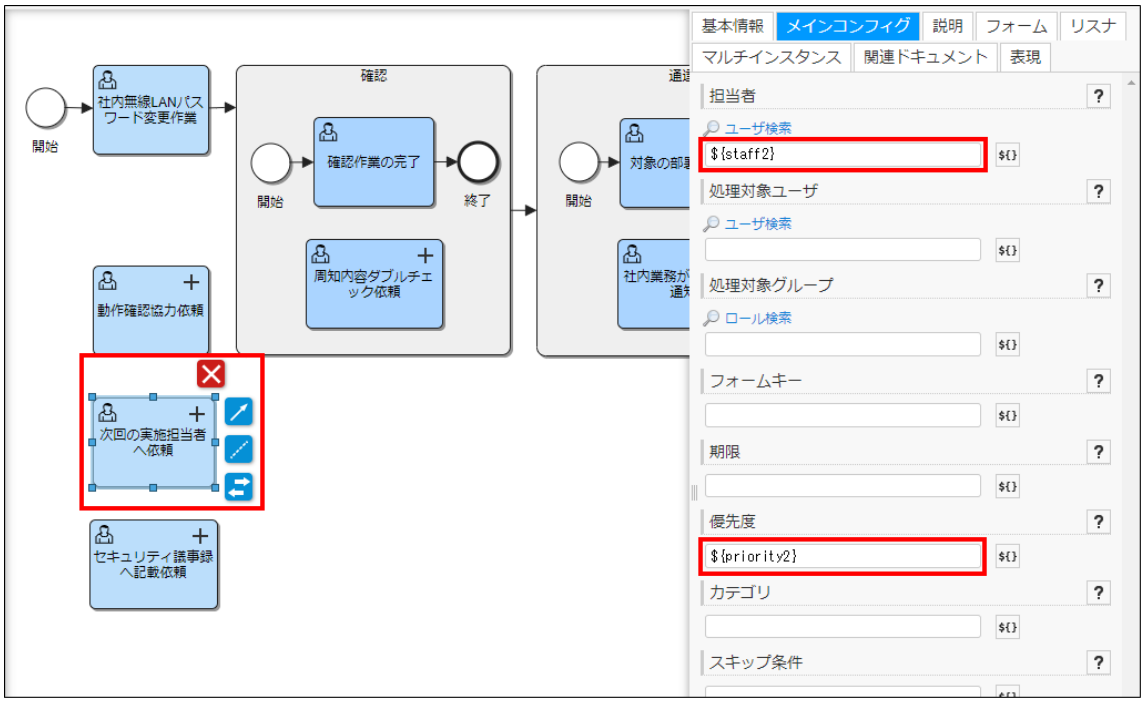
図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

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」

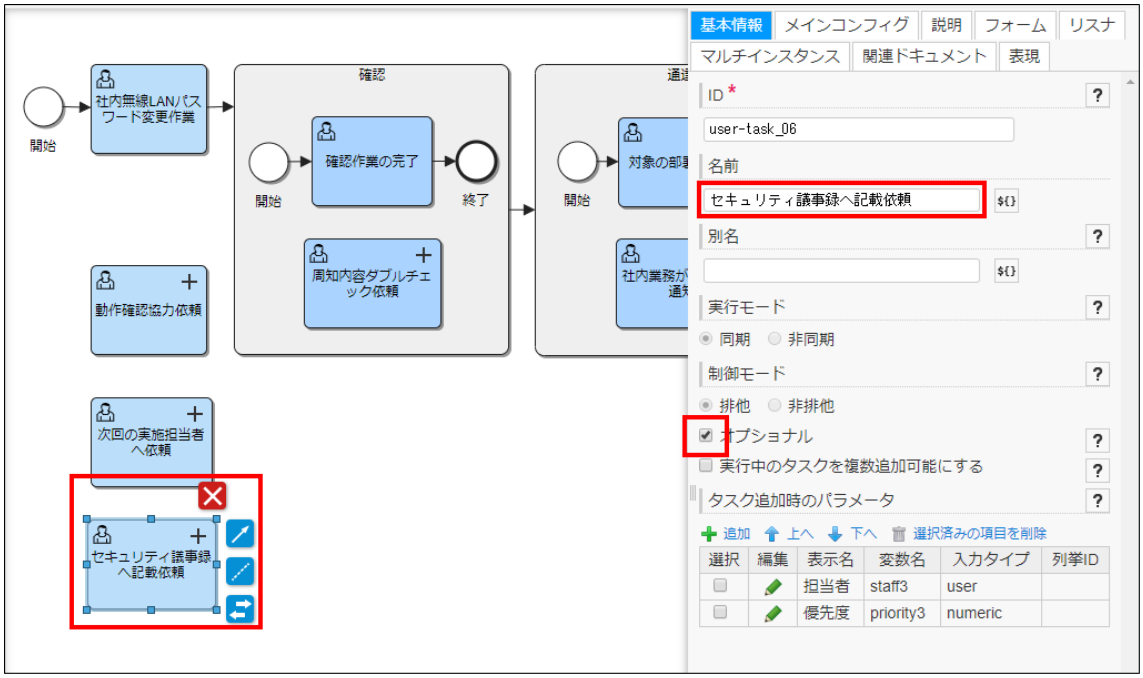
15. EL式を使用し、指定されたパラメータが各項目に設定されるようにします。
「ユーザタスク」をクリックし、「メインコンフィグ」タブから以下の項目を設定します。
    - 担当者：${staff2}
    - 優先度：${priority2}



図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

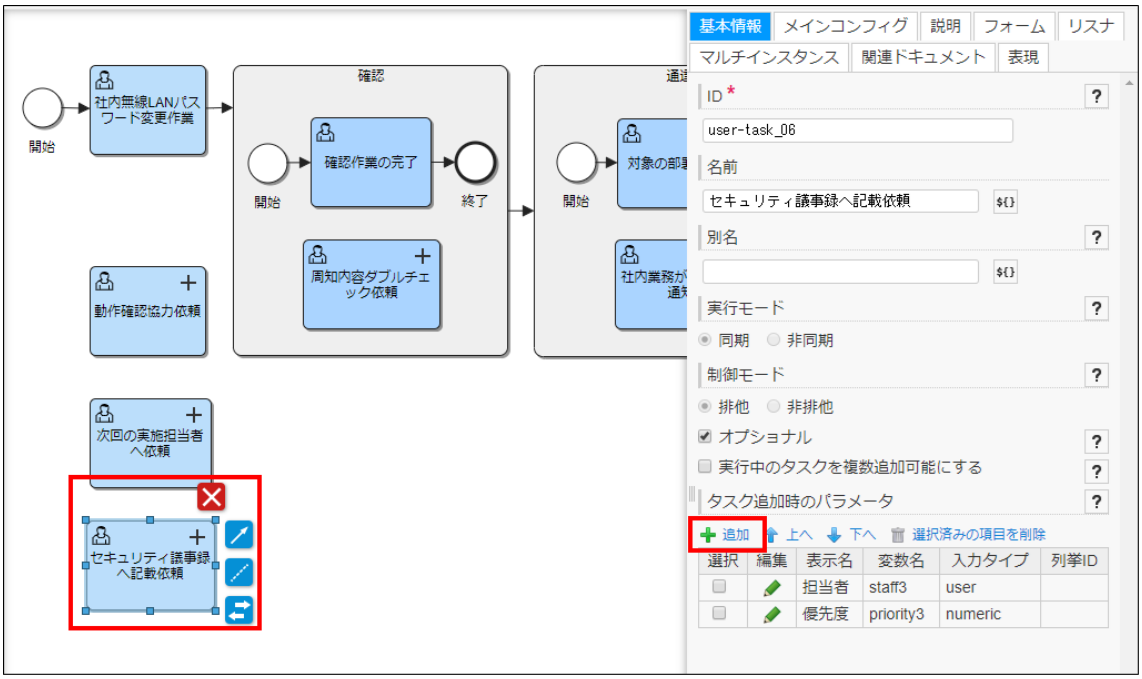
16. 上記と同じ手順で、3つ目のユーザタスクを追加します。
「ユーザタスク」を設置します。

17. 設置したユーザタスクをオプショナルタスクにします。
「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
    - オプショナル：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

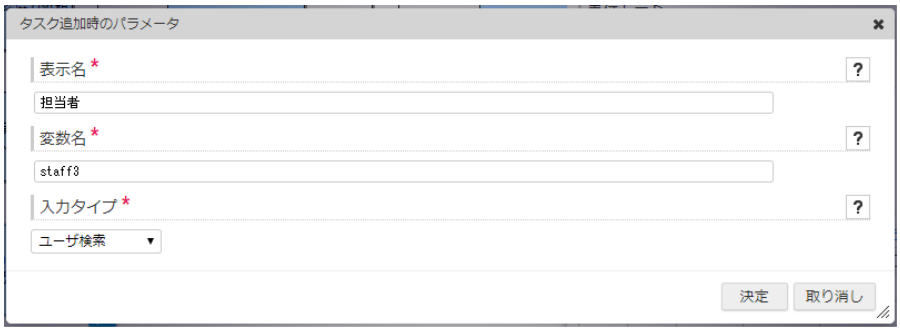
18. 「セキュリティ議事録へ記載依頼」タスクを追加する際に、指定できるパラメータの設定をします。
「基本情報」タブから「タスク追加時のパラメータ」の「追加」リンクをクリックしてください。



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。

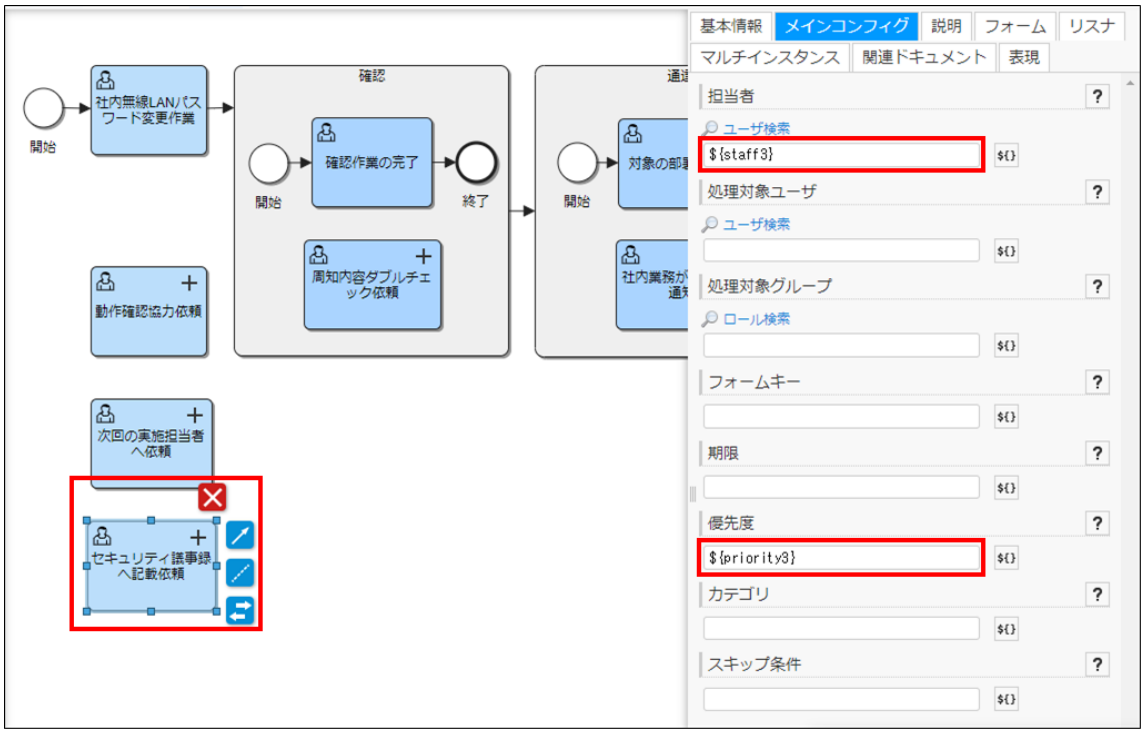
- 担当者
表示名：担当者
変数名：staff3
入力タイプ：ユーザ検索
- 優先度
表示名：優先度
変数名：priority3
入力タイプ：数字



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}



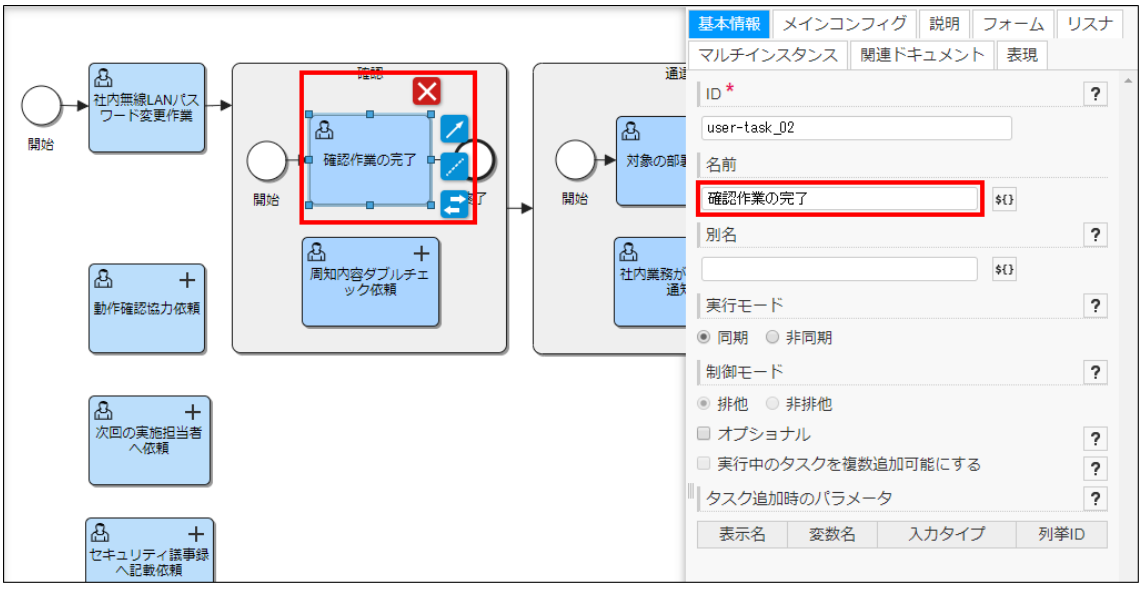
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

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。

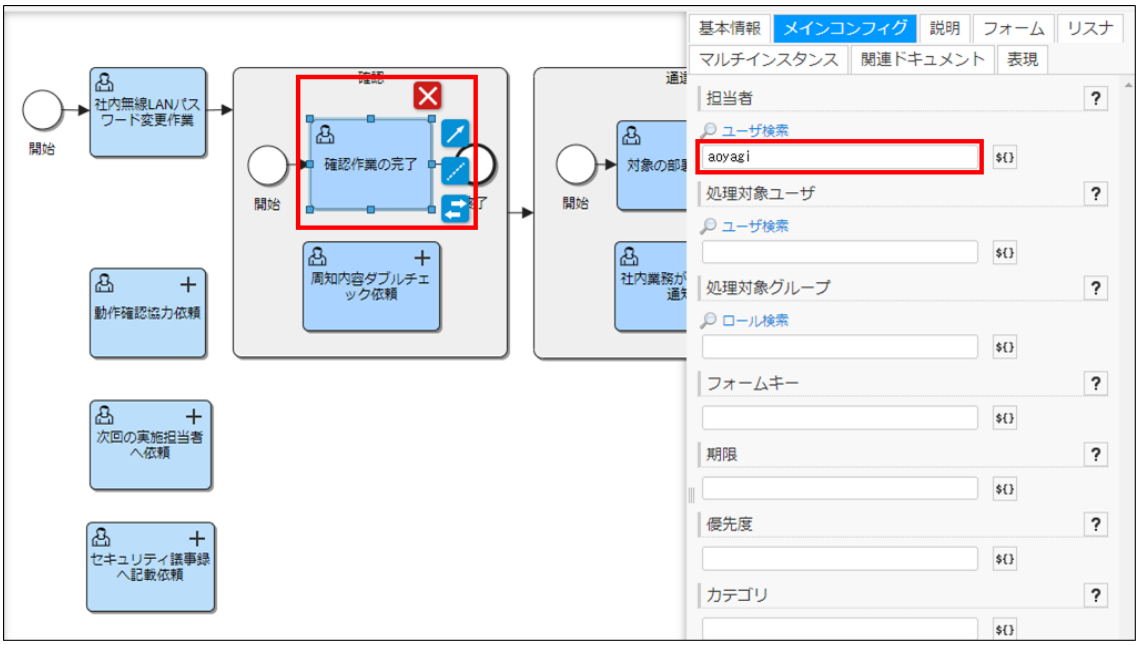
23. サブプロセスに「ユーザタスク」を設置します。

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。

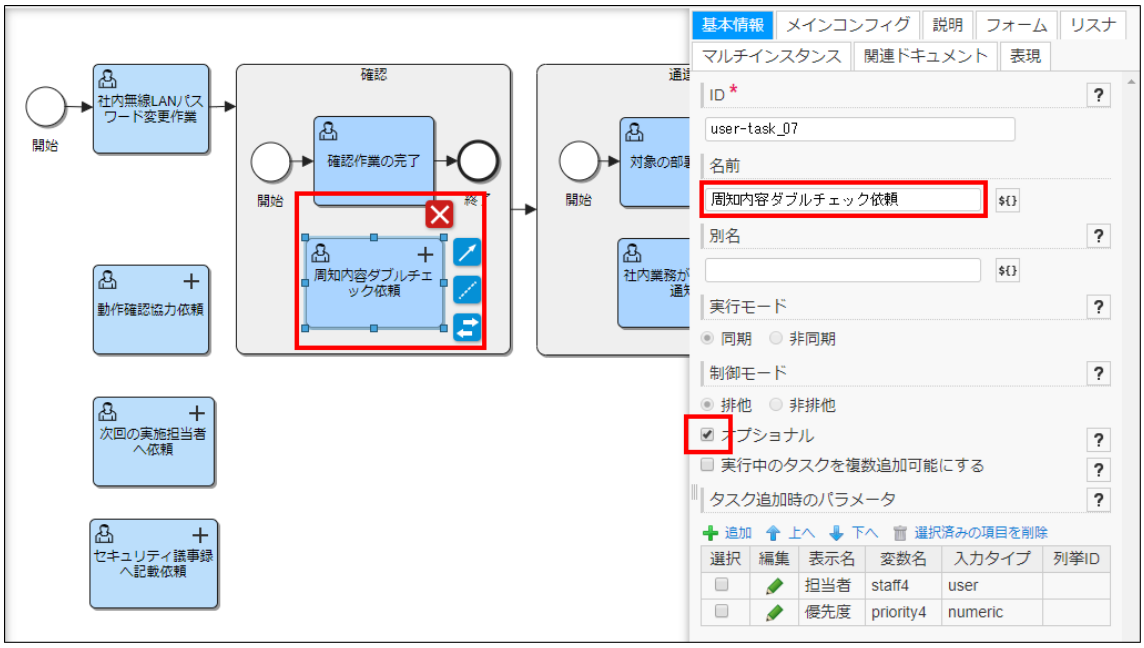


図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

26. 「終了イベント」を設置します。

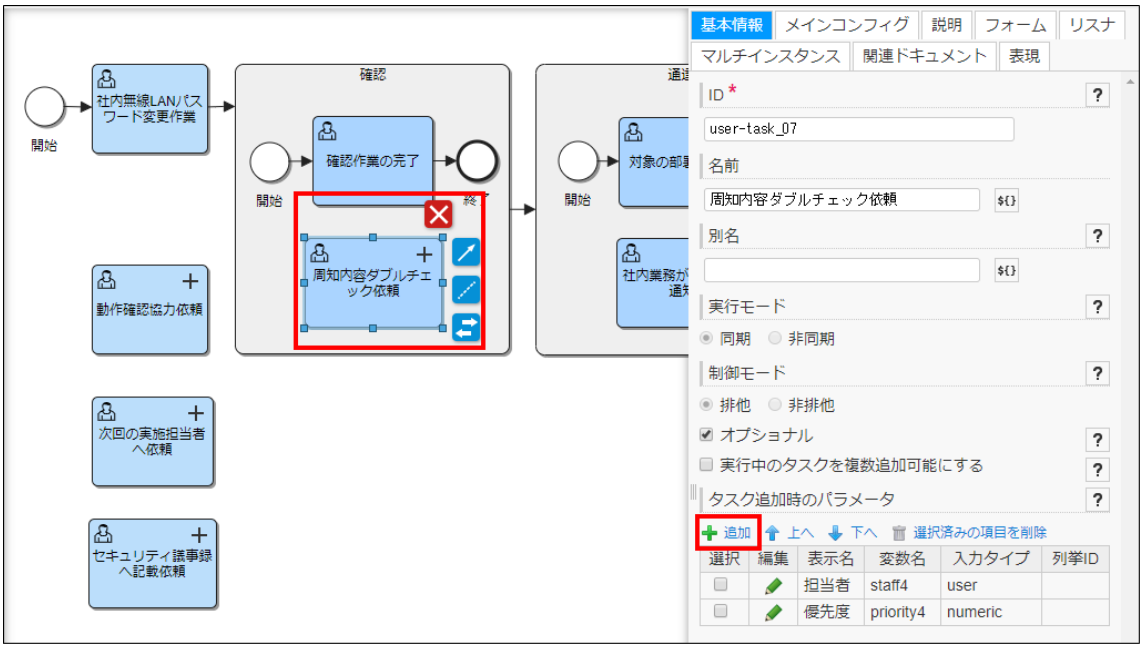
27. サブプロセス内にオプショナルタスクを設置します。
「ユーザタスク」を設置します。

28. 設置したユーザタスクをオプショナルタスクにします。
「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
    - オプショナル：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

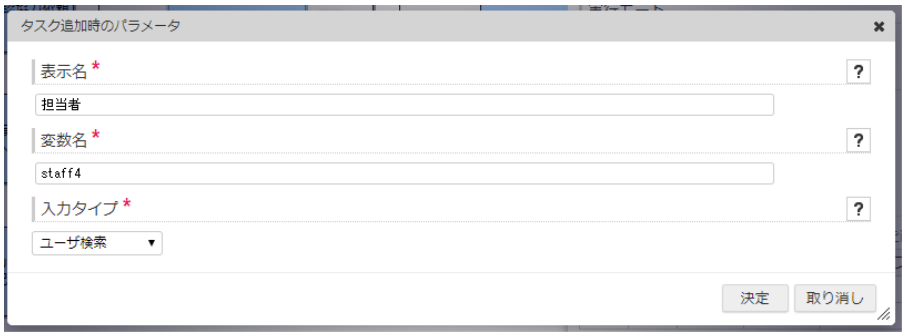
29. 「周知内容ダブルチェック依頼」タスクを追加する際に、指定できるパラメータの設定をします。
「ユーザタスク」をクリックし、「基本情報」タブから「タスク追加時のパラメータ」の「追加」リンクをクリックします。



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。

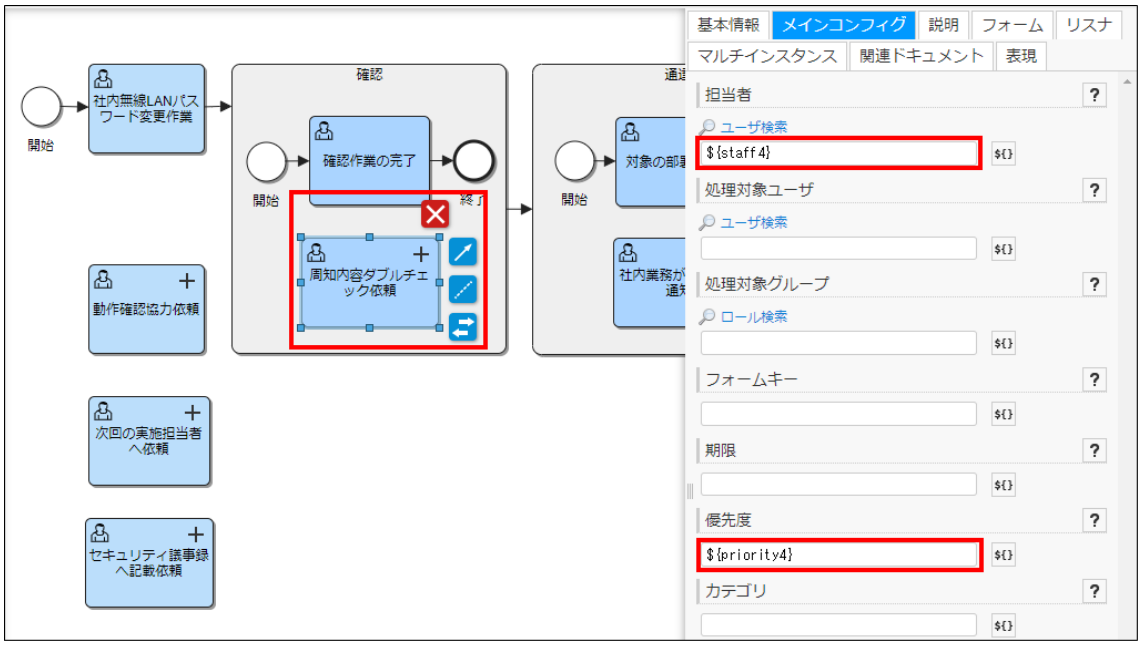
- 担当者
  表示名：担当者
  変数名：staff4
  入力タイプ：ユーザ検索
- 優先度
  表示名：優先度
  変数名：priority4
  入力タイプ：数字



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}



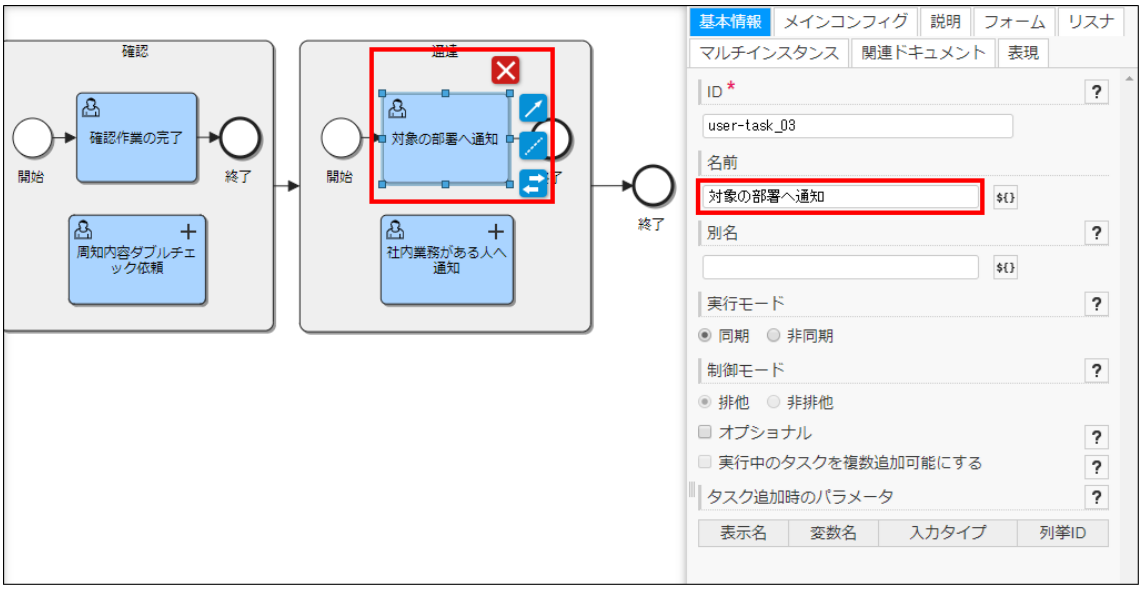
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

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。

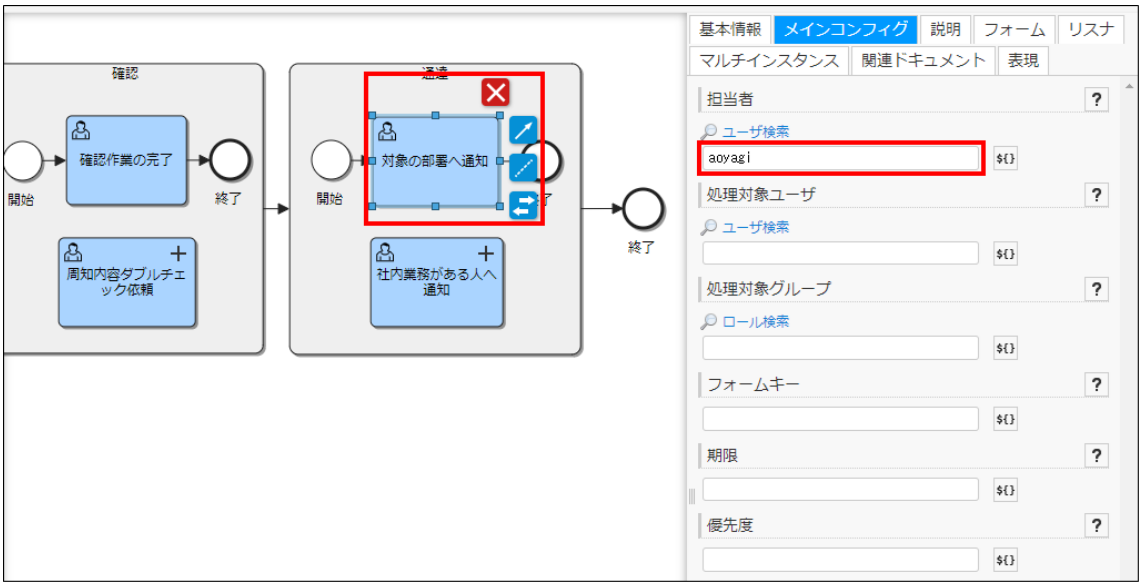
34. サブプロセスに「ユーザタスク」を設置します。

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。

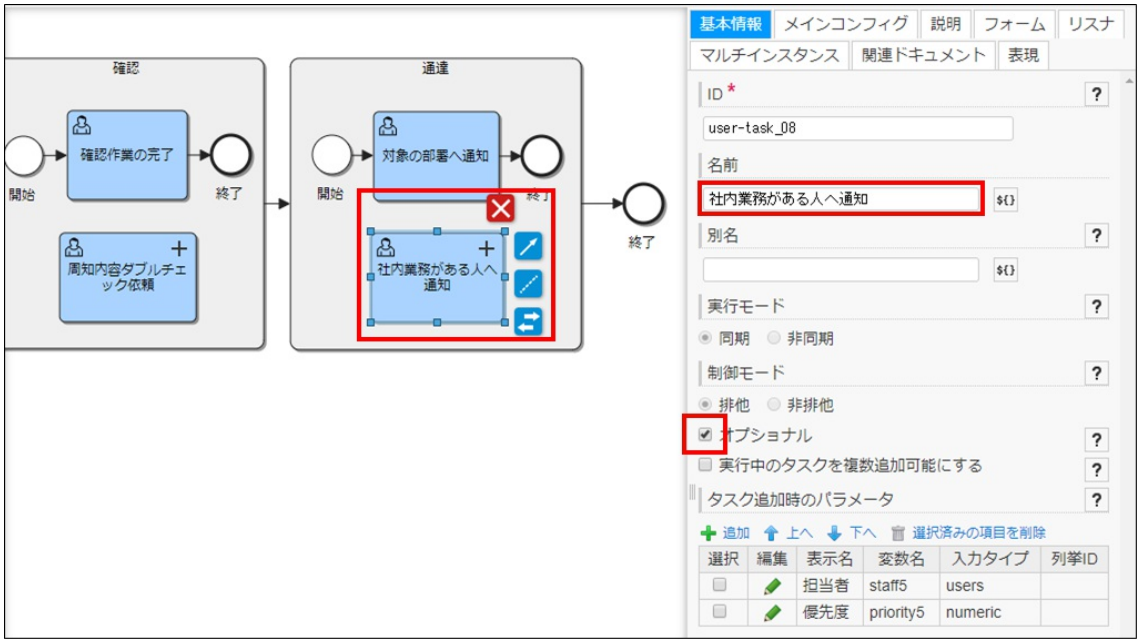


図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

37. 「終了イベント」を設置します。

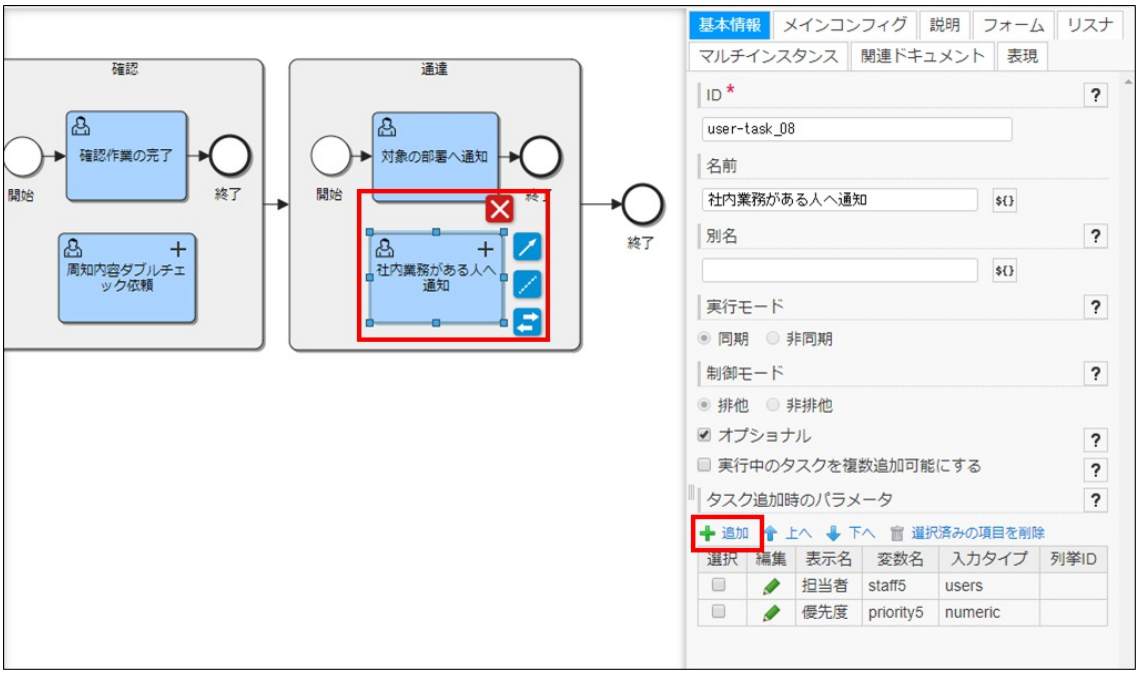
38. サブプロセス内にオプショナルタスクを設置します。
「ユーザタスク」を設置します。

39. 設置したユーザタスクをオプショナルタスクにします。
「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
    - オプショナル：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

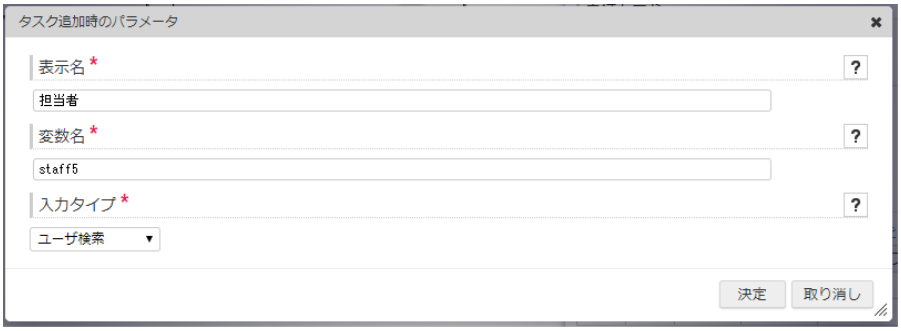
40. 「社内業務がある人へ通知」タスクを追加する際に、指定できるパラメータの設定をします。
「ユーザタスク」をクリックし、「基本情報」タブから「タスク追加時のパラメータ」の「追加」リンクをクリックします。



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。

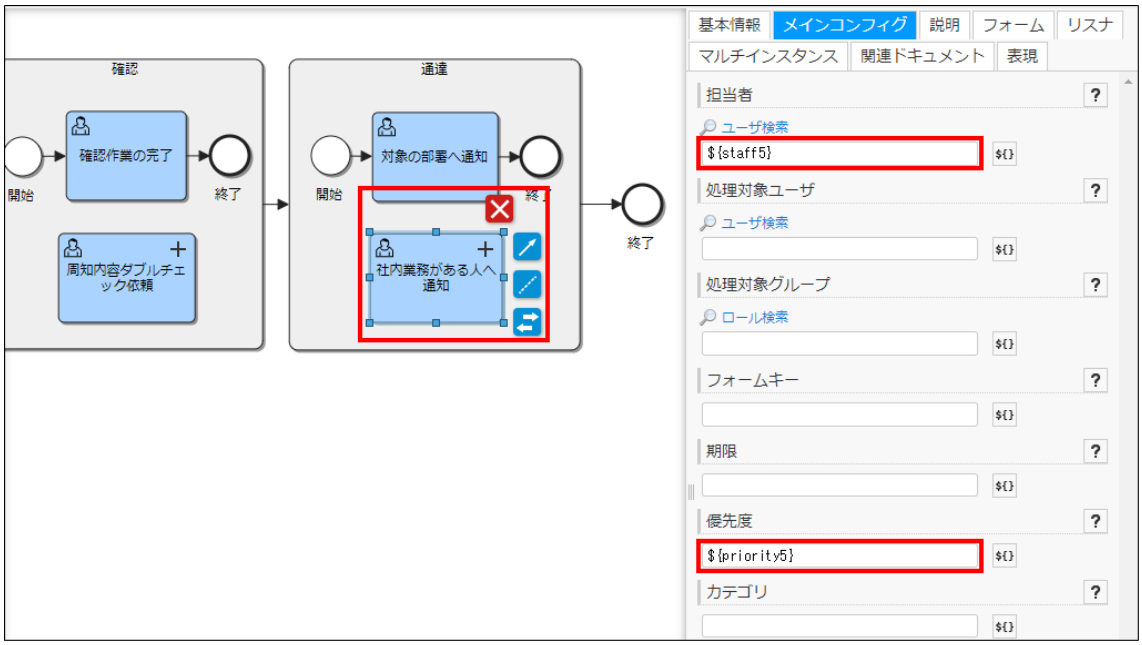
- 担当者
表示名：担当者
変数名：staff5
入力タイプ：ユーザ検索
- 優先度
表示名：優先度
変数名：priority5
入力タイプ：数字



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」

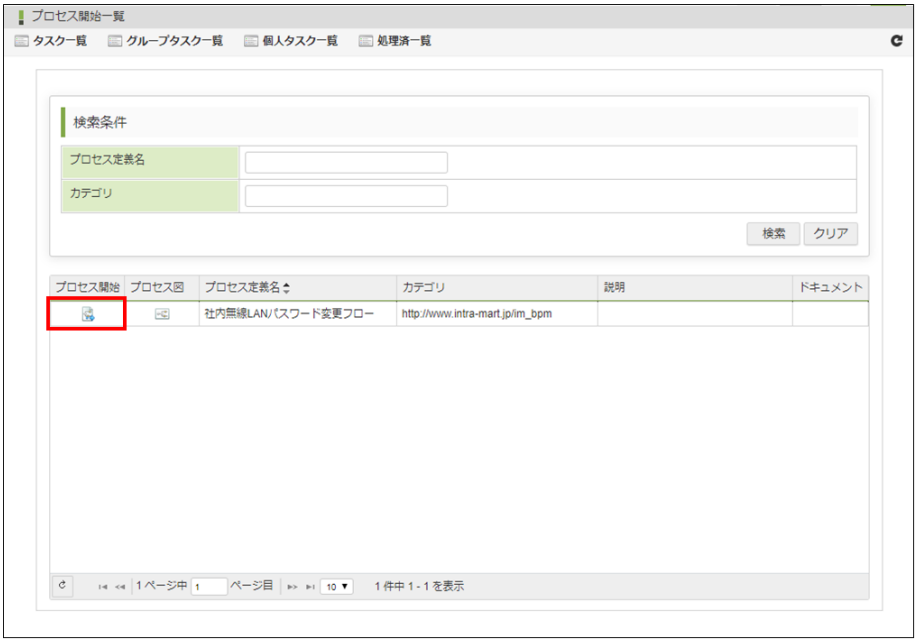
43. 「終了イベント」を設置します。

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。

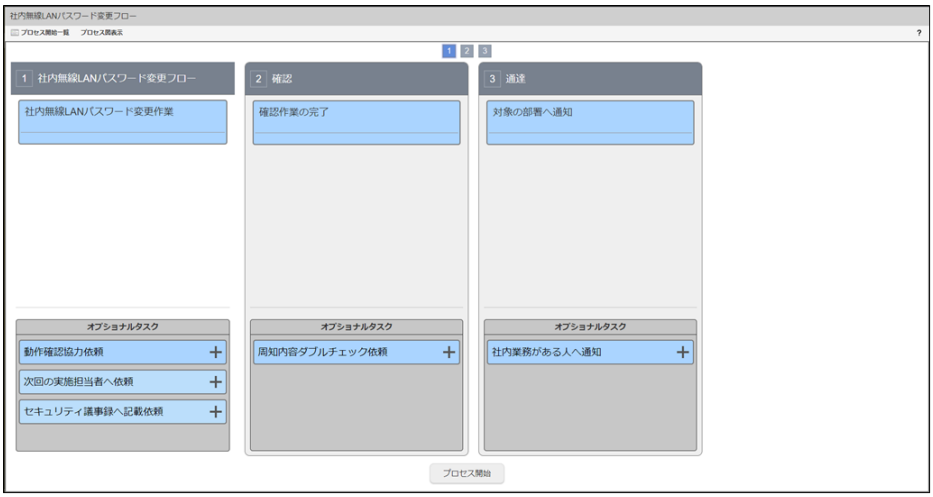
1. プロセスを開始します。
   「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示します。

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」、たまは「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ナルタスクは、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。

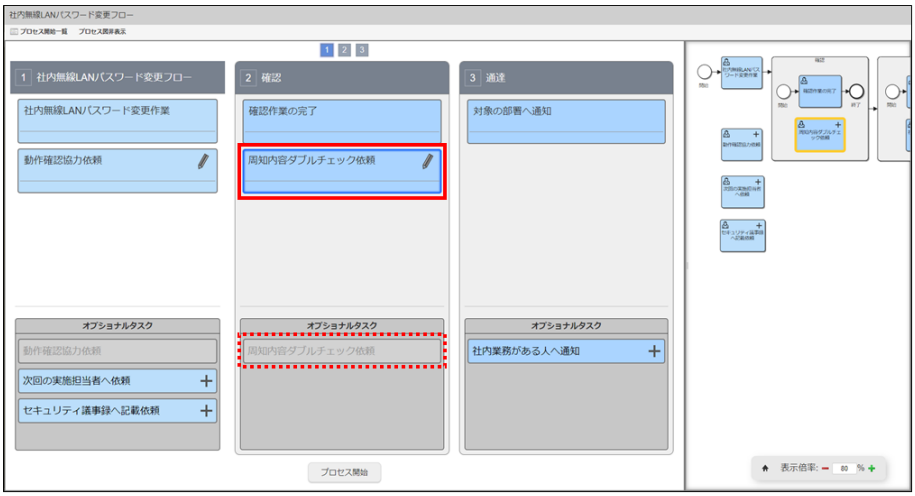
7. 「確認」サブプロセスに対して、オプショナルタスクを追加します。
「確認」サブプロセスのオプショナルタスク一覧から、「周知内容ダブルチェック依頼」をクリックします。

図：「タスク追加」

8. 「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

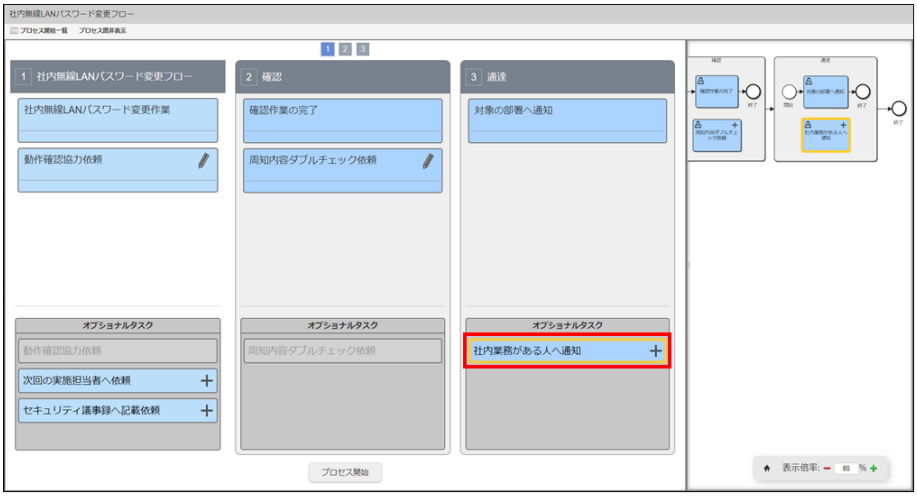
図：「タスクの追加」

9. 「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、オプショナルタスク一覧にある「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、オプショナルタスクを追加します。
「通達」サブプロセスのオプショナルタスク一覧から、「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

図：「タスクの追加」

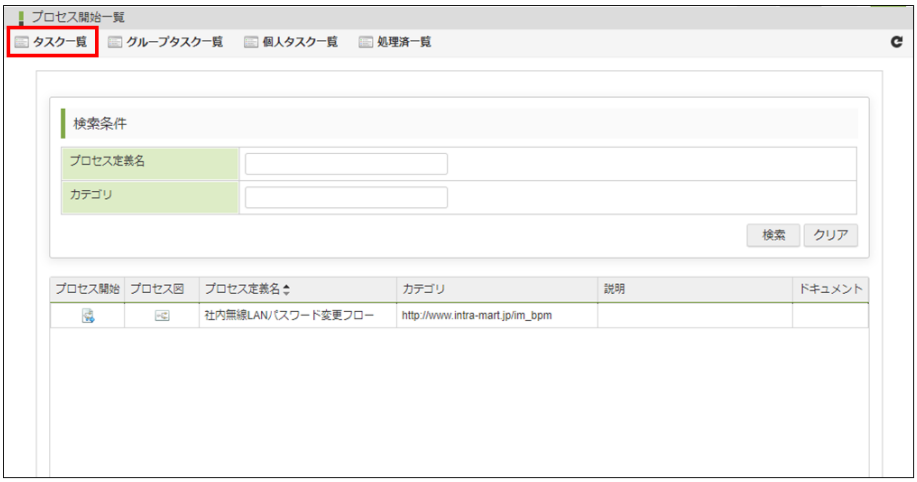
12. 「社内業務がある人への通知」のタスクが追加されたことを確認します。
また、オプショナルタスク一覧にある「社内業務がある人への通知」タスクがグレーアウトしていることを確認します。

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。

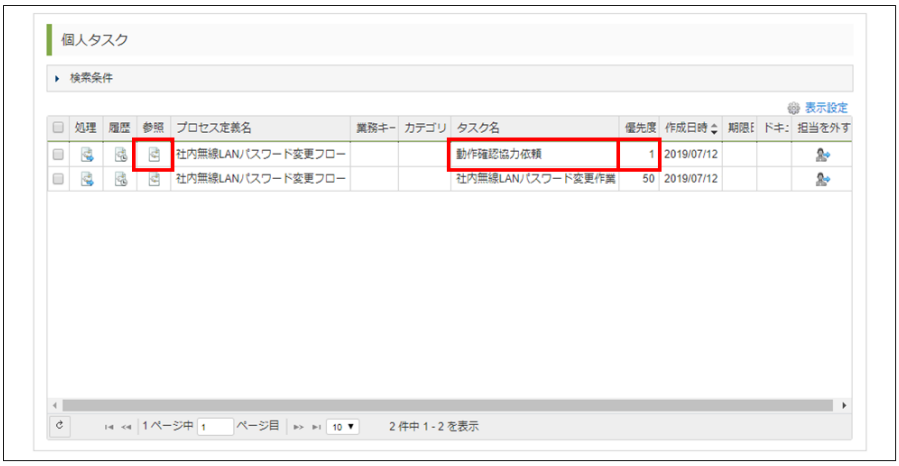
図：「タスク追加」

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」

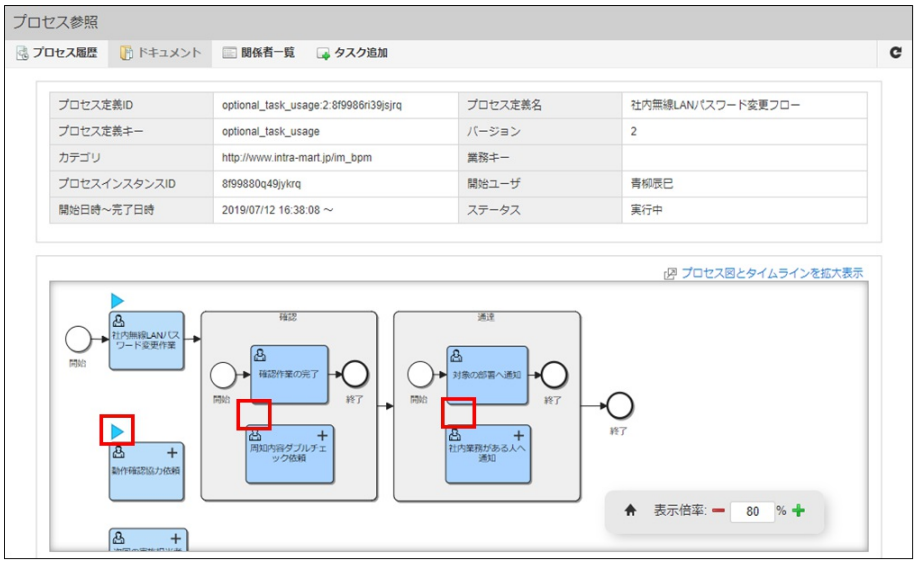
15. 対象のプロセス定義から、「動作確認協力依頼」タスクが優先度「1」で追加されていることを確認し、「 」をクリックします。



図：「タスク一覧」 - 「個人タスク」

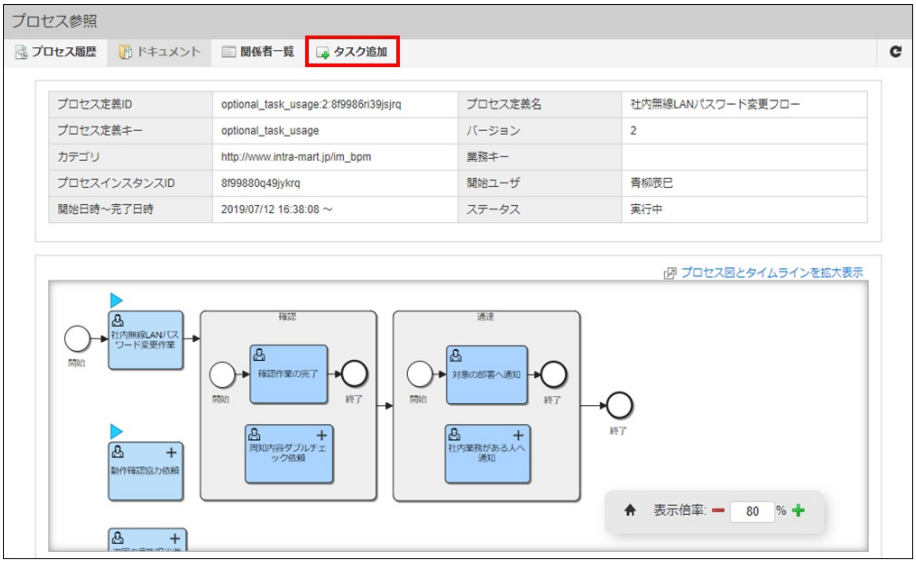
16. 表示される「プロセス参照」画面から、以下のことを確認します。
    - 「タスク追加」画面で追加した、「動作確認協力依頼」タスクに「▶」マークがついていること
    - 「タスク追加」画面で追加した「内容ダブルチェック依頼」タスクはまだ実行範囲ではないため、「▶」マークがついていないこと

      「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ナルタスクは、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、オプショナルタスクを追加します。
プロセスのオプショナルタスク一覧から、「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」

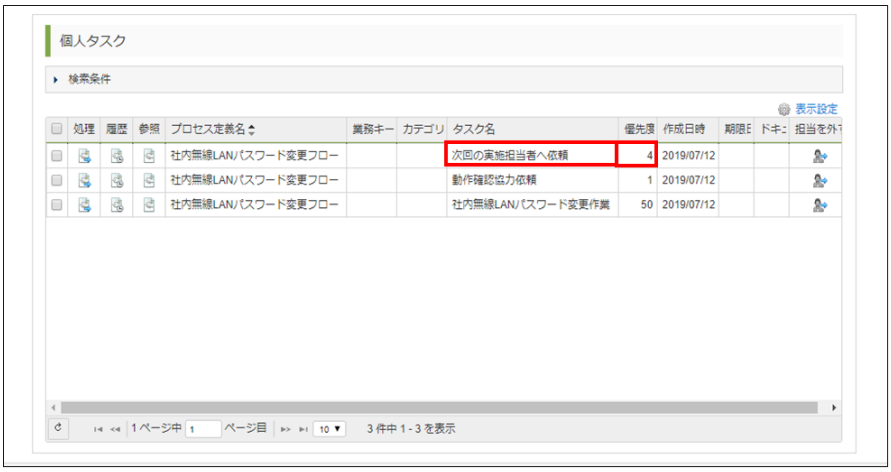
21. 「次回の実施担当者へ依頼」タスクが追加されたことを確認します。

    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、オプショナルタスク一覧にある「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」

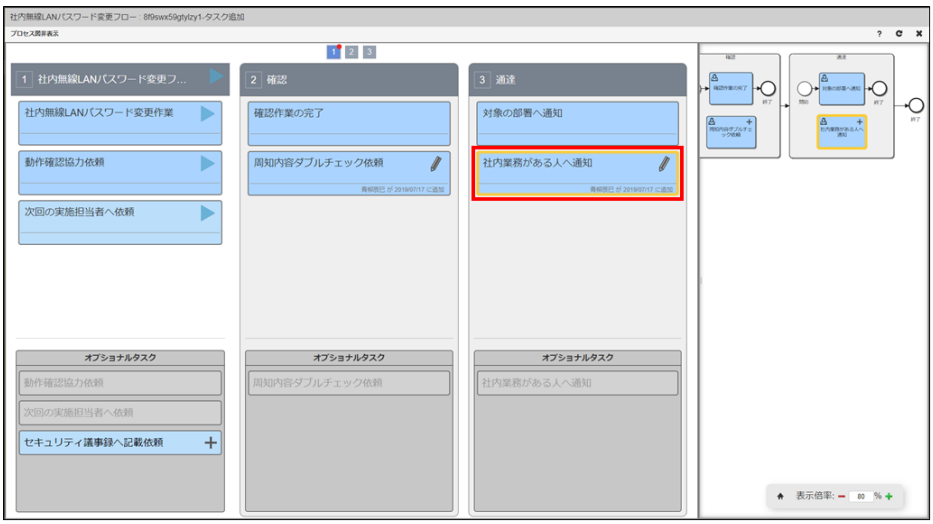
22. 「タスク一覧」画面から、オプショナルタスクが追加されたことを確認します。
    「個人タスク」に「次回の実施担当者へ依頼」タスクが優先度「4」で追加されていることを確認します。



図：「タスク一覧」- 「個人タスク」

23. 追加したオプショナル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。



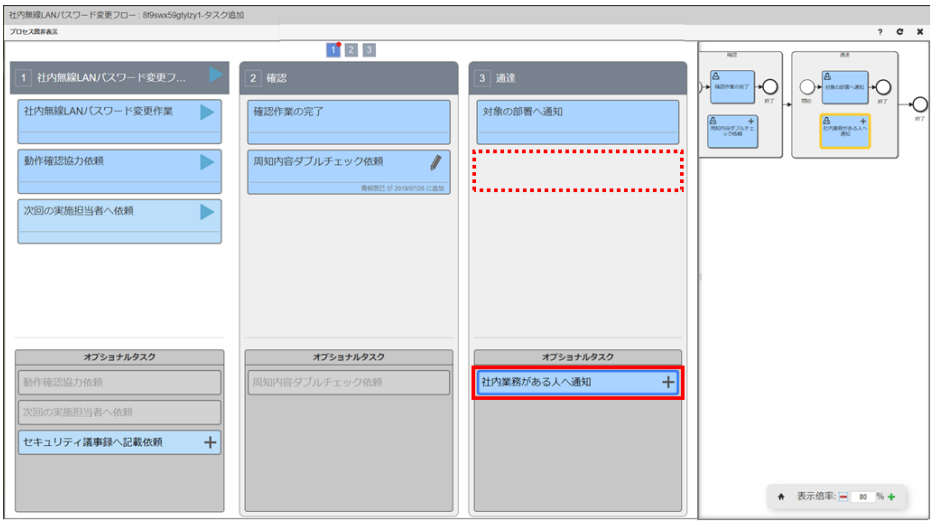
図：「タスク追加」

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**
>
> 実行範囲にいないオプショナルタスクは、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、オプショナルタスク一覧に戻ります。



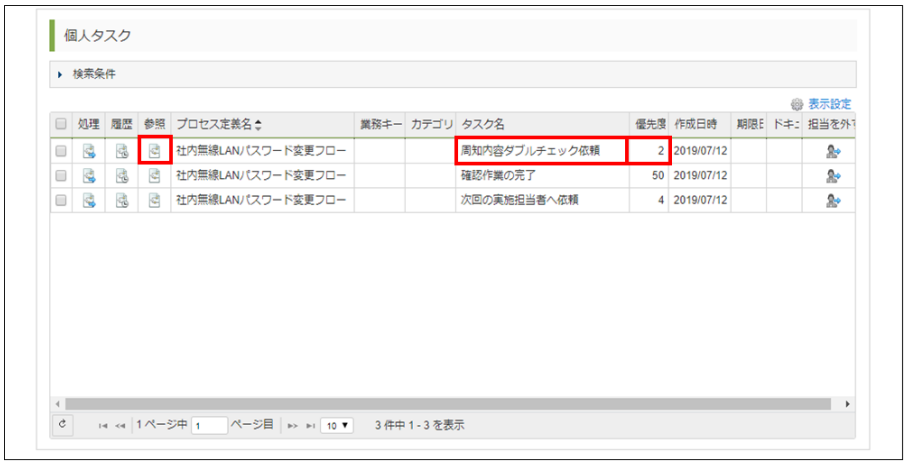
図：「タスク追加」

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。

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。



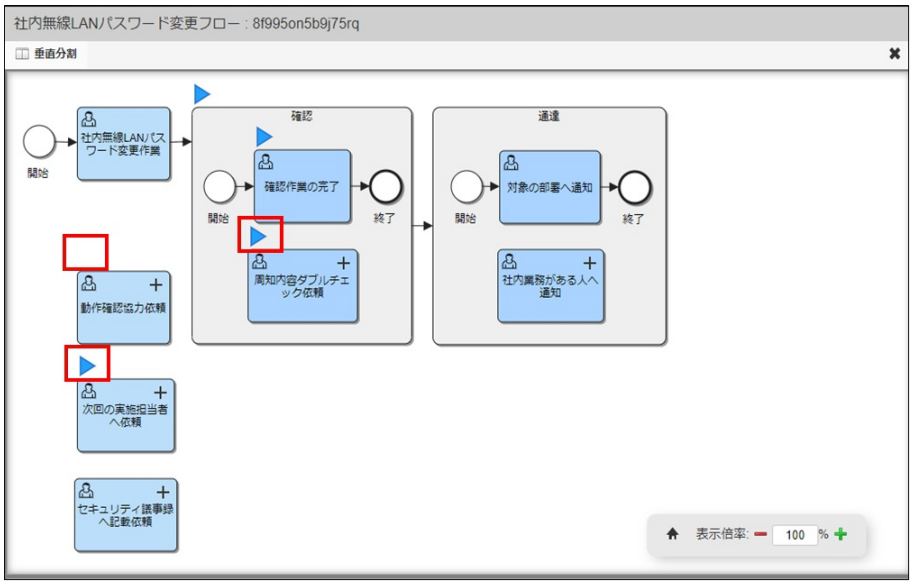
図：「タスク一覧」- 「個人タスク」

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

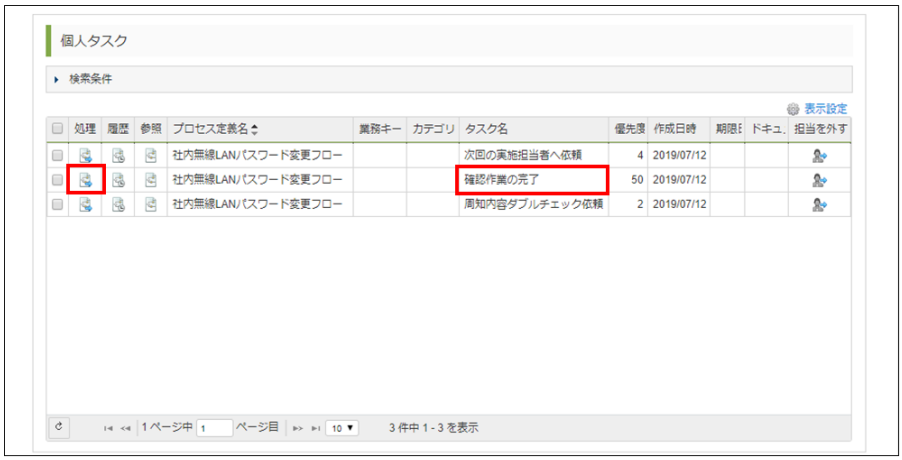
  プロセスの「次回の実施担当者へ依頼」は実行中であり、「▶」マークがついていること

- 「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. 追加したオプショナル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」

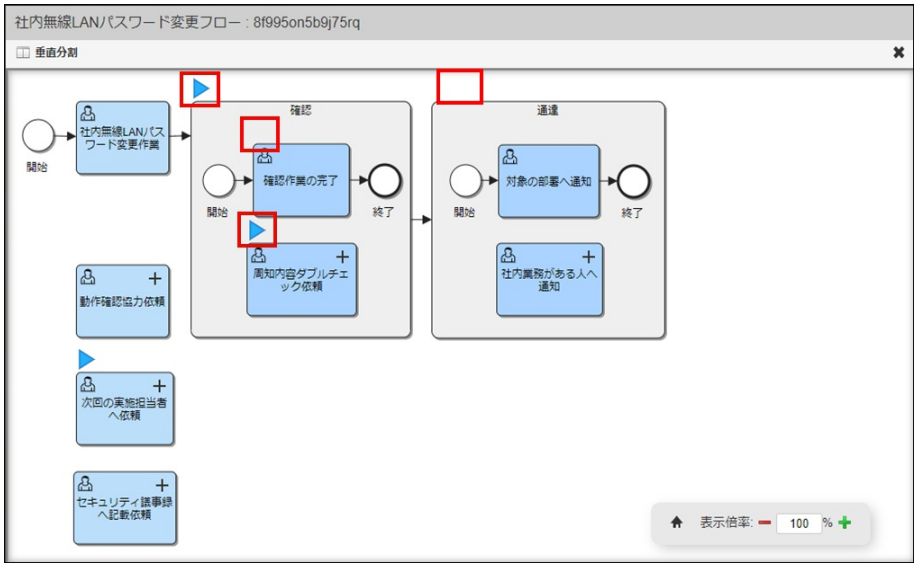
31. 「プロセス詳細」画面、または「プロセス図とタイムラインを拡大表示」から、プロセスの状態を確認します。
以下の事項を確認します。

- 「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

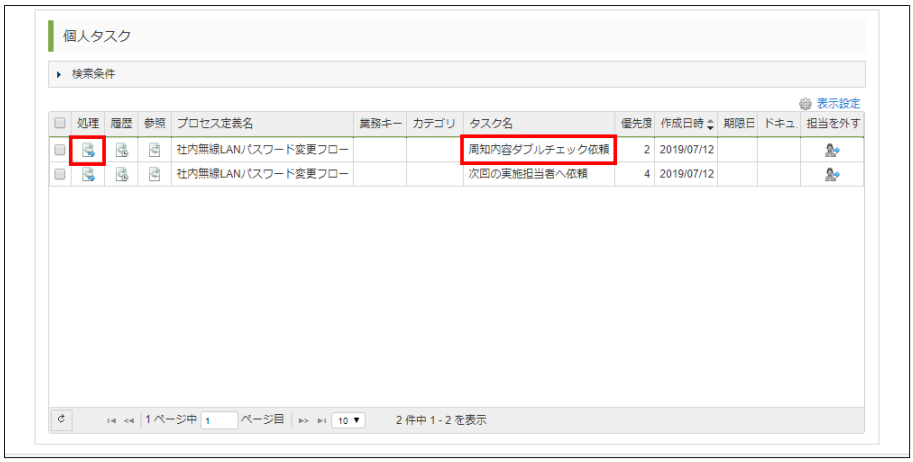
- 「通達」サブプロセスは実行中ではないため、「 」マークがついていないこと



図：「プロセス図とタイムラインを拡大表示」

32. オプショナルタスクを処理し、「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。

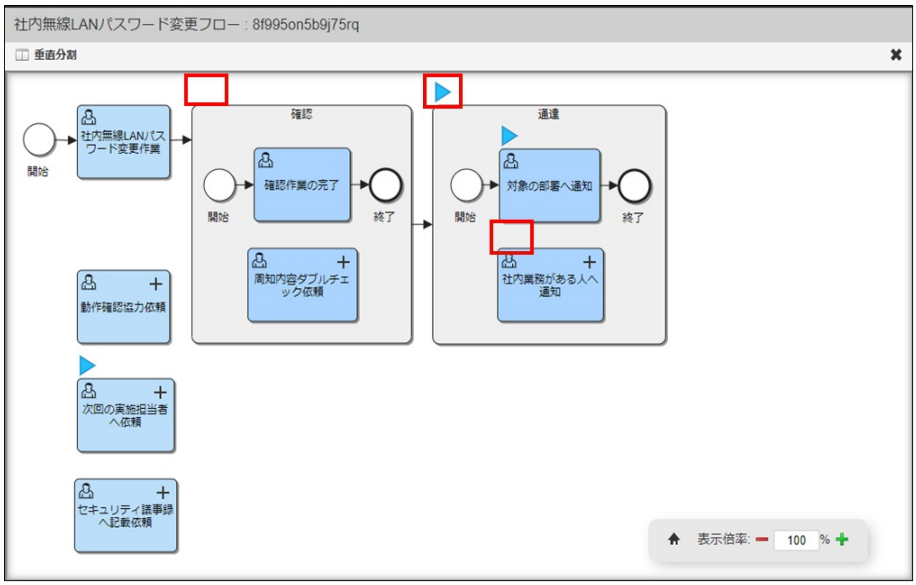
「タスク一覧」画面から、対象のプロセス定義の「」をクリックします。

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

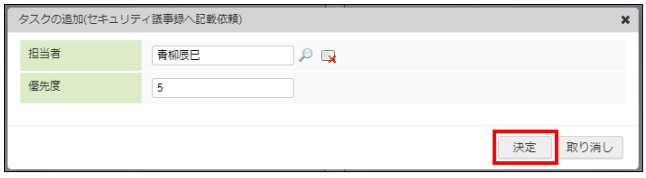


図：「プロセス図とタイムラインを拡大表示」

35. プロセスに対してのオプショナル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ナルタスク一覧にある「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

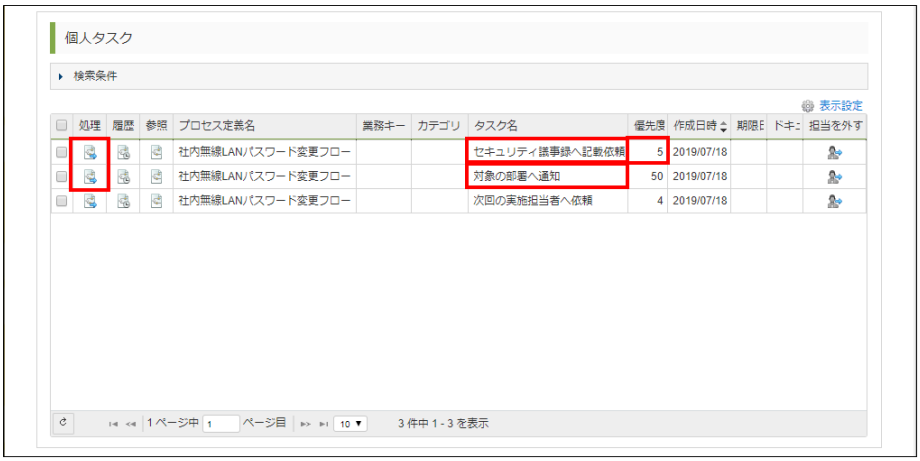


図：「タスクの追加」

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

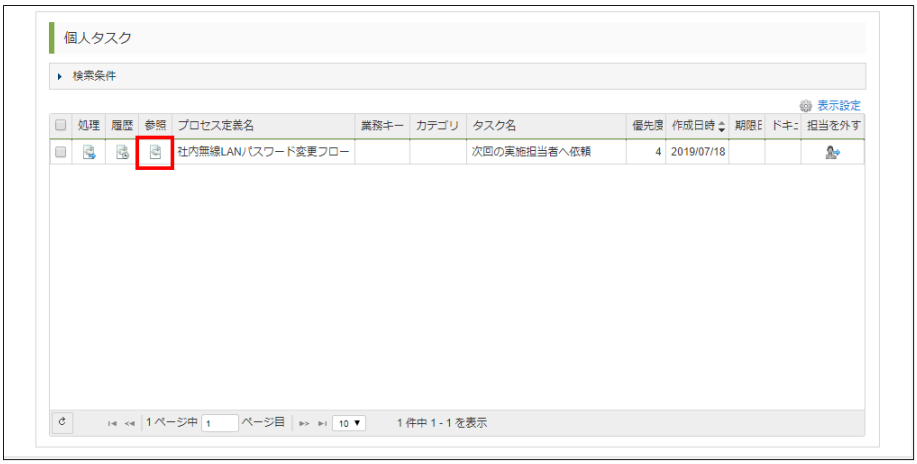
図：「タスク追加」

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。



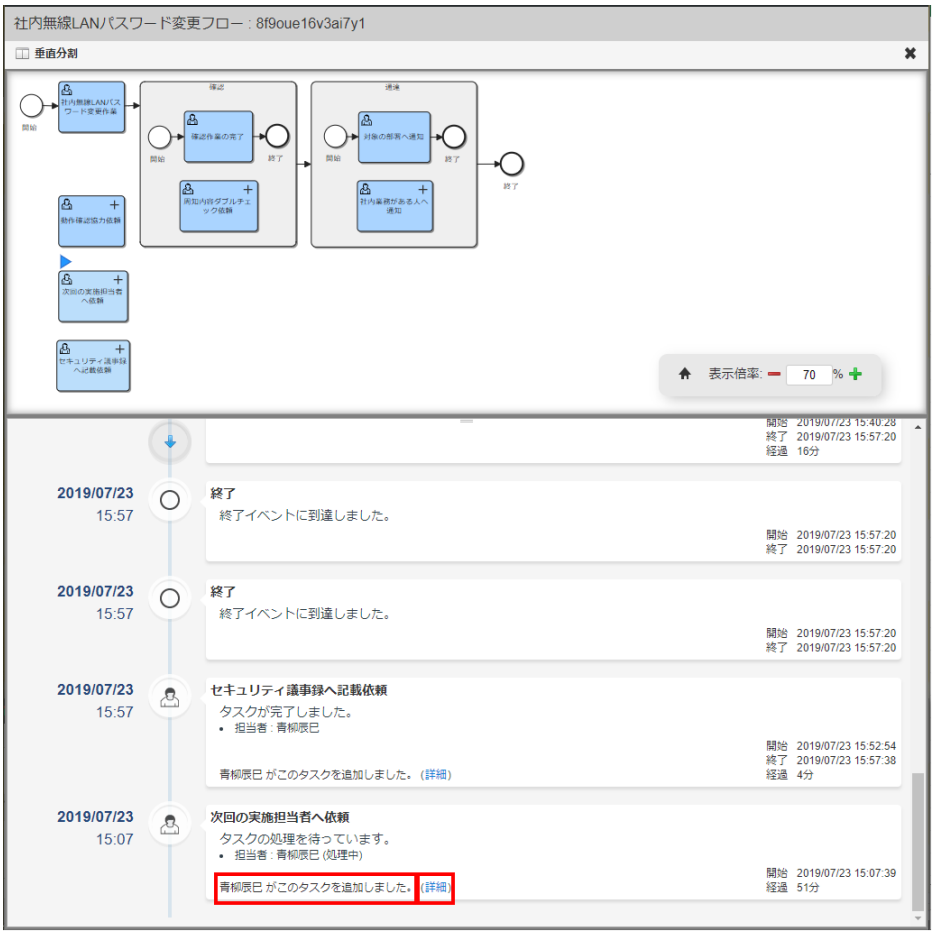
図：「タスク一覧」 - 「個人タスク」

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、オプショナル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、「オプショナルタスク詳細」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。



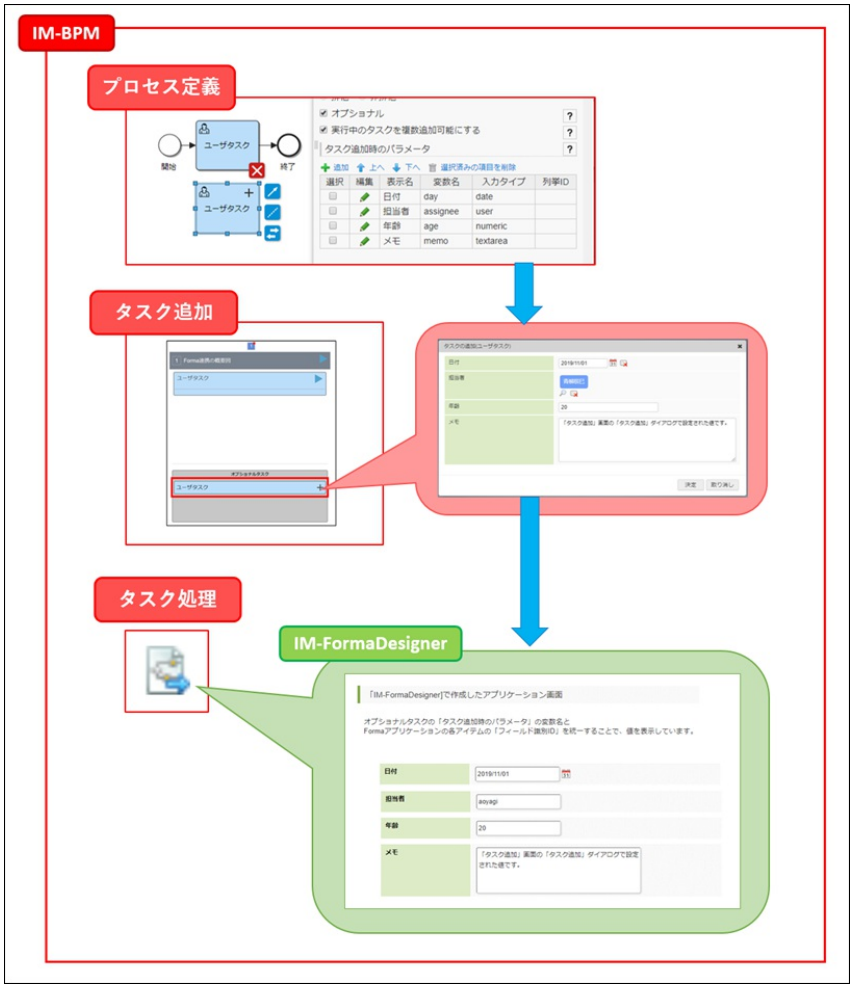
図：「プロセス図とタイムラインを拡大表示」

図：「オプショナルタスク詳細」

オプショナルタスクのパラメータをFormaアプリケーションと連携する

このチュートリアルでは、オプショナルタスクを追加する際に設定できるパラメータを「IM-FormaDesigner」で作成したアプリケーション画面で使用する方法を解説します。

オプショナルタスクの詳細については、「IM-BPM 仕様書」 - 「オプショナルタスク」 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材をインポートしてください。

- ロジックフロー

  im_logicdesigner-optional_task_usage-link_parameters_with_forma.zip

- Formaアプリケーション

  im_forma_designer-optional_task_usage-physical_checkup_questionnaire.zip
  im_forma_designer-optional_task_usage-physical_checkup_result.zip

> **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。
>
> - 「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」
> - ロジックフロー：「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」

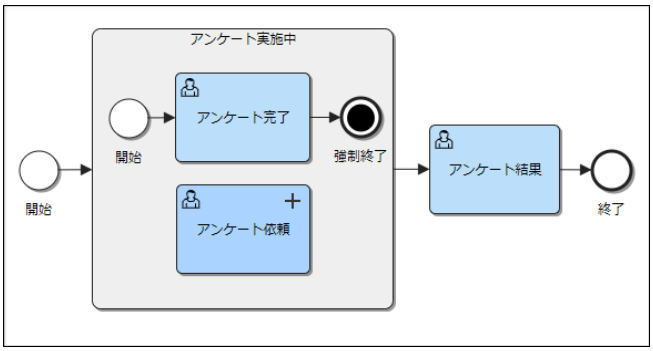
> **コラム**
>
> このチュートリアルで作成するプロセス定義は、以下のリンクからダウンロードできます。
>
> **optional_task_usage-link_parameters_with_forma.bpmn**
>
> プロセス定義のアップロード方法については、以下のリンクを参照してください。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

- プロセス定義を作成する
- 「IM-FormaDesigner」のアプリケーション画面を確認する
- 結果を確認する

プロセス定義を作成する

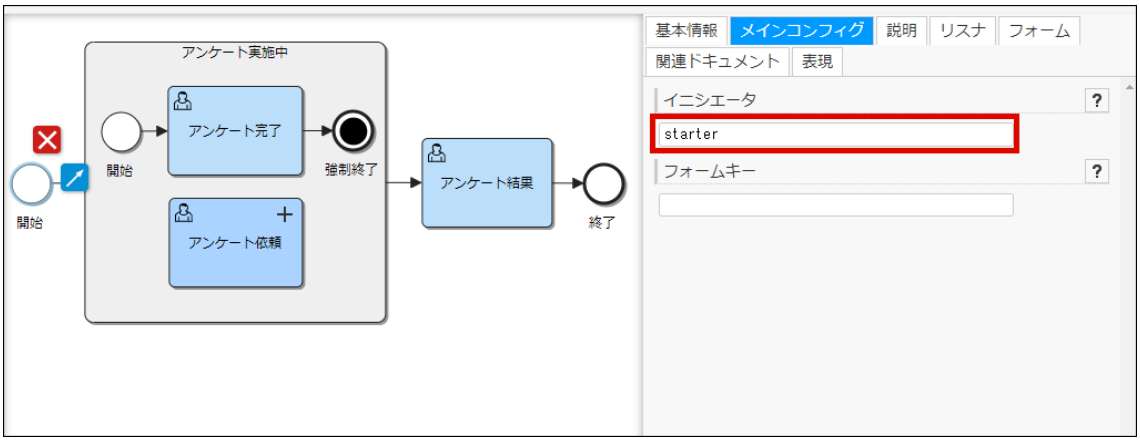
下記のプロセス図は、「健康診断のアンケートフロー」です。
アンケート実施者がプロセスを開始し、アンケートに回答してほしいユーザに対して「アンケート依頼」タスクを実行します。
Formaアプリケーションの「アクション設定」を使用し、「タスク追加時に設定された年齢が40歳以上だった場合、胃がん検診の内容（バリウム・胃カメラ）を選択させる」ようにします。
「アンケート完了」タスクを処理すると未回答のアンケートが強制終了され、「アンケート結果」タスクからアンケートの集計内容が確認できます。

- 「アンケート実施中」サブプロセス
  「健康診断アンケートフロー」を開始することで、最初に実行状態に入るサブプロセスです。
- 「アンケート完了」タスク
  オプショナルタスク「アンケート依頼」を実行している間、フローを開始したユーザに滞在するタスクです。
  このタスクを処理することで「アンケート実施中」サブプロセスが終了し、「アンケート結果」タスクに移行します。
- 「アンケート依頼」タスク
  複数追加可能な設定がされているオプショナルタスクです。
  タスクの担当者やアンケート回答者の年齢などを「タスク追加」画面のパラメータで設定します。
  実行されたタスクは、フォームキーで設定された「健康診断アンケート」画面を表示します。
  アンケートで入力された値は、タスクリスナで設定されている「IM-LogicDesigner」で計算されます。
- 「アンケート結果」タスク
  「アンケート実施中」サブプロセスが完了すると到達するタスクです。
  「IM-LogicDesigner」で集計した値を、フォームキーで設定された「アンケート結果」画面に表示します。



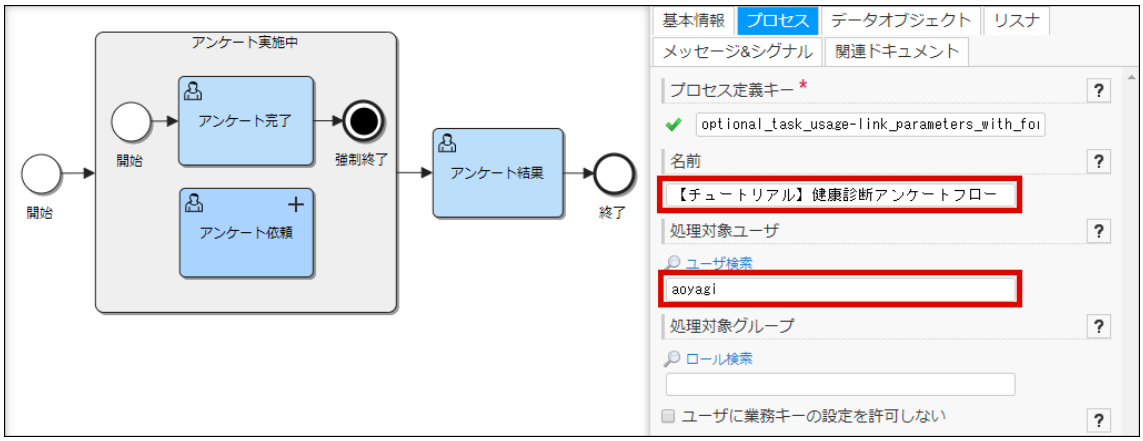
図：プロセス定義図

1. 「開始イベント」を設置します。
2. フローを開始したユーザを各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、「イニシエータ」に starter と設定します。



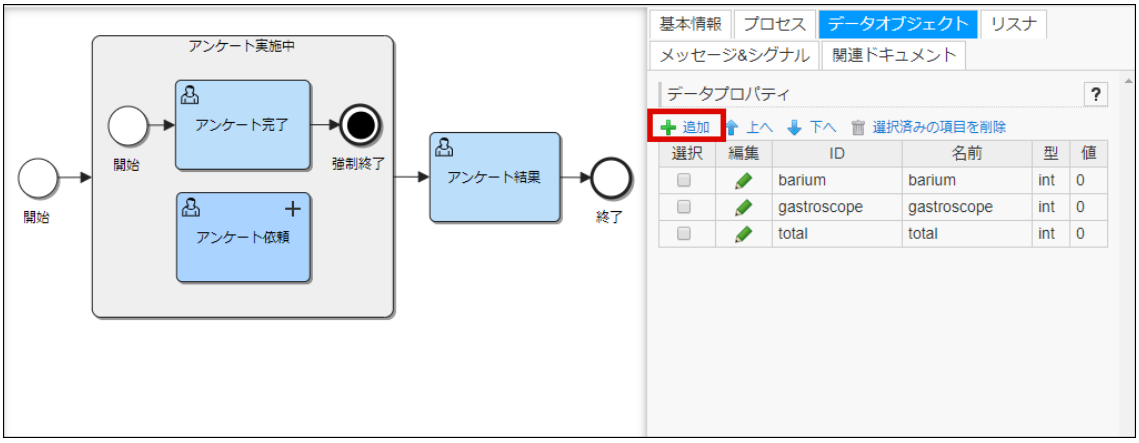
図：プロセス全体 - 「プロパティ」 - 「メインコンフィグ」

3. プロセス全体の設定を行います。
   キャンバ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、以下のように項目を設定します。
   - 名前：健康診断アンケートフロー
   - 処理対象ユーザ：aoyagi



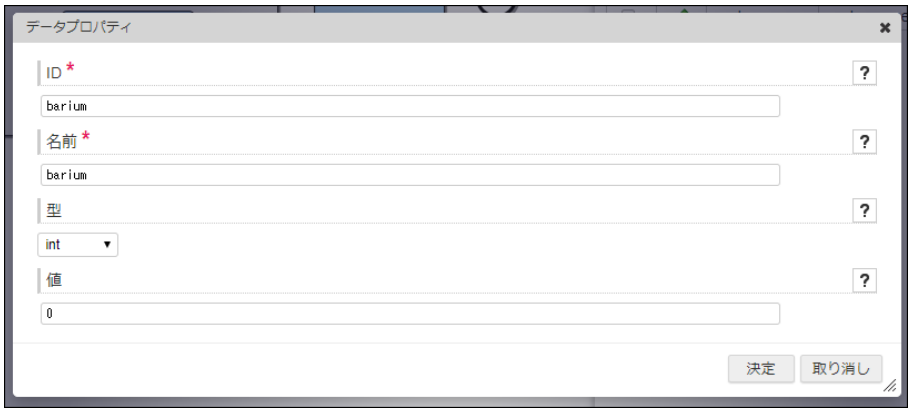
図：プロセス全体-「プロパティ」 -「プロセス」

4. Formaアプリケーションで使用する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」リンクをクリックします。



図：「ユーザタスク」 -「プロパティ」 -「データオブジェクト」

5. 「データプロパティ」ダイアログが表示されます。
   以下のように、3つのパラメータを設定します。
   - バリウム検査希望人数の初期値
     - ID：barium
     - 名前：barium
     - 型：int
     - 値：0
   - 胃カメラ検査希望人数の初期値
     - ID：gastroscope
     - 名前 :gastroscope
     - 型：int
     - 値：0
   - 回答合計人数の初期値
     - ID：total
     - 名前：total
     - 型：int
     - 値：0



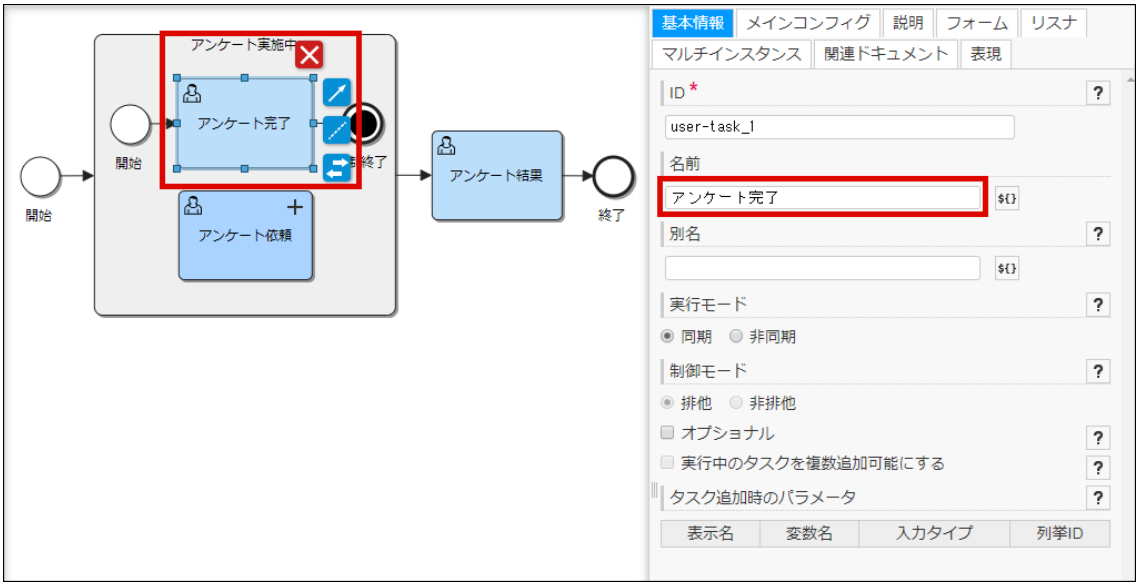
図：「データプロパティ」

6. アンケートの実施に関係するタスクを内包するサブプロセスを設置します。

7. サブプロセス内に「開始イベント」を設置します。

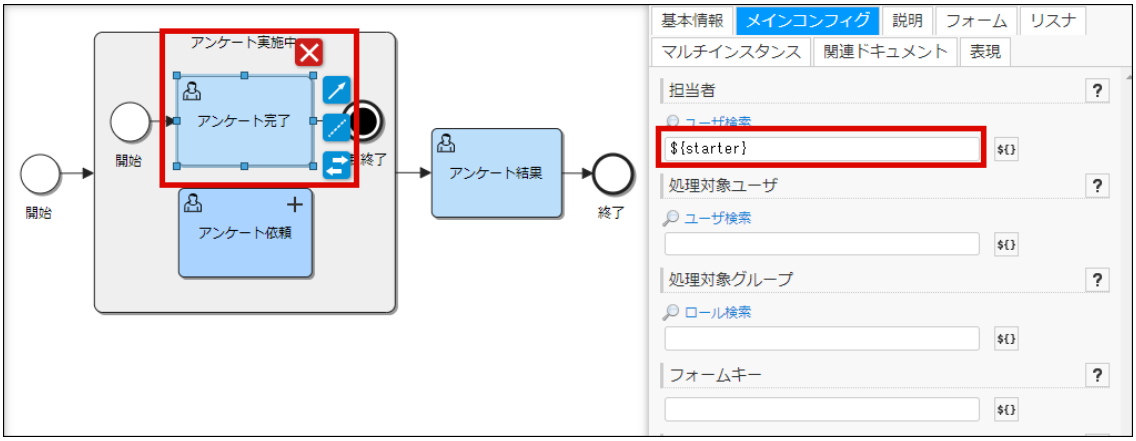
8. アンケートの実施を終了する際に処理する「アンケート完了」タスクを設置します。
   サブプロセス内に「ユーザタスク」を設置します。

9. 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」を選択した状態で、「基本情報」タブから「名前」に `アンケート完了` と設定します。



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

10. このプロセス定義を開始したユーザに対して「アンケート完了」タスクが実行されるように、EL式を設定します。
    「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 `${starter}` と設定します。



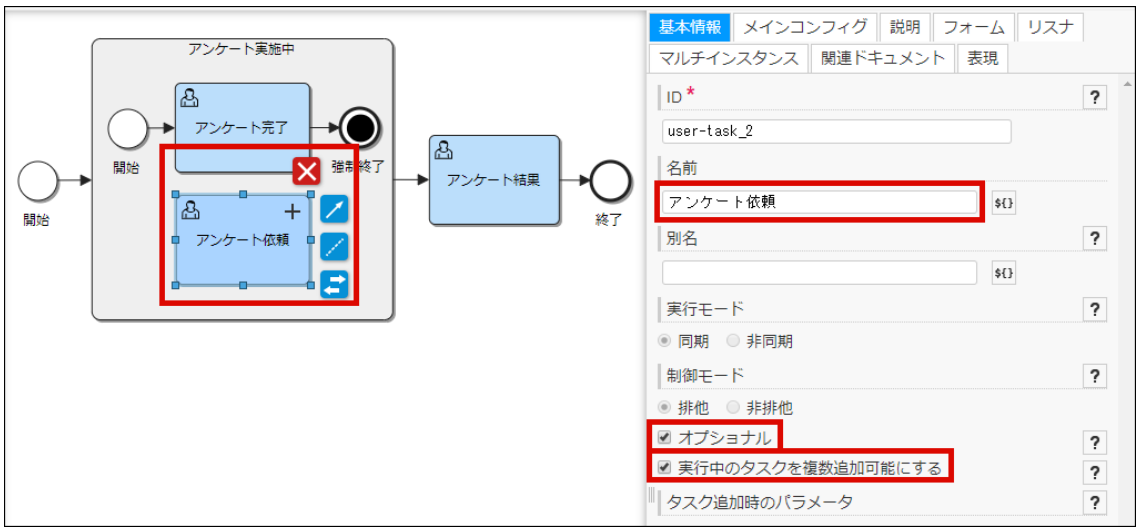
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

11. 「アンケート完了」タスクを処理することで「アンケート実施中」サブプロセスが終了するようにします。
    「強制終了イベント」を設置します。

12. アンケートを依頼したいユーザに対して作成する「アンケート依頼」タスクを設置します。

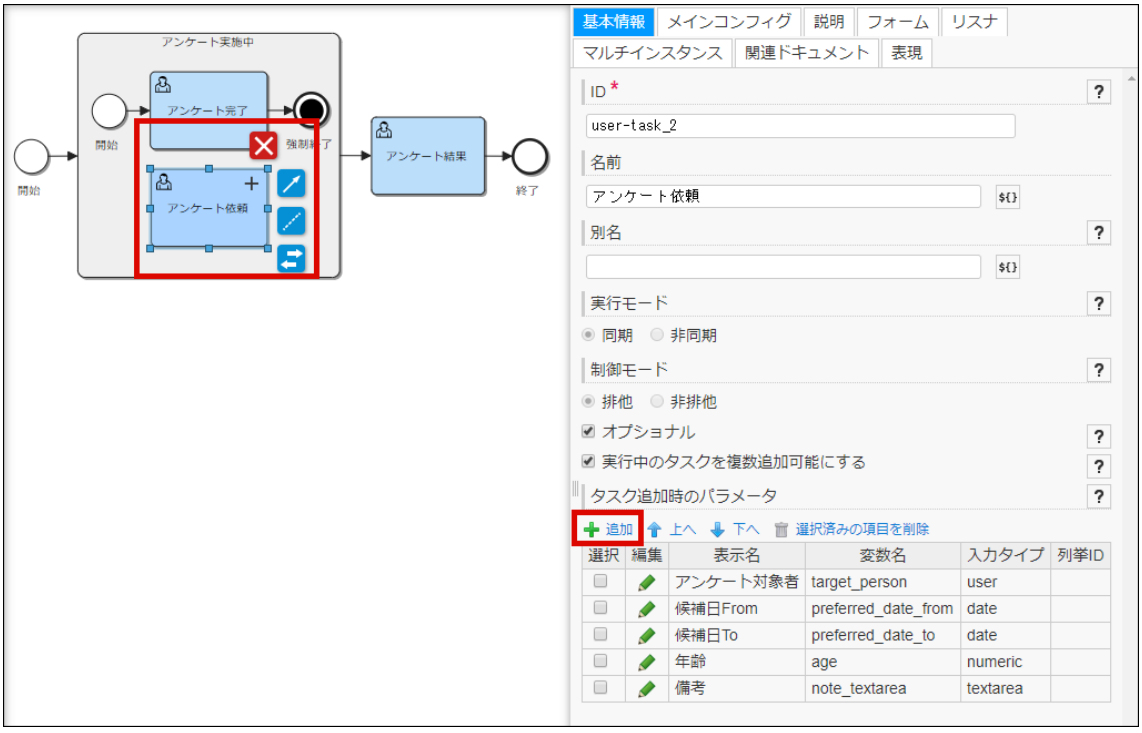
「ユーザタスク」を設置します。

13. 設置したユーザタスクを「オプショナルタスク」にします。
「ユーザタスク」を選択した状態で、「基本情報」タブから以下のように項目を設定します。
    - 名前：アンケート依頼
    - オプショナル：有効
    - 実行中のタスクを複数追加可能にする：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

14. 「オプショナルタスク」にパラメータを設定します。
ユーザタスクを選択した状態で、「基本情報」タブから「タスク追加時のパラメータ」にある「追加」リンクをクリックします。

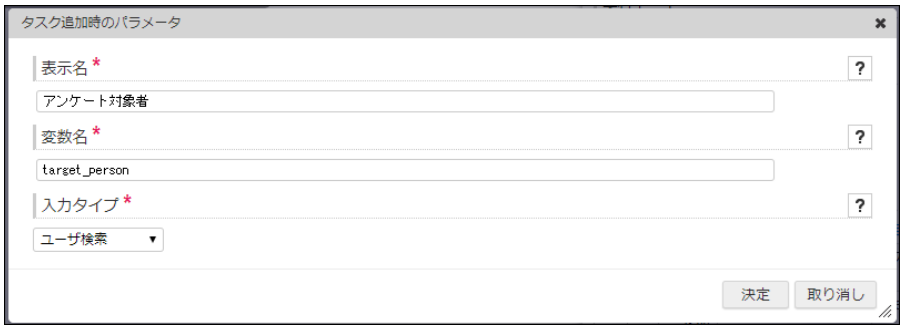


図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

15. 「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、5つのパラメータを設定します。
    - アンケート対象者
      表示名：アンケート対象者
      変数名：target_person
      入力タイプ：ユーザ検索
    - 候補日From
      表示名：候補日From
      変数名：preferred_date_from
      入力タイプ：日付
    - 候補日To
      表示名：候補日To

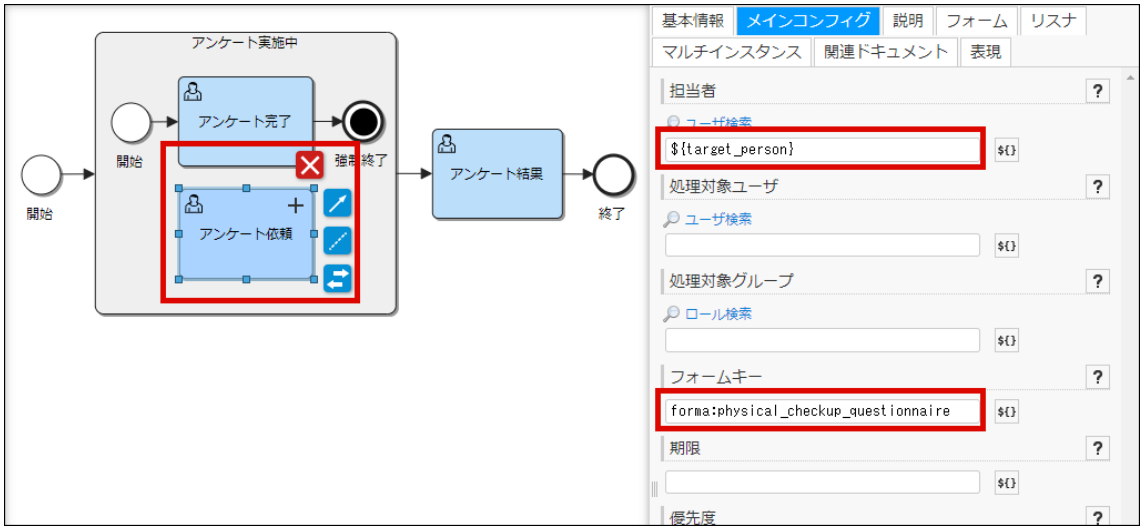
変数名：preferred_date_to
入力タイプ：日付

- 年齢
  表示名：年齢
  変数名：age
  入力タイプ：数字
- 備考
  表示名：備考
  変数名：note_textarea
  入力タイプ：複数行の文字列



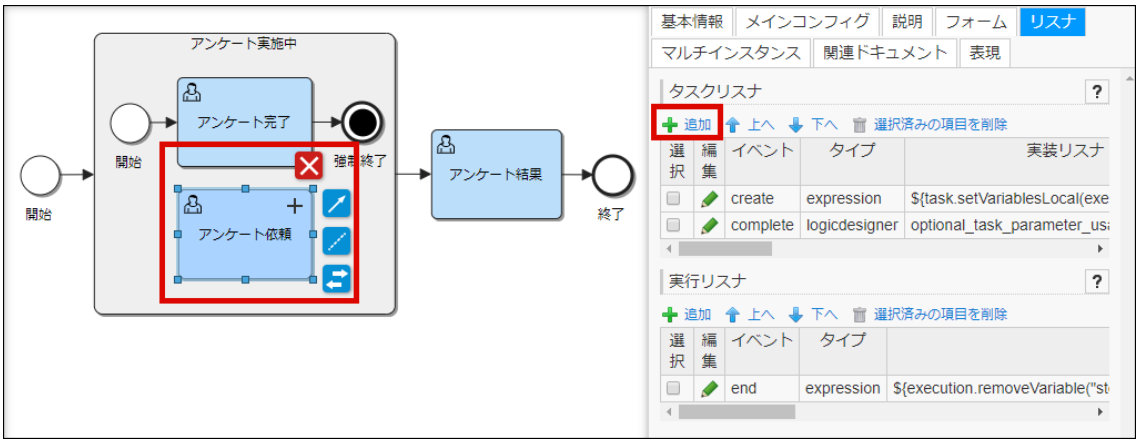
図：「タスク追加時のパラメータ」

16. 「タスク追加」画面で指定されたユーザを担当者に設定し、フォーム画面を紐づけるためEL式を設定します。
「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：${target_person}
    - フォームキー：forma:physical_checkup_questionnaire



図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

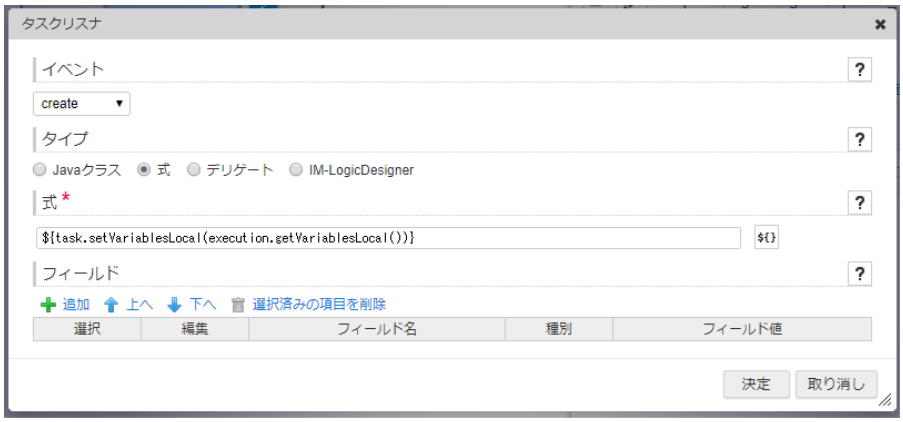
17. 「タスク追加」画面で入力される値を、ユーザタスクのローカル変数に詰め替える設定をします。
「ユーザタスク」を選択した状態で、「リスナ」タブから「タスクリスナ」にある「追加」リンクをクリックします。



図：「ユーザタスク」 - 「プロパティ」 - 「リスナ」

18. 「タスクリスナ」ダイアログから、下記のように項目を設定します。

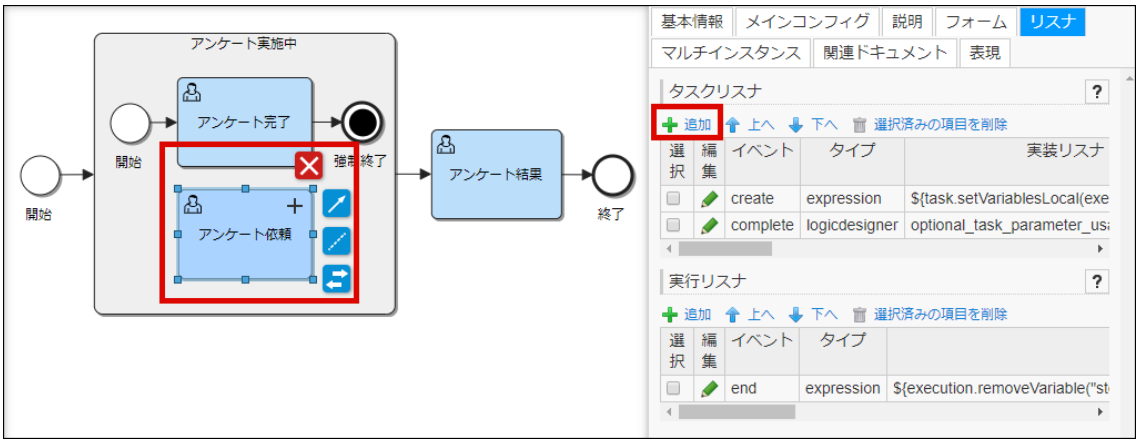
- イベント：create
- タイプ：式
- 式：${task.setVariablesLocal(execution.getVariablesLocal())}



図：「タスクリスナ」

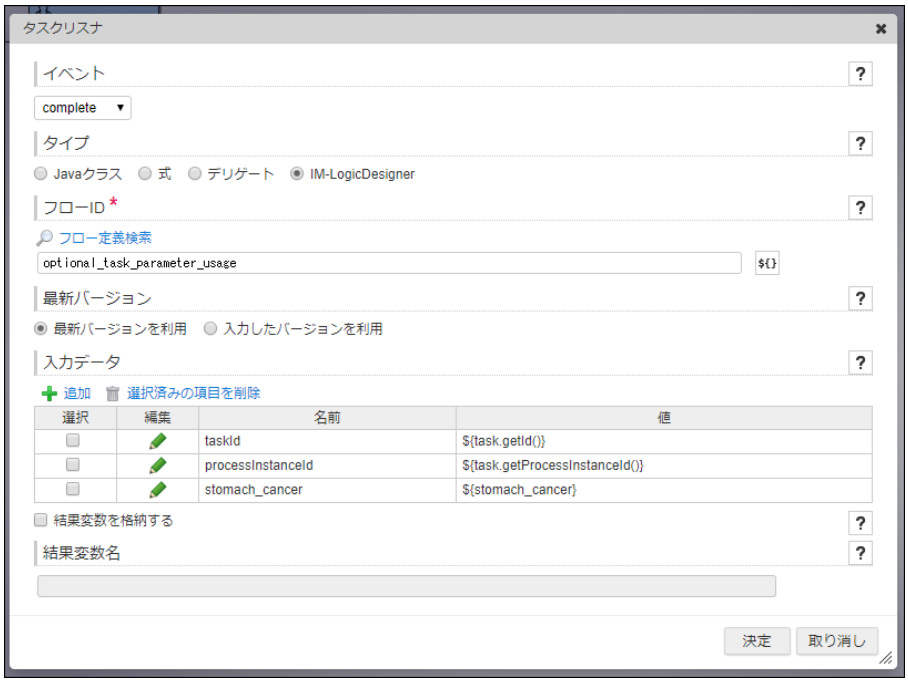
> **コラム**
>
> スコープとは、変数の参照可能範囲を指します。
> 変数のスコープは、それぞれ以下のような参照範囲を表します。
>
> - Process：当該のプロセス内のどこからでも参照できます。
> - Execution：当該のアクティビティが実行されているエグゼキューション内からのみ参照できます。
> - Task：当該のユーザタスクからのみ参照できます。
>
> Formaアプリケーションとの連携は、ProcessおよびTaskのスコープの変数のみ連携されるため、オプショナルタスクで追加した変数情報をTaskのスコープに詰め替えをしています。

19. アンケートの回答内容を「IM-LogicDesigner」で計算します。
「ユーザタスク」を選択した状態で、「リスナ」タブから「タスクリスナ」にある「追加」リンクをクリックします。



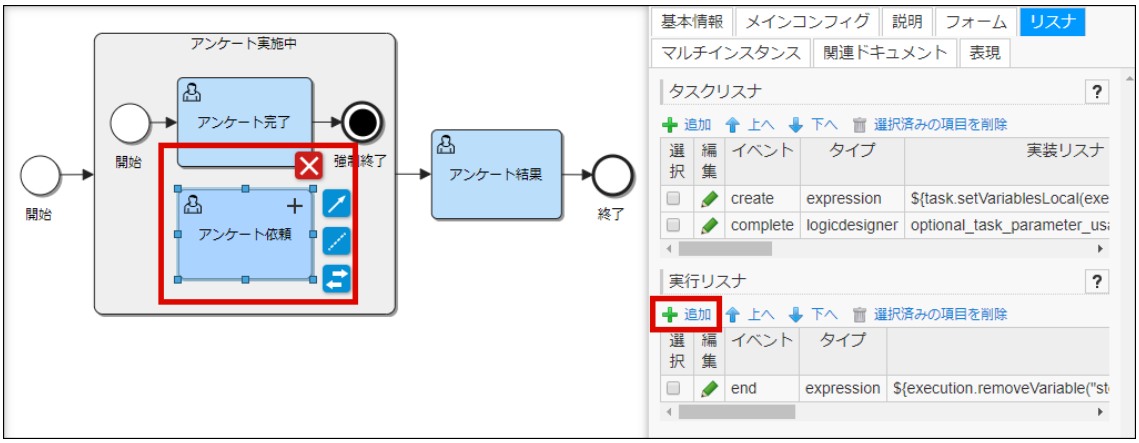
図：「ユーザタスク」 - 「プロパティ」 - 「リスナ」

20. 「タスクリスナ」ダイアログから、下記のように項目を設定します。
    - イベント：complete
    - タイプ：IM-LogicDesigner
    - フローID：optional_task_parameter_usage
    - 最新バージョン：最新バージョンを使用
    - 入力データ
        - taskId：${task.getId()}
        - processInstanceId：${task.getProcessInstanceId()}
        - stomach_cancer：${stomach_cancer}



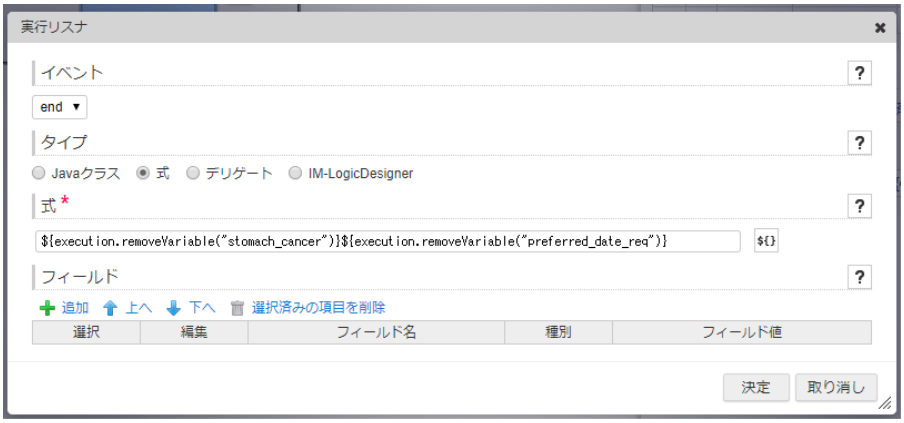
図：「タスクリスナ」

21. 複数追加した「アンケート依頼」タスクは同じForma画面を使用するため、各タスクの情報を混同しないようにリセットするEL式を設定します。
「ユーザタスク」を選択した状態で、「リスナ」タブから「実行リスナ」にある「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」

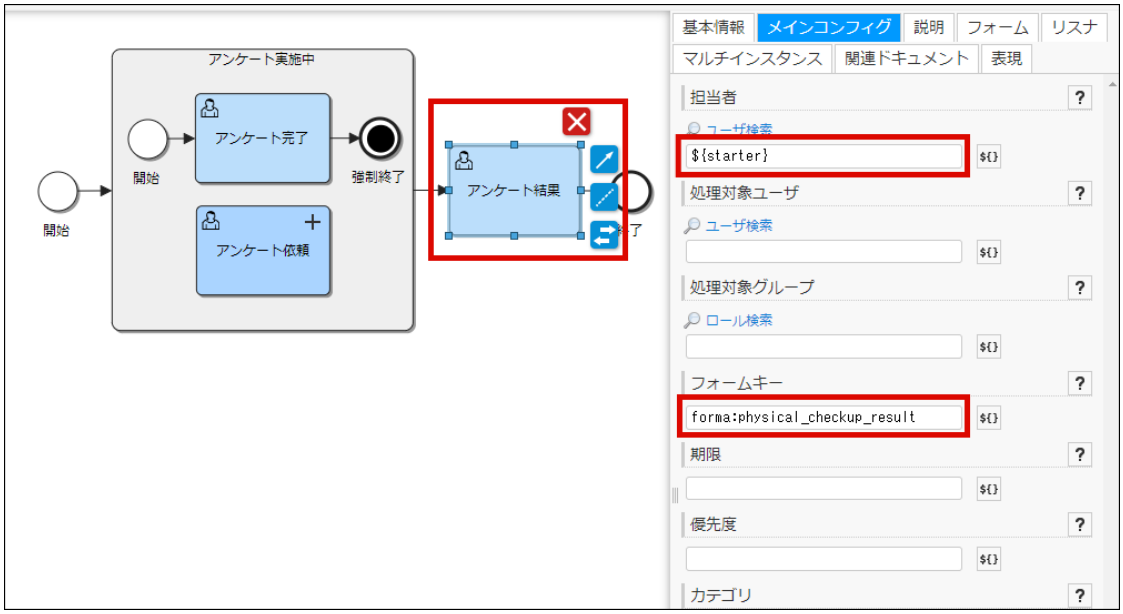
22. 「実行リスナ」ダイアログから、下記のように項目を設定します。
    - イベント：end
    - タイプ：式
    - 式：${execution.removeVariable("stomach_cancer")}${execution.removeVariable("preferred_date_req")}



図：「実行リスナ」

23. アンケートの集計結果を表示するForma画面を開く「アンケート結果」タスクを設置します。
プロセスを開始したユーザを担当者に設定し、フォーム画面を紐づけます。
「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：${starter}
- フォームキー：forma:physical_checkup_result



図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

24. 「終了イベント」を設置します。

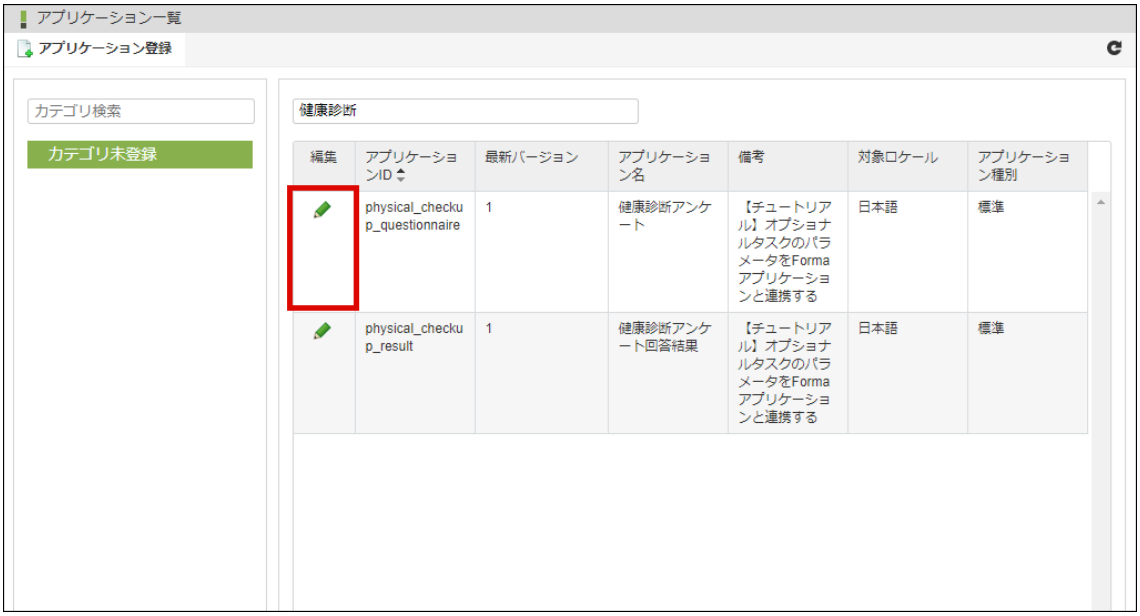
「IM-FormaDesigner」のアプリケーション画面を確認する

このチュートリアルでは、オプショナルタスクを追加時に設定するパラメータが「IM-FormaDesigner」と紐づいている部分の解説のみ行います。
「アンケート結果」タスクに設定したForma画面「physical_checkup_result」は、プロセスインスタンス変数を表示しているのみなので解説しません。
以降の説明は、ユーザタスク「アンケート依頼」に設定されているForma画面「physical_checkup_questionnaire」に関するものです。

「IM-FormaDesigner」の使用方法については「IM-FormaDesigner 作成者操作ガイド」を参照してください。

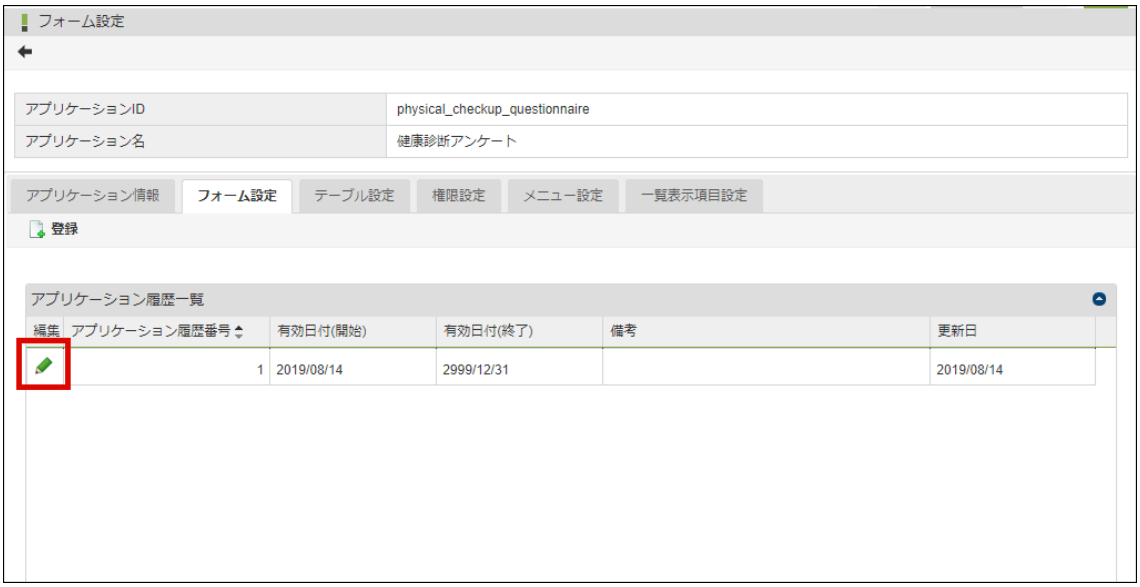
1. 「サイトマップ」→「Forma管理画面」→「Formaアプリ作成」→「アプリ一覧」をクリックして、「アプリケーション一覧」画面を表示します。

2. チュートリアルを開始する前にインポートした「健康診断アンケート」の「」をクリックし、「フォーム設定」画面を開きます。



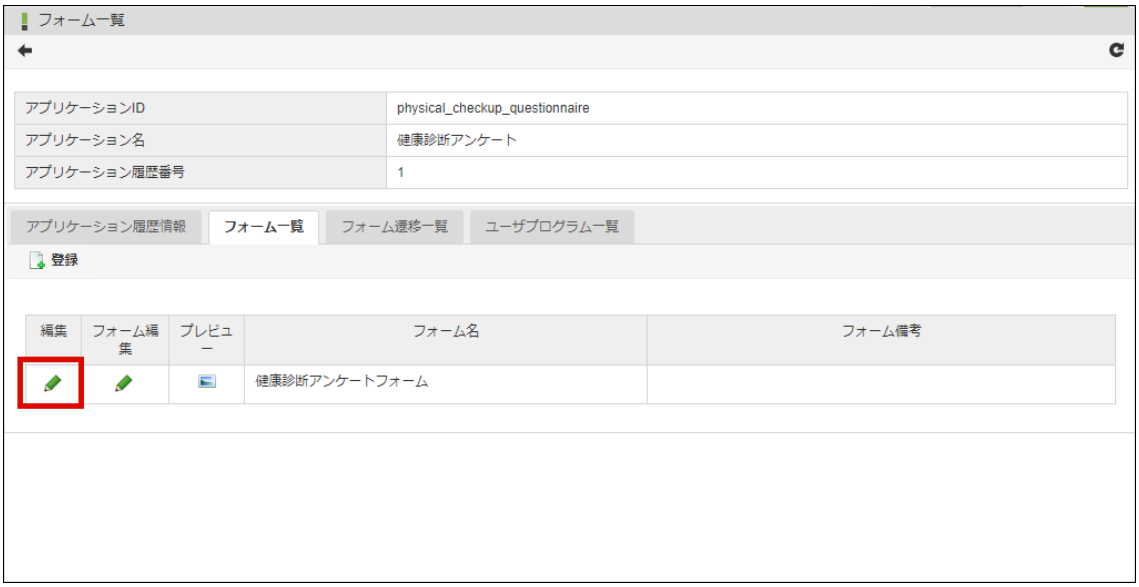
図：「アプリケーション一覧」

3. 「フォーム設定」画面の「フォーム設定」タブから、「アプリケーション履歴一覧」にある「」をクリックし、「フォーム一覧」画面を表示します。



図：「フォーム設定」

4. 「フォーム一覧」画面の「フォーム一覧」タブから、フォーム名「健康診断アンケートフォーム」の「」をクリックして「フォーム編集」画面を表示します。



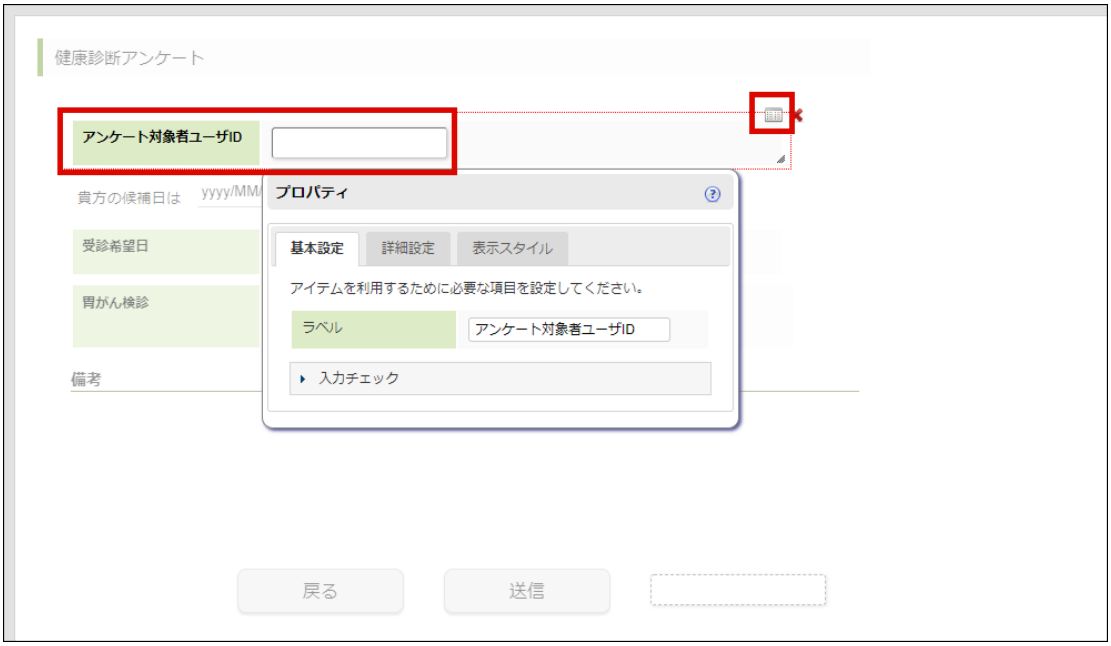
図：「フォーム一覧」

5. 「フォーム編集」画面が表示されます。

図：「フォーム編集」

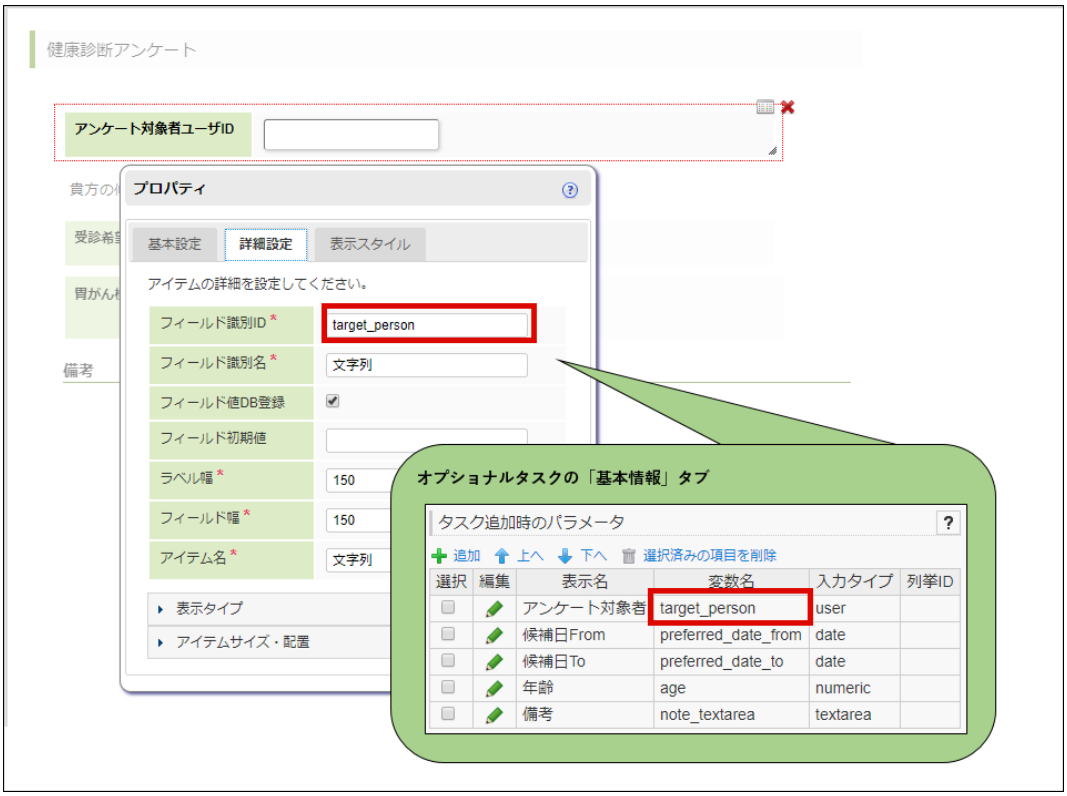
6. オプショナルタスクのパラメータを紐づけるには、「変数名」と入力アイテムの「フィールド識別ID」を統一します。
   アンケート対象者のユーザIDを表示する入力アイテム「文字列」のプロパティを確認します。

   入力アイテム「アンケート対象ユーザID」をダブルクリック、または、アイテムを選択した状態で表示される「 」をクリックします。



図：「フォーム編集」

7. 「プロパティ」ダイアログの「詳細設定」タブから、「フィールド識別ID」を確認します。
   オプショナルタスクの「タスク追加時のパラメータ」に設定した「変数名」と、入力アイテムの「フィールド識別ID」を統一しています。

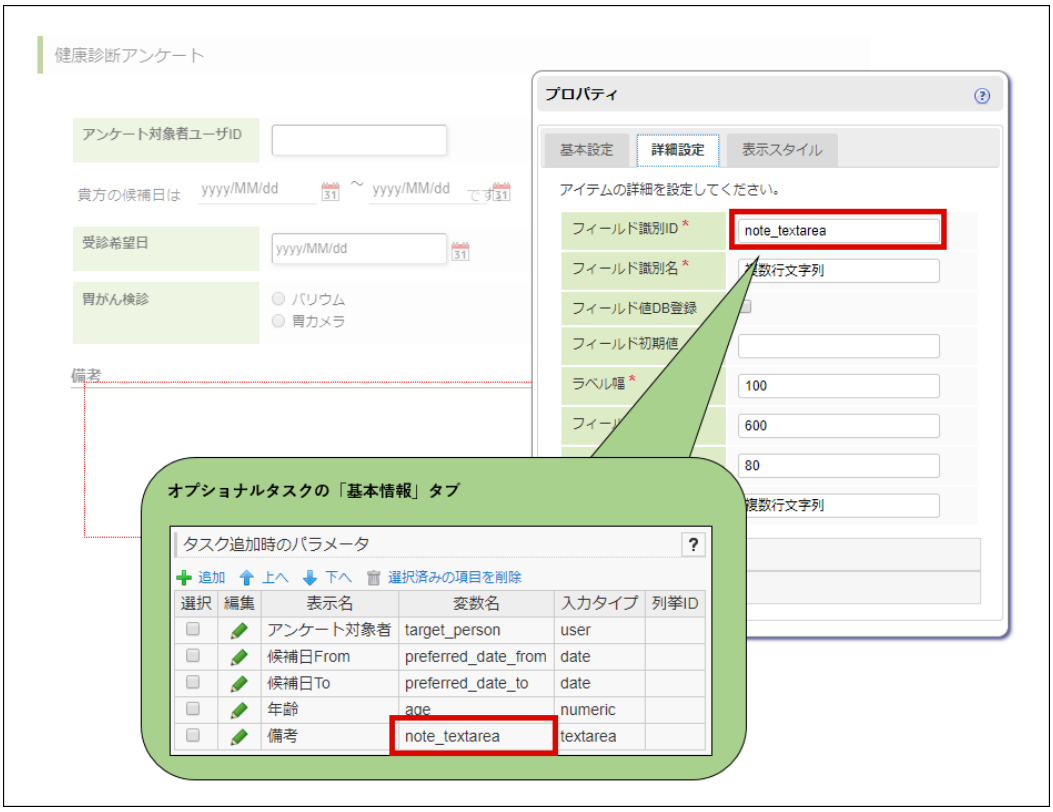


図：「フォーム編集」

8. 入力アイテム「期間」のプロパティを確認します。
オプショナルタスクの「タスク追加時のパラメータ」に設定した「変数名」と、入力アイテムの「フィールド識別ID」を統一しています。

図：「フォーム編集」

9. 表示アイテム「ラベル」のプロパティを確認します。
オプショナルタスクの「タスク追加時のパラメータ」に設定した「変数名」と、入力アイテムの「フィールド識別ID」を統一しています。



図：「フォーム編集」

10. 汎用アイテム「隠しパラメータ」のプロパティを確認します。
オプショナルタスクの「タスク追加時のパラメータ」に設定した「変数名」と、入力アイテムの「フィールド識別ID」を統一しています。

図：「フォーム編集」

11. 隠しパラメータ「age」を使用している箇所を確認します。
「フォーム編集」画面のツールバーにある「アクション設定」ボタンをクリックし、「イベント設定」ダイアログを表示します。

図：「フォーム編集」

12. 「胃がん検診を非表示にする」の設定を確認します。
「初期表示イベント」タブから、「胃がん検診を非表示にする」の「」ボタンをクリックします。

図：「イベント設定」

13. 「表示モード変換設定」ダイアログが表示されます。
アイテム「ラジオボタン」の表示モードが「非表示」にされていることを確認します。
条件がTrueだった場合、「ラジオボタン」が非表示にされるよう設定されていることがわかります。

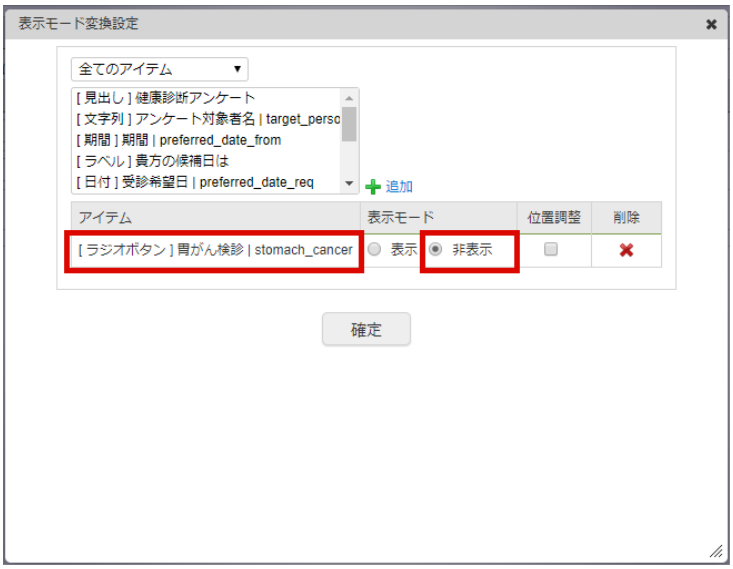


図: 「表示モード変換設定」

14. 「胃がん検診を非表示する」の条件を確認します。

    初期表示イベントタブから、説明が「胃がん検診を非表示する」の「 」ボタンをクリックします。

図：「イベント設定」

15. 「条件設定」ダイアログが表示されます。
    入力欄に「age < 40」と設定してあるのを確認します。
    隠しパラメータ「age」が40歳未満だった場合、Trueの判定が出るように設定されていることがわかります。

図: 「条件設定」

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. 「健康診断アンケートフロー」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、「健康診断アンケートフロー 」の「 」アイコンンをクリックします。

図：「プロセス開始一覧」

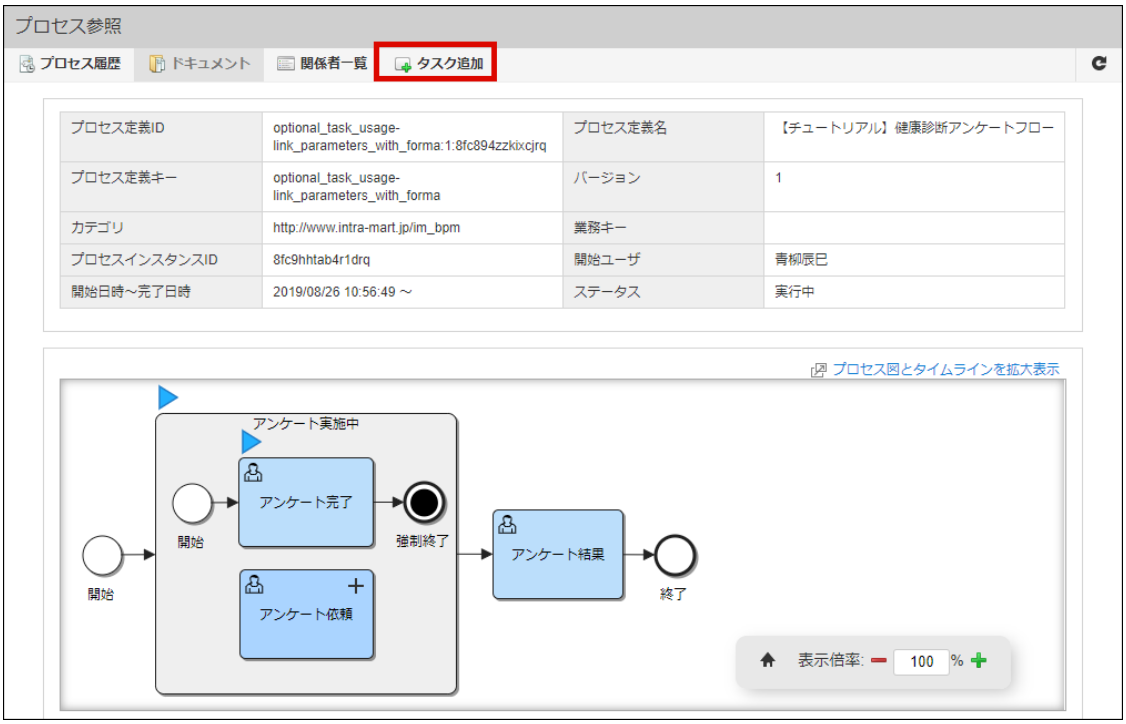
2. 開始されたプロセスを確認します。
「プロセス開始一覧」画面のツールバーにある「タスク一覧」をクリックし、「タスク一覧」画面を表示します。

図：「プロセス開始一覧」

3. 開始したプロセスに対して「オプショナルタスク」を追加します。
「タスク一覧」画面の「個人タスク」にある「健康診断アンケートフロー」の「 」アイコンをクリックし、「プロセス参照」画面を表示します。

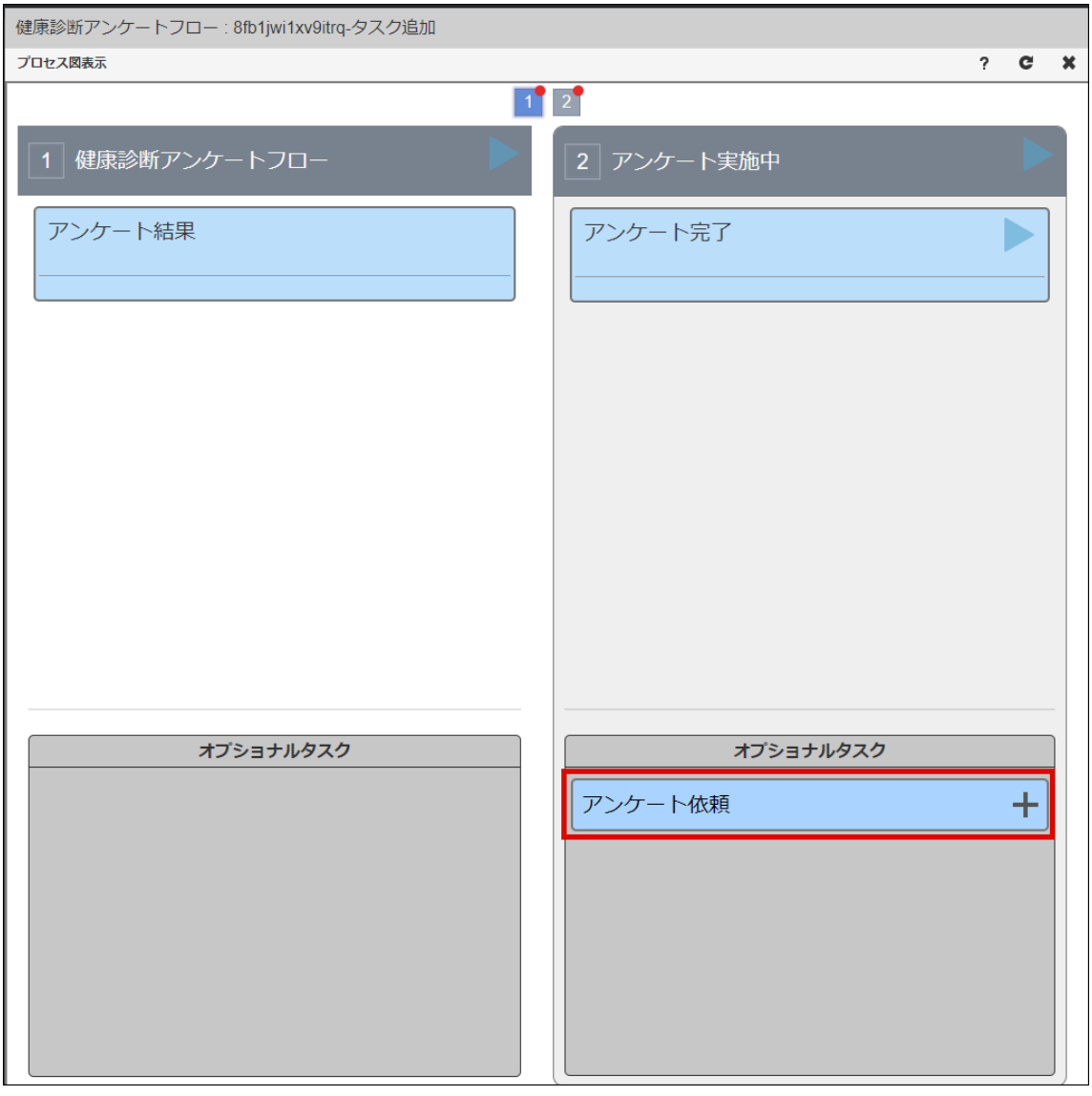
図：「タスク一覧」 - 「個人タスク」

4. 「プロセス参照」画面が表示されます。
ツールバーにある「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」

5. 「アンケート依頼」タスクを作成します。
「アンケート実施中」サブプロセスの中にあるオプショナルタスク「アンケート依頼」をクリックします。



図：「タスク追加」

6. 「タスクの追加」ダイアログが表示されます。

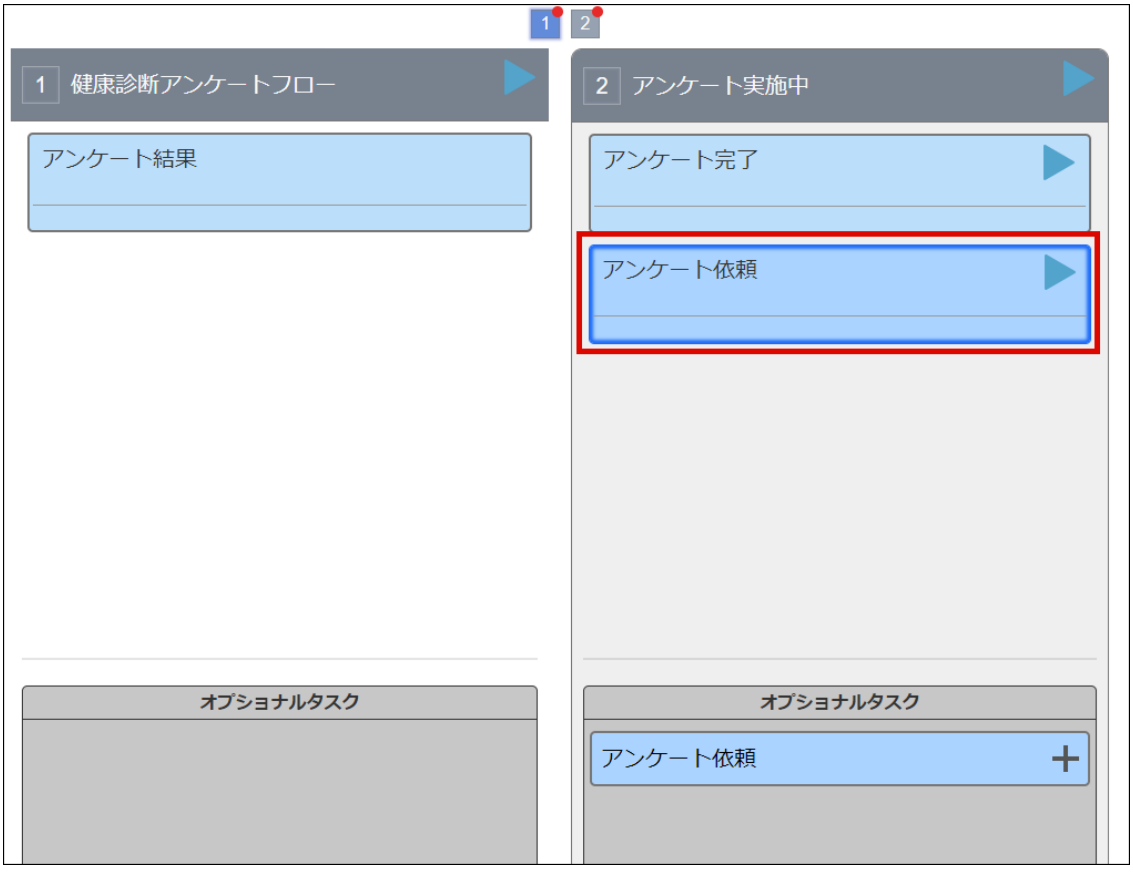
今回のチュートリアルでは結果確認動作を簡易化するため、全ての担当者に aoyagi を指定します。
まず、40歳以上のユーザに対するアンケートを作成するため、パラメータを以下のように設定して作成します。

- アンケート対象者：aoyagi
- 候補日From：任意（必須）
- 候補日To：任意（必須）
- 年齢：40
- 備考：任意

図：「タスクの追加」

> **注意**
> オプショナルタスクの「タスク追加時のパラメータ」には必須チェックが存在しないため、タスク発行の際には未入力の項目がないか注意してください。
> 実行中の範囲に対して追加したオプショナルタスクは編集できません。

7. 「アンケート依頼」のタスクが追加されたことを確認します。
   また、このオプショナルタスクは「複数追加を可能にする」と設定されているため、オプショナルタスク一覧にある「アンケート依頼」がグレーアウトしていないことを確認します。



図：「タスク追加」

8. 40歳未満に対するアンケートを作成します。
   「アンケート実施中」サブプロセスの中にあるオプショナルタスク「アンケート依頼」をクリックします。

図：「タスク追加」

9. 「タスクの追加」ダイアログが表示されます。
   40歳未満のユーザに対するアンケートを作成するため、パラメータを以下のように設定して作成します。
   - 担当者：aoyagi
   - 候補日From：任意（必須）
   - 候補日To：任意（必須）
   - 年齢：39
   - 備考：任意

図：「タスクの追加」

10. 「アンケート依頼」タスクの実行マークに「2」と表示され、かつ、オプショナルタスクはクリックできる状態であることを確認します。

図：「タスク追加」

11. 「アンケート依頼」タスクを処理するため、「タスク一覧」画面に戻ります。

12. 「タスク一覧」画面の「個人タスク一覧」に、「アンケート依頼」タスクが2件追加されているのを確認します。

図：「タスク一覧」 - 「個人タスク」

13. 40歳以上のユーザとしてアンケートに回答します。

先に作成された「アンケート依頼」タスクの「 」アイコンボタンをクリックします。

図：「タスク一覧」 - 「個人タスク」

14. FormaDesignerで作成された「健康診断アンケート」画面が表示されます。
    オプショナルタスクのパラメータで年齢を「40」と設定したため、Formaアプリケーションのアクション設定により「胃がん検診」の選択項目が表示されています。

図：健康診断アンケート（IM-FormaDesigner）

15. 任意の日付を選択し、今回は「バリウム」を選択して送信します。

図：健康診断アンケート（IM-FormaDesigner）

16. 「個人タスク」から、「アンケート依頼」タスクが1つだけ処理されていることを確認します。

図：「タスク一覧」 - 「個人タスク」

17. 40歳未満のユーザとしてアンケートに回答します。

    未処理である「アンケート依頼」タスクの「 」アイコンをクリックします。

図：「タスク一覧」 - 「個人タスク」

18. 「健康診断アンケート」画面が表示されます。
    オプショナルタスクのパラメータで年齢を「39」と設定したため、Formaアプリケーションの設定で「胃がん検診」の選択項目が表示されません。

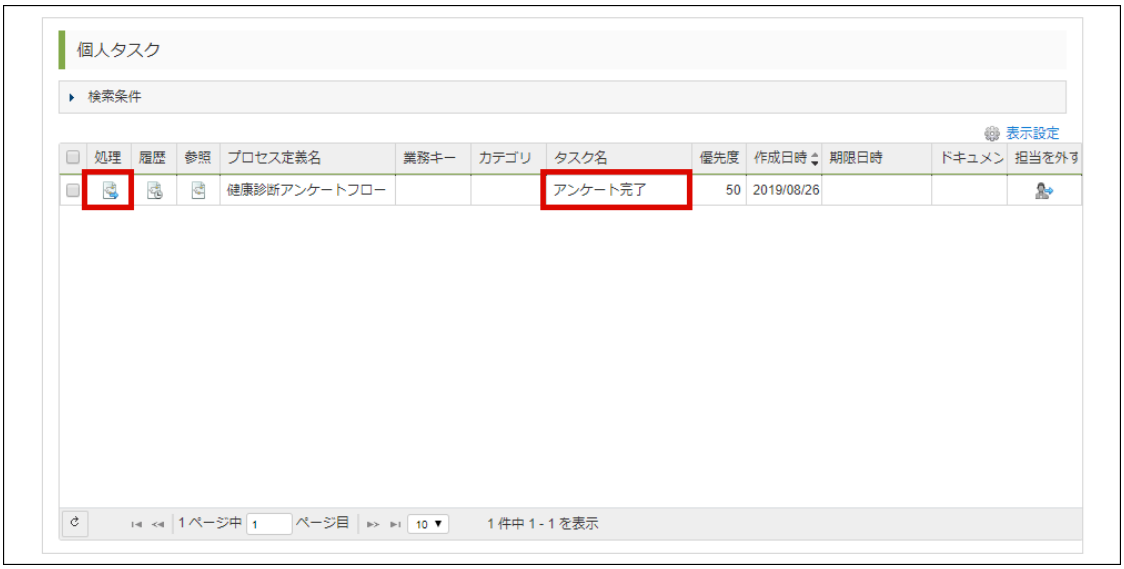
図：健康診断アンケート（IM-FormaDesigner）

19. 任意の日付を設定し、「送信」ボタンをクリックします。

図：健康診断アンケート（IM-FormaDesigner）

20. アンケートの集計結果を確認するため、「アンケート実施中」サブプロセスを完了させます。

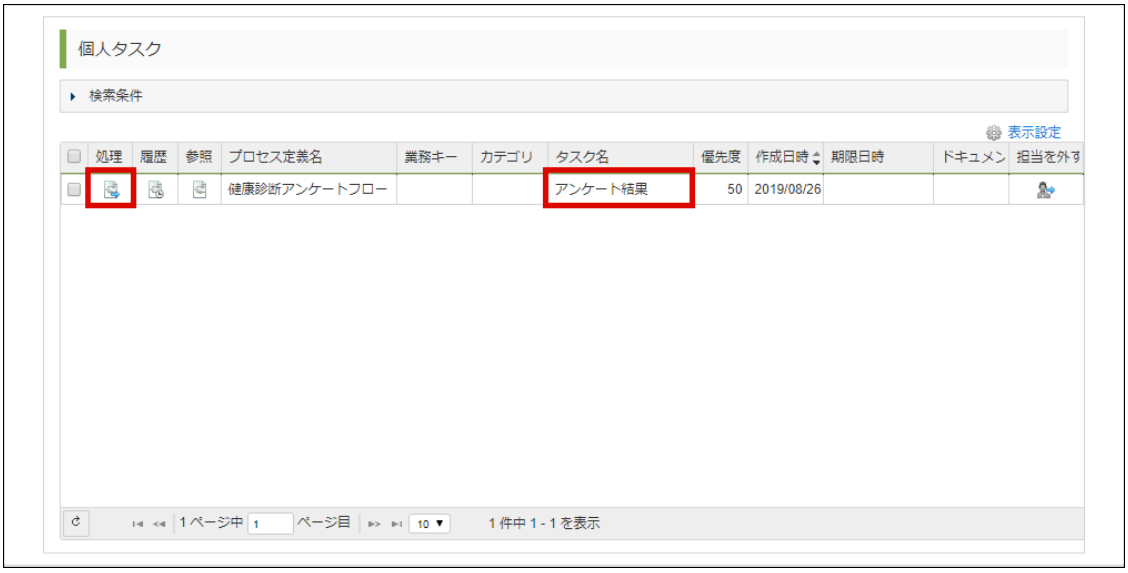
    個人タスクにある「アンケート完了」タスクの「 」アイコンをクリックし、タスクを処理します。



図：「タスク一覧」 -「個人タスク」

21. アンケートの集計結果を確認します。

    個人タスクにある「アンケート結果」タスクの「 」アイコンをクリックしします。



図：「タスク一覧」 -「個人タスク」

22. 「アンケート結果」画面が表示されます。
    上記の段取り通り、回答人数が「2」、バリウム希望者が「1」になっていることを確認し「確認」ボタンをクリックします。

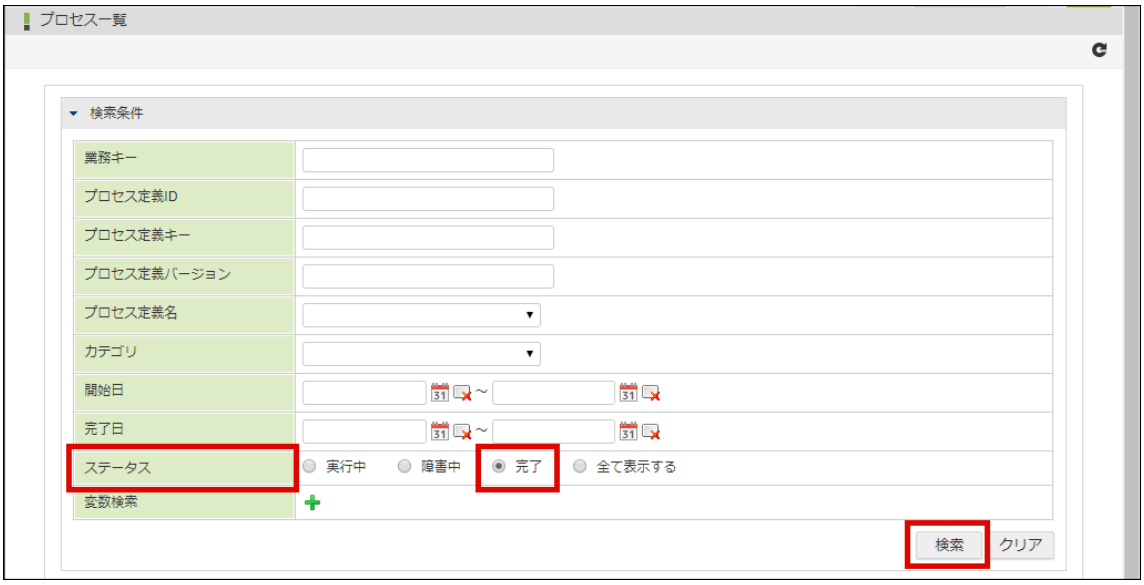
図：アンケート結果（IM-FormaDesigner）

23. 「健康診断アンケートフロー」が完了したため、「タスク一覧」画面の「個人タスク」にタスクがないことを確認します。

図：「タスク一覧」 - 「個人タスク」

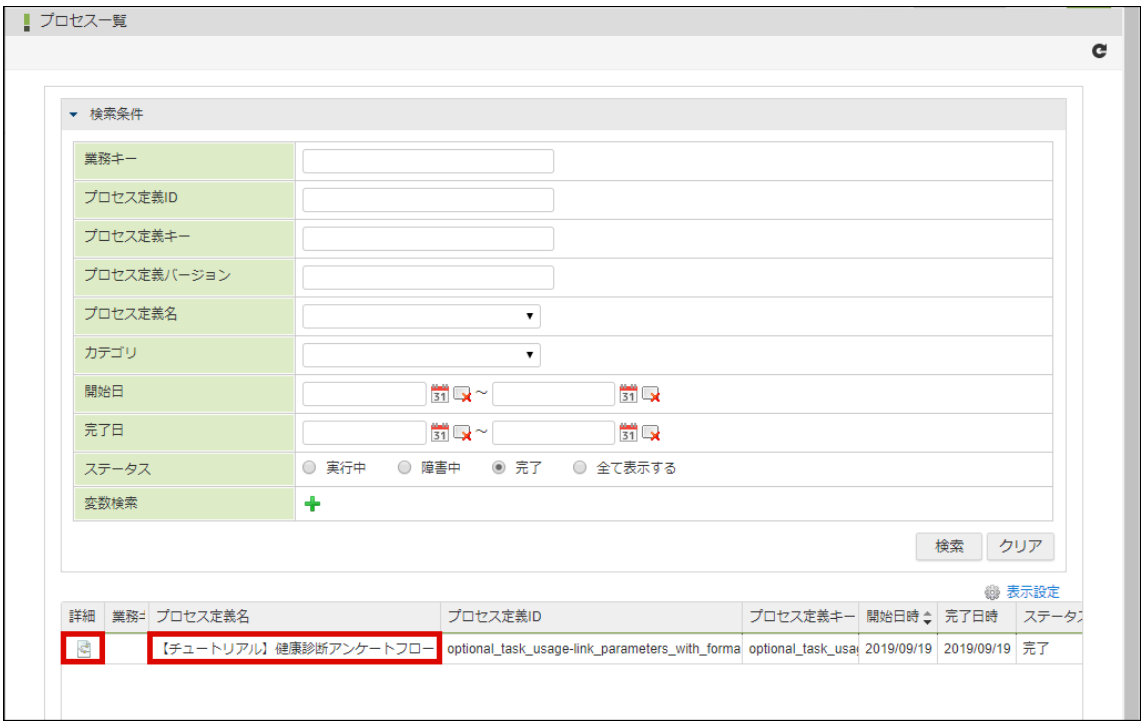
24. オプショナルタスクに設定したパラメータと、個人の回答結果を確認します。
「サイトマップ」→「BPM」→「プロセス一覧」画面を表示します。

25. 既に完了しているプロセスのため、初期表示のステータス「実行中」では表示されません。
「ステータス」のラジオボタンを「完了」、または、「全て表示する」にして「検索」ボタンをクリックします。



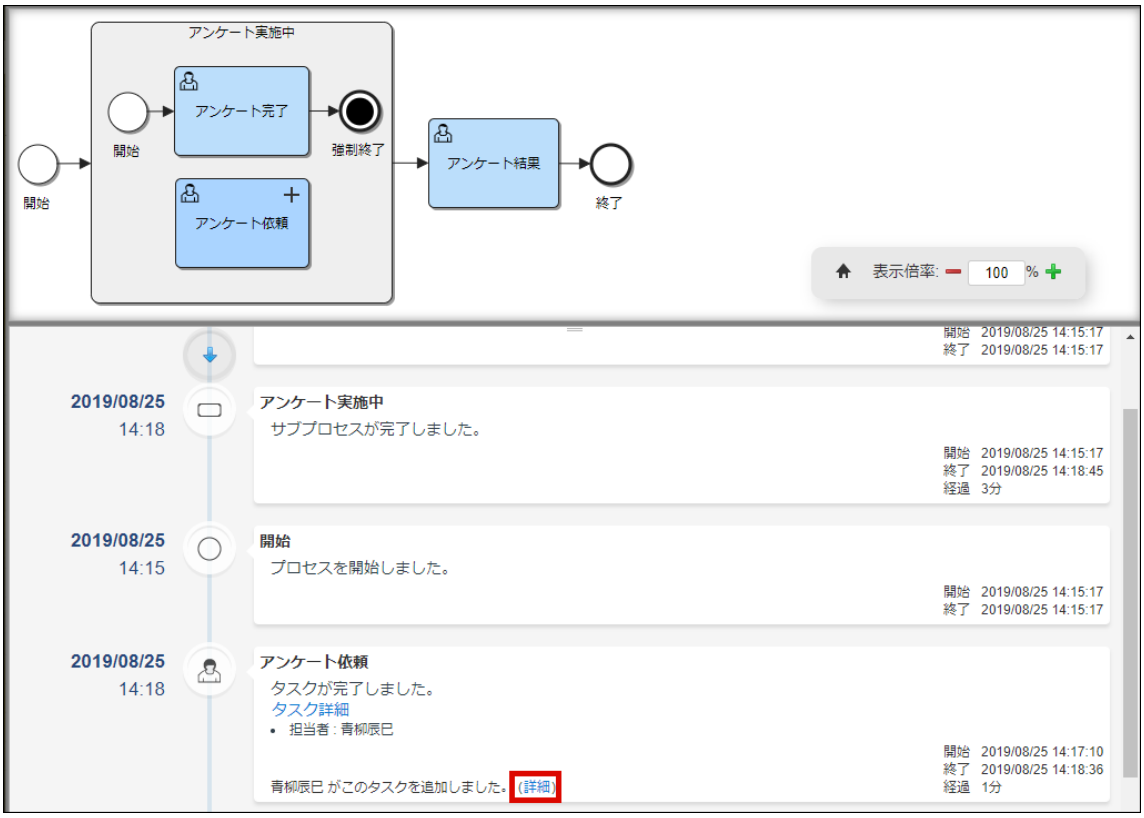
図：「プロセス一覧」

26. プロセス定義名「健康診断アンケートフロー」の「 」アイコンをクリックします。



図：「プロセス一覧」

27. オプショナルタスク詳細を確認します。
「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」画面から、プロセス履歴を表示します。
「アンケート依頼」タスクの「詳細」リンクをクリックします。

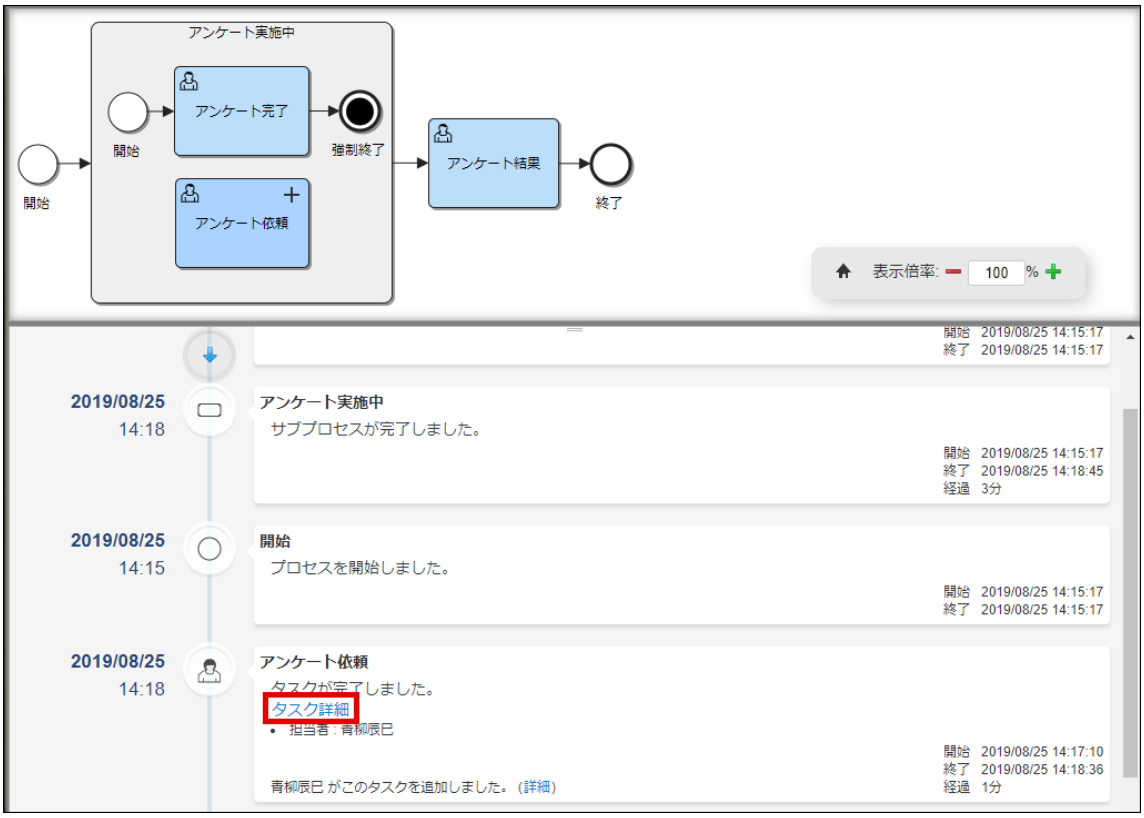


図：「プロセス図とタイムラインを拡大表示」

28. 「オプショナルタスク詳細」ダイアログが表示されます。
ここから、タスクを追加した際に設定したパラメータを確認できます。

図：「オプショナルタスク詳細」

29. 個人の回答結果を確認します。
プロセス履歴から「アンケート依頼」タスクの「タスク詳細」リンクをクリックします。



図：「プロセス図とタイムラインを拡大表示」

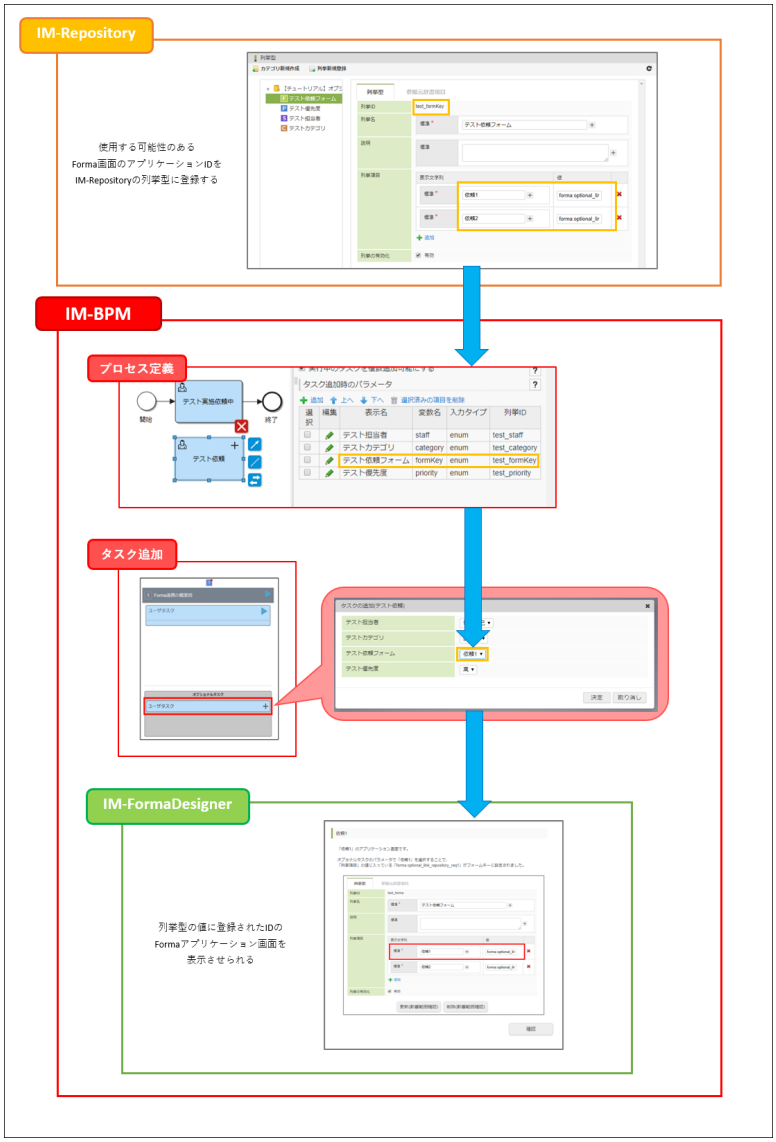
30. 新しいウィンドウで「健康診断アンケート」が表示されます。
ここから各個人の受診希望日、胃がん検診の希望検診内容などが確認できます。

図：健康診断アンケート（IM-FormaDesigner）

IM-Repositoryの列挙型をオプショナルタスクのパラメータで使用する

このチュートリアルでは、オプショナルタスクの「タスク追加」画面から追加するパラメータに対して「IM-Repository」の列挙型を使用する方法を解説します。
タスク追加時に使用するパラメータをあらかじめ「列挙型」として登録しておくことで、ユーザが追加できるパラメータを限定できます。

オプショナルタスクの詳細については、「IM-BPM 仕様書」 - 「オプショナルタスク」 もあわせて参照してください。
「IM-Repository」の詳細については、「IM-Repository ユーザ操作ガイド」 を参照してください。



図：概要図

このチュートリアルでは、既に「IM-Repository」に登録されている列挙型の確認は行いますが、新規登録は行いません。
解説に合わせて動作確認を行う場合、使用する列挙型のサンプルを以下のリンクからダウンロードし、インポートしてください。

im_repository-optional_task_usage-link_with_repository.json

列挙型のインポートについての詳細は「IM-Repository ユーザ操作ガイド」-「インポートを行う」を参照してください。

> **コラム**
>
> 列挙型の新規登録を実際に行いたい場合、「IM-Repository ユーザ操作ガイド」 - 「列挙」を参照してください。

プロセスを進めていく中で、「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材をインポートしてください。

im_forma_designer-optional_task_usage-optional_link_repository_req1.zip
im_forma_designer-optional_task_usage-optional_link_repository_req2.zip

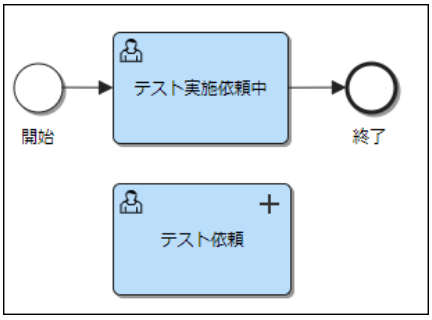
> **コラム**
>
> IM-FormaDesignerと連携する場合、FormaDesignerのユーザプログラムとして前処理、後処理の設定が必要です。
> フォームキーをEL式で定義している場合、デプロイ時に自動的に設定されません。
> 上記のサンプルには設定されていますが、各自で作成する場合は手動で登録する必要があります。
> 「IM-FormaDesigner連携」の詳細については、「IM-BPM 仕様書」 - 「IM-FormaDesigner連携」を参照してください。

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

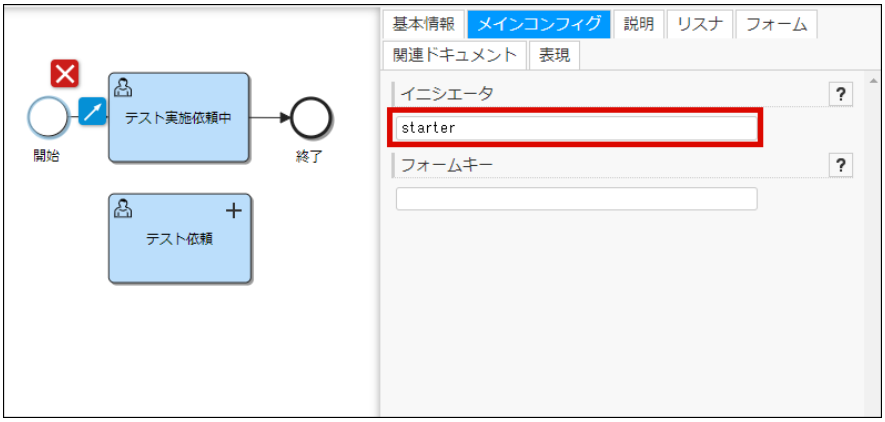
下記のプロセス図は、「テスト依頼フロー」です。
「テスト依頼フロー」を開始することで、開始者の個人タスク一覧に「テスト実施依頼中」タスクが滞在します。
「タスク追加」画面から、依頼したいテストの内容に沿ったパラメータを選択して「テスト依頼」タスクを実行します。
タスクのプロパティがパラメータを元に設定されるため、同じオプショナルタスクから依頼しても各タスクの情報や表示されるForma画面を変更できます。

- 「テスト実施依頼中」タスク
  オプショナルタスク「テスト依頼」を実行している間、フローを開始したユーザに滞在するタスクです。
  このタスクを保持している間に、オプショナルタスクを追加することでテストを依頼します。
- 「テスト依頼」タスク
  複数追加可能な設定がされているオプショナルタスクです。
  タスク追加時に選択できるパラメータとして、列挙型に登録された「表示文字列」をプルダウンに表示します。
  追加されるタスクのプロパティにはEL式が設定されており、選択した列挙型の「値」を取得して設定されます。



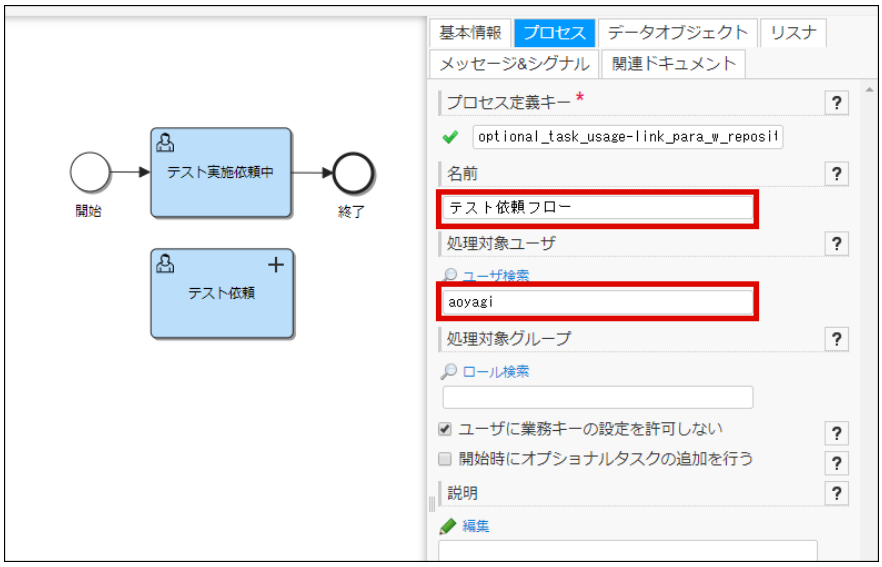
図：プロセス定義図

1. 「開始イベント」を設置します。
2. フローを開始したユーザを各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」タブから、「イニシエータ」に `starter` と設定します。



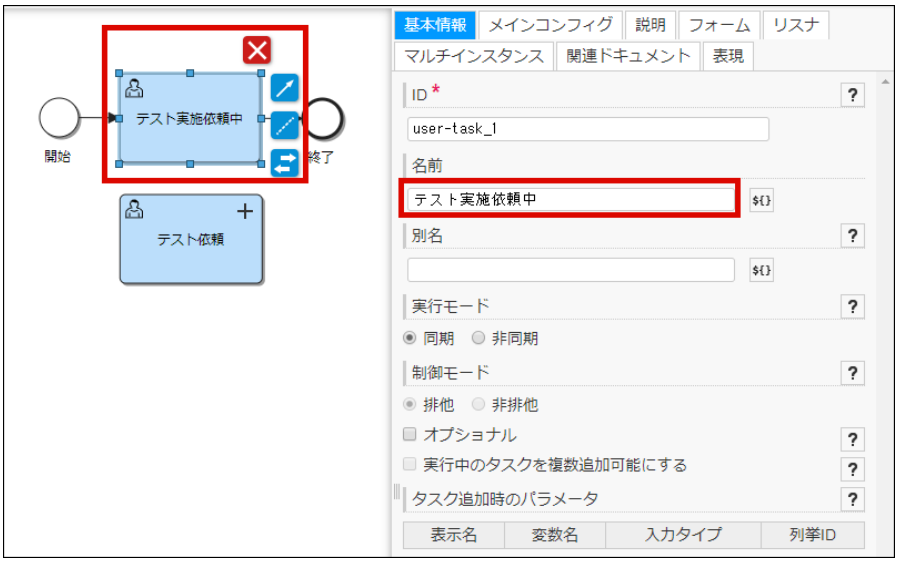
図：プロセス全体 - 「プロパティ」 - 「メインコンフィグ」

3. プロセス全体の設定を行います。
   キャンバ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、以下のように項目を設定します。
   - 名前：`テスト依頼フロー`
   - 処理対象ユーザ：`aoyagi`



図：プロセス全体-「プロパティ」 -「プロセス」

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」を選択した状態で、「基本情報」タブから「名前」に `テスト実施依頼中` と設定します。



図：「ユーザタスク」 -「プロパティ」 -「基本情報」

5. このプロセス定義を開始したユーザに対して「テスト実施依頼中」タスクが実行されるように、EL式を設定します。
「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 `${starter}` 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」

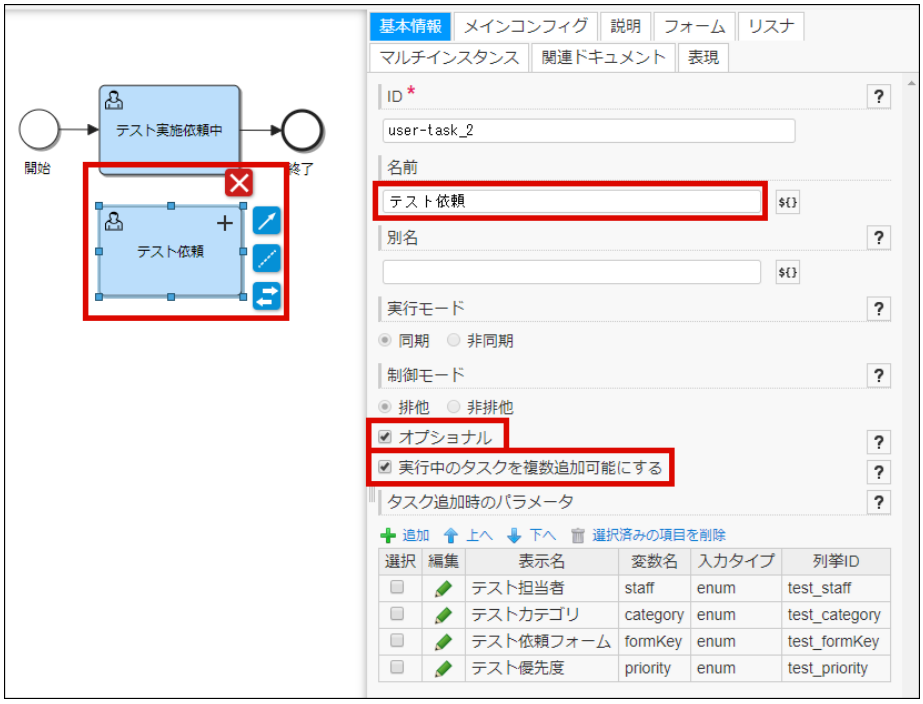
6. 「終了イベント」を設置します。

7. テストを依頼したいユーザに対して作成する「テスト依頼」タスクを設置します。
「ユーザタスク」を設置します。

8. 設置したユーザタスクを「オプショナルタスク」にします。

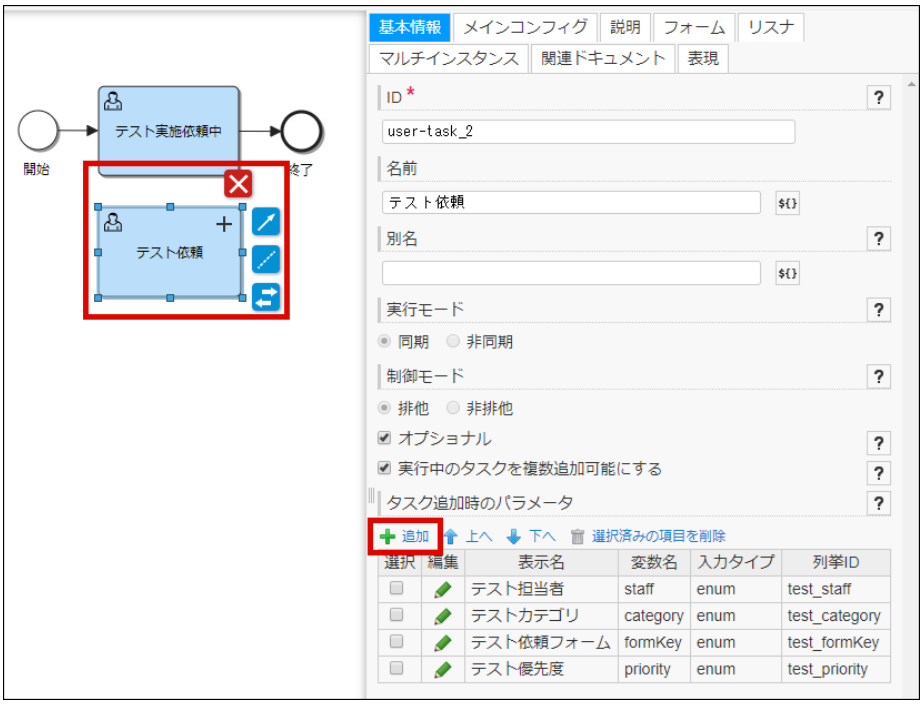
「ユーザタスク」を選択した状態で、「基本情報」タブから以下のように項目を設定します。

- 名前：テスト依頼
- オプショナル：有効
- 実行中のタスクを複数追加可能にする：有効



図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

9. 「オプショナルタスク」にパラメータを設定します。
「ユーザタスク」を選択した状態で、「基本情報」タブから「タスク追加時のパラメータ」にある「追加」リンクをクリックします。



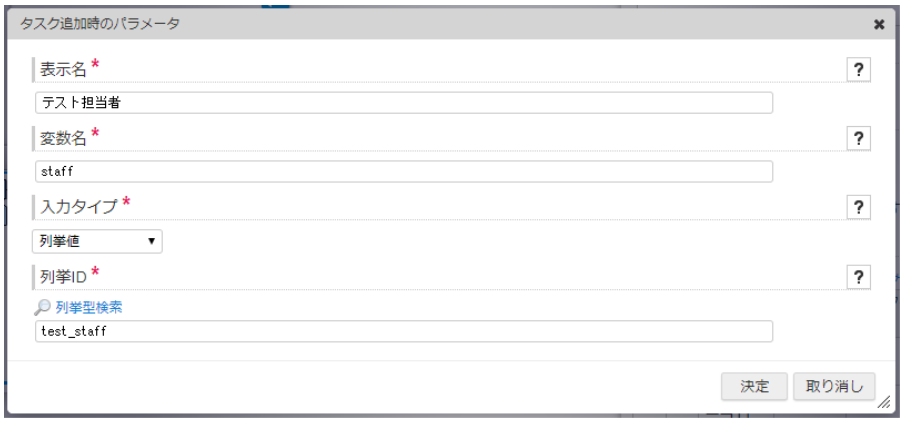
図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

10. 「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、4つのパラメータを設定します。

- テスト担当者
  表示名：テスト担当者
  変数名：staff
  入力タイプ：列挙値
  列挙ID：test_staff
- テストカテゴリ
  表示名：テストカテゴリ
  変数名：category
  入力タイプ：列挙値

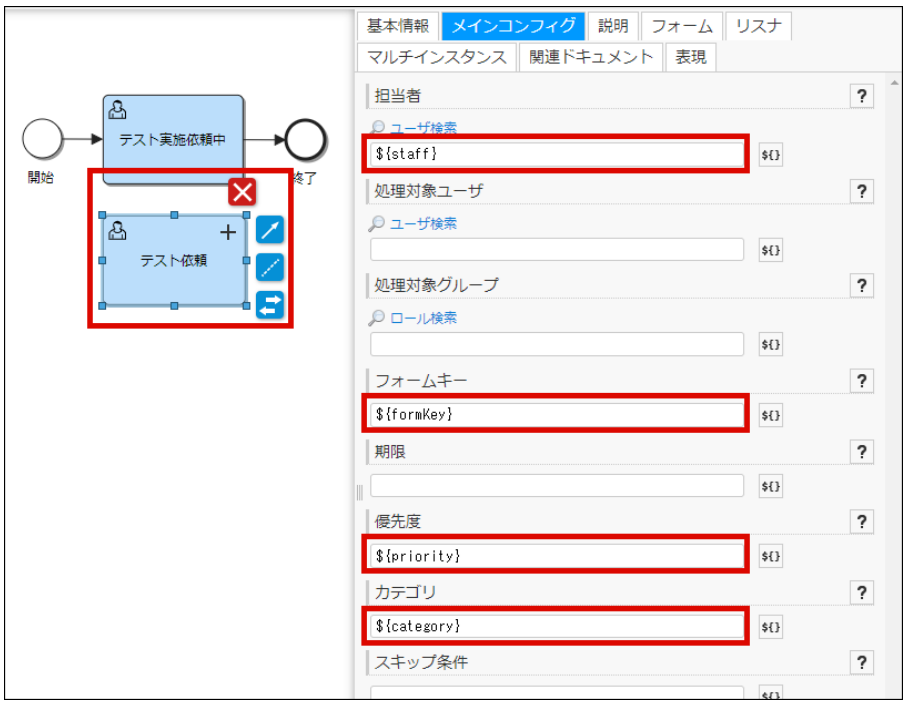
列挙ID：test_category

- テスト依頼フォーム
  表示名：テスト依頼フォーム
  変数名：formKey
  入力タイプ：列挙値
  列挙ID：test_formKey
- テスト優先度
  表示名：テスト優先度
  変数名：priority
  入力タイプ：列挙値
  列挙ID：test_priority



図：「タスク追加時のパラメータ」

11. 「タスク追加」画面で指定されたユーザを担当者に設定し、フォーム画面を紐づけるためEL式を設定します。
    「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：${staff}
    - フォームキー：${formKey}
    - 優先度：${priority}
    - カテゴリ：${category}



図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

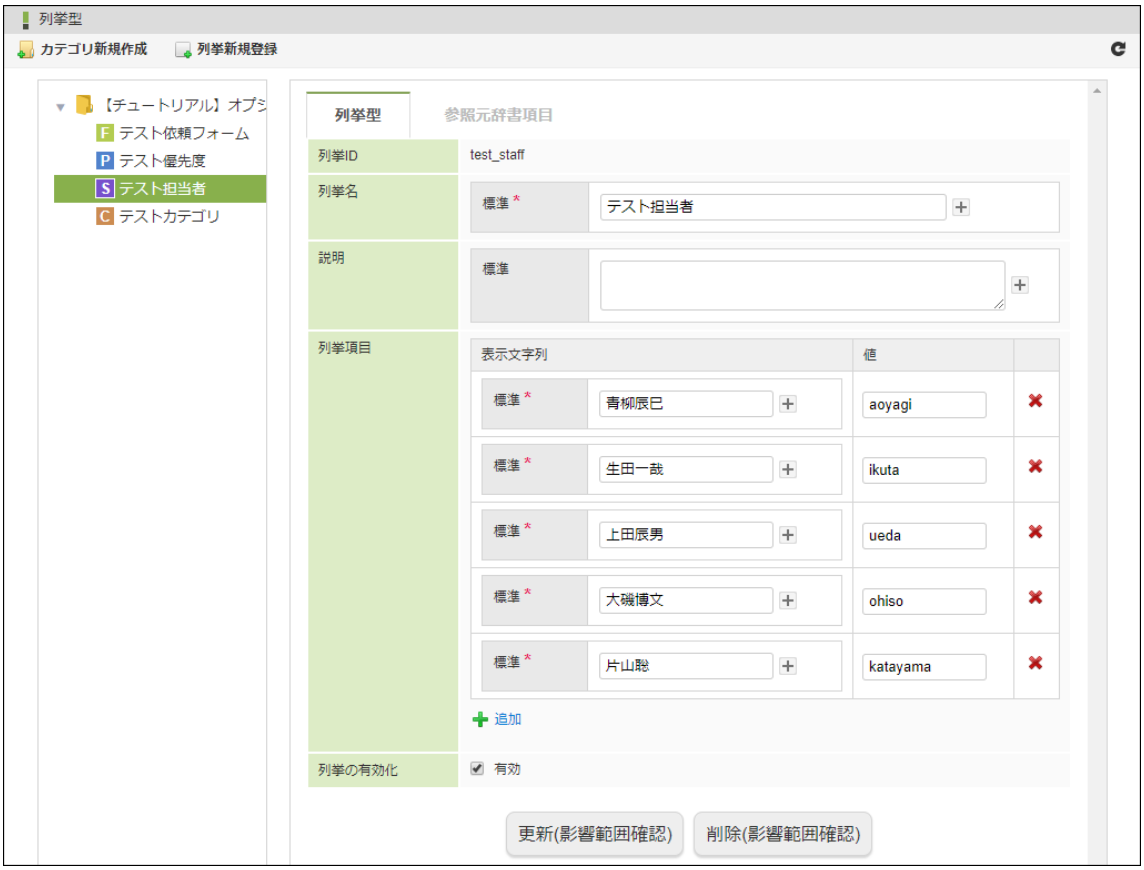
「IM-Repository」の列挙型を確認する

チュートリアル開始前にインポートした「IM-Repository」の列挙型を確認します。

1. 「サイトマップ」→「Repository」→「列挙型一覧」をクリックして、「列挙型」をクリックします。
2. 「オプショナルタスクのパラメータをIM-Repositoryの列挙型と連携する」カテゴリがあることを確認します。

図：「列挙型」

3. 列挙「テスト担当者」について確認します。
「タスク追加時のパラメータ」の「テスト担当者」に設定した列挙ID「test_staff」の内容です。
列挙項目の「表示文字列」に登録したユーザ名が、「タスクの追加」ダイアログから設定可能な候補としてプルダウンに表示されます。
列挙項目の「値」に登録したユーザIDはEL式「${staff}」で取得され、タスクを処理する担当者が設定されます。



図：「列挙型」 - 「テスト担当者」

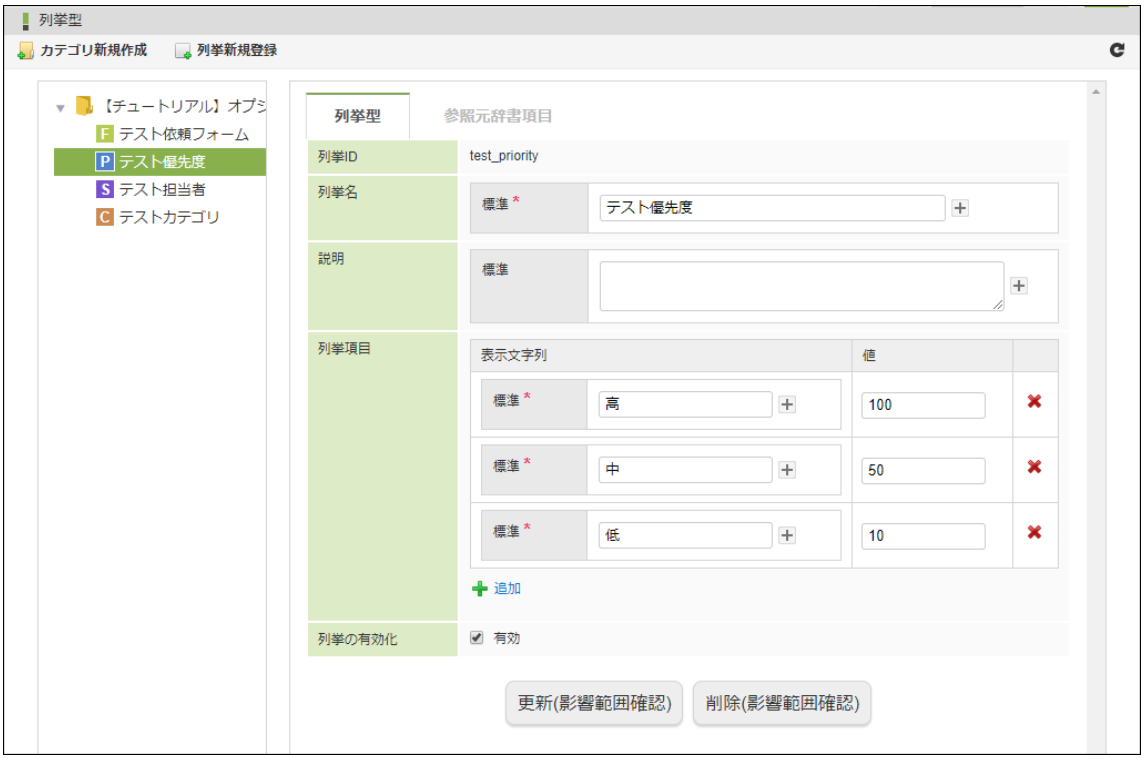
4. 列挙「テストカテゴリ」について確認します。
「タスク追加時のパラメータ」の「テストカテゴリ」に設定した列挙ID「test_category」の内容です。
列挙項目の「表示文字列」に登録した名称が、「タスクの追加」ダイアログから設定可能なテスト範囲としてプルダウンに表示されます。
列挙項目の「値」はEL式「${category}」で取得され、「個人タスク一覧」に表示されるタスクの「カテゴリ」に表示されます。

図：「列挙型」 - 「テストカテゴリ」

5. 列挙「テスト依頼フォーム」について確認します。
「タスク追加時のパラメータ」の「テスト依頼フォーム」に設定した列挙ID「test_formKey」の内容です。
列挙項目の「表示文字列」に登録したフォーム名称が、「タスクの追加」ダイアログから設定可能な候補としてプルダウンに表示されます。
列挙項目の「値」はEL式「${formKey}」で取得され、タスクを処理する際に表示するForma画面を設定します。

図：「列挙型」 - 「テスト依頼フォーム」

6. 列挙「テスト優先度」について確認します。
「タスク追加時のパラメータ」の「テスト優先度」に設定した列挙ID「test_priority」の内容です。
列挙項目の「表示文字列」に登録した指標が、「タスクの追加」ダイアログから設定可能な優先度としてプルダウンに表示されます。
列挙項目の「値」はEL式「${priority}」で取得され、「個人タスク一覧」に表示されるタスクの「カテゴリ」に表示されます。



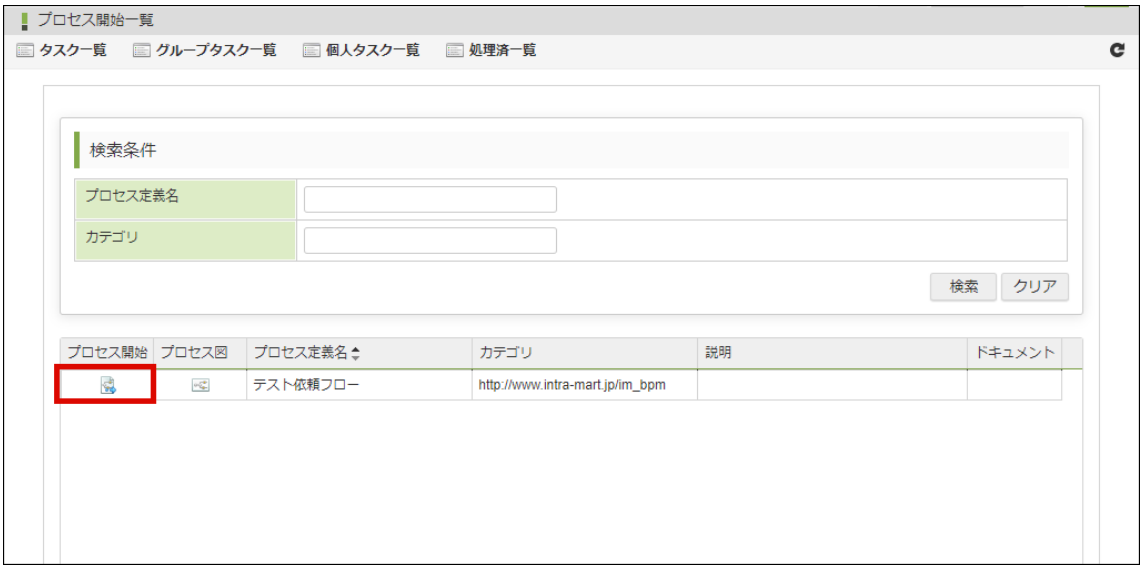
図：「列挙型」 - 「テスト優先度」

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. 「テスト依頼フロー」を開始します。

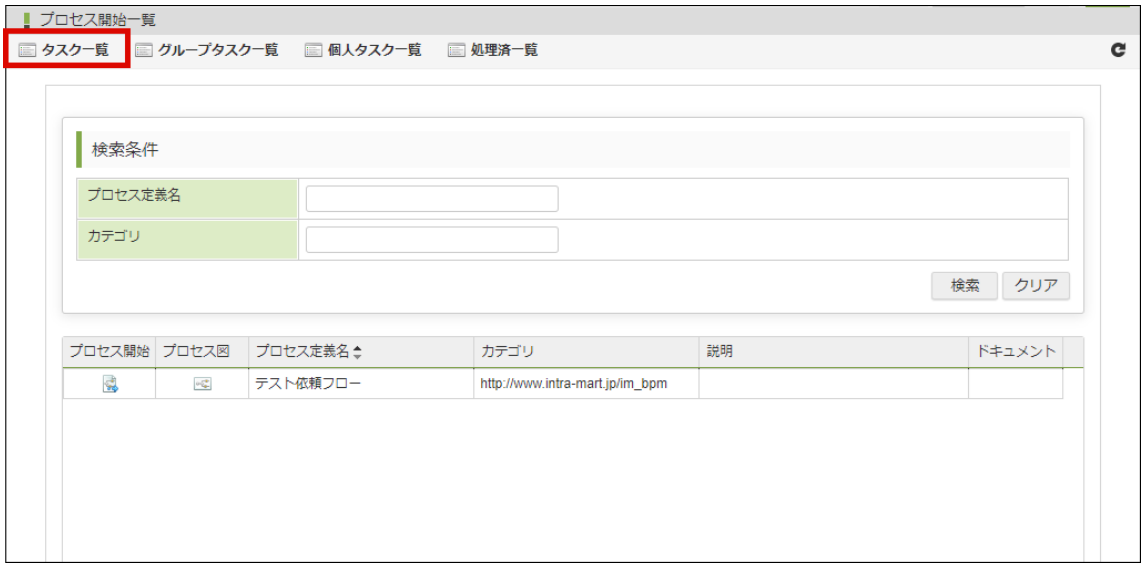
   「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示し、「テスト依頼フロー」の「 」アイコンをクリックします。



図：「プロセス開始一覧」

2. 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」

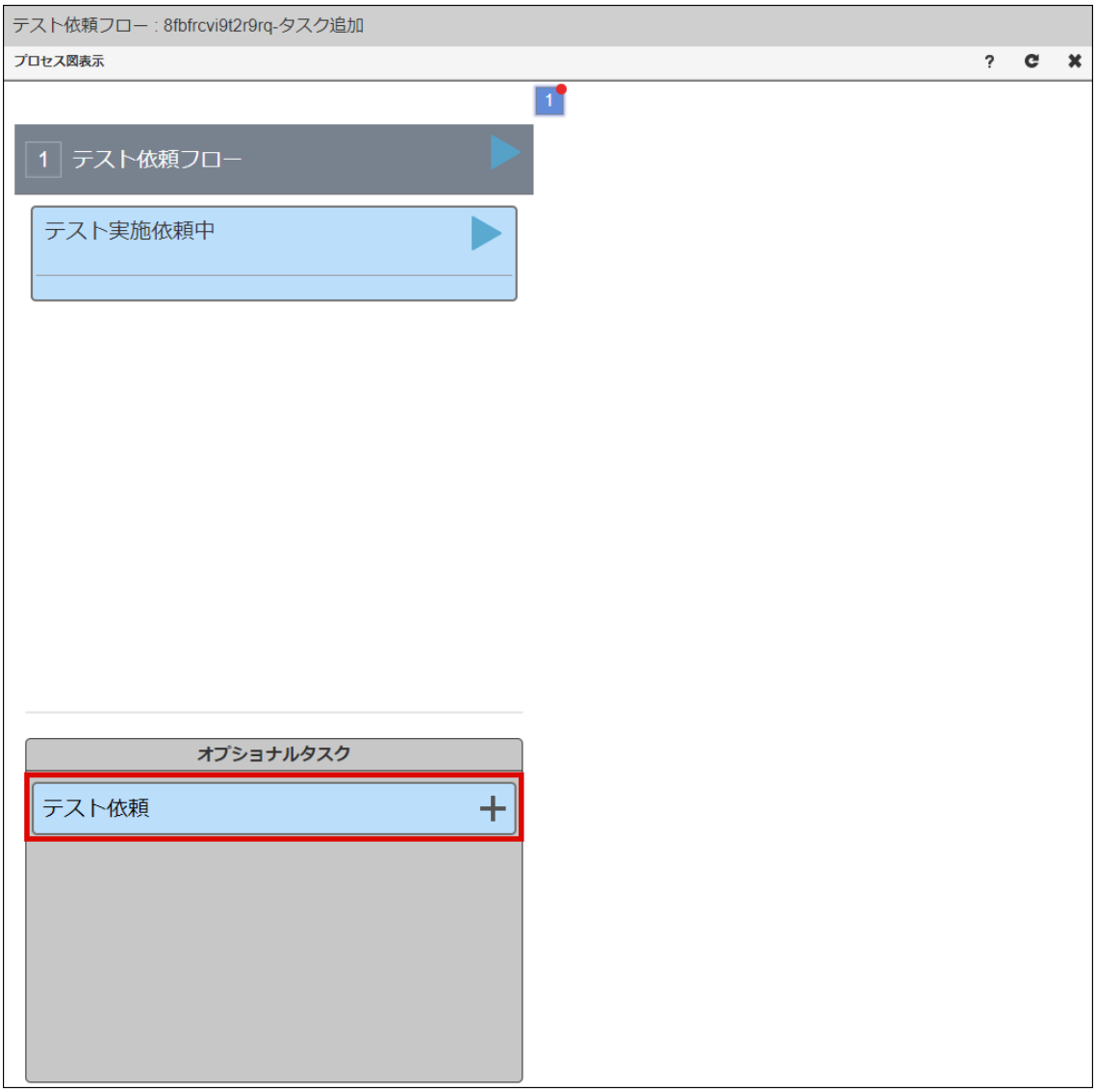
3. 開始したプロセスに対して「オプショナルタスク」を追加します。
   「タスク一覧」画面の「個人タスク」にある「テスト依頼フロー」の「 」をクリックし、「プロセス参照」画面を表示します。

図：「タスク一覧」 - 「個人タスク」

4. 開始したプロセスに対して「オプショナルタスク」を追加します。
   「プロセス参照」画面のツールバーから、「タスク追加」ボタンをクリックします。

図：「プロセス参照」

5. 開始したプロセスに対して「オプショナルタスク」を追加します。
「テスト依頼フロー」の直下にあるオプショナルタスク「テスト依頼」をクリックし、「タスクの追加」ダイアログを表示します。



図：「タスク追加」

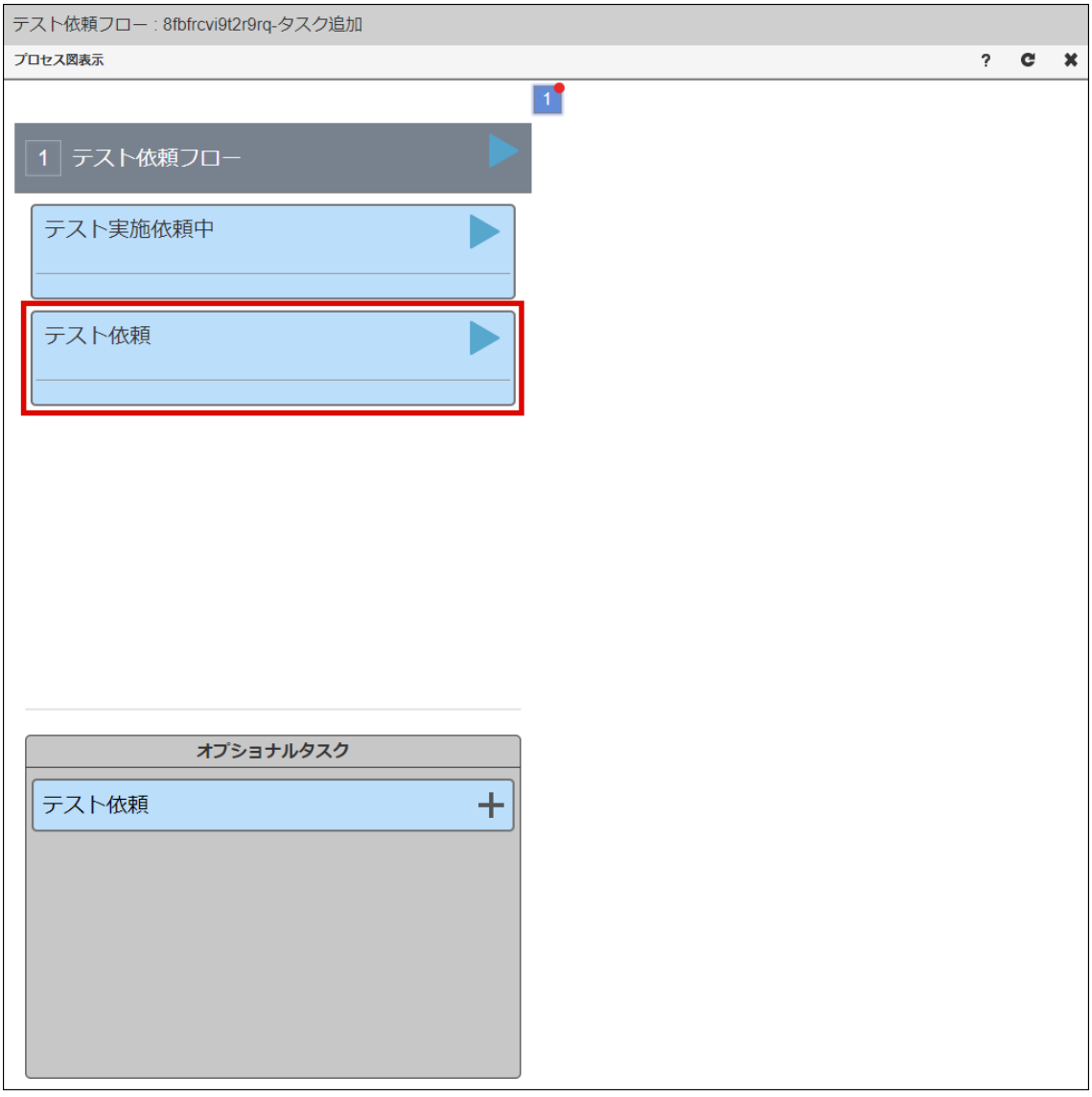
6. 「タスクの追加」ダイアログが表示されます。
今回のチュートリアルでは結果確認動作を簡易化するため、全ての担当者に「 青柳辰巳 」を指定します。

下記のように項目を設定することで「青柳さんに対して優先度の高い画面のテストを『依頼1』のFormaアプリケーション画面を使用して依頼」します。

- テスト担当者：青柳辰巳
- テストカテゴリ：画面
- テスト依頼フォーム: 依頼1
- テスト優先度：高

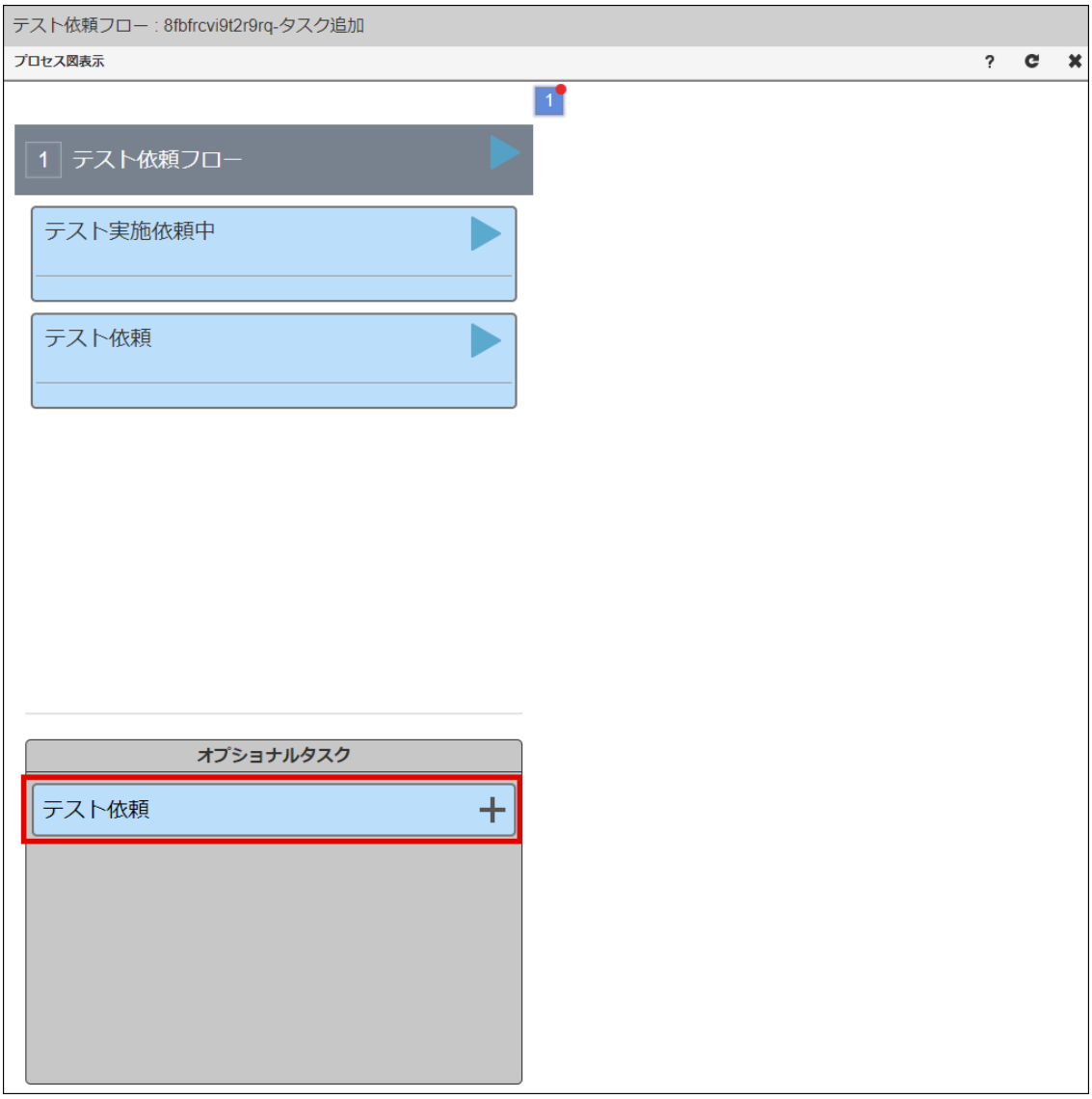
図：「タスクの追加」

7. 「テスト依頼」タスクが「タスク一覧」に追加され、かつ、「オプショナルタスク」はクリックできる状態であることを確認します。



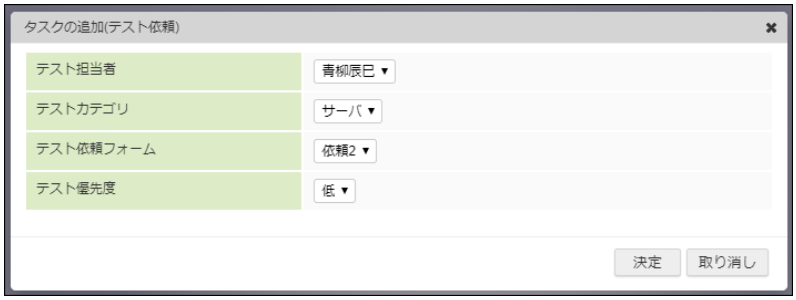
図：「タスク追加」

8. 結果を比較するためにもう一つオプショナルタスクを追加します。
   オプショナルタスク一覧の「テスト依頼」をクリックします。



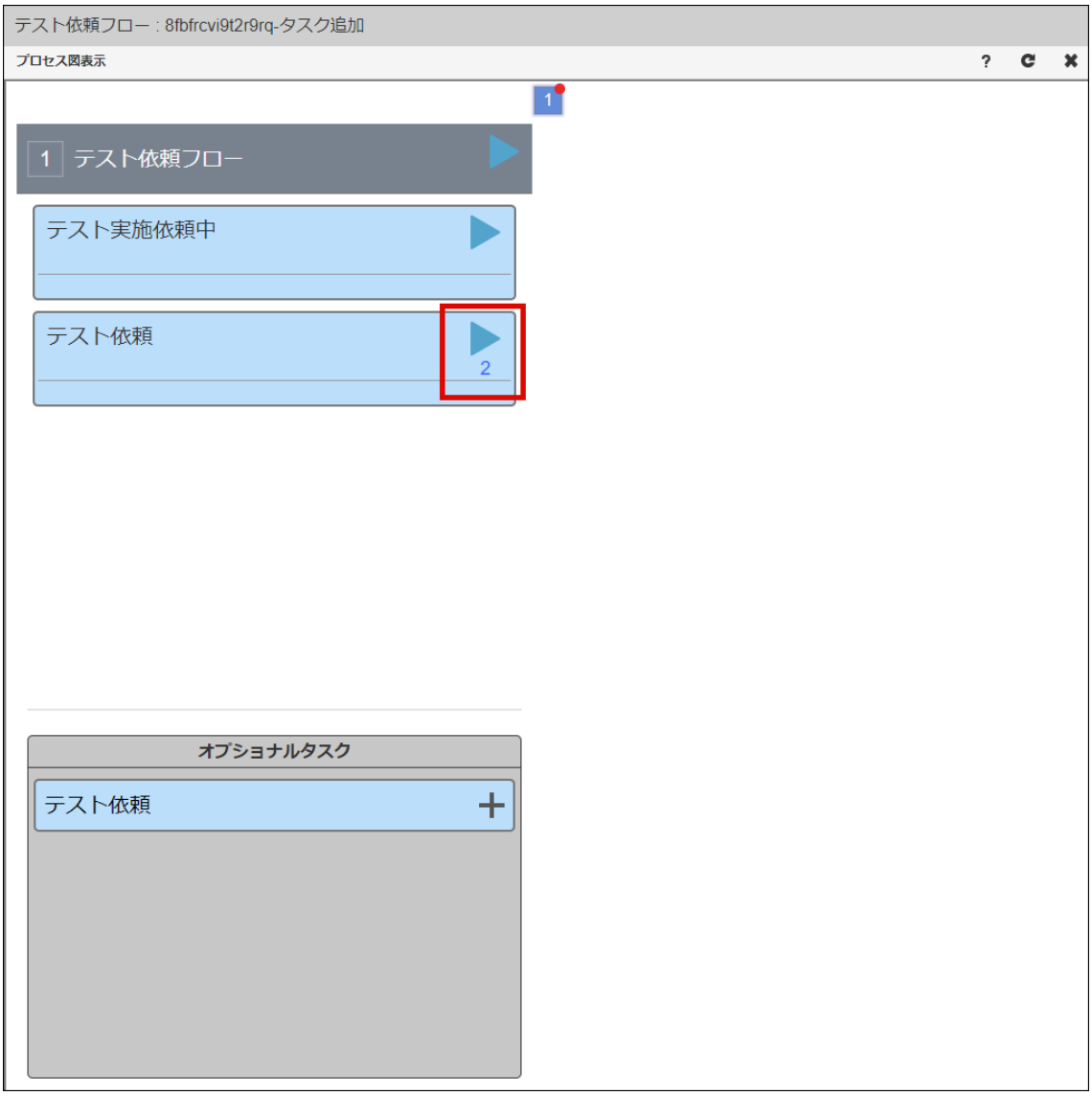
図：「タスク追加」

9. 「タスクの追加」ダイアログが表示されます。
   結果確認動作を簡易化するため、このタスクの担当者にも「 青柳辰巳 」を指定します。
   下記のように項目を設定することで「青柳さんに対して優先度の低いサーバのテストを『依頼2』のFormaアプリケーション画面を使用して依頼」します。
   - テスト担当者：青柳辰巳
   - テストカテゴリ：サーバ
   - テスト依頼フォーム: 依頼2
   - テスト優先度：低



図：「タスクの追加」

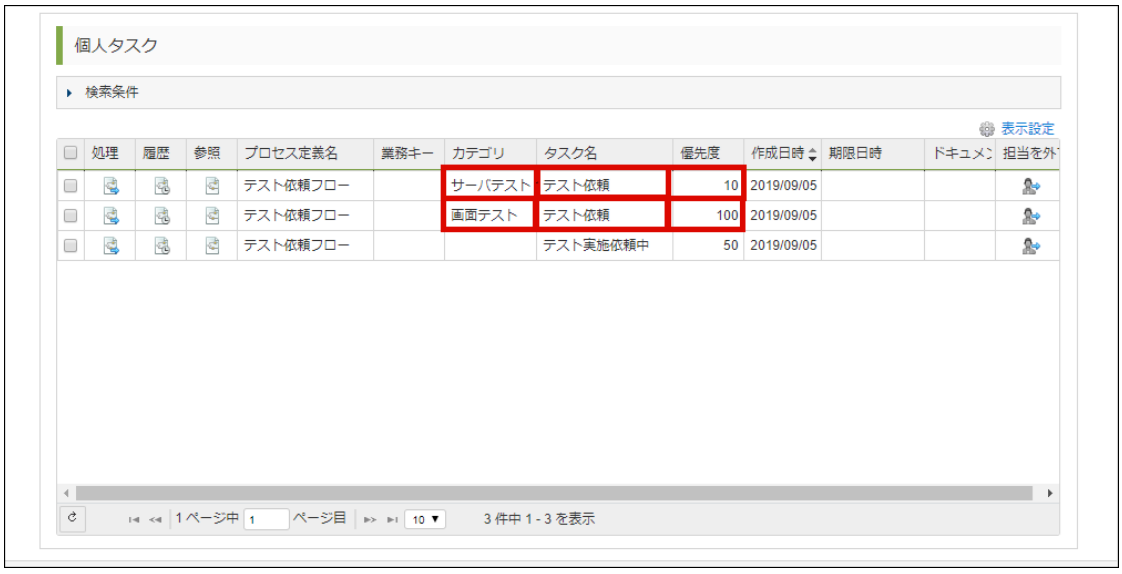
10. 「テスト依頼」タスクの実行マークに「2」と表示されことを確認します。



図：「タスク追加」

11. 各「テスト依頼」タスクを確認するため、「タスク一覧」画面に戻ります。

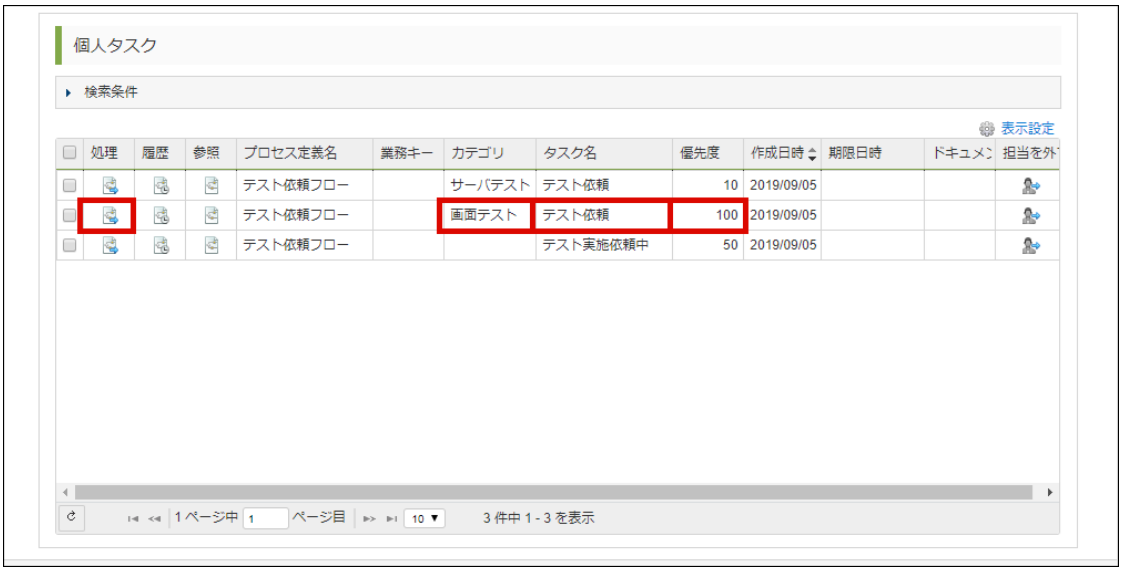
12. 「タスク一覧」画面の「個人タスク一覧」に、下記のような設定のタスクがあることを確認します。
    - カテゴリが「画面テスト」で優先度の高い（100）「テスト依頼」タスク
    - カテゴリが「サーバテスト」で優先度の低い（10）「テスト依頼」タスク



図：「タスク一覧」 - 「個人タスク」

13. 優先度の高いタスクから確認します。

    優先度が「100」と表示されているタスクの「 」ボタンをクリックします。



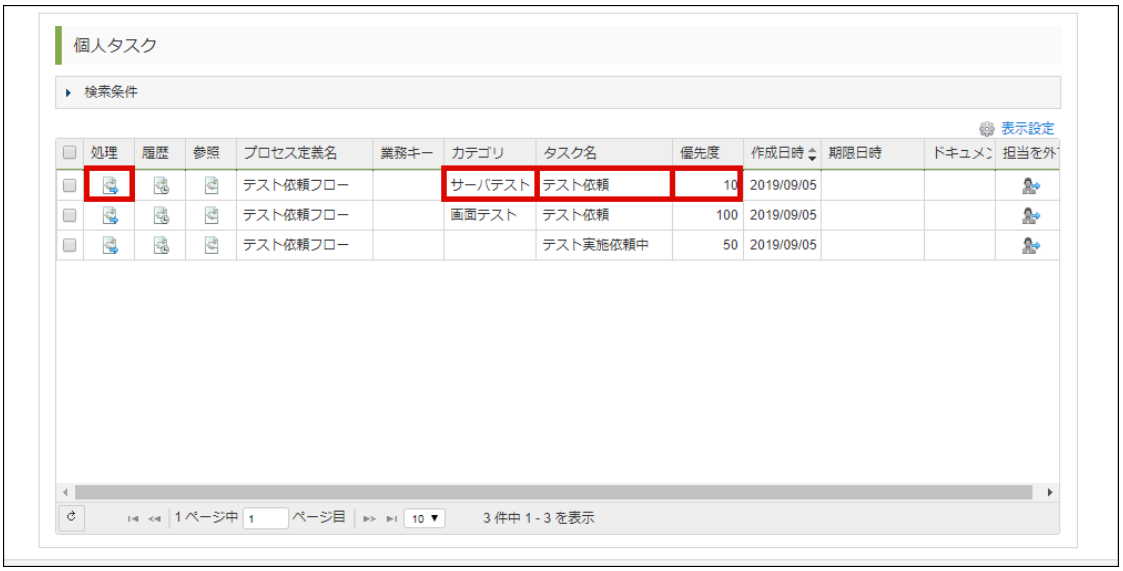
図：「タスク一覧」 - 「個人タスク」

14. タスク追加時に表示文字列の「依頼１」を選択したため、Formaアプリケーション「optional_link_repository_req1」が表示されます。
「確認」ボタンをクリックします。

図：依頼1（IM-FormaDesigner）

15. 次に、優先度の低いタスクを確認します。

優先度が「10」と表示されているタスクの「 」ボタンをクリックします。



図：「タスク一覧」 - 「個人タスク」

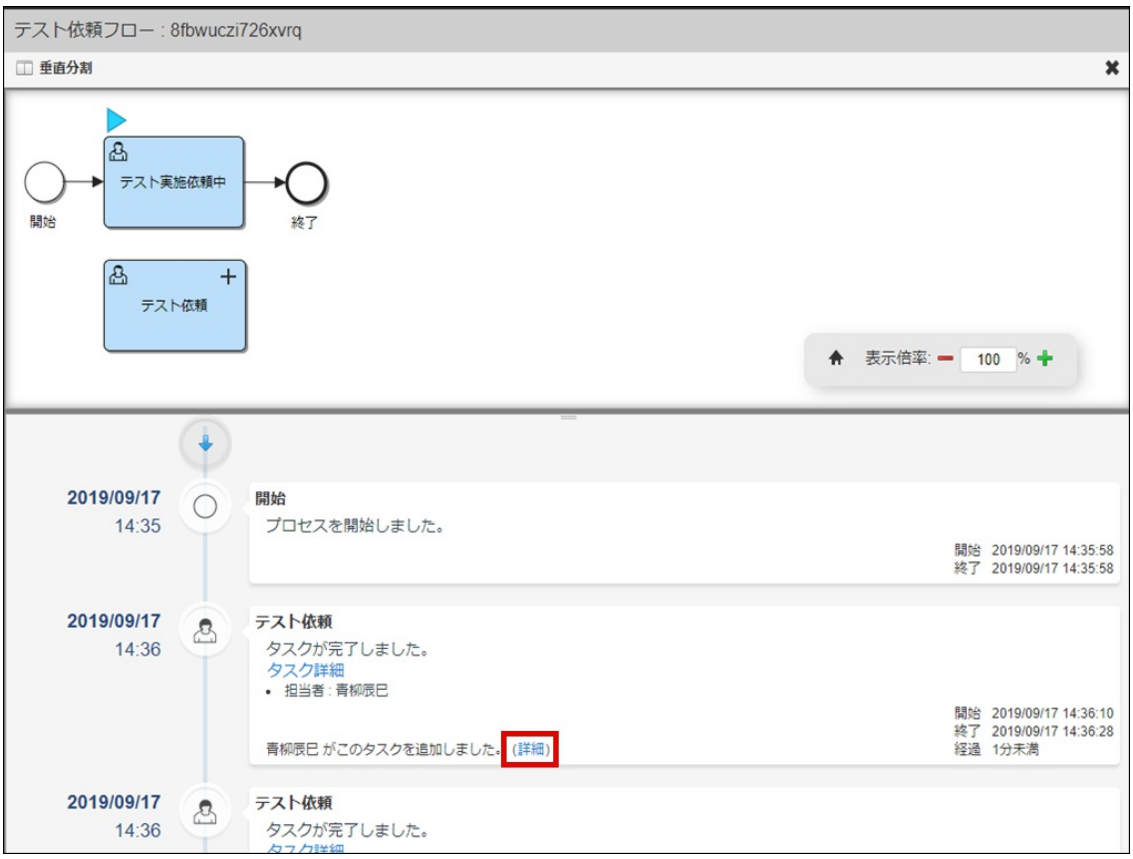
16. タスク追加時に表示文字列の「依頼2」を選択したため、Formaアプリケーション「optional_link_repository_req2」が表示されます。
「確認」ボタンをクリックします。

図：依頼2（IM-FormaDesigner）

17. 「タスク追加」画面で追加したパラメータの内容を確認します。
「タスク一覧」画面に戻り、個人タスクにある「テスト実施中」タスクの「参照」ボタンをクリックします。

図：「タスク一覧」 -「個人タスク」

18. 「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」画面から、プロセス履歴を確認します。
    最初に作成した「テスト依頼」タスクの「 」リンクをクリックします。



図：「プロセス図とタイムラインを拡大表示」

19. 「オプショナルタスク詳細」ダイアログが表示されます。
    「タスク追加」画面の「タスクの追加」ダイアログで選択した表示文字列が確認できます。

図：「オプショナルタスク詳細」

アドホックタスク

アドホックタスクを利用する

このチュートリアルでは、アドホック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。
アドホックタスクは、プロセス作成時に定義しきれなかった隠れたタスクを、プロセスの実行中に追加することのできるタスクです。

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

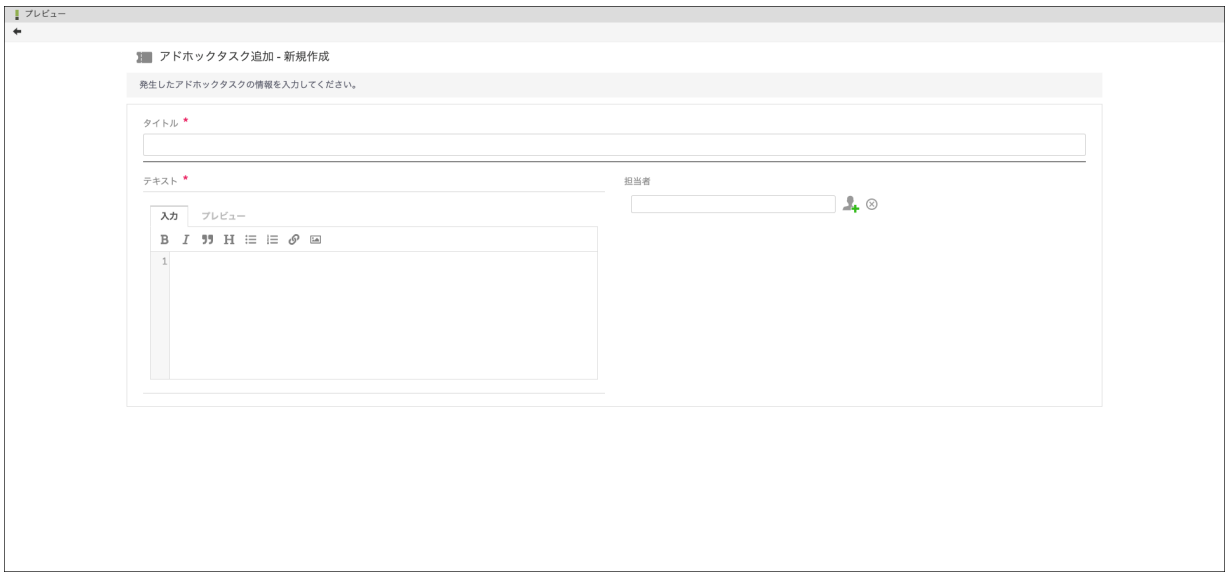
- チケットマスタを作成する
- プロセス定義を作成する
- アドホックタスクを追加する
- アドホックタスクのチケットを更新する
- チケットの履歴の確認/コメントをする
- アドホックタスクを処理する
- アドホックタスクが追加可能なタイミングを確認する

チケットマスタを作成する

アドホックタスクを利用するために、アドホックタスク追加時に表示するフォームを、チケットモジュールを利用して作成します。

> **コラム**
>
> チケットモジュールを利用するためには「チケット管理者」ロールと「IM-Repository 管理者」ロールが付与されている必要があります。
> チケットマスタの作成の詳細については以下のリンクを参照してください。
>
> - 「チケットモジュール管理者操作ガイド」→「チケットマスタカテゴリを作成する」
> - 「チケットモジュール管理者操作ガイド」→「チケットマスタを作成する」

下記の図は作成するチケットマスタのイメージです。



図：完成イメージ（チケットマスタ）

1. 「サイトマップ」→「Ticket」→「チケットマスタ管理」から、「チケットマスタ管理」画面を表示します。

2. チケットマスタカテゴリを作成します。「 」をクリックします。

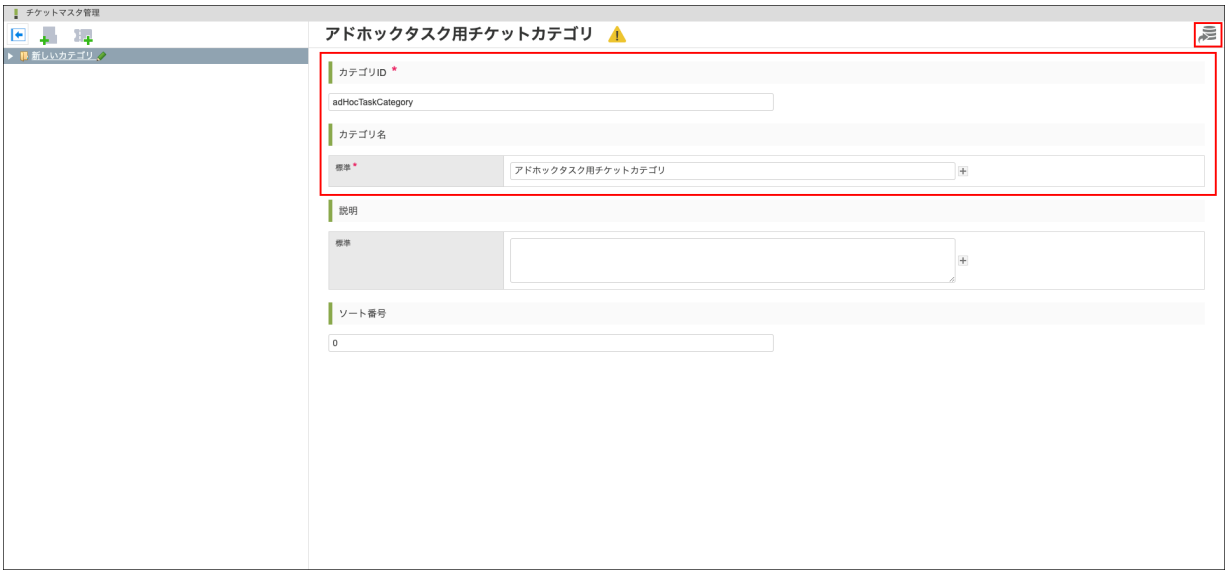
図：「チケットマスタ管理」

3. チケットマスタカテゴリの設定をします。

   表示されたフォームに対して以下のように項目を設定してください。

   - カテゴリID：adHocTaskCategory
   - カテゴリ名：アドホックタスク用チケットカテゴリ

   「 」をクリックします。



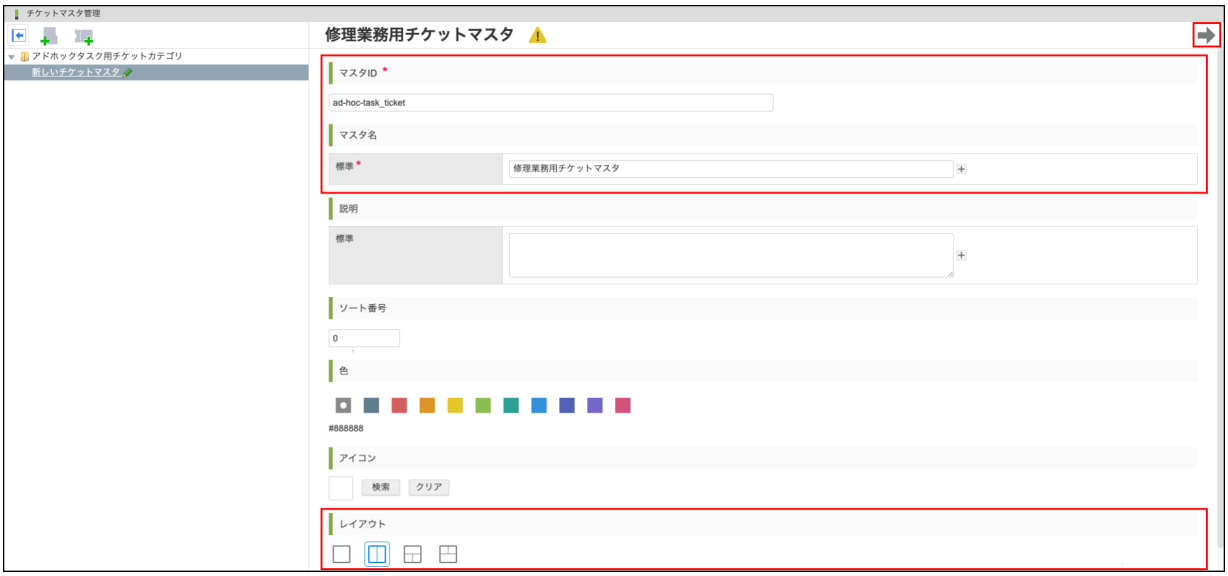
図：「チケットマスタ管理」 - 「カテゴリを追加」

4. チケットマスタの設定をします。

   表示されたフォームに対して、以下のように項目を設定します。
   ここで設定する「マスタID」が、IM-BPMのプロセス定義でアドホックタスクの設定を行う際に必要です。

   - マスタID：ad-hoc-task_ticket
   - マスタ名：修理業務用チケットマスタ
   - レイアウト：二列

   「 」をクリックします。



図：「チケットマスタ管理」 - 「チケットマスタ設定」

5. チケットマスタのフィールドの設定をします。

   表示されたフォームに対して、以下のように項目を設定します。

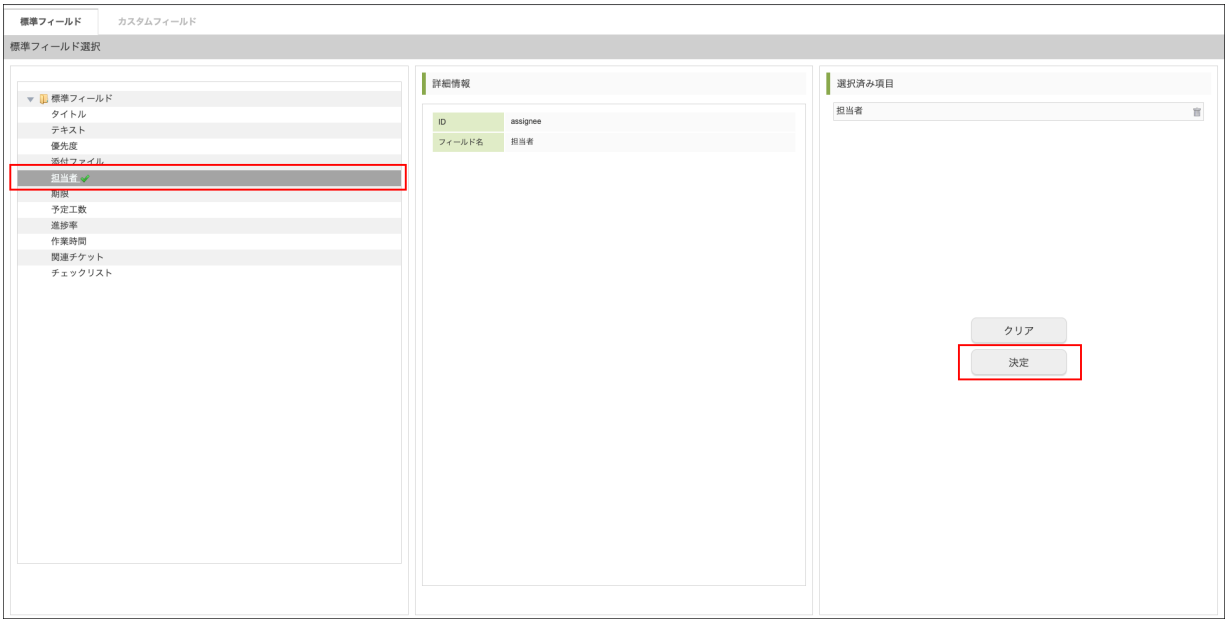
   - 見出し：修理プロセス - アドホックタスク
   - 概要：発生したアドホックタスクの情報を入力してください。

図：「チケットマスタ管理」 - 「フィールド設定」

6. 「 」をクリックします。

図：「チケットマスタ管理」 - 「レイアウト設定」 - 「標準フィールド選択」

7. 「標準フィールド選択」画面が表示されるので、「担当者」を選択して「決定」をクリックします。



図：「チケットマスタ管理」 - 「レイアウト設定」 - 「標準フィールド選択」

8. 「担当者」がレイアウトに追加されました。「 」をクリックし、チケットマスタは完成です。

図：「チケットマスタ管理」 - 「レイアウト設定」

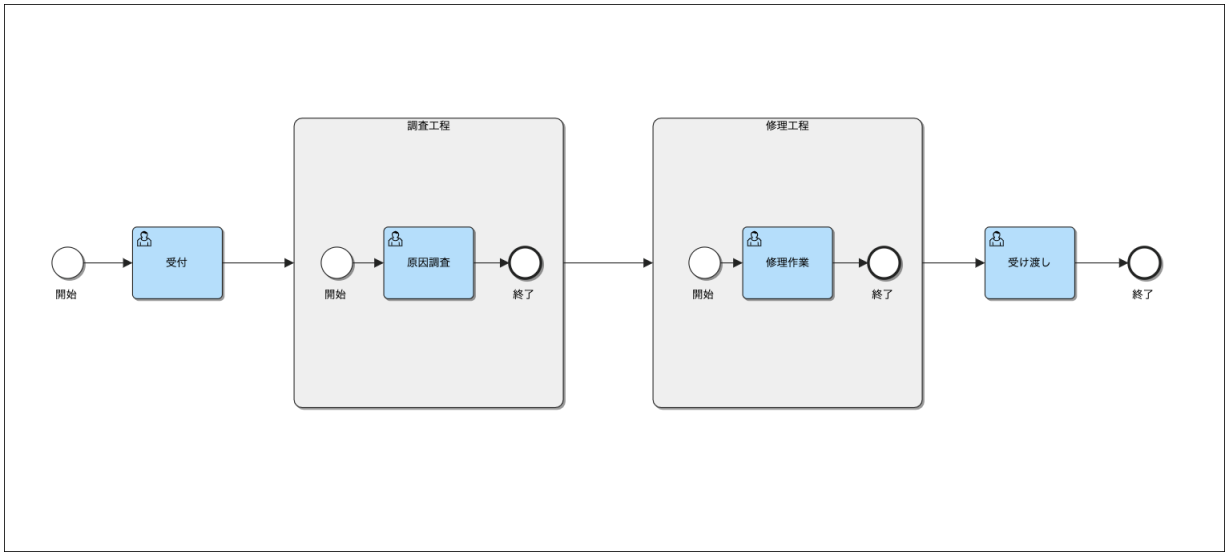
コラム

チケットマスタで以下の項目を設定できるようにした場合は、IM-BPMのアドホックタスクのそれぞれの項目と連動します。

- タイトル
- 担当者
- 優先度
- 期限

プロセス定義を作成する

アドホックタスクを追加可能としたプロセス定義を作成します。
下記の図は、本チュートリアルガイドにて利用するプロセス定義です。



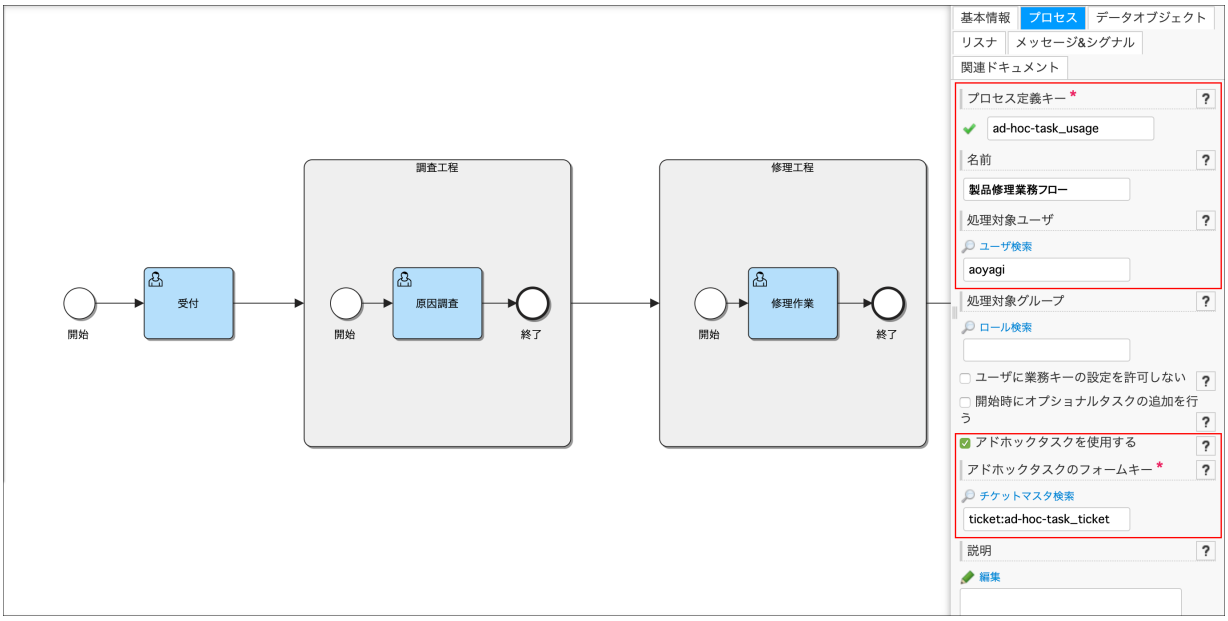
図：プロセス定義図

アドホックタスクを使用するためには、プロセス定義作成時にアドホックタスクの使用を許可する必要があります。

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、アドホックタスクを使用するように設定をします。

   キャンパスの空白部分をクリックし、プロパティエリア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ます。
   ここで設定する「アドホックタスクのフォームキー」が、作成したチケットマスタの「マスタID」です。

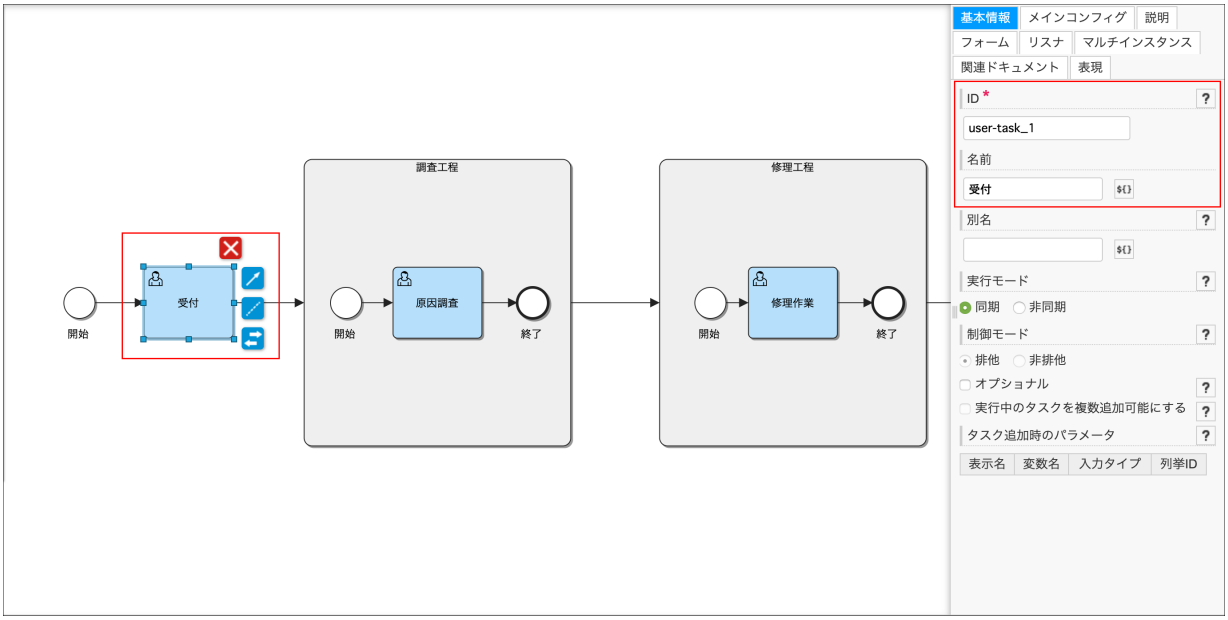
   - プロセス定義キー：ad-hock-task_usage
   - 名前：製品修理業務フロー
   - 処理対象ユーザ：aoyagi
   - アドホックタスクを使用する：有効
   - アドホックタスクのフォームキー：ticket:ad-hoc-task_ticket



図：「プロセス全体」 - 「プロパティ」 - 「プロセス」

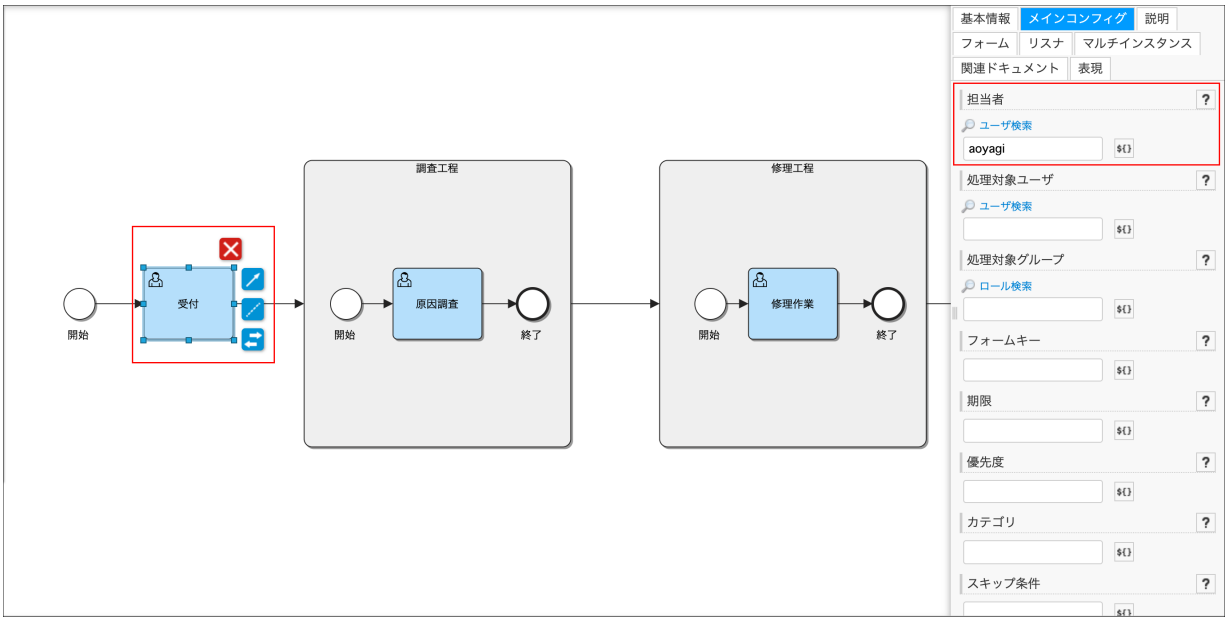
3. プロセス実行時の各タイミングでアドホックタスクの追加が可能であるかの確認を行うために、ユーザタスク1つとサブプロセス2つを設置します。

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　受付 と設定します。



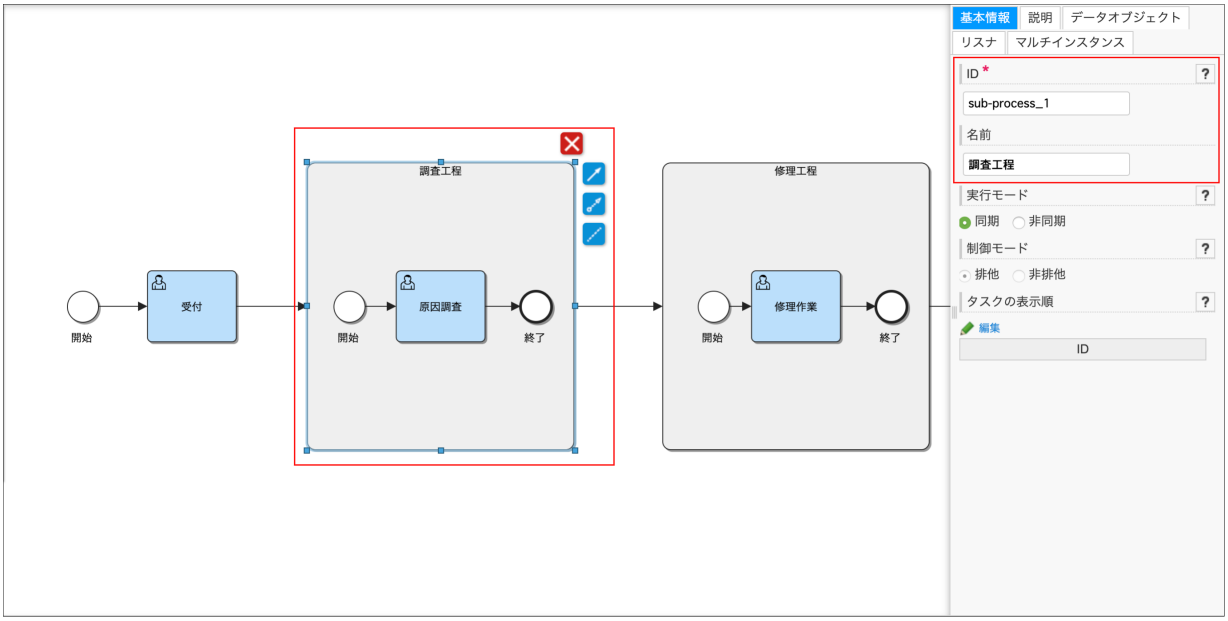
図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

5. 「受付」タスクに対して、担当者を設定します。
「受付」タスクを選択し「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」

6. 「タスク追加」画面でわかりやすくするために、配置した1つ目のサブプロセスに名前を付けます。
「サブプロセス」を選択し、「基本情報」タブから「名前」に　調査工程 と設定します。



図：「サブプロセス」 - 「プロパティ」 - 「基本情報」

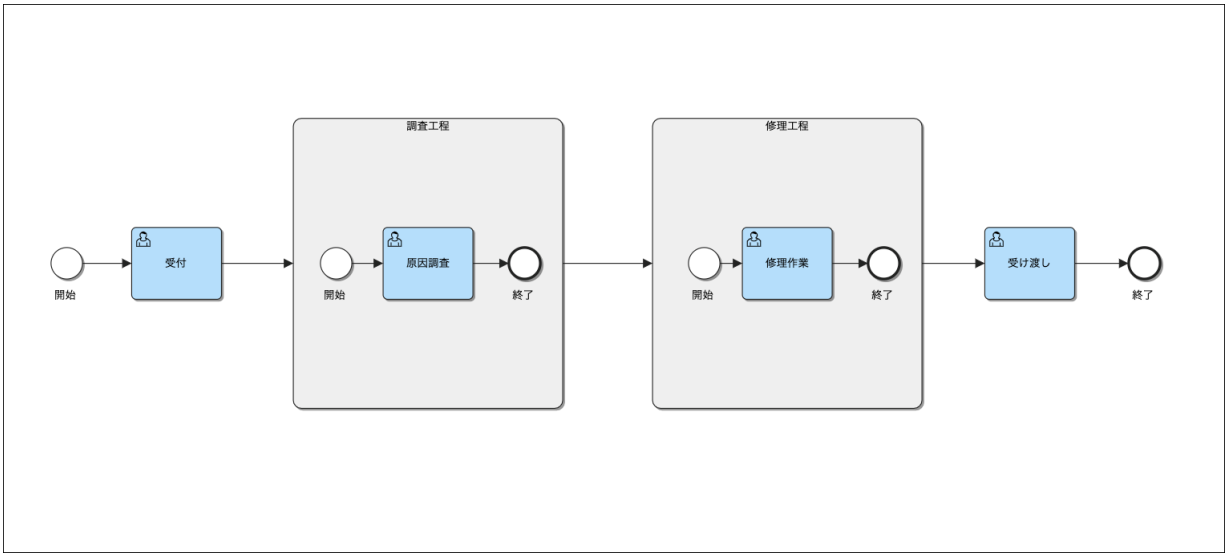
7. 「調査工程」サブプロセス内に以下のエレメントを設置します。
   - 「開始イベント」
   - 「ユーザタスク」
   - 「終了イベント」

8. 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、前述の「受付」タスクと同様の手順で、「調査工程」サブプロセス内のユーザタスクに対し名前と担当者を設定します。
   - 「ユーザタスク」を選択します。
   - 「基本情報」タブから「名前」に　原因調査 と設定します。
   - 「メインコンフィグ」タブから「担当者」に aoyagi と設定します。

9. 「タスク追加」画面でわかりやすくするために、前述の「調査工程」サブプロセスと同様の手順で、配置した2つ目のサブプロセスに名前を付けます。
   - 「サブプロセス」を選択します。
   - 「基本情報」タブから「名前」に　修理工程 と設定します。

10. 「修理工程」サブプロセス内に以下のエレメントを設置します。
    - 「開始イベント」
    - 「ユーザタスク」
    - 「終了イベント」

11. 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、前述の「受付」タスクと同様の手順で、「修理工程」サブプロセス内のユーザタスクに名前を付け、担当者を設定します。
    - 「ユーザタスク」を選択します。
    - 「基本情報」タブから「名前」に　修理作業 と設定します。
    - 「メインコンフィグ」タブから「担当者」に aoyagi と設定します。
12. プロセスの直下に「ユーザタスク」を設置します。
13. 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、前述の「受付」タスクと同様の手順で、ユーザタスクに名前を付け、担当者を設定します。
    - 「ユーザタスク」を選択します。
    - 「基本情報」タブから「名前」に　受け渡し と設定します。
    - 「メインコンフィグ」タブから「担当者」に aoyagi と設定します。
14. 以下に示すプロセス定義図を参考にエレメント同士をシーケンスフローでつなぎます。



図：プロセス定義図

15. プロセスに名前を付け保存します。

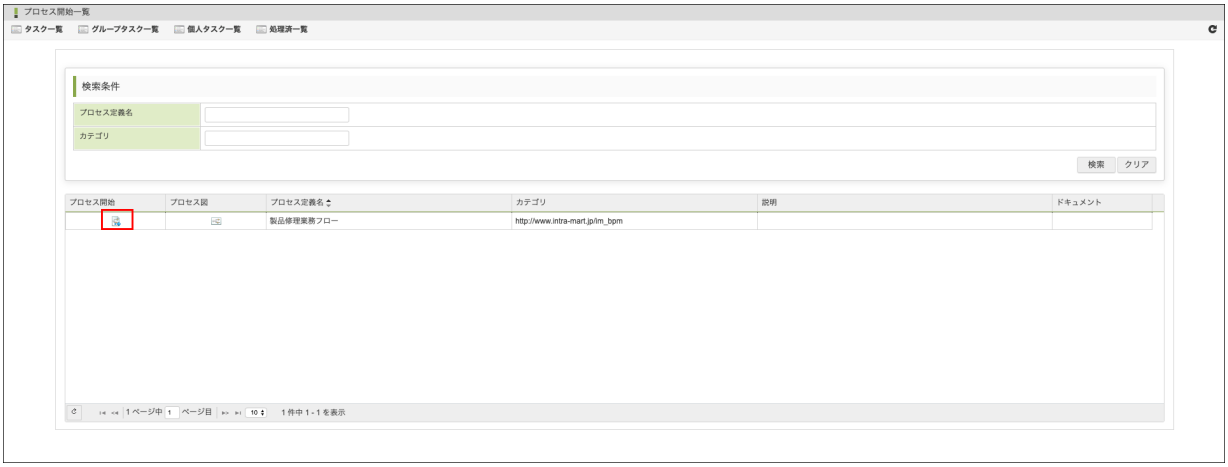
アドホックタスクを追加する

作成した「プロセス定義」を実行環境にデプロイし、アドホックタスクの追加を行います。

1. プロセスを開始します。

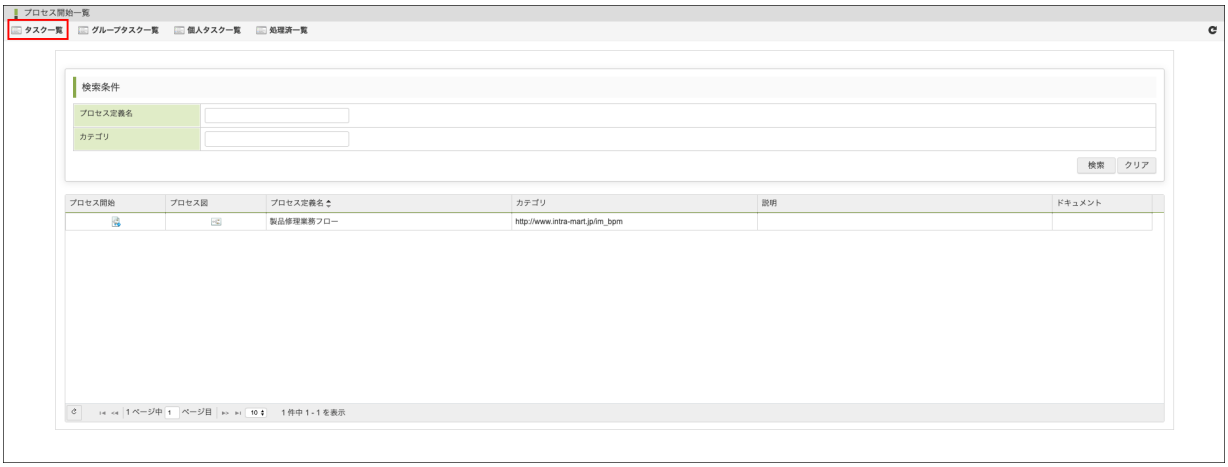
   「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示します。

2. 「プロセス開始一覧」画面から、対象となるプロセスの「 」をクリックします。



図：「プロセス開始一覧」

3. 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。

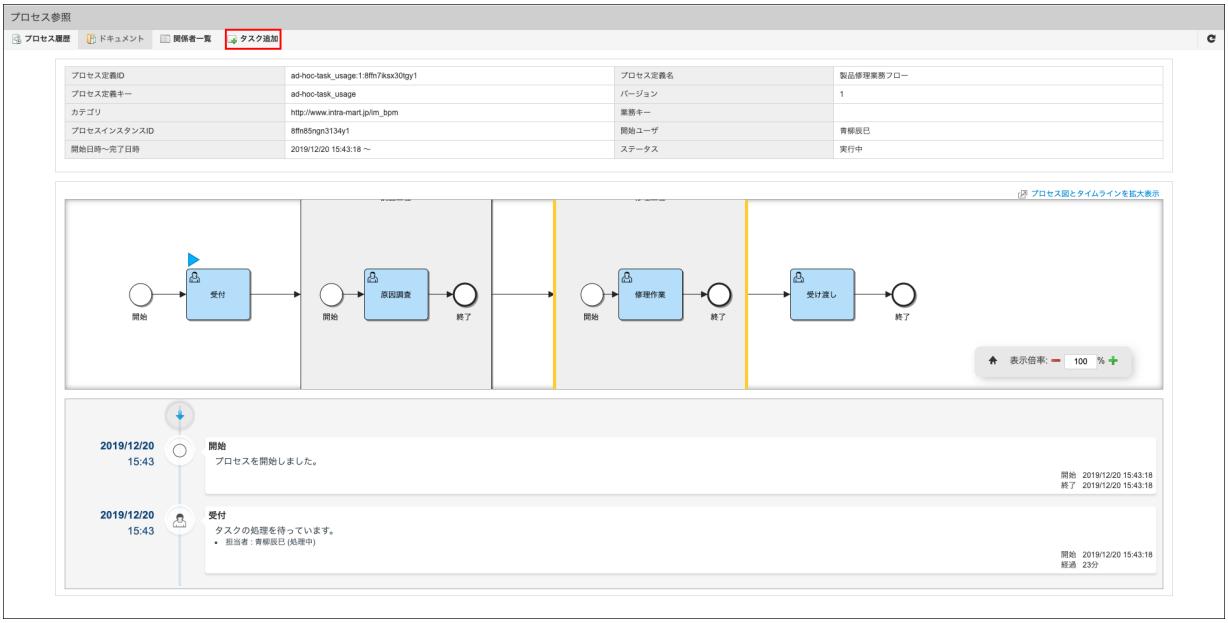


図：「プロセス開始一覧」

4. 対象のプロセス定義から、「受付」タスクが表示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」

5. 表示される「プロセス参照」画面のツールバーから「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」

6. 「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。

- プロセス直下の「その他のタスクを追加する」が表示されていて、選択可能なこと。

- サブプロセス「調査工程」にはまだ達していないため、「調査工程」直下の「追加可能なタスク」一覧内の「その他のタスクを追加する」はグレーアウトされていて選択不可能なこと。
- サブプロセス「修理工程」にもまだ達していないため、「修理工程」直下の「追加可能なタスク」一覧内の「その他のタスクを追加する」はグレーアウトされていて選択不可能なこと。

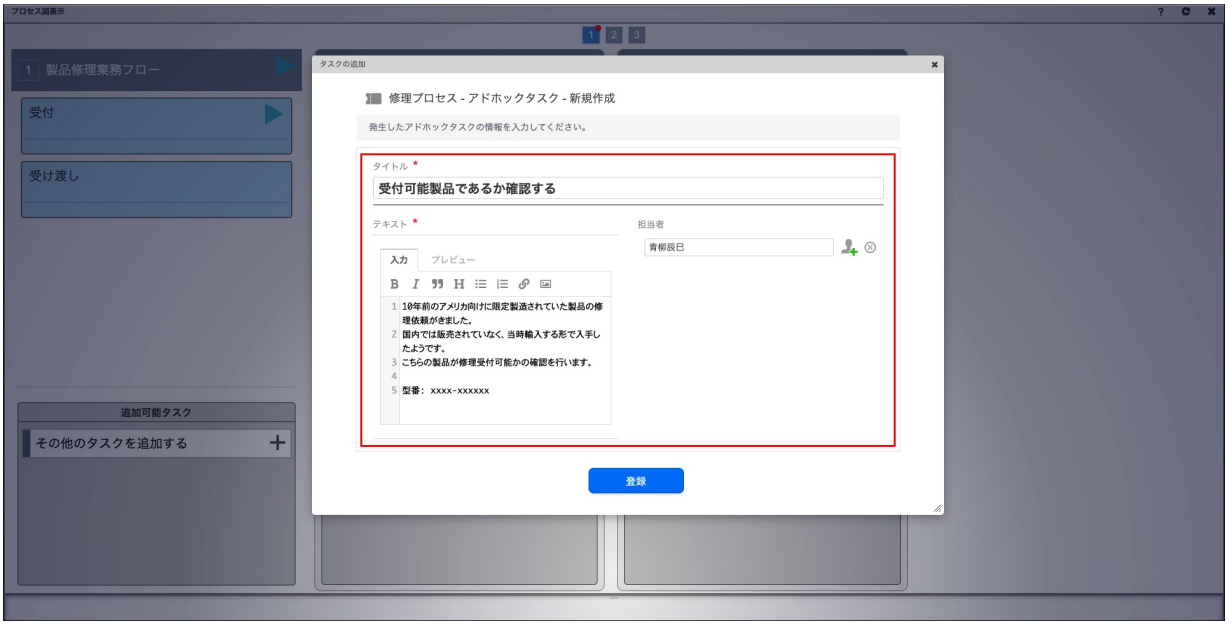
図：「タスク追加」

7. プロセス直下にアドホックタスクを追加します。
   - プロセス直下の「追加可能なタスク」一覧から「その他のタスクを追加する」を選択します。
8. プロセス定義作成時に設定したチケットの画面が表示されます。
9. チケットの項目のフォームに以下のように項目を設定します。
   - タイトル: 受付可能製品であるか確認する
   - テキスト:

     10年前のアメリカ向けに限定製造されていた製品の修理依頼がきました。
     国内では販売されていなく、当時輸入する形で入手したようです。
     この製品が修理受付可能かの確認を行います。

     型番: xxxx-xxxxxx

   - 担当者: 青柳辰巳



図：「タスクの追加」

10. 「登録」ボタンをクリックします。
11. アドホックタスクが追加されたことを確認します。

タスクのタイトルが、チケット作成時に入力した 受付可能製品であるか確認する になっていることを確認します。

図：「タスク追加」

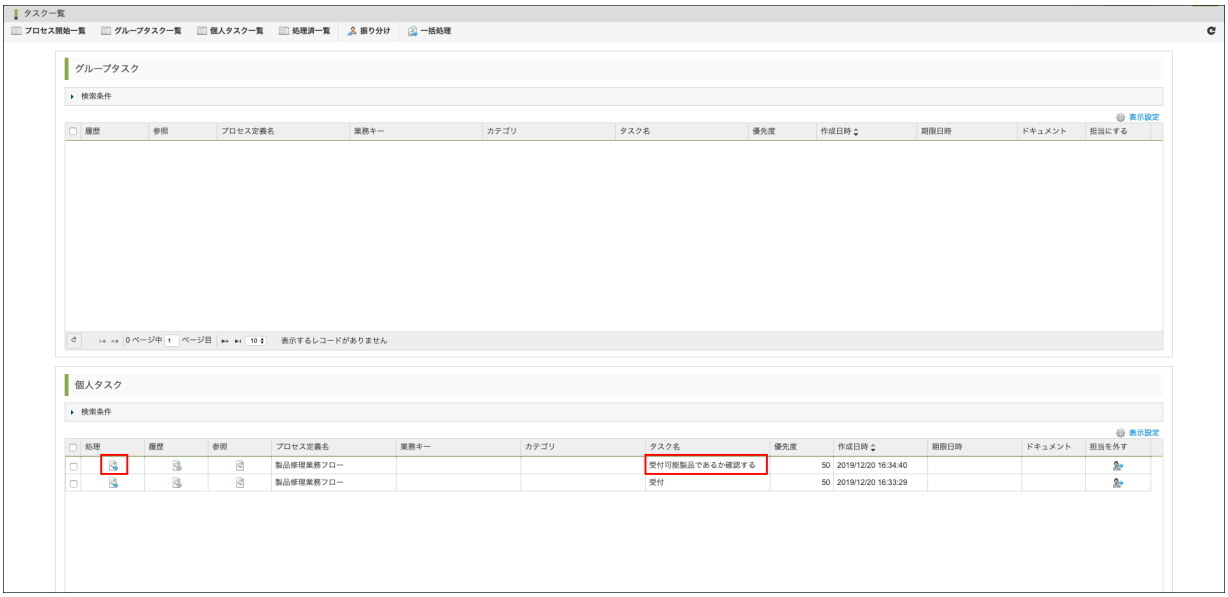
このように突発的に発生したタスクについても、プロセス内に追加することが可能です。

アドホックタスクのチケットを更新する

追加されたアドホックタスクのチケットの内容を更新して最新の状態を記録します。

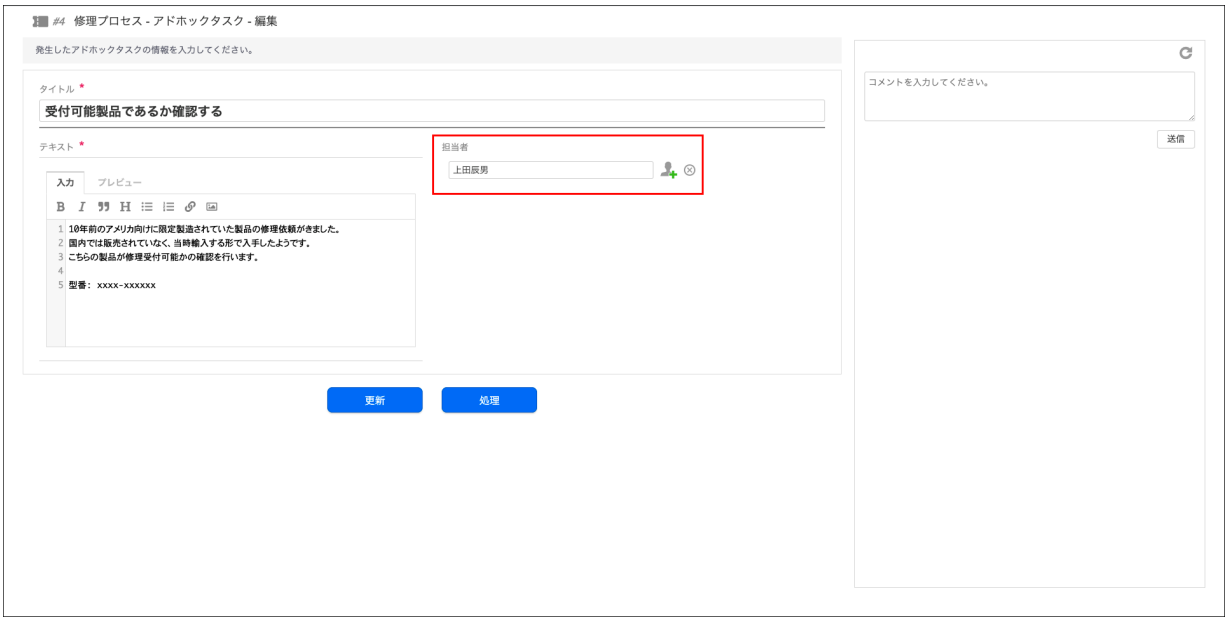
1. 「タスク一覧」をクリックし、「タスク一覧」画面を表示します。
2. 個人タスクの中に、追加した「受付可能製品であるか確認する」タスクが存在することを確認します。

   「 」をクリックし、処理画面を表示します。



図：「タスク一覧」

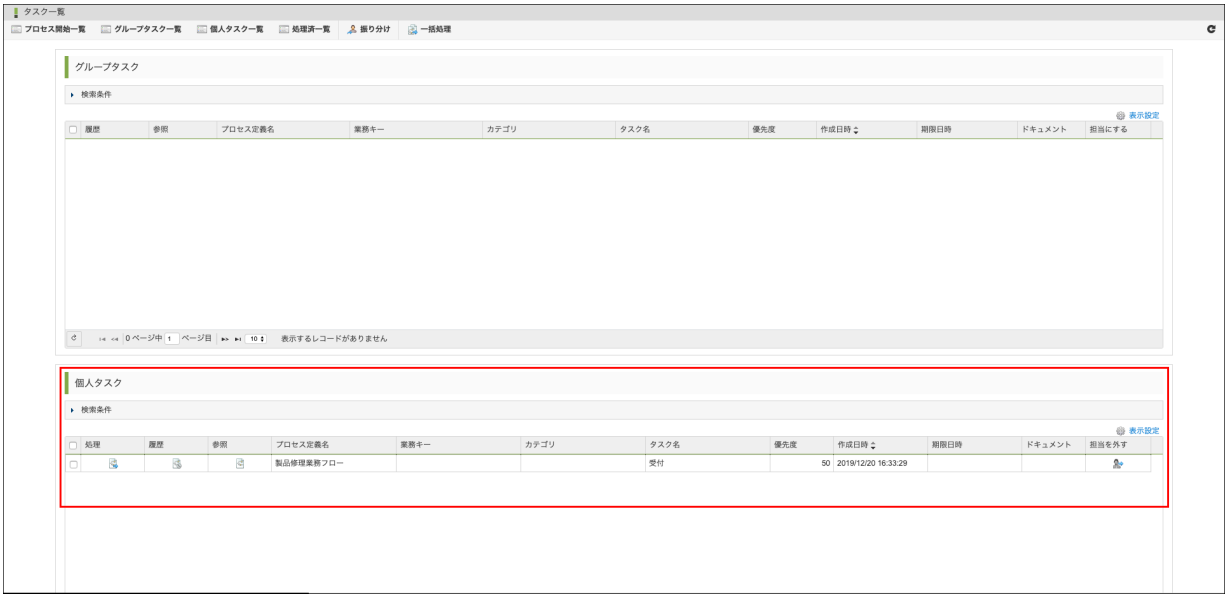
3. タスク追加時に入力したチケットの画面が表示されることを確認します。
4. 「担当者」を 青柳辰巳 から 上田辰男 に変更し、「更新」をクリックします。



図：「アドホックタスクチケット」

5. 「タスク一覧」画面が表示されることを確認します。

   この時、担当者を変更したため、 aoyagi ユーザの個人タスクには、該当のタスクが表示されません。



図：「タスク一覧」

> **コラム**
>
> 「担当者」以外にも、チケットマスタで以下の項目を設定できるようにした場合は、アドホックタスクの各項目と連動します。
>
> - タイトル
> - 優先度
> - 期限

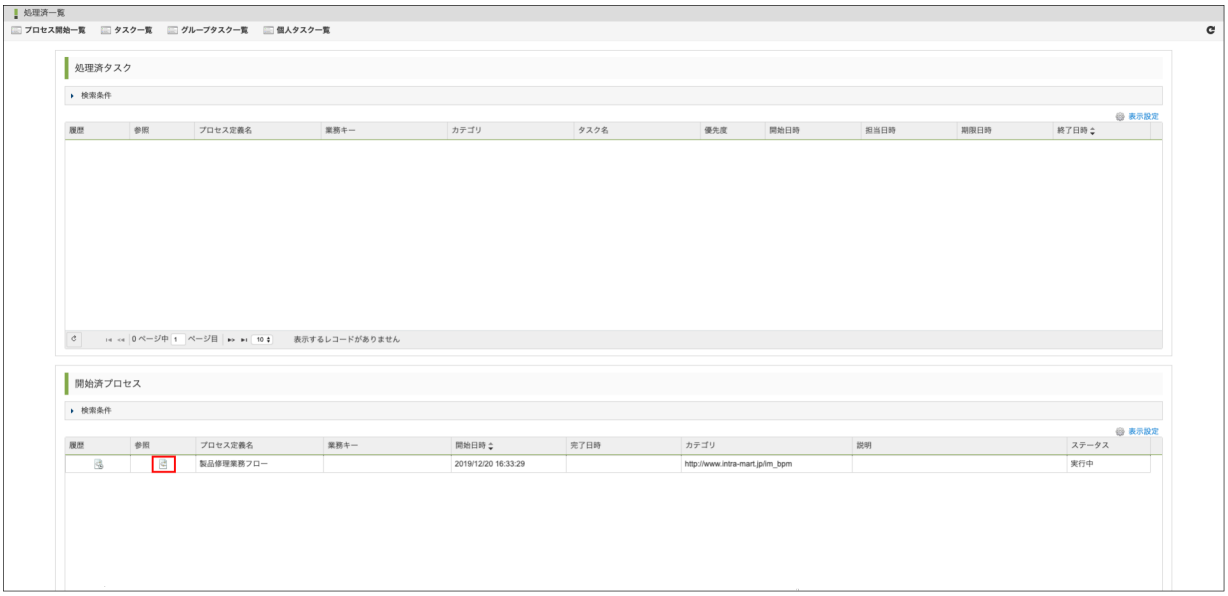
チケットの履歴の確認/コメントをする

チケットの更新履歴を確認し、コメントをします。
アドホックタスクのチケットには、以下の機能があります。

- チケットの更新履歴の確認をする。
- 関係者がタスクに対してコメントを残す。

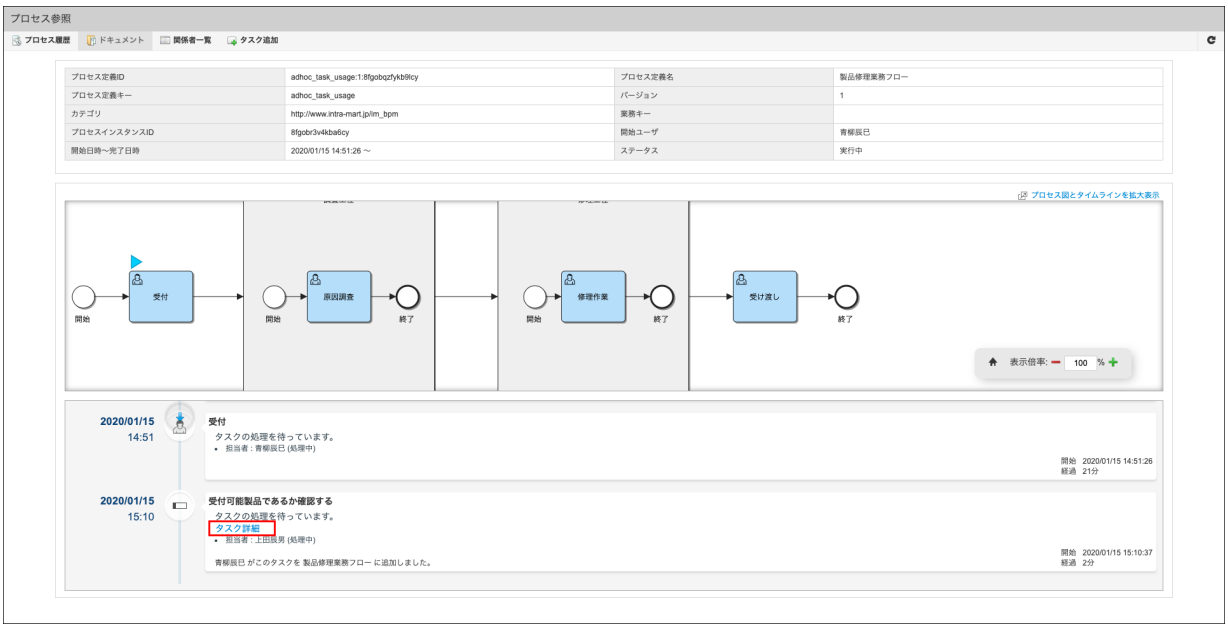
1. 該当プロセスを開始した aoyagi ユーザで「タスク一覧」画面から、「処理済一覧」画面を表示します。

2. 開始済プロセス内の該当プロセス定義の「 」をクリックします。



図：「処理済一覧」

3. プロセス履歴から追加したアドホックタスクの「タスク詳細」をクリックします。

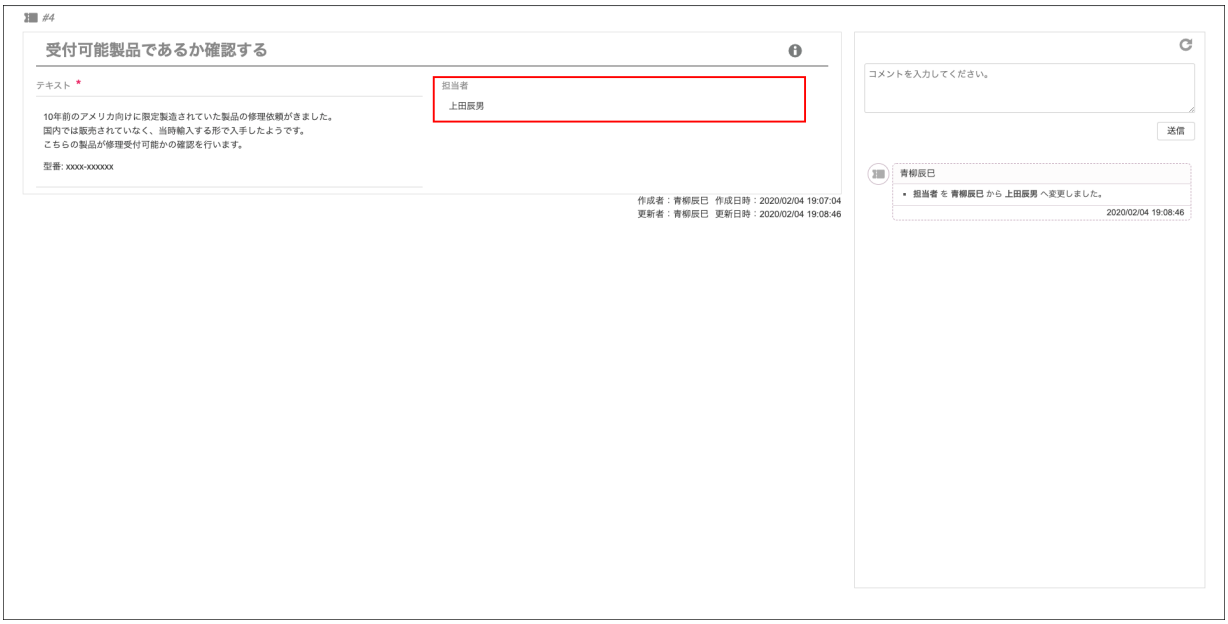


図：「プロセス参照」

4. チケット画面が表示され、右側の部分でチケットの更新履歴を確認します。

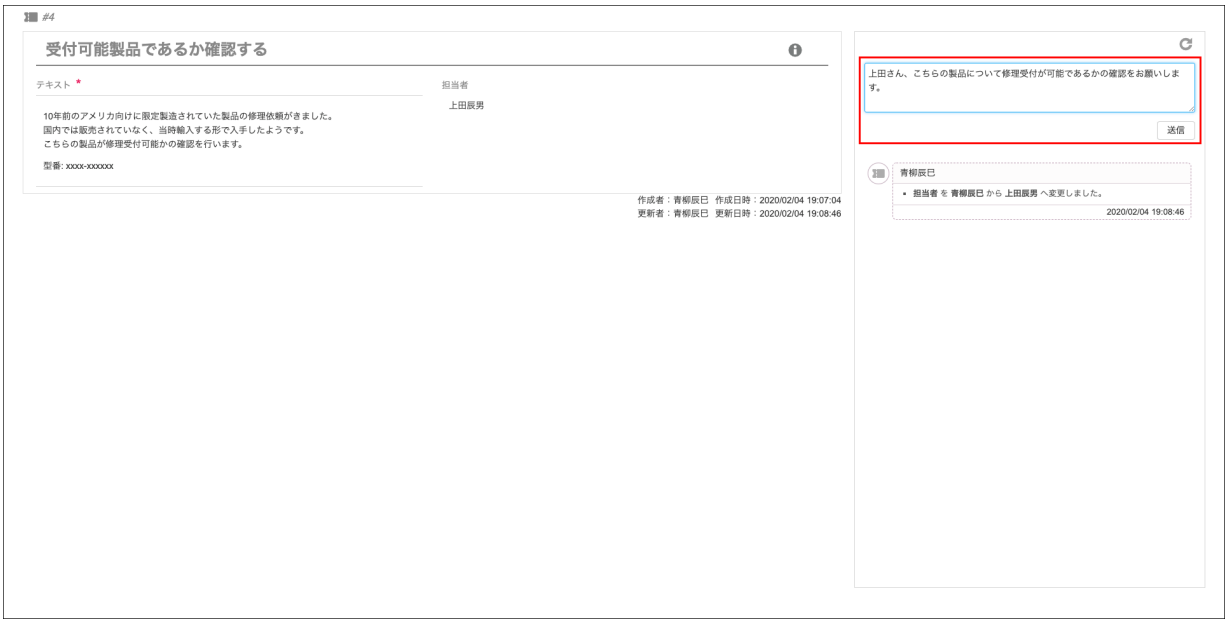
   今回の場合は、担当者が `青柳辰巳` から `上田辰男` に変更されたことが確認できます。

> **コラム**
> 表示している画面の大きさによっては更新履歴は画面下部に表示されます。



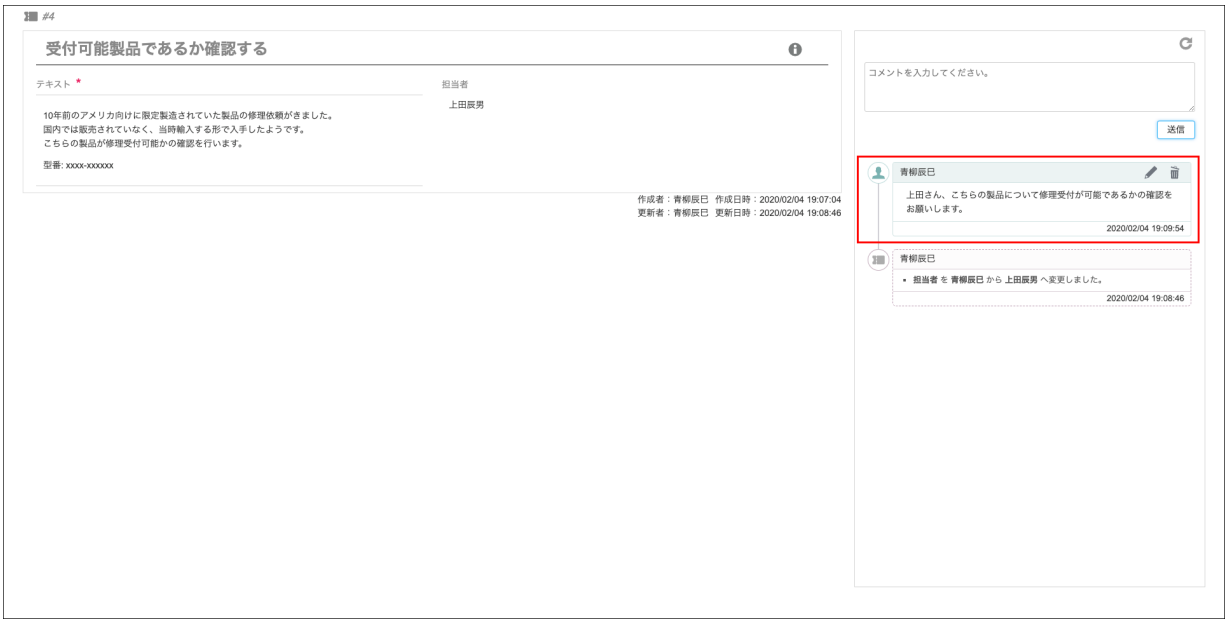
図：「アドホックタスクチケット」

5. 履歴上部のコメント入力欄に以下のように入力し、「送信」をクリックします。
   - 上田さん、この製品について修理受付が可能であるかの確認をお願いします。



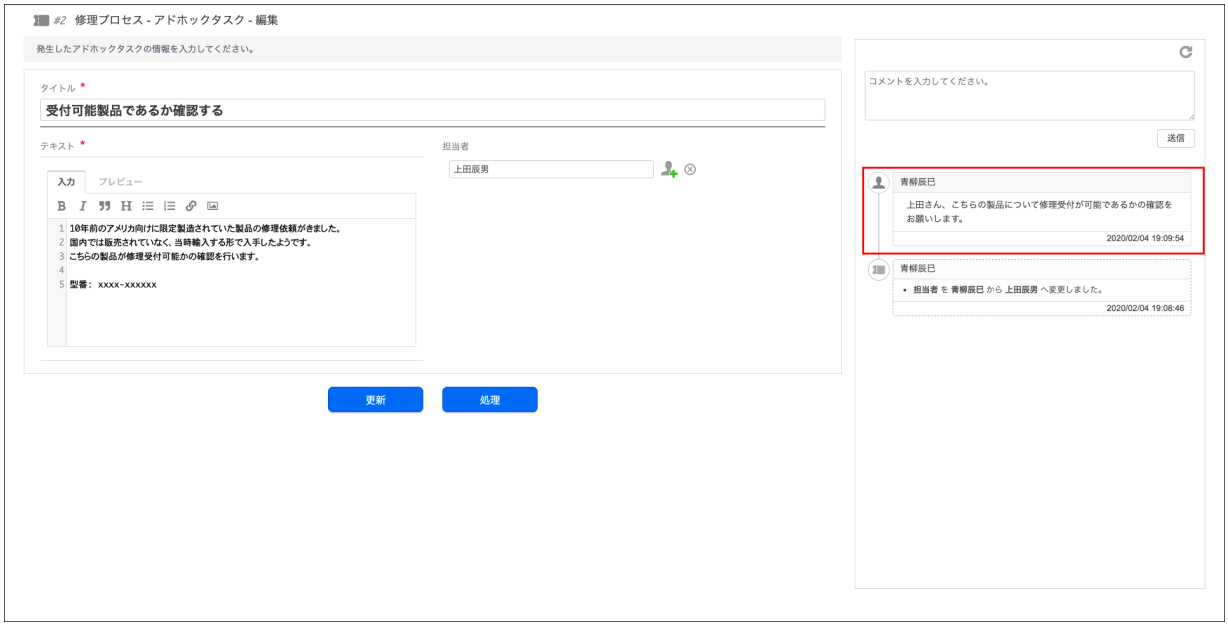
図：「アドホックタスクチケット」

6. 入力したコメントが、入力欄下部の履歴部分に表示されることを確認します。



図：「アドホックタスクチケット」

7. 他のユーザで、入力されたコメントの確認を行います。
   - 現在のタスク担当者である ueda ユーザでログインします。
   - 「タスク一覧」画面から、該当タスクの「 」をクリックします。
   - 「チケット」画面が表示され、aoyagi ユーザで入力したコメントを確認できます。



図：「アドホックタスクチケット」

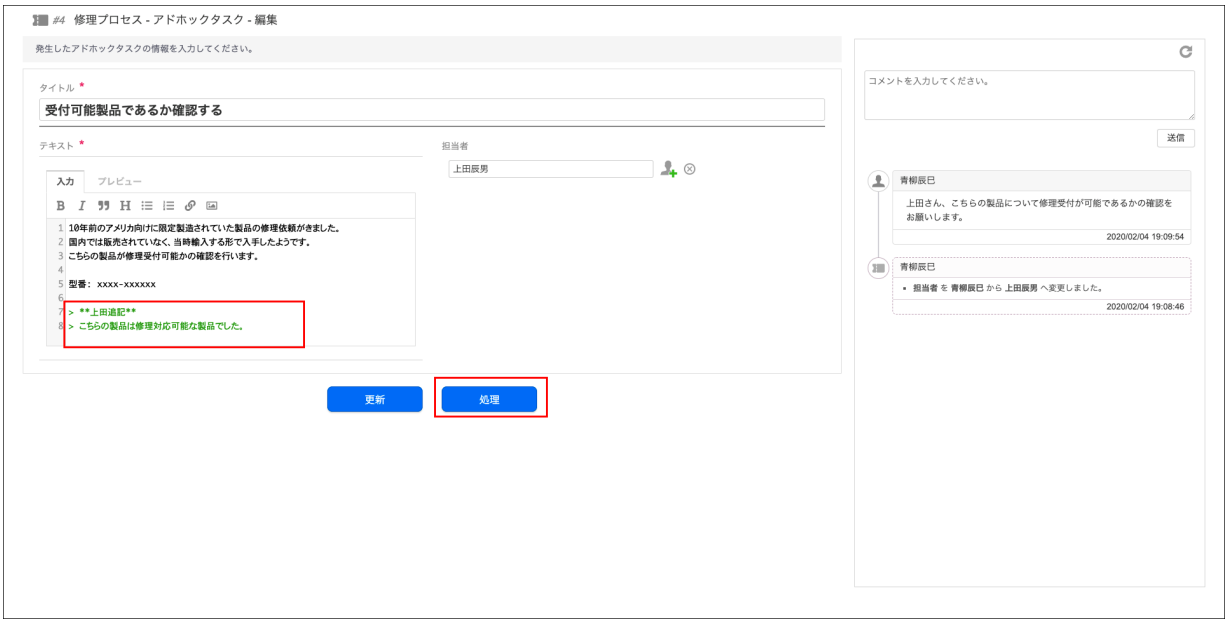
このように、突発的に発生したタスクに対して、関係者同士でコミュニケーションをとりながら進めることが可能です。

アドホックタスクを処理する

発生したアドホックタスクを処理します。

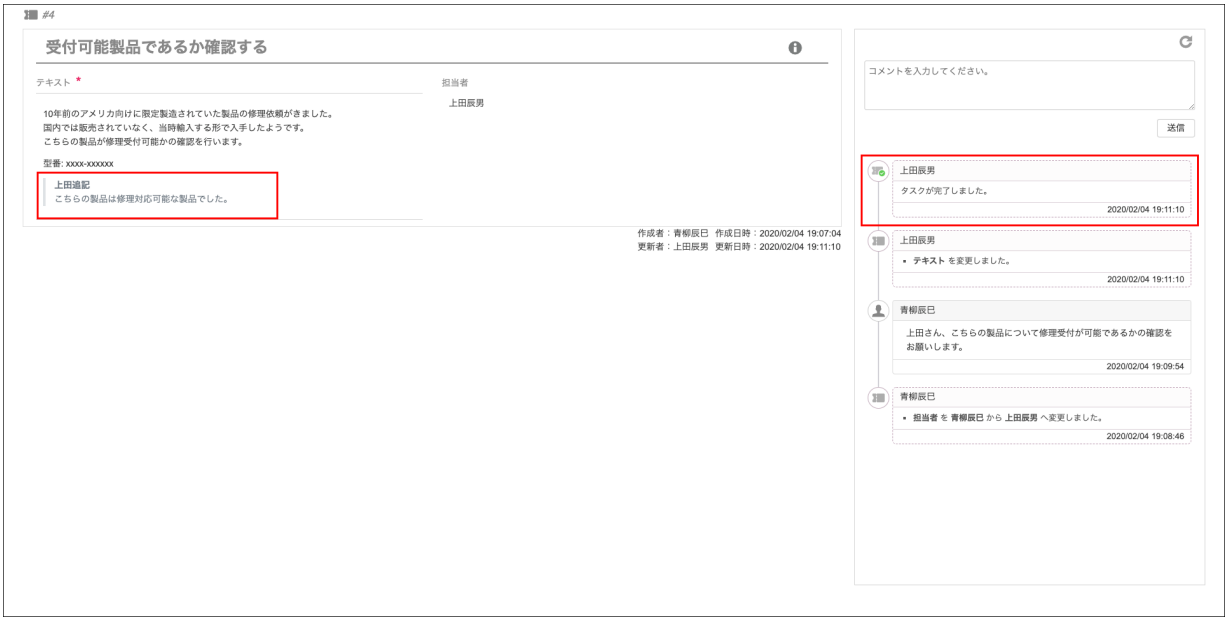
1. 現在アドホックタスクの担当者になっている ueda ユーザで「タスク一覧」画面を表示します。
2. 該当タスクの「 」をクリックし、処理画面を表示します。
3. チケット「テキスト」項目に以下の内容を追記し、「処理」ボタンをクリックします。

```
> **上田追記**
> この製品は修理対応可能な製品でした。
```



図：「アドホックタスクチケット」

4. 「タスク一覧」画面に戻り、処理したタスクは一覧内に表示されていないことが確認できます。
5. 「処理済一覧」をクリックします。
6. 「処理済タスク」の一覧内の該当タスクの「 」をクリックします。
7. プロセス履歴から該当タスクの「タスク詳細」をクリックします。
8. 以下の2点について、チケットが最新の情報となっていることが確認できます。
   - 表示されたチケットの画面が、処理のタイミングで更新した内容となっていることが確認できます。
   - 右側の履歴でタスクが完了していることが確認できます。

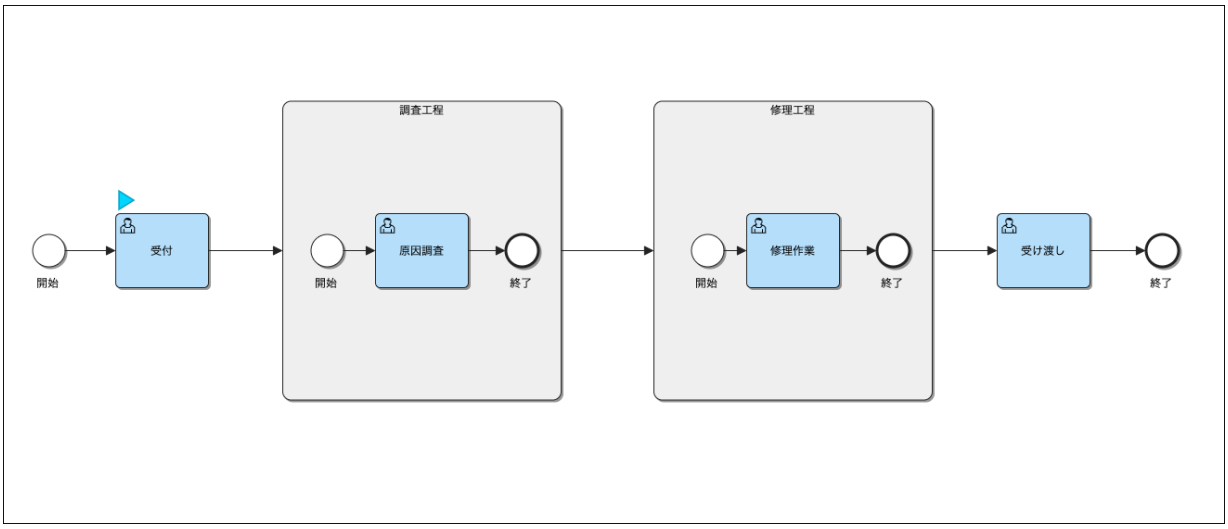


図：「アドホックタスクチケット」

アドホックタスクが追加可能なタイミングを確認する

アドホックタスクは、現在実行中のコンテナの位置に追加が可能です。
ここでは、アドホックタスクが追加可能なタイミングについて確認していきます。

1. 現在以下の状態です。

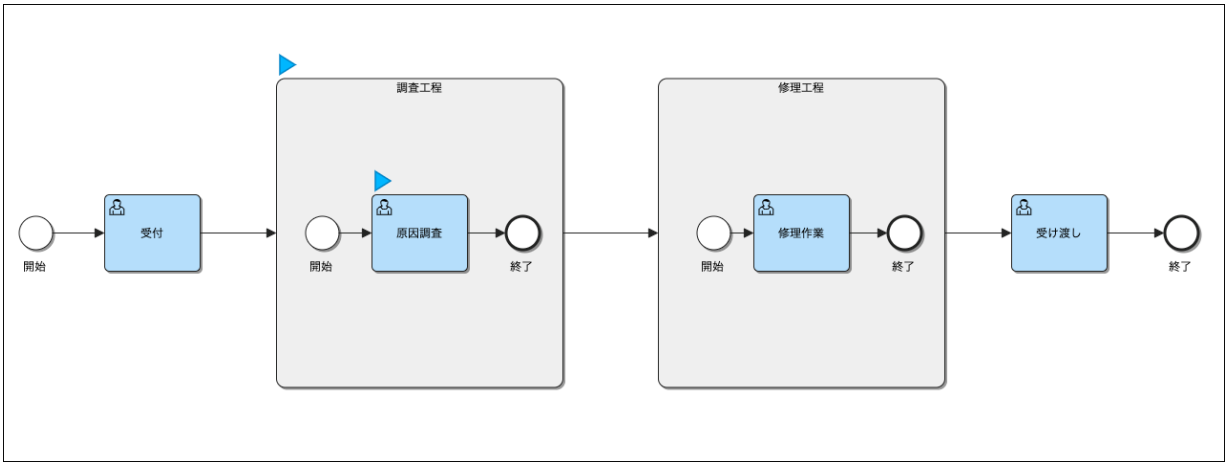


図：「現在実行状態」

2. この場合は、現在実行中のコンテナはルートプロセスのみです。そのため、アドホックタスクを追加できるのは、プロセス直下のみです。

図：「タスク追加」

3. aoyagi ユーザで「受付」タスクを完了させ「調査工程」サブプロセス内の「原因調査」タスクへ移動します。



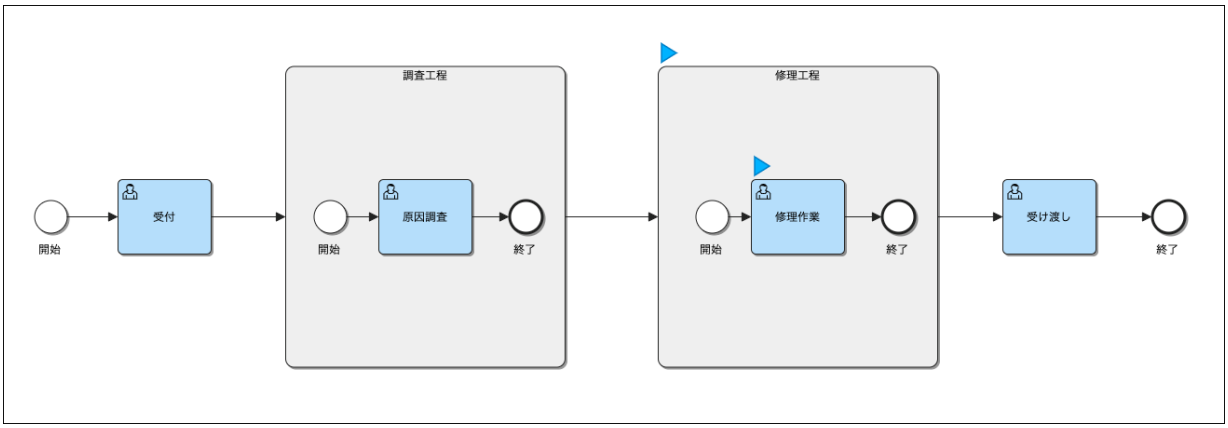
図：「現在実行状況」

4. この場合は、現在実行中のコンテナはルートプロセスと「調査工程」サブプロセスです。そのため、アドホックタスクを追加できるのは、プロセス直下と「調査工程」サブプロセスです。
   - 「調査工程」サブプロセスへアドホックタスクを追加した場合は、そのアドホックタスクが完了するまで「調査工程」サブプロセスは完了しません。
   - プロセス直下とへ追加した場合は、そのアドホックタスクが完了するまでプロセス全体は完了とはなりません。しかし「調査工程」サブプロセス

にあるタスクが全て完了した時点で、「調査工程」サブプロセスは完了となり「修理工程」サブプロセスへ遷移します。

図：「タスク追加」

5. 「aoyagi」ユーザで「原因調査」タスクを完了させ「修理工程」サブプロセス内の「修理作業」タスクへ移動します。



図：「現在実行状況」

6. この場合は、現在実行中のコンテナはルートプロセスと「修理工程」サブプロセスです。そのため、アドホックタスクを追加できるのは、プロセス直下と「修理工程」サブプロセスです。「調査工程」サブプロセスはすでに完了してしまっているので、アドホックタスクを追加することはできません。
   - 「修理工程」サブプロセスへアドホックタスクを追加した場合は、そのアドホックタスクが完了するまで「修理工程」サブプロセスは完了しません。
   - プロセス直下とへ追加した場合は、そのアドホックタスクが完了するまでプロセス全体は完了とはなりません。しかし「修理工程」サブプロセスにあるタスクが全て完了した時点で、「修理工程」サブプロセスは完了となり、プロセス直下「受け渡し」タスクへ遷移します。

図：「タスク追加」

プール、レーン

プールとレーンを利用して作業の関係者を明確にする

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

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

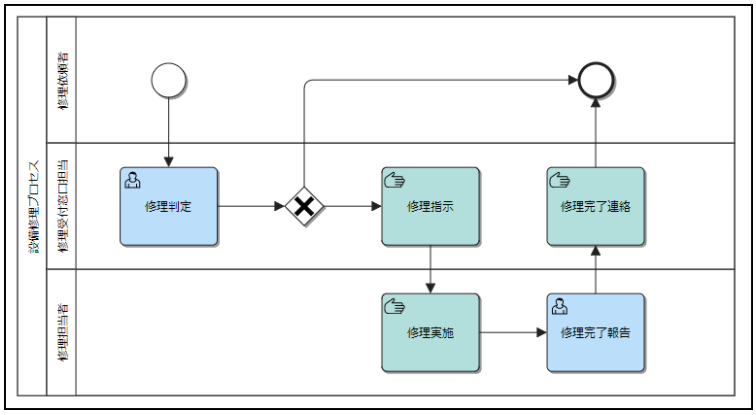
「プール」と「レーン」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「プール・レーン」もあわせて参照してください。

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

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

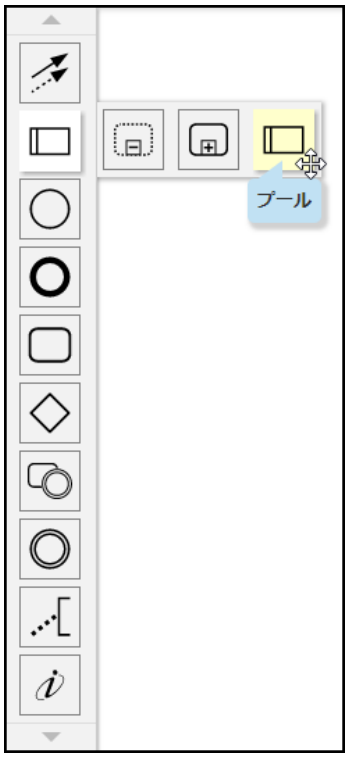
完成イメージ

このチュートリアルでは、以下のようなプロセスを作成します。



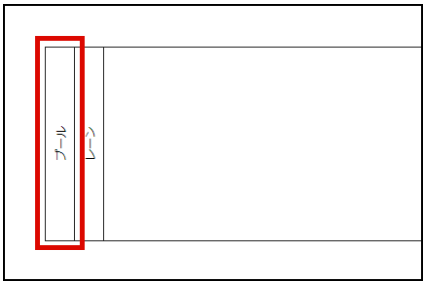
図：完成イメージ

プールを配置し、プロパティを設定する



図：パレット - コンテナ - プール

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



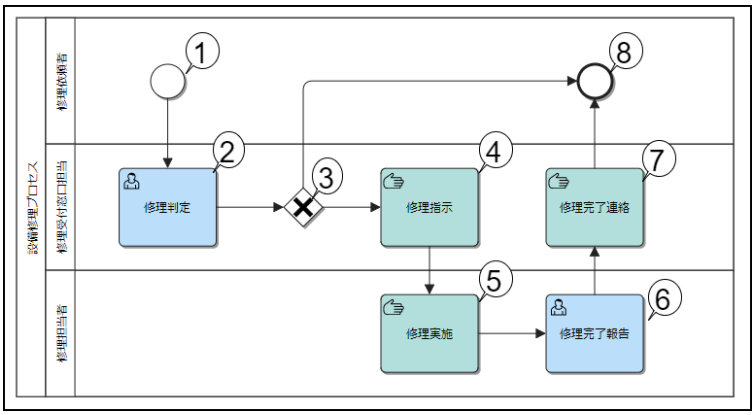
図：プールの選択

レーンを追加し、プロパティを設定する

1. プール、または、レーン選択時に表示される をクリックし、レーンを二つ追加します。
   - プールの**上部**にあるアイコンをクリックした場合、追加されるレーンはプール内の**最上部**に挿入されます。
   - プールの**下部**にあるアイコンをクリックした場合、追加されるレーンはプール内の**最下部**に挿入されます。
   - レーンの**上部**にあるアイコンをクリックした場合、追加されるレーンは**直上**に挿入されます。
   - レーンの**下部**にあるアイコンをクリックした場合、追加されるレーンは**直下**に挿入されます。
2. レーンのプロパティにて、「基本情報」タブの「名前」を上から順に以下のように設定します。
   - `修理依頼者`
   - `修理受付窓口担当`
   - `修理担当者`

エレメントを配置する

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



図：完成イメージ

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

補足

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

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 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。

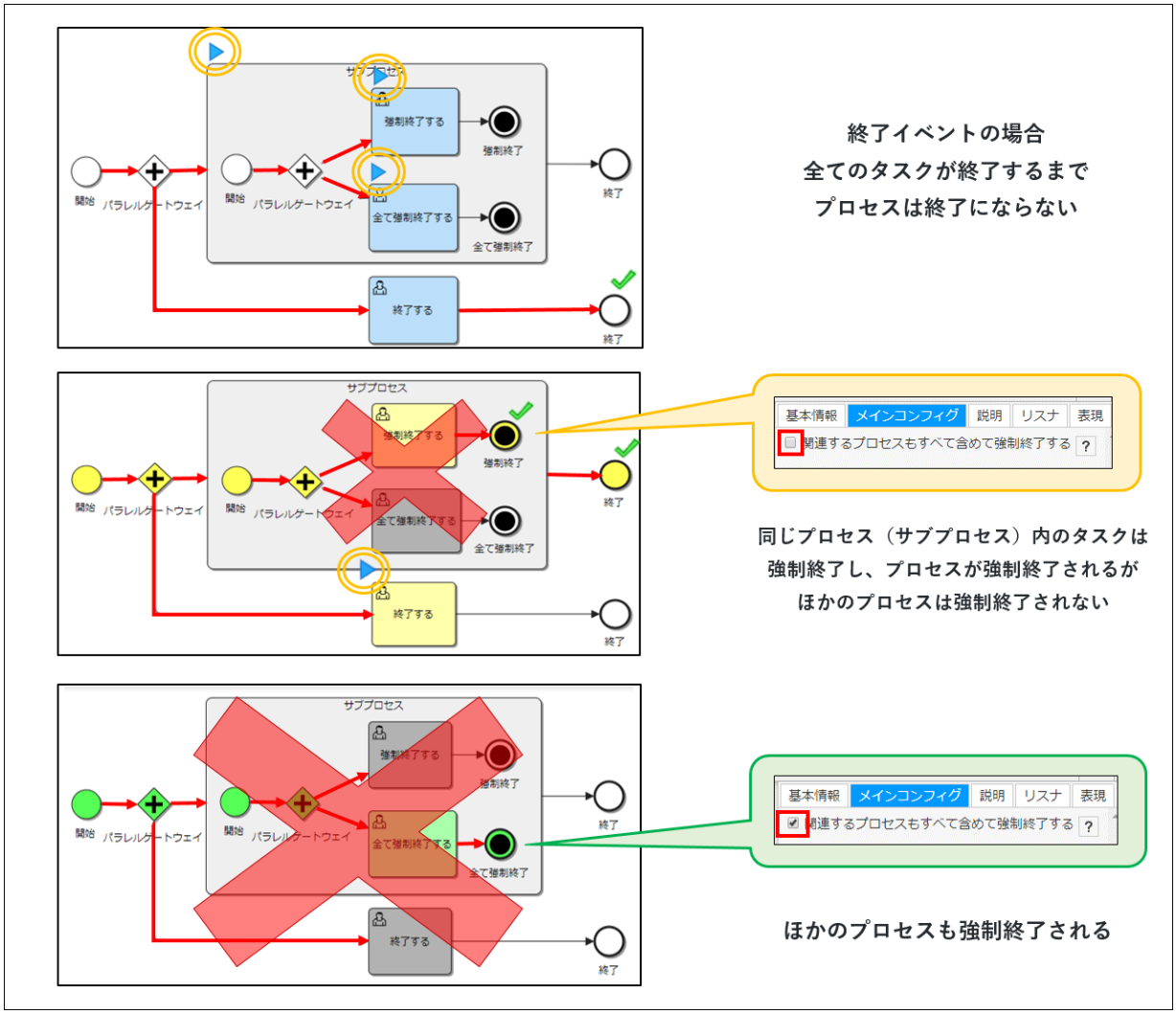


図 ： 概要図

> **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 操作ガイド」 - 「プロセス定義のアップロード」を参照してください。

> **コラム**
>
> このチュートリアルのサンプルでは、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。

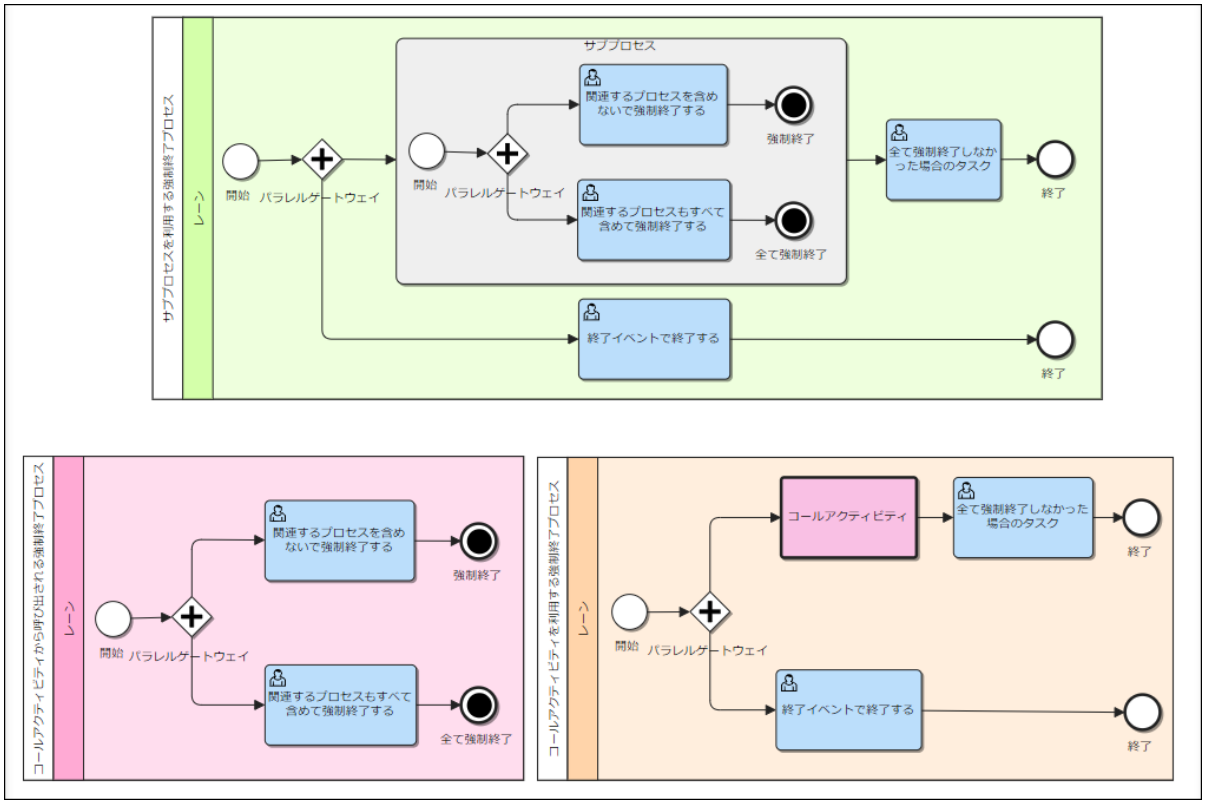
- プロセス定義を作成する
- 結果を確認する

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

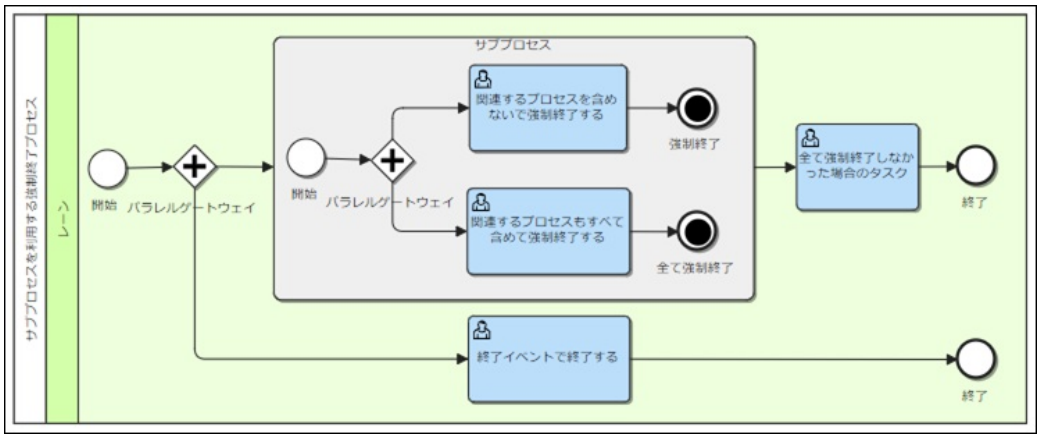
このチュートリアルでは、「サブプロセス」を利用するプロセスと、「コールアクティビティ」を利用するプロセスを作成します。
サブプロセス の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「コンテナ」 - 「サブプロセス」を参照してください。
コールアクティビティの詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「タスク」 - 「コールアクティビティ」を参照してください。



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」

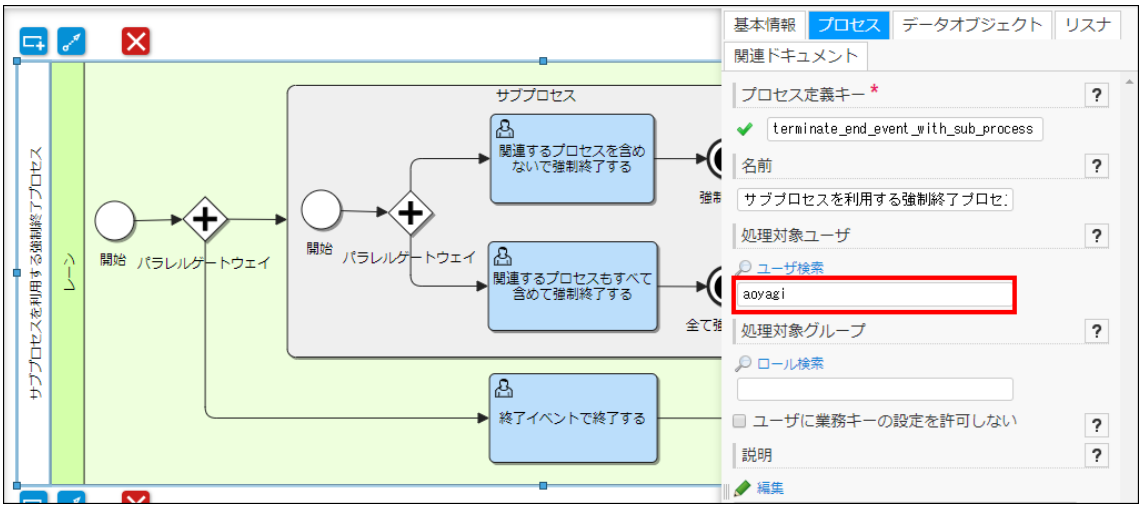
サブプロセスを利用して、強制終了するプロセス定義を作成する

「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。



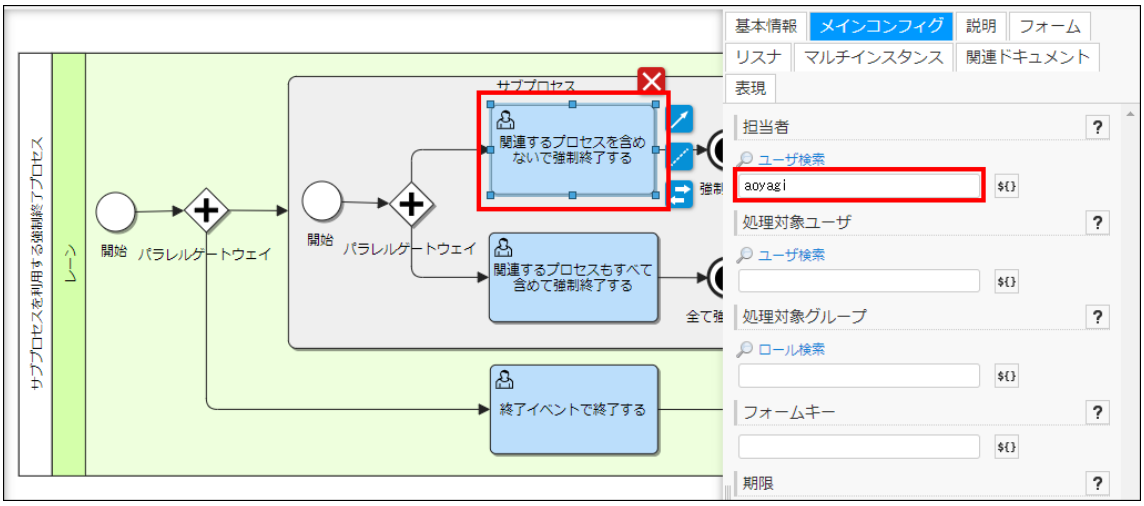
図：サブプロセスを利用する強制終了プロセス

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。



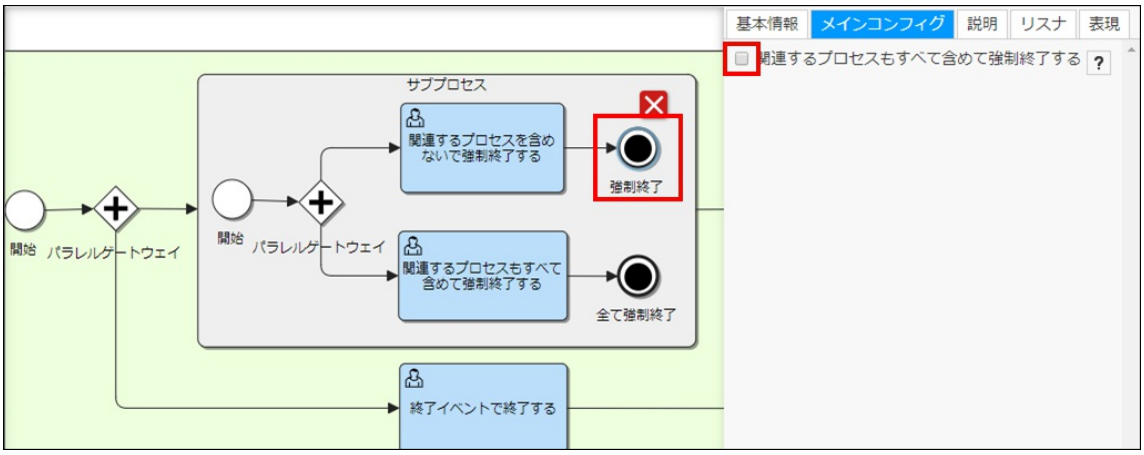
図：「プール」 - 「プロパティ」 - 「プロセス」 - 「処理対象ユーザ」

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」

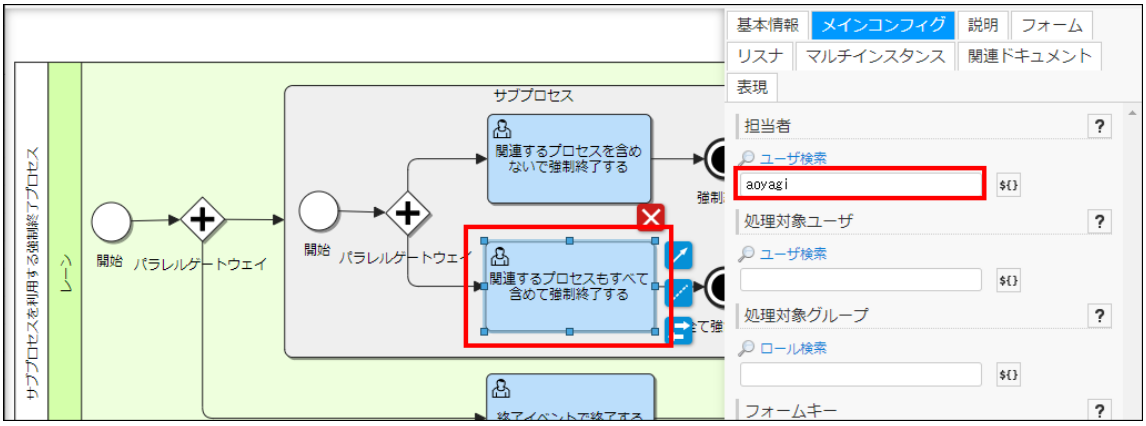
10. 「強制終了イベント」を設置します。
11. 「関連するプロセスもすべて含めて強制終了する」の設定を確認します。
    「メインコンフィグ」タブの「関連するプロセスもすべて含めて強制終了する」はデフォルトでチェックが入っていません。
    現在作成しているルートは設定不要です。



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。

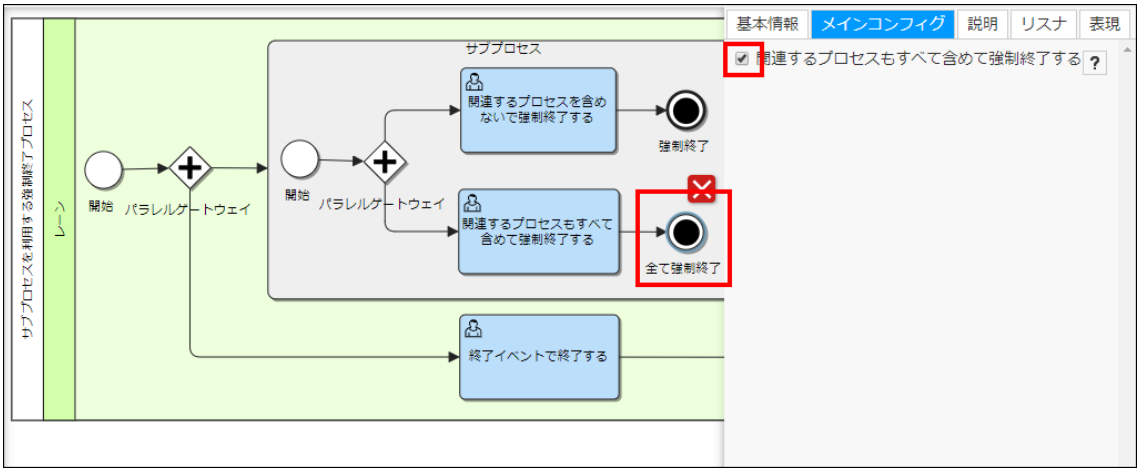
13. 「関連するプロセスもすべて含めて強制終了する」の担当者を設定します。
「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「 aoyagi 」と設定します。



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。

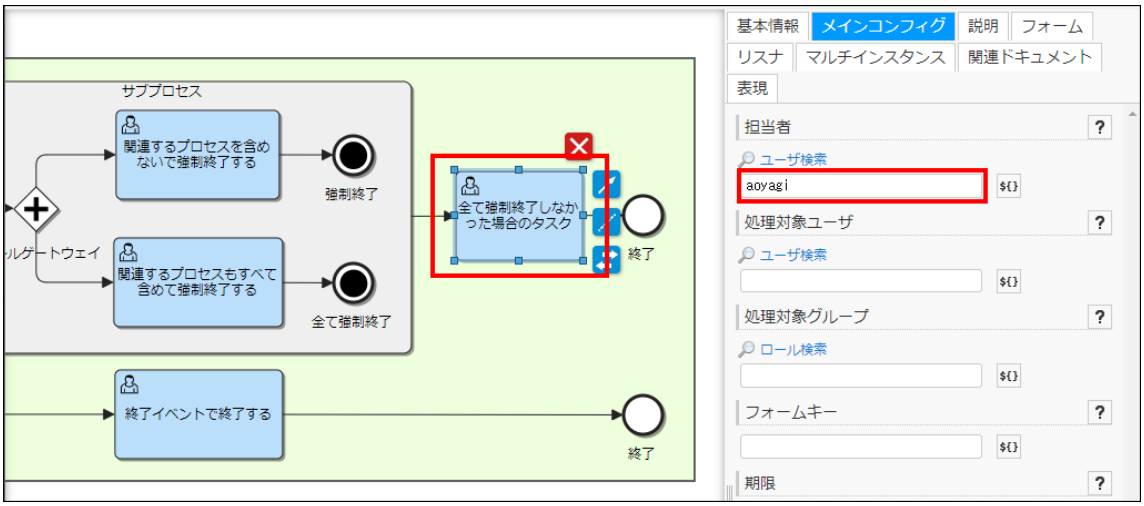
15. 「強制終了イベント」に到達した際に、関連するプロセスもすべて含めて強制終了させる設定を行います。
「強制終了イベント」を選択した状態で、「メインコンフィグ」タブにある「関連するプロセスもすべて含めて強制終了する」チェックボックスにチェックを入れます。



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。

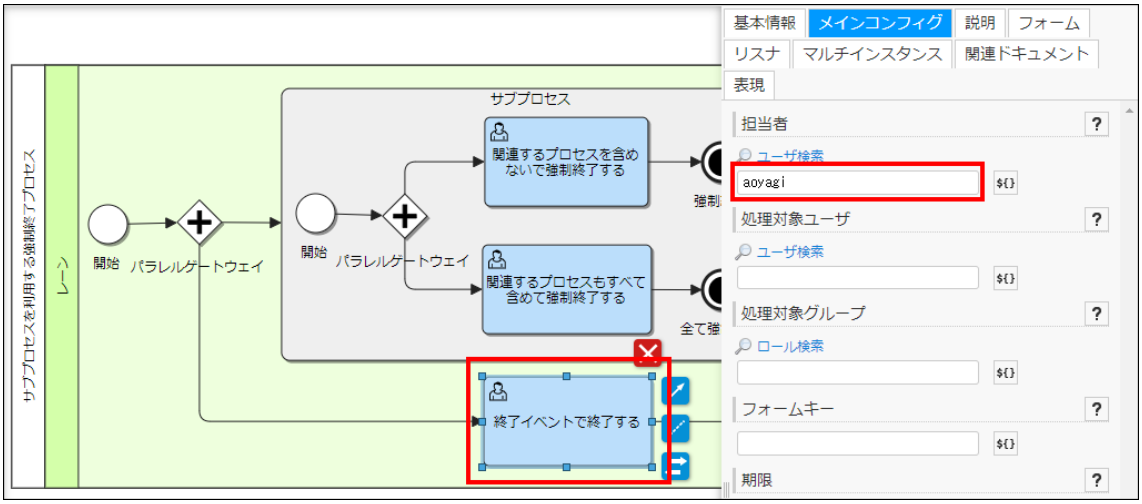


図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

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。

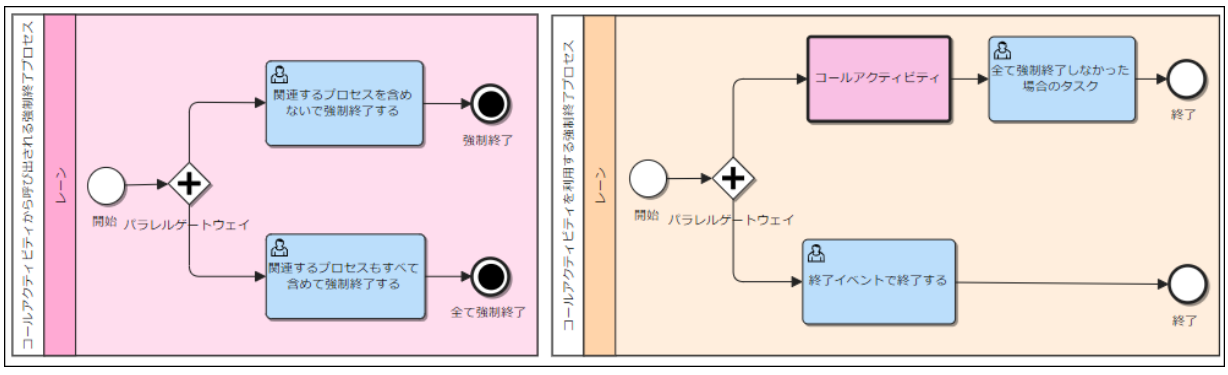


図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」 - 「担当者」

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。

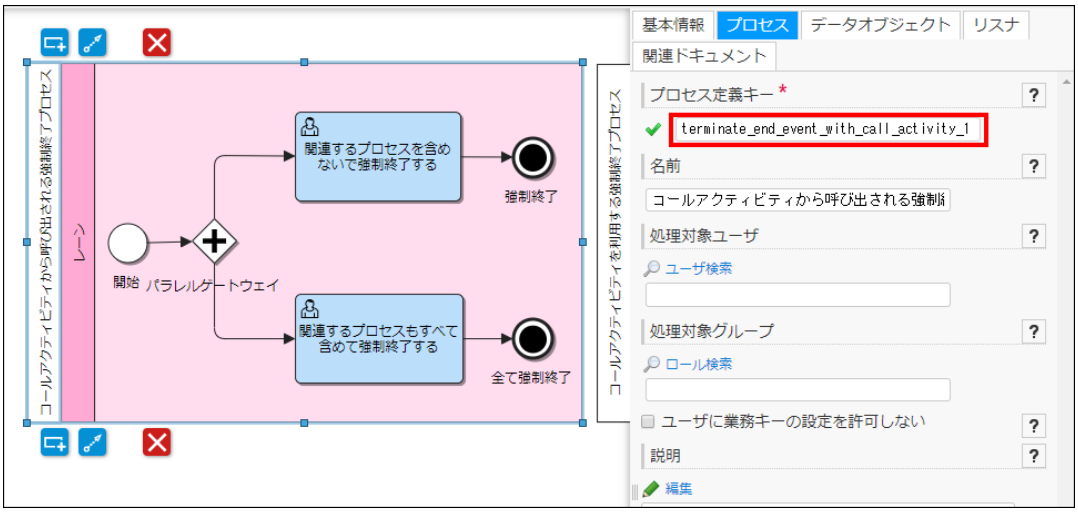


図：「コールアクティビティから呼び出される強制終了プロセス」と「コールアクティビティを利用する強制終了プロセス」

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。

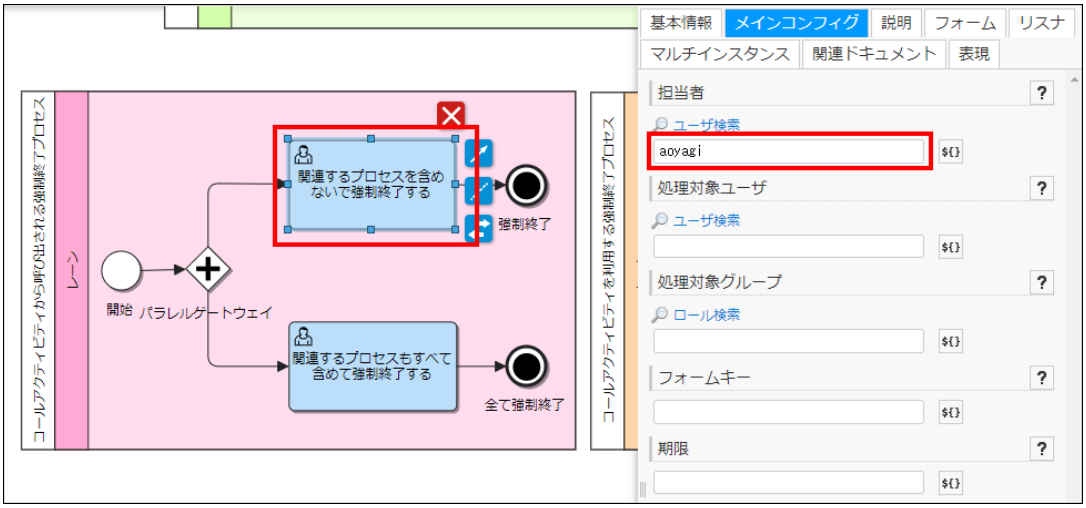


図：「プール」 - 「プロパティ」 - 「プロセス」 - 「プロセス定義キー」

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。

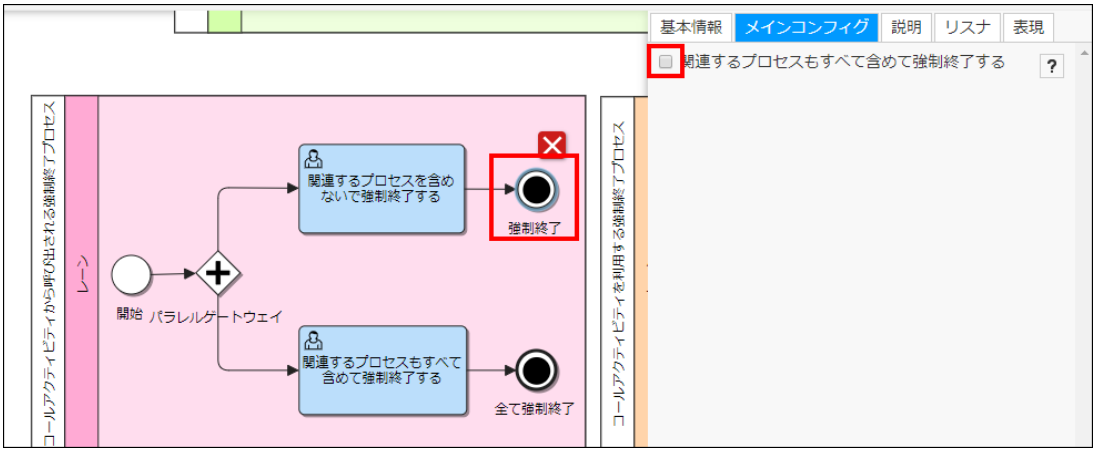
6. 「関連するプロセスを含めないで強制終了する」タスクの担当者を設定します。
   「ユーザタスク」を選択した状態で、「メインコンフィグ」タブから担当者に「aoyagi」と設定します。



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。

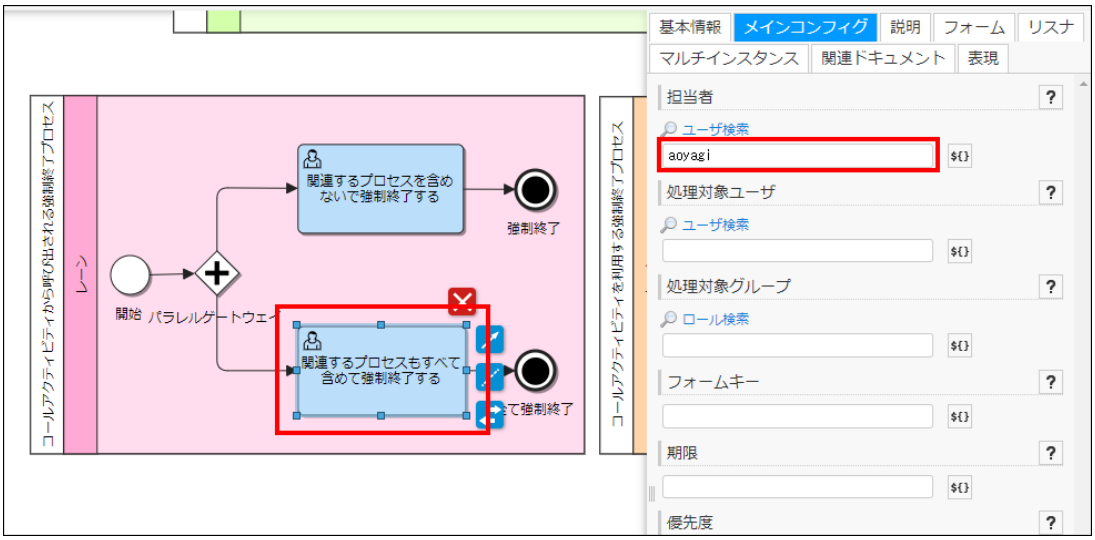


図：「強制終了イベント」 - 「プロパティ」 - 「メインコンフィグ」 - 「関連するプロセスもすべて含めて強制終了する」

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。

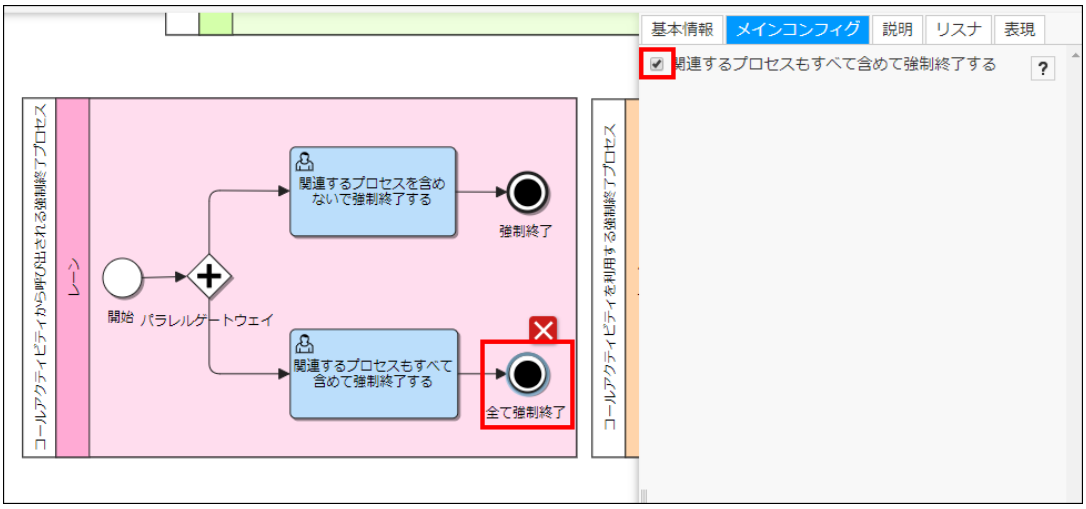
「ユーザタスク」を設置します。

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。

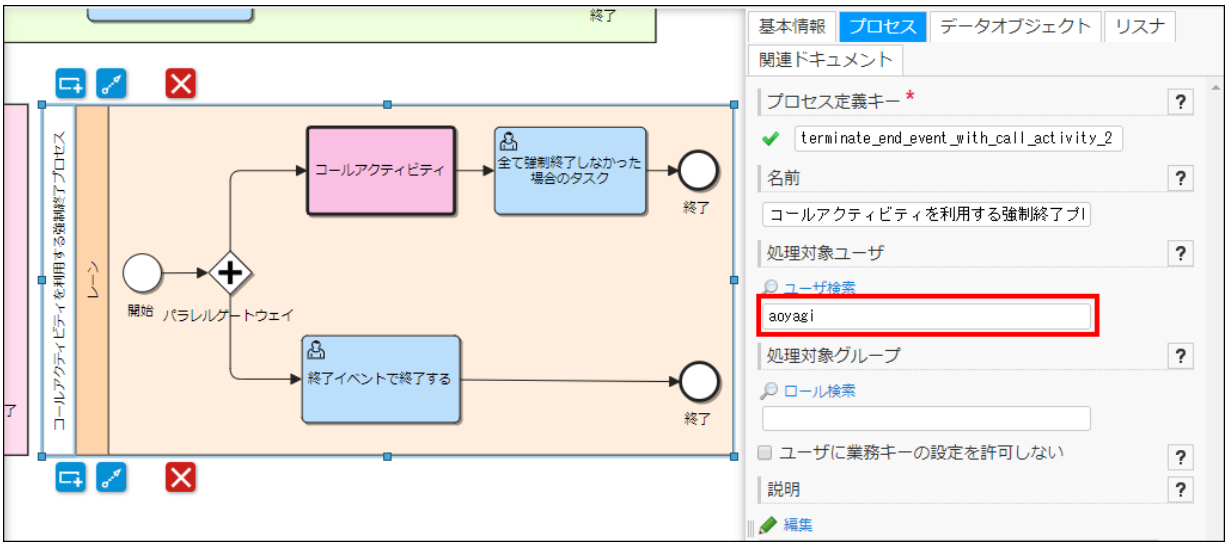


図：「強制終了イベント」 -「プロパティ」 -「メインコンフィグ」 -「関連するプロセスもすべて含めて強制終了する」

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。

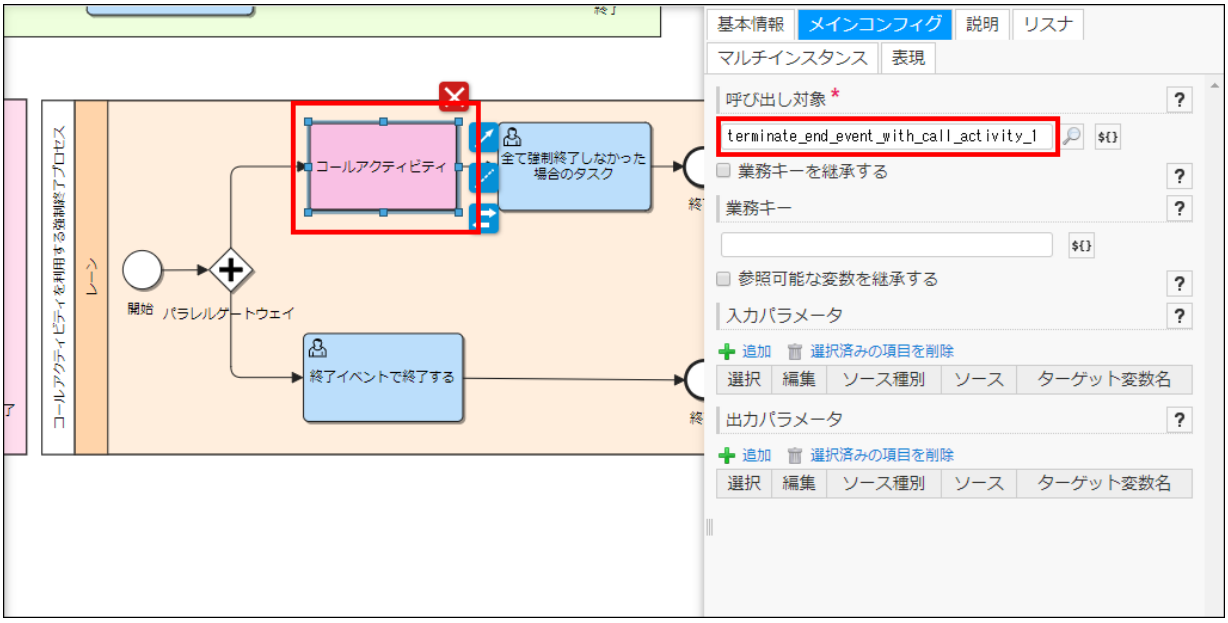


図：「プール」 - 「プロパティ」 - 「プロセス」 - 「処理対象ユーザ」

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。

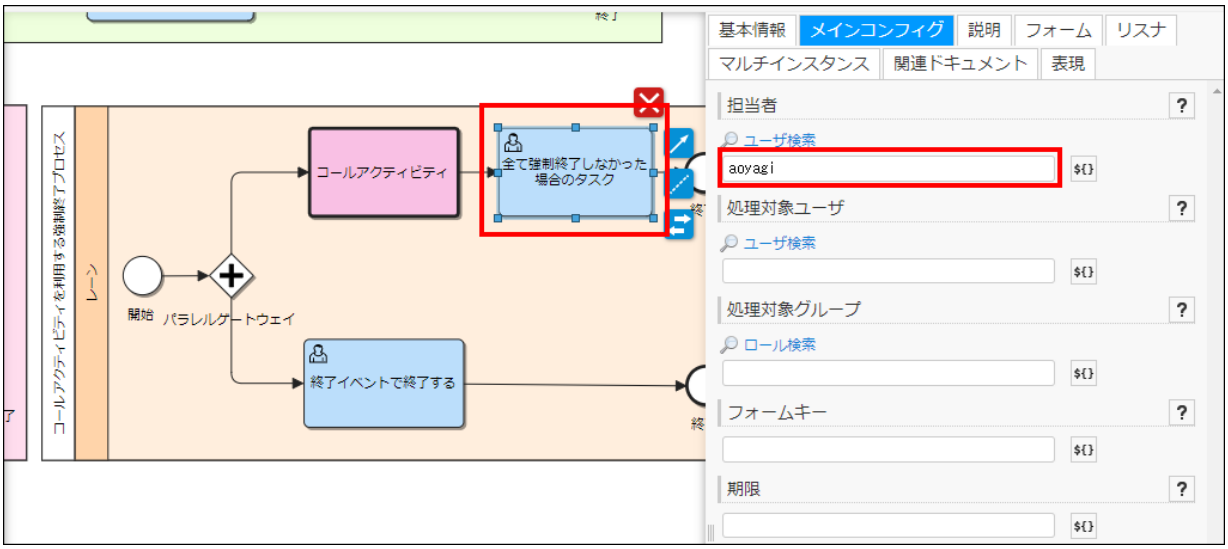
17. コールアクティビティで呼び出すプロセスを設定します。
「メインコンフィグ」タブから「呼び出し対象」に、「コールアクティビティから呼び出される強制終了プロセス」のプロセス定義ID「terminate_end_event_with_call_activity_1」を入力します。



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。

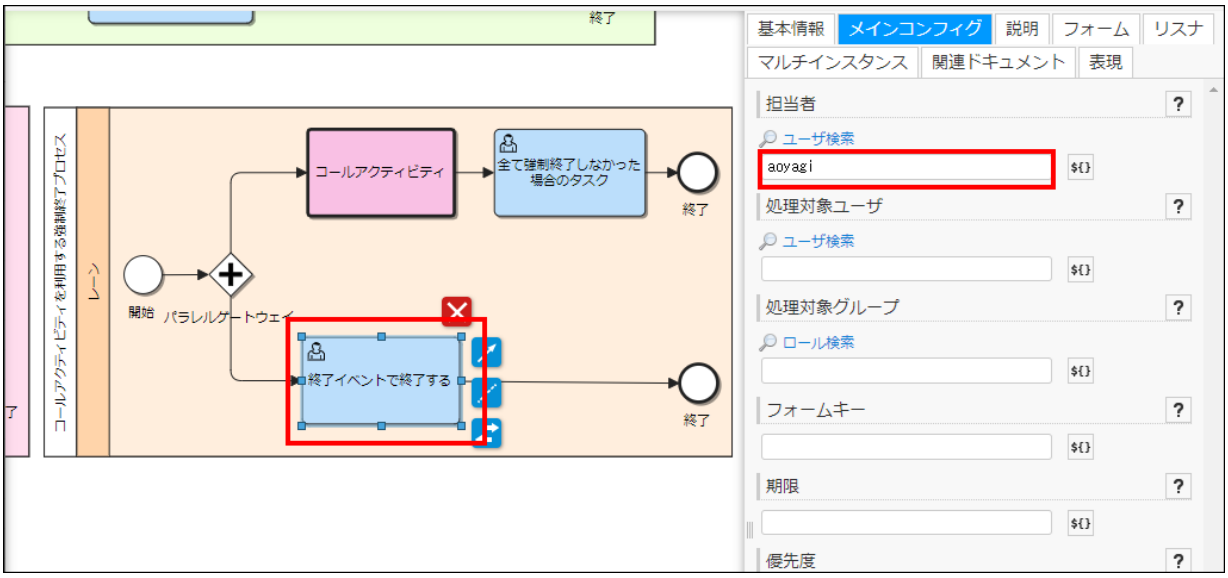


図：「ユーザタスク」 -「プロパティ」 -「メインコンフィグ」 -「担当者」

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。

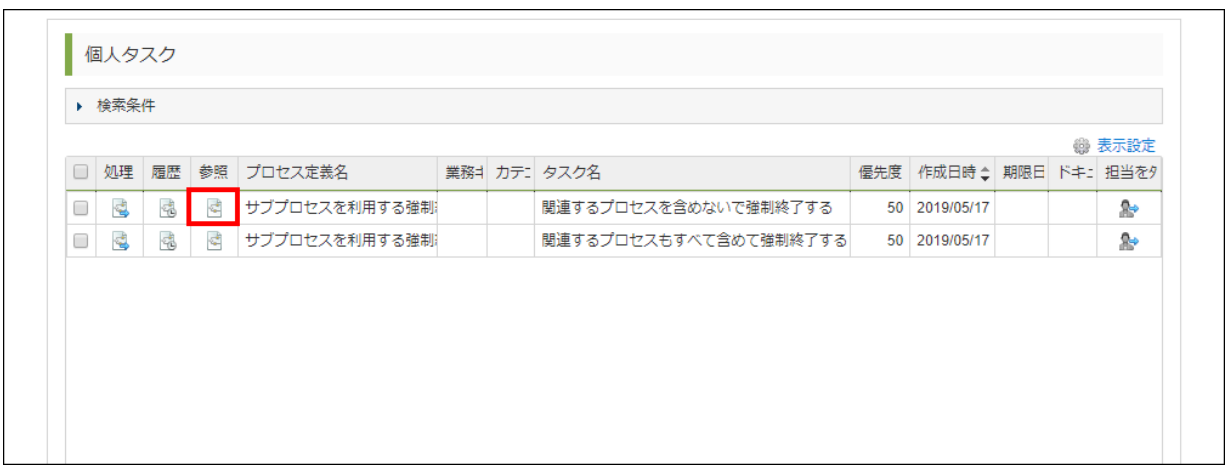


図: 「タスク一覧」 - 「個人タスク」

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

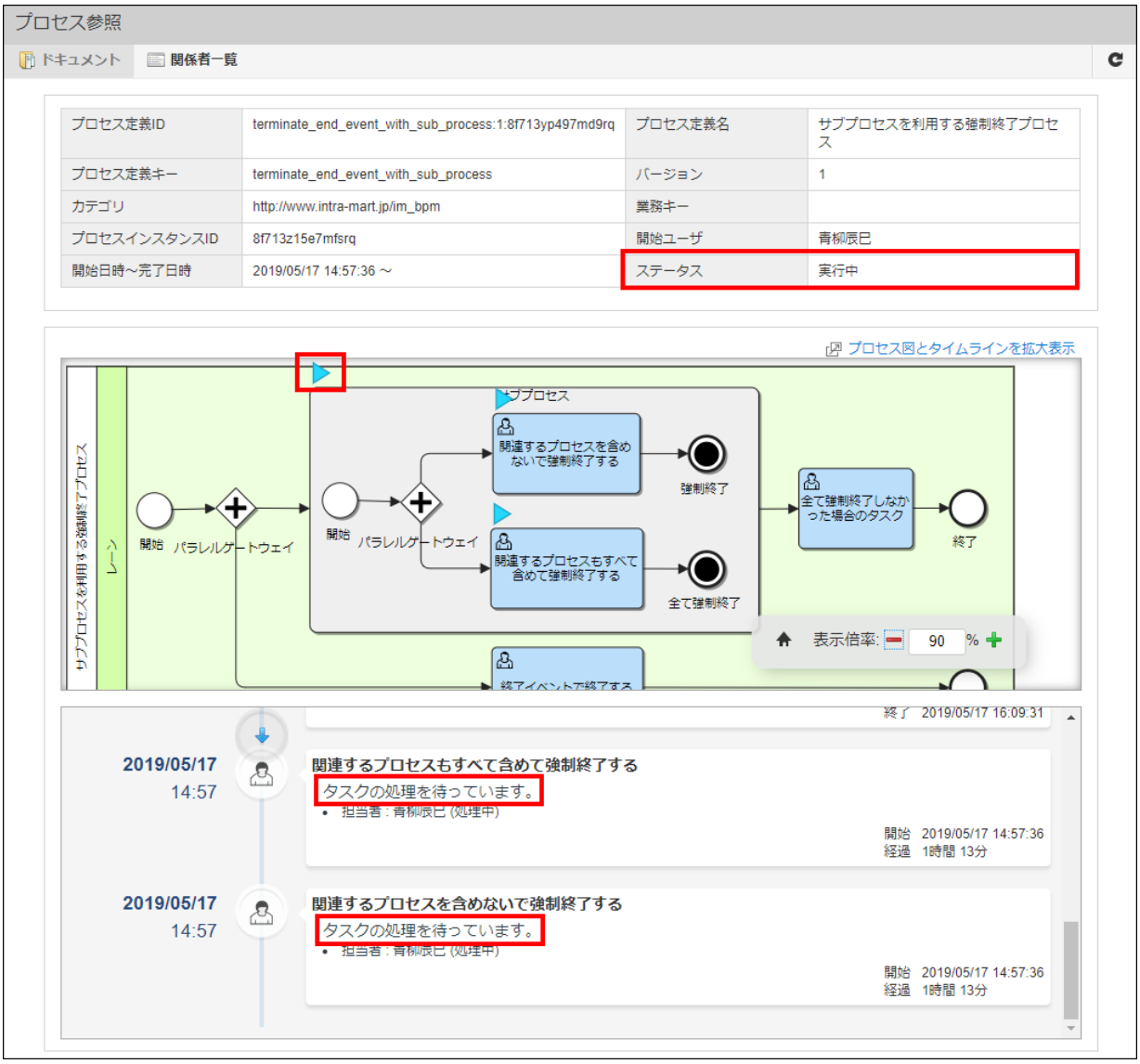


図: 「タスク一覧」 - 「個人タスク」

「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。

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、

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

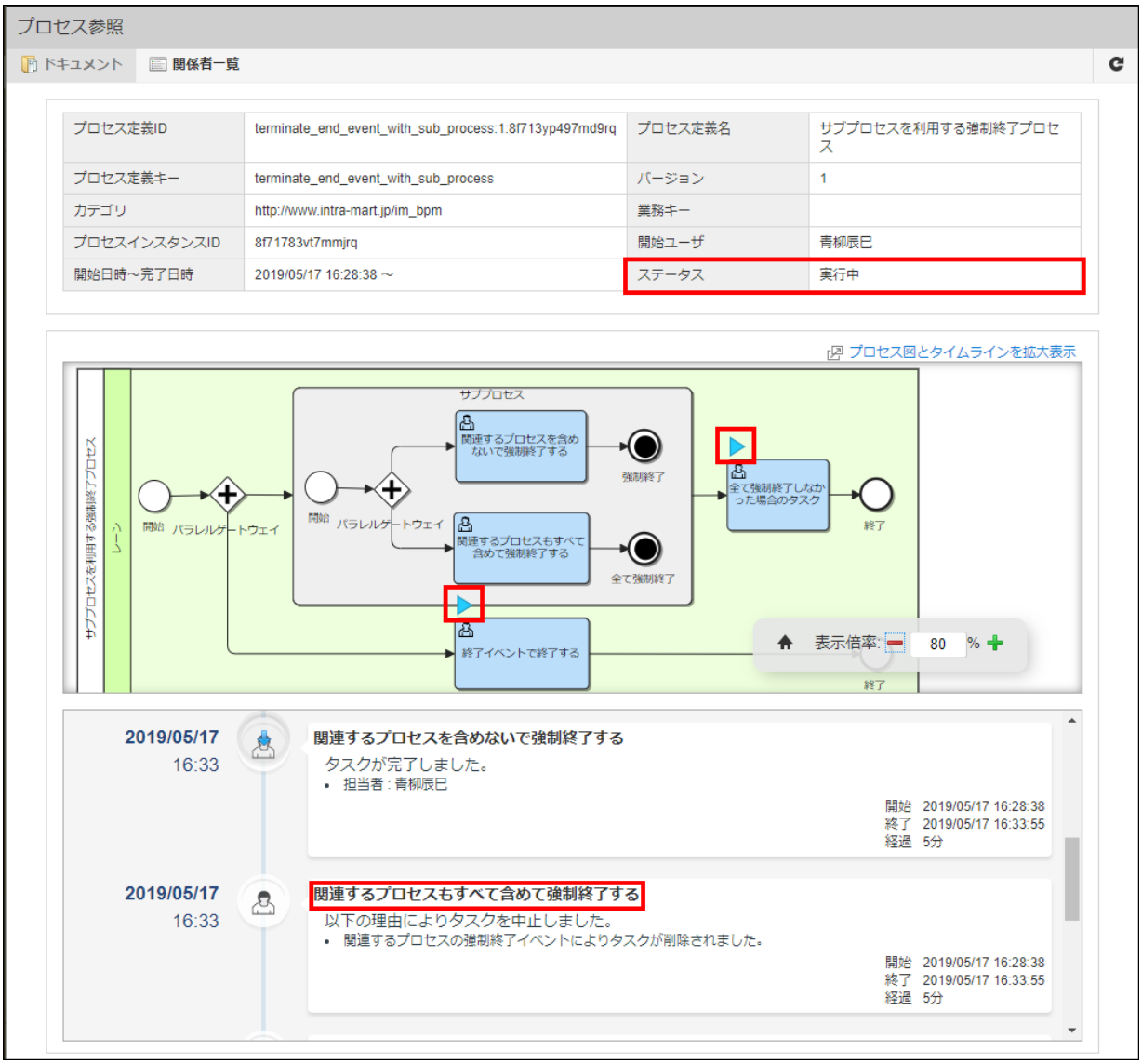


図: 「プロセス参照」

「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。

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。

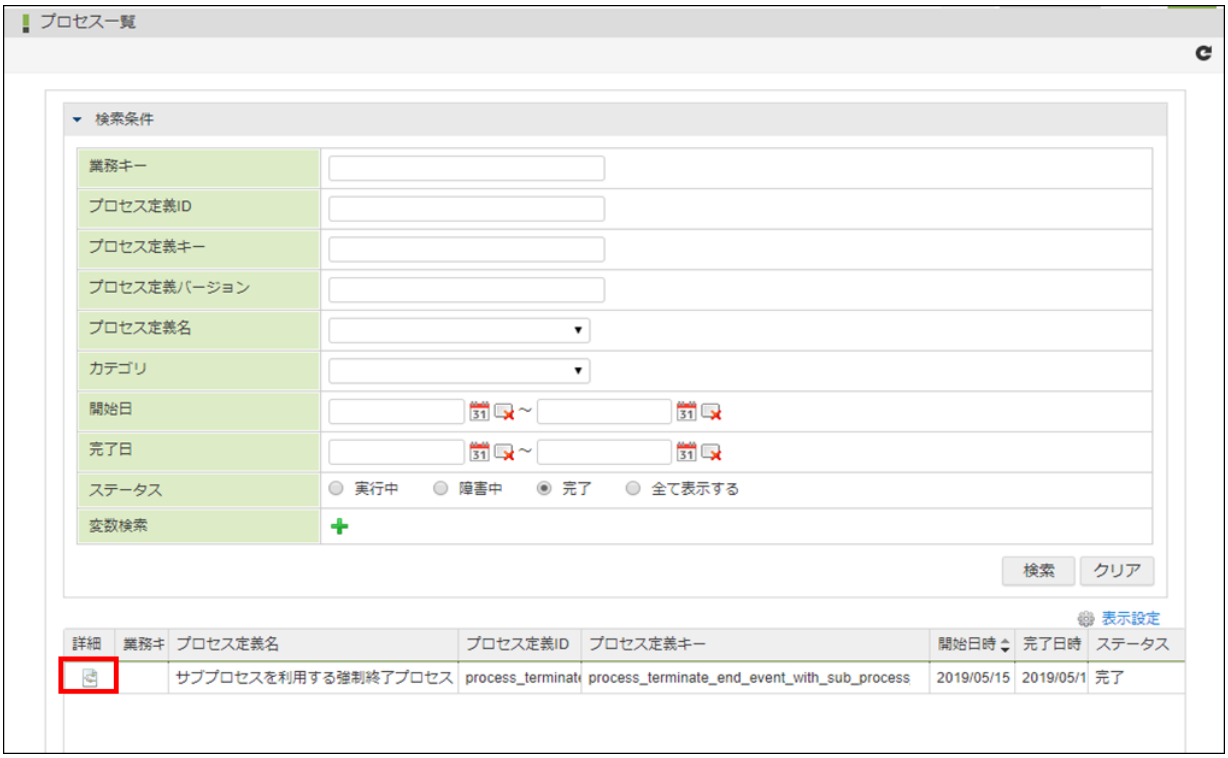


図: 「プロセス一覧」

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

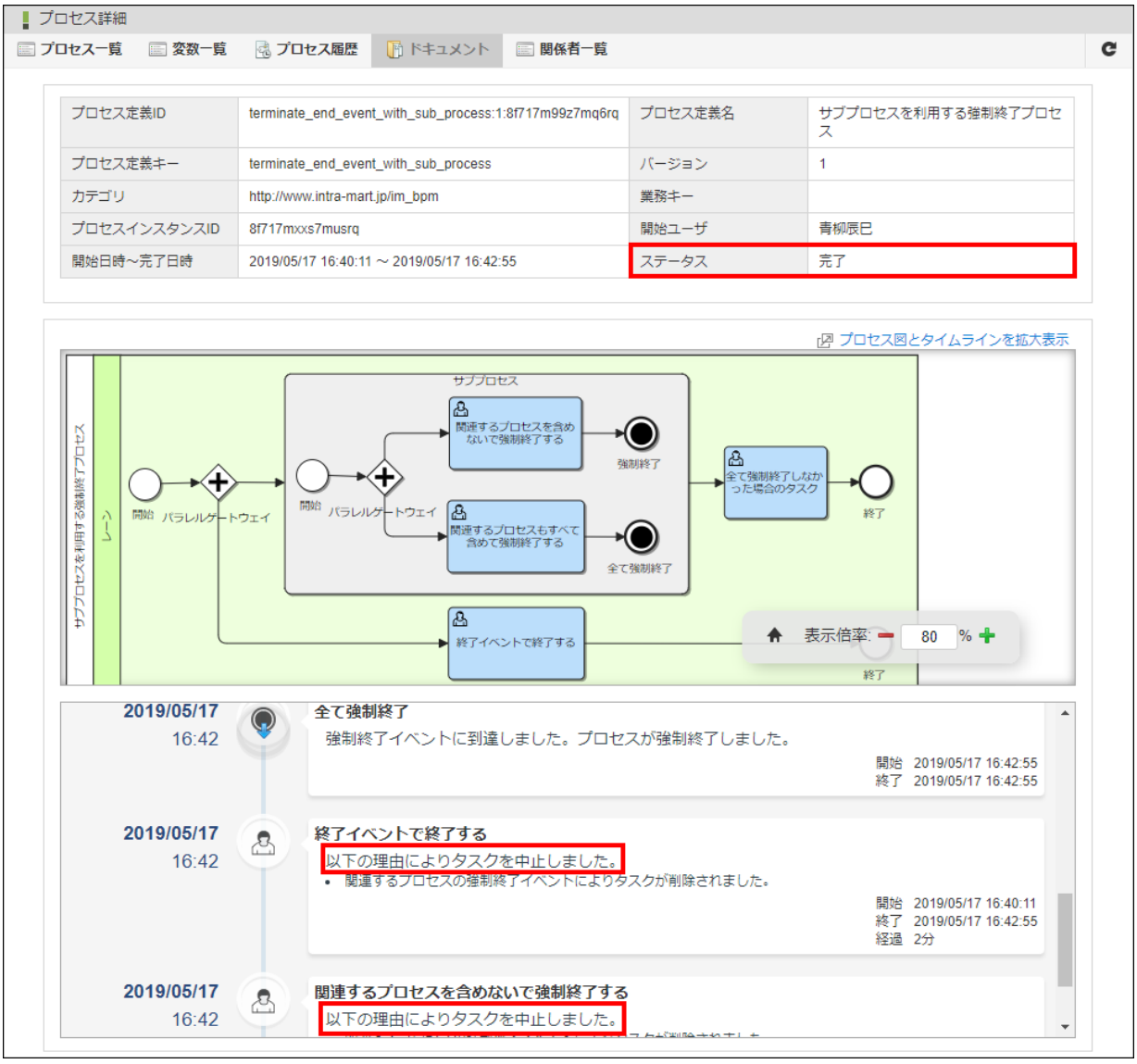
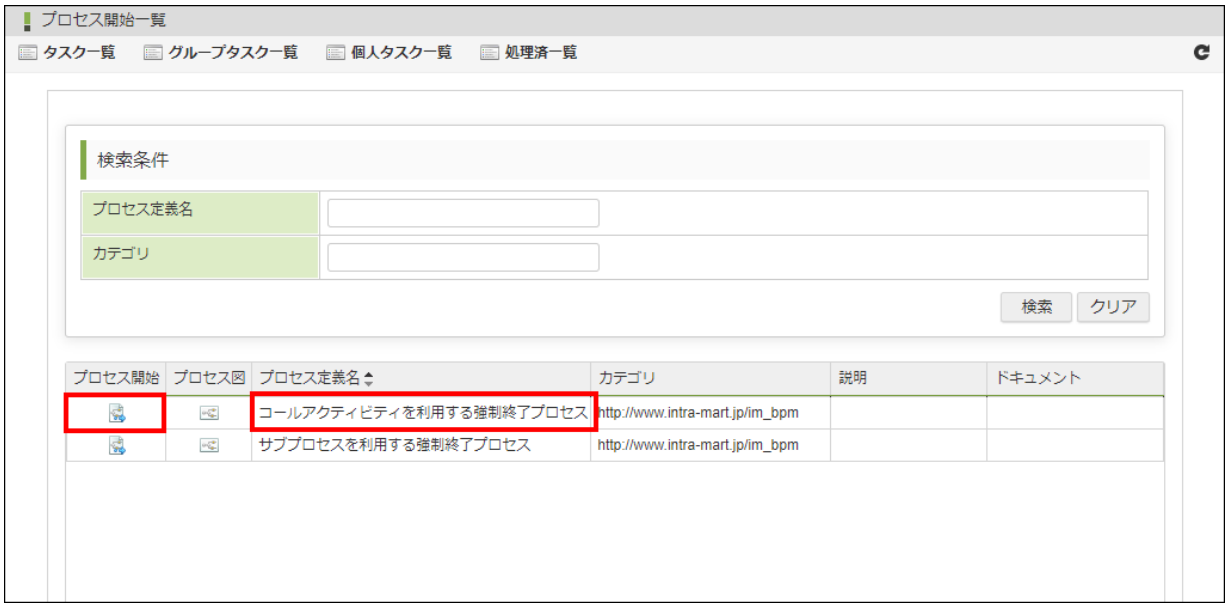


図: 「プロセス詳細」

「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。



図：「プロセス開始一覧」

終了イベ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。

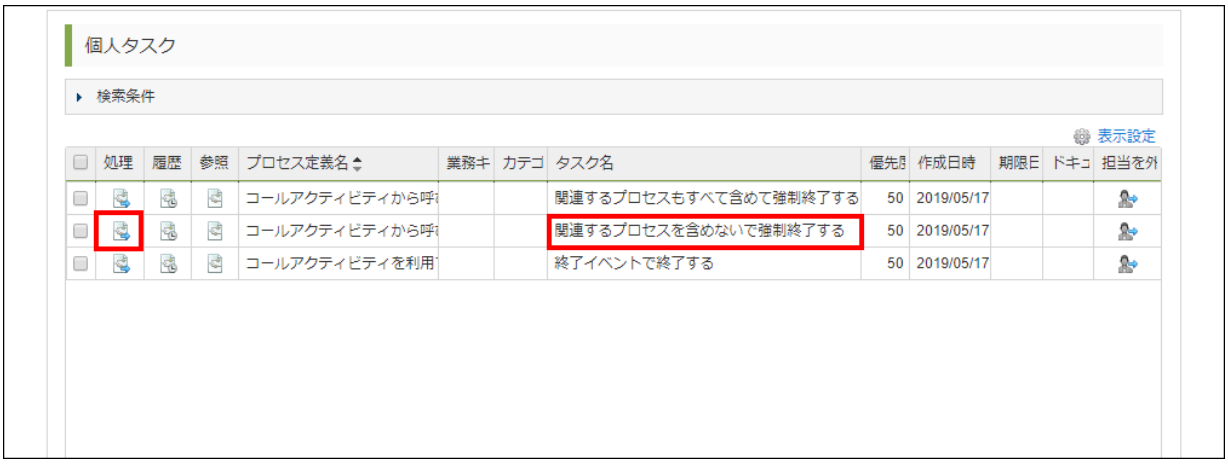


図: 「タスク一覧」 - 「個人タスク」

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。

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

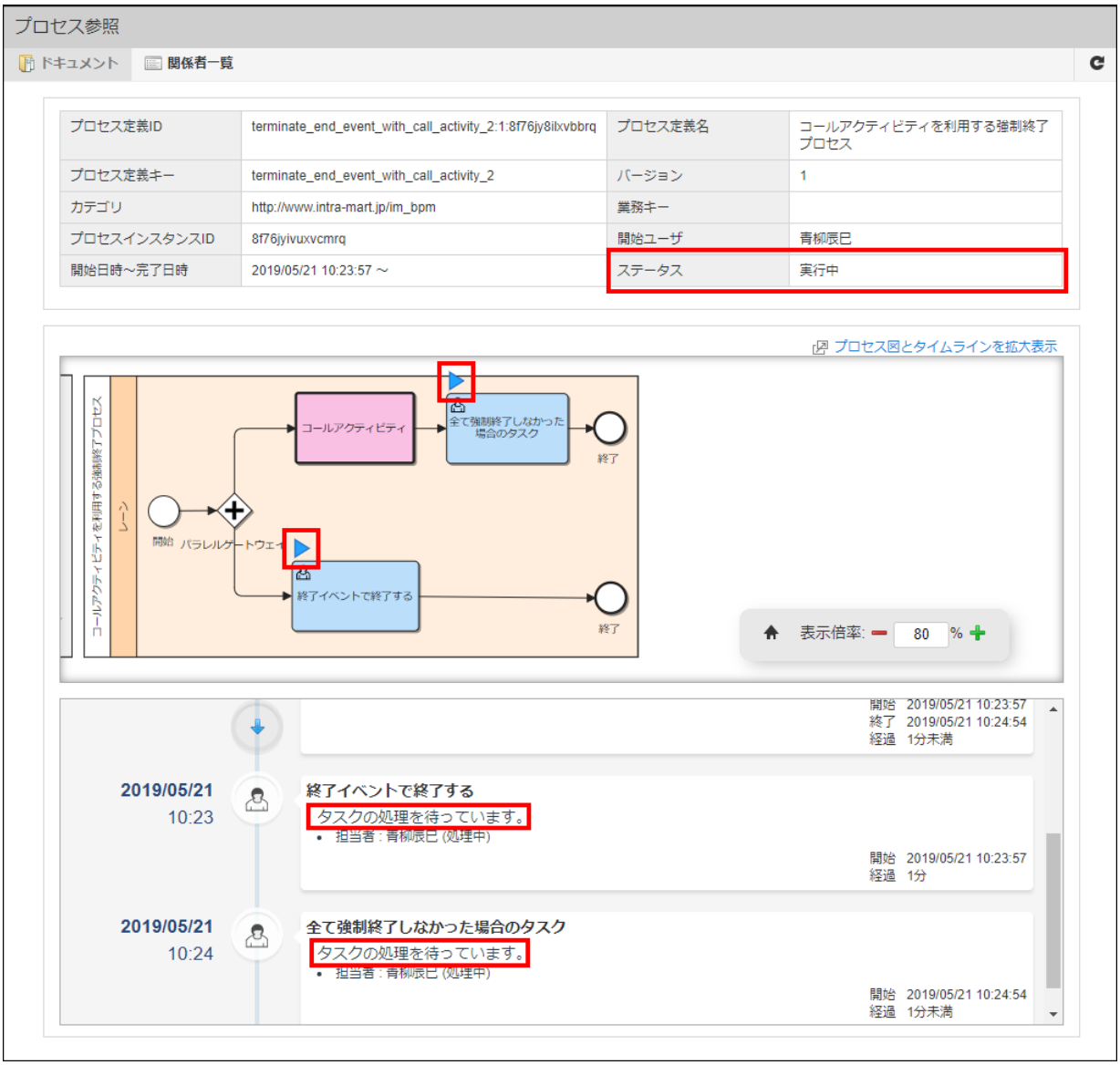


図:「プロセス詳細」

「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。

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。

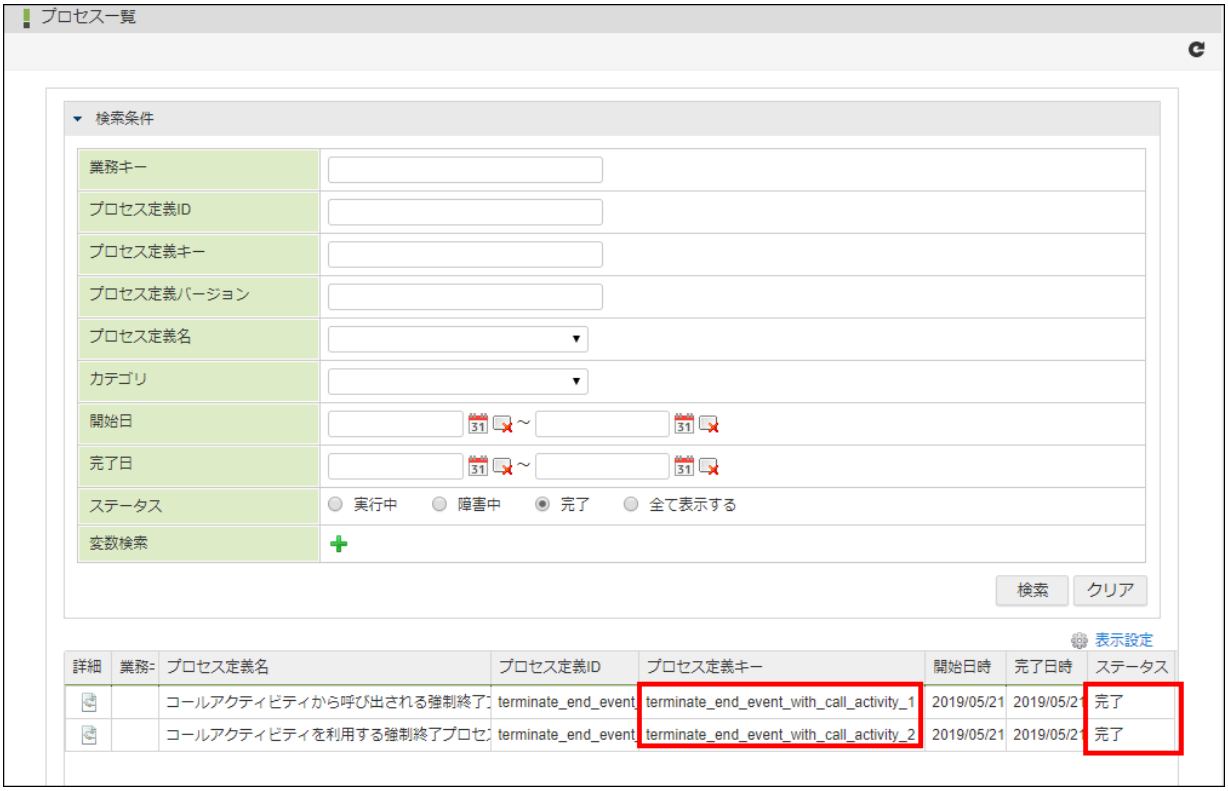


図: 「プロセス一覧」

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。

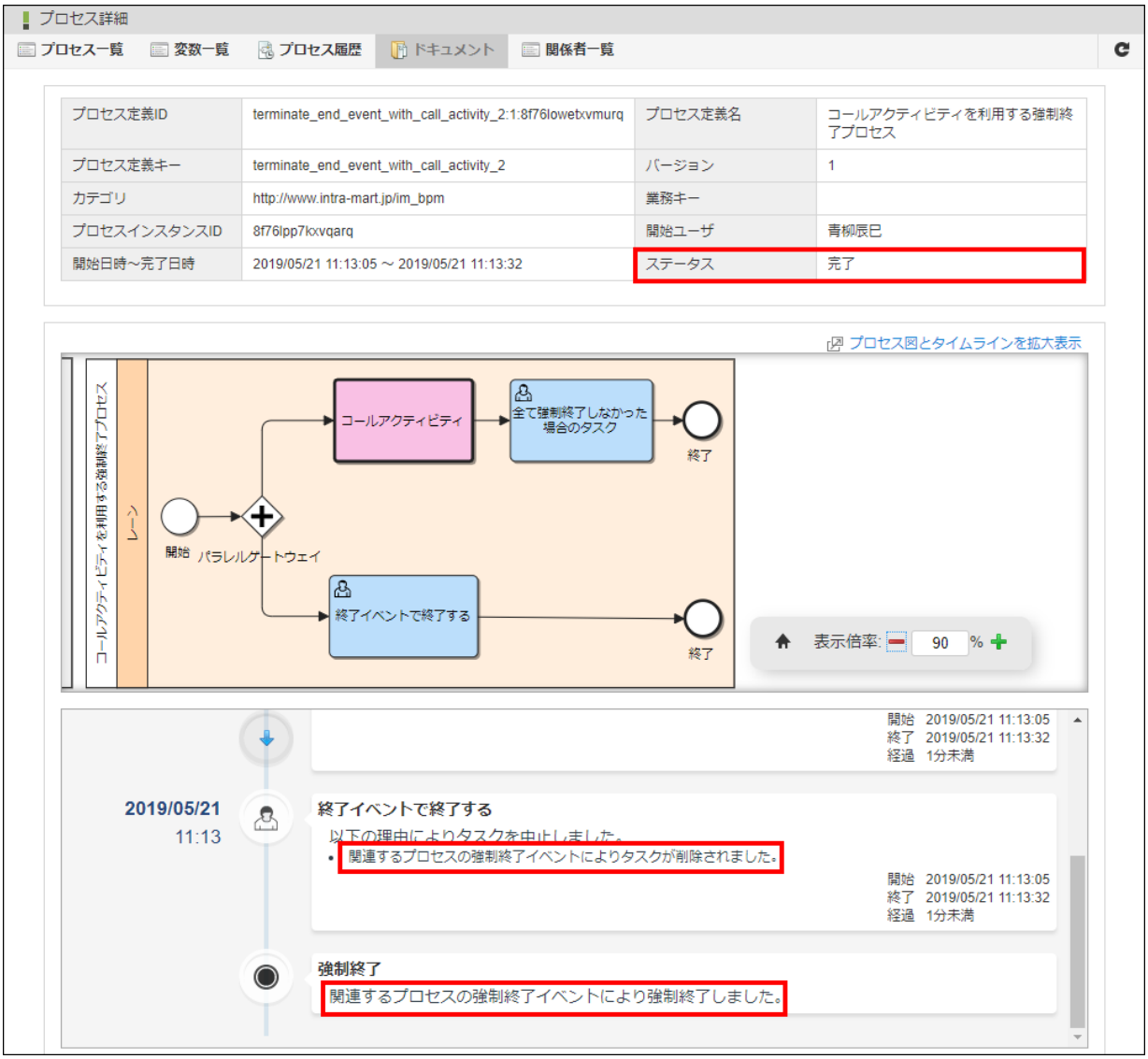


図:「プロセス詳細」

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 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。

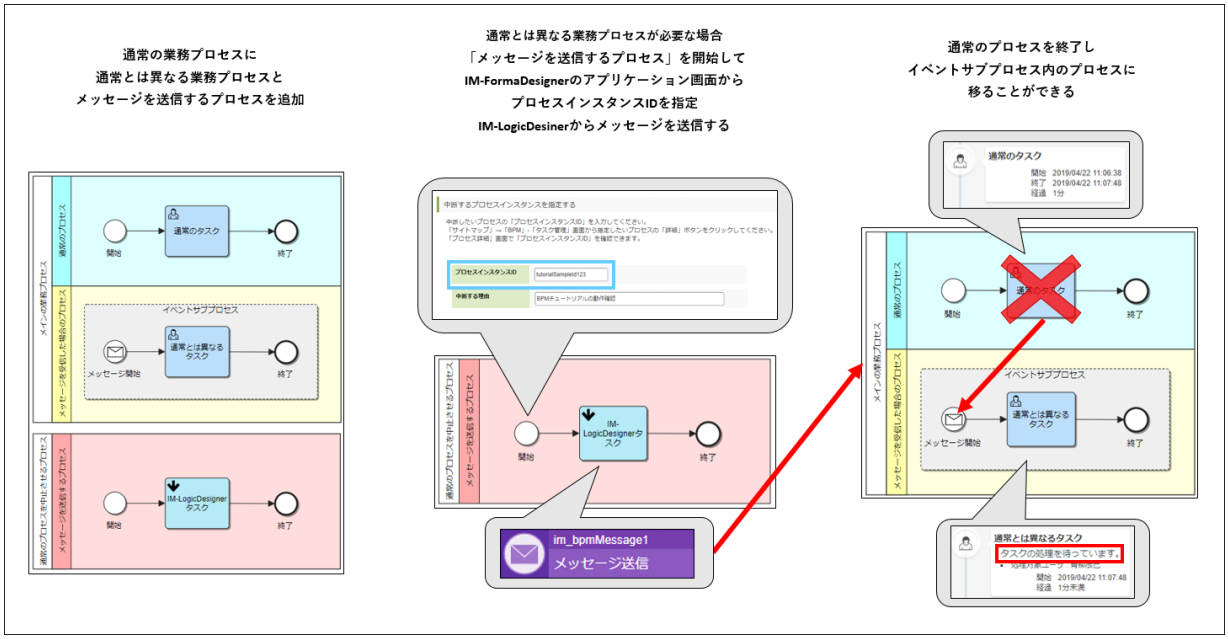


図 ： 概要図

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材をインポートしてください。

- ロジックフロー

  im_logic_designer-event_sub_process_usage.zip

- Formaアプリケーション

  im_forma_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 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: 「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-LogicDesignerのロジックフローを確認する
- 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
  1. ユーザタスクを処理する通常のプロセス
  2. 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
  1. 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、「IM-LogicDesignerタスク」を使用してメッセージを送信するプロセス

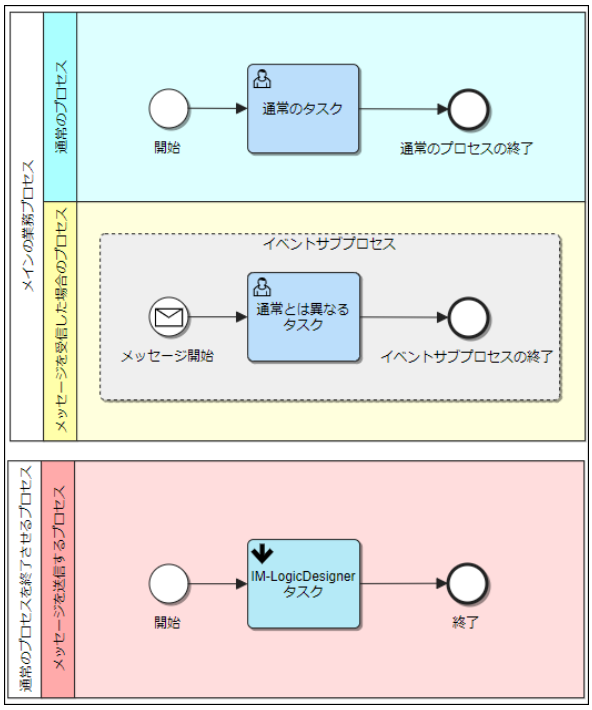
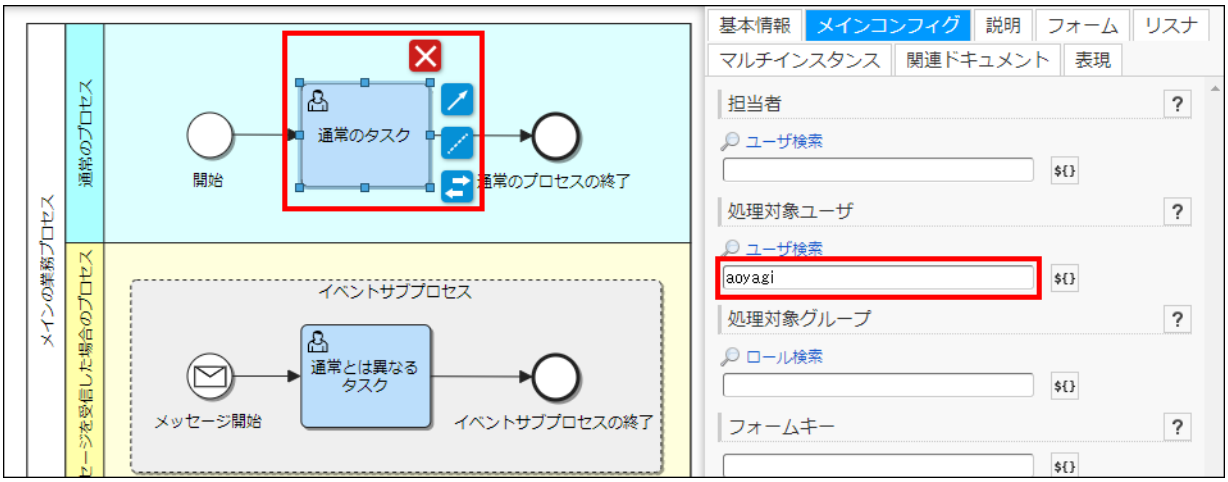


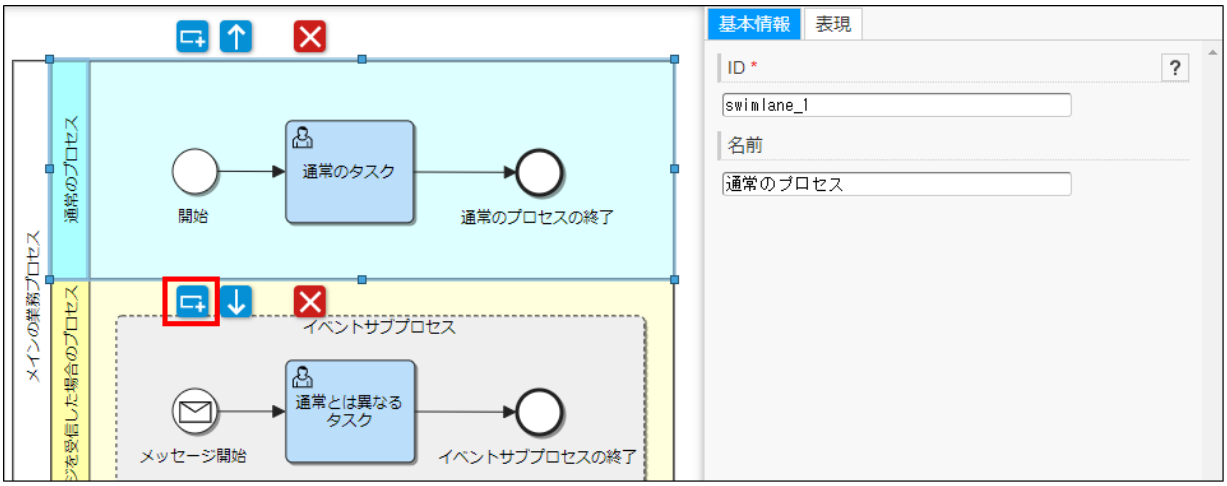
図 ： 完成イメージ

1. 「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。
2. 「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。
3. 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。
4. 「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。



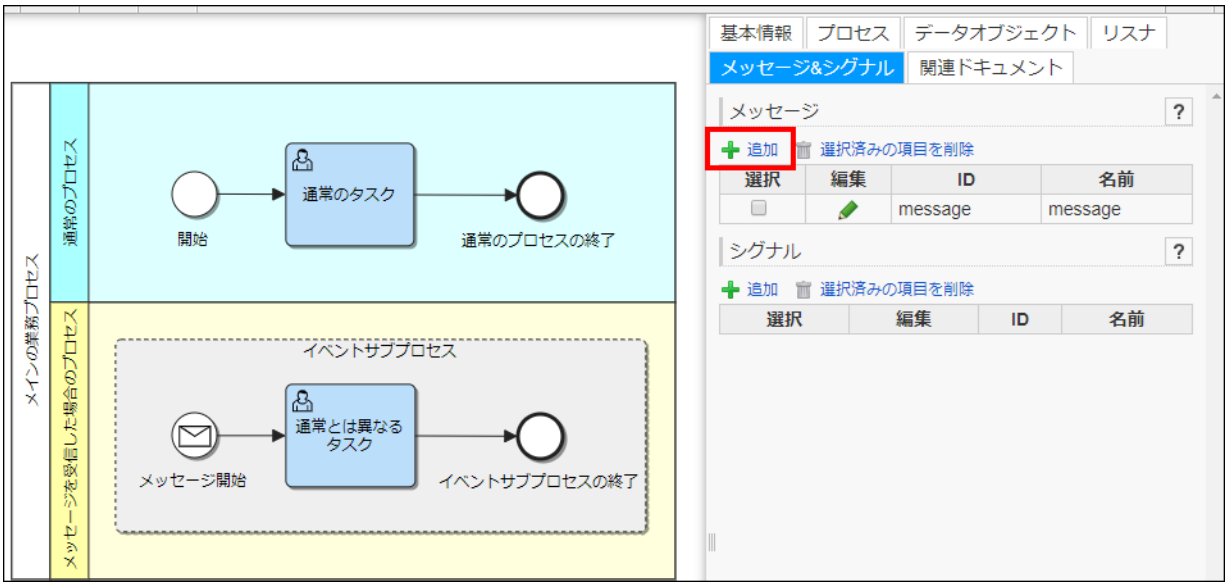
図：「メインの業務プロセス」 - 「通常のプロセス」 - 「プロパティ」 - 「メインコンフィグ」

5. 終了イベントを設置します。
6. 「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。



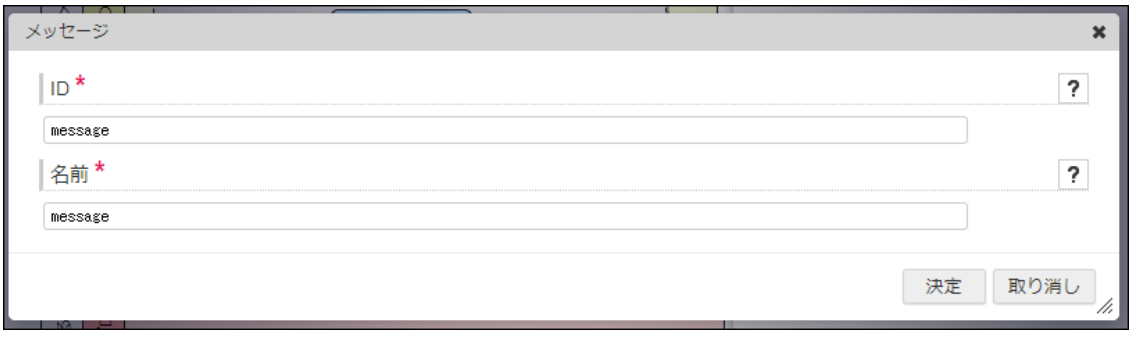
図：「メインの業務プロセス」 - 「通常のプロセス」

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バ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。



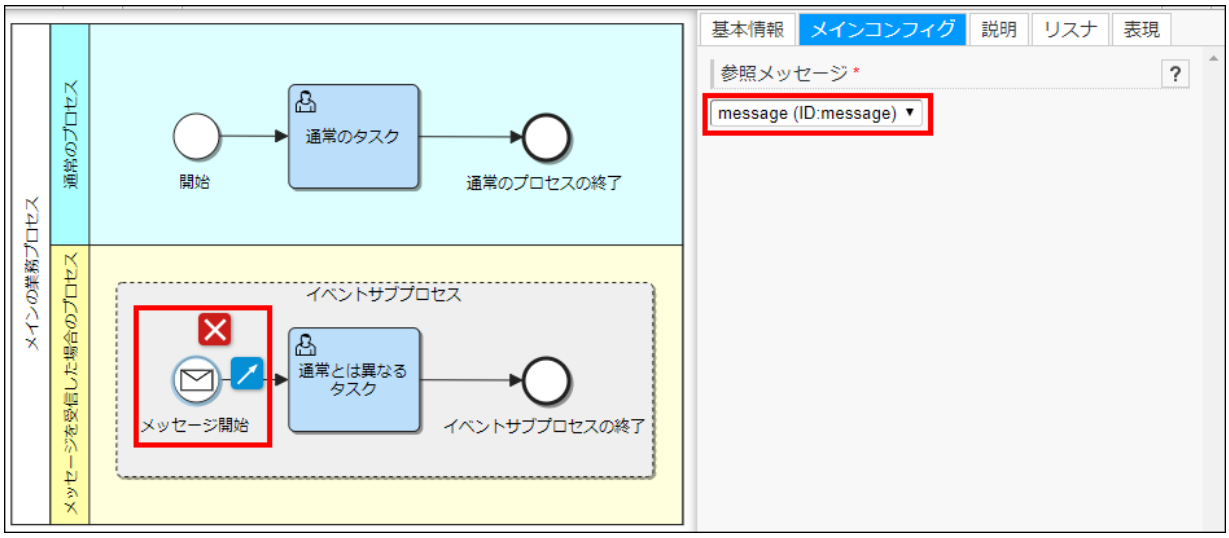
図：プロセス全体 - 「プロパティ」 - 「メッセージ&シグナル」

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」

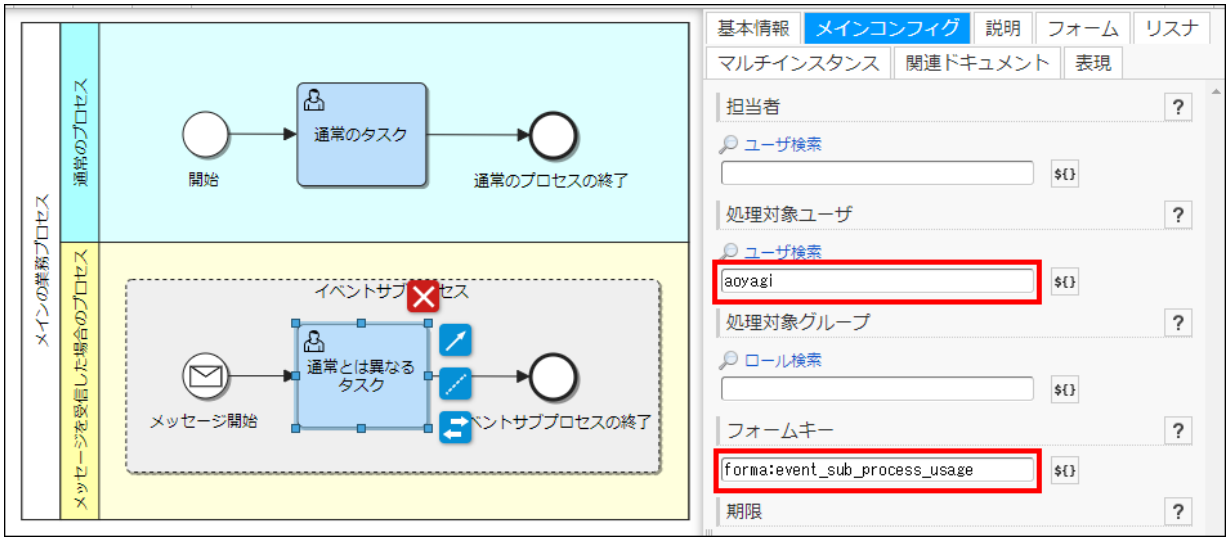
11. プロセス全体に登録した「メッセージ」を「メッセージ開始イベント」に紐づけます。
    「メッセージ開始イベント」をクリックし、「メインコンフィグ」タブの「参照メッセージ」から行います。
    ドロップダウンリストで「 message(ID:message) 」を選択します。



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」

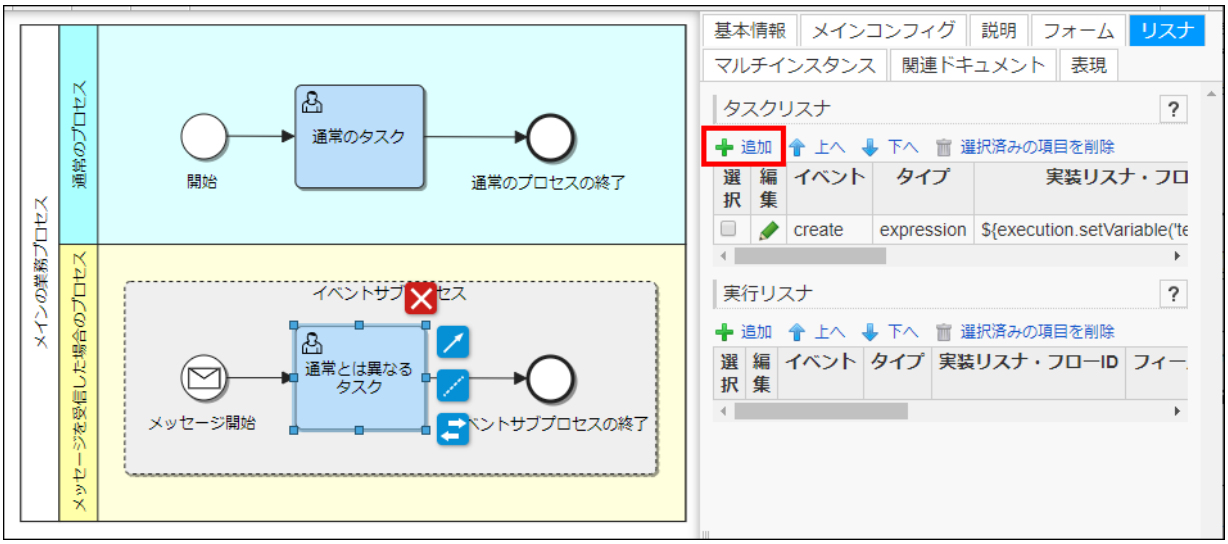
12. メッセージを受信した場合に処理するユーザタスク「通常とは異なるタスク」を設置します。

13. 処理対象ユーザと、表示するForma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



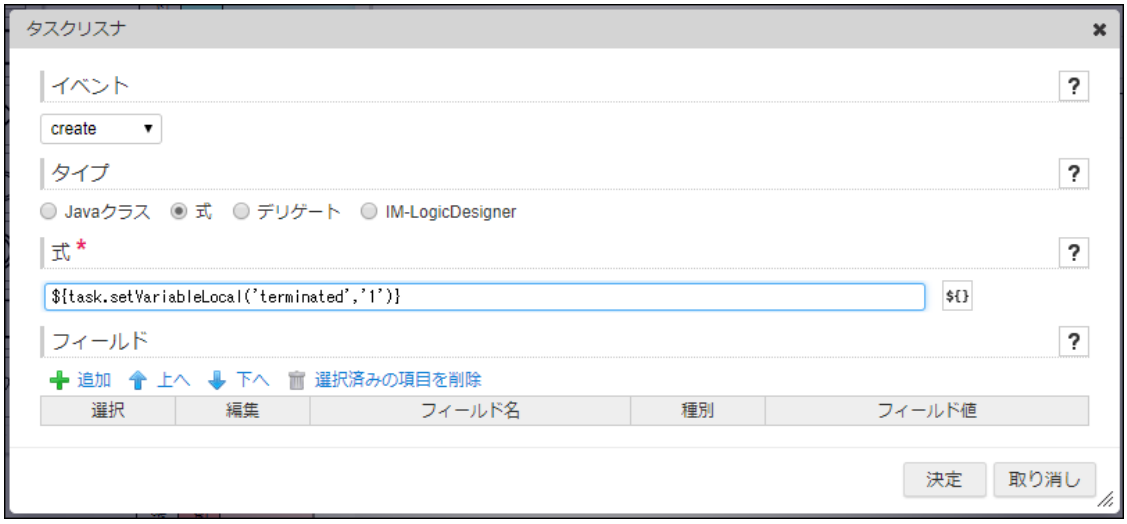
図：「メインの業務プロセス」 - 「メッセージを受信した場合のプロセス」 - 「イベントサブプロセス」 - 「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

14. 表示するForma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')}

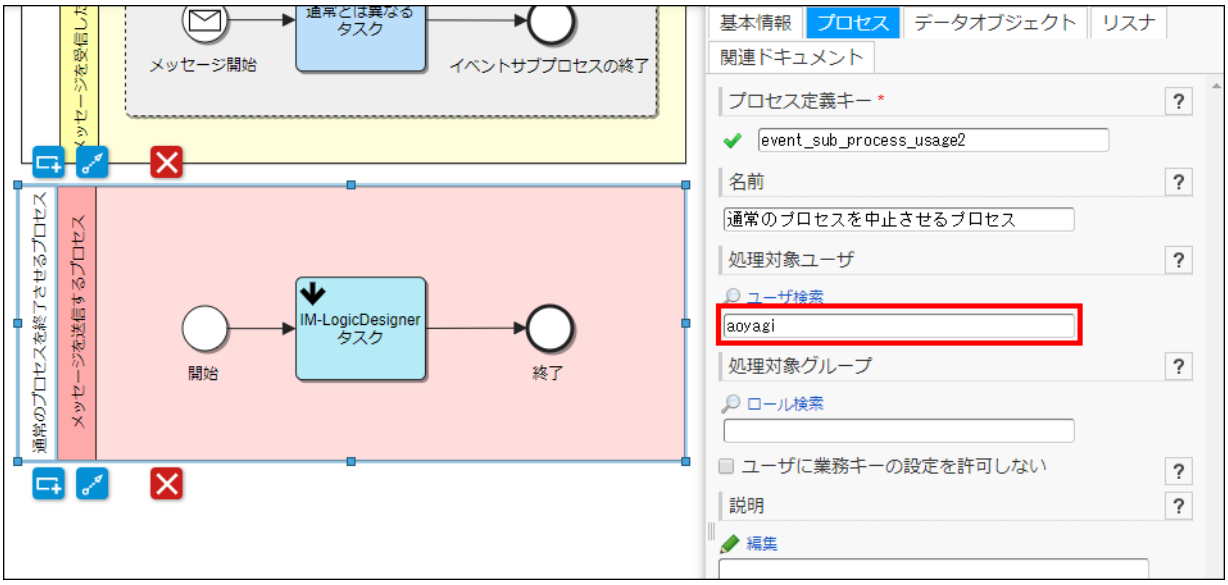


図：「タスクリスナ」

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。

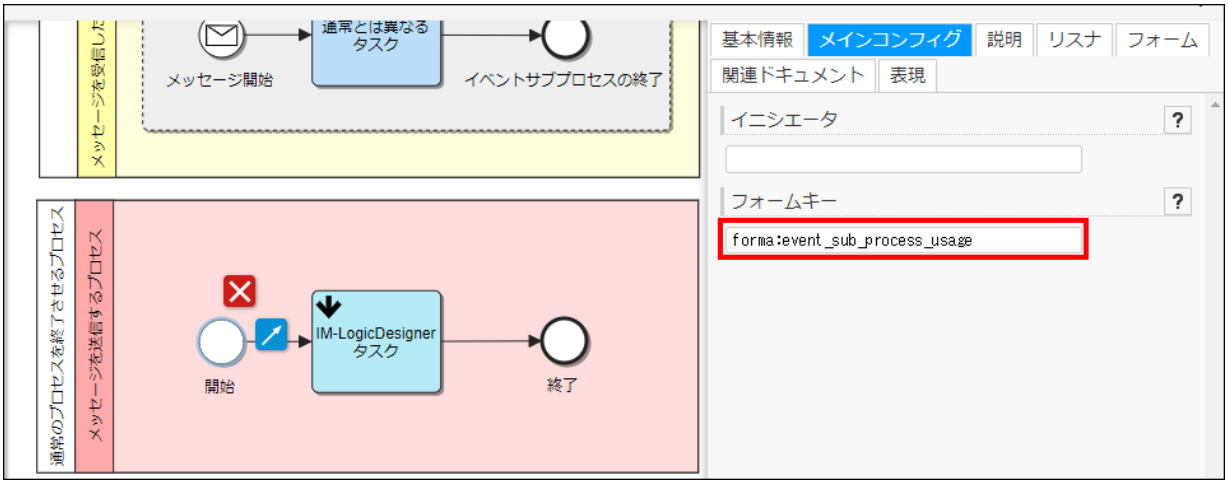
18. 「通常のプロセスを終了させるプロセス」の処理対象ユーザを設定します。
    「プロセス」タブから、処理対象ユーザに「aoyagi」を設定します。



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。

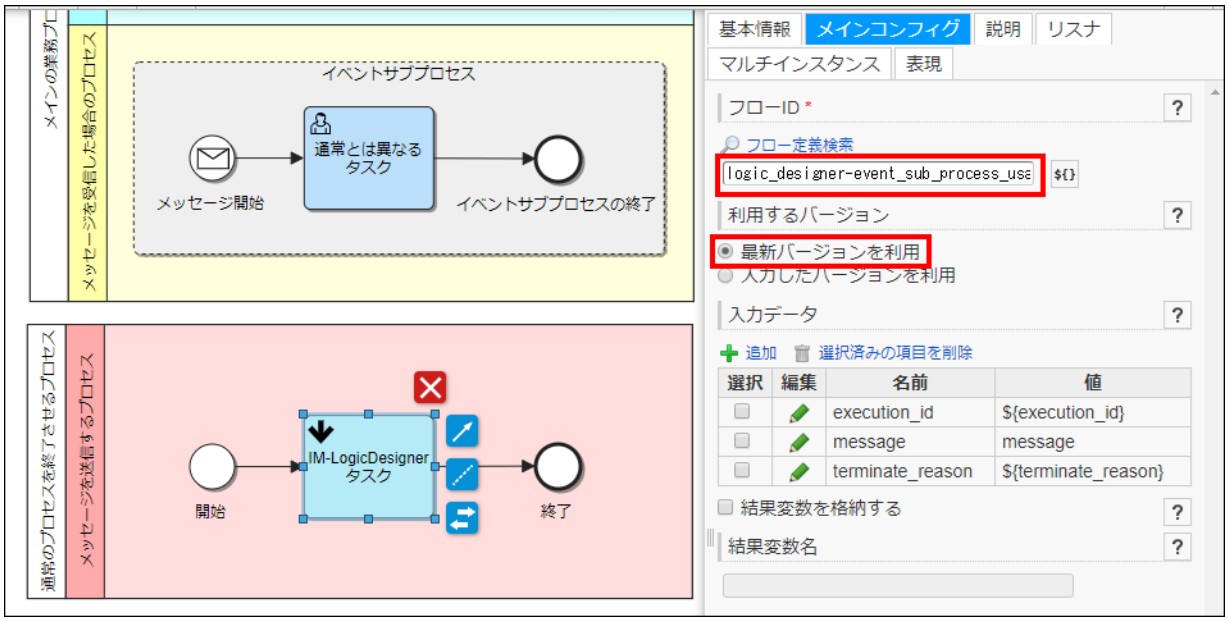
20. 「通常のプロセスを終了させるプロセス」を開始した際に表示するFormaアプリケーションを設定します。
開始イベントをクリックし、「メインコンフィグ」タブから「フォームキー」に「forma:event_sub_process_usage」と設定します。



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」

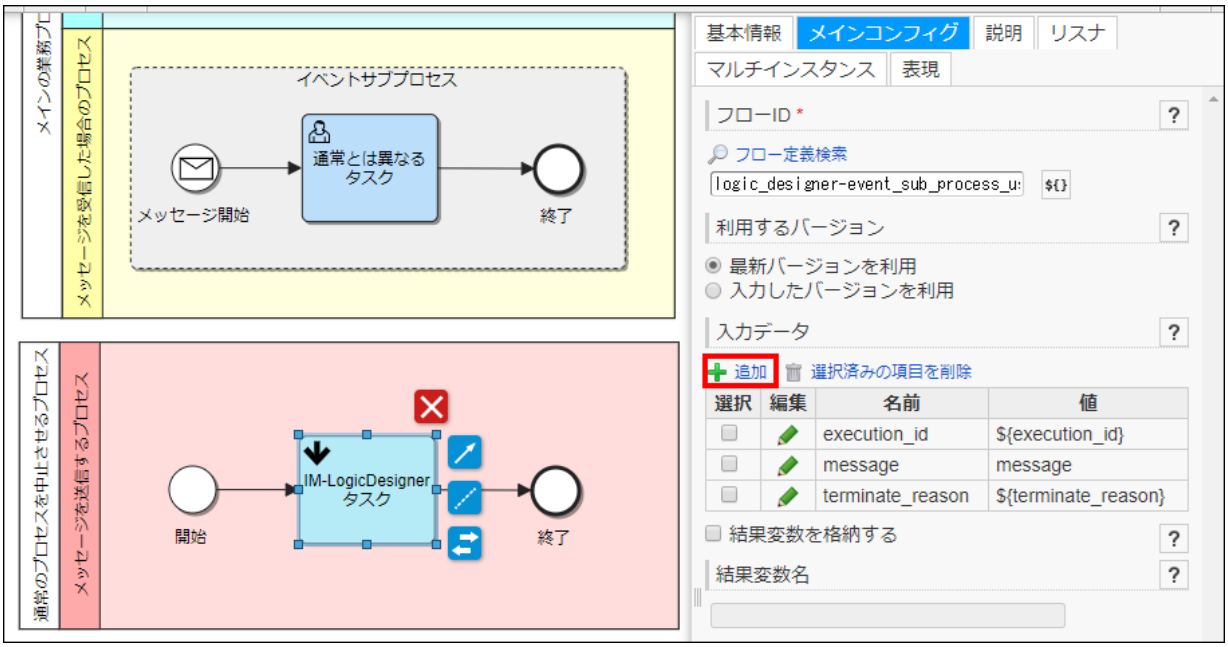
21. 「メッセージ開始イベント」に対してメッセージを送信する「IM-LogicDesignerタスク」を設置します。

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」

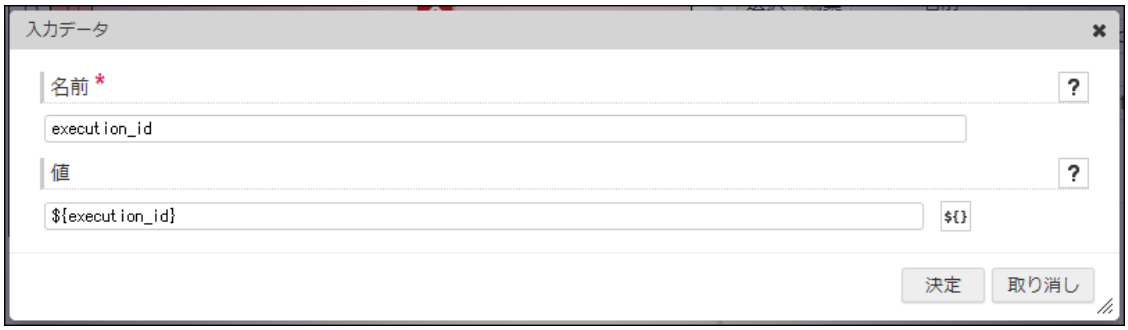
23. 「IM-LogicDesigner」に対する「入力データ」を設定します。
入力データの「追加」リンクをクリックします。



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}

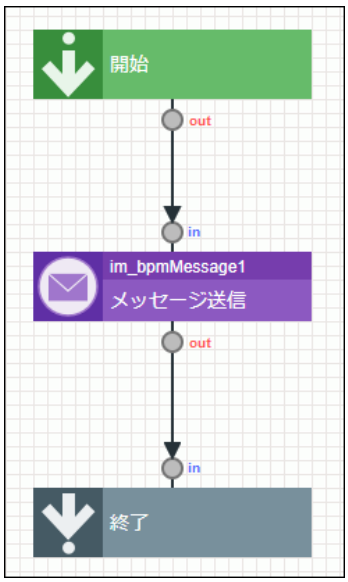


図：「入力データ」

25. 「終了イベント」を設置します。

IM-LogicDesignerのロジックフローを確認する

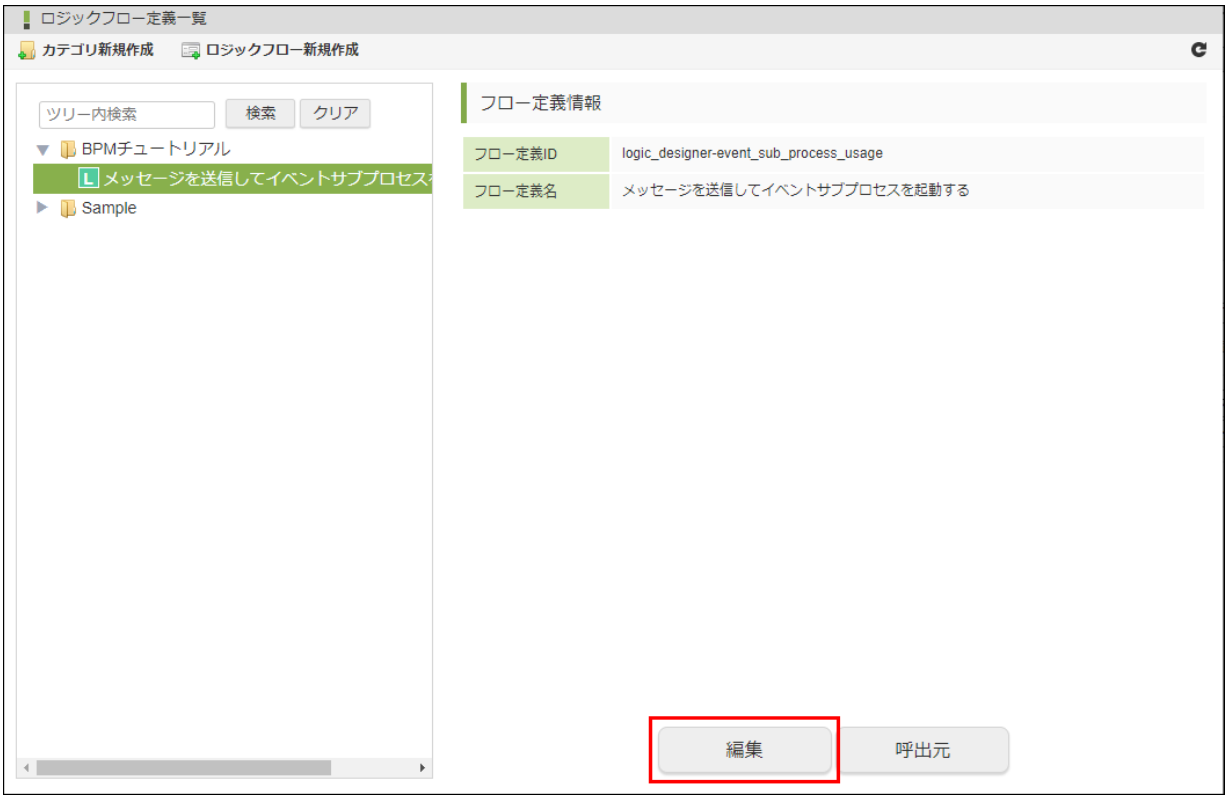
チュートリアル開始前にインポートしたIM-LogicDesignerのロジックフローを確認します。
上記BPMの「IM-LogicDesignerタスク」が実行されることで、メッセージを送信します。



完成イメージ（IM-LogicDesigner）

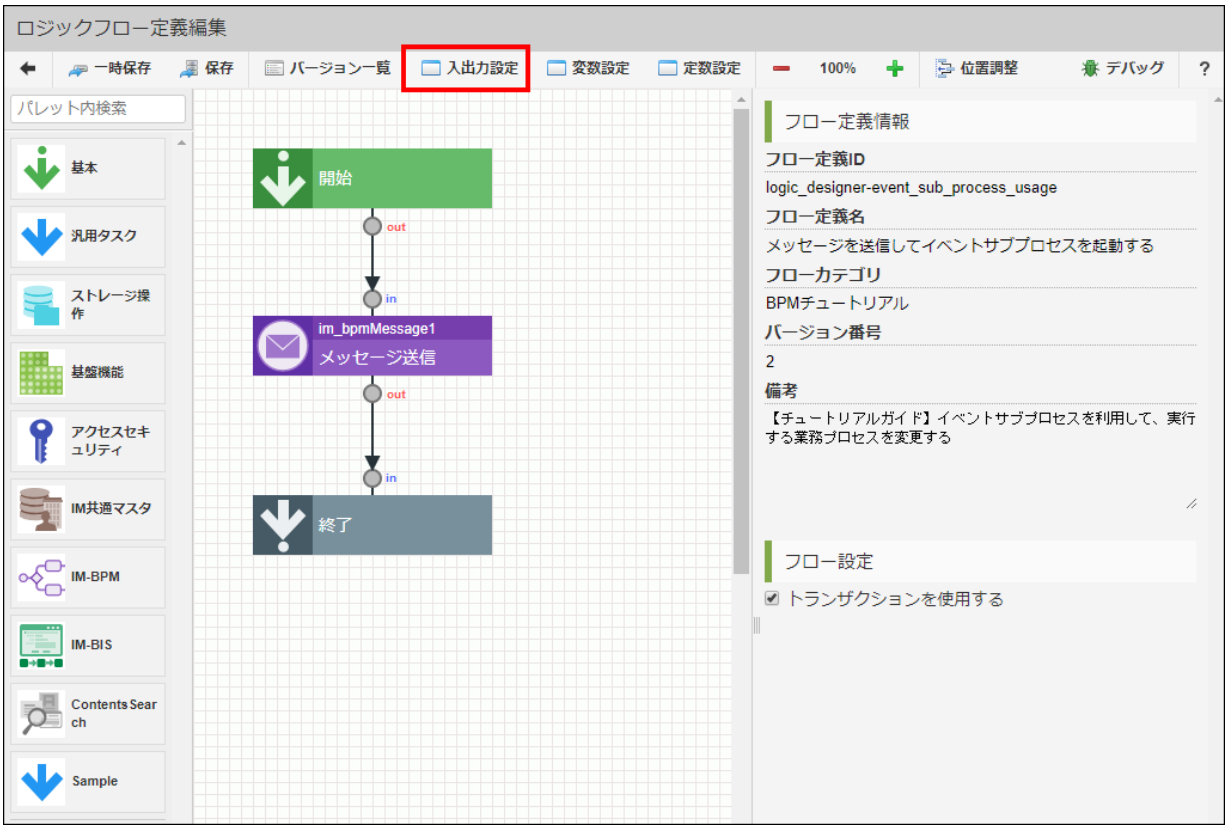
1. チュートリアル開始前にインポートしたロジックフローを確認します。
   「サイトマップ」→「IM-LogicDesigner」→「フロー定義一覧」→「ロジックフロー定義一覧」画面を表示します。

2. 左ペインのツリーから「BPMチュートリアル」カテゴリの開閉アイコンをクリックしカテゴリ配下を情報を表示します。
   フロー定義「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」を選択し「編集」をクリックし、「ロジックフロー定義編集」画面を開きます。



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。

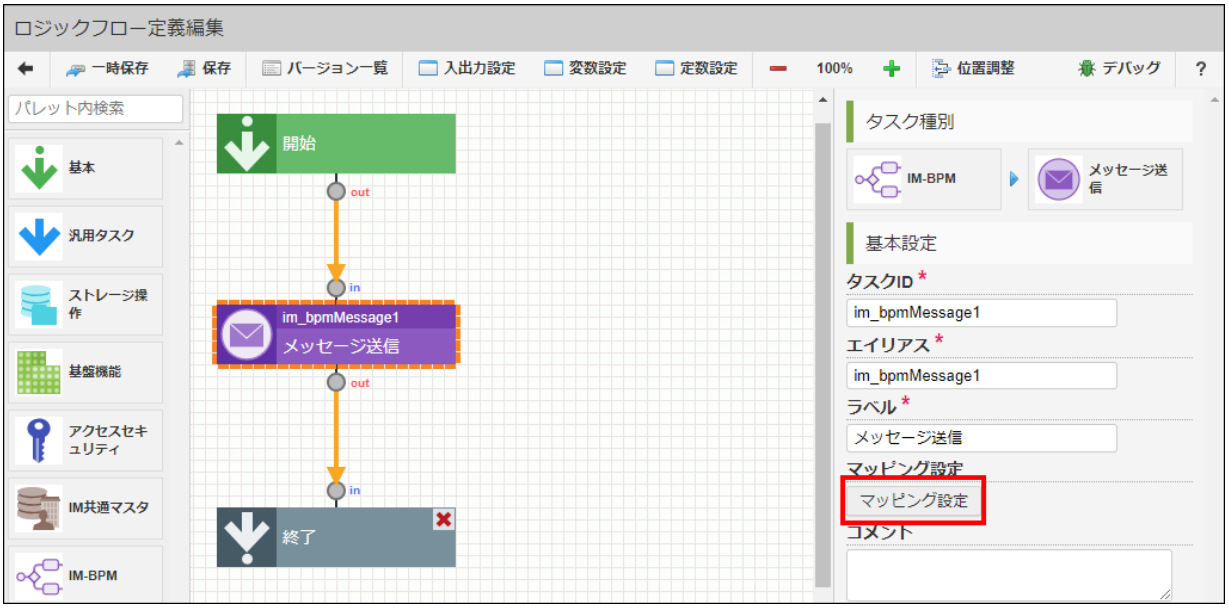


図：「ロジックフロー定義編集」

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」

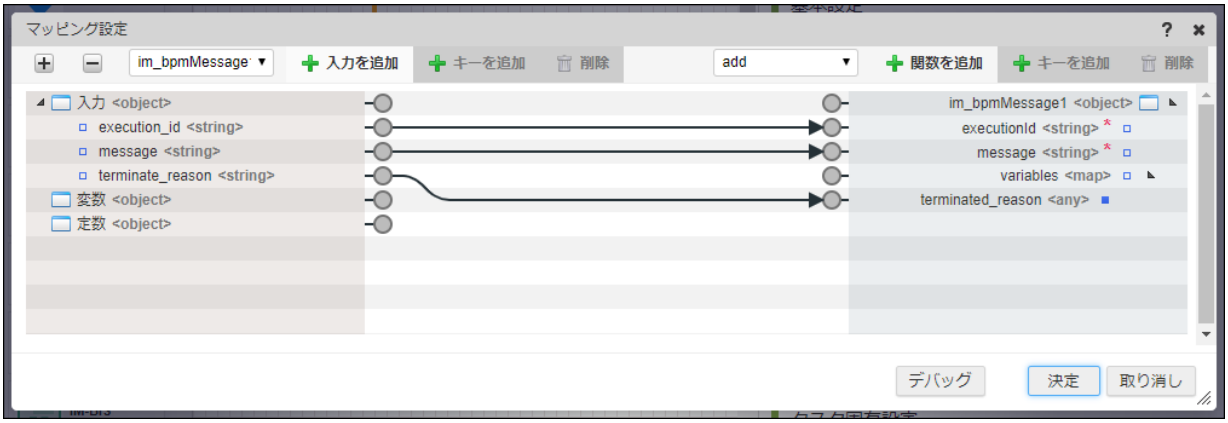
5. 「メッセージ送信」タスクのマッピングを確認します。
「メッセージ送信」タスクを選択し、基本設定の「マッピング設定」から「マッピング設定」ダイアログを表示します。



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」

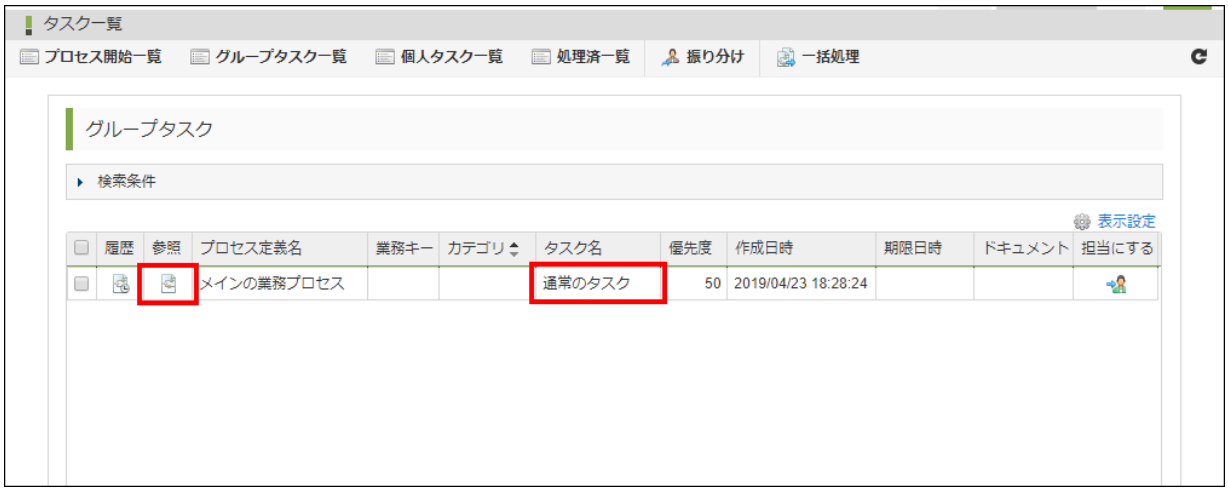
3. 実行したプロセスの状態と、プロセスインスタンスIDを確認します。
画面左上の「タスク一覧」をクリックしてください。

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。

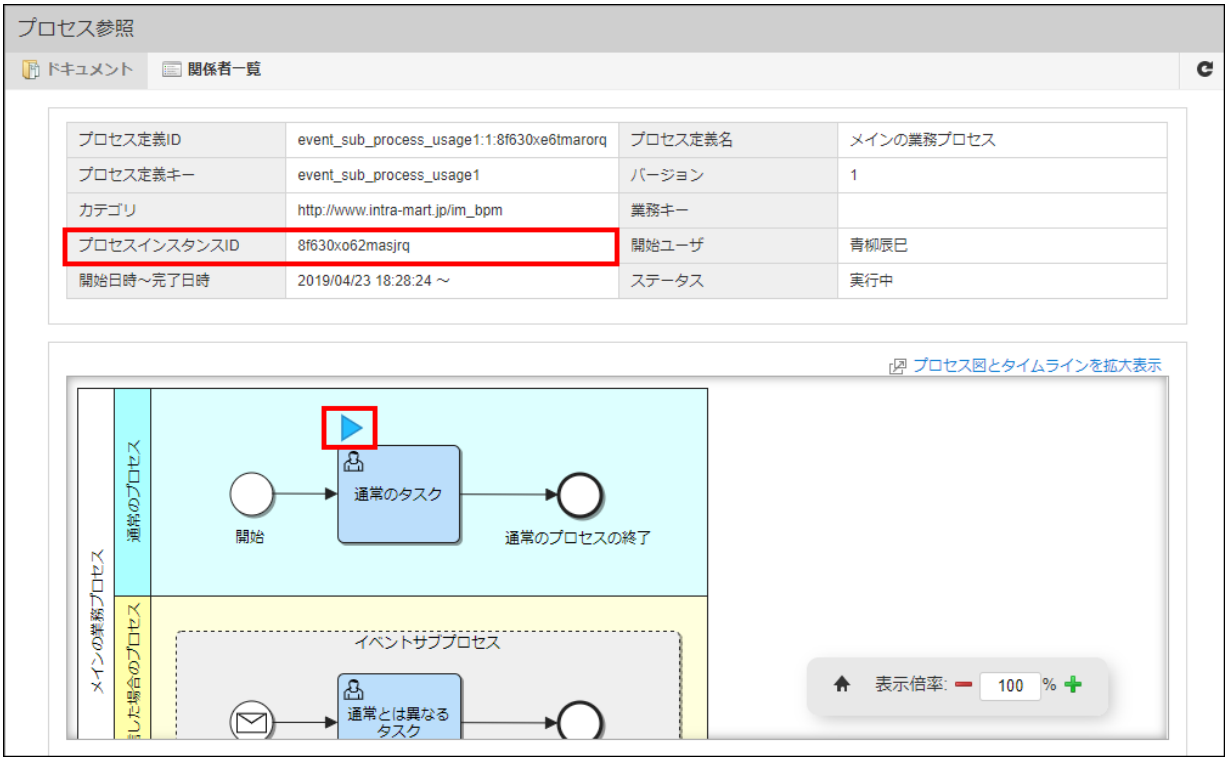
4. 「タスク一覧」画面から、下記の条件を満たすタスクの「 」ボタンをクリックし、「プロセス参照」画面を表示します。

- 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。



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。

図：「プロセス開始一覧」

7. Forma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。

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

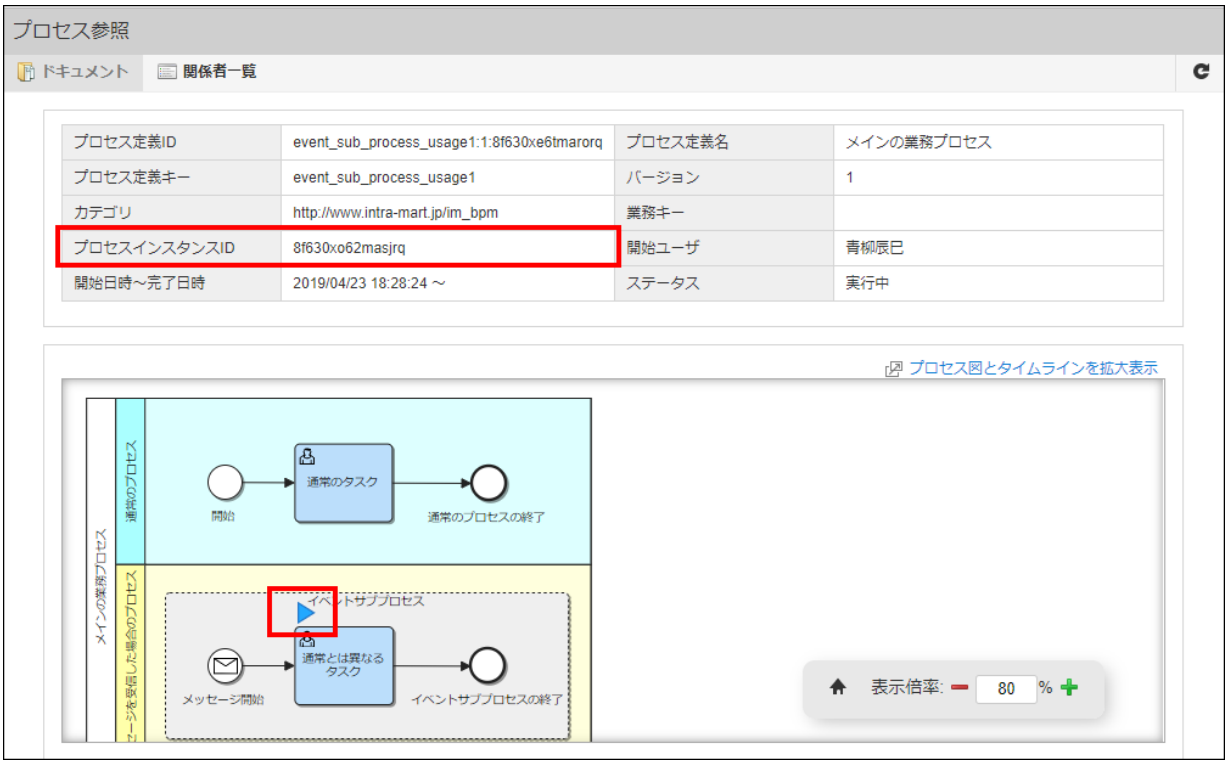
8. タスクの進行状態を確認します。

   「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

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。



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。

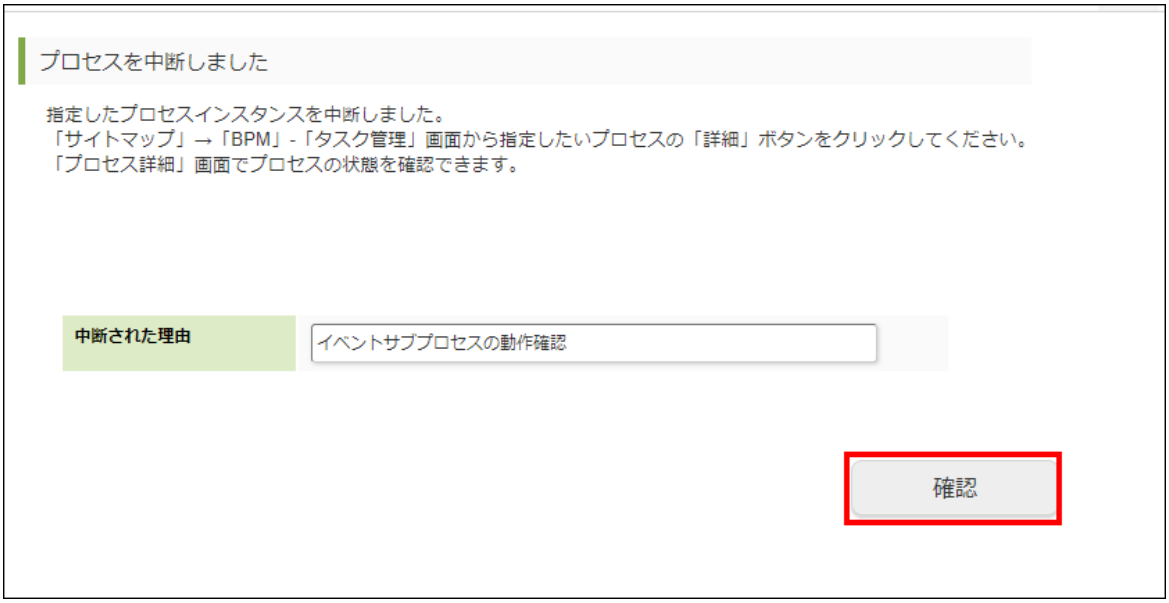
「タスク一覧」画面の「グループタスク」に振り分けられているタスクを、「 」ボタンをクリックします。

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。

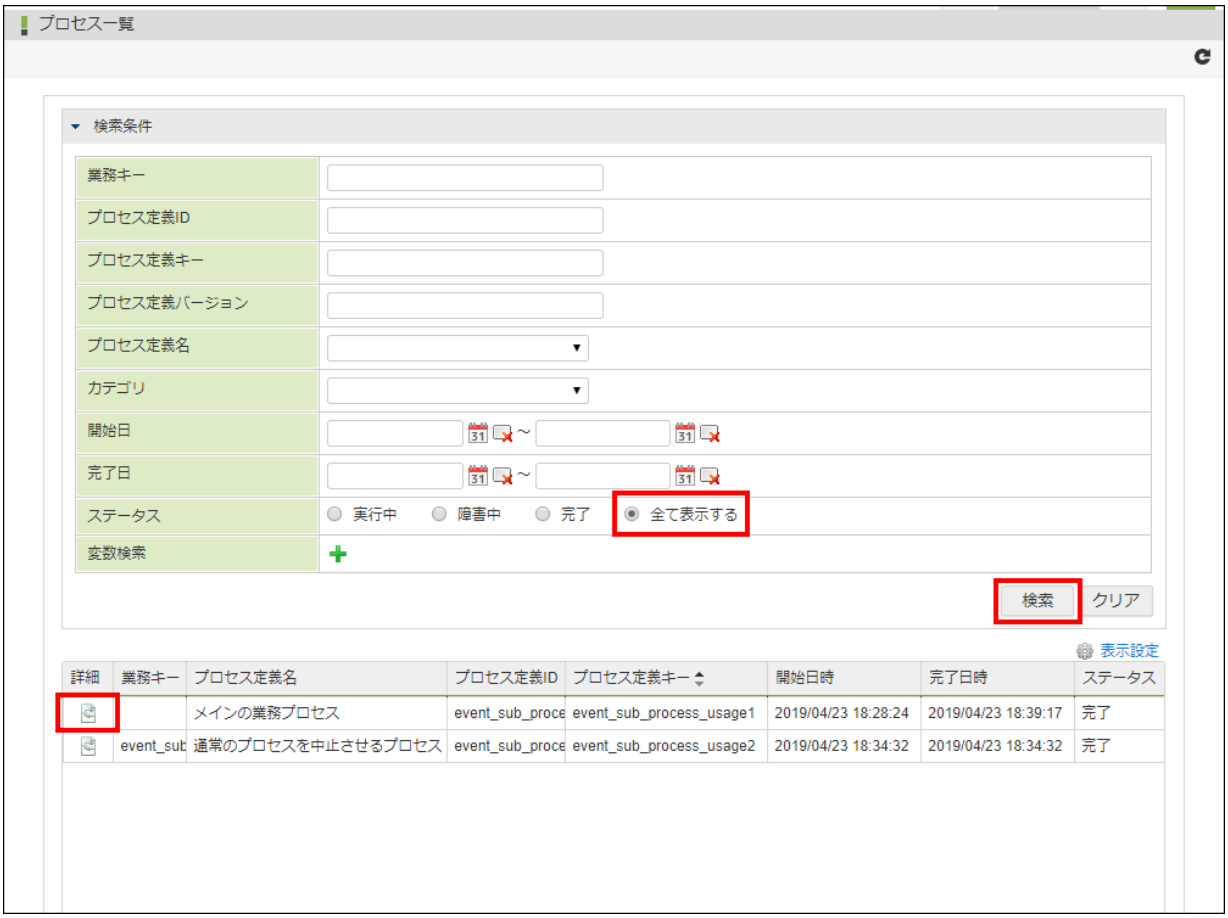
図：「プロセス詳細」

12. Forma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。



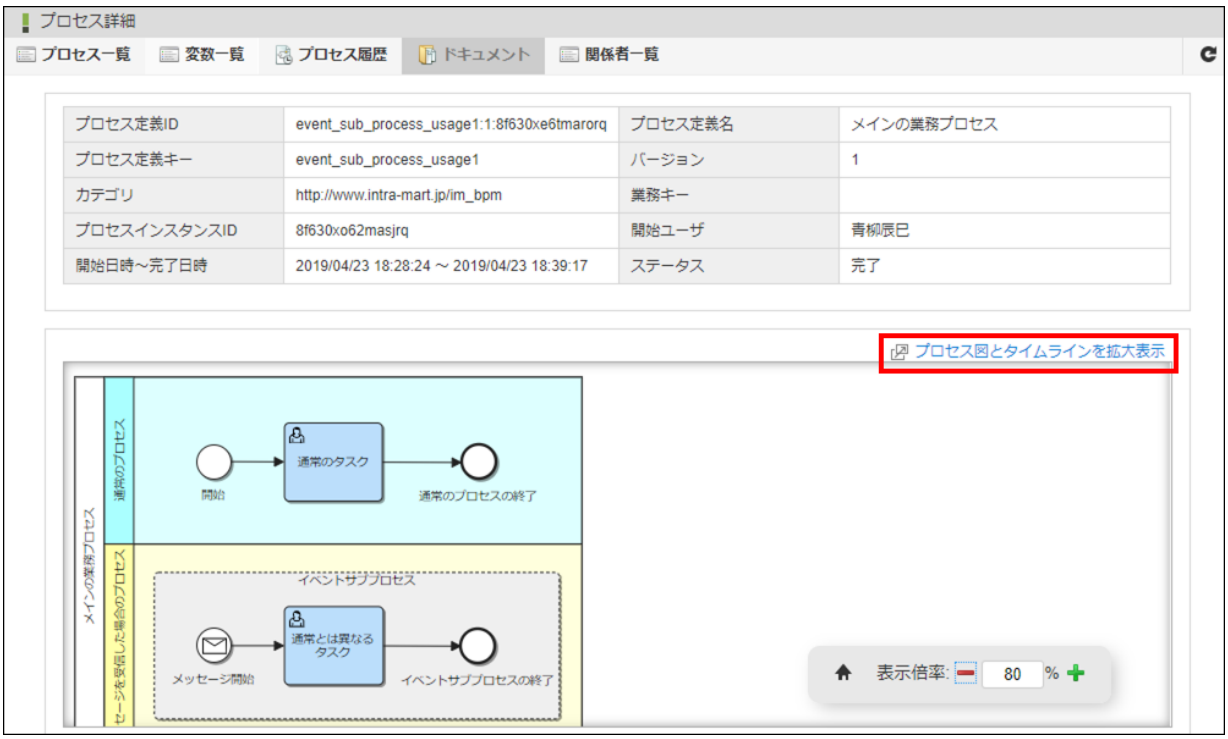
図：プロセスを中断しました（IM-FormaDesigner）

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。



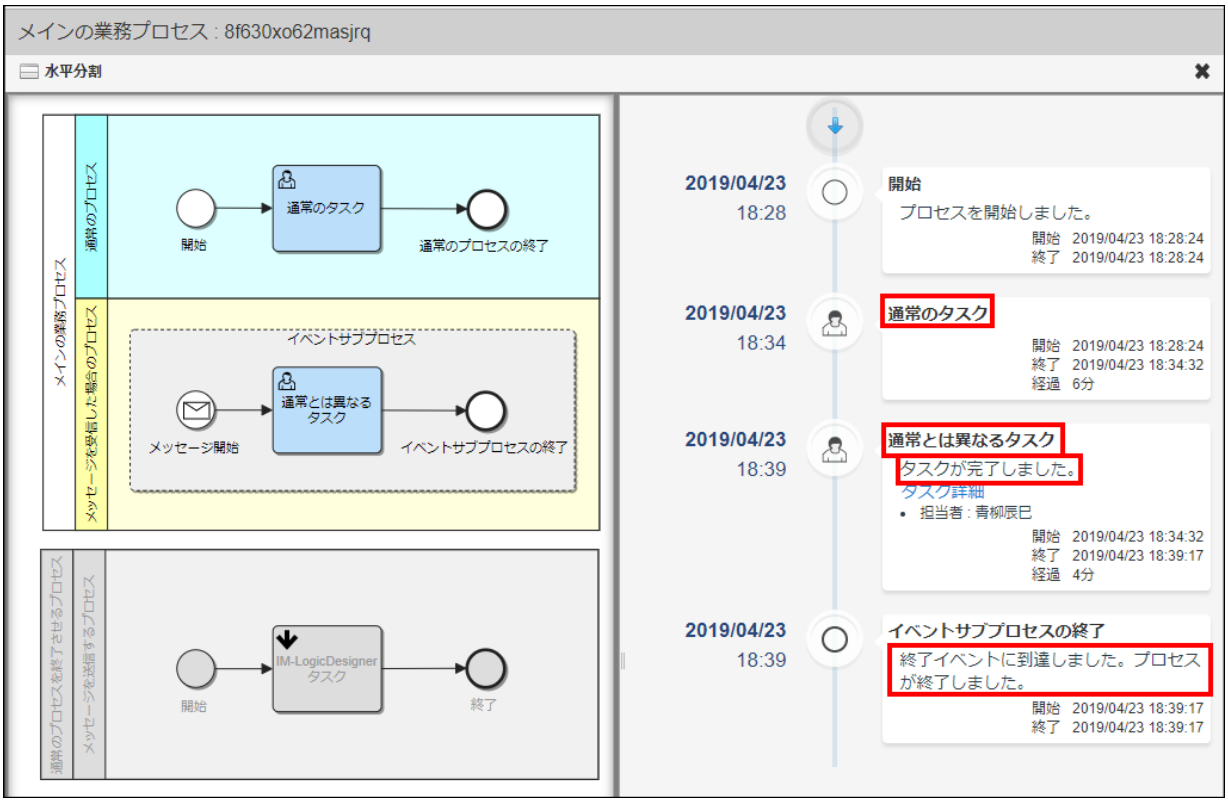
図：「プロセス詳細」

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」

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」

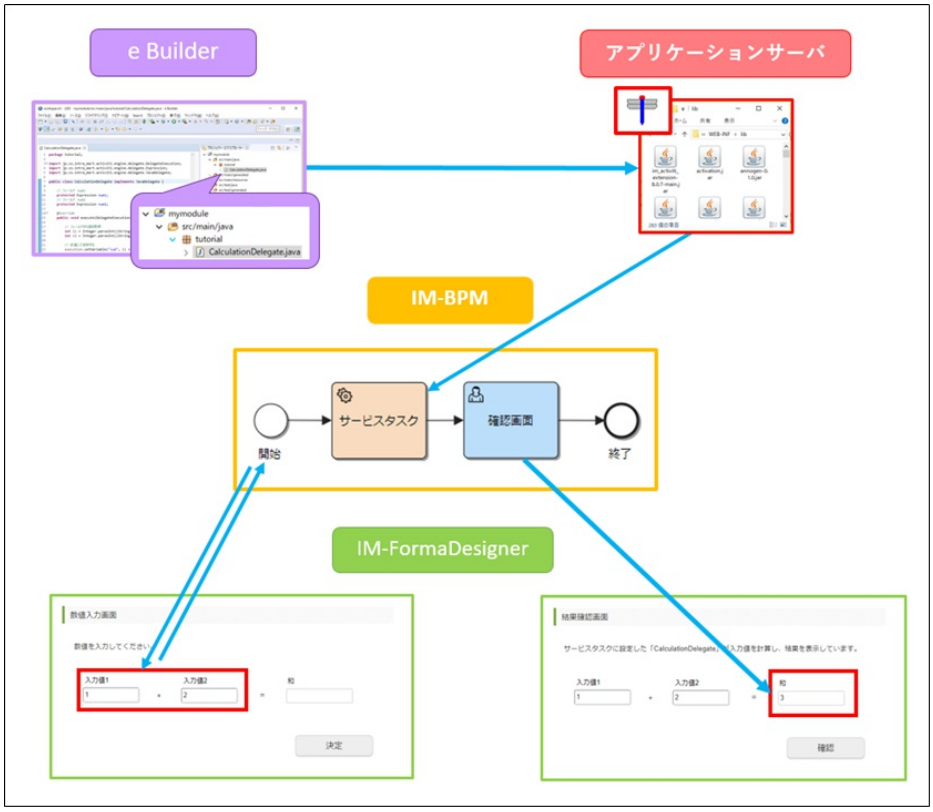
サービスタスク

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 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。
それぞれの手順のについては、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 操作ガイド」 -「プロセス定義のアップロード」

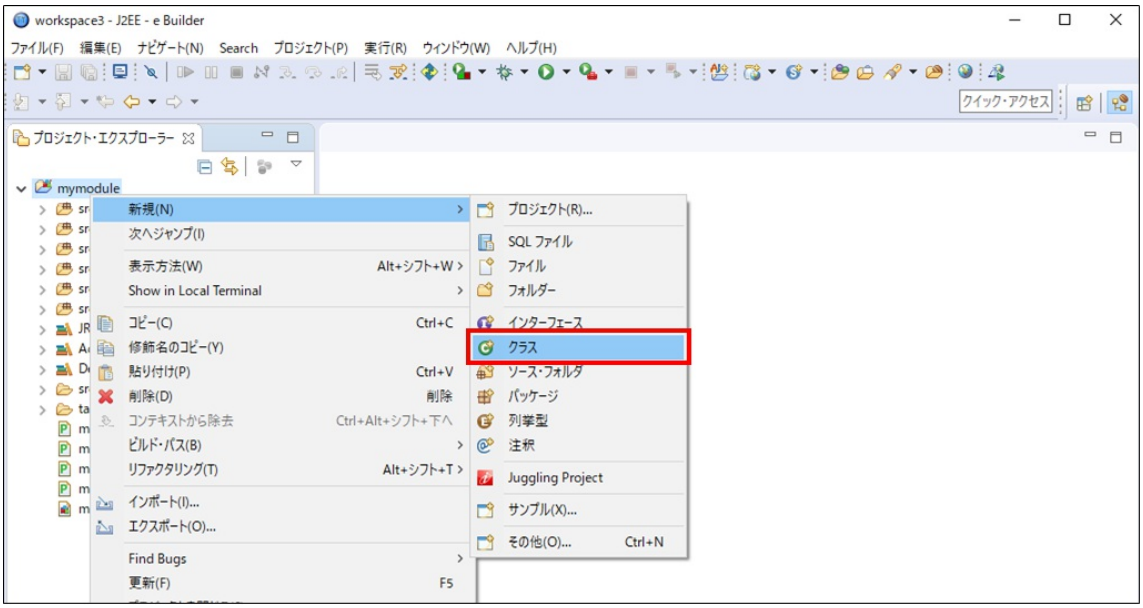
> **コラム**
>
> このチュートリアルで作成するモジュール・プロジェクトは、以下のリンクからダウンロードできます。
> **mymodule-1.0.0-service_task_usage.zip**
>
> 上記のモジュール・プロジェクトをエクスポートしたimmファイルは、以下のリンクからダウンロードできます。
> **mymodule-1.0.0-service_task_usage.imm**
>
> immファイルのインポート方法については、以下のリンクを参照してください。
>
> - immファイル：「intra-mart e Builder for Accel Platform 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

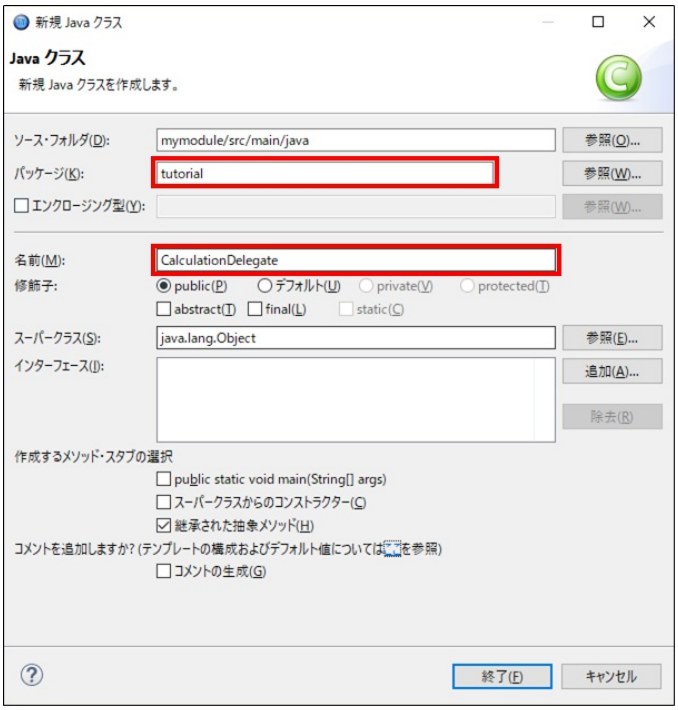
「サービスタスク」で実行したいJAVAクラスを「e Builder」で作成します。
今回は、「受け取った値を計算して設定する」という簡単なロジックを作成します。
「モジュール・プロジェクト」内で作成することにより、即時ローカル環境で実行可能です。

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。



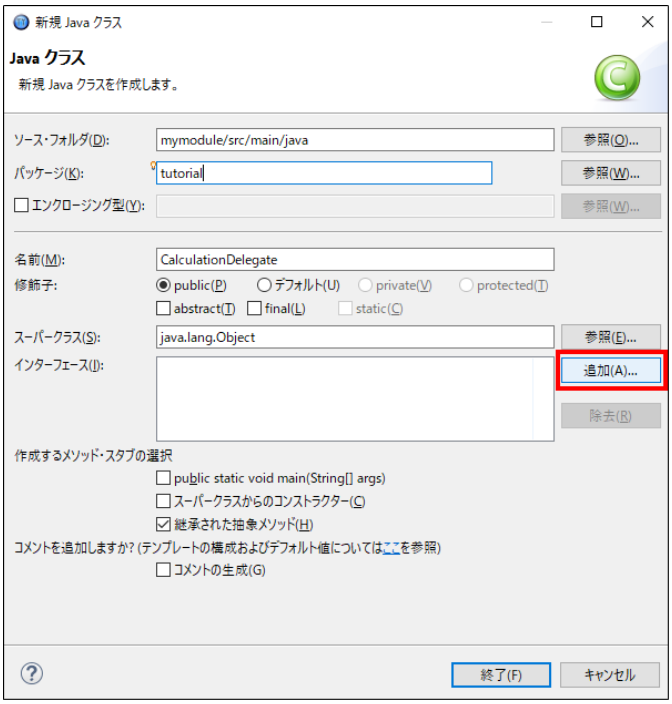
図：e Builder

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



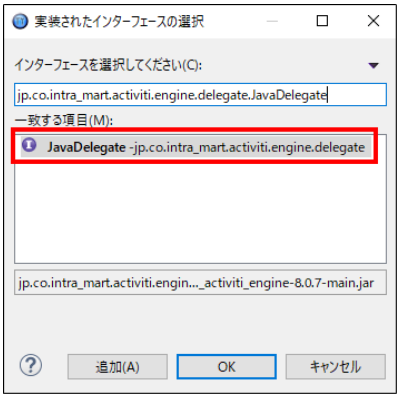
図：「新規Javaクラス」（e Builder）

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。



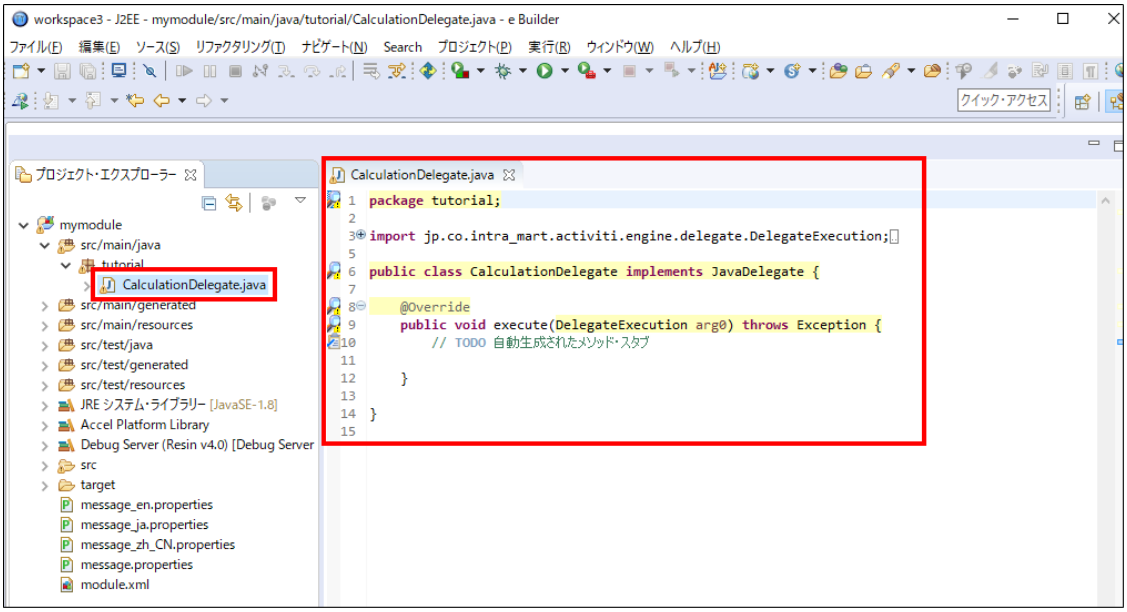
図：「新規Javaクラス」（e Builder）

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）

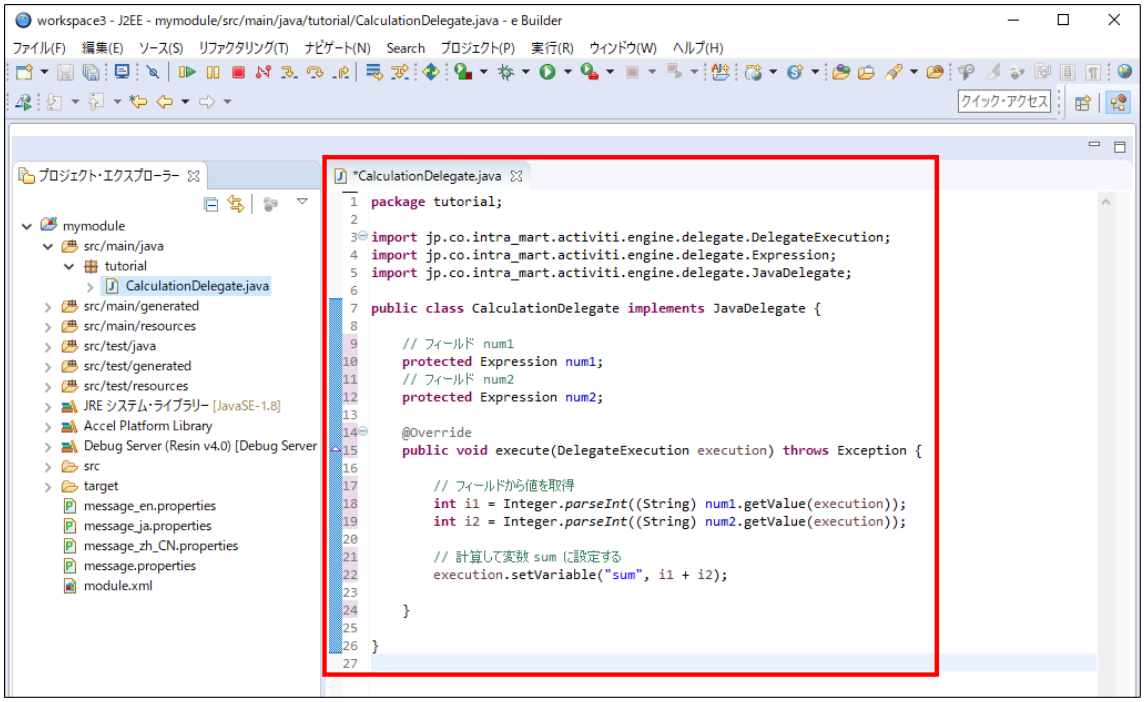
5. 「JavaDelegate」が実装された「CalculationDelegate.java」が作成されます。



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);

   }

}
```

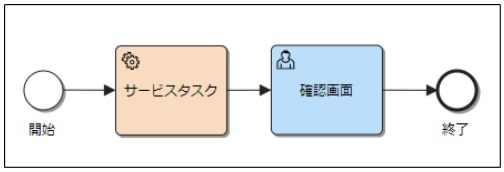


図：「CalculationDelegate」クラス（e Builder）

> **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

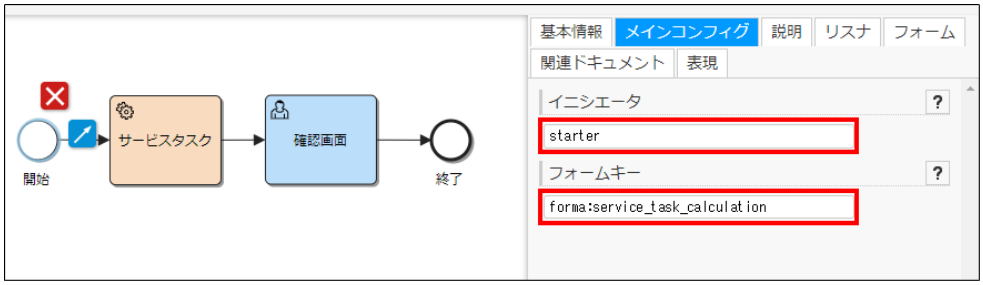
以下の図は、入力された値を計算して結果を表示するプロセスです。
「開始イベント」に設定した「IM-FormaDesigner」のアプリケーション画面で入力された値を「サービスタスク」で計算し、アプリケーション画面で表示します。



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



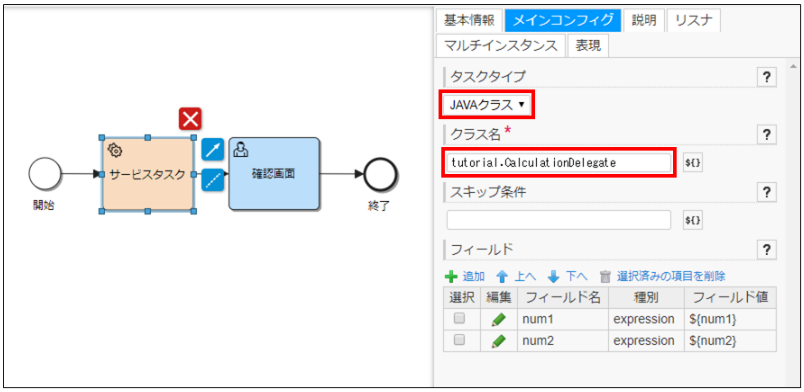
   図：「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

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。

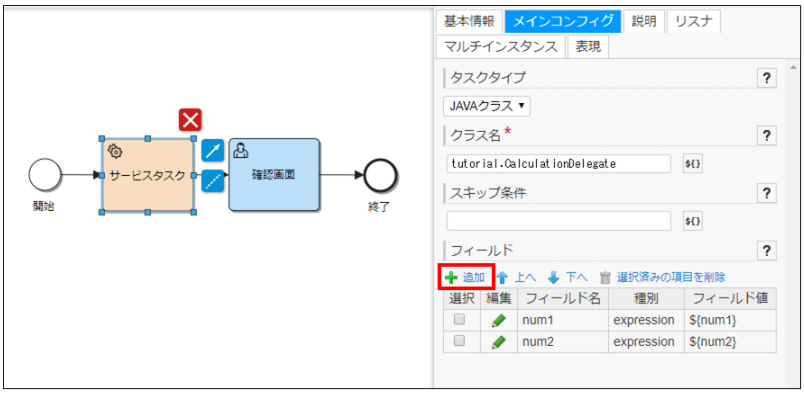
   図：プロセス全体 - 「プロパティ」 - 「プロセス」

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



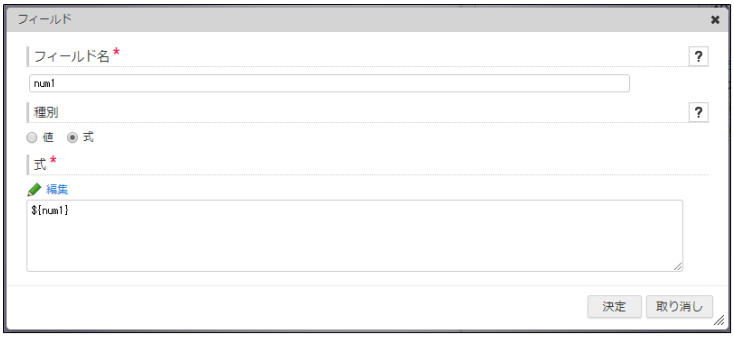
   図：「サービスタスク」- 「プロパティ」- 「メインコンフィグ」

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」

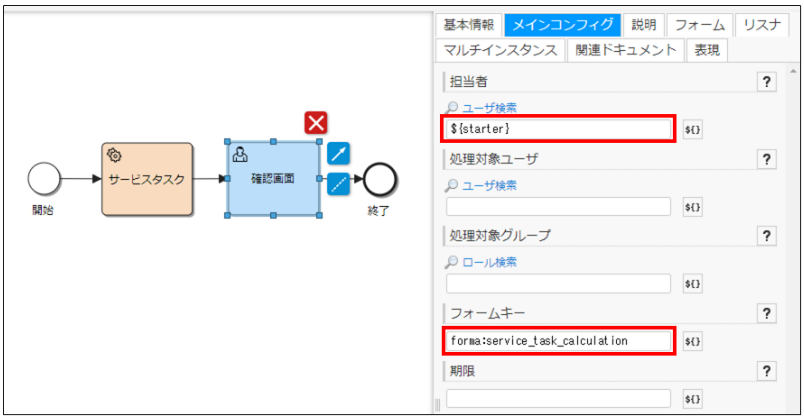
7. 以下のように項目を設定します。
   - 入力値1
     - フィールド名：num1
     - 種別：式
     - 式：${num1}
   - 入力値2
     - フィールド名：num2
     - 種別：式
     - 式：${num2}



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



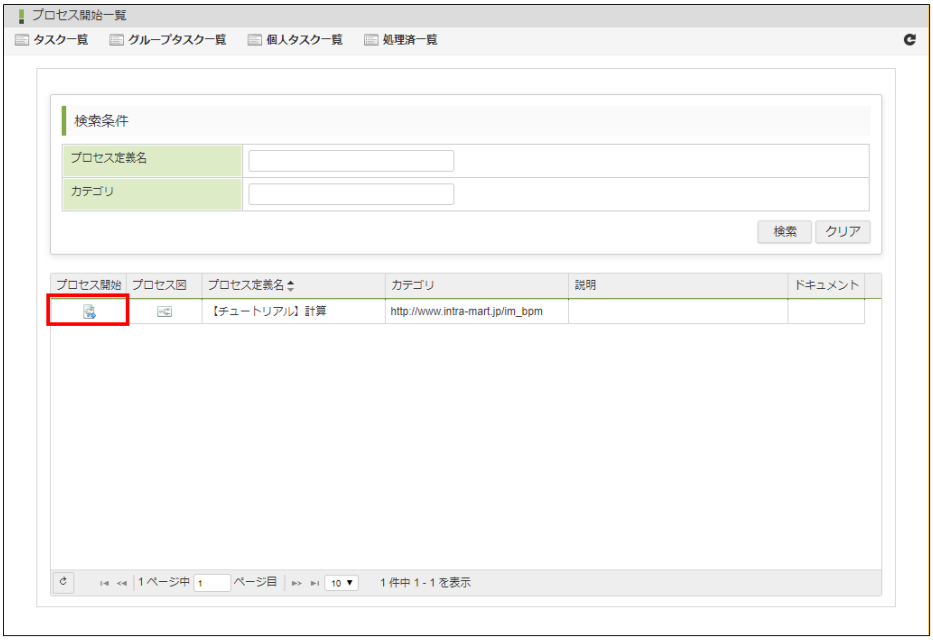
図：「確認画面」-「プロパティ」-「メインコンフィグ」

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。

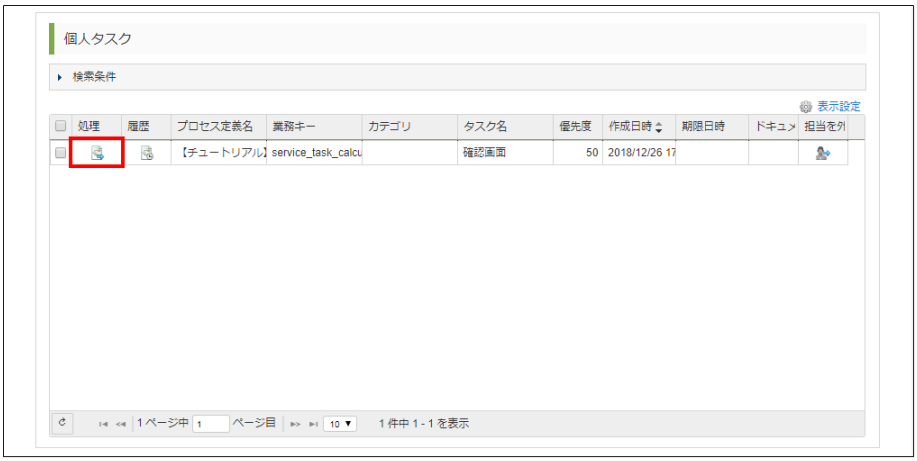


図：「プロセス開始一覧」

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。

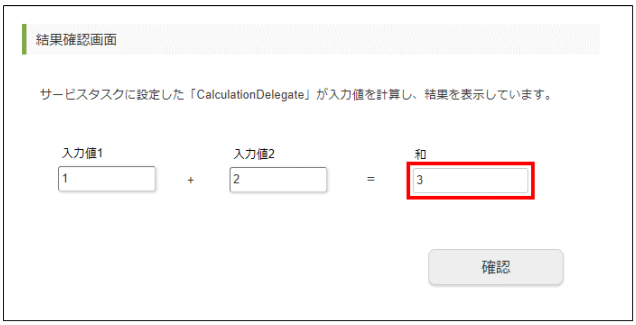
図：数値入力（IM-FormaDesigner）

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。



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。



図：結果確認（IM-FormaDesigner）

受信タスク

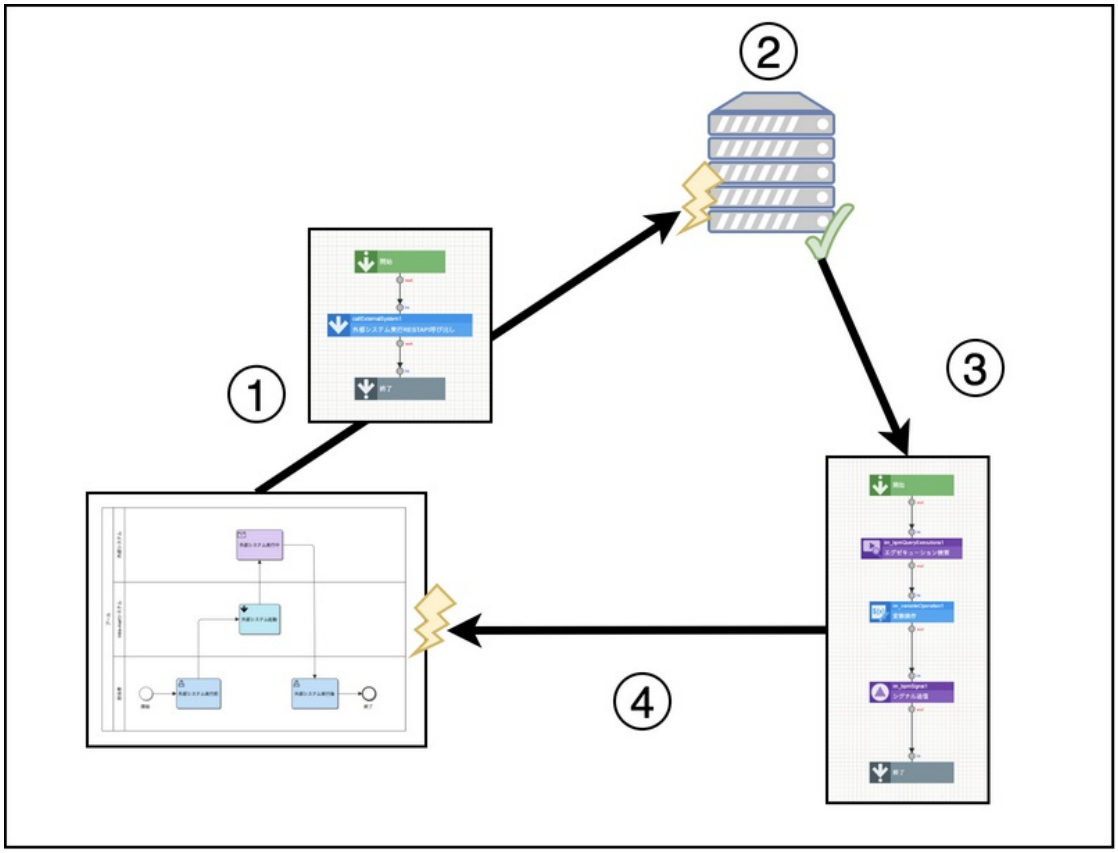
外部システムで実行されるタスクを受信タスクを用いて表現する

このチュートリアルでは、受信タスクを用いてIM-BPMのプロセスの中に外部のシステムで実行されるタスクを表現する方法を解説します。
外部システムで実行されるタスクの完了を待ち、完了後に後続のタスクへ実行状態が移ります。

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

本チュートリアルで作成するシステムの流れは、以下です。

1. 外部システムを実行したいタイミングでIM-LogicDesignerタスクから外部システムを実行します。
2. 外部システムの実行中はIM-BPMでは受信タスクに実行中マークが付与されています。
3. 外部システム完了のタイミングで、外部システムからintra-mart Accel Platformに対してIM-LogicDesignerで作成されたREST APIを呼び出します。
4. 呼び出されたロジックフローで対象のエグゼキューションを検索し、シグナルを送信することで受信タスクを完了します。

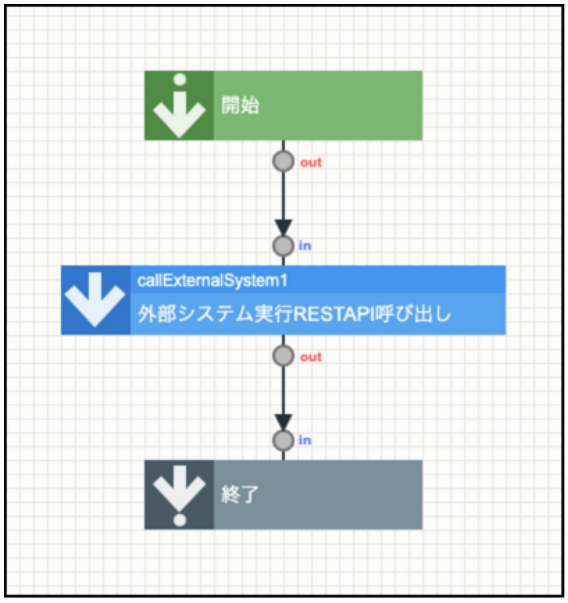


図：概要

> **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。
>
> - **プロセス定義**
>   - **receive-task-usage.bpmn**
> - **ロジックフロー**
>   - **im_logicdesigner-data-receive_task_usage.zip**
> - **IM-Authz（認可） ポリシー - XML形式定義情報**
>   - **authz-policy-receive_task_usage-LD.xml**
> - **それぞれの資材のアップロード方法については、以下を参照してください。**
>   - **プロセス定義:**
>   - アップロード方法については「IM-BPM プロセスデザイナ 操作ガイド」 - 「プロセス定義のアップロード」を参照してください。
>   - **ロジックフロー:**
>   - インポート方法については「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-Authz（認可） ポリシー - XML形式:**
>   - インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を参照してください。
>   - パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
>   - ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
>   または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

外部システムを実行するロジックフローを作成する

「IM-LogicDesigner」を使用して、IM-BPMと外部システムを繋ぐ部分のロジックフローを作成します。



図：完成イメージ（ロジックフロー）

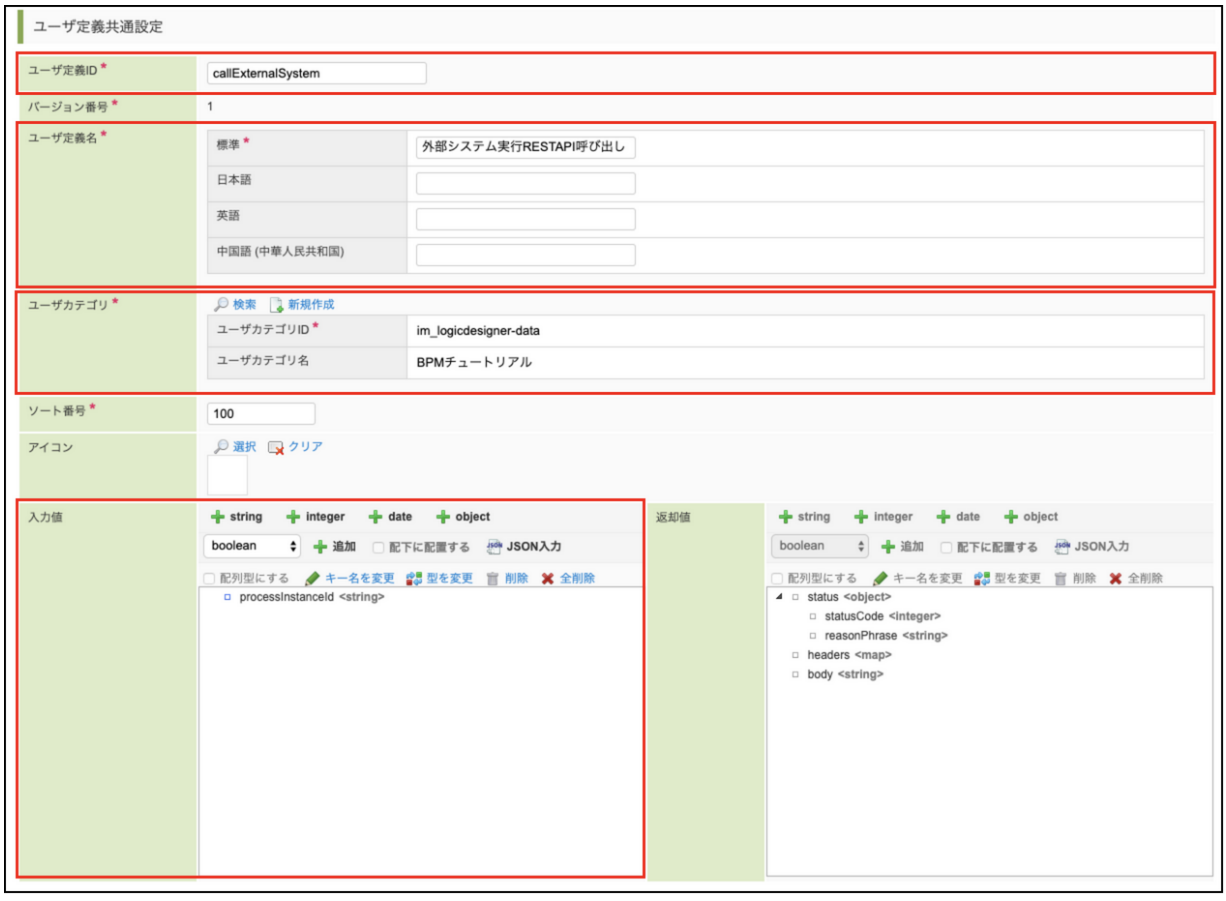
1. 外部システムを開始するためのREST APIを実行するための「IM-LogicDesigner」の「REST定義」を作成します。
   「サイトマップ」→「IM-LogicDesigner」→「ユーザ定義」→「REST定義新規作成」をクリックします。

2. 「ユーザ定義共通設定」項目をそれぞれ以下のように入力します
   - ユーザ定義ID: callExternalSystem
   - ユーザ定義名: 外部システム実行REST API呼び出し
   - ユーザカテゴリ:

入力項目名	入力内容
ユーザカテゴリID	im_logicdesigner-data
ユーザカテゴリ名	BPMチュートリアル

   - 入力値:

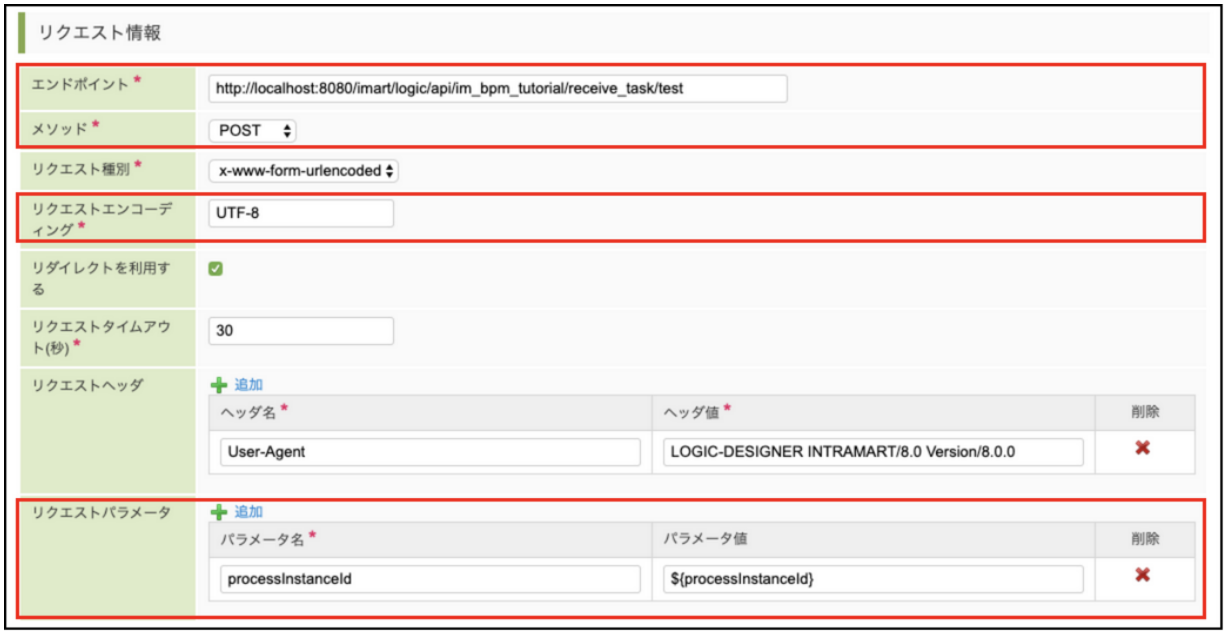
キー名	型
processInstanceId	<string>



図：「REST定義新規作成」-「ユーザ定義共通設定」

3. 「リクエスト情報」項目をそれぞれ以下のように入力します。
   - エンドポイント: <ベースURL>/logic/api/im_bpm_tutorial/receive_task/test
   - メソッド: POST
   - リクエストエンコーディング: UTF-8
   - リクエストパラメータ:

パラメータ名	パラメータ値
processInstanceId	${processInstanceId}



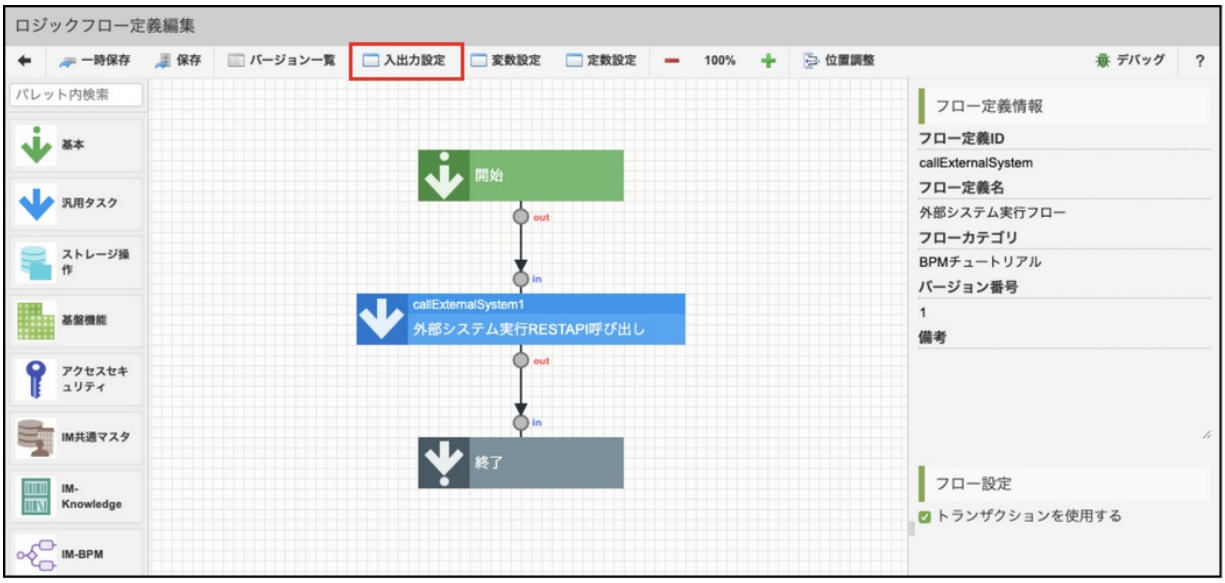
図：「REST定義新規作成」-「リクエスト情報」

> **コラム**
>
> このチュートリアルの上記項目は一例です。実際は利用されるシステムに合わせて入力内容を変更してください。
> このエンドポントを動作をさせるためには、以下の「IM-LogicDesigner」と「IM-Authz（認可） ポリシー - XML形式」の資材を追加でインポートする必要があります。
>
> - **ロジックフロー**
>   - **im_logicdesigner-data-receive_task_usage-extanal_system_dummy.zip**
> - **IM-Authz（認可） ポリシー - XML形式定義情報**
>   - **authz-policy-receive_task_usage-dummy_LD.xml**
> - **それぞれの資材のアップロード方法については、以下を参照してください。**
>   - **ロジックフロー:**
>   - インポート方法については「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-Authz（認可） ポリシー - XML形式:**
>   - インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を参照してください。
>   - パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
>   - ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
>   または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

4. 「登録」ボタンをクリックし、「REST定義」を保存します。

5. 作成した「REST定義」を呼び出すロジックフローを作成します。
   - 「サイトマップ」→「IM-LogicDesigner」→「フロー定義一覧」をクリックします。
   - 「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、「ロジックフロー定義編集」画面を表示します。

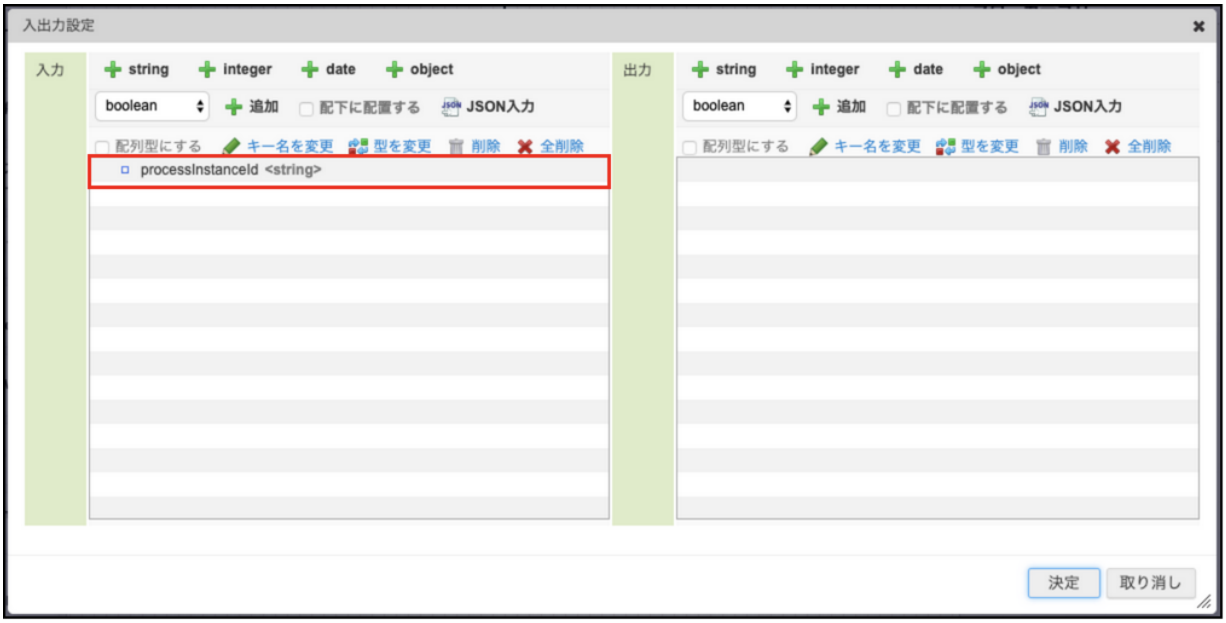
6. メニューバーの「入出力設定」をクリックし、「入力設定」を表示します。



図：「ロジックフロー定義編集」

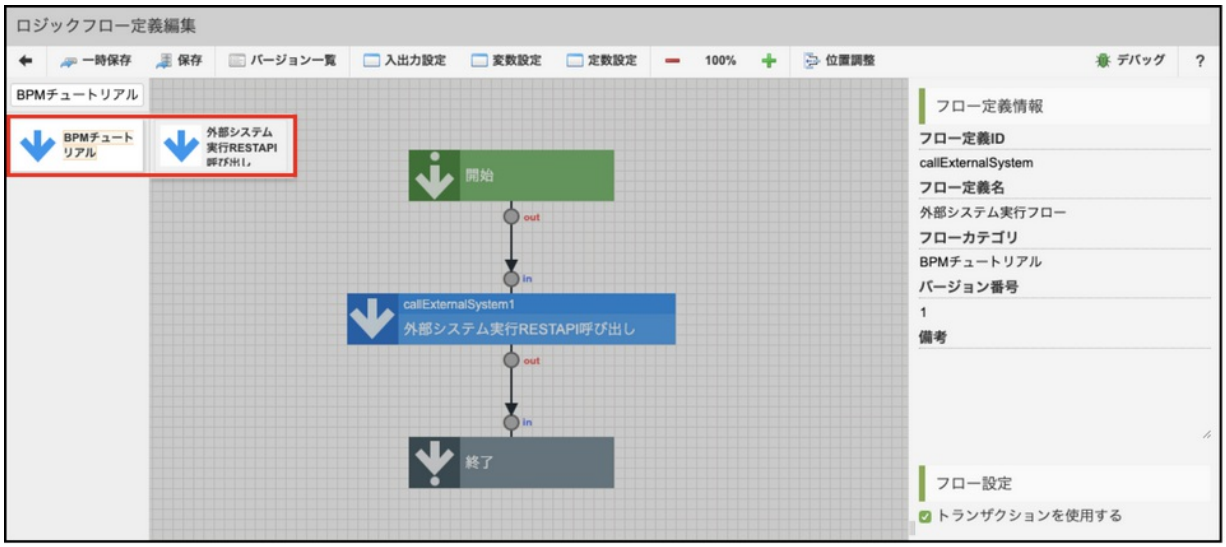
7. 「入力設定」を表示し、入力でプロセスインスタンスIDを受け取るように設定します。
   - 入力
     「IM-BPM」から渡ってきた、外部システムへ連携するためのプロセスインスタンスID

キー名	型
processInstanceId	<string>



図：「入出力設定」

8. 作成したユーザ定義を配置します。
   - パレット内の「IM-BPMチュートリアル」の一覧から「外部システム実行REST API呼び出し」を選び、フロー編集画面上に追加します。



図：「ロジックフロー定義編集」

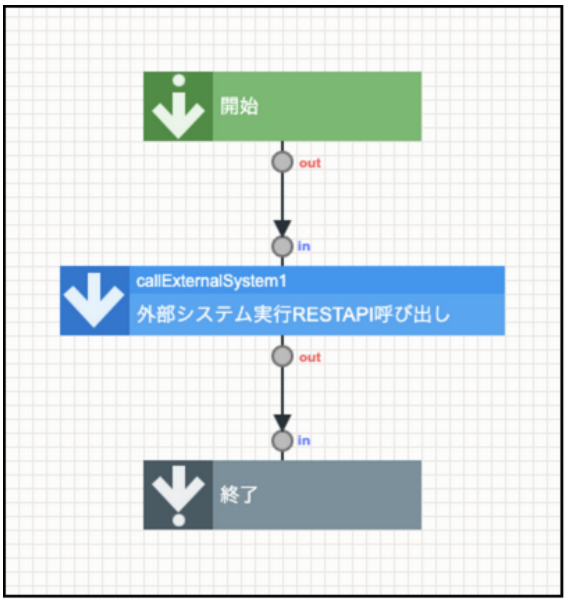
9. 「外部システム実行REST API呼び出し」のマッピング設定をします。
   - 以下のように線を繋ぎます。

入力（始点）	出力（終点）
入力<object> - processInstanceId	callExternalSystem1<object> - processInstanceId

図：「マッピング設定」 - 「外部システム実行REST API呼び出し」

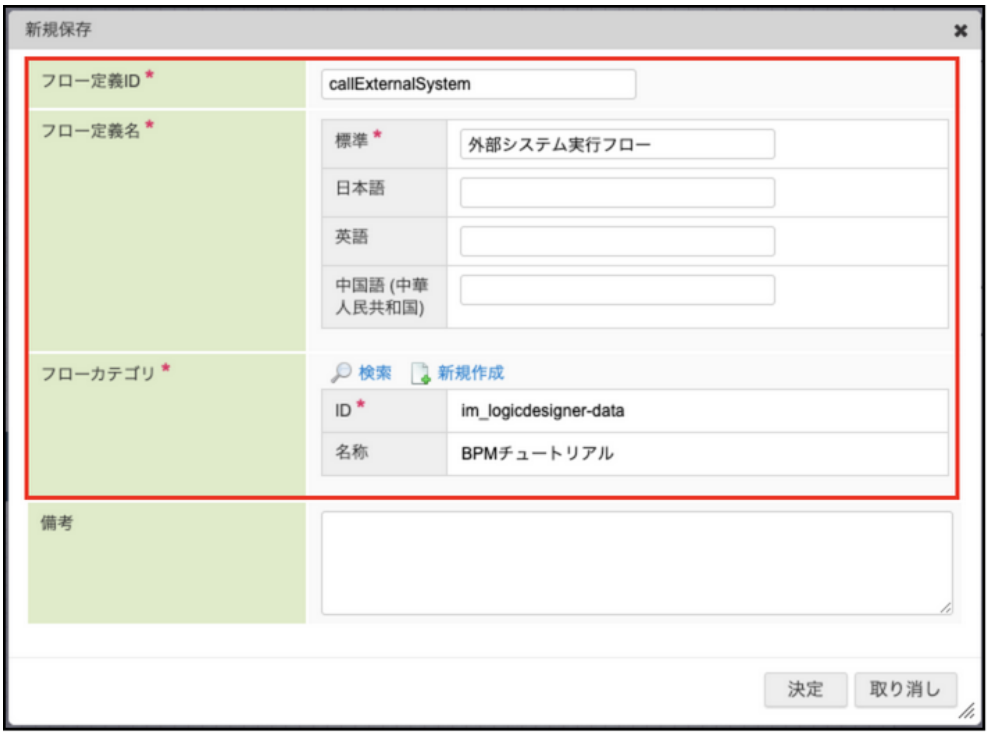
10. タスクを以下のようにつなぎます。

out	in
開始	外部システム実行REST API呼び出し
外部システム実行REST API呼び出し	終了



図：完成イメージ（ロジックフロー）

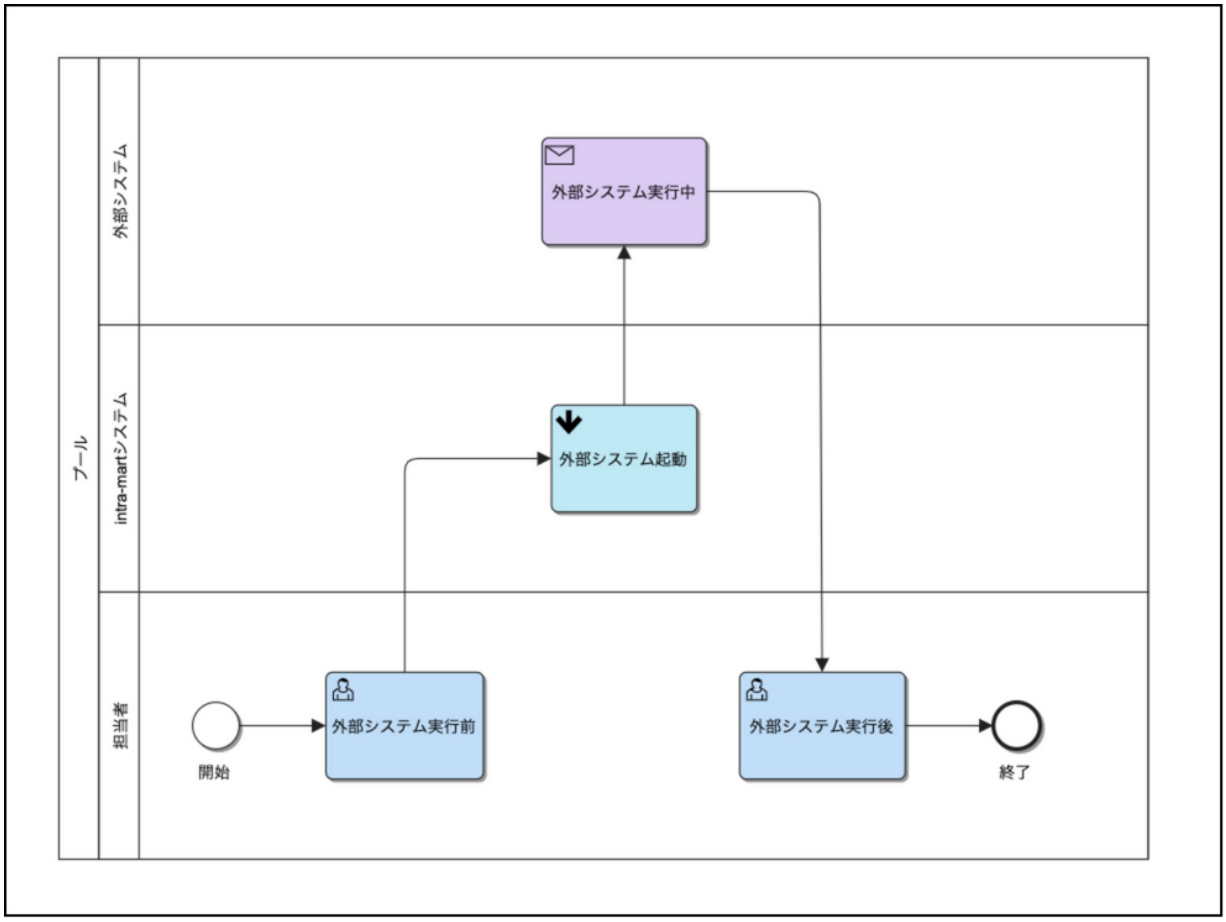
11. 「ロジックフロー定義編集」画面上部、ヘッダ内の「新規保存」をクリックします。
以下のように設定し、ロジックフロー定義を新規保存します。
    - フロー定義ID：callExternalSystem
    - フロー定義名：外部システム実行フロー
    - フローカテゴリ：
        - ID：im_logicdesigner-data
        - 名称：BPMチュートリアル



図：「新規保存」

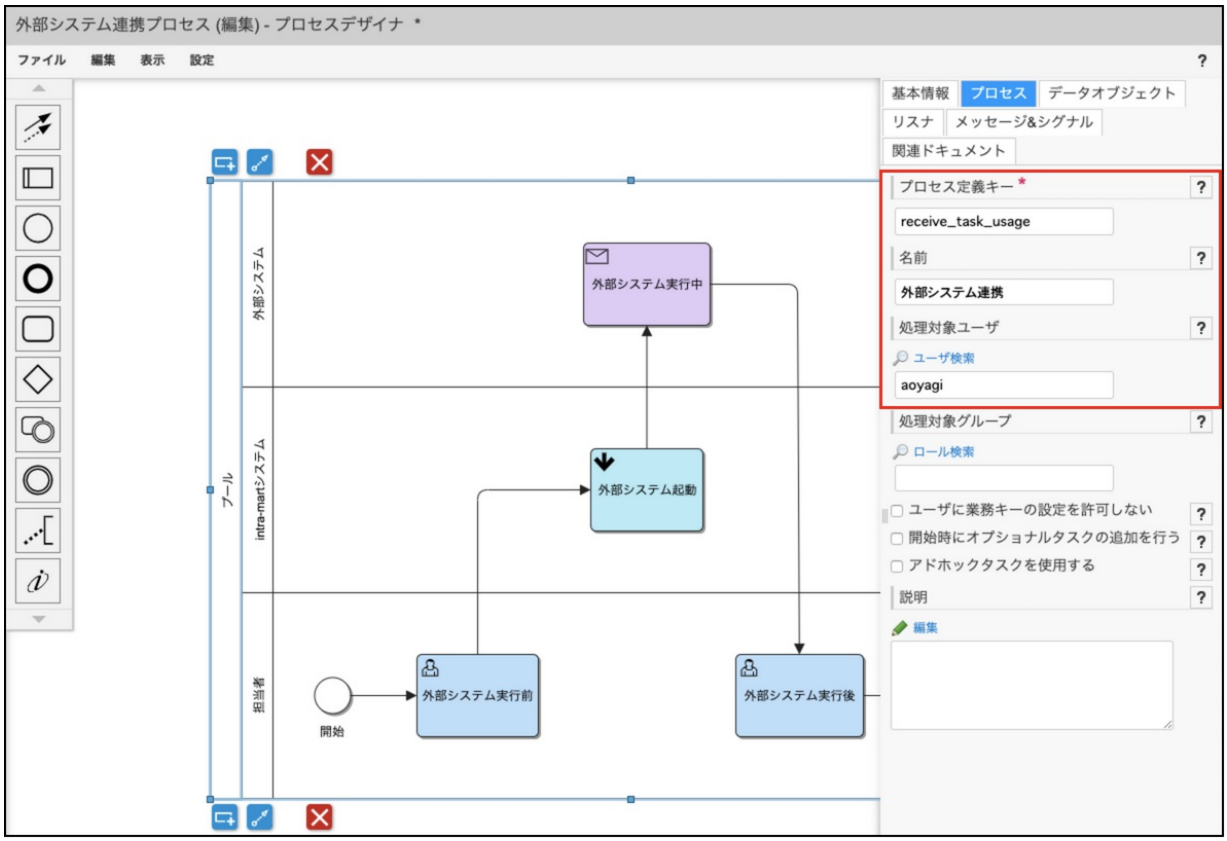
プロセス定義を作成する

上記のロジックフローを組み込んだプロセスを作成します。
このプロセスは、最初の「ユーザタスク」を処理すると「IM-LogicDesignerタスク」により外部システムが実行されます。
「受信タスク」で外部システムの完了のシグナルを待ちます。そのため、「受信タスク」が実行中となっている場合は外部システムが実行中であることが表現できます。
シグナルを受信した後、後続の「ユーザタスク」が実行されます。



図：プロセス定義図

1. 「プール」を設置します。
2. 「プール」に3つの「レーン」を設置し、上から順にそれぞれ以下のように名前をつけます。
   - 外部システム
   - intra-martシステム
   - 担当者
3. プロセス全体に対する設定を行います。
   「プール」を選択し、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：receive_task_usage
   - 名前：外部システム連携
   - 処理対象ユーザ：aoyagi

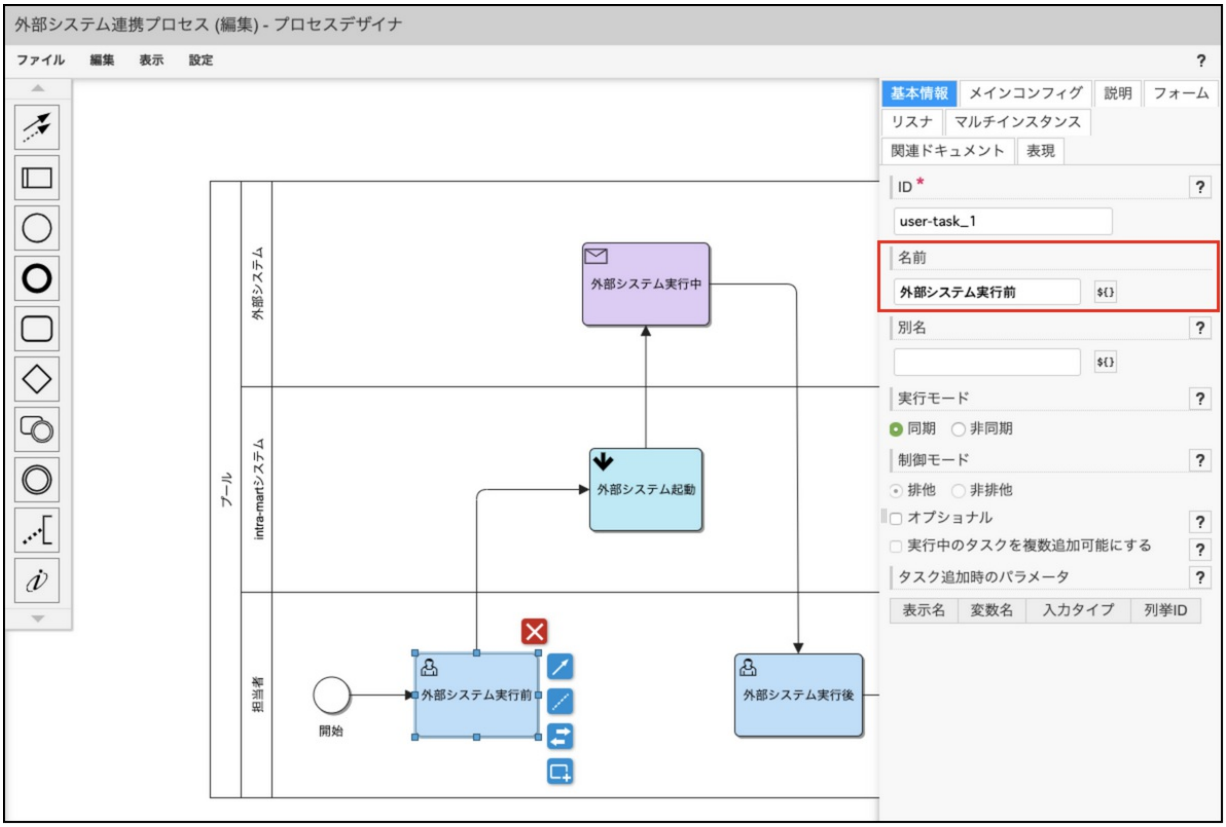


図：「プール」-「プロパティ」 -「プロセス」

4. 「開始イベント」を「担当者レーン」へ設置します。

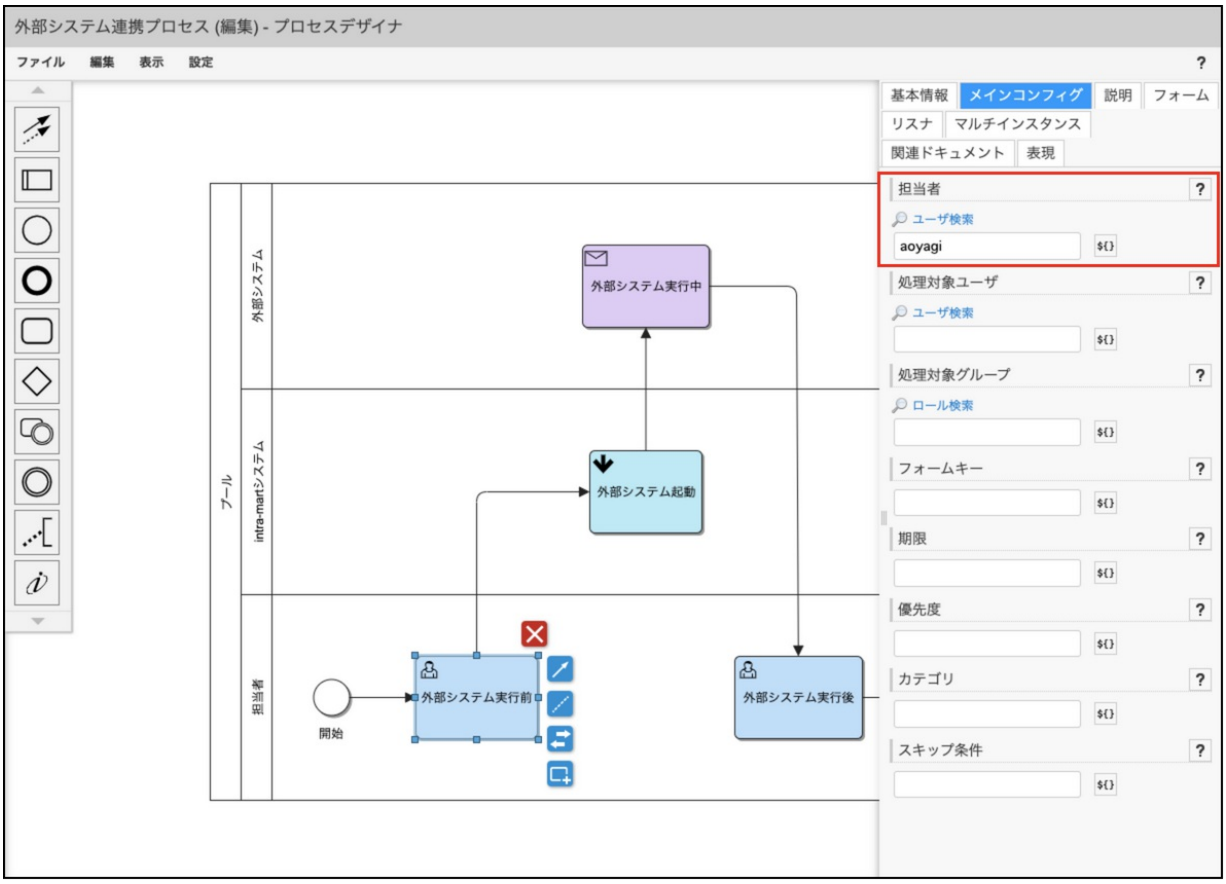
5. 外部システムを呼び出す前の状態を確認するために、「担当者レーン」に「ユーザタスク」を設置します。

6. 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 `外部システム実行前` と設定します。



図：「ユーザタスク」-「プロパティ」 -「基本情報」

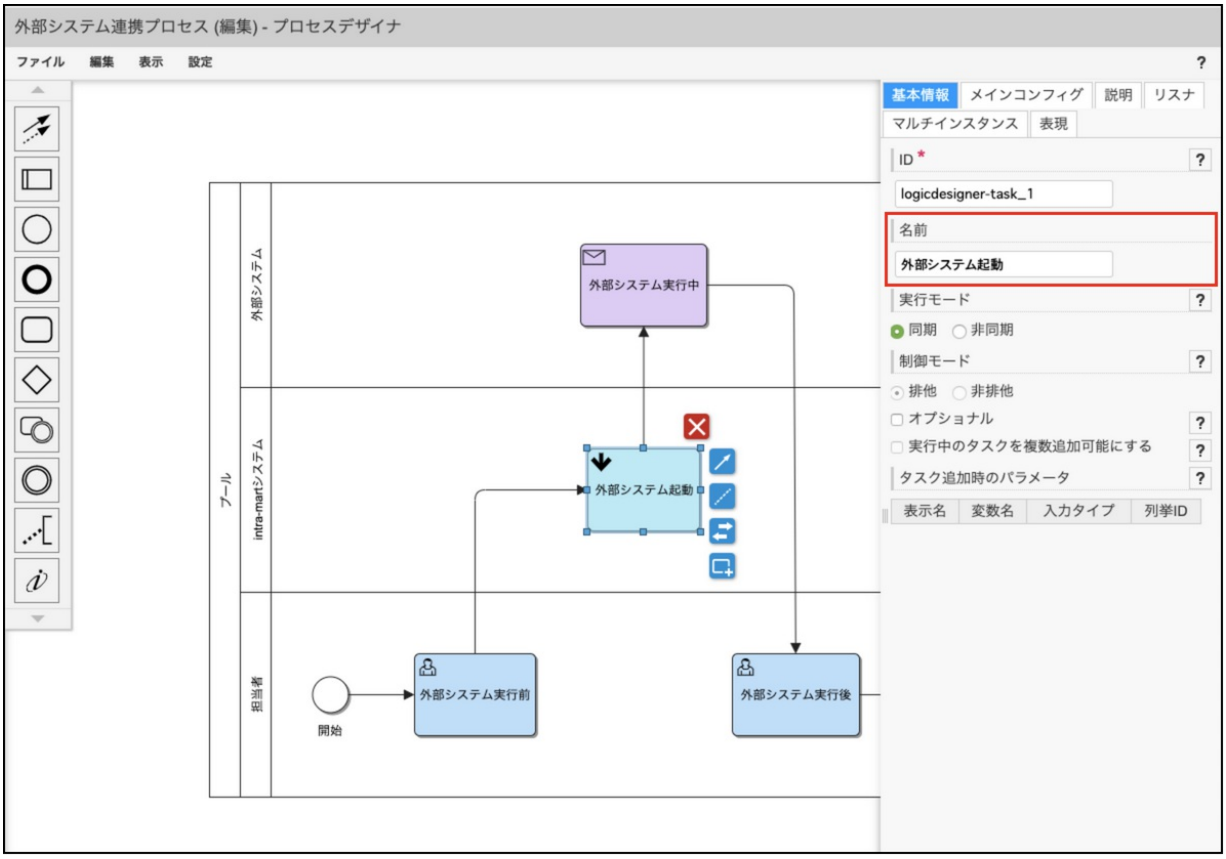
7. 「外部システム実行前」タスクに対して、担当者を設定します。
   「外部システム実行前」タスクを選択し「メインコンフィグ」タブから「担当者」に `aoyagi` と設定します。



図：「ユーザタスク」-「プロパティ」 -「メインコンフィグ」

8. 外部システムを実行するために「IM-LogicDesigner」を呼び出します。
「IM-LogicDesignerタスク」を選択し、「intra-martシステムレーン」へ設置します。

9. 設置した「IM-LogicDesignerタスク」に名前をつけます。
「基本情報」タブから、「名前」に　外部システム起動 と設定します。

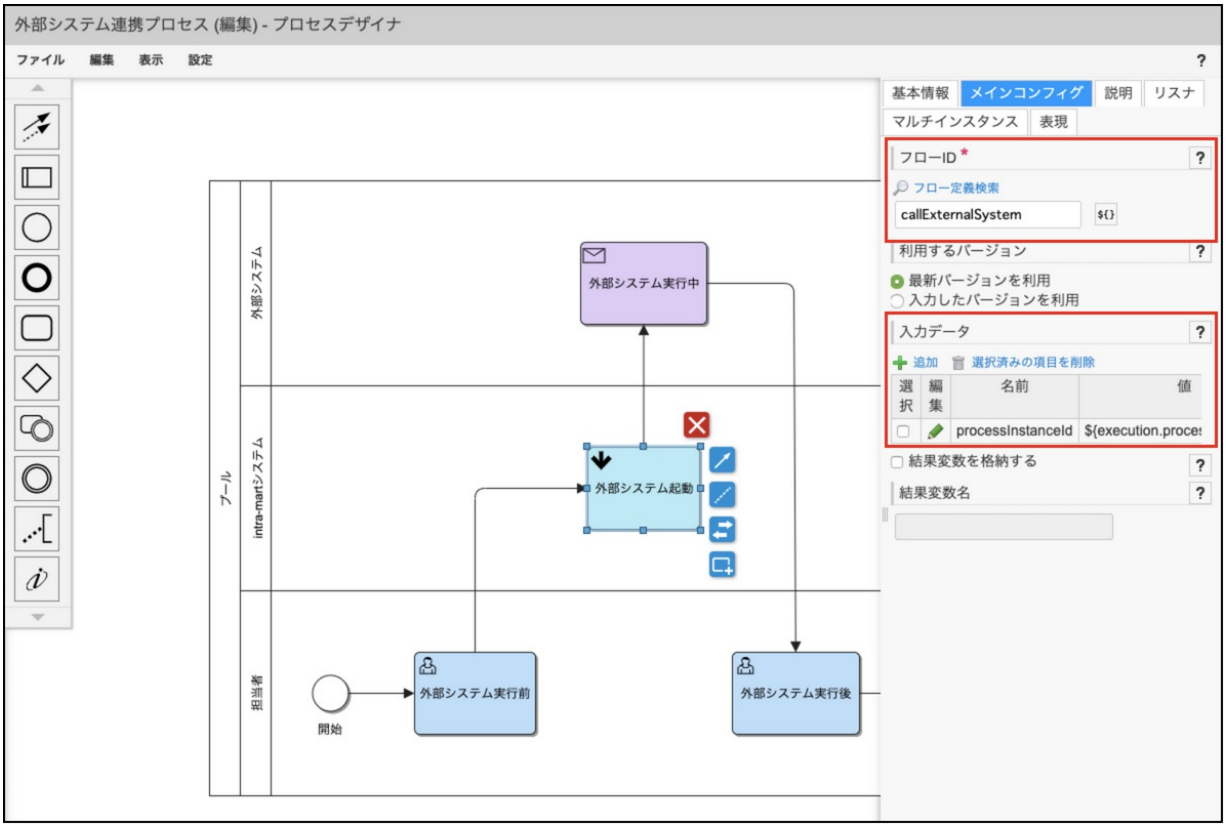


図：「IM-LogicDesignerタスク」-「プロパティ」 -「基本情報」

10. 呼び出すロジックフローの設定とロジックフローの入力値に渡すデータの設定をします。

「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。

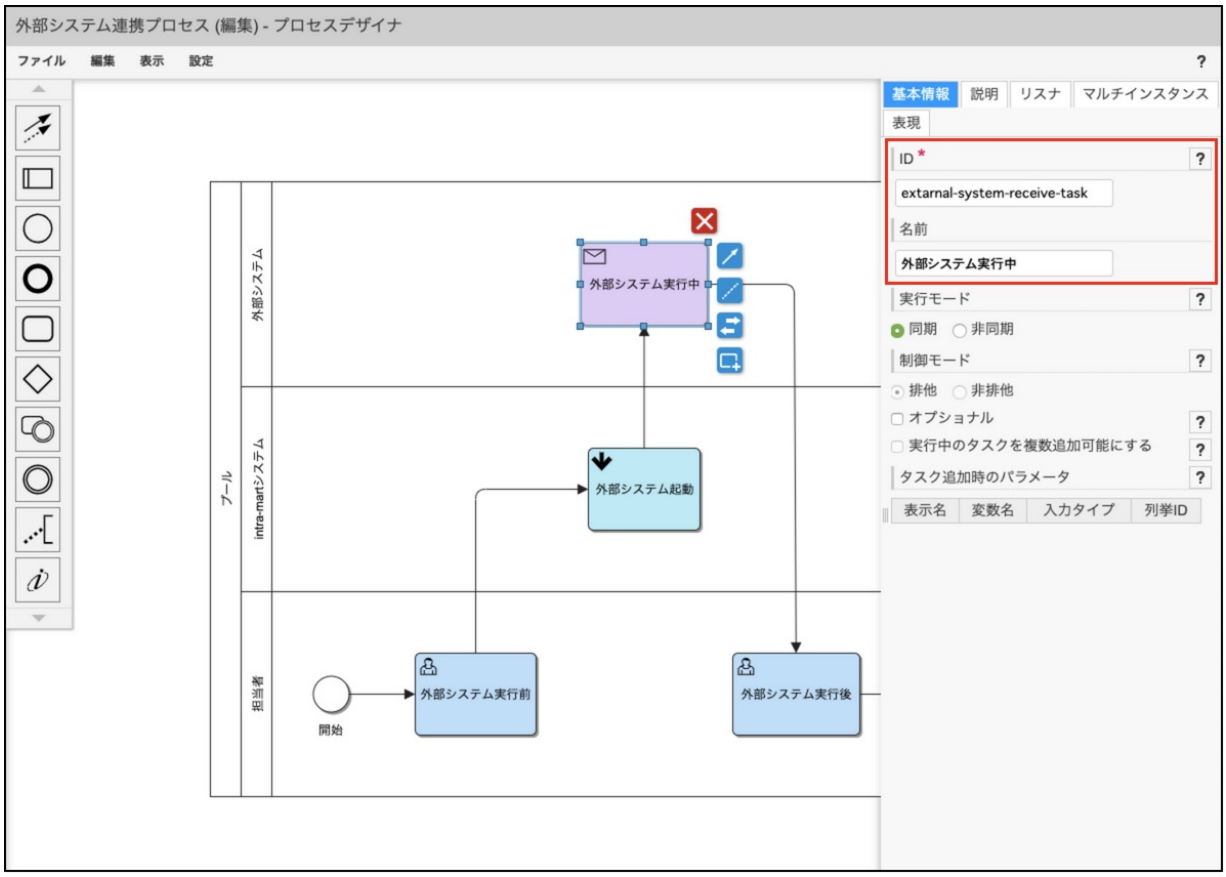
- フローID: callExternalSystem
- 利用するバージョン: 最新バージョンを利用
- 入力データ
  - 名前: processInstanceId
  - 値: ${execution.processInstanceId}



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

11. 外部システムの完了シグナルを受信するための「受信タスク」を「外部システムレーン」へ設置します。

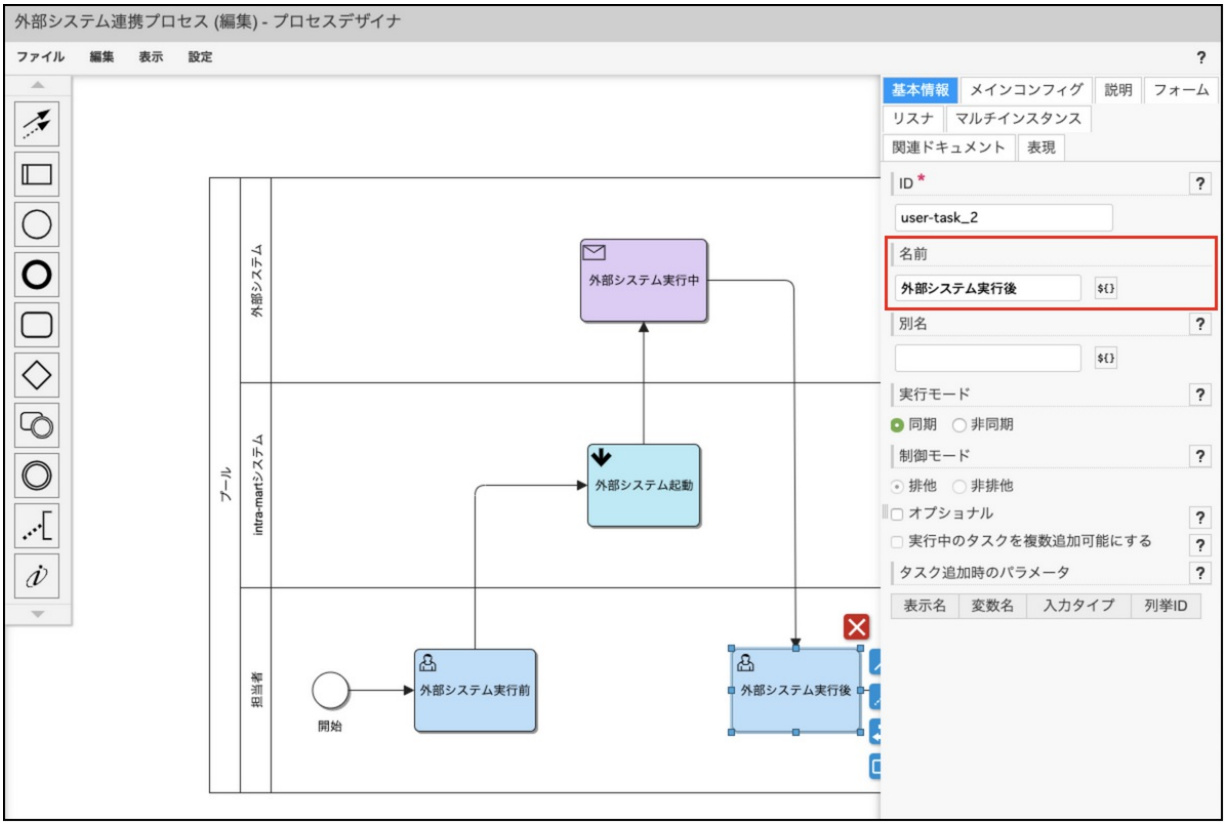
12. 設置した「受信タスク」にIDと名前をつけます。
「基本情報」タブで以下のように設定をします。
    - ID: external-system-receive-task
    - 名前: 外部システム実行中



図：「受信タスク」-「プロパティ」 -「基本情報」

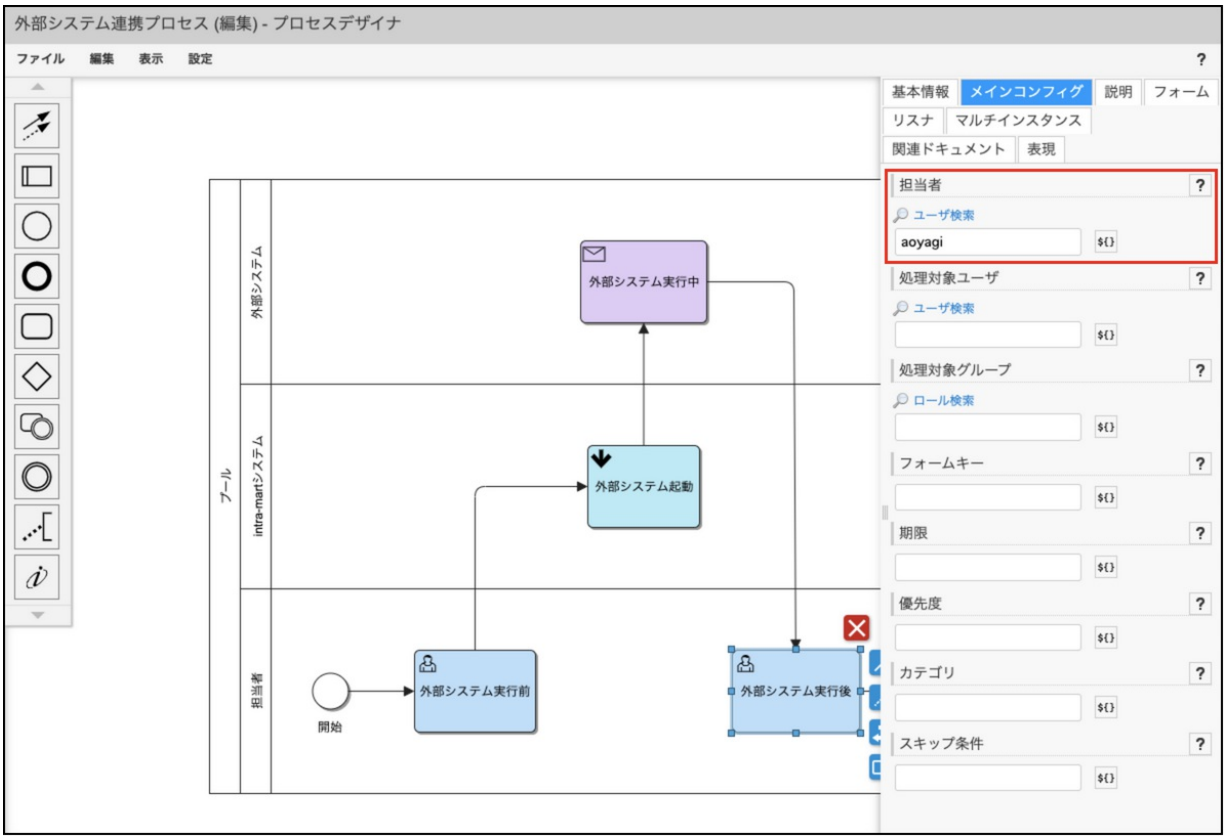
13. 外部システム完了後の状態を確認するために、「担当者レーン」に「ユーザタスク」を設置します。

14. 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　外部システム実行後 と設定します。



図：「ユーザタスク」-「プロパティ」 -「基本情報」

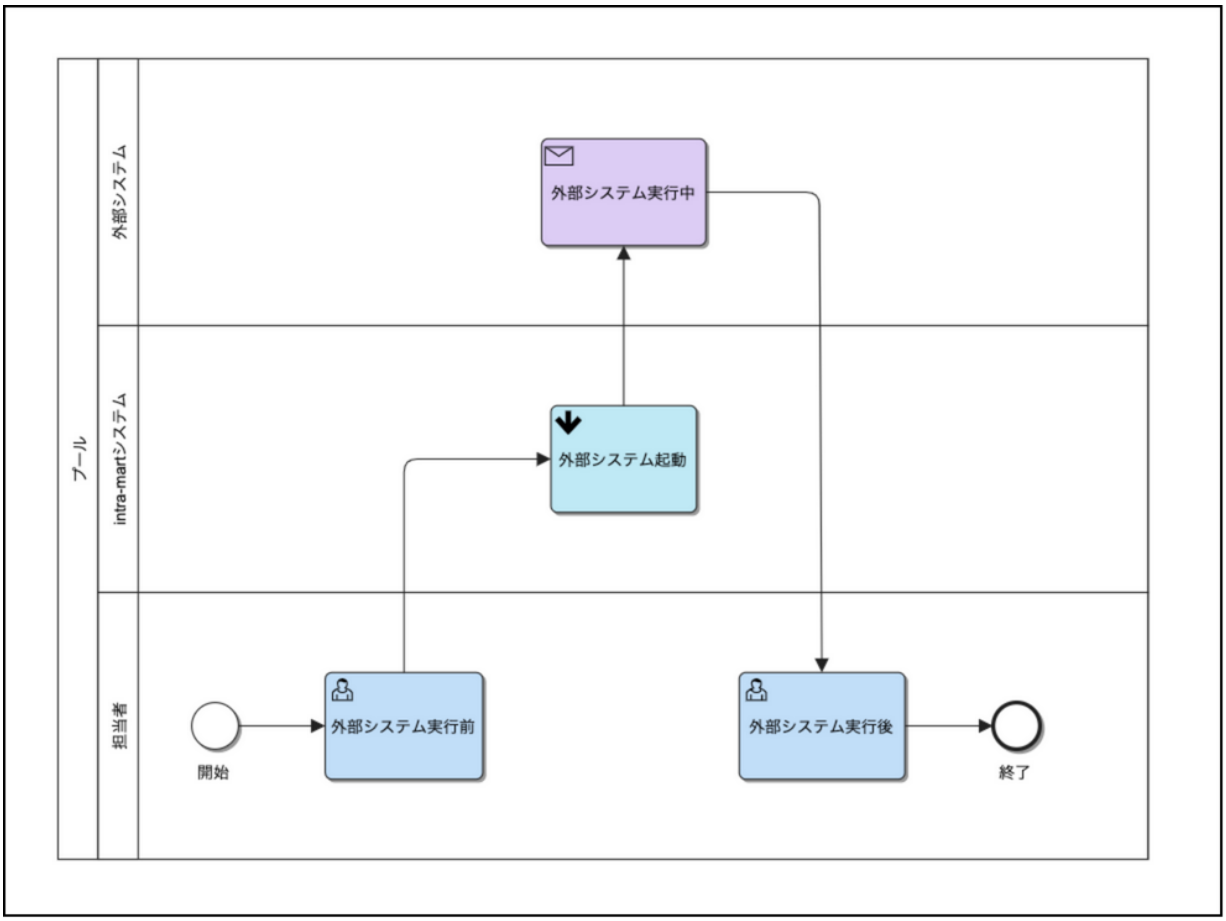
15. 「外部システム実行後」タスクに対して、担当者を設定します。
「外部システム実行後」タスクを選択し「メインコンフィグ」タブから「担当者」に aoyagi と設定します。



図：「ユーザタスク」-「プロパティ」 -「メインコンフィグ」

16. 「終了イベント」を「担当者レーン」へ設置します。

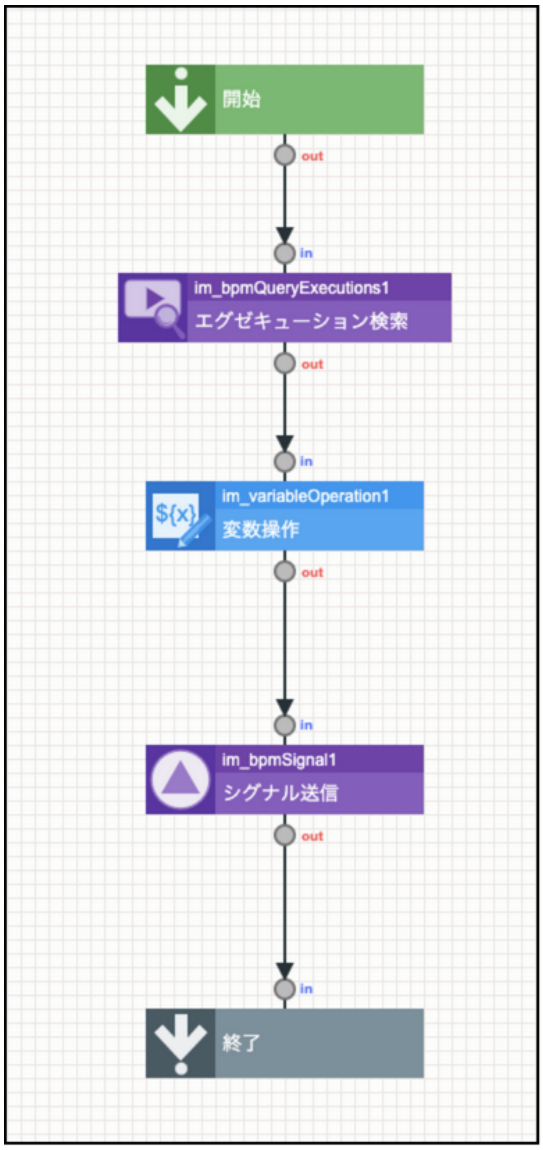
17. 以下に示すプロセス定義図を参考にエレメント同士をシーケンスフローでつなぎます。



図：プロセス定義図

外部システムの完了時に呼び出され、IM-BPMへシグナルを送信するロジックフローを作成する

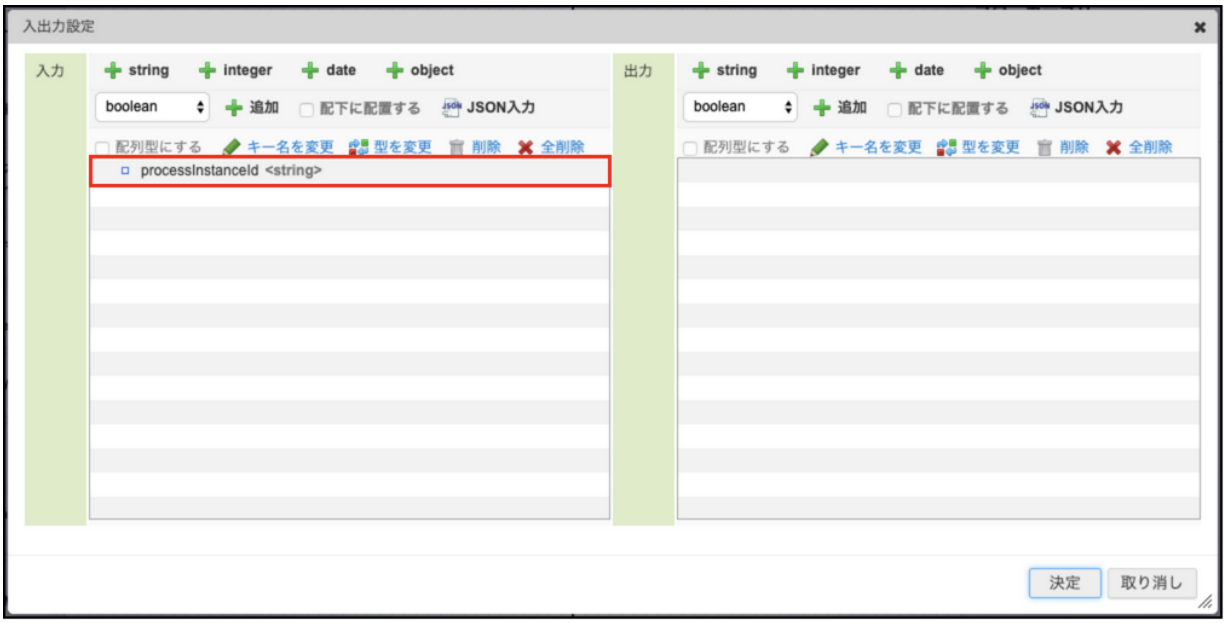
外部システムが完了したことをIM-BPMへ伝えるためのシグナルを発行するためのロジックフローを作成します。
作成したロジックフローにルーティングを設定し、REST APIとして外部システムから呼び出します。



図：完成イメージ（ロジックフロー）

1. ロジックフローを作成します。
   - 「サイトマップ」→「IM-LogicDesigner」→「フロー定義一覧」をクリックします。
   - 「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、「ロジックフロー定義編集」画面を表示します。
2. 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、入力でプロセスインスタンスIDを受け取るように設定します。
   - 入力
     外部システムから渡ってきた、連携のために外部システム実行時に渡していたIM-BPMのプロセスインスタンスID

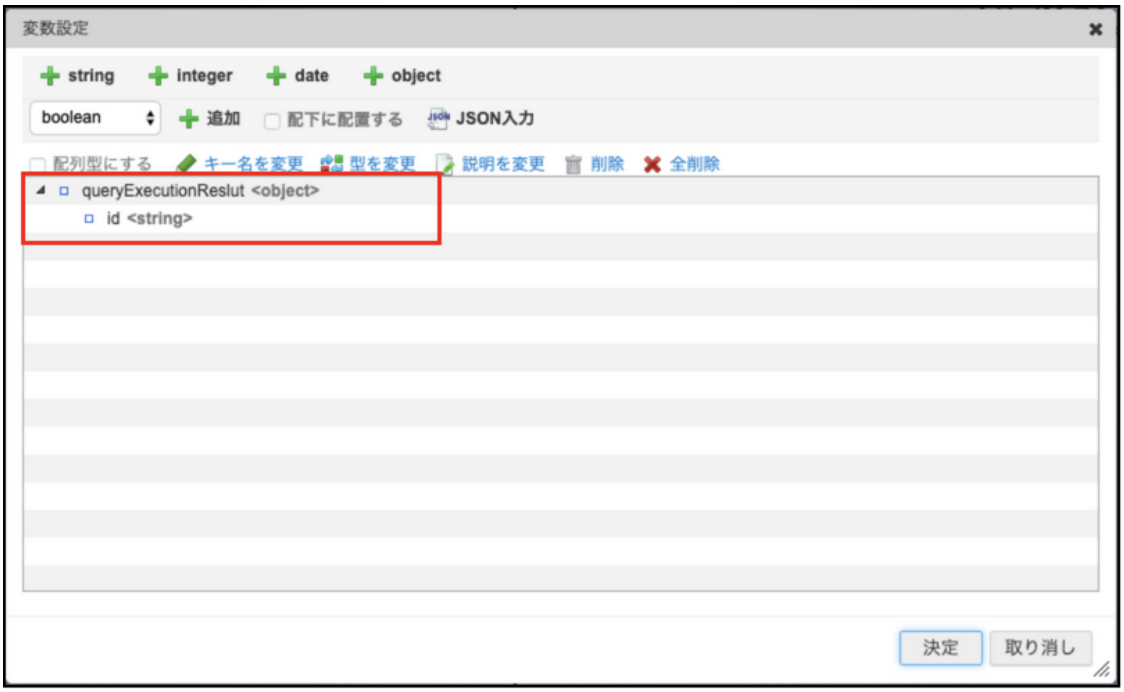
キー名	型
processInstanceId	<string>



図：「入出力設定」

3. メニューバーの「変数設定」をクリックし、「変数設定」を表示し、検索した結果対象となるエグゼキューションIDを保持する変数を設定します。

キー名	型
queryExecutionReslut	<object>
processInstanceId<object> - id	<string>



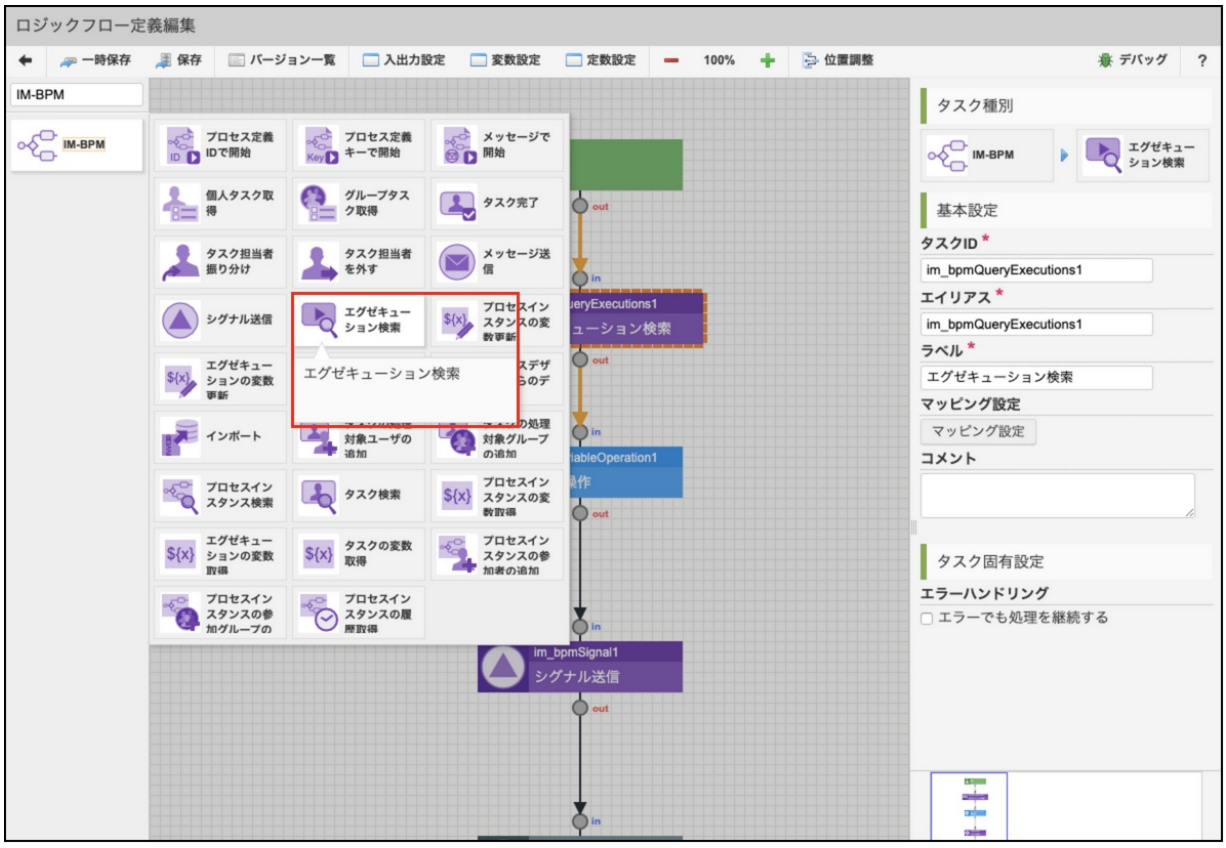
図：「変数設定」

4. メニューバーの「定数設定」をクリックし、「定数設定」を表示し、ロジックフローを実行する上で必要な定数を定義します。

定数ID	定数値	説明
zero	0	エグゼキューション検索で結果を取得するためのインデックス
activityId	external-system-receive-task	シグナル送信対象の受信タスクのアクティビティID

図：「定数設定」

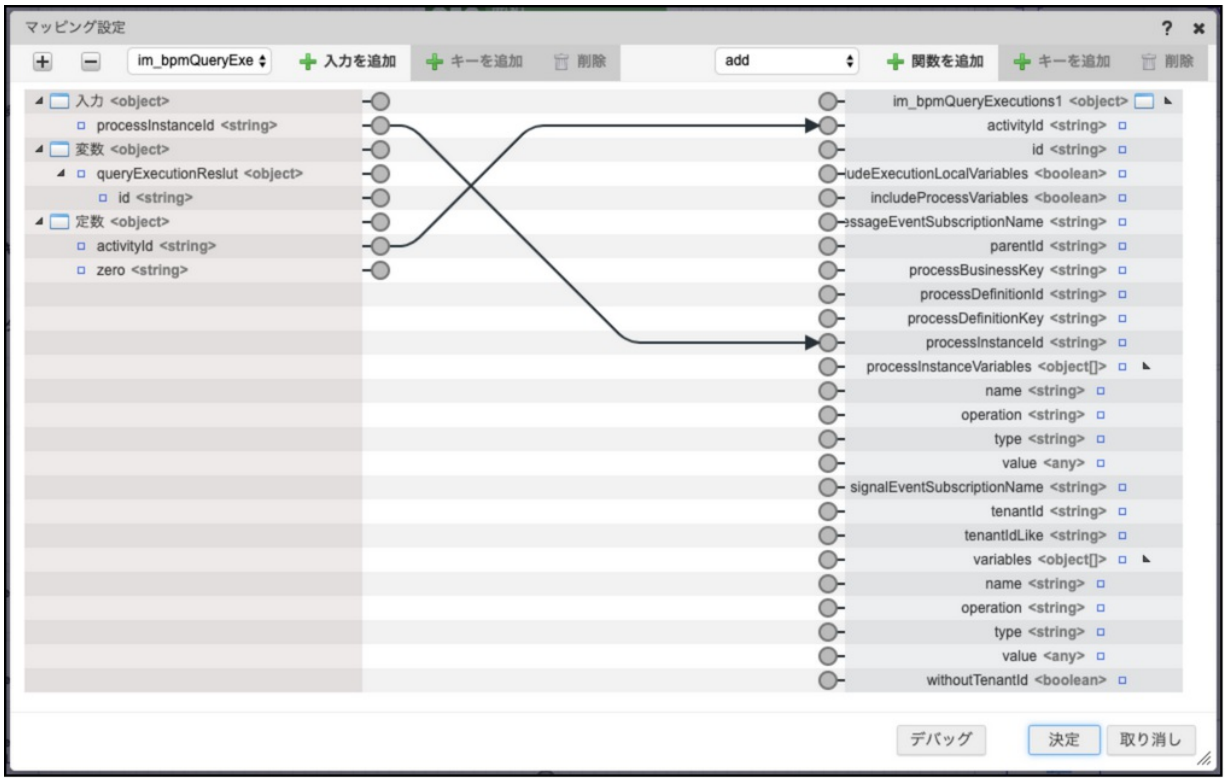
5. 「エグゼキューション検索」タスクを配置します。
   - パレット内の「IM-BPM」の一覧から「エグゼキューション検索」を選び、フロー編集画面上に追加します。



図：「ロジックフロー定義編集」

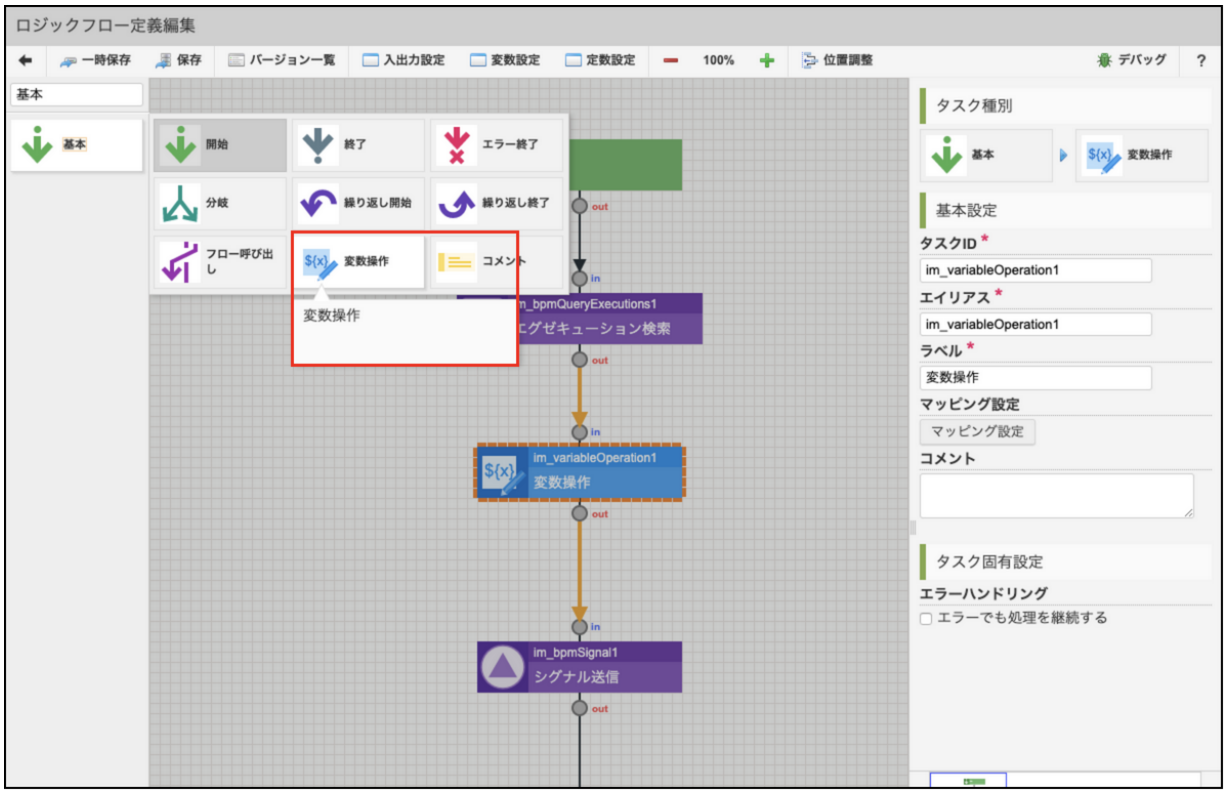
6. 「エグゼキューション検索」のマッピング設定をします。
   - 以下のように線を繋ぎます。

入力（始点）	出力（終点）
入力<object> - processInstanceId	im_bpmQueryExecutions1<object> - processInstanceId
定数<object> - activityId	im_bpmQueryExecutions1<object> - activityId



図：「マッピング設定」 -「エグゼキューション検索」

7. 「変数操作」タスクを配置します。
   - 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：「ロジックフロー定義編集」

8. 「変数操作」のマッピング設定をします。
   - 「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「get」を選択します。
     「関数を追加」をクリックします。

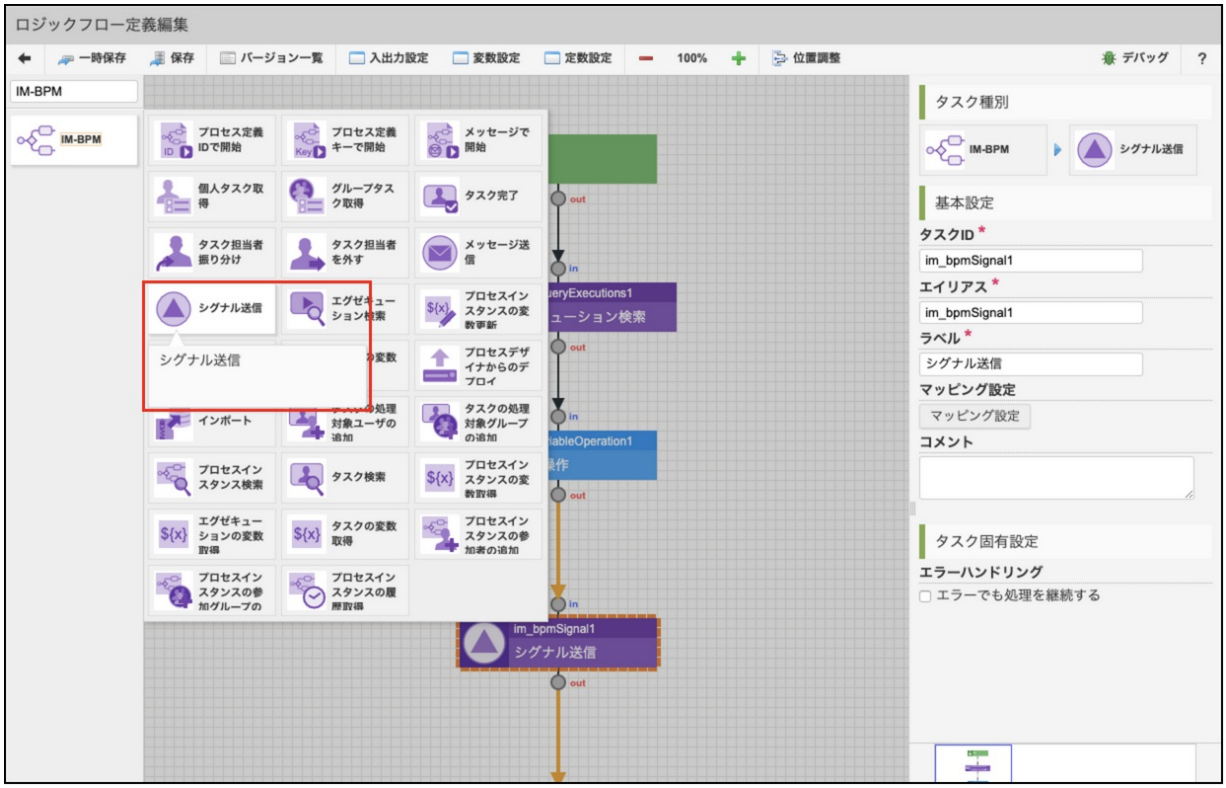
> **コラム**
>
> マッピング関数については、「IM-LogicDesigner仕様書」-「マッピング関数一覧」を参照してください。

   - 以下のように線を繋ぎます。

入力（始点）	出力（終点）
im_bpmQueryExecutions1<object> - queryExecutionsResults<object[]>	get: array
定数<object> - zero	get: index
get: out	定数<object> - queryExecutionReslut

図：「マッピング設定」 - 「変数操作」

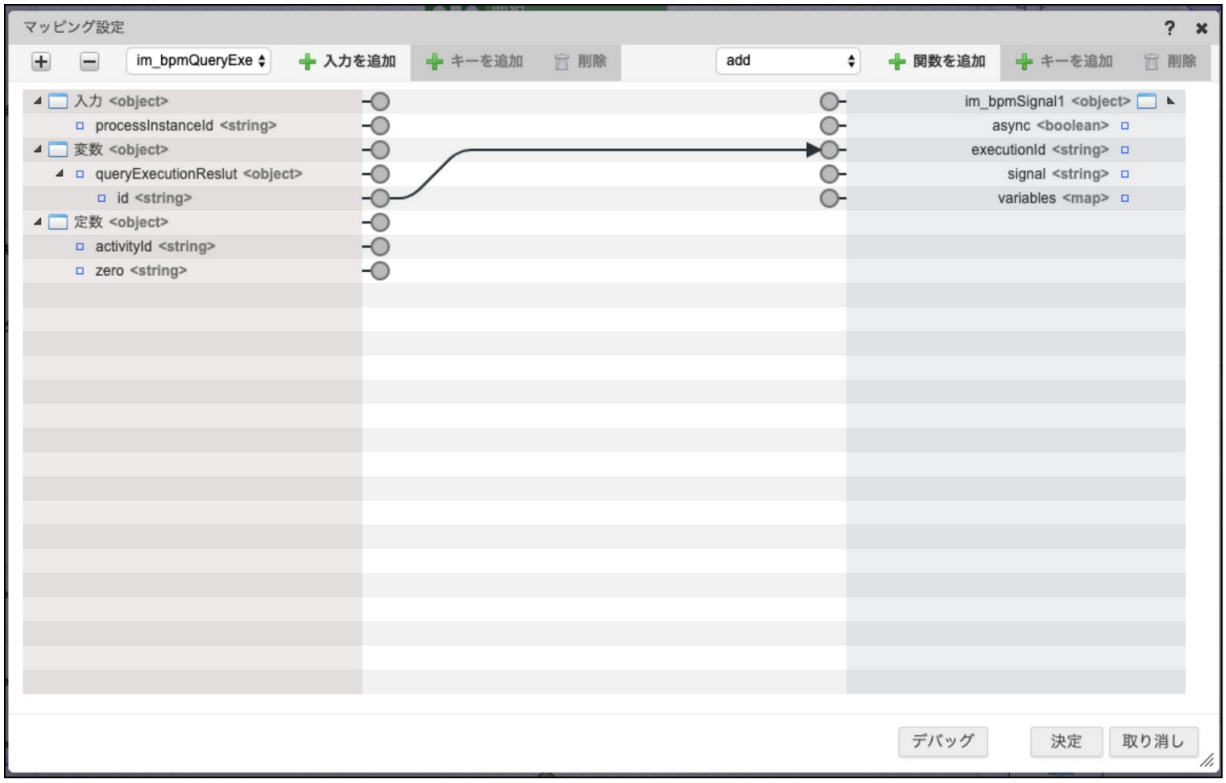
9. 「エグゼキューション検索」タスクを配置します。
   - パレット内の「IM-BPM」の一覧から「シグナル送信」を選び、フロー編集画面上に追加します。



図：「ロジックフロー定義編集」

10. 「シグナル送信」のマッピング設定をします。
    - 以下のように線を繋ぎます。

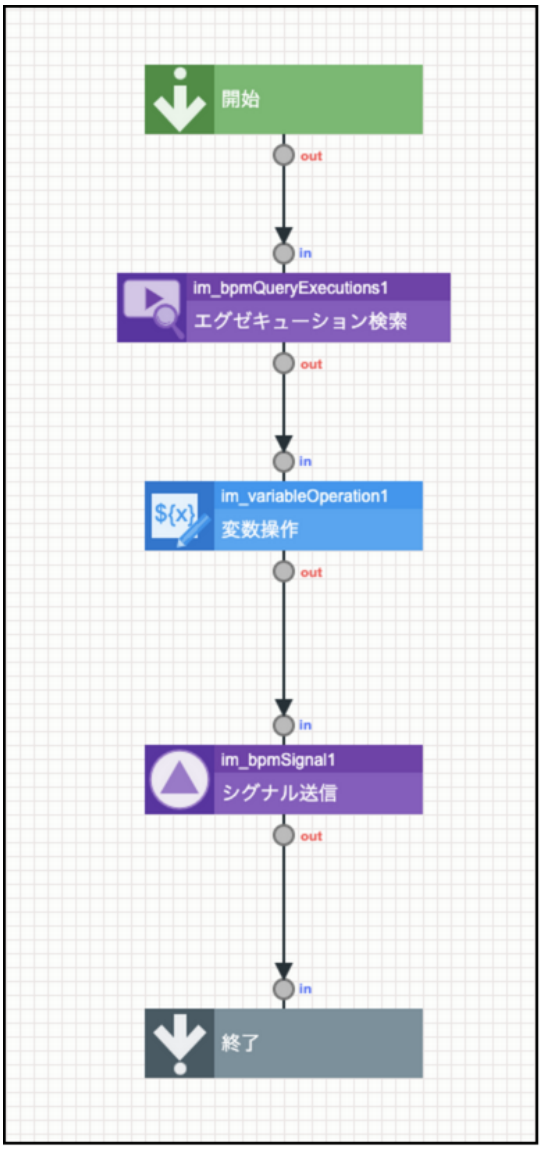
入力（始点）	出力（終点）
変数<object> - queryExecutionReslut<object> - id	im_bpmSignal1<object> - executionId



図：「マッピング設定」 - 「シグナル送信」

11. タスクを以下のようにつなぎます。

out	in
開始	エグゼキューション検索
エグゼキューション検索	変数操作
変数操作	シグナル送信
シグナル送信	終了



図：完成イメージ（ロジックフロー）

12. 「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：sendSignalReceiveTask
    - フロー定義名：外部システム完了時シグナル送信フロー
    - フローカテゴリ：
        - ID：im_logicdesigner-data
        - 名称：BPMチュートリアル

13. 作成したロジックフローに対してルーティング定義を作成し、REST APIとして外部システムから呼び出せるようにします。
    - 「サイトマップ」→「IM-LogicDesigner」→「ルーティング定義一覧」
    - 「ロジックフロールーティング定義一覧」画面、ツールバー内の「新規作成」をクリックし、「ルーティング定義編集」画面を表示します。

14. 「対象ロジックフロー定義情報」の対象フロー項目に以下を選択します。
    - フロー定義ID: sendSignalReceiveTask
    - フロー定義名: 外部システム完了時シグナル送信フロー

図：「ルーティング定義編集」-「対象ロジックフロー定義情報」

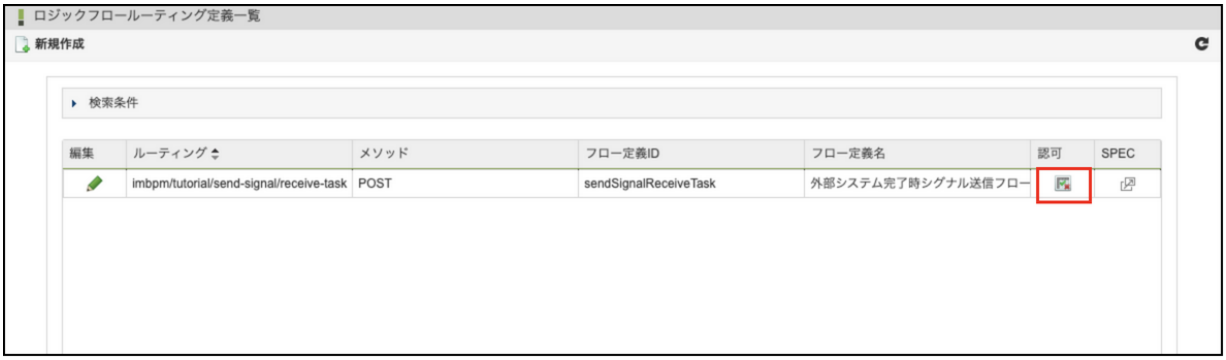
15. 「ロジックフロールーティング定義情報」の各項目に以下を入力します。

- ルーティング: imbpm/tutorial/send-signal/receive-task
- メソッド: POST
- 認可URI: imbpm/tutorial/send-signal/receive-task

図：「ルーティング定義編集」-「ロジックフロールーティング定義情報」

16. 「登録」ボタンをクリックし、ルーティング定義を保存します。

17. 「ロジックフロールーティング定義一覧」にて今回作成した「外部システム完了時シグナル送信フロー」の「 」をクリックし「認可設定」画面を表示します。



図：「ロジックフロールーティング定義一覧」

18. 「ゲストユーザ」、「認証済みユーザ」に対して許可します。

図：「外部システム完了時シグナル送信フロー」-「認可設定」

19. これにより、外部システムから <ベースURL>/logic/api/imbpm/tutorial/send-signal/receive-task に対して、対象のプロセスインスタンスIDをパラメータ processInstanceId に詰めて送信することで、該当の受信タスクが完了するようになりました。
外部システムの完了時の動作にこのREST APIへリクエストを送信するように設定してください。

> **コラム**
>
> このREST APIをcurlで使用する例は以下です。
>
> - curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ "processInstanceId": "プロセスインスタンスID" }' '<ベースURL>/logic/api/imbpm/tutorial/send-signal/receive-task'

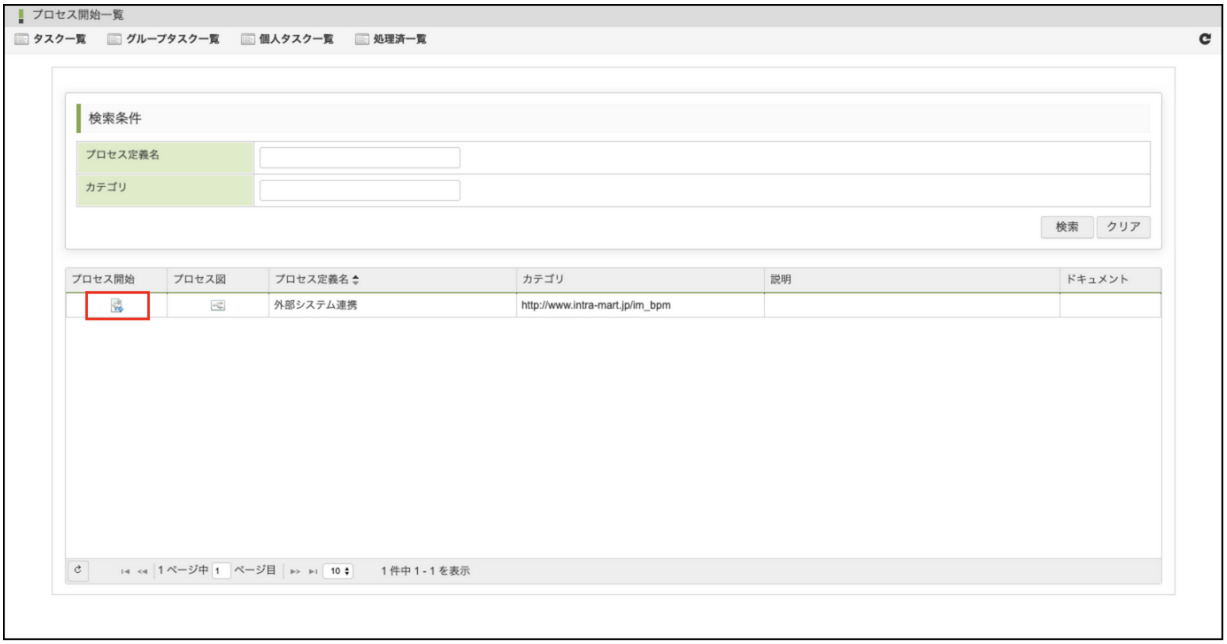
作成したプロセスの動作確認をする

作成した「プロセス定義」を実行環境にデプロイし、動作の確認を行います。

1. プロセスを開始します。

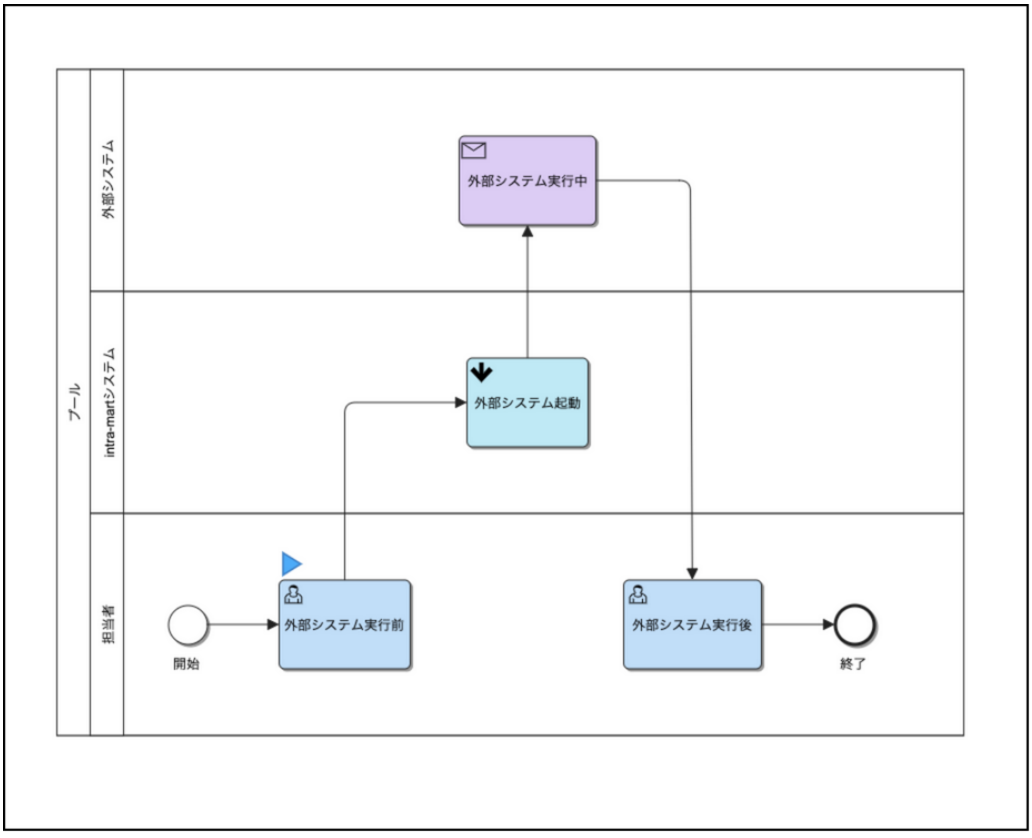
「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示します。

2. 「プロセス開始一覧」画面から、対象となるプロセスの「 」をクリックします。



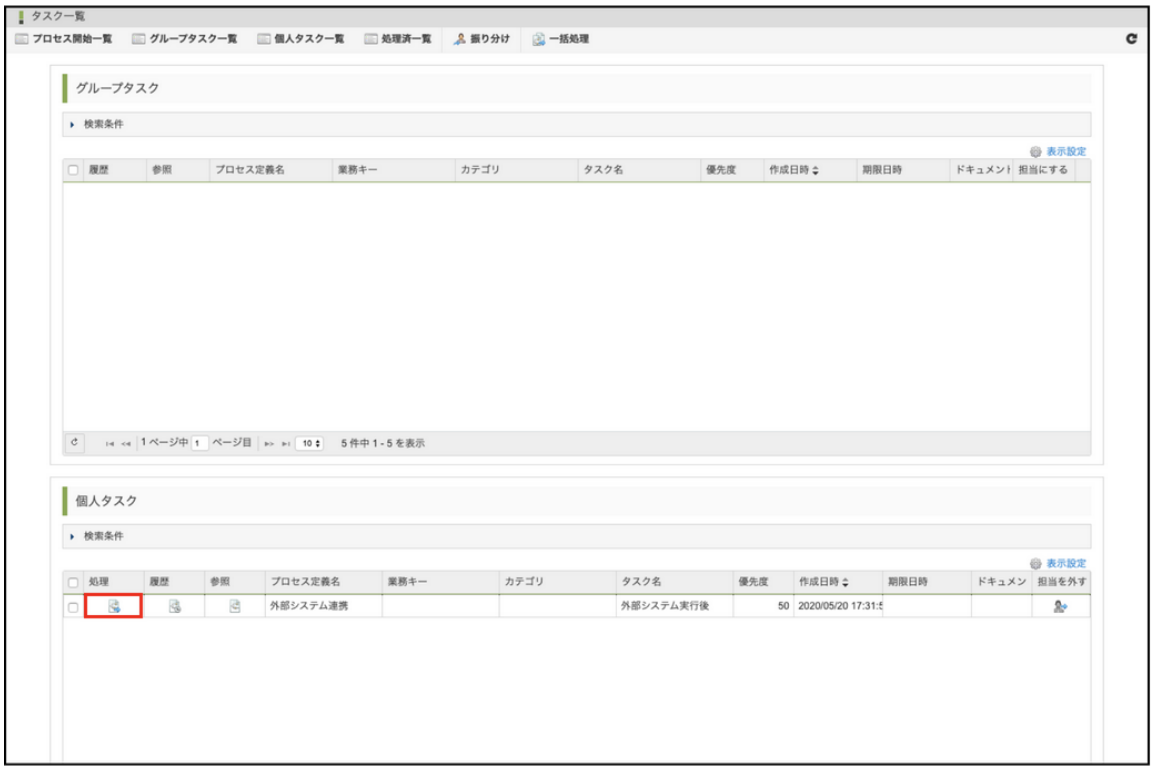
図：「プロセス開始一覧」

3. 開始直後のプロセスの状態は図のようになっており、まだ外部システムは実行されていません。



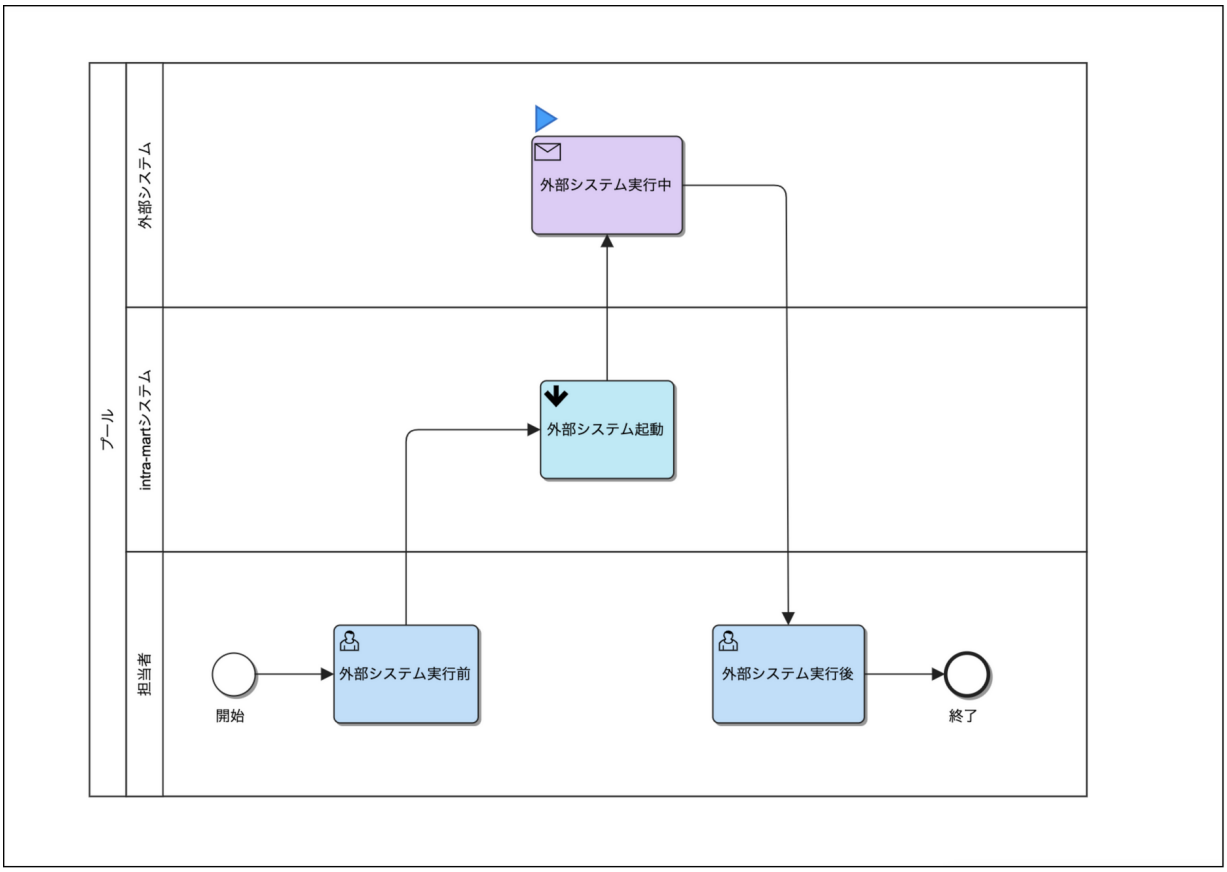
図：「プロセス図」

4. 「外部システム実行前」タスクを処理します。
   - 「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。
   - 「外部システム実行前」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」

5. 「外部システム実行前」タスクを処理することで、「外部システム起動」タスクへ到達します。
紐付けられた、ロジックフローによって外部システムが起動し、次の「外部システム実行中」へ到達します。
外部システムからのシグナルを受信するまではこのタスクに実行中マークが付与されます。



図：プロセス定義図

コラム

REST定義作成の際に、エンドポイントを 「<ベースURL>/logic/api/im_bpm_tutorial/receive_task/test」 と設定して追加資材をインポートしている場合は、渡されたプロセスインスタンスIDがログへ出力されています。
これは、追加資材の中のロジックフローにあるログ出力タスクによって、入力値の値をログへ出力するよう設定されているからです。

```
[INFO] j.c.i.f.l.e.g.OutputLogTask - □ 外部システムが実行されました。
プロセスインスタンスID: 8flr5lfgdq9i19a
```

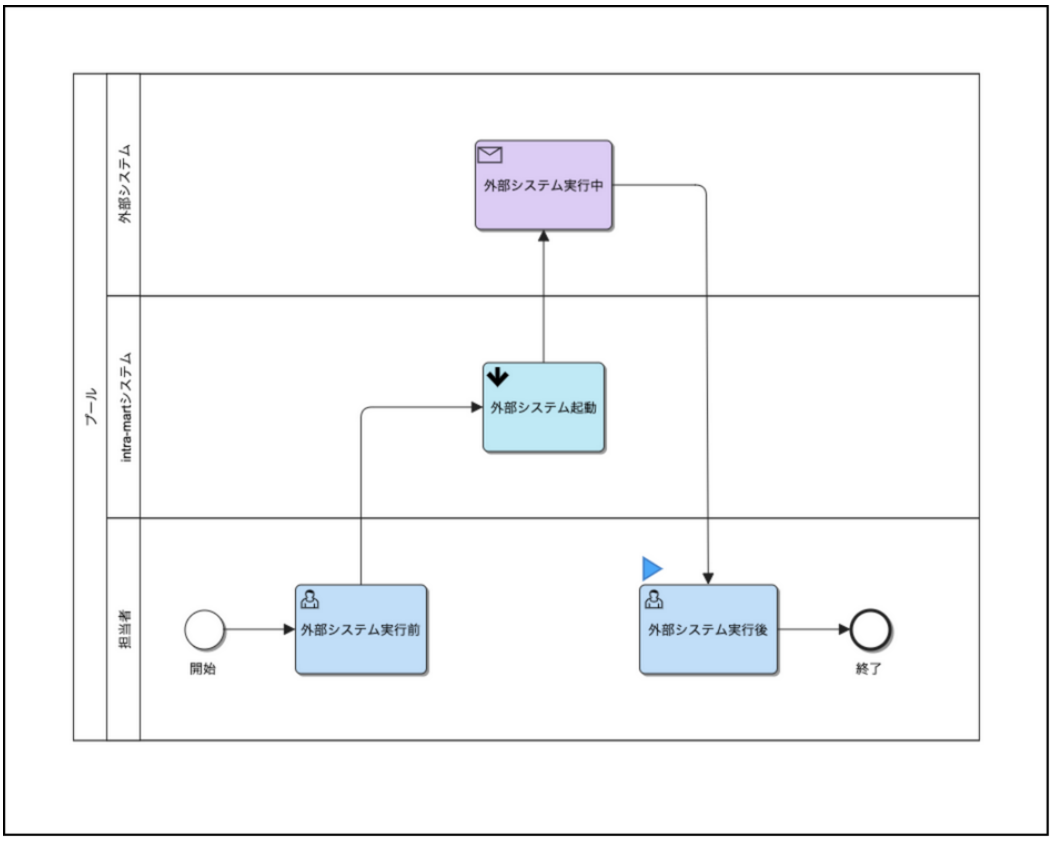
図：ログ出力

6. 外部システムの完了をintra-martへ知らせます。
   - 外部システムが完了した際に設定したリクエストが送信されることで、 ロジックフローが動作し、対象の受信タスクへシグナルが送信されます。

コラム

以下のcurlコマンドを実行することで、擬似的に外部システム完了のリクエストを送信できます。
`curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ "processInstanceId": "プロセスインスタンスID" }' '<ベースURL>/logic/api/imbpm/tutorial/send-signal/receive-task'`

7. 外部システムが完了しシグナルが送信されると、「外部システム実行中」タスクが完了し、「外部システム実行後」タスクに到達します。



図：プロセス定義図

コールアクティビティ

コールアクティビティを使用する

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

コラム

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

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

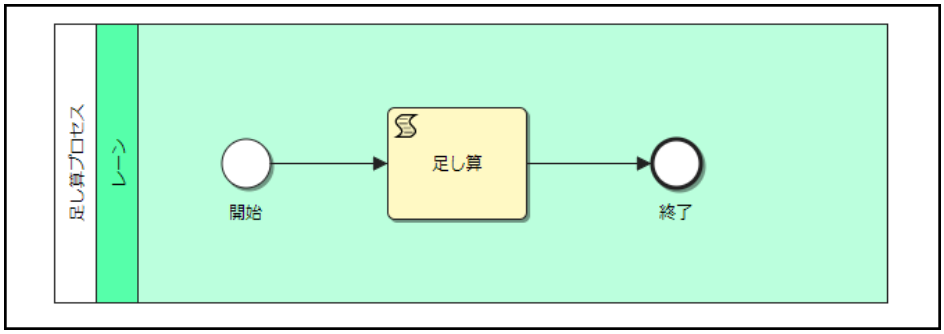
コラム

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

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

呼び出されるプロセス定義を作成する

コールアクティビティから呼び出されるプロセス定義を作成します。



図：呼び出されるプロセス定義

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

val + 数字 ： val1 , val2 , val3 , ・・・

スクリプトタスク「足し算」のスクリプトです。

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

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

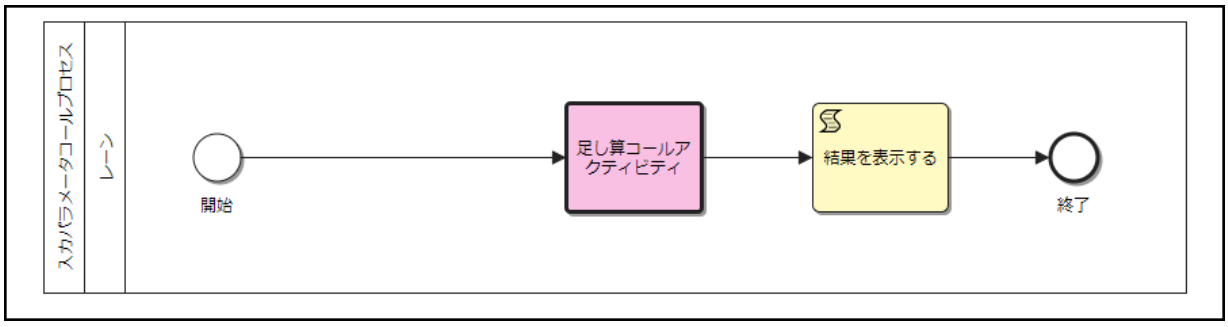
「プロセス定義キー」に addition_process を設定します。

図：プロセス定義キー

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

入力パラメータを設定してプロセス定義を呼び出す

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

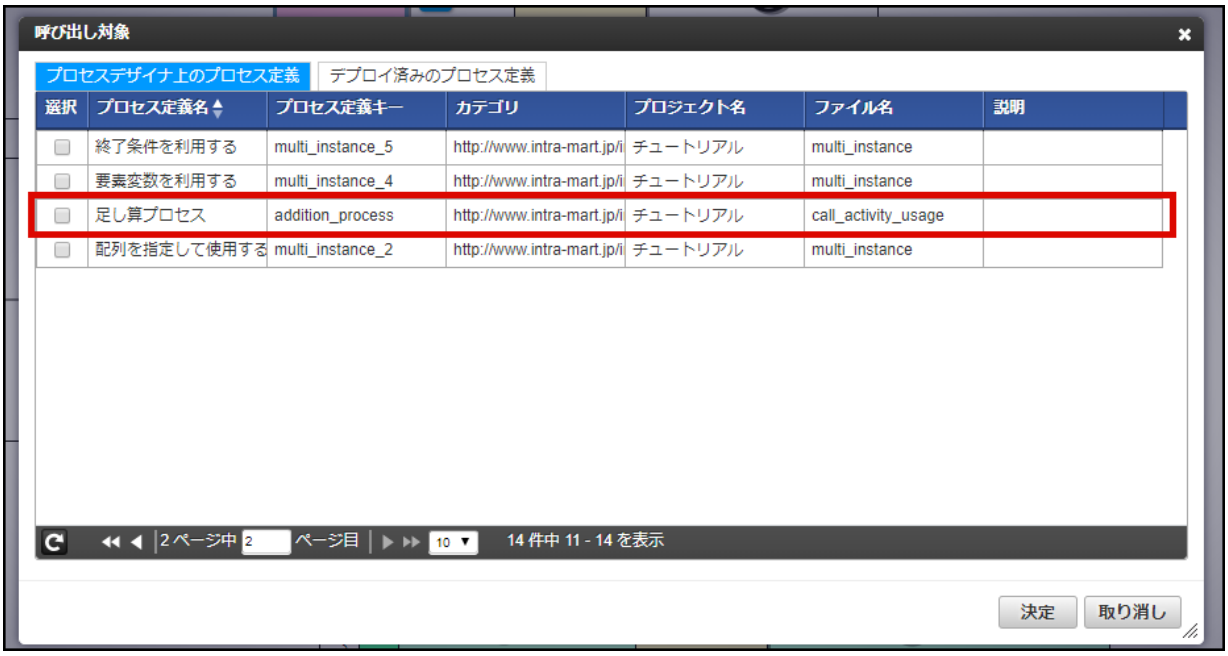


図：完成イメージ

図：コールアクティビティ - メインコンフィグ

1. 「呼び出し対象」に addition_process 指定します。

直接入力するか、プロセス定義検索（ ）を使用して指定します。



図：プロセス定義検索

> **コラム**
>
> プロセス定義検索
>
> 「プロセス定義検索」子画面で、参照可能なプロセス定義を検索し、「呼び出し対象」の項目へ設定できます。
>
> - **「プロセスデザイナ上のプロセス定義」タブ**
> 「プロセスデザイナ上のプロセス定義」タブでは、プロセスデザイナ上のプロセス定義が表示されます。
> 「参照」アクションが許可されているプロジェクトのプロセス定義が表示されます。
> - **「デプロイ済みのプロセス定義」タブ**
> 「デプロイ済みのプロセス定義」タブでは、既にデプロイされているプロセス定義が表示さます。
> プロセス定義の検索の認可がある場合のみ表示されます。
>
> プロジェクトの認可設定については「IM-BPM プロセスデザイナ 操作ガイド」-「プロジェクトの管理」-「プロジェクトの認可設定」を参照してください。

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

2. 「入力パラメータ」を設定します。

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

リンクをクリックします。

図：コールアクティビティ - メインコンフィグ - 入力パラメータ

1. 「入力パラメータ」の「ソース種別」に「式」を選択します。
2. 「ソース式」にEL式で数字を設定します。

   例： `${123}`
3. 「ターゲット変数名」に `val1` （val + 連番（1, 2, 3, ...））を設定します。
4. 上記の手順を繰り返し、足しあわせたい数字を全て「入力パラメータ」に設定します。

図：コールアクティビティ - メインコンフィグ

3. 「出力パラメータ」を設定します。

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

   リンクをクリックします。

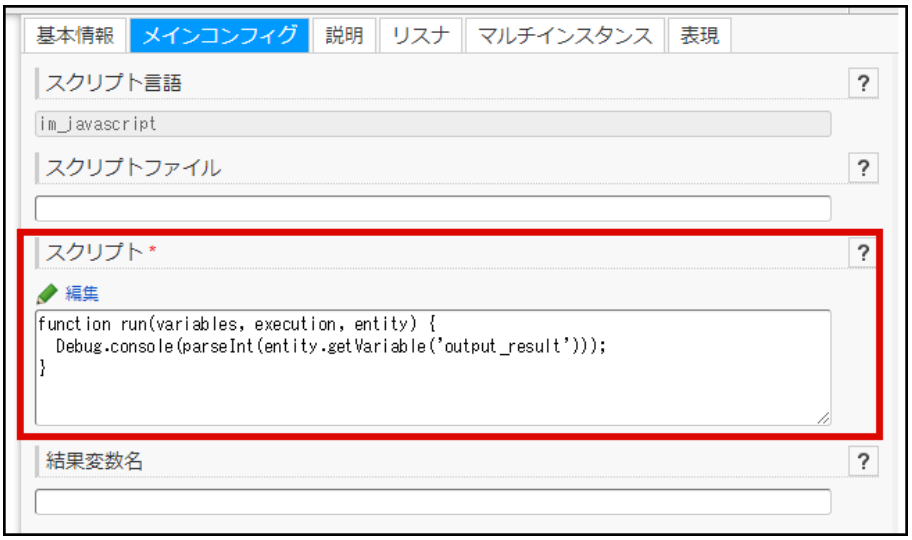
図：コールアクティビティ - メインコンフィグ - 出力パラメータ

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

4. 結果を表示する「スクリプトタスク」を作成します。

   「スクリプトタスク」のスクリプトです。

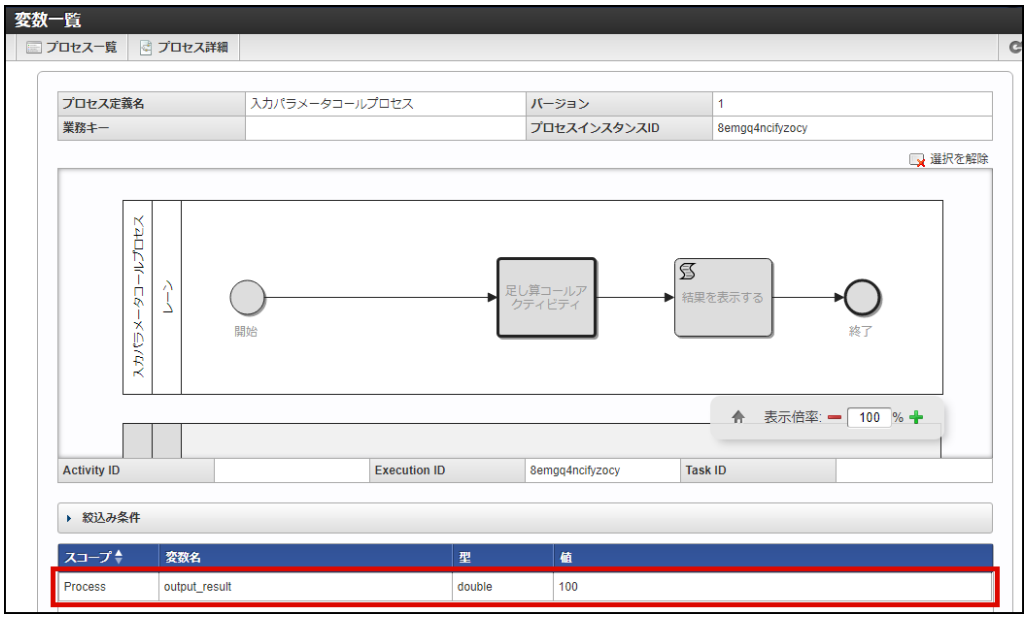
```
function run(variables, execution, entity) {
  Debug.console(parseInt(entity.getVariable('output_result')));
}
```



図：スクリプトタスク - メインコンフィグ - スクリプト

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

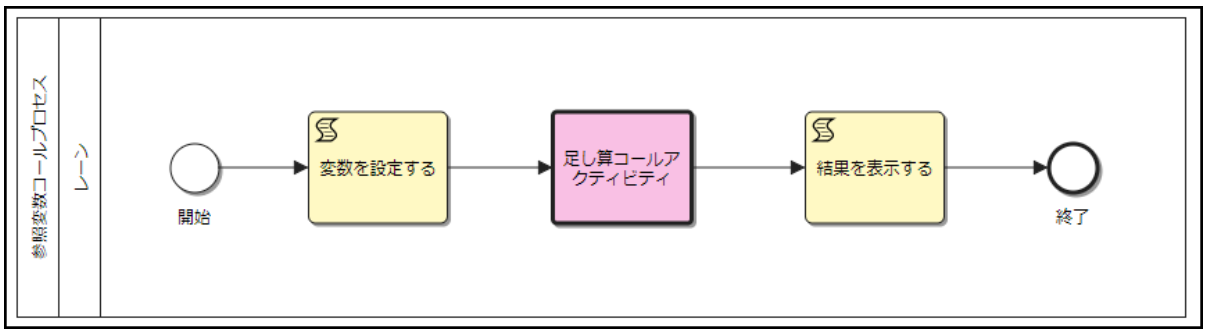
図：結果表示



図：変数一覧

参照可能な変数を継承してプロセス定義を呼び出す

コールアクティビティで参照可能な変数を継承してプロセス定義を呼びます。



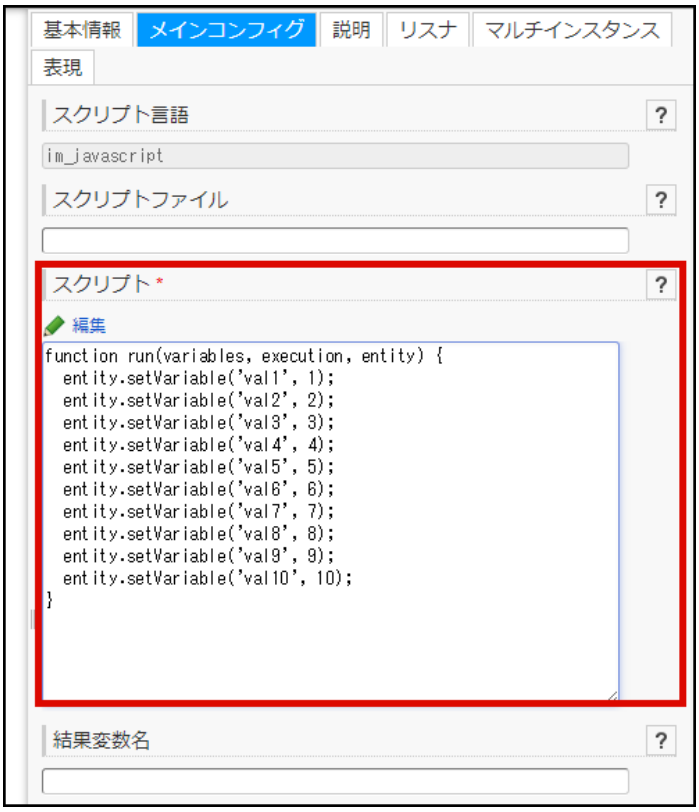
図：完成イメージ

図：コールアクティビティ - メインコンフィグ

1. 変数を設定するスクリプトタスクを作成します。

   「スクリプトタスク」のスクリプトです。

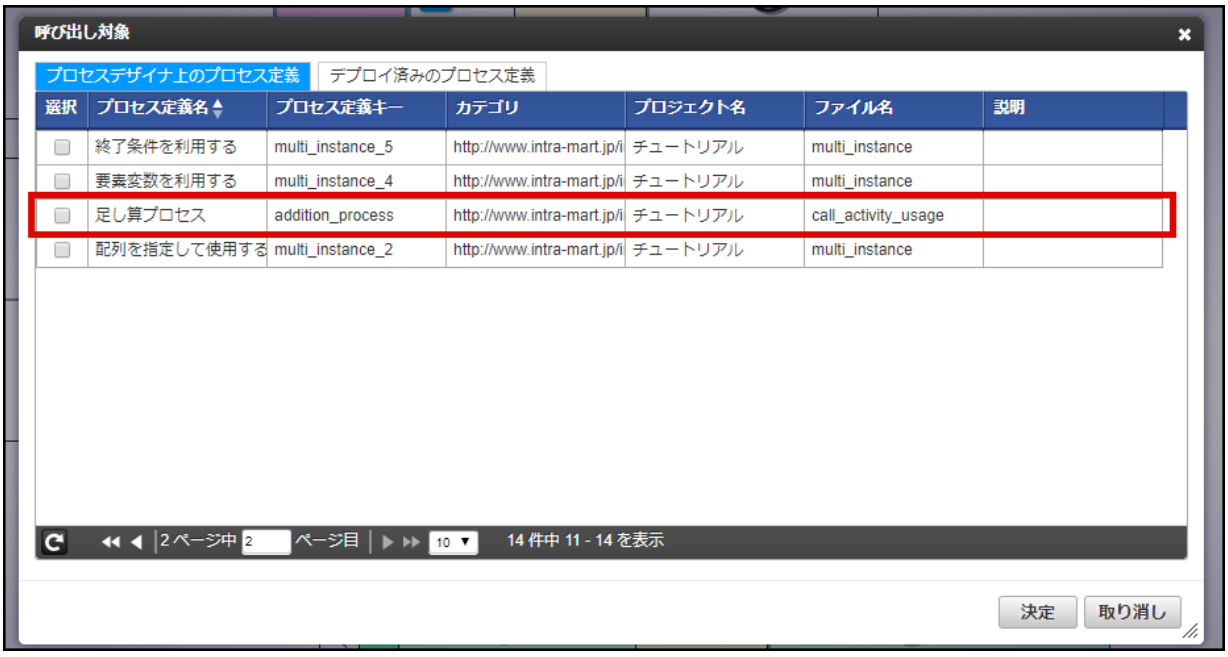
```
function run(variables, execution, entity) {
  entity.setVariable('val1', 1);
  entity.setVariable('val2', 2);
  entity.setVariable('val3', 3);
  entity.setVariable('val4', 4);
  entity.setVariable('val5', 5);
  entity.setVariable('val6', 6);
  entity.setVariable('val7', 7);
  entity.setVariable('val8',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



図：スクリプトタスク - メインコンフィグ - スクリプト

2. 「呼び出し対象」に addition_process 指定します。

   直接入力するか、プロセス定義検索を使用して指定します。



図：プロセス定義検索

3. 「参照可能な変数を継承する」を有効にします。

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

4. 「出力パラメータ」を設定します。

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

リンクをクリックします。

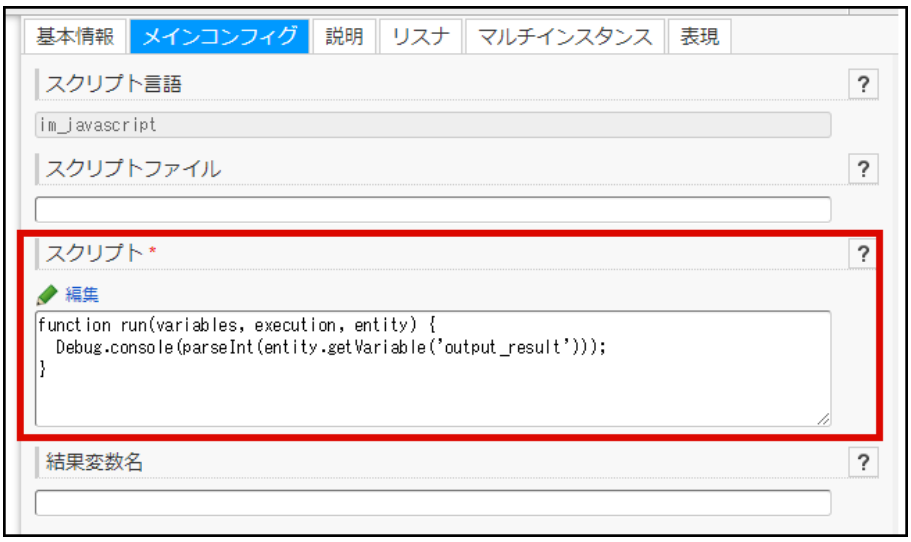
図：コールアクティビティ - メインコンフィグ - 出力パラメータ

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

5. 結果を表示する「スクリプトタスク」を作成します。

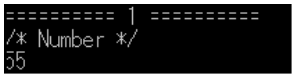
「スクリプトタスク」のスクリプトです。

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

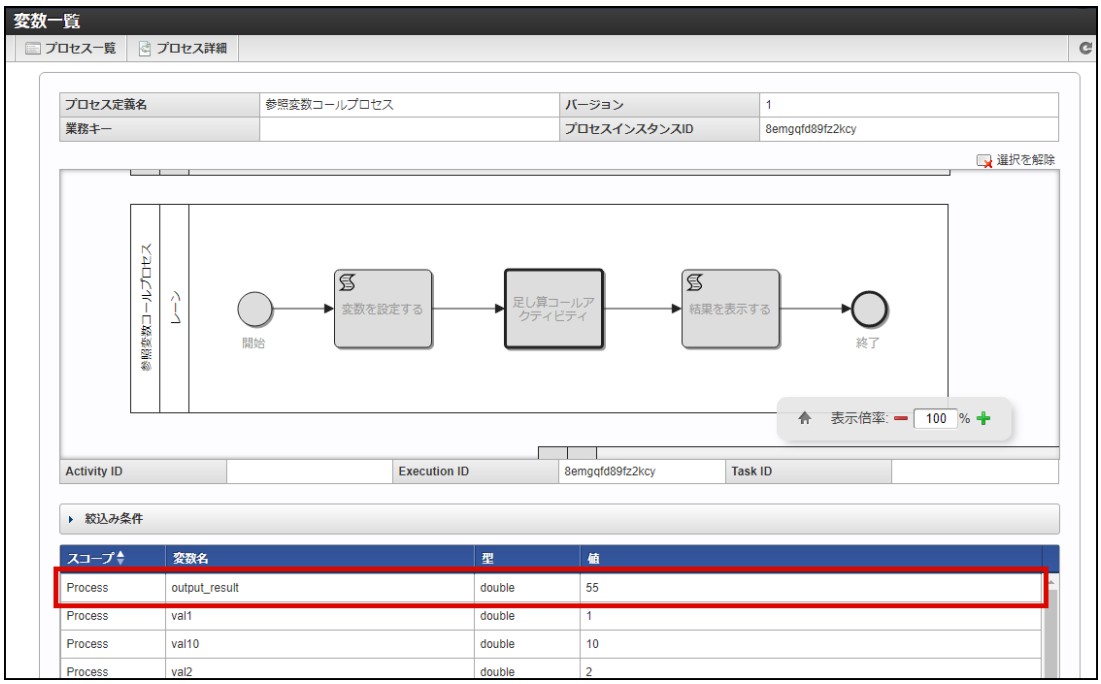


図：スクリプトタスク - メインコンフィグ - スクリプト

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



図：結果表示



図：変数一覧

コールアクティビティで呼び出すプロセス定義に業務キーを設定する

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

コラム

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

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

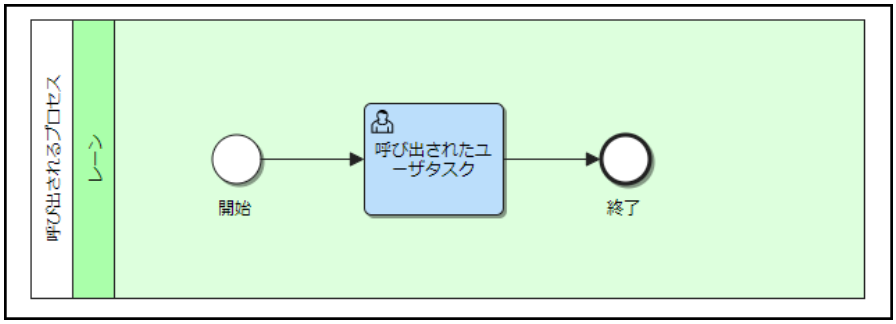
コラム

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

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

呼び出されるプロセス定義を作成する

コールアクティビティから呼び出されるプロセス定義を作成します。

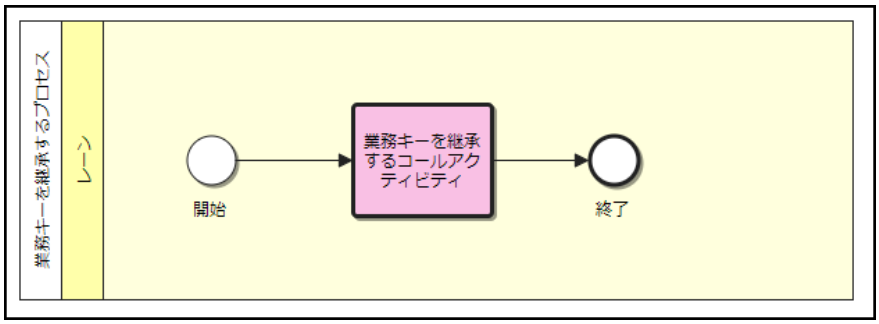


図：呼び出されるプロセス定義

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

呼び出し元のプロセス定義から業務キーを継承する

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



図：完成イメージ

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

   直接入力するか、プロセス定義検索（🔎）を使用して指定します。

   コラム

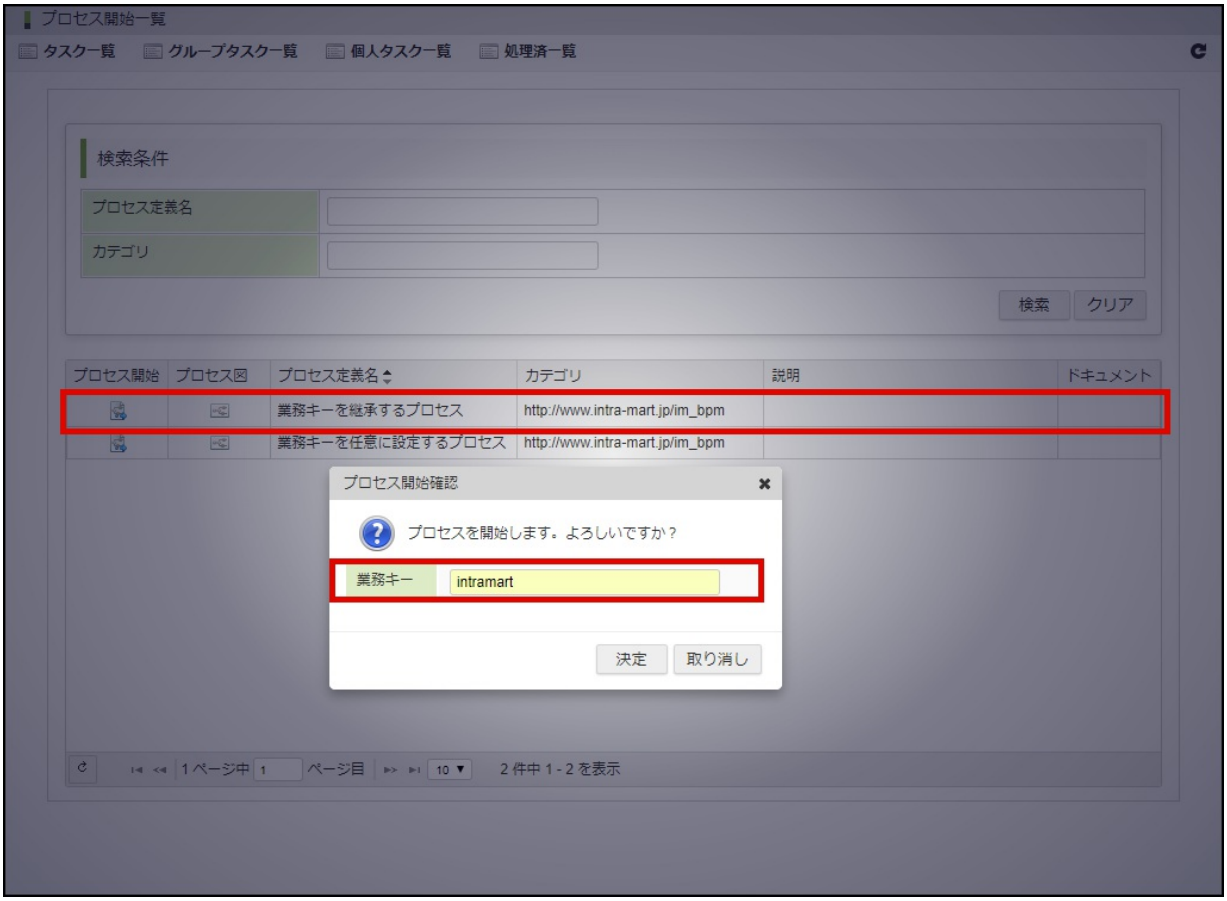
   「呼び出し対象」の指定は、「*コールアクティビティを使用する*」を参考にしてください。

2. 「業務キーを継承する」を有効にします。

図：コールアクティビティ - メインコンフィグ

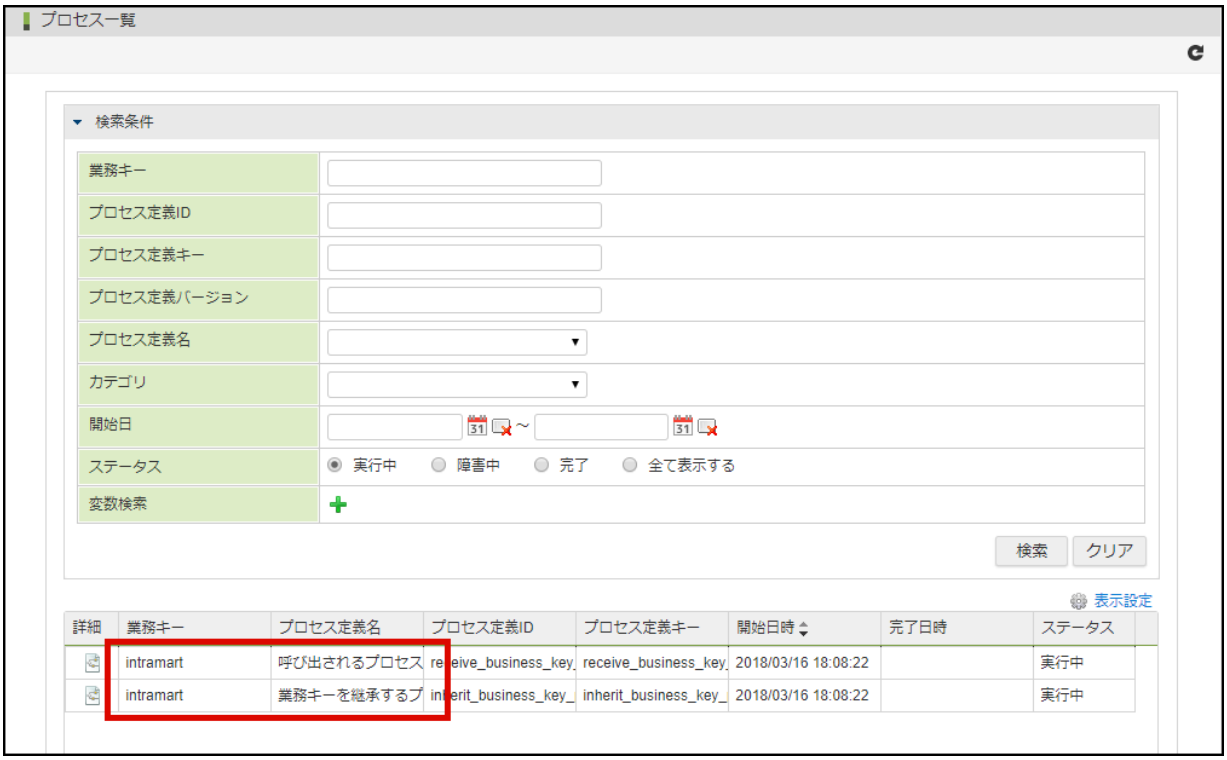
実行結果を確認します。

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



図：プロセス開始一覧

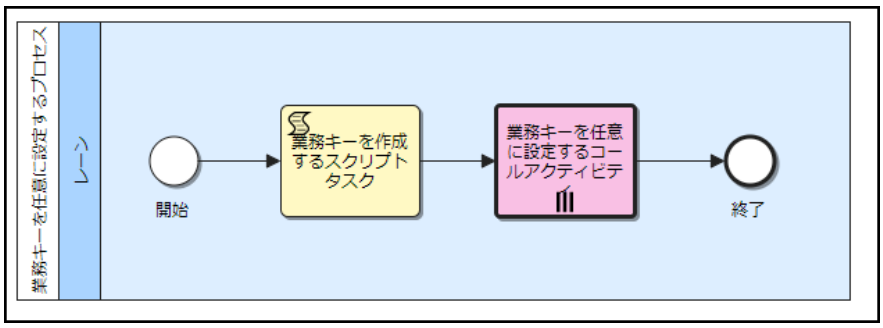
3. プロセス一覧で確認します。



図：プロセス一覧

任意に業務キーを設定する

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



図：完成イメージ

1. スクリプトタスクで、業務キーの配列を作成します。

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

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

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

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

   直接入力するか、プロセス定義検索（ ）を使用して指定します。

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

3. 「業務キー」に `${businessKey}` 指定します。

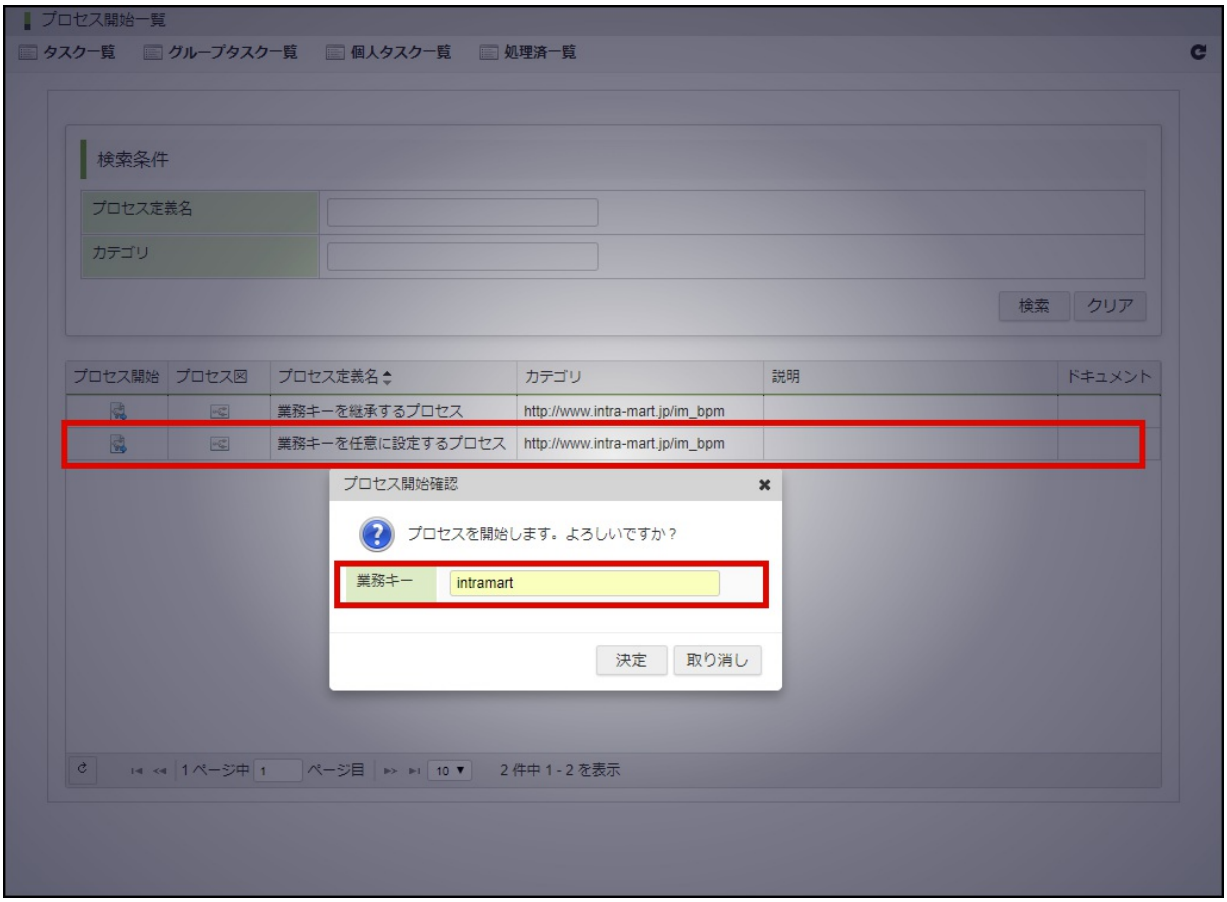
図：コールアクティビティ - メインコンフィグ

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

図： マルチインスタンス

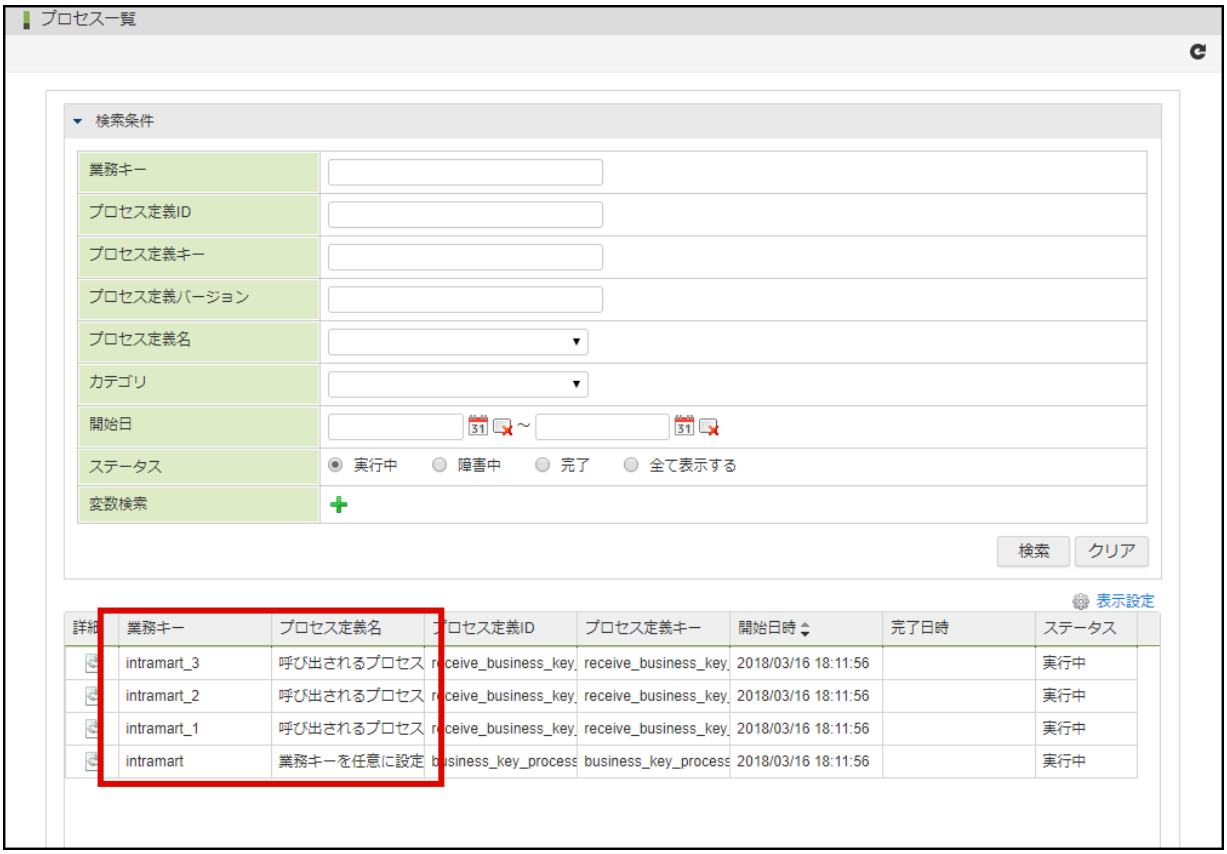
実行結果を確認します。

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



図：プロセス開始一覧

3. プロセス一覧で確認します。



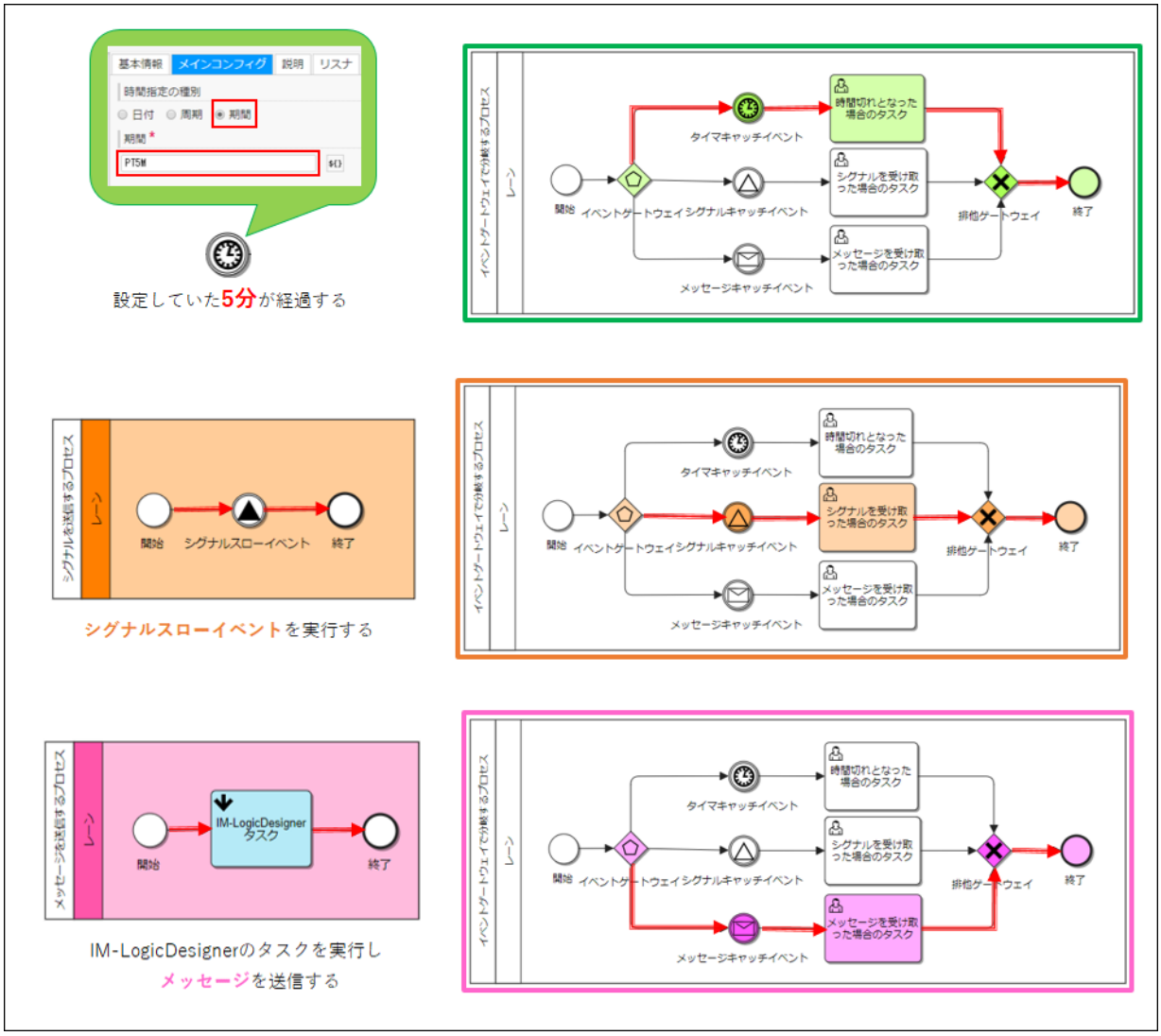
図：プロセス一覧

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、最初に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 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」のロジックフローを使用します。
チュートリアルを開始する前に以下の資材をインポートしてください。
im_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 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 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 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。

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 プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

コラム

このチュートリアルのサンプルでは、同一ファイル内に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。

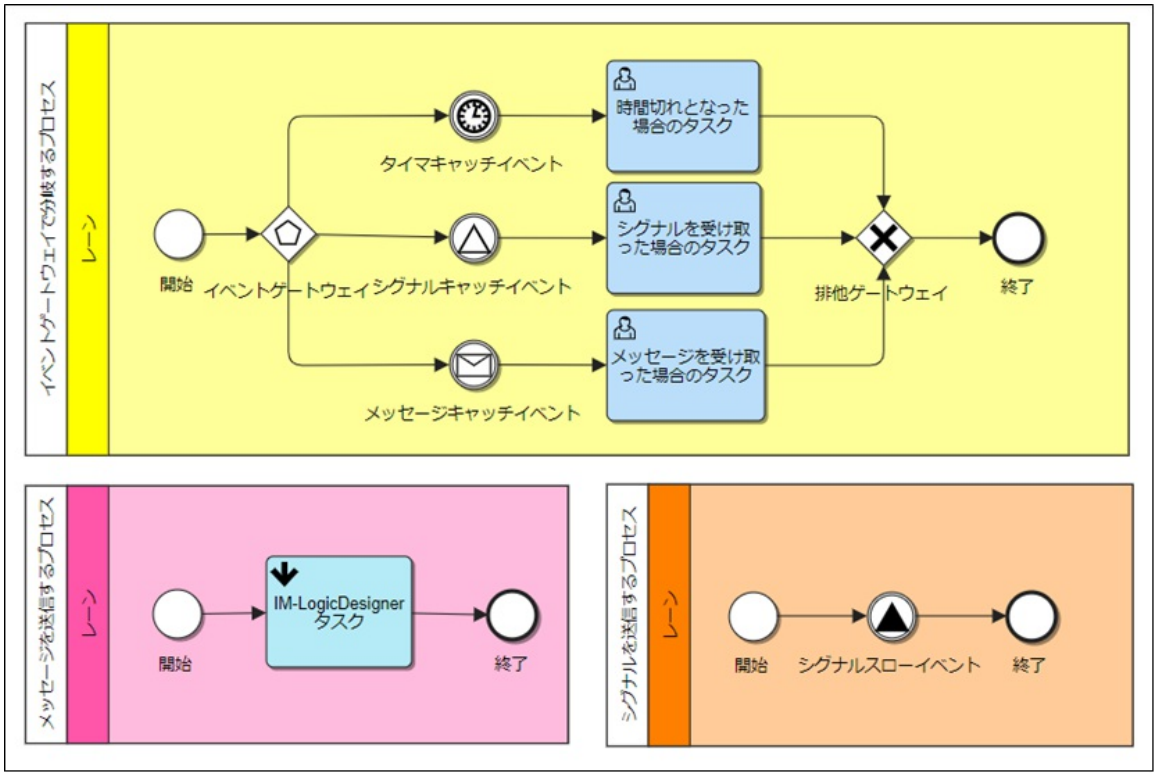
- プロセス定義を作成する
- 結果を確認する

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つのキャンバ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

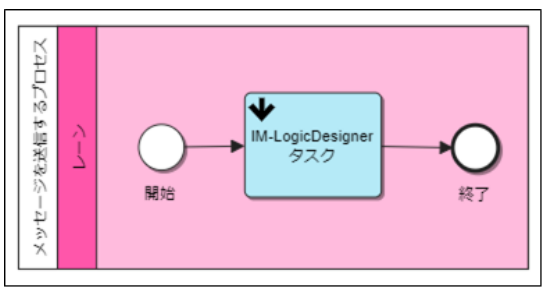
プロセスが進行する条件は「メッセージイベント・シグナルイベントのいずれかを受信する」か「タイマイベントで設定した時刻に到達する」です。



図：完成イメージ

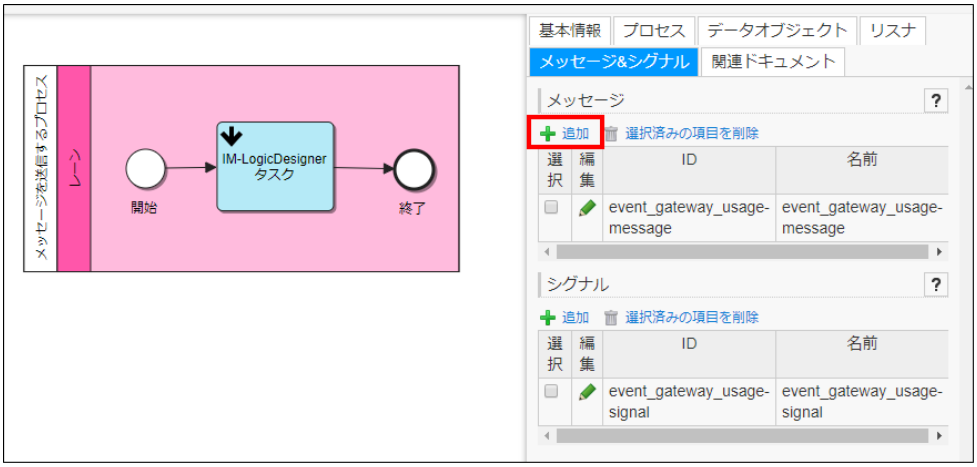
メッセージを送信するプロセス定義を作成する

「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。
   キャンバ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。



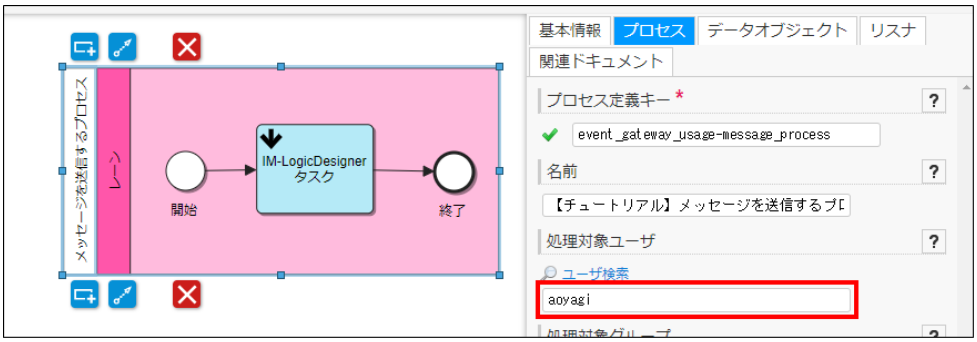
図：プロセス全体 - 「プロパティ」 - 「メッセージ&シグナル」

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」

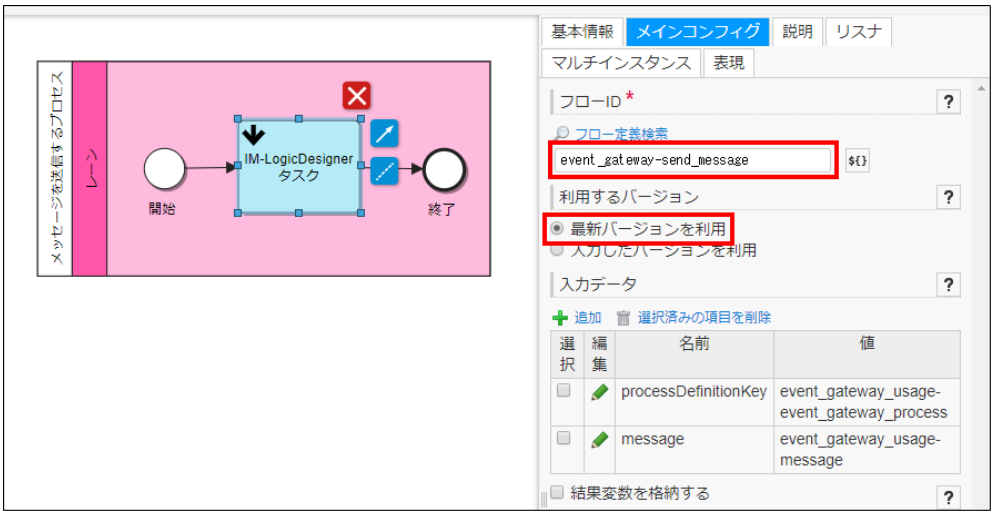
3. 「開始イベント」 を設置します。

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。



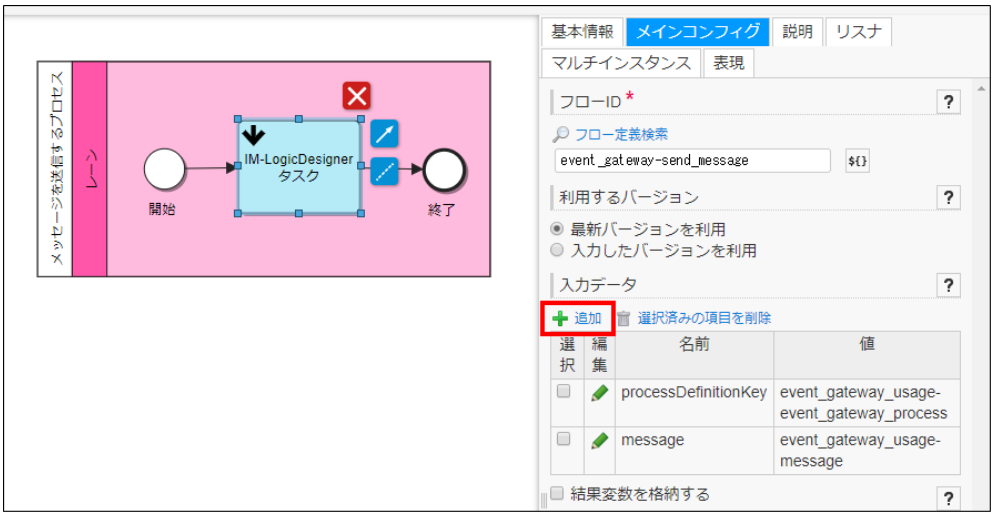
図： プロセス全体 - 「プロパティ」 - 「プロセス」

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」

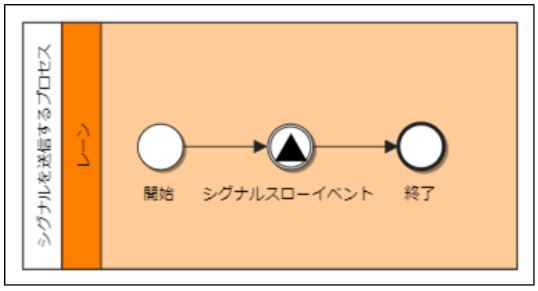
7. 以下のとおりに値を登録し、保存します。
   - 入力データ1
     - 名前：message
     - 値：event_gateway_usage-message
   - 入力データ2
     - 名前：processDefinitionKey
     - 値：event_gateway_usage-event_gateway_process

図：「入力データ」

8. 「終了イベント」を設置し、保存します。

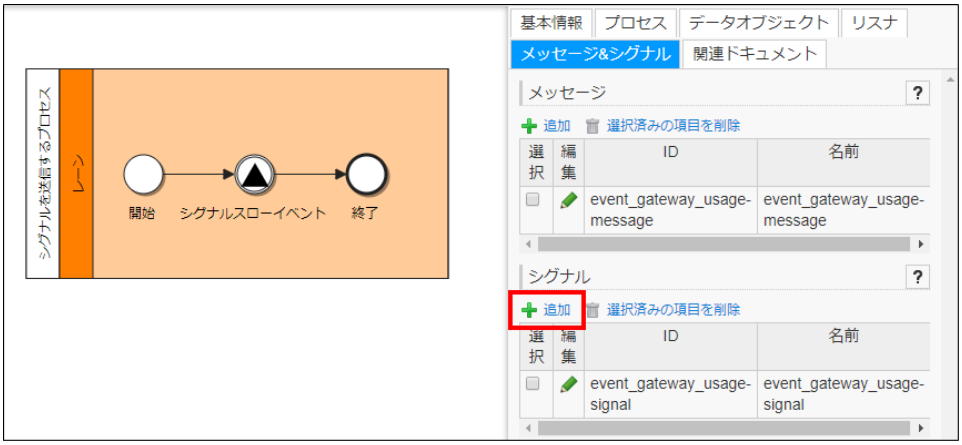
シグナルを送信するプロセス定義を作成する

「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。

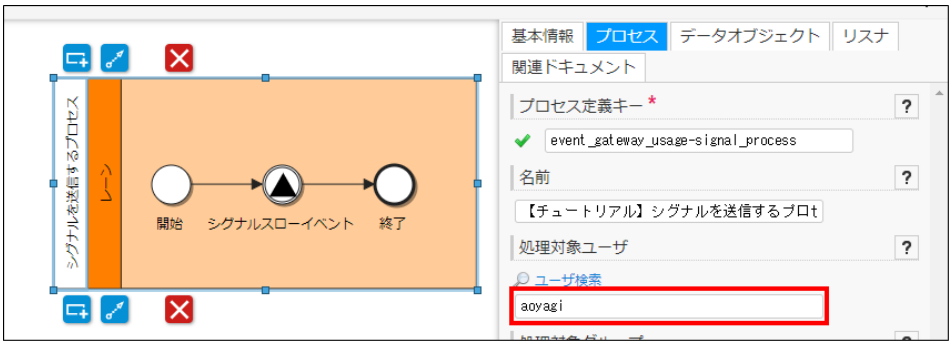


図：プロセス全体 -「プロパティ」 -「メッセージ&シグナル」

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」

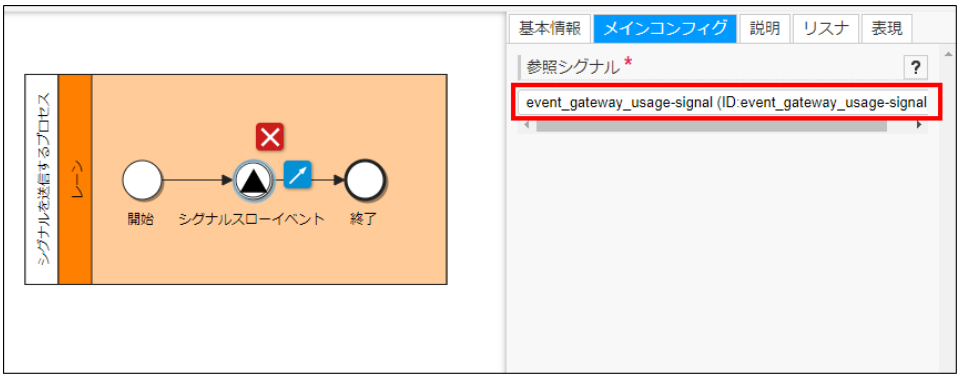
3. 「処理対象ユーザ」を設定します。
   プールをクリックし、「プロセス」タブから担当者を aoyagi に設定します。



図： プロセス全体 -「プロパティ」 -「プロセス」

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。

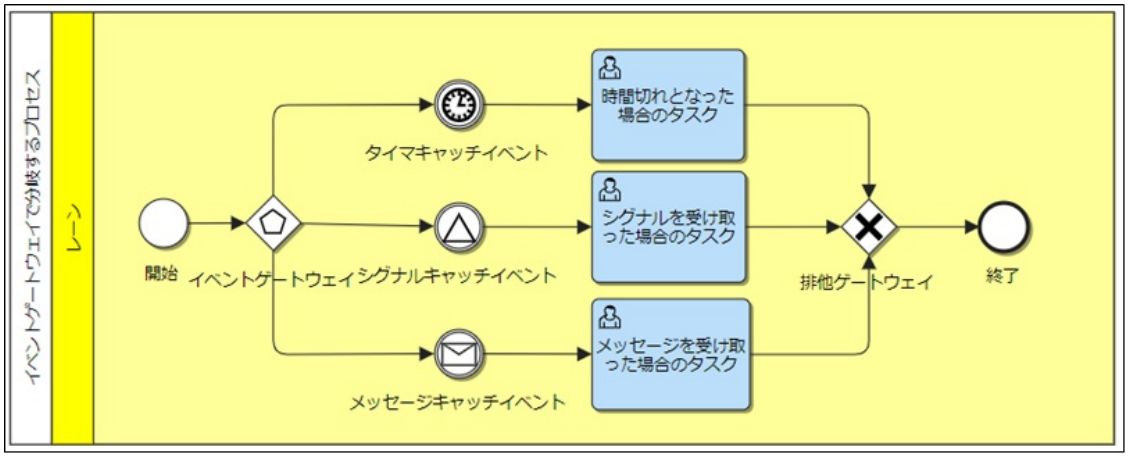


図：「シグナルスローイベント」-「プロパティ」 -「メインコンフィグ」

6. 「終了イベント」を設置し、保存します。

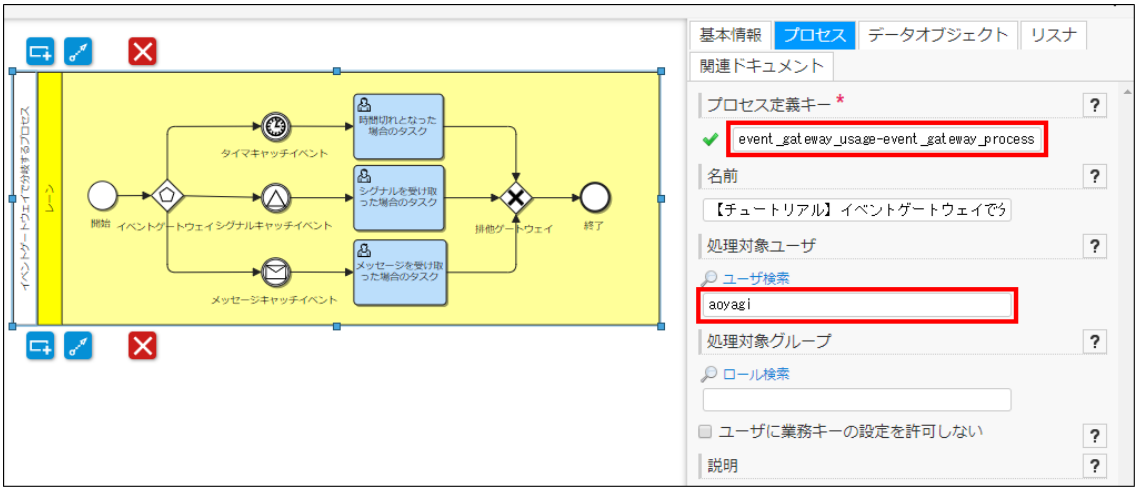
イベントゲートウェイで分岐するプロセス定義を作成する

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」

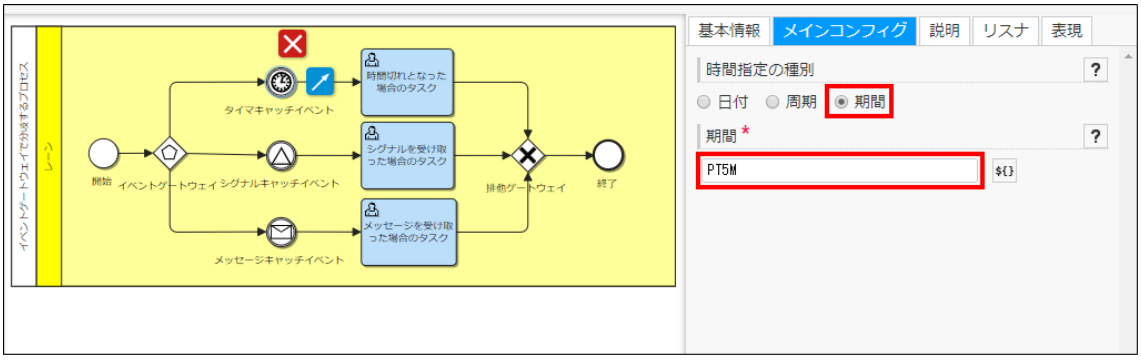
2. 「開始イベント」 を設置します。

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。

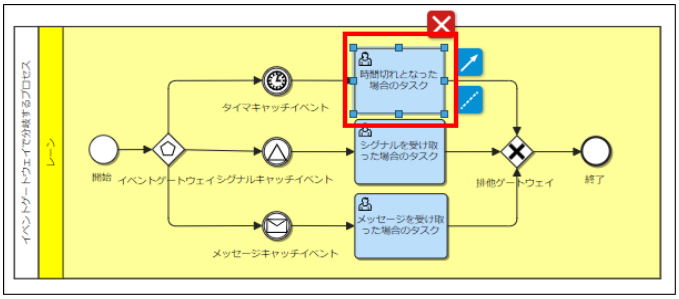
- 時間指定の種別：期間
- 期間：PT5M



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

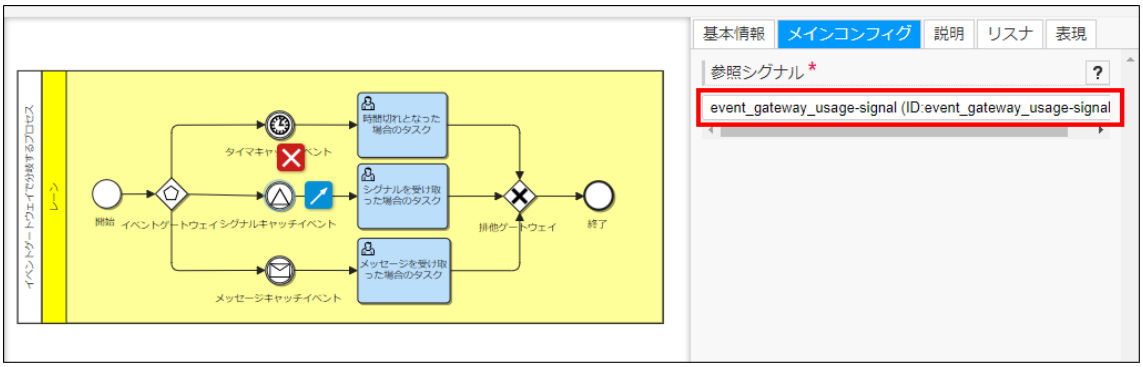
6. 「タイマキャッチイベント」が起動したことを確認するための「ユーザタスク」を設置します。



図：「ユーザタスク」

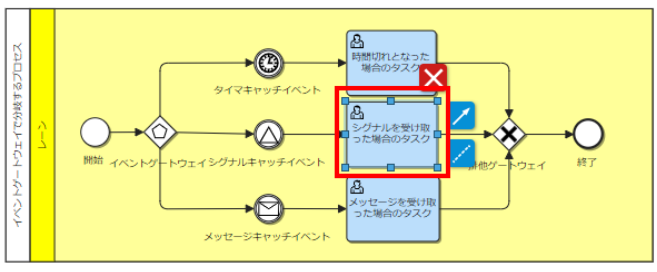
7. 2つ目の分岐である「シグナルキャッチイベント」を設置します。

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」

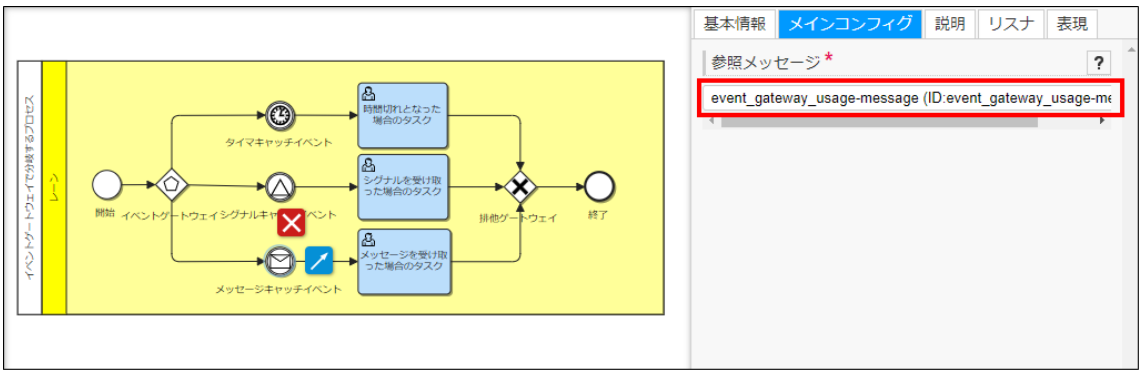
9. 「シグナルキャッチイベント」が起動したことを確認するための「ユーザタスク」を設置します。



図：「ユーザタスク」

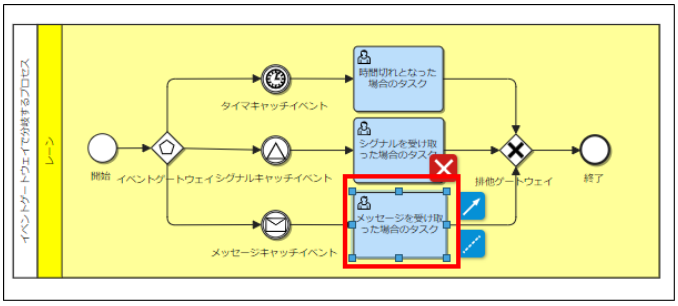
10. 3つ目の分岐である「メッセージキャッチイベント」を設置します。

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。

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、「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。

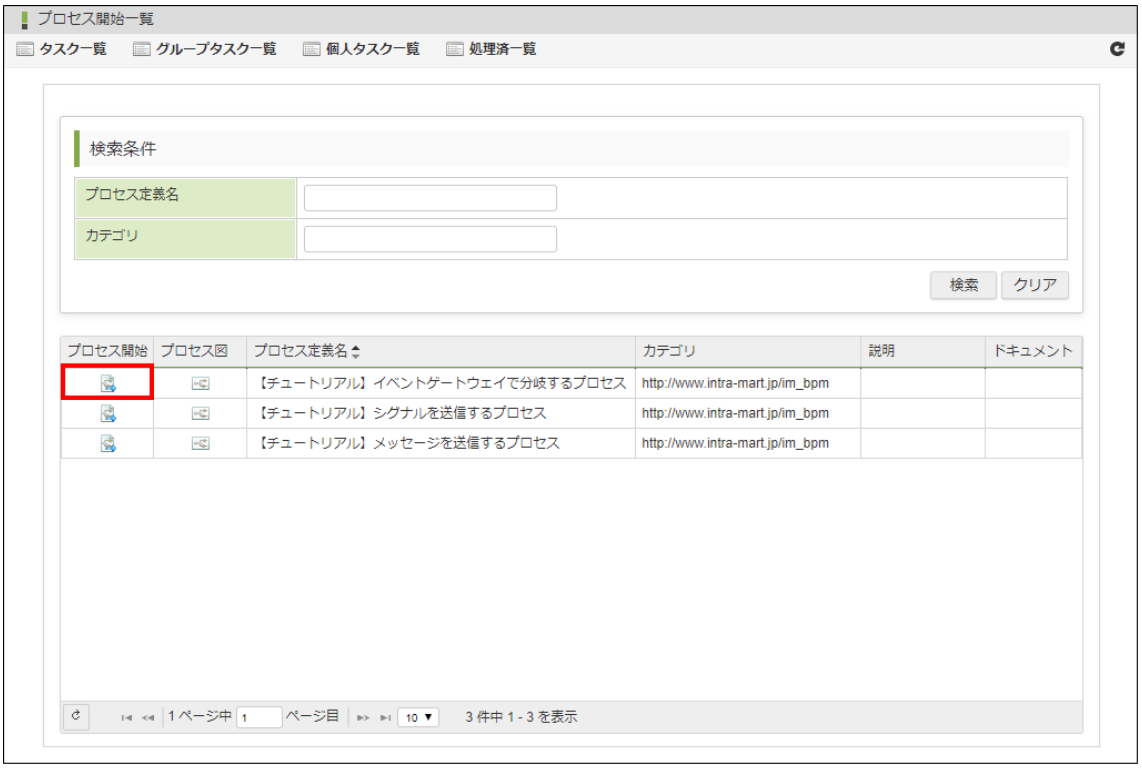


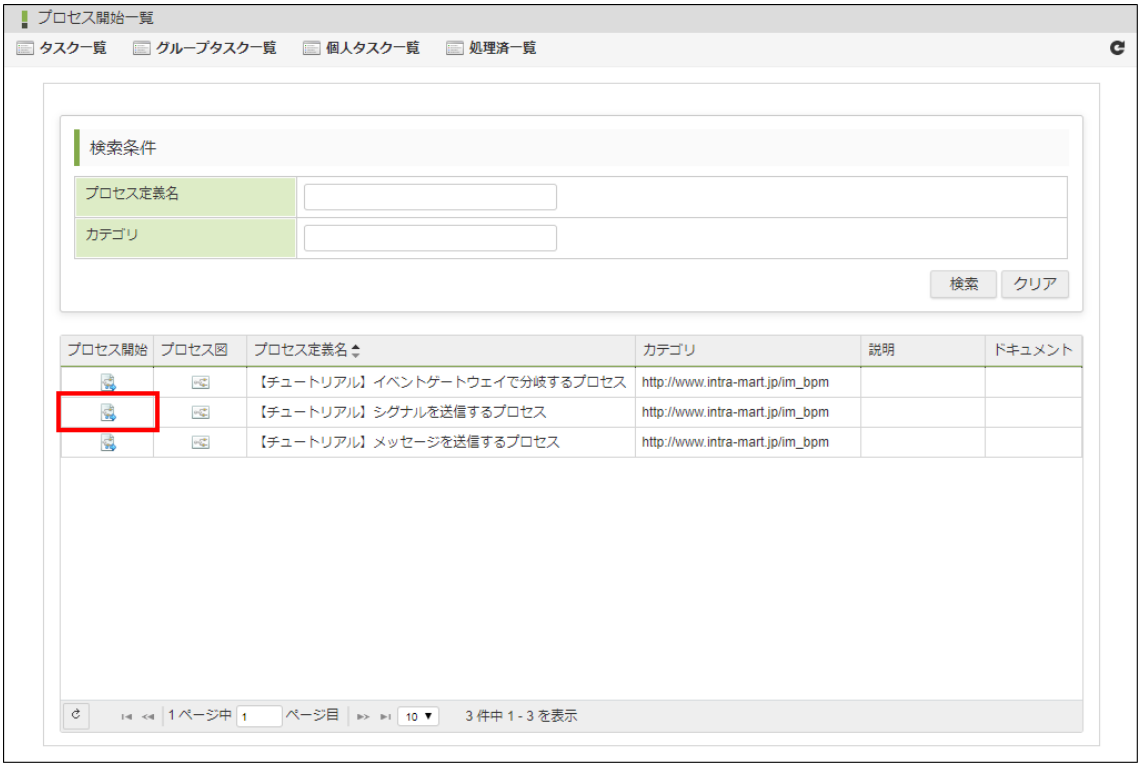
図: 「プロセス開始一覧」

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。

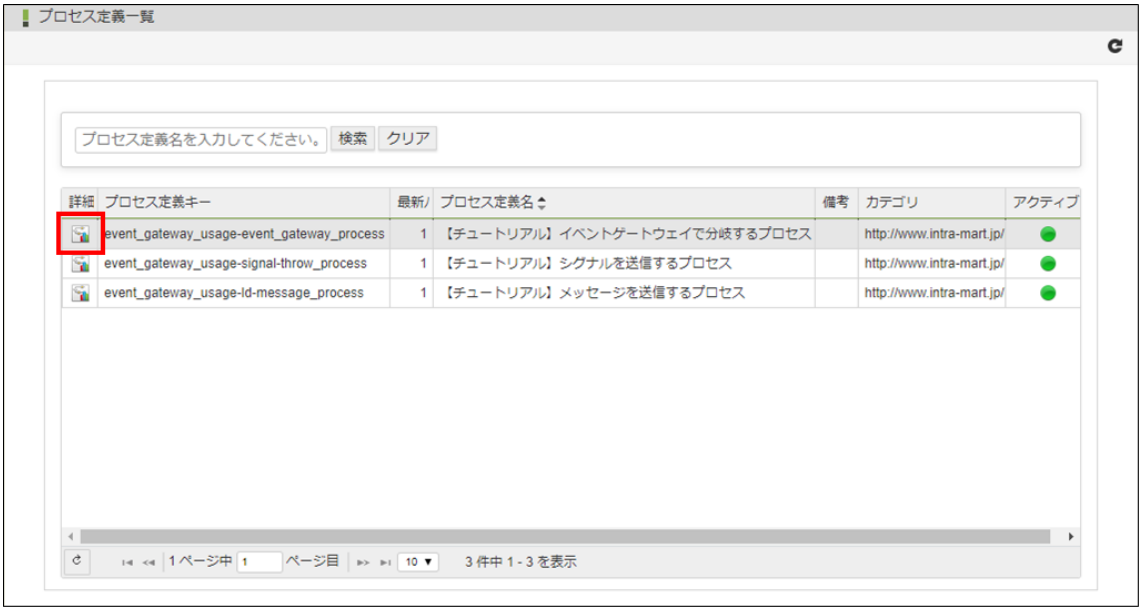
2. プロセス開始一覧から、「シグナルを送信するプロセス」の「 」アイコンをクリックします。



図：「プロセス開始一覧」

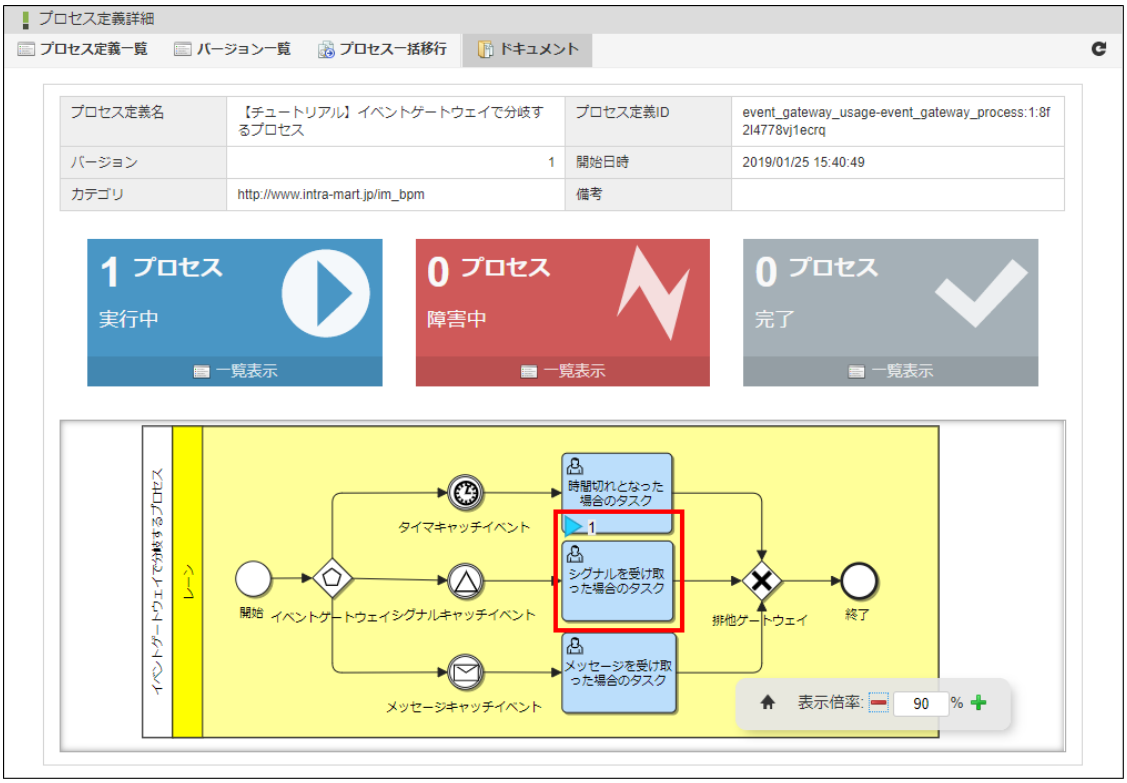
3. 「プロセス定義詳細」画面を表示します。

「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。

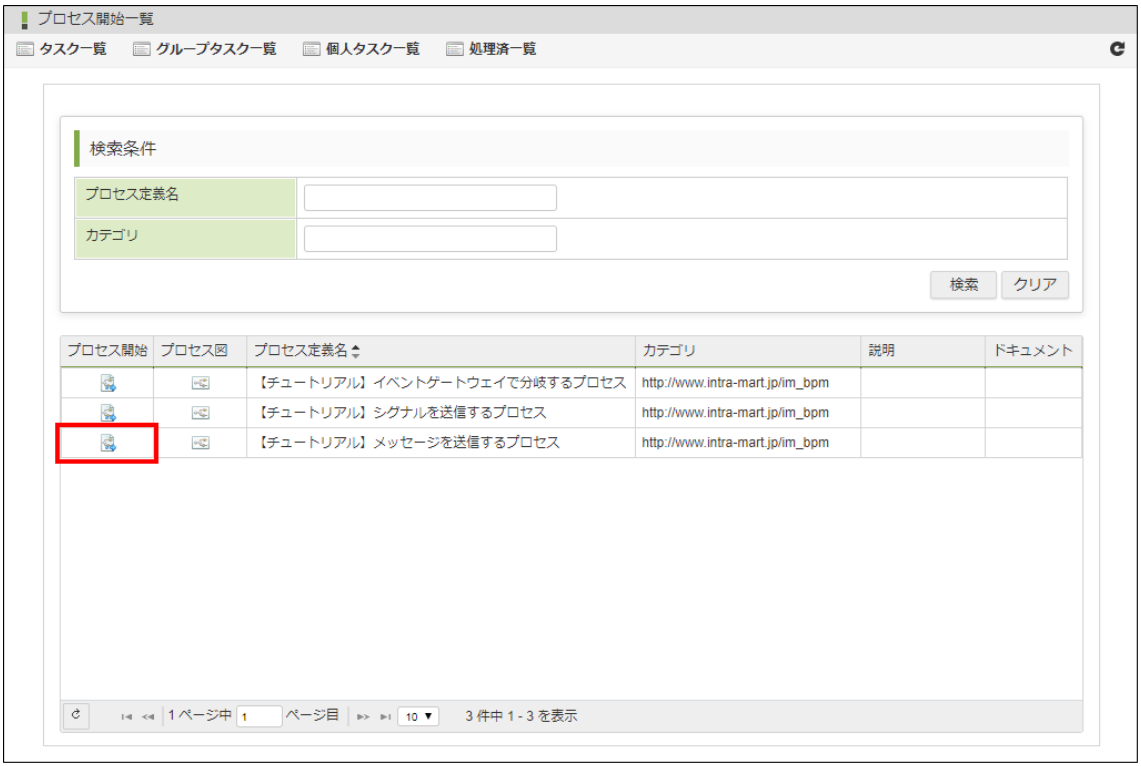


図：「プロセス定義詳細」

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。



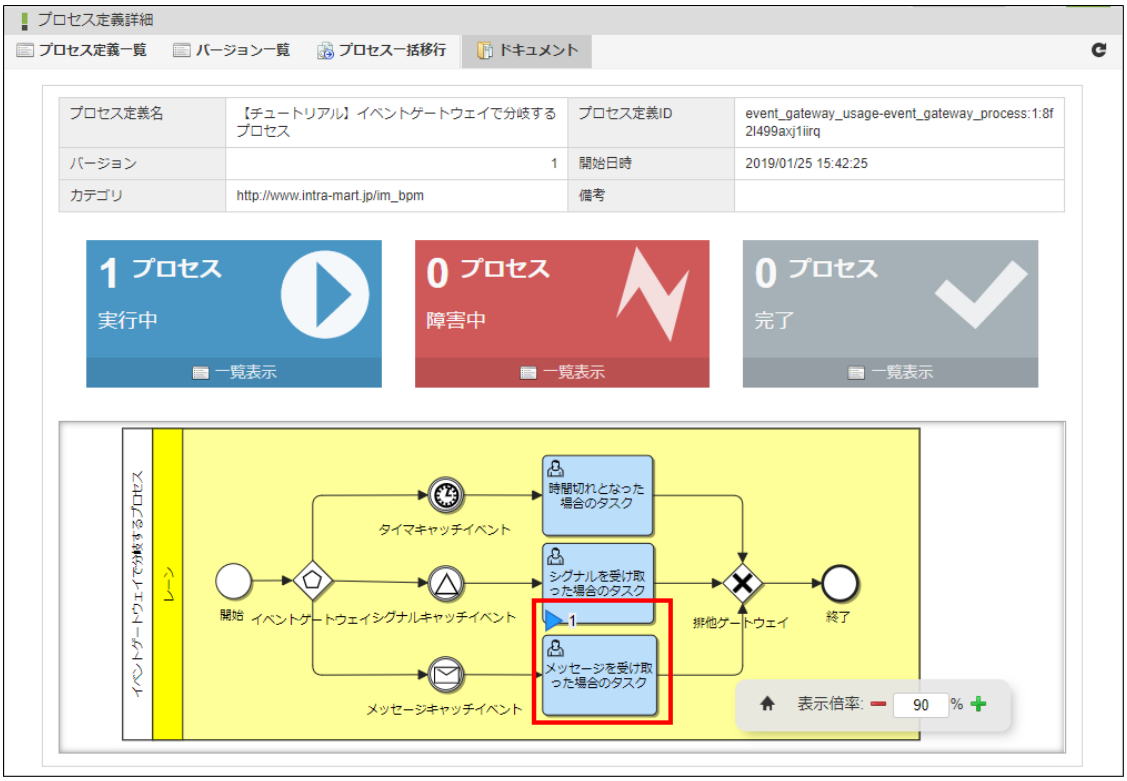
図：「プロセス開始一覧」

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」

タイマキャッチイベントへ分岐させる場合

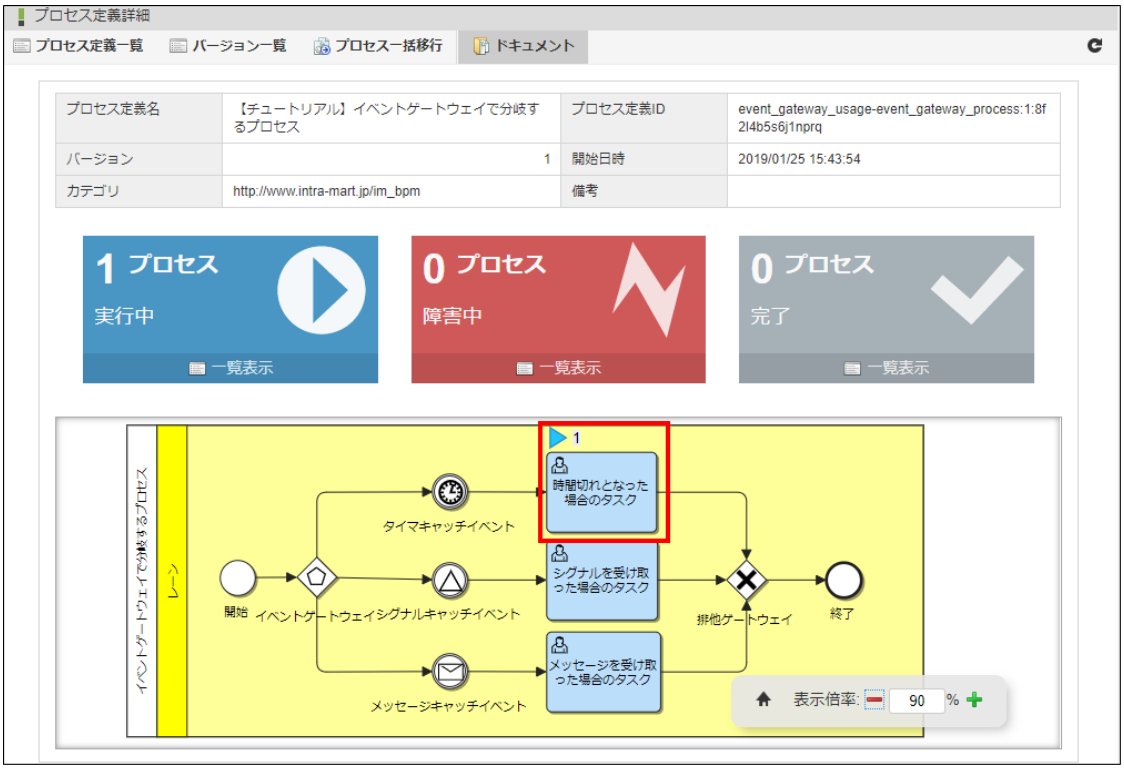
「タイマキャッチイベント」を受けて進行するタスクを確認します。
今回のタイマキャッチイベントは「 PT5M 」と設定したため、5分後にイベントゲートウェイから進行していることを確認します。

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

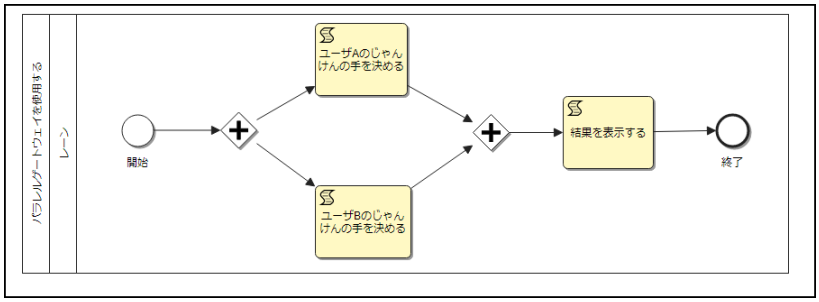
このチュートリアルでは、「パラレルゲートウェイ」を使用してプロセスの並列処理の開始や合流の定義方法を解説します。
「パラレルゲートウェイ」を使用することで、無条件で並列処理が行えます。
「パラレルゲートウェイ」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「ゲートウェイ」 - 「パラレルゲートウェイ」もあわせて参照してください。

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

- パラレルゲートウェイを使用し、並列処理を行う

パラレルゲートウェイを使用し、並列処理を行う

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

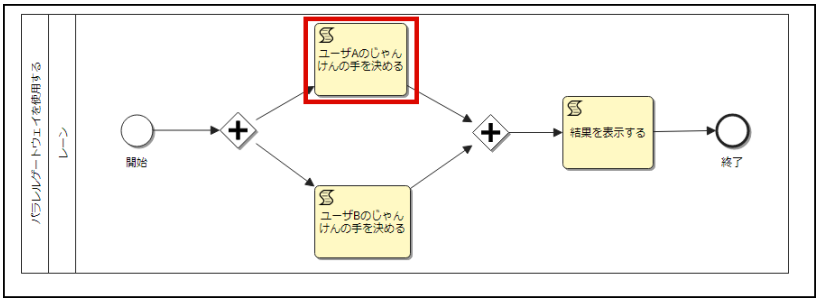


図：完成イメージ

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

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

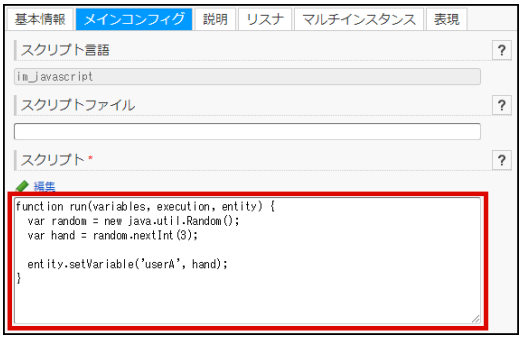
3. スクリプトタスク「ユーザAのじゃんけんの手を決める」 のプロパティを設定します。



図：スクリプトタスク「ユーザAのじゃんけんの手を決める」

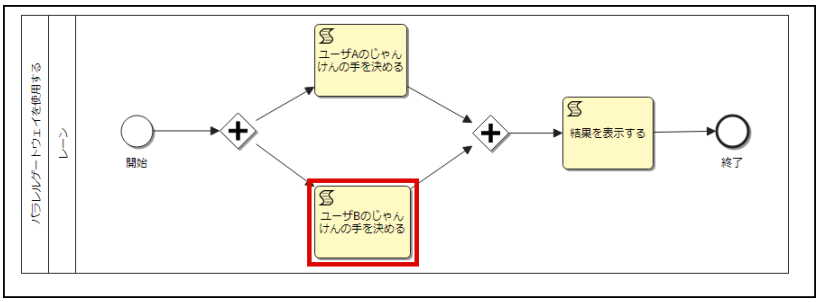
スクリプトタスク「ユーザAのじゃんけんの手を決める」にユーザAの手をランダムに選択し、変数 `userA` に格納するスクリプトを設定します。

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

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



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

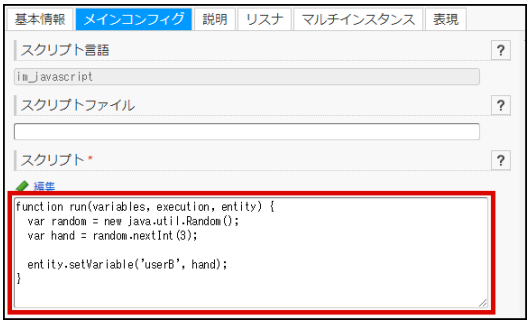
4. スクリプトタスク「ユーザBのじゃんけんの手を決める」 のプロパティを設定します。



図：スクリプトタスク「ユーザBのじゃんけんの手を決める」

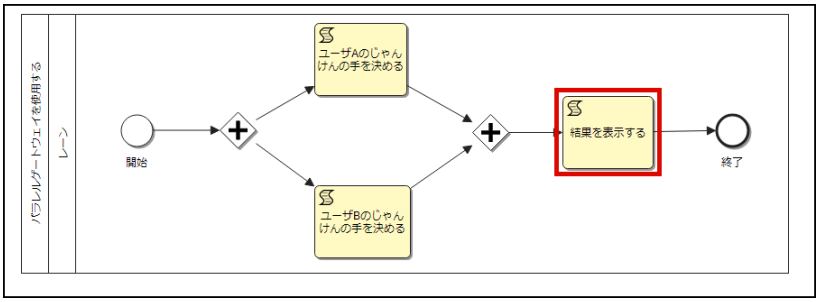
スクリプトタスク「ユーザBのじゃんけんの手を決める」にユーザBの手をランダムに選択し、変数 `userB` に格納するスクリプトを設定します。

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

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



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

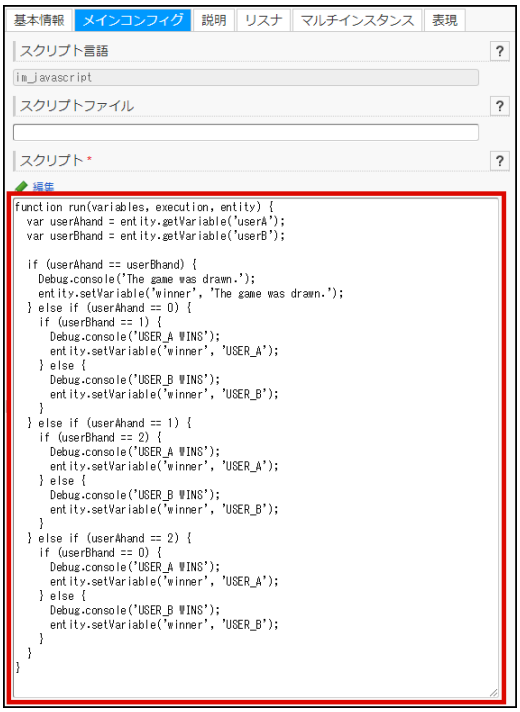
5. スクリプトタスク「結果を表示する」のプロパティを設定します。



図：スクリプトタスク「結果を表示する」

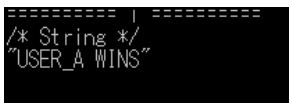
スクリプトタスク「結果を表示する」にユーザAとユーザBのじゃんけんの勝者を変数 `winner` に格納するスクリプトを設定します。

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

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

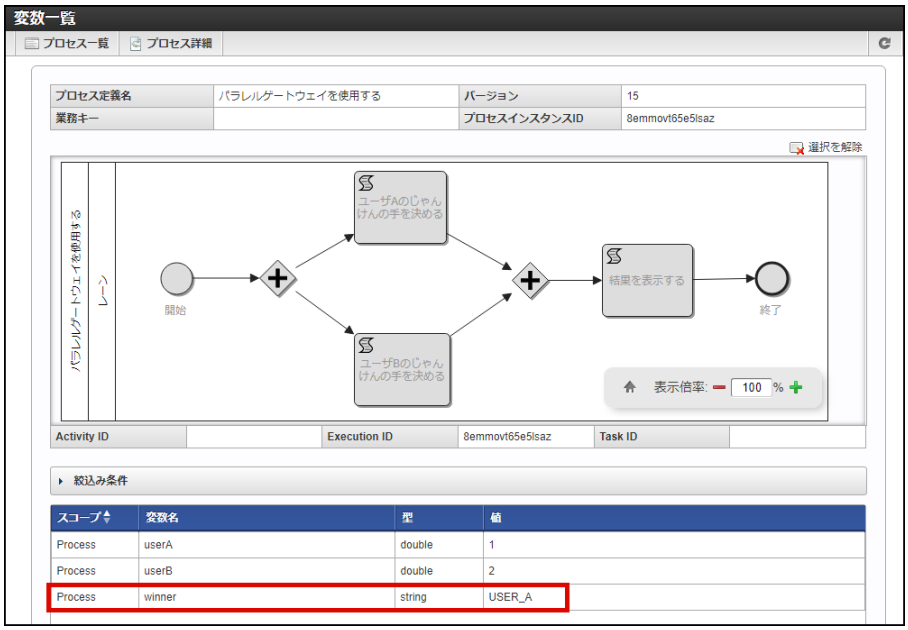


図：スクリプトタスク「結果を表示する」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

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



図：結果表示



図：変数一覧

排他ゲートウェイ

排他ゲートウェイを使用する

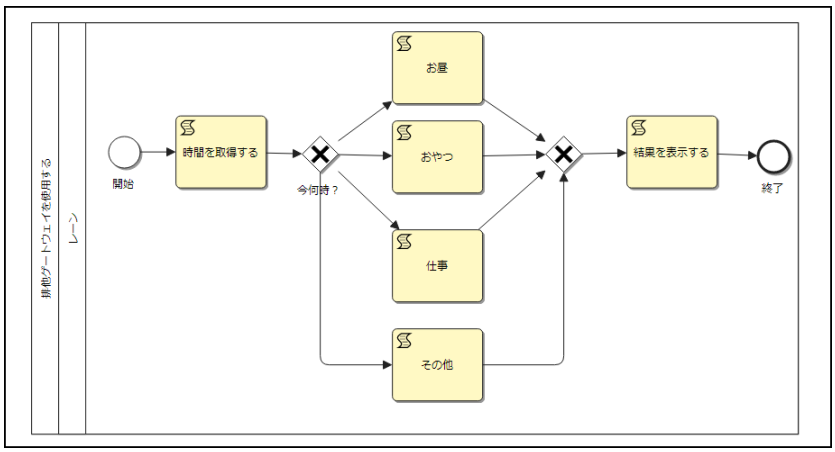
このチュートリアルでは、「排他ゲートウェイ」を使用してプロセスの遷移先の選択や合流の定義方法を解説します。
「排他ゲートウェイ」を使用することで、最初にtrueと評価された1つのシーケンスフローだけが継続して次の処理に進むフローを作成できます。
「排他ゲートウェイ」の詳細については、「IM-BPM プロセスデザイナ 操作ガイド」 - 「ゲートウェイ」 - 「排他ゲートウェイ」もあわせて参照してください。

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

- 排他ゲートウェイを使用し、複数の分岐先から1つの遷移先を選択する

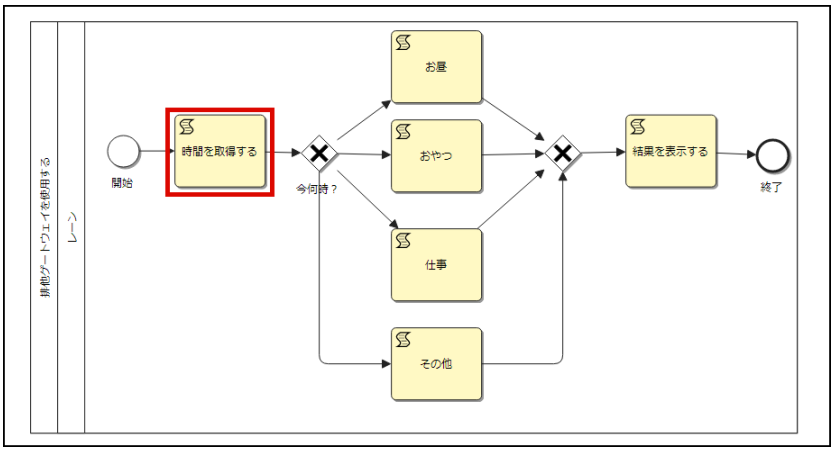
排他ゲートウェイを使用し、複数の分岐先から1つの遷移先を選択する

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



図：完成イメージ

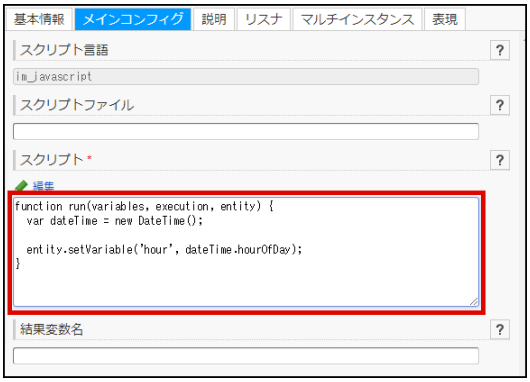
1. スクリプトタスク「時間を取得する」 のプロパティを設定します。



図：スクリプトタスク「時間を取得する」

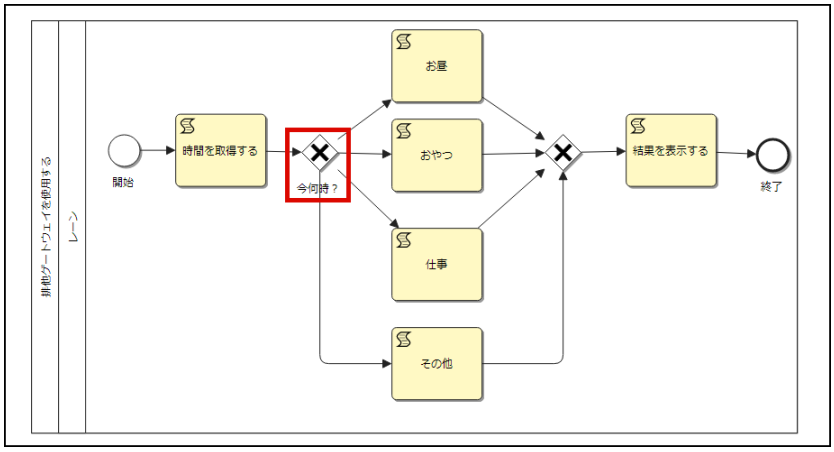
スクリプトタスク「時間を取得する」に、現在の時間を取得し変数 hour に格納するスクリプトを設定します。

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

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



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

2. 排他ゲートウェイのプロパティを設定します。

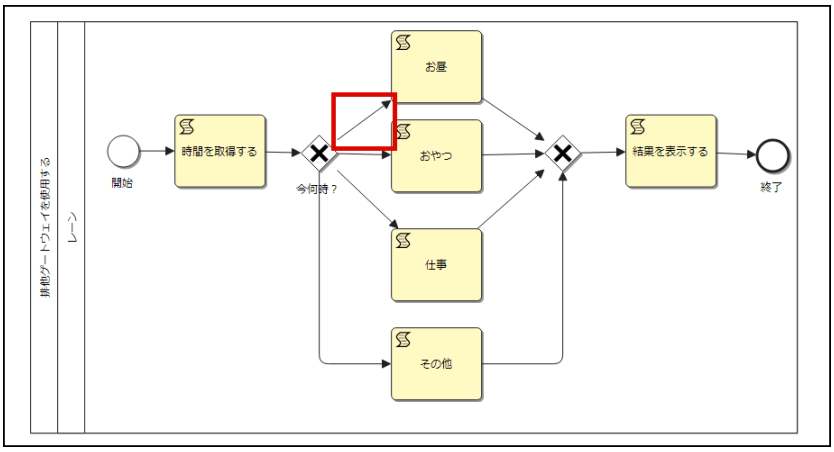


図：排他ゲートウェイ「今何時？」

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

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

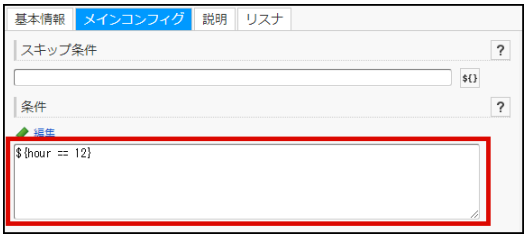
3. シーケンスフロー「sequence-flow_3」の条件を設定します。



図：シーケンスフロー「sequence-flow_3」

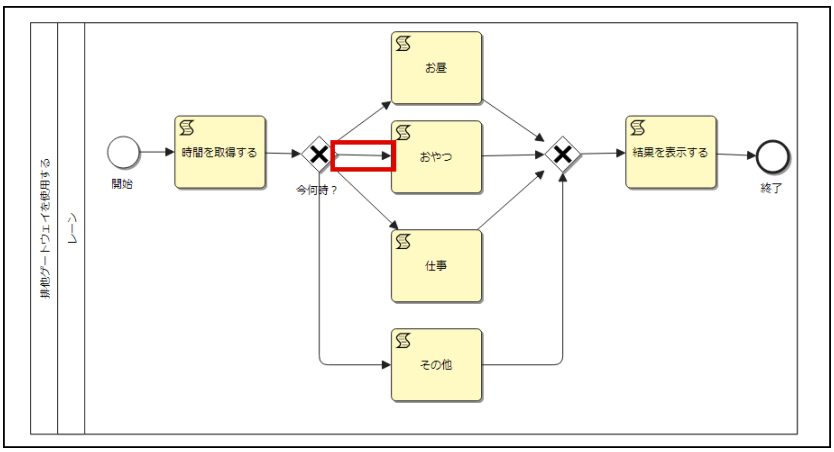
排他ゲートウェイ「今何時？」とスクリプトタスク「お昼」を接続しているシーケンスフロー「sequence-flow_3」 に、変数 `hour` が `12` の場合に真となる条件を設定します。

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



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

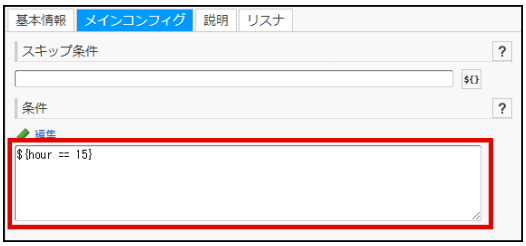
4. シーケンスフロー「sequence-flow_4」の条件を設定します。



図：シーケンスフロー「sequence-flow_4」

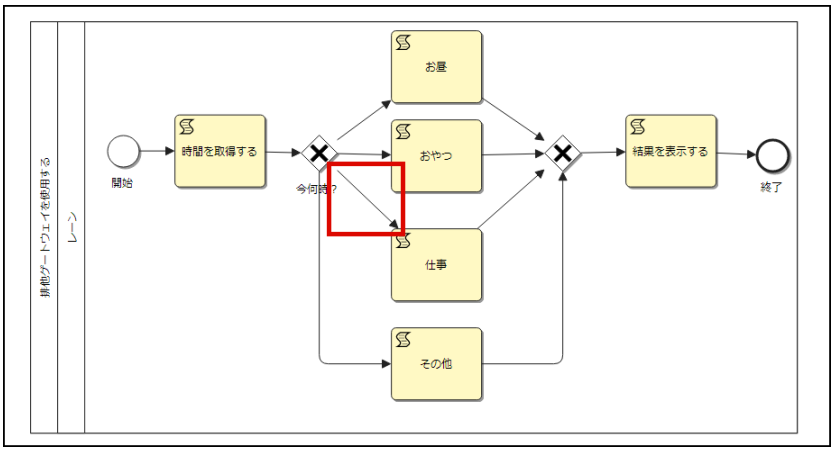
排他ゲートウェイ「今何時？」とスクリプトタスク「おやつ」を接続しているシーケンスフロー「sequence-flow_4」 に、変数 `hour` が `15` の場合に真となる条件を設定します。

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



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

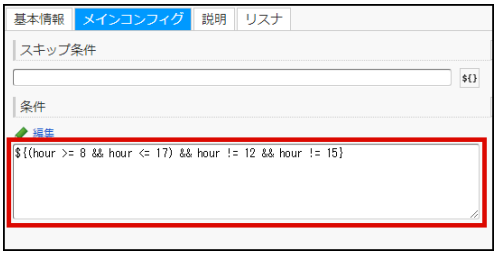
5. シーケンスフロー「sequence-flow_5」の条件を設定します。



図：シーケンスフロー「sequence-flow_5」

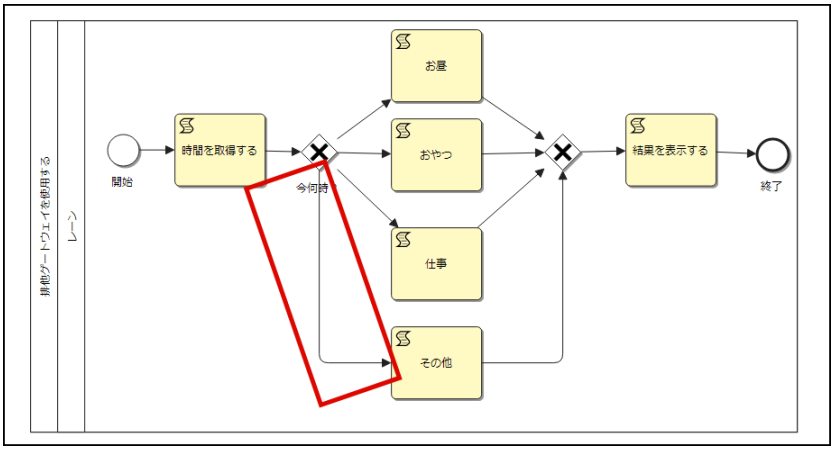
排他ゲートウェイ「今何時？」とスクリプトタスク「仕事」を接続しているシーケンスフロー「sequence-flow_5」 に、変数 `hour` が `8` 以上かつ `17` 以下で、 `12` ではなく、 `15` でもない場合に真となる条件を設定します。

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



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

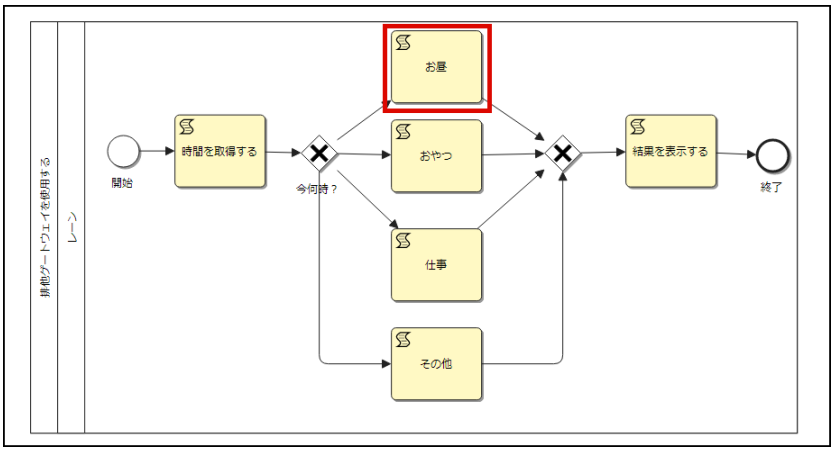
6. シーケンスフロー「sequence-flow_6」の条件を設定します。



図：スクリプトタスク「sequence-flow_6」

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

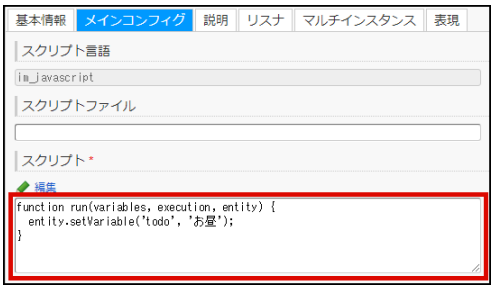
7. スクリプトタスク「お昼」 のプロパティを設定します。



図：スクリプトタスク「お昼」

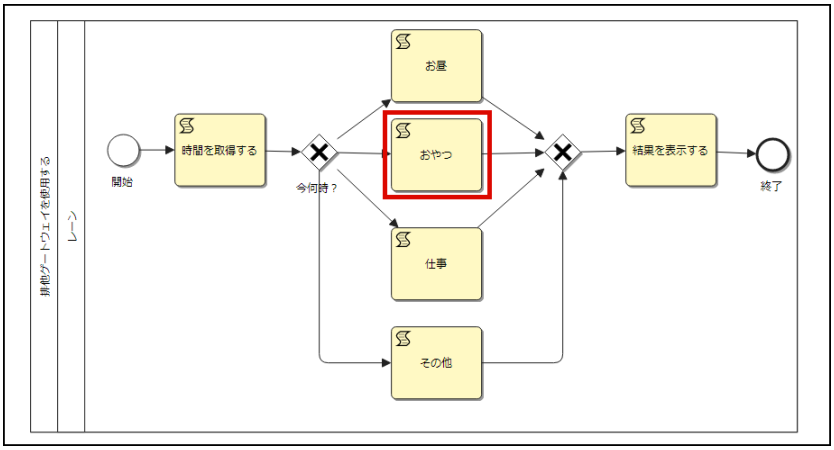
スクリプトタスク「お昼」のプロパティにて、作業名 お昼 を変数 todo に格納するスクリプトを設定します。

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



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

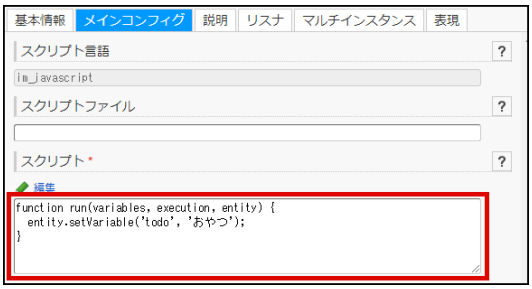
8. スクリプトタスク「おやつ」 のプロパティを設定します。



図：スクリプトタスク「おやつ」

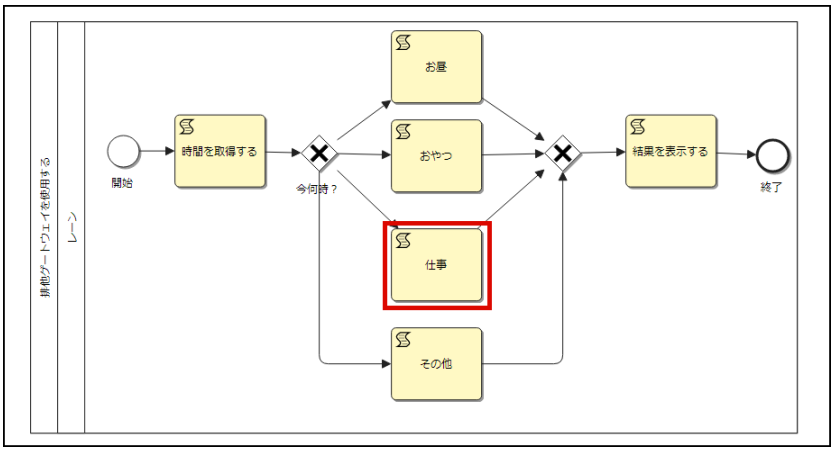
スクリプトタスク「おやつ」のプロパティにて、作業名 おやつ を変数 todo に格納するスクリプトを設定します。

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



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

9. スクリプトタスク「仕事」 のプロパティを設定します。



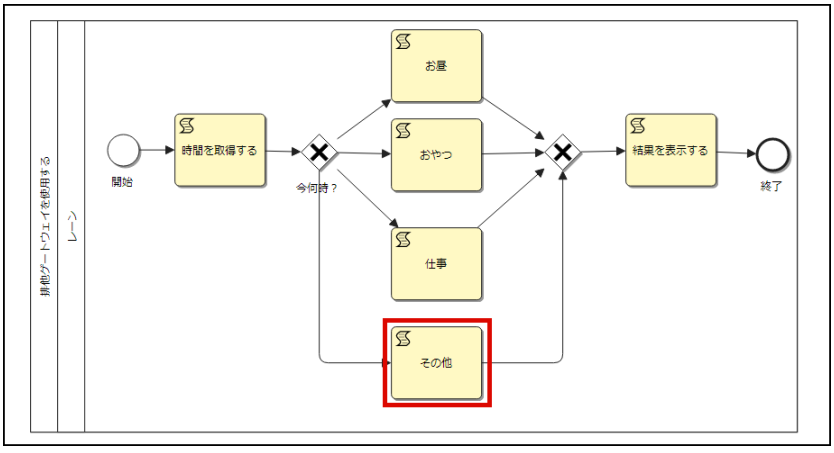
図：スクリプトタスク「仕事」

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

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

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

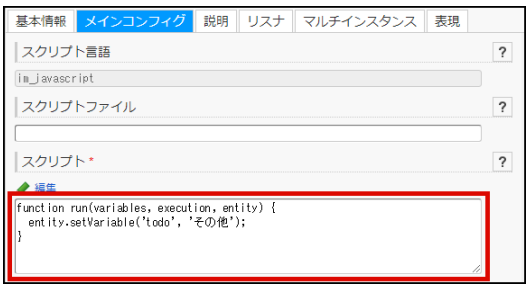
10. スクリプトタスク「その他」 のプロパティを設定します。



図：スクリプトタスク「その他」

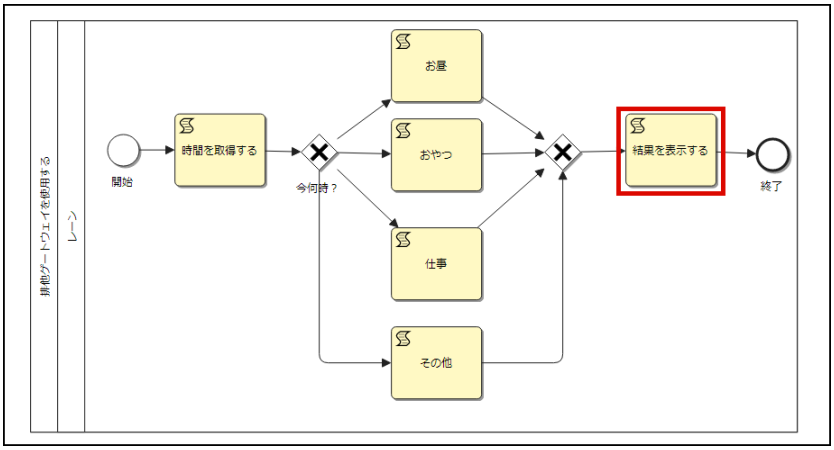
スクリプトタスク「その他」のプロパティにて、作業名 その他 を変数 todo に格納するスクリプトを設定します。

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



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

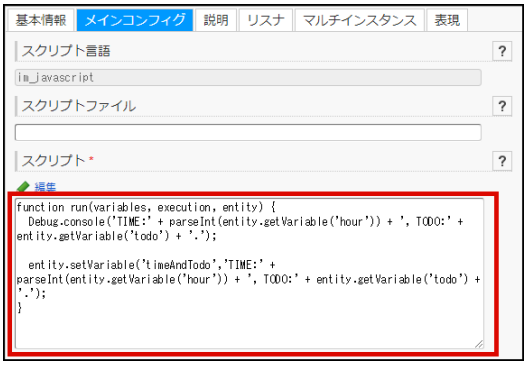
11. スクリプトタスク「結果を表示する」のプロパティを設定します。



図：スクリプトタスク「結果を表示する」

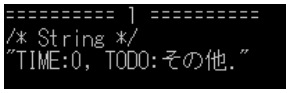
スクリプトタスク「結果を表示する」に、変数 hour と変数 todo を組み合わせ、変数 timeAndTodo に格納するスクリプトを設定します。

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

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

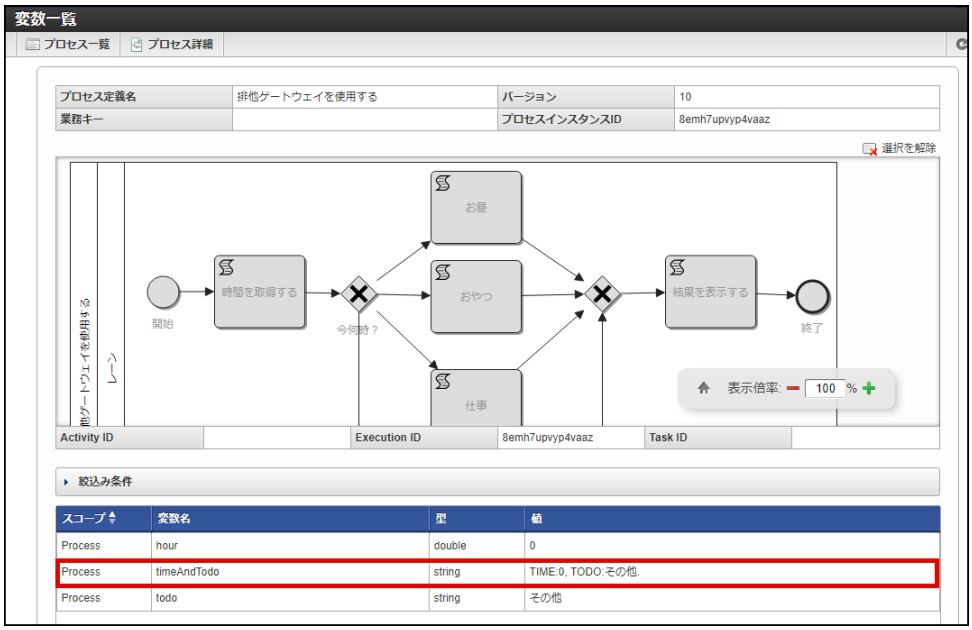


図：スクリプトタスク「結果を表示する」の「プロパティエリア」-「メインコンフィグ」-「スクリプト」

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



図：結果表示



図：変数一覧

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

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

包括ゲートウェイ

包括ゲートウェイを使用する

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

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

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

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

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

コラム

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

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

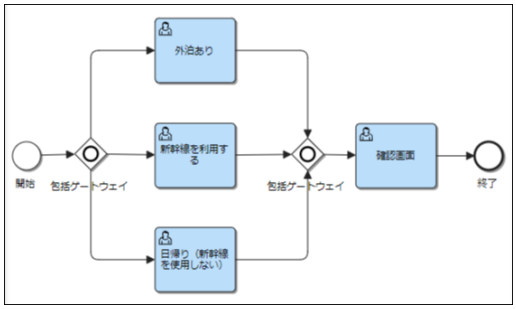
- プロセス定義を作成する
- 結果を確認する

プロセス定義を作成する

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

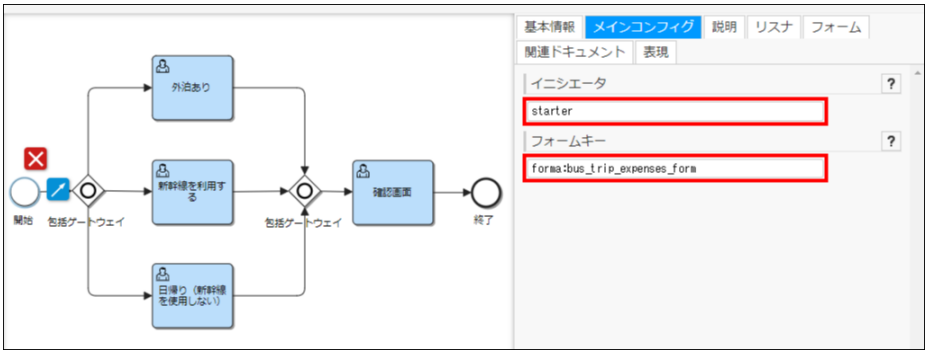
- 出張経費に「外泊費」を含めて申請する
- 出張経費に「新幹線の交通費」を含めて申請する
- 上記二つが当てはまらない申請をする

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



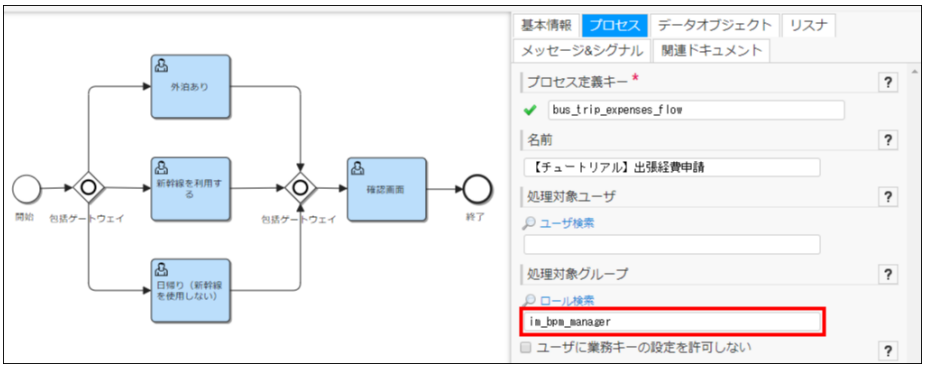
図：完成イメージ

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



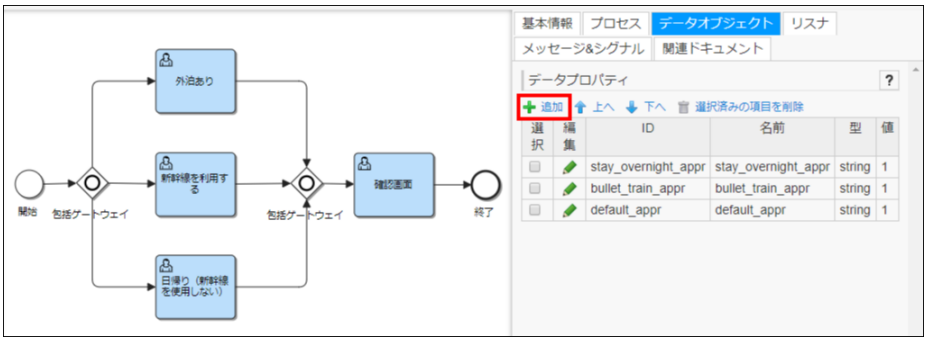
図：「開始イベント」 -「プロパティ」 -「メインコンフィグ」

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



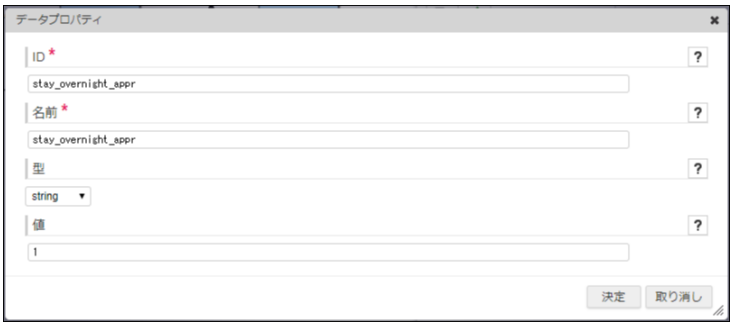
図：プロセス全体 - 「プロパティ」 - 「プロセス」

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



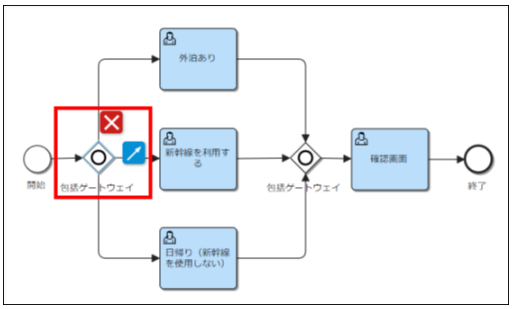
図：プロセス全体 - 「プロパティ」- 「データオブジェクト」

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



図：「データプロパティ」

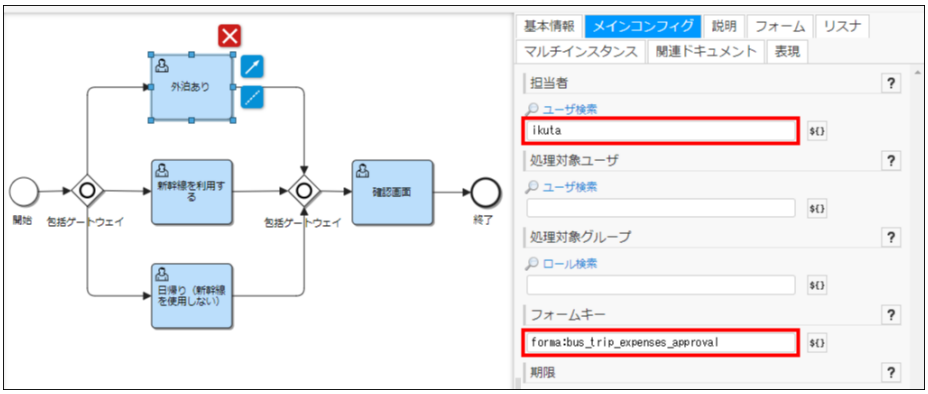
7. 「包括ゲートウェイ」を設置します。



図：「包括ゲートウェイ」

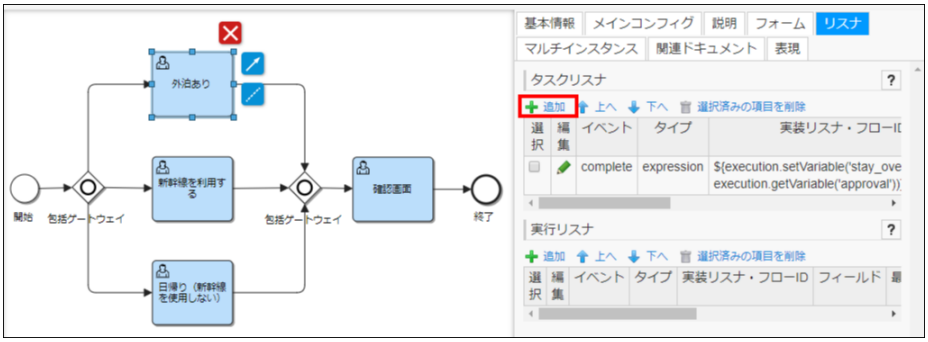
8. 「外泊あり」の出張経費申請の承認を「生田」に対して依頼するタスクを作成します。
   「承認画面」の「ユーザタスク」を設置します。

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



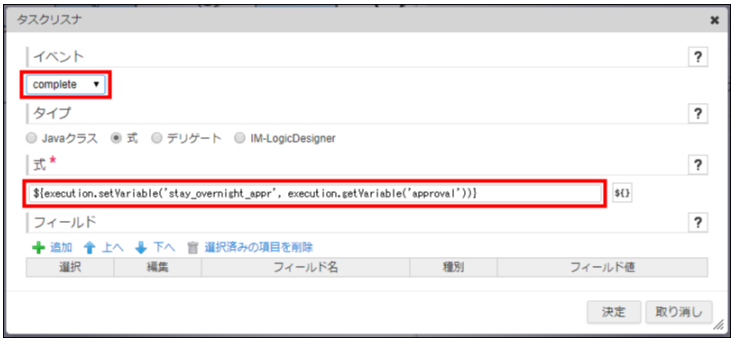
図：「ユーザタスク」 - 「プロパティ」- 「メインコンフィグ」

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



図：「ユーザタスク」 - 「プロパティ」- 「リスナ」

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

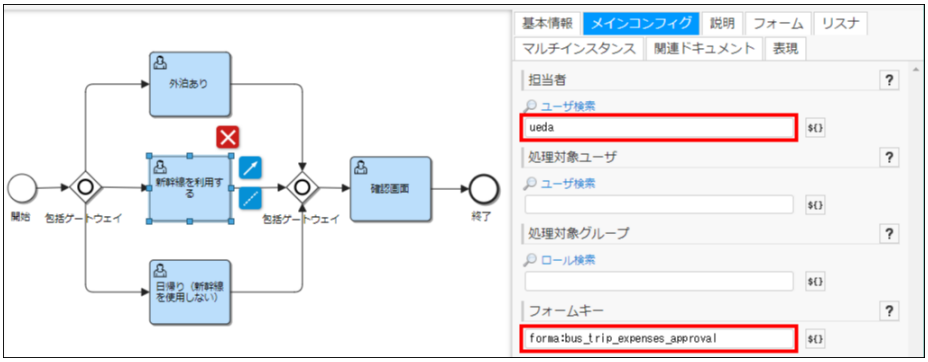


図：「タスクリスナ」

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

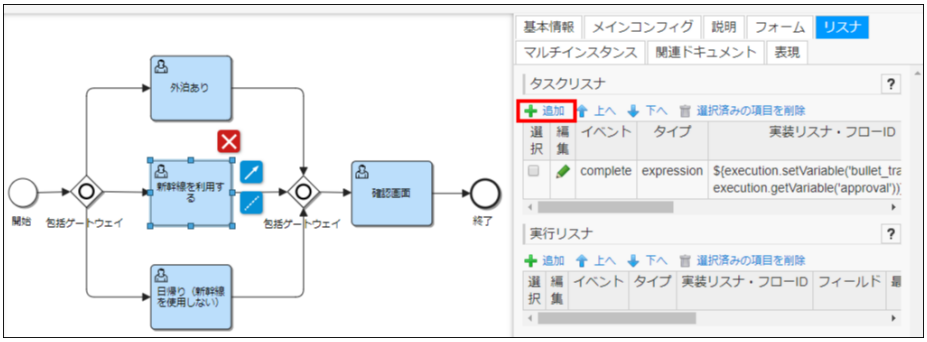
「承認画面」の「ユーザタスク」を設置します。

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



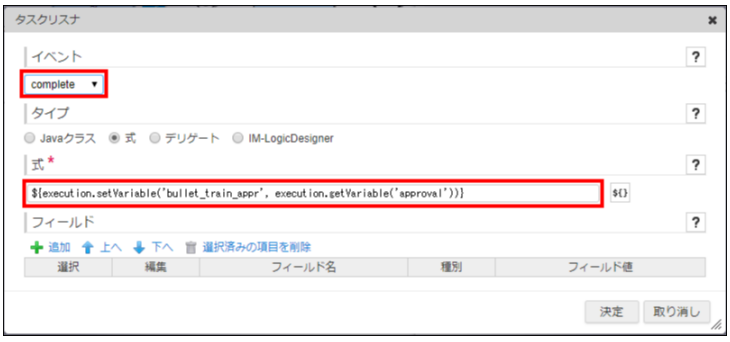
図：「ユーザタスク」 - 「プロパティ」- 「メインコンフィグ」

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



図：「ユーザタスク」 - 「プロパティ」- 「リスナ」

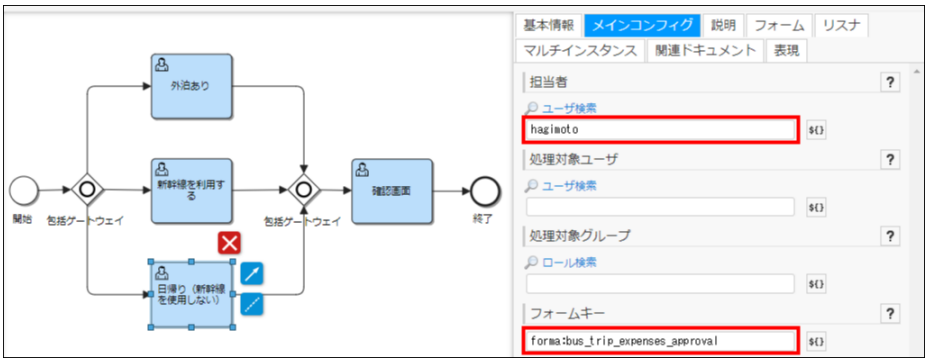
15. 「タスクリスナ」ダイアログで、以下のとおりに項目を設定します。
    - イベント ： complete
    - タイプ ： 式
    - 式 ： ${execution.setVariable('bullet_train_appr', execution.getVariable('approval'))}



図：「タスクリスナ」

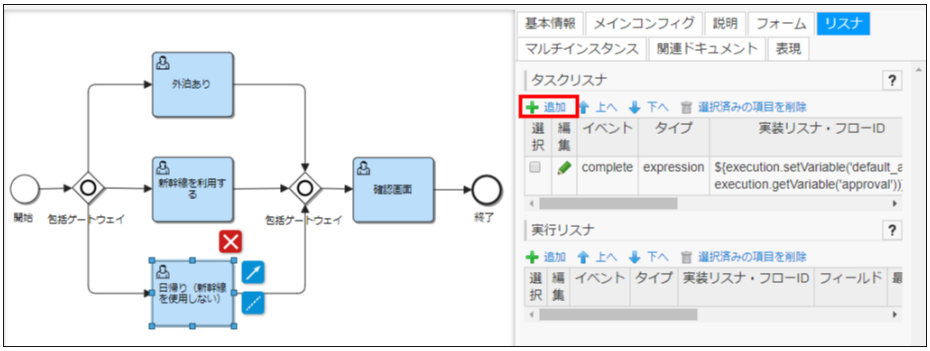
16. 「外泊あり」と「新幹線を利用する」が選択されなかった申請の場合、「萩本」に対して承認を依頼するタスクを配置します。
「承認画面」の「ユーザタスク」を設置します。

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



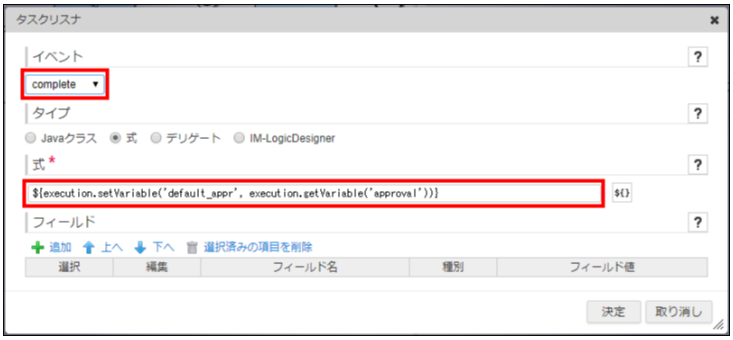
図：「ユーザタスク」 - 「プロパティ」- 「メインコンフィグ」

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



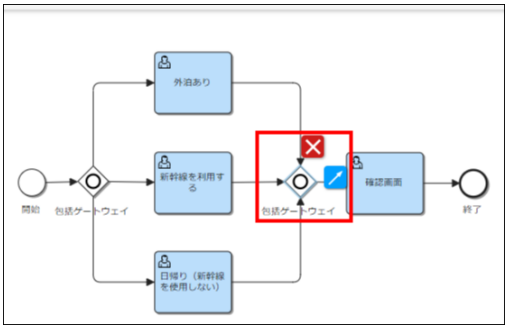
図：「ユーザタスク」 - 「プロパティ」- 「リスナ」

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



図：「タスクリスナ」

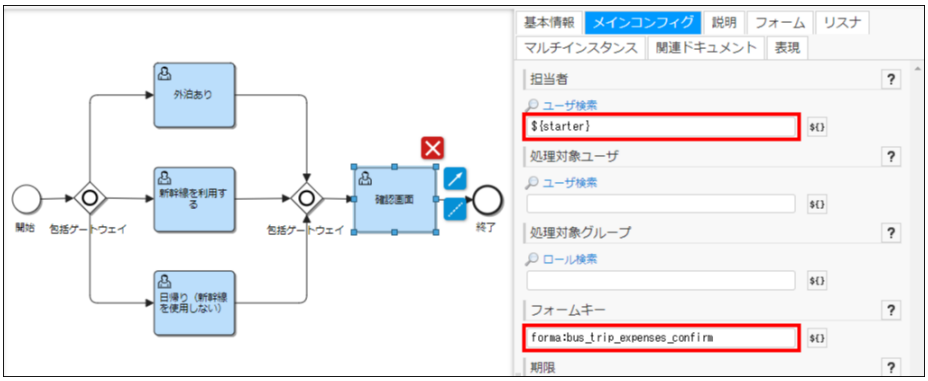
20. 他の承認の判定を待つため「包括ゲートウェイ」を設置します。



図：「包括ゲートウェイ」

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

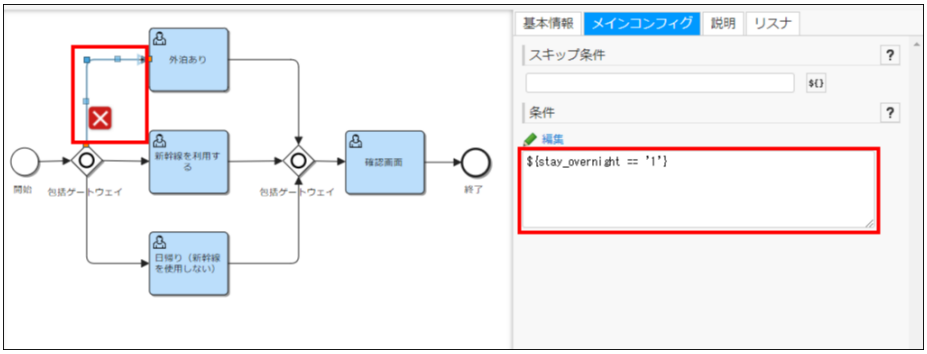
22. 開始イベントに設定した「イニシエータ」の値を使用し、申請者に対して「確認画面」を送信します。
「ユーザタスク」を選択し、「プロパティ」の「メインコンフィグ」から以下のとおりに項目を設定します。
   - 担当者 ： ${starter}
   - フォームキー ： forma:bus_trip_expenses_confirm



図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

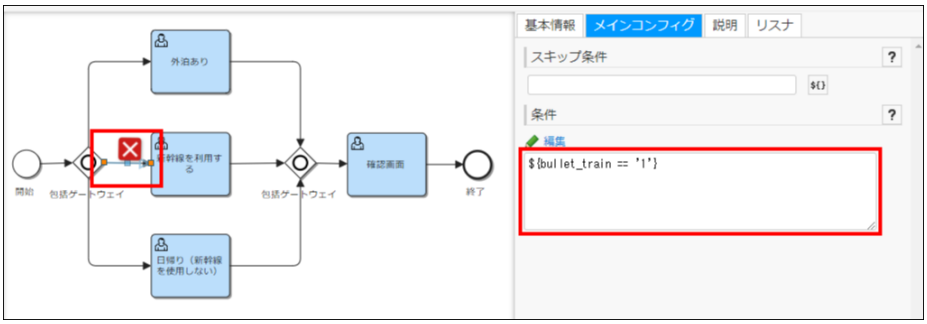
23. 終了イベントを設置します。

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



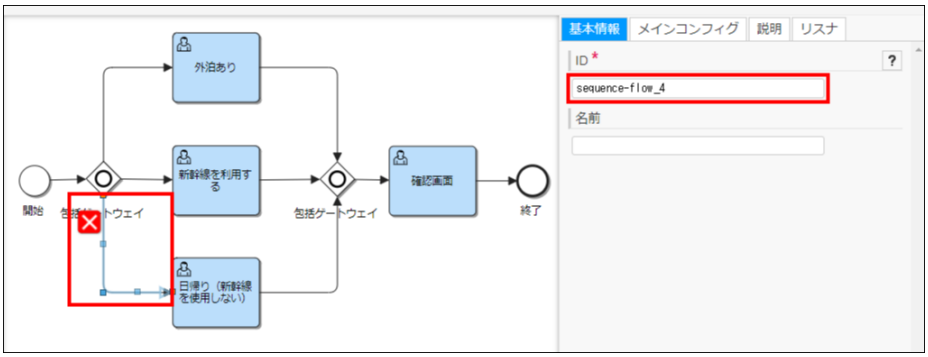
図：「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」

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



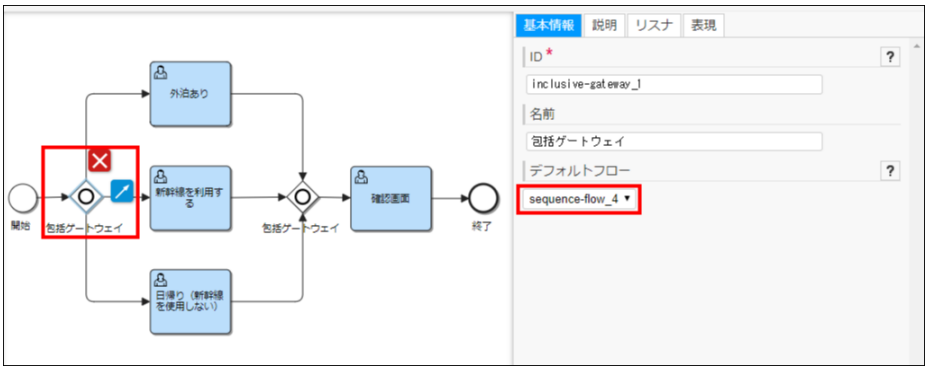
図：「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」

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



図：「シーケンスフロー」 - 「プロパティ」 - 「基本情報」

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



図：「包括ゲートウェイ」 -「プロパティ」 -「基本情報」

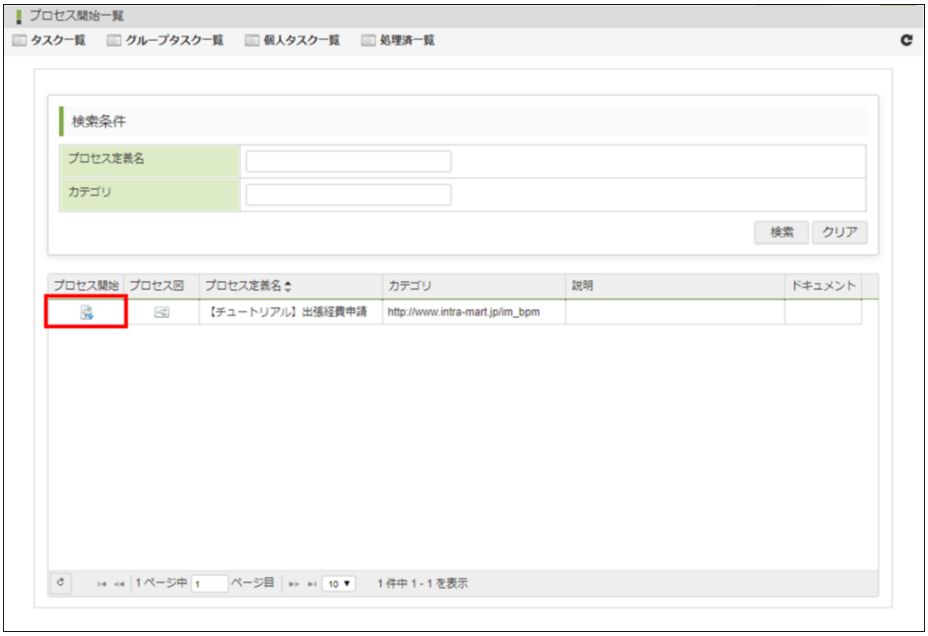
結果を確認する

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

デフォルトフローの結果を確認する

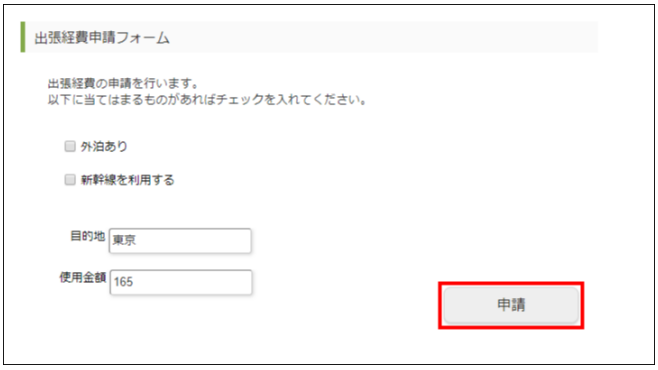
「包括ゲートウェイ」から、デフォルトフローへ進行する場合を確認します。

1. 「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示します。
2. プロセス開始一覧から、「 」アイコンをクリックします。



図：「プロセス開始一覧」

3. 「IM-FormaDesigner」で作成した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「 」をクリックします。

図：「タスク一覧」 -「個人タスク」

7. 「IM-FormaDesigner」で作成した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。

図：承認画面（IM-FormaDesigner）

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

10. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。

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

図：「タスク一覧」 -「個人タスク」

11. 「IM-FormaDesigner」で作成した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）

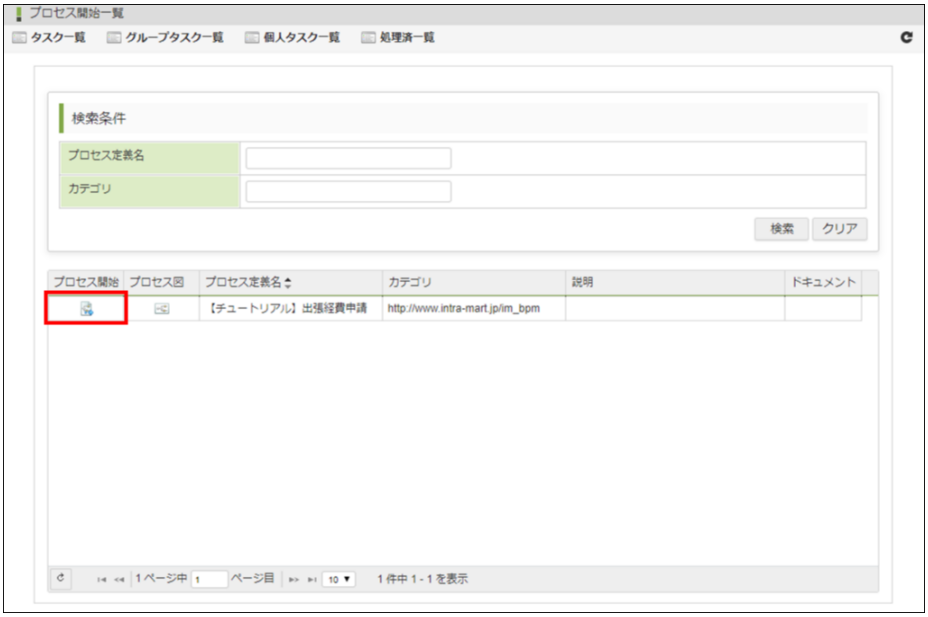
12. デフォルトフローに進行した場合、ほかのタスクは処理されません。
「外泊あり」の担当者である「生田」、「新幹線を利用する」の担当者である「上田」の個人タスク一覧に、依頼がないことを確認します。

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。

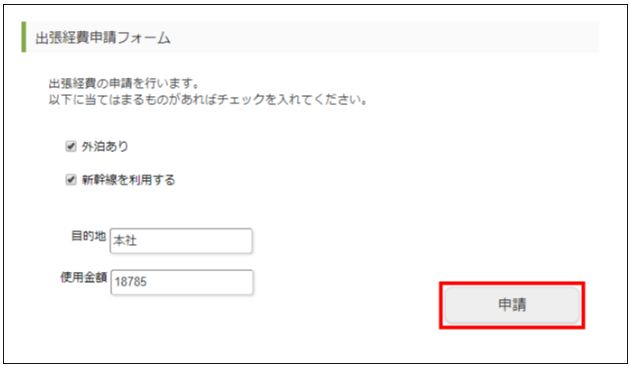
2. プロセス開始一覧から、「 」アイコンをクリックします。



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。

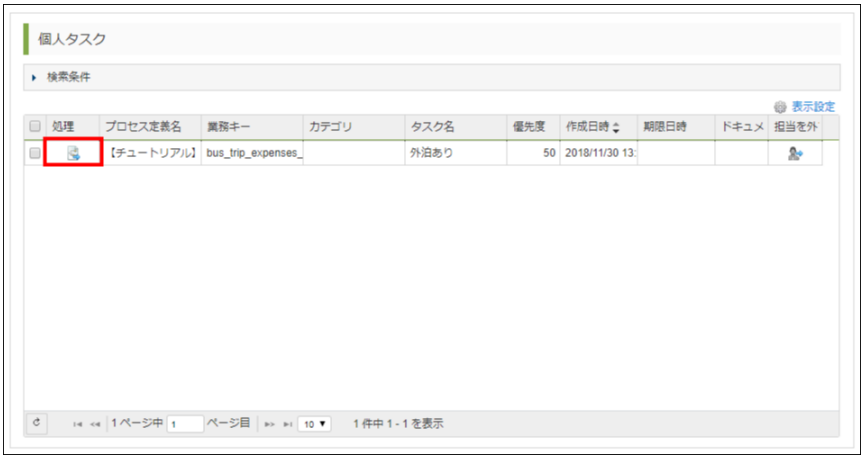


図：申請画面（IM-FormaDesigner）

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「 」をクリックします。



図：「タスク一覧」 -「個人タスク」

8. 「IM-FormaDesigner」で作成した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。

図：承認画面（IM-FormaDesigner）

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

11. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。

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

図：「タスク一覧」 -「個人タスク」

13. 「IM-FormaDesigner」で作成した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。

図：承認画面（IM-FormaDesigner）

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

16. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示します。

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

図：「タスク一覧」 - 「個人タスク」

17. 「IM-FormaDesigner」で作成した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。

図：確認画面（IM-FormaDesigner）

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

イベント

シグナルイベントを使用する

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

シグナルを送受信するイベントは、以下のとおりです。

シグナルを送信するイベント

- シグナルスローイベント

シグナルを受信するイベント

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

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

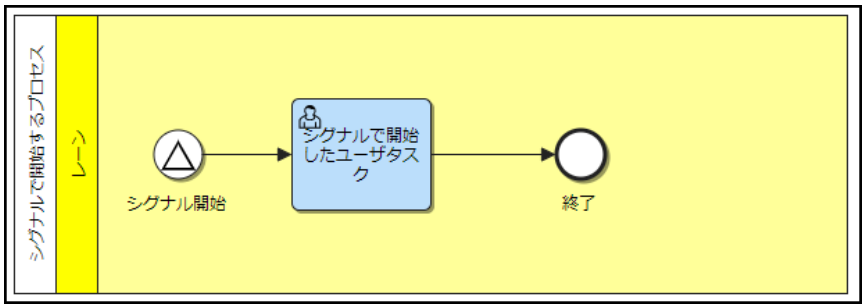
> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **event_signal_usage.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

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

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

シグナルを受信することにより開始するプロセス定義を作成する

シグナルを受信することにより開始するプロセス定義を作成します。

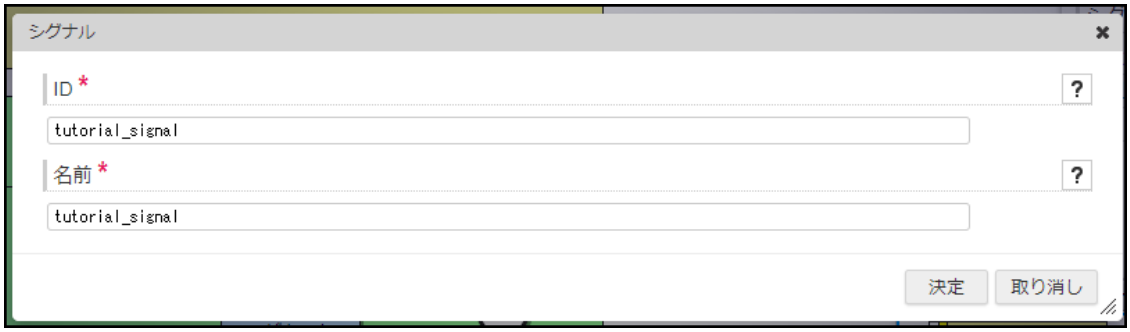


図：完成イメージ

1. 「シグナル」を登録します。
    1. 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」

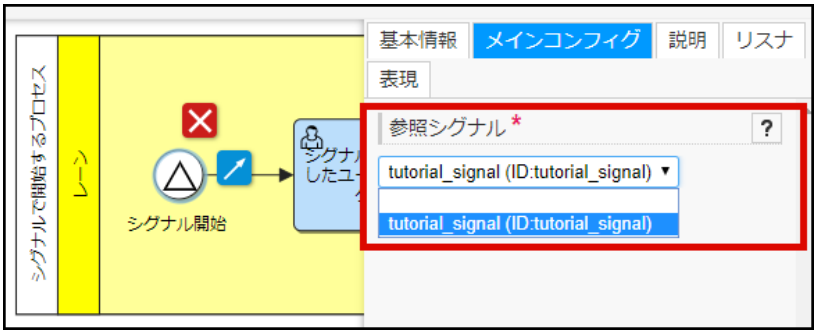
2. 「シグナル」の「ID」と「名前」を設定します。
   - ID：tutorial_signal
   - 名前：tutorial_signal



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. キャンバスの空白部分、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

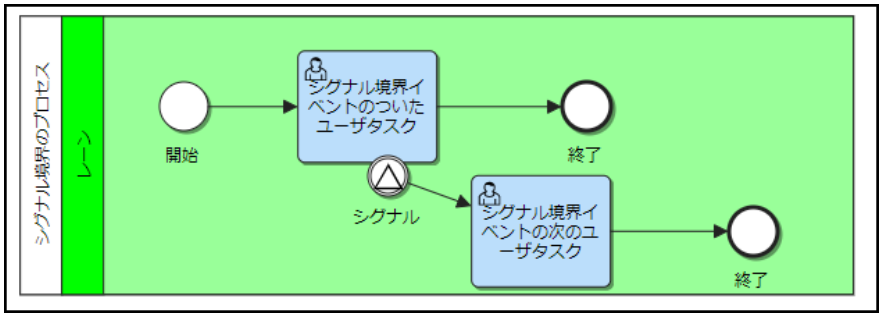
図：「プロパティ」 - 「プロセス」

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。

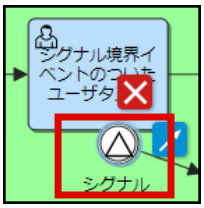
3. キャンバスの空白部分、または、プールをクリックし、「プロパティ」をプロセス全体切り替えます。
   「プロパティ」 の 「プロセス」タブから、項目を以下のように設定してください。
   - プロセス定義キー：event_signal_usage_boundary_process
   - 処理対象ユーザ：aoyagi

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

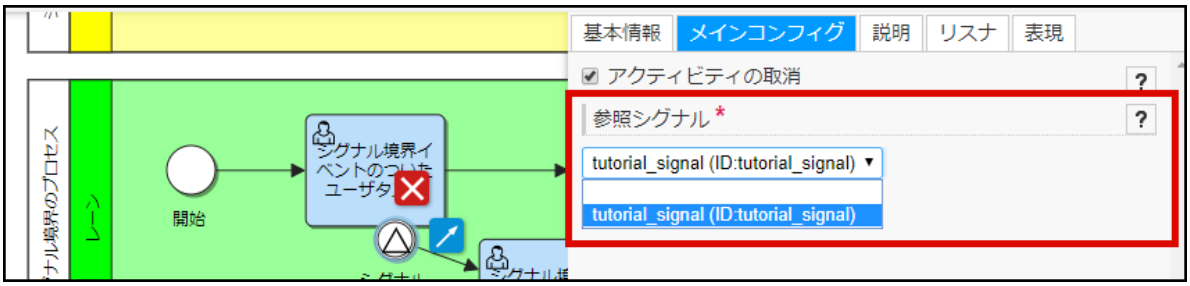
図：「ユーザタスク」

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。



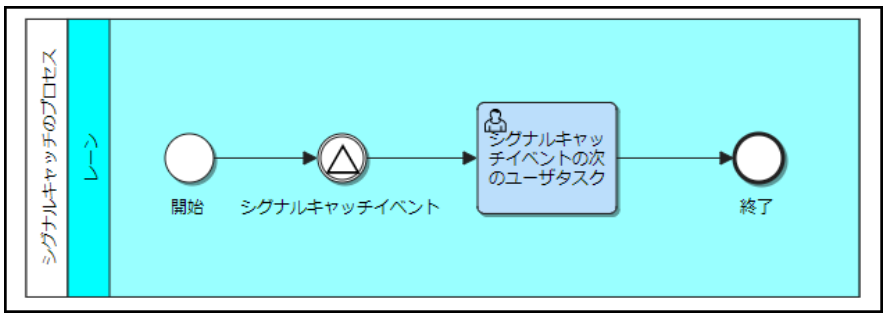
図：「シグナル境界イベント」 -「プロパティ」 -「メインコンフィグ」

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。

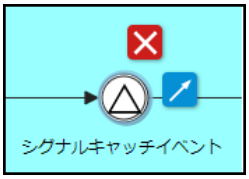
   登録方法は「*シグナルを受信することにより開始するプロセス定義を作成する*」を参照してください。

2. 「開始イベント」を配置します。

3. キャンバスの空白部分、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

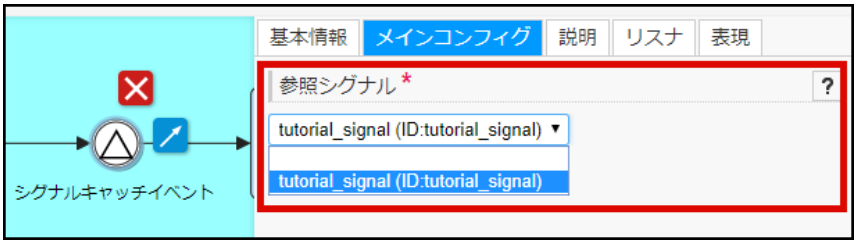
図：「プロパティ」 - 「プロセス」

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。



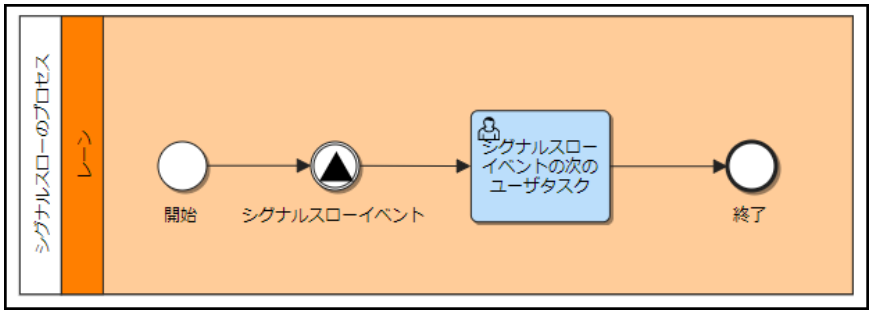
図：「シグナルキャッチイベント」 - 「プロパティ」 - 「メインコンフィグ」

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

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。

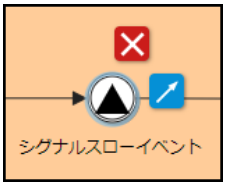
   登録方法は「*シグナルを受信することにより開始するプロセス定義を作成する*」を参照してください。

2. 「開始イベント」を配置します。

3. キャンバスの空白部分、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

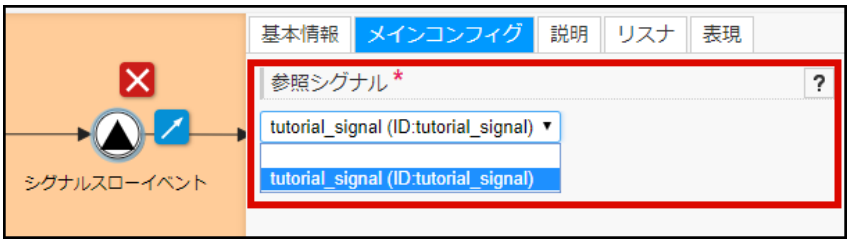
図：「プロパティ」 - 「プロセス」

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。



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」

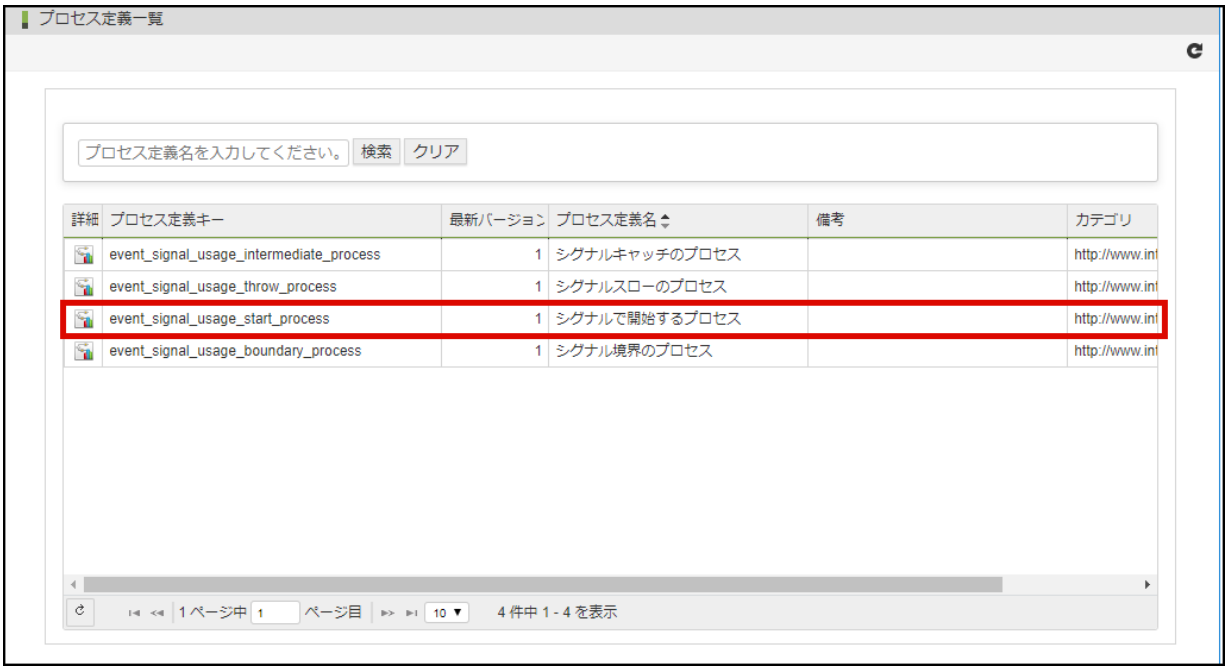
6. 「シグナルスローイベント」から次に進んだことを確認するために「ユーザタスク」を配置します。
   - 担当者 : aoyagi

図：「ユーザタスク」

実行結果を確認する

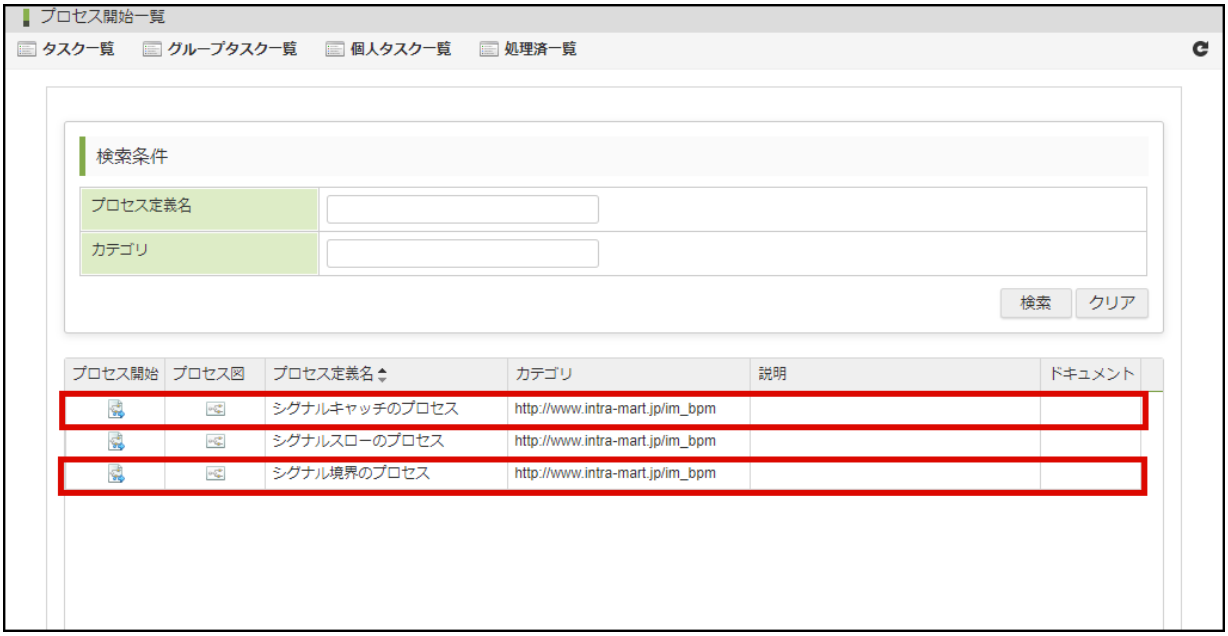
このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

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。



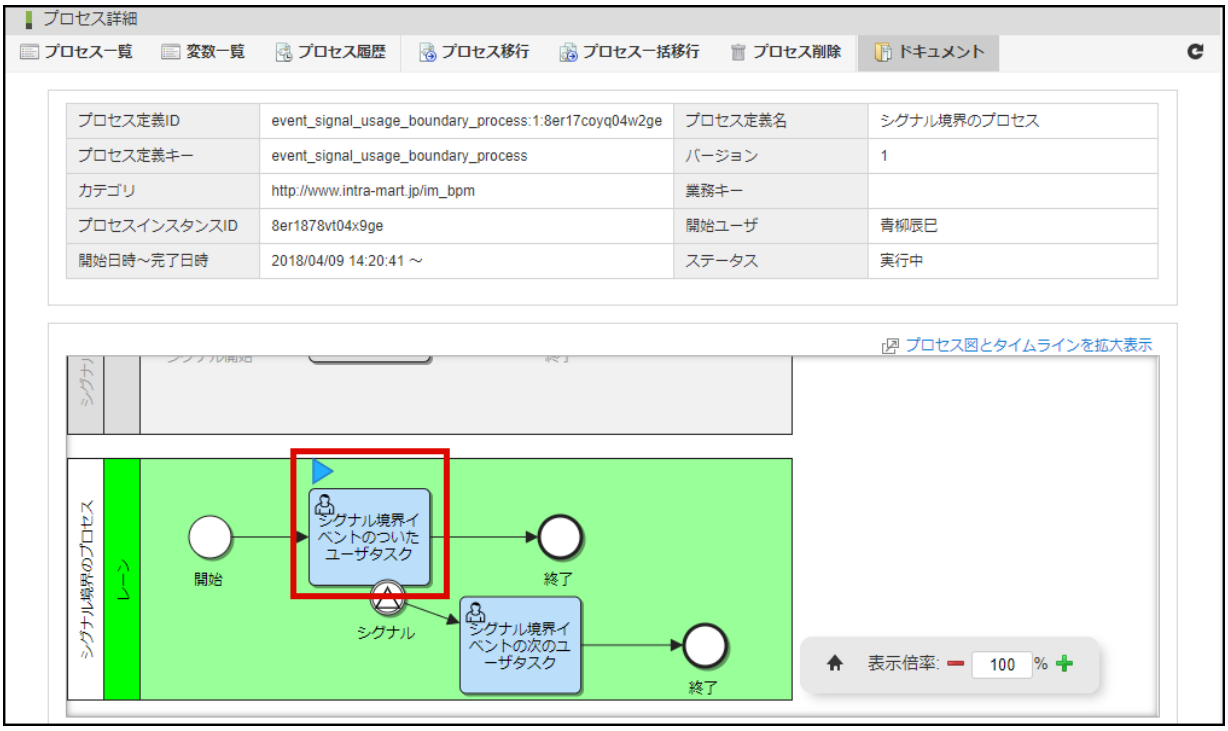
図：プロセス定義一覧 event_signal_usage_start_process

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。



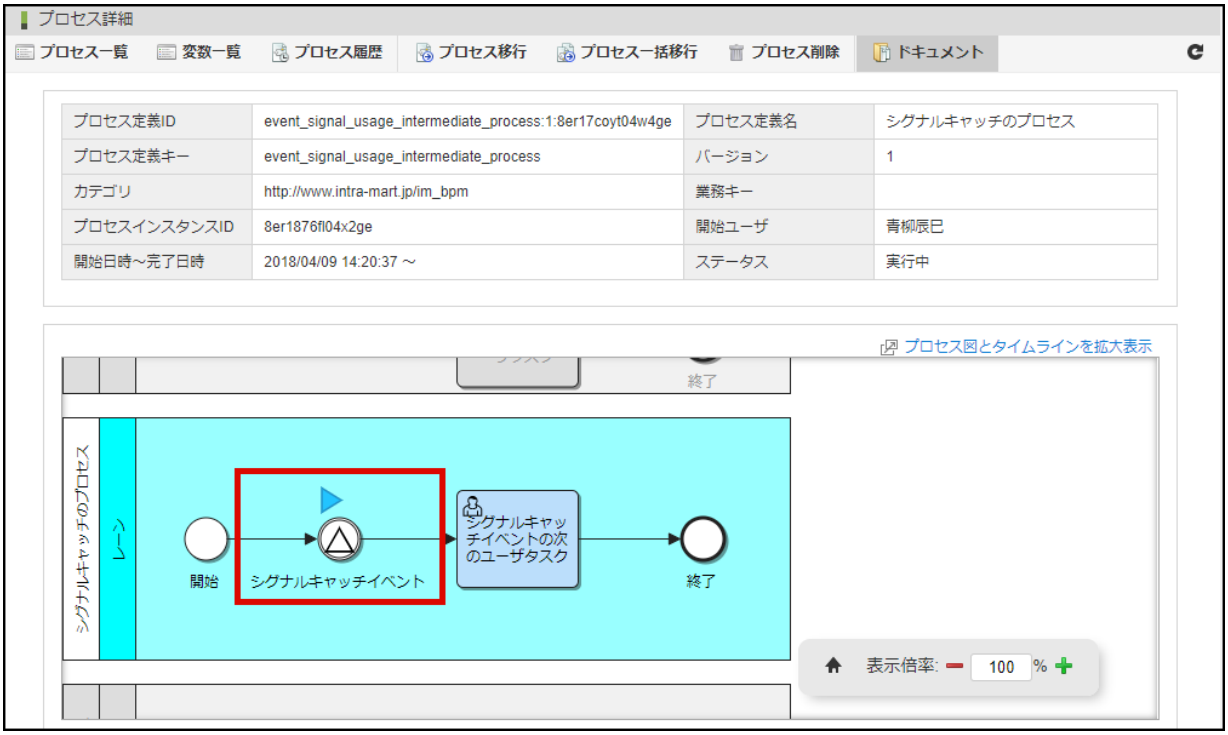
図：プロセス開始一覧

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。



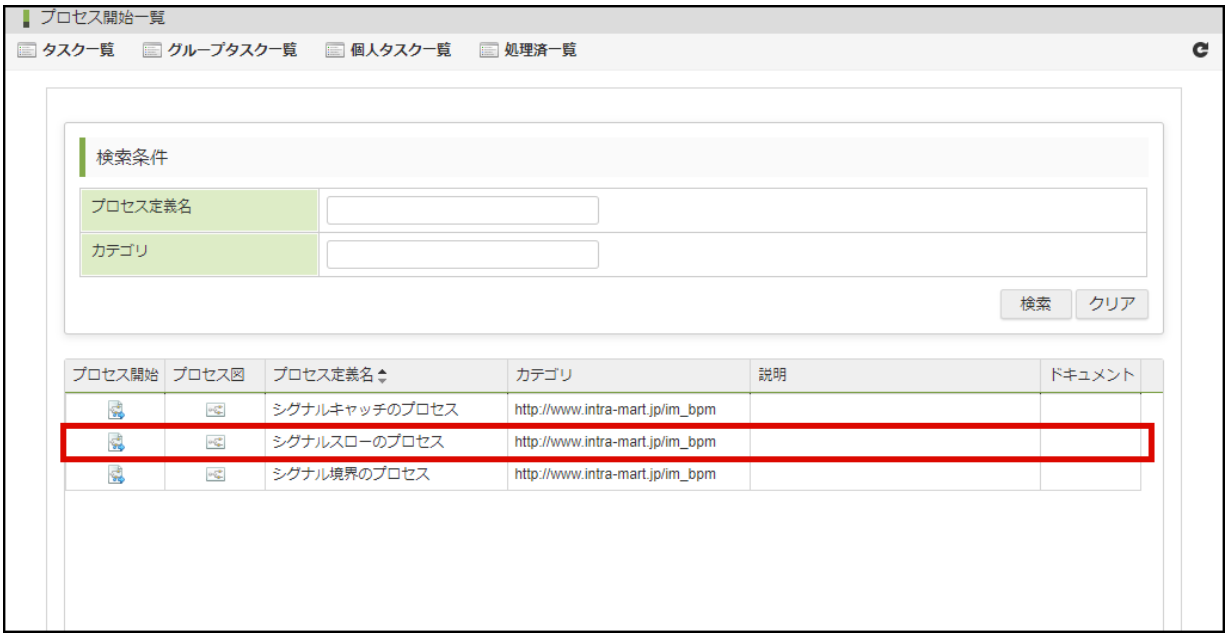
図：プロセス詳細 event_signal_usage_boundary_process

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」のロジックフローを使用します。
チュートリアルを開始する前に以下の資材をインポートしてください。
im_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**
>
> ロジックフローのインポートについての詳細は「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」を参照してください。

コラム

各イベントの詳細については、以下を参照してください。

- メッセージ開始イベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「メッセージ開始イベント」
- メッセージ境界イベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「メッセージ境界イベント」
- メッセージキャッチイベント：「IM-BPM プロセスデザイナ 操作ガイド」 - 「メッセージキャッチイベント」

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

コラム

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

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

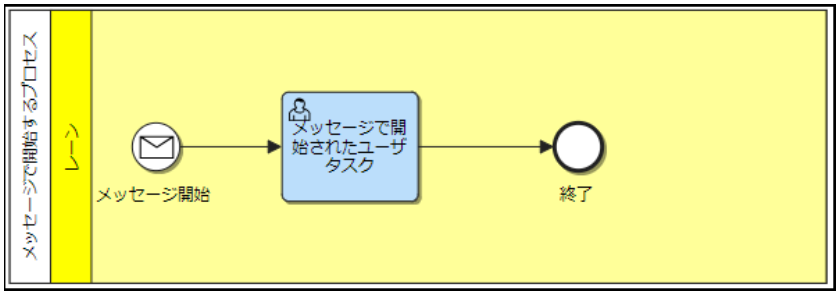
コラム

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

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

メッセージを受信することにより開始するプロセス定義を作成する

メッセージを受信することにより開始するプロセス定義を作成します。



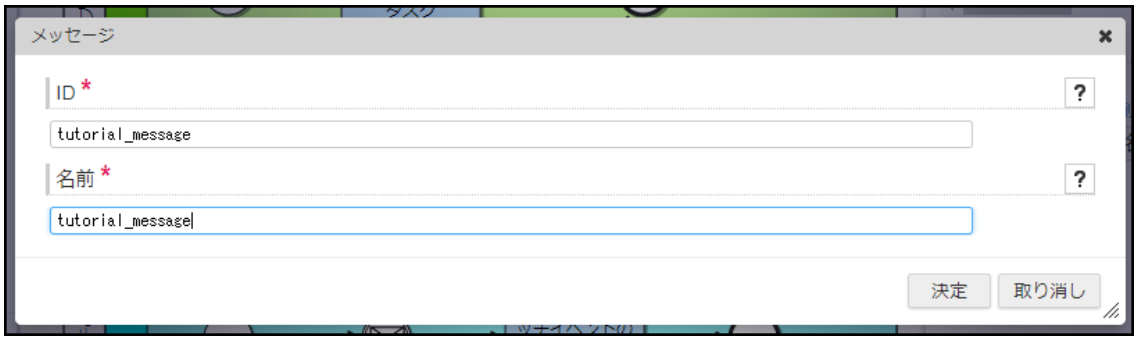
図：完成イメージ

1. 「メッセージ」を登録します。

    1. 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」

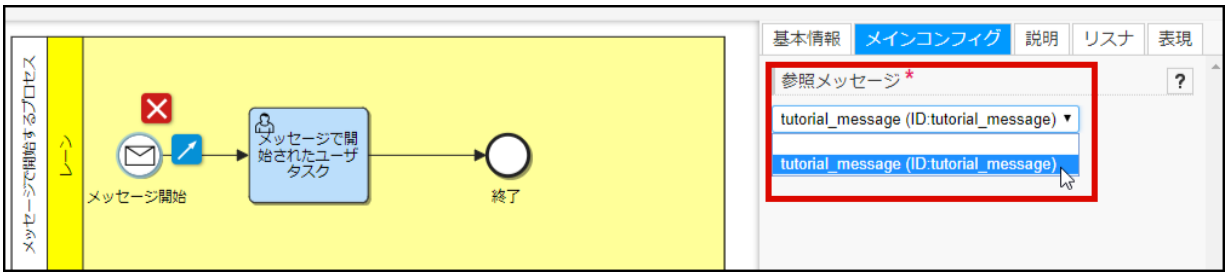
    2. 「メッセージ」ダイアログで、項目を以下のとおりに設定します。
        - ID : tutorial_message
        - 名前 : tutorial_message



図：メッセージ

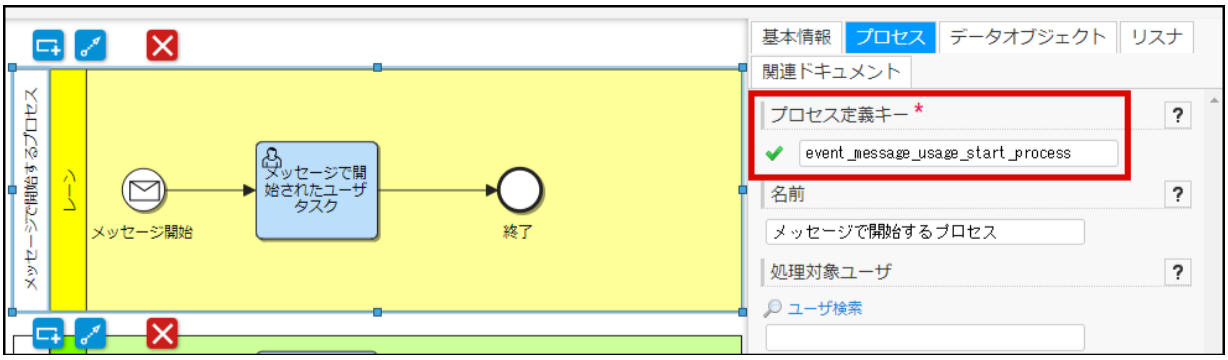
2. 「メッセージ開始イベント」を配置します。

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。



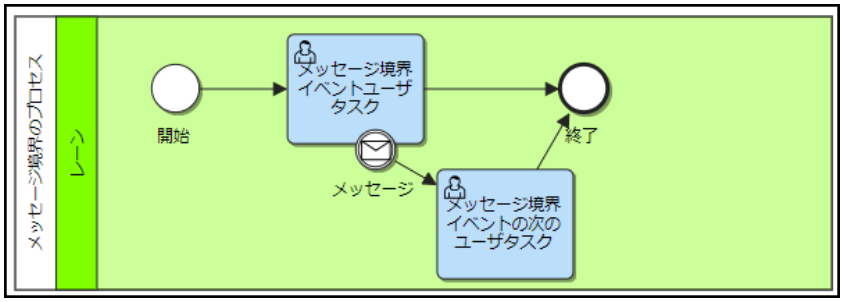
図：プロセス全体 - 「プロパティ」 - 「プロセス」

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。



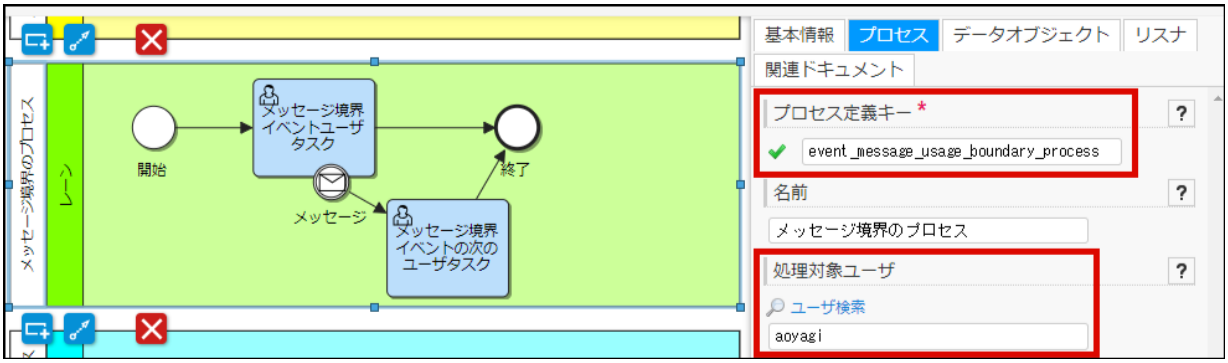
図：完成イメージ

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. 「ユーザタスク」を配置します。
   - 担当者：aoyagi

図：「ユーザタスク」

5. 「メッセージ境界イベント」を「ユーザタスク」に配置します。

図：「メッセージ境界イベント」

6. 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。

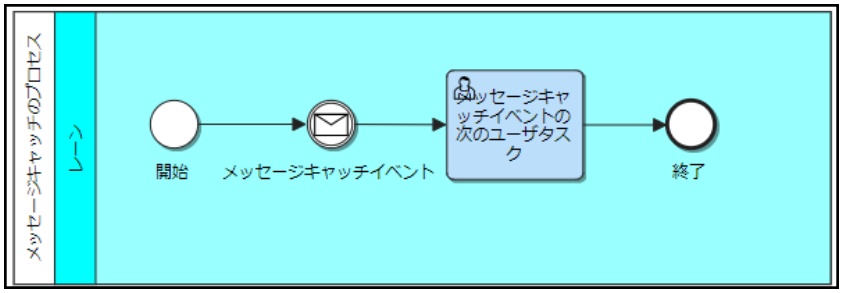
図：「メッセージ境界イベント」 - 「プロパティ」 - 「メインコンフィグ」

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。

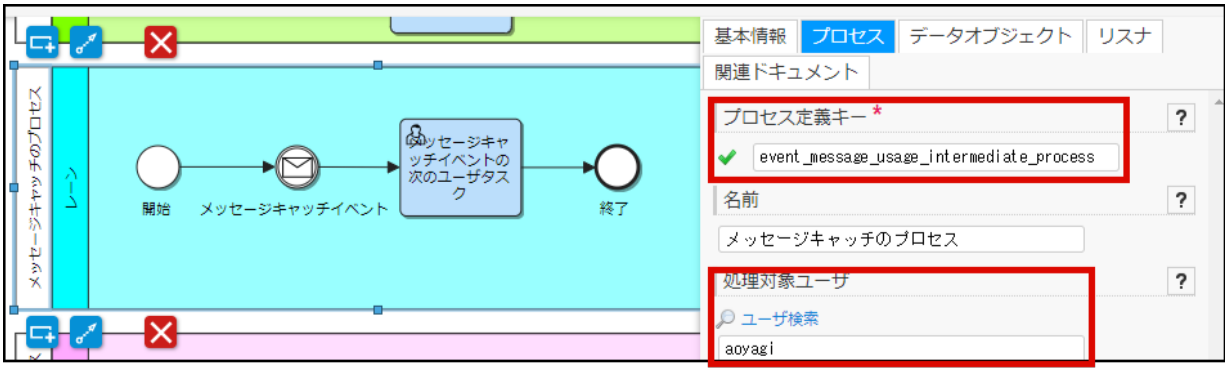


図：完成イメージ

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。

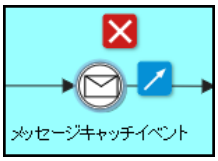
2. 「開始イベント」を配置します。

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_intermediate_process
   - 処理対象ユーザ：aoyagi



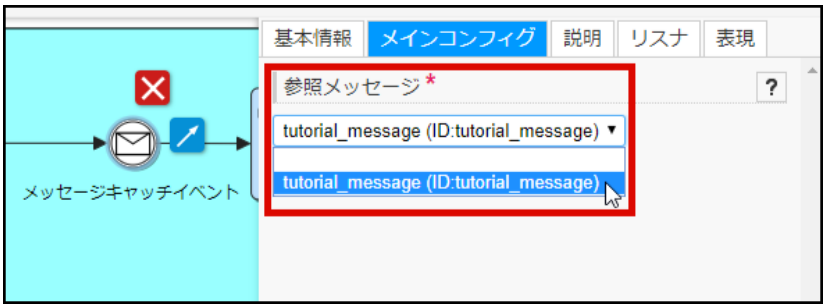
図：「プール」 - 「プロパティ」 - 「プロセス」

4. 「メッセージキャッチイベント」を配置します。



図：「メッセージキャッチイベント」

5. 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

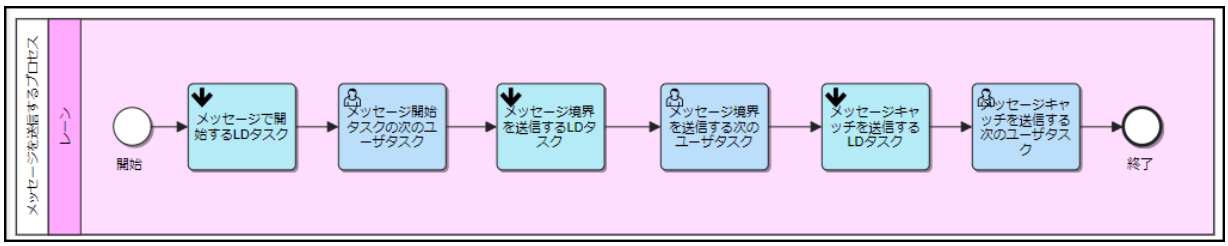
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

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。

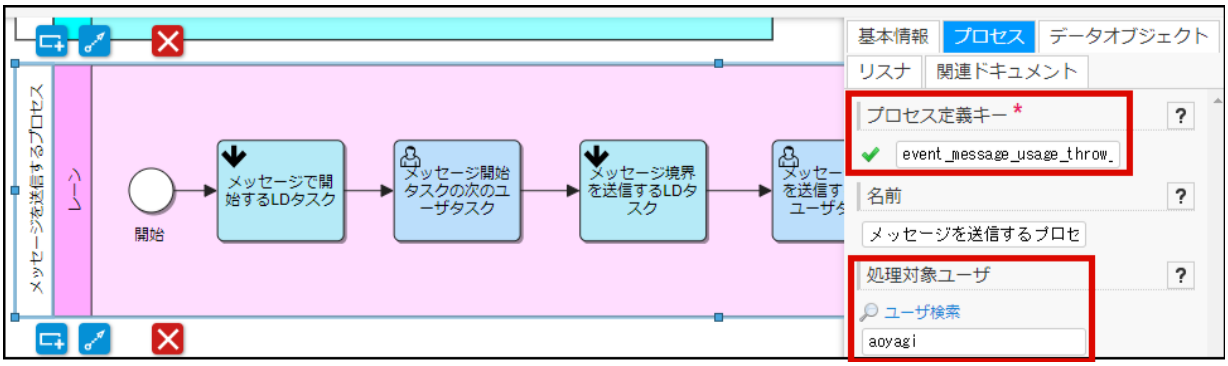
また、プロセスの進行状況を確認するため、「IM-LogicDesignerタスク」の他に「ユーザタスク」を配置しています。



図：完成イメージ

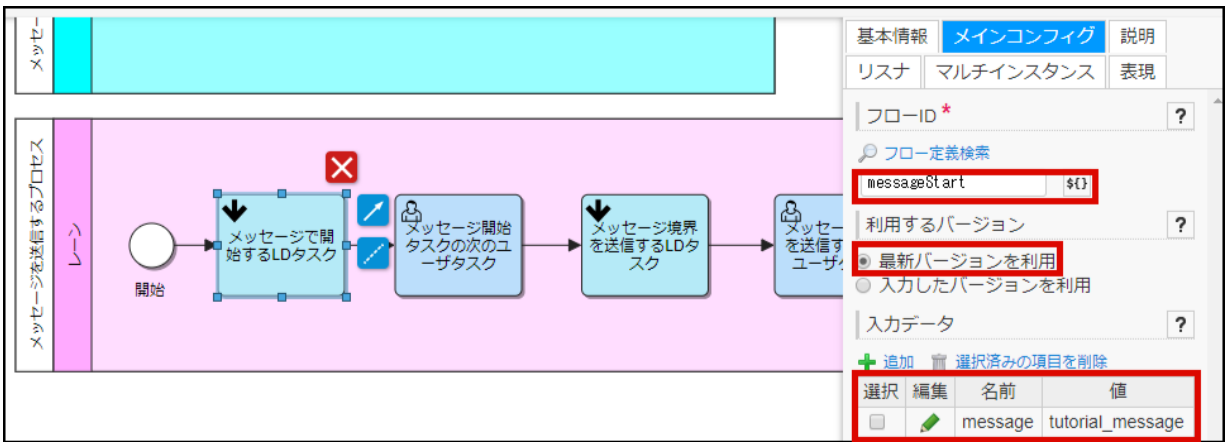
1. 「開始イベント」を配置します。

2. 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」

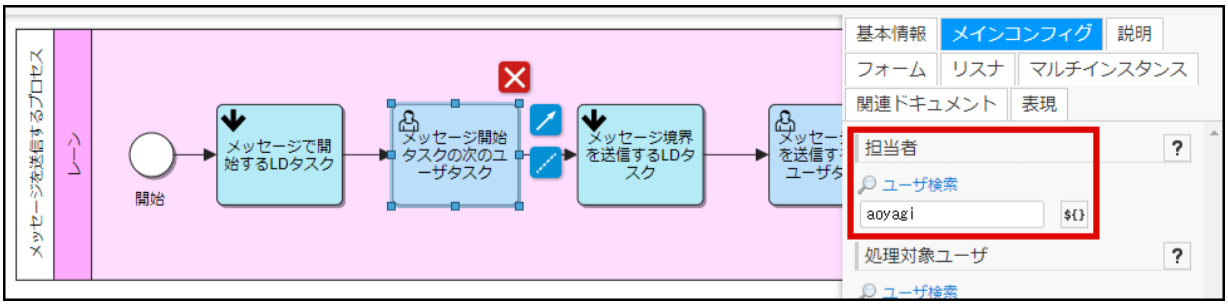
3. 「IM-LogicDesignerタスク」の「メインコンフィグ」で、以下のように項目を設定します。
   - フローID : messageStart
   - 利用するバージョン : 最新バージョンを利用
   - 入力データ
     - 名前 : message
     - 値 : tutorial_message



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

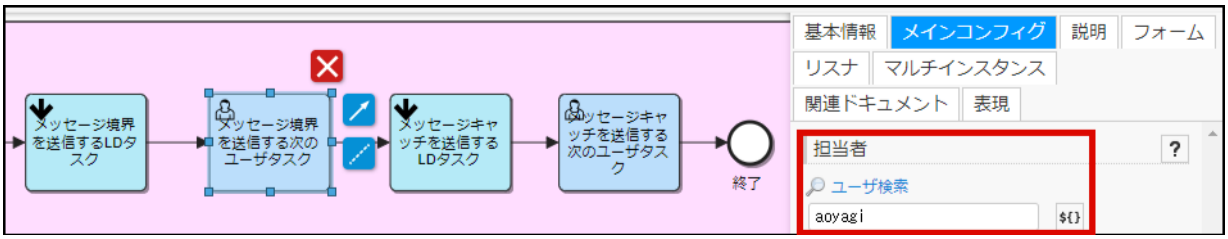


図：「ユーザタスク」

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

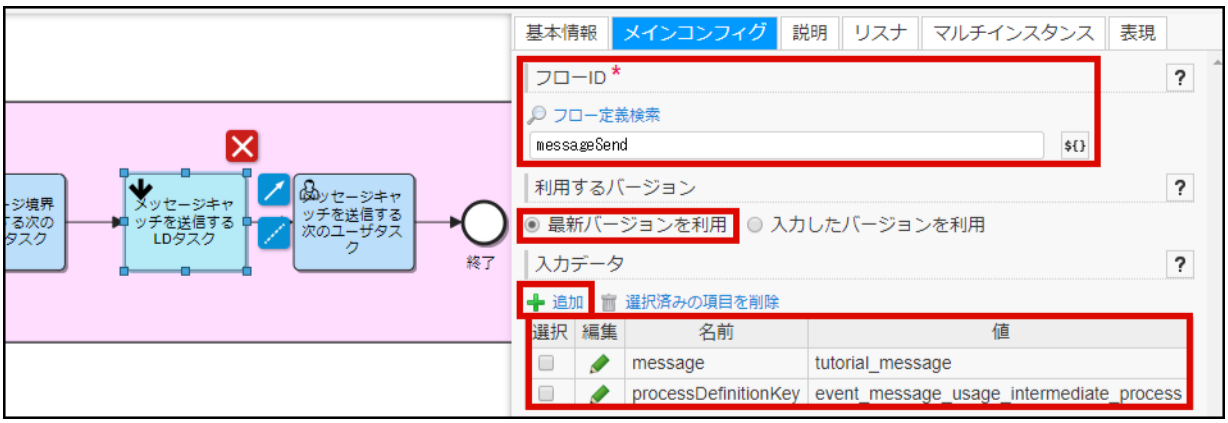
図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

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



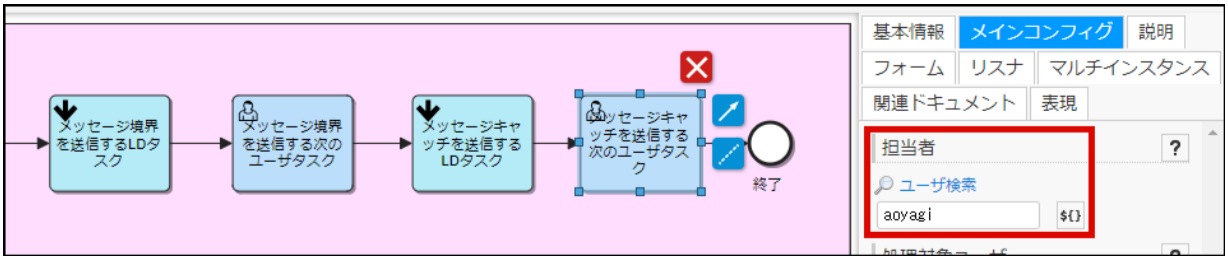
図：「ユーザタスク」

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。

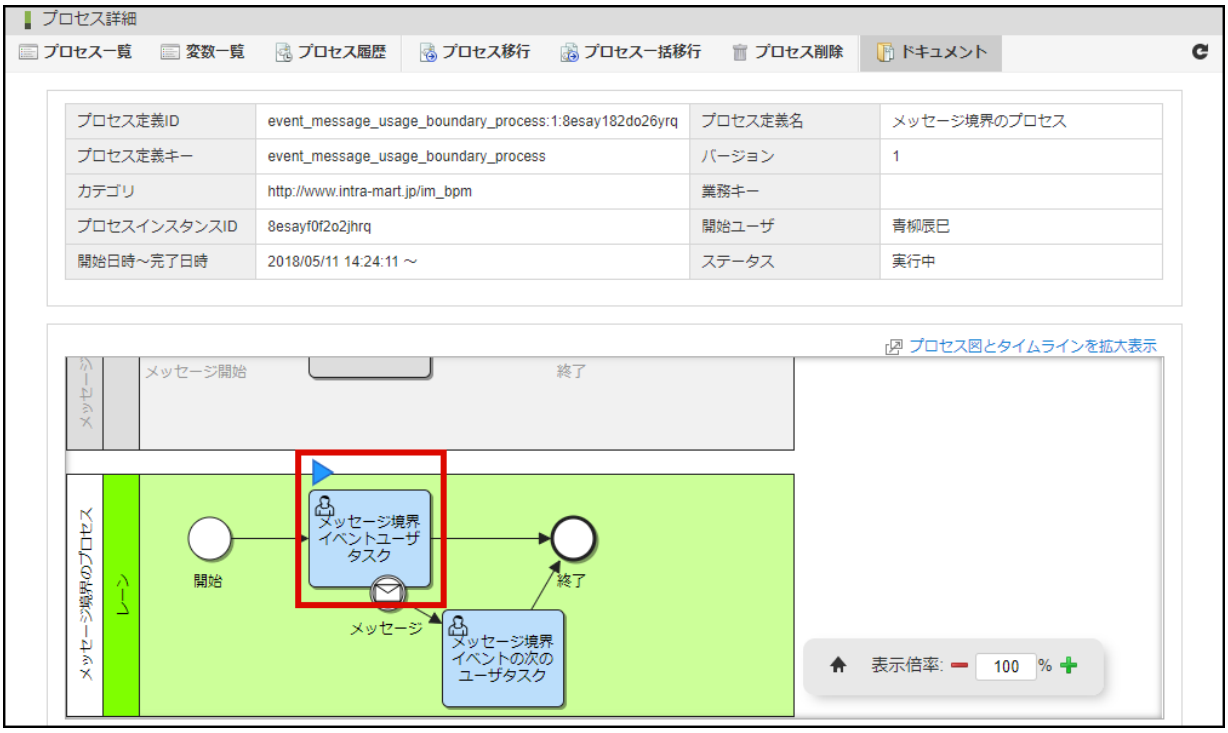
1. 「プロセス定義キー」 event_message_usage_start_process がプロセス定義として、デプロイされていることを確認します。

図：プロセス定義一覧 event_message_usage_start_process

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

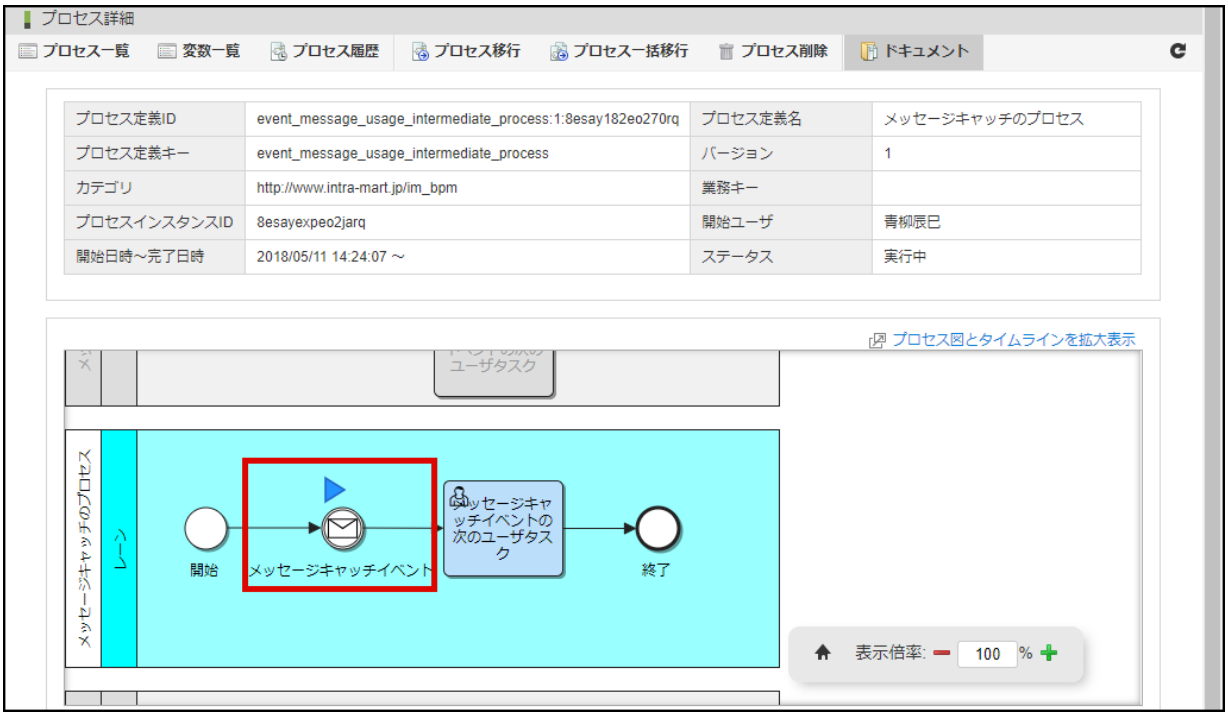
図：プロセス開始

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



図：プロセス詳細 event_message_usage_boundary_process

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



図：プロセス詳細 event_message_usage_intermediate_process

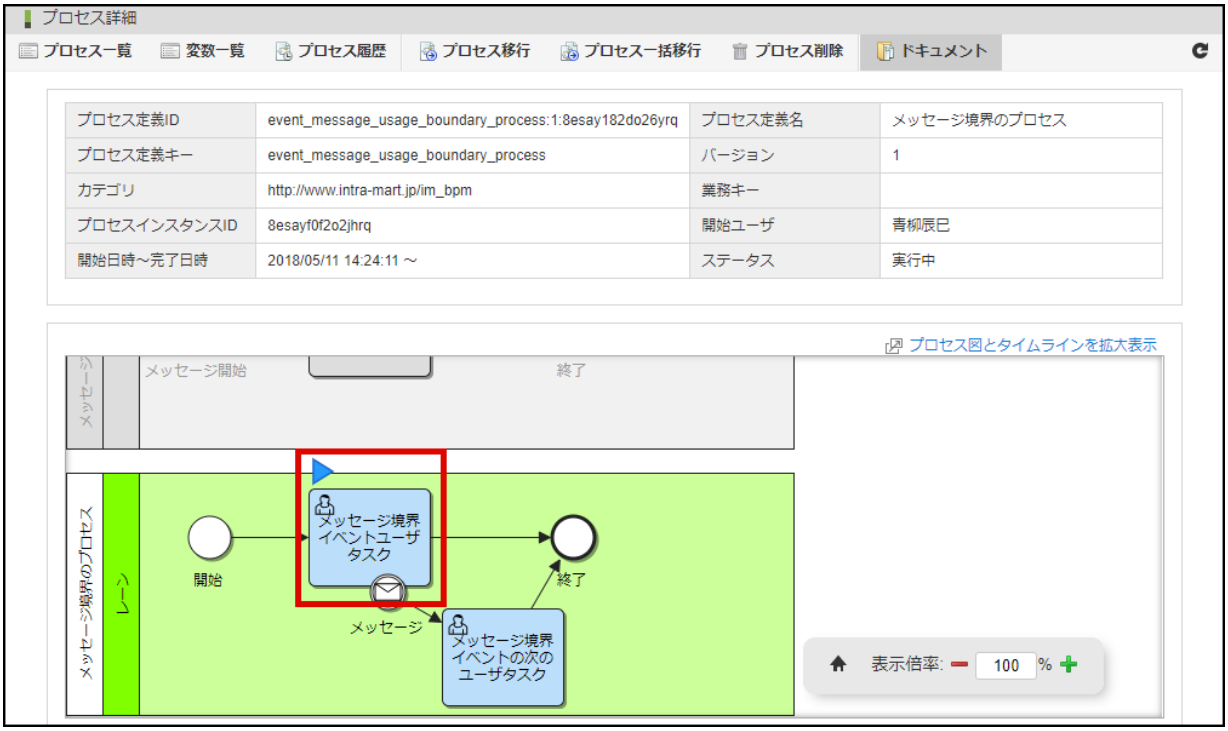
5. 「プロセス定義キー」 event_message_usage_throw_process のプロセスを開始します。

図：プロセス開始 event_message_usage_throw_process

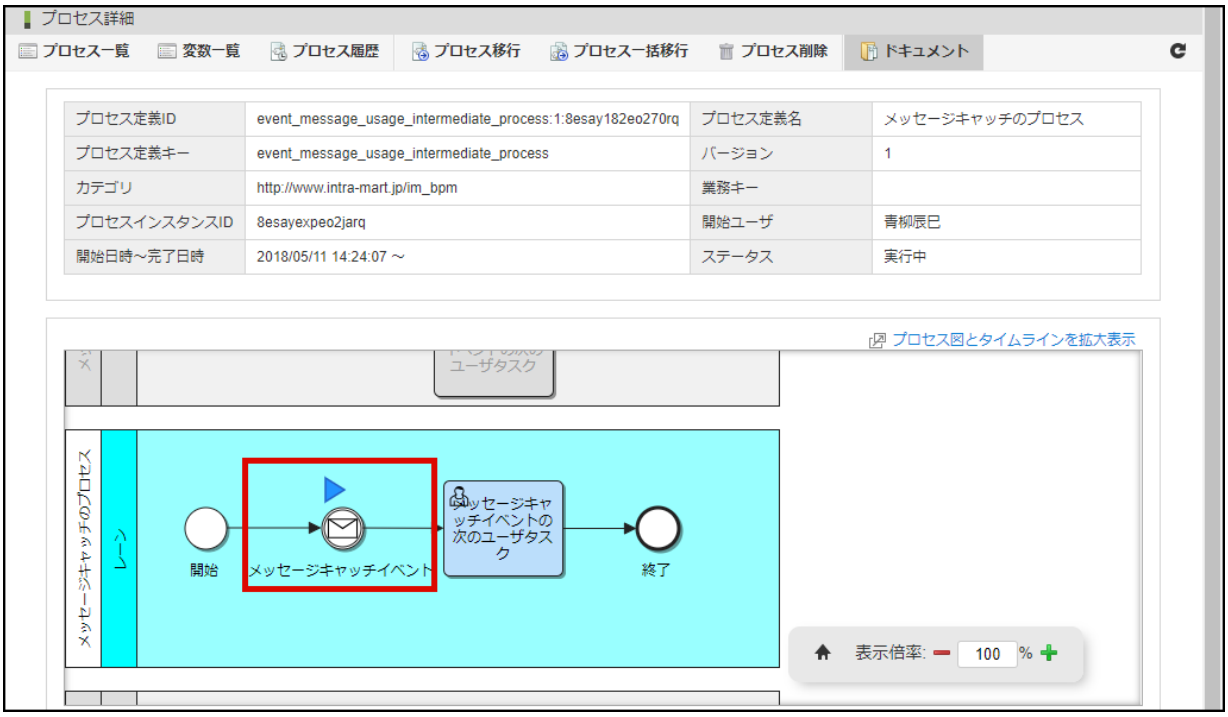
6. 「プロセス定義キー」 event_message_usage_start_process が開始されていることを確認します。

図：タスク一覧

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



図：プロセス詳細 event_message_usage_boundary_process



図：プロセス詳細 event_message_usage_intermediate_process

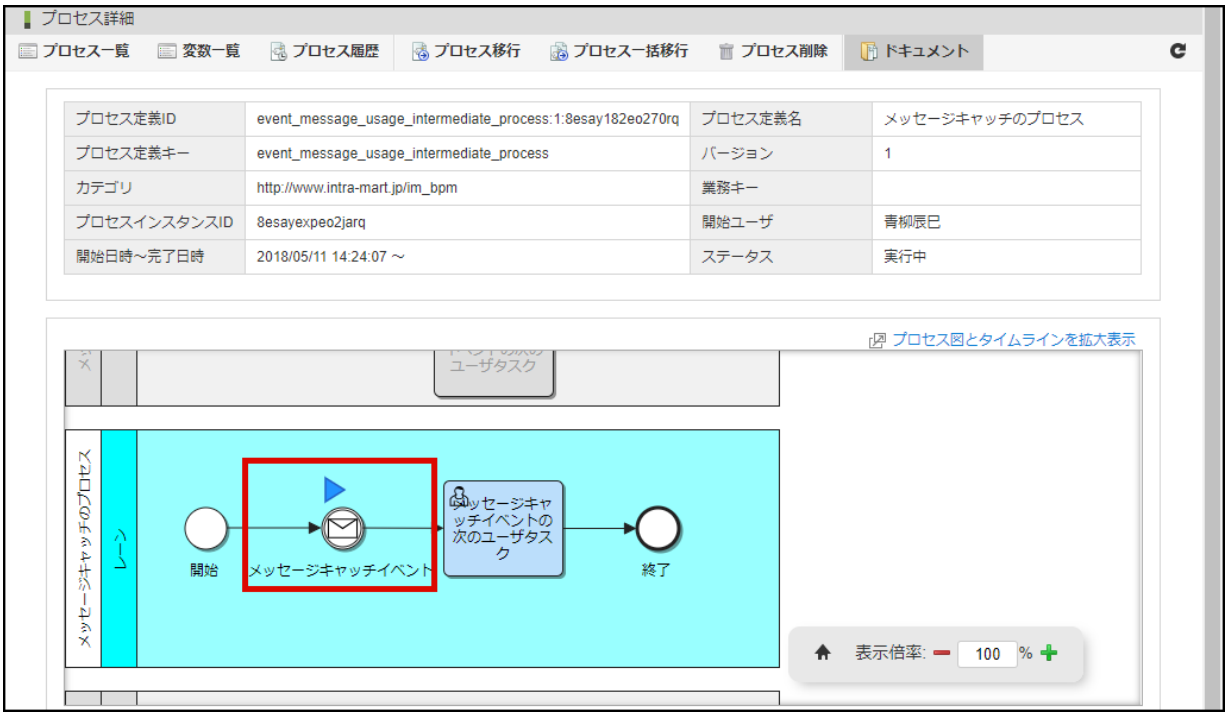
8. 「メッセージを送信するプロセス」の「ユーザタスク」を処理します。

図：タスク一覧

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

図：タスク一覧

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

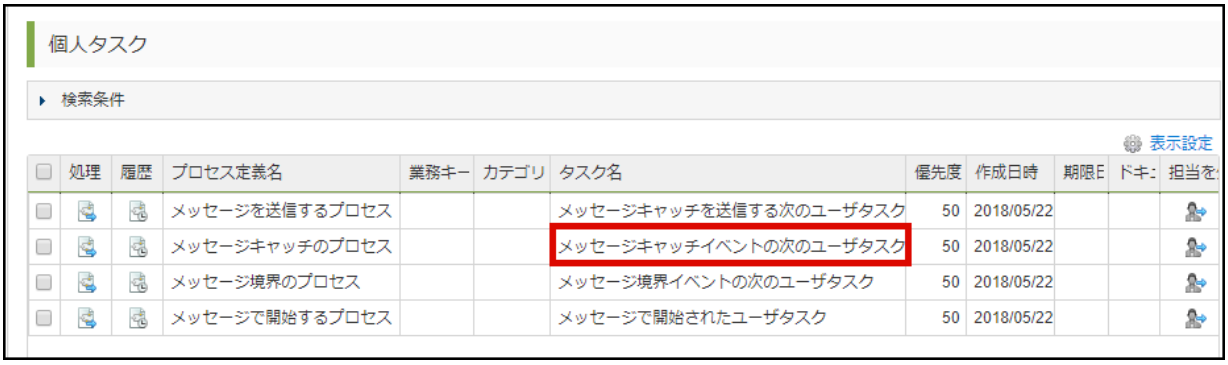


図：プロセス詳細 event_message_usage_intermediate_process

11. 「メッセージを送信するプロセス」の「ユーザタスク」を処理します。

図：タスク一覧

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



図：タスク一覧

タイマイベントを使用する

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

- タイマ開始イベント
- タイマ境界イベント
- タイマキャッチイベント

コラム

各イベントの詳細については、以下を参照してください。

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

コラム

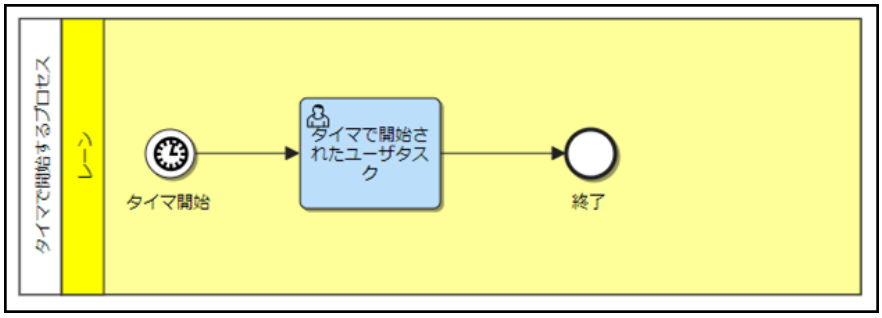
このチュートリアルで作成する「プロセス定義」のサンプルを、以下のリンクからダウンロードできます。
event_timer_usage.bpmn

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

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

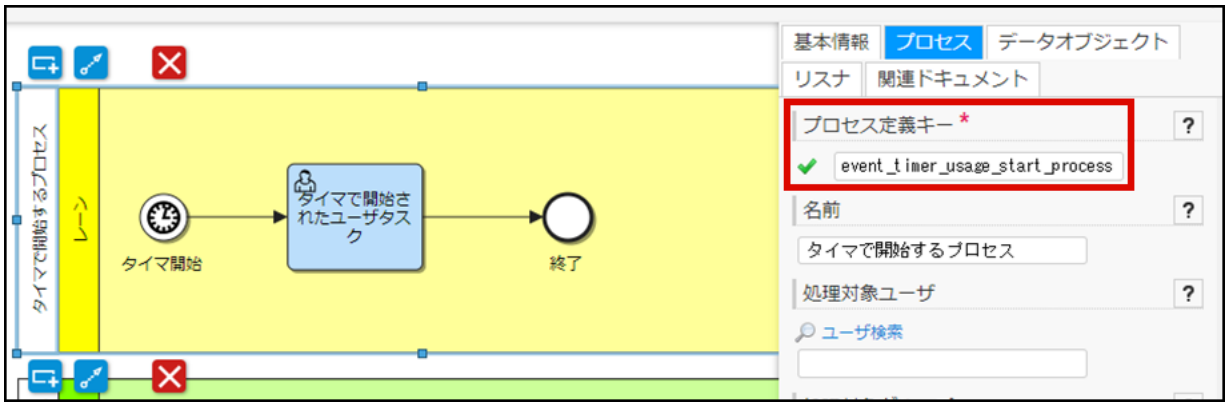
指定した時間間隔で複数回プロセスを開始するプロセス定義を作成する

指定した時間間隔で複数回プロセスを開始するプロセス定義を作成します。



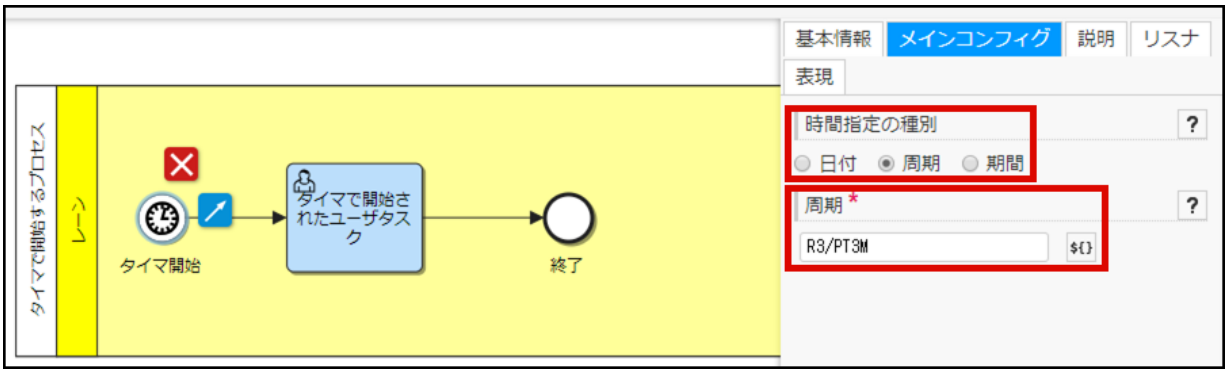
図：完成イメージ

1. キャンバスの空白部分、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。
   - 時間指定の種別：周期
   - 周期：R3/PT3M



図：「タイマ開始イベント 」 -「プロパティ」 -「メインコンフィグ」

コラム

R3/PT3Mは「3分間隔の繰り返しを3回行う」ことを、ISO 8601の形式に則って記述したものです。

例:10時間毎に5回開始→R5/PT10H
　1日毎に10回開始→R10/P1D

コラム

cron式で周期を指定する

「タイマイベント」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）」**

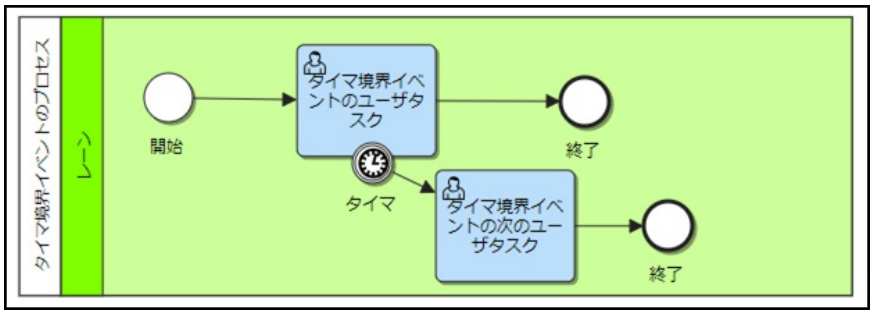
「#」は[曜日]フィールドに使用することで、週と曜日を設定できます。
左側の数字が曜日を表し、右側の数字が第何週目かを表します。
例：「4#2」→月の第2水曜日

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。

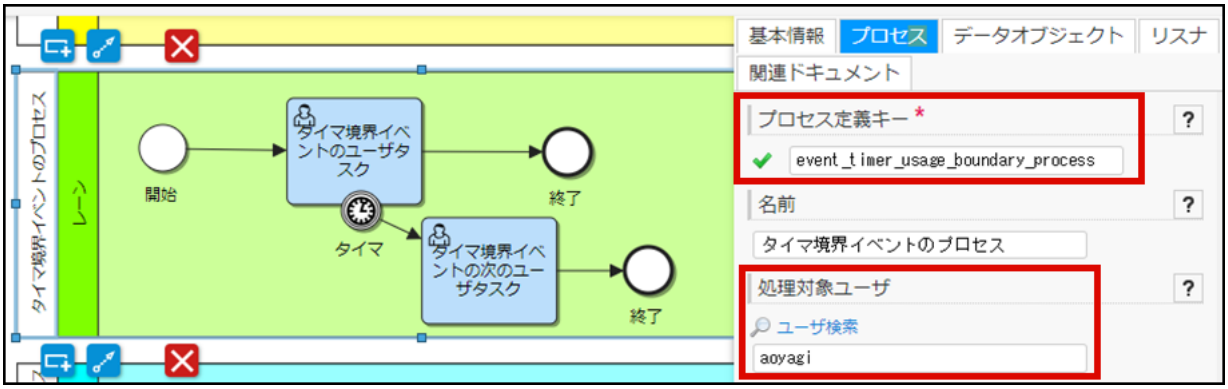
指定した時間経過によりユーザタスクを中断するプロセス定義を作成する

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。



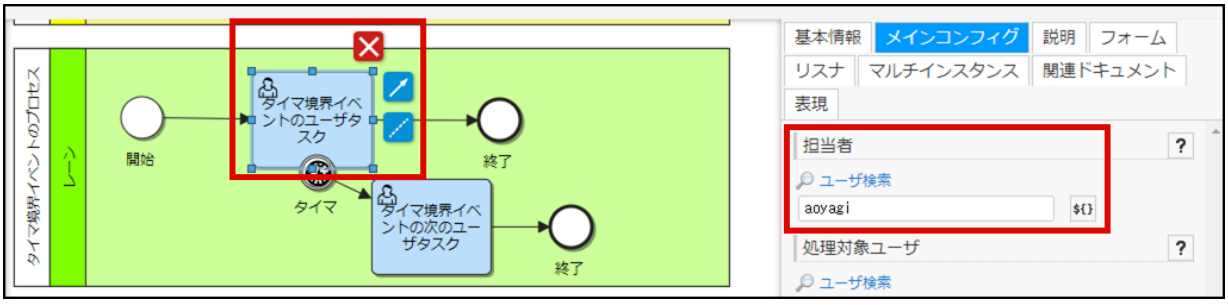
図：完成イメージ

1. 「開始イベント」を配置します。
2. キャンバスの空白部分、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



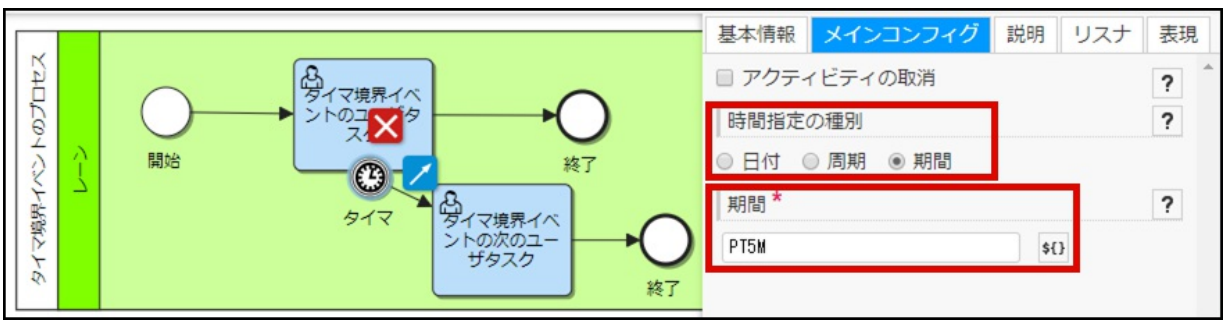
図：「タイマ境界イベントのプロセス」 - 「プロパティ」 - 「プロセス」

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」

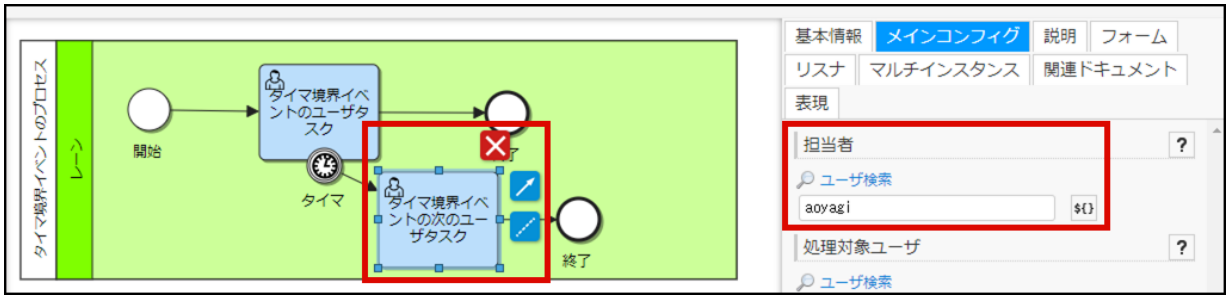
4. 5分経過することで、ユーザタスクを中断する「タイマ境界イベント」を配置します。
   タイマ境界イベントの「プロパティ」で以下の項目を設定します。
   - 時間指定の種別：期間
   - 期間：PT5M



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

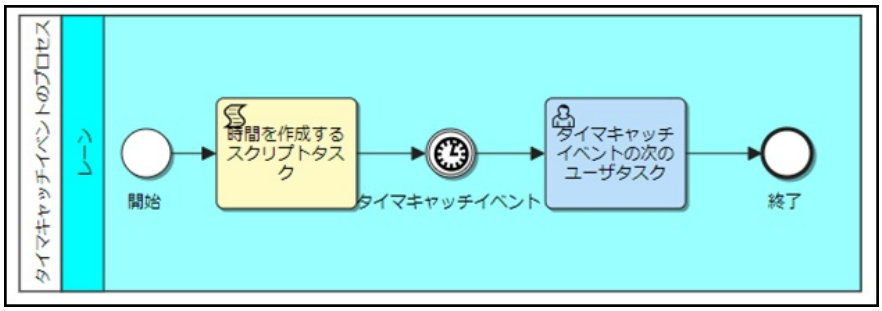
5. 指定した時間経過により中断されたことを確認するため、別のユーザタスクを配置します。
   - 処理対象ユーザ：aoyagi



図：「ユーザタスク」

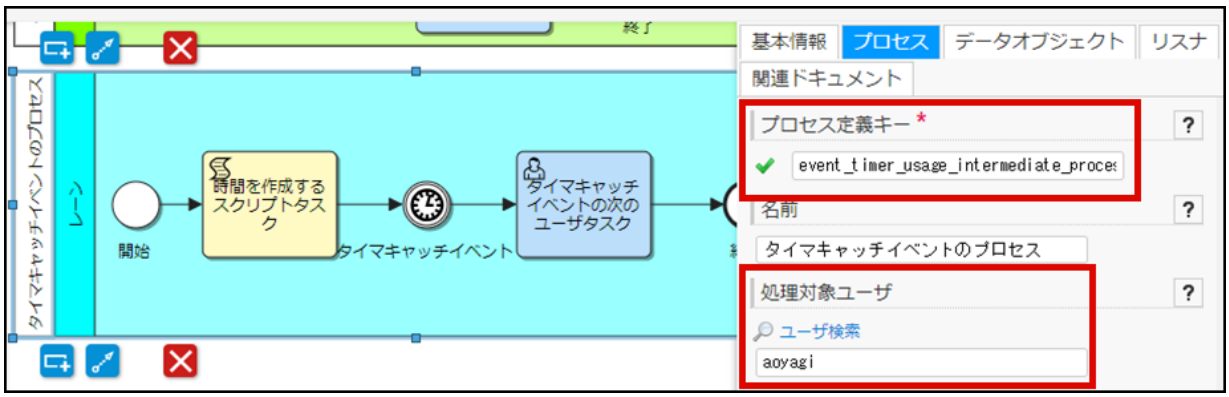
指定した時間にユーザタスクを開始するプロセス定義を作成する

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。



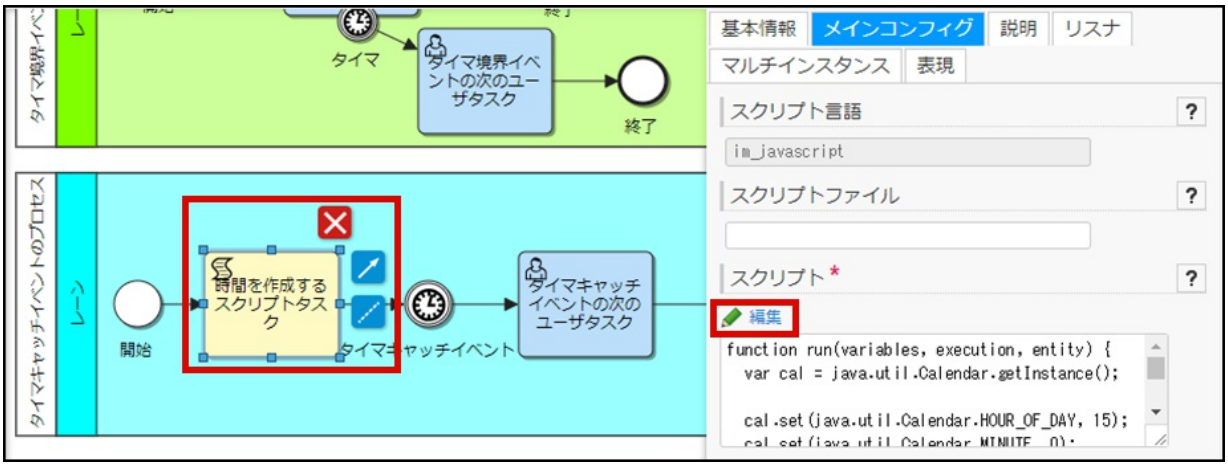
図：完成イメージ

1. 「開始イベント」を設置します。
2. キャンバスの空白部分、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」

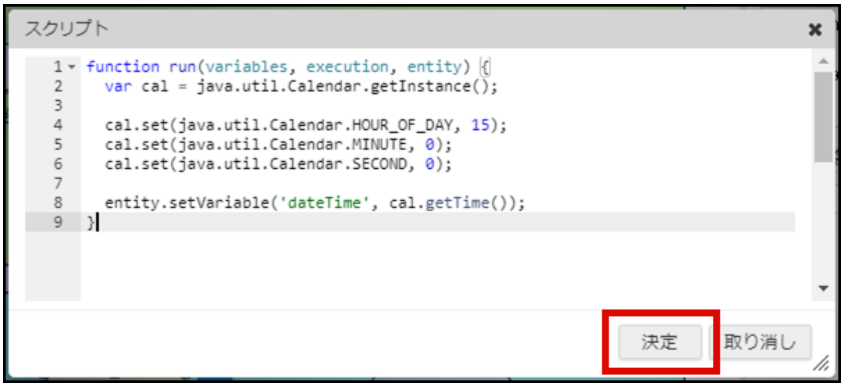
3. 「スクリプトタスク」を作成します。
   「スクリプトタスク」の「メインコンフィグ」タブから、「スクリプト」の「編集」リンクをクリックします。



図：「スクリプトタスク」 - 「プロパティ」 - 「メインコンフィグ」

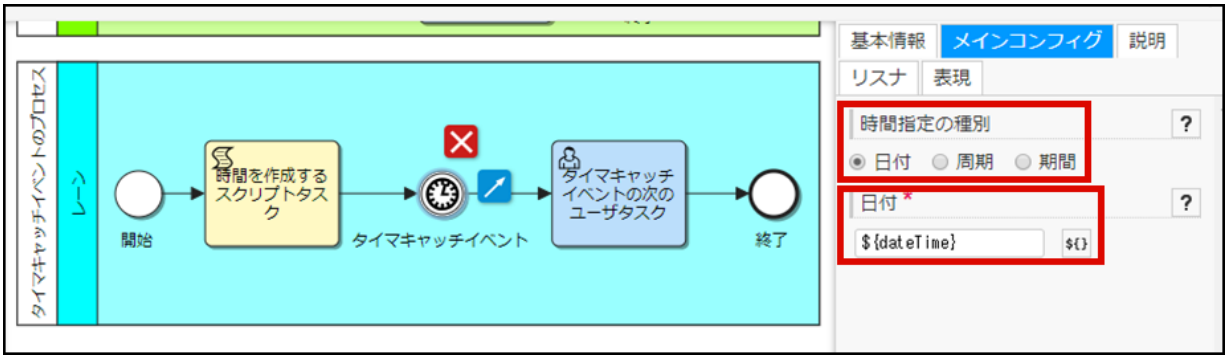
4. 「当日の15時」という情報を変数「 dateTime 」に設定するため、以下のスクリプトを設定します。

```
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());
}
```



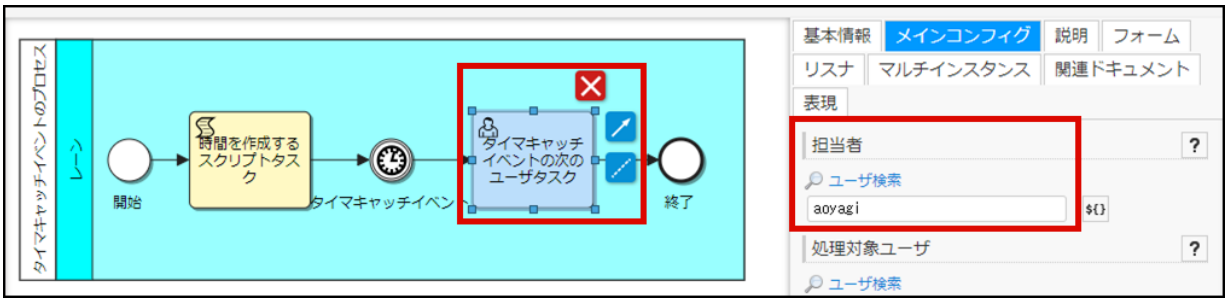
図：スクリプト編集のポップアップ

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



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

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



図：「ユーザタスク」

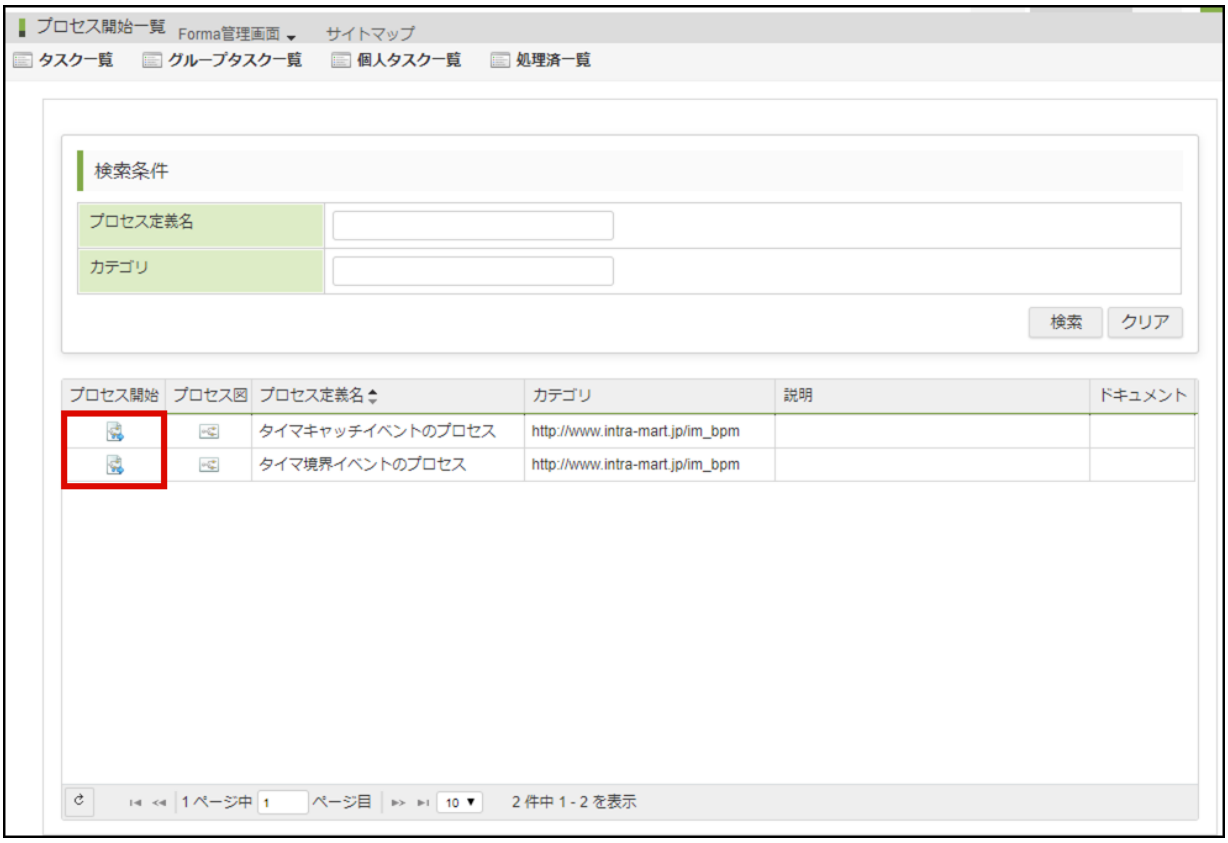
実行結果を確認する

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

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

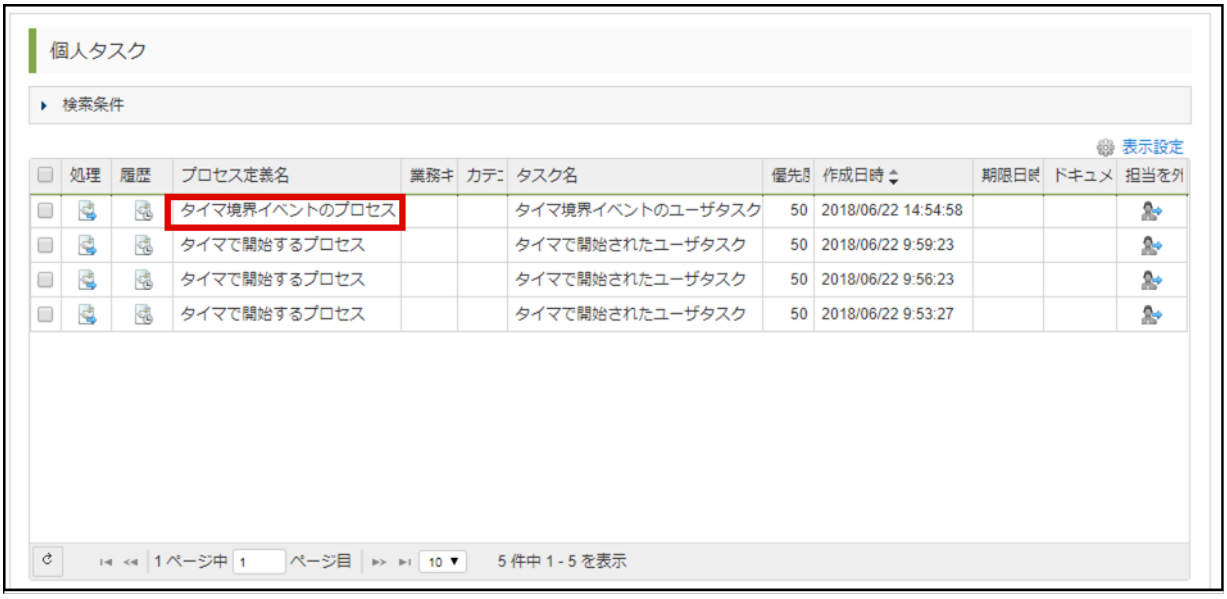
図：「タスク一覧」

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



図：「プロセス開始一覧」

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

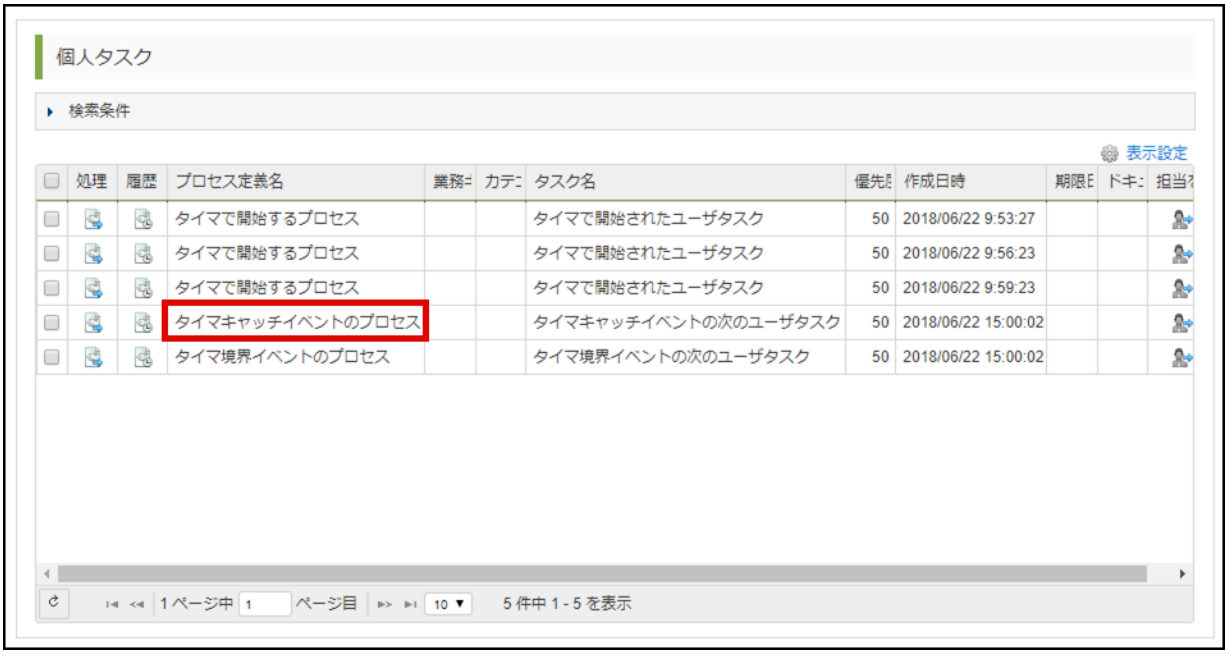


図：「タスク一覧」

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

図：「プロセス一覧」 - 「プロセス詳細」

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



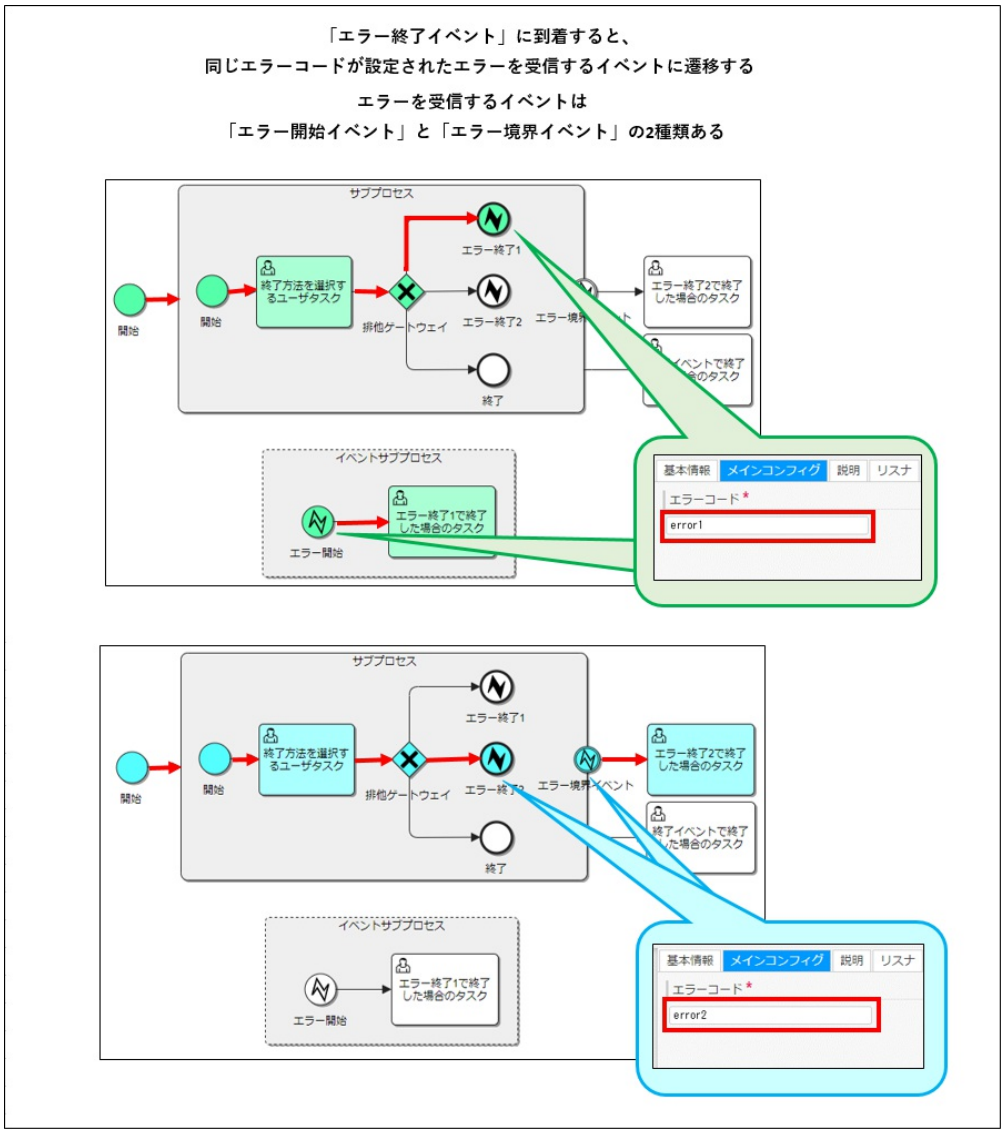
図：「タスク一覧」

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

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」

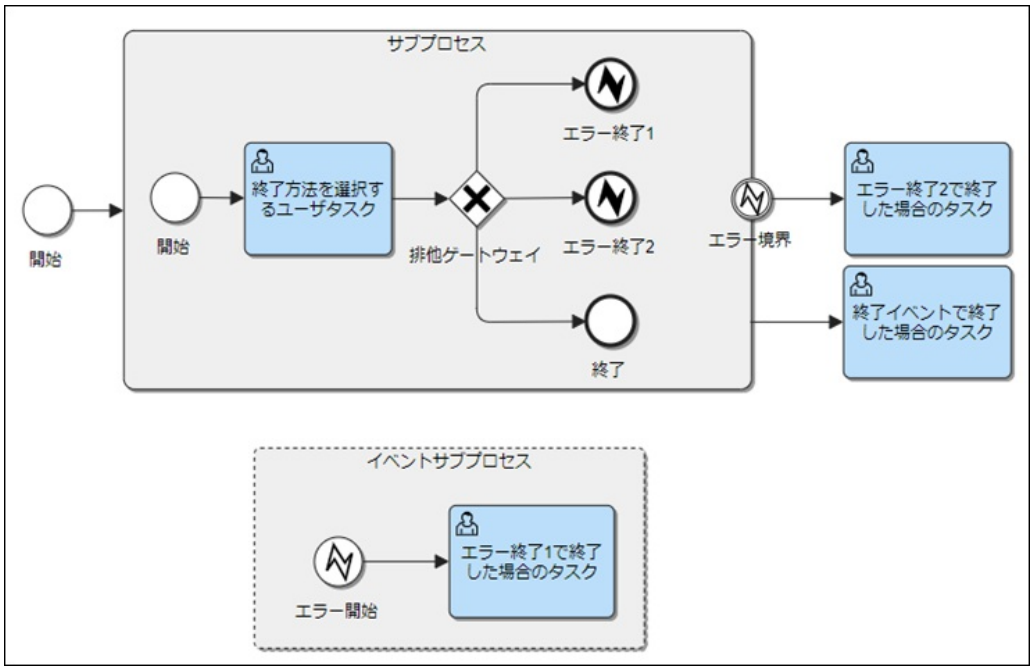
- プロセス定義を作成する
- 結果を確認する

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。

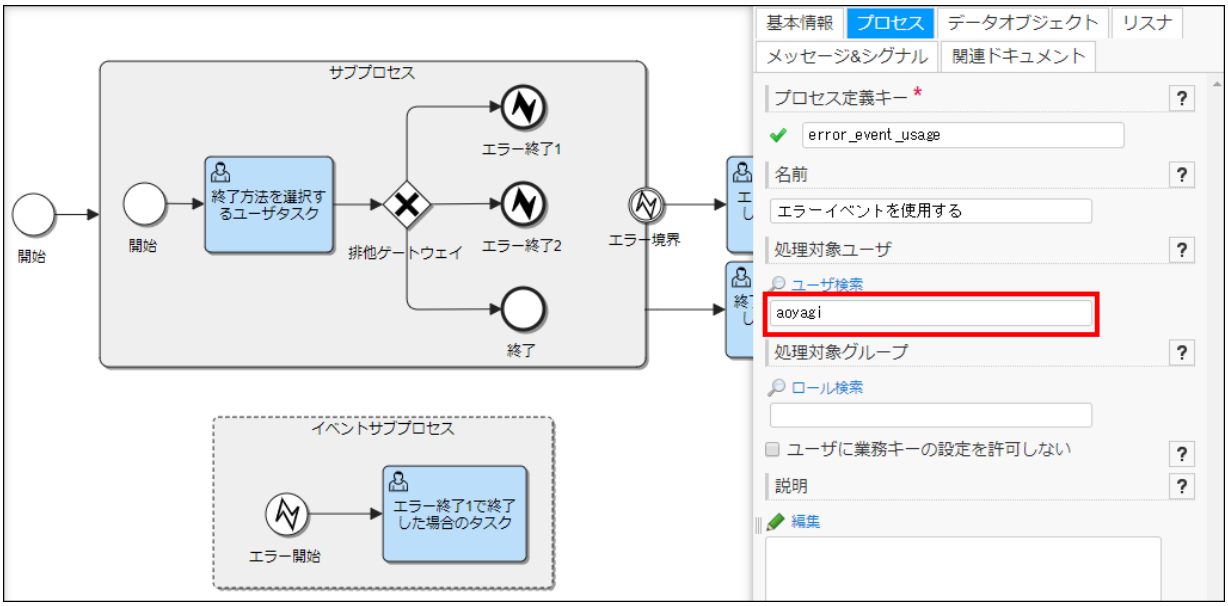


図：プロセス定義図「エラーイベントを使用する」

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。
   キャンバ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。



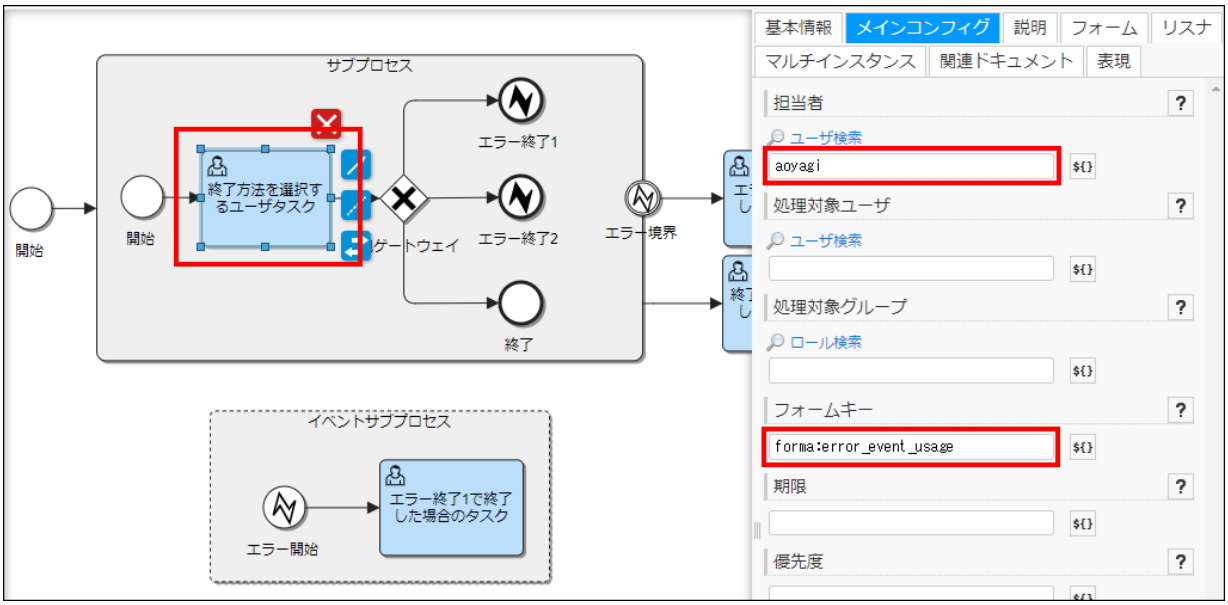
図：プロセス全体 - 「プロパティ」 - 「プロセス」 - 「処理対象ユーザ」

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

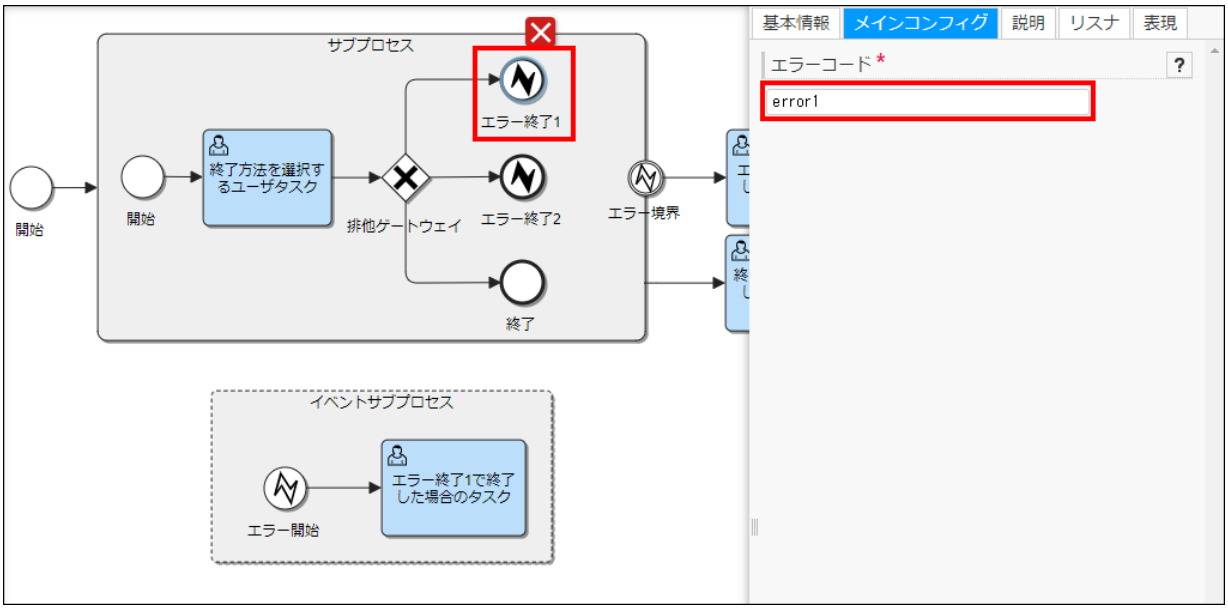


図：「サブプロセス」 - 「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

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。

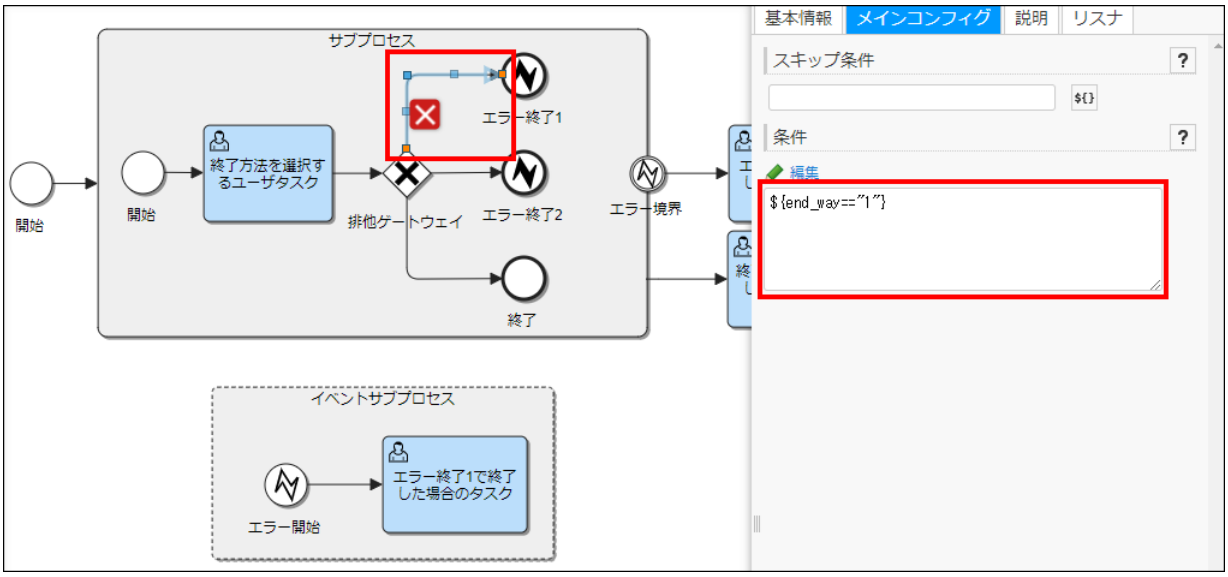
8. 「エラー終了1」からイベントサブプロセス内の「エラー開始」へ進行するルートを作成します。
   「エラー終了イベント」を設置します。

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。

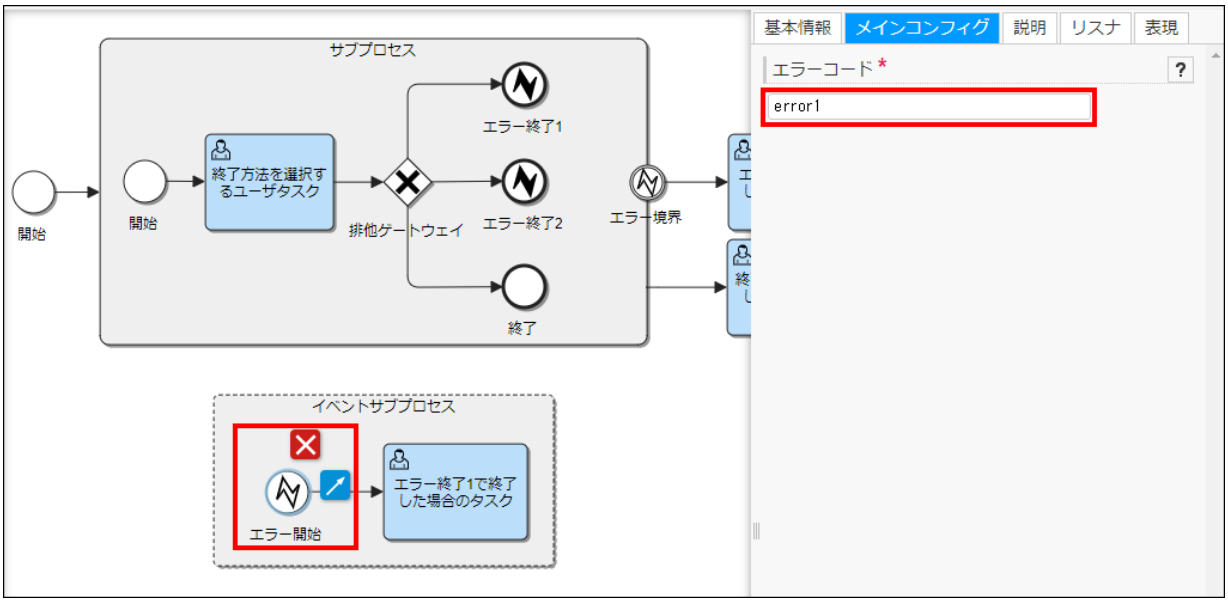


図：「サブプロセス」 - 「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」 - 「条件」

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。

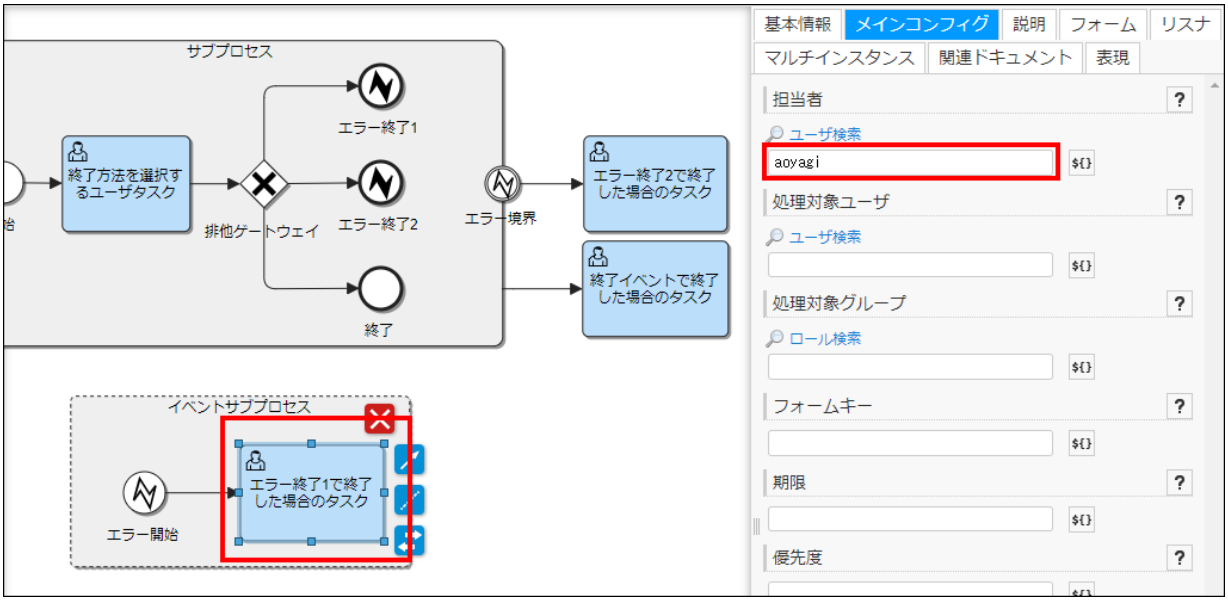
13. 「エラー終了1」に設定したエラーコードで紐づけます。
「エラー開始イベント」を選択した状態で、「メインコンフィグ」タブから「エラーコード」に「 `error1` 」と設定します。



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。

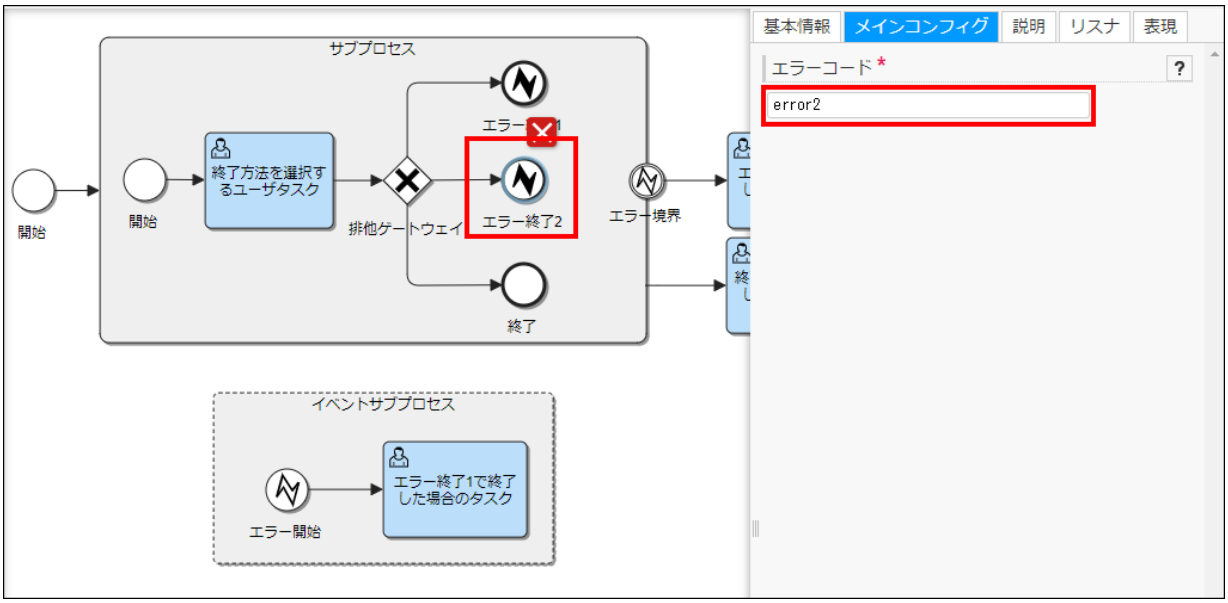
15. 「エラー終了1で終了した場合のタスク」の担当者を設定します。
「ユーザタスク」を選択した状態で「メインコンフィグ」タブから、担当者に「 aoyagi 」と設定します。



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」

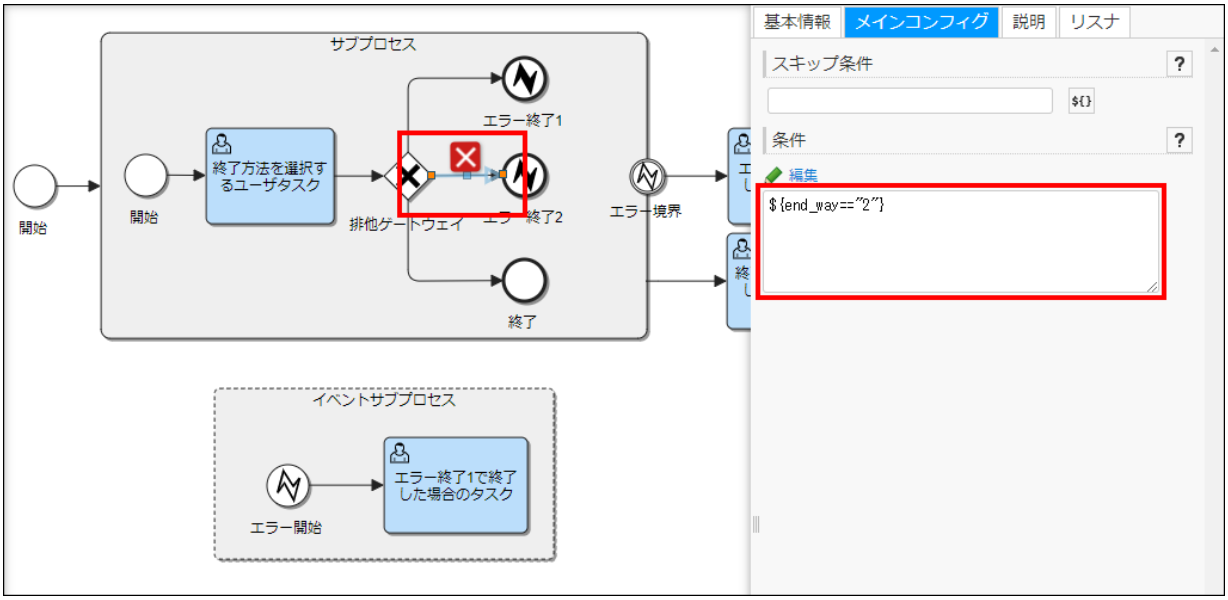
16. エラー終了イベントから、イベントサブプロセス内のエラー開始イベントへ進行するルートを作成します。
「エラー終了イベント」を設置します。

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」

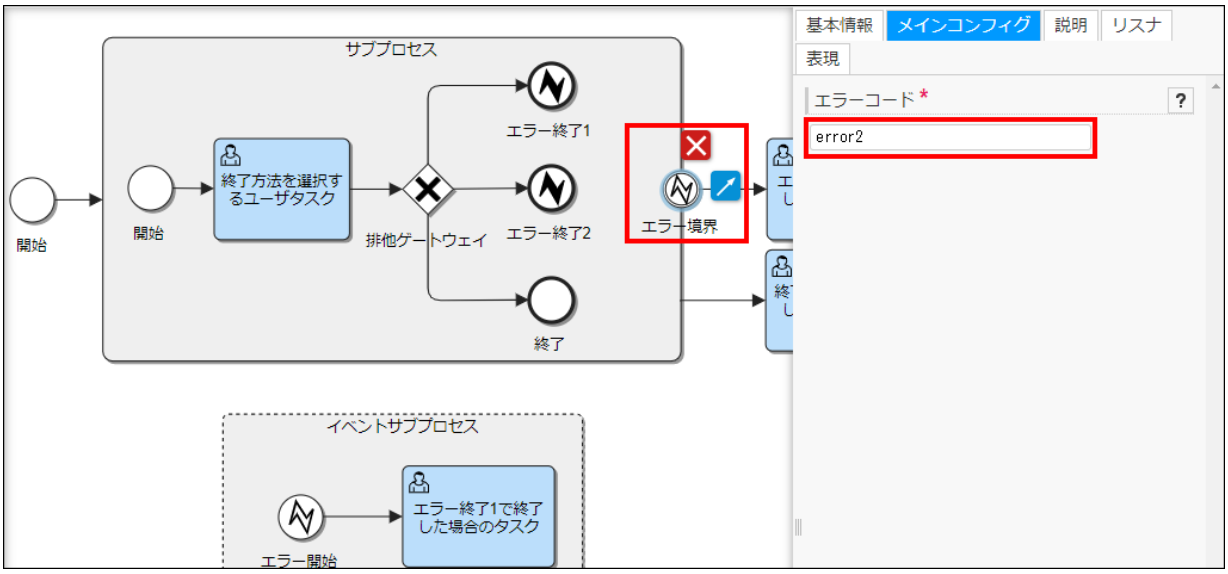
18. Forma画面で「エラー終了2」が指定された場合に「エラー終了2」へ進行するように条件付けをします。
「排他ゲートウェイ」から「エラー終了イベント」へ続く「シーケンスフロー」を選択します。
「メインコンフィグ」タブから、「条件」に `${end_way=="2"}` と設定します。



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。

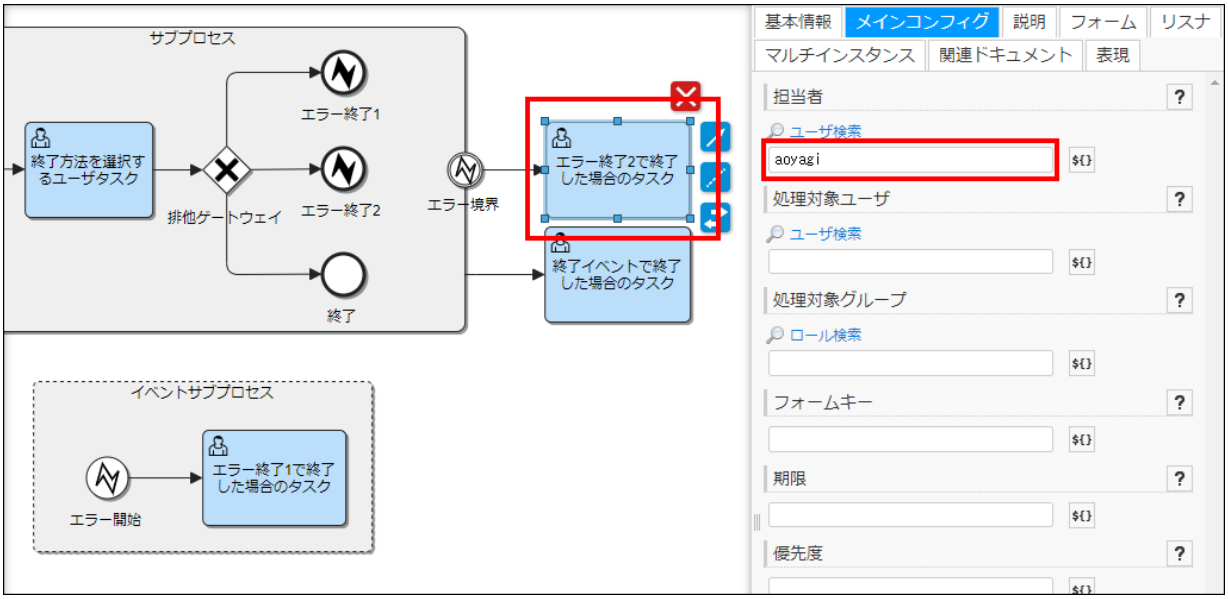
20. エラー終了イベントが送信するエラーコードを設定します。
「エラー境界イベント」を選択した状態で、「メインコンフィグ」タブから「エラーコード」に「 `error2` 」と設定します。



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。

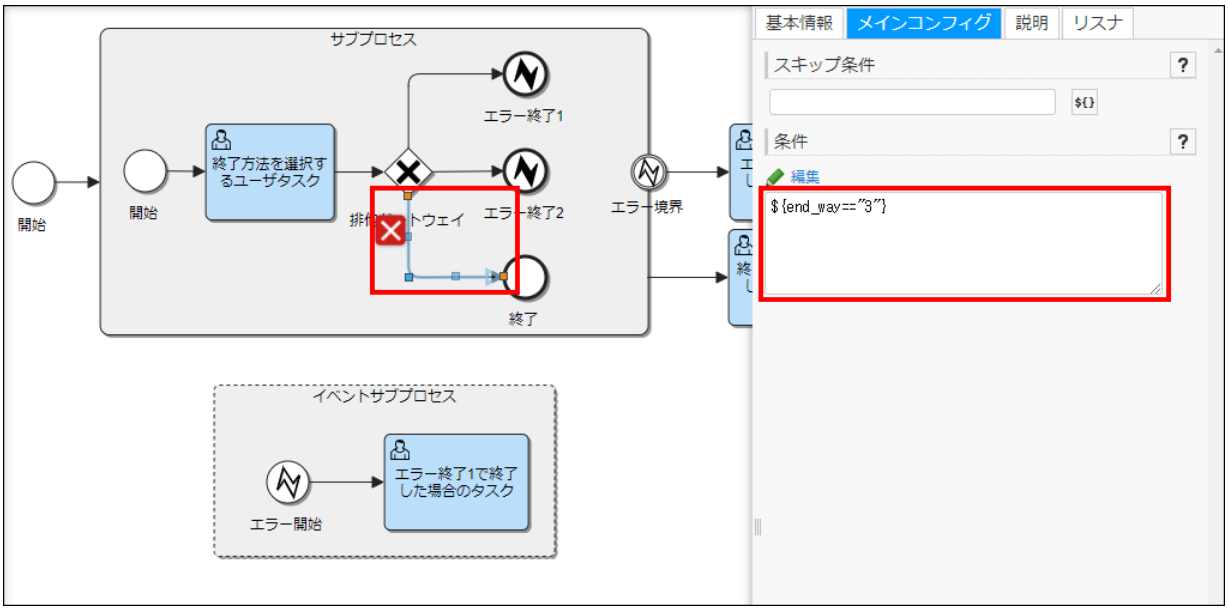
22. 「エラー終了2で終了した場合のタスク」の担当者を設定します。
「ユーザタスク」を選択した状態で「メインコンフィグ」タブから、担当者に「 aoyagi 」と設定します。



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。

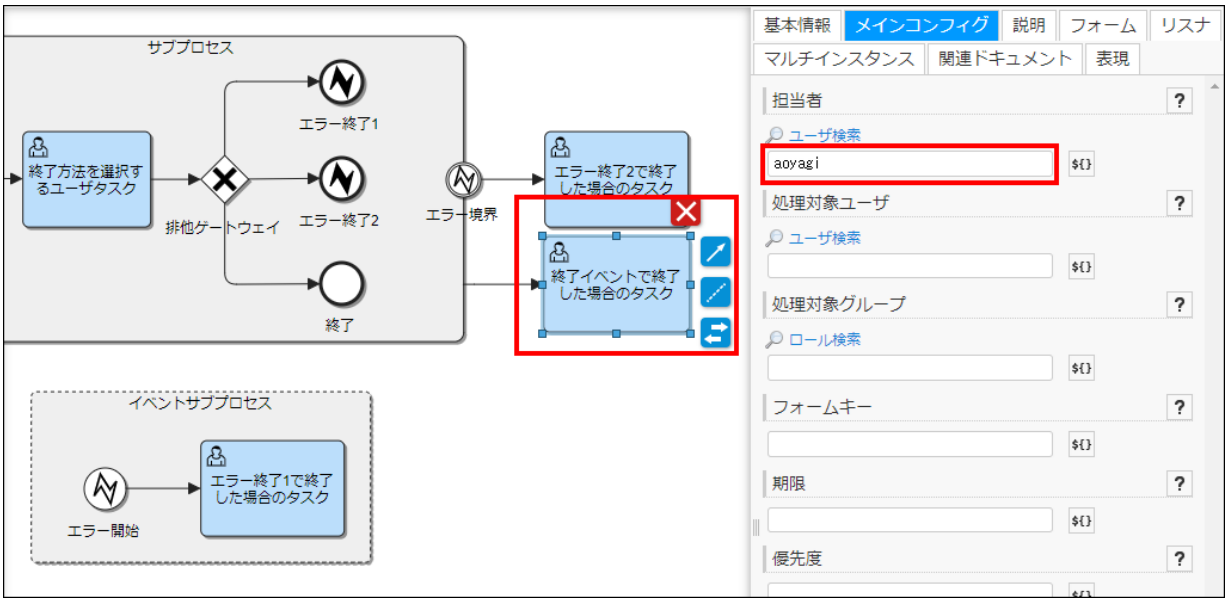
24. Forma画面で終了イベントが指定された場合に、終了イベントへ進行するように条件付けをします。
「排他ゲートウェイ」から「終了イベント」へ続く「シーケンスフロー」を選択します。
「メインコンフィグ」タブから、「条件」に `${end_way=="3"}` と設定します。



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」

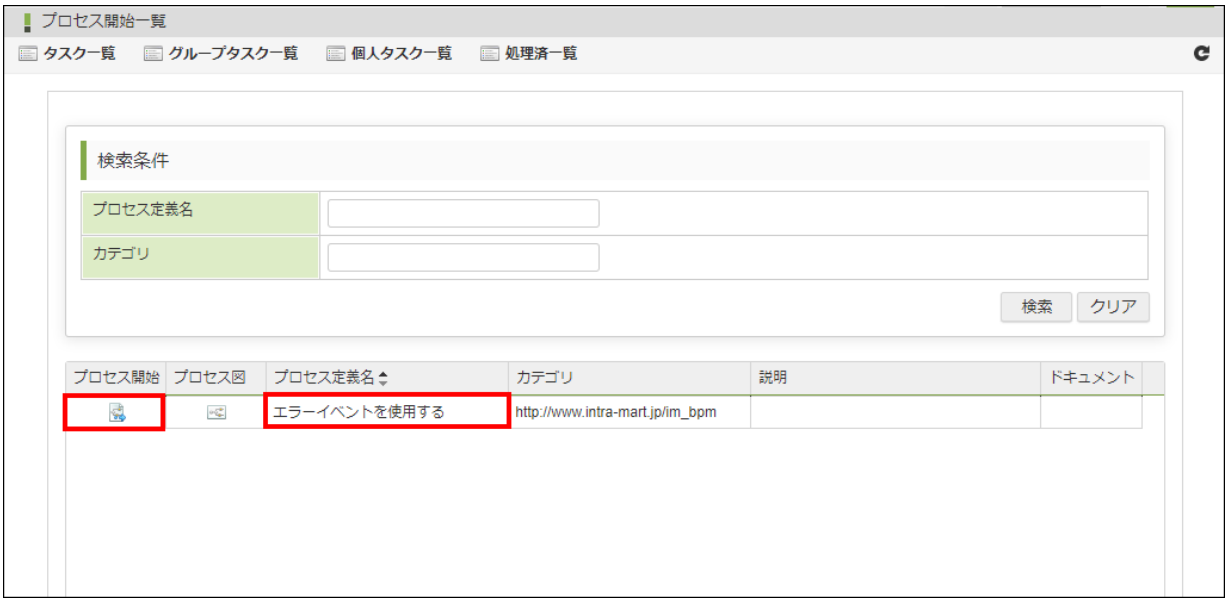
結果を確認する

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。



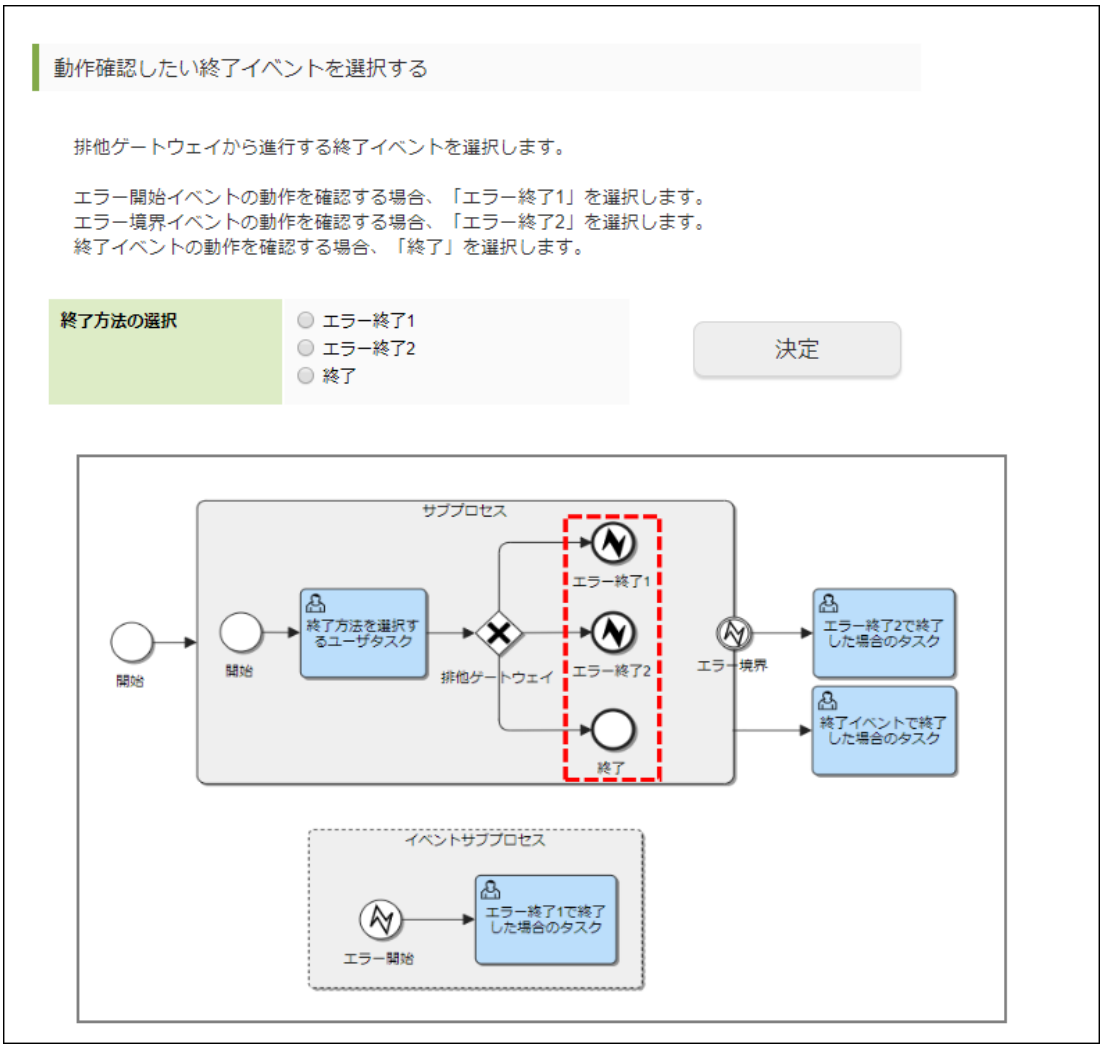
図：「プロセス開始一覧」

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「」アイコンをクリックして実行します。

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。

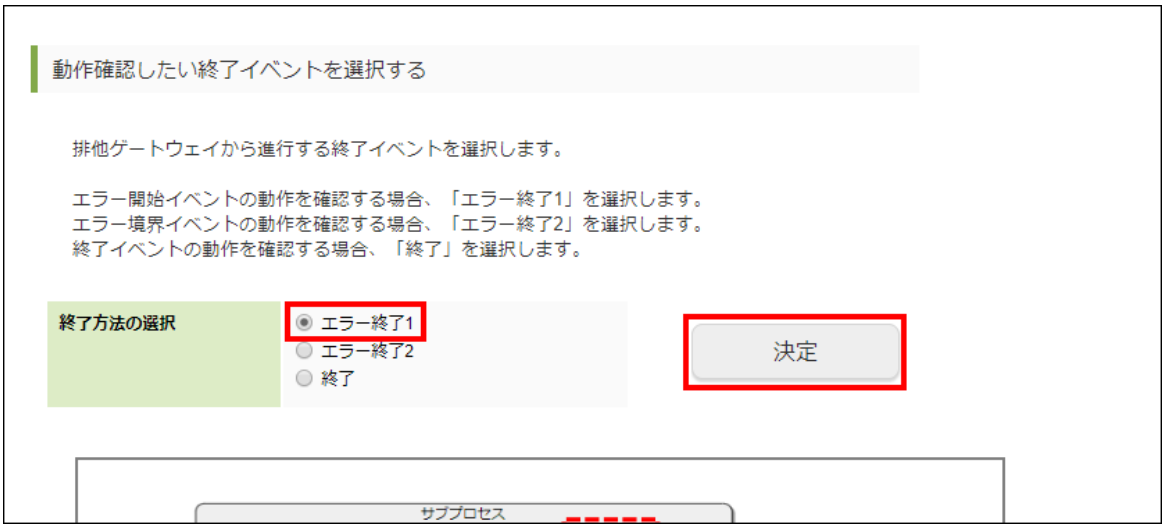


図：動作確認したい終了イベントを選択する(IM-FormaDesigner)

「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。



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。

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

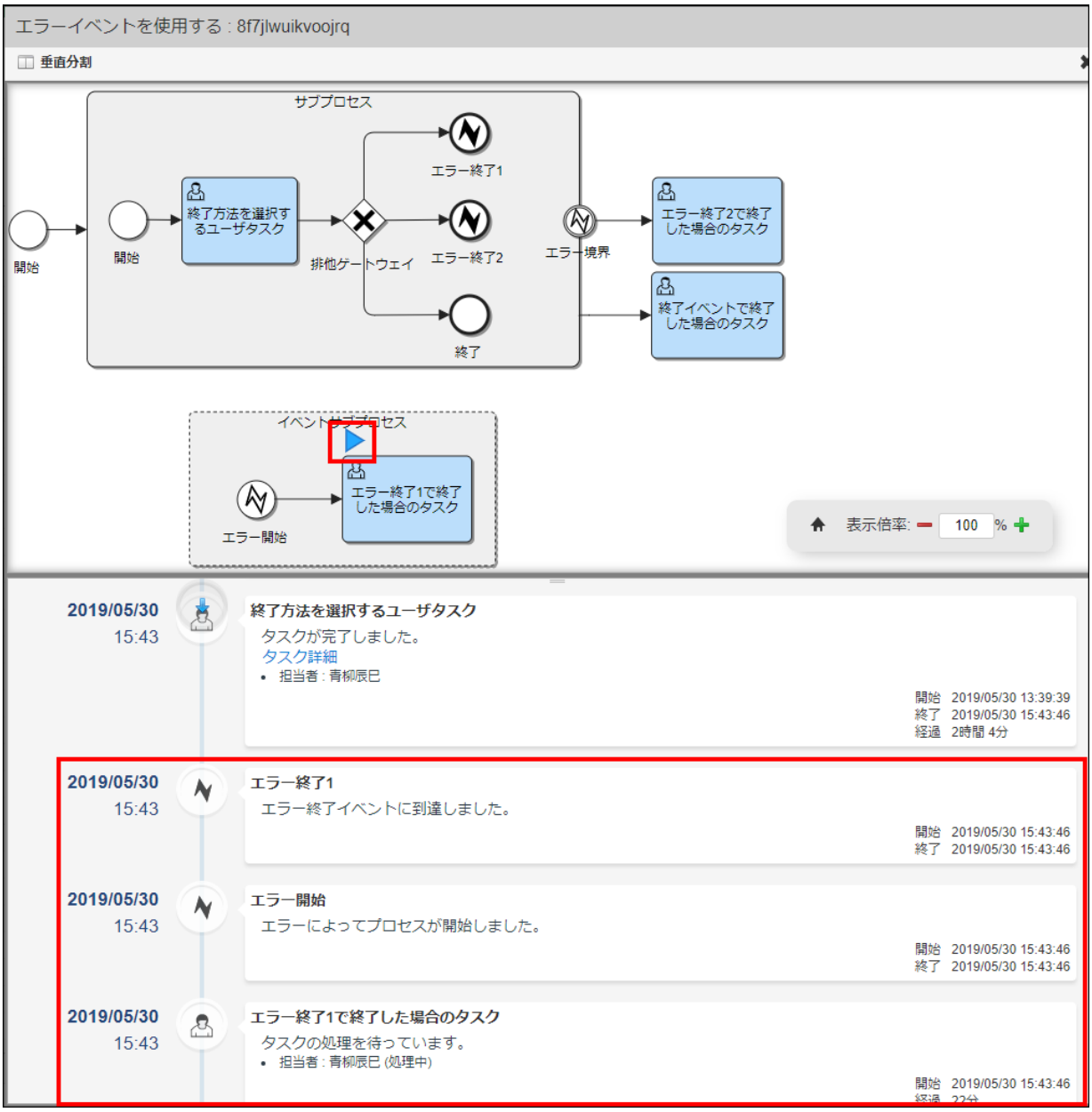
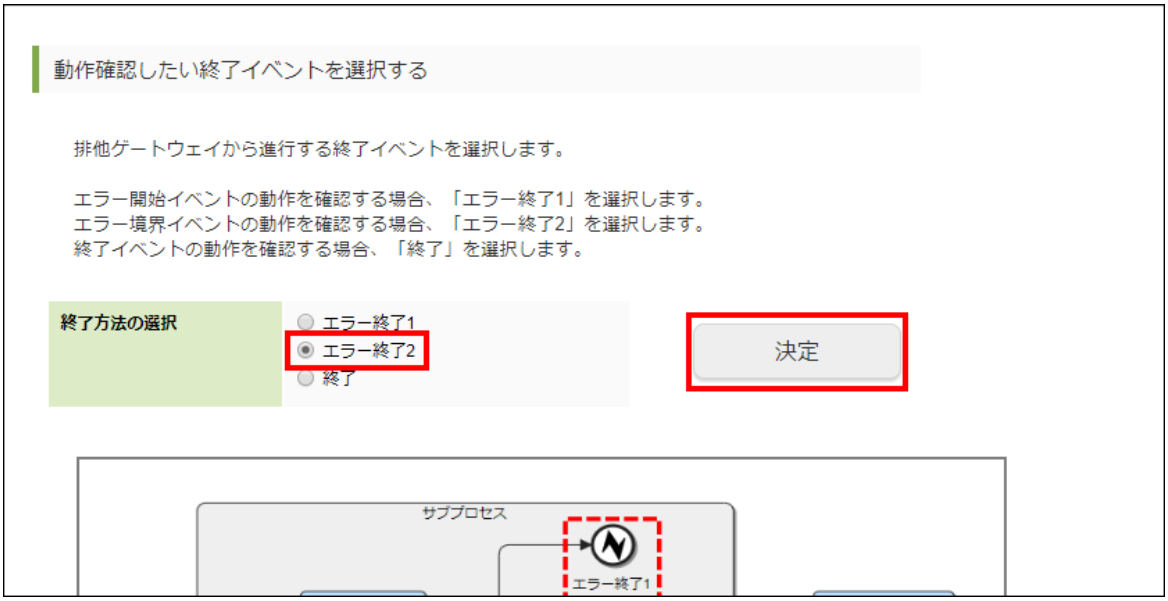


図:「プロセス詳細画面」

「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。



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。

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

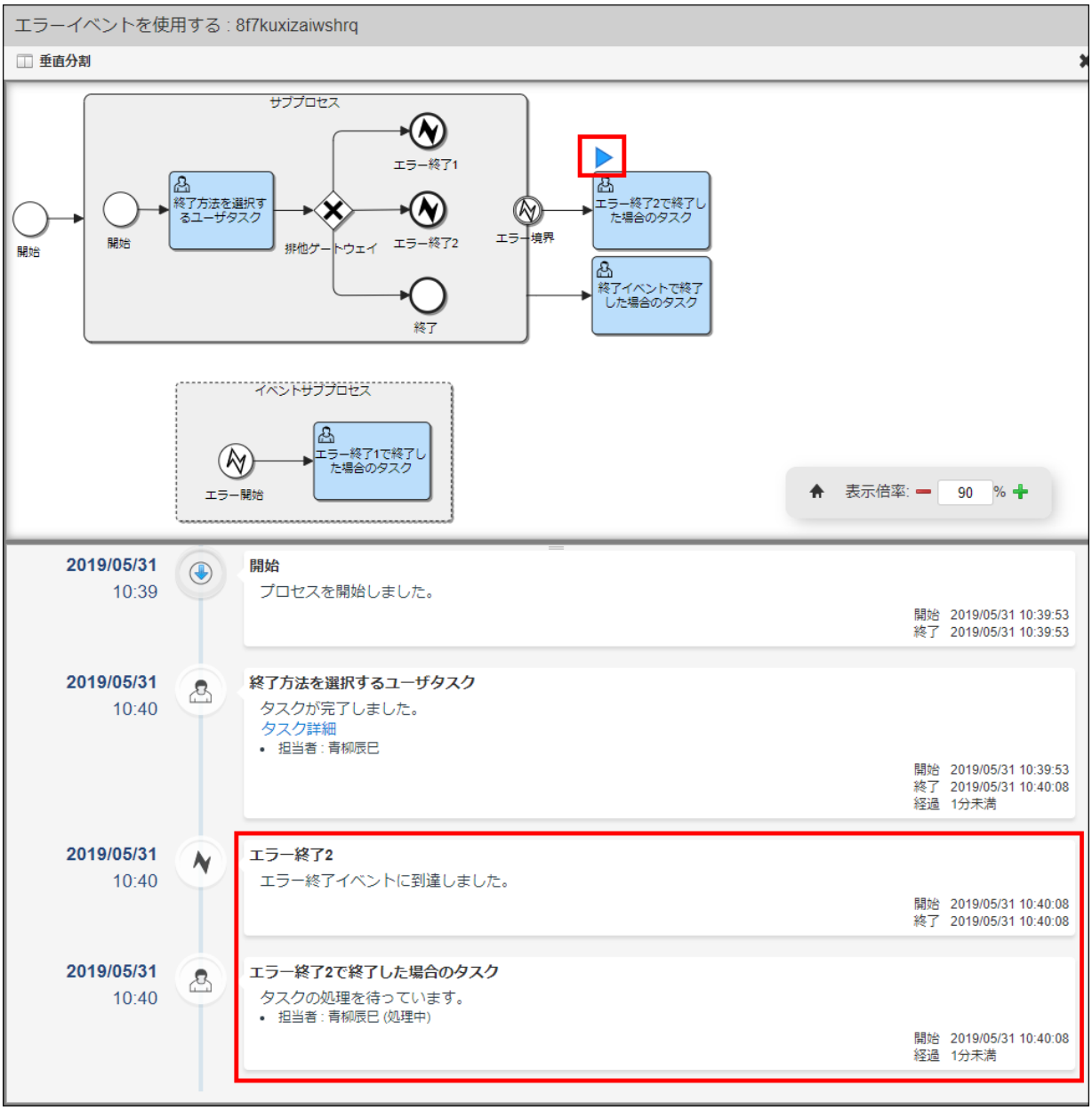


図: 「プロセス詳細画面」

「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。

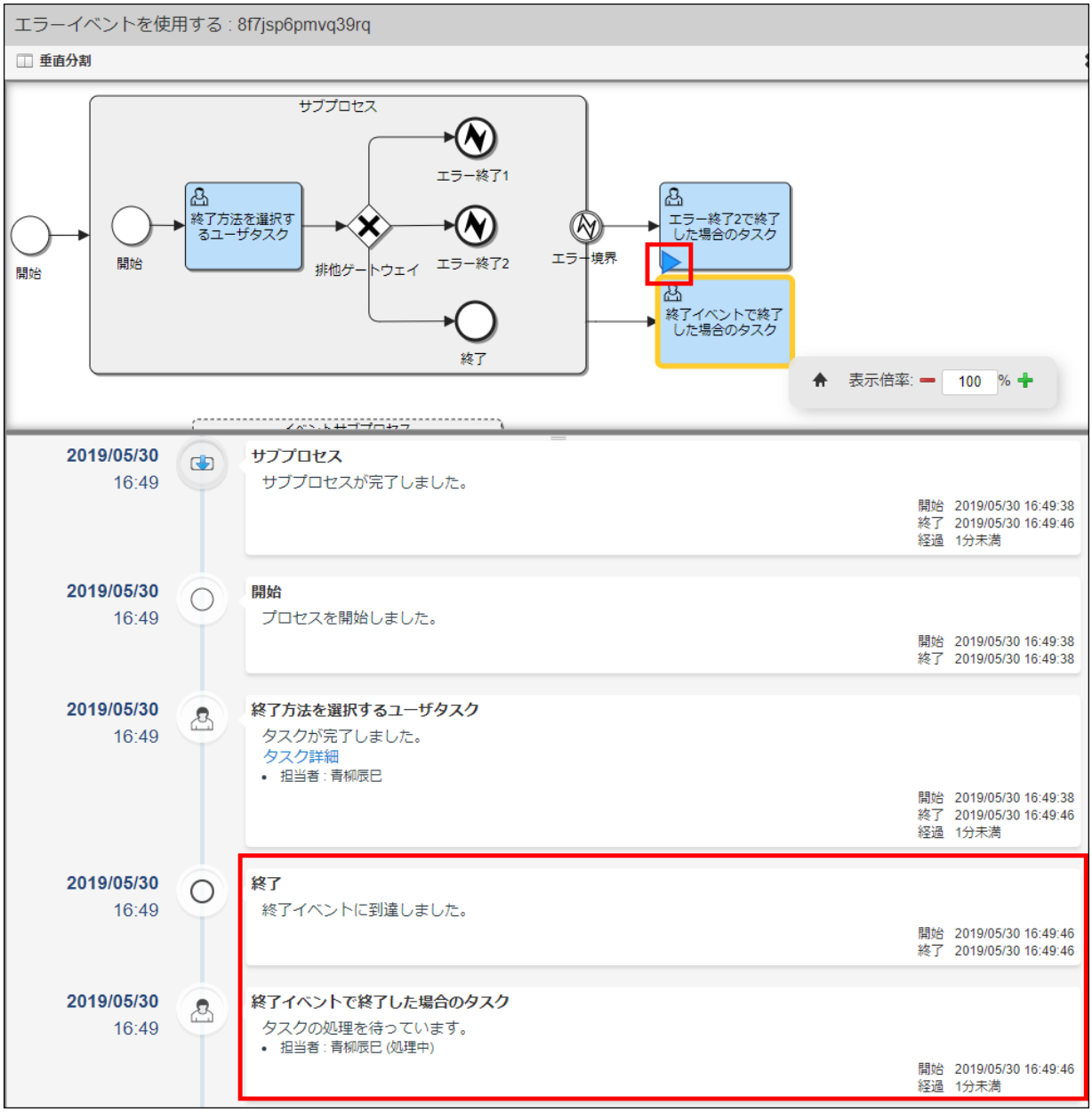


図:「プロセス詳細画面」

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。

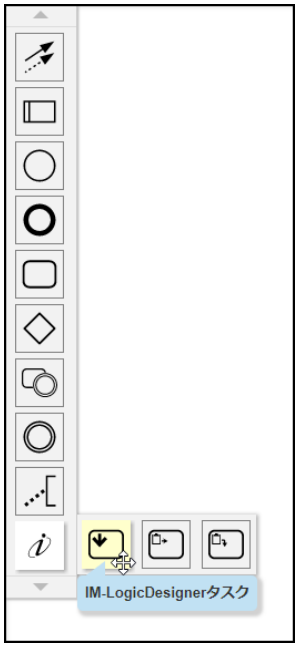
- 「IM-LogicDesigner」とは、intra-mart Accel Platform上でビジネスロジックを簡単に作成できるアプリケーションです。
- 「IM-LogicDesignerタスク」は、タスク中に設定された情報からIM-LogicDesignerで定義されたロジックフローの処理を行います。

「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 操作ガイド」 - 「IM-LogicDesignerタスク」もあわせて参照してください。

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

- IM-LogicDesignerタスクを配置する
- プロセスグローバルのデータ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。

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。

   > **コラム**
   >
   > フローIDを直接指定している場合は、「フロー定義編集」より指定のフローを編集できます。

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

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

   - **最新バージョンを利用**

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

   - **入力したバージョンを利用**

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

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

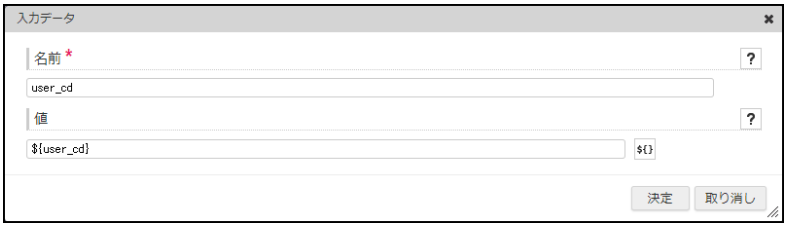
入力データを設定する

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

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

1. 「追加」をクリックし、ダイアログを開きます。

2. 「名前」に user_cd を入力し、「値」に ${user_cd} を入力します。



図：入力データダイアログ

3. 「決定」ボタンをクリックして、入力データの表に反映します。

実行結果の受け取りについて設定する

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

1. 「結果変数を格納する」チェックボックスをチェックONにします。

2. 「結果変数名」に result_data を入力します。
ロジックフローの実行後、プロセスグローバルの変数 result_data が作成され、ロジックフローの出力値 records が格納されます。

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

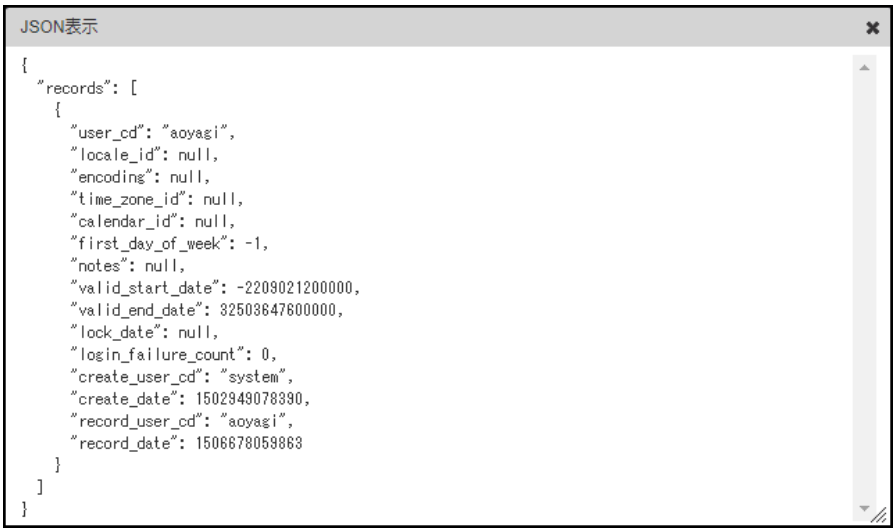
> **注意**
> 「結果変数を格納する」チェックボックスをオフにした場合、実行したロジックフローの出力値はすべて破棄されます。

実行結果を確認する

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

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

2. 「変数一覧」画面にて、変数 result_data の内容を確認します。



図：「変数一覧」画面 - JSON表示

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

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

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

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

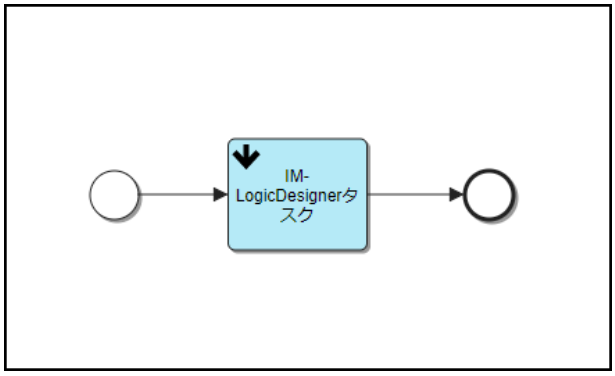
- 事前準備
- エレメントを配置する
- IM-LogicDesignerタスクのプロパティを設定する
- 開始イベントにFormaアプリケーションを設定する
- 実行結果を確認する

事前準備

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

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

エレメントを配置する



図：配置イメージ

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

IM-LogicDesignerタスクのプロパティを設定する

図：プロパティ入力イメージ

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

開始イベントにFormaアプリケーションを設定する

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

実行結果を確認する

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

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

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

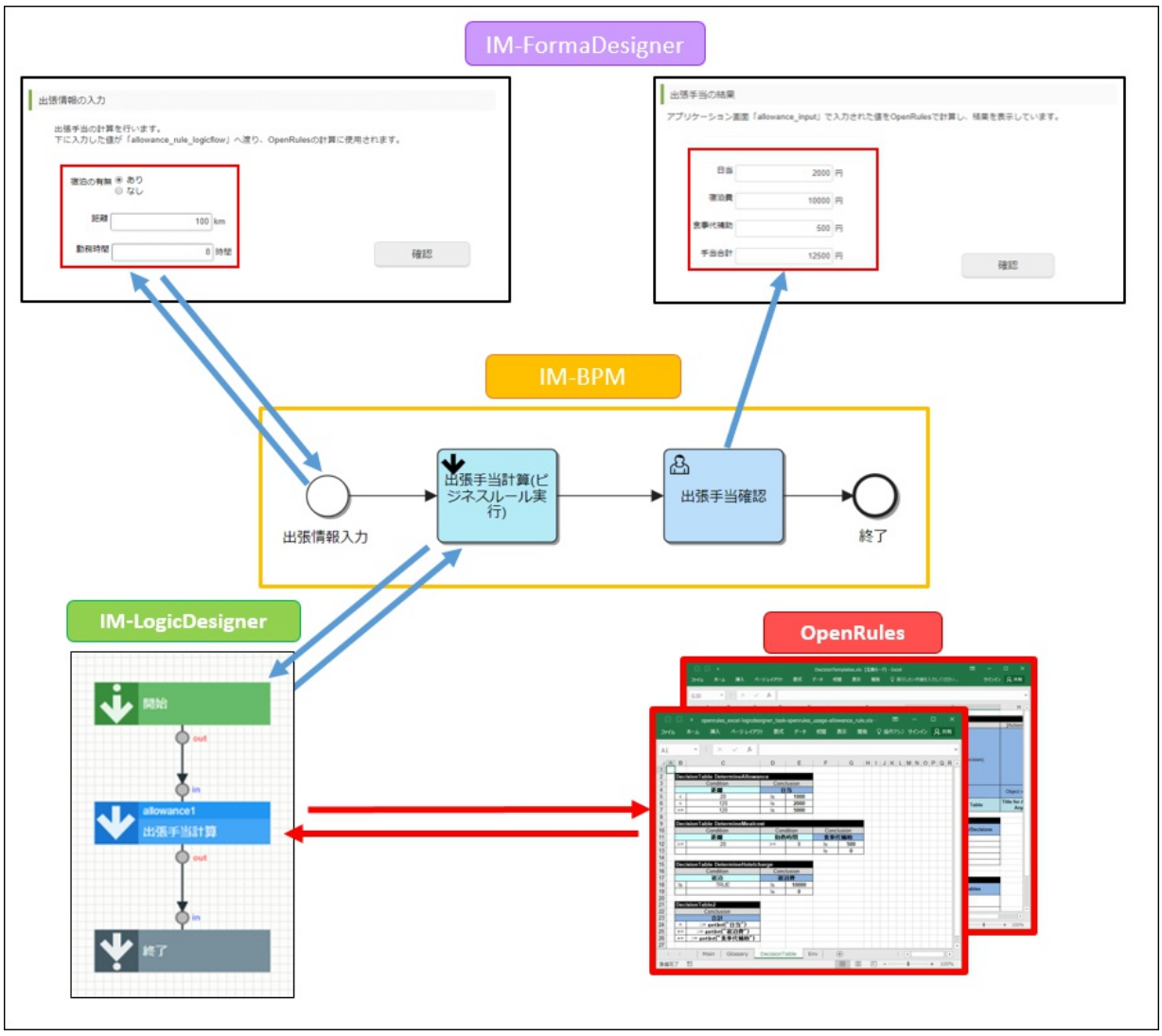
コラム

- プロセスのデプロイ方法については「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義/ケース定義のデプロイ」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。

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 ユーサ゛・マニュアル」参照してください。
「IM-LogicDesignerタスク」の基本的な使用方法については「*IM-LogicDesignerタスクを利用してユーザ情報を取得する*」を参照してください。



図：概要

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

openrules_excel-logicdesigner_task_openrules_usage-allowance_rule.xls

プロセスを進めていく中で、「IM-LogicDesigner」のロジックフローと「IM-FormaDesigner for Accel Platform」で作成した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材をインポートしてください。

- ロジックフロー

  im_logic_designer-logicdesigner_task_openrules_usage-allowance_rule_logic_flow.zip

- Formaアプリケーション

  im_forma_designer-logicdesigner_task_openrules_usage-allowance_input.zip
  im_forma_designer-logicdesigner_task_openrules_usage-allowance_confirm.zip

コラム

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

logicdesigner_task_openrules_usage-allowance.bpmn

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

コラム

各種インポート方法については、以下のリンクを参照してください。

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

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

環境構築を行う

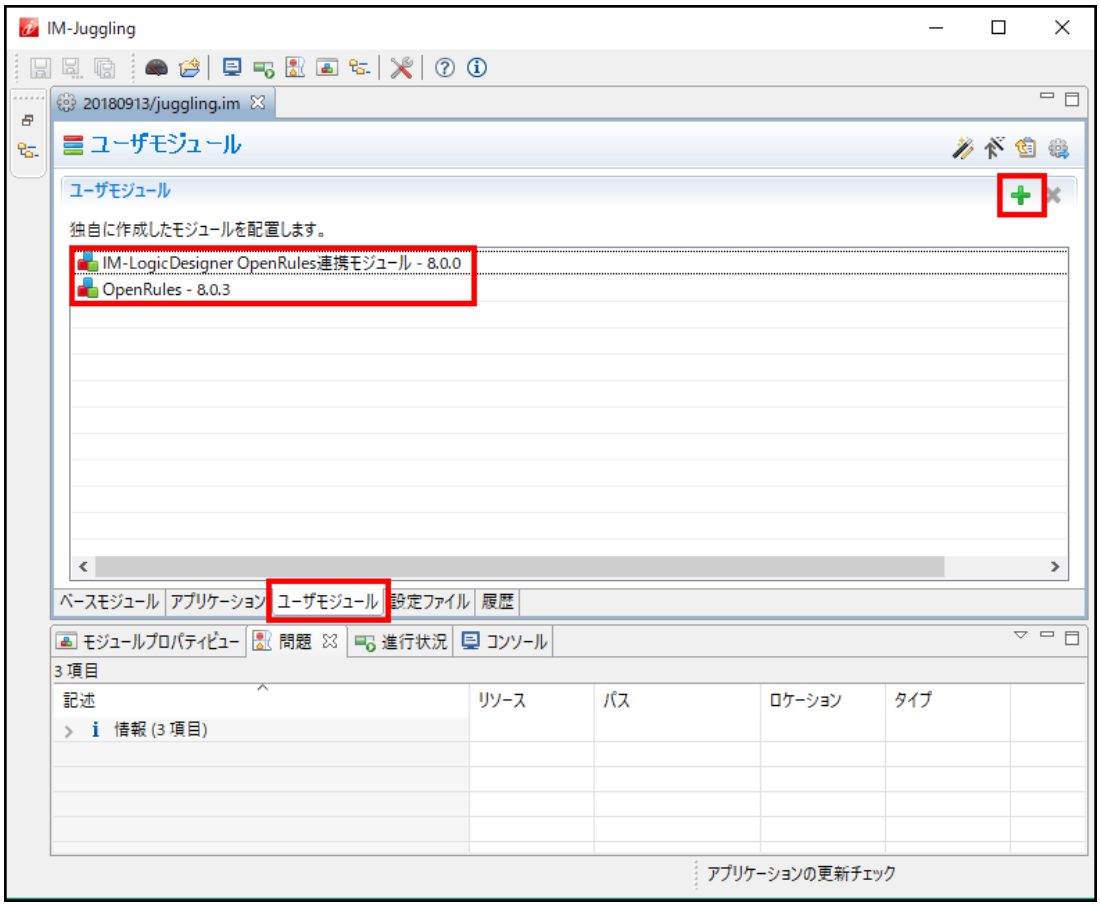
「IM-LogicDesigner」で「OpenRules」を使用するためには、専用の「ユーザモジュール」を追加する必要があります。
以下の手順のとおりに、環境構築をおこなってください。

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

   注意

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

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



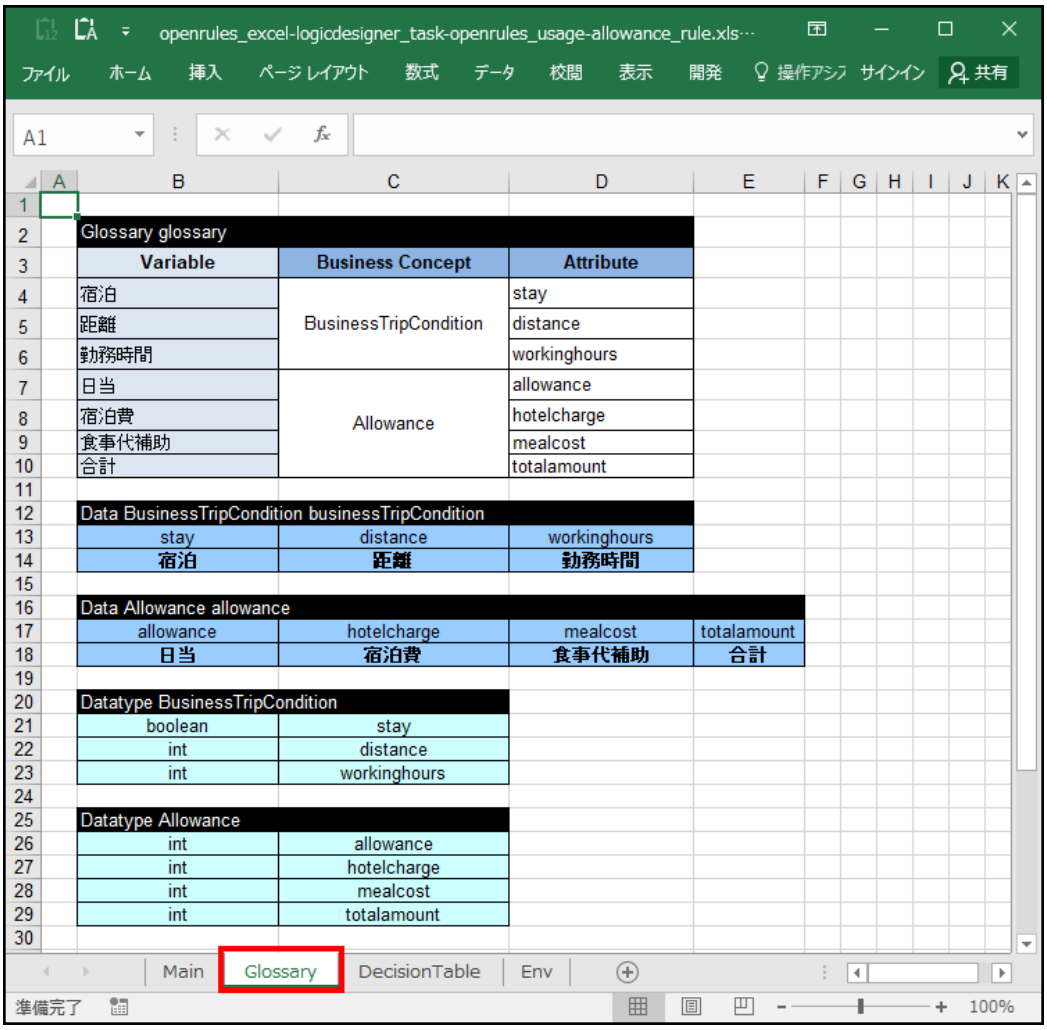
図：「IM-Juggling」 - 「ユーザモジュール」

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

EXCELファイルを確認・配置する

「OpenRules」で使用する「処理を実行するEXCELファイル」を確認・配置します。
このチュートリアルでは、「IM-FormaDesigner」の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。



Glossary glossary

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

Data BusinessTripCondition businessTripCondition

stay	distance	workinghours
宿泊	**距離**	**勤務時間**

Data Allowance allowance

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

Datatype BusinessTripCondition

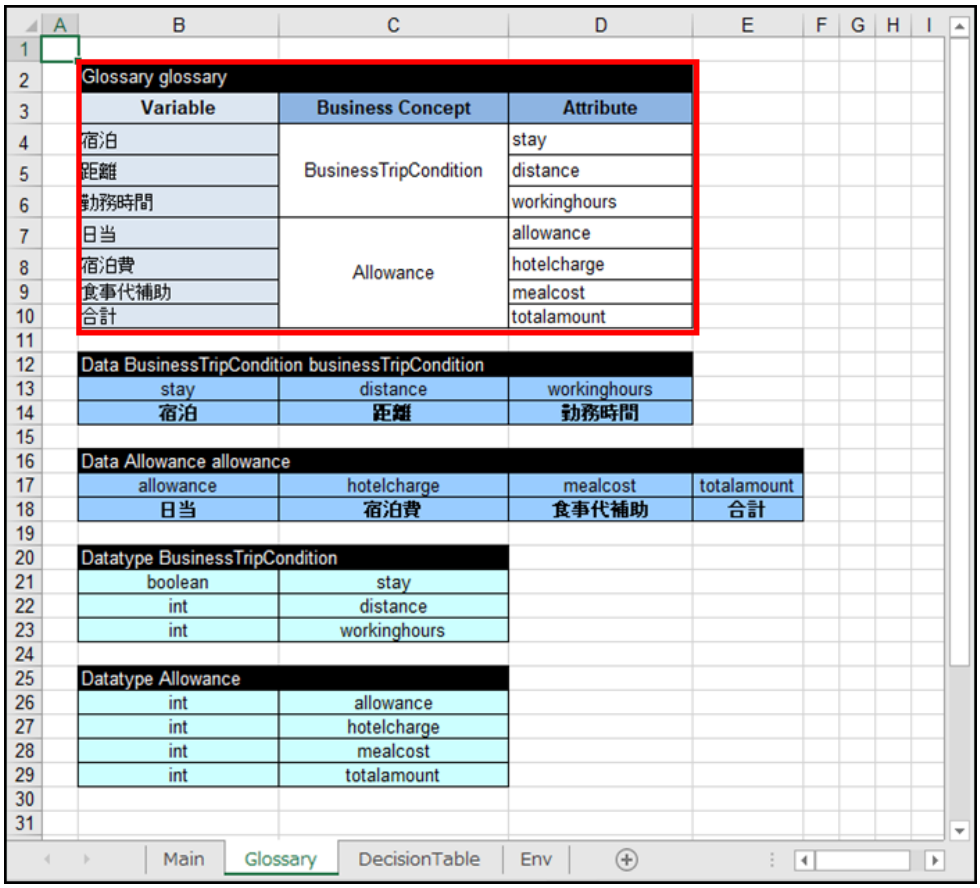
boolean	stay
int	distance
int	workinghours

Datatype Allowance

int	allowance
int	hotelcharge
int	mealcost
int	totalamount

図：「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。



Glossary glossary

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

Data BusinessTripCondition businessTripCondition

stay	distance	workinghours
宿泊	距離	勤務時間

Data Allowance allowance

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

Datatype BusinessTripCondition

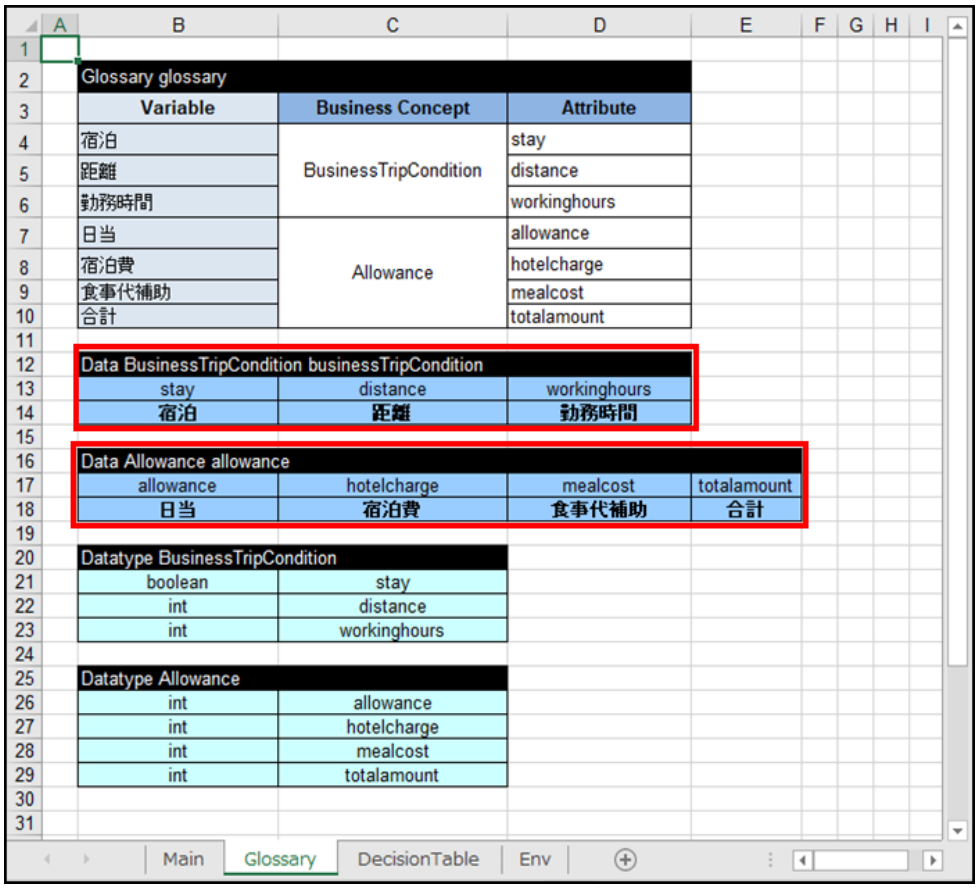
boolean	stay
int	distance
int	workinghours

Datatype Allowance

int	allowance
int	hotelcharge
int	mealcost
int	totalamount

図：「Glossary」シート - 「Glossary glossary」

- Data テーブル
  データを保持しているテーブルです。
  使用する「グループ（Business Concept）」と、「物理名（Attribute）」を設定します。
  - Data BusinessTripCondition businessTripCondition
    「IM-FormaDesigner」の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。



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

Data BusinessTripCondition businessTripCondition		
stay	distance	workinghours
宿泊	距離	勤務時間

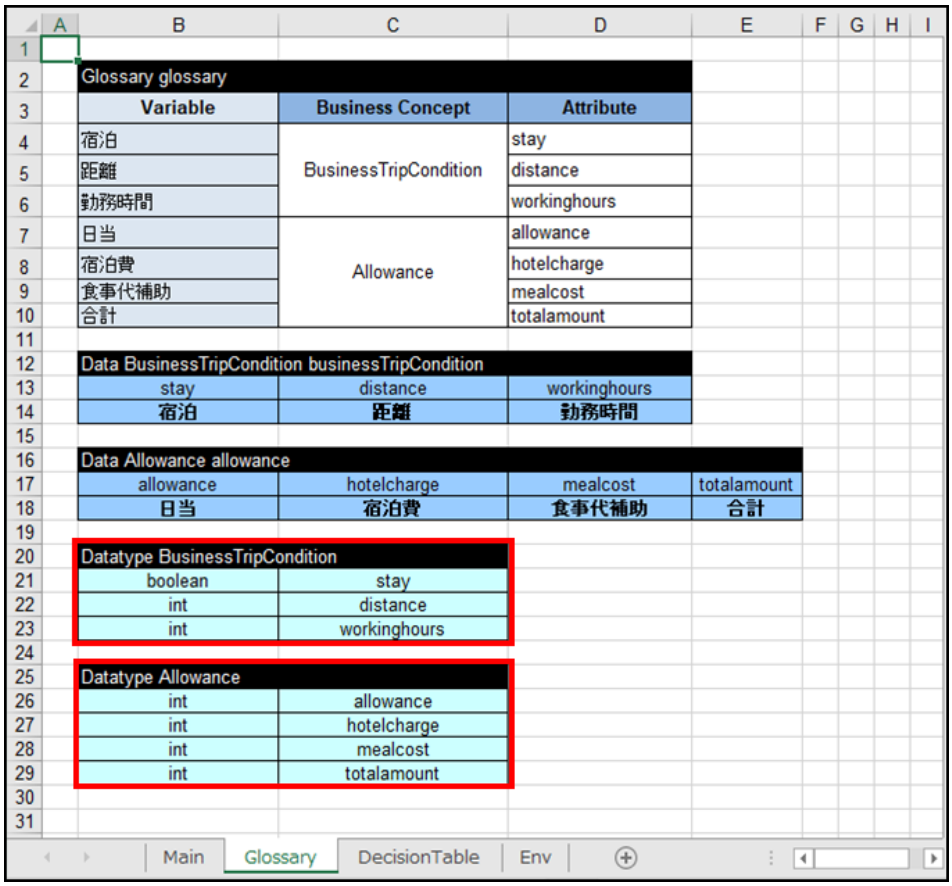
Data Allowance allowance			
allowance	hotelcharge	mealcost	totalamount
日当	宿泊費	食事代補助	合計

Datatype BusinessTripCondition	
boolean	stay
int	distance
int	workinghours

Datatype Allowance	
int	allowance
int	hotelcharge
int	mealcost
int	totalamount

図：「Glossary」シート - 「Data」テーブル

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



Glossary glossary

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

Data BusinessTripCondition businessTripCondition

stay	distance	workinghours
宿泊	距離	勤務時間

Data Allowance allowance

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

Datatype BusinessTripCondition

boolean	stay
int	distance
int	workinghours

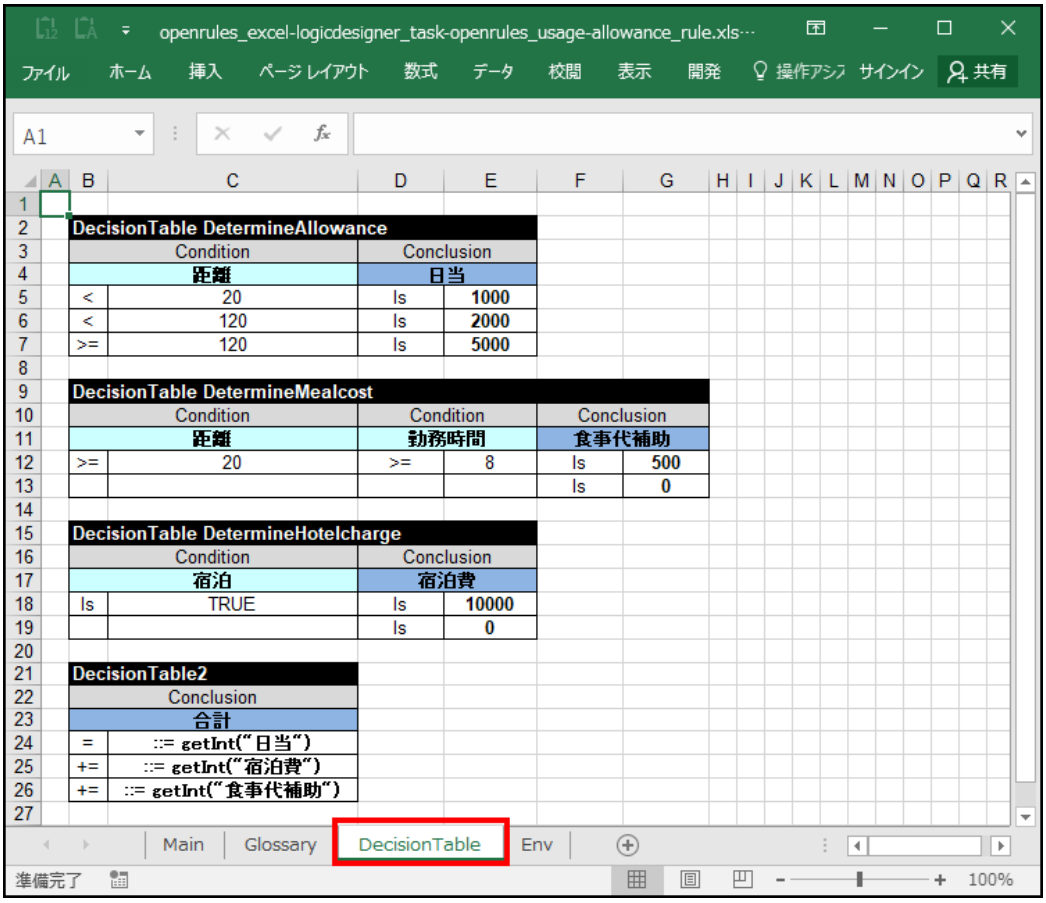
Datatype Allowance

int	allowance
int	hotelcharge
int	mealcost
int	totalamount

Main | Glossary | DecisionTable | Env

図：「Glossary」シート - 「Datatype」テーブル

2. 3枚目のシートの「DecisionTable」を確認します。



DecisionTable DetermineAllowance

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

DecisionTable DetermineMealcost

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

DecisionTable DetermineHotelcharge

Condition		Conclusion	
宿泊		宿泊費	
Is	TRUE	Is	10000
		Is	0

DecisionTable2

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

Main | Glossary | DecisionTable | Env

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

- DecisionTableテーブル

実行の条件と設定する値が定義されているテーブルです。

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

DecisionTable DetermineAllowance			
Condition		Conclusion	
距離		日当	
<	20	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		Condition		Conclusion	
距離		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	10000
		Is	0

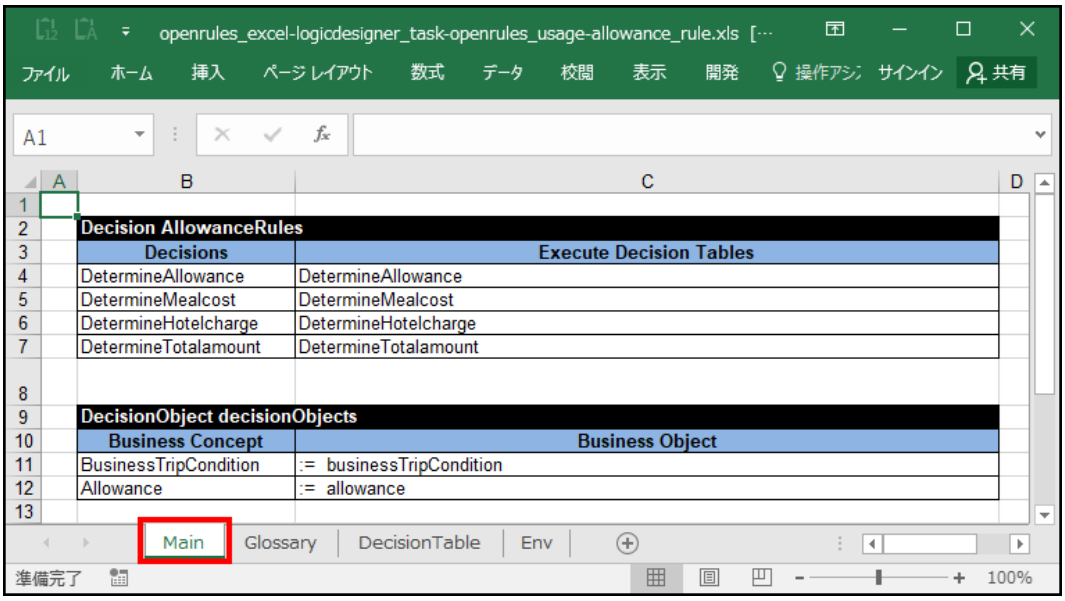
図：「DecisionTable」テーブル - 「DetermineHotelcharge」

- 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")

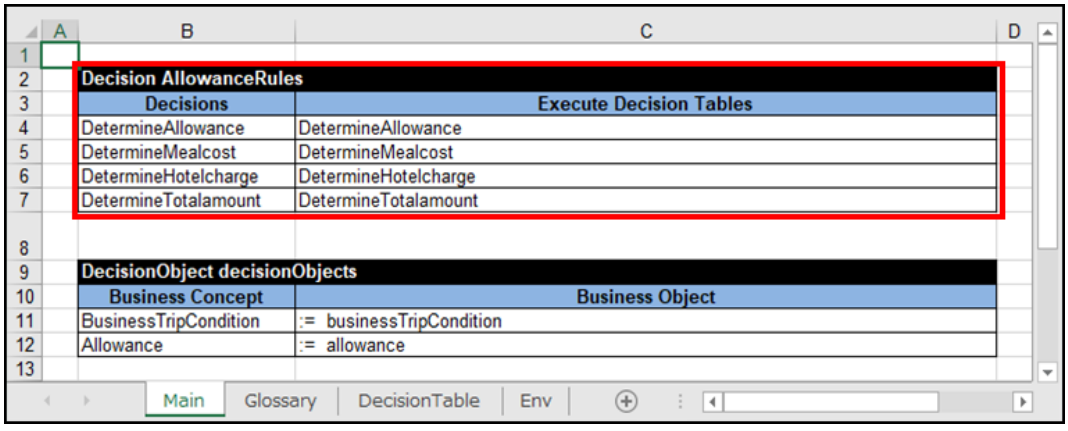
図：「DecisionTable2」テーブル - 「DetermineTotalamount」

3. 1枚目のシートの「Main」を確認します。



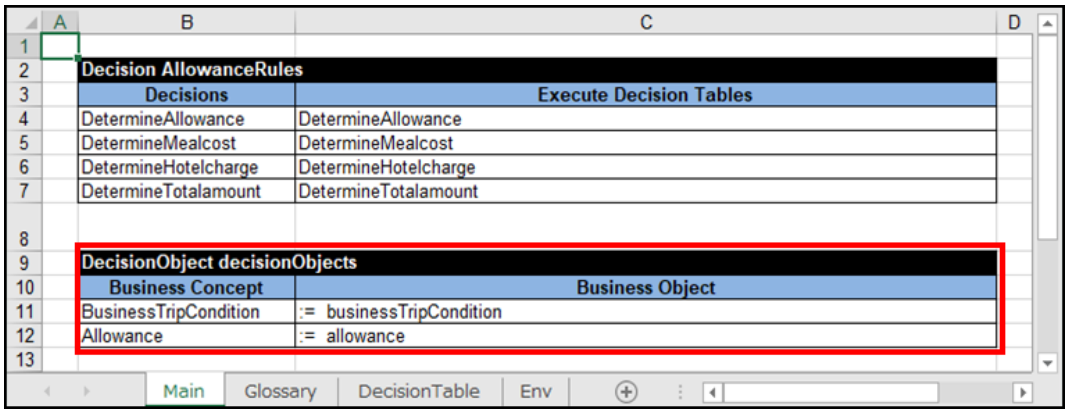
図：「openrules_excel-logicdesigner_task_openrules_usage-allowance_rule.xls」 - 「Main」シート

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



図：「Decision」テーブル - 「AllowanceRules」

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

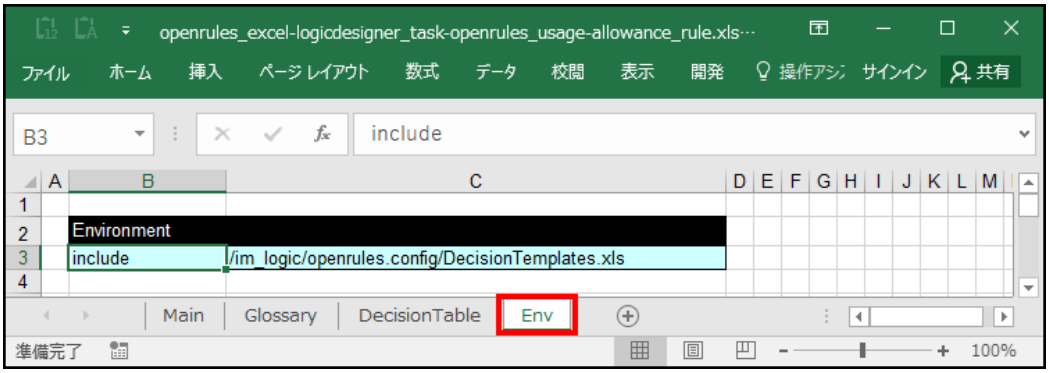


Decision AllowanceRules	
Decisions	Execute Decision Tables
DetermineAllowance	DetermineAllowance
DetermineMealcost	DetermineMealcost
DetermineHotelcharge	DetermineHotelcharge
DetermineTotalamount	DetermineTotalamount

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

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

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



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

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

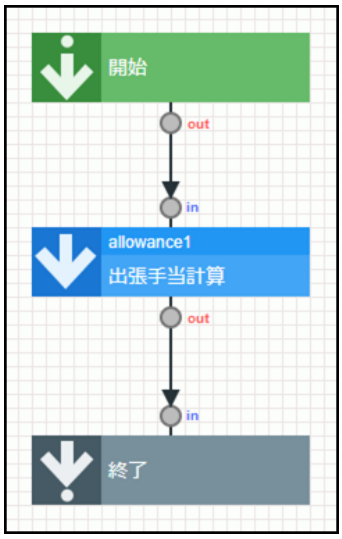
5. EXCELファイルの確認が終わったら、ファイルを「パブリックストレージ直下」に配置します。

図：「パブリックストレージ」直下

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

ロジックフローを確認する

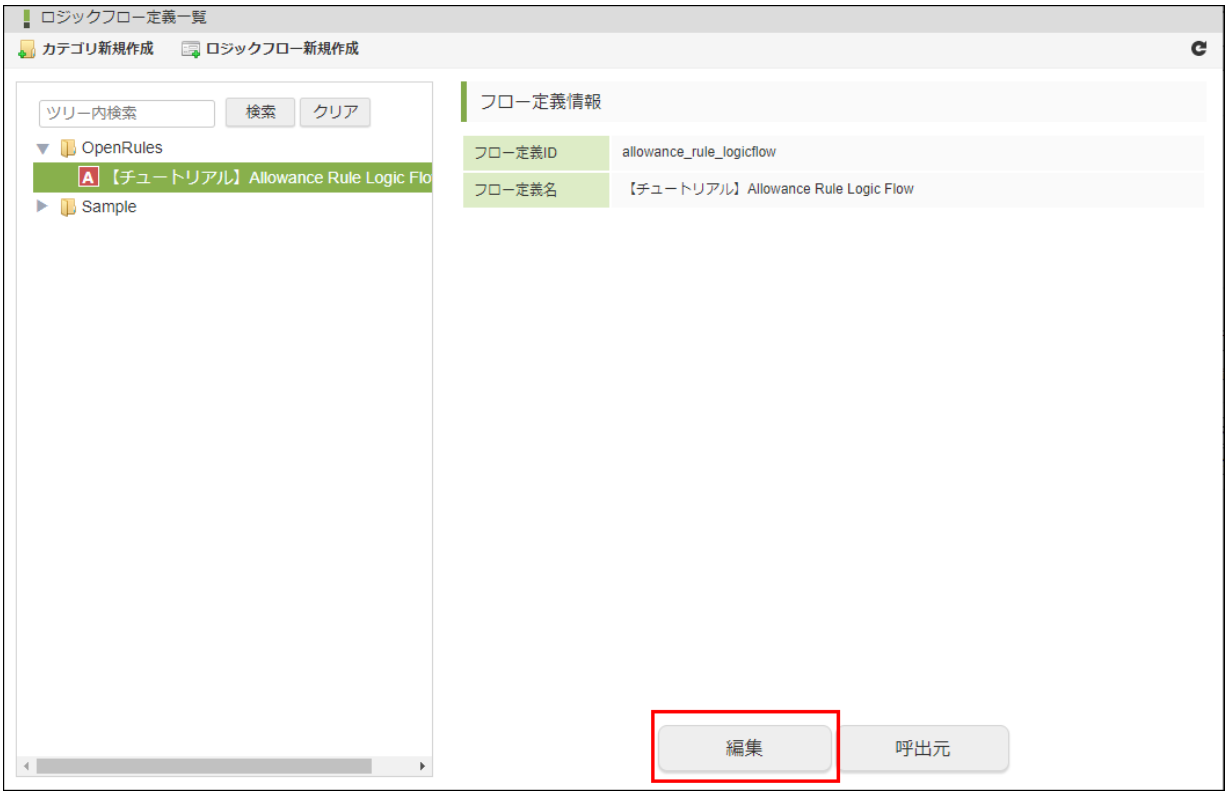
「IM-LogicDesigner」のロジックフローを確認します。
このチュートリアルでは、「IM-FormaDesigner」のFormaアプリケーションに入力された値を「OpenRules」で計算し、結果を返します。



図：ロジックフロー

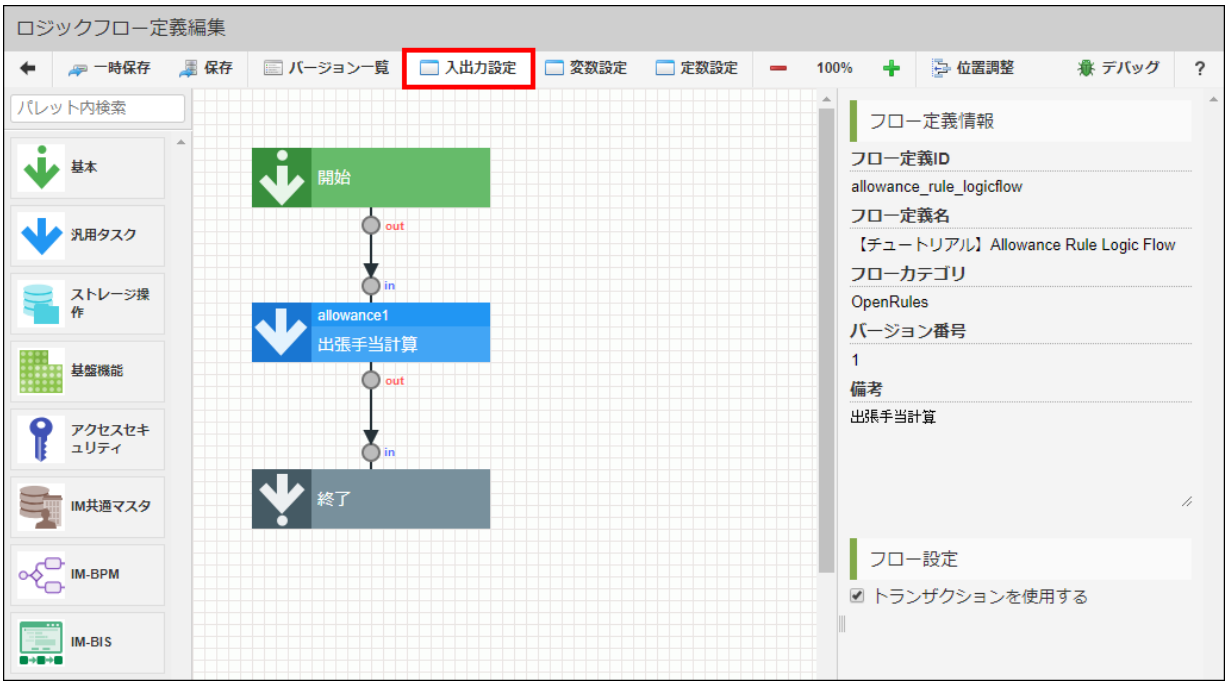
1. チュートリアル開始前にインポートしたロジックフローを確認します。
   「サイトマップ」→「IM-LogicDesigner」→「フロー定義一覧」→「ロジックフロー定義一覧」画面を表示します。

2. 左ペインのツリーから「OpenRules」カテゴリの開閉アイコンをクリックしカテゴリ配下を情報を表示します。
   フロー定義「【チュートリアル】Allowance Rule Logic Flow」を選択し編集ボタンをクリックし、「ロジックフロー定義編集」画面を開きます。



図：「ロジックフロー定義一覧」

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



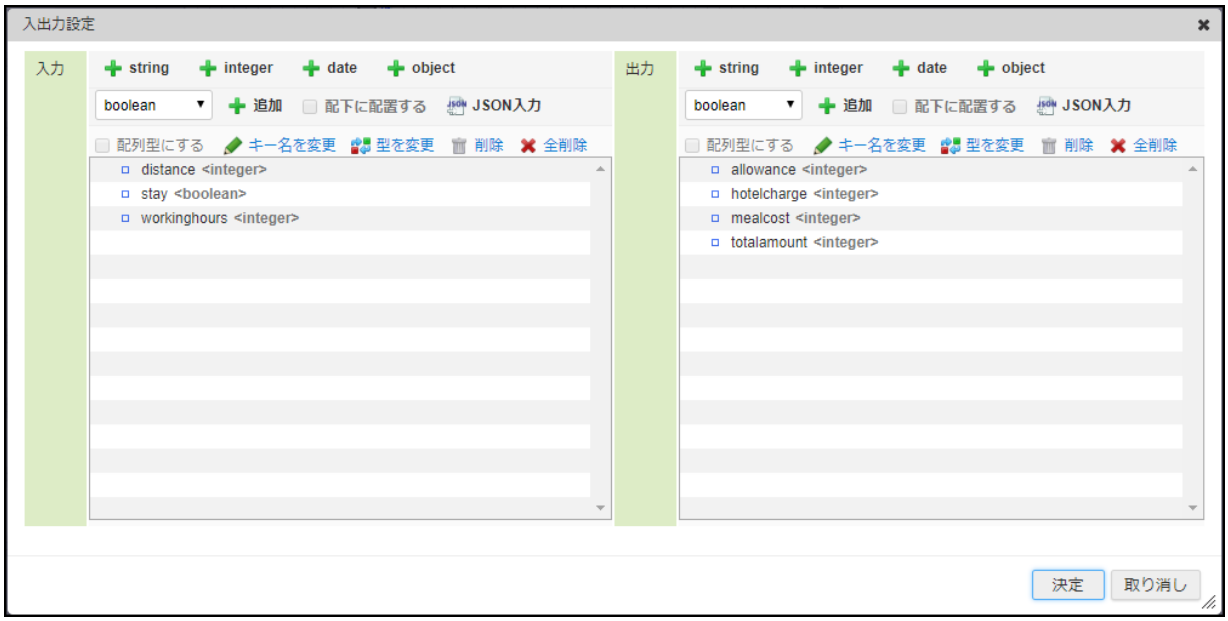
図：「ロジックフロー定義編集」

4. 設定されている「入出力値」を確認します。
   以下の値が登録されているのを確認します。
   - 入力
     「IM-FormaDesigner」の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>

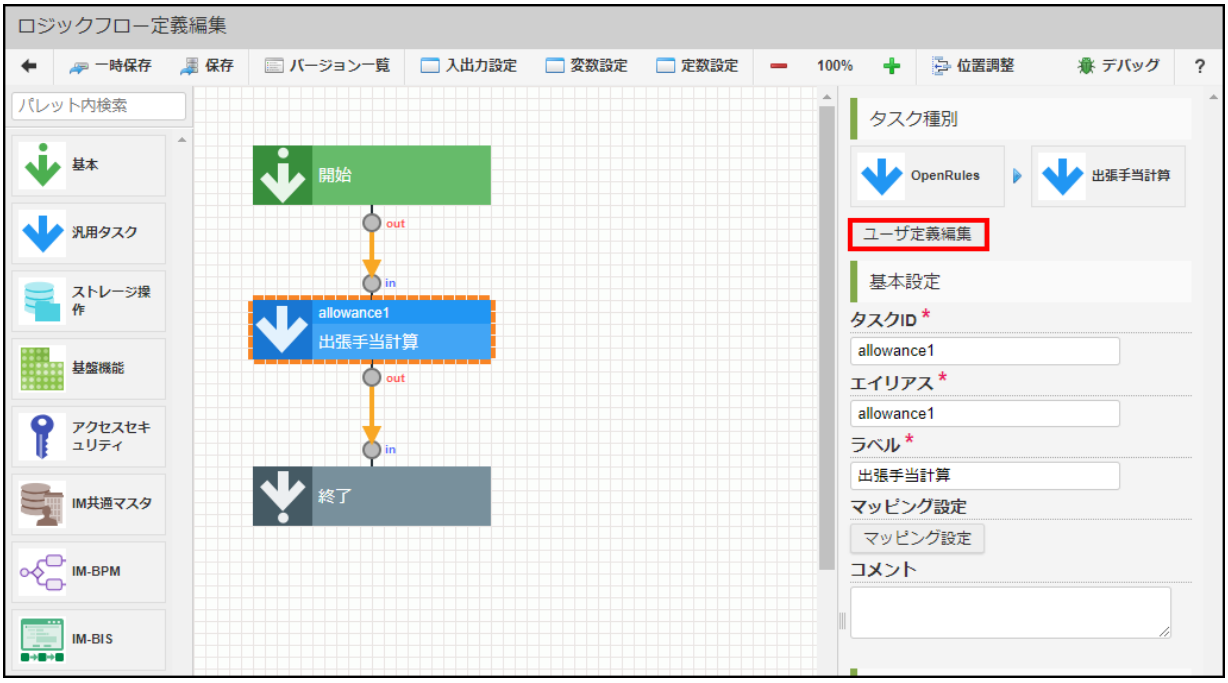
   - 出力
     「IM-FormaDesigner」のFormaアプリケーション「allowance_confirm」で表示するため、「IM-BPM」に返却する値

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。
以下のとおりに項目が設定されていることを確認してください。

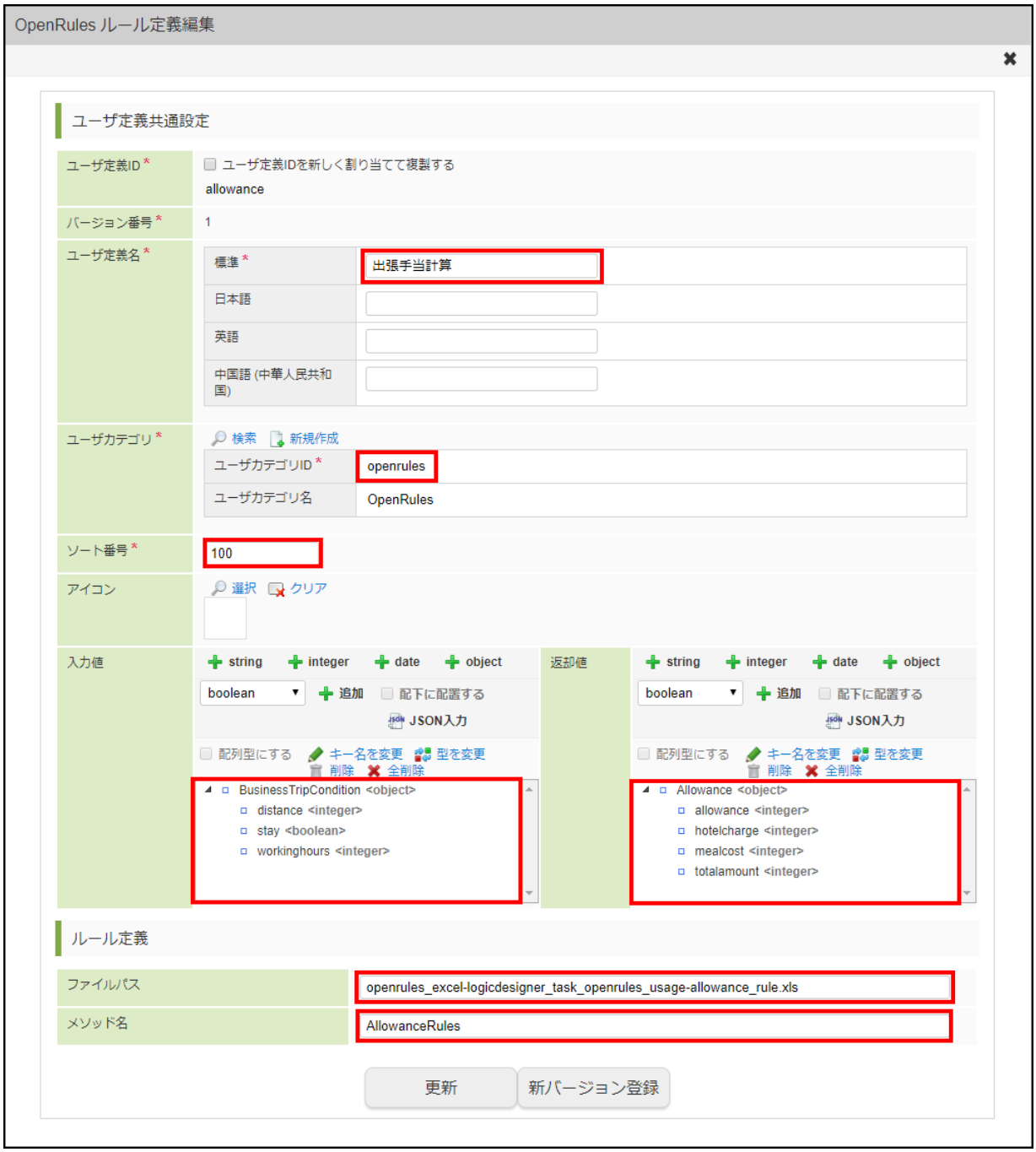
- ユーザ定義共通設定
  - ユーザ定義名：出張手当計算
  - ユーザカテゴリ：任意
  - ソート番号：任意
  - 入力値

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

  - 返却値

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

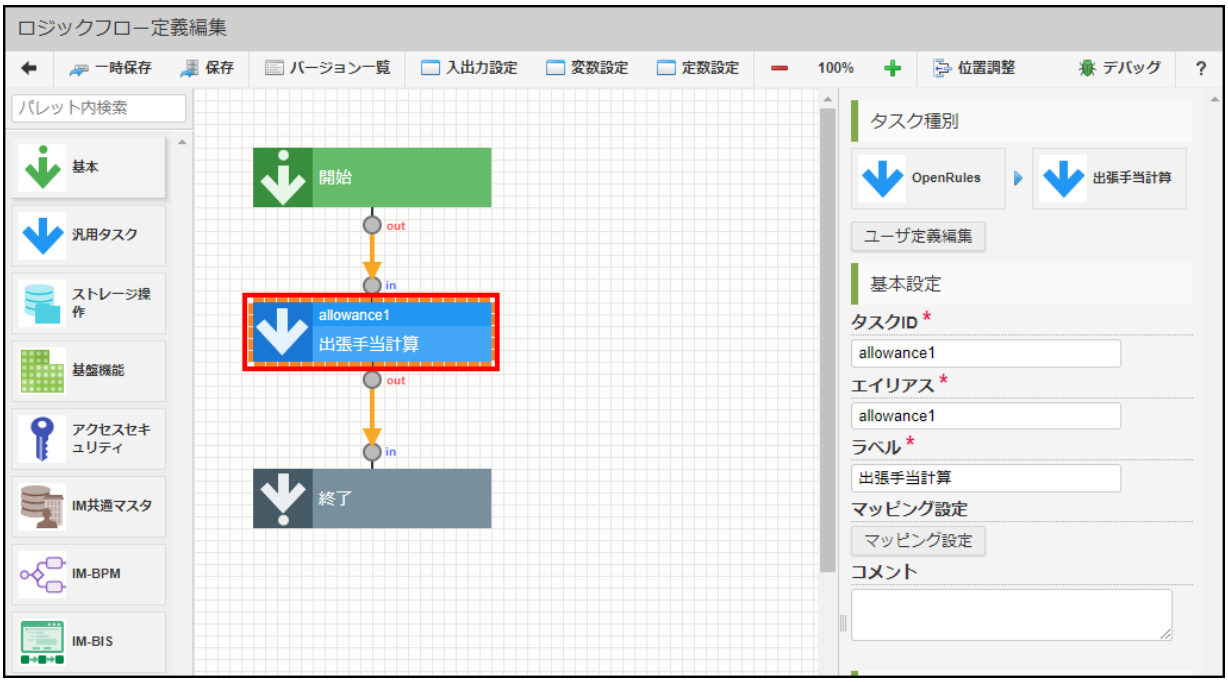
- ルール定義
  - ファイルパス： openrules_excel-logicdesigner_task_openrules_usage-allowance_rule.xls
  - メソッド名 : AllowanceRules



図：「OpenRulesルール定義編集」

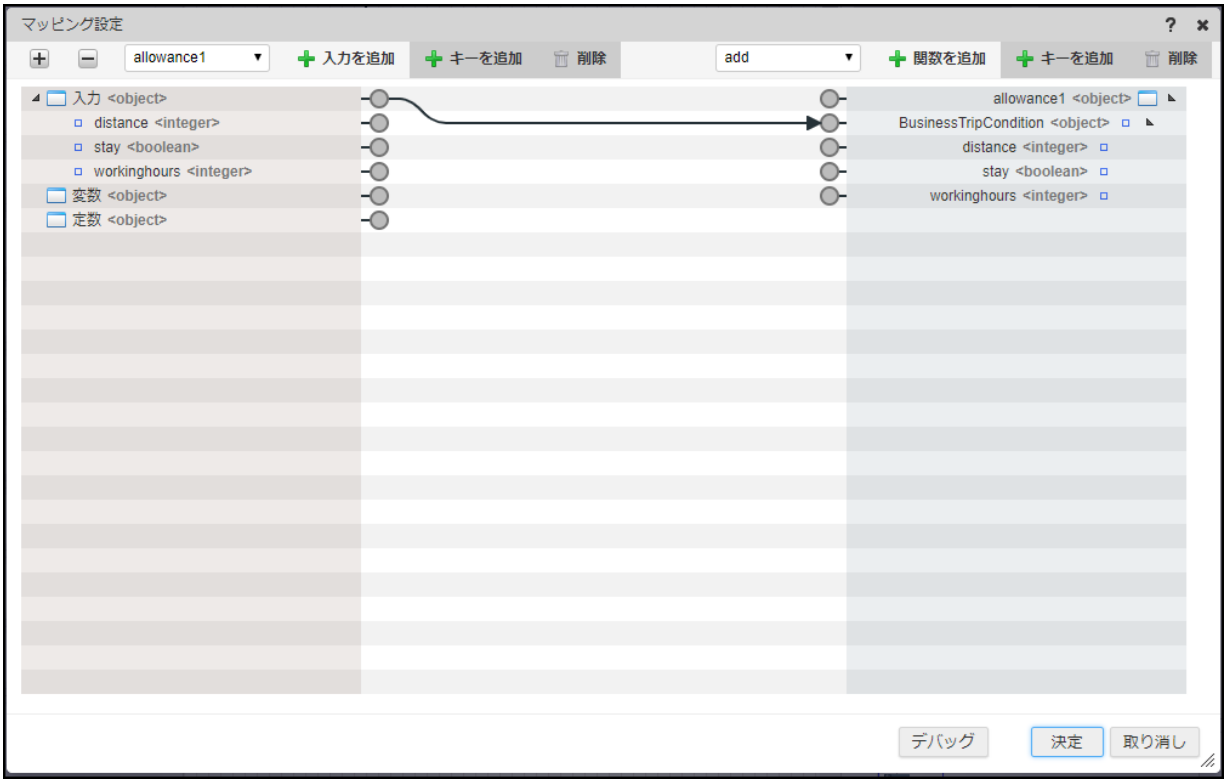
> **コラム**
>
> 「ルール定義」の「メソッド名」は、「処理を実行するEXCELファイル」の「Main」シート - 「Decision」テーブルに紐づいています。

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



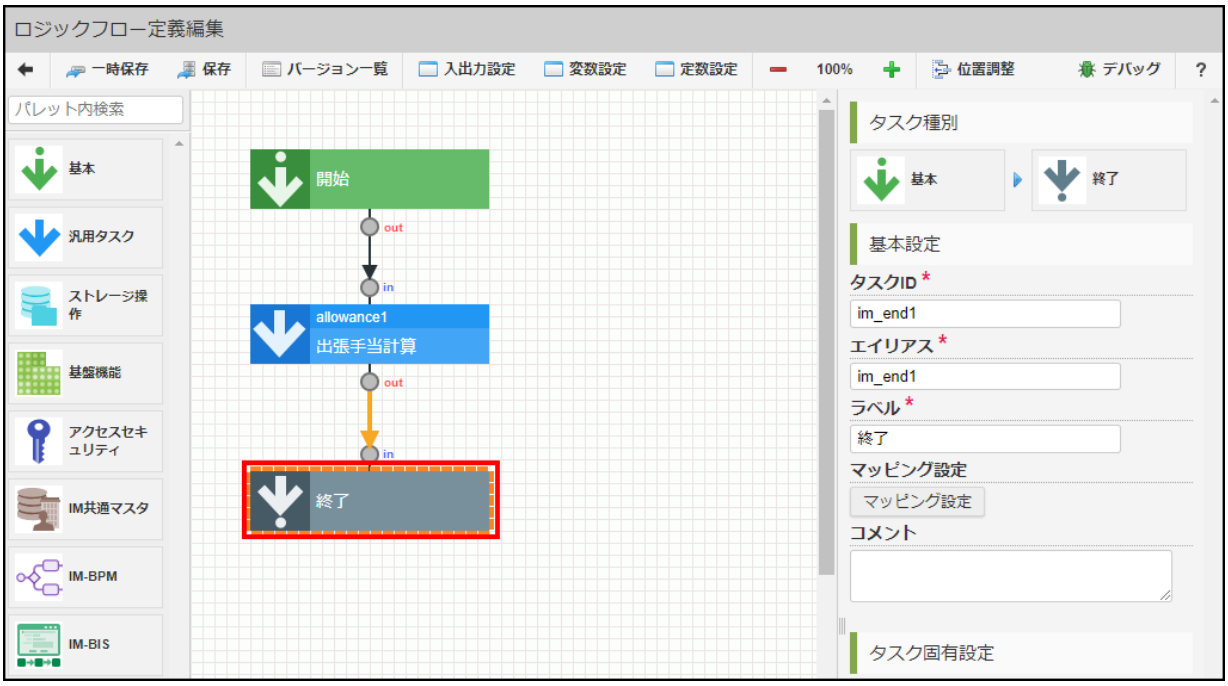
図：「ロジックフロー定義編集」

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



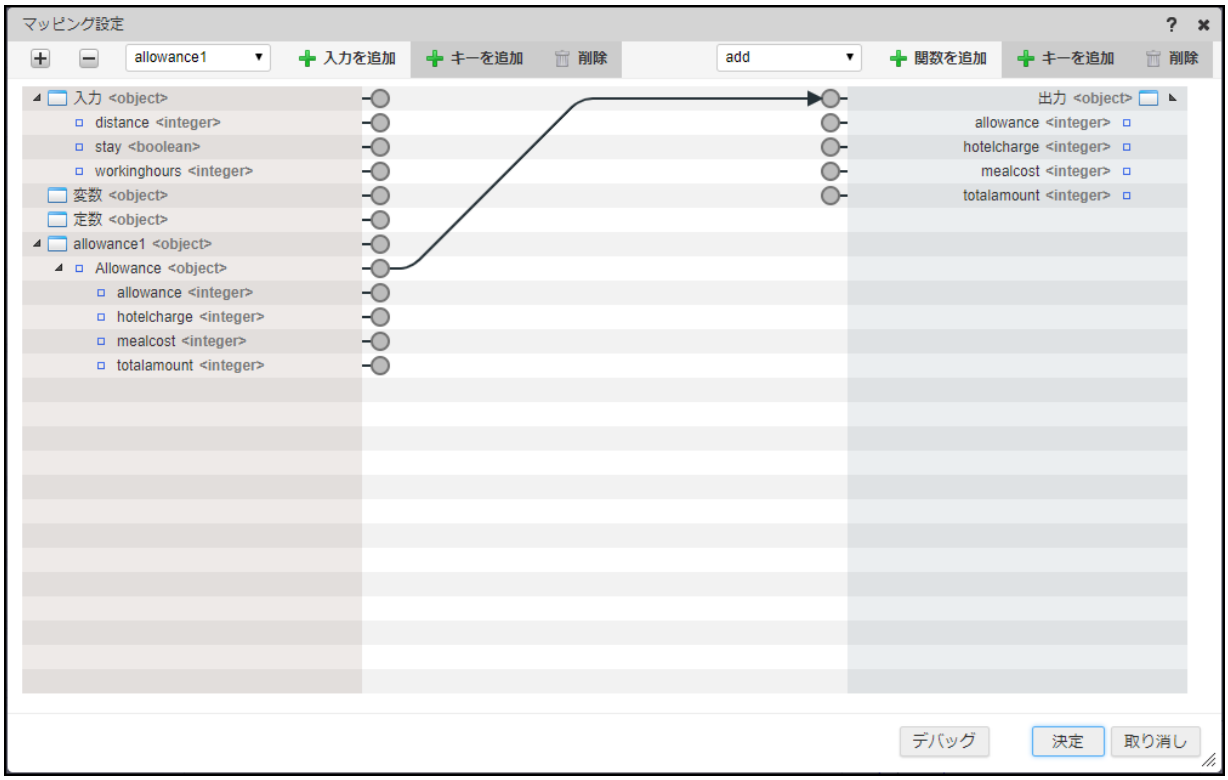
図：「マッピング設定」

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



図：「ロジックフロー定義編集」

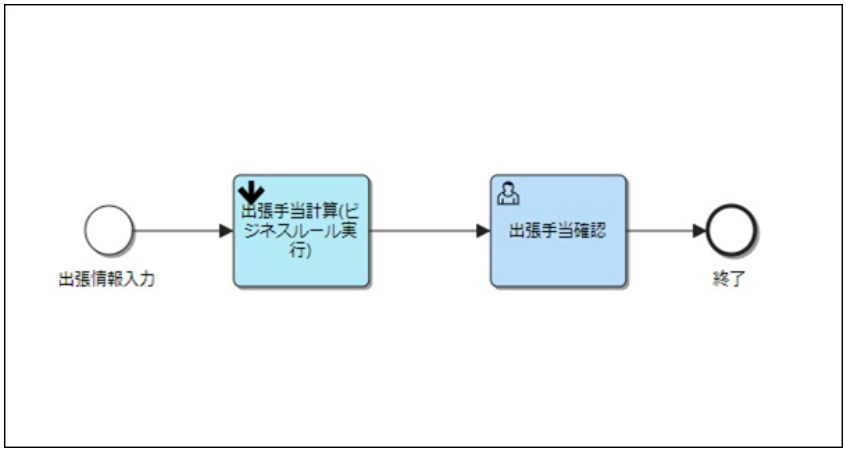
10. 左右で以下の値が登録・マッピングされていることを確認します。
    - 左側：Allowance<object>
      「OpenRules」が算出した値を「DecisionObject」テーブルで設定したオブジェクトに紐づけています。
    - 右側：出力<object>
      「IM-BPM」に返却する値を、「出力<object>」に紐づけています。



図：「マッピング設定」

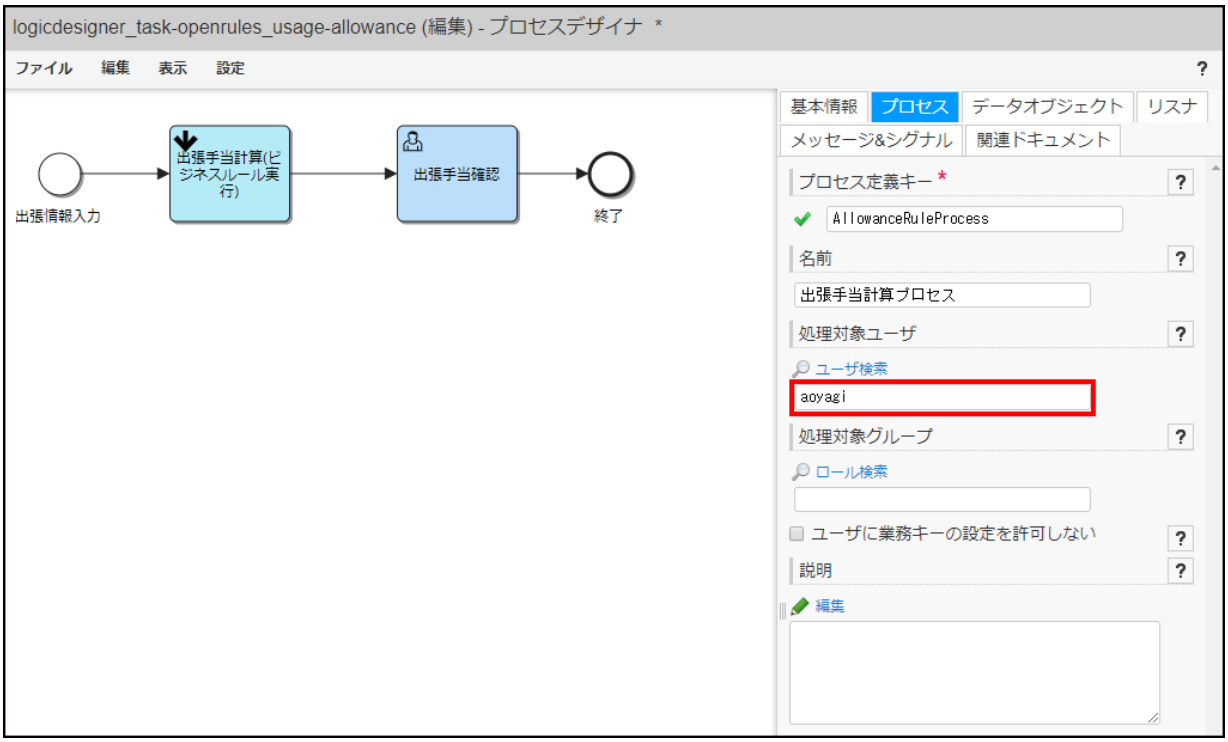
プロセス定義を作成する

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



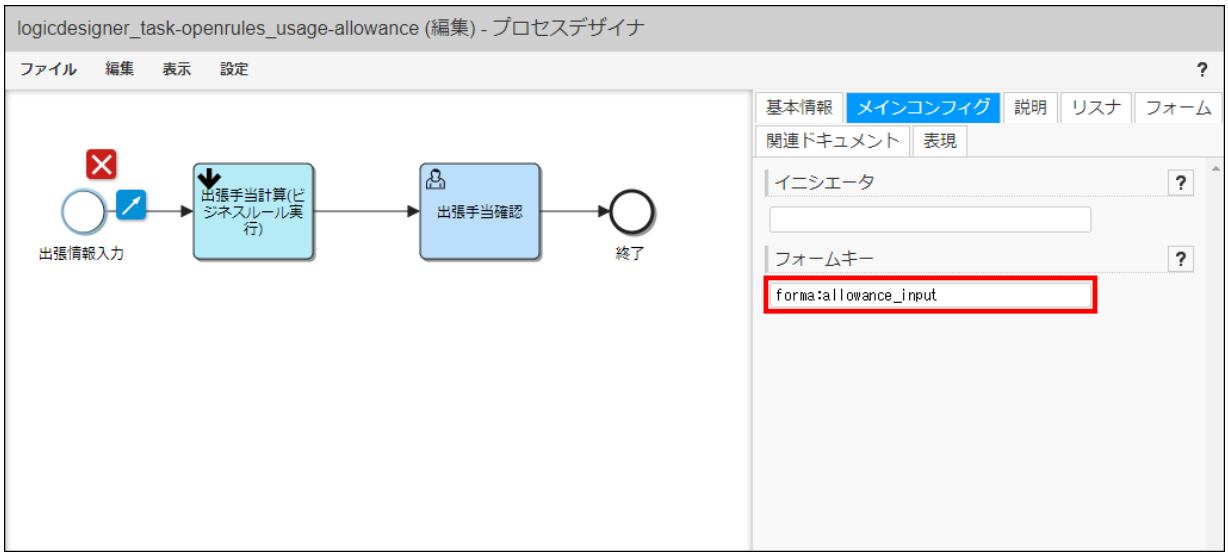
図：完成イメージ

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」

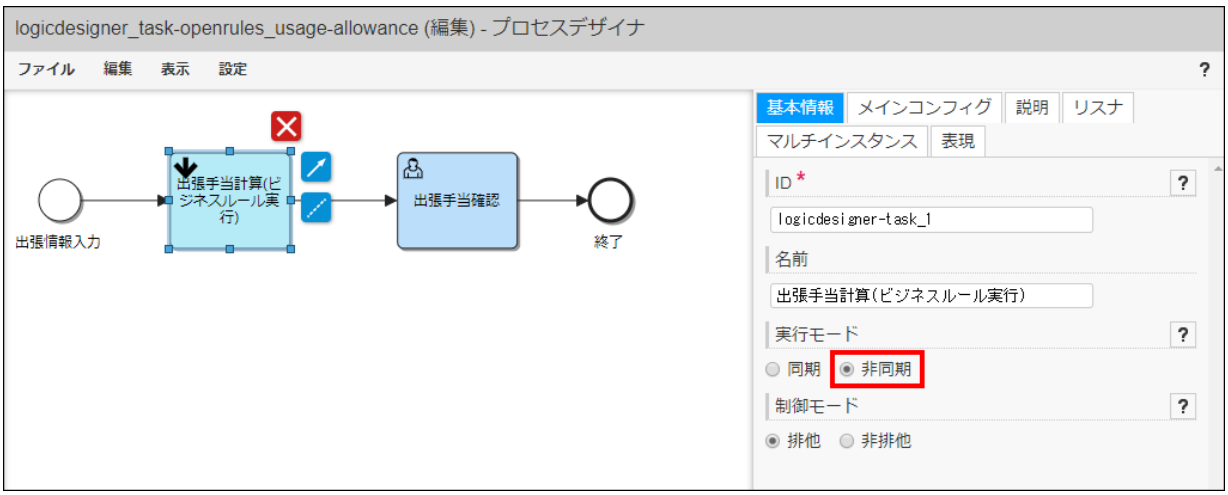
4. 「開始イベント」に、インポートした「IM-FormaDesigner」のアプリケーションを設定します。
   「開始イベント」を選択し、「メインコンフィグ」タブの「フォームキー」に以下の値を設定します。
   - フォームキー : forma:allowance_input



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。

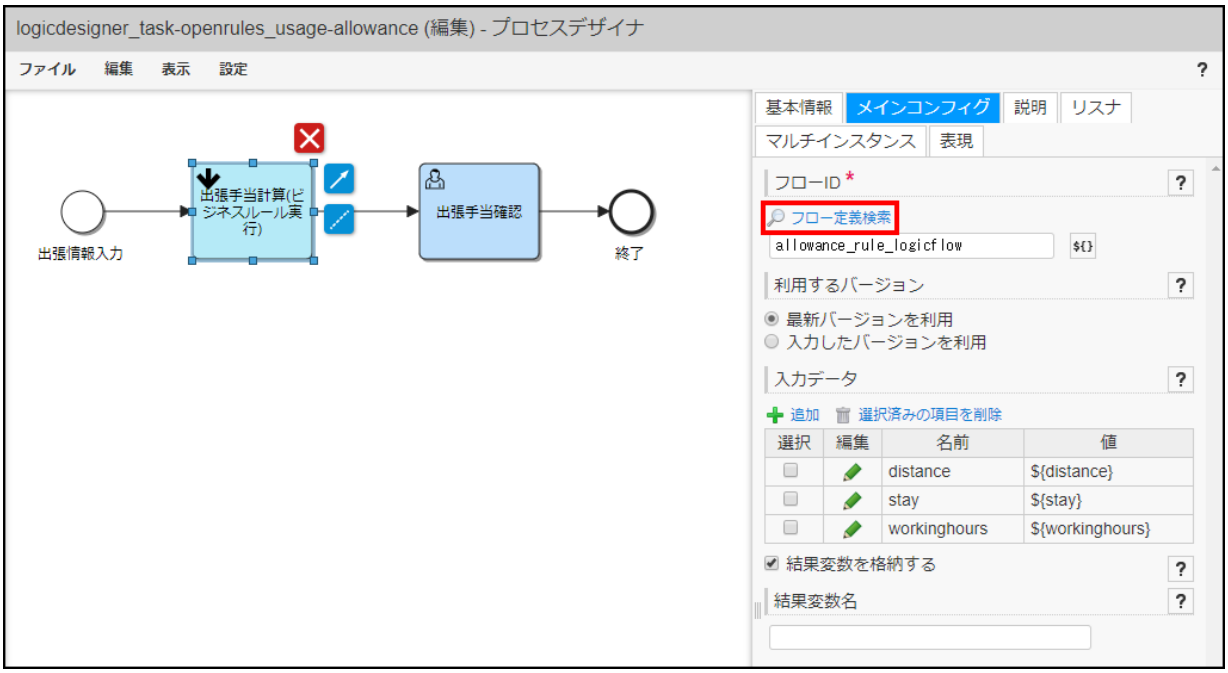
6. 「実行モード」の設定を行います。
   「基本情報」タブの「 実行モード」で、「非同期」を選択してください。



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

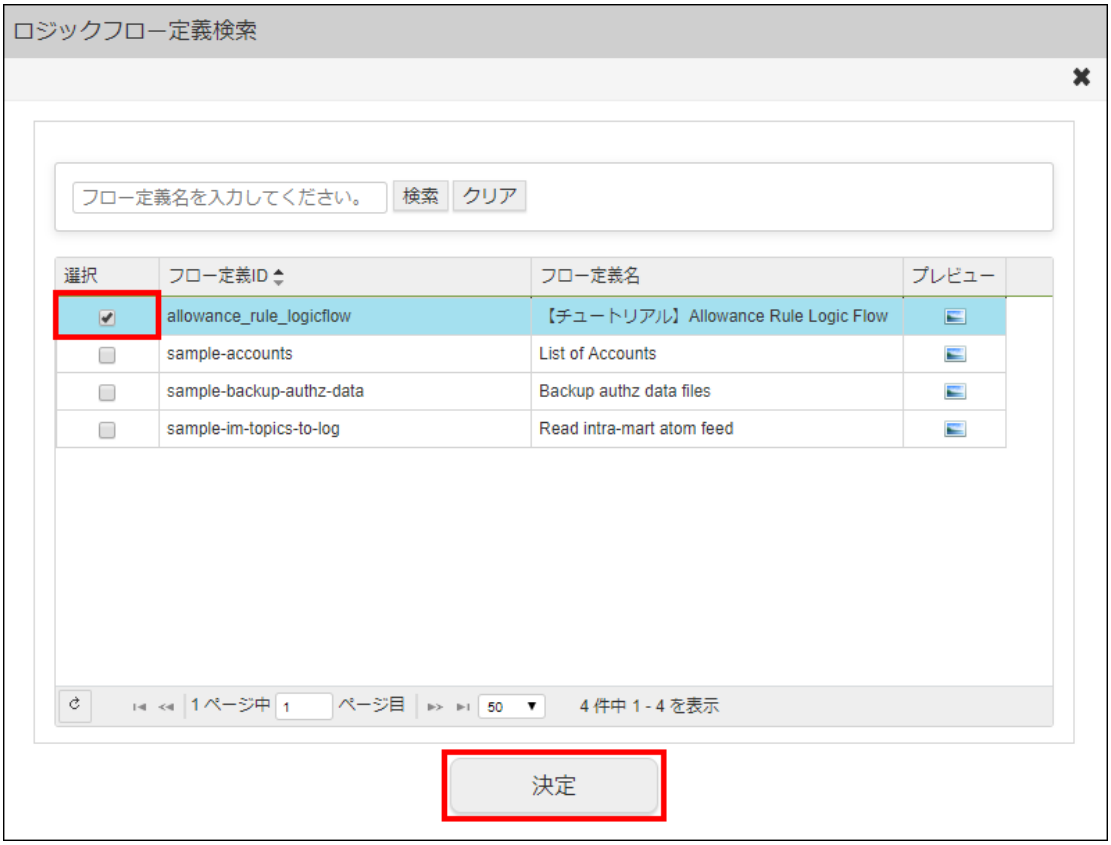
> **コラム**
>
> 「OpenRules」の実行は、時間がかかる可能性があります。
> 「IM-LogicDesignerタスク」を同期で実行すると、「OpenRules」の処理が終了するまで画面は応答しません。
> 「実行モード」を「非同期」にすることで、画面は即応答され「OpenRules」は非同期で実行されます。

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。



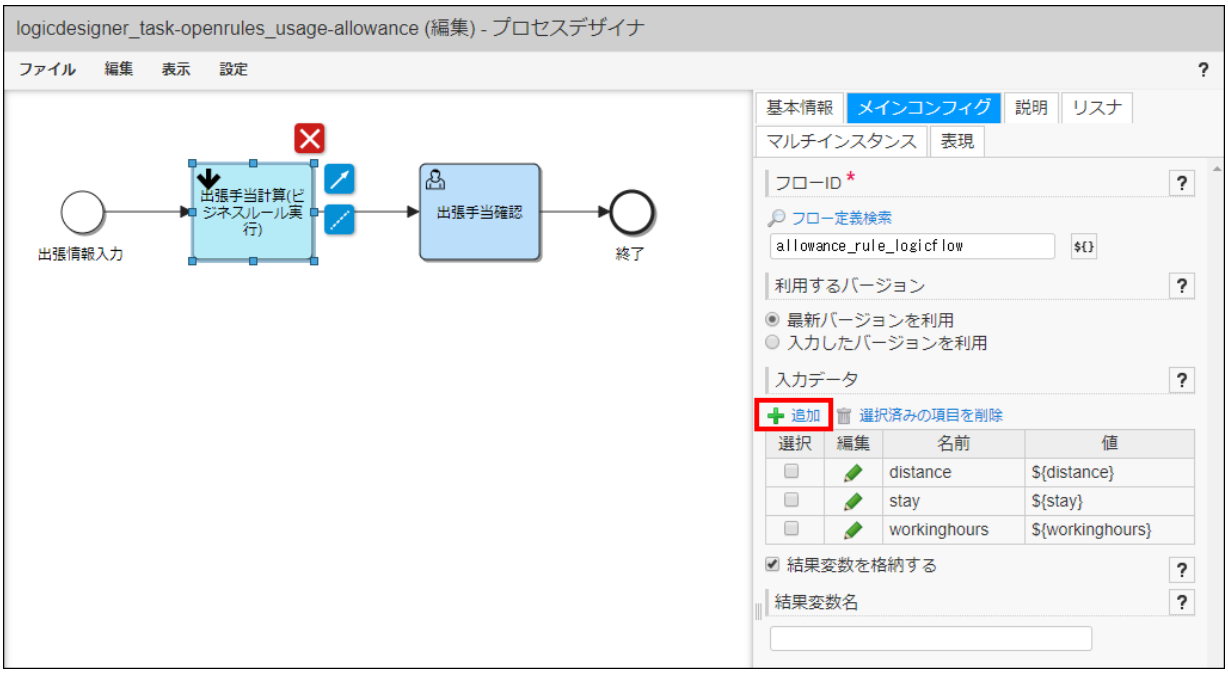
図：「IM-LogicDesignerタスク」-「プロパティ」 -「メインコンフィグ」

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。



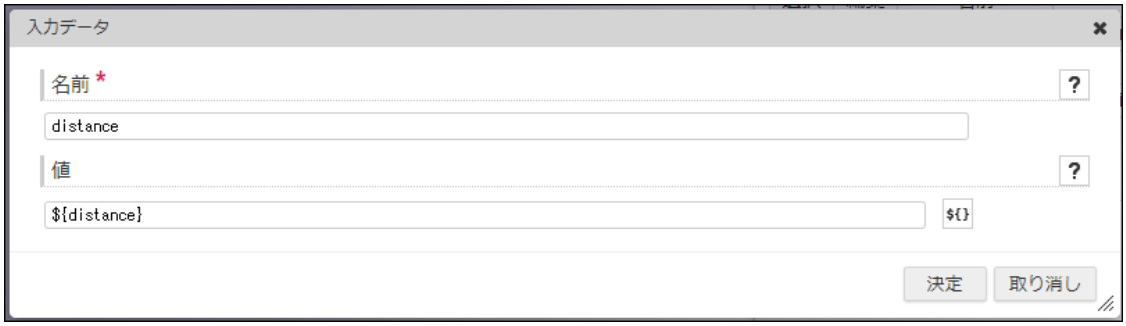
図：「ロジックフロー定義検索」

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。



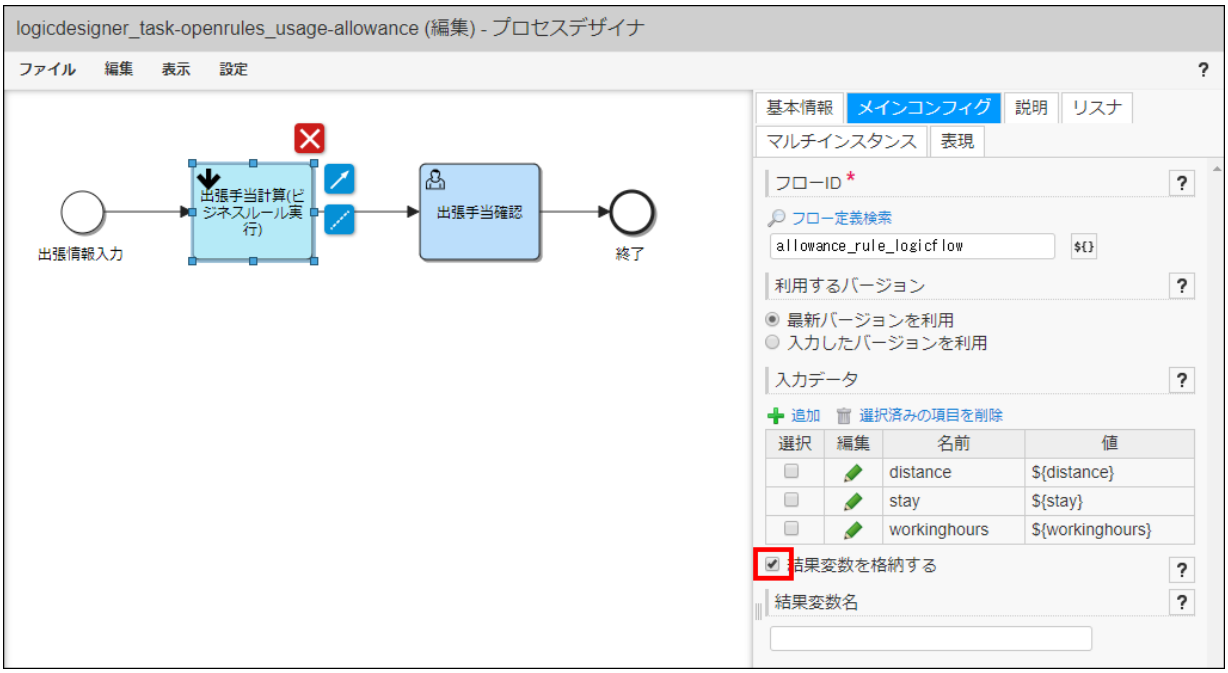
図：「IM-LogicDesignerタスク」 - 「プロパティ」 - 「メインコンフィグ」

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}



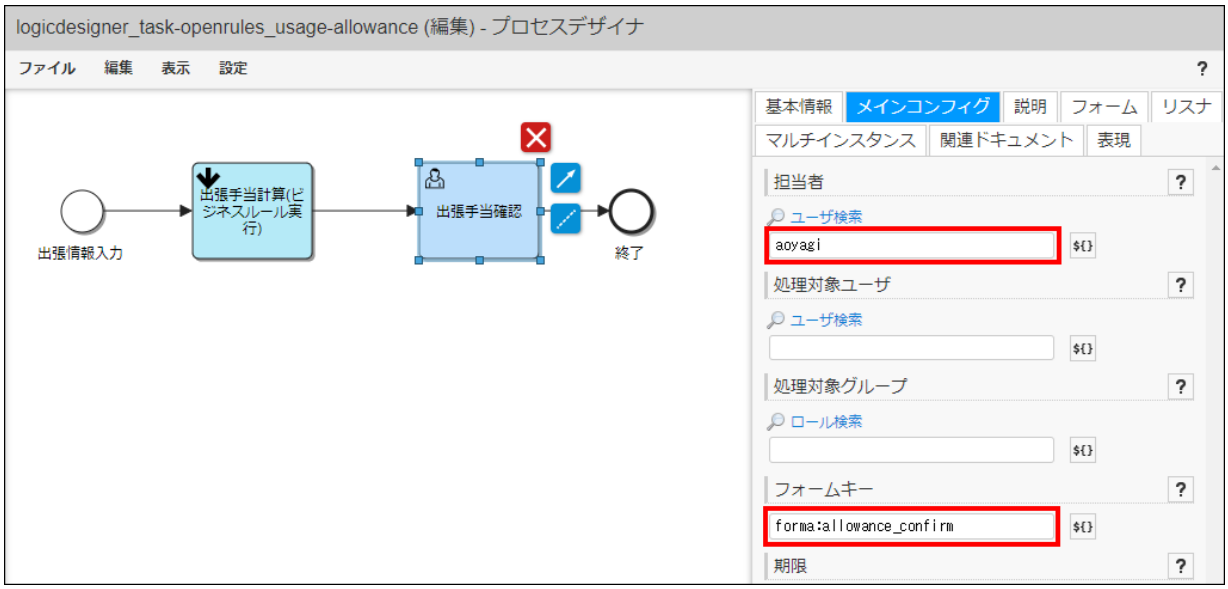
図：「入力データ」

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



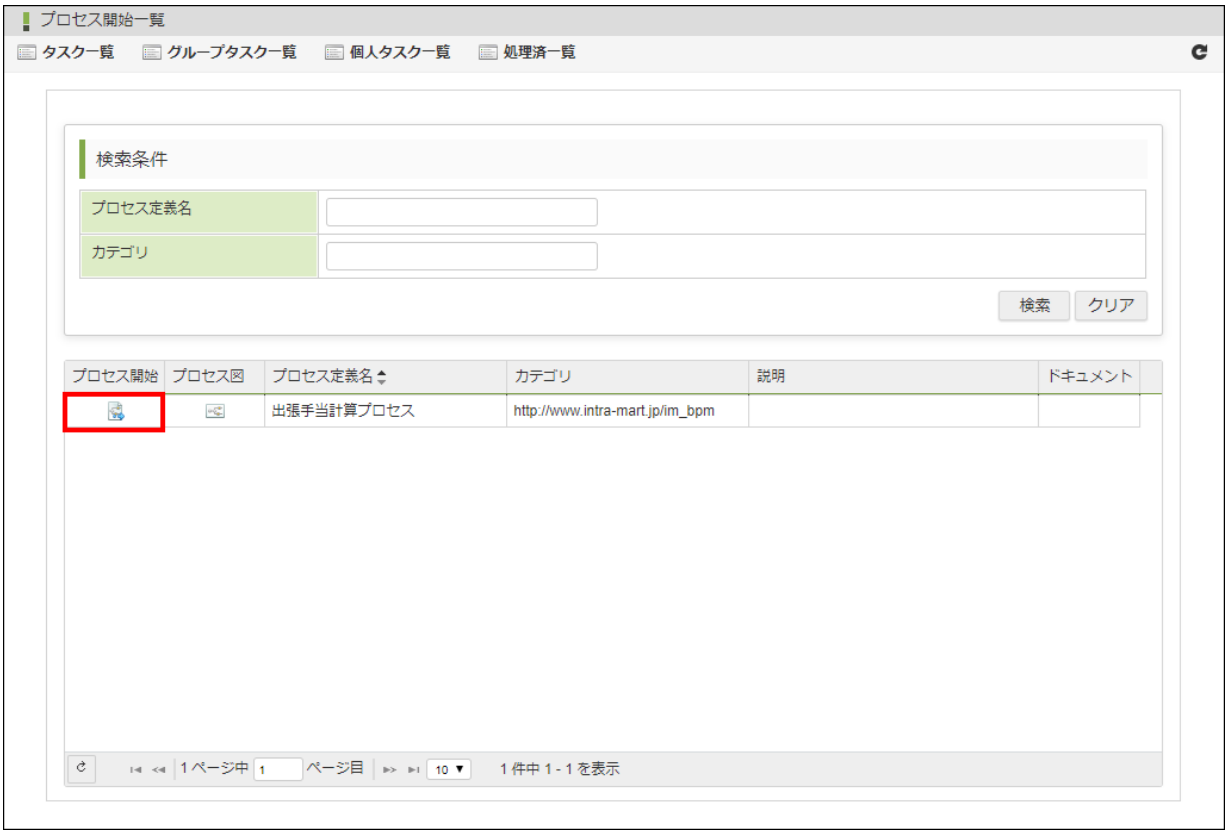
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

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。

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



図：「プロセス開始一覧」

3. 開始イベントに設定した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」

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

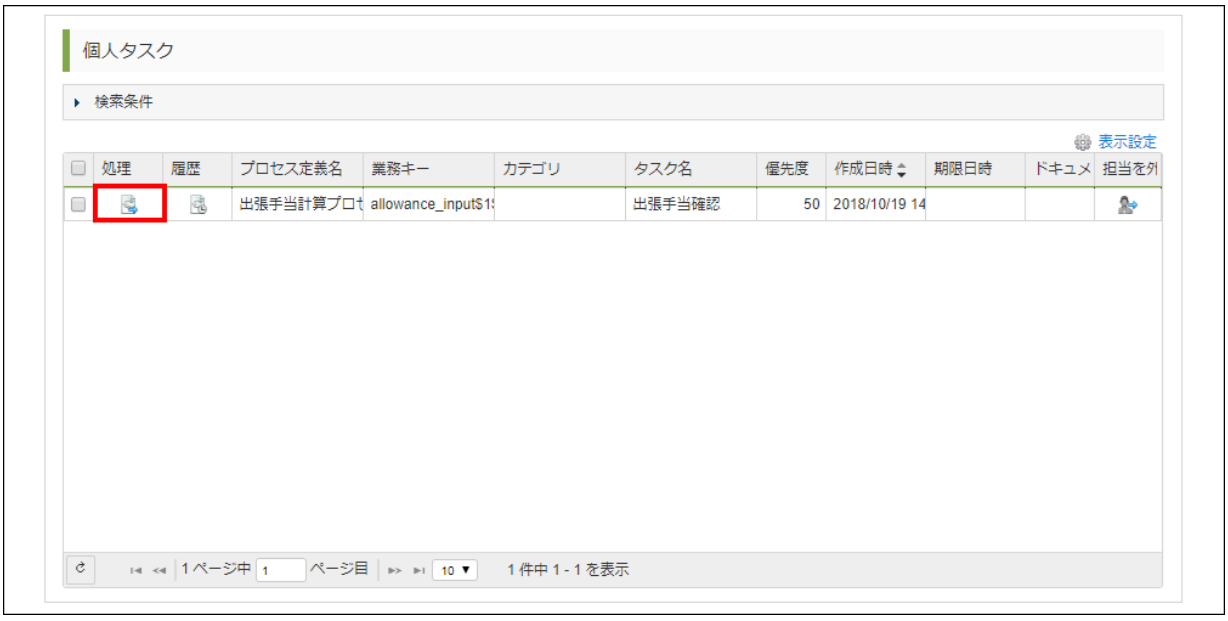


図 ： 「タスク一覧」 - 「個人タスク」

6. 結果が表示されます。

図 ：Formaアプリケーション「allowance_confirm」

IM-LogicDesignerタスクを利用してメッセージを送信する際に指定するエグゼキューションを取得する

このチュートリアルでは、「IM-LogicDesignerタスク」を利用して「プロセス定義キー」と「メッセージ名」をロジックフローに渡し、それに合致するエグゼキューションを取得したあとに確認のためログを出力する方法を解説します。

コラム

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

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

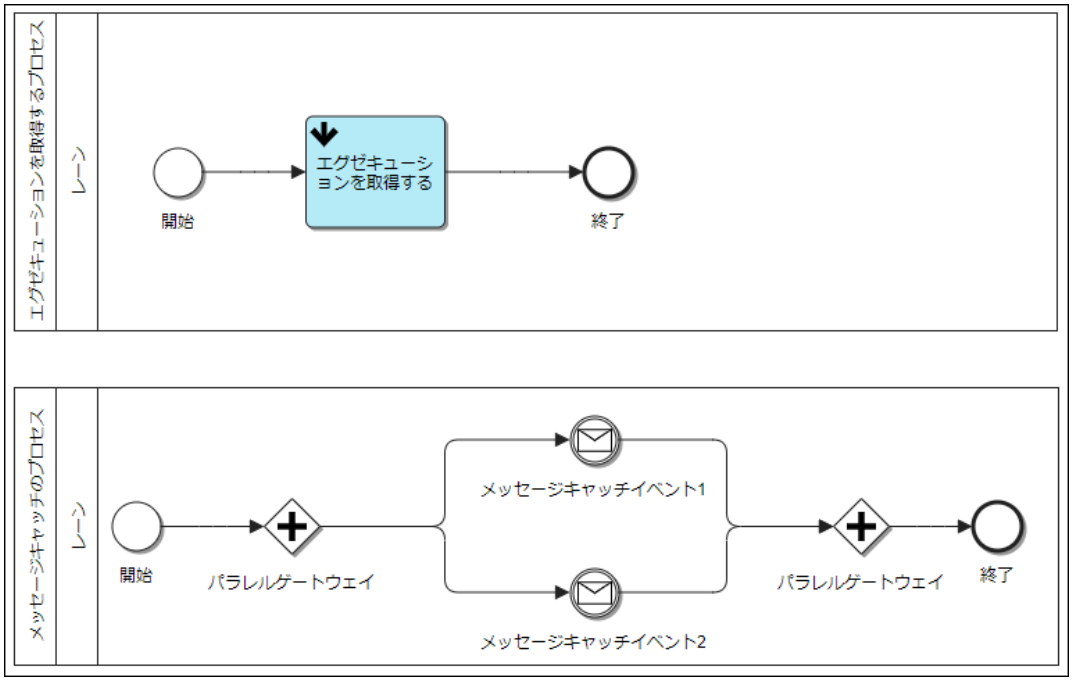
コラム

このチュートリアル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-logicdesigner_task_execution_usage.zip

インポート手順は ロジックフロー: 「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。

- エレメントを配置する
- プロセスのプロパティを設定する
- プール（エグゼキューションを取得するプロセス）のプロパティを設定する
- プール（メッセージキャッチのプロセス）のプロパティを設定する
- メッセージキャッチイベントのプロパティを設定する
- IM-LogicDesignerタスクのプロパティを設定する
- ロジックフロー定義を作成する
- 実行と実行結果を確認する

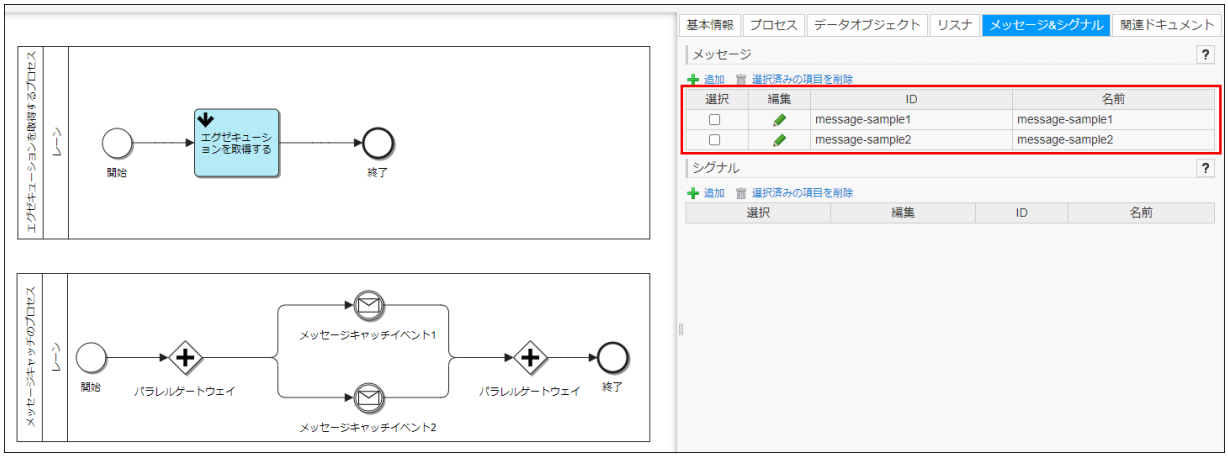
エレメントを配置する



図：完成イメージ

1. プールを2つ（エグゼキューションを取得するプロセスと、メッセージキャッチイベントを配置するプロセス）配置します。
2. エグゼキューションを取得するプロセスには開始イベント、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。
3. メッセージキャッチイベントを配置するプロセスには開始イベント、パラレルゲートウェイ、メッセージキャッチイベントを2つ、パラレルゲートウェイ 、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。

プロセスのプロパティを設定する

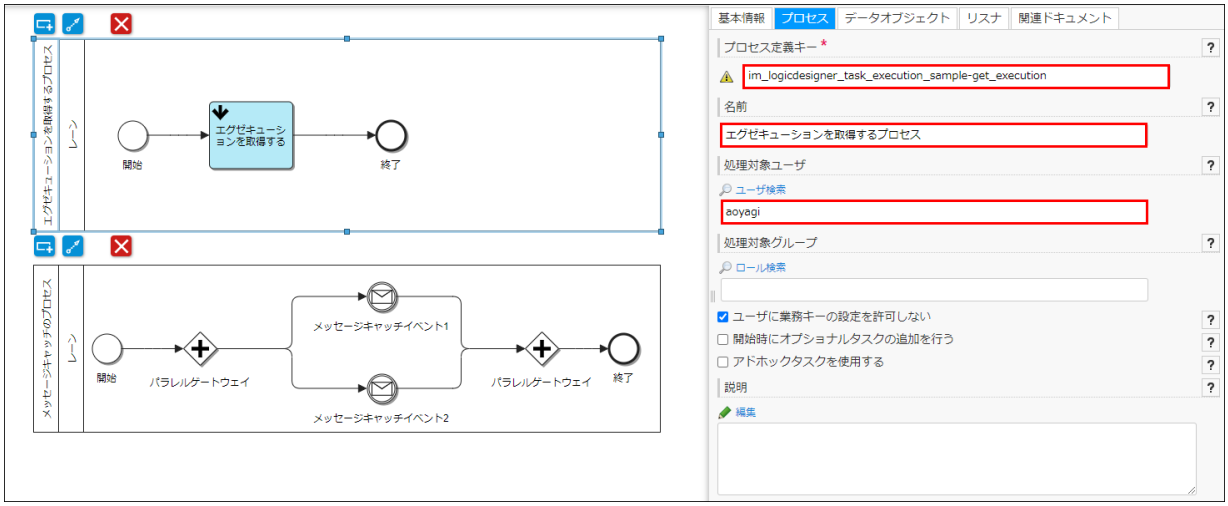


図：「プロパティ」 - 「メッセージ&シグナル」

1. プロセスのプロパティで「メッセージ&シグナル」タブを開きます。
2. 「メッセージ」に以下の値を追加します。

ID	名前
message-sample1	message-sample1
message-sample2	message-sample2

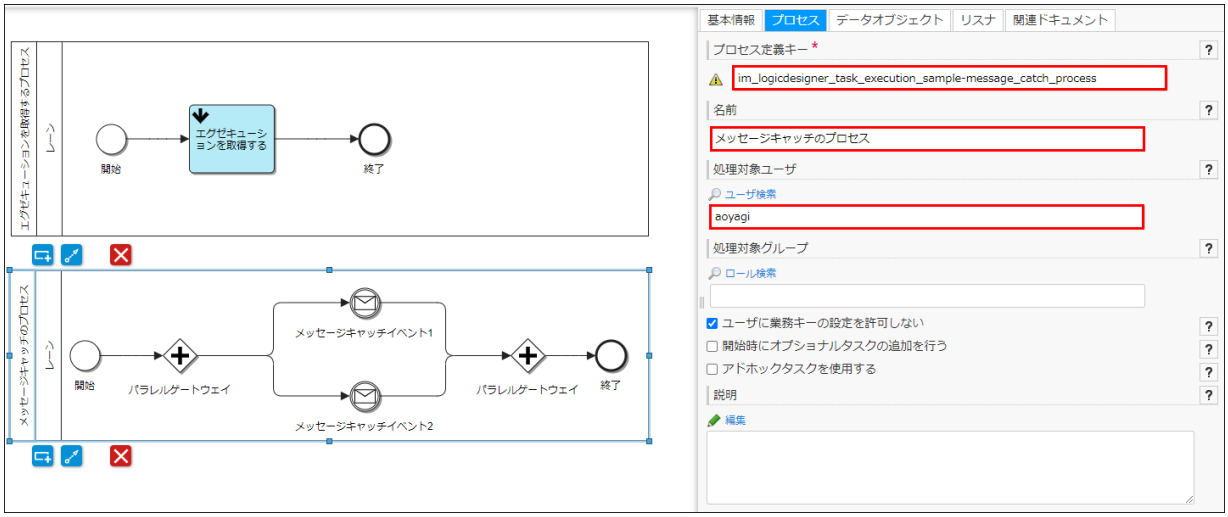
プール（エグゼキューションを取得するプロセス）のプロパティを設定する



図：「プール」 - 「プロパティ」 - 「プロセス」

1. プールのプロパティで「プロセス」タブを開きます。
2. エグゼキューションを取得するプロセスの「プロセス定義キー」に `im_logicdesigner_task_execution_sample-get_execution` を入力します。
3. エグゼキューションを取得するプロセスの「名前」に `エグゼキューションを取得するプロセス` を入力します。
4. 「処理対象ユーザ」に `aoyagi` を入力します。

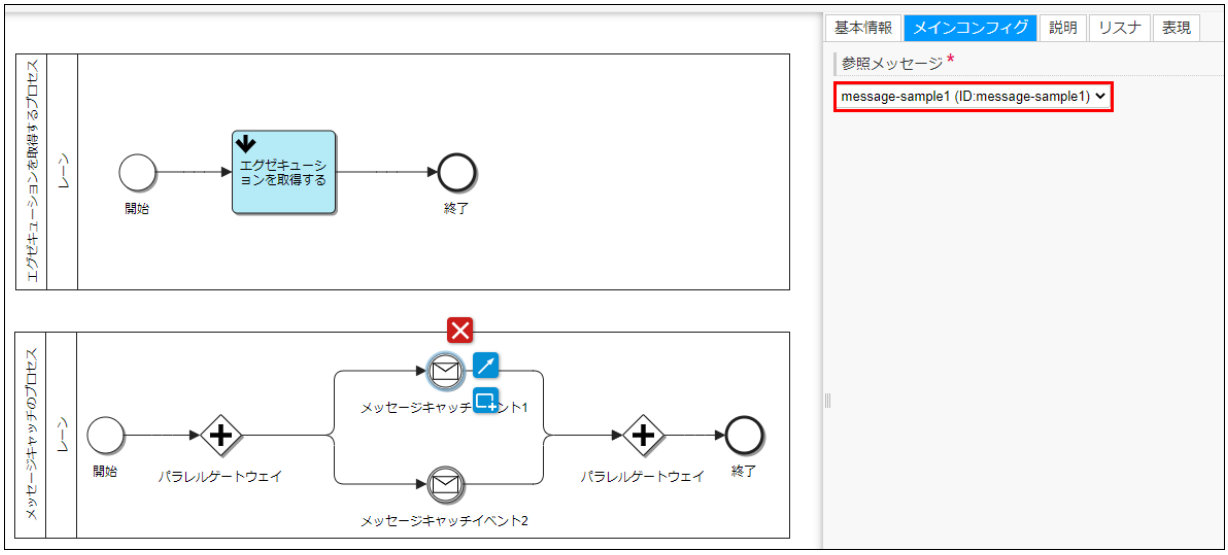
プール（メッセージキャッチのプロセス）のプロパティを設定する



図：「プール」 - 「プロパティ」 - 「プロセス」

1. プールのプロパティで「プロセス」タブを開きます。
2. 「プロセス定義キー」に im_logicdesigner_task_execution_sample-message_catch_process を入力します。
3. 「名前」に メッセージキャッチのプロセス を入力します。
4. 「処理対象ユーザ」に aoyagi を入力します。

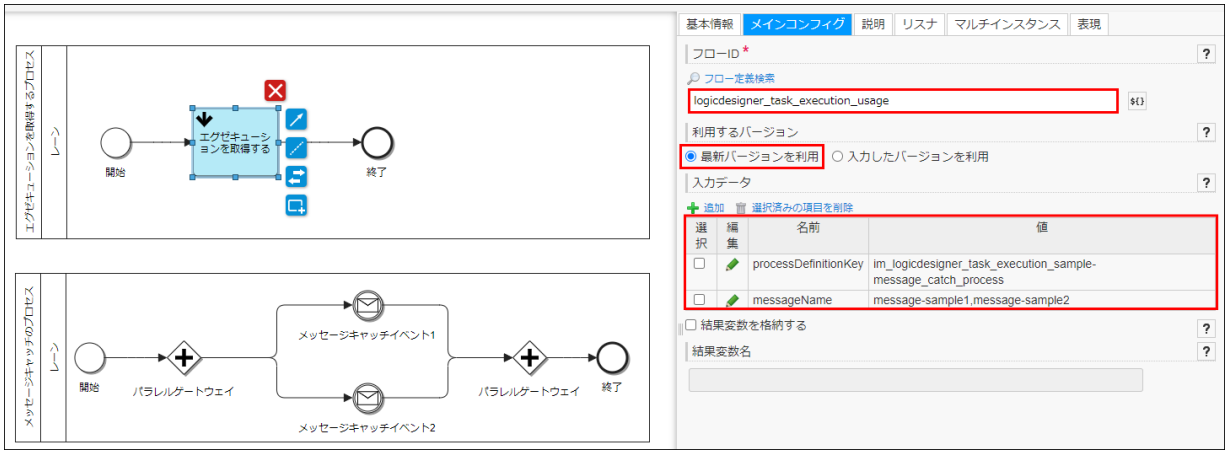
メッセージキャッチイベントのプロパティを設定する



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」

1. メッセージキャッチ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. 「メッセージキャッチイベント1」の「参照メッセージ」に message-sample1 を設定します。
3. 「メッセージキャッチイベント2」の「参照メッセージ」に message-sample2 を設定します。

IM-LogicDesignerタスクのプロパティを設定する



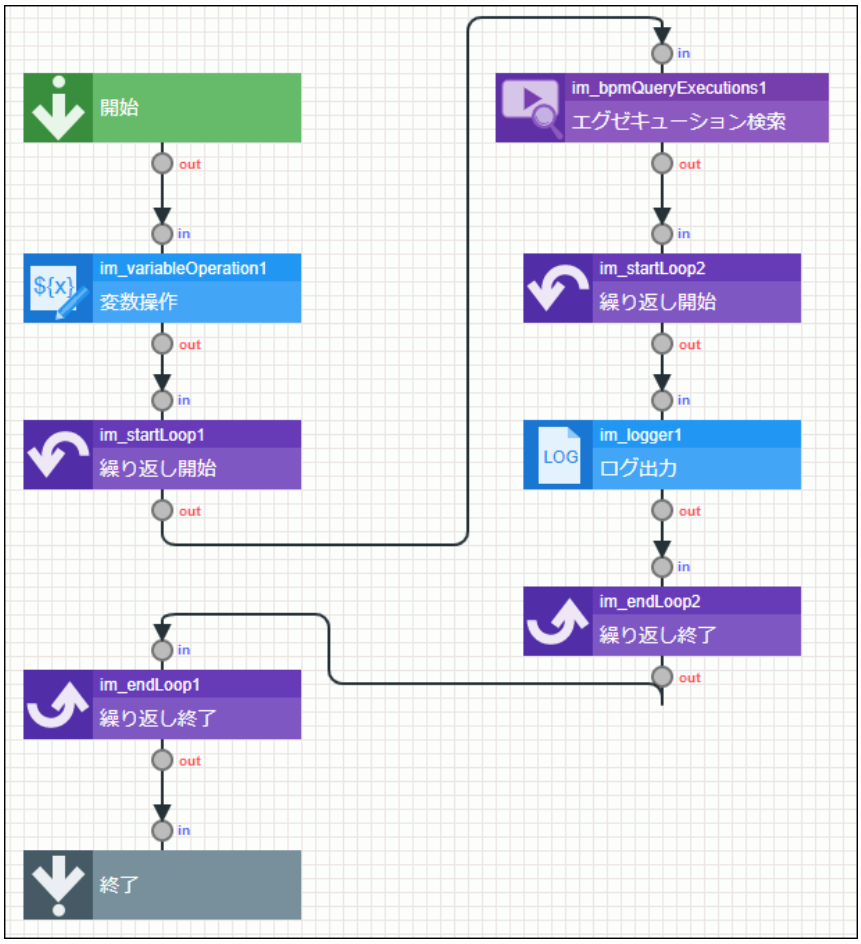
図：「IM-LogicDesignerタスク」 -「プロパティ」 -「メインコンフィグ」

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 `logicdesigner_task_execution_usage` を入力します。
3. 「利用するバージョン」で `最新バージョンを利用` を選択します。
4. 「入力データ」に以下の値を追加します。

名前	値
processDefinitionKey	im_logicdesigner_task_execution_sample-message_catch_process
messageName	message-sample1,message-sample2

ロジックフロー定義を作成する

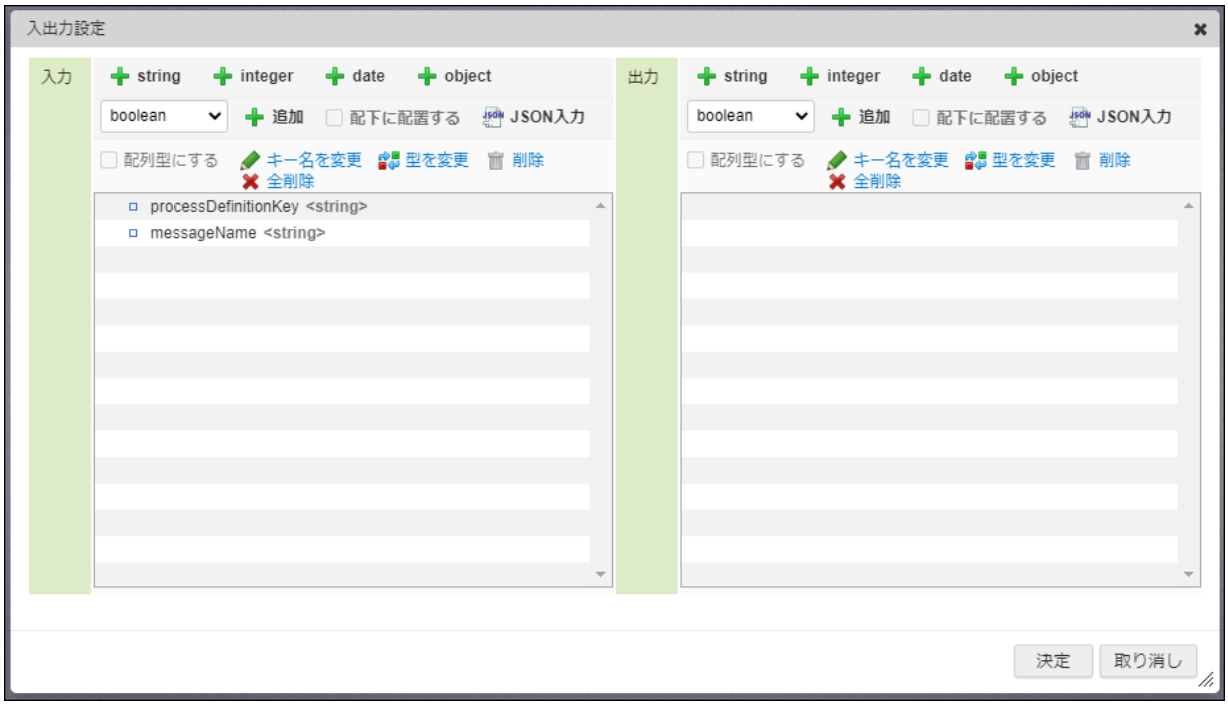
1. 「開始タスク」、「変数操作タスク」、「繰り返し開始タスク」、「エグゼキューション検索タスク」、「繰り返し開始タスク」、「ログ出力タスク」、「繰り返し終了タスク」、「繰り返し終了タスク」、「終了タスク」を配置し、順に接続します。



図：ロジックフロー

2. 入出力設定ダイアログを開き、入力に以下の値を追加します。

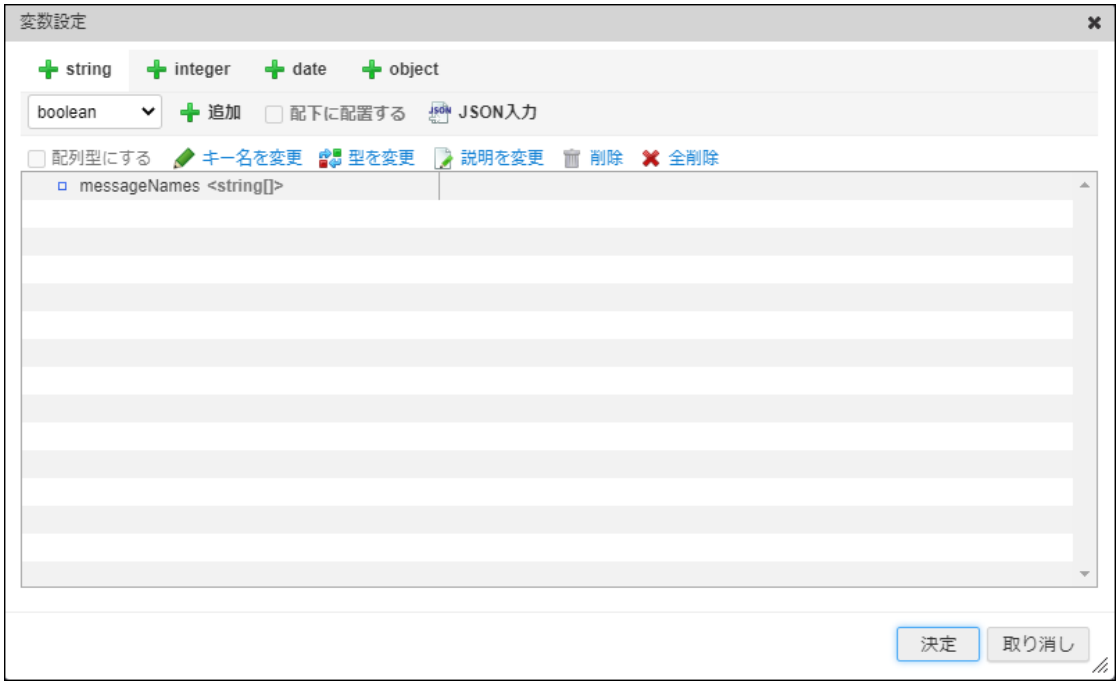
キー名	型
processDefinitionKey	<string>
messageName	<string>



図：「入出力設定」

3. 変数設定ダイアログを開き、以下の値を追加します。

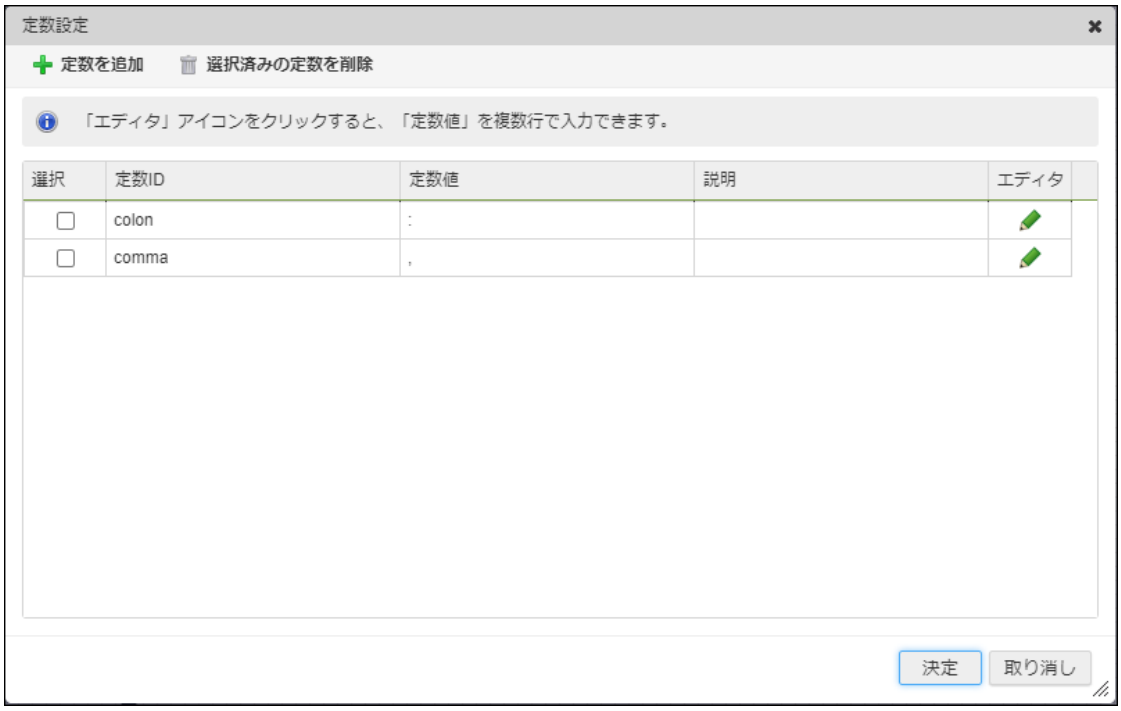
キー名	型
messageNames	<string[]>



図：「変数設定」

4. 定数設定ダイアログを開き、以下の値を設定します。

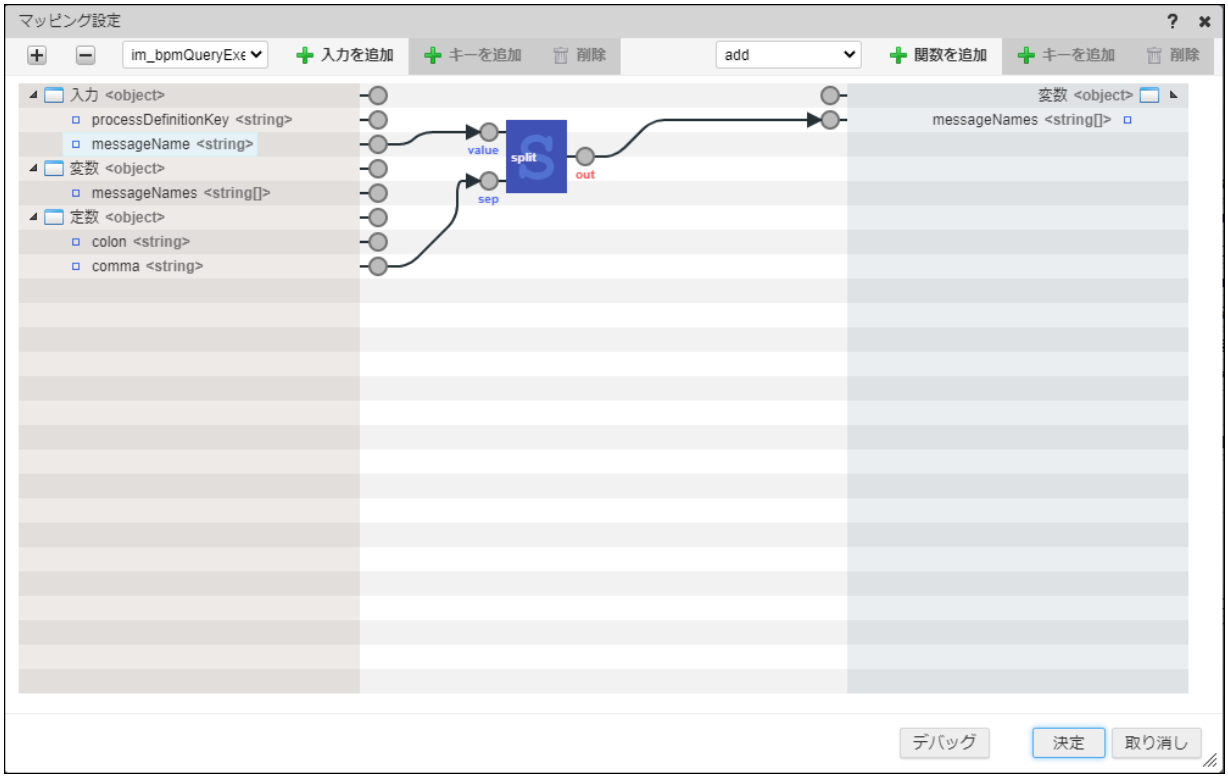
定数ID	定数値
colon	:
comma	,



図：「定数設定」

5. 変数操作タスクのマッピング設定を開き以下をマッピングします。

入力（始点）	出力（終点）
入力<object> - messageName<string>	split : value
定数<object> - comma<string>	split : sep
split : out	messageNames<string[]>



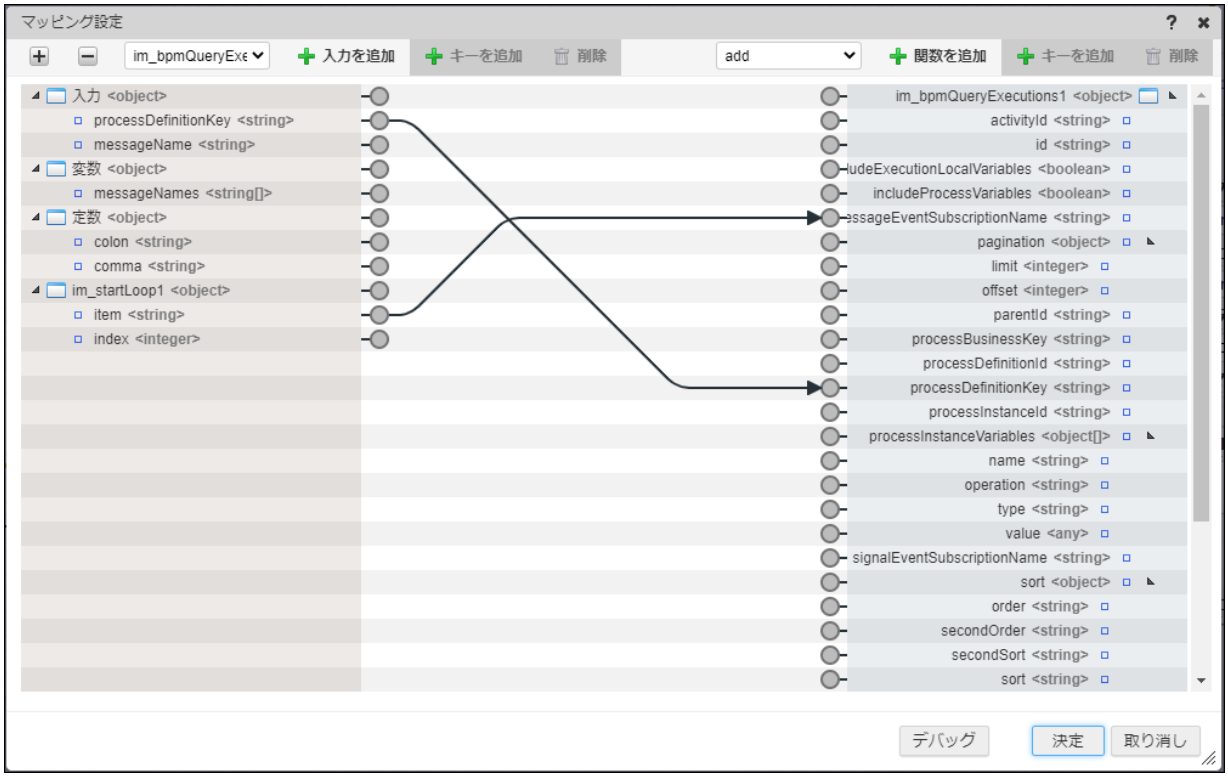
図：「変数操作タスク」 - 「マッピング設定」

6. 繰り返し開始タスク1の繰り返し対象プロパティに、 $variable/messageNames を設定します。

7. エグゼキューション検索タスクのマッピング設定を開き以下をマッピングします。

入力（始点）	出力（終点）
入力<object> - processDefinitionKey<string>	im_bpmQueryExecutions1<object> - processDefinitionKey<string>

入力（始点）	出力（終点）
im_startLoop1<object> - item<string>	im_bpmQueryExecutions1<object> - messageEventSubscriptionName<string>

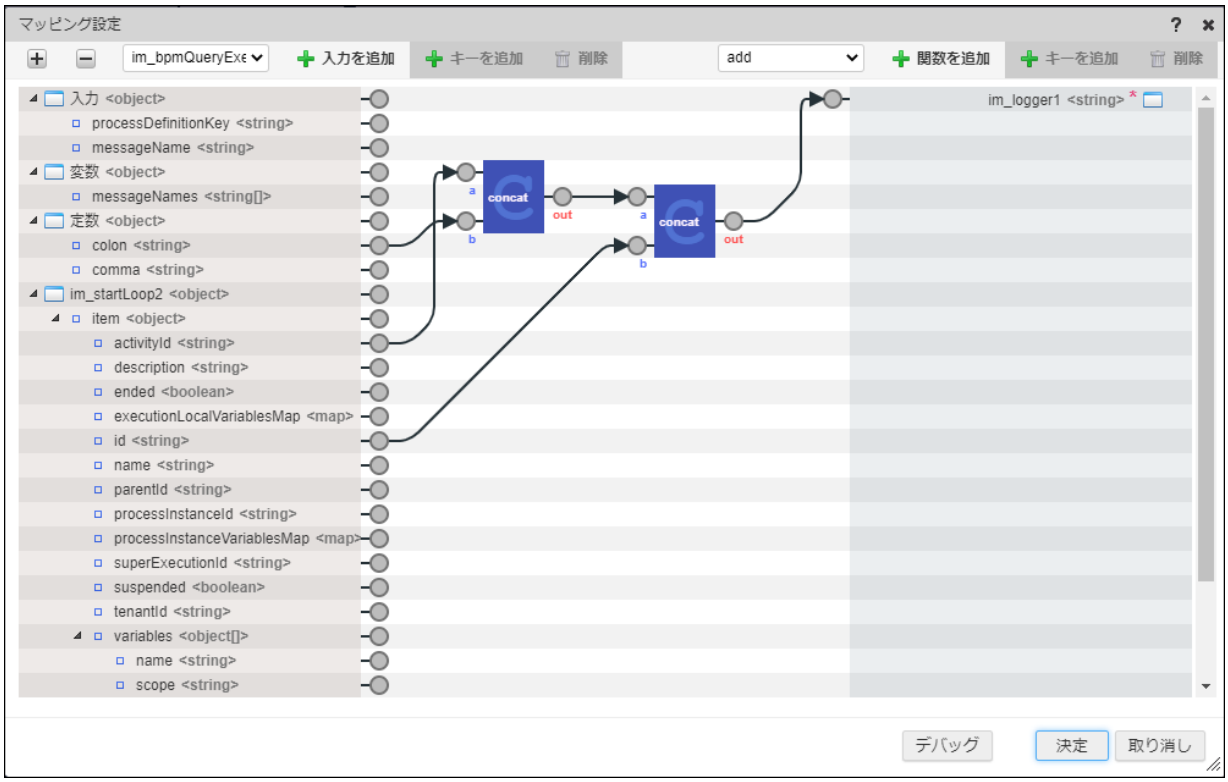


図：「エグゼキューション検索タスク」 - 「マッピング設定」

8. 繰り返し開始タスク2の繰り返し対象プロパティに、 im_bpmQueryExecutions1/queryExecutionsResults を設定します。

9. ログ出力タスクのマッピング設定を開き以下をマッピングします。

入力（始点）	出力（終点）
定数<object> - colon<string>	concat : b
im_startLoop2<object> - item<object> - activityId<string>	concat : a
concat : out	concat : a
im_startLoop2<object> - item<object> - id<string>	concat : b
concat : out	im_logger<string>



図：「ログ出力タスク」 -「マッピング設定」

10. 新規保存ボタンをクリックし、以下を設定しロジックフローを保存します。
    - フロー定義ID：logicdesigner_task_execution_usage
    - フロー定義名：【チュートリアル】logicdesigner_task_execution_usage
    - フローカテゴリ：
      - ID：im_logicdesigner-data
      - 名称：BPMチュートリアル

> コラム
> - ロジックフロー定義の作成方法については「IM-LogicDesigner ユーザ操作ガイド」を参照してください。

実行と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ます。また、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、プロセス定義名 メッセージキャッチのプロセス を開始します。
2. 「プロセス開始一覧」画面にて、プロセス定義名 エグゼキューションを取得するプロセス を開始します。
3. ログに以下のようにエグゼキューションIDが2つ出力されていることを確認します。

```
[INFO] j.c.i.f.l.e.g.OutputLogTask - [] message-catching-event_1:8frzpz6uwg485ca
[INFO] j.c.i.f.l.e.g.OutputLogTask - [] message-catching-event_2:8frzpz6v4g48aca
```

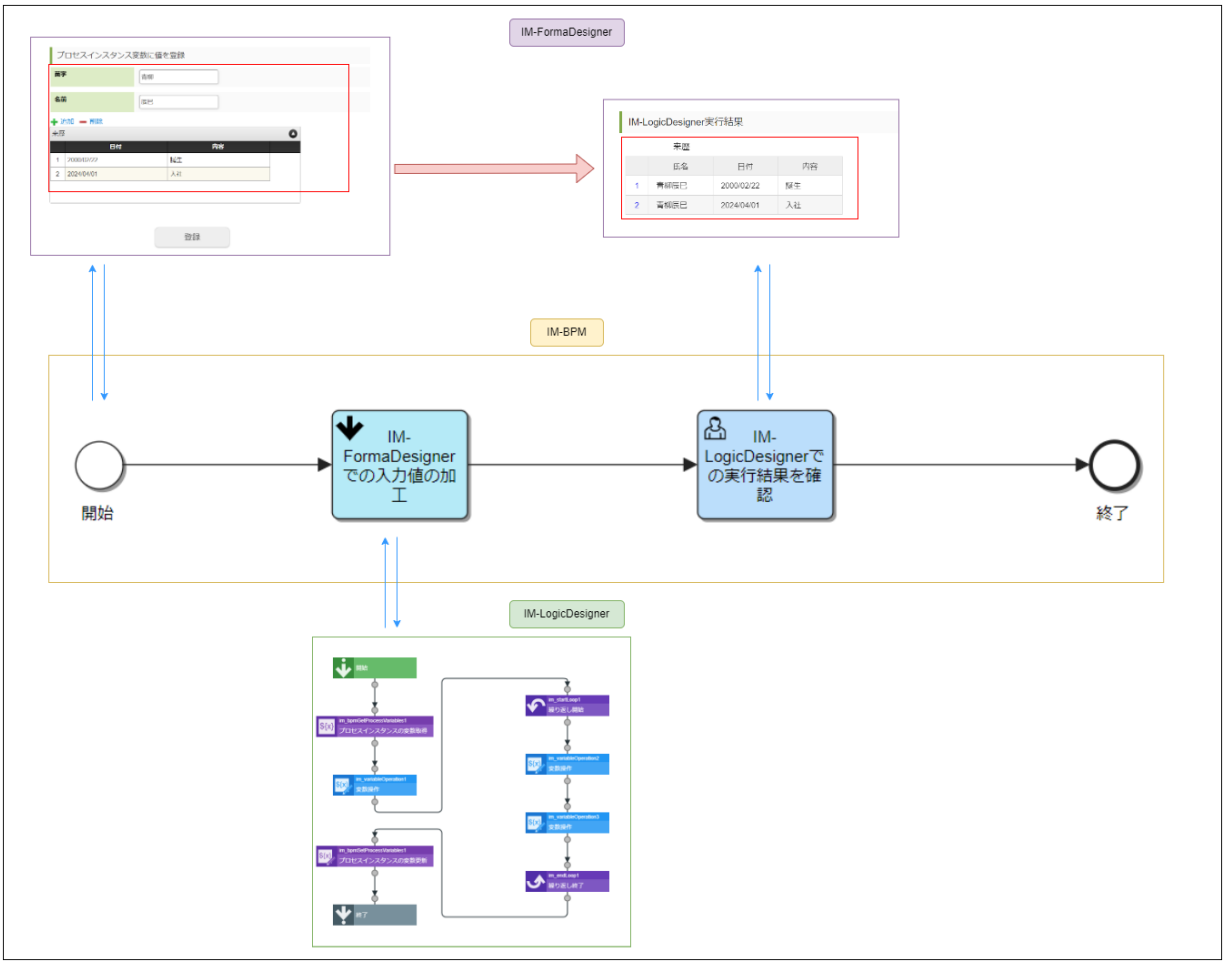
図：ログ出力イメージ

> コラム
> - プロセスのデプロイ方法については「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義/ケース定義のデプロイ」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。

IM-LogicDesignerタスクを用いて、IM-FormaDesignerで入力したテーブルアイテムの値を扱う

このチュートリアルでは以下の解説を行います。

- プロセスの開始時に入力した IM-FormaDesignerのテーブルアイテムの値を、「IM-LogicDesignerタスク」に渡す方法
- 「IM-LogicDesignerタスク」を用いて、 ユーザタスクのフォームとして利用する IM-FormaDesignerのテーブルアイテムに値を反映させる方法



図：概要

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

> **コラム**
>
> このチュートリアル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-logicdesigner_task_forma_tb_item.zip**
> **im_logicdesigner-data-logicdesigner_task_forma_tb_item_en.zip**
>
> インポート手順は ロジックフロー: 「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。

- 事前準備
- 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

事前準備

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

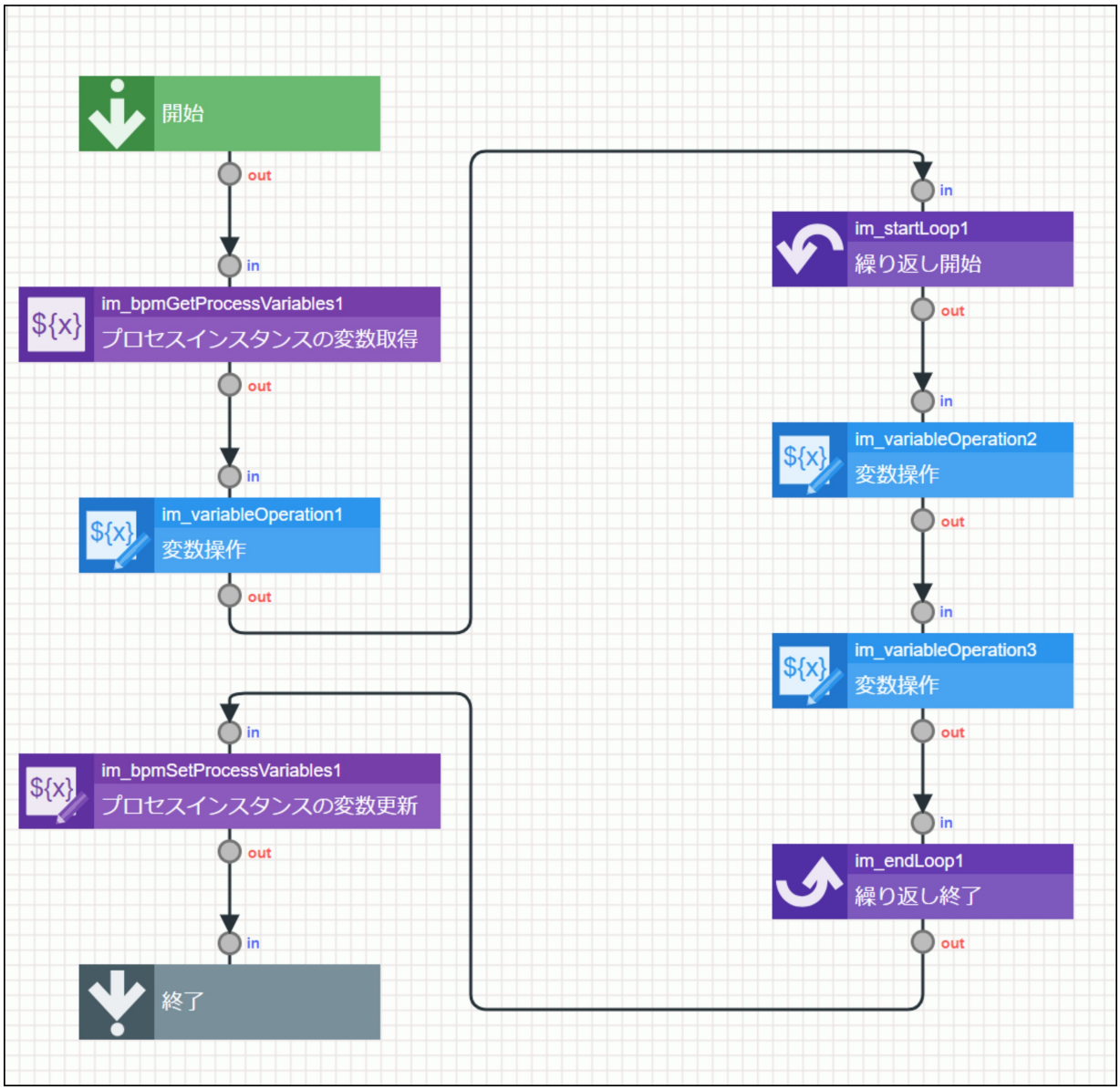
図：使用する入力フォーム

> **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。

ロジックフローを作成する

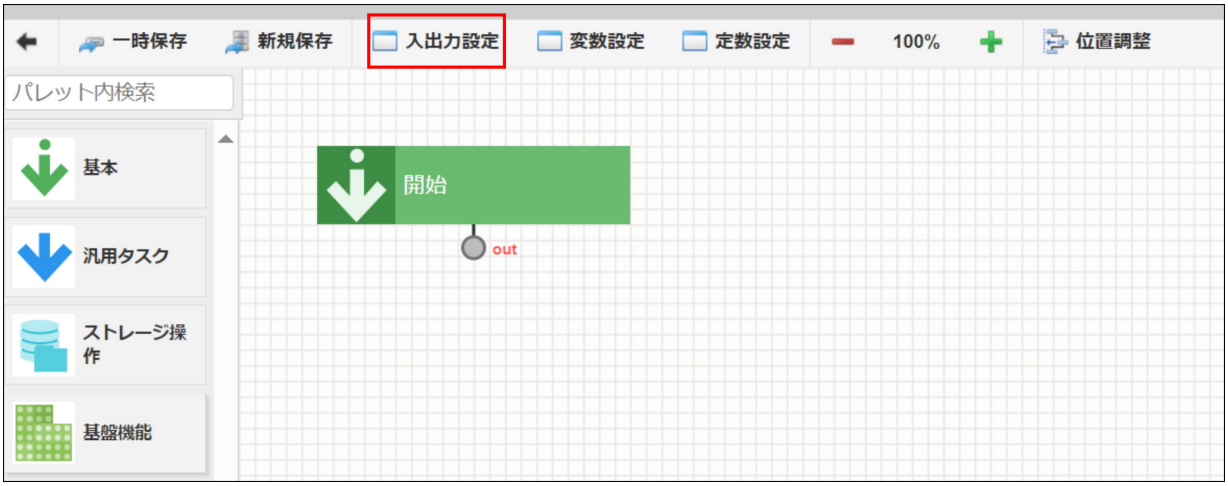
プロセスインスタンス変数からテーブルデータを受け取り、それをもとに新たなテーブルデータを作成しプロセスインスタンス変数に登録する ロジックフローを作成します。

1. 「開始タスク」、「プロセスインスタンスの変数取得タスク」、「変数操作タスク1」、「変数操作タスク2」、「繰り返し開始タスク」、「変数操作タスク3」、「変数操作タスク4」、「繰り返し終了タスク」、「プロセスインスタンスの変数更新タスク2」、「終了タスク」を配置し、順に接続します。



図：ロジックフロー

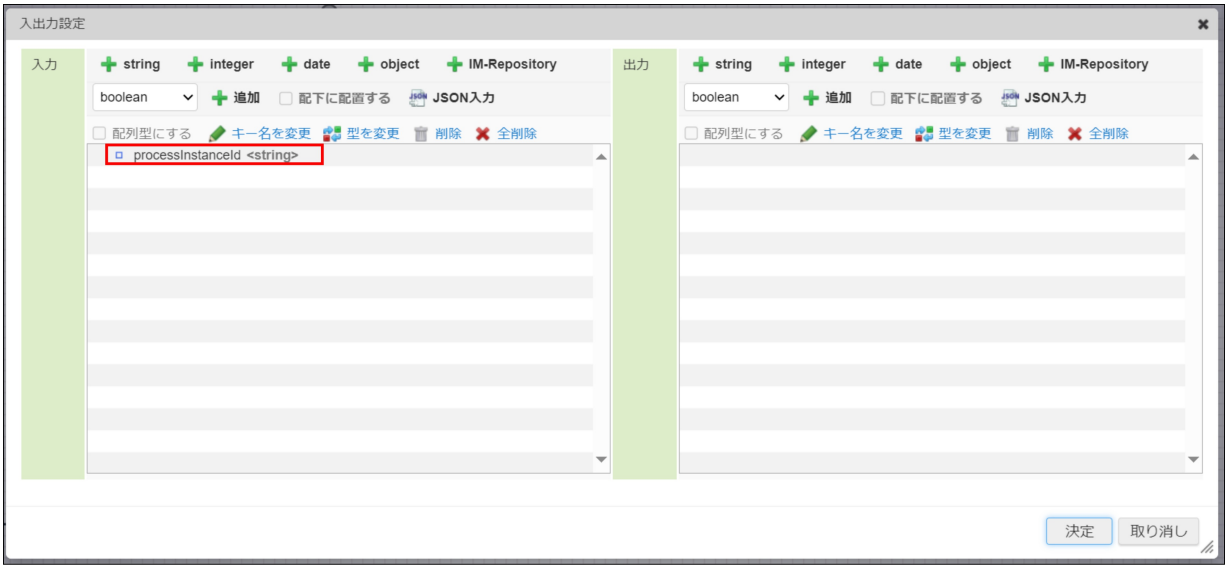
2. ツールバーの「入出力設定」をクリックします。



図：「ツールバー」 - 「入出力設定」

3. 入力値を以下のように設定します。

キー名	型
processInstanceId	<string>



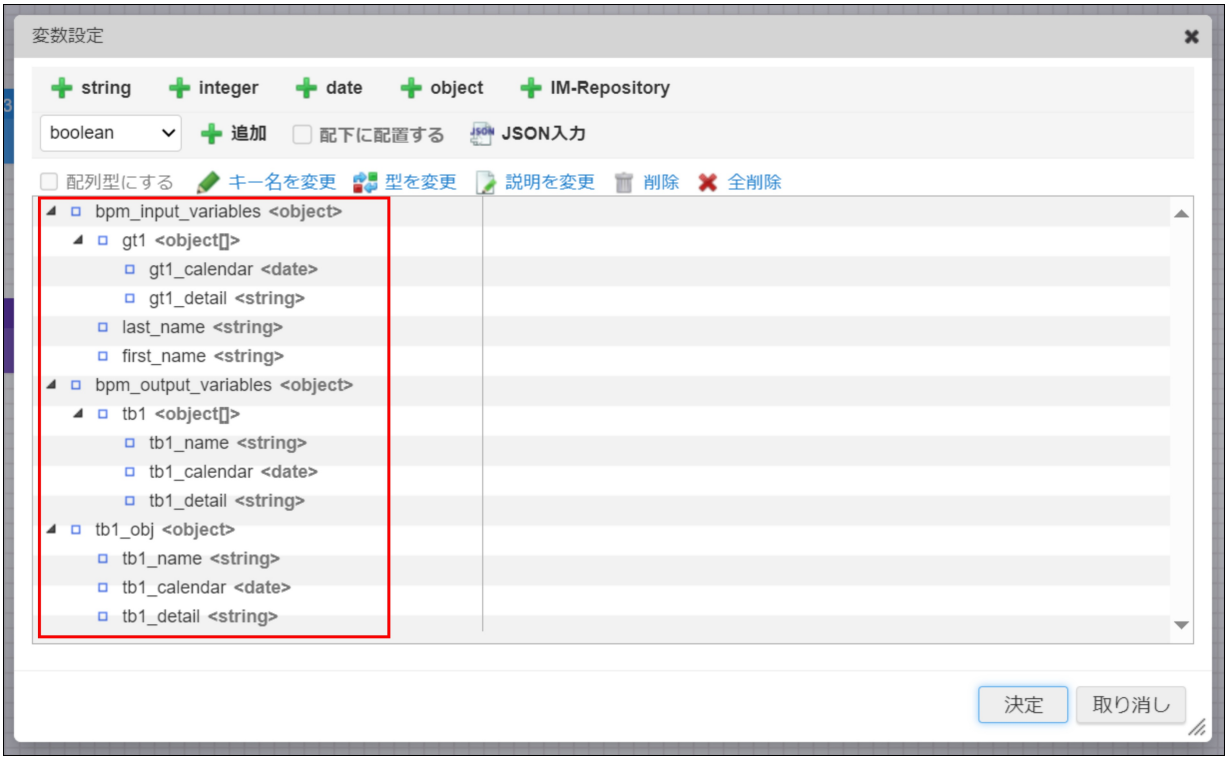
図：「入出力設定」 - 「入力」

> **コラム**
>
> 入力値として直接プロセスインスタンス変数を指定することも可能ですが、今回はプロセスインスタンスIDのみ入力します。
> ロジックフロー内でプロセスインスタンスIDからプロセスインスタンス変数を取得します。
> この方法には、以下の利点があります。
>
> - ロジックフローのデバッグ機能を使用する際に実行中のプロセスインスタンスのIDを用いてデバッグを行えること
> - IM-BPMプロセスのIM-LogicDesignerタスクの入力データにはプロセスインスタンスIDのみ設定するだけでよいこと

4. プロセスインスタンス変数を格納するための変数を設定します。
変数設定ダイアログを開き、以下の値を追加します。

キー名	型
bpm_input_variables - gt1 - gt1_calendar	<date>
bpm_input_variables - gt1 - gt1_detail	<string>
bpm_input_variables - last_name	<string>
bpm_input_variables - name	<string>
bpm_output_variables - tb1	<map[]>

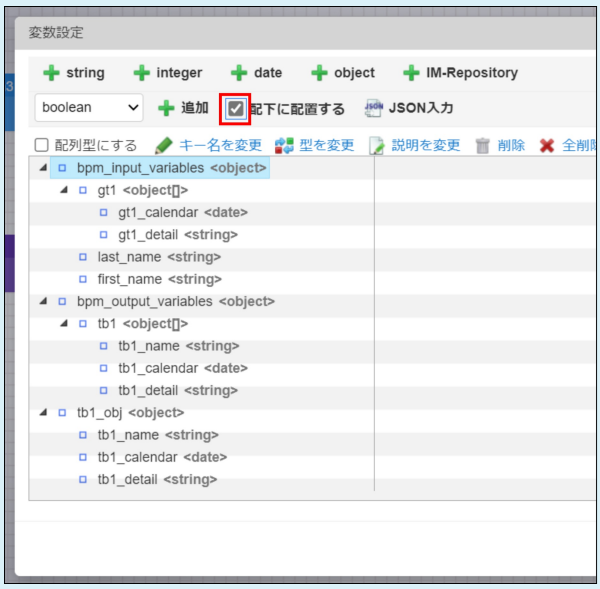
キー名	型
bpm_output_variables - fullname	<string>
tb1_obj - tb1_calendar	<date>
tb1_obj - tb1_detail	<string>



図：「変数設定」

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：「変数設定」

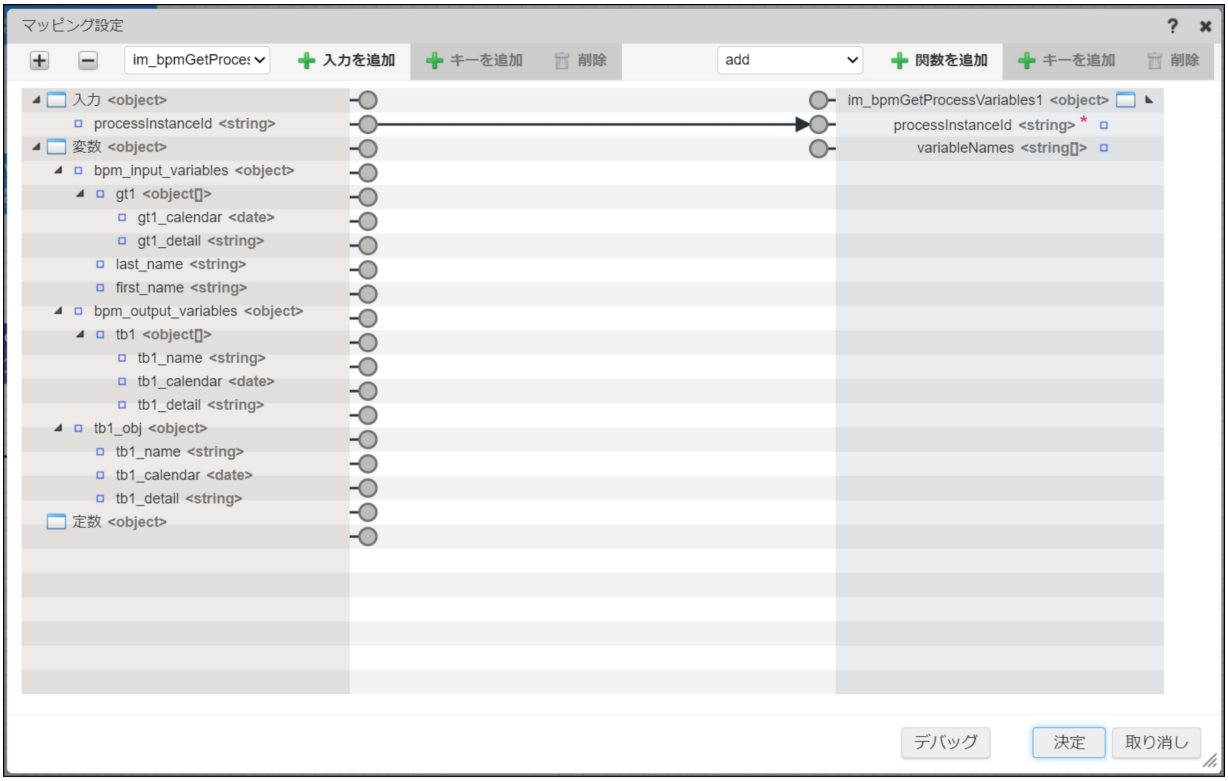
コラム

変数はJSON形式でも入力できます。
JSON入力については、「IM-LogicDesigner ユーザ操作ガイド」-「入出力/変数/定数を設定する」 -「JSON入力ダイアログ詳細」を参照してください。


```
{
  "bpm_input_variables": {
    "gt1": [
    {
      "gt1_calendar": null,
      "gt1_detail": ""
    }
    ],
    "last_name": "",
    "first_name": ""
  },
  "bpm_output_variables": {
    "tb1": [
    {
      "tb1_name": "",
      "tb1_calendar": null,
      "tb1_detail": ""
    }
    ]
  },
  "tb1_obj": {
    "tb1_name": "",
    "tb1_calendar": null,
    "tb1_detail": ""
  }
}
```


5. プロセスインスタンスIDから、プロセスインスタンス変数を取得します。
「プロセスインスタンスの変数取得タスク」のマッピング設定を開き以下をマッピングします。

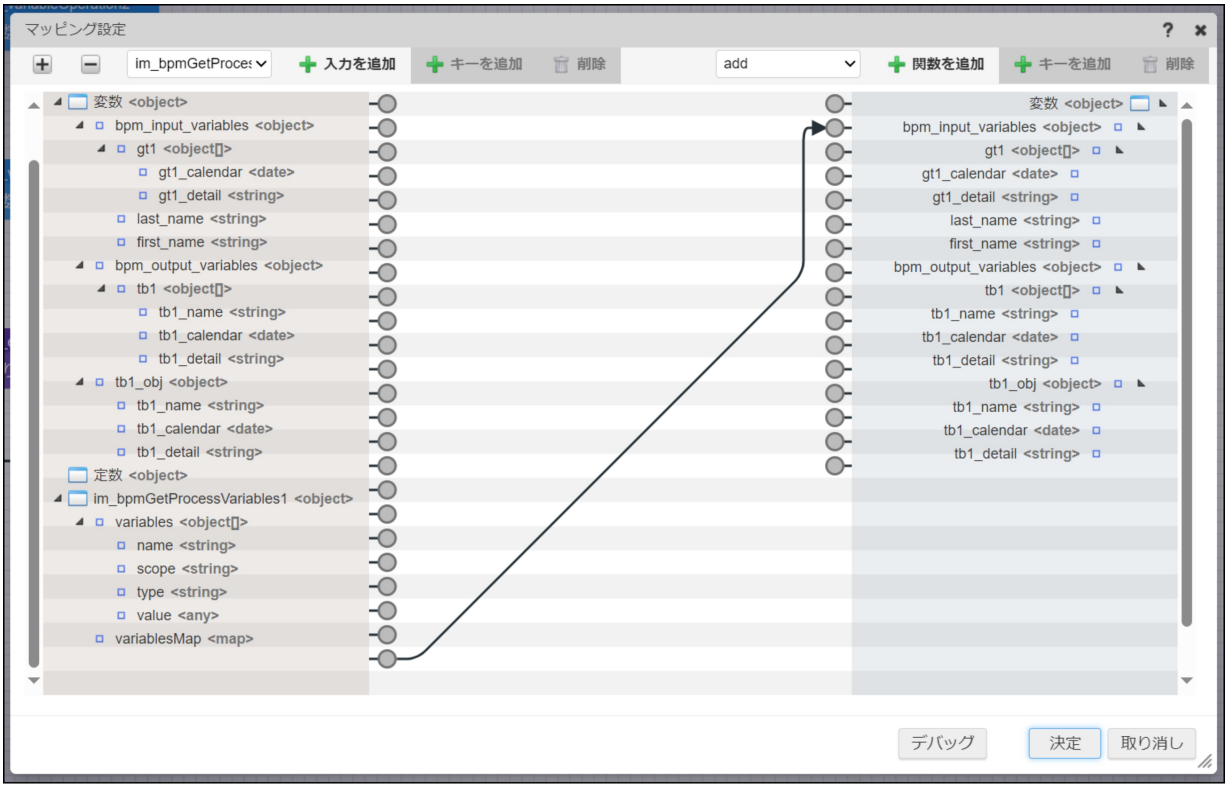
入力（始点）	出力（終点）
入力<object> - processInstanceId<string>	im_bpmGetProcessVariables1<object> - processInstanceId<string>



図：「プロセスインスタンスの変数取得タスク」 -「マッピング設定」

6. 「プロセスインスタンスの変数取得タスク」で取得した変数を ロジックフローの変数に設定します。
「変数操作タスク1」のマッピング設定を開き以下をマッピングします。

入力（始点）	出力（終点）
im_bpmGetProcessVariables1<object> - variablesMap<map>	変数<object> - bpm_input_variables<object>



図：「変数操作タスク1」 - 「マッピング設定」

コラム

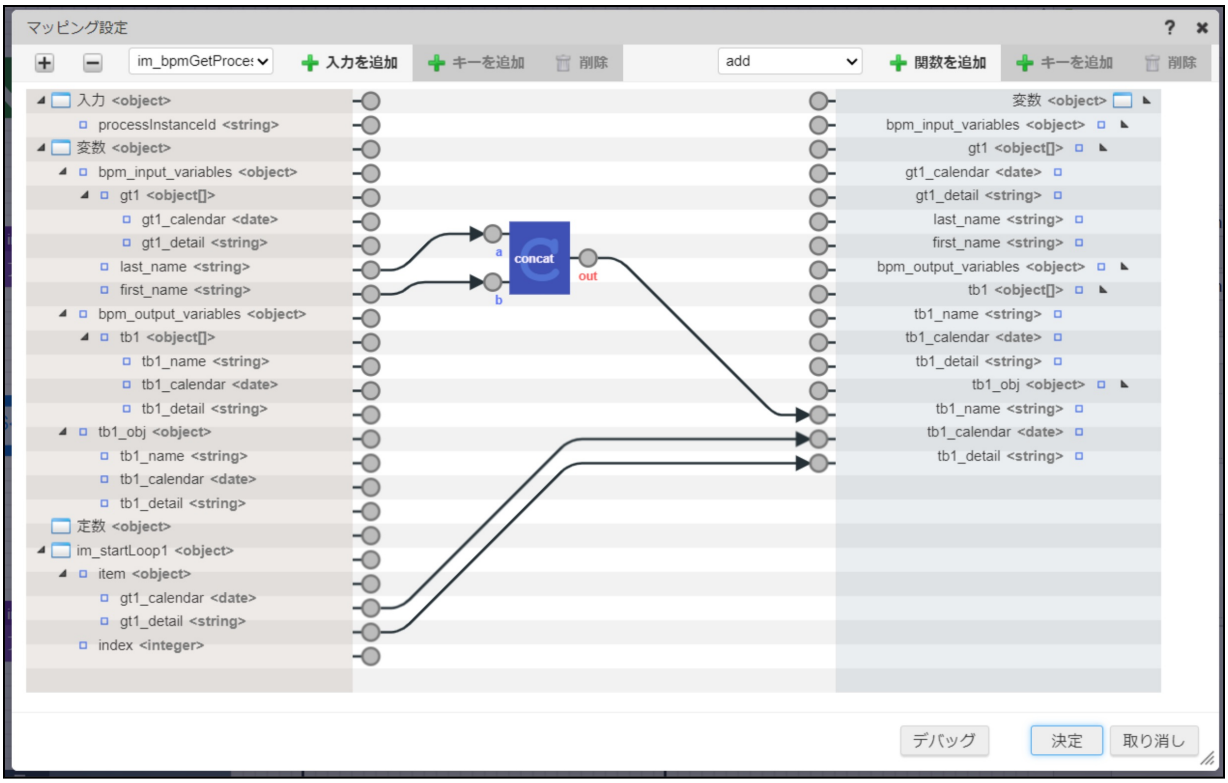
variablesMap には、key：変数名、value：値としてプロセスインスタンス変数がマップとして格納されています。
「プロセスインスタンスの変数取得タスク」については、「IM-LogicDesigner仕様書」-「タスク一覧」-「IM-BPM」 -「プロセスインスタンスの変数取得」を参照してください。

マッピングの左右で型が異なりますが、右側に定義されている子階層と同名のキーに対応する値が格納されたobject型のオブジェクトを生成後、右側に渡されます。
右側のオブジェクトの子階層には IM-BPMプロセスに登録される変数と同じ型・変数名の変数を設定しているため、自動でマッピングされます。

7. 取得した変数を、テーブルデータとして IM-BPMプロセスに返却するための変数に格納します。
まずは返却用の値のうち、テーブル1行分のデータを作成します。
「変数操作タスク2」のマッピング設定を開き以下をマッピングします。

入力（始点）	出力（終点）
変数<object> - bpm_input_variables<object> - last_name<string>	concat : a
変数<object> - bpm_input_variables<object> - first_name<string>	concat : b

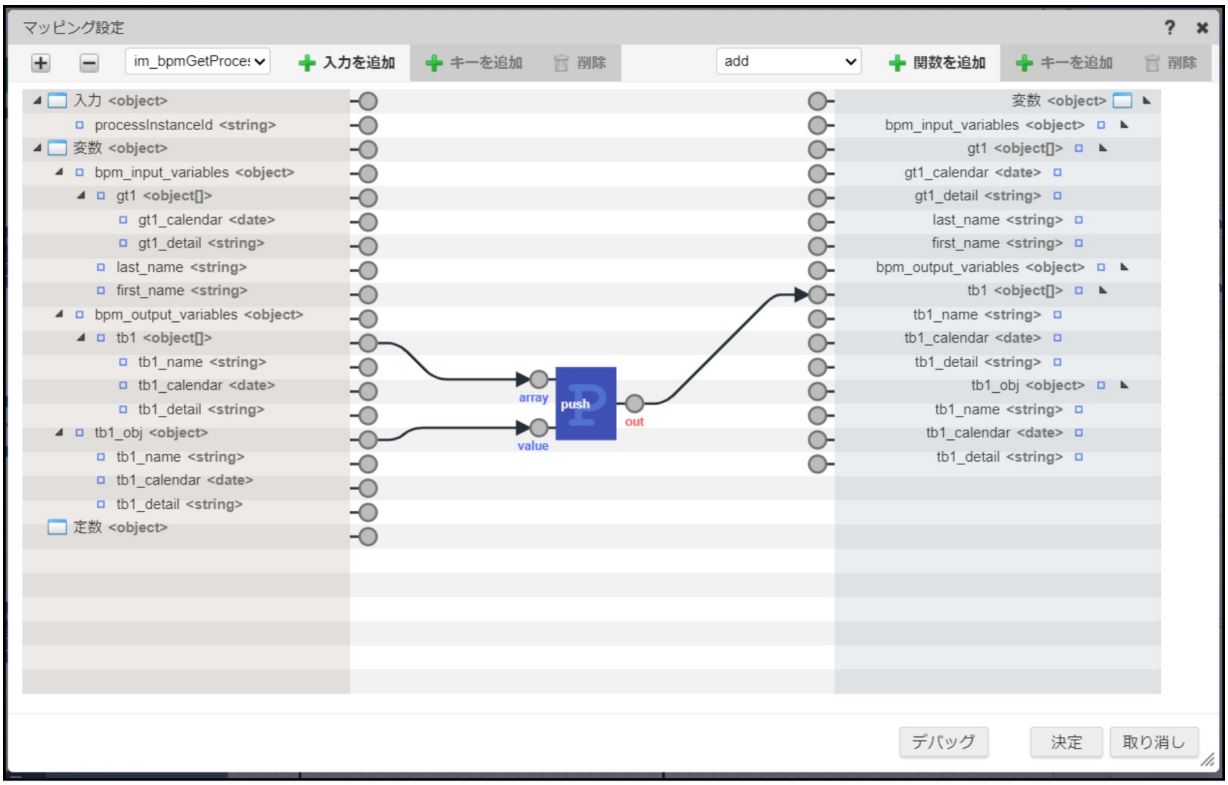
入力（始点）	出力（終点）
concat : out	変数<object> - bpm_output_variables<object> - tb1_name<string>
im_startLoop1<object> - item<object> - gt1_calendar<date>	変数<object> - tb1_obj<object> - tb1_calendar<date>
im_startLoop1<object> - item<object> - gt1_detail<string>	変数<object> - tb1_obj<object> - tb1_detail<string>



図：「変数操作タスク2」 - 「マッピング設定」

作成した1行分のテーブルデータを格納用の変数に設定します。
「変数操作タスク3」のマッピング設定を開き以下をマッピングします。

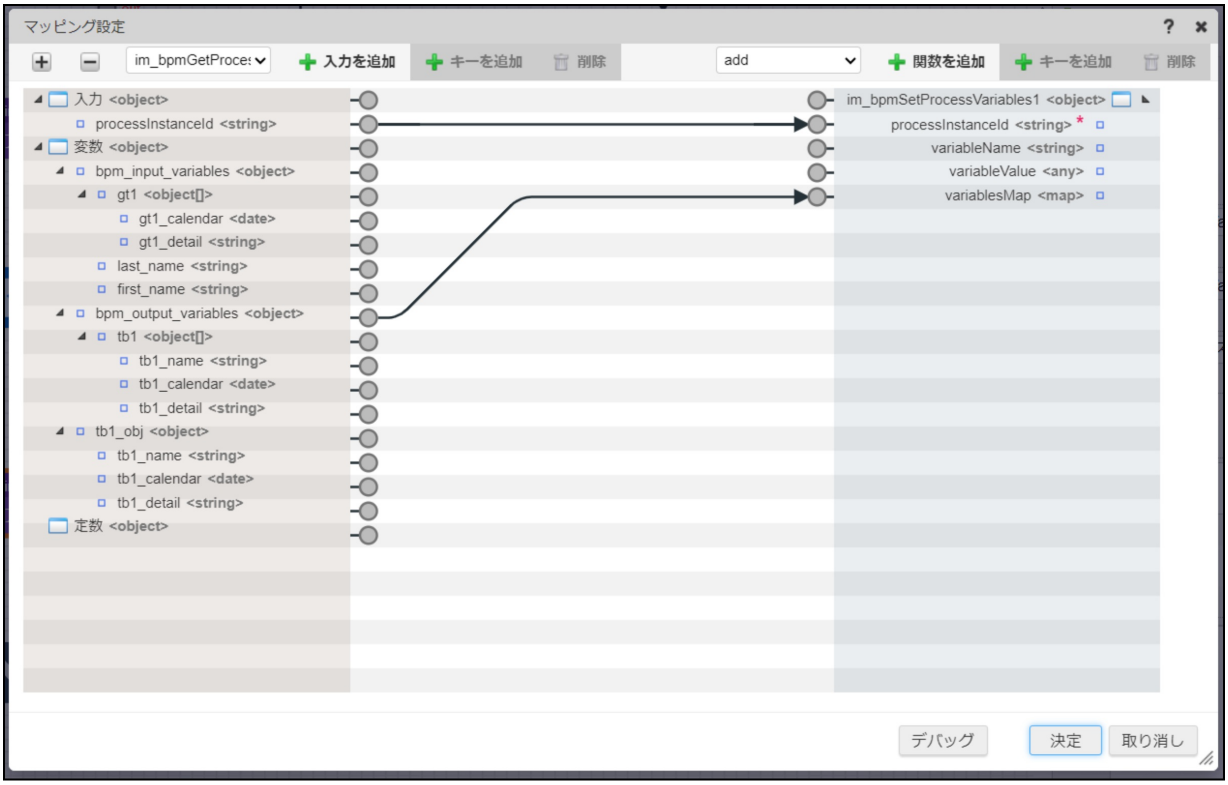
入力（始点）	出力（終点）
変数<object> - bpm_output_variables<object> - tb1<object[]>	push : array
変数<object> - tb1_obj<object>	tb1_obj<object> - tb1_detail<string>
push : out	変数<object> - bpm_output_variables<object> - tb1<object[]>



図：「変数操作タスク3」 - 「マッピング設定」

8. 「プロセスインスタンスの変数更新タスク」のマッピング設定を開き以下をマッピングします。

入力（始点）	出力（終点）
入力<object> - processInstanceId<string>	im_bpmGetProcessVariables1 - processInstanceId<string>
変数<object> - bpm_output_variables<object>	im_bpmSetProcessVariables1 - variablesMap<map>



図：「プロセスインスタンスの変数更新タスク」 - 「マッピング設定」

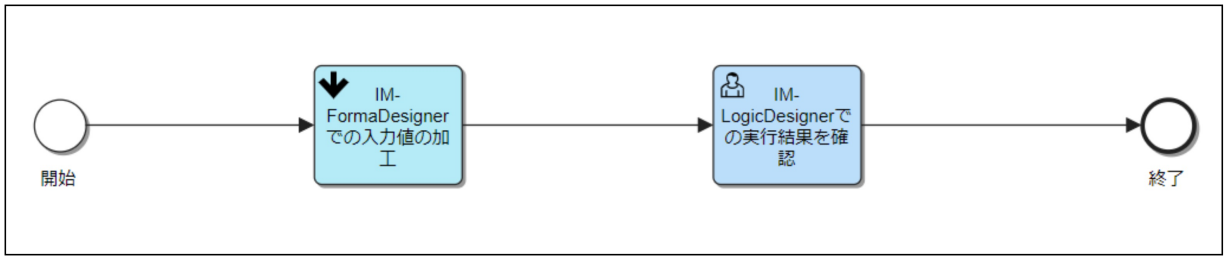
9. 新規保存ボタンをクリックし、以下を設定しロジックフローを保存します。
   - フロー定義ID : logicdesigner_task_forma_tb_item
   - フロー定義名 : 【チュートリアル】logicdesigner_task_forma_tb_item
   - フローカテゴリ :

- ID : im_logicdesigner-data
- 名称 : BPMチュートリアル

> コラム
> - ロジックフロー定義の作成方法については「IM-LogicDesigner ユーザ操作ガイド」を参照してください。

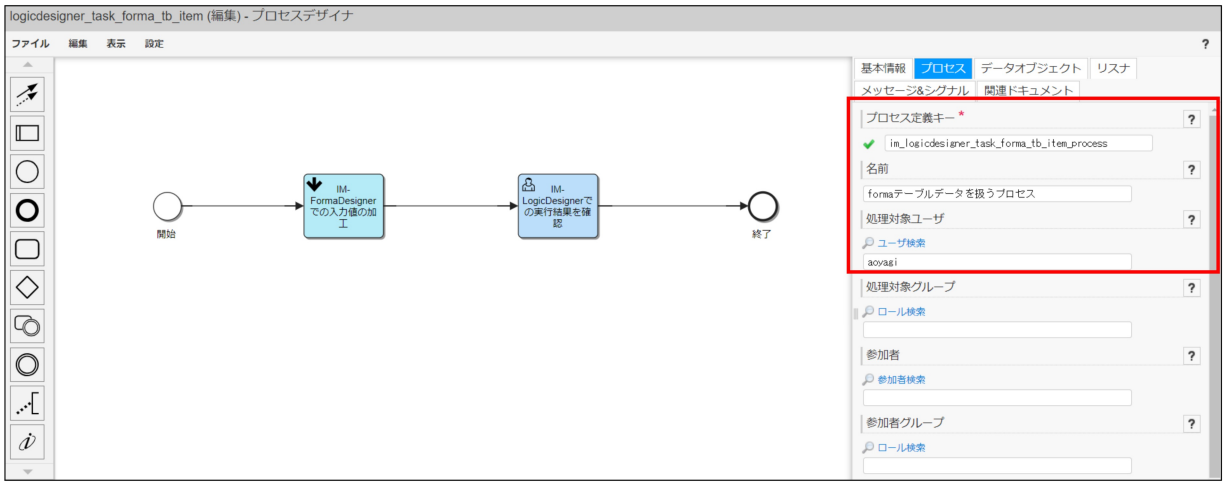
プロセス定義を作成する

1. 開始イベント、IM-LogicDesignerタスク、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。



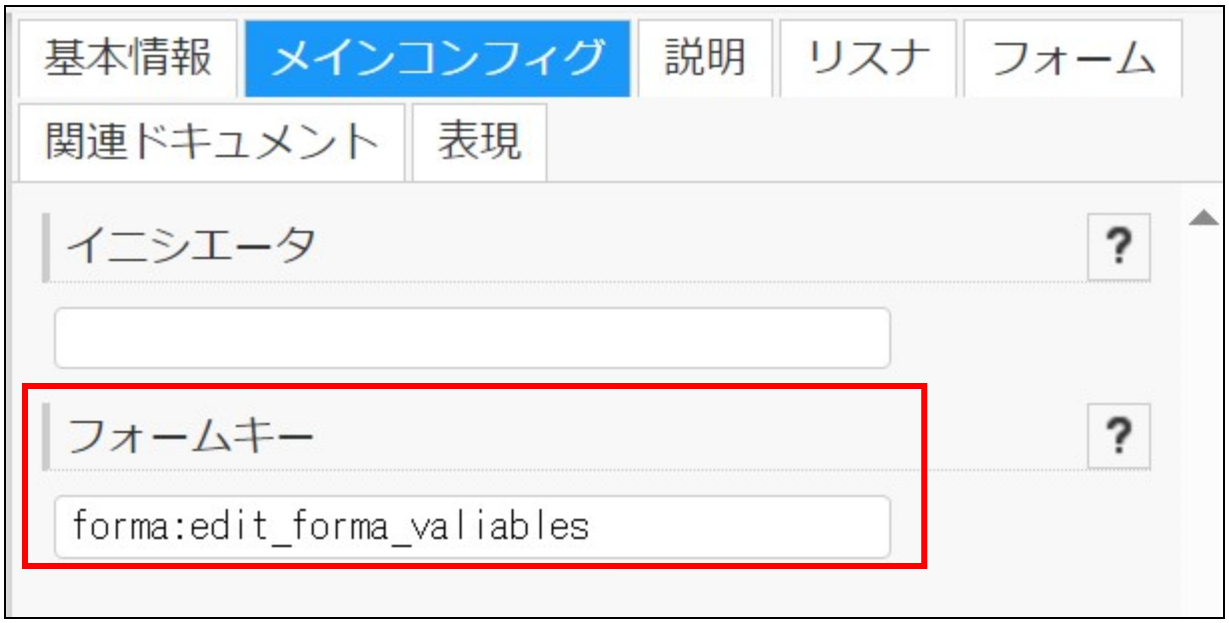
図：完成イメージ

2. 「プロセス」に、プロセス定義キーとプロセス名、処理対象ユーザを設定します。
余白を選択し、「プロセス」タブの「プロセス定義キー」、「名前」、「処理対象ユーザ」に以下の値を設定します。
   - プロセス定義キー : im_logicdesigner_task_forma_tb_item_process
   - 名前 : formaテーブルデータを扱うプロセス
   - 処理対象ユーザ : aoyagi



図：「プロパティ」 - 「プロセス」

3. 「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:edit_forma_variables



図：「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

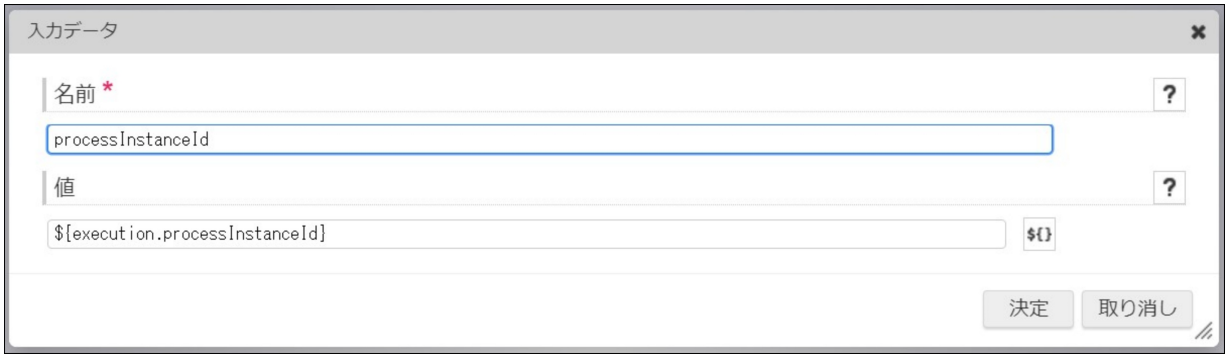
4. 「IM-LogicDesignerタスク」に、作成したロジックフローを設定します。
「IM-LogicDesignerタスク」を選択し、「メインコンフィグ」タブの「フローID」、「利用するバージョン」に以下の値を設定します。
   - フローID : logicdesigner_task_forma_tb_item
   - 利用するバージョン : 最新バージョンを利用

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

5. 「IM-LogicDesignerタスク」に、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」をクリックします。

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
  - 名前：processInstanceId
  - 値：${execution.processInstanceId}



図：「入力データ」

6. 「ユーザタスク」に、インポートした「IM-FormaDesigner」のアプリケーションと担当者を設定します。
   「ユーザタスク」を選択し、「メインコンフィグ」タブの「フォームキー」と「担当者」に以下の値を設定します。
   - 担当者：aoyagi
   - フォームキー：forma:edit_forma_variables

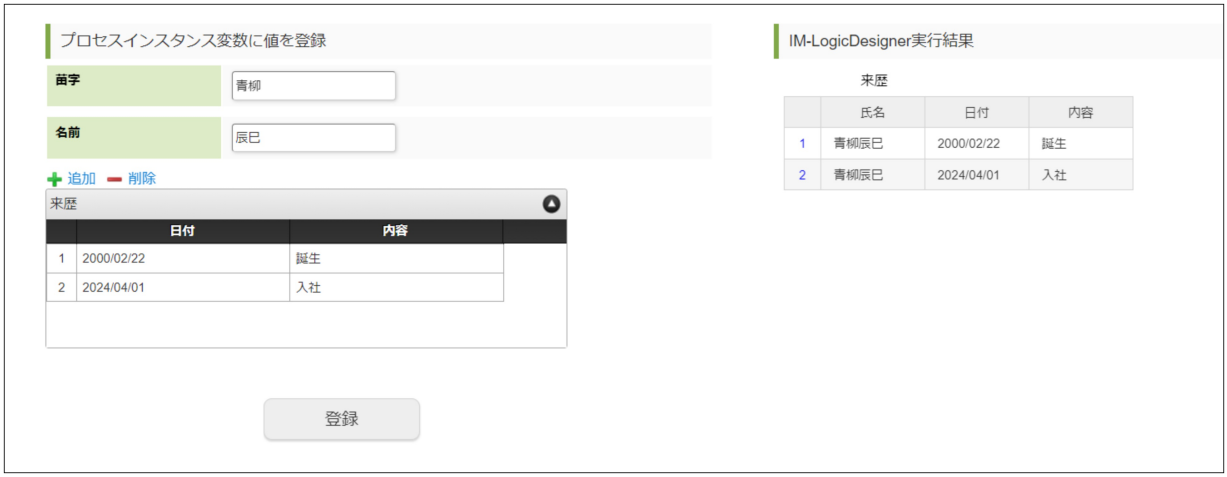
図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

7. 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します。また、実行した結果の確認を行います。

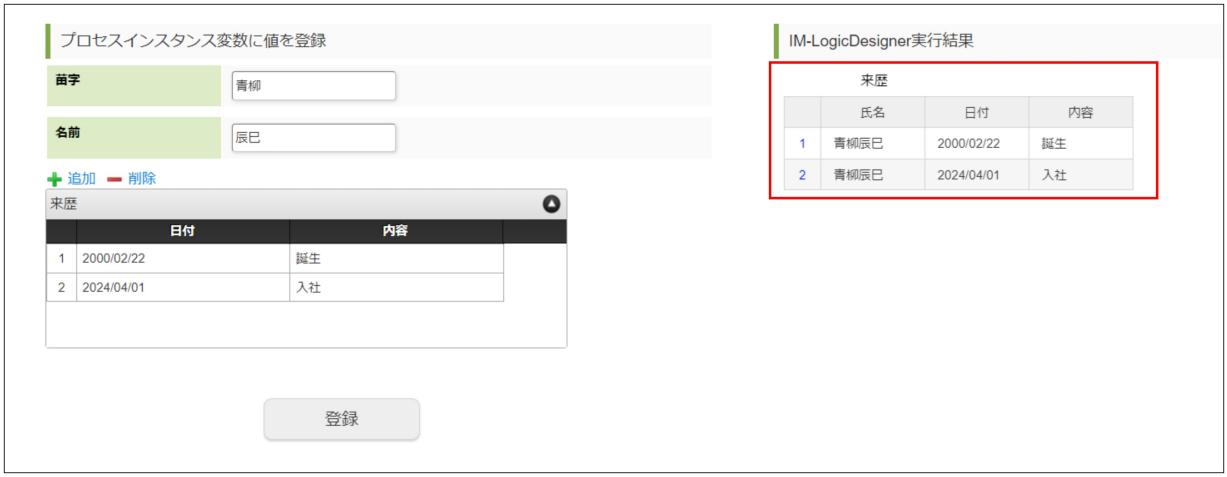
1. 「プロセス開始一覧」画面にて、プロセス定義名 formaデータを連携するプロセス を開始します。
2. 開始イベントに設定したFormaアプリケーション「edit_forma_variables」に遷移します。
   適当に入力し、「登録」をクリックします。



図：Formaアプリケーション「edit_forma_variables」

3. 「タスク一覧」 画面の個人タスクを確認します。
   タスク名「LogicDesigner実行結果を確認」の「処理」をクリックし、タスクを処理します。

4. フォーム右側のテーブルアイテムに、プロセスの開始時に入力したデータをもとに作成されたデータが入力されていることを確認します。



図：Formaアプリケーション「edit_forma_variables」

申請タスク

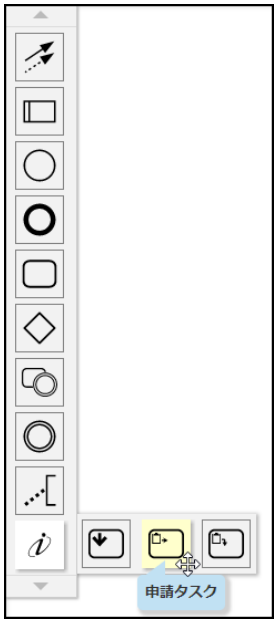
申請タスクを使用してワークフローと連携する

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

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

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

申請タスクを配置する



図：パレット - intra-mart - 申請タスク

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

実行するワークフローを設定する

1. プロパティエリアにて「メインコンフィグ」タブを開きます。

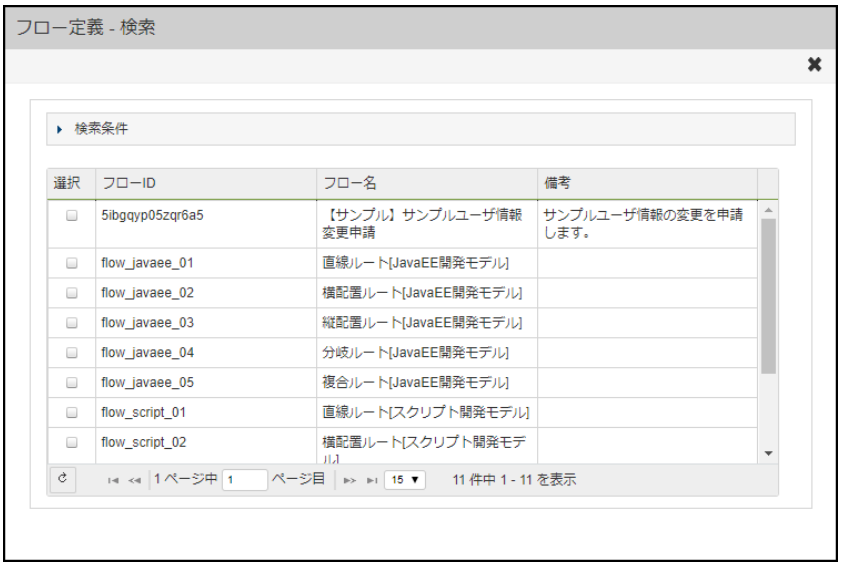
図：申請タスク：プロパティ - メインコンフィグ

2. 「アプリケーション」を設定します。

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

3. 「フローID」を設定します。

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



図：フロー定義検索ウィンドウ

4. 「案件名」を設定します。

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

5. 「申請実行者コード」を設定します。

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

図：申請実行者コードダイアログ

6. 「申請権限者コード」を設定します。

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

図：申請権限者コードダイアログ

7. 「入力データ」を設定します。

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

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

8. 「案件の処理結果を格納する変数名」を設定します。

   workflowResult を設定します。

9. 「案件のユーザデータを格納する」を設定します。

   チェックボックスを有効にします。

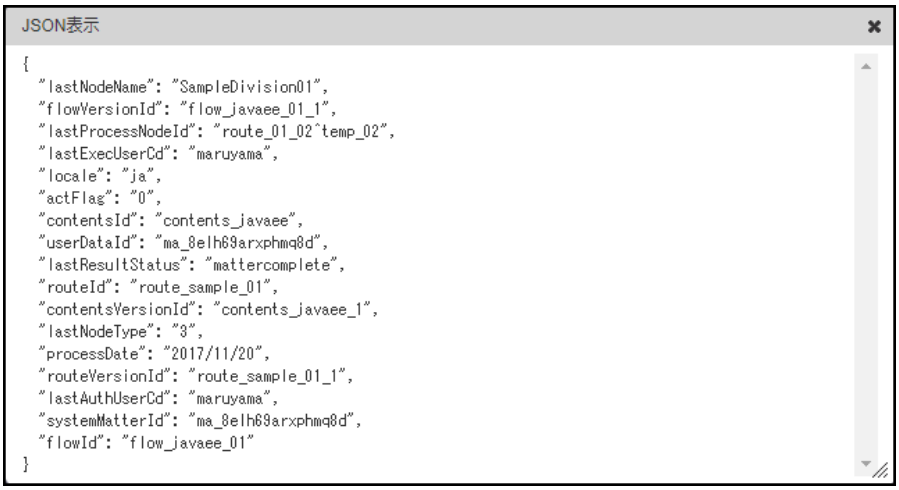
10. 「案件のユーザデータを格納する変数名」を設定します。

    workflowData を設定します。

実行結果を確認する

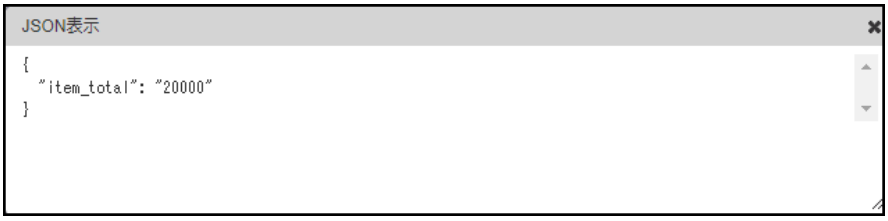
このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

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



図：「変数一覧」画面 - JSON表示

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



図：「変数一覧」画面 - JSON表示

起票タスク

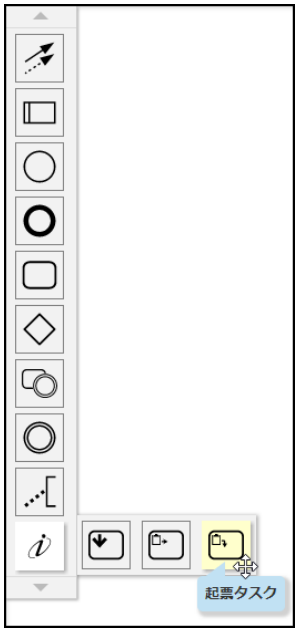
起票タスクを使用してワークフローと連携する

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

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

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

起票タスクを配置する



図：パレット - intra-mart - 起票タスク

1. パレットの にカーソルを合わせ、パレット右側に現れるintra-martタスクの一覧から をドラッグ＆ドロップして配置します。

2. 配置されたエレメントをクリックして選択することにより、プロパティエリアに選択したエレメントのプロパティが表示されます。

実行するワークフローを設定する

1. プロパティエリアにて「メインコンフィグ」タブを開きます。

図：起票タスク：プロパティ - メインコンフィグ

2. 「アプリケーション」を設定します。

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

3. 「フローID」を設定します。

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

図：フロー定義検索ウィンドウ

4. 「案件名」を設定します。

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

5. 「起票者コード」を設定します。

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

図：起票者コードダイアログ

6. 「案件の処理結果を格納する変数名」を設定します。

   workflowResult を設定します。

7. 「案件のユーザデータを格納する」を設定します。

   チェックボックスを有効にします。

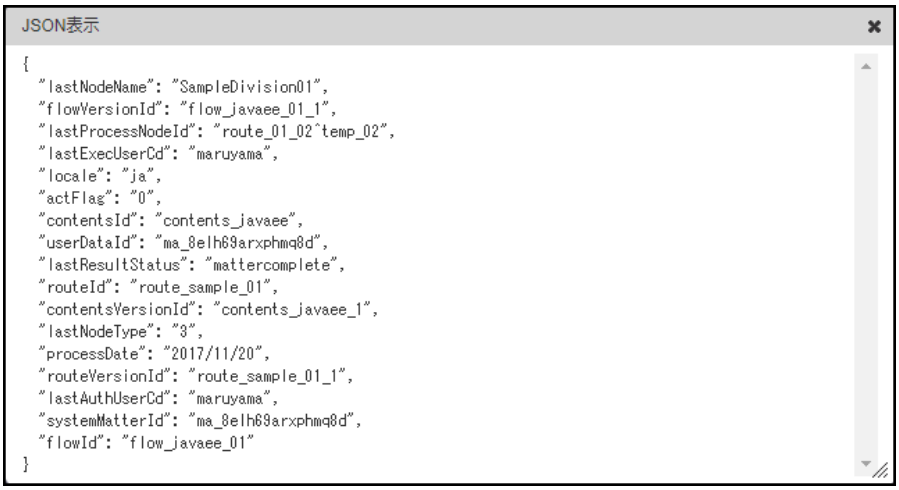
8. 「案件のユーザデータを格納する変数名」を設定します。

   workflowData を設定します。

実行結果を確認する

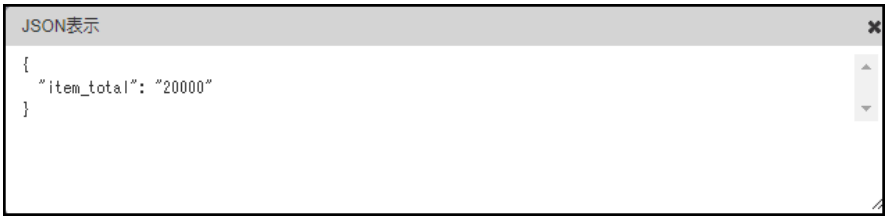
このチュートリアルで作成したプロセス定義を実行環境にデプロイし、実行した結果の確認を行います。

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



図：「変数一覧」画面 - JSON表示

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



図：「変数一覧」画面 - JSON表示

起票タスクを使用してワークフローの申請ノード処理対象者を動的に設定する

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

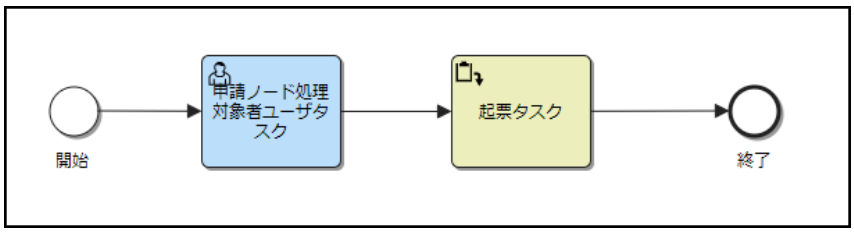
> **コラム**
>
> このチュートリアルで作成するプロセス定義のサンプルを以下のリンクからダウンロードできます。
> **draft_task_apply_user_plugin.bpmn**
>
> このサンプルは「プロセス定義アップロード」機能でプロジェクトにアップロードできます。
> アップロード手順は「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」を参照してください。

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

ワークフローの申請ノード処理対象者を動的に設定するプロセス定義を作成する

以下のユーザを「申請ノード処理対象者」に設定します。

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



図：完成イメージ

1. 開始イベントの「イニシエータ」に `starter` を設定します。

図：開始イベント：プロパティ - メインコンフィグ

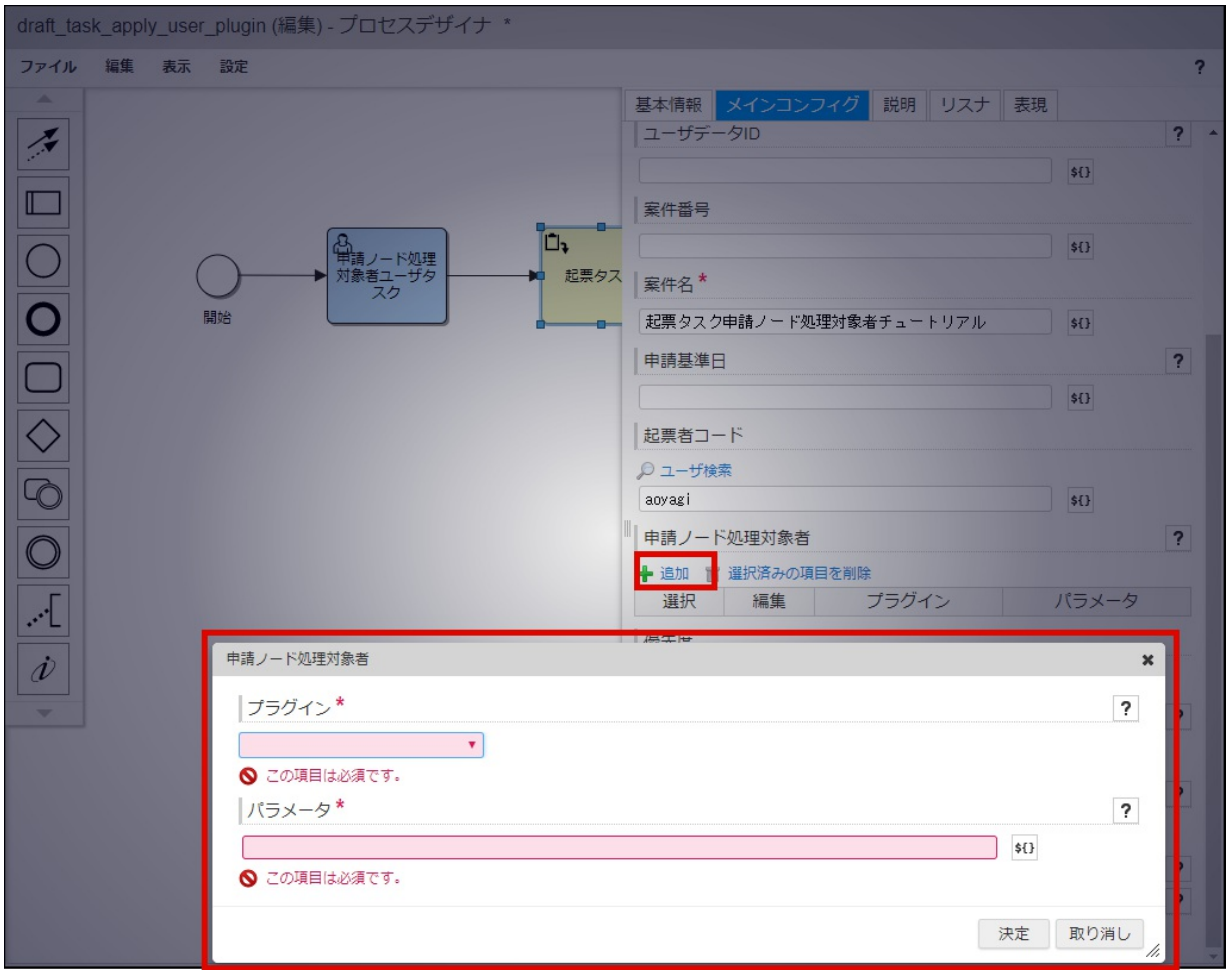
2. ユーザタスクの「ID」に draft_task_before_user_task を設定します。
3. ユーザタスクの「名前」に 申請ノード処理対象者ユーザタスク を設定します。

図：ユーザタスク：プロパティ - 基本情報

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

図：ユーザタスク：プロパティ - メインコンフィグ

5. 起票タスクの各プロパティを設定します。
   - アプリケーション: IM-Workflow
   - フローID: flow_javaee_01
   - 案件名: 起票タスク申請ノード処理対象者チュートリアル
   - 起票者コード: aoyagi
6. 起票タスクの「申請ノード処理対象者」を設定します。
   1. 「申請ノード処理対象者」の「追加」リンクをクリックします。



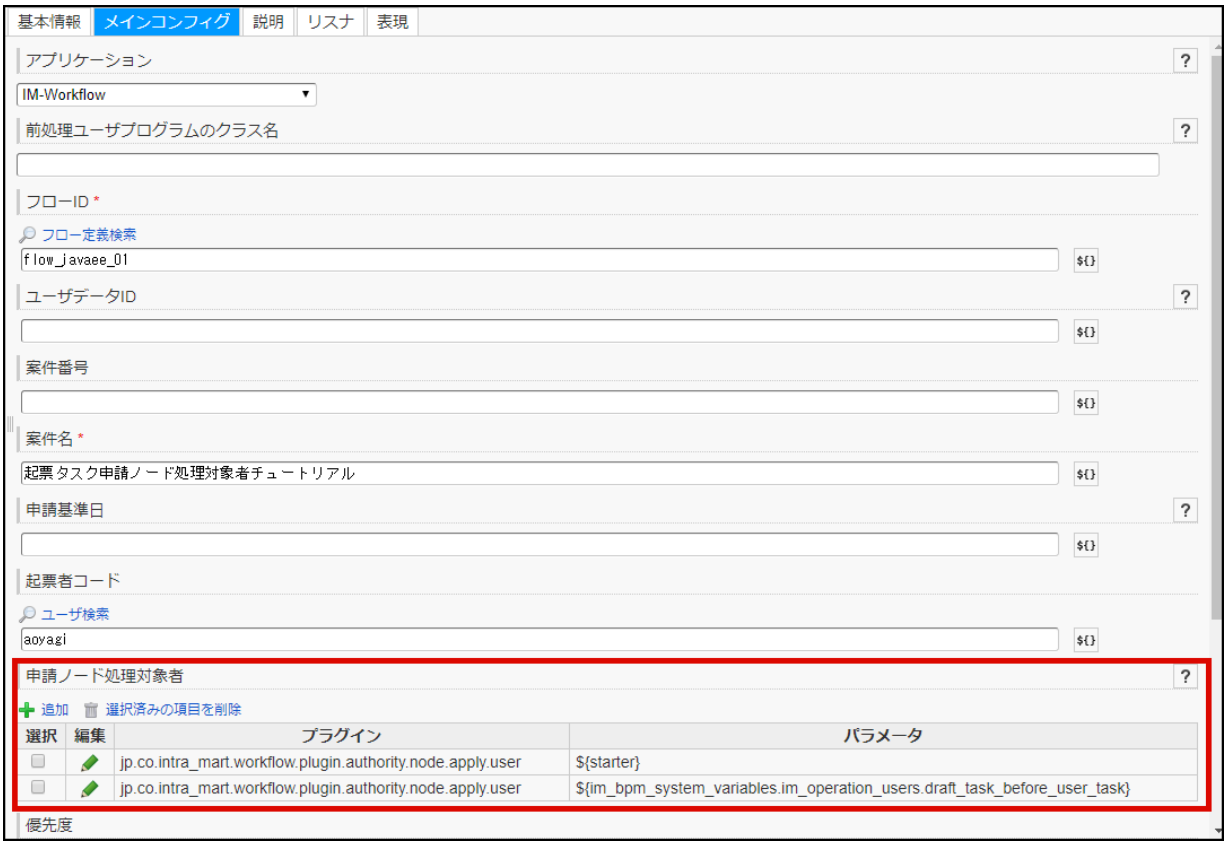
図：申請ノード処理対象者ダイアログ

2. 「プラグイン」と「パラメータ」を設定します。

図：申請ノード処理対象者ダイアログ

以下を設定します。

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



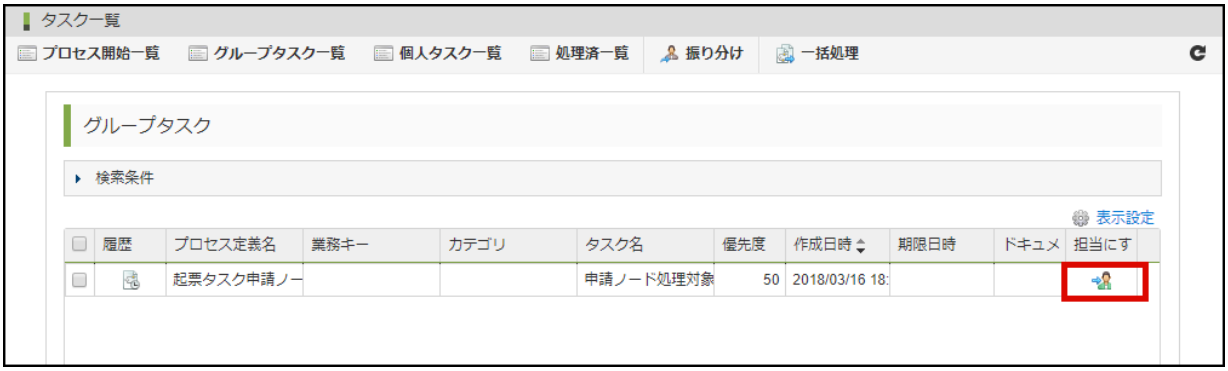
図：起票タスク：プロパティ - メインコンフィグ

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

実行結果を確認する

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

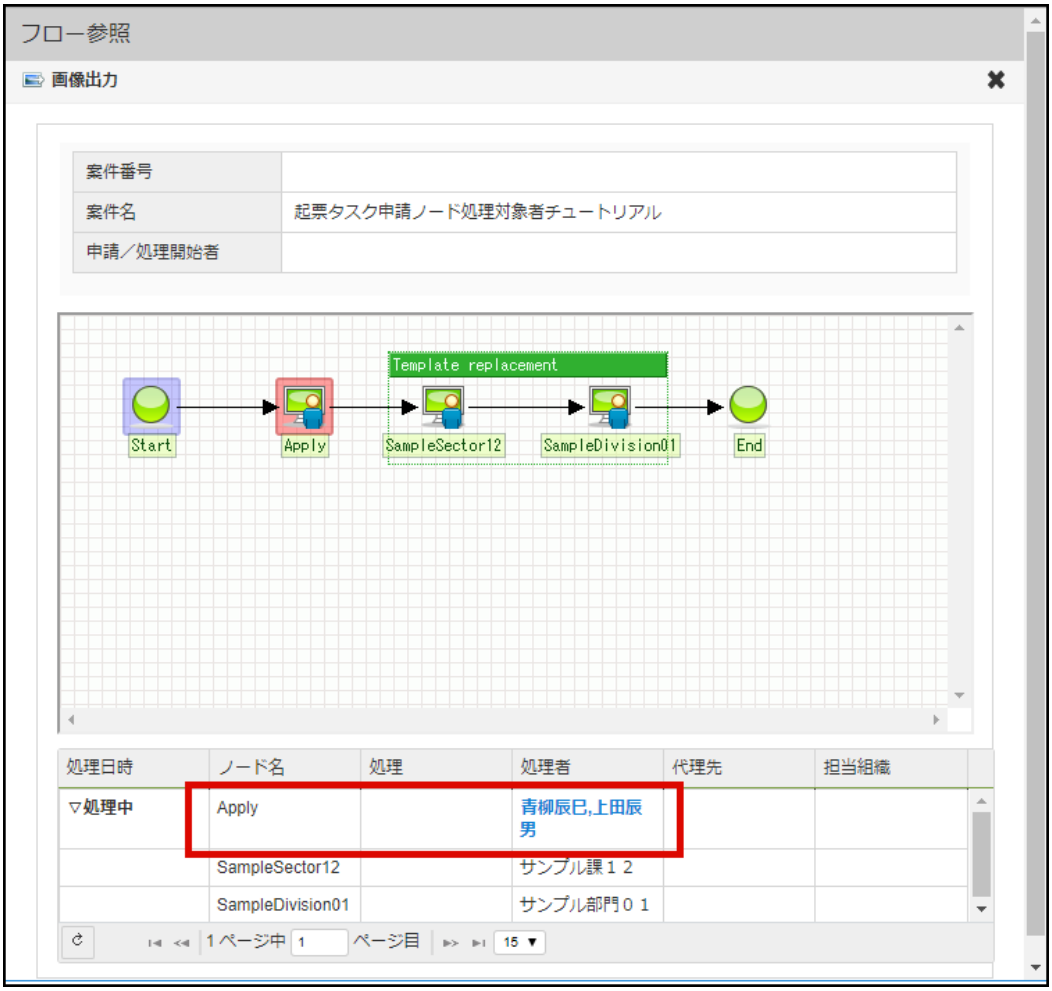
1. プロセスを開始すると、「申請ノード処理対象者ユーザタスク」がグループタスクに作成されます。
2. 「申請ノード処理対象者ユーザタスク」を担当にし、続けて処理を行います。
   プロセスを開始したユーザと異なるユーザで行います。「IM-BPM ユーザ」のロールが付与されたユーザが対象です。



図：担当にする

図：処理する

3. ワークフローの未処理一覧からフローを確認します。



図：フロー参照

起票タスクと変数を連携する

このチュートリアルでは、開始イベントとユーザタスクのフォームにFormaアプリケーション「【チュートリアル】起票タスクと変数を連携する」を設定し、起票タスクにIM-BIS「【チュートリアル】起票タスクと変数を連携する」を設定することで、起票タスクと変数を連携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。
IM-BIS for Accel Platform の詳細については、「IM-BIS 仕様書」を参照してください。

プロセスを進めていく中で、「IM-BIS for Accel Platform」のIM-BIS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材をインポートしてください。

- IM-BIS

  im_bis-draft_task_link_variables.zip

- Formaアプリケーション

  im_forma_designer-draft_task_link_variables.zip

また、「IM-BIS」を使用するには、「BIS担当者」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）」、「上田辰男（ユーザコード：ueda）」を検索し、以下の設定を行ってください。

- 「ロール」タブ - 「BIS担当者」ロールの追加

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。

- IM-BIS: 「IM-BIS システム管理者操作ガイド」-「一括インポート・エクスポートを行う」
- 「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」

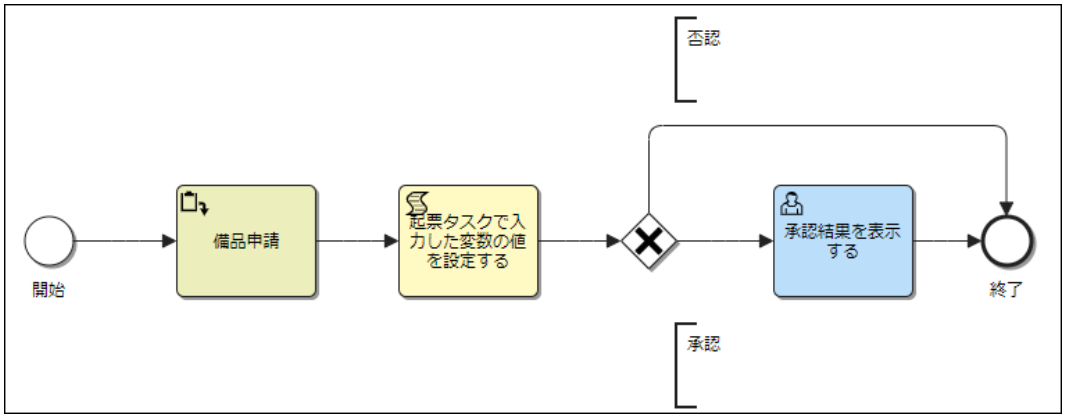
コラム

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_link_variables.bpmn

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

- エレメントを配置する
- 開始イベントのプロパティを設定する
- 起票タスクのプロパティを設定する
- スクリプトタスクのプロパティを設定する
- ユーザタスクのプロパティを設定する
- 排他ゲートウェイの分岐条件を設定する
- IM-BISのフィールド識別IDを確認する
- 実行結果を確認する

エレメントを配置する

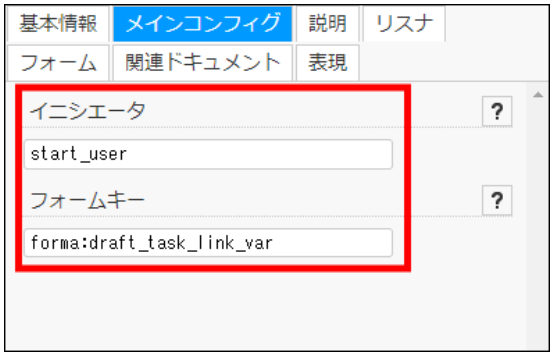


図：配置イメージ

1. 開始イベント、起票タスク、スクリプトタスク、排他ゲートウェイ、ユーザタスク、および、終了イベントを配置して、図のように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

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. 「イニシエータ」に `start_user` を入力します。
3. 「フォームキー」に `forma:draft_task_link_var` を入力します。



図：「開始イベント」 - 「プロパティ」 - 「メインコンフィグ」

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。

起票タスクのプロパティを設定する

1. 起票タスクのプロパティで「基本情報」タブを開きます。
2. 「名前」に 備品申請 を入力します。

図：「起票タスク」 -「プロパティ」 -「基本情報」

1. 起票タスクの「メインコンフィグ」タブを開きます。
2. 「アプリケーション」に IM-BIS を設定します。
3. 「フローID」を入力します。
   「 フローID」の「フロー定義検索」リンクをクリックします。
   フロー定義検索ウィンドウから「【チュートリアル】起票タスクと変数を連携する」を検索して選択すると、フローIDが自動入力されます。

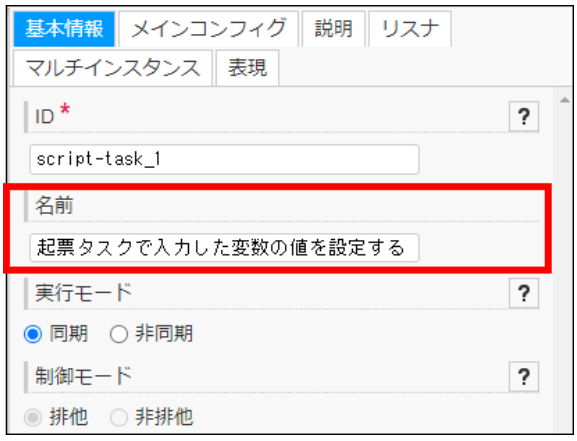
図：「起票タスク」 -「プロパティ 」-「メインコンフィグ」

1. 「案件名」に 備品申請 を入力します。
2. 「参照可能な変数を継承する」のチェックボックスを有効にします。
3. 「案件のユーザデータを格納する」のチェックボックスを有効にします。
4. 「案件のユーザデータを格納する変数名」 に draftTaskData を入力します。

図：「起票タスク」 -「プロパティ 」-「メインコンフィグ」

スクリプトタスクのプロパティを設定する

1. スクリプトタスクのプロパティで「基本情報」タブを開きます。
2. 「名前」に `起票タスクで入力した変数の値を設定する` を入力します。



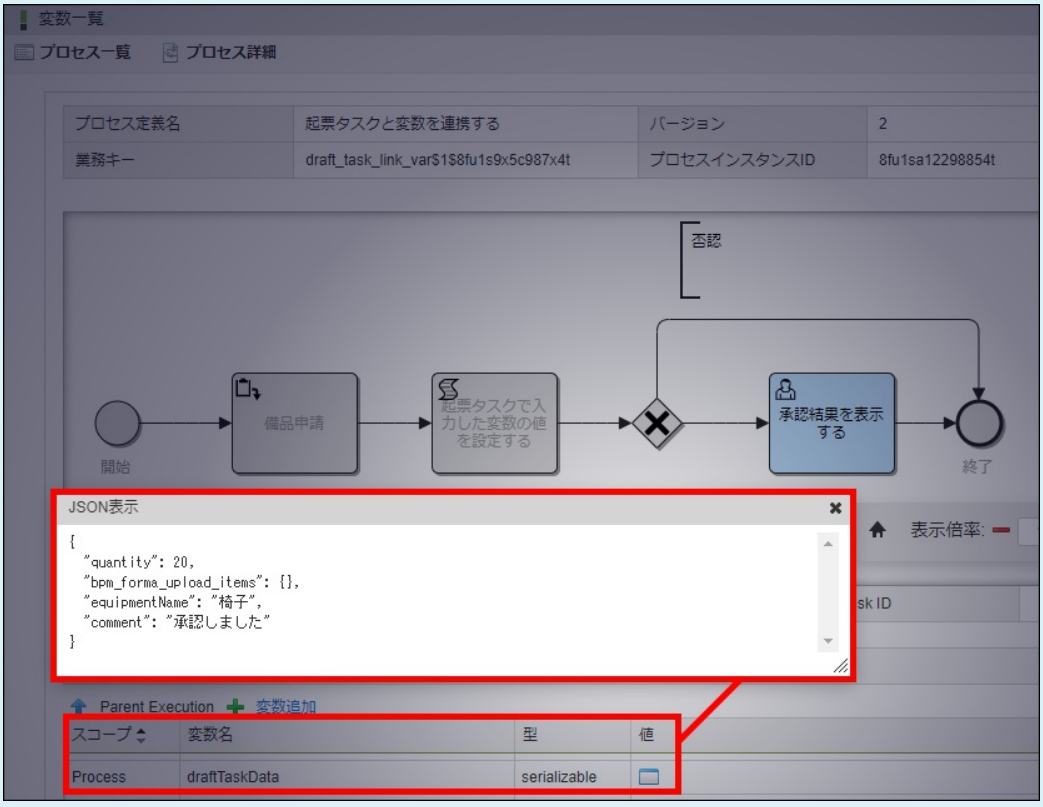
図：「スクリプトタスク」 -「プロパティ」 -「基本情報」

1. スクリプトタスクの「メインコンフィグ」タブを開きます。
2. 「メインコンフィグ」の「スクリプト」に設定する内容は、以下のとおりです。

```
function run(variables, execution, entity) {
  entity.setVariable('equipmentName', variables.draftTaskData.equipmentName);
  entity.setVariable('quantity', variables.draftTaskData.quantity);
  entity.setVariable('comment', variables.draftTaskData.comment);
  entity.setVariable('reference', 'true');
}
```

コラム

「案件のユーザデータを格納する変数名」を利用することによって、起票タスクに入力した値がフィールド識別IDをキーにして、プロセスインスタンスの変数にMap形式で格納されます。
IM-BPMではプロセスインスタンスの変数にフィールド識別IDと同じキーを設定することによって、Formaアプリケーションの値と連携できます。



図：「変数一覧」画面 - JSON表示
このチュートリアルでは、reference に true を設定することでFormaアプリケーションを参照で表示します。

ユーザタスクのプロパティを設定する

1. ユーザタスクのプロパティで「基本情報」タブを開きます。
2. 「名前」に 承認結果を表示する を入力します。

図：「ユーザタスク」 - 「プロパティ」 - 「基本情報」

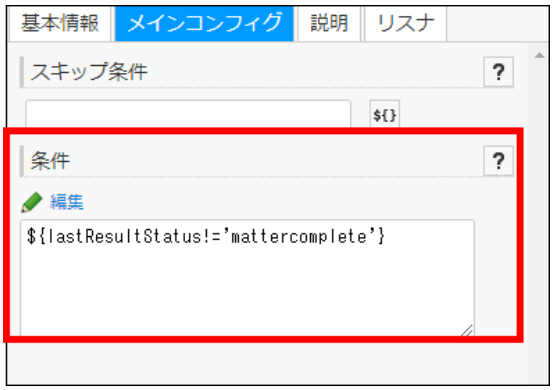
1. ユーザタスクのプロパティで「メインコンフィグ」タブを開きます。
2. 「担当者」に ${start_user} を入力します。
3. 「フォームキー」に forma:draft_task_link_var を入力します。

図：「ユーザタスク」 - 「プロパティ」 - 「メインコンフィグ」

排他ゲートウェイの分岐条件を設定する

1. 分岐の開始となる排他ゲートウェイに「デフォルトフロー」として sequence-flow_4 （ユーザタスク「承認結果を表示する」に接続されているシーケンスフロー）を設定します。
2. sequence-flow_6 （終了イベントに接続されているシーケンスフロー）の「メインコンフィグ」タブを開きます。
3. 「条件」に ${lastResultStatus!='mattercomplete'} を入力します。

図：「排他ゲートウェイ」 - 「プロパティ」 - 「基本情報」



図：「シーケンスフロー」 - 「プロパティ」 - 「メインコンフィグ」

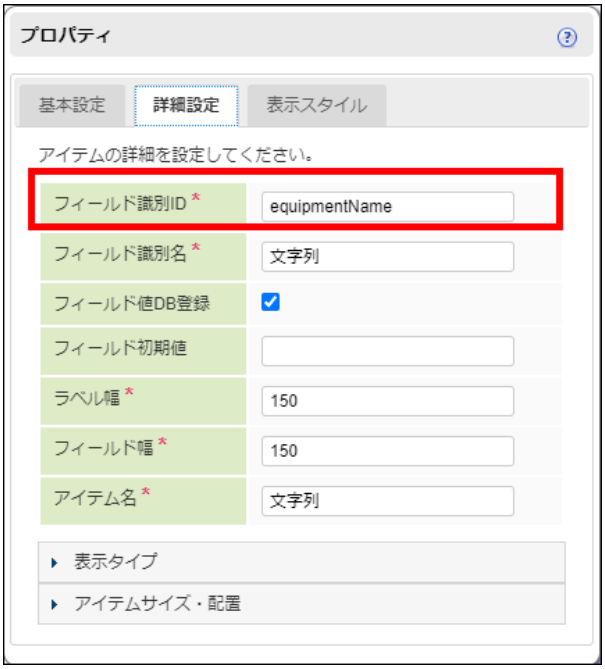
IM-BISのフィールド識別IDを確認する

1. 「サイトマップ」→「IM-BIS」→「システム管理者」→「IM-BIS作成」→「IM-BIS」から、「IM-BIS更新履歴」画面を表示します。
2. ヘッダメニューの「一覧」をクリックして、「IM-BIS一覧」画面を表示します。
3. インポートした「【チュートリアル】起票タスクと変数を連携する」の「編集」リンクをクリックし、「IM-BISフロー編集」画面を表示します。
4. 「申請」ノードの上で右クリックし、「開く」をクリックします。
5. 「フォーム編集」画面を表示します。

6. 「備品名」をクリックし、「プロパティ編集」をクリックします。

7. 「プロパティ」が表示されるので、「詳細設定」タブを開きます。

8. 「フィールド識別ID」がIM-BPMの変数と同一の `equipmentName` であることを確認してください。



図：「備品名」 - 「プロパティ」 - 「詳細設定」

9. 同様の手順で「個数」、「承認者用コメント欄」の「フィールド識別ID」も確認してください。

> **コラム**
>
> フィールド識別IDとIM-BPMの変数を同一のキーにすることによって、IM-BISのフォームのフィールドと連携できます。

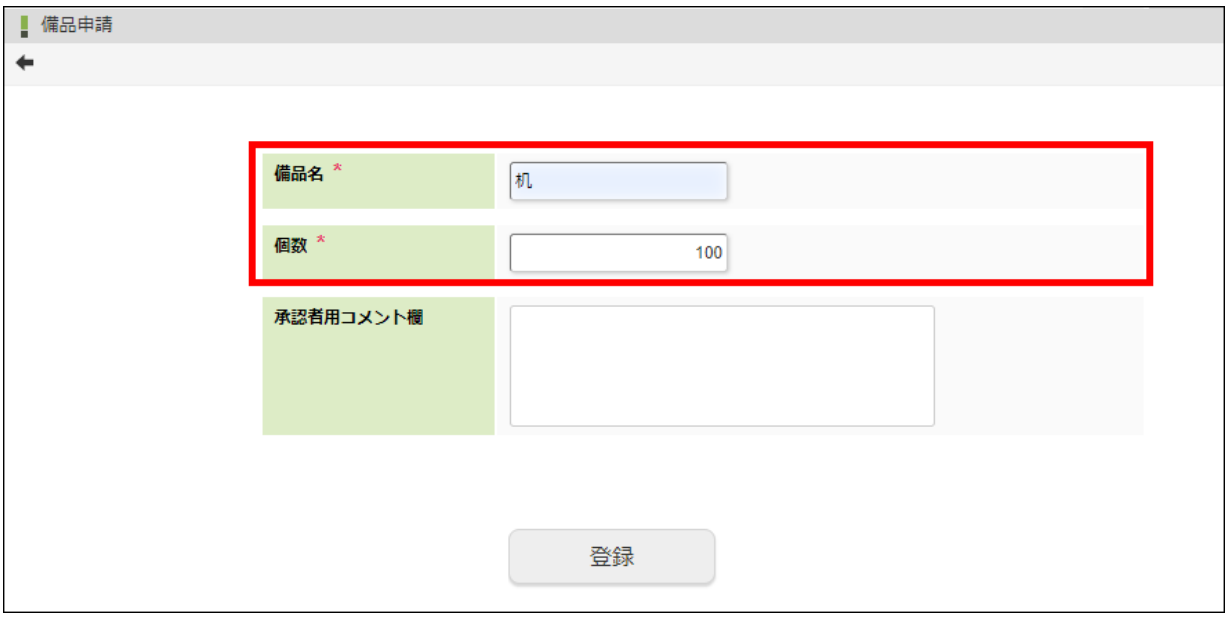
実行結果を確認する

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. ユーザ「青柳辰巳（ユーザコード：aoyagi）」でログインします。

2. 「プロセス開始一覧」画面にて当該プロセスを開始します。

3. Formaアプリケーションが表示されます。
   「備品名」と「個数」の項目に文字列と数値が入力できます。
   「承認者用コメント欄」の項目には入力できません。



図：「備品申請」画面

4. 「備品申請」画面にて、各入力フォームに値を入力して「登録」ボタンをクリックします。

5. 「サイトマップ」→「IM-BIS」→「一覧」→「案件一覧」画面にて、「備品申請」を検索し、起票した「備品申請」の「処理」をクリックします。
6. IM-BPMで入力した値を変更します。
7. 起票した「備品申請」を申請します。
8. ユーザ「青柳辰巳（ユーザコード：aoyagi）」からログアウトします。
9. ユーザ「上田辰男（ユーザコード：ueda）」でログインします。
10. 「サイトマップ」→「IM-BIS」→「一覧」→「案件一覧」画面を表示します。
11. 「備品申請」を検索し、申請された「備品申請」の「処理」をクリックします。
12. 「承認者用コメント欄」に値を入力します。
13. 「備品申請」を承認します。
14. ユーザ「上田辰男（ユーザコード：ueda）」からログアウトします。
15. ユーザ「青柳辰巳（ユーザコード：aoyagi）」でログインします。
16. 「タスク一覧」画面にて、「承認結果を表示する」タスクを検索し、「処理」をクリックします。
17. Formaアプリケーションが表示されます。
    IM-BISで入力、および、編集したすべての項目を参照できます。

図：「備品申請」画面

18. 「確認」ボタンをクリックすることで、当該プロセスは終了します。

> コラム
> - プロセスのデプロイ方法については「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義/ケース定義のデプロイ」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。

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

開始イベントのフォームにFormaアプリケーションを利用する

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

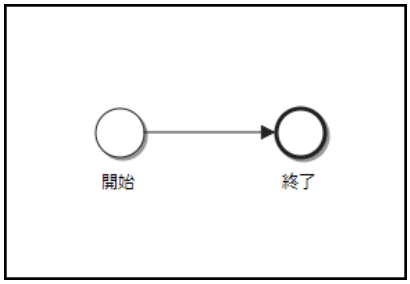
コラム

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

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

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

エレメントを配置する



図：配置イメージ

1. 開始イベント、終了イベントを配置してシーケンスフローで接続します。

開始イベントのプロパティを設定する

図：プロパティ入力イメージ

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

コラム

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

コラム

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

実行結果を確認する

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

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

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

図：「変数一覧」画面

コラム

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

ユーザタスクのフォームにFormaアプリケーションを利用する

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

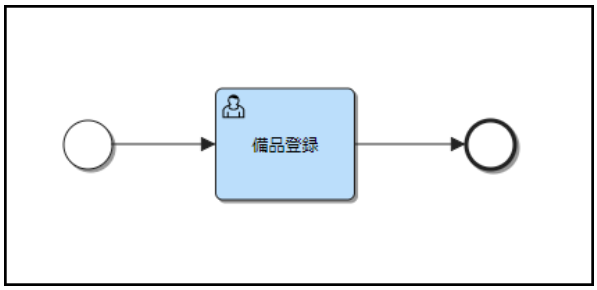
コラム

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

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

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

エレメントを配置する



図：配置イメージ

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

ユーザタスクのプロパティを設定する

図：プロパティ入力イメージ

1. ユーザタスクのプロパティで「メインコンフィグ」タブを開きます。
2. 「処理対象グループ」に im_bpm_user を入力します。
3. 「フォームキー」に forma:sample_app_equipment を入力します。

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

Formaアプリケーションに受け渡す変数を定義する

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

図：プロセス：プロパティ - データオブジェクト

1. プロセスグローバルのプロパティで、「データオブジェクト」タブを開きます。
2. 「データプロパティ」に equipment_name="コピー用紙"、maker="intra-mart製紙" を定義します。

> **コラム**
>
> プロセスグローバルのデータ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。

図：「備品登録」画面

3. 「備品登録」画面にて、各入力フォームに値を設定して登録します。

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

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

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

図：「変数一覧」画面

コラム

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

発展編

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

- ワークフローとの連携

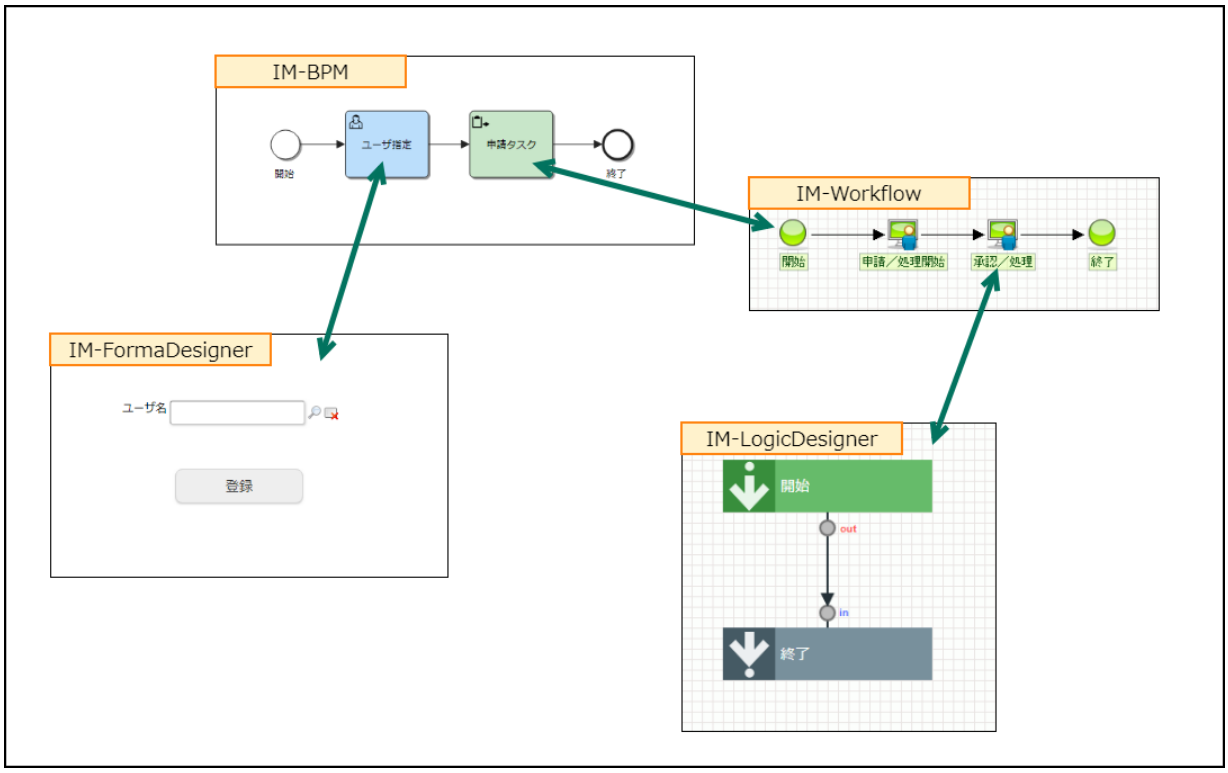
ワークフローとの連携

承認ノードの処理対象者をロジックフローで指定する

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

チュートリアルで作成するものは以下の4つです。

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



図：概要

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

コラム

このチュートリアルで作成する各種定義のサンプルを以下のリンクからダウンロードできます。

- プロセス定義
  approve_target_logicdesigner.bpmn
- IM-FormaDesignerアプリケーション情報
  app_select_target.zip
- IM-LogicDesigner ロジックフロー
  im_logicdesigner-data-workflow_target_plugin.zip
- IM-Workflow ロジックフロー管理テーブルデータ
  imw_m_logic_flow.csv
- IM-Workflow ルート定義およびフロー定義
  route_flow_target_ld.xml

これらのサンプルは、各アプリケーションの機能でインポート、または、アップロードして利用可能です。
詳細な手順については、以下のリンク先を参照してください。

- 「IM-BPM プロセスデザイナ 操作ガイド」-「プロセス定義のアップロード」
- 「IM-FormaDesigner 作成者操作ガイド」-「インポート・エクスポートを利用した IM-FormaDesigner のアプリケーションやデータソース定義の移行」
- 「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」
- 「TableMaintenance 管理者操作ガイド」-「テーブル・インポート」
- 「IM-Workflow 管理者操作ガイド」-「インポート/エクスポートを行う」

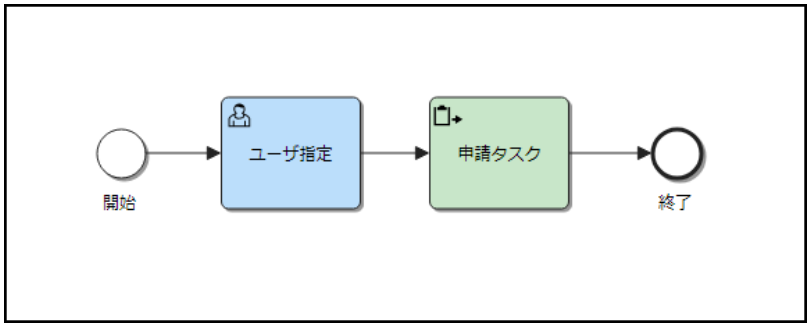
インポート、または、アップロードは、上記の順番で実行してください。
ロジックフローのインポートよりも前に、ロジックフロー管理テーブルデータやルート定義のインポートを行った場合、正しくデータが連携しません。

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

プロセス定義を作成する

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

エレメントを配置する



図：完成イメージ

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

ユーザタスクのプロパティを設定する

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

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

申請タスクのプロパティを設定する

図：「申請タスク」 -「プロパティ」 -「メインコンフィグ」

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

Formaアプリケーションを作成する

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

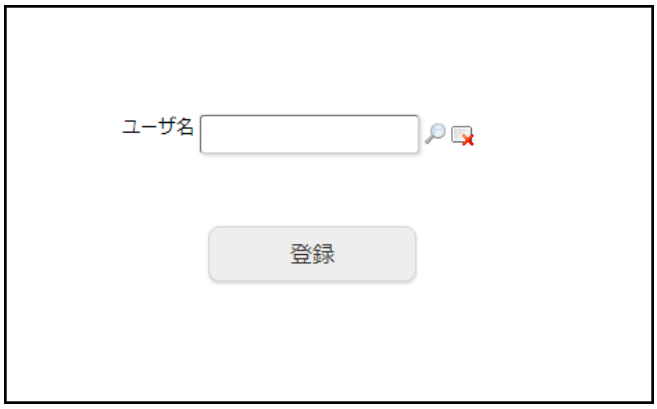
アプリケーションの基本情報を入力する

図：「アプリケーション登録」

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

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

フォームを編集する



図：完成イメージ（フォーム）

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

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

テーブルを登録する

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

テーブル名: imfr_ut_app_select_target

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」

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

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

権限を設定する

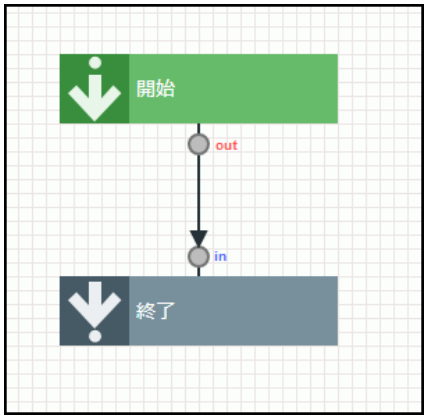
1. 「サイトマップ」→「Forma管理画面」→「アプリ一覧」から「アプリケーション一覧」画面を表示します。
2. アプリケーションID「 app_select_target 」の編集アイコンをクリックします。
3. 「権限設定」タブをクリックします。
4. 「+ロール」をクリックします。
5. 「追加」をクリックします。
6. 「ロール検索」ダイアログにて、「 IM-BPMユーザ（im_bpm_user）」ロールを検索して選択し、「決定」をクリックします。
7. 追加された「IM-BPMユーザ（im_bpm_user）」ロールの「権限」セレクトボックスにて、「登録・更新・削除可能」を選択しま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

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。

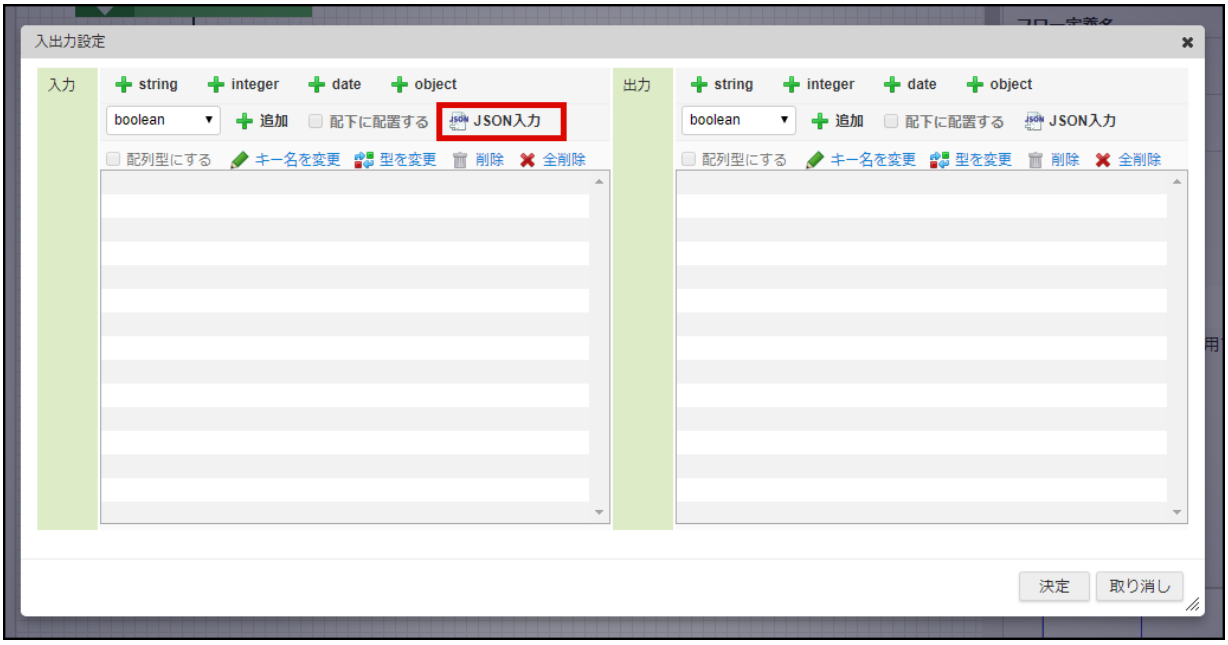
「サイトマップ」→「LogicDesigner」→「フロー定義一覧」から「ロジックフロー定義一覧」を表示し、「ロジックフロー新規作成」をクリックしてロジックフローの作成を開始します。



図：完成イメージ（ロジックフロー）

入出力設定を行う

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

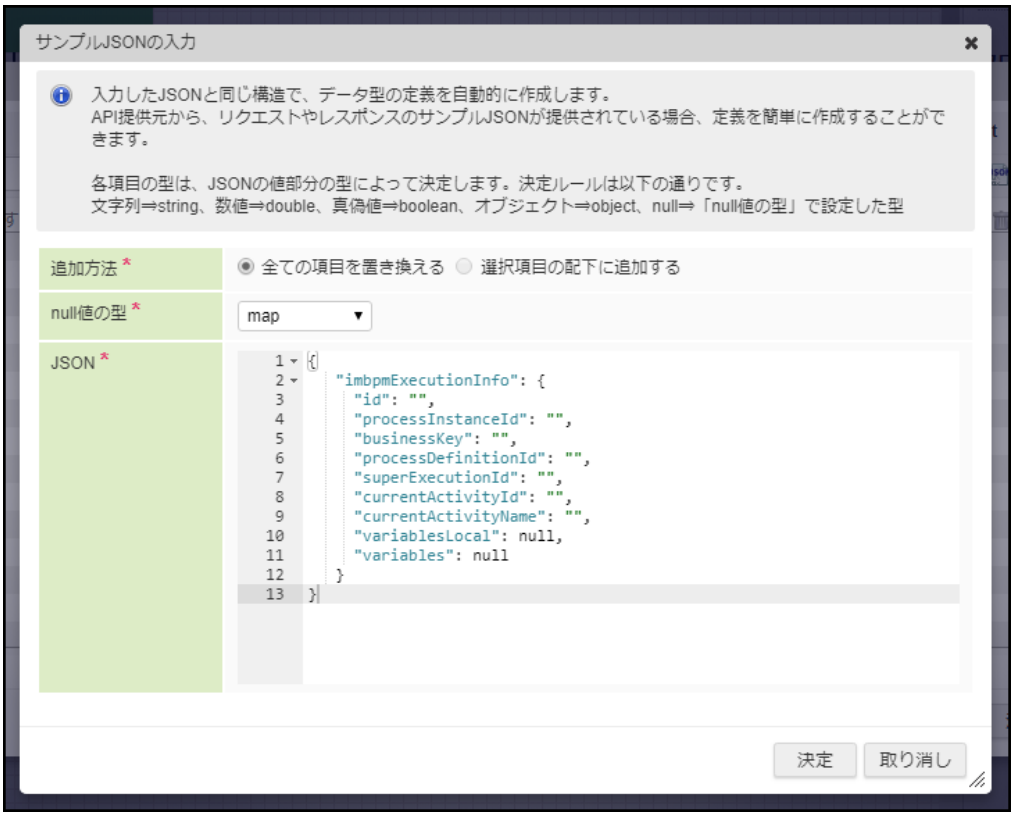


図：「入出力設定」 - 「JSON入力」

3. 「null値の型」にて map を選択します。

4. 「JSON」の内容を、以下の文字列で置き換えます。

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



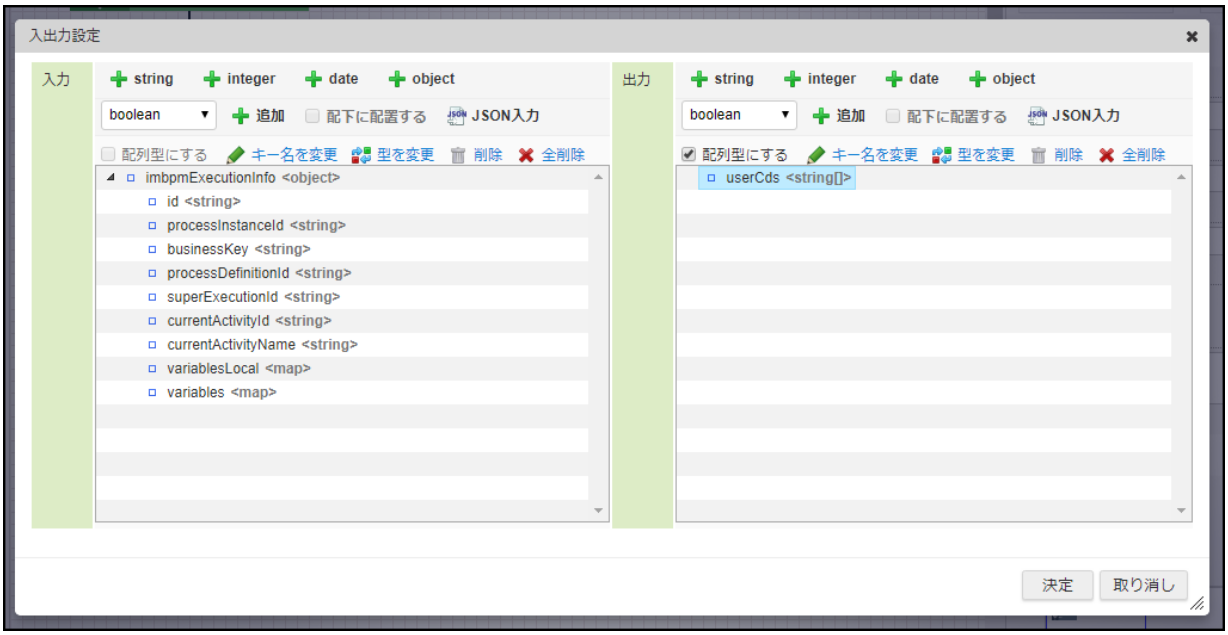
図：「サンプルJSONの入力」

5. 「決定」をクリックします。

6. 「出力」ペイン上部にある、「+string」をクリックします。

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

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



図：「入出力設定」（設定後）

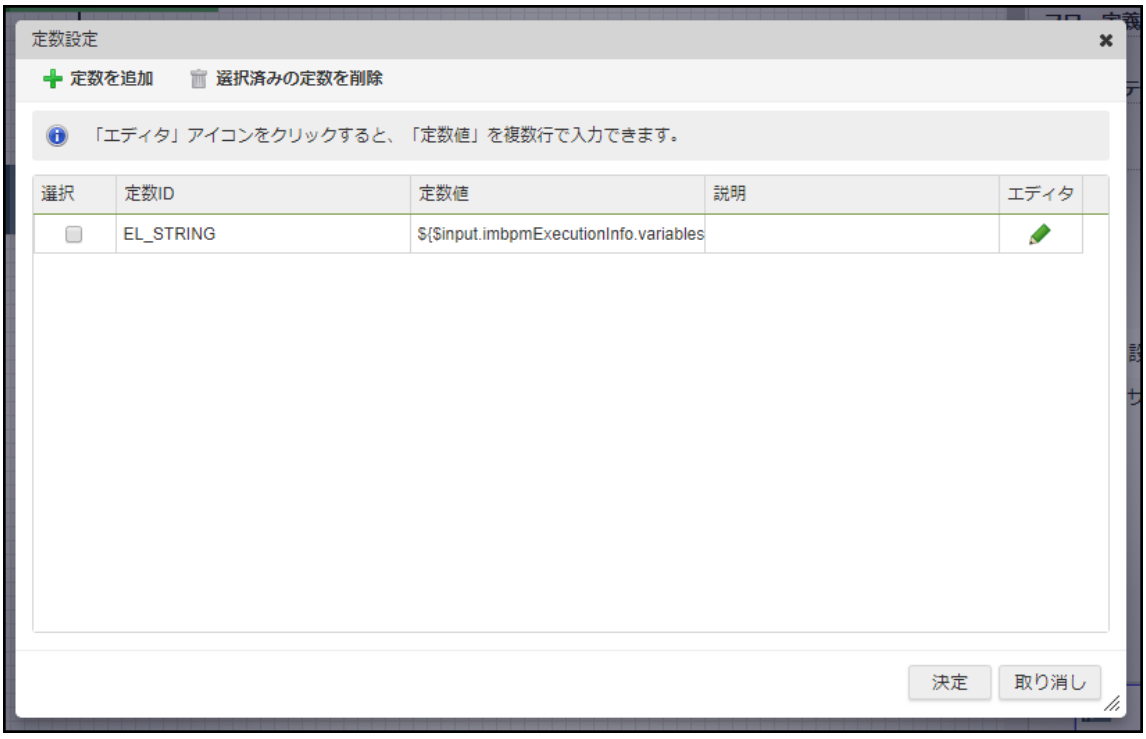
9. 「決定」をクリックします。

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

定数設定を行う

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

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



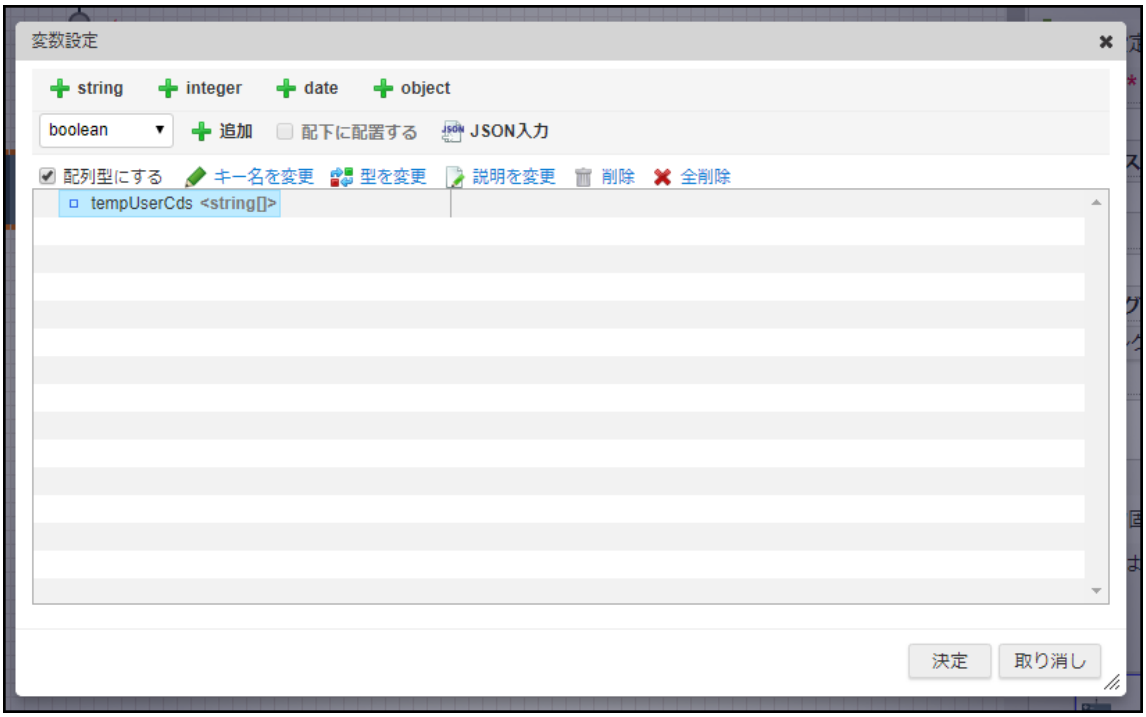
図：「定数設定」（設定後）

5. 「決定」をクリックします。

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

変数設定を行う

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



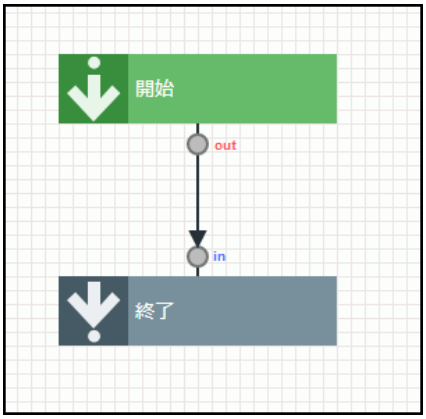
図：「変数設定」（設定後）

5. 「決定」をクリックします。

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

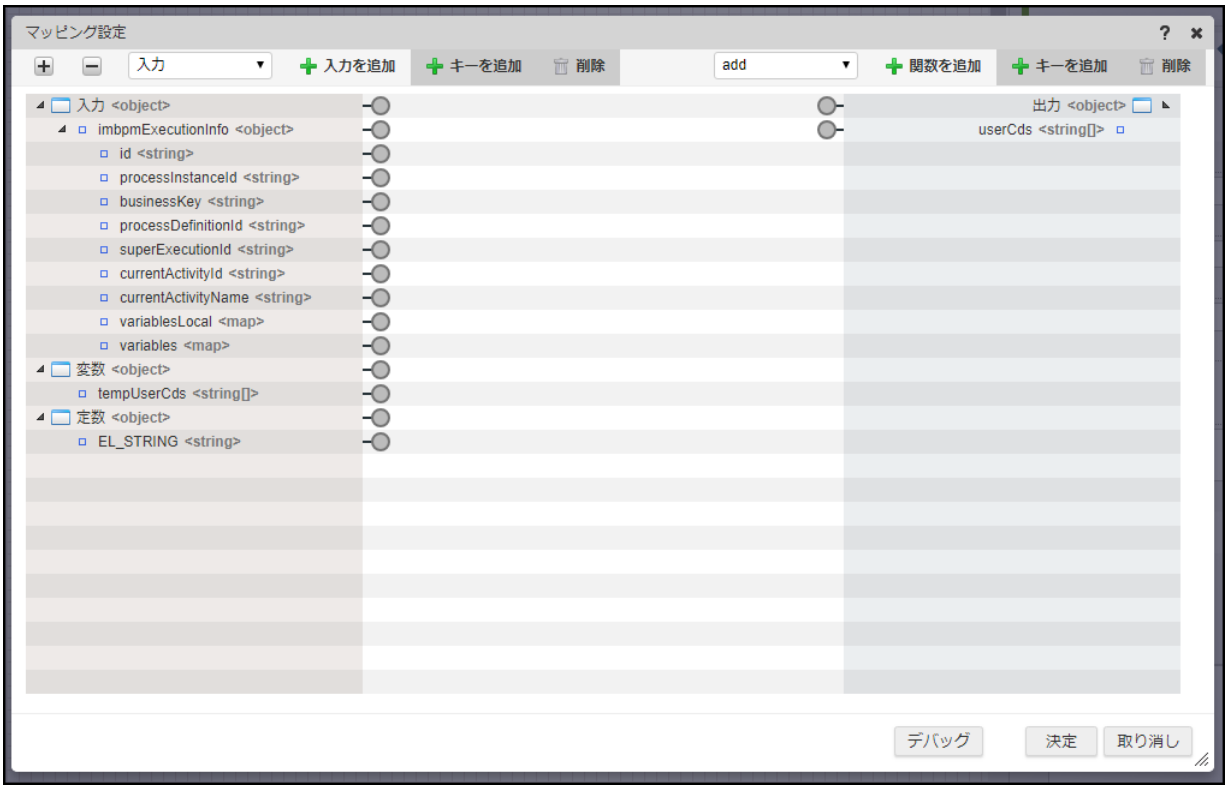
エレメントのマッピングを行う

1. 「開始」エレメントから「終了」エレメントに向けて、線を接続します。



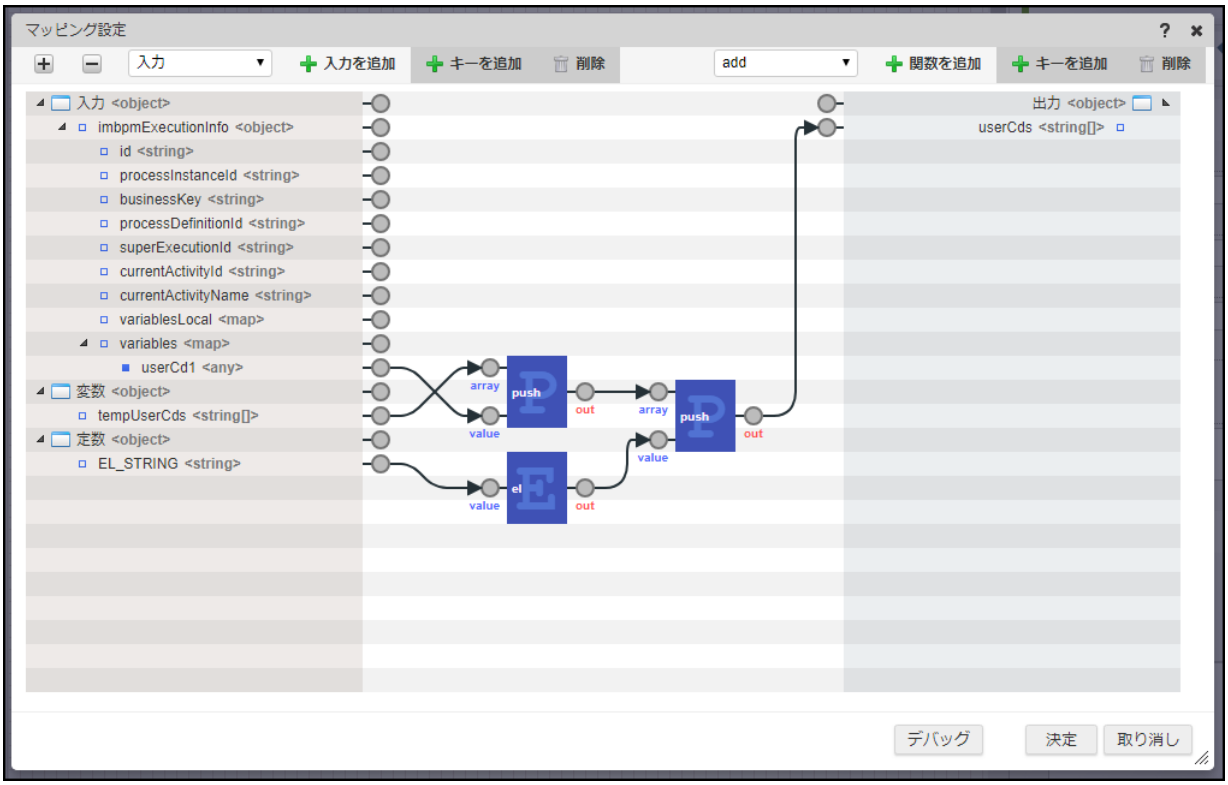
図：エレメントの接続

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



図：「終了」エレメント - 「マッピング設定」

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



図：「終了」エレメント - 「マッピング設定」（設定後）

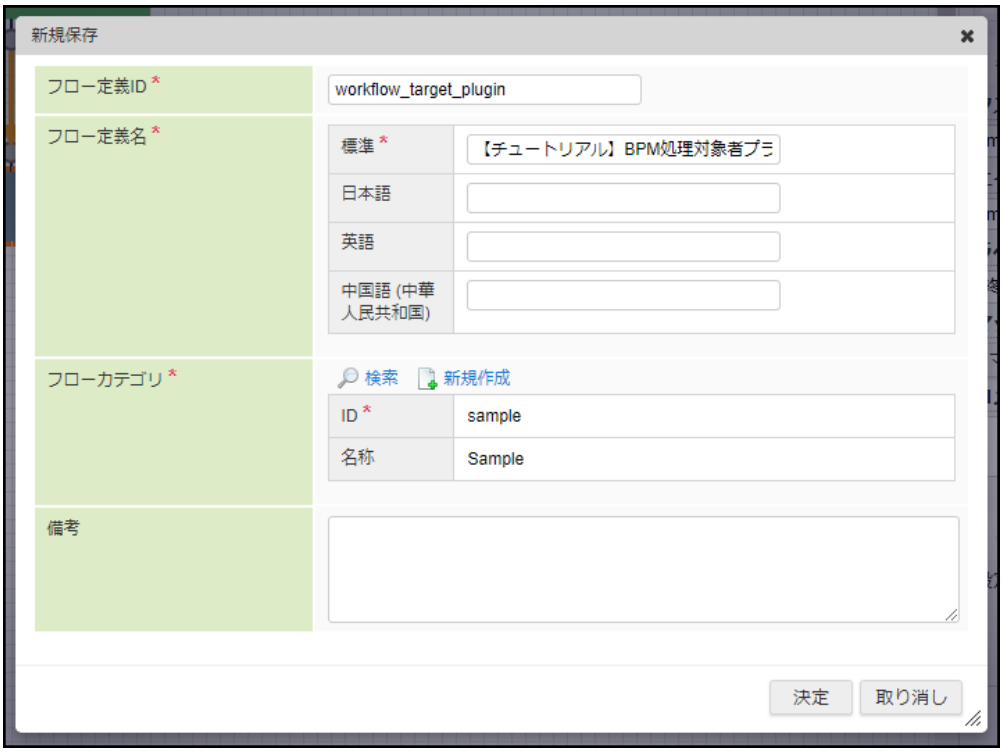
14. 「決定」をクリックします。

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

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

新規保存を行う

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



図：「新規保存」

5. 「決定」をクリックします。

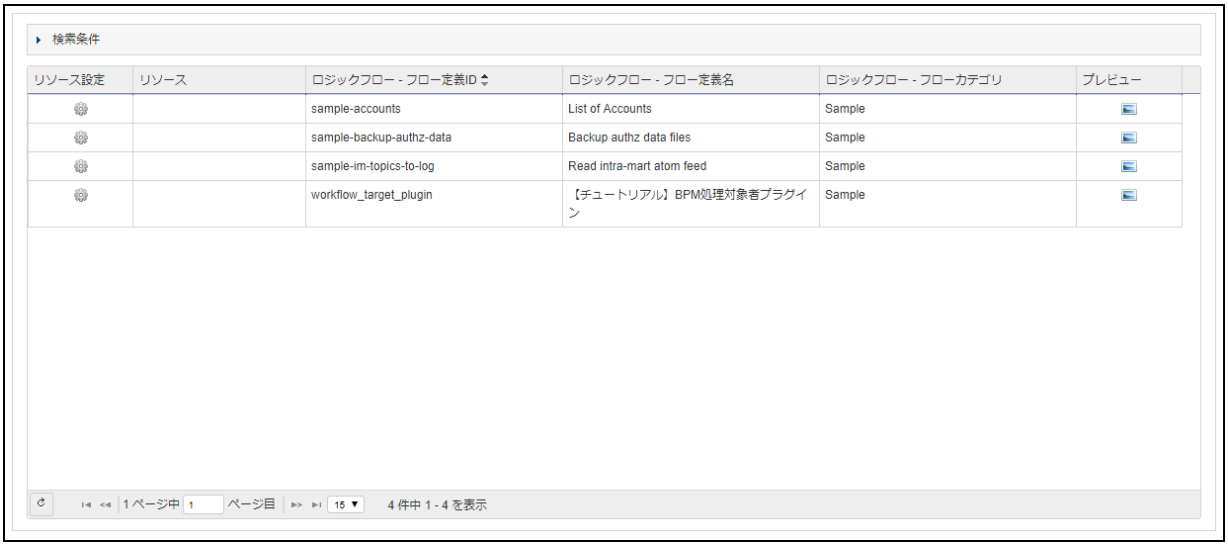
> **コラム**
>
> ここで設定した「フロー定義名」は、ワークフローの「履歴参照」画面などに処理者として表示されます。

ワークフローのルート定義・フロー定義を作成する

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

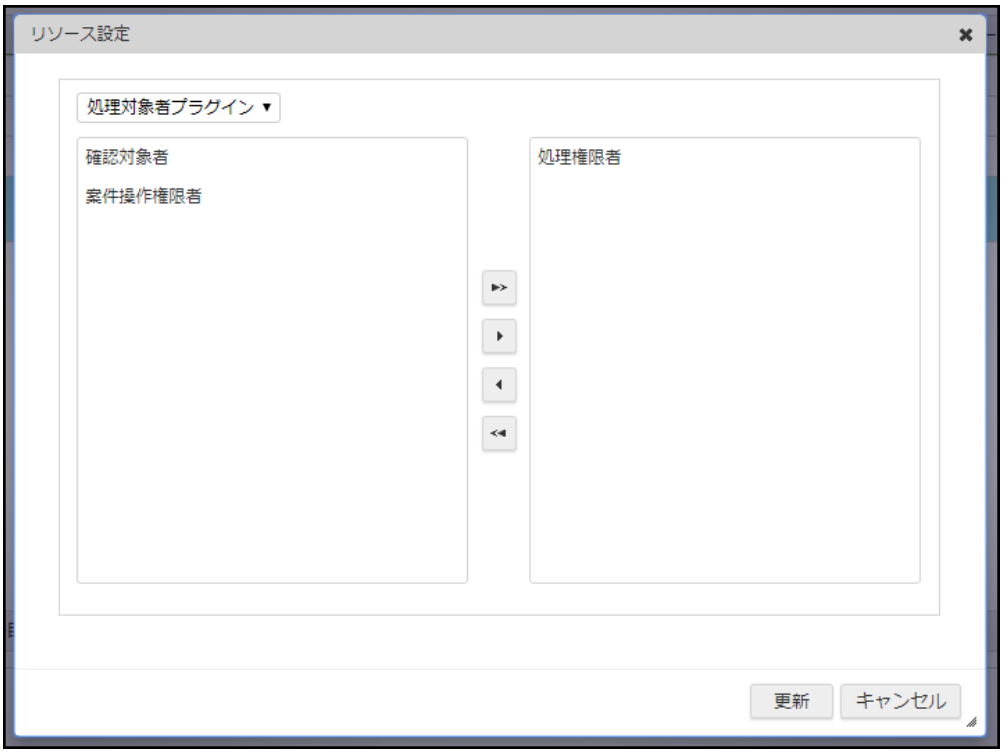
ロジックフローを管理する

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



図：「ロジックフロー管理」

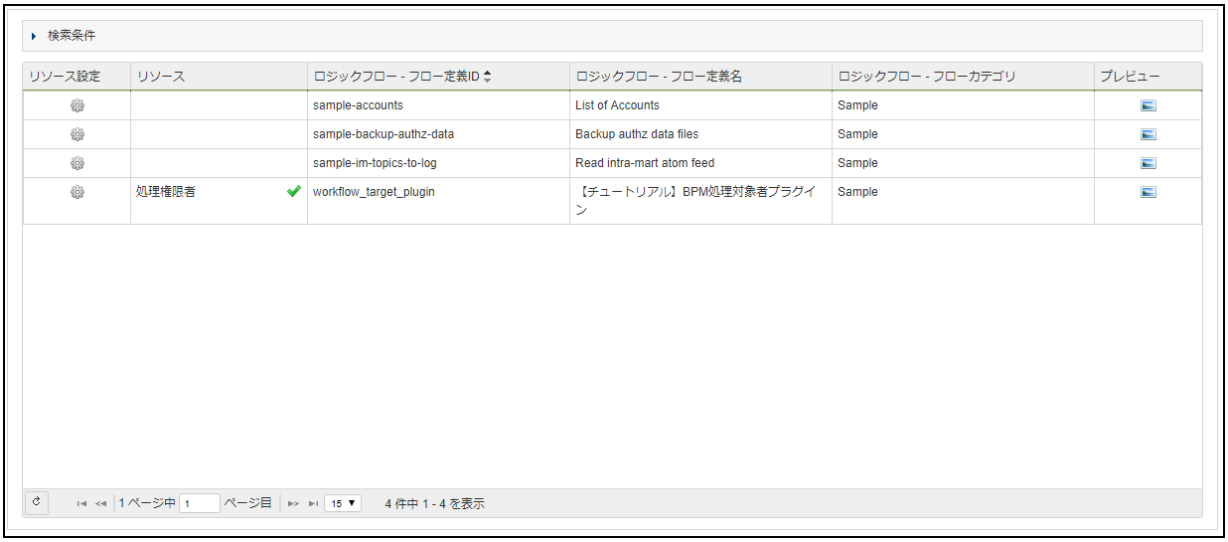
2. 作成した「【チュートリアル】BPM処理対象者プラグイン」について、「 」アイコンをクリックします。
3. 「リソース設定」ダイアログにて、セレクトボックスで「処理対象者プラグイン」を選択します。
4. 左ペインに表示されている「処理権限者」をクリックして選択します。
5. 中央に表示されている をクリックし、「処理権限者」を右ペインに移動します。



図：「リソース設定」

6. 「更新」をクリックします。

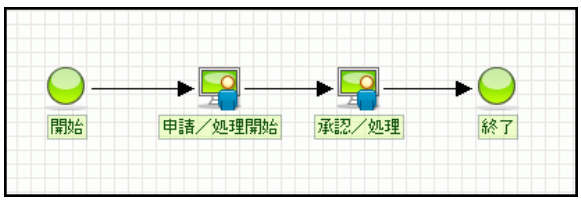
7. 「【チュートリアル】BPM処理対象者プラグイン」が、処理対象者プラグイン（処理権限者）として利用可能になりました。



図：「ロジックフロー管理」（設定後）

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

ルート定義を作成する



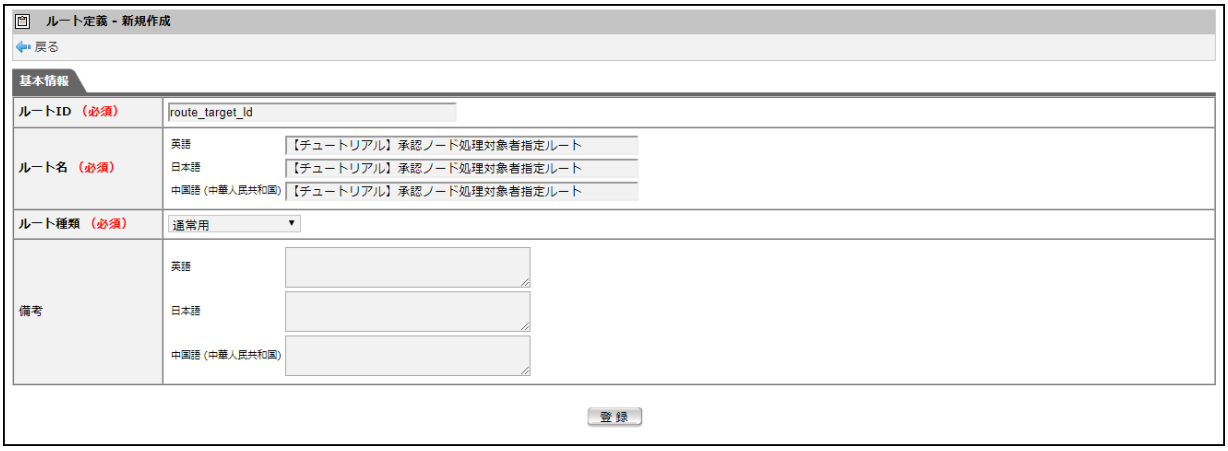
図：完成イメージ（ルート定義）

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

2. 「新規作成」をクリックします。

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

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



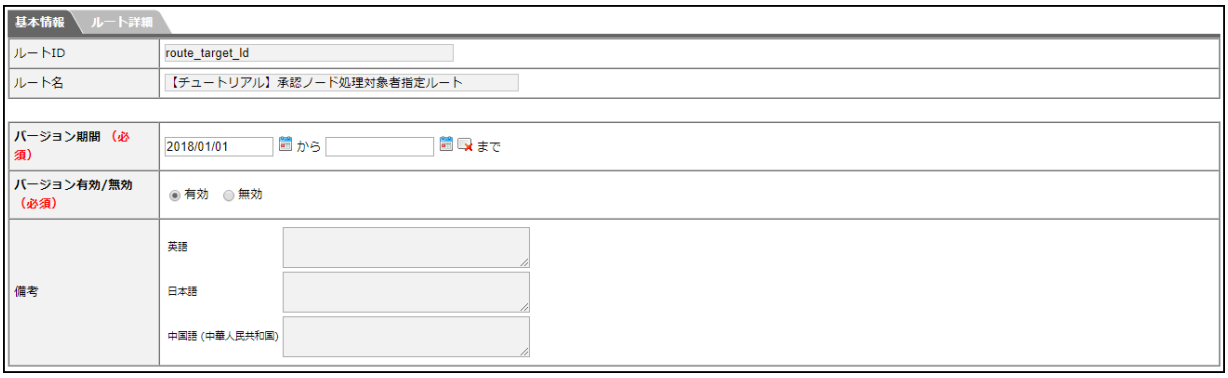
図：「ルート定義 - 新規作成」

5. 「登録」をクリックします。

6. 表示されている「バージョン」タブ画面にて、「新規作成」をクリックします。

7. 「基本情報」タブ画面にて、「バージョン期間」 の範囲開始日に 2018/01/01 を入力します。

8. 「バージョン有効/無効」にて、「 有効 」を選択します。



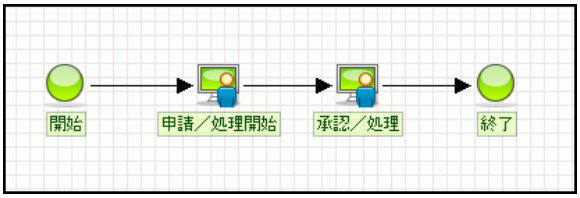
図：「ルート定義 - バージョン - 新規作成」 - 「基本情報」

9. 「ルート詳細」タブをクリックします。

10. 「承認／処理」ノードをドラッグし、ドロップして配置します。

11. 「申請／処理開始」ノードから「承認／処理」ノードに線を引きます。

12. 同様に、「承認／処理」ノードから「終了」ノードに線を引きます。

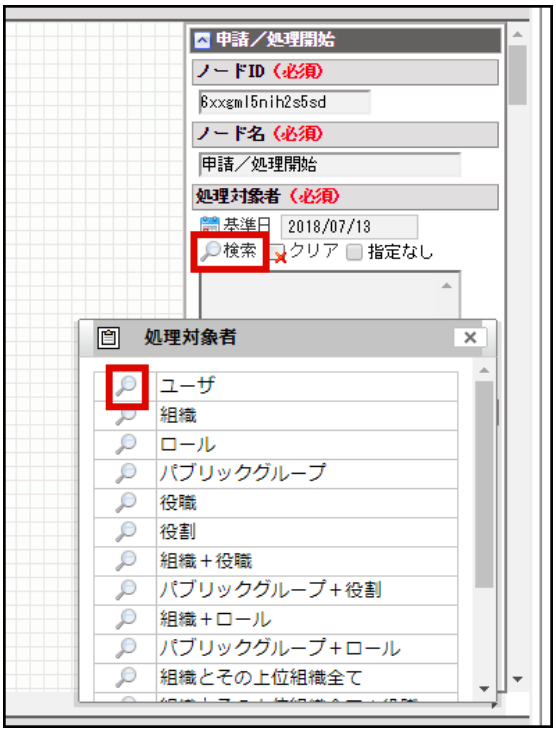


図：「ルート定義 - バージョン - 新規作成」 - 「ルート詳細」

13. 「申請／処理開始」ノードをクリックし、選択します。

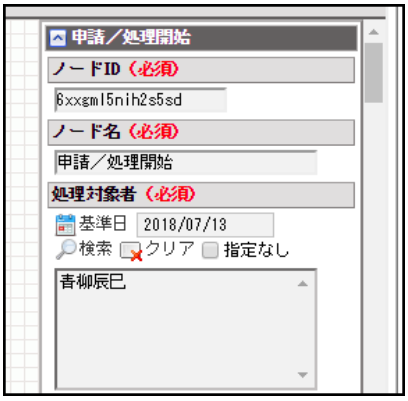
14. 右側に表示されたノード詳細エリアにて、「処理対象者」の「検索」をクリックします。

15. 表示された「処理対象者」一覧にて、「ユーザ」を選択し、虫眼鏡アイコンをクリックします。



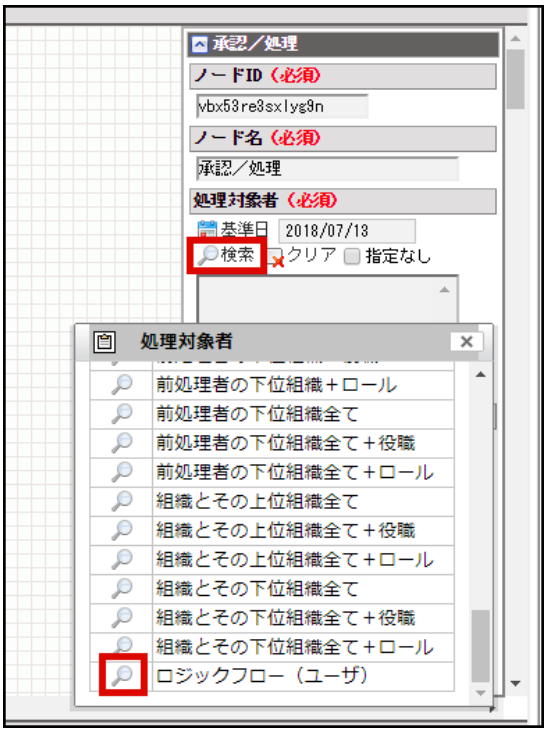
図：「ルート定義 - バージョン - 新規作成」 - 「申請／処理開始」ノード詳細

16. 表示された「ユーザ検索」ダイアログにて、「青柳辰巳」を検索して選択し、「決定」をクリックします。
17. 右側に表示されたノード詳細エリアにて、「処理対象者」に「青柳辰巳」が設定されているのを確認します。



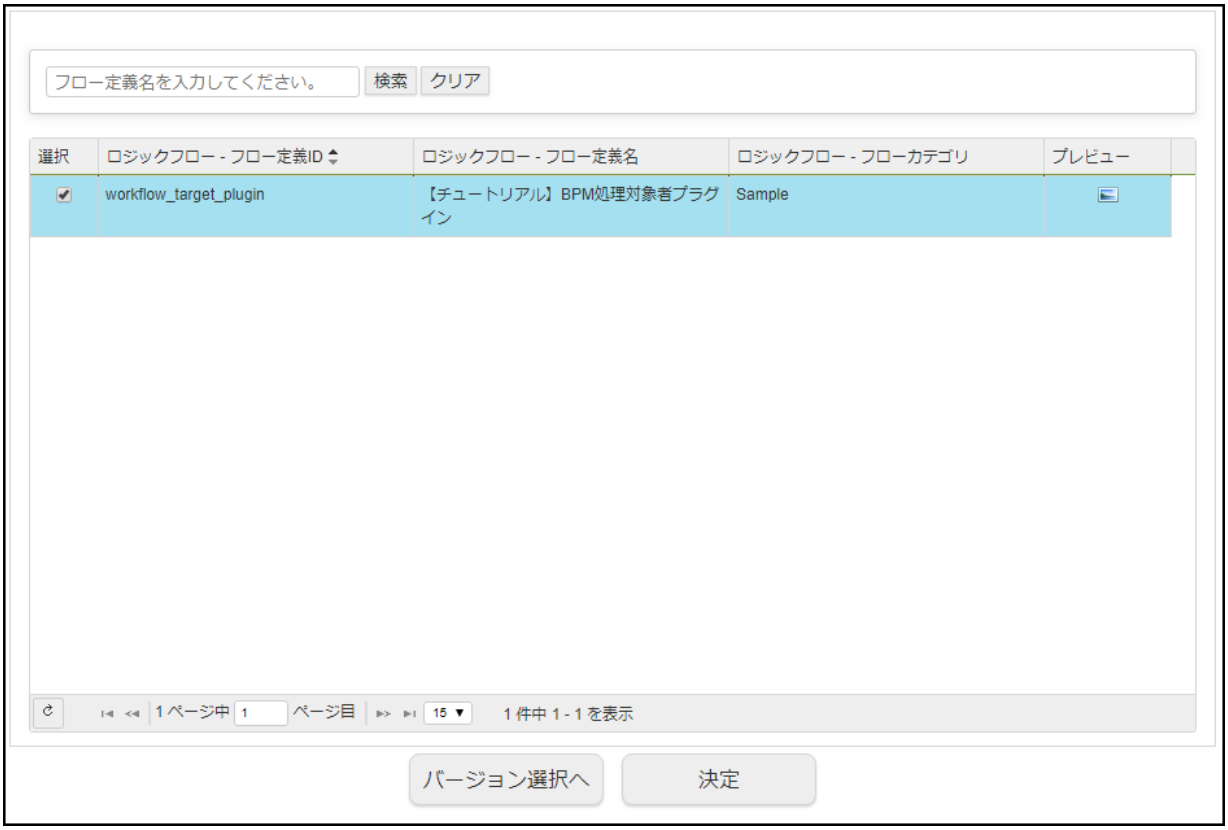
図：「ルート定義 - バージョン - 新規作成」 - 「申請／処理開始」ノード詳細（設定後）

18. 「承認／処理」ノードをクリックし、選択します。
19. 右側に表示されたノード詳細エリアにて、「処理対象者」の「検索」をクリックします。
20. 表示された「処理対象者」一覧にて、「ロジックフロー（ユーザ）」を選択し、虫眼鏡アイコンをクリックします。



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

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



図：「ロジックフロー検索」選択イメージ

22. 「決定」をクリックします。

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

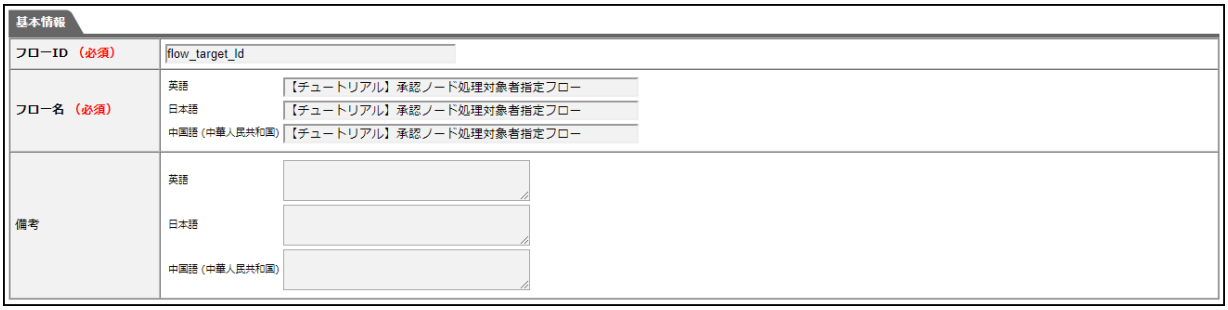
図：「ルート定義 - バージョン - 新規作成」 - 「承認／処理」ノード詳細（設定後）

24. 「登録」をクリックします。

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

フロー定義を作成する

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



図：「フロー定義 - 新規作成」

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

図：「フロー定義 - バージョン - 新規作成」 - 「基本情報」

11. 「登録」をクリックします。

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

実行結果を確認する

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

プロセスを開始し、ユーザタスクを処理する

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

図：「個人タスク」 - 「ユーザ指定」タスク - 「処理」

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

図：「【チュートリアル】処理対象者選択」入力画面

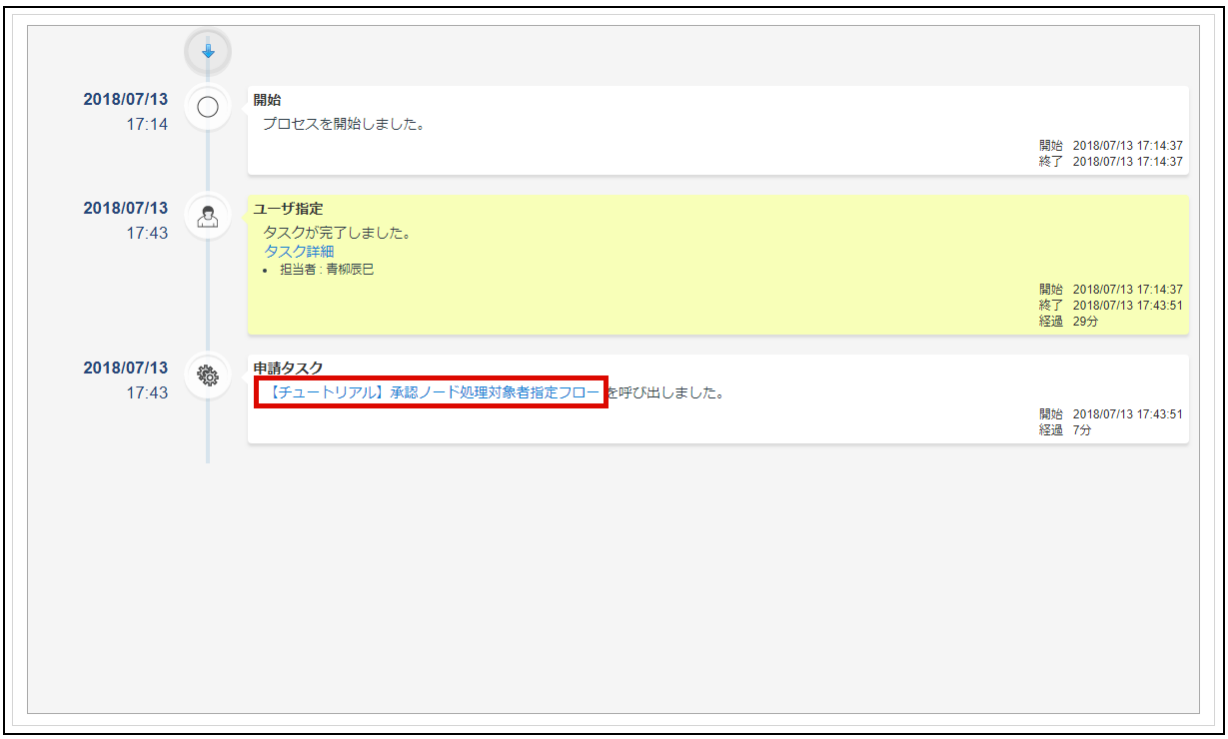
7. 「登録」をクリックします。

ワークフローの処理対象者を確認する

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

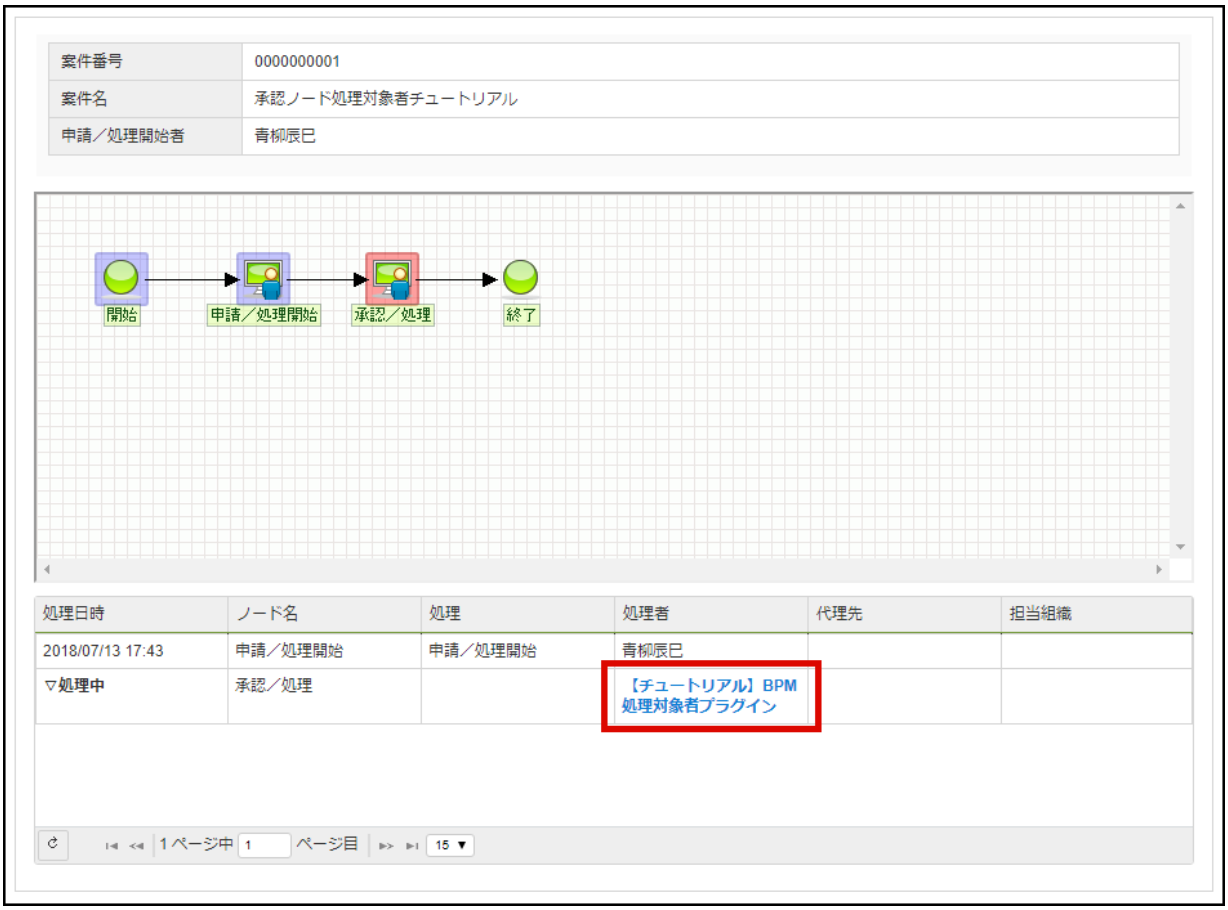
図：「処理済タスク」 - 「ユーザ指定」タスク - 「履歴」

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



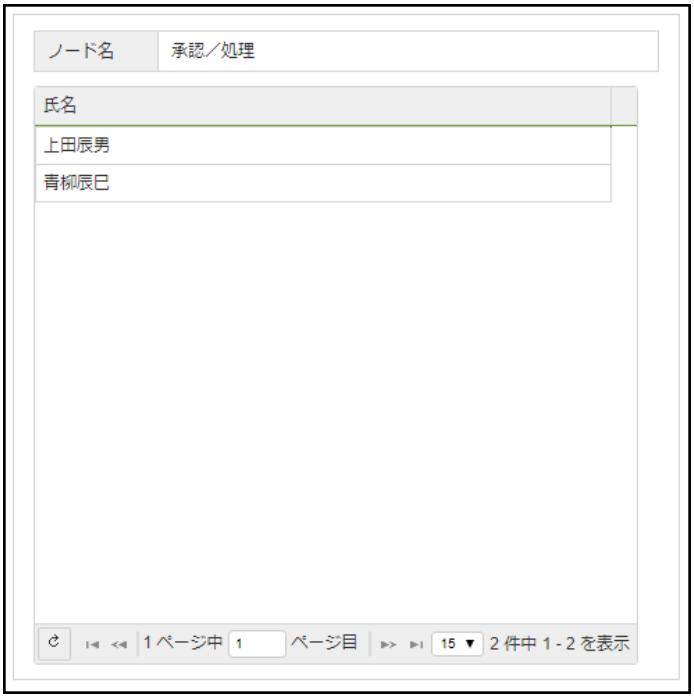
図：「履歴」 - 「申請」タスク - ワークフロー「履歴参照」へのリンク

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



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

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



図：「処理対象者状況確認」

本番環境への適用

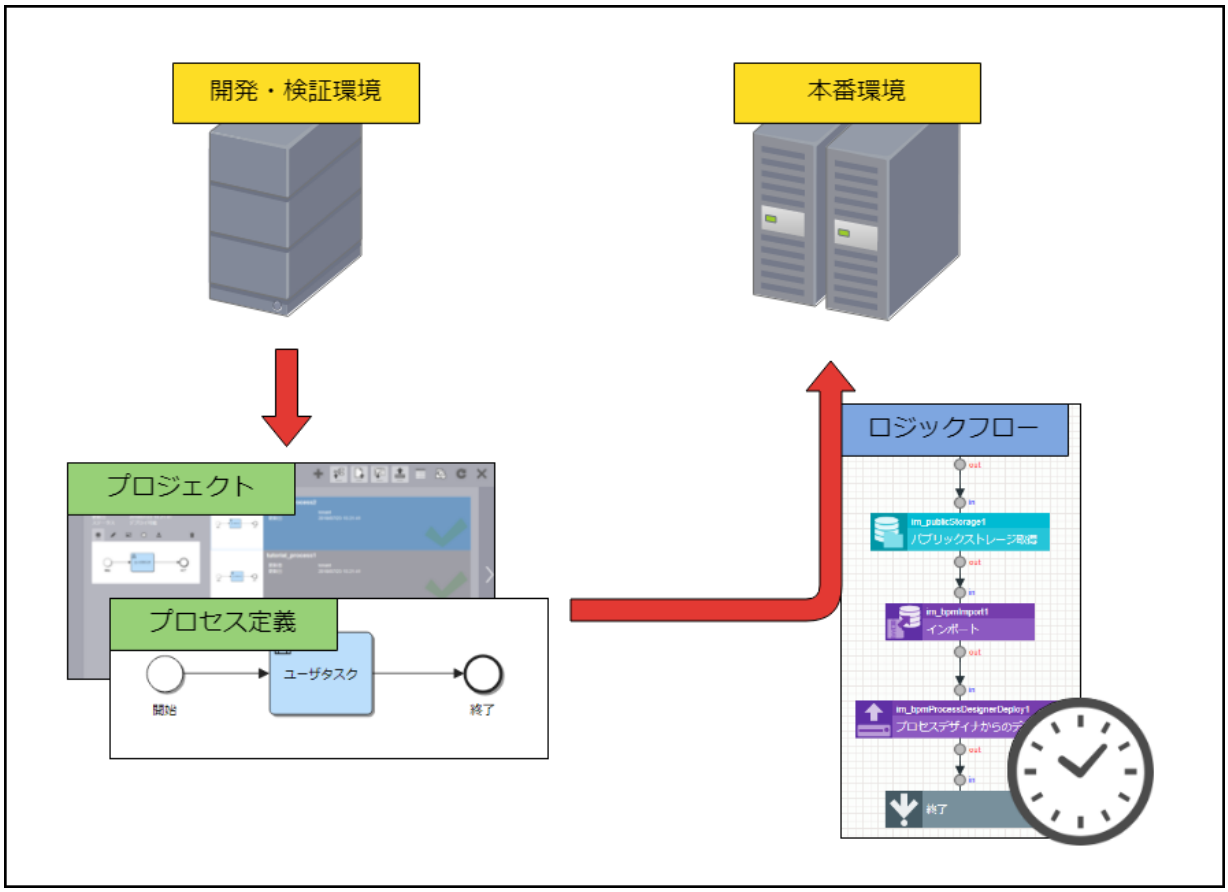
プロセス定義を本番環境へ適用するプロセスの管理方法について説明します。
このチュートリアルは「*事前準備*」が完了していることを前提としています。

プロセス定義のデプロイをスケジューリングする

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

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

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



図：概要

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

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

コラム

このチュートリアルで作成するロジックフロー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を作成する
- ロジックフローを実行するジョブネットを作成する
- 実行結果を確認する

プロセスデザイナのプロジェクトをエクスポートする

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

コラム

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

プロセスデザイナのプロジェクトをパブリックストレージに配置する

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

1. 「サイトマップ」→「テナント管理」→「ファイル操作」から、「ファイル操作」画面を表示します。
2. 「ファイル操作」画面、左側のツリーからパブリックストレージの直下を選択します。

図：「ファイル操作」

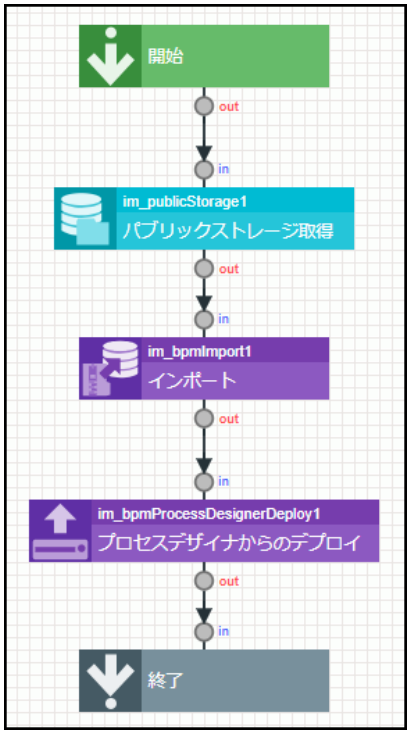
3. 「詳細」 -「ファイルアップロード」 -「ファイル追加」をクリックします。

このチュートリアルでエクスポートしたプロセスデザイナのプロジェクトを選択します。
サンプルデータを使用する場合は、 process_auto_deploy.zip を選択します。

図：「ファイル操作」 process_auto_deploy.zip

プロセスデザイナのプロジェクトのインポートとプロセス定義のデプロイを行うロジックフローを作成する

プロセスデザイナのプロジェクトをプロセスデザイナへインポートし、プロセスデザイナから IM-BPM Runtime へプロセス定義のデプロイを行うロジックフローを作成します。



図：完成イメージ（ロジックフロー）

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「ロジックフロー新規作成」をクリックし、「ロジックフロー定義編集」画面を表示します。
3. 「ロジックフロー定義編集」画面上部、ヘッダ内の「入出力設定」をクリックします。

   入力値を以下のように設定します。

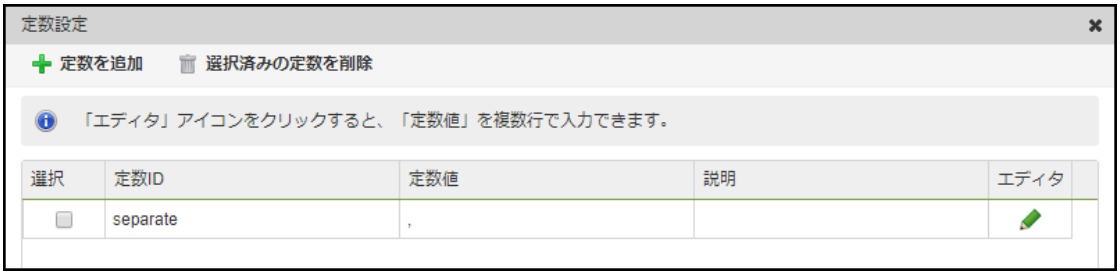
キー名	型
deploymentName	<string>
processResourceNameList	<string>
projectId	<string>
zipFilePath	<string>

図：「入出力設定」

4. 「ロジックフロー定義編集」画面上部、ヘッダ内の「定数設定」をクリックします。

   定数値を以下のように設定します。

定数ID	定数値
separate	,



図：「定数設定」

5. 「パブリックストレージ取得」を配置します。

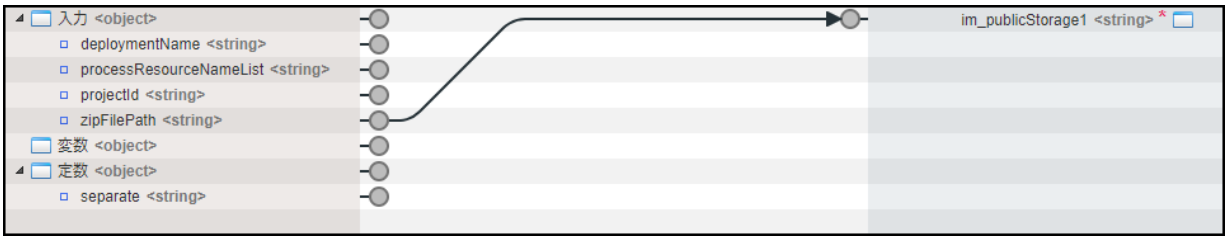
   パレット内の「ストレージ操作」の一覧から「パブリックストレージ取得」を選び、フロー編集画面上に追加します。

図：「パブリックストレージ取得」

6. 「パブリックストレージ取得」のマッピング設定をします。

   以下のように線を引きます。

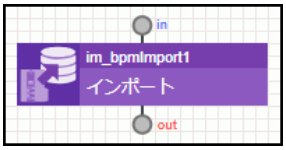
入力（始点）	出力（終点）
入力<object> - zipFilePath<string>	im_publicStorage1<string>



図：「マッピング設定」 - 「パブリックストレージ取得」

7. 「インポート」を配置します。

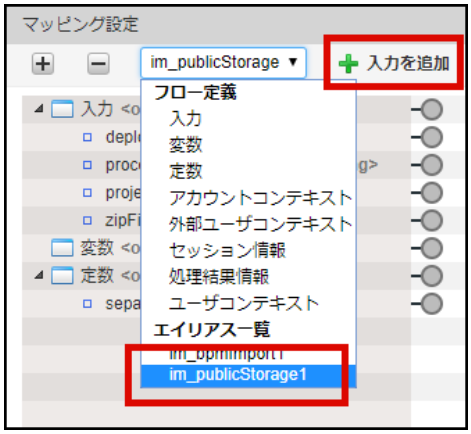
   パレット内の「IM-BPM」の一覧から「インポート」を選び、フロー編集画面上に追加します。



図：「インポート」

8. 「インポート」のマッピング設定をします。

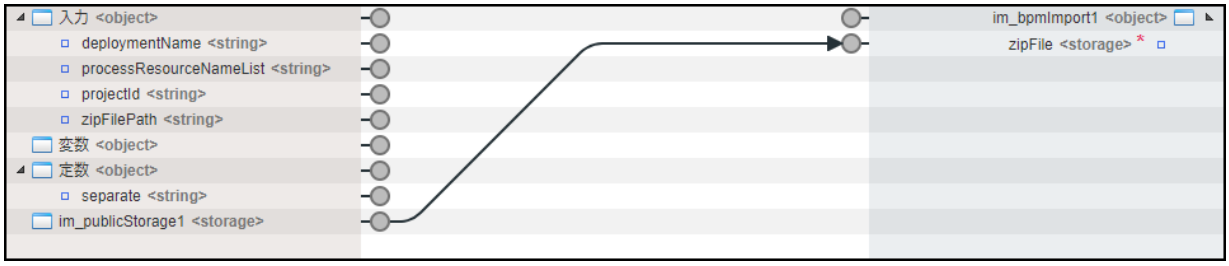
   1. 「マッピング設定」画面上部、ヘッダ内の左側に位置するセレクトボックスから「im_publicStorage1」を選択し、「入力を追加」をクリックします。



図：「マッピング設定」 - 「インポート」 - 「入力を追加」

   2. 以下のように線を引きます。

入力（始点）	出力（終点）
im_publicStorage1<storage>	im_bpmImport1<object> - zipFile<storage>



図：「マッピング設定」 - 「インポート」

> **コラム**
>
> 「インポート」についての詳細は、「IM-LogicDesigner仕様書」-「インポート」を参照してください。

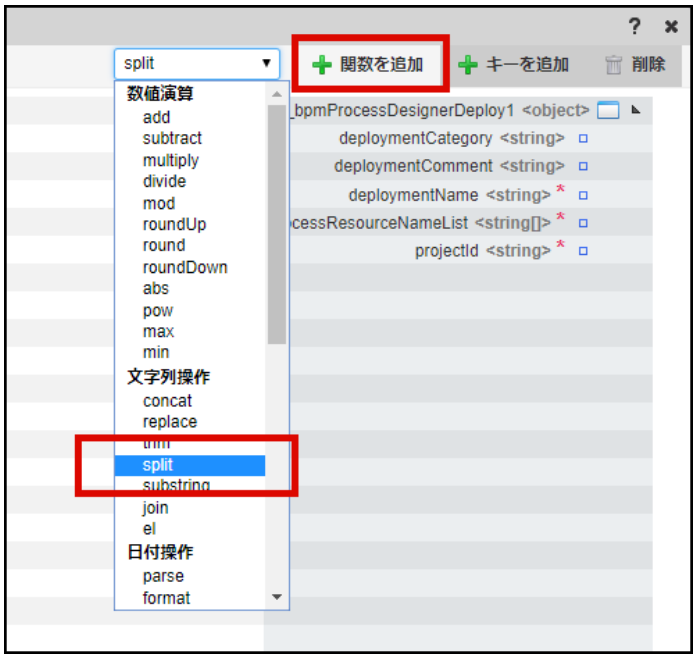
9. 「プロセスデザイナからのデプロイ」を配置します。

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

図：「プロセスデザイナからのデプロイ」

10. 「プロセスデザイナからのデプロイ」のマッピング設定をします。

    1. 「マッピング設定」画面上部、ヘッダ内の右側に位置するセレクトボックスから「split」を選択し、「関数を追加」をクリックします。

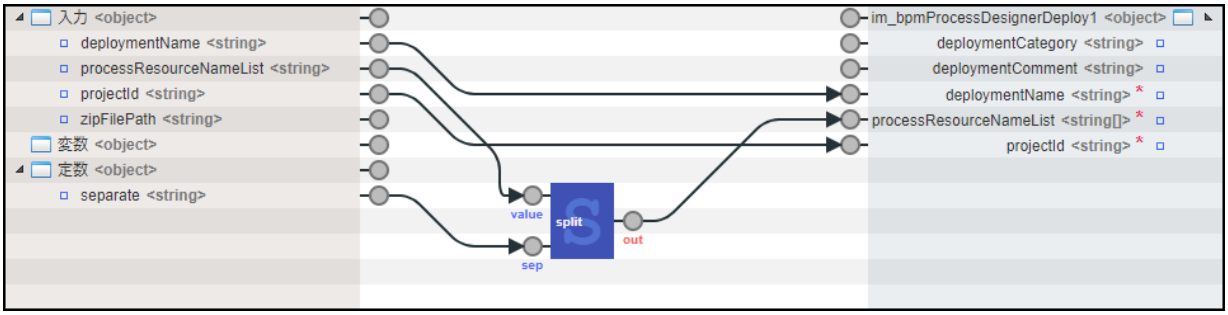


図：「マッピング設定」 - 「プロセスデザイナからのデプロイ」 - 「関数を追加」

> **コラム**
>
> マッピング関数については、「IM-LogicDesigner仕様書」-「マッピング関数一覧」を参照してください。

2. 以下のように線を引きます。

入力（始点）	出力（終点）
入力<object> - deploymentName<string>	im_bpmProcessDesignerDeploy1<object> - deploymentName<string>
入力<object> - processResourceNameList<string>	split : value
入力<object> - projectId<string>	im_bpmProcessDesignerDeploy1<object> - projectId<string>
定数<object> - separate<string>	split : sep
split : out	im_bpmProcessDesignerDeploy1<object> - processResourceNameList<string[]>



図：「マッピング設定」 - 「プロセスデザイナからのデプロイ」

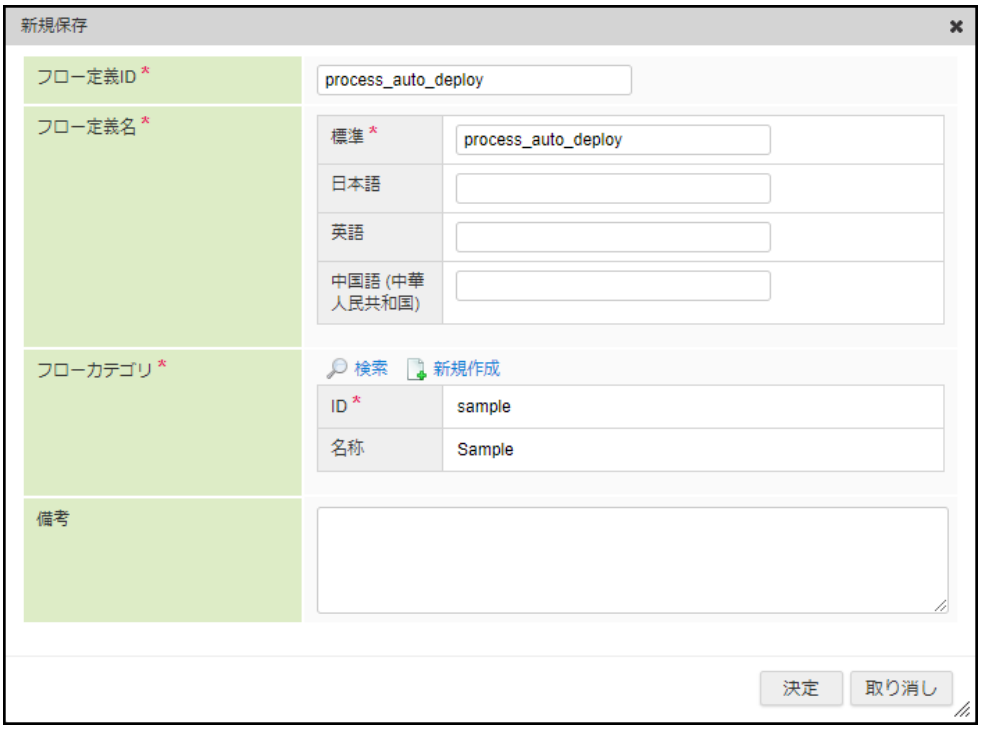
> **コラム**
>
> 「プロセスデザイナからのデプロイ」についての詳細は、「IM-LogicDesigner仕様書」-「プロセスデザイナからのデプロイ」を参照してください。

11. 「ロジックフロー定義編集」画面上部、ヘッダ内の「新規保存」をクリックします。

以下のように設定し、ロジックフローを新規保存します。

- フロー定義ID：process_auto_deploy
- フロー定義名：process_auto_deploy
- フローカテゴリ：
  - ID：sample

- 名称：Sample



図：「新規保存」

ロジックフローを実行するジョブネットを作成する

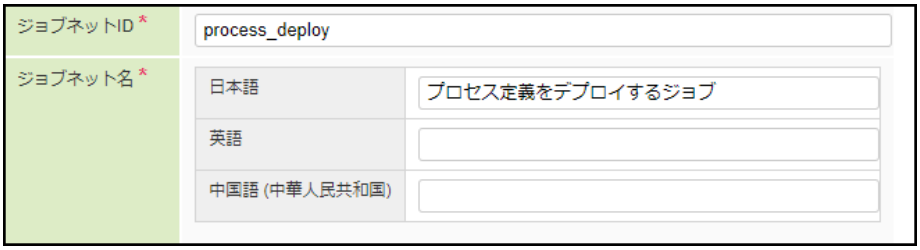
ロジックフローを実行するジョブネットを作成します。

> **コラム**
>
> このチュートリアルで作成するジョブネットのサンプルを、以下のリンクからダウンロードできます。
> **job-scheduler.xml**
>
> このサンプルはジョブネットの「ジョブインポート」でインポートできます。
> サンプル「job-scheduler.xml」 をパブリックストレージの直下に配置し、「ジョブネット管理」画面から、「テナントマスタ」 - 「インポート」 - 「ジョブインポート」を選択し、ジョブネットの「即時実行」をクリックすることでインポートできます。

1. 「サイトマップ」→「テナント管理」→「ジョブ管理」→「ジョブネット設定」から、「ジョブネット管理」画面を表示します。
2. 「ジョブネット管理」画面、ツールバー内の「ジョブネット新規作成」をクリックします。
3. 「基本情報」を設定します。

   以下のように設定します。

   - ジョブネットID：process_deploy
   - ジョブネット名：プロセス定義をデプロイするジョブ



図：「ジョブネット作成」 - 「基本情報」

4. 「実行時の情報」を設定します。
   1. 「実行ジョブ」 - 「ジョブを追加」をクリックし、「IM-LogicDesigner」 - 「フロー実行」を選択します。

図：「ジョブネット作成」 - 「実行時の情報」 - 「実行ジョブ」

> **コラム**
>
> フロー実行ジョブについての詳細は「ジョブ・ジョブネットリファレンス」-「フロー実行」を参照してください。

2. 「実行パラメータ」 - 「パラメータを追加」をクリックし、以下のように設定します。

キー	値
deploymentName	process_auto_deploy
flow_id	process_auto_deploy
processResourceNameList	tutorial_process1,tutorial_process2,tutorial_process3
projectId	processAutoDeploy
zipFilePath	process_auto_deploy.zip

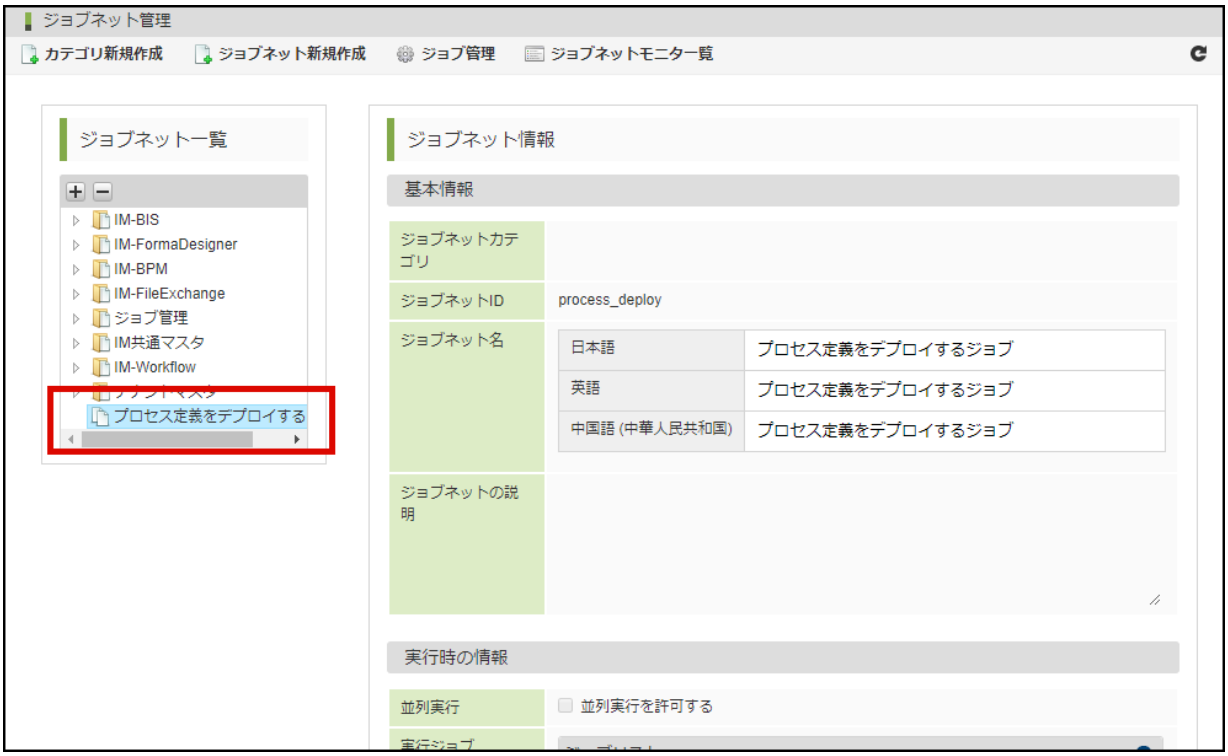
図：「ジョブネット作成」 - 「実行時の情報」 - 「実行パラメータ」

5. 「トリガ設定」を設定します。
    1. 「トリガ設定」のセレクトボックスから「日時指定」を選択し、「新規登録」を選択します。
    2. ジョブを実行する日時を指定します。
    3. 設定したトリガの「有効」のチェックボックスをオンにします。

実行結果を確認する

このチュートリアルで作成したジョブネットを実行し、実行結果の確認を行います。
ここでは、実行結果を確認するためにジョブネットの即時実行を行います。

1. 「サイトマップ」→「テナント管理」→「ジョブ管理」→「ジョブネット設定」から、「ジョブネット管理」画面を表示します。
2. 「ジョブネット管理」画面、左側のツリーから「プロセス定義をデプロイするジョブ」を選択します。



図：「ジョブネット管理」 - 「プロセス定義をデプロイするジョブ」

3. 「即時実行」をクリックします。

図：「ジョブネット管理」 - 「プロセス定義をデプロイするジョブ」- 「即時実行」

4. 「ジョブネット管理」画面、ツールバー内の「ジョブネットモニター一覧」をクリックします。

   プロセス定義をデプロイするジョブ が正常に実行されたことを確認します。

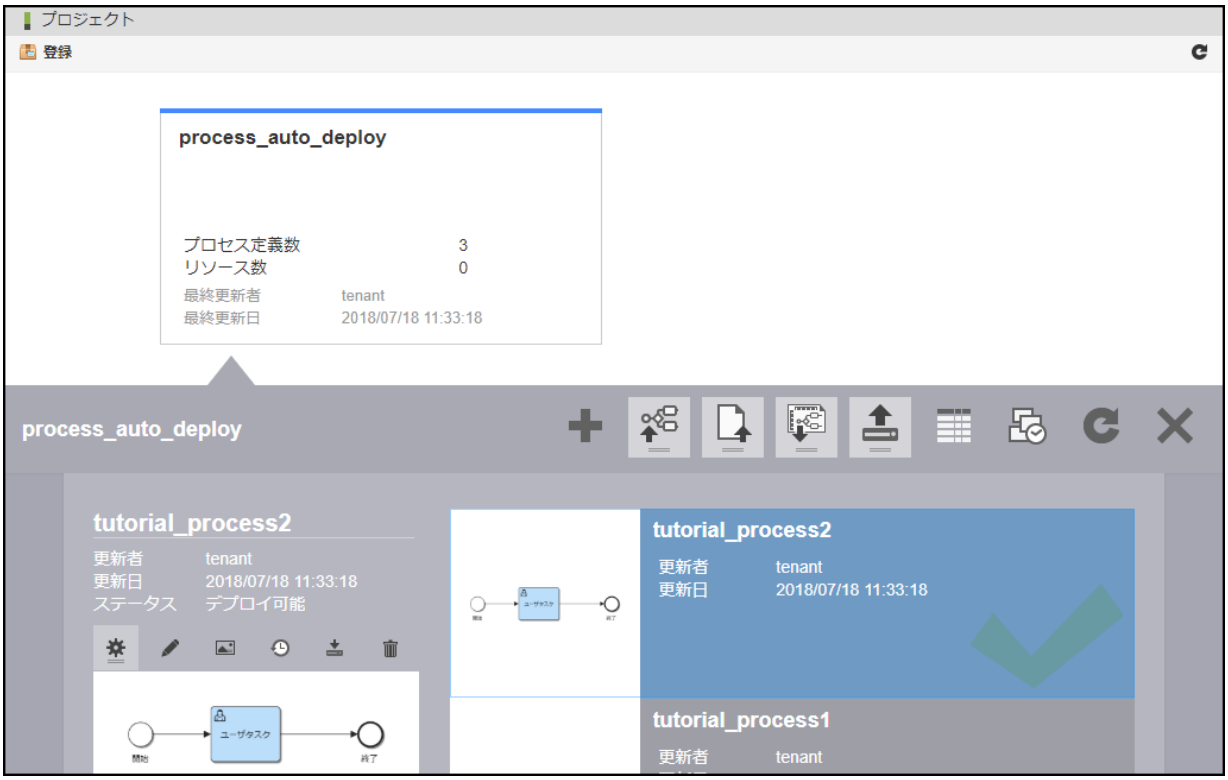
図：「ジョブネットモニター一覧」 プロセス定義をデプロイするジョブ

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

   サンプルデータを使用した場合は、プロセスデザイナのプロジェクト画面に以下のプロジェクトがインポートされていることを確認します。

   - プロジェクト名：process_auto_deploy
   - プロセス定義名：
     - tutorial_process1
     - tutorial_process2

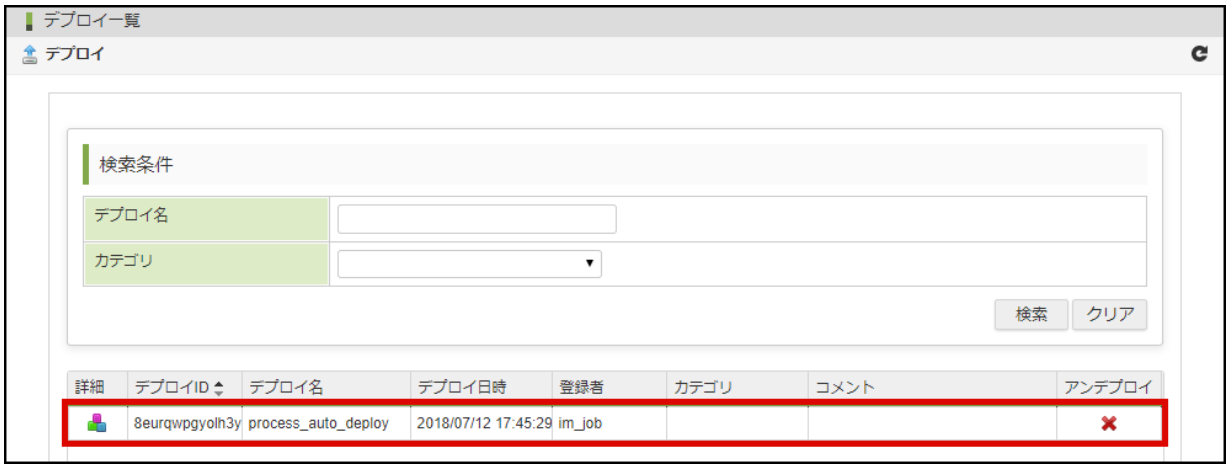
- tutorial_process3



図：「プロセスデザイナ」 process_auto_deploy

6. 「サイトマップ」→「BPM」→「デプロイ一覧」から、「デプロイ一覧」画面を表示します。

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

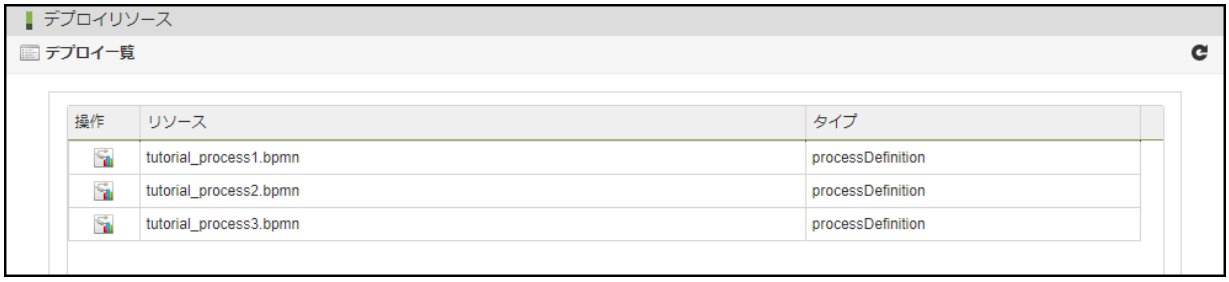


図：「デプロイ一覧」 process_auto_deploy

   2. 「デプロイ一覧」画面、一覧から「デプロイ名」 process_auto_deploy の「詳細」をクリックします。

      以下の「リソース」のファイルがデプロイ資材として展開されていることを確認します。

      - tutorial_process1.bpmn
      - tutorial_process2.bpmn
      - tutorial_process3.bpmn



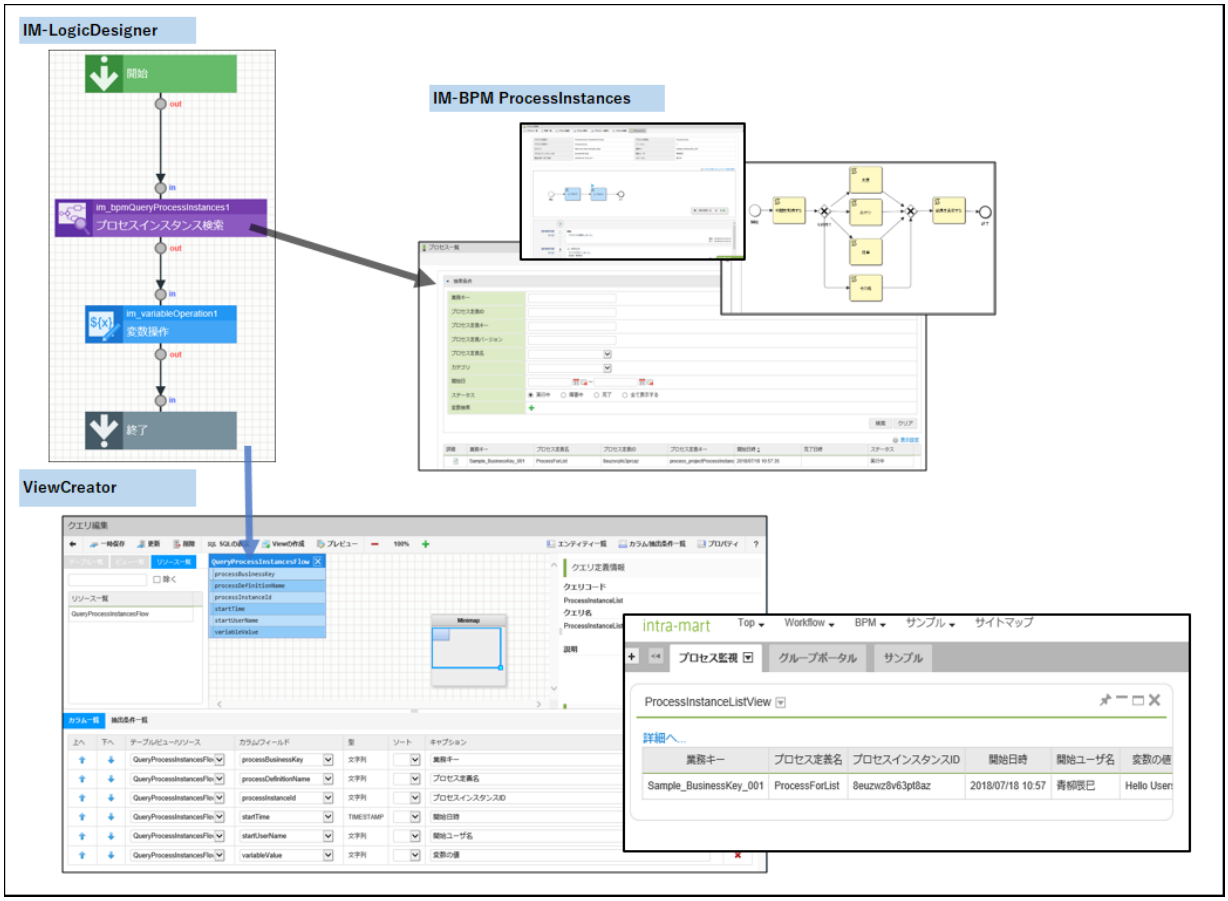
図：「デプロイリソース」

アドオン画面の作成

アドオン画面を作成する方法について説明します。
このチュートリアルは「*事前準備*」が完了していることを前提としています。

アドオンのプロセス一覧画面の作成

このチュートリアルでは、IM-LogicDesignerのロジックフローとViewCreatorを使用し、カスタマイズしたプロセスインスタンスの一覧画面を作成する方法を解説します。



図：作成物の構成

> **コラム**
>
> IM-LogicDesignerとは、intra-mart Accel Platform上でビジネスロジックを簡単に作成できるアプリケーションです。
> IM-LogicDesignerの詳細については、「IM-LogicDesigner仕様書」を参照してください。

> **コラム**
>
> ViewCreatorはintra-martのWeb画面上から、データベースなどのデータを使用して、表やグラフを簡単に作成できるツールです。
> ViewCreatorの詳細については、「ViewCreator 管理者操作ガイド」を参照してください。

コラム

このチュートリアルで使用する「デプロイメント情報」および、作成する「ロジックフロー定義」、「クエリ」、「データ参照」を以下のリンクからダウンロードできます。

- 「デプロイメント情報」
  im_bpm_process_instance_list.zip
- IM-LogicDesigner「ロジックフロー定義」
  im_logicdesigner-data-process-instance-list-view.zip
- ViewCreator「クエリ」
  ProcessInstanceList.xml
- ViewCreator「データ参照」
  im_bpm_process_instance_list_view.xml

各種インポート方法については、以下のリンクを参照してください。

- 「デプロイメント情報」：「IM-BPM ユーザ操作ガイド」-「インポート」
- IM-LogicDesignerの「ロジックフロー」：「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」

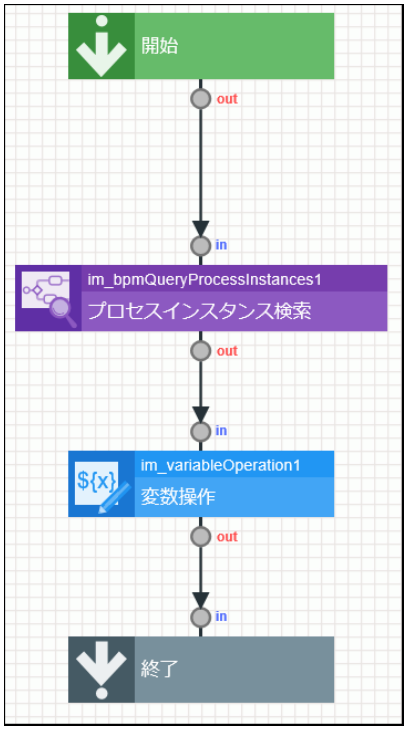
注意

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

- プロセスインスタンスのデータを取得するIM-LogicDesignerのロジックフローを作成する
- IM-LogicDesignerのロジックフローと連携したViewCreatorのクエリ・データ参照を作成する
- 作成したプロセスインスタンス一覧画面を確認する

プロセスインスタンスのデータを取得するIM-LogicDesignerのロジックフローを作成する

プロセスインスタンスの検索を行い、検索結果を出力項目`result<object[]>`として返却するIM-LogicDesignerのロジックフローを作成します。



図：完成イメージ

1. 「サイトマップ」→「LogicDesigner」→「フロー定義一覧」から、「ロジックフロー定義一覧」画面を表示します。
2. 「フロー定義一覧」画面、ツールバー内の「ロジックフロー新規作成」をクリックします。
3. 「ロジックフロー定義編集」画面上部、ヘッダ内の「定数設定」をクリックします。

   定数値を以下のように設定します。

定数ID	定数値
TRUE	true
processBusinessKeyLike	%BusinessKey%

図：定数設定

4. 「ロジックフロー定義編集」画面上部、ヘッダ内の「変数設定」をクリックします。

   変数に以下のパラメータを追加します。

   1. 「object」をクリックし、キー名をeditedQueryResultとします。

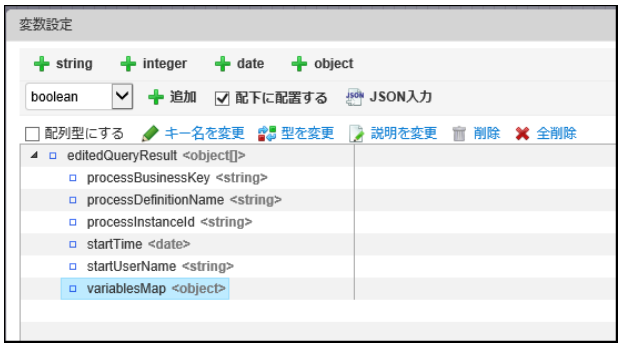
   2. 「配列型にする」にチェックを入れます。

キー名	型
editedQueryResult	object[]

      図：editedQueryResult<object[]>の設定

   3. 「配下に配置する」にチェックを入れ、editedQueryResult<object[]>パラメータの配下に以下の項目を設定します。

キー名	型
processBusinessKey	string
processDefinitionName	string
processInstanceId	string
startTime	date
startUserName	string
variablesMap	object

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

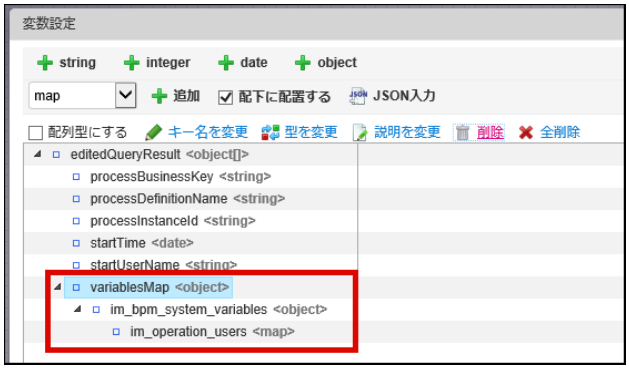
   4. editedQueryResult<object[]>配下のvariablesMap<object>を選択し、「配下に配置する」にチェックを入れ、以下の項目を設定します。

キー名	型
im_bpm_system_variables	object

   5. variablesMap<object>配下のim_bpm_system_variables<object>を選択し、「配下に配置する」にチェックを入れ、以下の項目を設定し

ます。

キー名	型
im_operation_users	map



図：variablesMap<object>の配下の項目設定

5. 「ロジックフロー定義編集」画面上部、ヘッダ内の「入出力設定」をクリックします。

   「出力」に以下のパラメータを追加します。

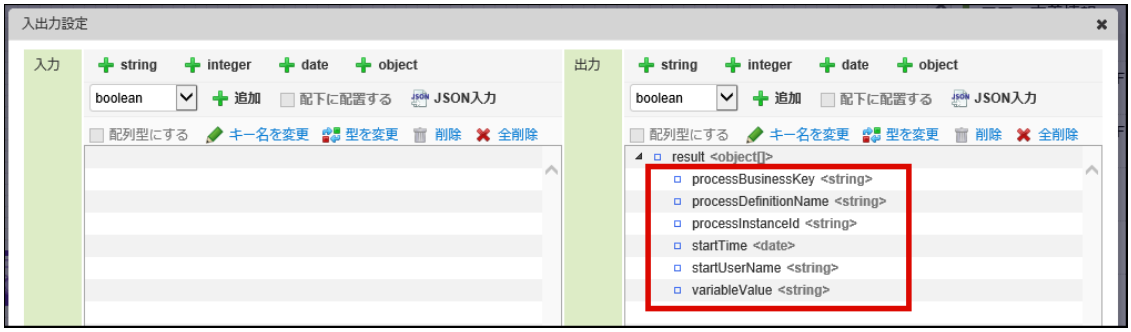
   1. 「object」をクリックし、キー名をresultとします。
   2. 「配列型にする」にチェックを入れます。

キー名	型
result	object[]

図：result<object[]>の設定

   3. 「配下に配置する」にチェックを入れ、result<object[]>パラメータの配下に以下のキーを設定します。

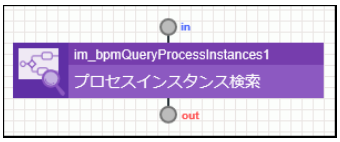
キー名	型
processBusinessKey	string
processDefinitionName	string
processInstanceId	string
startTime	date
startUserName	string
variableValue	string



図：result<object[]>の配下のキー設定

> **コラム**
>
> キーresult<object[]>は、ViewCreatorと連携する際には必須のパラメータです。
>
> ロジックフローの出力結果をリソースとして利用する際の詳細は「ViewCreator 管理者操作ガイド」-「ViewCreatorで利用可能なロジックフローの出力設定」を参照してください。

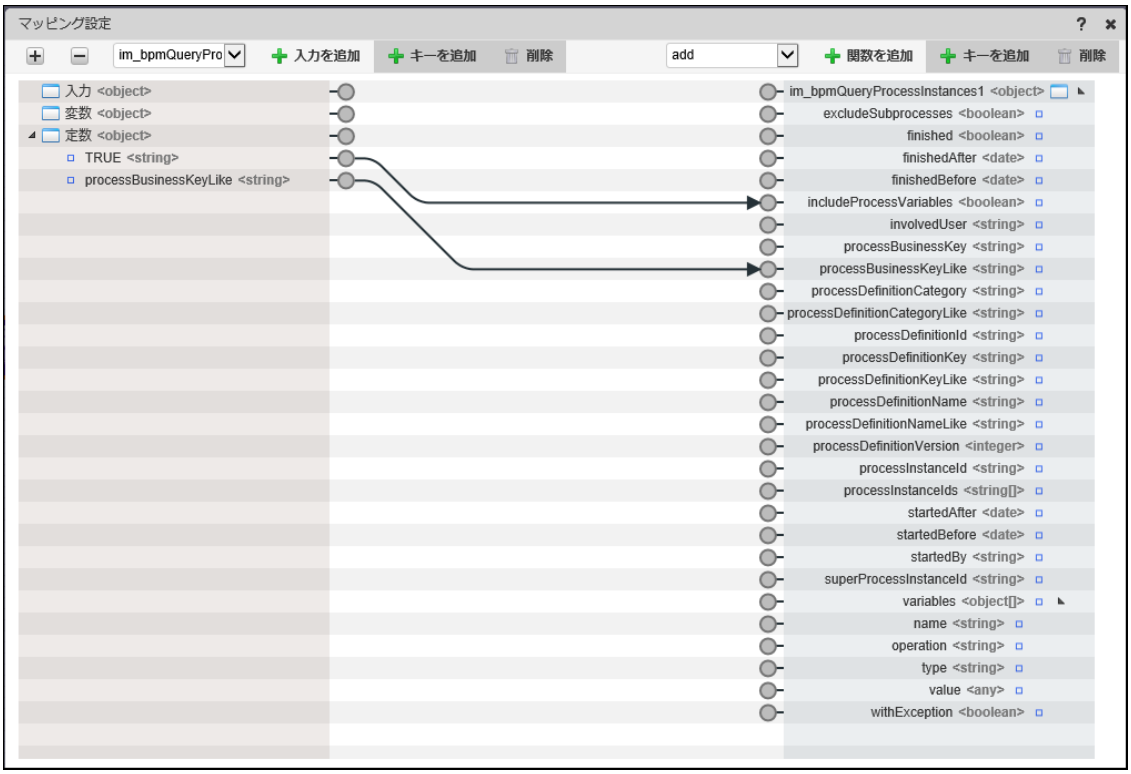
6. パレット内の「IM-BPM」の一覧から「プロセスインスタンス検索」を選び、フロー編集画面上に追加します。



図：「プロセスインスタンス検索」タスク

7. 「プロセスインスタンス検索」のプロパティ項目「マッピング設定」を開き、定数項目から「出力ペイン」項目のim_bpmQueryProcessInstances1<object>配下の各キーに対し、以下のようにマッピングを行います。

入力（始点）	出力（終点）
定数<object> - TRUE<string>	im_bpmQueryProcessInstances1<object> - includeProcessVariables<boolean>
定数<object> - processBusinessKeyLike<string>	im_bpmQueryProcessInstances1<object> - processBusinessKeyLike<string>



図：「プロセスインスタンス検索」タスクへのマッピング設定

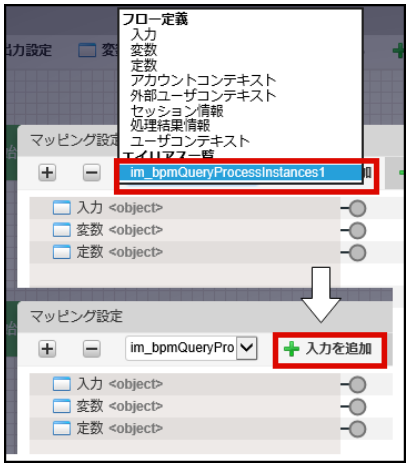
> **コラム**
>
> 上記の設定では、プロセスインスタンスを「業務キー」で部分一致検索し、返却される項目へ「プロセス変数を含む」オプションを設定しています。
>
> 「プロセスインスタンス検索」に関するタスクの詳細は「IM-LogicDesigner仕様書」-「プロセスインスタンス検索」を参照してください。

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

図：「変数操作」タスク

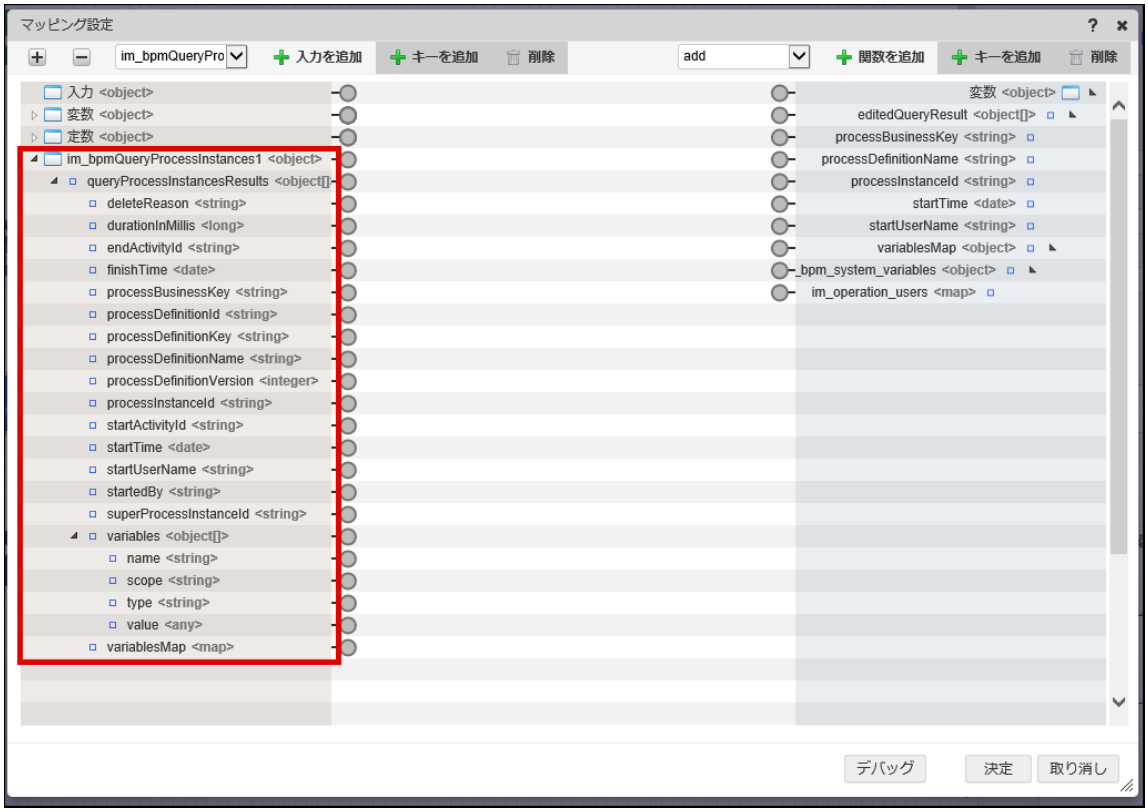
9. 「変数操作」のプロパティ項目「マッピング設定」を開き、以下のようにマッピングを行います。

   1. 「マッピング設定」画面上部、ヘッダ内の左上に位置するセレクトボックスからim_bpmQueryProcessInstances1を選択し、「入力を追加」をクリックします。



図：im_bpmQueryProcessInstances1を選択

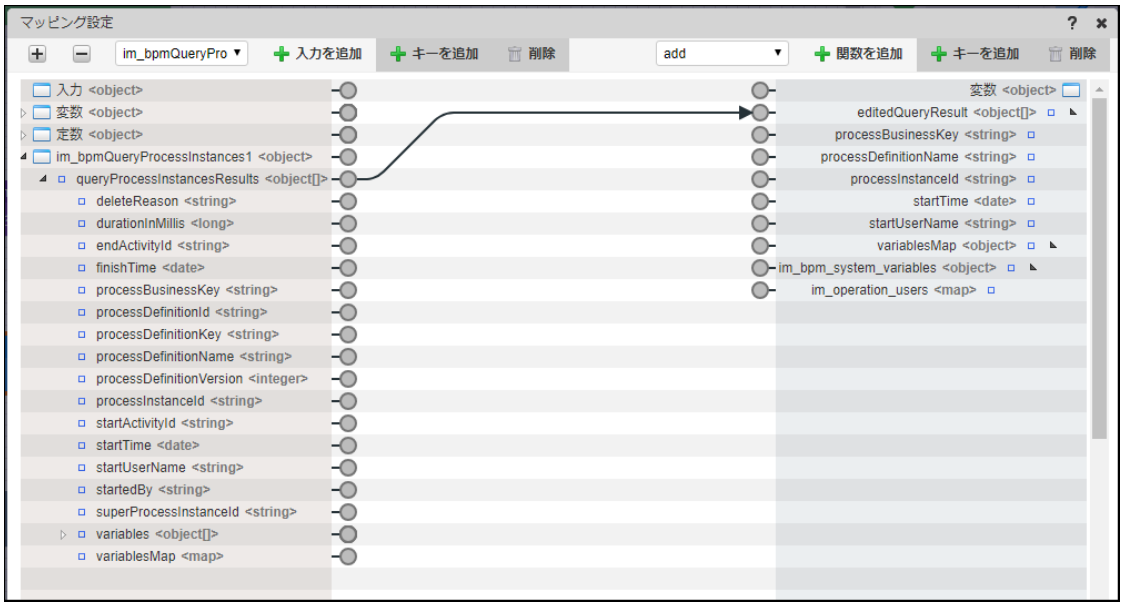
以下のように、「入力ペイン」項目にim_bpmQueryProcessInstances1<object>が追加されます。



図：im_bpmQueryProcessInstances1<object>の追加

   2. 「入力ペイン」項目のim_bpmQueryProcessInstances1<object>配下のqueryProcessInstancesResults<object[]>配下の各キーより変数editedQueryResult<object[]>の各キーに対して、以下のようにマッピングを行います。

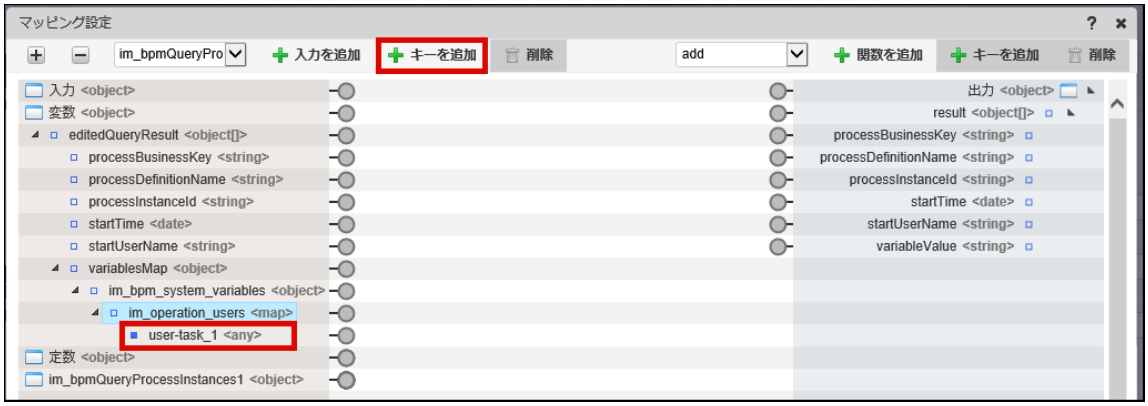
入力（始点）	出力（終点）
im_bpmQueryProcessInstances1<object> - queryProcessInstancesResults<object[]>	変数<object> - editedQueryResult<object[]>



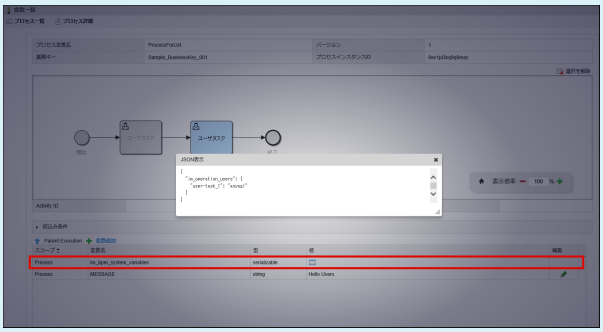
図：editedQueryResult<object[]>へのマッピング設定

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

    1. 「入力ペイン」項目の「変数」内のeditedQueryResult<object[]> - variablesMap<object> - im_bpm_system_variables<object> - im_operation_users<map>を選択し、「マッピング設定」画面上部、ヘッダ内の「キーを追加」をクリックしてキーの追加を行い、追加されたキーの名称にuser-task_1と入力します。

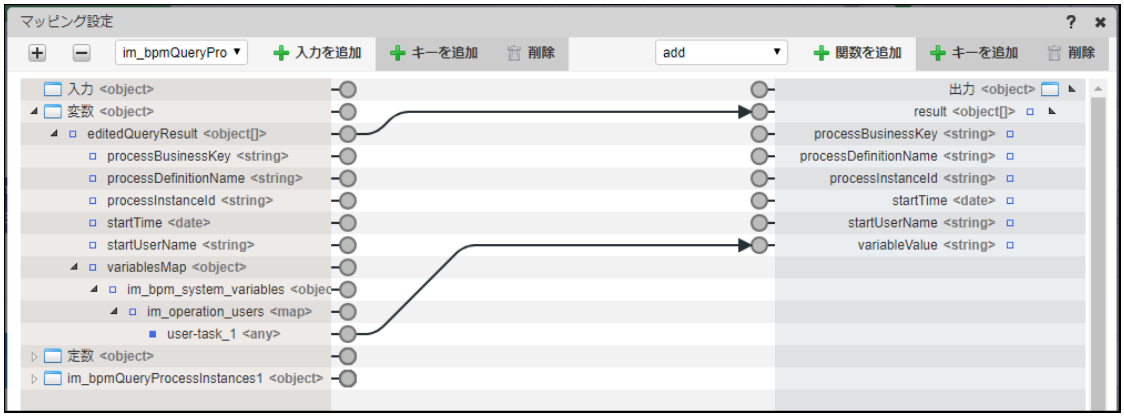


図：im_operation_users<map>へキーuser-task_1を追加

> **コラム**
>
> 上記の設定は、プロセスインスタンスが内部的に保持しているMap型の変数im_operation_usersから、タスク「user-task_1」の処理を行ったユーザのユーザコードを取得するための設定です。
>
> プロセスインスタンスが保持している変数については、以下のように標準機能の「変数一覧」画面でも確認できます。
>
> 
>
> 図：「変数一覧」画面
>
> 変数一覧画面についての詳細は「IM-BPM ユーザ操作ガイド」 - 「プロセスインスタンスの変数を確認する」を参照してください。

    2. 「入力ペイン」項目の「変数」内のeditedQueryResult<object[]>配下の各キーより、出力項目result<object[]>の配下の各キーに対して以下のようにマッピングを行います。

入力（始点）	出力（終点）
変数<object> - editedQueryResult<object[]>	出力<object> - result<object[]>
... - im_operation_users<map> - user-task_1<any>	出力<object> - result<object[]> - variableValue<string>



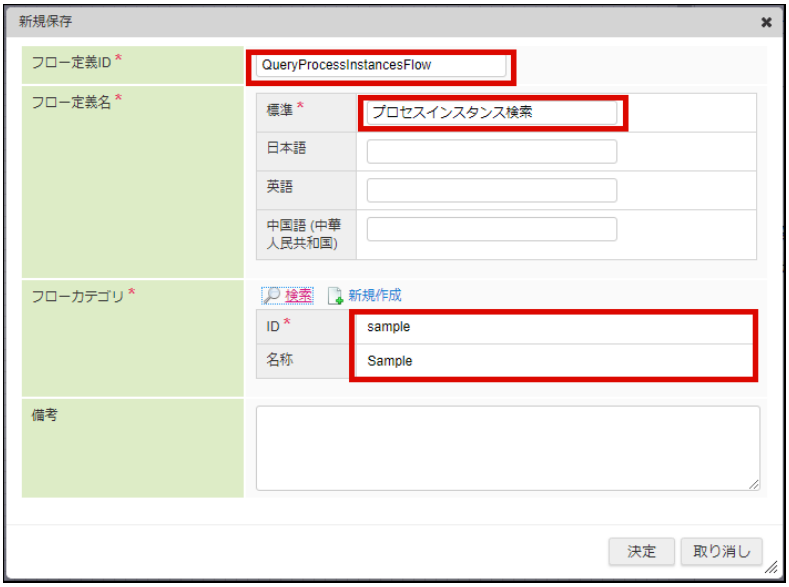
図：result<object[]>へのマッピング設定

11. 「開始」→「プロセスインスタンス検索」→「変数操作」→「終了」の順にタスクを接続します。

12. 「新規保存」をクリックし、以下を入力し、「決定」をクリックします。

<画面項目>

項目	設定値
フロー定義ID	QueryProcessInstancesFlow
フロー定義名（標準）	プロセスインスタンス検索
フローカテゴリ	Sample



図：新規保存ダイアログ

> **コラム**
>
> マッピング元とマッピング先で配下に同名の値を持っている場合、IM-LogicDesignerは自動でその値をマッピングします。
>
> 階層化された値のマッピングの詳細については、「IM-LogicDesigner チュートリアルガイド」-「階層化された入力値・出力値の定義（Object型の定義）」を参照してください。

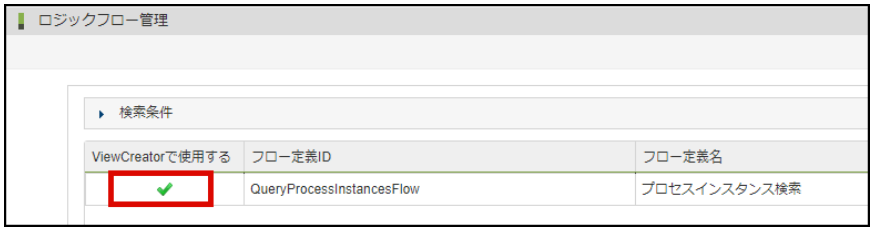
IM-LogicDesignerのロジックフローと連携したViewCreatorのクエリ・データ参照を作成する

IM-LogicDesignerのロジックフローからデータを取得し、ViewCreatorで表示するためのクエリ・データ参照の設定を作成します。

1. 外部データソース定義

1. 「サイトマップ」→「ViewCreator」→「外部データソース連携」→「ロジックフロー管理」をクリックし「ロジックフロー管理」画面を表示します。

2. 「ロジックフロー管理」画面のフロー定義ID一覧に表示されているフロー定義IDQueryProcessInstancesFlowの「ViewCreatorで使用する」にチェックを入れます。

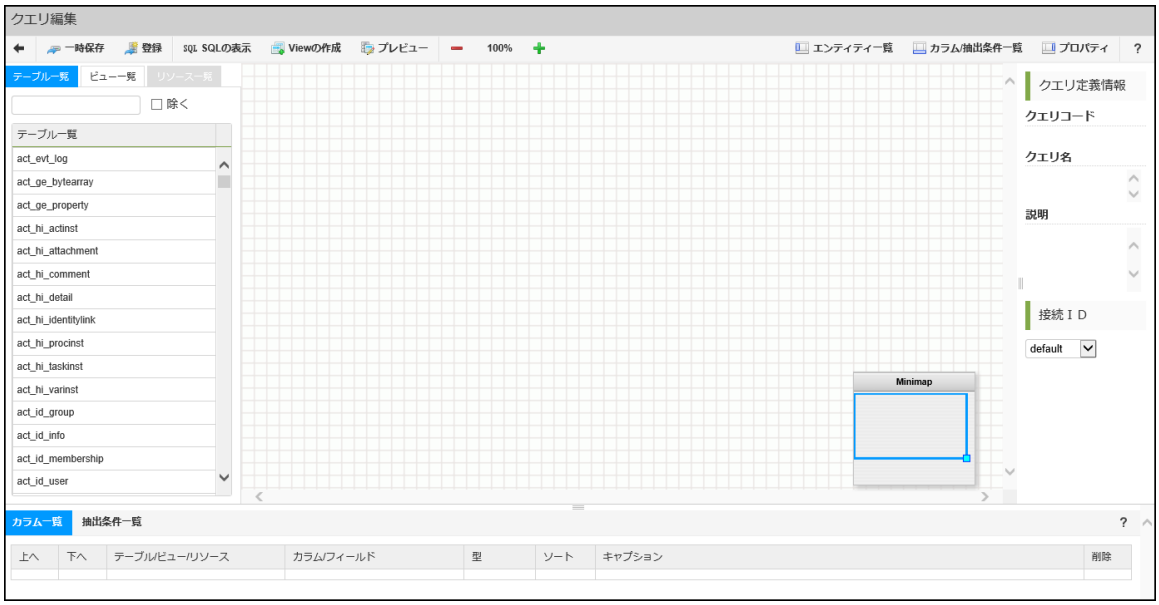


図：「ViewCreatorで使用する」にチェック

2. クエリの作成

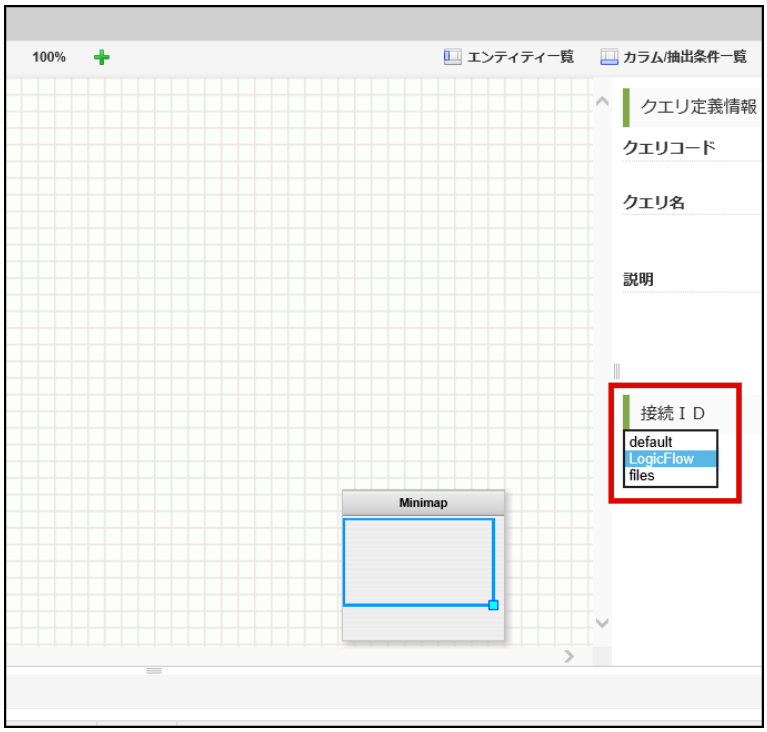
1. 「サイトマップ」→「ViewCreator」→「クエリ一覧」から、ViewCreatorの「クエリ一覧」画面を表示します。

2. クエリ一覧画面の「新規」をクリックして「クエリ編集」画面を表示します。



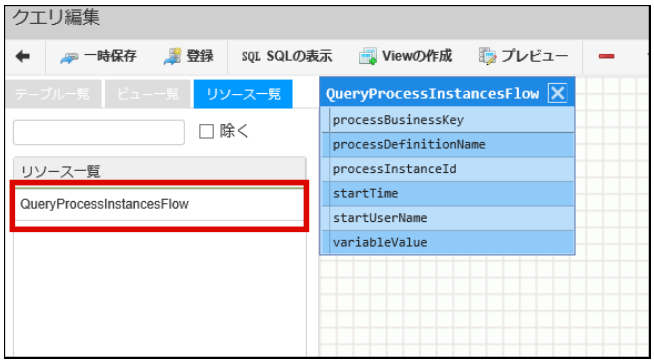
図：クエリの新規作成

3. 「接続ID」のプルダウンよりLogicFlowを選択します。



図：接続IDの選択

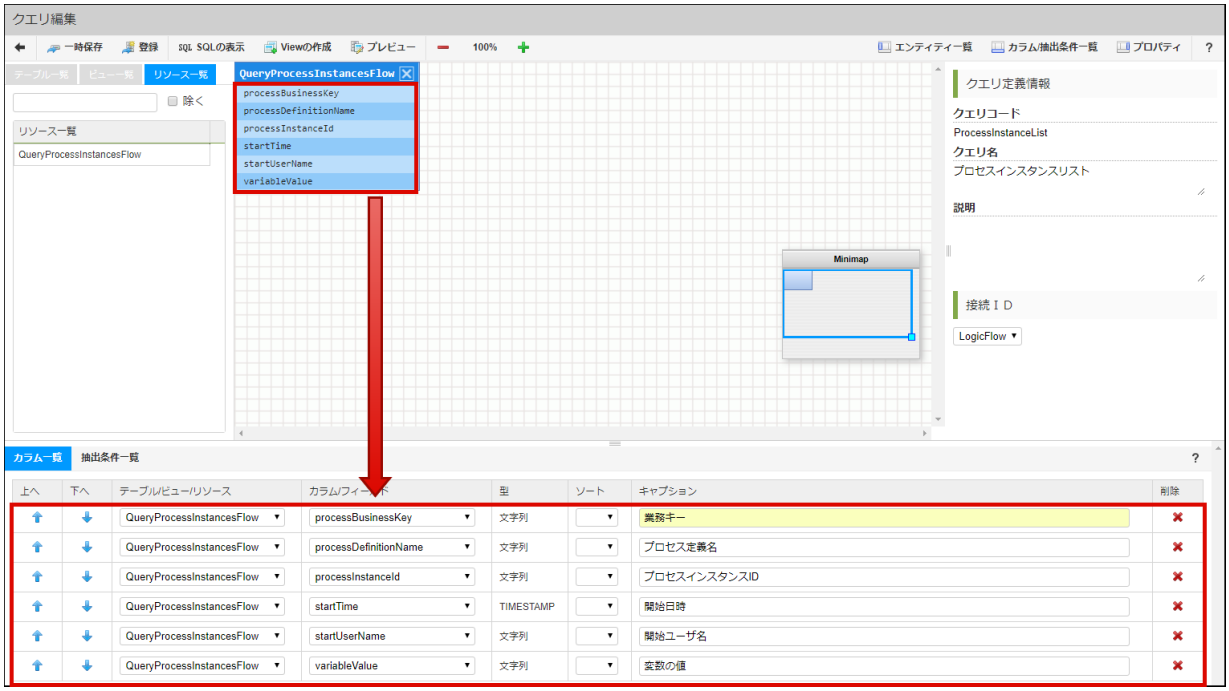
4. 「リソース一覧」よりQueryProcessInstancesFlowを選択します。



図：リソース一覧

5. デザイナ部分に表示されたリソースQueryProcessInstancesFlowより、以下の「フィールド」をダブルクリックし、クエリの「カラム」として選択し、選択した「カラム」に「キャプション」を設定します。

フィールド名	キャプション
processBusinessKey	業務キー
processDefinitionName	プロセス定義名
processInstanceId	プロセスインスタンスID
startTime	開始日時
startUserName	開始ユーザ名
variableValue	変数の値

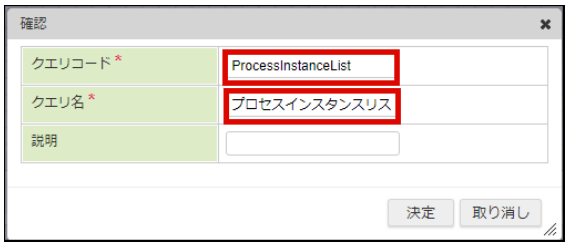


図：カラムの選択とキャプションの設定

6. 「クエリ編集」画面上部の「登録」ボタンをクリックし、以下の値を入力し、作成したクエリを登録します。

<画面項目>

項目	設定値
クエリコード	ProcessInstanceList
クエリ名	プロセスインスタンスリスト



図：登録確認ダイアログ

3. データ参照の作成

   1. 「サイトマップ」→「ViewCreator」→「クエリ一覧」から、ViewCreatorの「クエリ一覧」画面を表示します。

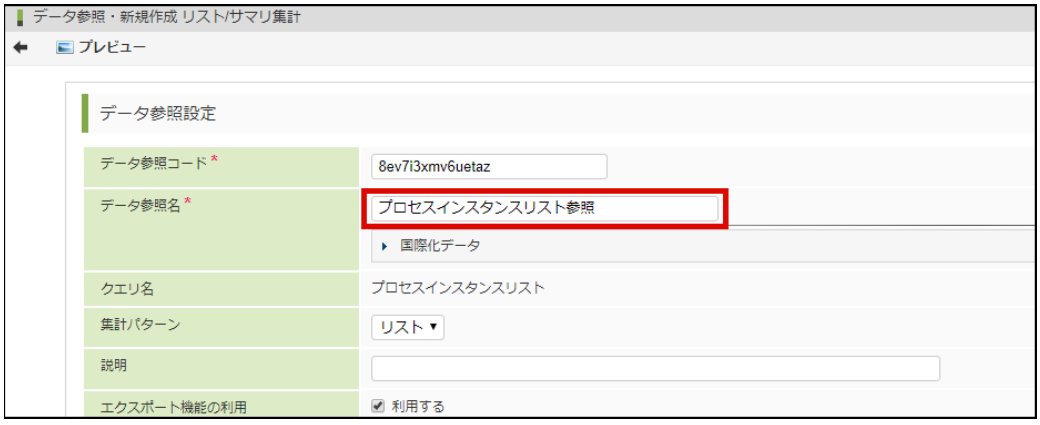
   2. プロセスインスタンスリストの「データ参照作成」欄のリスト集計作成（）をクリックし、「リスト集計の作成」画面を表示します。

図：リスト集計作成

   3. 「リスト集計の作成」画面の入力項目に以下の値を入力します。

      <画面項目>

項目	設定値
データ参照名	プロセスインスタンスリスト参照
参照権 - 認証済みユーザ	チェック



図：データ参照名

図：参照権 - 認証済みユーザ

   4. カラム一覧のカラムプロセス定義名（processDefinitionName）の「ソート順」プルダウンより「昇順」を、カラム開始日時（startTime）の「ソート順」プルダウンより「降順」をそれぞれ選択します。

図：ソートの設定

5. カラム一覧のカラム開始日時（startTime）の「フォーマット」にyyyy/MM/dd HH:mm:ssを設定します。

図：日付フォーマットの設定

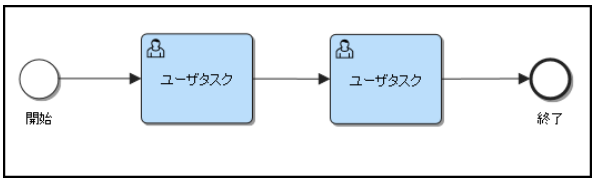
> **コラム**
>
> 上記の設定では、プロセス定義名（processDefinitionName）カラム（昇順）が第１ソートキー、開始日時（startTime）カラム（降順）が第２ソートキーです。
>
> 日付フォーマットや、「リスト集計の作成」についてのその他の設定の詳細は「ViewCreator 管理者操作ガイド」-「ViewCreator 管理者操作ガイド - リスト集計の作成」を参照してください。

6. 「登録して一覧へ戻る」をクリックします。

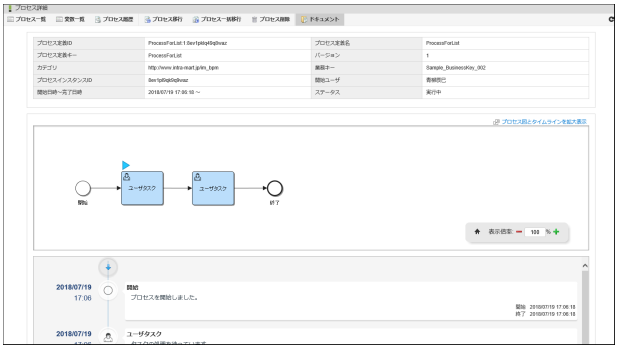
作成したプロセスインスタンス一覧画面を確認する

検索対象となるプロセスインスタンスを作成し、プロセスインスタンス一覧画面を確認します。

1. 検索対象となるプロセスインスタンスを作成するために、デプロイメント情報をインポートします。



図：インポートされるプロセス定義



図：実行中のプロセスインスタンスの詳細画面

コラム

デプロイされたプロセス定義を開始することにより、プロセスインスタンスが作成されます。

プロセスインスタンスの確認については、「IM-BPM ユーザ操作ガイド」-「プロセスインスタンスを確認する」を参照してください。

1. 「サイトマップ」→「BPM」→「インポート」画面を表示し、「インポートファイル」項目に、デプロイメント情報im_bpm_process_instance_list.zipを設定し、「実行」ボタンをクリックします。

図：インポート画面

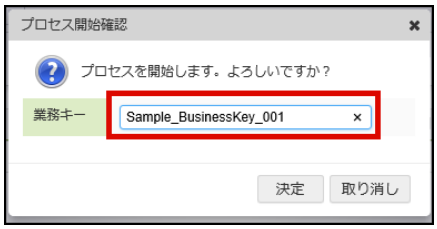
2. 「サイトマップ」→「BPM」→「デプロイ一覧」画面を表示し、デプロイ名DeployProcessForListが存在することを確認します。

図：デプロイ一覧画面

3. 「サイトマップ」→「BPM」→「プロセス開始一覧」画面を表示し、プロセス定義名ProcessForListのプロセス開始（ ）をクリックし、「プロセス開始確認」ダイアログの「業務キー」に以下の値を入力しプロセスを開始してください。

合計で3件のプロセスを開始します。

業務キー
Sample_BusinessKey_001
Sample_BusinessKey_002
Sample_ManagementProcessKey_001



図：プロセス開始確認ダイアログ

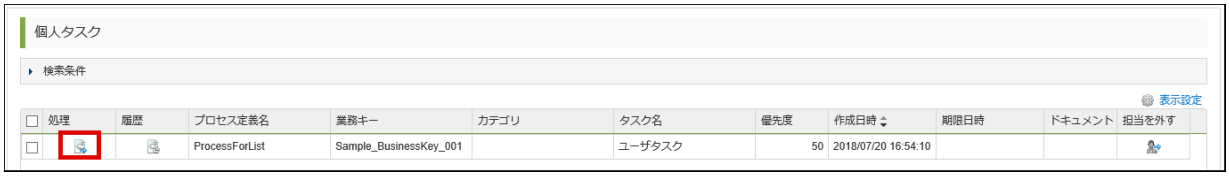
4. 「サイトマップ」→「BPM」→「タスク一覧」画面を表示し、グループタスクにプロセス定義名 ProcessForList のプロセスのユーザタスクが3件存在することを確認します。

図：タスク一覧画面

5. グループタスクの「業務キー」に Sample_BusinessKey_001 が設定されているユーザタスクの担当にする（）をクリックし、タスクの割り当てを行います。

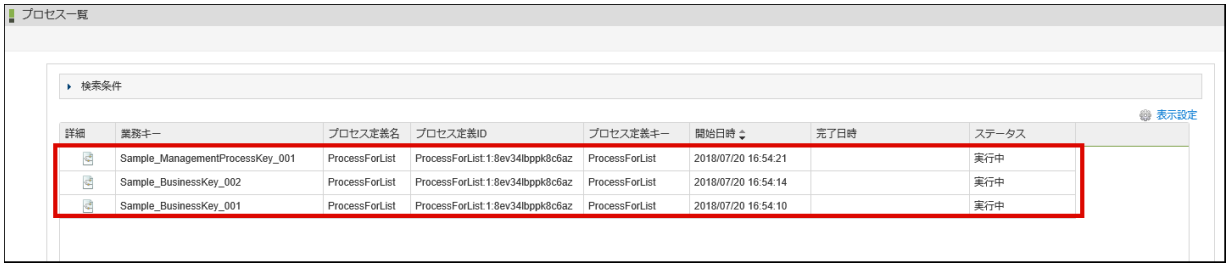
図：グループタスク一覧

6. 「業務キー」に Sample_BusinessKey_001 が設定されているユーザタスクが個人タスクに移動したことを確認し、個人タスクの処理（）をクリックし、処理確認ダイアログで「決定」をクリックし、タスクを処理します。



図：個人タスク一覧

7. 「サイトマップ」→「BPM」→「プロセス一覧」画面を表示し、プロセス定義名 ProcessForList のプロセスが3件存在することを確認します。



図：プロセス一覧画面

> **注意**
>
> 本チュートリアルではプロセスの開始を行うユーザは、「IM-BPM管理者（im_bpm_manager）」ロールを持つユーザであることを前提としています。
> プロセス定義の開始権限の詳細については「IM-BPM 仕様書」 - 「プロセス定義の開始権限」を参照してください。

2. ViewCreatorで作成したプロセスインスタンスの一覧画面を確認します。

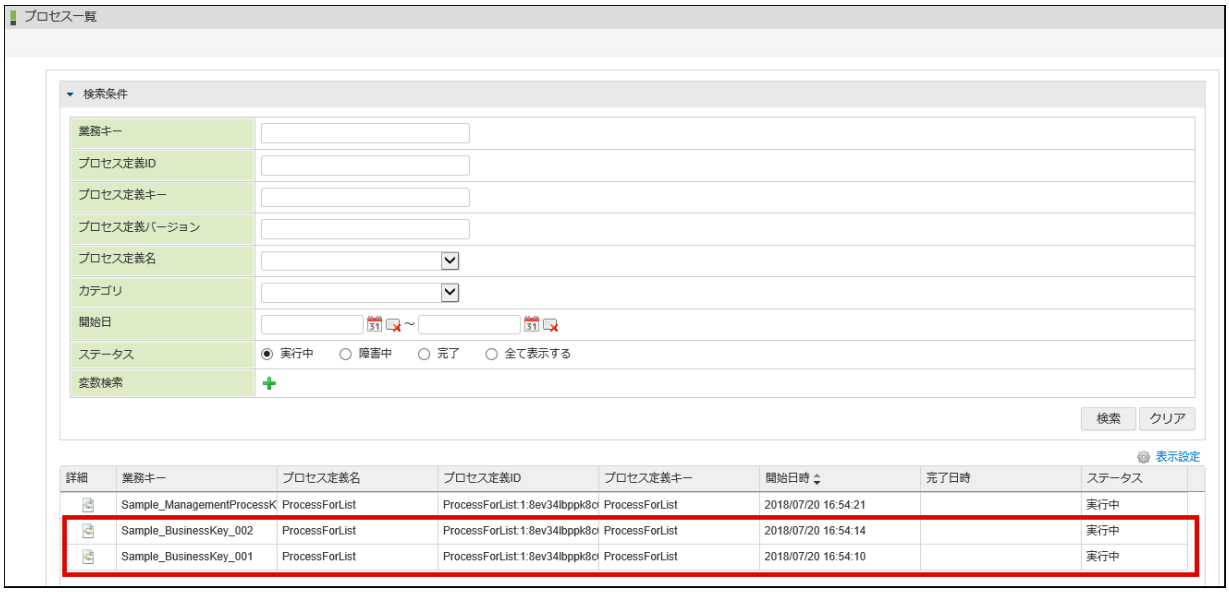
1. 「サイトマップ」→「ViewCreator」→「データ参照一覧」から、ViewCreatorの「データ参照一覧」画面を表示します。
2. 一覧に表示されている プロセスインスタンスリスト参照 のリンクをクリックします。

図：データ参照へのリンク

3. 「業務キー」に文字列 BusinessKey を含むプロセスインスタンスのリストが表示されることを確認します。

図：データ参照-プロセスインスタンス一覧

4. 「サイトマップ」→「BPM」→「プロセス一覧」画面を表示し、プロセス定義名「ProcessForList」のプロセスが存在することを確認します。



図：標準機能のプロセス一覧画面

5. 「プロセス一覧」画面に表示されている「業務キー」、「プロセス定義名」、「開始日時」が作成したデータ参照のプロセスインスタンス一覧の「業務キー」、「プロセス定義名」、「開始日時」と等しいこと、「プロセス一覧」画面には存在しない「プロセスインスタンスID」、「開始ユーザ名」、「変数の値」が表示されていることを確認します。

> **コラム**
>
> 作成したデータ参照はポートレットとして登録することで、ポータル画面へ表示することが可能です。
>
> ポートレットとして使用する場合の詳細については「ViewCreator 管理者操作ガイド」-「ポータルとの連携」を参照してください。

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、以下の資材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

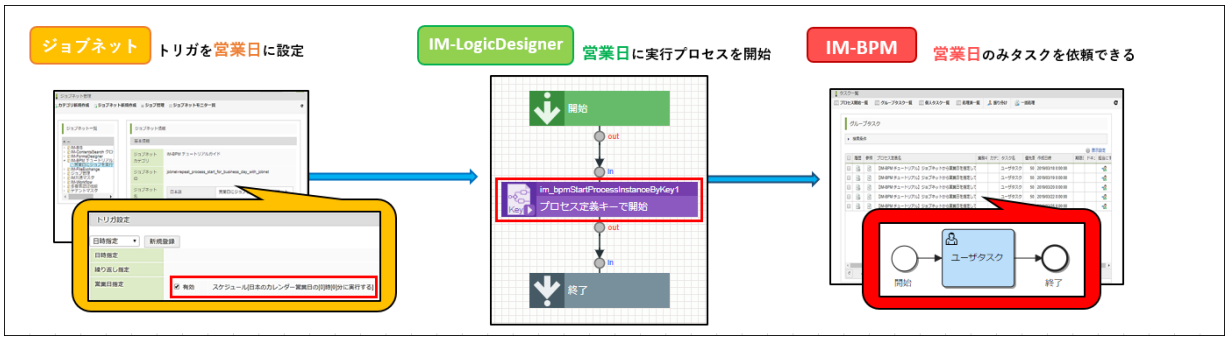


図 ： 概要図

> **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。

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

コラム

ストレージに対しての操作、およびジョブスケジューラの利用はテナント管理者が行えます。
詳細については、以下のリンク先を参照してください。

- 「システム管理者操作ガイド」-「ファイル操作」
- 「ジョブスケジューラ仕様書」

コラム

このチュートリアルで作成する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。

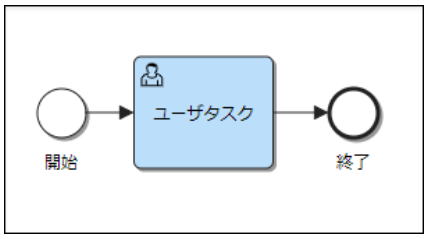


図 ： 完成イメージ

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。

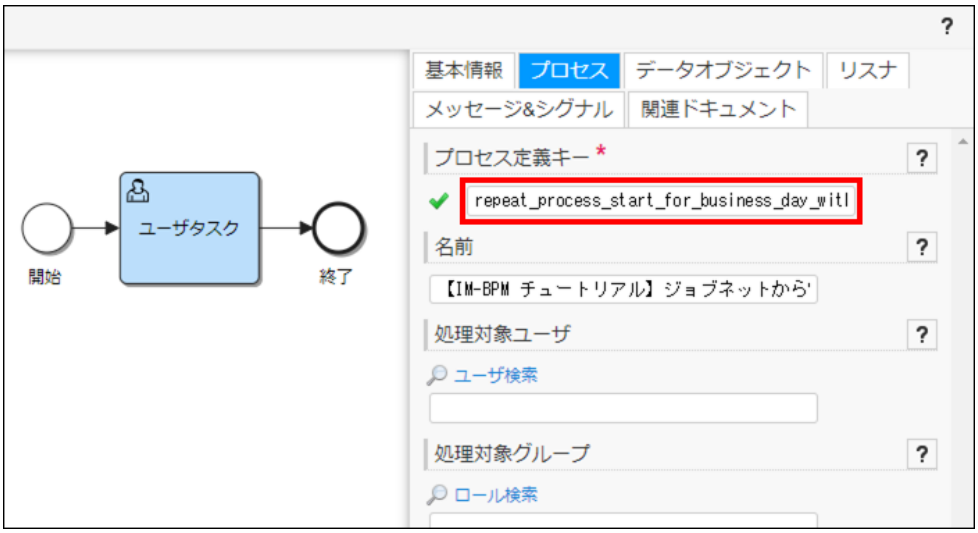


図 ： プロセス全体 - 「プロパティ」 - 「プロセス」

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。

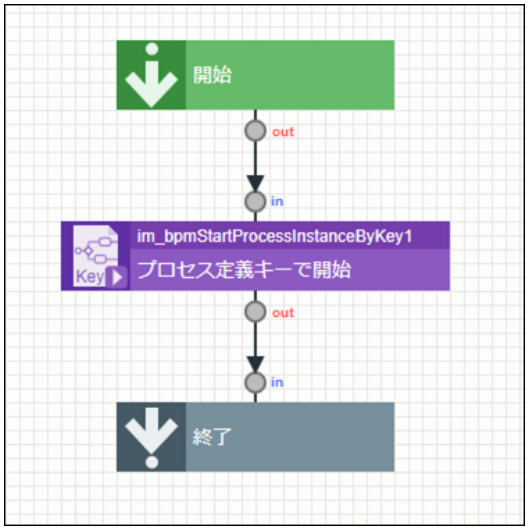


図 ： ロジックフロー（IM-LogicDesigner）

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. 左ペインのツリーから「チュートリアル」カテゴリの開閉アイコンをクリックしカテゴリ配下を情報を表示します。
   フロー定義「【チュートリアル】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」を選択し「編集」をクリックし、「ロジックフロー定義編集」画面を開きます。

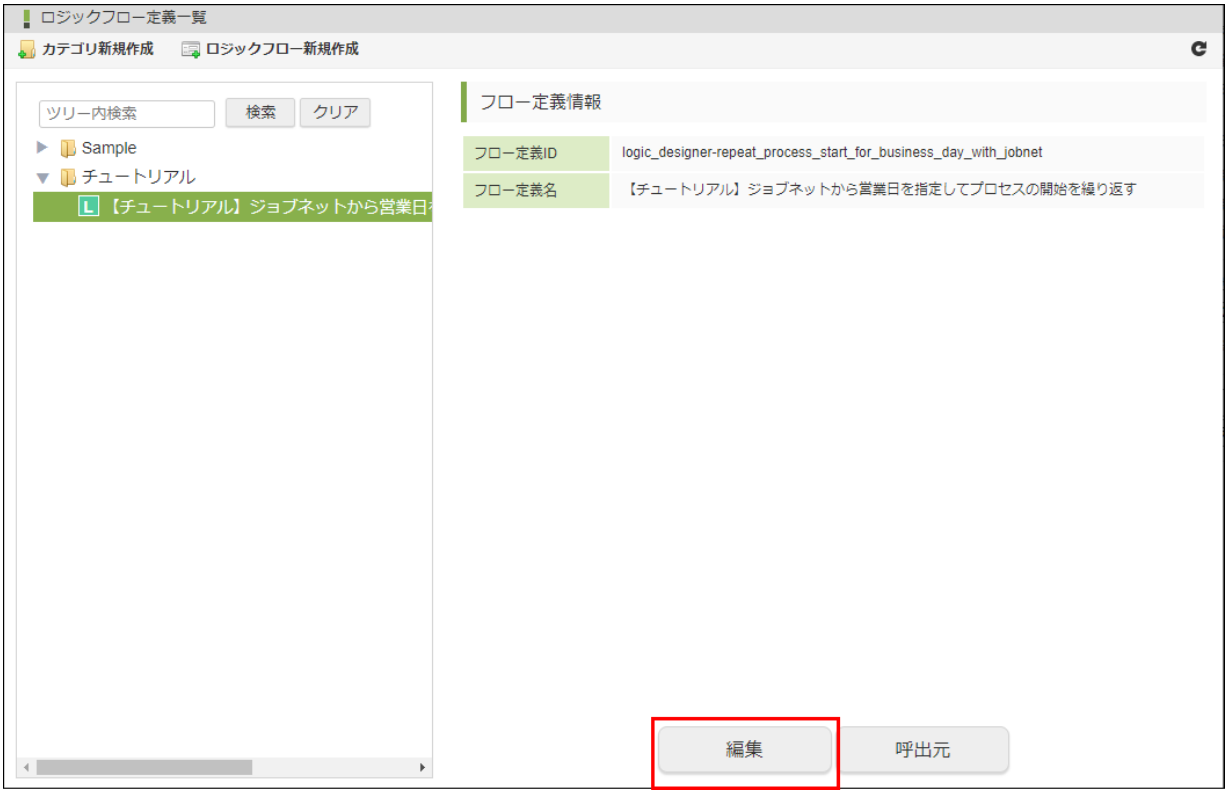


図 ：「ロジックフロー定義一覧」

3. 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。

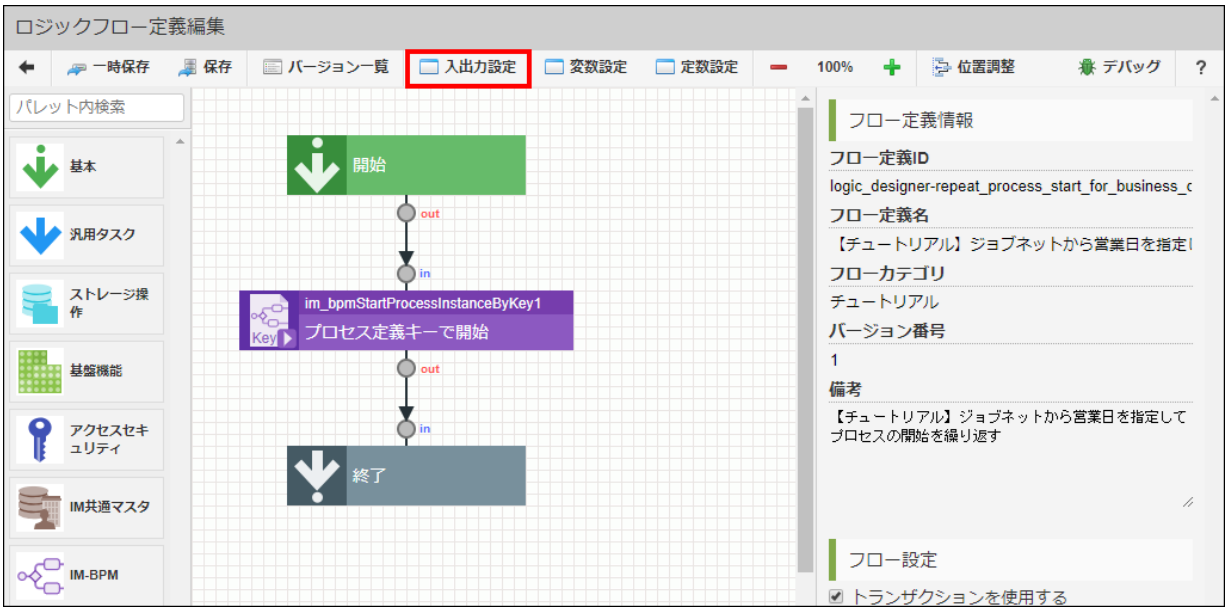


図 ：「ロジックフロー定義編集」

4. 「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」

5. 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。

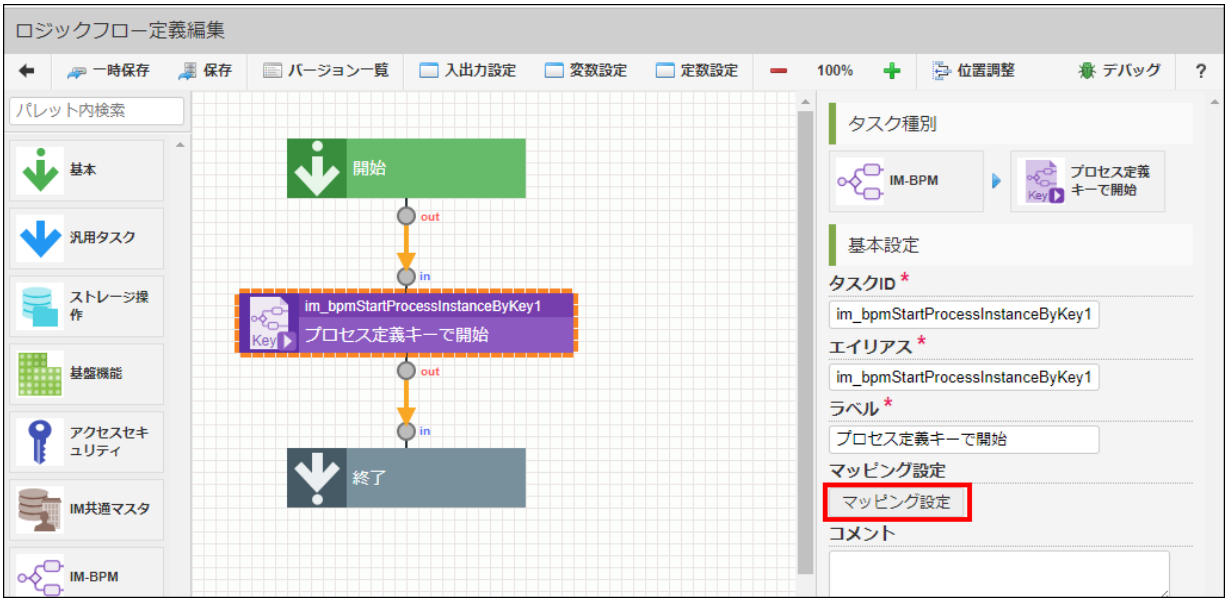


図 ： 「ロジックフロー定義編集」

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>

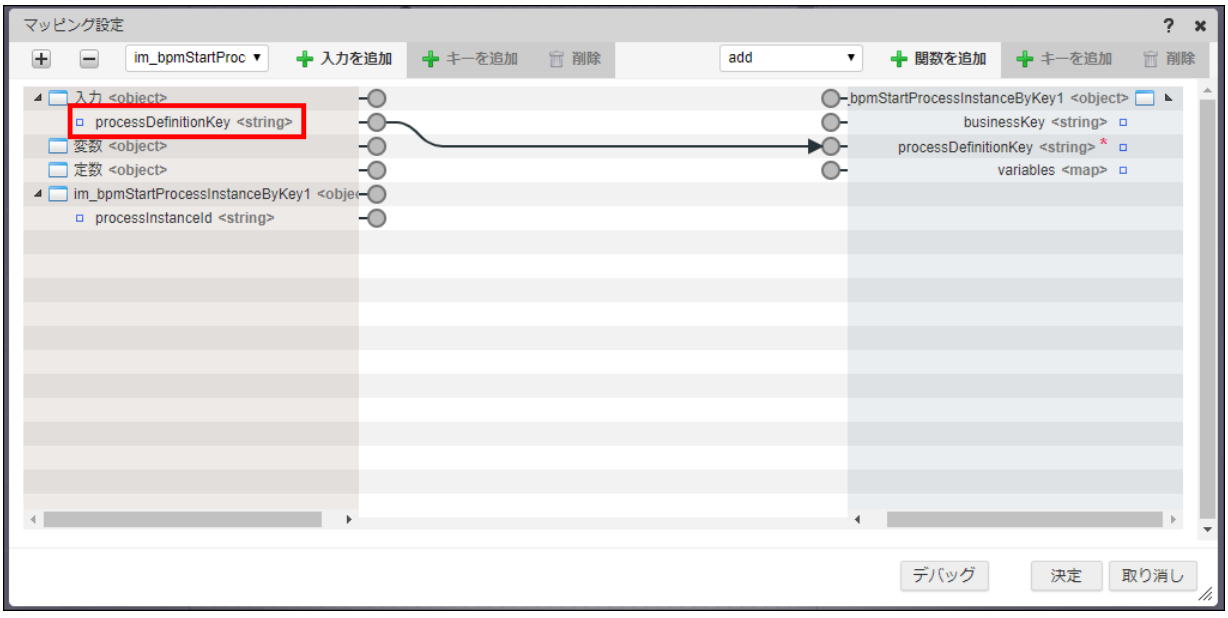


図 ： 「マッピング設定」

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。

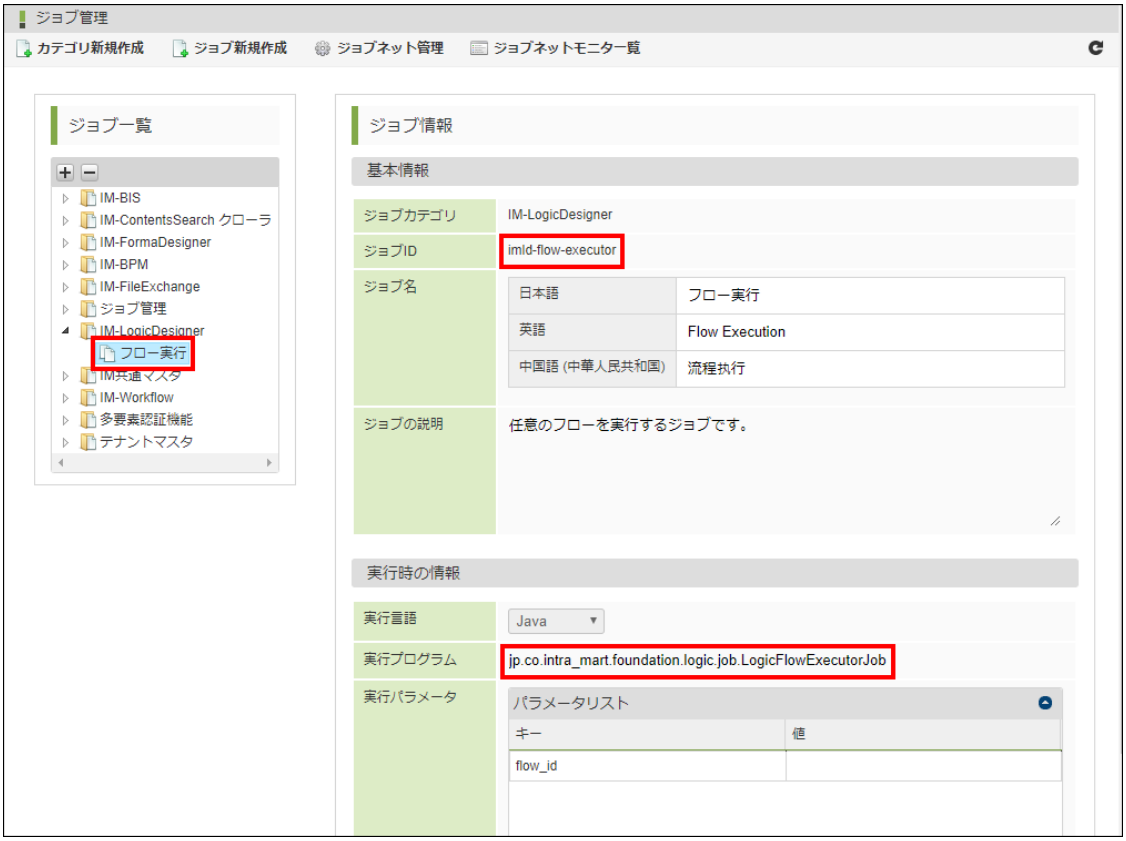


図 ： 「ジョブ管理」

> **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。

3. 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。

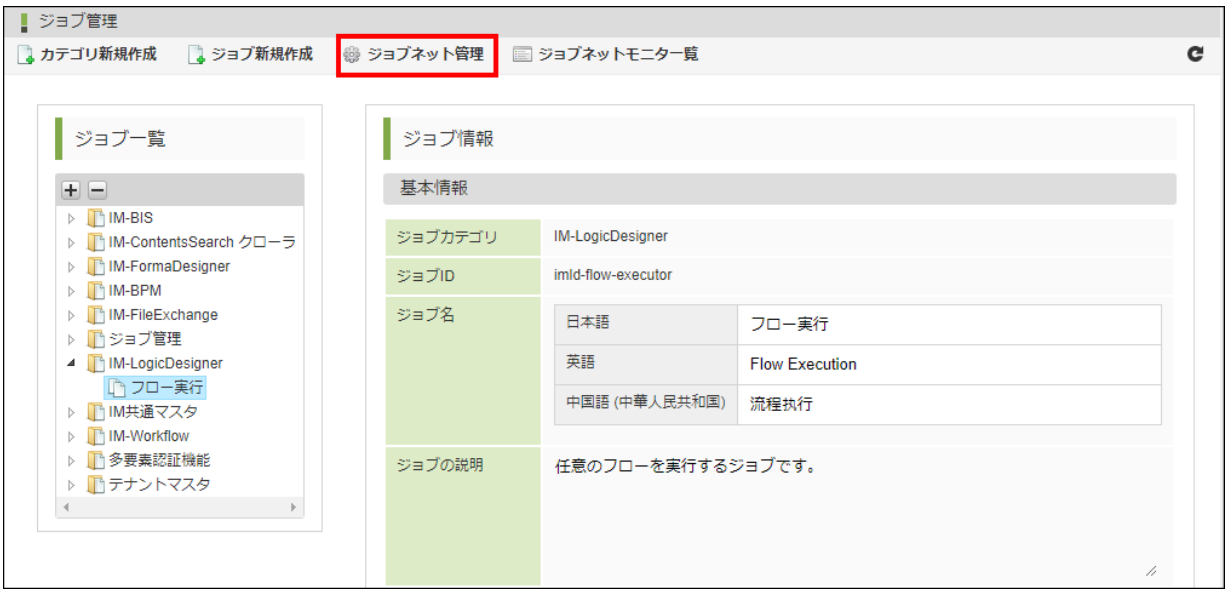


図 ： 「ジョブ管理」

4. 「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

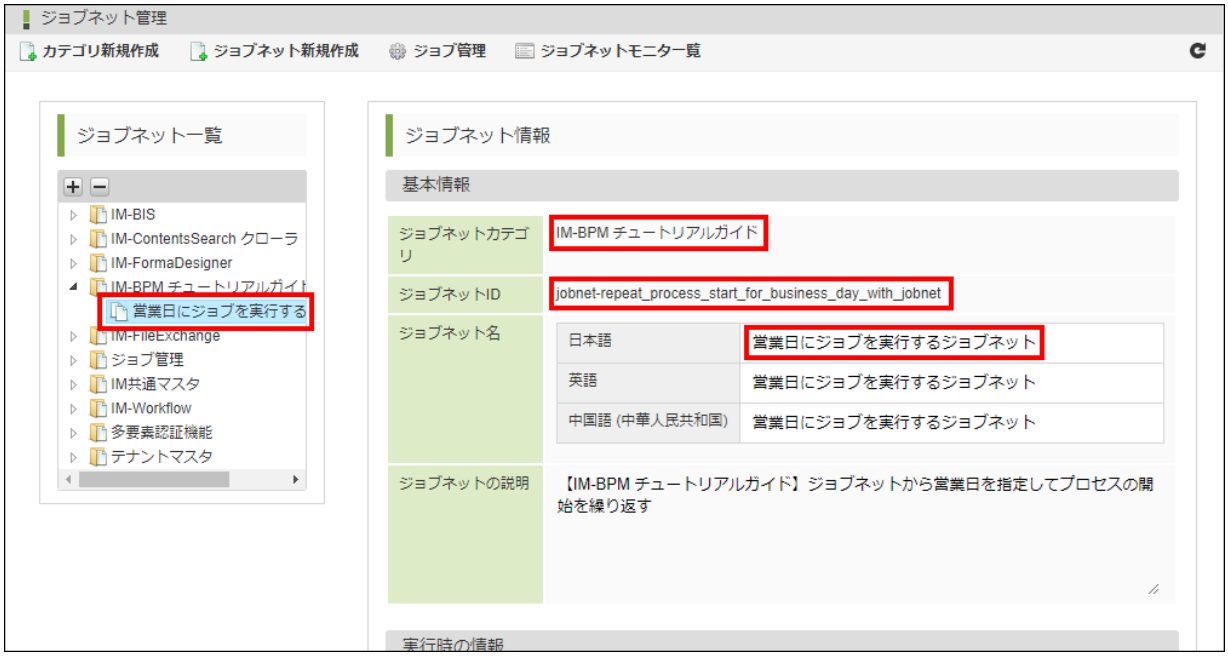


図 ： 「ジョブネット管理」 - 「基本情報」

- 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
    1. フローID
       キー ： flow_id
       値 ： logic_designer-repeat_process_start_for_business_day_with_jobnet
    2. 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「ジョブスケジューラ仕様書」 - 「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）はプロセスが開始しないため、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。

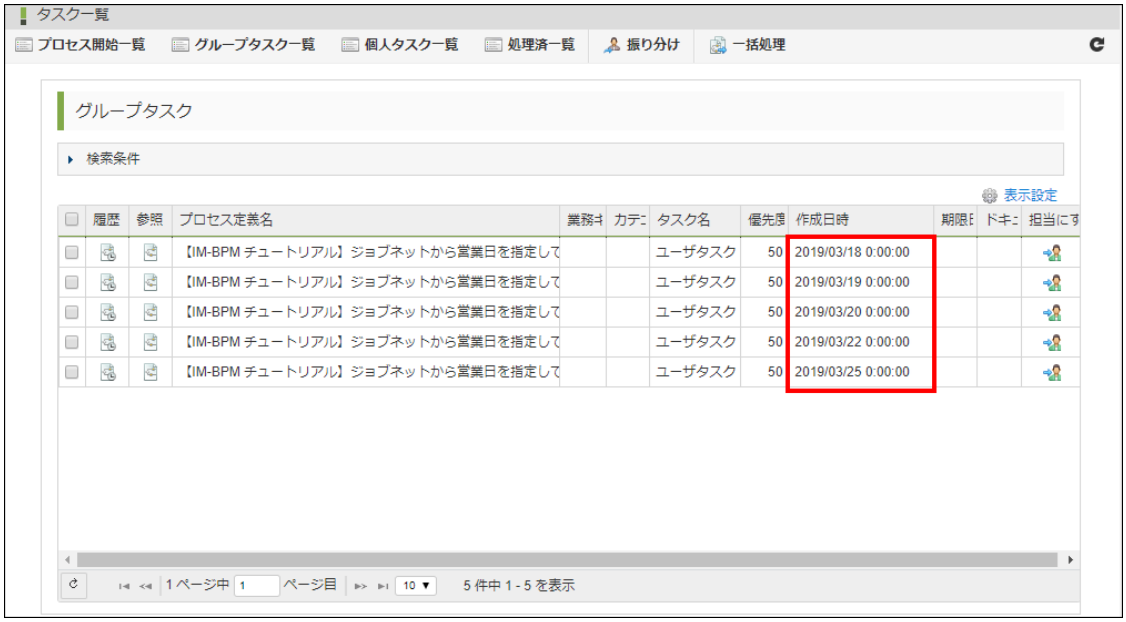


図 ： 「タスク一覧」- 「グループタスク」

差配管理

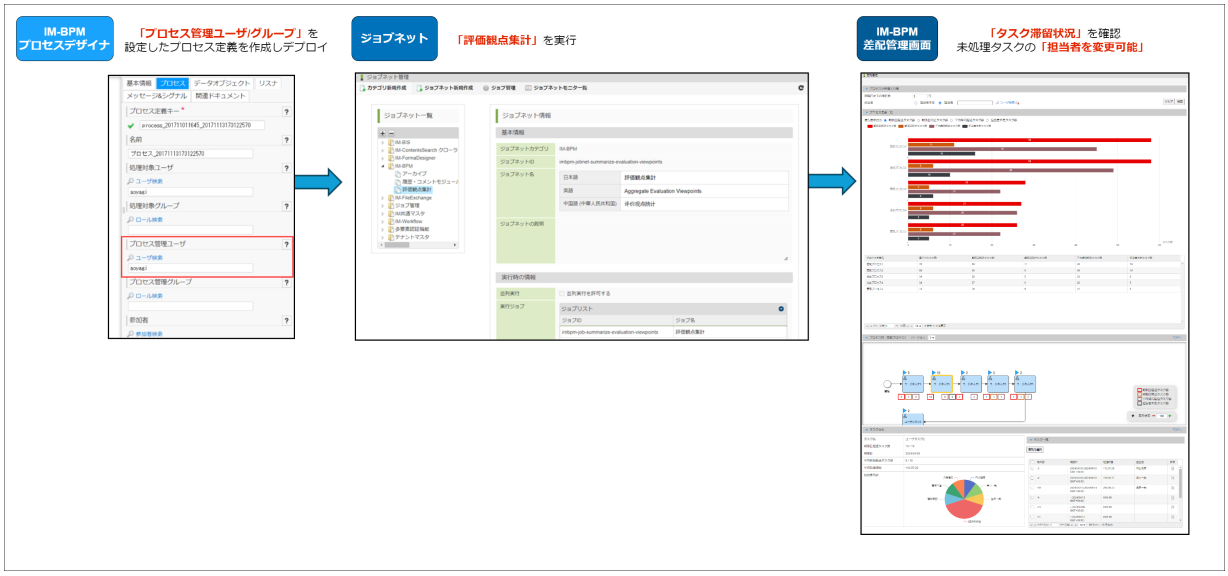
差配管理画面でプロセス内のタスク滞留状況を確認し、差配者を設定する方法について説明します。

このチュートリアルは、以下の2項目を前提条件としています。

1. *事前準備*
2. 差配機能利用の前提条件

差配管理画面

このチュートリアルでは、差配管理画面からプロセスごとのタスクの滞留状況確認と、タスクの担当者を変更する方法を解説します。



図：概要図

> **コラム**
>
> 差配管理の詳細については、「IM-BPM ユーザ操作ガイド」の「差配管理」を参照してください。

- プロセスのタスク状況を確認する
- タスクの担当者を変更する

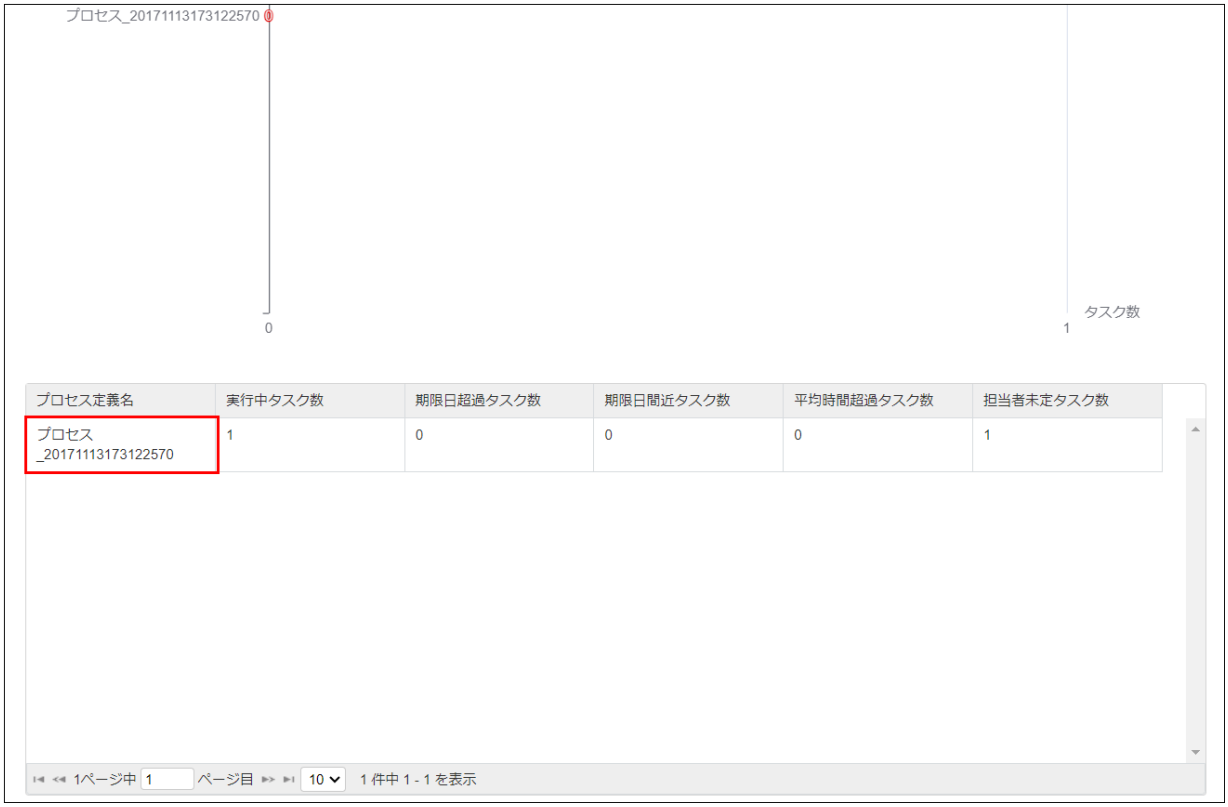
プロセスのタスク状況を確認する

プロセスが開始されたタスクの状況を差配管理画面で確認します。

1. 「サイトマップ」→「ジョブネット設定」から、「IM-BPM - 評価観点集計」のジョブを即時実行する。

   > **コラム**
   >
   > チュートリアルでは「評価観点集計」ジョブを即時実行していますが、デフォルト設定で毎日0時に実行されています。

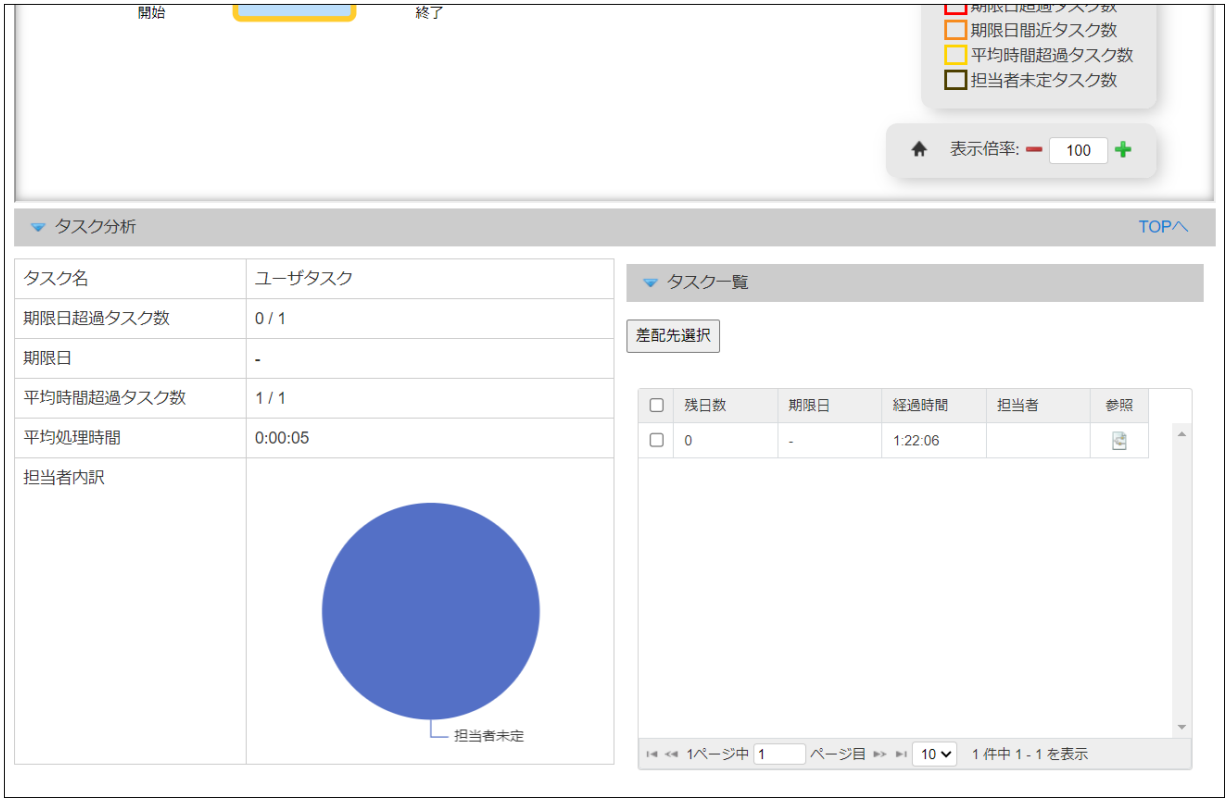
2. 「サイトマップ」→「BPM」→「差配管理」から、「差配管理」画面を表示します。
3. 「プロセス定義一覧」に表示されている「プロセス定義名」をクリックします。



図：「差配管理」-「プロセス定義一覧」

4. 表示された「プロセス図」の「ユーザタスク」をクリックします。

5. プロセスのタスク状況が確認できます。

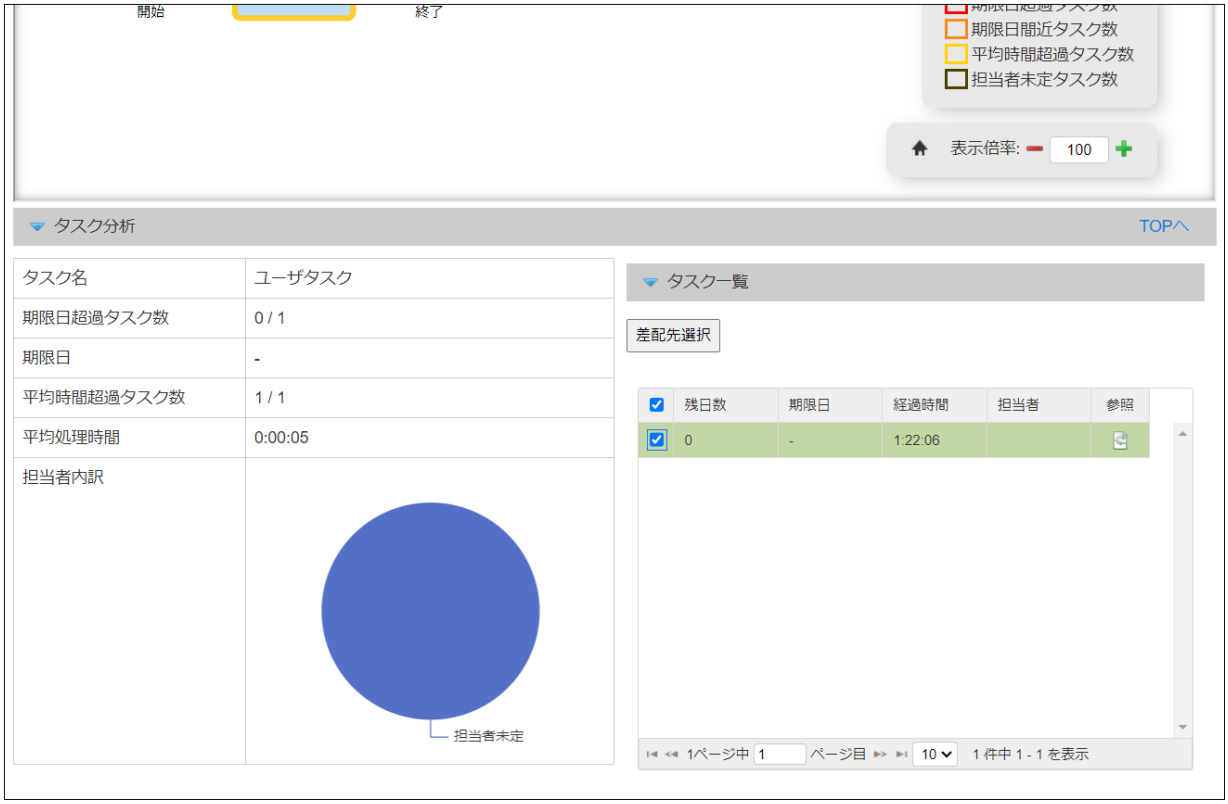


図：「差配管理」-「タスク分析」

タスクの担当者を変更する

差配管理画面で未処理タスクの担当者を変更します。

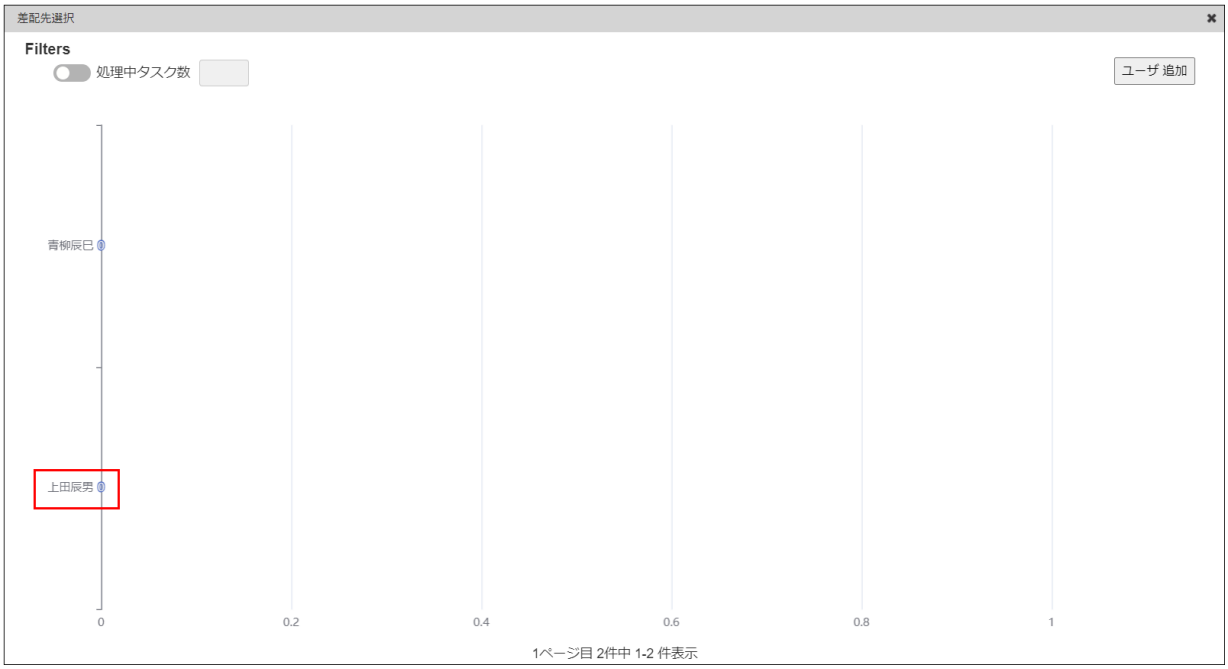
1. 表示されている「タスク一覧」から、タスクのチェックボックスをクリックします。



図：「差配管理」-「タスク分析」-「タスク一覧」

2. 「差配者選択」をクリックして、「担当者選択モーダル」を表示します。

3. 「ユーザ 追加」をクリックして、「上田辰男」を選択し「決定」をクリックします。



図：「差配管理」-「担当者選択モーダル」

4. 担当者変更先の「上田辰男」をクリックすると、確認ダイアログが表示されます。

5. 「決定」をクリックすると、タスクの担当者が変更されます。

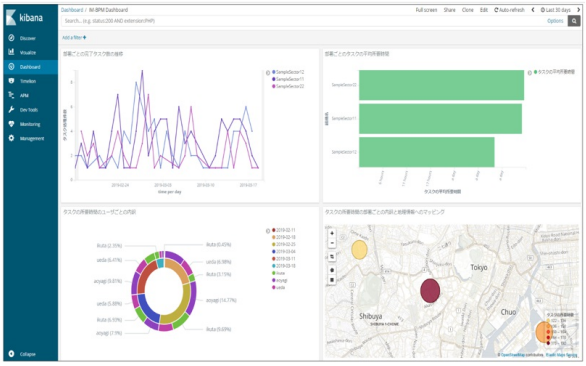
図：「差配管理」-「タスク分析」-「タスク一覧」

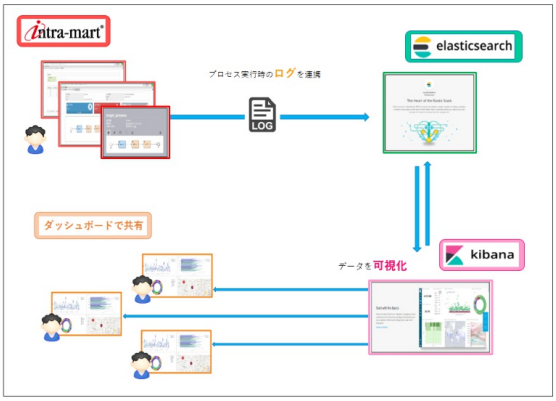
Elasticsearch/ Kibana連携機能

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
```

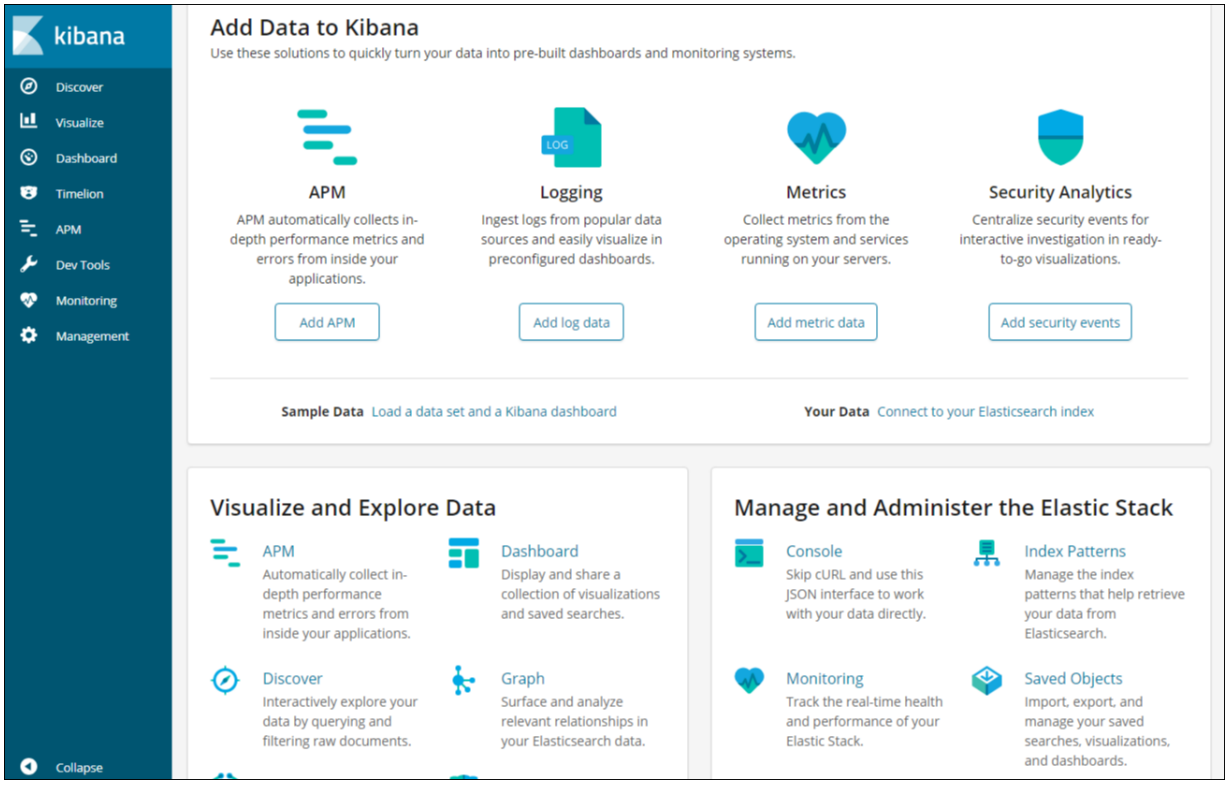
> **注意**
>
> Kibanaは標準では通信の暗号化や認証機能が使用できません。Kibanaは、セキュアなネットワーク上にて運用されることを推奨します。

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。

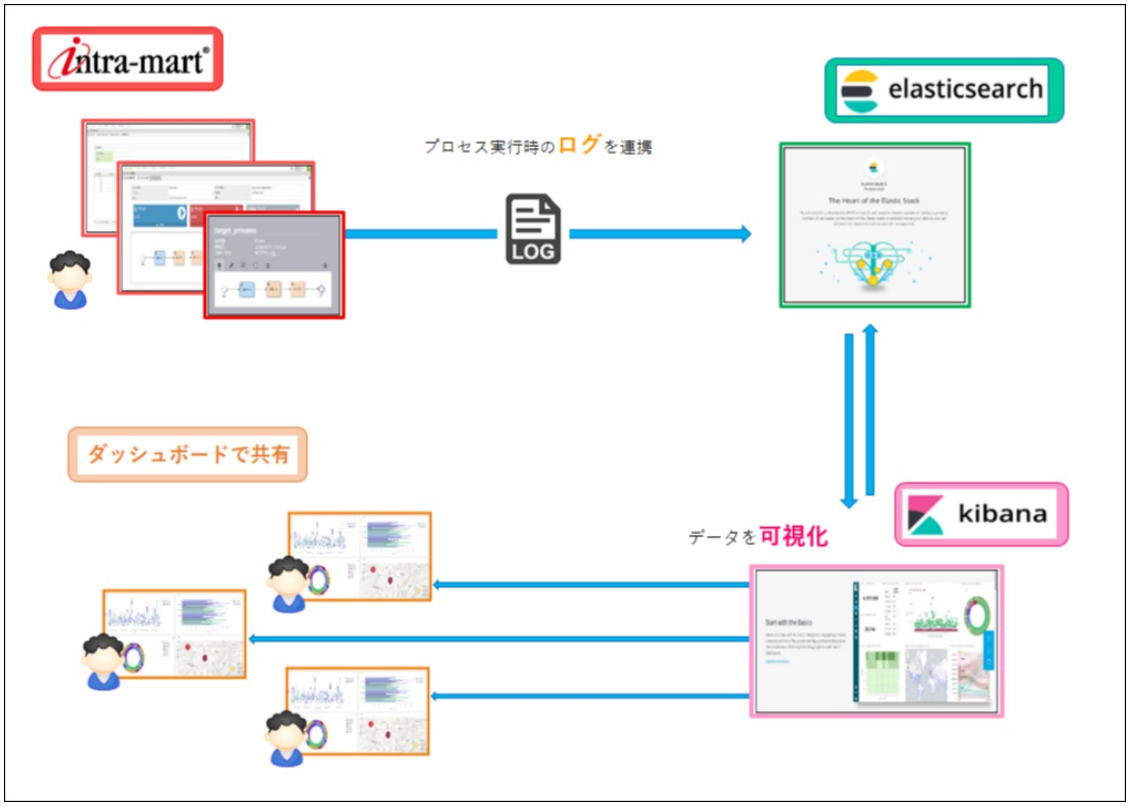
http://Kibanaのホスト名:5601/



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。



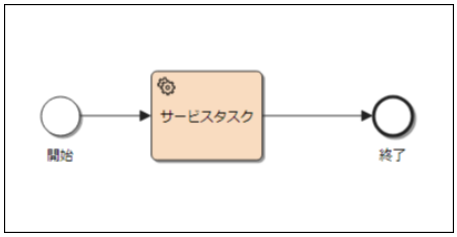
図：プロセスの実行とElasticsearchへの登録、Kibanaでの可視化

本チュートリアルで使用する時系列データを作成する

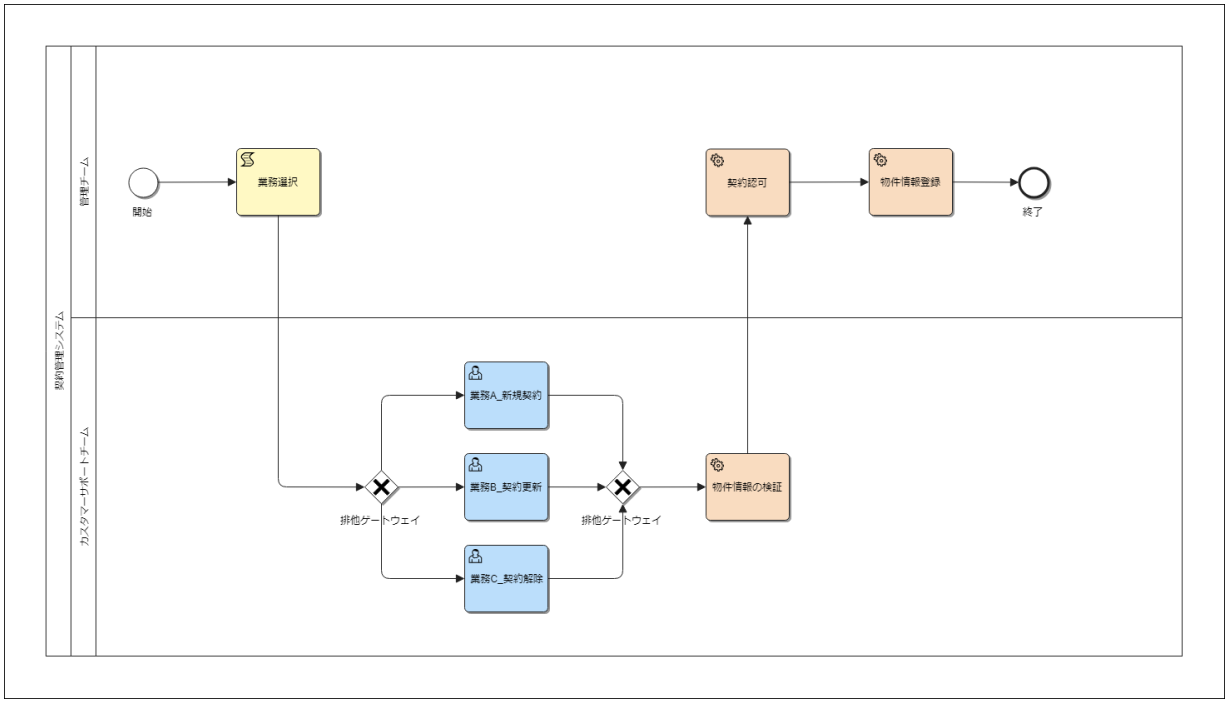
Kibana上で統計情報を可視化するにあたり、サンプルとなる時系列データを作成します。

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）

> **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

> **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。

> **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

> **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で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`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。

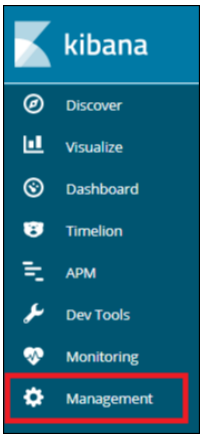
図：「プロセス開始一覧」画面

> **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

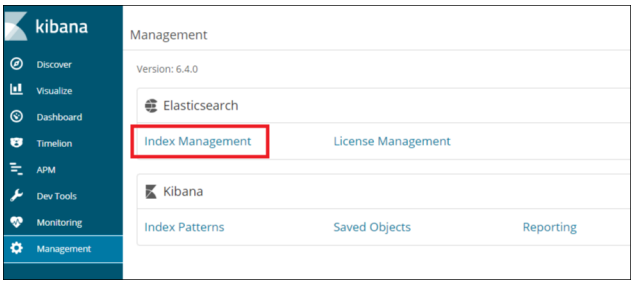
Kibana上でElasticsearchに登録されたデータを確認します。

1. Kibanaの「メインメニュー」→「Management」をクリックします。

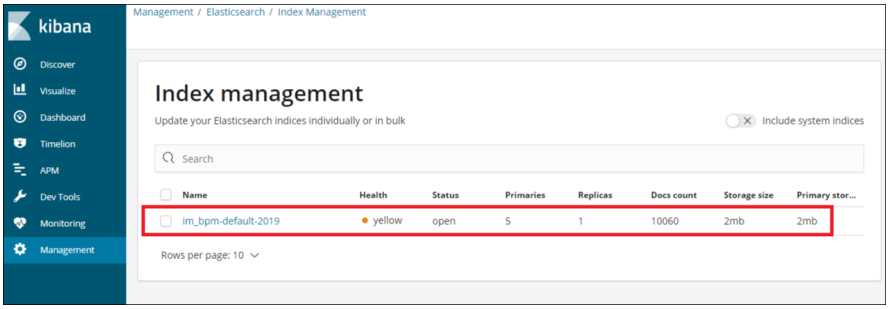


図：「Management」

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」

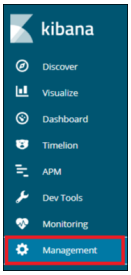


図：インデックス「im_bpm-default-YYYY」

Kibana上にインデックスパターンを作成する

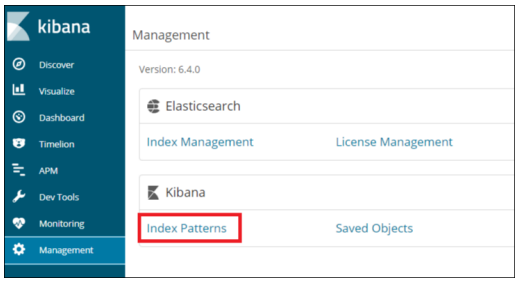
Kibana上にインデックスパターンを作成し、Elasticsearchのインデックスのフィールドを扱う際のルールの設定を行います。

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。

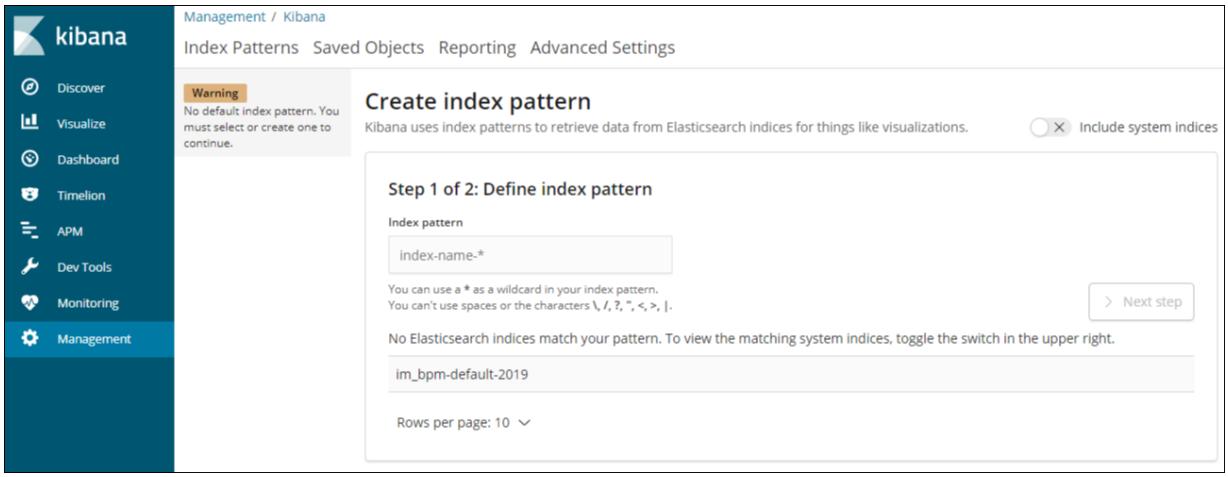


図：「Management」

2. 「Management」画面の「Kibana」→「Index Patterns」をクリックし、「Create index pattern」画面を開きます。



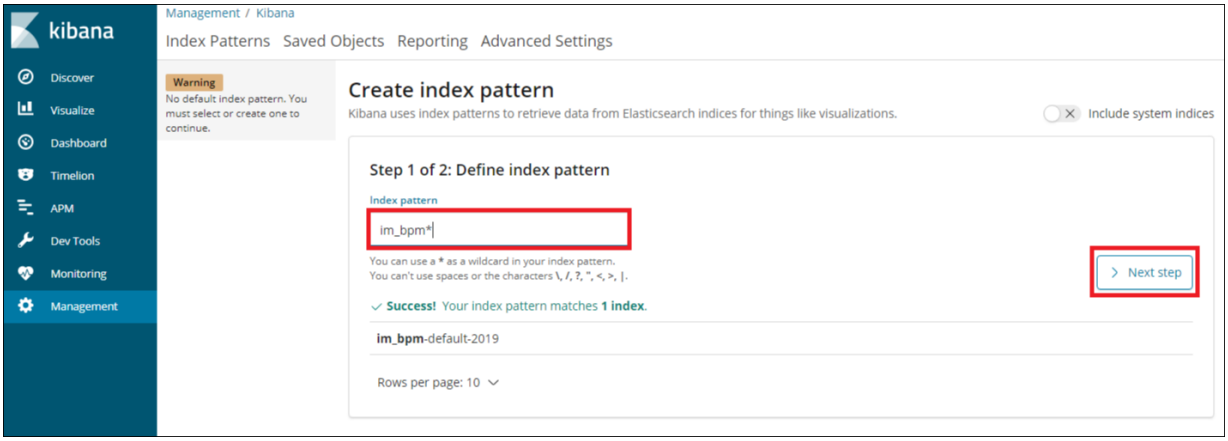
図：「Kibana」→「Index Patterns」



図：「Create index pattern」画面

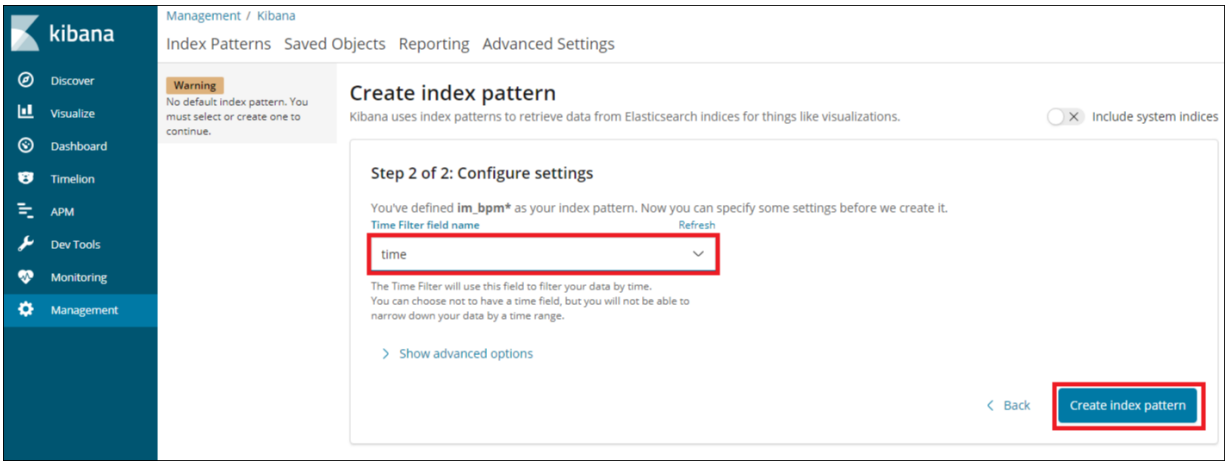
3. 「Create index pattern」画面の「Index pattern」にim_bpm*を指定し、「Next step」をクリックします。

項目名	設定値
Index pattern	im_bpm*



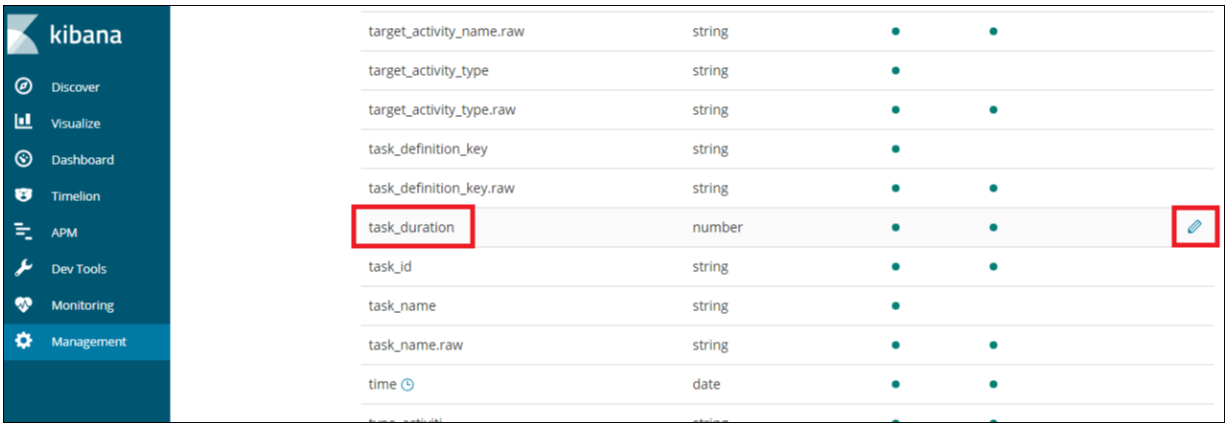
図：「Next step」をクリック

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

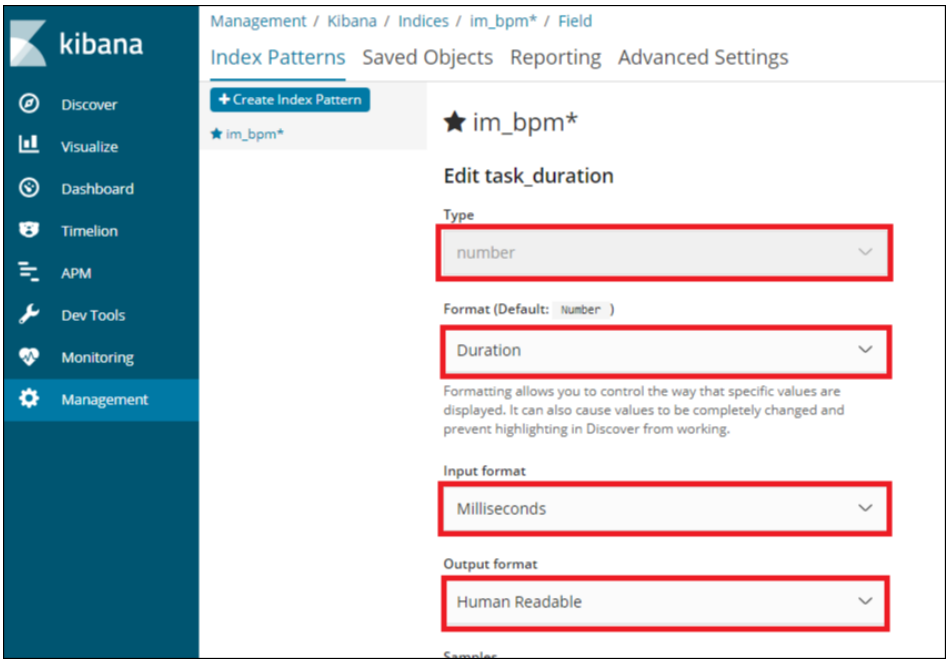
5. インデックスパターン「im_bpm*」の作成が完了し、フィールドの一覧が表示されたら、フィールド「task_duration」（タスクの所要時間）の「edit」アイコンをクリックします。



図：「task_duration」-「edit」

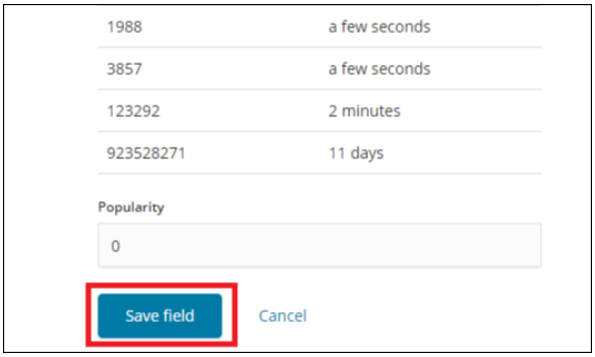
6. 「Edit task_duration」画面の各項目に下記の値を設定します。

項目名	設定値
Type	number（変更不可）
Format (Default: Number)	Duration
Input format	Milliseconds
Output format	Human Readable



図：「Edit task_duration」画面

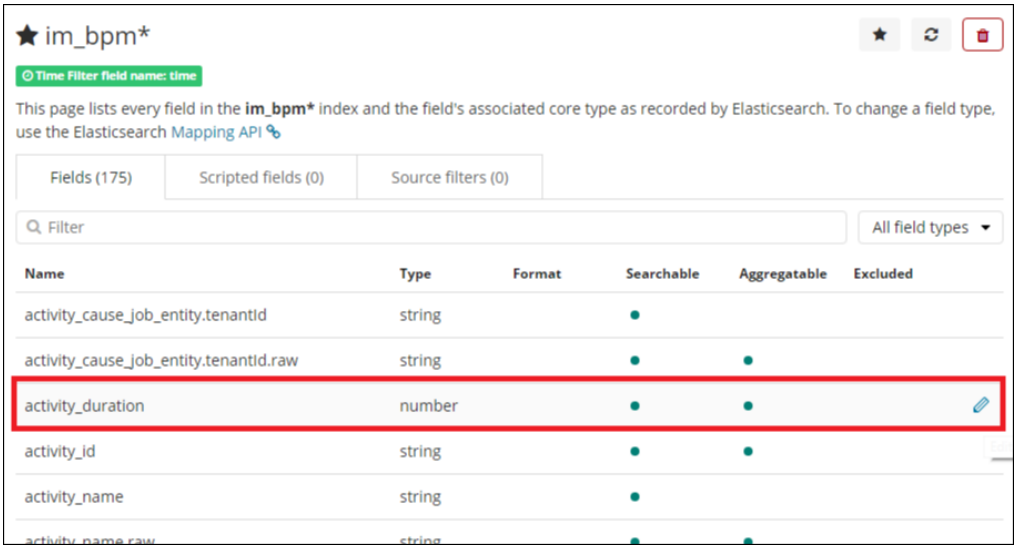
7. 「Save field」をクリックし、変更を保存します。



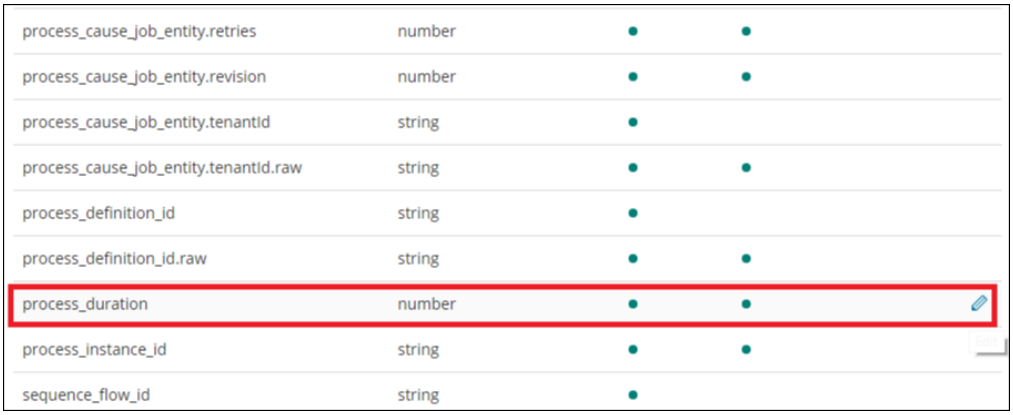
図：「Save field」をクリック

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）に対して行います。



図：フィールド「activity_duration」



図：フィールド「process_duration」

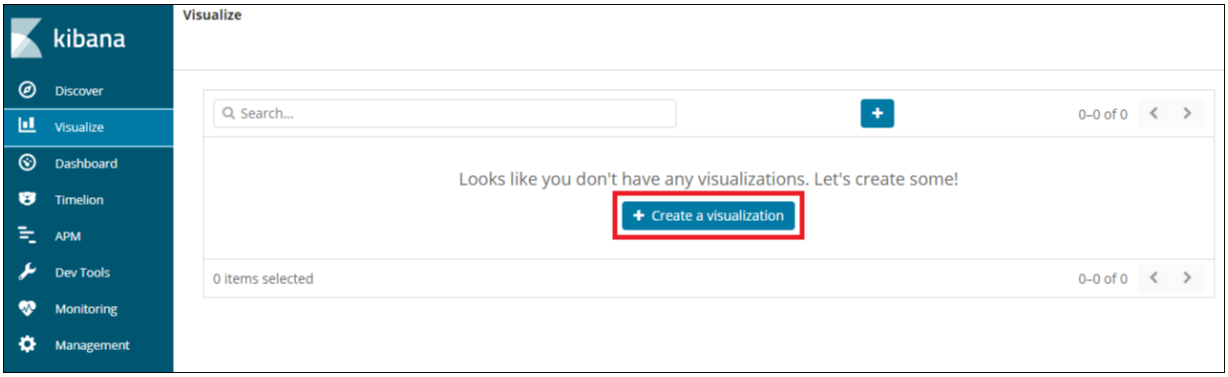
タスクの変数を使用して部署ごとの統計情報のグラフを作成する

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

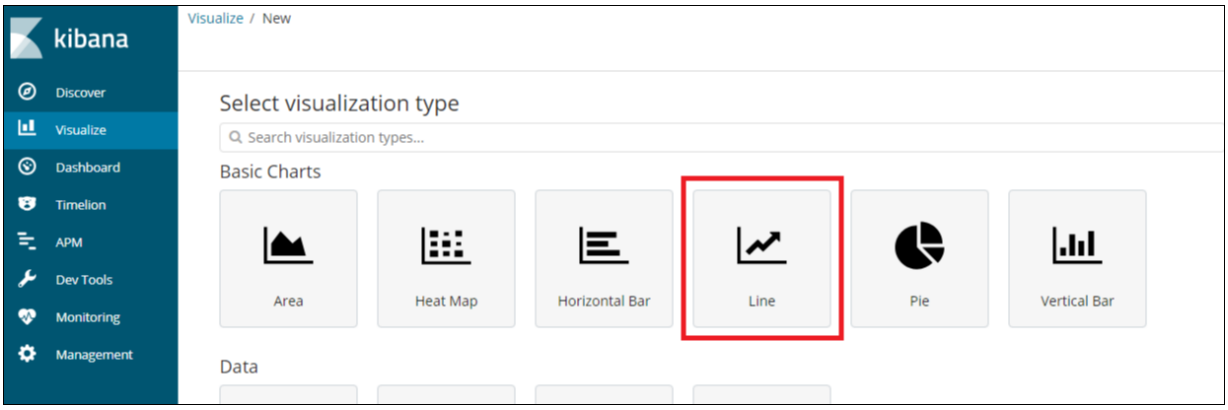
部署（処理したユーザの所属組織）ごとの、過去30日間の完了タスク数の推移のグラフを作成します。

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。



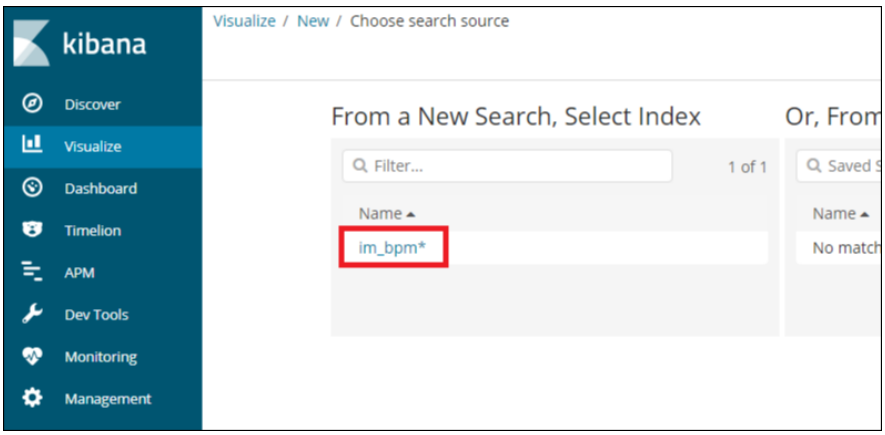
図：「Visualize」画面

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。



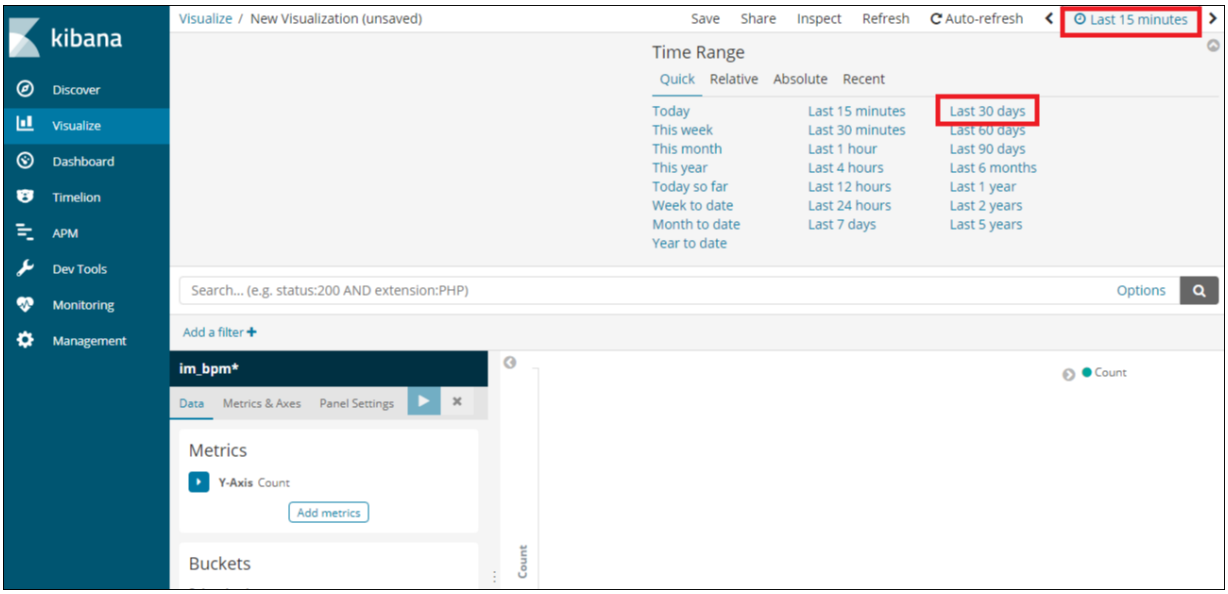
図：「Line」

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

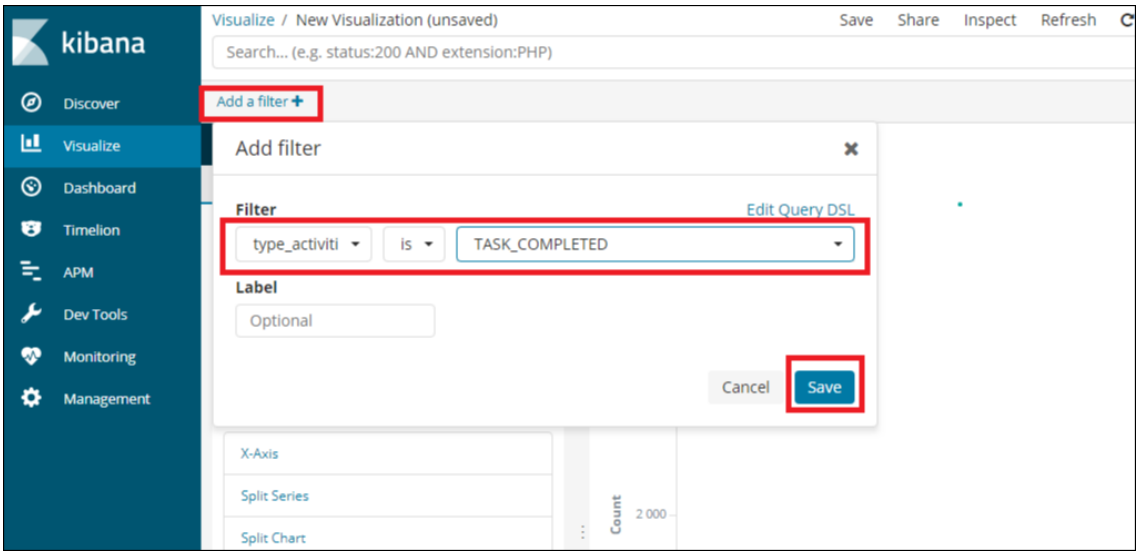
4. 集計対象の期間を指定します。
「Time Range」欄より「Last 30 days」を選択します。デフォルトは「Last 15 minutes」が表示されています。



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

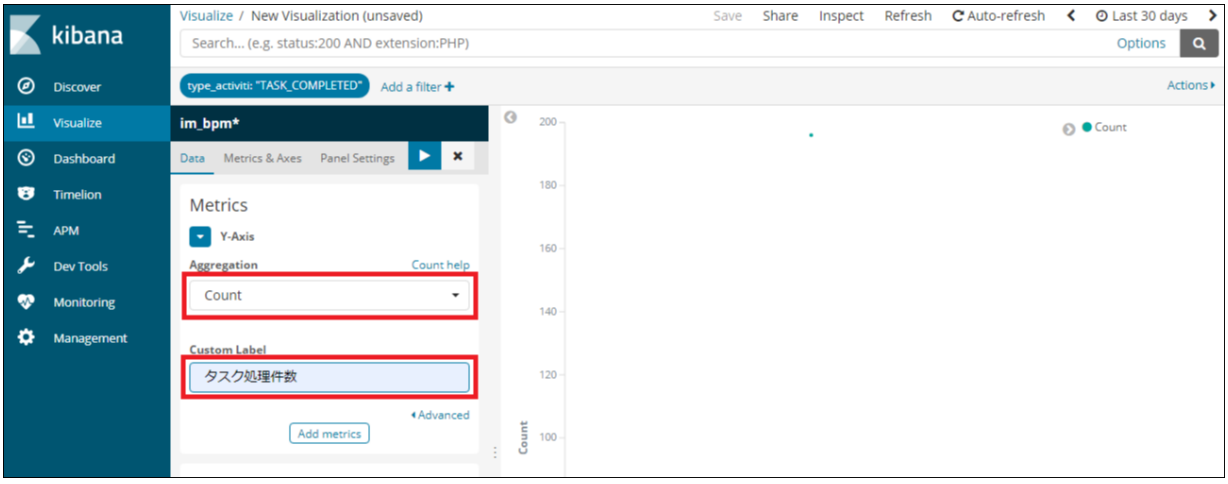


図：「Add a filter」

> **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

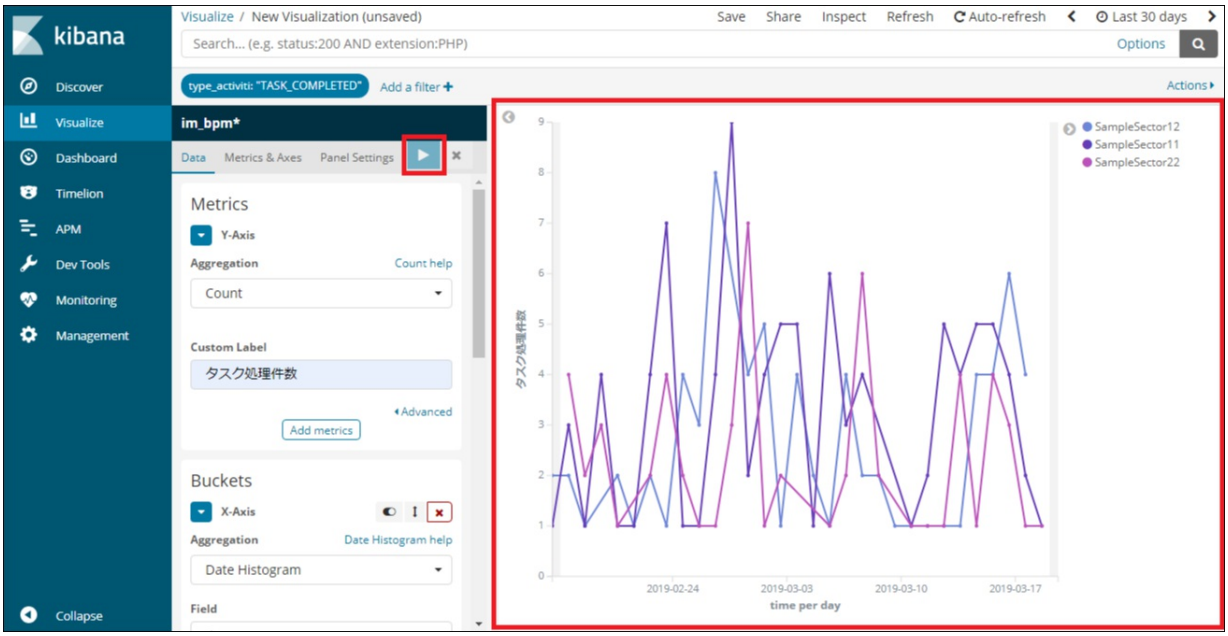
図：「X-Axis」

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」

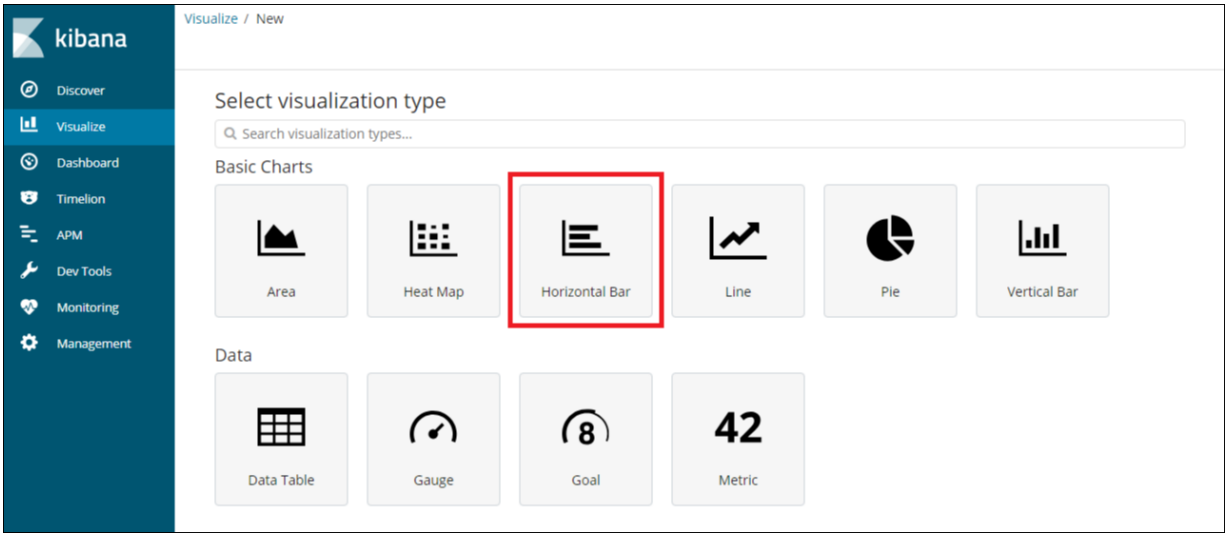
部署ごとのタスクの平均所要時間

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。

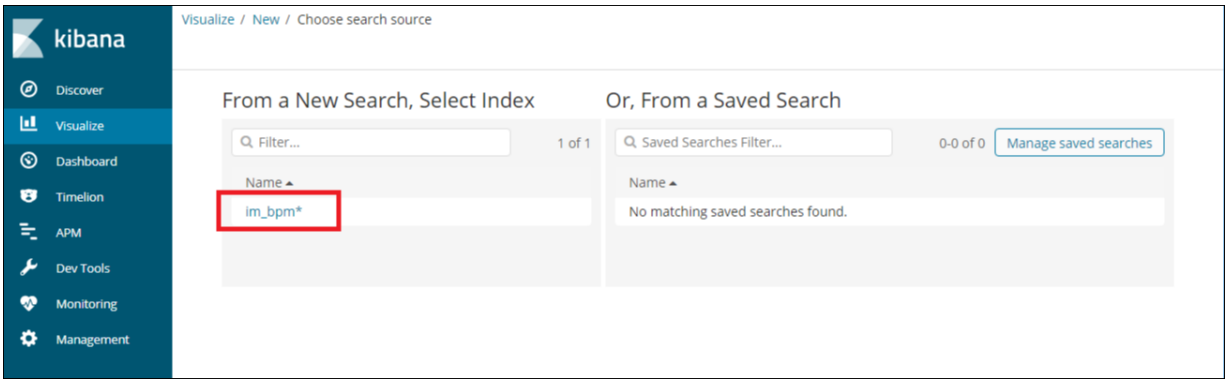
図：「Visualize」画面

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。



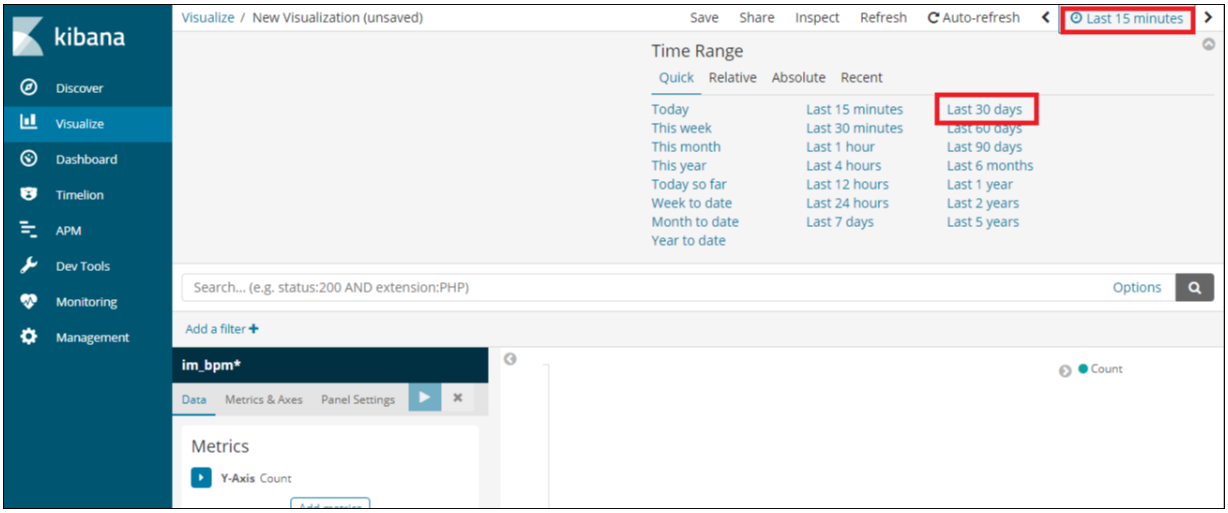
図：「Horizontal Bar」

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

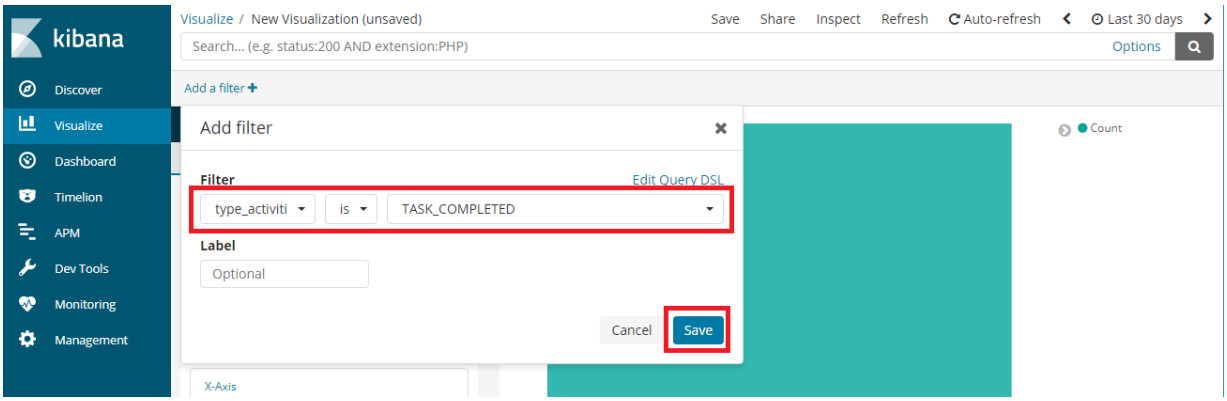
4. 集計対象の期間を指定します。
「Time Range」欄より「Last 30 days」を選択します。



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



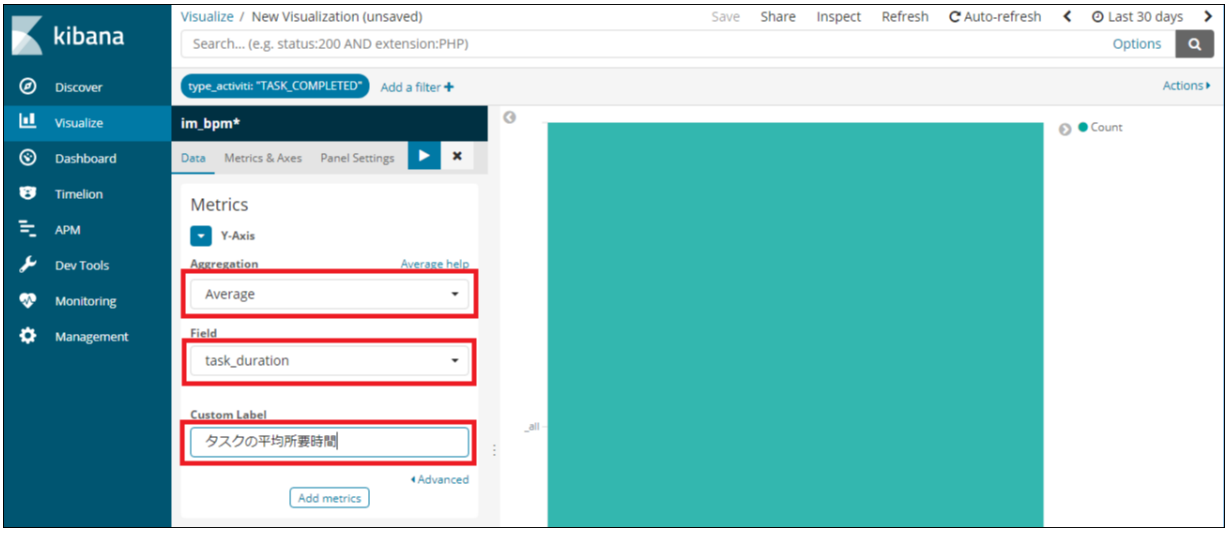
図：「Add a filter」

> **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

平均所要時間」を指定します。

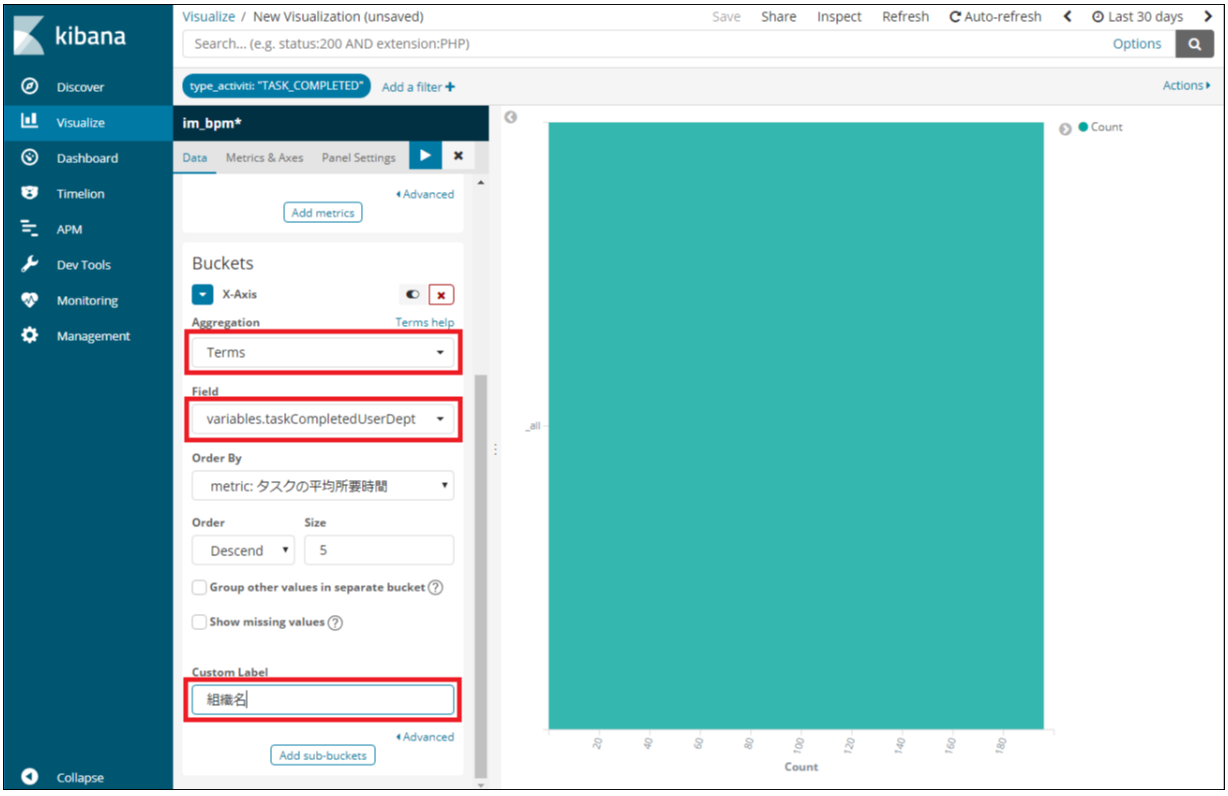
項目名	設定値
Aggregation	Average
Field	task_duration
Custom Label	タスクの平均所要時間



図：「Y-Axis」

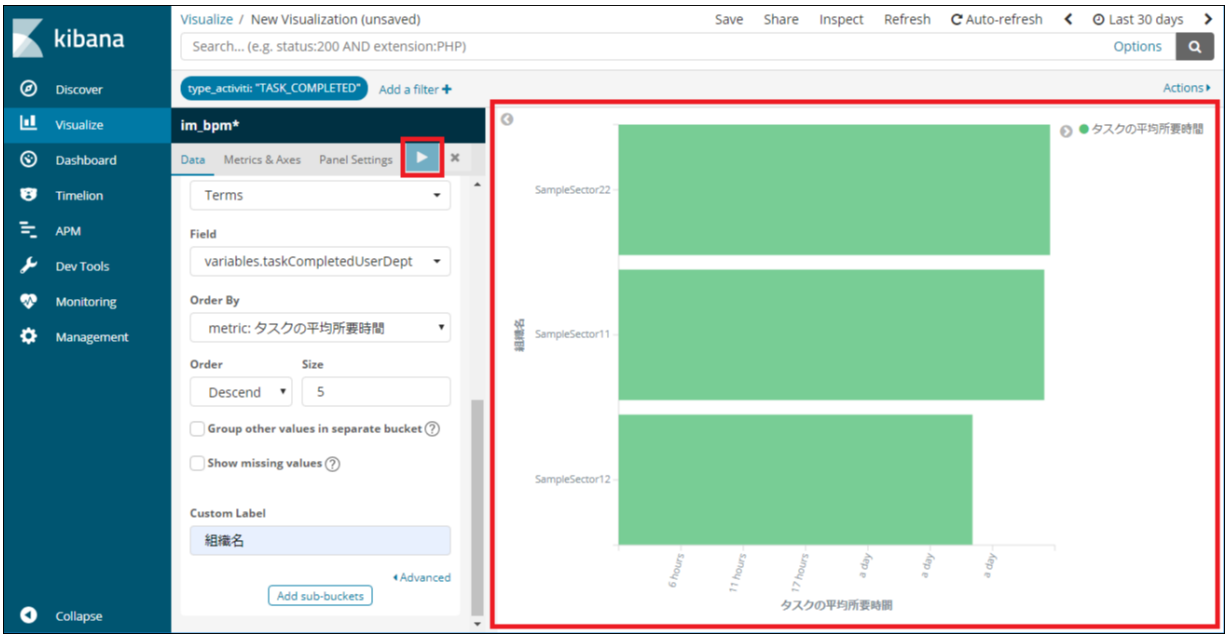
7. グラフの縦軸を指定します。
「Buckets」の「X-Axis」をクリックし、下記の値を設定します。

項目名	設定値
Aggregation	Terms
Field	variables.taskCompletedUserDept
Custom Label	組織名



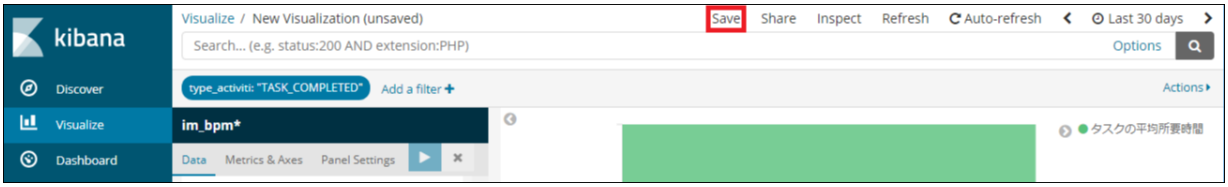
図：「X-Axis」

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」

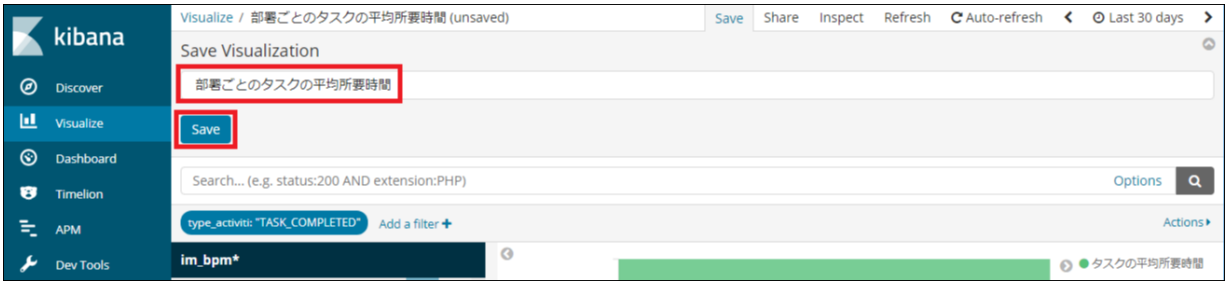
9. ヘッダメニューの「Save」をクリックします。



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

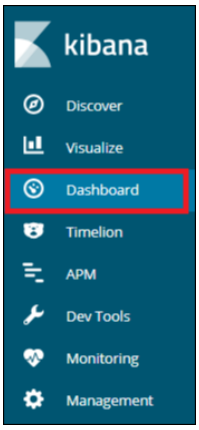


図：「Save Visualize」

Kibana上にダッシュボードを作成する

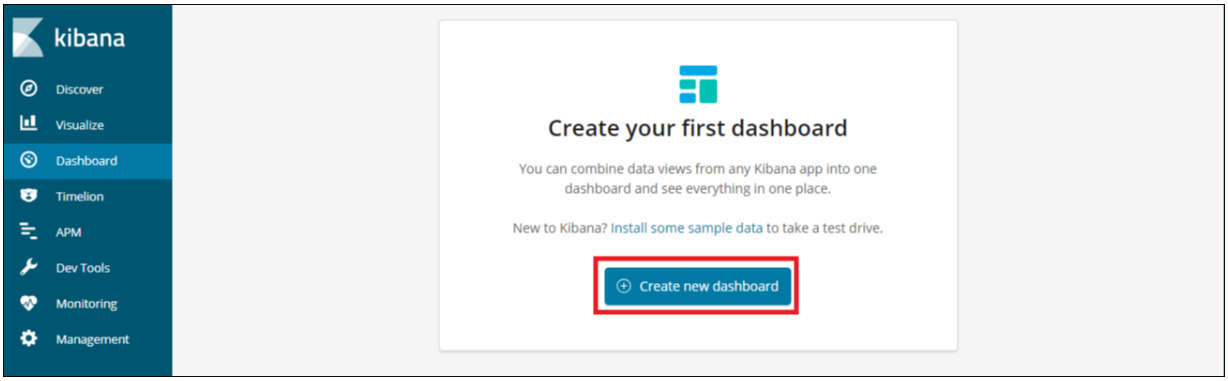
Kibana上に上記で作成した統計情報を表示するダッシュボードを作成します。

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。

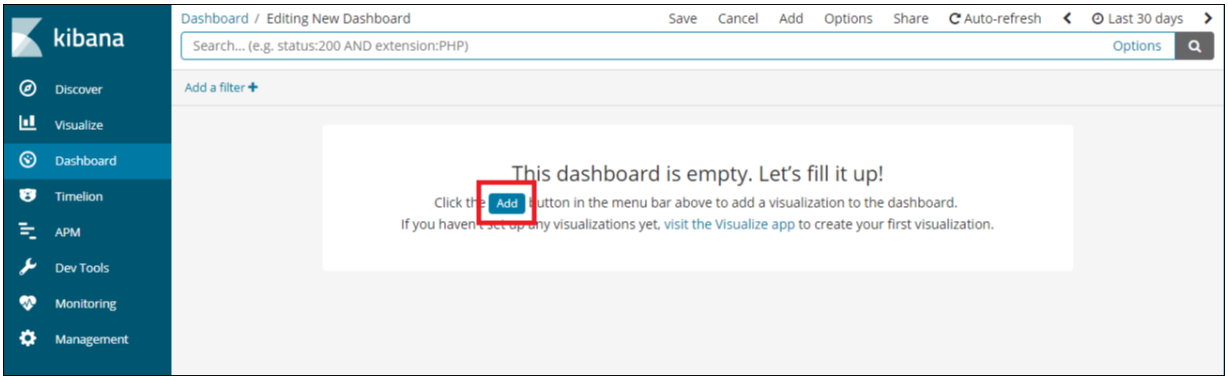


図：「Dashboard」

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。

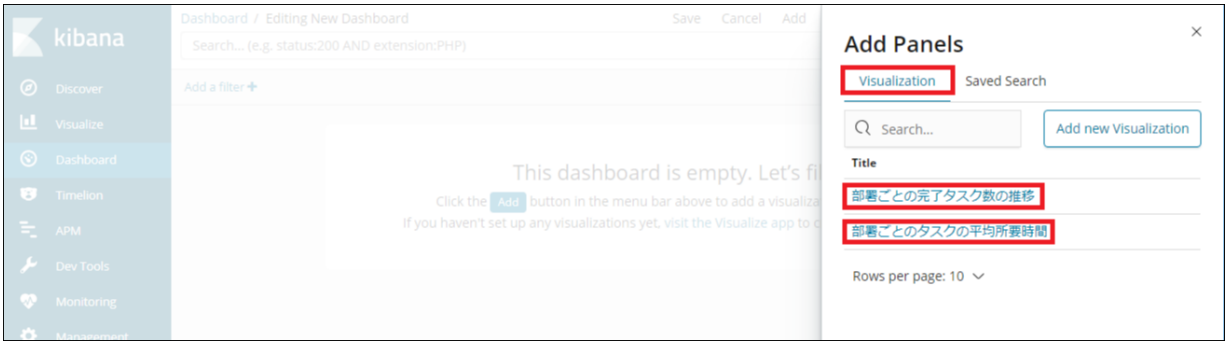


図：「Create new dashboard」



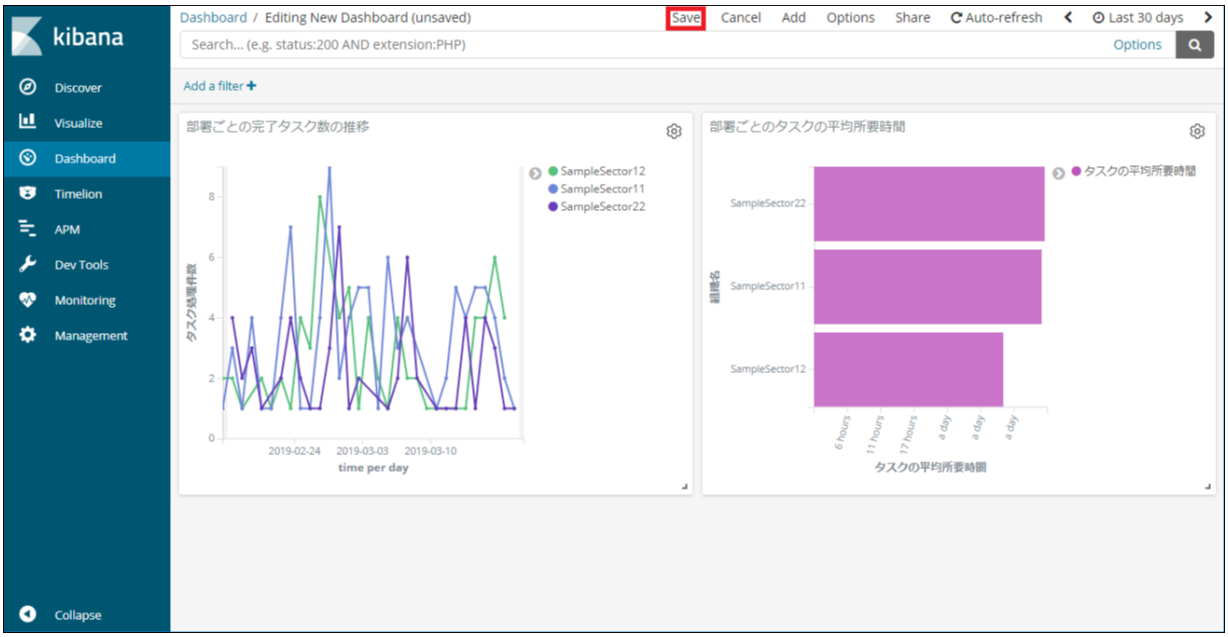
図：「Editing New Dashboard」→「Add」

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。



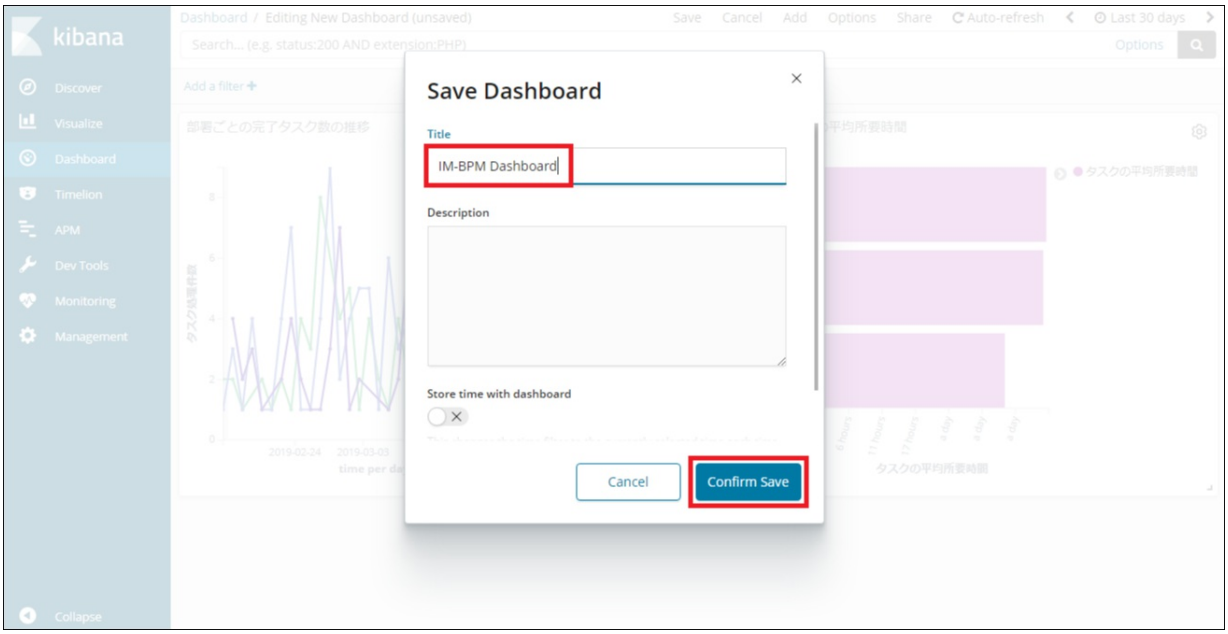
図：「Add Panels」→「Visualization」

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。



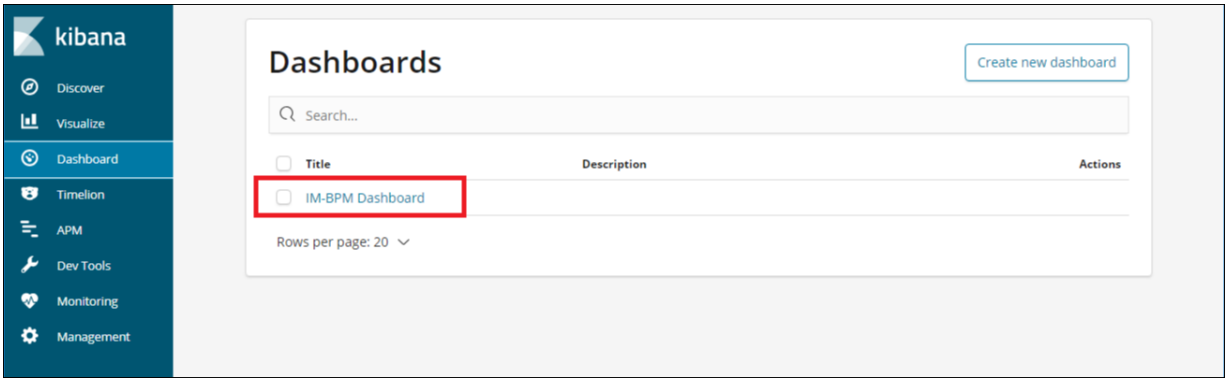
図：「Save」

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

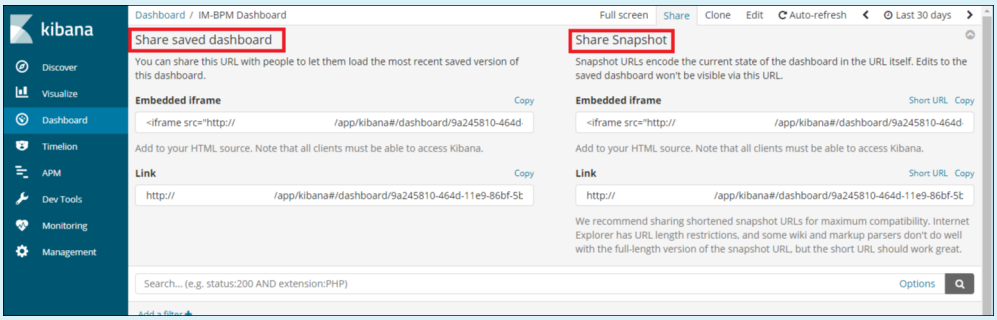
ダッシュボードの共有について

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。

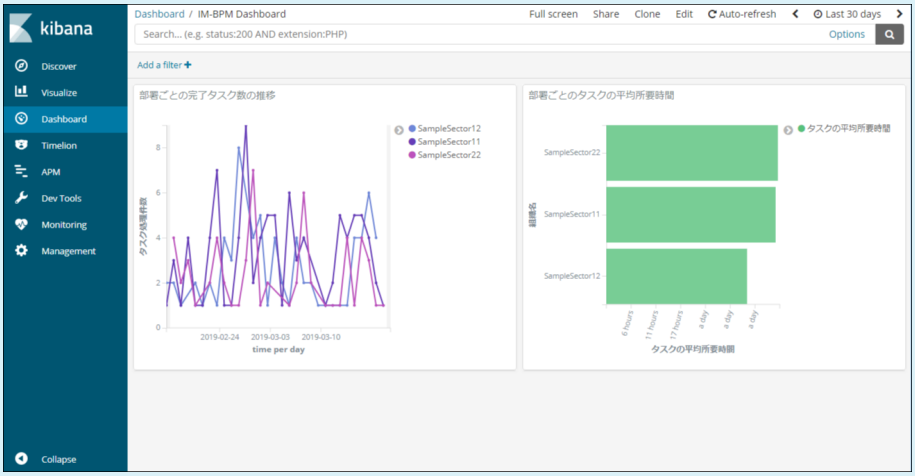
ダッシュボードを開いた状態で、右上の「Share」をクリックし、表示される各種の情報を共有してください。

「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」

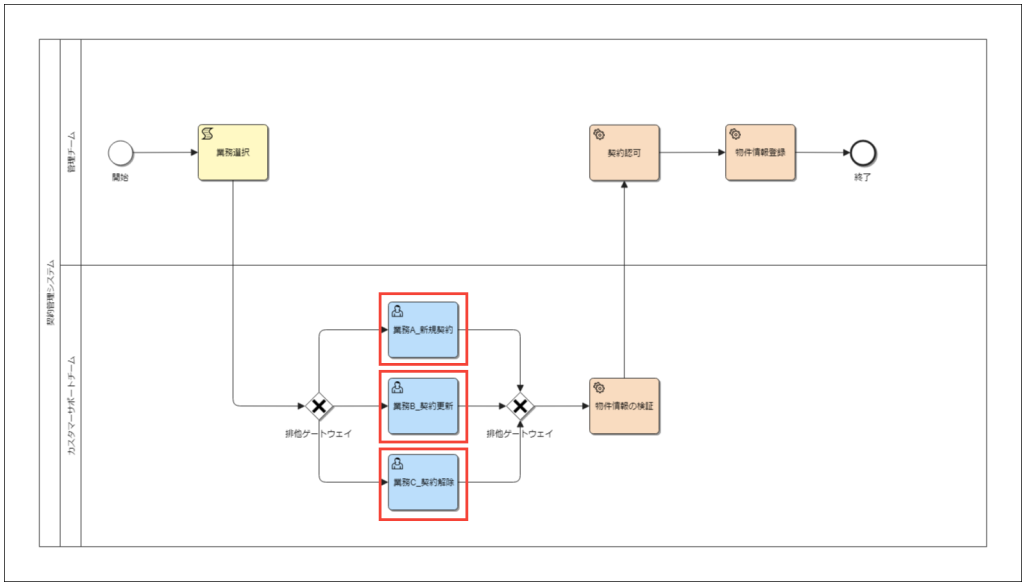


図：Kibanaの「ダッシュボード」画面

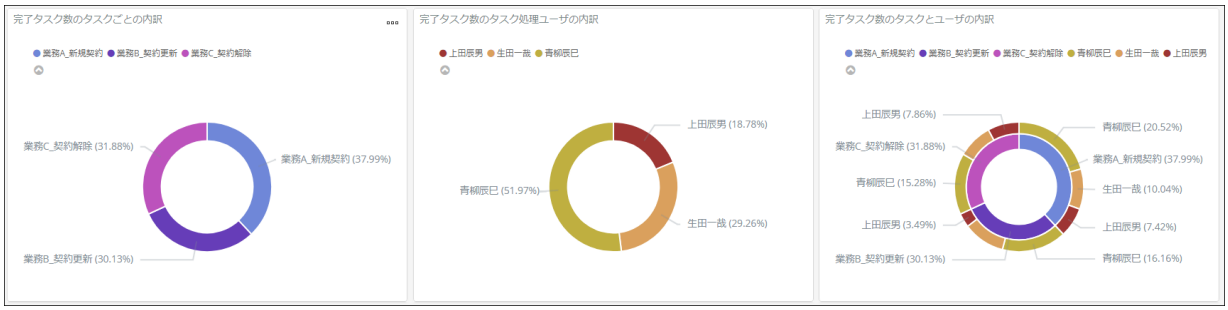
円グラフを使用して情報の内訳を表示する

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」

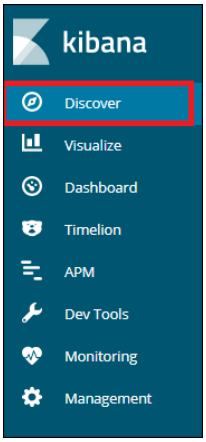
> **注意**
>
> 本チュートリアルは、事前に「*IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する*」の実施が完了していることを前提とします。

- 「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

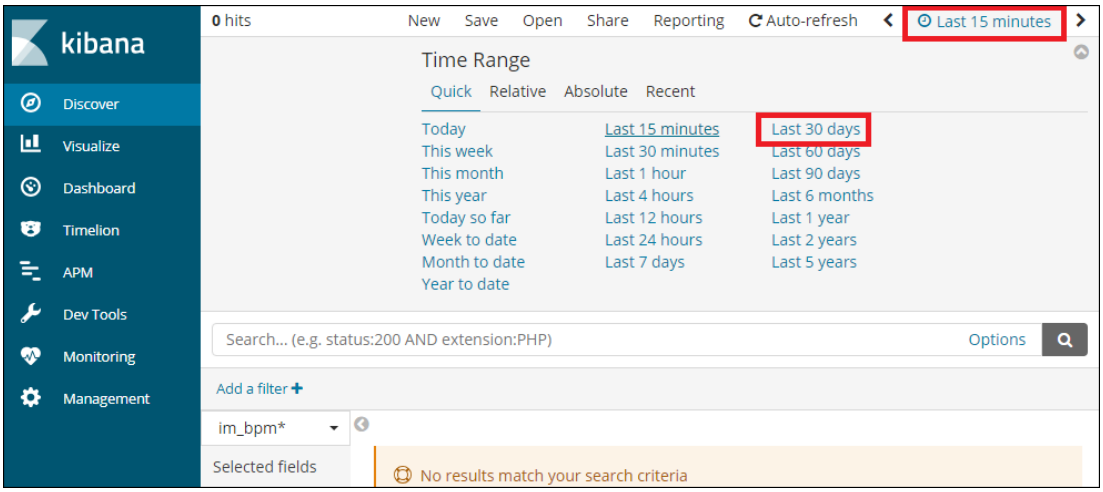
あらかじめフィルタ条件が設定されたデータセットを作成します。

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」

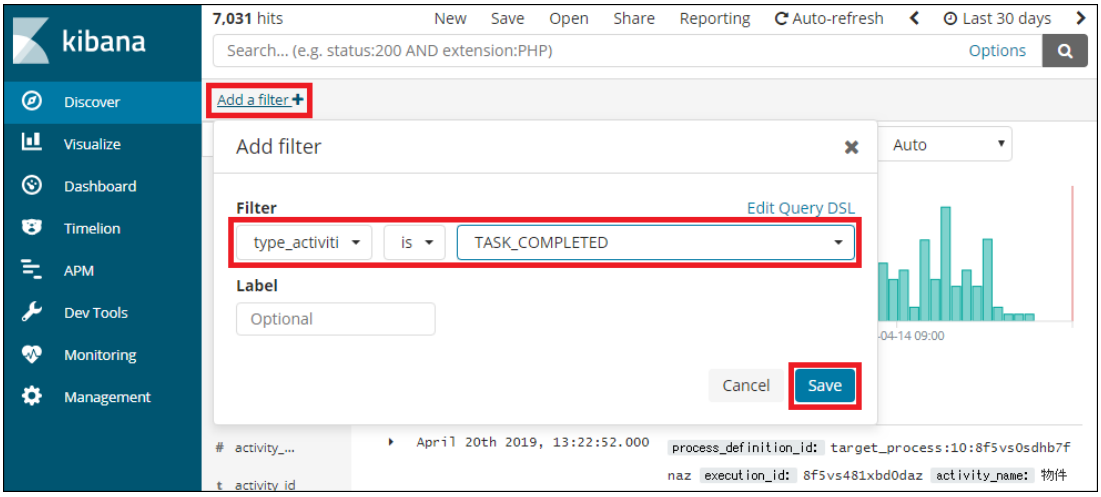
2. 集計対象の期間を指定します。
「Time Range」欄より「Last 30 days」を選択します。



図：「Time Range」の選択

3. 集計対象のイベントとして「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

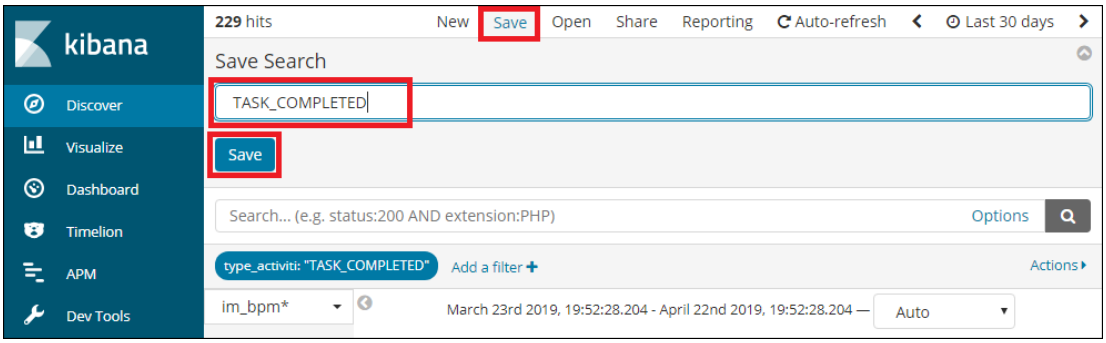


図：「Add a filter」

> **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

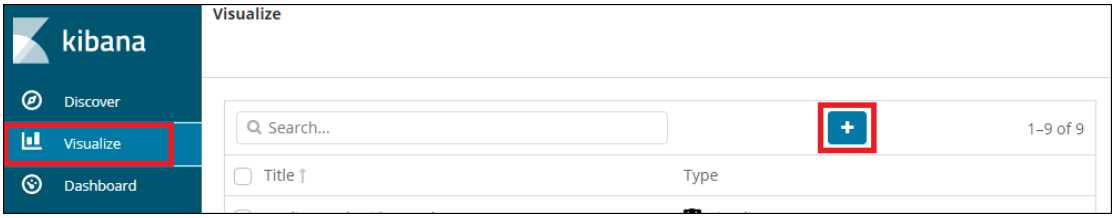


図：「Save Search」

「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

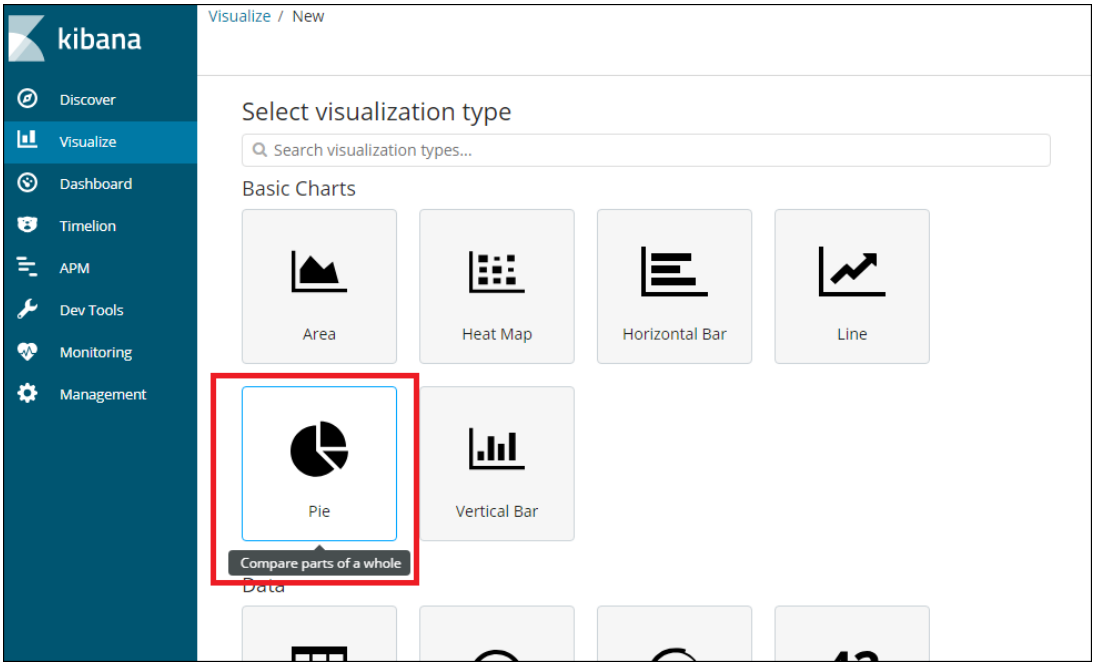
完了したタスク数のタスクごとの内訳を、円グラフを使用して可視化する方法を説明します。

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。



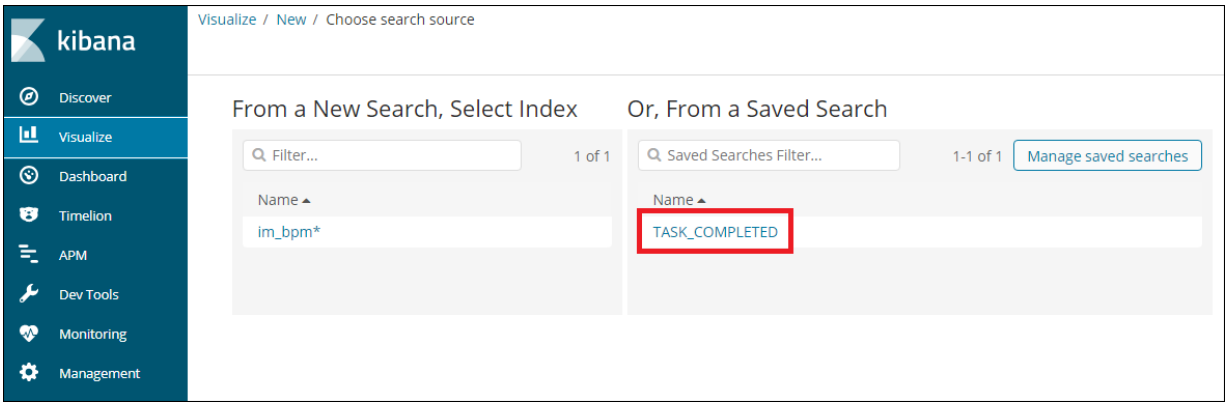
図：「Visualize」

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。



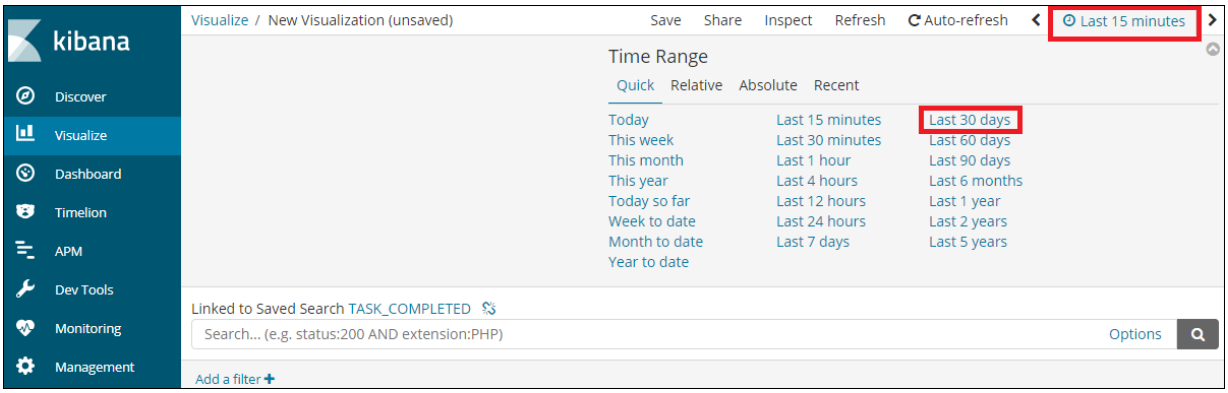
図：「Pie」

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」

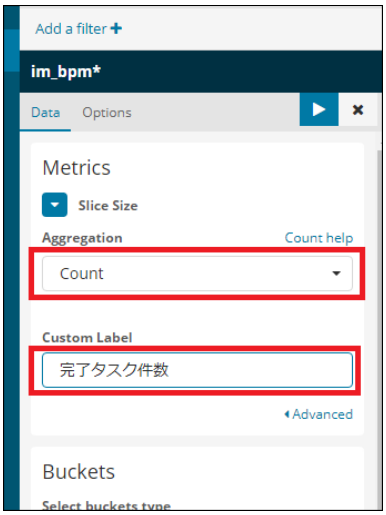
4. 集計対象の期間を指定します。
Kibanaでは、データセットのフィルタ条件に期間が設定されている場合でも、現在のセッション中の「Time Range」の設定が優先される場合があるため、再度、期間を指定します。
「Time Range」欄より「Last 30 days」を選択します。



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

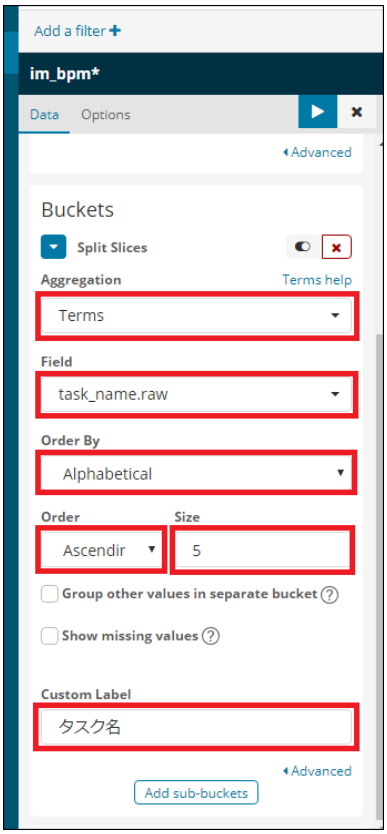


図：「Metrics」→「Slice Size」

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

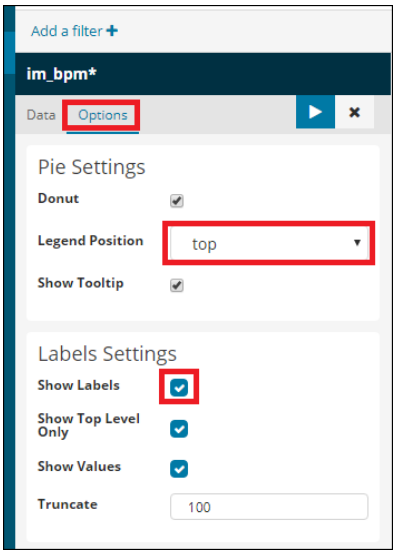
項目名	設定値
Field	task_name.raw
Order By	Alphabetical
Order	Ascending
Size	5
Custom Label	タスク名



図：「Buckets」→「Split Slices」

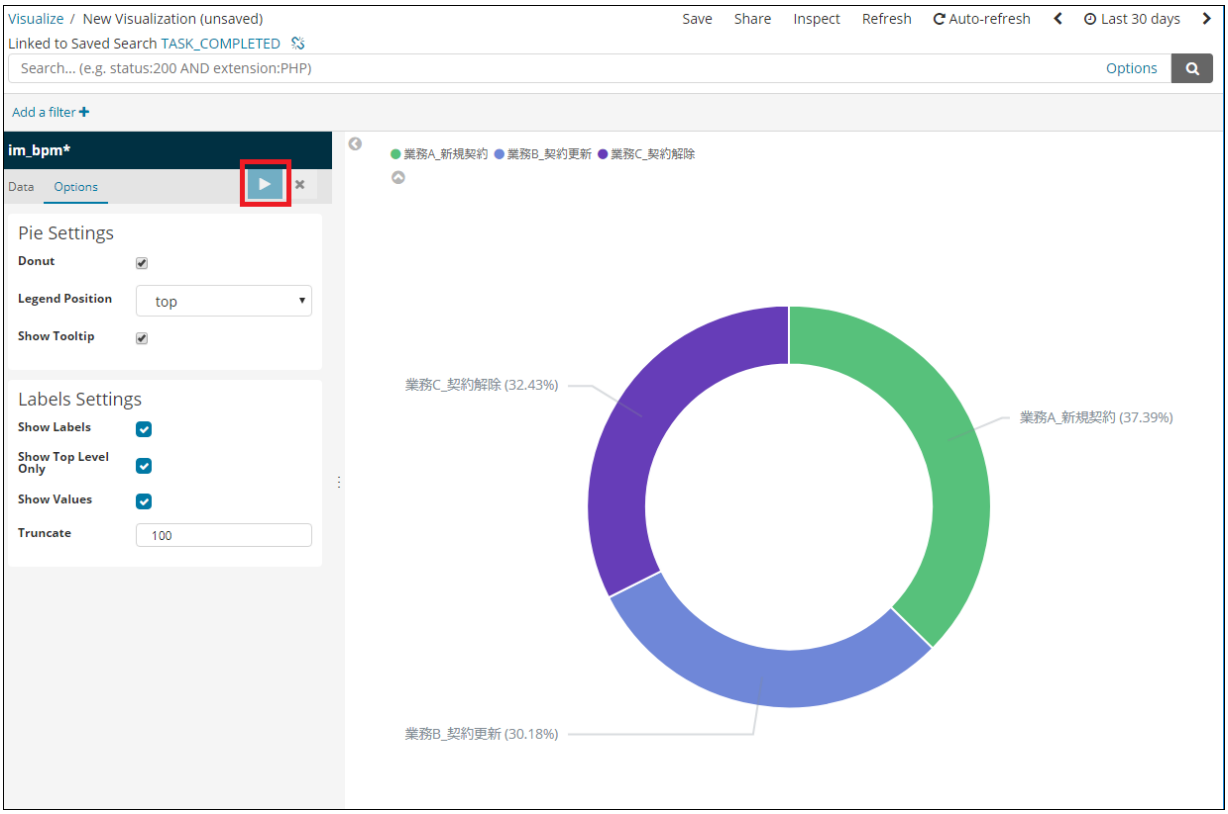
> **コラム**
>
> 「Order By」の「Size」指定について
>
> 「Order By」で指定した方法にてソートを行った後、上位何件を表示するかを「Size」で指定します。
>
> ここではアルファベット昇順でソートした結果の上位5件を表示します。
>
> 上記のSizeの指定で、ソート結果の上位X件を表示する設定（上記例では5件）を行った際、上位X件から漏れたデータは、「Group other values in separate bucket」チェックボックスにチェックを入れると、集計結果を合算して表示できます。ラベル名は「Label for other bucket」で指定したラベル名（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。



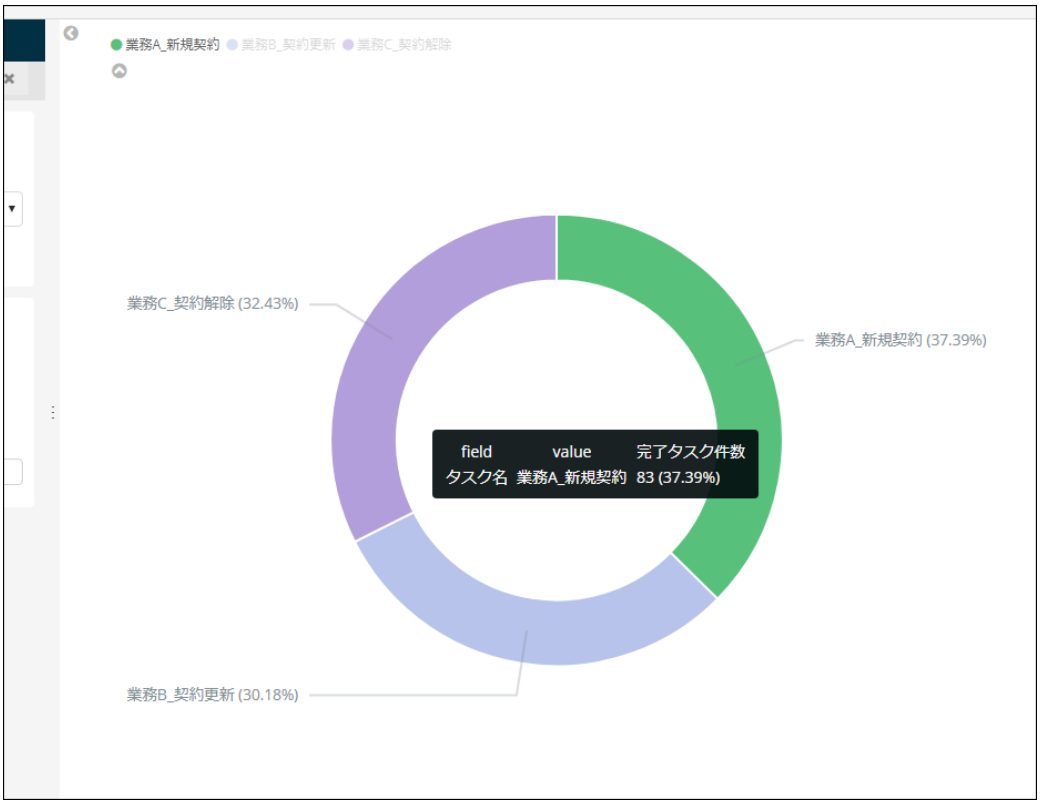
図：「Options」

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。

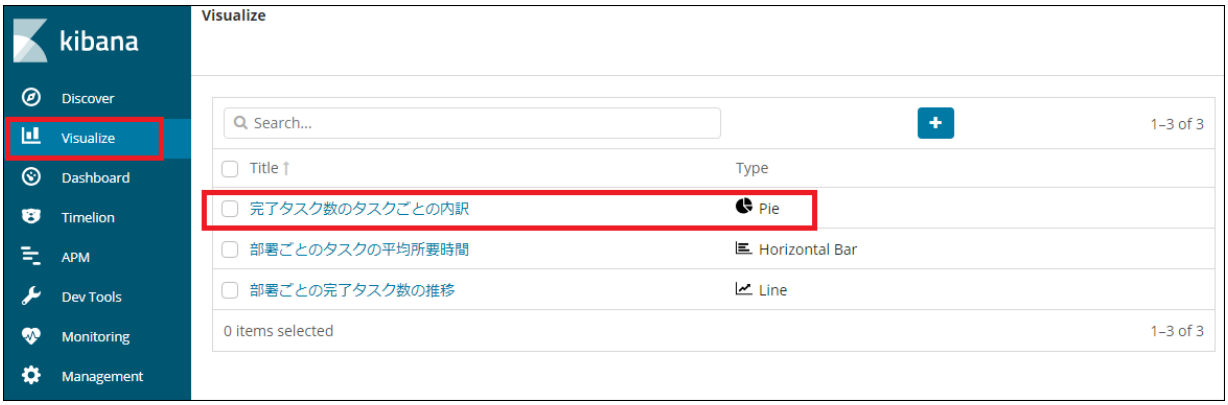
項目名	設定値
Save Visualization	完了タスク数のタスクごとの内訳

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。

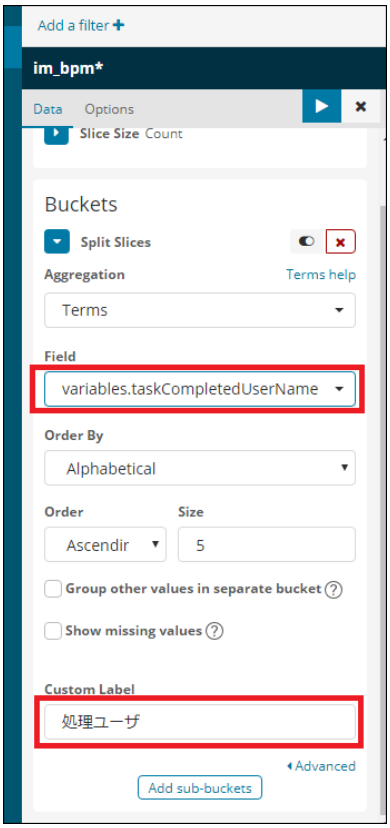
1. 「Kibana」のメインメニューより「Visualize」をクリックし「Visualize」画面を開き、「完了タスク数のタスクごとの内訳」をクリックします。



図：「Visualize」

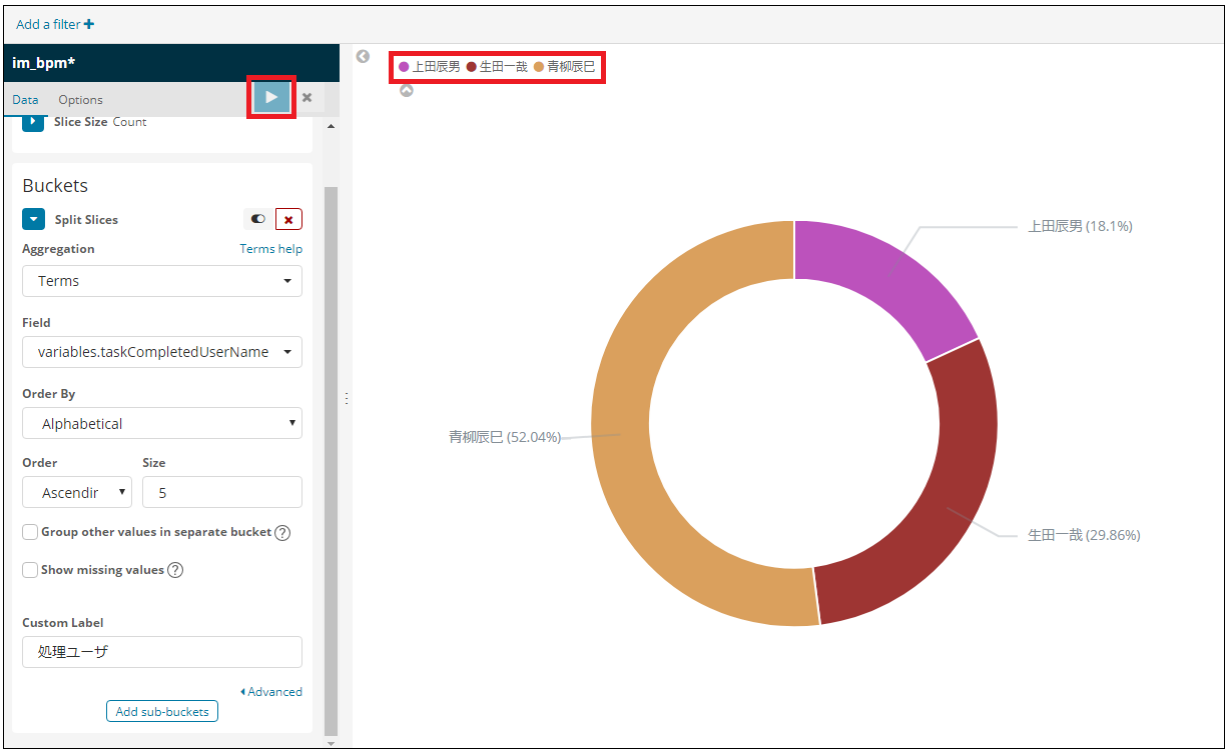
2. タスクの件数の分割単位を「処理ユーザ」ごとの分割に変更します。
「Buckets」→「Split Slices」をクリックし、各項目の値を下記の値に変更します。

項目名	設定値
Aggregation	Terms
Field	variables.taskCompletedUserName
Custom Label	処理ユーザ



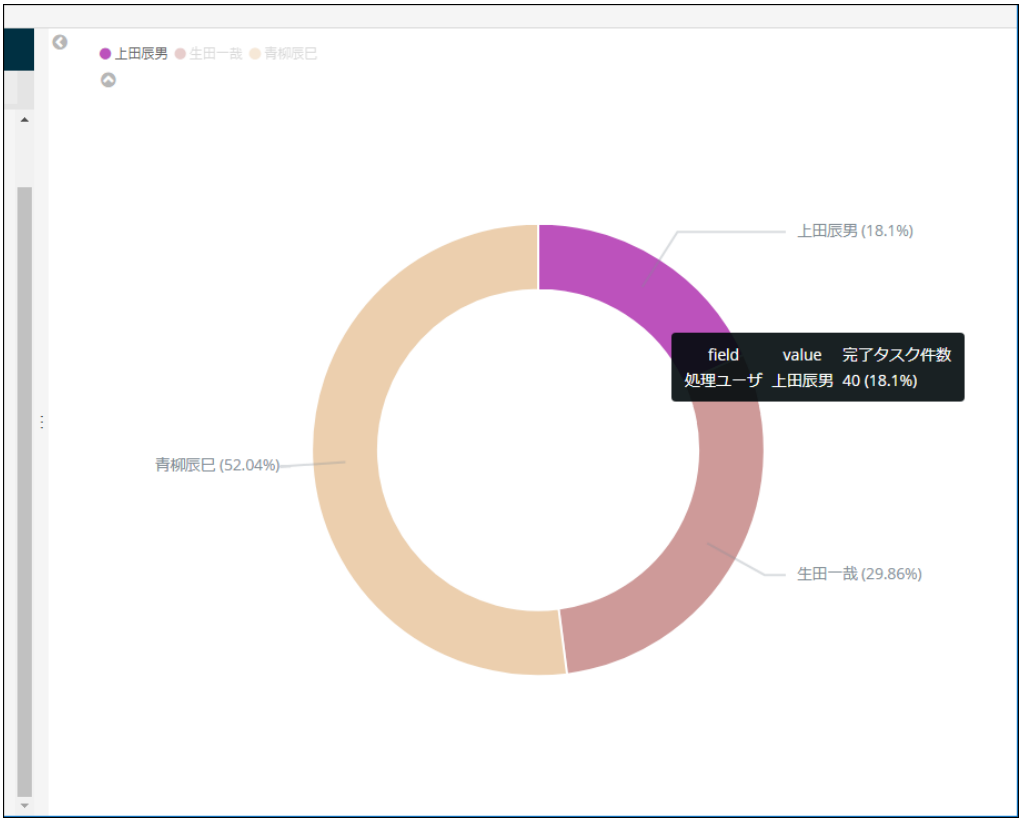
図：「Buckets」→「Split Slices」

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

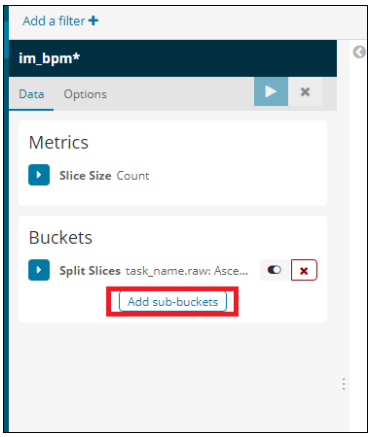
「*「完了タスク数のタスクごとの内訳」を円グラフで可視化する*」で作成した「完了タスク数のタスクごとの内訳」を修正し、完了タスク数のタスクごとの内訳とタスクを処理したユーザの内訳を、2階層の円グラフを使用して可視化する方法を説明します。

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。

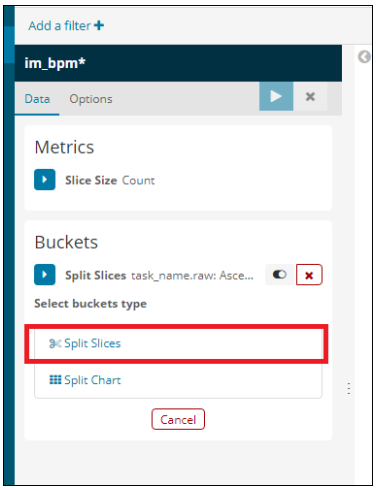
図：「Visualize」

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。

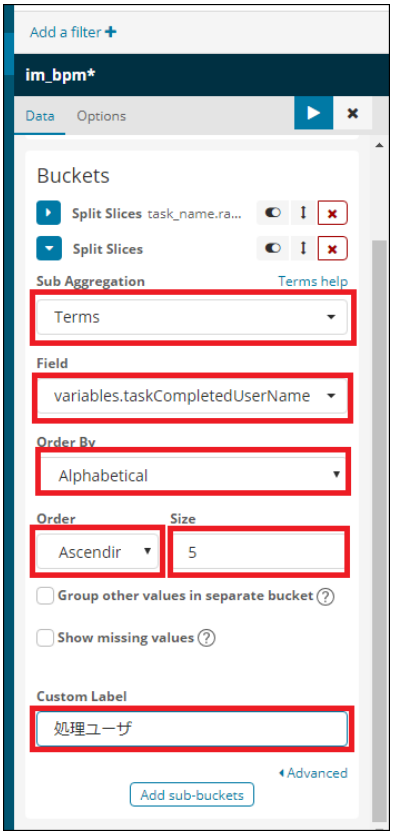
項目名	設定値
Aggregation	Terms
Field	variables.taskCompletedUserName
Order By	Alphabetical
Order	Ascending
Size	5
Custom Label	処理ユーザ



図：「Buckets」→「Add sub-buckets」

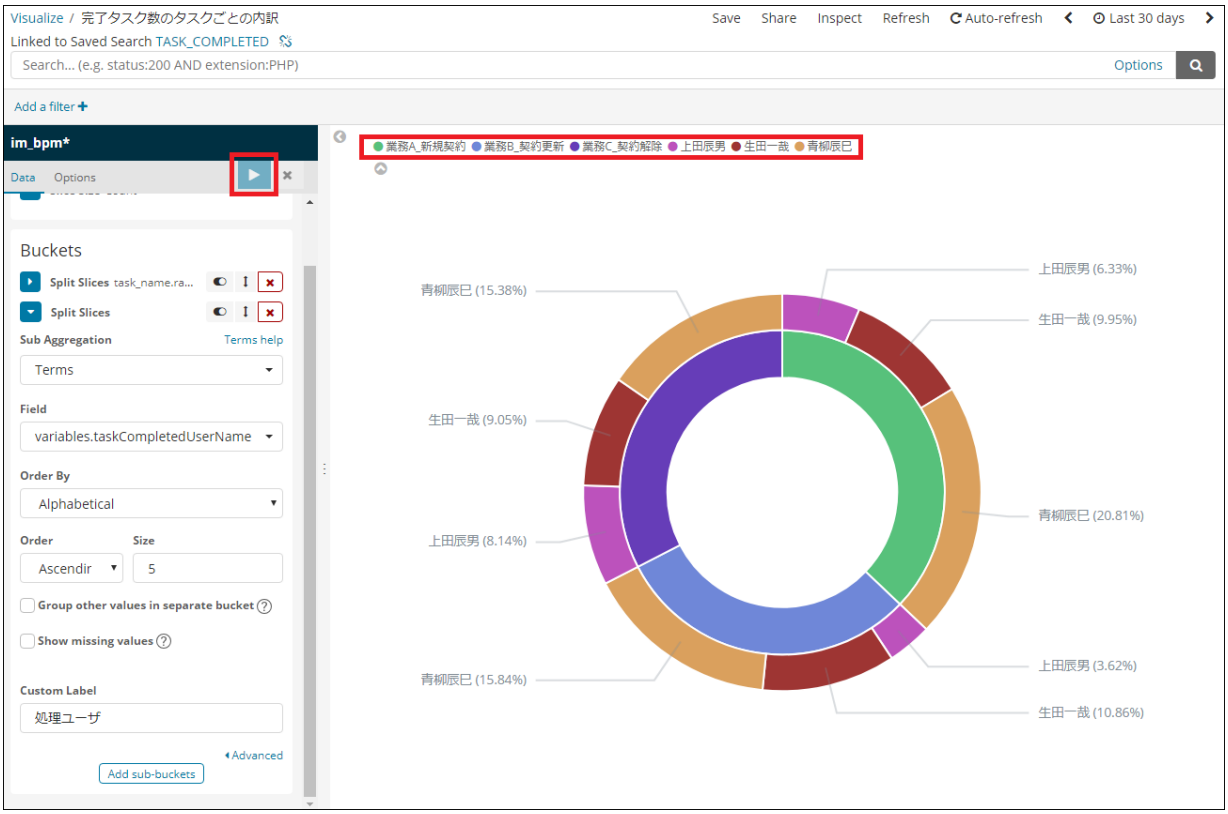


図：「Split Slices」



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。

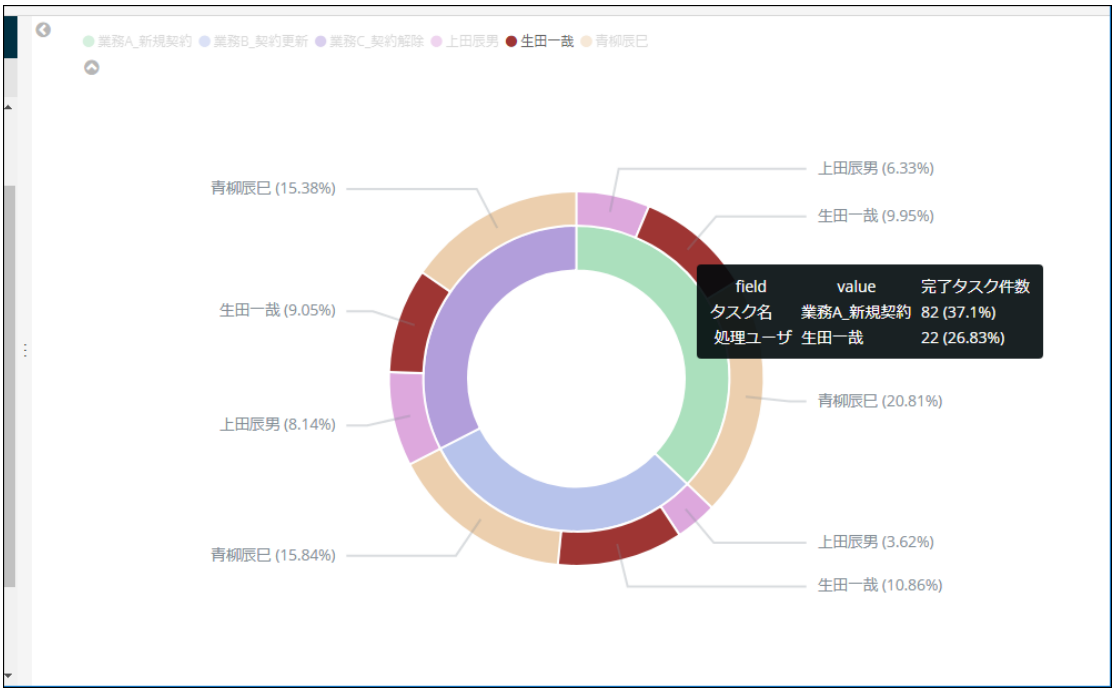


図：「Apply Changes」

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

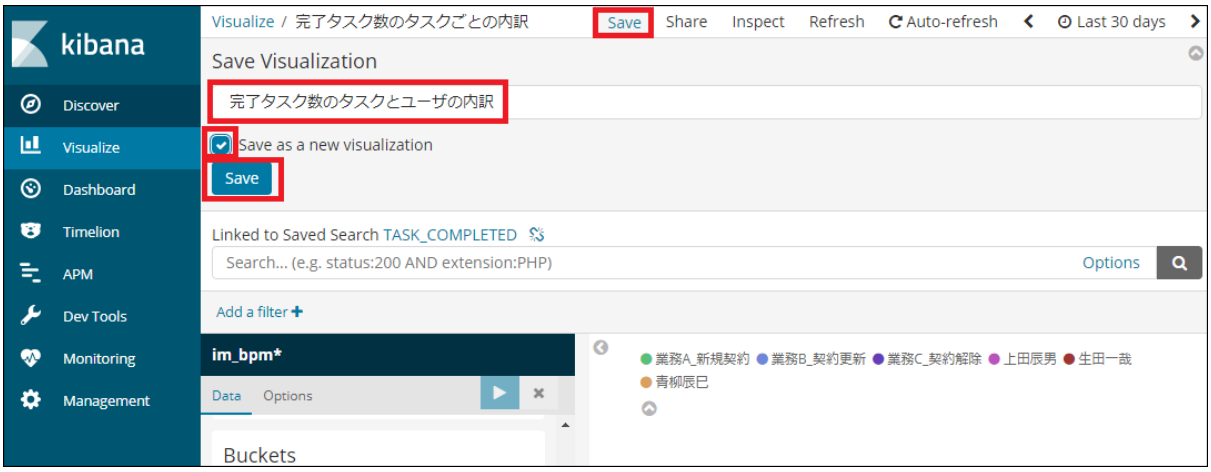
5. 外側の円グラフにマウスカーソルを移動し、「field」「value」「所要時間の内訳」がポップアップで表示されることを確認します。（「field」の詳細項目として「処理ユーザ」が表示されます。）



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

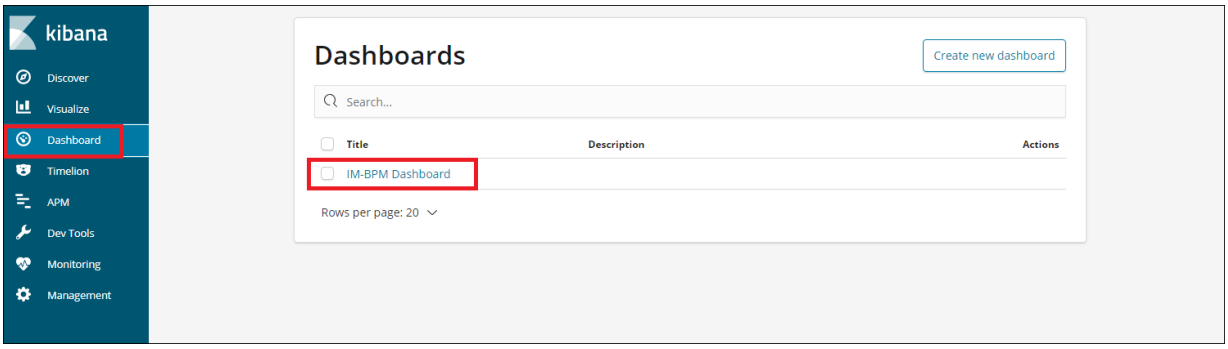


図：「Save Visualization」

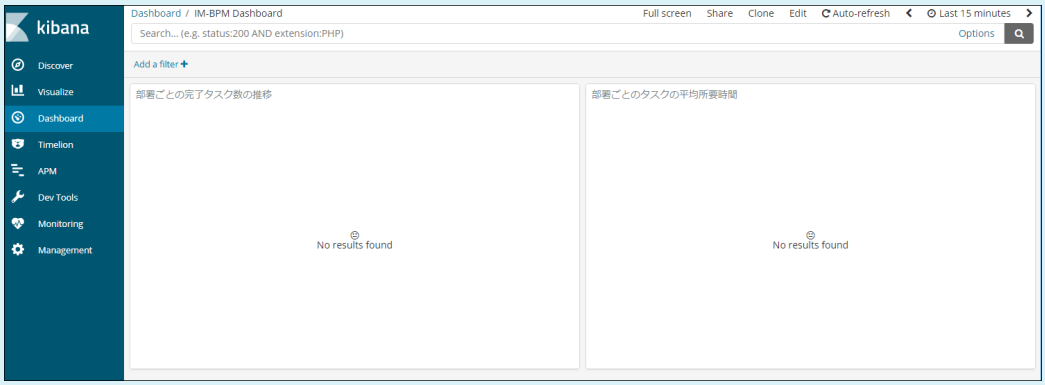
「完了タスク数のタスクごとの内訳」、「完了タスク数のタスク処理ユーザの内訳」、「完了タスク数のタスクとユーザの内訳」をダッシュボードに追加する

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。

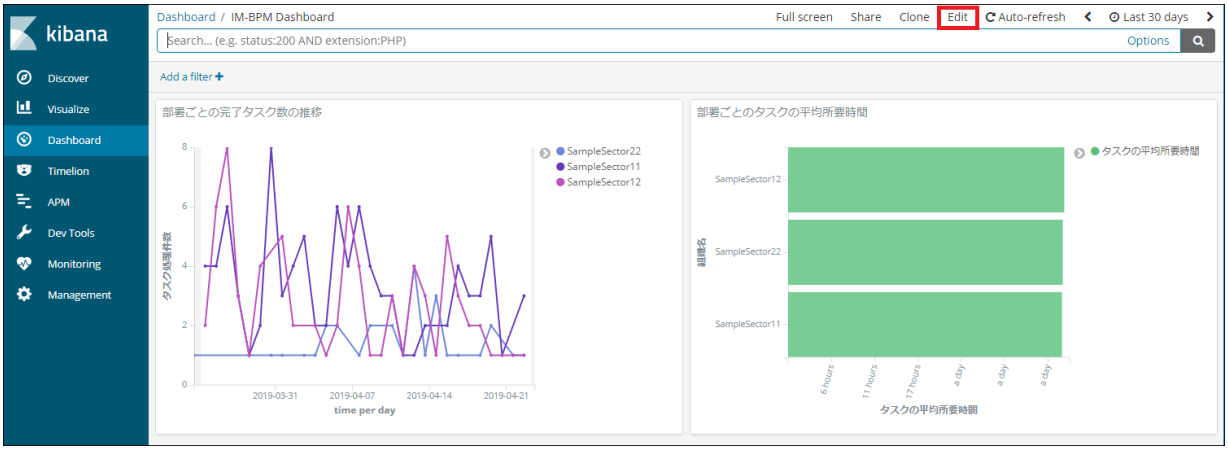
1. Kibanaのメインメニューより「Dashboard」をクリックし、「Dashboards」画面の「IM-BPM Dashboard」をクリックします。



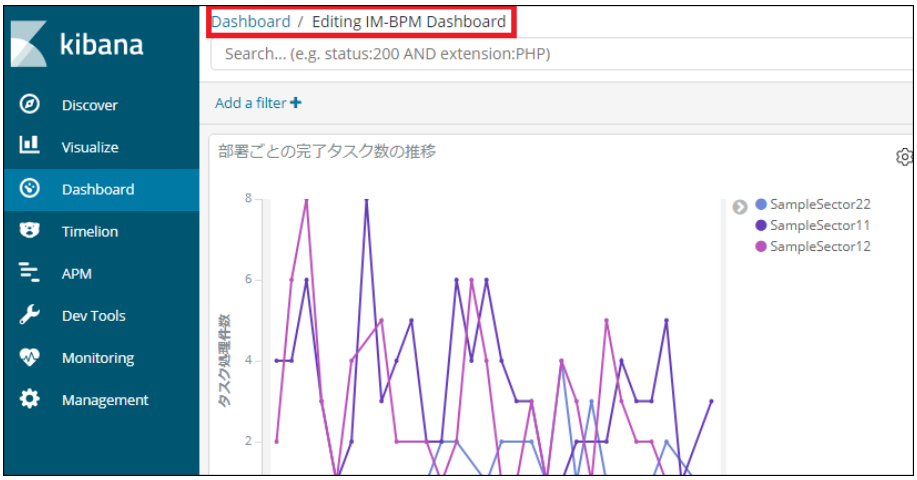
図：「Dashboards」画面

> **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」

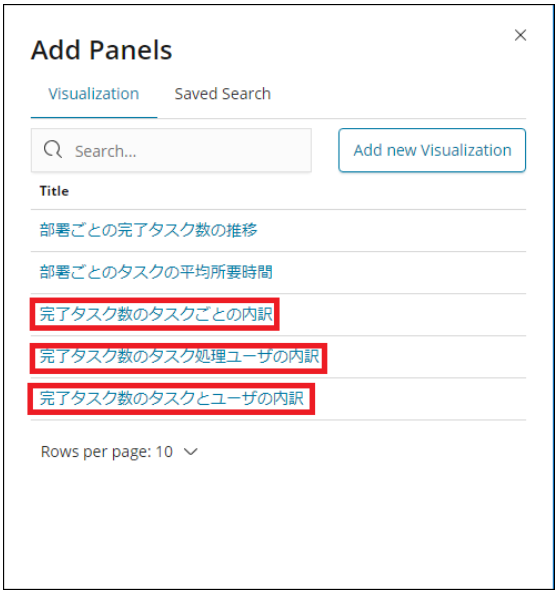


図：「Dashboard / Editing IM-BPM Dashboard」画面

3. ヘッダメニューの「Add」をクリックし、「Add Panels」を開きます。

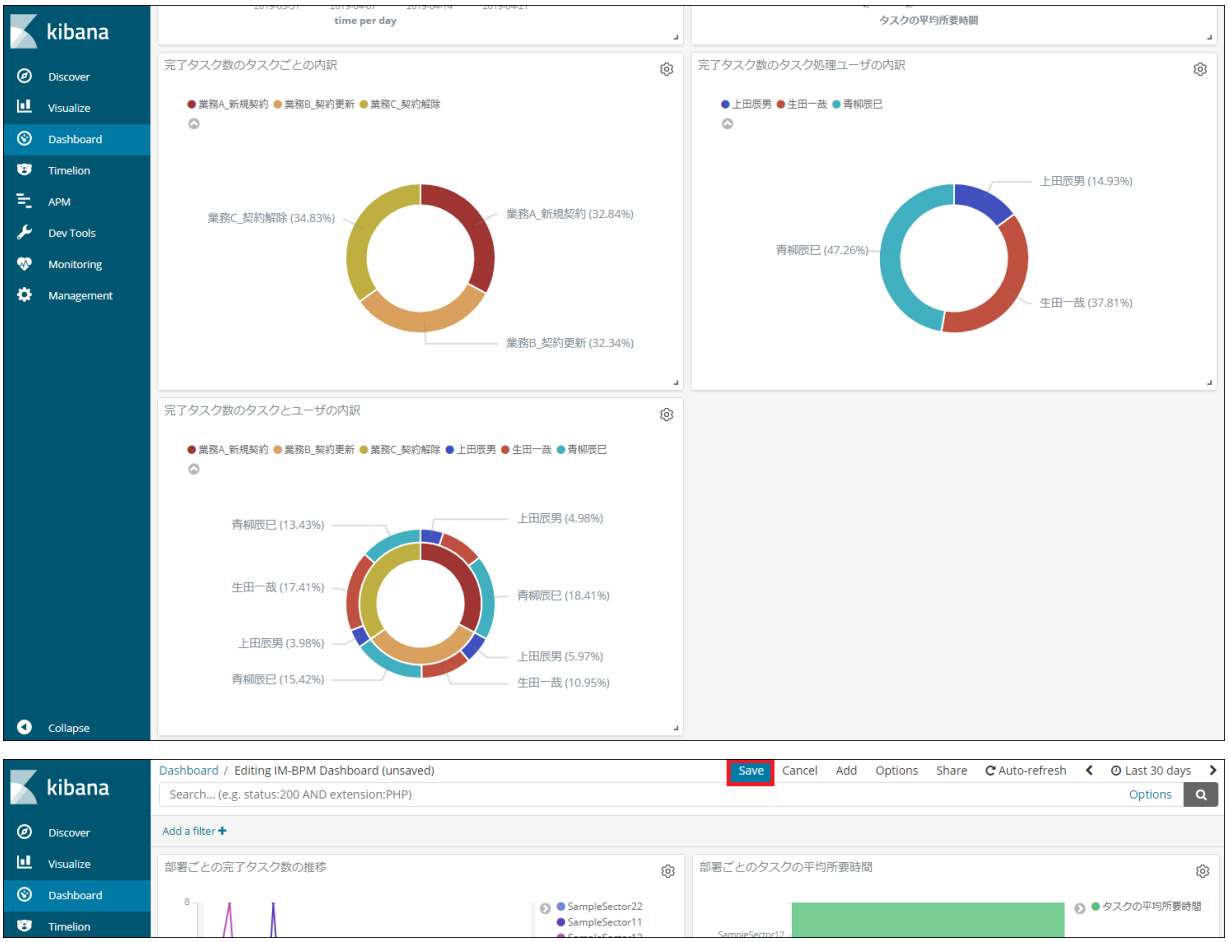
図：「Editing New Dashboard」→「Add」

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。



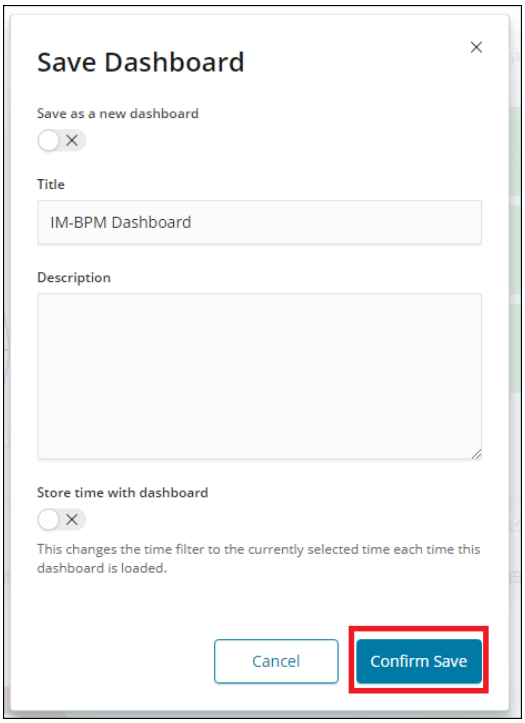
図：「Add Panels」→「Visualization」

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。

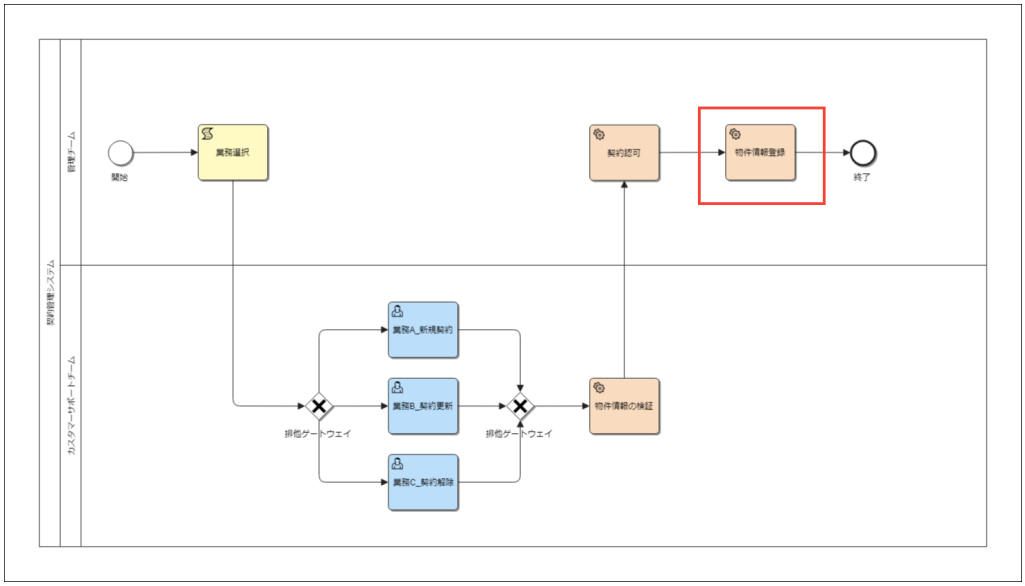


図：「Save Dashboard」

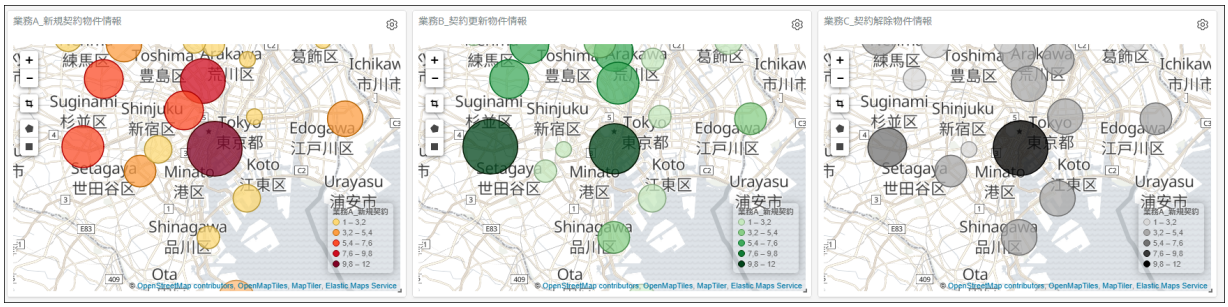
タスクの処理件数を地図上で可視化する

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」

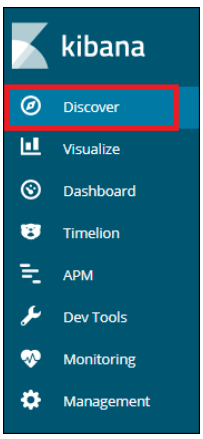
> **注意**
>
> 本チュートリアルは、事前に「*IM-BPM for Accel Platformのプロセスの実行時のログをKibanaのダッシュボードに表示する*」の実施が完了していることを前提とします。

- 「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。



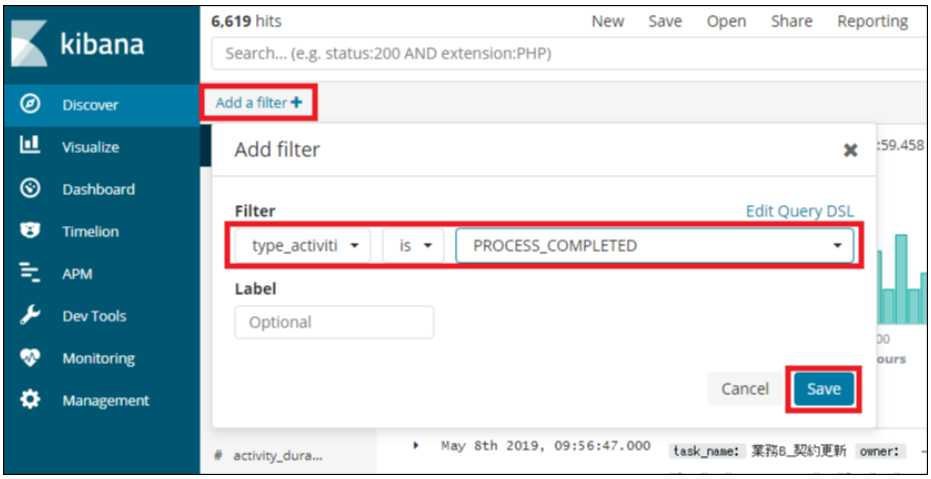
図：「Discover」

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

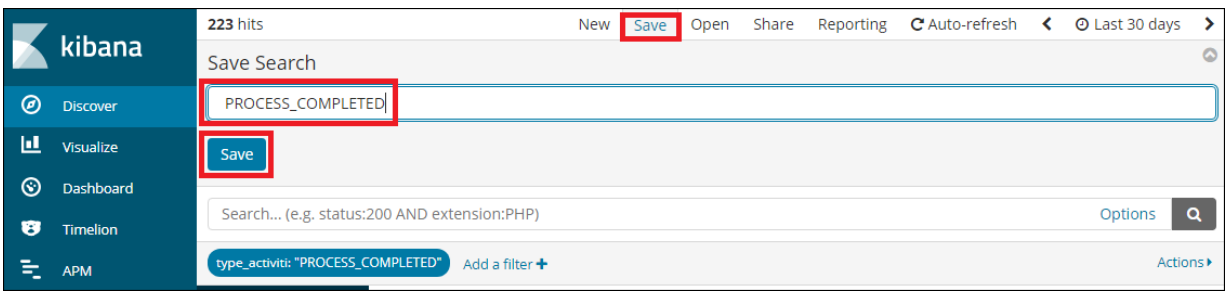


図：「Add a filter」

> **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」

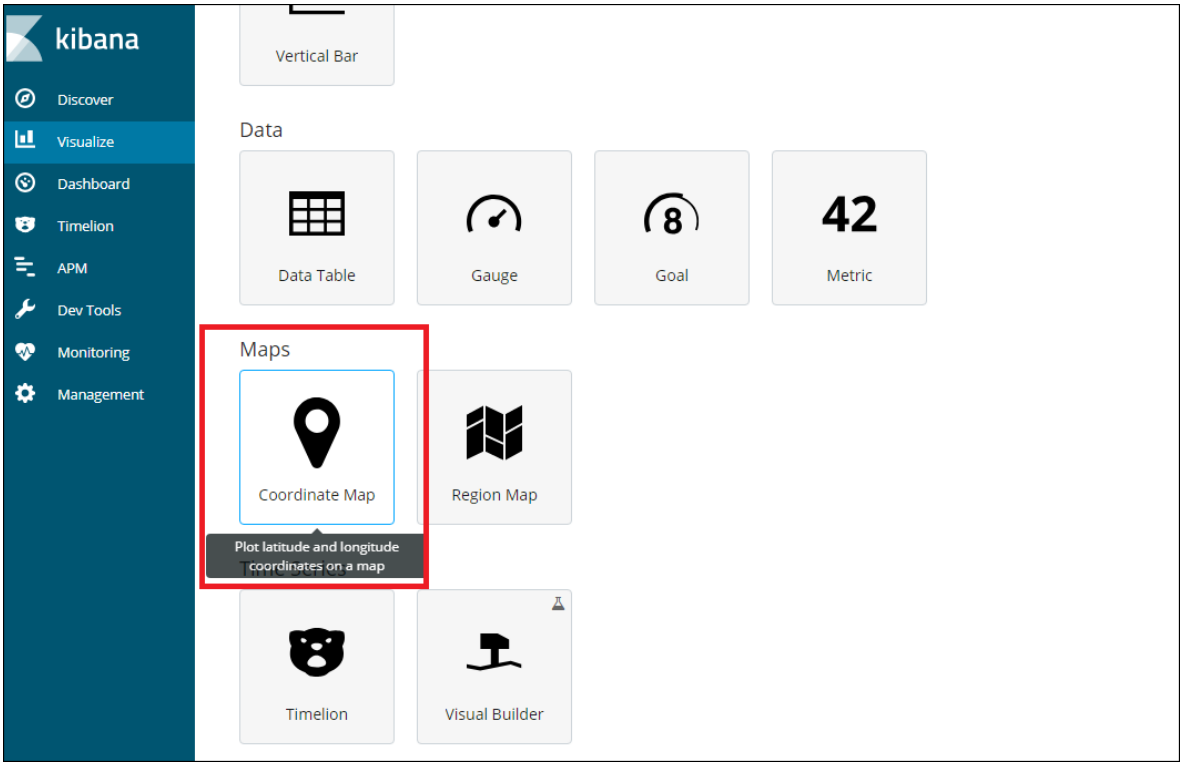
情報を地図上で可視化する

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。

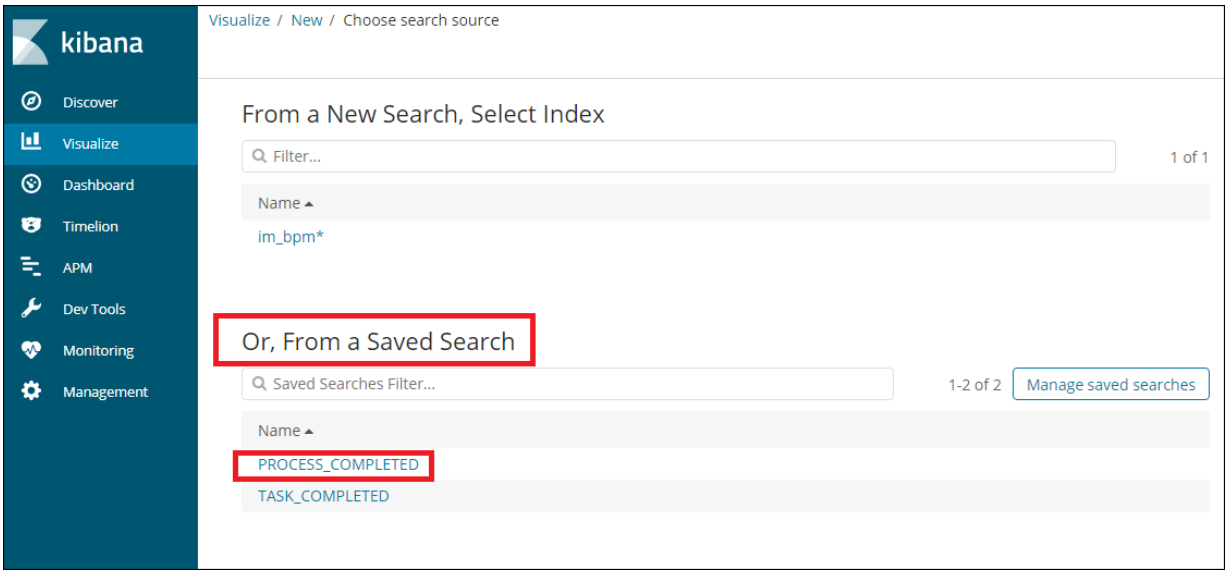
図：「Visualize」

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。



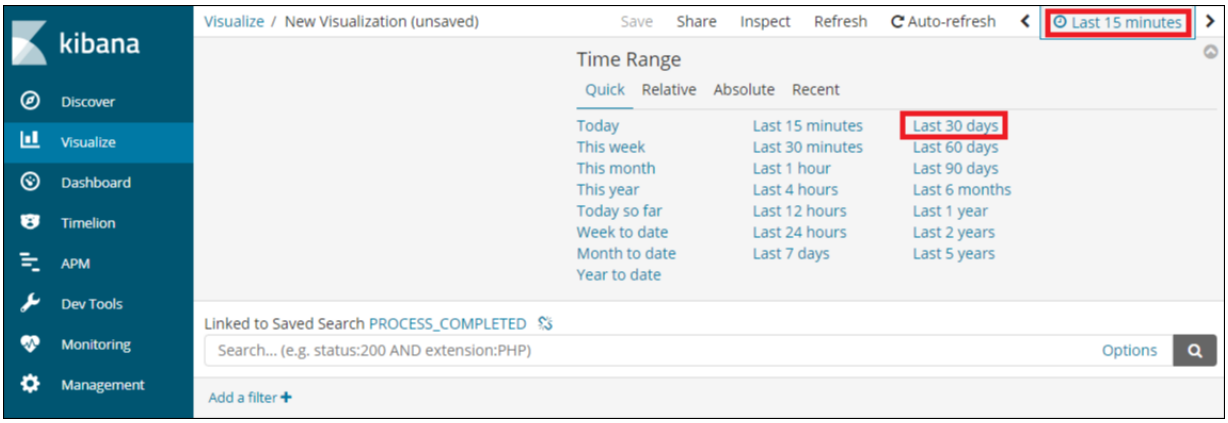
図：「Coordinate Map」

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」

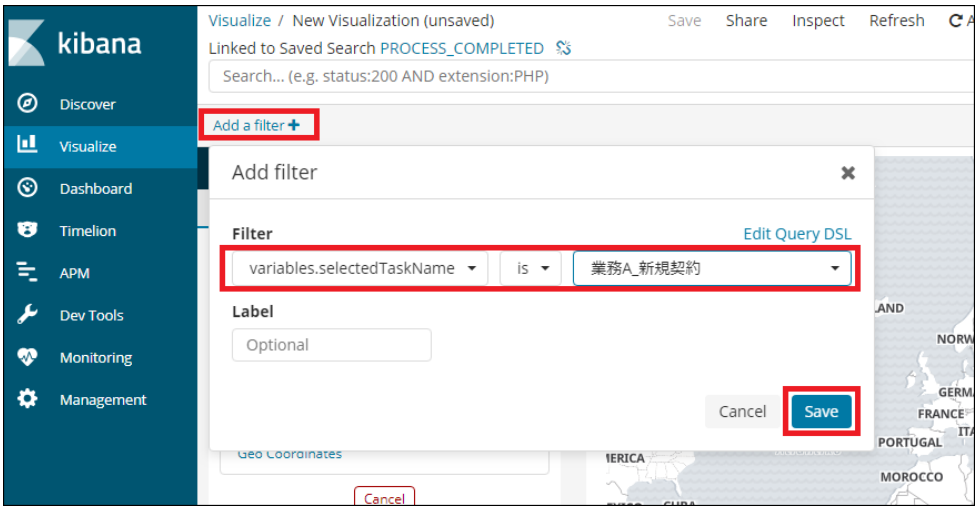
4. 集計対象の期間を指定します。
Kibanaでは、データセットのフィルタ条件に期間が設定されている場合でも、現在のセッション中の「Time Range」の設定が優先される場合があるため、再度、期間を指定します。
「Time Range」欄より「Last 30 days」を選択します。



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。

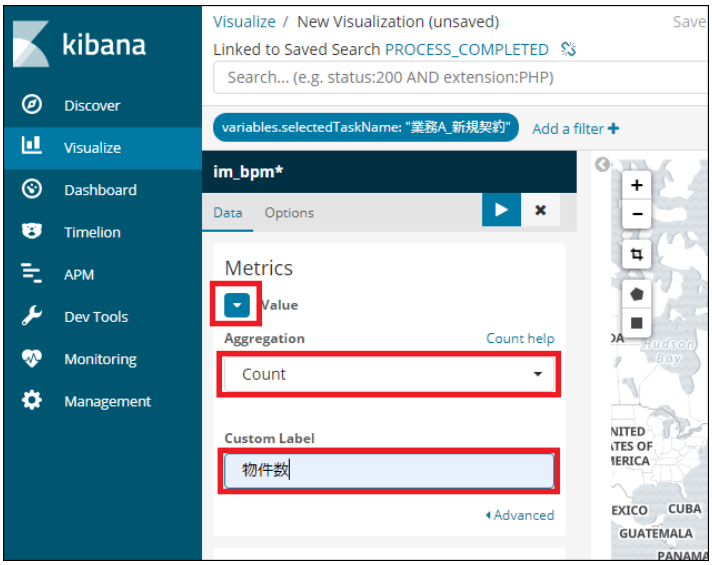
項目名	設定値
Fields...	variables.selectedTaskName
Operators...	is
Values...	業務A_新規契約



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。

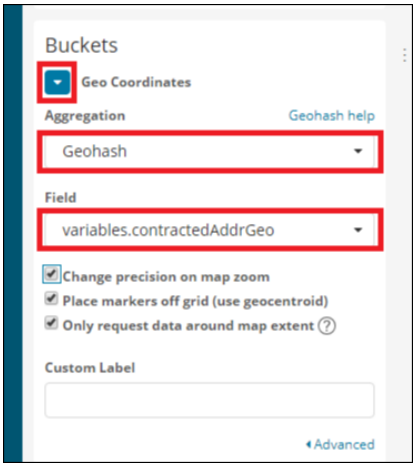
項目名	設定値
Aggregation	Count
Custom Label	物件数



図：「Data」→「Metrics」

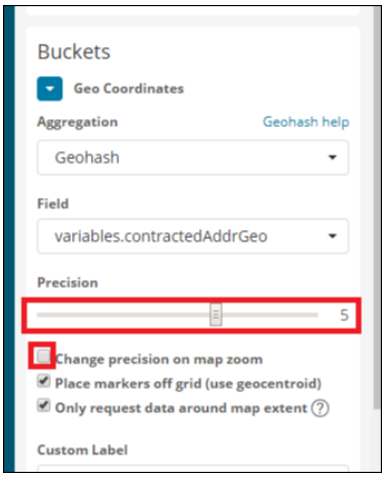
7. 緯度・経度情報を持つフィールドvariables.contractedAddrGeoを可視化の対象として設定します。
「Data」タブ→「Buckets」→「Geo Coordinates」をクリックし、下記の値を指定してください。

項目名	設定値
Aggregation	Geohash
Custom Label	variables.contractedAddrGeo



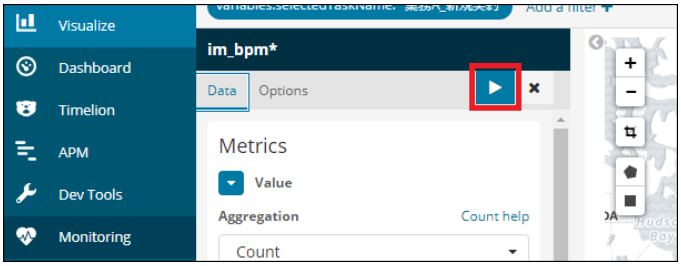
図：「Buckets」→「Geo Coordinates」

8. マップの拡大・縮小に追随し、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・縮小に追随せず、精度一定）を選択します。
「Change precision on map zoom」のチェックを外し、スライドバーでprecisionに「5」を設定します。



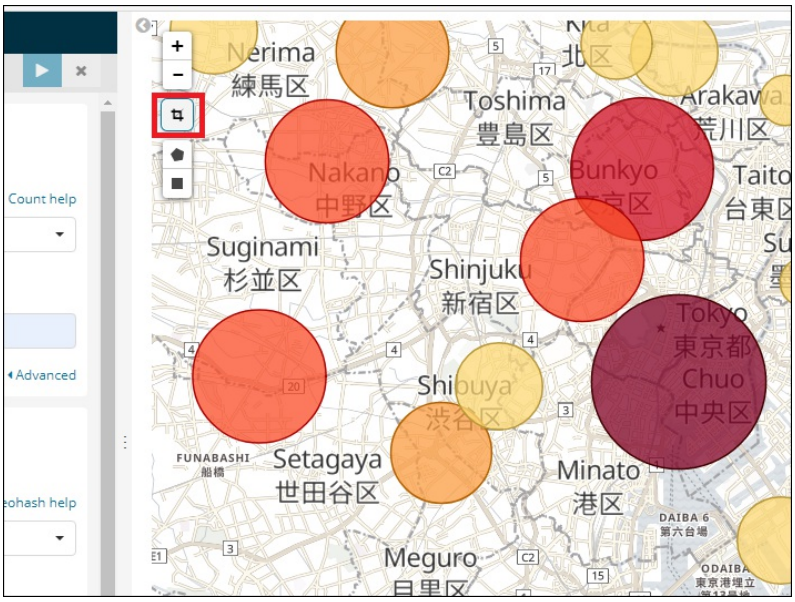
図：「Change precision on map zoom」

9. 「Apply changes」をクリックし設定を反映します。



図：「Apply changes」

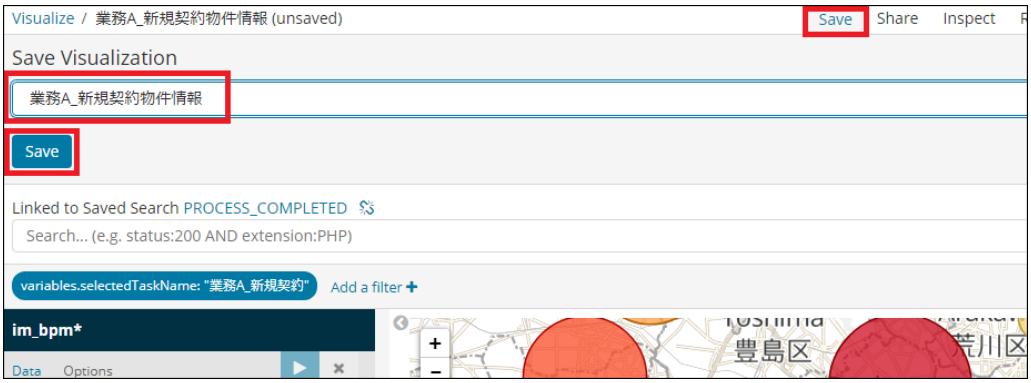
10. 「Fit Data Bounds」をクリックし地図の縮尺を調整し、地図上に物件情報が可視化されることを確認します。



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

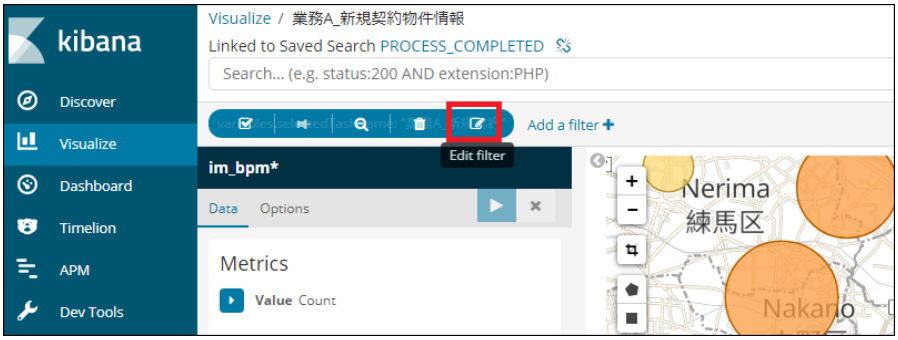
ここでは、「*情報を地図上で可視化する*」で作成した「業務A_新規契約物件情報」を修正し、「業務B_契約更新」業務で扱われた物件情報の可視化を行います。

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。

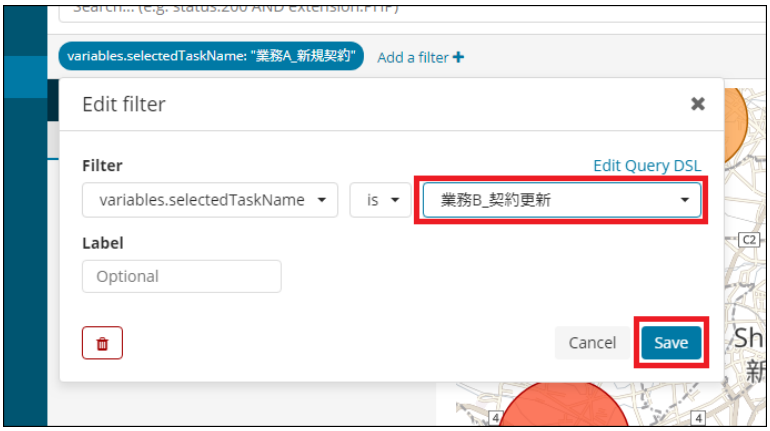
図：「Visualize」

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

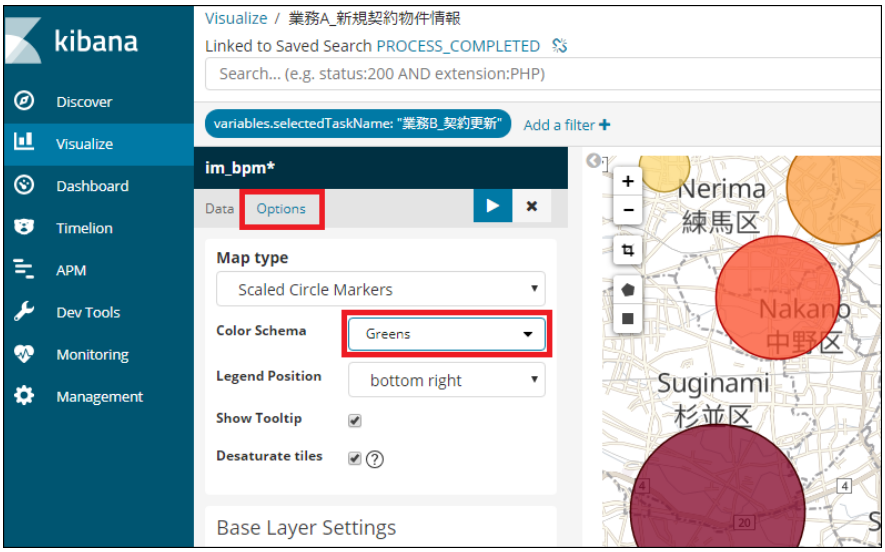


図：「Edit Filter」



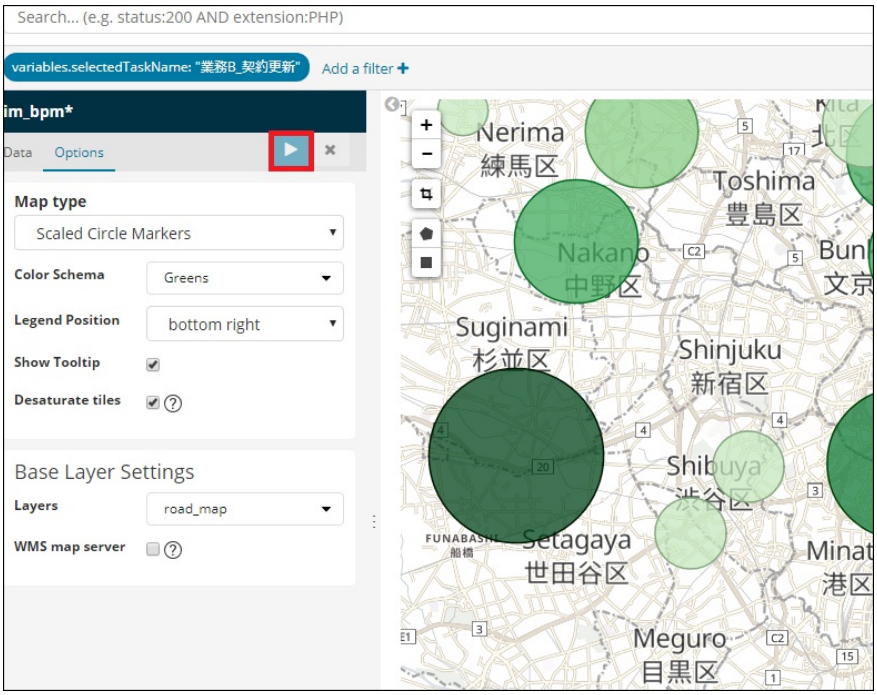
図：フィルタ「variables.selectedTaskName:”業務A_新規契約”」

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」

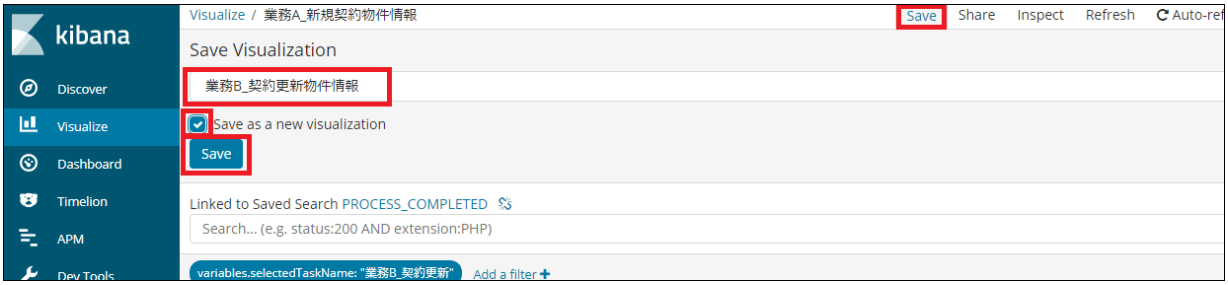
4. 「Apply Changes」をクリックし、修正内容が反映されることを確認します。



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



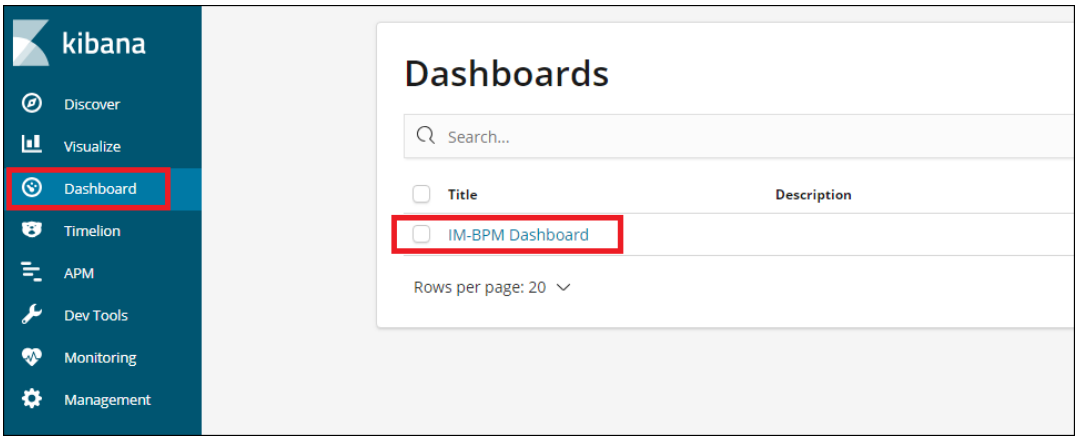
図：「Save Visualization」

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。

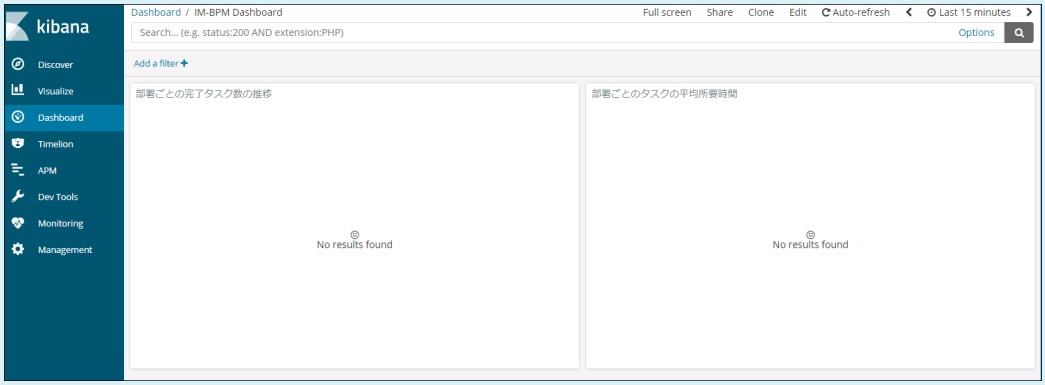
可視化された情報をダッシュボードに追加する

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。

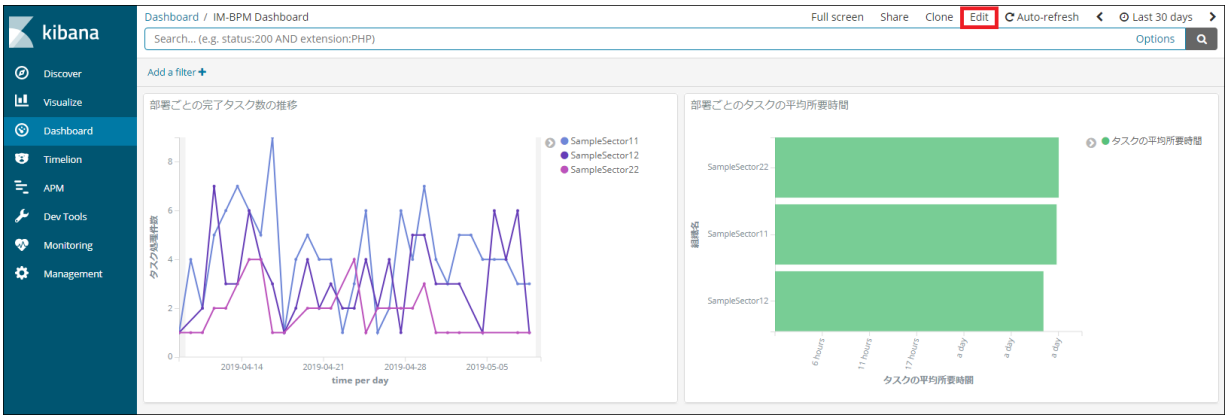
1. Kibanaのメインメニューより「Dashboard」をクリックし、「Dashboards」画面の「IM-BPM Dashboard」をクリックします。



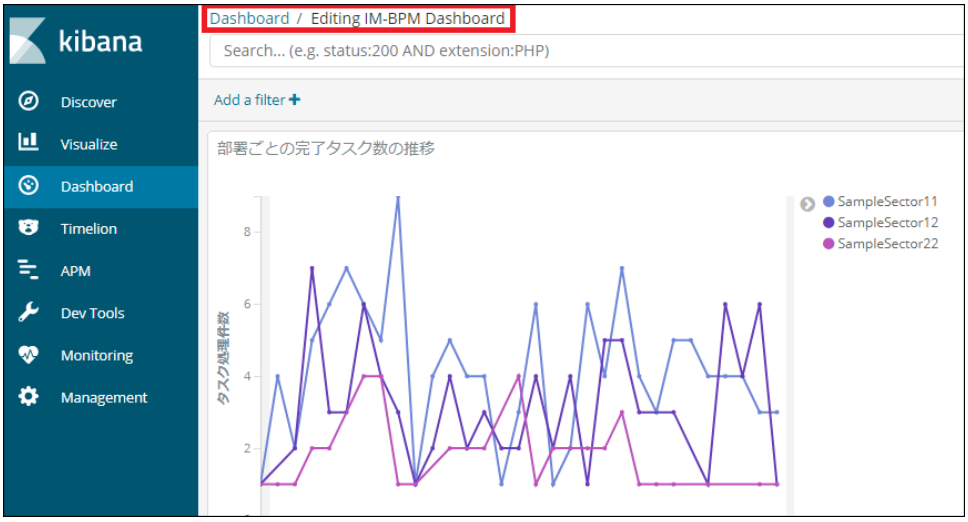
図：「Dashboards」画面

> **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。

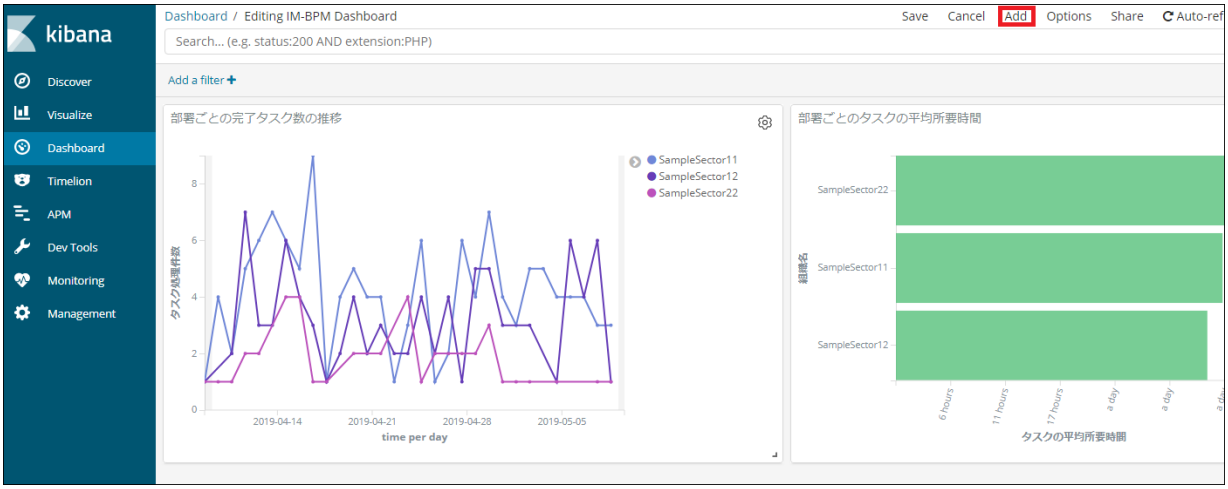


図：「IM-BPM Dashboard」



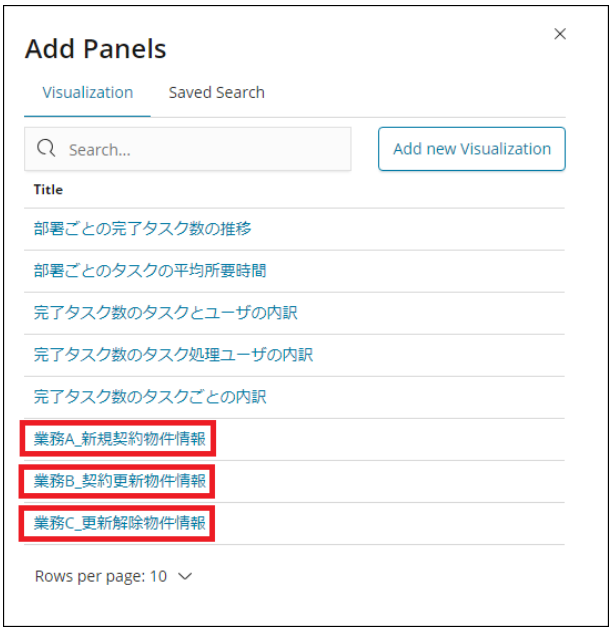
図：「Dashboard / Editing IM-BPM Dashboard」画面

3. ヘッダメニューの「Add」をクリックし、「Add Panels」を開きます。



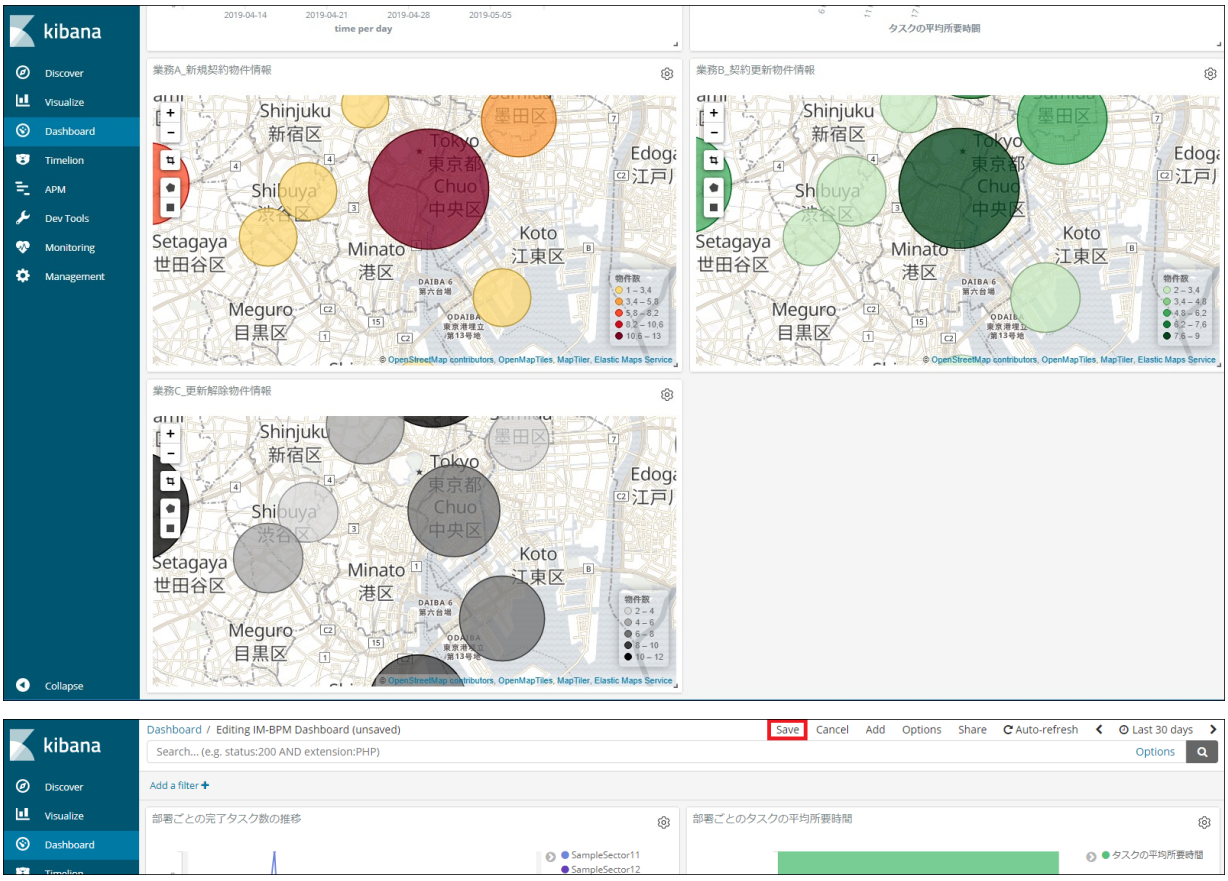
図：「Editing New Dashboard」→「Add」

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。



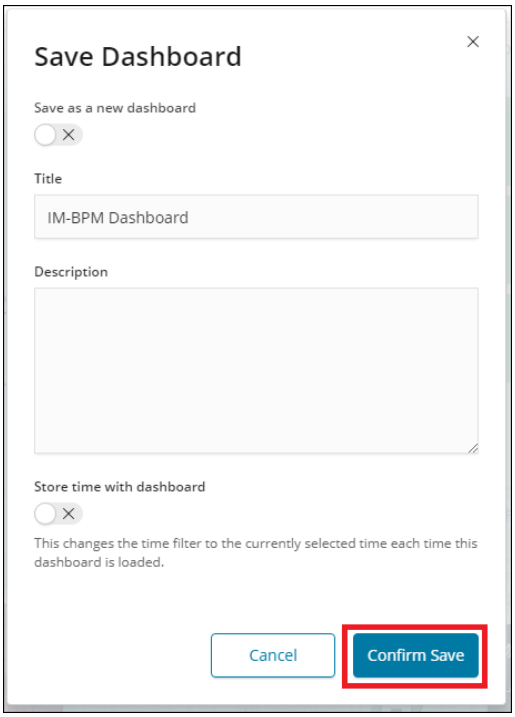
図：「Add Panels」→「Visualization」

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」

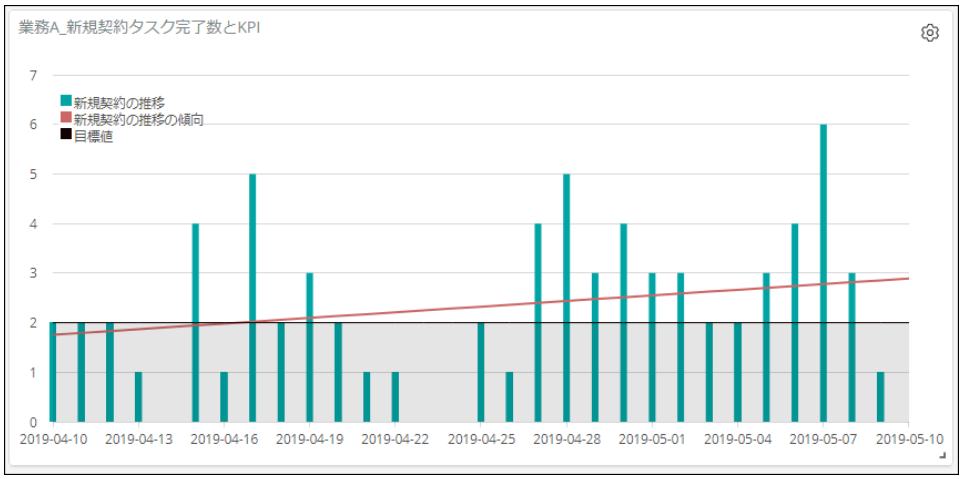
6. 「Confirm Save」をクリックします。



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

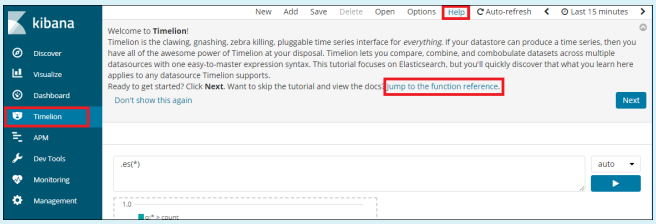
Kibanaの「Timelion」（タイムライン）機能を使用して、グラフ上にKPIとして静的な基準線と、グラフの傾向（増加傾向／減少傾向）を表示します。

コラム

「Timelion」画面と「Timelion Expression」について

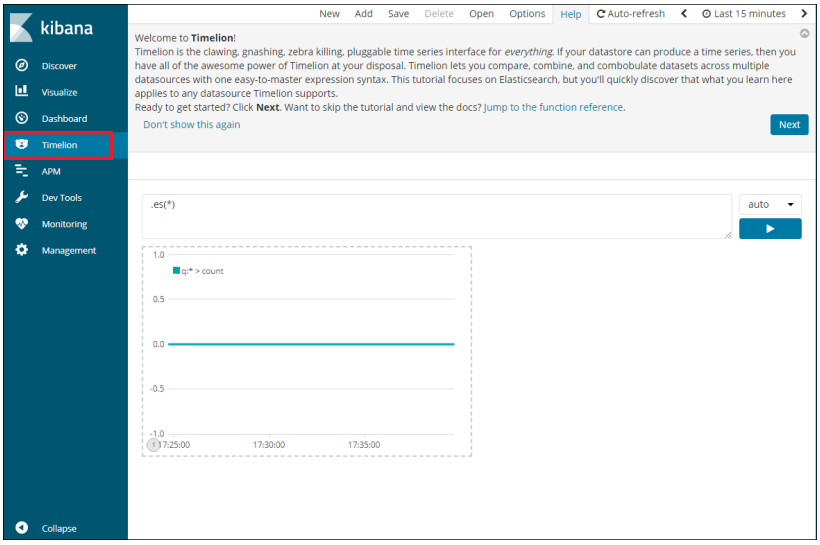
「Timelion」画面では、「Timelion Expression」という記法にもとづき、複雑な形式のグラフを表示したり、また、それらを組み合わせ、一つの図中に複数のグラフを重ねて表示するなど、様々な表現を行えます。

「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。



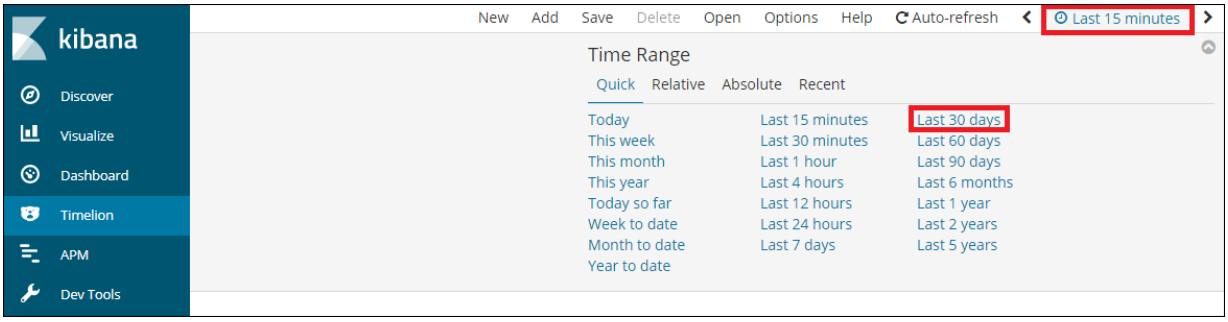
図：「Help」→「Function reference」

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。



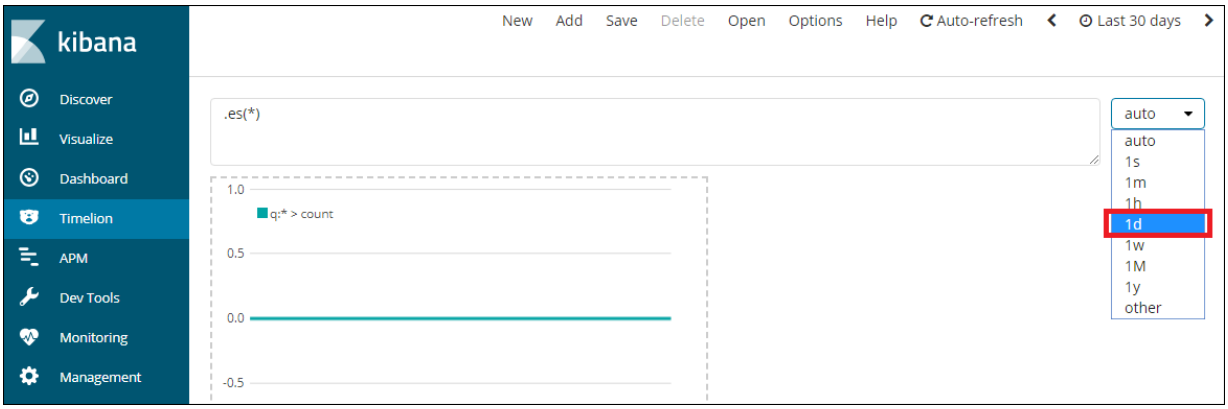
図：「Timelion」画面

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
`index='im_bpm*'`	可視化対象のインデックスパターンを指定します。
`q='type_activiti:PROCESS_COMPLETED AND variables.selectedTaskName:業務A_新規契約'`	抽出条件を表すクエリを指定します。
`metric='count'`	メトリクスとしてドキュメントの件数を指定します。
`timefield='time'`	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。

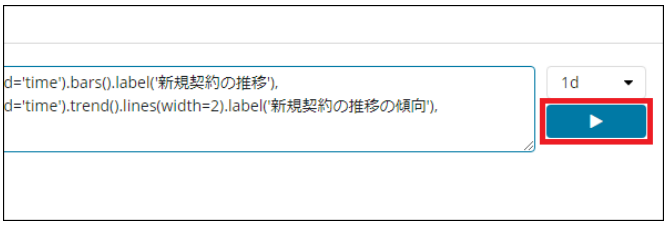
  - .color()関数をメソッドチェーン形式で呼び出し、グラフの色を指定します。

パラメータ	説明
#000000	16進数カラーコードを指定します。

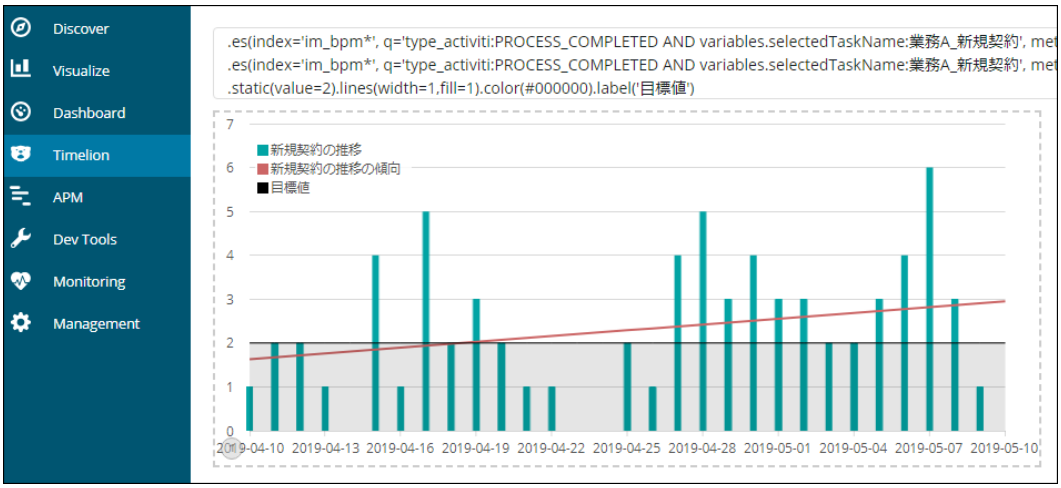
  - .label()関数をメソッドチェーン形式で呼び出し、凡例に表示されるラベルを設定します。

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。

5. 「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。



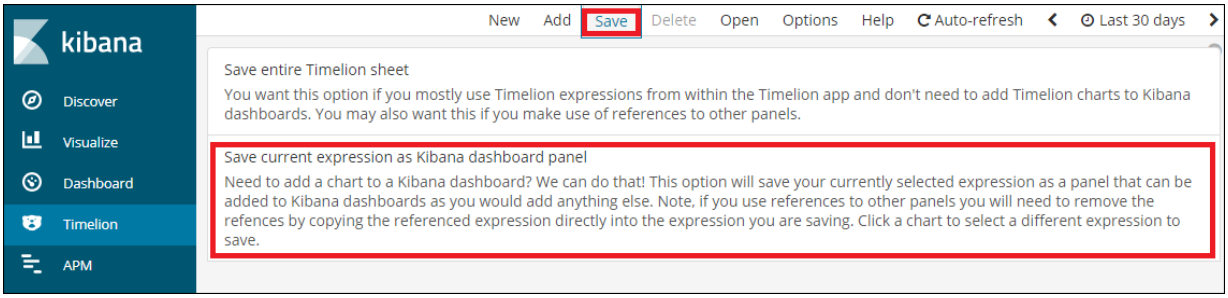
図：「Apply Changes」



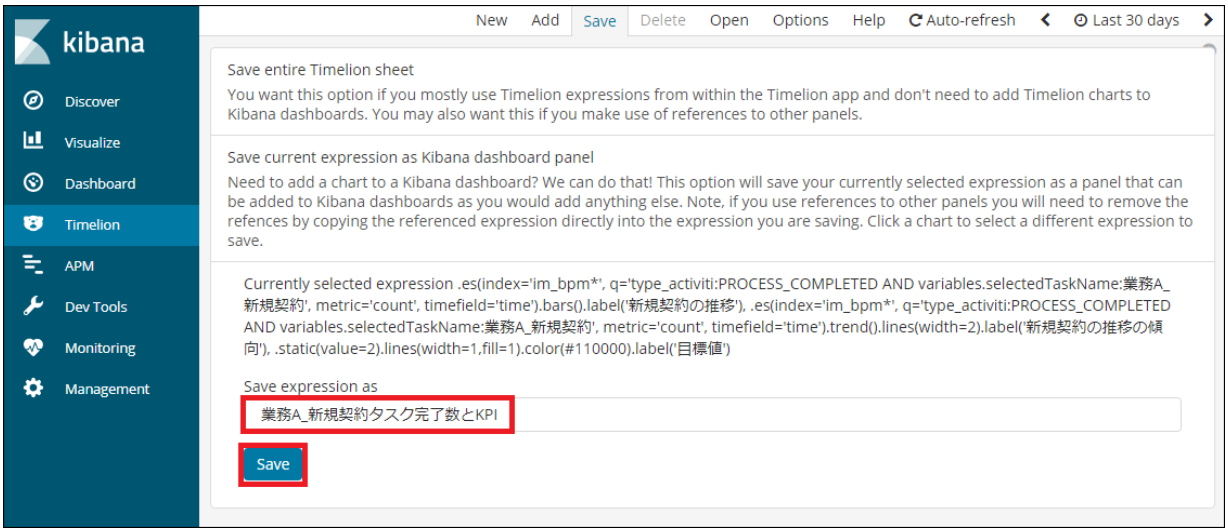
図：「新規契約の推移」の棒グラフと、「目標値」KPI、「新規契約の推移の傾向」

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」

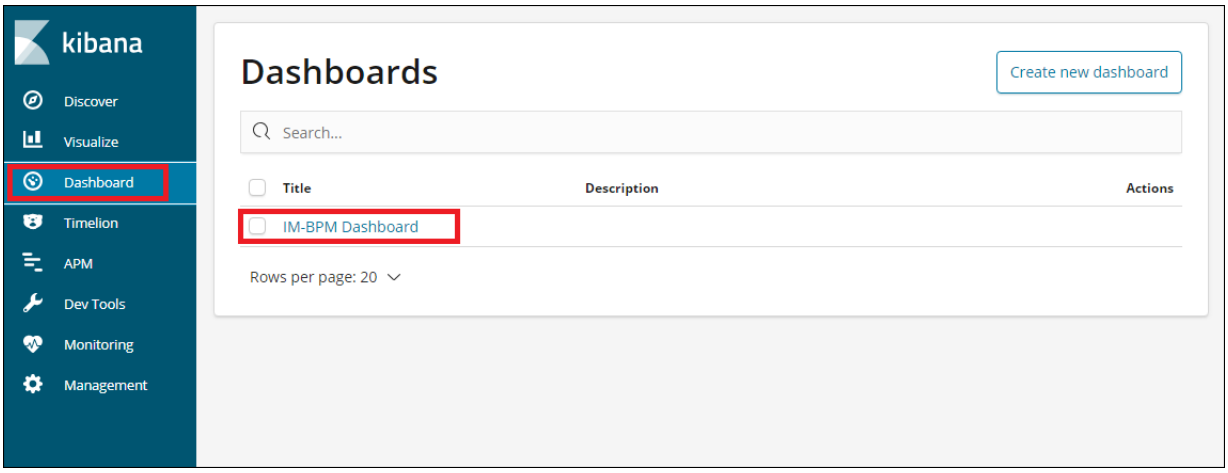


図：「Save expression as」

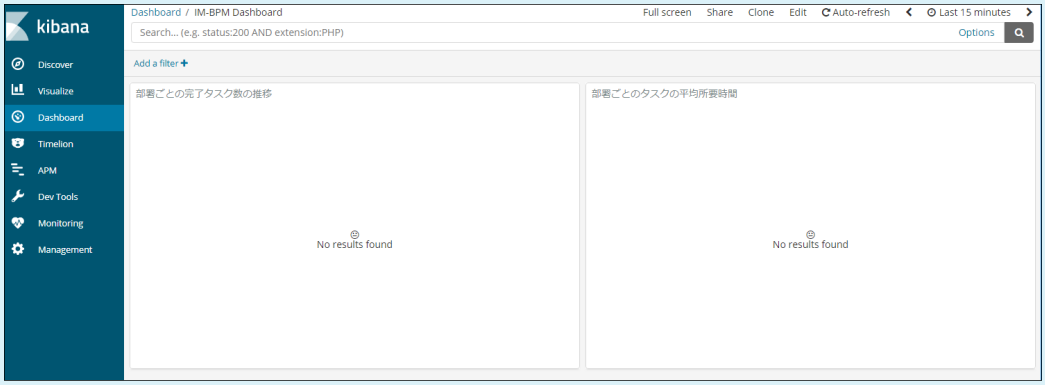
「業務A_新規契約タスク完了数とKPI」をダッシュボードに追加する

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。

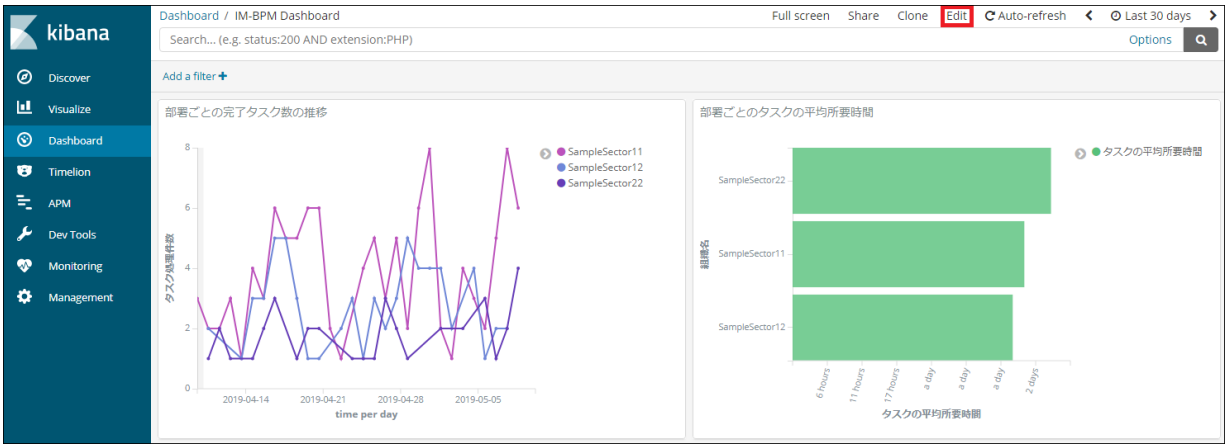
1. Kibanaのメインメニューより「Dashboard」をクリックし、「Dashboards」画面の「IM-BPM Dashboard」をクリックします。



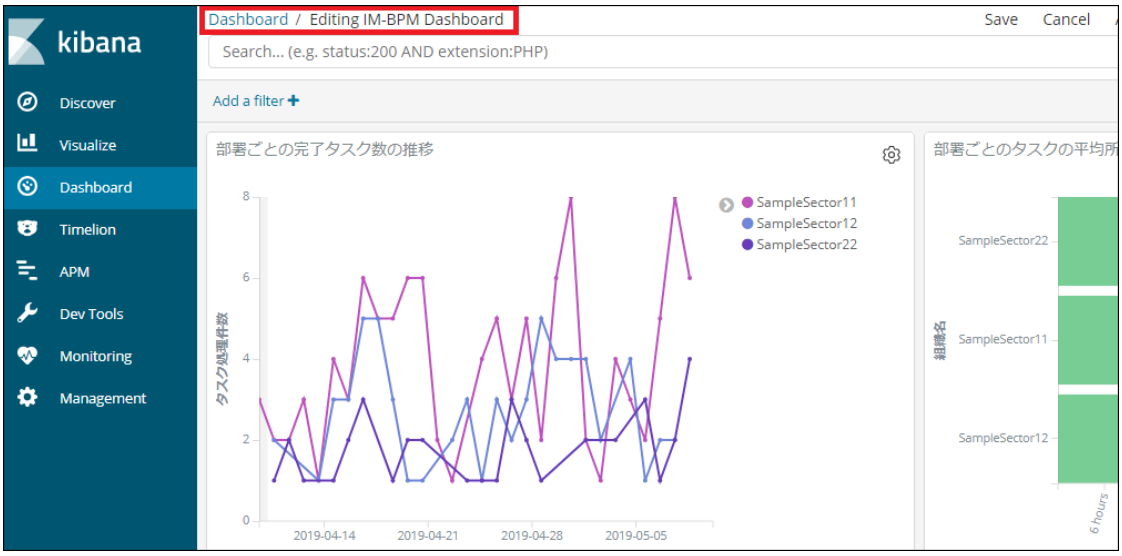
図：「Dashboards」画面

> **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。

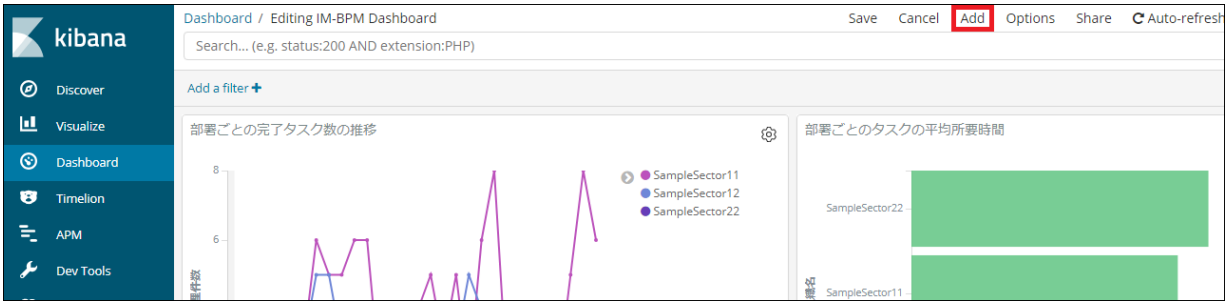


図：「IM-BPM Dashboard」



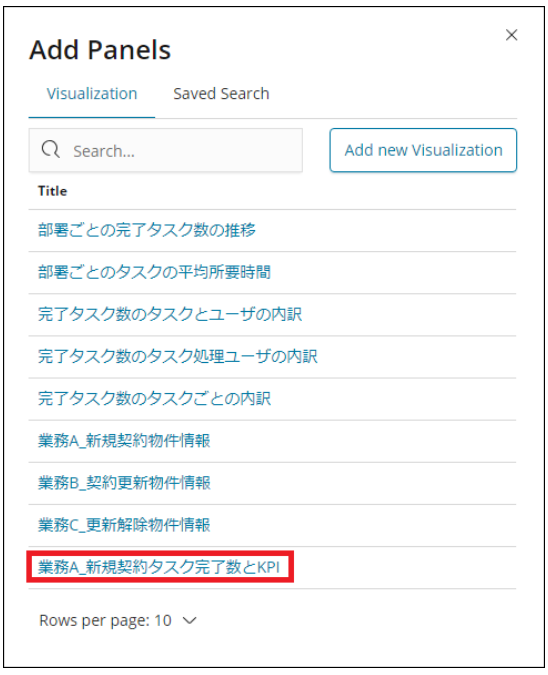
図：「Dashboard / Editing IM-BPM Dashboard」画面

3. ヘッダメニューの「Add」をクリックし、「Add Panels」を開きます。



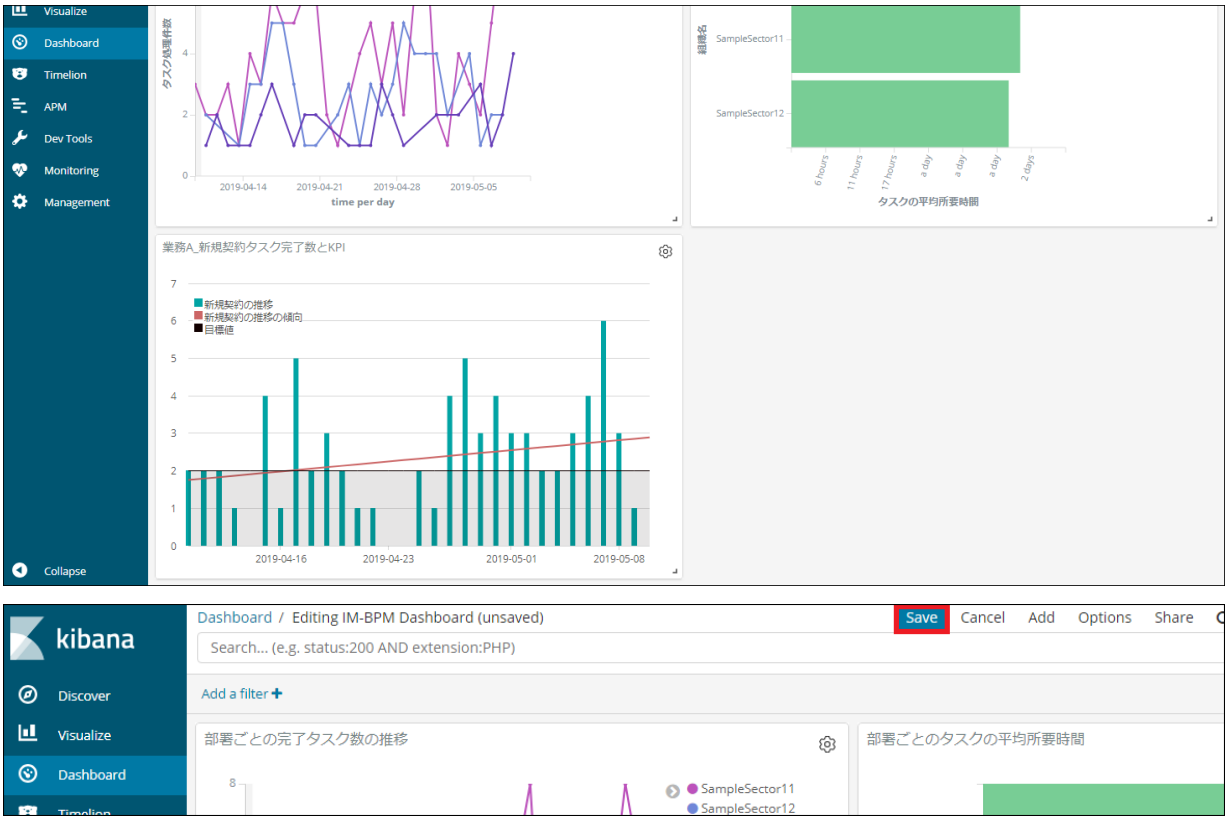
図：「Editing New Dashboard」→「Add」

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。



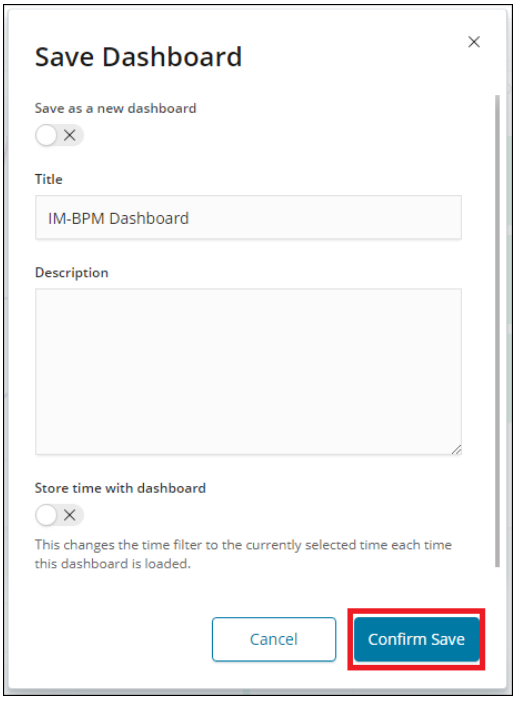
図：「Add Panels」→「Visualization」

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。



図：「Save Dashboard」

サンプル集

プロセスの作成

IM-BloomMaker

IM-BloomMakerのコンテンツをユーザ入力フォームとして利用する

IM-BPMのプロセス開始時のユーザ入力フォームおよびタスク処理時のユーザ入力フォームをIM-BloomMakerで作成するサンプルです。

図：「プロセスの開始」画面

図：「タスクの処理」画面

- 本サンプルの確認方法
- 本サンプルの構成資材
- ユーザモジュール定義情報（**im_bpm_tutorial_guide_bloommaker_process_and_task_form-8.0.0.imm**）の詳細
  - 前処理で実行されるJavaクラス
  - プロセスのリスナで実行されるJavaクラス
  - タスクのリスナで実行されるJavaクラス
- e Builder モジュール・プロジェクト定義情報（**im_bpm_tutorial_guide_bloommaker_process_and_task_form-8.0.0-sources.zip**）の詳細
- IM-BPM定義情報（**im_bpm-bloommaker_process_and_task_form.zip**）の詳細
  - IM-BPMのプロジェクト
  - IM-BPMのプロセス定義
  - IM-BPMのデプロイメント
- IM-Repository定義情報（**im-repository-export-data-bloommaker_process_and_task_form.xlsx**）の詳細
- IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_process_and_task_form.zip**）の詳細
  - IM-BloomMakerのコンテンツ
  - IM-BloomMakerのルーティング
- IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_process_and_task_form-BM.xml**）の詳細

注意

本サンプルは、intra-mart Accel Platformに「IM-BloomMaker for Accel Platform」モジュールが組み込まれている環境上で動作する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 : 2020 Summer(Zephirine)
- IM-BPM for Accel Platform : 8.0.12
- IM-BloomMaker for Accel Platform : 8.0.3

本サンプルの確認方法

1. 「*本サンプルの構成資材*」 の資材をそれぞれ組み込み／インポートします。
2. 全ての資材のインポートが完了した後、「サイトマップ」→「BPM」→「プロセス開始一覧」より「プロセス開始一覧」画面へ遷移します。

3. プロセス「bloommaker_process_and_task_form」の「プロセス開始」をクリックすることで、IM-BloomMakerを使用して作成された「プロセスの開始」画面が表示されます。この画面ではIM-Repositoryの「辞書項目」の「辞書項目の制約」などを利用した項目のバリデーションが行われます。

4. 「プロセスの開始」画面の表示が行われる際、IM-BloomMakerのルーティングに定義された前処理として、Javaクラス「sample.bloommaker_process_and_task_form.BPMProcessStartPreprocessor」が実行され、権限の確認や画面へのパラメータの検証、初期表示用の情報の取得などが行われます。

5. 「プロセスの開始」画面にて「業務キー」、「最初のタスクの担当者」、「説明」を入力し、「プロセス開始」ボタンをクリックすることで、プロセスが開始されます。

6. プロセスの開始時にプロセスの「start」イベントに設定されたリスナクラス「sample.bloommaker_process_and_task_form.BPMProcessStartValidationListener」が実行され、画面入力値由来の変数の検証が行われます。 ここでもIM-Repositoryの「辞書項目」の「辞書項目の制約」が使用されます。

7. 次に、「プロセスの開始」画面の「最初のタスクの担当者」に指定したユーザにてログインし、「サイトマップ」→「BPM」→「タスク一覧」より「タスク一覧」画面へ遷移します。

8. プロセス定義「bloommaker_process_and_task_form」のタスク「ユーザタスク」の処理を行うことで、「プロセスの開始」画面と同様にIM-BloomMakerを使用して作成された「タスクの処理」画面が表示されます。

9. 「プロセスの開始」画面と同様に、「タスクの処理」画面の表示が行われる際、IM-BloomMakerのルーティングに定義された前処理として、Javaクラス「sample.bloommaker_process_and_task_form.BPMTaskCompletePreprocessor」が実行され、権限の確認や画面へのパラメータの検証、初期表示用の情報の取得などが行われます。

10. プロセスの開始時と同様に、タスクの完了処理時にもタスクの「complete」イベントに設定されたリスナクラス「sample.bloommaker_process_and_task_form.BPMTaskCompleteValidationListener」が実行され、画面入力値由来の変数の検証が行われます。

本サンプルの構成資材

- ユーザモジュール、e Builder モジュール・プロジェクト定義情報
  im_bpm_tutorial_guide_bloommaker_process_and_task_form-8.0.0.imm
  im_bpm_tutorial_guide_bloommaker_process_and_task_form-8.0.0-sources.zip
- IM-BPM定義情報
  im_bpm-bloommaker_process_and_task_form.zip
- IM-Repository定義情報
  im-repository-export-data-bloommaker_process_and_task_form.xlsx
- IM-BloomMaker定義情報
  im_bloommaker-data-bloommaker_process_and_task_form.zip
- IM-Authz（認可） ポリシー - XML形式定義情報
  authz-policy-bloommaker_process_and_task_form-BM.xml

> **コラム**
>
> ユーザモジュールの組込みについて
>
> 組込み手順は「intra-mart Accel Platform セットアップガイド」-「作成したユーザモジュールを組み込む方法」を参照してください。

> **コラム**
>
> e Builder モジュール・プロジェクトについて
>
> e Builder モジュール・プロジェクトについては「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」を参照してください。

> **コラム**
>
> IM-BPM定義情報のインポートについて
>
> インポート手順は「IM-BPM ユーザ操作ガイド」-「インポート」を参照してください。

> **コラム**
>
> IM-Repository定義情報のインポートについて
>
> インポート手順は「IM-Repository ユーザ操作ガイド」-「辞書のインポートを行う」を参照してください。

コラム

IM-BloomMaker定義情報のインポートについて

インポート手順は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」-「定義ファイルをインポートする」を参照してください。

コラム

IM-Authz（認可） ポリシー - XML形式定義情報のインポートについて

インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を
パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
また、ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

ユーザモジュール定義情報（**im_bpm_tutorial_guide_bloommaker_process_and_task_form-8.0.0.imm**）の詳細

IM-BloomMakerで定義したユーザ入力フォームへのルーティング時の前処理（サーバサイド処理）や、プロセスリスナを使用したサーバサイドバリデーションを行うJavaクラスのサンプルを格納したユーザモジュール資材です。

前処理で実行されるJavaクラス

- sample.bloommaker_process_and_task_form.BPMProcessStartPreprocessor.java
  コンテンツ「【チュートリアル】start_process_form」へのルーティング時の前処理として実行されます。
  プロセス開始後に遷移するコールバックパスの妥当性の検証や、必須パラメータの確認、ユーザ権限の妥当性の検証、初期表示情報の取得などの処理を行います。
- sample.bloommaker_process_and_task_form.BPMTaskStartPreprocessor.java
  コンテンツ「【チュートリアル】complete_task_form」へのルーティング時の前処理として実行されます。
  タスク処理後に遷移するコールバックパスの妥当性の検証や、必須パラメータの確認、ユーザ権限の妥当性の検証、初期表示情報の取得などの処理を行います。

コラム

ユーザ権限の妥当性の検証について

IM-BPMではユーザがプロセスインスタンスの関係者であるか、権限を確認する機構が用意されています。
本サンプルでは、コンテンツにルーティングする際の前処理として、権限のチェックを行っています。

IM-BPMの関係者の詳細については「IM-BPM 仕様書」-「関係者権限」を参照してください。
IM-BPMとIM-BloomMakerの画面連携の詳細については「IM-BPM 仕様書」-「IM-BloomMaker画面連携」を参照してください。
IM-BPMとIM-BloomMakerの画面連携時のパラメータの詳細については「IM-BPM 仕様書」-「パラメータ」を参照してください。
IM-BPMとIM-BloomMakerの画面連携時の権限の検証の詳細については「IM-BPM 仕様書」-「検証」を参照してください。

プロセスのリスナで実行されるJavaクラス

- sample.bloommaker_process_and_task_form.BPMProcessStartValidationListener.java
  プロセス「bloommaker_process_and_task_form」の実行時のプロセスリスナとして実行されます。
  プロセス開始時のサーバサイドバリデーションとして、必須項目の有無の検証や、IM-Repositoryの辞書項目を使用しての画面入力値由来の変数の文字数の検証を行います。

タスクのリスナで実行されるJavaクラス

- sample.bloommaker_process_and_task_form.BPMTaskCompleteValidationListener.java
  プロセス「bloommaker_process_and_task_form」の実行時のタスクリスナとして実行されます。
  タスク処理時のサーバサイドバリデーションとして、必須項目の有無の検証や、IM-Repositoryの辞書項目を使用しての画面入力値由来の変数の文字数の検証を行います。

コラム

IM-Repositoryの辞書項目を使用したバリデーションについて

IM-Repositoryの辞書項目を使用したバリデーションの詳細については「IM-Repository拡張プログラミングガイド」-「制約用途に指定した制約に適合するかどうかを確認する」を参照してください。
IM-Repositoryの辞書項目については後述の「*IM-Repository定義情報（im-repository-export-data-bloommaker_process_and_task_form.xlsx）の詳細*」を参照してください。

e Builder モジュール・プロジェクト定義情報（**im_bpm_tutorial_guide_bloommaker_process_and_task_form-8.0.0-sources.zip**）の詳細

先述のユーザモジュール「bloommaker_process_and_task_form-8.0.0.imm」を作成するためのe Builder モジュール・プロジェクトです。各Javaクラスのソースコードなどが格納されています。

IM-BPM定義情報（**im_bpm-bloommaker_process_and_task_form.zip**）の詳細

プロセス開始時およびタスク処理時にIM-BloomMakerのフォームを呼び出すサンプルプロセスおよび、プロセスを格納するプロジェクト、プロセスのデプロイ情報を含む資材です。
本サンプルでは、上記の入力フォームの呼び出しのほか、プロセスやタスクのリスナを使用したサーバサイドバリデーションも行います。

IM-BPMのプロジェクト

- 「【チュートリアル】bloommaker_process_and_task_form」
  本サンプルで使用するプロセスを格納するためのプロジェクトです。

IM-BPMのプロセス定義

- 「bloommaker_process_and_task_form」
  プロセスの開始時や、タスクの処理時にIM-BloomMakerで作成された入力フォームを呼び出すプロセス定義です。
  プロジェクト「【チュートリアル】bloommaker_process_and_task_form」に格納されています。

> **コラム**
>
> プロセスの入力フォームについて
>
> プロセスの開始イベント、および、ユーザタスクに設定するフォームキーに対し画面連携方式を設定することが可能です。
> 本サンプルではこの機能を使用して、IM-BloomMakerのコンテンツと連携を行っています。
> フォームキーの詳細については「IM-BPM 仕様書」-「フォームキー」を参照してください。

> **コラム**
>
> プロセスのリスナについて
>
> 本サンプルでは、プロセス開始時のプロセスの「start」イベントおよびタスク処理時のタスクの「complete」イベントにてリスナを使用したバリデーションを実行しています。
> プロセスのリスナについての詳細は「IM-BPM プログラミングガイド」-「リスナ」を参照してください。

IM-BPMのデプロイメント

- 「【チュートリアル】bloommaker_process_and_task_form-deployment」
  本サンプルで使用するプロセス定義のデプロイメント定義情報です。

> **コラム**
>
> プロセスの開始について
>
> プロセスの開始手順は「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。

IM-Repository定義情報（**im-repository-export-data-bloommaker_process_and_task_form.xlsx**）の詳細

プロセス開始時のユーザ入力フォームとタスク処理時のユーザ入力フォームにて、項目のバリデーションに使用する辞書情報を含む資材です。
辞書項目の制約を使用したバリデーションについては「*ユーザモジュール定義情報（im_bpm_tutorial_guide_bloommaker_process_and_task_form-8.0.0.imm）の詳細*」を参照してください。

> **コラム**
>
> 辞書項目について
>
> 辞書項目の詳細については「IM-Repository ユーザ操作ガイド」-「辞書項目」を参照してください。

> **コラム**
>
> 辞書項目の制約について
>
> 辞書項目の制約の詳細については「IM-Repository ユーザ操作ガイド」-「辞書項目の制約」を参照してください。

IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_process_and_task_form.zip**）の詳細

プロセス開始時のユーザ入力フォームとタスク処理時のユーザ入力フォームの定義情報と、それらの入力フォームへのルーティング情報の定義が含まれる資材です。

IM-BloomMakerのコンテンツ

- 「BPMチュートリアル」-「【チュートリアル】start_process_form」
  プロセスの開始時に呼び出されるユーザ入力フォーム画面の定義情報です。IM-BPM REST API「プロセスインスタンス登録」を呼び出すアクションの定義やボタンクリックなどの画面イベントの定義、レイアウト等の情報が含まれます。

コラム

IM-BPM REST API「プロセスインスタンス登録」について

IM-BPM REST API「プロセスインスタンス登録」の詳細について「APIドキュメント」-「プロセスインスタンス登録」を参照してください。

- 「BPMチュートリアル」-「【チュートリアル】complete_task_form」
  タスクの処理時に呼び出されるユーザ入力フォーム画面の定義情報です。IM-BPM REST API「タスク登録」を呼び出すアクションの定義やボタンクリックなどの画面イベントの定義、レイアウト等の情報が含まれます。

コラム

IM-BPM REST API「タスク登録」について

IM-BPM REST API「タスク登録」の詳細について「APIドキュメント」-「タスク登録」を参照してください。

コラム

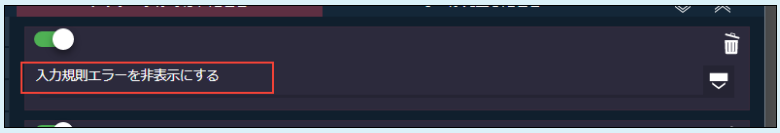
IM-Repositoryを使用したフォームのバリデーションについて

本サンプルではIM-Repositoryの辞書項目を使用し、フォーム上の入力フィールドの最大長や属性などのメタデータを設定します。

IM-Repositoryの辞書項目の新規登録については「IM-Repository ユーザ操作ガイド」-「辞書項目を新規登録する」を参照してください。

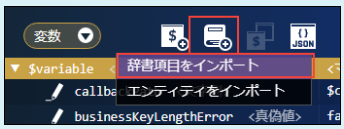
IM-BloomMakerでは、これらのメタデータを使用して項目のバリデーションを行えます。
ここでは、IM-Repositoryの辞書項目を使用したフォームのバリデーションの設定手順の一例を示します。

1. コンテナの初期表示アクションにて、「入力規則エラーを非表示にする」を実行します。



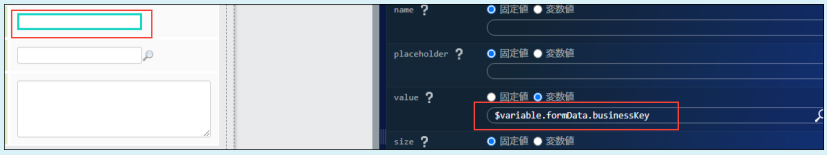
図：「入力規則エラーを非表示にする」

2. 「変数」タブの「IM-Repository から作成」-「辞書項目をインポート」よりIM-Repositoryの辞書項目を変数定義としてインポートします。



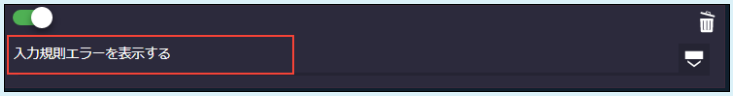
図：「IM-Repository から作成」-「辞書項目をインポート」

3. 上記の設定後、作成した変数をエレメントにバインドします。

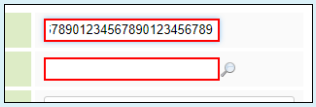


図：「変数をエレメントにバインド」

4. アクション「入力規則エラーを表示する」を実行することにより、制約違反となっている項目の周囲が赤く縁どられます。また、アクションの実行条件として、「変数〇が入力規則に即していないとき」を指定することでエラーメッセージの表示などを制御することもできます。



図：「入力規則エラーを表示する」



図：「入力エラーとなっている状態」

図：「変数〇が入力規則に即していないとき」

IM-BloomMakerのルーティング

- 「BPMチュートリアル」-「【チュートリアル】routing_start_process_form」
  コンテンツ「【チュートリアル】start_process_form」へアクセスするためのURLの指定や、アクセスする際に呼び出される前処理などの情報を定義するルーティング定義情報です。
- 「BPMチュートリアル」-「【チュートリアル】routing_complete_task_form」
  コンテンツ「【チュートリアル】complete_task_form」へアクセスするためのURLの指定や、アクセスする際に呼び出される前処理などの情報を定義するルーティング定義情報です。

IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_process_and_task_form-BM.xml**）の詳細

IM-BloomMakerのルーティング「【チュートリアル】routing_start_process_form」および「【チュートリアル】routing_complete_task_form」に対する認可設定です。
本サンプルでは「IM-BPM管理者」および「IM-BPMユーザ」に対し「参照」権限を付与します。

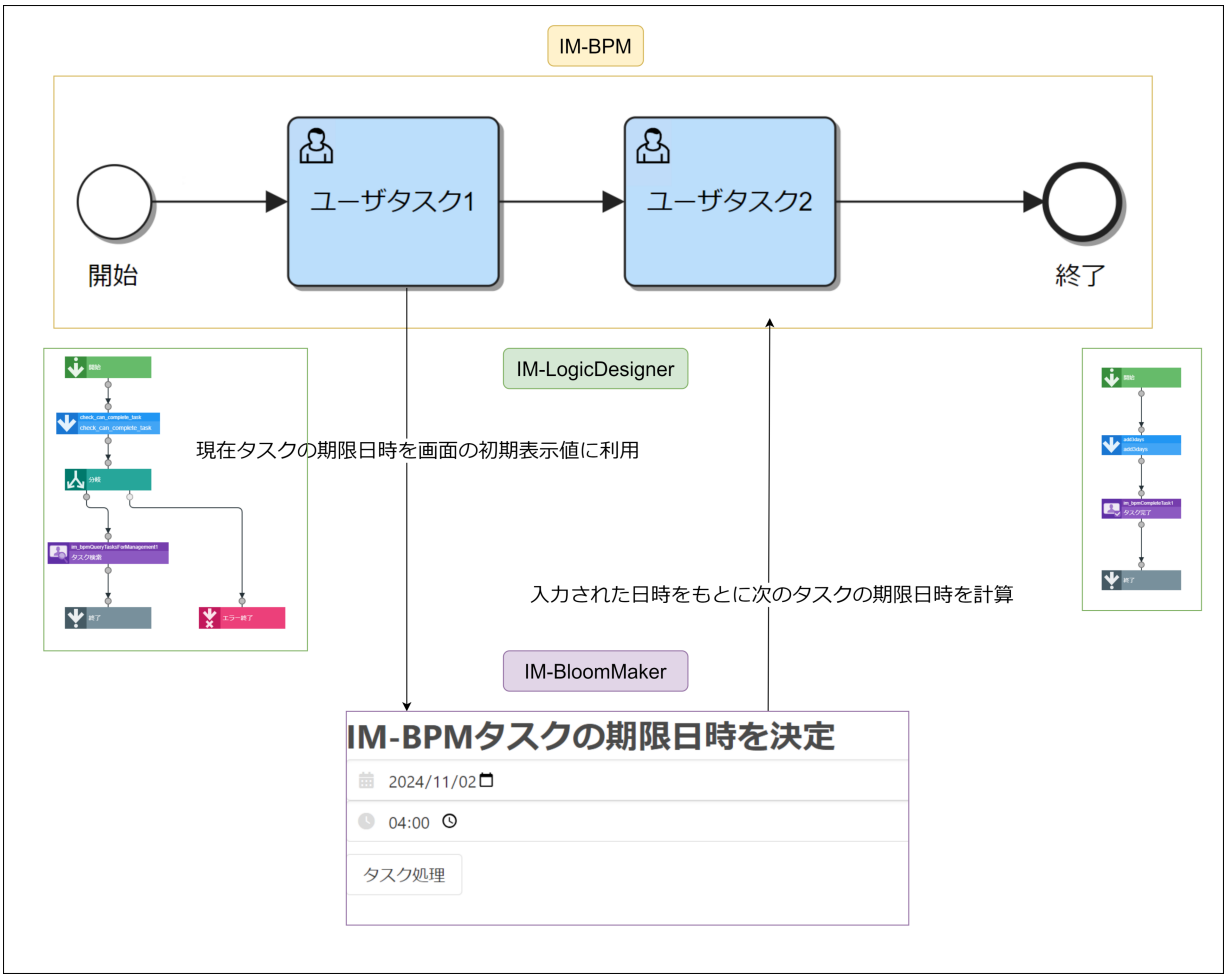
> コラム
>
> IM-BloomMakerのルーティングの認可について
>
> ルーティングの認可設定手順は「IM-BloomMaker for Accel Platform チュートリアルガイド」-「ルーティングの認可を設定する」を参照してください。

IM-BloomMakerで作成した画面の入力値をもとにタスクの期限日時を設定する

IM-BPMのタスク処理時のユーザ入力フォームをIM-BloomMakerで作成し、その入力値をもとに次のタスクの期限日時を設定するサンプルです。

本サンプルの要旨

- IM-BloomMakerを使用して画面を作成します。
- IM-LogicDesignerを使用してIM-BloomMakerで入力された値をもとに次タスクの期限日時を設定します。



図：概要図

- 本サンプルの確認方法
- 本サンプルの構成資材
- IM-BPM定義情報（**im_bpm-bloommaker_set_duedate.zip**）の詳細
  - IM-BPMのプロジェクト
  - IM-BPMのプロセス定義
  - IM-BPMのデプロイメント
- IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_set_duedate.zip**）の詳細
  - IM-LogicDesignerのロジックフロー定義
  - IM-LogicDesignerのルーティング定義
- IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_set_duedate.zip**）の詳細
  - IM-BloomMakerのコンテンツ
  - IM-BloomMakerのルーティング
- IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_set_duedate.xml**）の詳細

注意

本サンプルは、intra-mart Accel Platformに「IM-BloomMaker for Accel Platform」モジュールが組み込まれている環境上で動作する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 : 2024 Spring(Iris)
- IM-BPM for Accel Platform : 8.0.16
- IM-BloomMaker for Accel Platform : 8.0.13

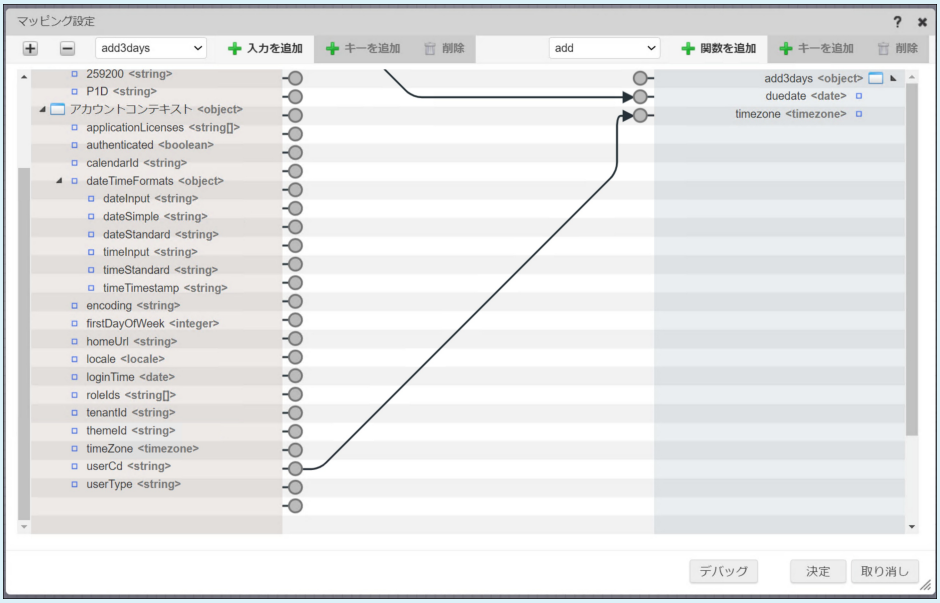
本サンプルの確認方法

1. 「*本サンプルの構成資材*」 の資材をそれぞれ組み込み／インポートしてください。
2. 全ての資材のインポートが完了した後、「サイトマップ」→「BPM」→「プロセス開始一覧」より「プロセス開始一覧」画面へ遷移します。
3. プロセス「【チュートリアル】process_bloommaker_set_duedate」の「プロセス開始」をクリックしプロセスを開始します。
4. 次に、「プロセスの開始」画面の「最初のタスクの担当者」に指定したユーザにてログインし、「サイトマップ」→「BPM」→「タスク一覧」より「タスク一覧」画面へ遷移します。
5. プロセス定義「【チュートリアル】process_bloommaker_set_duedate」のタスク「ユーザタスク1」の処理を行うことで、IM-BloomMakerを使用して作成された「タスクの処理」画面が表示されます。
6. 「タスクの処理」画面の表示が行われる際IM-BloomMakerのルーティングに定義された前処理として、IM-LogicDesignerのロジックフロー「【チュートリアル】preprocessing_bloommaker_set_duedate」が実行され、権限の確認と初期表示用の情報の取得が行われます。
7. 「タスクの処理」画面の「タスク処理」ボタン押下時に、IM-BloomMakerのアクションに設定したIM-LogicDesignerのロジックフロー「【チュートリアル】complete_bloommaker_set_duedate」が実行され、画面入力値由来の変数をもとにして次のタスクの期限日時に設定します。

コラム

IM-LogicDesignerのロジックフロー「【チュートリアル】complete_bloommaker_set_duedate」での日付の加算について

夏時間のある地域では、夏時間の開始や終了の際には一日の長さが23時間や25時間です。
ユーザのタイムゾーンで上記を考慮した日付加算を行うためには、スクリプト開発向けAPIであるDateTimeオブジェクトを使用する必要があります。
日時加算のロジックフローではDatetimeオブジェクトを利用したjavascriptユーザ定義を使用し、javascriptユーザ定義のマッピング設定にてアカウントコンテキストのタイムゾーンをマッピングさせています。



図：「add3daysタスク」 - 「マッピング設定」

Datetimeオブジェクトを利用したjavascriptユーザ定義については「*タスクリスナを利用して、ユーザタスクの期限日時を直前ユーザタスクの期限日時から3日後に設定する*」を参照してください。
スクリプト開発向けAPIであるDateTimeオブジェクトについては「intra-mart Accel Platform スクリプト開発向けim-BizAPI」-「DateTimeオブジェクト」を参照してください。
intra-mart Accel Platform のタイムゾーンや夏時間、日付と時刻の形式に関する考え方や仕様、制限事項については「intra-mart Accel Platform タイムゾーン仕様書」を参照してください。

注意

IM-BloomMakerやIM-BPMの夏時間への対応について

IM-BloomMakerやIM-BPMの夏時間への対応は2024 Spring(Iris)以降のバージョンでのみサポートされています。
2023 Autumn(Hollyhock)以前のバージョンでは、IM-BloomMakerやIM-BPMでの日時の表示や入力において夏時間の考慮が行われていません。

コラム

IM-BPMへの日時情報の受け渡しについて

IM-LogicDesignerでのIM-BPMとの日時情報の受け渡しには *date* 型の変数の使用を推奨します。
IM-BloomMakerからIM-LogicDesignerへ日時情報を渡す方法については、「intra-mart Accel Platform ローコード開発チュートリアルガイド」-「日時を保存する」を参照してください。
IM-LogicDesignerからIM-BloomMakerへ日時情報を渡す方法については、「intra-mart Accel Platform ローコード開発チュートリアルガイド」-「保存されている日時を画面に表示する」を参照してください。

本サンプルの構成資材

- IM-BPM定義情報
  im_bpm-bloommaker_set_duedate.zip
- IM-BPM定義情報（英語版）
  im_bpm-bloommaker_set_duedate_en.zip
- IM-LogicDesigner定義情報
  im_logicdesigner-data-bloommaker_set_duedate.zip
- IM-BloomMaker定義情報
  im_bloommaker-data-bloommaker_set_duedate.zip
- IM-Authz（認可） ポリシー - XML形式定義情報
  authz-policy-bloommaker_set_duedate.xml

> **コラム**
>
> IM-BPM定義情報のインポートについて
>
> インポート手順は「IM-BPM ユーザ操作ガイド」-「インポート」を参照してください。

> **コラム**
>
> IM-LogicDesigner定義情報のインポートについて
>
> インポート手順は「IM-LogicDesigner ユーザ操作ガイド」-「インポートを行う」を参照してください。

> **コラム**
>
> IM-BloomMaker定義情報のインポートについて
>
> インポート手順は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」-「定義ファイルをインポートする」を参照してください。

> **コラム**
>
> IM-Authz（認可） ポリシー - XML形式定義情報のインポートについて
>
> インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を参照してください。
> パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
> また、ジョブネット「認可（ポリシー）インポート」の実行時に実行パラメータfileを追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
> または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

IM-BPM定義情報（**im_bpm-bloommaker_set_duedate.zip**）の詳細

タスク処理時にIM-BloomMakerのフォームを呼び出すサンプルプロセス、プロセスを格納するプロジェクト、および、プロセスのデプロイ情報を含む資材です。

IM-BPMのプロジェクト

- 「【チュートリアル】project_bloommaker_set_duedate」
  本サンプルで使用するプロセスを格納するためのプロジェクトです。

IM-BPMのプロセス定義

- 「【チュートリアル】process_bloommaker_set_duedate」
  タスクの処理時にIM-BloomMakerで作成された入力フォームを呼び出すプロセス定義です。
  プロジェクト「【チュートリアル】project_bloommaker_set_duedate」に格納されています。

  > **コラム**
  >
  > ユーザタスクの入力フォームについて
  >
  > ユーザタスクに設定するフォームキーに対し画面連携方式を設定することが可能です。
  > 本サンプルではこの機能を使用して、IM-BloomMakerのコンテンツと連携を行っています。
  > フォームキーの詳細については「IM-BPM 仕様書」-「フォームキー」を参照してください。

IM-BPMのデプロイメント

- 「【チュートリアル】deploy_bloommaker_set_duedate」
  本サンプルで使用するプロセス定義のデプロイメント定義情報です。

  > **コラム**
  >
  > プロセスの開始について
  >
  > プロセスの開始手順は「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。

IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_set_duedate.zip**）の詳細

IM-BloomMakerのルーティング「【チュートリアル】routing_bloommaker_set_duedate」の前処理を行うIM-LogicDesignerのロジックフローや、IM-BloomMakerのコンテンツ「【チュートリアル】content_bloommaker_set_duedate」のアクション内で呼び出されるIM-LogicDesignerのロジックフローとそのロジックフローへのルーティング定義を含む資材です。
資材それぞれに対して英語版があります。

IM-LogicDesignerのロジックフロー定義

- 「BPMチュートリアル」-「【チュートリアル】preprocessing_bloommaker_set_duedate」
  IM-BloomMakerで作成したルーティング「【チュートリアル】routing_bloommaker_set_duedate」の前処理として実行されるIM-LogicDesignerのロジックフローです。
- 「BPMチュートリアル」-「【チュートリアル】complete_bloommaker_set_duedate」
  IM-BloomMakerで作成したコンテンツ「【チュートリアル】content_bloommaker_set_duedate」の「タスク処理」ボタン押下時のアクションとして実行されるIM-LogicDesignerのロジックフローです。

> **コラム**
>
> 「【チュートリアル】preprocessing_bloommaker_set_duedate」について
>
> 「【チュートリアル】preprocessing_bloommaker_set_duedate」ではログインユーザがタスクの実行権限があるかの検証も行っています。
> 実行権限の検証については「IM-BPM 仕様書」-「検証」を参照してください。

IM-LogicDesignerのルーティング定義

- 「routing_bloommaker_set_duedate」
  IM-LogicDesignerのロジックフロー「【チュートリアル】complete_bloommaker_set_duedate」へアクセスするためのURLなどの情報を定義するルーティング定義情報です。

IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_set_duedate.zip**）の詳細

プロセス開始時のユーザ入力フォームとタスク処理時のユーザ入力フォームの定義情報と、それらの入力フォームへのルーティング情報の定義が含まれる資材です。

IM-BloomMakerのコンテンツ

- 「BPMチュートリアル」-「【チュートリアル】content_bloommaker_set_duedate」
  IM-LogicDesignerのロジックフロー「【チュートリアル】complete_bloommaker_set_duedate」を呼び出すアクションの定義やボタンクリックなどの画面イベントの定義、レイアウト等の情報を含む、画面の定義情報です。

IM-BloomMakerのルーティング

- 「BPMチュートリアル」-「【チュートリアル】routing_bloommaker_set_duedate」
  コンテンツ「【チュートリアル】content_bloommaker_set_duedate」へアクセスするためのURLの指定や、アクセスする際に呼び出される前処理などの情報を定義するルーティング定義情報です。
  本サンプルでは前処理としてIM-LogicDesignerのロジックフロー「【チュートリアル】preprocessing_bloommaker_set_duedate」を呼び出します。

IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_set_duedate.xml**）の詳細

IM-BloomMakerのルーティング「【チュートリアル】routing_bloommaker_set_duedate」とIM-LogicDesignerのルーティング「routing_bloommaker_set_duedate」に対する認可設定です。
本サンプルでは「IM-BPM管理者」と「IM-BPMユーザ」に対し「参照」権限や「実行」権限を付与します。

> **コラム**
>
> IM-BloomMaker、IM-LogicDesignerのルーティングの認可について
>
> IM-BloomMakerのルーティングの認可設定手順は「IM-BloomMaker for Accel Platform チュートリアルガイド」-「ルーティングの認可を設定する」を参照してください。
> IM-LogicDesignerのルーティングの認可設定手順は「IM-LogicDesigner ユーザ操作ガイド」-「フロールーティングの認可設定」を参照してください。

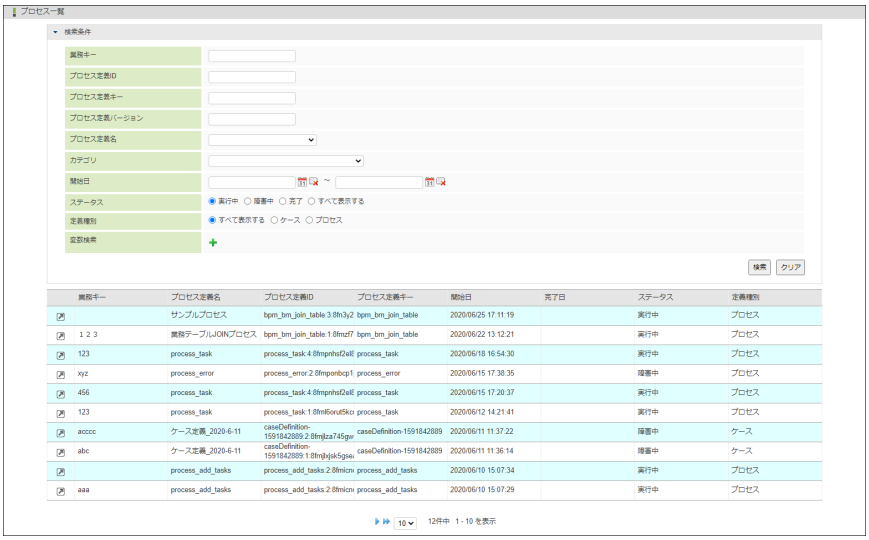
プロセスの管理

IM-BloomMakerを利用してアドオンのプロセス一覧画面を作成する（１）

IM-BPM REST APIでプロセスインスタンスの情報を取得し、テーブルコンポーネントへデータバインディングし表示する画面をIM-BloomMakerで作成するサンプルです。

本サンプルの要旨

- IM-BloomMakerを使用して画面を作成します。
- IM-BPM REST APIをIM-BloomMakerを使用して作成した画面より呼び出し、取得した情報を表示します。



図：「プロセス一覧」

- 本サンプルの確認方法
- 本サンプルの構成資材
- IM-LogicDesigner 定義情報（**im_logicdesigner-data_timezone.zip**）の詳細
  - IM-LogicDesignerのロジックフロー定義
  - ユーザ定義
- IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_list.zip**）の詳細
  - IM-BloomMakerのコンテンツ
  - IM-BloomMakerのルーティング
- IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list-BM.xml**）の詳細

注意

本サンプルは、intra-mart Accel Platformに「IM-BloomMaker for Accel Platform」モジュールが組み込まれている環境上で動作する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 : 2020 Summer(Zephirine)
- IM-BPM for Accel Platform : 8.0.12
- IM-BloomMaker for Accel Platform : 8.0.3

本サンプルの確認方法

1. 「*本サンプルの構成資材*」 の資材をそれぞれ組み込み／インポートしてください。
2. 全ての資材のインポートが完了したのち、「サイトマップ」→「BloomMaker」→「ルーティング定義一覧」より「ルーティング定義一覧」画面へ遷移します。
3. IM-BloomMakerのルーティング「【チュートリアル】routing_addon_process_list」の「URL」に定義されたパスにアクセスすることで、IM-BloomMakerを使用して作成された「プロセス一覧」画面が表示されます。
4. 「プロセス一覧」画面では、IM-BPM REST APIを使用して取得された情報を閲覧できます。
5. この画面は「IM-BPM管理者」権限をもつユーザが参照できます。

本サンプルの構成資材

- IM-LogicDesigner定義情報
  im_logicdesigner-data_timezone.zip
- IM-BloomMaker定義情報
  im_bloommaker-data-bloommaker_addon_process_list.zip
- IM-Authz（認可） ポリシー - XML形式定義情報
  authz-policy-bloommaker_addon_process_list-BM.xml

コラム

IM-LogicDesigner定義情報のインポートについて

インポート手順は「IM-LogicDesigner ユーザ操作ガイド」-「インポートを行う」を参照してください。

コラム

IM-BloomMaker定義情報のインポートについて

インポート手順は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」-「定義ファイルをインポートする」を参照してください。

コラム

IM-Authz（認可） ポリシー - XML形式定義情報のインポートについて

インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を参照してください。
パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
また、ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

IM-LogicDesigner 定義情報（**im_logicdesigner-data_timezone.zip**）の詳細

ユーザのアカウント情報に紐づくタイムゾーンの設定より、タイムゾーンにもとづいたオフセット値を取得するIM-LogicDesignerのロジックフローを含む資材です。取得されたオフセット値より、システム内部で保持している時刻情報からタイムゾーンにもとづいた時刻を計算し画面表示を行います。

IM-LogicDesignerのロジックフロー定義

- 「BPMチュートリアル」-「【チュートリアル】get_user_timezone_offset」
  ユーザのコンテキスト情報からタイムゾーンにもとづいたオフセット値を取得して返却するIM-LogicDesignerのロジックフローです。

ユーザ定義

- 「BPMチュートリアル」-「【チュートリアル】get_timezone_offset」
  タイムゾーンからオフセット値を取得するためのユーザ定義タスクです。
  IM-LogicDesignerのロジックフロー「【チュートリアル】get_user_timezone_offset」内で使用されます。

IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_list.zip**）の詳細

IM-BPM REST APIとIM-BloomMakerを使用してアドオンのプロセス一覧画面の定義と、その画面へのルーティングを定義する資材です。

IM-BloomMakerのコンテンツ

- 「BPMチュートリアル」-「【チュートリアル】addon_process_list」
  IM-BPM REST API「プロセスインスタンス検索」を呼び出すアクションの定義やボタンクリックなどの画面イベントの定義、レイアウト等の情報を含む、画面の定義情報です。

コラム

IM-BPM REST API「プロセスインスタンス検索」について

IM-BPM REST API「プロセスインスタンス検索」の詳細については「APIドキュメント」-「プロセスインスタンス検索」を参照してください。

コラム

IM-BloomMakerのテーブルエレメントの使い方については、下記の記事も併せて参照してください。

IM-BloomMaker「繰り返しエレメントでの変数の使い方」について

IM-BloomMaker「繰り返しエレメントでの変数の使い方」の詳細については「intra-mart Developer Site」-「IM-BloomMaker 繰り返しエレメントでの変数の使い方」を参照してください。

IM-BloomMaker「カスタムスクリプト内で $im.resolve を使った一覧画面の作成」について

IM-BloomMaker「カスタムスクリプト内で $im.resolve を使った一覧画面の作成」の詳細については「intra-mart Developer Site」-「IM-BloomMaker のカスタムスクリプト内で $im.resolve を使った一覧画面の作成」を参照してください。

IM-BloomMakerのルーティング

- 「BPMチュートリアル」-「【チュートリアル】routing_addon_process_list」
  コンテンツ「【チュートリアル】addon_process_list」へアクセスするためのURLの指定や、アクセスする際に呼び出される前処理などの情報を定義するルーティング定義情報です。
  本サンプルでは前処理としてIM-LogicDesignerのロジックフロー「【チュートリアル】get_user_timezone_offset」を呼び出します。
  「【チュートリアル】get_user_timezone_offset」は日付の出力に使用するユーザ毎のタイムゾーンにもとづいたオフセット値を取得する処理です。詳細は「*IM-LogicDesigner 定義情報（im_logicdesigner-data_timezone.zip）の詳細*」を参照してください。

IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list-BM.xml**）の詳細

IM-BloomMakerのルーティング「【チュートリアル】routing_addon_process_list」に対する認可設定です。
本サンプルでは「IM-BPM管理者」に対し「参照」権限を付与します。

コラム

IM-BloomMakerのルーティングの認可について

ルーティングの認可設定手順は「IM-BloomMaker for Accel Platform チュートリアルガイド」-「ルーティングの認可を設定する」を参照してください。

IM-BloomMakerを利用してアドオンのプロセス一覧画面を作成する（２）

IM-LogicDesignerのロジックフローでユーザが参照可能なプロセスインスタンスの情報を取得し、テーブルコンポーネントへデータバインディングし表示する画面をIM-BloomMakerで作成するサンプルです。

本サンプルの要旨

- IM-BloomMakerを使用して画面を作成します。
- ユーザが参照可能なプロセスインスタンスの情報を取得する処理をIM-LogicDesignerを使用して定義し、IM-BloomMakerを使用して作成した画面より呼び出し、取得した情報を表示します。

図：「プロセス一覧」

- 本サンプルの確認方法
- 本サンプルの構成資材
- IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_addon_process_list_user.zip**）の詳細
  - IM-LogicDesignerのロジックフロー定義
  - IM-LogicDesignerのロジックフローのルーティング定義
- IM-LogicDesigner 定義情報（**im_logicdesigner-data_timezone.zip**）の詳細
  - IM-LogicDesignerのロジックフロー定義
  - ユーザ定義
- IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_list_user.zip**）の詳細
  - IM-BloomMakerのコンテンツ
  - IM-BloomMakerのルーティング
- IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_user-BM.xml**）の詳細
- IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_user-LD.xml**）の詳細

注意

本サンプルは、intra-mart Accel Platformに「IM-BloomMaker for Accel Platform」モジュールが組み込まれている環境上で動作する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 : 2020 Summer(Zephirine)
- IM-BPM for Accel Platform : 8.0.12
- IM-BloomMaker for Accel Platform : 8.0.3

本サンプルの確認方法

1. 「*本サンプルの構成資材*」 の資材をそれぞれ組み込み／インポートしてください。
2. 全ての資材のインポートが完了したのち、「サイトマップ」→「BloomMaker」→「ルーティング定義一覧」より「ルーティング定義一覧」画面へ遷移します。
3. IM-BloomMakerのルーティング「【チュートリアル】routing_addon_process_list_user」の「URL」に定義されたパスにアクセスすることで、IM-BloomMakerを使用して作成された「プロセス一覧」画面が表示されます。
4. 「プロセス一覧」画面では、IM-LogicDesignerのロジックフローで取得された、画面を開いているユーザが参照可能なプロセスインスタンスの情報を閲覧できます。
5. この画面は「IM-BPM管理者」または「IM-BPMユーザ」権限をもつユーザが参照できます。

本サンプルの構成資材

- IM-LogicDesigner定義情報
  im_logicdesigner-data-bloommaker_addon_process_list_user.zip
  im_logicdesigner-data_timezone.zip
- IM-BloomMaker定義情報
  im_bloommaker-data-bloommaker_addon_process_list_user.zip
- IM-Authz（認可） ポリシー - XML形式定義情報
  authz-policy-bloommaker_addon_process_list_user-BM.xml
  authz-policy-bloommaker_addon_process_list_user-LD.xml

コラム

IM-LogicDesigner定義情報のインポートについて

インポート手順は「IM-LogicDesigner ユーザ操作ガイド」-「インポートを行う」を参照してください。

コラム

IM-BloomMaker定義情報のインポートについて

インポート手順は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」-「定義ファイルをインポートする」を参照してください。

コラム

IM-Authz（認可） ポリシー - XML形式定義情報のインポートについて

インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を参照してください。
パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
また、ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_addon_process_list_user.zip**）の詳細

先述のアドオン画面より呼び出され、ユーザが参照可能なプロセスの一覧情報を返却するIM-LogicDesignerのロジックフローとそのIM-LogicDesignerのロジックフローへのルーティング定義を含む資材です。

IM-LogicDesignerのロジックフロー定義

- 「【チュートリアル】search_process_instance」
  ユーザが参照可能なプロセスインスタンスの情報を取得して返却するIM-LogicDesignerのロジックフローです。

IM-LogicDesignerのロジックフローのルーティング定義

- 「process_instance」
  IM-LogicDesignerのロジックフロー「【チュートリアル】search_process_instance」へアクセスするためのURLなどの情報を定義するルーティング定義情報です。

IM-LogicDesigner 定義情報（**im_logicdesigner-data_timezone.zip**）の詳細

ユーザのアカウント情報に紐づくタイムゾーンの設定より、タイムゾーンにもとづいたオフセット値を取得するIM-LogicDesignerのロジックフローを含む資材です。取得されたオフセット値より、システム内部で保持している時刻情報からタイムゾーンにもとづいた時刻を計算し画面表示を行います。

IM-LogicDesignerのロジックフロー定義

- 「BPMチュートリアル」-「【チュートリアル】get_user_timezone_offset」
  ユーザのコンテキスト情報からタイムゾーンにもとづいたオフセット値を取得して返却するIM-LogicDesignerのロジックフローです。

ユーザ定義

- 「BPMチュートリアル」-「【チュートリアル】get_timezone_offset」
  タイムゾーンからオフセット値を取得するためのユーザ定義タスクです。
  IM-LogicDesignerのロジックフロー「【チュートリアル】get_user_timezone_offset」内で使用されます。

IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_list_user.zip**）の詳細

IM-LogicDesignerとIM-BloomMakerを使用してユーザが参照可能なプロセスを表示するアドオンのプロセス一覧画面の定義と、その画面へのルーティングを定義する資材です。

IM-BloomMakerのコンテンツ

- 「BPMチュートリアル」-「【チュートリアル】addon_process_list_user」
  IM-LogicDesignerのロジックフロー「【チュートリアル】search_process_instance」を呼び出すアクションの定義やボタンクリックなどの画面イベントの定義、レイアウト等の情報を含む、画面の定義情報です。

コラム

IM-BloomMakerのテーブルエレメントの使い方については、下記の記事も併せて参照してください。

IM-BloomMaker「繰り返しエレメントでの変数の使い方」について

IM-BloomMaker「繰り返しエレメントでの変数の使い方」の詳細については「intra-mart Developer Site」-「IM-BloomMaker 繰り返しエレメントでの変数の使い方」を参照してください。

IM-BloomMaker「カスタムスクリプト内で $im.resolve を使った一覧画面の作成」について

IM-BloomMaker「カスタムスクリプト内で $im.resolve を使った一覧画面の作成」の詳細については「intra-mart Developer Site」-「IM-BloomMaker のカスタムスクリプト内で $im.resolve を使った一覧画面の作成」を参照してください。

IM-BloomMakerのルーティング

- 「BPMチュートリアル」-「【チュートリアル】routing_addon_process_list_user」
  コンテンツ「【チュートリアル】addon_process_list_user」へアクセスするためのURLの指定や、アクセスする際に呼び出される前処理などの情報を定義するルーティング定義情報です。
  本サンプルでは前処理としてIM-LogicDesignerのロジックフロー「【チュートリアル】get_user_timezone_offset」を呼び出します。
  「【チュートリアル】get_user_timezone_offset」は日付の出力に使用するユーザ毎のタイムゾーンにもとづいたオフセット値を取得する処理です。詳細は「*IM-LogicDesigner 定義情報（im_logicdesigner-data_timezone.zip）の詳細*」を参照してください。

IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_user-BM.xml**）の詳細

IM-BloomMakerのルーティング「【チュートリアル】routing_addon_process_list_user」に対する認可設定です。
本サンプルでは「IM-BPM管理者」および「IM-BPMユーザ」に対し「参照」権限を付与します。

> **コラム**
>
> IM-BloomMakerのルーティングの認可について
>
> ルーティングの認可設定手順は「IM-BloomMaker for Accel Platform チュートリアルガイド」-「ルーティングの認可を設定する」を参照してください。

IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_user-LD.xml**）の詳細

IM-LogicDesignerのルーティング「process_instance」に対する認可設定です。
本サンプルでは「IM-BPM管理者」および「IM-BPMユーザ」に対し「参照」権限を付与します。

> **コラム**
>
> IM-LogicDesignerのルーティングの認可について
>
> ルーティングの認可設定手順は「IM-LogicDesigner ユーザ操作ガイド」-「フロールーティングの認可設定」を参照してください。

IM-BloomMakerを利用してアドオンのプロセス一覧画面を作成する（3）

IM-LogicDesignerのロジックフローで「ユーザが定義した業務テーブル」と「IM-BPMのテーブル」を結合して情報を取得し、テーブルコンポーネントへデータバインディングし表示する画面をIM-BloomMakerで作成するサンプルです。

本サンプルの要旨

- IM-BloomMakerを使用して画面を作成します。
- 「ユーザが定義した業務テーブル」と「IM-BPMのテーブル」を結合して取得する処理をIM-LogicDesignerを使用して定義し、IM-BloomMakerを使用して作成した画面より呼び出し、取得した情報を表示します。

図：「プロセス一覧」

- 本サンプルの確認方法
- 本サンプルの構成資材
- IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_addon_process_list_join_table.zip**）の詳細
  - IM-LogicDesignerのロジックフロー定義
  - IM-LogicDesignerのロジックフローのルーティング定義
  - IM-LogicDesignerのロジックフローのユーザ定義
- IM-BPM定義情報（**im_bpm-bloommaker_addon_process_list_join_table.zip**）の詳細
  - IM-BPMのプロジェクト
  - IM-BPMのプロセス定義
  - IM-BPMのデプロイメント
- IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_list_join_table.zip**）の詳細
  - IM-BloomMakerのコンテンツ
  - IM-BloomMakerのルーティング
- IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_join_table-BM.xml**）の詳細
- IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_join_table-LD.xml**）の詳細

注意

本サンプルは、intra-mart Accel Platformに「IM-BloomMaker for Accel Platform」モジュールが組み込まれている環境上で動作する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 : 2020 Summer(Zephirine)
- IM-BPM for Accel Platform : 8.0.12
- IM-BloomMaker for Accel Platform : 8.0.3

注意

本サンプルのSQL定義は PostgreSQLでのみ動作するサンプルです。

本サンプルの確認方法

1. 「*本サンプルの構成資材*」 の資材をそれぞれ組み込み／インポート、または、データベース上に作成してください。
2. 全ての資材のインポートと作成が完了したのち、「サイトマップ」→「BloomMaker」→「ルーティング定義一覧」より「ルーティング定義一覧」画面へ遷移します。
3. IM-BloomMakerのルーティング「【チュートリアル】routing_addon_process_list_join_table」の「URL」に定義されたパスにアクセスすることで、IM-BloomMakerを使用して作成された「プロセス一覧」画面が表示されます。
4. 「プロセス一覧」画面では、IM-LogicDesignerのロジックフローで取得された「ユーザが定義した業務テーブル」と「IM-BPMのテーブル」を結合した情報を閲覧できます。
5. この画面は「IM-BPM管理者」権限をもつユーザが参照できます。

本サンプルの構成資材

- サンプル業務テーブル「im_bpm_bloommaker_sample」作成用DDL（PostgreSQLの例）

```
CREATE TABLE
 im_bpm_bloommaker_sample (
  id varchar(64) NOT NULL PRIMARY KEY,
  proc_inst_id_ varchar(64),
  value1 varchar(255),
  value2 varchar(255)
 )
```

- IM-LogicDesigner定義情報
  im_logicdesigner-data-bloommaker_addon_process_list_join_table.zip
- IM-BPM定義情報
  im_bpm-bloommaker_addon_process_list_join_table.zip
- IM-BloomMaker定義情報
  im_bloommaker-data-bloommaker_addon_process_list_join_table.zip
- IM-Authz（認可） ポリシー - XML形式定義情報
  authz-policy-bloommaker_addon_process_list_join_table-BM.xml
  authz-policy-bloommaker_addon_process_list_join_table-LD.xml

コラム

ユーザ定義テーブルの作成方法について

ユーザ定義テーブルの作成手順は「システム管理者操作ガイド」-「SQLを実行する」を参照してください。
接続先を選択可能な場合、「テナントデータベース」を選択してください。

コラム

IM-LogicDesigner定義情報のインポートについて

インポート手順は「IM-LogicDesigner ユーザ操作ガイド」-「インポートを行う」を参照してください。

コラム

IM-BPM定義情報のインポートについて

インポート手順は「IM-BPM ユーザ操作ガイド」-「インポート」を参照してください。

コラム

IM-BloomMaker定義情報のインポートについて

インポート手順は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」-「定義ファイルをインポートする」を参照してください。

コラム

IM-Authz（認可） ポリシー - XML形式定義情報のインポートについて

インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を
パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
また、ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

コラム

IM-BPMのテーブル定義情報について

IM-BPMのテーブル定義情報は「intra-mart Accel Series ドキュメントライブラリ IM-BPM」より下記のテーブル定義書をダウンロードして参照してください。
「IM-BPM テーブル定義書（日本語）（Excel版）」
「IM-BPM テーブル定義書（英語）（Excel版）」
「IM-BPM テーブル定義書（中国語（中華人民共和国））（Excel版）」

IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_addon_process_list_join_table.zip**）の詳細

テーブル「im_bpm_bloommaker_sample」とIM-BPMのテーブルを結合して取得した情報を返却するIM-LogicDesignerのロジックフローとそのIM-LogicDesignerのロジックフローへのルーティング定義を含む資材です。

IM-LogicDesignerのロジックフロー定義

- 「【チュートリアル】insert_user_table」
  テーブル「im_bpm_bloommaker_sample」へプロセスの情報や、業務の固有データなどをインサートするIM-LogicDesignerのロジックフローです。
  このIM-LogicDesignerのロジックフローは後述のプロセス「*insert_user_table*」を開始することで呼び出せます。
- 「【チュートリアル】select_process_with_user_table」
  テーブル「im_bpm_bloommaker_sample」と「IM-BPMのテーブル」を結合して情報を取得するIM-LogicDesignerのロジックフローです。

IM-LogicDesignerのロジックフローのルーティング定義

- 「process_join」
  IM-LogicDesignerのロジックフロー「【チュートリアル】select_process_with_user_table」へアクセスするためのURLなどの情報を定義するルーティング定義情報です。

IM-LogicDesignerのロジックフローのユーザ定義

- 「【チュートリアル】exec_insert_user_table_sql」
  テーブル「im_bpm_bloommaker_sample」へデータをインサートするためのユーザ定義タスクです。
  IM-LogicDesignerのロジックフロー「【チュートリアル】insert_user_table」内で使用されます。
- 「【チュートリアル】exec_count_with_user_table_sql」
  テーブル「im_bpm_bloommaker_sample」と「IM-BPMのテーブル」を結合した情報の件数を取得するためのユーザ定義タスクです。
  IM-LogicDesignerのロジックフロー「【チュートリアル】select_process_with_user_table」内で使用されます。
- 「【チュートリアル】exec_select_with_user_table_sql」
  テーブル「im_bpm_bloommaker_sample」と「IM-BPMのテーブル」を結合して情報を取得するためのユーザ定義タスクです。
  IM-LogicDesignerのロジックフロー「【チュートリアル】select_process_with_user_table」内で使用されます。

IM-BPM定義情報（**im_bpm-bloommaker_addon_process_list_join_table.zip**）の詳細

テーブル「im_bpm_bloommaker_sample」へ情報をインサートする処理を伴うプロセスのサンプルを含む資材です。

IM-BPMのプロジェクト

- 「【チュートリアル】addon_process_list_join_table」
  本サンプルで使用するプロセスを格納するためのプロジェクトです。

IM-BPMのプロセス定義

- 「insert_user_table」
  テーブル「im_bpm_bloommaker_sample」へプロセスの情報や業務の固有データなどをインサートするIM-LogicDesignerのロジックフローを呼び出すプロセス定義です。
  プロジェクト「【チュートリアル】addon_process_list_join_table」に格納されています。

IM-BPMのデプロイメント

- 「【チュートリアル】addon_process_list_join_table-deployment」
  本サンプルで使用するプロセス定義のデプロイメント定義情報です。

> **コラム**
>
> プロセスの開始について
>
> プロセスの開始手順は「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。

IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_list_join_table.zip**）の詳細

IM-LogicDesignerとIM-BloomMakerを使用して、テーブル「im_bpm_bloommaker_sample」とIM-BPMのテーブルを結合して取得した情報を表示するアドオンのプロセス一覧画面の定義と、その画面へのルーティングを定義する資材です。

IM-BloomMakerのコンテンツ

- 「BPMチュートリアル」-「【チュートリアル】addon_process_list_join_table」
  IM-LogicDesignerのロジックフロー「【チュートリアル】select_process_with_user_table」を呼び出すアクションの定義やボタンクリックなどの画面イベントの定義、レイアウト等の情報を含む、画面の定義情報です。

> **コラム**
>
> IM-BloomMakerのテーブルエレメントの使い方については、下記の記事も併せて参照してください。
>
> IM-BloomMaker「繰り返しエレメントでの変数の使い方」について
>
> IM-BloomMaker「繰り返しエレメントでの変数の使い方」の詳細については「intra-mart Developer Site」-「IM-BloomMaker 繰り返しエレメントでの変数の使い方」を参照してください。
>
> IM-BloomMaker「カスタムスクリプト内で $im.resolve を使った一覧画面の作成」について
>
> IM-BloomMaker「カスタムスクリプト内で $im.resolve を使った一覧画面の作成」の詳細については「intra-mart Developer Site」-「IM-BloomMaker のカスタムスクリプト内で $im.resolve を使った一覧画面の作成」を参照してください。

IM-BloomMakerのルーティング

- 「BPMチュートリアル」-「【チュートリアル】routing_addon_process_list_join_table」
  コンテンツ「【チュートリアル】addon_process_list_join_table」へアクセスするためのURLの指定や、アクセスする際に呼び出される前処理などの情報を定義するルーティング定義情報です。

IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_join_table-BM.xml**）の詳細

IM-BloomMakerのルーティング「【チュートリアル】routing_addon_process_list_join_table」に対する認可設定です。
本サンプルでは「IM-BPM管理者」に対し「参照」権限を付与します。

> **コラム**
>
> IM-BloomMakerのルーティングの認可について
>
> ルーティングの認可設定手順は「IM-BloomMaker for Accel Platform チュートリアルガイド」-「ルーティングの認可を設定する」を参照してください。

IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_list_join_table-LD.xml**）の詳細

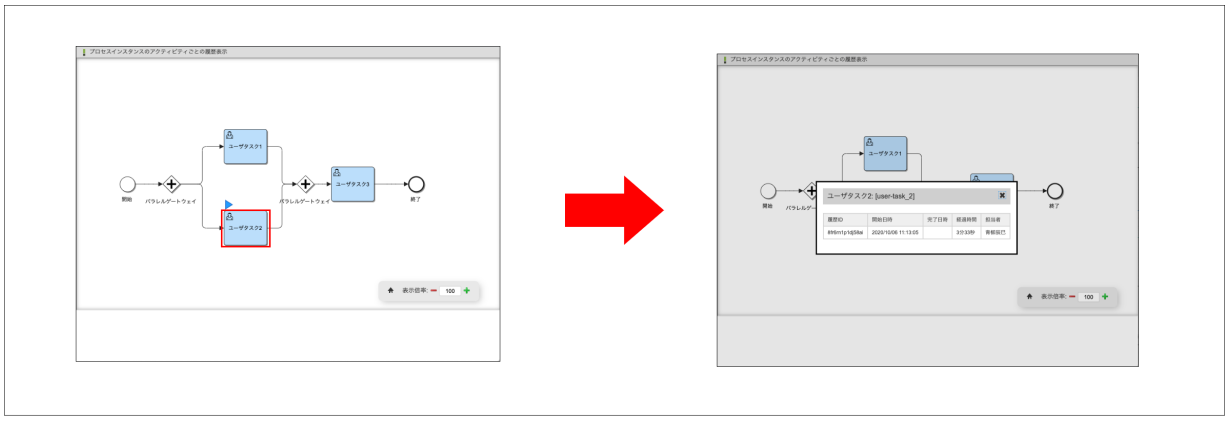
IM-LogicDesignerのルーティング「process_join」に対する認可設定です。
本サンプルでは「IM-BPM管理者」に対し「参照」権限を付与します。

> **コラム**
>
> IM-LogicDesignerのルーティングの認可について
>
> ルーティングの認可設定手順は「IM-LogicDesigner ユーザ操作ガイド」-「フロールーティングの認可設定」を参照してください。

プロセス図を表示するIM-BloomMakerエレメントを利用したアドオン画面の作成（1）

プロセス図を表示するIM-BloomMakerエレメントを利用してアクティビティクリック時に、そのアクティビティに関連する履歴を表示する画面のサンプルです。
IM-BPMのREST APIを利用して、プロセスインスタンスの履歴を取得します。
プロセス図を表示するIM-BloomMakerエレメントを配置し、プロセス図を表示します。
プロセス図内のアクティビティクリック時に、取得したインスタンスの履歴から関連するインスタンスの履歴をダイアログで表示します。

> **注意**
>
> 本サンプルは2020 Winter(Azalea)以降の環境でのみ動作します。



図：「完成イメージ」

- 本サンプルの構成資材
- 本サンプルの使用方法
- IM-BloomMaker定義情報（**im_bloommaker-data-process_diagram_bloommaker_element-show_activity_history.zip**）の詳細
  - コンテンツ定義
  - ルーティング定義

本サンプルの構成資材

- IM-BloomMaker定義情報
  im_bloommaker-data-process_diagram_bloommaker_element-show_activity_history.zip
- IM-Authz（認可） ポリシー - XML形式定義情報
  authz-policy-process_diagram_bloommaker_element-show_activity_history.xml

> **コラム**
>
> IM-BloomMaker定義情報のインポートについて
>
> インポート手順は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」-「定義ファイルをインポートする」を参照してください。

> **コラム**
>
> IM-Authz（認可） ポリシー - XML形式定義情報のインポートについて
>
> インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を
> パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
> また、ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
> または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

本サンプルの使用方法

1. サンプル資材をインポートします。
   IM-Authz（認可） ポリシー - XML形式定義情報をインポートする前にIM-BloomMaker定義情報をインポートしてください。
2. IM-BPMプロセス参照ユーザで `{ベースURL}/bpm/tutorial/process-diagram/show-history/{プロセスインスタンスID}` へアクセスし、サンプル画面を表示します。
3. 表示されるプロセス図内のアクティビティをクリックすることで、そのアクティビティの履歴が表示されます。

IM-BloomMaker定義情報（**im_bloommaker-data-process_diagram_bloommaker_element-show_activity_history.zip**）の詳細

コンテンツ定義

指定されたプロセスインスタンスの履歴情報を元に、アクティビティごとの履歴を表示する画面を作成します。
画面を作成する上でのポイントは以下のとおりです。

- コンテンツ内で使用する変数の準備
- 対象のプロセスインスタンスIDを画面のURLから判断する。
- IM-BPMのREST APIを使用してプロセスインスタンスの履歴情報を取得する。
- 画面描画時にプロセスインスタンスの履歴情報を取得するアクションを実行する。
- プロセス図中のアクティビティクリック時に実行するアクションを作成する。
- プロセス図中のアクティビティクリック時に表示するダイアログを作成する。
- プロセスインスタンスの実行情報のプロセス図を表示する。

1. コンテンツ内で使用する変数の準備
   コンテンツ内で使用する変数を定義します。
   一例として、サンプル資材内で定義されているJSON形式の変数を示します。

```
{
 "accessGetHistoricDataUrl": "",
 "historicDataResponce": {
  "error": false,
  "data": {
   "data": {
    "activities": [
     {
      "id": "",
      "activityId": "",
      "activityName": "",
      "assignee": "",
      "activityType": "",
      "processDefinitionId": "",
      "processDefinitionName": "",
      "processInstanceId": "",
      "executionId": "",
      "startTime": "",
      "endTime": "",
      "durationInMillis": 0,
      "tenantId": "",
      "taskId": "",
      "durationNowMillis": 0,
      "parentActivityId": "",
      "parentActivityName": ""
     }
```

```
      ],
      "variables": [],
      "comments": [],
      "variableUpdates": [],
      "formProperties": [],
      "migrations": [],
      "optionTaskActivities": [],
      "tasks": [
       {
        "assignee": "",
        "assigneeName": "",
        "candidateGroups": {},
        "candidateUsers": {},
        "claimTime": "",
        "executionId": "",
        "id": "",
        "name": "",
        "overdue": false,
        "priority": 50,
        "processDefinitionId": "",
        "processDefinitionName": "",
        "processDefinitionType": "",
        "processInstanceId": "",
        "startTime": "",
        "taskDefinitionKey": "",
        "tenantId": "",
        "variables": []
       }
      ]
     },
     "total": 0,
     "start": 0,
     "size": 0
    }
  },
  "triggerEventDetail": {
    "id": "",
    "name": "",
    "elementType": "",
    "processDefinitionId": "",
    "processDefinitionKey": "",
    "processDefinitionName": "",
    "eventName": ""
  },
  "dblClickData": {
    "id": "",
    "name": "",
    "elementType": "",
    "processDefinitionId": "",
    "processDefinitionKey": "",
    "processDefinitionName": "",
    "eventName": ""
  },
  "targetData": [
    {
      "activityId": "",
      "activityName": "",
      "activityType": "",
      "assignee": "",
      "assigneeName": "",
      "durationNowMillis": 0,
      "executionId": "",
      "id": "",
      "parentActivityId": "",
      "parentActivityName": "",
      "processDefinitionId": "",
      "processDefinitionName": "",
      "processInstanceId": "",
      "startTime": "",
      "tenantId": "",
      "showTime": "",
      "endTime": ""
    }
  ],
  "targetDataLength": 0
}
```


各変数の説明は以下のとおりです。

変数名	説明
accessGetHistoricDataUrl	プロセスインスタンスIDを含めたプロセスインスタンスの履歴情報を取得するREST APIのURL
historicDataResponce	プロセスインスタンスの履歴情報を取得するREST APIのレスポンス
triggerEventDetail	プロセス図中で対象のアクティビティに対しクリックなどのイベントが発行された際に、対象の情報を格納する変数
dblClickData	プロセス図中でダブルクリックされたアクティビティの情報
targetData	対象となるアクティビティIDに関連する履歴の情報
targetDataLength	対象となるアクティビティIDに関連する履歴の情報の件数

2. 対象のプロセスインスタンスIDを画面のURLから受け取る。
   画面のURLに入力されている値をIM-BloomMakerのコンテンツ定義で使用する場合はルーティング定義のURLの設定と入力値の設定が必要です。
   このプロセスインスタンスIDは後述の、プロセスインスタンスの履歴情報を取得するREST APIのURLや、プロセス図を表示するエレメントで使用します。

> **コラム**
>
> 入力値の設定に関する詳細は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」 - 「入力の設定方法」を参照してください。

3. IM-BPMのREST APIを使用してプロセスインスタンスの履歴情報を取得する。
   IM-BPMのプロセスインスタンス履歴の履歴データをプロセスインスタンスIDより取得するREST APIを利用し、履歴データを取得します。
   アクションエディタにて、アクションアイテム `カスタムスクリプトを実行する` を使用して、リクエストを送信するURLを組み立てます。
   `URL「」にリクエストを送信する` を使用して、組み立てたURLへリクエストを送信し、データを取得します。

> **コラム**
>
> IM-BPMのプロセスインスタンスの履歴情報を取得するREST APIの詳細については、「APIドキュメント」 - 「プロセスインスタンス履歴の履歴データ取得」を参照してください。

> **コラム**
>
> アクションの設定に関する詳細は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」 - 「アクションを設定する」を参照してください。

4. 画面描画時にプロセスインスタンスの履歴情報を取得するアクションを実行する。
   「コンテナ」を選択した状態で、プロパティの「ページ読み込み時」の項目で履歴情報を取得するアクションを指定します。

5. プロセス図中のアクティビティクリック時に表示するダイアログを作成する。
   プロセス図中のアクティビティクリック時に該当のアクティビティの履歴を表示するダイアログを作成します。

   「デザイナ」画面上部、コンテナヘッダの「 」アイコンをクリックし新たなページを作成します。

   「繰り返し（imui）」に分類されるエレメントの一覧から、「縦方向のテーブル（繰り返し）」を設置します。
   テーブルエレメントのプロパティ `list` に対象となるアクティビティIDに関連する履歴の情報の変数を指定します。
   テーブルエレメントの各列に表示させたい情報を設定します。

> **コラム**
>
> IM-BloomMakerで新たなページを作成する方法は、「IM-BloomMaker for Accel Platform チュートリアルガイド」 - 「ページを設定する」を参照してください。

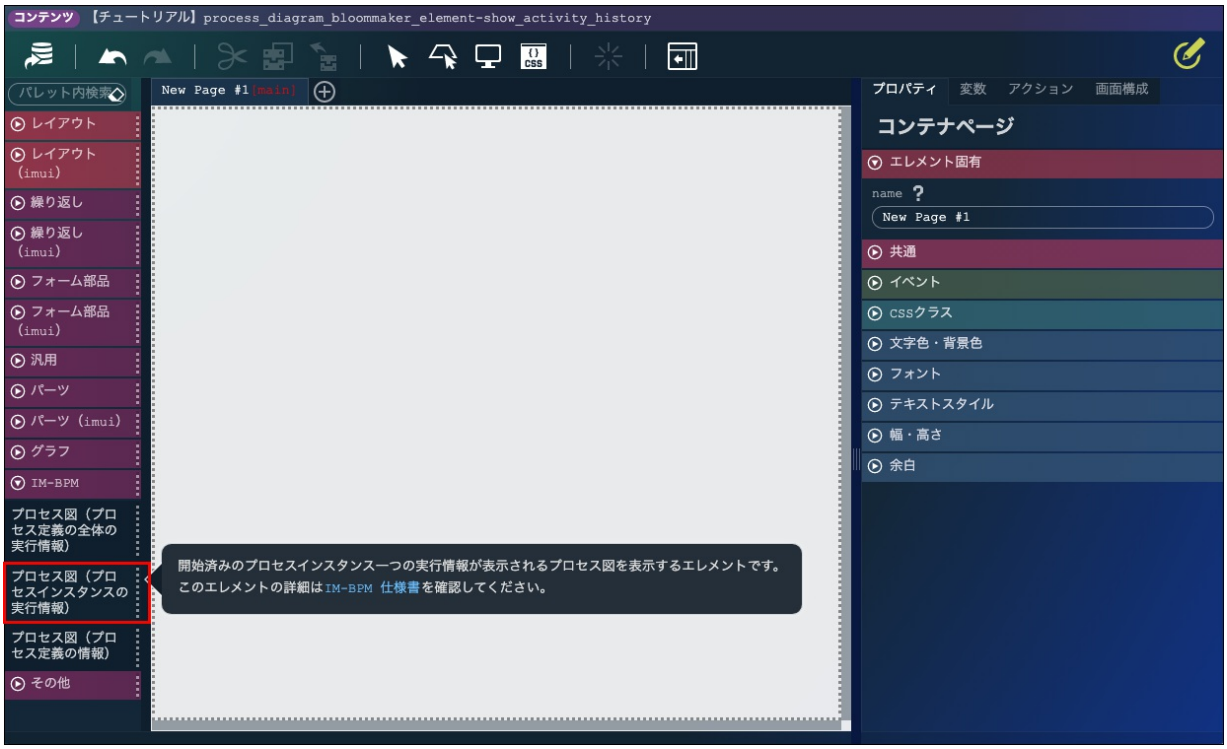
> **コラム**
>
> IM-BloomMakerの繰り返しエレメントの使用方法は、「intra-mart Developer Site」 - 「IM-BloomMaker 繰り返しエレメントでの変数の使い方」を参照してください。

6. プロセス図中のアクティビティクリック時に実行するアクションを作成する。
   プロセス図中のアクティビティクリック時に実行するアクションを作成します。
   取得してきたプロセスインスタンスの履歴情報の中から該当のアクティビティに関連する情報のみを取得します。
   アクションエディタにて、アクションアイテム `カスタムスクリプトを実行する` を使用して、履歴情報の中から対象のアクティビティIDに合致するもののみを取り出します。
   取得してきた履歴情報のレスポンスが格納されている変数名を、 `historicDataResponce` 対象のアクティビティIDを `activityId` とすると、以下の方法で対象のアクティビティIDに合致する情報のみを取得できます。

```
const activities = $im.resolve("$variable.historicDataResponce.data.data.activities");
const targetData = activities.filter((v) => {
   return v.activityId === activityId;
});
```

さらに、履歴情報の中に含まれる時刻の形式や、処理時間を見やすい文字列へ整形したい場合は、このカスタムスクリプト内で処理します。
整形が完了したらそのデータを、対象となるアクティビティIDに関連する履歴の情報の変数に格納します。
以上で、データを整形するカスタムスクリプトの作成は完了です。
アクションアイテム `ページ「」をダイアログで開く` を使用して、作成したダイアログ用のページを表示するよう設定します。

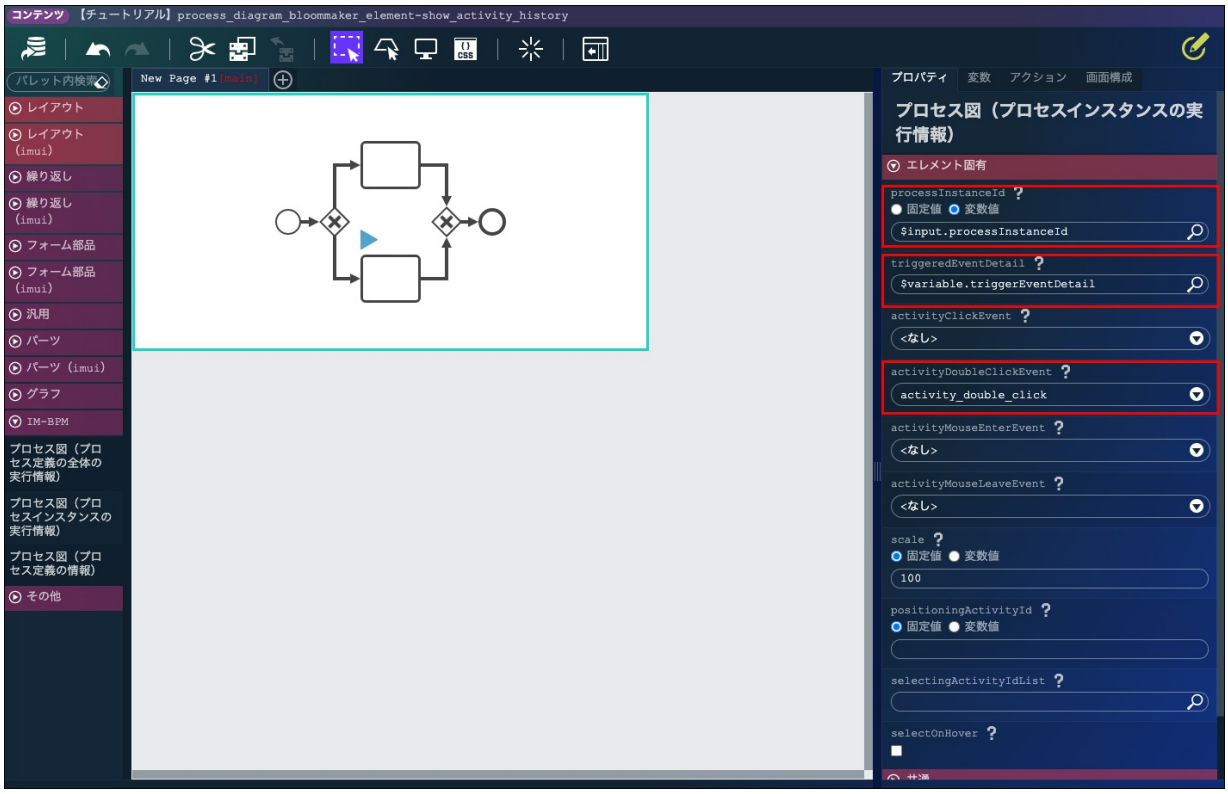
7. プロセスインスタンスの実行情報のプロセス図を表示する。
   - 「IM-BPM」に分類されるエレメントの一覧から、「プロセス図（プロセスインスタンスの実行情報）」を設置します。



図：エレメントパレット - 「IM-BPM」

   - プロセス図のエレメントのプロパティにて以下の値を設定します。

プロパティ名	説明
processInstanceId	入力値から取得したプロセスインスタンスID
triggerEventDetail	アクティビティに対してイベントが発行された際に、そのアクティビティの情報を格納する変数
activityDoubleClickEvent	アクティビティをダブルクリックした際に実行するアクション



図：エレメントパレット - プロパティ

> **コラム**
>
> プロセス図を表示するIM-BloomMakerエレメントの詳細は、「IM-BPM 仕様書」 - 「プロセス図を表示するIM-BloomMakerエレメント」を参照してください。

ルーティング定義

1. コンテンツ定義で作成した画面にアクセスをするために、ルーティング定義を作成します。
   ルーティング定義を作成する際のURLは、コンテンツ定義の入力値に渡すプロセス定義を含めた動的URLとする必要があります。

   > **コラム**
   >
   > ルーティングの登録に関する詳細は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」 - 「ルーティングを新規登録する」を参照してください。

2. ルーティング定義に認可を設定する。
   作成したルーティングにアクセスするためには、適切な認可の設定が必要です。
   本サンプルでは、「IM-BPMプロセス参照ユーザ」に対して「参照」権限を付与しています。

   > **コラム**
   >
   > ルーティングの認可設定の詳細は「IM-BloomMaker for Accel Platform チュートリアルガイド」-「ルーティングの認可を設定する」を参照してください。

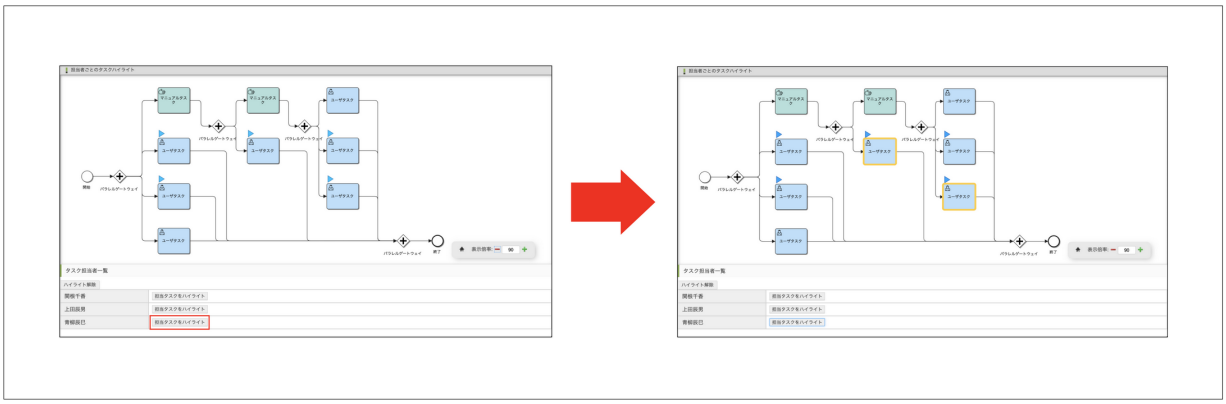
プロセス図を表示するIM-BloomMakerエレメントを利用したアドオン画面の作成（2）

プロセスインスタンスの実行中のタスクの担当者を一覧表示し、ボタンクリックでその担当者が担当するタスクをプロセス図上でハイライトする画面のサンプルです。

本サンプルの要旨

- IM-BPMのREST APIを利用して取得されるプロセスインスタンスの履歴から、現在実行中となっているタスクの担当者の一覧を取得し一覧で表示します。
- プロセス図を表示するIM-BloomMakerエレメントを配置し、プロセス図を表示します。
- 担当者一覧の表中のボタンをクリックすることで、そのユーザが担当しているタスクをプロセス図上でハイライト表示します。

> **注意**
>
> 本サンプルは2020 Winter(Azalea)以降の環境でのみ動作します。



図：「完成イメージ」

- 本サンプルの構成資材
- 本サンプルの使用方法
- IM-BloomMaker定義情報（**im_bloommaker-data-process_diagram_bloommaker_element-show_task_assignee.zip**）の詳細
  - コンテンツ定義
  - ルーティング定義

本サンプルの構成資材

- IM-BloomMaker定義情報
  im_bloommaker-data-process_diagram_bloommaker_element-show_task_assignee.zip
- IM-Authz（認可） ポリシー - XML形式定義情報
  authz-policy-process_diagram_bloommaker_element-show_task_assignee.xml

> **コラム**
>
> IM-BloomMaker定義情報のインポートについて
>
> インポート手順は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」-「定義ファイルをインポートする」を参照してください。

> **コラム**
>
> IM-Authz（認可） ポリシー - XML形式定義情報のインポートについて
>
> インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を
> パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
> また、ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
> または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

本サンプルの使用方法

1. サンプル資材をインポートします。
   IM-Authz（認可） ポリシー - XML形式定義情報をインポートする前にIM-BloomMaker定義情報をインポートしてください。
2. IM-BPMプロセス参照ユーザで `{ベースURL}/bpm/tutorial/process-diagram/show-task-assignee/{プロセスインスタンスID}` へアクセスし、サンプル画面を表示します。
3. 表示される「タスク担当者一覧」内の「担当タスクをハイライト」をクリックすることで、そのユーザが担当しているタスクがプロセス図上でハイライト表示されます。

IM-BloomMaker定義情報（**im_bloommaker-data-process_diagram_bloommaker_element-show_task_assignee.zip**）の詳細

コンテンツ定義

指定されたプロセスインスタンスの履歴情報を元に、プロセス図と現在実行中のタスクの担当者一覧を表示する画面を作成します。
画面を作成する上でのポイントは以下のとおりです。

- コンテンツ内で使用する変数の準備をする。
- 対象のプロセスインスタンスIDを画面のURLから判断する。

- IM-BPMのREST APIを使用してプロセスインスタンスの履歴情報を取得する。
- 取得した履歴情報の中から、現在実行中のタスクの担当者とその担当者が持つアクティビティIDの一覧を整理する。
- 画面描画時にプロセスインスタンスの履歴情報を取得するアクションを実行する。
- プロセスインスタンスの実行情報のプロセス図を表示する。
- プロセス図中のアクティビティをハイライトさせるアクションを作成する。
- 現在実行中のタスクの担当者とハイライトボタンの一覧を表で表示する。
- ハイライトを解除するボタンを設置する。

1. コンテンツ内で使用する変数の準備をする。
   コンテンツ内で使用する変数を定義します。
   一例として、サンプル資材内で定義されているJSON形式の変数を示します。


```
{
 "accessGetHistoricDataUrl": "",
 "historicDataResponce": {
  "error": false,
  "data": {
   "data": {
    "activities": [],
    "variables": [],
    "comments": [],
    "variableUpdates": [],
    "formProperties": [],
    "migrations": [],
    "optionTaskActivities": [],
    "tasks": [
     {
      "assignee": "",
      "assigneeName": "",
      "claimTime": "",
      "deleteReason": "",
      "durationInMillis": 0,
      "endTime": "",
      "executionId": "",
      "id": "",
      "name": "",
      "overdue": false,
      "priority": 0,
      "processDefinitionId": "",
      "processDefinitionName": "",
      "processDefinitionType": "",
      "processInstanceId": "",
      "startTime": "",
      "taskDefinitionKey": "",
      "tenantId": "",
      "variables": [],
      "workTimeInMillis": 0
     }
    ]
   },
   "total": 0,
   "start": 0,
   "size": 0
  }
 },
 "assigneeDict": [
  {
   "assignee": "",
   "assigneeName": "",
   "activityId": [
    ""
   ]
  }
 ],
 "highlightActivityIds": []
}
```


各変数の説明は以下のとおりです。

変数名	説明
accessGetHistoricDataUrl	プロセスインスタンスIDを含めたプロセスインスタンスの履歴情報を取得するREST APIのURL
historicDataResponce	プロセスインスタンスの履歴情報を取得するREST APIのレスポンス

変数名	説明
assigneeDict	現在実行中のアクティビティIDとアクティビティ名を担当者ごとに整理したデータ
highlightActivityIds	ハイライトさせたいアクティビティIDのリスト

2. 対象のプロセスインスタンスIDを画面のURLから判断する。
   画面のURLに入力されている値をIM-BloomMakerのコンテンツ定義で使用する場合はルーティング定義のURLの設定と入力値の設定が必要です。
   このプロセスインスタンスIDは後述の、プロセスインスタンスの履歴情報を取得するREST APIのURLや、プロセス図を表示するエレメントで使用します。

> **コラム**
>
> 入力値の設定に関する詳細は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」 - 「入力の設定方法」を参照してください。

3. IM-BPMのREST APIを使用してプロセスインスタンスの履歴情報を取得する。
   IM-BPMのプロセスインスタンス履歴の履歴情報をプロセスインスタンスIDより取得するREST APIを利用し、履歴情報を取得します。
   アクションエディタにて、アクションアイテム `カスタムスクリプトを実行する` を使用して、リクエストを送信するURLを組み立てます。
   `URL「」にリクエストを送信する` を使用して、組み立てたURLへリクエストを送信し、データを取得します。

> **コラム**
>
> IM-BPMのプロセスインスタンスの履歴情報を取得するREST APIの詳細については、「APIドキュメント」 - 「プロセスインスタンス履歴の履歴データ取得」を参照してください。

> **コラム**
>
> アクションの設定に関する詳細は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」 - 「アクションを設定する」を参照してください。

4. 取得した履歴情報の中から、現在実行中のタスクの担当者とその担当者が持つアクティビティIDの一覧を整理する。
   履歴情報を取得するアクションの中で `URL「」にリクエストを送信する` の後に、 `カスタムスクリプトを実行する` を設置して取得してきた情報を整理するカスタムスクリプトを実行します。
   最終的に以下の形式となるように整形します。

```
[
  {
    "assignee": "%担当者のユーザコード%",
    "assigneeName": "%担当者名%",
    "activityId": [
      "%アクティビティID%"
    ]
  }
]
```

   REST APIから取得してきた履歴情報は `$variable.historicDataResponce` に格納されているとすると、今回取得したいタスクの情報は以下のように格納されています。

情報	格納されている変数
アクティビティID	historicDataResponce.data.data.tasks.taskDefinitionKey
担当者のユーザコード	historicDataResponce.data.data.tasks.assignee
担当者名	historicDataResponce.data.data.tasks.assigneeName

   一例として、サンプル資材内で定義されているカスタムスクリプトを示します。

```
const tasks = $im.resolve("$variable.historicDataResponce.data.data.tasks")

const assigneeDict = []

tasks.forEach((value) => {
 // 担当者なしや完了しているタスクは対象外とする。
 if (typeof value.assignee === 'undefined' || typeof value.endTime !== 'undefined') {
  return;
 }

 const assigneeList = assigneeDict.map(v => v.assignee)
 const i = assigneeList.indexOf(value.assignee);
 if (i >= 0) {
  assigneeDict[i].activityId.push(value.taskDefinitionKey);
 } else {
  const newRecord = {
   assignee: value.assignee,
   assigneeName: value.assigneeName,
   activityId: [value.taskDefinitionKey]
  }
  assigneeDict.push(newRecord);
 }
})

$variable.assigneeDict = assigneeDict;
```

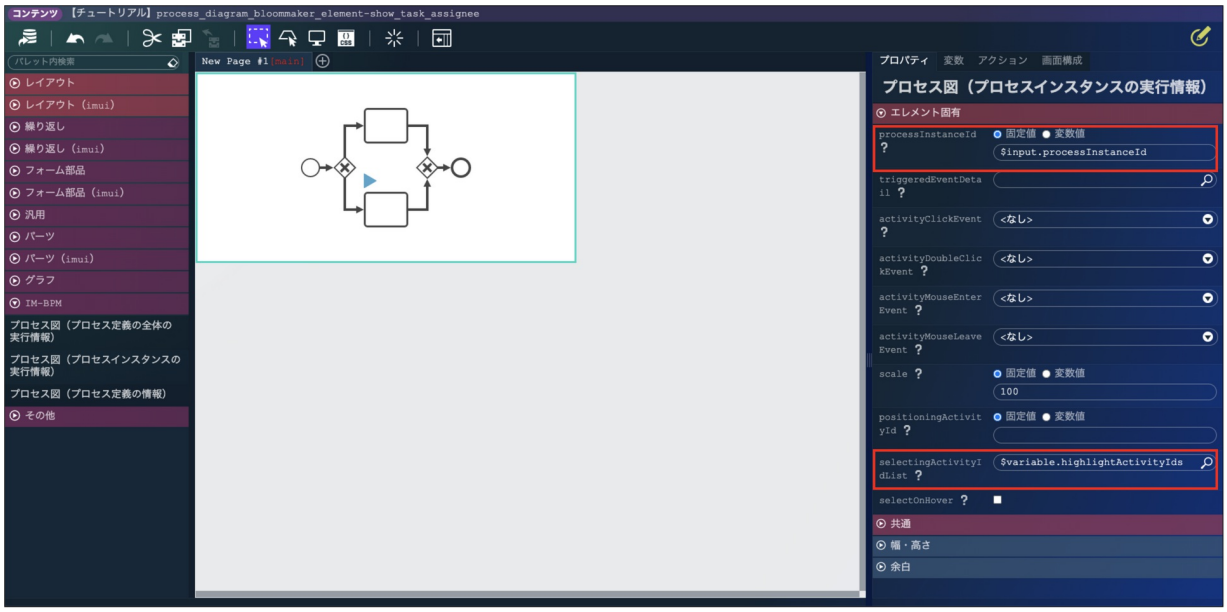
5. 画面描画時にプロセスインスタンスの履歴情報を取得するアクションを実行する。
   「コンテナ」を選択した状態で、プロパティの「ページ読み込み時」の項目で履歴情報を取得するアクションを指定します。

6. プロセスインスタンスの実行情報のプロセス図を表示する。
   - 「IM-BPM」に分類されるエレメントの一覧から、「プロセス図（プロセスインスタンスの実行情報）」を設置します。

図：エレメントパレット - 「IM-BPM」

- プロセス図のエレメントのプロパティにて以下の値を設定します。

プロパティ名	説明
processInstanceId	入力値から取得したプロセスインスタンスID
selectingActivityIdList	ハイライトさせたいアクティビティIDのリストを格納する変数



図：エレメントパレット - プロパティ

> **コラム**
>
> プロセス図を表示するIM-BloomMakerエレメントの詳細は、「IM-BPM 仕様書」 - 「プロセス図を表示するIM-BloomMakerエレメント」を参照してください。

7. プロセス図中のアクティビティをハイライト・アンハイライトさせるアクションを作成する。
   後述の手順で、表示タスク担当者の一覧を「横方向のテーブル（繰り返し）」を使用して作成します。
   一覧中にあるボタンがクリックされた際に実行する、アクティビティをハイライトさせるアクションを実装します。
   具体的には、ハイライトさせたいアクティビティIDのリスト格納する変数に対象のアクティビティIDを設定します。
   アクションエディタにて、アクションアイテム `変数「」に「」を代入する` を使用して、目的のIDの配列を代入します。
   繰り返しエレメントを使用しているため、対象の要素へは `$index` を指定することでアクセスできます。

   > **コラム**
   >
   > IM-BloomMakerの繰り返しエレメントの使用方法は、「intra-mart Developer Site」 - 「IM-BloomMaker 繰り返しエレメントでの変数の使い方」を参照してください。

   アンハイライトさせるアクションについても、同様にアクションアイテム `変数「」に「」を代入する` を使用して、空の配列を代入することで実現可能です。

8. 現在実行中のタスクの担当者とハイライトボタンの一覧を表で表示する。
   表示タスク担当者の一覧を `横方向のテーブル（繰り返し）` を使用して作成します。
   「繰り返し（imui）」に分類されるエレメントの一覧から、「横方向のテーブル（繰り返し）」を設置します。
   「横方向のテーブル（繰り返し）」エレメントのプロパティで以下を設定します。

プロパティ名	説明
`list`	現在実行中のアクティビティIDとアクティビティ名を担当者ごとに整理したデータ

   「横方向のテーブル（繰り返し）」内の「テーブルヘッダ」エレメントのプロパティで以下を設定します。
   なお、要素にアクセスする際は `$index` を使用します。

プロパティ名	説明
`textContent`	担当者名

   「横方向のテーブル（繰り返し）」内の「テーブルデータ」エレメント配下に、「フォーム部品（imui）」に分類されるエレメントの一覧から、「ボタン」を設置します。
   設置した「ボタン」エレメントのプロパティ項目「イベント」で以下を設定します。

プロパティ名	説明
`クリック時`	プロセス図中のアクティビティをハイライトさせるアクション

9. ハイライトを解除するボタンを設置する。
   「フォーム部品（imui）」に分類されるエレメントの一覧から、「ボタン」を設置します。
   設置した「ボタン」エレメントのプロパティ項目「イベント」で以下を設定します。

プロパティ名	説明
クリック時	プロセス図中のアクティビティをアンハイライトさせるアクション

ルーティング定義

1. コンテンツ定義で作成した画面にアクセスをするために、ルーティング定義を作成します。
   ルーティング定義を作成する際のURLは、コンテンツ定義の入力値に渡すプロセス定義を含めた動的URLとする必要があります。

> **コラム**
>
> ルーティングの登録に関する詳細は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」 - 「ルーティングを新規登録する」を参照してください。

2. ルーティング定義に認可を設定する。
   作成したルーティングにアクセスするためには、適切な認可の設定が必要です。
   本サンプルでは、「IM-BPMプロセス参照ユーザ」に対して「参照」権限を付与しています。

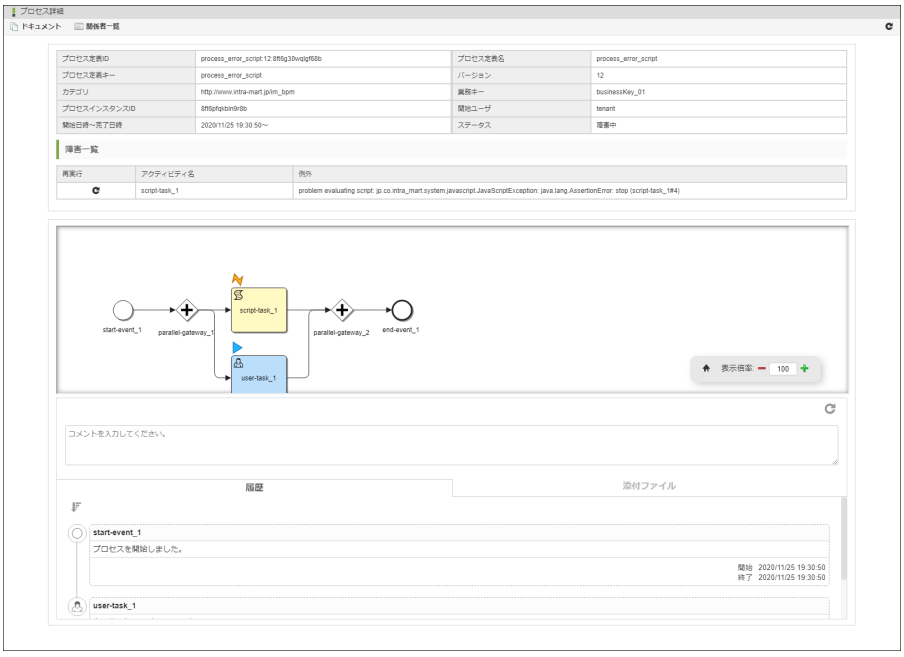
> **コラム**
>
> ルーティングの認可設定の詳細は「IM-BloomMaker for Accel Platform チュートリアルガイド」-「ルーティングの認可を設定する」を参照してください。

IM-BloomMakerを利用してアドオンのプロセス詳細画面を作成する

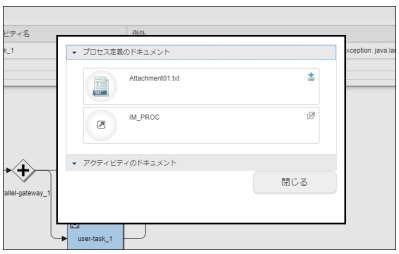
IM-LogicDesignerのロジックフローを使用してアドオンのプロセス詳細画面を作成するサンプルです。

本サンプルの要旨

- プロセスの詳細情報と、プロセスに紐づくドキュメント情報、関係者/関係グループ一覧、障害一覧情報、プロセス図、履歴情報を表示する画面をIM-BloomMakerを使用して作成します。



図：「プロセス詳細」



図：「ドキュメント」

図：「関係者/関係グループ一覧」

- 本サンプルの確認方法
- 本サンプルの構成資材
- IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_addon_process_detail.zip**）の詳細
  - IM-LogicDesignerのロジックフロー定義
- IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_detail.zip**）の詳細
  - IM-BloomMakerのコンテンツ
  - IM-BloomMakerのルーティング
- IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_detail-BM.xml**）の詳細

注意

本サンプルは、intra-mart Accel Platformに「IM-BloomMaker for Accel Platform」モジュールが組み込まれている環境上で動作する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 : 2020 Winter(Azalea)
- IM-BPM for Accel Platform : 8.0.13
- IM-BloomMaker for Accel Platform : 8.0.4

本サンプルの確認方法

1. 「*本サンプルの構成資材*」 の資材をそれぞれインポートしてください。
2. 全ての資材のインポートと作成が完了したのち、「サイトマップ」→「BloomMaker」→「ルーティング定義一覧」より「ルーティング定義一覧」画面へ遷移します。
3. IM-BloomMakerのルーティング「【チュートリアル】routing_addon_process_detail」の「URL」に定義されたパスにアクセスすることで、IM-BloomMakerを使用して作成された「プロセス詳細」画面が表示されます。（URL: `{ベースURL}/bpm/tutorial/process_detail/{プロセスインスタンスID}`）
4. この画面は「IM-BPM管理者」権限または、「IM-BPMプロセス参照ユーザ」権限をもつユーザが参照できます。

本サンプルの構成資材

- IM-LogicDesigner定義情報
  im_logicdesigner-data-bloommaker_addon_process_detail.zip
- IM-BloomMaker定義情報
  im_bloommaker-data-bloommaker_addon_process_detail.zip
- IM-Authz（認可） ポリシー - XML形式定義情報
  authz-policy-bloommaker_addon_process_detail-BM.xml

コラム

IM-LogicDesigner定義情報のインポートについて

インポート手順は「IM-LogicDesigner ユーザ操作ガイド」-「インポートを行う」を参照してください。

コラム

IM-BloomMaker定義情報のインポートについて

インポート手順は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」-「定義ファイルをインポートする」を参照してください。

> **コラム**
>
> IM-Authz（認可） ポリシー - XML形式定義情報のインポートについて
>
> インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を
> パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
> また、ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
> または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

IM-LogicDesigner 定義情報（**im_logicdesigner-data-bloommaker_addon_process_detail.zip**）の詳細

本画面の前処理として、画面で利用するIM-BPM REST APIの権限を確認するIM-LogicDesignerのロジックフローを含むIM-LogicDesignerの資材です。

IM-LogicDesignerのロジックフロー定義

- 「【チュートリアル】check_authority_re_execute_error_jobs」
  ユーザのコンテキスト情報から以下のIM-BPM REST APIの認可の有無の確認を行うIM-LogicDesignerのロジックフローです。
  - IM-BPM REST API「プロセス定義取得」の実行権限
  - IM-BPM REST API「障害中ジョブ取得」の実行権限
  - IM-BPM REST API「ジョブ実行」の実行権限

IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_detail.zip**）の詳細

プロセスの詳細情報と、プロセスに紐づくドキュメント情報、関係者/関係グループ一覧、障害一覧情報、プロセス図、履歴情報を表示するアドオンのプロセス詳細画面の定義と、その画面へのルーティングを定義する資材です。

IM-BloomMakerのコンテンツ

- 「BPMチュートリアル」-「【チュートリアル】addon_process_detail」
  プロセスの詳細と、プロセスに紐づくドキュメント情報、関係者/関係グループ一覧、障害一覧情報、プロセス図、履歴情報を表示する画面の定義情報です。

IM-BloomMakerのルーティング

- 「BPMチュートリアル」-「【チュートリアル】routing_addon_process_detail」
  コンテンツ「【チュートリアル】addon_process_detail」へアクセスするための下記のURLの指定や、アクセスする際に呼び出される前処理などの情報を定義するルーティング定義情報です。
  URL: {ベースURL}/bpm/tutorial/process_detail/{プロセスインスタンスID}

IM-Authz（認可） ポリシー - XML形式定義情報（**authz-policy-bloommaker_addon_process_detail-BM.xml**）の詳細

IM-BloomMakerのルーティング「【チュートリアル】routing_addon_process_detail」に対する認可設定です。
本サンプルでは「IM-BPM管理者」および、「IM-BPMプロセス参照ユーザ」に対し「参照」権限を付与します。

> **コラム**
>
> IM-BloomMakerのルーティングの認可について
>
> ルーティングの認可設定手順は「IM-BloomMaker for Accel Platform チュートリアルガイド」-「ルーティングの認可を設定する」を参照してください。

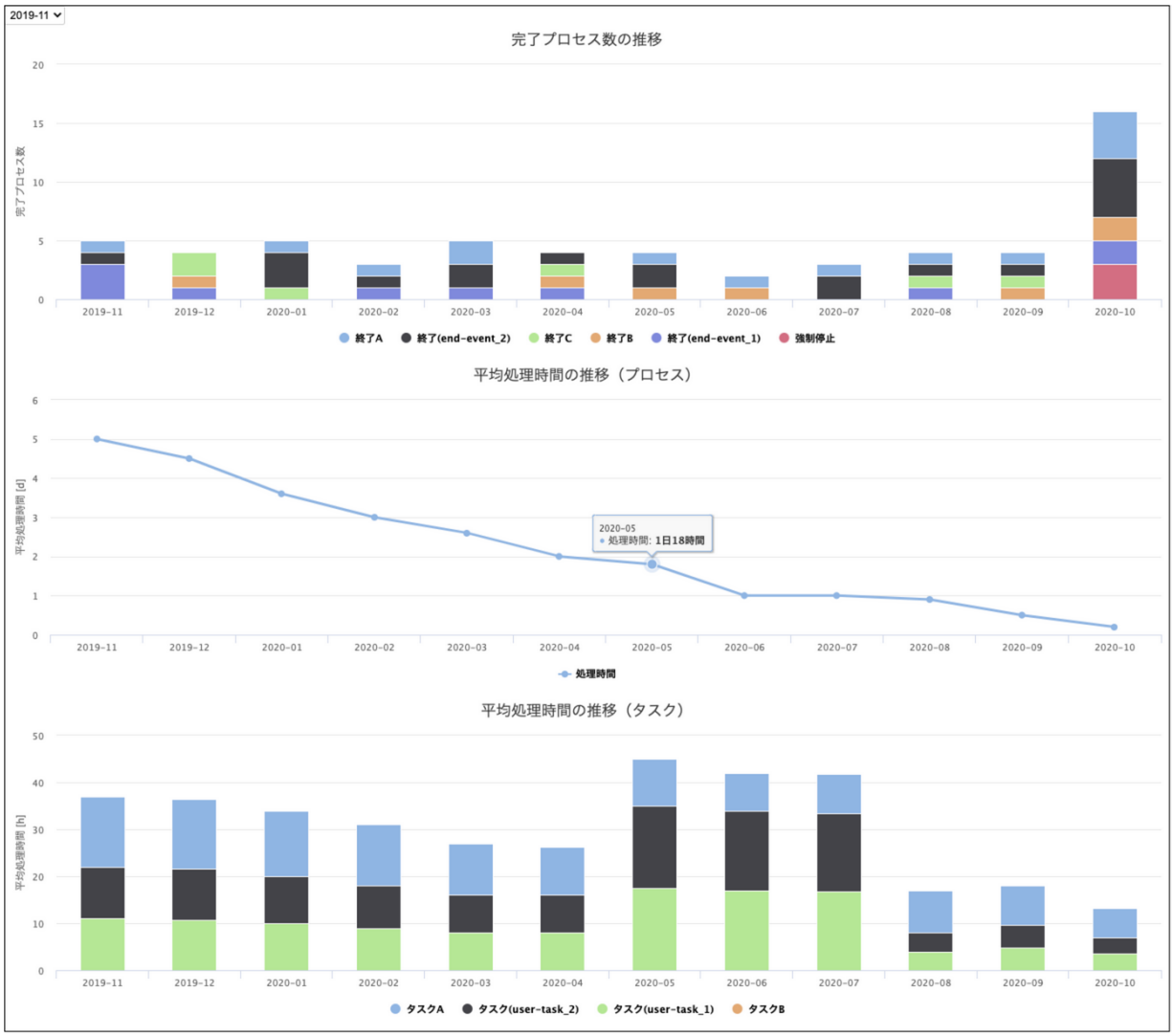
プロセスの分析

IM-BloomMakerを利用してプロセス定義のグラフを作成する

このチュートリアルでは、IM-LogicDesignerとIM-BloomMakerを利用して、プロセス定義のグラフを表示するページを作成します。
IM-LogicDesignerを利用してIM-BPMのテーブルから、プロセス定義の実行状態のデータを取得します。
取得したデータをIM-BloomMakerのグラフエレメントで表示することにより、プロセス定義の情報をグラフで表します。

> **注意**
>
> 本サンプルは2020 Winter(Azalea)以降の環境でのみ動作します。

> **注意**
>
> 本サンプルのSQL定義は PostgreSQLでのみ動作するサンプルです。



図：「完成イメージ」

- 本サンプルの構成資材
- 本サンプルの使用方法
- IM-LogicDesigner定義情報（**im_logicdesigner-data-bloommaker_addon_process_definition_graph.zip**）の詳細
  - ロジックフロー（**【チュートリアル】bloommaker_addon_process_definition_graph-get_graph_data_process_instances**）
  - ロジックフロー（**【チュートリアル】bloommaker_addon_process_definition_graph-get_graph_data_process_instances_tasks**）
  - ルーティング定義
- IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_definition_graph.zip**）の詳細
  - コンテンツ定義
  - ルーティング定義

本サンプルの構成資材

- IM-BloomMaker定義情報
  im_bloommaker-data-bloommaker_addon_process_definition_graph.zip
- IM-LogicDesigner定義情報
  im_logicdesigner-data-bloommaker_addon_process_definition_graph.zip
- IM-Authz（認可） ポリシー - XML形式定義情報
  authz-policy-bloommaker_addon_process_definition_graph-BM.xml
  authz-policy-bloommaker_addon_process_definition_graph-LD.xml

コラム

IM-BloomMaker定義情報のインポートについて

インポート手順は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」-「定義ファイルをインポートする」を参照してください。

コラム

IM-LogicDesigner定義情報のインポートについて

インポート手順は「IM-LogicDesigner ユーザ操作ガイド」-「インポート/エクスポート」を参照してください。

コラム

IM-Authz（認可） ポリシー - XML形式定義情報のインポートについて

インポート手順は「IM-Authz（認可）インポート・エクスポート仕様書」-「ポリシー - XML形式」を
パブリックストレージへのファイルの配置は「システム管理者操作ガイド」-「ファイル操作」を参照してください。
また、ジョブネット「認可(ポリシー)インポート」の実行時に実行パラメータfile を追加し、値に上記の認可ファイルのパブリックストレージ上のパスを指定してください。
または、パブリックストレージへの配置の際にファイル名をauthz-policy.xmlへリネームしてパブリックストレージのルート直下に配置し、ジョブを実行してください。

コラム

IM-BPMのテーブル定義情報について

IM-BPMのテーブル定義情報は「intra-mart Accel Series ドキュメントライブラリ IM-BPM」より下記のテーブル定義書をダウンロードして参照してください。
「IM-BPM テーブル定義書（日本語）（Excel版）」
「IM-BPM テーブル定義書（英語）（Excel版）」
「IM-BPM テーブル定義書（中国語（中華人民共和国））（Excel版）」

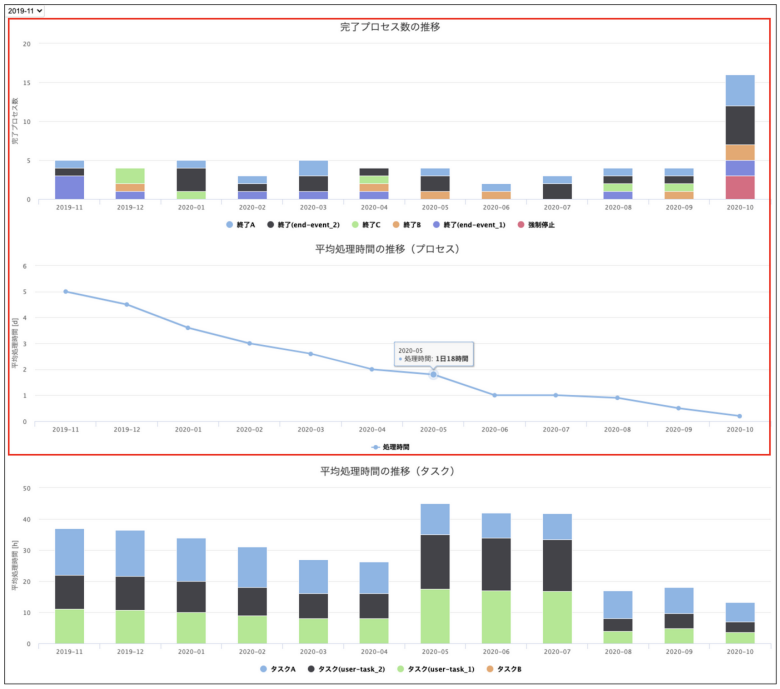
本サンプルの使用方法

1. サンプル資材をインポートします。
   IM-Authz（認可） ポリシー - XML形式定義情報をインポートする前にIM-BloomMaker定義情報とIM-LogicDesigner定義情報をインポートしてください。
2. {ベースURL}/bpm/tutorial/process/definition/graph/{プロセス定義ID} へアクセスし、サンプル画面を表示します。
3. 指定したプロセス定義の直近1年の完了プロセス数の推移や、平均処理時間の推移がグラフで表示されます。

IM-LogicDesigner定義情報（**im_logicdesigner-data-bloommaker_addon_process_definition_graph.zip**）の詳細

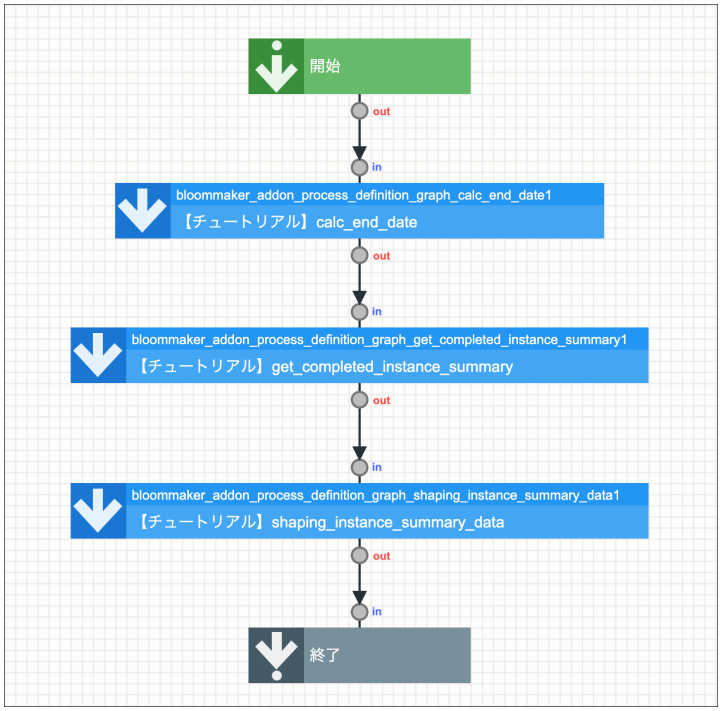
ロジックフロー（**【チュートリアル】bloommaker_addon_process_definition_graph-get_graph_data_process_instances**）

画面に表示されるグラフのうち、下図の赤枠の部分で表示されているグラフの表示に必要なデータを、データベースからIM-LogicDesignerを使用して取得する方法を説明します。



図：完成イメージ

プロセス定義の完了したインスタンスの数と平均処理時間を1ヶ月ごとに集計し、IM-BloomMakerのグラフエレメントへ渡すことのできる形式へ整形します。



図：「ロジックフロー（【チュートリアル】bloommaker_addon_process_definition_graph-get_graph_data_process_instances）」

このロジックフローの入出力設定は以下のように設定されています。
入力として検索開始日時とプロセス定義IDを受け取ります。
出力はIM-LogicDesignerの完了プロセス数と平均処理時間のデータを、IM-BloomMakerのグラフエレメントのプロパティである `series` と `xAxisCategories` へ渡せる状態に整形したデータです。

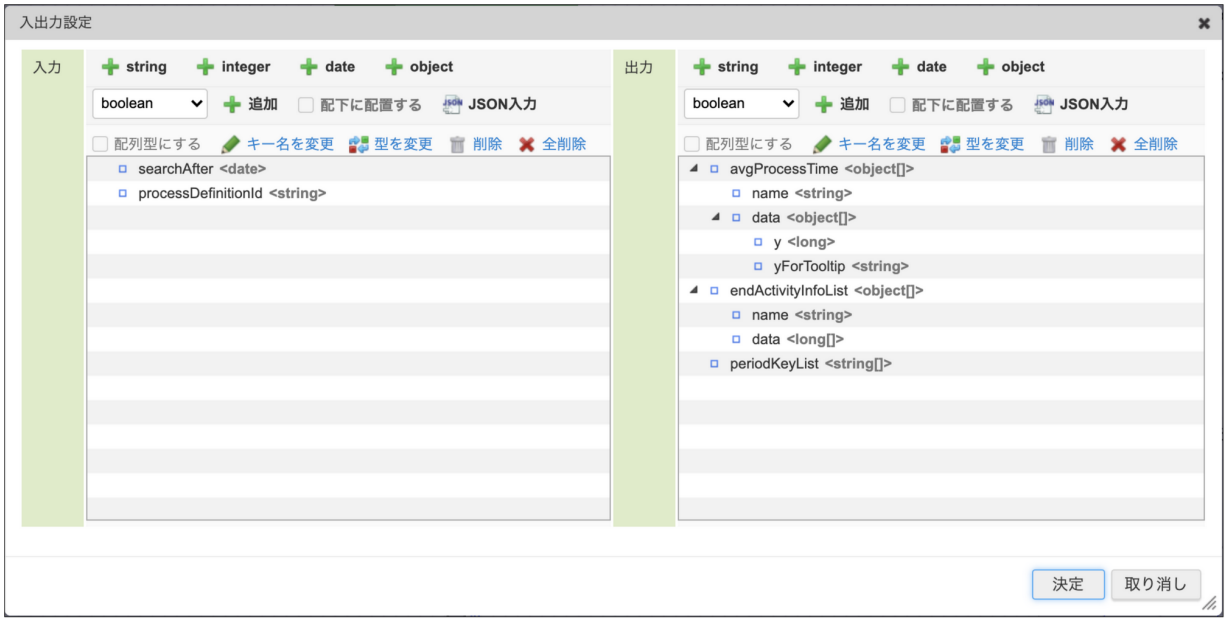
- 入力

キー名	型	説明
`searchAfter`	<date>	検索開始日時
`processDefinitionId`	<string>	プロセス定義ID

- 出力

キー名	型	説明

キー名	型	説明
avgProcessTime	<object[]>	IM-BloomMakerのグラフエレメント「折れ線グラフ」のプロパティ series に渡すためのデータ
avgProcessTime.name	<string>	データの名前（このサンプルでは、固定文字列で 処理時間 ）
avgProcessTime.data	<object[]>	データ
avgProcessTime.data.y	<long>	経過時間のミリ秒
avgProcessTime.data.yForTooltip	<string>	グラフのツールチップに表示する文字列
endActivityInfoList	<object[]>	IM-BloomMakerのグラフエレメント「棒グラフ」のプロパティ series に渡すためのデータ
endActivityInfoList.name	<string>	データの名前（このサンプルでは、固定文字列で 完了プロセス数 ）
endActivityInfoList.data	<double[]>	データ
periodKeyList	<string[]>	IM-BloomMakerのグラフエレメント「棒グラフ」のプロパティ xAxisCategories に渡すためのデータ



図：「入出力設定」

> **コラム**
>
> IM-BloomMakerのグラフエレメントについて
>
> IM-BloomMakerのグラフエレメントについての詳細は、「intra-mart Developer Site」-「IM-BloomMaker グラフエレメントの使い方（基本編）」を参照してください。

1. 検索終了日時を計算する。
   入力では検索の開始日時しか受け取っていないため、検索の完了日時を計算する必要があります。
   このサンプルでは、JavaScript定義を用いて、検索開始日時の11ヶ月後の月末を検索終了日時としています。
   このJavaScript定義は 【チュートリアル】bloommaker_addon_process_definition_graph-get_graph_data_process_instances_tasks のロジックフローで利用しているものと同一です

2. SQL定義でテーブルからデータを取得する。
   ACT_HI_ACTINST テーブルから、取得してきたデータの DURATION_ の値を 月ごとに ACT_ID_ でまとめた平均値を算出します。
   SQL定義の中のクエリは以下のように設定されています。

```
SELECT
   PROCINST.END_ACT_ID_ AS end_act_id,
   SUM(PROCINST.DURATION_) AS end_act_id_sum,
   COUNT(PROCINST.END_ACT_ID_) AS end_act_id_count,
   to_char((PROCINST.END_TIME_ + (interval '0 seconds')), 'yyyy-MM') AS period_key
FROM
   (SELECT
   PROC_DEF_ID_,
   CASE WHEN END_ACT_ID_ IS NULL THEN '*terminateEndEvent'
         ELSE END_ACT_ID_
         END AS END_ACT_ID_,
   DURATION_,
   END_TIME_
   FROM
   ACT_HI_PROCINST) PROCINST
WHERE
   PROCINST.END_TIME_ IS NOT NULL
   AND
   PROCINST.END_TIME_ >= /*startDate*/'2020-01-01'
   AND
   PROCINST.END_TIME_ <= /*endDate*/'2020-08-01'
   AND
   PROCINST.PROC_DEF_ID_ = /*processDefinitionId*/''
GROUP BY to_char((PROCINST.END_TIME_ + (interval '0 seconds')), 'yyyy-MM'), PROCINST.END_ACT_ID_
```

このSQL定義を実行すると、指定したプロセス定義の期間ごとのインスタンスの完了数と合計処理時間を、終了時のアクティビティIDごとに取得できます。
平均処理時間が存在しない期間のデータはこの中には含まれません。

> **コラム**
>
> このSQL文をカスタマイズすることで、独自の集計結果を取得できます。
> このSQL定義の出力に合わせて後述のJavaScript定義を修正する必要があります。

3. JavaScript定義を使用して、出力する形へ整形する。
   SQL定義で取得したデータをIM-BloomMakerのグラフコンポーネントへ渡せる形に整形します。
   最終的にグラフに表示させるためのデータの形は以下です。ツールチップを使用しない場合は `data` は経過時間の実数値の配列とします。
   今回は完了プロセス数と平均処理時間の二種類のグラフ用のデータを作成します。

```
[{
   name: "データ名"
   data: [                    // そのデータの変化の推移
      {
         y: 100000             // 経過時間の実数値
         yForTooltip: "1分40秒"  // グラフのツールチップに表示するための経過時間をみやすい形に整形した文字列
      },
      {
         y: 10
         yForTooltip: "1秒未満"
      },
      {
         y: 50000
         yForTooltip: "50秒"
      }
   ]
}]
```

一例として、本サンプルでは以下のような処理で実現しています。
平均処理時間については、そのままグラフに表示してしまうとわかりにくくなってしまいます。
そのため、グラフのツールチップに表示するための、読みやすい文字列に変換した値も合わせて生成しています。

```
/**
* mkPeriodKey
*
* 与えられた日付を"YYYY-MM"の形の文字列に変換する。
*
* @param input {date} - 変換元の日付データ
* @return {string} "YYYY-MM"形式の文字列
*/
function mkPeriodKey(date) {
   let monthVal = ('0' + (date.getMonth() + 1)).slice(-2);
   return date.getFullYear() + '-' + monthVal;
}

/**
* calcData
```

```
* calcData
*
* 与えられた期間で終了したプロセスインスタンスの合計と、平均処理時間を算出する。
*
* @param summaryEntity {Object[]} - SQL定義で取得したデータ
* @param summaryEntity.period_key {string} - YYYY-MM 形式の文字列
* @param summaryEntity.end_act_id {string} - アクティビティID
* @param summaryEntity.end_act_id_sum {long} - 合計処理時間
* @param summaryEntity.end_act_id_count {long} - 終了したインスタンス数
* @param periodKey {string} - "YYYY-MM"形式の文字列
* @return {Number[]} [その期間で終了したインスタンス数の合計, 平均処理時間]
*/
function calcData(summaryEntity, periodKey) {
    let filterdData = summaryEntity.filter(function (v) {
        return v.period_key === periodKey;
    });

    let processingTimeSum = filterdData.reduce(function(acc, current) {return acc + current.end_act_id_sum}, 0);
    let processInstanceCount = filterdData.reduce(function(acc, current) {return acc + current.end_act_id_count}, 0);

    let endActivitiEvent = filterdData.map(function(v) {
      return {
        id: v.end_act_id,
        data: v.end_act_id_count
      };
    });

    if (processInstanceCount === 0) {
        return [endActivitiEvent, 0];
    }
    return [endActivitiEvent, processingTimeSum/processInstanceCount];
}

/**
* timeFormat
*
* 経過時間のミリ秒を人が理解しやすい形に変換する。
*
* @param milSec {long} - 変換対象のミリ秒
* @param locale {string} - ロケールの文字列
* @return {string} 変換後の文字列
*/
function timeFormat(milSec, locale) {
  let i18n = {
    LESS_THAN_SECOND: {
      en: "Less than 1s",
      ja: "1秒未満",
      zh_CN: "1秒不足"
    },
    DAYS: {
      en: "Day(s)",
      ja: "日",
      zh_CN: "日"
    },
    HOURS: {
      en: "h",
      ja: "時間",
      zh_CN: "□□"
    },
    MINUTES: {
      en: "min",
      ja: "分",
      zh_CN: "分"
    },
    SECONDS: {
      en: "s",
      ja: "秒",
      zh_CN: "秒"
    }
  };
  let TO_SEC = 1000;
  let TO_MIN = TO_SEC * 60;
  let TO_HOUR = TO_MIN * 60;
  let TO_DAY = TO_HOUR * 24;

  if (milSec < TO_SEC) {
    return i18n.LESS_THAN_SECOND[locale];
  }

  let str = '';
```

```
  let _milSec = milSec;
  if (_milSec >= TO_DAY) {
    let val = Math.floor(_milSec/TO_DAY);
    str += val + i18n.DAYS[locale];
    _milSec -= val*TO_DAY;
  }
  if (_milSec >= TO_HOUR) {
    let val = Math.floor(_milSec/TO_HOUR);
    str += val + i18n.HOURS[locale];
    _milSec -= val*TO_HOUR;
  }
  if (_milSec >= TO_MIN) {
    let val = Math.floor(_milSec/TO_MIN);
    str += val + i18n.MINUTES[locale];
    _milSec -= val*TO_MIN;
  }
  if (_milSec >= TO_SEC) {
    let val = Math.floor(_milSec/TO_SEC);
    str += val + i18n.SECONDS[locale];
    _milSec -= val*TO_SEC;
  }
  return str;
}

/**
* timeList2graphDataList
*
* 経過時間のミリ秒のリストをBloomMakerのグラフ表示用の形式に変換します。
*
* @param timeList {long[]} - 変換対象のミリ秒のリスト
* @param locale {string} - ロケールの文字列
* @return {Object} BloomMakerのグラフ表示用の形式({y: 実数値, yForTooltip: ツールチップ用の値})
*/
function timeList2graphDataList(timeList, locale) {
  return timeList.map(function (v) {
    return {
      y: v,
      yForTooltip: timeFormat(v, locale)
    };
  });
}

/**
* run.
*
* @param input {Object} - task input data.
* @param input.summaryEntity {Object[]} - SQL定義で取得したデータ
* @param input.summaryEntity.period_key {string} - YYYY-MM 形式の文字列
* @param input.summaryEntity.end_act_id {string} - アクティビティID
* @param input.summaryEntity.end_act_id_sum {long} - 合計処理時間
* @param input.summaryEntity.end_act_id_count {long} - 終了したインスタンス数
* @param input.startDate {date} - 集計開始期間
* @param input.processDefinitionId {string} プロセス定義ID
* @return {Object} task result.
*/
function run(input) {
    let startDateFullYear = input.startDate.getFullYear();
    let startDateMonth = input.startDate.getMonth();

    let i18nTerminateEndEventLabel = {
      en: "Forced Termination",
      ja: "強制停止",
      zh_CN: "□制停止"
    };

    let id2nameDict = {'*terminateEndEvent': i18nTerminateEndEventLabel[input.locale]};
    let bpmRepositoryService = new bpm.RepositoryService();
    let bpmModelResource = bpmRepositoryService.getBpmnModel(input.processDefinitionId);
    bpmModelResource.data.processes.forEach(function (processes) {
      processes.flowElements.forEach(function(elements) {
        id2nameDict[elements.id] = elements.name;
      });
    });

    // 範囲となる月日の文字列のリストを作成
    let periodKeyList = [];
    for (let i=0; i<12; i++) {
        let targetMonth = i + startDateMonth;
        let targetDate = new Date(startDateFullYear, targetMonth);
        periodKeyList.push(mkPeriodKey(targetDate));
```

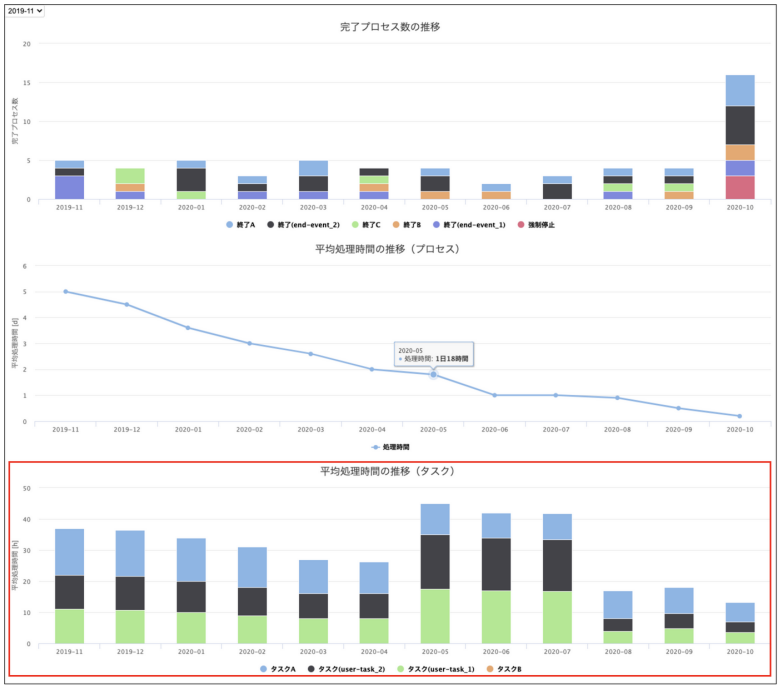
```
    periodKeyList.push(mkPeriodKey(targetDate));
  }

  // 今回出てくるアクティビティIDのリストを作成
  let endActIdList = input.summaryEntity.map(function (v) {
    return v.end_act_id;
  }).filter(function (v, i, array) {
    return array.indexOf(v) === i;
  });

  // 期間ごとの完了したインスタンスの合計数と平均処理時間を算出
  let processInstanceCntList = [];
  let averageProcessingTimeList = [];
  periodKeyList.forEach(function (periodKey) {
    let data = calcData(input.summaryEntity, periodKey, endActIdList);
    let processInstanceCnt = data[0];
    let averageProcessingTime = data[1];
    processInstanceCntList.push(processInstanceCnt);
    averageProcessingTimeList.push(averageProcessingTime);
  });

  let i18nAvgProcessTimeLabel = {
    en: "Processing time",
    ja: "処理時間",
    zh_CN: "□理□□"
  };

  let avgProcessTime = [{
    name: i18nAvgProcessTimeLabel[input.locale],
    data: timeList2graphDataList(averageProcessingTimeList, input.locale)
  }];

  let duplicateNameList = endActIdList.map(function(actId) {
   return id2nameDict[actId];
  }).filter(function(v, i ,arr) {
   return arr.indexOf(v) !== i;
  });

  let endActivityInfoList = endActIdList.map(function(actId) {
   return {
    name: duplicateNameList.indexOf(id2nameDict[actId]) < 0 ? id2nameDict[actId] : id2nameDict[actId] + '(' + actId + ')',
    data: processInstanceCntList.map(function(priodData) {
     let res = priodData.filter(function(v) {
      return v.id === actId;
     });
     return res.length === 0 ? 0 : res[0].data;
    })
   };
  });

  return {
    avgProcessTime: avgProcessTime,
    endActivityInfoList: endActivityInfoList,
    periodKeyList: periodKeyList
  };
}
```

> **コラム**
>
> 独自の集計結果を表示させるためには、前述のSQL定義の出力に合わせてJavaScript定義の入力と整形の処理を修正する必要があります。

4. JavaScript定義の結果を出力に繋ぎます。
   - これで、入力として検索開始日時とプロセス定義IDを受け取り、IM-BloomMakerのグラフエレメント用のデータを出力するロジックフローが完成しました。

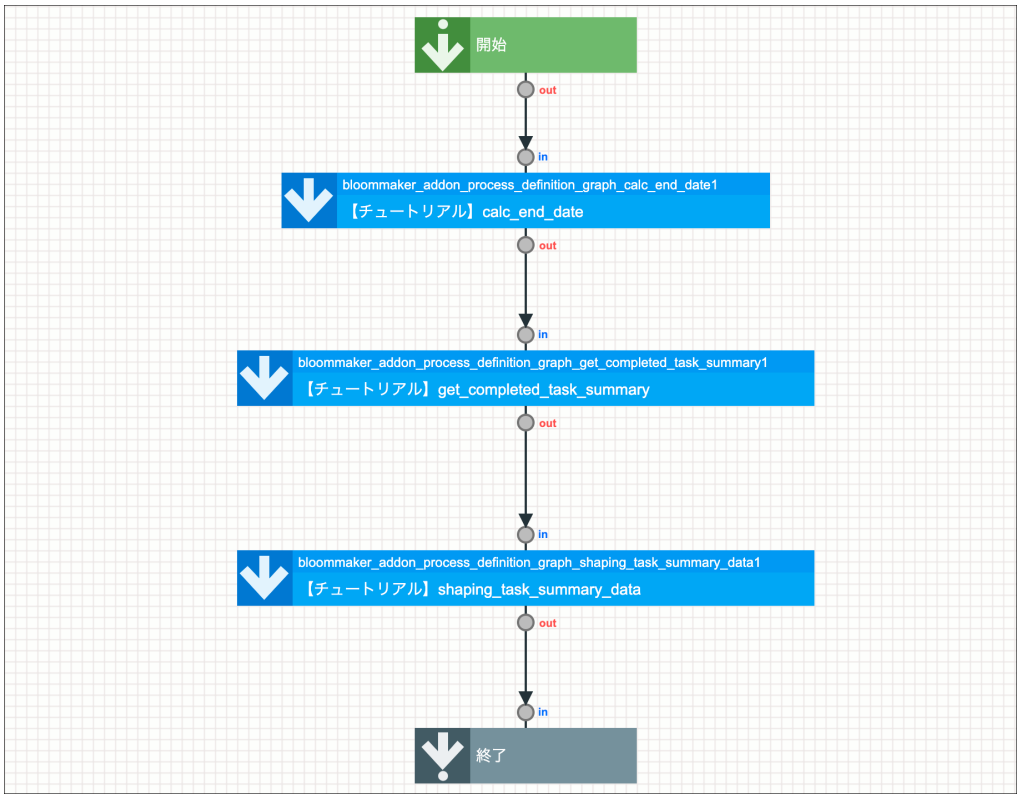
ロジックフロー（【チュートリアル】bloommaker_addon_process_definition_graph-get_graph_data_process_instances_tasks）

画面に表示されるグラフのうち、下図の赤枠の部分で表示されているグラフの表示に必要なデータを、データベースからIM-LogicDesignerを使用して取得する方法を説明します。



図：完成イメージ

プロセス定義の完了したタスクの情報を1ヶ月ごとに集計し、IM-BloomMakerのグラフエレメントへ渡すことのできる形式へ整形します。



図：「ロジックフロー（【チュートリアル】bloommaker_addon_process_definition_graph-get_graph_data_process_instances_tasks）」

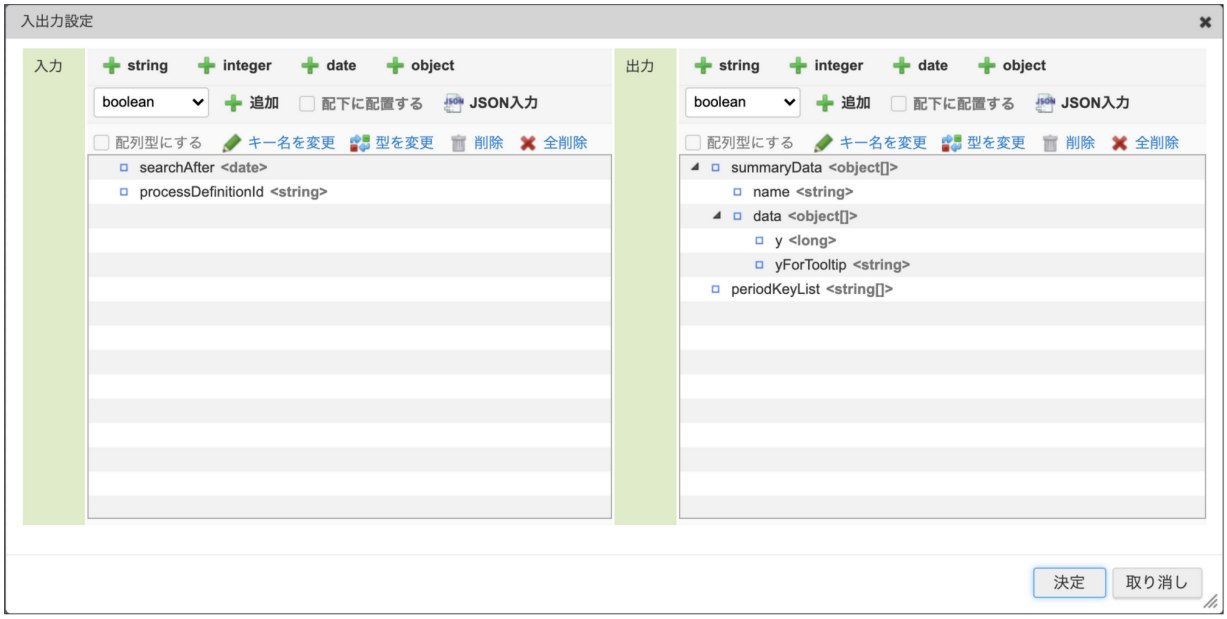
このロジックフローの入出力設定は以下のように設定されています。
入力として検索開始日時とプロセス定義IDを受け取ります。
出力はIM-BloomMakerのグラフエレメント「棒グラフ」のプロパティである `series` と `xAxisCategories` へ渡せる状態に整形されたデータです。

- 入力

キー名	型	説明
searchAfter	<date>	検索開始日時
processDefinitionId	<string>	プロセス定義ID

- 出力

キー名	型	説明
summaryData	<object[]>	IM-BloomMakerのグラフエレメント「棒グラフ」のプロパティ series に渡すためのデータ
summaryData.name	<string>	データの名前（このサンプルでは、タスク名）
summaryData.data	<object[]>	データ
summaryData.data.y	<long>	経過時間のミリ秒
summaryData.data.yForTooltip	<string>	グラフのツールチップに表示する文字列
periodKeyList	<string[]>	IM-BloomMakerのグラフエレメント「棒グラフ」のプロパティ xAxisCategories に渡すためのデータ



図：「入出力設定」

> **コラム**
>
> IM-BloomMakerのグラフエレメントについて
>
> IM-BloomMakerのグラフエレメントについての詳細は、「intra-mart Developer Site」-「IM-BloomMaker グラフエレメントの使い方（基本編）」を参照してください。

1. 検索終了日時を計算する。
   入力では検索の開始日時しか受け取っていないため、検索の完了日時を計算する必要があります。
   このサンプルでは、JavaScript定義を用いて、検索開始日時の11ヶ月後の月末を検索終了日時としています。

2. SQL定義でテーブルからデータを取得する。
   ACT_HI_ACTINST テーブルから、取得してきたデータの DURATION_ の値を 月ごとに ACT_ID_ でまとめた平均値を算出します。
   SQL定義の中のクエリは以下のように設定されています。

```
SELECT
    to_char((ACTINST.END_TIME_ + (interval '0 seconds')), 'yyyy-MM') AS period_key,
    ACT_ID_ AS act_id,
    ACT_NAME_ AS act_name,
    AVG(DURATION_) AS processing_time
FROM
    ACT_HI_ACTINST ACTINST
WHERE
    ACTINST.END_TIME_ IS NOT NULL
    AND
    ACTINST.END_TIME_ >= /*startDate*/'2020-01-01'
    AND
    ACTINST.END_TIME_ <= /*endDate*/'2020-06-01'
    AND
    ACTINST.PROC_DEF_ID_ = /*processDefinitionId*/''
    AND
    ACTINST.ACT_TYPE_ IN ('userTask', 'receiveTask', 'callActivity', 'imDraftWorkflowServiceTask', 'imApplyWorkflowServiceTask',
'imDraftFormaServiceTask', 'imApplyFormaServiceTask', 'imDraftBisServiceTask', 'imApplyBisServiceTask')
GROUP BY to_char((ACTINST.END_TIME_ + (interval '0 seconds')), 'yyyy-MM'), ACT_ID_, ACT_NAME_
ORDER BY period_key
```

このSQL定義を実行すると、アクティビティIDの特定期間の平均処理時間が取得できます。
平均処理時間が存在しないアクティビティIDの特定期間のデータはこの中には含まれません。

> **コラム**
>
> このSQL文をカスタマイズすることで、独自の集計結果を取得できます。
> このSQL定義の出力に合わせて後述のJavaScript定義を修正する必要があります。

3. JavaScript定義を使用して、出力する形へ整形する。
   SQL定義で取得したデータをIM-BloomMakerのグラフコンポーネントへ渡せる形に整形します。
   最終的にグラフに表示させるためのデータの形は以下です。
   表示させたいタスクの数だけこのデータを作成し、配列にする必要があります。

```
[{
    name: "データ名"
    data: [                    // そのデータの変化の推移
        {
            y: 100000            // 経過時間の実数値
            yForTooltip: "1分40秒"  // グラフのツールチップに表示するための経過時間をみやすい形に整形した文字列
        },
        {
            y: 10
            yForTooltip: "1秒未満"
        },
        {
            y: 50000
            yForTooltip: "50秒"
        }
    ]
}]
```

一例として、本サンプルでは以下のような処理で実現しています。

```
/**
* mkPeriodKey
*
* 与えられた日付を"YYYY-MM"の形の文字列に変換する。
*
* @param input {date} - 変換元の日付データ
* @return {string} "YYYY-MM"形式の文字列
*/
function mkPeriodKey(date) {
    let monthVal = ('0' + (date.getMonth() + 1)).slice(-2);
    return date.getFullYear() + '-' + monthVal;
}

/**
* getProcessingTime
*
* summaryEntity から指定されたタスク名・日付に該当する処理時間を返す。
* 存在しない場合は0を返す。
*
* @param summaryEntity {Object[]} - SQL定義で取得したデータ
* @param summaryEntity.period_key {string} - YYYY-MM 形式の文字列
* @param summaryEntity.act_id {string} - アクティビティID
```

```
* @param summaryEntity.processing_time {long} - タスクの処理時間
* @param summaryEntity.act_name {string} - タスク名
* @param actName {string} - 抽出対象のタスク名
* @param periodKey {string} - 抽出対象の日付文字列
* @return {number} 平均処理時間
*/
function getProcessingTime(summaryEntity, actName, periodKey) {
    let filterdData = summaryEntity.filter(function (v) {
        return v.act_name === actName && v.period_key === periodKey;
    });
    if (filterdData.length === 0) {
        return 0;
    }
    return filterdData[0].processing_time;
}

/**
* timeFormat
*
* 経過時間のミリ秒を人が理解しやすい形に変換する。
*
* @param milSec {long} - 変換対象のミリ秒
* @param locale {string} - ロケールの文字列
* @return {string} 変換後の文字列
*/
function timeFormat(milSec, locale) {
  let i18n = {
    LESS_THAN_SECOND: {
      en: "Less than 1s",
      ja: "1秒未満",
      zh_CN: "1秒不足"
    },
    DAYS: {
      en: "Day(s)",
      ja: "日",
      zh_CN: "日"
    },
    HOURS: {
      en: "h",
      ja: "時間",
      zh_CN: "□□"
    },
    MINUTES: {
      en: "min",
      ja: "分",
      zh_CN: "分"
    },
    SECONDS: {
      en: "s",
      ja: "秒",
      zh_CN: "秒"
    }
  };
  let TO_SEC = 1000;
  let TO_MIN = TO_SEC * 60;
  let TO_HOUR = TO_MIN * 60;
  let TO_DAY = TO_HOUR * 24;

  if (milSec === null || milSec < TO_SEC) {
    return i18n.LESS_THAN_SECOND[locale];
  }

  let str = '';
  let _milSec = milSec;
  if (_milSec >= TO_DAY) {
    let val = Math.floor(_milSec/TO_DAY);
    str += val + i18n.DAYS[locale];
    _milSec -= val*TO_DAY;
  }
  if (_milSec >= TO_HOUR) {
    let val = Math.floor(_milSec/TO_HOUR);
    str += val + i18n.HOURS[locale];
    _milSec -= val*TO_HOUR;
  }
  if (_milSec >= TO_MIN) {
    let val = Math.floor(_milSec/TO_MIN);
    str += val + i18n.MINUTES[locale];
    _milSec -= val*TO_MIN;
  }
  if (_milSec >= TO_SEC) {
```

```
  if (_milSec >= TO_SEC) {
    let val = Math.floor(_milSec/TO_SEC);
    str += val + i18n.SECONDS[locale];
    _milSec -= val*TO_SEC;
  }
  return str;
}

/**
* timeList2graphDataList
*
* 経過時間のミリ秒のリストをBloomMakerのグラフ表示用の形式に変換します。
*
* @param timeList {long[]} - 変換対象のミリ秒のリスト
* @param locale {string} - ロケールの文字列
* @return {Object} BloomMakerのグラフ表示用の形式({y: 実数値, yForTooltip: ツールチップ用の値})
*/
function timeList2graphDataList(timeList, locale) {
  return timeList.map(function (v) {
    return {
      y: v,
      yForTooltip: timeFormat(v, locale)
    };
  });
}

/**
* run.
*
* @param input {Object} - task input data.
* @param input.summaryEntity {Object[]} - SQL定義で取得したデータ
* @param input.summaryEntity.period_key {string} - YYYY-MM 形式の文字列
* @param input.summaryEntity.act_id {string} - アクティビティID
* @param input.summaryEntity.processing_time {long} - タスクの処理時間
* @param input.summaryEntity.act_name {string} - タスク名
* @param input.startDate {date} - 集計開始期間
* @return {Object} task result.
*/
function run(input) {
    let startDateFullYear = input.startDate.getFullYear();
    let startDateMonth = input.startDate.getMonth();

    // 範囲となる月日の文字列のリストを作成
    let periodKeyList = [];
    for (let i=0; i<12; i++) {
        let targetMonth = i + startDateMonth;
        let targetDate = new Date(startDateFullYear, targetMonth);
        periodKeyList.push(mkPeriodKey(targetDate));
    }

    // 今回出てくるアクティビティの情報のリストを作成
    let actInfoList = input.summaryEntity.map(function (v) {
        return {
          id: v.act_id,
          name: v.act_name
        };
    }).filter(function (v, i, array) {
        return array.map(function(info) {
          return info.id;
        }).indexOf(v.id) === i;
    });

    // 別IDでアクティビティ名が重複しているものを探す
    let duplicateNameList = actInfoList.map(function(info) {
      return info.name;
    }).filter(function(v, i ,arr) {
      return arr.indexOf(v) !== i;
    });

    // ActivityNameそれぞれに対してグラフ用のデータを作成する。
    let summaryEntity = actInfoList.map(function (actInfo) {
        let timeList = periodKeyList.map(function (periodKey) {
          return getProcessingTime(input.summaryEntity, actInfo.name, periodKey);
        });
        return {
          name: duplicateNameList.indexOf(actInfo.name) < 0 ? actInfo.name: actInfo.name + '(' + actInfo.id + ')',
          data: timeList2graphDataList(timeList, input.locale)
        };
    });
```

```
    return {
      summaryData: summaryEntity,
      periodKeyList: periodKeyList
    };
}
```

> **コラム**
>
> 独自の集計結果を表示させるためには、前述のSQL定義の出力に合わせてJavaScript定義の入力と整形の処理を修正する必要があります。

4. JavaScript定義の結果を出力に繋ぎます。
    - これで、入力として検索開始日時とプロセス定義IDを受け取り、IM-BloomMakerのグラフエレメント用のデータを出力するロジックフローが完成しました。

ルーティング定義

1. ルーティング定義の作成。
    - 作成したロジックフローをREST APIとして呼び出せるようにルーティング定義を作成します。
        - 本サンプルではそれぞれのロジックフローに対し、以下のルーティングを設定しています。
            - 【チュートリアル】bloommaker_addon_process_definition_graph-get_graph_data_process_instances_tasks: `api/bpm/tutorial/process-summary-of-completed-activities-group-by-month`
            - 【チュートリアル】bloommaker_addon_process_definition_graph-get_graph_data_process_instances: `api/bpm/tutorial/process-summary-of-completed-instances-group-by-month`
    - 作成した後に任意の認可を設定します。
        - 本サンプルでは、「IM-BPM管理者」に対して「実行」権限を付与しています。
    - 以上でIM-BloomMakerからロジックフローを呼び出す準備が完了しました。

> **コラム**
>
> ルーティングの認可設定の詳細は「IM-LogicDesigner ユーザ操作ガイド」-「フロールーティングの認可設定」を参照してください。

IM-BloomMaker定義情報（**im_bloommaker-data-bloommaker_addon_process_definition_graph.zip**）の詳細

コンテンツ定義

IM-LogicDesignerで作成したREST APIを利用して、プロセス定義の実行状態のグラフを表示する画面を作成します。
画面を作成する上でのポイントは以下の点です

- 対象のプロセス定義IDを画面のURLから判断する。
- アクションでIM-LogicDesignerで作成したAPIを使用してグラフ表示用のデータを取得する。
- Y軸が平均処理時間を表す場合は、y軸の単位を適切な値に変換して表示する。
- 取得してきたデータをグラフエレメントへマッピングする。
- ページを開いたタイミングで、データを取得し描画する。

1. 対象のプロセス定義IDを画面のURLから受け取る。
   画面のURLに入力されている値をIM-BloomMakerのコンテンツ定義で使用する場合はルーティング定義のURLの設定と入力値の設定が必要です。
   この入力値はリクエストパラメータとしてREST APIでのデータ取得で使用するため、リクエストパラメータ用の変数に代入をしてください。

   > **コラム**
   >
   > 入力値の設定に関する詳細は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」 - 「入力の設定方法」を参照してください。

2. アクションでIM-LogicDesignerで作成したAPIを使用してグラフ表示用のデータを取得する。
   アクションエディタにて、アクションアイテム `URL「」にリクエストを送信する` を使用してデータを取得します。

   > **コラム**
   >
   > アクションの設定に関する詳細は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」 - 「アクションを設定する」を参照してください。

3. Y軸が平均処理時間を表す場合は、y軸の単位を適切な値に変換して表示する。
   取得してきたデータのままだと、平均処理時間の単位がミリ秒となっているので、必要に応じて分や日などに変換をする必要があります。
   アクションアイテム `URL「」にリクエストを送信する` の下に `カスタムスクリプトを実行する` を配置します。
   カスタムスクリプトで平均処理時間の単位を変更処理を記述することにより、グラフに表示した際のY軸の値がわかりやすくなります。

4. 取得してきたデータをグラフエレメントへマッピングする。
    - 「グラフ」に分類されるエレメントの一覧から、「折れ線グラフ」または「棒グラフ」を設置します。
    - IM-LogicDesignerで作成したロジックフローの出力には以下の2種類の情報が含まれていますので、それぞれマッピングします。
        - グラフエレメントのプロパティ `series` にマッピングするためのグラフのメインとなるデータ
        - グラフエレメントのプロパティ `xAxisCategories` にマッピングするためのX軸のラベルに表示する文字列のデータ
    - ツールチップを表示させる場合は、グラフエレメントのプロパティ `tooltipPointFormat` に以下の値を設定します。
        - `<span style="color:{point.color}">●</span> {series.name}: <b>{point.yForTooltip}</b><br/>`

5. ページを開いたタイミングで、データを取得し描画する。
    - 作成したアクションを `アクション「」を実行する` を利用してひとまとめにしたアクションを作成します。
    - 「コンテナ」を選択した状態で、プロパティの「ページ読み込み時」の項目で作成したアクションを指定します。

6. 集計開始日時を指定させる。
   プロセス定義詳細画面にあるように集計の開始日時を選択可能とするには、以下の方法で実現できます。

    - プルダウンに表示させる値のラベルと、値の配列をそれぞれ変数に作成する。
    - 「フォーム部品」に分類されるエレメントの一覧から、「プルダウン」を設置する
    - プロパティ `labels` と `values` にそれぞれの配列を指定し、 `value` にリクエストパラメータの `searchAfter` の変数を指定する。
    - 選択が変更された時点でデータを取得するようにするため、プルダウンエレメントの入力値変更時イベントにデータ取得処理のアクションを指定する。

ルーティング定義

1. コンテンツ定義で作成した画面にアクセスをするために、ルーティング定義を作成します。
   ルーティング定義を作成する際のURLは、コンテンツ定義の入力値に渡すプロセス定義を含めた動的URLとする必要があります。

> **コラム**
>
> ルーティングの登録に関する詳細は「IM-BloomMaker for Accel Platform ユーザ操作ガイド」 - 「ルーティングを新規登録する」を参照してください。

2. ルーティング定義に認可を設定する。
   作成したルーティングにアクセスするためには、適切な認可の設定が必要です。
   本サンプルでは、「IM-BPM管理者」に対して「参照」権限を付与しています。

> **コラム**
>
> ルーティングの認可設定の詳細は「IM-BloomMaker for Accel Platform チュートリアルガイド」-「ルーティングの認可を設定する」を参照してください。