目次

改訂情報

変更年月日	変更内容
2016-08-01	初版
2016-08-31	第2版 下記のページの表記を修正しました。 ■ 「*概要*」 ■ 「*IM-Workflow/Forma/IM-BIS連携*」 ■ 「*マイグレーション*」 ■ 「*Elasticsearch連携機能*」
2016-12-01	第3版 下記のページの「業務キー」登録方法について加筆しました。 ■ 「*概要*」 ■ 「*プロセス*」
2017-04-01	第4版 下記を追加・変更しました。 ■ 「*業務キー*」に業務キーの入力有無の設定について加筆しました。 ■ 「*インポート/エクスポート*」を追加しました。 ■ 「*REST APIについて*」に「インポート/エクスポート」を追加しました。 ■ 「*フォームキー*」に「redirect:(PATH)」を加筆しました。 ■ 「*前処理と後処理*」に後処理の実行処理種別「一時保存」について加筆しました。 ■ 「*アーカイブ*」を追加しました。 ■ 「*ジョブ一覧*」を追加しました。 ■ 「*関連ドキュメント*」を追加しました。 ■ 「*ユーザタスク処理者の取得*」にシステム変数の格納方式の設定について加筆しました。 ■ 「*起票による連携*」にシステム変数の格納方式の設定について加筆しました。 ■ 「*申請による連携*」にシステム変数の格納方式の設定について加筆しました。 ■ 「*暗黙オブジェクト*」にシステム変数の格納方式の設定について加筆しました。
2017-08-01	第5版 下記について加筆しました。 ■ 「*権限*」に、追加されたロールおよび認可リソースについて加筆しました。 ■ 「*インポート/エクスポート*」に、プロセスデザイナのインポート/エクスポートについて加筆しました。
2017-12-01	第6版 下記を追加しました。 ■ 「*画面仕様*」を追加しました。
2017-12-22	第7版 下記について加筆しました。 ■ 「*トランザクション*」に、「非同期処理」および「排他制御と非排他制御」について加筆しました。 ■ 「タスク失敗時のトランザクション」の記述内容を「*ジョブ*」に統合しました。

変更年月日	変更内容
2018-04-01	第8版 下記を追加しました。 ▪ 「*IM-LogicDesigner リスナ*」を追加しました。

はじめに

本書の目的

本書ではIM-BPM for Accel Platformの機能概要と仕組みの詳細について説明します。

説明範囲は以下のとおりです。

- IM-BPM for Accel Platformの概要と用語
- IM-BPM for Accel Platformの機能仕様および機能詳細

対象読者

本書では次の開発者を対象としています。

- IM-BPM for Accel Platformの仕組みを理解したい
- IM-BPM for Accel Platformを利用して処理を実装したい
- IM-BPM for Accel Platformと連携した機能を実装したい

なお、本書では次の内容を理解していることが必須となります。

- intra-mart Accel Platformを理解している

本書の構成

本書は次の構成となっています。

- *概要*

  IM-BPM for Accel Platformの全体像、および、本書で利用する用語について説明します。

- *機能仕様*

  IM-BPM for Accel Platformの提供する各機能の詳細な仕組みについて説明します。

概要

IM-BPM for Accel Platform とは

IM-BPM for Accel Platform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 for Accel Platformの特徴は以下の通りです。

- BPMN2.0形式に対応した業務プロセスを実行することができます。
- IM-FormaDesignerと連携し、システムに必要となる画面と簡単に連携することができます。
- IM-LogicDesignerと連携し、システム間連携等の業務ロジックを簡単に呼び出すことができます。
- IM-BPM for Accel Platformでは、IM-Workflow/IM-BISと連携し、複数のワークフローを含めた業務プロセスを管理することができます。
- 作成した業務プロセスは、その業務プロセスの実行方法をモニタリングすることができます。
  これにより業務プロセスの見直しを図り改善につなげることが可能です。
- 作成した業務プロセスはREST APIにより外部システムから呼び出すことができます。

用語

BPMN

ビジネスプロセスモデリング表記法 (Business Process Modeling Notation) を指します。
IM-BPM for Accel Platformでは、BPMN2.0に対応しています。

プロセスダイアグラム

BPMNにより記述された業務プロセス図を指します。

プロセス定義

業務プロセスの定義した単位を指します。
BPMN2.0により記述されたダイアグラム図及びその実行に必要な情報を含みます。
プロセス定義は、バージョンによる管理が行われます。

プロセス定義キー

業務プロセスに対し付与された一意となるIDです。
IM-BPM Designer上で設定を行います。

プロセス定義ID

デプロイされたプロセス定義に対し一意に割り当てられるIDです。
プロセス定義IDはデプロイ時に決定されるため任意のIDを割り当てることは出来ません。
プロセス定義IDのフォーマットは {プロセス定義キー}:{バージョン番号}:{自動採番されたID} となります。

カテゴリ

プロセス定義をカテゴライズするために利用します。
IM-BPM Designer上で設定を行います。
カテゴリはBPMN2.0 (XML) 内で名前空間として利用されます、その為URI形式で設定することを推奨します。

> 例: http://www.intra_mart.jp/bpmn/MY_CATEGORY

プロセスインスタンス

プロセス定義を元とし、実行されたインスタンスを指します。
一つのプロセス定義に付き複数のプロセスインスタンスが存在します。
プロセスインスタンスの実行元となるプロセス定義及びそのバージョンは固定です。
プロセス定義(バージョンを含む)を元に実行されたプロセスインスタンスは終了するまで同じプロセス定義、バージョンで実行されます。

プロセスインスタンスID

プロセスインスタンスに付与されるIDです。
日付をベースとした一意となるIDが付与されます。

業務キー

プロセスインスタンスに付与可能な一意となるキーです。
プロセスインスタンスIDが自動的に採番されるIDであるのに対して、業務キーは任意のキーを設定することが可能です。
以下の方法により業務キーを登録できます。

- プロセス開始時の確認ダイアログでの入力
- 開始イベントに設定したIM-FormaDesignerのアプリケーションにて、フィールド識別IDを `bpm_business_key` とした画面アイテムでの入力
- APIによる登録

エグゼキューション

プロセスインスタンスに含まれる実行単位を指します。
並列(Parallel Gateway)等が含まれるプロセス定義等、プロセスインスタンスに含まれる並列に実行される一つ一つの単位をエグゼキューションと表現します。
複数のエグゼキューションに分かれるようなプロセスインスタンスの場合、内部で実行されるJavaスレッドも共に複数スレッドで動作します。

デプロイメント

IM-BPM Designerにより作成されたアーティファクト(成果物)をデプロイした単位を指します。
一つのデプロイメントには複数のプロセス定義が含まれる場合があります。
デプロイメントには、実行プログラム、ルール定義等が含まれています、それらは全て同一のデプロイメント内で共有することができます。
その為、一つのデプロイメントに含まれる複数のプロセス定義から、同じ実行プログラム、ルール定義等を共有することができます。

タスク

プロセスインスタンスが、ユーザタスクアクティビティに到達した場合、タスクが生成されます。
タスクは、担当者、処理対象ユーザ、処理対象ロールを設定することができます。
担当者が割り当てられていない状態を表現するグループタスクと、担当者が割り当てられている個人タスクが存在します。

グループタスク

担当者が設定されていない状態のタスクを指します。
グループタスクは、担当者を設定することにより個人タスクとなります。

個人タスク

担当者が設定された状態のタスクを指します。
担当者が設定されているタスクは、その担当者以外操作を行うことは出来ません。

機能仕様

権限

IM-BPM for Accel Platformの権限について説明します。

ロール

IM-BPM for Accel Platformでは、標準で以下の4つのロールが用意されています。

- IM-BPM管理者
- IM-BPMユーザ
- IM-BPMプロセス参照ユーザ
- IM-BPM開発者

IM-BPM管理者 (im_bpm_manager)

IM-BPM for Accel Platformに関するすべての権限を有するロールです。
プロセスの管理からREST APIの呼び出しまですべての権限を有しています。

IM-BPMユーザ (im_bpm_user)

IM-BPM for Accel Platformを利用者を想定したロールです。
IM-BPMユーザは、IM-BPM管理者により用意された業務プロセスを実施する役割を持ちます。
IM-BPMユーザは、業務プロセスに対する操作、参照する権限は有していません。
IM-BPMユーザはタスクに関する操作のみがおこなえます。

IM-BPMプロセス参照ユーザ (im_bpm_process_reference_user)

IM-BPM管理者により用意された業務プロセスを参照する権限を有したロールです。
IM-BPMユーザロールと組み合わせて利用することを想定しています。
IM-BPMユーザロールと組み合わせることにより実行されたプロセスインスタンスに対する履歴の参照、業務プロセス図の参照を行うことができます。

IM-BPM開発者 (im_bpm_developer)

プロセスデザイナを利用してプロジェクトおよびプロセスを開発する権限を有したロールです。
プロジェクトに対して特定のサブジェクト（会社・組織・ロールなど）に認可を設定しておき、そのサブジェクトとIM-BPM開発者ロールを組み合わせて利用することを想定しています。
IM-BPM開発者は、自分の所属会社・組織などに応じて、許可されたプロジェクトについて許可された処理を行うことができます。

認可

IM-BPM for Accel Platformでは、以下の4種類のリソース種別が用意されています。

- IM-BPM REST API

- 画面・処理
- IM-BPM デザイナ プロジェクト
- IM-BPM デザイナ REST API

IM-BPM REST API

IM-BPM for Accel Platformが提供するREST APIに関する認可リソース種別です。
IM-BPMに関するREST APIに関する権限設定を行うことが可能です。

> **コラム**
>
> Swagger UIを利用してAPIを実行する。
>
> IM-BPM REST APIは全てWeb API Makerを利用しています。
> その為、REST APIに対するSwagger定義が参照できます。
> Swagger UIを利用することにより IM-BPMで提供されているAPIを画面上から確認、実行することが可能です。
> http(s)://{HOST}:{PORT}/{CONTEXT_PATH}/swagger_ui/?url=/{CONTEXT_PATH}/api-docs/im_bpm_rest へアクセスすることにより確認がおこなえます。
> ※ プロトコル、HOST, PORT, CONTEXT_PATHは環境に合わせて置換してください。

> **注意**
>
> REST APIの権限に関して
>
> IM-BPMが提供する画面はAjaxによりREST APIの呼び出しが行われています。
> その為、REST APIの認可の設定次第では正常に画面が表示されなくなる場合があります。

画面・処理

IM-BPM for Accel Platformが提供する画面に関するリソース種別です。
IM-BPM for Accel Platformデフォルトでは上記ロール、および、テナント管理者に対して認可設定が行われています。
プロセスデザイナについては、デフォルトではIM-BPM管理者とIM-BPM開発者に対して認可設定が行われています。

IM-BPM プロセスデザイナ プロジェクト

プロセスデザイナが提供するプロジェクトに関するリソース種別です。
プロジェクト別に設定し、IM-BPM開発者に対してどのプロジェクトのどのアクションを許可するかの権限設定を行うことが可能です。
以下の4つのアクションを保持しています。

- 管理
- 編集
- デプロイ
- 参照

IM-BPM管理者に対しては、デフォルトで全てのプロジェクトの全てのアクションが許可されています。

IM-BPM プロセスデザイナ REST API

プロセスデザイナが提供するREST APIに関する認可リソース種別です。
プロセスデザイナ画面から呼び出されるREST APIに関する権限設定を行うことが可能です。

デプロイメント

デプロイメントの仕様について説明します。

永続化

IM-BPM for Accel Platformに対してデプロイを行った場合、デプロイされた情報は全てデータベースに永続化されます。

> **コラム**
>
> ストレージの利用
>
> IM-BPM for Accel Platformでは、テナント環境セットアップ用の資材を除いてストレージを使用していません。

クラスローダ

IM-BPM for Accel Platformではデプロイメント単位にクラスローダが用意されます。
これにより、同一デプロイメントにおいてサービスタスク等のプログラムが関連するプログラムの共有が可能です。
クラスローダが用意されるタイミングは、そのデプロイメントに関するリソースに対するアクセスが発生したタイミングとなります。
したがって、プロセス定義の参照や、プロセスインスタンスの実行等が行われなかった場合にはクラスローダが作成されることはありません。

リソースの展開先

クラスローダはそのクラスローダを生成する直前に、デプロイメントに含まれるリソースを以下の場所に展開します。

{Webアプリケーション配備先}/WEB-INF/work/_jar/{テナントID}/{デプロイメントID}配下

アンデプロイ時にはディレクトリを含めて削除が行われます。

> **注意**
>
> デプロイメントに含まれるリソースに関して
>
> デプロイメントに含まれるjarファイル等は全て展開された状態となります。
> したがって、複数のjarに同一のリソースは含めないようにしてください。
> また、直接配備先ディレクトリ配下のファイルを編集、差し替えは行わないでください。

> **コラム**
>
> 配備先ディレクトリの削除
>
> アプリケーションサーバが停止している状態であれば配備先ディレクトリは削除することが可能です。
> 削除後、デプロイメントされたリソースに対する参照が行われたタイミングで再度配備が行われます。

リソースの配置先に関して

デプロイメント用のクラスローダは、親クラスローダとしてWebアプリケーションの持つクラスローダが設定されています。
その為、実行プログラム(Javaクラス、ルール定義)は、デプロイメントに含めず、アプリケーションとして配備することも可能です。
アプリケーションとして事前に作成、配備する場合にはeBuilder等を用いてIM-Jugglingユーザモジュールを作成しwarファイルに含めてください。

アンデプロイ

アンデプロイが行われた場合、デプロイしたリソースに関する実行情報、履歴も共に削除されます。
その為、デプロイメントに含まれていたプロセス定義、及びプロセスインスタンスは全て削除されます。
プロセス定義を無効化する目的の場合には、プロセス定義自体を無効化してください。

アンデプロイされたプロセスインスタンスとCallActivity

アンデプロイ時にプロセス定義、プロセスインスタンスが削除されますが、別のデプロイメント、別のプロセスインスタンスからCallActivityを利用して
アンデプロイ対象となったプロセスインスタンスを呼び出していた場合、呼び出し元のプロセスは呼び出し先のフロー完了待ちとなり待機状態となります。
この状態を回避する場合には、プロセスインスタンスのマイグレーション機能を利用し、該当のCallActivityに待機しているプロセスインスタンスの移行を行ってください。

プロセス

プロセスについて説明します。

プロセス定義

業務プロセスの定義した単位を指します。
BPMN2.0により記述されたダイアグラム図およびその実行に必要な情報を含みます。

プロセス定義キー

業務プロセスに対し付与されたキーです。
IM-BPM Designer上からIDとして設定を行います。
プロセス定義キーは複数のバージョンのプロセス定義を跨って一意です。

バージョニング

業務プロセスは複数のバージョンを持ちます。
同一のプロセス定義キーは、同一プロセス定義とみなし新しいプロセス定義がデプロイされたタイミングで新しいバージョンとして登録されます。
バージョンを指定せず業務プロセスを開始する場合は常に最新バージョンが実行されます。

プロセス定義ID

デプロイされたプロセス定義に対し一意に割り当てられるIDです。
プロセス定義IDはデプロイ時に決定されるため任意のIDを割り当てることは出来ません。
プロセス定義IDのフォーマットは {プロセス定義キー}:{バージョン番号}:{自動採番されたID} です。
このプロセス定義IDは主にAPI等を利用して業務プロセスを開始させる場合に利用されます。

プロセス定義名

業務プロセスに対し付与された一意となるIDです。
IM-BPM Designer上で設定を行います。

> **注意**
>
> プロセス定義の多言語化に関して
>
> プロセス定義名は単一の項目として扱います。
> その為、多言語化には対応していません。

カテゴリ

プロセス定義をカテゴライズするために利用します。
IM-BPM Designer上で設定を行います。
カテゴリはBPMN2.0 (XML) 内で名前空間として利用されます、その為URI形式で設定することを推奨します。

例: http://www.intra_mart.jp/bpmn/MY_CATEGORY

> **注意**
>
> カテゴリの多言語化に関して
>
> カテゴリは単一の項目です。
> その為、多言語化には対応していません。

状態

プロセス定義は以下の状態を持ちます。

- 有効
- 無効

無効状態のプロセス定義は新規に業務プロセスの開始は行えません。
既に実行中の業務プロセスは処理を行うことができます。

権限

プロセス定義は、そのプロセス定義を開始することが可能な権限情報を持ちます。
権限は、以下の2種類の設定が行えます。

- ロール
- ユーザコード

それぞれの権限は、OR条件にて判断されます。
また、権限はカンマにより区切ることにより複数設定することが可能です。

プロセスインスタンス

プロセス定義を元とし、実行されたインスタンスを指します。
一つのプロセス定義につき複数のプロセスインスタンスが存在します。
プロセスインスタンスの実行元となるプロセス定義およびそのバージョンは固定です。
プロセス定義（バージョンを含む）を元に実行されたプロセスインスタンスは終了するまで同じプロセス定義、バージョンで実行されます。

> **コラム**
>
> 後述するプロセスマイグレーション機能を利用することにより実行中のプロセスインスタンスを別バージョンに移行することができます。

プロセスインスタンスID

プロセスインスタンスに付与されるIDです。
日付をベースとした一意となるIDが付与されます。

> **注意**
> プロセスインスタンスIDの採番方法を別の方法に変更することは出来ません。

業務キー

プロセスインスタンスに付与可能な一意となるキーです。
プロセスインスタンスIDが自動的に採番されるIDであるのに対して、業務キーは任意のキーを設定することが可能です。
以下の方法により業務キーを登録できます。

- プロセス開始時の確認ダイアログでの入力
- 開始イベントに設定したIM-FormaDesignerのアプリケーションにて、フィールド識別IDを `bpm_business_key` とした画面アイテムでの入力
- APIによる登録

> **コラム**
>
> プロセス開始時の確認ダイアログでの入力については、プロセス定義の設定により入力有無を変更することができます。
> 業務キーの入力有無の設定に関しては、「IM-BPM Designer 操作ガイド」を参照してください。

変数

プロセスインスタンスは、その業務プロセスで利用される任意の変数を格納することができます。
変数として利用可能な型は以下の通りです。

- string

  文字列です、4000バイト以内の文字列の場合は、一覧画面等から検索として利用することが可能です。
  4000バイトを超えた場合には、データベース上のBLOBカラムに格納される為検索を行うことは出来ません。

- integer

  数値型です。
  java.lang.Integer型に相当します。

- short

  数値型です。
  java.lang.Short型に相当します。

- long

  数値型です。
  java.lang.Long型に相当します。

- double

  数値型です。
  java.lang.Double型に相当します。

- boolean

  数値型です。
  java.lang.Boolean型に相当します。

- date

  日付型です。
  java.util.Date型に相当します。

- serializable

  上記データ型に該当しない場合には、serializableとみなし扱われます。
  このデータ型の場合、変数を用いた検索には利用できません。
  java.io.Serializableインタフェースを実装したデータ型のみ格納できます。

変数スコープ

変数は2種類のスコープを持ちます。

- プロセスインスタンス

  プロセスインスタンス内からアクセス可能なスコープです。
  主に、アクティビティ間でのデータの受け渡し等が必要となる場合にはこのスコープを利用します。

- タスク

  タスク、アクティビティに閉じたスコープの変数です。
  特定のアクティビティの実行結果等を永続化する場合等に利用します。

> **注意**
>
> 変数の数に関して
>
> 変数に上限は存在しませんが、変数の内容はデータベース上に格納されます。
> 変数が多い場合、レコード数が相対的に増えるためパフォーマンスに影響を与える可能性があります。
> 変数が多い場合には業務で利用されるキーのみを変数として扱う、または複数の変数を直列化可能な
> java.util.Map型等のオブジェクトに格納しserializable型の変数として扱う等の検討を行ってください。

履歴

プロセスインスタンスの実行履歴です。
履歴は全てデータベースに永続化されます。
履歴には、以下の5種類が存在します。

- プロセスインスタンス履歴

  プロセスインスタンス自身の開始、終了に関する履歴です。

- 変数履歴

プロセスインスタンス変数、またはタスク変数操作に関する履歴です。
常に最新の変数情報の履歴のみが永続化されています。

- アクティビティ履歴

  プロセスに含まれるアクティビティに関する履歴です。
  アクティビティに対する到着時間と処理完了までの間隔等が保存されています。

- タスク履歴

  ユーザタスク、またはワークフロー連携タスクに関する履歴です。
  タスクに対する処理者、到着時間と処理完了までの間隔等が保存されています。

- 詳細履歴

  変数履歴と違い、変数に対する操作を含めた全ての履歴です。
  この詳細履歴はデフォルトの設定では無効です。
  この履歴は非常にレコード数が膨大となるため、運用環境等では注意が必要です。

コラム

履歴は、設定ファイルによる履歴レベル設定により内容が変化します。
履歴は”none”, “activity”, “audit”(デフォルト), “full”の設定が可能です。
履歴に”none”、”activity”を設定した場合、一部の画面からその履歴情報が閲覧できなくなるため注意が必要です。

注意

2016 Summer(Nirvana)時点では、audit以外の履歴レベルでは一部の画面が正常に表示できません。
その為、履歴レベルの変更は行わないでください。

関係者

プロセスインスタンス内のユーザタスク等を処理したユーザは、プロセスインスタンスの関係者として記録されます。
プロセスインスタンスの関係者は、ユーザタスクの処理履歴、プロセスインスタンスの履歴等に対して参照が行えます。

タスク

IM-BPM for Accel Platformのタスクについて説明します。

属性

タスクは以下の属性を持ちます。

- id
  - タスクを一意に識別するためのIDです。
- name
  - タスク名です。
    IM-BPM Designerにて設定された名前、または別名が利用されます。
    タスク名はEL式による埋め込みが評価された結果の内容が格納されます。

- description
  - タスクに関する詳細です。
- assignee
  - タスク担当者です、ユーザが一人だけ担当者として設定することが可能です。
- priority
  - 優先度です、数値にて設定します。
    設定はIM-BPM Designer上から行います。
    デフォルトは50です。
- due_date
  - 有効期限です。
    有効期限は表示情報としてのみ利用され、プロセスの実行に影響は与えません。
    プロセスの実行に対して有効期限を制御したい場合には、タイマー境界イベントを利用します。
    画面上では利用していません。
- candidate_users
  - 処理対象ユーザです。
    カンマ区切りで複数のユーザを指定することができます。
- candidate_groups
  - 処理対象グループです。
    カンマ区切りで複数のグループを指定することができます。
    IM-BPM for Accel Platformでは、ロールを利用することができます。

タスクの状態

IM-BPM for Accel Platformでは、タスクの状態を以下の2つの状態として扱います。

- 担当者が未設定の場合

  担当者が未設定のタスクは、グループタスクとして扱われます。
  グループタスクは、処理対象ユーザ、または処理対象グループ(ロール)を持つユーザから参照されます。
  グループタスクは、そのタスクを直接処理することは出来ません。

- 担当者が設定されている場合

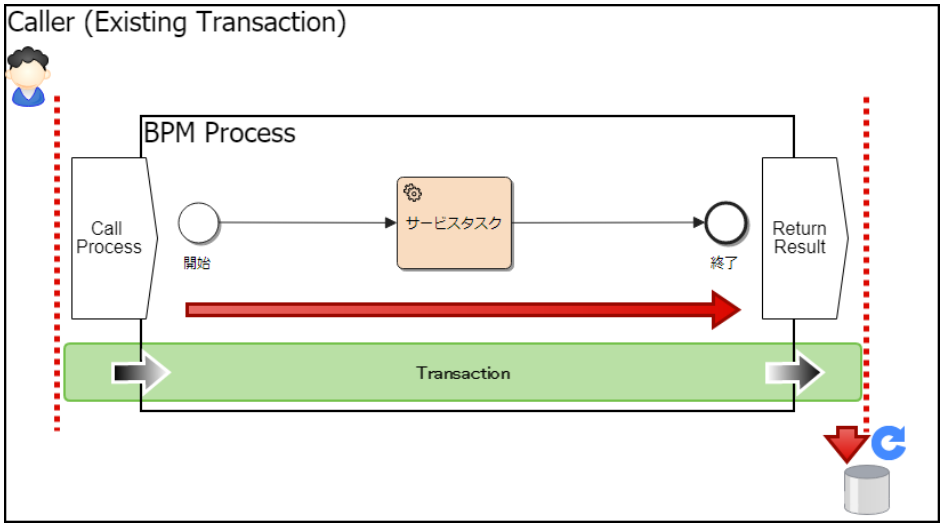
  担当者が設定されているタスクは、個人タスクとして扱われます。
  担当者として設定されているユーザ以外は、そのタスクを操作することは出来ません。
  個人タスクは、タスクを処理することができます。
  個人タスクは担当者を未設定に変更しグループタスクとする事ができます。

プロセスの実行

プロセスの実行に関する仕様について説明します。

トランザクション

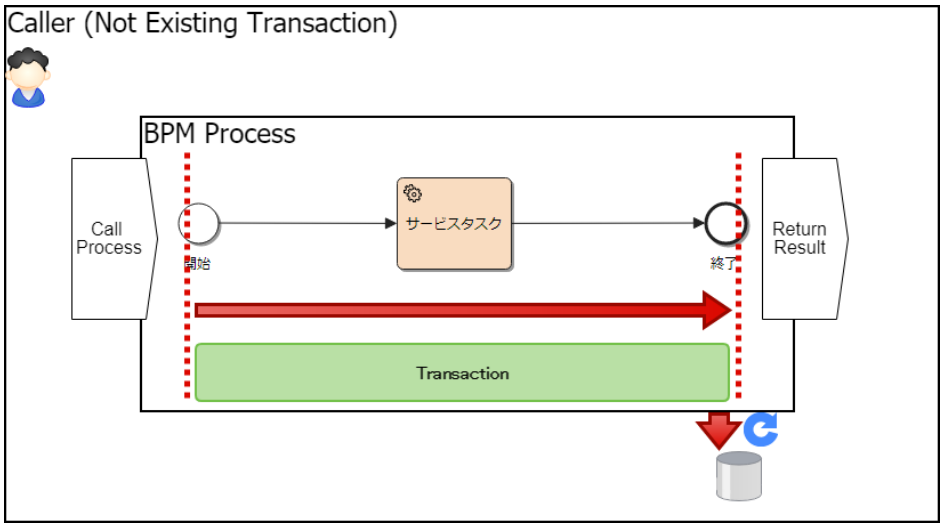
プロセスを処理する際のデータベーストランザクションは、JTA(Java Transaction API)を用いて制御されます。



図：呼び出し元がトランザクションを開始している場合のイメージ

プロセスの処理時に、処理の呼び出し元（上図中ではCaller (Existing Transaction)）が明示的にトランザクションを開始している場合には、そのトランザクションが継続して利用されます。
この場合には、プロセスの処理内でコミットまたはロールバックが実施されず、呼び出し元の処理にトランザクションの制御が移譲されます。例えば、プロセスの処理が正常に実行された場合でも、呼び出し元の処理によりロールバックが実施された場合は、プロセスの処理結果もロールバックされます。

> **注意**
>
> プロセスの処理内で例外が発生した場合
>
> プロセスの処理内で例外が発生した場合には、UserTransaction#setRollbackOnly()が実行されるため、処理の呼び出し元にてコミットを実施することはできません。



図：呼び出し元がトランザクションを開始していない場合のイメージ

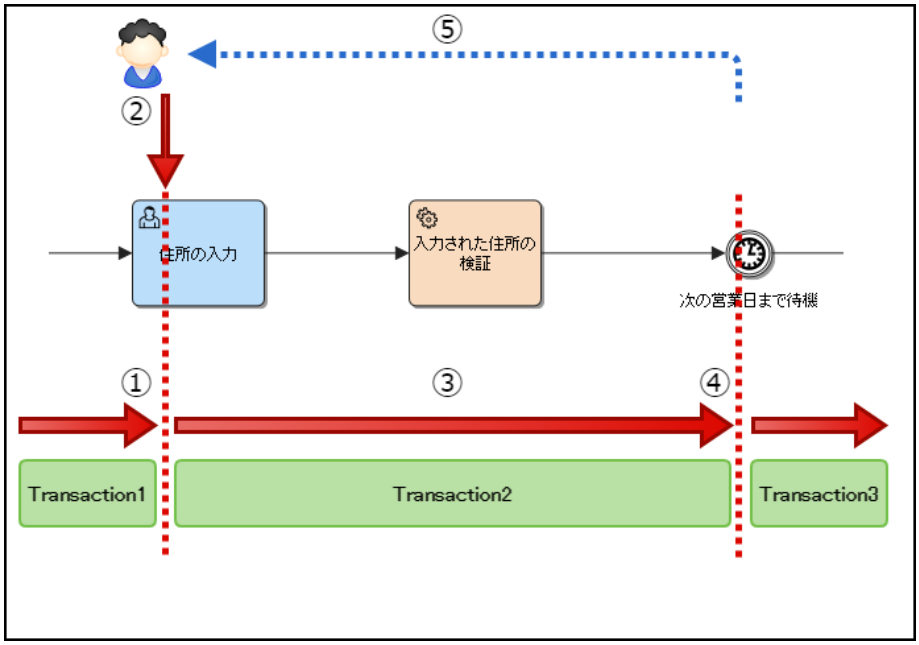
プロセスの処理の呼び出し元にてトランザクションが開始されていない場合には、プロセスの処理内部でトランザクションの開始およびコミットまたはロールバックが実施されます。

トランザクションの範囲

データベーストランザクションの範囲は、プロセス定義の設定内容により変わります。
待機を伴うアクティビティ、非同期実行オプションが設定されたアクティビティがトランザクションの開始／終了の境界地点であり、それらの地点の間のアクティビティの実行が1トランザクションの範囲です。
待機を伴うアクティビティは以下の内容を指します。

- **ユーザタスク**
- **受信タスク**
- **コールアクティビティ**
  ※ただし、呼び出し先のプロセス内に待機を伴うアクティビティが含まれる場合のみ
- **イベントゲートウェイ**
- **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**
- **タイマキャッチイベント**
- **申請タスク**
- **起票タスク**

また、次節で説明する非同期処理（非同期実行オプションが設定されたアクティビティ）では、プロセスを実行しているスレッドとは異なるスレッド（バックグラウンドスレッド）が処理を行います。これはコールアクティビティの呼び出し先のプロセス内に非同期処理がある場合でも同様です。

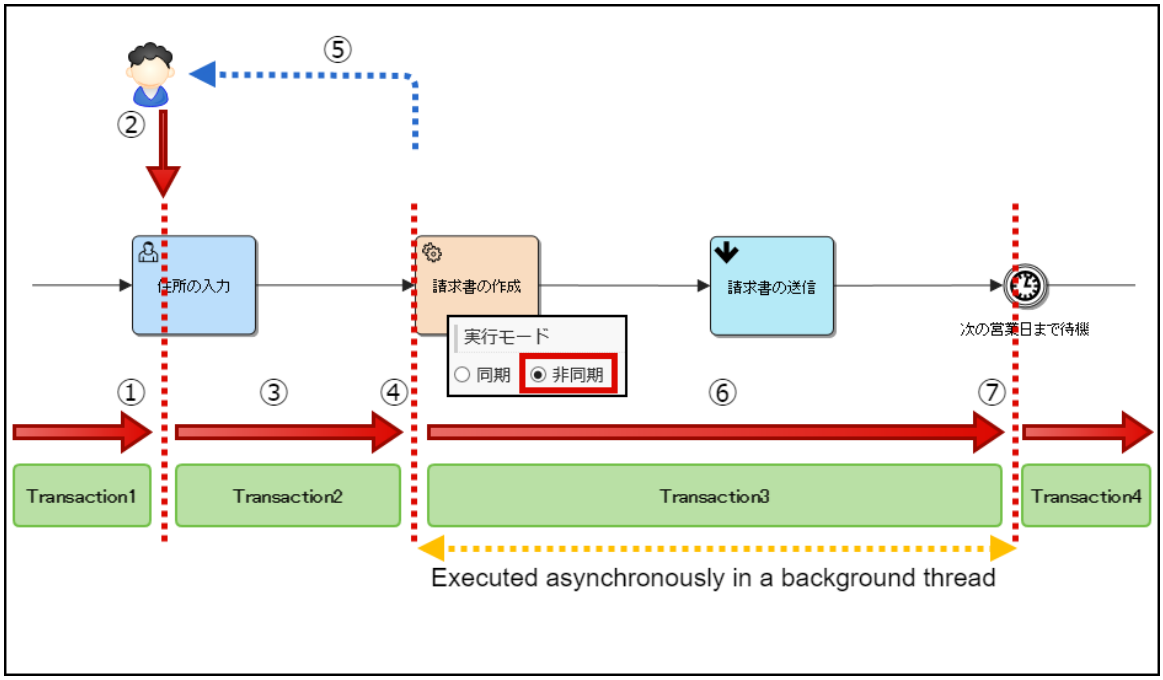


図：トランザクション境界のイメージ

項番	説明
1	待機を伴うアクティビティ（ユーザタスク）に到達したため、トランザクション1が終了
2	ユーザ操作（住所の入力）
3	ユーザ操作（住所の入力）によりアクティビティの待機が解除され、トランザクション2を開始
4	待機を伴うアクティビティ（タイマキャッチイベント）に到達したため、トランザクション2が終了
5	ユーザ操作（住所の入力）の結果として発生したトランザクションが確定し、応答を返却

非同期処理

アクティビティには、非同期実行オプションが設定可能です。
待機を伴うアクティビティに該当しないアクティビティでも、非同期実行オプションが設定されている場合、トランザクションの開始／終了の境界地点として扱われます。
非同期実行は、バックグラウンドスレッド（分散環境時においてはプロセスの実行を行っているAPサーバ上のスレッドとは限りません）で実行されます、その為トランザクションは別途制御されます。
バックグラウンドスレッドの詳細については、後述の「*ジョブ*」の非同期ジョブ実行管理スレッドの項を参照してください。

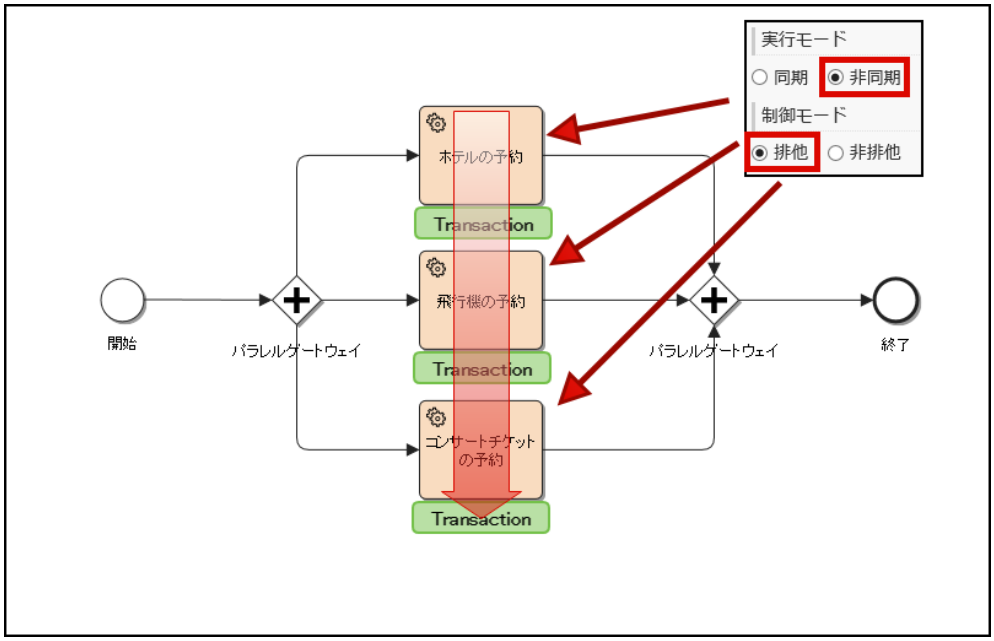


図：非同期実行設定のプロセス実行イメージ

項番	説明
1	待機を伴うアクティビティ（ユーザタスク）に到達したため、トランザクション1が終了
2	ユーザ操作（住所の入力）
3	ユーザ操作（住所の入力）によりアクティビティの待機が解除され、トランザクション2を開始
4	非同期実行オプションが設定されたアクティビティに到達したため、バックグラウンドスレッドに処理を委譲するメッセージを登録し、トランザクション2を終了
5	ユーザ操作（住所の入力）の結果として発生したトランザクションによる処理の結果を返却
6	処理委譲のメッセージを取得したバックグラウンドスレッドがトランザクション3を開始
7	待機を伴うアクティビティ（タイマキャッチイベント）に到達したため、トランザクション3が終了

順次実行制御（排他制御）と同時実行制御（非排他制御）

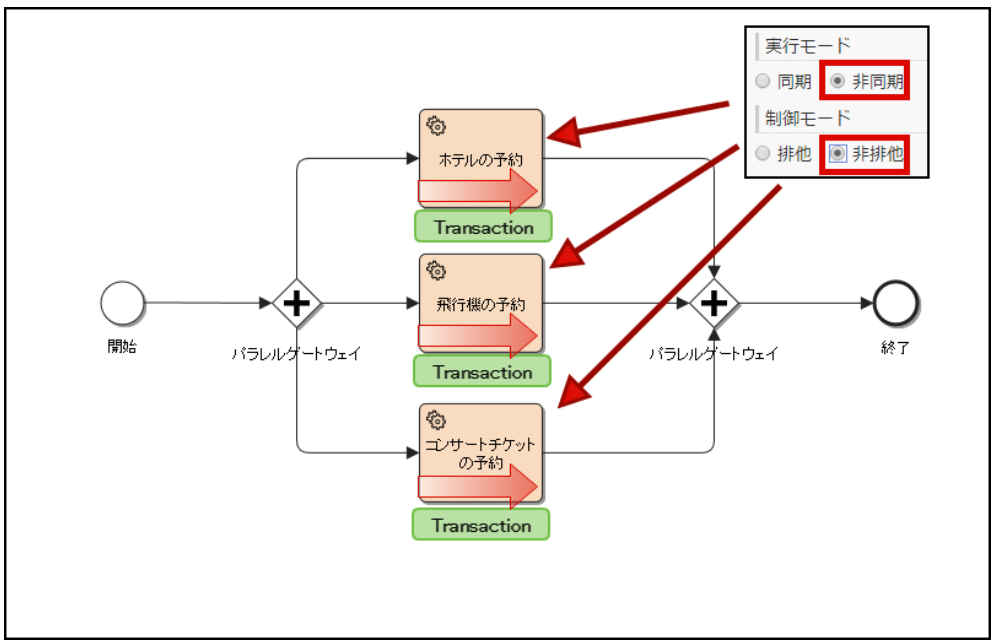
非同期実行オプションを設定した場合、排他制御オプションを設定可能です。
プロセスインスタンス内で並列に配置されているアクティビティに排他制御が設定されている場合、排他制御が設定されたアクティビティは全て同一のバックグラウンドスレッドにて順次かつ別トランザクションで処理され、排他制御が設定された処理同士が同時に実行されることはありません。
下記の例では、「ホテルの予約」、「飛行機の予約」、「コンサートチケットの予約」が非同期（バックグラウンドスレッド）で実行されますが、同一のバックグラウンドスレッドに全ての処理が委譲されるため、「ホテルの予約」、「飛行機の予約」、「コンサートチケットの予約」が同時に実行されることはありません。



図：排他制御の非同期実行オプションを設定したプロセス実行イメージ

> **コラム**
>
> 順次実行制御（排他制御）の範囲
>
> 順次実行制御（排他制御）にて制御を行える範囲は、プロセスインスタンス内のみです。同一のプロセス定義から生成された他のプロセスインスタンスであっても制御の範囲外です。

非同期実行オプションを設定した場合、非排他制御オプションを設定することも可能です。
非排他制御が設定されている場合、処理を委譲するバックグラウンドスレッドを限定しません。そのため、複数のバックグラウンドスレッドが処理を行える状態であった場合、非排他制御が設定されたアクティビティはそれぞれ別のバックグラウンドスレッドにより同時に実行されます。
下記の例では、「ホテルの予約」、「飛行機の予約」、「コンサートチケットの予約」がそれぞれ別のバックグラウンドスレッドにより同時に実行されます。



図：非排他制御の非同期実行オプションを設定したプロセス実行イメージ

注意

非排他制御の設定

非排他制御の非同期実行オプションを設定しプロセスの実行を行うと、複数のバックグラウンドスレッドにより並列に処理を行うことが出来ますが、各トランザクションが複数のテーブルへ同時に更新を行った場合などで、お互いに待ち状態となってしまった場合（デッドロック）、プロセスの実行状態が次に進めず、待ち状態に留まります。
非排他制御の非同期実行オプションを設定する際には、十分な検討を行ってください。

エグゼキューション

エグゼキューションはプロセスインスタンス内の実行単位を示します。
エグゼキューションは、並列ゲートウェイ （ParallelGateway）や、非同期実行オプション、タイマキャッチイベント等により変化します。

コラム

エグゼキューションIDの取得

エグゼキューションIDは、EL式より取得することができます。
${execution.id} と記述する事により取得できます。
取得したエグゼキューションIDはメッセージキャッチイベントや変数の操作時に利用します。
EL式および暗黙オブジェクトについては後述のEL式についてを参照してください。

ジョブ

ジョブはバックグラウンドで処理が行われる単位です。
IM-BPM for Accel Platformでは、プロセスエンジン単位でバックグラウンド実行用のジョブ管理スレッドが起動します。

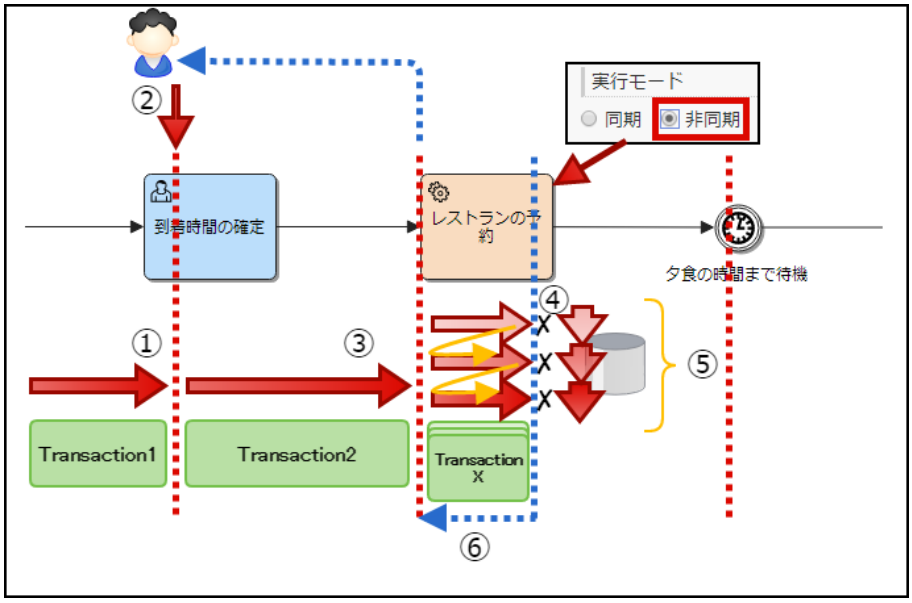
- 非同期ジョブ実行管理スレッド

  非同期ジョブの実行を管理するスレッドです。
  設定において定められた時間間隔でジョブテーブルを監視し、条件の成立する非同期ジョブが存在した場合、そのジョブを別スレッドで実行します。

- タイマ実行管理スレッド

  タイマが設定されたジョブを管理するスレッドです。
  設定において定められた時間間隔でジョブテーブルを監視し、条件の成立するジョブが存在した場合、そのジョブを別スレッドで実行します。

ジョブ実行は、ジョブ取得時にロックされていないジョブを対象とします。
ジョブ実行直前に、そのジョブに対してロックを行います。 （ジョブレコードの更新）
ジョブのロックは、楽観的排他制御を用いて行われます。
また、ジョブ実行中に何らかの例外が発生した場合、ジョブの実行に失敗したという内容をデータベースへコミットします。
その上で、ジョブ管理スレッドは指定された回数ジョブの実行をリトライし、指定回数分リトライに失敗した場合、それ以上リトライを行いません。



図：何らかの例外が発生し、リトライを行った場合のイメージ

項番	説明
1	待機を伴うアクティビティ（ユーザタスク）に到達したため、トランザクション1が終了
2	ユーザ操作（到着時間の確定）
3	ユーザ操作（到着時間の確定）によりアクティビティの待機が解除され、トランザクション2を開始
4	非同期処理（バックグラウンドスレッド）にて、予約可能なレストランを自動で確認し予約処理を行ったが、どのレストランも予約が失敗したため、サービスタスク（レストランの予約）の実行に失敗したという内容でジョブレコードを更新
5	一定時間毎に指定された回数（この例では2回）に達するまでサービスタスク（レストランの予約）の実行をリトライ
6	指定された回数リトライを行っても、サービスタスク（レストランの予約）の実行が成功しなかったため、サービスタスク（レストランの予約）の再実行のトリガを待機

> **コラム**
>
> ジョブの実行環境
>
> 分散環境時においては、ジョブはどのサーバ上で動作するか保証はされていません。

> **コラム**
>
> ジョブのトランザクション制御
>
> ジョブ実行時のスレッドは、JavaEEコンテキストが存在するスレッドにて動作します。その為、JTA等を利用したデータベーストランザクションの管理が可能です。

マイグレーション

プロセスマイグレーションについて説明します。

概要

マイグレーション機能は、実行中のプロセスインスタンスを別のプロセス定義のプロセスインスタンスとして移行する為の機能を提供します。

マイグレーションを行う際の条件

マイグレーション先として指定可能なアクティビティ

マイグレーション先として指定可能なアクティビティは以下の通りです。

- 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
- IM-LogicDesignerタスク
- 申請タスク
- 起票タスク
- マニュアルタスク
- レシーブタスク
- CallActivity
- ビジネスルールタスク
- サブプロセス
- イベントゲートウェイ
- タイマーキャッチイベント
- 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
- シグナルスローイベント
- Noneスローイベント

変数

マイグレーション実行前、とマイグレーション実行後におけるプロセスインスタンス変数はそのまま受け継がれます。
移行先のプロセス定義実行に必要な変数が不足している場合等は、直接変数を格納しているデータベースを修正する等の対応が必要となります。

また、プロセス定義の変数は登録・更新が行われません。

リスナ

マイグレーション実行時における、リスナーの動作は以下の通りです。

- マイグレーション元
  - アクティビティのリスナは動作しません。
  - ユーザタスクのリスナは動作しません。
  - プロセス定義の終了リスナは動作しません。
- マイグレーション先
  - プロセス定義の開始リスナは動作しません。
  - アクティビティの開始リスナは動作しません。
  - ユーザタスクの開始リスナは動作します。

移行不可能なアクティビティ先

マイグレーションにおいて、移行先として利用できないアクティビティは以下の通りです。

- イベントサブプロセスへの移行は行えません。
- イベントゲートウェイの遷移先として指定されているイベントに対しては移行できません。

非同期タスク

マイグレーション先が非同期タスクの場合でも、非同期として処理は行われません。
これは移行のための履歴を作成するため同期処理として実行する必要があるためです。

インポート/エクスポート

IM-BPM for Accel Platformのインポート/エクスポート機能について説明します。

インポート/エクスポートで扱う情報

インポート/エクスポートでは以下の情報を扱います。

- デプロイメント
  - デプロイしたファイルです。デプロイファイル内に含まれる各資材を個別に扱うことはできません。
- 一覧表示設定
  - 「テナント」スコープで保存した各画面の一覧表示設定情報です。「ユーザ」スコープで保存された情報は対象外です。
- プロセスデザイナ
  - プロセスデザイナで管理されるプロジェクトです。プロジェクトに含まれる各資材を個別に扱うことは出来ません。

ファイルフォーマット

エクスポート機能では、以下のファイルをアーカイブ（zip）して出力します。

- デプロイメント
  - 「deployment」フォルダに格納されます。
  - デプロイされた形式のままエクスポートファイルに含まれます。
- 一覧表示設定
  - 「listTableSetting」フォルダに格納されます。
  - ファイル名: im-bpm-listtable-setting.xml
- プロセスデザイナ
  - 「wd_project」フォルダに格納されます。
  - プロセスデザイナで管理されるプロジェクト情報がエクスポートされます。
  - プロジェクトに含まれるプロセス定義およびリソースは、有効な情報のみエクスポートされます。履歴情報はエクスポートされません。

インポート/エクスポート時の動作

IM-BPM for Accel Platformのインポート/エクスポート機能の動作仕様は以下の通りです。

エクスポート

エクスポート機能は、「*ファイルフォーマット*」に記載のフォーマットでIM-BPM for Accel Platformに関するデータを出力します。

インポート

インポート機能は、「*ファイルフォーマット*」に記載のフォーマットでアーカイブされたZIPファイルをもとに、アーカイブファイルに含まれるすべてのIM-BPM for Accel Platformに関するデータを取り込みます。

インポートが失敗した場合の動作

インポートが失敗した場合は、それまで行われたすべてのインポート処理がロールバックされます。

インポート先に同一プロセス定義キーのプロセス定義が存在する場合の動作

デプロイメントのインポートにおいて、既に同一のプロセス定義キーのプロセス定義がインポート先に存在する場合、インポート先のプロセス定義のバージョンが上がります。
インポートしたプロセス定義を最新バージョンとします。

インポート先に同一画面、かつ同一設定キーの一覧表示設定が存在する場合の動作

一覧表示設定のインポートにおいて、既に同一画面、かつ同一設定キーの一覧表示設定がインポート先に存在する場合、インポート先の一覧表示設定を上書きで更新します。

インポート先に同一プロジェクトIDのプロジェクトが存在する場合の動作

プロジェクトのインポートにおいて、既に同一プロジェクトIDのプロジェクトがインポート先に存在する場合、プロジェクト名などを上書きします。
プロジェクトに含まれるプロセス定義およびリソースは、同一の資材（システム内部で採番された番号が同一のもの）を除いて、上書きで更新します。

画面連携

画面連携の仕様について説明します。

項目

画面連携について

IM-BPM for Accel Platformでは、ユーザタスクの処理画面を自由に設定することが可能です。
以下に代表的な連携方式を示します。

- IM-FormaDesignerにより作成された画面
- TERASOLUNA Frameworkにより作成された画面
- スクリプト開発モデルにより作成された画面
- フレームワークを利用せず、HTMLのみで構成された画面

IM-BPM for Accel Platformでは、IM-FormaDesigner連携と、その他スクラッチ画面連携の2種類の連携方法が用意されています。

フォームキー

IM-BPM Designerを利用し、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。

- forma:{APPLICATION_ID}

  IM-FormaDesignerと連携するための方式です。
  “forma:”プレフィクスと共に、IM-FormaDesignerにより作成されたアプリケーションIDを設定してください。

- forward:(PATH)

  スクラッチ画面と連携するための方式です。
  “forward:”プレフィクスと共に、遷移先の画面パスを指定してください。
  画面パスは必ず “/” から始まる必要があります。
  例: https://example.org/imart/foo/bar 画面と連携する場合
  　forward:/foo/bar

- redirect:(PATH)

  スクラッチ画面と連携するための方式です。
  “redirect:”プレフィクスと共に、遷移先のURLを指定してください。
  例: https://example.org/foo/bar 画面と連携する場合
  　redirect:https://example.org/foo/bar

IM-FormaDesigner連携

IM-FormaDesignerにより作成された画面と連携を行うためには以下の条件が必要です。

- アプリケーション種別が標準であること
- テーブル設定が行われていること
- 権限設定が行われていること

  IM-BPM for Accel Platformと連携する場合権限として “登録” と “参照” が必要です。

前処理と後処理

IM-FormaDesigner連携機能を利用する場合、必ずフォームに対するユーザプログラムとして前処理、後処理が必要です。

- 前処理: jp.co.intra_mart.activiti.extension.forma.BPMPreProcessExecutor
- 後処理: jp.co.intra_mart.activiti.extension.forma.BPMPostProcessExecutor

前処理の実行処理種別 “登録”、”編集”、”参照”と後処理の実行処理種別 “登録”、”更新”、”削除”、”一時保存”にユーザプログラムを登録する必要があります。
この前処理、後処理は IM-BPM for Accel Platformから呼び出されていない場合、何も処理を行いません。
前処理、後処理はプロセス定義がデプロイされた際に、自動的に登録が行われます。

> **注意**
>
> フォームキーにEL式が含まれている場合、前処理、後処理はデプロイ時に登録されません。
> 例: forma:${dynamicFormKey}
> この場合には、事前に前処理、後処理を手動で登録する必要があります。

前処理では、以下の検証が行われます。

- プロセス開始の場合、処理対象プロセスに設定された権限（ユーザ、ロール）を保持していない場合例外を通知する
- タスク処理の場合、処理対象ユーザタスクに設定された権限（ユーザ、ロール）を保持していない場合例外を通知する
- タスク実行履歴からの参照の場合、処理対象ユーザタスクに設定された権限（ユーザ、ロール）を保持していない場合例外を通知する

前処理では以下の処理が行われます。

- プロセスインスタンスに格納されている変数情報を取得し、フォームの初期値として設定する。

  フォームの初期値として利用されるプロセスインスタンス変数は、フォームに含まれるアイテムIDと同名の変数が利用されます。

後処理では、以下の検証が行われます。

- プロセス開始の場合、処理対象プロセスに設定された権限（ユーザ、ロール）を保持していない場合例外を通知する
- タスク開始の場合、処理対象ユーザタスクに設定された権限（ユーザ、ロール）を保持していない場合例外を通知する

後処理では、以下の処理が行われます。

- プロセス開始の場合、{アプリケーションID}${アプリケーション番号}${FormaID}を業務キーとして設定する。
- プロセス開始の場合、フォームに入力された内容をプロセスインスタンス変数として業務プロセスの開始を行う。
- タスク処理の場合、フォームに入力された内容をプロセスインスタンス変数としてユーザタスクを進める。

> **コラム**
>
> タスク処理の場合以下のデータは変数として保存されません。
>
> - im_frで始まるデータ
> - im_で始まるデータ
> - __が前後に付与されているデータ
>
> 上記のデータはユーザタスク固有のローカル変数として保存されます。

スクラッチ画面連携

パラメータ

スクラッチ画面連携では、プロセスの実行に必要なパラメータがリクエストパラメータとして受け渡されます。
画面表示時のパターンと、リクエストパラメータについて説明します。

- プロセス開始時における画面初期表示パラメータ

  callbackPath: 戻り先のURL
  processDefinitionId: プロセス定義ID

- タスク処理時における画面初期表示パラメータ

  callbackPath: 戻り先のURL
  taskId: タスクID

- プロセス開始時の情報を履歴から参照する場合の画面初期表示パラメータ

  callbackPath: 戻り先のURL
  historicProcessInstanceId: 実行済プロセスインスタンスID

- タスク処理後の情報を履歴から参照する場合の画面初期表示パラメータ

  callbackPath: 戻り先のURL
  historicTaskId: タスクID

それぞれの画面において処理を実行した後、戻り先のURL (callbackPath)パラメータにリダイレクトを行ってください。
タスクの一括処理を行った場合、callbackPathには、一括処理用のtransactionIdパラメータが埋め込まれており、次の処理対象画面へ遷移するURLです。

検証

スクラッチ画面連携を行う場合、それぞれの画面において業務プロセスおよびタスクに対する権限の検証を行ってください。
検証処理を実装しなかった場合、第三者から画面情報が閲覧できる状態になる可能性があります。

- プロセス開始画面の場合

  プロセス定義に設定された権限の検証を行う必要があります。
  プロセス開始時の処理画面の場合、RepositoryService#getIdentityLinksForProcessDefinition メソッドによりプロセス定義に設定された権限情報の取得ができます。

  履歴画面からの参照の場合、HistoryService#getHistoricIdentityLinksForProcessInstance メソッドによりプロセスインスタンスに紐づく関係者の取得ができます。

- タスク画面の場合

  タスクに設定された権限の検証を行う必要があります。
  タスク処理時のタスク処理画面の場合、TaskService#getIdentityLinksForTask メソッドによりタスクに設定された権限情報の取得ができます。

  履歴画面からの参照の場合、HistoryService#getHistoricIdentityLinksForTask メソッドによりタスクに紐づく関係者の取得ができます。

ユーザタスク処理者の取得

ユーザタスク処理後、そのユーザタスクを実際に処理したユーザコードをEL式より取得することができます。
暗黙のオブジェクトとして、“im_operation_users” オブジェクトが用意されています。
この “im_operation_users” オブジェクトは、Map型となり、キーとしてユーザタスクのID、値としてユーザタスクを処理したユーザコードを持ちます。
例: ユーザタスク(id: foo_user_task) を処理したユーザコードをEL式にて取得する場合
    ${im_operation_users.foo_user_task}

プロセス定義によっては、同一のユーザタスクを複数回呼び出す場合が存在します。
同一のユーザタスクが複数回呼びだされた場合には、最後にそのユーザタスクを処理したユーザコードが格納されます。

コラム
ユーザタスクに指定するIDにハイフン等EL式で直接利用できない文字が含まれている場合には ${im_operation_users['foo-bar-baz']} 形式で指定することにより値の取得ができます。

注意
システム変数の格納方式の設定

システム変数の格納方式の設定(is-system-variable-save-as-object)がtrueの場合は、暗黙のオブジェクト “im_bpm_system_variables” オブジェクトの要素として”im_operation_users” オブジェクトが格納されます。

例: ユーザタスク(id: foo_user_task) を処理したユーザコードをEL式にて取得する場合
${im_bpm_system_variables.im_operation_users.foo_user_task}

システム変数の格納方式の設定の詳細については、「 IM-BPM 設定ファイルリファレンス 」-「 システム変数の格納方式の設定 」を参照してください。

IM-LogicDesigner連携

IM-LogicDesigner連携の仕様について説明します。

項目

IM-LogicDesignerタスク

「IM-LogicDesignerタスク」を利用することで、プロセス内からIM-LogicDesignerで作成した任意のロジックフローを呼び出すことが可能です。

ロジックフローへの入力データの構築

IM-LogicDesignerタスクのプロパティ「入力データ」を設定することで、ロジックフローに対して変数のデータなどを連携することが可能です。

ロジックフローの出力データの返却

IM-LogicDesignerタスクのプロパティ「結果変数を格納する」を設定することで、ロジックフローの実行結果をプロセスインスタンス変数に格納することが可能です。格納される実行結果は、ロジックフロー実行結果オブジェクトです。

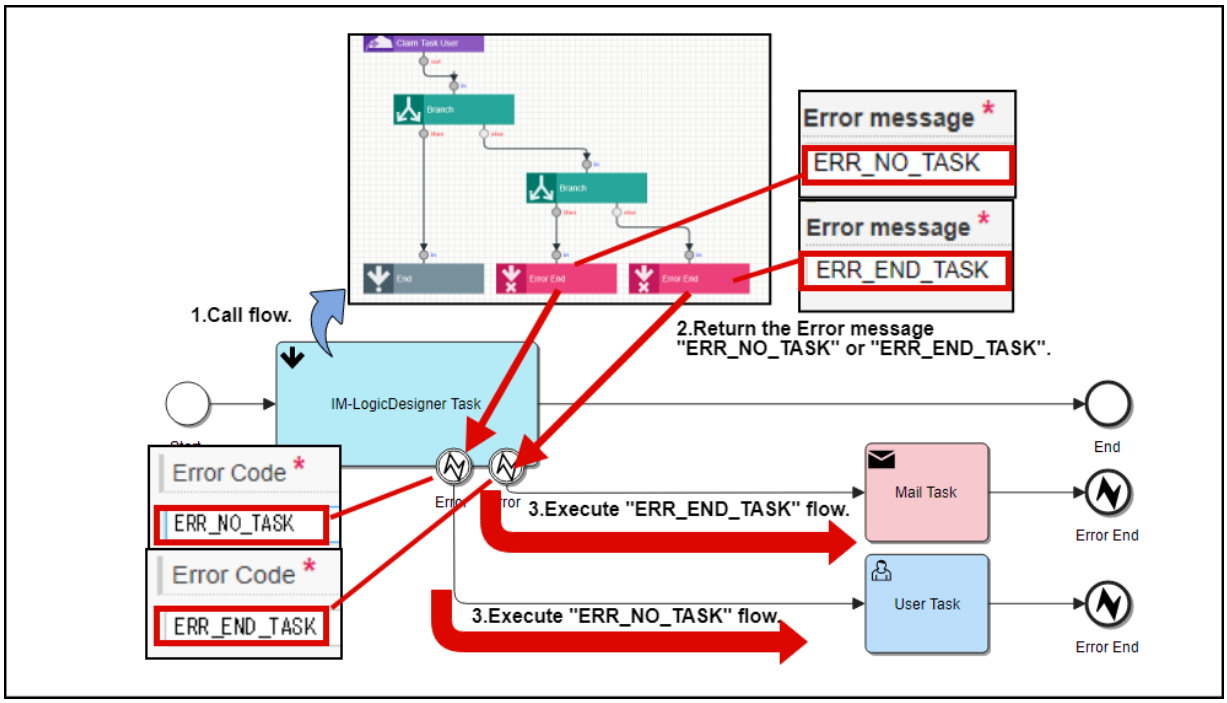
ロジックフロー実行結果オブジェクトは、IM-LogicDesignerタスクのプロパティ「結果変数名」に設定した名前のプロセスインスタンス変数に格納されます。プロパティ「結果変数名」を設定しなかった場合は、ロジックフロー実行結果オブジェクトの第1階層のプロパティがそれぞれプロセスインスタンス変数として格納されます。

エラーハンドリング

ロジックフローでは、フロー中にエラー終了エレメントを配置することが可能です。エラー終了エレメントは、エラーメッセージプロパティに任意のエラーメッセージを設定することができます。

プロセス側ではエラーキャッチイベントを配置することで、IM-LogicDesignerのロジックフローからスローされたエラーをキャッチすることが可能です。スローされたエラーは、そのエラーメッセージに合致するエラーコードが設定されたエラーキャッチイベントにてキャッチされます。このため、ロジックフロー内で複数のエラー終了エレメントが配置されて

いた場合に、プロセス側ではそれぞれのエラーを判別し、処理を進めることが可能です。



図：エラーハンドリングのイメージ

IM-LogicDesignerリスナ

エグゼキューションやユーザタスクのイベントリスナとして、IM-LogicDesignerで作成したロジックフローを設定することが可能です。IM-LogicDesignerリスナを設定することで、イベントの発生時に設定したロジックフローが呼び出されます。

- 本機能は、プロセスデザイナを利用してプロセス定義を作成する場合のみ設定可能です。
- IM-BPM Designerを利用してプロセス定義を作成する場合は、本機能を設定することはできません。

ロジックフローへの入力データの構築

入力データ

IM-LogicDesignerリスナのプロパティ「入力データ」を設定することで、ロジックフローに対して変数のデータなどを連携することが可能です。

暗黙オブジェクト

ロジックフローの入力値として、暗黙オブジェクトを受け取ることが可能です。暗黙オブジェクトの詳細については、「*暗黙オブジェクト*」を参照してください。

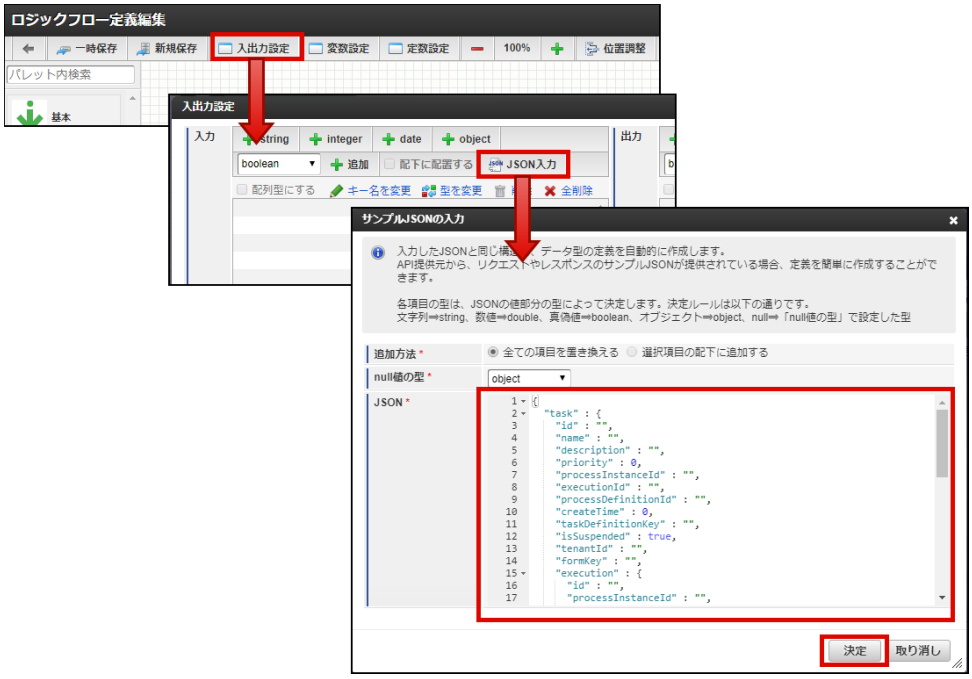
下記のjsonをロジックフローの「入出力設定」の「入力」に設定してください。

- 実行リスナとしてロジックフローを使用する場合

```
{
 "execution" : {
   "id" : "",
   "processInstanceId" : "",
   "eventName" : "",
   "businessKey" : "",
   "processDefinitionId" : "",
   "parentId" : "",
   "superExecutionId" : "",
   "currentActivityId" : "",
   "currentActivityName" : "",
   "tenantId" : "",
   "variablesLocal" : {},
   "variables" : {}
 }
}
```

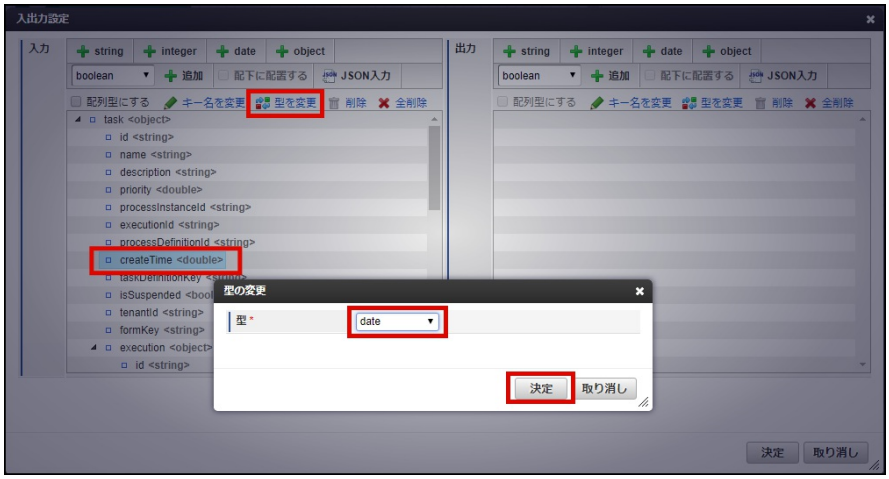
- タスクリスナとしてロジックフローを使用する場合

```
{
 "task" : {
  "id" : "",
  "name" : "",
  "description" : "",
  "priority" : 0,
  "processInstanceId" : "",
  "executionId" : "",
  "processDefinitionId" : "",
  "createTime" : 0,
  "taskDefinitionKey" : "",
  "isSuspended" : true,
  "tenantId" : "",
  "formKey" : "",
  "execution" : {
   "id" : "",
   "processInstanceId" : "",
   "eventName" : "",
   "businessKey" : "",
   "processDefinitionId" : "",
   "parentId" : "",
   "superExecutionId" : "",
   "currentActivityId" : "",
   "currentActivityName" : "",
   "tenantId" : "",
   "variablesLocal" : {},
   "variables" : {}
  },
  "eventName" : "",
  "delegationState" : "",
  "owner" : "",
  "assignee" : "",
  "dueDate" : 0,
  "category" : "",
  "candidates" : [{
   "type": "",
   "userId": "",
   "groupId": "",
   "taskId": "",
   "processDefinitionId": "",
   "processInstanceId": ""
  }],
  "variablesLocal" : {},
  "variables" : {}
 }
}
```



図：ロジックフローの「入出力設定」のイメージ

createTime と dueDate は「型を変換」から型を date に変換してください。



図：「型を変換」のイメージ

ロジックフローの出力データの返却

IM-LogicDesignerリスナのプロパティ「結果変数を格納する」を設定することで、ロジックフローの実行結果をプロセスインスタンス変数に格納することが可能です。格納される実行結果は、ロジックフロー実行結果オブジェクトです。

ロジックフロー実行結果オブジェクトは、IM-LogicDesignerリスナのプロパティ「結果変数名」に設定した名前のプロセスインスタンス変数に格納されます。プロパティ「結果変数名」を設定しなかった場合は、ロジックフロー実行結果オブジェクトの第1階層のプロパティがそれぞれプロセスインスタンス変数として格納されます。

エラーハンドリング

IM-LogicDesignerリスナにおいては、エラーキャッチイベントでエラーをハンドリングすることができません。
ロジックフローがエラー終了エレメントで終了した場合、例外が発生し、トランザクションが全てロールバックされます。

OpenRules連携

OpenRules連携の仕様について説明します。

対応フォーマット

OpenRules連携では、xls形式のファイルのみ対応しています。
xlsx形式及び、DMN形式のフォーマットには対応していません。

リソース

IM-BPM for Accel Platform におけるOpenRules連携では、OpenRulesにおける設定ファイル群 (openrules.config) もデプロイする必要があります。
BPMプロジェクト内に設定ファイル群 (openrules.config) も含めて配置してください。
OpenRules連携では、ビジネスルールタスク実行時に、クラスローダによりWEB-INF/work/_jars配下に展開されたルールファイル、設定ファイル群を参照し実行します。

パラメータ

OpenRules連携では、ビジネスルールタスクのパラメータとして、以下の項目を設定する必要があります。

- ルールファイル

  Excel(xls)により定義されたルールファイル

- デシジョン (Decision)

  実行するデシジョン

- 入力コンセプト (Input Concepts)

  入力パラメータとして利用するコンセプト
  コンセプトに含まれる各パラメータが、プロセスインスタンスの持つ変数名と一致している場合、その値が利用されます。

- 出力コンセプト (Output Concepts)

  出力として指定されているコンセプト名
  出力コンセプトに含まれるデータをプロセスインスタンスに返却します。

- 結果変数名

  結果変数名が未指定の場合は、出力コンセプトの内容をそのままプロセスインスタンスの変数としてマージします。
  結果変数名が指定されている場合、出力コンセプトの内容をプロセスインスタンスの変数に結果変数名を利用してセットします。

IM-Workflow/Forma/IM-BIS連携

IM-Workflow/Forma/IM-BIS連携の仕様について説明します。

項目

IM-Workflow/Forma/IM-BIS連携について

IM-BPM for Accel Platformでは、ワークフロー連携として以下の連携機能が存在します。

- IM-Workflow連携

  コンテンツとしてIM-FormaDesignerを利用していないワークフローを指します。

- IM-FormaDesigner IM-Workflow連携機能との連携

  コンテンツとしてIM-FormaDesignerと連携しているワークフローを指します。

- IM-BIS連携

  それぞれの連携では、起票、および申請の連携が可能です。

起票による連携

起票による連携では、起票タスク (DraftTask)による連携が行われます。
起票タスクの処理の流れは以下の通りです。

- 起票用パラメータを作成する

  プロセスインスタンス側から受け渡されたフローID、ユーザデータID、案件名、案件番号、 起票者コード、優先度等を元に起票用パラメータを作成します。

- 前処理プログラムの指定がある場合には、前処理を実行します。

  前処理は、縦配置、横配置等複雑な起票パラメータが必要となる場合に利用します。
  それぞれ以下のインタフェースを実装する必要があります。

  - IM-Workflow: jp.co.intra_mart.activiti.engine.delegate.ImWorkflowDraftPreprocess
  - IM-FormaDesigner + IM-Workflow: jp.co.intra_mart.activiti.engine.delegate.ImFormaDraftPreprocess
  - IM-BIS: jp.co.intra_mart.activiti.engine.delegate.ImBisDraftPreprocess
- ワークフローの起票
- タスク情報の登録

  ワークフロー連携における状態管理として、タスク情報の登録を行います。
  タスク情報には、タスクローカルな変数として以下の内容が格納されます。

  - systemMatterId: システム案件ID
  - matterNumber: 案件番号
  - userDataId: ユーザデータID
  - matterResultName: 案件結果変数名
  - manipulateMatter: タスクキャンセル時の案件操作方法
  - flowName_{locale}: ロケール分のフロー名

> **注意**
>
> システム変数の格納方式の設定
>
> システム変数の格納方式の設定(is-system-variable-save-as-object)がtrueの場合は、タスクローカルな変数 “im_bpm_system_variables” オブジェクトの要素として”systemMatterId” , “matterNumber” , “userDataId” , “matterResultName”, “manipulateMatter”, “flowName_{locale}”が格納されます。
>
> システム変数の格納方式の設定の詳細については、「 IM-BPM 設定ファイルリファレンス 」-「 システム変数の格納方式の設定 」を参照してください。

- IM-Workflow側の案件プロパティに連携情報を保存します。

  ワークフロー案件終了時に、プロセスインスタンスを処理するため、案件プロパティに以下の内容を登録します。
  以下の案件プロパティは外部から操作を行わないでください。

  - im_bpm_task_id: タスクID

申請による連携

申請による連携では、申請タスク (ApplyTask)による連携が行われます。
申請タスクの処理の流れは以下の通りです。

- 申請用パラメータを作成する

  プロセスインスタンス側から受け渡された以下のパラメータを元に、申請用パラメータを作成します。

  - フローID
  - ユーザデータID
  - 案件番号
  - 案件名
  - 申請基準日
  - 申請実行者コード
  - 申請権限者コード
  - 権限者会社コード
  - 権限者組織セットコード
  - 権限者組織コード
  - 優先度
  - 処理コメント
- 前処理プログラムの指定がある場合には、前処理を実行します。

  前処理は、縦配置、横配置等複雑な起票パラメータが必要となる場合に利用します。
  それぞれ以下のインタフェースを実装する必要があります。

  - IM-Workflow: jp.co.intra_mart.activiti.engine.delegate.ImWorkflowApplyPreprocess
  - IM-FormaDesigner + IM-Workflow: jp.co.intra_mart.activiti.engine.delegate.ImFormaApplyPreprocess
  - IM-BIS: jp.co.intra_mart.activiti.engine.delegate.ImBisApplyPreprocess
- ワークフローの申請
- タスク情報の登録

  ワークフロー連携における状態管理として、タスク情報の登録を行います。
  タスク情報には、タスクローカルな変数として以下の内容が格納されます。

- systemMatterId: システム案件ID
- matterNumber: 案件番号
- userDataId: ユーザデータID
- matterResultName: 案件結果変数名
- manipulateMatter: タスクキャンセル時の案件操作方法
- flowName_{locale}: ロケール分のフロー名

> **注意**
>
> システム変数の格納方式の設定
>
> システム変数の格納方式の設定(is-system-variable-save-as-object)がtrueの場合は、タスクローカルな変数 “im_bpm_system_variables” オブジェクトの要素として”systemMatterId” , “matterNumber” , “userDataId” , “matterResultName”, “manipulateMatter”, “flowName_{locale}”が格納されます。
>
> システム変数の格納方式の設定の詳細については、「 IM-BPM 設定ファイルリファレンス 」-「 システム変数の格納方式の設定 」を参照してください。

- IM-Workflow側の案件プロパティに連携情報を保存します。

  ワークフロー案件終了時に、プロセスインスタンスを処理するため、案件プロパティに以下の内容を登録します。
  以下の案件プロパティは外部から操作を行わないでください。

  - im_bpm_task_id: タスクID

ワークフロー終了時の連携

ワークフローが終了した場合には、IM-BPM側のプロセスが再開されます。
ワークフローが終了する際のパターンは以下の通りです。

- 承認終了: approveend
- 最終承認: mattercomplete
- 否認: deny
- 取りやめ: discontinue
- 案件操作: matterhandle
- 管理者による案件削除: activedelete
- 削除: delete

ワークフロー終了時には、ワークフローの情報がプロセスインスタンスの変数として格納されます。
結果は以下の項目として参照できます。

- 正常に完了した場合

  承認、否認、案件操作の場合を指します。

  - locale: ロケール
  - processDate: 処理日時
  - systemMatterId: システム案件ID
  - userDataId: ユーザデータID
  - contentsId: コンテンツID
  - contentsVersionId: コンテンツバージョンID
  - routeId: ルートID
  - routeVersionId: ルートバージョンID

    - flowId: フローID
    - flowVersionId: フローバージョンID
    - actFlag: 承認フラグ
    - lastAuthUserCd: 最終承認ユーザコード
    - lastExecUserCd: 最終実行者コード
    - lastNodeType: 最終ノード種別
    - lastProcessNodeId: 最終処理ノードID
    - lastNodeName: 最終処理ノード名
    - lastResultStatus: 案件結果
- 案件削除が行われた場合
    - locale: ロケール
    - systemMatterId: システム案件ID
    - userDataId: ユーザデータID
    - lastResultStatus: 案件結果 （案件削除のため”activedelete”）
- BPM側から終了した場合

    プロセスマイグレーション等が行われた場合です。

    - systemMatterId: システム案件ID
    - userDataId: ユーザデータID
    - lastResultStatus: 案件結果 （結果が存在しないため null）

起票タスク(DraftTask)、申請タスク(ApplyTask)の属性 結果変数名が指定されている場合には案件プロパティの値を結果変数名を利用してプロセスインスタンスの変数として格納します。
結果変数名が未指定の場合には、プロセスインスタンス変数に案件プロパティの内容がマージされます。

IM-FormaDesignerまたはIM-BISを利用している場合には、Formaデータの内容も変数にマージされます。
案件プロパティとFormaデータ共に同一のキー名が存在した場合にはFormaデータが優先されます。

案件プロパティにおける数値型はLong型、または小数を持つ数値の場合Double型としてプロセスインスタンス変数に格納されます。

トランザクション

ワークフロー連携機能は全て IM-BPM for Accel Platformのトランザクション管理内で実行されます。
IM-BPM for Accel Platformでは、IM-Workflowに intra-mart Accel Platform 2016 Summer(Nirvana) より導入されたフロー情報の保存先としてストレージを使用しないモードにおける実行のみサポートします。

Elasticsearch連携機能

Elasticsearchとの連携について説明します。

項目

Elasticsearch連携について

Elasticsearchは、Elastic社が提供するオープンソース全文検索エンジンです。
https://www.elastic.co/products/elasticsearch

Elasticsearchは、データをJSON形式で格納することができます。
Elasticsearch連携では、プロセスインスタンスに関する情報をElasticsearchに保存する為の機能です。
保存されたデータは、Elastic社が提供するKibanaを利用することにより可視化する事ができます。
Elasticsearch連携ではプロセスインスタンスに含まれる変数情報も含めて全てElasticsearchに保存するため
業務プロセスで利用されている任意のデータを可視化することができます。

Elasticsearch連携方式

Elasticsearchでは、TCP TransportとHTTPによる連携方式があります。
Elasticsearch連携は全てHTTPによるデータの登録を行います。

連携のタイミングは、プロセスの実行が完了する直前です。
Elasticsearchに対し、httpによるバルクリクエストを行います。
連携に失敗した場合、プロセスインスタンスの処理は継続します。

登録されるデータ

登録されるデータ例:

```
{
 "_index": "im_bpm-default-20160725",
 "_type": "im_bpm",
 "_id": "AVYfNl7gDFByk9GXT5rc",
 "_score": null,
 "_source": {
  "source_activity_id": "servicetask1",
  "sequence_flow_id": "flow2",
  "type_activiti": "SEQUENCEFLOW_TAKEN",
  "target_activity_type": "endEvent",
  "source_activity_type": "serviceTask",
  "target_activity_id": "endevent1",
  "user_cd": "master",
  "process_definition_id": "service:1:8e28em7kuuyr2m7",
  "source_activity_name": "Service Task",
  "execution_id": "8e2ao18qwuztbm7",
  "process_instance_id": "8e2ao18qwuztbm7",
  "target_activity_behavior_class":
"jp.co.intra_mart.activiti.engine.impl.bpmn.behavior.NoneEndEventActivityBehavior",
  "source_activity_behavior_class": "jp.co.intra_mart.activiti.engine.impl.bpmn.helper.ClassDelegate",
  "target_activity_name": "End",
  "time": "2016-07-25T08:21:12+0900",
  "variables": {
   "foo": "FOO",
   "bar": "BAR"
  }
 },
 "fields": {
  "time": [
   1469402472000
  ]
 },
 "sort": [
  1469402472000
 ]
}
```


連携イベント

Elasticsearchに登録されるプロセス内のイベントデータ配下の種類が存在します。

- アクティビティの開始: activity_started
- アクティビティの完了: activity_completed
- アクティビティのキャンセル: activity_cancelled
- アクティビティに対するシグナルの受信: activity_signaled
- アクティビティに対するメッセージの受信: activity_messge_received
- シーケンスフローによる遷移: sequenceflow_taken
- ユーザタスクの作成: task_creted
- ユーザタスクに対する担当者の割当: task_assigned
- ユーザタスクの完了: taks_completed
- プロセスインスタンスの開始: process_started
- プロセスインスタンスの終了: process_completed
- プロセスインスタンスのエラー終了: process_completed_with_error_end_event
- プロセスインスタンスのキャンセル: process_cancelled

上記のイベントは設定ファイルによりそれぞれデータの登録の制御が可能です。
詳細は設定ファイルリファレンスを御覧ください。

以下のイベントは、プロセスインスタンスに含まれる変数情報も含めてElasticsearchに登録されます。

- ユーザタスクの完了: taks_completed
- プロセスインスタンスの終了: process_completed
- プロセスインスタンスのエラー終了: process_completed_with_error_end_event

インデックスのパターン

Elasticsearchに対してデータを登録する際、index, typeの指定が行えます。
設定ファイルに含めるindex, typeはプレースホルダを埋め込む事により動的に変更することが可能です。

例: im_bpm-${yyyyMMdd}

利用可能な置換文字は以下の通りです。

- yyyyMMdd: 年月日
- yyyyMM: 年月
- yyyy: 年
- MM: 月
- dd: 日
- tenantId: テナントID

インデックステンプレート例

Elasticsearchは、事前にindex templateを登録する必要があります。

index template例:

```
{
 "template": "im_bpm-default-*",
 "mappings": {
  "im_bpm": {
   "properties": {
    "type_activiti": {
     "type": "string",
     "index": "not_analyzed"
    },
    "time": {
     "type": "date",
     "format": "dateOptionalTime"
    },
    "user_cd": {
     "type": "string",
     "index": "not_analyzed"
    },
    "user_name": {
     "type": "string",
     "index": "not_analyzed"
    },
    "process_definition_id": {
     "type": "string",
     "index": "not_analyzed"
    },
    "process_instance_id": {
     "type": "string",
     "index": "not_analyzed"
```

```
    },
    "execution_id": {
      "type": "string",
      "index": "not_analyzed"
    },
    "activity_id": {
      "type": "string",
      "index": "not_analyzed"
    },
    "activity_name": {
      "type": "string",
      "index": "not_analyzed"
    },
    "activity_type": {
      "type": "string",
      "index": "not_analyzed"
    },
    "behavior_class": {
      "type": "string",
      "index": "not_analyzed"
    },
    "activity_cause": {
      "type": "string",
      "index": "not_analyzed"
    },
    "error_code": {
      "type": "string",
      "index": "not_analyzed"
    },
    "message_name": {
      "type": "string",
      "index": "not_analyzed"
    },
    "message_data": {
      "type": "string",
      "index": "not_analyzed"
    },
    "signal_name": {
      "type": "string",
      "index": "not_analyzed"
    },
    "signal_data": {
      "type": "string",
      "index": "not_analyzed"
    },
    "activity_duration": {
      "type": "long"
    },
    "nested_process_definition_id": {
      "type": "string",
      "index": "not_analyzed"
    },
    "nested_process_instance_id": {
      "type": "string",
      "index": "not_analyzed"
    },
    "process_cause": {
      "type": "string",
      "index": "not_analyzed"
    },
    "process_duration": {
      "type": "long"
    },
```

```
      "sequence_flow_id": {
        "type": "string",
        "index": "not_analyzed"
      },
      "source_activity_id": {
        "type": "string",
        "index": "not_analyzed"
      },
      "source_activity_name": {
        "type": "string",
        "index": "not_analyzed"
      },
      "source_activity_type": {
        "type": "string",
        "index": "not_analyzed"
      },
      "source_activity_behavior_class": {
        "type": "string",
        "index": "not_analyzed"
      },
      "target_activity_id": {
        "type": "string",
        "index": "not_analyzed"
      },
      "target_activity_name": {
        "type": "string",
        "index": "not_analyzed"
      },
      "target_activity_type": {
        "type": "string",
        "index": "not_analyzed"
      },
      "target_activity_behavior_class": {
        "type": "string",
        "index": "not_analyzed"
      },
      "task_id": {
        "type": "string",
        "index": "not_analyzed"
      },
      "task_name": {
        "type": "string",
        "index": "not_analyzed"
      },
      "task_definition_key": {
        "type": "string",
        "index": "not_analyzed"
      },
      "description": {
        "type": "string",
        "index": "not_analyzed"
      },
      "assignee": {
        "type": "string",
        "index": "not_analyzed"
      },
      "owner": {
        "type": "string",
        "index": "not_analyzed"
      },
      "category": {
        "type": "string",
        "index": "not_analyzed"
```

```
      },
      "due_date": {
        "type": "date",
        "format": "dateOptionalTime"
      },
      "form_key": {
        "type": "string",
        "index": "not_analyzed"
      },
      "priority": {
        "type": "string",
        "index": "not_analyzed"
      },
      "task_duration": {
        "type": "long"
      }
    }
  }
 }
}
```

コラム

上記テンプレートに加えて、プロセスインスタンスで利用される変数 (variables) 配下のプロパティを個別に指定する必要があります。

HTTP Proxy

Elasticsearchに対してHTTP Proxyが必要となる場合には、Proxyの指定をシステムプロパティで行います。

例:
　httpの場合
　　-Dhttp.proxyHost=127.0.0.1 -Dhttp.proxyPort=9080
　httpsの場合
　　-Dhttps.proxyHost=127.0.0.1 -Dhttps.proxyPort=9080

利用されているアプリケーションサーバに対して設定を行ってください。
　Resinの場合には、{RESIN_HOME}/conf/resin.properties に含まれる jvm_args プロパティに追記します。

マルチテナント

マルチテナントについて説明します。

項目
- バーチャルテナント

バーチャルテナント

IM-BPM for Accel Platformではバーチャルテナント単位でプロセス実行エンジンを持ちます。
また、プロセスに関連したデータベーステーブルは全てテナントデータベース内において管理されます。

注意

Activitiでは、マルチテナントとして同一テーブル内に定義したテナントIDを利用したテナント管理や、データベーススキーマをテナント毎に切り替える機能が存在しますが、 IM-BPM for Accel Platform ではこれらのテナント管理機構はサポートしていません。
API等に含まれるパラメータ “tenantId” を指定する必要はありません。

Java API

Java APIについて説明します。

パッケージ

IM-BPM for Accel Platformが提供する機能は、 jp.co.intra_mart.activiti パッケージです。

IM-BPM for Accel Platformでは、Javaパッケージ名に *.impl.* が含まれているクラス、インタフェースは直接の利用を推奨しません。
.impl. が含まれたクラス、インタフェースは将来的に互換性の無い変更を加える可能性があります。

Javaターゲットバージョン

IM-BPM for Accel Platformの提供するJava API群は全て Java7に対応したコード、バイトコードとなります。
Java7以降の環境で動作します。

プロセスエンジンの取得

テナント毎に管理されたプロセス実行エンジンは jp.co.intra_mart.activiti.engine.ProcessEngineFactory クラスから取得する事ができます。
Activiti標準で提供されている jp.co.intra_mart.activiti.engine.ProcessEngines クラスからは取得できません。

API

APIの詳細に関しては、APIドキュメントを御覧ください。

REST API

REST APIについて説明します。

項目

REST APIについて

IM-BPM for Accel Platformが提供するREST APIは、HTTPプロトロルを使用しプロセスの実行に関する様々な処理を呼び出すことが可能です。
REST APIはコンテンツタイプとして、application/json (JSON)形式のみ対応しています。
REST APIとして利用可能な機能は以下のとおりです。

- デプロイ
- プロセスエンジン情報の取得
- 実行状態の操作
- フォームデータの操作
- アクティビティ履歴の操作
- 履歴詳細に対する操作
- プロセスインスタンス履歴に対する操作
- ユーザタスクに関する操作
- 変数操作履歴
- ジョブの操作
- 全体管理情報の取得
- プロパティリソースへのアクセス
- プロセス定義情報に対する操作
- プロセスインスタンスに対する操作
- シグナルに関する操作
- ユーザタスクに関する操作
- インポート/エクスポート

認証方式

REST APIは以下の認証方式に対応しています。

- Cookieに紐づくセッションの認証状態に依存する方式

  アプリケーションサーバが発行するセッションIDおよびアカウントコンテキストの状態に依存する方式です。
  コンテキストの状態を確認後、認可によるチェックが行われます。

- Basic認証

  Basic認証による認証方式です。
  認証後、ログイン状態として扱われ認可によるチェックが行われます。

> **コラム**
> Basic認証を利用する場合には、httpsプロトコルの利用を強く推奨します。

認可

REST APIは全ての呼び出し先に対し認可リソースを持ちます。
intra-mart Accel Platform認可設定において、IM-BPM REST API認可種別が存在します。

> **注意**
>
> IM-BPM for Accel Platformの画面では、画面表示に必要となる情報を全てREST API経由で取得しています。
> その為、認可設定の変更により画面が正常に動作しなくなる場合があります。

エンドポイント

REST APIは認証方式によって、呼び出し先のエンドポイントが異なります。

- Cookieに紐づくセッションの認証状態に依存する方式

  http(s)://{HOST}:{PORT}/{CONTEXT_PATH}/api/bpm/....
  プロトコル、ホスト名、ポート番号、コンテキストパスは環境にあわせて置き換えてください。
  例:
  https://example.org/imart/api/bpm/management/engine

- Basic認証

  http(s)://{HOST}:{PORT}/{CONTEXT_PATH}/api/basic/bpm/....
  プロトコル、ホスト名、ポート番号、コンテキストパスは環境にあわせて置き換えてください。
  例:
  https://example.org/imart/api/basic/bpm/management/engine

Swagger

REST APIはSwagger Specに対応しています。
intra-mart Accel Platformに組み込まれているSwagger UIを通じてREST APIの確認、実行ができます。

Swagger UIは、以下のURLから確認することができます。

http(s)://{HOST}:{PORT}/{CONTEXT_PATH}/swagger_ui/?url=/{CONTEXT_PATH}/api-docs/im_bpm_rest
プロトコル、ホスト名、ポート番号、コンテキストパスは環境にあわせて置き換えてください。
例:
https://example.org/imart/swagger_ui/?url=/imart/api-docs/im_bpm_rest

> **コラム**
>
> Swagger Specを元に、Swagger CodeGenを利用するとREST APIクライアントコードの自動生成を行うことが可能です。
> https://github.com/swagger-api/swagger-codegen
>
> Swagger CodeGenにより作成されたスタブコードの動作保証は行っておりません。

REST APIドキュメント

APIの詳細に関しては、REST APIドキュメントを御覧ください。

EL式

EL式について説明します。

暗黙オブジェクト

ELで利用可能な暗黙オブジェクトは以下の通りです。

- authenticatedUserId (String)

  認証済みユーザコード
  プロセスを実行する際のユーザコードが格納されています。
  このユーザコードはアカウントコンテキストの持つユーザコードと同等です。
  タスクを非同期に実行、タイマーをトリガとした処理、並列ゲートウェイ等、実行単位が別のスレッドとなる処理を介した場合には、値にnullが格納されます。

- execution (DelegateExecution)

  実行時のエグゼキューションです。
  ${execution.id}: エグゼキューションIDの取得
  ${execution.processInstanceId}: プロセスインスタンスIDの取得
  ${execution.processBusinessKey}: 業務キーの取得
  ${execution.processDefinitionId}: プロセス定義IDの取得
  ${execution.superExecutionId}: 親エグゼキューションID
  ${execution.currentActivityId}: 実行中のアクティビティID
  ${execution.currentActivityName}: 実行中のアクティビティ名
  ${execution.getVariable(“varName”)}: 変数の取得

- task (DelegateTask)

  実行中のタスク情報です。
  この暗黙オブジェクトは、ユーザタスク等に設定するタスクリスナでのみ利用可能です。

- im_operation_users (Map<String, String>)

  ユーザタスクを処理したユーザコードを持ちます。
  ${im_operation_users[‘my-user-task’]} “my-user-task”というIDを持つタスクを処理したユーザコードの取得

> **注意**
>
> システム変数の格納方式の設定
>
> システム変数の格納方式の設定(is-system-variable-save-as-object)がtrueの場合は、暗黙オブジェクト “im_bpm_system_variables” オブジェクトの要素として”im_operation_users” オブジェクトが格納されます。
>
> 例: ユーザタスク(id: my-user-task) を処理したユーザコードをEL式にて取得する場合
> ${im_bpm_system_variables.im_operation_users[‘my-user-task’]}
>
> システム変数の格納方式の設定の詳細については、「 IM-BPM 設定ファイルリファレンス 」-「 システム変数の格納方式の設定 」を参照してください。

変数の操作

変数の存在確認

暗黙オブジェクト execution を利用します。
${execution.getVariable('myVarName') != null}

変数の追加

暗黙オブジェクト execution を利用します。
DelegateExpression等で利用します。
${execution.setVariable('myVarName', 'VALUE')}

変数の削除

暗黙オブジェクト execution を利用します。
DelegateExpression等で利用します。
${execution.removeVariable('myVarName')}

アーカイブ

IM-BPM for Accel Platform では完了したプロセスインスタンスのデータを退避する機能を用意しており、これを「アーカイブ」機能と呼びます。

アーカイブの仕様について説明します。

項目

アーカイブ機能

- アーカイブの対象は、完了したプロセスインスタンスです。
- アーカイブ用のテーブルにデータを退避します。
- アーカイブは、アーカイブジョブ（「*ジョブ一覧*」 - 「*アーカイブ*」）によって実行されます。

アーカイブ対象の指定

アーカイブ対象は、以下の指定された条件を満たす完了したプロセスインスタンスです。

- プロセスインスタンスの開始日時
- プロセスインスタンスの終了日時
- プロセス定義キー
- プロセス定義バージョン
- プロセスインスタンスID

各条件の指定方法については、「*ジョブ一覧*」 - 「*アーカイブ*」を参照してください。

条件を指定しない場合は、全ての完了したプロセスインスタンスがアーカイブの対象とされます。

アーカイブするデータの保存先

アーカイブするデータは、全てデータベースに保存されます。

プロセスインスタンスの開始日時の月単位でテーブルが作成されます。

テーブル名は以下の規則に沿って作成されます。

プロセスインスタンスの開始された月を数字6桁で表します。

- （例） 2016年11月1日 ⇒ 201611
- （例） 2017年1月15日 ⇒ 201701

元のテーブル名の一部分を、上記の数字に置き換えて作成します。

- （例） ACT_HI_PROCINST ⇒ ACT_201611_PROCINST
- （例） ACT_HI_ACTINST ⇒ ACT_201701_ACTINST

アーカイブする対象のテーブルは以下です。

元のテーブル名	アーカイブされるテーブル名（ **2017年4月の場合**）
ACT_HI_PROCINST	ACT_201704_PROCINST
ACT_HI_ACTINST	ACT_201704_ACTINST
ACT_HI_TASKINST	ACT_201704_TASKINST
ACT_HI_VARINST	ACT_201704_VARINST
ACT_HI_DETAIL	ACT_201704_DETAIL
ACT_HI_COMMENT	ACT_201704_COMMENT
ACT_HI_ATTACHMENT	ACT_201704_ATTACHMENT
ACT_HI_IDENTITYLINK	ACT_201704_IDENTITYLINK
im_act_hi_migration_actinst	im_act_201704_mg_actinst（テーブル名の文字制限により一部省略しています。）
ACT_GE_BYTEARRAY	ACT_201704_BYTEARRAY

コラム

コールアクティビティで呼び出されたプロセスインスタンスについては、呼び出し元のプロセスインスタンスの開始日付に依存します。
プロセスインスタンスAの開始日時が2017年1月25日で、プロセスインスタンスBをコールアクティビティで2017年2月1日に呼び出した場合、どちらも2017年1月の開始日時による規則に沿ってアーカイブされます。
プロセスインスタンスBが完了していても、プロセスインスタンスAが完了していない場合はアーカイブされません。

ジョブ一覧

IM-BPM for Accel Platform では以下のジョブを利用しています。

機能	ジョブ名	説明
アーカイブ	*アーカイブ*	完了したプロセスインスタンスのデータをアーカイブするジョブです。

アーカイブ

ジョブ概要

完了したプロセスインスタンスのデータをアーカイブするジョブです。
完了していないプロセスインスタンスは、アーカイブされません。

実行パラメータ

- ジョブに指定するパラメータリストです。

キー	値	名前	必須	デフォルト値
startedBefore		プロセスインスタンスの開始日付		
finishedBefore		プロセスインスタンスの終了日付		
startedBeforeDays		プロセスインスタンスの開始日付の日数指定		
finishedBeforeDays		プロセスインスタンスの終了日付の日数指定		
startedBeforeMonths		プロセスインスタンスの開始日付の月数指定		
finishedBeforeMonths		プロセスインスタンスの終了日付の月数指定		
processDefinitionKeyIn		プロセス定義キー（複数指定）		
processDefinitionKeyNotIn		除外するプロセス定義キー（複数指定）		
processDefinitionVersion		プロセス定義バージョン		

- **startedBefore（プロセスインスタンスの開始日付）**
  プロセスインスタンスの開始日付を”yyyy-MM-dd”形式で指定します。

- **finishedBefore（プロセスインスタンスの終了日付）**
  プロセスインスタンスの終了日付を”yyyy-MM-dd”形式で指定します。

- **startedBeforeDays（プロセスインスタンスの開始日付の日数指定）**
  現在日から過去の日数を指定します。指定された日付より以前に開始されたプロセスインスタンスを対象とします。
  （例）3を指定した場合、現在日が2017-01-17の場合、2017-01-13 23:59:999以前の開始したプロセスインスタンスを対象とします。

- **finishedBeforeDays（プロセスインスタンスの終了日付の日数指定）**
  現在日から過去の日数を指定します。指定された日付より以前に終了されたプロセスインスタンスを対象とします。
  （例）3を指定した場合、現在日が2017-01-17の場合、2017-01-13 23:59:999以前の終了したプロセスインスタンスを対象とします。

- **startedBeforeMonths（プロセスインスタンスの開始日付の月数指定）**

現在月から過去の月数を指定します。指定された日付より以前に開始されたプロセスインスタンスを対象とします。
（例）3を指定した場合、現在日が2017-01-17の場合、2016-09-30 23:59:999以前の開始したプロセスインスタンスを対象とします。

- **finishedBeforeMonths（プロセスインスタンスの終了日付の月数指定）**
  現在月から過去の月数を指定します。指定された日付より以前に終了されたプロセスインスタンスを対象とします。
  （例）3を指定した場合、現在日が2017-01-17の場合、2016-09-30 23:59:999以前の終了したプロセスインスタンスを対象とします。

- **processDefinitionKeyIn（プロセス定義キー（複数指定））**
  カンマ区切りで対象にするプロセス定義キーを指定します。指定しない場合、全てのプロセス定義を対象とします。

- **processDefinitionKeyNotIn（除外するプロセス定義キー（複数指定））**
  カンマ区切りで除外するプロセス定義キーを指定します。指定しない場合、全てのプロセス定義を対象とします。

- **processDefinitionVersion（プロセス定義バージョン）**
  プロセス定義バージョンを指定します。指定しない場合、全てのバージョンを対象とします。

ジョブネット

- このジョブが使用するジョブネットです。

  アーカイブ

関連ドキュメント

関連ドキュメントについて説明します。

概要

プロセス定義および、限られたアクティビティに関連したドキュメントを紐づけることができます。

関連ドキュメントの設定方法について

関連ドキュメントは以下に対して設定することができます。

- プロセス定義
- スタートイベント
- ユーザタスク
- マニュアルタスク

> **コラム**
>
> 設定方法の詳細は、「IM-BPM Designer 操作ガイド」 - 「関連ドキュメント」を参照してください。

関連ドキュメントの設定種別について

関連ドキュメントは、「デプロイ資材に含めるドキュメント」と「外部サイト」を選択することができます。
デプロイ資材に含めるドキュメントの場合は、デプロイ資材のドキュメントのファイル名を指定します。
外部サイトの場合は、URLを指定します。

デプロイ資材に含めるドキュメントの場合は、各関連する画面からダウンロードすることができます。
外部サイトの場合は、各関連する画面からURLに対するページを開きます。

> **コラム**
>
> 設定種別の詳細は、「IM-BPM Designer 操作ガイド」 - 「関連ドキュメント」を参照してください。
> 関連する画面については、「IM-BPM ユーザ操作ガイド」を参照してください。

画面仕様

一覧画面リクエストパラメータ

IM-BPM for Accel Platformの各一覧画面は、特定のリクエストパラメータを引き渡すことで、表示内容の絞り込みを行えます。

> **コラム**
>
> intra-mart Accel Platform のメニュー機能で、リクエストパラメータを引数として一覧画面に引き渡すことが可能です。
> 詳細については、「テナント管理者操作ガイド」-「メニューを設定する」を参照してください。

項目

ユーザ向け画面

プロセス開始一覧画面

プロセス開始一覧画面は、以下のリクエストパラメータを受け取ることが可能です。

項目名	キー	値
プロセス定義名	retains%5Bcondition%5D%5BnameLike%5D	文字列（部分一致）
カテゴリ	retains%5Bcondition%5D%5BcategoryLike%5D	文字列（部分一致）

タスク一覧画面

タスク一覧画面は、以下のリクエストパラメータを受け取ることが可能です。

グループタスク一覧

項目名	キー	値
プロセス定義名	group.retains%5Bcondition%5D%5BprocessDefinitionNameLike%5D	文字列（部分一致）
業務キー	group.retains%5Bcondition%5D%5BprocessInstanceBusinessKeyLike%5D	文字列（部分一致）
カテゴリ	group.retains%5Bcondition%5D%5BtaskCategory%5D	文字列（完全一致）
タスク名	group.retains%5Bcondition%5D%5BnameLike%5D	文字列（部分一致）

項目名	キー	値
優先度（最小値）	group.retains%5Bcondition%5D%5BminimumPriority%5D	数値（整数値）
優先度（最大値）	group.retains%5Bcondition%5D%5BmaximumPriority%5D	数値（整数値）
作成日時（from）	group.retains%5Bcondition%5D%5BcreatedAfter%5D	日付（YYYY%2FMM%2FDD）
作成日時（to）	group.retains%5Bcondition%5D%5BcreatedBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時（from）	group.retains%5Bcondition%5D%5BdueAfter%5D	日付（YYYY%2FMM%2FDD）
期限日時（to）	group.retains%5Bcondition%5D%5BdueBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時なし	group.retains%5Bcondition%5D%5BwithoutDueDate%5D	真偽値（true/false）

個人タスク一覧

項目名	キー	値
プロセス定義名	user.retains%5Bcondition%5D%5BprocessDefinitionNameLike%5D	文字列（部分一致）
業務キー	user.retains%5Bcondition%5D%5BprocessInstanceBusinessKeyLike%5D	文字列（部分一致）
カテゴリ	user.retains%5Bcondition%5D%5BtaskCategory%5D	文字列（完全一致）
タスク名	user.retains%5Bcondition%5D%5BnameLike%5D	文字列（部分一致）
優先度（最小値）	user.retains%5Bcondition%5D%5BminimumPriority%5D	数値（整数値）
優先度（最大値）	user.retains%5Bcondition%5D%5BmaximumPriority%5D	数値（整数値）
作成日時（from）	user.retains%5Bcondition%5D%5BcreatedAfter%5D	日付（YYYY%2FMM%2FDD）
作成日時（to）	user.retains%5Bcondition%5D%5BcreatedBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時（from）	user.retains%5Bcondition%5D%5BdueAfter%5D	日付（YYYY%2FMM%2FDD）
期限日時（to）	user.retains%5Bcondition%5D%5BdueBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時なし	user.retains%5Bcondition%5D%5BwithoutDueDate%5D	真偽値（true/false）

グループタスク一覧画面

グループタスク一覧画面は、以下のリクエストパラメータを受け取ることが可能です。

項目名	キー	値
プロセス定義名	retains%5Bcondition%5D%5BprocessDefinitionNameLike%5D	文字列（部分一致）
業務キー	retains%5Bcondition%5D%5BprocessInstanceBusinessKeyLike%5D	文字列（部分一致）
カテゴリ	retains%5Bcondition%5D%5BtaskCategory%5D	文字列（完全一致）
タスク名	retains%5Bcondition%5D%5BnameLike%5D	文字列（部分一致）
優先度（最小値）	retains%5Bcondition%5D%5BminimumPriority%5D	数値（整数値）
優先度（最大値）	retains%5Bcondition%5D%5BmaximumPriority%5D	数値（整数値）
作成日時（from）	retains%5Bcondition%5D%5BcreatedAfter%5D	日付（YYYY%2FMM%2FDD）
作成日時（to）	retains%5Bcondition%5D%5BcreatedBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時（from）	retains%5Bcondition%5D%5BdueAfter%5D	日付（YYYY%2FMM%2FDD）
期限日時（to）	retains%5Bcondition%5D%5BdueBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時なし	retains%5Bcondition%5D%5BwithoutDueDate%5D	真偽値（true/false）

個人タスク一覧画面

個人タスク一覧画面は、以下のリクエストパラメータを受け取ることが可能です。

項目名	キー	値
プロセス定義名	retains%5Bcondition%5D%5BprocessDefinitionNameLike%5D	文字列（部分一致）
業務キー	retains%5Bcondition%5D%5BprocessInstanceBusinessKeyLike%5D	文字列（部分一致）
カテゴリ	retains%5Bcondition%5D%5BtaskCategory%5D	文字列（完全一致）
タスク名	retains%5Bcondition%5D%5BnameLike%5D	文字列（部分一致）
優先度（最小値）	retains%5Bcondition%5D%5BminimumPriority%5D	数値（整数値）
優先度（最大値）	retains%5Bcondition%5D%5BmaximumPriority%5D	数値（整数値）
作成日時（from）	retains%5Bcondition%5D%5BcreatedAfter%5D	日付（YYYY%2FMM%2FDD）

項目名	キー	値
作成日時（to）	retains%5Bcondition%5D%5BcreatedBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時（from）	retains%5Bcondition%5D%5BdueAfter%5D	日付（YYYY%2FMM%2FDD）
期限日時（to）	retains%5Bcondition%5D%5BdueBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時なし	retains%5Bcondition%5D%5BwithoutDueDate%5D	真偽値（true/false）

処理済一覧画面

処理済一覧画面は、以下のリクエストパラメータを受け取ることが可能です。

項目名	キー	値
プロセス定義名	retains%5Bcondition%5D%5BprocessDefinitionNameLike%5D	文字列（部分一致）
業務キー	retains%5Bcondition%5D%5BprocessBusinessKeyLike%5D	文字列（部分一致）
カテゴリ	retains%5Bcondition%5D%5BtaskCategory%5D	文字列（完全一致）
タスク名	retains%5Bcondition%5D%5BtaskNameLike%5D	文字列（部分一致）
優先度（最小値）	retains%5Bcondition%5D%5BtaskMinPriority%5D	数値（整数値）
優先度（最大値）	retains%5Bcondition%5D%5BtaskMaxPriority%5D	数値（整数値）
作成日時（from）	retains%5Bcondition%5D%5BtaskCreatedAfter%5D	日付（YYYY%2FMM%2FDD）
作成日時（to）	retains%5Bcondition%5D%5BtaskCreatedBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時（from）	retains%5Bcondition%5D%5BdueDateAfter%5D	日付（YYYY%2FMM%2FDD）
期限日時（to）	retains%5Bcondition%5D%5BdueDateBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時なし	retains%5Bcondition%5D%5BwithoutDueDate%5D	真偽値（true/false）
終了日時（from）	retains%5Bcondition%5D%5BtaskCompletedAfter%5D	日付（YYYY%2FMM%2FDD）
終了日時（to）	retains%5Bcondition%5D%5BtaskCompletedBefore%5D	日付（YYYY%2FMM%2FDD）

管理者向け画面

プロセス一覧画面

プロセス一覧画面は、以下のリクエストパラメータを受け取ることが可能です。

項目名	キー	値
業務キー	searchForm%5BbusinessKey%5D	文字列（完全一致）
プロセス定義ID	searchForm%5BprocessDefinitionId%5D	文字列（完全一致）
プロセス定義キー	searchForm%5BprocessDefinitionKey%5D	文字列（完全一致）
プロセス定義バージョン	searchForm%5BprocessDefinitionVersion%5D	数値（整数値）
プロセス定義名	searchForm%5BprocessDefinitionName%5D	文字列（完全一致）
カテゴリ	searchForm%5BprocessDefinitionCategory%5D	文字列（完全一致）
開始日（from）	searchForm%5BstartedAfter%5D	日付（YYYY%2FMM%2FDD）
開始日（to）	searchForm%5BstartedBefore%5D	日付（YYYY%2FMM%2FDD）
ステータス	searchForm%5Bstatus%5D	以下のいずれかの固定文字列 running error finished
XXX番目の変数名	searchForm%5Bvariables%5D%5BXXX%5D%5Bname%5D	文字列（完全一致）
XXX番目の変数の型	searchForm%5Bvariables%5D%5BXXX%5D%5Btype%5D	以下のいずれかの固定文字列 string integer long double date boolean
XXX番目の変数の演算子	searchForm%5Bvariables%5D%5BXXX%5D%5Boperation%5D	以下のいずれかの固定文字列 equals notEquals equalsIgnoreCase like lessThan lessThanOrEquals greaterThan greaterThanOrEquals
XXX番目の変数の値	searchForm%5Bvariables%5D%5BXXX%5D%5Bvalue%5D	ー

タスク管理一覧画面

タスク管理一覧画面は、以下のリクエストパラメータを受け取ることが可能です。

項目名	キー	値
プロセス定義名	condition%5BprocessDefinitionNameLike%5D	文字列（部分一致）
業務キー	condition%5BprocessInstanceBusinessKeyLike%5D	文字列（部分一致）
カテゴリ	condition%5BtaskCategory%5D	文字列（完全一致）
タスク名	condition%5BnameLike%5D	文字列（部分一致）
優先度（最小値）	condition%5BminimumPriority%5D	数値（整数値）
優先度（最大値）	condition%5BmaximumPriority%5D	数値（整数値）
担当者未割り当て	condition%5Bunassigned%5D	真偽値（true/false）
担当者	condition%5Bassignee%5D	ユーザコード（完全一致）
作成日時（from）	condition%5BcreatedAfter%5D	日付（YYYY%2FMM%2FDD）
作成日時（to）	condition%5BcreatedBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時（from）	condition%5BdueAfter%5D	日付（YYYY%2FMM%2FDD）
期限日時（to）	condition%5BdueBefore%5D	日付（YYYY%2FMM%2FDD）
期限日時なし	condition%5BwithoutDueDate%5D	真偽値（true/false）