関係者外秘

PJ関係者限り

[プロジェクト名]

ワークフロー設計ガイド

第１．０版

ＹＹＹＹ年Ｍ月Ｄ日

[会社名]

[部門名]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | 版数 | 変更日 | 区分 | 変更箇所 | | 変更内容 | 担当者 |
| 頁 | 項番 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

[1． 本書について 1](#_Toc399240713)

[2． 前提条件 2](#_Toc399240714)

[3． BPMNモデリングツールのインストール 2](#_Toc399240715)

[3.1. camunda Modeler のダウンロードおよびインストール 2](#_Toc399240716)

[4． ワークフロー定義 3](#_Toc399240717)

[4.1. ワークフロー定義ファイルの新規作成 4](#_Toc399240718)

[4.2. ワークフローの記述（基本編） 5](#_Toc399240719)

[4.2.1. プールおよびレーンの設定 6](#_Toc399240720)

[4.2.1. 開始イベントの設定 7](#_Toc399240721)

[4.2.2. ユーザタスクの設定 8](#_Toc399240722)

[4.2.3. ゲートウェイの設定 10](#_Toc399240723)

[4.2.4. シーケンスフローの設定 11](#_Toc399240724)

[4.2.5. 境界イベントの設定 13](#_Toc399240725)

[4.2.6. 停止イベントの設定 15](#_Toc399240726)

[4.3. ワークフローの記述（応用編） 16](#_Toc399240727)

[4.3.1. マルチインスタンスタスク 16](#_Toc399240728)

[4.3.2. バッチ処理で実行するタスク 18](#_Toc399240729)

[4.3.3. 標準表記と異なる進行条件 19](#_Toc399240730)

[5． 承認履歴/承認状況（ステータス）に関する設計 20](#_Toc399240731)

[5.1. 承認履歴 20](#_Toc399240732)

[5.2. 承認状況（ステータス） 20](#_Toc399240733)

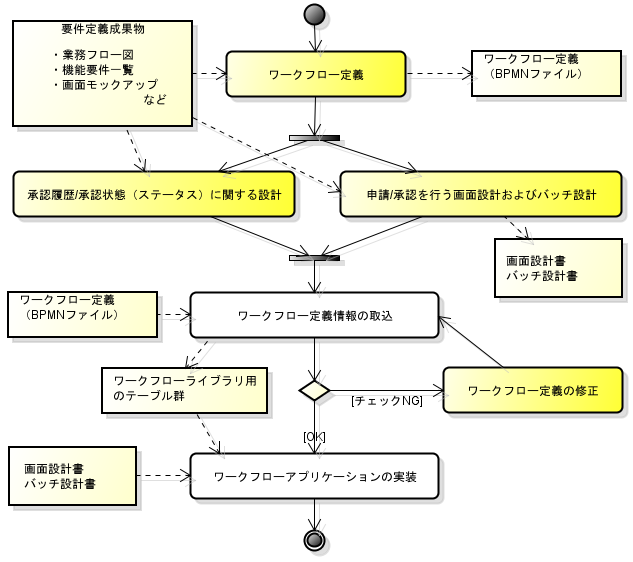
[6． 画面設計およびバッチ設計との関連 21](#_Toc399240734)

# 本書について

本書は、Nablarchの「ワークフローライブラリ」を利用して、ワークフローアプリケーションを開発するための「設計ガイド」である。

ワークフローライブラリは、アプリケーションの一部に「申請～承認を行う機能」が含まれる開発案件をターゲットとしており、フローの進行制御に必要な “ワークフローAPI”を中心とする機能を提供している。

ワークフローの設計から実装までの作業は、以下の通りであり、本書では、黄色に塗り潰された作業を説明対象とする。



|  |  |
| --- | --- |
| ＜作業内容の参照先について＞ | |
| 作業名 | 作業内容の参照先 |
| ワークフロー定義 | 本書の「４．ワークフロー定義」 |
| 承認履歴/承認状態（ステータス）に関する設計 | 本書の「５．承認履歴/承認状況（ステータス）に関する設計」 |
| 申請/承認を行う画面設計およびバッチ設計 | 本書の「６．画面設計およびバッチ設計との関連」 |
| ワークフロー定義情報の取込 | 「ワークフロー定義データ作成ツール」の解説書 |
| ワークフロー定義の修正 | 本書の「４．ワークフロー定義」  （但し、取込時のチェック内容については、「ワークフロー定義データ作成ツール」の解説書を参照。） |
| ワークフローアプリケーションの実装 | 以下の関連ドキュメント  ・ワークフローライブラリの解説書  ・ワークフローライブラリを利用した業務アプリケーション実装ガイド |
| ※黄色に塗り潰された作業を本書の説明対象とする。 | |

# 前提条件

“BPMN2.0（<http://www.omg.org/spec/BPMN/2.0/>）の記法に基づき、業務要件にあった申請/承認フローを描く作業”を「ワークフロー定義」と用語を定義する。

ワークフロー定義は、BPMN2.0準拠のモデリングツールを利用することを前提とする。

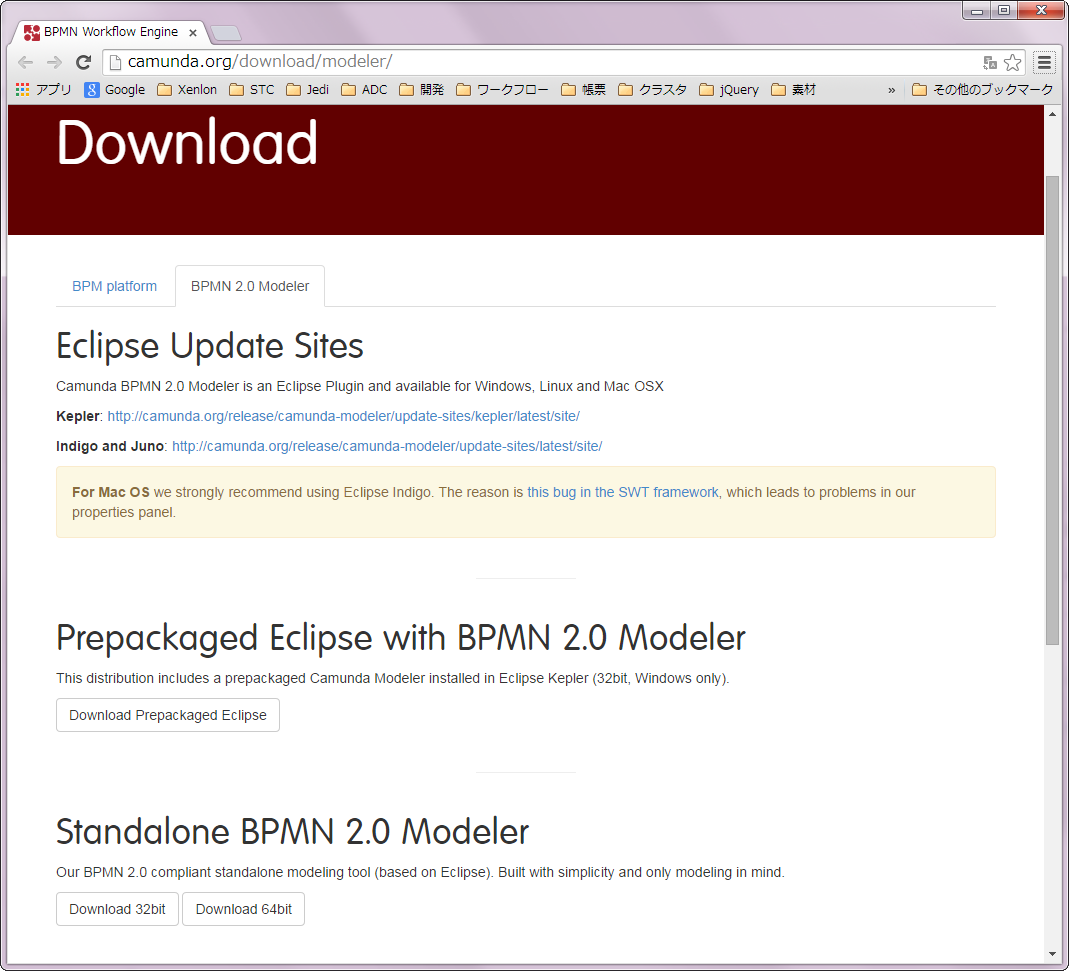
BPMN2.0準拠のモデリングツールには、様々な無償/有償ツールがあるが、Nablarchワークフローライブラリでは、「camunda Modeler（http://camunda.org/bpmn/tool/）」を利用して開発および動作確認を行っているため、「camunda Modeler」の利用を前提に説明を行う。

# BPMNモデリングツールのインストール

## camunda Modeler のダウンロードおよびインストール

以下のダウンロードサイトから「スタンドアロン版（Standalone）」をダンロードし、標準インストールを実施すること。

<http://camunda.org/download/modeler/>



※2014/09/08時点の最新バージョンは、

Ver1.2.0（2014/5/26 Released）

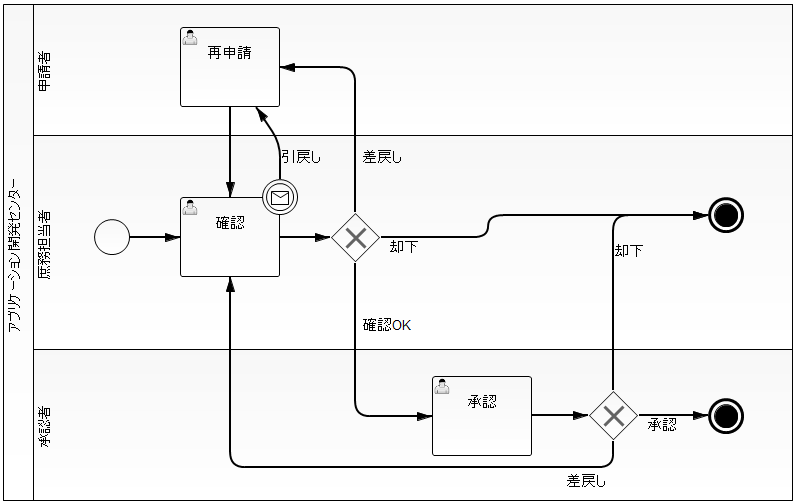
# ワークフロー定義

「camunda Modeler」を利用して、ワークフロー定義を行う。

ワークフロー定義は、「開始イベント」「シーケンスフロー」「ユーザタスク」「ゲートウェイ」「停止イベント」といった主要な要素を使って行う。

まずは、Nablarchワークフローライブラリに同梱されている交通費申請（サンプルアプリケーション）のワークフロー定義を例に基本的な記述方法を説明する。

【例】交通費申請のワークフロー定義



## ワークフロー定義ファイルの新規作成

「camunda Modeler」のメニューから[File]-[New]-[BPMN 2.0 File(external)]を選択する。

ファイル名は、以下のルールに従い命名すること。

ファイル名の命名ルール：　　ワークフローID\_ワークフロー名\_バージョン\_適用日.bpmn　　　※拡張子は、bpmnとする。

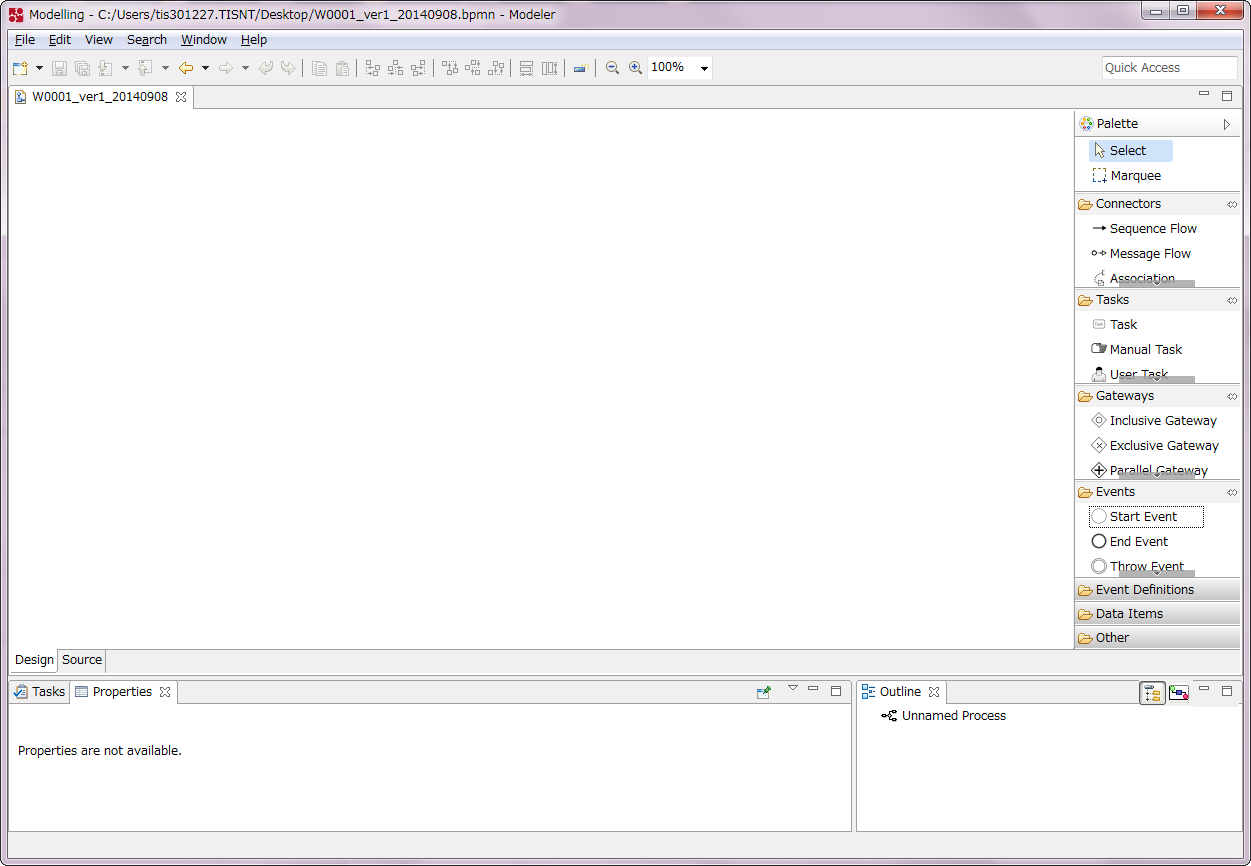
[例]　WF0001\_交通費申請\_ver1\_20140908.bpmn

|  |  |
| --- | --- |
| ワークフローID | ワークフロー定義を一意に識別するためのID。他のワークフローと重複しないように採番すること。  [例] WF0001  ※ワークフローIDを含むワークフローに関するID体系の詳細は、「方式設計別冊（ＩＤ体系）」を参照。 |
| ワークフロー名 | ワークフローの名前（日本語可）を設定する。  [例] 交通費申請 |
| バージョン | システム運用後にワークフロー定義が変更されること考慮し、ワークフロー定義のバージョンを設定する。  必ず「適用日」とセットで設定すること。  ・接頭語として必ず　ver（半角小文字）を付与すること。  ・ver1のように”ver”＋バージョン番号（半角数字）と設定すること。  [例] ver1 |
| 適用日 | システム運用後にワークフロー定義が変更されること考慮し、ワークフロー定義の適用日を設定する。  必ず「バージョン」とセットで設定すること。  ・日付書式は、yyyyMMdd とすること。  [例] 20140908 |

## ワークフローの記述（基本編）

ワークフロー定義ファイルを新規に作成すると、以下のビューが起動される。

画面右側の「Palette（パレット）」から必要な要素をドラッグ＆ドロップで配置し、直感的な操作でフローを描く。



**パレット**

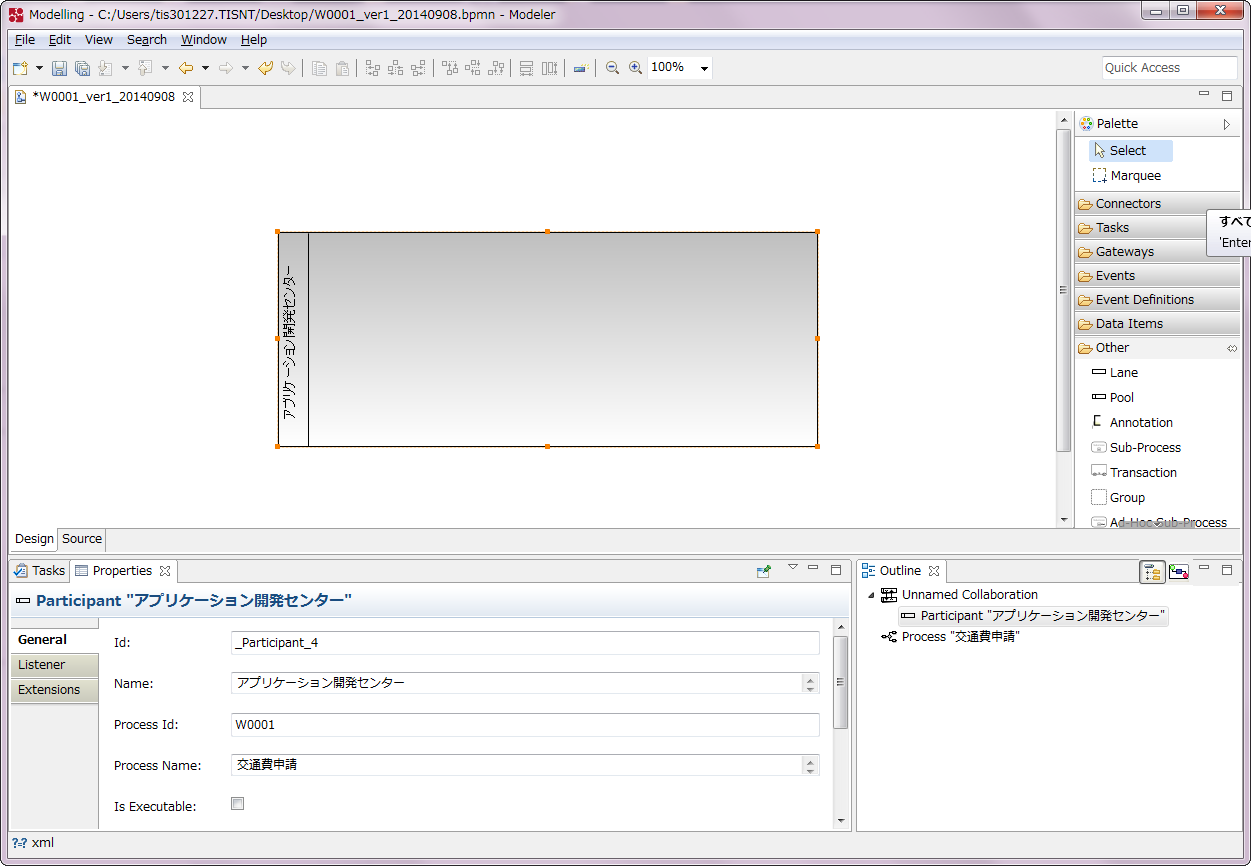
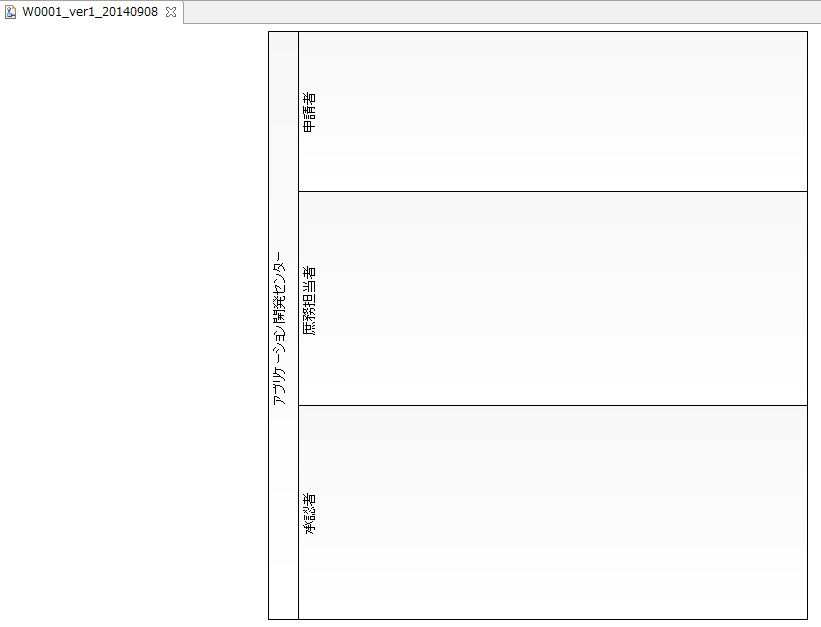
[注意]

ワークフローライブラリで利用できる要素については、

ワークフローライブラリの解説書を参照。

### プールおよびレーンの設定

まず、「プール」および「レーン」の要素を配置する。



複数のレーンを縦に並べる場合は、

プール名の所にドラッグしてください。

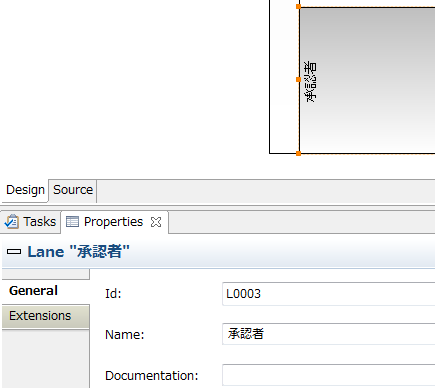


**プール**

**レーン**

[注意]

「Id:」にレーンID、「Name：」に役割名/部署名などのレーン名を必ず設定すること。



「レーン」のプロパティ

[注意]

プールは、必ず１つだけ設定すること。

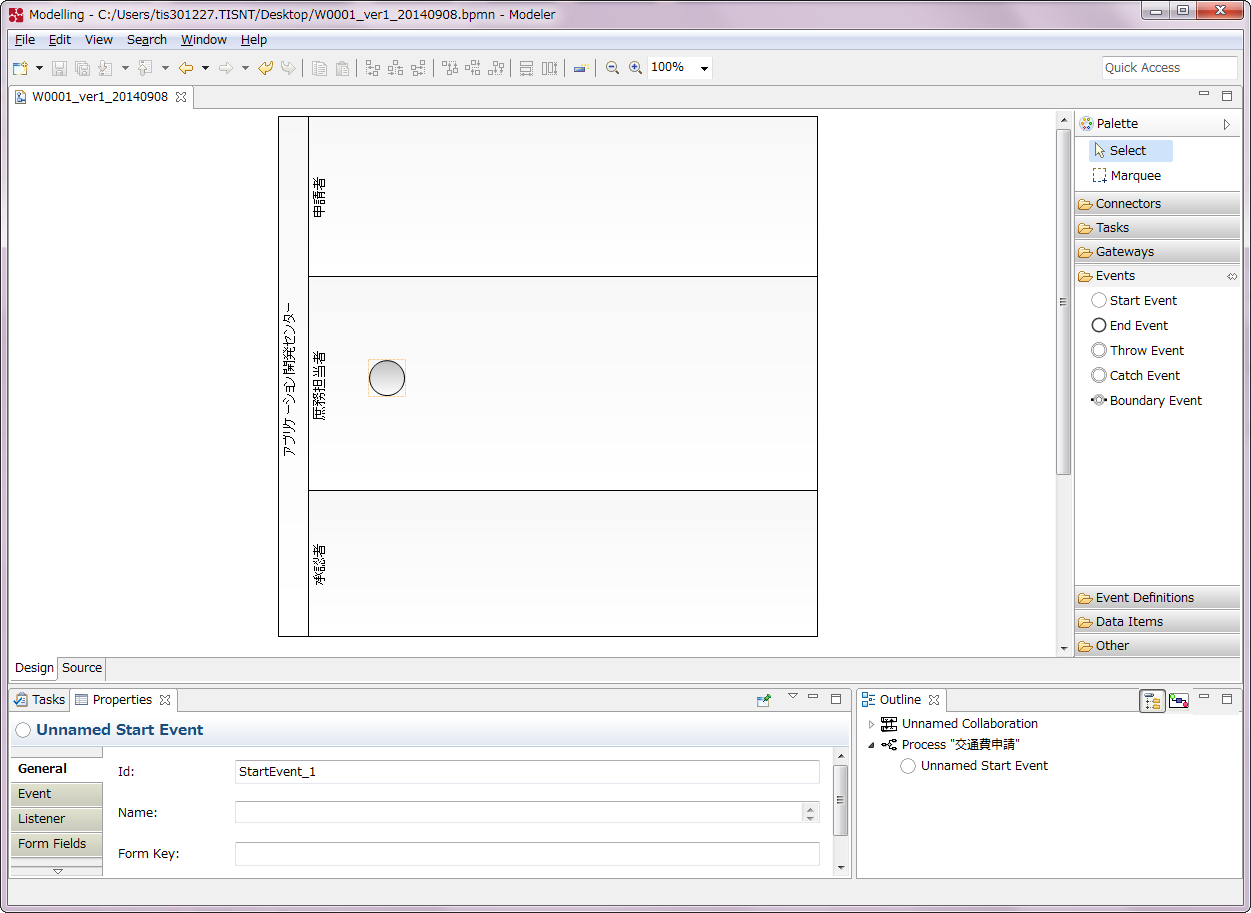
レーンは、プール内に必ず１つ以上設定すること。

|  |  |
| --- | --- |
| プール | プールは、企業やビジネスロール（売り手、買い手、製造者）、部門といったワークフローの関係者を表す。  ※プールは、プログラム上では利用されない要素であるため、プロパティの設定は不要。 |
| レーン | 役割や部署などによって、プール内を分割する場合にレーンを使用する。  ※レーンIDは、プログラム上で**タスクに対するユーザ/グループの割り当てをする際に指定するID**として利用する。  ※レーンは、認可機能で管理するグループ（組織）と合せることも可能。（固有のグループ情報を設定することも可能。） |

### 開始イベントの設定

「開始イベント」は、ワークフローの開始位置を示す。

１つのワークフロー定義に“必ず１つの開始イベント”を設定すること。



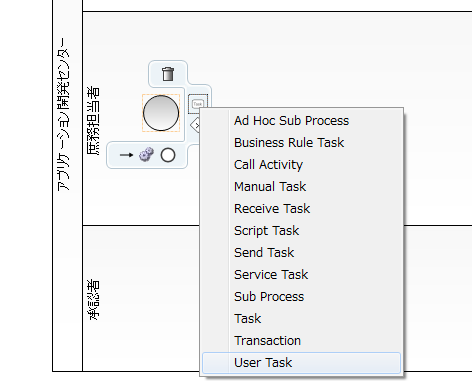
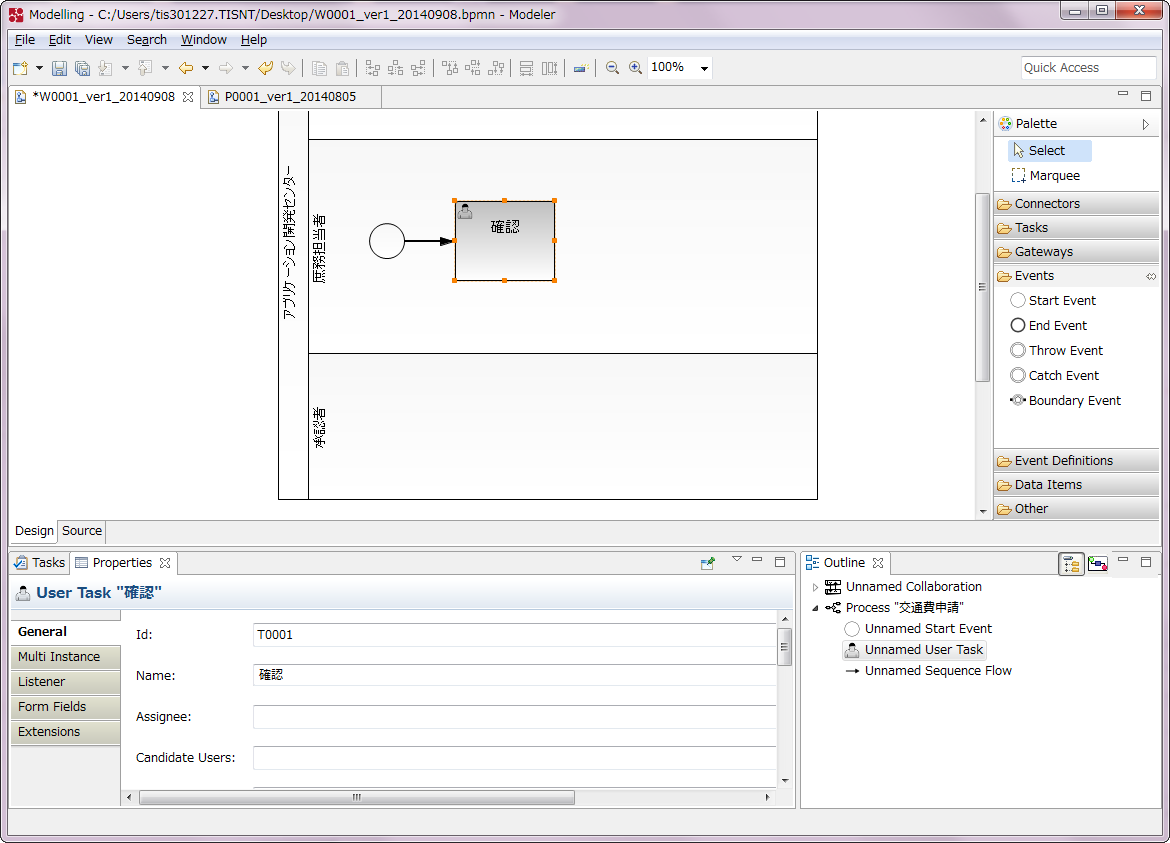
[注意]

必ず「１つだけ」設定すること。

Nameプロパティ以下は、設定不要。

### ユーザタスクの設定

ワークフロー上の承認行為（承認/却下/差戻し 等）は、「ユーザタスク」という要素を使う。



1. **アイコンをクリック**

**③「ユーザタスク」が追加される**

1. **User Taskを選択**

|  |  |
| --- | --- |
| タスクID | 「Id:」にタスクIDを設定すること。  ※タスクIDは、プログラム上で**タスクに対するユーザ/グループの割り当てをする際に指定するID**として利用する。 |
| タスク名 | 「Name:」に適切なタスク名を設定すること。 |

■「下書き保存」の扱いについて

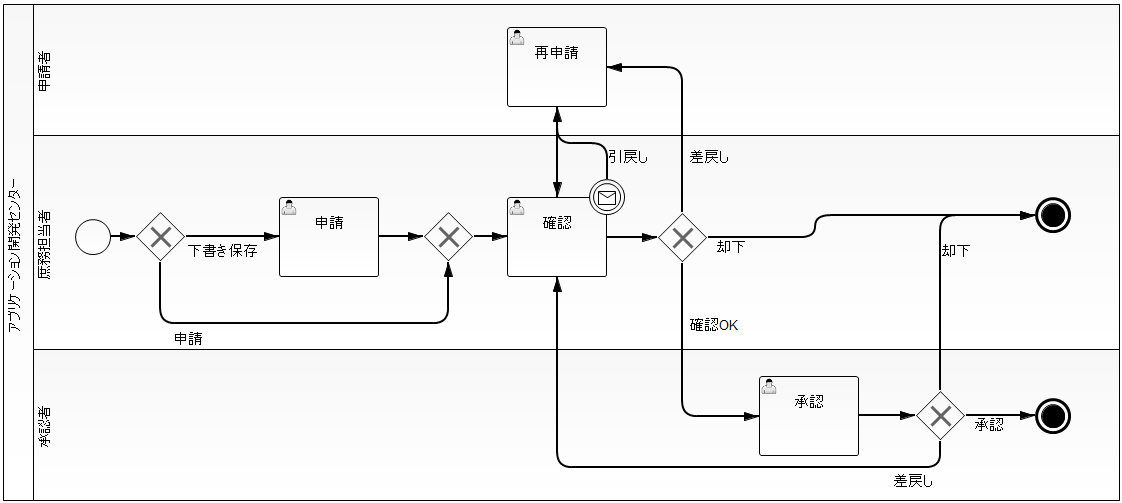
申請データの「下書き保存」は、言葉のとおり、“未申請の状態”。

Nablarchワークフローライブラリでは、**「下書き保存の状態は、ワークフローの開始前」と捉え、ワークフロー定義に記述しない。**

＜ワークフロー定義に記述しない理由＞

BPMN2.0の記法を用いて、以下のように「下書き保存」からワークフローを描くことができるが、

「下書き後に申請情報を削除する場合、ワークフローAPI側の情報も整合を取らなければならない」など、アプリケーションの実装が煩雑になるため。

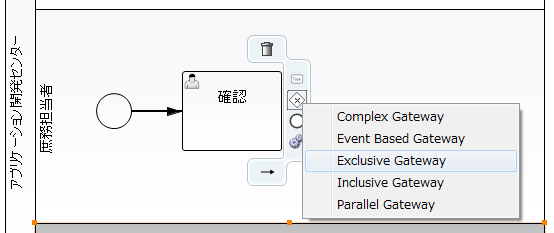


[注意]

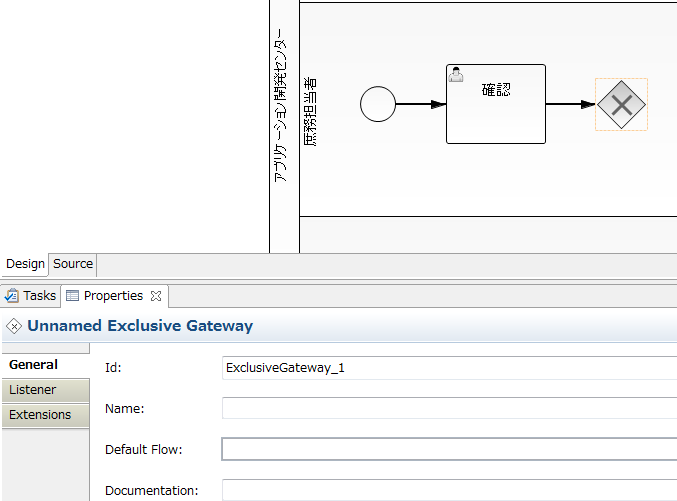
下書き保存後からワークフローを開始するような記述をしないこと。

### ゲートウェイの設定

「ユーザタスク」の実行後に承認/却下などの判定結果に応じてフローが分岐する場合は、「ゲートウェイ」という要素を使う。



1. **アイコンをクリック**



**②Exc lusive Gatewayを選択**

[注意]

Nablarchワークフローライブラリで使用できるゲートウェイは

「Exclusive Gateway（XORゲートウェイ）」のみ。

その他のゲートウェイは、ワークフローAPI側が対応していない。

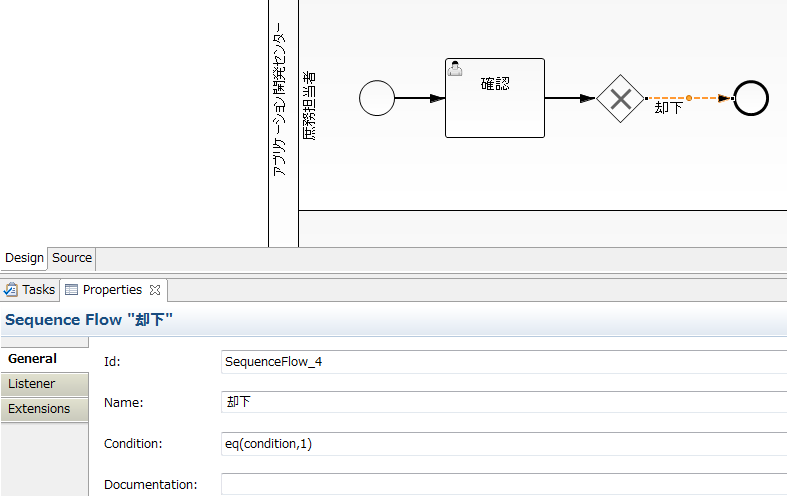
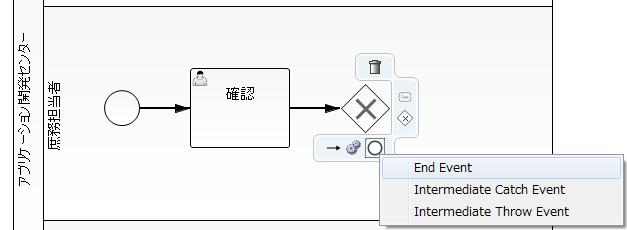
**③XORゲートウェイが追加される**

Nameプロパティ以下は、設定不要。

### シーケンスフローの設定

ゲートウェイ通過後の分岐は、「シーケンスフロー」の要素を複数設定することで表現する。

例えば、確認行為の結果、[却下]と判定された場合、申請自体が却下で終了となるため、以下のようにフローを描く。



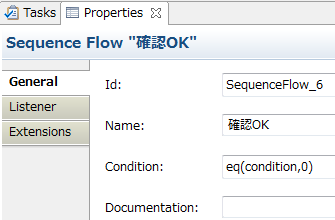
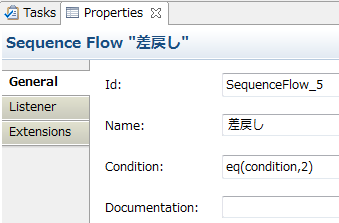
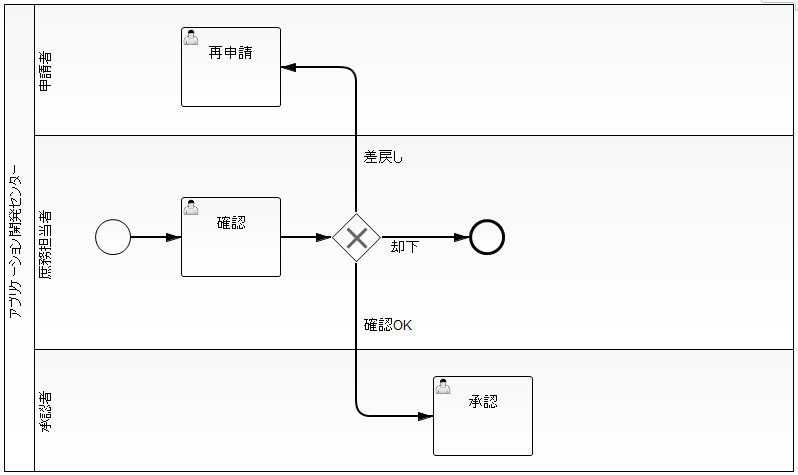
1. **アイコンをクリック**

**③シーケンスフローが追加される**

**②End Eventを選択**

|  |  |
| --- | --- |
| シーケンスフロー名 | 「Name:」に承認/却下等の適切な名前を設定すること。 |
| 進行条件 | 「Condition:」にこのシーケンスフローへの進行条件を設定すること。  例えば、eq(condition,1)という記述は、conditionという変数に1がセットされていた場合に、このシーケンスフローへ進むという制御がワークフローAPIによって行なわれる。  ※プログラム上では、「確認」のタスクを完了させるタイミングで、「キー＝”condition ”、値＝”1”」を格納したMap型の変数をワークフローAPIに渡すことで、進行方向を指示する。 |

庶務担当者による「確認」タスクの結果、確認OK/却下/差戻しで分岐する場合の記述例は、以下の通り。



[注意]

進行条件の値は、重複しないように設定すること。

交通費申請の例では、0:確認OK、1:却下、2:差戻し と設定。

＜進行条件の「標準表記」＞

eq(変数, 期待値)： 変数の値と期待値が一致する場合

ne(変数, 期待値)： 変数の値と期待値が不一致の場合

gt(変数, 期待値)： 変数の値と期待値より大きい場合

ge(変数, 期待値)： 変数の値と期待値以上の場合

lt(変数, 期待値)： 変数の値と期待値より小さい場合

le(変数, 期待値)： 変数の値と期待値以下の場合

[例] “入力金額が10万円以上”という進行条件の場合、ge(condition, 100000) と設定する。

「標準表記」を使わずに、進行条件を判定するコンポーネント

を設定することも可能。

詳細は、「4.3.3標準表記と異なる進行条件」を参照。

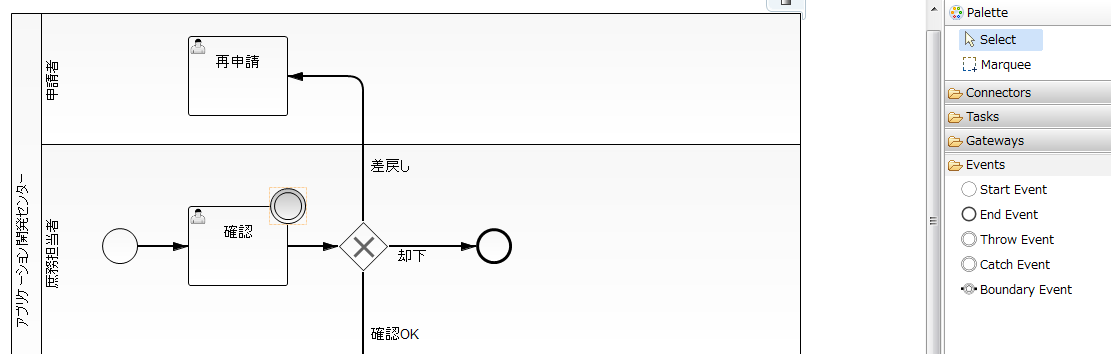


### 境界イベントの設定

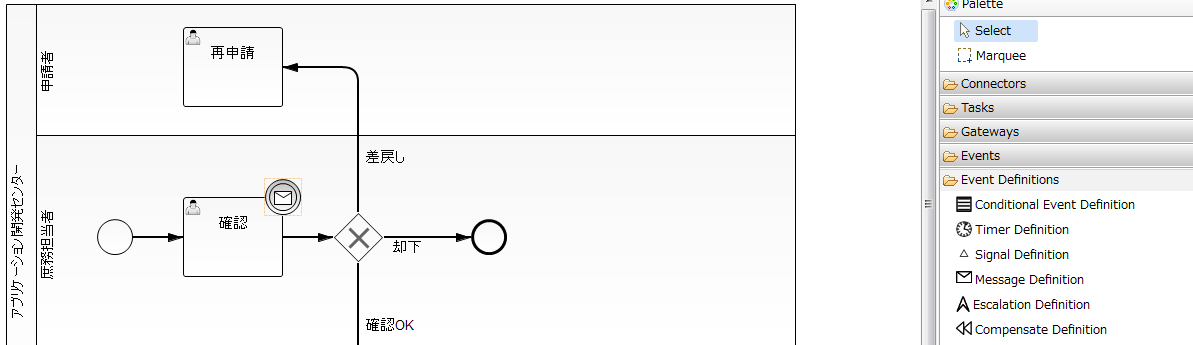
境界イベントは、あるタスクがアクティブな状態の時に、**アサインに関係なくアクテイブな状態を中断し、通常のフローとは別の進行制御を行いたい場合**に使うことができる。

例えば、庶務担当者による「確認」タスクがアクティブな状態で、申請者（このタスクにアサインされていないユーザ）が申請内容の不備に気が付き、自身に戻したい場合がある。

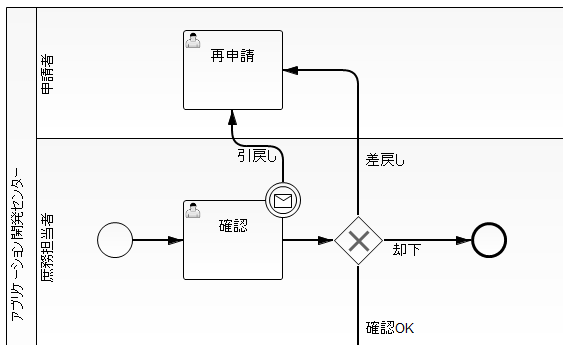
このような行為を「引戻し」と呼ぶが、この「引戻し」をワークフローとして定義する場合は、「境界イベント」の要素を使う。



1. **割り込みたいタスクにBoundary Eventをドラッグ＆ドロップする**



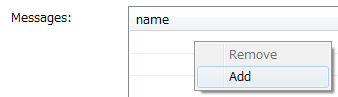
**②Boundary EventにMessage Definitionをドラッグ＆ドロップする**



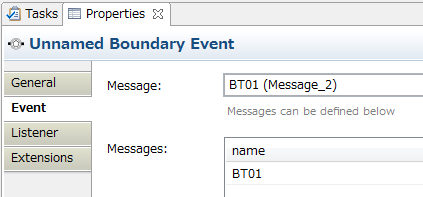
**③Boundary Eventから中断後に進行するシーケンスフローを追加する**

**【補足】**

**はじめに、この場所で「右クリック」し、Addを選択後、name属性に“境界イベントトリガーID”を追加する必要がある。**



**④中断のトリガーとして受け取る「境界イベントトリガーID」を設定する**



「Massage:」のところで追加した”境界イベントトリガーID”を設定する。

[注意]

境界イベントトリガーIDは、１つのワークフロー定義内で一意になるように採番すること。

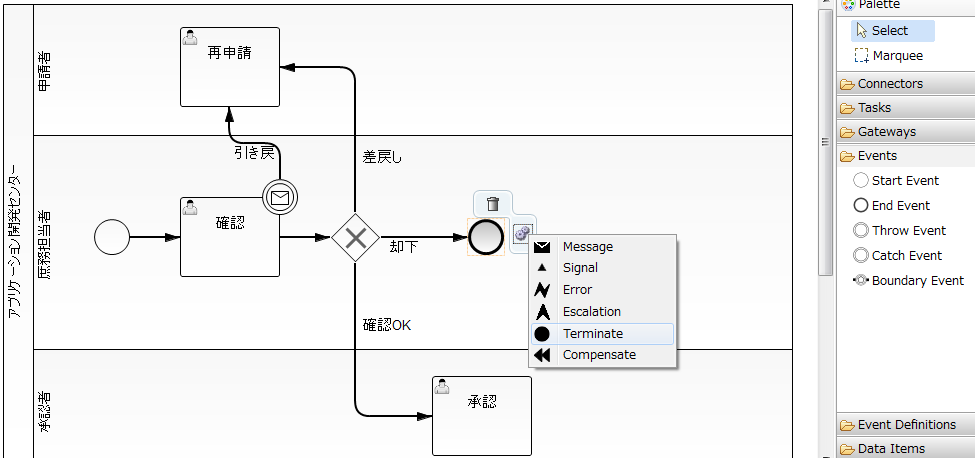
※境界イベントトリガーIDは、プログラム上で中断を起こす際に指定するIDとして利用する。

※境界イベントトリガーIDのID体系については、「方式設計別冊（ＩＤ体系）」を参照。

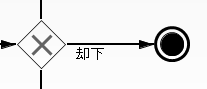
### 停止イベントの設定

開始したワークフローを終了させるには、「停止イベント」の要素を使う。

交通費申請の例では、『庶務担当者による「確認」タスクで“却下”と判定された場合』および『承認者による「承認」タスクで“承認/却下”と判定された場合』にワークフローを終了するので、停止イベントを２箇所設定している。



1. **End Eventをドラッグ＆ドロップする**
2. **歯車のアイコンをクリックし、Terminateを選択する**



停止イベントのNameプロパティ以下は、設定不要。

## ワークフローの記述（応用編）

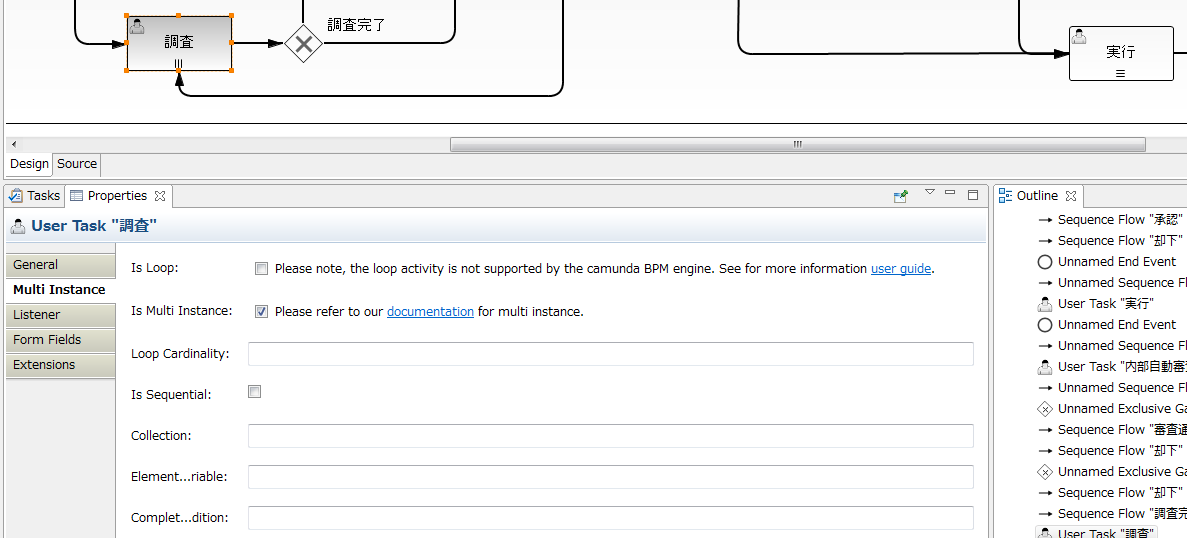
応用編では、「マルチインスタンスタスク」「バッチ処理で実行するタスク」「標準表記と異なる進行条件」について説明する。

### マルチインスタンスタスク

１つのユーザタスクに複数のユーザを割り当てることが可能です。このようなタスクをマルチインスタンスタスクと呼ぶ。

マルチインスタンスタスクには、割り当てたユーザ全員を並列にアクティブとする「Parallel（並列）」と

決められた順序で直列にアクティブとする「Sequential（直列）」の２種類がある。



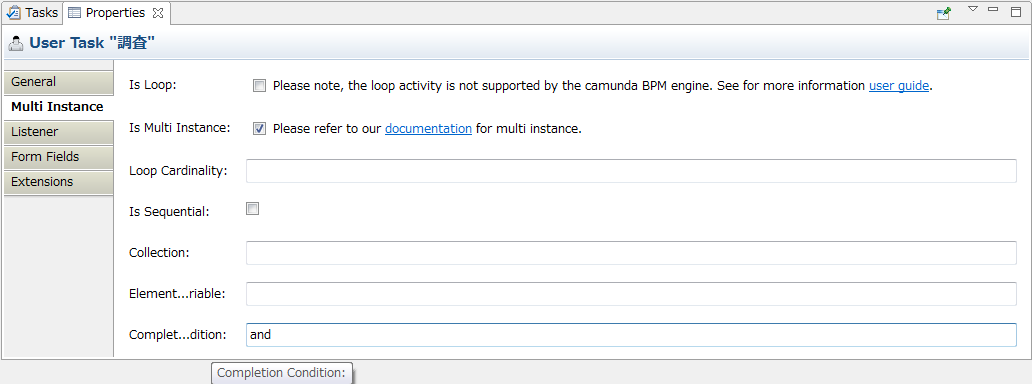
**Sequential（順次）**

**Parallel（並行）**

**Sequential（順次）の場合、チェックボックスをONにする**

マルチインスタンスタスクの場合、以下２つの「タスク完了条件（Completion Condition）」を設定することが可能。

* 割り当てたユーザ全員が承認したら次に進む（AND承認）
* 割り当てたユーザのうち、誰かが承認したら次に進む（OR承認）



**全員承認で次に進む場合、and と設定**

**誰か一人が承認した時、次に進む場合、or(1) と設定**

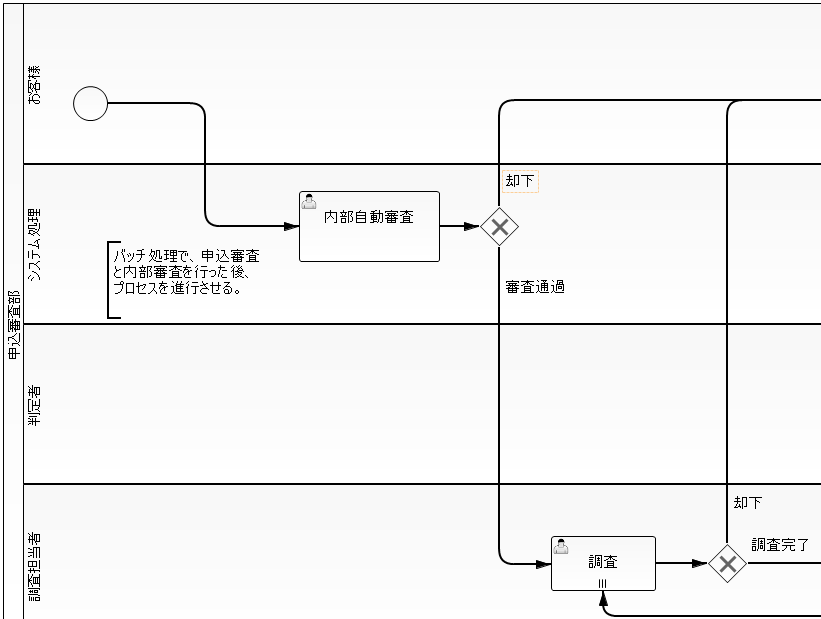
**二人以上承認した時など、人数を指定することも可能　[例] or(2)**



### バッチ処理で実行するタスク

ワークフロー上のタスクをバッチ処理によって実行する場合でも、「ユーザタスク」の要素を使って表現可能。

例えば、「日中にお客様から申し込みされたデータを夜間バッチで審査し、フローを進行する」という処理をワークフロー上に定義したい場合、以下のように記述する。



「内部自動審査」は、バッチ処理。

この例では、日中にお客様から申込された時点で、ワークフローが開始される。

夜間バッチ処理では、開始済みの複数のワークフローに対して審査処理を行い、

審査結果をもとにフローを次に進めるよう設計する。



“審査結果”に応じて、審査通過/却下のどちらに進ませるかについては、

標準表記の進行条件を使い「審査通過：eq(condition, 0)」「却下：eq(condition, 1)」

と設定できるが、進行条件判定用のコンポーネントを用意し、そのコンポーネントに

判定を委ねることも可能。

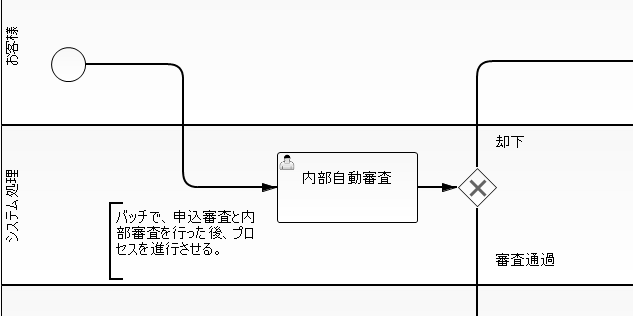
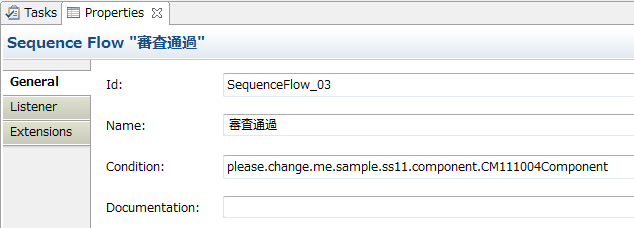
詳細は、「4.3.3標準表記と異なる進行条件」を参照。



### 標準表記と異なる進行条件

「[承認][却下]など、画面上のボタン押下とフローの進行が単純に結びついているケース」や、「入力金額の上限によるフローの進行ケース」などの単純なケースは、標準表記（「eq(変数,期待値)」など）を使い、進行条件を設定できる。

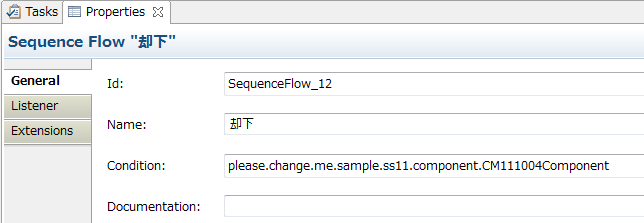
しかし、「DBの情報を参照し、業務的なロジックを通じて進行方向を判定するようなケース」では、進行条件判定用のコンポーネントを用意し、そのコンポーネントに判定を委ねることも可能。



**・進行条件判定用コンポーネントの「完全修飾名」を設定すること**

**・却下、審査通過ともに同じコンポーネントで条件判定したいため、**

**「同じ進行条件判定コンポーネント名」を設定すること**



[補足]

・複数のワークフロー定義上で、同じ進行条件判定が使える場合、「進行条件判定用のコンポーネント化」は有効。

・進行条件判定用コンポーネントは、「業務コンポーネント」と同様に設計する。

# 承認履歴/承認状況（ステータス）に関する設計

Nablarchワークフローライブラリ側では、申請したデータの「承認状況（ステータス）」や、いつ誰がどのタスクを実行したかを記録する「承認履歴」を管理していない。

理由は、交通費申請、休暇申請など、申請の種類ごとに個別のステータスで管理したいケース、全ての申請を同じステータスで管理したいケースなど、PJ要件に依存するため。

よって、PJ要件に合わせて、「承認状況（ステータス）」「承認履歴」に関する設計を行うこと。

## 承認履歴

「申請の種類ごとに個別の承認履歴を管理したい」「内部統制や監査に必要な証跡として承認履歴を管理したい」など、

PJ要件に合わせて、「承認履歴」の管理方法について設計すること。

## 承認状況（ステータス）

ワークフローの進行状況をどのように見せるかについて、設計する必要がある。

■全ての申請種類について、同じ見せ方をしたいケースの例

「申請中」「差戻し中」「引き戻し中」「取消」「却下」「承認」

■申請の種類ごとに個別の見せ方をしたいケースの例

「自動一時審査待ち」「調査待ち」「調査結果判定待ち」「上位判定待ち」「審査合格」

[注意]

画面上のボタン（[引戻し][申請][承認][却下]など）に対する表示/非表示制御の判断に「承認状況（ステータス）」は利用せず、「アクティブフローノード（アクティブなタスク）」を利用すること。

また、[承認]ボタン等の押下時処理における分岐条件も同様に、「承認状況（ステータス）」は利用せず、「アクティブフローノード（アクティブなタスク）」を利用すること。

# 画面設計およびバッチ設計との関連

ワークフロー上で、「利用者が申請や承認行為を行う画面」や「一括して承認/却下の自動判定を行うバッチ処理」は、通常の画面設計/バッチ設計と同様の作業を行う必要がある。

しかし、「どのワークフロー定義に基づいて、申請を開始するのか？」や「ワークフロー上のタスクにどのユーザを担当として割り当てるか？」等の情報は、ワークフローの設計情報として明確にする必要がある。

ワークフローアプリケーションの実装に必要な設計情報は、以下のとおり。

|  |  |
| --- | --- |
| 「ワークフローの開始」に関する設計情報 | ワークフローの「開始タイミング」および「ワークフローID」を明確にする必要がある。  ・開始タイミング：　『交通費申請画面上の[申請]ボタン押下時処理』などが該当。  ・ワークフローID：　どのワークフロー定義に基づいてワークフローを開始するのかを決める情報。    　画面設計の場合、開始タイミングに該当する「画面イベント」に対象のワークフローIDおよびワークフロー名を記述する。  　バッチ設計の場合、「処理詳細」に対象のワークフローIDおよびワークフロー名を記述する。 |
| 「ワークフロー上のタスクに対する担当割り当て」に関する設計情報 | ワークフロー上の各タスクに対して、「どのユーザ」または「どのグループ」を割り当てるのかを明確にする必要がある。  ・ワークフローの開始時、全てのタスクにユーザまたはグループを割り当てることも可能。  ・フロー進行の都度、ユーザまたはグループを割り当てることも可能。  　画面設計の場合、開始タイミングに該当する「画面イベント」に“担当割り当てに関する設計情報”を記述する。  　バッチ設計の場合、「処理詳細」に“担当割り当てに関する設計情報”を記述する。 |
| 「ワークフローの進行」に関する設計情報 | 承認者が、申請内容を画面上で確認し、[承認][却下][差戻し]ボタンを押下する場合など、「ワークフローを進行させるタイミング」および「進行条件の判定に使うパラメータ」を明確にする必要がある。  ・ワークフローを進行させるタイミング：　承認者側は、[承認][却下][差戻し]など、申請者側は、[引戻し][取消]などが該当。  ・進行条件の判定に使うパラメータ：　「画面入力された決裁金額」などが該当。  　画面設計の場合、開始タイミングに該当する「画面イベント」に“フロー進行に関する設計情報”を記述する。  　バッチ設計の場合、「処理詳細」に“フロー進行に関する設計情報”を記述する。 |