目次

改訂情報

変更年月日	変更内容
2012-10-01	初版
2012-12-21	第２版　下記を追加・変更しました ■ 「 *リクエストパラメータ* 」の 「imwSerialProcParams」 に関する説明を追加・修正しました。 ■ 「 *実装例* 」にコールバック関数が受け取れる情報について説明を追加しました。 ■ 「 *特記事項* 」、「 *IM-Workflow バージョン8.0.2 における改善* 」を追加しました。 ■ 「 *処理完了後の画面遷移* 」章の見出しを変更しました。また 「 *処理完了後の画面遷移* 」章以下の章立てを見直し、説明を追加しました。 　■ 変更前： 処理画面から受け取るリクエストパラメータ 　■ 変更後： 処理完了後の画面遷移 ■ 「 *ユーザコンテンツと連続処理／連続確認の連携方法* 」を追加しました。
2013-04-01	第３版　下記を追加・変更しました ■ 「 *前提条件* 」を追加しました。 ■ 「 *リクエストパラメータ* 」 に、スマートフォン用画面の説明を追記しました。 ■ 「 *画面の作成* 」 に、スマートフォン用画面の説明を追記しました。 ■ 「 *画面の作成* 」 の制限事項に、章の説明を追記しました。 ■ 「 *画面* 」 に、スマートフォン用画面の説明を追記しました。 ■ 「 *呼び出し画面の初期表示値指定* 」 に、スマートフォン用画面の説明を追記しました。 ■ 「 *画面入力情報の保持* 」 に、スマートフォン用画面の説明を追記しました。 ■ 「 *呼び出し画面からのコールバック関数の指定* 」 に、スマートフォン用画面の説明を追記しました。 ■ 「 *処理完了後の画面遷移* 」 に、スマートフォン用画面の説明を追記しました。 ■ 「 *ユーザコンテンツと連続処理／連続確認の連携方法* 」 に、スマートフォン用画面の説明を追記しました。 ■ 「 *PC版ユーザコンテンツをスマートフォン用画面としても利用する* 」 を追加しました。 ■ 上記のほか、誤字脱字などを修正しました。
2013-07-01	第４版　下記を追加・変更しました ■ 「 *実装例* 」の実装サンプル記述を修正しました。 ■ 「 *標準画面を非同期で実行する場合の注意点* 」を追加しました。
2013-10-01	第５版　下記を追加・変更しました ■ 「 *リクエストパラメータ* 」の記述を修正しました。 　■ 「imwAuthUserCode」 に関するただし書きを削除しました。
2014-01-01	第６版　下記を追加・変更しました ■ サンプルjavaソースのプログラムパスを修正しました。 ■ 「 *リクエストパラメータ* 」の記述を修正しました。 　■ 「imwGroupId」 が非推奨である旨を記述しました。 ■ 「 *ユーザコンテンツ画面への不正な直接アクセスを抑止する* 」を追加しました。
2014-04-01	第７版　下記を追加・変更しました ■ 「 *案件処理系APIと画面動作仕様の違い* 」を追加しました。

変更年月日	変更内容
2014-08-01	第８版　下記を追加・変更しました ■ 「 *リクエストパラメータ* 」に再申請時のimwAuthUserCodeに関する説明を追加しました。
2014-09-01	第９版　下記を追加・変更しました ■ 「 *画面の作成* 」に開発モデル（パス種別）に対する説明を追加しました。 ■ 「 *詳細画面の呼び出し* 」を追加しました。 ■ 「 *処理対象者プラグインの構成* 」の説明を追加しました。
2014-12-01	第１０版　下記を追加・変更しました ■ 「 *リクエストパラメータ* 」の 「imwSysDateTargetExpandFlag」 に関する説明を追加・修正しました。
2014-12-24	第１１版　下記を追加・変更しました ■ 「 *意図しないURLに対するバリデーション* 」を追加しました。 ■ 「 *動的処理対象者設定機能* 」を追加しました。
2015-04-01	第１２版　下記を追加・変更しました ■ 「 *リクエストパラメータ* 」に 「imwShortCutFlag」 を追加しました。 ■ 「 *意図しないURLに対するバリデーション* 」に関する説明を修正しました。
2015-08-01	第１３版　下記を追加・変更しました ■ 「 *案件終了処理（トランザクションなし）* 」を追加しました。 ■ 「 *テンプレート* 」に「案件終了処理（トランザクションなし）」を追加しました。 ■ 「 *動的処理対象者設定機能* 」にスマートフォン版の説明を追加しました。
2015-12-01	第１４版　下記を追加・変更しました ■ 「 *案件終了処理（トランザクションなし）* 」に、エラー発生時のロールバックに関する説明を追加しました。
2016-04-01	第１５版　下記を追加・変更しました ■ 「 *画面の作成* 」に、 TERASOLUNA Server Framework for Java (5.x) に関する説明を追加しました。 ■ 「 *画面の作成* 」の制限事項を削除し、リリースノート に記載しました。 ■ 「 *ユーザプログラムの作成* 」に、パラメータと返却値に関する説明を追加しました。 ■ 連携先に TERASOLUNA Server Framework for Java (5.x) を追加しました。
2016-08-01	第１６版　下記を追加・変更しました ■ 「 *案件終了処理* 」の戻り値の説明を変更しました。 ■ 「 *到達処理* 」の戻り値の説明を変更しました。 ■ 「 *案件終了処理（トランザクションなし）* 」の戻り値の説明を変更しました。 ■ 「 *リクエストパラメータ* 」のimwUserCode、imwNodeIdの説明を追加しました。
2017-04-01	第１７版　下記を追加・変更しました ■ 「 *到達処理* 」のパラメータ「メール送信可否」の説明を追加しました。 ■ 「 *画面* 」のThemeBuilderの指定内容を修正しました。

変更年月日	変更内容
2017-08-01	第１８版　下記を追加・変更しました ■ 「 *案件終了処理* 」のパラメータ「メール送信可否」の説明を追加しました。 ■ 以下のユーザプログラムに対し、複数のユーザプログラムを設定されている際にエラーが発生した場合の動作の説明を追加しました。 　■ *案件開始処理* 　■ *案件終了処理* 　■ *アクション処理* 　■ *案件終了処理（トランザクションなし）* ■ 以下のユーザプログラムに対し、resultFlagの説明を修正しました。 　■ *案件開始処理* 　■ *案件終了処理* 　■ *アクション処理* 　■ *到達処理* 　■ *分岐開始処理* 　■ *案件終了処理（トランザクションなし）* ■ 「 *処理対象者プラグインの作成* 」の説明を変更しました。
2017-12-01	第１９版　下記を追加・変更しました ■ 以下のページに「動的処理対象者設定」におけるノードスキップの方法を追加しました。 　■ *動的承認ノード・確認ノード* 　■ *横配置ノード・縦配置ノード* ■ 「 *アクション処理* 」にアクション処理におけるユーザパラメータの説明を追加しました。

はじめに

目的

本書は、IM-Workflow で利用することが可能な画面およびモジュールを作成する方法について説明します。

本書は、IM-Workflow の機能を使用する方法を記述しています。

本書で使用するサンプルプログラムはあくまでも、IM-Workflow の機能およびAPI等の使用方法を理解することに主眼をおいています。
そのため、必ずしもベストなコーディング方法とはいえない方法もあえて取っている個所があります。
あくまでも、サンプルとしての位置付けでとらえるようにしてください。

前提条件

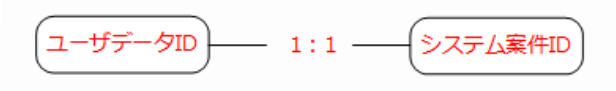
本書に記述されているサンプルプログラムは、JavaEE開発モデルおよびスクリプト開発モデルで記述されています。
そのため、JavaEE開発モデルおよびスクリプト開発モデルに関する理解は必須です。
各開発モデルに関しては、付属する各種マニュアルおよびAPIリストを参照してください。
本書を理解するには、基本的な IM-Workflow に関する理解が必要です。付属する各種マニュアル、APIリスト、および制限事項を参照してください。
本書に記述されているサンプルプログラムのパスは、以下のディレクトリ配下のパスです。
＜（展開したwar）/WEB-INF/＞

準備

IM-Workflow のサンプルプログラムを実行するための準備をします。
「intra-mart Accel Platform セットアップガイド」を参考に、IM-Workflow が動作する環境を構築します。
製品のインストール後は、システム管理者でログインし、メニュー[テナント環境セットアップ]より、テナント環境セットアップを行い、 サンプルデータセットアップ も必ず行ってください。
本書に記述されているJavaEE開発モデルの[javaファイル]は、配置する場所を示します。
実際に配置されているファイルは、[classファイル]です。
JavaEE開発モデル[javaファイル]のサンプルプログラムについては、製品メディアに保存されています。
また、製品最新情報ダウンロードページ（ Products Information Site ）から入手することもできます。

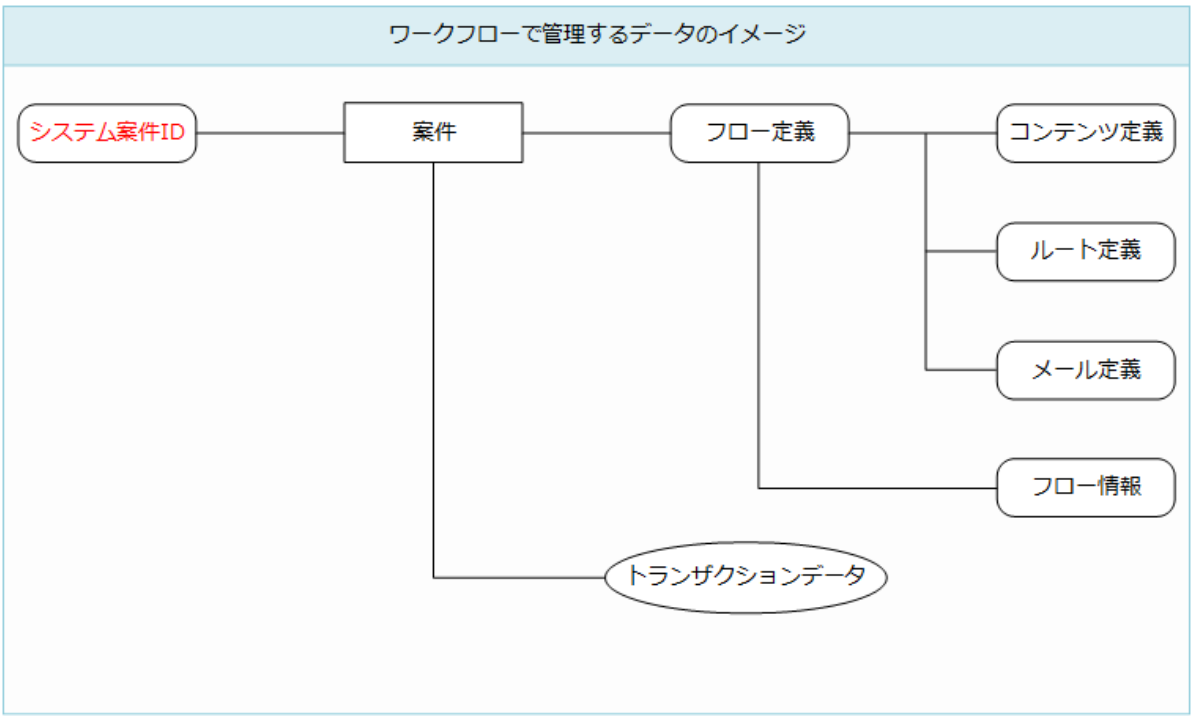
ユーザアプリケーションと IM-Workflow の関係

ユーザアプリケーションデータと IM-Workflow のデータは、それぞれ”ユーザデータID“と“システム案件ID”という2つのキーによって一意に特定されます。
2つのキーは1対1の関係で関連付けられます。



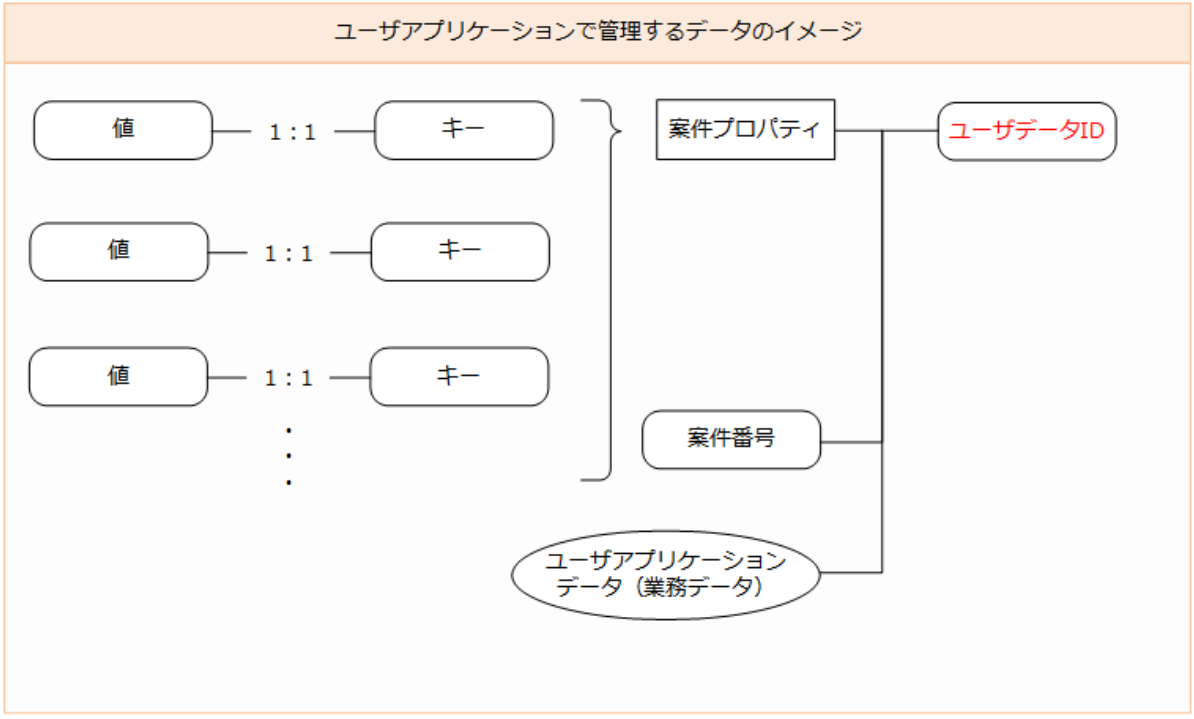
システム案件ID

システム案件IDとは、IM-Workflow において一意となるキーです。
IM-Workflow のモジュールにおいて採番され、外部より指定することはできません。
システム案件IDは、IM-Workflow のAPIやタグライブラリ等で案件を特定するために使用され、画面等に表示されることはありません。



ユーザデータID

ユーザデータIDとは、ユーザアプリケーション側で一意となるキーとして、ユーザアプリケーションで採番するキーです。
申請または起票を行う場合に、 IM-Workflow の提供するAPIおよびタグライブラリの引数として渡されます。
ユーザデータIDは、システム案件IDと同様に、 IM-Workflow のAPIやタグライブラリ等で案件を特定するために使用され、画面等に表示されることはありません。



案件プロパティ

案件プロパティとは、いわゆるKey-Value Storeです。「Key（キー）」と「Value（値）」のペアからなるデータモデルを案件単位に IM-Workflow で保存します。
IM-Workflow が提供するAPIを通じて、任意のタイミングにおいて、登録・更新・削除および取得が可能です。
また、 IM-Workflow が提供する各種一覧画面に表示したり、分岐条件におけるルール定義で参照する値として使用することができます。

リクエストパラメータ

各種一覧画面から呼び出される申請および処理等の画面で、必要な情報をリクエストパラメータとして受け取る事ができます。

No	パラメータ（物理名）	パラメータ（論理名）	詳細
1	imwGroupId	グループID	非推奨です。 過去との互換のために残されています。
2	imwUserCode	処理者CD（ログイン ユーザ本人のユーザコード）	
3	imwPageType	画面種別	表示された画面の種別 ・申請画面 ・一時保存画面 ・申請（起票案件）画面 ・再申請画面 ・処理画面 ・確認画面 ・処理詳細 ・参照詳細 ・確認詳細 ・過去案件詳細 ・申請画面（スマートフォン用） ・一時保存画面（スマートフォン用） ・申請（起票案件）画面（スマートフォン用） ・再申請画面（スマートフォン用） ・処理画面（スマートフォン用） ・確認画面（スマートフォン用）

No	パラメータ（物理名）	パラメータ（論理名）	詳細
4	imwUserDataId	ユーザデータID	
5	imwSystemMatterId	システム案件ID	
6	imwNodeId	処理対象ノードID	処理対象のノードが縦配置・横配置ノードの場合は、展開後のノードIDが渡されます。
7	imwArriveType	到達種別	
8	imwAuthUserCode	権限者CD	ログインユーザが案件を処理する際に選択可能な権限者コードです。具体的には、ログインユーザ本人や、ログインユーザを代理先として代理設定されている場合は代理元ユーザコードが該当します。 権限者が複数存在する場合、当パラメータは配列で渡されます。※ ただし、権限者が複数存在する場合でも、申請/一時保存画面表示の際は一覧上で権限者が特定されているため、特定済みの権限者コードのみが渡されます。
9	imwApplyBaseDate	申請基準日	「yyyy/mm/dd」形式
10	imwContentsId	コンテンツID	
11	imwContentsVersionId	コンテンツバージョンID	
12	imwRouteId	ルートID	
13	imwRouteVersionId	ルートバージョンID	
14	imwFlowId	フローID	
15	imwFlowVersionId	フローバージョンID	
16	imwSerialProcParams	連続処理パラメータ	連続処理用のパラメータ **IM-Workflow バージョン8.0.2 より、当パラメータは無効になりました。** 必ず空文字（””）が渡されるため、ユーザコンテンツ間での当パラメータの引き回しは不要です。 連続処理用の情報は「imwCallOriginalParams」に内包されます。
17	imwCallOriginalParams	呼出元パラメータ	呼出元ページのパラメータ
18	imwCallOriginalPagePath	呼出元ページパス	呼出元のページパス
19	imwSysDateTargetExpandFlag	システム日で対象者を展開するフラグ	“0”：無効、”1”：有効
20	imwShortCutFlag	ショートカットフラグ	ショートカットアクセスURLから詳細画面が表示されたことを示すフラグです。 ※IM-Workflow バージョン8.0.10より追加されました。 “0”：ショートカットアクセスURLからの遷移ではない “1”：ショートカットアクセスURLからの遷移

※imwAuthUserCode（権限者CD）について、各開発モデルでの取得例は以下の通りです。
ここで記載している内容は、次の観点において共通です。

- クライアントタイプ

スクリプト開発モデル

```
function init(request) {
   var imwAuthUserCodeList = request.getParameterValues("imwAuthUserCode"); //権限者CDの配列
}
```

JavaEE開発モデル

```
HttpServletRequest request = getRequest();
String[] imwAuthUserCodeList = request.getParameterValues("imwAuthUserCode"); //権限者CDの配列
```

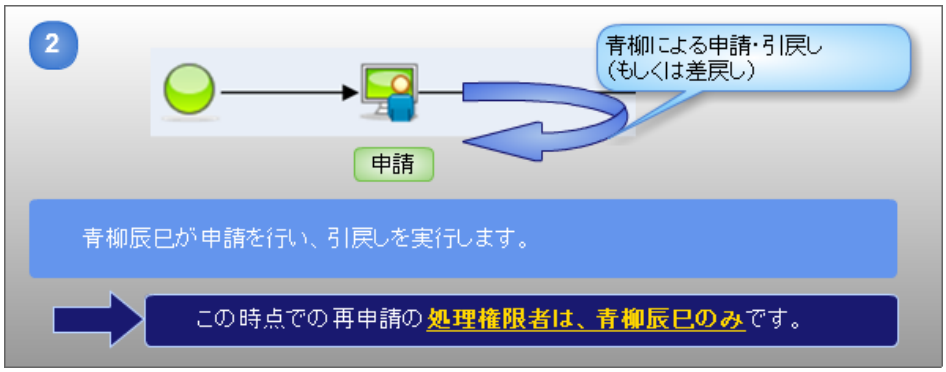
画面別取得可否一覧

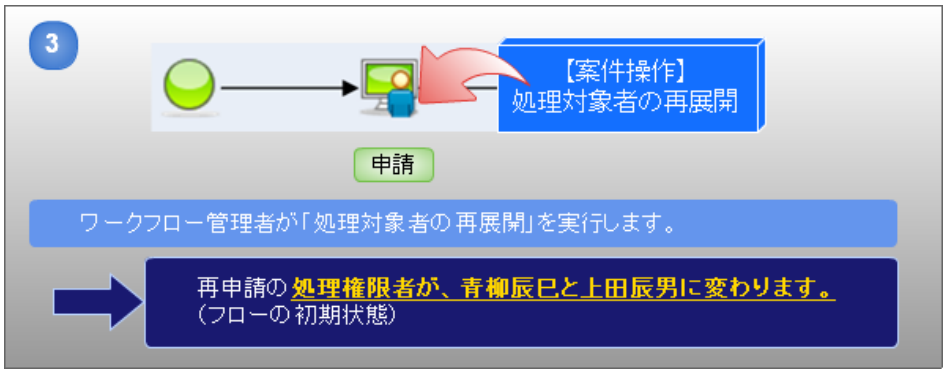
No	パラメータ	申請※	一時保存※	起票※	再申請※	処理※	確認※	処理詳細	参照詳細	確認詳細	過去案件詳細
1	imwGroupId	○	○	○	○	○	○	○	○	○	○
2	imwUserCode	○	○	○	○	○	○	○	○	○	○
3	imwPageType	○	○	○	○	○	○	○	○	○	○
4	imwUserDataId	-	○	○	○	○	○	○	○	○	○
5	imwSystemMatterId	-	-	○	○	○	○	○	○	○	○
6	imwNodeId	○	○	○	○	○	○	-	-	-	-
7	imwArriveType	○	○	○	○	○	-	-	-	-	-
8	imwAuthUserCode	○	○	○	○	○	-	-	-	-	-
9	imwApplyBaseDate	○	○	○	○	○	○	○	○	○	○
10	imwFlowId	○	○	○	○	○	○	○	○	○	○
11	imwFlowVersionId	○	○	○	○	○	○	○	○	○	○
12	imwContentsId	○	○	○	○	○	○	○	○	○	○
13	imwContentsVersionId	○	○	○	○	○	○	○	○	○	○
14	imwRouteId	○	○	○	○	○	○	○	○	○	○
15	imwRouteVersionId	○	○	○	○	○	○	○	○	○	○
16	imwSerialProcParams	-	-	○ -	○ -	○ -	○ -	-	-	-	-
17	imwCallOriginalParams	○	○	○	○	○	○	-	-	-	-
18	imwCallOriginalPagePath	○	○	○	○	○	○	-	-	-	-
19	imwSysDateTargetExpandFalg	○	○	○	○	○	○	○	○	○	-
20	imwShortCutFlag	-	-	-	-	-	-	○	○	○	○

＜「○」 ： 取得可能 ／ 「-」 ： 取得不可能＞

※スマートフォン用の場合も同様です。
ただし、「imwAuthUserCode」 のみ、スマートフォン用の起票、再申請、処理画面では取得することができません。

- **再申請時の「imwAuthUserCode」**
  リクエストパラメータの「imwAuthUserCode」は、基本的には特定の1ユーザコードを返却しますが、以下の図の例のような操作を行った場合には、複数のユーザコードを返却します。





案件処理系APIと画面動作仕様の違い

画面上からの操作とは異なり、案件処理系API（Webサービスを含む）を直接利用して案件の処理を行う場合は、業務的なチェックを行わずに処理が実行されます。
ここでいう業務的なチェックとは、以下のようなチェックを指します。

- API引数として指定した処理権限者が、到達処理で展開された処理対象者に含まれるか
- API引数として指定した処理権限者と処理実行者が異なる場合、両者間で有効な代理設定が存在するか

画面上からの操作と同等の機能を、APIを利用して独自に実装する場合は、案件処理APIの実行前に各種API（処理権限判定APIなど）を併せて利用してください。

画面の作成

この章では、IM-Workflow が提供する案件の各処理画面と連携するための画面実装の基本部分について、「画面種別」、「開発モデル」、「クライアントタイプ」の観点で説明します。

上記観点の内訳は、次のとおりです。

- 画面種別
  - 申請画面
  - 一時保存画面
  - 申請（起票案件）画面
  - 再申請画面
  - 処理画面
  - 確認画面
  - 処理詳細
  - 参照詳細
  - 確認詳細
  - 過去案件詳細
- 開発モデル
  - スクリプト開発モデル

    （「パス種別：スクリプト開発モデル」で利用するユーザコンテンツ画面）
  - JavaEE開発モデル

    （「パス種別：JavaEE開発モデル」で利用するユーザコンテンツ画面）
  - JSP、 TERASOLUNA Server Framework for Java (5.x)

    （「パス種別：JSP or Servlet」で利用するユーザコンテンツ画面）
- クライアントタイプ
  - PC
  - スマートフォン

また、画面作成における応用実装について、「*カスタマイズ*」で説明しています。
必要に応じて参照してください。

申請画面の呼び出し

IM-Workflow で提供する申請を行うための画面（以下、申請画面）と連携する方法を説明します。

Contents

申請画面を表示するためには、IM-Workflow が提供するタグライブラリおよびClient-side JavaScript APIを使用します。

スクリプト開発モデル

IM-Workflow 用のタグライブラリの使用方法については、「APIドキュメント」も併せて参照してください。

PC用画面の場合

申請画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記のIMARTタグを記述します。

```
<imart type="head">

<imart type="workflowOpenPageCsjs" />

</imart>
```

申請画面と連携する画面のボディ部に、下記のIMARTタグを記述します。

IMARTタグに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。
「imwUserDataId」は、申請一覧画面からのリクエストパラメータには含まれません。
ファンクション・コンテナで採番する必要があります。

```
<imart type="workflowOpenPage"
    name="applyForm"
    id="applyForm"
    method="POST"
    target="_top"
    imwUserDataId=oRequest.imwUserDataId
    imwAuthUserCode=oRequest.imwAuthUserCode
    imwApplyBaseDate=oRequest.imwApplyBaseDate
    imwNodeId=oRequest.imwNodeId
    imwFlowId=oRequest.imwFlowId>
</imart>
```

下記のClient-side JavaScript APIを実行することにより、申請画面が表示されます。

```
<script type="text/javascript">

  workflowOpenPage('0');

</script>
```

スマートフォン用画面の場合

申請画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記のIMARTタグを記述します。

```
<imart type="head">

<imart type="spWorkflowOpenPageCsjs" />

</imart>
```

申請画面と連携する画面のボディ部に、下記のIMARTタグを記述します。

IMARTタグに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。
「imwUserDataId」は、申請一覧画面からのリクエストパラメータには含まれません。
ファンクション・コンテナで採番する必要があります。

```
<imart type="spWorkflowOpenPage"
    name="applyForm"
    id="workflowOpenPageForm"
    method="POST"
    target="_top"
    imwUserDataId=$data.imwUserDataId
    imwAuthUserCode=$data.imwAuthUserCode
    imwApplyBaseDate=$data.imwApplyBaseDate
    imwNodeId=$data.imwNodeId
    imwFlowId=$data.imwFlowId>
</imart>
```

下記のClient-side JavaScript APIを実行することにより、申請画面が表示されます。

```
<script type="text/javascript">

   workflowOpenPage4Sp('10');

</script>
```

JavaEE開発モデル

IM-Workflow 用のタグライブラリの使用方法については、「APIドキュメント」も併せて参照してください。

PC用画面の場合

申請画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:workflowOpenPageCsjs />

</imui:head>
```

申請画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。
「imwUserDataId」は、申請一覧画面からのリクエストパラメータには含まれません。
ServiceControllerなどで採番する必要があります。

```
<workflow:workflowOpenPage
        name="applyForm"
        id="applyForm"
        method="POST"
        target="_top"
        imwUserDataId='<%=(String)request.getAttribute("imwUserDataId")%>'
        imwAuthUserCode='<%=(String)request.getAttribute("imwAuthUserCode")%>'
        imwApplyBaseDate='<%=(String)request.getAttribute("imwApplyBaseDate")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'
        imwFlowId='<%=(String)request.getAttribute("imwFlowId")%>'>
</workflow:workflowOpenPage>
```

下記のClient-side JavaScript APIを実行することにより、申請画面が表示されます。

```
<script type="text/javascript">

   workflowOpenPage('0');

</script>
```

スマートフォン用画面の場合

申請画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:spWorkflowOpenPageCsjs />

</imui:head>
```

申請画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。
「imwUserDataId」は、申請一覧画面からのリクエストパラメータには含まれません。
ServiceControllerなどで採番する必要があります。

```
<workflow:spWorkflowOpenPage
        name="applyForm"
        id="applyForm"
        method="POST"
        target="_top"
        imwUserDataId='<%=(String)request.getAttribute("imwUserDataId")%>'
        imwAuthUserCode='<%=(String)request.getAttribute("imwAuthUserCode")%>'
        imwApplyBaseDate='<%=(String)request.getAttribute("imwApplyBaseDate")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'
        imwFlowId='<%=(String)request.getAttribute("imwFlowId")%>'>
</ workflow:spWorkflowOpenPage>
```

下記のClient-side JavaScript APIを実行することにより、申請画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage4Sp('10');

</script>
```

JSP、 TERASOLUNA Server Framework for Java (5.x)

JavaEE開発モデルに準じます。
TERASOLUNA Server Framework for Java (5.x) を利用して実装する場合には、「IM-Workflow TERASOLUNA Server Framework プログラミングガイド 」も併せて参照してください。

一時保存画面の呼び出し

IM-Workflow で提供する一時保存を行うための画面（以下、一時保存画面）と連携する方法を説明します。

Contents

一時保存画面を表示するためには、IM-Workflow が提供するタグライブラリおよびClient-side JavaScript APIを使用します。

スクリプト開発モデル

IM-Workflow 用のタグライブラリの使用方法については、「APIドキュメント」も併せて参照してください。

PC用画面の場合

一時保存画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記のIMARTタグを記述します。

```
<imart type="head">

<imart type="workflowOpenPageCsjs" />

</imart>
```

一時保存画面と連携する画面のボディ部に、下記のIMARTタグを記述します。

IMARTタグに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="workflowOpenPage"
      name="tempForm"
      id="tempForm"
      method="POST"
      target="_top"
      imwUserDataId=oRequest.imwUserDataId
      imwAuthUserCode=oRequest.imwAuthUserCode
      imwApplyBaseDate=oRequest.imwApplyBaseDate
      imwNodeId=oRequest.imwNodeId
      imwFlowId=oRequest.imwFlowId>
</imart>
```

下記のClient-side JavaScript APIを実行することにより、一時保存画面が表示されます。

```
<script type="text/javascript">

   workflowOpenPage('1');

</script>
```

スマートフォン用画面の場合

一時保存画面と連携する画面のヘッダ部（<imart type=“head”> ～ </imart>）に、下記のIMARTタグを記述します。

```
<imart type=“head”>

<imart type="spWorkflowOpenPageCsjs" />

</imart>
```

一時保存画面と連携する画面のボディ部に、下記のIMARTタグを記述します。

IMARTタグに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="spWorkflowOpenPage"
      name="tempForm"
      id="tempForm"
      method="POST"
      target="_top"
      imwUserDataId=$data.imwUserDataId
      imwAuthUserCode=$data.imwAuthUserCode
      imwApplyBaseDate=$data.imwApplyBaseDate
      imwNodeId=$data.imwNodeId
      imwFlowId=$data.imwFlowId>
</imart>
```

下記のClient-side JavaScript APIを実行することにより、一時保存画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage4Sp('11');

</script>
```

JavaEE開発モデル

IM-Workflow 用のタグライブラリの使用方法については、「 APIドキュメント 」も併せて参照してください。

PC用画面の場合

一時保存画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:workflowOpenPageCsjs />

</imui:head>
```

一時保存画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:workflowOpenPage
        name="tempForm"
        id="tempForm"
        method="POST"
        target="_top"
        imwUserDataId='<%=(String)request.getAttribute("imwUserDataId")%>'
        imwAuthUserCode='<%=(String)request.getAttribute("imwAuthUserCode")%>'
        imwApplyBaseDate='<%=(String)request.getAttribute("imwApplyBaseDate")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'
        imwFlowId='<%=(String)request.getAttribute("imwFlowId")%>'>
</workflow:workflowOpenPage>
```

下記のClient-side JavaScript APIを実行することにより、一時保存画面が表示されます。

```
<script type="text/javascript">

   workflowOpenPage('1');

</script>
```

スマートフォン用画面の場合

一時保存画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:spWorkflowOpenPageCsjs />

</imui:head>
```

一時保存画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:spWorkflowOpenPage
        name="tempForm"
        id="tempForm"
        method="POST"
        target="_top"
        imwUserDataId='<%=(String)request.getAttribute("imwUserDataId")%>'
        imwAuthUserCode='<%=(String)request.getAttribute("imwAuthUserCode")%>'
        imwApplyBaseDate='<%=(String)request.getAttribute("imwApplyBaseDate")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'
        imwFlowId='<%=(String)request.getAttribute("imwFlowId")%>'>
</ workflow:spWorkflowOpenPage>
```

下記のClient-side JavaScript APIを実行することにより、一時保存画面が表示されます。

```
<script type="text/javascript">

    workflowOpenPage4Sp('11');

</script>
```

JSP、 TERASOLUNA Server Framework for Java (5.x)

JavaEE開発モデルに準じます。
TERASOLUNA Server Framework for Java (5.x) を利用して実装する場合には、「IM-Workflow TERASOLUNA Server Framework プログラミングガイド 」も併せて参照してください。

申請（起票案件）／再申請／処理画面の呼び出し

IM-Workflow で提供する申請（起票案件）／再申請／処理を行うための画面（以下、処理画面）と連携する方法を説明します。

Contents

処理画面を表示するためには、IM-Workflow が提供するタグライブラリおよびClient-side JavaScript APIを使用します。

スクリプト開発モデル

IM-Workflow 用のタグライブラリの使用方法については、「APIドキュメント」も併せて参照してください。

PC用画面の場合

処理画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記のIMARTタグを記述します。

```
<imart type="head">

<imart type="workflowOpenPageCsjs" />

</imart>
```

処理画面と連携する画面のボディ部に、下記のIMARTタグを記述します。

IMARTタグに指定する属性は、通常未処理一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="workflowOpenPage"
      name="approveForm"
     id="approveForm"
      method="POST"
      target="_top"
      imwSystemMatterId=$data.imwSystemMatterId
      imwNodeId=$data.imwNodeId >
</imart>
```

下記のClient-side JavaScript APIを実行することにより、処理画面が表示されます。

申請（起票案件）

```
<script type="text/javascript">

   workflowOpenPage('2');

</script>
```

再申請

```
<script type="text/javascript">

   workflowOpenPage('3');

</script>
```

処理

```
<script type="text/javascript">

   workflowOpenPage('4');

</script>
```

スマートフォン用画面の場合

処理画面と連携する画面のヘッダ部（<imart type=“head”> ～ </imart>）に、下記のIMARTタグを記述します。

```
<imart type="head">

<imart type="spWorkflowOpenPageCsjs" />

</imart>
```

処理画面と連携する画面のボディ部に、下記のIMARTタグを記述します。

IMARTタグに指定する属性は、通常未処理一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="spWorkflowOpenPage"
      name="approveForm"
     id="approveForm"
      method="POST"
      target="_top"
      imwSystemMatterId=$data.imwSystemMatterId
      imwNodeId=$data.imwNodeId >
</imart>
```

下記のClient-side JavaScript APIを実行することにより、処理画面が表示されます。

申請（起票案件）

```
<script type="text/javascript">

   workflowOpenPage4Sp('12');

</script>
```

再申請

```
<script type="text/javascript">

   workflowOpenPage4Sp('13');

</script>
```

処理

```
<script type="text/javascript">

   workflowOpenPage4Sp('14');

</script>
```

JavaEE開発モデル

IM-Workflow 用のタグライブラリの使用方法については、「APIドキュメント」も併せて参照してください。

PC用画面の場合

処理画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:workflowOpenPageCsjs />

</imui:head>
```

処理画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常未処理一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:workflowOpenPage
        name="approveForm"
        id="approveForm"
        method="POST"
        target="_top"
        imwSystemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'>
</workflow:workflowOpenPage>
```

下記のClient-side JavaScript APIを実行することにより、処理画面が表示されます。

申請（起票案件）

```
<script type="text/javascript">

   workflowOpenPage('2');

</script>
```

再申請

```
<script type="text/javascript">

   workflowOpenPage('3');

</script>
```

処理

```
<script type="text/javascript">

   workflowOpenPage('4');

</script>
```

スマートフォン用画面の場合

処理画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:spWorkflowOpenPageCsjs />

</imui:head>
```

処理画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常未処理一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:spWorkflowOpenPage
        name="approveForm"
        id="approveForm"
        method="POST"
        target="_top"
        imwSystemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'>
</workflow:spWorkflowOpenPage >
```

下記のClient-side JavaScript APIを実行することにより、処理画面が表示されます。

申請（起票案件）

```
<script type="text/javascript">

   workflowOpenPage4Sp('12');

</script>
```

再申請

```
<script type="text/javascript">

   workflowOpenPage4Sp('13');

</script>
```

処理

```
<script type="text/javascript">

   workflowOpenPage4Sp('14');

</script>
```

JSP、 TERASOLUNA Server Framework for Java (5.x)

JavaEE開発モデルに準じます。
TERASOLUNA Server Framework for Java (5.x) を利用して実装する場合には、「IM-Workflow TERASOLUNA Server Framework プログラミングガイド 」も併せて参照してください。

確認画面の呼び出し

IM-Workflow で提供する確認を行うための画面（以下、確認画面）と連携する方法を説明します。

Contents

確認画面を表示するためには、IM-Workflow が提供するタグライブラリおよびClient-side JavaScript APIを使用します。

スクリプト開発モデル

IM-Workflow 用のタグライブラリの使用方法については、「APIドキュメント」も併せて参照してください。

PC用画面の場合

確認画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記のIMARTタグを記述します。

```
<imart type="head">

<imart type="workflowOpenPageCsjs" />

</imart>
```

確認画面と連携する画面のボディ部に、下記のIMARTタグを記述します。

IMARTタグに指定する属性は、通常確認一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="workflowOpenPage"
    name="confirmForm"
    id="confirmForm"
    method="POST"
    target="_top"
    imwSystemMatterId=$data.imwSystemMatterId
    imwNodeId=$data.imwNodeId>
</imart>
```

下記のClient-side JavaScript APIを実行することにより、確認画面が表示されます。

```
<script type="text/javascript">

   workflowOpenPage('5');

</script>
```

スマートフォン用画面の場合

確認画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記のIMARTタグを記述します。

```
<imart type="head">

<imart type="spWorkflowOpenPageCsjs" />

</imart>
```

確認画面と連携する画面のボディ部に、下記のIMARTタグを記述します。

IMARTタグに指定する属性は、通常確認一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="spWorkflowOpenPage"
    name="confirmForm"
    id="confirmForm"
    method="POST"
    target="_top"
    imwSystemMatterId=$data.imwSystemMatterId
    imwNodeId=$data.imwNodeId>
</imart>
```

下記のClient-side JavaScript APIを実行することにより、確認画面が表示されます。

```
<script type="text/javascript">

   workflowOpenPage4Sp('15');

</script>
```

JavaEE開発モデル

IM-Workflow 用のタグライブラリの使用方法については、「APIドキュメント」も併せて参照してください。

PC用画面の場合

確認画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:workflowOpenPageCsjs />

</imui:head>
```

確認画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常確認一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:workflowOpenPage
        name="confirmForm"
        id="confirmForm"
        method="POST"
        target="_top"
        imwSystemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'>
</workflow:workflowOpenPage>
```

下記のClient-side JavaScript APIを実行することにより、確認画面が表示されます。

```
<script type="text/javascript">

   workflowOpenPage('5');

</script>
```

スマートフォン用画面の場合

確認画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:spWorkflowOpenPageCsjs />

</imui:head>
```

確認画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常確認一覧画面から取得したリクエストパラメータを指定します。

```
<workflow: spWorkflowOpenPage
        name="confirmForm"
        id="confirmForm"
        method="POST"
        target="_top"
        imwSystemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'>
</workflow:spWorkflowOpenPage>
```

下記のClient-side JavaScript APIを実行することにより、確認画面が表示されます。

```
<script type="text/javascript">

   workflowOpenPage4Sp('15');

</script>
```

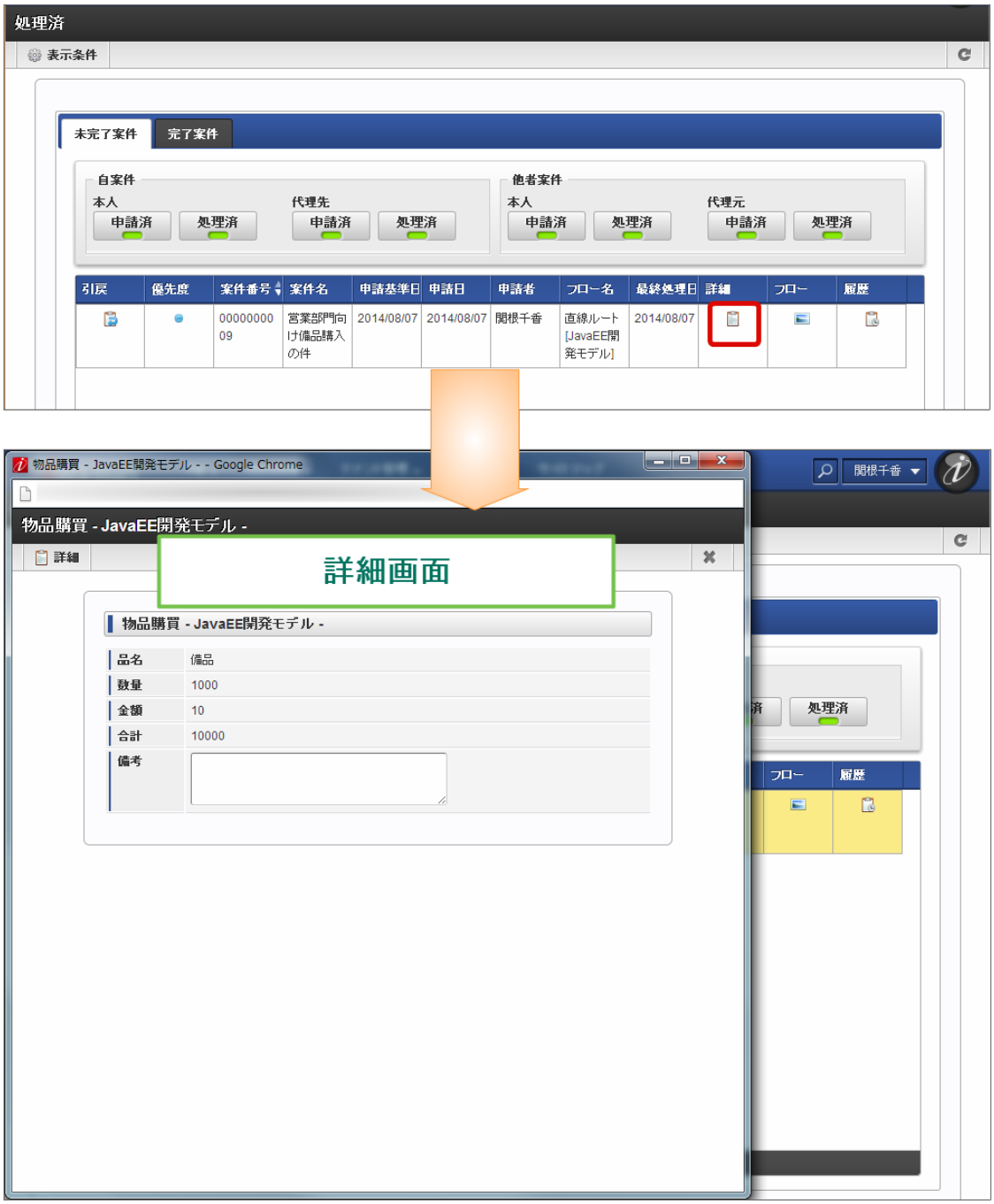
JSP、 TERASOLUNA Server Framework for Java (5.x)

JavaEE開発モデルに準じます。
TERASOLUNA Server Framework for Java (5.x) を利用して実装する場合には、「IM-Workflow TERASOLUNA Server Framework プログラミングガイド 」も併せて参照してください。

詳細画面の呼び出し

詳細画面は、他の処理画面と異なり、IM-Workflow が提供する標準処理画面との連携を行わないため、タグライブラリ「workflowOpenPageCsjs」および「workflowOpenPage」を実装する必要はありません。

呼び出し時の特徴として、詳細画面はポップアップで表示されます。



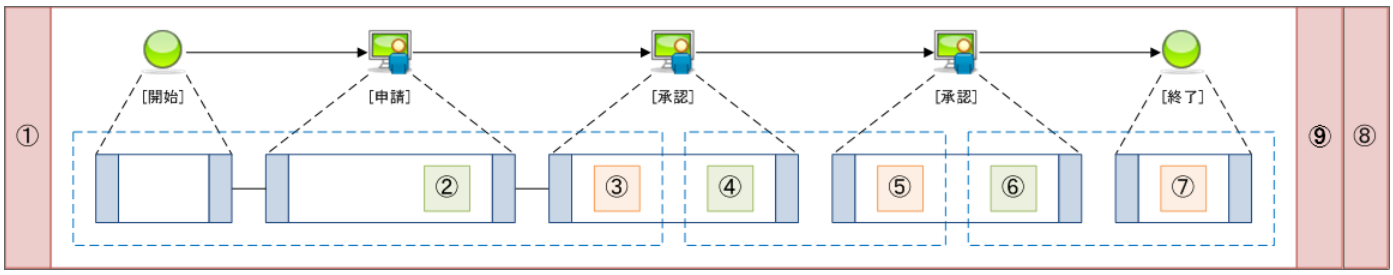
通常は、ポップアップ表示された詳細画面にも、呼び出し元画面と同様のヘッダ・フッタが表示されます。
ヘッダが表示されていると、別画面へのメニュー遷移や、ログアウト処理などが可能です。
これらの処理がポップアップ画面で行われた場合、呼び出し元画面での動作が不安定となる可能性があります。
そのため、詳細画面では、特別な理由がない限り、テーマのヘッダ・フッタを非表示とする対応をとることを推奨します。

テーマのヘッダ・フッタを非表示とするには、詳細画面のパスに対し、ページビルダー　HeadOnlyThemeBuilder を適用します。

- 設定場所：

```
<WEB-INF/conf/theme-head-only-path-config/*.xml>
```

- 設定方法の詳細については、「UIデザインガイドライン（ PC 版）」を参照してください。



No	処理名	項番
1	案件開始処理	①
2	案件終了処理	⑧
3	アクション処理	② ④ ⑥
4	到達処理	③ ⑤ ⑦
5	案件終了処理（トランザクションなし）	⑨

案件開始処理

Contents
- パラメータ
- 返却値

案件開始処理とは、案件が開始する時に、一度実行される処理です。下記の場合に実行されます。

- 申請者が”申請”を行った場合
- “起票”の案件を作成した場合（APIのみ）

案件開始処理は、 IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中ではDBトランザクション制御を行うことはできません。

パラメータ

案件開始処理では、以下の情報をパラメータとして受け取る事ができます。

No	パラメータ（物理名）	パラメータ（論理名）	属性	詳細
1	loginGroupId	ログイングループID	文字列	処理実行時のログイングループID ログイングループIDは、テナントIDと同値です。
2	localeId	ロケールID	文字列	処理実行時のロケールID
3	targetLocales	ターゲットロケールID	文字列[]	システムで利用しているロケールID
4	contentsId	コンテンツID	文字列	該当案件のコンテンツID
5	contentsVersionId	コンテンツバージョンID	文字列	該当案件のコンテンツバージョンID
6	routeId	ルートID	文字列	該当案件のルートID
7	routeVersionId	ルートバージョンID	文字列	該当案件のルートバージョンID
8	flowId	フローID	文字列	該当案件のフローID
9	flowVersionId	フローバージョンID	文字列	該当案件のフローバージョンID

No	パラメータ（物理名）	パラメータ（論理名）	属性	詳細
10	applyBaseDate	申請基準日	文字列	申請基準日（”yyyy/MM/dd”）
11	processDate	処理日	文字列	処理日（”yyyy/MM/dd”）
12	systemMatterId	システム案件ID	文字列	該当案件のシステム案件ID
13	userDataId	ユーザデータID	文字列	該当案件のユーザデータID
14	parameter	実行プログラムパス	文字列	該当処理の実行プログラムパス

返却値

案件開始処理では、以下の情報を返却します。（任意/必須・・・●：必須　△：任意）

- スクリプト開発モデル

No	返却値（物理名）	返却値（論理名）	属性	必須/任意	詳細
1	resultFlag	結果フラグ	真偽値	△	true：成功　false：失敗 指定しない場合、成功（true）として扱います。
2	message	結果メッセージ	文字列	△	結果フラグが失敗の場合のみ、設定します。

※ 結果メッセージを設定した場合、画面にエラーメッセージを表示します。

> **コラム**
>
> 結果フラグとして”false”（失敗）を設定した場合の動作は以下の通りです。
>
> - 該当の案件開始処理以外に設定されている後続の案件開始処理は実行されません。

- JavaEE開発モデル

  返却値はありません。

  ※「jp.co.intra_mart.foundation.workflow.exception.WorkflowExternalException」をスローした場合、引数に設定したエラーメッセージを画面に表示します。詳細は「 APIドキュメント 」を参照してください。

> **コラム**
>
> WorkflowExternalException をスローした場合の動作は以下の通りです。
>
> - 該当の案件開始処理以外に設定されている後続の案件開始処理は実行されません。

案件終了処理

Contents

案件終了処理とは、案件が終了する時に、一度実行される処理です。下記の場合に実行されます。

- 最後の承認者が“承認”を行った場合
- 承認者が“承認終了”を行った場合
- 承認者が“否認”を行った場合
- 申請者が“取止め”を行った場合
- 案件操作で終了ノードに到達した場合

案件終了処理は、直前のアクション処理や到達処理とは独立した処理（トランザクション）です。
そのため、案件終了処理でエラーが発生した場合、直前の処理を戻すこと（ロールバック）はできません。

案件終了処理は、 IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中ではDBトランザクション制御を行うことはできません。

パラメータ

案件終了処理では、以下の情報をパラメータとして受け取る事ができます。

No	パラメータ（物理名）	パラメータ（論理名）	属性	詳細
1	loginGroupId	ログイングループID	文字列	処理実行時のログイングループID ログイングループIDは、テナントIDと同値です。
2	localeId	ロケールID	文字列	処理実行時のロケールID
3	targetLocales	ターゲットロケールID	文字列[]	システムで利用しているロケールID
4	contentsId	コンテンツID	文字列	該当案件のコンテンツID
5	contentsVersionId	コンテンツバージョンID	文字列	該当案件のコンテンツバージョンID
6	routeId	ルートID	文字列	該当案件のルートID
7	routeVersionId	ルートバージョンID	文字列	該当案件のルートバージョンID
8	flowId	フローID	文字列	該当案件のフローID
9	flowVersionId	フローバージョンID	文字列	該当案件のフローバージョンID
10	applyBaseDate	申請基準日	文字列	申請基準日（”yyyy/MM/dd”）
11	processDate	処理日	文字列	処理日（”yyyy/MM/dd”）
12	systemMatterId	システム案件ID	文字列	該当案件のシステム案件ID
13	userDataId	ユーザデータID	文字列	該当案件のユーザデータID
14	parameter	実行プログラムパス	文字列	該当処理の実行プログラムパス
15	actFlag	代理フラグ	文字列	”0”：本人にて処理　”1”：代理者にて処理
16	lastProcessNodeId	最終処理ノードID	文字列	最終処理のノードID
17	lastAuthUserCd	最終処理権限者コード	文字列	最終処理の処理権限者コード
18	lastExecUserCd	最終処理実行者コード	文字列	最終処理の処理実行者コード
19	lastResultStatus	最終処理結果ステータス	文字列	最終処理の処理結果ステータス [1]
20	mailIds	メールテンプレートID	文字列[]	メール種別「処理結果通知」で設定されているメールID
21	imBoxIds	ImBoxId	文字列[]	IMBox種別「処理結果通知」で設定されているIMBoxID
22	mailReplaceMap	メール置換文字情報	Map	メールの置換文字情報 [2]
23	imBoxReplaceMap	IMBox置換文字情報	Map	IMBoxの置換文字情報 [2]

[1] ・・・コード値は、「 APIドキュメントのIM-Workflow CodeList 」を参照してください。

[2] *(1, 2)* ・・・置換文字列と置換内容の詳細は、「 IM-Workflow 仕様書 別紙 」を参照してください。

返却値

案件終了処理では、以下の情報を返却します。（任意/必須・・・●：必須　△：任意）

- スクリプト開発モデル

No	返却値（物理名）	返却値（論理名）	属性	必須/任意	詳細

No	返却値（物理名）	返却値（論理名）	属性	必須/任意	詳細
1	resultFlag	結果フラグ	真偽値	△	true：成功　false：失敗 指定しない場合、成功（true）として扱います。
2	message	結果メッセージ	文字列	△	結果フラグが失敗の場合のみ、設定します。
3	data	メール送信可否	真偽値	●	このパラメータは「処理結果通知」の送信制御を行います。 このパラメータによる制御はメールに限らず、IMBox などにも適用されます。 true：送信する　false：送信しない

※ 結果メッセージを設定した場合、例外ログにエラーメッセージを出力します。

> **コラム**
>
> 結果フラグとして”false”（失敗）を設定した場合の動作は以下の通りです。
>
> - 該当の案件終了処理以外に設定されている後続の案件終了処理は実行されません。
> - 処理結果通知を設定している場合、「メール送信可否」の設定値に関係なく、処理結果通知による通知は行われません。

- JavaEE開発モデル

No	返却値（論理名）	属性	必須/任意	詳細
1	メール送信可否	真偽値	●	このパラメータは「処理結果通知」の送信制御を行います。 このパラメータによる制御はメールに限らず、 IMBox などにも適用されます。 true：送信する　false：送信しない

※ 「jp.co.intra_mart.foundation.workflow.exception.WorkflowExternalException」をスローした場合、引数に設定したエラーメッセージを例外ログに出力します。詳細は「 APIドキュメント 」を参照してください。

> **コラム**
>
> WorkflowExternalException をスローした場合の動作は以下の通りです。
>
> - 該当の案件終了処理以外に設定されている後続の案件終了処理は実行されません。
> - 処理結果通知を設定している場合、「メール送信可否」の設定値に関係なく、処理結果通知による通知は行われません。

アクション処理

Contents

アクション処理とは、下記のような行為を行った場合に実行される処理です。

No	アクション	メソッド
1	申請	apply
2	再申請	reapply
3	申請（一時保存）	applyFromTempSave
4	申請（未処理）	applyFromUnapply

No	アクション	メソッド
5	取止め	discontinue
6	引戻し	pullBack
7	差戻し後引戻し	sendBackToPullBack
8	承認	approve
9	承認終了	approveEnd
10	否認	deny
11	差戻し	sendBack
12	保留	reserve
13	保留解除	reserveCancel
14	案件操作	matterHandle
15	一時保存（新規登録）	tempSaveCreate
16	一時保存（更新）	tempSaveUpdate
17	一時保存（削除）	tempSaveDelete

アクション処理は、IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中ではDBトランザクション制御を行うことはできません。

パラメータ

アクション処理では、以下の情報をパラメータとして受け取る事ができます。

No	パラメータ（物理名）	パラメータ（論理名）	属性	詳細
1	loginGroupId	ログイングループID	文字列	処理実行時のログイングループID ログイングループIDは、テナントIDと同値です。
2	localeId	ロケールID	文字列	処理実行時のロケールID
3	targetLocales	ターゲットロケールID	文字列[]	システムで利用しているロケールID
4	contentsId	コンテンツID	文字列	該当案件のコンテンツID
5	contentsVersionId	コンテンツバージョンID	文字列	該当案件のコンテンツバージョンID
6	routeId	ルートID	文字列	該当案件のルートID
7	routeVersionId	ルートバージョンID	文字列	該当案件のルートバージョンID
8	flowId	フローID	文字列	該当案件のフローID
9	flowVersionId	フローバージョンID	文字列	該当案件のフローバージョンID
10	applyBaseDate	申請基準日	文字列	申請基準日（”yyyy/MM/dd”）
11	processDate	処理日	文字列	処理日（”yyyy/MM/dd”）
12	systemMatterId	システム案件ID	文字列	該当案件のシステム案件ID
13	userDataId	ユーザデータID	文字列	該当案件のユーザデータID
14	matterName	案件名	文字列	該当案件の案件名
15	matterNumber	案件番号	文字列	該当案件の案件番号
16	priorityLevel	優先度	文字列	該当処理の優先度　※1
17	parameter	実行プログラムパス	文字列	該当処理の実行プログラムパス
18	actFlag	代理フラグ	文字列	”0”：本人にて処理　”1”：代理者にて処理
19	nodeId	ノードID	文字列	該当処理のノードID

No	パラメータ（物理名）	パラメータ（論理名）	属性	詳細
20	nextNodeIds	移動先（次ノード）ノードID	文字列[]	以下処理種別の場合に、移動先ノードIDが設定されます。 差戻し/引戻し/案件操作
21	authUserCd	処理権限者コード	文字列	処理権限者コード
22	execUserCd	処理実行者コード	文字列	処理実行者コード
23	resultStatus	処理結果ステータス	文字列	該当処理の処理結果ステータス [1]
24	authCompanyCode	権限会社コード	文字列	以下処理種別の場合、権限会社コードが設定されます。 申請/再申請/申請（一時保存）/申請（未処理） 取止め/承認/承認終了/否認/差戻し
25	authOrgzSetCode	権限組織セットコード	文字列	以下処理種別の場合、権限組織セットコードが設定されます。 申請/再申請/申請（一時保存）/申請（未処理） 取止め/承認/承認終了/否認/差戻し
26	authOrgzCode	権限組織コード	文字列	以下処理種別の場合、権限組織コードが設定されます。 申請/再申請/申請（一時保存）/申請（未処理） 取止め/承認/承認終了/否認/差戻し
27	processComment	処理コメント	文字列	該当処理の処理コメント
28	lumpProcessFlag	一括処理フラグ	文字列	”0”：通常承認　”1”：一括承認
29	autoProcessFlag	自動処理フラグ	文字列	”0”：手動処理　”1”：自動処理（到達処理での自動承認や、バッチでの自動処理）

[1] ・・・コード値は、「APIドキュメントのIM-Workflow CodeList」を参照してください。

コラム

アクション処理でユーザパラメータを利用し、以下の処理を行った場合には、ユーザパラメータには「空」または「Null」が連携されます。

- ユーザパラメータに「Null」が連携される処理
  - 引戻し
  - 差戻し後引戻し
  - 承認（「自動処理」を利用する場合）
  - 否認（「自動処理」を利用する場合）
  - 差戻し（「自動処理」を利用する場合）
  - 案件操作
  - 一時保存（削除）
- ユーザパラメータに「空」が連携される処理
  スクリプト開発モデルでは「空のオブジェクト」、JavaEE開発モデルでは「空のMap」を返却します。
  - 承認（「一括処理」を利用する場合）

「承認」等のアクション処理でユーザパラメータを利用する場合には、「lumpProcessFlag（一括処理フラグ）」・「autoProcessFlag（自動処理フラグ）」や、「matterHandle（案件操作）」メソッドを条件分岐に活用するなどの方法を検討してください。

返却値

アクション処理では、以下の情報を返却します。（任意/必須・・・●：必須　△：任意）

- スクリプト開発モデル

No	返却値（物理名）	返却値（論理名）	属性	必須/任意	詳細

No	返却値（物理名）	返却値（論理名）	属性	必須/任意	詳細
1	resultFlag	結果フラグ	真偽値	△	true：成功　false：失敗 指定しない場合、成功（true）として扱います。
2	message	結果メッセージ	文字列	△	結果フラグが失敗の場合のみ、設定します。
3	data	案件番号	文字列 （最大20バイト）	△	申請/再申請/申請（一時保存）/申請（未処理）の場合のみ、null以外の場合に案件番号を上書きします。

※ 結果メッセージを設定した場合、画面にエラーメッセージを表示します。

> **コラム**
>
> 結果フラグとして”false”（失敗）を設定した場合の動作は以下の通りです。
>
> - 該当のアクション処理以外に設定されている後続のアクション処理は実行されません。

- JavaEE開発モデル

No	返却値（論理名）	属性	必須/任意	詳細
1	案件番号	文字列 （最大20バイト）	△	申請/再申請/申請（一時保存）/申請（未処理）の場合のみ、null以外の場合に案件番号を上書きします。

※「jp.co.intra_mart.foundation.workflow.exception.WorkflowExternalException」をスローした場合、引数に設定したエラーメッセージを画面に表示します。詳細は「 APIドキュメント 」を参照してください。

> **コラム**
>
> WorkflowExternalException をスローした場合の動作は以下の通りです。
>
> - 該当のアクション処理以外に設定されている後続のアクション処理は実行されません。

到達処理

Contents
- パラメータ
- 返却値

到達処理とは、ノードに到達した場合に実行される処理です。
この処理は、アクション処理や IM-Workflow の内部処理とは独立した処理（thread）として実行されます。
そのため、到達処理でエラーが発生した場合、直前の処理を戻すこと（ロールバック）はできません。
（直前のアクション処理とは、トランザクションも別です。）

このプログラム中で、データベースの登録／更新／削除処理を行う場合は、独自にDBトランザクション制御を行ってください。

下記のような場合に実行されます。

- 前ノードの処理者が、”申請”または”承認”を行って到達した場合
- 他のノードから、”差戻し”され到達した場合
- “引戻し”を行って到達した場合
- 案件操作で到達した場合

パラメータ

到達処理では、以下の情報をパラメータとして受け取る事ができます。

No	パラメータ（物理名）	パラメータ（論理名）	属性	詳細
1	loginGroupId	ログイングループID	文字列	処理実行時のログイングループID ログイングループIDは、テナントIDと同値です。
2	localeId	ロケールID	文字列	処理実行時のロケールID
3	targetLocales	ターゲットロケールID	文字列[]	システムで利用しているロケールID
4	contentsId	コンテンツID	文字列	該当案件のコンテンツID
5	contentsVersionId	コンテンツバージョンID	文字列	該当案件のコンテンツバージョンID
6	routeId	ルートID	文字列	該当案件のルートID
7	routeVersionId	ルートバージョンID	文字列	該当案件のルートバージョンID
8	flowId	フローID	文字列	該当案件のフローID
9	flowVersionId	フローバージョンID	文字列	該当案件のフローバージョンID
10	applyBaseDate	申請基準日	文字列	申請基準日（”yyyy/MM/dd”）
11	processDate	処理日	文字列	処理日（”yyyy/MM/dd”）
12	systemMatterId	システム案件ID	文字列	該当案件のシステム案件ID
13	userDataId	ユーザデータID	文字列	該当案件のユーザデータID
14	matterName	案件名	文字列	該当案件の案件名
15	matterNumber	案件番号	文字列	該当案件の案件番号
16	priorityLevel	優先度	文字列	該当処理の優先度　※1
17	parameter	実行プログラムパス	文字列	該当処理の実行プログラムパス
18	actFlag	代理フラグ	文字列	”0”：本人にて処理　”1”：代理者にて処理
19	nodeId	ノードID	文字列	該当処理のノードID
20	preNodeId	前ノードID	文字列[]	前処理のノードID
21	preNodeAuthUserCd	前ノード処理権限者コード	文字列	前処理の処理権限者コード
22	preNodeExecUserCd	前ノード処理実行者コード	文字列	前処理の処理実行者コード
23	preNodeResultStatus	前ノード処理結果ステータス	文字列	前処理の処理結果ステータス [1]
24	preNodeAuthCompanyCode	前ノード権限会社コード	文字列	前処理の権限会社コード
25	preNodeAuthOrgzSetCode	前ノード権限組織セットコード	文字列	前処理の権限組織セットコード
26	preNodeAuthOrgzCode	前ノード権限組織コード	文字列	前処理の権限組織コード
27	preNodeProcessComment	前ノード処理コメント	文字列	前処理の処理コメント
28	mailIds	メールテンプレートID	文字列[]	到達したノードのメール種別「処理依頼」で設定されているメールID
29	imBoxIds	IMBoxId	文字列[]	到達したノードのIMBox種別「処理依頼」で設定されているIMBoxID
30	mailReplaceMap	メール置換文字情報	Map	メールの置換文字情報 [2]
31	imBoxReplaceMap	IMBox置換文字情報	Map	IMBoxの置換文字情報 [2]

[1] ・・・コード値は、「APIドキュメントのIM-Workflow CodeList」を参照してください。

[2] *(1, 2)* ・・・置換文字列と置換内容の詳細は、「IM-Workflow 仕様書 別紙」を参照してください。

返却値

到達処理では、以下の情報を返却します。（任意/必須・・・●：必須　△：任意）

- スクリプト開発モデル

No	返却値（物理名）	返却値（論理名）	属性	必須/任意	詳細
1	resultFlag	結果フラグ	真偽値	△	true：成功　false：失敗 指定しない場合、成功（true）として扱います。
2	message	結果メッセージ	文字列	△	結果フラグが失敗の場合のみ、設定します。
3	data	メール送信可否	真偽値	●	このパラメータは「処理依頼」の送信制御を行います。 このパラメータによる制御はメールに限らず、IMBox 、 IM-Notice などにも適用されます。 true：送信する　false：送信しない

※ 結果メッセージを設定した場合、例外ログにエラーメッセージを出力します。

> **コラム**
>
> 結果フラグとして”false”（失敗）を設定した場合の動作は以下の通りです。
>
> - 該当の到達処理以外に設定されている後続の到達処理は実行されません。
> - 処理依頼を設定している場合、「メール送信可否」の設定値に関係なく、処理依頼による通知が行われます。

- JavaEE開発モデル

No	返却値（論理名）	属性	必須/任意	詳細
1	メール送信可否	真偽値	●	このパラメータは「処理依頼」の送信制御を行います。 このパラメータによる制御はメールに限らず、 IMBox 、 IM-Notice などにも適用されます。 true：送信する　false：送信しない

※「jp.co.intra_mart.foundation.workflow.exception.WorkflowExternalException」をスローした場合、引数に設定したエラーメッセージを例外ログに出力します。詳細は「 APIドキュメント 」を参照してください。

> **コラム**
>
> WorkflowExternalException をスローした場合の動作は以下の通りです。
>
> - 該当の到達処理以外に設定されている後続の到達処理は実行されません。
> - 処理依頼を設定している場合、「メール送信可否」の設定値に関係なく、処理依頼による通知が行われます。

分岐開始処理

Contents

分岐開始処理とは、分岐開始ノードで「ユーザプログラムで分岐する」を選択した場合に、実行される処理です。
分岐先ノード毎に順番に実行されます。

分岐開始処理は、IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中ではDBトランザクション制御を行うことはできません。

分岐開始処理において、ルート遷移可否として 遷移する（true） を返却することにより、実行中の分岐開始処理が設定された分岐先ノードに進みます。

全ての分岐開始処理のルート遷移可否が 遷移しない（false） の場合は、案件は分岐開始ノードで停止します。
このような場合は、案件操作処理で案件を進めてください。

パラメータ

分岐開始処理では、以下の情報をパラメータとして受け取る事ができます。

No	パラメータ（物理名）	パラメータ（論理名）	属性	詳細
1	loginGroupId	ログイングループID	文字列	処理実行時のログイングループID ログイングループIDは、テナントIDと同値です。
2	localeId	ロケールID	文字列	処理実行時のロケールID
3	targetLocales	ターゲットロケールID	文字列[]	システムで利用しているロケールID
4	contentsId	コンテンツID	文字列	該当案件のコンテンツID
5	contentsVersionId	コンテンツバージョンID	文字列	該当案件のコンテンツバージョンID
6	routeId	ルートID	文字列	該当案件のルートID
7	routeVersionId	ルートバージョンID	文字列	該当案件のルートバージョンID
8	flowId	フローID	文字列	該当案件のフローID
9	flowVersionId	フローバージョンID	文字列	該当案件のフローバージョンID
10	applyBaseDate	申請基準日	文字列	申請基準日（”yyyy/MM/dd”）
11	processDate	処理日	文字列	処理日（”yyyy/MM/dd”）
12	systemMatterId	システム案件ID	文字列	該当案件のシステム案件ID
13	userDataId	ユーザデータID	文字列	該当案件のユーザデータID
14	parameter	実行プログラムパス	文字列	該当処理の実行プログラムパス
15	nodeId	ノードID	文字列	該当処理のノードID

返却値

分岐開始処理では、以下の情報を返却します。（任意/必須・・・●：必須　△：任意）

- スクリプト開発モデル

No	返却値（物理名）	返却値（論理名）	属性	必須/任意	詳細
1	resultFlag	結果フラグ	真偽値	△	true：成功　false：失敗 指定しない場合、成功（true）として扱います。
2	message	結果メッセージ	文字列	△	結果フラグが失敗の場合のみ、設定します。
3	data	ルート遷移可否	真偽値	●	[分岐]　true：遷移する　false：遷移しない [結合]　true：結合する　false：結合しない

※ 結果メッセージを設定した場合、画面にエラーメッセージを表示します。

- JavaEE開発モデル

No	返却値（論理名）	属性	必須/任意	詳細
1	ルート遷移可否	真偽値	●	[分岐]　true：遷移する　false：遷移しない [結合]　true：結合する　false：結合しない

※ 「jp.co.intra_mart.foundation.workflow.exception.WorkflowExternalException」をスローした場合、引数に設定したエラーメッセージを画面に表示します。詳細は「 APIドキュメント 」を参照してください。

分岐終了処理

分岐終了処理とは、分岐終了ノードで「ユーザプログラムで分岐終了する」を選択した場合に、実行される処理です。
分岐終了ノードに案件が到達する度に実行されます。

分岐終了処理は、 IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中ではDBトランザクション制御を行うことはできません。

分岐終了処理において、ルート遷移可否として 結合する（true） を返却することにより、未到達のノードを待たずに次のノードに進みます。
全てのノードが到達しても結果が全て 結合しない（false） の場合は、案件は分岐終了ノードで停止します。
このような場合は、案件操作処理で案件を進めてください。

パラメータ

分岐終了処理のパラメータは、「 *分岐開始処理* 」の「 *パラメータ* 」と同様です。

返却値

分岐終了処理の返却値は、「 *分岐開始処理* 」の「 *返却値* 」と同様です。

案件終了処理（トランザクションなし）

案件終了処理（トランザクションなし）とは、案件終了処理のトランザクションとは独立した処理です。
案件が終了する時に、一度実行されます。

案件終了処理（トランザクションなし）は、直前のアクション処理や到達処理とは独立した処理です。
そのため、案件終了処理（トランザクションなし）でエラーが発生した場合、直前の処理を戻す（ロールバック）ことはできません。

また、案件終了処理（トランザクションなし）は、案件終了処理の前に実行されます。
そのため、案件終了処理でエラーが発生した場合、案件終了処理（トランザクションなし）は実施済みとなり、処理を戻す（ロールバック）ことはできません。

このプログラム中で、データベースの登録／更新／削除処理を行う場合は、独自にDBトランザクション制御を行ってください。

下記の場合に実行されます。

- 最後の承認者が”承認”を行った場合
- 承認者が”承認終了”を行った場合
- 承認者が”否認”を行った場合
- 申請者が”取止め”を行った場合
- 案件操作で終了ノードに到達した場合

パラメータ

案件終了処理（トランザクションなし）では、以下の情報をパラメータとして受け取る事ができます。

No	パラメータ（物理名）	パラメータ（論理名）	属性	詳細

No	パラメータ（物理名）	パラメータ（論理名）	属性	詳細
1	loginGroupId	ログイングループID	文字列	処理実行時のログイングループID ログイングループIDは、テナントIDと同値です。
2	localeId	ロケールID	文字列	処理実行時のロケールID
3	targetLocales	ターゲットロケールID	文字列[]	システムで利用しているロケールID
4	contentsId	コンテンツID	文字列	該当案件のコンテンツID
5	contentsVersionId	コンテンツバージョンID	文字列	該当案件のコンテンツバージョンID
6	routeId	ルートID	文字列	該当案件のルートID
7	routeVersionId	ルートバージョンID	文字列	該当案件のルートバージョンID
8	flowId	フローID	文字列	該当案件のフローID
9	flowVersionId	フローバージョンID	文字列	該当案件のフローバージョンID
10	applyBaseDate	申請基準日	文字列	申請基準日（”yyyy/MM/dd”）
11	processDate	処理日	文字列	処理日（”yyyy/MM/dd”）
12	systemMatterId	システム案件ID	文字列	該当案件のシステム案件ID
13	userDataId	ユーザデータID	文字列	該当案件のユーザデータID
14	parameter	実行プログラムパス	文字列	該当処理の実行プログラムパス
15	actFlag	代理フラグ	文字列	”0”：本人にて処理　”1”：代理者にて処理
16	lastProcessNodeId	最終処理ノードID	文字列	最終処理のノードID
17	lastAuthUserCd	最終処理権限者コード	文字列	最終処理の処理権限者コード
18	lastExecUserCd	最終処理実行者コード	文字列	最終処理の処理実行者コード
19	lastResultStatus	最終処理結果ステータス	文字列	最終処理の処理結果ステータス [1]

[1] ・・・コード値は、「APIドキュメントのIM-Workflow CodeList」を参照してください。

返却値

案件終了処理（トランザクションなし）では、以下の情報を返却します。（任意/必須・・・●：必須　△：任意）

- スクリプト開発モデル

No	返却値（物理名）	返却値（論理名）	属性	必須/任意	詳細
1	resultFlag	結果フラグ	真偽値	△	true：成功　false：失敗 指定しない場合、成功（true）として扱います。
2	message	結果メッセージ	文字列	△	結果フラグが失敗の場合のみ、設定します。

※ 結果メッセージを設定した場合、例外ログにエラーメッセージを出力します。

> **コラム**
>
> 結果フラグとして”false”（失敗）を設定した場合の動作は以下の通りです。
>
> - 該当の案件終了処理（トランザクションなし）以外に設定されている後続の案件終了処理（トランザクションなし）は実行されません。

- JavaEE開発モデル

返却値はありません。

※「jp.co.intra_mart.foundation.workflow.exception.WorkflowExternalException」をスローした場合、引数に設定したエ

ラーメッセージを例外ログに出力します。詳細は「 APIドキュメント 」を参照してください。

コラム

WorkflowExternalException をスローした場合の動作は以下の通りです。

- 該当の案件終了処理（トランザクションなし）以外に設定されている後続の案件終了処理（トランザクションなし）は実行されません。

未完了案件削除処理リスナー

未完了案件削除処理リスナーとは、未完了案件を削除した際に実行されるプログラムです。
通常、「案件操作」画面より“案件削除“を行った場合、または未完了案件を削除するAPIを実行した際に呼び出されます。

未完了案件削除処理リスナーは、通常 **「コンテンツ定義」** に設定します。

また、テナント単位で処理を行う場合は、下記のファイルに設定します。

```
%PUBLIC_STORAGE_PATH%/im_workflow/conf/param/param_group_%テナントID%.xml
```

```
<!--
   未完了案件削除リスナーの種類
      [java] or [script] or []（指定なし）
      []（指定なし）を設定した場合はリスナーを起動しない
-->
<param>
   <param-name>delete-active-matter-type</param-name>
   <param-value></param-value>
</param>
<!--
   未完了案件削除リスナーのパス
   1. 案件削除リスナーの種類がjava：パッケージ名
   2. 案件削除リスナーの種類がscript：WEB-INF/jsspからのパス
-->
<param>
   <param-name>delete-active-matter-listener-path</param-name>
   <param-value></param-value>
</param>
```

※[ワークフローパラメータ]画面からも設定することが可能です。

設定方法の詳細については、「 IM-Workflow 管理者操作ガイド 」 または「 IM-Workflow 仕様書 」を参照してください。

完了案件削除処理リスナー

完了案件削除処理リスナーとは、完了案件を削除した際に実行されるプログラムです。
通常、「参照」画面の完了案件タブより案件の“削除“を行った場合、または完了案件を削除するAPIを実行した際に呼び出されます。

完了案件削除処理リスナーは、通常 **「コンテンツ定義」** に設定します。

また、テナント単位で処理を行う場合は、下記のファイルに設定します。

```
%PUBLIC_STORAGE_PATH%/im_workflow/conf/param/param_group_%テナントID%.xml
```

```
<!--
  完了案件削除リスナーの種類
    [java] or [script] or []（指定なし）
    []（指定なし）を設定した場合はリスナーを起動しない
-->
<param>
  <param-name>delete-complete-matter-listener-type</param-name>
  <param-value></param-value>
</param>
<!--
  完了案件削除リスナーのパス
  1. 案件削除リスナーの種類がjava：パッケージ名
  2. 案件削除リスナーの種類がscript：WEB-INF/jssp　からのパス
-->
<param>
  <param-name>delete-complete-matter-listener-path</param-name>
  <param-value></param-value>
</param>
```

※[ワークフローパラメータ]画面からも設定することが可能です。

設定方法の詳細については、「 IM-Workflow 管理者操作ガイド 」 または「 IM-Workflow 仕様書 」を参照してください。

過去案件削除処理リスナー

過去案件削除処理リスナーとは、過去案件を削除した際に実行されるプログラムです。

過去案件削除処理リスナーは、通常 **「コンテンツ定義」** に設定します。

また、テナント単位で処理を行う場合は、下記のファイルに設定します。

```
%PUBLIC_STORAGE_PATH%/im_workflow/conf/param/param_group_%テナントID%.xml
```

```
<!--
  過去案件削除リスナーの種類
    [java] or [script] or []（指定なし）
    []（指定なし）を設定した場合はリスナーを起動しない
-->
<param>
  <param-name>delete-archive-matter-listener-type</param-name>
  <param-value></param-value>
</param>
<!--
  過去案件削除リスナーのパス
  1. 案件削除リスナーの種類がjava：パッケージ名
  2. 案件削除リスナーの種類がscript：WEB-INF/jsspからのパス
-->
<param>
  <param-name>delete-archive-matter-listener-path</param-name>
  <param-value></param-value>
</param>
```

※[ワークフローパラメータ]画面からも設定することが可能です。

設定方法の詳細については、「 IM-Workflow 管理者操作ガイド 」または「 IM-Workflow 仕様書 」 を参照してください。

案件退避処理リスナー

案件退避処理リスナーとは、案件を退避した際に実行されるプログラムです。
通常、ジョブ「IM-Workflow/アーカイブ」を実行した際に呼び出されます。

案件退避処理リスナーは、通常 **「コンテンツ定義」** に設定します。

また、テナント単位で処理を行う場合は、下記のファイルに設定します。

```
%PUBLIC_STORAGE_PATH%/im_workflow/conf/param/param_group_%テナントID%.xml
```

```
<!--
   案件退避リスナーの種類
      [java] or [script] or []（指定なし）
      []（指定なし）を設定した場合はリスナを起動しない
-->
<param>
   <param-name>archive-proc-listener-type</param-name>
   <param-value>java</param-value>
</param>
<!--
   案件退避リスナーのパス
   1. 案件退避リスナーの種類がjava：パッケージ名
   2. 案件退避リスナーの種類がscript：WEB-INF/jsspからのパス
-->
<param>
   <param-name>archive-proc-listener-path</param-name>
   <param-value></param-value>
</param>
```

※[ワークフローパラメータ]画面からも設定することが可能です。

設定方法の詳細については、「 IM-Workflow 管理者操作ガイド 」または「 IM-Workflow 仕様書 」を参照してください。

Appendix

テンプレート

ユーザプログラムおよび各リスナーのプログラムを作成する際のテンプレートが提供されています。

- スクリプト開発モデル
  <./jssp/src/sample/im_workflow/template/>

No	処理	物理名
1	案件開始処理	MatterStartProcess.js
2	案件終了処理	MatterEndProcess.js
3	アクション処理	ActionProcess.js
4	到達処理	ArriveProcess.js
5	分岐開始処理／分岐終了処理	RuleCondition.js
6	未完了案件削除処理リスナー	WorkflowActvMatterDeleteListener.js
7	完了案件削除処理リスナー	WorkflowCplMatterDeleteListener.js
8	過去案件削除処理リスナー	WorkflowArcMatterDeleteListener.js
9	案件退避処理リスナー	WorkflowMatterArchiveListener.js
10	処理対象者プラグイン	WorkflowAuthorityExecEventListener.js
11	案件終了処理（トランザクションなし）	MatterEndProcessNoTran.js

- JavaEE開発モデル
  JavaEE開発モデル[javaファイル]のサンプルプログラムについては、製品メディアに保存されています。
  また、製品最新情報ダウンロードページ（http://www.intra-mart.jp/download/product/index.html）から入手することもできます。
  <%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample/workflow/template/>

No	処理	物理名
1	案件開始処理	MatterStartProcess.java
2	案件終了処理	MatterEndProcess.java
3	アクション処理	ActionProcess.java
4	到達処理	ArriveProcess.java
5	分岐開始処理／分岐終了処理	RuleCondition.java
6	未完了案件削除処理リスナー	WorkflowActvMatterDeleteListener.java
7	完了案件削除処理リスナー	WorkflowCplMatterDeleteListener.java
8	過去案件削除処理リスナー	WorkflowArcMatterDeleteListener.java
9	案件退避処理リスナー	WorkflowMatterArchiveListener.java
10	処理対象者プラグイン	WorkflowAuthorityExecEventListener.java
11	クローラ登録文書追加リスナー	WorkflowCrawlingAddListener.java
12	案件終了処理（トランザクションなし）	MatterEndProcessNoTran.java

サンプルプログラム

IM-Workflow のインストール時”サンプルデータセットアップ”を行い、サンプルデータをインポートした場合に使用できるサンプルプログラムについて説明します。

サンプルプログラムは、スクリプト開発モデルとJavaEE開発モデルのサンプルプログラムがあります。
開発モデルの違いはありますが、どちらのサンプルも「物品購買」の申請書であり、動作仕様は同一です。

画面

Contents

申請／一時保存／申請（起票案件）／再申請画面

PC用画面とスマートフォン用画面について説明します。

PC用画面

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/apply.html>
<./jssp/src/sample/im_workflow/purchase/screen/apply.js>
```

- JavaEE開発モデル

```
<./lib/im_workflow-8.0.x-sample.jar/service-config-imw_sample_purchase.xml>
```

```
<service>
  <service-id>apply</service-id>
  <controller-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApplyServiceController</controller-class>
  <transition-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApplyServiceTransition</transition-class>
  <next-page>
    <page-path>/sample/im_workflow/purchase/apply.jsp</page-path>
  </next-page>
</service>
```

スマートフォン用画面

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/apply.html>
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/apply.js>
```

- JavaEE開発モデル

```
<./lib/im_workflow_smartphone-8.0.x-sample.jar/service-config-imw_sp_sample_purchase.xml>
```

```
1 <service>
2   <service-id>apply</service-id>
3   <controller-
4 class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApplyServiceController</controller-class>
5   <transition-
6 class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApplyServiceTransition</transition-class>
7   <next-page>
8     <page-path>/sample/im_workflow_smartphone/purchase/apply.jsp</page-path>
  </next-page>
</service>
```

処理画面

PC用画面とスマートフォン用画面について説明します。

PC用画面

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/approve.html>
<./jssp/src/sample/im_workflow/purchase/screen/approve.js>
```

- JavaEE開発モデル

```
<./lib/im_workflow-8.0.x-sample.jar/service-config-imw_sample_purchase.xml>
```

```
<service>
  <service-id>approve</service-id>
  <controller-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApproveServiceController</controller-
class>
  <transition-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApproveServiceTransition</transition-
class>
  <next-page>
    <page-path>/sample/im_workflow/purchase/approve.jsp</page-path>
  </next-page>
</service>
```

スマートフォン用画面

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/approve.html>
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/approve.js>
```

- JavaEE開発モデル

```
<./lib/im_workflow_smartphone-8.0.x-sample.jar/service-config-imw_sp_sample_purchase.xml>
```

```
<service>
 <service-id>approve</service-id>
 <controller-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApproveServiceController</controller-
class>
 <transition-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApproveServiceTransition</transition-
class>
 <next-page>
  <page-path>/sample/im_workflow_smartphone/purchase/approve.jsp</page-path>
 </next-page>
</service>
```

確認画面

PC用画面とスマートフォン用画面について説明します。

PC用画面

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/approve.html>
<./jssp/src/sample/im_workflow/purchase/screen/approve.js>
```

- JavaEE開発モデル

```
<./lib/im_workflow-8.0.x-sample.jar/service-config-imw_sample_purchase.xml>
```

```
<service>
 <service-id>confirm</service-id>
 <controller-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ConfirmServiceController</controller-class>
 <transition-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ConfirmServiceTransition</transition-class>
 <next-page>
  <page-path>/sample/im_workflow/purchase/confirm.jsp</page-path>
 </next-page>
</service>
```

スマートフォン用画面

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/confirm.html>
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/confirm.js>
```

- JavaEE開発モデル

```
<./lib/im_workflow_smartphone-8.0.x-sample.jar/service-config-imw_sp_sample_purchase.xml>
```

```
<service>
  <service-id>confirm</service-id>
  <controller-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ConfirmServiceController</controller-class>
  <transition-
class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ConfirmServiceTransition</transition-class>
  <next-page>
    <page-path>/sample/im_workflow_smartphone/purchase/confirm.jsp</page-path>
  </next-page>
</service>
```

処理詳細／参照詳細／過去案件詳細／確認詳細画面

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/detail.html>
<./jssp/src/sample/im_workflow/purchase/screen/detail.js>
```

- JavaEE開発モデル

```
<./lib/im_workflow-8.0.x-sample.jar/service-config-imw_sample_purchase.xml>
```

```
1 <service>
2   <service-id>detail</service-id>
3   <controller-
4 class>jp.co.intra_mart.sample.workflow.purchase.controller.service.DetailServiceController</controller-class>
5   <transition-
6 class>jp.co.intra_mart.sample.workflow.purchase.controller.service.DetailServiceTransition</transition-class>
7   <next-page>
8     <page-path>/sample/im_workflow/purchase/detail.jsp</page-path>
  </next-page>
</service>
```

処理詳細／参照詳細／過去案件詳細／確認詳細画面（以下、詳細画面）では、コンテンツ定義で定義した画面が表示されます。
そのため、詳細画面にIM-Workflowの情報（案件名や添付ファイルなど）を表示する場合は、IM-Workflow が提供するタグライブラリを使用します。

案件の情報を表示するためのタグライブラリです。

案件番号	0000000018
案件名	物品購買
申請者	円山益男
申請基準日	2012/09/19

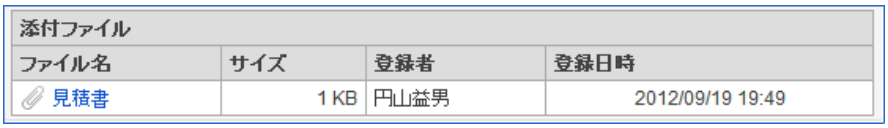
- スクリプト開発モデル　detail.html

```
43 | </header>
44 | <imart type="workflowMatterData" systemMatterId=$data.imwSystemMatterId
45 |                                  displayItem="matter_number,matter_name,apply_user,apply_base_date" />
46 | <table class="imui-form">
```

- JavaEE開発モデル detail.jsp

```
51 | </header>
52 | <workflow:workflowMatterData systemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
53 |                              displayItem="matter_number,matter_name,apply_user,apply_base_date" />
54 | <table class="imui-form">
```

案件の添付ファイルを表示するためのタグライブラリです。



添付ファイル			
ファイル名	サイズ	登録者	登録日時
見積書	1 KB	円山益男	2012/09/19 19:49

- スクリプト開発モデル　detail.html

```
69 |  </table>
70 |  <imart type="workflowMatterFile" systemMatterId=$data.imwSystemMatterId />
71 | </div>
```

- JavaEE開発モデル detail.jsp

```
78 |  </table>
79 |  <workflow:workflowMatterFile systemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
/>
80 | </div>
```

※ 案件に添付ファイルがない場合は、表示されません。

これらの詳細画面は、スマートフォン版 IM-Workflow から画面遷移した際、PC用の画面を新しいウィンドウで開きます。
スマートフォンからの画面遷移でPC用の画面を表示させたい場合は、明示的にクライアントタイプをPCに切り替える必要があります。

- スクリプト開発モデル detail.js

```
19 | function init ( request ) {
20 |    ClientTypeSwitcher.oneTimeSwitchTo('pc');
21 |
```

- JavaEE開発モデル detail.jsp

```
7 | <%
8 | ClientTypeSwitcher.oneTimeSwitchTo("pc");
9 | %>
```

ClientTypeSwitcher について、詳細は「APIドキュメント」を参照してください。

また、新しいウィンドウで表示する画面にグローバルナビやマイメニューを表示させないようにするには、以下のフィルターに画面のパスを追加する必要があります。

```
<./conf/theme-head-only-path-config/*.xml>
```

- スクリプト開発モデル

```
7 |  <path>/sample/im_workflow/purchase/screen/detail</path>
```

- JavaEE開発モデル

```
8 |  <path>/imw_sample_purchase-detail.service</path>
```

ユーザプログラム

Contents

アクション処理プログラム

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/action/ActionProcess1.js>
```

- JavaEE開発モデル

```
<%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample/workflow/purchase/action/ActionProcess1.java>
```

サンプルデータでは[ActionProcess1]を申請ノードのアクション処理として定義されています。

[ActionProcess1]では、下記の2つの処理を行っています。

- ユーザアプリケーションのデータをテーブルに保存する。
  - 申請または一時保存を行った場合に、画面に入力された情報をユーザアプリケーションで定義している独自のテーブルに登録／更新しています。
- 案件番号を採番する。
  - 案件番号は、申請のアクション処理で設定する必要があります。
  - ここでは、IM-Workflow が提供する「WorkflowNumberingManager#getNumber()」で案件番号の採番を行っています。

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/action/ActionProcess2.js>
```

- JavaEE開発モデル

```
<%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample/workflow/purchase/action/ActionProcess2.java>
```

サンプルデータでは[ActionProcess2]を申請ノードのアクション処理として定義しています。
[ActionProcess2]では、画面から入力された「数量×金額」である”合計金額”を案件プロパティとして登録する処理を行っています。

案件終了処理プログラム

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/action/MatterEndProcess.js>
```

- JavaEE開発モデル

```
<%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample/workflow/purchase/action/MatterEndProcess.java>
```

サンプルデータでは[MatterEndProcess]を案件終了処理として定義しています。

[MatterEndProcess]では、ユーザアプリケーションで定義している独自のテーブルの更新処理を行っています。

分岐開始処理プログラム

- フロー定義“分岐ルート[スクリプト開発モデル]“で使用されている分岐開始処理プログラム

```
<./jssp/src/sample/im_workflow/purchase/action/RuleCondition1.js>
<./jssp/src/sample/im_workflow/purchase/action/RuleCondition2.js>
```

```
<./jssp/src/sample/im_workflow/purchase/action/RuleCondition3.js>
```

- フロー定義“分岐ルート[JavaEE開発モデル]“で使用されている分岐開始処理プログラム

```
<%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample/workflow/purchase/action/RuleCondition1.java>
<%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample/workflow/purchase/action/RuleCondition2.java>
<%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample/workflow/purchase/action/RuleCondition3.java>
```

[RuleCondition1]では、”合計金額”が10000未満の場合に結果フラグとして成功（true）を返却します。

[RuleCondition2]では、”合計金額”が10000以上50000未満の場合に結果フラグとして成功（true）を返却します。

[RuleCondition3]では、”合計金額”が50000以上の場合に結果フラグとして成功（true）を返却します。

リスナー

Contents

未完了案件削除処理リスナー

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/listener/WorkflowActvMatterDeleteListener.js>
```

- JavaEE開発モデル

```
<%サンプルプログラムディレクトリ%
  /jp/co/intra_mart/sample/workflow/purchase/listener/WorkflowActvMatterDeleteListener.java>
```

[WorkflowActvMatterDeleteListener]では、下記の2つの処理を行っています。

- ユーザアプリケーションのデータをテーブルから削除する。
  - 申請時に登録したユーザアプリケーションのデータを案件削除と同タイミングで削除しています。
- 案件プロパティを削除する。
  - 申請時に案件プロパティに登録した“合計金額”を案件プロパティから削除しています。

完了案件削除処理リスナー

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/listener/WorkflowCplMatterDeleteListener.js>
```

- JavaEE開発モデル

```
<%サンプルプログラムディレクトリ%
  /jp/co/intra_mart/sample/workflow/purchase/listener/WorkflowCplMatterDeleteListener.java>
```

[WorkflowCplMatterDeleteListener]では、次の処理を行っています。

- 申請時に登録したユーザアプリケーションのデータを案件削除と同タイミングで削除しています。

※ 案件プロパティの情報は、案件削除のタイミングで IM-Workflow モジュールが自動的に削除しますので、個別の削除は不要です。

過去案件削除処理リスナー

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/listener/WorkflowArcMatterDeleteListener.js>
```

- JavaEE開発モデル

```
<%サンプルプログラムディレクトリ%
  /jp/co/intra_mart/sample/workflow/purchase/listener/WorkflowArcMatterDeleteListener.java>
```

[WorkflowArcMatterDeleteListener]では、次の処理を行っています。

- 申請時に登録したユーザアプリケーションのデータを案件削除と同タイミングで削除しています。

※ 案件プロパティの情報は、案件削除のタイミングで IM-Workflow モジュールが自動的に削除しますので、個別の削除は不要です。

案件退避処理リスナー

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/listener/WorkflowMatterArchiveListener.js>
```

- JavaEE開発モデル

```
<%サンプルプログラムディレクトリ%
  /jp/co/intra_mart/sample/workflow/purchase/listener/WorkflowMatterArchiveListener.java>
```

[WorkflowMatterArchiveListener]では、次の処理を行っています。

- ユーザアプリケーションで定義している独自のテーブルの更新処理を行っています。

呼び出し画面の初期表示値指定

ここで記載している内容は、次の観点で共通です。

- 開発モデル
- クライアントタイプ

IM-Workflow で提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の呼び出し時に、呼び出し画面における初期表示値を外部指定する方法を説明します。

指定可能なパラメータ

「workflowOpenPage」タグの内部に下記パラメータを記述することにより、呼び出し画面における初期表示値を外部指定することが可能です。

No	パラメータ（物理名）	パラメータ（論理名）	呼び出し画面側の対応項目	動作対象呼び出し画面
1	imwMatterName	案件名	案件名	申請／一時保存／申請（起票案件）／再申請
2	imwComment	コメント	コメント	すべて
3	imwForcedParamFlag	強制パラメータフラグ	※動作制御用フラグ	-

また、下記のような条件のとき、「imwForcedParamFlag」（強制パラメータフラグ）の値に”1”を指定した場合のみ、初期表示値指定が反映されます。
「imwForcedParamFlag」（強制パラメータフラグ）の値に”1”を指定しない場合、または、「imwForcedParamFlag」（強制パラメータフラグ）を記述しない場合は、登録されている情報が優先されます。

No	呼び出し画面	条件
1	申請	一時保存からの申請時
2	一時保存	一時保存情報の再保存時
3	申請（起票案件）	-
4	再申請	-

実装例

サンプルとして提供されている「物品購買」の申請書において、申請画面で入力される「品名」を「案件名」に、「備考」を「コメント」に初期表示する例です。

なお、サンプルはPC用画面のみ用意しています。
スマートフォン用画面の場合も全体の流れは同じです。実装中で使用するタグライブラリや Client-side JavaScript API が異なることに注意してください。

下記のプログラムが、初期表示を行うための処理が記述されたプログラムです。

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/apply_display.html>
```

- JavaEE開発モデル

```
<（展開したwar）/sample/im_workflow/purchase/apply_display.jsp>
```

これらのファイルを、以下のファイル名に変更し、上書き保存することで、申請画面において本機能の動作確認を行うことが出来ます。

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/apply.html>
```

- JavaEE開発モデル

```
<（展開したwar）/sample/im_workflow/purchase/apply.jsp>
```

以下のような処理を記述することで、初期表示を行うことが出来ます。

```
<imart type="head">
<title>
   <imart type="string" value=$msg.cap010 escapeXml="true" escapeJs="false" />
</title>

<imart type="workflowOpenPageCsjs" />
<script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
<script type="text/javascript">
 .
 .
 .

// 入力された品名と備考を、案件名、コメントにセットします。
function setParam() {
   $('#imwMatterName').val($('#item_name').val());
   $('#imwComment').val($('#item_comment').val());
   $('#imwForcedParamFlag').val('1');
}

 $(function(){
 .
 .
 .
    $('#openPage1').click(function(){
       // 値をセットする処理を呼び出します。
       setParam();
       workflowOpenPage('1');
    });
 .
 .
 .

 }

 </script>
 </imart>

 <imart type="workflowOpenPage"
   name="workflowOpenPageForm"
   id="workflowOpenPageForm"
   method="POST"
   target="_top"
   imwUserDataId=$data.imwUserDataId
   imwSystemMatterId=$data.imwSystemMatterId
   imwAuthUserCode=$data.imwAuthUserCode
   imwApplyBaseDate=$data.imwApplyBaseDate
   imwNodeId=$data.imwNodeId
   imwFlowId=$data.imwFlowId
   imwCallOriginalParams=$data.imwCallOriginalParams
   imwNextScriptPath=$data.imwCallOriginalPagePath>

//案件名、コメント値を保持するエリアを定義します。
<input type="hidden" name="imwMatterName" id="imwMatterName" />
<input type="hidden" name="imwComment" id="imwComment" />
<input type="hidden" name="imwForcedParamFlag" id="imwForcedParamFlag" />
 .
 .
 .
 </imart>
```

処理対象者プラグインの作成

IM-Workflow の各ノードに指定する「処理対象者」に、独自に作成した処理対象者を追加する方法を説明します。

IM-Workflow の処理対象者は、プラグインという形で機能を拡張できます。
プラグインを追加する場合には、拡張ポイントに応じた内容でプラグインの実装を作成し、対象の拡張ポイントへPluginするための設定ファイルを記述します。
拡張ポイントと、プラグインの関係は intra-mart Accel Platform のAPIである”PluginManager”によって管理されます。

対象ノード（拡張ポイント）

処理対象者プラグインは、ノードの種類により「extension point」が決められています。

No	ノード	extension point
1	承認（※1）	jp.co.intra_mart.workflow.plugin.authority.node.approve
2	承認（※2）	jp.co.intra_mart.workflow.plugin.authority.node.approve.static
3	動的承認	jp.co.intra_mart.workflow.plugin.authority.node.dynamic
4	確認	jp.co.intra_mart.workflow.plugin.authority.node.confirm

※1　前ノードが、”申請ノード”または”承認ノード”の場合
※2　前ノードが、”申請ノード”または”承認ノード”以外の場合

サンプルの説明

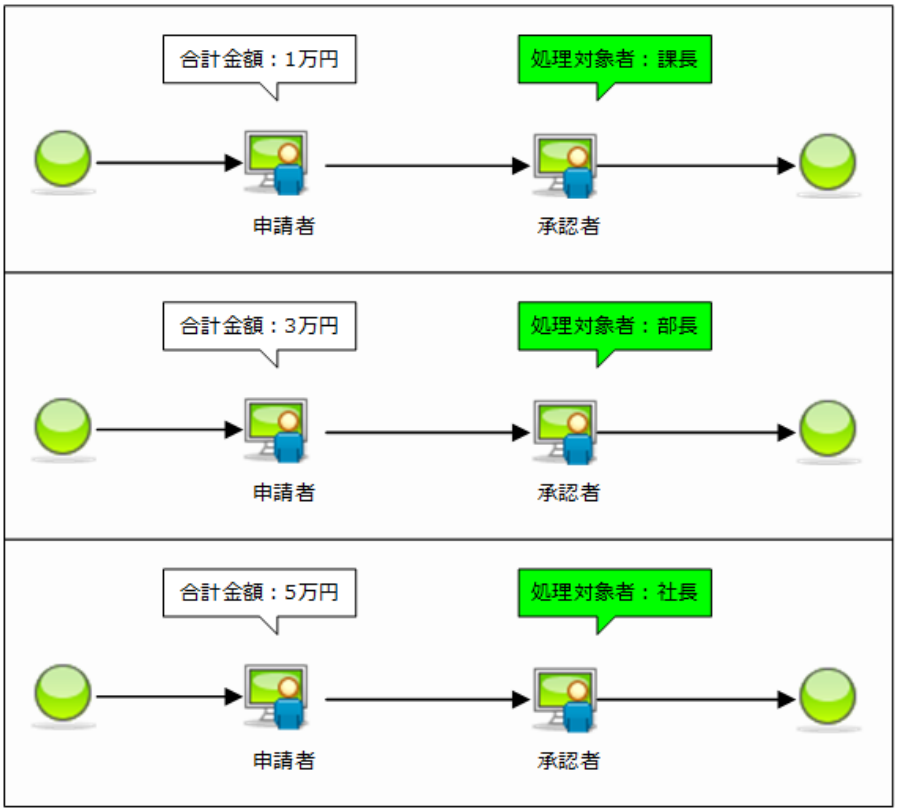
サンプルで提供する“処理対象者プラグイン”は、同じくサンプルで提供されている“物品購買“の画面と連携しています。

“物品購買“の画面で入力された「数量」と「金額」からの「合計金額」により、次の承認者を決定します。

具体的には、「合計金額」により、

- 1万円未満
  - 課長
- 1万円以上かつ5万円未満
  - 部長
- 5万円以上
  - 社長

と、処理対象者に役職が割り当てられます。



サンプルの実行準備

ここでは、承認ノードに対して、「合計金額」で処理対象者を決めるプラグインを使用してみます。

下記のファイルを編集します。

```
<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve/plugin.xml>
```

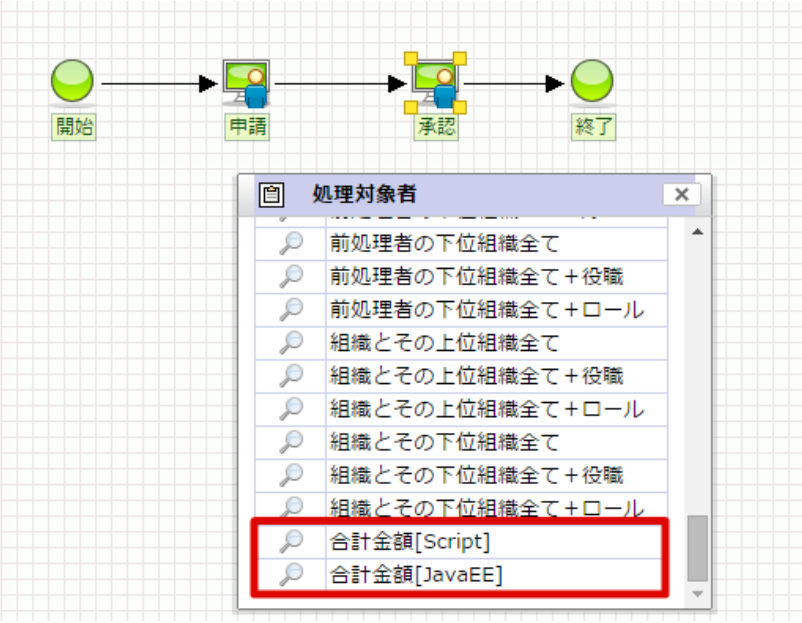
スクリプト開発モデル

JavaEE開発モデル

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
    <extension point="jp.co.intra_mart.workflow.plugin.authority.node.approve" >

        <authority
            name="%jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.script"
            id="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.script"
            version="7.2.0"
            rank="910"
            enable="true">
            <configPage>
                <script pagePath="sample/im_workflow/purchase/plugin/authority/item_total/itemTotalConfig">
                    <parameter key="pluginName" value="SAMPLE.IMW.CAP.030" />
                </script>
            </configPage>
            <extend>
                <script
            file="sample/im_workflow/purchase/plugin/authority/item_total/WorkflowAuthorityExecEventListener" />
            </extend>
        </authority>

        <authority
            name="%jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.javaee"
            id="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.javaee"
            version="7.2.0"
            rank="920"
            enable="true">
            <configPage>
                <javaee applicationId="imw_sample_purchase" serviceId="authority_item_total">
                    <parameter key="pluginName" value="SAMPLE.IMW.CAP.031" />
                </javaee>
            </configPage>
            <extend>
                <java
class="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.item_total.WorkflowAuthorityExecEventListener" />
            </extend>
        </authority>

    </extension>
</plugin>
```

上記ファイルを編集後、サーバを再起動します。

[ルート定義]画面より、次のようなルートを作成します。
承認ノードの処理対象者の検索を行うと、下記のように「合計金額[Script]」および「合計金額[JavaEE]」が表示されます。



「合計金額[Script]」および「合計金額[JavaEE]」は、実装方法（開発言語）の違いによるもので、処理内容に関して違いはありません。

「合計金額[Script]」または「合計金額[JavaEE]」を選択し、ルートを作成します。
次に、[フロー定義]画面より、上記で作成したルート定義を使用したフロー定義を作成します。

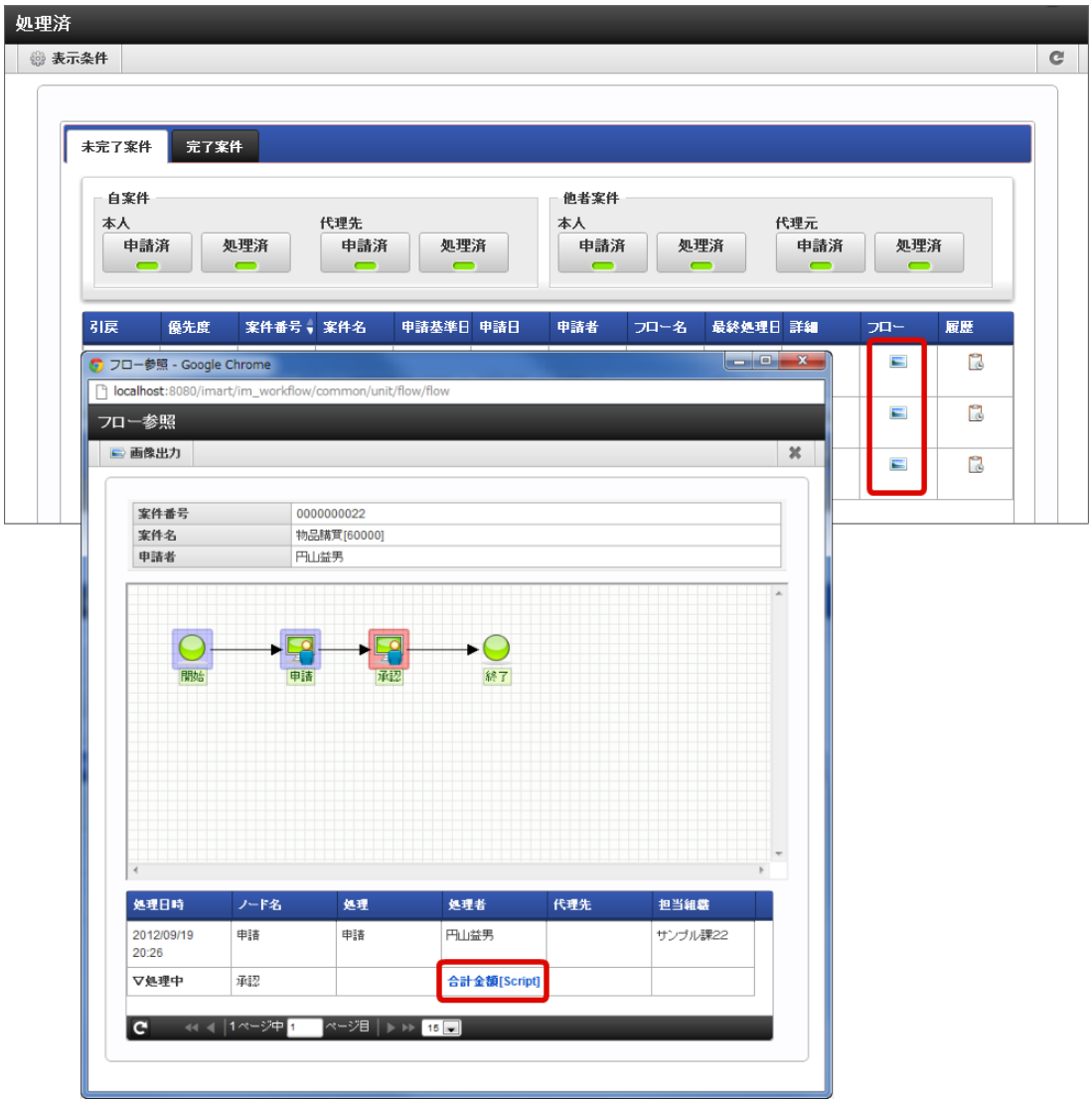
この時、コンテンツは、サンプルで提供されている「スクリプト開発モデル」または、「JavaEE開発モデル」を選択してください。

サンプルの実行

「*サンプルの実行準備*」で作成したフロー定義で申請を行います。

入力した「数量」と「金額」からの「合計金額」により、承認ノードの処理対象者が変わることを確認します。

[処理済]一覧画面より、申請を行った案件のフローを参照します。



- 合計金額が1万円未満の場合

- 合計金額が1万円以上かつ5万円未満の場合

- 合計金額が5万円以上の場合

「合計金額」により、処理対象者が違うことを確認します。

処理対象者プラグインの構成

処理対象者プラグインを作成するには、次の3ファイルを作成する必要があります。

Contents
- plugin.xmlファイル定義
  - pluginタグ設定
  - configPage設定
  - extend設定

plugin.xmlファイル定義

「plugin.xml」は、”PluginManager”によって管理されるファイルです。
処理対象者プラグインを新規に作成する場合は、plugin.xml を新規作成します。
IM-Workflow が提供するサンプルの plugin.xml を参考に作成されることを推奨します。

intra-mart Accel Platform が提供するプラグインの仕様については、以下のドキュメントも合わせて参照してください。

- 「 プラグイン仕様書 」-「 プラグインの構成 」

作成したプラグインを intra-mart Accel Platform 上に追加する方法は以下のドキュメントを参照してください。

- 「 IM-Workflow 管理者操作ガイド 」-「 処理対象者プラグインを設定する 」

本項では、次の plugin.xml の内容をもとに、処理対象者プラグインを構成するそれぞれの要素、属性について説明します。

/plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve/plugin.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <extension point="jp.co.intra_mart.workflow.plugin.authority.node.approve" >

    <authority
      id="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.script"
      name="%jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.script"
      version="8.0.99"
      rank="910"
      enable="true">
      <configPage>
        <script pagePath="sample/im_workflow/purchase/plugin/authority/item_total/itemTotalConfig">
          <parameter key="pluginName" value="SAMPLE.IMW.CAP.030" />
        </script>
      </configPage>
      <extend>
        <script
      file="sample/im_workflow/purchase/plugin/authority/item_total/WorkflowAuthorityExecEventListener" />
      </extend>
    </authority>

    <authority
      id="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.javaee"
      name="%jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.javaee"
      version="8.0.99"
      rank="920"
      enable="true">
      <configPage>
        <javaee applicationId="imw_sample_purchase" serviceId="authority_item_total">
          <parameter key="pluginName" value="SAMPLE.IMW.CAP.031" />
        </javaee>
      </configPage>
      <extend>
        <java
class="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.item_total.WorkflowAuthorityExecEventListener"
/>
      </extend>
    </authority>

  </extension>
</plugin>
```

pluginタグ設定

処理対象者プラグインで重要になるのは、下記の要素です。

<extension point>	処理対象者プラグインを差し込むノードの種類により、 <extension point>が変わります。 差し込みたいノードの<extension point>を指定します。

<configPage>	<script pagePath> <javaee applicationId serviceId>	<configPage>は、[ルート定義]画面において、 ノードに設定する処理対象者の一覧画面から、 処理対象者プラグインが選択されたときに呼ばれる プログラムです。 このプログラムは、スクリプト開発モデルおよび、 JavaEE開発モデルで記述することが可能です。 スクリプト開発モデルでプログラムを作成する場合は、 <script pagePath>にパスを指定します。 JavaEE開発モデルでプログラムを作成する場合は、 applicationId および serviceId を指定します。
< extend >	<script file> <java class>	< extend > に指定するプログラムは、処理対象者を 決定するプログラムが対象です。 このプログラムは、スクリプト開発モデルおよび、 JavaEE開発モデルで記述することが可能です。 スクリプト開発モデルでプログラムを作成する場合は、 <script file>にパスを指定します。 JavaEE開発モデルでプログラムを作成する場合は、 <java class>にパッケージを指定します。

その他の要素、属性については、「 PluginManagerのAPIドキュメント 」を参照してください。

サンプルの plugin.xml としては、下記を用意しています。

- 承認ノード

```
<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve/plugin.xml>
```

- 承認ノード

```
<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.static/plugin.xml>
```

- 動的承認ノード

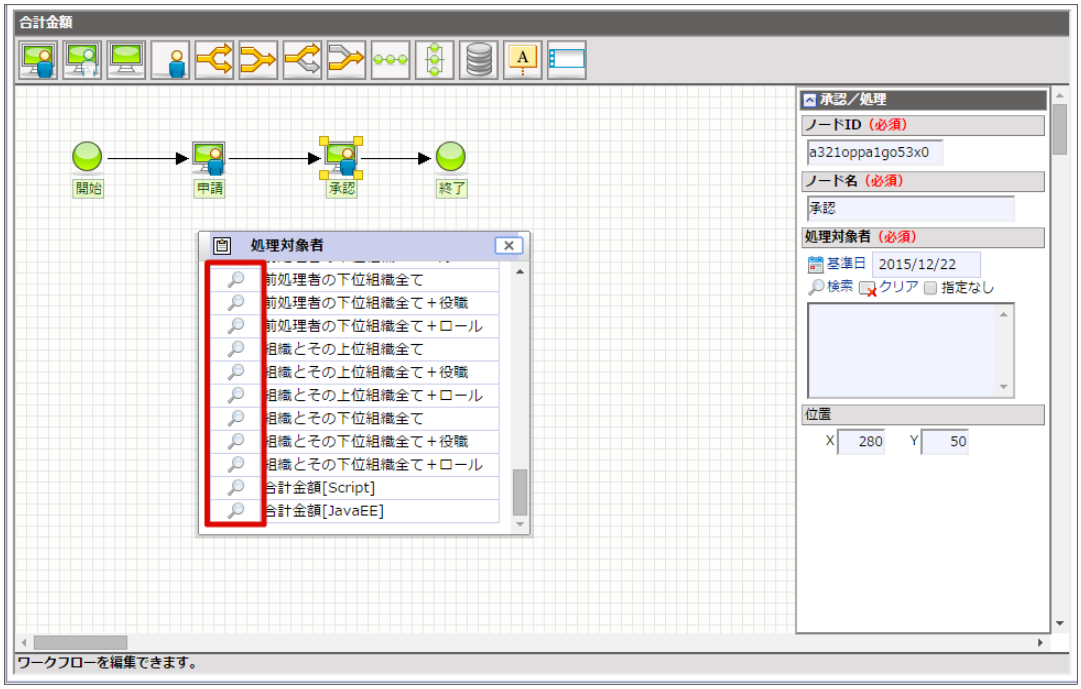
```
<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.dynamic/plugin.xml>
```

- 確認ノード

```
<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.confirm/plugin.xml>
```

configPage設定

前出の plugin.xml で＜configPage＞として指定したプログラムを作成します。
[ルート定義]画面において、ノードに設定する処理対象者の一覧画面から、処理対象者プラグインが選択されたときに呼ばれるプログラムです。
製品が提供している処理対象者プラグインのように検索を利用したい場合には、<configPage>で指定したプログラム内で検索画面を呼び出すように実装してください。



選択された対象者プラグインの情報を、[ルート定義]画面に引き渡します。

サンプルプログラムとしては、下記を用意しています。

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/plugin/authority/item_total/itemTotalConfig.html>
<./jssp/src/sample/im_workflow/purchase/plugin/authority/item_total/itemTotalConfig.js>
```

- JavaEE開発モデル

```
<./lib/im_workflow-8.0.0-sample.jar/service-config-imw_sample_purchase.xml>
-------------------------------------------------------------------------------------------------------------------

<service>
   <service-id>authority_item_total</service-id>


<controller-class>jp.co.intra_mart.sample.workflow.purchase.plugin.authority.item_total.controller.service.ItemTotalConfig



<transition-class>jp.co.intra_mart.sample.workflow.purchase.plugin.authority.item_total.controller.service.ItemTotalConfig


   <next-page>
      <page-path>/sample/workflow/purchase/plugin/authority/item_total/itemTotalConfig.jsp</page-path>
   </next-page>
</service>
```

extend設定

前出の plugin.xml で＜extend＞として指定したプログラムを作成します。
処理対象者を決定する際に実行されるプログラムです。
ここで指定するプログラムには、次の３つのメソッドを実装する必要があります。

メソッド	概要
execute(WorkflowAuthorityParameter workflowParam, WorkflowMatterParameter matterParam)	処理対象者を取得するメソッド 対象のノードに案件が到達したときに実行されます。

メソッド	概要
getTargetUserList(WorkflowAuthorityParameter workflowParam,WorkflowMatterParameter matterParam, WorkflowSortCondition[] sort)	処理対象ユーザの一覧を取得するメソッド ロケール単位にユーザモデルのリスト（List<UserDataModel>）を返却する必要があります。 [案件操作]-[ノード編集]画面の「状況確認」ボタン押下時に表示される[対象者状況確認]画面で使用されます。
getDisplayName(WorkflowAuthorityParameter workflowParam)	処理対象者プラグインの名称を取得するメソッド ロケール単位に取得してください。 プラグインの名称を表示するため使用されます。

サンプルプログラムとしては、下記を用意しています。

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/plugin/authority/item_total/WorkflowAuthorityExecEventListener.js>
```

- JavaEE開発モデル

```
<%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample
        /workflow/purchase/plugin/authority/item_total/WorkflowAuthorityExecEventListener.java>
```

画面入力情報の保持

ここで記載している内容は、次の観点において共通です。

- 開発モデル
- クライアントタイプ

申請画面、一時保存画面、申請（起票案件）画面、再申請画面、処理画面、確認画面において、「閉じる」リンク（PC用画面）または 「戻る」リンク（スマートフォン用画面） によって各画面を閉じた後に画面の再表示を行ったとき、入力内容を保持した状態で画面表示されます。

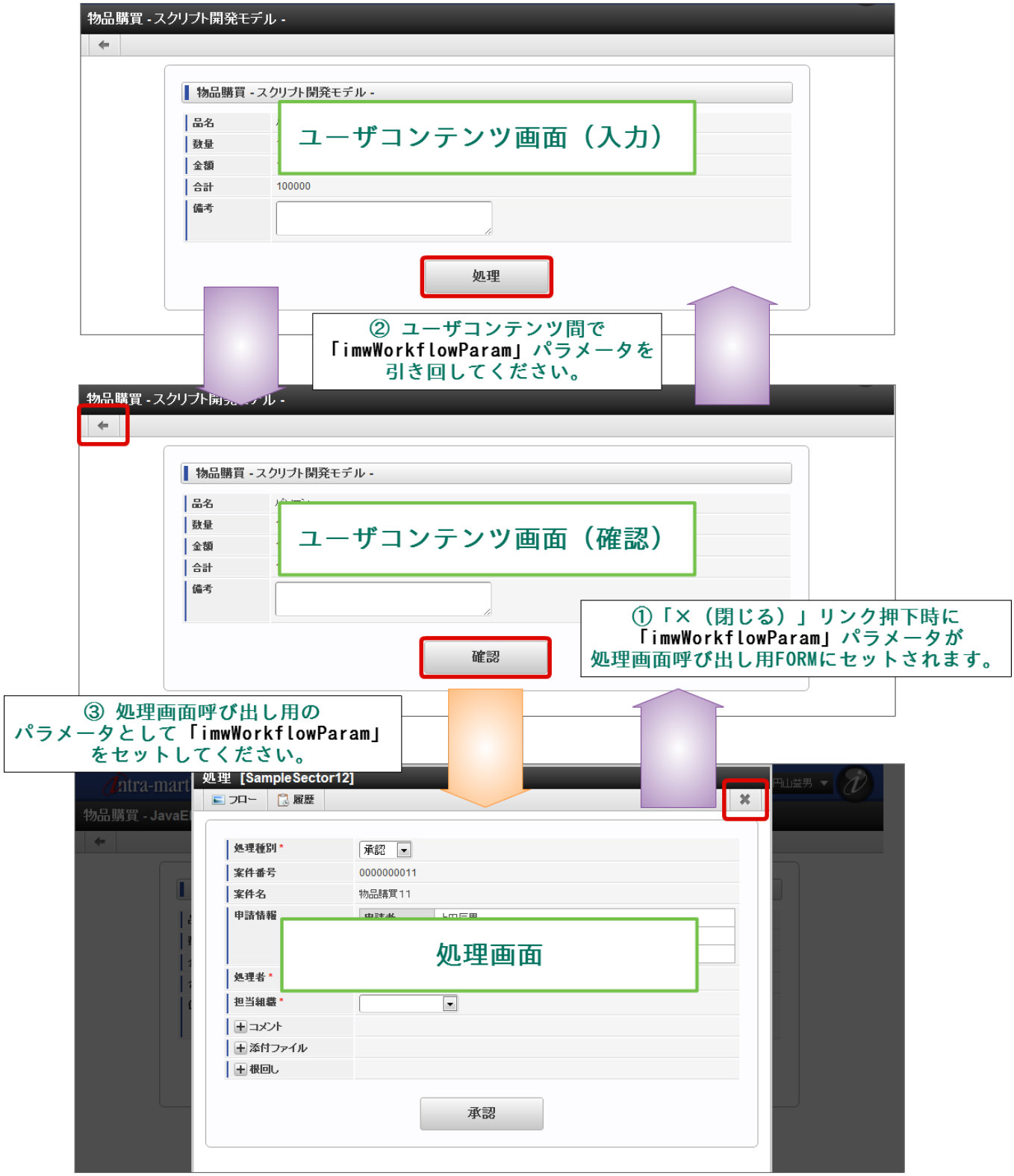
当機能の仕様概要は以下の通りです。

- 各処理画面の「閉じる」「戻る」リンク押下時に、呼出元ユーザコンテンツ内の画面呼出用タグライブラリによって生成されたFORMに対して「imwWorkflowParams」 というパラメータ名のhiddenタグを追加し、そのタグに入力情報を格納
- 再度画面表示した際にリクエストパラメータとして 「imwWorkflowParams」 が含まれている場合、画面の初期表示処理で保持情報による復元表示を実行

リクエストパラメータの受け渡しによって入力情報再表示が行われるため、ユーザコンテンツが単一画面構成の場合は意識する必要がありませんが、複数画面で構成されている場合は以下対応が必要です。

各処理画面を閉じてからユーザコンテンツ間の画面遷移が行われ、その後入力内容を保持した状態で各処理画面の再表示を行う必要がある場合、「imwWorkflowParams」パラメータをユーザコンテンツ間で引き回し、各処理画面表示用のタグライブラリのコンテンツ内に「imwWorkflowParams」パラメータをhiddenタグで明示的に記述してください。

＜ユーザコンテンツ　複数画面構成での実装イメージ＞



呼び出し画面からのコールバック関数の指定

ここで記載している内容は、次の観点において共通です。

- 開発モデル
- クライアントタイプ

申請画面、一時保存画面、申請（起票案件）画面、再申請画面、処理画面、確認画面において、「閉じる」リンク（PC用画面） または「戻る」リンク（スマートフォン用画面）によって各画面を閉じる際のコールバック関数を指定可能です。
またコールバック関数は、「*処理完了後の画面遷移*」 に記載のパラメータ（imwNext～）の指定を行っていない場合、IM-Workflowで提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の処理完了後にも実行されます。

呼出元のユーザコンテンツ画面の関数を実行する方法について説明します。

実装例

サンプルとして提供されている「物品購買」の申請書において、GreyBoxで表示される申請画面の閉じる処理が実行された際に、「物品購買」の申請書で定義された関数をコールバック関数として実行する例です。

なお、サンプルはPC用画面のみ用意しています。
スマートフォン用画面の場合も全体の流れは同じです。実装中で使用するタグライブラリや Client-side JavaScript API が異なることに注意してください。

下記のプログラムが、コールバック関数の実行を行うための処理が記述されたプログラムです。

- スクリプト開発モデル

  ```
  <./jssp/src/sample/im_workflow/purchase/screen/apply_callback.html>
  ```

- JavaEE開発モデル

  ```
  <（展開したwar）/sample/im_workflow/purchase/apply_callback.jsp>
  ```

上記ファイルを、以下のファイル名に変更し、上書き保存することで、申請画面において本機能の動作確認を行うことが出来ます。

- スクリプト開発モデル

  ```
  <./jssp/src/sample/im_workflow/purchase/screen/apply.html>
  ```

- JavaEE開発モデル

  ```
  <（展開したwar）/sample/im_workflow/purchase/apply.jsp>
  ```

以下のような処理を記述することで、コールバック関数の実行を行うことが出来ます。

```
<imart type="head">

<imart type="workflowOpenPageCsjs"/>
<script type="text/javascript">

function onClickOpenPage(pageType) {
    if (pageType != "1") {
        if(!inputCheck()) {
            return;
        }
    }

    // 申請画面を表示する際に、コールバック関数を引数で指定します。
    workflowOpenPage(pageType, callbackFnc);
}

// 申請画面を「閉じる」または「戻る」際に実行する処理を記載します。
function callbackFnc() {
    alert("Callback function is executed.");
}
.
.
.
<imart type="form" name="backForm" method="POST" page=$data.imwCallOriginalPagePath>
    <imart type="hidden" imwCallOriginalParams=$data.imwCallOriginalParams />
</imart>
```

IM-Workflow で提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の処理完了後にコールバック関数が実行された場合、コールバック関数は処理された案件の情報を引数として受け取ることができます。

```
function callbackFnc(result) {
    alert("Callback function is executed.");
    alert( result.imwSystemMatterId );    // システム案件ID
    alert( result. imwUserDataId );    // ユーザデータID
}
```

処理種別と受け取ることのできる情報の関係は以下の通りです。

処理種別	システム案件ID imwSystemMatterId	ユーザデータID imwUserDataId
申請	○	-
再申請	○	-
申請（起票案件）	○	-
一時保存	-	○
処理	○	-
確認	○	-

＜「○」 ： 取得可能 ／ 「-」 ： 取得不可能＞

標準画面を非同期で実行する場合の注意点

IM-Workflow バージョン 8.0.4より標準画面の処理を非同期に行う機能が追加されました。
この機能が有効の場合、標準画面の呼び出し元画面で指定されたコールバック関数の振る舞いが異なります。

IM-Workflow で提供する各処理が非同期として受付された時点で処理完了を各処理画面に通知します。
ほぼ処理開始の時点で通知するイメージです。
従いまして、標準画面の呼び出し元画面で指定されたコールバック関数が実行された時点では各処理が完了していない可能性が高いです。
そのため処理種別が申請の場合は、システム案件IDを受け取ることはできません。

特記事項

Contents
- IM-Workflow バージョン8.0.2 における改善

IM-Workflow バージョン8.0.2 における改善

IM-Workflow バージョン8.0.2 から、連続処理／連続確認中のコールバック呼び出しの動作仕様を改善しています。

- IM-Workflow バージョン8.0.1 までの動作仕様
  - コールバック関数の指定有無に関わらず、コールバック関数は実行されません。
- IM-Workflow バージョン8.0.2 以降の動作仕様
  - 「*処理完了後の画面遷移*」 に記載のパラメータ（imwNext～）を指定しない場合には各処理完了後にコールバック関数が実行されます。

※ IM-Workflow スマートフォン では、連続処理／連続確認機能が存在しないため、上記を意識する必要はありません。

処理完了後の画面遷移

ここで記載している内容は、次の観点において共通です。

- 開発モデル
- クライアントタイプ

IM-Workflow で提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の処理後に、任意の画面に遷移する

ことが可能です。

遷移先を指定するためのパラメータ

IM-Workflow で提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の呼び出し時、
「workflowOpenPage」タグの属性に下記パラメータを記述すると、処理完了後の遷移先を指定する事ができます。

No	パラメータ（物理名）	省略	説明
1	imwNextScriptPath	可	処理完了後に遷移する画面のスクリプトパス 処理後の遷移先がスクリプト開発画面の場合に指定が必要です。
2	imwNextApplicationId	可	処理完了後に遷移する画面のアプリケーションID 処理後の遷移先がjavaEE開発画面の場合に指定が必要です。
3	imwNextServiceId	可	処理完了後に遷移する画面のサービスID 処理後の遷移先がjavaEE開発画面の場合に指定が必要です。
4	imwNextPagePath	可	処理完了後に遷移する画面のページパス 処理後の遷移先がJSP or Servletの場合に指定が必要です。

実現したい画面遷移によって指定する属性を決定してください。

- 処理後にユーザコンテンツの呼出元一覧画面に遷移したい場合
  - 「imwNextScriptPath」に、一覧から渡された「imwCallOriginalPagePath」を指定してください。
    ※ 連続処理、連続確認の場合は、次の案件ノードがあれば、該当のユーザコンテンツに遷移します。
    次の案件ノードがなければ、呼出元一覧画面に遷移します。
- 処理後に任意の画面に遷移したい場合
  - 「imwNext～」に、遷移先の画面パスを指定してください。
- 処理後にユーザコンテンツ独自のコールバック関数を実行して処理画面を閉じる、または処理画面を閉じることのみ実行したい場合
  - 「imwNext～」に何も設定しないでください。

遷移先画面が受け取ることのできるリクエストパラメータ

遷移元の処理画面の種類によって、遷移先では下記の情報をリクエストパラメータとして受け取る事ができます。

No	遷移元処理画面	パラメータ（物理名）	パラメータ（論理名）	備考
1	申請／再申請／申請（起票案件）／処理／確認	imwSystemMatterId	システム案件ID	-
2	一時保存	imwUserDataId	ユーザデータID	-
3	すべて	imwCallOriginalParams	呼出元パラメータ	ユーザコンテンツが一覧画面からリクエストパラメータとして受け取ることのできる値と同じ値が受け取れます。
4	すべて ※連続処理／連続確認中の場合のみ	imwCallOriginalPagePath	呼出元ページパス	ユーザコンテンツが一覧画面からリクエストパラメータとして受け取ることのできる値と同じ値が受け取れます。

特記事項

Contents

IM-Workflow バージョン8.0.2 における改善

IM-Workflow バージョン8.0.2 から、連続処理／連続確認中の画面遷移仕様を改善しています。

- IM-Workflow バージョン8.0.1 までの動作仕様
  - 処理完了後の遷移先指定は無視されます。
  - 処理完了後の遷移先指定の有無に関わらず、処理完了後は次の案件のユーザコンテンツが表示されます。
- IM-Workflow バージョン8.0.2 以降の動作仕様
  - 処理完了後の遷移先指定が行われている場合、処理完了後は指定された画面に遷移します。

※ IM-Workflow スマートフォン では、連続処理／連続確認機能が存在しないため、上記を意識する必要はありません。

意図しないURLに対するバリデーション

「imwNextPagePath」には、ルーティング設定によってマッピングされたURLが指定されることを前提としていますが、絶対URLを指定された場合でも動作します。
これは処理完了後の画面遷移先として IM-Workflow と連携する外部システムのURLも設定可能とするためで、 IM-Workflow Ver.7.2.x との互換性を保つための仕様です。

「workflowOpenPage」タグライブラリ、および「spWorkflowOpenPage」タグライブラリでは、明示的に外部システム等のURLを指定される可能性のある「imwNextPagePath」と、「imwNextApplicationId」および「imwNextServiceId」に対して、テナント単位で設定が可能な「セーフURLリスト」に存在するかチェックを行います。

チェックには、intra-mart Accel Platform が提供するAPI 「SafeUrlManager#isSafe(String url)」 を使用しています。

ワークフローエンジンとしては上述の通りURLバリデーションチェックを行っていますが、同様のチェックがユーザコンテンツで必要となる場合があります。

たとえば、ユーザコンテンツでの画面遷移処理（例：「imwCallOriginalPagePath」を利用して一覧へ戻る）における遷移先パスが何らかのタイミング（ユーザコンテンツ間での画面遷移の最中など）で意図しないURLに改ざんされた場合、改ざん後のURLへの遷移を防止するためには、URLバリデーションチェック処理を独自に実装していただく必要があります。

チェックには 「SafeUrlManager#isSafe(String url)」 が利用可能です。
APIの詳細については 「 APIドキュメント 」 を参照してください。
また、セーフURLの設定方法については 「 テナント管理者操作ガイド 」 を参照してください。

「SafeUrlManager#isSafe(String url)」を利用したURLバリデーションチェックのサンプル実装（スクリプト開発モデル）を示します。

```
var safeUrlManager = new SafeUrlManager();
var result = safeUrlManager.isSafe(url);
if (result.error) {
   // URLチェック実行に失敗した場合の処理
   Transfer.toErrorPage(
      {
         "title" : "URL バリデーションエラー",
         "message" : "URLチェック実行に失敗しました。管理者に連絡してください。"
      }
   );
}
if (!result.data) {
   // 指定したURLがセーフURLリストに存在しない場合の処理
   Transfer.toErrorPage(
      {
         "title" : "URL バリデーションエラー",
         "message" : "安全ではないURLが指定されています。管理者に連絡してください。"
      }
   );
}
```

ユーザコンテンツと連続処理／連続確認の連携方法

ここで記載している内容は、次の観点において共通です。

- 開発モデル

「workflowOpenPage」タグの属性「imwNext～」を指定してIM-Workflow処理後に任意の画面（呼出元一覧画面以外の画面）に遷移した場合、または「workflowOpenPage」タグの属性「imwNext～」を指定せずに IM-Workflow 処理後の画面遷移を行わない場合の、ユーザコンテンツと連続処理／連続確認の連携方法について説明します。

なお、 IM-Workflow スマートフォン では、連続処理／連続確認機能が存在しないため、ここで記載の事項を意識する必要はありません。

連続処理／連続確認を継続実行する

連続処理、連続確認を継続し、次の案件ノードに対応するユーザコンテンツ画面に遷移するためには、次の実装を行ってください。

- 一覧から渡された「imwCallOriginalPagePath」が指し示す画面に遷移してください。
- 一覧から渡された「imwCallOriginalParams」を遷移先画面へのリクエストパラメータとして設定してください。

連続処理／連続確認を中断する

連続処理、連続確認を中断し、一覧から渡された「imwCallOriginalPagePath」が指し示す画面に遷移するためには、「imwCallOriginalPagePath」への画面遷移の前に、セッションからクライアント固有情報を削除してください。

セッションキーは、 **“IMW_LAST_PROCESSED_MATTER_INFO_IN_SERIAL”** です。

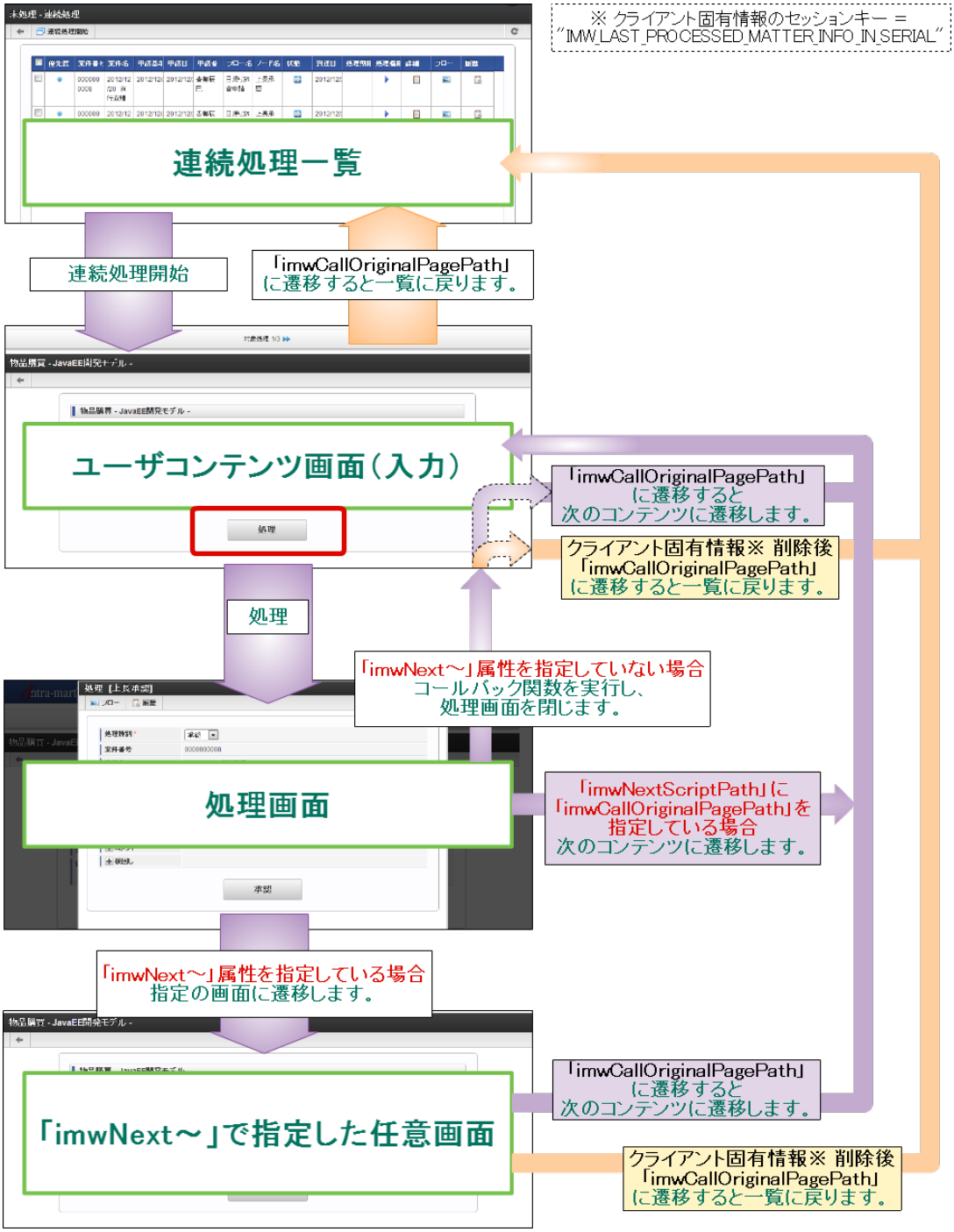
スクリプト開発モデルでセッションからクライアント固有情報を削除する場合は、次のメソッドを利用します。

```
Client.remove(Strign key)
```

JavaEE開発モデルでセッションからクライアント固有情報を削除する場合は、次のメソッドを利用します。

```
HttpSession.removeAttribute(java.lang.String name)
```

連続処理の場合の画面遷移を図示します。なお、連続確認の場合も画面遷移は同様です。



PC版ユーザコンテンツをスマートフォン用画面としても利用する

ここで記載している内容は、次の観点で共通です。

- 開発モデル

PC版ユーザコンテンツとして作成した画面を、スマートフォン用画面として動作させる方法を説明します。

この方法を採用すると、PC版ユーザコンテンツとスマートフォン版ユーザコンテンツをひとつの画面でまかなうことも可能です。

ただし、スマートフォン端末でPC版ユーザコンテンツを表示した場合、さまざまな制限事項があります。
そのため、PC版のユーザコンテンツとスマートフォン版のユーザコンテンツは、それぞれ独自に実装することを推奨します。

詳しくは「 リリースノート 」 の制限事項を参照してください。

必要な作業

マスタ設定と、実装の修正を行う必要があります。

Contents

マスタ定義のスマートフォン用画面設定を行う

「サイトマップ」-「マスタ定義-コンテンツ定義」 より、設定対象のコンテンツ定義に対し「画面」を選択しスマートフォン用の画面定義を新規作成、または編集してください。
画面パスとして、スマートフォン用の画面として利用するPC版ユーザコンテンツを指定してください。

以降、必要に応じて、フロー定義の個別設定などを行ってください。

マスタ定義の新規作成、編集手順は「 IM-Workflow 管理者操作ガイド 」を参照してください。

以上を行ったうえで、スマートフォン端末で対象のフローの申請画面を表示すると、PC版ユーザコンテンツが表示されます。

ただし、この状態ではPC版ユーザコンテンツにスマートフォン用の画面テーマが適用されてしまい、レイアウトが崩れてしまう場合があります。

そこで、PC版ユーザコンテンツの実装に対して修正を行います。

クライアントタイプをPCに切り替える

ユーザコンテンツの実装において、クライアントタイプをPCに切り替える必要があります。

画面表示を行う際のサーバサイドロジックにおいて、ClientTypeSwitcher.oneTimeSwitchTo を利用し、ユーザコンテンツとして表示する画面のクライアントタイプを無条件でPCに切り替えてください。

```
ClientTypeSwitcher.oneTimeSwitchTo("pc");
```

ClientTypeSwitcher について、詳細は「 APIドキュメント 」を参照してください。

実装の修正を行う対象は、スマートフォン用画面として動作させるPC版ユーザコンテンツすべてです。

以上を行うことで、レイアウトが崩れることなくPC版ユーザコンテンツをスマートフォン端末で表示することができます。

この状態で、 IM-Workflow が提供する案件の各処理画面（GreyBox上に表示される画面）が正常に表示されない（画面が表示されない、画面レイアウトが崩れる）場合のみ、以降の作業を行ってください。

補足修正

ワークフロー処理を実行する画面を表示するためのClient-side JavaScript API 「workflowOpenPage」 の引数として、各種一覧画面からリクエストパラメータとして受け取った「画面種別（imwPageType）」を **そのまま** 受け渡している場合、修正が必要です。

クライアントタイプがスマートフォンの場合、各種一覧からは画面種別としてスマートフォン用画面の値が受け渡されます。
「workflowOpenPage」の引数には、PC用の画面種別の値を受け渡してください。

画面種別のクライアントタイプ別対応は下表のとおりです。

画面種別 クライアントタイプ	申請	一時保存	申請 （起票案件）	再申請	処理	確認
PC	0	1	2	3	4	5
スマートフォン	10	11	12	13	14	15

ユーザコンテンツ画面への不正な直接アクセスを抑止する

ここで記載している内容は、次の観点で共通です。

- クライアントタイプ

IM-Workflow 標準機能では、IM-Workflow の各種一覧画面からユーザコンテンツ画面に遷移することができます。
この場合、IM-Workflow の標準機能は、ログインユーザが対象のコンテンツ画面の表示権限を保持しているか判定を行い、権限がない場合はエラー画面を表示します。

上記の通常遷移時以外の場合、IM-Workflow の標準機能によるユーザコンテンツ画面の表示権限の判定が行われません。
例えば、ユーザコンテンツ画面のURLに直接アクセスが行われた場合、IM-Workflow の標準機能による表示権限の判定が行われないため、ユーザコンテンツ画面のつくりによっては、表示権限を持たないユーザにユーザコンテンツ画面の内容を閲覧されてしまう可能性があります。

上記の状態でも、ユーザコンテンツ画面の表示後に各種処理（申請、承認など）を実行するタイミングでは、IM-Workflow の標準機能による処理権限の判定が行われるため、不正な処理が実行されてしまうことはありません。

ただし、表示権限のないユーザにユーザコンテンツ画面を閲覧されてしまうことが運用上の問題となる場合には、以降の対応を行うことにより、ユーザコンテンツ画面への不正な直接アクセスを抑止することが可能です。

対象者

以下の対応を検討している方を対象としています。

- ユーザコンテンツ画面へのアクセス権限について、セキュリティ強化を図りたい方
- intra-mart Accel Platform の認可機構を利用し、ユーザコンテンツ画面の表示権限を制御したい方
- すでに実施済みのセキュリティ対応について、IM-Workflow 標準の方法に切替えたい方

対象パス種別

ユーザコンテンツ定義の画面定義において、以下のパス種別として登録する画面を対象としています

- JavaEE開発モデル
- JPS or Servlet

パス種別が「スクリプト開発モデル」であるユーザコンテンツ画面については、スクリプト開発のセキュアな機構で直接のアクセスが抑止されているため、対応の必要はありません。

対応方法

Contents
- 認可設定
- ユーザコンテンツ画面の追加開発（カスタマイズ）

対応方法としては、以下のいずれかを選択可能です。

1. 認可設定
2. ユーザコンテンツ画面の追加開発（カスタマイズ）

どちらの方法を選択すべきかは、下表を参照してください。

要件	推奨する対応方法
ユーザコンテンツ画面の実装を改修することができない	認可設定
アクセス権設定を認可機構で統一的に扱いたい	認可設定
IM-Workflowの標準機能と同等のユーザコンテンツ画面表示権限判定を実行したい	ユーザコンテンツ画面の追加開発（カスタマイズ）

以降では、それぞれの対応方法の詳細について説明します。
運用形態や影響範囲を考慮の上、適当な方法を選択してください。

認可設定

認可機構により、ユーザコンテンツ画面を「リソース」として登録し、アクセス権設定を行います。
認可の仕様については「 認可仕様書 」を参照してください。

認可設定による対応の特徴は以下の通りです。

- ユーザの権限を認可機構で集約して管理することが可能です。
- ユーザコンテンツ画面の実装の改修は不要です。
- IM-Workflow のルート定義で設定される処理対象者を包含する範囲で認可設定を行う必要があります。
  - 例として、同一の申請用ユーザコンテンツ画面を、フローAとフローBで流用している場面を想定します。
    - フローAはルートAを利用しており、申請ノードの処理対象者は「サンプル課１１」です。
    - フローBはルートBを利用しており、申請ノードの処理対象者は「サンプル部門０２」です。
  - この場合、申請用ユーザコンテンツ画面の認可設定としては、「サンプル課１１」と「サンプル部門０２」からの実行を許可する設定を行う必要があります。

以降では、 IM-Workflow のコンテンツ定義における「パス種別」ごとに、認可設定を行う際の参考となるドキュメントを紹介します。

パス種別「JavaEE開発モデル」の場合

「 移行ガイド 」 の「 im-JavaEE Framework 」の認可設定部分を参照してください。

パス種別「JSP or Servlet」の場合

TERASOLUNA Server Framework for Java (5.x) を利用して実装している場合、「TERASOLUNA Server Framework for Java (5.x) プログラミングガイド 」の 「 認可 」を参照してください。

ユーザコンテンツ画面の追加開発（カスタマイズ）

IM-Workflow が提供するタグライブラリ、またはAPIを利用し、ユーザコンテンツ画面の表示権限を判定します。

ユーザコンテンツ画面の表示権限とは、特定の案件を処理、または参照する場合に利用されるユーザコンテンツ画面を、IM-Workflow 標準の各種一覧画面（フロー一覧、未処理一覧など）から表示することのできる権限のことを指します。

ユーザコンテンツ画面の追加開発による対応の特徴は以下の通りです。

- IM-Workflow の標準機能と同等のユーザコンテンツ表示権限判定を行うことが可能です。
- ユーザコンテンツ画面の実装の改修が必要です。

対応方法としては、以下のいずれかを選択可能です。

- タグライブラリによる対応
- APIによる対応

タグライブラリによる対応

クライアントタイプ別で、ユーザコンテンツ画面の表示権限判定用タグライブラリが用意されています。

- クライアントタイプ＝PC
  - 「workflowUserCnotentsAuth」
- クライアントタイプ＝スマートフォン

- 「spWorkflowUserCnotentsAuth」

ユーザコンテンツ画面で上記のタグライブラリを利用するのみで、ユーザコンテンツ画面の表示権限の判定を行うことが可能です。
表示権限がない場合、HTTP 403エラーが発生します。

タグライブラリによる対応を行う場合は、「 APIドキュメント 」を併せて参照してください。

推奨実装

以下のルールで実装を行うことを推奨します。

1. ユーザコンテンツ画面に「*リクエストパラメータ* 」として受け渡されたパラメータを、すべてリクエストスコープの属性として格納します。
2. タグライブラリを引数省略の形式で利用します。

   ユーザコンテンツ画面が複数画面構成の場合、追加で下記実装を行うことを推奨します。
3. 「*リクエストパラメータ* 」としてユーザコンテンツ画面に受け渡されたパラメータを、ユーザコンテンツ画面間を遷移する際に引き回します。そのうえで、上記の1、2の実装を各ユーザコンテンツ画面で行います。

上記のルールを採用することにより、以下の実装上のメリットがあります。

- タグライブラリを統一的な手法で組み込むことが可能です。
- ひとつのユーザコンテンツ画面が複数の画面種別に対応した実装となっている場合でも、画面種別の差異によってタグライブラリに指定するパラメータを切り替える必要がなくなります。

実装例

IM-Workflow のJavaEE開発モデルの以下のサンプルをもとに、推奨実装の1、2の例を紹介します。

- クライアントタイプ＝PCの場合
  - アプリケーションID　：　imw_sample_purchase
  - サービスID　：　apply
- クライアントタイプ＝スマートフォンの場合
  - アプリケーションID　：　imw_sp_sample_purchase
  - サービスID　：　apply

このサンプルでは、以下の画面種別に対応しています。

- 申請画面
- 一時保存画面
- 申請（起票案件）画面
- 再申請画面

それでは、順を追って実装例を示します。

1. ユーザコンテンツ画面に「*リクエストパラメータ* 」として受け渡されたパラメータを、すべてリクエストスコープの属性として格納します。

   HttpServletRequest#setAttribute(String, String) を利用し、リクエストスコープの属性にパラメータを格納します。
   ※サンプルではあらかじめ実装されています。

   ```
   <%サンプルプログラムディレクトリ%
     /jp/co/intra_mart/sample/workflow/purchase/controller/service/ApplyServiceTransition.java>
   ```

```
1  package jp.co.intra_mart.sample.workflow.purchase.controller.service;
2
3  import javax.servlet.http.HttpServletRequest;
4
5  import jp.co.intra_mart.framework.base.service.DefaultTransition;
6  import jp.co.intra_mart.framework.base.service.ServicePropertyException;
7  import jp.co.intra_mart.framework.base.service.TransitionException;
8
9  public class ApplyServiceTransition extends DefaultTransition {
10
11    @Override
12    public String getNextPage() throws ServicePropertyException, TransitionException {
13      final ApplyServiceResult serviceResult = (ApplyServiceResult) getResult();
14      return getNextPagePath(serviceResult.getNextPageServiceId());
15    }
16
17    @Override
18    public void setInformation() throws TransitionException {
19      final HttpServletRequest request = getRequest();
20      final ApplyServiceResult serviceResult = (ApplyServiceResult) getResult();
21      // 受け渡されたパラメータをリクエストスコープに設定します。
22      request.setAttribute("imwGroupId", serviceResult.getImwGroupId());
23      request.setAttribute("imwUserCode", serviceResult.getImwUserCode());
24      request.setAttribute("imwPageType", serviceResult.getImwPageType());
25      request.setAttribute("imwUserDataId", serviceResult.getImwUserDataId());
26      request.setAttribute("imwSystemMatterId", serviceResult.getImwSystemMatterId());
27      request.setAttribute("imwNodeId", serviceResult.getImwNodeId());
28      request.setAttribute("imwArriveType", serviceResult.getImwArriveType());
29      request.setAttribute("imwAuthUserCode", serviceResult.getImwAuthUserCode());
30      request.setAttribute("imwApplyBaseDate", serviceResult.getImwApplyBaseDate());
31      request.setAttribute("imwContentsId", serviceResult.getImwContentsId());
32      request.setAttribute("imwContentsVersionId", serviceResult.getImwContentsVersionId());
33      request.setAttribute("imwRouteId", serviceResult.getImwRouteId());
34      request.setAttribute("imwRouteVersionId", serviceResult.getImwRouteVersionId());
35      request.setAttribute("imwFlowId", serviceResult.getImwFlowId());
36      request.setAttribute("imwFlowVersionId", serviceResult.getImwFlowVersionId());
37      request.setAttribute("imwCallOriginalParams", serviceResult.getImwCallOriginalParams());
38      request.setAttribute("imwCallOriginalPagePath", serviceResult.getImwCallOriginalPagePath());
39
40      request.setAttribute("item_name", serviceResult.getItemName());
41      request.setAttribute("item_amount", serviceResult.getItemAmount());
42      request.setAttribute("item_price", serviceResult.getItemPrice());
43      request.setAttribute("item_total", serviceResult.getItemTotal());
44      request.setAttribute("item_comment", serviceResult.getItemComment());
45    }
46 }
```

2. タグライブラリを引数省略の形式で利用します

   タグライブラリを画面実装に追加します。

   - クライアントタイプ＝PCの場合

   <（展開したwar）/sample/im_workflow/purchase/apply.jsp>

```
<%@ page contentType="text/html; charset=utf-8" pageEncoding="utf-8" %>
<%@ taglib prefix="imartj2ee" uri="http://www.intra-mart.co.jp/taglib/core/framework" %>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui" %>
<%@ taglib prefix="imart" uri="http://www.intra-mart.co.jp/taglib/core/standard" %>
<%@ taglib prefix="workflow" uri="http://www.intra-mart.co.jp/taglib/imw/workflow" %>
<imartj2ee:HelperBean id="bean"
class="jp.co.intra_mart.sample.workflow.purchase.controller.view.CommonHelperBean"/>

<!-- タグライブラリ（引数省略） -->
<workflow:workflowUserContentsAuth />

<imui:head>
.
.
.
```

   - クライアントタイプ＝スマートフォンの場合

   <（展開したwar）/sample/im_workflow_smartphone/purchase/apply.jsp>

```
<%@ page contentType="text/html; charset=utf-8" pageEncoding="utf-8" %>
<%@ taglib prefix="imartj2ee" uri="http://www.intra-mart.co.jp/taglib/core/framework" %>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui" %>
<%@ taglib prefix="imart" uri="http://www.intra-mart.co.jp/taglib/core/standard" %>
<%@ taglib prefix="workflow" uri="http://www.intra-mart.co.jp/taglib/imw/workflow-smartphone" %>
<imartj2ee:HelperBean id="bean"
class="jp.co.intra_mart.sample.workflow.purchase.controller.view.CommonHelperBean"/>

<!-- タグライブラリ（引数省略） -->
<workflow:spWorkflowUserContentsAuth />

<imui:head>
```

APIによる対応

タグライブラリによる対応では実現できない要件がある場合は、ユーザコンテンツ画面の表示権限の判定APIを利用することで、任意の動作をさせることが可能です。
具体的には、次のような場合を想定します。

- 業務ロジックのとの兼ね合いで、タグライブラリを利用することができない場合
- 表示権限がないと判定された際、HTTP 403エラーではなく任意の処理を行いたい場合

対応するAPIは「jp.co.intra_mart.foundation.workflow.util.auth.WorkflowAuthUtil」です。
権限判定の結果は boolean 値で返却されるため、結果をうけて任意の処理を行うことが可能です。
詳細は「 APIドキュメント 」を参照してください。

動的処理対象者設定機能

「動的処理者対象者設定」機能とは、申請/処理画面のフロー設定項目をユーザコンテンツ画面からのリクエストパラメータで設定できる機能です。

ここで記載している内容は、次の観点で共通です。

- 開発モデル
- クライアントタイプ

なお、当機能は以下のバージョン以降で利用可能です。

- PC版 IM-Workflow 2014 Winter(Iceberg) 8.0.9 PATCH 001
- スマートフォン版 IM-Workflow 2015 Summer(Karen) 8.0.11

機能概要

「動的処理対象者設定」機能では、以下を実現可能です。

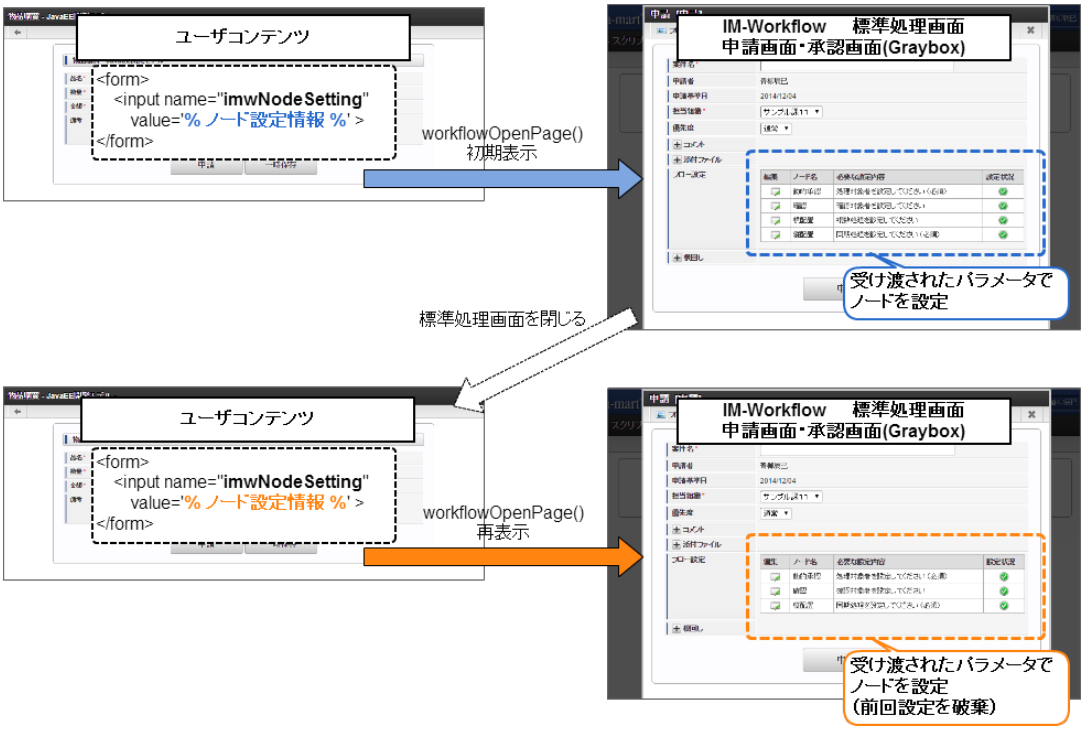
- 処理対象者の決定
  - ビジネスロジックによって決定した処理対象者を、標準処理画面で設定可能なノードに反映する
- 処理対象者検索時の暗黙条件の指定
  - 標準処理画面で設定可能なノードにおいて、利用者が処理対象者を検索・設定する際の暗黙条件を指定し、検索結果の絞り込みを行う

当機能によって設定が可能なノードは以下の通りです。

- 動的承認ノード
- 確認ノード
- 横配置ノード
- 縦配置ノード

利用方法

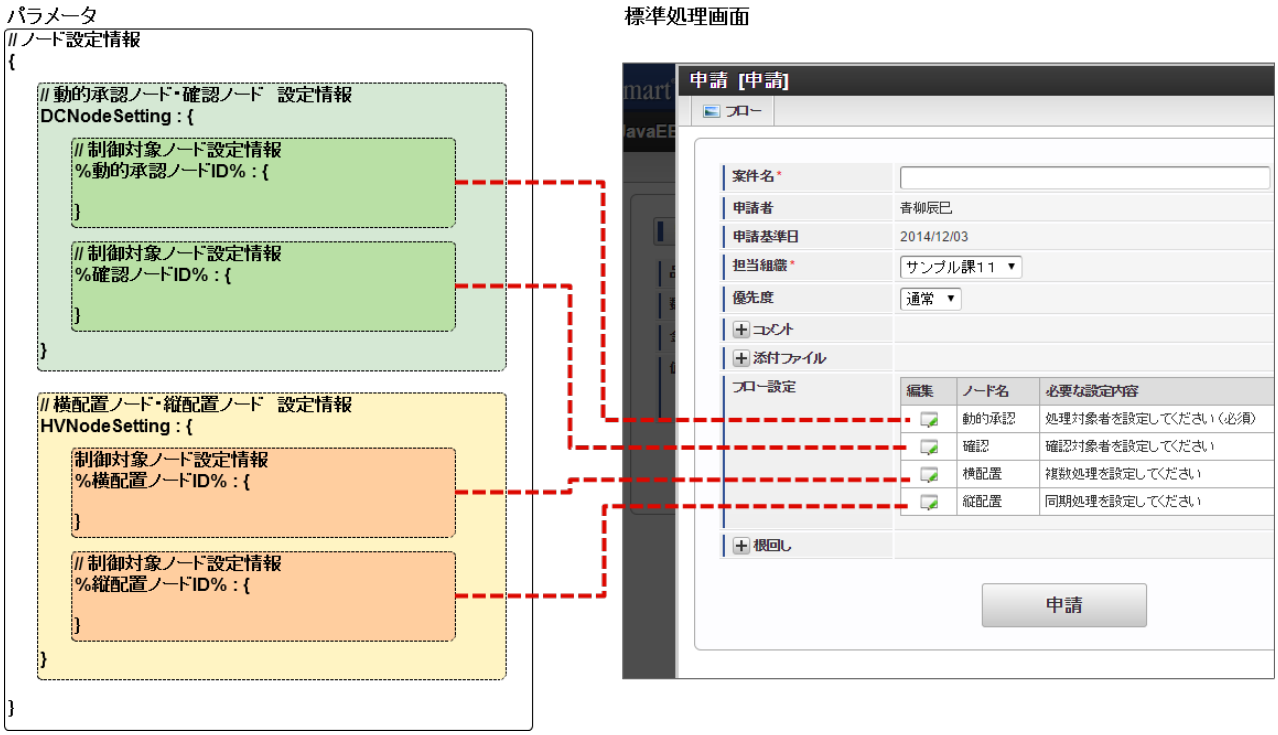
IM-Workflow の標準処理画面を表示する際に、ユーザコンテンツからパラメータを送信することで、フロー設定、およびノード設定を行います。
パラメータの送信方法は、設定用パラメータオブジェクトをJSON文字列に変換し、「imwNodeSetting」というキーで標準処理画面に受け渡します。



標準処理画面を閉じた後、ユーザコンテンツから標準処理画面を再度表示する際に改めてパラメータを受け渡した場合、前回の設定情報を破棄し、新しく受け渡されたパラメータによって設定されます。

標準処理画面を閉じた後、ユーザコンテンツから標準処理画面を再度表示する際にパラメータを受け渡さない場合、前回の設定情報を保持して再表示されます。

パラメータとフロー設定、ノード設定の概念図を以下に示します。



設定対象のノード単位で情報を作成し、ノード種別（動的承認ノード・確認ノード / 横配置ノード・縦配置ノード）でまとめ、最終的にひとつのパラメータとして生成し、標準処理画面に受け渡すことで各種設定を行います。

当機能を利用するノードと設定される対象のノードは、あらかじめフロー定義のノード設定において、処理対象者設定可能ノードの設定が行われている必要があります。
つまり、フロー設定を行う場合は、当機能を利用する・しないに関わらず、上図のように標準処理画面でフロー設定としてノードが表示される状態となるよう、フロー定義のノード設定が行われている必要があります。
処理対象者設定可能ノードの設定が行われていないノードに対してパラメータを送信しても、パラメータは無視され、当機能は実行されません。

以降の章で、具体的な利用方法やパラメータの詳細について説明します。

利用例

当機能の利用例を、ユーザコンテンツから送信するパラメータの例とともに紹介します。
なお、この章では申請時の例のみ掲載していますが、承認時にも当機能を利用することは可能で、申請時と流れは同様です。
また、この章ではスクリプト開発モデルでの実装例を紹介します。

処理対象者設定

動的承認ノードと横配置ノードに対し、処理対象者を設定する場合を例示します。

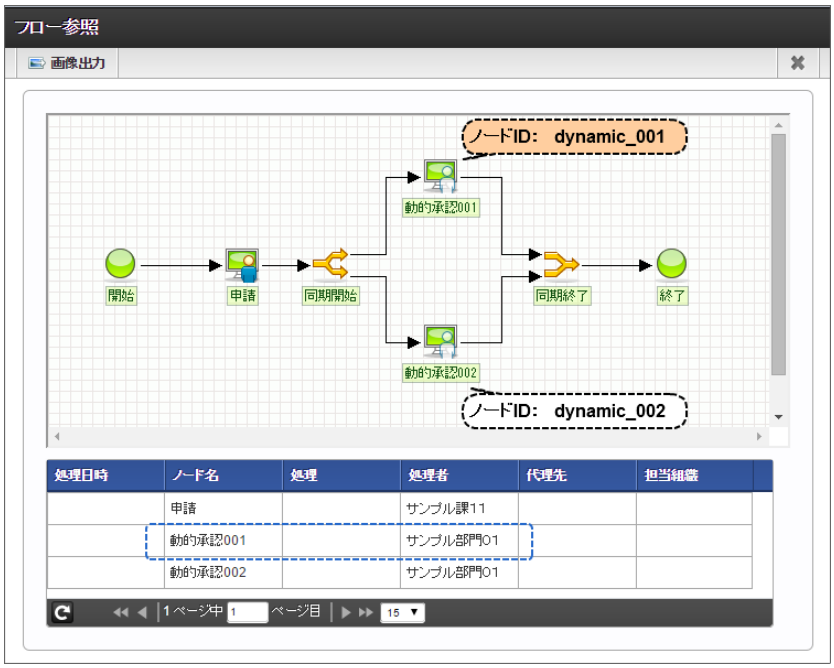
Contents

動的承認ノードの処理対象者をシステムで決定する

ユーザコンテンツ側で決定した処理対象者を、動的承認ノードに反映します。
また、標準処理画面からは動的承認ノードの設定を行わせない制御を実施します。

前提

申請時に動的承認ノードの設定を行います。
利用するフローは次の通りです。動的承認ノードがふたつ存在し、処理対象者は両方とも「サンプル部門０１」に設定されています。
また、ふたつの動的承認ノードの設定を申請ノードで行えるようフロー定義を設定します。
今回は、「動的承認001」（ノードID：dynamic_001）を対象に設定を行います。



実装例

設定用パラメータを生成します。

```
var nodeSetting = {

 "DCNodeSetting" : {

  "dynamic_001" : {  // 設定対象のノードIDをプロパティ名とする

   "displayFlag" : false,   // 画面表示をしない

   "processTargetConfigs" : [   // 任意の処理対象者を指定
    {
     // ユーザ：maruyama
     "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
     "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.user",
     "parameter" : "maruyama"
    },
    {
     // ロール：IM-Workflow ユーザ
     "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
     "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.role",
     "parameter" : "im_workflow_user"
    }
   ]
  }
 }
};
```

上記のパラメータをJSON文字列に変換します。スクリプト開発モデルでは、ImJson#toJSONString メソッドを利用します。

```
ImJson.toJSONString(nodeSetting);
```

JSON文字列に変換したパラメータを、「imwNodeSetting」というキーで標準処理画面に受け渡します。
ユーザコンテンツ画面で利用するタグライブラリ「workflowOpenPage」のボディ部にパラメータを定義するなどの対応を行い、標準処理画面が表示される際に生成したパラメータが受け渡されるよう実装してください。

- クライアントタイプ＝PC の場合

```
<imart type="workflowOpenPage" >

 <input type="hidden" name="imwNodeSetting" value="%JSON文字列に変換したパラメータ%" >

</imart>
```

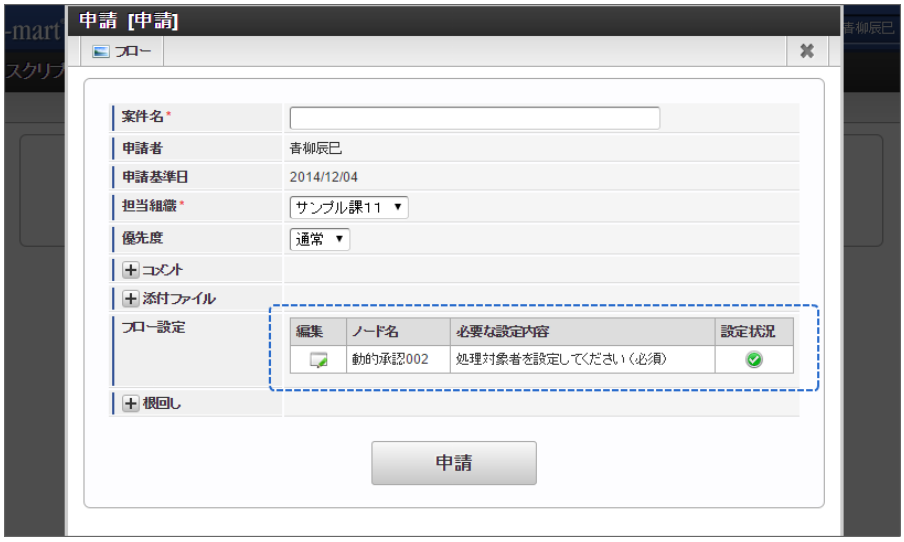
- クライアントタイプ＝スマートフォンの場合

```
<imart type="spWorkflowOpenPage" >

 <input type="hidden" name="imwNodeSetting" value="%JSON文字列に変換したパラメータ%" >

</imart>
```

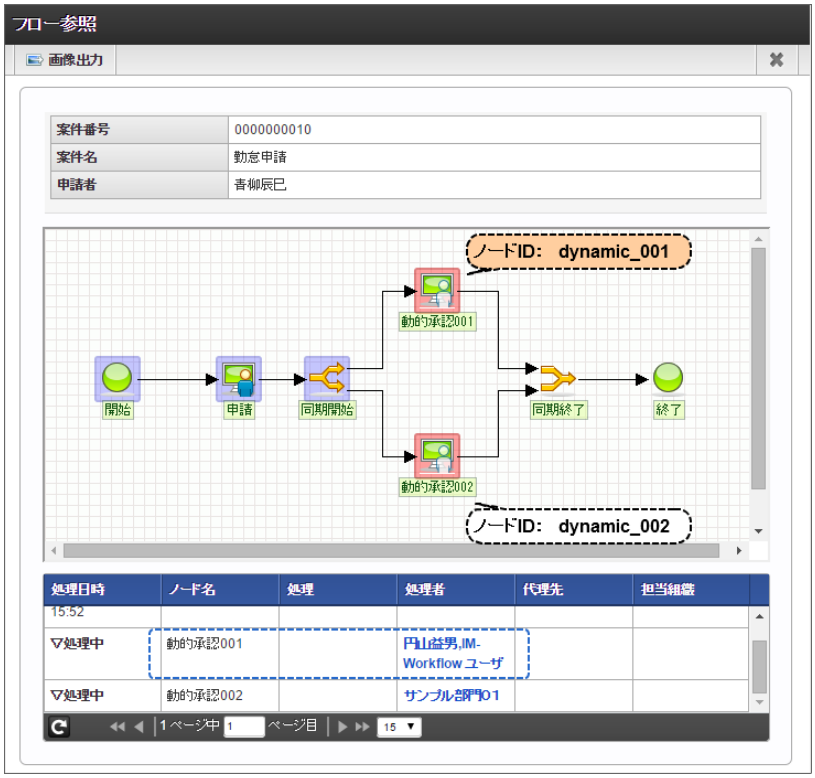
JSON文字列をクライアントに送信したりvalueとして設定する際は、必要に応じて適切なエスケープ処理を行ってください。

動作結果

標準処理画面に上述の「imwNodeSetting」パラメータを受け渡した場合、標準処理画面の「フロー設定」欄には、「動的承認001」ノードが表示されません。
一方、パラメータで未指定の「動的承認002」ノードは画面上に表示されます。



申請を行います。
その結果、「動的承認001」ノードの処理対象者はパラメータで指定したとおりに設定されました。

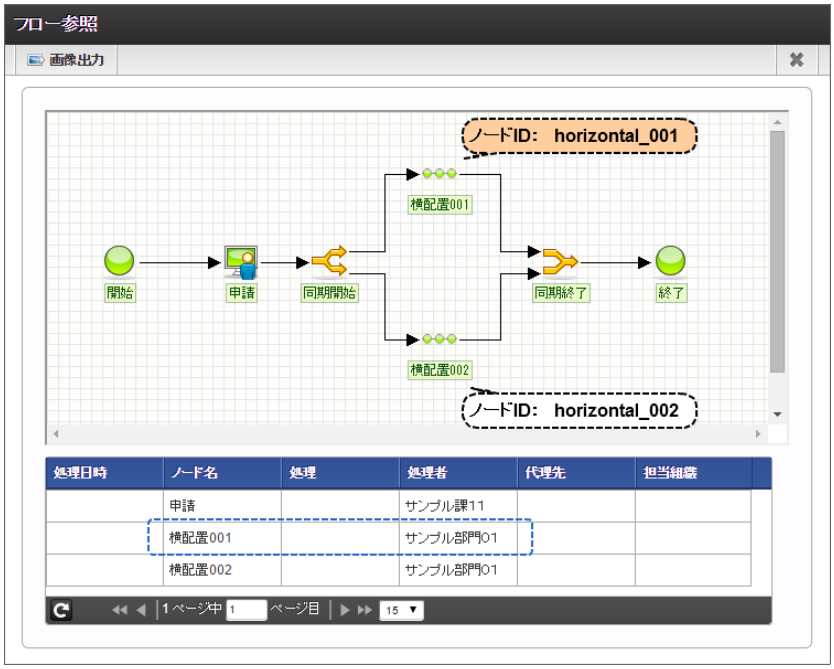


横配置ノードの配置数、処理対象者をシステムで決定する

ユーザコンテンツ側で決定した処理対象者を、横配置ノードに反映します。
また、標準処理画面からは動的承認ノードの設定を行わせない制御を実施します。

前提

申請時に横配置ノードの設定を行います。
利用するフローは次の通りです。
横配置ノードがふたつ存在し、処理対象者は両方とも「サンプル部門０１」に設定されています。
横配置ノードの割当可能ノード数は、ふたつとも　最小＝１、最大＝３　に設定されています。
また、ふたつの横配置ノードの設定を申請ノードで行えるようフロー定義を設定します。
今回は、「横配置001」（ノードID：horizontal_001）を対象に設定を行います。



実装例

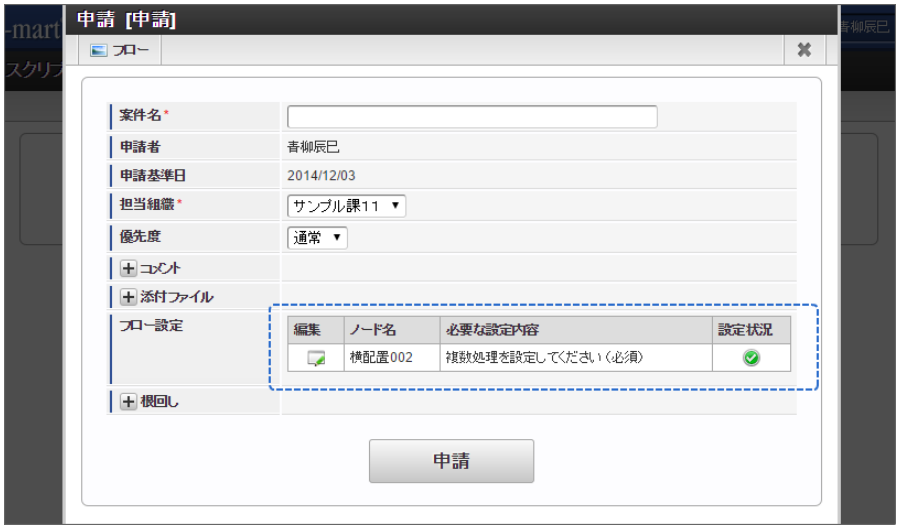
設定用パラメータを生成します。

```
var nodeSetting = {

 "HVNodeSetting" : {

  "horizontal_001" : {  // 設定対象のノードIDをプロパティ名とする

   "displayFlag" : false,   // 画面表示をしない

   "matterNodeExpansions" : [   // ノード展開情報を指定

    // ひとつめの展開ノード
    {
     "nodeName" : "node_name_001",  // ノード名

     "processTargetConfigModel" : [   // 任意の処理対象者を指定
      {
       // 組織：サンプル課１１
       "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
       "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.department",
       "parameter" : "comp_sample_01^comp_sample_01^dept_sample_11"
      },
      {
       // 組織：サンプル課１２
       "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
       "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.department",
       "parameter" : "comp_sample_01^comp_sample_01^dept_sample_12"
      }
     ]
    },
    // ふたつめの展開ノード
    {
     "nodeName" : "node_name_002",  // ノード名

     "processTargetConfigModel" : [   // 任意の処理対象者を指定
      {
       // 組織：サンプル課２１
       "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
       "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.department",
       "parameter" : "comp_sample_01^comp_sample_01^dept_sample_21"
      },
      {
       // 組織：サンプル課２２
       "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
       "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.department",
       "parameter" : "comp_sample_01^comp_sample_01^dept_sample_22"
      }
     ]
    }
   ]
  }
 }
};
```

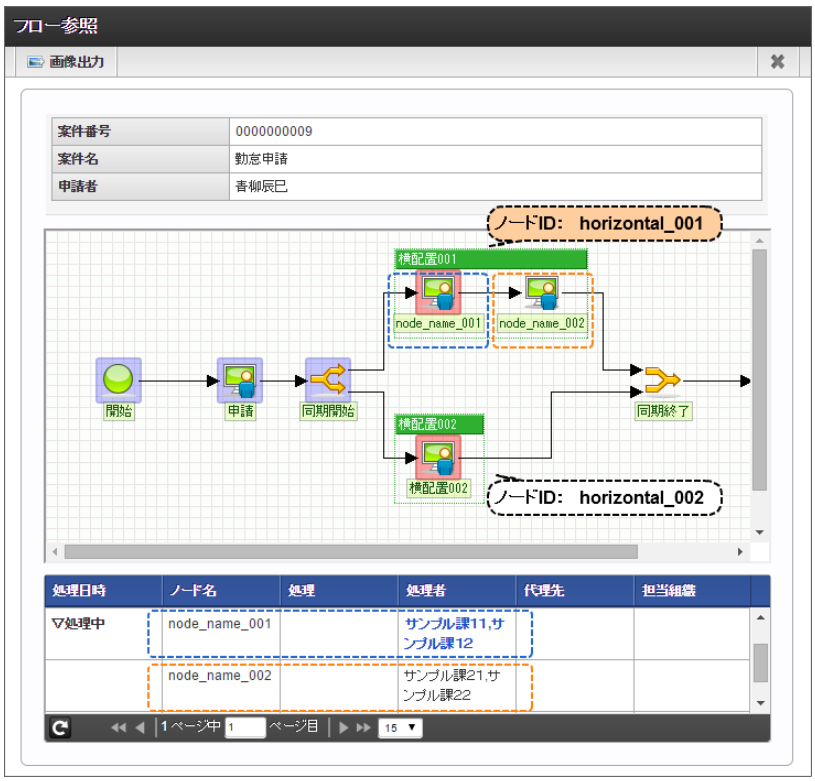
パラメータをJSON文字列に変換し、「imwNodeSetting」として標準処理画面に受け渡します。

動作結果

標準処理画面に上述の「imwNodeSetting」パラメータを受け渡した場合、標準処理画面の「フロー設定」欄には、「横配置001」ノードが表示されません。
一方、パラメータで未指定の「横配置002」ノードは画面上に表示されます。



申請を行います。
その結果、「横配置001」ノードの処理対象者はパラメータで指定したとおりに展開されました。



検索時の暗黙条件

動的承認ノードと横配置ノードに対し、検索時の暗黙条件を設定する場合を例示します。

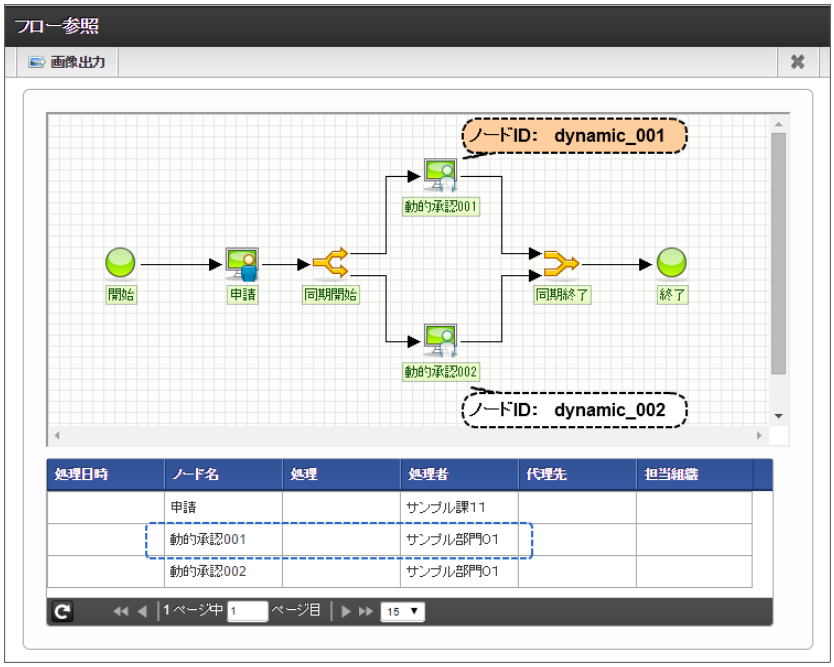
Contents

動的承認ノードに設定可能な処理対象者を制限する

標準処理画面から動的承認ノードの処理対象者を検索する際、ユーザコンテンツ側で決定した暗黙条件を適用します。

前提

申請時に動的承認ノードの設定を行います。
利用するフローは次の通りです。動的承認ノードがふたつ存在し、処理対象者は両方とも「サンプル部門０１」に設定されています。
また、ふたつの動的承認ノードの設定を申請ノードで行えるようフロー定義を設定します。
今回は、「動的承認001」（ノードID：dynamic_001）を対象に設定を行います。



実装例

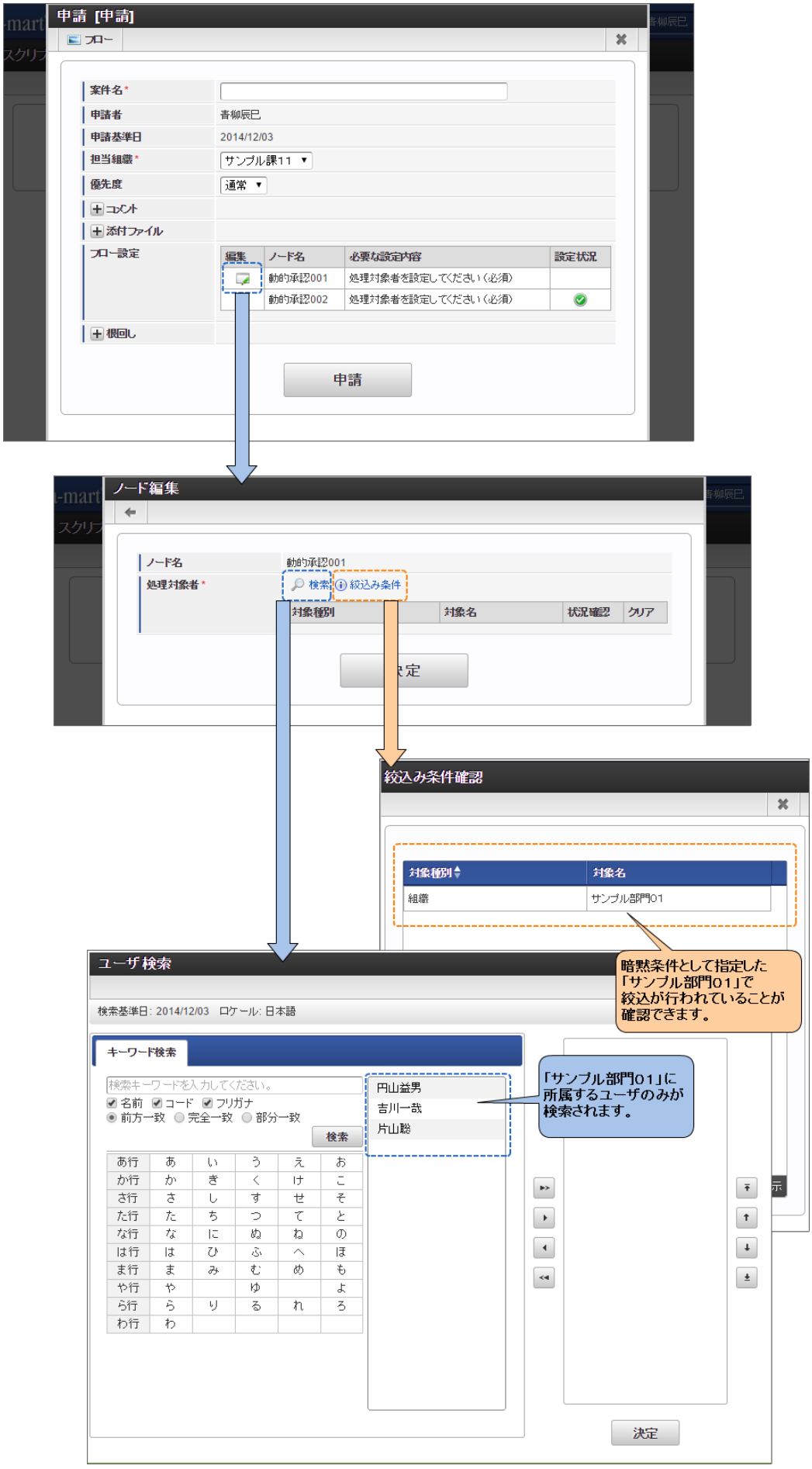
設定用パラメータを生成します。

```
var nodeSetting = {

 "DCNodeSetting" : {

  "dynamic_001" : {  // 設定対象のノードIDをプロパティ名とする

   "displayFlag" : true,  // 画面表示をする

   "searchCondition" : {  // 処理対象者の検索時条件を指定

    "criteria" : {   // 暗黙条件を指定

     "department_set_list" : [
      {
       // 組織：サンプル部門０１
       "company_cd" : "comp_sample_01",
       "department_set_cd" : "comp_sample_01",
       "department" : {
        "department_cd" : "dept_sample_10",
        "compare" : "eq"
       }
      }
     ]
    }
   },

   "processTargetConfigs" : []  // 処理対象者を指定（明示的に0件指定）
  }
 }
};
```

パラメータをJSON文字列に変換し、「imwNodeSetting」として標準処理画面に受け渡します。

動作結果

標準処理画面に上述の「imwNodeSetting」パラメータを受け渡した場合、標準処理画面の「フロー設定」欄には、「動的承認001」「動的承認002」ノードが表示されます。
「動的承認001」のノード編集画面を表示し、「検索」リンクを押下すると、「ユーザ検索（キーワード　タブ）」が表示されます。
この画面で検索を実行すると、パラメータで指定した通り、「サンプル部門０１」に所属するユーザのみが検索されます。



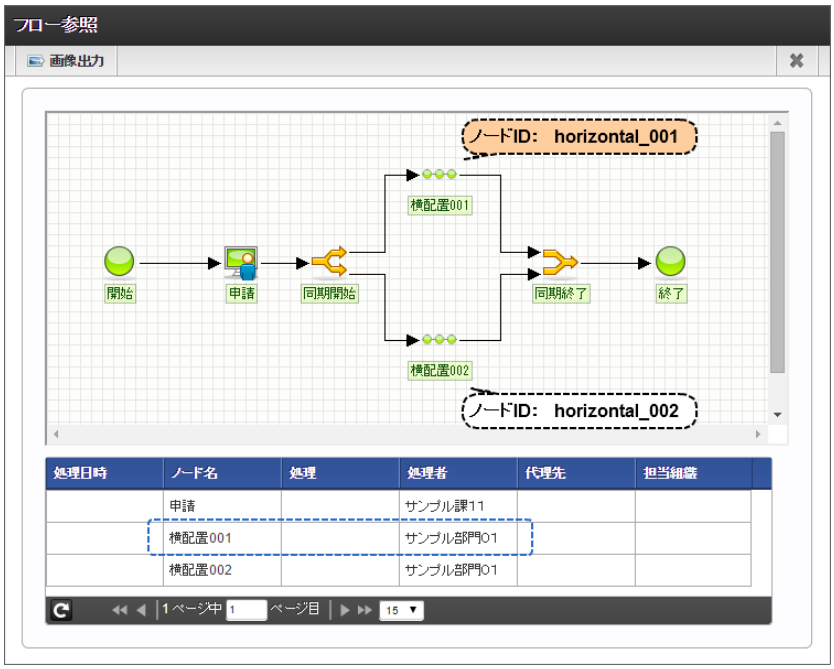
なお、パラメータによる設定を行っていない「動的承認002」の場合、検索結果に対する絞り込みは行われません。

横配置ノードに設定可能な処理対象者を制限する

標準処理画面から横配置ノードの処理対象者を検索する際、ユーザコンテンツ側で決定した暗黙条件を適用します。

前提

申請時に横配置ノードの設定を行います。
利用するフローは次の通りです。
横配置ノードがふたつ存在し、処理対象者は両方とも「サンプル部門０１」に設定されています。
横配置ノードの割当可能ノード数は、ふたつとも　最小＝１、最大＝３　に設定されています。
また、ふたつの横配置ノードの設定を申請ノードで行えるようフロー定義を設定します。
今回は、「横配置001」（ノードID：horizontal_001）を対象に設定を行います。



実装例

設定用パラメータを生成します。

```
var nodeSetting = {

 "HVNodeSetting" : {

  "horizontal_001" : {  // 設定対象のノードIDをプロパティ名とする

   "displayFlag" : true,  // 画面表示をする

   "matterNodeExpansions" : [  // ノード展開情報を指定

    // ひとつめの展開ノード
    {
     "nodeName" : "node_name_001",  // ノード名

     "searchCondition" : {  // 処理対象者の検索時条件を指定

      "criteria" : {  // 暗黙条件を指定

       "department_set_list" : [
        {
         // 組織：サンプル課１１ / 役職：課長
         "company_cd" : "comp_sample_01",
         "department_set_cd" : "comp_sample_01",
         "department" : {
          "department_cd" : "dept_sample_11",
          "compare" : "eq"
         },
         "post" : {
          "post_cd" : "ps003",
          "compare" : "eq"
         }
        }
       ]
      }
     },

     "processTargetConfigModel" : []  // 処理対象者を指定（明示的に0件指定）
    },

    // ふたつめの展開ノード
    {
     "nodeName" : "node_name_002",  // ノード名

     "searchCondition" : {  // 処理対象者の検索時条件を指定

      "criteria" : {  // 暗黙条件を指定

       "department_set_list" : [
        {
         // 組織：サンプル会社 / 役職：社長
         "company_cd" : "comp_sample_01",
         "department_set_cd" : "comp_sample_01",
         "department" : {
          "department_cd" : "comp_sample_01",
          "compare" : "eq"
         },
         "post" : {
          "post_cd" : "ps001",
          "compare" : "eq"
         }
        }
       ]
      }
     },

     "processTargetConfigModel" : []  // 処理対象者を指定（明示的に0件指定）
    }
   ]
  }
 }
```

```
70    }
71  };
```

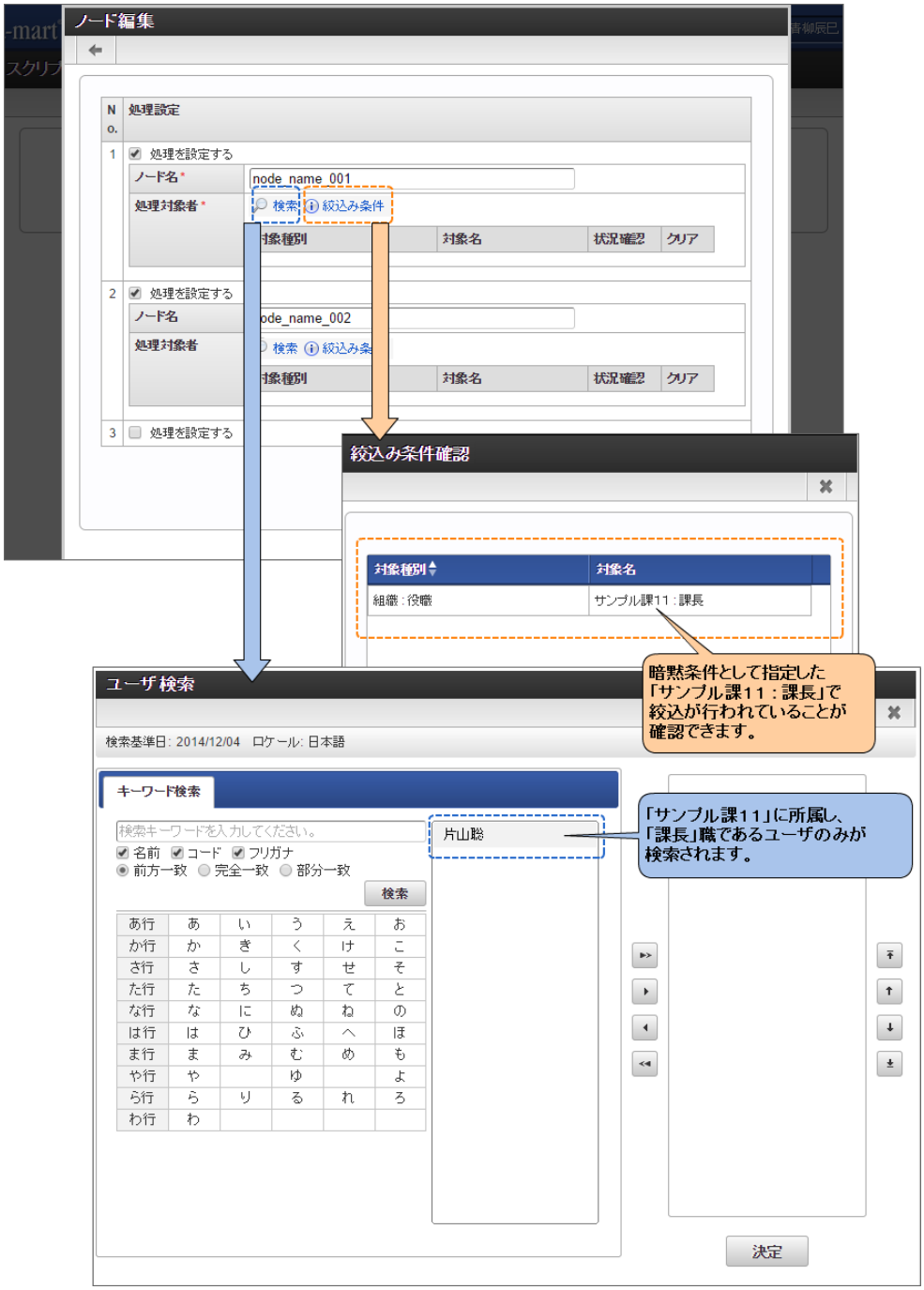
パラメータをJSON文字列に変換し、「imwNodeSetting」として標準処理画面に受け渡します。

動作結果

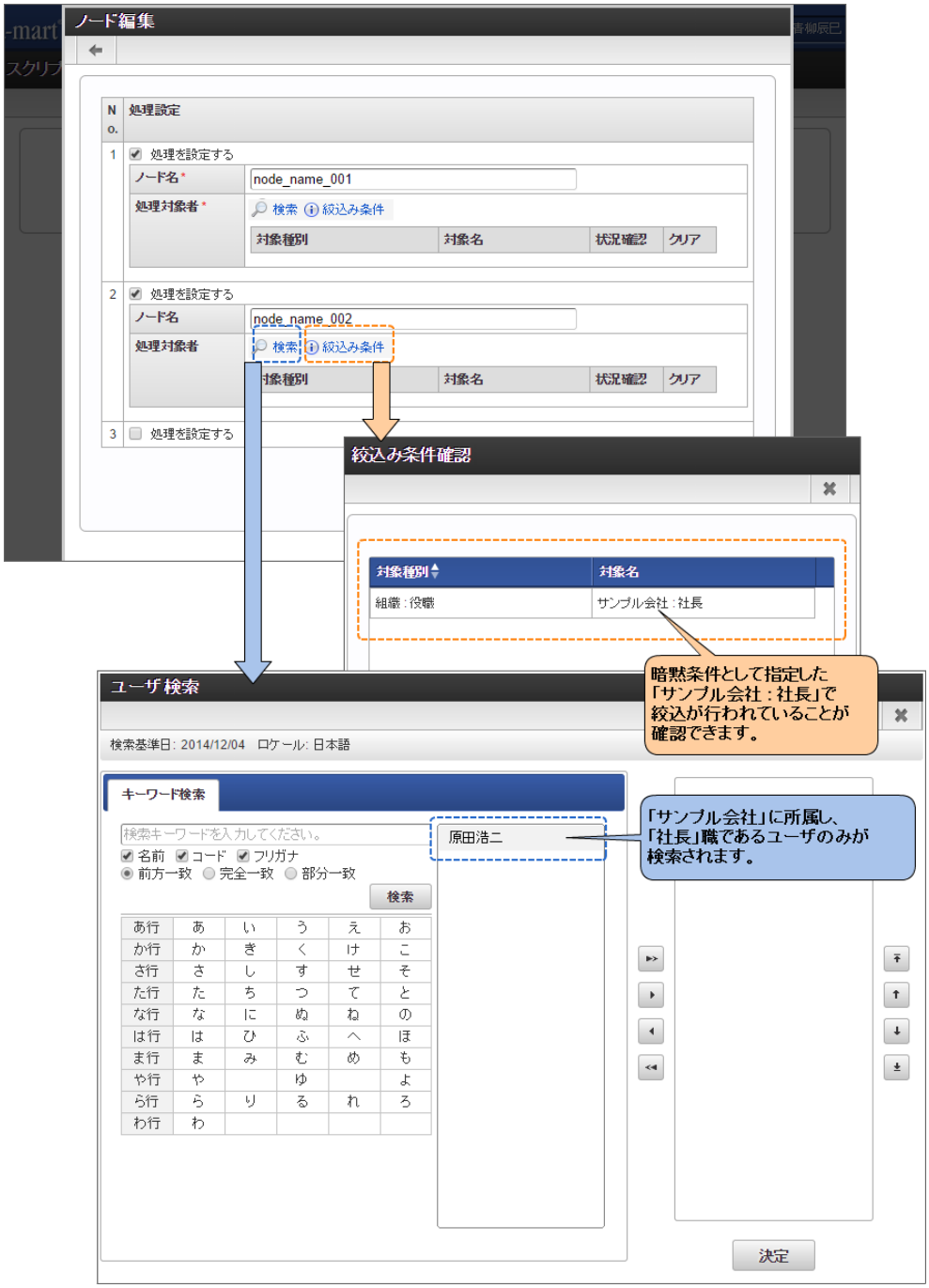
標準処理画面に上述の「imwNodeSetting」パラメータを受け渡した場合、標準処理画面の「フロー設定」欄には、「横配置001」「横配置002」ノードが表示されます。

「横配置001」のノード編集画面を表示します。

まず、ひとつめの展開ノード「node_name_001」に対する「検索」リンクを押下すると、「ユーザ検索（キーワード　タブ）」が表示されます。
この画面で検索を実行すると、パラメータで指定した通り、「サンプル課１１」に所属し、「課長」職であるユーザのみが検索されました。



次に、ふたつめの展開ノード「node_name_002」に対する「検索」リンクを押下すると、「ユーザ検索（キーワード　タブ）」が表示されます。
この画面で検索を実行すると、パラメータで指定した通り、「サンプル会社」に所属し、「社長」職であるユーザのみが検索されました。



なお、パラメータによる設定を行っていない「横配置002」の場合は、検索結果に対する絞り込みは行われません。

パラメータ詳細

機能を利用するうえで指定が必要なパラメータの詳細について記述します。

構造概要

パラメータは、個々のノードに対する設定情報オブジェクトを、ノード種別（動的承認ノード・確認ノード / 横配置ノード・縦配置ノード）で取りまとめた構造です。

個々のノード設定情報は、標準処理画面上で「フロー設定」として表示される各ノードに関連付けられます。

コードで表現すると、以下の通りです。ノードIDは例です。

```
{
  // 動的承認ノード・確認ノード　設定情報
  "DCNodeSetting" : {

    // 設定対象ノードIDをプロパティキーとして指定（ノードIDが”dynamic_001”の場合の例）
    "dynamic_001" : {
      // 設定用の各種パラメータを指定

    },
    // 設定対象ノードIDをプロパティキーとして指定（ノードIDが”confirm_001”の場合の例）
    "confirm_001" : {
      // 設定用の各種パラメータを指定

    }
  },

  // 横配置ノード・縦配置ノード　設定情報
  "HVNodeSetting" : {

    // 設定対象ノードIDをプロパティキーとして指定（ノードIDが”horizontal_001”の場合の例）
    "horizontal_001" : {
      // 設定用の各種パラメータを指定

    },
    // 設定対象ノードIDをプロパティキーとして指定（ノードIDが”vertical_001”の場合の例）
    "vertical_001" : {
      // 設定用の各種パラメータを指定

    }
  }
}
```

個々のノード設定情報の構造はノード種別によって異なります。
以降の章で詳細を解説します。

動的承認ノード・確認ノード

Contents

動的承認ノード、および確認ノードに対する設定を行う場合、「DCNodeSetting」オブジェクトに、設定対象のノード単位で設定情報を定義します。

動的承認ノード、また確認ノードをいずれかひとつ設定する際のパラメータ例を示します。

```
{
  // 動的承認ノード・確認ノード　設定情報
  "DCNodeSetting" : {

    // 設定対象ノードIDをプロパティキーとして指定
    "%ノードID%" : {

      //利用者に標準処理画面上からノード設定を行わせるか否かを制御します。
      "displayFlag" : true,

      // ノード編集画面において「有効」チェックボックスの初期値を制御します。
      "enableFlag" : true,

      // 処理対象者の検索時条件を指定します。
      "searchCondition" : {

        "criteria" : {
          "department_set_list" : [
            {
              "company_cd" : "comp_sample_01",
              "department_set_cd" : "comp_sample_01",
              "department" : {
                "department_cd" : "dept_sample_10",
                "compare" : "ge"
              }
            }
          ]
        }
      },

      //処理対象プラグイン情報を指定します。
      "processTargetConfigs" : [
        {
          "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
          "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.user",
          "parameter" : "maruyama"
        },
        {
          "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
          "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.role",
          "parameter" : "im_workflow_user"
        }
      ]
    }
  }
}
```

以降では、個々のパラメータについて説明します。

displayFlag

利用者に標準処理画面上からノード設定を行わせるか否かを制御します。

設定値・設定する内容	利用者に標準処理画面上からノード設定を行わせるか否かを制御します。 true の場合、標準処理画面の「フロー設定」項目に表示します。 false の場合、標準処理画面の「フロー設定」項目に表示しません。
単位・型	真偽値
省略時の動作	true （表示する）
親オブジェクト	設定対象ノードオブジェクト

enableFlag

ノード編集画面において「有効」チェックボックスの初期値を制御します。

設定値・設定する内容	**この設定は、動的承認ノードに対してのみ有効です。** ノード編集画面において「有効」チェックボックスの初期値を制御します。 この設定を行った場合、「有効」チェックボックスを表示します。 true の場合、有効状態で初期表示します。 false の場合、無効状態で初期表示します。 displayFlagと組み合わせて、両方にfalseを設定した場合、該当の動的承認ノードを削除（ノードスキップ）できます。
単位・型	真偽値
省略時の動作	「有効」チェックボックスの表示有無は、フロー定義における設定（「動的承認ノードの削除」設定）に従います。 「有効」チェックボックスが表示される設定の場合、チェックの有無は現在のフロー状態に応じて決定されます。
親オブジェクト	設定対象ノードオブジェクト

searchCondition

処理対象者の検索時条件を指定します。

設定値・設定する内容	処理対象者の検索時条件を指定します。 「criteria」プロパティを設定することで、検索時の暗黙条件を指定可能です。 **当プロパティを指定した場合、処理対象者を検索する際に利用可能なプラグインと検索タブは「ユーザ検索（キーワード　タブ）」のみです。**
単位・型	オブジェクト（次のプロパティを定義可能） ▪ criteria
省略時の動作	検索条件（暗黙条件）指定なしで動作します。
親オブジェクト	設定対象ノードオブジェクト

criteria

IM-共通マスタのユーザ検索（キーワード　タブ）に対する暗黙条件を指定します。
ユーザ検索（キーワード　タブ）に対する暗黙条件の仕様については「 IM-共通マスタ 検索画面仕様書 」を参照してください。

設定値・設定する内容	ユーザ検索（キーワード　タブ）に対する暗黙条件を指定可能です。 具体的には、「 IM-共通マスタ 検索画面起動引数一覧 」において以下に該当する引数を指定可能です。 ▪ 対象となる検索画面・タブ 機能グループ ＝ 「ユーザ検索画面」 検索画面タブ ＝ 「キーワード」 ▪ 対象となる引数 分類 ＝ 「暗黙条件」
単位・型	オブジェクト 有効な暗黙条件、および暗黙条件の構造については、「 IM-共通マスタ 検索画面起動引数一覧 」を参照してください。
省略時の動作	searchCondition を指定した場合、省略することはできません。 searchCondition ・criteriaの両方が省略された場合、検索条件（暗黙条件）指定なしで動作します。
親オブジェクト	searchCondition

processTargetConfigs

処理対象プラグイン情報を指定します。

設定値・設定する内容	処理対象プラグイン情報を指定します。 配列の要素として処理対象プラグインオブジェクトを複数設定可能です。
単位・型	配列（各要素はオブジェクト（次のプロパティを定義）） ▪ extensionPointId　（拡張ポイントID） ▪ pluginId　（プラグインID） ▪ parameter　（パラメータ） ※指定可能な拡張ポイントIDは次のとおりです。 ▪ 動的承認ノード： jp.co.intra_mart.workflow.plugin.authority.node.dynamic ▪ 確認ノード： jp.co.intra_mart.workflow.plugin.authority.node.confirm ※利用可能なプラグインIDについては「 IM-Workflow 仕様書 」の「 処理権限者プラグイン一覧 」を参照してください。

省略時の動作	現在設定済みの処理対象者が適用されます。
親オブジェクト	設定対象ノードオブジェクト

横配置ノード・縦配置ノード

Contents

横配置ノード、および縦配置ノードに対する設定を行う場合、「HVNodeSetting」オブジェクトに、設定対象のノード単位で設定情報を定義します。

横配置ノード、または縦配置ノードをいずれかひとつ設定する際のパラメータ例を示します。

```
{
  // 横配置ノード・縦配置ノード　設定情報
  "HVNodeSetting" : {

    // 設定対象ノードIDをプロパティキーとして指定
    "%ノードID%" : {

      //利用者に標準処理画面上からノード設定を行わせるか否かを制御します。
      "displayFlag" : true,

      // 処理対象者の検索時条件を指定します。
      "searchCondition" : {
        "criteria" : {
          "department_set_list" : [
            {
              "company_cd" : "comp_sample_01",
              "department_set_cd" : "comp_sample_01",
              "department" : {
                "department_cd" : "dept_sample_10",
                "compare" : "ge"
              }
            }
          ]
        }
      },

      // ノード展開情報を指定します。展開するノード数分オブジェクトを定義します。
      "matterNodeExpansions" : [

        // ひとつめの展開ノード
        {
          // ノード名を指定します。
          "nodeName" : "node_name_001",

          // 処理対象者の検索時条件を指定します。
          "searchCondition" : {
            "criteria" : {
              "department_set_list" : [
                {
                  "company_cd" : "comp_sample_01",
                  "department_set_cd" : "comp_sample_01",
                  "department" : {
                    "department_cd" : "dept_sample_10",
                    "compare" : "ge"
                  }
                }
              ]
            }
          },

          // 処理対象プラグイン情報を指定します。
          "processTargetConfigModel" : [
            {
              "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
              "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.department",
              "parameter" : "comp_sample_01^comp_sample_01^dept_sample_11"
            },
            {
              "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
              "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.department",
              "parameter" : "comp_sample_01^comp_sample_01^dept_sample_12"
            }
          ]
        },

        // ふたつめの展開ノード
        {
          // ノード名を指定します。
          "nodeName" : "node_name_002",
```

```
            // 処理対象者の検索時条件を指定します。
            "searchCondition" : {
              "criteria" : {
                "department_set_list" : [
                  {
                    "company_cd" : "comp_sample_01",
                    "department_set_cd" : "comp_sample_01",
                    "department" : {
                      "department_cd" : "dept_sample_20",
                      "compare" : "eq"
                    }
                  }
                ]
              }
            },

            // 処理対象プラグイン情報を指定します。
            "processTargetConfigModel" : [
              {
                "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
                "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.department",
                "parameter" : "comp_sample_01^comp_sample_01^dept_sample_21"
              },
              {
                "extensionPointId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic",
                "pluginId" : "jp.co.intra_mart.workflow.plugin.authority.node.dynamic.department",
                "parameter" : "comp_sample_01^comp_sample_01^dept_sample_22"
              }
            ]
          }
        ]
      }
    }
}
```

以降では、個々のパラメータについて説明します。

displayFlag

利用者に標準処理画面上からノード設定を行わせるか否かを制御します。

設定値・設定する内容	利用者に標準処理画面上からノード設定を行わせるか否かを制御します。 true の場合、標準処理画面の「フロー設定」項目に表示します。 false の場合、標準処理画面の「フロー設定」項目に表示しません。
単位・型	真偽値
省略時の動作	true（表示する）
親オブジェクト	設定対象ノードオブジェクト

searchCondition

処理対象者の検索時条件を指定します。
当パラメータは、以下の用途でそれぞれ設定が可能です。

- 設定対象の横配置ノード・縦配置ノード単体における全体設定
- ノード展開情報単位の個別設定

上記を同時に指定した場合、個別設定は全体設定より優先して動作します。

設定値・設定する内容	処理対象者の検索時条件を指定します。 「criteria」プロパティを設定することで、検索時の暗黙条件を指定可能です。 **当プロパティを指定した場合、処理対象者を検索する際に利用可能なプラグインと検索タブは「ユーザ検索（キーワード タブ）」のみです。** **この挙動は、全体設定・個別設定を問わず、どちらか一方でも当プロパティを指定した場合に適用されます。**
単位・型	オブジェクト（次のプロパティを定義可能） ▪ criteria
省略時の動作	検索条件（暗黙条件）指定なしで動作します。
親オブジェクト	設定対象ノードオブジェクト、または matterNodeExpansions

criteria

IM-共通マスタのユーザ検索（キーワード　タブ）に対する暗黙条件を指定します。
ユーザ検索（キーワード　タブ）に対する暗黙条件の仕様については「 IM-共通マスタ 検索画面仕様書 」を参照してください。

設定値・設定する内容	ユーザ検索（キーワード　タブ）に対する暗黙条件を指定可能です。 具体的には、「 IM-共通マスタ 検索画面起動引数一覧 」において以下に該当する引数を指定可能です。 ▪ 対象となる検索画面・タブ 機能グループ ＝ 「ユーザ検索画面」 検索画面タブ ＝ 「キーワード」 ▪ 対象となる引数 分類 ＝ 「暗黙条件」
単位・型	オブジェクト 有効な暗黙条件、および暗黙条件の構造については、「 IM-共通マスタ 検索画面起動引数一覧 」を参照してください。
省略時の動作	searchCondition を指定した場合、省略することはできません。 searchCondition ・criteriaの両方が省略された場合、検索条件（暗黙条件）指定なしで動作します。
親オブジェクト	searchCondition

matterNodeExpansions

ノード展開情報を指定します。展開するノード数分オブジェクトを定義します。

設定値・設定する内容	ノード展開情報を指定します。展開するノード数分オブジェクトを定義します。 **displayFlag** が **false（表示しない）** の場合の動作仕様は以下の通りです。 ▪ フロー定義で設定されている「割当可能ノード数」の「最小」「最大」値による制限は行わず、当パラメータで定義したノード数で展開されます。 **displayFlag** が **true（表示する）** の場合の動作仕様は以下の通りです。 ▪ 展開可能なノード数の最小個数は、常にフロー定義で設定されている「割当可能ノード数」の「最小」値です。フロー定義の最小値より当パラメータで指定したノード数の方が小さい場合、処理を行うためには標準画面上での追加設定が必要です。 ▪ 展開可能なノード数の最大個数は、以下の値のうち大きい方が適用されます。 　▪ フロー定義で設定されている「割当可能ノード数」の「最大」値 　▪ 当パラメータで指定したノード数 対象のノードを削除（ノードスキップ）する場合、以下の設定を行ってください。 ▪ フロー定義の「割当可能ノード数（最小）」が0の場合 matterNodeExpansionsの設定を省略、または空で指定してください。 ▪ フロー定義の「割当可能ノード数（最小）」が1以上の場合 displayFlagにFalse、matterNodeExpansionsに空を指定してください。
単位・型	配列（各要素はオブジェクト（次のプロパティを定義可能）） ▪ nodeName ▪ processTargetConfigModel ▪ searchCondition
省略時の動作	**displayFlag** が **false（表示しない）** の場合の動作仕様は以下の通りです。 ▪ 設定対象のノードが未展開の場合、「割当可能ノード数」の「最小」値のノード数で展開されます。その際の処理対象者は、ルート定義で設定された処理対象者です。 ▪ 設定対象のノードがすでに展開されている場合、現在設定済みのノード展開情報と処理対象者が適用されます。 **displayFlag** が **true（表示する）** の場合の動作仕様は以下の通りです。 ▪ フロー定義で設定された「割当可能ノード数」の「最小」値・「最大」値、およびルート定義で設定された処理対象者で動作します。
親オブジェクト	設定対象ノードオブジェクト

nodeName

ノード名を指定します。

設定値・設定する内容	ノード名を指定します。
単位・型	String
省略時の動作	設定対象の横配置ノード・縦配置ノードのノード名が適用されます。
親オブジェクト	matterNodeExpansions

processTargetConfigModel

処理対象プラグイン情報を指定します。

設定値・設定する内容	処理対象プラグイン情報を指定します。 配列の要素として処理対象プラグインオブジェクトを複数設定可能です。
単位・型	配列（各要素はオブジェクト（次のプロパティを定義）） ▪ extensionPointId （拡張ポイントID） ▪ pluginId （プラグインID） ▪ parameter （パラメータ） ※指定可能な拡張ポイントIDは次のとおりです。 jp.co.intra_mart.workflow.plugin.authority.node.dynamic ※利用可能なプラグインIDについては「 IM-Workflow 仕様書 」の「 処理権限者プラグイン一覧 」を参照してください。
省略時の動作	現在設定済みの処理対象者が適用されます。
親オブジェクト	matterNodeExpansions