IM-Workflow
Ver.7.2

プログラミングガイド

2014/04/01 第 6 版

＜＜　変更履歴　＞＞

変更年月日	変更内容
2010/04/28	初版
2010/07/30	第 2 版 ● 「2.2 リクエストパラメータ」に説明を追記しました。 ● 「3.5　制限事項」を追加しました。 ● 「4.2　案件終了処理」に説明を追記しました。 ● 「4.3　アクション処理」に説明を追記しました。 ● 「5.1　未完了案件削除処理リスナー」の説明を修正しました。 ● 「5.2　完了案件削除処理リスナー」の説明を修正しました。 ● 「5.3　過去案件削除処理リスナー」の説明を修正しました。 ● 「5.4　案件退避処理リスナー」の説明を修正しました。 ● 「6.1　テンプレート」に説明を追加しました。 ● 「7.1　呼び出し画面の初期表示値指定」を追加しました。 ● 「7.2　処理対象者プラグインの作成」を追加しました。 ● 「7.3　IM-Workflowクローラリスナーの作成」を追加しました。
2010/10/29	第 3 版 ● 「1.3　準備」の説明を修正しました。 ● 「2.2　リクエストパラメータ」の説明を修正しました。 ● 「3.3　申請（起票案件）／再申請／処理画面の呼び出し」の説明を修正しました。 ● 「3.5　制限事項」の説明を修正しました。 ● 「5.3　過去案件削除処理リスナー」の説明を修正しました。 ● 「7.4　画面入力情報の保持」を追加しました。
2010/11/10	第 4 版 ● 「2.2　リクエストパラメータ」の説明を修正しました。
2011/04/01	第 5 版 ● 「2.2 リクエストパラメータ」の説明を修正しました。 ● 「3.3 申請（起票案件）／再申請／処理画面の呼び出し」の説明を修正しました。 ● 「3.4 確認画面の呼び出し」の説明を修正しました。 ● 「3.5 制限事項」の説明を修正しました。 ● 「4.5 分岐開始処理」の説明を修正しました。 ● 「4.6 分岐終了処理」の説明を修正しました。 ● 「6.1 テンプレート」、「6.2サンプルプログラム」、「7.2.5.3 < extend >に指定するプログラム」、「7.3 IM-Workflowクローラリスナーの作成」のJavaEE開発モデルのサンプルプログラムのパスを修正しました。 ● 「7.1.2 実装例」の説明を修正しました。 ● 「7.5 呼び出し画面からのコールバック関数の指定」を追加しました。
2014/04/01	第 6 版 ● 「2.3 案件処理系APIと画面動作仕様の違い」を追加しました。

＜＜　目次　＞＞

1 はじめに

1.1 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。

1.2 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になります。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。

1.3 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。
製品に付属するインストールガイドを参考に、IM-Workflow が動作する環境を構築します。

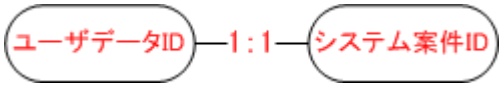
製品のインストール後は、システム管理者でログインし、メニュー[ライセンス]より、初期データインポートを行い、サンプルデータインポートも必ず行ってください。

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(http://www.intra-mart.jp/download/product/index.html)から入手することもできます。

2 概要

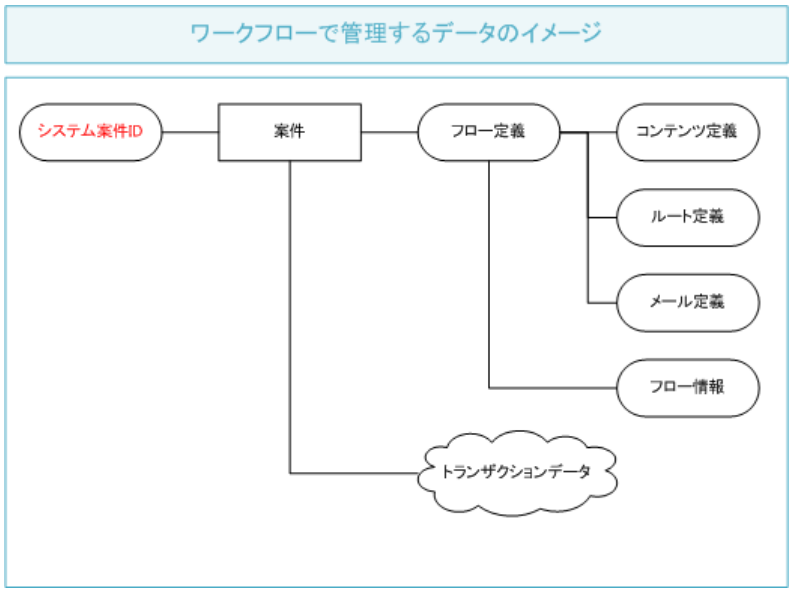
2.1 ユーザアプリケーションデータとIM-Workflowの関係

ユーザアプリケーションデータと IM-Workflow のデータは、”ユーザデータ ID“と“システム案件 ID”をいう 2 つのキーが 1 対 1 の関係となります。



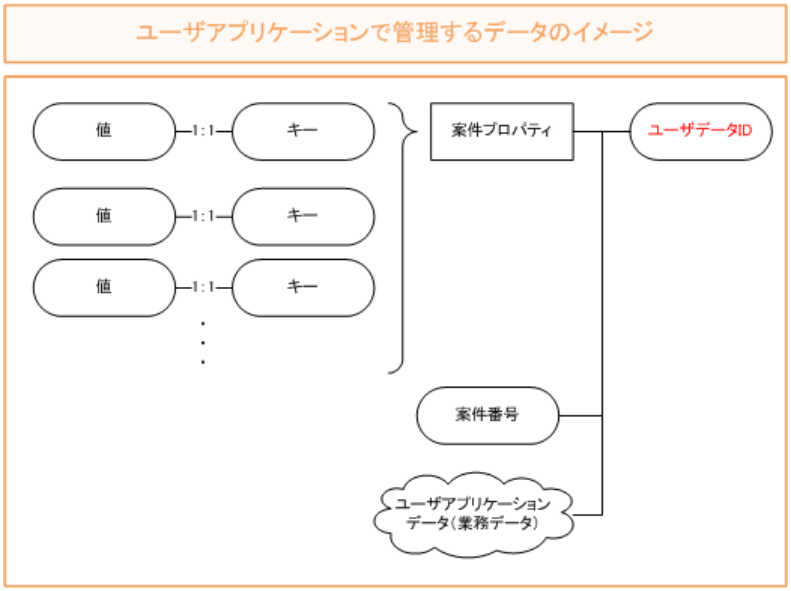
2.1.1 システム案件ID

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。



2.1.2 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。
システム案件 ID 同様に、API やタグライブラリにおいて、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。



2.1.3 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となります。
また、IM-Workflow が提供する各種一覧画面に表示することや分岐条件において、ルール定義の値として使用することができます。

2.2 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	ログイングループ ID	
2	imwUserCode	処理者 CD	
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
4	imwUserDataId	ユーザデータ ID	
5	imwSystemMatterId	システム案件 ID	
6	imwNodeId	ノード ID	
7	imwArriveType	到達種別	
8	imwAuthUserCode	権限者 CD	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 CD 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
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

※imwAuthUserCode（権限者 CD）について、各開発モデルでの取得例を以下に記述します。

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
```

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	○	○	○	○	○	○	-	-	-	-

＜「○」：取得可能 ／「-」：取得不可能＞

2.3 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。

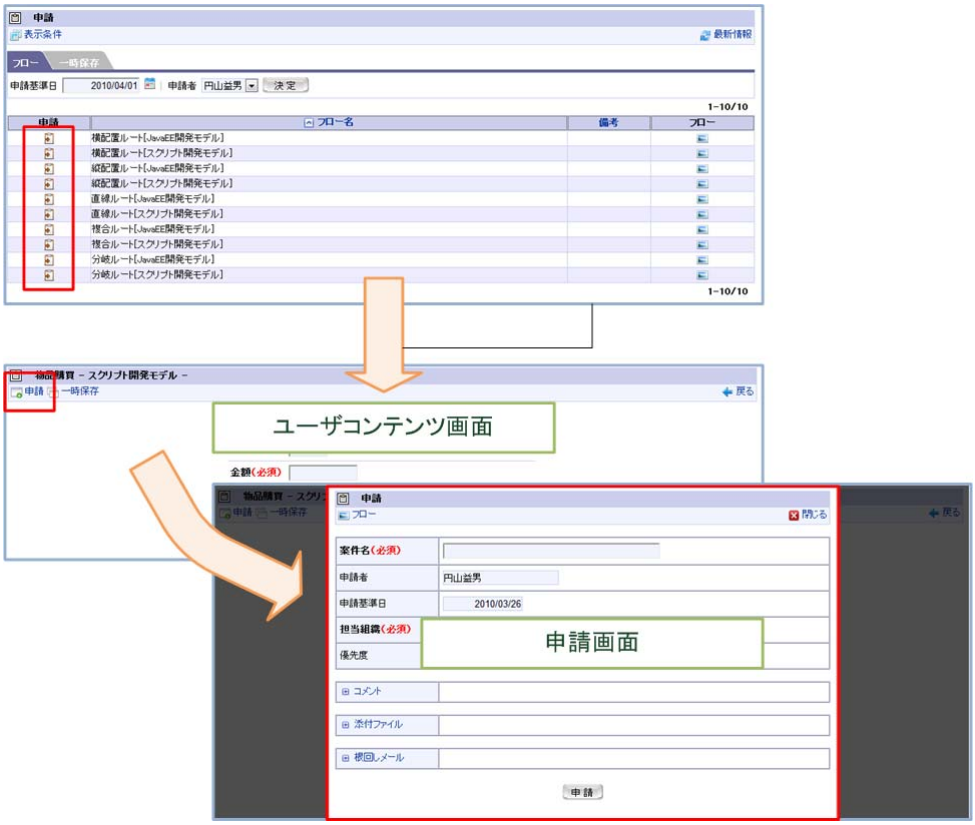
- API 引数として指定した処理権限者が、到達処理で展開された処理対象者に含まれるか
- API 引数として指定した処理権限者と処理実行者が異なる場合、両者間で有効な代理設定が存在するか

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。

3 画面の作成

3.1 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。



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。

3.1.1 スクリプト開発モデル

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

```
<head>

<imart type="workflowOpenPageCsjs" />

</head>
```

申請画面と連携する画面のボディ部(<body> ～ </body>)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。

```
<body>

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

</body>
```

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>
```

3.1.2 JavaEE開発モデル

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

```
<head>

<workflow:workflowOpenPageCsjs />

</head>
```

申請画面と連携する画面のボディ部（<body> ～ </body>）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。

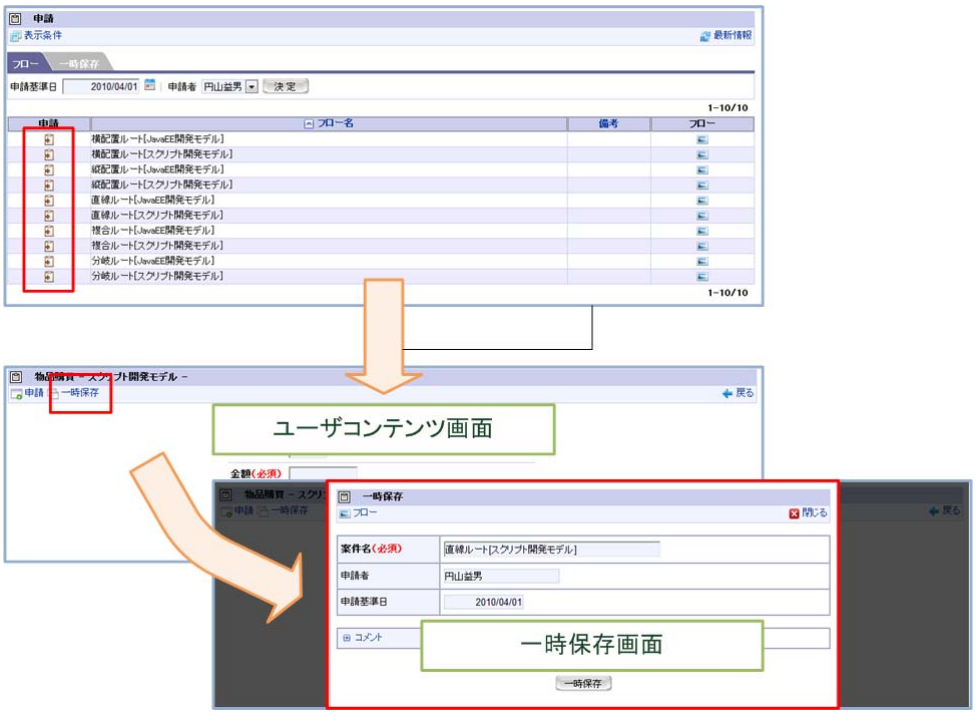
```
<body>

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

</body>
```

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>
```

3.2 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。



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。

3.2.1 スクリプト開発モデル

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

```
<head>

<imart type="workflowOpenPageCsjs" />

</head>
```

一時保存画面と連携する画面のボディ部(<body> ～ </body>)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。

```
<body>

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

</body>
```

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>
```

3.2.2 JavaEE開発モデル

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

```
<head>

<workflow:workflowOpenPageCsjs />

</head>
```

一時保存画面と連携する画面のボディ部（<body> ～ </body>）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。

```
<body>

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

</body>
```

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>
```

3.3 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。

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。

3.3.1 スクリプト開発モデル

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

```
<head>

<imart type="workflowOpenPageCsjs" />

</head>
```

処理画面と連携する画面のボディ部(<body> ～ </body>)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。

```
<body>

<imart type="workflowOpenPage"
 name="approveForm"
          method="POST"
          target="IM_MAIN"
          imwSystemMatterId=oRequest.imwSystemMatterId
          imwNodeId=oRequest.imwNodeId >
</imart>
</body>
```

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>
```

3.3.2 JavaEE開発モデル

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

```
<head>

<workflow:workflowOpenPageCsjs />

</head>
```

処理画面と連携する画面のボディ部(<body> ～ </body>)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。

```
<body>

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

</body>
```

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

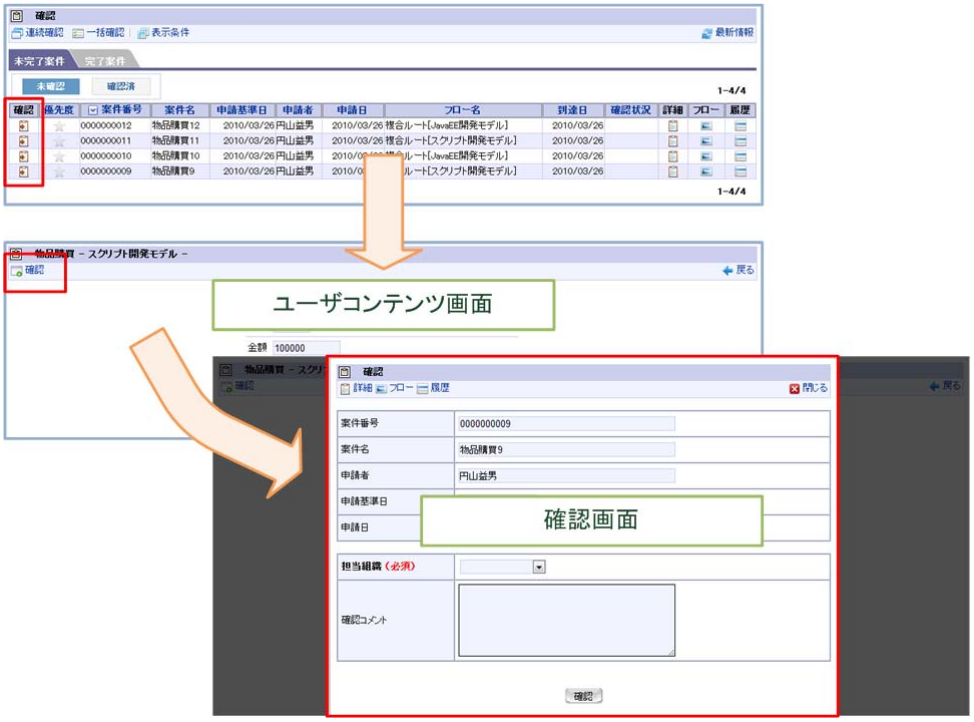
```
<script type="text/javascript">

workflowOpenPage('3');

</script>
```

■ 処理

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

workflowOpenPage('4');

</script>
```

3.4 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。



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。

3.4.1 スクリプト開発モデル

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

```
<head>

<imart type="workflowOpenPageCsjs" />

</head>
```

確認画面と連携する画面のボディ部(<body> ～ </body>)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。

```
<body>

<imart type="workflowOpenPage"
        name="confirmForm"
        method="POST"
        target="IM_MAIN"
        imwSystemMatterId=oRequest.imwSystemMatterId
        imwNodeId=oRequest.imwNodeId>
</imart>

</body>
```

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>
```

3.4.2 JavaEE開発モデル

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

```
<head>

<workflow:workflowOpenPageCsjs />

</head>
```

確認画面と連携する画面のボディ部(<body> ～ </body>)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。

```
<body>

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

</body>
```

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>
```

3.5 制限事項

3.5.1 imwプレフィックスのパラメータについて

「workflowOpenPage」タグでは、ワークフロー処理時の制御用に imw プレフィックスの hidden タグを複数出力します。
パラメータ名が重複すると処理が正常に行われない恐れがあるため、以下で明示的に記載を許可されたもの以外、imw プレフィックス名称のパラメータは記述しないでください。

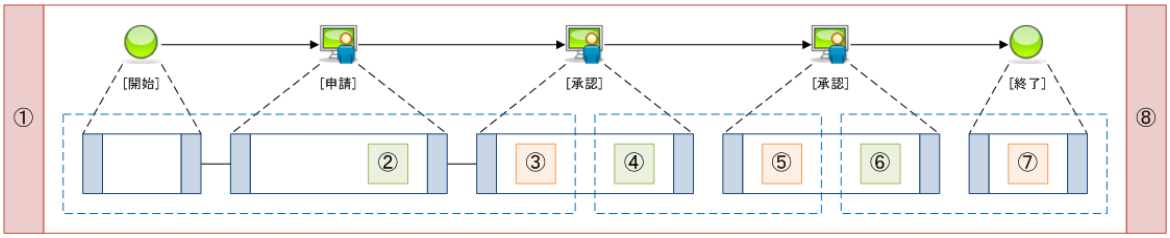
- 7.1 呼び出し画面の初期表示値指定
- 7.4 画面入力情報の保持

3.5.2 タグライブラリの非推奨属性について

「workflowOpenPage」タグでは、セキュリティ対応のため、以下の属性は非推奨扱いとなりました。
属性値を指定した場合も問題なく動作しますが、指定値は無視され、セッションから取得した情報により動作します。

- 「imwGroupId」： ログイングループ ID
- 「imwUserCode」： 処理者 CD （代理先ユーザ CD）

4 ユーザプログラムの作成



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

4.1 案件開始処理

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。

4.2 案件終了処理

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）となります。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（ロールバック）することはできません。

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。

4.3 アクション処理

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
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。

4.4 到達処理

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(ロールバック)することはできません。
(直前のアクション処理とは、トランザクションも別となります。)

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

4.5 分岐開始処理

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下さい。

4.6 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下さい。

5 その他プログラムの作成

5.1 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、ログイングループ単位で処理を行う場合は、下記のファイルに設定します。

%StorageService%/workflow/conf/param/param_group_%ログイングループ ID%.xml

```
<!--
	未完了案件削除リスナの種類
		[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>delete-active-matter-type</param-name>
	<param-value></param-value>
</param>
<!--
	未完了案件削除リスナのパス
	1. 案件削除リスナの種類が java：パッケージ名
	2. 案件削除リスナの種類が script：js 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」を参照下さい。

5.2 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、ログイングループ単位で処理を行う場合は、下記のファイルに設定します。

%StorageService%/workflow/conf/param/param_group_%ログイングループ ID%.xml

```
<!--
    完了案件削除リスナの種類
        [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>delete-complete-matter-listener-type</param-name>
    <param-value></param-value>
</param>
<!--
    完了案件削除リスナのパス
    1. 案件削除リスナの種類が java：パッケージ名
    2. 案件削除リスナの種類が script：%ResourceService%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」を参照下さい。

5.3 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、ログイングループ単位で処理を行う場合は、下記のファイルに設定します。

%StorageService%/workflow/conf/param/param_group_%ログイングループ ID%.xml

```
----------------------------------------------------------------------------------------------------------------------

<!--
	過去案件削除リスナの種類
		[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>delete-archive-matter-listener-type</param-name>
	<param-value></param-value>
</param>
<!--
	過去案件削除リスナのパス
	1. 案件削除リスナの種類が java：パッケージ名
	2. 案件削除リスナの種類が script：%ResourceService%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」を参照下さい。

5.4 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、バッチ処理「IM-Workflow アーカイブバッチ」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、ログイングループ単位で処理を行う場合は、下記のファイルに設定します。

%StorageService%/workflow/conf/param/param_group_%ログイングループ ID%.xml

```
----------------------------------------------------------------------------------------------------------------

<!--
    案件退避リスナの種類
        [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>
<!--
    案件退避リスナのパス
    1. 案件退避リスナの種類が java：パッケージ名
    2. 案件退避リスナの種類が script：%ResourceService%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」を参照下さい。

6 Appendix

6.1 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

%ResourceService%/pages/src/sample/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

■ 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%/src/main/java/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

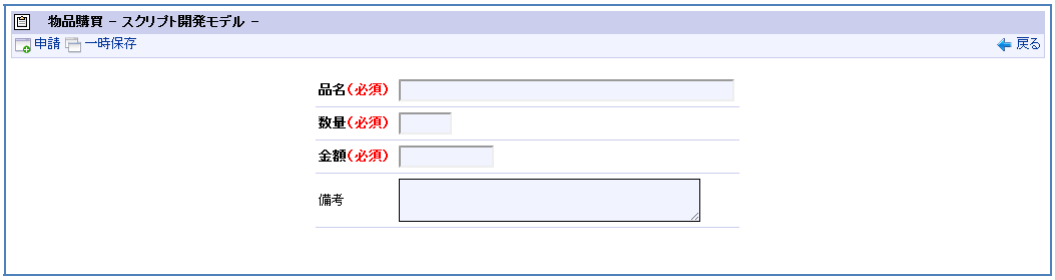
6.2 サンプルプログラム

IM-Workflow のインストール時”サンプルのインストール”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。

6.2.1 画面

6.2.1.1 申請／一時保存／申請（起票案件）／再申請画面



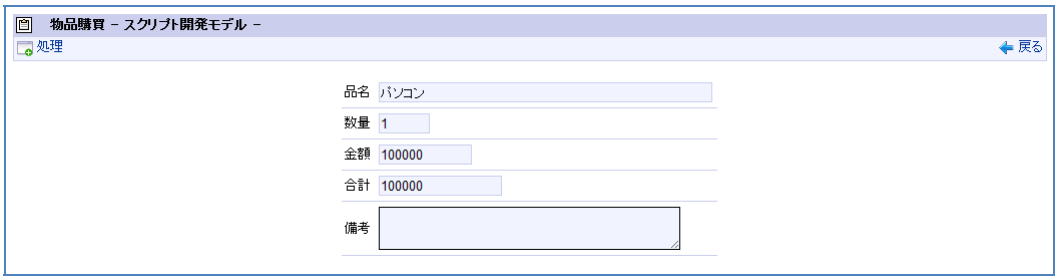
■ スクリプト開発モデル

```
%ResourceService%/pages/src/sample/workflow/purchase/screen/apply.html
%ResourceService%/pages/src/sample/workflow/purchase/screen/apply.js
```

■ JavaEE 開発モデル

```
%ApplicationRuntime%/doc/imart/WEB-INF/classes/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/workflow/purchase/apply.jsp</page-path>
    </next-page>
  </service>
```

6.2.1.2 処理画面



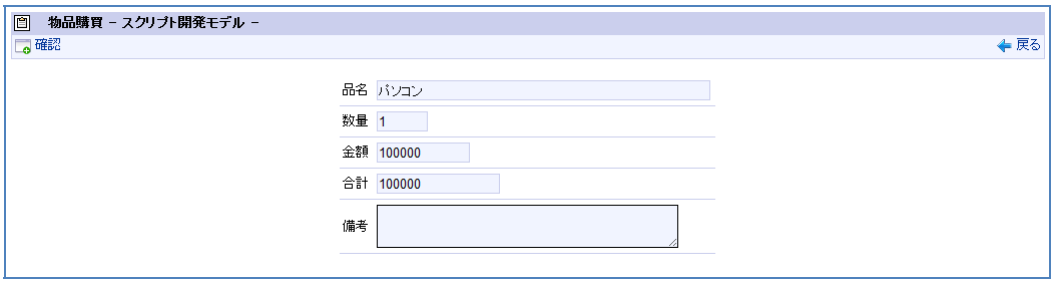
■ スクリプト開発モデル

```
%ResourceService%/pages/src/sample/workflow/purchase/screen/approve.html
%ResourceService%/pages/src/sample/workflow/purchase/screen/approve.js
```

■ JavaEE 開発モデル

```
%ApplicationRuntime%/doc/imart/WEB-INF/classes/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/workflow/purchase/approve.jsp</page-path>
    </next-page>
  </service>
```

6.2.1.3 確認画面



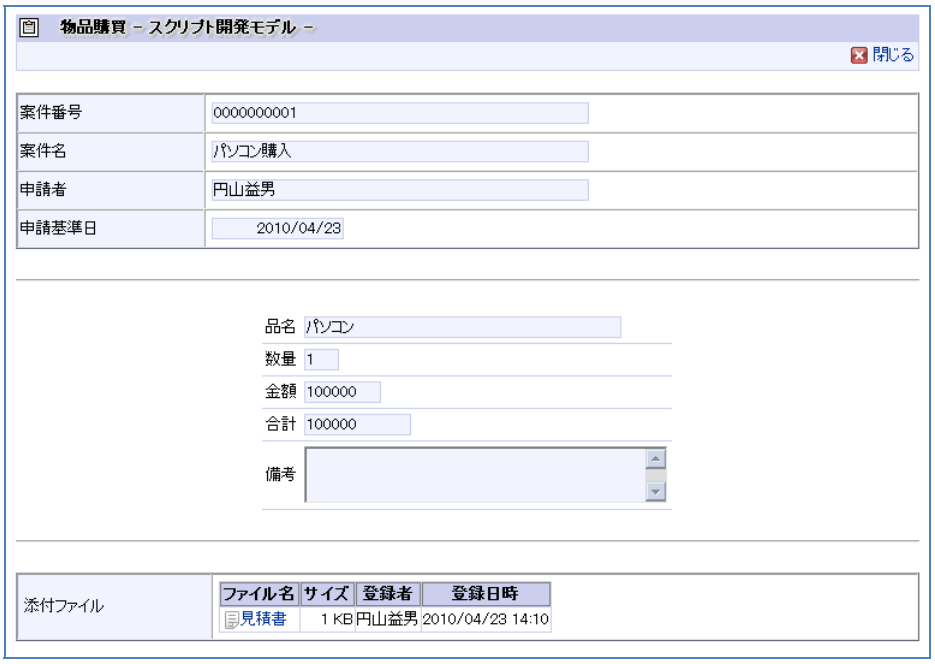
■ スクリプト開発モデル

```
%ResourceService%/pages/src/sample/workflow/purchase/screen/confirm.html
%ResourceService%/pages/src/sample/workflow/purchase/screen/confirm.js
```

■ JavaEE 開発モデル

```
%ApplicationRuntime%/doc/imart/WEB-INF/classes/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/workflow/purchase/confirm.jsp</page-path>
    </next-page>
  </service>
```

6.2.1.4 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

```
%ResourceService%/pages/src/sample/workflow/purchase/screen/detail.html
%ResourceService%/pages/src/sample/workflow/purchase/screen/detail.js
```

■ JavaEE 開発モデル

```
%ApplicationRuntime%/doc/imart/WEB-INF/classes/service-config-imw_sample_purchase.xml
----------------------------------------------------------------------------------------------------------------------

  <service>
    <service-id>detail</service-id>
    <controller-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.
                                                        DetailServiceController</controller-class>
    <transition-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.
                                                        DetailServiceTransition</transition-class>
    <next-page>
      <page-path>/sample/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。

■ スクリプト開発モデル

```
16 | <br>
17 | <imart type="workflowMatterData" systemMatterId=oRequest.imwSystemMatterId
18 |                                          displayItem="matter_number,matter_name,apply_user,apply_base_date" />
19 | <br>
```

■ JavaEE 開発モデル

```
22 | <br>
23 | <workflow:workflowMatterData systemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
24 |                                  displayItem="matter_number,matter_name,apply_user,apply_base_date" />
25 | <br>
```

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。

■ スクリプト開発モデル

```
76 | <br>
77 | <imart type="workflowMatterFile" systemMatterId=oRequest.imwSystemMatterId />
78 | </body>
```

■ JavaEE 開発モデル

```
77 | <br>
78 | <workflow:workflowMatterFile systemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>' />
79 | </body>
```

◆ 案件に添付ファイルがない場合は、表示されません。

6.2.2 ユーザプログラム

6.2.2.1 アクション処理プログラム

■ スクリプト開発モデル

```
%ResourceService%/pages/src/sample/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%/src/main/java/
        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

```
%ResourceService%/pages/src/sample/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%/src/main/java/
        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されています。

[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。

6.2.2.2 案件終了処理プログラム

■ スクリプト開発モデル

```
%ResourceService%/pages/src/sample/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%/src/main/java/
        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されています。

[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。

6.2.2.3 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

```
%ResourceService%/pages/src/sample/workflow/purchase/action/RuleCondition1.js
%ResourceService%/pages/src/sample/workflow/purchase/action/RuleCondition2.js
%ResourceService%/pages/src/sample/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%/src/main/java/
        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%/src/main/java/
        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%/src/main/java/
        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に成功（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に成功（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に成功（true）を返却します。

6.2.3 リスナー

6.2.3.1 未完了案件削除処理リスナー

■ スクリプト開発モデル

```
%ResourceService%/pages/src/sample/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%/src/main/java/
        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。

6.2.3.2 完了案件削除処理リスナー

■ スクリプト開発モデル

```
%ResourceService%/pages/src/sample/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%/src/main/java/
        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。

6.2.3.3 過去案件削除処理リスナー

■ スクリプト開発モデル

%ResourceService%/pages/src/sample/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%/src/main/java/
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。

6.2.3.4 案件退避処理リスナー

■ スクリプト開発モデル

%ResourceService%/pages/src/sample/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%/src/main/java/
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。

7 カスタマイズ

7.1 呼び出し画面の初期表示値指定

IM-Workflow で提供する各処理（申請／再申請／申請（未申請）／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。

7.1.1 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。

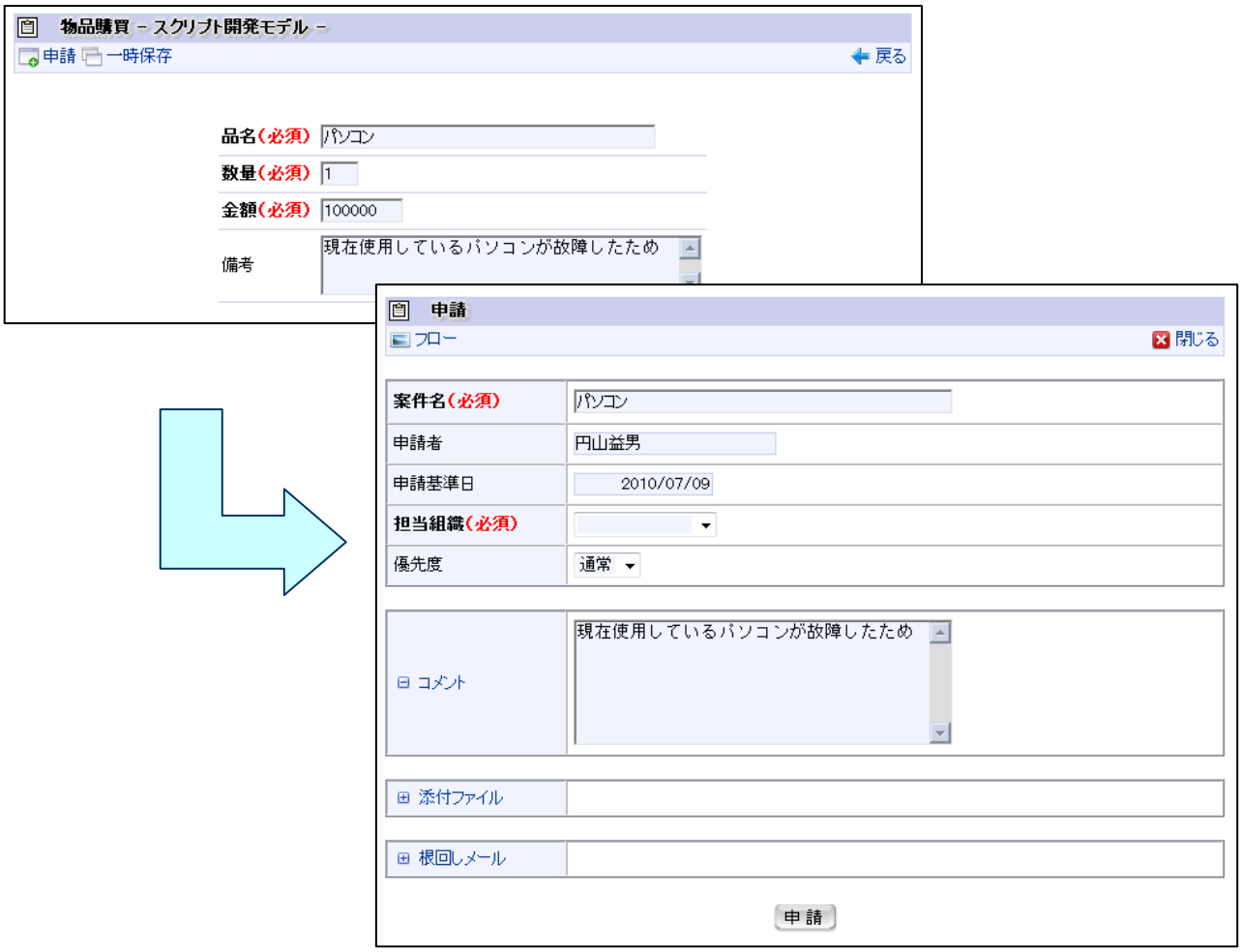
No	パラメータ（物理名）	パラメータ（論理名）	呼び出し画面側の対応項目	動作対象呼び出し画面
1	imwMatterName	案件名	案件名	申請／一時保存／申請（起票案件）／再申請
2	imwComment	コメント	コメント	すべて
3	imwForcedParamFlag	強制パラメータフラグ	※動作制御用フラグ	-

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	-

7.1.2 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。



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となります。

■ スクリプト開発モデル

%ResourceService%/pages/src/sample/workflow/purchase/screen/apply_display.html

■ JavaEE 開発モデル

%ApplicationRuntime%/doc/imart/sample/workflow/purchase/apply_display.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

%ResourceService%/pages/src/sample/workflow/purchase/screen/apply.html

■ JavaEE 開発モデル

%ApplicationRuntime%/doc/imart/sample/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。

```
<html>
<head>
<imart type="imDesignCss" />
<imart type="workflowOpenPageCsjs" />
<script type="text/javascript">
function onClickOpenPage(pageType) {
    if (pageType != "1") {
        if(!inputCheck()) {
            return;
        }
    }

    document. workflowOpenPageForm.imwMatterName.value = document. workflowOpenPageForm.item_name.value;
    document. workflowOpenPageForm.imwComment.value    = document. workflowOpenPageForm.item_comment.value;
    document. workflowOpenPageForm.imwForcedParamFlag.value = "1";

    workflowOpenPage(pageType);
}

.
.
.

</table>

<input type="hidden" name="imwMatterName" value="">
<input type="hidden" name="imwComment" value="">
<input type="hidden" name="imwForcedParamFlag" value="">

</imart>

<imart type="form" name="backForm" method="POST" page=oRequest.imwCallOriginalPagePath>
    <imart type="hidden" imwCallOriginalParams=oRequest.imwCallOriginalParams />
</imart>
</body>
</html>
```

7.2 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るようになっています。
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 WebPlatform / AppFramework の API である”PluginManager”によって管理されます。
そのため、PluginManager について理解しておく必要があります。PluginManager の詳細については「intra-mart WebPlaform / AppFramework デベロッパーズガイド」を参照ください。

7.2.1 対象ノード

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

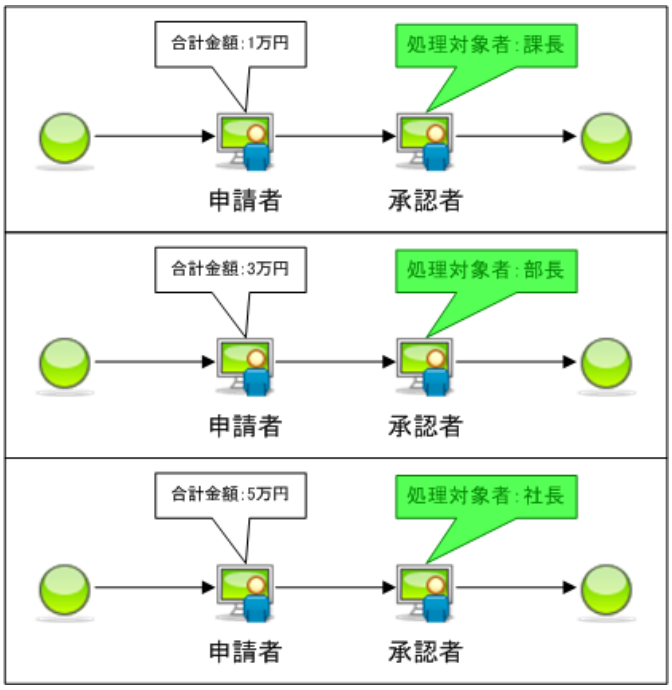
7.2.2 サンプルの説明

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。



7.2.3 サンプルの実行準備

ここでは、承認ノードの対して、「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。

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

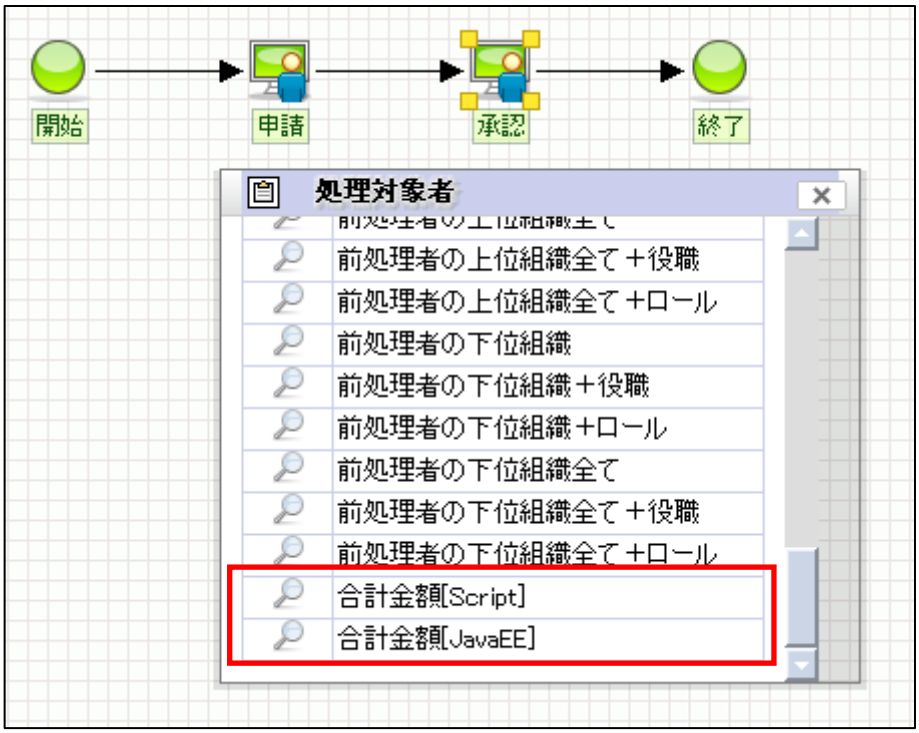
```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
    <extension point="jp.co.intra_mart.workflow.plugin.authority.node.approve" >

        <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/workflow/purchase/plugin/authority/item_total/itemTotalConfig">
                    <parameter key="pluginName" value="SAMPLE.IMW.CAP.030" />
                </script>
            </configPage>
            <extend>
                <script file="sample/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>
```

スクリプト開発モデル

JavaEE 開発モデル

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。



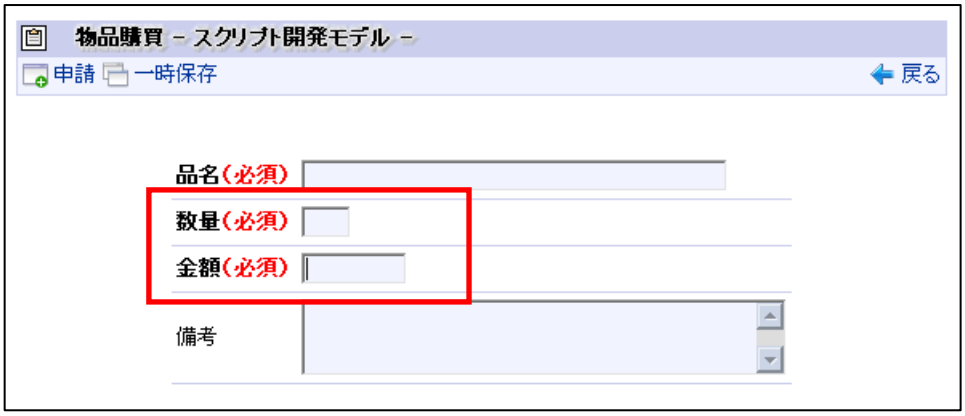
「合計金額[Script]」および「合計金額[JavaEE]」は、実装方法（開発言語）の違いによるもので、処理内容に関しては違いがありません。

「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。

7.2.4 サンプルの実行

「7.2.3 サンプルの実行準備」で作成したフロー定義で申請を行ないます。



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。

7.2.5 処理対象者プラグインについて

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。

7.2.5.1 「plugin.xml」

「plugin.xml」は、”PluginManager”によって管理されるファイルです。
詳細は、”PluginManager”のドキュメントを参照ください。

```
<?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/workflow/purchase/plugin/authority/item_total/itemTotalConfig">
                    <parameter key="pluginName" value="SAMPLE.IMW.CAP.030" />
                </script>
            </configPage>
            <extend>
                <script file="sample/workflow/purchase/plugin/authority/i
                                                        tem_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>
```

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>	<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、この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、このプログラムを作成する場合は、applicationId および serviceId を指定します。
	<javaee applicationId serviceId>	
< extend >	<script file>	< extend > に指定するプラグラムは、処理対象者を決定するプログラムとなります。 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、この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、この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。
	<java class>	

サンプルは、下記のようになります。

- 承認ノード

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

- 承認ノード

```
%ServerManager%/ plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.static/
                                                                                    plugin.xml
```

- 動的承認ノード

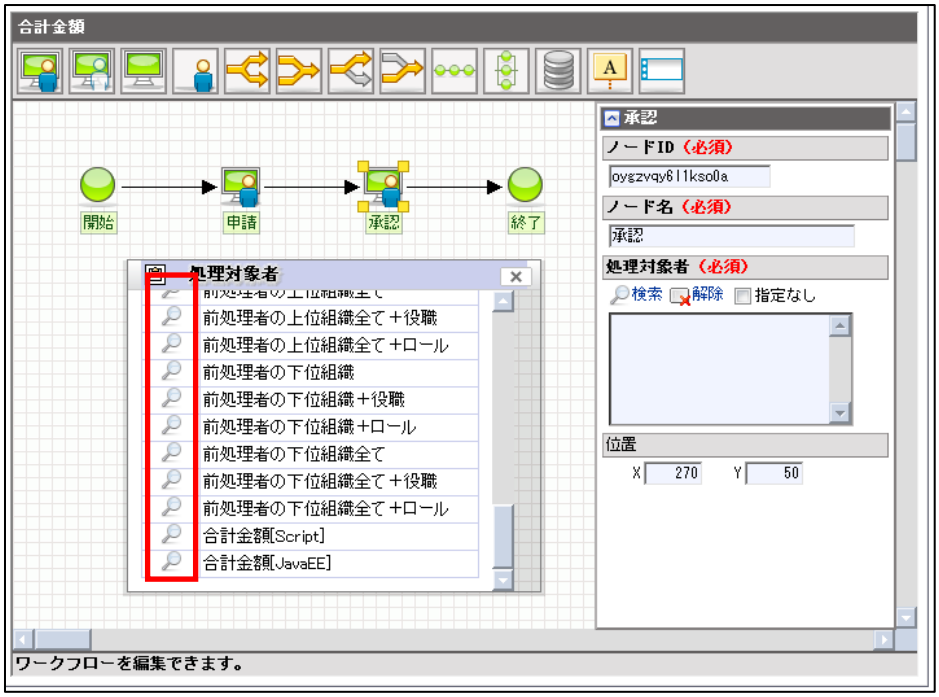
```
%ServerManager%/ plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.dynamic/
                                                                                    plugin.xml
```

- 確認ノード

```
%ServerManager%/ plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.confirm/
                                                                                    plugin.xml
```

7.2.5.2 <configPage>に指定するプログラム

[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。



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は、下記のようになります。

■ スクリプト開発モデル

```
%ResourceService%/ pages/src/sample/workflow/purchase/plugin/authority/item_total/itemTotalConfig.html
%ResourceService%/ pages/src/sample/workflow/purchase/plugin/authority/item_total/itemTotalConfig.js
```

■ JavaEE 開発モデル

```
%ApplicationRuntime%/doc/imart/WEB-INF/classes/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.ItemTotalConfigServiceController</controller-class>
    <transition-class>jp.co.intra_mart.sample.workflow.purchase.plugin.authority.
                                item_total.controller.service.ItemTotalConfigTransition</transition-class>
    <next-page>
       <page-path>/sample/workflow/purchase/plugin/authority/item_total/itemTotalConfig.jsp</page-path>
    </next-page>
  </service>
```

7.2.5.3 < extend >に指定するプログラム

処理対象者を決定する際に実行されるプログラムとなります。
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	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。
getTargetUserList	処理対象ユーザの一覧を取得するメソッド [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	処理対象者プラグインの名称を取得するメソッド 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は、下記のようになります。

■ スクリプト開発モデル

%ResourceService%/ pages/src/sample/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%/src/main/java/jp/co/intra_mart/sample/
workflow/purchase/plugin/authority/item_tota/WorkflowAuthorityExecEventListener.java

7.3 IM-Workflowクローラリスナーの作成

IM-Workflow クローラ実行時に、アプリケーションデータを同時にインデックスします。

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、アプリケーションデータである「品名」を追加登録するサンプルを示します。

■ IM-Workflow クローラ登録文書追加リスナー

%サンプルプログラムディレクトリ%/src/main/java/
jp/co/intra_mart/sample/workflow/purchase/listener/WorkflowCrawlingAddListener.java

```
/**
 * 登録文書に "物品購買"のアプリケーションデータ 「品名」 を追加します。
 *
 * @param   inputDoc   入力文書
 * @param   loginGroupId   ログイングループ ID
 * @param   localeId   ロケール ID
 * @param   systemMatterId   システム案件 ID
 * @param   userDataId   ユーザデータ ID
 * @throws   SolrCrawlerException   リスナー処理で例外が発生した場合
 */
public void addDocumentInfo(IntramartSolrInputDocument inputDoc, String loginGroupId,
        String localeId, String systemMatterId, String userDataId) throws SolrCrawlerException {

    try {
        // ①引数のパラメータ情報から、該当の購買情報を取得する
        SessionInfo sessionInfo = AccessSecurityManager.getInstance().getSessionInfo();
        ConfigurationUserInfo userInfo = new ConfigurationUserInfo();
        userInfo.setUserID(sessionInfo.getUser());
        userInfo.setLoginGroupID(loginGroupId);

        EventManager eventManager = EventManager.getEventManager();
        SelectEvent selectEvent = (SelectEvent)eventManager.
                createEvent("imw_sample_purchase", "select", userInfo);
        selectEvent.setUserDataId(userDataId);

        SelectEventResult eventResult = (SelectEventResult)eventManager.dispatch(selectEvent, true);

        // ②この案件情報が追加対象となるかの判定を行う。
        if (eventResult.getModelObject() == null) {
            return;
        }

        // ③引数の文書登録用オブジェクトに、アプリケーションデータを設定する。(NULL 値の設定不可)
        // 1. テキストフィールドに品名を追加
        inputDoc.addText(eventResult.getModelObject().getItemName());

        // 2. ダイナミックフィールドに品名を追加
        inputDoc.addField("item_name_string", eventResult.getModelObject().getItemName());

        // 文書種別の追加(親文書種別 imw > purchase とする)
        inputDoc.addType("purchase");

    } catch (Exception e) {
        throw new SolrCrawlerException(e.getMessage(), e);
    }
}
```

■ 業務テンプレート

%ApplicationRuntime%/doc/imart/ sample/workflow/purchase/solr/purchase_template.jsp

サンプルプログラムを動作させるには、下記のファイルに、

%Server Manager%/conf/ system-install.xml

以下の設定を記述します。

```
<listener>
  ・・・・・・・・・・・
  <imwcrawler-add-listener>
      <listener-class>jp.co.intra_mart.sample.workflow.purchase.listener.WorkflowCrawlingAddListener</listener-class>
  </imwcrawler-add-listener>
</listener>
```

下記のファイルに、

%Server Manager%/conf/ solr-display-config_imw.xml

以下の設定を記述します。

```
<document>
  ・・・・・・・・・・・
    <document-type id = "purchase">
        <parent-document-type>imw</parent-document-type>
        <display-string-key>SAMPLE.IMW.CAP.012</display-string-key>
        <templete-url>sample/workflow/purchase/solr/purchase_template.jsp</templete-url>
        <require-fields>
            <field name="matter_number_string"/>
            <field name="flow_name_string"/>
            <field name="item_name_string"/>
        </require-fields>
    </document-type>
</document>
```

7.4 画面入力情報の保持

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。

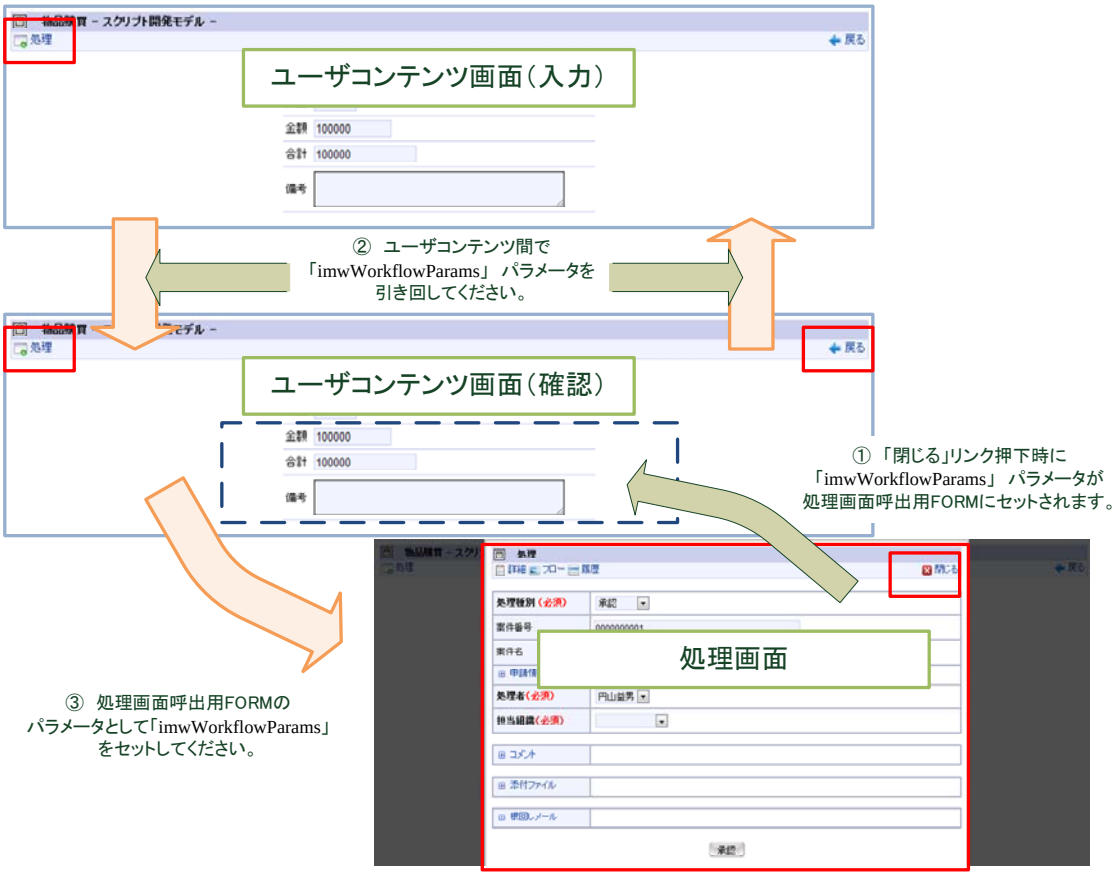
当機能の仕様概要は以下の通りです。

- 各処理画面の「閉じる」リンク押下時に、呼出元ユーザコンテンツ中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、そこに入力情報を格納
- 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 Body 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 ＞



7.5 呼び出し画面からのコールバック関数の指定

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」リンクによって各画面を閉じる際のコールバック関数として、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。

7.5.1 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。

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となります。

■ スクリプト開発モデル

```
%ResourceService%/pages/src/sample/workflow/purchase/screen/apply_callback.html
```

■ JavaEE 開発モデル

```
%ApplicationRuntime%/doc/imart/sample/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

```
%ResourceService%/pages/src/sample/workflow/purchase/screen/apply.html
```

■ JavaEE 開発モデル

```
%ApplicationRuntime%/doc/imart/sample/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。

```
<html>
<head>
<imart type="imDesignCss" />
<imart type="workflowOpenPageCsjs" />
<script type="text/javascript">
function onClickOpenPage(pageType) {
	if (pageType != "1") {
		if(!inputCheck()) {
			return;
		}
	}

	workflowOpenPage(pageType, callbackFnc);
}

function callbackFnc() {
	alert("Callback function is executed.");
}

.
.
.

<imart type="form" name="backForm" method="POST" page=oRequest.imwCallOriginalPagePath>
	<imart type="hidden" imwCallOriginalParams=oRequest.imwCallOriginalParams />
</imart>
</body>
</html>
```

IM-Workflow Ver.7.2
プログラミングガイド

2014/04/01 第6版

TEL: 03-5549-2821
FAX: 03-5549-2816
E-MAIL: info@intra-mart.jp
URL: http://www.intra-mart.jp/