intra-mart WebPlatform/AppFramework Ver.7.0

FormatCreator　プログラミングガイド

2008/07/07 初版

<< 変更履歴 >>

変更年月日	変更内容
2008/07/07	初版

<< 目次 >>

1 はじめに 1
1.1 本書の目的 1
1.2 前提条件 1
1.3 準備 1
2 クラス指定 2
2.1 サンプル実行の準備 2
2.2 サンプルの実行 3
2.3 サンプルプログラムの解説 4
2.3.1 申請書画面から引き渡されるパラメータについて 4
2.3.2 申請書画面への引き渡すパラメータについて 5
3 審議タスクにおいて追記する機能 6
3.1 サンプルの解説 6
3.2 サンプル実行の準備 7
3.3 サンプルの実行 9
3.4 サンプルプログラムの解説 10
3.4.1 “申請者”タスクの後処理プログラム 10
3.4.2 “第 1 承認者”タスクの後処理プログラム 12
4 分岐プログラムの作成 13
4.1 サンプルの解説 13
4.2 サンプル実行の準備 15
4.3 サンプルの実行 17
4.4 サンプルプログラムの解説 19
5 FormatCreator で提供するAPI 20
5.1 サンプル実行の準備 20
5.2 サンプルプログラムの解説 22
5.2.1 getApplicationBasicDataメソッド 22
5.2.2 getApplicationDataメソッド 23
5.2.3 getApplicationDataListメソッド 23
5.2.4 getApplicationDataItemメソッド 24
5.2.5 getPDFDataメソッド 24
6 申請書をサーバ上に保存する 26
6.1 サンプル実行の準備 26
6.2 サンプルの実行 28
6.3 サンプルプログラムの解説 30

1 はじめに

1.1 本書の目的

本書は、FormatCreator のサンプルプログラムを利用して、FormatCreator の拡張的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、FormatCreator 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、スクリプト開発モデルで記述されています。そのため、スクリプト開発モデルに関する理解は必須です。スクリプト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。
- 本書は、FormatCreator の解説を目的とするものではありません。そのため本書を理解するには、基本的な FormatCreator 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。
- 本書は、ワークフロー（ワークフローモジュール）の解説を目的とするものではありません。そのため本書を理解するには、基本的なワークフロー（ワークフローモジュール）に関する理解が必要になります。ワークフローについては、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。

1.3 準備

FormatCreator のサンプルプログラムを実行するための準備をします。
各製品に付属するインストールガイドを参考に、FormatCreator が動作する環境を構築します。

本書では、以下の環境を用いて説明をおこないます。

- intra-mart WebPlatform (アドバンスト)

各製品については、最新パッチを適応することを推奨します。

各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。

2 クラス指定

クラス指定とは、申請書作成時に“クラス指定“項目を設定することで、作成した申請書から任意のプログラムを呼び出すことが可能となる機能です。

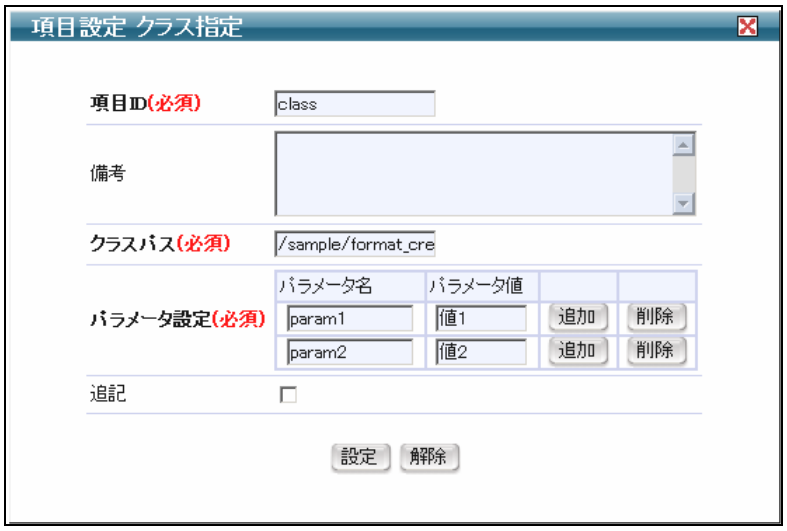
ここでは、以下のサンプルプログラムを例に説明を行います。

%Resource Service%/pages/src/sample/format_creator/class/class_sample.html
%Resource Service%/pages/src/sample/format_creator/class/class_sample.js

2.1 サンプル実行の準備

1. [FormatCreator] - [マスタメンテナンス] – [申請書メンテナンス]画面より、次のような申請書を作成します。

＜申請書のプレビュー画面＞



＜項目設定　クラス指定画面＞

クラスパスには、以下のプログラム(拡張子なし)を指定します。
項目 ID　　：任意の ID
クラスパス　　：/sample/format_creator/class/class_sample
パラメータ設定　：param1 - 値 1
　　Param2 - 値 2

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

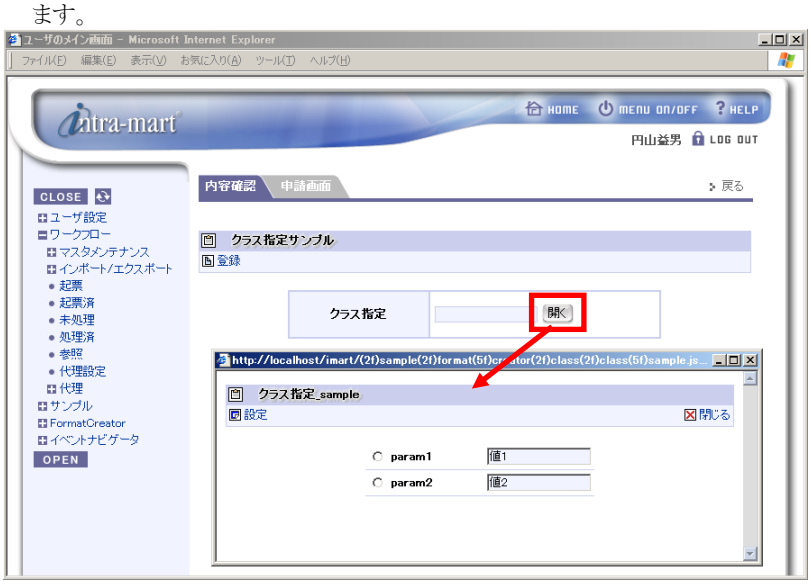
2. [FormatCreator] - [マスタメンテナンス] – [ワークフロー連携]画面より、申請書をワークフローに連携します。

2.2 サンプルの実行

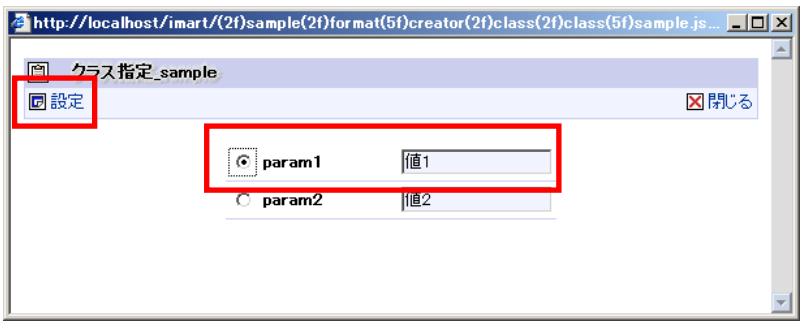
1. [ワークフロー] - [起票]]画面より、作成した申請書を選択し、起票画面を表示します。

＜起票画面＞

2. クラス設定項目の「開く」ボタンを押下すると指定したサンプルプログラムがポップアップ画面で表示されます。



3. ポップアップで開いたサンプルプログラム画面で親画面に引き渡すパラメータを選択して「設定」ボタンを押下します。



＜クラス指定のサンプルプログラム画面＞

4. 以上の手順を行うことにより、任意作成したプログラムからの値を取得し、申請書に表示することが可能となります。

＜任意のプログラム画面から値を取得した画面＞

2.3 サンプルプログラムの解説

2.3.1 申請書画面から引き渡されるパラメータについて

クラス指定で指定したクラスパス(プログラム)には、クラス指定で指定したパラメータ名／パラメータ値が request オブジェクトとして引き渡されます。
また、“呼び出し元の識別用パラメータ”として、「call_back」というパラメータが request オブジェクトとして引き渡されます。

＜サンプルプログラムの request オブジェクト＞

パラメータ名	パラメータ値
param1(任意のパラメータ)	値 1
param2(任意のパラメータ)	値 2
call_back(固有のパラメータ)	呼び出し元の識別用パラメータ

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

2.3.2 申請書画面への引き渡すパラメータについて

作成するプログラム（HTML）では、呼び出し元(申請書)画面へ値を格納する処理を必ず実装する必要があります。

サンプルプログラムでは、「設定」ボタンの押下時に CSJS 関数「call_back」が実行されます。
呼び出し元(申請書)画面へ値を格納する処理は、25 行目に実装しています。
「<IMART type="string" value=callBack></IMART>」は、request オブジェクトから取得したパラメータ「call_back」をバインドした値です。
「sel_value」は、画面から選択した値を格納しています。

＜サンプルプログラム（class_sample.html）＞

```
07  function call_back() {
08      var sel_value;
09
10      // 選択された値の格納
11      if(document.mainForm.example.value != undefined && document.mainForm.example.checked){
12          sel_value = document.mainForm.example.value;
13      }else{
14          for(var i = 0;i<document.mainForm.example.length;i++){
15              if(document.mainForm.example[i].checked == true){
16                  sel_value = document.mainForm.example[i].value;
17              }
18          }
19      }
20
21      if(sel_value == undefined){
22          sel_value = "";
23      }
24
25      window.opener.document.<IMART type="string" value=callBack></IMART>.value = sel_value;
26      window.close();
27  }
```

＜サンプルプログラム（class_sample.js）＞

```
 2  var callBack;

12  function init(request) {
13
14      // 初期化
15      var i =1;
16      var cnt = 0;
17      var cnt_flg = true;
18      param_list = new Array();
19      callBack = request.call_back;
20
```

3 審議タスクにおいて追記する機能

審議タスクにおいて追記する機能とは、審議タスクの処理者（以下、承認者）が申請書の内容を追加／更新する機能です。

ここでは、FormatCreator の申請書サンプルデータである**「01 勤怠関連 休暇申請」**と以下のサンプルプログラムを例に説明を行います。

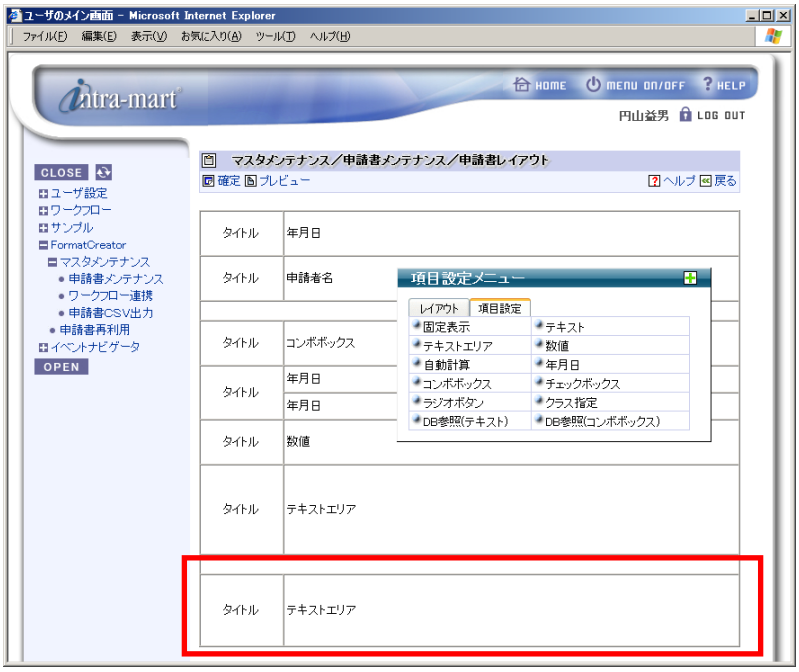
%Resource Service%/pages/src/sample/format_creator/appendix/draft_post_sample.js
%Resource Service%/pages/src/sample/format_creator/appendix/approve_post_sample1.js

ここでは WebPlatform/AppFramework で提供する API を使用しますので、API リストも合わせて参照してください。

3.1 サンプルの解説

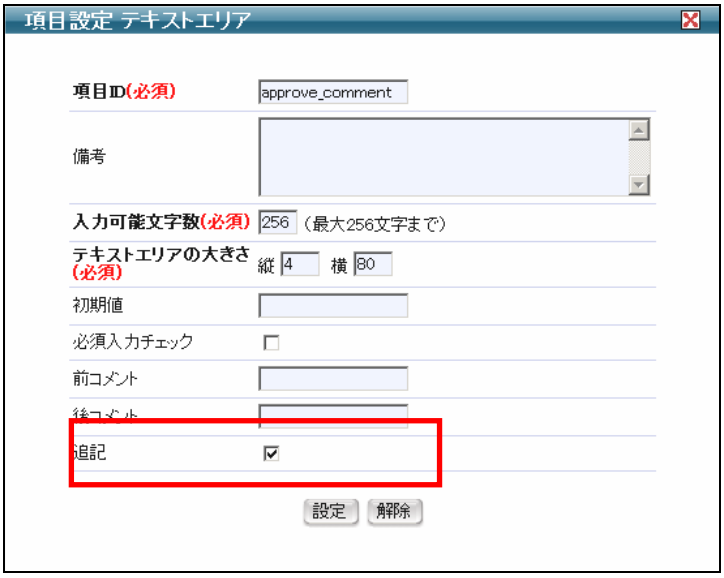
申請書サンプルデータである**「01 勤怠関連 休暇申請」**では、一番下の行にある「タイトル」と「テキストエリア」の項目に「追記」のチェックが入っています。

「追記」のチェックがついている項目は、起票タスク（申請／再申請）の際に表示されない項目として設定されます。審議タスク（承認）においては、「表示/非表示」・「追記可/追記不可」の状態を FormatCreator が提供する API を使用して制御することが出来ます。



＜「01 勤怠関連 休暇申請」の申請書メンテナンス画面＞

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.



＜項目設定　テキストエリア画面＞

3.2 サンプル実行の準備

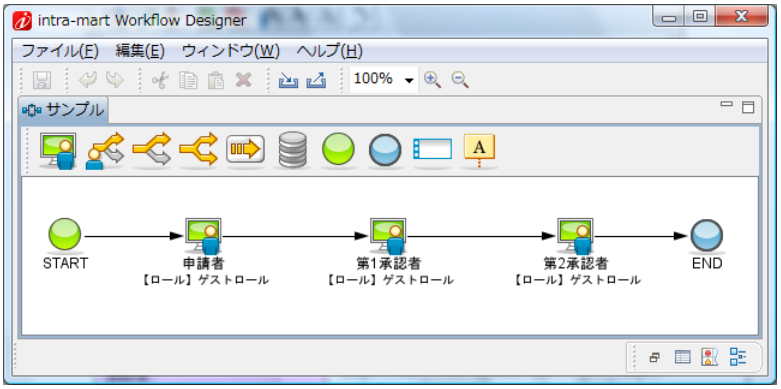
1. [FormatCreator] - [マスタメンテナンス] – [ワークフロー連携]画面より、「01 勤怠関連 休暇申請」をワークフローに連携します。

＜ワークフロー連携画面＞

1. [ワークフロー] - [マスタメンテナンス] – [プロセス定義]画面より、先ほどワークフロー連携を行ったプロセス定義を選択して編集を行います。

<プロセス定義一覧画面>

2. ルート設定画面より、以下のようなフローに変更します。



■ 申請者の設定

キャプション	申請者
後処理	/sample/format_creator/appendix/draft_post_sample
対象者	【ロール】ゲストロール

■ 第 1 承認者の設定

キャプション	第 1 承認者
後処理	/sample/format_creator/appendix/approve_post_sample1
対象者	【ロール】ゲストロール

■ 第 2 承認者の設定

キャプション	第 2 承認者
後処理	使用しない
対象者	【ロール】ゲストロール

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

3.3 サンプルの実行

1. [ワークフロー] - [起票]]画面より、作成した申請書を選択し、起票画面を表示します。

「追記」のチェックが入っていた項目が表示されていないことが確認できます。

申請書の内容を入力し、申請処理を行ってください。

＜起票画面＞

2. [ワークフロー] - [未起票]]画面より、先ほど申請した案件を選択し、審議画面を表示します。

「追記」のチェックが入っていた項目が表示され、テキストエリアが入力可能になっているのが確認できます。

“承認コメント”の内容を入力し、承認処理を行ってください。

＜審議画面＞

3. [ワークフロー] - [未起票]]画面より、先ほど申請および承認した案件を選択し、審議画面を表示します。

「追記」のチェックが入っていた項目が表示され、“第1承認者”が入力した内容が表示されているのが確認できます。

また、“第 2 承認者”は入力した内容が変更できないことを確認できます。

＜審議画面＞

3.4 サンプルプログラムの解説

3.4.1 “申請者”タスクの後処理プログラム

”第 1 承認者”タスクの審議画面で、「追記」のチェックが入っていた項目が表示されテキストエリアが入力可能になるようにするには、”申請者”タスクの後処理プログラムで changePostScriptFlag メソッドを実行し、追記フラグの状態を変更する必要があります。

”申請者”タスクの後処理プログラムで、changePostScriptFlag メソッドを実行しなかった場合は、申請画面と同じように何も表示されません。

後処理のサンプルプログラムは、以下を参照してください。

%Resource Service%/pages/src/sample/format_creator/appendix/draft_post_sample.js

サンプルプログラム内の 37 行目から記述されている以下の部分を参照してください。

```
36    // ユーザアプリケーションキー取得
37    var oWkfProcess = new WkfProcess( processDefCd , versionCd , processCd , groupId );
38    var processInfo = oWkfProcess.getProcessInfo();
```


Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

ワークフローAPI の WkfProcess オブジェクトを利用して、プロセス情報を取得しています。
FormatCreator では、申請時にワークフローのアプリケーションキー配列の第 1 要素に「パラメータコード」を、第 2 要素に「申請書コード」を設定していますので、WkfProcess オブジェクトを使用することでプロセス情報内にあるアプリケーションキー情報から「パラメータコード」、「申請書コード」を取得することが出来ます。

次にサンプルプログラム内の 122 行目から記述されている以下の部分を参照してください。

```
117  // Format Creator 追記フラグの状態を変更します。
118  // この変更内容によって、追記の可否・表示の有無が変わります。
119  var ary = new Array();
120
121  // 承認コメントのタイトルを表示へ変更
122  var obj = new Object();
123  obj.item_id = "title7";
124  obj.postscript_flag = FcSwitchPostScriptFlag.PS_READONLY;
125  ary[0] = obj;
126
127  // 承認コメントのテキストエリアを追記可能へ変更
128  obj = new Object();
129  obj.item_id = "approve_comment";
130  obj.postscript_flag = FcSwitchPostScriptFlag.PS_WRITE;
131  ary[1] = obj;
```

追記フラグを変更する項目の「項目情報オブジェクト」を生成しています。

配列に複数の「項目情報オブジェクト」を設定することで、changePostScriptFlag メソッドを一度実行するだけで複数の項目の追記フラグを変更することが出来ます。
サンプルプログラムでは、“承認コメントタイトル(title7)”と“承認コメントテキストエリア(approve_comment)”の追記フラグを変更するので、2つの「項目情報オブジェクト」を生成し配列に設定しています。

“承認コメントタイトル”は状態を「表示」に変更するので、定数「FcSwitchPostScriptFlag.PS_READONLY」を指定しています。
“承認コメントテキストエリア”は状態を「追記可能」に変更するので、定数「FcSwitchPostScriptFlag.PS_WRITE」を指定しています。

【postscript_flag の指定(定数)】

非表示	FcSwitchPostScriptFlag.PS_INVISIBLE
表示(追記不可)	FcSwitchPostScriptFlag.PS_READONLY
表示(追記可能)	FcSwitchPostScriptFlag.PS_WRITE

次にサンプルプログラム内の 134 行目から記述されている以下の部分を参照してください。

```
133  // 追記フラグの変更を実行します
134  var oFcSwitch = new FcSwitchPostScriptFlag(userId , groupId);
135  var result = oFcSwitch.changePostScriptFlag(processInfo.app_key[0], processInfo.app_key[1], ary);
136  if (!result.isSuccess) {
137  return result.getMessage;
138  }
```

FcSwitchPostFlag オブジェクトのコンストラクタを生成し、changePostScriptFlag メソッドを実行しています。

changePostScriptFlag メソッドは、次のようなパラメータを指定します。

- ◆ 第 1 引数 ： パラメータコード
- ◆ 第 2 引数 ： 申請書コード
- ◆ 第 3 引数 ： 項目情報オブジェクト配列

3.4.2 “第 1 承認者”タスクの後処理プログラム

”第 2 承認者”タスクの審議画面で、「追記」のチェックが入っていた項目が表示され、“第 1 承認者”タスクの審議画面で入力された情報を表示する（入力できない状態にする）には、”申請者”タスクの後処理プログラムと同様に、”第 1 承認者”タスクの後処理プログラムで changePostScriptFlag メソッドを実行し、追記フラグの状態を変更する必要があります。

後処理のサンプルプログラムは、以下を参照してください。

%Resource Service%/pages/src/sample/format_creator/appendix/approve_post_sample1.js

次にサンプルプログラム内の 40 行目から記述されている以下の部分を参照してください。

```
39    // 承認コメントのテキストエリアを追記不可へ変更
40    obj = new Object();
41    obj.item_id = "approve_comment";
42    obj.postscript_flag = FcSwitchPostScriptFlag.PS_READONLY;
43    ary[0] = obj;
```

”申請者”タスクの後処理プログラムと同様に、追記フラグを変更する項目の「項目情報オブジェクト」を生成しています。
ここでは、“承認コメントテキストエリア(approve_comment)”の追記フラグのみを変更するので、1つの「項目情報オブジェクト」を生成し配列に設定しています。

“承認コメントテキストエリア”は状態を「表示」に変更するので、定数「FcSwitchPostScriptFlag.PS_ READONLY」を指定しています。
“承認コメントタイトル”は状態を変更しません。

あとは、”申請者”タスクの後処理プログラムと同様に、FcSwitchPostFlag オブジェクトのコンストラクタを生成し、changePostScriptFlag メソッドを実行しています。

このように、「追記フラグ」の状態を変更することで、タスク毎に項目の表示/非表示、追記の可否を自由に変更することが出来ます。

このサンプル内では「追記」のチェックが入っている項目のみを変更していますが、「追記」のチェックが入っていない項目においても、「追記フラグ」の状態を変更することは可能です。
この場合は、“申請者”タスクで入力した項目を承認者が変更するといったことも可能です。

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

4 分岐プログラムの作成

FormatCreator で作成された申請書で入力された値を使用して、ワークフローの分岐プログラムを作成する方法を説明します。

ここでは、FormatCreator の申請書サンプルデータである「**23 総務関連 稟議申請**」と以下のサンプルプログラムを例に説明を行います。

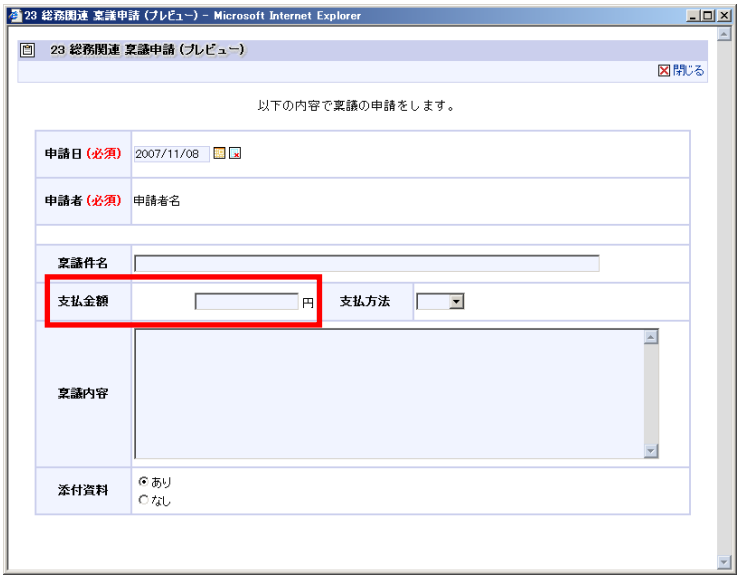
%Resource Service%/pages/src/sample/format_creator/appendix/branch_condition_sample.js

ここでは WebPlatform/AppFramework で提供する API を使用しますので、API リストも合わせて参照してください。

4.1 サンプルの解説

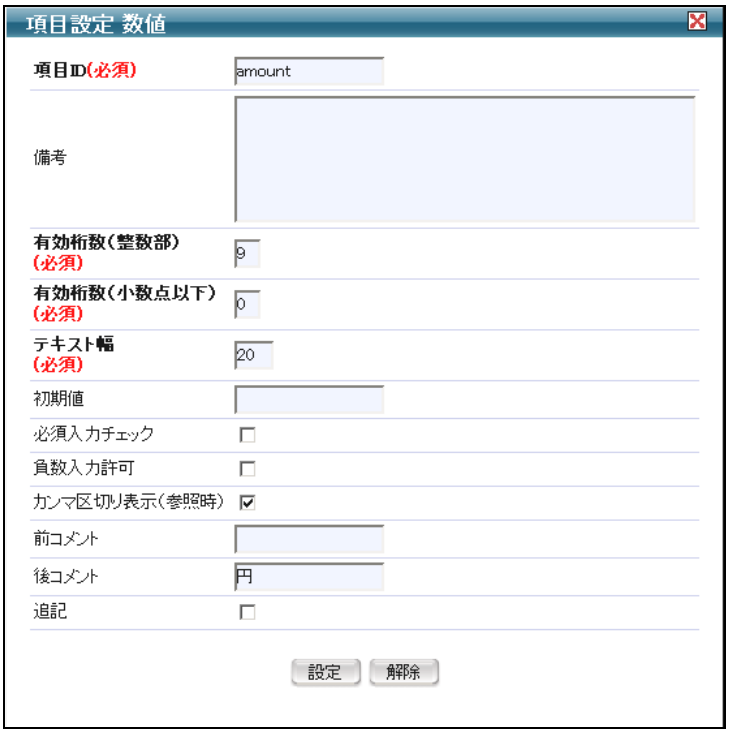
「**23 総務関連 稟議申請**」の入力項目である"支払金額"を条件にして、ワークフローのルートを制御する方法を説明します。

ここで説明するサンプルプログラムは、"支払金額"を分岐の条件プログラムで取得し、"支払金額"が 10,000 円以上の場合は承認者へ、10,000 円未満の場合はワークフローが完了するルートとなっています。



<「23 総務関連 稟議申請」プレビュー画面>

＜「23 総務関連 稟議申請」の申請書メンテナンス画面＞



＜項目設定 数値画面＞

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

4.2 サンプル実行の準備

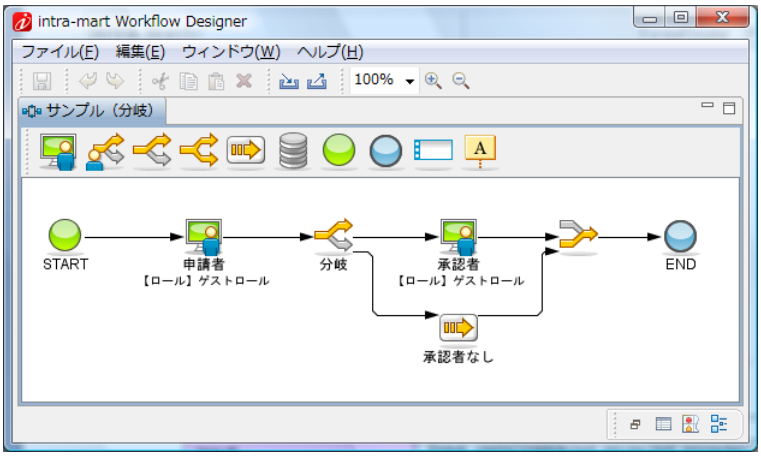
1. [FormatCreator] - [マスタメンテナンス] – [ワークフロー連携]画面より、「23 総務関連 稟議申請」をワークフローに連携します。

＜ワークフロー連携画面＞

2. [ワークフロー] - [マスタメンテナンス] – [プロセス定義]画面より、先ほどワークフロー連携を行ったプロセス定義を選択して編集を行います。

＜プロセス定義一覧画面＞

3. ルート設定画面より、以下のようなフローに変更します。



■ 申請者の設定

キャプション	申請者
後処理	/format_creator/common/post_process/post_processing
対象者	【ロール】ゲストロール

■ 承認者の設定

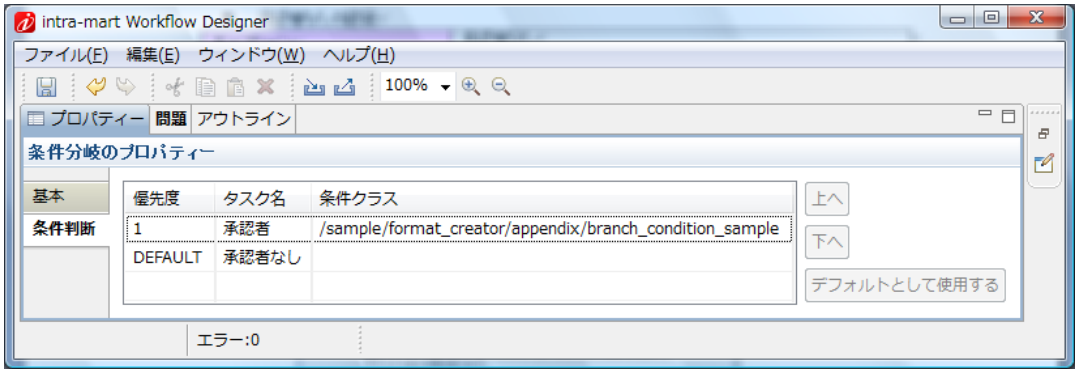
キャプション	承認者
後処理	使用しない
対象者	【ロール】ゲストロール

■ 承認者なしの設定

キャプション	承認者なし

■ 分岐の設定

キャプション	分岐
分岐種別	単一ルート
条件クラス	/sample/format_creator/appendix/branch_condition_sample



Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

4.3 サンプルの実行

1. [ワークフロー] - [起票]]画面より、作成した申請書を選択し、起票画面を表示します。
“支払金額”に「10000」を入力し、申請処理を行ってください。

＜起票画面＞

2. 同様に、[ワークフロー] - [起票]]画面より、作成した申請書を選択し、起票画面を表示します。
今度は、“支払金額”に「9999」を入力し、申請処理を行ってください。

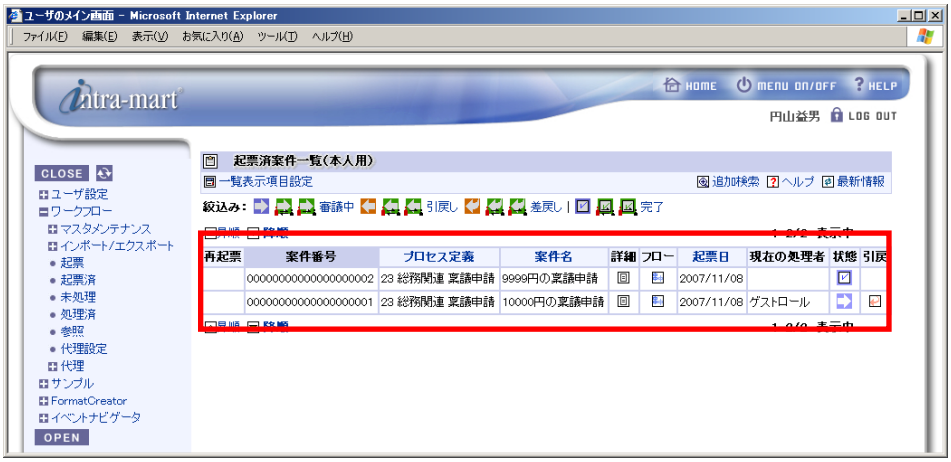
＜起票画面＞

3. [ワークフロー] - [起票済]]画面より、申請した案件を確認します。

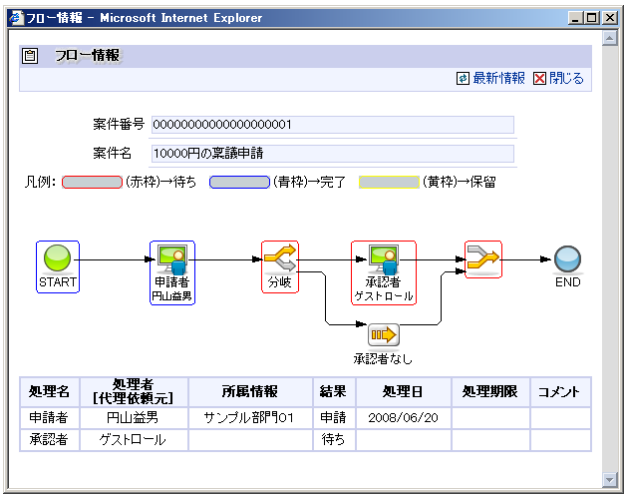
“支払金額”に「10000」を入力した案件は、状態が審議中であることが確認できます。

“支払金額”に「9999」を入力した案件は、状態が完了であることが確認できます。

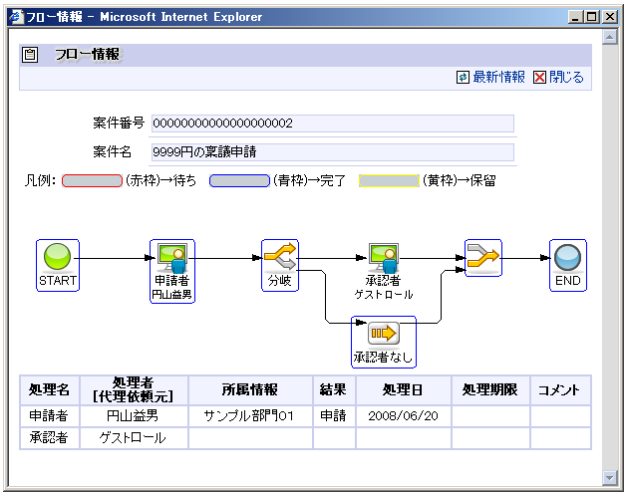
合わせて、フロー情報画面より、フローの情報も確認することが出来ます。



＜起票済画面＞



<「支払金額:10000 円を入力した案件」のフロー情報画面>



<「支払金額:9999 円を入力した案件」のフロー情報画面＞

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

4.4 サンプルプログラムの解説

”分岐”タスクの条件プログラムで getApplicationDataItem メソッドを実行し、申請書で入力された値を取得しています。
その入力された値を条件に、フローの制御を行います。

後処理のサンプルプログラムは、以下を参照してください。

%Resource Service% /pages/src/sample/format_creator/appendix /branch_condition_sample.js

サンプルプログラム内の 37 行目から記述されている以下の部分を参照してください。

```
34    // ユーザアプリケーションキー取得
35    var oWkfProcess = new WkfProcess( processDefCd , versionCd , processCd , groupId );
36    var processInfo = oWkfProcess.getProcessInfo();
```

ワークフローAPI の WkfProcess オブジェクトを利用して、プロセス情報を取得しています。
FormatCreator では、申請時にワークフローのアプリケーションキー配列の第 1 要素に「パラメータコード」を、第 2 要素に「申請書コード」を設定していますので、WkfProcess オブジェクトを使用することでプロセス情報内にあるアプリケーションキー情報から「パラメータコード」、「申請書コード」を取得することが出来ます。

次にサンプルプログラム内の 39 行目から記述されている以下の部分を参照してください。

```
38    // 申請書情報取得の取得
39    var oFcApp = new FcApplication(userId , groupId);
40    var result = oFcApp.getApplicationDataItem(processInfo.app_key[0], processInfo.app_key[1], "amount");
41    if ( result.isSuccess ) {
42          if ( result.data[0].item_set_value != "" ) {
43                amount = Number(result.data[0].item_set_value);
44          }
45    }
```

FcApplication オブジェクトのコンストラクタを生成し、getApplicationDataItem メソッドを実行しています。

getApplicationDataItem メソッドは、次のようなパラメータを指定します。

- 第 1 引数 ： パラメータコード
- 第 2 引数 ： 申請書コード
- 第 3 引数 ： 項目ID

次にサンプルプログラム内の 48 行目から記述されている以下の部分を参照してください。

```
47    // ”支払金額”の判断
48    if ( amount >= 10000 ) {
49          bRet = true;
50    }
```

getApplicationDataIte メソッドを実行して取得された“支払金額”が、10000 円以上の場合は、返却値に“真（true）”をセットしています。

あとは、ワークフローモジュールにおいて、ルートの制御が行われます。

5 FormatCreator で提供する API

FormatCreator で提供される API FcApplication オブジェクトについて説明します。

FcApplication オブジェクトは、主に申請した申請書の内容を取得する API です。

ここでは、FormatCreator の申請書サンプルデータである「**02 勤怠関連 休日勤務申請**」と以下のサンプルプログラムを例に説明を行います。

%Resource Service%/pages/src/sample/format_creator/appendix/approve_post_sample2.js

ここでは WebPlatform/AppFramework で提供する API を使用しますので、API リストも合わせて参照してください。

5.1 サンプル実行の準備

1. [FormatCreator] - [マスタメンテナンス] – [ワークフロー連携]画面より、「02 勤怠関連 休日勤務申請」をワークフローに連携します。

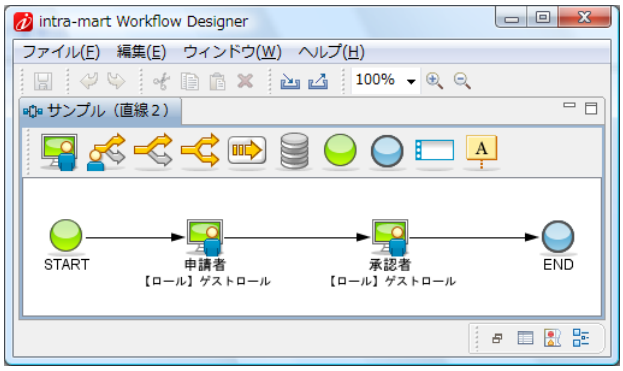
＜ワークフロー連携画面＞

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

2. [ワークフロー] - [マスタメンテナンス] – [プロセス定義]画面より、先ほどワークフロー連携を行ったプロセス定義を選択して編集を行います。

＜プロセス定義一覧画面＞

3. ルート設定画面より、以下のようなフローに変更します。



■ 申請者の設定

キャプション	申請者
後処理	/format_creator/common/post_process/post_processing
対象者	【ロール】ゲストロール

■ 承認者の設定

キャプション	承認者
後処理	/sample/format_creator/appendix/approve_post_sample2
対象者	【ロール】ゲストロール

5.2 サンプルプログラムの解説

ここで説明するサンプルプログラムは、後処理プログラムとして実装されています。FormatCreator で提供される API を使用し、取得した結果を Service Platform のコンソール上に出力しています。

サンプル実行の準備が終了したら、ワークフローから申請／承認と処理を行ってください。
承認まで終了したら、stdout.log ファイルを参照してください。

stdout.log ファイルは以下の場所に出力されます。

% Service Platform % /log/stdout.log

後処理のサンプルプログラムは、以下を参照してください。

%Resource Service%/pages/src/sample/format_creator/appendix/approve_post_sample2.js

5.2.1 getApplicationBasicData メソッド

このメソッドは申請書の申請書基本情報を取得する場合に使用します。
getApplicationBasicData メソッドは、次のようなパラメータを指定します。

- 第 1 引数 ： 申請書コード

処理結果には引数に指定された条件に該当する申請書の「申請書基本情報オブジェクト」情報を取得することが出来ます。
処理結果が成功(true)の場合に、「処理結果オブジェクト」の取得情報(data)プロパティに「申請書基本情報オブジェクト」が格納されています。

実装例は、サンプルプログラム内の 41 行目からの下記の部分を参照してください。

```
41    result = oFcApplication.getApplicationBasicData(processInfo.app_key[1]);
42    if (!result.isSuccess) {
43        // データ取得失敗時はエラーメッセージを返却
44        return result.getMessage;
45    }
46    if (result.data.length > 0) {
47        // コンソール上に取得したデータを表示します。
48        Debug.print("------ application_cd     : " + result.data[0].application_cd);
49        Debug.print("------ application_name : " + result.data[0].application_name);
50        Debug.print("------ note                : " + result.data[0].note);
51    }
```

サンプルプログラムの実行結果は、stdout.log ファイルの“1[getApplicationBasicData]Debug—start”から“1[getApplicationBasicData]Debug--end”までを参照してください。
「申請書基本情報オブジェクト」の内容が取得できていることが確認できます。

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

5.2.2 getApplicationData メソッド

このメソッドは申請した申請書の内容を取得する場合に使用します。
getApplicationData メソッドは、次のようなパラメータを指定します。

- 第 1 引数 ： パラメータコード
- 第 2 引数 ： 申請書コード

処理結果には引数に指定された条件に該当する申請書の「申請書情報オブジェクト」情報を取得することが出来ます。
処理結果が成功(true)の場合に、「処理結果オブジェクト」の取得情報(data)プロパティに「申請書情報オブジェクト」が格納されています。

実装例は、サンプルプログラム内の 57 行目からの下記の部分を参照してください。

```
57    result = oFcApplication.getApplicationData(processInfo.app_key[0], processInfo.app_key[1]);
58    if (!result.isSuccess) {
59          // データ取得失敗時はエラーメッセージを返却
60          return result.getMessage;
61    }
62    if (result.data.length > 0) {
63          debugApp(result.data[0]);
64    }
```

サンプルプログラムの実行結果は、stdout.log ファイルの“2[getApplicationData]Debug--start”から“2[getApplicationData]Debug--end”までを参照してください。
「申請書情報オブジェクト」の内容が取得できていることが確認できます。

5.2.3 getApplicationDataList メソッド

このメソッドは申請した申請書の内容を一括で取得する場合に使用します。
getApplicationDataList メソッドは、次のようなパラメータを指定します。

- 第 1 引数 ： 申請書コード

処理結果には引数に指定された条件に該当する申請書の「申請書情報オブジェクト」情報配列を取得することが出来ます。
処理結果が成功(true)の場合に、「処理結果オブジェクト」の取得情報(data)プロパティに「申請書情報オブジェクト」配列が格納されています。

実装例は、サンプルプログラム内の 70 行目からの下記の部分を参照してください。

```
70    result = oFcApplication.getApplicationDataList(processInfo.app_key[1]);
71    if (!result.isSuccess) {
72          // データ取得失敗時はエラーメッセージを返却
73          return result.getMessage;
74    }
75    for(var i = 0; i < result.data.length; i++) {
76          debugApp(result.data[i]);
77    }
```

サンプルプログラムの実行結果は、stdout.log ファイルの“3[getApplicationDataList]Debug--start”から“3[getApplicationDataList]Debug--end”までを参照してください。
「申請書情報オブジェクト」配列の内容が取得できていることが確認できます。

5.2.4 getApplicationDataItem メソッド

このメソッドは申請した申請書の項目内容を取得する場合に使用します。
getApplicationDataItem メソッドは、次のようなパラメータを指定します。

- 第 1 引数 ： パラメータコード
- 第 2 引数 ： 申請書コード
- 第 3 引数 ： 項目 ID

処理結果には引数に指定された条件に該当する申請書の「申請書項目情報オブジェクト」情報を取得することが出来ます。
処理結果が成功(true)の場合に、「処理結果オブジェクト」の取得情報(data)プロパティに「申請書項目情報オブジェクト」が格納されています。

実装例は、サンプルプログラム内の 83 行目からの下記の部分を参照してください。

```
83    result = oFcApplication.getApplicationDataItem(processInfo.app_key[0],
                                                     processInfo.app_key[1],"approve_comment");
84    if (!result.isSuccess) {
85          // データ取得失敗時はエラーメッセージを返却
86          return result.getMessage;
87    }
88    for (var j = 0; j < result.data.length; j++) {
89          Debug.print("----- item_set_value       : " + result.data[j].item_set_value);
90          Debug.print("----- item_set_group_cd    : " + result.data[j].item_set_group_cd);
91          Debug.print("----- item_set_cd          : " + result.data[j].item_set_cd);
92          Debug.print("----- postscript_flag      : " + result.data[j].postscript_flag);
93    }
```

サンプルプログラムの実行結果は、stdout.log ファイルの“4[getApplicationDataItem]Debug--start”から“4[getApplicationDataItem]Debug--end”までを参照してください。
「申請書項目情報オブジェクト」の内容が取得できていることが確認できます。

5.2.5 getPDFData メソッド

このメソッドは申請した申請書の PDF ファイルデータを取得する場合に使用します。
getApplicationDataItem メソッドは、次のようなパラメータを指定します。

- 第 1 引数 ： パラメータコード
- 第 2 引数 ： 申請書コード
- 第 3 引数 ： フロー表示フラグ(true/false)
- 第 4 引数 ： 改ページフラグ（true/faise 「フロー表示フラグ=treu の場合のみ有効」）

処理結果には引数に指定された条件に該当する申請書の「PDF ファイルデータ(バイナリデータ)」を取得することが出来ます。
フロー表示フラグに true を指定すると、申請書と共にワークフローのフロー情報を出力することが出来ます。
フロー表示フラグが true の場合に、改ページフラグを true にすると申請書とフロー情報が別のページとして出力できます。

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

処理結果が成功(true)の場合に、「処理結果オブジェクト」の取得情報(data)プロパティに「PDF ファイルデータ(バイナリデータ)」が格納されています。

実装例は、以下を参照してください。

%Resource Service%/pages/ platform/src/formatCreator/wkf/ applied_view.js

getPDF 関数の下記の部分を参照してください。

```
var app = new FcApplication();
var result = app.getPDFData(userApplicationKeys[0], userApplicationKeys[1], flow_flag, next_flag);
if(result.isSuccess) {
	// PDF ファイルデータ取得成功時
	Module.download.send(result.data, "default.pdf");
} else {
	// PDF ファイルデータ取得失敗時
	Module.alert.back("FC_SE0997", result.getMessage);
}
```

アプリケーション共通モジュール「Module.download オブジェクト send メソッド」を使用することで、取得した PDF ファイルデータをクライアントへ送信し、指定された申請書の PDF ファイルをクライアントでダウンロードできるようにしています。

また、「VirtualFile オブジェクト save メソッド」を使用すれば、指定された申請書の PDF ファイルを Storage Service 上の任意のフォルダに配置することも可能です。

6 申請書をサーバ上に保存する

ワークフローが終了した案件の申請書をサーバ上(Storage Service)に PDF ファイルとして保存する方法を説明します。

FormatCreator の申請書サンプルデータである「**25 社内システム関連 メール取得申請**」と以下のサンプルプログラムを例に説明を行います。

%Resource Service%/pages/src/sample/format_creator/appendix/end_execution_sample.js

ここでは WebPlatform/AppFramework で提供する API を使用しますので、API リストも合わせて参照してください。

6.1 サンプル実行の準備

1. [FormatCreator] - [マスタメンテナンス] – [ワークフロー連携]画面より、「25 社内システム関連 メール取得申請」をワークフローに連携します。

＜ワークフロー連携画面＞

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

2. [ワークフロー] - [マスタメンテナンス] – [プロセス定義]画面より、先ほどワークフロー連携を行ったプロセス定義を選択して編集を行います。

＜プロセス定義一覧画面＞

3. ルート設定画面より、以下のようなフローに変更します。

■ 申請者の設定

キャプション	申請者
後処理	/format_creator/common/post_process/post_processing
対象者	【ロール】ゲストロール

■ 承認者の設定

キャプション	承認者
後処理	使用しない
対象者	【ロール】ゲストロール

■ END の設定

キャプション	end
案件終了時実行クラス	/sample/format_creator/appendix/end_execution_sample

6.2 サンプルの実行

1. [ワークフロー] - [起票]]画面より、作成した申請書を選択し、起票画面を表示します。申請書の内容を入力し、申請処理を行ってください。

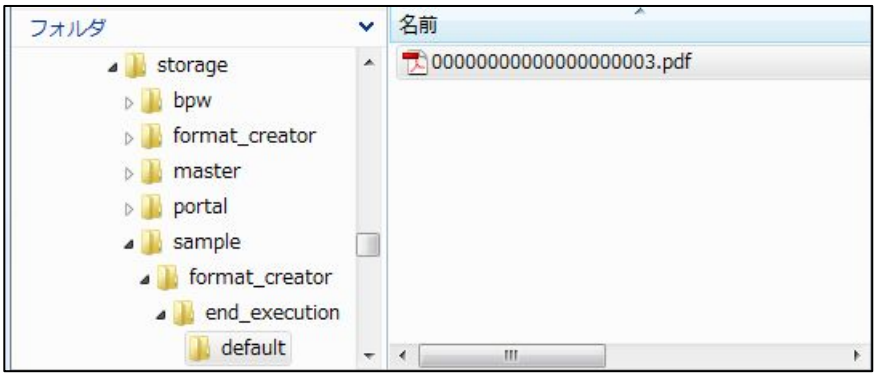
＜起票画面＞

2. [ワークフロー] - [未処理]]画面より、申請された案件を選択し、審議画面を表示します。申請書の内容を確認し、承認処理を行ってください。

＜審議画面＞

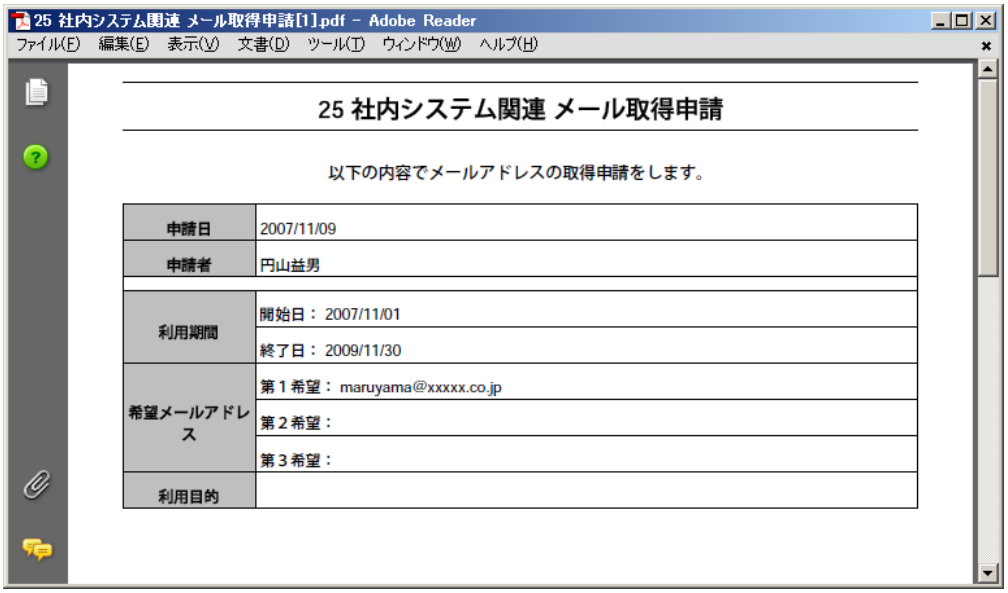
Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

3. 承認処理が正常に終了すると、サーバ上の[Storage Service]に PDF ファイルの申請書が保存されています。



＜Windows のエクスプローラ画面＞

4. ファイルをを確認することが出来ます。



＜表示した申請書のファイル＞

6.3 サンプルプログラムの解説

ワークフローの案件終了時にサンプルプログラムを実行することで、サーバ上（Storage Service）にファイルを保存する方法を説明します。

ワークフローの終了タスクに案件終了時実行クラスを指定しています。
サンプルプログラムでは、FormatCreator の API 「FcApplication オブジェクト getPDFData メソッド」を使用します。

案件終了時実行クラスのサンプルプログラムは、以下を参照してください。

%Resource Service% /pages/src/sample/format_creator/appendix /end_execution_sample.js

サンプルプログラムで行っている手順は以下のようになります。

① ワークフローのプロセス情報(アプリケーションキー情報)を取得する。
② PDF ファイルデータを取得する。
③ PDF ファイルデータを Storage Service 内の指定ディレクトリへ配置する。

サンプルプログラム内の 40 行目から記述されている以下の部分を参照してください。

```
39    // 案件情報よりアプリケーションキーを取得する。
40    var oWkfProcess = new WkfProcess(processDefCd , versionCd , processCd,groupId);
41    var processInfo = oWkfProcess.getProcessInfo();
42    if(isBlank(processInfo) || isBlank(processInfo.app_key) || processInfo.app_key.length < 2){
43        // アプリケーションキーがない場合
44        return STATUS_COMPLETED;
45    }
```

ここでは、アプリケーションキー情報に設定されている「パラメータコード」、「申請書コード」を取得する為に、ワークフローAPI の WkfProcess オブジェクトを利用して、プロセス情報を取得しています。

ここまでが①に該当します。

Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

次に、サンプルプログラム内の 48 行目から記述されている以下の部分を参照してください。

```
47    // PDF 作成
48    var fcApplication = new FcApplication(userId, groupId);
49    var pdfData = fcApplication.getPDFData(processInfo.app_key[0], processInfo.app_key[1], false, false);
50    if(!pdfData.isSuccess) {
51          // PDF 出力失敗
52          return pdfData.getMessage;
53    }
```

ここでは、FcApplication オブジェクト getPDFData メソッドを使用して、申請書の PDF ファイルデータを取得しています。

ここまでが②に該当します。

次に、サンプルプログラム内の 56 行目から記述されている以下の部分を参照してください。

```
55    // PDF ファイルを対象フォルダに配置
56    var targetFile = new VirtualFile(releasePath + processCd + ".pdf");
57    if(!targetFile.save(pdfData.data)) {
58          // ファイル出力失敗
59          return STATUS_FAULT;
60    }
```

ここでは、申請書の PDF ファイルデータを Storage Service 上の指定ディレクトに配置しています。
Storage Service へ PDF ファイルを配置する際には、 アプリケーション共通モジュール VirtualFile オブジェクトを使用しています。

ここまでが③に該当します。

以上が、サーバ上（Storage Service）にファイルを保存する方法になりますが、サーバ上に保存したファイルをダウンロードする仕組みは別途必要になります。

intra-mart WebPlatform/AppFramework Ver.7.0
FormatCreator　プログラミングガイド

2008/07/07 初版

Copyright 2000-2008 株式会社NTTデータ イントラマート
All rights Reserved.
TEL: 03-5549-2821
FAX: 03-5549-2816
E-MAIL: info@intra-mart.jp
URL: http://www.intra-mart.jp/