intra-mart WebPlatform/AppFramework
Ver.7.0

BPM|Designer 操作ガイド

2008/09/30 第2版

❖ 変更履歴

変更年月日	変更内容
2008/08/22	初版
2008/09/30	第2版　インポートデータ作成時に出力XMLの範囲指定方法追加（2.6.2.1）

Contents

第1章　基本編 1

はじめに 2

1.1.1 用語解説 2

1.1.2 概要 4

1.1.3 デザイナの画面構成 5

プロジェクトの作成 11

1.2.1 操作手順 11

ダイアグラムの作成 14

1.3.1 .bpmの作成 14

1.3.2 業務フローのデザイン 15

ヒューマンタスク 22

1.4.1 ヒューマンタスクの種類 22

1.4.2 比較 23

1.4.3 XFormを使用したヒューマンタスクの作成手順 23

マッピング 33

1.5.1 データの持ち方について 34

1.5.2 データのマッピング 34

1.5.3 XPath関数 37

1.5.4 分岐条件/ループ条件のマッピング 38

1.5.5 ビルトインデータ 39

デプロイ 41

管理コンソール 44

1.7.1 権限の設定 44

1.7.2 管理コンソールの表示 45

第2章　応用編 49

権限設定 50

2.1.1 概要 50

2.1.2 権限の種類 50

2.1.3 権限の設定 50

リレーション・フォーム 68

2.2.1 概要 68

2.2.2 リレーション・フォームの作成 69

2.2.3 フォーム・リレーション エディタ 71

2.2.4 モデル生成 78

2.2.5 ヒューマンタスクの配置 81

外部WebService連携 82

2.3.1 WebService連携とは 82

2.3.2 WebServiceの取込 82

2.3.3 WebService連携用タスクの配置 89

2.3.4 WebService呼出時のパラメータ設定 106

2.3.5 WebService連携時の例外を補足する 107

2.3.6 イントラマートへのログイン 111

プロセスのWebService化 120

2.4.1 概要 120

2.4.2 WebService化手順 120

2.4.3 WebServiceプロセスのWSDL 123

2.4.4 例外メッセージの設定 126

非同期処理プロセス 128

2.5.1 非同期処理プロセスとは 128

2.5.2 相関セットキー 128
2.5.3 メッセージ中間イベント 129
2.5.4 外部WebServiceを利用した非同期プロセス 146
リスク・コントロールマトリックス 160
2.6.1 概要 160
2.6.2 操作手順 165
2.6.3 その他 211
進捗通知 221
2.7.1 概要 221
2.7.2 事前登録パターン 222
2.7.3 都度登録パターン 230
2.7.4 Webサービスの処理ユーザの登録 234
2.7.5 注意点 235
2.7.6 進捗一覧画面 236

第3章 参考 237

リファレンス 238
3.1.1 データベースのクリーニング 238
3.1.2 プロセス変数の作成 238
3.1.3 Escalateタスクの利用 240
3.1.4 処理タスクの期限の設定 241
3.1.5 カスタムXPath関数の作成 243

intra-mart WebPlatform/AppFramework

第1章　基本編

1.1 はじめに

BPM|DesignerとはBPMN(Business Process Modeling Notation)に基づいた表記法で、業務プロセス（ビジネス・プロセス・ダイアグラム）を記述するためのクライアントツールです。作成した業務プロセスはBPM|Serverに配置（デプロイ）することで、実行できるようになります。
本書では、ダイアグラムを作成するためのプロジェクトの作成からデプロイするまでの操作手順について解説します。
BPM|Designerのインストール方法については「BPM|Designerセットアップガイド」、デプロイ後の実行方法については「BPMユーザ操作ガイド」および、2.3 外部WebService連携を参照してください。

1.1.1 用語解説

ビジネス・プロセス・ダイアグラム（ダイアグラム）	BPMNに基づいて作成された業務プロセスを指します。 BPM|Designerでは.bpmという拡張子のファイルとして作成します。
プロセス（業務プロセス）	一連の業務の流れを指します。ビジネス・プロセス・ダイアグラムと同義です。
iWP/iAF	intra-mart WebPlatform/AppFrameworkを指します。

1.1.1.1 BPMNについて

BPMNとはダイアグラムの表記法のことで、定義された様々なアクティビティを使用して業務プロセスを表現します。UMLが技術者寄りの表記法であるのに対して、BPMNは業務担当者やビジネスアナリストのためのコミュニケーションツールです。ゆえに技術者がBPMNを記述する際には、システムフローにならないように注意する必要があります。ここでは、ダイアグラムの例とアクティビティの簡単な概要について記述します。BPMNの詳細については、市販の書籍等を参照してください。

■ プール

プロセスに関わるアクターを示します。プール内には、そのアクターが処理する業務プロセスを後述のタスク、イベント、ゲートウェイなどを組み合わせて記述します。アクターには、その処理を担当する人やシステムなどが含まれます。

■ Task タスク

アクターが処理する業務を表します。シンプルな業務処理を表すタスクの他、人が処理を行うヒューマンタスク（業務画面）や、WebServiceなど他システムが処理するタスク等があります。

■ ◯イベント

業務プロセスのアクション（トリガー）を表します。データの送受信をトリガーにするメッセージイベントや、時間の経過によって発生するタイマーイベントがあります。イベントは、開始、中間、終了に分類され、例えば開始メッセージイベントは他のプールからメッセージを受信すると、プロセスが開始するイベントです。

■ ◇ゲートウェイ

業務プロセスを分岐させる場合に使用します。ゲートウェイには、いくつかの分岐の中から１つを選択することができるほか、複数に分岐させる（並列処理）ことも可能です。

■ ➞シーケンスフロー

業務の流れを表します。イベント、タスク、ゲートウェイをシーケンスフローで接続することで１つの業務プロセスになります。プールをまたぐシーケンスフローを記述することはできません。
また、シーケンスフローにデータの流れは含まれません。データのやりとりについてはメッセージフローで表記します。

■ ⊙⇢メッセージフロー

データの流れを表します。プロセスの中で使用可能なデータはプール単位で共有され、メッセージフロー、は、あるプールで保持しているデータを別のプールに対して送信する目的で使用されます。
つまり、同一プール内でデータのやりとりをする必要は無いため、メッセージフローはプールをまたぐ形で記述されます。
また、実際のデータのやり取りは主にプール内に配置されたタスクやイベント同士で行われます。

ビジネス・プロセスダイアグラム － 例

1.1.2 概要

ビジネス・プロセス・ダイアグラムは以下のような手順で作成します。

プロジェクトの作成	最初にプロジェクトを作成します。プロジェクトにはダイアグラムの他、ダイアグラムを構成するリソース等を格納します。
ダイアグラムの作成	作成済みプロジェクトの下にダイアグラム(.bpmファイル)を作成します。
プールの配置	ダイアグラムに登場する、アクターをプールとして配置します。
アクティビティの配置とシーケンスフローの作成	処理の流れをプールにタスクやイベントなどのアクティビティを配置し、シーケンスフローでつなぎます。
メッセージフローの作成	ダイアグラム内でやり取りされるデータの流れについて、メッセージフローを記述します。データのフォーマットはXMLスキーマで定義します。
ヒューマンワークフローと権限	人が処理するための画面(ヒューマンタスク)を配置します。XFormとリレーションフォームの2種類があります。作成したヒューマンタスクには処理権限を付与します。
マッピング	データを後続の処理タスクへ渡します。メッセージフロー作成時に定義したXMLスキーマの要素同士をマッピングします。
デプロイ	作成したダイアグラムをサーバに配置(デプロイ)します。

外部WebServiceとの連携	ダイアグラム内から、他システムのWebServiceを呼び出す機能です。WSDLファイル内で定義されたオペレーションをドラッグ&ドロップ操作のみで、タスクとして組み込むことができます。
リスク・コントロールマトリックス	タスクに対してリスク・コントロールを関連付ける機能です。 ダイアグラム単位で、関連付けられたリスク・コントロールをPDFファイルに出力することが可能です。
進捗通知	iWP/iAFで公開されているWebServiceをダイアグラムに組み込むことで、業務の進捗状況を確認できる機能です。

1.1.3 デザイナの画面構成

BPM|Designerは、１つのメインウィンドウ（ダイアグラムを作成する等）と、複数のサブウィンドウ（ファイルの一覧や、詳細設定等）で構成されます（メインウィンドウを「エディタ」、サブウィンドウを「ビュー」と呼びます）。このエディタとビューの組み合わせを１セットとして１つの作業環境（パースペクティブ）を構成します。
パースペクティブは以下の３種類が用意されており、作業内容に合わせて切り替えることができます。

- Designer（Modeling）
- フォーム・エディタ
- RCM

BPM|DesignerはIntalio|BPMS Designerをベースとしています。
各機能の詳細については、以下のサイトも参考にしてください。
http://bpms.intalio.com/reference-guides.html

1.1.3.1 Designer(Modeling)

プロジェクトやダイアグラムの新規作成～デプロイ、ダイアグラムのデザイン、マッピング等を行います。「Modeling」では、ダイアグラムの表示領域を大きくすることで、よりモデリングを行いやすくなっています。

Desginerパースペクティブは、以下のように構成されます。

1 ビジネス・プロセス・ダイアグラム

ダイアグラムを作成/編集します。ダイアグラム内に配置するアクティビティ（タスクやゲートウェイ等）は、2．Paletteから選択します。ダイアグラムを保存（ファイル→保存）すると、*.bpmファイルが作成されます。このファイルは、3．プロセス・エクスプローラで確認することができます。

2 Palette

ダイアグラム内に配置するアクティビティやプール等の一覧を表示します。以下の手順で配置します。

①配置したいアクティビティをクリック

②ダイアグラム エディタ上で配置したい場所をクリック

3 プロセス・エクスプローラ

ダイアグラムを構成する要素（ファイル）の一覧を表示します。ダイアグラムを含む全てのファイルは「プロジェクト」の下で管理されます（プロジェクトについては「1．2 プロジェクトの作成」を参照してください）。プロジェクトの中には「build」ディレクトリが自動的に作成され、bpmファイルを元に自動生成されたbpelファイルやwsdlファイル等が格納されます。

4 Outline

ダイアグラムの概要（全体図）を表示します。ビジネス・プロセス・ダイアグラム エディタで表示している領域は強調表示され、マウスでドラッグすることが可能です。

5 マッパー

タスクで処理するデータの受け渡し設定（マッピング）を行います。詳細は「1．5マッピング」を参照してください。

6 プロパティー

アクティビティやプールのプロパティを設定します。

■Process Executionタブ

ラベル：各アクティビティ等の表示名。
テクニカル・ネーム：BPMServerで実行するときに使用される名称。半角英数字（先頭文字は半角英字）が推奨されます。

■Workflowタブ

ヒューマンタスクの実行権限を設定します。詳細は「2．1 権限設定」を参照してください。

その他の項目等の詳細については、以下のサイトも参考にして下さい。
http://bpms.intalio.com/reference-guides.html

1.1.3.2 フォーム・エディタ

ヒューマンタスク用の画面（XForm）を作成します。
フォーム・エディタパースペクティブは以下のように構成されます。

1 Form Editor

業務画面（*.xform)を作成/編集します。2．ワークフローパレットで選択したコンポーネントを使用して画面レイアウトを作成します。作成した業務画面を保存すると.xformファイル（画面レイアウト定義）とXMLスキーマ（入出力データのフォーマット定義）が生成されます。これらのファイルは、プロセス・エクスプローラで確認することができます。

2 ワークフローパレット

業務画面に配置するコンポーネントを選択して、Form Editor上に配置します。

3 プロパティー

コンポーネント単位で、以下のような属性情報を設定します。

テキストフィールド：ラベル名、読み取り専用属性、入力チェック（必須、数値チェック等）

チェックボックス：項目名、値等

1.1.3.3 RCM

リスク・コントロールの参照/編集やタスクへの関連付け、及びPDFファイルの出力等を行います。

詳細は「2．４リスク・コントロールマトリックス」を参照して下さい。

1.2 プロジェクトの作成

BPM|Designerでは、業務フロー（ダイアグラム）あるいは、業務フローを構成する要素（XFormやWSDLなど）を１つのまとまりとして管理するためのビジネス・プロセス・プロジェクトを最初に作成します。（以下、「プロジェクト」と表記します）

1.2.1 操作手順

1 ツールバーから、「ファイル」→「新規」→「ビジネス・プロセス・プロジェクト」を選択します。

2 「プロジェクト名」を入力します。（半角英数）

「終了」ボタンをクリックします。

3 プロセス・エクスプローラに作成したプロジェクトが表示されます。

- 「build」ディレクトリについて
  プロジェクトを作成すると、「build」というディレクトリが自動的に作成されます。
  このディレクトリには、作成したダイアグラムを元に自動生成されたBPELファイルやWSDL及び、XMLスキーマ等が格納されます。このディレクトリの中に、独自で作成したファイル等は置かないようにしてください。

Column

プロジェクトのインポート/エクスポートについて

作成したプロジェクトのバックアップを取る、あるいは他のマシンにデータを移動等を行う場合はインポート/エクスポートを使用します。いくつかある方法の中から、例としてzipファイルでインポート/エクスポートする方法について解説します。

エクスポート：

1．ツールバーから、「ファイル」→「エクスポート」を選択します。
2．エクスポートダイアログで「一般」→「アーカイブ・ファイル」を選択します。

3．「次へ」をクリックします。
4．「宛先・アーカイブファイル」にファイルの出力パスを入力します。

「ZIPフォーマットで保管」を選択して、「終了」をクリックします。

インポート：

1．ツールバーから、「ファイル」→「インポート」を選択します。
2．インポートダイアログで「一般」→「既存プロジェクトをワークスペースへ」を選択して「次へ」をクリックします。
3．「アーカイブ・ファイルの選択」ラジオボタンをクリックします。
4．「参照」ボタンをクリックして、インポートするzipファイルを選択します。

5．「プロジェクト」でインポートするプロジェクトを選択して「終了」をクリックします。

1.3 ダイアグラムの作成

ダイアグラムの作成と業務フローの描画（デザイン）を行います。

1.3.1 .bpmの作成

プロジェクトの下に、ダイアグラム（*.bpmファイル）を作成します。

1 ツールバーから、「ファイル」→「新規」→「ビジネス・プロセス・ダイアグラム」を選択します。

2 ファイル名を入力します。（半角英数）

「終了」をクリックします。

3 ダイアグラム(*.bpmファイル)が作成されます。

1.3.2 業務フローのデザイン

作成されたダイアグラムファイルに業務フローをデザインします。

1.3.2.1 Paletteの表示

Paletteタブを選択します。以降の操作ではPaletteビューからイベントなどのアクティビティを選択して、ダイアグラム上に配置していきます。

1.3.2.2 イベントの配置

イベントとは業務フローの中で発生する何らかのアクションのことで、以下のようなイベントがあります。

	空開始イベント	ダイアグラム全体の中で、最初の起点となるポイントを示します。 ダイアグラム中に１つだけ配置します。
	空終了イベント	プロセスが終了するポイントに配置します。
	メッセージ開始イベント	メッセージ（データ）を受信するとプロセスを開始します。
	メッセージ中間イベント	データの送信および受信、またはその両方を行います。 受信の場合は、データが届くまで後続の処理の実行を待機します。
	メッセージ終了イベント	他のプールへメッセージの送信をしてプロセスを終了します。
	タイマー開始イベント	設定された時間が経過するとプロセスを開始します。
	エラー中間イベント	タスク内で処理を実行中に発生したエラーをキャッチします。

■ダイアグラムへイベントを追加する

ここではダイアグラムを開始するために、「空開始イベント」を追加します。

1 「パレット」の中から「空開始イベント」を選択します。

2 ダイアグラムへ追加します。

● ドラッグ＆ドロップではなく、以下のように操作します。
①パレットの中から空開始イベントを選択（クリック）
②プール上の任意の位置をクリック

1.3.2.3 タスクの配置

1 イベントの配置と同様に、パレットからタスクを選択してフローに追加します。
（※ダイアグラムを新規作成すると、タスクが１つだけ自動的に配置されます）

Column

アクティビティの変更について

アクティビティはダイアグラムに配置した後に変更することが可能です。

1．アクティビティを右クリック→「アクティビティタイプの変更」を選択

※業務フローを記述する際は、最初に業務要件をタスクとして洗い出しておき、後で適切なアクティビティに変更するというアプローチを取ると、効率的にフローを作成することができます。

2 タスクに名前を付ける

「プロパティー」ビューを使用して、タスクのラベル名を設定します。

ラベル	このタスクの処理を表す日本語名を入力します
テクニカル・ネーム	BPM|Serverで実行するときにプログラムが解釈するための名前です。 半角英数字で入力します。

- タスク以外のアクティビティに対しても、ラベルの設定が可能です。
  ダイアグラムを読みやすくするために、ケースに応じて適切なラベルを設定します。

1.3.2.4 シーケンスフローの描画

パレットからシーケンスフローを選択して、空開始イベントからタスクへドラッグ＆ドロップします。

- ダイアグラムを保存するとエラーが表示されますが、これはメッセージの定義とマッピングが設定されていないためであり、ここではそのまま操作を進めるものとします。

1.3.2.5 ゲートウェイ/ループタスクの配置

業務処理によっては、分岐や繰り返しを行う場合があります。BPMNでは、ゲートウェイやループタスクとして表現します。また、いくつかのタスクをサブプロセスとしてまとめてループさせることもできます。

- 分岐処理やループ処理には、分岐条件やループの終了条件の設定が必要です。
  これらの条件設定については、「1．5マッピング」で説明します。

主なゲートウェイを以下に示します。

	排他的分岐 （データ準拠）	設定された分岐の中から１つが選択されます。３つ以上に分岐することも可能です。設定された分岐条件に従ってパスが選択されます。
	排他的分岐 （イベント準拠）	後続フローに設定された中間イベントの内、最初にアクティブとなったパスが選択されます。その他の分岐は無効となり、必ず１つのパスが有効になります。
	並列分岐	後続の分岐が同時並列で実行されます。

■排他的分岐（イベント準拠）の例

下記の例では、ゲートウェイ後続に配置された２つのメッセージ中間イベントの内、先にデータを受信した方のパスが有効になります。つまりこの例では、先に「受注完了」してしまうと、受注キャンセル用のデータは処理されないことになります。

■ダイアグラムへゲートウェイを追加する

排他的分岐（データ準拠）を配置して、後続の一部のタスクが処理される場合と、されない場合のフローを記述します。フローの最後に空終了イベントを置いて、業務プロセスを終了させます。

タスクと同様に、ゲートウェイにも分岐条件をラベルに記述すると処理が分かりやすくなります。

Column

ゲートウェイの結合について

ゲートウェイを使用することで、複数の処理フローの同期をとってから後続の処理を実行させることが可能です。

サンプルフロー１）

サンプルフロー２）

サンプルフロー１の場合、TaskDは前工程の3つのタスク（TaskA、TaskB、TaskC）がすべて処理されるまで実行されません。
サンプルフロー２の場合は、３つのタスクのうちどれか１つがTaskDに到達した段階で、他の２つのタスクの状況に関係なく、処理が続行されます。
結合用のゲートウェイは分岐するときに使用したゲートウェイと同じものを使用します。
（サンプルフロー１の場合は並列で分岐しているので、結合時も並列分岐が配置されています）
また排他的分岐の場合も、結合時にはゲートウェイを置くことが推奨されます。

1.4 ヒューマンタスク

タスクの中にはコンピュータによって自動処理されるものの他に、人が入力をする、あるいは判断を行う必要があるケースがあります。そのような、人が処理するタスク（ヒューマンタスク）用の、画面作成について解説します。

例）

1.4.1 ヒューマンタスクの種類

ヒューマンタスクの作成方法には、以下の２つがあります。

- XForm
- リレーションフォーム

前者は、「フォーム・エディタ」パースペクティブを使用して画面をデザインします。
それに対して後者の場合は、iWP/iAF上で開発されたJavaEEやスクリプト開発の画面をヒューマンタスクとして使用します。

1.4.2 比較

XFormとリレーションフォームの特徴を以下に示します。

	XForm	リレーションフォーム
作成の容易さ	◎専用のエディタで、コントロールを配置することと、プロパティの設定のみで作成可能	△XMLスキーマやAPI等の知識が必要 △画面の作成にはコーディングが必要
レイアウトの自由度	×細かなデザインの変更は不可	◎imartタグやAPIを使用して複雑なレイアウトを作成可能
入力チェック	△必須や数値などの基本的な入力チェックは可能	○コーディング次第で何でも可能
ビジネスロジックの記述	×ビジネスロジックの記述は不可能 サーバサイド処理はXFormとは別のタスクでWebServiceと連携する等で実装します。	○コーディング次第で何でも可能 １つのリレーションフォームタスク内で、複数の画面間を遷移をする、あるいはデータベースとの通信をすることも可能です

シンプルな画面を簡単に作成する場合はXFormを使用し、より複雑なレイアウトの画面やタスク内でビジネスロジックやデータベースと通信する必要がある場合等は、リレーションフォームの使用を検討します。

1.4.3 XFormを使用したヒューマンタスクの作成手順

実際に、XFormを作成する手順について解説します。
リレーションフォームの作成手順については「２．２リレーション・フォーム」を参照してください。

1.4.3.1 .xformの作成

XForm（*.xform）ファイルを新規作成します。

1 ツールバーから、「ファイル」→「新規」→「ワークフロー　フォーム・エディタ」を選択します。

2 ファイル名を入力します（半角英数）

「終了」をクリックします。

XForm(*.xform)ファイルと、XMLスキーマ(*.xsd)ファイルが作成されます。
（XFormファイルは、専用のエディタで開かれた状態になります）

- .xformファイルと同時に自動生成されるXMLスキーマは、XForm画面の入出力データ定義です。
  詳細は「1.4.3.5メッセージフローの描画」で解説します。

1.4.3.2 Form Editor

「フォーム・エディタ」パースペクティブを選択して、画面レイアウトを作成します。

■コンポーネントの配置

「ワークフローパレット」からコンポーネントをドラッグ＆ドロップして配置します。

「プロパティー」で詳細を設定します。

■コンポーネントの主なプロパティ

ラベル	各種コンポーネントの表示名です。テキストフィールド等の場合は、コンポーネントの左側に表示されます。
テクニカル・ネーム	データマッピング時に使用される名前です。半角英数字で入力します。 マッピング時に、項目の名称として使用されます。 マッピングの詳細は「1．5マッピング」を参照して下さい。
アイテム	チェックボックス、ラジオボタン、コンボボックスの場合に使用します。 選択肢の表示名と、値をアイテム・プロパティダイアログで設定します。

※アイテム・プロパティダイアログは、以下のようにして表示します。

1.4.3.3 プールの配置

作成したXFormの業務画面を使用するアクター（処理担当者）用にプールを追加します。
（タスク等と同様に、パレットからプールを選択してダイアグラムに配置します）

追加したプールを選択してプロパティーを開き、XForm用のプールを実行不可にします。
（※ラベル、テクニカル・ネームも入力しておきます）

Column

実行可能/不可プールについて

「実行可能プール」とはBPM|Server上で処理されるタスクを配置するプールのことで、それ以外（BPM|Serverの外部）とデータのやり取りを行うタスクは「実行不可プール」に配置します。
具体的には、以下のタスクは実行不可プールに配置します。

○ヒューマンタスク　→　人がデータ処理を行うタスク
○WebServiceタスク　→　外部のシステムがデータ処理を行うタスク

また「実行可能jプール」では、一番最初に処理されるアクティビティは外部からデータを受け取る必要があり、２つのうちのどちらかのタスクからデータを受け取るようにする必要があります。

※実行不可プール内に記述された処理フローは、BPM|Serverで実行されません。外部システムや人が実施する処理を仕事を分かりやすくフロー化する目的で使用することができます。

1.4.3.4 ダイアグラムへの組み込み

プールに.xformファイルをドラッグ＆ドロップします。

■ヒューマンタスクの種類

プールにドロップすると、以下のドロップダウンメニューが表示されます。XForm画面はダイアグラムの中で、以下の４種類に分類されます。

	初期処理タスク	ダイアグラムを実行する際の、最初の画面として使用する場合に選択します。ダイアグラム中で１つだけ配置することが出来ます。初期処理タスクからは、画面でユーザが入力したデータを受け取ることが出来ます。
	未処理タスク	ダイアグラム実行途中で、人の判断を必要とする場合に使用します。 未処理タスクを選択すると、２つのタスク（createタスクとcompleteタスク）がダイアグラム上に配置されます。createタスクでは、画面に表示するデータを受け取り、completeタスクからは画面で入力されたデータが出力されます。
	通知タスク	ユーザに情報を通知する目的で使用します。通知タスクでは、画面に表示するデータを渡すことができますが、画面からデータを受け取ることはできません。そのため通知タスクでの処理結果によらず、後続の処理が実行されます。
	委譲タスク	他のヒューマンタスクから処理を委譲されることを想定して使用します。 例えば、ある未処理タスクが一定期間処理されなかった場合に、別のユーザに処理権限を委譲する等です。

■配置例

ドロップダウンメニューを選択すると、ヒューマンスクがダイアグラム上に作成されます。

1.4.3.5 メッセージフローの描画

パレットから「メッセージコネクション」を選択して、データの送信元から送信先に向かってドラッグ＆ドロップします。

■初期処理タスク画面からデータを受け取る

初期処理タスク画面からデータを送信するリクエスト・メッセージフローが描画されます。

レスポンスも同様に描画します。

- XFormとデータをやり取りする場合は、必ず送信と受信をセットで描画します

■未処理タスクとのデータ送受信

初期処理の場合と同様に、メッセージフローを描画します。但し、以下の点に注意します。

1．XForm側のcreateタスクとcompleteタスク用に、データを送受信するためのタスクを用意します。
2．メッセージフローの描画順は以下のようにします。
　①createタスクへリクエストを送信する　→　画面へ表示するデータの送信
　②createタスクからレスポンスを受信する
　③completeタスクからリクエストを送信する　→　画面で入力されたデータの受信
　④completeタスクがレスポンスを受信する

Column

ダイアグラムを見やすくする

ダイアグラムに追加したXFormタスクやXFormからのデータを受信するタスクは、アクティビティタイプを変更する等で、プロセス全体が見やすくなります。
例）

上記の例では、以下のようにしてダイアグラムを修正しています
1）初期処理タスクをメッセージ終了イベントに変換、ラベルを「見積データ送信」に修正
2）「見積データ入力」タスクをダイアグラムに追加
3）空開始イベントを「見積データ入力」タスクの前に移動、シーケンスフローを再描画
4）「見積処理」タスクをメッセージ開始イベントに変換、ラベルを「見積データ受信」に修正
5）「決裁処理-開始」、「決裁処理-完了」タスクをそれぞれ、メッセージ中間イベントに変換してラベル名を「見積データ送信」、「決裁結果受信」に修正
6）未処理（create）タスク、未処理（complete）タスクをそれぞれ、メッセージ開始イベントとメッセージ終了イベントに変更して、ラベル名を「見積データ受信」、「決裁結果送信」に修正
7）「見積データ受信」、「決裁結果送信」の間に、「決裁処理」タスクを追加

※新しく追加した「見積データ入力」タスクと「決裁処理」タスクは、実行不可プール上のタスクなので、BPM|Serverで実行されません。ここでは、人が実施する作業を分かりやすくする目的で配置しています。

1.4.3.6 メッセージのデータフォーマットについて

メッセージフローでは、XML形式でデータの送信及び受信が行われます。よって、メッセージフローを作成する場合は、やり取りされるデータについてXMLスキーマを別途用意する必要があります。

XMLスキーマファイルをメッセージフロー上にドラッグ＆ドロップすることで、そのメッセージフローのデータフォーマットの定義（関連付け）ができます。

ただし以下の場合は、特にXMLスキーマを作成することなく、フォーマット定義が行われます。

■XForm（リレーションフォーム）の場合

フォームの作成と同時に、画面レイアウト等に合わせて必要なXMLスキーマが自動生成されます。
また、メッセージフロー描画時に関連付けも自動的に行われます。

■外部WebService呼び出しの場合

WSDL内に定義されている入出力パラメータの定義に従って、メッセージフロー描画時に自動的にフォーマット定義と関連付けが行われます。
詳細は「2．３ 外部WebService連携」を参照してください。

1.5 マッピング

画面から受け取ったデータを別の画面に表示する、あるいはWebServiceに対して必要なデータを渡す、または受け取る等する必要が場合は、「マッパー」を使用してデータのマッピングを行います。

■マッピングの操作

コネクタ（矢印のアイコン）を選択して、マッピング元のデータ項目から、マッピング先のデータ項目へマウス操作で線をつなぎます。

1.5.1 データの持ち方について

BPMにおけるデータの扱い方は、従来の考え方とは大きく異なります。

■従来のワークフローモデルにおけるデータの流れ

■BPMモデルにおけるデータの流れ

従来のワークフローモデルでは、業務シーケンスフローに付随する形でデータも同時に流れるというのが一般的です。それに対してBPMモデルにおいては、業務フローとデータフロー（メッセージフロー）は完全に分離されます。データはプール単位で共有され、同一プール内であれば、いつでも自由に共有データを使用することができます。逆にメッセージフローが必要なケースは、プール間でデータをやり取りする場合のみと言えます。

1.5.2 データのマッピング

外部のプールから受け取ったデータは、メッセージフローのデータフォーマット定義（XMLスキーマ）に従って、自動的に共有データ領域へ格納され、プール内の他のタスク等で使用可能な状態になります。つまり他のプール（主に、BPM|Server→XForm、リレーションフォームまたは、WebService）へデータを送信する場合にマッピングを使用します。

次のフローでは３箇所で外部とデータのやり取りをしていますが、①と③は外部（XForm画面）からのデータ受信であり、マッピングの必要はありません。
しかし、②においてはXForm画面へ渡す値をマッピングで定義する必要があります。

例）初期処理画面から、未処理画面へのデータマッピング

初期処理タスクで、title(テキストフィールド)とnotes（テキストエリア）に入力された値がそれぞれ未処理画面へ引き継がれます。

1.5.3 XPath関数

マッパーではシンプルなデータの引継ぎ以外に、XPath関数を使用してデータの加工や判定を行うことができます。

次の例では、concat()関数を使用して文字列連結した結果をマッピングしています。

■XPath関数の配置方法

「オペレータの追加」アイコンをクリックして、マッパー内に配置、関数名を入力します。

配置後、コネクタを使用してマッピングを作成します。
このとき関数への入力は左側へ接続し、出力は右側から受け取ります。

● XPath関数は、「ウィンドウ」メニューから、「ビューの表示」→「マッパー・パレット」ビューで選択することもできます

1.5.4 分岐条件/ループ条件のマッピング

ゲートウェイの分岐条件や、ループの終了条件もマッピングで設定します。

1.5.4.1 ゲートウェイの分岐条件

ダイアグラムで分岐条件を設定するゲートウェイを選択して、マッパーを表示します。

マッパーでは条件を設定する分岐を選択後、conditionにtrueまたはfalseが渡るようにマッピングを作成します。上記の例では、=（イコール）関数を使用して、２つの値を比較した結果をマッピングしています。

Column

分岐のデフォルト選択

分岐ルートを右クリックすると、分岐の「デフォルト選択」の設定ができます。これは、他のルートの分岐条件をいずれも満たせなかった場合に選択されるルートとなります。
下の例では、YESのルートに設定されている条件が満たせなかった場合は全てNOのルートが選択されます。

1.5.4.2 ループの終了条件

ゲートウェイの場合と同様に、ループの場合も同様に終了条件を設定する「ルーピングタスク」または、「ルーピングサブプロセス」をダイアグラム上で選択します。

マッパーが表示されたら「ループ」タブを選択して、終了条件（trueまたはfalse）をwhileに対してマッピングします。

1.5.5 ビルトインデータ

マッパーでは、ユーザがメッセージフローで定義したデータ以外に使用可能なデータがあります。

1.5.5.1 プロセスインスタンスID

BPM|Serverでは、１つのダイアグラムを元に複数の処理案件（インスタンス）が作成されます。
フロー実行時に、この処理案件を一意に識別するためのIDが内部的に作成され「$ode:pid」という変数に格納されます。このIDはプロセスが開始すると同時に採番され、終了するまで同じ値が使用されます。

業務案件を外部システムで識別する必要がある場合や、相関セットキーとして使用します。
相関セットキーについては、「２．３ 外部WebService連携」を参照して下さい。

1.5.5.2 ヒューマンタスクのメタデータ

ヒューマンタスクにはいくつかのメタデータがあり、マッピング時に利用することが出来ます。

■ヒューマンタスクから受け取ることができるデータ

ヒューマンタスクが処理された後、受け取ることが可能なデータです。

@taskId	ヒューマンタスクを一意に識別するためのIDで、案件が到達したときに採番されます。ループ等の使用により、案件が複数回同じヒューマンタスクに到達した場合は、そのたびに違うIDが割り振られます。
@user	ヒューマンタスクを処理したユーザが格納されます。ログイングループIDも含めた形で渡されます。フォーマットは以下の通りです。 ログイングループID + ″¥″ + ユーザID　　例）default¥aoyagi

■ヒューマンタスクに渡すことができるデータ（taskMetaData）

ヒューマンタスクに処理が渡る前に渡すことが可能なデータです。

creationDate	ヒューマンタスクに案件が到達（依頼）した日付として指定することができます。 デフォルトでは、BPM|Serverのシステム日付がセットされます。 タスク一覧（未処理一覧、通知一覧など）に表示されます。
description	タスク一覧で、案件名等として表示されます。 この設定は、固定値の場合はプロパティービュー内の「workflow」タブ→「説明」で入力することが可能です。しかし、表示内容を動的に変更したい場合は、description に対してデータをマッピングします。分かりやすい日本語の日本語のタイトル等を設定することが推奨されます。
userOwner	ヒューマンタスクを処理可能なユーザのユーザIDを設定します。ログイングループIDも含めた形で、以下のフォーマットで指定します。 ログイングループID + ″¥″ + ユーザID　　例）default¥aoyagi 複数のユーザを指定する場合は、″,″（カンマ）で区切ります。 この設定は、ヒューマンタスクの処理権限が動的に変化する場合に使用します。 固定のユーザに権限を付与する場合は、「2．１ 権限設定」を参照してください。
roleOwner	ヒューマンタスクの処理権限として、ロール、組織、パブリックグループ等を動的に割り当てる場合に使用します。複数指定する場合は、″,″（カンマ）で区切ります。 フォーマットについては、別冊の「BPM 説明資料」を参照してください。

1.6 デプロイ

作成したプロセスをBPM|Serverにデプロイして、実行可能な状態にします。

1 デプロイ・ダイアログの起動

プロセス・エクスプローラでデプロイするダイアグラムを選択して、デプロイアイコンをクリック、または「プロジェクト」メニューから「プロジェクトを実装」を選択します。

- 同じプロセスを複数回デプロイすると、BPM|Server上では古いプロセスは実行不可能(Retire)ステータスとなり最後にデプロイされたプロセスが有効になります。(但し、実行中の古いプロセスインスタンスは古いプロセスで進行します。)
  しかし、ヒューマンタスク(初期処理タスク)を含むプロセスの場合、かつ初期処理タスクに変更が加えられた場合、新しい初期処理タスクと認識され、デプロイ前とデプロイ後の両方の初期処理タスクが表示される場合があります。

2 デプロイ・ダイアログ

BPM|Serverのアドレス（デプロイ先）を入力します。

設定後、「終了」をクリックするとデプロイが完了します。

■サーバ設定

URL	BPM|Serverのアドレスを入力します。 http://<%BPMサーバのアドレス%>:8080/ode
プロセスのデプロイ	プロセスをデプロイに含めるかどうかを設定します。（必ずチェックします）
プロセスのアーカイブ	プロセスのデプロイデータをアーカイブファイルに出力する場合はチェックします。

■ワークフロー設定

URL	BPM|Serverのアドレスを入力します。 http://<%BPMサーバのアドレス%>:8080/wds
フォームのデプロイ	プロセス内に、XFormが含まれる場合はチェックします。
フォームのアーカイブ	ヒューマンタスクのデプロイデータをアーカイブファイルに出力する場合はチェックします。

Column

デプロイ結果の確認

デプロイダイアログから「終了」をクリックし、BPM|Serverへデプロイを行うと、「進行状況」ビューにデプロイ結果が表示されます。
正常にデプロイが行われた場合、下図の様に表示されます。

デプロイに失敗した場合は、メッセージに「Intalio|BPMS Designer deploymentがエラーを返しました」と表示されます。
メッセージのリンクをクリックして頂く事により、失敗したデプロイ処理のエラーログが表示されます。

また、ビジネス・プロセス内に、xFormファイルが含まれていない場合は、デプロイ時にワークフロー設定を行う必要がありません。
その為、デプロイ後の「進行状況」ビューには、「Intalio|Workflow のデプロイは行われませんでした」と表示されますが、デプロイ処理に問題はありません。

1.7 管理コンソール

BPM|Serverにデプロイされたプロセスや、実行されたプロセスインスタンスの状態等は管理コンソールで確認することが可能です。
管理コンソールを使用するためには、iWP/iAFとBPM|Serverが起動中であることと、権限の設定が必要です。サーバのインストールと起動については、セットアップガイドを参照してください。

1.7.1 権限の設定

ログイングループ管理者でログインして、アカウント設定画面を開きます。

権限設定タブを開き、以下のロールを割り当てます。

ロール名	可能な操作
bpmadmin （BPM管理者）	全ての操作が可能
bpmprocessmanager （BPMプロセス管理者）	全プロセスとプロセスインスタンスの参照 プロセスの起動 プロセスインスタンスの再開、保留、強制終了

※詳細は「BPM説明資料」の「2．3 BPM管理画面のアクセス制御」を参照してください

- メニュー設定等で、メニュー項目に権限を追加しただけでは有効になりません。
  管理コンソールを使用するためには、必ず上記のロールをアカウント設定で割り当てる必要があります。

1.7.2 管理コンソールの表示

一般ユーザでログインして、「BPM」メニューの中から「管理」をクリックします。

■Intalioのログイン画面が表示された場合

管理コンソールでログアウトリンクをクリックする等した場合、下記の画面が表示されます。
このような場合は、×ボタンをクリックして一度ブラウザのウィンドウを閉じた上で、「管理」メニューから再度開きなおしてください。

- 「管理コンソール」は、「Intalio|BPMS Console」を利用しています。詳細は下記サイトをご参照下さい。
  http://bpms.intalio.com/reference-guides/intalio-bpms-console.html

1.7.2.1 プロセス一覧画面

BPM|Serverにデプロイされているプロセス（ダイアグラム）の一覧です。
プロセスの状態を一覧で確認することが出来ます。

Lifecycle	プロセスが実行可能（ACTIVE）か、実行不可（RETIRED）であるかを示します。
In Progress	実行中のプロセスインスタンス数です。
Failure	実行中にシステムエラーが発生したインスタンスの数です。
Suspended	サーバダウン等の理由で実行が一時中断されているインスタンスの数です。
Failed	BPMNの「エラー終了イベント」で終了したインスタンスの数です。
Terminated	BPMNの「ターミネート終了イベント」で終了したインスタンスの数です。
Completed	実行が正常終了したインスタンスの数です。

■プロセス一覧画面での操作

Start	一覧で選択したプロセスのインスタンスを作成・実行します
Activate	「RETIRED」のプロセスを「ACTIVE」にします。
Retire	「ACTIVE」のプロセスを「RETIRED」にします。
Deploy	アーカイブファイルをデプロイします。アーカイブファイルの作成については「1．6．デプロイ」を参照してください。（※ヒューマンタスクのデプロイはできません）
Undeploy	プロセスをアンデプロイします。

- プロセス名の後ろの[Vxx]は、バージョン番号です。同じプロセスを複数回デプロイすると、バージョン番号が繰り上がり、前回デプロイ時のバージョンは自動的に「RETIRED」になります。

1.7.2.2 インスタンス一覧画面

実行中または、実行されたプロセスインスタンスの一覧です。
中断されたプロセスインスタンスの再開や、強制終了及び削除ができます。

State	インスタンスの実行状態です。 In Progress, Failure, Suspended, Failed, Terminated, Completedのいずれかが表示されます。
Failures	発生したエラーの件数です。
Started	プロセスインスタンスが作成・開始された日時です。
LastActive	プロセスインスタンスが最後に実行された日時です。

■インスタンス一覧画面での操作

Invoke	プロセスインスタンスを呼び出します。
Resume	一時中断中のプロセスを手動で再開します。
Suspend	プロセスの実行を一時中断します。
Terminate	プロセスの実行を強制終了します。
Delete	一覧で選択したプロセスインスタンスを削除します。（実行中のインスタンスは削除できません）
Delete All	全てのプロセスインスタンスを削除します。（実行中のインスタンスは削除できません）

● 「プロセス一覧」画面及び「インスタンス一覧」画面の「プロセス名」リンクから、それぞれプロセスやプロセスインスタンスのダイアグラム図等の詳細を確認することができます。

Column

TMP(TaskManagementProcess)について

「TaskManager(TMP:TaskManagementProcess)」は、ヒューマンタスク（未処理）を処理するために必要なビルトインプロセスです。削除（アンデプロイ）してしまうと、ヒューマンタスクを含むプロセスの実行ができなくなります。

・TaskManagementProcess

ヒューマンタスク（未処理）に処理が到達するとTMPが実行され、StateがIn Progress（実行中）のインスタンスが生成されます。
このStateは、ユーザによってヒューマンタスクの処理が完了されるとCompleteとなります。

・TMPから生成されたインスタンスの例

intra-mart WebPlatform/AppFramework

第2章 応用編

2.1 権限設定

2.1.1 概要

BPM|Designerではダイアグラムのヒューマンタスクとプールへ、プロセス実行時のアクセス権限を設定することができます。
BPM|Serverでプロセスを実行した際に、ここで設定した権限が有効化されます。プロセス実行時は、権限を設定したヒューマンタスクまたはプールに対し、設定された権限をもつユーザ以外からアクセスすることはできません。
ダイアグラムのヒューマンタスクとプールへの権限設定は、BPM|Designerの[プロパティー] ビューの[Workflow]を使用して行います。

2.1.2 権限の種類

ヒューマンタスクとプールには、様々な権限を設定でき、実行ユーザの制限を柔軟に行うことができます。
１つのヒューマンタスク、プールに対して、複数の権限を設定することも可能です。使用できる権限は以下のようになります。

＜権限の種類＞
- ユーザ
- ロール
- 組織
- 組織 ＋ ロール
- 組織 ＋ 役職
- パブリックグループ ＋ ロール

例えば、特定の部署（組織）に所属する部長（役職）や、特定のロールを持っている人だけが実行できるような権限設定を行うことができます。

2.1.3 権限の設定

権限の設定を行う[Workflow] は、ヒューマンタスクまたはプールをクリックすることで、[プロパティー] ビューに表示されます。[Workflow]の各検索ボタンをクリックすると、各権限の検索画面が表示されます。これら検索画面から権限を選択し、権限の設定を行います。

■[プロパティー] ビューの[Workflow]

説明	ヒューマンタスクの説明を入力します。ここで入力した値は、intra-martの[BPM]-[プロセス]の一覧中の[プロセス名]、[BPM]-[未処理]、[BPM]-[処理済]、[BPM]-[通知]の一覧中の[案件名]に表示されます。 プロセス一覧、未処理一覧等の詳細については「BPM ユーザ操作ガイド」を参照して下さい。
ユーザ検索	[ユーザ検索]画面を表示し、検索を行います。
ロール検索	[ロール検索]画面を表示し、検索を行います。
組織検索	[組織検索]画面を表示し、検索を行います。

パブリックグループ検索	[パブリックグループ検索]画面を表示し、検索を行います。
削除	ヒューマンタスク、プールに設定した権限の削除を行います。
コード	検索画面で選択したコードを表示します。
名称	検索画面で選択した名称を表示します。
種別	検索画面で選択した種別を表示します。検索画面で絞り込み条件を指定した場合、検索した条件によって表示される種別が変わります。種別に表示される権限は以下のとおりです。 ＜種別の表示＞ ・（ユーザ検索の場合） ユーザ ・（ロール検索の場合） ロール ・（組織検索の場合） 組織 ・（組織検索で役職を絞り込み条件で指定した場合） 組織＋役職 ・（組織検索でロールを絞り込み条件で指定した場合） 組織＋ロール ・（パブリック検索の場合） パブリックグループ ・（パブリックグループ検索でロールを絞り込み条件で指定した場合） パブリックグループ＋ロール

権限の設定で使用する検索画面は、[ユーザ検索]画面、[ロール検索]画面、[組織検索]画面、[パブリックグループ検索]画面です。

■[ユーザ検索]画面

■ [ロール検索]画面

■[組織検索]画面

■[パブリックグループ検索]画面

2.1.3.1 ヒューマンタスクとプール

例として、以下のダイアグラムのヒューマンタスクとプールへ権限を設定します。

1 まず、ヒューマンタスクへ権限を設定します。権限を設定するヒューマンタスクを選択し、ヒューマンタスクに関連した[プロパティー]ビューを表示します。

● 権限を設定するタスクは「xForm」または、「リレーションフォーム」である必要があります。

2 [プロパティー]ビューの[Workflow]を選択します。

3 [Workflow]の[ユーザ検索]ボタンをクリックします。[ユーザ検索]画面が表示されます。

● [ユーザ検索]、[ロール検索]、[組織検索]、[パブリックグループ検索]ボタンで検索を行う前に、[認証情報入力]画面にてintra-martの認証情報を入力しておく必要があります。

・[プロジェクト]-[プロパティー]の[イントラマート]-[認証情報入力]

[認証情報入力]画面の詳細については、コラム「認証情報の設定」を参照ください

4 [ユーザ検索]画面でユーザの検索を行います。検索結果は、画面の中央部に表示されます。ここでは、[頭文字]の[あ行]を選択します。

5 検索結果から権限を与えるユーザを選択し、[選択]をクリックすると、選択したユーザが[選択一覧]に表示されます。[終了]ボタンをクリックし、検索画面を閉じます。

6 [ユーザ検索]画面を閉じると、[Workflow]に選択したユーザが反映され、ヒューマンタスクに が表示されます。(権限を複数設定した場合は、 が表示されます。)

7 次にプールへ権限を設定します。権限を設定するプールを選択し、[プロパティー]ビューを表示します。[プロパティー]ビューから[Workflow]を選択します。

- 権限を設定するプールは、「プールを実行不可能に設定」に設定されている必要があります。

・プール上で右クリックし、[プールを実行不可能に設定]

8 [Workflow]の[組織検索]ボタンをクリックします。[組織検索]画面が表示されます。

9 [組織検索]画面で組織の検索を行います。検索結果は、画面の左部に表示されます。ここでは、[キーワード]に「comp」を入力し、検索を実行しています。

10 検索結果から権限を与える組織を選択し、[選択]をクリックすると、選択した組織が[選択一覧]に表示されます。[終了]ボタンをクリックし、検索画面を閉じます。

11 [組織検索]画面を閉じると、[Workflow]に選択した組織が反映され、プールに が表示されます。(権限を複数設定した場合は、 が表示されます。)最後に、 [保管]をクリックし、設定内容を保存します。

2.1.3.2 権限設定時の注意点

ここでは、権限を設定する際の注意点について説明します。以降の説明においては、iWP/iAFにサンプルデータがインポートされていることを前提とします。サンプルデータのインポートの詳細については、別冊「intra-mart WebPlatfrom／AppFrameworkーインストールガイド」の7章を参照ください。

■ 権限の優先順

ヒューマンタスクとプールの両方に権限を設定した場合、プロセス実行時、ヒューマンタスクに設定した権限が優先されます。また、プールに権限を設定し、ヒューマンタスクに権限の設定をしない場合は、プールに設定した権限が有効になります。

例）

このダイアグラムのプールとヒューマンタスクに、以下の権限の設定を行ったとします。

＜権限の設定＞

・プール　・・・（ロール）level3

コード	名称	種別
level3	level3	ロール

・ヒューマンタスクA　・・・指定なし

コード	名称	種別

・ヒューマンタスクB　・・・（ロール）level2

コード	名称	種別
level2	level2	ロール

ダイアグラムをBPM|Serverへデプロイし、プロセスを実行した場合、ヒューマンタスクA、Bに付与される権限は、以下のようになります。

＜プロセス実行時に付与される権限＞

ヒューマンタスクA	（ロール）level3	←プールの権限を付与
ヒューマンタスクB	（ロール）level2	←ヒューマンタスクBの権限を付与

→　ヒューマンタスクAは、ロール「level3」以上の権限を持ったユーザで実行することができます。
→　ヒューマンタスクBは、ロール「level2」以上の権限を持ったユーザで実行することができます。

■ 複数権限設定

１つのヒューマンタスク、プールに、複数の権限を設定した場合、プロセス実行時、設定した権限はOR条件で実行されます。

例）

このダイアグラムのヒューマンタスクAに、以下の権限の設定を行ったとします。

＜権限の設定＞

コード	名称	種別
hagimoto	萩本順子	ユーザ
comp_sample_01	サンプル会社	組織

ダイアグラムをBPM|Serverへデプロイし、プロセスを実行した場合、ヒューマンタスクAに付与される権限は以下のようになります。

＜プロセス実行時に付与される権限＞

ヒューマンタスクA	（ユーザ）hagimoto OR （組織）comp_sample_01

→　ヒューマンタスクAは、「hagimoto」または「comp_sample_01」に所属するユーザで実行することができます。

Column

認証情報の設定

[Workflow]でユーザ等の検索を行う際、事前にintra-martへのログイン認証情報を設定しておく必要があります。
設定した認証情報は、BPMプロジェクト単位で保持されます。そのためBPMプロジェクトを新規作成した場合は、必ず認証情報の設定が必要となります。

＜認証情報の設定手順＞
1：BPM|Designerのメニュー[プロジェクト]より[プロパティー]を選択し、[プロパティー]画面を開きます。

2：[プロパティー]画面から、[イントラマート]-[認証情報入力]を選択し、[認証情報入力]画面を開き、必要な情報を入力します。

[認証情報入力]画面

ID （必須）	imtra-martのユーザIDを入力します。
Password	imtra-martのパスワードを入力します。
GroupID （必須）	imtra-martのログイングループIDを入力します。
接続先ホスト （必須）	imtra-martのURLを入力します。URLはコンテキストパスまでを指定します。
パスワード保管	パスワード保管の有無を設定します。
ログイン	imtra-martへのログインを行います。
ログアウト	imtra-martからのログアウトを行います。

3. ［ログイン］ボタンをクリックし、intra-martへのログインが成功すると、画面上部にメッセージ「ログインしました。」が表示されます。［OK］ボタンをクリックして、［認証情報入力］画面を閉じます。

Column

検索基準日の設定

[Workflow]で権限の設定を行う際、期間化を伴う情報（ユーザ・組織・パブリックグループ）を検索する場合には、デフォルトではiWP/iAFインストールマシンのシステム日付で検索を行いますが、任意の検索基準日を設定することができます。
設定した検索基準日は、BPMプロジェクト単位で保持されます。そのためBPMプロジェクトを新規作成した場合は、必要に応じて検索基準日の設定を行ってください。

＜検索基準日の設定手順＞
1：BPM|Designerのメニュー[プロジェクト]より[プロパティー]を選択し、[プロパティー]画面を開きます。

2：[プロパティー]画面から、[イントラマート]-[検索基準日]を選択し、[検索基準日]画面を開きます。検索基準日のラジオボタンにチェックをし、表示されたカレンダーボックスの年・月・日それぞれに指定したい年月日を設定をします。

[検索基準日]画面

2.2 リレーション・フォーム

2.2.1 概要

リレーション・フォームは、ビジネス・プロセス・ダイアグラム中に、ヒューマンタスクを配置する為のフォームです。

intra-mart画面との連携に必要な設定をGUIツールにより自動的に作成する事が可能です。

2.2.2 リレーション・フォームの作成

リレーション・フォームを作成する為の操作手順を以下に示します。

1 [ファイル(F)] → [新規(N)] → [intra-mart フォーム・リレーション エディタ]を選択します。

- リレーション・フォームを作成する場合、事前にビジネス・プロセス・プロジェクトを作成する必要があります。

2 リレーション・フォーム作成ウィザードが表示されます。
ファイル名を入力し、終了を押下します。

3 リレーション・フォームファイルが作成され、リレーション・フォーム エディタが開かれます。

2.2.3 フォーム・リレーション エディタ

フォーム・リレーション エディタは、ヒューマンタスクの配置に必要な設定を行う為のエディタです。iWP/iAFと連携する為に必要な設定、及びヒューマンタスクとその他のタスクとの連携に必要な設定を行います。

画面名	ヒューマンタスクとして割り当てるタスク名を設定します。
ページ種別	intra-mart画面における開発モデルを設定します。
パス	intra-mart画面のURLを設定します。
フォーム名前空間	ヒューマンタスクに割り当てられる名前空間が表示されます。 名前空間は自動的に設定される為、入力を行う必要はありません。
入力値	ヒューマンタスクに受け渡される入力値を設定します。
出力値	ヒューマンタスクから受け渡す出力値を設定します。
追加(属性)	入力値/出力値のデータ構造にXML属性を追加します。
追加(要素)	入力値/出力値のデータ構造にXML要素を追加します。
削除	選択された属性/要素を削除します。
モデル生成	入力値/出力値に対応したソースコードを出力する為のウィザードを表示します。

2.2.3.1 連携画面の設定

リレーション・フォームでは、設定により様々な開発モデルで作成された画面との連携を行う事が可能です。
連携を行う画面の開発モデルによりページ種別項目、パス項目の設定方法が異なりますのでご注意下さい。
設定例を下記に記します。

■JavaEE開発モデル

ページ種別	JavaEE開発モデル
パス	[アプリケーションID]-[サービスID].service 例: SampleApplication-SampleService.service

■スクリプト開発モデル

ページ種別	Script開発モデル
パス	[画面パス].jssp 例: sample/list/sampleList.jssp

■その他開発モデル

JavaEE開発モデル、スクリプト開発モデル以外の開発モデルにより作成された画面との連携を行う場合に指定します。
主に、Servlet、JSP、Struts等で作成された画面を指定する場合に利用します。

ページ種別	Servlet/JSP等
パス	[開発モデルで指定されているURL指定方法] 例: Strutsの場合: sample/sampleService.do Servletの場合: sample/SampleServlet JSPの場合: sample/sample.jsp

■外部ページ

iWP/iAF以外の外部ページを利用する場合に指定します。
外部ページにはcgi, php, ruby等の言語によるページとの連携を行う事が可能です。

ページ種別	その他
パス	[外部ページURL] 例: http://web.intra-mart/service.php

- 外部ページを利用した場合、言語に応じてプロセス処理用のAPIを実装する必要があります。

2.2.3.2 入力値/出力値の設定

入力値/出力値設定では、ヒューマンタスクと、その他のタスクとの連携を行う際に必要となるデータ構造の設定を行います。
タスク間のデータは全てXML形式で受け渡されます。入力値/出力値設定では、XMLデータ構造の設定を行います。

■Root要素

入力値/出力値XML構造の最上位の要素を示します。
要素の名称は必ずinput/outputが指定されます。
Root要素を選択、またはダブルクリックする事により[プロパティー]ビューに詳細が表示されます。

クラス名	モデル出力により、ソースコードを出力する際に利用される型名を設定します。

■属性

XML属性を示します。
属性を追加するには、ツリーから属性の所属する親要素を選択し、追加ボタンを押下します。
ツリー上の属性を選択、またはダブルクリックする事により[プロパティー]ビューに詳細が表示されます。

デフォルト値	属性値が指定されなかった場合のデフォルト値を設定します。
種類	属性値の型を設定します。
名前	属性の名称を設定します。

- 属性値を設定した場合、属性の所属する要素(タグ)内にはテキストノードを格納する事は出来ません。

■要素

XML要素を示します。
要素を追加するには、ツリーから要素の所属する親要素を選択し、追加ボタンを押下します。
ツリー上の要素を選択、またはダブルクリックする事により[プロパティー]ビューに詳細が表示されます。

デフォルト値	要素値が指定されなかった場合のデフォルト値を設定します。 子要素、属性が存在する場合は設定できません。
最小要素数	要素が出現する最小数を設定します。
最大要素数	要素が出現する最大数を設定します。 上限を設定しない場合は*を設定してください。
種類	属性値の型を設定します。 子要素、属性が存在する場合は設定できません。
名前	属性の名称を設定します。

■要素、属性の削除

要素、属性を削除するには、削除を行いたい要素、属性を選択後、削除ボタンを押下します。
Root要素を削除する事は出来ません。

■予約属性

出力値には予約属性が存在します。
この予約属性はシステム上で利用される為変更、削除を行う事は出来ません。
また、予約属性にはイントラマート固有の情報が設定される為、他のタスクとのデータ連携を行う際に利用する事が可能です。

taskId	ヒューマンタスク固有のIDが設定されます。
participantToken	システムが利用する認証情報が設定されます。 通常、この属性を利用する必要はありません。
user	システムが利用するユーザコードが設定されます。 グループID¥ユーザID形式で設定されます。
formUrl	システムが利用するフォームURLが設定されます。 通常、この属性を利用する必要はありません。
imUser	ヒューマンタスクの処理を行ったintra-martユーザIDが設定されます。
imGroup	ヒューマンタスクの処理を行ったintra-martグループIDが設定されます。
imId	ヒューマンタスクの処理を行った後、一意のIDが自動的に採番され設定されます。

Column

要素、属性の種類

要素、属性に指定可能な種類は下記の種類が指定可能です。

string: 文字列
dateTime: 日付+時刻型（例：9999-12-31T23:59:59.999）
boolean: 真偽（true、false）
int: 整数型（−2147483648～2147483647）
long: 長整数型（−9223372036854775808～9223372036854775807）
short: 短整数型（−32768～32767）
double: 64bit倍精度浮動小数点
float: 32bit単精度浮動小数点
integer: 整数型（−2147483648～2147483647）
decimal: 10進数
date: 日付型（例: 9999-12-31）
time: 時刻型（例：23:59:59.999）

それぞれの種類はW3Cにより標準化されたXMLSchemaに準拠しています。
XMLSchemaに関する詳細な情報は下記URLをご参考下さい。
http://www.w3.org/TR/xmlschema-2/

2.2.4 モデル生成

フォーム・リレーション エディタでは、JavaEE開発モデル、スクリプト開発モデル向けに入力値/出力値に対応したソースコードを出力する機能があります。
モデルを生成する事により、ヒューマンタスクに関するプログラミングを大幅に低減することが可能です。

2.2.4.1 モデル生成ウィザード

フォーム・リレーション エディタ右下部にあるモデル生成ボタンを押下することによりモデル生成ウィザードが表示されます。

1 言語選択

出力を行いたい言語を選択します。

JavaEE開発モデル	Java言語によるソースコードを出力します。
スクリプト開発モデル	JavaScript言語によるソースコードを出力します。

2 出力先選択

ソースコードを出力するフォルダを選択します。

3 出力パッケージ入力

1.言語選択において、JavaEE開発モデルを選択した場合、出力するJavaパッケージ名を設定します。

4 ソースコード出力

終了ボタンを押下する事により、指定したフォルダに入力値/出力値に対応したソースコードが出力されます。

Column

出力されるソースコードの仕様

モデル生成ウィザードにより生成されたソースコードは言語固有の仕様に沿ったソースコードが出力されます。
生成されたソースコードの利用方法に関しては、プログラミングガイド(xxxxx)をご参考ください。

<Java言語>
JSR-222 (JAXB: Java Architecture for XML Binding 2.0)に準拠したソースコートが出力されます。
JAXBに関する詳細な情報は下記URLをご参考下さい。
https://jaxb.dev.java.net/

<JavaScript言語>
ECMAにより標準化されたECMAScript for XML (E4X)を利用したソースコードが出力されます。
E4Xに関する詳細な情報は下記URLをご参考下さい。
http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-357.pdf

2.2.5 ヒューマンタスクの配置

作成したリレーション・フォームは、ビジネス・プロセス・ダイアグラムに配置する事によりヒューマンタスクが作成されます。
ヒューマンタスクの配置方法は、xFormsを利用した場合と同様の操作で配置を行う事が可能です。
詳細は、「1.4ヒューマンタスク」を参照してください。

2.3 外部WebService連携

この章では、WebServiceとの連携方法を紹介します。

2.3.1 WebService連携とは

ビジネス・プロセス・ダイアグラム中にWebService連携用のタスクを配置する事により、ビジネス・プロセス実行時に、WebServiceの呼出を行い連携する機能です。

2.3.2 WebServiceの取込

ビジネス・プロセス・ダイアグラム中にWebServiceを配置する為には、ビジネス・プロセス・プロジェクトに対し、WSDLファイルのインポートを行う必要があります。
本節では、WebServiceが既に配置済みである事を前提として説明します。
intra-mart上でのWebService配置方法に関しては「Web サービス プログラミングガイド」を参照してください。
また、ビジネス・プロセス・プロジェクトは作成済みの状態を前提として説明します。
ビジネス・プロセス・プロジェクトの作成は「1.2プロジェクトの作成」を参照し、事前にプロジェクトの作成を行ってください。

2.3.2.1 WSDLファイルのインポート

WSDLファイルのインポート手順を以下に示します。

1 メニュー[ファイル(F)] → [インポート(I)]を選択します。

2 インポートウィザードダイアログが表示されます。

[Intalio|BPMS] → [リモートサービスのインポート]を選択し、[次へ(N)]を押下します。

3 次にWSDLファイルのロケーションを入力します。

WSDLファイルのURLを入力し、[次へ(N)]を押下します。

- WSDLファイルが既にローカルファイルシステム上に存在する場合は、画面右部[参照...]ボタンを押下しローカルファイルシステム上に存在するWSDLファイルを指定します。

4 WSDLファイルが正常に読み込まれた場合、下記の画面が表示されます。
WSDLファイルの配置先を指定し、[終了(F)]を押下しWSDLファイルのインポートを完了します。

リモートサービス URL	WSDLファイルのURLが表示されます。 複数のWSDLファイルを読み込んだ場合、インポートを行うWSDLファイルを選択します。
相対ローカルファイル名	プロジェクトへインポートする際のパスを設定します。
全て選択	WSDLファイルを全て選択します。
選択を全て解除	WSDLファイルを全て選択解除します。
インポート先フォルダ	WSDLファイルをインポートするフォルダを指定します。 インポート先のプロジェクトを指定してください。
ダウンロードするリソースにインデントとフォーマットを有効にする	WSDLファイルをインポートする際に、WSDLファイルの内容を整形する場合に指定します。
Local Resource	プロジェクトに既にWSDLファイルが格納されている場合、WSDLファイルの内容が表示されます。
リモートリソース	インポートを行うWSDLファイルの内容が表示されます。

5 WSDLファイルがインポートされました。
インポートされたWSDLファイルは、[プロセス・エクスプローラ]ビュー内に表示されます。

2.3.2.2 WSDLファイルの参照

プロセス・エクスプローラビューからWSDLファイル構造の参照が行えます。

名前空間	WSDL内に定義されている名前空間が表示されます。
サービス	wsdl:serviceで定義されたサービスが表示されます。
ポート	wsdl:portで定義されたポートが表示されます。
オペレーション	wsdl:operationで定義されたオペレーションが表示されます。
ポートタイプ	wsdl:potTypeで定義されたポートタイプが表示されます。
バインディング	wsdl:bindingで定義されたバインディングが表示されます。
要素	wsdl:typesで定義された要素が表示されます。
タイプ	WSDL内に定義されている型が表示されます。

Column

WebServiceの検索

BPM|DesignerにはWebServiceを検索する為のWeb サービス・エクスプローラが含まれています。
Web サービス・エクスプローラを利用すると下記の方法を用いてWebServiceを検索する事が可能となります。
・WSDLファイルの直接参照
・WS-Inspection(WSIL)によるWebServiceの検索
・Service Repository(UDDI)によりWebServiceの検索
また、Web サービス・エクスプローラ上から直接Webサービスを実行する事も可能です。

Web サービス・エクスプローラは、メニュー[Run] → [Launch the Web Service Explorer]から利用する事が可能です。
Web サービス・エクスプローラの詳細は下記のURLをご参考下さい。
http://www.eclipse.org/webtools/jst/components/ws/1.0/tutorials/WebServiceExplorer/WebServiceExplorer.html

2.3.3 WebService連携用タスクの配置

取込を行ったWebServiceをビジネス・プロセス・ダイアグラム上に配置する方法を示します。

2.3.3.1 配置可能なWebService連携用タスクの種類

WebServiceを配置する場合、取込んだWSDLファイルから、配置したいオペレーションをビジネス・プロセス・ダイアグラム図上へドラッグ&ドロップし、配置を行います。
また、WebServiceの配置方法はプールの種別(実行可能プール、実行不可プール)により異なる場合があります。
また、WebServiceは単一のタスクとして配置する事や、複数のタスクとして配置を行う事が可能です。
WebServiceの配置方法が異なる場合でも、実際にWebServiceが呼出される際の処理方法は全て同一となります。その為、ビジネス・プロセス・ダイアグラム図に記述する業務フロー図に合わせ、適切な配置方法を選択する事が可能となります。

■ 実行不可プール上にWebServiceを配置

実行不可プール上にタスクを配置(単一タスク)	実行不可プール上にWebService連携用タスクを作成する場合に利用します。 作成されるWebService連携用タスクは単一のタスクが作成されます。
実行不可プール上にタスクを配置(複数タスク)	実行不可プール上にWebService連携用タスクを作成する場合に利用します。 作成されるWebService連携用タスクは受信用、送信用のタスクが作成されます。 また、WebServiceが例外を通知する可能性がある場合、例外処理用に分岐された例外タスクが配置されます。

■ 実行可能プール上にWebServiceを配置

実行可能プールへ配置	実行可能プール上にWebService連携用タスクを作成する場合に利用します。
実行可能プールへ配置(例外補足付き)	実行可能プールへWebService連携用タスクを作成する場合に利用します。 WebServiceが例外を通知する可能性がある場合、例外処理用のサブプロセスタスクが作成されます。 例外処理に関しては後述する[WebService連携時の例外処理]を参照してください。

■ タスクに配置

実行不可プール上のタスクに配置	実行不可プール上に用意された空のタスクをWebService連携用タスクに変更する場合に利用します。 変更されたタスクは実行不可プール上にタスクを配置した場合に作成されるWebService連携用タスクと同一のタスクとなります。
実行可能プール上のタスクに配置	実行可能プール上に用意された空のタスクをWebService連携用タスクに変更する場合に利用します。 変更されたタスクは実行可能プール上にタスクを配置した場合に作成されるWebService連携用タスクと同一のタスクとなります。

2.3.3.2 実行不可プール上にWebService連携用タスクを配置する(単一タスク)

本項では、実行不可プール上にWebService連携用のタスクを作成する手順を示します。
本項で説明する方法では、実行不可プール上にWebServiceを表現する単一のタスクが作成されます。
この方法はWebService連携用タスクを配置する際最もシンプルな表現となります。

1 **WebServiceを呼出す為のプール、タスクを作成します。**
プール、タスクの作成方法は「1.3ダイアグラムの作成」等を参照してください。

2 インポートを行ったWSDLファイルから、連携を行いたいオペレーションを実行不可プールへドラッグします。

3 オペレーションをドロップするとコンテキストメニューが表示されます。
コンテキストメニュー先頭の[サービス *** のポート *** にバインドするオペレーション *** を提供(単一タスク)を選択し、WebServiceタスクを配置します。

4 WebService連携用タスクが作成されます。

5 実行プールに配置されているWebService呼出し用タスクと、WebService連携用タスクをメッセージコネクションで接続します。

メッセージコネクションの接続順序は、WebService呼出し用タスク → WebService連携用タスクを先に接続し、

その後、WebService連携用タスク → WebService呼出し用タスクの順に接続します。

2.3.3.3 実行不可プールにWebServiceタスクを配置する(複数タスク)

本項では、実行不可プール上にWebService連携用のタスクを作成する手順を示します。
前述した単一タスクを配置する方法と違い、実行不可プール上に受信、送信を表現するタスクが生成される為、それぞれのタスクを組み合わせる事によりサービスの内容をビジネス・プロセス・ダイアグラム図上に記述する事が可能となります。

1 単一タスクとしてWebServiceタスクを配置する場合と同様にWebServiceを呼出す為のプール、タスクを作成します。

2 インポートを行ったWSDLファイルから、連携を行いたいオペレーションを実行不可プールへドラッグします。

3 オペレーションをドロップするとコンテキストメニューが表示されます。
コンテキストメニュー3番目の[サービス *** のポート *** にバインドするオペレーション *** を提供]を選択し、WebServiceタスクを配置します。

4 WebServiceタスクが配置されました。

[オペレーション名]_receiveタスクは、WebServiceの呼出を表現するタスクです。
[オペレーション名]_replyタスクは、WebServiceの応答を表現するタスクとなります。

5 実行プールに配置されているタスクと、WebServiceタスクをメッセージコネクションで接続します。

メッセージコネクションは、送信用メッセージコネクションとして呼出し用のタスク → [オペレーション名]_receiveタスクを接続します。
受信用メッセージコネクションとして、[オペレーション名]_replyタスク → 呼出し用タスクの順に接続します。

2.3.3.4 実行可能プールにWebServiceタスクを配置する

本項では実行可能プール上にWebService連携用のタスクを作成する手順を示します。
実行プール上にWebService連携用のタスクを配置した場合、WebServiceとの連携を行う為に必要な情報が全て実行可能プール上に保持される為、実行不可プール上は自由にアクティビティの配置が行えます。

1 実行プールにWebServiceタスクを配置する場合も、実行プール、実行不可プール、タスクをそれぞれを作成します。

2 インポートを行ったWSDLファイルから、連携を行いたいオペレーションを実行プールへドラッグします。

3 オペレーションをドロップするとコンテキストメニューが表示されます。
コンテキストメニュー2番目の[サービス *** のポート *** にバインドするオペレーション *** を起動]を選択し、WebServiceタスクを配置します。

4 WebServiceタスクが実行プールに配置されました。
フロー図にWebServiceタスクの前後をフローコネクタで接続します。

5 実行プールに配置されたWebServiceタスクから非実行プールに対しメッセージコネクションを接続します。
非実行プールはBPM|Server上で実行されない為、自由にビジネス・プロセス・ダイアグラム図の記述を行う事が可能です。

2.3.3.5 実行不可プール上のタスクに配置する

本項では、実行不可プール上のタスクをWebService連携用タスクに変更する為の手順を示します。
本手順は、既にビジネス・プロセス・ダイアグラム図が記述されている場合に利用します。
実行不可プール上のタスクをWebService連携用のタスクに変更した場合、前述の実行不可プール上にWebService連携用タスクを作成した場合と同様の形式となります。

1 **WebServiceを表現するタスクを含めたプール、タスクを作成します。**
プール、タスクの作成方法に関しては「1.3ダイアグラムの作成」を参照してください。

2 インポートを行ったWSDLファイルから、連携を行いたいオペレーションを実行不可プール上にあるWebServiceを表現するタスクへドラッグします。

3 オペレーションをドロップするとコンテキストメニューが表示されます。
コンテキストメニュー下から2番目の[サービス *** のポート *** にバインドするオペレーション ***を提供]を選択し、WebServiceタスクを配置します。

4 実行不可プールにあるタスクがWebServiceタスクに設定されました。

2.3.3.6 実行可能プール上のタスクに配置する

本項では、実行可能プール上のタスクをWebService連携用タスクに変更する為の手順を示します。
本手順は、既にビジネス・プロセス・ダイアグラム図が記述されている場合に利用します。
実行可能プール上のタスクをWebService連携用のタスクに変更した場合、前述の実行可能プール上にWebService連携用タスクを作成した場合と同様の形式となります。

1 **WebServiceを表現するタスクを含めたプール、タスクを作成します。**
プール、タスクの作成方法に関しては「1.3ダイアグラムの作成」を参照してください。

2 インポートを行ったWSDLファイルから、連携を行いたいオペレーションを実行可能プール上にあるWebServiceを表現するタスクへドラッグします。

3 オペレーションをドロップするとコンテキストメニューが表示されます。
コンテキストメニュー最下部の[サービス *** のポート *** にバインドするオペレーション ***を起動]を選択し、WebServiceタスクを配置します。

4 実行可能プール上にあるタスクがWebServiceタスクに設定されました。

2.3.4 WebService呼出時のパラメータ設定

WebServiceを呼出し時パラメータを設定するには、[マッパー]ビューを利用します。
実行可能プールに配置されたWebServiceと連携を行っているタスクを選択する事により、[マッパー]ビュー内でWebService呼出し時のパラメータのマッピングが行えます。
[マッパー]ビュー右部にWebService呼出時のパラメータが表示されます。
マッピングに関しては「1.5マッピング」を参照してください。

2.3.5 WebService連携時の例外を補足する

WebServiceには例外を通知する可能性のあるオペレーションが存在します。
ダイアグラムを記述する際に、WebService連携時の例外を補足する為の方法を示します。

1 実行可能プール上にあるWebService呼出用のタスクを選択し、ツールバー内にある[グループシェープ]ボタンを押下します。

2 選択したタスクを囲むサブプロセスが作成されました。

3 作成されたサブプロセスを選択し、[プロパティー]ビューの[Fault on failure]項目をtrueに変更します。

- [Fault on failure]をtrueに設定しない場合、WebService呼出を行った際に例外が通知されるとBPM|Serverはエラーと判断し実行中のプロセスを終了します。

4 次に、[Palette]ビューからエラー中間イベントを選択し、サブプロセス下部にエラー中間イベントを配置します。

5 作成したエラー中間イベントを選択し、[プロパティー]ビューの[Catch]項目にWebServiceが通知する可能性のある例外を選択します。

- 複数の例外を通知する可能性のあるWebServiceの場合、手順4～5を繰り返し、エラー中間イベントを複数配置します。

6 ビジネス・プロセス・ダイアグラム上に、例外処理用のサブプロセスを作成し、エラー中間イベントと接続します。

例外処理用のサブプロセス内に例外が通知された場合のフローを記述します。

Column

例外補足用サブプロセスの動作

例外処理用のサブプロセスは、サブプロセス内の処理が完了した場合、例外発生元のサブプロセスを抜け、次のタスクへと遷移します。
例外処理用サブプロセス内でプロセスを終了させたい場合は、例外処理用のサブプロセス内にターミネート終了イベント、またはエラー終了イベント等の終了イベントを配置し、プロセスを終了させてください。
但し、サブプロセス内に空終了イベントを配置した場合は、例外処理用サブプロセスを抜け次のタスクへと遷移します。

2.3.6 イントラマートへのログイン

本項ではintra-mart WebService認証モジュールを利用し作成されたWebServiceとの連携方法を説明します。

intra-mart WebService認証モジュールに関しては「Web サービス プログラミングガイド」を参照してください。

intra-mart認証モジュールが組み込まれたWebServiceとの連携を行う場合、認証用ダイジェストが必要となります。

BPM|Designerでは、[マッパー]ビューを利用し、データのマッピングを行う際に拡張xpath関数を利用し、intra-mart認証モジュール連携用の認証ダイジェストを生成する事が可能です。

本項では、拡張xpath関数を利用し、認証ダイジェストを生成する方法を記します。

2.3.6.1 xpathファイルのインポート

1 xpathファイルのインポート取り込みを行います。
ファイルメニューから、[インポート(I)...]を選択します。

- BPM|Designerでは、標準のXPath関数以外に、独自のXPath関数を定義する事が可能です。XPath関数の定義方法は後述の「3.1.5カスタムXPath関数の作成」を参照してください。

2 インポート・ソースの選択を行います。

インポート・ソースツリーから[一般] → [ファイル・システム]を選択し画面下部にある[次へ >(N)]ボタンを押下します。

3 インポートするxpathファイルを選択します。

画面上部にある[ソース・ディレクトリー(Y)]項目にBPM|Designerインストールフォルダ/etc/functionsフォルダを選択します。

フォルダ選択後、画面右部にxpathファイルが表示されます、imart.xpathファイルを選択し画面下部[終了(F)]ボタンを押下します。

4 imart.xpathファイルがインポートされた事を確認します。

5 インポートしたimart.xpathファイルを[データエディター]ビュー右部に表示されている実行プロセスにドラッグ&ドロップしxpath関数の登録を行います。

6 [データ・エディター]ビュー右部にnamespace: xmlns:imart="jp.co.intra_mart…"項目が追加されたことを確認します。

- [データ・エディター]を介してxpathファイルをダイアグラム上に登録する事により拡張XPath関数を利用する事が可能となります。

7 次に、BPM|Designer上部にあるメニュー[ウィンドウ(W)] → [ビューの表示(V)] → [マッパーパレット]を選択し、[マッパーパレット]ビューを表示します。
[マッパーパレット]下部[カスタムXPathファンクション]内に先ほど登録したXPath関数が表示されます。

- [マッパー]ビューには、利用可能なXPath関数が表示されます。利用する関数を[マッパー]ビュー中央部にドラッグ&ドロップする事によりXPath関数を配置する事が可能です。
- [カスタムXPathファンクション]項目は、XPath関数を登録したプールをダイアグラム上で選択している場合にのみ表示されます。

8 [マッパー]ビューを表示し、実行プール上にあるWebService連携を行っているタスクを選択します。

intra-mart WebService認証モジュールを利用したWebServiceの場合、[マッパー]ビュー右部のWebServiceパラメータにwsUserInfo項目が表示されます。

9 [マッパー]ビューを右部に表示されるWebServiceパラメータの中にあるwsUserInfo項目に値のマッピングを行います。
マッピング方法に関しては「1.5マッピング」を参照してください。

BPMプロセスからWebService認証モジュール用にマッピングする値を下記に示します。

authType (認証種別)	認証種別を設定します。 "BPMS"を指定します。
loginGroupID	ログイングループを指定する項目です。 ログイングループはパスワードダイジェストに含まれる為、指定する必要はありません。
password	パスワードダイジェストを指定してます。 パスワードダイジェストは後述するimart:getDigest関数で生成します。
userID	ユーザIDを指定する項目です。 ユーザIDはパスワードダイジェストに含まれる為、指定する必要はありません。

10 wsUserInfo/authTypeに対して文字リテラル"BPMS"をマッピングします。

11 [マッパーパレット]ビュー → [17.カスタムXPathファンクション] → [intra-mart custom functions] → [1.imart:getDigest]を選択し、[マッパー]ビュー中央部にドラッグ&ドロップし、パスワードダイジェスト生成用関数 imart:getDigestを配置します。

- imart:getDigest関数は2種類の引数による呼出が可能です。
- groupId, userId, password, separatorの4つの引数を指定する事が可能です。
- groupIdAndUserId, passwordの2つの引数を指定する事が可能です。groupIdAndUserId項目は"グループID¥ユーザID"形式で指定してください。

12 配置したimart:getDigest関数に引数として"グループID¥ユーザID"及び"パスワード"を連結し、戻り値をwsUserInfo/passowrd項目に連結します。これによりWebService呼出時にダイジェスト化された認証情報が送信されるようになります。

- 本項では、文字リテラルとしてアカウント/パスワードをimart:getDigest関数に引数として連結していまが[マッパー]ビュー左部に表示されるパラメータとimage:getDigestを連結する事により動的に引数を指定する事が可能となります。
- intra-mart WebService認証モジュールにより認証が失敗した場合、呼出を行ったWebServiceは例外を通知します。例外処理方法は前述の[**WebService連携時の例外処理**]を参照してください。

2.4 プロセスのWebService化

2.4.1 概要

ビジネス・プロセス・ダイアグラムに記述した実行プールはデプロイ後、BPMServerによりWebService化されます。
WebServiceとして提供されたプロセスは、外部から呼出を行う事が可能となります。

2.4.2 WebService化手順

本項ではビジネス・プロセスをWebServiceとして作成する手順を示します。
事前にビジネス・プロセス・プロジェクトを作成して下さい。
ビジネス・プロセス・プロジェクトの作成方法は「1.2プロジェクトの作成」を参照してください。
WebServiceとして呼出を行う際に必要な引数、戻り値の型はXMLスキーマにより定義を行います。
事前にXMLスキーマの作成を行ってください。

1 実行可能プールと実行不可プールを作成します。

- プール、タスク、イベント等に日本語名を使用する場合、[プロパティー]ビューからテクニカル・ネームを指定してください。

2 実行可能プール上にメッセージ開始イベント、メッセージ終了イベントを配置します。

3 実行不可プール上にプロセス呼出用タスクを配置し、メッセージ開始イベント、メッセージ終了イベントに対しそれぞれメッセージコネクションを接続します。

4 プロセス呼出時の引数パラメータ型を設定します。プロセス・エクスプローラ上のXMLスキーマファイルを展開し、メッセージ開始イベントとプロセス呼出用タスクを接続しているメッセージコネクションへドラッグ&ドロップします。

5 メッセージコネクション上にXMLスキーマ型をドロップする事によりコンテキストメニューが表示されます。[スキーマ・エレメント[XMLスキーマ型名称]をメッセージのコンテンツとしてセットする。]を選択し、パラメータ型を割当てます。

6 メッセージコネクションにパラメータ型が割当てられました。

7 プロセス完了時の戻り値パラメータ型を設定します。引数パラメータ型設定時と同様にプロセス・エクスプローラ上のXMLスキーマファイルを展開し、メッセージ開始イベントとプロセス呼出用タスクを接続しているメッセージコネクションへドラッグ&ドロップし、パラメータ型を割当てます。

パラメータ割当て後、デプロイを行う事によりWebServiceとしてプロセスを呼び出す事が可能となります。

2.4.3 WebServiceプロセスのWSDL

本項ではWebServiceとしてデプロイされたプロセスのWSDLファイルの取得方法を示します。

2.4.3.1 WSDLファイルパスの命名規則

デプロイが行われたプロセスは、ビジネス・プロセス・ダイアグラム図中に記述された内容を元にWSDLファイルパス、及びWebServiceエンドポイントが決定されます。

WSDLファイルパス、及びWebServiceエンドポイントの決定に利用される値

プロジェクト名	ビジネス・プロセス・プロジェクト名
ダイアグラム名	ビジネス・プロセス・ダイアグラムファイル(.bpm)ファイル名
実行不可プール名	実行プールとメッセージコネクションにより接続されている実行不可プールの名称 実行不可プールのプロパティ項目[テクニカル・ネーム]が指定されている場合[テクニカル・ネーム]値が優先されます。
実行プール名	実行プールの名称 実行プールのプロパティ項目[テクニカル・ネーム]が指定されている場合[テクニカル・ネーム]値が優先されます。

WSDLファイルパス
http://%BPMサーバホスト名%:%BPMサーバポート%/ode/processes/%プロジェクト名%/%ダイアグラム名%/%実行プール名%/%実行不可プール名%?wsdl

WebServiceエンドポイント
http://%BPMサーバホスト名%:%BPMサーバポート%/ode/processes/%プロジェクト名%/%ダイアグラム名%/%実行プール名%/%実行不可プール名%

- WebServiceとしてデプロイされたプロセスの一覧は下記URLから参照する事が可能です。
  http://%BPMサーバホスト名%:BPMサーバポート%/ode/services/listServices
- WebService化されたプロセスのオペレーション名は開始メッセージイベントの名称、または開始メッセージイベントのプロパティ項目[テクニカル・ネーム]値が利用されます。
  開始メッセージイベント名を指定指定無い場合"EventStartMessage"がオペレーション名となります。
  詳細は書くWSDLファイルの内容をご確認下さい。

■例

下記ビジネス・プロセス・ダイアグラム図をデプロイした場合のWSDLファイルパス、及びWebServiceエンドポイントの例を示します。

BPMサーバホスト	bpm.intra_mart.co.jp
BPMサーバポート	8080
プロジェクト名	Project
ダイアグラム名	Diagram
実行不可プール名	NonExecutePool
実行プール名	ExecutePool

WSDLファイルパス
http://bpm.intra_mart.co.jp:8080/ode/processes/Project/Diagram/ExecutePool/NonExecutePool?wsdl

WebServiceエンドポイント
http://bpm.intra_mart.co.jp:8080/ode/processes/Project/Diagram/ExecutePool/NonExecutePool

Column

WebService化されたプロセスとの連携

WebServiceとして定義されたプロセスは別のプロセスからWebService連携を行う事により呼出が可能となります。

呼出先のプロセスを定義したプロジェクト配下にあるbuildフォルダ内のwsdl, xsdファイルを呼出元のプロジェクト配下へコピーします。
（呼出先のプロジェクト配下にあるbuildフォルダにはコピーしないで下さい）

呼出元プロジェクトへコピーしたWSDLファイルは通常のWebServiceとしてプロセスから呼び出す事が可能となります。
WebService連携方法は「2.3外部WebService連携」を参照してください。

2.4.4 例外メッセージの設定

WebServiceとして作成するプロセスは、WebService例外を通知する為の設定を行う事が可能です。
本項ではWebService例外を設定する為の手順を示します。

1 複数のメッセージを返却するビジネス・プロセス・ダイアグラム図を準備します。
メッセージコネクションに設定するメッセージ型はそれぞれ返却用の型、例外用の型と分けて設定する事が可能です。

2 例外メッセージを返却するメッセージ終了イベントを選択し、[プロパティー]ビューを表示します。

3 [プロパティー]ビューから[Response fault name]項目の値に例外メッセージ名を指定します。

4 例外メッセージとして設定が変更されました。

メッセージコネクションに指定したメッセージ形式のアイコンが例外を示すアイコンに変更されます。

- 例外メッセージの内容は、メッセージ終了イベントの前にデータマッピング用のタスクを配置し、値のマッピングを行う事により指定が可能です。
  マッピング方法に関しては「1.5マッピング」を参照してください。

2.5 非同期処理プロセス

2.5.1 非同期処理プロセスとは

同期処理プロセスは、メッセージコネクションで接続された外部のサービスとの連携を行う場合、必ずレスポンスが返るまでブロックされる仕組みです。
それに対し、非同期処理プロセスはメッセージコネクションで接続された外部のサービスを呼出した後、応答を待たずに即座に次のアクティビティへ進む事が可能です。
短時間で処理が完了する外部のサービスの場合は同期処理プロセスによる処理で問題ありませんが、長時間にわたるサービス(例: ドキュメントワークフロー)との連携を行う場合非同期処理プロセスとしてプロセスの作成を行います。

2.5.2 相関セットキー

相関セットキーとは、ビジネス・プロセスが外部からのメッセージ(コールバック)を受け取った際に実行中のビジネス・プロセスを一意に判断する為のキーとなります。
相関セットキーが設定されていない場合、実行中のプロセスを一意に判断する事が出来ません。
非同期処理プロセスを作成する場合、必ず必要となります。

2.5.3 メッセージ中間イベント

メッセージ中間イベントは外部へのメッセージ送信、または外部からのメッセージを受信する為のイベントです。
非同期処理プロセスを作成する場合、外部サービスからのメッセージを受信(コールバック)する為の中間メッセージイベントを配置します。
本項では外部からのメッセージを受信するメッセージ中間イベントの配置方法及び、メッセージ受信時にプロセスを一意に判断する為の相関セットキーの作成方法を示します。

2.5.3.1 動作概要

本項で作成例として提示するメッセージ中間イベントを利用したビジネス・プロセスの動作概要を説明します。

1 プロセスの呼出元を表現する[クライアント]プール上にある[プロセス呼出]タスクから[プロセス]プール上にある[プロセス開始]イベントを呼出します。

2 [プロセス開始]イベントが呼出される事により、ビジネス・プロセスが開始され、[プロセス開始]イベントの次に配置されている[メッセージ受信]イベントへ到達します。

3 [メッセージ受信]イベントは、[クライアント]プール上にある[プロセス再開]タスクからのメッセージを受信するまで待機します。

4 [クライアント]プール上にある[プロセス再開]タスクから[プロセス]プール上にある[メッセージ受信]イベントに対し、メッセージを送信する事によりビジネス・プロセスが再開されます。

5 再開されたビジネス・プロセスは[プロセス終了]イベントへ到達し、ビジネス・プロセスが完了します。

- ビジネス・プロセスの開始はWebServiceとして呼出を行います。
  WebServiceプロセスに関しては「2.3外部WebService連携」を参照してください。
- [メッセージ受信]イベントに対して送信されるメッセージには、相関セットキーが含まれます。
  相関セットキーにより、実行中のプロセスを一意に判断し、ビジネス・プロセスが再開されます。
- [メッセージ受信]イベントに対するメッセージはWebServiceプロセスに定義されたメッセージ受信イベント用のオペレーションを外部から呼出します。

2.5.3.2 ビジネス・プロセス・ダイアグラムの作成

1 ビジネス・プロセス・プロジェクトを作成し、ビジネス・プロセス・ダイアグラムを記述します。
各プール、タスク、イベント等のアクティビティのプロパティを下記の表の通りに設定します。

[クライアント]プール

テクニカル・ネーム	Client

[プロセス呼出]タスク

テクニカル・ネーム	CallProcess

[プロセス再開]タスク

テクニカル・ネーム	ResumeProcess

[プロセス]プール

テクニカル・ネーム	Process

[プロセス開始]イベント（メッセージ開始イベント）

テクニカル・ネーム	StartEvent
Incoming message variable name	StartEventMessage

[メッセージ受信]イベント（メッセージ中間イベント）

テクニカル・ネーム	ResumeEvent
Incoming message variable name	ResumeEventMessage

[プロセス終了]イベント（空メッセージ終了イベント）

テクニカル・ネーム	EndEvent

- アクティビティに日本語名を付けた場合、必ずテクニカル・ネームを設定する必要があります。
- Incoming message variable nameは、プール間に設定されているメッセージコネクションの名称を設定します。
  設定した変数名は[マッパー]ビュー、[データ・エディタ]ビューから利用する事が可能です。

2.5.3.3 相関セットキーの作成

本項ではプロセスを一意に判断する為の相関セットキーの作成方法を示します。
相関セットキーは、メッセージに対し設定する事が可能です。

1 [データ・エディタ]ビューを表示します。

2 [データ・エディタ]右部に表示されている[Process]プールを選択します。

3 [Process]プールを右クリックし、コンテキストメニューを表示し、[<bpel:property>を作成]を選択します。

4 コンテキストメニューから[<bpel:property>を作成]を選択し、プロパティの作成を行います。
作成ボタンを押下し、プロパティを作成します。

- プロパティ名を変更する場合、name="prop"部分(name属性)を変更してください。
- 本項ではデフォルト値propを利用します。

5 BPELプロパティが作成されました。

作成されたBPELプロパティは[Process]プール配下にある名前空間

[namespace: xmlns:this="http://thisdiagram/Process"]内に作成されます。

- 名前空間namespace: xmlns:this="…"は実行可能プール名により決定されます。
  本項では[Process]プールの名前が利用されますが、プール名により名前空間が
  namespace:xmlns:this="http://thisdiagram/[実行可能プール名]"形式となりますのでご注意下さい。

6 次に[データ・エディタ]ビュー左部にある[Process]プール内に表示されている[declarations]を選択します。

7 [{$} declarations]を右クリックし、コンテキストメニューを表示し、[CorrelationSetを生成]を選択します。

8 変数を生成ダイアログが表示されます。
相関セットキーとして利用する変数名を入力し、作成ボタンを押下します。

- 本項では相関セットキーとしてcorrelationSetKeyという名称で変数を作成します。

9 相関セットキーが作成されました。

- 相関セットキー作成直後は、プロパティとの関連付けが行われていない為、赤字でエラーとして表示されます。

10 作成した相関セットキーはBPELプロパティを関連付ける必要があります。
関連付けを行うには、BPELプロパティを相関セットキーにドラッグ&ドロップします。

11 相関セットキーにBPELプロパティが関連付けられました。

2.5.3.4 相関の設定

作成した相関セットキーは、メッセージに対して関連付け(相関の設定)を行う事により外部からのメッセージと、ビジネス・プロセス一意に紐付けが行えます。
本項では相関の設定手順を示します。

1 相関セットキーの実体であるBPELプロパティに対し、メッセージのエイリアスを設定します。
相関セットとして利用するメッセージ変数にBPELプロパティエイリアスを作成します。
[データ・エディタ]ビュー左部より[Process]プール → [declarations] → [$StartEventMessage] → [body] → [text:string]を選択します。

2 [text: string]を右クリックし、コンテキストメニューを表示し、[<bpel:propertyAlias>を作成...]を選択します。

- BPELプロパティエイリアスは、メッセージ変数内にある、型情報に対して指定を行ってください。
  メッセージに対してBPELプロパティエイリアスの設定を行う事は出来ません。

3 propertyName属性に、BPELプロパティ名を指定後、作成ボタンを押下し、BPELプロパティエイリアスを作成します。

- BPELプロパティ名にデフォルトprop以外の名称を指定した場合はpropertyName属性を編集して下さい。
- 本項ではデフォルト値propを利用します。

4 BPELプロパティエイリアスが作成されました。

5 [$StartEventMessage]と同様に、[$ResumeEventMessage]に対してもBPELプロパティエイリアスを作成します。

- 作成を行うエイリアスの実体BPELプロパティは同一のBPELプロパティ(this:prop)を指定してください。

6 次に、メッセージに対し、相関セットキーを割当てる為の設定を行います。
[データ・エディタ]ビュー左部 [Process]プール → [<シーケンス>] → [StartEvent] → [incoming for $StartEventMessage...] に対し、作成した相関セットキー[$correlationSetKey]をドラッグ&ドロップし関連付けを行います。

7 [incoming for $StartEventMessage...]に相関セットキーとの関連付けが行われました。

8 次に、関連付けた相関情報に関する設定を行います。

関連付けにより作成された[<bpel:correlation set=...>]要素を右クリックし、コンテキストメニューを表示し、[編集]を選択します。

9 編集ダイアログから、bpel:correlation要素にinitiate属性を追加し、適用ボタンを押下します。

initiate	相関の開始を表します。 ここでは、相関セットキーとして受け渡されるメッセージの為"yes"を設定します。

10 [incoming for $StartEventMessage...]に対し、相関情報の設定が行われました。

11 同様に、[Process]プール → [<シーケンス>] → [ResumeEvent] → [incoming for $ResumeEventMessage...]に対し、作成した相関セットキー[$correlationSetKey]をドラッグ&ドロップし関連付けを行います。

12 [incoming for $ResumeEventMessage...]に相関セットキーとの関連付けが行われました。

13 次に、関連付けた相関情報に関する設定を行います。

関連付けにより作成された[<bpel:correlation set=...>]要素を右クリックし、コンテキストメニューを表示し、[編集]を選択します。

- BPELプロパティエイリアスは、メッセージ変数内にある、型情報に対して指定を行ってください。
  メッセージに対してBPELプロパティエイリアスの設定を行う事は出来ません。

14 編集ダイアログから、bpel:correlation要素にinitiate属性を追加し、適用ボタンを押下します。

initiate	相関の開始を表します。 前述した[incoming for $StartEventMessage...]において既に相関の開始が表現されている為、ここでは相関の開始ではないことを示す"no"を指定します。

15 [incoming for $ResumeEventMessage...]に対し、相関情報の設定が行われました。

2.5.3.5 相関を利用した外部からのビジネス・プロセスの呼出

前項で作成したプロセスをデプロイする事により、外部からビジネス・プロセスの開始、及び再開をWebServiceとして呼び出す事が可能となります。

ビジネス・プロセスの開始を表すプロセス開始(StartEvent)及び、メッセージを受信し、ビジネス・プロセスを再開するメッセージ受信(ResumeEvent)がそれぞれWebServiceのオペレーションとして提供されます。
それぞれのオペレーションを呼出す際のパラメータとして、相関セットキーとなるユニークな値を受け渡す事により、ビジネス・プロセスを一意に判断しメッセージを受け渡す事が可能です。

- BPELプロパティエイリアスは、メッセージ変数内にある、型情報に対して指定を行ってください。
- 外部からWebServiceとしてビジネス・プロセスを呼出す方法は「2.4プロセスのWebService化」を参照してください。

2.5.4 外部WebServiceを利用した非同期プロセス

前述の非同期処理プロセスでは、相関セットキーをビジネス・プロセス開始時に受取るという形で相関を表現していましが、ビジネス・プロセス内部で相関セットキーを作成する事も可能です。
本項では、ビジネス・プロセス内部で相関セットキーを生成し、生成した相関セットキーを外部のWebServiceに対して通知する手順を示します。

2.5.4.1 動作概要

本項で作成例として提示するメッセージ中間イベントを利用したビジネス・プロセスの動作概要を説明します。

1 プロセスの呼出元を表現する[クライアント]プール上にある[プロセス呼出]タスクから[プロセス]プール上にある[プロセス開始]イベントを呼出します。

2 [プロセス開始]イベントが呼出される事により、ビジネス・プロセスが開始され、[プロセス開始]イベントの次に配置されている[メッセージ受信]イベントへ到達します。

3 [メッセージ受信]イベントは、[クライアント]プール上にある[プロセス再開]タスクからのメッセージを受信するまで待機します。

4 [クライアント]プール上にある[プロセス再開]タスクから[プロセス]プール上にある[メッセージ受信]イベントに対し、メッセージを送信する事によりビジネス・プロセスが再開されます。

5 再開されたビジネス・プロセスは[プロセス終了]イベントへ到達し、ビジネス・プロセスが完了します。

- ビジネス・プロセスの開始はWebServiceとして呼出を行います。
  WebServiceプロセスに関しては「2.4プロセスのWebService化」を参照してください。
- [WebService]呼出タスクから送信するメッセージには相関セットキーが含まれます。
- [メッセージ受信]イベントに対して送信されるメッセージには、相関セットキーが含まれます。
  相関セットキーにより、実行中のプロセスを一意に判断し、ビジネス・プロセスが再開されます。
- [メッセージ受信]イベントに対するメッセージはWebServiceプロセスに定義されたメッセージ受信イベント用のオペレーションを外部から呼出します。
- プロセス中から呼出されるWebServiceに関しては、どのような形式のWebServiceでも利用する事が可能です。

2.5.4.2 ビジネス・プロセス・ダイアグラムの作成

1 ビジネス・プロセス・プロジェクトを作成し、ビジネス・プロセス・ダイアグラムを記述します。
各プール、タスク、イベント等のアクティビティのプロパティを下記の表の通りに設定します。

[クライアント]プール

テクニカル・ネーム	Client

[プロセス呼出]タスク

テクニカル・ネーム	CallProcess

[プロセス]プール

テクニカル・ネーム	Process

[プロセス開始]イベント（メッセージ開始イベント）

テクニカル・ネーム	StartEvent
Incoming message variable name	StartEventMessage

[WebService呼出]タスク

テクニカル・ネーム	CallWebService
Outgoing message variable name	WebServiceRequestMessage
Incoming message variable name	WebServiceResponseMessage

[メッセージ受信]イベント（メッセージ中間イベント）

テクニカル・ネーム	ResumeEvent
Incoming message variable name	ResumeEventMessage

[プロセス終了]イベント（空メッセージ終了イベント）

テクニカル・ネーム	EndEvent

[WebService]プール

テクニカル・ネーム	WebService

[WebService]タスク

テクニカル・ネーム	WebService

[プロセス再開]タスク

テクニカル・ネーム	ResumeProcess

- アクティビティに日本語名を付けた場合、必ずテクニカル・ネームを設定する必要があります。
- Incoming message variable name、Outgoing message variable nameは、プール間に設定されているメッセージコネクションの名称を設定します。
  設定した変数名は[マッパー]ビュー、[データエディタ]ビューから利用する事が可能です。

2.5.4.3 相関セットキーの作成

非同期処理プロセス作成方法と同様に相関セットキーの作成を行います。
相関セットキーの作成方法は前述の「2.5.3.3相関セットキーの作成」を参照してください。

- 本項では、相関セットキーとしてcorrelationSetKeyという名称で変数を作成します。

2.5.4.4 相関の設定

作成した相関セットキーは、メッセージに対して関連付け(相関の設定)を行う事により呼出を行う外部WebServiceを通じてキー値を通知する事が可能です。
その後、外部からのメッセージを受信する際に通知したキー値を受取る事により、送信時のビジネス・プロセスインスタンスと一意に紐付けが行えます。
本項では相関の設定手順を示します。

1 相関セットキーの実体であるBPELプロパティに対し、メッセージのエイリアスを設定します。
相関セットとして利用するメッセージ変数にBPELプロパティエイリアスを作成します。
[データ・エディタ]ビュー左部より[Process]プール → [declarations] →
[$WebServiceRequestMessage]に含まれるWebService呼出用パラメータにおいて、相関セットとして利用するメッセージ変数を選択します。

2 選択したメッセージ変数を右クリックし、コンテキストメニューを表示し、[<bpel:propertyAlias>を作成...]を選択します。

- BPELプロパティエイリアスは、メッセージ変数内にある、型情報に対して指定を行ってください。
  メッセージに対してBPELプロパティエイリアスの設定を行う事は出来ません。

3 propertyName属性に、BPELプロパティ名を指定後、作成ボタンを押下し、BPELプロパティエイリアスを作成します。

- BPELプロパティ名にデフォルトprop以外の名称を指定した場合はpropertyName属性を編集して下さい。
- 本項ではデフォルト値propを利用します。

4 BPELプロパティエイリアスが作成されたことを確認します。

5 同様に、[$ResumeEventMessage]に対してもBPELプロパティエイリアスを作成します。

- 作成を行うエイリアスの実体BPELプロパティは同一のBPELプロパティ(this:prop)を指定してください。

6 次に、メッセージに対し、相関セットキーを割当てる為の設定を行います。
[データ・エディタ]ビュー左部 [Process]プール → [<シーケンス>] →
[CallWebService] →[out going for $WebServiceRequestMessage...] に対し、
作成した相関セットキー[$correlationSetKey]をドラッグ&ドロップし関連付けを行います。

7 [outgoing for $WebServiceRequestMessage...]に相関セットキーとの関連付けが行われました。

8 次に、関連付けた相関情報に関する設定を行います。

関連付けにより作成された[<bpel:correlation set=...>]要素を右クリックし、コンテキストメニューを表示し、[編集]を選択します。

9 編集ダイアログから、bpel:correlation要素にinitiate属性及びpattern属性を追加し、適用ボタンを押下します。

initiate	相関の開始を表します。 ここでは、相関セットキーとしてを外部に送信し、相関の開始を行う為"yes"を指定します。
pattern	相関のパターンを表します。 外部WebServiceに対し相関セットキーの送信を行う場合必ず"request"を指定してください。

10 [$WebServiceRequestMessage]に対し、相関情報の設定が行われたことを確認します。

11 同様に、[Process]プール → [<シーケンス>] → [ResumeEvent] → [incoming for $ResumeEventMessage...]に対し、作成した相関セットキー[$correlationSetKey]をドラッグ&ドロップし関連付けを行います。

12 [incoming for $ResumeEventMessage...]に相関セットキーとの関連付けが行われた事を確認します。

13 次に、関連付けた相関情報に関する設定を行います。

関連付けにより作成された[<bpel:correlation set=...>]要素を右クリックし、コンテキストメニューを表示し、[編集]を選択します。

- BPELプロパティエイリアスは、メッセージ変数内にある、型情報に対して指定を行ってください。
  メッセージに対してBPELプロパティエイリアスの設定を行う事は出来ません。

14 編集ダイアログから、bpel:correlation要素にinitiate属性を追加し、適用ボタンを押下します。

initiate	相関の開始を表します。 前述した[outgoing for $WebServiceRequestMessage...]において既に相関の開始が表現されている為、ここでは相関の開始ではないことを示す"no"を指定します。

15 [$ResumeEventMessage]に対し、相関情報の設定が行われました。

2.5.4.5 相関セットキー値のマッピング

前項で作成したプロセスは、外部WebServiceの呼出を行う際のメッセージとして相関セットキーが送信されます。
相関セットキーの送信が行われた時点で相関の開始が行われますので、相関セットキーを送信する際にキー値を設定する必要があります。
本項ではキー値を実行中のプロセス固有のIDであるプロセスインスタンスID($ode:pid)を利用し相関セットキーを送信する為の設定方法を記します。

1 [プロセスプール]から、外部WebServiceを呼出している[WebService呼出]タスクを選択します。

2 [マッパー]ビューを表示し、右部にWebService呼出用パラメータが表示されている事を確認します。

3 前項において設定を行った相関セットキーと関連付けられたパラメータに対し、プロセスインスタンスIDのマッピングを行います。

- [マッパー]を利用したデータマッピング方法に関しては「1.5マッピング」を参照してください。

2.5.4.6 相関を利用した外部からのビジネス・プロセスの呼出

前項で作成したプロセスをデプロイする事により、外部からビジネス・プロセスの開始、再開に必要なキー情報が外部のWebServiceに送信されます。
前述の[2.5.3 メッセージ中間イベント]プロセスと違い、プロセスを開始する際に相関セットキーを送信する必要がなく、外部WebServiceが相関セットキーを受信する事が可能です。
ビジネス・プロセスの再開は、外部WebServiceが受信した相関セットキーを用いてビジネス・プロセスを再開するメッセージ受信(ResumeEvent)イベントに対しメッセージを送信する事で再開が可能です。

- 外部からWebServiceとしてビジネス・プロセスを呼出す方法は[2.4プロセスのWebService化]を参照してください。

2.6 リスク・コントロールマトリックス

2.6.1 概要

内部統制に対応する為にリスクおよびコントロール情報を管理し、業務フローに関連付けてリスクマネジメントを向上させる為のツールです。
リスクコントロールマトリックスでは以下の機能を有します。

- 既存のリスクおよびコントロールデータがエクセルにある場合、エクセルのデータをインポート機能により取り込むことが可能です。（エクセルのデータは XML 形式に変換する必要があります）
- リスクおよびコントロールデータの新規登録ができます。
- ダイアグラム中のタスクにリスクおよびコントロールをドラッグ＆ドロップで関連付けできます。
- タスクに設定されたリスクおよびコントロールの一覧表示データを PDF で出力できます。
- リスクおよびコントロールデータはキーワード等による検索ができます。
- 検索されたデータは編集・削除・コピーおよび、リスクとコントロール双方からの関連付けが出来ます。
- エクスポート機能でデータのバックアップができます。（XML 形式）

＜リスクコントロールマトリックス概要図＞

■用語の説明

このツールで使用する用語の説明を行います。

リスク	業務に対するリスクを定義したもの。 [Risk]画面で新規登録するか、インポートで取り込むことにより登録される。
コントロール	リスクに対するコントロールを定義したもの。 [Control]画面で新規登録するか、インポートで取り込むことにより登録される。
RCM	リスクコントロールマトリックスの略称。
リレーション	関連付け情報の意。 リレーションには[リスク]-[コントロール]と、[リスク（またはコントロール）]-[タスク]の２つがある。
XMLデータ	リスク・コントロールデータをXML形式で表現したインポートデータ。
XMLスキーマ	リスク・コントロールデータのフォーマット定義。 XMLデータをBPM|Designer に認識させるためのXMLスキーマ情報なので、XMLデータをインポートする前にスキーマインポートを行う必要があります。
フォーマット定義	XMLデータのカラム配列。

■初期設定

リスクコントロールマトリックスの機能を使用する場合、パースペクティブでRCMを選択して下さい。
メイン画面として、[Risk]画面と[Control]画面、詳細画面として[RCM-Detail]画面が表示されます。

■画面の説明

[Risk]画面と[Control]画面ではリスクまたはコントロールの検索や編集、関連付け等を行います。
リスクやコントロールをクリックすると[RCM-Detail]画面に詳細情報が表示されます。

[Risk]画面
親ツリーにリスク、子ツリーにコントロールが表示されます。
リスクの検索、追加、編集、削除、コピーができます。
また、リスクに対するコントロールとのリレーションの追加と削除ができます。

[Control]画面
親ツリーにコントロール、子ツリーにリスクが表示されます。
コントロールの検索、追加、編集、削除、コピーができます。
また、コントロールに対するリスクとのリレーションの追加と削除ができます。

[RCM-Detail]画面（リスクが上段）

[Risk]画面でリスクまたはコンとロールをクリックした場合に表示されます。

リスクが上段に表示され、下段にはリスクとリレーション関係にあるコントロールが表示されます。

また、タスクに関連付けられたリスクをクリックした場合にもこのように表示されます。

[RCM-Detail]画面（コントロールが上段）

[Control]画面でリスクまたはコンとロールをクリックした場合に表示されます。

コントロールが上段に表示され、下段にはコントロールとリレーション関係にあるリスクが表示されます。

また、タスクに関連付けられたコントロールをクリックした場合にもこのように表示されます。

■機能の説明

リスクコントロールマトリックスには以下の機能があります。

インポート	ＸＭＬデータを、BPM|Designer に取込みリスク・コントロールデータを登録します。
検索	BPM|Designer に登録されたリスク・コントロールデータを検索します。 検索結果はメイン画面（[Risk]画面、[Control]画面）に表示されます。 各リスクやコントロールをクリックすると[RCM-Detail]画面に詳細情報が表示されます。
編集	リスク・コントロールデータの追加・削除・編集・コピーを行います。
関連付け	リスク・コントロールデータを、[リスク]や[コントロール]、もしくは[タスク]と関連付けます。 （リスク同士、またはコントロール同士は関連付けできません）
ＲＣＭ出力	タスクに関連付けられたリスク･コントロールデータを一覧表示でＰＤＦデータ出力します。
エクスポート	BPM|Designer に登録されたリスク・コントロールデータをエクスポート機能で外部出力します。 （ＸＭＬスキーマ、ＸＭＬデータ）
スキーマのインポート	ＸＭＬスキーマを、BPM|Designer に取込みリスク・コントロールのフォーマット定義を登録します。 初期設定ではデフォルトＸＭＬスキーマが設定されていますが､フォーマット定義の変更を行った場合にはスキーマのインポートが必要となります。

■その他

インポートデータの作成を行う際に、以下の環境とファイルが必要となります。

環境

この機能を使用するには、Microsoft Office Professional Edition 2003 が必要です。

その他の場合は以下のAdd-in情報を参照下さい。

http://www.microsoft.com/downloads/details.aspx?FamilyId=72852247-6AFD-425C-83B1-1F94E4AC2775&displaylang=en

ファイル

また、下記で使用するフォーマットRCM一覧.xls は、以下の場所に格納されています。

＜BPM|Designer インストールフォルダ＞/etc/rcm/フォーマットRCM一覧.xls

2.6.2 操作手順

各機能の操作手順を説明します。

2.6.2.1 インポート

インポートを行う為には、インポートデータを準備する必要があります。
以下の説明はＸＭＬデータの作成とＸＭＬデータインポート手順になります。

■インポートデータの作成

インポートデータは既存のリスク/コントロールデータをXML形式に変換して作成します。

1 フォーマットRCM一覧.xls を開く。

2 既存のリスク/コントロールデータをフォーマットRCM一覧.xls へ反映。

3 出力XMLの範囲指定を行う場合は、メニュー ＞XML Tools＞Convert Range to XML List を選択し、[Convert Range to XML List]画面で範囲指定を行って下さい。

OKボタン押下の際は「No, first row contains data」 を選択して下さい。

（「Yes, first row contains column names」を選択するとデフォルトスキーマにマッチしないXMLデータが生成されてしまいます）

詳細は、2.6.3.1（2）XMLスキーマ（フォーマット定義）生成を参照。

4 エクセルのメニュー＞データ＞XML＞エクスポートを選択し、[XMLのエクスポート]画面を表示。

5 任意の場所にXMLデータを作成。

■XMLデータインポート

リスク・コントロールデータを BPM|Designer にインポートします。

1 BPM|Designer のメニュー＞RCM＞XMLデータインポートを選択し、[XMLインポート]画面を表示。

2 [XMLインポート]画面の参照ボタンをクリックし、ファイル選択画面を表示。

3 インポートファイルを指定する。

4 OKボタンをクリックし、インポート処理を実行。

5 インポートが終了した旨のメッセージを表示。

2.6.2.2 検索

リスク、コントロールそれぞれに検索機能があります。
検索は以下の条件を指定し、検索結果を絞り込んでいく機能です。
・プロセス
・キーワード（キーワードの条件指定により検索方法を選択可能）
・アサーション
初期状態は全件が検索結果の対象となります。
プロセス、キーワード、アサーションを指定していくと検索結果がAND条件で絞り込まれていきます。

■リスク検索

リスクは以下の手順で検索します。

1 [Risk]画面のメニュー＞検索 を選択し、[リスク検索]画面を表示します。

2 [リスク検索]画面で検索条件を指定し、OKボタンをクリック。

● 初期値で検索を行うと全件検索になります。

3 [Risk]画面に検索結果が表示されます。

4 ツリーを開くとリレーション関係のあるコントロールが配下に表示されます。

■コントロール検索

コントロールは以下の手順で検索します。

1 [Control]画面のメニュー＞検索 を選択し、[コントロール検索]画面を表示します。

2 [コントロール検索] 画面で検索条件を指定し、OKボタンをクリック。

● 初期値で検索を行うと全件検索になります。

3 [Control]画面に検索結果が表示されます。

4 ツリーを開くとリレーション関係のあるリスクが配下に表示されます。

2.6.2.3 編集

リスク、コントロールは以下の機能によりデータの編集が行えます。

追加　新規にリスク又はコントロールを追加します。

削除　現在あるリスク又はコントロールを削除します。

編集　リスク又はコントロールの情報を変更します。

コピー　リスク又はコントロールをコピーし、リスクNo（又はコントロールNo）を新規採番して作成します。

■リスク追加

リスクの追加は[リスク追加]画面より行います。

Noが重複するリスクは作成できません。Noとリスクは入力必須項目です。

1 [Risk]画面のメニュー＞追加 を選択し、[リスク追加]画面を表示します。

2 [リスク追加]画面表示。Noはデフォルト値として自動採番された値がセットされます。

3 各項目(No, リスク, チェック項目)を入力し、OKボタンクリック。

4 [Risk]画面に新規リスク(RA02)が追加される。

■リスク編集

リスクの編集は[リスク編集]画面より行います。
No、リスク、チェック項目を変更できます。
Noを変更する場合、他のNoと重複する番号は入力エラーとなります。
Noとリスクは入力必須項目です。

1 対象リスクで右クリックしポップアップメニューを表示。
編集を選択し、[リスク編集]画面を表示します。

2 [リスク編集]画面表示。

3 各項目の内容を変更し、OKボタンクリック。

4 [Risk]画面の対象リスクの内容が変更される。

5 対象リスクをクリックし、[RCM-Detail]画面で詳細情報が変更されていることを確認。

＜変更前＞

RCM-Detail

No	リスク	実在性	網羅性	権利と義...	評価の妥...	期間配...	表示の妥...
RA02	リスク(RA02)タイトル	○			○		

＜変更後＞

RCM-Detail

No	リスク	実在性	網羅性	権利と義...	評価の妥...	期間配...	表示の妥...
RA02	リスク(RA02)タイトル コメント追加		○	○			

■リスク削除

リスクの削除を行います。対象リスクに関連するリレーションも同時に削除されます。

1 対象リスクで右クリックしポップアップメニューを表示。削除を選択。

2 削除する旨のメッセージが表示されるので、はいボタンクリック。
（いいえを選択すると処理はキャンセルされます）

3 リスクが削除されたことを確認。

■リスクコピー

対象リスクの情報をコピーしてリスクを新規作成します。

1 対象リスクで右クリックしポップアップメニューを表示。コピーを選択。

2 対象リスクの情報がセットされた[リスク追加]画面表示。

3 各項目（No，リスク，チェック項目）を編集し、OKボタンクリック。

4 [Risk]画面に新規作成されたリスクが追加される。

■コントロール追加

コントロールの追加は[コントロール追加]画面より行います。
Noが重複するコントロールは作成できません。Noとコントロールは入力必須項目です。

1 [Control]画面のメニュー＞追加 を選択し、[コントロール追加]画面を表示します。

2 [コントロール追加]画面表示。Noは自動採番された番号がセットされます。

3 各項目（No，コントロール，チェック項目，テキスト項目）を入力し、OKボタンクリック。

4 [Control]画面に新規コントロール（CA03）が追加される。

■コントロール編集

コントロールの編集は[コントロール編集]画面より行います。
No、コントロール、チェック項目、テキスト項目を変更できます。
Noを変更する場合、他のNoと重複する番号は入力エラーとなります。
Noとコントロールは入力必須項目です。

1 対象コントロールで右クリックしポップアップメニューを表示。

編集を選択し、[コントロール編集]画面を表示します。

2 [コントロール編集]画面表示。

3 各項目の内容を変更し、OKボタンクリック。

4 [Control]画面の対象コントロールの内容が変更される。

5 対象コントロールをクリックし、[RCM-Detail]画面で詳細情報が変更されていることを確認。

＜変更前＞

＜変更後＞

■コントロール削除

コントロールの削除を行います。対象コントロールに関連するリレーションも同時に削除されます。

1 対象コントロールで右クリックしポップアップメニューを表示。削除を選択。

2 削除する旨のメッセージが表示されるので、はいボタンクリック。
（いいえを選択すると処理はキャンセルされます）

3 コントロールが削除されたことを確認。

■コントロールコピー

対象コントロールの情報をコピーしてコントロールを新規作成します。

1 対象コントロールで右クリックしポップアップメニューを表示。コピーを選択。

2 対象コントロールの情報がセットされた[コントロール追加]画面表示。

3 各項目（No，コントロール，チェック項目，テキスト項目）を編集し、OKボタンクリック。

4 [Control]画面に新規作成された，コントロールが追加される。

2.6.2.4 関連付け

リスク、コントロールは以下の関連付けができます。
リスクとコントロールの関連付け
タスクとの関連付け
それぞれリレーション元となるリスク（またはコントロール）を選択し、関連付けを行います。

■リスクとコントロールの関連付け

（1）リスクにコントロールを関連付ける

1 関連元となる対象リスクで右クリックしポップアップメニューを表示。リレーションを選択。

2 [リスクリレーション]画面表示。条件を指定して検索ボタンをクリック。
検索条件の設定詳細については、コントロール検索を参照して下さい。

3 検索結果（コントロール）が表示される。

（既に関連付けられているコントロール[CA02, CA15]は検索結果から除外される）

4 関連付けるコントロール（CA01）をチェックし、関連付けを行う。

（チェックがない場合、選択されていない旨のエラーメッセージが表示される）

5 [Risk]画面で対象リスクにコントロールが関連付けされたことを確認。

6 [Control]画面で逆のリレーションを確認。

（2）コントロールにリスクを関連付ける

1 関連元となる対象コントロールで右クリックしポップアップメニューを表示。リレーションを選択。

2 [コントロールリレーション]画面表示。条件を指定して検索ボタンをクリック。

検索条件の設定詳細については、リスク検索を参照して下さい。

3 検索結果(リスク)が表示される。
(既に関連付けられているリスクは検索結果から除外される)

4 関連付けるリスク(RA04, RA05)をチェックし、関連付けを行う。
(チェックがない場合、選択されていない旨のエラーメッセージが表示される)

5 [Control]画面で対象コントロールにリスクが関連付けされたことを確認。

6 [Risk]画面で逆のリレーションを確認。

（3）関連付けを削除（リスクが親ツリーの場合）

1 [Risk]画面の対象コントロールで右クリックしポップアップメニューを表示。リレーション削除を選択。

2 削除する旨のメッセージが表示されるので、はいボタンクリック。
（いいえを選択すると処理はキャンセルされます）

3 [Risk]画面で親リスク(RA01)と(CA01)のリレーションが削除されていることを確認。

4 [Control]画面で逆のリレーションからも(RA01)と(CA01)のリレーションが削除されていることを確認。

（4）関連付けを削除（コントロールが親ツリーの場合）

1 [Control]画面の対象リスクで右クリックしポップアップメニューを表示。リレーション削除を選択。

2 削除する旨のメッセージが表示されるので、はいボタンクリック。
（いいえを選択すると処理はキャンセルされます）

3 [Control]画面で親コントロール(CA01)と(RA01)のリレーションが削除されていることを確認。

4 [Risk]画面で逆のリレーションからも(CA01)と(RA01)のリレーションが削除されていることを確認。

Column

削除の種類について

リスクコントロールマトリックスの削除処理では、削除対象が二種類あります。
一つはリスクやコントロールそのものを削除する場合。
もう一つはリスクとコントロールのリレーションを削除する場合です。

● リスクやコントロールそのものを削除する場合
　親ツリーのリスクやコントロールを指定して削除した場合、対象のリスク（またはコントロール）が削除されます。
また、削除対象リスク（またはコントロール）を含むリレーションも同時に削除されます。

●リレーションを削除する場合
　子ツリーのリスクやコントロールを指定してリレーション削除した場合、対象のリスク（またはコントロール）自体は削除されず、親ツリーと子ツリーの関係に表現されるリレーション（関連付け）のみが削除されます。

●リスク・コントロールとリレーションの関係

■タスクとリスク/コントロールの関連付け

タスクとリスク（またはコントロール）の関連付けは以下の手順で行います。

（1）タスクにリスクを関連付ける

1 [Risk]画面で対象リスク(RA01)をドラッグし、対象タスク（見積書作成画面）上でドロップします。

2 対象タスクにリスクが関連付けられます。

3 [Control]画面のリスク(RA06)を選択しても関連付けできます。

（2）タスクにコントロールを関連付ける

1 [Control]画面で検索を行い、対象コントロール(CA01)をタスク(見積書作成画面)に関連付けます。

2 対象タスクにコントロールが関連付けられます。

3 [Risk]画面のリスク(CA02)を選択しても関連付けできます。

2.6.2.5 RCM出力

BPM|Designer でタスクに関連付けたリスク/コントロールの情報はPDFファイルで一覧表示ができます。キーワードで検索し、出力したいプロセスを指定（複数可）して出力します。

■PDFファイルの出力

1 RCM＞RCM出力を選択し、[RCM出力]画面を表示。

2 [RCM出力]画面表示。条件を指定して検索ボタンをクリック。

3 検索結果が表示されるので、出力したいプロセスを選択（チェックボックスにチェック）。

4 参照ボタンをクリックし、ファイル選択画面を表示。PDFファイルの出力先のパス（ファイル名）を指定する。

5 出力先のパスが入力されたこを確認後、出力ボタンをクリックしRCM出力処理を実行。

- 出力PDFではプロセス名ごとにグルーピング表示されます。
  全く同じ名前のプロセス名を複数選択するとデータがマージされてしまいます。
  運用上、同名のプロセス名は使用しないで下さい。

6 RCMデータが出力された旨のメッセージを表示。

7 出力されたPDFファイル確認。

RcmSample01.pdf - Adobe Reader

ファイル(F) 編集(E) 表示(V) 文書(D) ツール(T) ウィンドウ(W) ヘルプ(H)

1 / 1 94.3% 検索

プロセス名： RcmSampleDiagram01

リスクNo.	リスク	実在性	網羅性	権利と義務の帰属	期間配分の適切性	表示の妥当性	貸借の妥当性	不正防止	CTRL No	コントロール	予防	発見	マニュアル	システム	主的	従的	統制環境	リスク評価	情報伝達	統制活動	監視活動	ITへの対応	実施時期	年間発生件数	サンプル数	コントロールの実施部署	関連規程マニュアル等	コメント
RA05	規定で定められた売掛債権回収期間と異なる見積を提出することにより、債権回収が遅れてしまうリスク			●	●				CA16	【コントロール16　販売管理システムによる支払期限の自動設定】提出する見積書はすべて販売管理システムから自動作成するが、このとき規定で定められた支払期限情報が自動で設定される。	●			●	●				●		●	随時			営業本部	会計事務	
-	-								CA01	【コントロール01　見積金額決定・値引きの承認】営業責任者／BPO責任者は、営業担当者が起案した値引き伺いを査閲し、粗利（採算）に問題ないかを検討のうえ、承認を行う。	●		●			●			●	●		随時			営業本部	販売規程 利益管理	ここでは、毎月得意先ごとの製品別売価を決定し、これを在庫管理システムの販売単価マスターに登録し、販売価格はこれを利用した自動計算を行うことが前提となっています。コントロール01同様、このような場合はマスター登録の正確性を保証するコントロールが必要となります。 このコントロールは、マスターに登録すべき販売単価自体をオーソライズする手続です。但し、その登録の正確性を保証するものではなく営業上の妥当性（粗利等）を検討するものであり、また詳細な査閲が期待できないため、従的コントロールとしています。
									CA15	【コントロール15　見積－見積チェックリストによる送付前の確認】社印の押印／見積書の送付は営業事務担当者が行う。営業事務担当者は見積チェックリストを確認し、決裁が終了していない見積書には押印／送付を行わない。	●		●						●			随時			マーケテ本部(含む)		
									CA02	【コントロール02　見積金額承認・見積書金額の承認】SH責任者は、見積書に記載されている得意先、品名を確認し、見積単価、見積合計金額が特約店向け価格と一致していることを照合の上、承認を行う。	●		●			●			●	●		随時			営業本部	販売規程 利益管理	コントロール02はマスター登録する内容の上長による承認です。販売単価を未承認で実施することを防止することができ、必要なコントロールではありますが、これだけではマスター登録結果の正確性を保証できないため、従的コントロールとしています。

- 1 -

2008/06/04 14:59

■PDFファイルの構成

＜全体＞

プロセス名

プロセス名： RcmSampleDiagram01

リスクNo.	リスク	実在性	網羅性	権利と義務の帰属	期間配分の適切性	表示の妥当性	表示の妥当性	資産保全	不正防止	CTRL_No	コントロール	予防	発見	マニュアル	システム	主的	従的	統制環境	リスク評価	情報伝達	統制活動	監視活動	ITへの対応	実施時期	年間発生件数	サンプル数	コントロールの実施部署	関連規程マニュアル等	コメント
RA05	規定で定められた 売掛債権回収期間と異なる見積を提出することにより、 債権回収が遅れてしまうリスク			●	●					CA16	【コントロール16　販売管理システムによる支払期限の自動設定】 提出する見積書はすべて販売管理システムから自動作成するが、このとき規定で定められた支払期限情報が自動で設定される。	●			●	●					●		●	随時			営業本部	会計事務	
-	-									CA01	【コントロール01　見積金額決定-値引きの承認】 営業責任者 / BPG責任者は、営業担当者が起案した値引き伺いを査閲し、粗利（探算）に問題ないかを検討のうえ、承認を行う。	●		●			●				●	●		随時			営業本部	販売規程 利益管理	ここでは、毎月得意先ごとの製品別売価を決定し、これを在庫管理システムの販売単価マスターに登録し、販売価格はこれを利用した自動計算を行うことが前提となっています。 コントロール01同様、このような場合はマスター登録の正確性を保証するコントロールが必要となります。 このコントロールは、マスターに登録すべき販売単価自体をオーソライズする手続です。但し、その登録の正確性を保証するものではなく営業上の妥当性（粗利等）を検討するものであり、また詳細な査閲が期待できないため、従的コントロールとしています。
										CA15	【コントロール15　見積－見積チェックリストによる送付前の確認】 社印の押印 / 見積書の送付は営業事務担当者が行う。営業事務担当者は見積チェックリストを確認し、決裁が終了していない見積書には押印 / 送付を行わない。	●		●							●			随時			マーケテ本部(営3)		
										CA02	【コントロール02　見積金額承認-見積書金額の承認】 SH責任者は、見積書に記載されている得意先、品名を確認し、見積単価、見積合計金額が特約店向け価格と一致していることを照合の上、承認を行う。	●		●			●				●	●		随時			営業本部	販売規程 利益管理	コントロール02はマスター登録する内容の上長による承認です。販売単価を未承認で実施することを防止することができ、必要なコントロールではありますが、これだけではマスター登録結果の正確性を保証できないため、従的コントロールとしています。

リスク

コントロール

ページ番号

作成日

- 1 -

2008/06/04 14:59

＜リスク＞

リスクNo.	リスク	実在性	網羅性	権利と義務の帰属	期間配分の適切性	表示の妥当性	表示の妥当性	資産保全	不正防止	CTRL_No	コントロール
RA05	規定で定められた 売掛債権回収期間と異なる見積を提出することにより、 債権回収が遅れてしまうリスク			●	●					CA16	【コントロール16　販売管理システムによる支払期限の自動設定】 提出する見積書はすべて販売管理システムから自動作成するが、このとき規定で定められた支払期限情報が自動で設定される。
-	-									CA01	【コントロール01　見積金額決定-値引きの承認】 営業責任者 / BPG責任者は、営業担当者が起案した値引き伺いを査閲し、粗利（探算）に問題ないかを検討のうえ、承認を行う。
										CA15	【コントロール15　見積－見積チェックリストによる送付前の確認】 社印の押印 / 見積書の送付は営業事務担当者が行う。営業事務担当者は見積チェックリストを確認し、決裁が終了していない見積書には押印 / 送付を行わない。
										CA02	【コントロール02　見積金額承認-見積書金額の承認】 SH責任者は、見積書に記載されている得意先、品名を確認し、見積単価、見積合計金額が特約店向け価格と一致していることを照合の上、承認を行う。

リスクに関連付けられたコントロールがリスクの右に表示される。

コントロールのみの場合はリスクは“－”で表示される。

＜コントロール＞

CTRL_No	コントロール	予防	発見	マニュアル	システム	主的	従的	統制環境	リスク評価	情報伝達	統制活動	監視活動	ITへの対応	実施時期	年間発生件数	サンプル数	コントロールの実施部署	関連規程マニュアル等	コメント
CA16	【コントロール16　販売管理システムによる支払期限の自動設定】 提出する見積書はすべて販売管理システムから自動作成するが、このとき規定で定められた支払期限情報が自動で設定される。	●			●	●					●		●	随時			営業本部	会計事務	
CA01	【コントロール01　見積金額決定-値引きの承認】 営業責任者 / BPG責任者は、営業担当者が起案した値引き伺いを査閲し、粗利（採算）に問題ないかを検討のうえ、承認を行う。	●		●			●				●	●		随時			営業本部	販売規程 利益管理	ここでは、毎月得意先ごとの製品別売価を決定し、これを在庫管理システムの販売単価マスターに登録し、販売価格はこれを利用した自動計算を行うことが前提となっています。 コントロール01同様、このような場合はマスター登録の正確性を保証するコントロールが必要となります。 このコントロールは、マスターに登録すべき販売単価自体をオーソライズする手続です。但し、その登録の正確性を保証するものではなく営業上の妥当性（粗利等）を検討するものであり、また詳細な査閲が期待できないため、従的コントロールとしています。
CA15	【コントロール15　見積－見積チェックリストによる送付前の確認】 社印の押印 / 見積書の送付は営業事務担当者が行う。営業事務担当者は見積チェックリストを確認し、決裁が終了していない見積書には押印 / 送付を行わない。	●		●							●			随時			マーケテ本部(営3)		
CA02	【コントロール02　見積金額承認-見積書金額の承認】 SH責任者は、見積書に記載されている得意先、品名を確認し、見積単価、見積合計金額が特約店向け価格と一致していることを照合の上、承認を行う。	●		●			●				●	●		随時			営業本部	販売規程 利益管理	コントロール02はマスター登録する内容の上長による承認です。販売単価を未承認で実施することを防止することができ、必要なコントロールではありますが、これだけではマスター登録結果の正確性を保証できないため、従的コントロールとしています。

2.6.2.6 エクスポート

エクスポートには、XMLデータエクスポートとスキーマエクスポートがあります。
エクスポートされたXMLデータとXMLスキーマはバックアップデータとして使用できます。

■XMLデータエクスポート

BPM|Designer のリスク/コントロールデータを外部ファイルにエクスポートします。

1 RCM＞XMLデータエクスポートを選択し、[XMLエクスポート]画面を表示。

2 [XMLエクスポート]画面の参照ボタンをクリックし、ファイル選択画面を表示。

3 エクスポート先のパス(ファイル名)を指定する。

4 OKボタンをクリックし、エクスポート処理を実行。

5 エクスポートが終了した旨のメッセージを表示。

■Excelへの取込

エクスポートされたXMLデータをExcelに取込みます。

1 フォーマットRCM一覧.xlsのメニュー＞データ＞XML＞インポートを選択し、[XMLのインポート]画面表示。

2 インポートデータを選択し、インポートボタン押下。

3 データがフォーマットRCM一覧.xlsに取り込まれる。

		要件(アサーション)										CTRL タイプ1		CTRL タイプ2		主従		基本的要素					
リスクNo.	リスク	実在性	網羅性	権利と義務の帰属	評価の妥当性	期間配分の適切性	表示の妥当性	資産保全	不正防止(コンプライアンス)	CTRL No.	コントロール	予防	発見	マニュアル	システム	主的	従的	統制環境	リスク評価	情報伝達	統制活動	監視活動	ITへの対応2
RA01	承認手続きを経ない見積書が送付されることにより、実際より過大な受注金額が計上されるリスク。	○		○					○	CAI5	【コントロール15 見積～見積チェックリストによる送付前の確認】 社印の押印／見積書の送付は営業事務担当者が行う。営業事務担当者は見積チェックリストを確認し、決裁が終了していない見積書には押印／送付を行わない。	○		○							○		
					○	○			○	CAO2	【コントロール02 見積金額承認-見積書金額の承認】 SH後任者は、見積書に記載されている得意先、品名を確認し、見積単価、見積合計金額が特約/店向け価格と一致していることを照合の上、承認を行う。	○		○			○				○	○	
RA06	過大な値引き／不適切な価格設定を行うことで利益の当期計上額が正しくなくなるリスク				○	○			○	CAO1	【コントロール01 見積金額決定-値引きの承認】 営業責任者／BPG責任者は、営業担当者が起案した値引き(増)を査閲し、粗利(採算)に問題ないかを検討のうえ、承認を行う。	○		○			○				○	○	
RA04	未承認の取引先と取引を行うことにより、不適切な販売を行い、売掛金が回収されないリスク			○	○					CAI4	【コントロール14 取引先の与信調査】 取引先の支払い能力を調査するため、商用の信用情報データベース／それに代替するもの(企業ホームページの情報／BPGの与信管理表など)を元に与信調査を行う。	○		○			○		○			○	
RA05	規定で定められた売掛債権回収期間と異なる見積を提出することにより、債権回収が遅れてしまうリスク			○	○					CAI6	【コントロール16 販売管理システムによる支払期限の自動設定】 提出する見積書はすべて販売管理システムから自動作成するが、このとき規定で定められた支払期限情報が自動で設定される。	○			○	○					○		○

フォーマット

■スキーマエクスポート

BPM|Designer のXMLスキーマデータを外部ファイルにエクスポートします。
初期設定時はデフォルトXMLスキーマがエクスポートされます。

1 RCM＞スキーマエクスポートを選択し、[スキーマエクスポート]画面を表示。

2 [スキーマエクスポート]画面の参照ボタンをクリックし、ファイル選択画面を表示。

3 エクスポート先のパス(ファイル名)を指定する。

4 OKボタンをクリックし、エクスポート処理を実行。

5 エクスポートが終了した旨のメッセージを表示。

2.6.3 その他

2.6.3.1 フォーマット定義の変更方法

インストール時に設定されるデフォルトXMLスキーマ（フォーマット定義）ではなく、自社のフォーマットに合わせたXMLスキーマを使用したい場合について説明します。
上記、スキーマエクスポートで取得されたXMLスキーマを元に説明します。

■フォーマット定義（XMLスキーマ）の編集

フォーマット定義は以下の属性の編集が可能です。エクスポートしたXMLスキーマをテキストエディターで開き編集します。

属性	物理名	設定値	説明
オブジェクトタイプ	objType	risk control	カラムの情報がリスクかコントロールかを指定。
グループ名	group	任意（グループ名）	BPM|Designer で表示する際のグループ名を指定。 グループなしの場合は空白指定。
表示名	viewName	任意（項目名）	BPM|Designer で表示する際の項目名を指定。
表示タイプ	viewType	text texarea boolean	BPM|Designer で表示する際の入力タイプを指定。 text：１行文字列。 texarea ： 複数行の文字列。 boolean ： チェックボックス。

各xsd:elementタグはカラムを表します。カラムの属性として上記表のobjType、group、viewName、viewTypeが含まれます。これらの属性を修正するとBPM|Designer 上の表示が変更されます。

```
<xsd:element viewType="text" viewName="リスクNo." group="" objType="risk" minOccurs="0" nillab
<xsd:element viewType="textarea" viewName="リスク" group="" objType="risk" minOccurs="0" nilla
<xsd:element viewType="boolean" viewName="実在性" group="要件（アサーション）" objType="risk"
<xsd:element viewType="boolean" viewName="網羅性" group="要件（アサーション）" objType="risk"
<xsd:element viewType="boolean" viewName="権利と義務の帰属" group="要件（アサーション）" objTy
<xsd:element viewType="boolean" viewName="評価の妥当性" group="要件（アサーション）" objType="
<xsd:element viewType="boolean" viewName="期間配分の適切性" group="要件（アサーション）" objTy
<xsd:element viewType="boolean" viewName="表示の妥当性" group="要件（アサーション）" objType="
<xsd:element viewType="boolean" viewName="資産保全" group="" objType="risk" minOccurs="0" nill
<xsd:element viewType="boolean" viewName="不正防止（コンプライアンス）" group="" objType="risk
<xsd:element viewType="text" viewName="CTRLNo." group="" objType="control" minOccurs="0" nilla
<xsd:element viewType="textarea" viewName="コントロール" group="" objType="control" minOccurs=
<xsd:element viewType="boolean" viewName="予防" group="CTRLタイプ1" objType="control" minOccur
<xsd:element viewType="boolean" viewName="発見" group="CTRLタイプ1" objType="control" minOccur
<xsd:element viewType="boolean" viewName="マニュアル" group="CTRLタイプ2" objType="control" mi
<xsd:element viewType="boolean" viewName="システム" group="CTRLタイプ2" objType="control" minO
<xsd:element viewType="boolean" viewName="主的" group="主従" objType="control" minOccurs="0" n
<xsd:element viewType="boolean" viewName="従的" group="主従" objType="control" minOccurs="0" n
<xsd:element viewType="boolean" viewName="統制環境" group="基本的要素" objType="control" minOc
<xsd:element viewType="boolean" viewName="リスク評価" group="基本的要素" objType="control" min
<xsd:element viewType="boolean" viewName="情報伝達" group="基本的要素" objType="control" minOc
<xsd:element viewType="boolean" viewName="統制活動" group="基本的要素" objType="control" minOc
<xsd:element viewType="boolean" viewName="監視活動" group="基本的要素" objType="control" minOc
                                  me="ＩＴへの対応" group="基本的要素" objType="control" m
                                  実施時期" group="" objType="control" minOccurs="0" nill
                                  年間発生件数" group="" objType="control" minOccurs="0"
                                  サンプル数" group="" objType="control" minOccurs="0" ni
                                  コントロールの実施部署" group="" objType="control" minO
<xsd:el          viewType="text" viewName="関連規定マニュアル等" group="" objType="control" minOcc
<xsd:element viewType="textarea" viewName="コメント" group="" objType="control" minOccurs="0"
```

element タグ 1 つが各カラムに該当する。属性情報はこの element タグごとに追記する。

＜画面での表示＞

コントロール編集
表示名
表示タイプ（text）
編集内容を設定し、OKボタンを押下して下さい。
No
A01
コントロール
【コントロール01　見積金額決定-値引きの承認】
営業責任者／BPG責任者は、営業担当者が起案した値引
グループ
項目
CTRLタイプ1
予防
CTRLタイプ1
発見
CTRLタイプ2
マニュアル
CTRLタイプ2
システム
主従
主的
主従
従的
基本的要素
統制環境
基本的要素
リスク評価
基本的要素
情報伝達
基本的要素
統制活動
基本的要素
ITへの対応
実施時期
随時
年間発生件数
サンプル数
コントロールの実施部署
営業本部
関連規定マニュアル等
販売規程利益管理規程
関連法制度
コメント
ここでは、毎月得意先ごとの製品別売価を決定し、
このコントロールは、マスターに登録すべき販売単価
表示タイプ（textarea）
グループ名
表示タイプ（boolean）
OK
キャンセル

■フォーマット定義（XMLスキーマ）の新規作成

フォーマットRCM一覧.xls を変更し、新たにフォーマット定義（XMLスキーマ）を作成する手順について説明します。

（１）フォーマットの変更

今回は、カラムを２つ追加、１つ削除のケースで修正を行います。

フォーマットRCM一覧.xls を開き以下の修正を行います。

1 カラム[不正防止（コンプライアンス）]の右に、エクセルの挿入を使ってカラム[重要科目・現金]を追加します。

2 カラム[監視活動]を削除

3 カラム[関連規程　マニュアル等]の後に、カラム[関連法制度]を挿入

（2）XMLスキーマ（フォーマット定義）生成

XMLスキーマ（フォーマット定義）を生成するには変換したいデータの範囲を指定し、Excelの機能を使いXMLスキーマを生成します。

1 メニュー ＞ XML Tools ＞ Convert Range to XML List を選択し、[Convert Range to XML List]画面を表示。

2 [Convert Range to XML List]画面の範囲指定(-)ボタンをクリックし、範囲指定バーを表示する。

3 XMLデータの範囲を指定(項目名は含めない)し、範囲指定バーの決定ボタンをクリック。

4 [Convert Range to XML List]画面に指定範囲が設定されるので、「No, first row contains data」を選択し、OKボタンをクリック。

5 新規に項目名の行が追加されたことを確認し、元の項目名の行をコピーする。
コピー後、元の項目名の行は削除し保存する。

6 メニュー ＞XML Tools＞Create XSD files for the XML Schema at the active cell を選択し、フォーマット定義（XMLスキーマ）を生成。

文字コード（UTF-8）を選択し任意のファイル名(import_schema.xsd)で保存する。

```
ファイル(F)  編集(E)  書式(O)  表示(V)  ヘルプ(H)
<?xml version='1.0' encoding='UTF-16'?>
<!-- Created from XmlMap.Name: Root_対応付け2 -->
<!-- XmlMap.DataBinding.SourceUrl:  -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:element nillable="true" name="Root">
                <xsd:complexType>
                        <xsd:sequence minOccurs="0">
                                <xsd:element minOccurs="0" maxOccurs="unbounded" nillable="true" name="Row" form="unqualified">
                                        <xsd:complexType>
                                                <xsd:sequence minOccurs="0">
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column1" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column2" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column3" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column4" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column5" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column6" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column7" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column8" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column9" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column10" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column11" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column12" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column13" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column14" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column15" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column16" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column17" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column18" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column19" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column20" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column21" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column22" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column23" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column24" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column25" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column26" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column27" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column28" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column29" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column30" form="unqualified"/>
                                                        <xsd:element minOccurs="0" nillable="true" type="xsd:string" name="Column31" form="unqualified"/>
                                                </xsd:sequence>
                                        </xsd:complexType>
                                </xsd:element>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
</xsd:schema>
```

xmlタグのencoding を、UTF-8に設定する。

```
<?xml version='1.0' encoding='UTF-8'?>
<!-- Created from XmlMap.Name: Root_対応付け2 -->
<!-- XmlMap.DataBinding.SourceUrl:  -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

（3）フォーマット定義（スキーマ）の属性情報付加

生成されたフォーマット定義（スキーマ）は、そのままではRCM機能では認識されませんので、いくつかの属性情報を付加する必要があります

必要な属性情報については、フォーマット定義（XMLスキーマ）の編集を参照ください。

（4）スキーマインポート

新規作成されたXMLスキーマを BPM|Designer にインポートします。

1 BPM|Designer のメニュー＞RCM＞スキーマインポートを選択し、[スキーマインポート]画面を表示。

2 [スキーマインポート]画面の参照ボタンをクリックし、ファイル選択画面を表示。

3 インポートファイルを指定する。

4 OKボタンをクリックし、インポート処理を実行。

5 警告メッセージ表示。OKボタンをクリックすると処理を続行。

- 初期化されるとリスク・コントロールデータはクリアされます。

6 インポートが終了した旨のメッセージを表示。

（5）XMLデータインポート

新規にXMLスキーマが登録されると、インポートするXMLデータはXMLスキーマに合わせたものが必要になります。

「2.6.2.1 インポート」のインポートデータの作成を参照し、新規にXMLデータの作成を行いインポートを行って下さい。

2.7 進捗通知

2.7.1 概要

進捗通知は、業務プロセス内に処理を行ったユーザがプロセスの進捗状況、処理者情報を確認する為に、進捗を通知するタスク（以下「進捗通知タスク」）を配置しWebサービスを利用してタスクの状態・処理ユーザを登録・更新する機能です。
この機能を利用して進捗通知タスクの登録を行った業務プロセスは、処理ユーザが[進捗一覧]画面で業務案件の進捗状態を確認できるようになります。

■進捗状況の管理の単位

進捗状況の管理の単位は、プロセス（プール）単位（BPM|Serverがプロセスインスタンス ＩＤを割り当てる単位）となります。
プロセスから別プロセスの呼び出しを行っている場合は、それぞれのプロセスにプロセスインスタンス ＩＤが割り当てられます。

■提供Webサービス

進捗通知機能では、プロセス開始時にプロセス内に存在する全進捗通知タスクを登録し、以降はステータスの更新と処理ユーザの登録を行う「事前登録パターン」と、処理が到達した時点で毎回進捗通知タスクの登録と処理ユーザの登録を行う「都度登録パターン」の二つのWebサービスを提供します。

各サービスの詳細は、以降の章[2.7.2 事前登録パターン][2.7.3 都度登録パターン]で詳しく説明します。

2.7.2 事前登録パターン

事前登録パターンとは、プロセス開始時にプロセス内で使用する全進捗通知タスクを事前に登録し、プロセスが進捗通知タスクに到達した時点で、該当タスクの状態を[未処理]から[完了]へ更新するパターンの進捗通知機能です。

<図 1 業務フロー例>

- [進捗通知タスク登録]に[進捗通知タスク1]の更新処理を含めることができます。
  詳細は登録処理のMapping「進捗タスクフラグ」の項目を参照してください。

このパターンのサービスを利用したプロセスの場合、[進捗一覧-詳細]画面には全進捗通知タスクが表示され、未到達のタスクの状態は[未処理]、到達済のタスクは[完了]と表示されることになります

進捗通知タスク２が完了した時点の[進捗一覧-詳細]画面

処理ID	処理名	ユーザ	処理日	状態
1	見積処理	青柳辰巳	2008/05/20 19:35	完了
2	受注処理	円山益男	2008/05/25 11:37	完了
3	契約締結処理			未処理
4	支出審査処理			未処理

事前登録パターンのサービスでは、二つのオペレーションを提供しています。

■進捗通知タスク登録処理（registAllNoticeTasks）
プロセスで使用する全進捗通知タスクとプロセスの基本情報の登録処理を行います。

■進捗通知タスク更新処理（updateNoticeTask）
進捗通知タスクのステータスの更新と処理ユーザの登録処理を行います。

<図 1 業務フローのダイアグラム例>

2.7.2.1 タスクの配置

進捗通知タスクは基本的にプロセス内で進捗通知を行いたい場所に任意で配置が可能ですが、以下の場所への配置は必須となります。
（配置しなかった場合は、進捗一覧画面に正しく進捗情報が表示されない可能性があります。）

① プロセス開始直後
　⇒　プロセスの基本情報と全進捗通知タスクの登録を行います。
②「xForm」または「Relation Form」を使用したヒューマンタスクの後
　⇒　処理ユーザを登録します。
③ プロセスの終了前
　⇒　プロセスステータスを完了にします。

2.7.2.2 WSDLファイルの取り込み

進捗通知機能を利用する場合は、最初に事前登録パターンのWebサービスのWSDLファイルの取り込みを行う必要があります。
以降はiWP/aFWのIPアドレスを<%HTTP_ADRESS%>、ポート番号を<%HTTP_PORT%>と略します。

1 WSDLファイルの取得

以下のURLからWSDLファイルを取得して、ビジネス・プロセス・プロジェクト（以下プロジェクト）内に配置します。
ファイル名称は任意ですが、拡張子はwsdlとしてください。

```
http:// <%HTTP_ADDRESS%>:<%HTTP_PORT%>/imart/services/NoticeProgressPreRegistServic
e?wsdl
```

2 XSDファイルの取得

以下のURLからXmlスキーマファイルを取得します。ファイルの名称は[noticeProgressTask.xsd]とし、プロジェクト内に配置します。

```
http:// <%HTTP_ADDRESS%>:<%HTTP_PORT%>/imart/services/ NoticeProgressPreRegistServic
e?xsd=noticeProgressTask.xsd
```

3 WSDLファイルの編集

手順1で取得したWSDLファイル内のスキーマロケーションを以下のように変更します。

行：16行目

【変更前】

```
<xs:import namespace="http://notice_progress.webservice.bpms.system.intra_mart.co.jp/t
asks" schemaLocation=" NoticeProgressPreRegistService?xsd=noticeProgressTask.xsd"/>
```

⇒

【変更後】

```
<xs:import namespace="http://notice_progress.webservice.bpms.system.intra_mart.co.jp/t
asks" schemaLocation="noticeProgressTask.xsd"/>
```

2.7.2.3 進捗通知タスクのスキーマ定義

事前登録パターンの進捗通知タスクの定義は、2.7.2.1で取得した[noticeProgressTask.xsd]ファイルで行います。

ファイルをXml Schema Editerで開きソースを表示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://notice_progress.webservice.bpms.s
        <element name = "noticeProgressTask">
        <complexType>
            <sequence>
                <element name="noticeTaskId_1" type="string" minOccurs="1" maxOccurs="1" default="1"/>
                <element name="noticeTaskId_2" type="string" minOccurs="1" maxOccurs="1" default="2"/>
                <element name="noticeTaskId_3" type="string" minOccurs="1" maxOccurs="1" default="3"/>
                <element name="noticeTaskId_4" type="string" minOccurs="1" maxOccurs="1" default="4"/>
                <element name="noticeTaskId_5" type="string" minOccurs="1" maxOccurs="1" default="5"/>

                <element name="noticeTaskName_1" type="string" minOccurs="1" maxOccurs="1" default=""/>
                <element name="noticeTaskName_2" type="string" minOccurs="1" maxOccurs="1" default=""/>
                <element name="noticeTaskName_3" type="string" minOccurs="1" maxOccurs="1" default=""/>
                <element name="noticeTaskName_4" type="string" minOccurs="1" maxOccurs="1" default=""/>
                <element name="noticeTaskName_5" type="string" minOccurs="1" maxOccurs="1" default=""/>
            </sequence>
        </complexType>
        </element>
</schema>
```

<編集前noticeProgressTask.xsd >

一つの進捗通知タスク情報は以下の二つの要素から構成されています。

設定する進捗通知タスク分、各要素の定義を行います。

noticeTaskId_×	進捗通知ID default属性に設定した値が、[進捗一覧-詳細]画面の処理IDとして表示されます。また、ステータス更新時のキーとなります。 デフォルトでは1から連番で数字を設定していますが、任意の値に変更が可能です。 変更する場合は、プロセス内で一意になるIDを設定してください。 [進捗一覧-詳細]画面では、**処理IDの昇順**で進捗通知タスク情報を表示します。
noticeTaskName_×	進捗通知タスク名 default属性に定義した値が、[進捗一覧-詳細]画面の処理名として表示されます。

設定するタスク数がデフォルトの5個以下の場合は、不要な要素を削除してください。また6個以上になる場合は、noticeTaskId_6,noticeTaskName_6・・・・という形式で要素を追加してください。

- noticeTaskId_×,noticeTaskName_×に設定する数字が同じものが同一タスク情報と見なされます。
- 必ず数字のペアで連番になるように定義を行ってください。

■ <図1 業務フロー>の進捗通知タスクのスキーマ定義編集例

手順1．図1の業務フローには進捗通知タスクは4つしか存在しない為、不要タスクnoticeTaskId_5，noticeTaskName_5を削除

手順2．noticeTaskName_1～4のdefault属性に進捗通知タスク名を設定

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://notice_progress.webservice.bpms.system.intr
        <element name = "noticeProgressTask">
        <complexType>
            <sequence>
                <element name="noticeTaskId_1" type="string" minOccurs="1" maxOccurs="1" default="1"/>
                <element name="noticeTaskId_2" type="string" minOccurs="1" maxOccurs="1" default="2"/>
                <element name="noticeTaskId_3" type="string" minOccurs="1" maxOccurs="1" default="3"/>
                <element name="noticeTaskId_4" type="string" minOccurs="1" maxOccurs="1" default="4"/>

                <element name="noticeTaskName_1" type="string" minOccurs="1" maxOccurs="1" default="見積処理"/>
                <element name="noticeTaskName_2" type="string" minOccurs="1" maxOccurs="1" default="受注処理"/>
                <element name="noticeTaskName_3" type="string" minOccurs="1" maxOccurs="1" default="契約締結処理"/>
                <element name="noticeTaskName_4" type="string" minOccurs="1" maxOccurs="1" default="支出審査処理"/>
            </sequence>
        </complexType>
        </element>
</schema>
```

<編集後noticeProgressTask.xsd>

2.7.2.4 サービスの操作設定

■ 登録処理

登録オペレーション[registAllNoticeTasks]を対象タスクにドラッグ＆ドロップします。

■ 更新処理

更新オペレーション[updateNoticeTask]を対象タスクにドラッグ＆ドロップします。

設定方法の詳細は　本ドキュメントの「2.3 外部Webサービス連携」を参照して下さい。

2.7.2.5 マッピング

■ 登録処理

登録処理では、intra-mart認証情報、プロセス基本情報のマッピング定義を行います。
下図は、進捗通知タスク登録と進捗通知タスク１を一つのタスクに設定した場合のマッピング例です。

<登録処理 Mapping例>

【リクエストメッセージ】

1．Intra-mart認証情報　[wsUserInfo]

authTpye	認証タイプ	デフォルトで[BPMS]が設定済
password	intra-mart Webサービス連携用ダイジェスト	2.3.6 イントラマートへのログインを参照してください。

2．プロセス基本情報　[processInfo]

businessProcessName	業務プロセス名	[進捗一覧]画面のプロセス定義名に表示されます。
processName	案件名	[進捗一覧]画面の案件名に表示されます。
processInstanceId	プロセスインスタンスID	「$ode:pid」 固定 [進捗一覧]画面の案件番号に表示されます。
startUser	開始ユーザ	[進捗一覧-詳細]画面のユーザに表示されます。 このタスクを進捗通知タスクとして扱う場合に設定します。処理ユーザが存在しない場合は設定不要です。 (1) ヒューマンタスクの後に処理ユーザを登録する場合 ヒューマンタスクからのリクエストメッセージのtaskOutput要素>output　>@user属性　とMapping (2) 外部Webサービスの処理ユーザを登録する場合 2.7.4 Webサービスの処理ユーザの登録を参照してください。
startDate	プロセス開始日	date型(形式:yyyy-mm-ddT24h:mi:ss) 省略可　省略した場合は、システム日時が設定されます。 [進捗一覧]画面の開始日に表示されます。
noticeTaskFlg	進捗タスクフラグ	このタスクを進捗通知タスクとして扱う場合は"1"を設定します。 その場合は、"noticeTaskId_1"のタスクステータスが完了となります。 進捗通知タスクの登録のみを行い、進捗通知タスクとして扱わない場合は、Mapping不要です。

3．進捗通知情報　[NoticeProgressTask]

外部ファイル[noticeProgressTask.xsd]で定義済の為、Mappingは不要です。

■ 更新処理

更新処理では、intra-mart認証情報、タスク情報のマッピング定義を行います。

<更新処理 Mapping例>

【リクエストメッセージ】

1．Intra-mart認証情報 [wsUserInfo]

authTpye	認証タイプ	デフォルトで[BPMS]が設定済
password	intra-mart Webサービス連携用ダイジェスト	2.3.6 イントラマートへのログインを参照してください。

2．タスク情報 [taskInfo]

processInstanceId	プロセスインスタンスID	「$ode:pid」固定
noticeTasikId	進捗通知タスクID	[noticeProgressTask.xsd]ファイルで定義されている該当の進捗通知タスクIDを設定します。

user	処理ユーザ	[進捗一覧-詳細]画面のユーザに表示されます。 (1) ヒューマンタスクの後に処理ユーザを登録する場合 ヒューマンタスクからのリクエストメッセージのtaskOutput要素>output >@user属性 とMapping (2) Webサービスの処理ユーザを登録する場合 2.7.4 Webサービスの処理ユーザの登録を参照してください。
tranDate	処理日	date型(形式:yyyy-mm-ddT24h:mi:ss) 省略可 省略した場合は、システム日時が設定されます。 [進捗一覧-詳細]画面の処理日に表示されます。
endTaskFlg	最終タスクフラグ	このタスクがプロセス内の最終のタスクの場合は「1」を設定します。 最終タスク以外はMapping不要です。

2.7.3 都度登録パターン

都度登録パターンとは、プロセスが進捗通知タスクに到達した時点で、毎回進捗通知タスクを登録していくパターンの進捗通知機能です。
複雑なプロセスで、事前にプロセス内で使用する全進捗通知タスクを登録するのが難しい場合などで使用します。

<図2 業務フロー例>

このパターンのサービスを利用したプロセスの場合、[進捗一覧-詳細]画面には処理が到達済のタスクのみ状態が[完了]で表示されることになります。

進捗通知タスク２が完了した時点の[進捗一覧-詳細]画面

処理ID	処理名	ユーザ	処理日	状態
1	見積処理	青柳辰巳	2008/05/20 19:35	完了
2	受注処理	円山益男	2008/05/25 11:37	完了

都度登録パターンのサービスでは、登録処理のオペレーションのみを提供しています。

■進捗通知タスク登録処理[registNoticeTask]
進捗通知タスクの登録、処理ユーザの登録、プロセス基本情報の登録(最初のタスクのみ)を行います。

<図2 業務フローのダイアグラム例>

2.7.3.1 タスクの配置

進捗通知タスクは基本的にプロセス内で進捗通知を行いたい場所に任意で配置が可能ですが、以下の場所への配置は必須となります。
（配置しなかった場合は、進捗一覧画面に正しく進捗情報が表示されない可能性があります。）

① プロセス開始直後
　⇒　プロセスの基本情報を登録します。
②「xForm」または「Relation Form」を使用したヒューマンタスクの後
　⇒　処理ユーザを登録します。
③ プロセスの終了前
　⇒　プロセスステータスを完了にします。

2.7.3.2 WSDLファイルの取り込み

以下のURLからWSDLファイルを取得して、プロジェクト内に配置します。
ファイル名称は任意ですが、拡張子はwsdlとしてください。

```
http:// <%HTTP_ADDRESS%>:<%HTTP_PORT%>/imart/services/NoticeProgressEachTimeRegistS
ervice?wsdl
```

2.7.3.3 サービスの操作設定

登録オペレーション[registNoticeTask]を対象タスクにドラッグ＆ドロップします。

設定方法の詳細は　本ドキュメントの「2.3 外部Webサービス連携」を参照して下さい。

2.7.3.4 マッピング

■ 登録処理

1番目の登録処理では、intra-mart認証情報、プロセス基本情報、タスク情報のマッピング定義を行います。
2番目以降の登録処理では、ntra-mart認証情報、タスク情報のマッピング定義を行います。

<1番目の登録処理 Mapping例>

<2番目以降の登録処理 Mapping例>

【リクエストメッセージ】

1．Intra-mart認証情報 [wsUserInfo]

autyTpye	認証タイプ	デフォルトで[BPMS]が設定済
password	intra-mart Web サービス連携用ダイジェスト	2.3.6 イントラマートへのログインを参照してください。

2．プロセス基本情報　[processInfo]　**プロセスインスタンスID以外は１番目の進捗通知タスクのみ要設定**

businessProcessName	業務プロセス名	[進捗一覧]画面のプロセス定義名に表示されます。
processName	案件名	[進捗一覧]画面の案件名に表示されます。
processInstanceId	プロセスインスタンスID	「$ode:pid」 固定 進捗一覧画面の案件番号に表示されます。
startDate	プロセス開始日	date型（形式：yyyy-mm-ddT24h:mi:ss） 省略可　省略した場合は、システム日時が設定されます。 [進捗一覧]画面の開始日に表示されます。

3．タスク情報　[taskInfo]

noticeTaskId	進捗通知タスクID	任意　省略した場合は一意の値（[4桁の連番]＋ユニークなID）が自動的に振られます。 [進捗一覧-詳細]画面の処理IDに表示されます。 [進捗一覧-詳細]画面では、**処理IDの昇順**で進捗通知タスク情報を表示します。
noticeTaskName	進捗通知タスク	[進捗一覧-詳細]画面の処理名に表示されます。
user	処理ユーザ	[進捗一覧-詳細]のユーザに表示されます。 (1) ヒューマンタスクの後に処理ユーザを登録する場合 ヒューマンタスクからのリクエストメッセージのtaskOutput要素>output　>@user属性　とMapping (2) Webサービスの処理ユーザを登録する場合 2.7.4 Webサービスの処理ユーザの登録を参照してください。
tranDate	処理日	date型（形式：yyyy-mm-ddT24h:mi:ss） 省略可　省略した場合は、システム日時が設定されます。 [進捗一覧-詳細]画面の処理日に表示されます。
endTaskFlg	最終タスクフラグ	このタスクがプロセス内の最終のタスクの場合は「1」を設定します。 最終タスク以外はMapping不要です。

2.7.4 Webサービスの処理ユーザの登録

[進捗一覧]画面に表示される業務案件は、基本的にログインユーザが業務案件内のヒューマンタスクで処理を行った案件になります。
ここでは、業務プロセス内で外部Webサービス連携を行っていた場合、外部Webサービス（ワークフローなど）で処理を行ったユーザが、進捗状況を確認できるようにする為の設定について説明します。

■ Webサービス側の設定

プロセスに渡す情報の中に、登録したい処理ユーザの情報を以下の構造・要素名で格納します。

noticeProgressUserInfo		処理ユーザ情報	
	tranDate	処理日	date型（形式：yyyy-mm-ddT24h:mi:ss） 省略可　省略した場合は、システム日時が設定されます。 [進捗一覧-詳細]画面の処理日に表示されます。
	userCd	処理ユーザCD	[進捗一覧-詳細]画面のユーザに表示されます。

<スキーマ構造例>

処理ユーザ情報[noticeProgressUserInfo]の要素数に制限はありません。
複数の要素を設定することで、複数の処理ユーザを登録することが可能となります。

■ マッピング

マッピング元：Webサービスからのリクエストまたはレスポンスの
処理ユーザ情報[noticeProgressUserInfo]の親要素
（上記の例の場合は、[webServiceUserInfo]になります。）

マッピング先：進捗通知Webサービスのリクエストのタスク情報[taskinfo] 内の[user]要素

<Mapping例>

2.7.5 注意点

業務プロセスで進捗通知サービスを利用する場合、プロセス内で同一サービス・同一オペレーションを進捗通知タスクの数分、複数回呼び出すことになります。

- 事前登録パターンの更新処理
- 都度登録パターンの登録処理

このときリクエストメッセージが同一スコープにある場合は、最初のMappingで各要素に設定した値は、次のMapping時にも引き継がれています。
適切に値の設定・または空文字の設定による値の初期化を行ってください。

例）リクエストメッセージが同一スコープにある場合の登録処理
一つ目の進捗通知タスクの登録処理時に、処理ユーザCDに「aoyagi」を設定し、二つ目の進捗通知タスクの登録処理時には、処理ユーザCDに何も設定しなかった場合は、処理ユーザCDには最初に設定した「aoyagi」が設定されている状態のままになっており、二つ目の進捗通知タスクの処理ユーザにも「aoyagi」が登録されることになります。

- Intra-mart認証情報については、生成した認証情報は1度しか使えないため、進捗通知サービスを呼ぶ際には毎回Mappingにてintra-mart認証情報の設定を行ってください。

2.7.6 進捗一覧画面

[進捗一覧]画面には、ログインユーザが処理を行った業務案件の中で、進捗通知機能を利用して進捗通知タスクの登録を行った業務案件が表示されます。
この画面では、処理を行ったプロセスの進捗状態、処理状況の確認が行えます。
詳細は、「BPM ユーザ操作ガイド　1.7 進捗一覧」を参照して下さい。

【進捗一覧画面イメージ】

【進捗一覧詳細画面イメージ】（事前登録パターン）

【進捗一覧詳細画面イメージ】（都度登録パターン）

intra-mart WebPlatform/AppFramework

第3章　参考

3.1 リファレンス

3.1.1 データベースのクリーニング

実行が終了したプロセスインスタンスの情報はデータベースに蓄積されていきます。
以下のSQLをデータベースに発行することで、実行が完了したインスタンスを削除することができます。

<%BPM|Serverインストールディレクトリ%>/database/oracle_clean.sql

- 必ずデータベースのバックアップを行ってから実行するようにして下さい

3.1.2 プロセス変数の作成

業務フロー作成の過程で、プール内で使用可能な変数が必要になる場合があります。
例としてループタスクの終了条件に、カウンタ変数を使用したいケース等です。
前提として、プロセス変数を定義するためにXMLスキーマをプロジェクト内に作成しておきます。

1 プロセス変数の選択

プロセスエクスプローラでXMLスキーマを展開して、プロセス変数として使用する要素を表示します。

XMLスキーマの中の要素をプールへドラッグ＆ドロップします。

2 変数の初期化

編集初期化用のタスク等を作成して、マッパーを使ってプロセス変数の初期化を行います。

3 変数の利用

初期化されたプロセス変数は、後ろのタスク等で自由に読み書き可能になります。

以下の例では、ループタスクの終了条件として使用しています。

3.1.3 Escalateタスクの利用

エスカレートタスクは、他の未処理タスク等から処理を委譲する目的で使用します。
エスカレートする際に、委譲元から委譲先へマッパーを使用してタスクIDをマッピングします。

以下の例では、タイマー中間を使用して、未処理タスクで一定時間経過した場合はエスカレートタスクへ処理を委譲しています。

3.1.4 処理タスクの期限の設定

処理タスクを配置した場合、依頼延期期間及び処理有効期限の設定を行う事が可能です。
依頼延期期間及び処理有効期限の設定はそれぞれマッパーを使用して設定を行います。

3.1.4.1 依頼延期期間の設定

依頼遅延期間を設定する事により、未処理タスクとして登録される処理を遅らせる事が出来ます。
その為、即時に処理を依頼したくない場合、また将来的に依頼が必要となる処理において有効です。
依頼遅延期間の指定は、処理タスクに定義された[処理タスクデータ型] → [taskMetaData] → [scheduledActions] → [deferActivation]項目配下に指定を行います。

until	特定の日時まで延期を行いたい場合に使用してください。 値は日付をあらわすdateTime型でマッピングを行ってください。 指定された日付がシステム日付より過去の場合は即時に処理タスクが作成されます。
for	特定の期間延期を行いたい場合に使用してください。 値は日時の期間をあわらすduration型でマッピングを行ってください。 指定された期間が経過した後、処理タスクが作成されます。

3.1.4.2 処理有効期間の設定

処理有効期間を設定する事により、未処理タスクとして登録される処理の有効期限を設定する事が出来ます。
有効期限の指定は、処理タスクに定義された[処理タスクデータ型] → [taskMetaData] → [scheduledActions] → [expiration]項目配下に指定を行います。

until	特定の日時を有効期限として指定する場合に使用してください。 値は日付をあらわすdateTime型でマッピングを行います。 指定された日付がシステム日付より過去の場合は即時に有効期限切れとなります。
for	有効期限を日時ではなく、処理タスクが作成されてからの期間として設定したい場合に使用してください。 値は日時の期間をあわらすduration型でマッピングを行います。 指定された期間が経過する間に完了処理が行われなかった場合有効期限切れとなります。

3.1.4.3 有効期限の切れたタスクの扱い

有効期限切れとなった処理タスクはそのタスクで完了しませんのでご注意ください。
有効期限切れのタスクから有効期限切れを表すメッセージが送られ、次の処理へ進みます。
その為、有効期限を設定した処理タスクは処理メッセージを受け取った後、有効期限切れであるかの判断を行う等の処理が必要となります。
有効期限切れとなった場合のメッセージは、[処理タスクメッセージ型] → [status]項目に"DeadlineReached"文字列が格納されます。
また、処理画面において入力されるべき値等は全て格納されていない状態でメッセージが送信される為ご注意ください。

3.1.5 カスタムXPath関数の作成

[マッパー]ビューにおいて利用するxPath関数は、独自のxPath関数を作成する事が出来ます。
カスタムxPath関数を定義する事でユニークなIDを生成する、日付の変換を行うといった処理が組み込めます。
カスタムxPath関数は全てJavaにより実装を行う必要があり、また実装したJavaクラスの情報を記したxpathファイルを作成する必要があります。
本項ではカスタムxPath関数の作成方法を記します。

3.1.5.1 Javaクラスの作成

カスタムxPath関数から呼出されるクラスを作成します。
ここでは例として日付を引数として受け取り、引数の日付に対する月末日を返却するクラスを作成します。

1 Javaクラスの作成を行います。

```
package xpath.functions;

import java.util.Calendar;
import java.util.Date;

/**
 * カスタム xPath 関数クラス.
 *
 */
public class DateFunctions {

	/**
	 * 月末日を取得します.
	 *
	 * @param date 日付
	 * @return 月末日
	 */
	public static Date getLast(Date date) {

		// Calendar に変換.
		Calendar calendar = Calendar.getInstance();
		calendar.setTime(date);

		// 月末日をセット
		calendar.set(Calendar.DATE, calendar.getActualMaximum(Calendar.DATE));

		return calendar.getTime();
	}
}
```

- 必ずパッケージ名を指定してください。
- カスタムxPath関数として作成するメソッドには必ずstatic修飾子が必要です。

2 作成したJavaクラスをBPM|Server上のクラスパスの通っている場所に配置します。
作成したクラスをjarファイルとしてアーカイブ化後、%BPM|Serverインストールフォルダ%/lib配下に配置します。

- jarファイルの作成は使用するIDE、またはjarコマンドを利用して作成を行ってください。

Javaクラスを作成するに当たり、利用可能な引数は下記の通りとなります。
BPM|Designer上の型とJavaクラスの型を一致させる必要がありますのでご注意下さい。

xs:boolean	boolean, java.lang.Boolen, java.lang.String
xs:dateTime, xs:date	java.util.Date, java.lang.String
xs:decimal	java.math.BigDecimal, java.lang.String
xs:double	double, java.lang.Double, java.lang.String
xs:duration	java.lang.String
xs:float	float, java.lang.Float
xs:integer	int, java.lang.Integer
xs:string	java.lang.String
xs:anyURI	java.net.URL, java.net.URL, java.lang.String

3.1.5.2 xpathファイルの作成

作成したxPath関数クラスの情報をxpathファイルとして記述します。

1 xpathファイルの作成を行います、テキストエディタ等で作成を行ってください。
ここでは例として先の説明で作成した月末日を算出するJavaクラスを利用する為の定義を記します。

```
<div1 id="Custom date functions" xmlns:cdf="java:xpath.functions.DateFunctions">
  <head>custom date functions</head>
  <div2 id="func-getLast">
    <head>cdf:getLast</head>
    <proto name="getLast" return-type="xs:dateTime" returnEmptyOk="no"
     returnSeq="no" returnVaries="no" isSchema="yes" isDatatype="no"
     isSpecial="no" isOp="no">
      <arg name="date" type="xs:dateTime" emptyOk="no"/>
    </proto>
    <p>引数の日付を元に月末日を算出します。</p>
  </div2>
</div1>
```

- xpathファイルを作成した場合、必ず文字エンコードをUTF-8で保存する必要がありますのでご注意下さい。

xPathファイルの記述形式

独自のxPath関数を作成する場合に必要なxpathファイルの記述形式は下記の形式となります。

```
<div1 id="{カスタム xPath 関数群 id}" xmlns:{プレフィクス}="java:{Java クラス} ">
  <head>{カスタム xPath 関数群説明}</head>
  <div2 id="{関数 id} ">
    <head>{プレフィクス}:{関数名} </head>
    <proto name="{関数名} " return-type="{戻り値 XML 型} "
      returnEmptyOk="{空戻り値許可} " returnSeq="no" returnVaries="no"
       isSchema="yes" isDatatype="no" isSpecial="no" isOp="no">
      <arg name="{引数名} " type="{引数 XML 型} " emptyOk="{空引数許可} "/>
      …
    </proto>
    <p>{関数説明} </p>
  </div2>
  …
</div1>
```

- div2要素は複数記述する事が可能です。
  1つのxpathファイルに複数のxPath関数を定義する場合に使用します。
- arg要素は複数記述する事が可能です。
  引数が複数存在する場合に使用します。
  また、引数が不要の場合、arg要素を記述する必要はありません。

カスタムxPath関数群id	カスタムxPath関数のカテゴリ名を指定します。 自由にカテゴリ名を指定する事が可能です。
プレフィクス	カスタムxPath関数で利用する名前空間のprefixを指定します。 ビジネス・プロセス・ダイアグラムで使用する名前空間と一致しないよう注意が必要です。
Javaクラス	作成したカスタムxPath関数クラスをパッケージ名からの完全名で指定してください。
カスタムxPath関数群説明	デザイナに表示されるカテゴリ名です。 自由にカテゴリ名を指定する事が可能です。 カスタムxPath関数群idと同様の内容で問題ありません。
関数id	カスタムxPath関数idを指定します。 通常、func-関数名形式で指定を行います。
関数名	作成したカスタムxPath関数クラスで用意したJavaメソッド名を指定してください。
戻り値XML型	関数の戻り値として扱うXMLスキーマ型を指定します。
空戻り値許可	戻り値が存在する場合"no"を指定してください。 実装したJavaメソッドがvoid型の場合"yes"を指定してください。
引数名	引数名を指定します。 通常、実装したJavaメソッド引数名に一致させます。
引数XML型	引数として受け渡すXMLスキーマ型を指定します。
空引数許可	引数が必須の場合"yes"必須で無い場合、または空文字等が受け渡される場合は"no"を指定します。
関数説明	BPM|Designer上に表示される説明を記述します。

3.1.5.3 カスタムxPath関数の利用

作成したカスタムxPath関数は、ビジネス・プロセス・プロジェクトに取込む事により利用可能となります。
作成したxpathファイルを取込む事により、[マッパー]ビューからの利用が可能となります。
xpathファイルの取込み方法、関数の呼び出し方法は「イントラマートへのログイン」を参照してください。

A

Activate 46
authTpye 226

B

<bpel:propertyAlias>を作成 138
<bpel:property>を作成 133
.bpm 14
BPM|Designer 2
bpmadmin 44
BPMN 2
bpmprocessmanager 44
BPM 管理者 44
BPM プロセス管理者 44
businessProcessName 227, 233

C

Catch 109
Completed 46
CorrelationSet を生成 135

D

DeadlineReached 242
declarations 134
deferActivation 241
Delete 47
Delete All 47
Deploy 46
Designer 6

E

endTaskFlg 229, 233
Escalate タスク 240
expiration 242

F

Failed 46
Failure 46
Failures 47
Fault on failure 108
for 241, 242
Form Editor 9, 25
formUrl 76

G

GroupID 66

I

ID 66
imart:getDigest 118
imGroup 76
imId 76
imUser 76
In Progress 46
Incoming message variable name 131, 148
initiate 142, 144, 154, 156
input 73
intra-mart フォーム・リレーション エディタ 69
Invoke 47
iWP/iAF 2

J

JavaEE 開発モデル 72, 79

L

LastActive 47
Lifecycle 46

M

Modeling 6

N

NoticeProgressEachTimeRegistService?wsdl 231
NoticeProgressPreRegistService?wsdl 223
NoticeProgressTask 227
noticeProgressTask.xsd 223
noticeTasikId 228
noticeTaskFlg 227
noticeTaskId 233
noticeTaskId_× 225
noticeTaskName 233
noticeTaskName_× 225

O

$ode:pid 39
oracle_clean.sql 238
Outgoing message variable name 148
Outline 7
output 73

P

Palette 6, 15
participantToken 76
password 66, 226
pattern 154
Process Execution 8
processInfo 227, 233
processInstanceId 227, 228, 233
processName 227, 233

R

RCM 10, 161

RCM 出力 200
registAllNoticeTasks.....222, 226
registNoticeTask 230, 231
Response fault name 127
Resume..... 47
Retire..... 46

S

scheduledActions.....241, 242
Start..... 46
startDate 227, 233
Started 47
startUser 227
State 47
Suspend..... 47
Suspended 46

T

taskId..... 76
taskInfo 228, 233
TaskManagementProcess..... 48
taskMetaData 40, 241, 242
Terminate 47
Terminated 46
tranDate 229, 233

U

Undeploy..... 46
until 241, 242
updateNoticeTask 222, 226
user 76, 229, 233

W

WebService エンドポイント 123
WebService 連携時の例外を補足する..... 107
WebService の検索 88
WebService の取込 82
WebService プロセスの WSDL 123
WebService 呼出時のパラメータ設定..... 106
WebService 連携..... 82
WebService 連携用タスクの配置 89
Workflow 8, 51, 67
WSDL ファイル 223, 231
WSDL ファイルのインポート 83
WSDL ファイルの参照 87
WSDL ファイルパス..... 123
wsUserInfo 117, 226, 228, 232

X

XForm 9, 22, 28
.xform の作成..... 23
XML スキーマ 32, 121, 161, 211, 238
Xml スキーマファイル..... 223
XML データ 161
XML データインポート..... 168
XML データエクスポート 206
xPath 関数..... 243
XPath 関数 37, 116
xpath ファイルのインポート 112
xpath ファイルの作成 245

あ

アイテム 26
アクティビティ 2

い

委譲タスク 28
イベント 3, 16
インスタンス一覧画面 47
イントラマートへのログイン 111

か

外部ページ 72
カスタム XPath ファンクション 116
画面名..... 71
管理コンソール 44
管理コンソールの表示 45

く

クラス名 73
グループシェープ 107

け

ゲートウェイ..... 3, 20
ゲートウェイの結合..... 21
ゲートウェイの分岐条件 38
権限設定時の注意点..... 62
権限の種類 50
権限の設定 44, 50
権限の優先順..... 62
検索基準日の設定 67
検索（コントロール） 172
検索（リスク） 170

こ

コード（権限） 52
コピー（コントロール） 186
コピー（リスク） 179
コントロール..... 161
コンポーネント 26

さ

サーバ設定 42
最小要素数 75
最大要素数 75
削除（権限） 52
削除（属性/要素） 71
削除（コントロール） 185
削除（リスク） 177
サブプロセス..... 107, 110

し

シーケンスフロー 3, 19
出力値 71
出力パッケージ入力 80
種別（権限） 52
種類 74, 75
初期処理タスク 28
処理タスクの期限の設定 241
進捗通知 221
進捗通知タスク更新処理 222
進捗通知タスク登録処理 222, 230

す

スキーマエクスポート 209
スクリプト開発モデル 72, 79

せ

接続先ホスト 66
説明（権限） 51

そ

相関セットキー 128, 129, 132, 146, 149
組織検索 51, 54
その他開発モデル 72

た

ダイアグラム 2, 14, 28
タスク 2, 17, 197

つ

追加（コントロール） 181
追加（属性） 71
追加（要素） 71
追加（リスク） 173
通知タスク 28

て

データ・エディタ 132
データ構造 73
データの持ち方 34
データマッピング 35
テクニカル・ネーム 19, 26, 131, 148
デフォルト値 74, 75
デプロイ 41

な

名前 74, 75

に

入力値 71
入力値/出力値の設定 73
認証情報の設定 65

は

パス 71
パスワード保管 66
パブリックグループ検索 52, 54

ひ

ビジネス・プロセス・ダイアグラム 2, 6
ビジネス・プロセス・プロジェクト 11
非同期処理プロセス 128
ヒューマンタスク 22, 55, 62, 73, 81
ヒューマンタスクの種類 28
ビルトインデータ 39

ふ

プール 2, 27, 55, 59, 62
フォーマット定義 161
フォーム・リレーション エディタ 71
フォーム・エディタ 9
フォーム名前空間 71
プロジェクトのインポート/エクスポート 12
プロセス 2
プロセス・エクスプローラ 7
プロセス一覧画面 46
プロセスインスタンス ID 39, 157, 221
プロセスの WebService 化 120
プロセス変数 238
プロパティー 8, 10, 51

へ

ページ種別 71
編集（コントロール） 182
編集（リスク） 175
変数の初期化 239
変数の利用 239

ま

マッパー 7, 33, 239, 243
マッピング 33

み

未処理タスク 28

め

名称（権限） 52
メタデータ 40
メッセージコネクション 29, 121, 126
メッセージのデータフォーマット 32
メッセージフロー 3, 29, 32

も

モデル生成 71, 78

ゆ

ユーザ検索 51, 53

ら

ラベル 19, 26

り

リスク 161
リモートサービスのインポート 84
リレーション 161, 177, 185, 188
リレーション・フォーム 22, 68

る

ループタスク 20, 239
ループの終了条件 39

れ

例外メッセージの設定 126
連携画面の設定 72

ろ

ロール検索 51, 53
ログアウト 66
ログイン 66

わ

ワークフロー設定 42
ワークフローパレット 10, 25

intra-mart WebPlatform/AppFramework Ver.7.0

2008/09/30 第２版

BPM|Designer 操作ガイド

株式会社 NTT データ イントラマート
〒107-0052 東京都港区赤坂 2-17-22 赤坂ツインタワー本館 3 階
TEL(03)5549-2821 FAX(03)5549-2816
E-mail : info@intra-mart.jp
ホームページ : http://www.intra-mart.jp
Copyright 2000-2008 株式会社 NTT データ イントラマート All rights Reserved.

※本マニュアルに記載されている社名および商品名は、一般に各社の商標および登録商標です。