OpenRules for IM-BIS 連携ガイド

- 1. 改訂情報
- 2. はじめてご利用の方へ
  - 2.1. OpenRules for IM-BIS 連携ガイドとは
  - 2.2. OpenRules に関するドキュメントの読み進め方
  - 2.3. 本書の構成
  - 2.4. 前提事項
- 3. OpenRules for IM-BIS 連携のメリット
  - 3.1. OpenRules の活用で得られるメリット
  - 3.2. OpenRules と IM-BIS の連携で実現できること
- 4. IM-BIS 連携の概要
  - 4.1. OpenRules の定義ファイルを構成する基本要素
  - 4.2. IM-BIS 連携の概要（画面上の計算や入力チェック）
  - 4.3. IM-BIS 連携の概要（処理対象者検索の絞込み、処理対象者の動的な設定）
- 5. まずは IM-BIS 連携を実行してみよう
  - 5.1. ハンズオンシナリオ（Hello! OpenRules）の概要
  - 5.2. OpenRules のルール定義ファイルを作成する
  - 5.3. IM-BIS と連携したフローを作成する
  - 5.4. 作成したルールを実行する
- 6. 複合条件（AND/OR）を利用して、旅費精算の日当を計算してみよう
  - 6.1. ハンズオンシナリオ（出張手当申請の作成）の概要
  - 6.2. OpenRules で AND/ORを含む条件を定義したルールを作成する
  - 6.3. IM-BIS の画面（フォーム）のイベントと OpenRules を連携する
  - 6.4. 出張旅費精算申請で日当を計算してみよう
- 7. 自動車保険の計算と入力チェックをしてみよう
  - 7.1. ハンズオンシナリオ（自動車保険申請の作成）の概要
  - 7.2. OpenRules で自動車保険申請のルールを作成する
  - 7.3. IM-BIS と連携したフローを作成する
  - 7.4. 自動車保険の申し込みワークフローを実行してみよう
- 8. 複雑なルールを利用して、取引先の与信管理をしてみよう
  - 8.1. ハンズオンシナリオ（取引先与信管理フローの作成）の概要
  - 8.2. OpenRules で取引先に対する与信枠や信用度を評価するルールを作成する
  - 8.3. IM-BIS と連携したフローを作成する
  - 8.4. 取引先の与信管理ワークフローを実行してみよう
- 9. 入力チェックされた金額に応じて、ルートの形を変えてみよう
  - 9.1. ハンズオンシナリオ（稟議フローの作成）の概要
  - 9.2. ワークフローの動的承認者を決定する OpenRules のルール定義ファイルを作成する
  - 9.3. IM-BIS のフローの横配置ノードに OpenRules を連携する
  - 9.4. 作成したルールを実行する
- 10. OpenRules for IM-BIS 連携の運用
  - 1. データソース定義の登録・更新でエラーが発生する
  - 2. ルールの実行時にエラーが発生する、正しく動作しない
  - 3. OpenRules でデバッグをするための設定
- 11. OpenRules のキーワードリファレンス
  - 11.1. テーブルタイプ（TableType）
  - 11.2. 条件として利用できるキーワード
  - 11.3. 結果・処理として利用できるキーワード
  - 11.4. 動的処理対象者設定時に利用できるキーワード
  - 11.5. 特殊なキーワード / テーブルタイプ固有のキーワード
  - 11.6. Methodで利用できるキーワード（API）
  - 11.7. OpenRules が提供するメソッド
  - 11.8. OpenRules のバージョンによる記法の差異
- 12. IM-BIS / OpenRules のバージョン対応表
- 13. OpenRules 7.0.0 注意事項
- 14. ダウンロード
  - 14.1. ハンズオンのサンプルモジュール（完成版）
  - 14.2. OpenRules のテンプレート
  - 14.3. 各種定義ファイルのインポートの手順

改訂情報

変更年月日	変更内容
2015-04-01	初版
2015-08-01	第２版　下記を変更しました ▪ 「*IM-BIS / OpenRules のバージョン対応表*」に IM-BIS 2015 Summer（8.0.8）を追加しました
2015-12-01	第３版　下記を変更しました ▪ 「*IM-BIS / OpenRules のバージョン対応表*」に OpenRules 6.3.3 を追加しました ▪ 「*利用できる演算子*」に演算子「Does Not Contain」を追加しました
2016-04-01	第４版　下記を変更しました ▪ 「*IM-BIS / OpenRules のバージョン対応表*」の IM-BIS のバージョン表記を変更しました
2016-08-01	第５版　下記を変更しました ▪ 「*項目名のマッピング（Glossary）*」に名称を変更する場合の記載方法を追加しました
2017-08-01	第６版　下記を変更しました ▪ 機械翻訳による可読性向上のため、全体的に内容の見直しを行いました。 ▪ OpenRules モジュールの8.0.3（ OpenRules 6.4.2を同梱 ）リリースに伴い、新しく利用できる機能等をリファレンスに追加しました。 バージョンごとの変更内容の概要については、以下を参照してください。 ▪ *OpenRules のバージョンによる記法の差異* ▪ 「*ハンズオンのサンプルモジュール（完成版）*」にインポート時の注意事項を追加しました。
2017-12-01	第７版　下記を変更しました ▪ 「*IM-BIS / OpenRules のバージョン対応表*」のバージョン表記を変更し、ユーザモジュールのファイル名を追加しました。 ▪ ハンズオンシナリオ・サンプルについて、 OpenRules 6.2.6 から 6.3.0 以降で変更された以下の内容を反映しました。 ▪ *Method からの Decision インスタンスに含まれるパラメータへのアクセス方法の変更*
2018-04-01	第８版　下記を追加しました ▪ 「*ルールの実行時にエラーが発生する、正しく動作しない*」にdecision誤りによるエラーの解決方法を追加しました。
2019-04-01	第９版　下記を変更しました ▪ 「*IM-BIS / OpenRules のバージョン対応表*」に OpenRules 7.0.0 を追加しました。 ▪ 「*OpenRules 7.0.0 注意事項*」を追加しました。 ▪ 「*ダウンロード*」のサンプルを 「*OpenRules 7.0.0 注意事項*」に対応しました。

はじめてご利用の方へ

OpenRules for IM-BIS 連携ガイドとは

この「 OpenRules for IM-BIS 連携ガイド」は、初めて OpenRules ・ IM-BIS 連携開発を行うユーザのみなさまが、ご自分で必要な設定やルールの開発をするための支援を目的としたドキュメントです。
このドキュメントの各種シナリオを行っていただくと、ルールの活用方法や実現できる機能をご理解いただくことができます。

OpenRules に関するドキュメントの読み進め方

「 OpenRules for IM-BIS 連携ガイド」は、「 IM-BIS 」の外部連携の一つ、ルール（OpenRules）との連携に特化して、説明しています。

本書での操作に当たっては、 IM-BIS の各種操作ガイドを一通り確認し、基本操作について理解いただいていることが前提です。
以下の表を参考にして、スキルレベルに合わせたマニュアルを参照してから本書をお読みください。

スキルレベル	対応するマニュアル
レベル1	▪ IM-BIS セットアップガイド ▪ intra-mart Accel Platform セットアップガイド
レベル2	▪ IM-BIS ビギナーズガイド
レベル3	▪ IM-BIS システム管理者操作ガイド ▪ IM-BIS 業務管理者操作ガイド ▪ IM-BIS ユーザ 操作ガイド
レベル4	▪ OpenRules for IM-BIS 連携ガイド （本書）

本書の構成

本書は、以下の構成です。

- 改訂情報

  本書の初版発行以降の改訂履歴情報を掲載しております。

- OpenRules for IM-BIS 連携のメリット

  OpenRules と IM-BIS の連携に関する概要について説明しております。
  最初に本書の扱う OpenRules と IM-BIS の連携の全体像を把握することができます。

- IM-BIS 連携の概要

  OpenRules と IM-BIS を連携した開発の特徴や概要について説明しております。
  開発における OpenRules と IM-BIS の連携開発の利用シーンなどを確認したい場合にご利用ください。

- まずは IM-BIS 連携を実行してみよう

  一から OpenRules でルールを定義し、 IM-BIS との連携の設定、フローの実行までの一覧の流れを確認するためのハンズオンです。
  OpenRules と IM-BIS を連携するための基本操作を確認することができます。

- 複合条件（AND/OR）を利用して、旅費精算の日当を計算してみよう

  OpenRules でルールを定義する場合の複合条件（AND/OR）の設定方法を、旅費精算のワークフローを作成しながら確認するためのハンズオンです。
  OpenRules と IM-BIS を連携するための基本操作を理解した上で操作いただくと効果的な内容です。

- 自動車保険の計算と入力チェックをしてみよう

  自動車保険の申請ワークフローの作成を通じて、 OpenRules で高度な条件設定や、処理結果に基づいた入力チェックを設定する手順を確認するためのハンズオンです。
  本ハンズオンは、 OpenRules 、 IM-BIS の応用的な内容を含んでおりますので、一通り基本操作を確認した上で実行してください。

- 複雑なルールを利用して、取引先の与信管理をしてみよう

  新規・既存の取引先与信管理ワークフローの作成を通じて、 OpenRules でのさまざまな条件設定や、評価順序のコントロールなどの手順を確認するためのハンズオンです。
  本ハンズオンは、 OpenRules の応用的な内容を含んでおりますので、一通り基本操作を確認した上で実行してください。

- 入力チェックされた金額に応じて、ルートの形を変えてみよう

IM-BIS の「動的処理者設定（外部連携）」を OpenRules と連携する場合の設定方法について確認するためのハンズオンです。
本ハンズオンでは、 IM-BIS の動的処理者設定の仕様と合わせてご確認いただくと効果的な内容です。

- OpenRules for IM-BIS 連携の運用
  OpenRules for IM-BIS 連携の開発時に、ルールの定義や実行時に発生すエラーや対処方法について説明しております。
  トラブルシューティングとして困った場合にご利用ください。
- OpenRules のキーワードリファレンス
  OpenRules でのルールを定義する際に利用できるキーワードと利用方法などを説明しております。
  業務要件に基づく高度なルールを定義する際などにご利用ください。
- ダウンロード
  OpenRules でのルールを定義するためのExcelファイルのテンプレートや各種ハンズオンの完成版定義ファイルを公開しております。
  ハンズオンシナリオの解答としてご利用ください。

前提事項

本書は、下記の前提条件に基づいた設定での操作を説明しております。
利用している環境や製品のバージョン違いによって、操作方法や外観などに差異が発生する場合には、適宜読み替えてください。

- 本書の操作環境
  - Microsoft Excel 2010
  - intra-mart Accel Platform 2015 Spring (Juno) 8.0.10 / 2014 Winter (Iceberg) 8.0.9
  - IM-FormaDesigner for Accel Platform 8.0.9 / 8.0.8-PATCH_001
  - IM-BIS for Accel Platform 8.0.7 / 8.0.6-PATCH_002
  - OpenRules 8.0.1 （ OpenRules 6.3.2 Alpha Evaluation Version ）

OpenRules for IM-BIS 連携のメリット

IM-BIS との連携には、以下のような特徴があります。

OpenRules の活用で得られるメリット

OpenRules とは

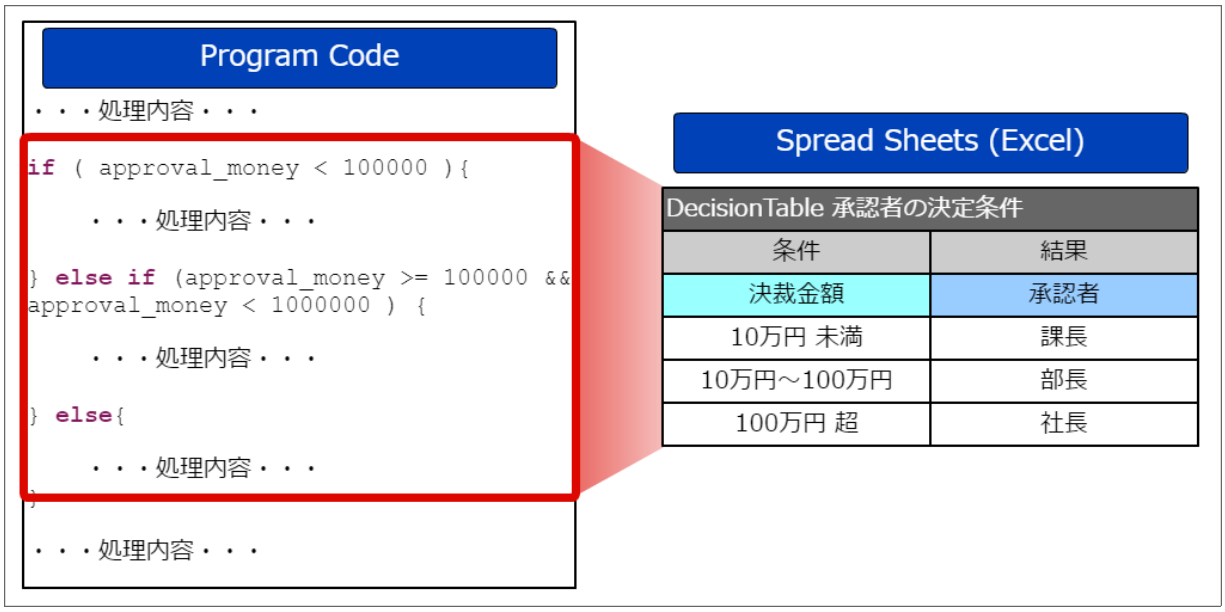
OpenRules とは、OpenRules社の提供するBRMS製品(BRMS＝Business Rule Management System)です。
OpenRules をはじめとするBRMSとは、業務の複雑なロジック部分のみを分離・独立させ、業務担当者に複雑な業務ロジックの開発を可能にするしくみです。

OpenRules を活用すると、以下のようなメリットを得ることができます。

Excel表形式で条件判定のロジックを記述することができる

OpenRules との連携では、条件判定ロジックをExcelのシンプルな表で表現することができます。
このため、プログラムを扱えない業務担当者でも条件判定のロジックを記述することができます。

従来は左の図のようにプログラムコードで実装した業務ロジックは、 OpenRules では右のようにスプレッドシード（Excel）で表現できます。



業務担当者が業務ロジックの変更に対応できるようになる

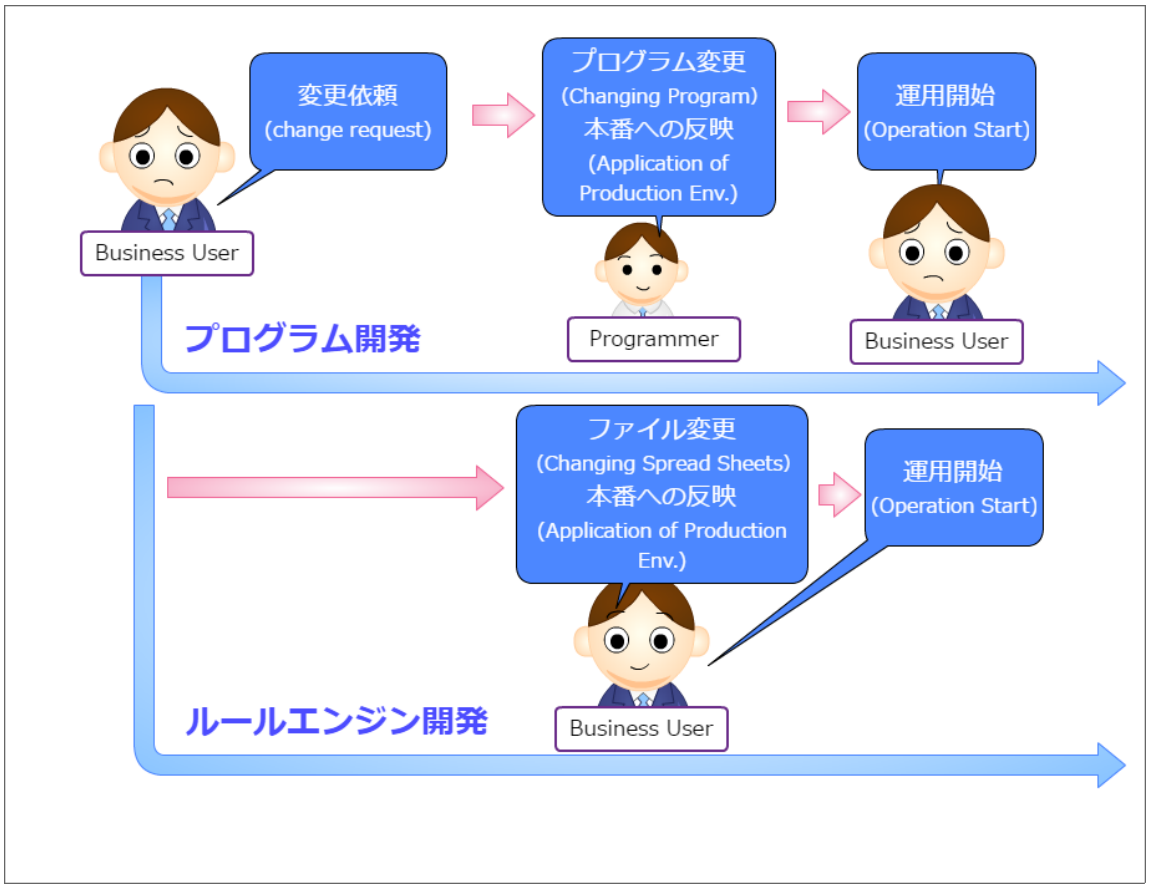
従来のプログラムによるロジックでは、業務の変更に伴うロジックの変更が発生した場合、プログラムを修正しなくてはならないため、プログラム開発者しか対応できませんでした。
OpenRules では、そのようなロジック部分のみをルールの実態であるExcelファイルに集約するため、業務担当者であってもロジックの変更に対応できます。
このため、業務の変更に伴うロジックへの対応で発生するプログラム開発にかかるコストや期間を縮小することができ、メンテナンスコストの抑制が実現できます。

例えば、「日当の金額変更」といった場合には、従来のプログラム（図中の上段フロー）では、ユーザから変更依頼をプログラマに発行し、プログラマによる変更や本番環境への適用作業を経て運用開始となるため、以下のような問題が発生していました。

- 軽微な対応内容であってもプログラマ（開発・運用ベンダ）に依頼が必要
- プログラマ側では、変更に対する影響調査や反映作業に時間を要する
- 場合によってはサーバの再起動を伴うために業務に支障をきたす可能性がある

OpenRules （図中の下段フロー）を利用した場合にはユーザ自身が対応作業を行うことができます。

- ルールの変更はExcelファイルの編集で可能なため、ユーザが実施可能
- 影響範囲はExcelファイルのみとなるため、調査に要する時間や反映がプログラムへの変更の場合より少なくなる可能性がある
- Excelファイルによるルールの変更はサーバ再起動が不要なため、業務に影響を及ぼすことなく対応可能



OpenRules と IM-BIS の連携で実現できること

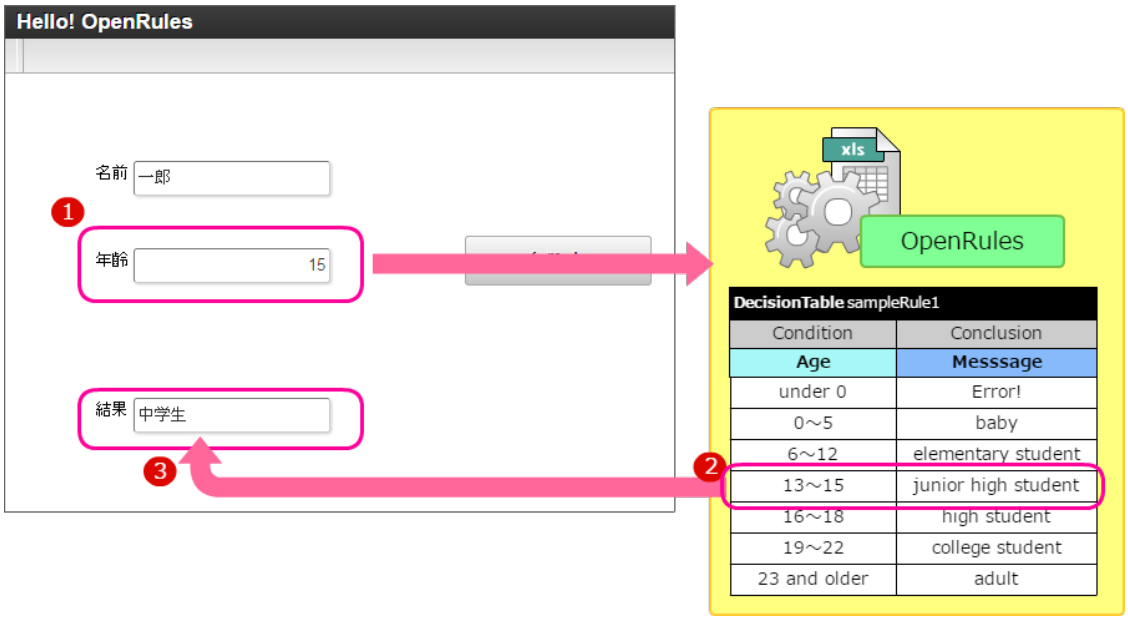
OpenRules は、 IM-BIS と連携して利用することができます。
ここでは、 IM-BIS との連携の概要について説明します。

OpenRules と IM-BIS の連携で実現できる処理

OpenRules と IM-BIS の連携では、Excelファイルの定義で、以下の処理を実現できます。

画面上での入力チェックや計算のロジック

OpenRules と IM-BIS の連携では、Excelファイルの定義で以下のようなロジックを記述することができます。



【図の説明】

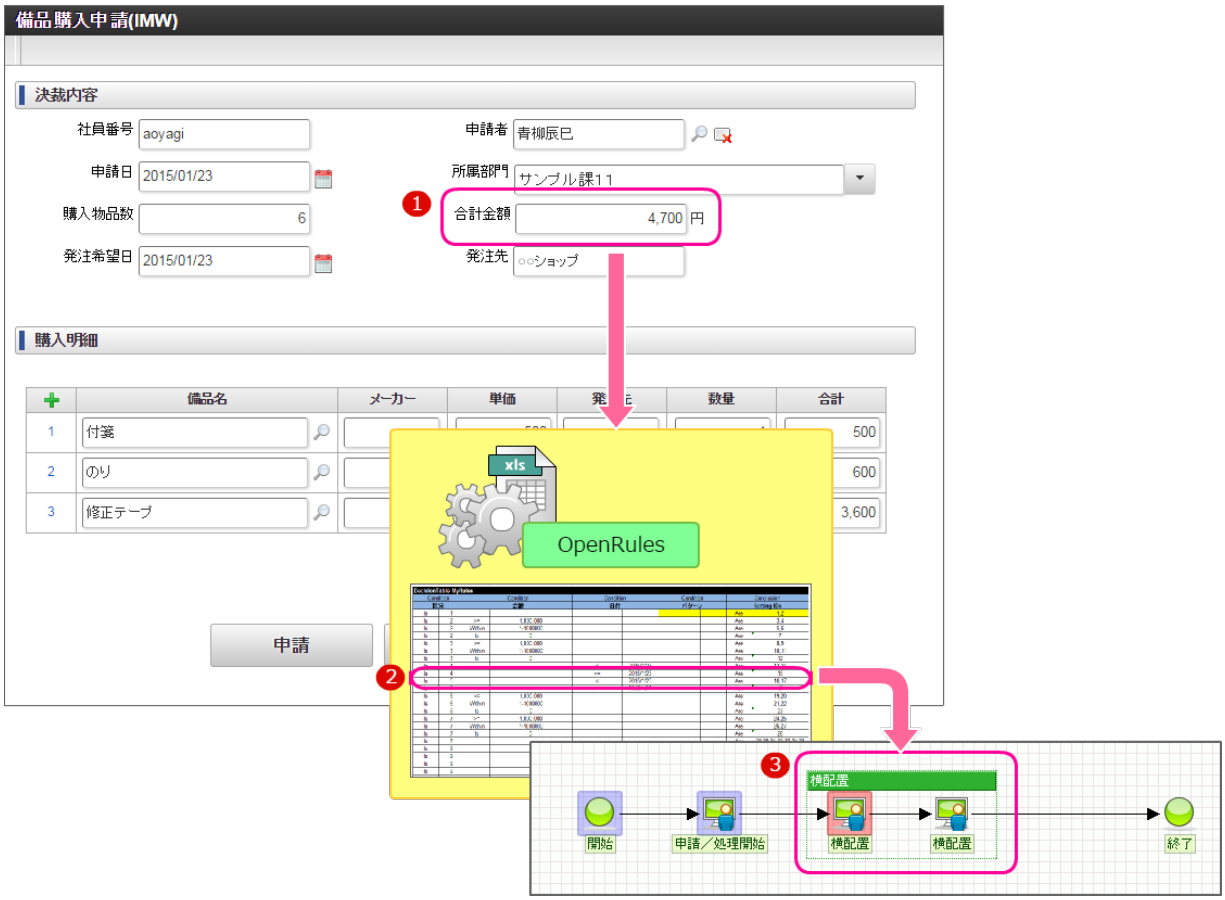
1. 年齢の入力後、ルールを実行します。

2. 条件を評価し、合致する条件の処理を実行します。

3. ルールエンジンから受け取った内容を表示します。

ワークフローの動的な処理対象者決定のロジック

OpenRules をワークフローの処理対象者決定に利用すると、申請時に承認者を決定するようなワークフローでの条件をExcelファイルで設定することができます。



【図の説明】

1. 購入する備品の合計金額を計算して申請します。

2. 条件を評価し、合致する条件の処理対象者を返却します。

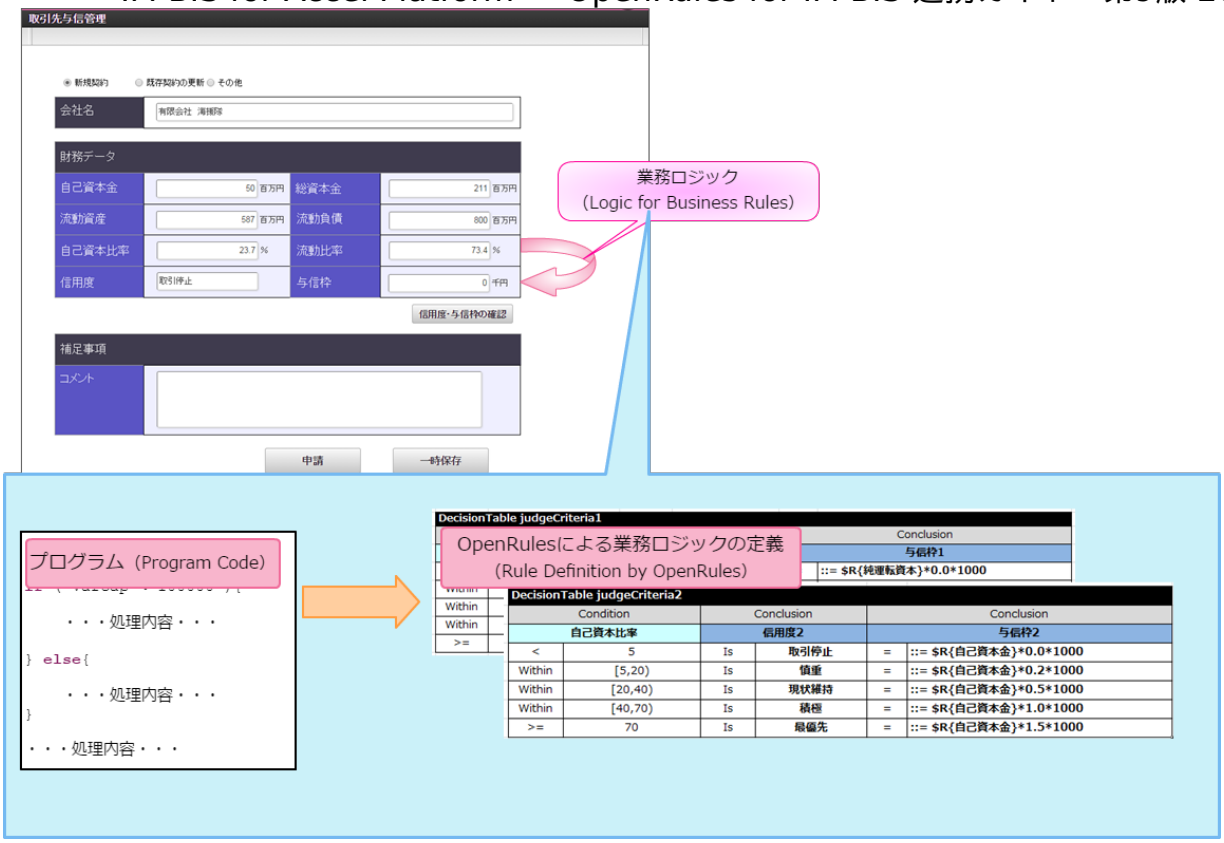
3. ルールエンジンから受け取った内容に基づいて複数の承認者をノードに設定します。

OpenRules と IM-BIS の連携を利用して得られるメリット

OpenRules と IM-BIS の連携では、Excelファイルの定義で、以下の処理を実現できます。

業務ユーザによる業務ロジックのメンテナンスが可能です

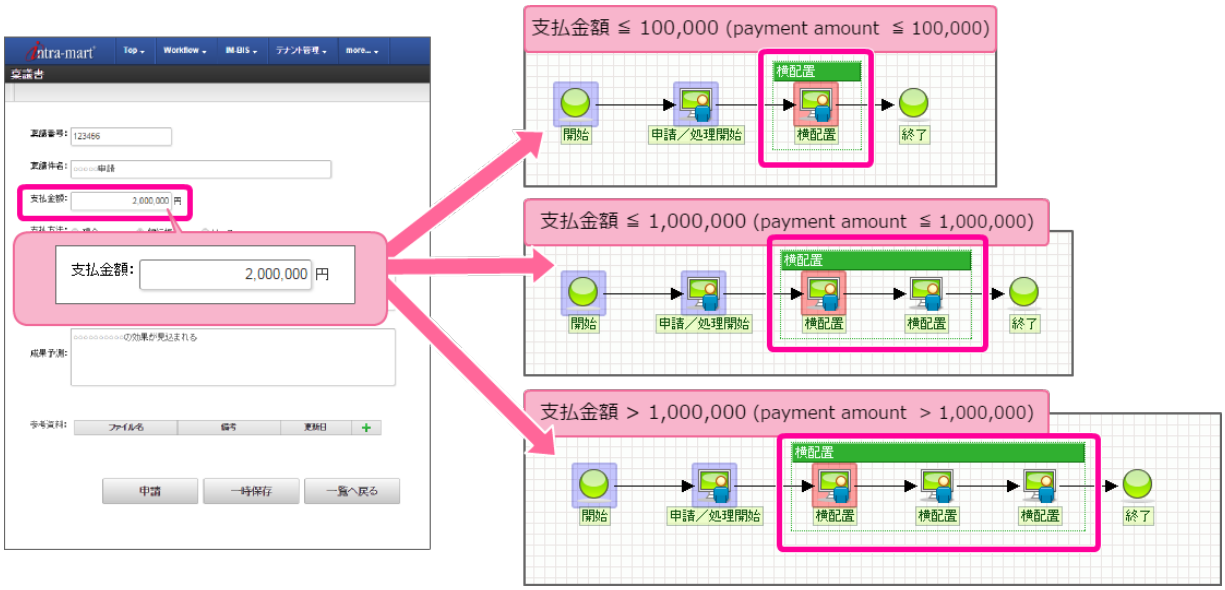
従来の IM-FormaDesigner や IM-BIS の開発で複雑な入力チェックなどを行うためには、スクリプト開発などのプログラム開発が必要でしたが、
OpenRules ではExcelファイルでプログラムのロジックを表現することができます。
このため、業務ユーザによる業務ロジックのメンテナンスが可能です。



業務データに基づくルートを設定できます

従来の開発では、ワークフローにおける動的処理対象者の決定を自動化したり、検索時の対象を絞り込むためには、プログラムで条件を返却する（絞込みの範囲の）プラグインを記述する必要がありましたが、 OpenRules ではExcelに条件と返却するプラグインを設定することでロジックを実装できます。
このため、 IM-BIS ・ OpenRules 連携を利用すると、処理対象者決定ロジックの簡素化、業務ユーザによるメンテナンスを実現することができます。

以下の図のように申請書の金額に基づいて承認者の人数を変更するためのロジックも IM-BIS と OpenRules の連携で実現できます。



IM-BIS 連携の概要

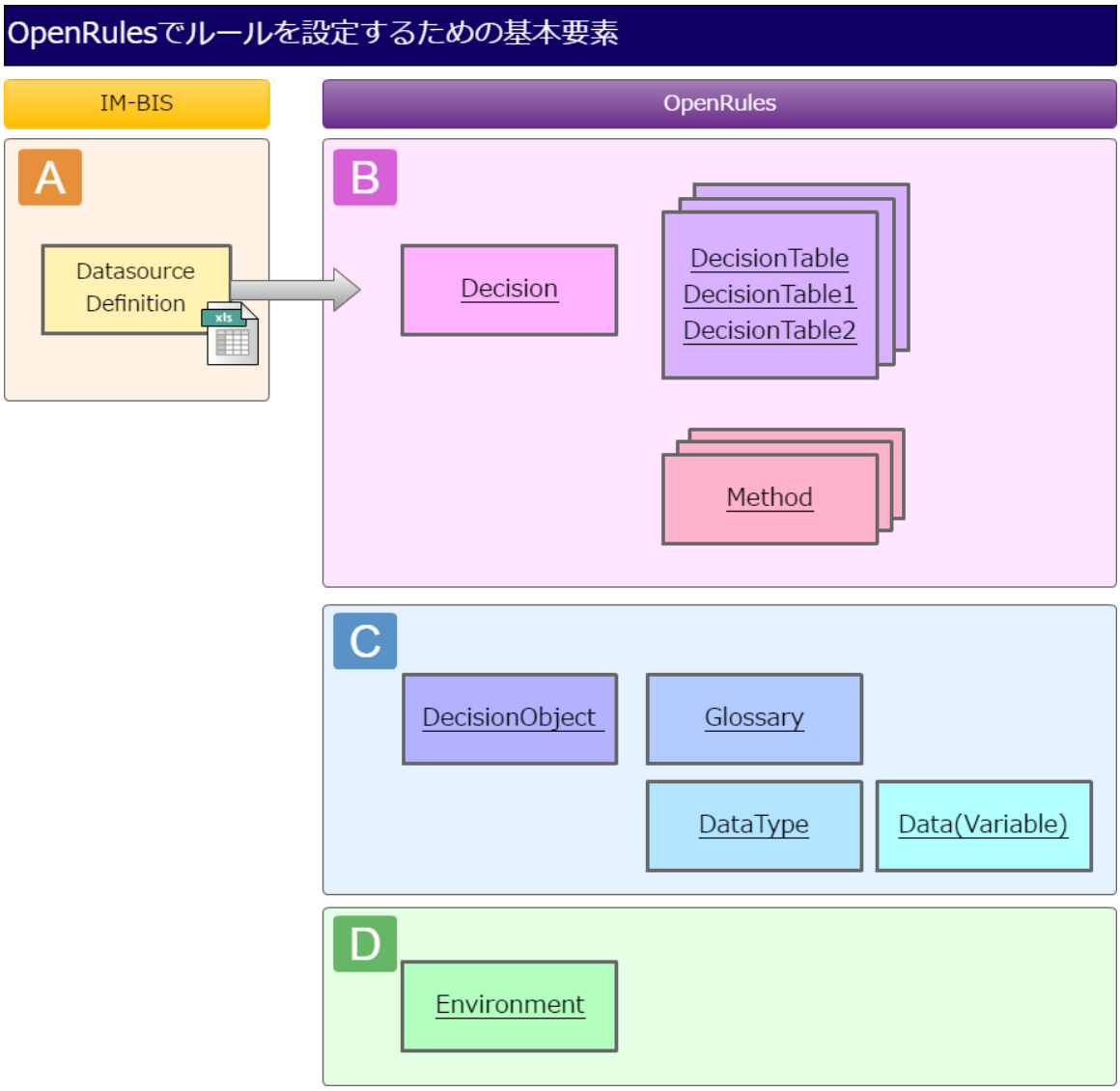
本章では、 IM-BIS 連携の概要、 OpenRules の開発の基本的な情報について説明しています。

OpenRules の定義ファイルを構成する基本要素

OpenRules の定義は、Excelファイルに集約されます。
OpenRules を扱うために必要な基本要素について説明します。

OpenRules の構成要素（テーブルタイプ）

OpenRules で実行するためのルールを記述したExcelファイルには、以下のような要素で構成されます。
以下の図の要素の単位は「テーブルタイプ（TableType）」と呼び、テーブルタイプごとに決まった形式で記述します。

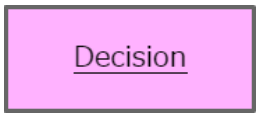


A. データソース定義

OpenRules で実行する条件はすべてExcelファイルで定義し、 IM-BIS のデータソース定義に登録します。

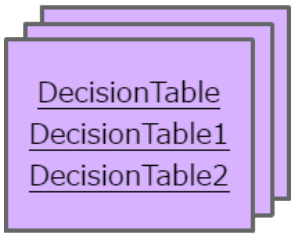
B. ルールを実行するための要素

ルールを実行するための要素には、以下の3つがあります。
ルールの実行時には、 *Decision* の内容に従って、入出力内容の受け渡しや *DecisionTable* を順番に実行します。



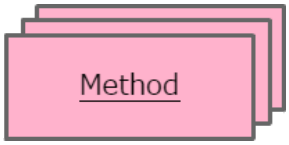
処理内容、順序の設定
ルールの中心。コントローラの役割を持っています。

- 実行するファイルに含まれるDecisionTable、Methodなどの実行条件や順番を設定
- 実行結果を出力（返却）項目のグループ（オブジェクト）に設定



ルールの表
ルールの実体。条件と処理内容が書いてあります。
１つのルールのExcelファイルには、複数のDecisionTableを記述できます。
実行する方法によって、DecisionTable / DecisionTable1 / DecisionTable2の３つのタイプがあります。

- 実行する条件と条件に合致したときの処理を設定
- 処理の順序は、Decisionの定義で決定



Javaプログラム
ルールの拡張項目。Javaプログラムで自由に処理を記述することができます。
記述したMethodは、DecisionTable等から呼び出して実行します。

C. ルールで参照する項目の定義や初期化を設定するための要素

ルールを実行する際の条件や処理結果の格納に利用する項目を定義するための要素には、以下の3つがあります。
ルールの実行時には、 *Decision* の内容に従って、入出力内容の受け渡しや *DecisionTable* を順番に実行します。

オブジェクトのインスタンスの設定
ルールで利用する項目を、一定のグループ（オブジェクト）単位にBIS（データマッパー）と受け渡しを行います。

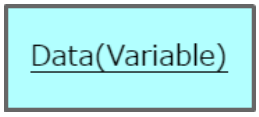
- 入出力の項目群（オブジェクト）の受け渡し方法を設定

項目の論理名/物理名のマッピング
ルールで利用する項目の論理名と物理名の管理、項目のグループ（オブジェクト）を管理します。

- ルールで利用する項目の論理名（業務担当者向けの名前）と物理名（プログラム向けの名前）のマッピングを設定

項目のデータ型設定
ルールで利用する項目のデータ型を管理します。

- Glossaryでマッピングした項目の物理名に基づいたデータ型を設定



項目の初期値設定
ルールで利用する項目の初期値を管理します。
ここで設定した初期値は実行前処理で利用されます。

- ルールで利用する項目の初期値を設定

D. ルールを構成するファイルに関する情報

環境設定
ルールを構成するファイルの情報を管理します。

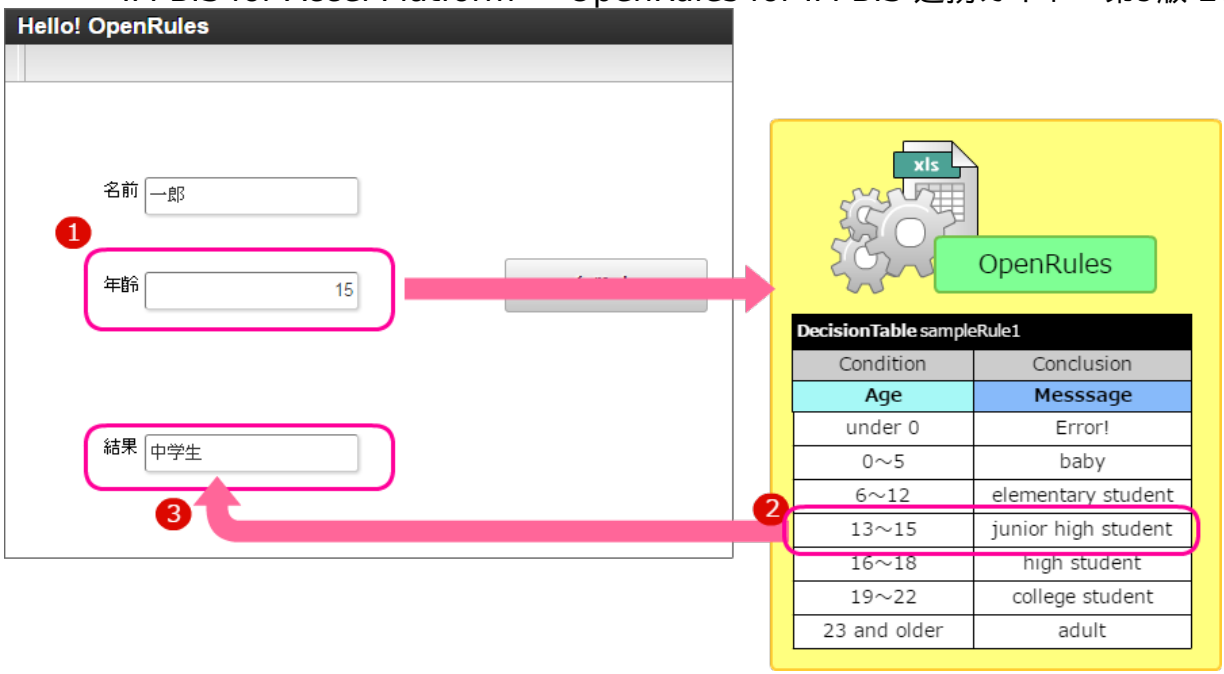
- IM-BIS との連携情報（IntramartTemplate.xls）を設定
- ルールが複数のファイルに分かれている場合の関連するファイル名を設定

IM-BIS 連携の概要（画面上の計算や入力チェック）

IM-BIS 連携の概要について説明します。

画面上の計算などを行う場合の実行イメージ

画面上の計算などを行う場合には、以下のようなイメージで実行できます。

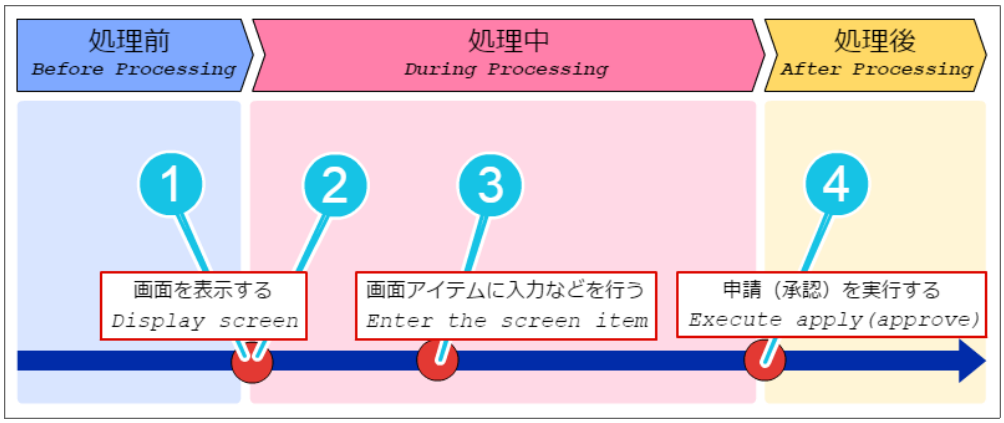


【図の説明】

1. 年齢の入力後、ルールを実行します。
2. 条件を評価し、合致する条件の処理を実行します。
3. ルールエンジンから受け取った内容を表示します。

画面上の計算などを行う場合の実行のタイミング

IM-BIS の連携は、以下のタイミングで処理を実行します。



【図の説明】

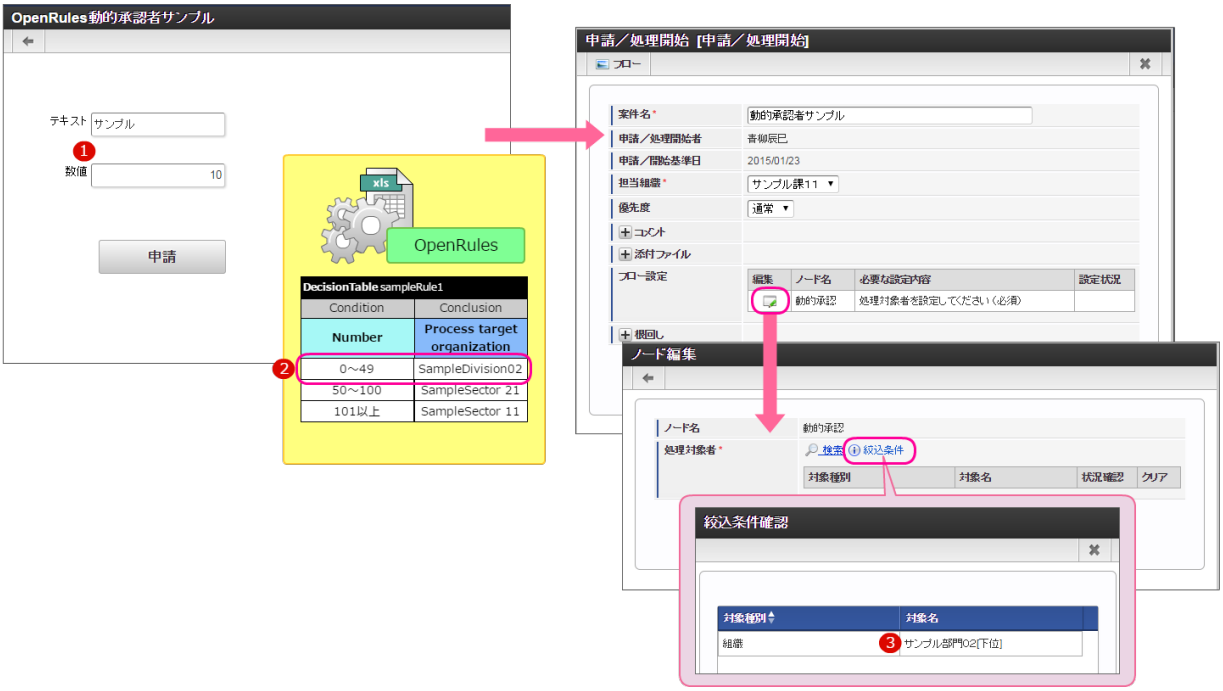
1. **前処理**
   画面の表示前に行われる処理です。
2. **初期表示イベント**
   画面を表示するタイミングで行われる処理です。
3. **アイテムイベント、テーブルイベント**
   画面上のアイテムへの入力などによって行われる処理です。
4. **後処理**
   申請や承認の実行後に行われる処理です。

IM-BIS 連携の概要（処理対象者検索の絞込み、処理対象者の動的な設定）

IM-BIS 連携の概要について説明します。
IM-BIS の動的処理者設定の機能では、「処理対象者の設定」と「処理対象者を選択する際の検索条件の絞込設定」を利用することができます。

処理対象者検索の絞込みを行う場合の実行イメージ

処理対象者検索の絞込みを行う場合には、以下のようなイメージで実行できます。

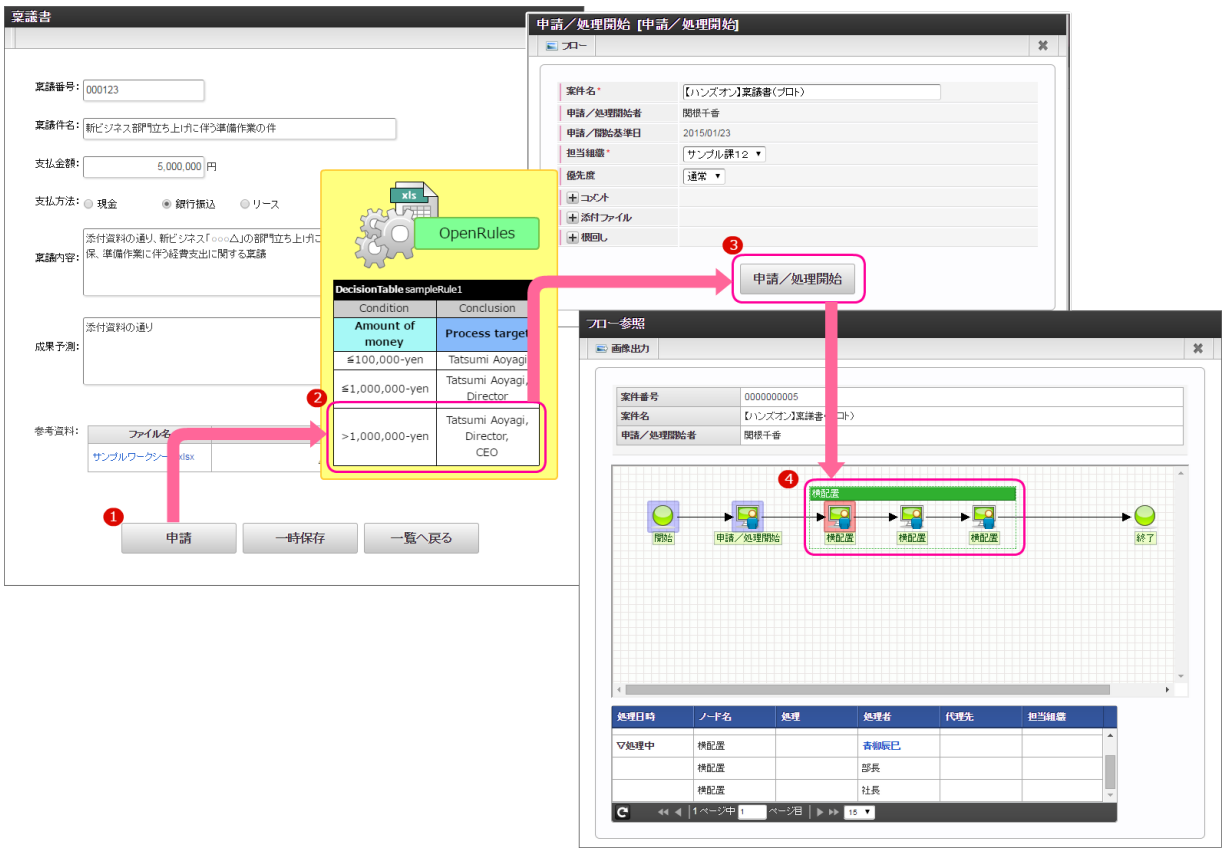


【図の説明】

1. 任意の数字を入力して申請を行います。
2. 条件を評価し、合致する条件の組織を処理対象組織として返却します。
3. ルールエンジンから受け取った内容に基づいて処理対象者の検索の絞込条件を設定します。

処理対象者を自動的に設定する場合の実行イメージ

処理対象者を自動的に設定する場合には、以下のようなイメージで実行できます。



【図の説明】

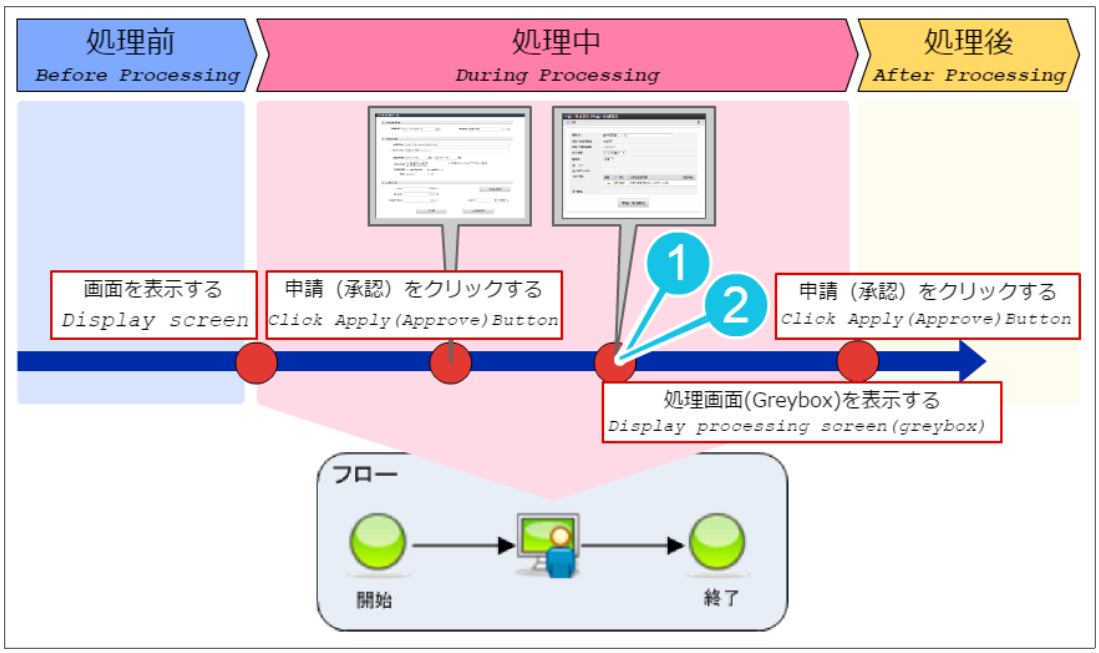
1. 必要な内容を入力して申請を行います。
2. 条件を評価し、合致する条件の組織を処理対象者として返却します。

3. 内部で自動的に後続のノードの処理対象者に返却された処理対象者を設定して申請を実行します。

4. ルールエンジンの実行結果に基づいて横配置ノードの処理対象者が展開されます。

IM-BIS 連携で動的に処理対象者を設定する場合の実行のタイミング

IM-BIS の連携は、以下のタイミングで処理を実行します。



【図の説明】

1. **処理対象者の検索の絞り込み条件**
   標準処理画面（greybox）での処理対象者検索に対して暗黙の検索条件を設定します。

2. **動的処理対象者の設定**
   動的承認ノードや縦配置・横配置ノードへ処理対象者を自動的に設定します。

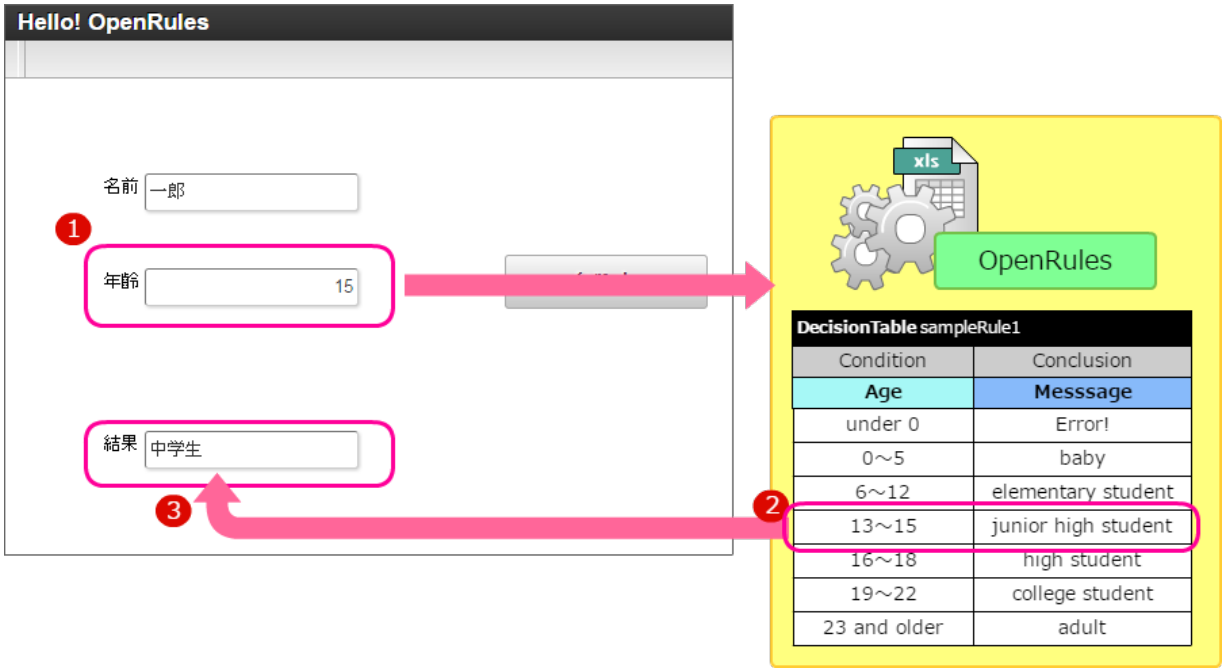
まずは IM-BIS 連携を実行してみよう

本章では、シンプルな IM-BIS ・ OpenRules 連携のアプリケーション、フローの開発、実行方法を実践するシナリオをまとめています。
なお、ハンズオンの実行前には「 IM-BIS ビギナーズガイド 」の「5.1 ワークフロー・業務プロセスの管理者を登録する」を参考にして、BIS管理者、BIS担当者ロールの付与を実行してください。

ハンズオンシナリオ（Hello! OpenRules）の概要

このシナリオでは、以下のようなルールの作成と、そのルールを実行するためのフローを作成します。

- ユーザが名前と年齢を入力し、「イベント」を実行すると、年齢に合わせたメッセージを返却して結果に表示する

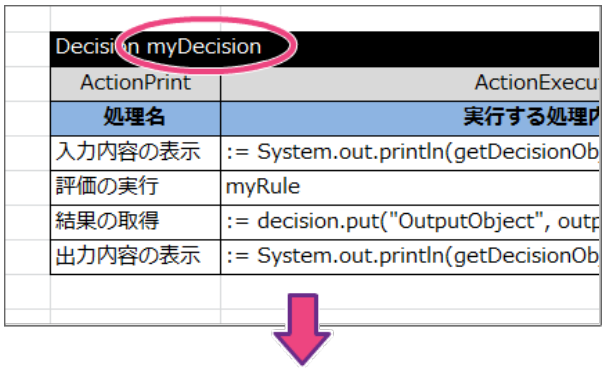


このルールの作成の大まかな手順は、以下の通りです。

OpenRules による値の受け渡しの設定手順

Excelファイルの作成

- OpenRules の実行に必要なExcelファイルをローカルで作成してください。



データソース定義の登録

- 作成したExcelファイルをアップロードしてください。
- 利用する項目（In/Out）の設定を行ってください。

アクション設定の開始

- フォームデザイナで「アクション設定」をクリックしてください。
  - ノードの前処理/後処理に設定する場合には、フロー編集で右クリック後に「外部連携」をクリックしてください。

イベント設定

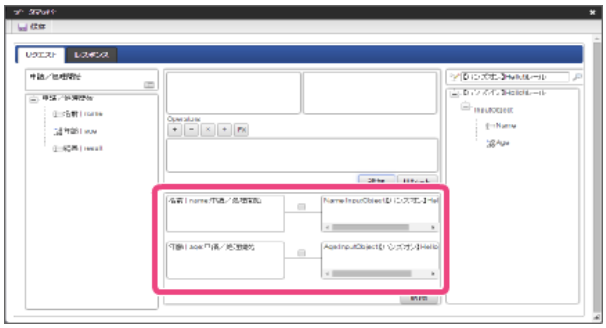
- トリガとなる「画面アイテム」と「イベントタイプ」（アイテムイベント、テーブルイベント）を選択してください。

アクション設定

- 「外部連携」を選択してください。

データマッパー設定

- データソース定義のパラメータと画面の項目をマッピングしてください。



このシナリオを通して習得できる知識

このシナリオを実行すると、以下の知識を理解することができます。

- OpenRules で利用できる簡単なルールの作成方法
- OpenRules のルール定義ファイルを IM-BIS と連携して実行するための方法

このシナリオを学習する前に必要な知識

このシナリオの実行に当たり、下記の知識を事前に確認してください。

- IM-BIS でのフローの作成・実行方法
- Excelファイルの編集方法

OpenRules のルール定義ファイルを作成する

OpenRules では、Excelファイルで条件や、条件が合致したときの処理を表で作成できます。

このハンズオンでは、簡単なルールをExcelファイルに定義する手順を確認することができます。

ルールを定義するExcelファイルを作成する手順

- このシナリオで作成するルールの概要
- ルールのExcelファイルを作成する手順
- Excelファイルにルールの表（ DecisionTable ）を作成する
- Excelファイルに項目名のマッピング表（ Glossary ）を作成する
- Excelファイルに項目とデータ型の定義（ Datatype ）を作成する
- Excelファイルに項目の初期値（ Data ）を作成する
- Excelファイルにオブジェクトのインスタンスの設定（ DecisionObject ）を作成する
- 入出力処理やルールの実行設定（ Decision ）を作成する
- 環境設定（ Environment ）を作成する

このシナリオで作成するルールの概要

- 作成するルールの内容

  入力値の年齢に基づいて、条件に合致したメッセージを返却する

  - 入力値：年齢
  - 出力値：メッセージ

  条件と返却するメッセージの組み合わせは、以下の通りです。

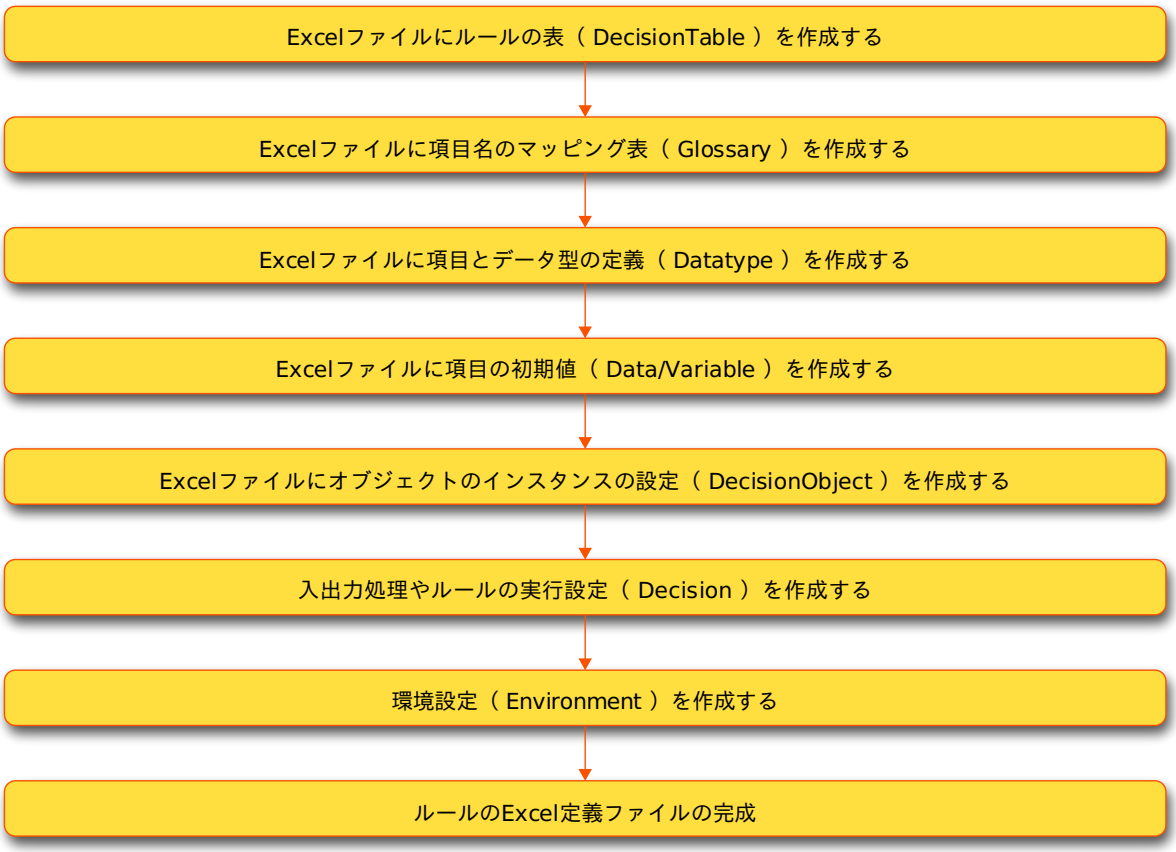
  - 0歳未満（負数）の場合、「エラー！」と返却する
  - 0歳以上6歳未満の場合、「幼児」と返却する
  - 6歳以上13歳未満の場合、「小学生」と返却する
  - 13歳以上16歳未満の場合、「中学生」と返却する

- 16歳以上19歳未満の場合、「高校生」と返却する
- 19歳以上23歳未満の場合、「大学生」と返却する
- 23歳以上の場合、「社会人」と返却する

ルールのExcelファイルを作成する手順

新規にExcelファイルを作成し、 OpenRules の実行に必要な表（テーブル）を順番に作成していきます。
このシナリオでは、以下の図の流れで作成していきます。

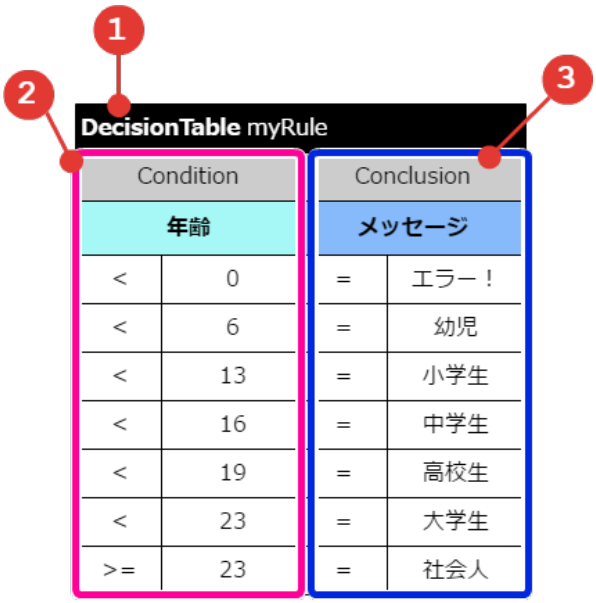
- Hello! OpenRules の手順



Excelファイルにルールの表（ DecisionTable ）を作成する

最初に、ルールの実体である「 *DecisionTable* 」の書き方を確認しましょう。
「 *DecisionTable* 」では、実際に実行する処理の「条件」と「条件に合致したときの処理」の組み合わせを表にします。

おおまかには、以下の図のように条件と処理をExcel上にまとめます。



1. **テーブルタイプ**
   表の種類を表します。

2. **条件**
   条件は、以下の形で構成されます。

   1行目：列が条件・評価（結果）のどのタイプかを表すキーワード
   2行目：評価に利用する項目名
   3行目～：演算子と基準値のパターン

3. **評価（条件に合致した時の処理・値）**
   評価は、以下の形で構成されます。

   1行目：列が条件・評価（結果）のどのタイプかを表すキーワード
   2行目：評価に利用する項目名
   3行目～：演算子と評価値のパターン

ルールの表（ DecisionTable ）のヘッダ部分を作成する

表のヘッダは、 OpenRules のキーワードや項目名をルールに従って設定する必要があります。
最初にヘッダとして必要な内容を設定しましょう。

1. パソコンでExcelを起動し、新規にブックを作成します。
2. OpenRules では、設定を含めたすべての表の1行目に「(1) キーワード」と「(2)テーブルの名称」を半角スペースを空けて記述します。

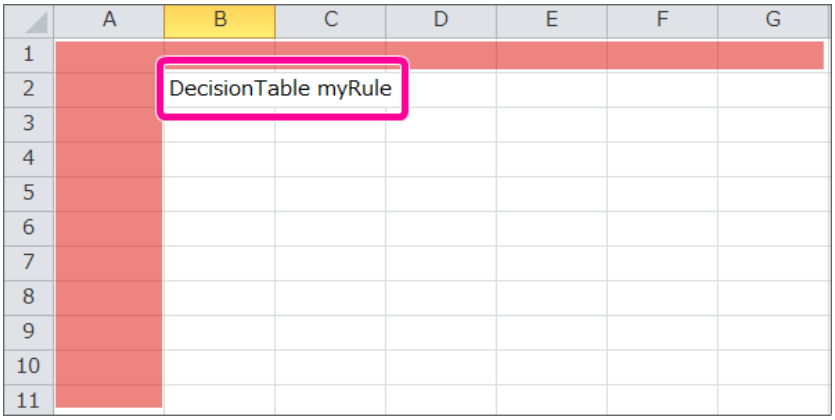
3. 最初に作成するのは、条件と処理をまとめた表として扱うため、キーワードは「 *DecisionTable* 」と設定してください。

4. ヘッダには以下の形式でキーワードとテーブル名を入力してください。

```
[キーワード]+半角スペース+[テーブル名]
```

   テーブル名は、任意の名称を半角英数字で入力してください。
   1つのデータソース定義に設定するExcelファイル内で、テーブル名が一意となるように設定してください。

   Excelファイルのシート上には、以下の図の通りに入力してください。
   OpenRules のしくみ上、1番上と1番左端の行と列には何も入力してはいけません。（以下の図では赤色の範囲が該当します。）
   そのため、1つずつ行と列を空けた上でテーブルタイプとテーブル名を入力してください。



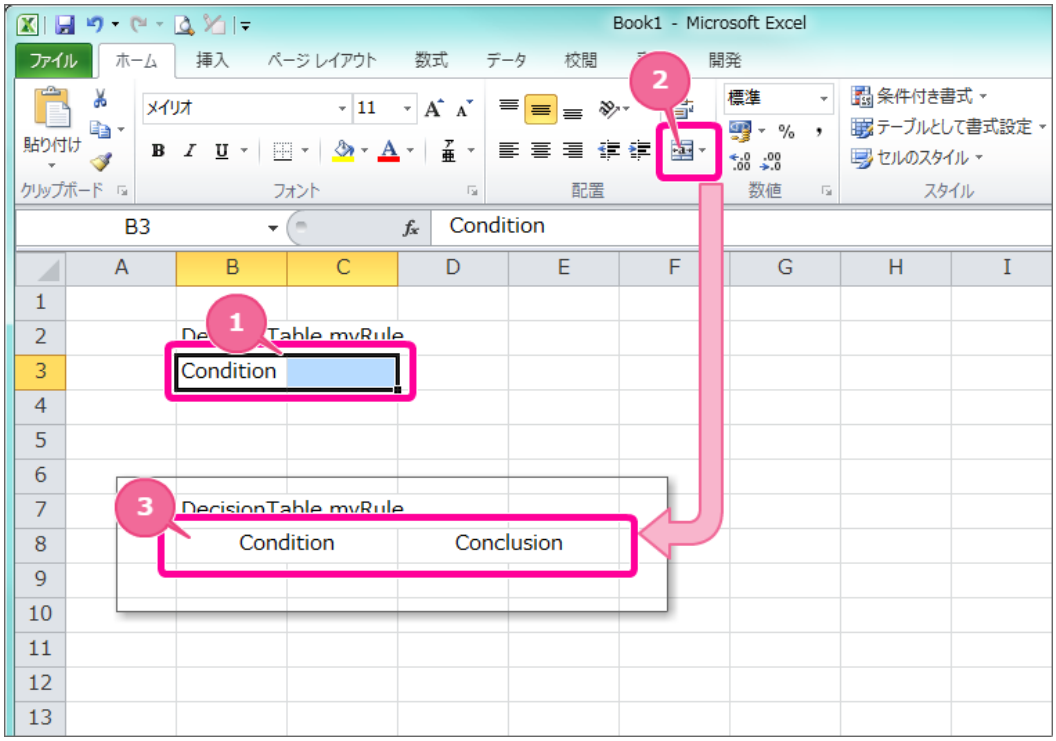
5. 2行目には、「条件」・「処理」に該当するキーワードを入力してください。

DecisionTable myRule
keyword for condition and conclusion

- 「条件」を表すキーワードとして、左のセルに「Condition」を入力してください。
- 「処理」を表すキーワードとして、右のセルに「Conclusion」を入力してください。

DecisionTable myRule	
Condition	Conclusion

Excelファイルのシート上には、以下のように記述してください。
条件や処理には、演算子の項目、値の項目の2列で構成します。
したがって、Excelファイル上では条件、処理の単位で結合セルに設定してください。



【*DecisionTable*のサブヘッダを設定する手順】

1. 「Condition」と入力したセルと隣のセルをドラッグして選択状態にしてください。
2. 「セルを結合して中央揃え」をクリックしてください。
3. 「Condition」「Conclusion」のそれぞれのセルを同様に結合してください。

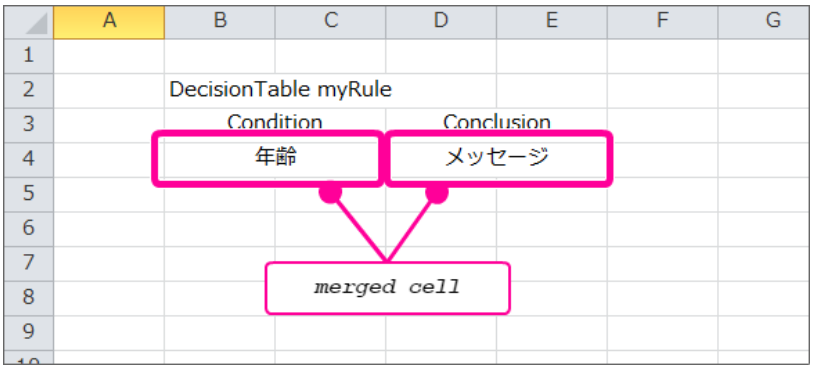
6. 3行目には、「条件」・「処理」の項目名（論理名）を入力してください。

DecisionTable myRule	
Condition	Conclusion
name for condition	*item name for conclusion*

左のセルには、「条件の項目の論理名」として「年齢」と入力してください。
右のセルには、「処理の項目の論理名」として「メッセージ」と入力してください。

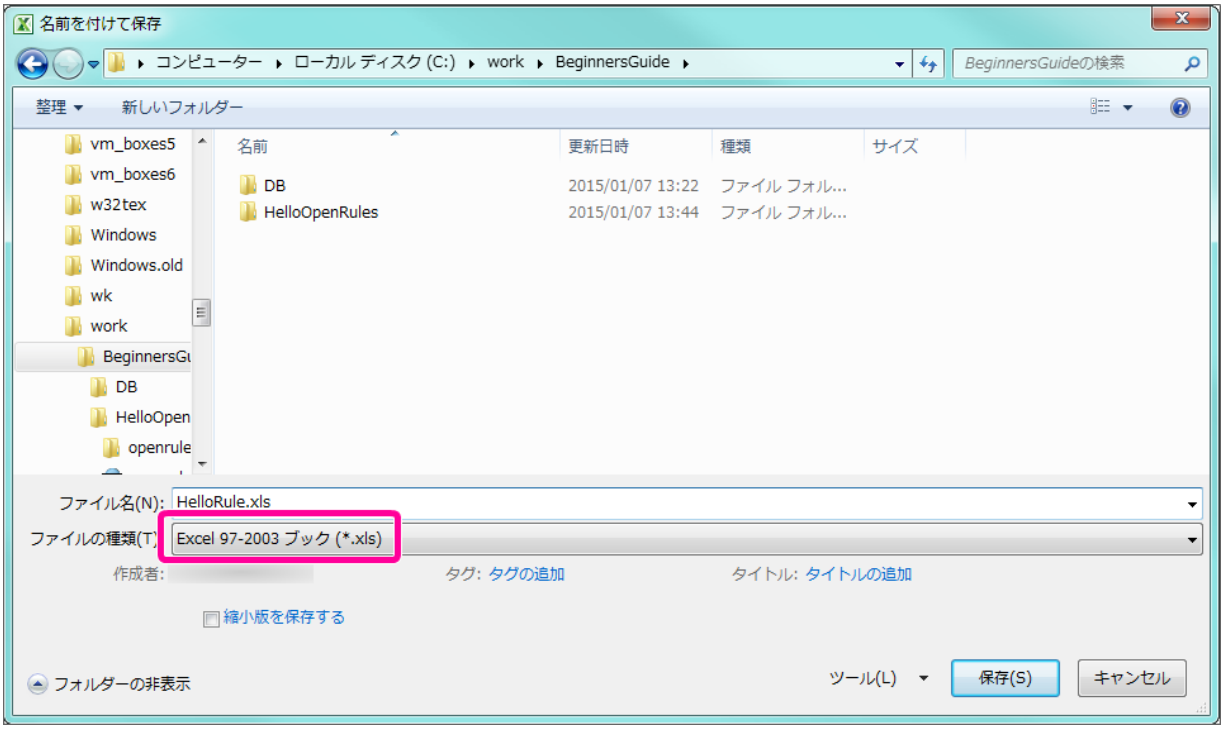
DecisionTable myRule	
Condition	Conclusion
年齢	メッセージ

各項目の論理名のセルも結合セルに設定してください。



7. これで、ヘッダが設定できましたので、一度保存し、次の手順に進みましょう。

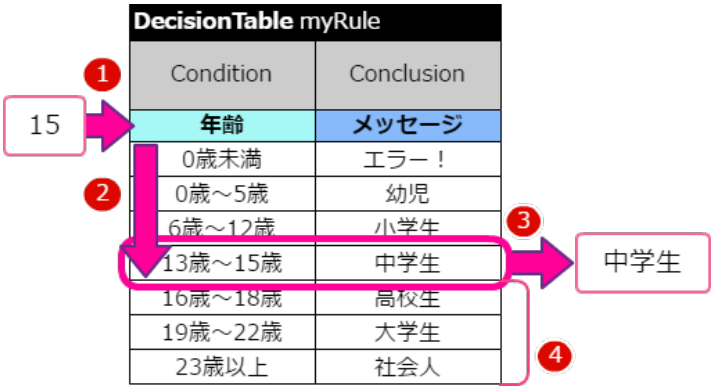
8. ファイルを保存する際には、形式を「Excel 97-2003ブック（*.xls）」としてください。
   xls形式以外で保存した場合、データソース定義に登録することができません。



ルールの表（ DecisionTable ）の条件と処理（返却する値）を設定する

ルールの表のヘッダができましたので、実際に評価（判断）する項目の基準値と、処理（返却する値）を設定していきましょう。

1. OpenRules の「 *DecisionTable* 」は、上から順に条件を評価（判断）し、条件に合致したらその行の処理（返却する値）を実行して終了するしくみです。
   以下の図は入力値から条件の判定後、値を返却するまでの一連の流れを表しています。

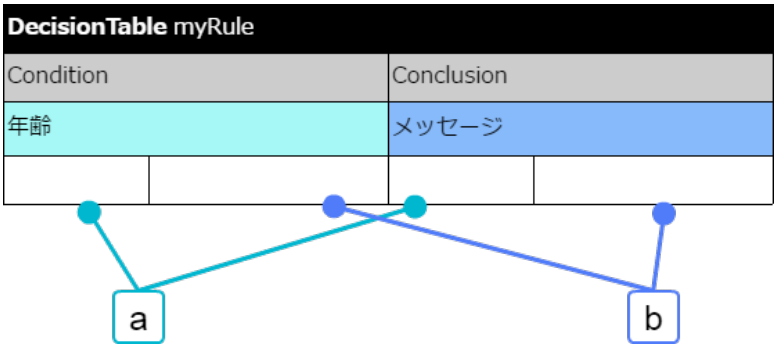


【処理の流れ】

1. 条件の「年齢」の入力値に「15」を受け取ります。
2. 入力値に基づいて条件を上から順に評価します。
3. 条件に合致した時点で評価を終了し、対応した行の値を返却します。

4. 合致した条件の行より下の行は評価されません。

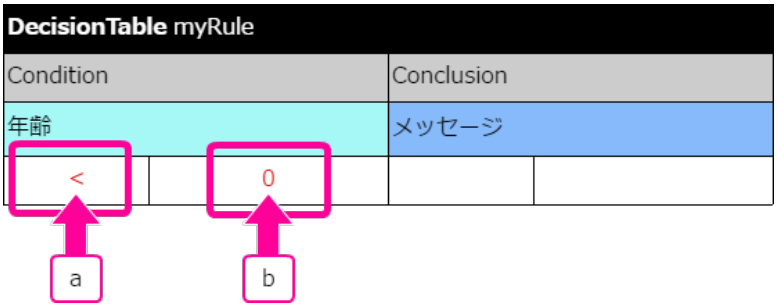
2. 「条件」と「処理」には、演算子と値を入力します。



a. 演算子
　条件（condition）または処理（conclusion）に対して、b.の値との比較方法を入力してください。

b. 値
　a.の演算子と比較、または代入する値を入力してください。

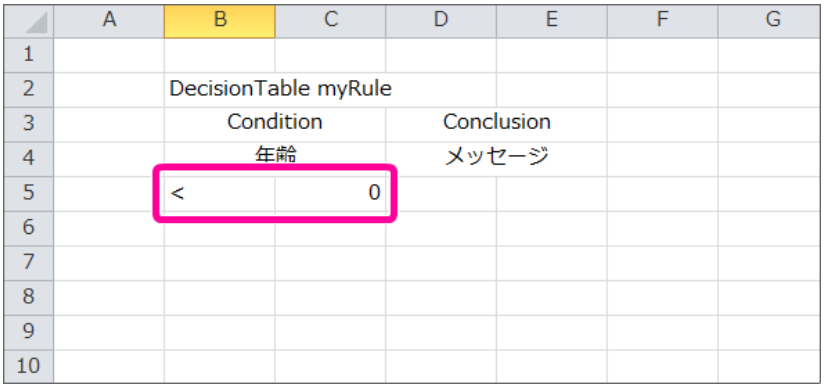
3. 最初の条件に、「年齢が0歳未満」と設定します。



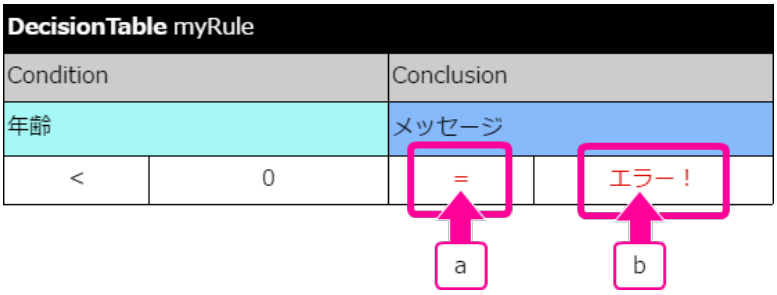
a. 演算子
　「未満」を表す「<」を入力してください。

b. 値
　「0」を入力してください。

Excelファイルのシート上には、以下の図の通り入力してください。



4. 「年齢が0歳未満」の条件に合致した場合のメッセージを設定します。



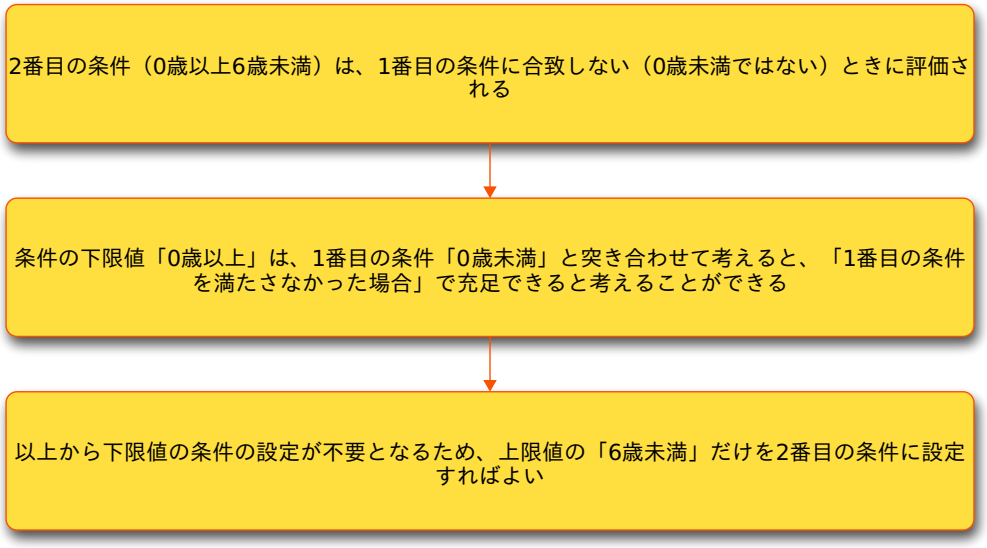
a. 演算子
　「代入」を表す「=」を入力してください。

b. 値
　「エラー！」と入力してください。

Excelファイルのシート上には、以下の図の通り入力してください。

	A	B	C	D	E	F
1						
2		DecisionTable myRule				
3		Condition		Conclusion		
4		年齢		メッセージ		
5		<		=	エラー！	
6						
7						
8						
9						
10						

5. 次の条件と処理の「0歳以上6歳未満の場合、”幼児”を返却する」を表にするために整理します。
2番目以降の条件は、「1番目（その条件より上の条件）に合致しない」かつ「2番目以降の条件」と評価（判断）されます。
条件にすべき内容は以下の流れで考えることができます。



DecisionTable myRule			
Condition		Conclusion	
年齢		メッセージ	
<	0	=	エラー！
<	6		

a　b

a. 演算子
「未満」を表す「<」を入力してください。

b. 値
「6」を入力してください。

Excelファイルのシート上には、以下の図の通り入力してください。

	A	B	C	D	E	F
1						
2		DecisionTable myRule				
3		Condition		Conclusion		
4		年齢		メッセージ		
5		<	0	=	エラー！	
6		<	6			
7						
8						
9						

6. 「年齢が6歳未満」の条件に合致した場合のメッセージを設定します。

DecisionTable myRule			
Condition		Conclusion	
年齢		メッセージ	
<	0	=	エラー！
<	6	=	幼児

a　b

a. 演算子
「代入」を表す「=」を入力してください。

b. 値
「幼児」と入力してください。

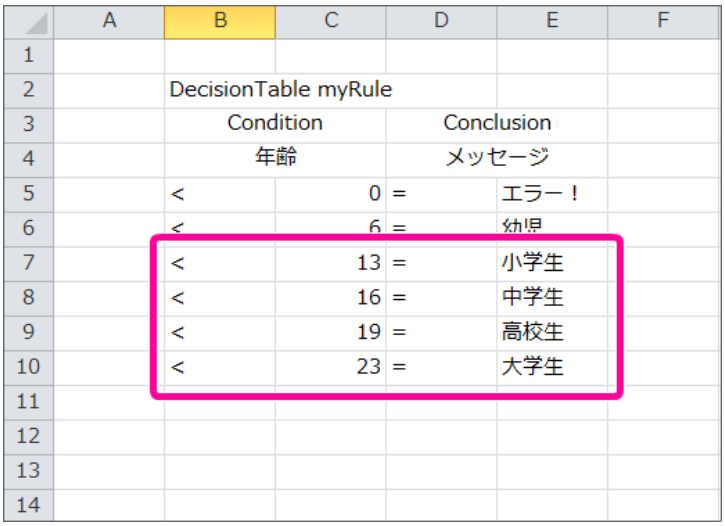
Excelファイルのシート上には、以下の図の通り入力してください。

	A	B	C	D	E	F
1						
2		DecisionTable myRule				
3		Condition		Conclusion		
4		年齢		メッセージ		
5		<	0	=	エラー！	
6		<	6	=	幼児	
7						
8						
9						
10						

7. 以下の3番目～6番目の条件は、2番目と同様に設定できますので、2番目と同様の形で設定します。
   - 6歳以上13歳未満の場合、「小学生」と返却する
   - 13歳以上16歳未満の場合、「中学生」と返却する
   - 16歳以上19歳未満の場合、「高校生」と返却する
   - 19歳以上23歳未満の場合、「大学生」と返却する

DecisionTable myRule			
Condition		Conclusion	
年齢		メッセージ	
<	0	=	エラー！
<	6	=	幼児
<	13	=	小学生
<	16	=	中学生
<	19	=	高校生
<	23	=	大学生

Excelファイルのシート上には、以下の図の通り入力してください。

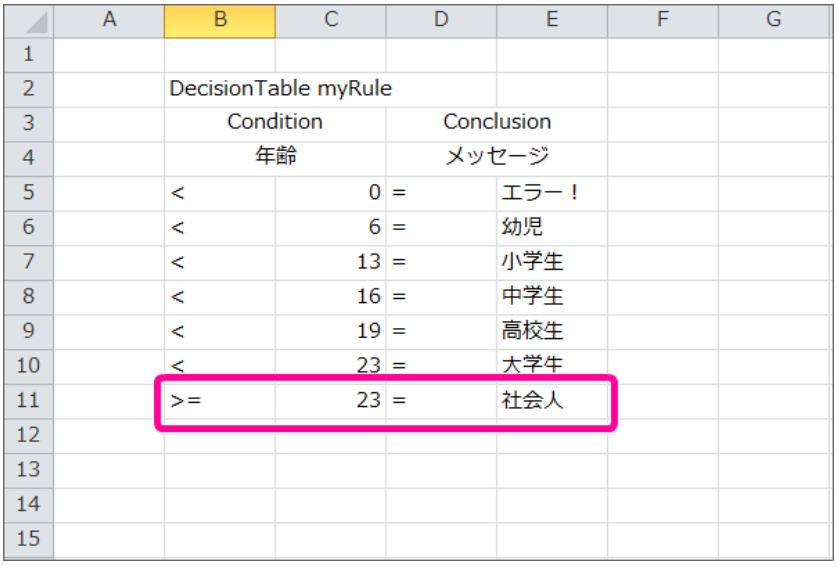


	A	B	C	D	E	F
1						
2		DecisionTable myRule				
3		Condition		Conclusion		
4		年齢		メッセージ		
5		<	0	=	エラー！	
6		<	6	=	幼児	
7		<	13	=	小学生	
8		<	16	=	中学生	
9		<	19	=	高校生	
10		<	23	=	大学生	
11						
12						
13						
14						

8. 最後の7番目の条件は、演算子を「以上」を表すものに変更し、2番目と同様の形で設定します。
メッセージには、「社会人」と入力します。

DecisionTable myRule			
Condition		Conclusion	
年齢		メッセージ	
<	0	=	エラー！
<	6	=	幼児
<	13	=	小学生
<	16	=	中学生
<	19	=	高校生
<	23	=	大学生
>=	23	=	社会人

Excelファイルのシート上には、以下の図の通り入力してください。



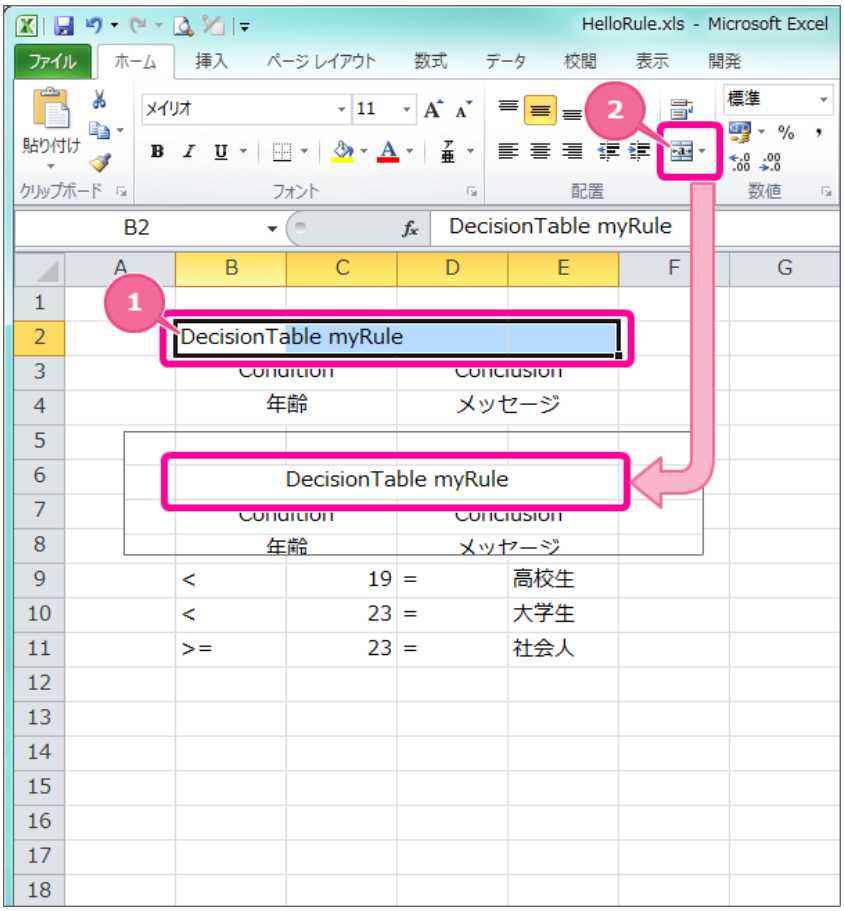
	A	B	C	D	E	F	G
1							
2		DecisionTable myRule					
3		Condition		Conclusion			
4		年齢		メッセージ			
5		<	0	=	エラー！		
6		<	6	=	幼児		
7		<	13	=	小学生		
8		<	16	=	中学生		
9		<	19	=	高校生		
10		<	23	=	大学生		
11		>=	23	=	社会人		
12							
13							
14							
15							

9. これで、ヘッダ・明細とも設定できましたので、一度保存し、次の手順に進みましょう。

ルールの表（ DecisionTable ）のレイアウトを整える

ルールの表のヘッダ・明細に必要な値を入力しましたので、レイアウトを整えます。

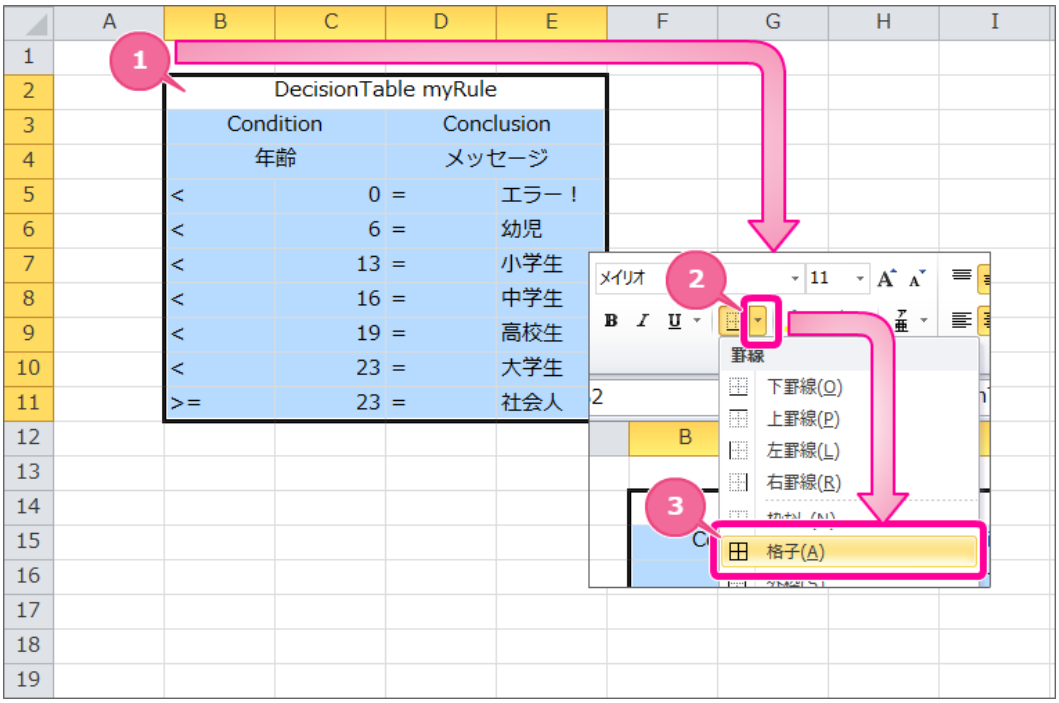
1. OpenRules のヘッダの1行目は、条件と処理の列の範囲で結合する必要がありますので、4列分を結合してください。



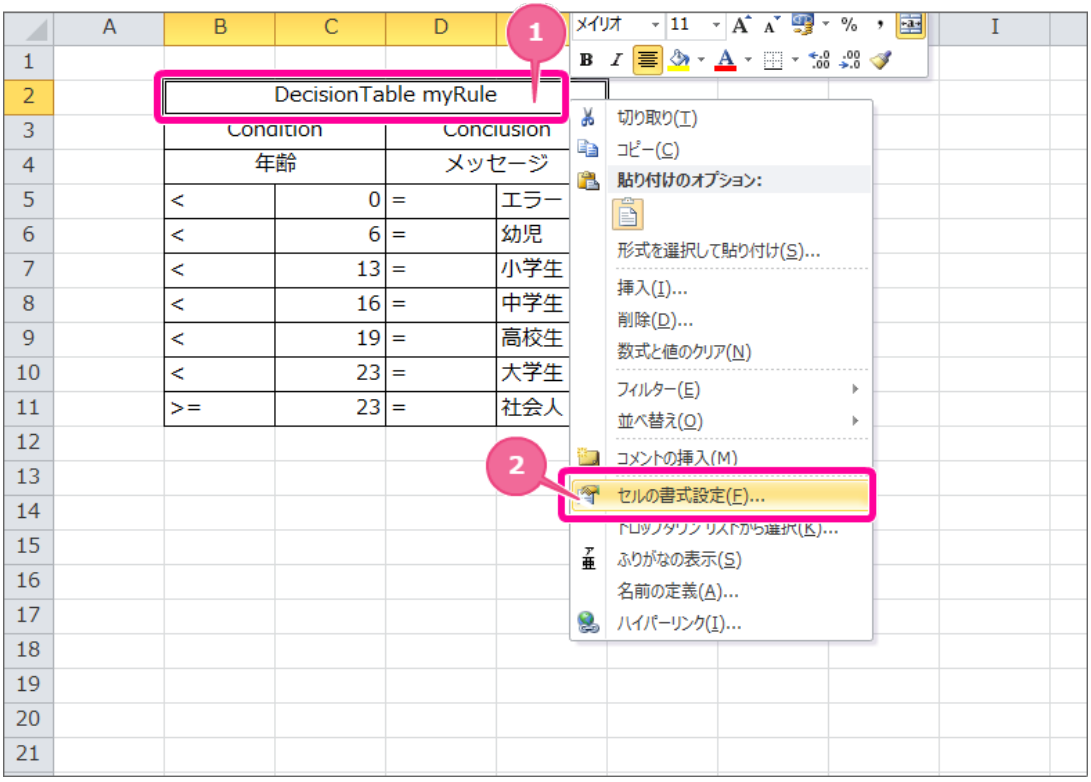
【ヘッダーのセルを結合する手順】

1. ヘッダの行のセル4つ分をドラッグで選択状態にしてください。
2. 「セルを結合して中央揃え」をクリックしてください。
3. これでヘッダーのセルを結合できました。

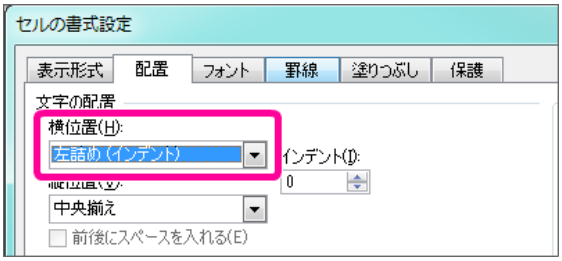
2. 各セルの境界線をわかりやすくするために罫線を引きます。
罫線の対象のセルを選択し、メニューの罫線から「格子」をクリックしてください。



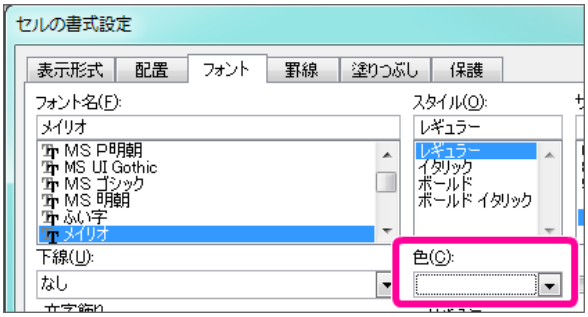
3. 各セルの書式を OpenRules の標準に合わせて設定します。
ヘッダーのセルの上で右クリックし、メニューの「セルの書式設定」から書式を設定しましょう。



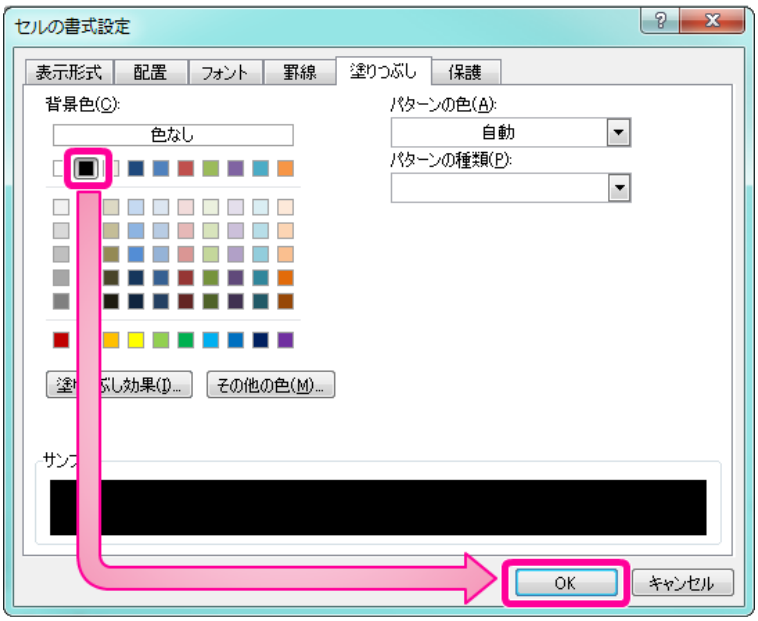
4. 「配置」タブをクリック後に、文字の配置の横位置を「左詰め（インデント）」に変更してください。



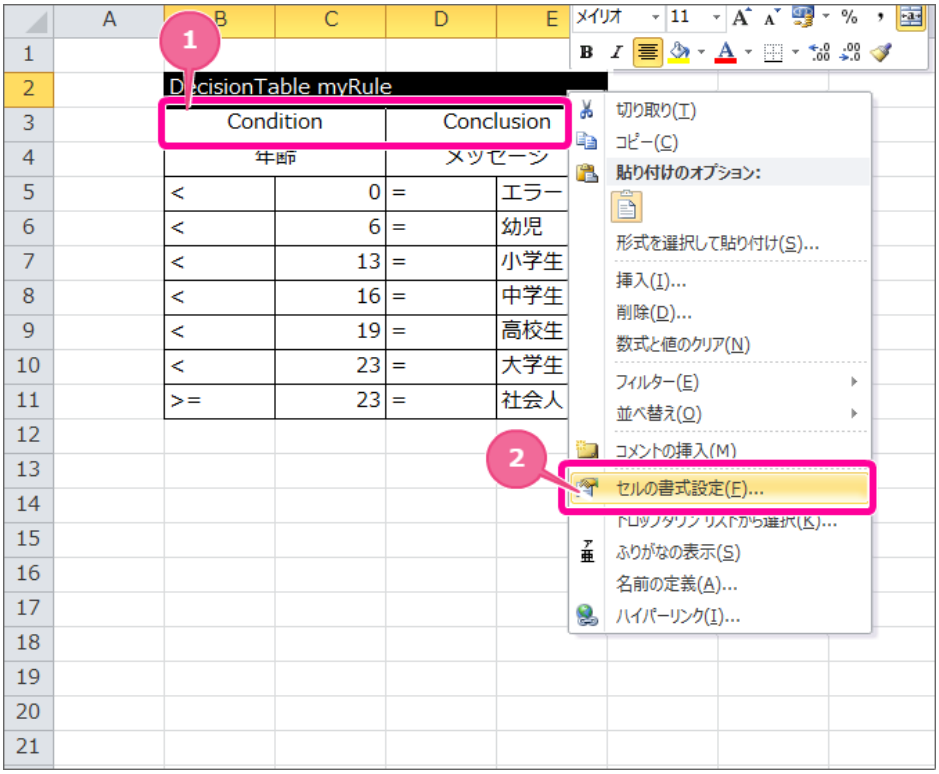
5. 「フォント」タブをクリック後に、色を「白」に変更してください。



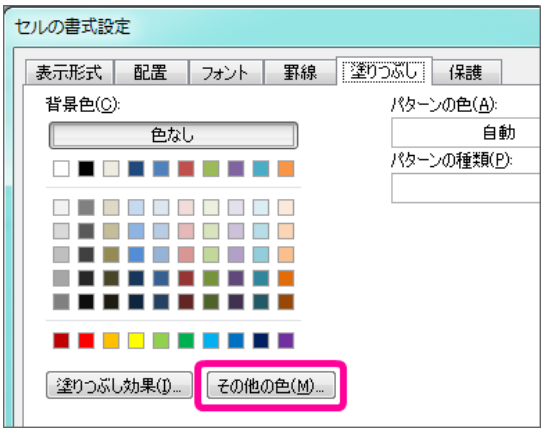
6. 「塗りつぶし」タブをクリック後に、背景色を「黒」に変更してください。
「OK」をクリックして変更内容を反映させてください。



7. 2行目のセルの上で右クリックし、メニューの「セルの書式設定」から書式を設定しましょう。



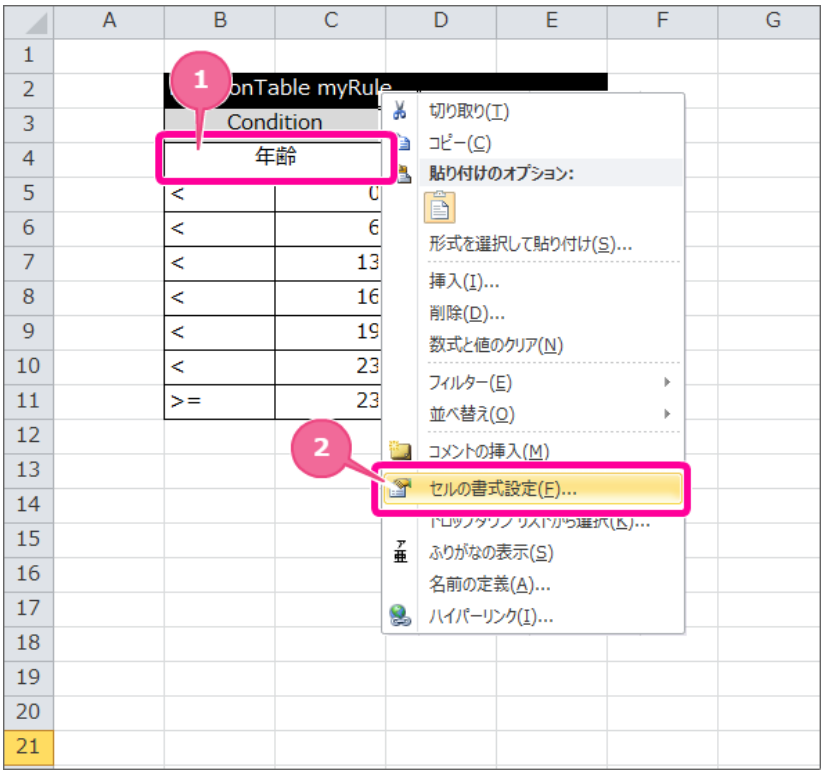
8. 「塗りつぶし」タブをクリック後に、「その他の色」をクリックしてください。



9. 「ユーザー設定」タブをクリック後に、以下の内容の色コードを入力し、最後に「OK」をクリックしてください。
   - カラーモデル：RGB
   - 赤（R）：217

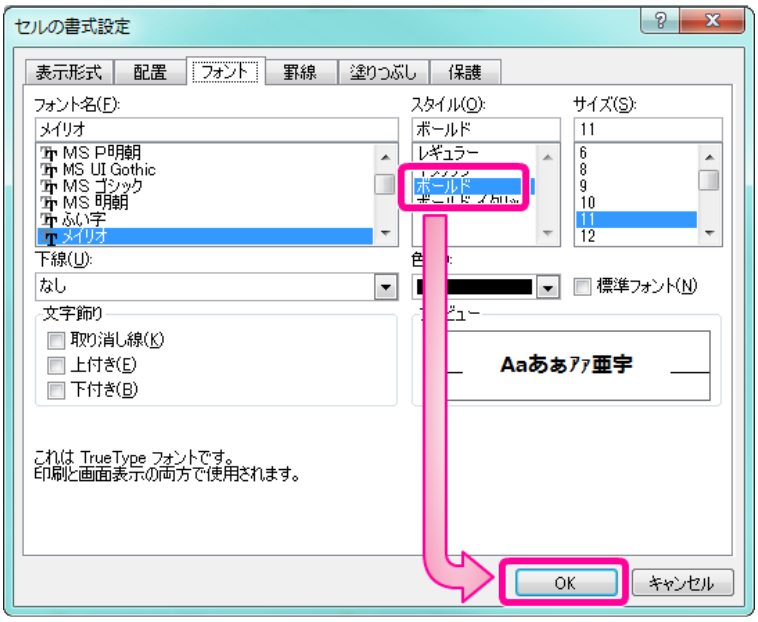
- 緑（G）：217
- 青（B）：217

10. 3行目の「条件」セルの上で右クリックし、メニューの「セルの書式設定」から書式を設定しましょう。

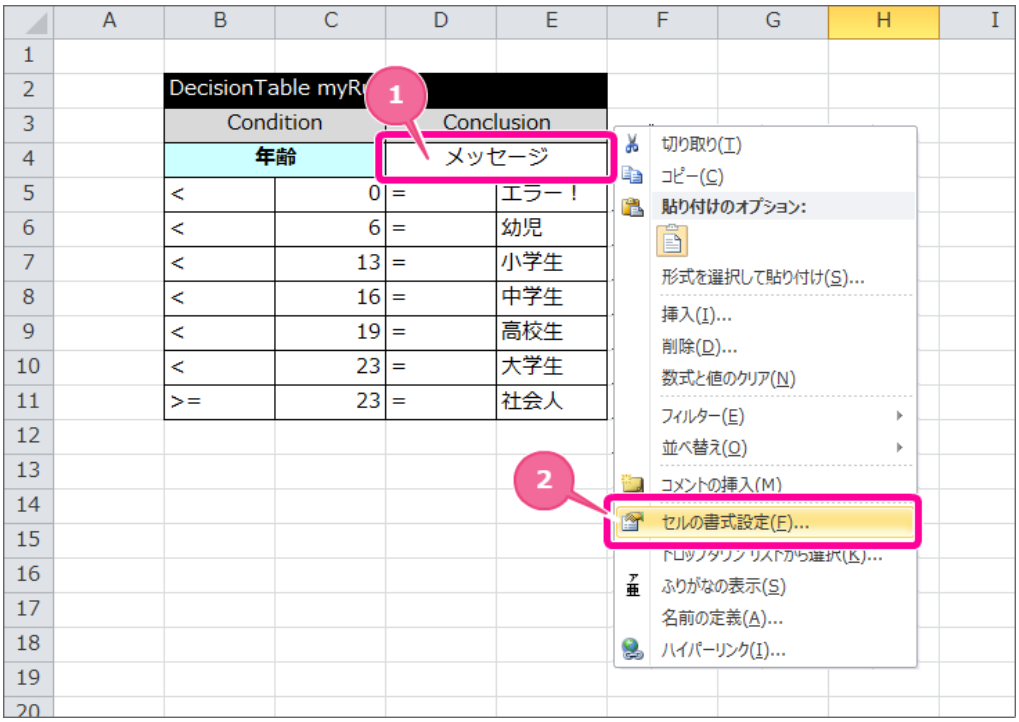


11. 「塗りつぶし」タブをクリック後に、「その他の色」をクリックしてください。
「標準」タブから中心の左上の色を選択してください。
（「ユーザー設定」から設定する場合、「RGB/204,255,255」として設定してください。）

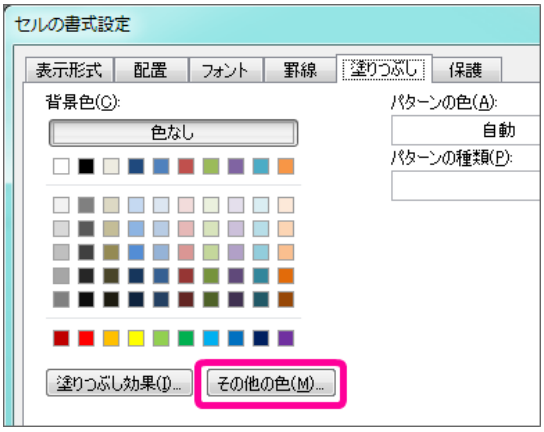
12. 「フォント」タブをクリック後、スタイルを「ボールド」に変更し、最後に「OK」をクリックしてください。



13. 3行目の「処理」セルの上で右クリックし、メニューの「セルの書式設定」から書式を設定しましょう。

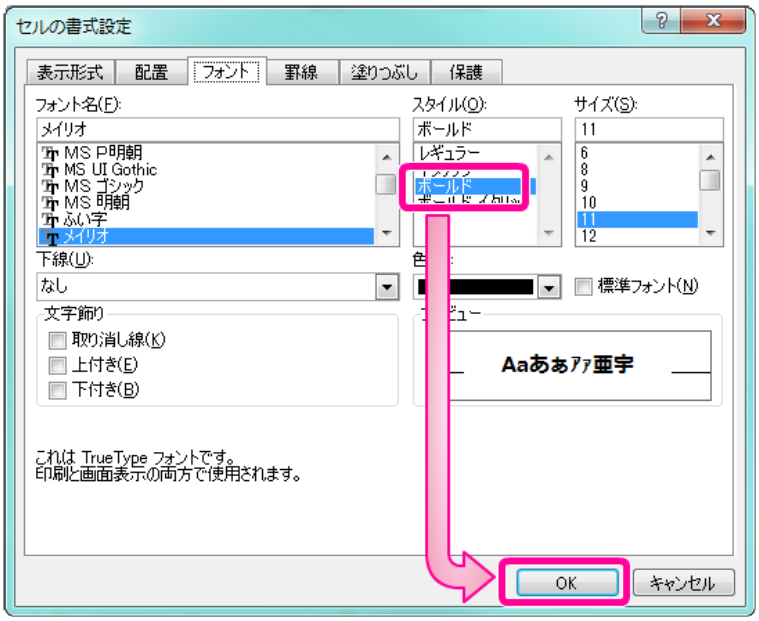


14. 「塗りつぶし」タブをクリック後に、「その他の色」をクリックしてください。



15. 「ユーザー設定」タブをクリック後に、以下の内容の色コードを入力し、最後に「OK」をクリックしてください。
    - カラーモデル：RGB
    - 赤（R）：141
    - 緑（G）：180
    - 青（B）：226

16. 「フォント」タブをクリック後、スタイルを「ボールド」に変更し、最後に「OK」をクリックしてください。



17. ルールの表（DecisionTable）が完成しましたので、ファイルを保存します。
次に実行するために必要な表を追加していきます。

	A	B	C	D	E	F
1						
2		DecisionTable myRule				
3		Condition		Conclusion		
4		年齢		メッセージ		
5		<	0	=	エラー！	
6		<	6	=	幼児	
7		<	13	=	小学生	
8		<	16	=	中学生	
9		<	19	=	高校生	
10		<	23	=	大学生	
11		>=	23	=	社会人	
12						
13						

Excelファイルに項目名のマッピング表（ Glossary ）を作成する

ルールの基本の表の「 *DecisionTable* 」では、条件や処理の項目に論理名を設定しましたが、 OpenRules を実行するためには論理名と物理名をマッピングする必要があります。
続いて、項目名のマッピング表（ *Glossary* ）を作成していきましょう。

Glossary glossary		
A 論理名	B オブジェクト	C 物理名
名前	InputObject	Name
年齢		Age
メッセージ	OutputObject	message

A. 論理名
論理名（業務担当者向けの用語）は、以下の形で構成されます。

1行目：列のタイプ（論理名/オブジェクト名/物理名）を表す名称
2行目～：ルールで利用する項目の論理名

B. オブジェクト名
オブジェクト名（項目をまとめるグループの単位）は、以下の形で構成されます。

1行目：列のタイプ（論理名/オブジェクト名/物理名）を表す名称
2行目～：ルールで利用する項目をまとめたオブジェクト名

C. 物理名
物理名（プログラム向けの名前）は、以下の形で構成されます。

1行目：列のタイプ（論理名/オブジェクト名/物理名）を表す名称
2行目～：ルールで利用する項目の物理名

項目名のマッピング表（ Glossary ）のヘッダ部分を作成する

表のヘッダは、 OpenRules のキーワードや項目名をルールに従って設定する必要があります。

最初にヘッダとして必要な内容を設定しましょう。

1. 先の手順で作成したExcelファイルを開きます。
2. マッピング表のキーワードは「 *Glossary* 」と記述します。

Glossary	テーブルの名前

3. テーブルの名称は、「glossary」と入力します。

Glossary	glossary

4. ヘッダには以下の形式でキーワードとテーブル名を入力してください。

```
[キーワード]+半角スペース+glossary
```

> **コラム**
>
> *Glossary* のテーブル名は”glossary”と命名してください。
> 命名しない場合には *Method* でGlossary型のテーブルをまとめる手順が必要です。
> 詳細は *Glossary* を参照してください。

Excelファイルに設定する各表（テーブル）は、他のテーブルと1つ以上行・列を空ける必要があります。
（以下の図では赤色の範囲には何も入力しないでください。）

	A	B	C	D	E	F	G
1							
2		DecisionTable myRule					
3		Condition		Conclusion			
4		**年齢**		**メッセージ**			
5		<	0	=	エラー！		
6		<	6	=	幼児		
7		<	13	=	小学生		
8		<	16	=	中学生		
9		<	19	=	高校生		
10		<	23	=	大学生		
11		>=	23	=	社会人		
12							
13		Glossary glossary					
14							
15							
16							

5. 2行目には、各列のラベルを設定します。
OpenRules の標準では、「Variable」「Business Concept」「Attribute」に設定されています。
今回はわかりやすくするために左から「論理名」「オブジェクト」「物理名」と設定してください。

Glossary glossary		
論理名	オブジェクト	物理名

> **コラム**
>
> *Glossary* の列ラベル部分は、自由に名前を設定することができます。
> 「Variable」「Business Concept」「Attribute」の並び替えはできません。

Excelファイルのシート上には、このように記述します。

11	>=	23	=	社会人	
12					
13	Glossary glossary				
14	論理名	オブジェクト	物理名		
15					
16					

6. これで、ヘッダが設定できましたので、一度保存し、次の手順に進みましょう。

項目名のマッピング表（ Glossary ）の項目・グループ・属性を設定する

項目名のマッピング表のヘッダができましたので、項目グループと物理名のマッピング情報を設定していきましょう。

1. 論理名の内容を入力します。
   上から順に「名前」「年齢」「メッセージ」と入力します。

Glossary glossary		
論理名	オブジェクト	物理名
名前		
年齢		
メッセージ		

   Excelファイルのシート上には、このように記述します。

11	>=	23	=	社会人	
12					
13	Glossary glossary				
14	論理名	オブジェクト	物理名		
15	名前				
16	年齢				
17	メッセージ				
18					
19					

2. 続いて、項目名の内容を設定します。
   項目名は、半角英数字で用語に対応した名称を設定する必要があります。
   このシナリオでは、上から順に「Name」「Age」「message」と入力します。

Glossary glossary		
論理名	オブジェクト	物理名
名前		Name
年齢		Age
メッセージ		message

   Excelファイルのシート上には、このように記述します。

11	>=	23	=	社会人	
12					
13	Glossary glossary				
14	論理名	オブジェクト	物理名		
15	名前		Name		
16	年齢		Age		
17	メッセージ		message		
18					
19					

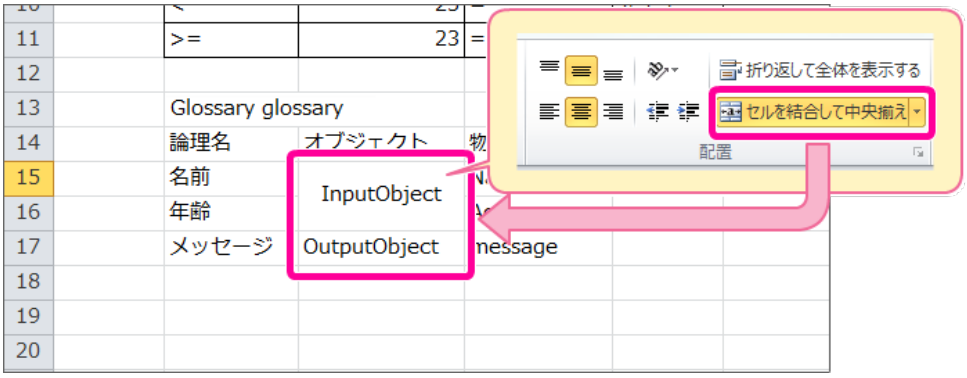
3. 業務用語のマッピング表（Glossary）では、実行時に初期化するためのグループで属性をまとめます。
   このシナリオでは、入力項目のオブジェクト、出力項目のオブジェクトでまとめます。
   用語に「名前」を設定した行に「InputObject」、「メッセージ」を設定した行に「OutputObject」を設定します。

Glossary glossary		
論理名	オブジェクト	物理名
名前	InputObject	Name
年齢		Age
メッセージ	OutputObject	message

「InputObject」には、2つの属性が含まれているため、「InputObject」は対象の属性の行数のセルを結合してください。

Glossary glossary		
論理名	オブジェクト	物理名
名前	InputObject	Name
年齢		Age
メッセージ	OutputObject	message

Excelファイルのシート上には、このように記述します。
「InputObject」のセルは結合対象のセル範囲を選択後、「セルを結合して中央揃え」をクリックしてください。

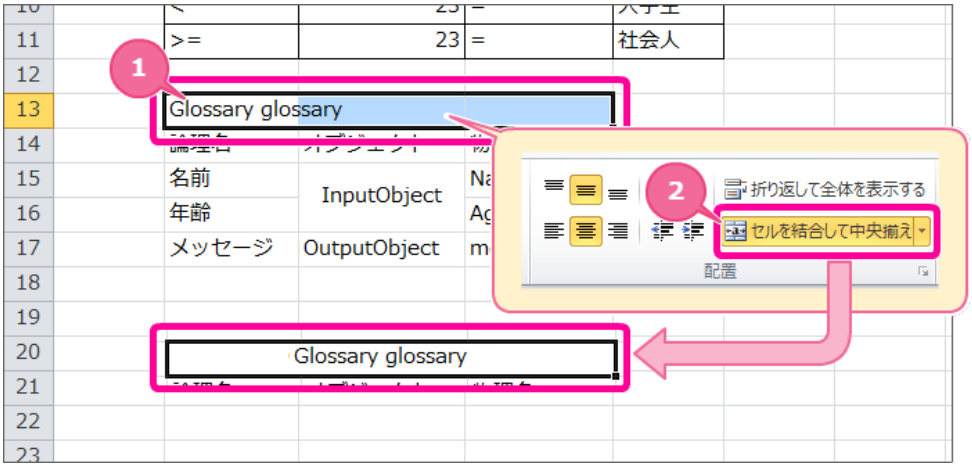


4. これで、業務用語のマッピング表（Glossary）の各項目が設定できましたので、一度保存し、次の手順に進みましょう。

業務用語のマッピング表（ Glossary ）のレイアウトを整える

ルールの表のヘッダ・明細に必要な値を入力しましたので、レイアウトを整えます。

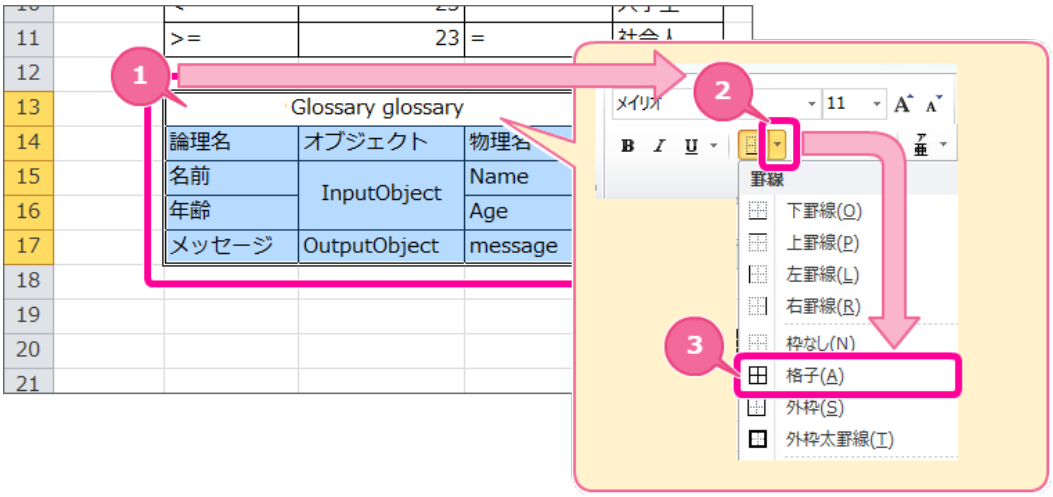
1. OpenRules のヘッダの1行目は、「論理名」「オブジェクト」「物理名」の列の範囲で結合する必要がありますので、3列分のセルを結合してください。



【ヘッダーのセルを結合する手順】

1. ヘッダの行のセル3つ分をドラッグで選択状態にしてください。
2. 「セルを結合して中央揃え」をクリックしてください。
3. これでヘッダーのセルを結合できました。

2. 各セルの境界線をわかりやすくするために罫線を引きます。
罫線の対象のセルを選択し、メニューの罫線から「格子」をクリックしてください。

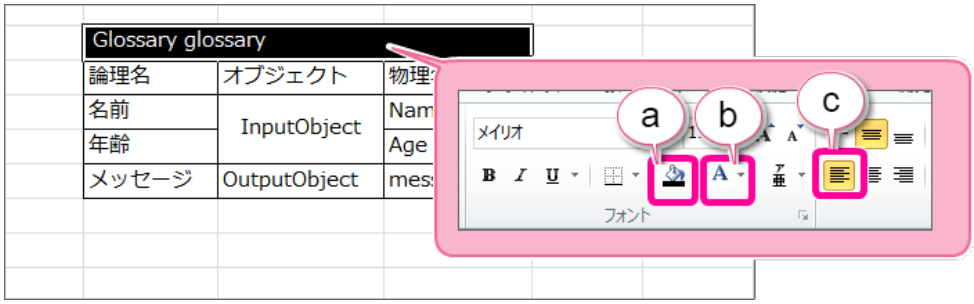


3. 各セルの書式を OpenRules の標準に合わせて設定します。
   1行目のヘッダを以下の通りに設定します。

   a. セルの背景色（塗りつぶし）：黒

   b. セルの文字の色：白

   c. セルの揃え方：左揃え

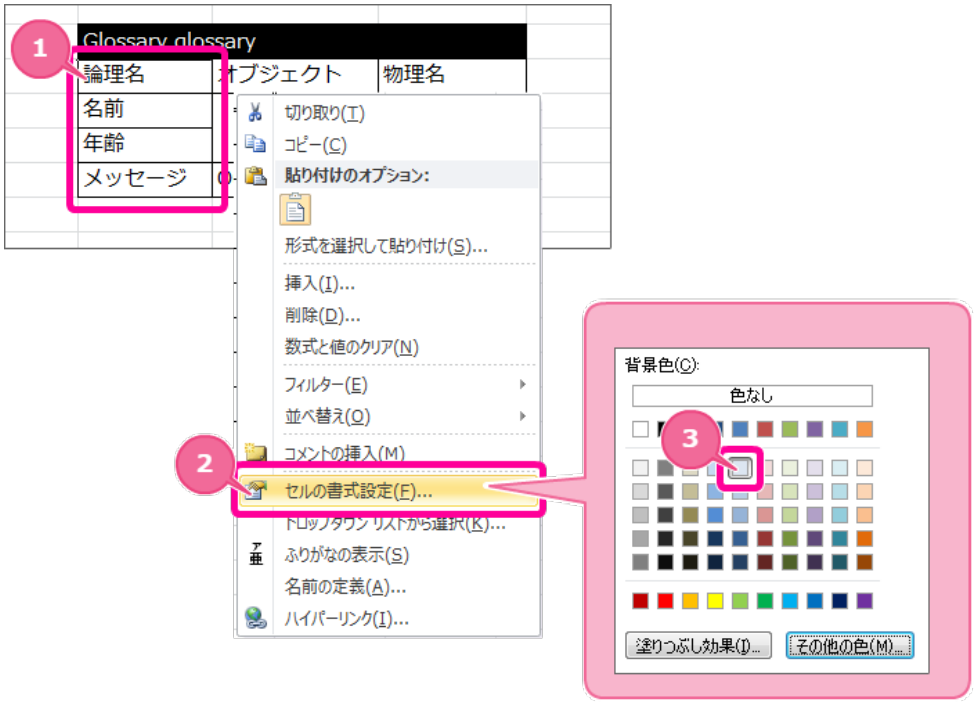


4. 論理名の列のヘッダ・明細を以下の手順で設定してください。

   1. 論理名の列のセル範囲を選択してください。

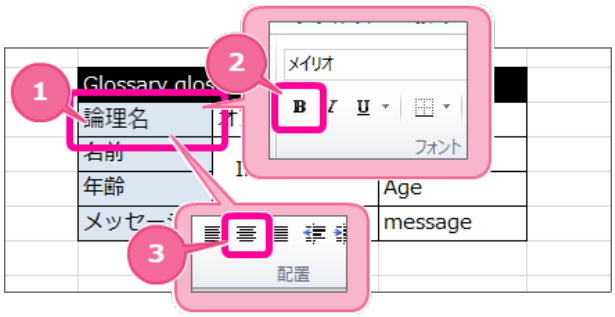
   2. 選択範囲で右クリックし、メニューから「セルの書式設定」をクリックしてください。

   3. 「塗りつぶし」タブをクリック後、画像の場所の色をクリックしてください。
      （「ユーザー設定」から設定する場合、「RGB/220,230,241」として設定してください。）

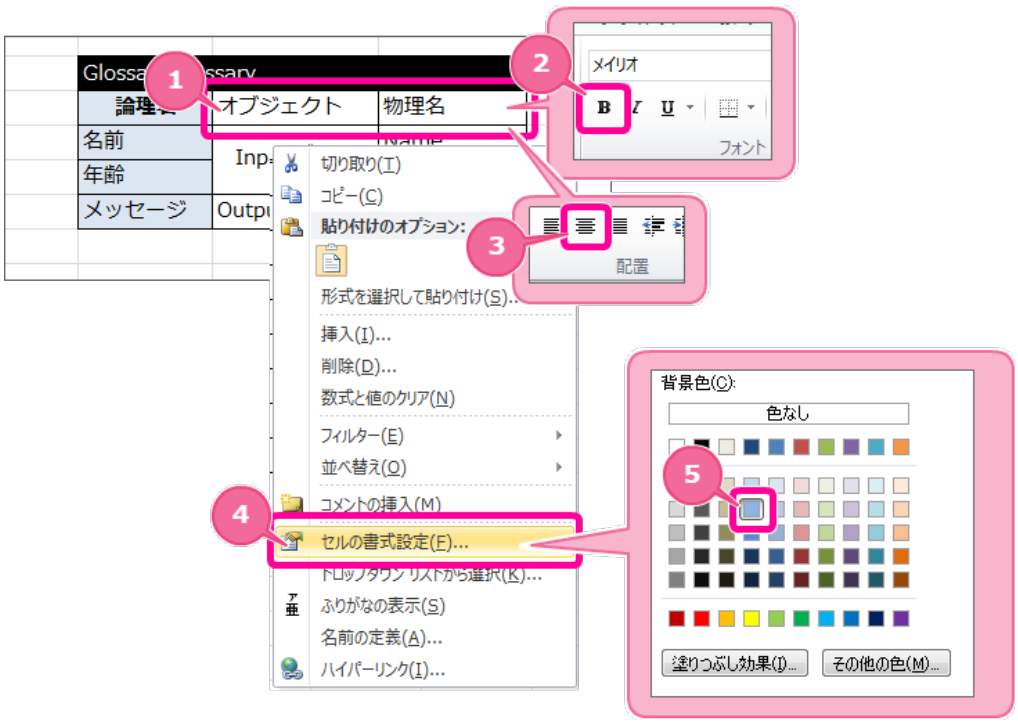


5. 論理名の列のヘッダを以下の手順で設定してください。

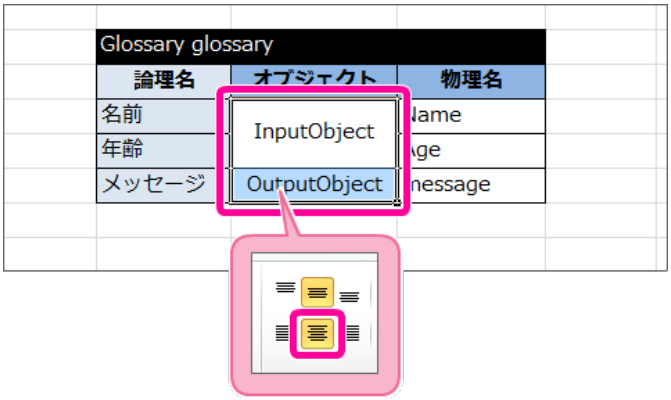
1. 論理名のヘッダのセルを選択してください。
2. フォントから「ボールド」をクリックしてください。
3. 配置から横方向の文字揃えの「中央揃え」をクリックしてください。



6. オブジェクト・物理名のヘッダを以下の手順で設定してください。
   1. オブジェクト・物理名のヘッダのセルを選択してください。
   2. フォントから「ボールド」をクリックしてください。
   3. 配置から横方向の文字揃えの「中央揃え」をクリックしてください。
   4. 選択範囲で右クリックし、メニューから「セルの書式設定」をクリックしてください。
   5. 「塗りつぶし」タブをクリック後、画像の場所の色をクリックしてください。
      （「ユーザー設定」から設定する場合、「RGB/141,180,226」として設定してください。）



7. 最後にオブジェクトの明細の揃え方を中央揃えにしてください。

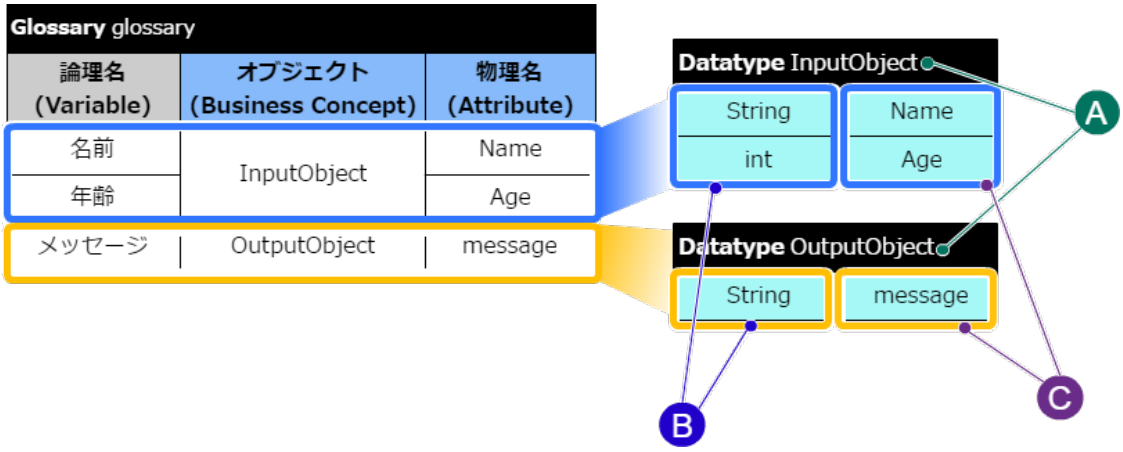


8. 項目名のマッピング表（Glossary）が完成しましたので、ファイルを保存します。

次にルールで利用する項目やデータ型の定義の表を追加していきます。

Excelファイルに項目とデータ型の定義（ Datatype ）を作成する

項目とデータ型の定義の表の「 *Datatype* 」で利用する処理用の項目を設定していきましょう。
「 *Datatype* 」は、「 *Glossary* 」で定義したオブジェクトと物理名に基づいて各項目に対応するデータ型を設定します。

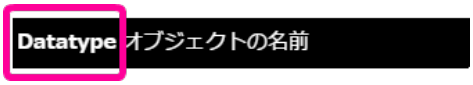


A. オブジェクト名
Glossary で定義したオブジェクトです。
B. データ型
OpenRules ではこのデータ型と項目の定義に基づいて、処理を行います。
C. 項目名（物理名）
Glossary で定義した項目の物理名です。

項目とデータ型の定義（ Datatype ）のヘッダ部分を作成する

表のヘッダは、 OpenRules のキーワードや項目名をルールに従って設定する必要があります。
最初にヘッダとして必要な内容を設定しましょう。

1. 先の手順で作成したExcelファイルを開きます。
2. 項目とデータ型の定義のキーワードは「 *Datatype* 」と記述します。



3. ヘッダには以下の形式でキーワードとテーブル名を入力してください。

```
[キーワード]+半角スペース+[オブジェクト名]
```

> **コラム**
> テーブルの名称は、業務用語のマッピング表（ *Glossary* ）で「Business Concept」（このハンズオンで「オブジェクト」とした列）にする必要があります。
> 入力項目のオブジェクト「InputObject」と入力してください。

この「 *Datatype* 」も、他の表と同様に、同じシートに作成する場合には、1つ以上行・列を空けて作成します。
以下の図中の赤色部分には何も入力しないでください。

11	>=	23	=
12			
13	Glossary glossary		
14	論理名	オブジェクト	物理名
15	名前	InputObject	Name
16	年齢		Age
17	メッセージ	OutputObject	message
18			
19	Datatype InputObject		
20			
21			
22			
23			

4. *Datatype* は、オブジェクト単位に表を作成する必要があります。
   同様の手順で出力項目のグループ（OutputObject）のヘッダも作成しましょう。

Datatype	InputObject

Datatype	OutputObject

ここまで設定した状態のExcelのシートは、以下の通りです。

Glossary glossary		
論理名	オブジェクト	物理名
名前	InputObject	Name
年齢		Age
メッセージ	OutputObject	message

Datatype InputObject　　Datatype OutputObject

5. これで、ヘッダが設定できましたので、一度保存し、次の手順に進みましょう。

項目とデータ型の定義（ Datatype ）のデータ型と項目名を設定する

項目とデータ型の定義のヘッダができましたので、項目とデータ型を設定していきましょう。

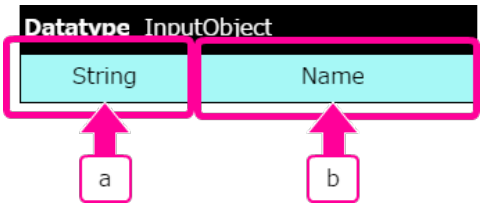
1. 最初に入力項目のグループ（InputObject）の表を作成していきましょう。
   「 *Datatype* 」では、左側にデータ型、右側に業務用語のマッピング表（ *Glossary* ）で設定した項目の物理名を入力してください。

Datatype	InputObject
データ型	項目の物理名

2. 入力項目のオブジェクトの項目「名前」を定義します。

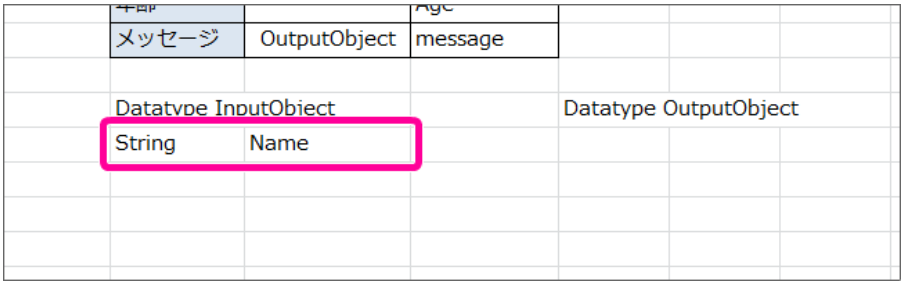
   a. データ型を入力してください。
      この場合は「名前」に対応するデータ型のため、「String」としてください。

   b. 物理名を入力してください。
      この場合は「名前」に対応する物理名のため、「Name」としてください。



> コラム
>
> *Datatype* では、データ型・項目の物理名とも大文字・小文字を区別しています。
> *Glossary* で定義した内容とずれないように記述してください。

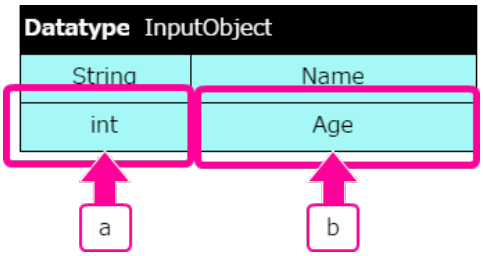
Excelファイルのシート上には、このように記述します。



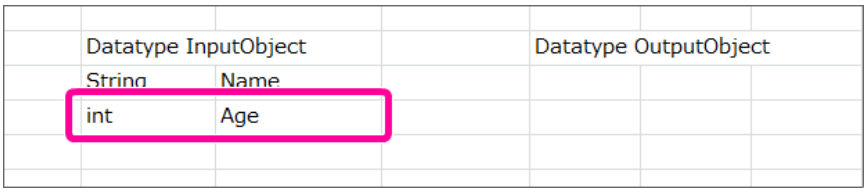
3. 次に「年齢」を定義します。

   a. データ型を入力してください。
      この場合は「年齢」に対応するデータ型のため、「int」としてください。

   b. 物理名を入力してください。
      この場合は「年齢」に対応する物理名のため、「Age」としてください。



Excelファイルのシート上には、このように記述します。

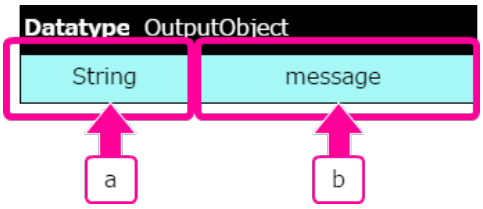


4. これで、入力項目のグループ（InputObject）の表が設定できましたので、次に出力項目のグループ（OutputObject）を作成していきましょう。

5. 出力項目のグループの項目「メッセージ」を定義します。

   a. データ型を入力してください。
      この場合は「メッセージ」に対応するデータ型のため、「String」としてください。

   b. 物理名を入力してください。
      この場合は「メッセージ」に対応する物理名のため、「message」としてください。



Excelファイルのシート上には、このように記述します。

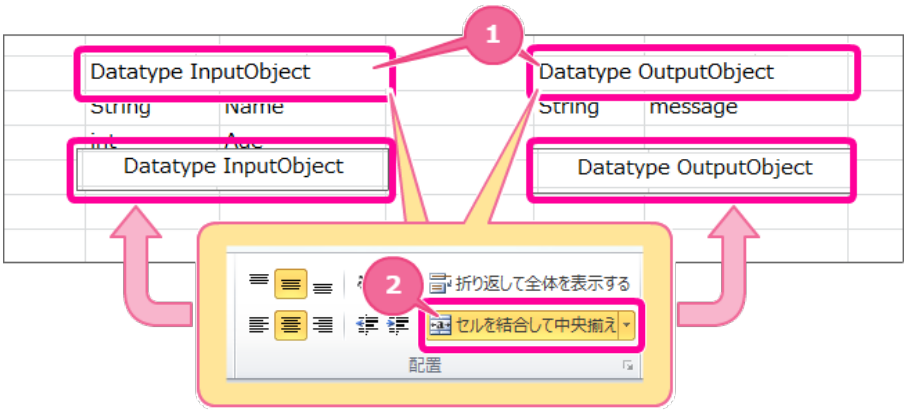
	Datatype InputObject		Datatype OutputObject	
	String	Name	String	message
	int	Age		

6. これで、項目とデータ型の定義（Datatype）が設定できましたので、一度保存し、次の手順に進みましょう。

項目とデータ型の定義（ Datatype ）のレイアウトを整える

ルールの表のヘッダ・明細に必要な値を入力しましたので、レイアウトを整えます。

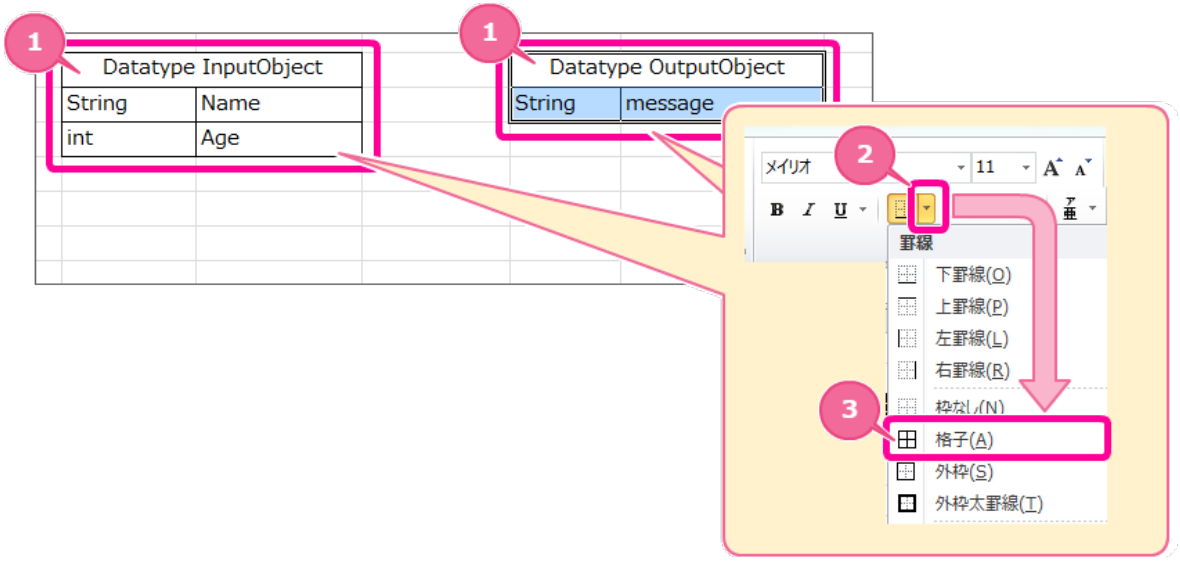
1. OpenRules のヘッダの1行目は、「データ型」「項目名」の列の範囲で結合する必要がありますので、2列分を結合します。



【ヘッダーのセルを結合する手順】

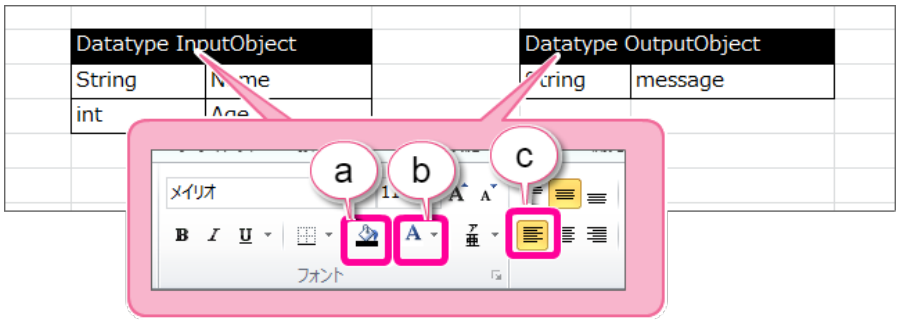
1. ヘッダの行のセル2つ分をドラッグで選択状態にしてください。
2. 「セルを結合して中央揃え」をクリックしてください。
3. これでヘッダーのセルを結合できました。

2. 各セルの境界線をわかりやすくするために罫線を引きます。
罫線の対象のセルを選択し、メニューの罫線から「格子」をクリックしてください。



3. 各セルの書式を OpenRules の標準に合わせて設定してください。
1行目のヘッダを以下の通りに設定します。

   a. セルの揃え方：左揃え
   b. セルの背景色（塗りつぶし）：黒
   c. セルの文字の色：白



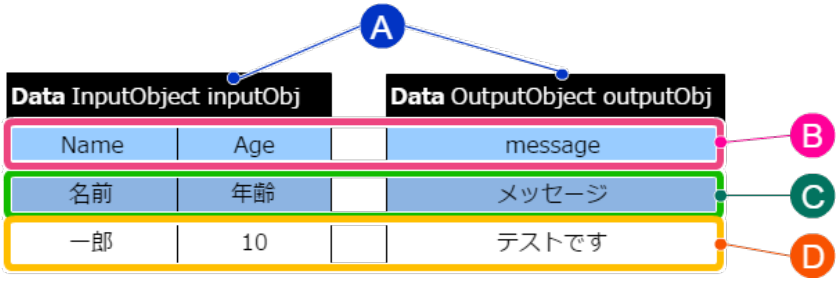
4. 残りの明細のセルを以下の手順で設定してください。

    1. 明細のセル範囲を選択してください。
    2. 配置から横方向の文字揃えの「中央揃え」をクリックしてください。
    3. 選択範囲で右クリックし、メニューから「セルの書式設定」をクリックしてください。
    4. 「塗りつぶし」タブをクリック後、「その他の色」をクリックしてください。
    5. 画像の場所の色をクリックしてください。
       （「ユーザー設定」から設定する場合、「RGB/204,255,255」として設定してください。）

5. 項目とデータ型の定義（Datatype）が完成しましたので、ファイルを保存します。
   次に実行するために必要な表を追加していきます。

Excelファイルに項目の初期値（ Data ）を作成する

項目の初期値の表の「 *Data/Variable* 」で利用する処理用の項目を設定していきましょう。



A. **オブジェクト、インスタンス**
Glossary で定義したオブジェクトです。
実行時にインスタンスを生成するためのインスタンス名をオブジェクトの後に記述します。

B. **物理名**
Glossary で定義した物理名です。

C. **論理名**
Glossary で定義した論理名です。

D. **初期値**
オブジェクトの初期化で利用します。

項目の初期値（ Data ）のヘッダ部分を作成する

表のヘッダは、 OpenRules のキーワードや項目名をルールに従って設定する必要があります。
最初にヘッダとして必要な内容を設定しましょう。

1. 先の手順で作成したExcelファイルを開きます。
2. 項目の初期値を設定するための表のキーワードは「 *Data/Variable* 」のいずれかを記述します。
   今回のハンズオンでは、「Data」と定義して進めていきます。

Data	オブジェクトの名前

3. キーワードに続けて、テーブルで定義するためのオブジェクトの型は、業務用語のマッピング表（ *Glossary* ）で「Business Concept」（このハンズオンで「オブジェクト」とした列）を設定してください。
   入力項目のグループ「InputObject」と入力します。

Data	InputObject

4. オブジェクト名「InputObject」に続いて、 *Data/Variable* のヘッダにはインスタンス名を記述します。
   今回のハンズオンでは、インスタンス名を「inputObj」と入力します。

Data	InputObject	inputObj

5. 「InputObject」の初期値を設定する表のヘッダには以下の形式でキーワードとオブジェクトの型、インスタンス名を入力してください。

   ```
   [キーワード]+半角スペース+[オブジェクト名]+半角スペース+[インスタンス名]
   ```

   この「 *Data/Variable* 」も、他の表と同様に、他のテーブルと1つ以上行・列を空けて作成する必要があります。
   Excelのシートの状況は以下の通りです。

19	Datatype InputObject			Datatype OutputObject	
20	String	Name		String	Name
21	int	Age			
22					
23	Data InputObject inputObj				
24					
25					
26					
27					
28					

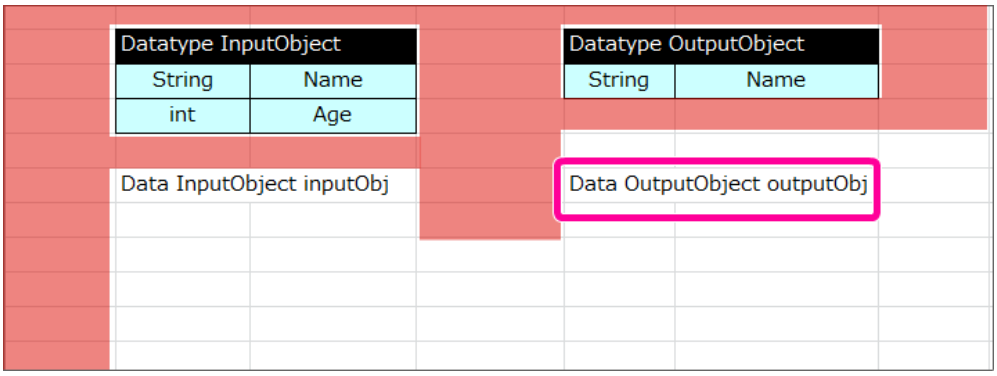
6. 同様の手順で出力項目のオブジェクト（OutputObject）のヘッダも作成しましょう。
   出力項目のインスタンス名は「outputObj」とします。

Data	OutputObject outputObj

7. 「OutputObject」の初期値を設定する表のヘッダも、先のInputObjectと同じように以下の形式でキーワードとオブジェクトの型、インスタンス名を入力してください。

```
[キーワード]+半角スペース+[オブジェクト名]+半角スペース+[インスタンス名]
```

Excelのシートの状況は以下の通りです。

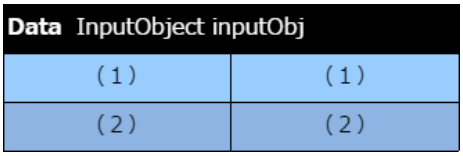


8. これで、ヘッダが設定できましたので、一度保存し、次の手順に進みましょう。

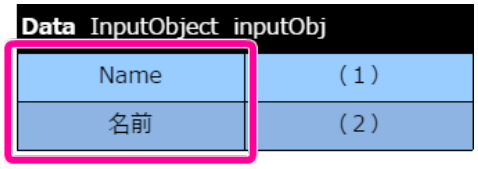
項目の初期値（ Data ）のサブヘッダ部分を作成する

Data/Variable は、サブヘッダに各オブジェクトの項目の物理名と論理名を設定します。

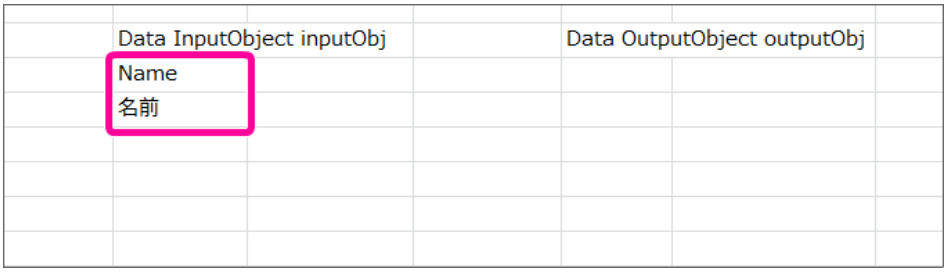
1. 最初に入力項目のオブジェクト「InputObject」のサブヘッダを設定していきましょう。
   以下の図中のサブヘッダの(1)に物理名、(2)に論理名を入力します。



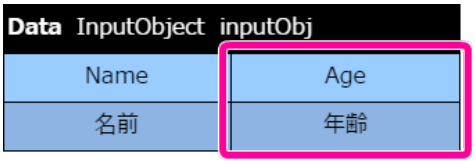
2. 左の列に「名前」の項目を設定しますので、(1)物理名に「Name」、(2)論理名に「名前」と入力してください。



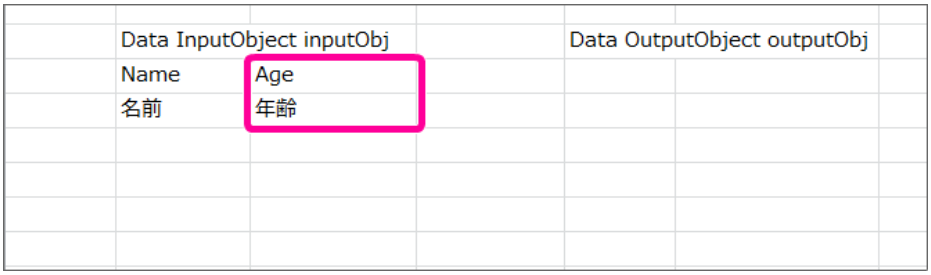
Excelのシートの状況は以下の通りです。



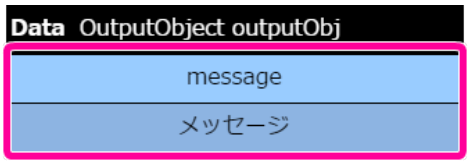
3. 右の列に「年齢」の項目を設定しますので、(1)物理名に「Age」、(2)論理名に「年齢」と入力してください。



Excelのシートの状況は以下の通りです。



4. 同様に出力項目のオブジェクト「OutputObject」のサブヘッダを設定していきましょう。
   サブヘッダの1行目に物理名の「message」、2行目に論理名の「メッセージ」と入力してください。



Excelのシートの状況は以下の通りです。

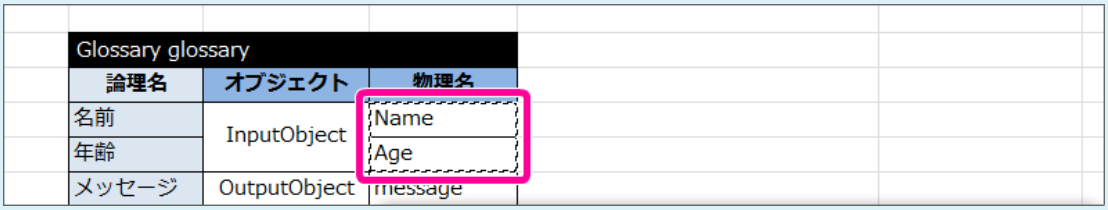
	Data InputObject inputObj		Data OutputObject outputObj
	Name	Age	message
	名前	年齢	メッセージ

コラム

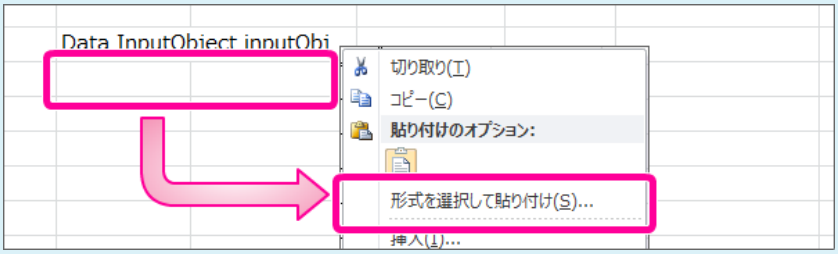
Data/Variable を *Glossary* から便利に作成するには

Glossary を先に作ってから *Data/Variable* を作成する場合、以下の手順で作成すると便利です。

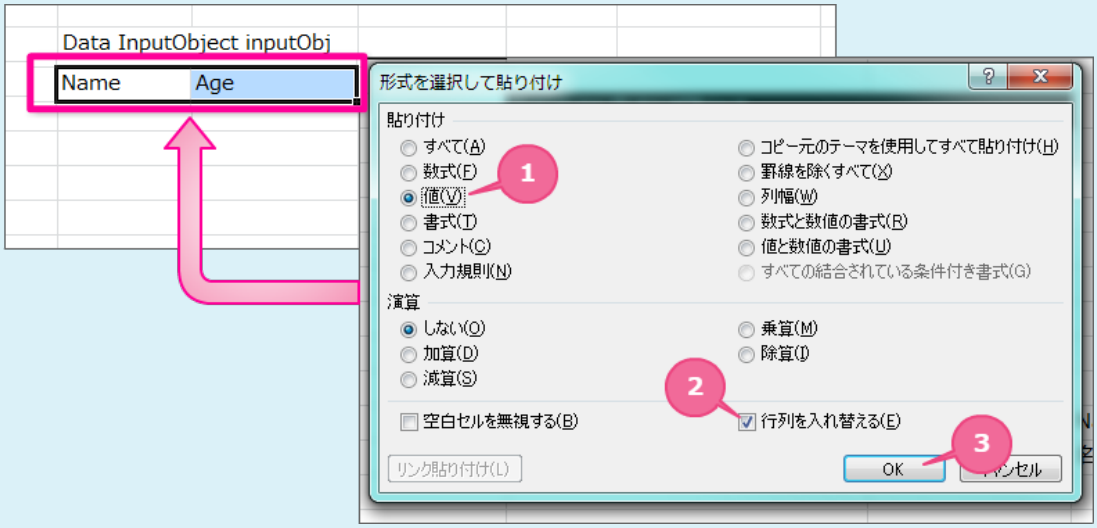
1. 作成済みの *Glossary* から特定のオブジェクトに含まれる物理名のセルをコピーしてください。



2. *Data/Variable* を作成する場所のセルで右クリックし、「形式を選択して貼り付け」をクリックしてください。



3. 「値」のラジオボタンをオンにし、「行列を入れ替える」チェックボックスをオンにしてから「OK」をクリックしてください。
以下のように *Data/Variable* への設定が簡単に行えます。



項目の初期値（ Data ）の項目と初期値を設定する

項目の初期値の表のヘッダ、サブヘッダができましたので、初期値を設定していきましょう。

1. 作成した表の明細（以下の図の白いセル(1)）に項目をインスタンス化（初期化）する際の初期値を設定します。
ルールの実行時に値が設定されていない場合にはエラーが発生するため、空文字以外の値を設定するようにしてください。

Data InputObject inputObj	
Name	Age
名前	年齢
（1）	（1）

2. 項目「名前」の初期値として、左の列の明細に「太郎」と入力してください。

Data InputObject inputObj	
Name	Age
名前	年齢
太郎	（1）

Excelのシートの状況は以下の通りです。

	Data InputObject inputObj		Data OutputObject outputObj
	Name	Age	message
	名前	年齢	メッセージ
	太郎		

3. 項目「年齢」の初期値として、右の列の明細に「0」と入力してください。

Data InputObject inputObj	
Name	Age
名前	年齢
太郎	0

Excelのシートの状況は以下の通りです。

	Data InputObject inputObj		Data OutputObject outputObj
	Name	Age	message
	名前	年齢	メッセージ
	太郎	0	

4. 項目「メッセージ」の初期値として、「テストメッセージ」と入力してください。

Data OutputObject outputObj
message
メッセージ
テストメッセージ

Excelのシートの状況は以下の通りです。

	Data InputObject inputObj		Data OutputObject outputObj
	Name	Age	message
	名前	年齢	メッセージ
	太郎	0	テストメッセージ

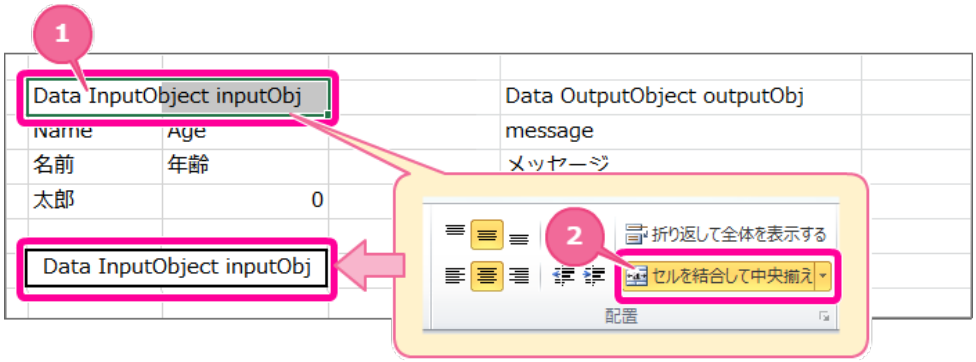
5. これで、項目の初期値（Data）が設定できましたので、一度保存し、次の手順に進みましょう。

項目の初期値（ Data ）のレイアウトを整える

ルールの表のヘッダ・明細に必要な値を入力しましたので、レイアウトを整えます。

1. OpenRules のヘッダの1行目は、サブヘッダ・明細の列の範囲で結合する必要がありますので、2列分を結合します。

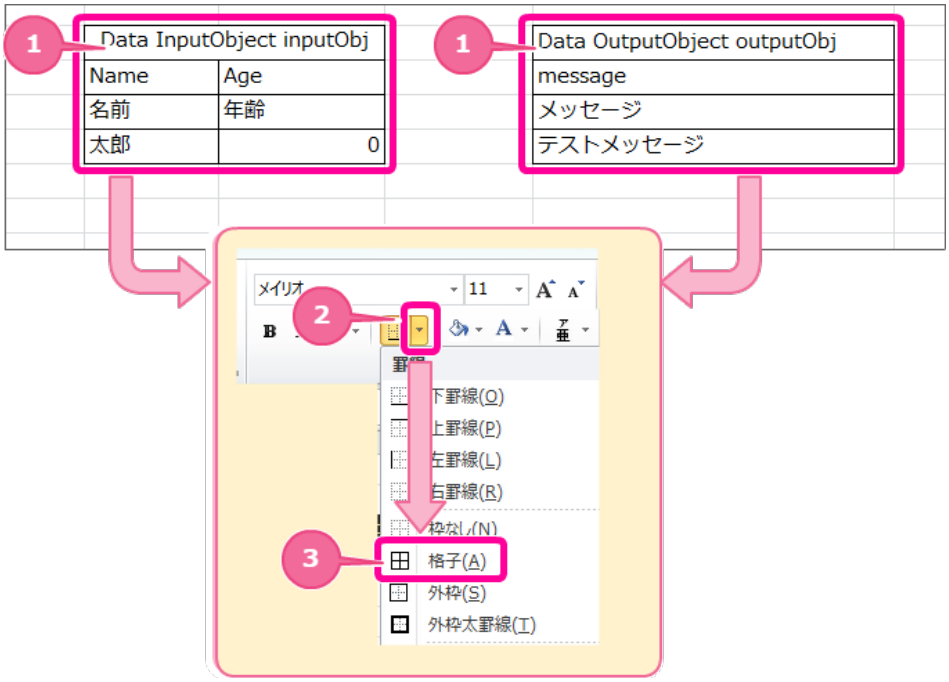
出力項目のオブジェクト（OutputObject）は、1列で構成されていますので、結合する必要はありません。



【ヘッダーのセルを結合する手順】

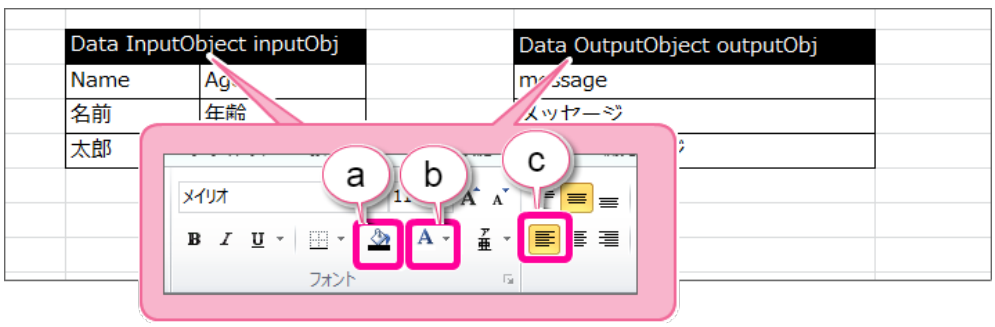
1. ヘッダの行のセル2つ分をドラッグで選択状態にしてください。
2. 「セルを結合して中央揃え」をクリックしてください。
3. これでヘッダーのセルを結合できました。

2. 各セルの境界線をわかりやすくするために罫線を引きます。
罫線の対象のセルを選択し、メニューの罫線から「格子」をクリックしてください。



3. 各セルの書式を OpenRules の標準に合わせて設定します。
1行目のヘッダを以下の通りに設定します。

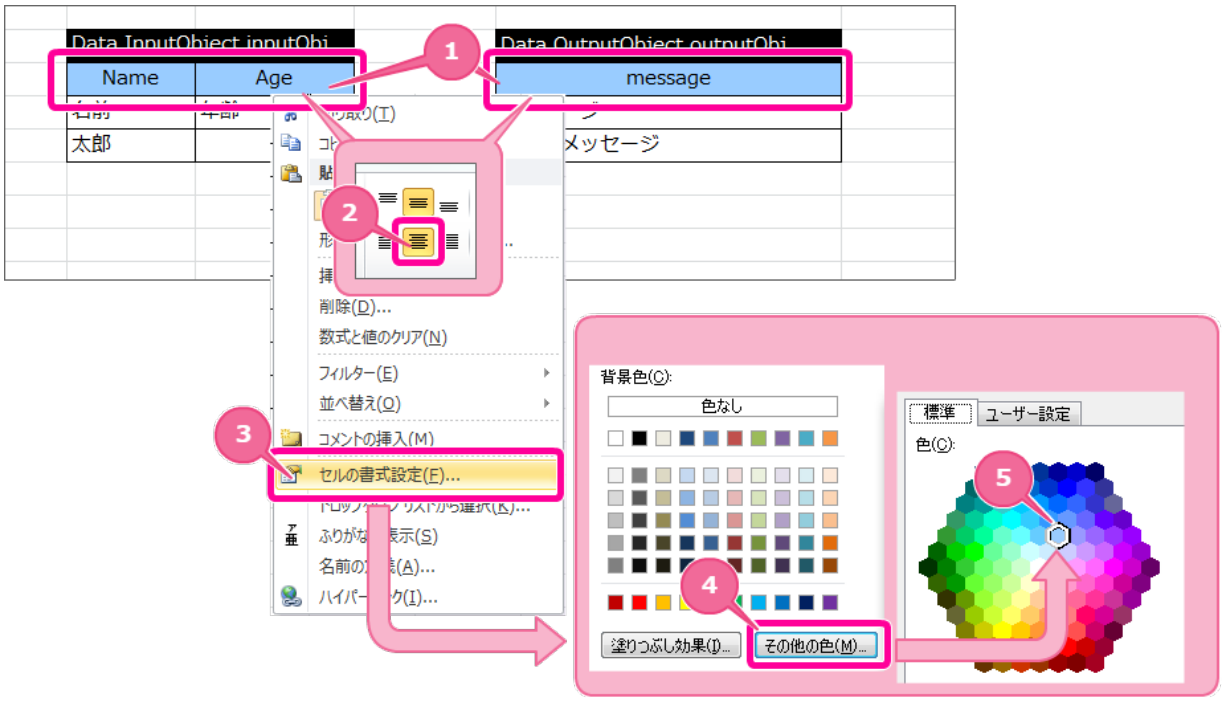
a. セルの背景色（塗りつぶし）：黒
b. セルの文字の色：白
c. セルの揃え方：左揃え



4. サブヘッダの物理名を以下の手順で設定してください。

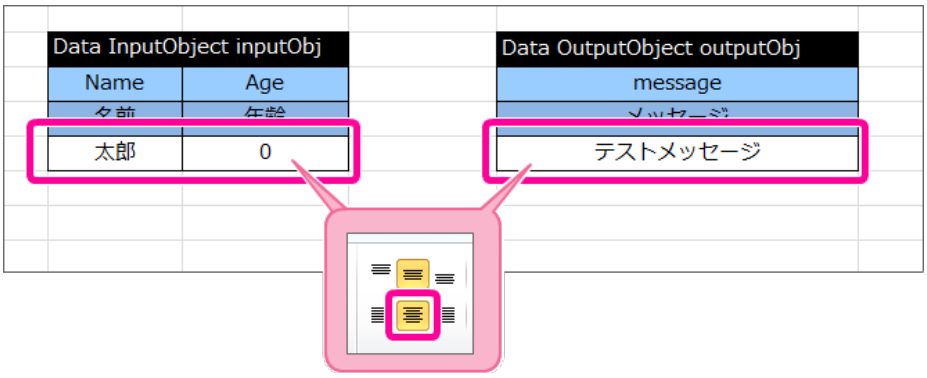
1. InputObject、OutputObjectのそれぞれの物理名のヘッダのセル範囲を選択してください。

2. 配置から横方向の文字揃えの「中央揃え」をクリックしてください。
3. 選択範囲で右クリックし、メニューから「セルの書式設定」をクリックしてください。
4. 「塗りつぶし」タブをクリック後、「その他の色」をクリックしてください。
5. カラーパレットから画像の場所の色をクリックしてください。
   （「ユーザー設定」から設定する場合、「RGB/153,204,255」として設定してください。）



5. 3行目のサブヘッダの論理名も同様に以下の手順で設定してください。
   1. InputObject、OutputObjectのそれぞれの物理名のヘッダのセル範囲を選択してください。
   2. 配置から横方向の文字揃えの「中央揃え」をクリックしてください。
   3. 選択範囲で右クリックし、メニューから「セルの書式設定」をクリックしてください。
   4. 「塗りつぶし」タブをクリック後、「その他の色」をクリックしてください。
   5. 「ユーザー設定」から以下の色を指定してください。
      - カラーモデル： RGB
      - 赤（R）：141
      - 緑（G）：180
      - 青（B）：226

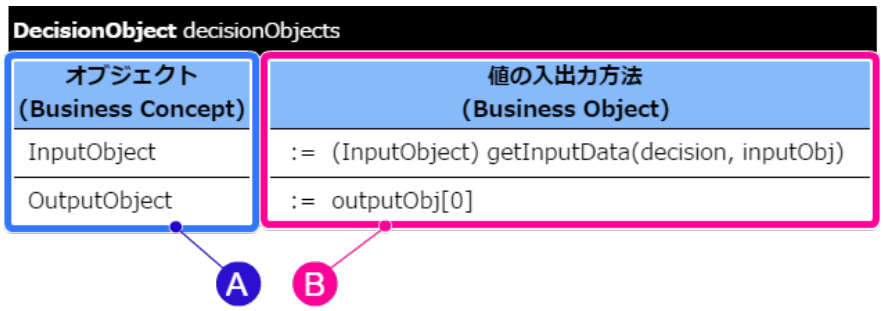
6. 最後に明細を中央揃えに設定してください。



7. 項目の初期値（Data）が完成しましたので、ファイルを保存します。
   次に実行するために必要な表を追加していきます。

Excelファイルにオブジェクトのインスタンスの設定（ DecisionObject ）を作成する

項目の初期値の表の「 *DecisionObject* 」で利用するオブジェクトの値の入出力方法を設定していきましょう。



A. オブジェクト	**B. 値の入出力方法**
Glossary で定義したオブジェクトです。	それぞれのオブジェクトに対して、値の受け渡し方法を記述します。

オブジェクトのインスタンスの設定（ DecisionObject ）のヘッダ部分を作成する

表のヘッダは、 OpenRules のキーワードや項目名をルールに従って設定する必要があります。
最初にヘッダとして必要な内容を設定しましょう。

1. 先の手順で作成したExcelファイルを開きます。
2. オブジェクトのインスタンスの設定の表のキーワードは「 *DecisionObject* 」と記述します。

3. テーブルの名称は、”decisionObjects”と入力します。
   “decisionObjects”以外の名前を設定した場合、実行時にエラーが発生します。

4. ヘッダには以下の形式でキーワードとテーブル名を入力してください。

```
[キーワード]+半角スペース+decisionObjects
```

   Excelファイルに設定する各表（テーブル）は、他のテーブルと1つ以上行・列を空ける必要があります。
   （以下の図では赤色の範囲には何も入力しないでください。）

	A	B	C	D	E	F
22						
23		Data InputObject inputObj			Data OutputObject outputObj	
24		Name	Age		message	
25		名前	年齢		メッセージ	
26		太郎	0		テストメッセージ	
27						
28				DecisionObject decisionObjects		
29						
30						
31						
32						

オブジェクトのインスタンスの設定（ DecisionObject ）のサブヘッダ部分を作成する

DecisionObject は、サブヘッダにオブジェクトとオブジェクトにインスタンス（入出力値）を割り当てる処理を記述します。

1. *DecisionObject* では、 *Glossary* で定義したオブジェクトへのインスタンスの割り当て方法や値の設定方法を設定します。
   (1)に対象のオブジェクト、(2)に割り当てを行うための式を記述しましょう。

DecisionObject decisionObjects	
(1)	(2)

2. 今回のハンズオンでは以下の図の通りに、左を「オブジェクト」、右を「値の入出力方法」とサブヘッダのラベルに設定してください。

DecisionObject decisionObjects	
(1)	(2)

Decision ject decisionObjects	
オブジェクト	値の入出力方法

Excelのシートの状況は以下の通りです。

	DecisionObject decisionObjects	
	オブジェクト	値の入出力方法

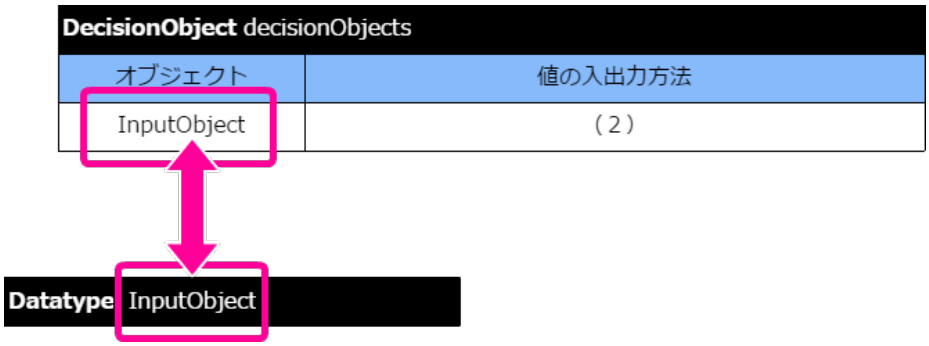
オブジェクトへのインスタンスの割当処理を設定する

内部処理・出力用のオブジェクトの定義の表のヘッダ、サブヘッダができましたので、インスタンスの割当処理を設定していきましょう。

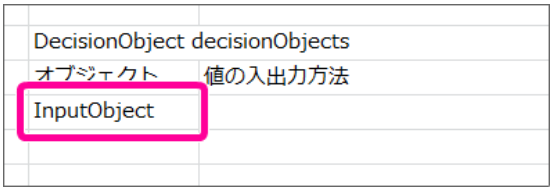
1. 3行目に(1)オブジェクト単位で(2)インスタンスの割当処理を設定します。

DecisionObject decisionObjects	
オブジェクト	値の入出力方法
(1)	(2)

2. 最初にオブジェクト「InputObject」の設定をします。
   左に「InputObject」と記述しましょう。
   この部分の記述内容は、 *Datatype* で定義したオブジェクトと一致させてください。



Excelのシートの状況は以下の通りです。

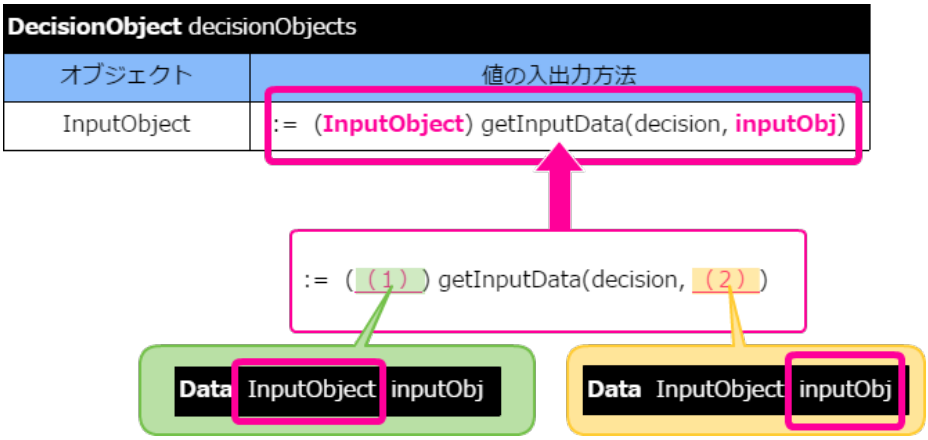


3. 「InputObject」は、 IM-BIS のデータマッパーと連携して入力値を受け取るように設定します。
以下の形式でインスタンスを割り当てる処理を記述してください。

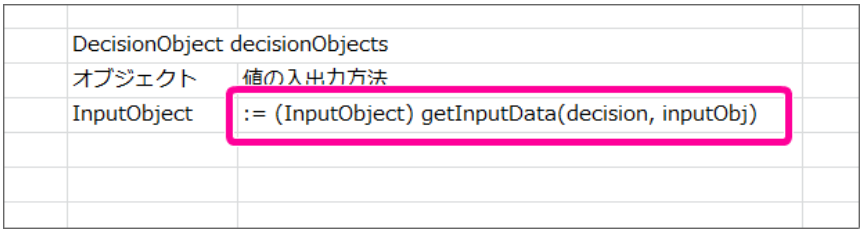
```
:= (オブジェクト名) getInputData(decision, オブジェクトのインスタンス名)
```

4. *Datatype* や *Data/Variable* で設定したオブジェクトやインスタンス名に基づいて、 *DecisionObject* での割り当て方法を以下のように(2)に記述してください。

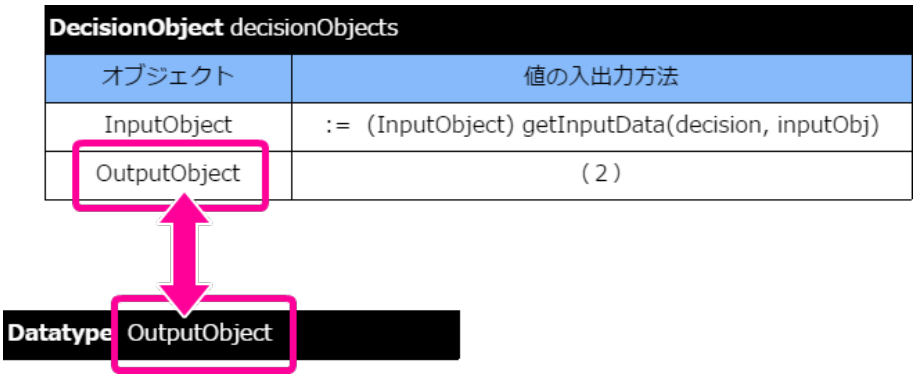
```
:= (InputObject) getInputData(decision, inputObj)
```



Excelのシートの状況は以下の通りです。



5. 次にオブジェクト「OutputObject」の設定をします。
右に「OutputObject」と記述しましょう。
この部分の記述内容は、 *Datatype* で定義したオブジェクトと一致させてください。



Excelのシートの状況は以下の通りです。

	DecisionObject decisionObjects	
	オブジェクト	値の入出力方法
	InputObject	:= (InputObject) getInputData(decision, inputObj)
	OutputObject	

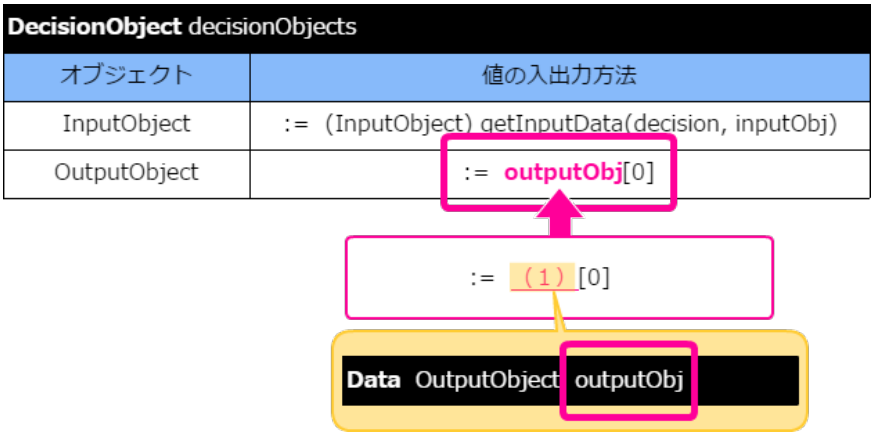
6. 「OutputObject」では、 OpenRules の処理結果を受け取るように設定します。
処理中・処理結果を受け取るためには、以下の式を(2)に記述してください。

```
:= （出力項目のオブジェクトのインスタンス名）[0]
```

DecisionObject decisionObjects	
オブジェクト	値の入出力方法
InputObject	:= (InputObject) getInputData(decision, inputObj)
OutputObject	(2)

7. *Data/Variable* で設定したオブジェクトやインスタンス名に基づいて、 *DecisionObject* には以下のように(2)に記述してください。

```
:= outputObj[0]
```



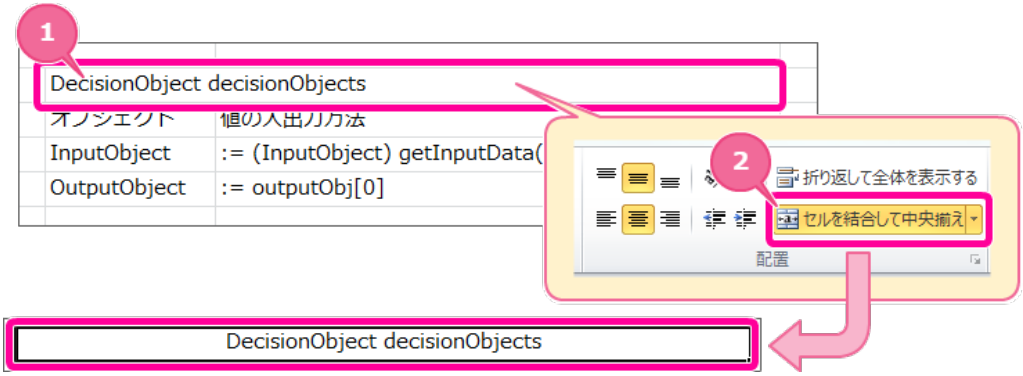
Excelのシートの状況は以下の通りです。

	DecisionObject decisionObjects	
	オブジェクト	値の入出力方法
	InputObject	:= (InputObject) getInputData(decision, inputObj)
	OutputObject	:= outputObj[0]

オブジェクトのインスタンスの設定（ DecisionObject ）のレイアウトを整える

ルールの表のヘッダ・明細に必要な値を入力しましたので、レイアウトを整えます。

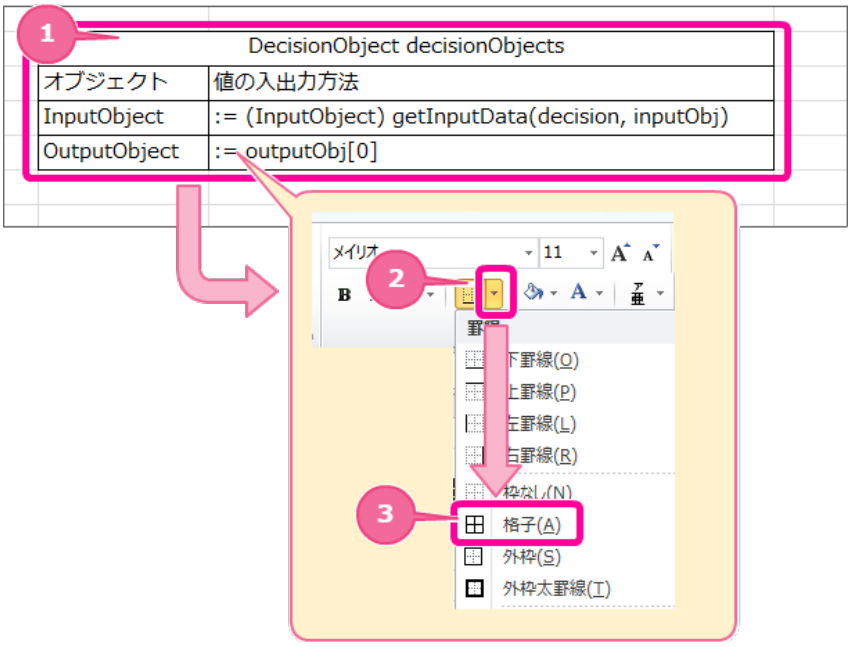
1. OpenRules のヘッダの1行目は、「オブジェクト」「値の入出力方法」の列の範囲で結合する必要がありますので、2列分のセルを結合してください。



【ヘッダーのセルを結合する手順】

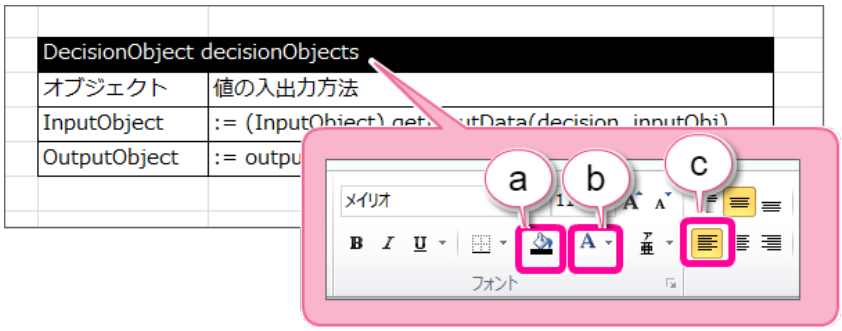
1. ヘッダの行のセル2つ分をドラッグで選択状態にしてください。
2. 「セルを結合して中央揃え」をクリックしてください。
3. これでヘッダーのセルを結合できました。

2. 各セルの境界線をわかりやすくするために罫線を引きます。
罫線の対象のセルを選択し、メニューの罫線から「格子」をクリックしてください。



3. 各セルの書式を OpenRules の標準に合わせて設定します。
1行目のヘッダを以下の通りに設定します。

   a. セルの背景色（塗りつぶし）：黒
   b. セルの文字の色：白
   c. セルの揃え方：左揃え



4. サブヘッダを以下の手順で設定してください。

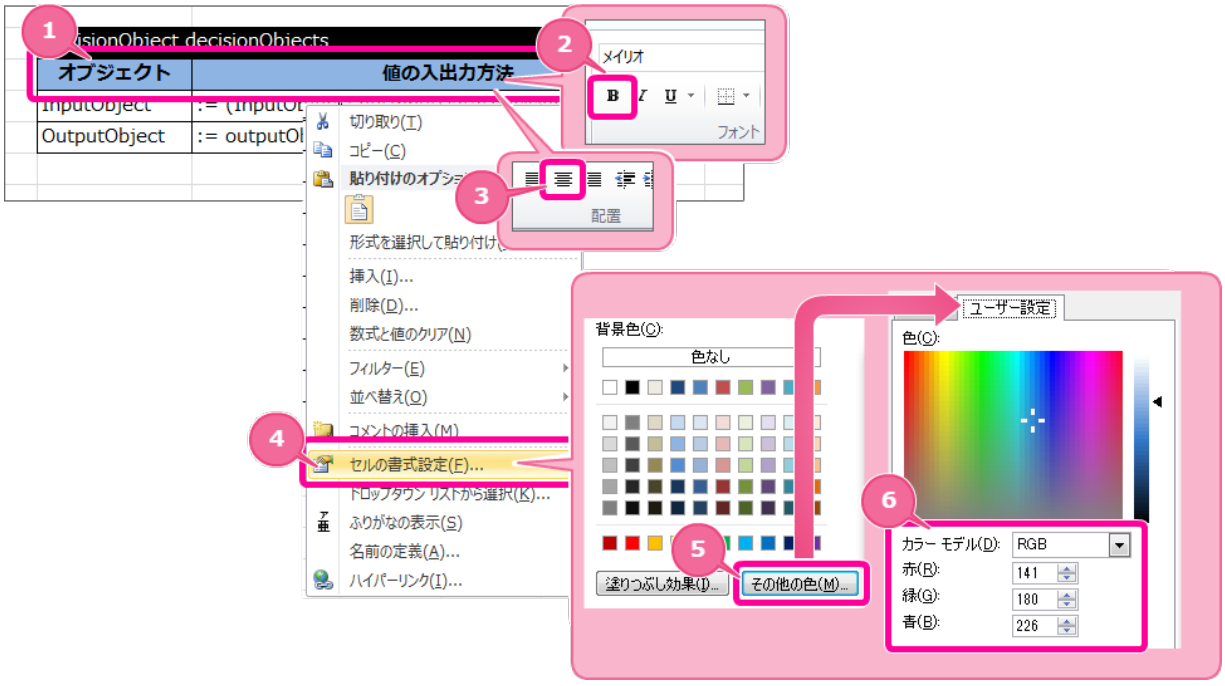
   1. *DecisionObject* のヘッダのセル範囲を選択してください。
   2. フォントから「ボールド」をクリックしてください。

3. 配置から横方向の文字揃えの「中央揃え」をクリックしてください。

4. 選択範囲で右クリックし、メニューから「セルの書式設定」をクリックしてください。

5. 「塗りつぶし」タブをクリック後、「その他の色」をクリックしてください。

6. 「ユーザー設定」から以下の色を指定してください。
   - カラーモデル： RGB
   - 赤（R）：141
   - 緑（G）：180
   - 青（B）：226



5. 内部処理・出力用のオブジェクトの定義（DecisionObject）が完成しましたので、ファイルを保存します。
   次に実行するために必要な表を追加していきます。

入出力処理やルールの実行設定（ Decision ）を作成する

ルールを実行する際に *DecisionTable* の実行順などをコントロールするための表の「 *Decision* 」を設定していきましょう。

Decision myDecision

ActionPrint	ActionExecute
処理名 (Decisions)	**実行する処理 (Execute Decision Tables)**
入力内容の表示	:= System.out.println(getDecisionObject("InputObject"))
評価の実行	executeDecision
結果の取得	:= decision.put("OutputObject", outputObj)
出力内容の表示	:= System.out.println(getDecisionObject("OutputObject"))

A　B

A. 処理名
処理名は、以下の形で構成されます。

1行目：列のタイプを表すキーワード
2行目：列の名前（ラベル）
3行目～：処理名

B. 実行する処理
実行する処理の列は、以下の形で構成されます。

1行目：列のタイプを表すキーワード
2行目：列の名前（ラベル）
3行目～：処理を表す式、DecisionTable名

入出力処理やルールの実行設定（ Decision ）のヘッダ部分を作成する

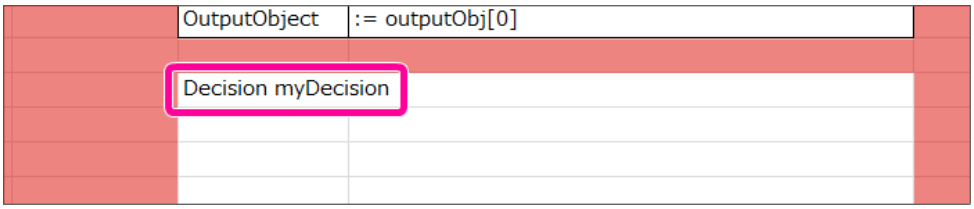
表のヘッダは、 OpenRules のキーワードや項目名をルールに従って設定する必要があります。
最初にヘッダとして必要な内容を設定しましょう。

1. 先の手順で作成したExcelファイルを開きます。
2. 入出力処理やルールの実行設定の表のキーワードは「 *Decision* 」と記述します。

3. ヘッダには以下の形式でキーワードとテーブル名を入力してください。
   テーブル名は半角英数字で構成できていれば任意の名称を設定してください。
   このハンズオンでは「myDecision」と設定して進めていきます。

```
[キーワード]+半角スペース+[テーブル名]
```

   この「 *Decision* 」も、他の表と同様に、同じシートに作成する場合には、1つ以上行・列を空けて作成します。
   以下の図中の赤色部分には何も入力しないでください。

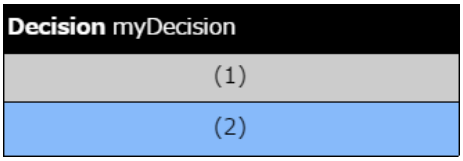


4. これで、ヘッダが設定できましたので、一度保存し、次の手順に進みましょう。

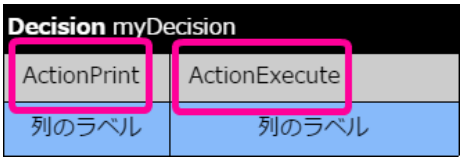
入出力処理やルールの実行設定（ Decision ）のサブヘッダ部分を作成する

Decision は、サブヘッダに各列のタイプとラベルを記述します。

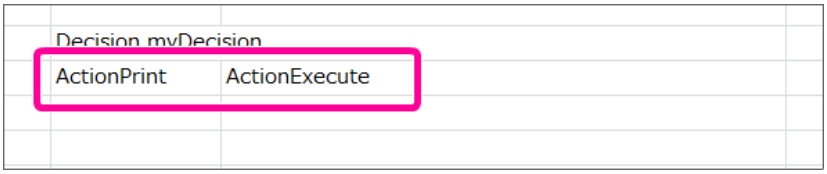
1. *Decision* では、さまざまな列タイプを設定し、列タイプに応じて必要な内容を設定します。
   サブヘッダでは、(1)列タイプを指定し、ユーザにわかりやすくするための(2)列ラベルを設定していきます。



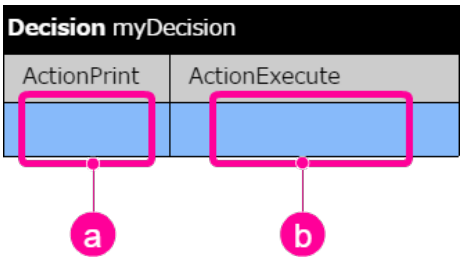
2. *Decision* での必須の列タイプは「 *ActionPrint* 」「 *ActionExecute* 」です。
   今回のハンズオンでは必須の項目のみを設定します。
   （その他に設定できる列タイプは、「 *Decision* 」の「サブヘッダに利用できるキーワード」を参照してください。）
   サブヘッダの1行目に、左から「ActionPrint」「ActionExecute」と記述してください。



   Excelのシートの状況は以下の通りです。



3. *Decision* のサブヘッダの2行目はユーザが列の意味を理解するためのラベルとして利用されます。
   今回は、「 *ActionPrint* 」の(a)ラベルに「処理名」、「 *ActionExecute* 」の(b)ラベルには「実行する処理内容」と設定してください。



Excelのシートの状況は以下の通りです。

4. これで、サブヘッダが設定できましたので、一度保存し、次の手順に進みましょう。

入出力処理やルールの実行設定（ Decision ）の処理内容と順序を設定する

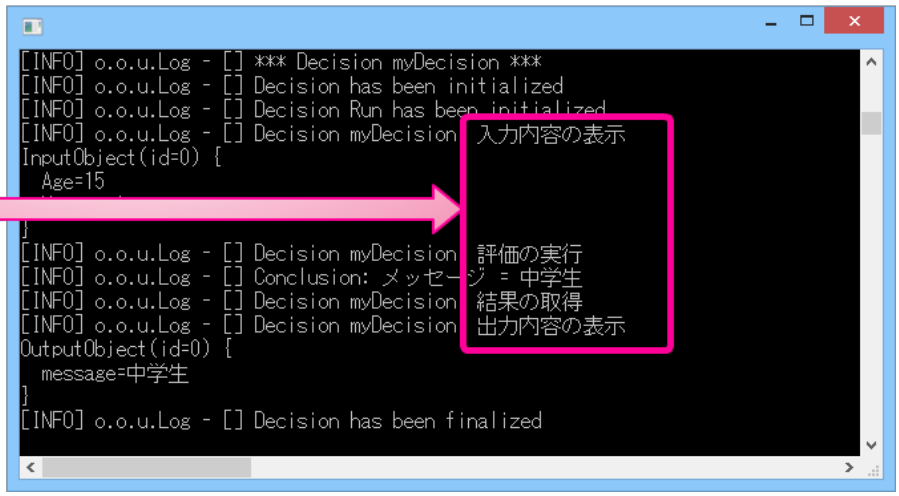
入出力処理やルールの実行設定（ *Decision* ）のヘッダができましたので、評価を行う *DecisionTable* の名前や順序を設定していきましょう。

1. OpenRules の「 *Decision* 」は、上から順に *DecisionTable* を実行するしくみです。
   左側の列に処理を表すためのラベル（名前）、右側の列に実行する *DecisionTable* や *Method* を呼び出すための式を記述します。

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
処理のラベル	実行するDecisionTableやMethod、Javaのコードなど

2. 今回は処理の最初と最後に、値の入出力がどのように行われるかを確認できるように、入力項目のオブジェクトと出力項目のオブジェクトをコンソール上に出力するための処理を記述します。
   先に左側の処理のラベルに、今回実行させる処理の名前を記述してください。
   - 今回の処理内容（番号は処理順を表します。）
     1. 入力項目のオブジェクト（InputObject）の内容のコンソールへの出力
     2. 作成したルールの表（myRule）の実行
     3. 出力項目のオブジェクト（OutputObject）へのルールの実行結果の割当て
     4. 出力項目のオブジェクト（OutputObject）の内容のコンソールへの出力

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
入力内容の表示	実行するDecisionTableやMethod、Javaのコードなど
評価の実行	実行するDecisionTableやMethod、Javaのコードなど
結果の取得	実行するDecisionTableやMethod、Javaのコードなど
出力内容の表示	実行するDecisionTableやMethod、Javaのコードなど



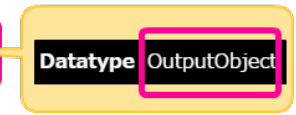
Excelのシートの状況は以下の通りです。

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
入力内容の表示	
評価の実行	
結果の取得	
出力内容の表示	

3. 続いて、実行する処理の式を記述します。
最初に、今回のハンズオンでは、どのように入出力が行われているかを確認してから処理を実行します。
そのために処理の最初に入力項目のオブジェクト、最後に出力用のオブジェクトの内容を出力する式を記述してください。
オブジェクトの内容をコンソールに出力するためのJavaの標準出力の式は以下の通りです。

```
// 入力項目のオブジェクトの出力
:= System.out.println(getDecisionObject("InputObject"))

// 出力項目のオブジェクトの出力
:= System.out.println(getDecisionObject("OutputObject"))
```

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
入力項目の内容の表示	:= System.out.println(getDecisionObject("InputObject"))
評価の実行	実行するDecisionTableやMethod、Javaのコードなど
結果の取得	実行するDecisionTableやMethod、Javaのコードなど
出力項目の内容の表示	:= System.out.println(getDecisionObject("OutputObject"))



> **コラム**
> この標準出力のコードは、 *Decision* での必須項目ではありませんが、 OpenRules へ値の入出力が想定通りに行われているかなどのデバッグ時に活用できます。

Excelのシートの状況は以下の通りです。

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
入力内容の表示	:= System.out.println(getDecisionObject("InputObject"))
評価の実行	
結果の取得	
出力内容の表示	:= System.out.println(getDecisionObject("OutputObject"))

4. 次の行に *DecisionTable* を実行するために、作成した *DecisionTable* の名前を記述しましょう。

```
// DecisionTableの実行
myRule
```

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
入力項目の内容の表示	:= System.out.println(getDecisionObject("InputObject"))
評価の実行	myRule
結果の取得	実行するDecisionTableやMethod、Javaのコードなど
出力項目の内容の表示	:= System.out.println(getDecisionObject("OutputObject"))

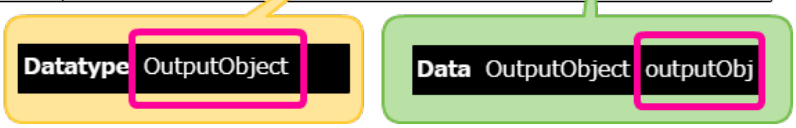
Excelのシートの状況は以下の通りです。

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
入力内容の表示	:= System.out.println(getDecisionObject("InputObject"))
評価の実行	myRule
結果の取得	
出力内容の表示	:= System.out.println(getDecisionObject("OutputObject"))

5. *DecisionTable* の実行（評価）結果を出力項目のオブジェクトのインスタンスに格納する処理を記述しましょう。

```
// 出力項目のオブジェクトのインスタンスへの実行結果のセット
:= decision.put("OutputObject", outputObj)
```

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
入力項目の内容の表示	:= System.out.println(getDecisionObject("InputObject"))
評価の実行	myRule
結果の取得	:= decision.put("OutputObject", outputObj)
出力項目の内容の表示	:= System.out.println(getDecisionObject("OutputObject"))



Excelのシートの状況は以下の通りです。

	Decision myDecision	
	ActionPrint	ActionExecute
	処理名	実行する処理内容
	入力内容の表示	:= System.out.println(getDecisionObject("InputObject"))
	評価の実行	myRule
	結果の取得	:= decision.put("OutputObject", outputObj)
	出力内容の表示	:= System.out.println(getDecisionObject("OutputObject"))

6. 最後にレイアウトを整えて完成させましょう。

入出力処理やルールの実行設定（ Decision ）のレイアウトを整える

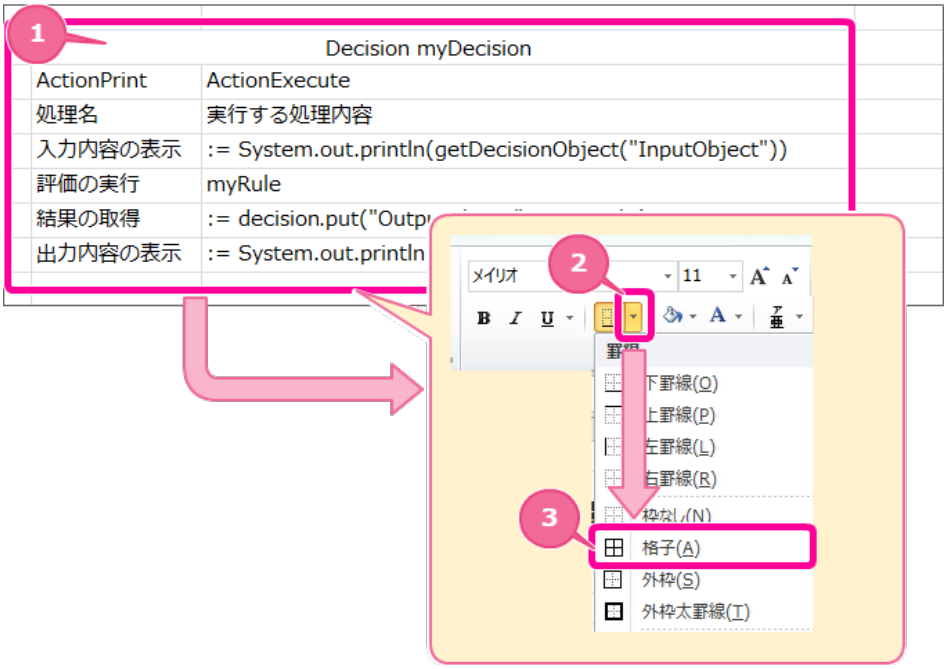
ルールの表のヘッダ・明細に必要な値を入力しましたので、レイアウトを整えます。

1. OpenRules のヘッダの1行目は、サブヘッダ・明細の列の範囲で結合する必要がありますので、2列分を結合します。

【ヘッダーのセルを結合する手順】

1. ヘッダの行のセル2つ分をドラッグで選択状態にしてください。
2. 「セルを結合して中央揃え」をクリックしてください。
3. これでヘッダーのセルを結合できました。

2. 各セルの境界線をわかりやすくするために罫線を引きます。
罫線の対象のセルを選択し、メニューの罫線から「格子」をクリックしてください。

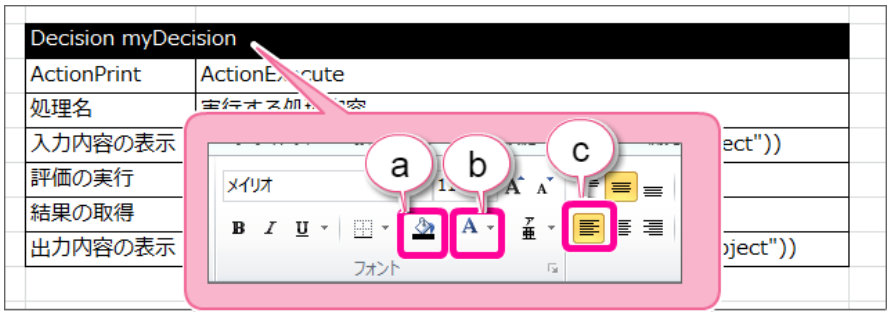


3. 各セルの書式を OpenRules の標準に合わせて設定します。
1行目のヘッダを以下の通りに設定します。

   a. セルの背景色（塗りつぶし）：黒

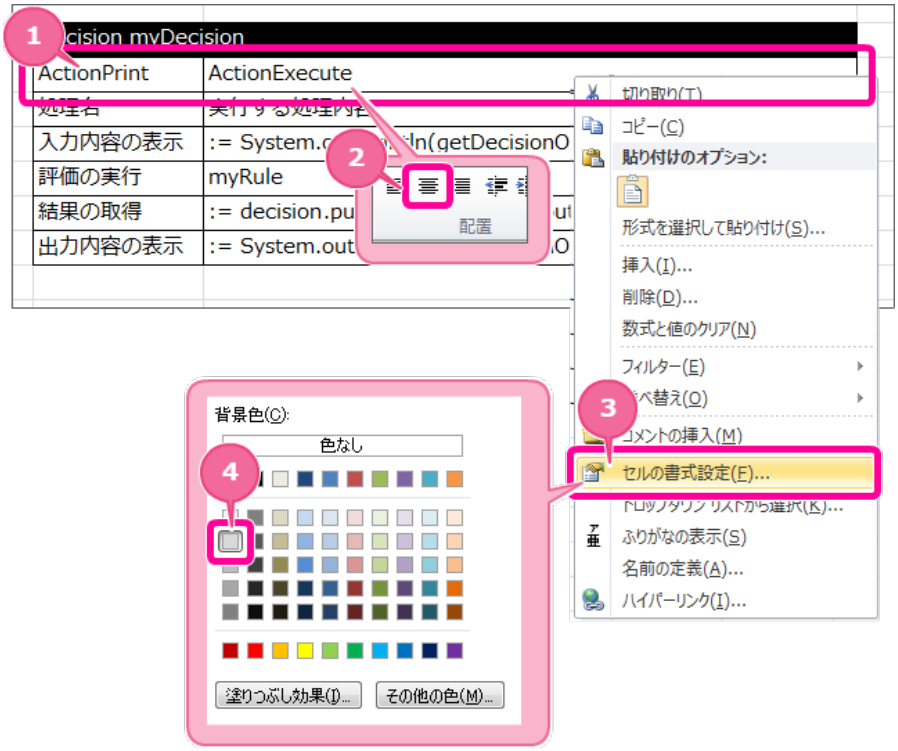
   b. セルの文字の色：白

c. セルの揃え方：左揃え



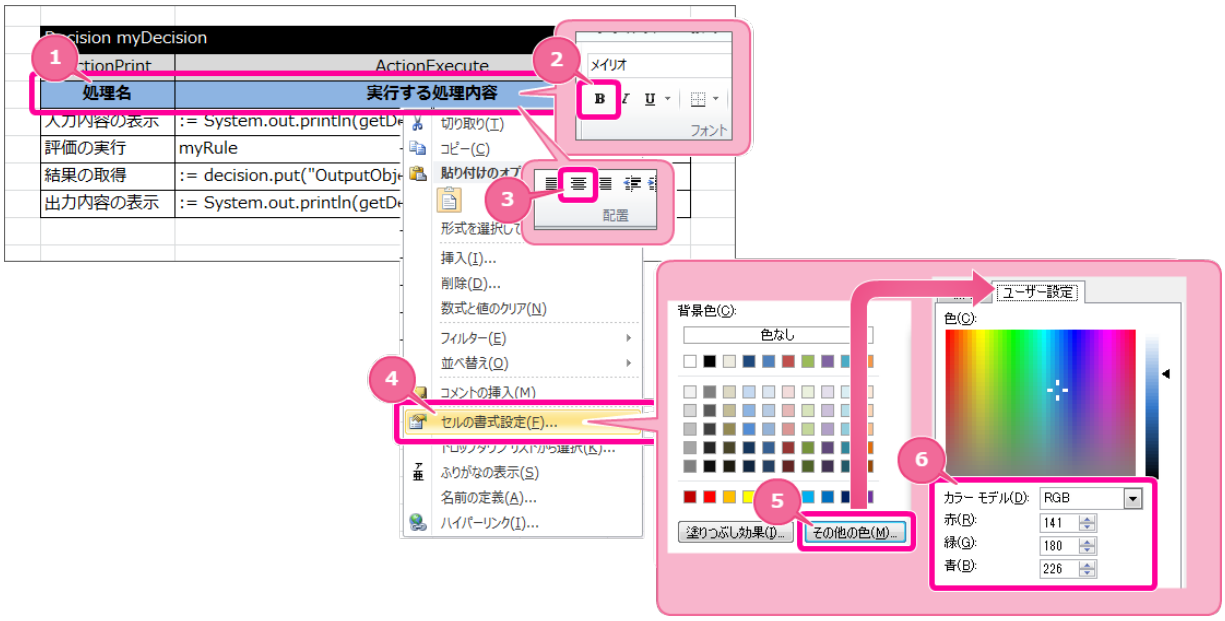
4. 2行目のサブヘッダを以下の手順で設定してください。

   1. *Decision* の2行目のサブヘッダのセル範囲を選択してください。
   2. 配置から横方向の文字揃えの「中央揃え」をクリックしてください。
   3. 選択範囲で右クリックし、メニューから「セルの書式設定」をクリックしてください。
   4. 「塗りつぶし」タブをクリック後、画像の場所の色をクリックしてください。
      （「ユーザー設定」から設定する場合、「RGB/217,217,217」として設定してください。）



5. 3行目のサブヘッダを以下の手順で設定してください。

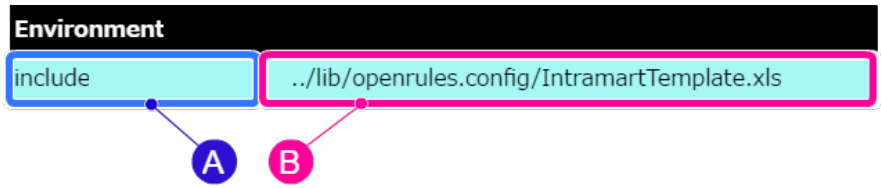
   1. *Decision* の3行目のサブヘッダのセル範囲を選択してください。
   2. フォントから「ボールド」をクリックしてください。
   3. 配置から横方向の文字揃えの「中央揃え」をクリックしてください。
   4. 選択範囲で右クリックし、メニューから「セルの書式設定」をクリックしてください。
   5. 「塗りつぶし」タブをクリック後、「その他の色」をクリックしてください。
   6. 「ユーザー設定」から以下の色を指定してください。
      - カラーモデル： RGB
      - 赤（R）：141
      - 緑（G）：180
      - 青（B）：226



6. 入出力処理やルールの実行設定（ Decision ）が完成しましたので、ファイルを保存します。
   次に環境設定の表を追加していきます。

環境設定（ Environment ）を作成する

ルールを実行する際に必要な環境設定の表の「 *Environment* 」を設定していきましょう。



A. include/import
ルールの実行時に参照するExcelファイルやJavaパッケージなどを読み込むためのキーワードです。

B. ファイル名/パッケージ名
A.のキーワードに合わせたファイル名（ファイルパス）、またはパッケージ名です。

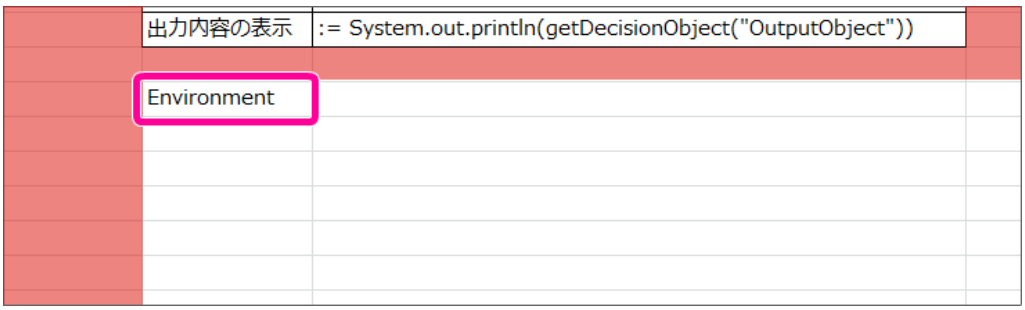
IntramartTemplate.xls
IM-BISと連携するために必要な処理などが含まれるExcelファイルです。BISとの連携時には必須の設定です。

環境設定（ Environment ）のヘッダ部分を作成する

表のヘッダは、 OpenRules のキーワードや項目名をルールに従って設定する必要があります。
最初にヘッダとして必要な内容を設定しましょう。

1. 先の手順で作成したExcelファイルを開きます。
2. マッピング表のキーワードは「 *Environment* 」と記述します。
   テーブル名等をつける必要はありませんので、これでヘッダが完成します。

この「 *Environment* 」も、他の表と同様に、同じシートに作成する場合には、1つ以上行・列を空けて作成します。
以下の図中の赤色部分には何も入力しないでください。



3. これで、ヘッダが設定できましたので、一度保存し、次の手順に進みましょう。

環境設定（ Environment ）に IM-BIS との連携情報のファイルを設定する

ルールの実行時には、 IM-BIS との連携に必要な情報が定義されている「IntramartTemplate.xls」を参照する必要があります。
Environment で「IntramartTemplate.xls」を参照する設定を追加しましょう。

1. 先の手順で作成したExcelファイルを開きます。
2. (1)には、参照するリソースのタイプに合わせたキーワードを指定します。
   (2)には、参照するリソースのファイルパル・ファイル名やパッケージ名等を指定します。

Environment	
(1)	(2)

3. 今回は、 IM-BIS との連携に必要な設定を行うために、キーワードとして「include」、対象にはExcelファイル「IntramartTemplate.xls」を以下のパスで指定してください。

```
../lib/openrules.config/IntramartTemplate.xls
```

Environment	
include	../lib/openrules.config/IntramartTemplate.xls

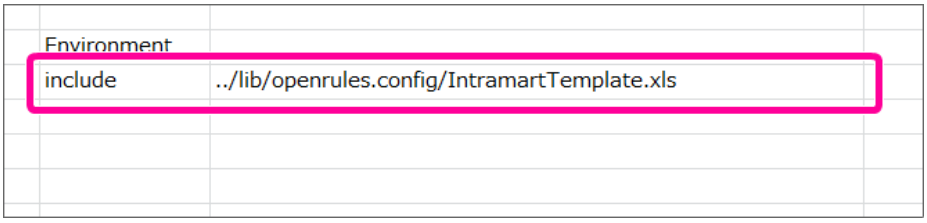
> コラム
>
> この設定で参照しているExcelファイルは、以下の場所に配置されています。
>
> ```
> %CONTEXT_PATH%/WEB-INF/im_bis/datasource/rule/lib/openrules.config
> ```

4. これで、必要な情報を設定できましたので、一度保存し、次の手順に進みましょう。

環境設定（ Environment ）のレイアウトを整える

ルールの表のヘッダ・明細に必要な値を入力しましたので、レイアウトを整えます。

1. OpenRules のヘッダの1行目は、サブヘッダ・明細の列の範囲で結合する必要がありますので、2列分を結合します。



【ヘッダーのセルを結合する手順】

1. ヘッダの行のセル2つ分をドラッグで選択状態にしてください。
2. 「セルを結合して中央揃え」をクリックしてください。
3. これでヘッダーのセルを結合できました。

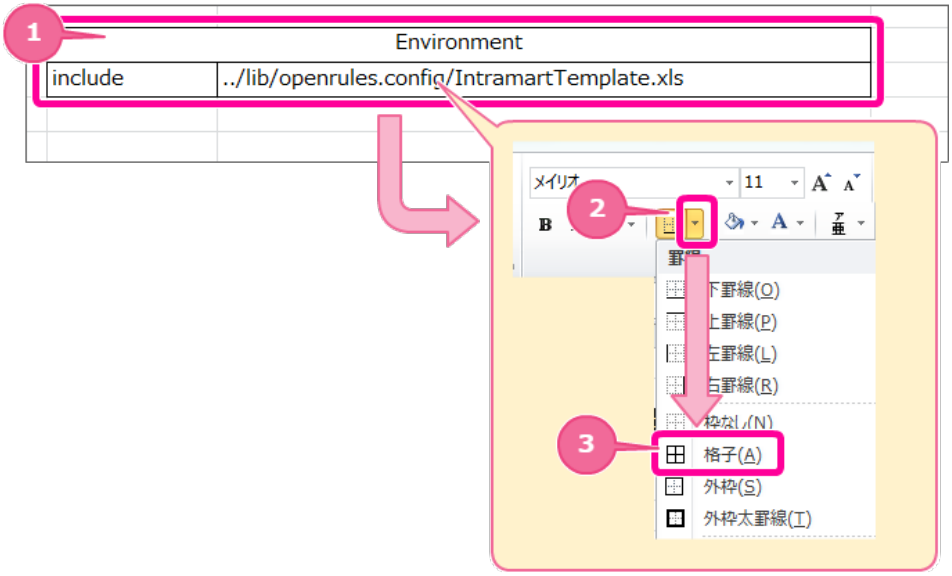
2. 各セルの境界線をわかりやすくするために罫線を引きます。
   罫線の対象のセルを選択し、メニューの罫線から「格子」をクリックしてください。

3. 各セルの書式を OpenRules の標準に合わせて設定します。
   1行目のヘッダを以下の通りに設定します。

   a. セルの背景色（塗りつぶし）：黒

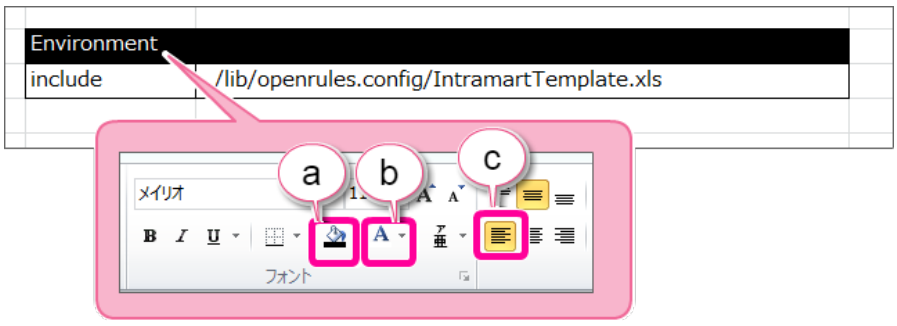
   b. セルの文字の色：白

   c. セルの揃え方：左揃え



4. 残りのセルを以下の手順で設定してください。

   1. *Environment* のヘッダ以外のセル範囲を選択してください。
   2. 選択範囲で右クリックし、メニューから「セルの書式設定」をクリックしてください。
   3. 「塗りつぶし」タブをクリック後、「その他の色」をクリックしてください。
   4. 「標準」タブで画像の場所の色をクリックしてください。
      （「ユーザー設定」から設定する場合、「RGB/204,255,255」として設定してください。）
      - カラーモデル： RGB
      - 赤（R）：204
      - 緑（G）：255
      - 青（B）：255



5. これで、 OpenRules のExcelの定義ファイルが作成できました。
   次の手順で、データソース定義に登録し、 IM-BIS との連携を設定していきましょう。

IM-BIS と連携したフローを作成する

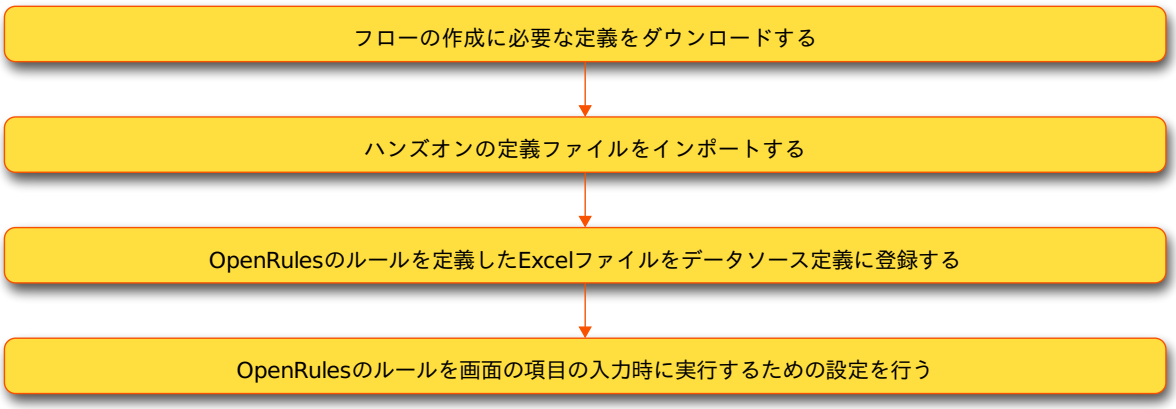
先の手順で作成したExcelのルール定義ファイルを IM-BIS と連携するためのフローの作成を進めていきましょう。

ルールと連携したフローを作成する手順
- OpenRules と IM-BIS を連携するための手順
- フローの作成に必要な定義をダウンロードする
- ハンズオンの定義ファイルをインポートする
- OpenRules のルールを定義したExcelファイルをデータソース定義に登録する
- IM-BIS のイベントにルールを実行するための設定を行う

OpenRules と IM-BIS を連携するための手順

この手順では、作成したExcelのルール定義ファイルをデータソース定義に登録し、 IM-BIS の画面アイテムのイベントに設定するまでの手順を確認していきます。

- Hello! OpenRules を IM-BIS と連携する手順



フローの作成に必要な定義をダウンロードする

ハンズオンで作成するフローのベースとなる各種定義ファイルをインポートします。
最初に下記のリンクからファイルをダウンロードしてください。
「 IM-Workflow 定義」のみダウンロード後に解凍してください。

- IM-Workflow 定義
  imw_hello_openrules.zip
- BIS定義
  bis_hello_openrules.zip
- Formaアプリケーション定義
  forma_hello_openrules.zip

ハンズオンの定義ファイルをインポートする

先の手順でダウンロードしたファイルを「 *各種定義ファイルのインポートの手順* 」に従ってインポートしてください。

OpenRules のルールを定義したExcelファイルをデータソース定義に登録する

必要な準備が整いましたので、 IM-BIS と連携するために、 OpenRules のルールを定義したExcelファイルをデータソース定義に登録していきましょう。

データソース定義の基本情報を登録する

データソース定義の基本情報を登録しましょう。

1. サイトマップの「 IM-BIS 」-「データソース定義」をクリックしてください。

2. 「登録」をクリックしてください。

3. 以下の通りに入力後、「登録」をクリックしてください。

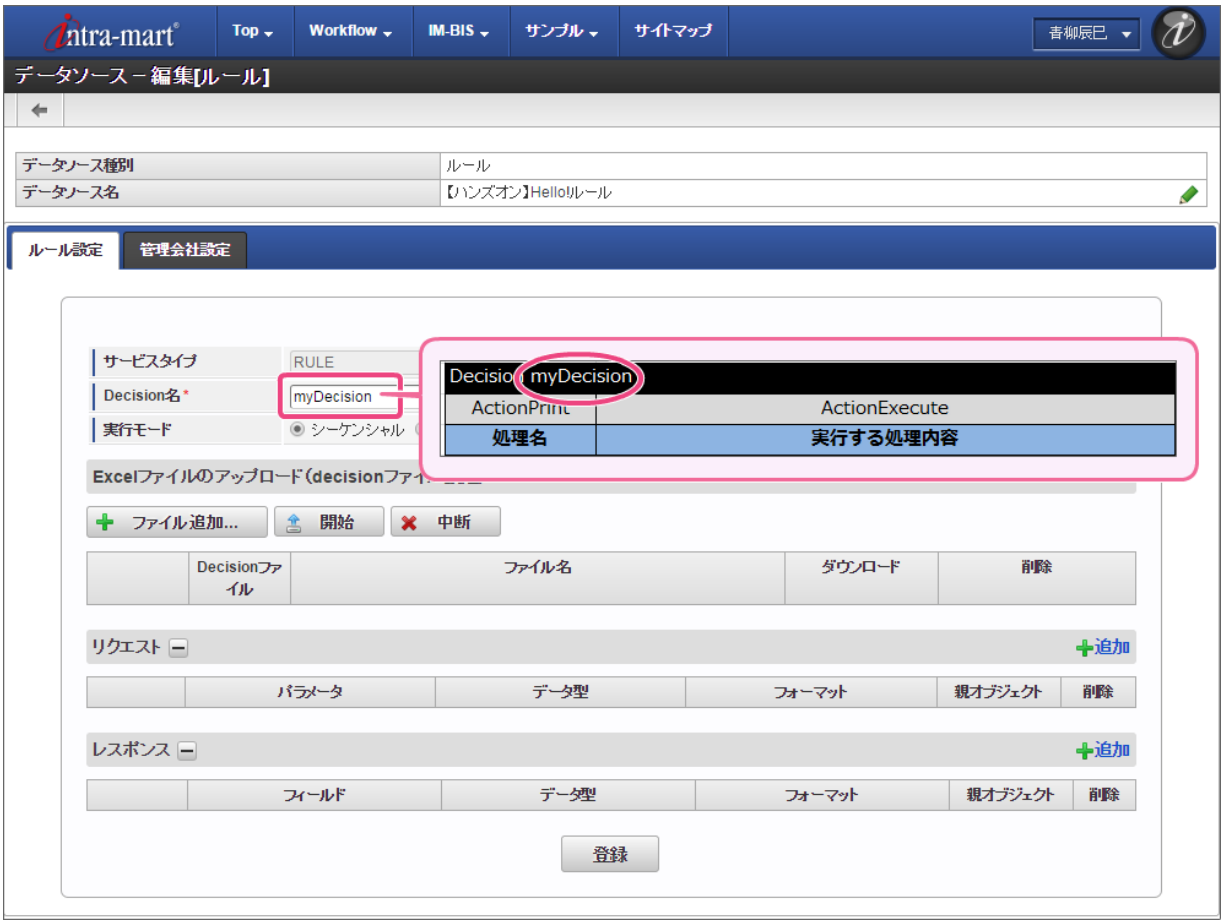
1. データソース種別
   「ルール」を選択してください。

2. データソース名
   「【ハンズオン】Hello! ルール」と入力してください。

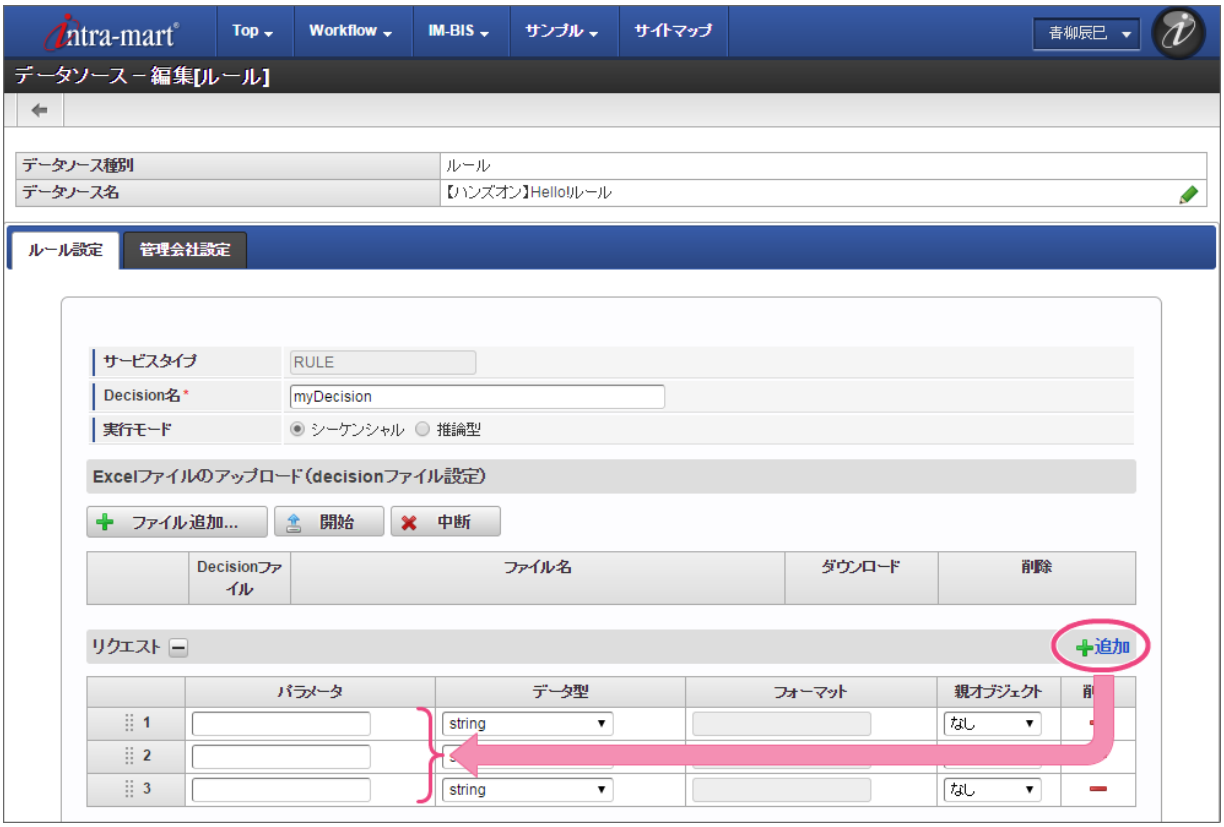
OpenRules の詳細情報を登録する

データソース定義に OpenRules のファイルやパラメータを設定しましょう。

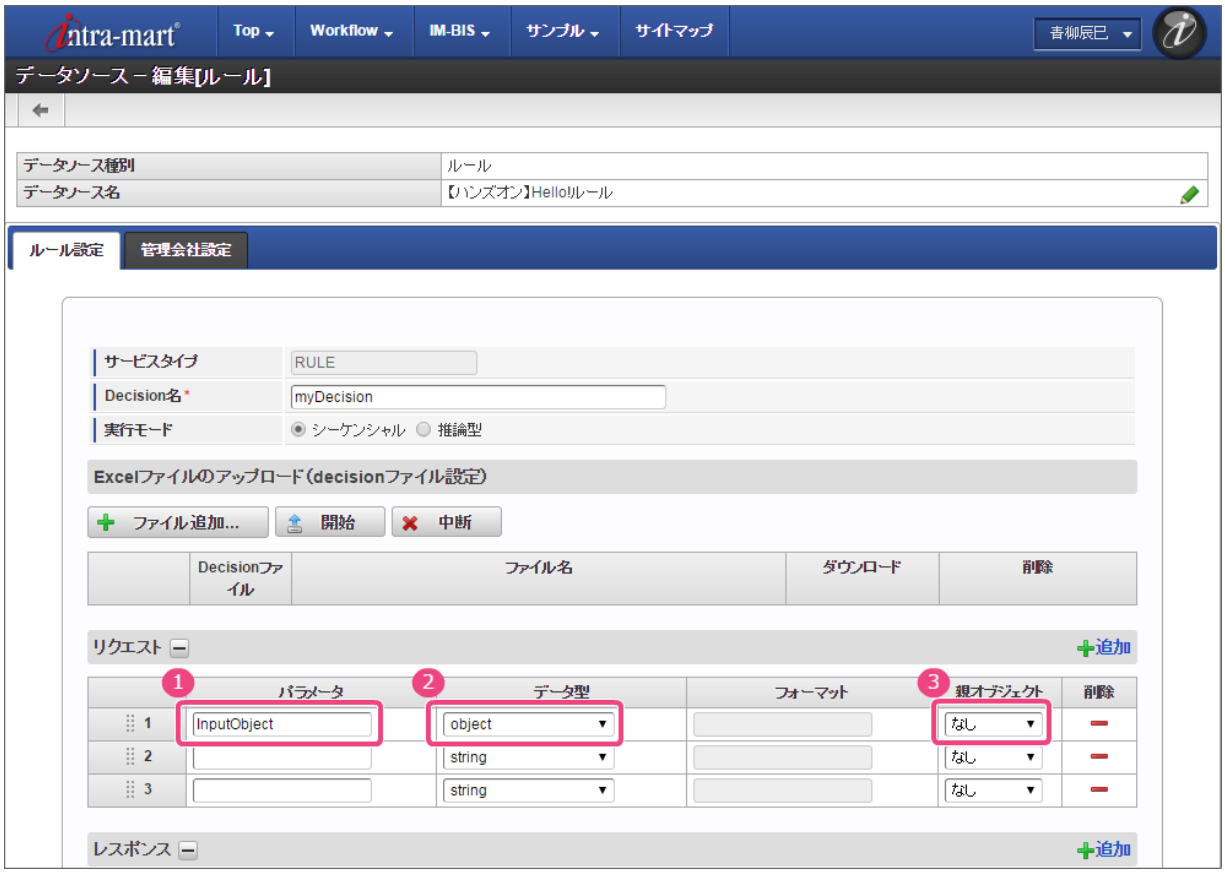
1. 先の手順「 *入出力処理やルールの実行設定（ Decision ）を作成する* 」で定義した *Decision* 名を「Decision名」に入力してください。



2. 「リクエスト」には、「*Excelファイルに項目名のマッピング表（Glossary）を作成する*」の *Glossary* で定義した「InputObject」のオブジェクトと項目（物理名）を登録します。
   オブジェクトと項目の入力欄を追加するために、「追加」を3回クリックしてください。
   クリック後には、3行分の入力欄が表示されます。



3. 1行目は、以下の通りに入力してください。



1. パラメータ

   InputObject

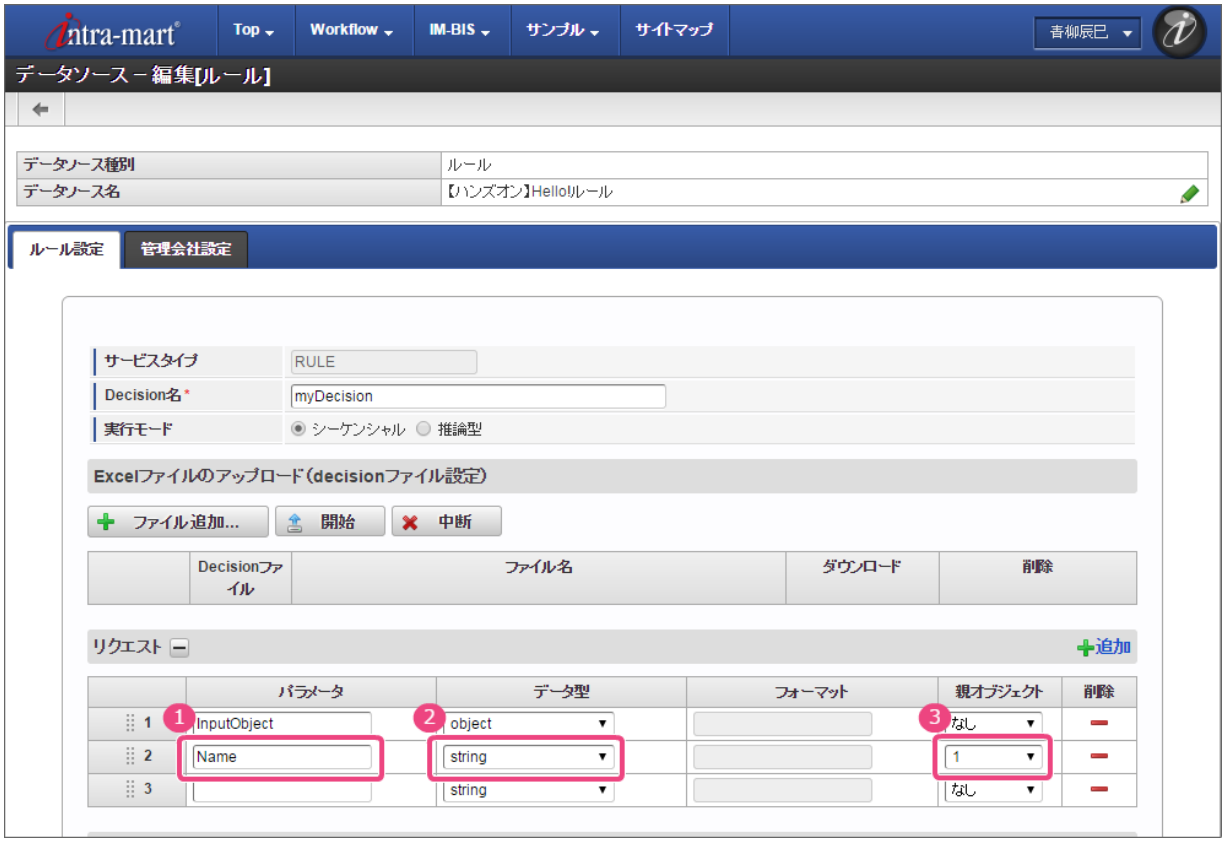
2. データ型

   object

3. 親オブジェクト

   なし

4. 2行目は、以下の通りに入力してください。



1. パラメータ

   Name

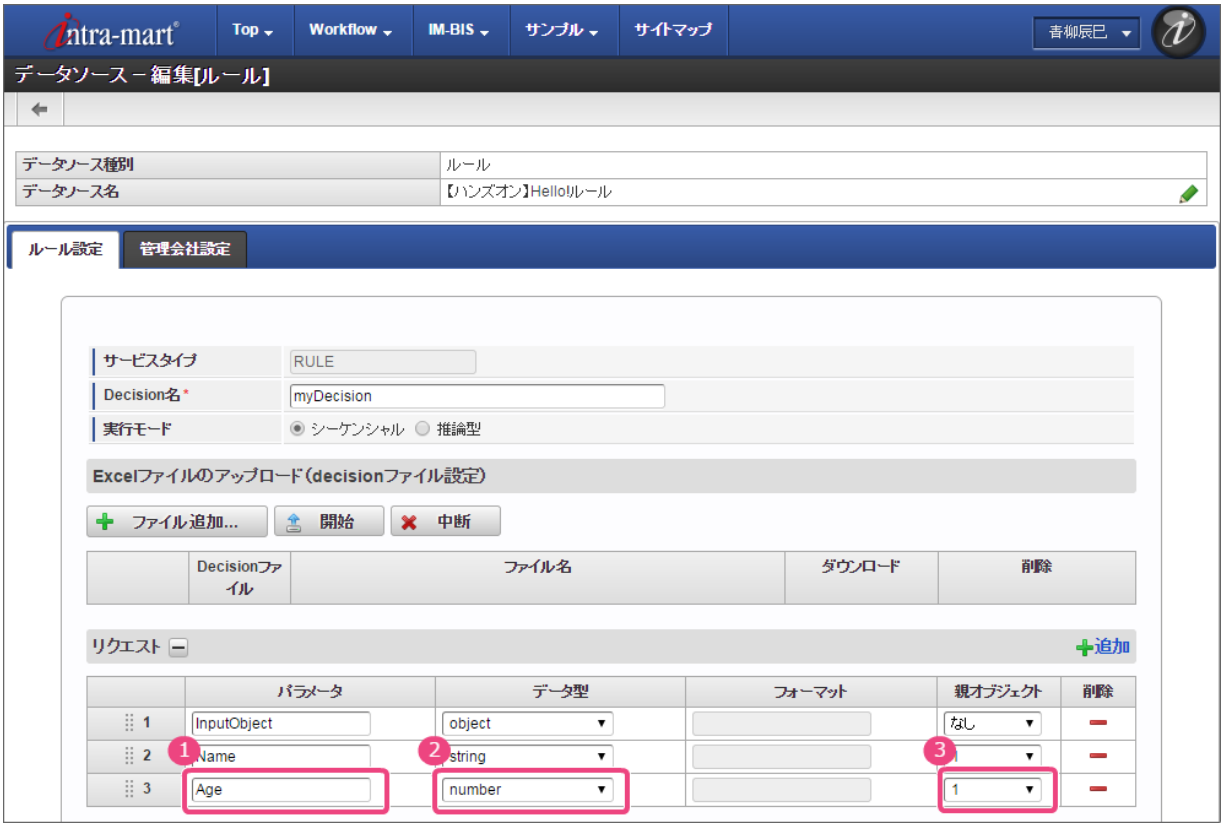
2. データ型

   string

3. 親オブジェクト

   1 *
   * 番号は「リクエスト」の表の行番号を表します。この場合、1行目の「InputObject」を親オブジェクトにする、という設定を表します。

5. 3行目は、以下の通りに入力してください。



1. パラメータ

   Age

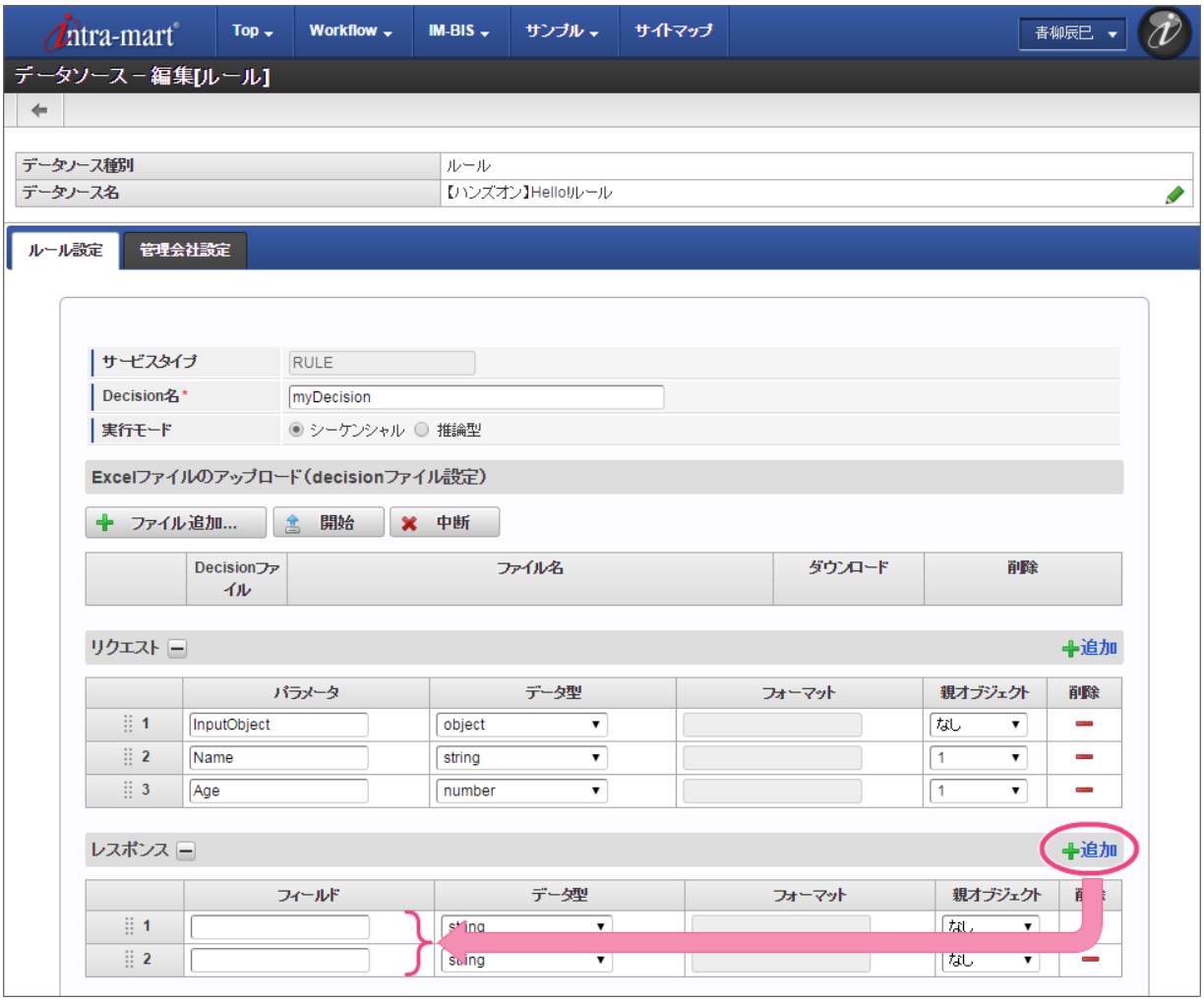
2. データ型

   number

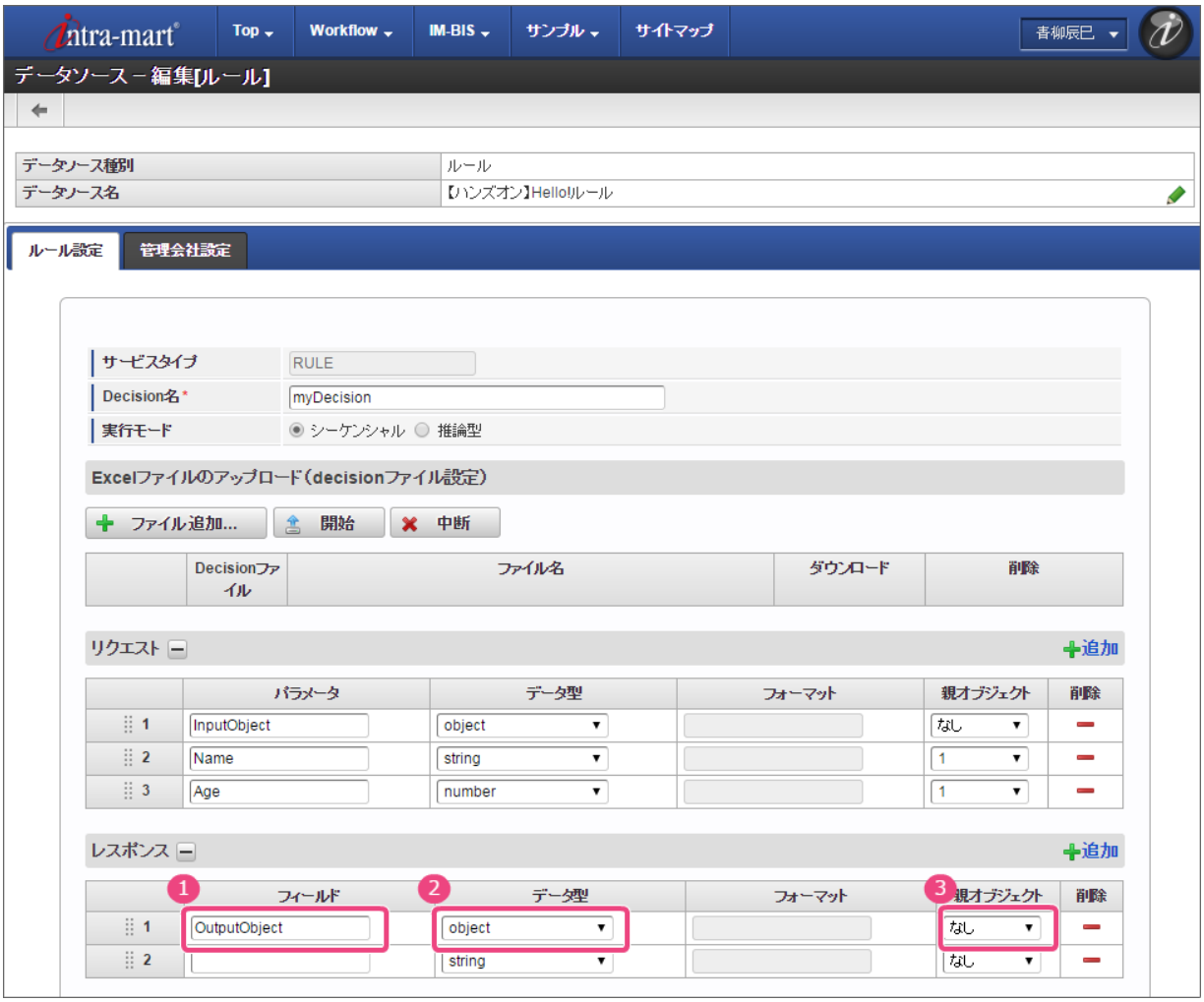
3. 親オブジェクト

   1 *
   * 番号は「リクエスト」の表の行番号を表します。この場合、1行目の「InputObject」を親オブジェクトにする、という設定を表します。

6. 「レスポンス」には、「*Excelファイルに項目名のマッピング表（Glossary）を作成する*」の *Glossary* で定義した「OutputObject」のオブジェクトと項目（物理名）を登録します。
   オブジェクトと項目の入力欄を追加するために、「追加」を2回クリックしてください。
   クリック後には、2行分の入力欄が表示されます。



7. 1行目は、以下の通りに入力してください。



1. フィールド

   OutputObject

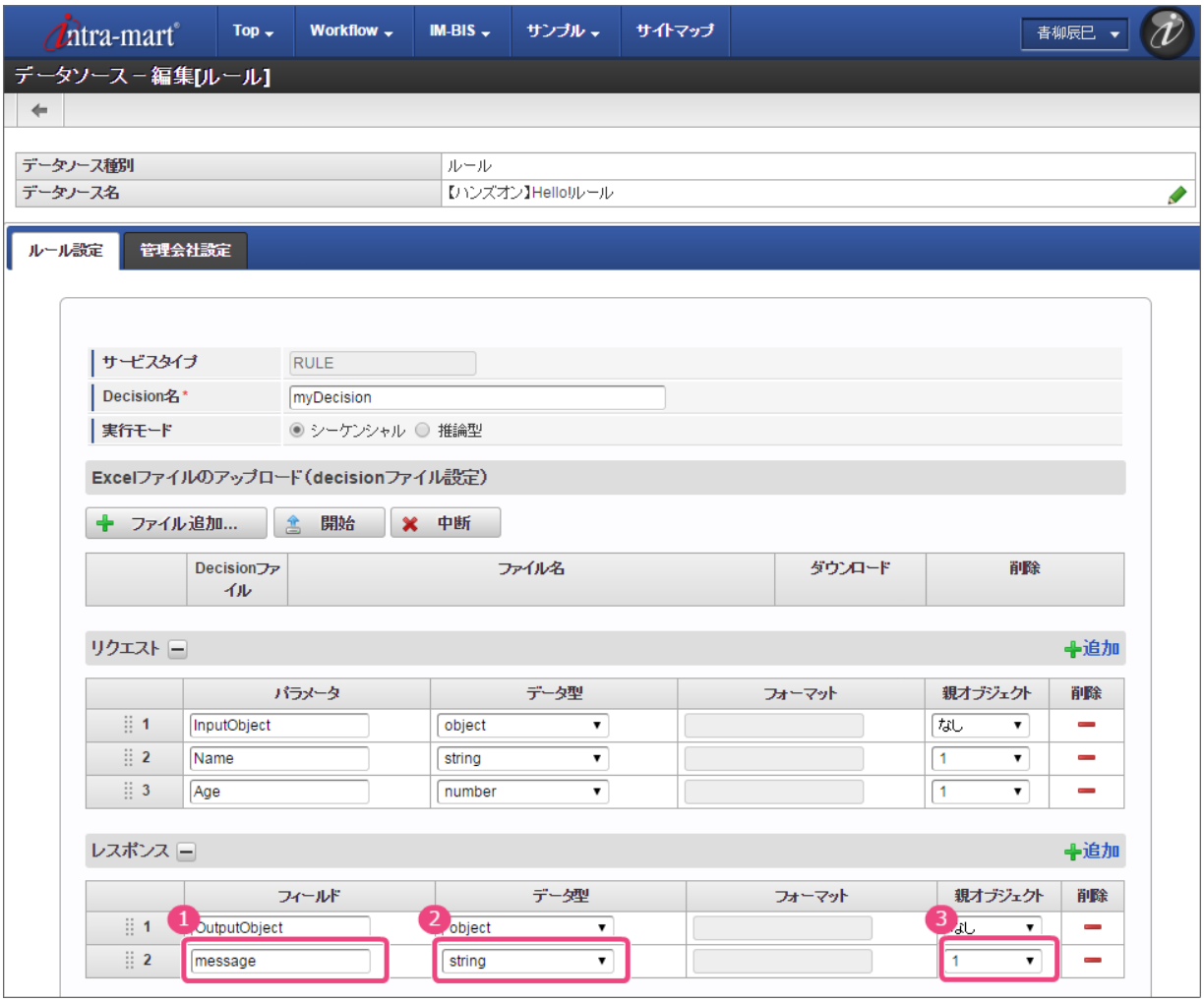
2. データ型

   object

3. 親オブジェクト

   なし

8. 2行目は、以下の通りに入力してください。



1. フィールド

   message

2. データ型

   string

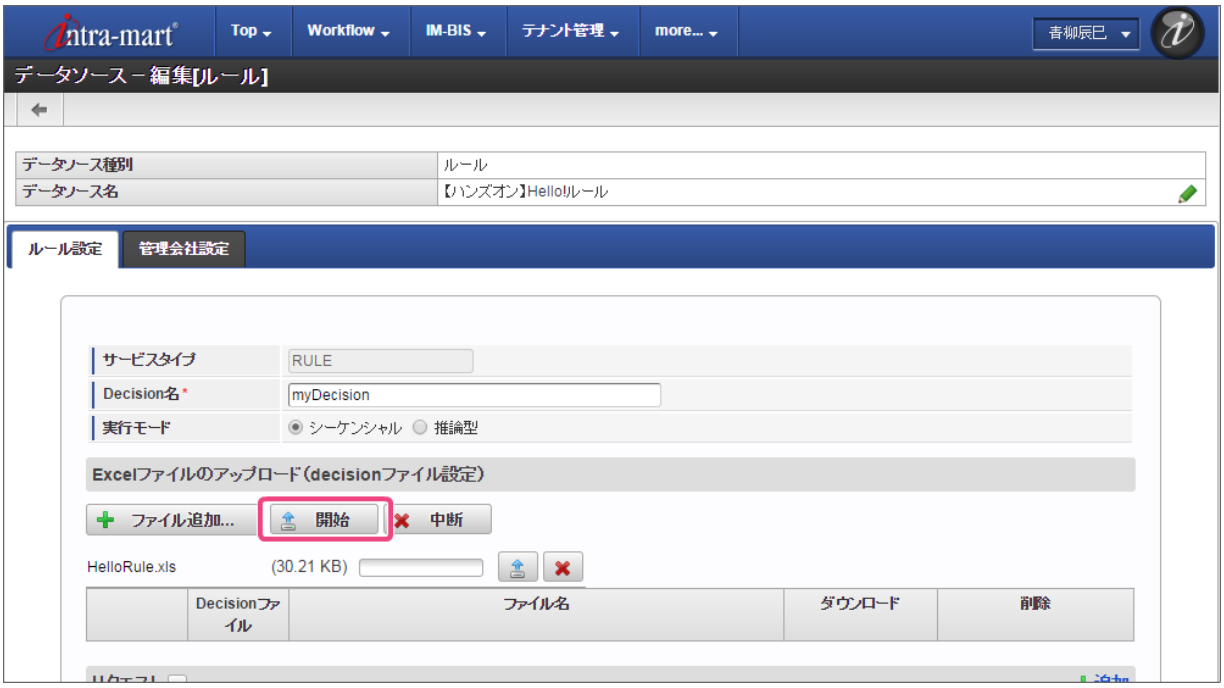
3. 親オブジェクト

   1 *

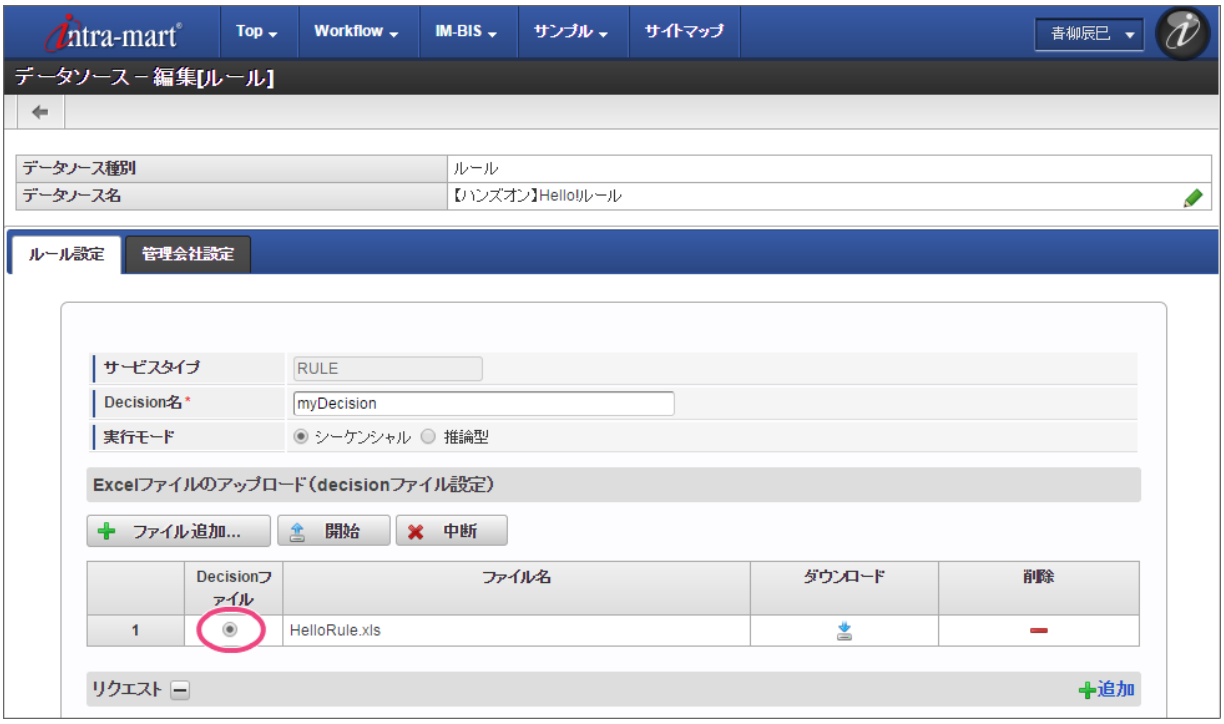
   * 番号は「レスポンス」の表の行番号を表します。この場合、1行目の「OutputObject」を親オブジェクトにする、という設定を表します。

9. 作成したExcelのルール定義ファイルをアップロードするために「ファイル追加」をクリックしてください。

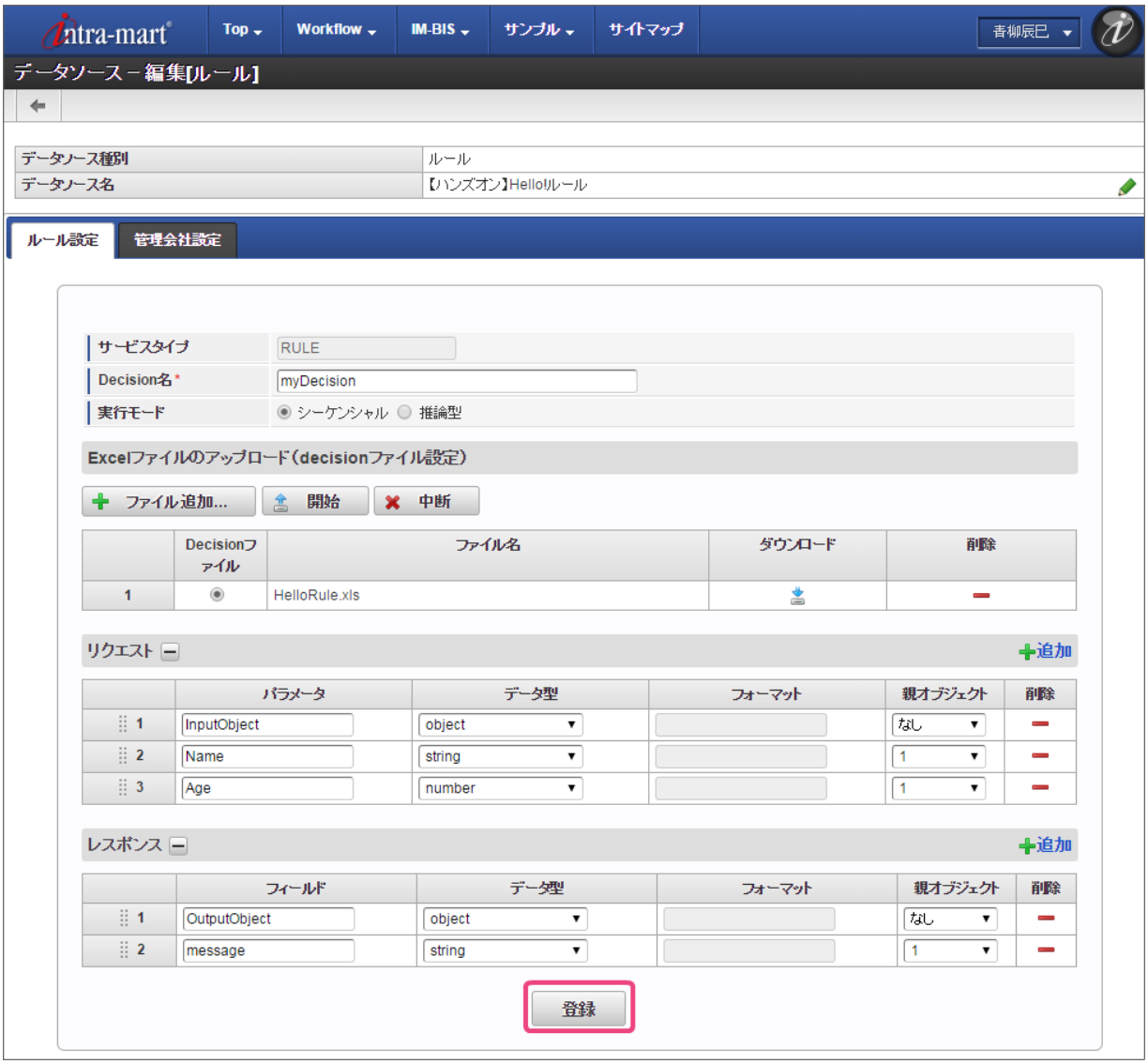
10. 「開始」をクリックして、ファイルをアップロードしてください。



11. 「Decisionファイル」のラジオボタンをクリックしてください。



12. 最後に「登録」をクリックして、データソース定義を登録してください。



13. これで OpenRules のルールを定義したExcelファイルをデータソース定義として登録することができました。

IM-BIS のイベントにルールを実行するための設定を行う

登録したデータソース定義を利用して、 IM-BIS の画面アイテムのイベントにルールの実行を設定しましょう。

フォーム（画面）の編集を開始する

画面アイテムにイベントを設定するために、フォーム（画面）の編集を開始しましょう。

1. サイトマップの「 IM-BIS 」をクリックしてください。

2. 「一覧」をクリックしてください。

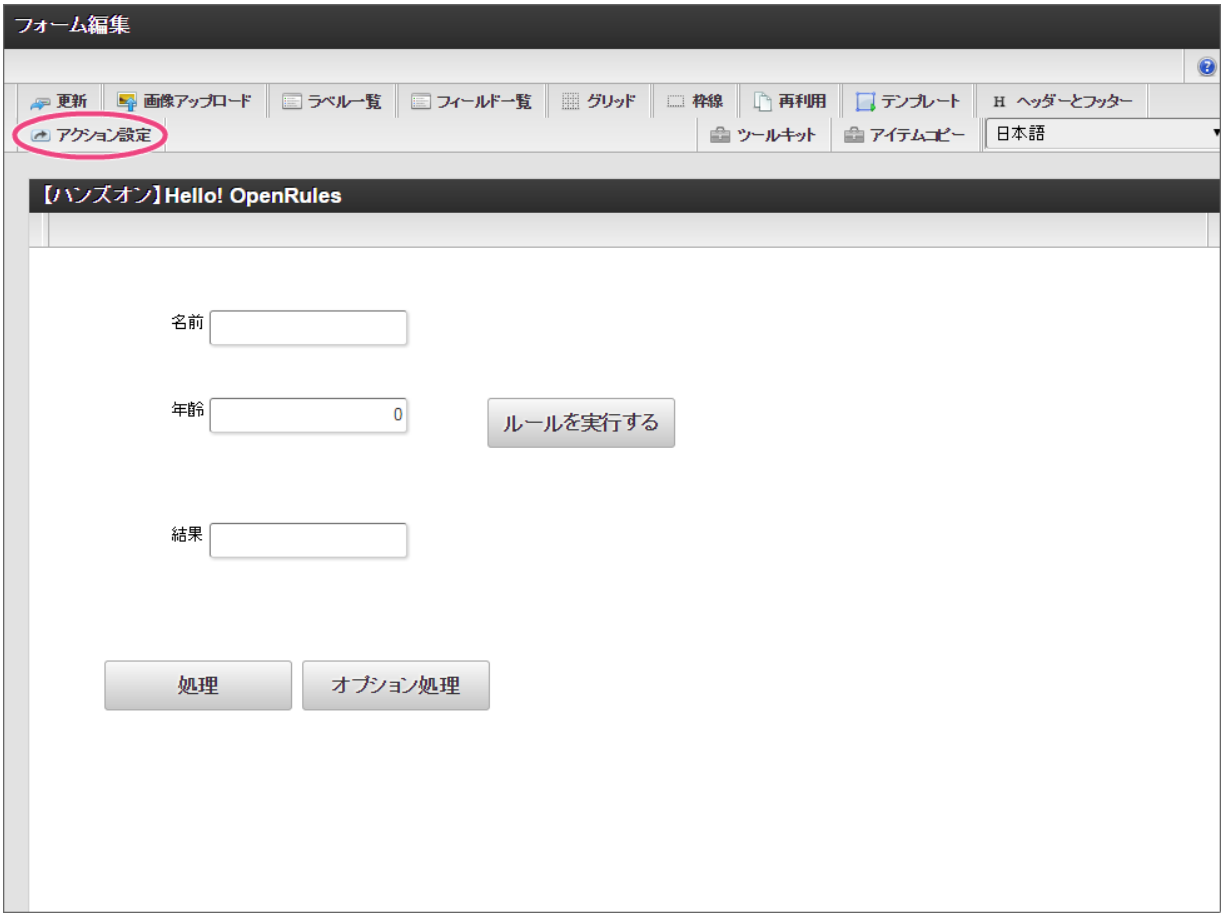
3. インポートしたフローの「【ハンズオン】Hello! OpenRules」の編集をクリックしてください。

4. 「申請／処理開始」のアイコンをダブルクリックして、フォーム編集画面（フォーム・デザイナ）を表示してください。

画面のアクションイベントにルールの実行を設定する

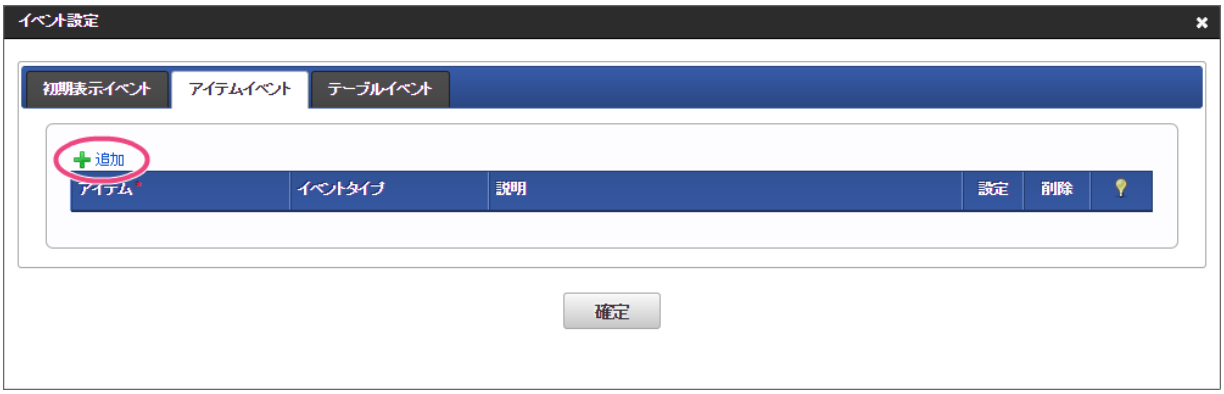
フォーム（画面）の編集画面で、画面アイテムにルールを実行するイベントを設定しましょう。

1. フォーム編集画面を表示したら「アクション設定」をクリックしてください。



2. 「アイテムイベント」をクリックして、表示するタブを切り替えます。

3. 「追加」をクリックしてください。



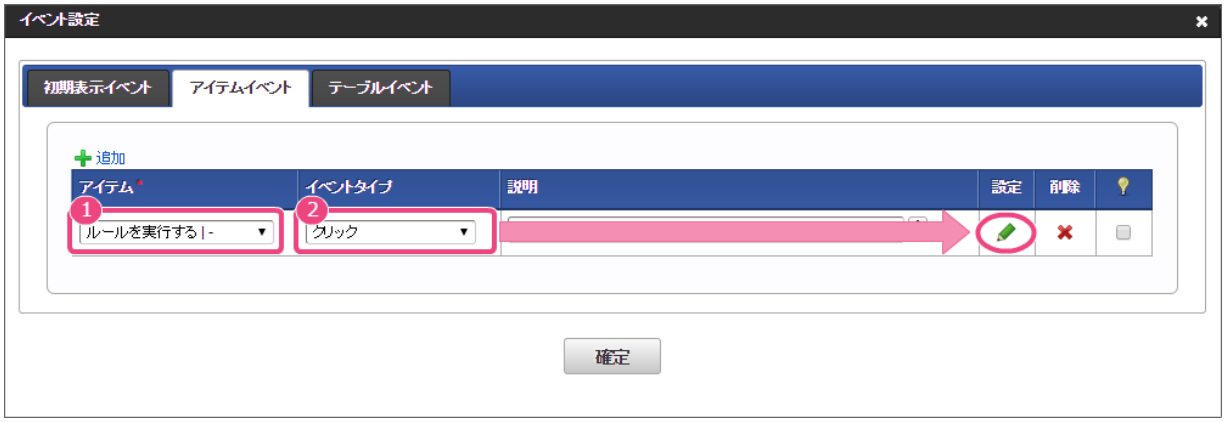
4. アイテムとイベントタイプを以下の通りに変更後、「」をクリックしてください。

    1. アイテム

        ルールを実行する | -（ボタン（イベント））

    2. イベントタイプ

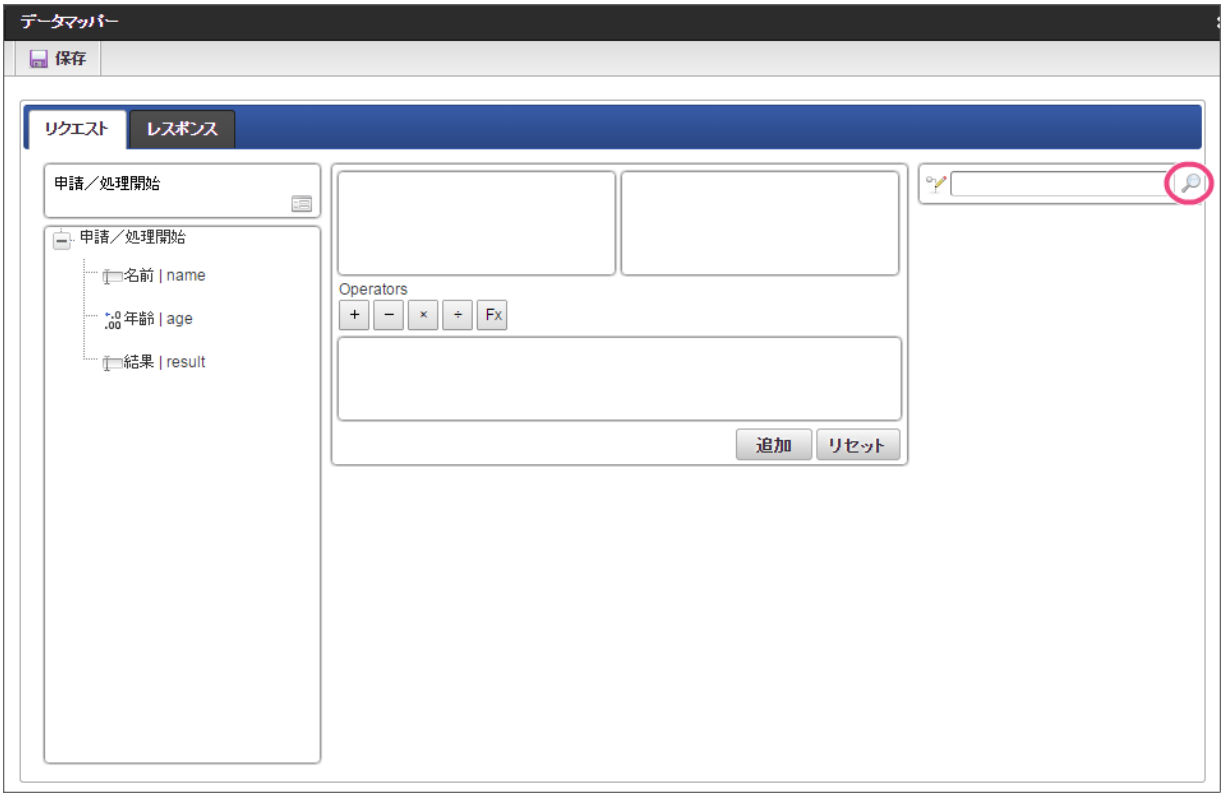
        クリック



5. 「追加」をクリックしてください。

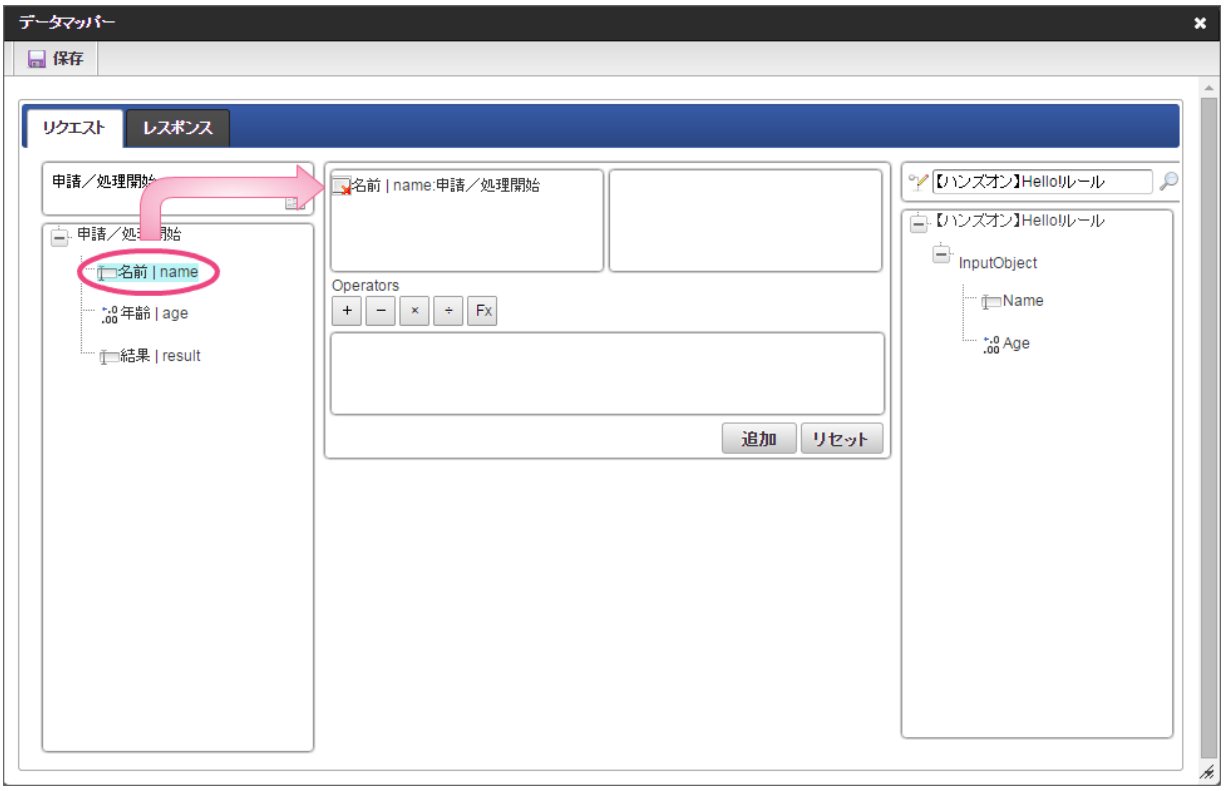
6. 「アクション」を「外部連携」に設定後、「 」をクリックしてください。

7. 「データマッパー」で右上の をクリックしてください。

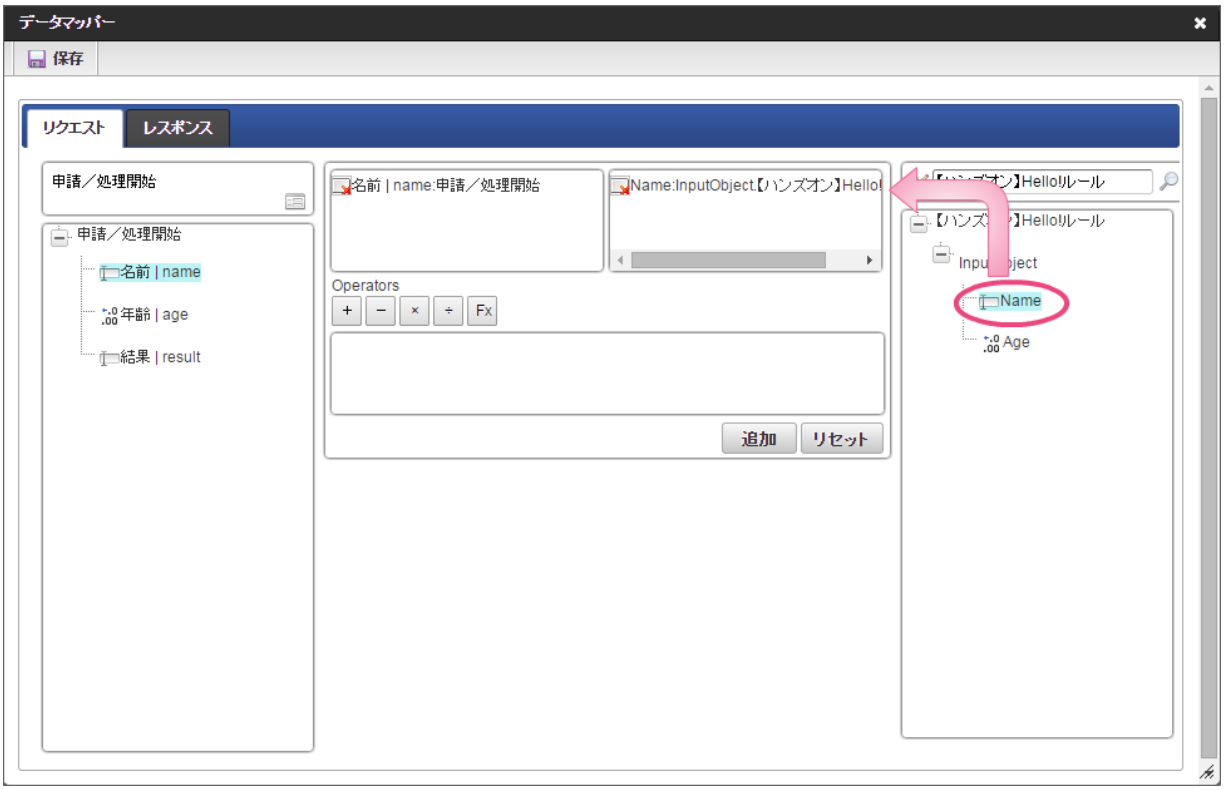


8. 登録したデータソース定義「【ハンズオン】Hello!ルール」をクリックしてください。

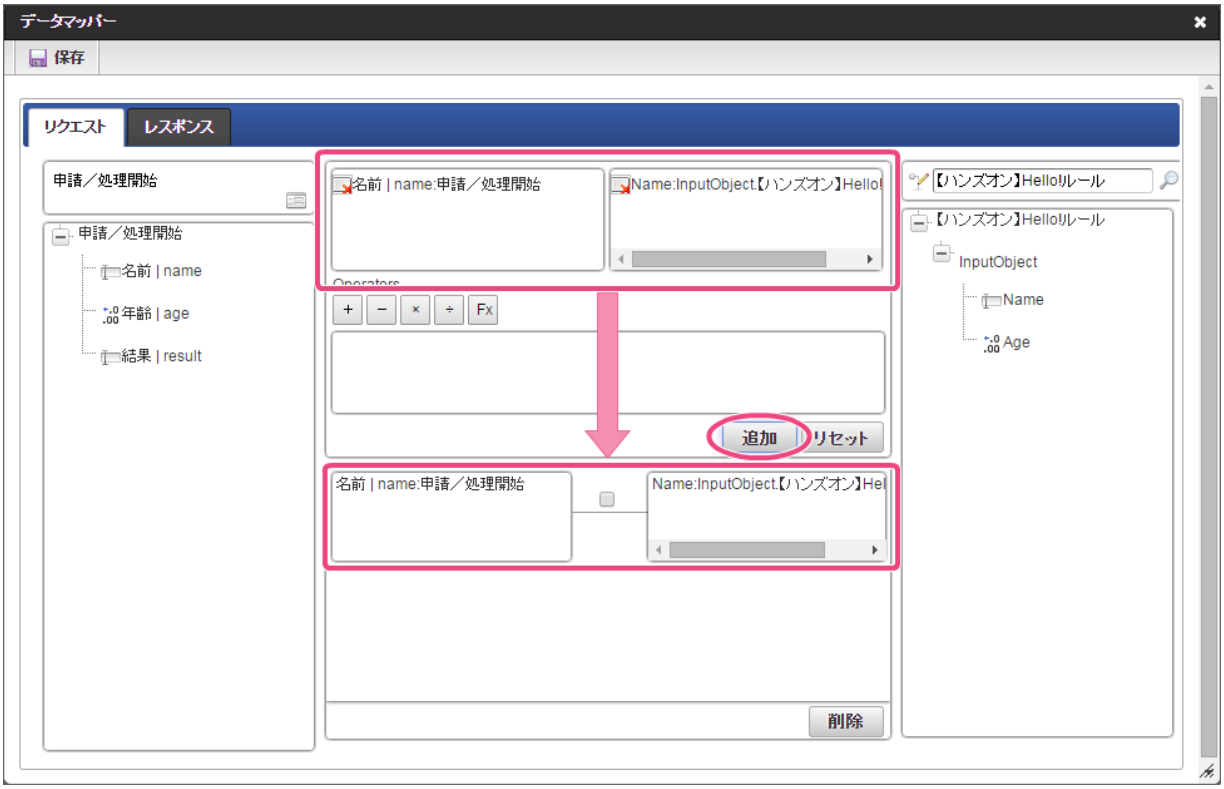
9. 左の欄から「名前」をクリックしてください。
   クリック後、中央左の欄に「名前 | name: 申請／処理開始」と表示されます。



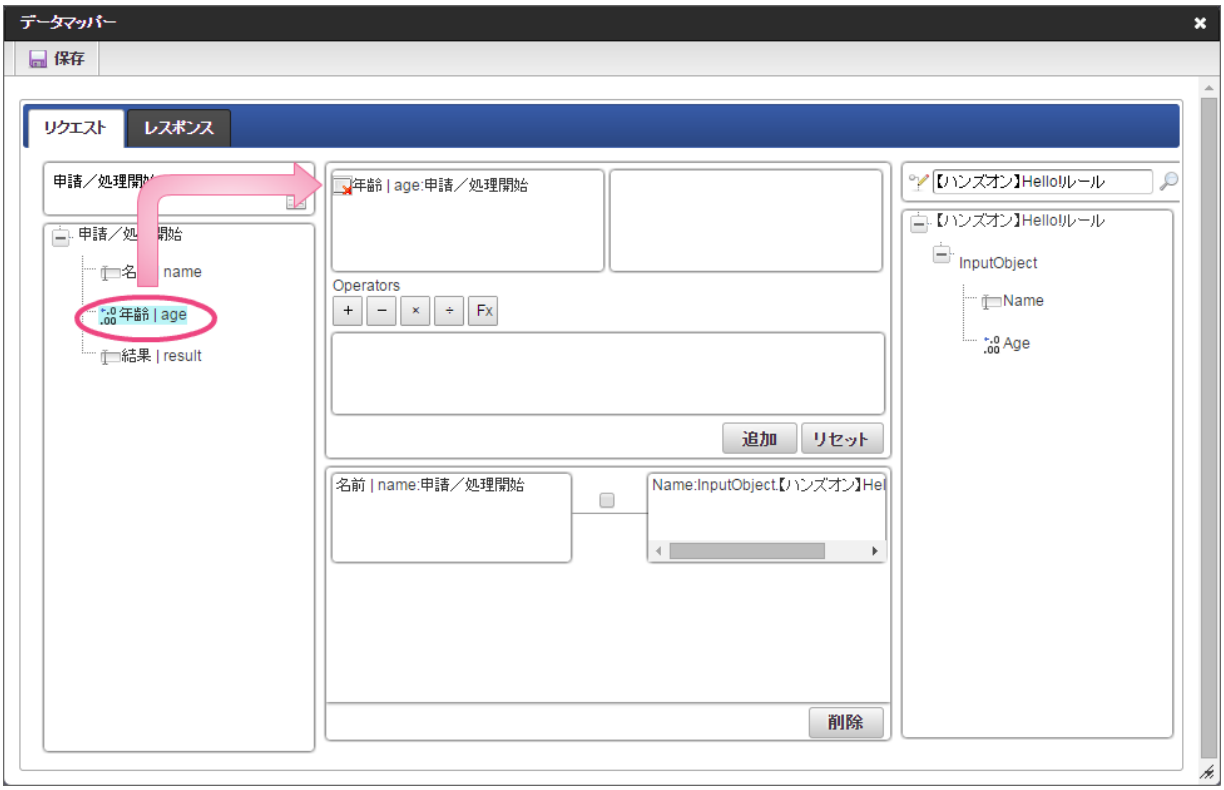
10. 右の欄から「Name」をクリックしてください。
    クリック後、中央右の欄に「Name:InputObject.【ハンズオン】Hello! ルール」と表示されます。



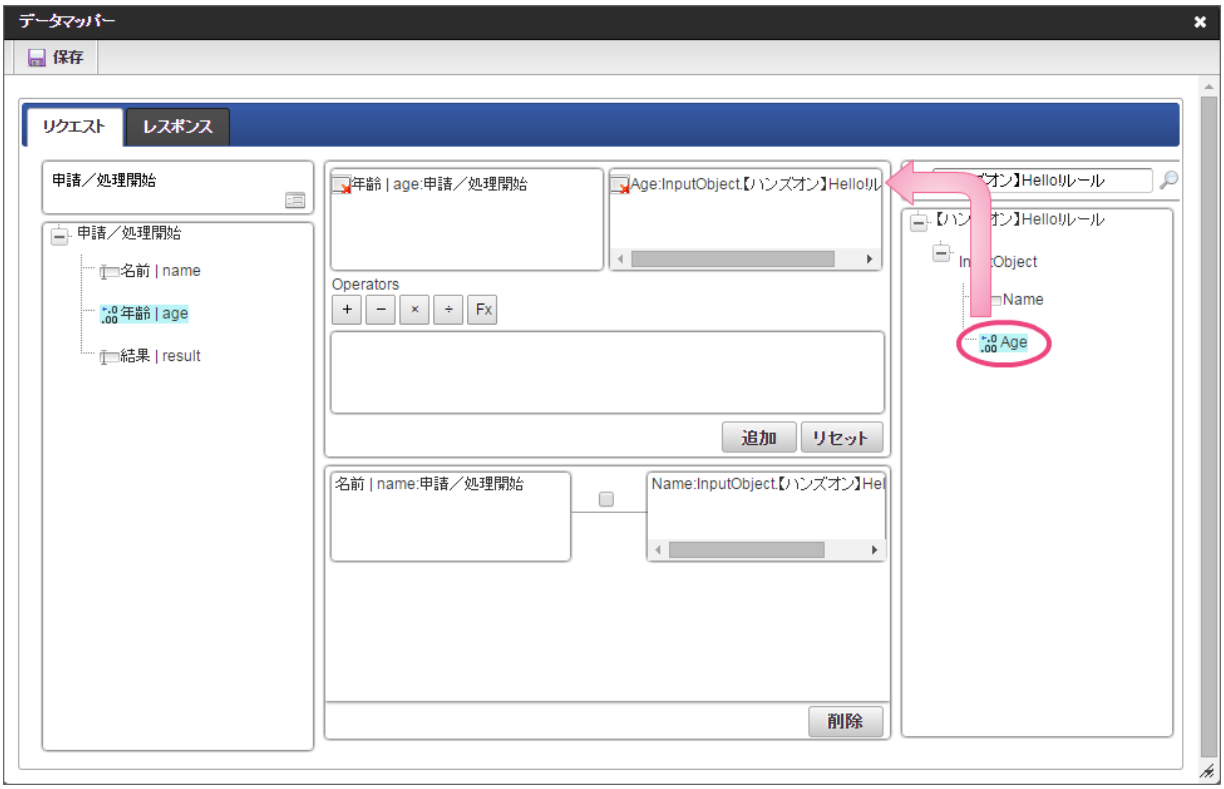
11. 「追加」をクリックしてください。
フォームの「名前」とデータソース定義の「Name」のマッピングが設定され、中央下段の欄に表示されます。



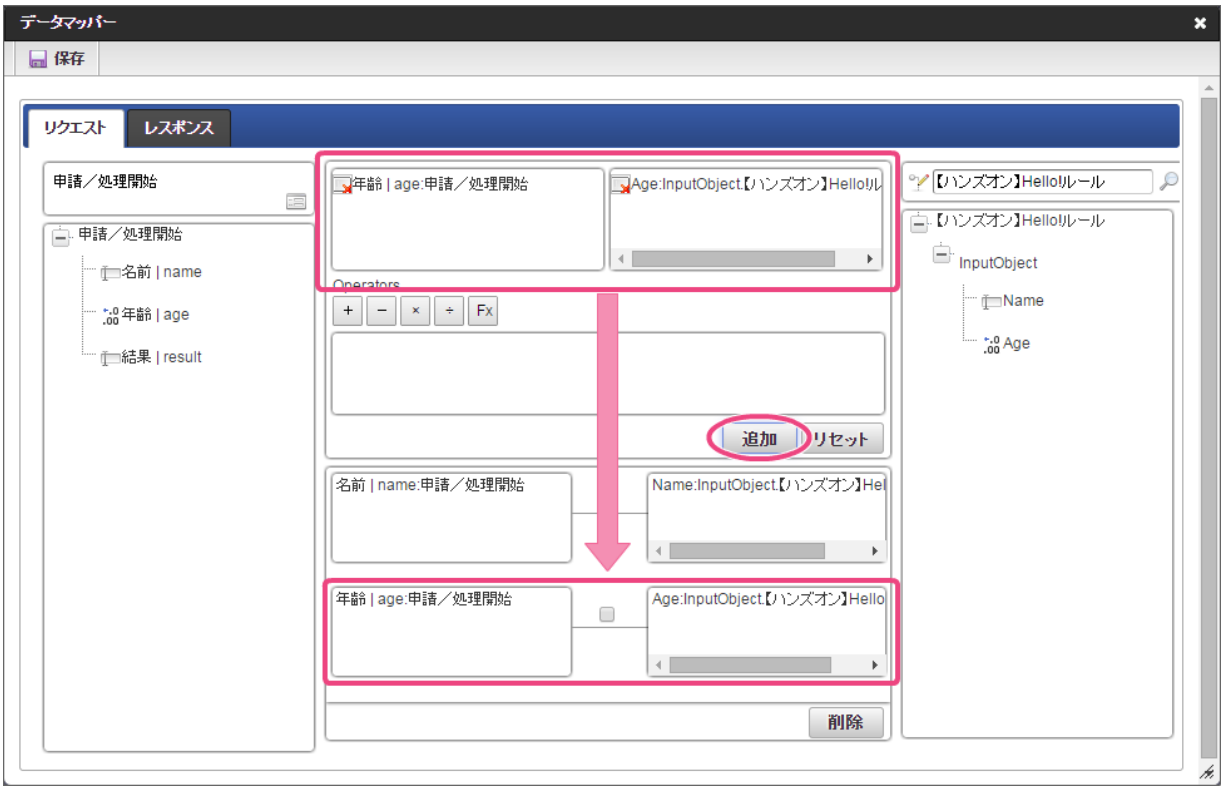
12. 左の欄から「年齢」をクリックしてください。
クリック後、中央左の欄に「年齢 | age: 申請／処理開始」と表示されます。



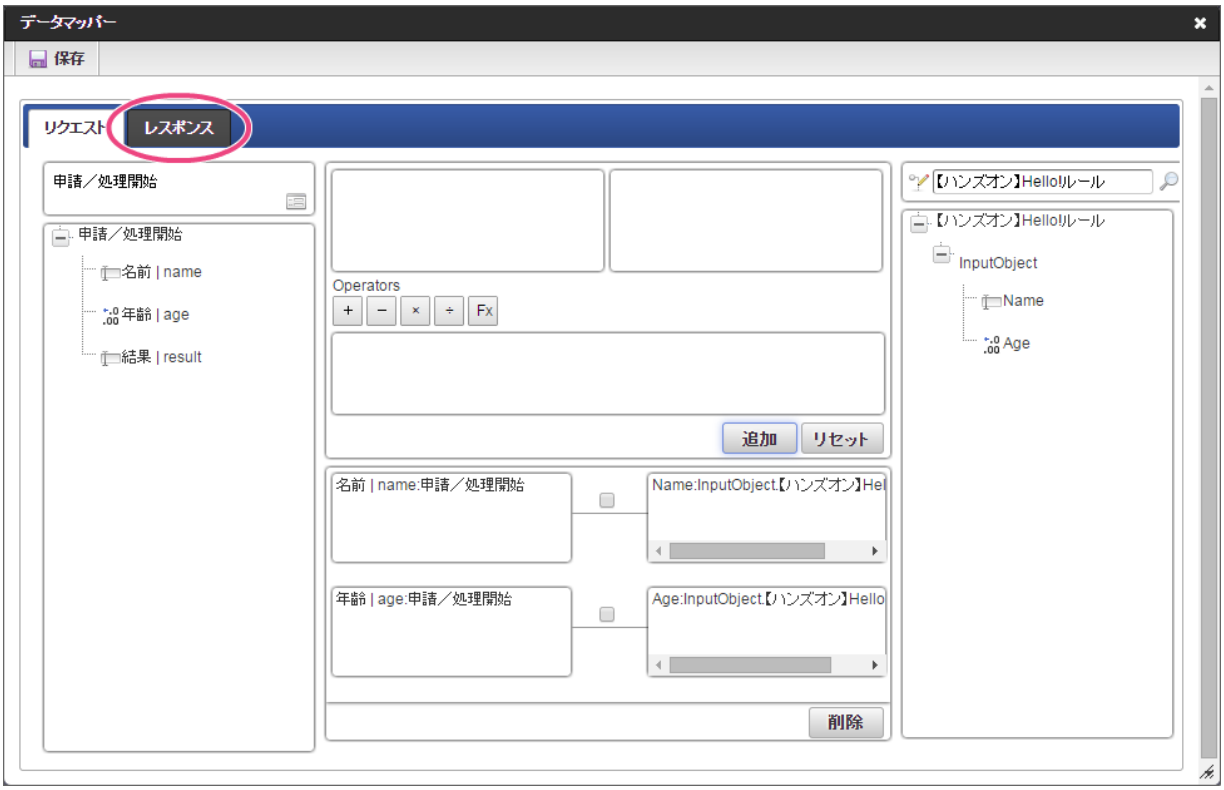
13. 右の欄から「Age」をクリックしてください。
クリック後、中央右の欄に「Age:InputObject.【ハンズオン】Hello! ルール」と表示されます。



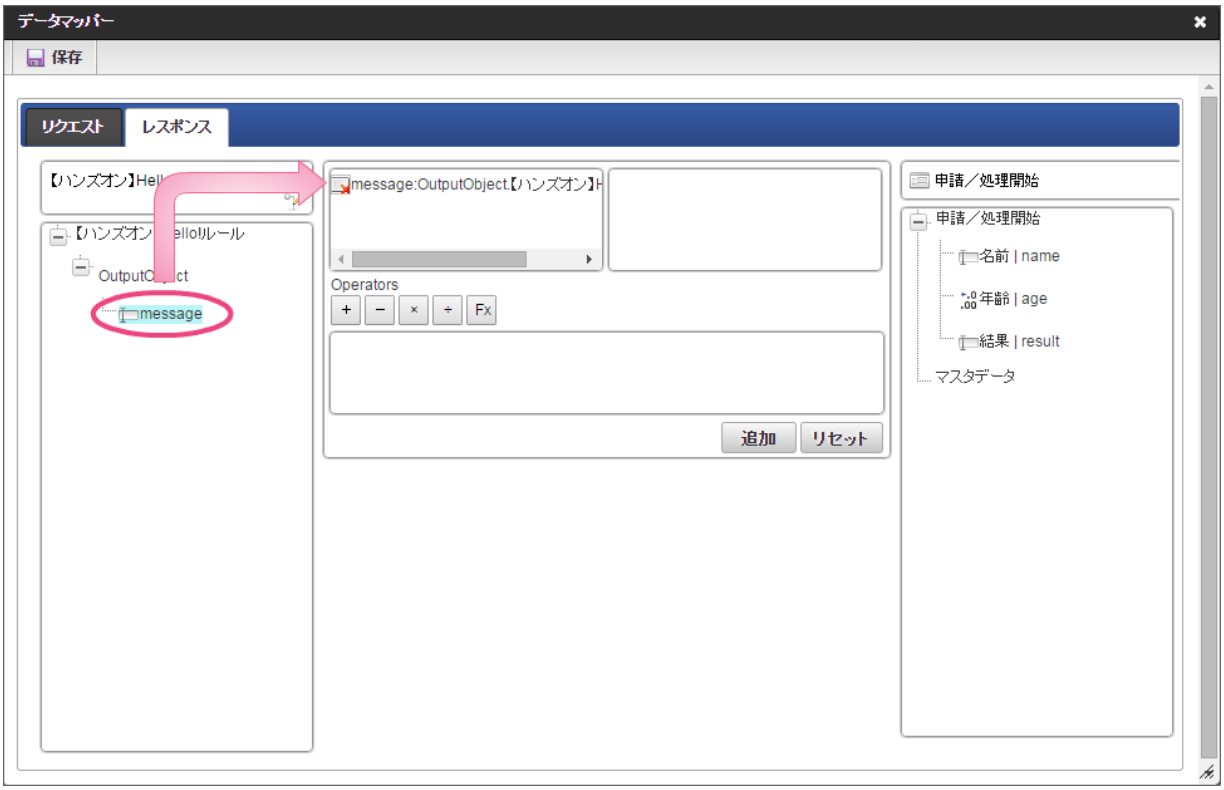
14. 「追加」をクリックしてください。
フォームの「年齢」とデータソース定義の「Age」のマッピングが設定され、中央下段の欄に表示されます。



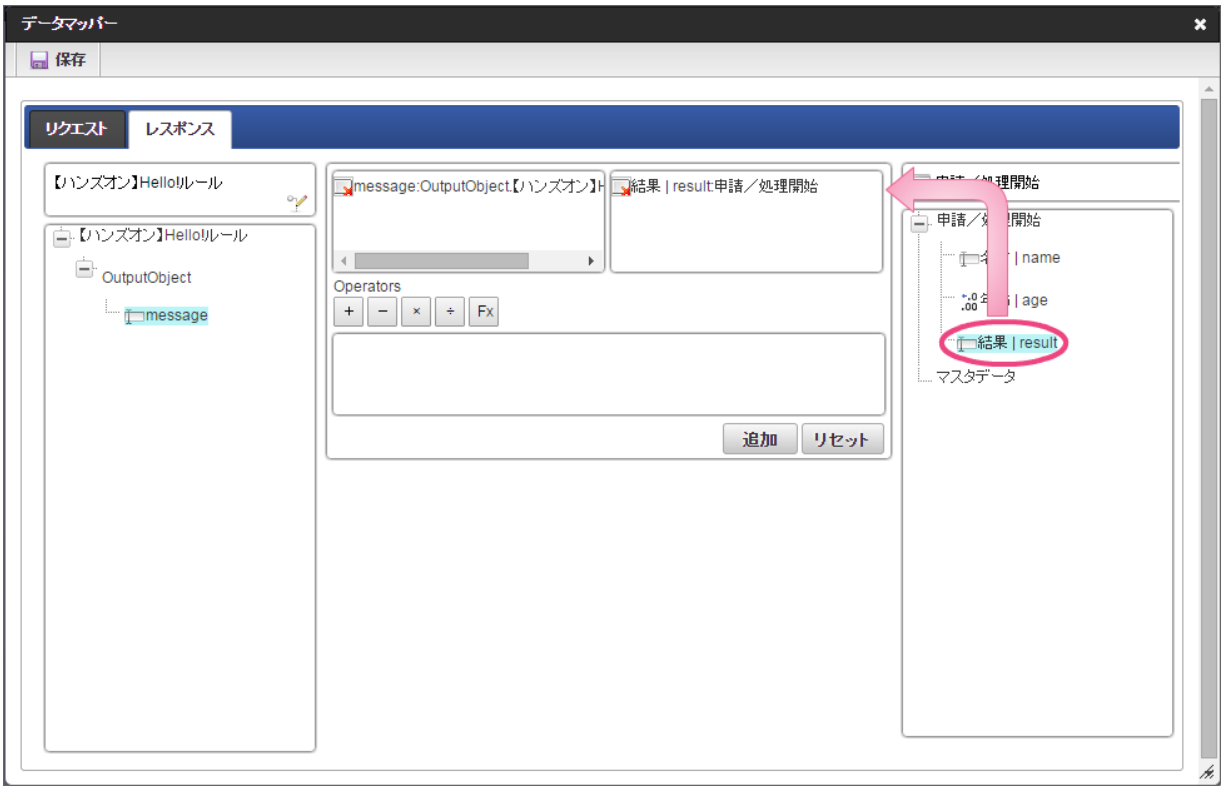
15. リクエストの設定が完了しましたので、レスポンスの設定を行うために「レスポンス」タブをクリックしてください。



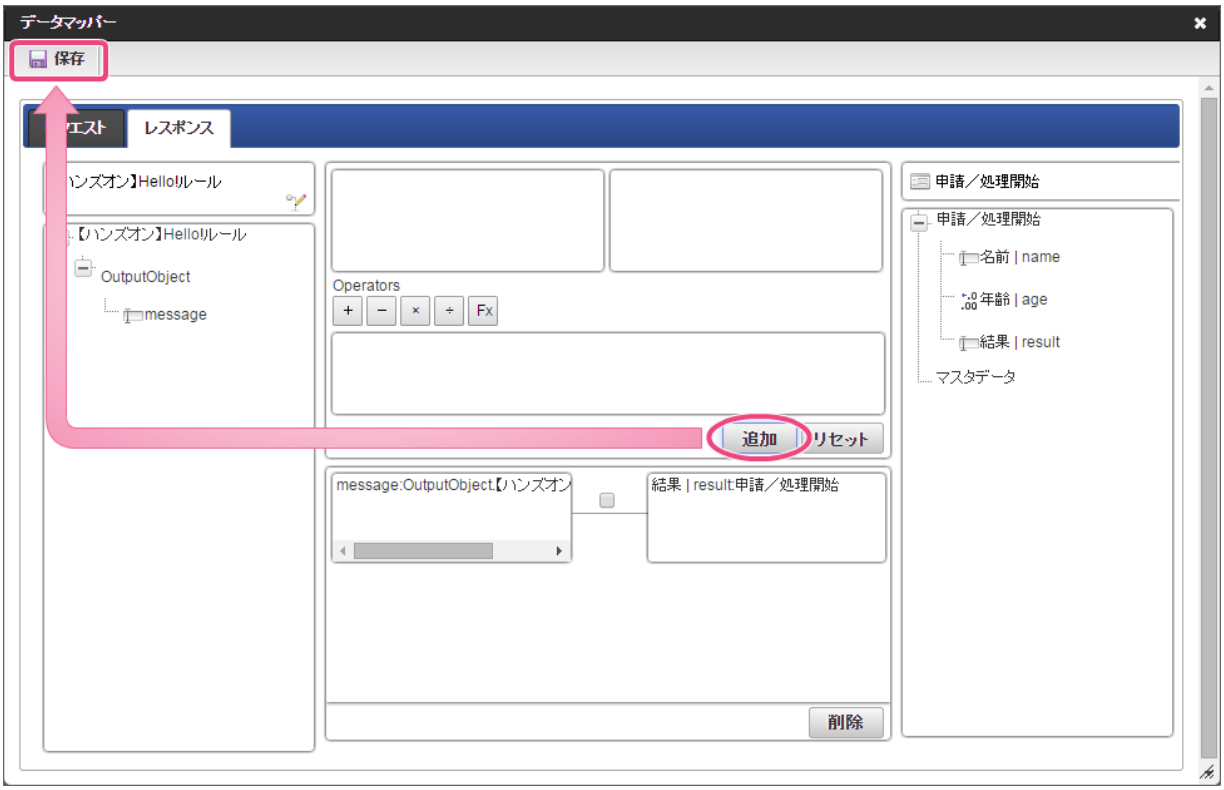
16. 左の欄から「message」をクリックしてください。
    クリック後、中央左の欄に「message:OutputObject.【ハンズオン】Hello! ルール」と表示されます。



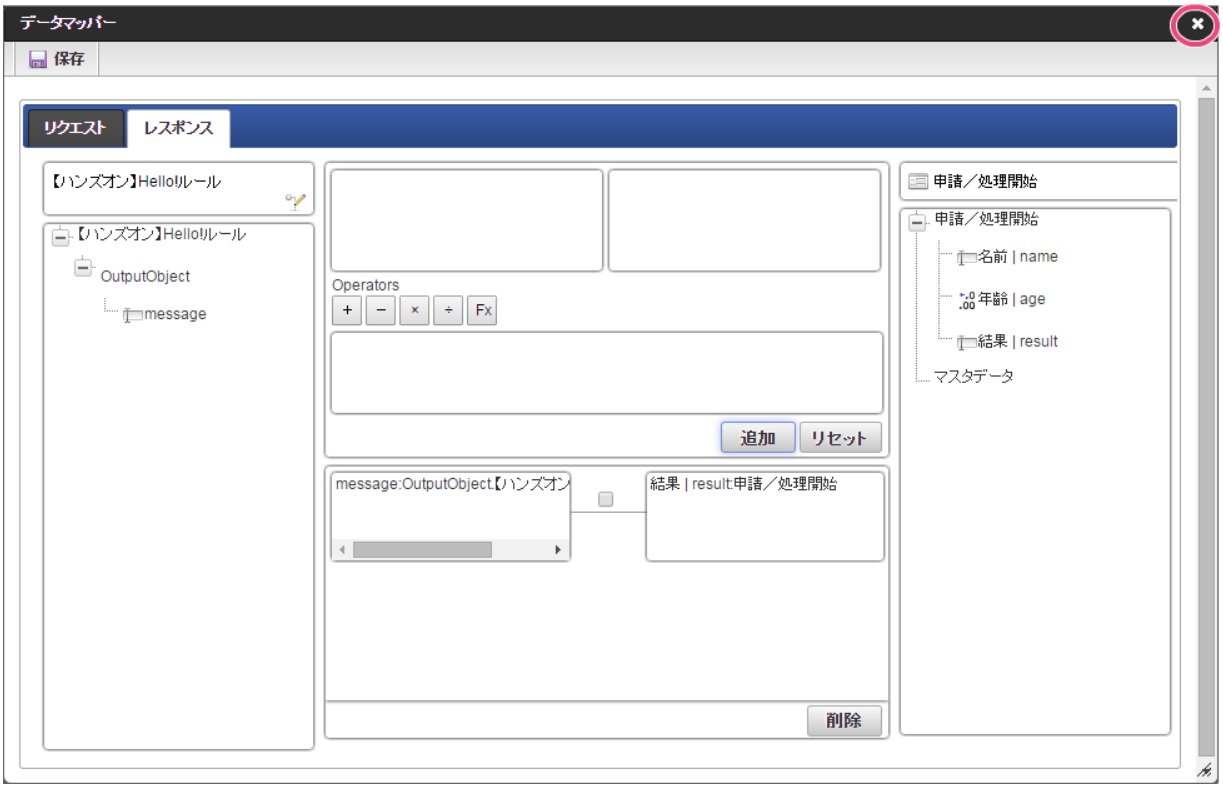
17. 右の欄から「結果」をクリックしてください。
クリック後、中央右の欄に「結果 | result」と表示されます。



18. 「追加」、「保存」の順にクリックしてください。



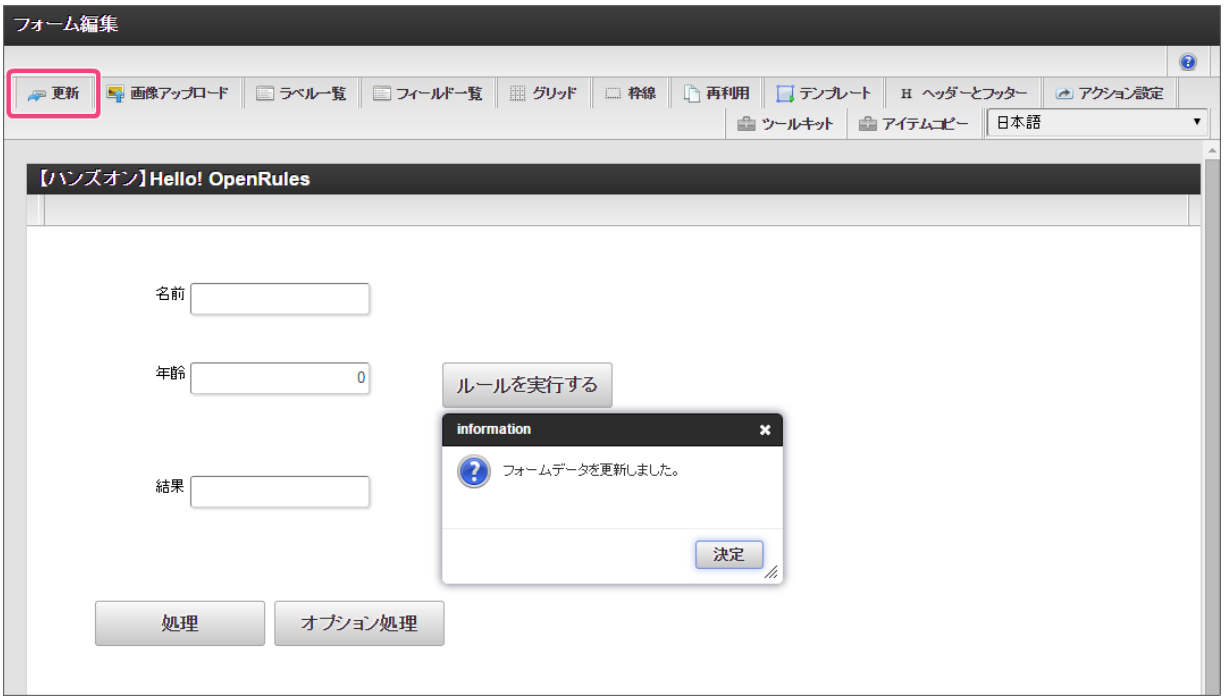
19. 正常に保存できたら、「データマッパー」は右上の「 ✖ 」をクリックして閉じてください。



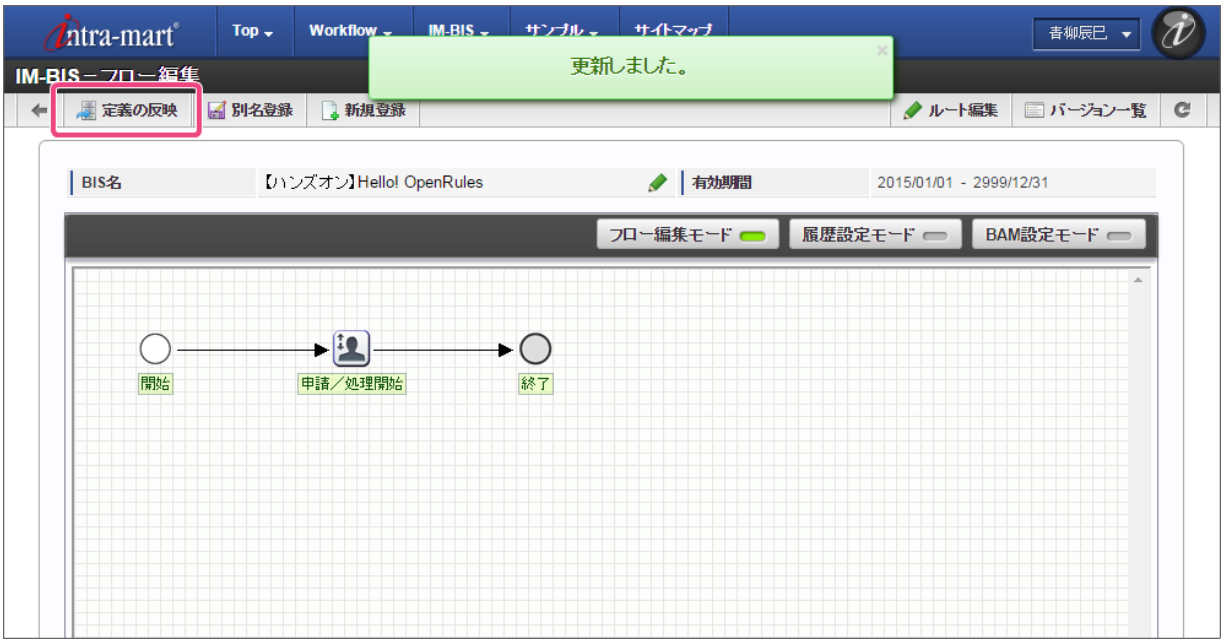
20. アクション設定で「確定」をクリックしてください。

21. イベント設定で「確定」をクリックしてください。

22. 「更新」をクリックして、フォーム（画面）を保存してください。



23. 最後に「定義の反映」をクリックして、フローの実行準備を行ってください。



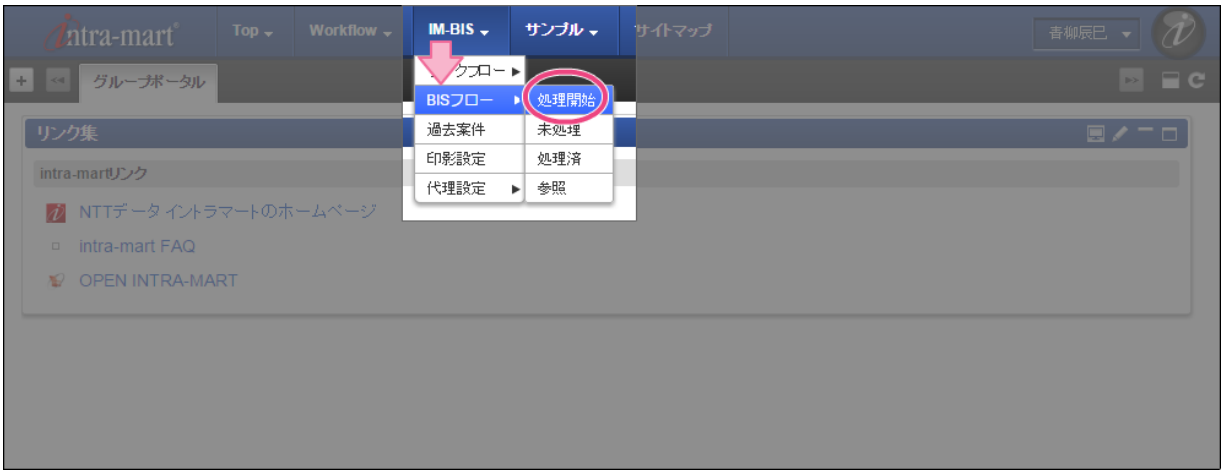
作成したルールを実行する

これまでのシナリオで作成した IM-BIS のフローを使って、ルールを実行してみましょう。

ルールと連携したフローを実行する手順
- OpenRules を IM-BIS の画面上で実行するための手順

OpenRules を IM-BIS の画面上で実行するための手順

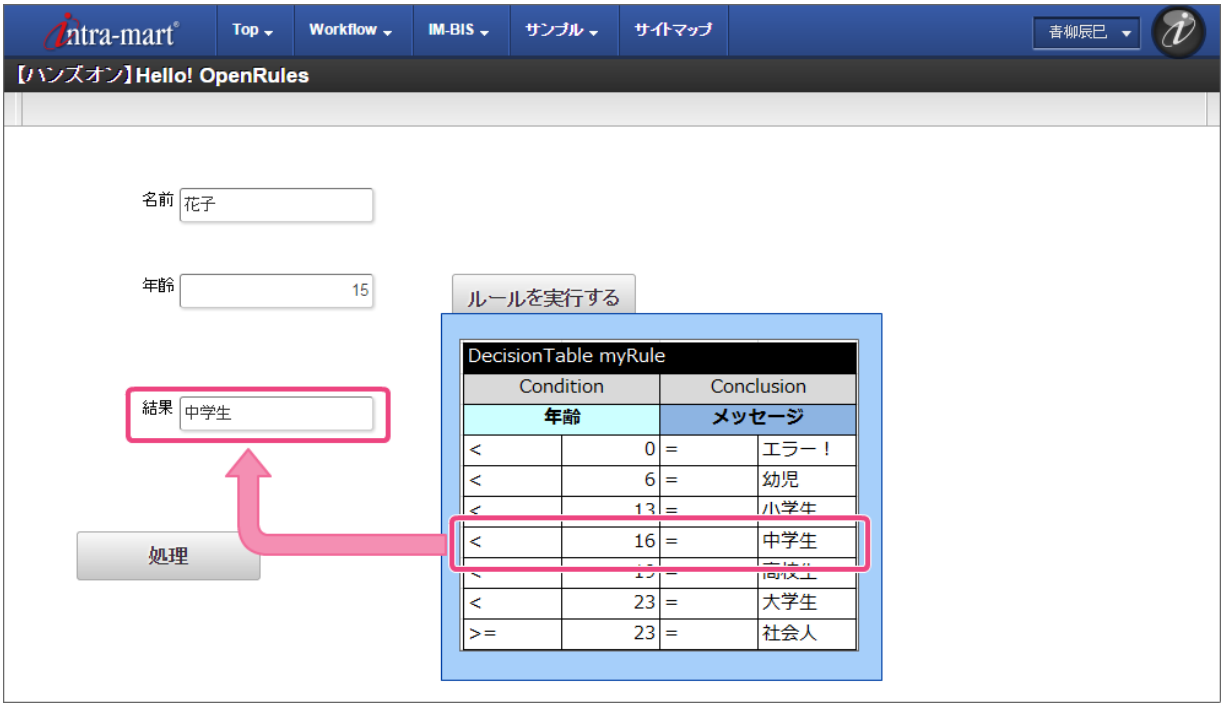
1. 「BIS担当者」ロールを付与したユーザでログインしましょう。
   （このマニュアルでは、「青柳辰巳」（ユーザコード：aoyagi）でログインします。）
2. 上部のメニューの「 IM-BIS 」→「BISフロー」の順にマウスを重ねてから「処理開始」をクリックしてください。



3. 「【ハンズオン】Hello! OpenRules」の「 申請／処理開始」をクリックしてください。

4. (1) 名前と(2) 年齢に任意の値を入力し、「ルールを実行する」をクリックしてください。

5. 入力した年齢に応じた結果が返却されることが確認できました。



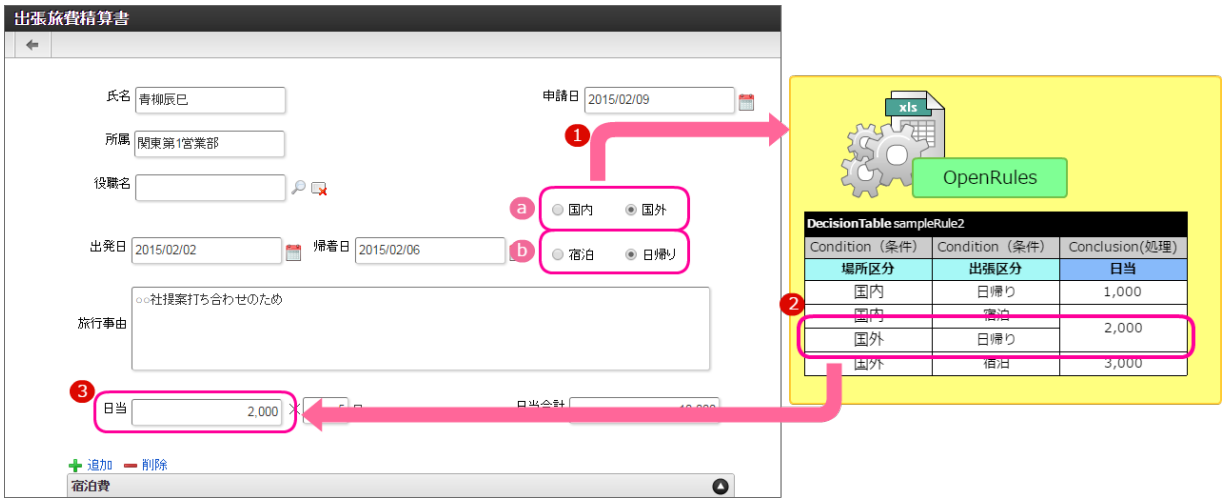
複合条件（AND/OR）を利用して、旅費精算の日当を計算してみよう

本章では、 OpenRules の応用への第一歩として、旅費精算の日当計算のハンズオンを通じて、複合条件（AND/OR）を含むルールの作成を実践するシナリオをまとめています。

ハンズオンシナリオ（出張手当申請の作成）の概要

このシナリオでは、複合条件（AND/OR）を含むルールを利用したフローを作成します。

- ユーザが出張旅費の申請を入力すると、入力内容に応じた日当金額を計算して画面に表示する



【処理概要】

1. (a)「場所区分」（国内／国外）、(b)「出張区分」（宿泊／日帰り）を入力すると、ルールが実行される
2. 以下の条件を評価し、合致する条件の結果を「日当」として返却する

   【日当支給に関する規則】

   1. 出張場所が国内かつ日帰りである場合
      日当は1,000円
   2. 出張場所が国内かつ宿泊を伴う場合、または出張場所が国外かつ日帰りの場合
      日当は2,000円
   3. 出張場所が国外かつ宿泊を伴う場合
      日当は3,000円
3. 返却結果を画面に表示する

このシナリオを通して習得できる知識

このシナリオを実行すると、以下の知識を理解することができます。

- OpenRules での複合条件を用いたルールの定義方法
- OpenRules の汎用テンプレートを拡張したルールの定義方法

このシナリオを学習する前に必要な知識

このシナリオの実行に当たり、下記の知識を事前に確認してください。

- IM-BIS 、 IM-FormaDesigner の定義ファイルのインポート方法

OpenRules で AND/ORを含む条件を定義したルールを作成する

OpenRules で複雑な条件の1つとして、AND（条件○○と条件△△の両方を満たす場合）やOR（条件○○、または条件△△のどちらかを満たす場合）は、セルの結合などを利用して表現します。

このハンズオンでは、出張旅費精算フローをサンプルに、ANDやORを組み合わせた条件の記述方法を確認することができます。

また、 OpenRules のルール定義のファイルでは、 項目を論理名で扱うことができますが、 IM-BIS のセレクトボックスなどの画面アイテムでは、OpenRules に受け渡す値は半角英数字にする必要があるため、セレクトボックスなどの値を扱う場合には、ルール定義ファイルで表示値・送信値のマッピングを行わなければなりません。
このハンズオンでは、そのようなアイテムの一種として、ラジオボタンの値を扱う場合のマッピング方法も同時に確認していきます。

ルールを定義するExcelファイルを作成する手順
- このシナリオで作成するルールの概要
- OpenRules で AND/OR条件を表現する方法
- ルールのExcelファイルを作成する手順
- 旅費精算の日当計算のハンズオンを開始するための準備
- 日当を計算するための DecisionTable を作成する
- OpenRules でアイテムの送信値・表示値のマッピングを設定する
- OpenRules で実行する DecisionTable の順序をコントロールする
- 実行に必要な定義を設定する

このシナリオで作成するルールの概要

- 作成するルールの内容
  1. 画面（フォーム）から受け渡されたラジオボタンの送信値を表示値に変換した上で、条件を評価する
  2. 出張旅費精算申請の「場所区分」「出張区分」に応じて、日当金額を設定する
     - 入力値：場所区分、出張区分
     - 出力値：日当

  条件と返却する日当の金額の組み合わせは、以下の通りです。

  - 業務命令に基づく出張に係る日当

場所区分	出張区分	日当金額
国内	日帰り	1,000円
	宿泊	2,000円
国外	日帰り	2,000円
	宿泊	3,000円

  - 場所区分が「国内」かつ出張区分が「日帰り」の場合、日当は1,000円
  - 場所区分が「国内」かつ出張区分が「宿泊」の場合、
    または、場所区分が「国外」かつ出張区分が「日帰り」の場合、日当は2,000円
  - 場所区分が「国外」かつ出張区分が「宿泊」の場合、日当は3,000円

OpenRules で AND/OR条件を表現する方法

このハンズオンを開始する前に、 OpenRules で、AND/ORを利用した条件を表現する方法を確認しましょう。
AND/ORを伴う条件の記述方法には、以下の方法があります。

1つの項目に対してのOR条件（いずれかと一致する）

1つの項目のとりうる値を「OR条件」（いずれかと一致する）を表現するには、演算子の「Is One Of」を利用することができます。
以下は、都道府県から地方名を判断するために「Is One Of」を使った例です。

DecisionTable sampleArea			
Condition		Conclusion	
国（Country）		**地域（Area）**	
Is One Of	日本（Japan）， 中華人民共和国（China）， シンガポール（Singapore）	=	アジア （Asia）
Is One Of	フランス（France）， ドイツ（Germany）， イタリア（Italy）	=	ヨーロッパ （Europe）

上の表では、以下の通りに条件を評価します。

- 国が「日本」「中国」「シンガポール」のいずれかと一致する場合、「地域」に「アジア」という値を返却します。
- 国が「フランス」「ドイツ」「イタリア」のいずれかと一致する場合、「地域」に「ヨーロッパ」という値を返却します。

2つ以上の項目に対してのAND/OR条件（両方の条件と一致する、いずれかの条件と一致する）

2つ以上の項目を組み合わせたAND/OR条件は、 *DecisionTable* での行やセルの結合によって表現することができます。

- 「AND条件」は、 *DecisionTable* の同じ行に異なる項目に対する条件の基準値を記述することで表現することができます。
- 「OR条件」は、 *DecisionTable* の対象の複数の条件の評価（結果）のセルを結合することで表現することができます。

以下は、「製品○○の売上額」と「製品△△の売上額」に基づいて営業成績を評価する例です。

DecisionTable businessResult						
Condition		Condition		Conclusion		
製品○○の売上 (Sales of Product oo)		**製品△△の売上** (Sales of Product ΔΔ)		**営業評価** (Evaluation of sales performance)		
>=	1,000,000	>=	500,000	=	A（Good）	a
>=	1,000,000			=	B（Satisfactory）	b
		>=	500,000			
				=	C（Poor）	c

上の表では、以下の通りに条件を評価します。

a. **AND条件**
「製品○○の売上が100万円以上」と「製品△△の売上が50万円以上」の両方を満たす場合、営業評価として「A」を返却します。

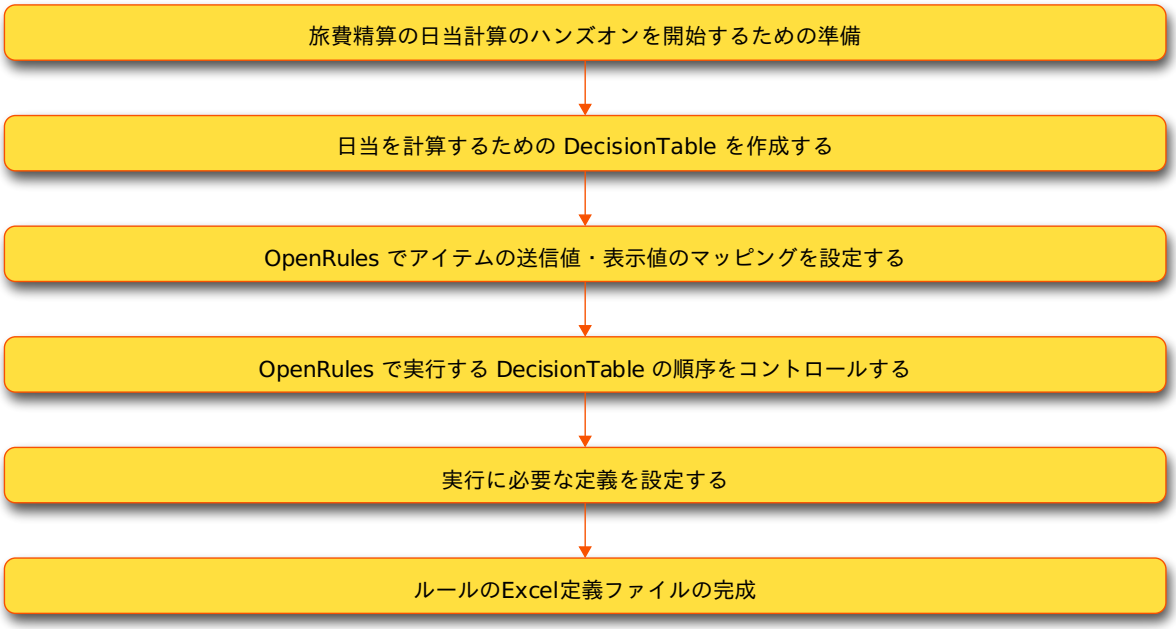
b. **OR条件**
「製品○○の売上が100万円以上」と「製品△△の売上が50万円以上」のいずれかを満たす場合、営業評価として「B」を返却します。

c. **無条件（デフォルト条件）**
上の条件のどれにも合致しない場合、営業評価として「C」を返却します。

ルールのExcelファイルを作成する手順

今回は、 *OpenRules のテンプレート* の「汎用テンプレート」を変更しながらルール定義のExcelファイルを作成します。
このシナリオでは、以下の図の流れで作成していきます。

- 旅費精算の日当を計算するルールの作成の手順



旅費精算の日当計算のハンズオンを開始するための準備

汎用テンプレートの編集を開始する

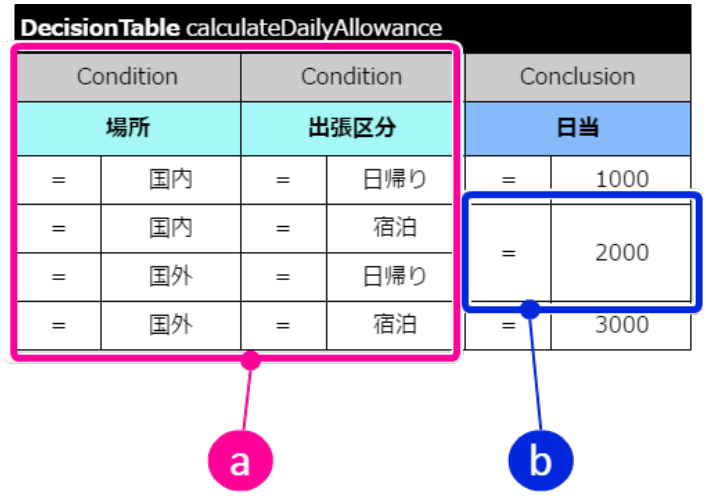
このハンズオンでは、ダウンロードの章で公開しているテンプレートを変更しながら、申請画面で実行可能な OpenRules の処理を定義します。
まずは、テンプレートファイルを入手しましょう。

1. *OpenRules のテンプレート* から「汎用テンプレート」をダウンロードしてください。
2. ファイルを別名で保存した後に、ファイルの編集を開始します。

日当を計算するための DecisionTable を作成する

最初に、申請書に入力された「出張区分」と「場所区分」に基づいて日当を計算する *DecisionTable* を作成しましょう。

以下の図のようにExcel上にまとめていきます。



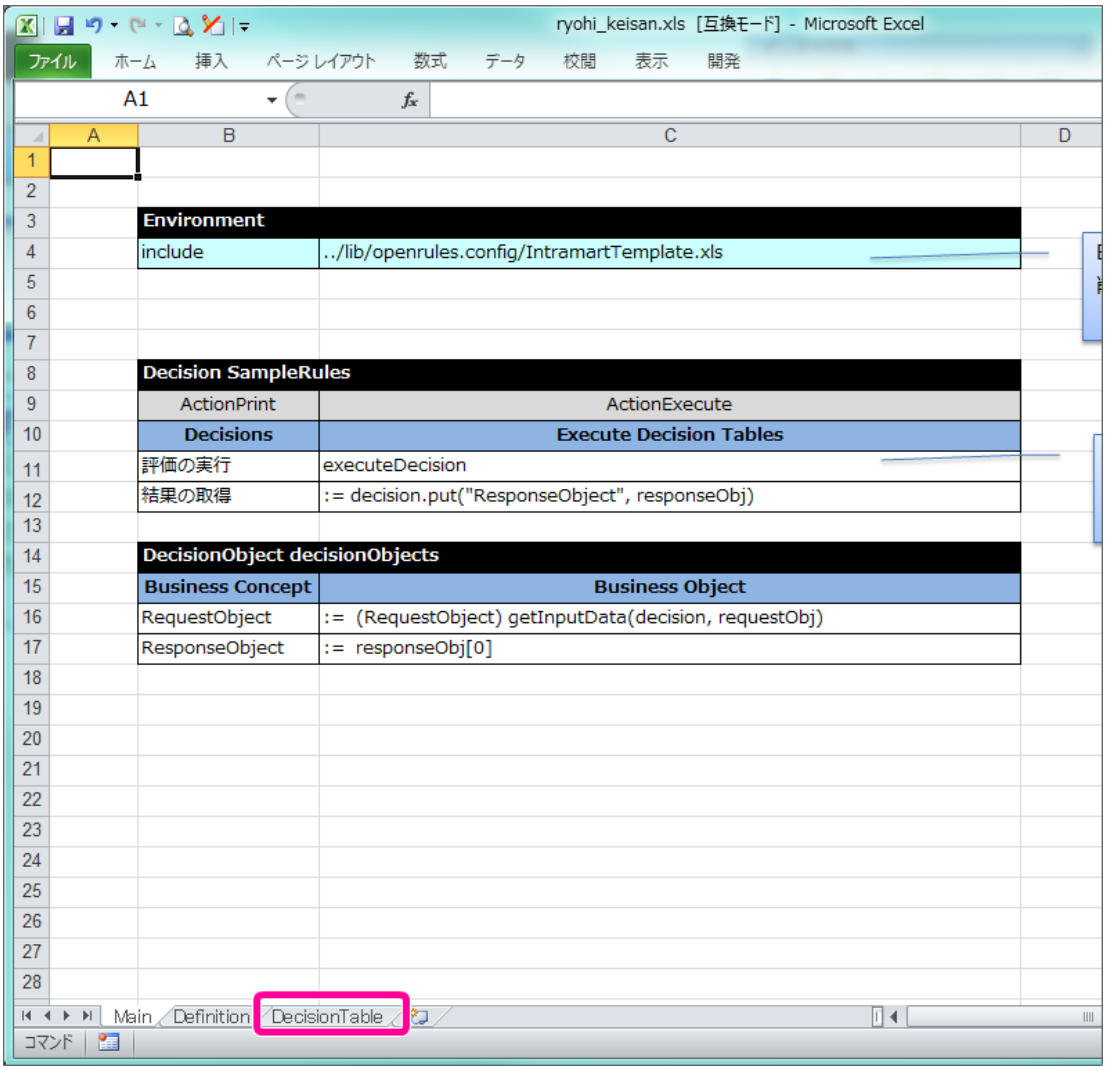
a. **異なる複数の項目（列）の条件**
   OpenRules において、条件が複数の項目（列）で構成される場合、「AND」（全ての項目の条件を満たす）条件として扱われます。

b. **異なる複数のパターン（行）の条件**
   OpenRules において、評価の列を縦方向に結合した場合、「OR」（いずれかの組み合わせの条件を満たす）条件として、結合セルに隣接する条件を満たした場合には同じ結果を返却します。

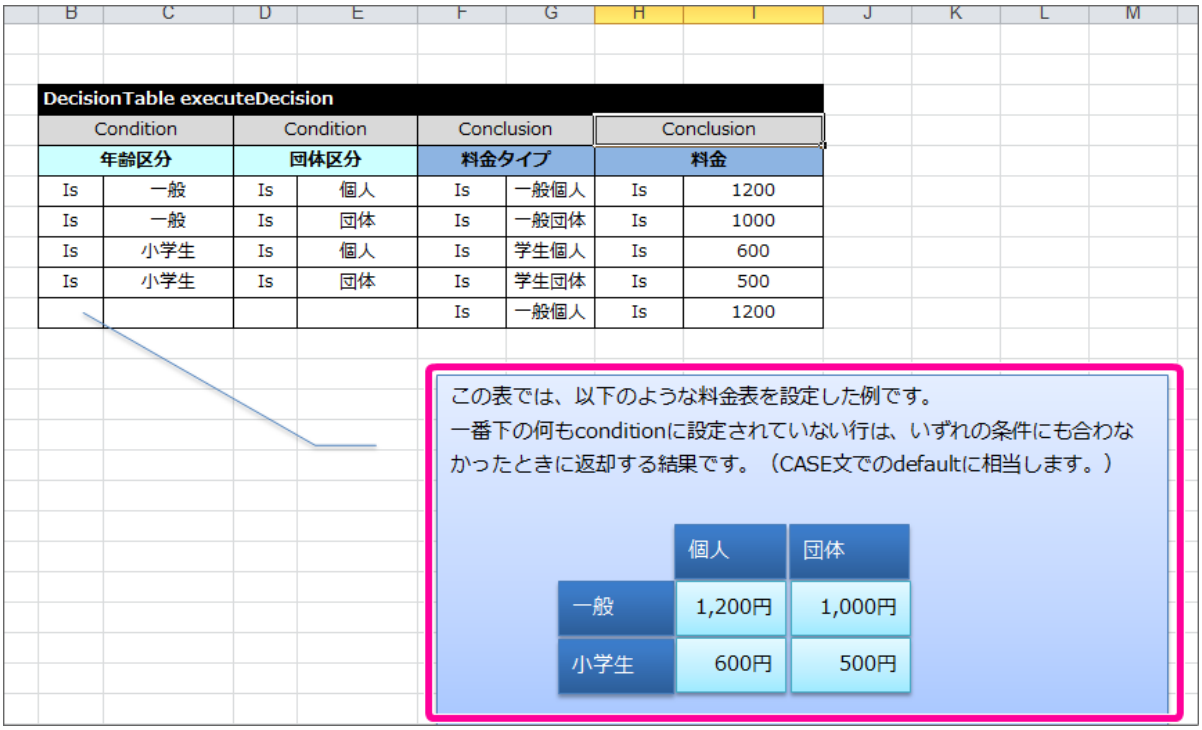
DecisionTable の項目を設定する

最初に場所（国内・国外）と出張区分（日帰り・宿泊）から日当を計算する *DecisionTable* を作成しましょう。

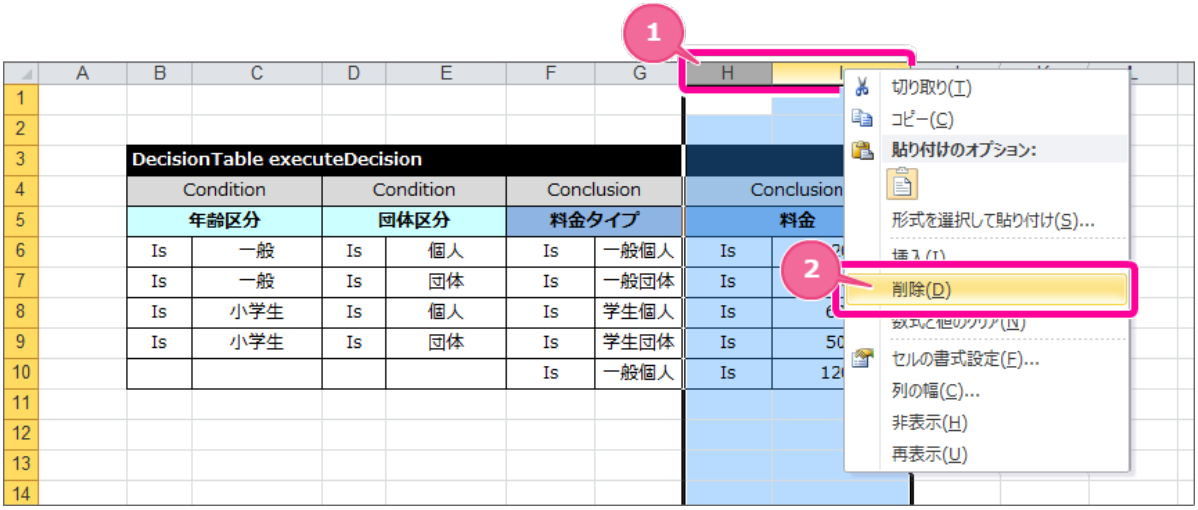
1. 編集中のExcelファイルの「DecisionTable」シートを表示してください。



2. テンプレートには、説明が添付されていますので、枠で囲まれた部分を削除してください。

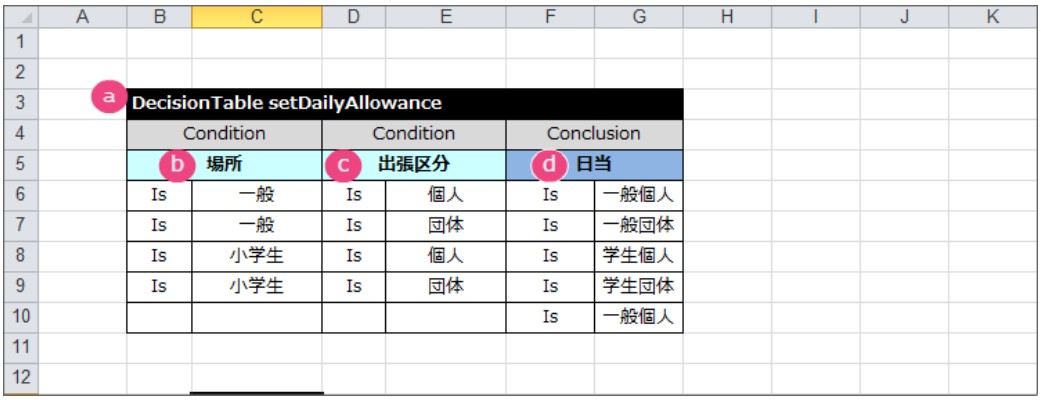


3. 今回作成する *DecisionTable* は、*Condition* が2項目、 *Conclusion* が1項目となるため、以下の手順で不要な列を削除してください。



1. 不要な *Conclusion* の列（図中ではH列とI列）をドラッグして選択状態にし、右クリックしてください。
2. メニューから「削除」をクリックしてください。

4. *DecisionTable* のテーブル名と項目名を以下の通りに入力してください。



a. テーブル名
   キーワード *DecisionTable* から半角スペースを空けて「setDailyAllowance」と入力してください。

b. 条件の項目の論理名(1)
   「場所」と入力してください。

c. 条件の項目の論理名(2)
   「出張区分」と入力してください。

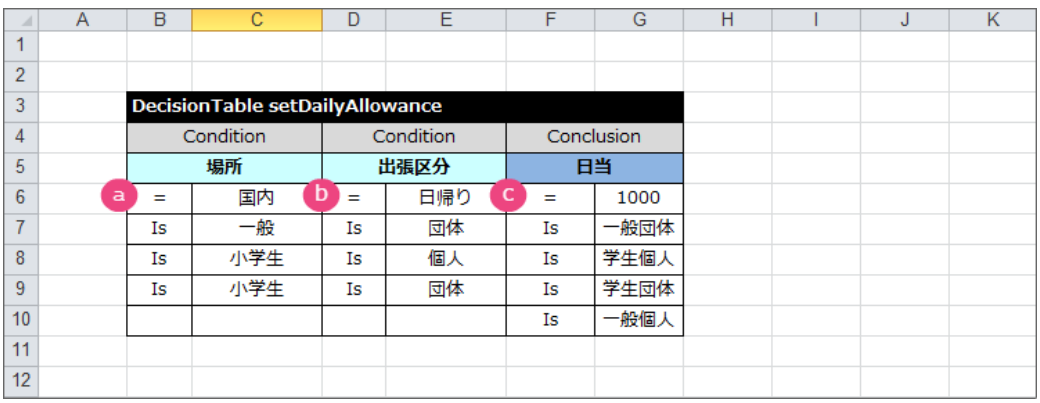
d. 結果の項目の論理名
   「日当」と入力してください。

5. これで、 *DecisionTable* の項目が設定できましたので、各行に条件と評価を入力していきましょう。

DecisionTable の条件・評価を設定する

先の手順で設定した *DecisionTable* に条件と評価（結果）を設定していきましょう。

1. まずは、場所が「国内」、出張区分が「日帰り」の条件と日当「1,000円」を設定します。
   以下の通り、1行目の条件・評価を入力してください。



a. 場所
- 左のセルに演算子として「=」を入力してください。
- 右のセルに比較する値として「国内」を入力してください。

b. 出張区分
- 左のセルに演算子として「=」を入力してください。
- 右のセルに比較する値として「日帰り」を入力してください。

c. 日当
- 左のセルに演算子として「=」を入力してください。
- 右のセルに条件に合致したときに返却する値として「1000」を入力してください。

2. 続いて、場所区分が「国内」、出張区分が「宿泊」の条件と日当「2,000円」を設定します。
以下の通り、2行目の条件・評価を入力してください。

	DecisionTable setDailyAllowance					
	Condition		Condition		Conclusion	
	場所		出張区分		日当	
	=	国内	=	日帰り	=	1000
a	=	国内	b =	宿泊	c =	2000
	Is	小学生	Is	個人	Is	学生個人
	Is	小学生	Is	団体	Is	学生団体
					Is	一般個人

a. 場所
- 左のセルに演算子として「=」を入力してください。
- 右のセルに比較する値として「国内」を入力してください。

b. 出張区分
- 左のセルに演算子として「=」を入力してください。
- 右のセルに比較する値として「宿泊」を入力してください。

c. 日当
- 左のセルに演算子として「=」を入力してください。
- 右のセルに条件に合致したときに返却する値として「2000」を入力してください。

3. 場所区分が「国外」、出張区分が「日帰り」の条件と日当「2,000円」を設定します。
以下の通り、3行目の条件・評価を入力してください。

	DecisionTable setDailyAllowance					
	Condition		Condition		Conclusion	
	場所		出張区分		日当	
	=	国内	=	日帰り	=	1000
	=	国内	=	宿泊	=	2000
a	=	国外	b =	日帰り	c =	2000
	Is	小学生	Is	団体	Is	学生団体
					Is	一般個人

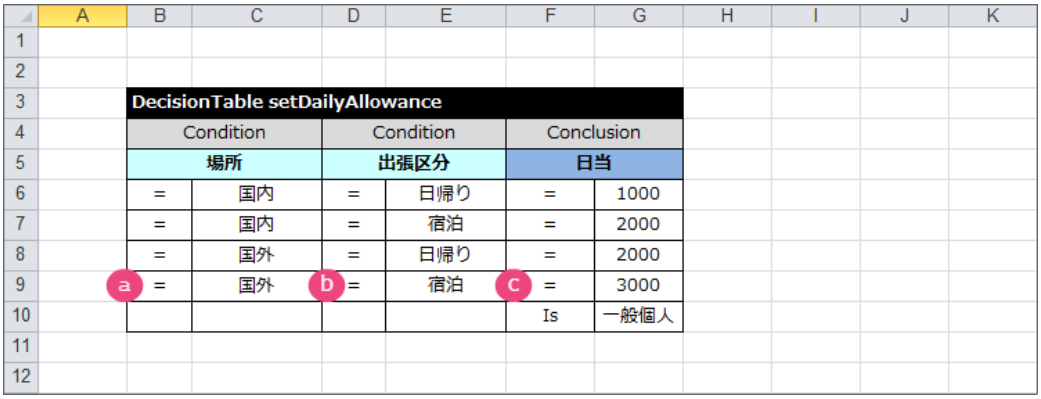
a. 場所
- 左のセルに演算子として「=」を入力してください。
- 右のセルに比較する値として「国外」を入力してください。

b. 出張区分
- 左のセルに演算子として「=」を入力してください。
- 右のセルに比較する値として「日帰り」を入力してください。

c. 日当
- 左のセルに演算子として「=」を入力してください。
- 右のセルに条件に合致したときに返却する値として「2000」を入力してください。

4. 場所区分が「国外」、出張区分が「宿泊」の条件と日当「3,000円」を設定します。

以下の通り、4行目の条件・評価を入力してください。



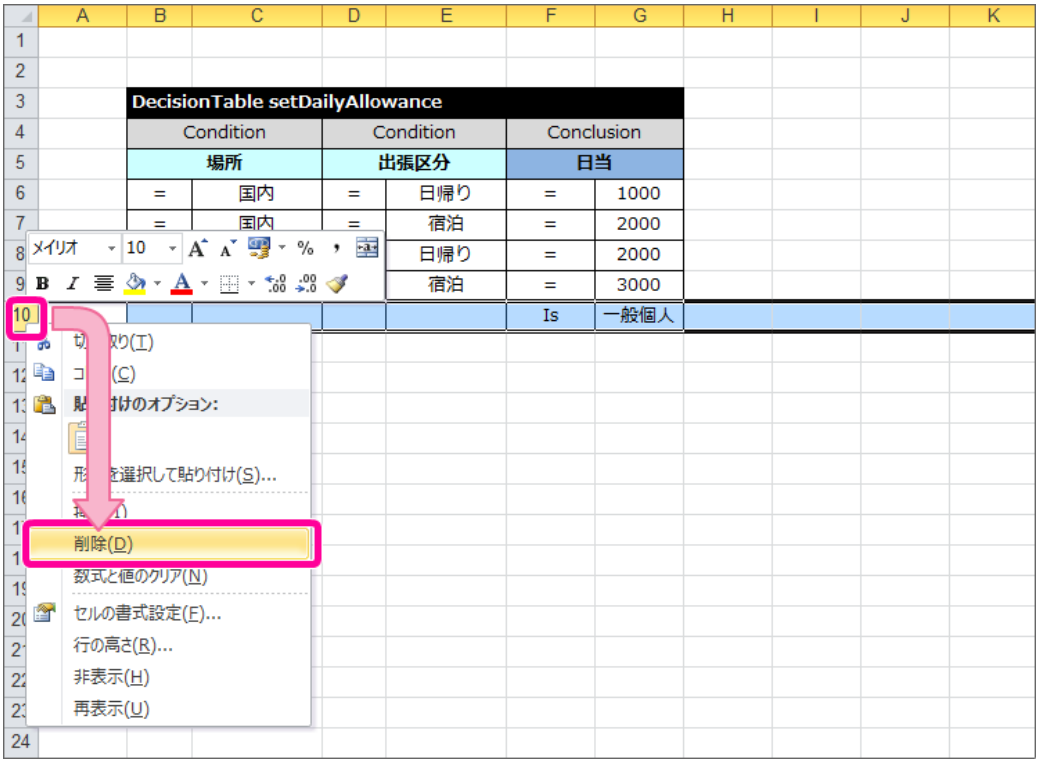
DecisionTable setDailyAllowance						
Condition		Condition		Conclusion		
場所		出張区分		日当		
=	国内	=	日帰り	=	1000	
=	国内	=	宿泊	=	2000	
=	国外	=	日帰り	=	2000	
a =	国外	b =	宿泊	c =	3000	
				Is	一般個人	

a. 場所
- 左のセルに演算子として「=」を入力してください。
- 右のセルに比較する値として「国外」を入力してください。

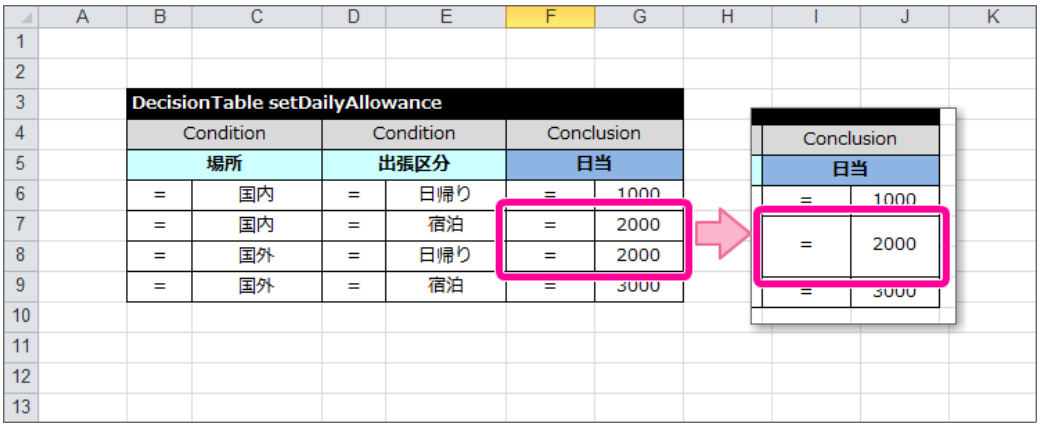
b. 出張区分
- 左のセルに演算子として「=」を入力してください。
- 右のセルに比較する値として「宿泊」を入力してください。

c. 日当
- 左のセルに演算子として「=」を入力してください。
- 右のセルに条件に合致したときに返却する値として「3000」を入力してください。

5. 5行目については、不要な行のため、対象の行の上（図中では10行目）で右クリックし、削除をクリックしてください。



6. 作成した *DecisionTable* を確認すると、結果に「日当」の「2,000円」が2回登場しており、対応する条件２つのどちらかを満たす、つまりOR条件で設定する必要があります。
OpenRules では、OR条件を設定するには、セルを結合する必要があるため、この日当の演算子・値のセルを結合してください。



DecisionTable setDailyAllowance

Condition		Condition		Conclusion	
場所		出張区分		日当	
=	国内	=	日帰り	=	1000
=	国内	=	宿泊	=	2000
=	国外	=	日帰り	=	2000
=	国外	=	宿泊	=	3000

Conclusion	
日当	
=	1000
=	2000
=	3000

7. これで、日当を計算する *DecisionTable* が完成しましたので、一度ファイルを保存しましょう。

OpenRules でアイテムの送信値・表示値のマッピングを設定する

先の手順で *DecisionTable* では、申請画面で入力された「場所」や「出張区分」の値に基づいて、日当を計算するように作成しました。
後の手順で IM-BIS と連携する場合の画面（フォーム）では、これらの項目は「ラジオボタン」とで設定しており、 OpenRules には半角英数字のコードの「送信値」が受け渡されます。

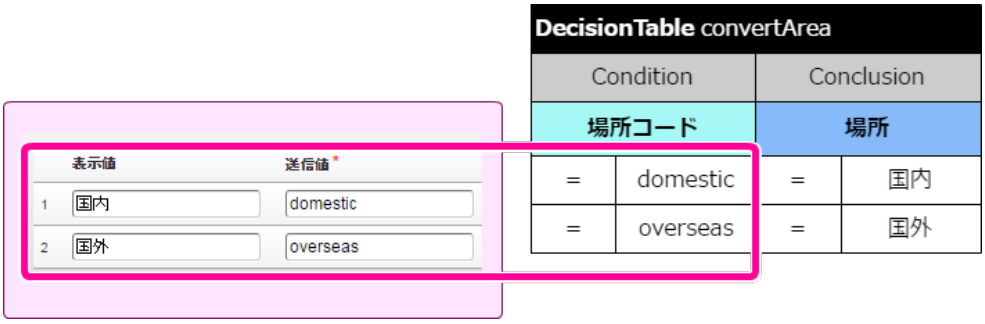
a. 場所（ラジオボタン）とアイテムのプロパティ設定

b. 出張区分（ラジオボタン）とアイテムのプロパティ設定

このまま OpenRules と IM-BIS を連携すると、 *DecisionTable* は表示値で条件が記載されていることに対し、画面からは送信値が受け渡されるため 、OpenRules で正しく評価することができません。
そのため、 OpenRules の定義内で表示値・送信値のマッピングを行う *DecisionTable* を設定し、事前に送信値を表示値に変換する処理を行うための *DecisionTable* を作成します。

以下の図のように項目単位で表示値・送信値をマッピングする *DecisionTable* をまとめていきます。

- 「場所」は、 *DecisionTable* 「convertArea」を利用して、 OpenRules 内で送信値と表示値の変換を行います。
  左側が IM-BIS のラジオボタンの値の設定、右側が OpenRules の変換 *DecisionTable* です。

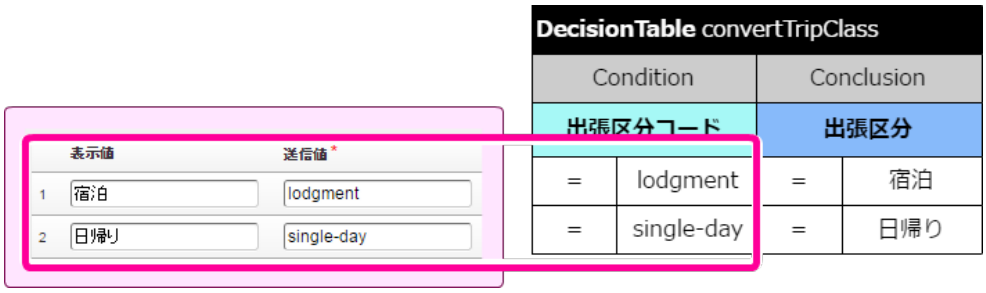


表示値		送信値
1	国内	domestic
2	国外	overseas

DecisionTable convertArea

Condition		Conclusion	
場所コード		場所	
=	domestic	=	国内
=	overseas	=	国外

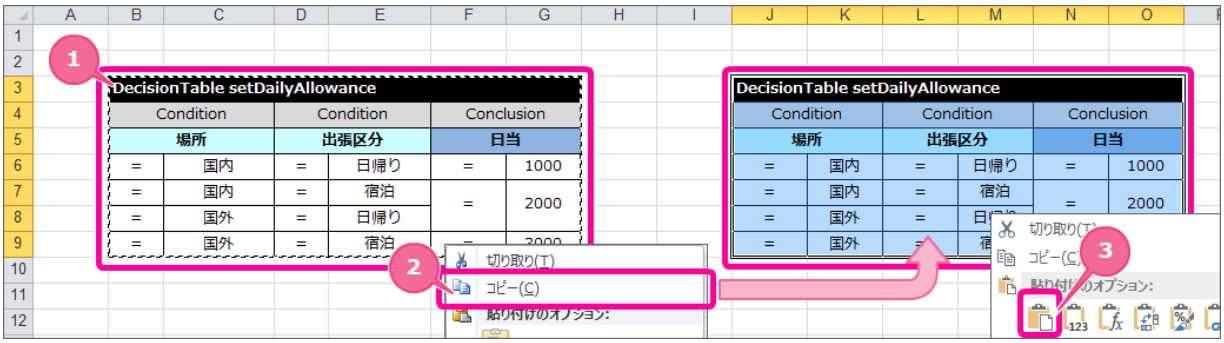
- 「出張区分」は、 *DecisionTable* 「convertTripClass」を利用して、 OpenRules 内で送信値と表示値の変換を行います。
  左側が IM-BIS のラジオボタンの値の設定、右側が OpenRules の変換 *DecisionTable* です。



場所のマッピング *DecisionTable* を作成する

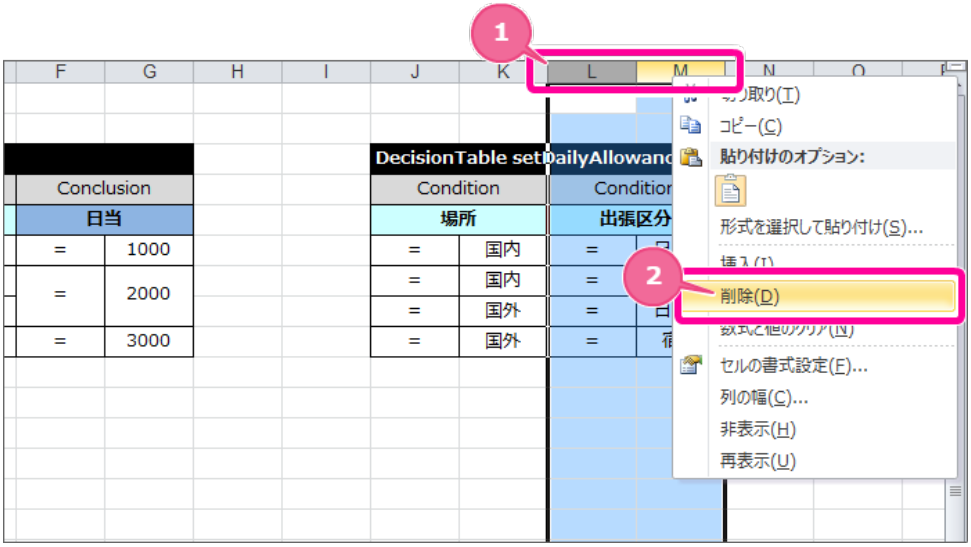
マッピングが必要な項目「場所」に対応するマッピングの *DecisionTable* を作成しましょう。

1. 編集したExcelファイルを開きます。
2. 「DecisionTable」シートを表示します。
3. 以下の手順で作成済みの *DecisionTable* をコピーして新しい *DecisionTable* を作成してください。



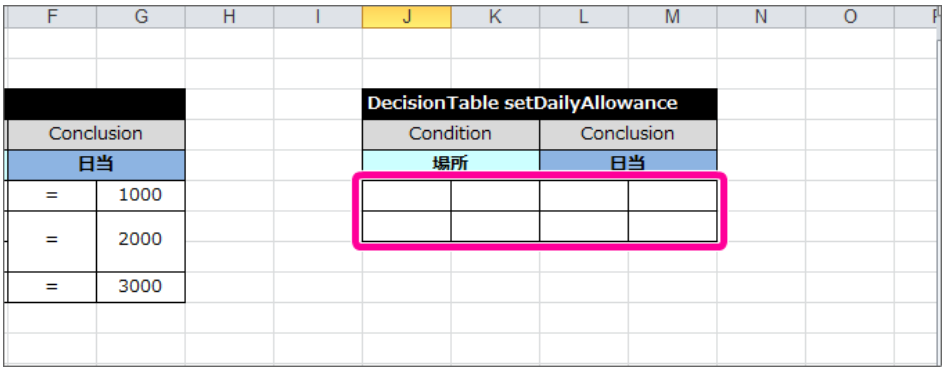
1. 作成したテーブルの範囲を選択してください。
2. 右クリック後、「コピー」をクリックしてください。
3. コピーしたテーブルから1セル以上行・列を空けたセルで右クリックし、「貼り付け」を選択してください。

4. コピーで作成した *DecisionTable* に対し、以下の手順で不要な列を削除してください。



1. 不要な列（図中ではL列とM列）をドラッグして選択状態にし、右クリックしてください。
2. メニューから「削除」をクリックしてください。

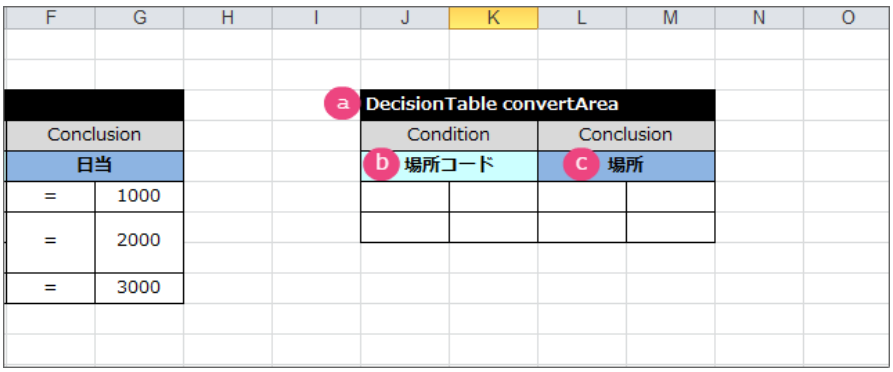
5. コピーした *DecisionTable* は、ラジオボタンの値の個数にあわせて、明細が2行になるように不要な行を削除してください。



Conclusion	
日当	
=	1000
=	2000
=	3000

DecisionTable setDailyAllowance			
Condition		Conclusion	
場所		日当	

6. *DecisionTable* のテーブル名と項目名を以下の通りに入力してください。



Conclusion	
日当	
=	1000
=	2000
=	3000

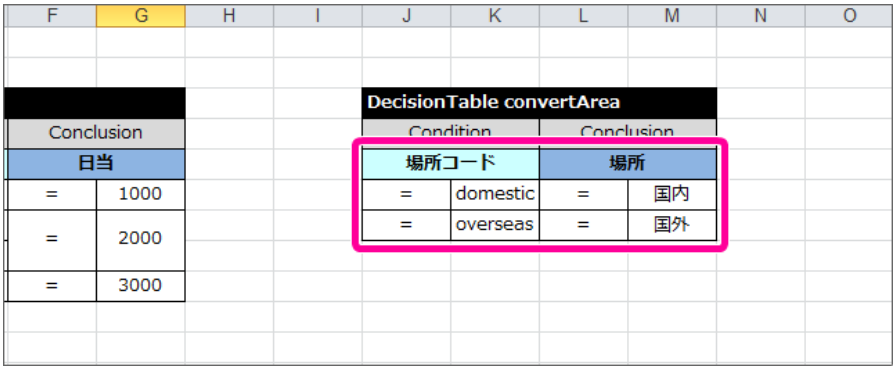
a DecisionTable convertArea			
Condition		Conclusion	
b 場所コード		c 場所	

a. テーブル名
   キーワード *DecisionTable* から半角スペースを空けて「convertArea」と入力してください。

b. 条件の項目の論理名
   「場所コード」と入力してください。

c. 結果の項目の論理名
   「場所」と入力してください。

7. *DecisionTable* の明細には、以下の通りに「場所」のラジオボタンの表示値・送信値を入力してください。



Conclusion	
日当	
=	1000
=	2000
=	3000

DecisionTable convertArea			
Condition		Conclusion	
場所コード		場所	
=	domestic	=	国内
=	overseas	=	国外

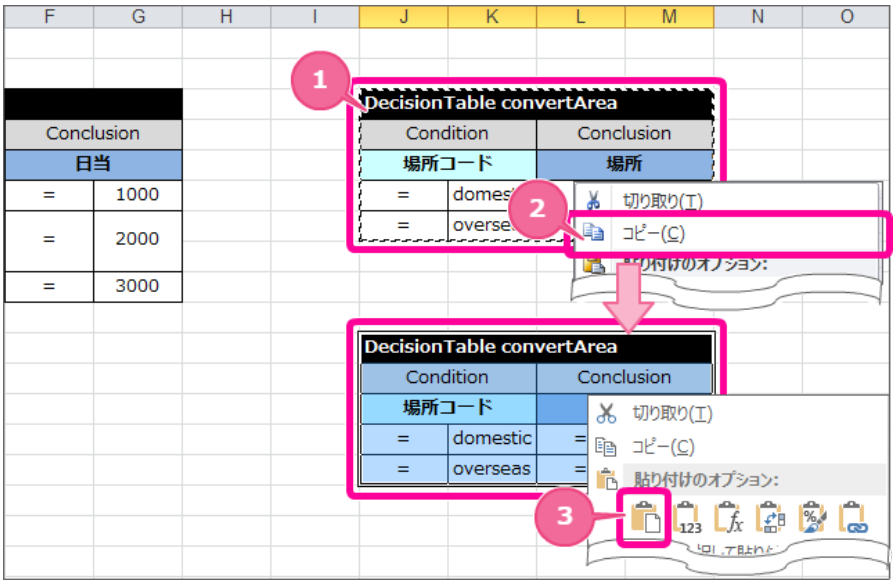
場所コード	場所
domestic	国内
overseas	国外

8. これで場所のマッピング *DecisionTable* が作成できましたので、引き続き出張区分を作成していきましょう。

出張区分のマッピング *DecisionTable* を作成する

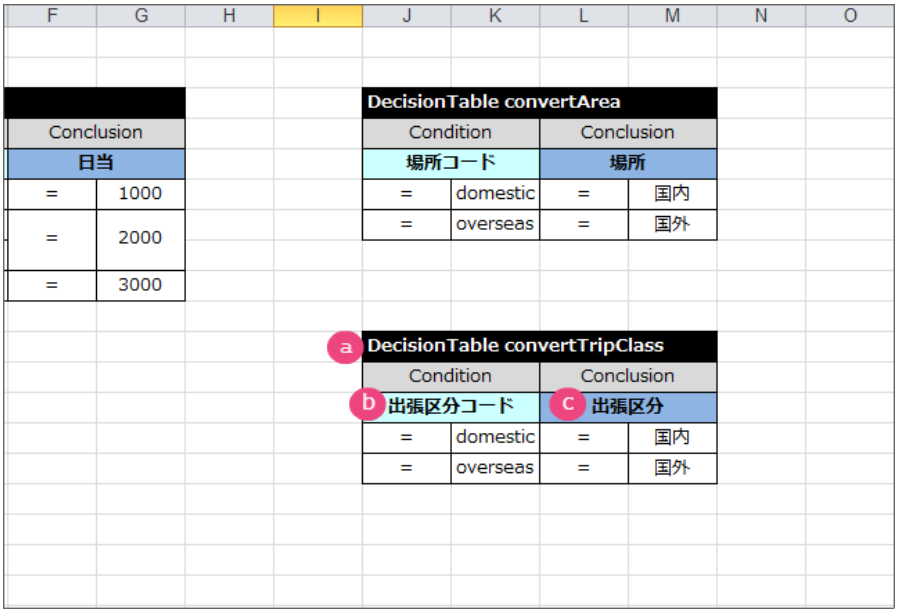
同じようにしてマッピングが必要な項目「出張区分」に対応するマッピングの *DecisionTable* を作成しましょう。

1. 以下の手順で作成済みの *DecisionTable* 「convertArea」をコピーして新しい *DecisionTable* を作成してください。



1. 作成したテーブル「convertArea」の範囲を選択してください。
2. 右クリック後、「コピー」をクリックしてください。
3. コピーしたテーブルから1セル以上行・列を空けたセルで右クリックし、「貼り付け」を選択してください。

2. *DecisionTable* のテーブル名と項目名を以下の通りに入力してください。

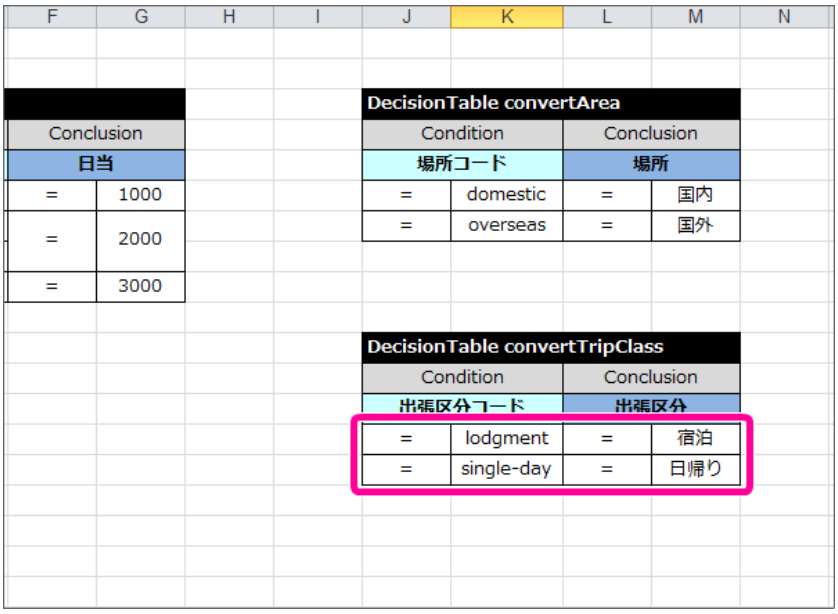


a. テーブル名
　キーワード *DecisionTable* から半角スペースを空けて「convertTripClass」と入力してください。

b. 条件の項目の論理名
　「出張区分コード」と入力してください。

c. 結果の項目の論理名
　「出張区分」と入力してください。

3. *DecisionTable* の明細には、以下の通りに「出張区分」のラジオボタンの表示値・送信値を入力してください。



Conclusion	
日当	
=	1000
=	2000
=	3000

DecisionTable convertArea			
Condition		Conclusion	
場所コード		場所	
=	domestic	=	国内
=	overseas	=	国外

DecisionTable convertTripClass			
Condition		Conclusion	
出張区分コード		出張区分	
=	lodgment	=	宿泊
=	single-day	=	日帰り

出張区分コード	出張区分
lodgment	宿泊
single-day	日帰り

4. これで、表示値と送信値を OpenRules でマッピングする表ができました。

OpenRules で実行する DecisionTable の順序をコントロールする

ここまでの手順で、場所区分・出張区分のマッピング、日当の計算で３つの *DecisionTable* を作成しました。
次の手順では、マッピングの *DecisionTable* 、日当の計算の *DecisionTable* の順に実行されるようにコントロールするための *Decision* を設定します。

以下の図のように *Decision* をまとめていきます。

Decision sampleDailyAllowance			
Condition		ActionPrint	ActionExecute
場所		**Decisions**	**Execute Decision Tables**
Is One Of	国内,国外	場所の変換	convertArea
		出張区分の変換	convertTripClass
		日当の判定	calculateDailyAllowance
		結果の取得	:= decision.put("ResponseObject", responseObj)

a

b

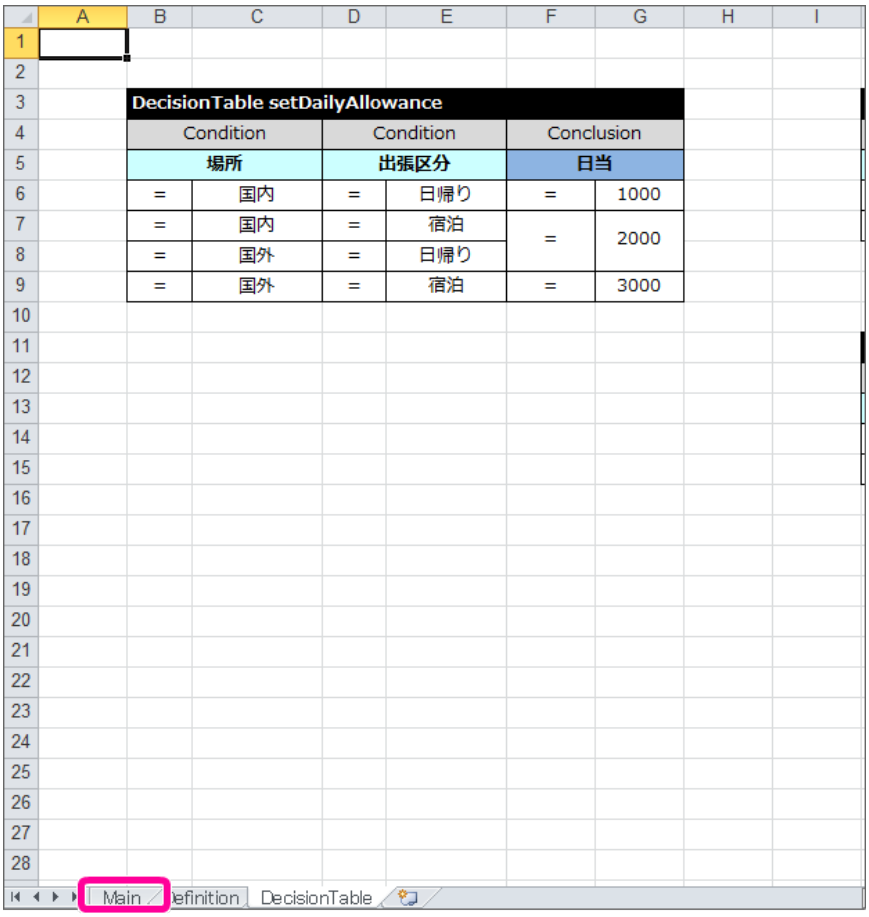
a. *DecisionTable*
1つの *Decision* で実行対象の *DecisionTable* を複数記述できます。
Decision では上から順に記述された *DecisionTable* の評価や処理を実行します。

b. *Condition*
今回のハンズオンでは扱いませんが、 *Decision* で *Condition* を記載した場合には条件に合致したときだけ特定の *DecisionTable* の評価を実行することが設定できます。

Decision を作成する

ルールのコントロールを行う *Decision* を作成しましょう。

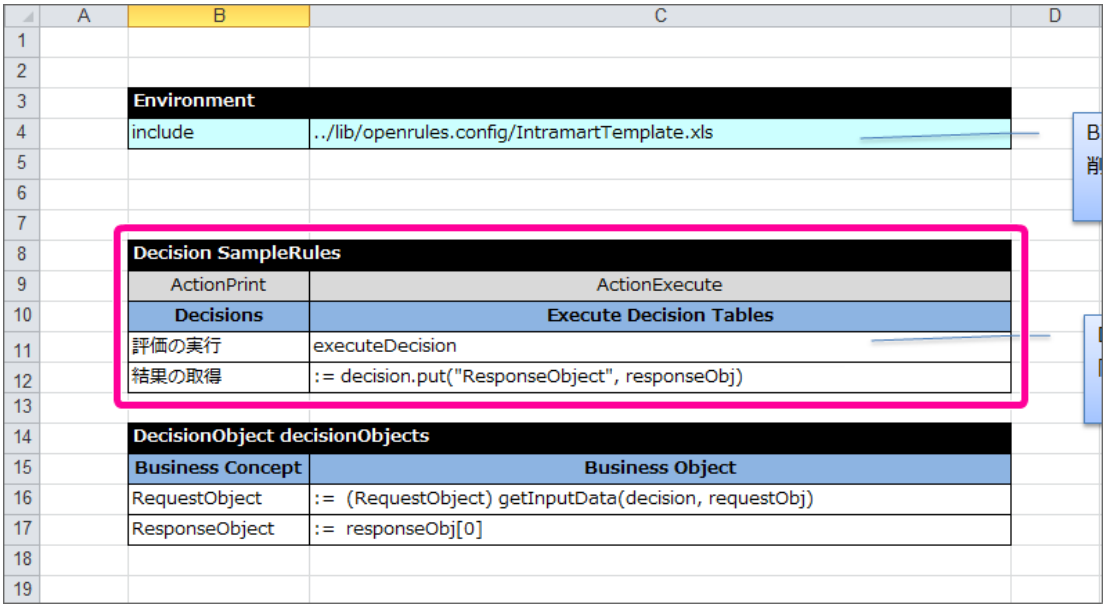
1. 編集したExcelファイルを開きます。
2. 「Main」タブをクリックして「Main」シートを表示してください。



DecisionTable setDailyAllowance					
Condition		Condition		Conclusion	
場所		出張区分		日当	
=	国内	=	日帰り	=	1000
=	国内	=	宿泊	=	2000
=	国外	=	日帰り		
=	国外	=	宿泊	=	3000

Main | efinition | DecisionTable

3. *Decision* を表示してください。
（*DecisionTable* を作成した時と同様に、シート上のコメントは不要であれば削除してください。）



Environment	
include	../lib/openrules.config/IntramartTemplate.xls

Decision SampleRules	
ActionPrint	ActionExecute
Decisions	**Execute Decision Tables**
評価の実行	executeDecision
結果の取得	:= decision.put("ResponseObject", responseObj)

DecisionObject decisionObjects	
Business Concept	**Business Object**
RequestObject	:= (RequestObject) getInputData(decision, requestObj)
ResponseObject	:= responseObj[0]

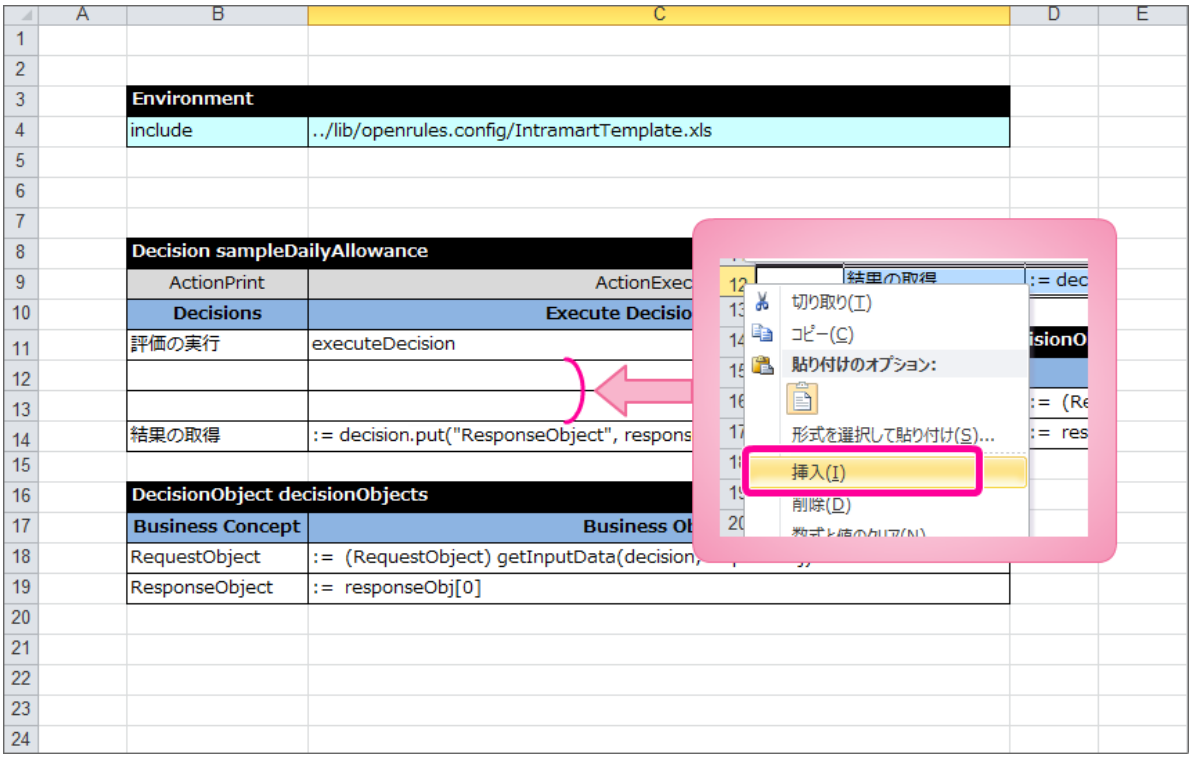
4. キーワード *Decision* から半角スペースを空けて「sampleDailyAllowance」と入力してください。

Environment	
include	../lib/openrules.config/IntramartTemplate.xls

Decision sampleDailyAllowance	
ActionPrint	ActionExecute
Decisions	**Execute Decision Tables**
評価の実行	executeDecision
結果の取得	:= decision.put("ResponseObject", responseObj)

DecisionObject decisionObjects	
Business Concept	**Business Object**
RequestObject	:= (RequestObject) getInputData(decision, requestObj)
ResponseObject	:= responseObj[0]

5. 今回のハンズオンでは、1つの *Decision* で3つの *DecisionTable* を実行します。
   3つ分の *DecisionTable* を記述できるように間に2行追加してください。



6. 最初に場所の *DecisionTable* を実行するための設定を行います。
   1行目の内容を以下の通りに変更してください。

Environment		
include	../lib/openrules.config/IntramartTemplate.xls	
Decision sampleDailyAllowance		
ActionPrint	ActionExecute	
Decisions	**Execute Decision Tables**	
場所の変換	convertArea	
結果の取得	:= decision.put("ResponseObject", responseObj)	
DecisionObject decisionObjects		
Business Concept	**Business Object**	
RequestObject	:= (RequestObject) getInputData(decision, requestObj)	
ResponseObject	:= responseObj[0]	

ActionPrint	ActionExecute
場所の変換	convertArea

7. 次に出張区分の *DecisionTable* を実行するための設定を行います。
   以下の通り2行目に入力してください。

Environment		
include	../lib/openrules.config/IntramartTemplate.xls	
Decision sampleDailyAllowance		
ActionPrint	ActionExecute	
Decisions	**Execute Decision Tables**	
場所の変換	convertArea	
出張区分の変換	convertTripClass	
結果の取得	:= decision.put("ResponseObject", responseObj)	
DecisionObject decisionObjects		
Business Concept	**Business Object**	
RequestObject	:= (RequestObject) getInputData(decision, requestObj)	
ResponseObject	:= responseObj[0]	

ActionPrint	ActionExecute
出張区分の変換	convertTripClass

8. 最後に日当計算の *DecisionTable* を実行するための設定を行います。
   以下の通り3行目に入力してください。

	A	B	C	D	E
1					
2					
3		**Environment**			
4		include	../lib/openrules.config/IntramartTemplate.xls		
5					
6					
7					
8		**Decision sampleDailyAllowance**			
9		ActionPrint	ActionExecute		
10		**Decisions**	**Execute Decision Tables**		
11		場所の変換	convertArea		
12		出張区分の変換	convertTripClass		
13		日当の計算	setDailyAllowance		
14		結果の取得	:= decision.put("ResponseObject", responseObj)		
15					
16		**DecisionObject decisionObjects**			
17		**Business Concept**	**Business Object**		
18		RequestObject	:= (RequestObject) getInputData(decision, requestObj)		
19		ResponseObject	:= responseObj[0]		
20					
21					
22					
23					

ActionPrint	ActionExecute
日当の計算	setDailyAllowance

9. ここまでで、 *DecisionTable* の実行順を *Decision* で設定することができました。
   Excelファイルを保存し、次の手順に進みましょう。

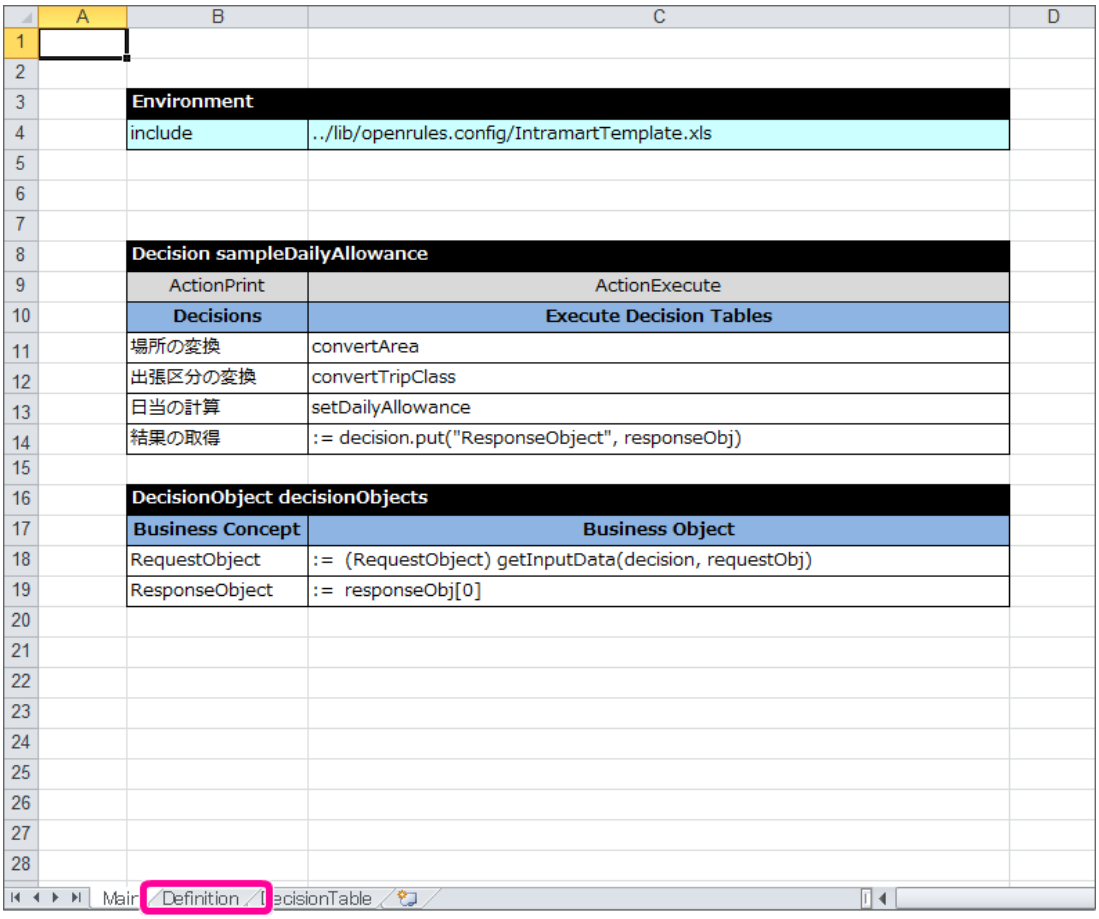
実行に必要な定義を設定する

ルールの中心となる *Decision* 、 *DecisionTable* が完成しましたので、実行に必要なその他の定義を設定していきましょう。

Glossary に利用する項目を定義する

DecisionTable で使っている項目を確認しながら、 *Glossary* を定義していきましょう。

1. 編集中のExcelファイルの「Definition」タブをクリックして「Definition」シートを表示してください。

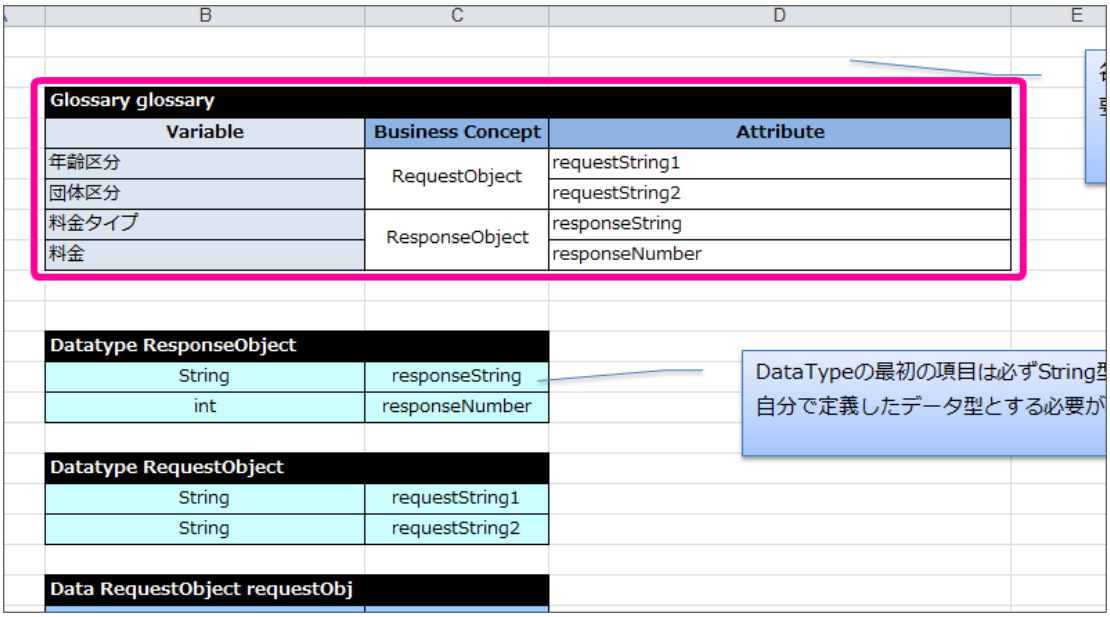


Environment	
include	../lib/openrules.config/IntramartTemplate.xls

Decision sampleDailyAllowance	
ActionPrint	ActionExecute
Decisions	**Execute Decision Tables**
場所の変換	convertArea
出張区分の変換	convertTripClass
日当の計算	setDailyAllowance
結果の取得	:= decision.put("ResponseObject", responseObj)

DecisionObject decisionObjects	
Business Concept	**Business Object**
RequestObject	:= (RequestObject) getInputData(decision, requestObj)
ResponseObject	:= responseObj[0]

2. このシートの *Glossary* を表示してください。
（*DecisionTable* を作成した時と同様に、シート上のコメントは不要であれば削除してください。）



Glossary glossary		
Variable	**Business Concept**	**Attribute**
年齢区分	RequestObject	requestString1
団体区分		requestString2
料金タイプ	ResponseObject	responseString
料金		responseNumber

Datatype ResponseObject	
String	responseString
int	responseNumber

Datatype RequestObject	
String	requestString1
String	requestString2

Data RequestObject requestObj

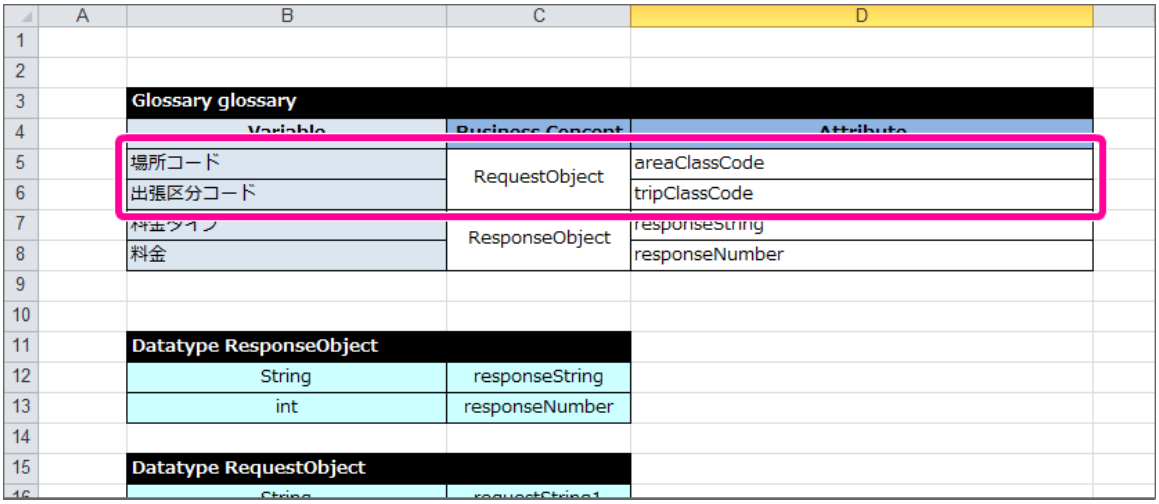
DataTypeの最初の項目は必ずString型
自分で定義したデータ型とする必要が

3. *Glossary* を定義するためには、 *DecisionTable* で利用している項目を確認する必要があります。
今回のハンズオンの *DecisionTable* で扱っている項目には、以下の5つがあります。
   - 場所コード
   - 出張区分コード
   - 場所
   - 出張区分
   - 日当

   このうち、画面（フォーム）から OpenRules に受け渡される項目（場所コード、出張区分コード）を「RequestObject」の項目として定義してください。

Variable	BusinessConcept	Attribute
場所コード	RequestObject	areaClassCode

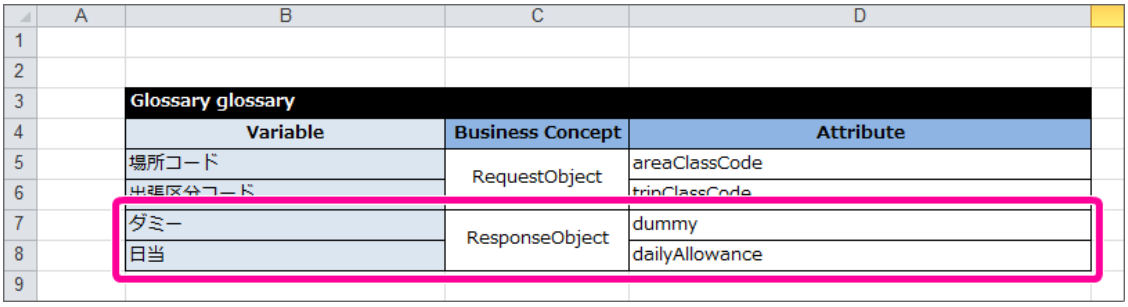
Variable	BusinessConcept	Attribute
出張区分コード		tripClassCode

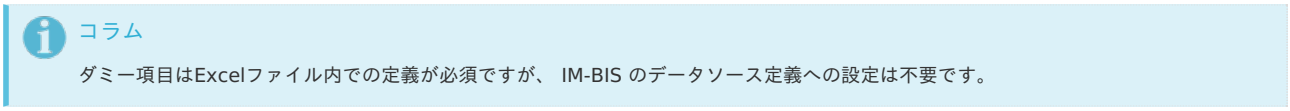


4. 続いて、 OpenRules から画面（フォーム）に返却する項目を「ResponseObject」として定義してください。
ResponseObjectに含まれる項目は数値型の「日当（dailyAllowance）」のみですが、このまま定義すると OpenRules の制約（*Datatype* の「技術的な制約事項」参照 ）で正しく実行できません。
そのため、ダミー項目を定義した後に、日当を定義してください。

Glossary には、以下のように定義します。

Variable	BusinessConcept	Attribute
ダミー	ResponseObject	dummy
日当		dailyAllowance





5. *DecisionTable* で扱っている項目のうち、RequestObjectとResponseObjectのどちらにも属さない項目は内部項目のオブジェクト（Internal）として定義します。
この時点で定義していない「場所区分」「出張区分」について、以下のように定義してください。

Variable	BusinessConcept	Attribute
場所区分	Internal	area
出張区分		tripClass

Glossary glossary		
Variable	**Business Concept**	**Attribute**
場所コード	RequestObject	areaClassCode
出張区分コード		tripClassCode
ダミー	ResponseObject	dummy
日当		dailyAllowance
場所	Internal	area
出張区分		tripClass

Glossary に基づいて *Datatype* と *Data/Variable* を定義する

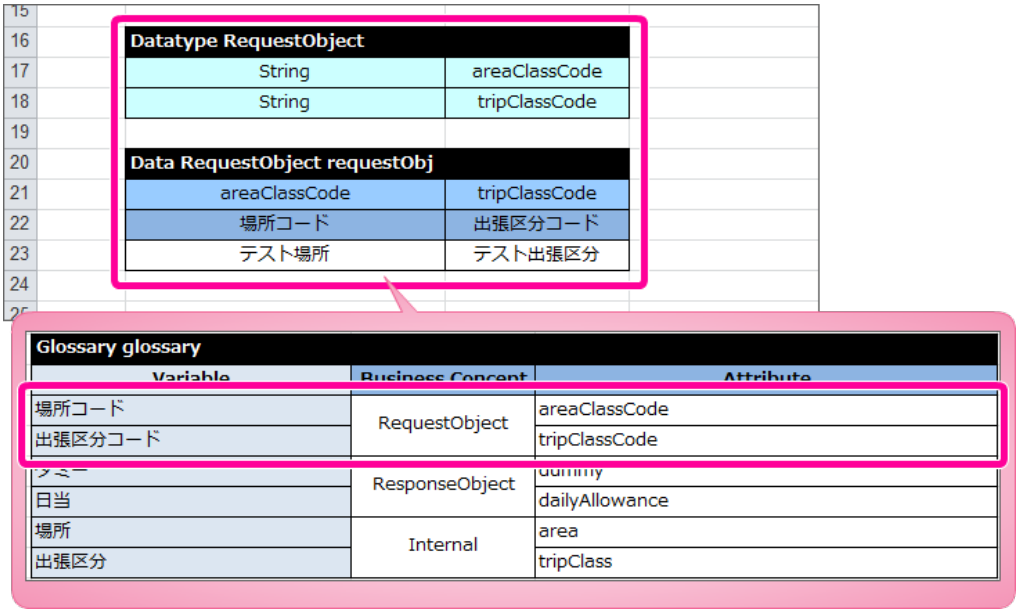
Glossary を利用して *Datatype* と *Data/Variable* を定義していきましょう。

1. *Glossary* が定義できましたので、これに基づいて *Datatype* と *Data/Variable* を定義していきます。
   同じ「Definition」シートのRequestObjectの *Datatype* と *Data/Variable* を以下のように定義してください。
   - *Datatype*

Datatype RequestObject	
String	areaClassCode
String	tripClassCode

   - *Data/Variable*

Data RequestObject requestObj	
areaClassCode	tripClassCode
場所コード	出張区分コード
テスト場所	テスト出張区分



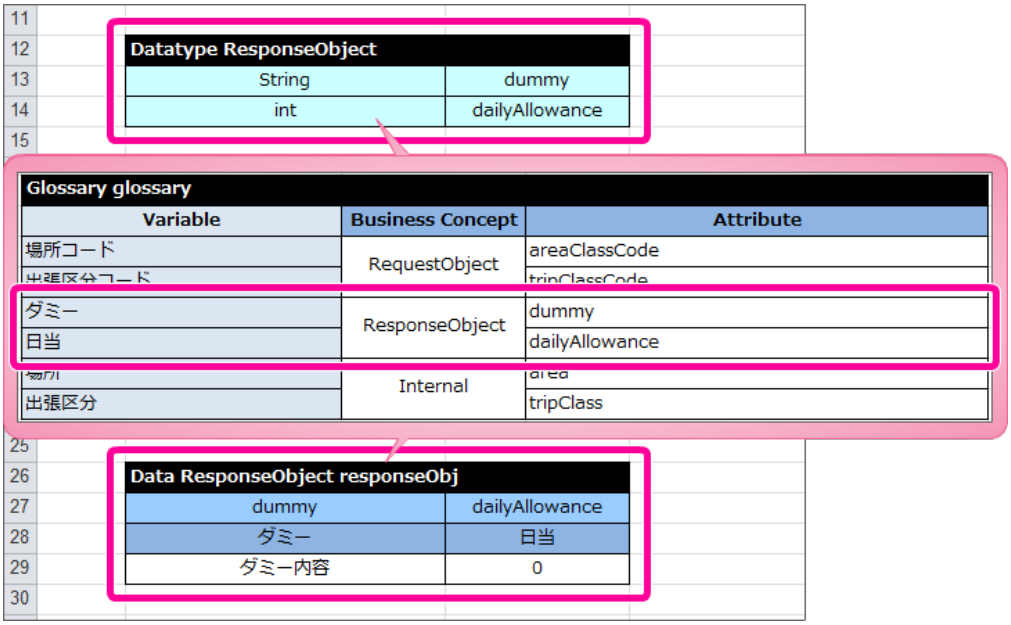
2. ResponseObjectの *Datatype* と *Data/Variable* を以下のように定義してください。
   - *Datatype*

Datatype ResponseObject	
String	dummy
int	dailyAllowance

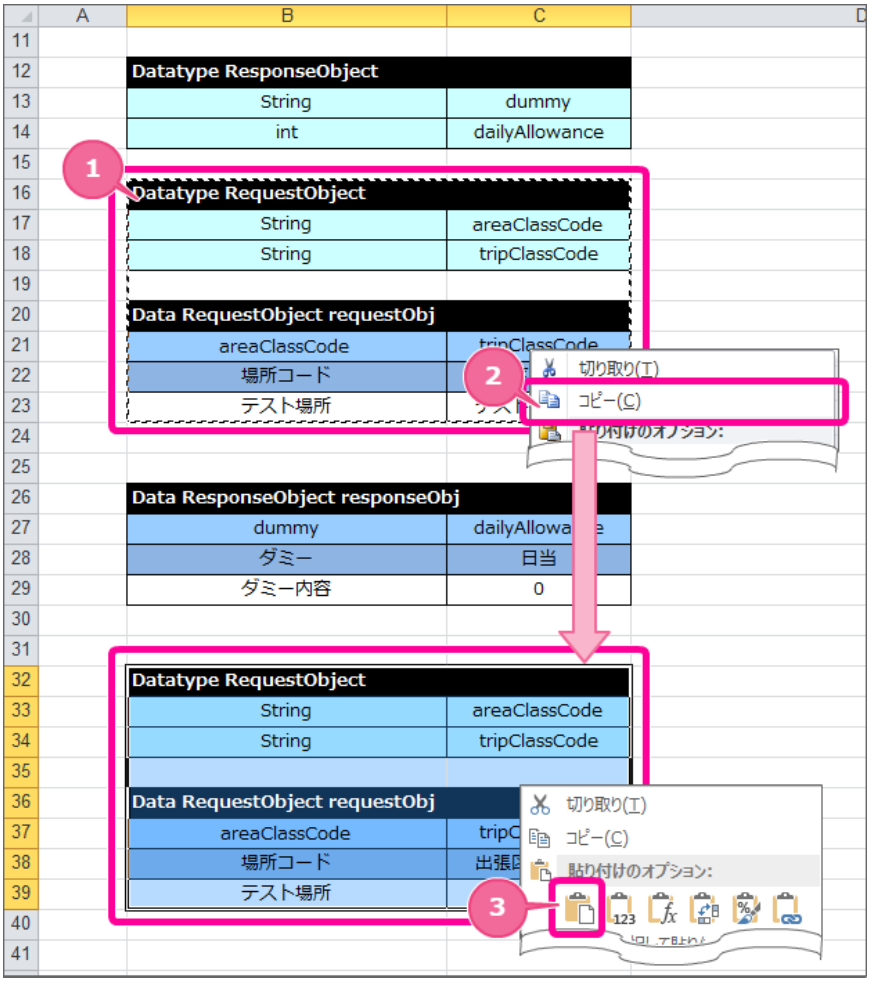
   - *Data/Variable*

Data ResponseObject responseObj	
dummy	dailyAllowance

Data ResponseObject responseObj	
ダミー	日当
ダミー内容	0



3. 同様にInternalの *Datatype* と *Data/Variable* を定義するために、先に作成したRequestObjectの定義を以下の手順でコピーしてください。



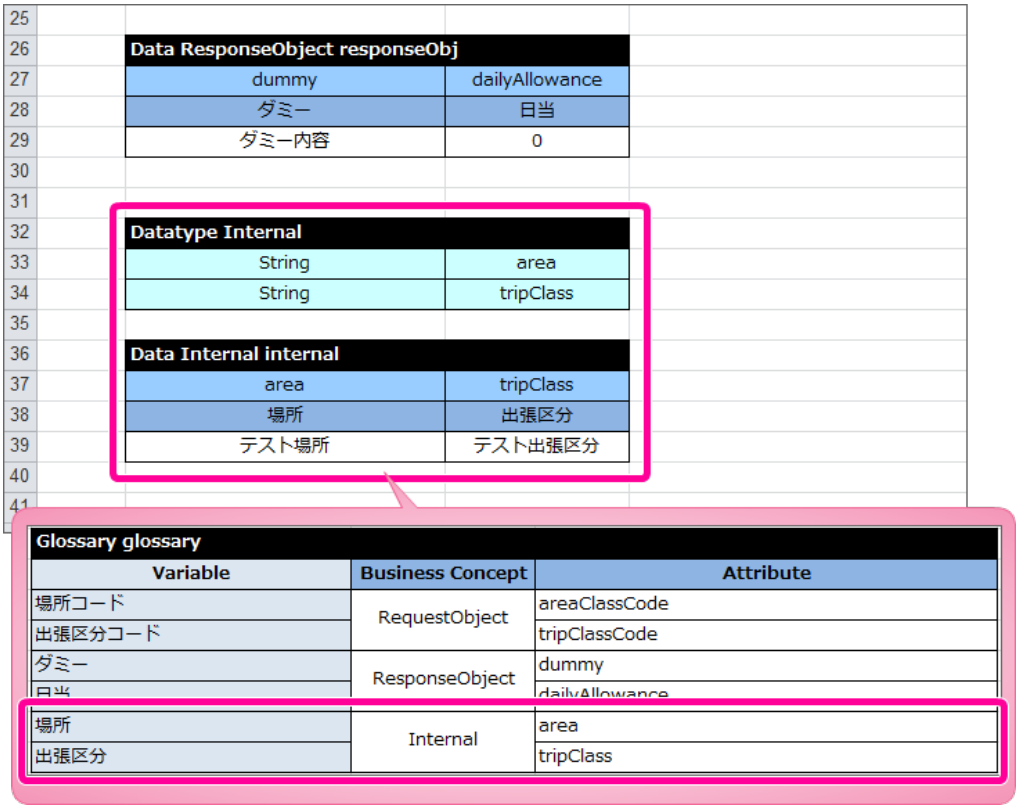
1. RequestObjectの *Datatype* と *Data/Variable* の範囲を選択してください。
2. 右クリック後、「コピー」をクリックしてください。
3. コピーしたテーブルから1セル以上行・列を空けたセルで右クリックし、「貼り付け」を選択してください。

4. コピーした定義を以下のように変更してください。
   - *Datatype*

Datatype Internal	
String	area
String	tripClass

- *Data/Variable*

Data Internal internal	
area	tripClass
場所	出張区分
テスト場所	テスト出張区分

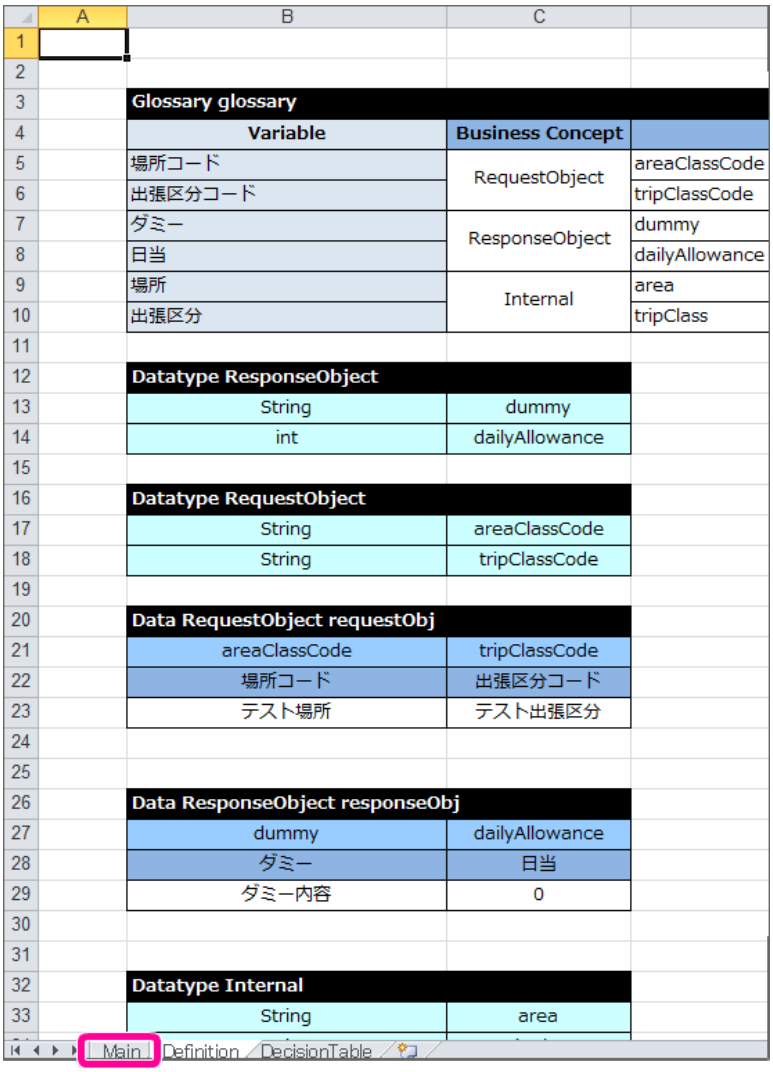


5. これで *Datatype* と *Data/Variable* が定義できましたので、一度保存します。

DecisionObject を定義してルールを完成させる

最後に *DecisionObject* を定義して、Excelのルール定義ファイルを完成させましょう。

1. 「Main」タブをクリックして「Main」シートを表示してください。



Glossary glossary		
Variable	**Business Concept**	
場所コード	RequestObject	areaClassCode
出張区分コード		tripClassCode
ダミー	ResponseObject	dummy
日当		dailyAllowance
場所	Internal	area
出張区分		tripClass

Datatype ResponseObject	
String	dummy
int	dailyAllowance

Datatype RequestObject	
String	areaClassCode
String	tripClassCode

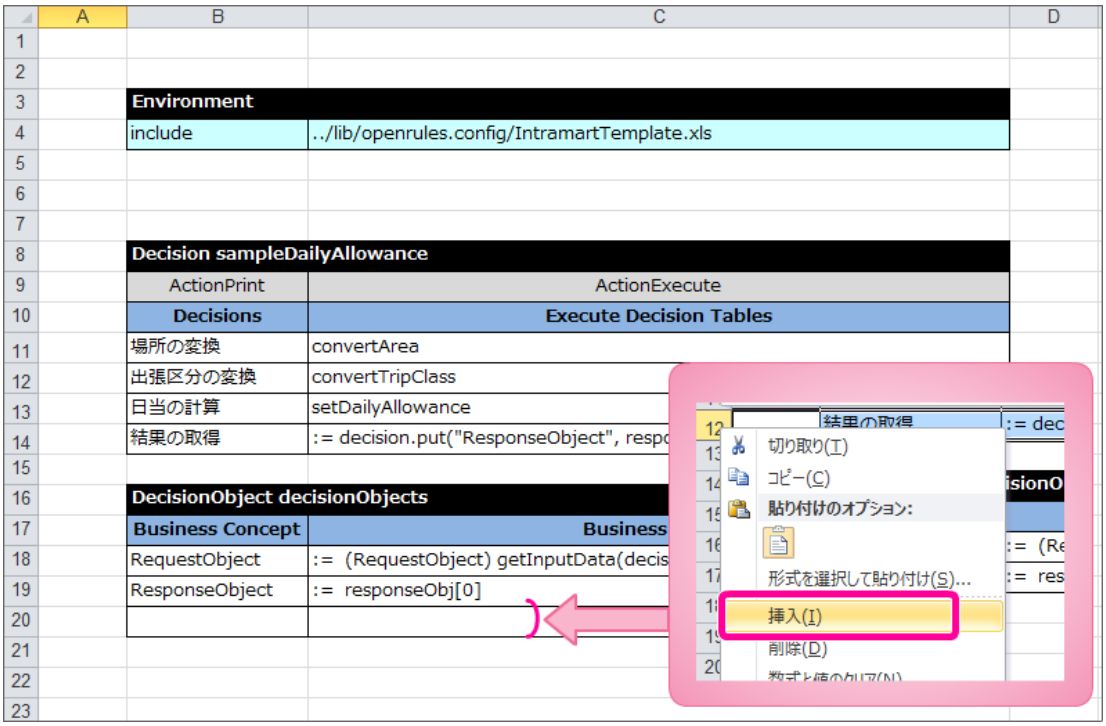
Data RequestObject requestObj	
areaClassCode	tripClassCode
場所コード	出張区分コード
テスト場所	テスト出張区分

Data ResponseObject responseObj	
dummy	dailyAllowance
ダミー	日当
ダミー内容	0

Datatype Internal	
String	area

Main | Definition | DecisionTable

2. 「Main」シートで *DecisionObject* を設定していきます。
RequestObject、ResponseObjectのオブジェクトについては、テンプレートと同じ設定となっているため、設定は不要です。
RequestObject、ResponseObjectの下にInternalの設定用に1行追加してください。



Environment	
include	../lib/openrules.config/IntramartTemplate.xls

Decision sampleDailyAllowance	
ActionPrint	ActionExecute
Decisions	**Execute Decision Tables**
場所の変換	convertArea
出張区分の変換	convertTripClass
日当の計算	setDailyAllowance
結果の取得	:= decision.put("ResponseObject", respo

DecisionObject decisionObjects	
Business Concept	**Business**
RequestObject	:= (RequestObject) getInputData(decis
ResponseObject	:= responseObj[0]

3. 追加した行には、以下のように設定してください。

DecisionObject decisionObjects	
Business Concept	Business Object

DecisionObject decisionObjects	
Internal	:= internal[0]

	A	B	C	D	E
1					
2					
3		**Environment**			
4		include	../lib/openrules.config/IntramartTemplate.xls		
5					
6					
7					
8		**Decision sampleDailyAllowance**			
9		ActionPrint	ActionExecute		
10		**Decisions**	**Execute Decision Tables**		
11		場所の変換	convertArea		
12		出張区分の変換	convertTripClass		
13		日当の計算	setDailyAllowance		
14		結果の取得	:= decision.put("ResponseObject", responseObj)		
15					
16		**DecisionObject decisionObjects**			
17		**Business Concept**	**Business Object**		
18		RequestObject	:= (RequestObject) getInputData(decision, requestObj)		
19		ResponseObject	:= responseObj[0]		
20		Internal	:= internal[0]		
21					
22					

> **コラム**
> このハンズオンのように、処理用のオブジェクト等を定義した場合には、 *DecisionObject* でインスタンス化の処理を設定することができます。

4. これで、 OpenRules で日当を計算するExcelのルール定義ファイルの設定が完了しましたので、ファイルを保存します。
   IM-BIS の画面（フォーム）と連携するための設定を行っていきましょう。

IM-BIS の画面（フォーム）のイベントと OpenRules を連携する

先の手順で作成したExcelのルール定義ファイルを IM-BIS と連携するためのフローの作成を進めていきましょう。

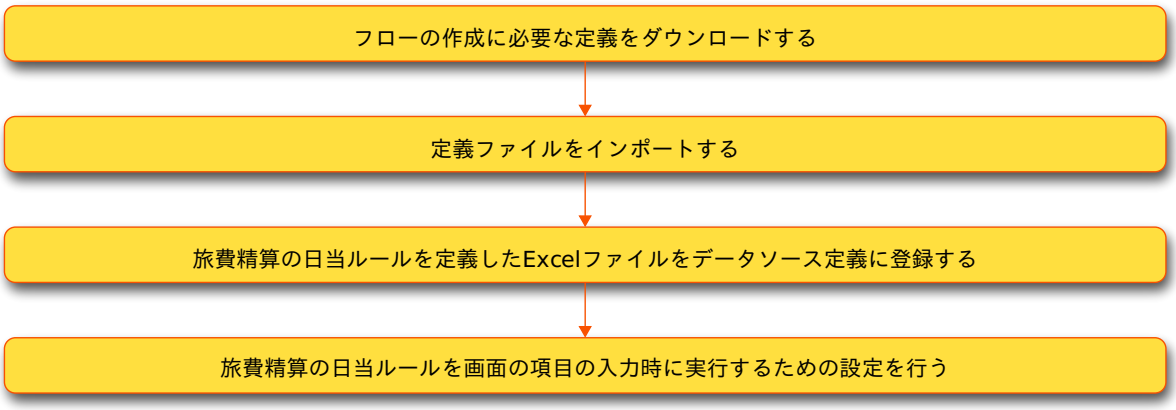
ルールと連携したフローを作成する手順

- OpenRules と IM-BIS を連携するための手順
- フローの作成に必要な定義をダウンロードする
- ハンズオンの定義ファイルをインポートする
- 旅費精算の日当ルールを定義したExcelファイルをデータソース定義に登録する
- 旅費精算の日当ルールを画面の項目の入力時に実行するための設定を行う

OpenRules と IM-BIS を連携するための手順

この手順では、作成したExcelのルール定義ファイルをデータソース定義に登録し、 IM-BIS のイベント設定でルールの実行を設定するまでの手順を確認していきます。

- 旅費精算の日当ルールを IM-BIS と連携する手順



フローの作成に必要な定義をダウンロードする

ハンズオンで作成するフローのベースとなる各種定義ファイルをインポートします。
最初に下記のリンクからファイルをダウンロードしてください。
「 IM-Workflow 定義」のみダウンロード後に解凍してください。

- IM-Workflow 定義
  imw_travel_expenses.zip
- BIS定義
  bis_travel_expenses.zip
- Formaアプリケーション定義
  forma_travel_expenses.zip

ハンズオンの定義ファイルをインポートする

先の手順でダウンロードしたファイルを「 *各種定義ファイルのインポートの手順* 」に従ってインポートしてください。

旅費精算の日当ルールを定義したExcelファイルをデータソース定義に登録する

IM-BIS と連携するために、 OpenRules のルールを定義したExcelファイルをデータソース定義に登録していきましょう。

データソース定義の基本情報を登録する

データソース定義の基本情報を登録しましょう。

1. サイトマップの「 IM-BIS 」から「データソース定義」をクリックしてください。

2. 「登録」をクリックしてください。

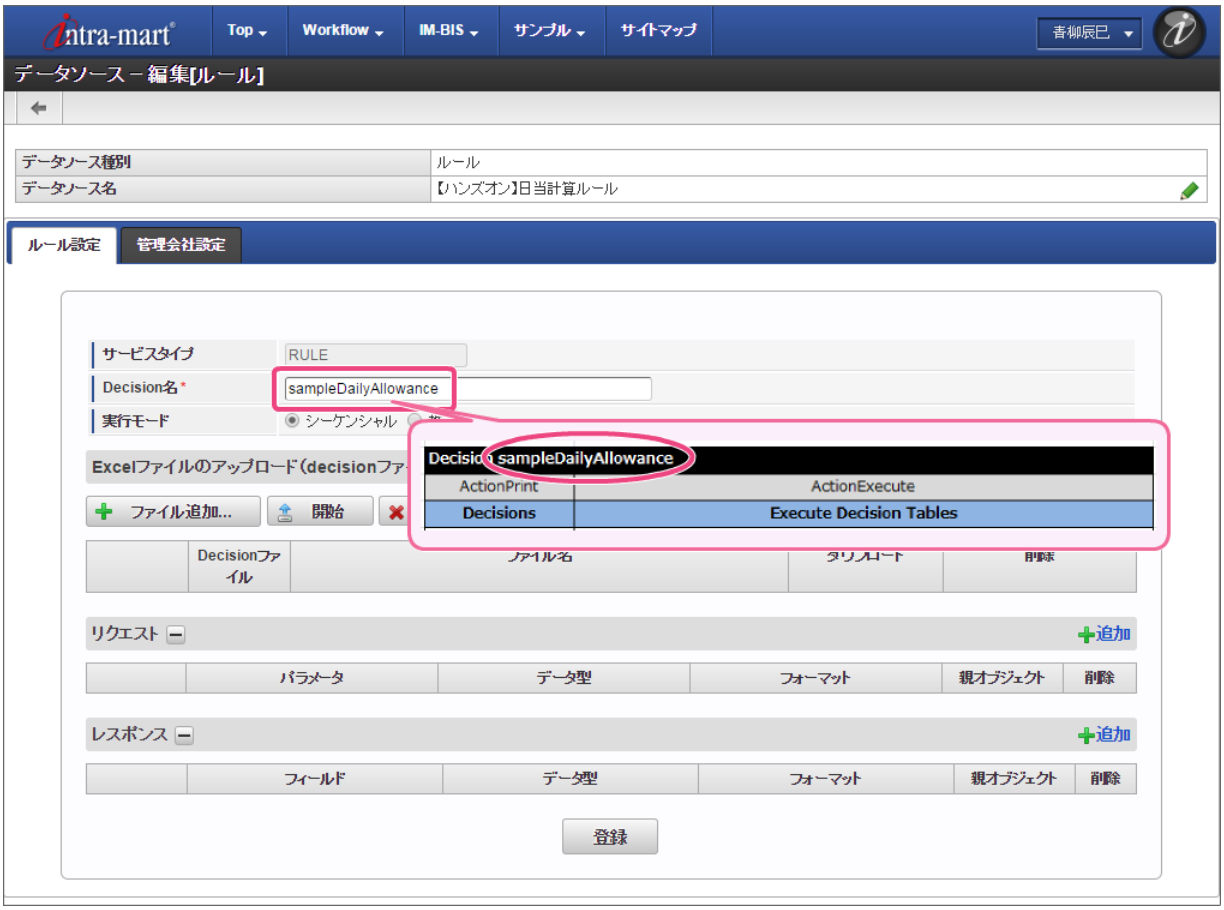
3. 以下の通りに入力後、「登録」をクリックしてください。

1. データソース種別
   「ルール」を選択してください。
2. データソース名
   「【ハンズオン】日当計算ルール」と入力してください。

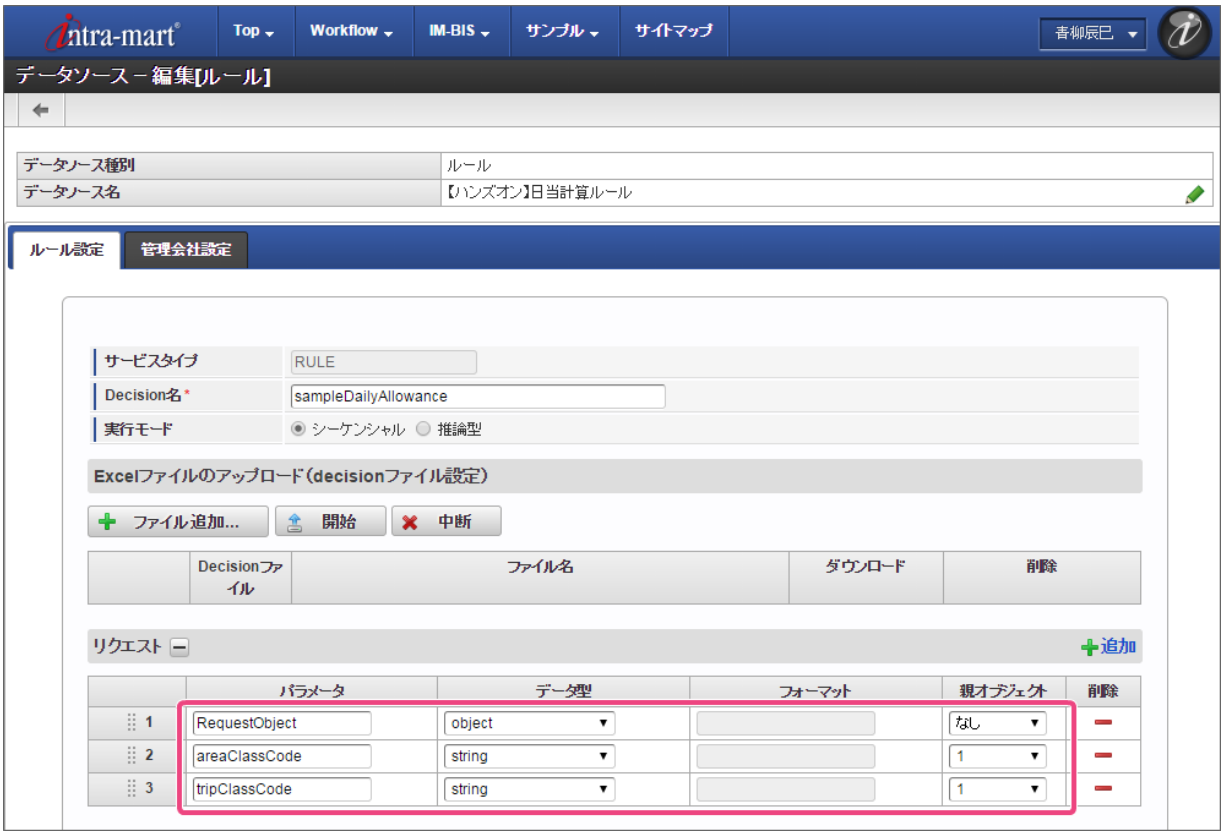
OpenRules の詳細情報を登録する

データソース定義に OpenRules のファイルやパラメータを設定しましょう。

1. 「Decision名」には、Excelファイルの *Decision* の名前「sampleDailyAllowance」を入力してください。



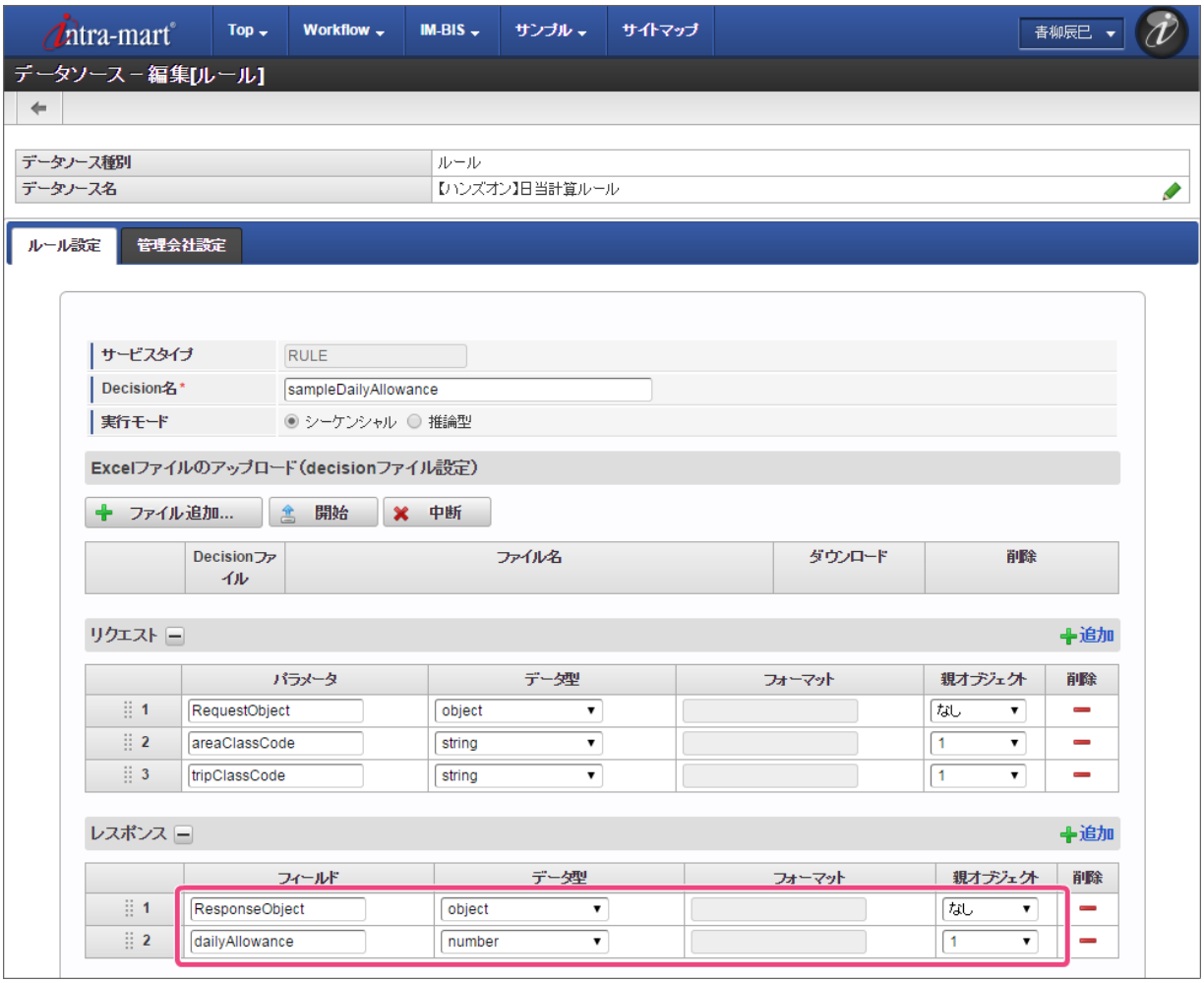
2. 「リクエスト」には、 *Glossary* で定義した「RequestObject」のオブジェクトと項目（物理名）を登録します。
入力欄を追加し、以下の通りに設定してください。



パラメータ	データ型	親オブジェクト
RequestObject	object	なし
areaClassCode	string	1
tripClassCode	string	1

3. 同様に「レスポンス」には、「ResponseObject」のオブジェクトと項目（物理名）を登録します。

入力欄を追加し、以下の通りに設定してください。



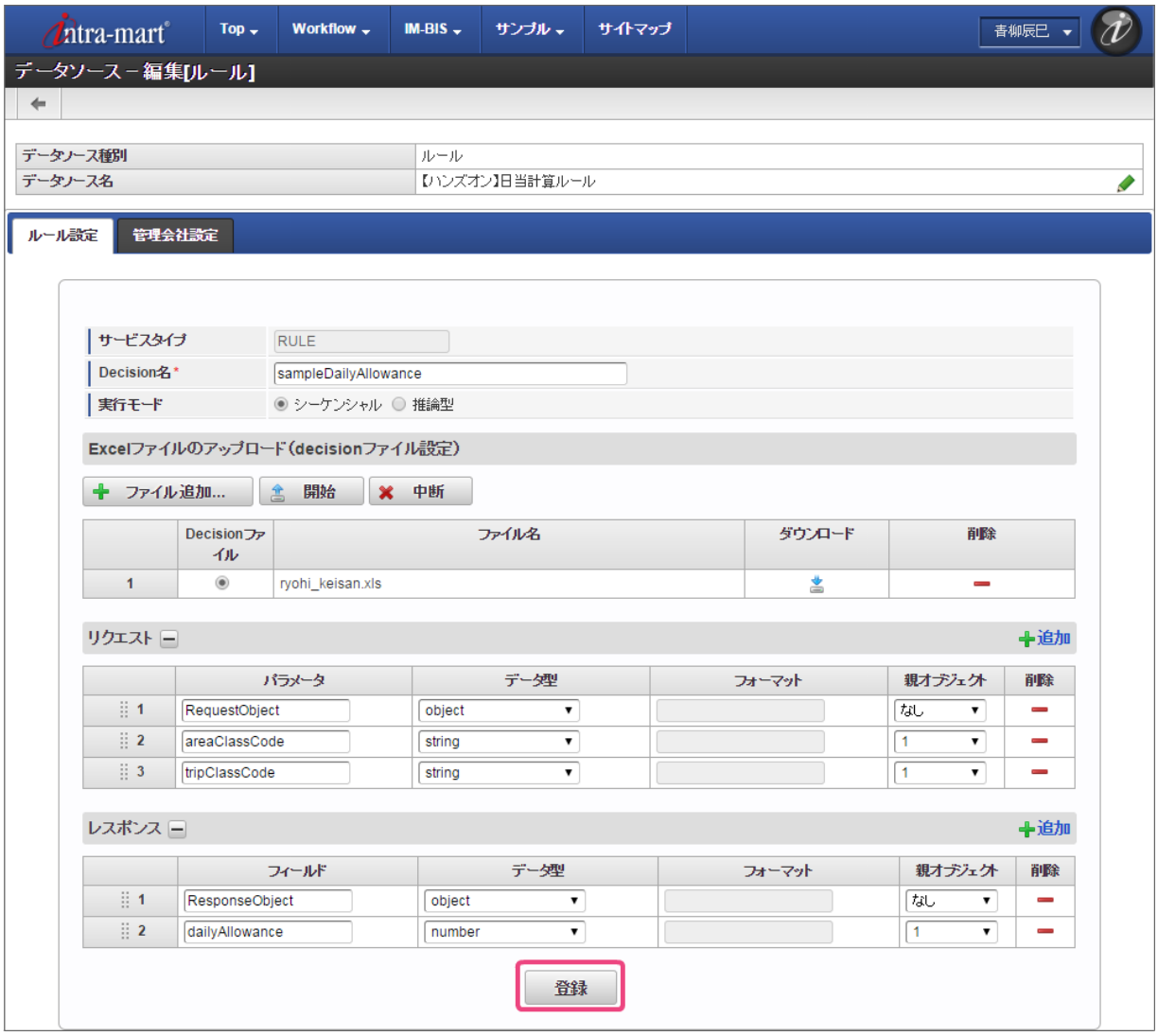
フィールド	データ型	親オブジェクト
ResponseObject	object	なし
dailyAllowance	number	1

4. 作成したExcelのルール定義ファイルをアップロードするために「ファイル追加」をクリックしてください。

5. 「開始」をクリックして、ファイルをアップロードしてください。

6. 「Decisionファイル」のラジオボタンをクリックしてオンにしてください。

7. 最後に「登録」をクリックして、データソース定義を登録してください。



8. これで OpenRules のルールを定義したExcelファイルをデータソース定義として登録することができました。

旅費精算の日当ルールを画面の項目の入力時に実行するための設定を行う

登録したデータソース定義を利用して、 IM-BIS の画面の入力のタイミングでルールが実行されるように設定していきましょう。

フォーム編集画面を表示する

画面アイテムにイベントを設定するために、フォーム（画面）の編集を開始しましょう。

1. サイトマップの「 IM-BIS 」をクリックしてください。

2. 「一覧」をクリックしてください。

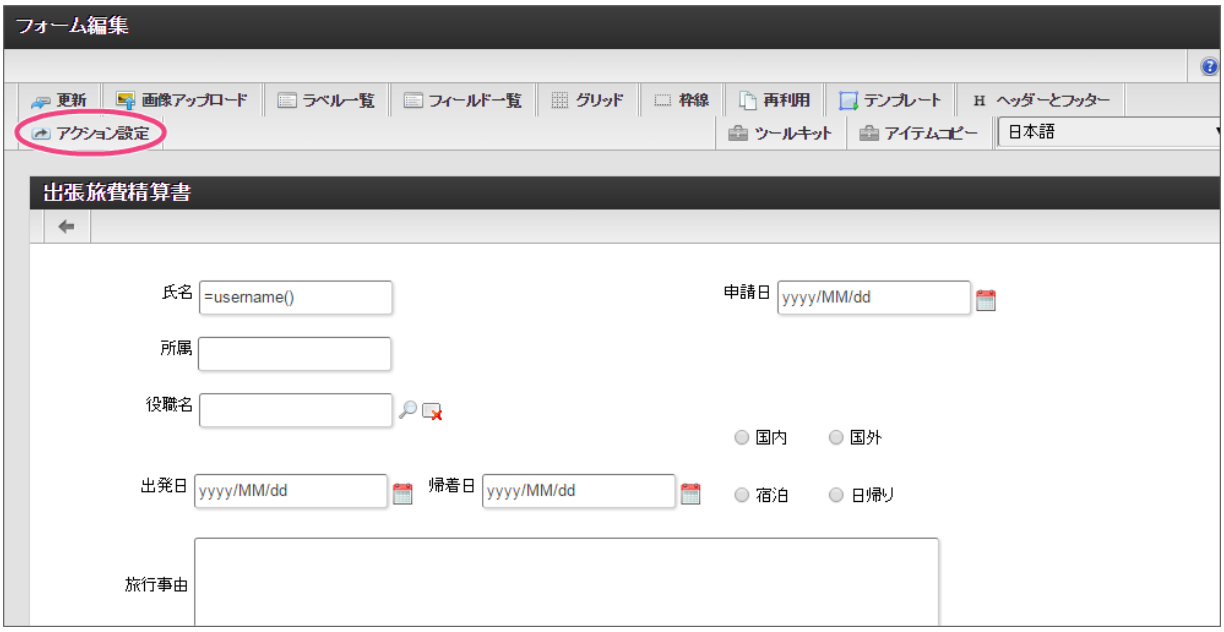
3. インポートしたフローの「【ハンズオン】旅費精算申請」の をクリックしてください。

4. 「申請／処理開始」のアイコンをダブルクリックして、フォーム編集画面（フォーム・デザイナ）を表示してください。

アクション設定で OpenRules との連携情報を設定する

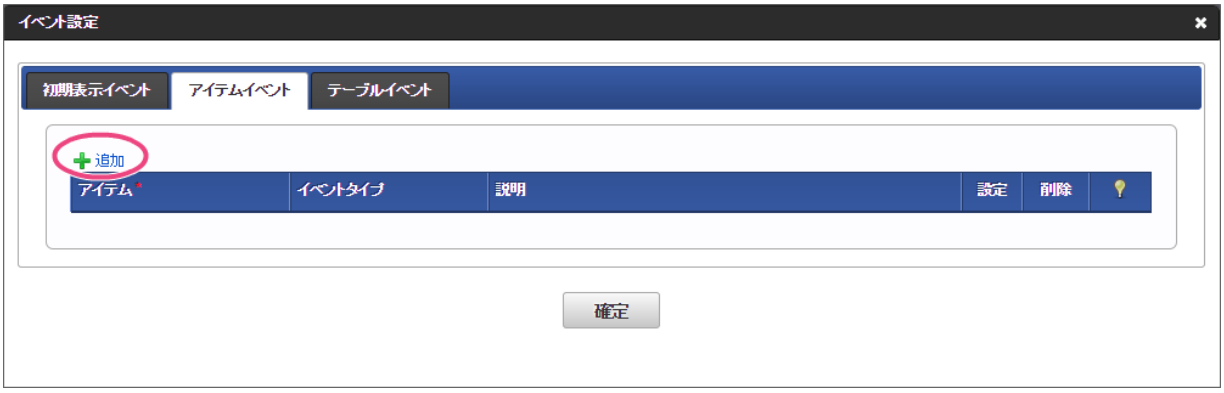
アクション設定で、画面の項目への入力のタイミングでのルールの実行を設定しましょう。
今回のハンズオンでは、場所・出張区分の値が変更されたタイミングでルールを実行するように設定します。

1. フォーム編集画面を表示したら「アクション設定」をクリックしてください。



2. 「アイテムイベント」をクリックして、表示するタブを切り替えます。

3. 「追加」をクリックしてください。



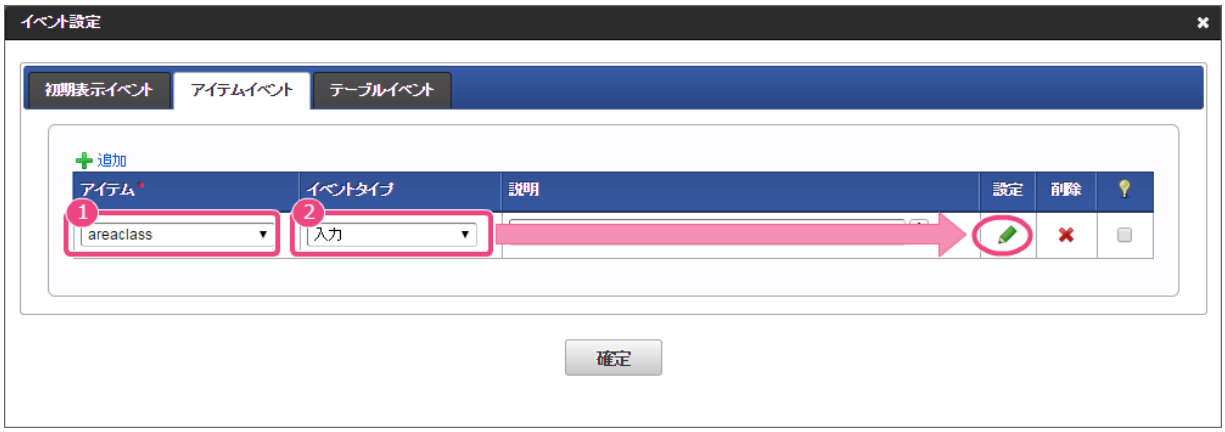
4. アイテムとイベントタイプを以下の通りに変更後、「」をクリックしてください。

    1. アイテム

        areaclass

    2. イベントタイプ

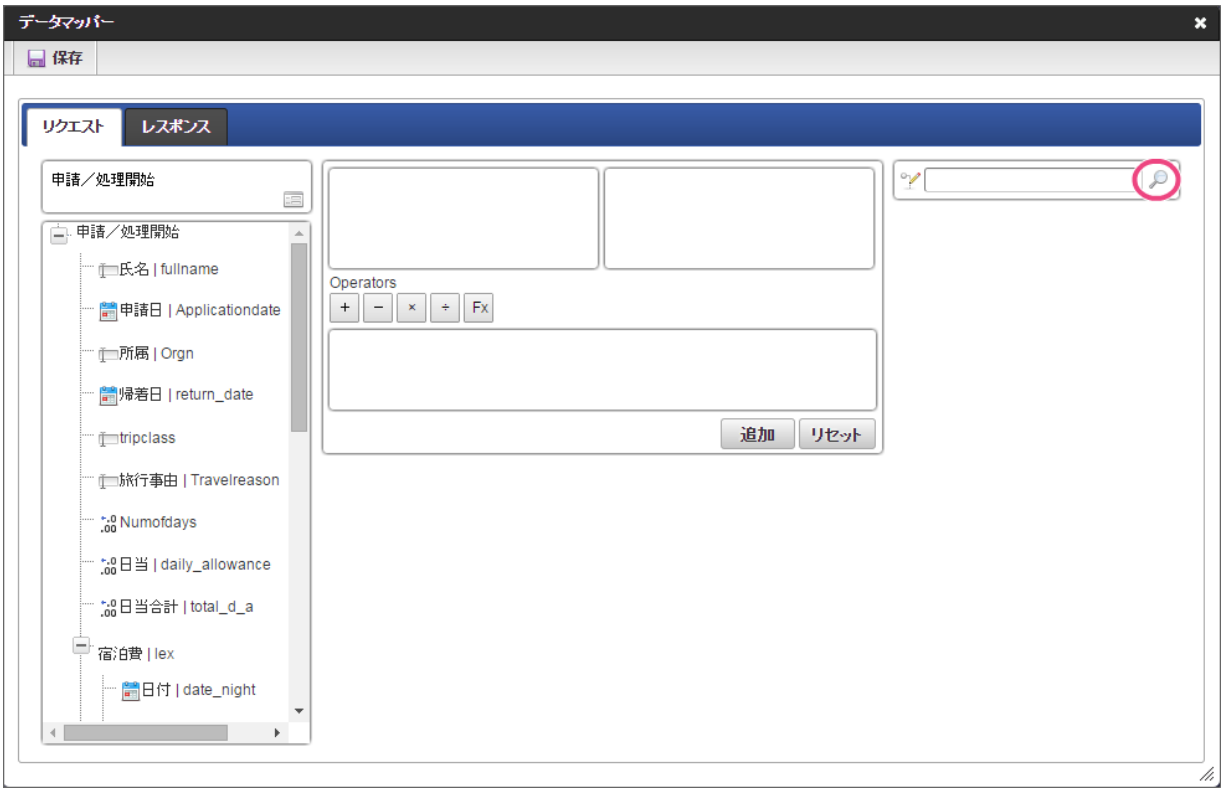
        入力



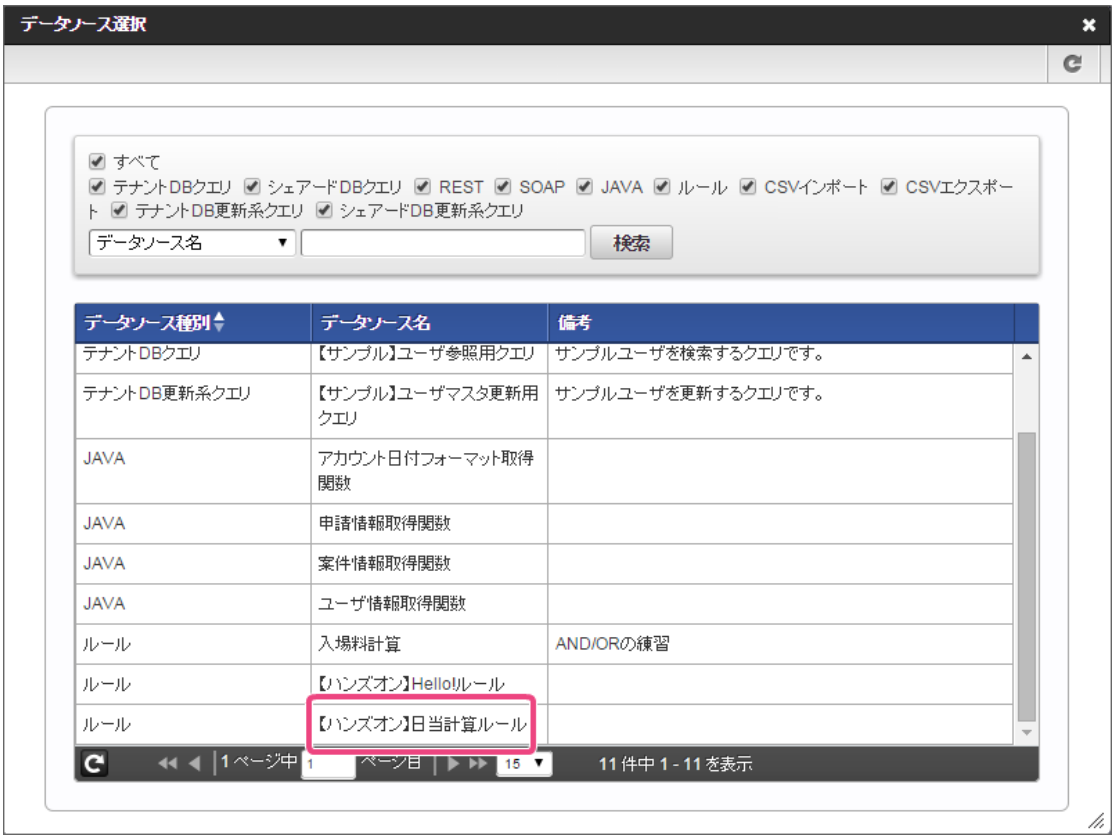
5. 「追加」をクリックしてください。

6. 「アクション」を「外部連携」に設定後、「 」をクリックしてください。

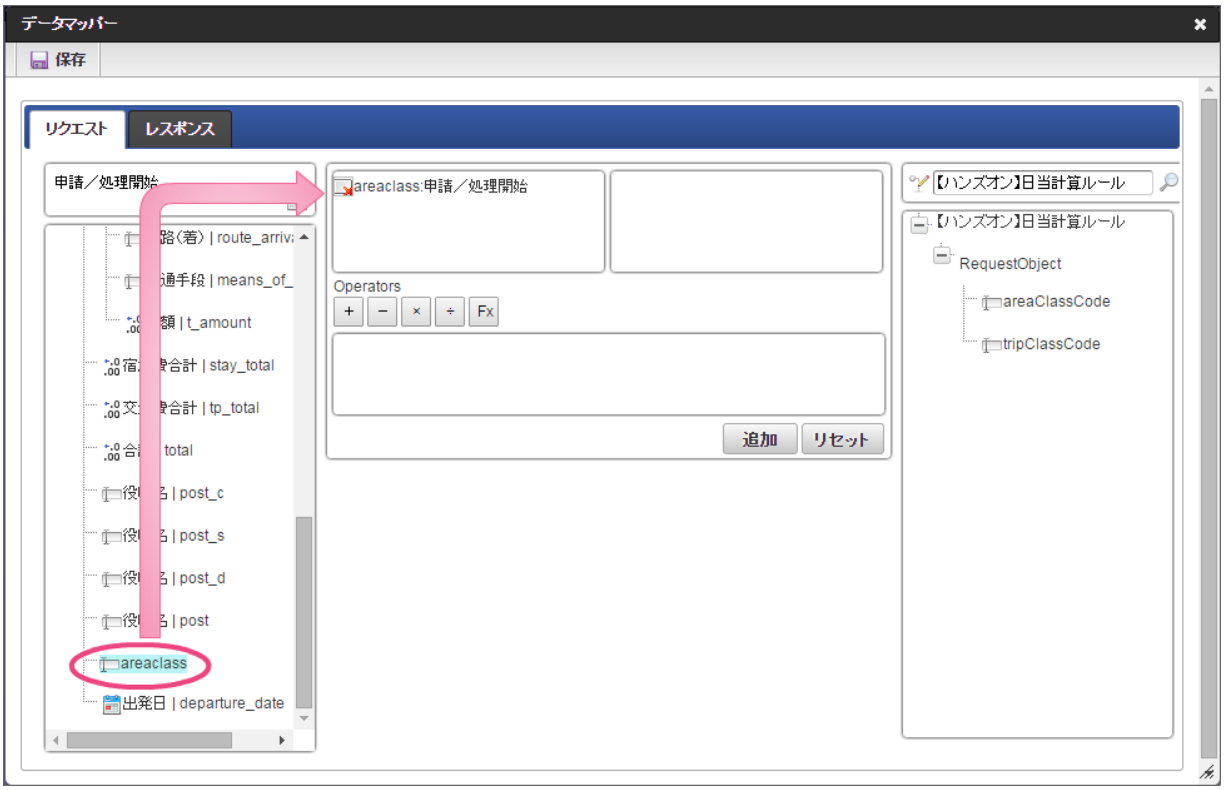
7. 「データマッパー」で右上の をクリックしてください。



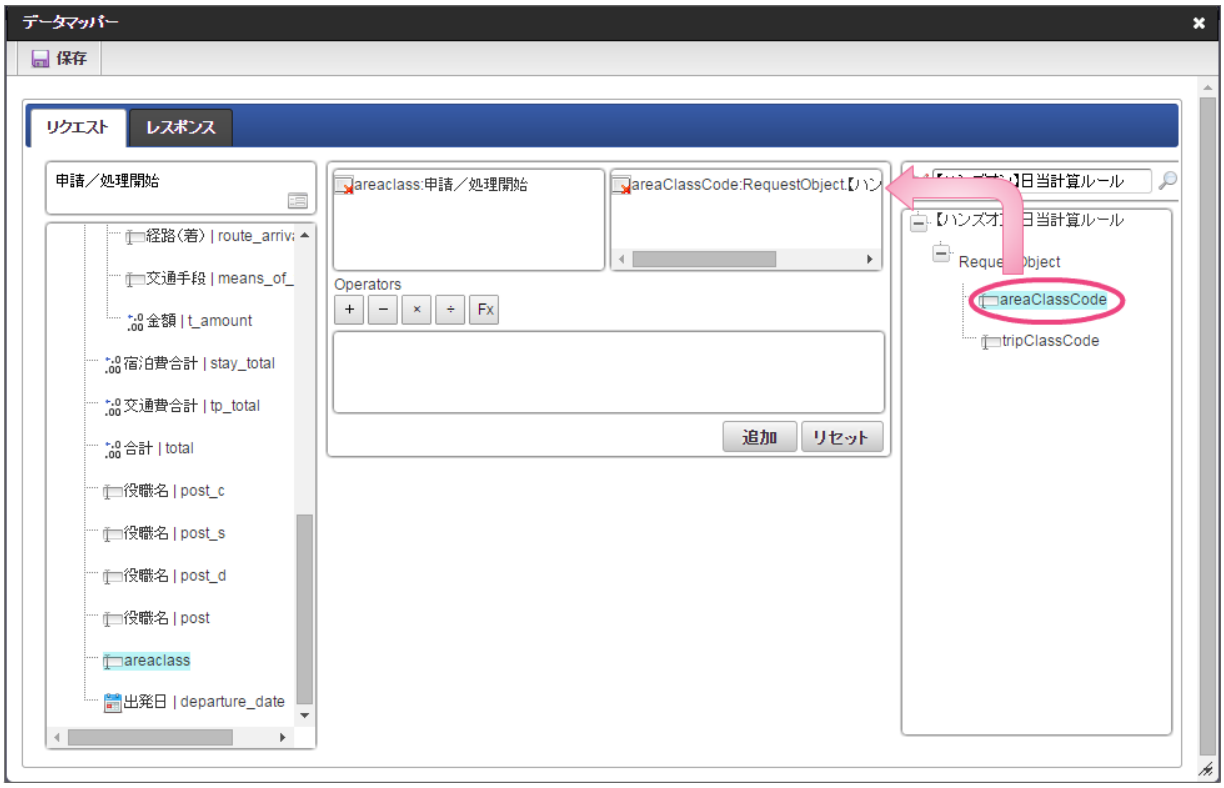
8. データソース選択から作成したルール「【ハンズオン】日当計算ルール」をクリックしてください。



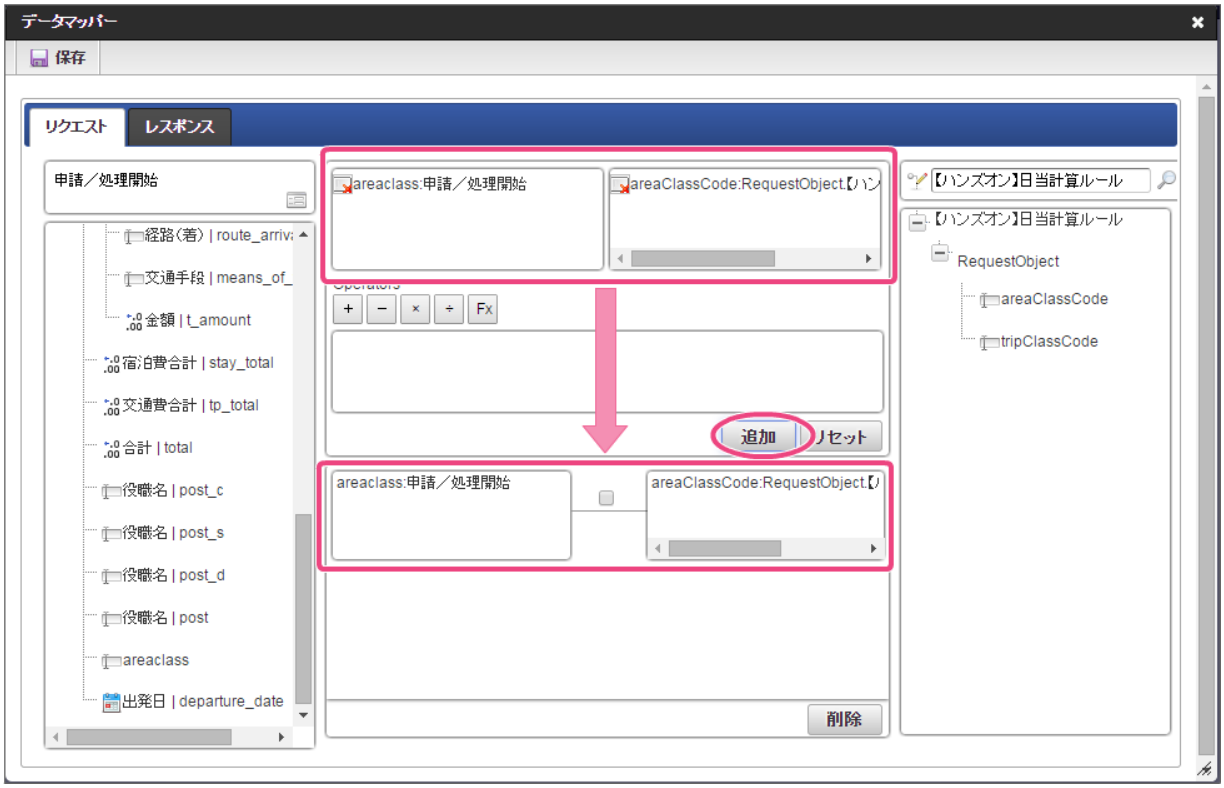
9. 左の欄から「areaclass」をクリックしてください。
   クリック後、中央左の欄に「areaclass: 申請／処理開始」と表示されます。



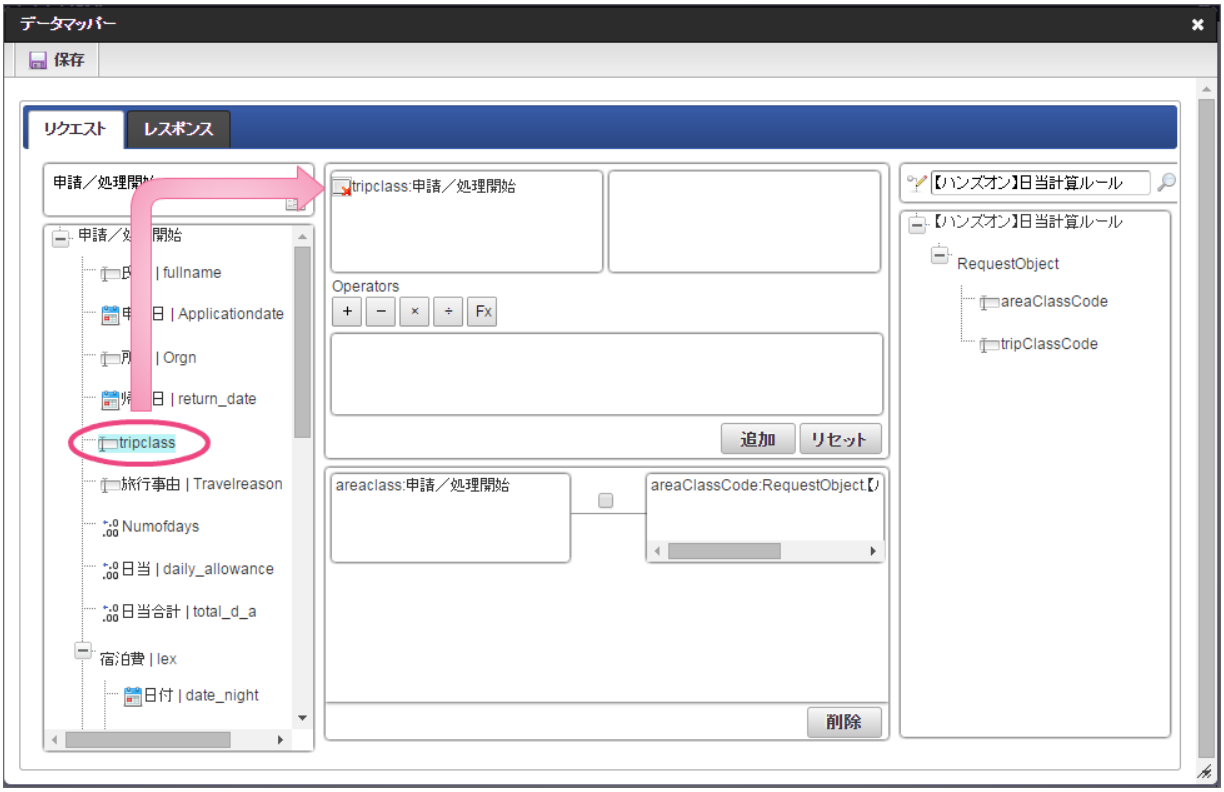
10. 右の欄から「areaClassCode」をクリックしてください。
クリック後、中央右の欄に「areaClassCode:RequestObject.【ハンズオン】日当計算ルール」と表示されます。



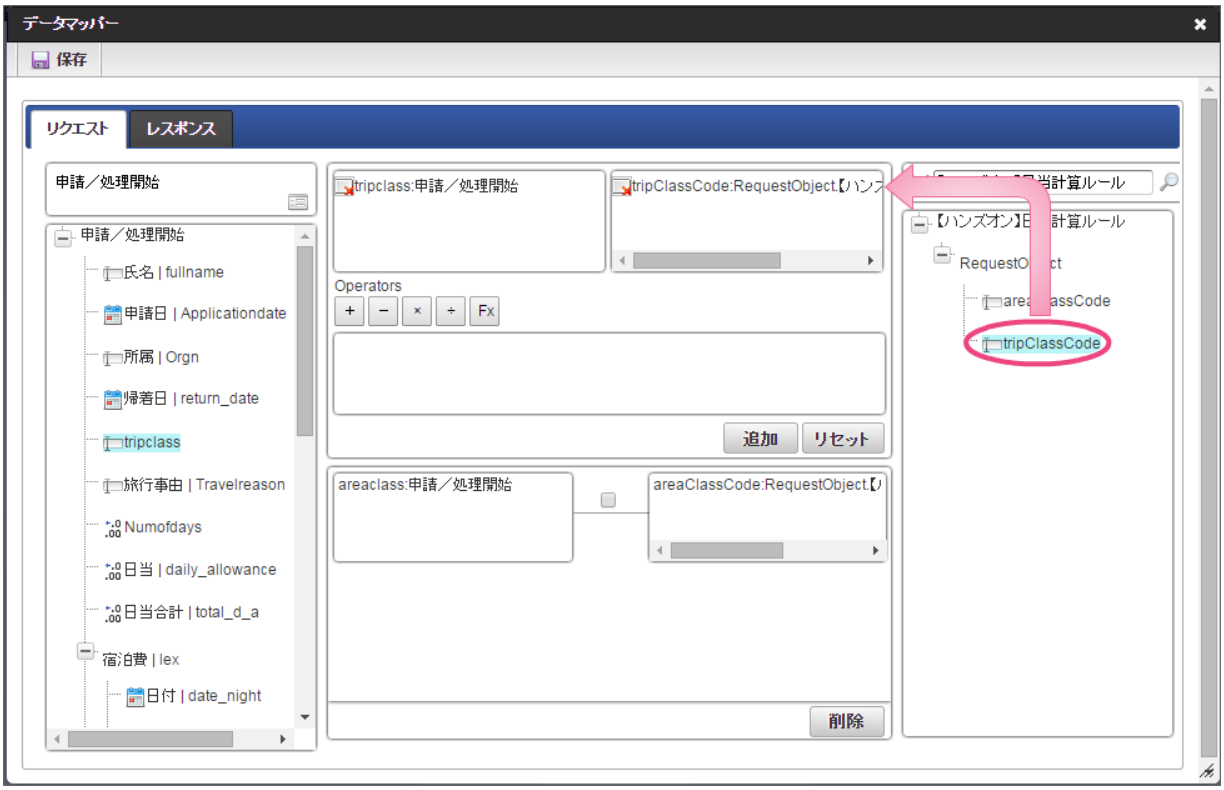
11. 「追加」をクリックしてください。
フォームの「areaclass」とデータソース定義の「areaClassCode」のマッピングが設定され、中央下段の欄に表示されます。



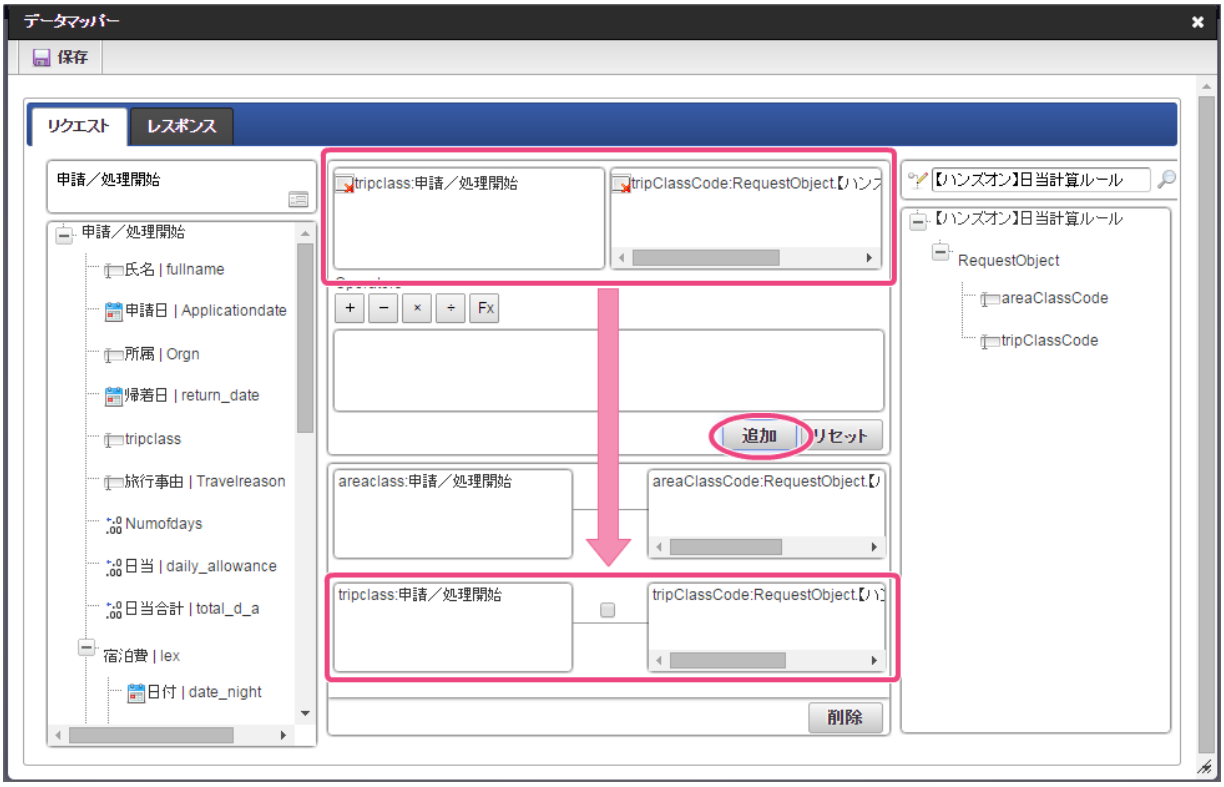
12. 左の欄から「tripClass」をクリックしてください。
クリック後、中央左の欄に「tripClass: 申請／処理開始」と表示されます。



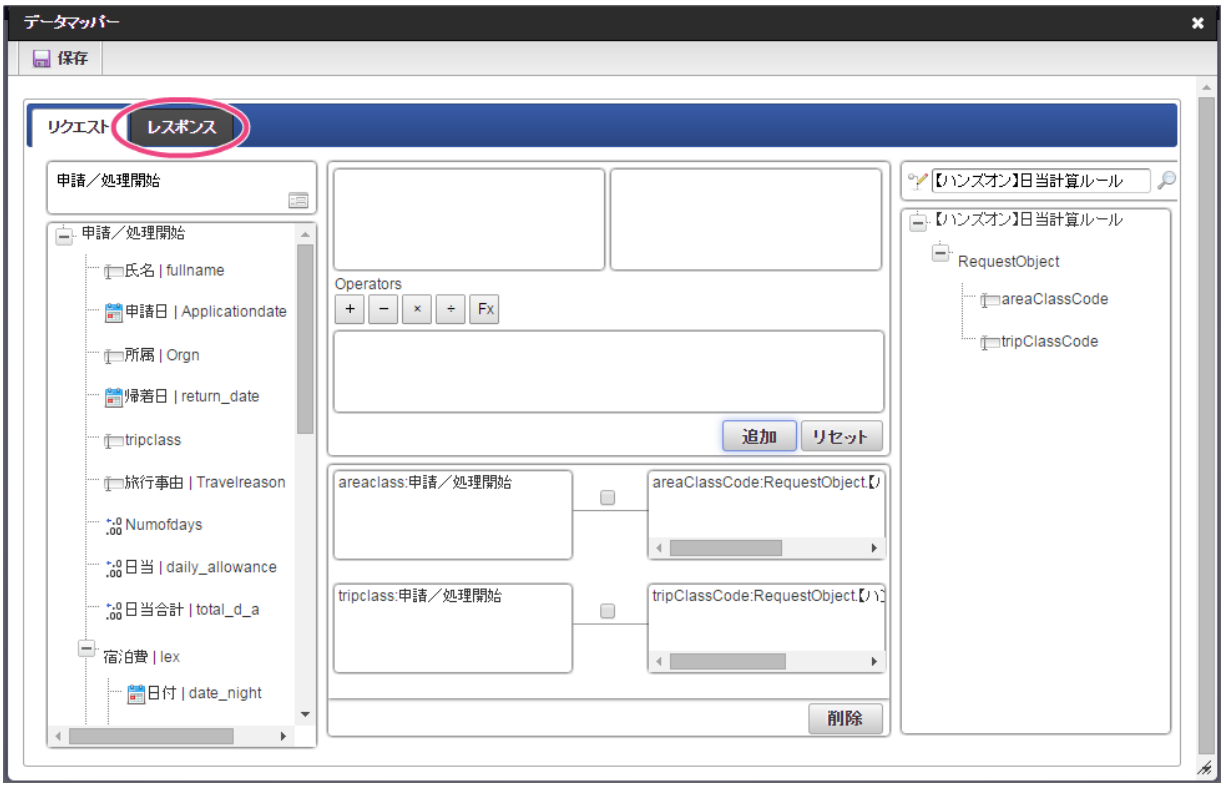
13. 右の欄から「tripClassCode」をクリックしてください。
クリック後、中央右の欄に「tripClassCode:RequestObject.【ハンズオン】日当計算ルール」と表示されます。



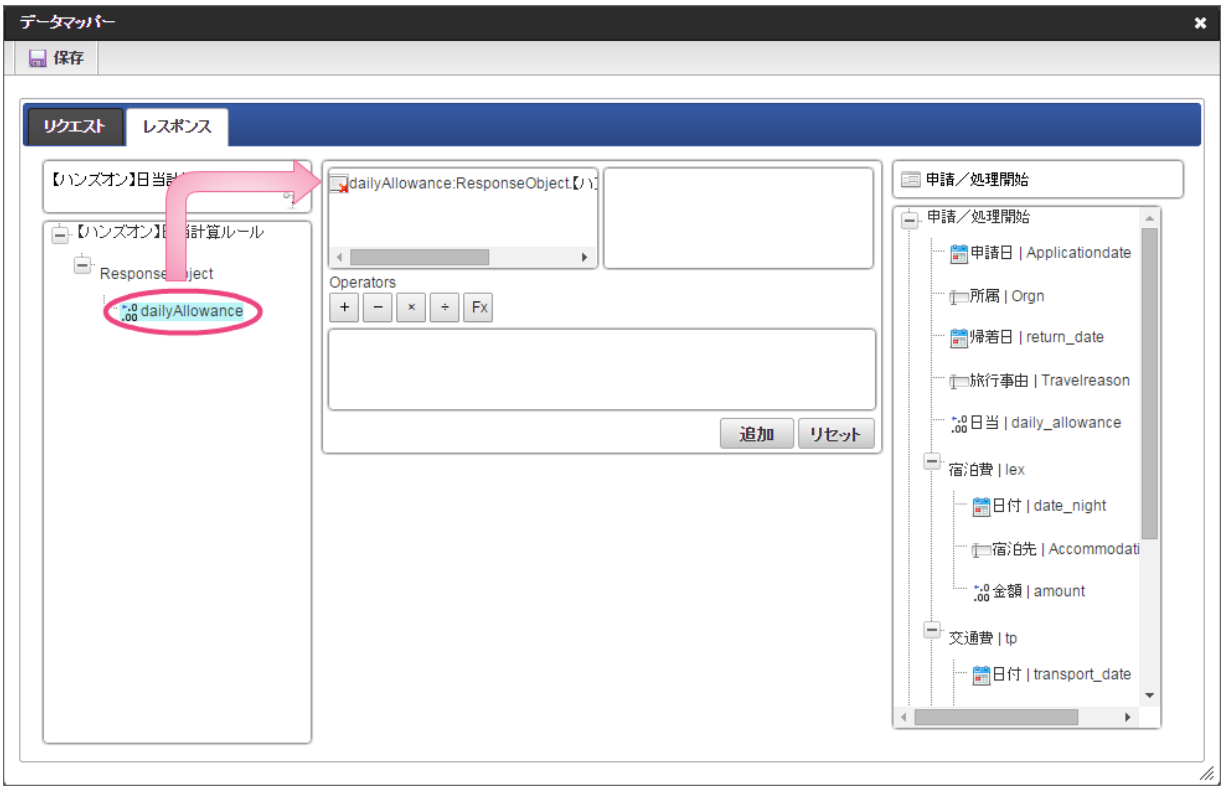
14. 「追加」をクリックしてください。
フォームの「tripClass」とデータソース定義の「tripClassCode」のマッピングが設定され、中央下段の欄に表示されます。



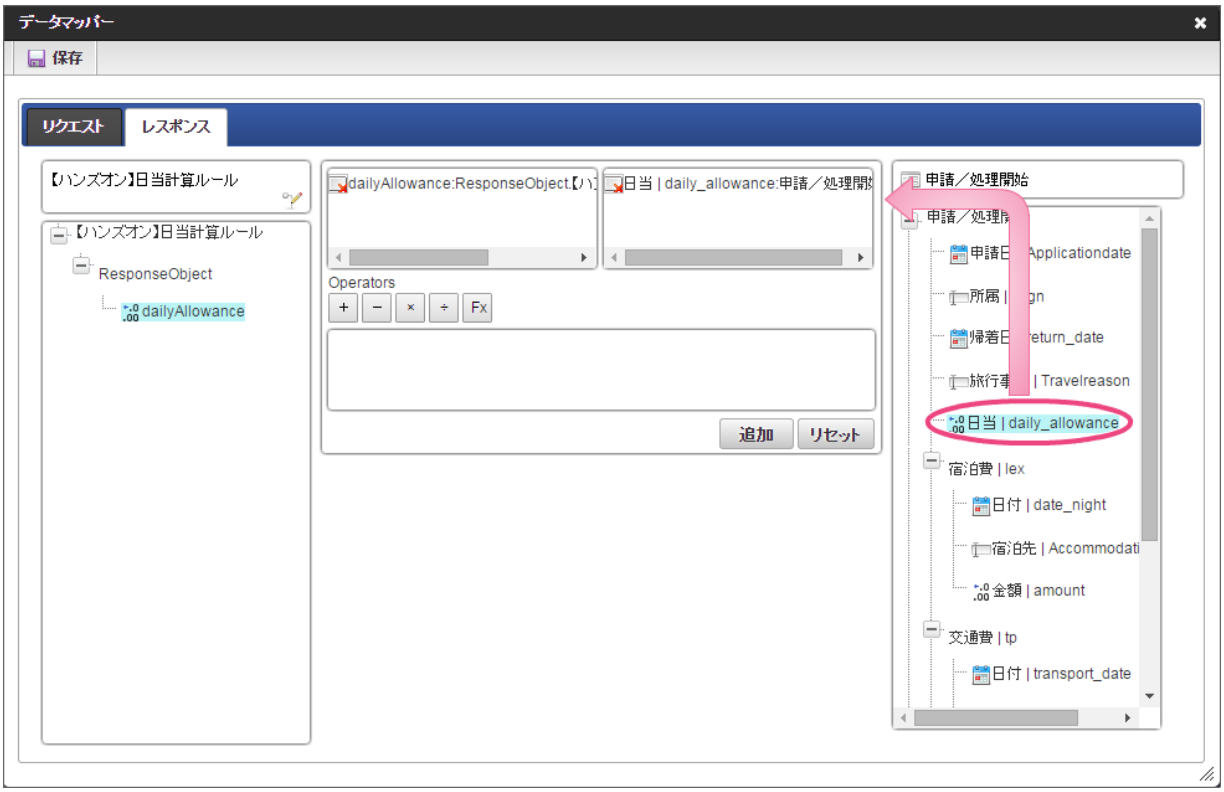
15. リクエストの設定が完了しましたので、レスポンスの設定を行うために「レスポンス」タブをクリックしてください。



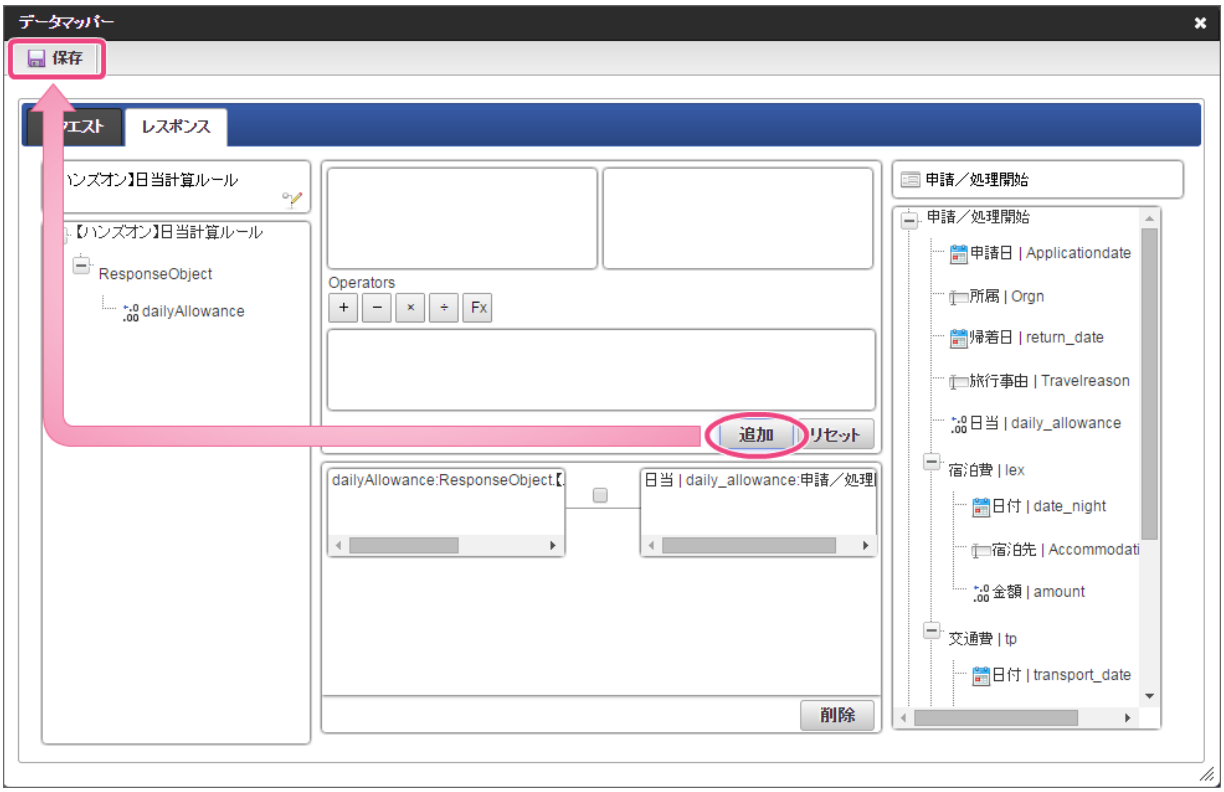
16. 左の欄から「dailyAllowance」をクリックしてください。
クリック後、中央左の欄に「dailyAllowance:ResponseObject.【ハンズオン】日当計算ルール」と表示されます。



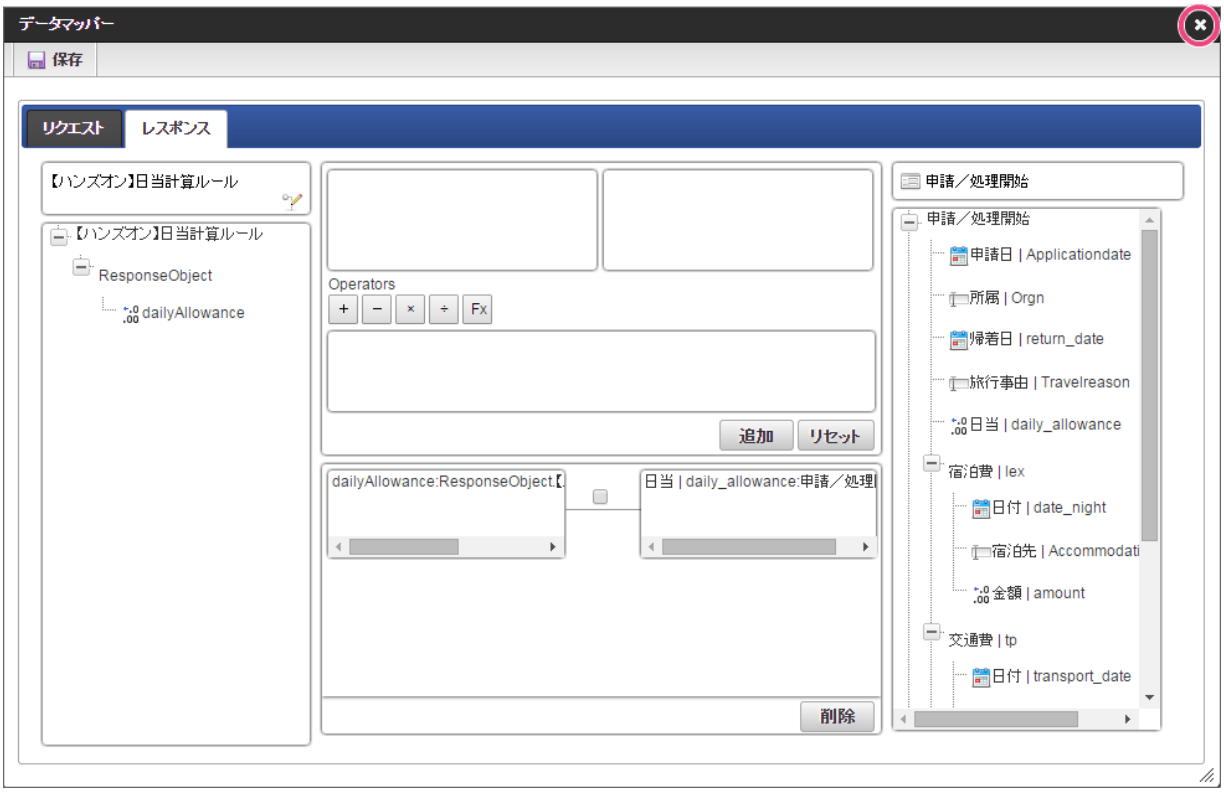
17. 右の欄から「日当」をクリックしてください。
クリック後、中央右の欄に「日当 | daily_allowance: 申請／処理開始」と表示されます。



18. 「追加」、「保存」の順にクリックしてください。



19. 正常に保存できたら、「データマッパー」は右上の「 ✖ 」をクリックして閉じてください。



20. アクション設定で「確定」をクリックしてください。

21. イベント設定で「確定」をクリックしてイベントの設定を保存してください。

22. 引き続き、出張区分の入力時のアクションイベントを設定するために、「＋ 追加」をクリックしてください。

23. 追加した行のアイテムとイベントタイプを以下の通りに変更後、「 」をクリックしてください。

    1. アイテム

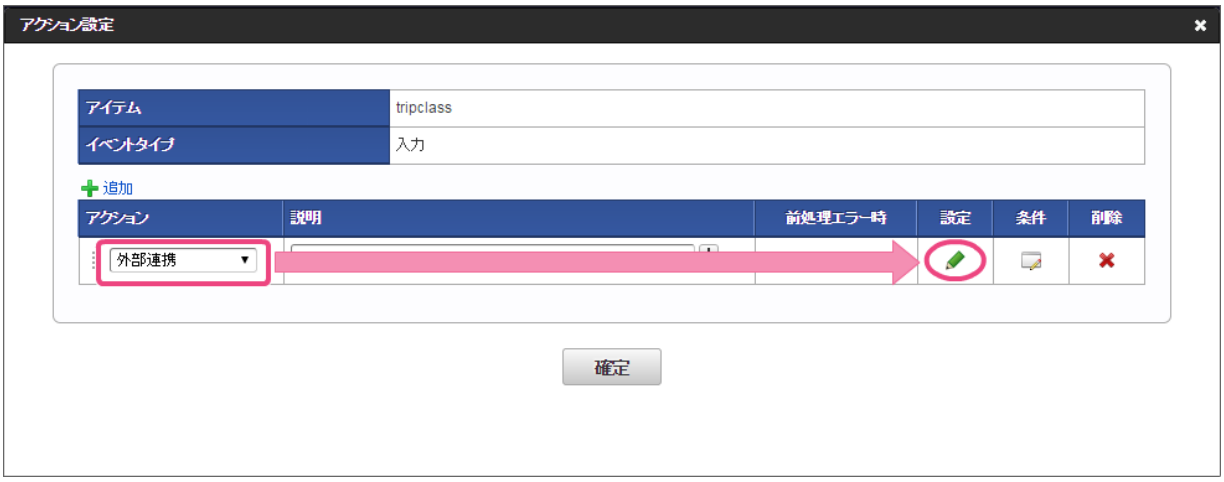
       tripclass

    2. イベントタイプ

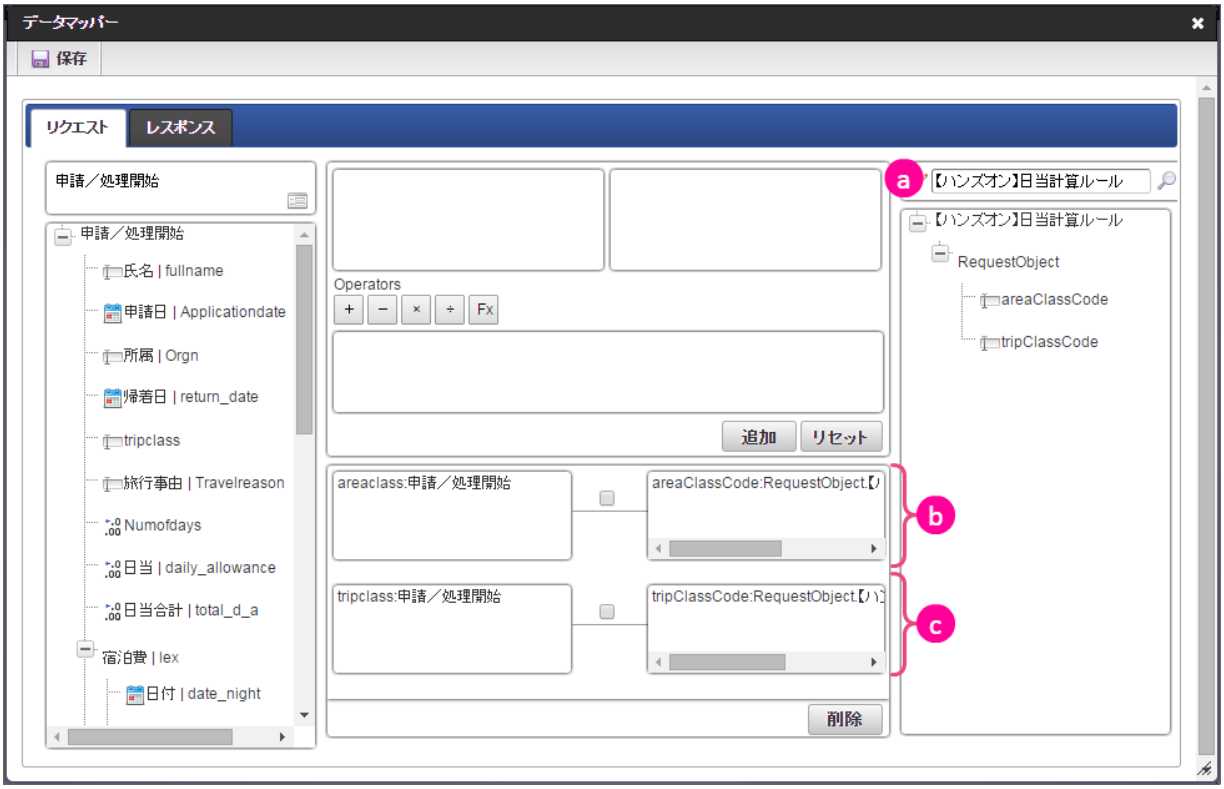
       入力

24. 「 追加」をクリックしてください。

25. 「アクション」を「外部連携」に設定後、「 」をクリックしてください。



26. 「データマッパー」で以下の通りに設定してください。

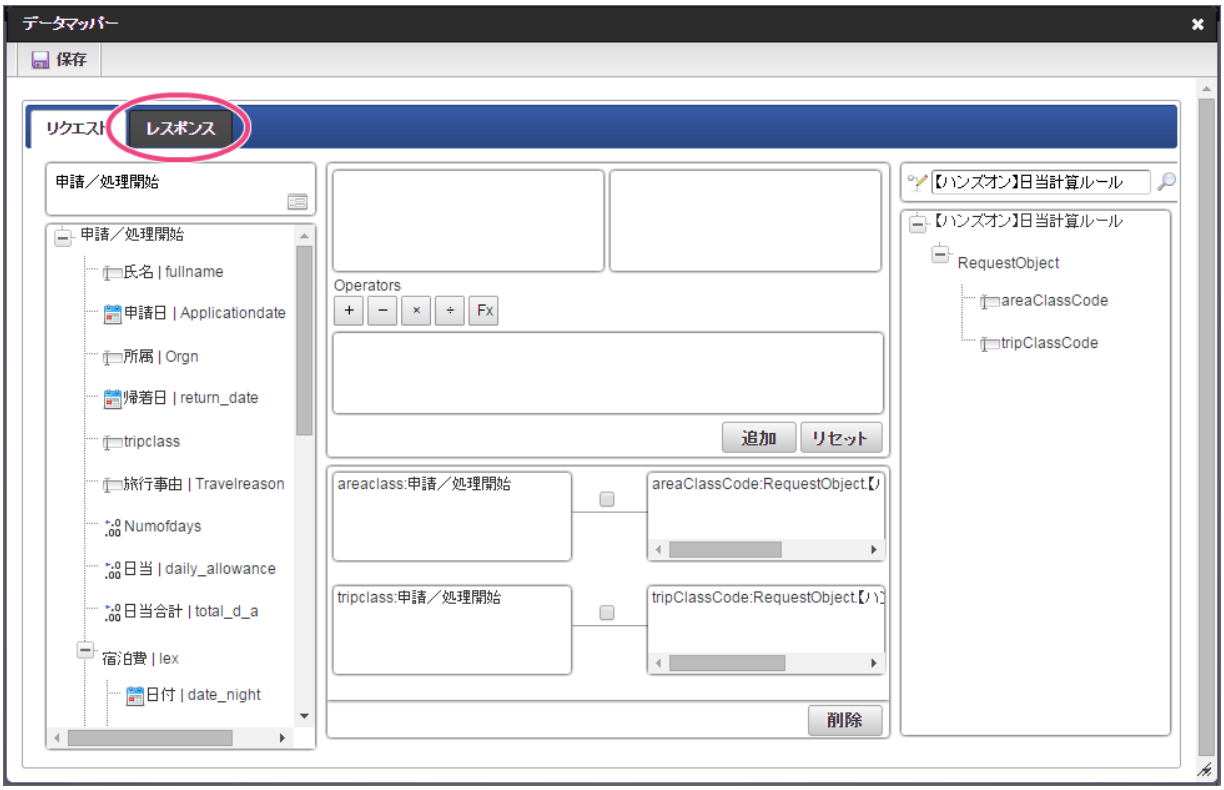


a. データソース定義
「【ハンズオン】日当計算ルール」を選択してください。

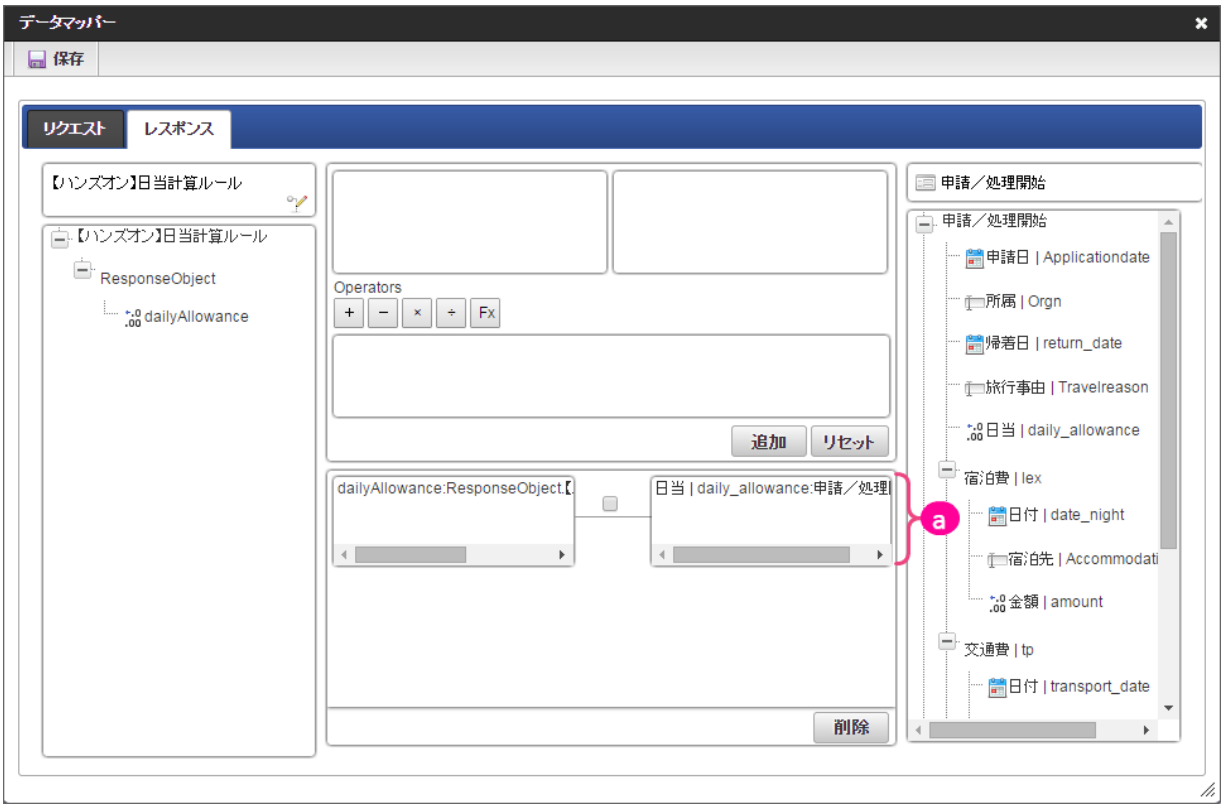
b. 場所のマッピング
左の画面（フォーム）項目の「areaclass」と右のデータソース定義のリクエストパラメータの「areaClassCode」をマッピングしてください。

c. 出張区分のマッピング
左の画面（フォーム）項目の「tripclass」と右のデータソース定義のリクエストパラメータの「tripClassCode」をマッピングしてください。

27. リクエストの設定が完了しましたので、レスポンスの設定を行うために「レスポンス」タブをクリックしてください。

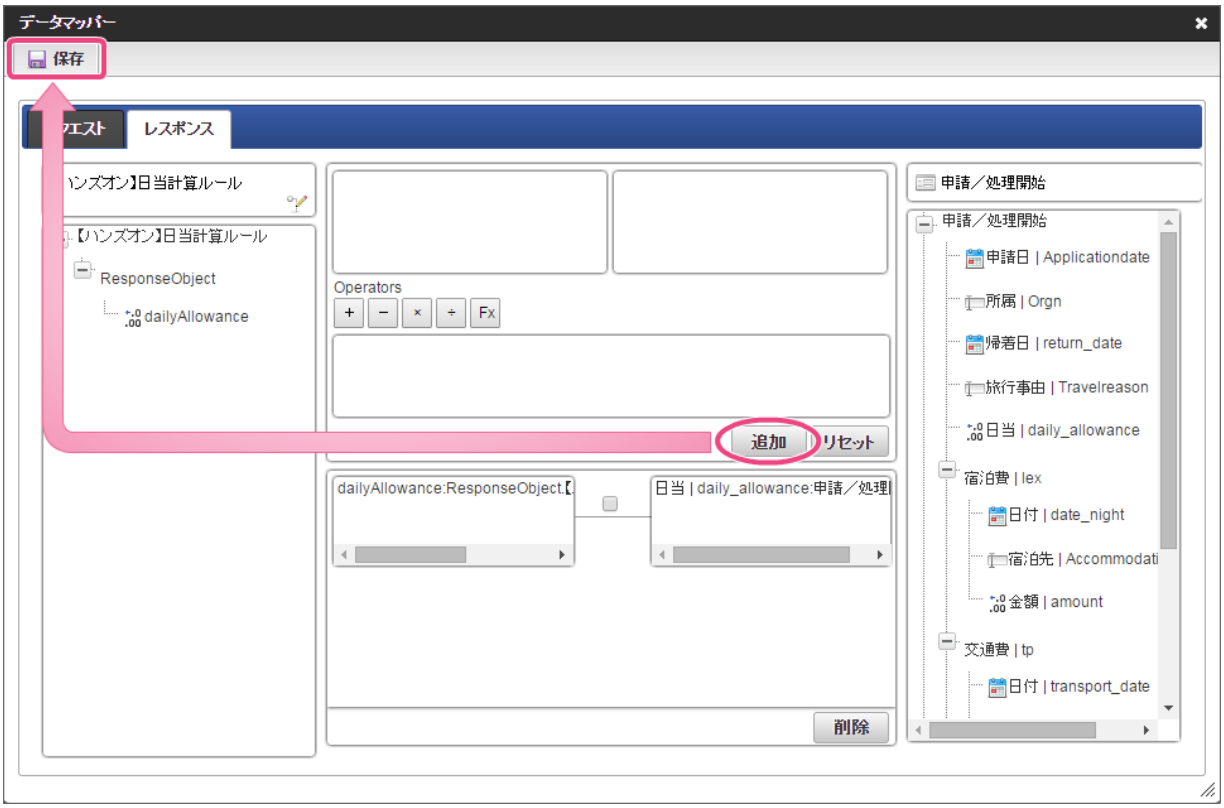


28. 「データマッパー」で以下の通りに設定してください。

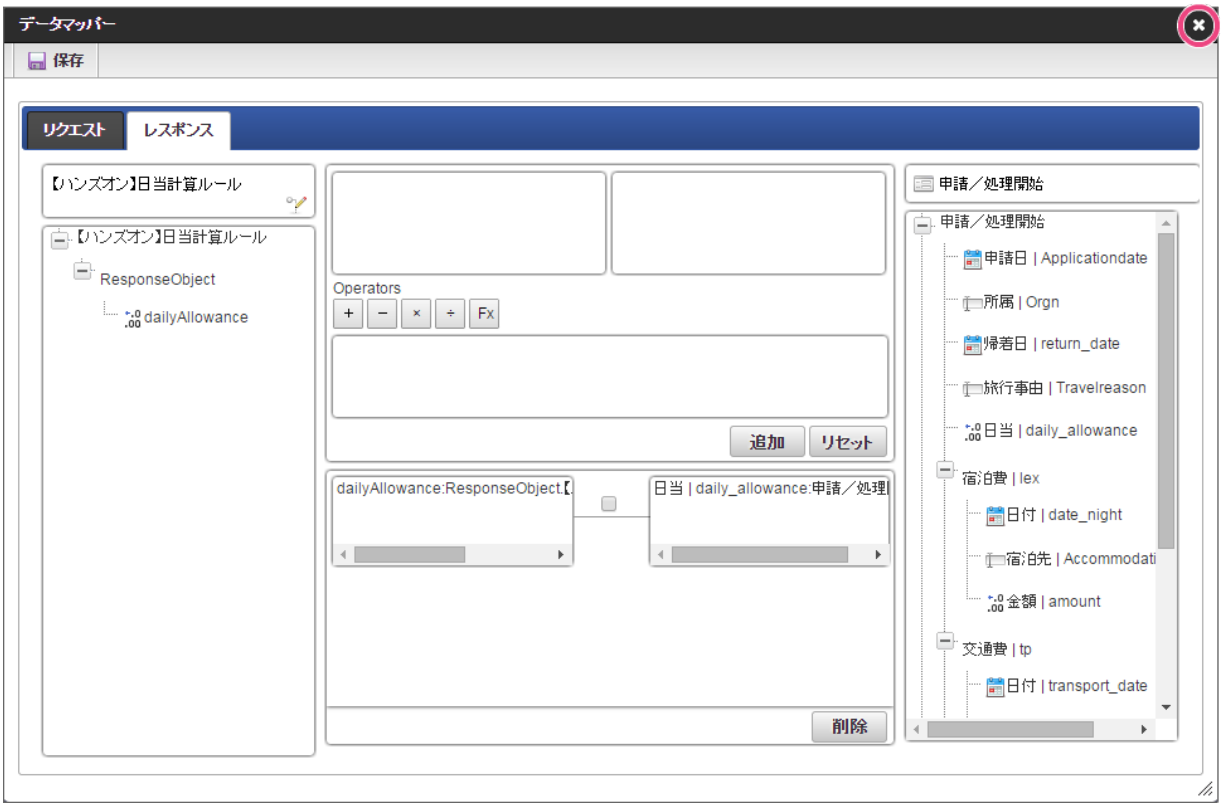


a. 日当のマッピング
　左のデータソース定義のレスポンスフィールドの「dailyAllowance」と右の画面（フォーム）項目の「日当」をマッピングしてください。

29. 「追加」、「保存」の順にクリックしてください。



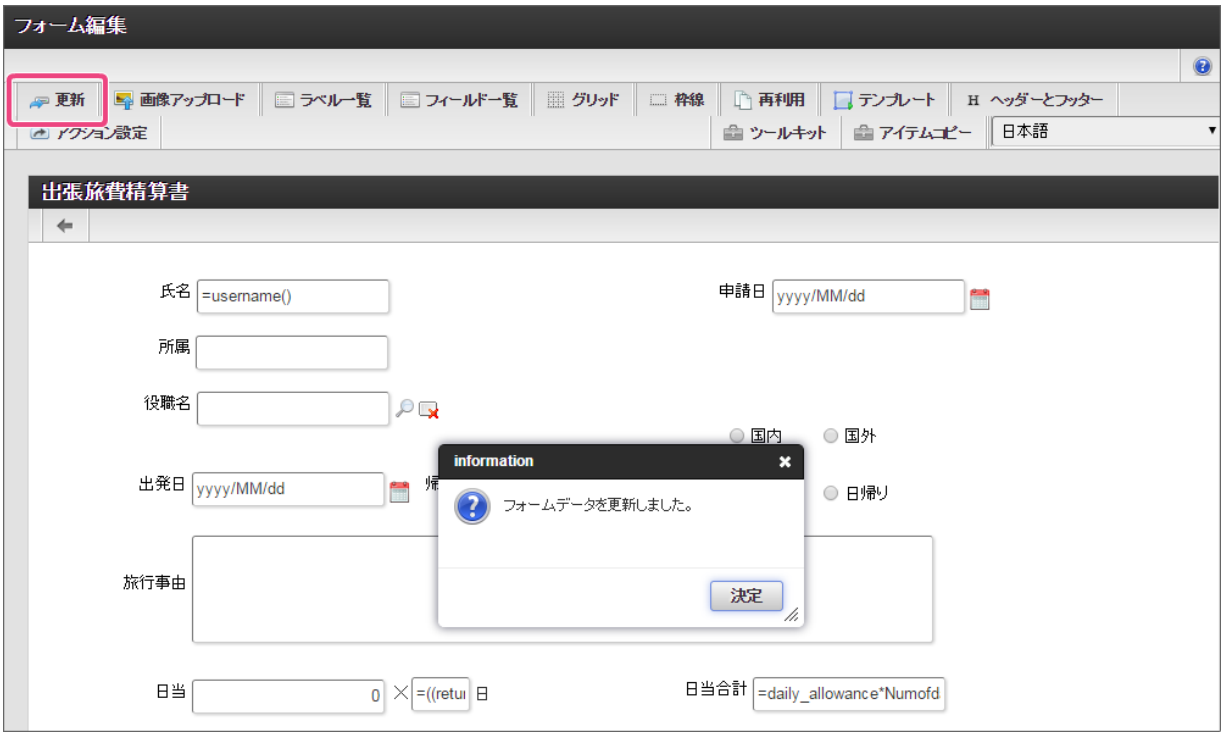
30. 正常に保存できたら、「データマッパー」は右上の「 ✖ 」をクリックして閉じてください。



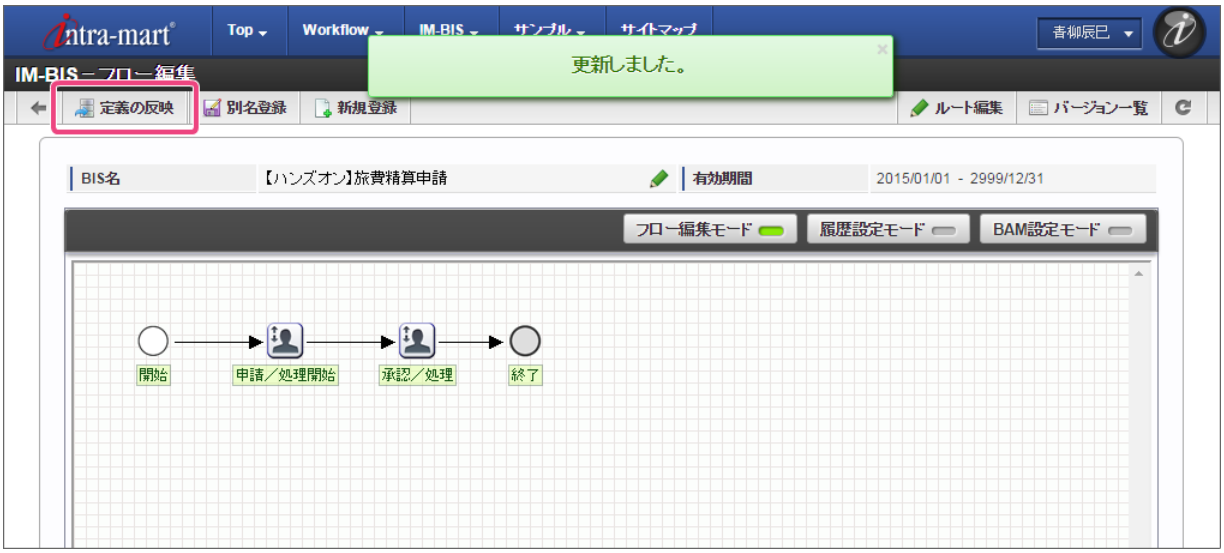
31. アクション設定で「確定」をクリックしてください。

32. イベント設定で「確定」をクリックしてください。

33. 「更新」をクリックして、フォーム（画面）を保存してください。



34. 最後に「定義の反映」をクリックして、フローの実行準備を行ってください。



35. これで、 OpenRules の結果に基づいて日当計算ができるようになりました。
次の手順で、実行して日当が正しく計算されるのを確認してみましょう。

出張旅費精算申請で日当を計算してみよう

これまでのシナリオで作成した IM-BIS のフローを使って、日当の計算を実行してみましょう。

ルールと連携したフローを実行する手順
- 旅費精算の申請画面でルールを実行する

旅費精算の申請画面でルールを実行する

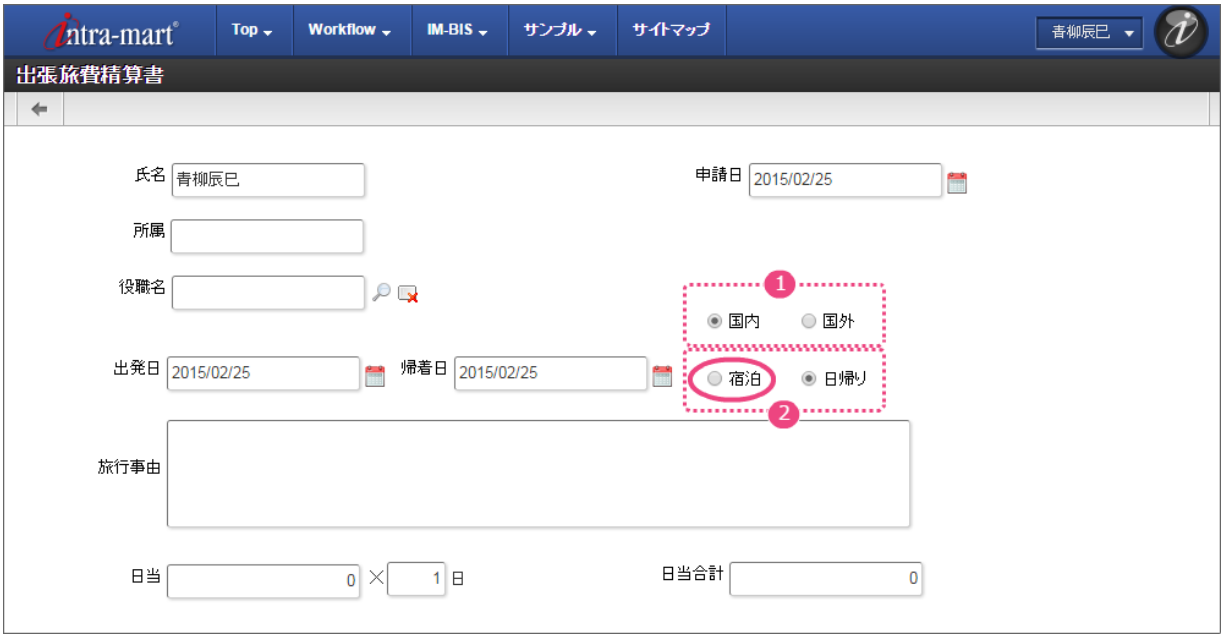
作成したワークフローの申請画面でルールを実行してみましょう。

1. 「BIS担当者」ロールを付与したユーザでログインしましょう。
（このマニュアルでは、「青柳辰巳」（ユーザコード：aoyagi）でログインします。）

2. 上部のメニューの「 IM-BIS 」→「ワークフロー」の順にマウスを重ねてから「申請」をクリックしてください。

3. 「【ハンズオン】旅費精算申請」の「 申請／処理開始」をクリックしてください。

4. 申請画面で(1)場所、(2)宿泊区分の値を変更してみましょう。
   （例として、ここでは「宿泊区分」を「宿泊」に変更します。）



5. OpenRules で設定した通りに、日当の金額が変わることが確認できました。
   また、日当の金額の変更に応じて、日当合計も正しく計算されています。

6. この後は、申請や承認を行ってみてください。

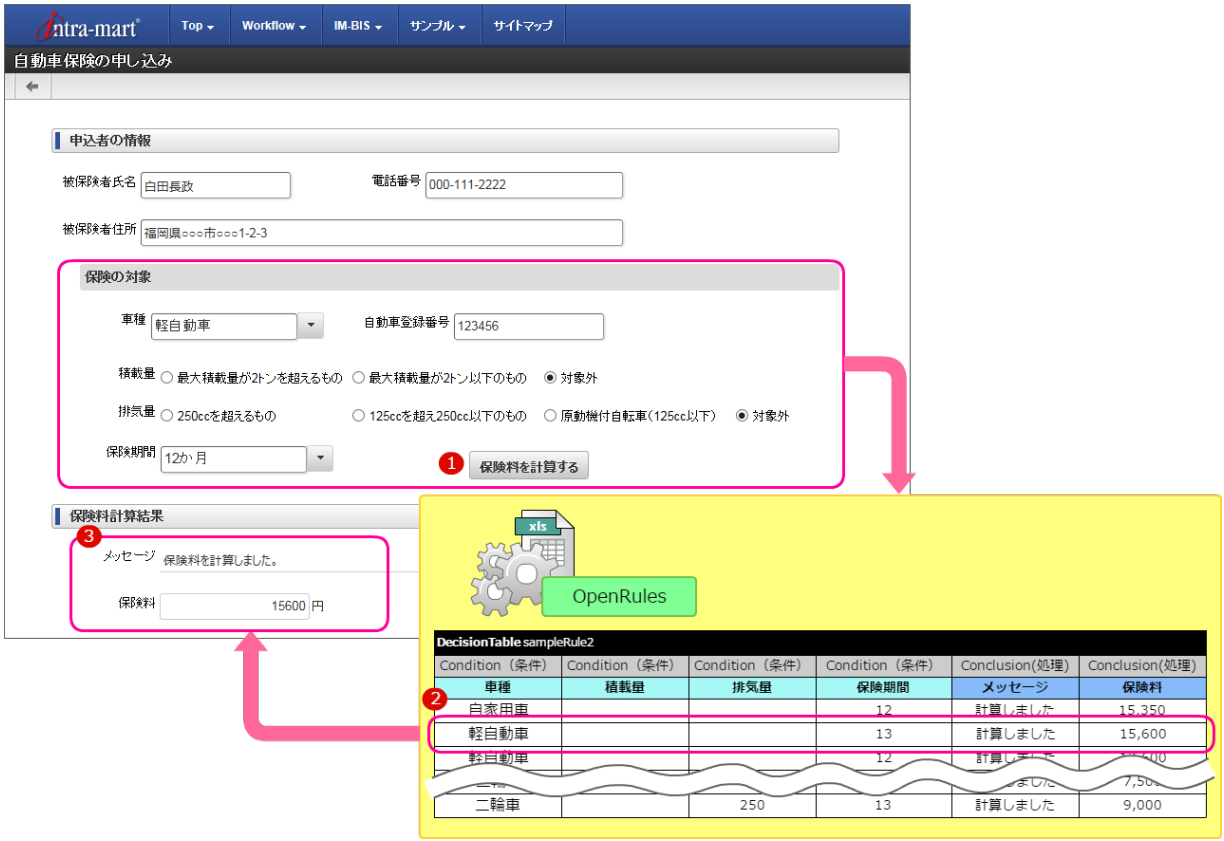
自動車保険の計算と入力チェックをしてみよう

本章では、 OpenRules for IM-BIS 連携の画面項目との値の受け渡しなどを伴うフローの開発方法について説明しております。
シナリオを実行しながら、高度なルールの記述方法を確認することができます。

ハンズオンシナリオ（自動車保険申請の作成）の概要

このシナリオでは、ルールで入力チェックを行うフローを作成します。

- ユーザが自動車保険の審査項目を入力すると、入力内容に応じた保険料を計算、または入力内容の不備を画面に表示する



このシナリオを通して習得できる知識

このシナリオを実行すると、以下の知識を理解することができます。

- OpenRules での演算子の利用方法
- OpenRules の評価結果を利用した入力チェックの実装方法

このシナリオを学習する前に必要な知識

このシナリオの実行に当たり、下記の知識を事前に確認してください。

- IM-BIS 、 IM-FormaDesigner の定義ファイルのインポート方法
- 「 *複合条件（AND/OR）を利用して、旅費精算の日当を計算してみよう* 」に紹介されているラジオボタンアイテムの送信値・表示値の逆マッピングの方法

OpenRules で自動車保険申請のルールを作成する

OpenRules で自動車保険の保険料を計算するためのルールを作成します。

このハンズオンでは、自動車保険の申し込み申請フローをサンプルに、「いずれかと一致」などの演算子やルールの結果に基づく入力チェックの実装方法を確認することができます。

また、このハンズオンでは、登録ずみのデータソース定義のExcelのルール定義ファイルを更新しながら、ルール変更時の対応手順を確認します。

ルールを定義するExcelファイルを作成する手順
- このシナリオで作成するルールの概要
- ルールのExcelファイルを作成する手順
- 自動車保険料計算のハンズオンを開始するための準備
- DecisionTable の条件となる値を確認する
- 保険対象の内容のチェックと保険料を計算するための DecisionTable を作成する

このシナリオで作成するルールの概要

- 作成するルールの内容
  1. 自動車保険の契約対象を条件に、保険料を計算する。
     保険対象に関する情報が正しく設定されていない場合には、入力チェックでエラーとする。
     - 入力値：車種・保険期間・（車種により）最大積載量・（車種により）排気量
     - 出力値：保険料、メッセージ

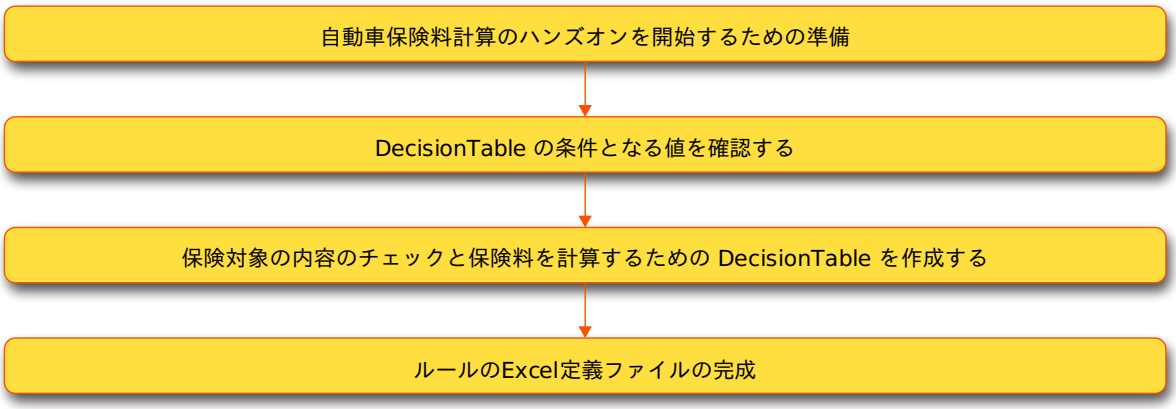
  条件と返却する保険料の組み合わせは、以下の通りです。
  （以下の表内に登場しない条件の組み合わせは、入力チェックエラーとして扱います。）

		保険期間						
車種/区分		**12ヶ月契約**	**13ヶ月契約**	**24ヶ月契約**	**25ヶ月契約**	**36ヶ月契約**	**37ヶ月契約**	**48ヶ月契約**
普通自動車		16,400円	17,300円	27,800円	28,800円	39,100円	40,000円	－
軽自動車		15,600円	16,500円	26,400円	27,200円	36,900円	37,800円	－
トラック	積載量 2トン以下	24,000円	25,600円	43,100円	44,600円	－	－	－
	積載量 2トン超	35,700円	38,300円	66,200円	68,700円	－	－	－
バイク	排気量 125cc未満（原付）	7,300円	－	9,900円	－	12,400円	－	14,900円
	排気量 125cc～250cc（軽二輪車）	9,500円	－	14,300円	－	19,000円	－	23,600円
	排気量 250cc（小型二輪車）	9,200円	9,600円	16,400円	14,000円	18,000円	18,400円	－

ルールのExcelファイルを作成する手順

今回は、あらかじめ登録済みのデータソース定義のルールを更新します。
このシナリオでは、以下の図の流れで作成していきます。

- 自動車保険料を計算するルールの作成の手順



自動車保険料計算のハンズオンを開始するための準備

データソース定義をダウンロードする

このハンズオンでは、あらかじめ登録済みのデータソース定義のルールを更新する方法を行います。
まずは、対象のデータソース定義のファイルを入手しましょう。

1. 以下のリンクからデータソース定義のファイルをダウンロードしてください。

datasource_auto_insurance.zip

データソース定義ファイルをインポートする

先の手順でダウンロードしたファイルをデータソース定義からインポートし、変更を行うExcelのルール定義ファイルを入手します。

1. サイトマップの「 IM-BIS 」から「データソース定義インポート」をクリックしてください。

2. インポートファイルに、先にダウンロードしたデータソース定義ファイルを選択し、「インポート実行」をクリックしてください。

3. 以下のように表示されたら、データソース定義ファイルが正常にインポートできました。

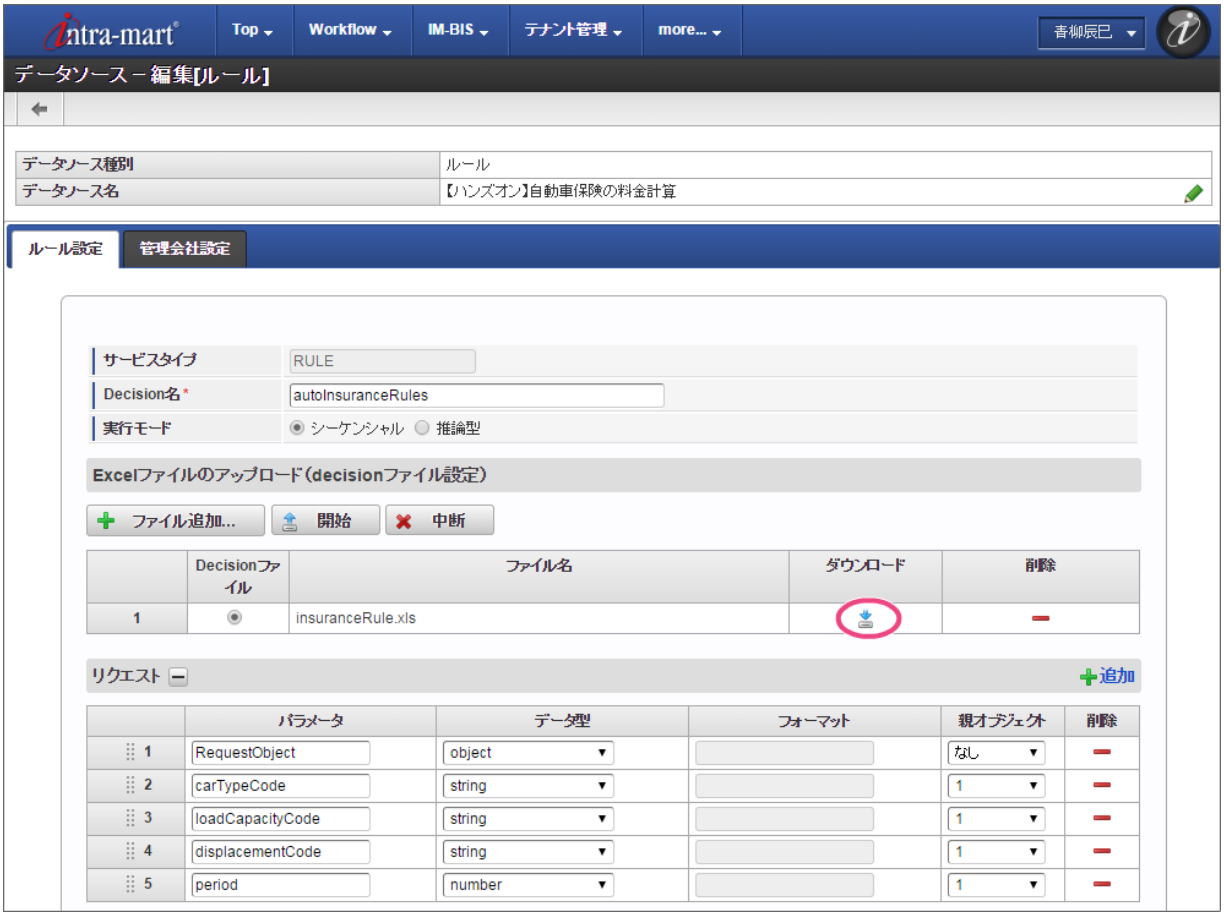
データソース定義からExcelのルール定義ファイルをダウンロードする

データソース定義からExcelのルール定義ファイルを入手します。

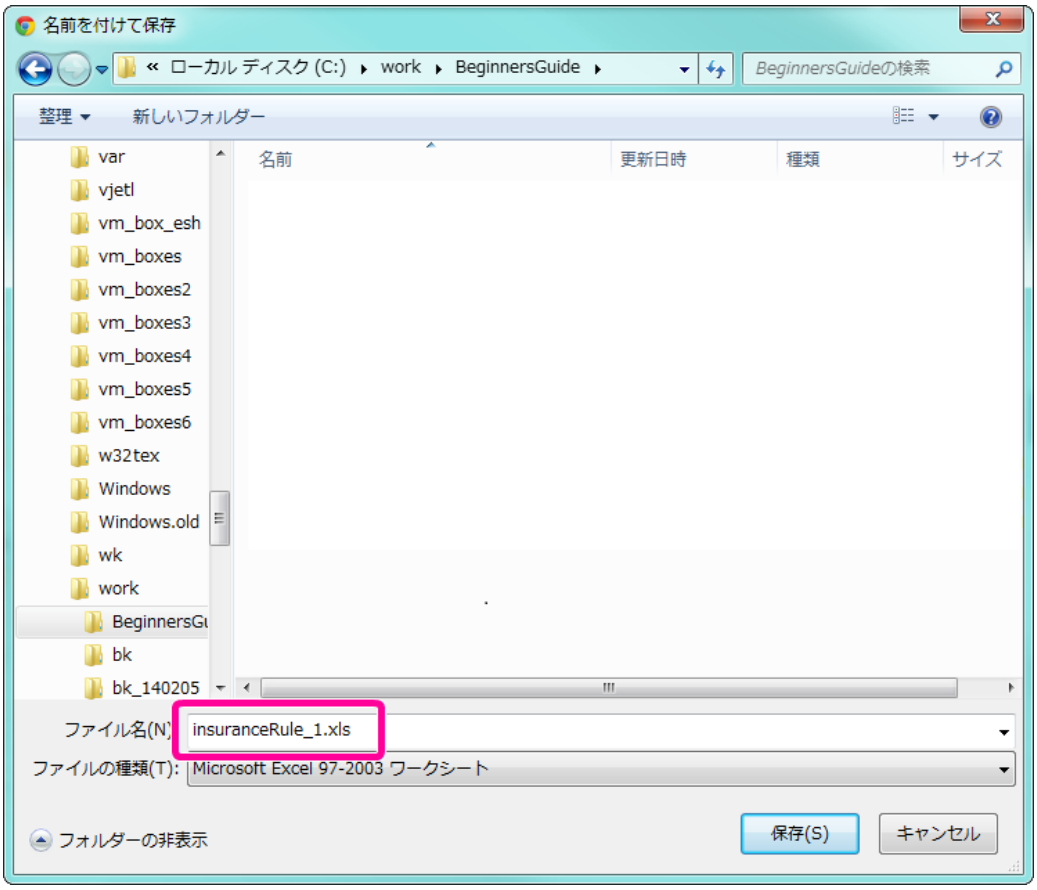
1. サイトマップの「 IM-BIS 」から「データソース定義」をクリックしてください。

2. 一覧からインポートしたデータソース定義の編集をクリックしてください。

3. データソース定義に付属しているExcelファイルの「ダウンロード」をクリックしてください。



4. ダウンロードしたファイルは、別名で保存してください。
   ここでは例として、「insuranceRule_1.xls」と設定して進めていきます。



5. これで、Excelファイルを編集する準備ができましたので、次の手順に進みます。

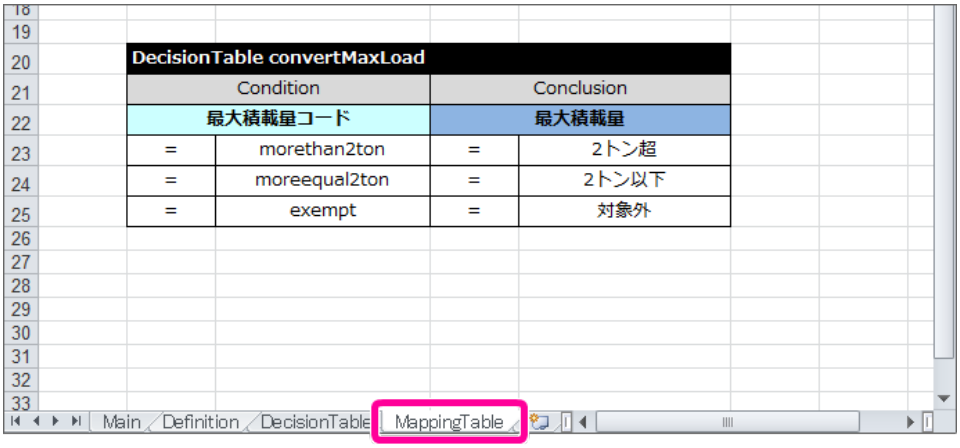
DecisionTable の条件となる値を確認する

今回は、Excelのルール定義ファイル上に、申請画面のラジオボタンの送信値と *DecisionTable* に記述するための値（論理名）のマッピングが定義されています。
DecisionTable の作成の前に、マッピングテーブルの内容を確認しましょう。

マッピングの *DecisionTable* の内容を確認する

条件の値に利用されているラジオボタンのマッピングの設定を確認しましょう。

1. Excelファイルの「MappingTable」シートを表示してください。



2. 車種、排気量、最大積載量に関する値のマッピングが定義されていることを確認することができます。
   これから作成する *DecisionTable* では、これらの値を条件に利用しますので、項目名と値を確認してください。
   画面（フォーム）のラジオボタンの設定との組み合わせは以下の通りです。
   - 左（Condition）の設定 → ラジオボタンの送信値
   - 右（Conclusion）の設定 → ラジオボタンの表示値

DecisionTable convertCarType

Condition		Conclusion	
車種コード		車種	
=	ordinary	=	普通自動車
=	light	=	軽自動車
=	truck	=	トラック
=	bike	=	バイク

DecisionTable convertDisplacement

Condition		Conclusion	
排気量コード		排気量	
=	small	=	125cc
=	medium	=	125cc-250cc
=	large	=	250cc
=	exempt	=	対象外

DecisionTable convertMaxLoad

Condition		Conclusion	
最大積載量コード		最大積載量	
=	morethan2ton	=	2トン超
=	moreequal2ton	=	2トン以下
=	exempt	=	対象外

各表のマッピング対象は以下の通りです。

a. 車種のマッピングに利用する *DecisionTable*

b. 排気量のマッピングに利用する *DecisionTable*

c. 最大積載量のマッピングに利用する *DecisionTable*

3. 確認した値に基づいて、次の手順から *DecisionTable* を作成していきましょう。

保険対象の内容のチェックと保険料を計算するための DecisionTable を作成する

最初に、申請書に入力された保険の対象に関する情報に基づいて自動車保険の保険料を計算する *DecisionTable* を作成しましょう。
ダウンロードしたExcelのルール定義ファイルには、 *Data/Variable* や *Decision* などは定義されていますが、 *DecisionTable* がありませんので、ハンズオンでは、この *DecisionTable* を作成します。

以下の図のようにExcel上にまとめます。

DecisionTable executeDecision

Condition		Condition		Condition		Condition		Conclusion		Conclusion	
車種		最大積載量		排気量		保険期間		メッセージ		保険料	
=	普通自動車					=	12	=	保険料を計算しました。	=	16400
=	普通自動車					=	13	=	保険料を計算しました。	=	17300
=	トラック							=	入力内容に誤りがあります。	=	0
=	バイク							=	入力内容に誤りがあります。	=	0

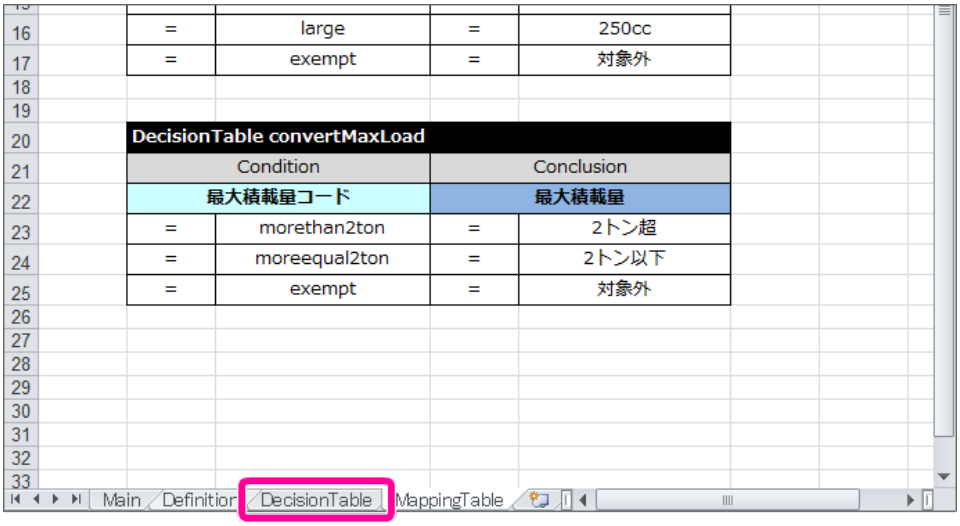
A. **正常パターン（保険料を計算）**
業務上の入力値の組み合わせとして、正しい組み合わせの場合には、メッセージと対応する保険料を返却するようにします。

B. **エラーパターン（入力チェック）**
入力値の組み合わせで、業務上誤った組み合わせに対しても、メッセージと保険料を返却します。
IM-BIS との連携時にメッセージ内容を利用した入力チェックを設定し、正しくない場合には申請を行えないように設定します。

DecisionTable の項目を設定する

DecisionTable の列を必要な分だけ追加し、条件と評価の項目名を設定していきましょう。

1. 編集中のExcelファイルの「DecisionTable」シートを表示してください。



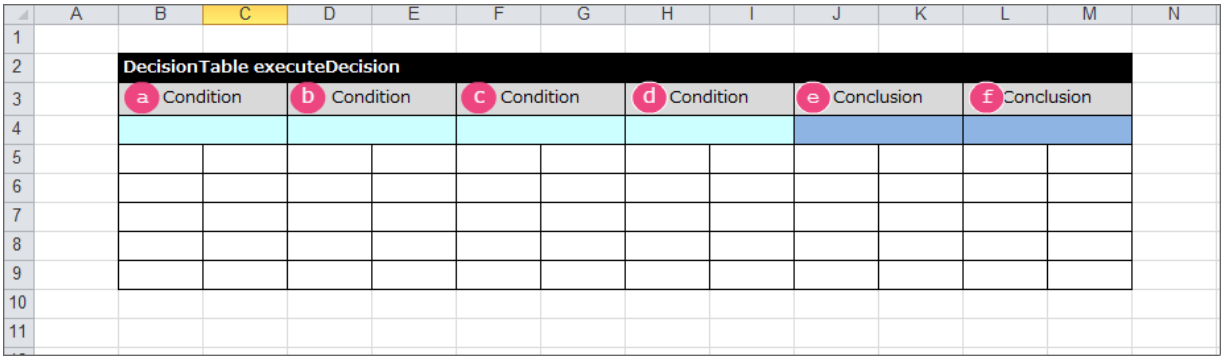
16	=	large	=	250cc
17	=	exempt	=	対象外

DecisionTable convertMaxLoad

Condition		Conclusion	
最大積載量コード		**最大積載量**	
=	morethan2ton	=	2トン超
=	moreequal2ton	=	2トン以下
=	exempt	=	対象外

Main / Definition / DecisionTable / MappingTable

2. 今回作成する *DecisionTable* にあわせた表が記載されていますので、*Condition* と *Conclusion* のキーワードと項目名（論理名）を設定してください。

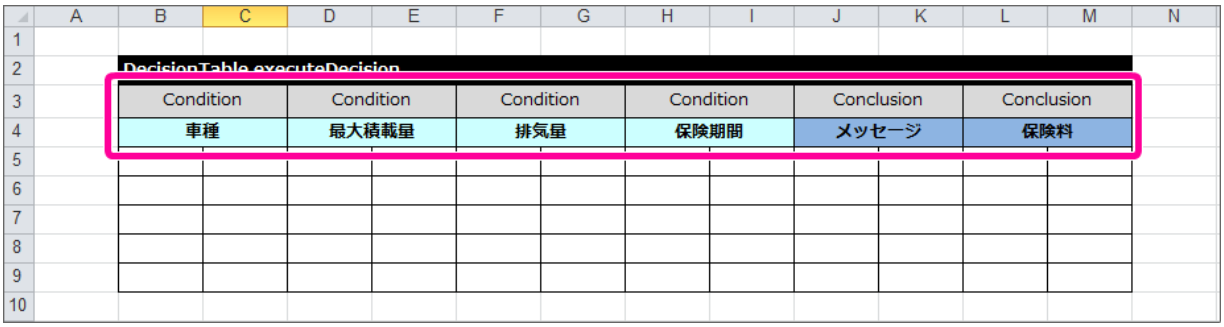


DecisionTable executeDecision

a Condition	b Condition	c Condition	d Condition	e Conclusion	f Conclusion

a	b	c	d	e	f
Condition	Condition	Condition	Condition	Conclusion	Conclusion
車種	最大積載量	排気量	保険期間	メッセージ	保険料

3. サブヘッダのキーワード、項目名（論理名）が以下のように設定できたら、一度ファイルを保存します。



DecisionTable executeDecision

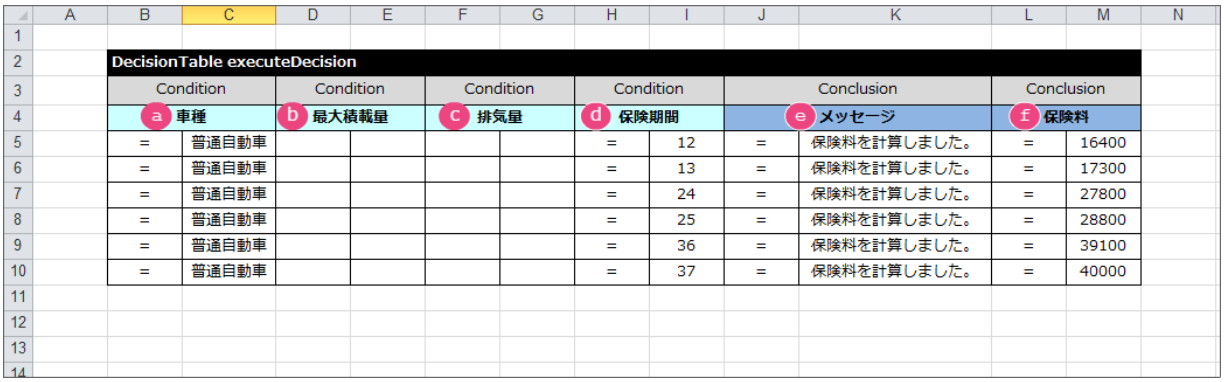
Condition	Condition	Condition	Condition	Conclusion	Conclusion
車種	**最大積載量**	**排気量**	**保険期間**	**メッセージ**	**保険料**

4. これで、 *DecisionTable* の項目が設定できましたので、各行に条件と評価を入力していきましょう。

DecisionTable の正常パターンの条件・評価を設定する

先の手順で設定した *DecisionTable* に保険料が正しく計算されるパターンの条件と評価（結果）を設定していきましょう。

1. 車種が「普通自動車」の保険料が計算されるパターンの条件と評価（メッセージ・保険料）を設定します。
   「普通自動車」では、最大積載量や排気量は条件に含まれないため、空欄（無条件）とし、以下のように設定してください。



DecisionTable executeDecision

Condition	Condition	Condition	Condition	Conclusion	Conclusion
a 車種	b 最大積載量	c 排気量	d 保険期間	e メッセージ	f 保険料
= 普通自動車			= 12	= 保険料を計算しました。	= 16400
= 普通自動車			= 13	= 保険料を計算しました。	= 17300
= 普通自動車			= 24	= 保険料を計算しました。	= 27800
= 普通自動車			= 25	= 保険料を計算しました。	= 28800
= 普通自動車			= 36	= 保険料を計算しました。	= 39100
= 普通自動車			= 37	= 保険料を計算しました。	= 40000

a	b	c	d	e	f
Condition	**Condition**	**Condition**	**Condition**	**Conclusion**	**Conclusion**
車種	最大積載量	排気量	保険期間	メッセージ	保険料
= 普通自動車			= 12	= 保険料を計算しました。	= 16400
= 普通自動車			= 13	= 保険料を計算しました。	= 17300
= 普通自動車			= 24	= 保険料を計算しました。	= 27800
= 普通自動車			= 25	= 保険料を計算しました。	= 28800
= 普通自動車			= 36	= 保険料を計算しました。	= 39100
= 普通自動車			= 37	= 保険料を計算しました。	= 40000

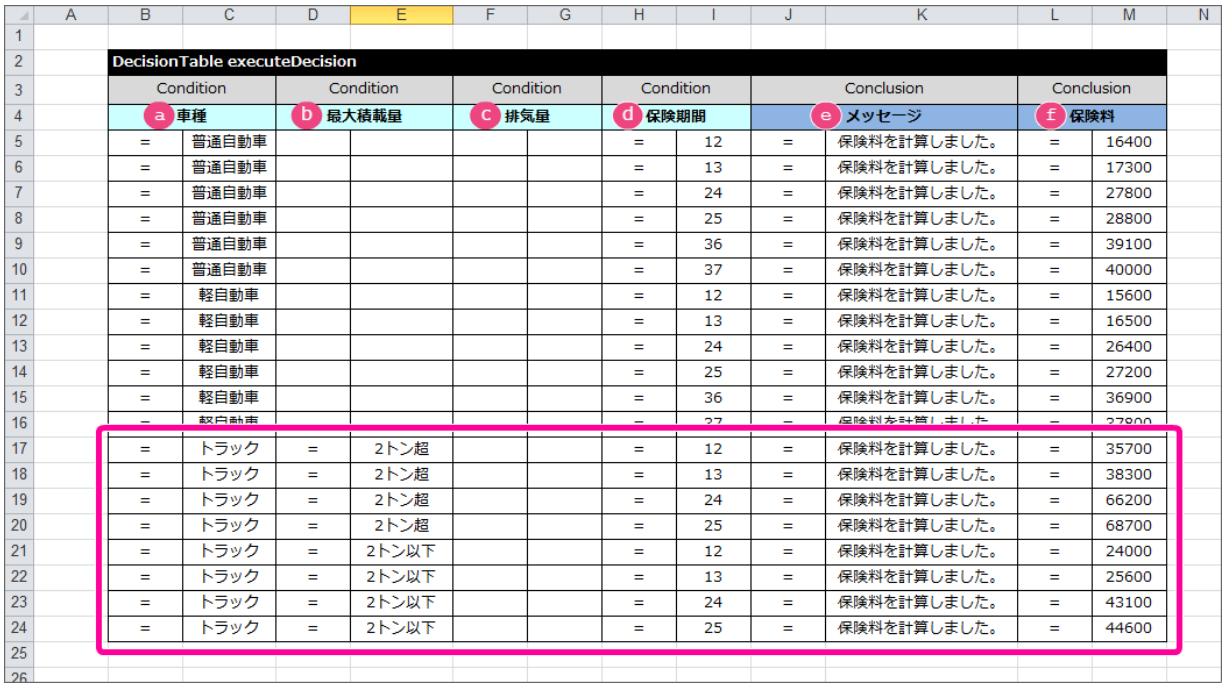
2. 車種が「軽自動車」の保険料が計算されるパターンの条件と評価（メッセージ・保険料）を設定します。
「軽自動車」も、最大積載量や排気量は条件に含まれないため、空欄（無条件）とし、普通自動車と同様に以下のように設定してください。

DecisionTable executeDecision

Condition	Condition	Condition	Condition	Conclusion	Conclusion
a 車種	b 最大積載量	c 排気量	d 保険期間	e メッセージ	f 保険料
= 普通自動車			= 12	= 保険料を計算しました。	= 16400
= 普通自動車			= 13	= 保険料を計算しました。	= 17300
= 普通自動車			= 24	= 保険料を計算しました。	= 27800
= 普通自動車			= 25	= 保険料を計算しました。	= 28800
= 普通自動車			= 36	= 保険料を計算しました。	= 39100
= 普通自動車			= 37	= 保険料を計算しました。	= 40000
= 軽自動車			= 12	= 保険料を計算しました。	= 15600
= 軽自動車			= 13	= 保険料を計算しました。	= 16500
= 軽自動車			= 24	= 保険料を計算しました。	= 26400
= 軽自動車			= 25	= 保険料を計算しました。	= 27200
= 軽自動車			= 36	= 保険料を計算しました。	= 36900
= 軽自動車			= 37	= 保険料を計算しました。	= 37800

a	b	c	d	e	f
Condition	**Condition**	**Condition**	**Condition**	**Conclusion**	**Conclusion**
車種	最大積載量	排気量	保険期間	メッセージ	保険料
= 軽自動車			= 12	= 保険料を計算しました。	= 15600
= 軽自動車			= 13	= 保険料を計算しました。	= 16500
= 軽自動車			= 24	= 保険料を計算しました。	= 26400
= 軽自動車			= 25	= 保険料を計算しました。	= 27200
= 軽自動車			= 36	= 保険料を計算しました。	= 36900
= 軽自動車			= 37	= 保険料を計算しました。	= 37800

3. 車種が「トラック」の保険料が計算されるパターンの条件と評価（メッセージ・保険料）を設定します。
「トラック」では、追加条件として最大積載量が考慮されるため、排気量のみ空欄（無条件）とし、以下のように設定してください。

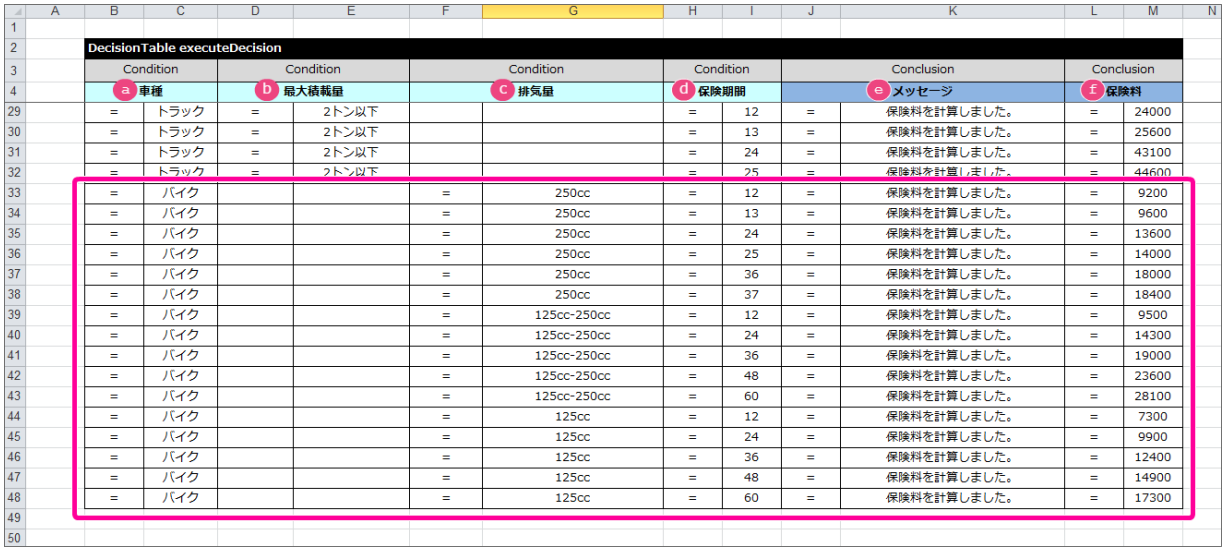


DecisionTable executeDecision

Condition	Condition	Condition	Condition	Conclusion	Conclusion
a 車種	b 最大積載量	c 排気量	d 保険期間	e メッセージ	f 保険料
= 普通自動車			= 12	= 保険料を計算しました。	= 16400
= 普通自動車			= 13	= 保険料を計算しました。	= 17300
= 普通自動車			= 24	= 保険料を計算しました。	= 27800
= 普通自動車			= 25	= 保険料を計算しました。	= 28800
= 普通自動車			= 36	= 保険料を計算しました。	= 39100
= 普通自動車			= 37	= 保険料を計算しました。	= 40000
= 軽自動車			= 12	= 保険料を計算しました。	= 15600
= 軽自動車			= 13	= 保険料を計算しました。	= 16500
= 軽自動車			= 24	= 保険料を計算しました。	= 26400
= 軽自動車			= 25	= 保険料を計算しました。	= 27200
= 軽自動車			= 36	= 保険料を計算しました。	= 36900
= トラック	= 2トン超		= 12	= 保険料を計算しました。	= 35700
= トラック	= 2トン超		= 13	= 保険料を計算しました。	= 38300
= トラック	= 2トン超		= 24	= 保険料を計算しました。	= 66200
= トラック	= 2トン超		= 25	= 保険料を計算しました。	= 68700
= トラック	= 2トン以下		= 12	= 保険料を計算しました。	= 24000
= トラック	= 2トン以下		= 13	= 保険料を計算しました。	= 25600
= トラック	= 2トン以下		= 24	= 保険料を計算しました。	= 43100
= トラック	= 2トン以下		= 25	= 保険料を計算しました。	= 44600

a	b	c	d	e	f
Condition	**Condition**	**Condition**	**Condition**	**Conclusion**	**Conclusion**
車種	最大積載量	排気量	保険期間	メッセージ	保険料
= トラック	= 2トン超		= 12	= 保険料を計算しました。	= 35700
= トラック	= 2トン超		= 13	= 保険料を計算しました。	= 38300
= トラック	= 2トン超		= 24	= 保険料を計算しました。	= 66200
= トラック	= 2トン超		= 25	= 保険料を計算しました。	= 68700
= トラック	= 2トン以下		= 12	= 保険料を計算しました。	= 24000
= トラック	= 2トン以下		= 13	= 保険料を計算しました。	= 25600
= トラック	= 2トン以下		= 24	= 保険料を計算しました。	= 43100
= トラック	= 2トン以下		= 25	= 保険料を計算しました。	= 44600

4. 車種が「バイク」の保険料が計算されるパターンの条件と評価（メッセージ・保険料）を設定します。
「バイク」では、追加条件として排気量が考慮されるため、最大積載量のみ空欄（無条件）とし、以下のように設定してください。



DecisionTable executeDecision

Condition	Condition	Condition	Condition	Conclusion	Conclusion
a 車種	b 最大積載量	c 排気量	d 保険期間	e メッセージ	f 保険料
= トラック	= 2トン以下		= 12	= 保険料を計算しました。	= 24000
= トラック	= 2トン以下		= 13	= 保険料を計算しました。	= 25600
= トラック	= 2トン以下		= 24	= 保険料を計算しました。	= 43100
= トラック	= 2トン以下		= 25	= 保険料を計算しました。	= 44600
= バイク		= 250cc	= 12	= 保険料を計算しました。	= 9200
= バイク		= 250cc	= 13	= 保険料を計算しました。	= 9600
= バイク		= 250cc	= 24	= 保険料を計算しました。	= 13600
= バイク		= 250cc	= 25	= 保険料を計算しました。	= 14000
= バイク		= 250cc	= 36	= 保険料を計算しました。	= 18000
= バイク		= 250cc	= 37	= 保険料を計算しました。	= 18400
= バイク		= 125cc-250cc	= 12	= 保険料を計算しました。	= 9500
= バイク		= 125cc-250cc	= 24	= 保険料を計算しました。	= 14300
= バイク		= 125cc-250cc	= 36	= 保険料を計算しました。	= 19000
= バイク		= 125cc-250cc	= 48	= 保険料を計算しました。	= 23600
= バイク		= 125cc-250cc	= 60	= 保険料を計算しました。	= 28100
= バイク		= 125cc	= 12	= 保険料を計算しました。	= 7300
= バイク		= 125cc	= 24	= 保険料を計算しました。	= 9900
= バイク		= 125cc	= 36	= 保険料を計算しました。	= 12400
= バイク		= 125cc	= 48	= 保険料を計算しました。	= 14900
= バイク		= 125cc	= 60	= 保険料を計算しました。	= 17300

a	b	c	d	e	f
Condition	**Condition**	**Condition**	**Condition**	**Conclusion**	**Conclusion**
車種	最大積載量	排気量	保険期間	メッセージ	保険料
= バイク		= 250cc	= 12	= 保険料を計算しました。	= 9200

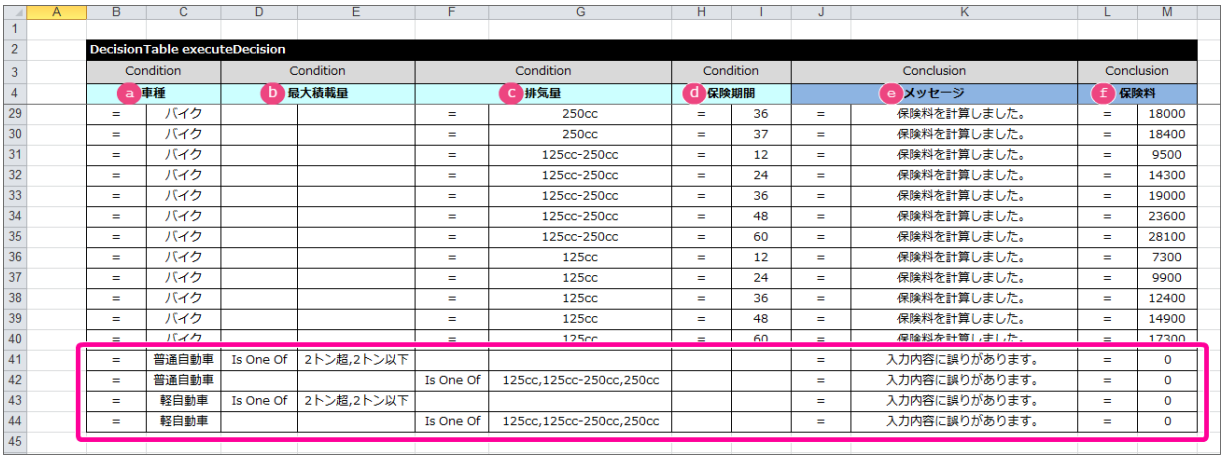
a	b	c	d	e	f
Condition	Condition	Condition	Condition	Conclusion	Conclusion
= バイク		= 250cc	= 13	= 保険料を計算しました。	= 9600
= バイク		= 250cc	= 24	= 保険料を計算しました。	= 13600
= バイク		= 250cc	= 25	= 保険料を計算しました。	= 14000
= バイク		= 250cc	= 36	= 保険料を計算しました。	= 18000
= バイク		= 250cc	= 37	= 保険料を計算しました。	= 18400
= バイク		= 125cc-250cc	= 12	= 保険料を計算しました。	= 9500
= バイク		= 125cc-250cc	= 24	= 保険料を計算しました。	= 14300
= バイク		= 125cc-250cc	= 36	= 保険料を計算しました。	= 19000
= バイク		= 125cc-250cc	= 48	= 保険料を計算しました。	= 23600
= バイク		= 125cc-250cc	= 60	= 保険料を計算しました。	= 28100
= バイク		= 125cc以下	= 12	= 保険料を計算しました。	= 7300
= バイク		= 125cc以下	= 24	= 保険料を計算しました。	= 9900
= バイク		= 125cc以下	= 36	= 保険料を計算しました。	= 12400
= バイク		= 125cc以下	= 48	= 保険料を計算しました。	= 14900
= バイク		= 125cc以下	= 60	= 保険料を計算しました。	= 17300

5. これで、 *DecisionTable* の保険料が計算されるパターンが作成できましたので、一度ファイルを保存しましょう。

DecisionTable のエラーパターンの条件・評価を設定する

続いて *DecisionTable* に保険料の一覧に保険料が設定されていない条件となった場合のエラーパターンの条件と評価（結果）を設定していきましょう。

1. エラーのパターンのひとつとして、車種に最大積載量や排気量が誤って設定された場合には、保険料を0とし、エラーメッセージを返却するようにします。
   このエラーを表現するためには、車種が「普通自動車」「軽自動車」の場合には、最大積載量や排気量に対象外以外の値がセットされている場合には、エラーと判断させるようにしましょう。
   条件の演算子には、「Is One Of」を利用して、最大積載量や排気量に設定される値をカンマ区切りで設定しましょう。

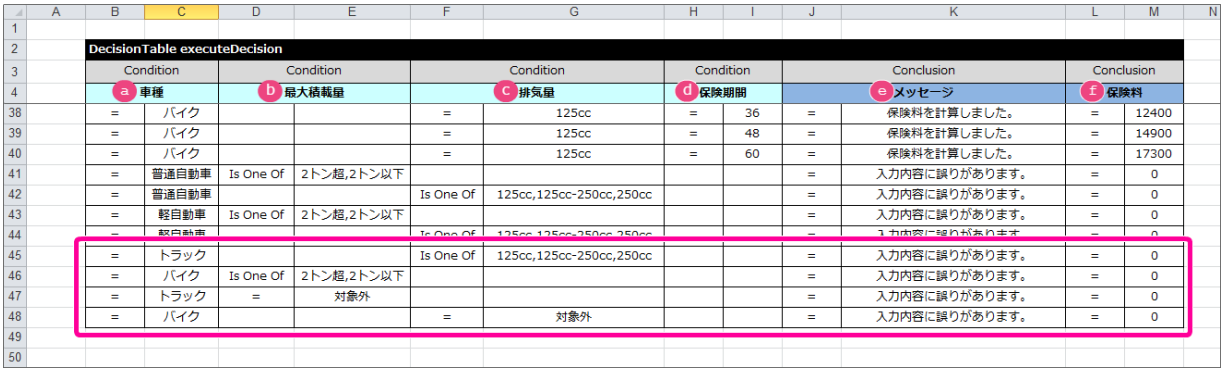


DecisionTable executeDecision

Condition	Condition	Condition	Condition	Conclusion	Conclusion
a 車種	b 最大積載量	c 排気量	d 保険期間	e メッセージ	f 保険料
= バイク		= 250cc	= 36	= 保険料を計算しました。	= 18000
= バイク		= 250cc	= 37	= 保険料を計算しました。	= 18400
= バイク		= 125cc-250cc	= 12	= 保険料を計算しました。	= 9500
= バイク		= 125cc-250cc	= 24	= 保険料を計算しました。	= 14300
= バイク		= 125cc-250cc	= 36	= 保険料を計算しました。	= 19000
= バイク		= 125cc-250cc	= 48	= 保険料を計算しました。	= 23600
= バイク		= 125cc-250cc	= 60	= 保険料を計算しました。	= 28100
= バイク		= 125cc	= 12	= 保険料を計算しました。	= 7300
= バイク		= 125cc	= 24	= 保険料を計算しました。	= 9900
= バイク		= 125cc	= 36	= 保険料を計算しました。	= 12400
= バイク		= 125cc	= 48	= 保険料を計算しました。	= 14900
= バイク		= 125cc	= 60	= 保険料を計算しました。	= 17300
= 普通自動車	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= 普通自動車		Is One Of	125cc,125cc-250cc,250cc	= 入力内容に誤りがあります。	= 0
= 軽自動車	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= 軽自動車		Is One Of	125cc,125cc-250cc,250cc	= 入力内容に誤りがあります。	= 0

a	b	c	d	e	f
Condition	Condition	Condition	Condition	Conclusion	Conclusion
車種	最大積載量	排気量	保険期間	メッセージ	保険料
= 普通自動車	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= 普通自動車		Is One Of 125cc,125cc-250cc,250cc		= 入力内容に誤りがあります。	= 0
= 軽自動車	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= 軽自動車		Is One Of 125cc,125cc-250cc,250cc		= 入力内容に誤りがあります。	= 0

2. 先に設定したように、トラックでは排気量、バイクでは、最大積載量が設定されている場合をエラーとします。
   普通自動車・軽自動車と同様に、演算子「Is One Of」を利用して、以下のように設定しましょう。
   また、トラックやバイクを選択されている場合には、必要な最大積載量や排気量の設定値が「対象外」となる場合にもエラーとする必要がありますので、合わせて設定してください。



DecisionTable executeDecision

Condition	Condition	Condition	Condition	Conclusion	Conclusion
a 車種	b 最大積載量	c 排気量	d 保険期間	e メッセージ	f 保険料
= バイク		= 125cc	= 36	= 保険料を計算しました。	= 12400
= バイク		= 125cc	= 48	= 保険料を計算しました。	= 14900
= バイク		= 125cc	= 60	= 保険料を計算しました。	= 17300
= 普通自動車	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= 普通自動車		Is One Of 125cc,125cc-250cc,250cc		= 入力内容に誤りがあります。	= 0
= 軽自動車	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= トラック		Is One Of 125cc,125cc-250cc,250cc		= 入力内容に誤りがあります。	= 0
= バイク	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= トラック	= 対象外			= 入力内容に誤りがあります。	= 0
= バイク		= 対象外		= 入力内容に誤りがあります。	= 0

a	b	c	d	e	f
Condition	**Condition**	**Condition**	**Condition**	**Conclusion**	**Conclusion**
車種	最大積載量	排気量	保険期間	メッセージ	保険料
= トラック		Is One Of 125cc,125cc-250cc,250cc		= 入力内容に誤りがあります。	= 0
= バイク	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= トラック	= 対象外			= 入力内容に誤りがあります。	= 0
= バイク		= 対象外		= 入力内容に誤りがあります。	= 0

3. エラーのパターンとして、車種ごとに決められた保険期間でない期間が設定された場合には、保険料を0とし、エラーメッセージを返却するようにします。
   DecisionTable では、上から順番に条件を評価しますので、車種が「普通自動車」で先に設定したパターンのいずれにも合致しない場合、と記述することで表現できます。
   DecisionTable の最後の行に、各車種の名前を条件にして、保険料とエラーメッセージを以下のように設定してください。

DecisionTable executeDecision

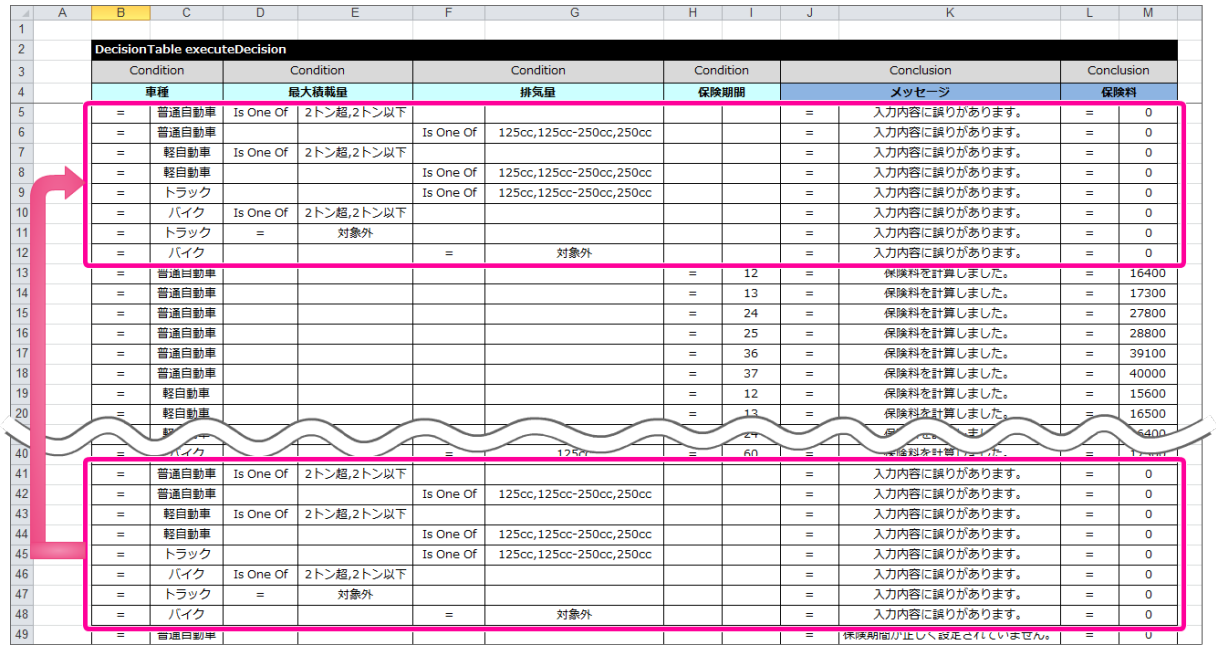
Condition	Condition	Condition	Condition	Conclusion	Conclusion
a 車種	b 最大積載量	c 排気量	d 保険期間	e メッセージ	f 保険料
= 普通自動車	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= 普通自動車		Is One Of 125cc,125cc-250cc,250cc		= 入力内容に誤りがあります。	= 0
= 軽自動車	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= 軽自動車		Is One Of 125cc,125cc-250cc,250cc		= 入力内容に誤りがあります。	= 0
= トラック		Is One Of 125cc,125cc-250cc,250cc		= 入力内容に誤りがあります。	= 0
= バイク	Is One Of 2トン超,2トン以下			= 入力内容に誤りがあります。	= 0
= トラック	= 対象外			= 入力内容に誤りがあります。	= 0
= バイク		= 対象外		= 入力内容に誤りがあります。	= 0
= 普通自動車				= 保険期間が正しく設定されていません。	= 0
= 軽自動車				= 保険期間が正しく設定されていません。	= 0
= トラック				= 保険期間が正しく設定されていません。	= 0
= バイク				= 保険期間が正しく設定されていません。	= 0

a	b	c	d	e	f
Condition	**Condition**	**Condition**	**Condition**	**Conclusion**	**Conclusion**
車種	最大積載量	排気量	保険期間	メッセージ	保険料
= 普通自動車				= 保険期間が正しく設定されていません。	= 0
= 軽自動車				= 保険期間が正しく設定されていません。ecli	= 0
= トラック				= 保険期間が正しく設定されていません。	= 0
= バイク				= 保険期間が正しく設定されていません。	= 0

DecisionTable の正常パターン、エラーパターンの評価順をコントロールする

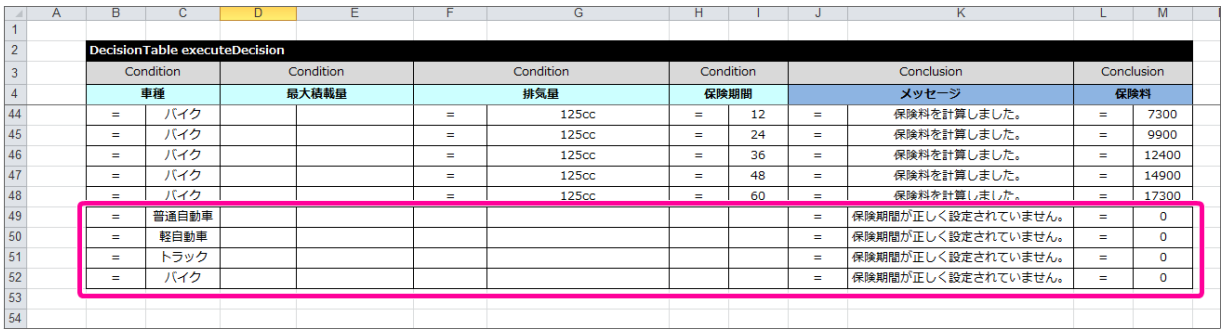
最後に *DecisionTable* に設定した正常パターン、エラーパターンの条件の評価の順序を考慮して、正しく評価が行われるように並び替えをしましょう。

1. OpenRules の *DecisionTable* はExcelの表を左上から右下に向かって評価を行います。
   車種「普通自動車」や「軽自動車」では、最大積載量や排気量が空欄（無条件）の設定となっているため、今の設定で実行した場合には、最大積載量や排気量に誤った値が設定されていてもエラーと判定しません。
   車種によって最大積載量や排気量に誤った値が設定されたときに、正しくエラーと評価できるように、「*DecisionTable* のエラーパターンの条件・評価を設定する」の手順1～2で作成した条件を1番上の行に移動してください。



DecisionTable executeDecision													
Condition		Condition		Condition		Condition			Conclusion		Conclusion		
車種		最大積載量		排気量		保険期間			メッセージ		保険料		
=	普通自動車	Is One Of	2トン超,2トン以下					=	入力内容に誤りがあります。	=	0		
=	普通自動車			Is One Of	125cc,125cc-250cc,250cc			=	入力内容に誤りがあります。	=	0		
=	軽自動車	Is One Of	2トン超,2トン以下					=	入力内容に誤りがあります。	=	0		
=	軽自動車			Is One Of	125cc,125cc-250cc,250cc			=	入力内容に誤りがあります。	=	0		
=	トラック			Is One Of	125cc,125cc-250cc,250cc			=	入力内容に誤りがあります。	=	0		
=	バイク	Is One Of	2トン超,2トン以下					=	入力内容に誤りがあります。	=	0		
=	トラック	=	対象外					=	入力内容に誤りがあります。	=	0		
=	バイク			=	対象外			=	入力内容に誤りがあります。	=	0		
=	普通自動車					=	12	=	保険料を計算しました。	=	16400		
=	普通自動車					=	13	=	保険料を計算しました。	=	17300		
=	普通自動車					=	24	=	保険料を計算しました。	=	27800		
=	普通自動車					=	25	=	保険料を計算しました。	=	28800		
=	普通自動車					=	36	=	保険料を計算しました。	=	39100		
=	普通自動車					=	37	=	保険料を計算しました。	=	40000		
=	軽自動車					=	12	=	保険料を計算しました。	=	15600		
=	軽自動車					=	13	=	保険料を計算しました。	=	16500		
=	軽自動車					=	24	=	保険料を計算しました。	=	26400		
=	バイク			=	125cc	=	60	=	保険料を計算しました。	=	17300		
=	普通自動車	Is One Of	2トン超,2トン以下					=	入力内容に誤りがあります。	=	0		
=	普通自動車			Is One Of	125cc,125cc-250cc,250cc			=	入力内容に誤りがあります。	=	0		
=	軽自動車	Is One Of	2トン超,2トン以下					=	入力内容に誤りがあります。	=	0		
=	軽自動車			Is One Of	125cc,125cc-250cc,250cc			=	入力内容に誤りがあります。	=	0		
=	トラック			Is One Of	125cc,125cc-250cc,250cc			=	入力内容に誤りがあります。	=	0		
=	バイク	Is One Of	2トン超,2トン以下					=	入力内容に誤りがあります。	=	0		
=	トラック	=	対象外					=	入力内容に誤りがあります。	=	0		
=	バイク			=	対象外			=	入力内容に誤りがあります。	=	0		
=	普通自動車							=	保険期間が正しく設定されていません。	=	0		

2. 保険期間の入力チェックについては、保険期間が正常パターンに記述されている条件のどれとも一致しない場合をエラーとします。
   そのため、正常パターンのどれにも一致しない場合に評価が行われるようにそのまま一番下に移動してください。



DecisionTable executeDecision											
Condition		Condition		Condition		Condition		Conclusion		Conclusion	
車種		最大積載量		排気量		保険期間		メッセージ		保険料	
=	バイク			=	125cc	=	12	=	保険料を計算しました。	=	7300
=	バイク			=	125cc	=	24	=	保険料を計算しました。	=	9900
=	バイク			=	125cc	=	36	=	保険料を計算しました。	=	12400
=	バイク			=	125cc	=	48	=	保険料を計算しました。	=	14900
=	バイク			=	125cc	=	60	=	保険料を計算しました。	=	17300
=	普通自動車							=	保険期間が正しく設定されていません。	=	0
=	軽自動車							=	保険期間が正しく設定されていません。	=	0
=	トラック							=	保険期間が正しく設定されていません。	=	0
=	バイク							=	保険期間が正しく設定されていません。	=	0

3. これで、Excelファイルの条件の評価順を正しく設定することができました。
   今回のハンズオンのExcelのルール定義ファイルには、その他に必要な定義は設定済みとなっているため、ファイルを保存してください。
   完成した *DecisionTable* は、以下の図の通りです。
   次の手順で IM-BIS の画面（フォーム）と連携するための設定を行っていきましょう。

	A	B	C	D	E	F	G	H	I	J	K	L	M	
1														
2		DecisionTable executeDecision												
3		Condition		Condition		Condition		Condition		Conclusion		Conclusion		
4		車種		最大積載量		排気量		保険期間		メッセージ		保険料		
5		=	普通自動車	Is One Of	2トン超,2トン以下					=	入力内容に誤りがあります。	=	0	
6		=	普通自動車			Is One Of	125cc,125cc-250cc,250cc			=	入力内容に誤りがあります。	=	0	
7		=	軽自動車	Is One Of	2トン超,2トン以下					=	入力内容に誤りがあります。	=	0	
8		=	軽自動車			Is One Of	125cc,125cc-250cc,250cc			=	入力内容に誤りがあります。	=	0	
9		=	トラック			Is One Of	125cc,125cc-250cc,250cc			=	入力内容に誤りがあります。	=	0	
10		=	バイク	Is One Of	2トン超,2トン以下					=	入力内容に誤りがあります。	=	0	
11		=	トラック	=	対象外					=	入力内容に誤りがあります。	=	0	
12		=	バイク			=	対象外			=	入力内容に誤りがあります。	=	0	
13		=	普通自動車					=	12	=	保険料を計算しました。	=	16400	
14		=	普通自動車					=	13	=	保険料を計算しました。	=	17300	
15		=	普通自動車					=	24	=	保険料を計算しました。	=	27800	
16		=	普通自動車					=	25	=	保険料を計算しました。	=	28800	
17		=	普通自動車					=	36	=	保険料を計算しました。	=	39100	
18		=	普通自動車					=	37	=	保険料を計算しました。	=	40000	
19		=	軽自動車					=	12	=	保険料を計算しました。	=	15600	
20		=	軽自動車					=	13	=	保険料を計算しました。	=	16500	
21		=	軽自動車					=	24	=	保険料を計算しました。	=	26400	
22		=	軽自動車					=	25	=	保険料を計算しました。	=	27200	
23		=	軽自動車					=	36	=	保険料を計算しました。	=	36900	
24		=	軽自動車					=	37	=	保険料を計算しました。	=	37800	
25		=	トラック	=	2トン超			=	12	=	保険料を計算しました。	=	35700	
26		=	トラック	=	2トン超			=	13	=	保険料を計算しました。	=	38300	
27		=	トラック	=	2トン超			=	24	=	保険料を計算しました。	=	66200	
28		=	トラック	=	2トン超			=	25	=	保険料を計算しました。	=	68700	
29		=	トラック	=	2トン以下			=	12	=	保険料を計算しました。	=	24000	
30		=	トラック	=	2トン以下			=	13	=	保険料を計算しました。	=	25600	
31		=	トラック	=	2トン以下			=	24	=	保険料を計算しました。	=	43100	
32		=	トラック	=	2トン以下			=	25	=	保険料を計算しました。	=	44600	
33		=	バイク			=	250cc	=	12	=	保険料を計算しました。	=	9200	
34		=	バイク			=	250cc	=	13	=	保険料を計算しました。	=	9600	
35		=	バイク			=	250cc	=	24	=	保険料を計算しました。	=	13600	
36		=	バイク			=	250cc	=	25	=	保険料を計算しました。	=	14000	
37		=	バイク			=	250cc	=	36	=	保険料を計算しました。	=	18000	
38		=	バイク			=	250cc	=	37	=	保険料を計算しました。	=	18400	
39		=	バイク			=	125cc-250cc	=	12	=	保険料を計算しました。	=	9500	
40		=	バイク			=	125cc-250cc	=	24	=	保険料を計算しました。	=	14300	
41		=	バイク			=	125cc-250cc	=	36	=	保険料を計算しました。	=	19000	
42		=	バイク			=	125cc-250cc	=	48	=	保険料を計算しました。	=	23600	
43		=	バイク			=	125cc-250cc	=	60	=	保険料を計算しました。	=	28100	
44		=	バイク			=	125cc	=	12	=	保険料を計算しました。	=	7300	
45		=	バイク			=	125cc	=	24	=	保険料を計算しました。	=	9900	
46		=	バイク			=	125cc	=	36	=	保険料を計算しました。	=	12400	
47		=	バイク			=	125cc	=	48	=	保険料を計算しました。	=	14900	
48		=	バイク			=	125cc	=	60	=	保険料を計算しました。	=	17300	
49		=	普通自動車							=	保険期間が正しく設定されていません。	=	0	
50		=	軽自動車							=	保険期間が正しく設定されていません。	=	0	
51		=	トラック							=	保険期間が正しく設定されていません。	=	0	
52		=	バイク							=	保険期間が正しく設定されていません。	=	0	
53														
54														

IM-BIS と連携したフローを作成する

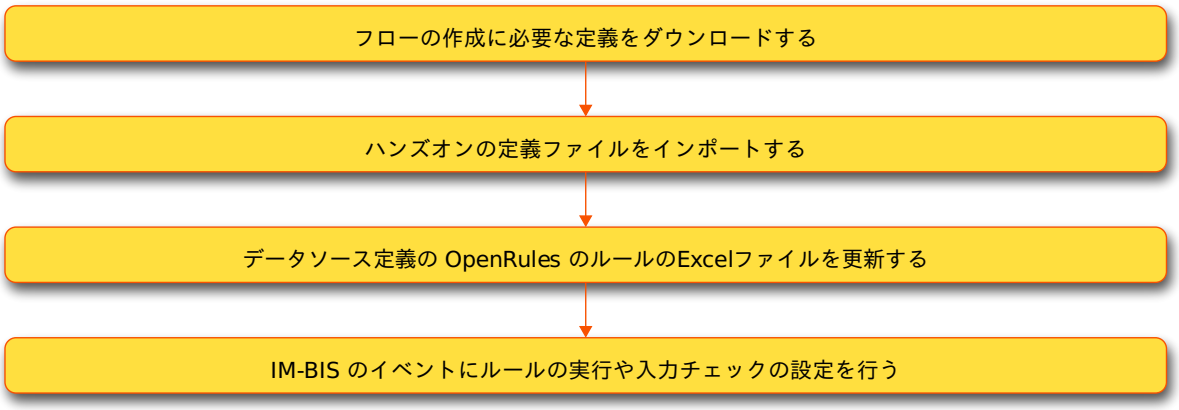
先の手順で作成したExcelのルール定義ファイルを IM-BIS と連携するためのフローの作成を進めていきましょう。

ルールと連携したフローを作成する手順

- OpenRules と IM-BIS を連携するための手順
- フローの作成に必要な定義をダウンロードする
- ハンズオンの定義ファイルをインポートする
- データソース定義の OpenRules のルールのExcelファイルを更新する
- IM-BIS のイベントにルールの実行や入力チェックの設定を行う

OpenRules と IM-BIS を連携するための手順

この手順では、作成したExcelのルール定義ファイルをデータソース定義に登録し、 IM-BIS の画面アイテムのイベントに設定するまでの手順を確認していきます。



フローの作成に必要な定義をダウンロードする

ハンズオンで作成するフローのベースとなる各種定義ファイルをインポートします。
最初に下記のリンクからファイルをダウンロードしてください。
「 IM-Workflow 定義」のみダウンロード後に解凍してください。

- IM-Workflow 定義
  imw_auto_insurance.zip
- BIS定義
  bis_auto_insurance.zip
- Formaアプリケーション定義
  forma_auto_insurance.zip

ハンズオンの定義ファイルをインポートする

先の手順でダウンロードしたファイルを「 *各種定義ファイルのインポートの手順* 」に従ってインポートしてください。

データソース定義の OpenRules のルールのExcelファイルを更新する

データソース定義に添付されている OpenRules のルールのExcelファイルを更新しましょう。

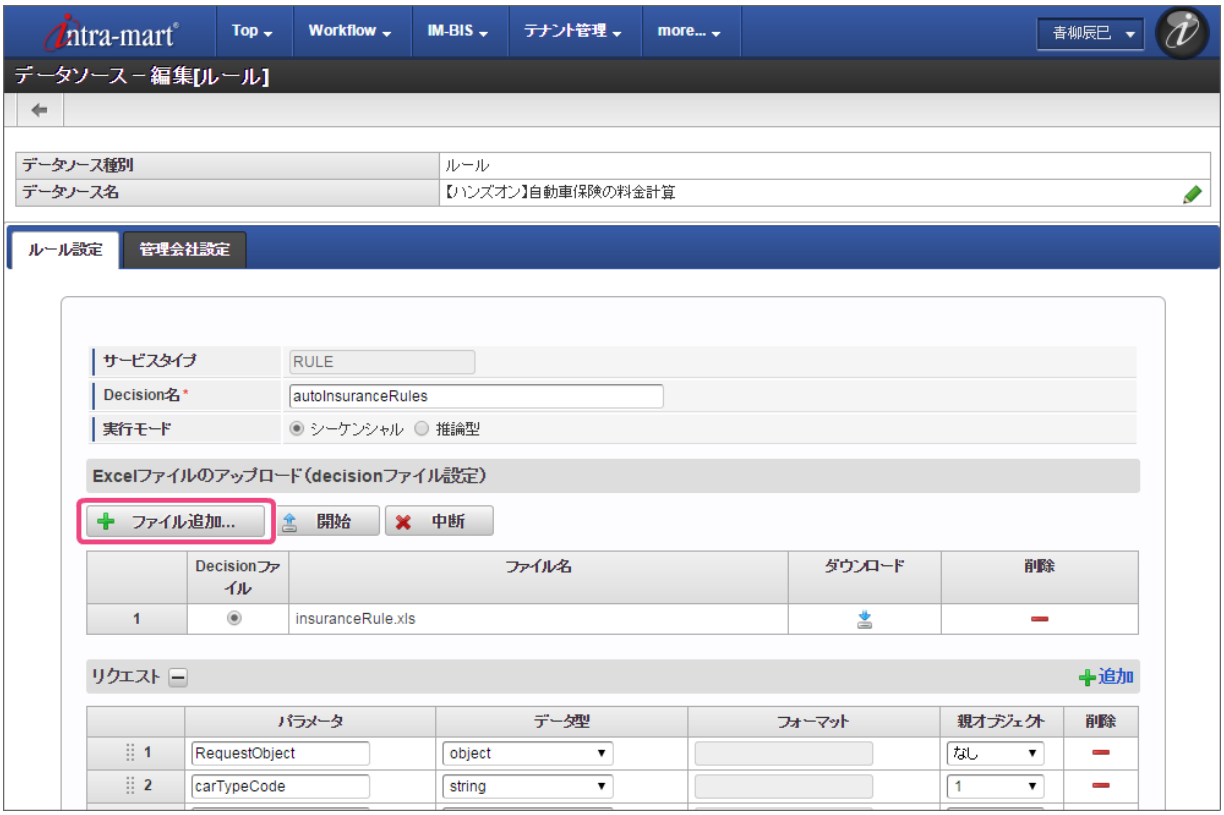
データソース定義のExcelファイルを更新する

データソース定義のExcelファイルを更新しましょう。

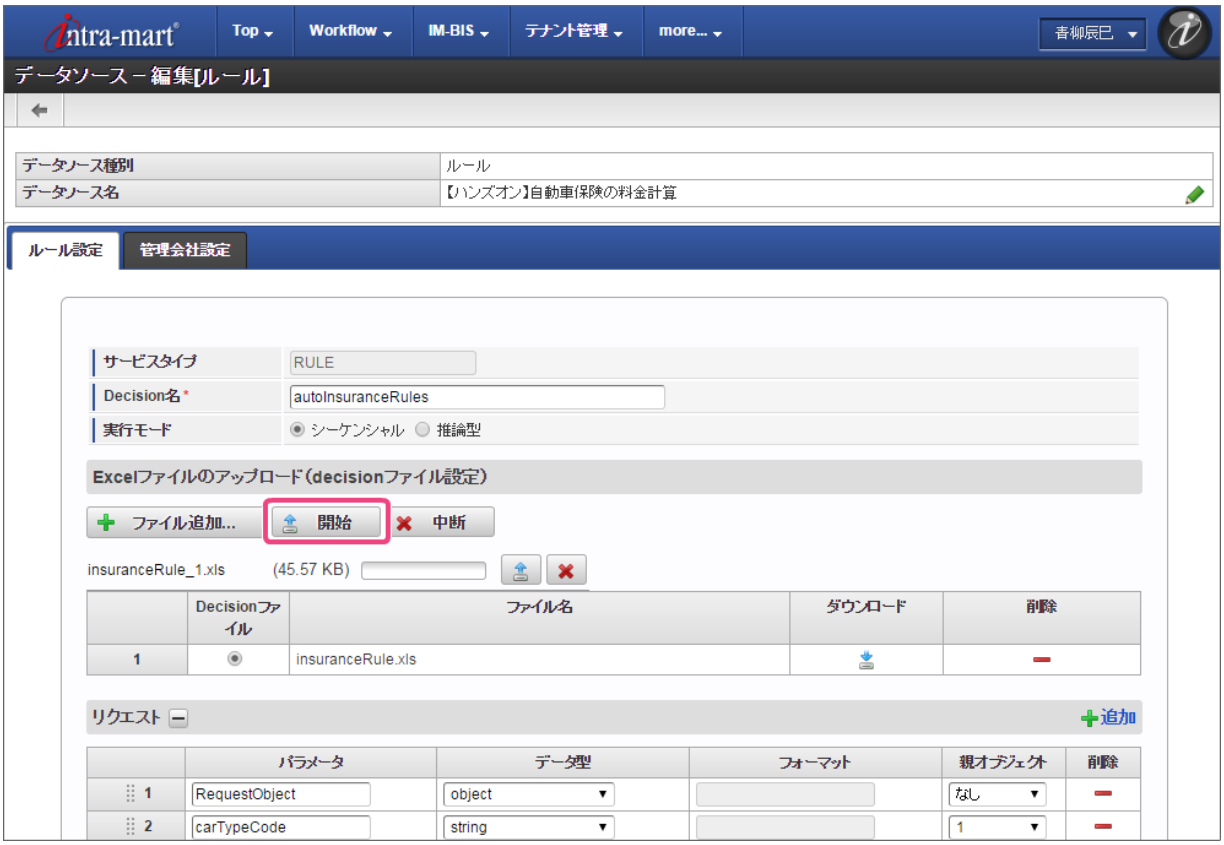
1. サイトマップの「 IM-BIS 」から「データソース定義」をクリックしてください。

2. インポートしたデータソース定義の「【ハンズオン】自動車保険の料金計算」の「 」をクリックしてください。

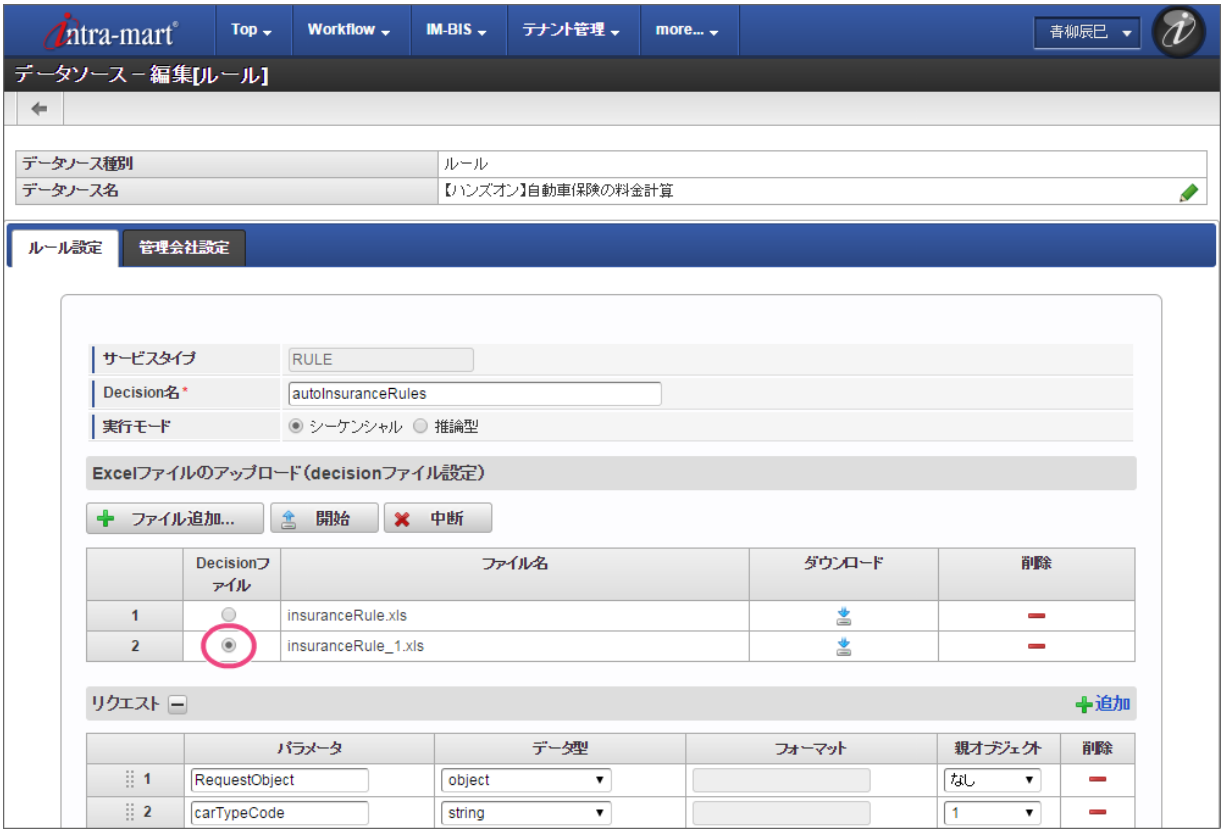
3. 「ファイル追加」をクリック後、作成したExcelのルール定義ファイル「insuranceRule_1.xls」を選択してください。



4. 「開始」をクリックして、ルール定義ファイルをアップロードしてください。



5. 追加したファイルの「Decisionファイル」のラジオボタンをクリックしてオンにしてください。



> **コラム**
> 登録済みのデータソース定義に含まれるルール定義ファイルは、このようにしてファイルを別名で保存し、Decisionファイルの設定を変更することで履歴保存することができます。

6. 「更新」をクリックしてデータソース定義の内容を保存してください。

7. これで、データソース定義のルール内容を更新できました。
   続いて、参照しているフローの設定を行います。

IM-BIS のイベントにルールの実行や入力チェックの設定を行う

更新したデータソース定義を利用して、 IM-BIS の画面にルールの実行や入力チェックを設定しましょう。

フォーム（画面）の編集を開始する

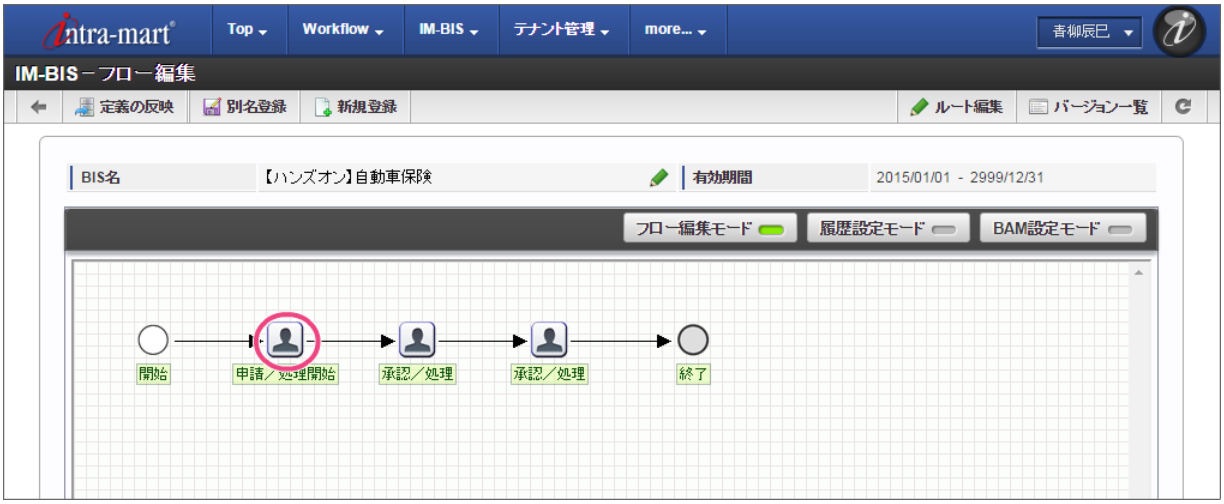
画面の設定を開始するために、フォーム（画面）の編集画面を表示しましょう。

1. サイトマップの「 IM-BIS 」をクリックしてください。

2. 「一覧」をクリックしてください。

3. インポートしたフローの「【ハンズオン】自動車保険」の をクリックしてください。

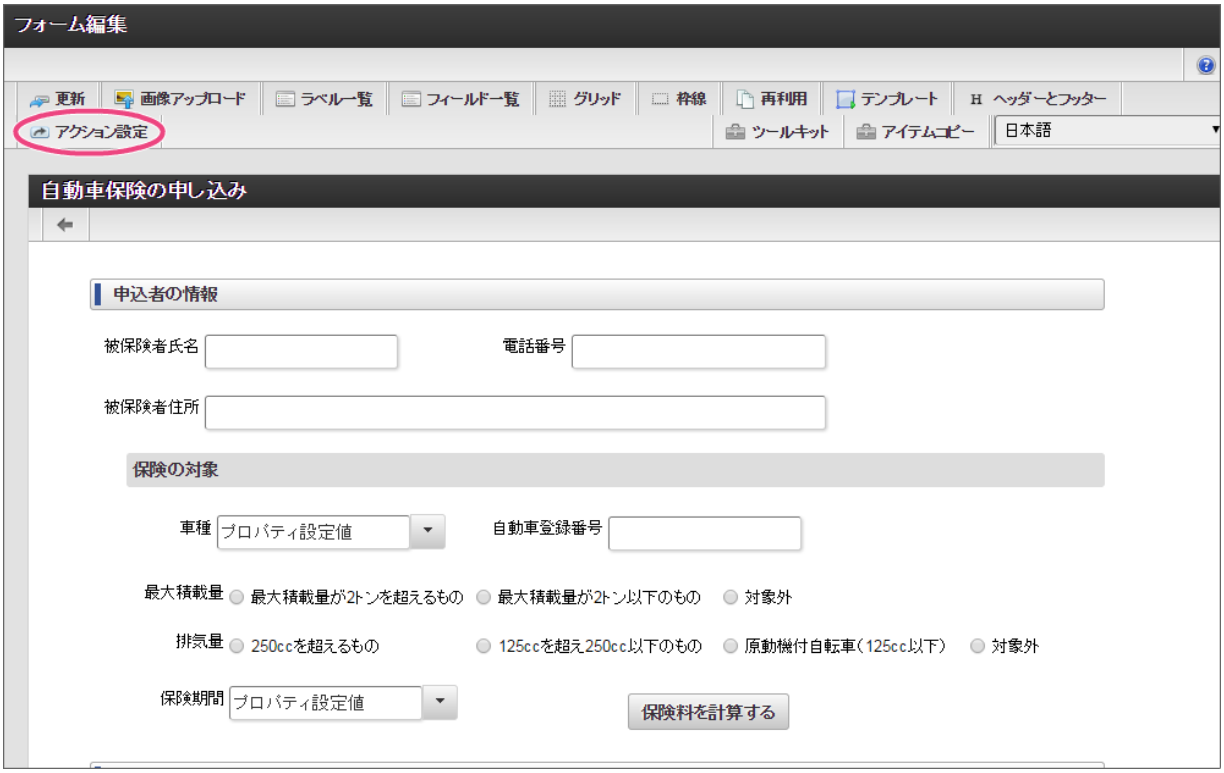
4. 「申請／処理開始」のアイコンをダブルクリックして、フォーム編集画面（フォーム・デザイナ）を表示してください。



画面のアクションイベントにルールの実行を設定する

フォーム（画面）の編集画面で、画面アイテムにルールを実行するイベントを設定しましょう。

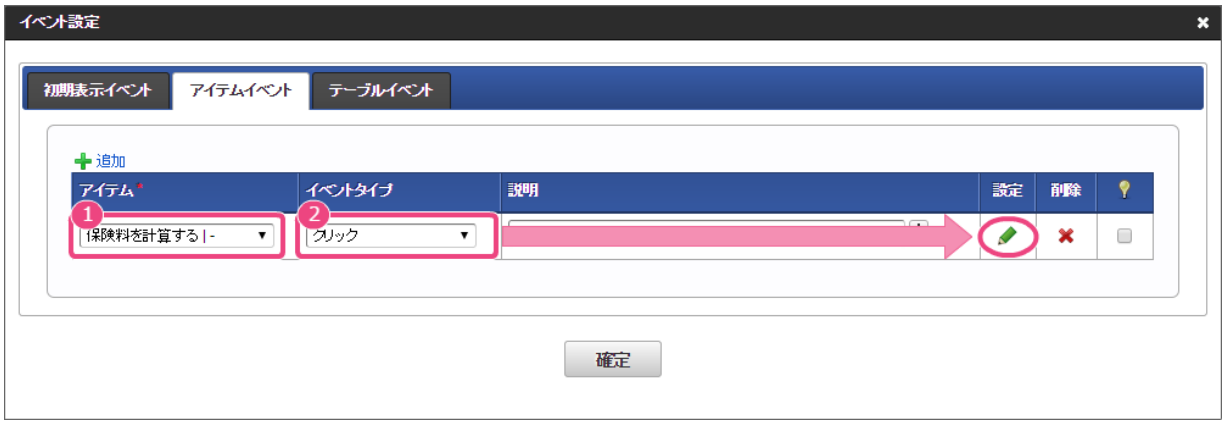
1. フォーム編集画面を表示したら「アクション設定」をクリックしてください。



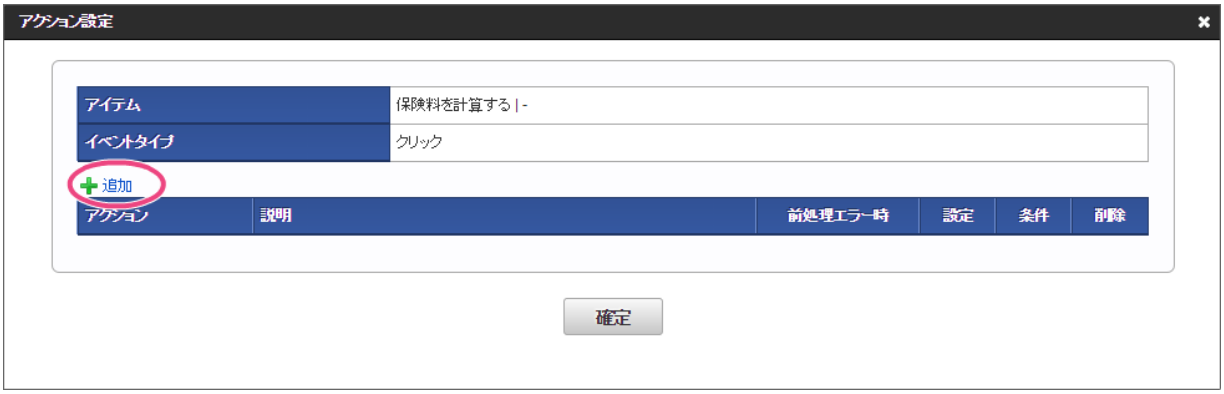
2. 「アイテムイベント」をクリックして、表示するタブを切り替えます。

3. 「追加」をクリックしてください。

4. アイテムとイベントタイプを以下のように変更し、「」をクリックしてください。
   - アイテム
     保険料を計算する | -（ボタン（イベント））
   - イベントタイプ
     クリック

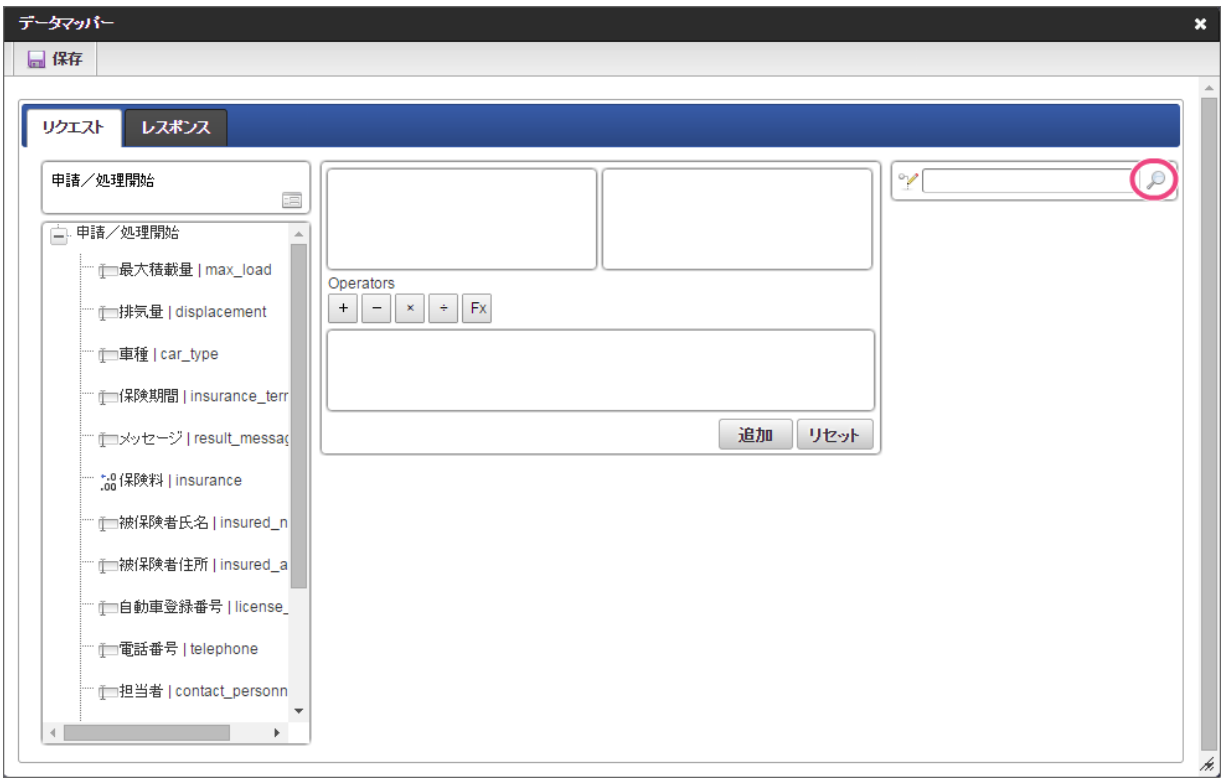


5. 「追加」をクリックしてください。

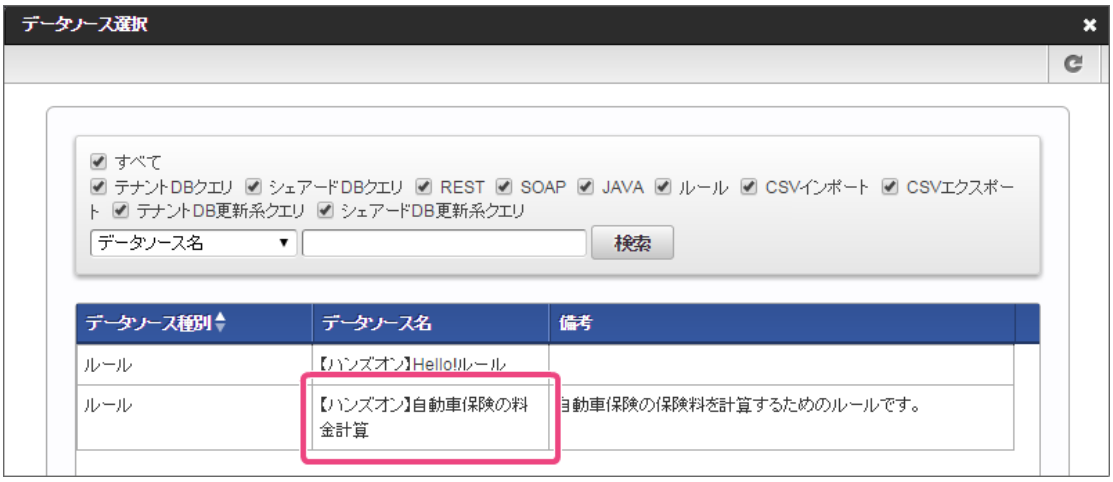


6. 「アクション」を「外部連携」にし、「」をクリックしてください。

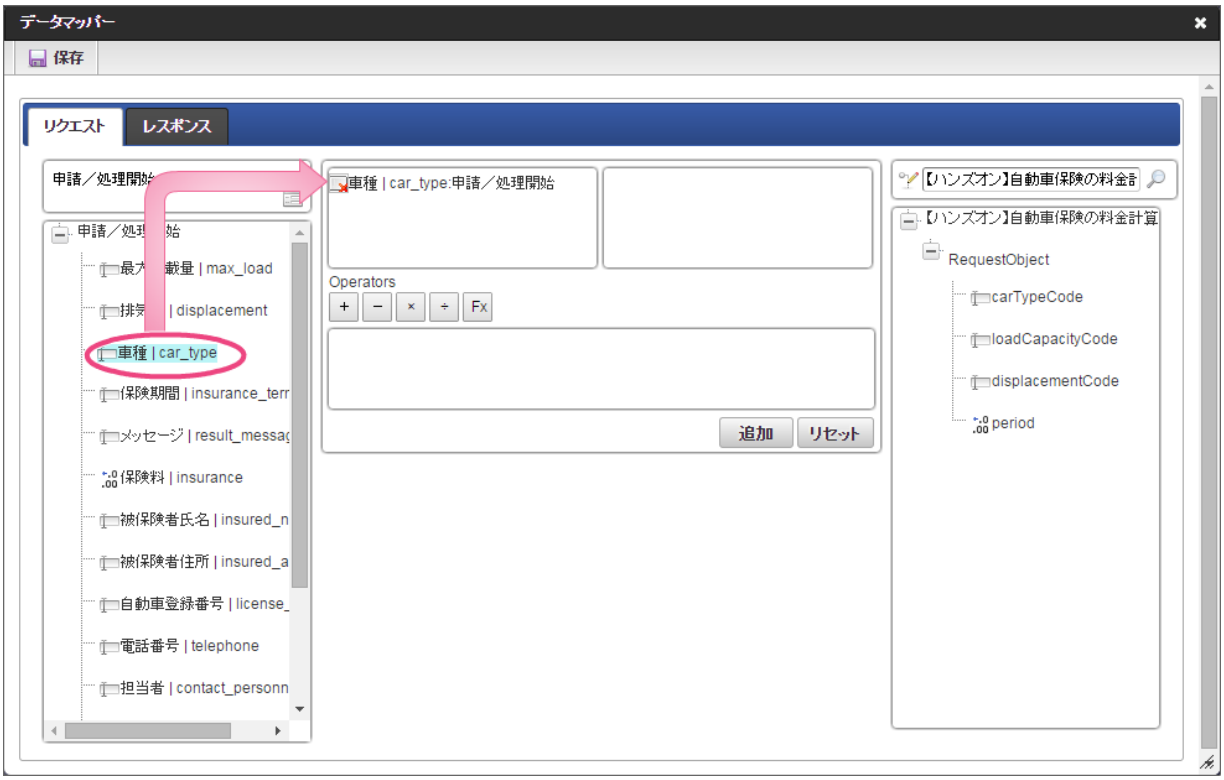
7. 「データマッパー」で右上の🔎をクリックしてください。



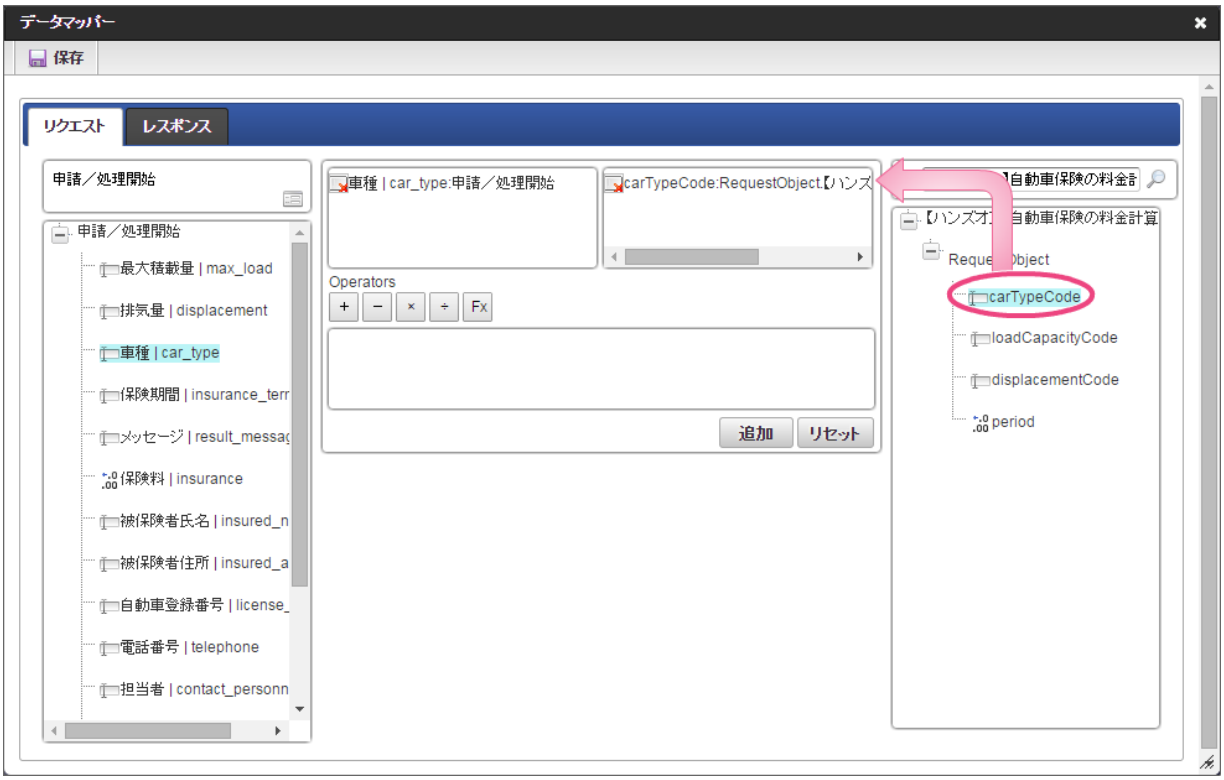
8. 登録したデータソース定義「【ハンズオン】自動車保険の料金計算」をクリックしてください。



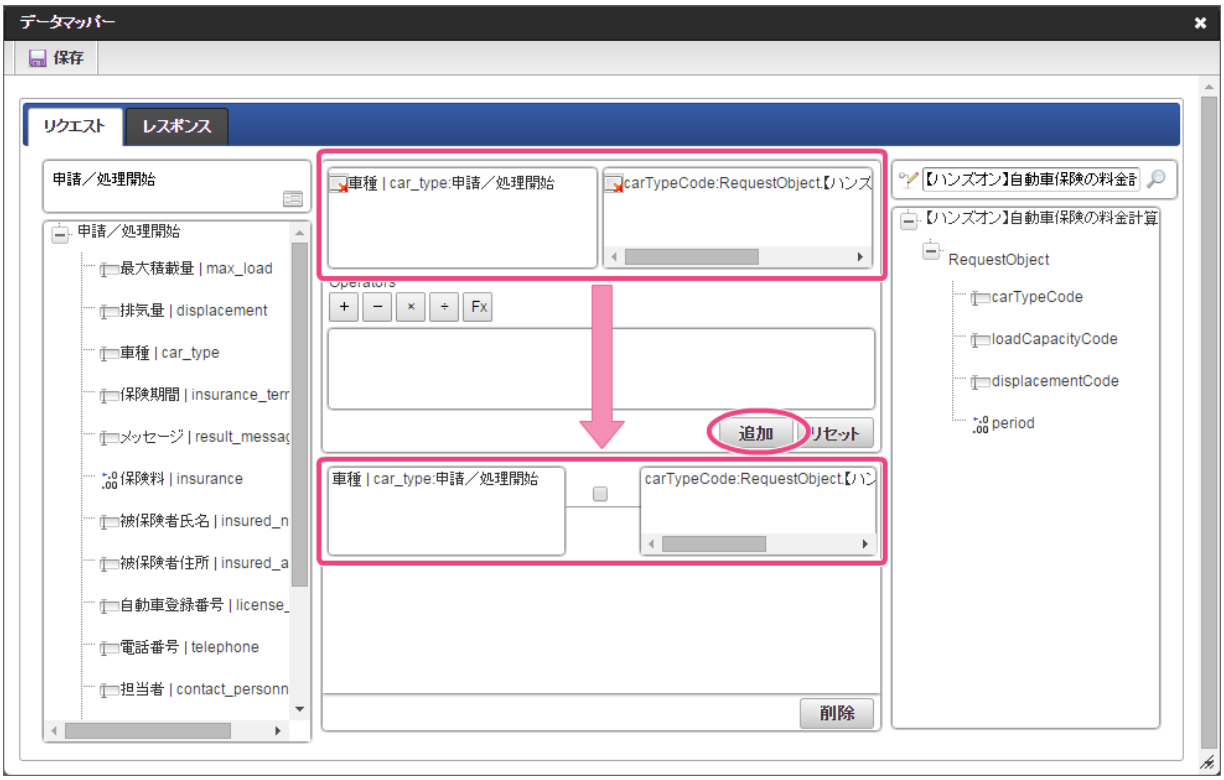
9. 左の欄から「車種」をクリックしてください。
   クリック後、中央左の欄に「車種 | car_type: 申請／処理開始」と表示されます。



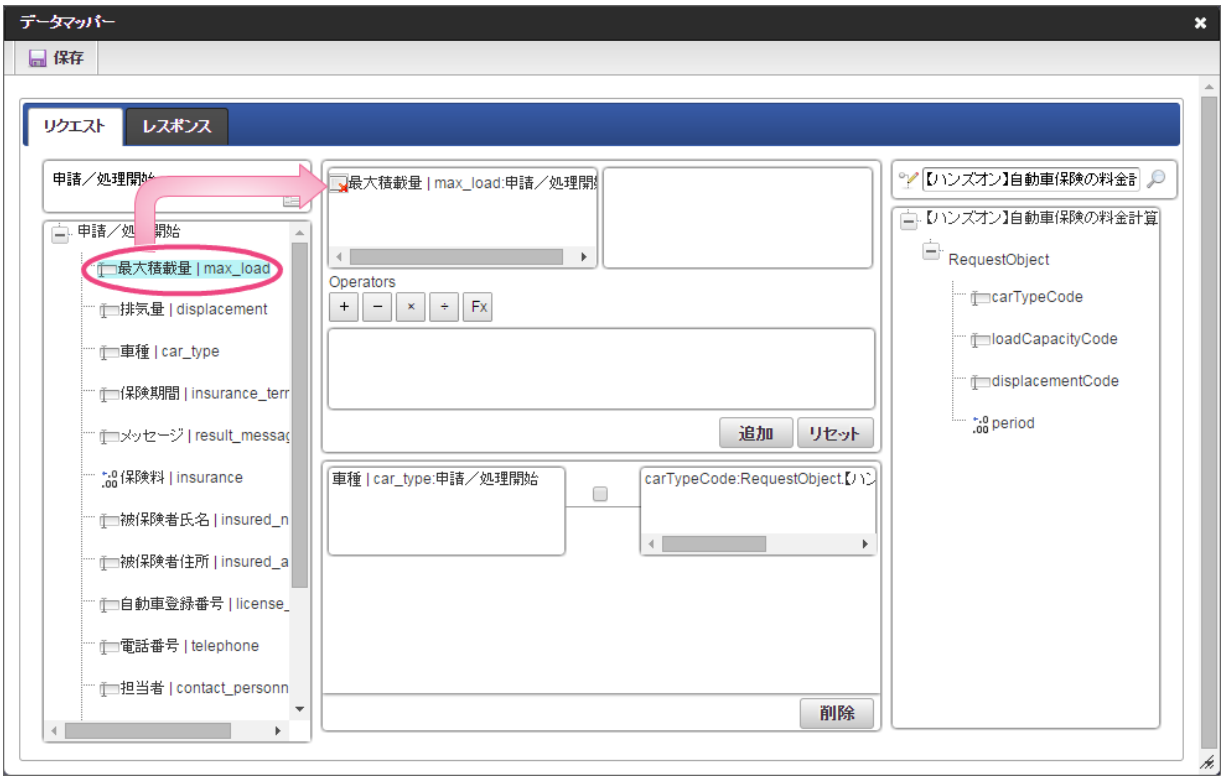
10. 右の欄から「carTypeCode」をクリックしてください。
クリック後、中央右の欄に「carTypeCode:RequestObject.【ハンズオン】自動車保険の料金計算」と表示されます。



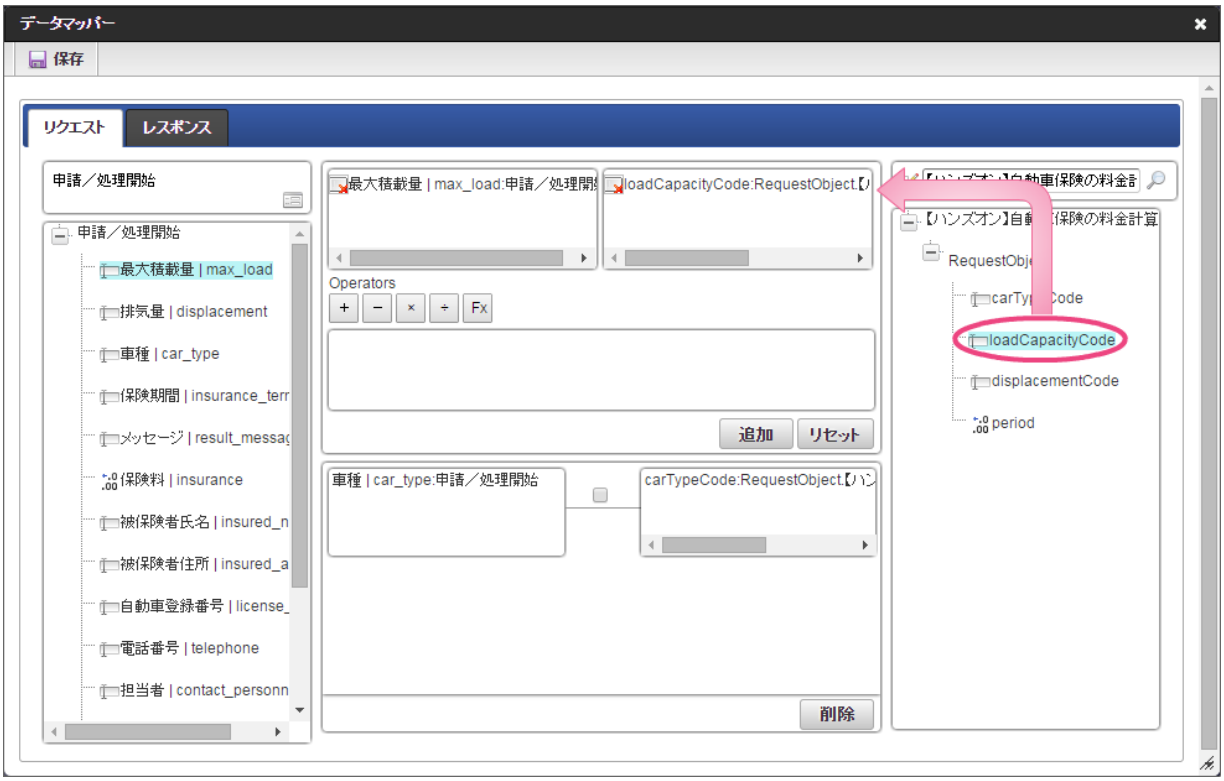
11. 「追加」をクリックしてください。
フォームの「車種」とデータソース定義の「carTypeCode」のマッピングが設定され、中央下段の欄に表示されます。



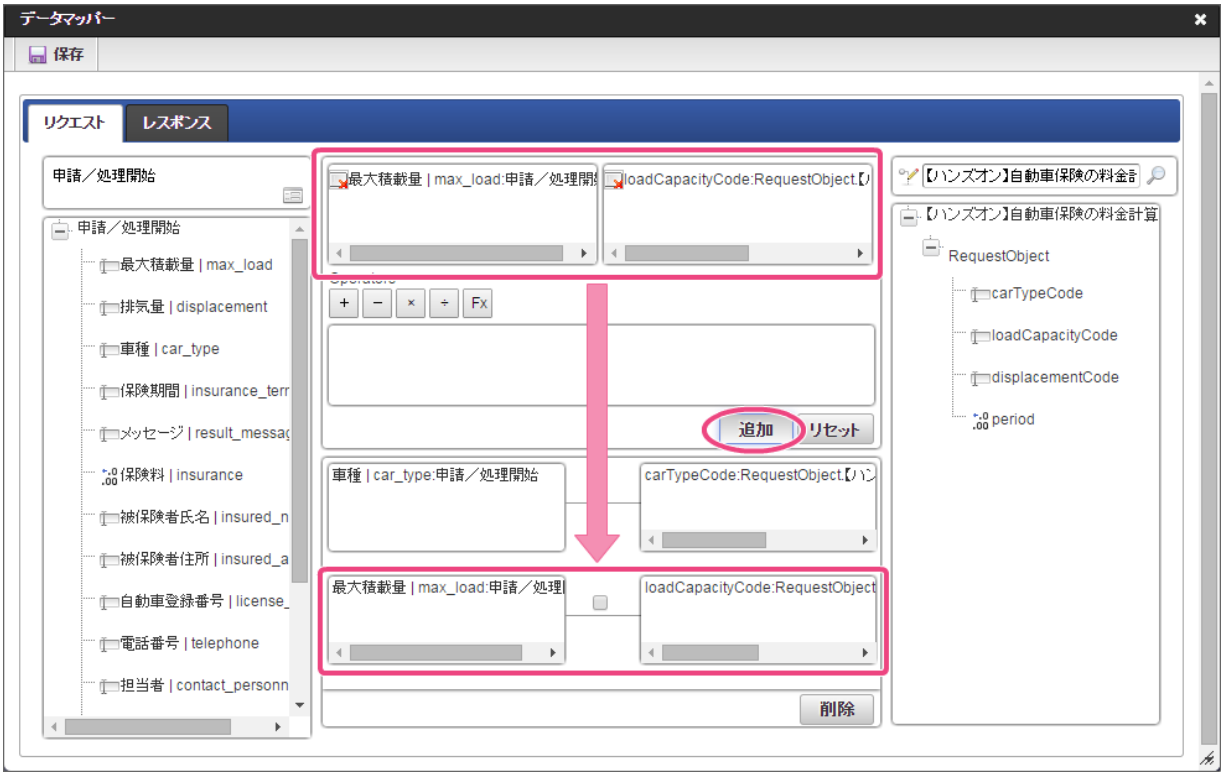
12. 左の欄から「最大積載量」をクリックしてください。
クリック後、中央左の欄に「最大積載量 | max_load：申請／処理開始」と表示されます。



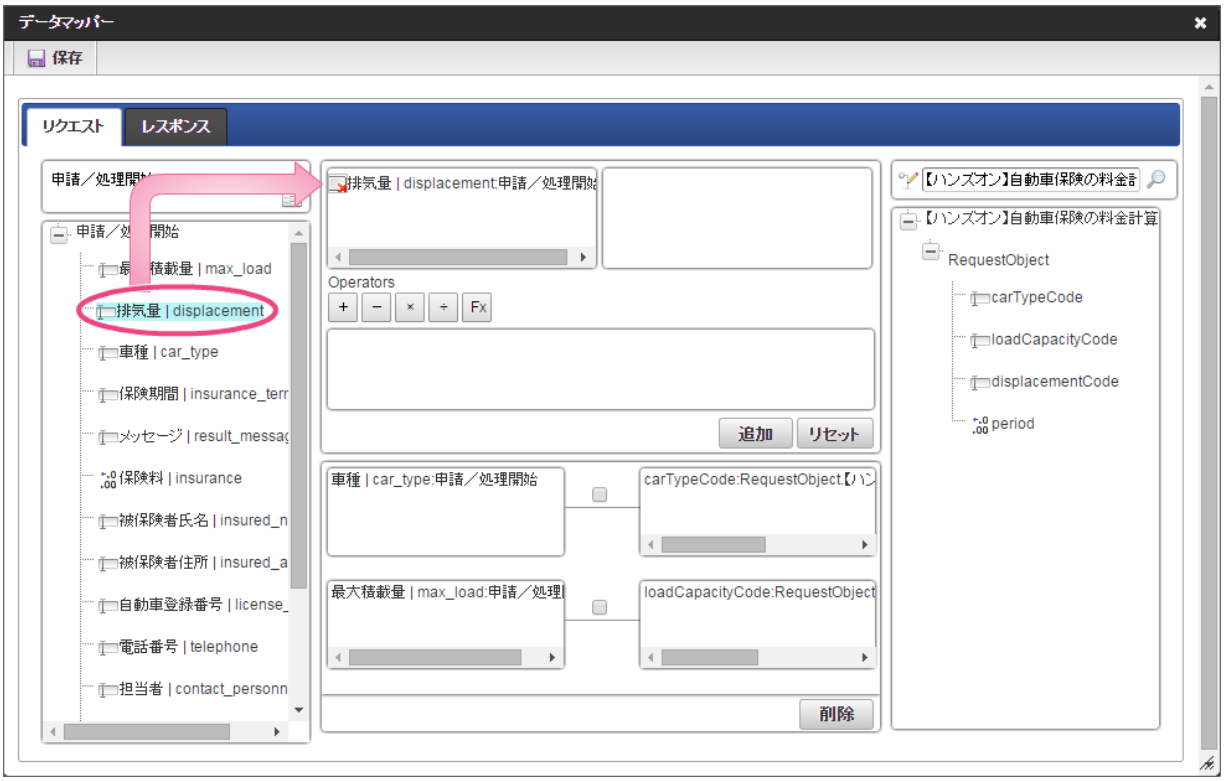
13. 右の欄から「loadCapacityCode」をクリックしてください。
クリック後、中央右の欄に「loadCapacityCode:RequestObject.【ハンズオン】自動車保険の料金計算」と表示されます。



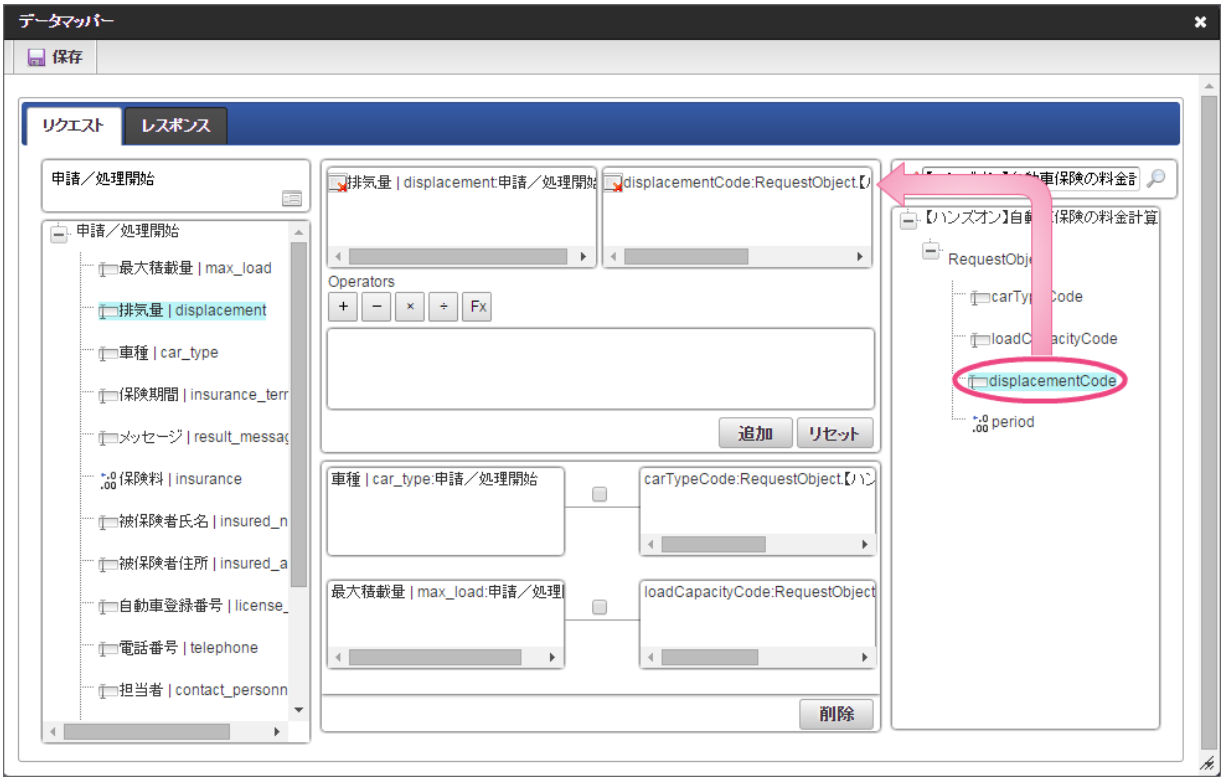
14. 「追加」をクリックしてください。
フォームの「最大積載量」とデータソース定義の「loadCapacityCode」のマッピングが設定され、中央下段の欄に表示されます。



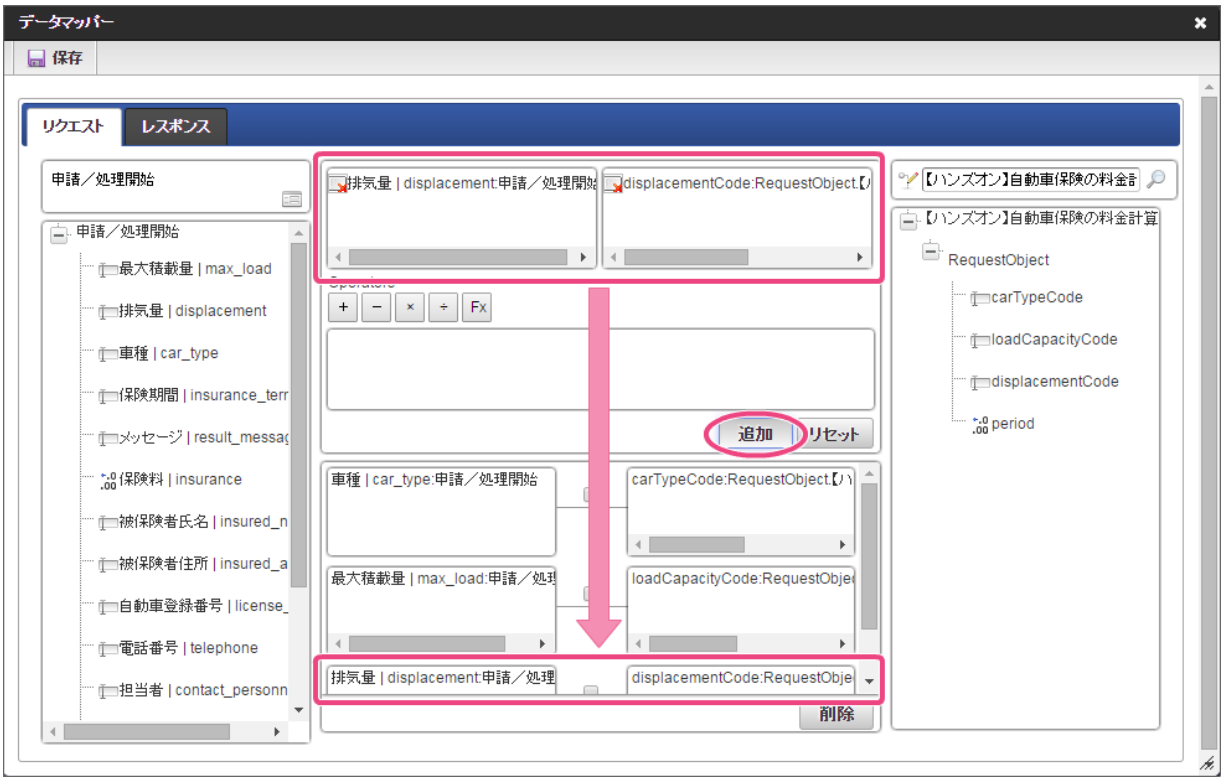
15. 左の欄から「排気量」をクリックしてください。
クリック後、中央左の欄に「排気量 | displacement: 申請／処理開始」と表示されます。



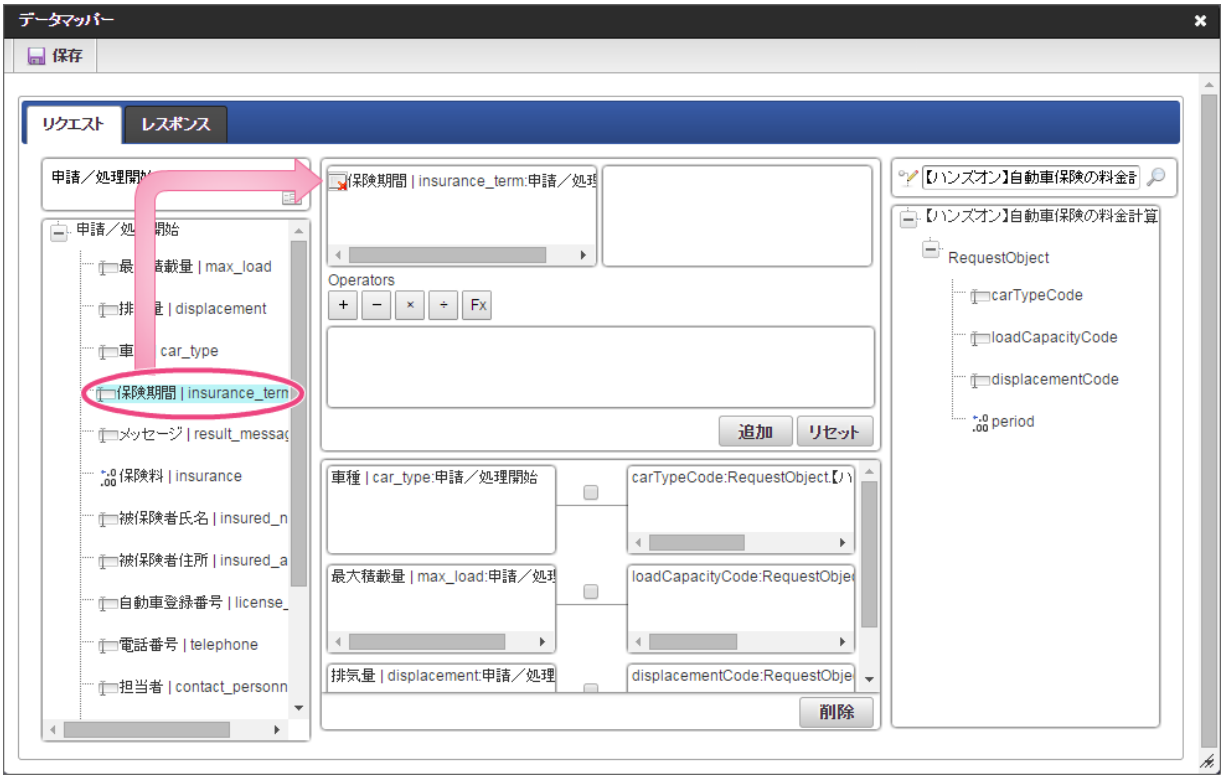
16. 右の欄から「displacementCode」をクリックしてください。
クリック後、中央右の欄に「displacementCode:RequestObject.【ハンズオン】自動車保険の料金計算」と表示されます。



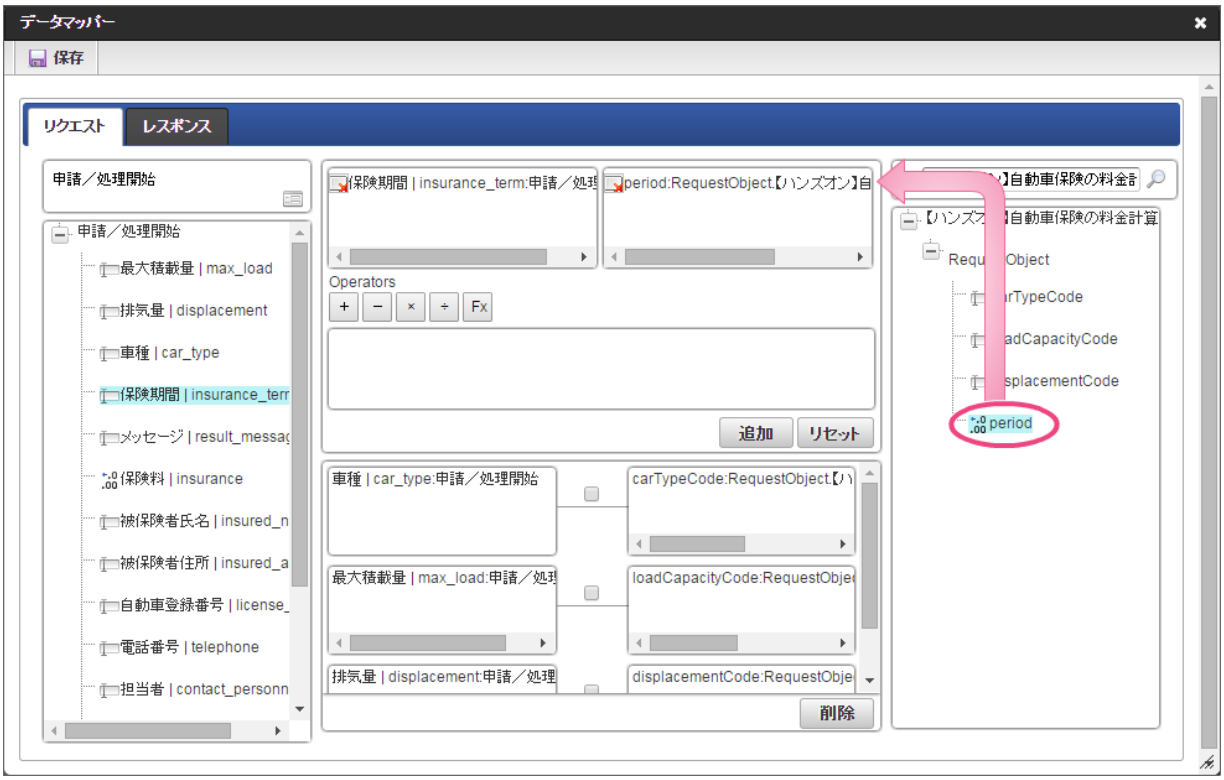
17. 「追加」をクリックしてください。
フォームの「排気量」とデータソース定義の「displacementCode:RequestObject.【ハンズオン】自動車保険の料金計算」のマッピングが設定され、中央下段の欄に表示されます。



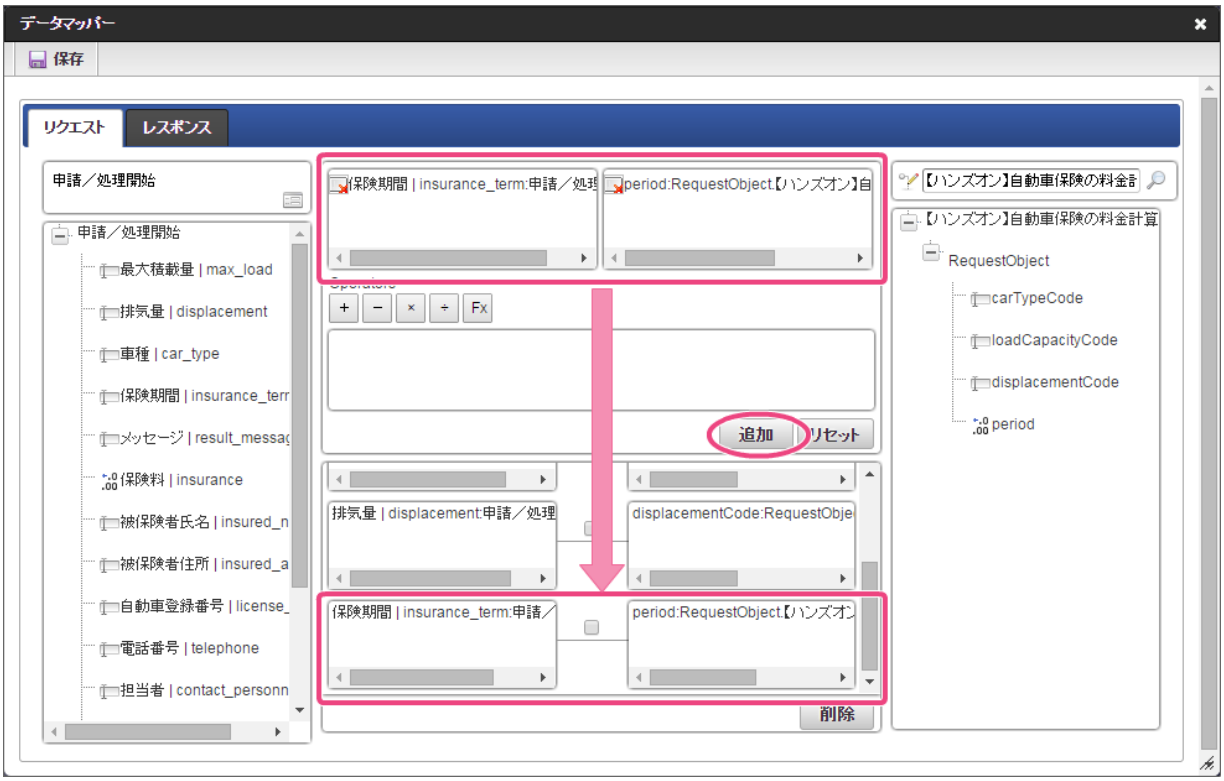
18. 左の欄から「保険期間」をクリックしてください。
クリック後、中央左の欄に「保険期間 | insurance_term: 申請／処理開始」と表示されます。



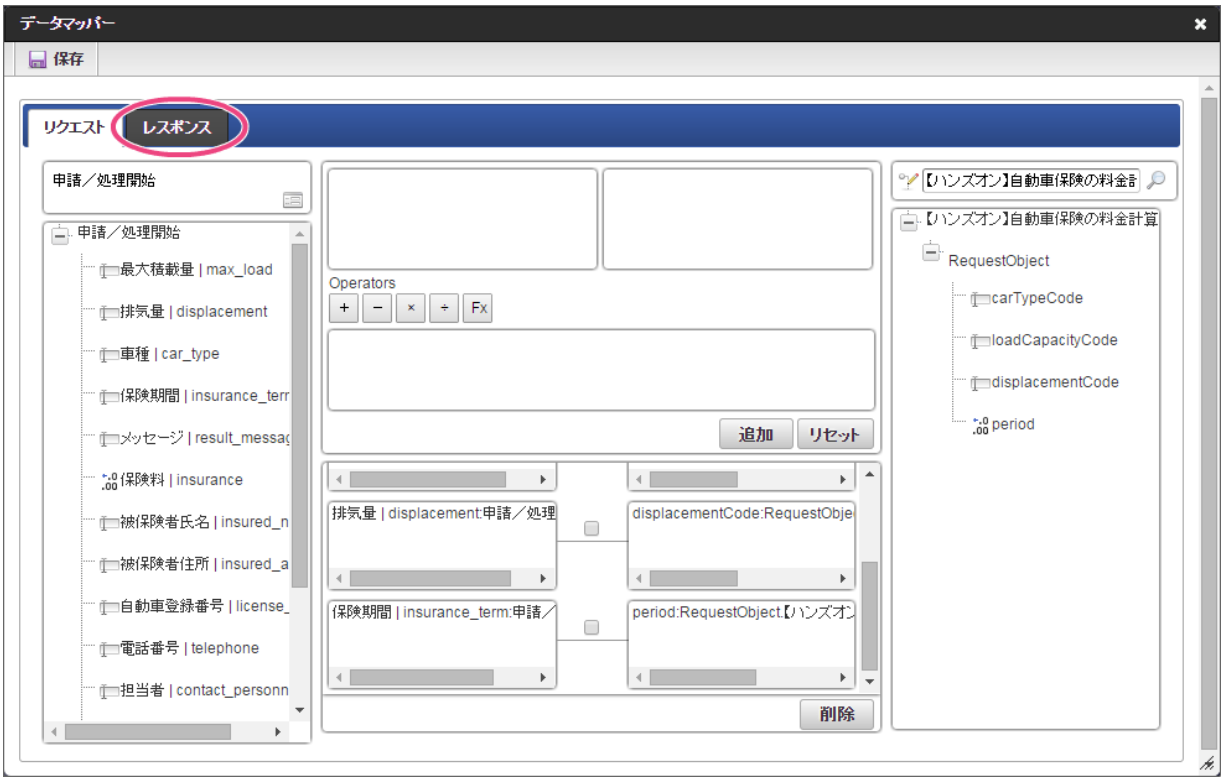
19. 右の欄から「period」をクリックしてください。
クリック後、中央右の欄に「period:RequestObject.【ハンズオン】自動車保険の料金計算」と表示されます。



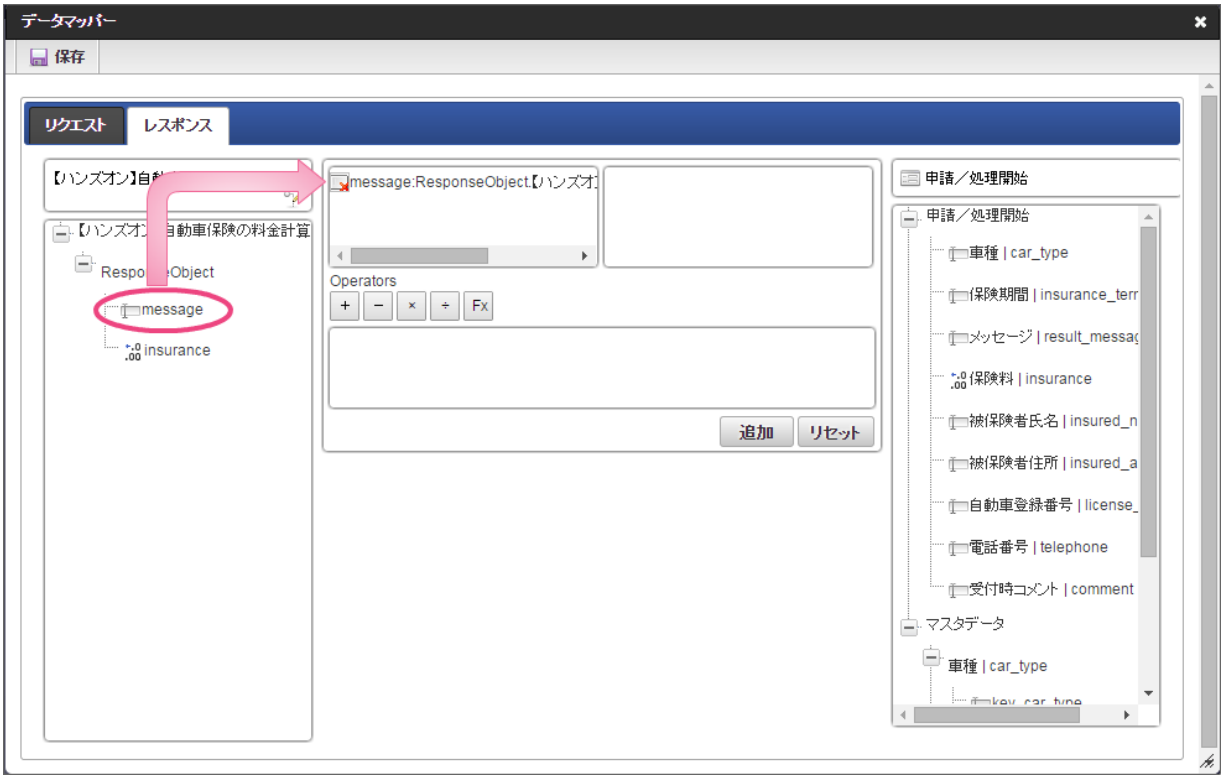
20. 「追加」をクリックしてください。
フォームの「保険期間」とデータソース定義の「period」のマッピングが設定され、中央下段の欄に表示されます。



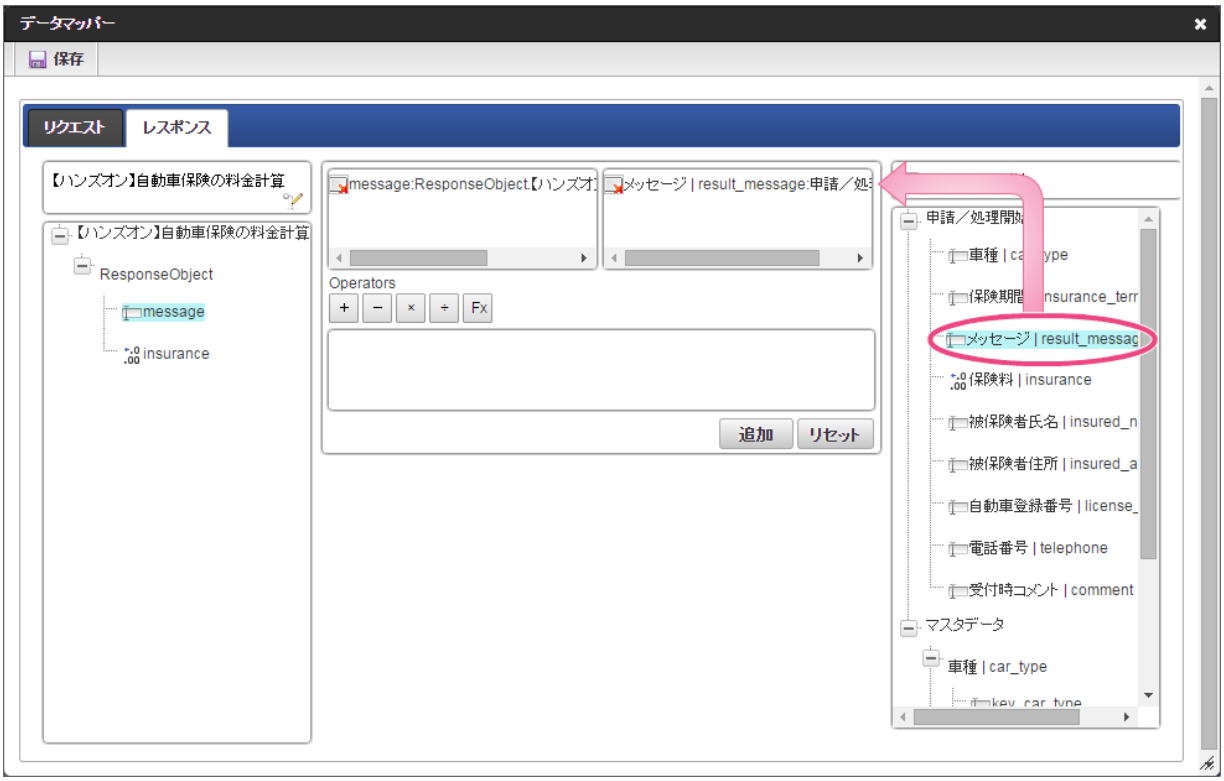
21. リクエストの設定が完了しましたので、レスポンスの設定を行うために「レスポンス」タブをクリックしてください。



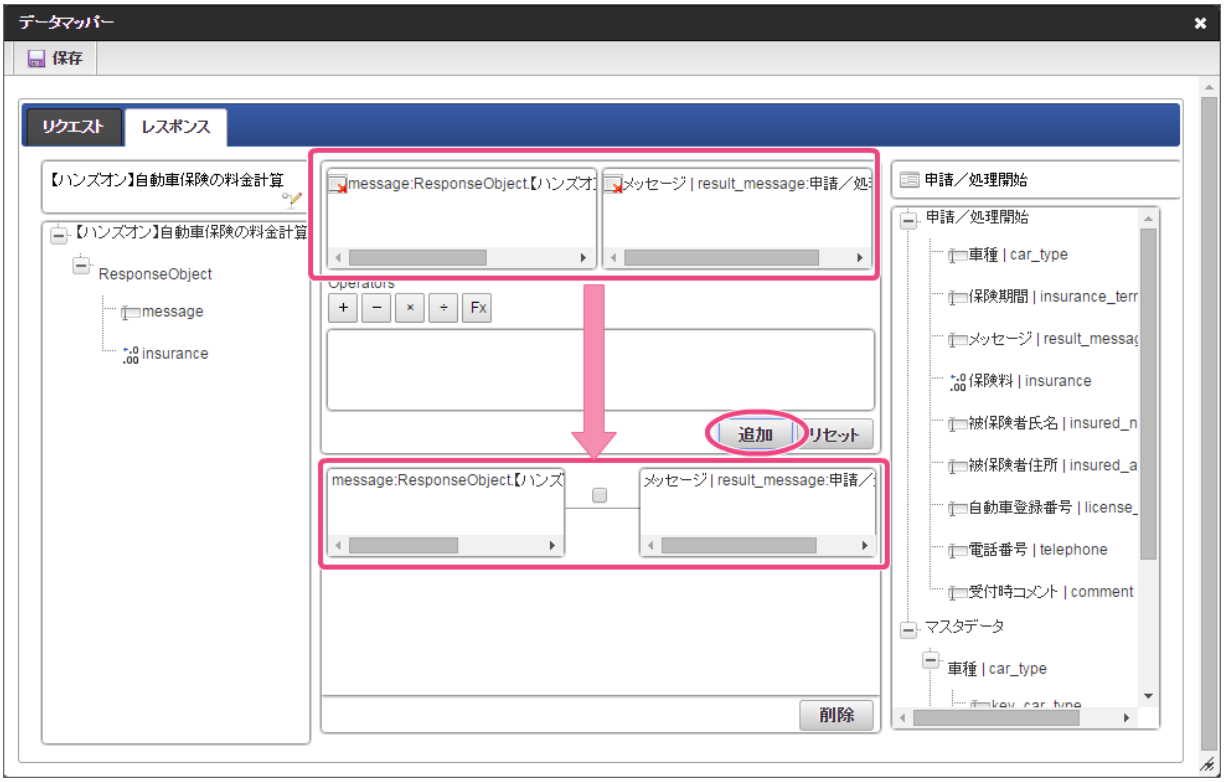
22. 左の欄から「message」をクリックしてください。
クリック後、中央左の欄に「message:ResponseObject.【ハンズオン】自動車保険の料金計算」と表示されます。



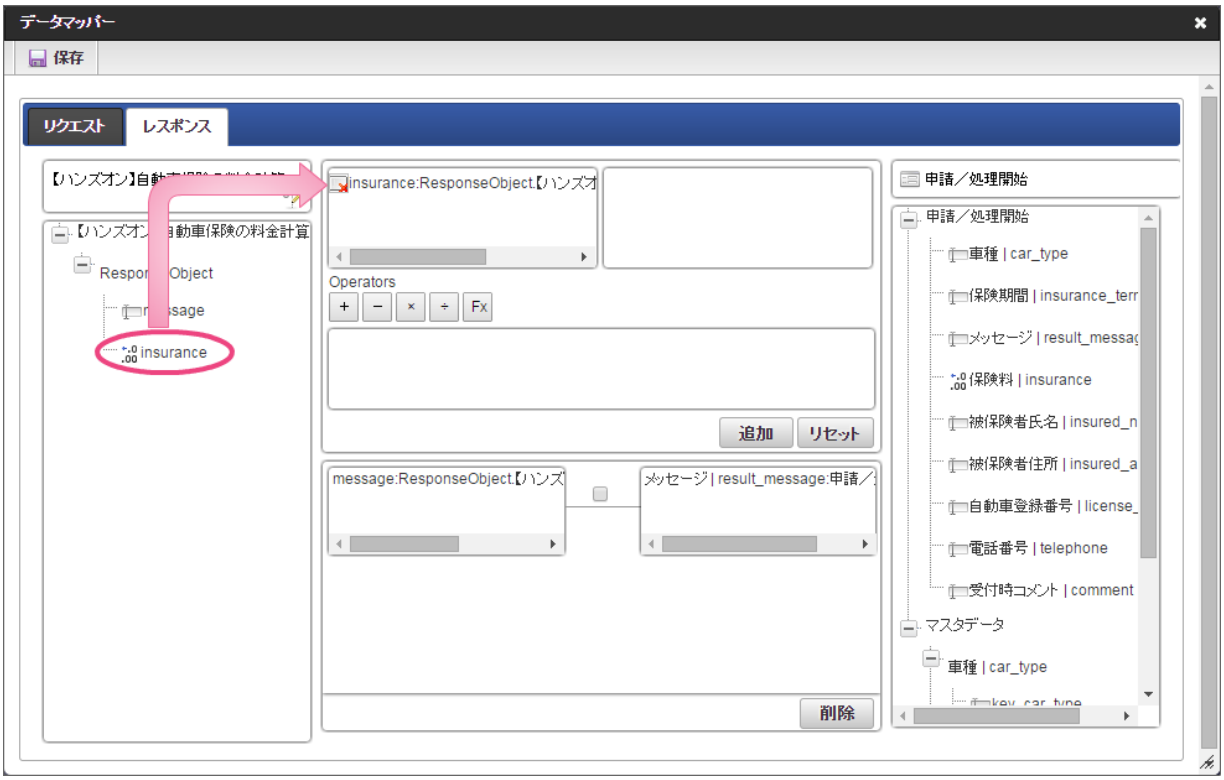
23. 右の欄から「メッセージ」をクリックしてください。
クリック後、中央右の欄に「メッセージ | result_message: 申請／処理開始」と表示されます。



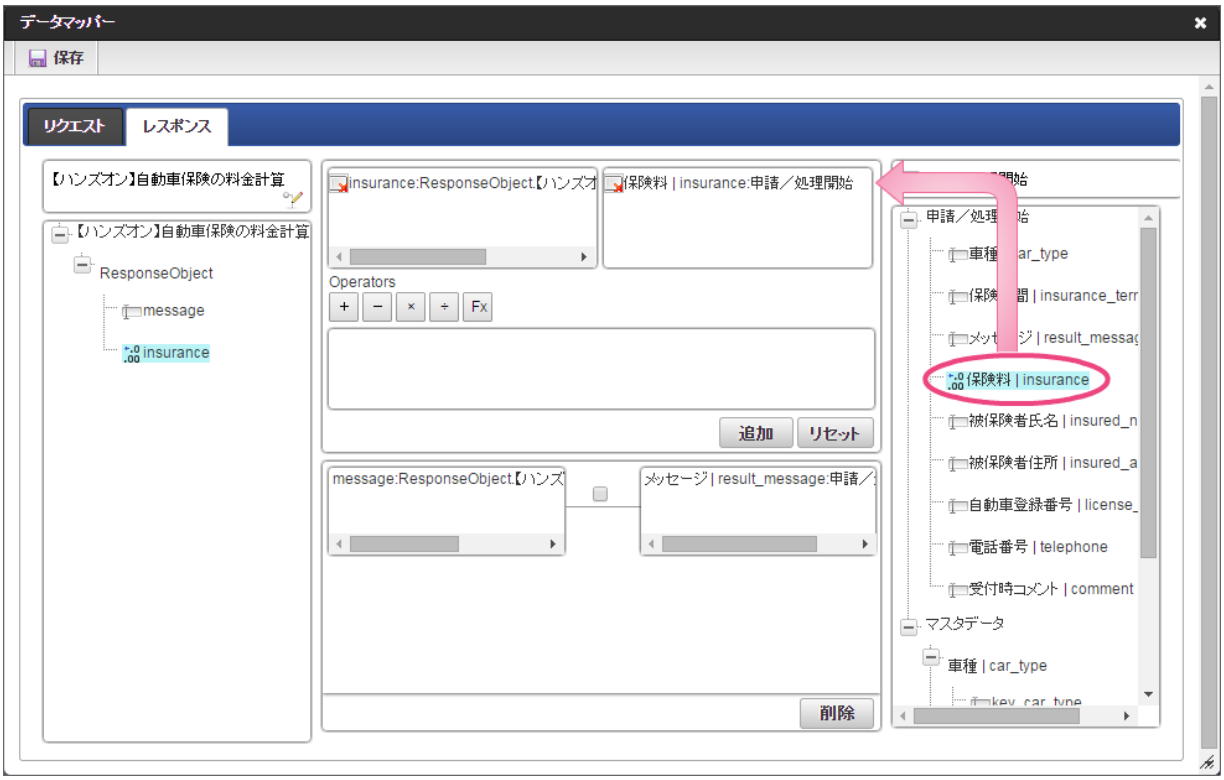
24. 「追加」をクリックしてください。
データソース定義の「message」とフォームの「メッセージ」のマッピングが設定され、中央下段の欄に表示されます。



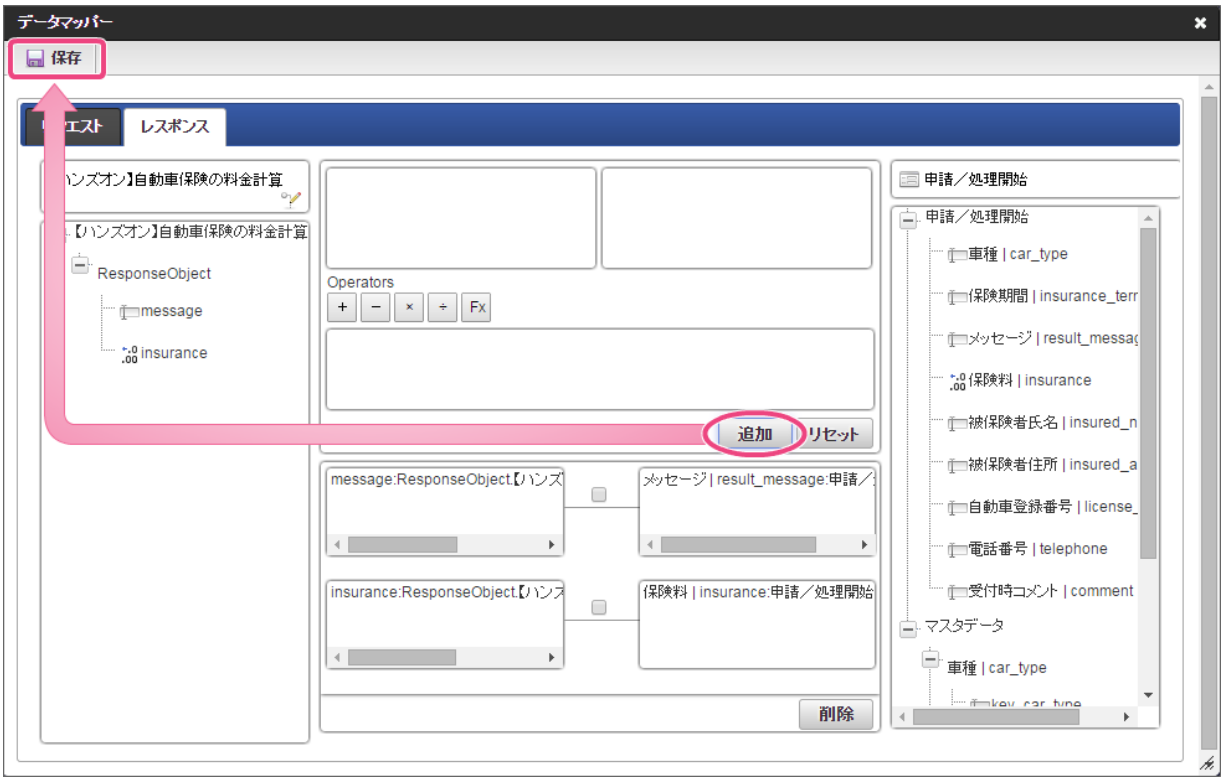
25. 左の欄から「insurance」をクリックしてください。
クリック後、中央左の欄に「insurance:ResponseObject.【ハンズオン】自動車保険の料金計算」と表示されます。



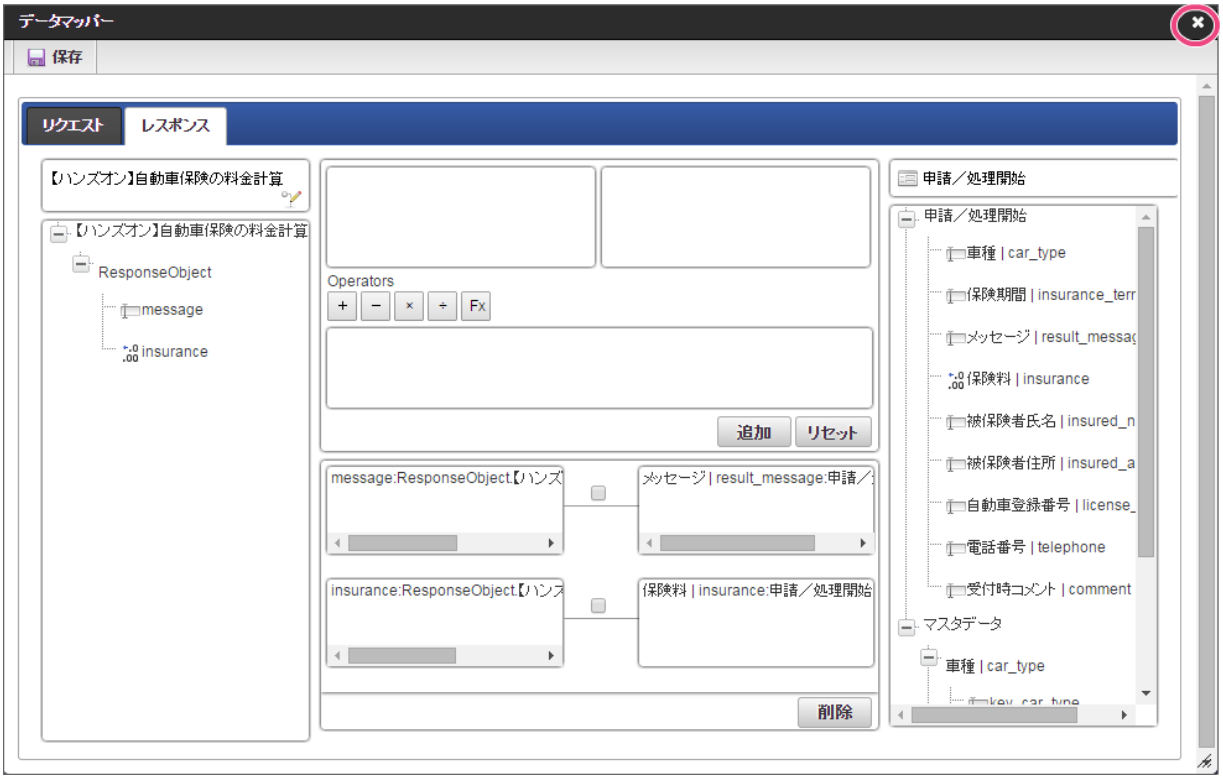
26. 右の欄から「保険料」をクリックしてください。
クリック後、中央右の欄に「保険料 | insuance: 申請／処理開始」と表示されます。



27. 「追加」、「保存」の順にクリックしてください。



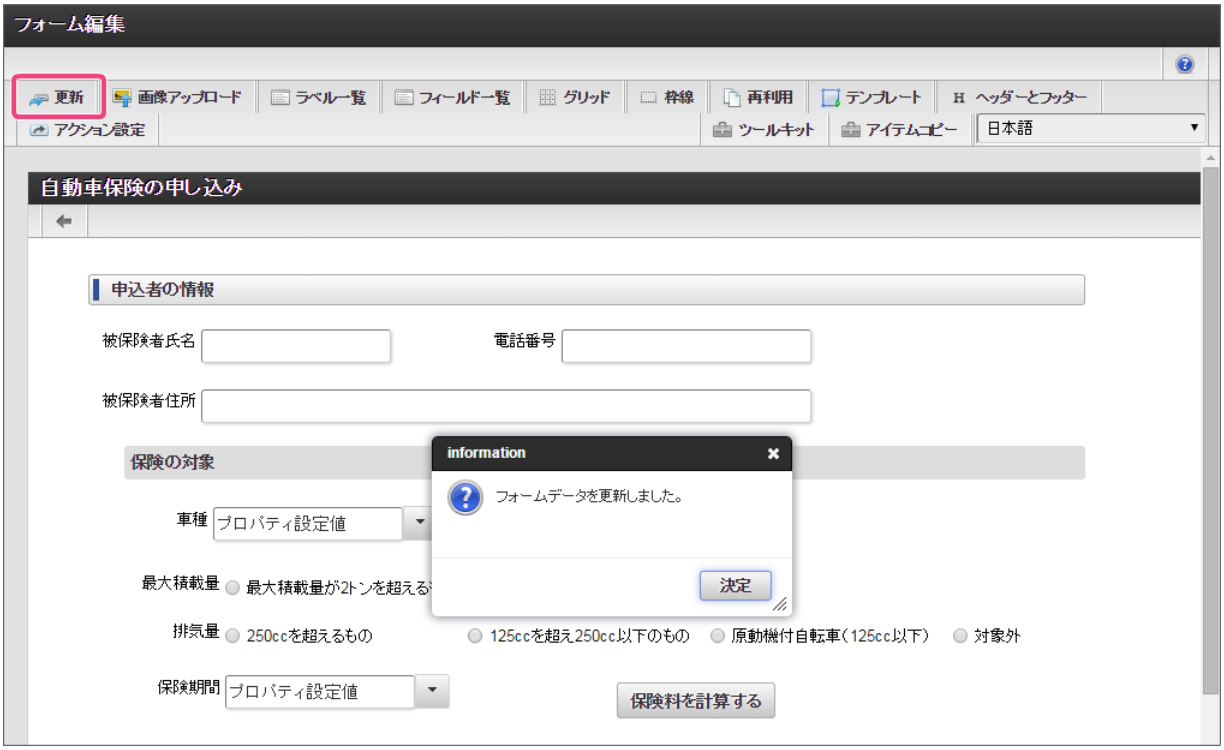
28. 正常に保存できたら、「データマッパー」は右上の「 ✖ 」をクリックして閉じてください。



29. アクション設定で「確定」をクリックしてください。

30. イベント設定で「確定」をクリックしてください。

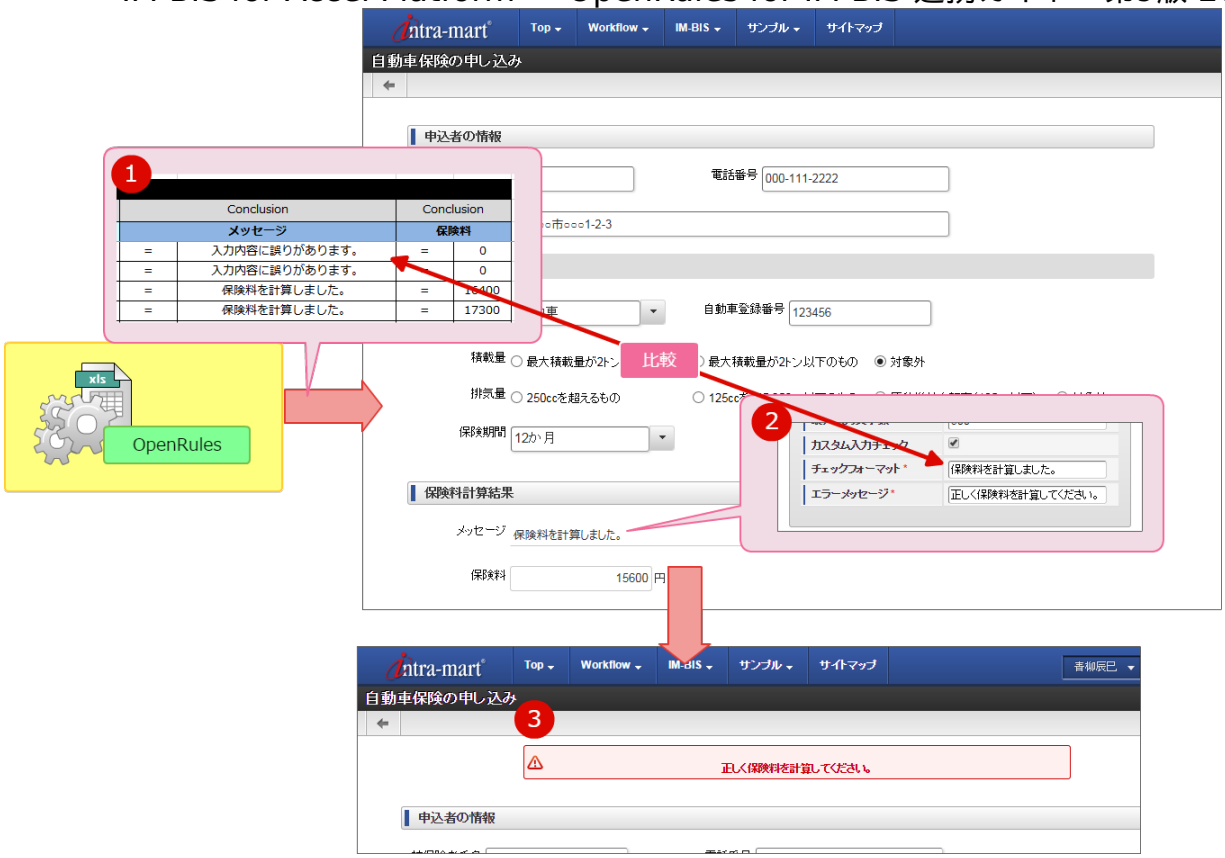
31. 「更新」をクリックして、フォーム（画面）を保存してください。



ルールから返却するメッセージを利用した入力チェックを設定する

ルールから返却するメッセージを利用して、申請ボタンをクリックしたタイミングで入力チェックを行えるように設定しましょう。

このハンズオンでは、アイテムの入力チェック「カスタム入力チェック」とルールから返却するメッセージを組み合わせて、入力チェックを設定します。

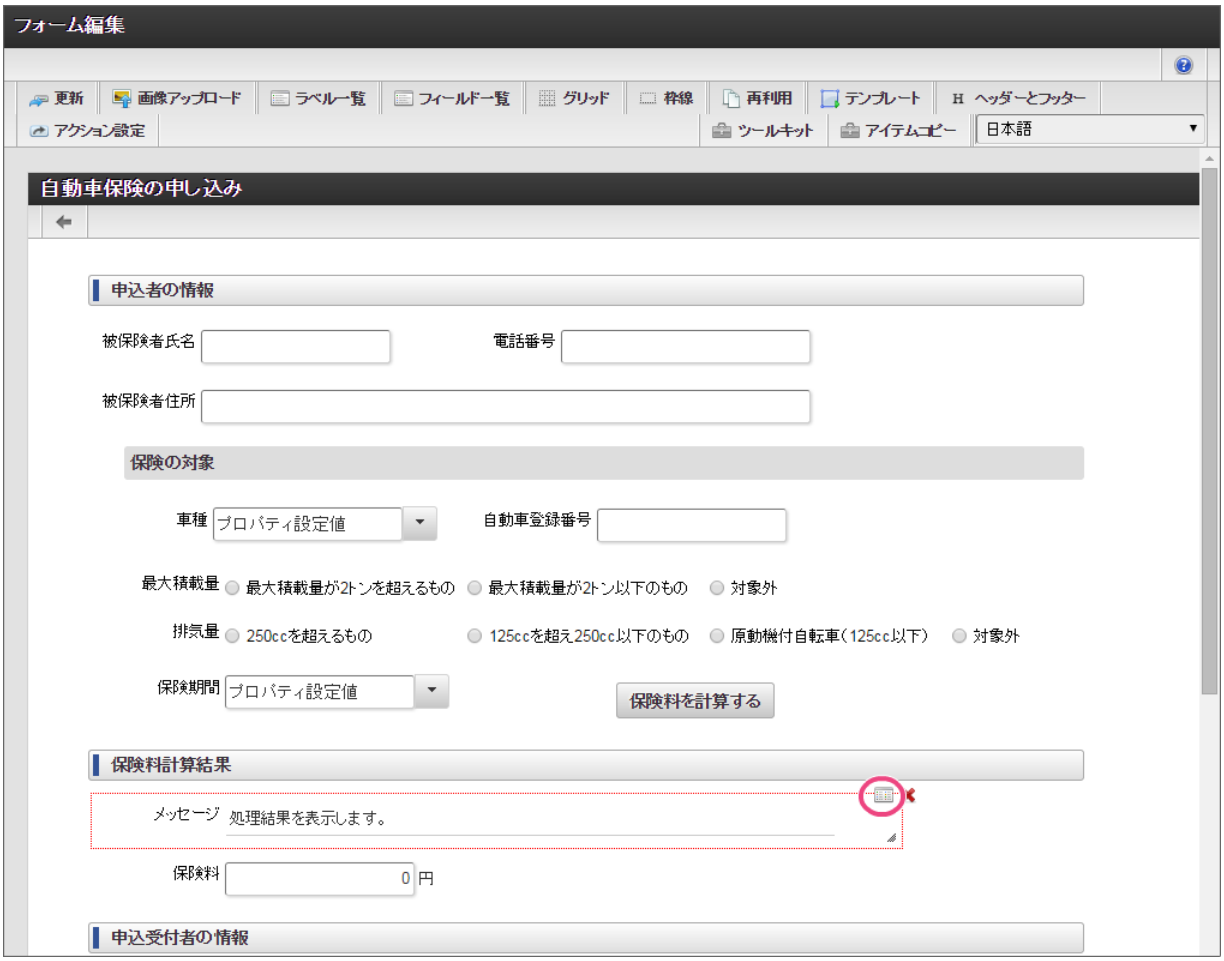


【入力チェック～エラーメッセージの処理の流れ】

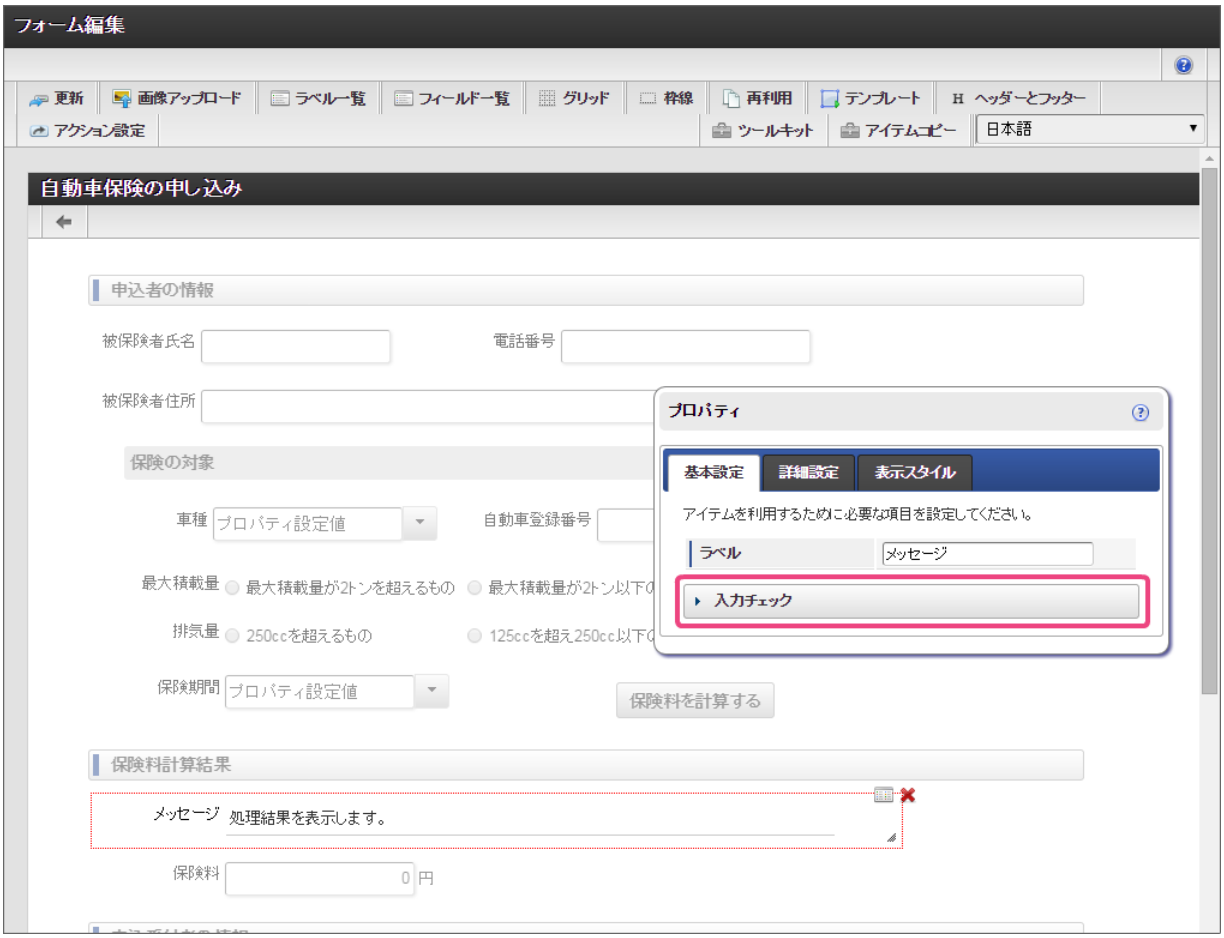
1. OpenRules からの処理結果として、「メッセージ」と「保険料」を返却します。
2. 画面（フォーム）でメッセージを表示するアイテムのプロパティ「カスタム入力チェック」の「チェックフォーマット」を利用してエラーメッセージの表示をコントロールします。
   - OpenRules からのメッセージ ＝ チェックフォーマットに指定したメッセージ
     →正しい内容と判断し、エラーメッセージを表示しません。
   - OpenRules からのメッセージ ≠ チェックフォーマットに指定したメッセージ
     →誤った内容と判断し、エラーメッセージを表示します。
3. 上記のチェックが「申請」ボタンをクリック時に実行され、エラーの場合にはメッセージを画面上部に表示します。

以下の手順で入力チェックを設定しましょう。

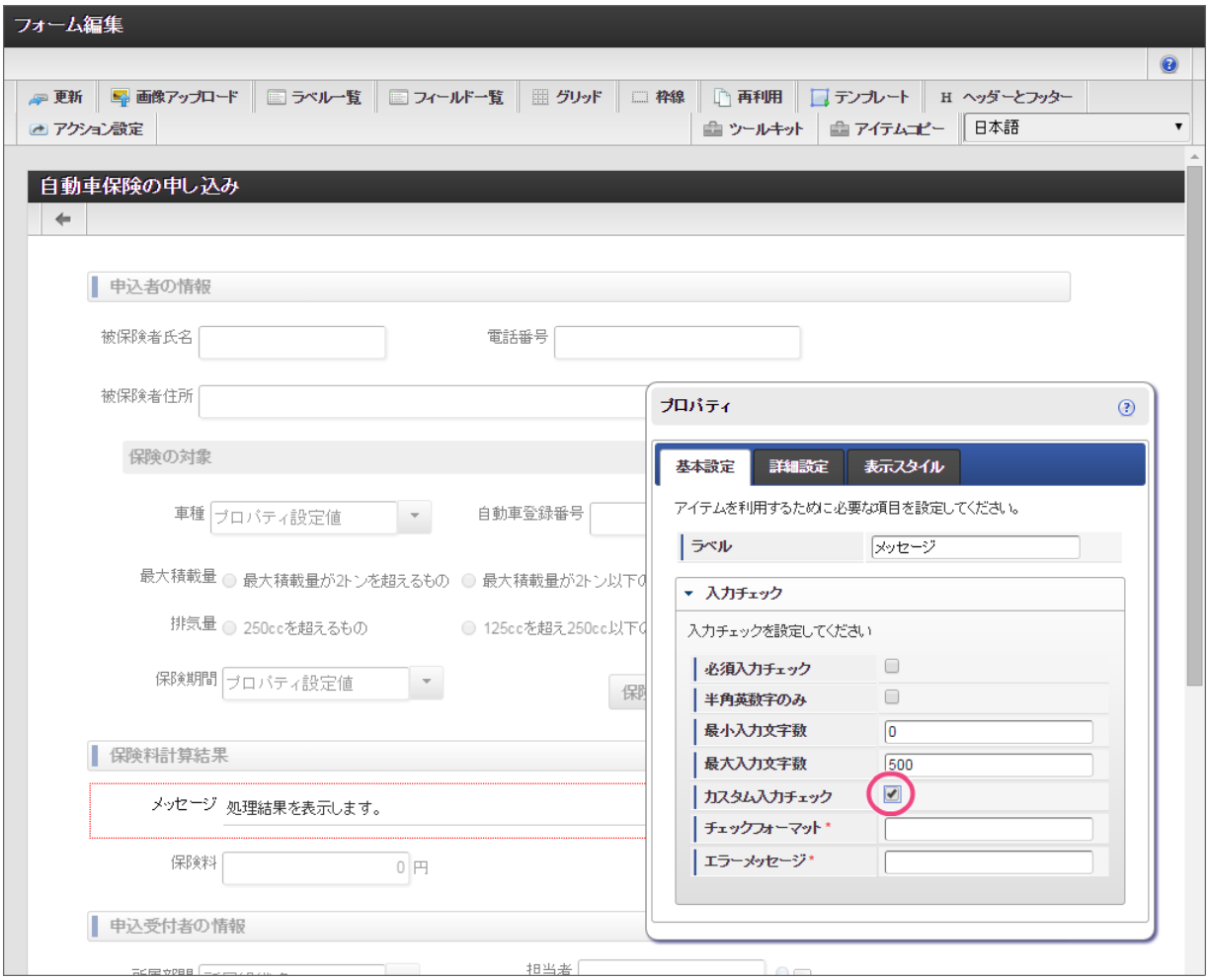
1. フォーム編集画面で「メッセージ」のアイテムを選択し、プロパティアイコンをクリックしてプロパティを表示してください。



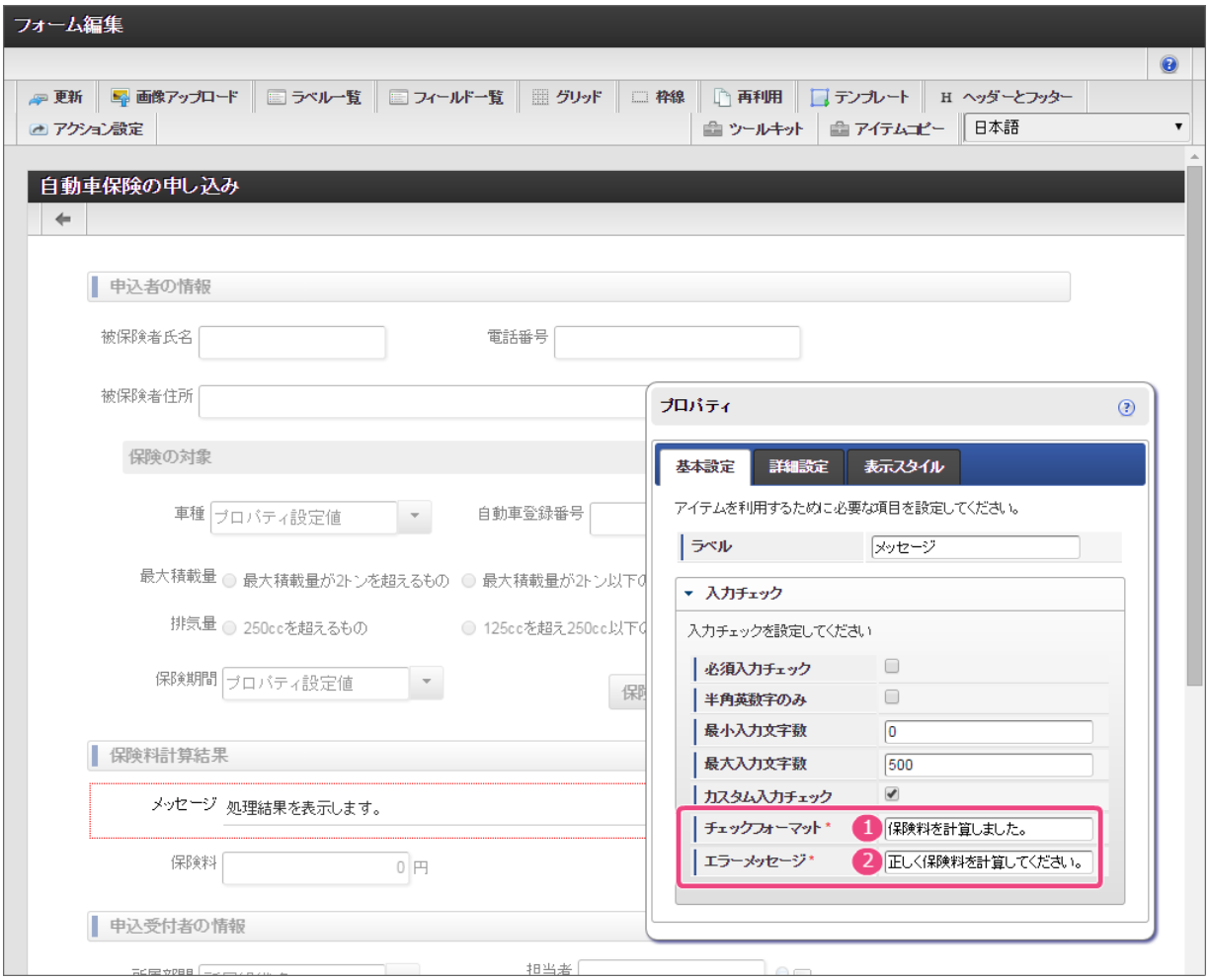
2. プロパティの「入力チェック」をクリックしてください。



3. 「カスタム入力チェック」のチェックボックスをクリックして、オンにしてください。



4. チェックフォーマットとエラーメッセージを以下の通りに設定してください。



設定項目	設定値
(1)チェックフォーマット	保険料を計算しました。
(2)エラーメッセージ	正しく保険料を計算してください。

5. 「更新」をクリックして、フォーム（画面）を保存してください。

ルールから値を返却する項目の値を変更できないように設定する

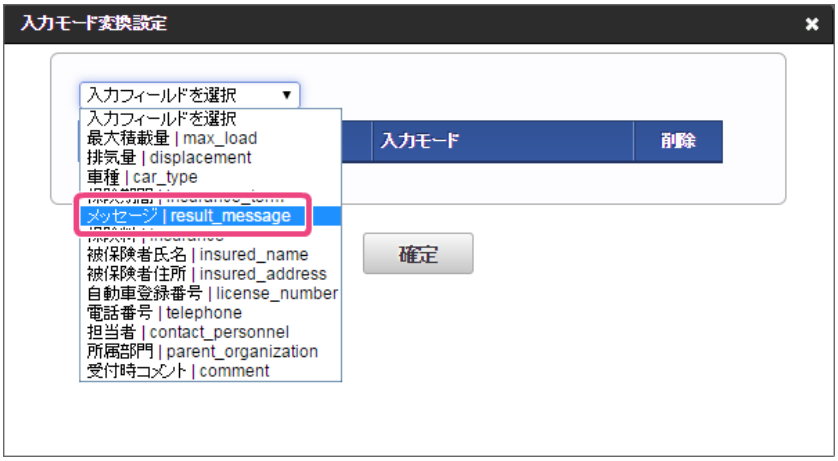
ルールから返却する値を格納する項目には、ユーザが値を変更できないように入力モード変換を設定しましょう。

1. フォーム編集画面で「アクション設定」をクリックしてください。

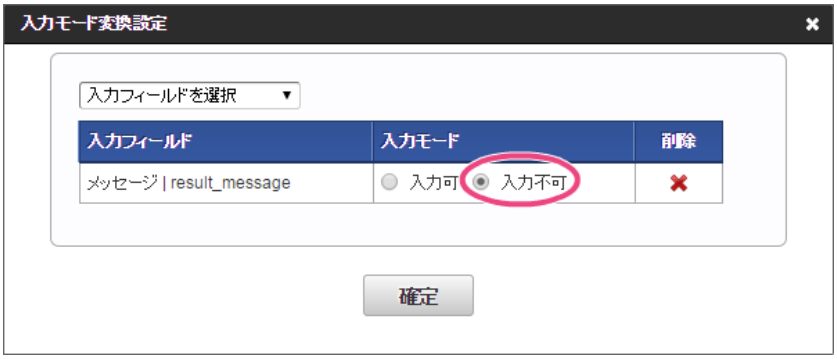
2. 「初期表示イベント」で「＋」アイコンをクリックしてください。

3. 「アクション」を「入力モード変換」にし、「 」をクリックしてください。

4. 「入力フィールド」から「メッセージ」を選択してください。



5. 「入力不可」に設定してください。

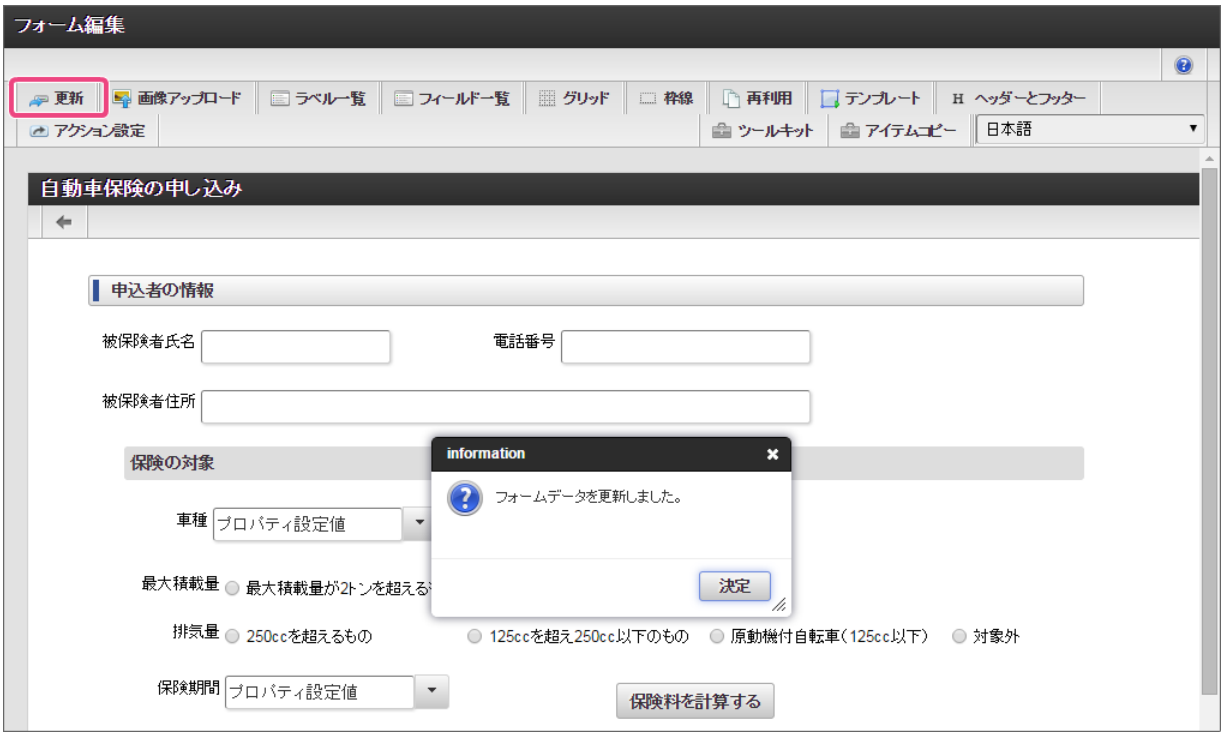


6. 同様の手順で、入力フィールドに「保険料」を追加し、入力モードを「入力不可」に設定してください。

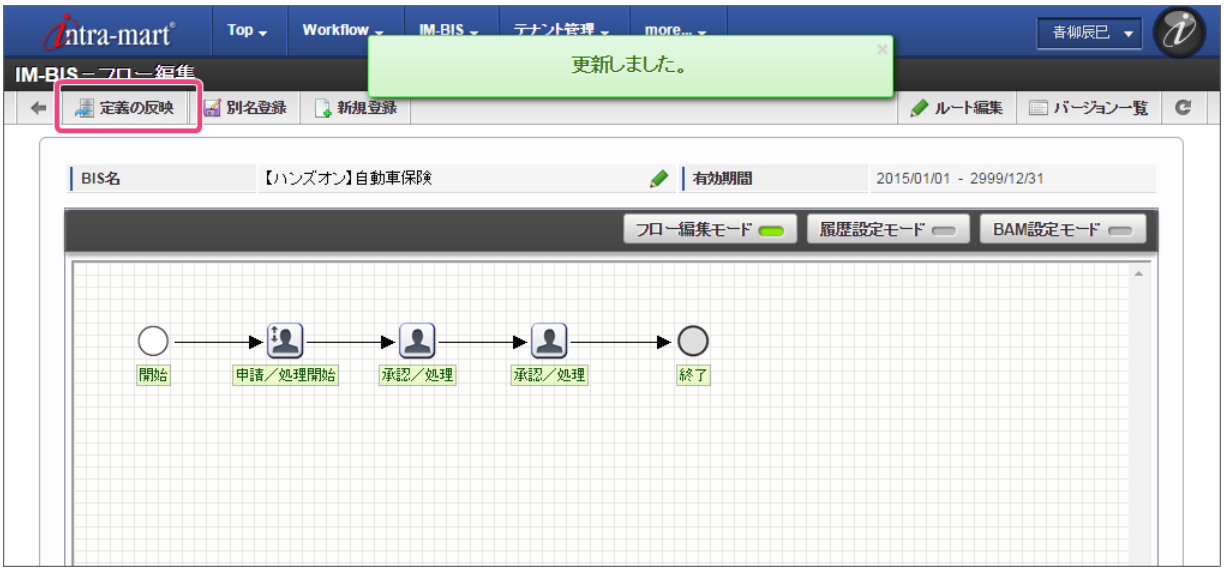
7. 以下の通りに入力モード変換が設定できたら、「確定」をクリックしてください。

8. イベント設定で「確定」をクリックしてください。

9. 「更新」をクリックして、フォーム（画面）を保存してください。
   フォームが正常に保存できたら右上の「 x 」でフォーム編集画面を閉じます。



10. 最後に「定義の反映」をクリックして、フローを実行できるようにします。



11. これで、必要な設定作業はすべて完了しましたので、実際にフローで申請・承認を行ってみましょう。

自動車保険の申し込みワークフローを実行してみよう

これまでのシナリオで作成した IM-BIS のフローを使って、保険料の計算を実行してみましょう。

ルールと連携したフローを実行する手順

- 自動車保険申し込みの申請画面でルールを実行する
- 自動車保険申し込みの承認を実行する

自動車保険申し込みの申請画面でルールを実行する

作成したワークフローの申請画面でルールを実行してみましょう。

申請画面を表示する

1. 「BIS担当者」ロールを付与したユーザでログインしましょう。
   （このマニュアルでは、「青柳辰巳」（ユーザコード：aoyagi）でログインします。）

2. 上部のメニューの「 IM-BIS 」→「ワークフロー」の順にマウスを重ねてから「申請」をクリックしましょう。

3. 「【ハンズオン】自動車保険」の「申請／処理開始」をクリックして申請画面を表示します。

ルールから返却されたエラーによって申請時にエラーメッセージを表示する

最初に入力した保険料の内容が誤っている場合に適切にエラーメッセージが表示され、申請を実行できないことを確認してみましょう。

1. 申請画面で、車種の値は変更せずに「最大積載量」の値だけを変更してみましょう。

2. 「保険料を計算する」をクリックしてください。

3. ルールが実行され、メッセージには「入力内容に誤りがあります。」と表示されます。
   このまま「申請」をクリックしてみましょう。

4. 申請画面の上部にカスタム入力チェックで設定したエラーメッセージが表示され、申請が実行できないことが確認できました。

保険料を計算し、申請を実行する

保険の対象の入力値の組み合わせを正しいパターンに変更して、申請してみましょう。

1. 申請画面で、正しく保険料が計算される組み合わせの値を入力してみましょう。
   例として、バイク（軽二輪車）の36ヶ月契約として入力してください。

2. 「保険料を計算する」をクリックしてください。

3. ルールが実行され、メッセージには「保険料を計算しました。」、保険料には入力した内容に基づく金額が表示されます。
   その他の項目に入力し、「申請」をクリックしてみましょう。

4. エラーは発生せずに、申請の処理画面が表示されます。
   案件名を変更し、「申請／処理開始」をクリックしましょう。

5. 申請を行うことができました。
   続いて、承認を実行してみましょう。

自動車保険申し込みの承認を実行する

先ほど申請した案件を承認しましょう。

1. 「BIS担当者」ロールを付与したユーザでログインしましょう。
   （このマニュアルでは、「青柳辰巳」（ユーザコード：aoyagi）でログインします。）

2. 上部のメニューの「 IM-BIS 」→「ワークフロー」の順にマウスを重ねてから「未処理」をクリックしましょう。

3. 申請した「【ハンズオン】自動車保険」の案件が表示されます。
   「 処理」をクリックして承認画面を表示してください。

4. 申請内容がそのまま表示されていることが確認できました。
   「確認者コメント」を入力し、「承認」をクリックしましょう。

5. 承認を行うことができました。
   同様の手順で、最後のノードの承認も行い、案件を完了しましょう。

6. これで自動車保険の保険料を計算するワークフローの実行について確認することができました。

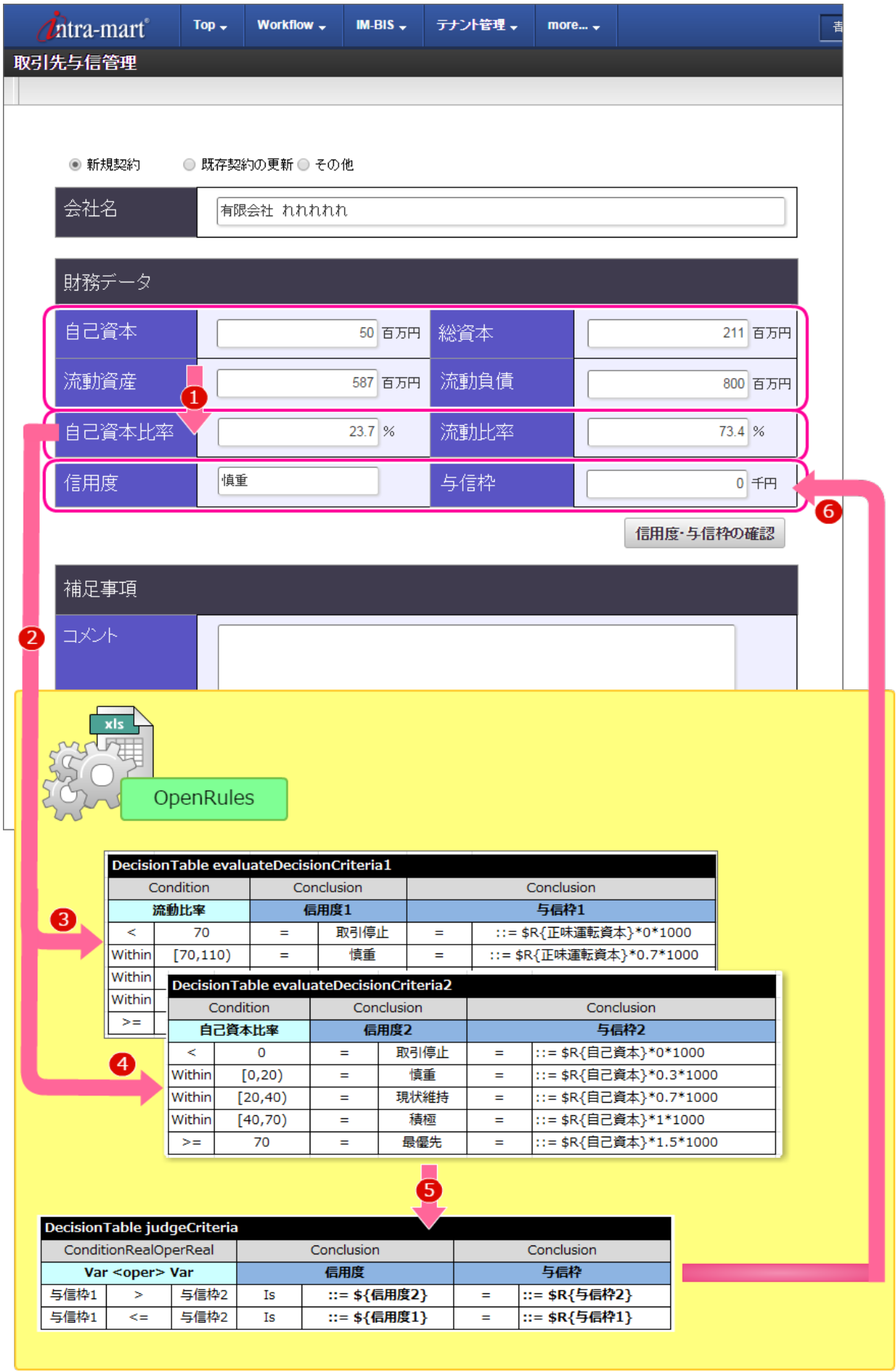
複雑なルールを利用して、取引先の与信管理をしてみよう

IM-BIS を利用したフローで複数のデシジョンテーブルを実行したり、計算を行う複雑なルールを作成してみましょう。

ハンズオンシナリオ（取引先与信管理フローの作成）の概要

このシナリオでは、取引先与信管理のフローを作成します。

- 画面に入力した取引先の資本などの情報に基づいて、与信額と信用度を算出します。



【処理の流れ】

1. 入力された自己資本、総資本、流動資産、流動負債に基づいて、自己資本比率、流動比率を計算する
2. 算出された流動比率や自己資本比率を利用してルールを実行する
3. 流動比率に基づいた信用度（信用度1）、与信枠（与信枠1）を返却する

4. 自己資本比率に基づいた信用度（信用度2）、与信枠（与信枠2）を返却する
5. 3と4で返却された信用度と与信枠を比較し、最終的に返却する信用度と与信枠を決定する
6. 返却された信用度と与信枠を画面に表示する

このシナリオを通して習得できる知識

このシナリオを実行すると、以下の知識を理解することができます。

- OpenRules での演算子の利用方法
- OpenRules での複数の *DecisionTable* の実行順のコントロール
- *DecisionTable2* と *DecisionTable* の処理の違いを利用したルールの設定方法

このシナリオを学習する前に必要な知識

このシナリオの実行に当たり、下記の知識を事前に確認してください。

- IM-BIS 、 IM-FormaDesigner の定義ファイルのインポート方法
- OpenRules のExcelの定義ファイルの基本的な構成

OpenRules で取引先に対する与信枠や信用度を評価するルールを作成する

OpenRules では、 *Glossary* で登録した項目の値同士を比較して条件や計算を行うことができます。

このハンズオンでは、取引先与信管理フローをサンプルに、 OpenRules で以下の処理を実行するための設定方法を確認します。

- 条件（ *Conclusion* ）での「○○以上△△未満」で評価する
- 異なる *DecisionTable* の結果を比較する
- *Glossary* の項目を使って計算を実行する
- 評価結果を再評価して変更する

ハンズオンで扱っている OpenRules の記法は、高度な内容を含んでおります。
適宜「 *OpenRules のキーワードリファレンス* 」を参照しながら進めるようにしてください。

ルールを定義するExcelファイルを作成する手順

- このシナリオで作成するルールの概要
- OpenRules で 数値の範囲指定や計算を行う方法
- ルールのExcelファイルを作成する手順
- 取引先与信管理のハンズオンを開始するための準備
- 作成するルールの構成を決める
- ルールの構成でまとめた４つの DecisionTable を作成する
- 実行に必要な定義を設定する

このシナリオで作成するルールの概要

- 作成するルールの内容

  サンプル会社では、取引先の与信管理を以下のように評価しており、取引先の情報に基づいて与信枠や信用度を決定しています。
  このハンズオンでは、与信枠の計算や信用度の評価を OpenRules を利用して実行します。

  - 評価基準1
    - 取引先企業の流動資産・流動負債に基づく流動比率に応じて、信用度・与信枠を以下の表に基づいて設定する。

流動比率 [1]	**信用度**	**与信枠の計算式** [2]
70%未満	取引停止	正味運転資本*0
70%以上110%未満	慎重	正味運転資本*0.7
110%以上160%未満	現状維持	正味運転資本*1
160%以上200%未満	積極	正味運転資本*1.4
200%以上	最優先	正味運転資本*2

  - 補足

    [1] 流動比率 = 流動資産 / 流動負債

[2] 正味運転資本 = 流動資産 - 流動負債

- 評価基準2
  - 取引先企業の総資本・自己資本に基づく自己資本比率に応じて、信用度・与信枠を以下の表に基づいて設定する。

自己資本比率 [3]	信用度	与信枠の計算式
0%未満	取引停止	自己資本*0
0%以上20%未満	慎重	自己資本*0.3
20%以上40%未満	現状維持	自己資本*0.7
40%以上70%未満	積極	自己資本*1
70%以上	最優先	自己資本*1.5

    - 補足

      [3] 自己資本比率 = 自己資本 / 総資本

- 評価基準1と評価基準2の結果を比較し、与信枠が小さい方を最終結果とする。
  ただし、与信枠が負数となった場合には、0に設定する。

OpenRules で 数値の範囲指定や計算を行う方法

このハンズオンを開始する前に、ハンズオンで作成するルールに必要な OpenRules の記法を確認しましょう。

条件として数値の範囲を指定する

このハンズオンでは、評価基準1・評価基準2の条件で「～以上で～未満の場合」というものがあります。
このような数値型の項目の特定の値の範囲を条件に指定するためには、「 *演算子* 」の「Within」とさまざまな記号を組み合わせて設定します。

- 項目Aが10以上、20以下を条件とする場合
  「～以上～以下」を表現するには、演算子に「Within」を使い、以下のいずれかの式を記述してください。

DecisionTable sample1

Condition		Conclusion	
項目A		メッセージ	
Within	10-20	=	例1
Within	between 10 and 20	=	例2

```
// "-"を利用する
%下限値%-%上限値%

// "between A and B"を利用する
between %下限値% and %上限値%
```

- 項目Aが10以上、20未満を条件とする場合
  「～以上～未満」を表現するには、演算子に「Within」を使い、以下の式を記述してください。

DecisionTable sample1

Condition		Conclusion	
項目A		メッセージ	
Within	[10;20)	=	例1
Within	[10,20)	=	例2

```
// 以上・未満の際は前後の記号が異なる点に注意してください
[%下限値%;%上限値%)
```

Glossary で定義した項目の値を取得する

このハンズオンでは、評価基準1・評価基準2の結果を比較し、与信枠の小さい方の値を最終的な与信枠の結果として設定します。
複数の *DecisionTable* の結果を設定した項目から別の項目に値をセットするには、 *$(getString)* などの「マクロ」という記法を利用します。

“$”で開始する「マクロ」を利用せずに項目名を記述した場合には、その項目名は項目名と扱われずに、書いたとおりの「文字列の値」として扱います。

OpenRules が提供するマクロ（一部抜粋）

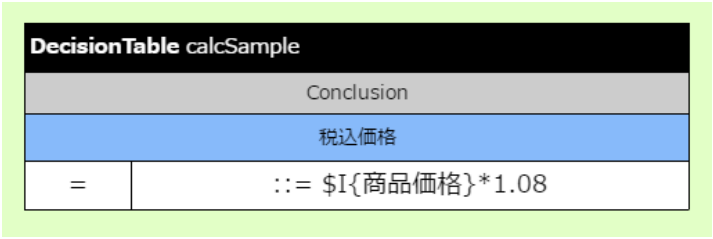
マクロの形式	対応する項目のデータ型
${項目名（論理名）}	文字型（java.lang.String）
$I{項目名（論理名）}	整数型（int）
$R{項目名（論理名）}	浮動小数点数型（double）

上記のマクロを利用するには、 「 *Datatype* 」で定義したデータ型に対応したマクロを利用してください。

このハンズオンでは、マクロを使い、値の取得方法や値の計算を行います。

OpenRules の処理中に計算を行う

DecisionTable では、条件に合致した場合の評価として、計算を行うことができます。
このハンズオンでは、各評価基準の条件に合わせて与信枠を *Conclusion* で設定していきます。



OpenRules では、Javaで利用できる演算子（+,-,*,/,%）を用いた計算を実行できます。
計算式の先頭には”::=”を記述してください。

Glossary で定義した項目同士の値を比較する

これまでのハンズオンでは、項目と特定の値を比較する条件を扱いましたが、このハンズオンでの「評価基準1の与信枠と評価基準2の与信枠の比較」のような項目同士の比較を行う場合には、項目同士を比較するためのキーワードを利用します。
「 *ConditionIntOperInt* 」や「 *ConditionRealOperReal* 」などのキーワードを利用すると、2つの項目の値同士の比較をすることができます。

異なる項目同士の比較を行う場合には、サブヘッダに「Condition[データ型]Oper[データ型]」というキーワードを利用します。
比較のキーワードは、データ型に対応して以下の種類が提供されています。

- int
  ConditionIntOperInt
- real（double）
  ConditionRealOperReal
- Date
  ConditionDateOperDate

1つの *DecisionTable* の処理中に結果を変更する

このハンズオンでは、「評価基準1の与信枠と評価基準2の与信枠を比較して、小さい方の値を与信枠に設定するが、 **負数の場合には、0をセットする** 」という要件があります。
この要件を実現するためには、 *DecisionTable* の評価後に再度負数かどうかの評価を行う必要があります。
一度評価した結果の再評価を実行する方法の1つとして、 *DecisionTable2* を利用すると、後から評価された条件の結果によって先に評価された結果を変更することができます。

DecisionTable2 DT2sample

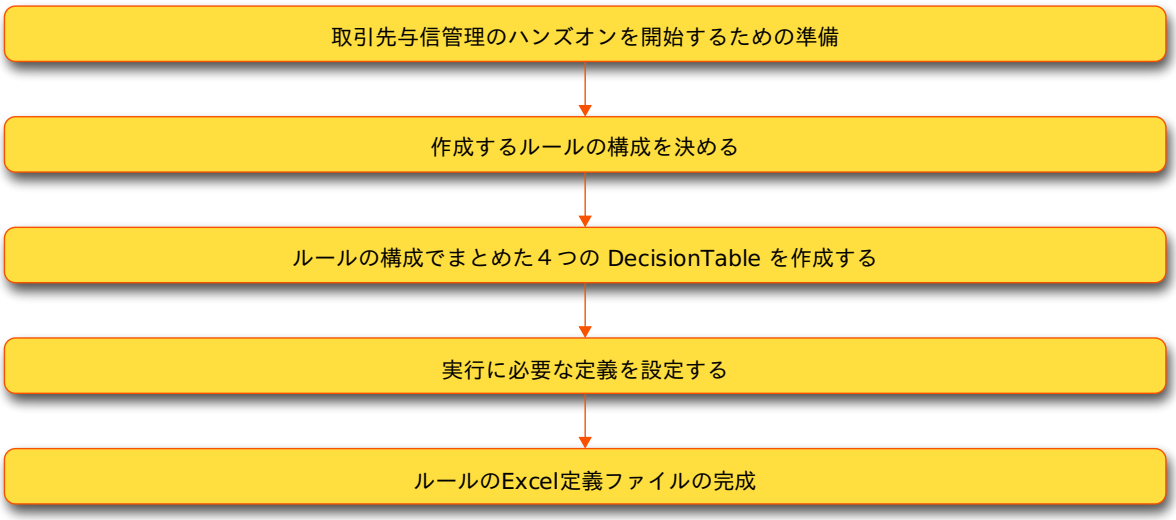
Condition		Conclusion	
与信枠		与信枠	
		=	::= {与信枠1}
<	0	=	0

DecisionTable2（DecisionTableMultiHit）では、一度評価した項目の値を変更することができます。

ルールのExcelファイルを作成する手順

今回は、 *OpenRules のテンプレート* の「汎用テンプレート」を変更しながらルール定義のExcelファイルを作成します。
このシナリオでは、以下の図の流れで作成していきます。

- 取引先に対する与信枠や信用度を評価するルールの作成の手順



取引先与信管理のハンズオンを開始するための準備

汎用テンプレートの編集を開始する

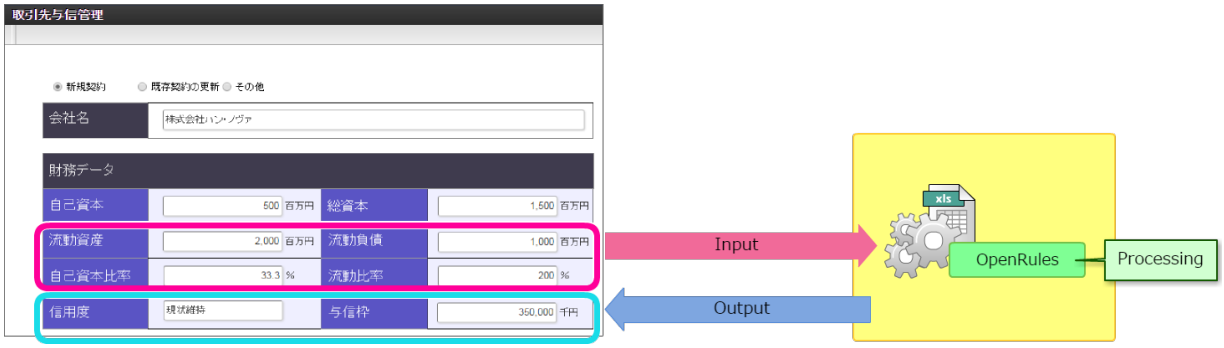
このハンズオンでは、ダウンロードの章で公開しているテンプレートを変更しながら、 OpenRules で動的処理者設定を定義していきます。
まずは、テンプレートファイルを入手しましょう。

1. *OpenRules のテンプレート* から「汎用テンプレート」をダウンロードしてください。
2. ファイルを別名で保存した後に、ファイルの編集を開始してください。
   （この手順では、例としてファイル名を「yoshinkanri.xls」として進めます。）

作成するルールの構成を決める

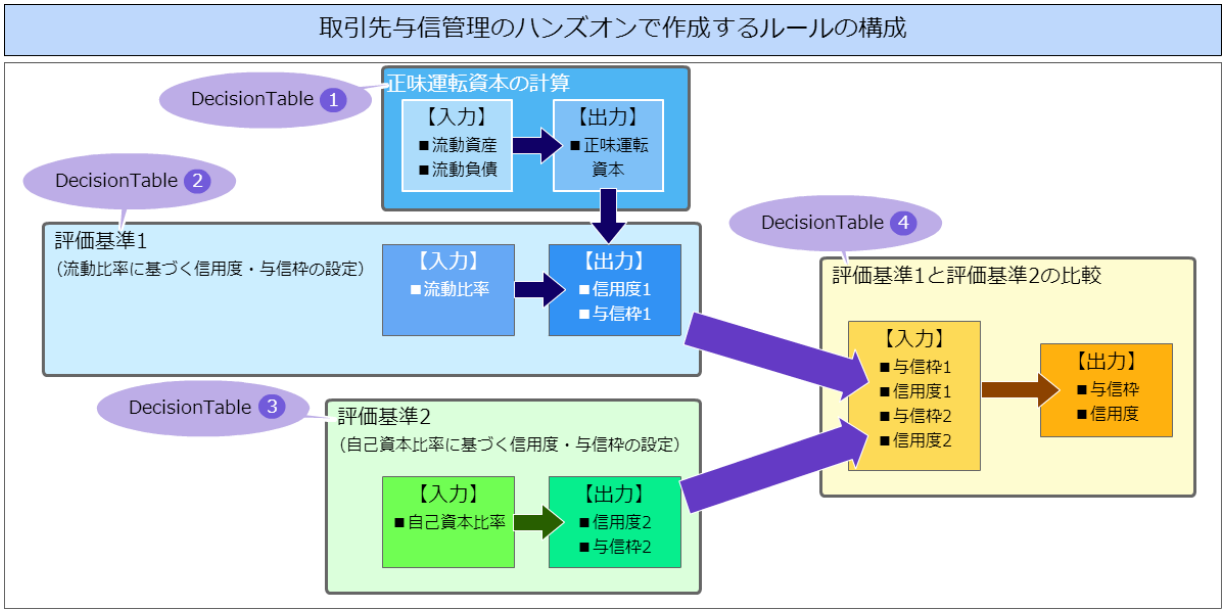
今回のハンズオンは、評価基準1の評価→評価基準2の評価など、複数の *DecisionTable* の実行を行う必要があるため、最初に *DecisionTable* の順序や単位といったルールの構成を決めます。

このハンズオンでのフォーム（画面）と OpenRules の値の受け渡しは、以下のイメージです。



- 入力
  画面から以下の項目の入力値を OpenRules に渡します。
  - 流動資産
  - 流動負債
  - 自己資本比率
- 処理
  受け渡された値に基づいて、以下の処理を実行します。
  - 評価基準1の評価
  - 評価基準2の評価
  - 評価基準1と評価基準2の比較
- 出力
  OpenRules から受け渡された値に基づいて、以下の結果を画面に表示します。
  - 信用度
  - 与信枠

上記の処理を実現するためには、以下の図のように、複数の *DecisionTable* をExcel上にまとめていきます。



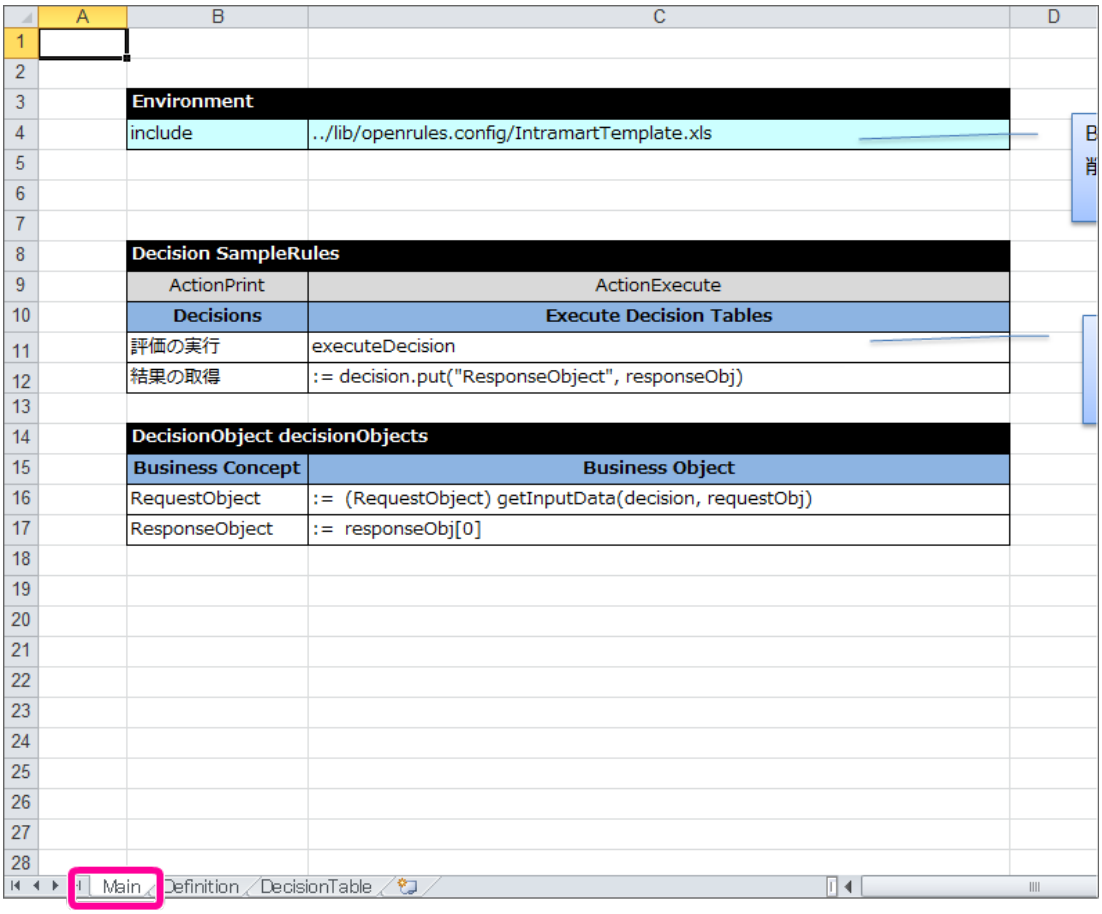
ハンズオンで実行したいルールの処理を整理すると、以下のように4つの *DecisionTable* を定義して実現できることが分かります。

1. Decisiontable(1)
   評価基準1の与信枠を計算するための「正味運転資本」の計算
   - 入力・・・流動資産、流動負債
   - 出力・・・正味運転資本
2. Decisiontable(2)
   評価基準1（流動比率に基づく信用度・与信枠の設定）の評価
   - 入力・・・「Decisiontable(1)」で算出した流動比率
   - 出力・・・信用度1、与信枠1（評価基準1に基づく信用度と与信枠の値）
3. Decisiontable(3)
   評価基準2（自己資本比率に基づく信用度・与信枠の設定）の評価
   - 入力・・・自己資本比率
   - 出力・・・信用度2、与信枠2（評価基準2に基づく信用度と与信枠の値）
4. Decisiontable(4)
   評価基準1と評価基準2の結果の比較に基づく結果（信用度・与信枠）の設定
   - 入力・・・「Decisiontable(2)」と「Decisiontable(3)」で取得した信用度と与信枠
   - 出力・・・入力で設定した与信枠と信用度の比較結果

DecisionTable の実行順を *Decision* にまとめる

上の手順で確認した通り、今回のハンズオンでは、4つの *DecisionTable* を利用することと実行順序がわかりましたので、 *Decision* にまとめていきましょう。

1. 編集中のExcelファイルの「Main」シートを表示してください。



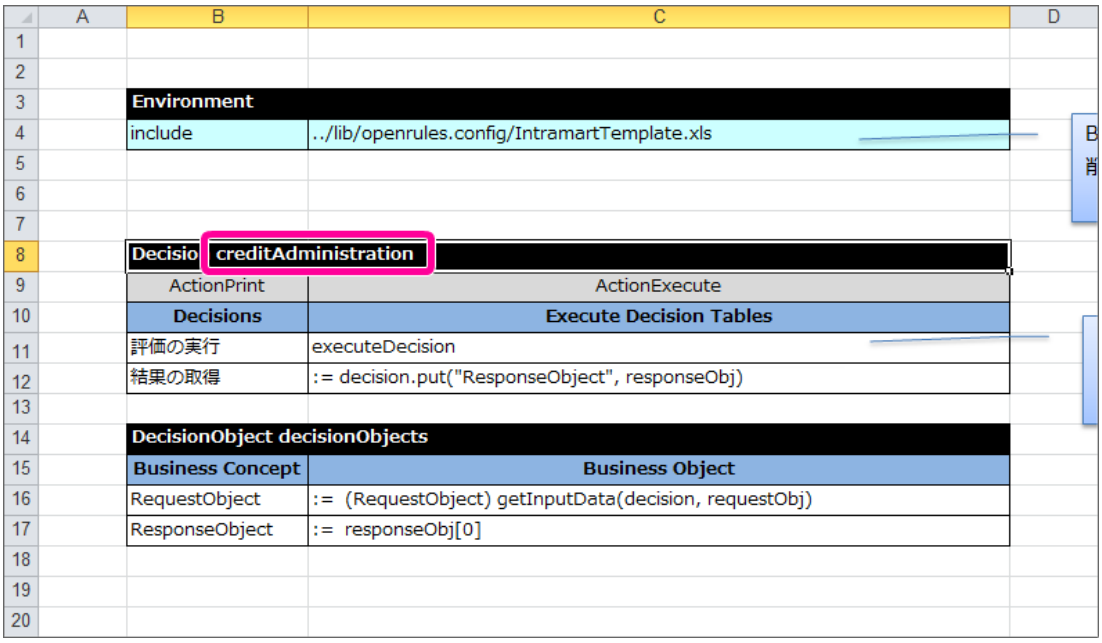
Environment	
include	../lib/openrules.config/IntramartTemplate.xls

Decision SampleRules	
ActionPrint	ActionExecute
Decisions	**Execute Decision Tables**
評価の実行	executeDecision
結果の取得	:= decision.put("ResponseObject", responseObj)

DecisionObject decisionObjects	
Business Concept	**Business Object**
RequestObject	:= (RequestObject) getInputData(decision, requestObj)
ResponseObject	:= responseObj[0]

Main Definition DecisionTable

2. *Decision* のテーブル名を「creditAdministration」に変更してください。

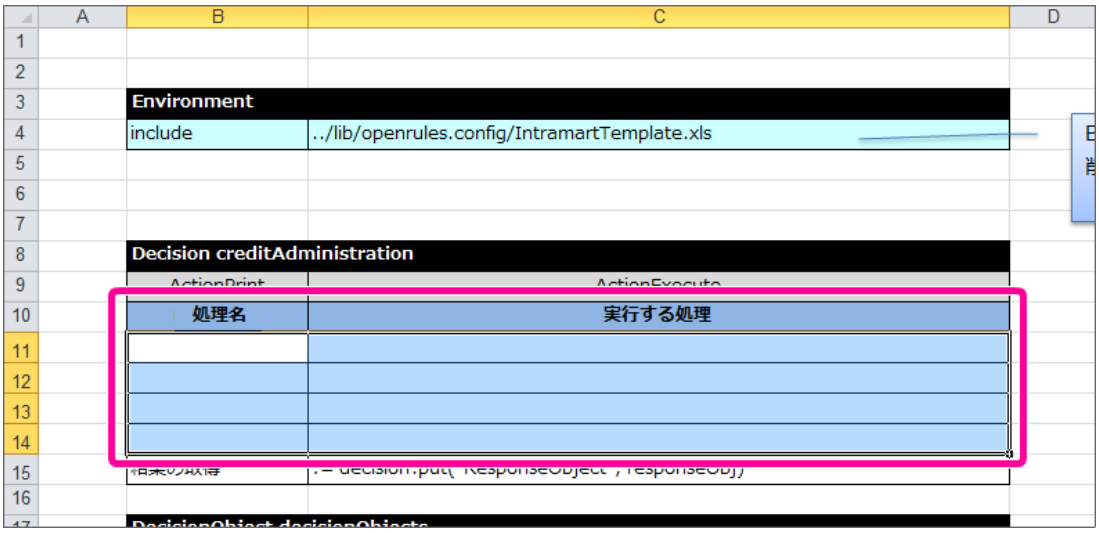


Environment	
include	../lib/openrules.config/IntramartTemplate.xls

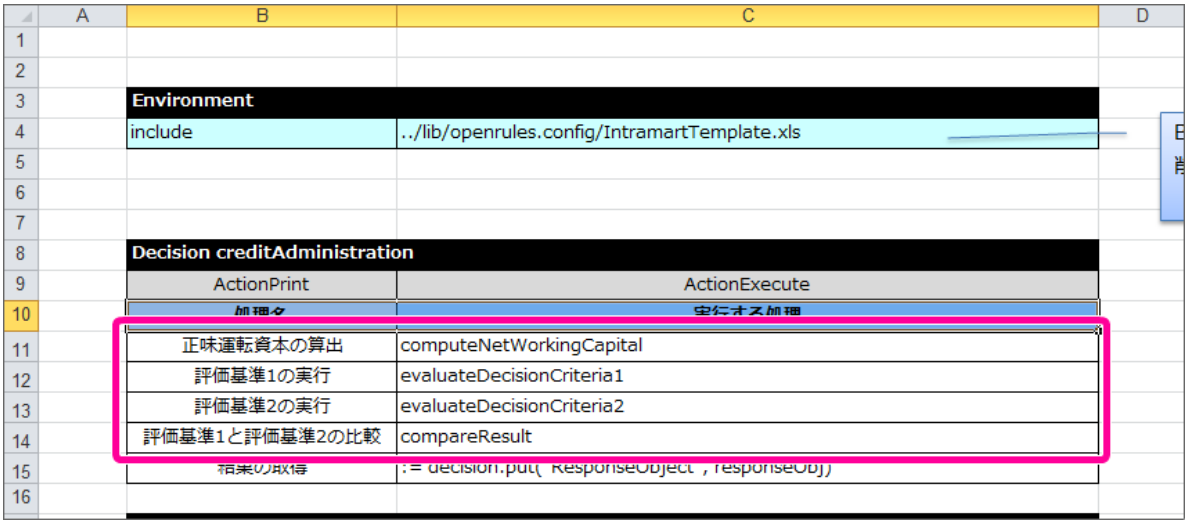
Decisio creditAdministration	
ActionPrint	ActionExecute
Decisions	**Execute Decision Tables**
評価の実行	executeDecision
結果の取得	:= decision.put("ResponseObject", responseObj)

DecisionObject decisionObjects	
Business Concept	**Business Object**
RequestObject	:= (RequestObject) getInputData(decision, requestObj)
ResponseObject	:= responseObj[0]

3. *Decision* に記載済みの「評価の実行」の行を削除してください。
代わりにハンズオンで作成する4つの *DecisionTable* 分の行として、4行追加してください。



4. 追加した行には、実行順に合わせて処理名と *DecisionTable* の名前を記述してください。



ActionPrint	ActionExecute
処理名	実行する処理
正味運転資本の算出	computeNetWorkingCapital
評価基準1の実行	evaluateDecisionCriteria1
評価基準2の実行	evaluateDecisionCriteria2
評価基準1と評価基準2の比較	compareResult

5. これで *Decision* ができましたので、ファイルを保存してください。
   引き続き、 *DecisionTable* を作成していきましょう。

ルールの構成でまとめた４つの DecisionTable を作成する

上の手順で確認した構成に含まれる4つの *DecisionTable* を順番に作成していきましょう。

正味運転資本を算出する DecisionTable を作成する

最初に *DecisionTable* の列を必要な形にし、条件と評価の項目名を設定していきましょう。

1. 編集中のExcelファイルの「DecisionTable」シートを表示してください。

Environment	
include	../lib/openrules.config/IntramartTemplate.xls

Decision creditAdministration	
ActionPrint	ActionExecute
処理名	**実行する処理**
正味運転資本の算出	computeNetWorkingCapital
評価基準1の実行	evaluateDecisionCriteria1
評価基準2の実行	evaluateDecisionCriteria2
評価基準1と評価基準2の比較	compareResult
結果の取得	:= decision.put("ResponseObject", responseObj)

2. テンプレートには、説明が添付されていますので、枠で囲まれた部分を削除してください。

DecisionTable executeDecision							
Condition		Condition		Conclusion		Conclusion	
年齢区分		**団体区分**		**料金タイプ**		**料金**	
Is	一般	Is	個人	Is	一般個人	Is	1200
Is	一般	Is	団体	Is	一般団体	Is	1000
Is	小学生	Is	個人	Is	学生個人	Is	600
Is	小学生	Is	団体	Is	学生団体	Is	500
				Is	一般個人	Is	1200

この表では、以下のような料金表を設定した例です。
一番下の何もconditionに設定されていない行は、いずれの
かったときに返却する結果です。（CASE文でのdefaultに

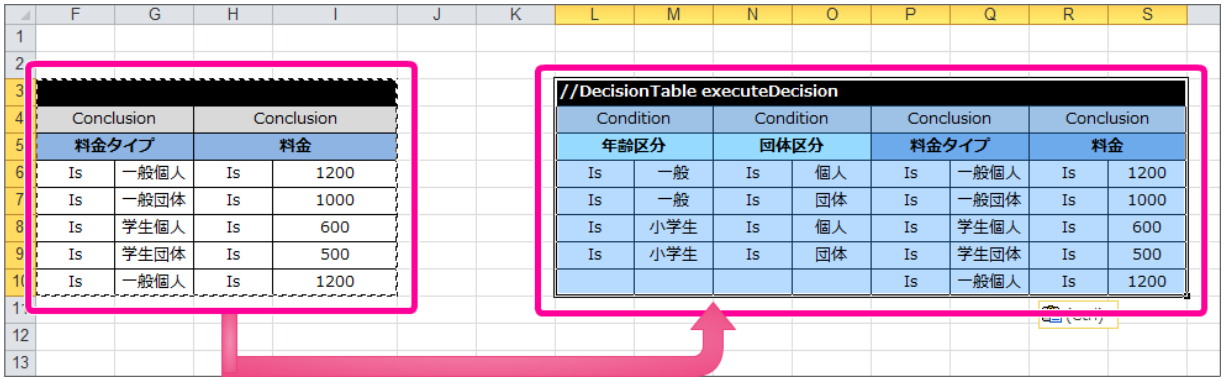
	個人	団体
一般	1,200円	1,000円
小学生	600円	500円

3. テンプレートの *DecisionTable* をコピーして作成できるように、テーブルのヘッダの先頭に “//”を記述してください。

//DecisionTable executeDecision							
Condition		Condition		Conclusion		Conclusion	
年齢区分		**団体区分**		**料金タイプ**		**料金**	
Is	一般	Is	個人	Is	一般個人	Is	1200
Is	一般	Is	団体	Is	一般団体	Is	1000
Is	小学生	Is	個人	Is	学生個人	Is	600
Is	小学生	Is	団体	Is	学生団体	Is	500
				Is	一般個人	Is	1200

> **コラム**
>
> OpenRules のコメントアウトの方法
>
> OpenRules で実行時に参照させたくないテーブルに対して、コメントアウトすることができます。
> 参照させたくないテーブルのメインヘッダに"//"を追加すると、そのテーブルは実行時に参照されません。

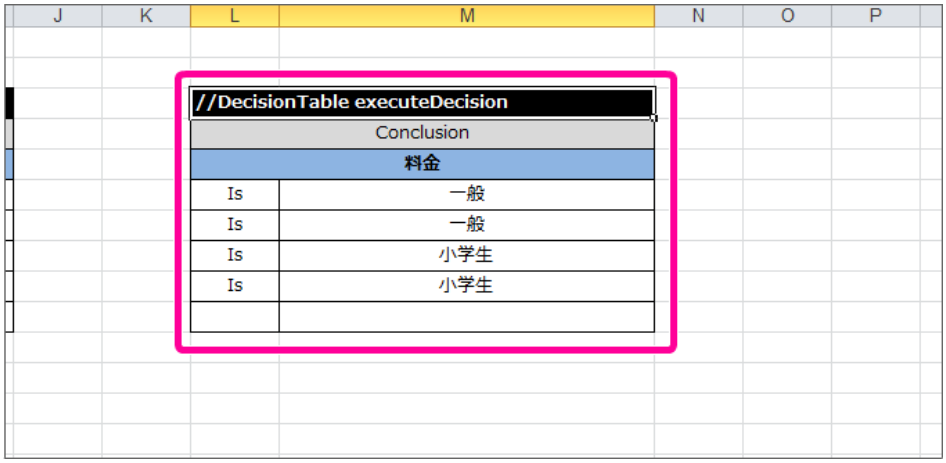
4. 先ほどヘッダを変更した *DecisionTable* をコピーして、1列・1行以上空けて貼り付けてください。



Conclusion		Conclusion	
料金タイプ		料金	
Is	一般個人	Is	1200
Is	一般団体	Is	1000
Is	学生個人	Is	600
Is	学生団体	Is	500
Is	一般個人	Is	1200

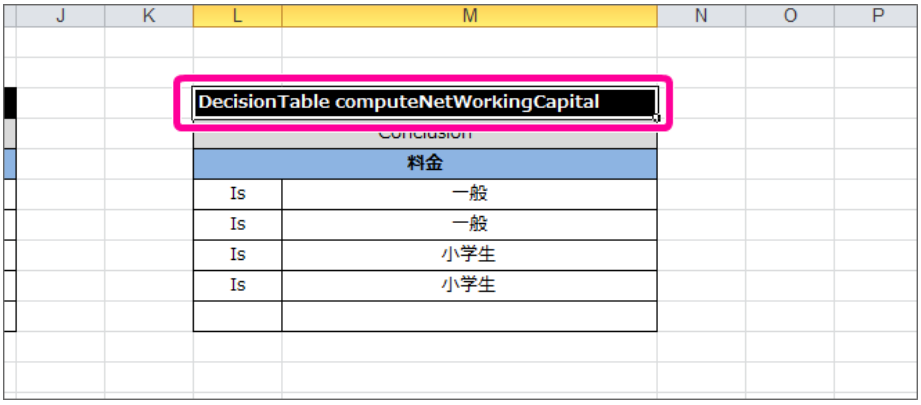
//DecisionTable executeDecision							
Condition		Condition		Conclusion		Conclusion	
年齢区分		団体区分		料金タイプ		料金	
Is	一般	Is	個人	Is	一般個人	Is	1200
Is	一般	Is	団体	Is	一般団体	Is	1000
Is	小学生	Is	個人	Is	学生個人	Is	600
Is	小学生	Is	団体	Is	学生団体	Is	500
				Is	一般個人	Is	1200

5. この *DecisionTable* は、無条件に正味運転資本の計算のみを行う *DecisionTable* として利用するため、 *Conclusion* だけの2列のレイアウトに変更してください。



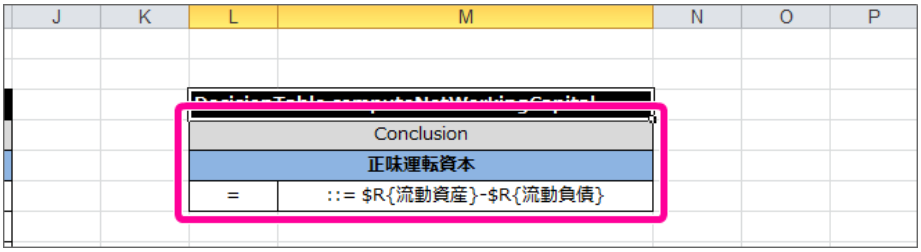
//DecisionTable executeDecision	
Conclusion	
料金	
Is	一般
Is	一般
Is	小学生
Is	小学生

6. テーブル名から、コメントアウトの"//"を削除し、 *Decision* で設定した名前（computeNetWorkingCapital）に変更してください。



DecisionTable computeNetWorkingCapital	
料金	
Is	一般
Is	一般
Is	小学生
Is	小学生

7. 設定する項目名（論理名）と式を以下のように設定してください。
この *DecisionTable* では、1つの式を実行させる処理だけを記述しますので、式を書いたら残りの行は削除してください。
項目同士の計算を行うには、 *$R(getReal)* を利用して式を書くようにしましょう。
今回のハンズオンで使用する数値項目は、double型で定義するため、 *$R(getReal)* としています。
int型の数値を扱う場合は、 *$I(getInt)* を利用してください。



Conclusion	
正味運転資本	
=	::= $R{流動資産}-$R{流動負債}

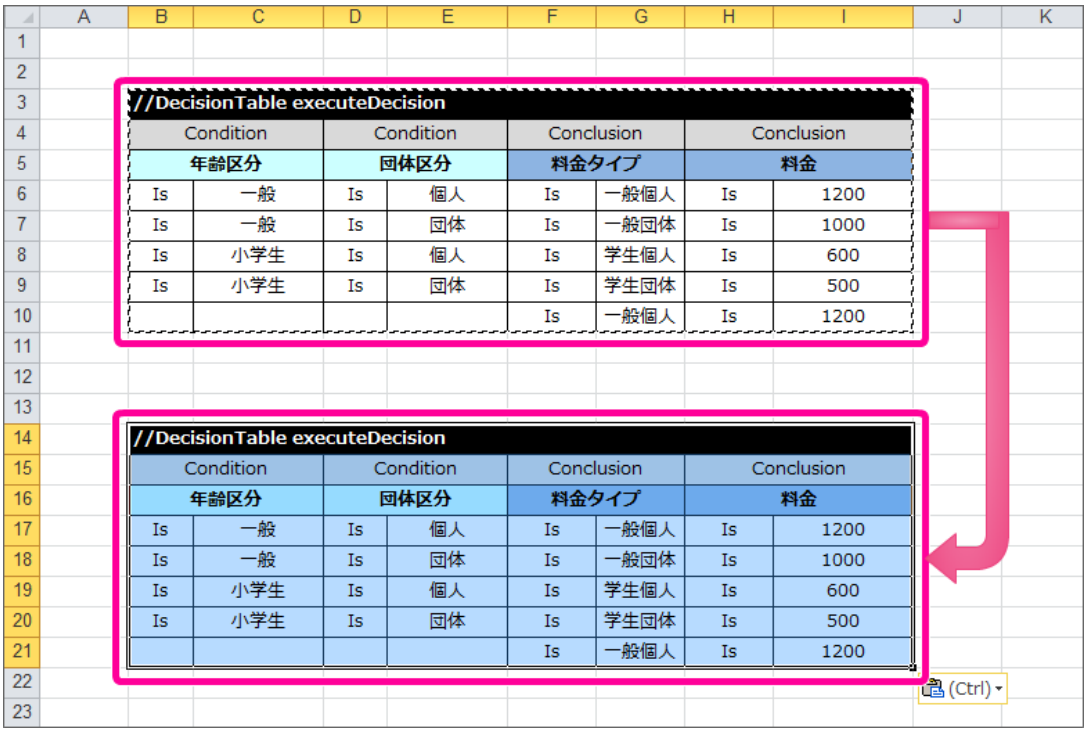
DecisionTable	computeNetWorkingCapital
	Conclusion
	正味運転資本
=	::= $R{流動資産}-$R{流動負債}

8. これで正味運転資本を算出するための *DecisionTable* ができましたので、保存してください。
   続いて、評価基準1の処理を行う *DecisionTable* を作成しましょう。

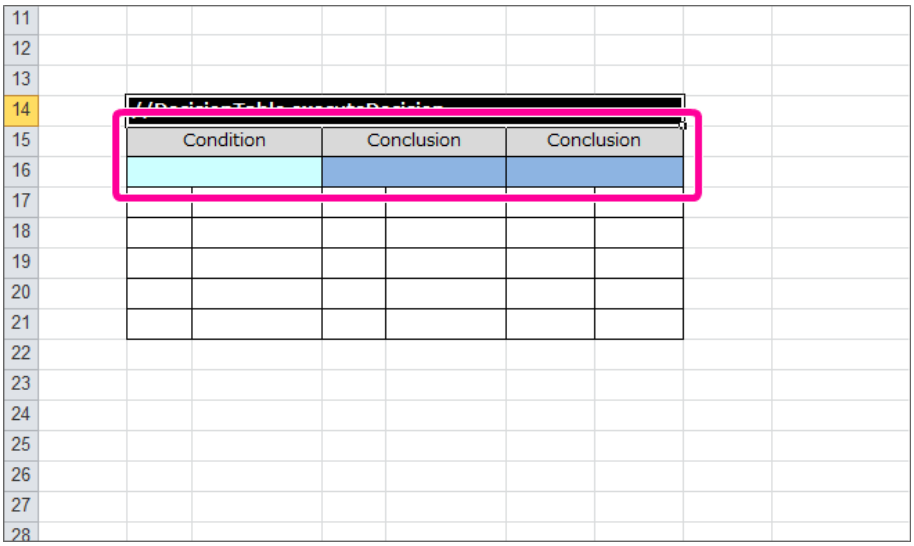
評価基準1を実行する *DecisionTable* を作成する

流動比率に基づいて、先ほど算出した正味運転資本から与信枠を計算し、信用度とともに返却する *DecisionTable* をまとめます。

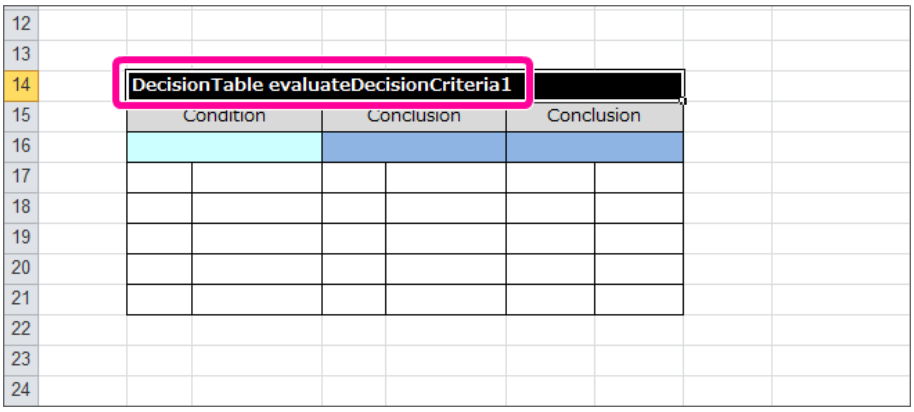
1. 先ほどと同様の手順で、テンプレートの *DecisionTable* をコピーして任意の場所に貼り付けてください。



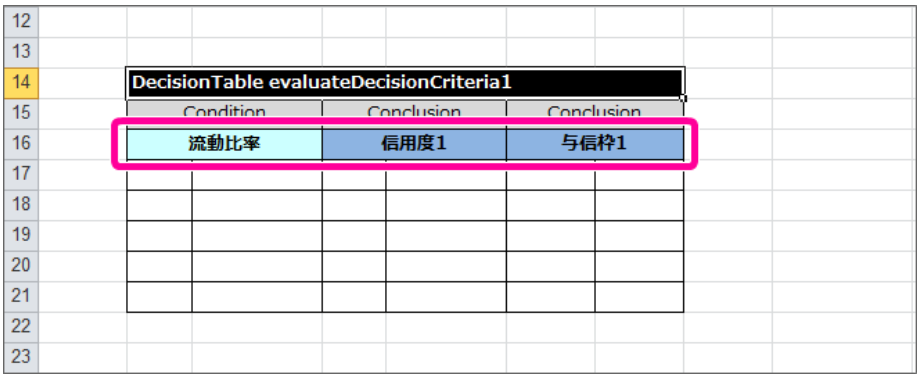
2. この *DecisionTable* は、入力項目に「流動比率」、出力項目に「信用度」「与信枠」を配置できるように、 *Condition* を1列、 *Conclusion* を2列にしてください。
   このとき、テーブルの2行目、3行目のサブヘッダ部分の結合セルの設定を解除した場合には、忘れずにセルの結合を行ってください。



3. テーブル名から、コメントアウトの"//"を削除し、 *Decision* で設定した名前（evaluateDecisionCriteria1）に変更してください。

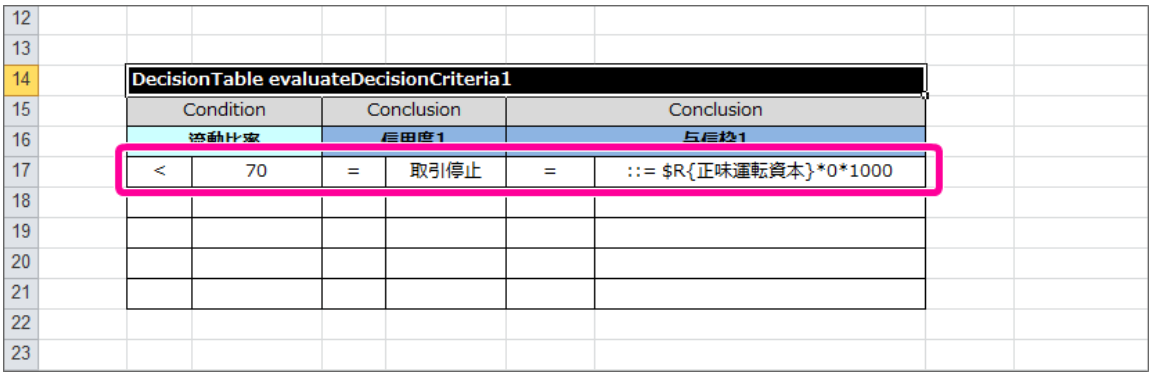


4. 設定する項目名（論理名）を以下のように設定してください。



DecisionTable evaluateDecisionCriteria1		
Condition	Conclusion	Conclusion
流動比率	信用度1	与信枠1

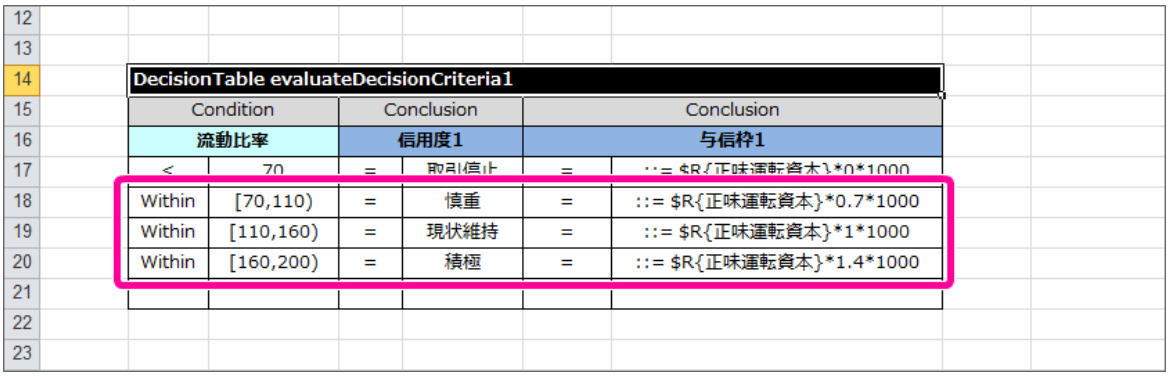
5. 流動比率が70%未満の場合を評価するための条件と結果を以下のように設定してください。



DecisionTable evaluateDecisionCriteria1		
Condition	Conclusion	Conclusion
流動比率	信用度1	与信枠1
< 70	= 取引停止	= ::= $R{正味運転資本}*0*1000

6. 続いて、流動比率が「○○以上△△未満」の場合の３つの条件と評価を設定してください。
このハンズオンの最初で確認した通り、「○○以上△△未満」を表現する場合には、演算子に”Within”を指定し、”[“と”)”を組み合わせて表現します。

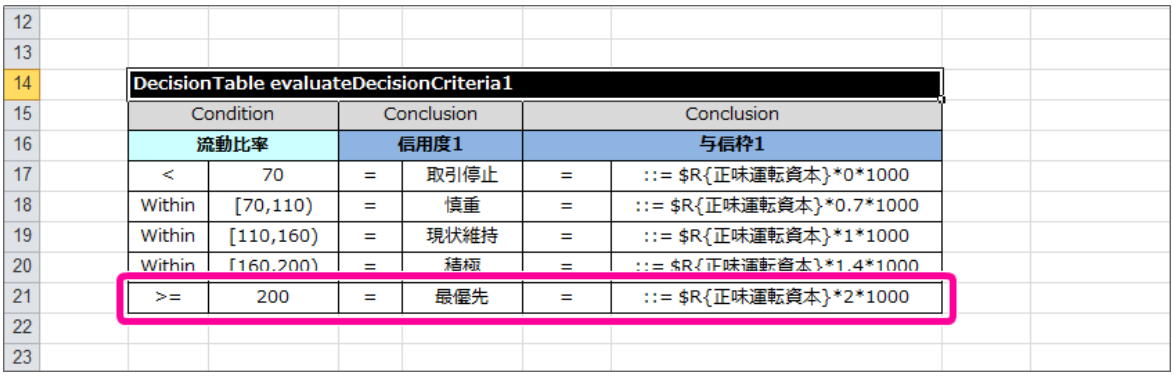
```
//「○○以上△△未満」の記法

[<下限値>,<上限値>)
```



DecisionTable evaluateDecisionCriteria1					
Condition		Conclusion		Conclusion	
流動比率		信用度1		与信枠1	
<	70	=	取引停止	=	::= $R{正味運転資本}*0*1000
Within	[70,110)	=	慎重	=	::= $R{正味運転資本}*0.7*1000
Within	[110,160)	=	現状維持	=	::= $R{正味運転資本}*1*1000
Within	[160,200)	=	積極	=	::= $R{正味運転資本}*1.4*1000

DecisionTable evaluateDecisionCriteria1		
Condition	Conclusion	Conclusion
流動比率	信用度1	与信枠1
Within [70,110)	= 慎重	= ::= $R{正味運転資本}*0.7*1000
Within [110,160)	= 現状維持	= ::= $R{正味運転資本}*1*1000
Within [160,200)	= 積極	= ::= $R{正味運転資本}*1.4*1000

7. 流動比率が200%以上の場合の条件と評価を設定してください。



DecisionTable evaluateDecisionCriteria1					
Condition		Conclusion		Conclusion	
流動比率		信用度1		与信枠1	
<	70	=	取引停止	=	::= $R{正味運転資本}*0*1000
Within	[70,110)	=	慎重	=	::= $R{正味運転資本}*0.7*1000
Within	[110,160)	=	現状維持	=	::= $R{正味運転資本}*1*1000
Within	[160,200)	=	積極	=	::= $R{正味運転資本}*1.4*1000
>=	200	=	最優先	=	::= $R{正味運転資本}*2*1000

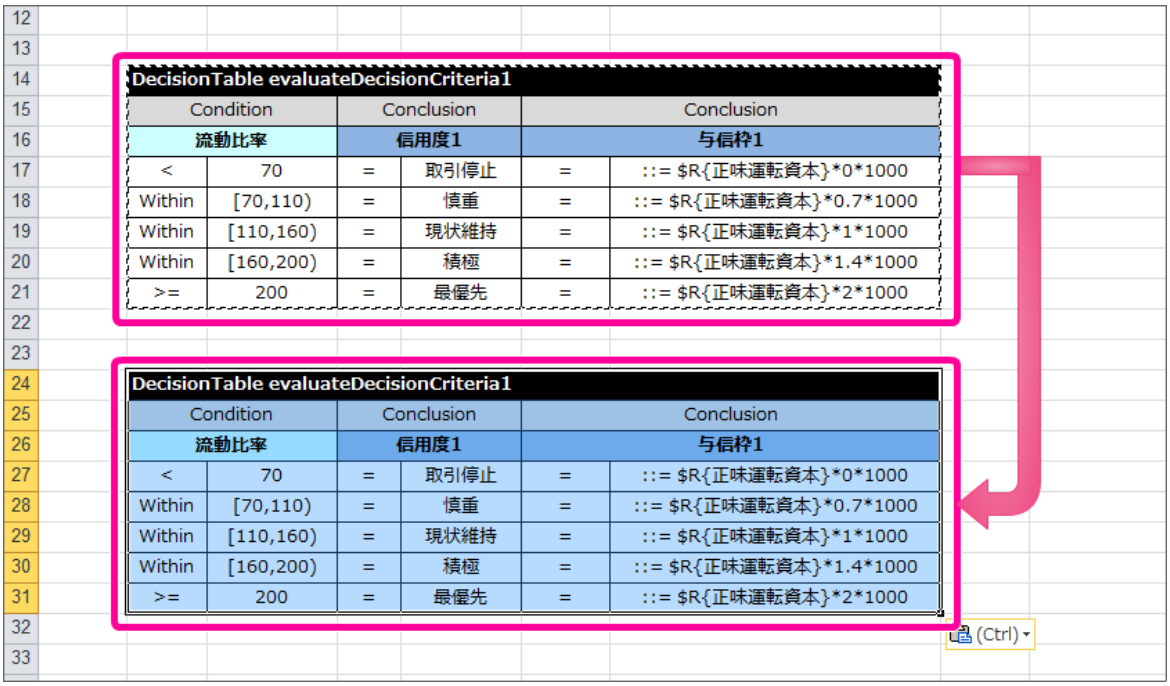
DecisionTable evaluateDecisionCriteria1		
Condition	Conclusion	Conclusion
流動比率	信用度1	与信枠1
>= 200	= 最優先	= ::= $R{正味運転資本}*2*1000

8. これで評価基準1を行うための *DecisionTable* ができましたので、保存してください。
   続いて、評価基準2の処理を行う *DecisionTable* を作成しましょう。

評価基準2を実行する *DecisionTable* を作成する

自己資本比率に基づいて、与信枠を計算し、信用度とともに返却する *DecisionTable* をまとめます。

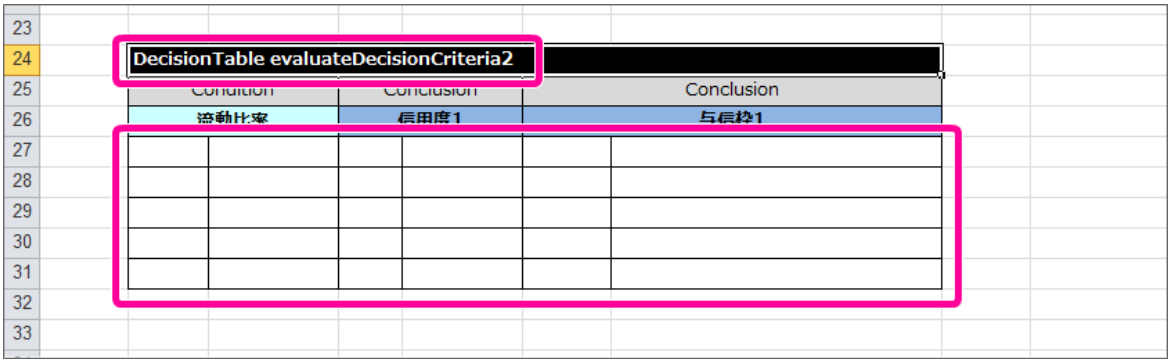
1. 先の手順で作成した *DecisionTable* 「evaluateDecisionCriteria1」と似たレイアウトのテーブルを作成していきますので、先に作成したテーブルをコピーして貼り付けてください。



DecisionTable evaluateDecisionCriteria1					
Condition		Conclusion		Conclusion	
流動比率		信用度1		与信枠1	
<	70	=	取引停止	=	::= $R{正味運転資本}*0*1000
Within	[70,110)	=	慎重	=	::= $R{正味運転資本}*0.7*1000
Within	[110,160)	=	現状維持	=	::= $R{正味運転資本}*1*1000
Within	[160,200)	=	積極	=	::= $R{正味運転資本}*1.4*1000
>=	200	=	最優先	=	::= $R{正味運転資本}*2*1000

DecisionTable evaluateDecisionCriteria1					
Condition		Conclusion		Conclusion	
流動比率		信用度1		与信枠1	
<	70	=	取引停止	=	::= $R{正味運転資本}*0*1000
Within	[70,110)	=	慎重	=	::= $R{正味運転資本}*0.7*1000
Within	[110,160)	=	現状維持	=	::= $R{正味運転資本}*1*1000
Within	[160,200)	=	積極	=	::= $R{正味運転資本}*1.4*1000
>=	200	=	最優先	=	::= $R{正味運転資本}*2*1000

2. テーブル名を *Decision* で設定した名前（evaluateDecisionCriteria2）に変更してください。
   また、明細の行の内容を削除してください。



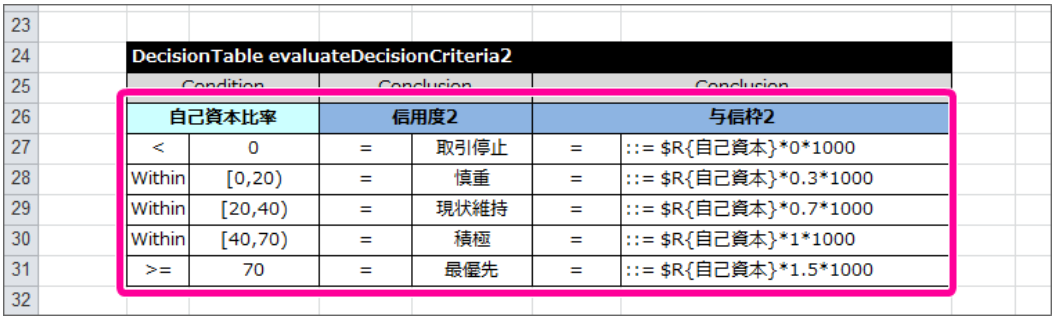
DecisionTable evaluateDecisionCriteria2					
Condition		Conclusion		Conclusion	
流動比率		信用度1		与信枠1	

3. 設定する項目名（論理名）を以下の通りに設定してください。

DecisionTable evaluateDecisionCriteria2					
Condition		Conclusion		Conclusion	
自己資本比率		信用度2		与信枠2	

DecisionTable evaluateDecisionCriteria2		
Condition	Conclusion	Conclusion
自己資本比率	信用度2	与信枠2

4. 評価基準1の *DecisionTable* と同様に、条件と評価を設定してください。



DecisionTable evaluateDecisionCriteria2					
Condition		Conclusion		Conclusion	
自己資本比率		信用度2		与信枠2	
<	0	=	取引停止	=	::= $R{自己資本}*0*1000
Within	[0,20)	=	慎重	=	::= $R{自己資本}*0.3*1000
Within	[20,40)	=	現状維持	=	::= $R{自己資本}*0.7*1000
Within	[40,70)	=	積極	=	::= $R{自己資本}*1*1000
>=	70	=	最優先	=	::= $R{自己資本}*1.5*1000

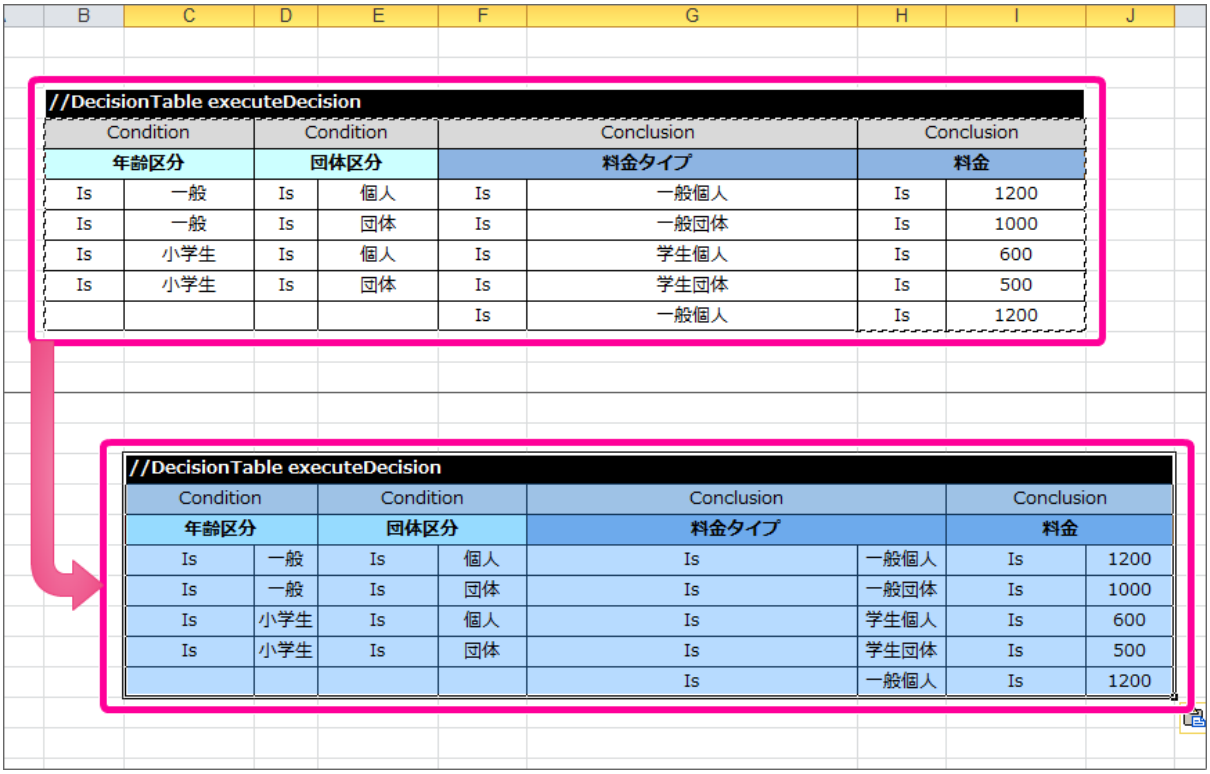
DecisionTable evaluateDecisionCriteria2		
Condition	Conclusion	Conclusion
自己資本比率	信用度2	与信枠2
< 0	= 取引停止	= ::= $R{自己資本}*0*1000
Within [0,20)	= 慎重	= ::= $R{自己資本}*0.3*1000
Within [20,40)	= 現状維持	= ::= $R{自己資本}*0.7*1000
Within [40,70)	= 積極	= ::= $R{自己資本}*1*1000
>= 70	= 最優先	= ::= $R{自己資本}*1.5*1000

5. これで評価基準2を行うための *DecisionTable* ができましたので、保存してください。
   この後は、評価基準1と評価基準2の比較を行う *DecisionTable* を作成しましょう。

評価基準1の結果と評価基準2の結果を比較する *DecisionTable* を作成する

これまで作成した評価基準1の *DecisionTable* の結果と、評価基準2の *DecisionTable* の結果を比較する *DecisionTable* を作成していきましょう。

1. テンプレートの *DecisionTable* をコピーして任意の場所に貼り付けてください。



//DecisionTable executeDecision							
Condition		Condition		Conclusion		Conclusion	
年齢区分		団体区分		料金タイプ		料金	
Is	一般	Is	個人	Is	一般個人	Is	1200
Is	一般	Is	団体	Is	一般団体	Is	1000
Is	小学生	Is	個人	Is	学生個人	Is	600
Is	小学生	Is	団体	Is	学生団体	Is	500
				Is	一般個人	Is	1200

//DecisionTable executeDecision							
Condition		Condition		Conclusion		Conclusion	
年齢区分		団体区分		料金タイプ		料金	
Is	一般	Is	個人	Is	一般個人	Is	1200
Is	一般	Is	団体	Is	一般団体	Is	1000
Is	小学生	Is	個人	Is	学生個人	Is	600
Is	小学生	Is	団体	Is	学生団体	Is	500
				Is	一般個人	Is	1200

2. この *DecisionTable* では、評価基準1の結果と評価基準2の結果を比較した後に、「与信枠が負数となるときには0に設定する」処理を設定します。
   ここで作成する *DecisionTable* では、比較を行い、最後の結果を0にする処理まで行います。
   テーブル名からコメントアウトの"//"を削除し、テーブルのキーワードを *DecisionTable2* に変更してください。

DecisionTable2	xecuteDecision						
Condition		Condition		Conclusion		Conclusion	
年齢区分		団体区分		料金タイプ		料金	
Is	一般	Is	個人	Is	一般個人	Is	1200
Is	一般	Is	団体	Is	一般団体	Is	1000
Is	小学生	Is	個人	Is	学生個人	Is	600
Is	小学生	Is	団体	Is	学生団体	Is	500
				Is	一般個人	Is	1200

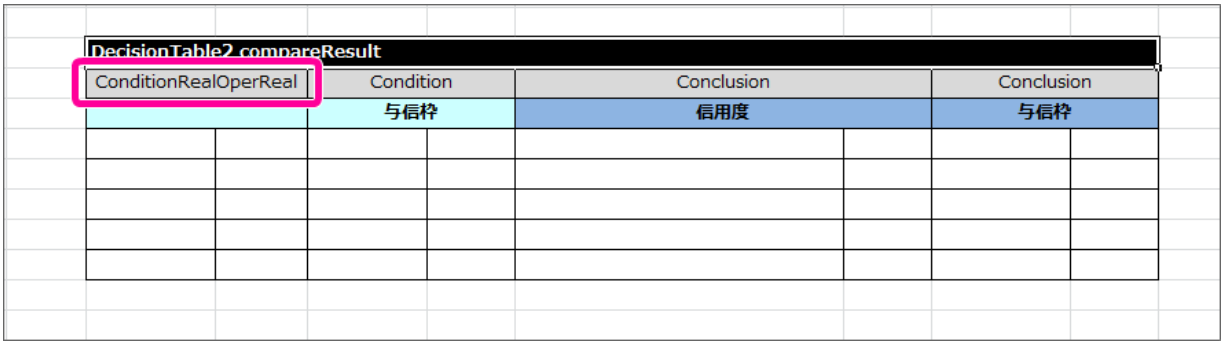
> **コラム**
>
> *DecisionTable2* は、今までのハンズオンで作成してきた *DecisionTable* と処理方法が少し異なります。
> 詳細については「 *OpenRules のキーワードリファレンス* 」で確認してください。

3. テーブル名を *Decision* で設定した名前（compareResult）に変更してください。
   また、サブヘッダや明細の内容を以下の内容に変更してください。
   このとき、1番左の列が空白となっている点と、 *Condition* と *Conclusion* に「与信枠」が2度登場している点については、後で説明していきますので、そのままにしておいてください。

DecisionTable2 compareResult							
		Condition		Conclusion		Conclusion	
		与信枠		信用度		与信枠	

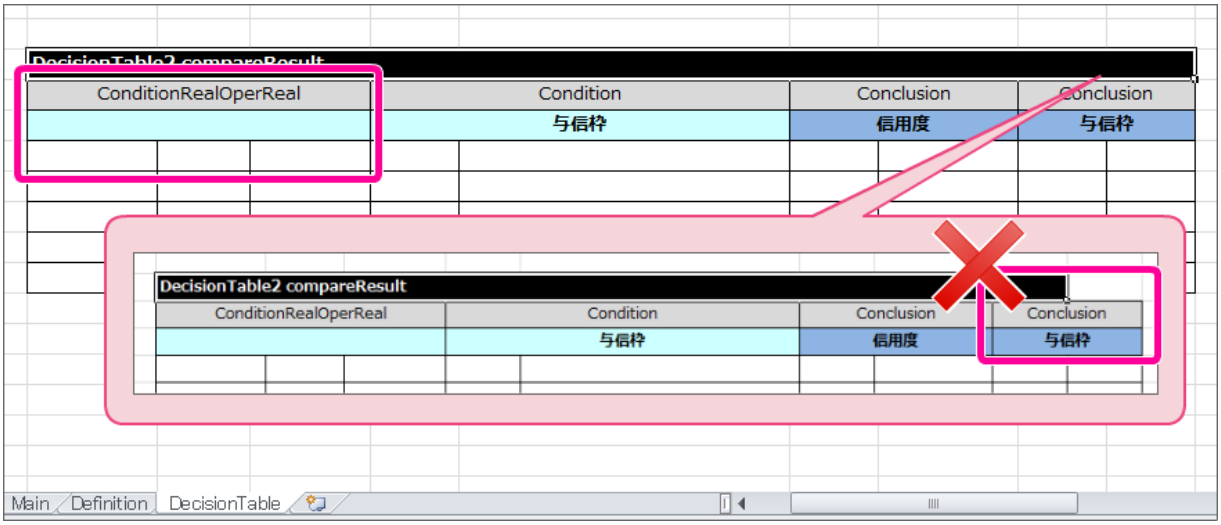
DecisionTable2 compareResult			
	Condition	Conclusion	Conclusion
	与信枠	信用度	与信枠

4. 最初に「評価基準1の実行」の結果と「評価基準2の実行」の結果を比較する条件を記述してください。
   項目同士の比較を条件に指定する場合には、 *Condition* ではなく専用のキーワードを利用します。
   今回の「与信枠1」「与信枠2」は、double型として扱うため、 *ConditionRealOperReal* と入力してください。

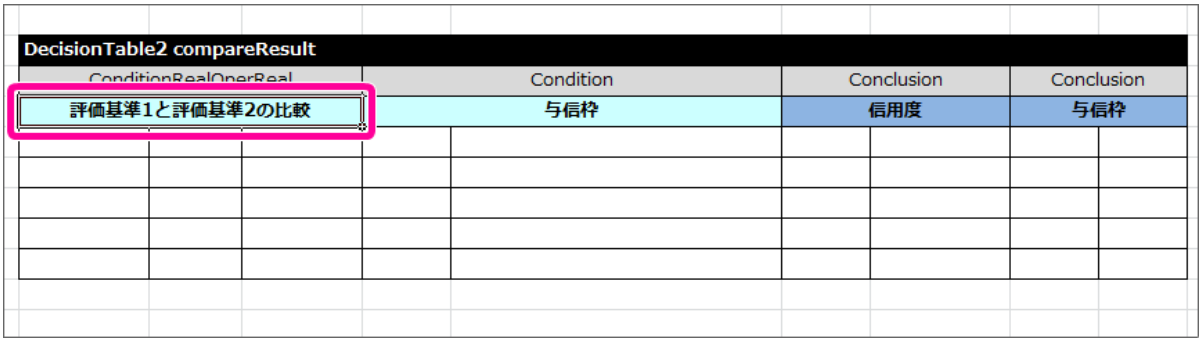


DecisionTable2 compareResult							
ConditionRealOperReal		Condition		Conclusion		Conclusion	
		与信枠		信用度		与信枠	

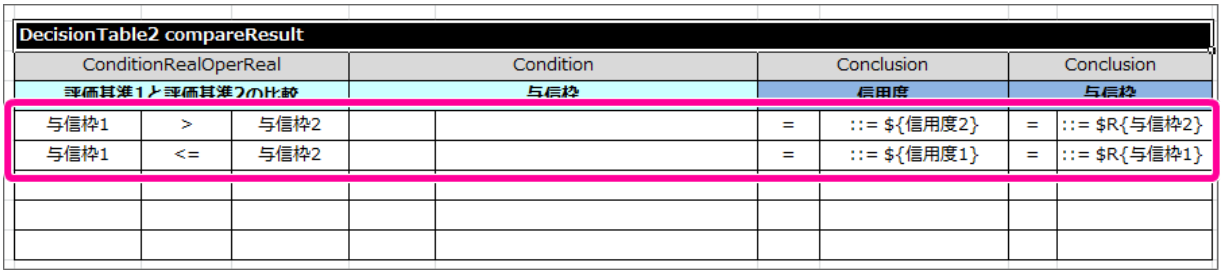
5. *ConditionRealOperReal* では、「左の比較対象の項目名」「演算子」「右の比較対象の項目名」という３つの項目で構成します。
   テーブルに1列挿入し、 *ConditionRealOperReal* の列が３つのセルとなるように設定してください。
   このとき、ヘッダから列がはみ出した場合には、セル結合を実行してすべての列がヘッダの列の範囲に含まれるようにしてください。



6. サブヘッダの内容には、 わかりやすくするために「評価基準1と評価基準2の比較」と入力してください。

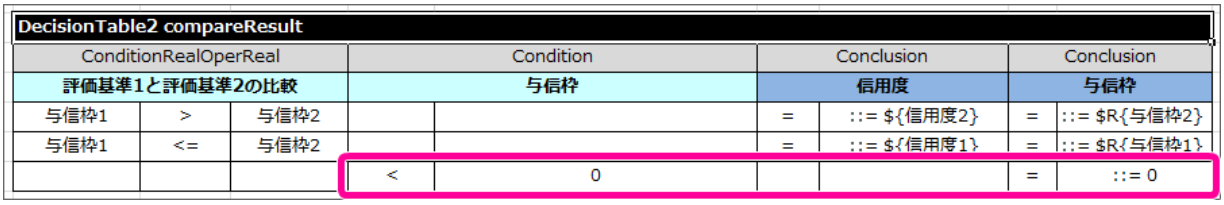


7. 条件について、「評価基準1の結果が大きい」パターンと「評価基準2の結果が同じか等しくなる」パターンに分けて条件を入力してください。
結果については、 *Conclusion* を設定し、値が小さい方の評価基準の項目を設定するようにしてください。



DecisionTable2 compareResult				
ConditionRealOperReal		Condition	Conclusion	Conclusion
評価基準1と評価基準2の比較		与信枠	信用度	与信枠
与信枠1	>	与信枠2	= ::= ${信用度2}	= ::= ${与信枠2}
与信枠1	<=	与信枠2	= ::= ${信用度1}	= ::= ${与信枠1}

8. 最後に、比較した結果からセットした「与信枠」が負数の場合には0をセットするという処理を追加します。
この処理を追加するために、条件（ *Condition* ）とした与信枠に「0未満」を表す条件を設定してください。



DecisionTable2 compareResult				
	ConditionRealOperReal	Condition	Conclusion	Conclusion
	評価基準1と評価基準2の比較	与信枠	信用度	与信枠
		< 0		= ::= 0

9. これで必要な *DecisionTable* をすべて作成しましたので、保存してください。

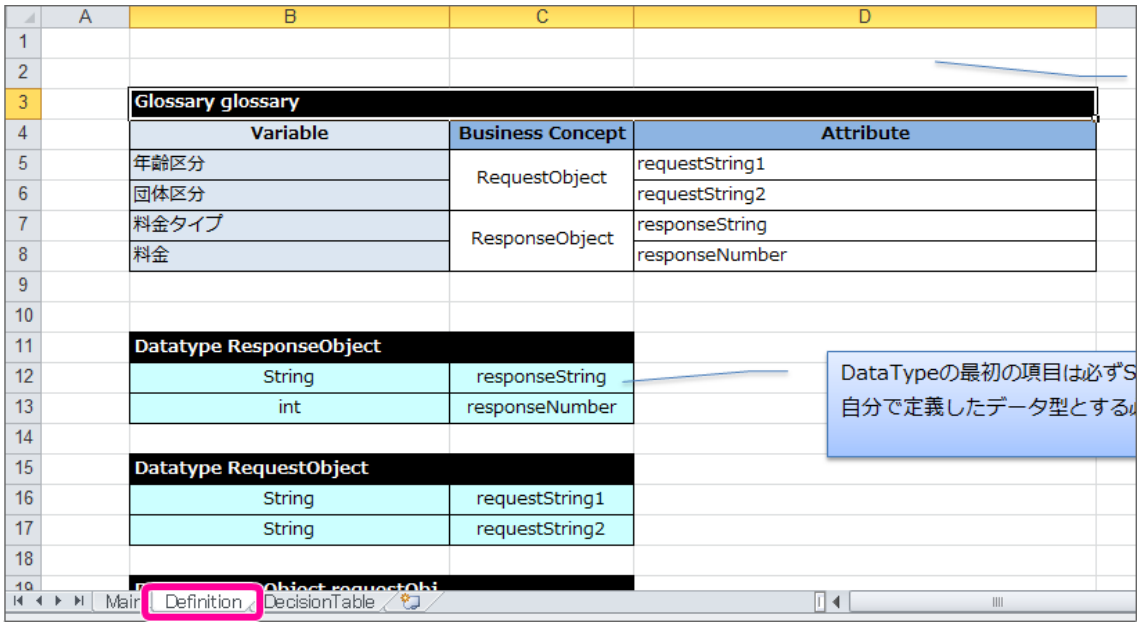
実行に必要な定義を設定する

ルールの中心となる *Decision* 、 *DecisionTable* が完成しましたので、実行に必要なその他の定義を設定していきましょう。

Glossary に利用する項目を定義する

DecisionTable で使っている項目を確認しながら、 *Glossary* を定義していきましょう。

1. 最初に、今回作成した *DecisionTable* で登場した項目名を確認しましょう。
   4つの *DecisionTable* で利用している以下の項目を定義する必要があります。
   - 正味運転資本の算出
     - 流動資産
     - 流動負債
     - 正味運転資本
   - 評価基準1の実行
     - 流動比率
     - 信用度1
     - 与信枠1
     - 正味運転資本
   - 評価基準2の実行
     - 自己資本比率
     - 信用度2
     - 与信枠2
     - 自己資本
   - 評価基準1と評価基準2の比較
     - 与信枠1
     - 与信枠2
     - 信用度1
     - 信用度2
     - 与信枠
     - 信用度
2. 上で洗い出した項目について、「画面から渡される項目」を「入力項目のグループ」（RequestObject）、「画面に返却する項目」を「出力項目のグループ」（ResponseObject）、「どちらにも属さない項目」を「処理用のグループ」（Internal）に分類してください。
   - 入力項目のグループ（RequestObject）
     - 流動資産
     - 流動負債
     - 流動比率
     - 自己資本
     - 自己資本比率
   - 出力項目のグループ（ResponseObject）
     - 与信枠
     - 信用度
   - 処理のグループ（Internal）
     - 信用度1
     - 信用度2
     - 与信枠1
     - 与信枠2
     - 正味運転資本
3. *Glossary* を作成するために、「Definition」シートを表示してください。



Glossary glossary		
Variable	Business Concept	Attribute
年齢区分	RequestObject	requestString1
団体区分		requestString2
料金タイプ	ResponseObject	responseString
料金		responseNumber

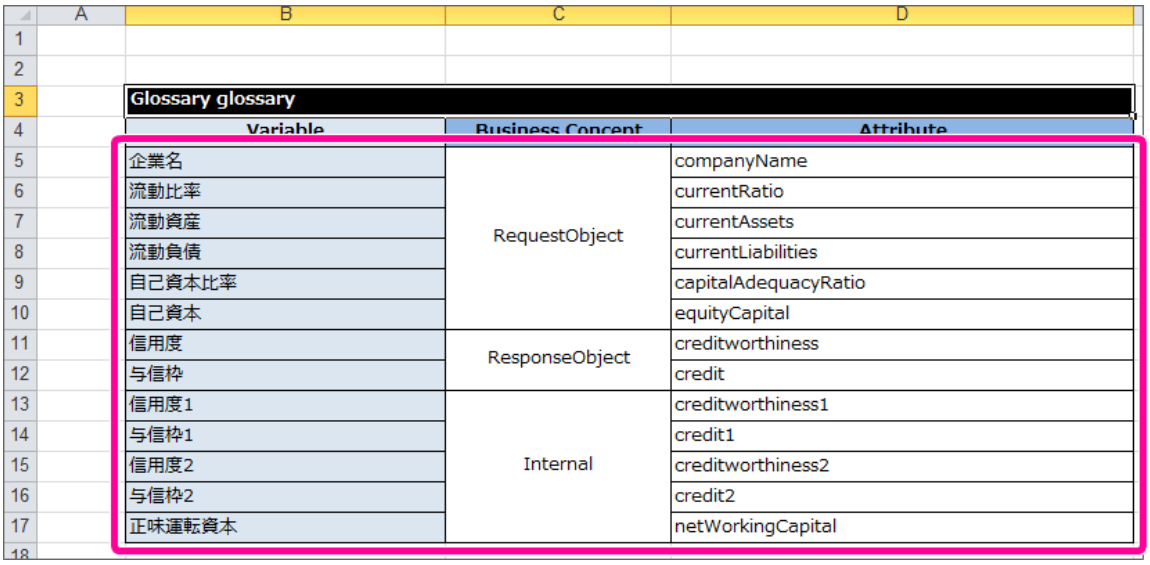
Datatype ResponseObject	
String	responseString
int	responseNumber

Datatype RequestObject	
String	requestString1
String	requestString2

DataTypeの最初の項目は必ずS
自分で定義したデータ型とする

Main | Definition | DecisionTable

4. 整理した項目と分類に基づいて、*Glossary* を作成していきますが、「入力項目のグループ」（RequestObject）については、すべて数値項目で構成されています。
   そのままRequestObjectを数値項目のみで定義すると OpenRules の制約事項に抵触するため、String項目として、「企業名」を追加して作成してください。



Glossary glossary		
Variable	Business Concept	Attribute
企業名	RequestObject	companyName
流動比率		currentRatio
流動資産		currentAssets
流動負債		currentLiabilities
自己資本比率		capitalAdequacyRatio
自己資本		equityCapital
信用度	ResponseObject	creditworthiness
与信枠		credit
信用度1	Internal	creditworthiness1
与信枠1		credit1
信用度2		creditworthiness2
与信枠2		credit2
正味運転資本		netWorkingCapital

5. *Glossary* が作成できましたので、保存してください。
   引き続き、 *Glossary* に基づいて、 *Datatype* と *Data/Variable* を定義しましょう。

Glossary から *Datatype* と *Data/Variable* を定義する

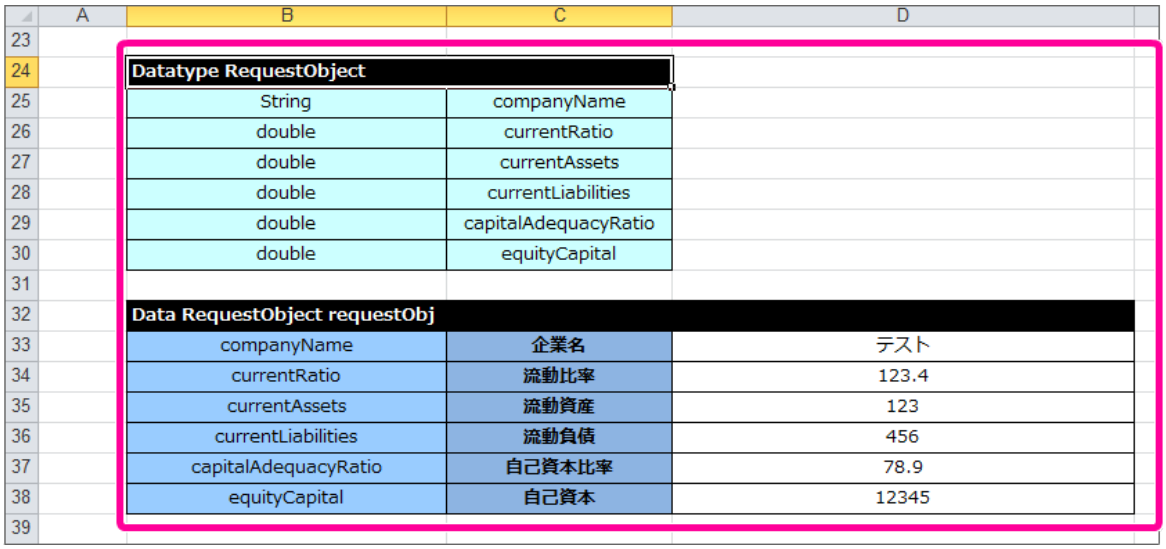
Glossary を利用して *Datatype* と *Data/Variable* を定義していきましょう。

1. 作成した *Glossary* から「入力項目のグループ」（RequestObject）の定義を以下の通りに設定してください。
   - *Datatype*

Datatype RequestObject	
String	companyName
double	currentAssets
double	currentLiabilities
double	currentRatio
double	equityCapital
double	capitalAdequacyRatio

   - *Data/Variable*

Data RequestObject requestObj		
companyName	企業名	テスト企業
currentAssets	流動資産	1000000
currentLiabilities	流動負債	400000
currentRatio	流動比率	40
equityCapital	自己資本	2000000
capitalAdequacyRatio	自己資本比率	40



Datatype RequestObject	
String	companyName
double	currentRatio
double	currentAssets
double	currentLiabilities
double	capitalAdequacyRatio
double	equityCapital

Data RequestObject requestObj		
companyName	**企業名**	テスト
currentRatio	**流動比率**	123.4
currentAssets	**流動資産**	123
currentLiabilities	**流動負債**	456
capitalAdequacyRatio	**自己資本比率**	78.9
equityCapital	**自己資本**	12345

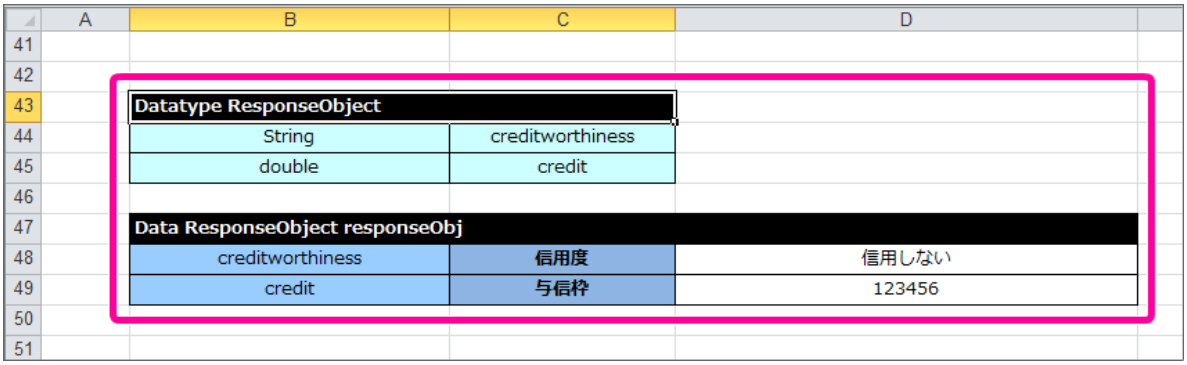
> **コラム**
>
> *Data/Variable* は、このハンズオンのように項目数が多くなると、横に長くなって表示しづらくなるため、行・列を入れ替えて設定することができます。
> この場合にも、メインヘッダの列の範囲に必要な列が含まれるようにセル結合を行ってください。

2. 作成した *Glossary* から「出力項目のグループ」（ResponseObject）の定義を設定してください。
   - *Datatype*

Datatype ResponseObject	
String	creditworthiness
double	credit

   - *Data/Variable*

Data ResponseObject	**responseObj**	
creditworthiness	信用度	信用度初期値
credit	与信枠	0



Datatype ResponseObject	
String	creditworthiness
double	credit

Data ResponseObject responseObj		
creditworthiness	**信用度**	信用しない
credit	**与信枠**	123456

3. 作成した *Glossary* から「処理のグループ」（Internal）の定義を設定してください。
   - *Datatype*

Datatype Internal	
String	creditworthiness1
String	creditworthiness2
double	credit1
double	credit2
double	netWorkingCapital

- *Data/Variable*

Data Internal internal		
creditworthiness1	信用度1	信用度初期値
creditworthiness2	信用度2	信用度初期値
credit1	与信枠1	0
credit2	与信枠2	0
netWorkingCapital	正味運転資本	0

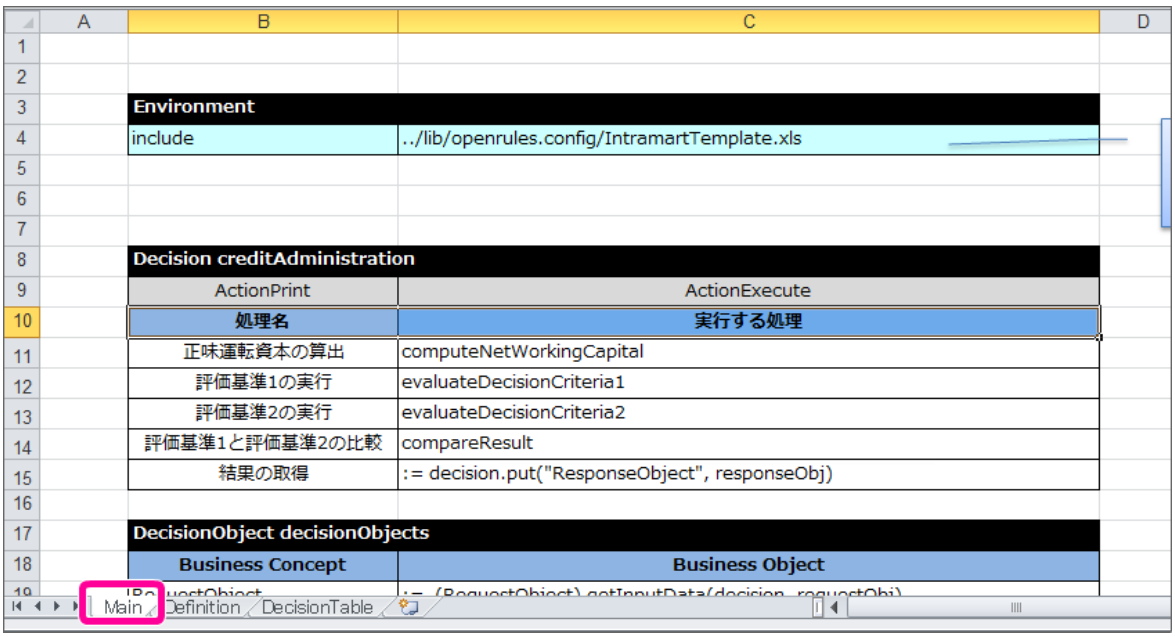
Datatype Internal	
String	creditworthiness1
double	credit1
String	creditworthiness2
double	credit2
double	netWorkingCapital

Data Internal internal		
creditworthiness1	信用度1	信用しない1
credit1	与信枠1	11111
creditworthiness2	信用度2	信用しない2
credit2	与信枠2	22222
netWorkingCapital	正味運転資本	33333

DecisionObject を定義してルールを完成させる

最後に *DecisionObject* を定義して、Excelのルール定義ファイルを完成させましょう。

1. 「Main」シートを表示してください。



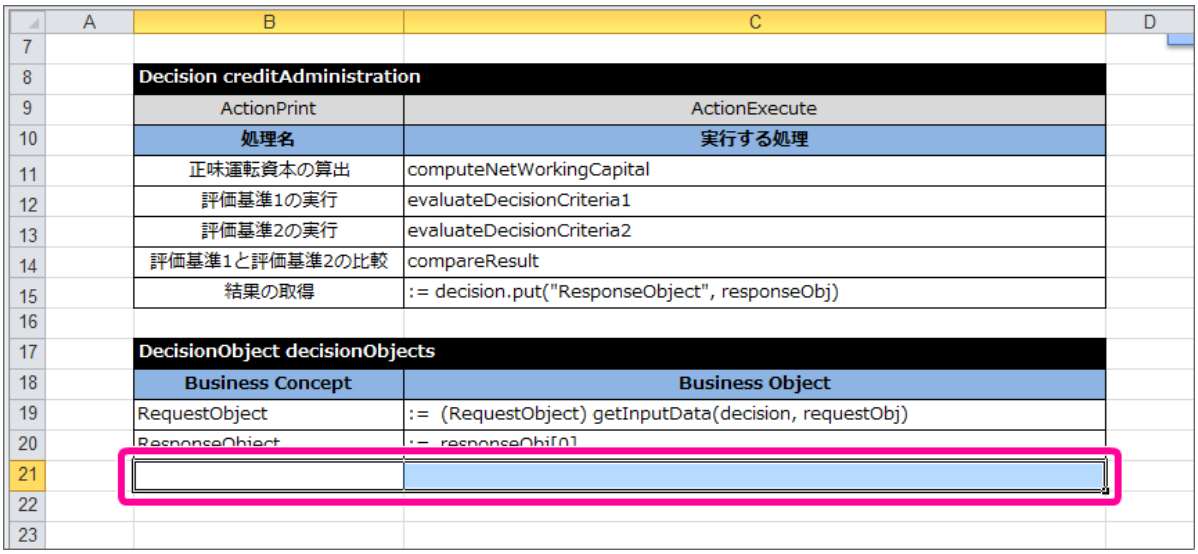
Environment	
include	../lib/openrules.config/IntramartTemplate.xls

Decision creditAdministration	
ActionPrint	ActionExecute
処理名	実行する処理
正味運転資本の算出	computeNetWorkingCapital
評価基準1の実行	evaluateDecisionCriteria1
評価基準2の実行	evaluateDecisionCriteria2
評価基準1と評価基準2の比較	compareResult
結果の取得	:= decision.put("ResponseObject", responseObj)

DecisionObject decisionObjects	
Business Concept	Business Object

2. 「Main」シートで *DecisionObject* を設定していきます。
入力項目、出力項目のオブジェクトについては、テンプレートと同じ設定となっているため、その下に内部項目のオブジェクト用に1行追加してくだ

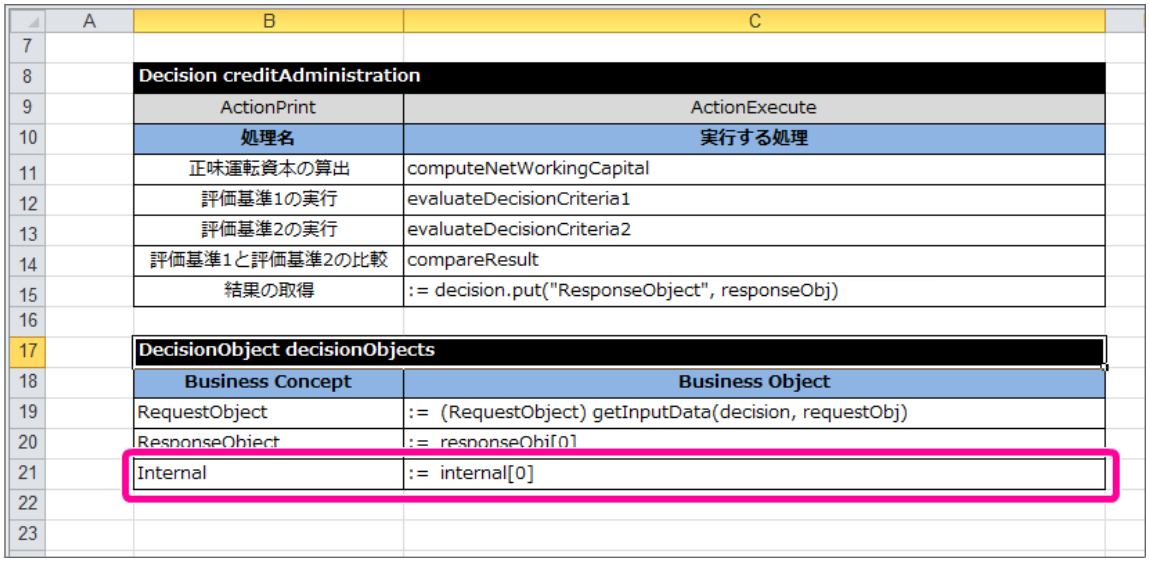
さい。



	A	B	C	D
7				
8		**Decision creditAdministration**		
9		ActionPrint	ActionExecute	
10		**処理名**	**実行する処理**	
11		正味運転資本の算出	computeNetWorkingCapital	
12		評価基準1の実行	evaluateDecisionCriteria1	
13		評価基準2の実行	evaluateDecisionCriteria2	
14		評価基準1と評価基準2の比較	compareResult	
15		結果の取得	:= decision.put("ResponseObject", responseObj)	
16				
17		**DecisionObject decisionObjects**		
18		**Business Concept**	**Business Object**	
19		RequestObject	:= (RequestObject) getInputData(decision, requestObj)	
20		ResponseObject	:= responseObj[0]	
21				
22				
23				

3. 追加した行には、以下の通りに設定してください。

DecisionObject decisionObjects	
Business Concept	Business Object
Internal	:= internal[0]



	A	B	C
7			
8		**Decision creditAdministration**	
9		ActionPrint	ActionExecute
10		**処理名**	**実行する処理**
11		正味運転資本の算出	computeNetWorkingCapital
12		評価基準1の実行	evaluateDecisionCriteria1
13		評価基準2の実行	evaluateDecisionCriteria2
14		評価基準1と評価基準2の比較	compareResult
15		結果の取得	:= decision.put("ResponseObject", responseObj)
16			
17		**DecisionObject decisionObjects**	
18		**Business Concept**	**Business Object**
19		RequestObject	:= (RequestObject) getInputData(decision, requestObj)
20		ResponseObject	:= responseObj[0]
21		Internal	:= internal[0]
22			
23			

4. これで、 OpenRules で取引先の与信管理をするためにExcelのルール定義ファイルの設定が完了しましたので、ファイルを保存してください。
IM-BIS の画面（フォーム）と連携するための設定を行っていきましょう。

IM-BIS と連携したフローを作成する

先の手順で作成したExcelのルール定義ファイルを IM-BIS と連携するためのフローの作成を進めていきましょう。

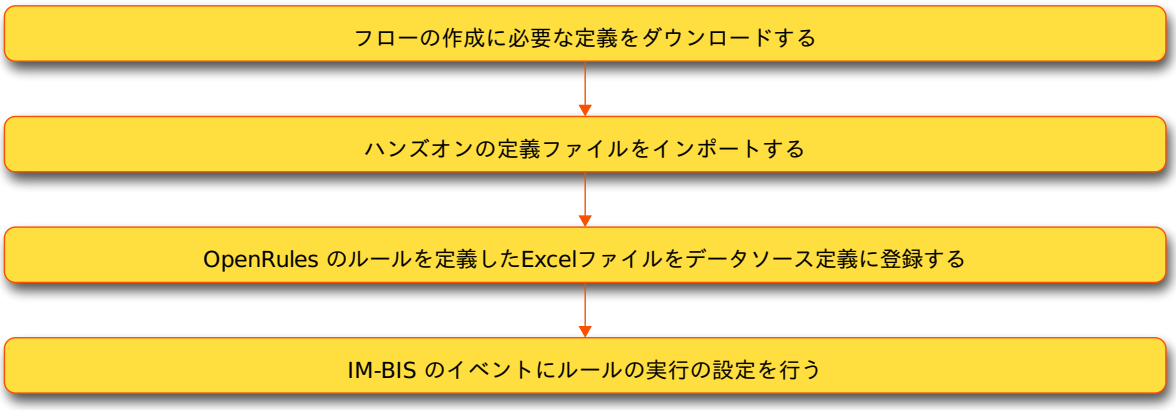
ルールと連携したフローを作成する手順

- OpenRules と IM-BIS を連携するための手順
- フローの作成に必要な定義をダウンロードする
- ハンズオンの定義ファイルをインポートする
- OpenRules のルールを定義したExcelファイルをデータソース定義に登録する
- IM-BIS のイベントにルールの実行の設定を行う

OpenRules と IM-BIS を連携するための手順

この手順では、作成したExcelのルール定義ファイルをデータソース定義に登録し、 IM-BIS の画面アイテムのイベントに設定するまでの手順を確認していきます。

- 取引先与信管理のルールをBISと連携する手順



フローの作成に必要な定義をダウンロードする

ハンズオンで作成するフローのベースとなる各種定義ファイルをインポートします。
最初に下記のリンクからファイルをダウンロードしてください。
「 IM-Workflow 定義」のみダウンロード後に解凍してください。

- IM-Workflow 定義
  imw_credit_management.zip
- BIS定義
  bis_credit_management.zip
- Formaアプリケーション定義
  forma_credit_management.zip

ハンズオンの定義ファイルをインポートする

先の手順でダウンロードしたファイルを「 *各種定義ファイルのインポートの手順* 」に従ってインポートしてください。

OpenRules のルールを定義したExcelファイルをデータソース定義に登録する

IM-BIS と連携するために、 OpenRules のルールを定義したExcelファイルをデータソース定義に登録していきましょう。

データソース定義の基本情報を登録する

データソース定義の基本情報を登録しましょう。

1. サイトマップの「 IM-BIS 」から「データソース定義」をクリックしてください。

2. 「登録」をクリックしてください。

3. 以下の通りに入力後、「登録」をクリックしてください。

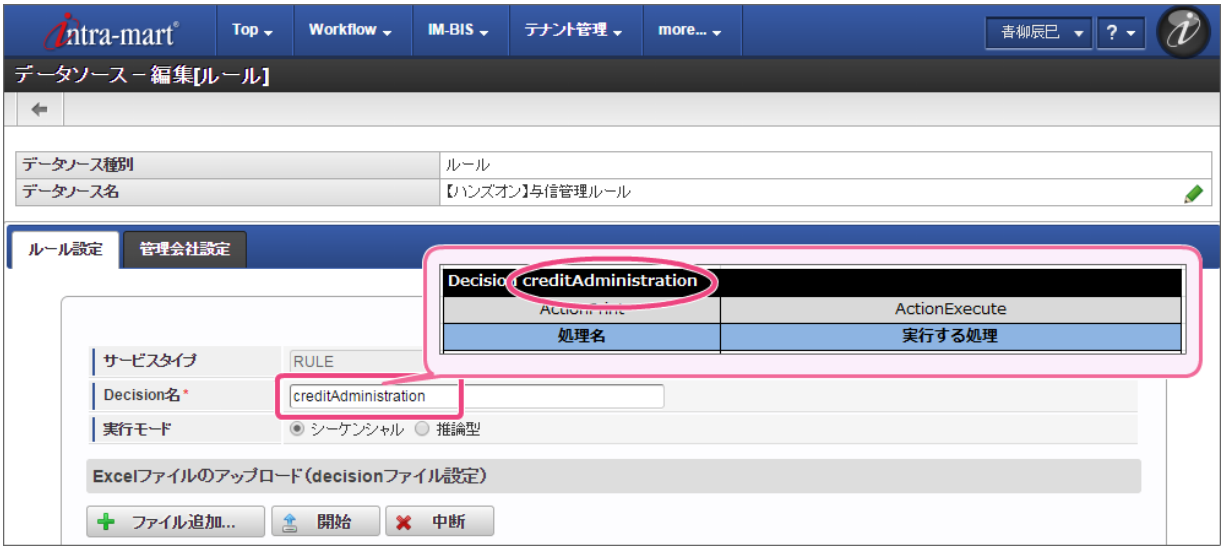
1. データソース種別
   「ルール」を選択してください。

2. データソース名
   「【ハンズオン】与信管理ルール」と入力してください。

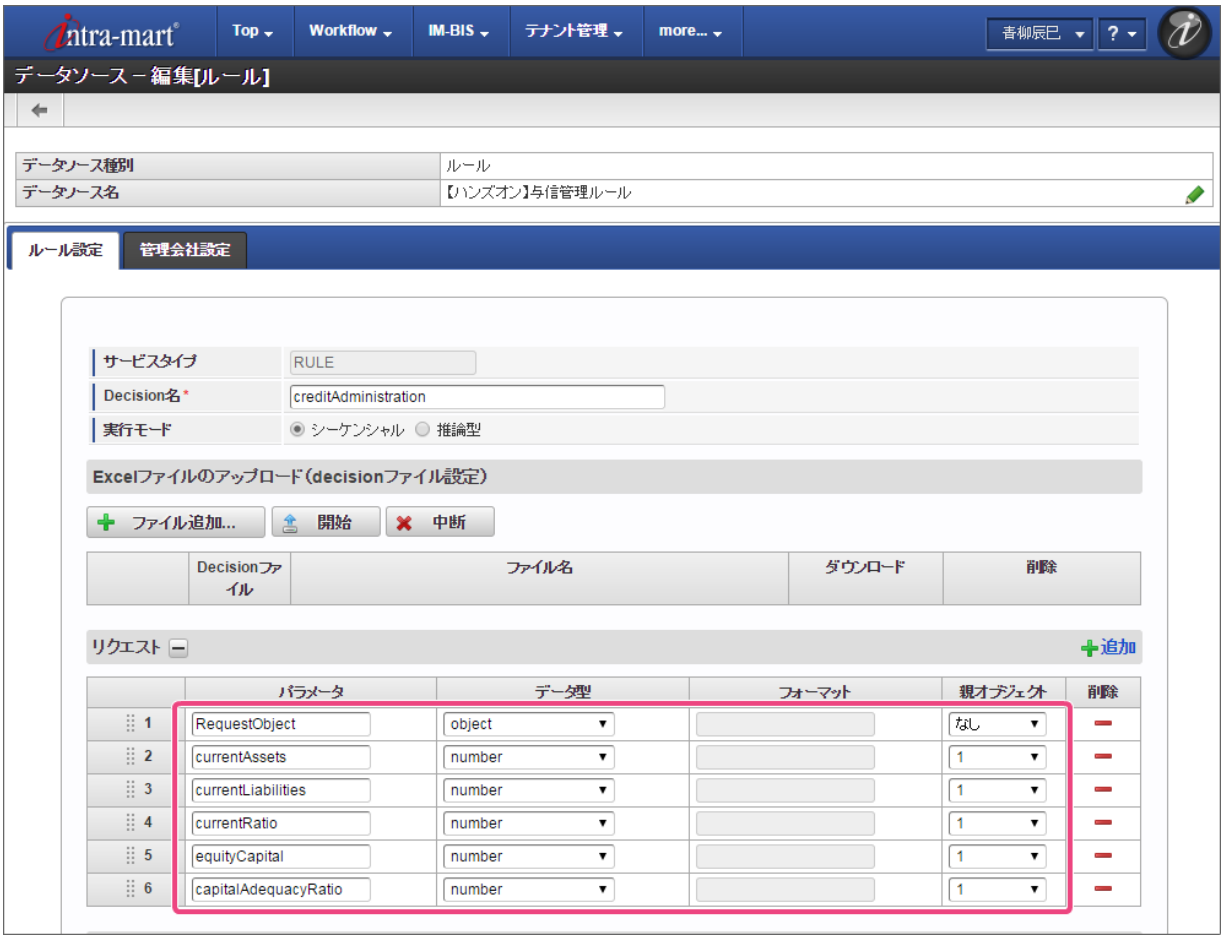
OpenRules の詳細情報を登録する

データソース定義に OpenRules のファイルやパラメータを設定しましょう。

1. 「Decision名」には、Excelファイルの *Decision* の名前「creditAdministration」を入力してください。



2. 「リクエスト」には、 *Glossary* で定義した「RequestObject」のオブジェクトと項目（物理名）を登録します。
入力欄を追加し、以下の通りに設定してください。



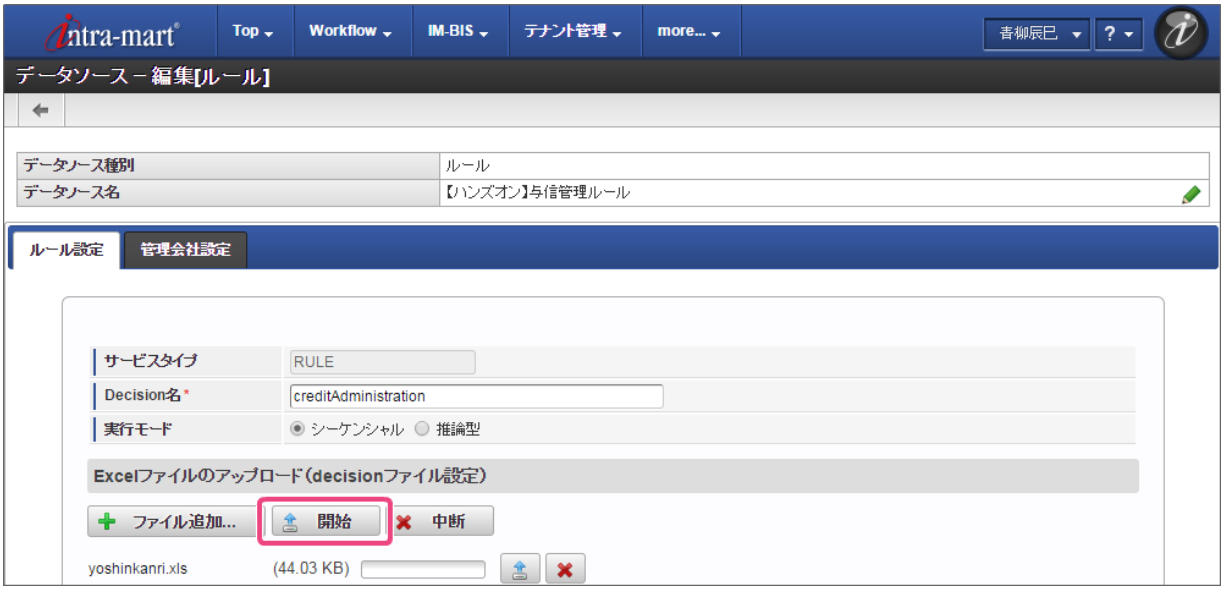
パラメータ	データ型	親オブジェクト
RequestObject	object	なし
currentAssets	number	1
currentLiabilities	number	1
currentRatio	number	1
equityCapital	number	1
capitalAdequacyRatio	number	1

3. 同様に「レスポンス」には、「ResponseObject」のオブジェクトと項目（物理名）を登録します。
入力欄を追加し、以下の通りに設定してください。

フィールド	データ型	親オブジェクト
ResponseObject	object	なし
creditworthiness	string	1
credit	number	1

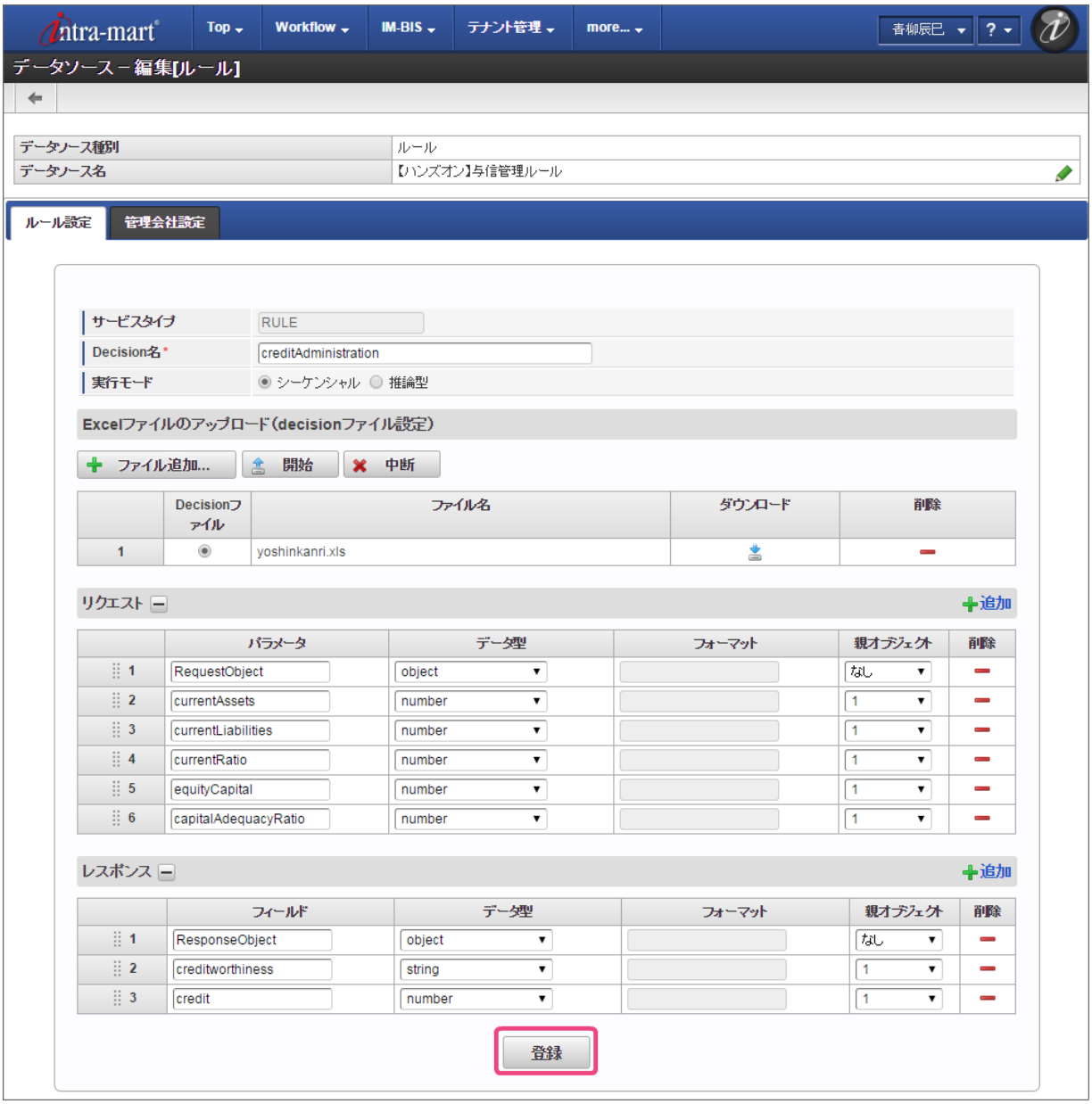
4. 作成したExcelのルール定義ファイルをアップロードするために「ファイル追加」をクリックしてください。

5. 「開始」をクリックして、ファイルをアップロードしてください。



6. 「Decisionファイル」のラジオボタンをクリックしてオンにしてください。

7. 最後に「登録」をクリックして、データソース定義を登録してください。



8. これで OpenRules のルールを定義したExcelファイルをデータソース定義として登録することができました。

IM-BIS のイベントにルールの実行の設定を行う

更新したデータソース定義を利用して、 IM-BIS の画面にルールの実行のアクションを設定しましょう。

フォーム（画面）の編集を開始する

画面の設定を開始するために、フォーム（画面）の編集画面を表示しましょう。

1. サイトマップの「 IM-BIS 」をクリックしてください。

2. 「一覧」をクリックしてください。

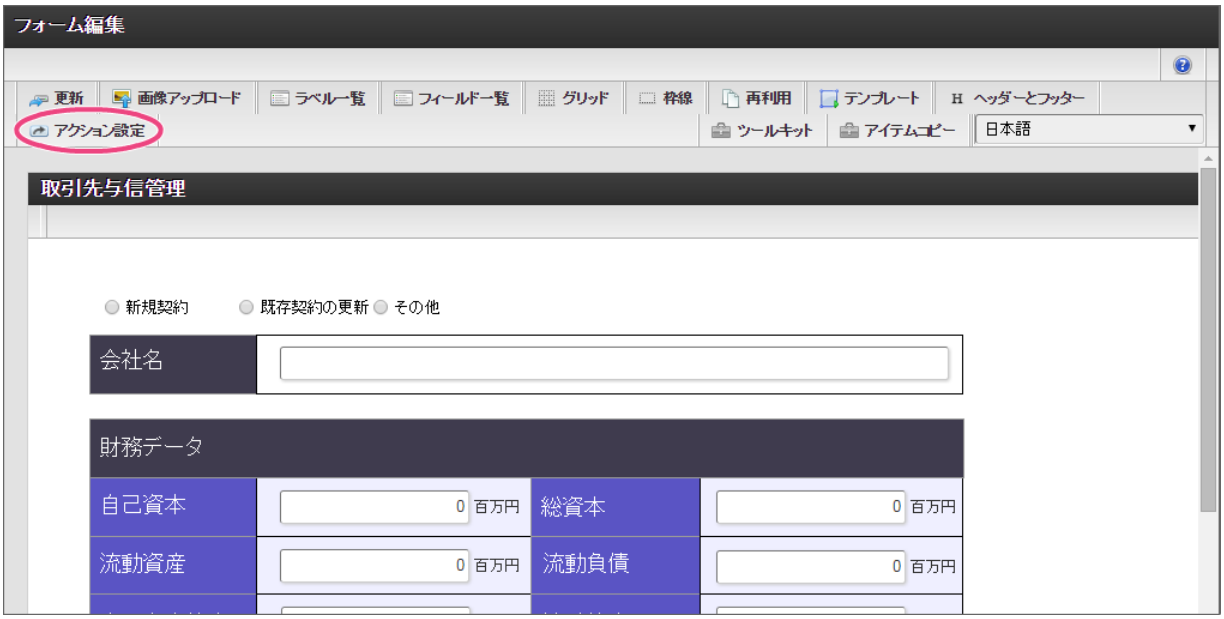
3. インポートしたフローの「【ハンズオン】取引先与信管理」の をクリックしてください。

4. 「申請／処理開始」をダブルクリックして、フォーム編集画面（フォーム・デザイナ）を表示してください。

画面のアクションイベントにルールの実行を設定する

フォーム（画面）の編集画面で、画面アイテムにルールを実行するイベントを設定しましょう。

1. フォーム編集画面を表示したら「アクション設定」をクリックしてください。



2. 「アイテムイベント」をクリックして、表示するタブを切り替えます。

3. 「 追加」をクリックしてください。

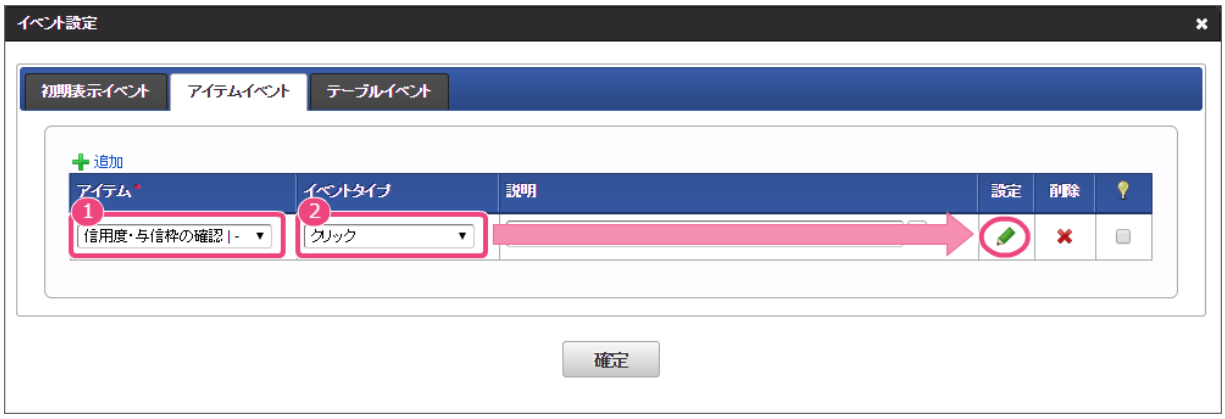
4. アイテムとイベントタイプを以下の通りに変更後、「 」をクリックしてください。

    1. アイテム

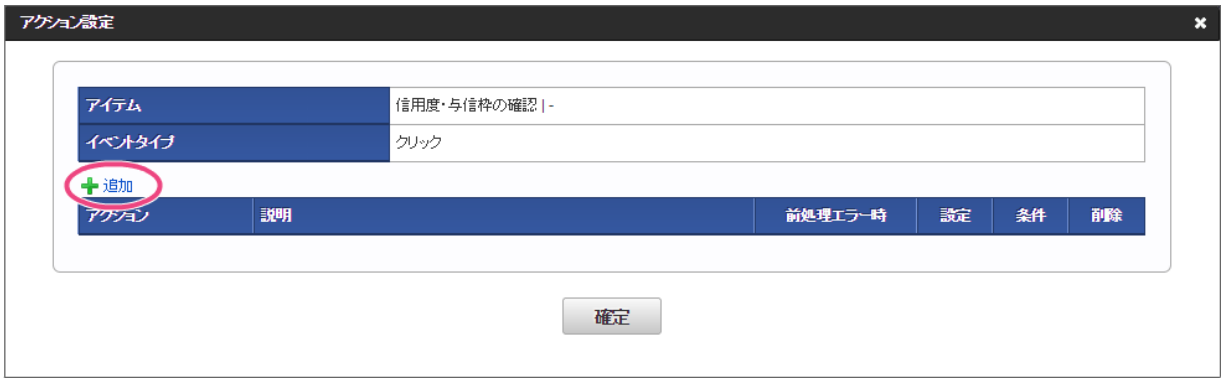
        信用度・与信枠の確認 | - （ボタン（イベント））

    2. イベントタイプ

        クリック

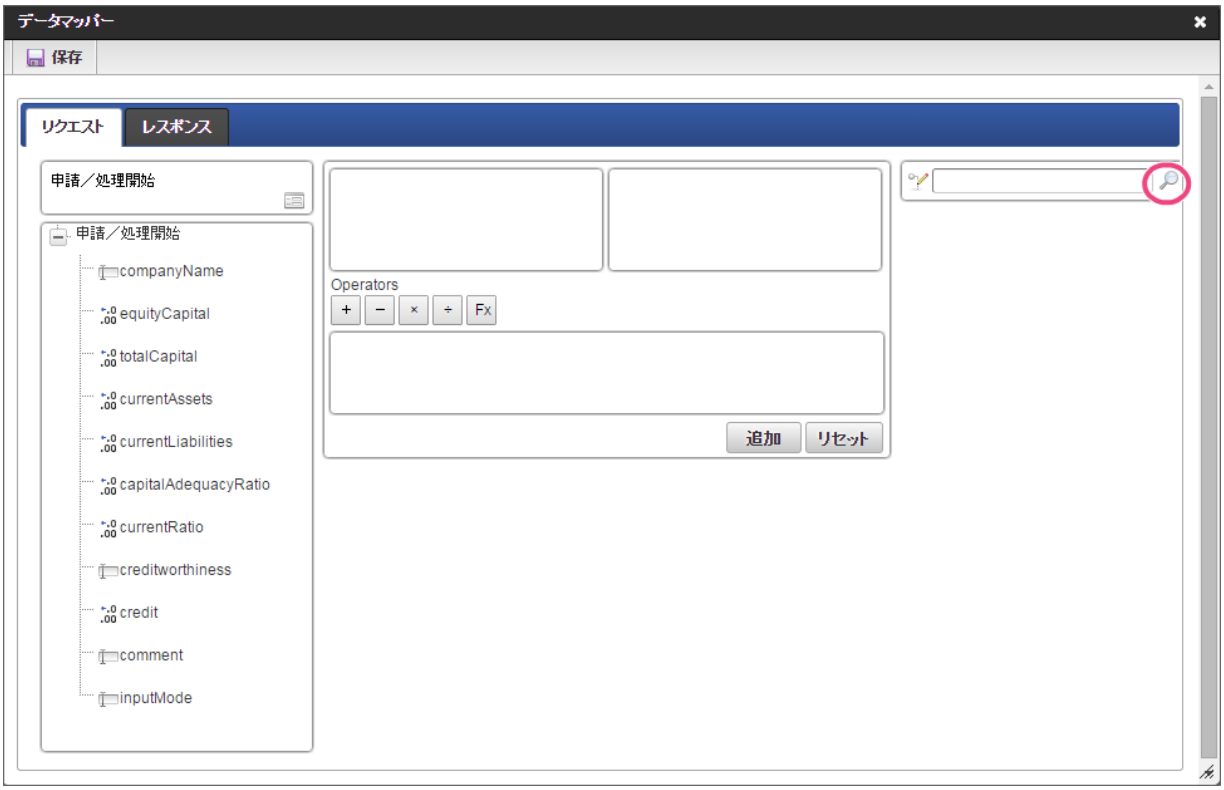


5. 「追加」をクリックしてください。



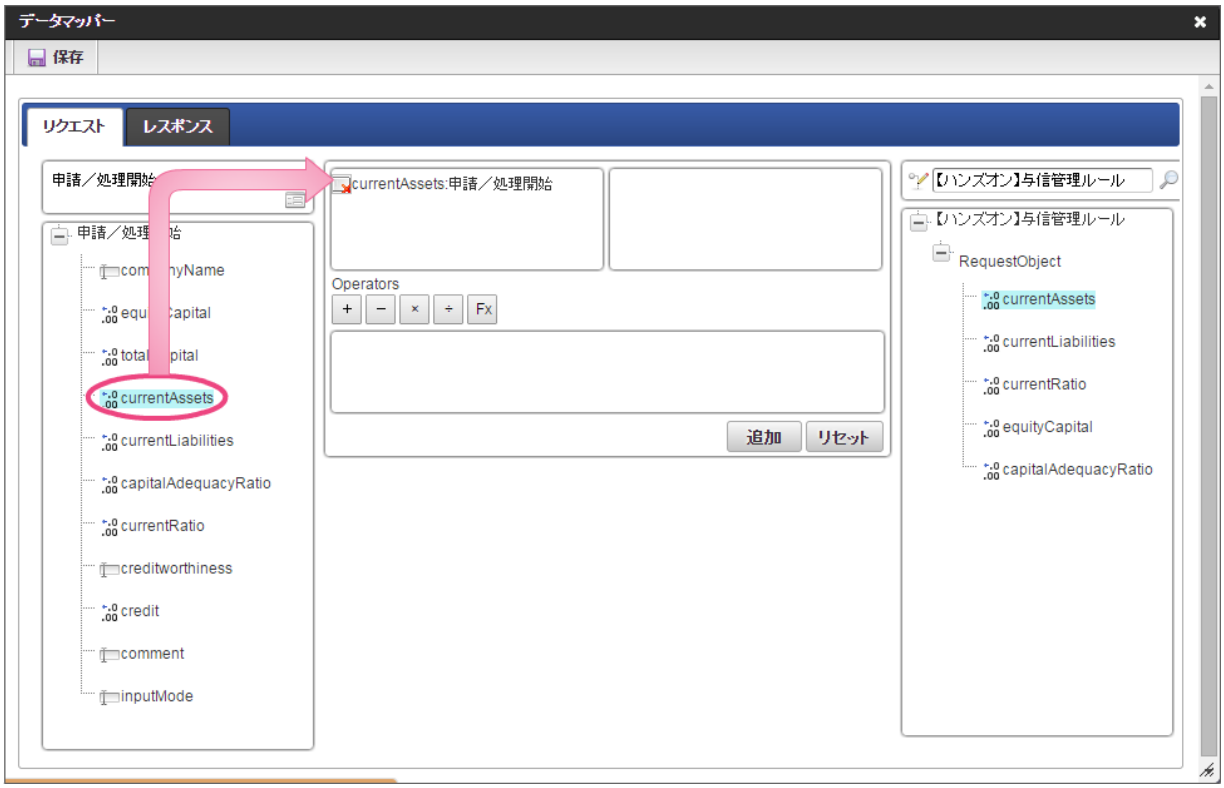
6. 「アクション」を「外部連携」に設定後、「 」をクリックしてください。

7. 「データマッパー」で右上の をクリックしてください。

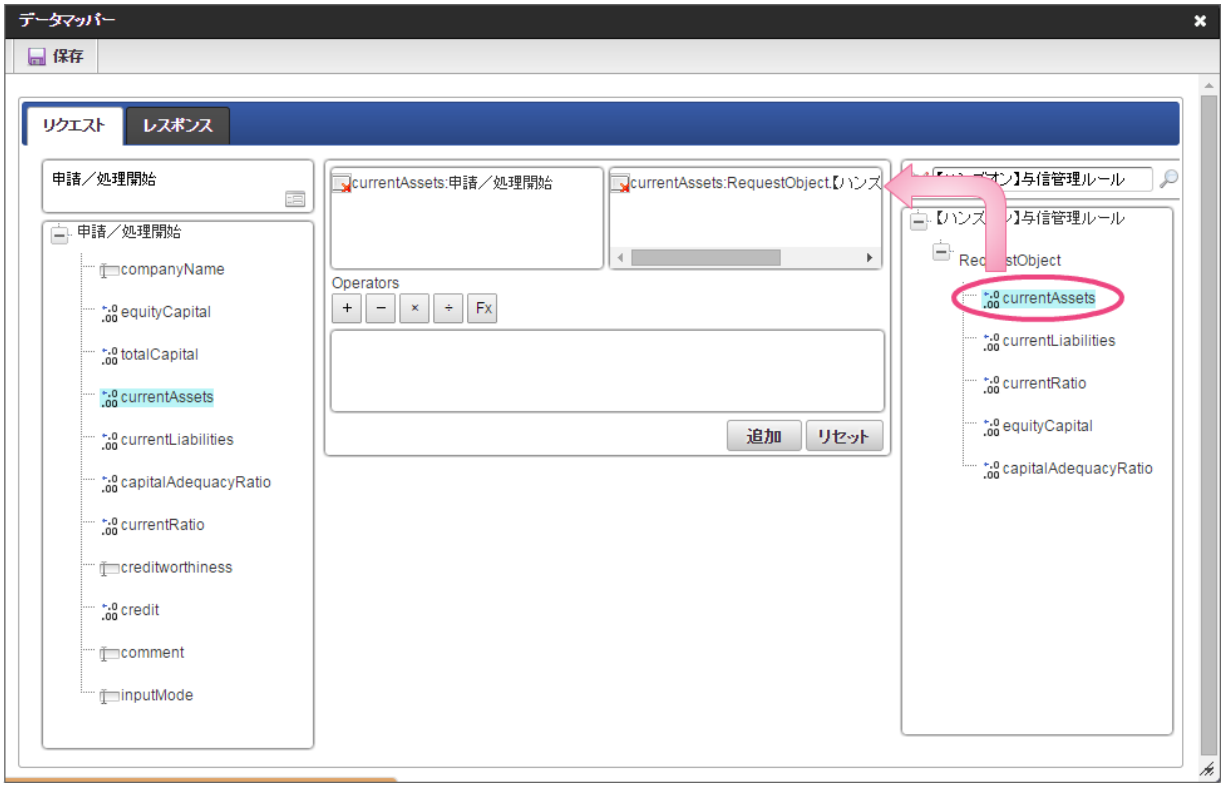


8. データソース選択から作成したルール「【ハンズオン】与信管理ルール」をクリックしてください。

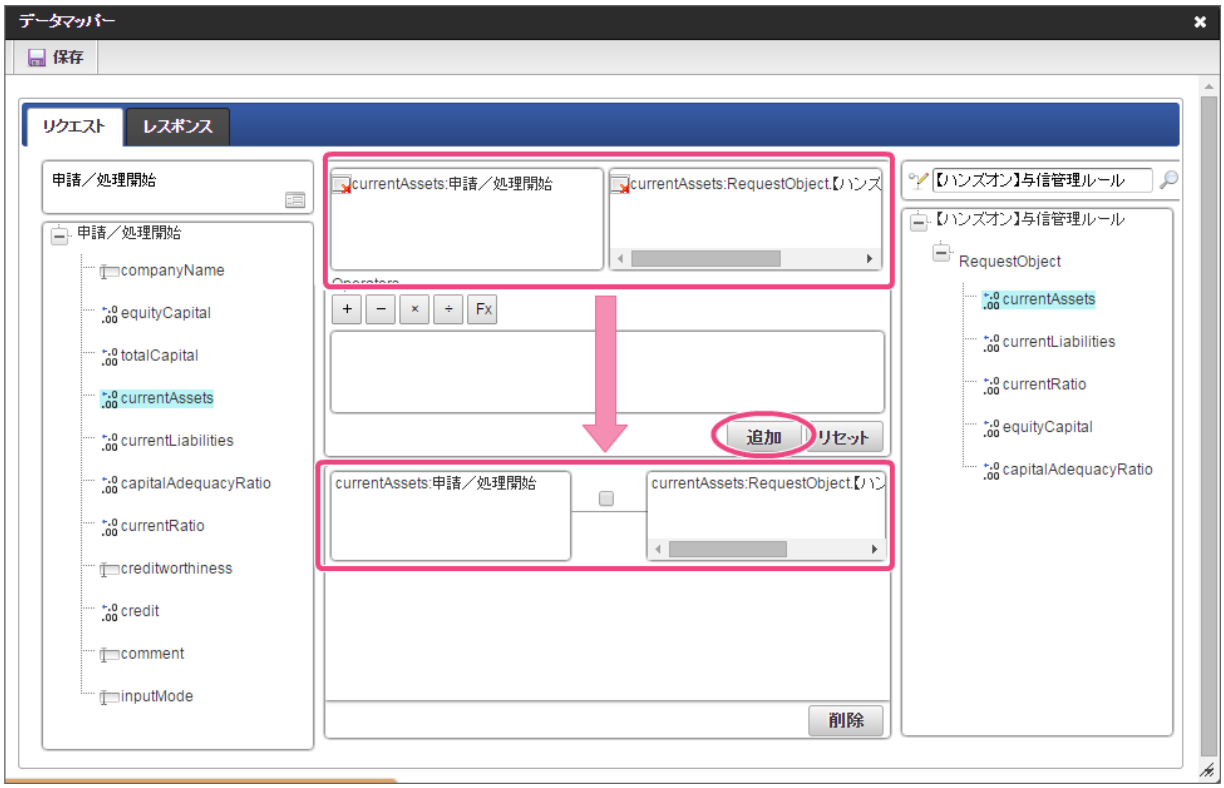
9. 左の欄から「currentAssets」をクリックしてください。
   クリック後、中央左の欄に「currentAssets: 申請／処理開始」と表示されます。



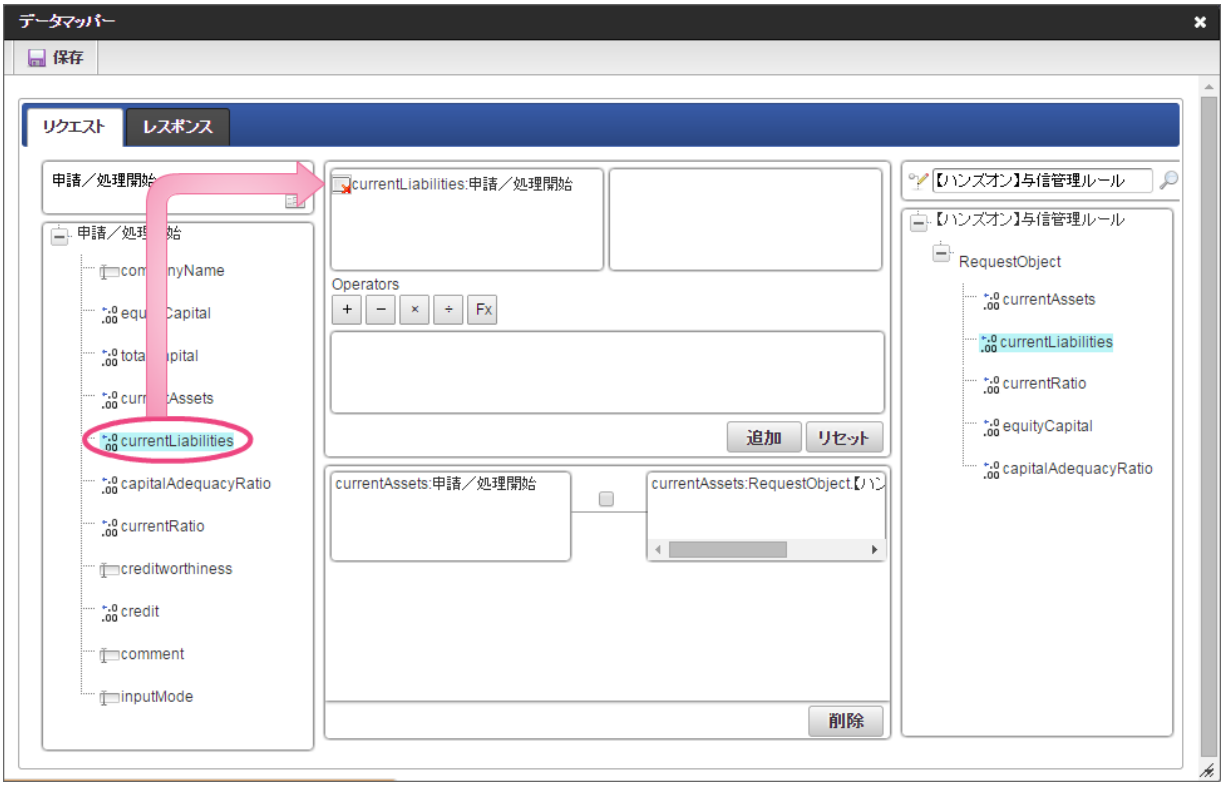
10. 右の欄から「currentAssets」をクリックしてください。
クリック後、中央右の欄に「currentAssets:RequestObject.【ハンズオン】与信管理ルール」と表示されます。



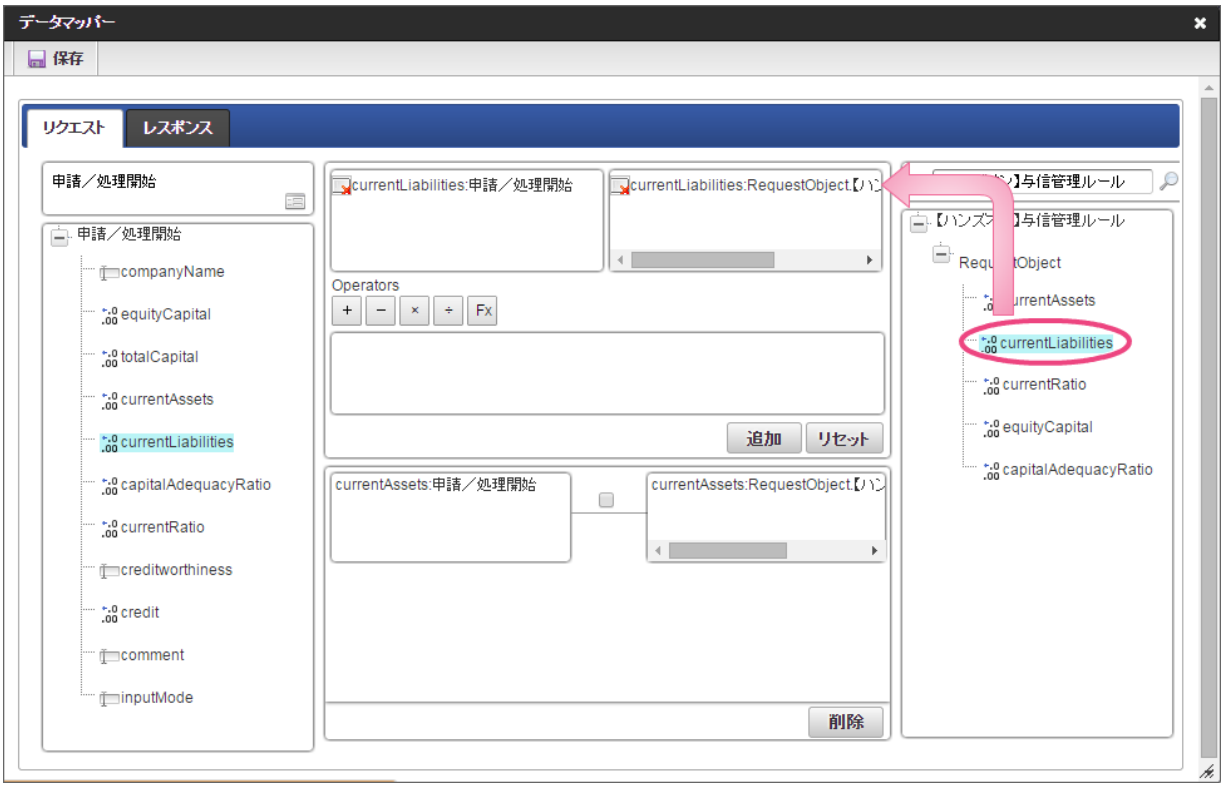
11. 「追加」をクリックしてください。
フォームの「currentAssets」とデータソース定義の「currentAssets」のマッピングが設定され、中央下段の欄に表示されます。



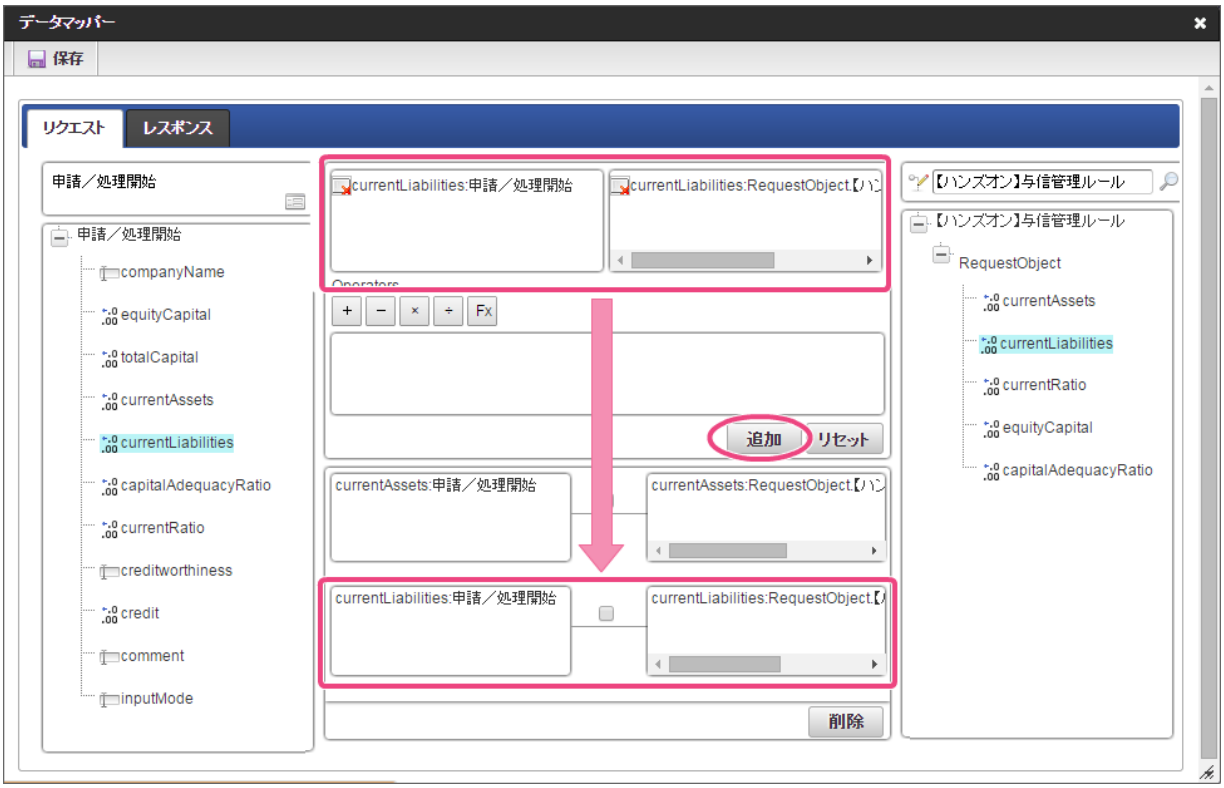
12. 左の欄から「currentLiabilities」をクリックしてください。
クリック後、中央左の欄に「currentLiabilities: 申請／処理開始」と表示されます。



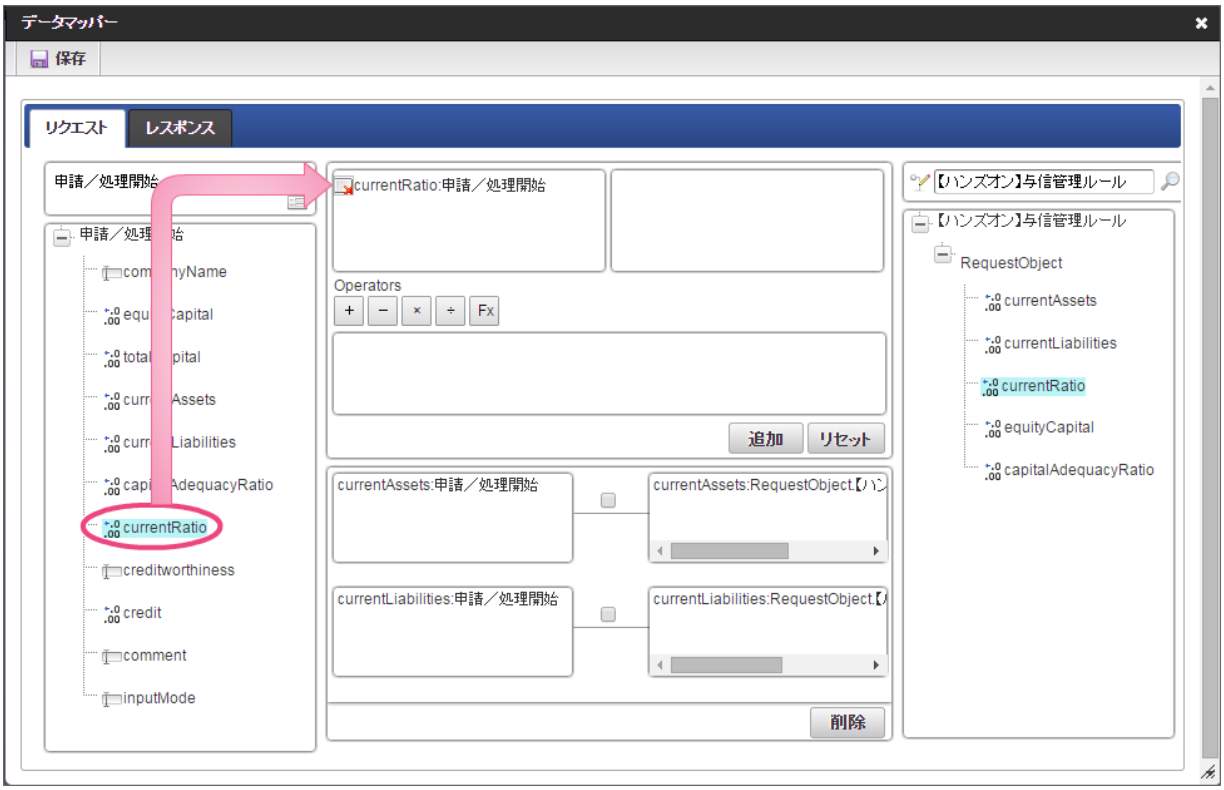
13. 右の欄から「currentLiabilities」をクリックしてください。
クリック後、中央右の欄に「currentLiabilities:RequestObject.【ハンズオン】与信管理ルール」と表示されます。



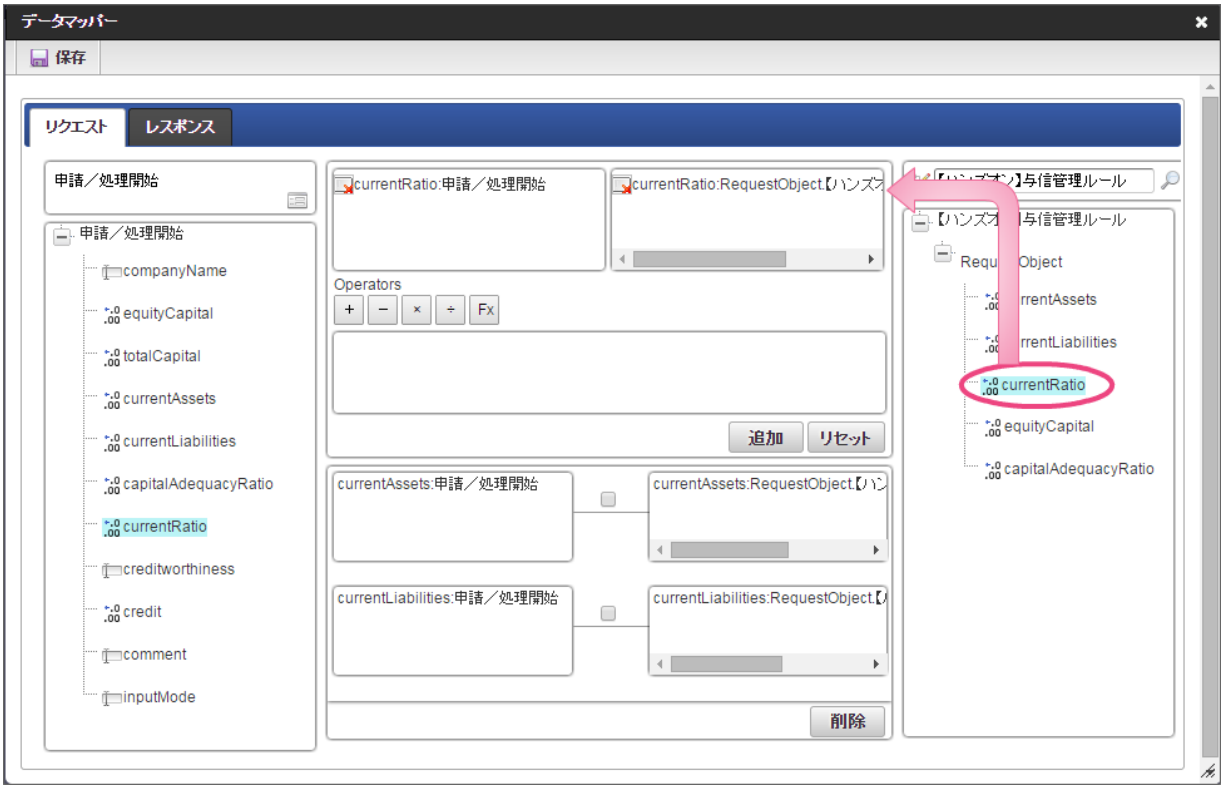
14. 「追加」をクリックしてください。
フォームの「currentLiabilities」とデータソース定義の「currentLiabilities」のマッピングが設定され、中央下段の欄に表示されます。



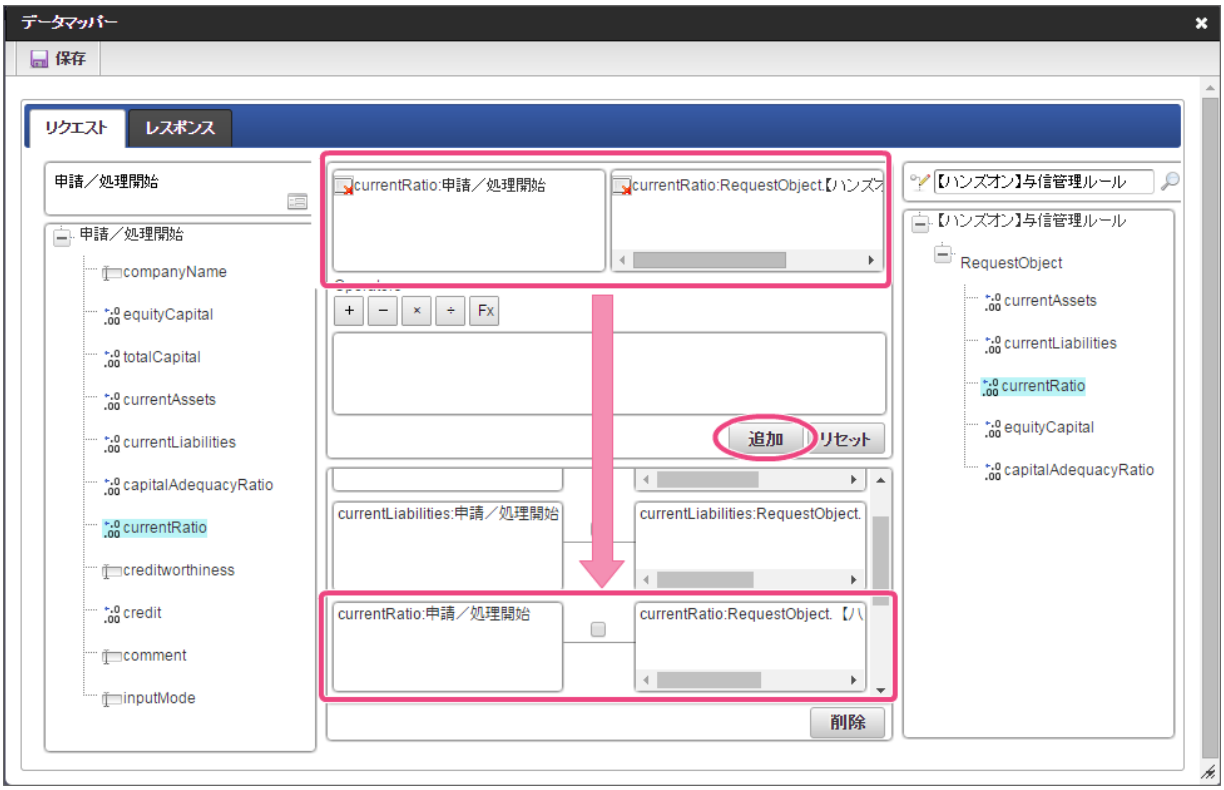
15. 左の欄から「currentRatio」をクリックしてください。
クリック後、中央左の欄に「currentRatio: 申請／処理開始」と表示されます。



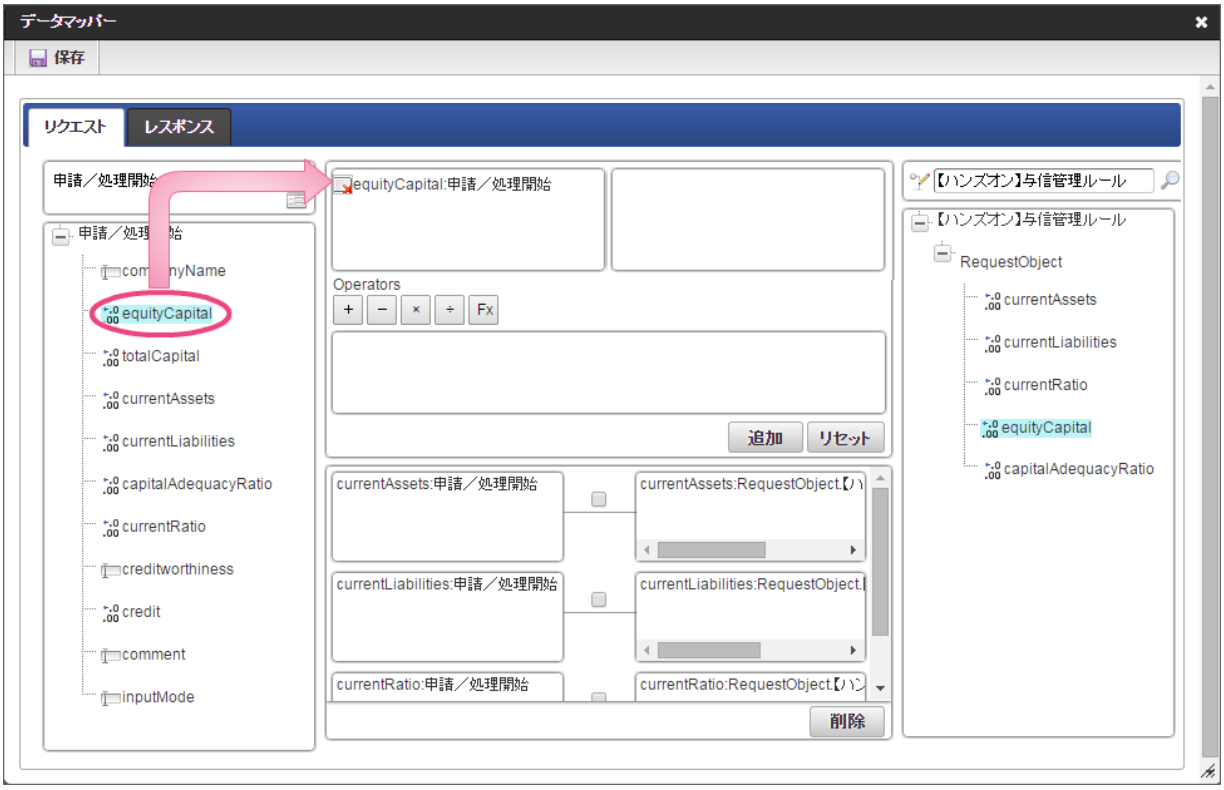
16. 右の欄から「currentRatio」をクリックしてください。
クリック後、中央右の欄に「currentRatio:RequestObject.【ハンズオン】与信管理ルール」と表示されます。



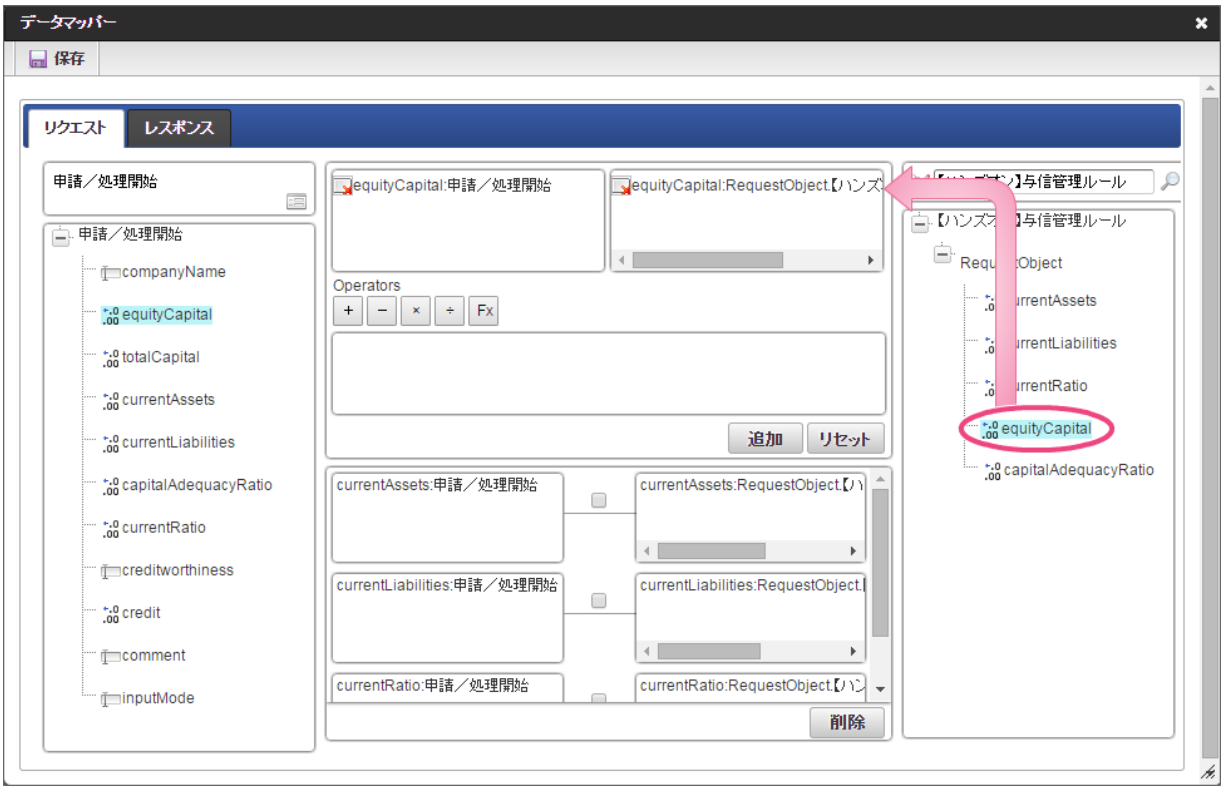
17. 「追加」をクリックしてください。
フォームの「currentRatio」とデータソース定義の「currentRatio」のマッピングが設定され、中央下段の欄に表示されます。



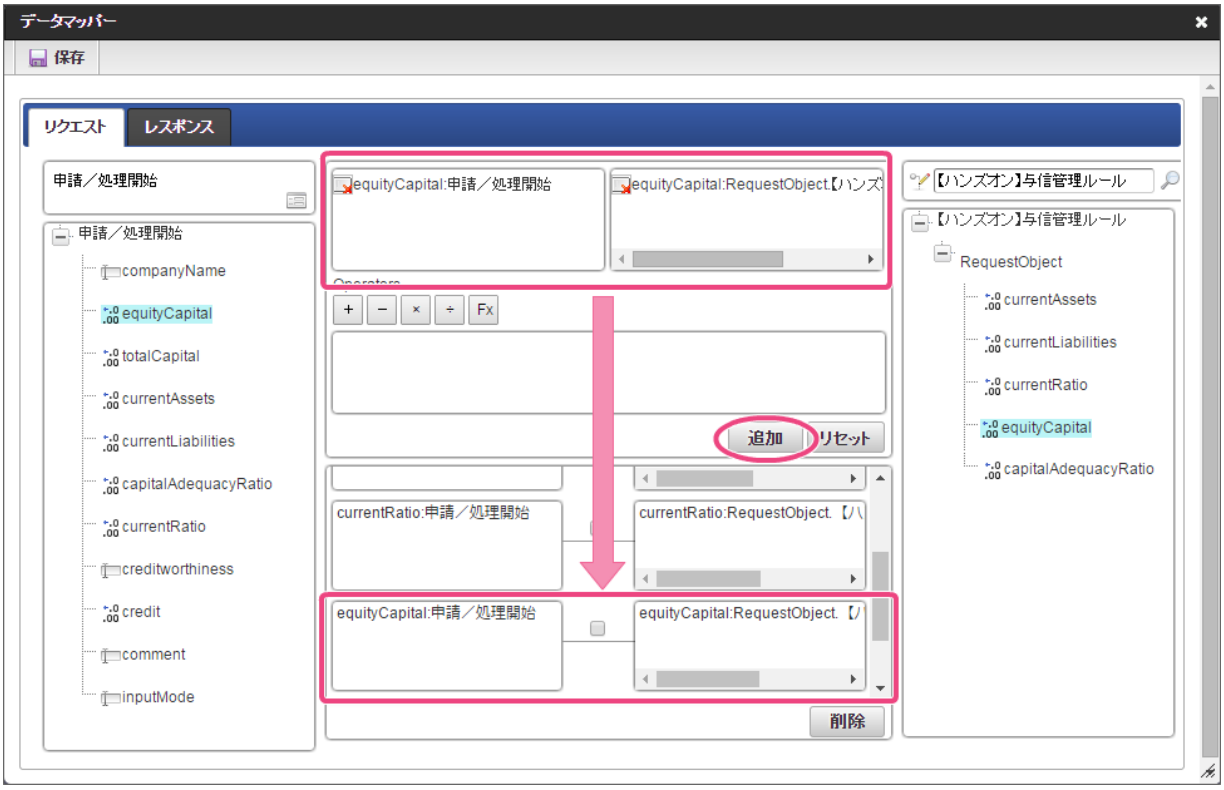
18. 左の欄から「equityCapital」をクリックしてください。
クリック後、中央左の欄に「equityCapital: 申請／処理開始」と表示されます。



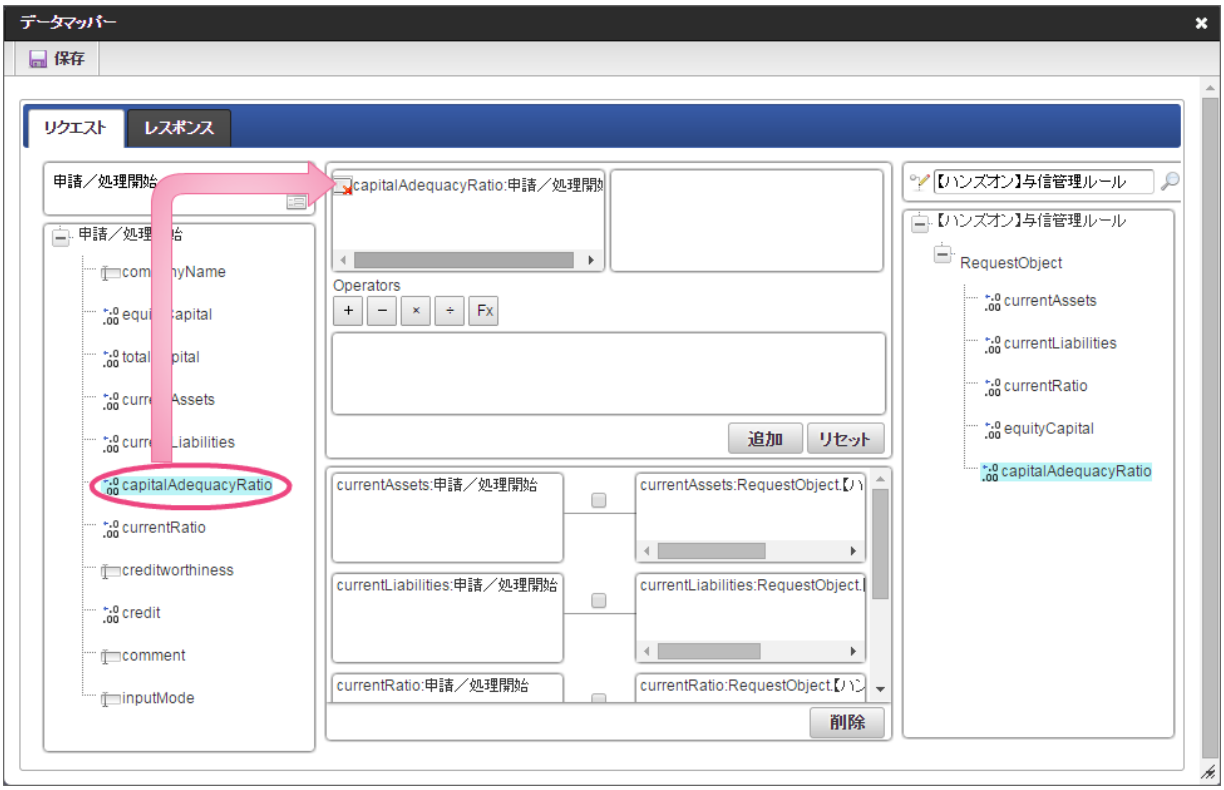
19. 右の欄から「equityCapital」をクリックしてください。
クリック後、中央右の欄に「equityCapital:RequestObject.【ハンズオン】与信管理ルール」と表示されます。



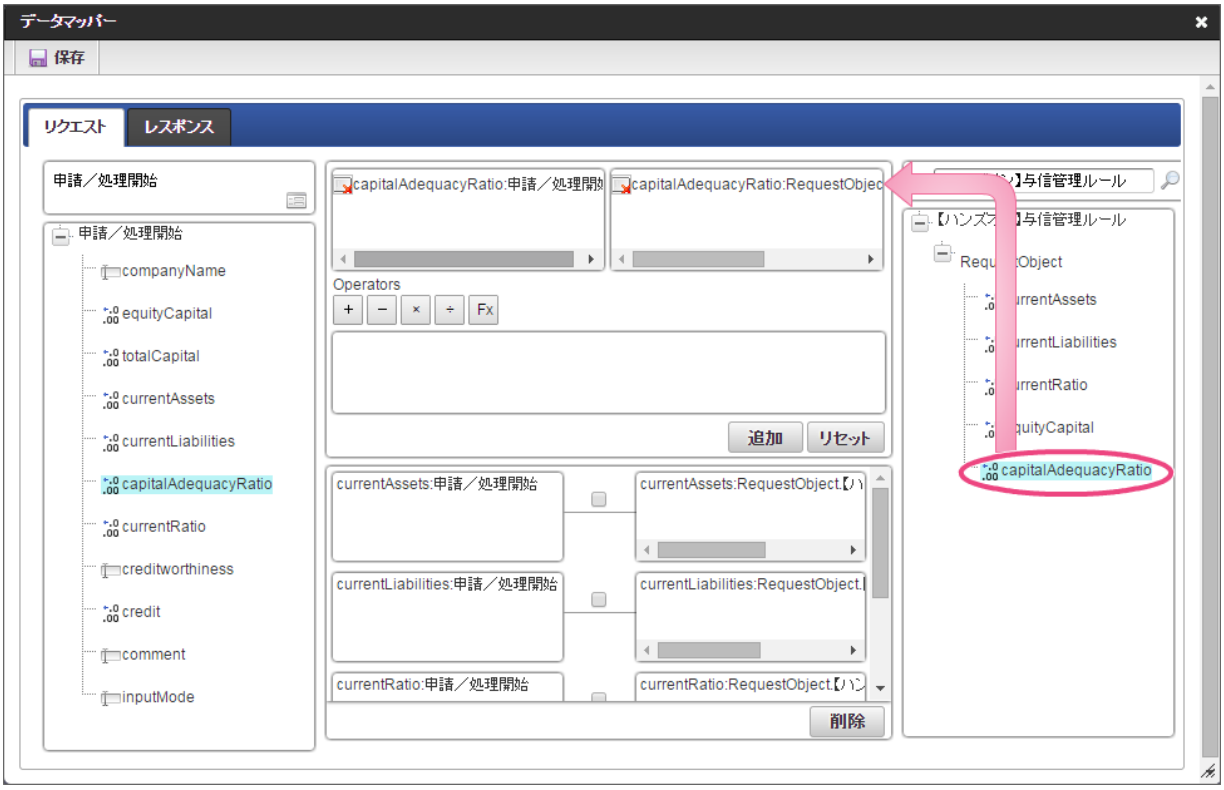
20. 「追加」をクリックしてください。
フォームの「equityCapital」とデータソース定義の「equityCapital」のマッピングが設定され、中央下段の欄に表示されます。



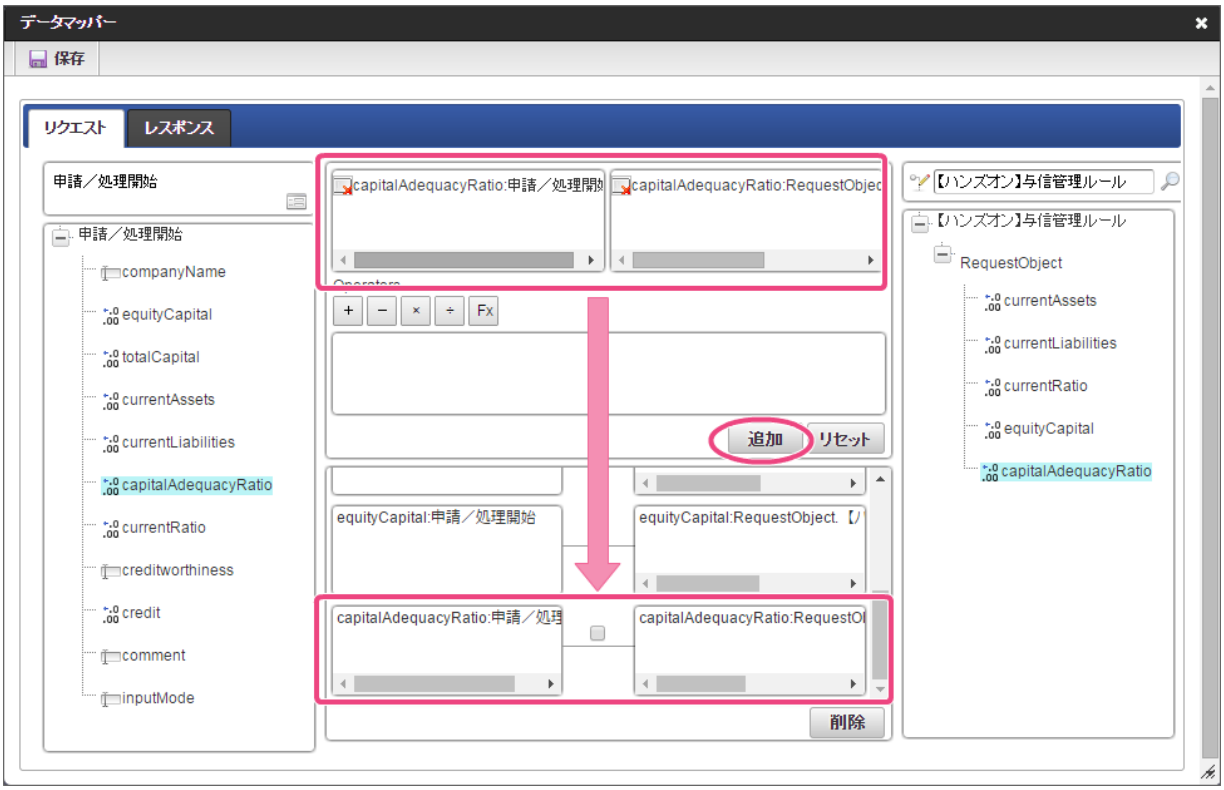
21. 左の欄から「capitalAdequacyRatio」をクリックしてください。
クリック後、中央左の欄に「capitalAdequacyRatio: 申請／処理開始」と表示されます。



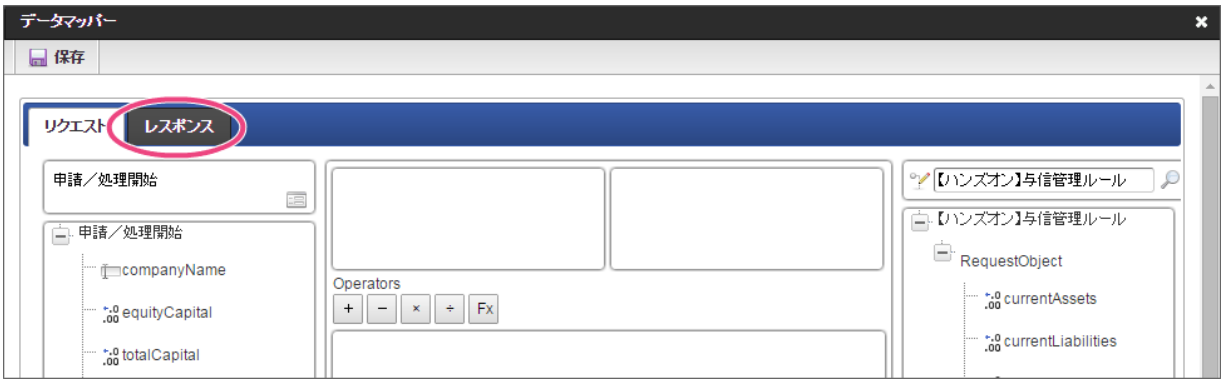
22. 右の欄から「capitalAdequacyRatio」をクリックしてください。
クリック後、中央右の欄に「capitalAdequacyRatio:RequestObject.【ハンズオン】与信管理ルール」と表示されます。



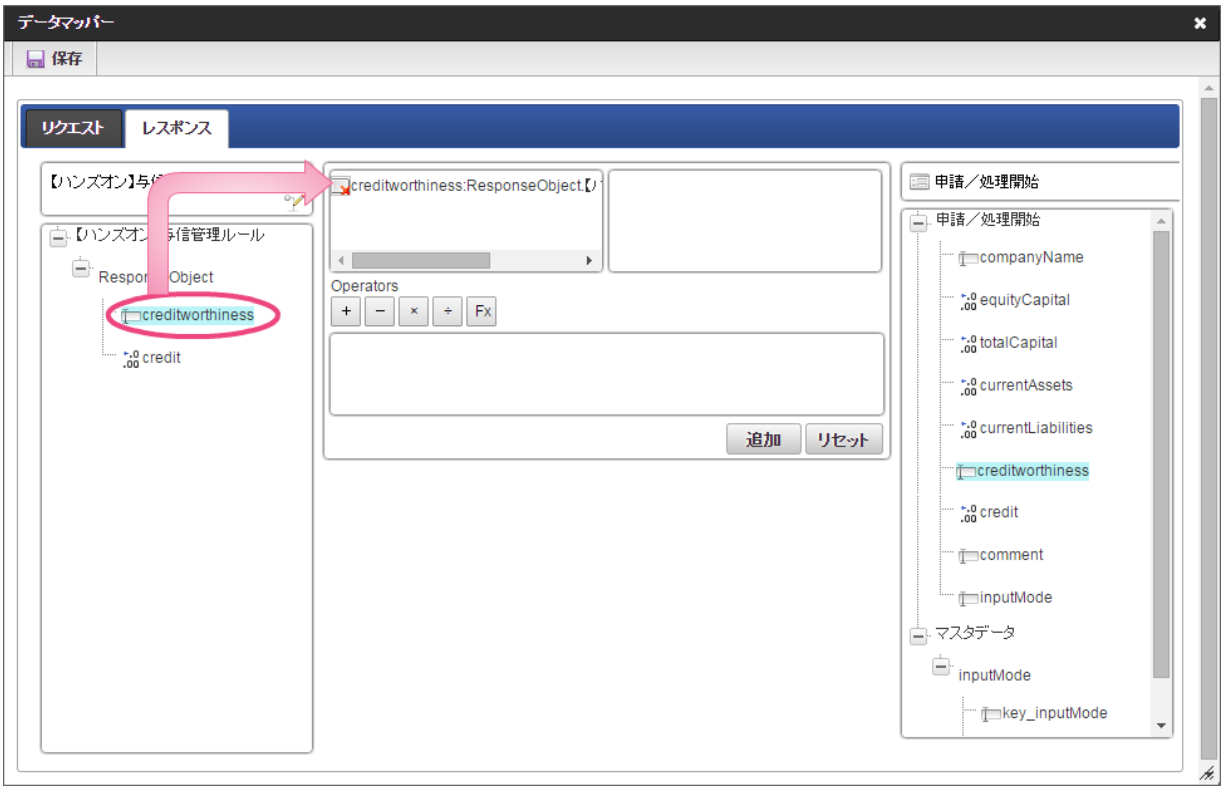
23. 「追加」をクリックしてください。
フォームの「capitalAdequacyRatio」とデータソース定義の「capitalAdequacyRatio」のマッピングが設定され、中央下段の欄に表示されます。



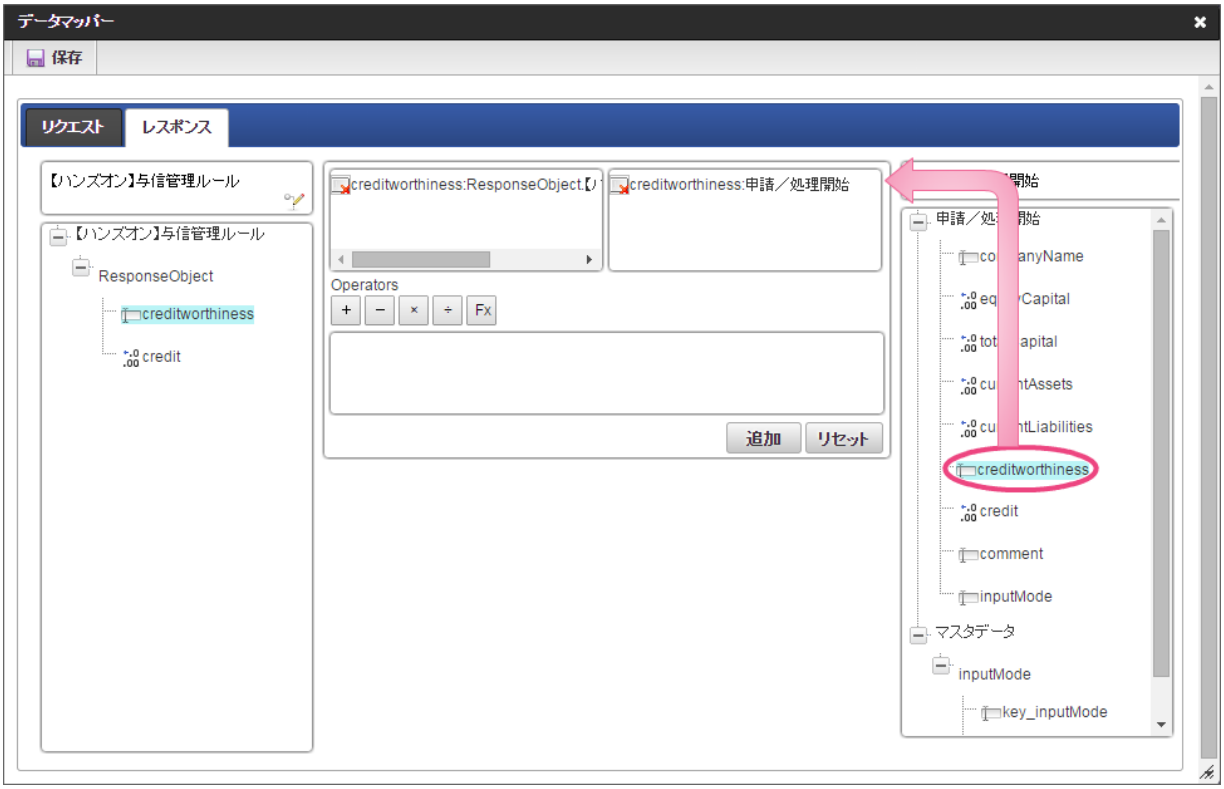
24. リクエストの設定が完了しましたので、レスポンスの設定を行うために「レスポンス」タブをクリックしてください。



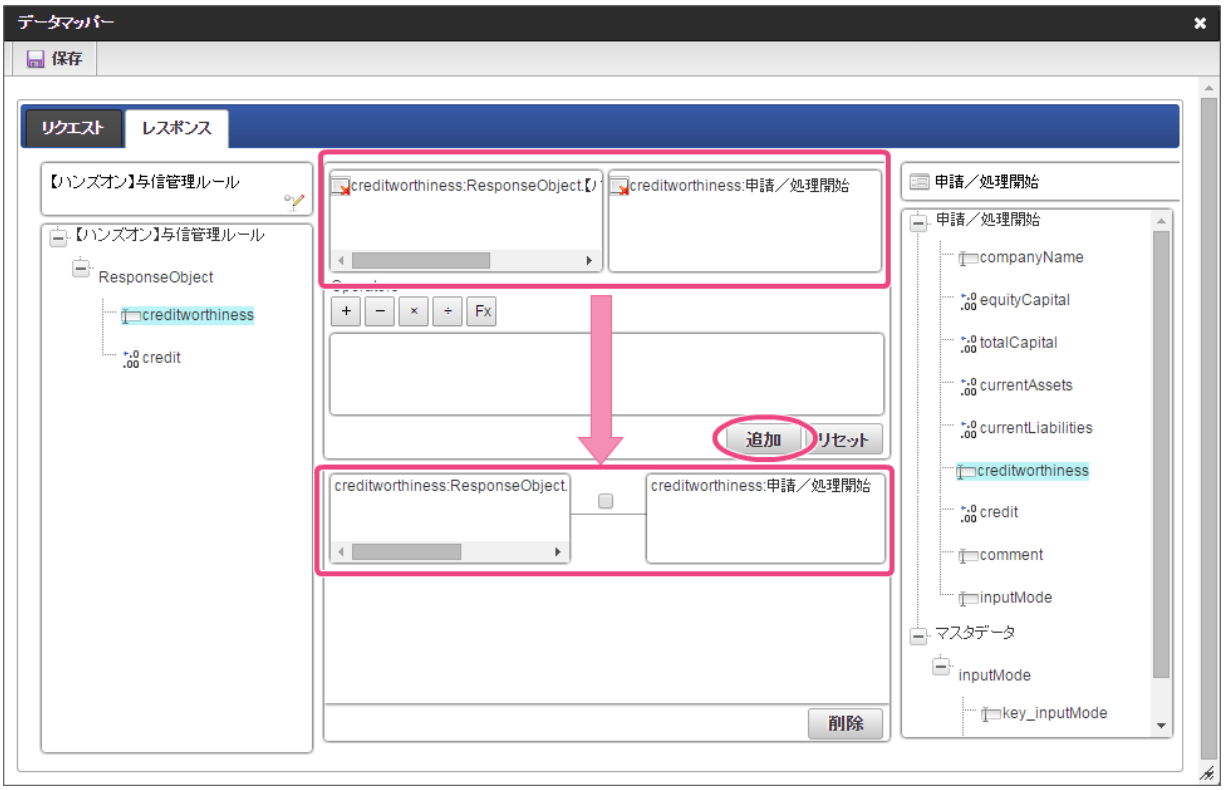
25. 左の欄から「creditworthiness」をクリックしてください。
クリック後、中央左の欄に「creditworthiness: ResponseObject.【ハンズオン】与信管理ルール」と表示されます。



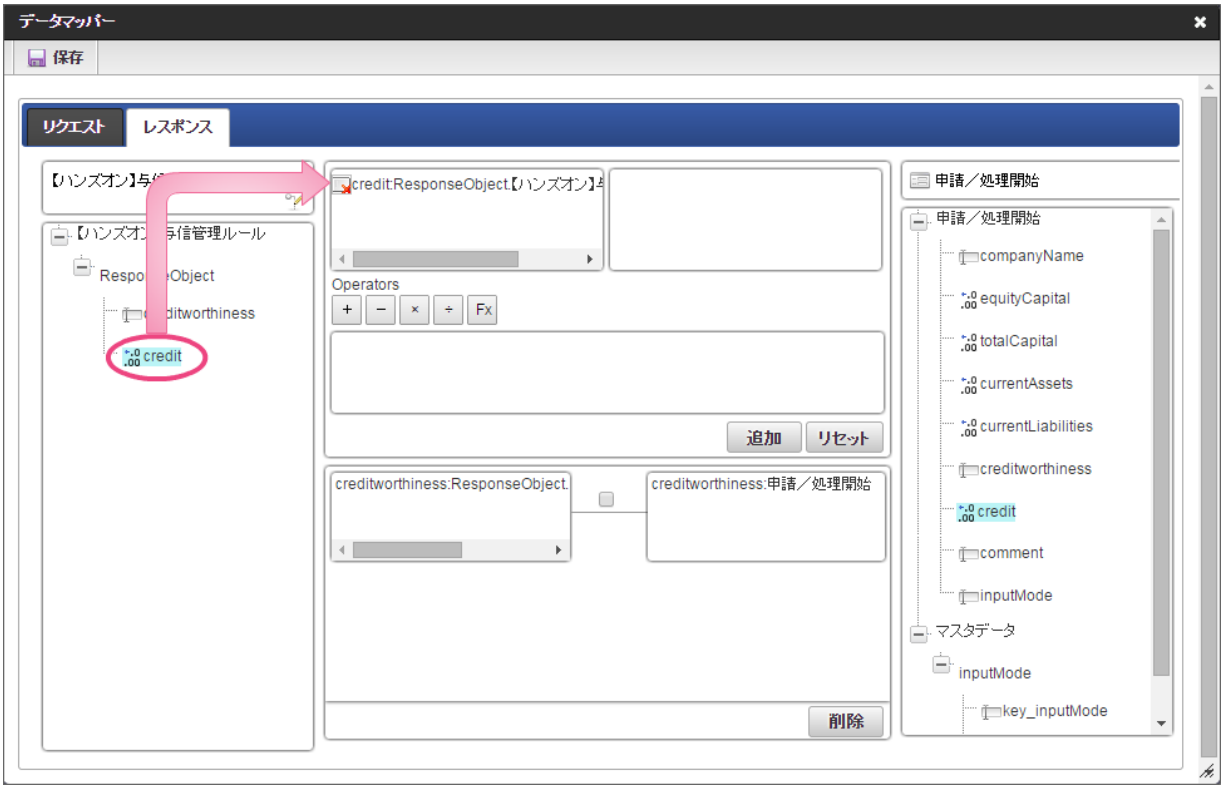
26. 右の欄から「creditworthiness」をクリックしてください。
クリック後、中央右の欄に「creditworthiness: 申請／処理開始」と表示されます。



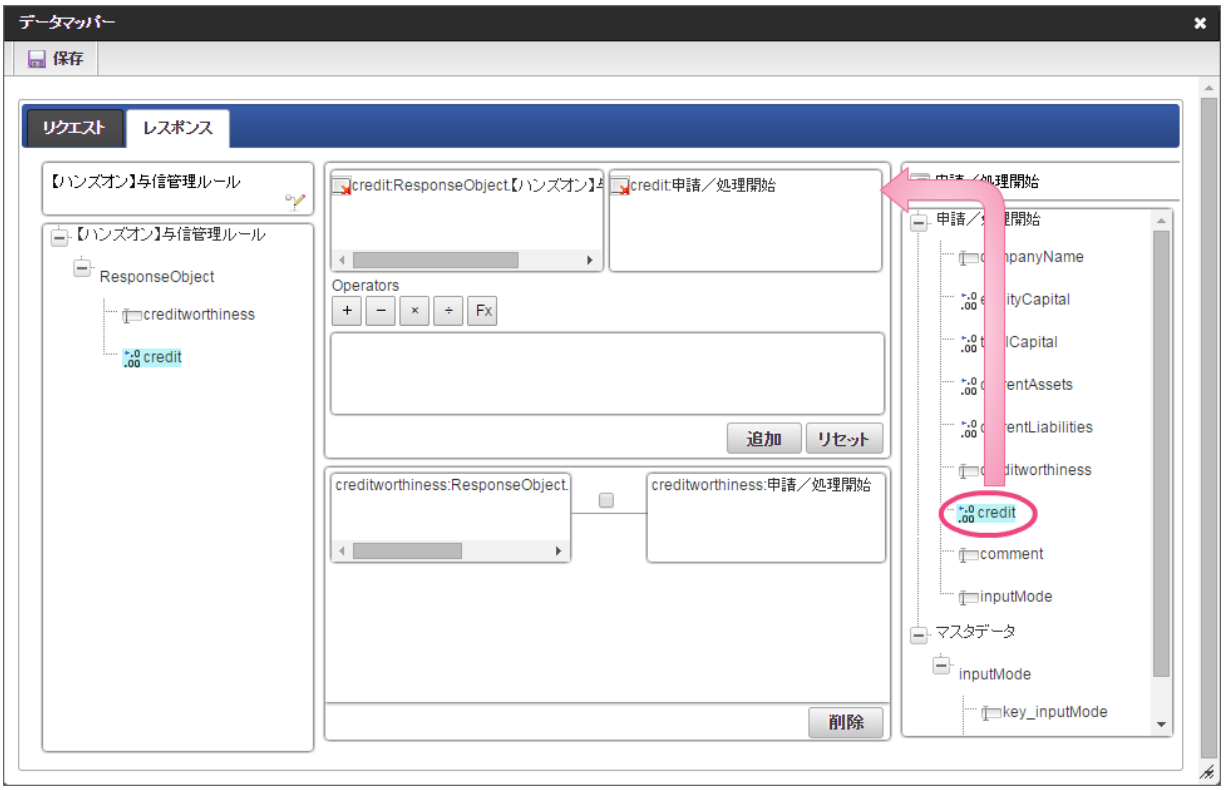
27. 「追加」をクリックしてください。
データソース定義の「creditworthiness」とフォームの「creditworthiness」のマッピングが設定され、中央下段の欄に表示されます。



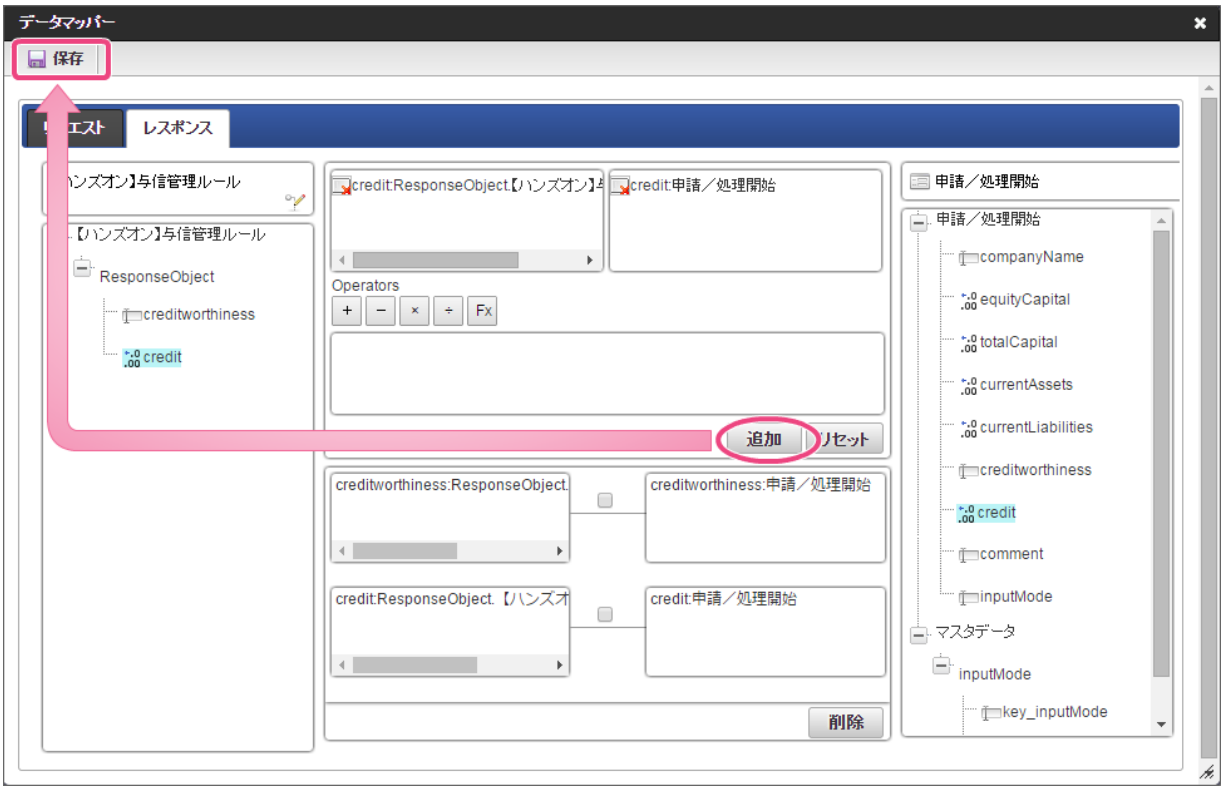
28. 左の欄から「credit」をクリックしてください。
クリック後、中央左の欄に「credit: ResponseObject.【ハンズオン】与信管理ルール」と表示されます。



29. 右の欄から「credit」をクリックしてください。
クリック後、中央右の欄に「credit: 申請／処理開始」と表示されます。



30. 「追加」、「保存」の順にクリックしてください。

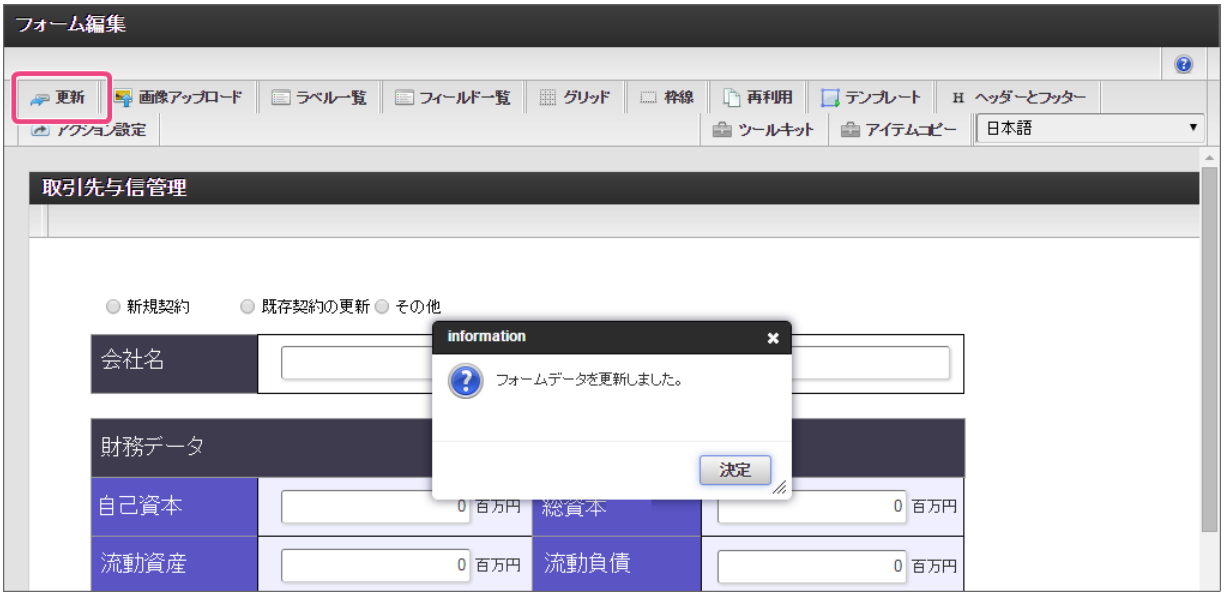


31. 正常に保存できたら、「データマッパー」は右上の「 ✖ 」をクリックして閉じてください。

32. アクション設定で「確定」をクリックしてください。

33. イベント設定で「確定」をクリックしてイベントの設定を保存してください。

34. 「更新」をクリックして、フォーム（画面）を保存してください。



35. 最後に「定義の反映」をクリックして、フローの実行準備を行ってください。

36. これで、必要な設定作業はすべて完了しましたので、実際にフローで申請・承認を行ってみましょう。

取引先の与信管理ワークフローを実行してみよう

これまでのシナリオで作成した IM-BIS のフローを使って、取引先の信用度・与信枠の評価を実行してみましょう。

ルールと連携したフローを実行する手順

- 取引先の与信管理の申請画面でルールを実行する
- 取引先の与信管理の承認を実行する

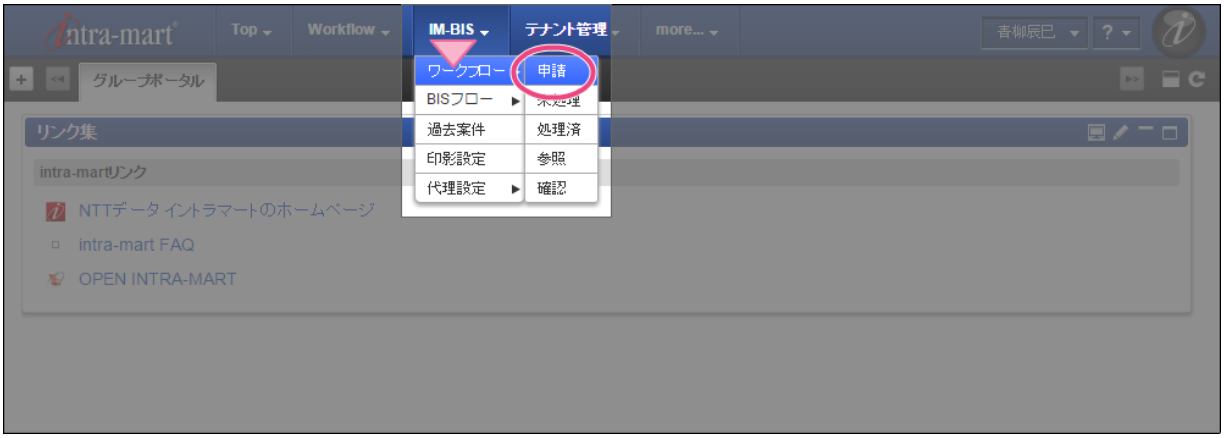
取引先の与信管理の申請画面でルールを実行する

作成したワークフローの申請画面でルールを実行してみましょう。

申請画面を表示する

1. 「BIS担当者」ロールを付与したユーザでログインしましょう。
（このマニュアルでは、「青柳辰巳」（ユーザコード：aoyagi）でログインします。）

2. 上部のメニューの「 IM-BIS 」→「ワークフロー」の順にマウスを重ねてから「申請」をクリックしましょう。



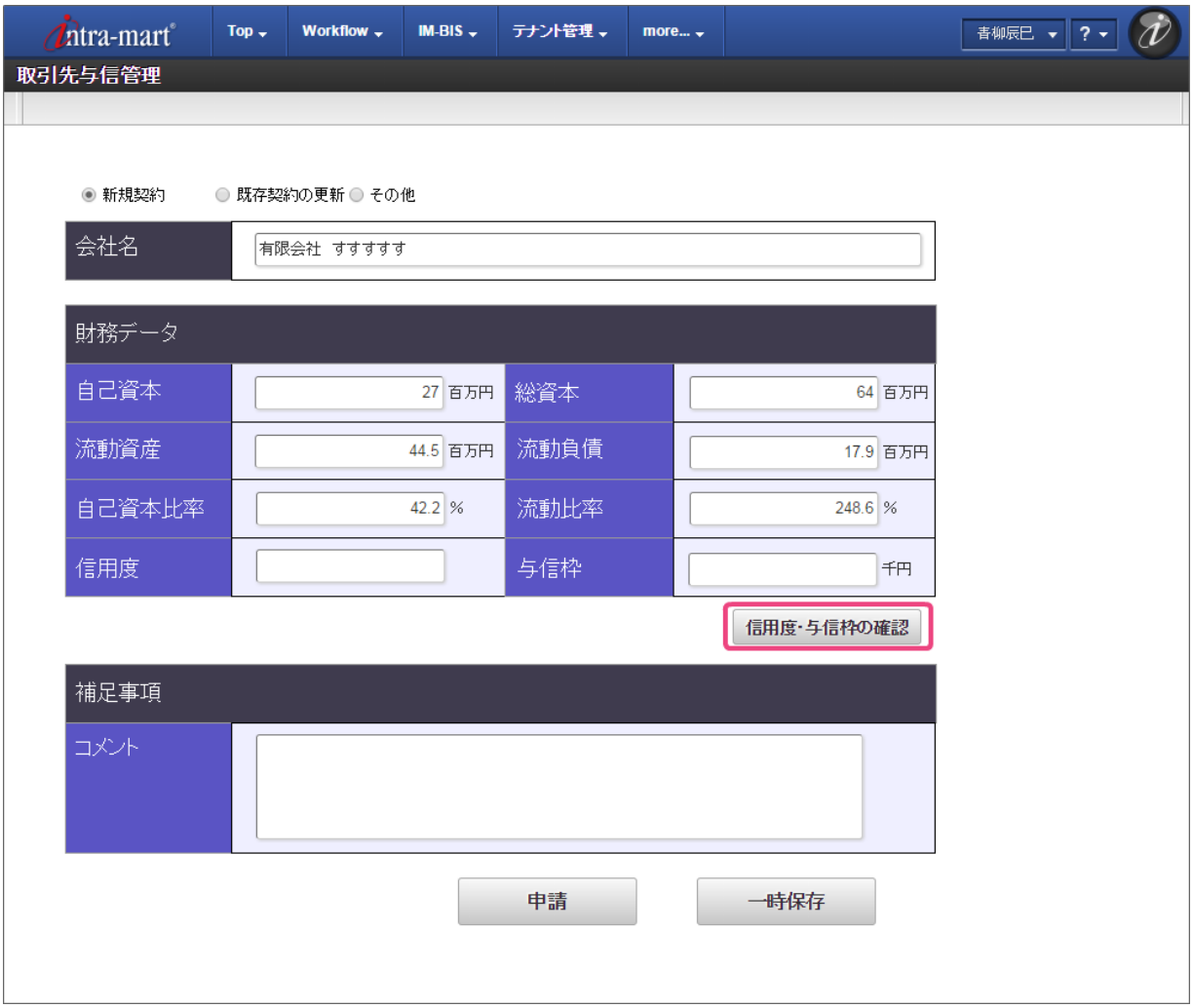
3. 「【ハンズオン】取引先与信管理」の「 申請／処理開始」をクリックして申請画面を表示します。

取引先の情報を入力して信用度・与信枠を評価し、申請を実行する

取引先に関する資産情報を入力して信用度・与信枠を確認し、申請してみましょう。

1. 申請画面で、取引先企業の資産情報を入力しましょう。

2. 「信用度・与信枠の確認」をクリックしてルールを実行してみましょう。



3. ルールが実行され、結果に基づく信用度や与信枠が表示されます。
   その他の項目に入力し、「申請」をクリックしてみましょう。

4. 申請の処理画面が表示されます。
   案件名を変更し、「申請／処理開始」をクリックしましょう。

5. 申請を行うことができました。
   続いて、承認を実行してみましょう。

取引先の与信管理の承認を実行する

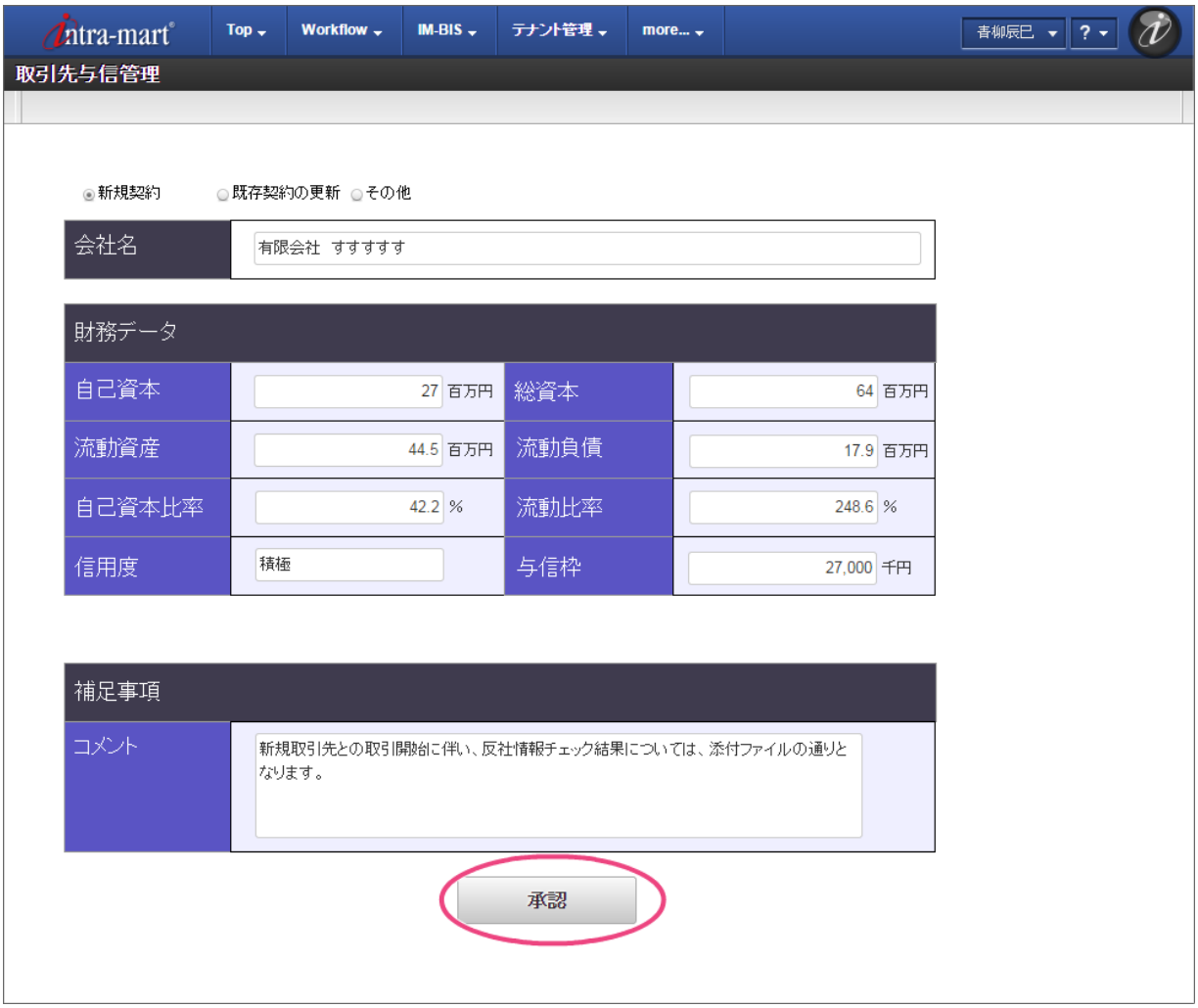
先ほど申請した案件を承認しましょう。

1. 「BIS担当者」ロールを付与したユーザでログインしましょう。
   （このマニュアルでは、「青柳辰巳」（ユーザコード：aoyagi）でログインします。）

2. 上部のメニューの「 IM-BIS 」→「ワークフロー」の順にマウスを重ねてから「未処理」をクリックしましょう。

3. 申請した「【ハンズオン】取引先与信管理」の案件が表示されます。
   「 処理」をクリックして承認画面を表示してください。

4. 申請内容がそのまま表示されていることが確認できました。
   「承認」をクリックしましょう。



5. このフローは承認で完了するため、案件が完了しました。
   以上で、取引先の与信管理のワークフローの実行について確認することができました。

入力チェックされた金額に応じて、ルートの形を変えてみよう

この章では、 IM-BIS を利用したワークフローの承認者を OpenRules の結果に基づいて設定するための手順を説明しています。

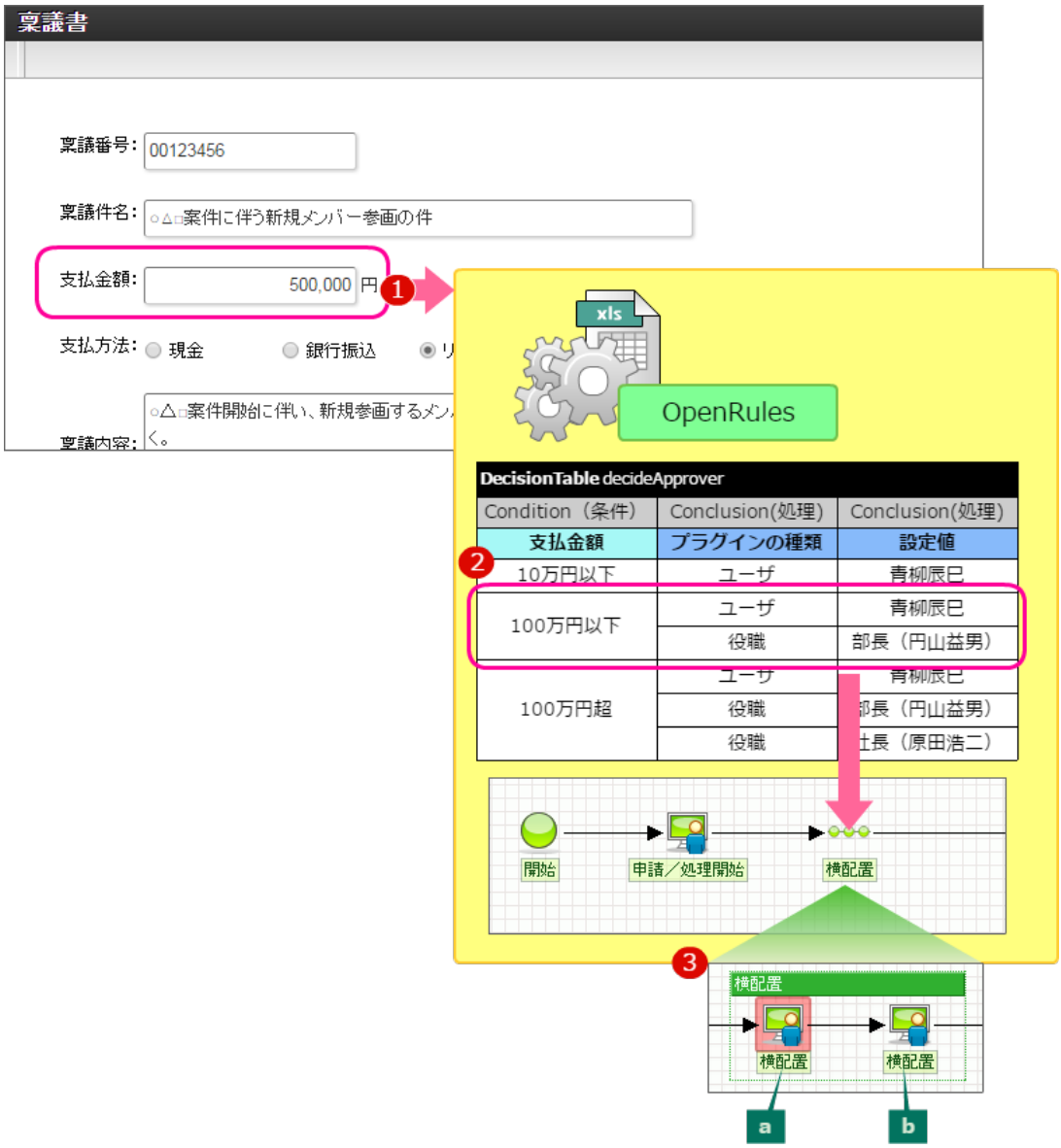
動的処理対象者の設定で OpenRules を利用する場合には、 IM-BIS の外部連携の動的処理対象者設定に合わせる必要があるため、画面上の値の処理の連携の設定時と以下の点が異なります。

テーブルタイプ（TableType）	**違い**
DecisionTable	▪ IM-BIS （データマッパー）のレスポンスのオブジェクトやフィールドの形式が固定 ▪ 返却する値は設定する役職やユーザのコードにする必要がある
Glossary	▪ レスポンス（返却）のオブジェクトの形式・パラメータが固定 ▪ リクエスト（入力）については、自由に設定できる
Data/Variable	▪ テーブルのキーワードが「Variable」限定になる ▪ レスポンス（返却）のオブジェクトの形式が固定
Datatype	▪ レスポンス（返却）のオブジェクトの形式・データ型が固定
Decision	▪ レスポンス（返却）のオブジェクトにインスタンスをセットするための記述方法が変わる
DecisionObject	▪ 処理結果をレスポンス（返却）のオブジェクトにセットするための式の記述方法が変わる

ハンズオンシナリオ（稟議フローの作成）の概要

このシナリオでは、以下のようなルールに基づいて、承認者が変更されるフローを作成します。

- 申請者が入力した申請書の「支払金額」の金額に基づいて、次の承認者の人数が変わるフローです。
  - 支払金額が10万円以下 → ユーザ（青柳辰巳）
  - 支払金額が100万円以下 → ユーザ（青柳辰巳）、役職：部長（円山益男）
  - 支払金額が100万円超 → ユーザ（青柳辰巳）、役職：部長（円山益男）、役職：社長（原田浩二）



1. 支払金額を入力して申請と同時にルールを実行する。
2. OpenRules で受け取った支払金額に基づいて条件を評価し、合致する条件の処理対象者を返却する。
3. 返却された処理対象者に基づいて横配置ノードを展開し、以下の処理対象者を設定する。
    a. 結果の「プラグイン：ユーザ、設定値：青柳辰巳」を設定
    b. 結果の「プラグイン：役職、設定値：部長（円山益男）」を設定

このシナリオを通して習得できる知識

このシナリオを実行すると、以下の知識を理解することができます。

- OpenRules でルールの評価結果を IM-BIS のルートの「動的承認」や「縦配置」「横配置」のノードの処理対象者に設定するための記述方法

このシナリオを学習する前に必要な知識

このシナリオの実行に当たり、下記の知識を事前に確認してください。

- IM-BIS での外部連携を利用した動的処理対象者設定の手順
  詳細については、以下のドキュメントを参照してください。
    - 「IM-BIS 業務管理者操作ガイド」-「動的ノード（動的承認、縦配置、横配置）の処理対象者条件を設定する」
    - 「IM-BIS 仕様書」-「動的処理対象者設定に関する仕様」

ワークフローの動的承認者を決定する OpenRules のルール定義ファイルを作成する

ワークフローと OpenRules を連携して、処理対象者を動的に設定するためには、 IM-BIS の動的承認者設定の決まりに基づいて返却値を設定する必要があります。

このハンズオンでは、稟議フローをサンプルとして、申請書の金額に基づいて承認者を変更できるルールを作成します。

ルールを定義するExcelファイルを作成する手順
- このシナリオで作成するルールの概要
- ルールのExcelファイルを作成する手順
- 動的処理者設定のハンズオンを開始するための準備
- Excelファイルに返却する処理対象者の一覧を作成する
- 条件と処理対象者一覧を関連付ける
- 実行に必要な定義を設定する

このシナリオで作成するルールの概要

- 作成するルールの内容

  申請書の「支払金額」に応じて、次のノードの承認者（処理対象者）を変更する

  - 入力値：支払金額
  - 出力値：次の横配置ノードの処理対象者

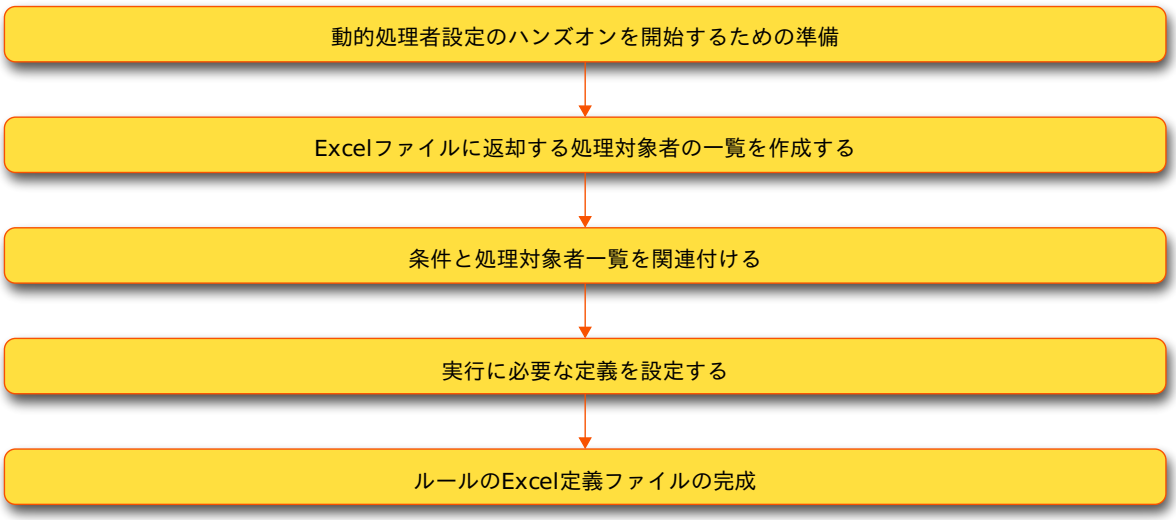
  条件と次の処理対象者の組み合わせは、以下の通りです。

  - 支払金額が10万円以下の場合、「ユーザ：青柳辰巳」を設定する
  - 支払金額が100万円以下の場合、「ユーザ：青柳辰巳」「役職：部長」（円山益男）を設定する
  - 支払金額が100万円超の場合、「ユーザ：青柳辰巳」「役職：部長」（円山益男）「役職：社長」（原田浩二）の3人を設定する
  - 支払金額がいずれの条件にも当てはまらない場合、「ユーザ：青柳辰巳」「役職：部長」（円山益男）「役職：社長」（原田浩二）の3人を設定する

ルールのExcelファイルを作成する手順

新規にExcelファイルを作成し、 OpenRules の実行に必要な表（テーブル）を順番に作成していきます。
このシナリオでは、以下の図の流れで作成していきます。

- ワークフローの動的承認者を決定する OpenRules のルール定義ファイルの作成の手順



動的処理者設定のハンズオンを開始するための準備

このハンズオンを開始するための準備をする

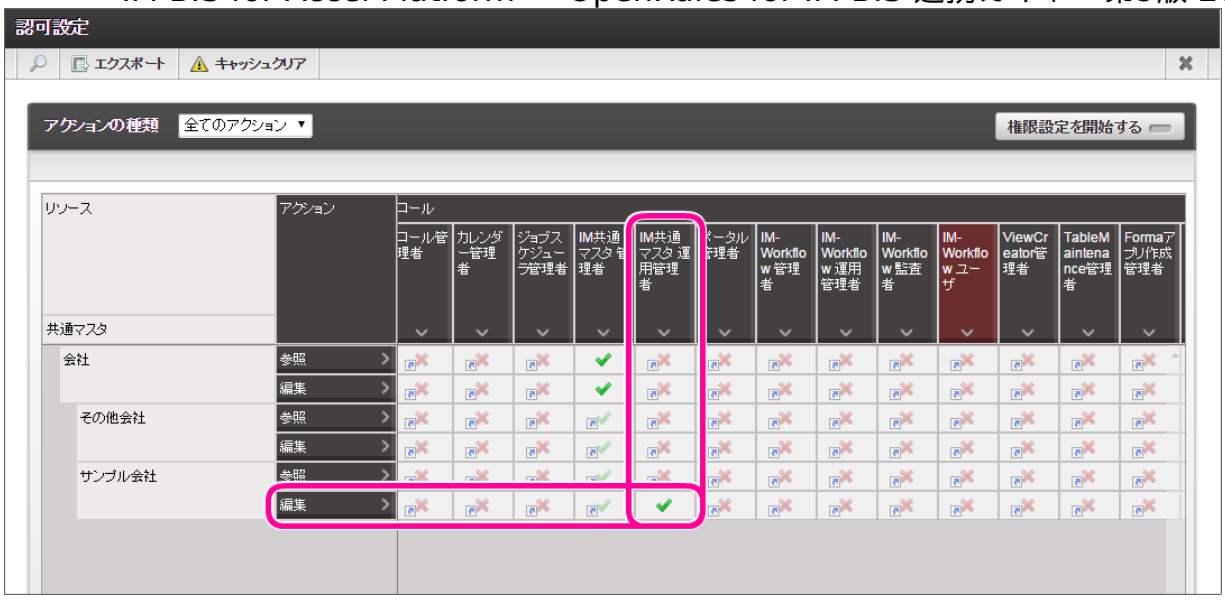
このハンズオンの作業中に、処理対象者のユーザなどを設定するために、IM共通マスタのデータを参照する必要があります。
ハンズオンの実行前には、以下の設定を行ってください。

対象のユーザへの「IM 共通マスタ 運用管理者」ロールの付与

設定方法は「 テナント管理者操作ガイド 」の「ロールを設定する」を参照してください。

「IM 共通マスタ 運用管理者」ロールへの「サンプル会社」の認可の参照の許可設定

「 IM-共通マスタ 管理者操作ガイド 」の「会社情報の参照権を登録する」の手順を参考に、「IM共通マスタ運用管理者」ロールに対して、認可の「サンプル会社」の「編集」を「許可」に設定してください。



動的処理対象者設定テンプレートの編集を開始する

このハンズオンでは、ダウンロードの章で公開しているテンプレートを変更しながら、 OpenRules で動的処理者設定を定義していきます。
まずは、テンプレートファイルを入手しましょう。

1. *OpenRules のテンプレート* から「動的処理対象者設定テンプレート」をダウンロードしてください。
2. ファイルを別名で保存した後に、ファイルの編集を開始してください。

Excelファイルに返却する処理対象者の一覧を作成する

処理対象者を返却するルールを作成する場合には、最初に返却する処理対象者の一覧を作成します。
この一覧では、処理対象者の設定値を識別するID情報と、処理対象者プラグインの種類（ユーザや組織、役職など）、プラグインにあわせた設定値（ユーザコード、組織コード等）をまとめます。

以下のように設定する処理対象者のプラグインの種類（組織やユーザ、役職など）と対応するコード（組織コードやユーザコードなど）をExcel上にまとめていきます。

1. 支払金額と処理対象者の関係をまとめた *DecisionTable* に基づいて、処理対象者の一覧を作成します。

DecisionTable decideApprover

Condition（条件）	Conclusion(処理)	Conclusion(処理)
支払金額	**プラグインの種類**	**設定値**
10万円以下	ユーザ	青柳辰巳
100万円以下	ユーザ	青柳辰巳
	役職	部長（円山益男）
100万円超	ユーザ	青柳辰巳
	役職	部長（円山益男）
	役職	社長（原田浩二）

重複する処理対象者は1つにまとめた結果は以下の表の通りです。

プラグイン種類	設定内容
ユーザ	青柳辰巳
役職	部長（円山益男）
役職	社長（原田浩二）

2. IM-BIS のルートに処理対象者を設定するために必要な情報を以下の表にまとめます。
先の手順でまとめた３つの処理対象者は、それぞれ1行ずつ設定を記述しています。
表の形式の詳細は「 *【動的処理者設定】処理対象者プラグイン設定一覧（Data IMDynamicUser）* 」を参照してください。

Data **IMDynamicUser** settings											
id	process SetNo	code	company Cd	department SetCd	userCd	departmentCd	compare	postCd	publicGroup SetCd	publicGroup Cd	roleCd
ID	処理設定No	**判定用コード**	会社コード	会社組織セットコード	ユーザコード	組織コード	比較	役職コード	パブリックグループセットコード	パブリックグループコード	役割コード
1	1	1			aoyagi				a		
2	2	4	comp_sample_01	comp_sample_01				ps002	b		
3	3	4	comp_sample_01	comp_sample_01				ps001	c		

a. ユーザ：青柳辰巳の設定です。

b. 役職：部長（円山益男）の設定です。

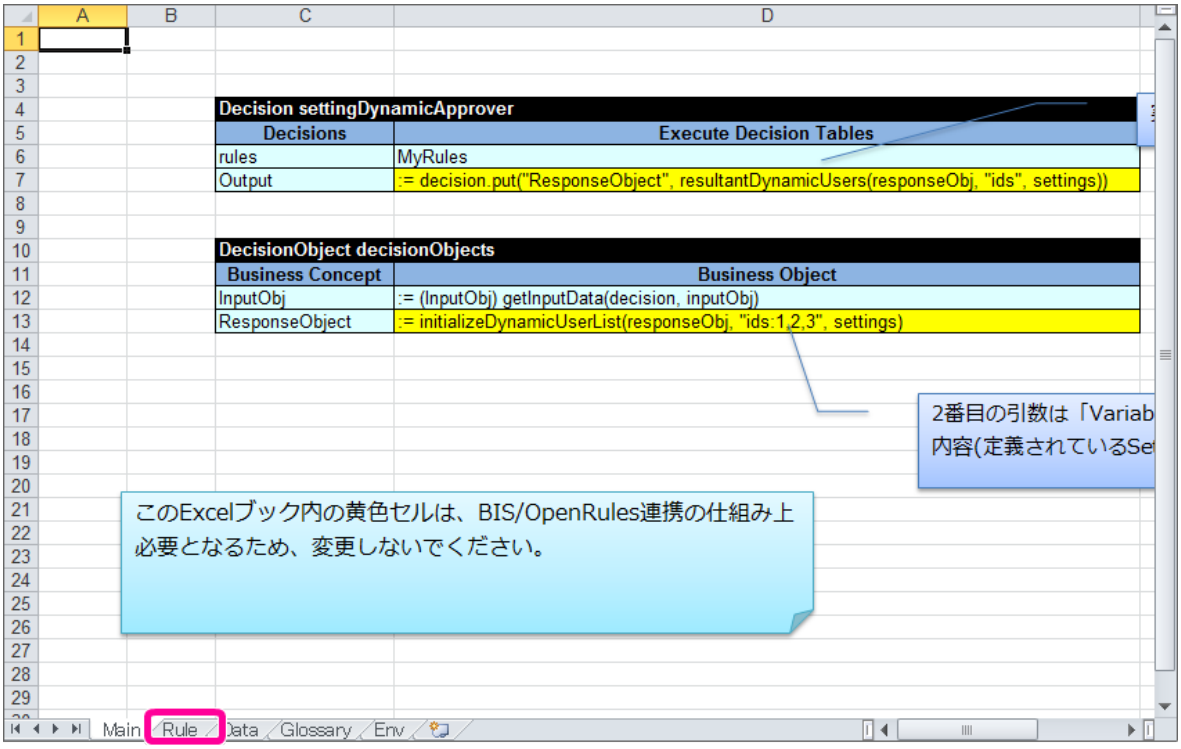
c. 役職：社長（原田浩二）の設定です。

表の青枠で囲まれた左の列のIDを *DecisionTable* の返却値（ *Conclusion* ）に設定すると、 IM-BIS のルートの処理対象者として扱うことができます。

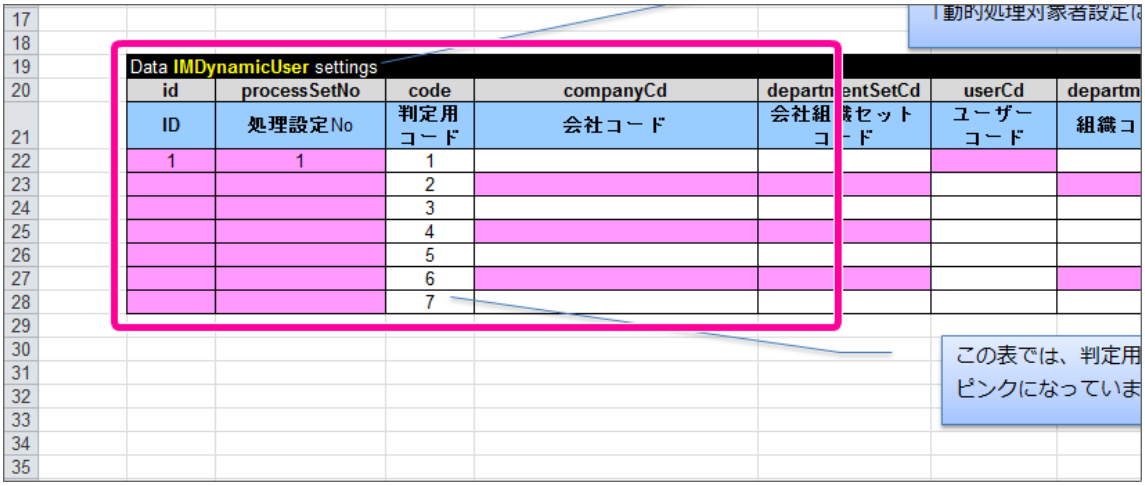
処理対象者一覧の入力欄を必要な行のみに修正する

テンプレートの「処理対象者一覧」は、判定用コード（プラグインの種類）の全ての入力欄が表示されています。
ハンズオンで必要な判定用コードのみを表示するように変更しましょう。

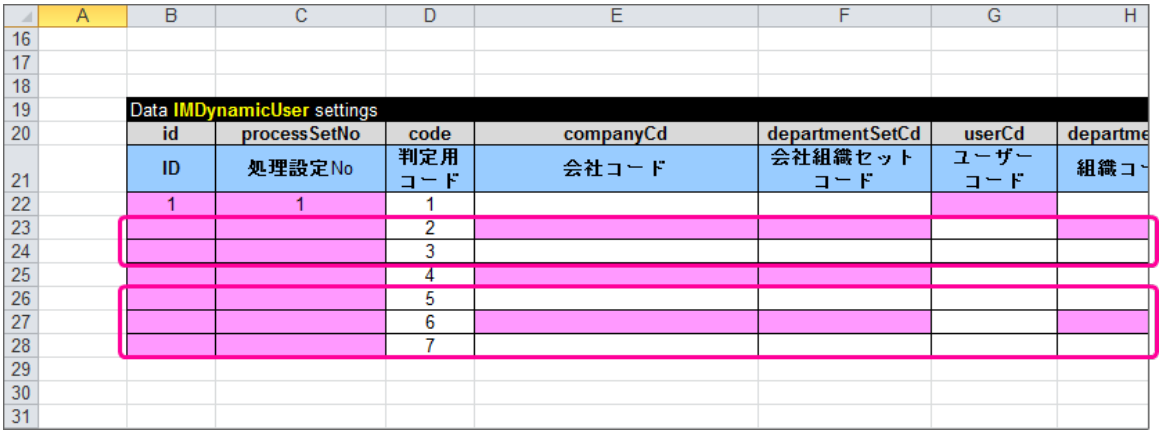
1. 編集中のExcelファイルの「Rule」シートを表示してください。



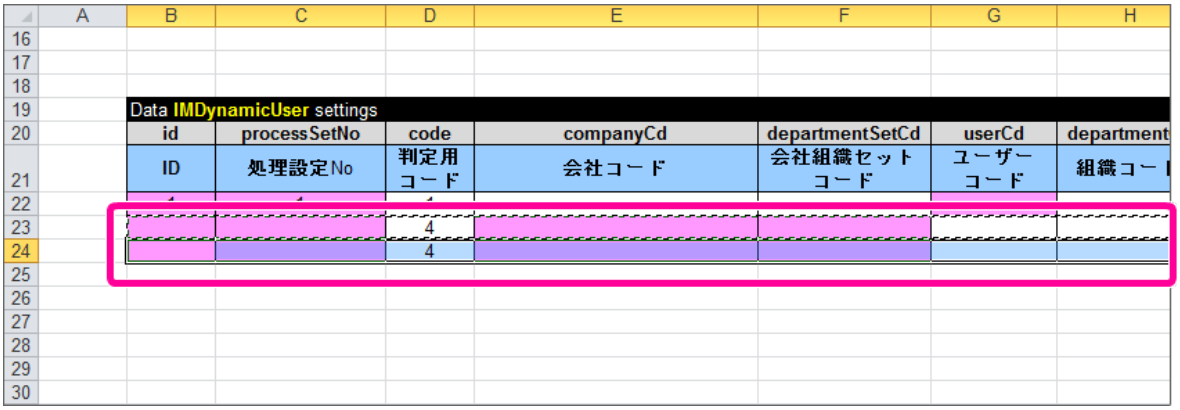
2. 「Rule」シートから「Data IMDynamicUser settings」テーブルを表示してください。
（シート上の説明は、削除してください。）



3. 今回のハンズオンでは、処理対象者プラグインの種類が「ユーザ」と「役職」の2種類を扱います。
   この2種類のプラグインを表す判定用コードは「1」（ユーザ）と「4」（役職）です。
   「判定用コード」の値が「1」と「4」の行を残し、それ以外の行をExcelの表から削除してください。
   以下の図の赤枠の行が削除対象です。



4. 「役職」を使った処理対象者のパターンは、「部長」と「社長」の2パターンがあります。
   判定用コードが「4」（役職）の行をコピーして2行にしてください。

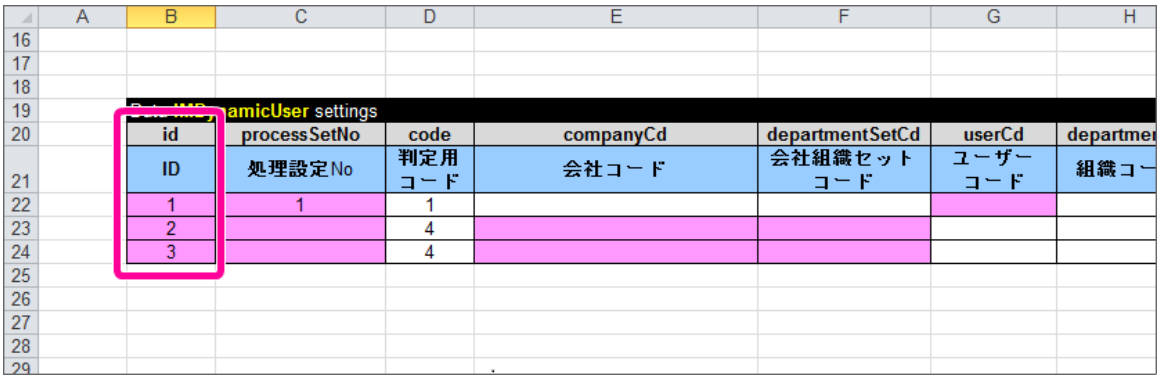


5. これで、処理対象者一覧の入力欄を必要な行のみに変更することができました。
   続いて、各行に必要な設定値を入力していきましょう。

処理対象者に必要な設定値を確認して入力する

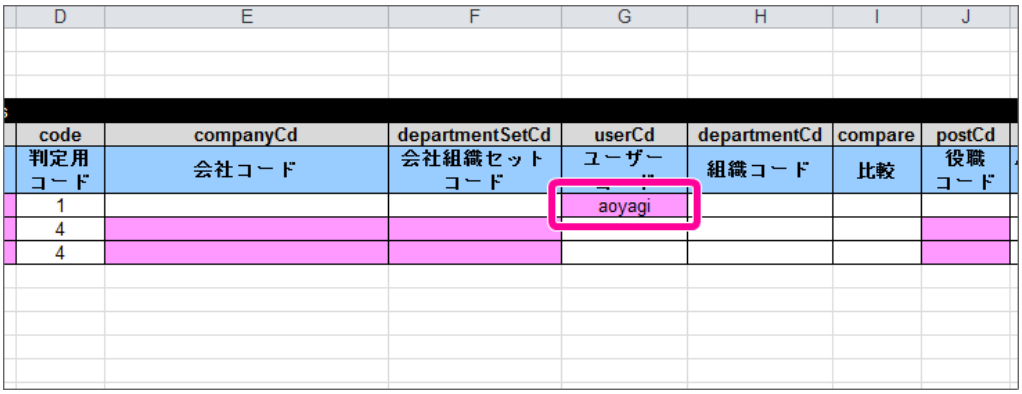
テンプレートには、判定用コードに合わせた必須項目がピンク色のセルで表されています。
このセルを埋める形で処理対象者として返却するために必要な情報を設定していきましょう。

1. 変更した処理対象者一覧の「対象者ID」（ID）は、ユニークな番号を設定する必要があります。
   上から順に1、2、3と番号を入力してください。



id	processSetNo	code	companyCd	departmentSetCd	userCd	departmer
ID	処理設定No	判定用コード	会社コード	会社組織セットコード	ユーザーコード	組織コー
1	1	1				
2		4				
3		4				

2. 続いて、1行目の処理対象者には「ユーザ：青柳辰巳」と返却するための設定を行いましょう。
「ユーザ：青柳辰巳」の設定内容として「ユーザコード」の列に「aoyagi」と入力してください。

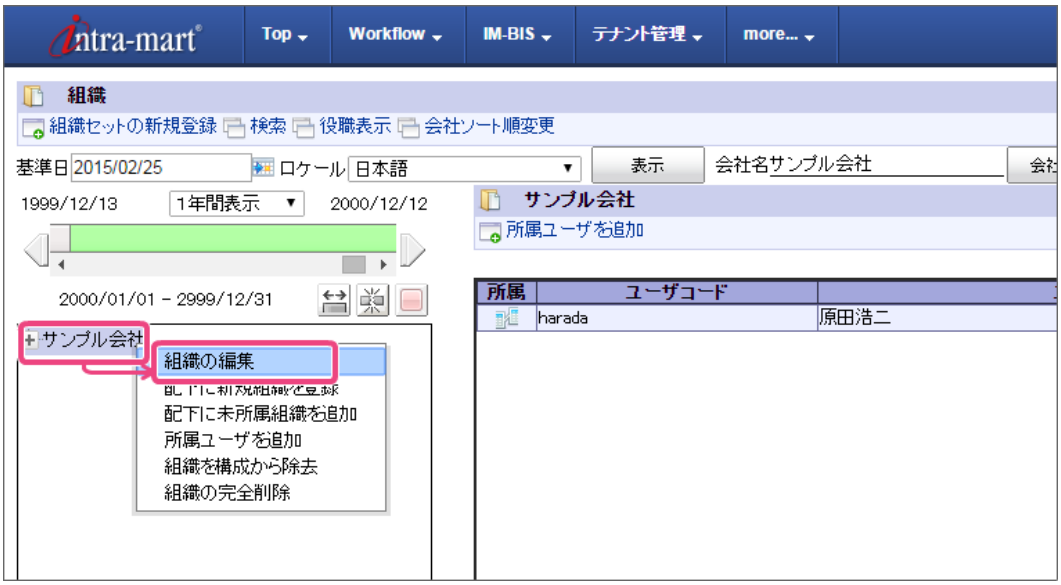


code	companyCd	departmentSetCd	userCd	departmentCd	compare	postCd
判定用コード	会社コード	会社組織セットコード	ユーザー	組織コード	比較	役職コード
1			aoyagi			
4						
4						

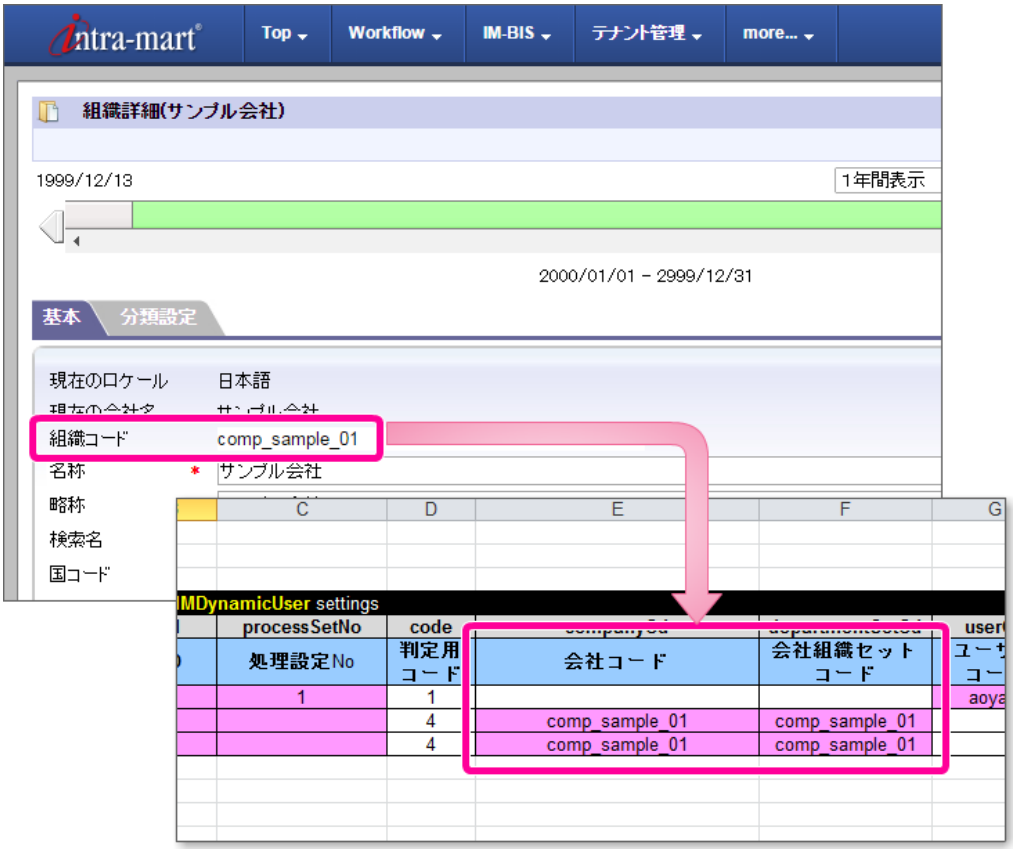
> **コラム**
> 対象のユーザのユーザコードが分からない場合には、IM共通マスタのユーザのメンテナンス画面から確認してください。

3. 2行目、3行目には、役職の「部長」や「社長」を返却するための設定を行います。
設定に必要な情報を確認するために、IM共通マスタのメンテナンス情報を表示してください。
「青柳辰巳」でログイン後、サイトマップから「共通マスタ」の「組織」をクリックしてください。

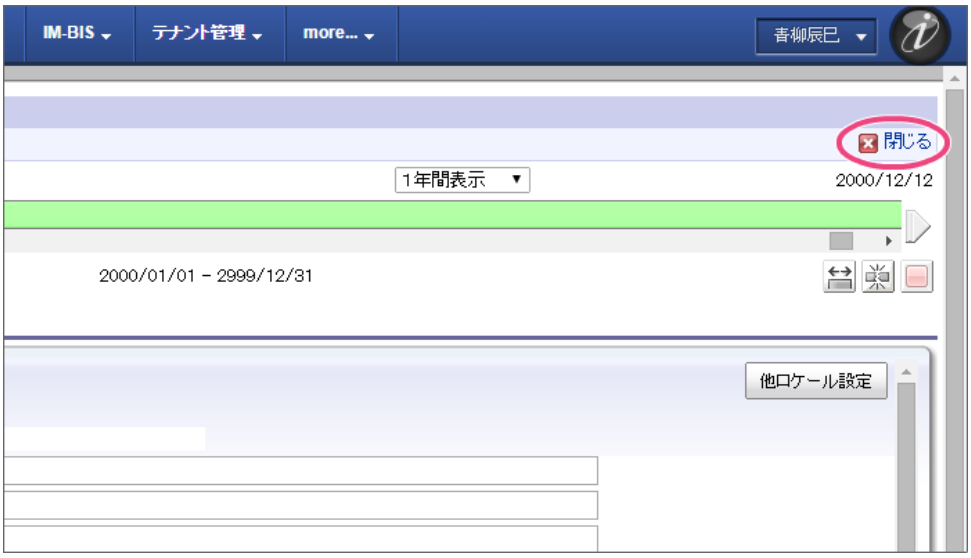
4. 「役職」を設定する際には、「会社コード」「会社組織セットコード」が必要です。
「会社コード」「会社組織セットコード」を確認するために、「サンプル会社」を右クリック後に表示されるメニューから「組織の編集」をクリックしてください。



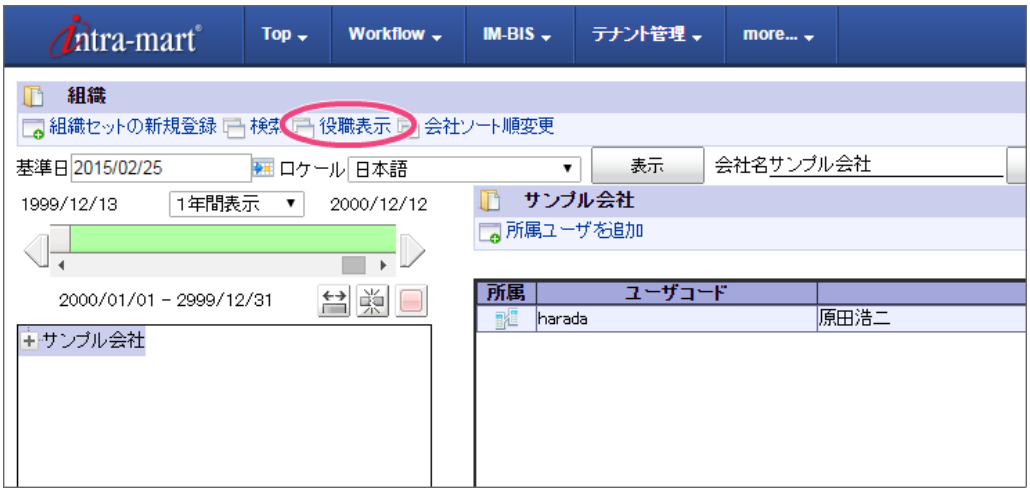
5. 組織セットが1つしかない会社の場合には、この表示されている会社の「組織コード」が「会社コード」「会社組織セットコード」に該当します。
この「組織コード」をExcelの「会社コード」「会社組織セットコード」のセルに入力してください。



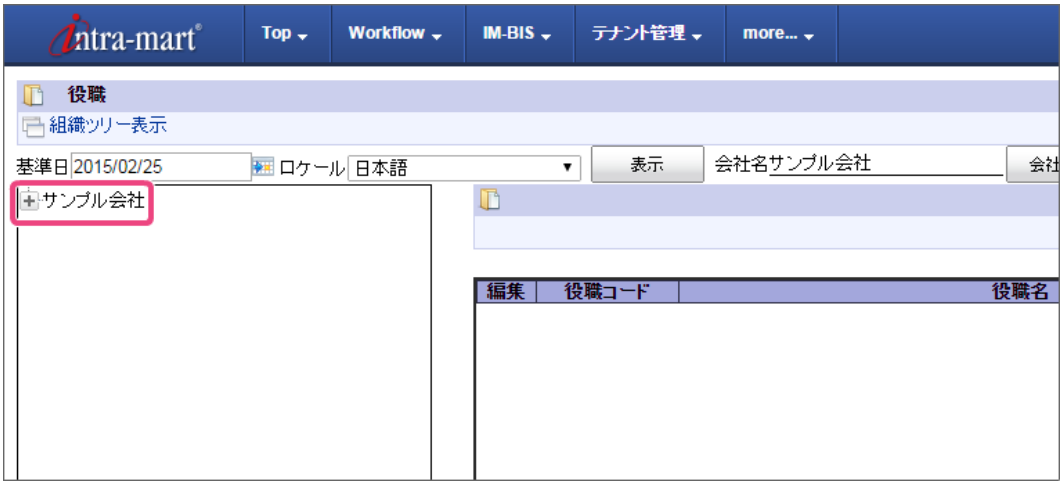
6. 会社（組織）の編集画面を右上の「閉じる」をクリックして閉じてください。



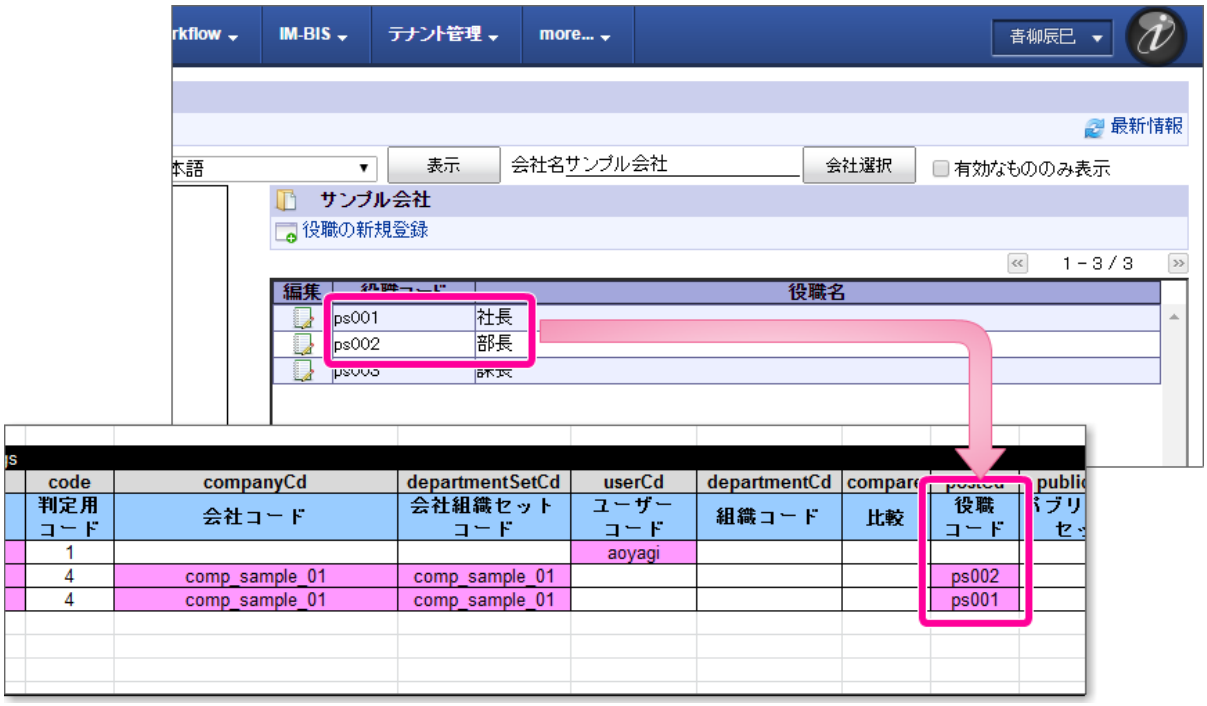
7. 「役職表示」をクリックして、「役職」のメンテナンス画面を表示してください。



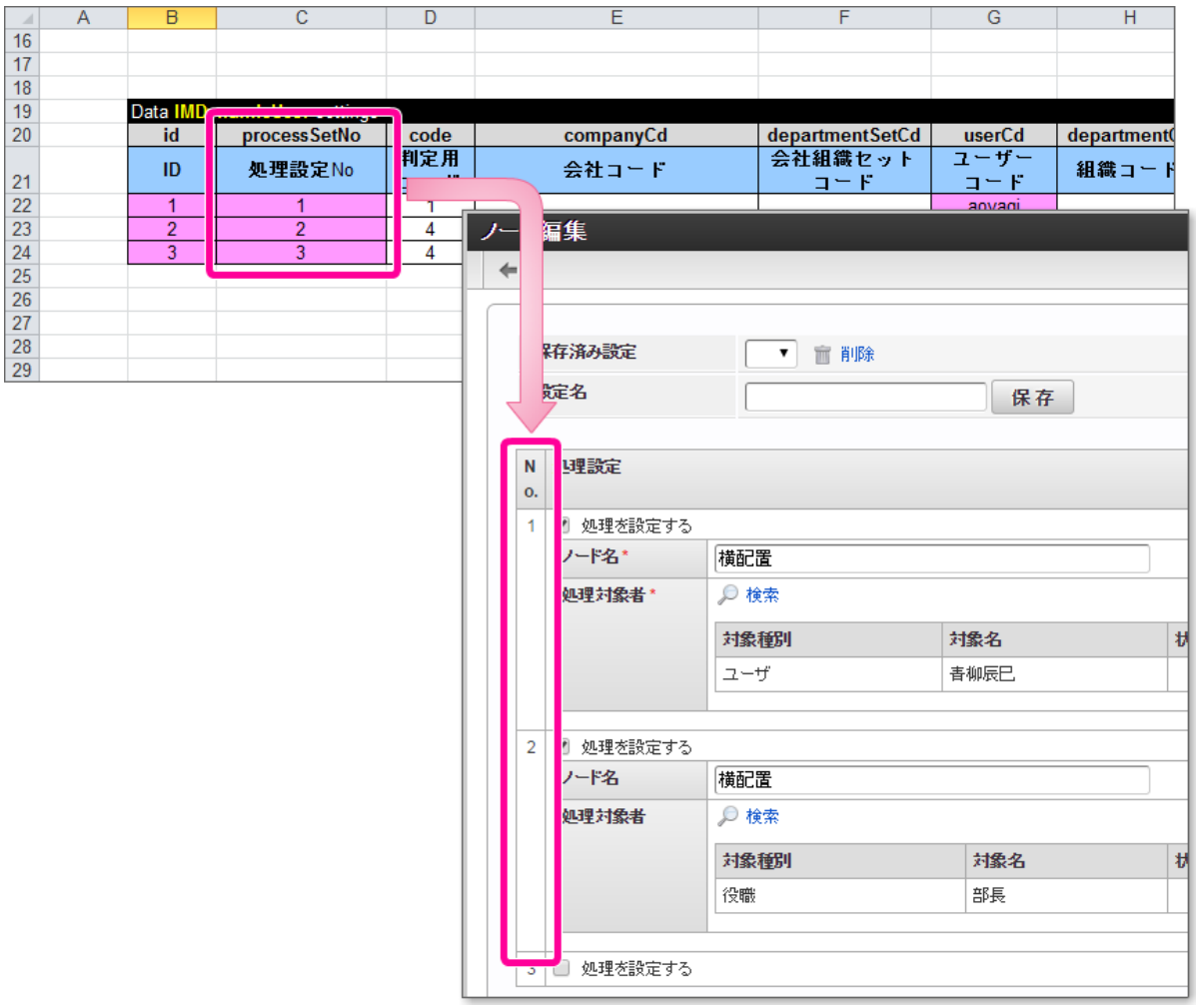
8. 左側から「会社名（サンプル会社）」をクリックしてください。



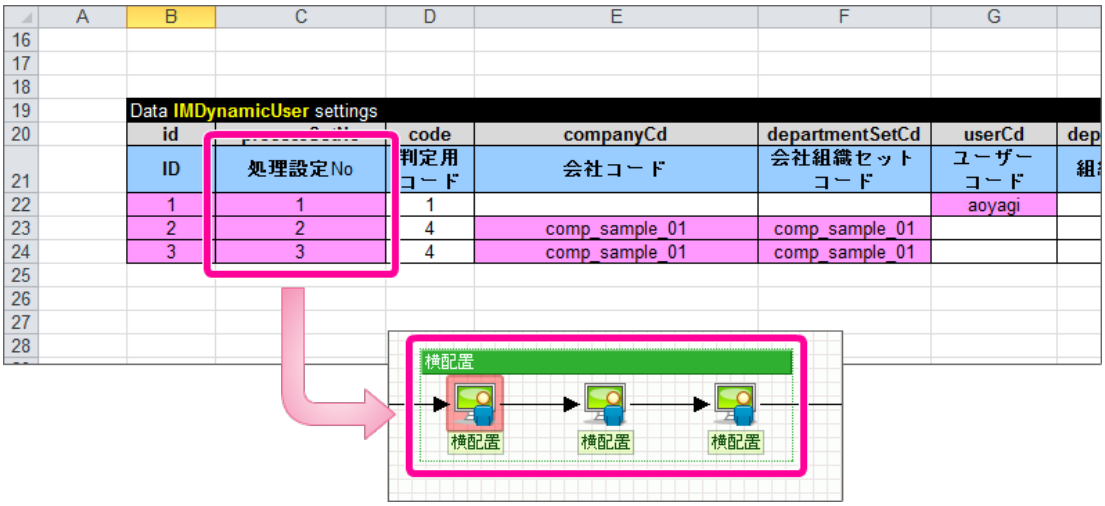
9. 右側に役職の一覧が表示されますので、表示されている役職の役職コードをExcelファイルの「役職コード」に入力してください。



10. 「処理設定No」の設定値は、以下のように縦配置・横配置ノードの設定画面で設定するノードの順に対応しています。
それぞれ表の上から順にノードに設定できるように、処理設定Noには1、2、3と入力してください。



11. 処理設定Noを1、2、3と異なる数字に設定したため、実際にフローを実行したときには以下の図のように3つのノードとして展開されます。



Data IMDynamicUser settings

id	processSetNo	code	companyCd	departmentSetCd	userCd	dep
ID	処理設定No	判定用コード	会社コード	会社組織セットコード	ユーザーコード	組
1	1	1			aoyagi	
2	2	4	comp_sample_01	comp_sample_01		
3	3	4	comp_sample_01	comp_sample_01		

> **コラム**
> この「処理設定No」を同じ番号にした場合には、1つのノードの処理対象者として設定されるため、青柳辰巳・部長・社長のいずれかが承認した時点で完了するフローが設定できます。

12. これで、処理対象者一覧が完成しましたので、一度ファイルを保存しましょう。

条件と処理対象者一覧を関連付ける

処理対象者の一覧を作成できましたので、この一覧の結果と条件を関連付けるための表を作成します。

以下の図のように、処理対象者一覧の表と処理対象者を決定するための *DecisionTable* をIDとSetting IDsによって関連付けていきます。

- 処理対象者の一覧を定義した *Data/Variable*

Data IMDynamicUser settings

id	processSetNo	code	companyCd	departmentSetCd	userCd	departmentCd	compare	postCd	publicGroupSetCd	publicGroupCd	roleCd
ID	処理設定No	判定用コード	会社コード	会社組織セットコード	ユーザーコード	組織コード	比較	役職コード	パブリックグループセットコード	パブリックグループコード	役割コード
1	1	1			aoyagi						
2	2	4	comp_sample_01	comp_sample_01				ps002			
3	3	4	comp_sample_01	comp_sample_01				ps001			

- 処理対象者を支払金額に基づいて決定する *DecisionTable*

DecisionTable decideApprover

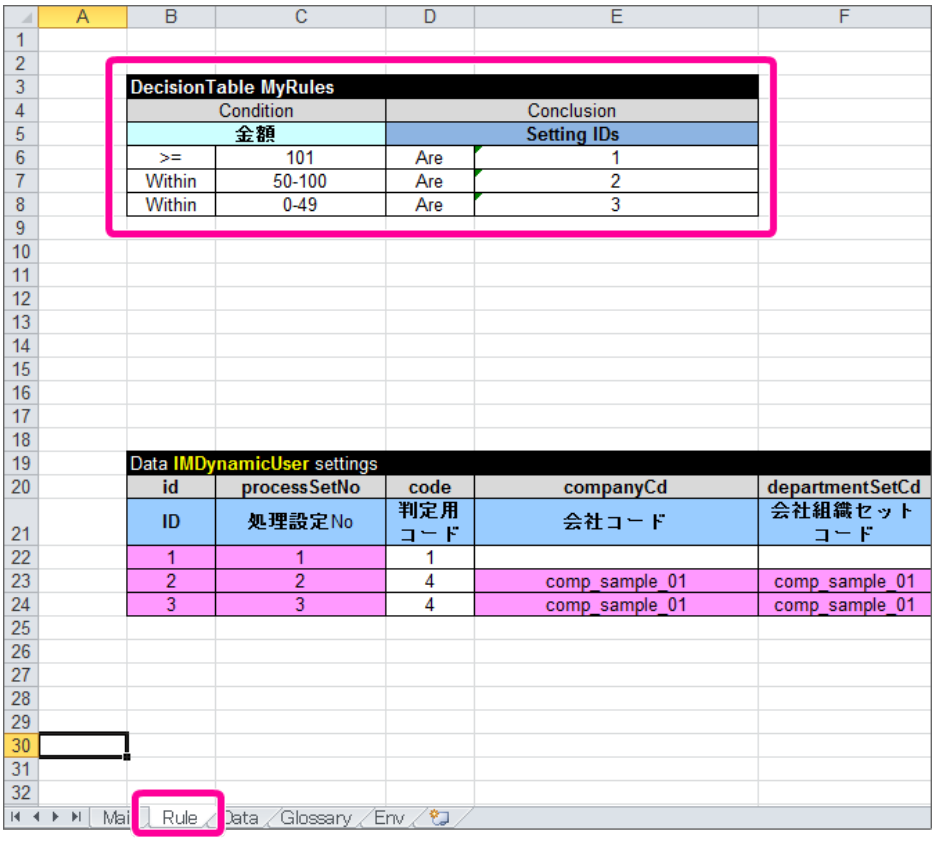
Condition		Conclusion	
金額		Setting IDs	
<=	100,000	Are	1
<=	1,000,000	Are	1,2
>	1,000,000	Are	1,2,3

Setting IDs
IM-BIS の動的処理対象者設定と連携する場合の「処理対象者一覧設定のID列の値」を表します。
DecisionTable に記述されたIDの値に基づいて、ルートの処理対象者を設定するために必要な情報を処理対象者一覧の表から取得します。

DecisionTable に条件を設定する

今回作成するワークフローでは、画面の「支払金額」に応じて、処理対象者を変更します。
まず、「支払金額の条件」をExcel上に設定していきましょう。

1. 編集したExcelファイルを開いてください。
2. 「Rule」シートの「DecisionTable」を表示してください。



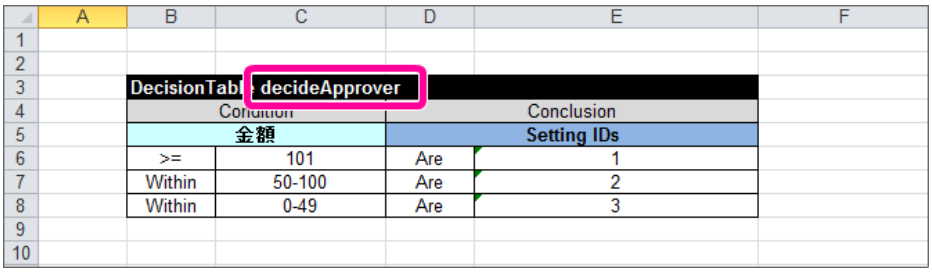
DecisionTable MyRules

Condition		Conclusion	
金額		Setting IDs	
>=	101	Are	1
Within	50-100	Are	2
Within	0-49	Are	3

Data IMDynamicUser settings

id	processSetNo	code	companyCd	departmentSetCd
ID	処理設定No	判定用コード	会社コード	会社組織セットコード
1	1	1		
2	2	4	comp_sample_01	comp_sample_01
3	3	4	comp_sample_01	comp_sample_01

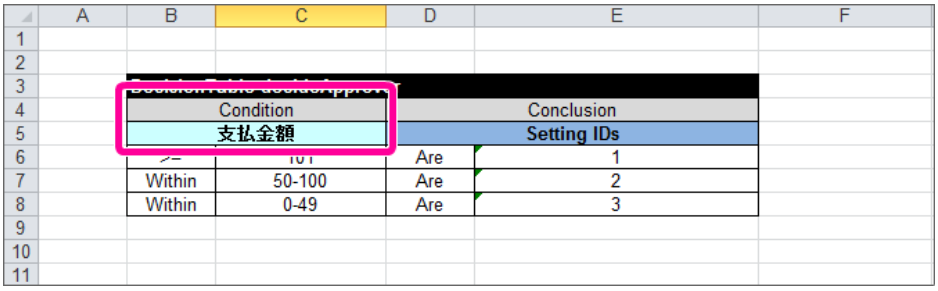
3. テーブルの名前を「decideApprover」に変更してください。



DecisionTable decideApprover

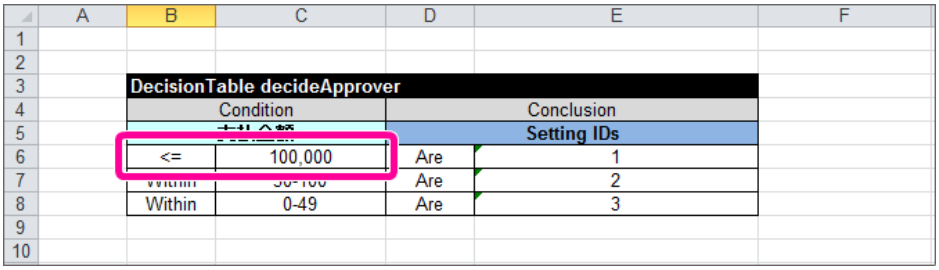
Condition		Conclusion	
金額		Setting IDs	
>=	101	Are	1
Within	50-100	Are	2
Within	0-49	Are	3

4. *Condition* の項目を「支払金額」に変更してください。



Condition		Conclusion	
支払金額		Setting IDs	
>=	101	Are	1
Within	50-100	Are	2
Within	0-49	Are	3

5. 1行目には、「支払金額が10万円以下」の条件を記述します。
演算子と組み合わせて表現するために、左に「<=」、右に「100000」と入力してください。

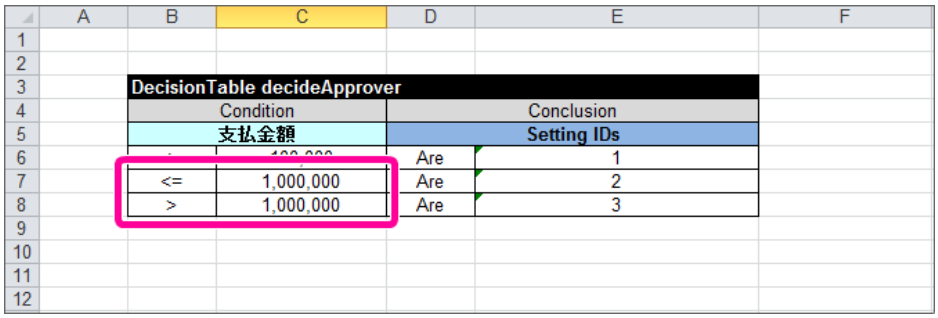


DecisionTable decideApprover

Condition		Conclusion	
		Setting IDs	
<=	100,000	Are	1
		Are	2
Within	0-49	Are	3

6. 同じように残りの条件を以下の通りに入力してください。
   - 2行目に設定する条件：支払金額が100万円以下
     左：<=
     右：1000000
   - 3行目に設定する条件：支払金額が100万円超

左：>
右：1000000



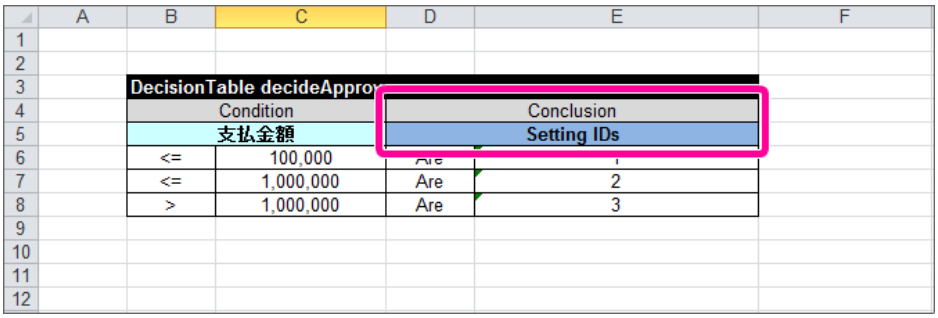
DecisionTable decideApprover			
Condition		Conclusion	
支払金額		Setting IDs	
<=	1,000,000	Are	1
<=	1,000,000	Are	2
>	1,000,000	Are	3

7. これで、条件が設定できましたので、次の手順で結果を設定していきましょう。

DecisionTable の結果に処理対象者を設定する

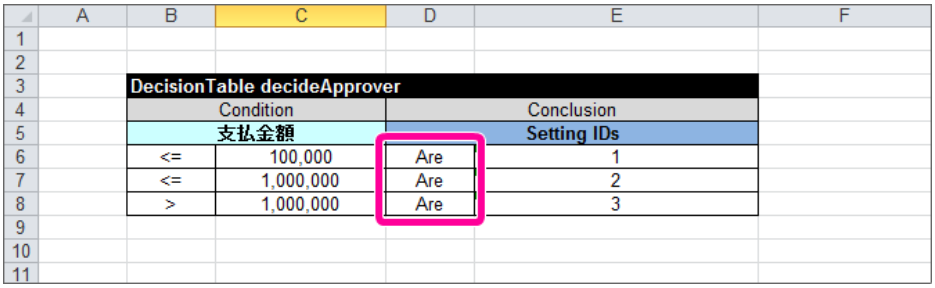
先の手順で設定した条件の結果に処理対象者を設定していきましょう。

1. 編集中の *DecisionTable* の *Conclusion* には、「Setting IDs」と設定されていることを確認してください。
   これは、処理対象者を返却する列を表します。



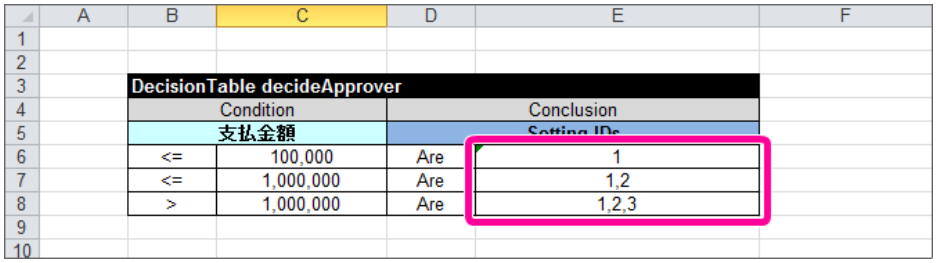
DecisionTable decideApprover			
Condition		Conclusion	
支払金額		Setting IDs	
<=	100,000		
<=	1,000,000	Are	2
>	1,000,000	Are	3

2. *Conclusion* の演算子は、「Are」が設定されています。
   処理対象者を返却する場合には、複数の処理対象者を返却することができるように、演算子「Are」を使います。



DecisionTable decideApprover			
Condition		Conclusion	
支払金額		Setting IDs	
<=	100,000	Are	1
<=	1,000,000	Are	2
>	1,000,000	Are	3

3. *Conclusion* の値の欄には、先に作成した処理対象者一覧（Data IMDynamicUser）の「ID」の値を入力しましょう。
   各条件に合わせて、以下の通りに入力してください。
   2行目や3行目のように複数の処理対象者を返却する場合には、カンマ区切りで入力してください。
   - 1行目：1（ユーザ：青柳辰巳）
   - 2行目：1,2（ユーザ：青柳辰巳、役職：部長）
   - 3行目：1,2,3（ユーザ：青柳辰巳、役職：部長、役職：社長）



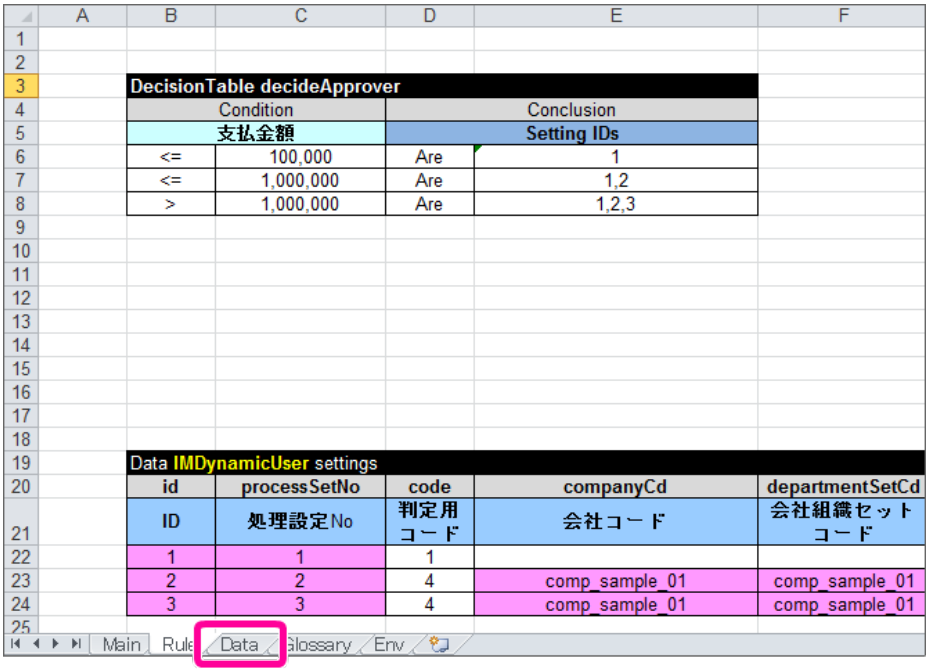
DecisionTable decideApprover			
Condition		Conclusion	
支払金額			
<=	100,000	Are	1
<=	1,000,000	Are	1,2
>	1,000,000	Are	1,2,3

4. これで、条件に応じて処理対象者を返却するためのルールが作成できました。
   最後に、Excelファイル内で必要な定義を設定しておきましょう。

実行に必要な定義を設定する

ルールの表が完成しましたので、実行に必要なその他の定義を設定していきましょう。

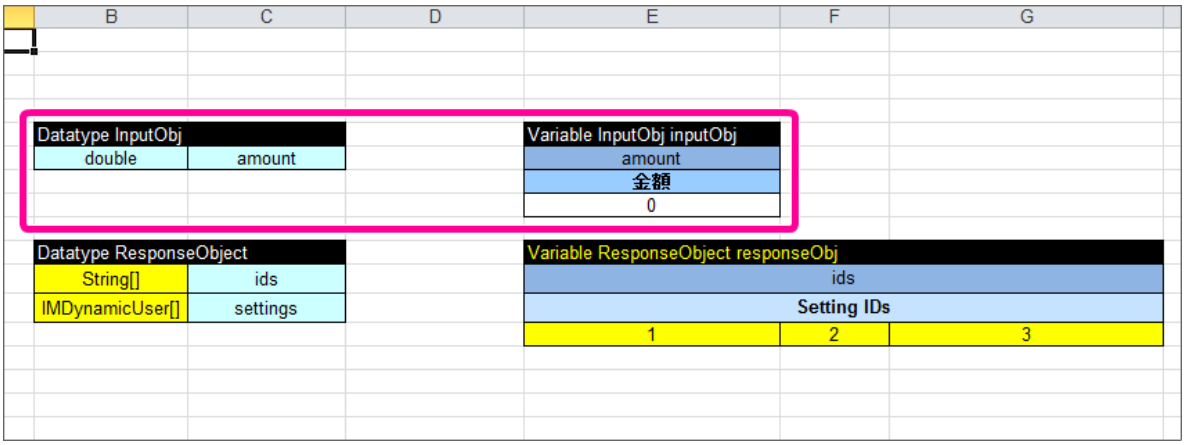
1. 編集中のExcelファイルの「Data」シートを表示してください。



DecisionTable decideApprover			
Condition		Conclusion	
支払金額		Setting IDs	
<=	100,000	Are	1
<=	1,000,000	Are	1,2
>	1,000,000	Are	1,2,3

Data IMDynamicUser settings				
id	processSetNo	code	companyCd	departmentSetCd
ID	処理設定No	判定用コード	会社コード	会社組織セットコード
1	1	1		
2	2	4	comp_sample_01	comp_sample_01
3	3	4	comp_sample_01	comp_sample_01

2. このシート中の *ResponseObject* は、実行に必要な設定が行われているため、何も変更しません。
入力に対応する「InputObject」に関する定義に必要な設定を行っていきましょう。



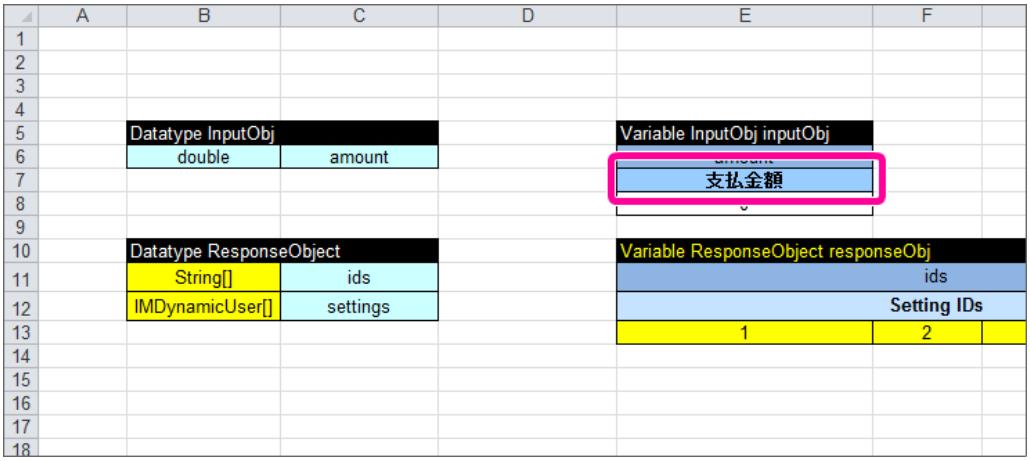
Datatype InputObj	
double	amount

Variable InputObj inputObj
amount
金額
0

Datatype ResponseObject	
String[]	ids
IMDynamicUser[]	settings

Variable ResponseObject responseObj		
ids		
Setting IDs		
1	2	3

3. InputObjの「Variable」の表で項目の論理名を「支払金額」と変更しましたので、同じように「支払金額」に変更してください。



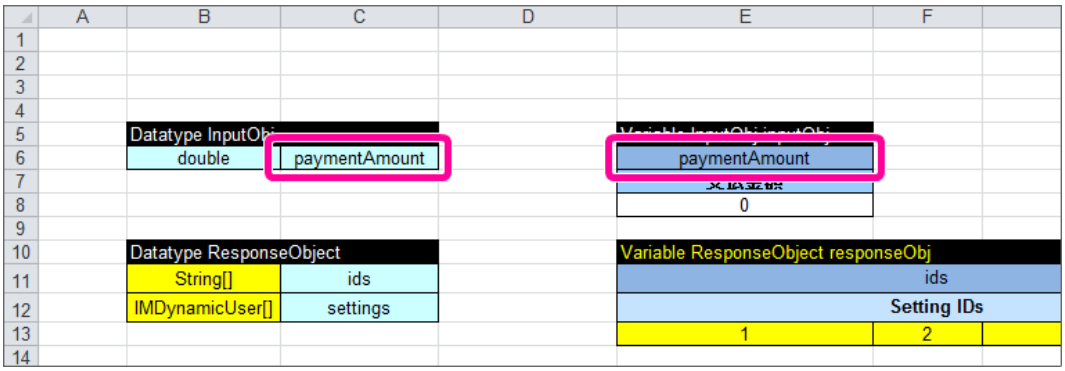
Datatype InputObj	
double	amount

Variable InputObj inputObj
支払金額

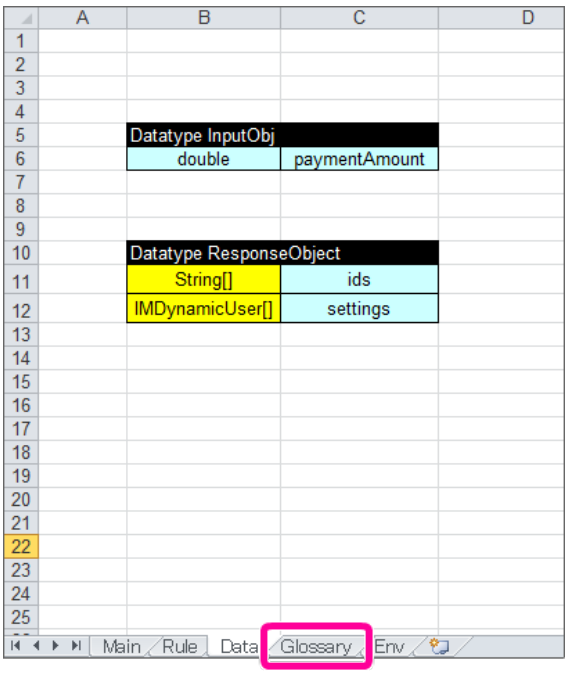
Datatype ResponseObject	
String[]	ids
IMDynamicUser[]	settings

Variable ResponseObject responseObj	
ids	
Setting IDs	
1	2

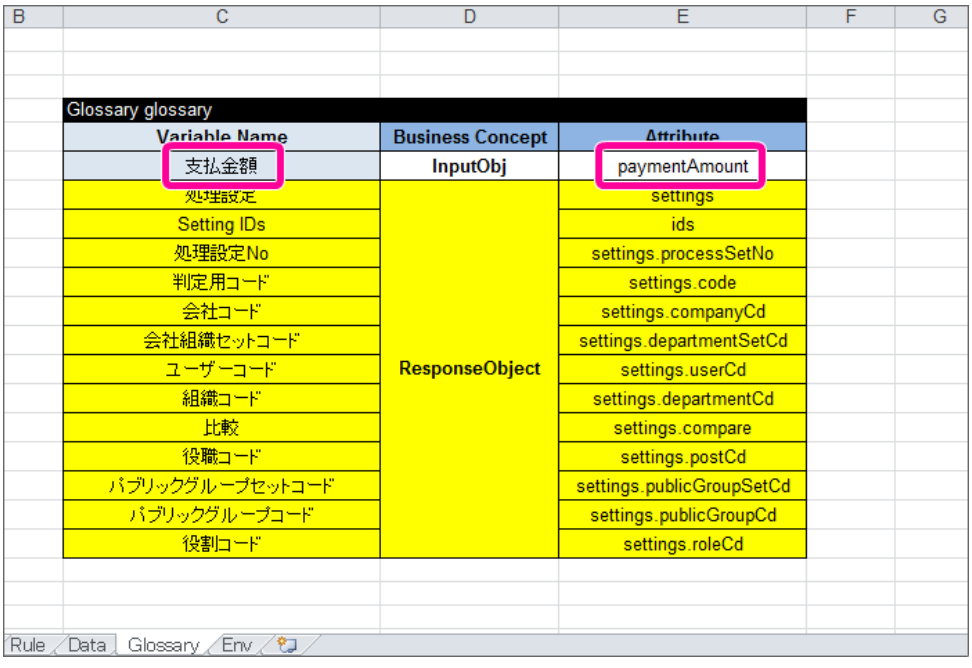
4. 続いて、支払金額に対応する物理名を設定するために、InputObjのDatatype、Variableに「paymentAmount」と設定してください。



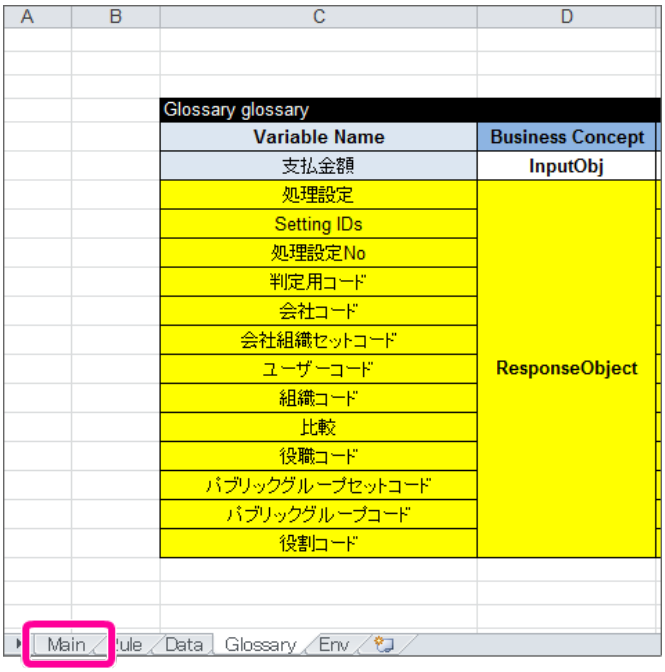
5. 「Glossary」シートを表示してください。



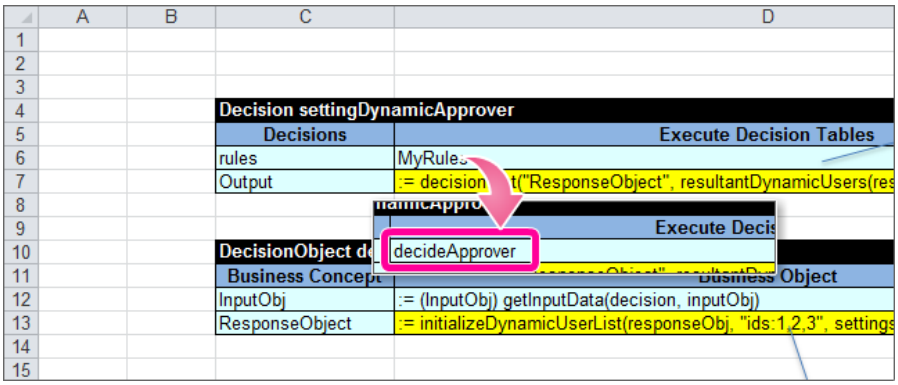
6. このシート中の *ResponseObject* も変更する必要はありません。
InputObjの項目の論理名・物理名を「支払金額」「paymentAmount」に変更してください。



7. 「Main」シートを表示してください。



8. 「Main」シートでは、実行する *Decision* 以外には、必要な設定が行われていますので、何も変更する必要はありません。
   Decision の「Decisions」列に「rules」と記載されている行の「Execute Decision Tables」の値を、先に設定した *DecisionTable* の名前「decideApprover」に変更してください。



9. これで、 OpenRules で処理対象者を決定するためのExcelファイルの設定が完了しましたので、ファイルを保存してください。
   IM-BIS のワークフローと連携するための設定を行っていきましょう。

IM-BIS のフローの横配置ノードに OpenRules を連携する

先の手順で作成したExcelのルール定義ファイルを IM-BIS と連携するためのフローの作成を進めていきましょう。

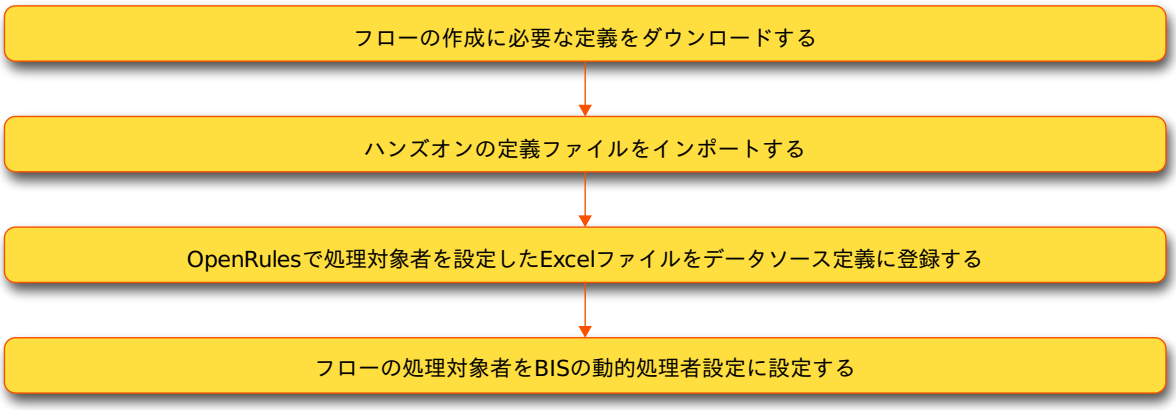
ルールと連携したフローを作成する手順

- OpenRules と IM-BIS を連携するための手順
- フローの作成に必要な定義をダウンロードする
- ハンズオンの定義ファイルをインポートする
- OpenRules で処理対象者を設定したExcelファイルをデータソース定義に登録する
- フローの処理対象者をBISの動的処理対象者設定に設定する

OpenRules と IM-BIS を連携するための手順

この手順では、作成したExcelのルール定義ファイルをデータソース定義に登録し、 IM-BIS の動的処理者設定に設定するまでの手順を確認していきます。

- BISの動的処理者設定に OpenRules を設定する手順



フローの作成に必要な定義をダウンロードする

ハンズオンで作成するフローのベースとなる各種定義ファイルをインポートします。
最初に下記のリンクからファイルをダウンロードしてください。
「 IM-Workflow 定義」のみダウンロード後に解凍してください。

- IM-Workflow 定義
  imw_request_for_approval.zip
- BIS定義
  bis_request_for_approval.zip
- Formaアプリケーション定義
  forma_request_for_approval.zip

ハンズオンの定義ファイルをインポートする

先の手順でダウンロードしたファイルを「 *各種定義ファイルのインポートの手順* 」に従ってインポートしてください。

OpenRules で処理対象者を設定したExcelファイルをデータソース定義に登録する

IM-BIS と連携するために、 OpenRules のルールを定義したExcelファイルをデータソース定義に登録していきましょう。

データソース定義の基本情報を登録する

データソース定義の基本情報を登録しましょう。

1. サイトマップの「 IM-BIS 」から「データソース定義」をクリックしてください。

2. 「登録」をクリックしてください。

3. 以下の通りに入力後、「登録」をクリックしてください。

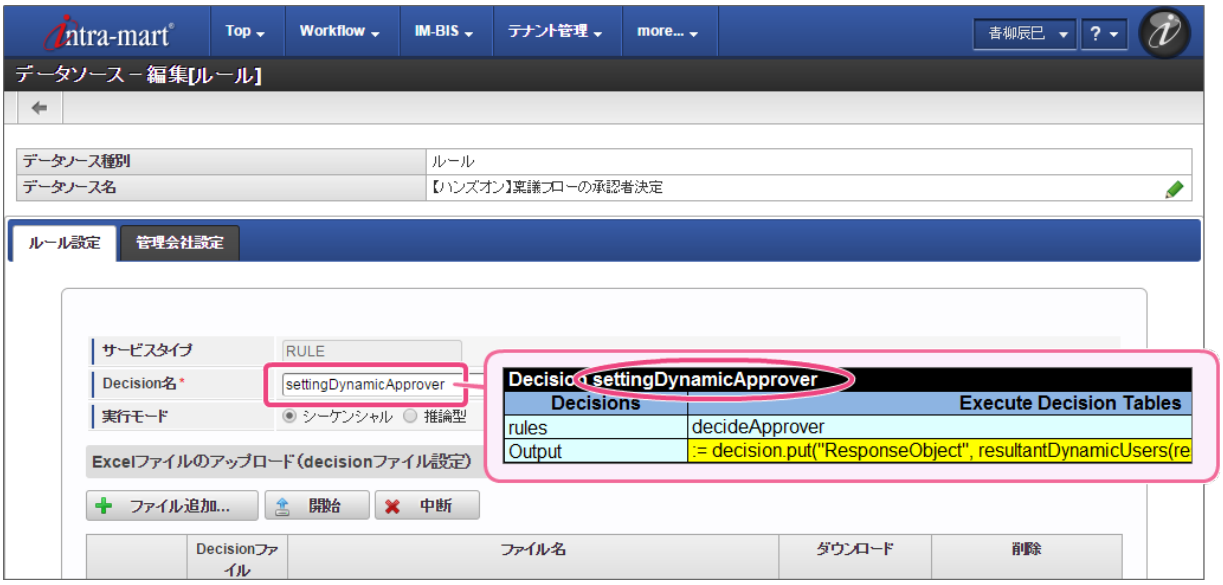
1. データソース種別
   「ルール」を選択してください。

2. データソース名
   「【ハンズオン】稟議フローの承認者決定」と入力してください。

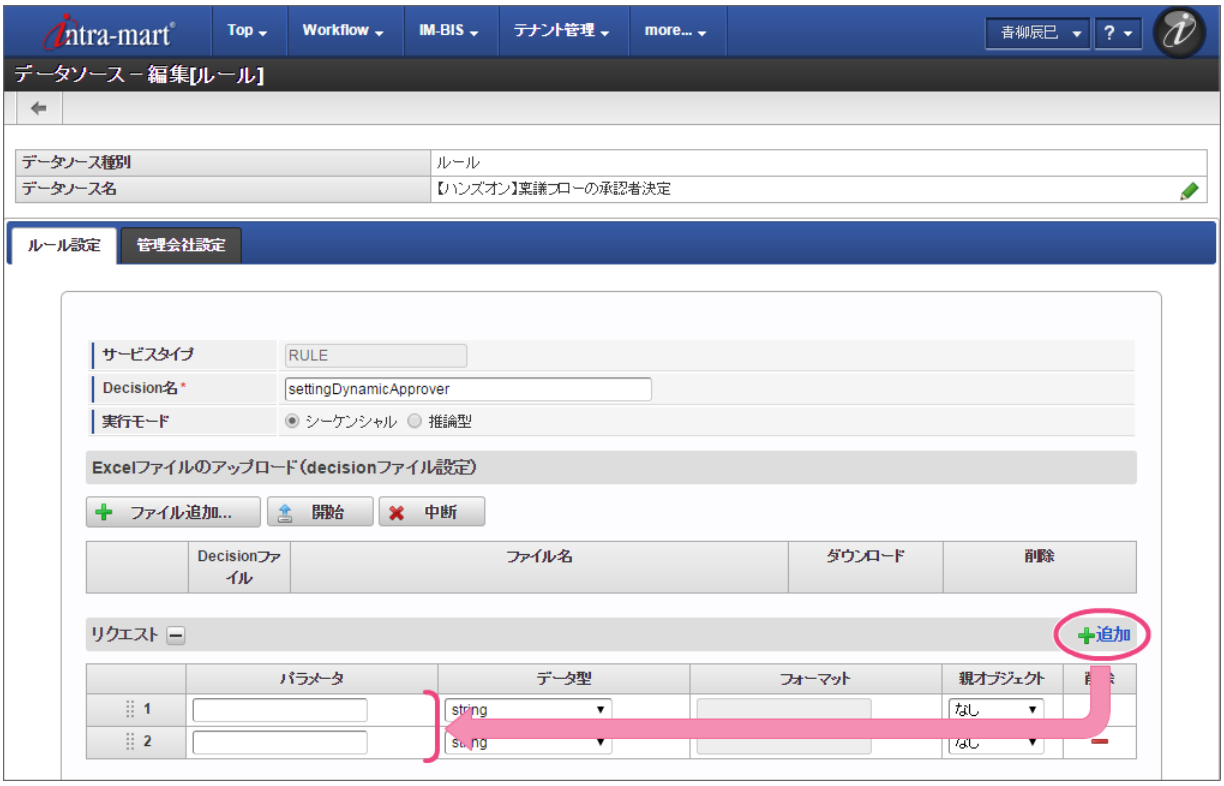
OpenRules の詳細情報を登録する

データソース定義に OpenRules のファイルやパラメータを設定しましょう。

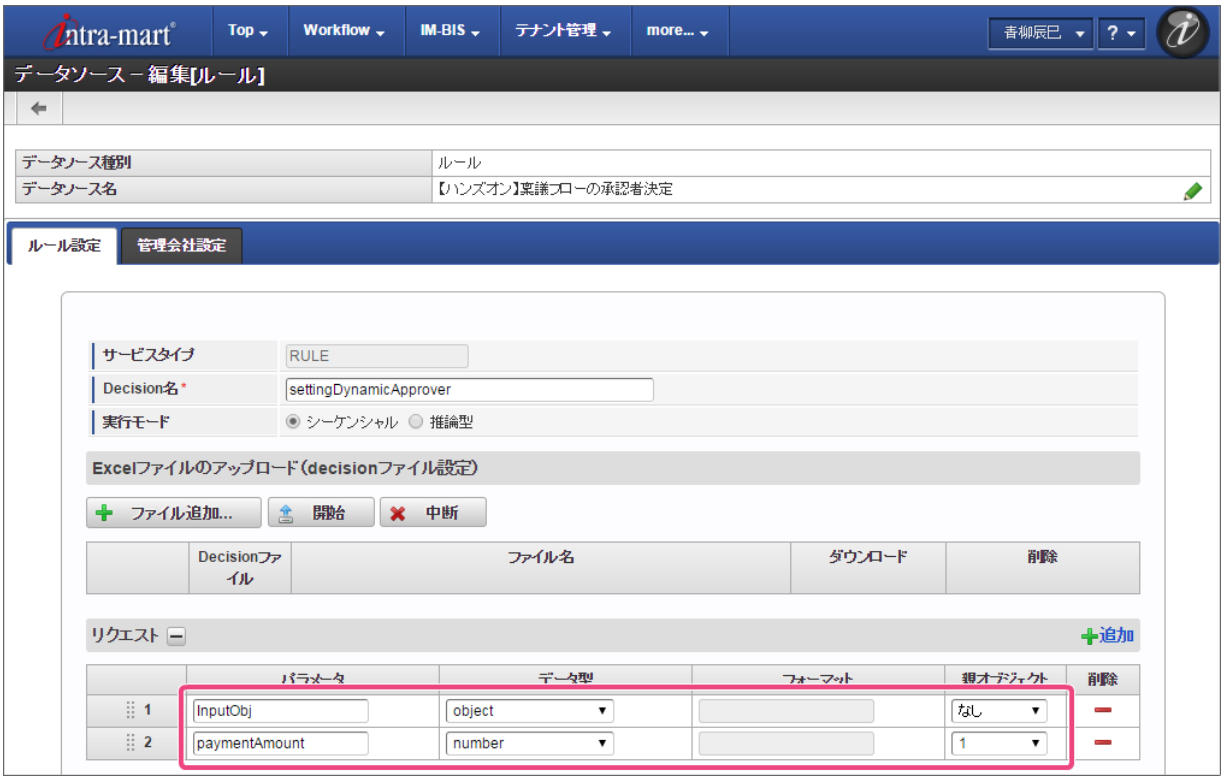
1. 「Decision名」には、Excelファイルの *Decision* の名前「settingDynamicApprover」を入力してください。



2. 「リクエスト」には、 *Glossary* で定義した「InputObject」のオブジェクトと項目（物理名）を登録します。
「＋追加」を2回クリックし、入力欄を2行追加してください。

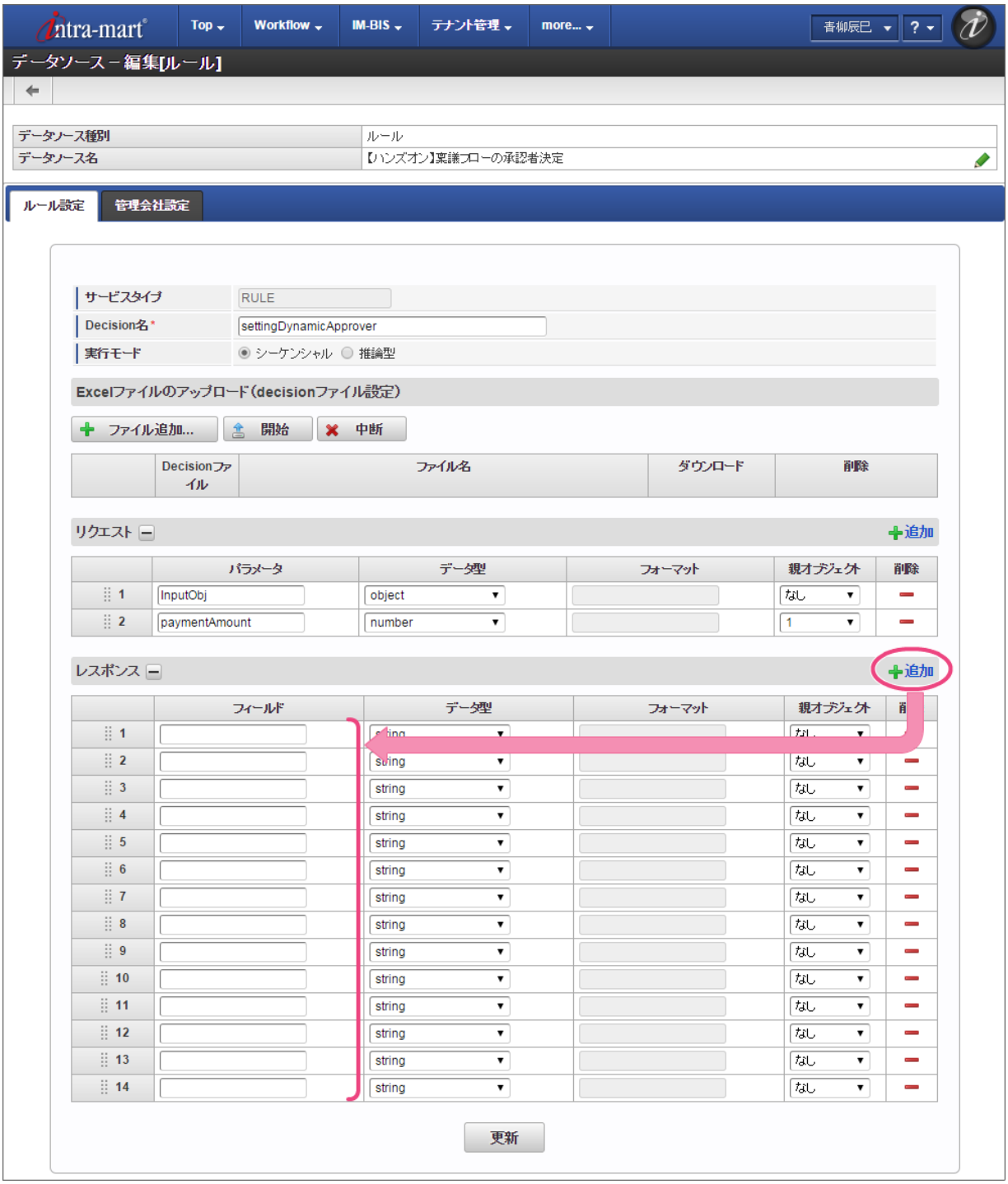


3. 追加したリクエストパラメータには以下の通りに設定してください。

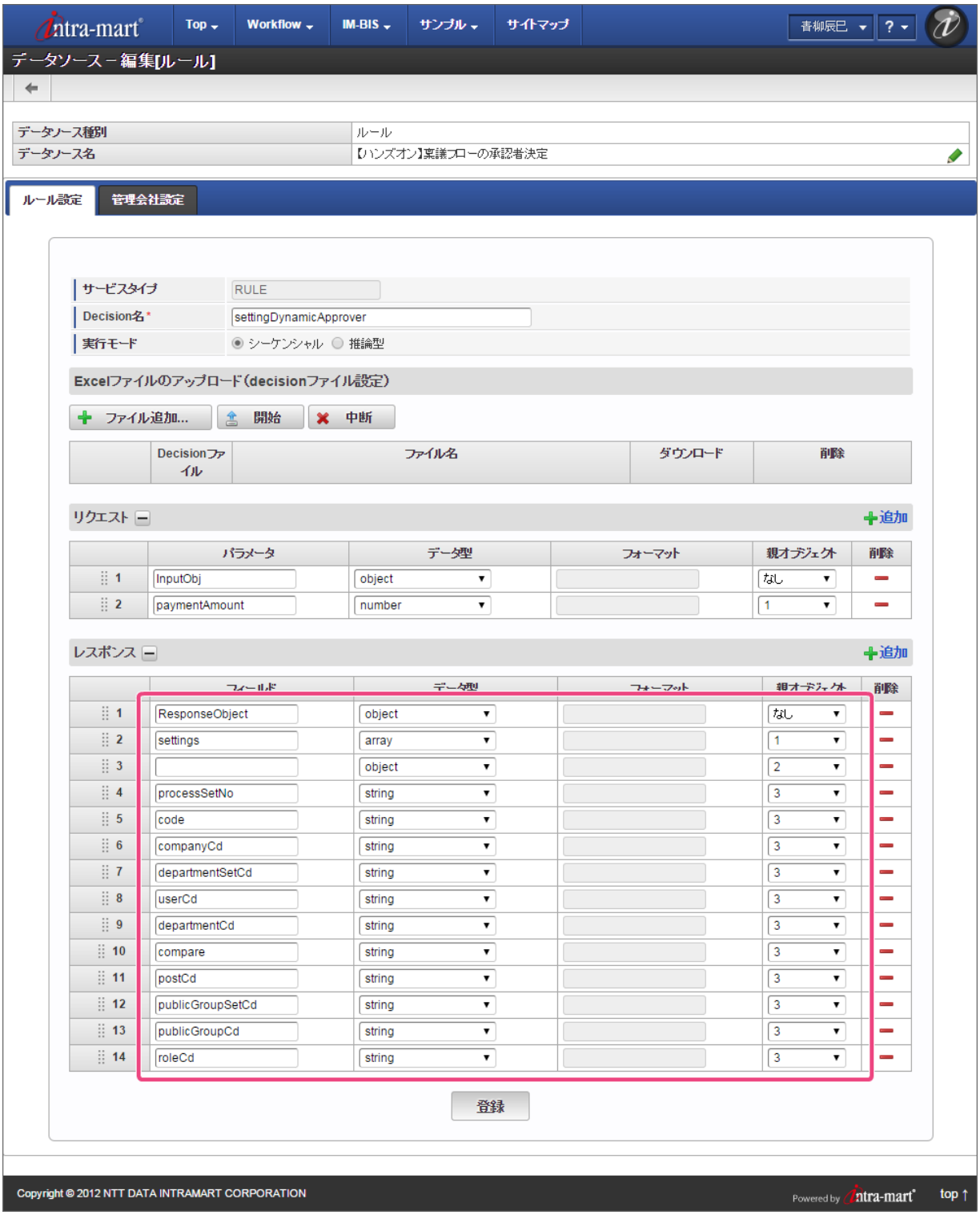


パラメータ	データ型	親オブジェクト
InputObj	object	なし
paymentAmount	number	1

4. 「レスポンス」には、 *Glossary* で定義した「 *ResponseObject* 」のオブジェクトと項目（物理名）を登録します。
「 追加」を14回クリックし、入力欄を14行追加してください。



5. 追加したレスポンスフィールドには以下の通りに設定してください。

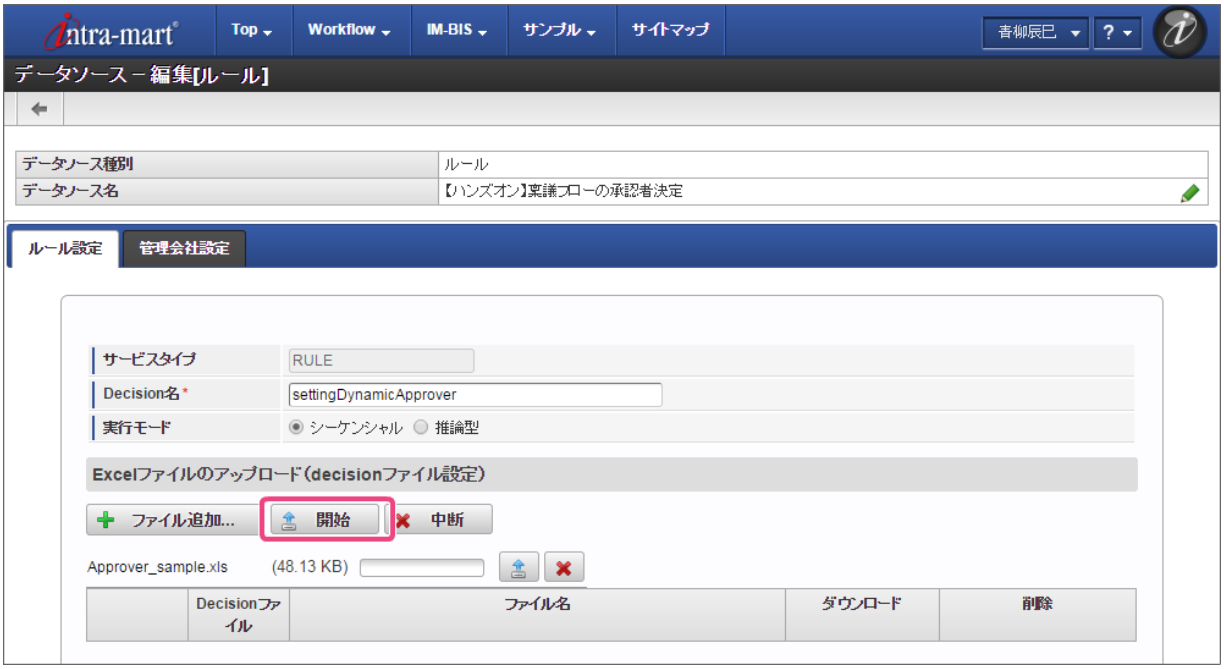


フィールド	データ型	親オブジェクト
ResponseObject	object	なし
settings	array	1
（空欄のまま）	object	2
processSetNo	string	3
code	string	3
companyCd	string	3
departmentSetCd	string	3
userCd	string	3
departmentCd	string	3
compare	string	3
postCd	string	3

フィールド	データ型	親オブジェクト
publicGroupSetCd	string	3
publicGroupCd	string	3
roleCd	string	3

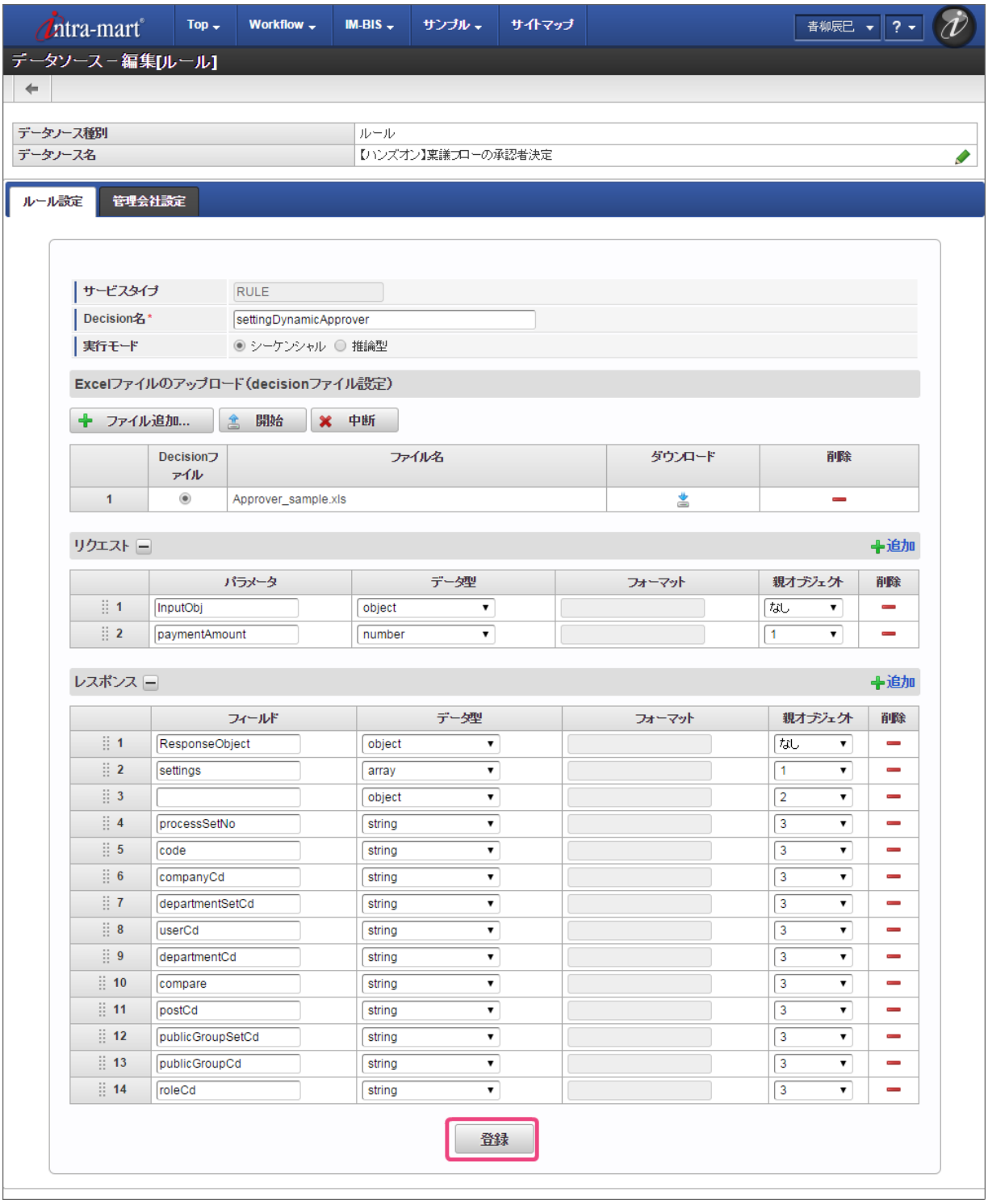
6. 作成したExcelのルール定義ファイルをアップロードするために「ファイル追加」をクリックしてください。

7. 「開始」をクリックして、ファイルをアップロードしてください。



8. 「Decisionファイル」のラジオボタンをクリックしてオンにしてください。

9. 最後に「登録」をクリックして、データソース定義を登録してください。



10. これで OpenRules のルールを定義したExcelファイルをデータソース定義として登録することができました。

フローの処理対象者をBISの動的処理対象者設定に設定する

登録したデータソース定義を利用して、 IM-BIS の横配置ノードの処理対象者を OpenRules で設定できるように進めていきしょう。

横配置ノードに動的処理者設定画面を表示する

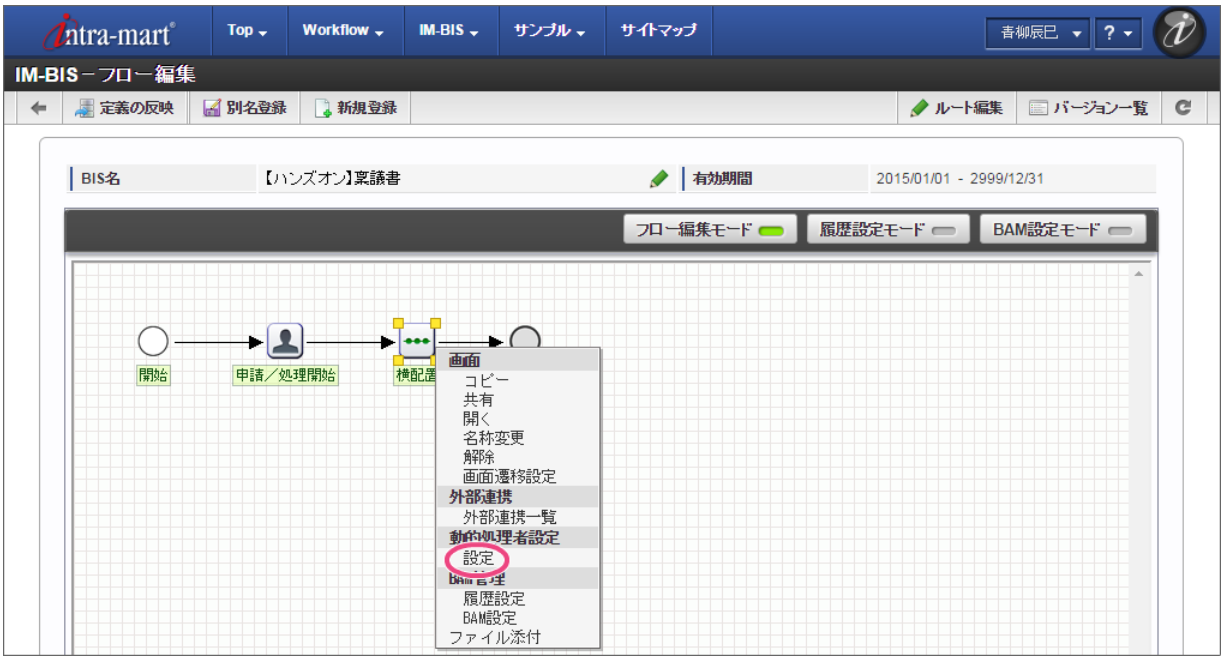
横配置ノードの動的処理者設定画面を表示して、設定を開始しましょう。

1. サイトマップの「 IM-BIS 」をクリックしてください。

2. 「一覧」をクリックしてください。

3. インポートしたフローの「【ハンズオン】稟議書」の をクリックしてください。

4. 「横配置」を右クリック後、コンテキストメニューから「動的処理者設定」の「設定」をクリックしてください。



5. この横配置ノードの処理対象者を OpenRules の結果に基づいて設定するために、「動的処理者設定」を「外部連携設定」に変更してください。

6. 「動的処理者設定」を外部連携で実行する設定ができました。
   引き続き、次の手順で OpenRules との連携情報を設定しましょう。

動的処理者設定で OpenRules との連携情報を設定する

動的処理者設定で、自動的に処理対象者（承認者）を決定するための連携情報を設定しましょう。

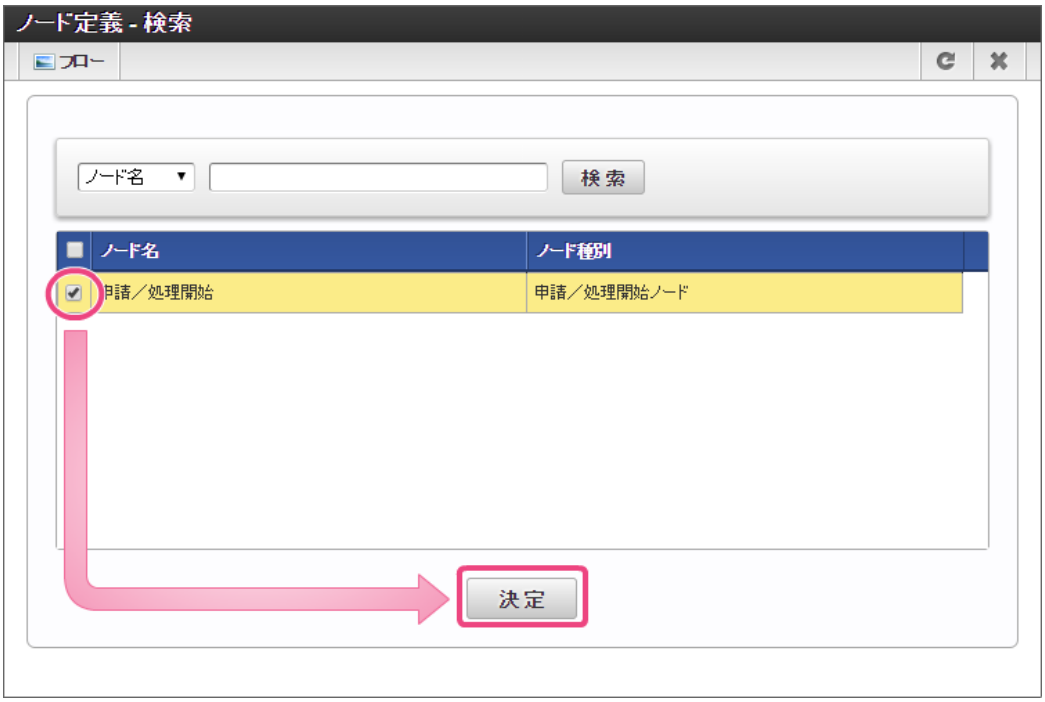
1. ルールの結果に基づいて自動的に承認者を決定するために「動的処理者設定」の「処理対象者の設定方法」を「自動で設定する」に設定してください。

2. 自動的に設定された承認者を申請者が変更できないようにするために「処理対象者の変更」を「不可」に設定してください。

> コラム
> この画面の設定内容の詳細については、「 IM-BIS 業務管理者操作ガイド 」の「 動的処理対象者設定 」を参照してください。

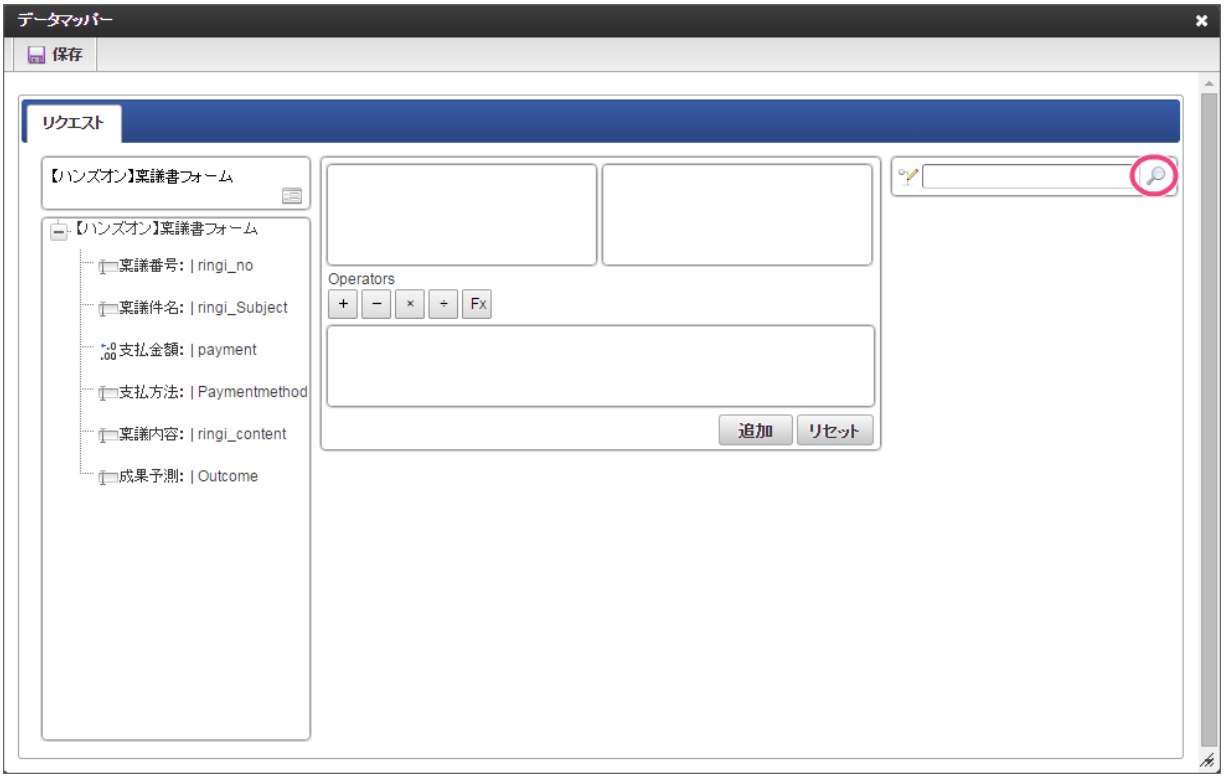
3. この横配置ノードの処理対象者を決定するノードとして、「処理対象者判定ノード」には「申請／処理開始」ノードを設定します。
「処理対象者判定ノード」の「 追加」をクリックしてください。

4. 「申請／処理開始」ノードのチェックボックスをオンにし、「決定」をクリックしてください。



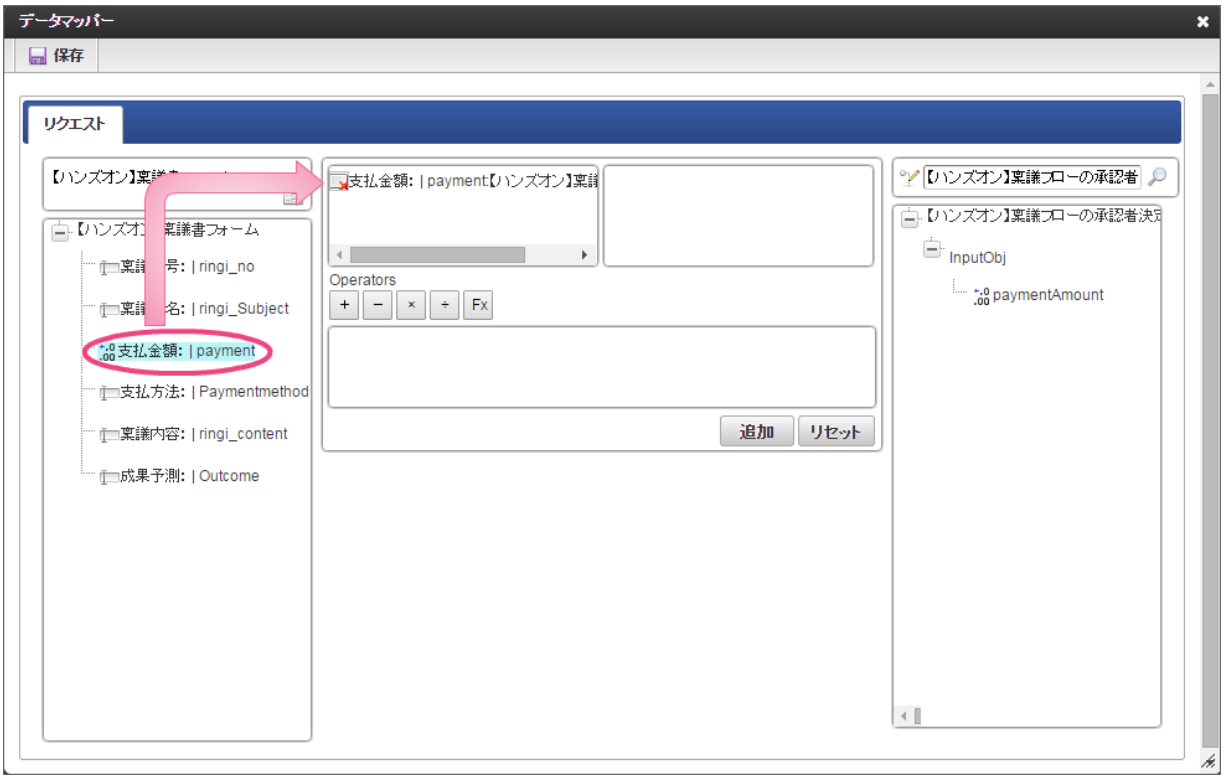
5. OpenRules と画面の入力項目のマッピングを設定するために、「外部連携」の をクリックしてください。

6. 右の欄から「 」をクリックしてください。

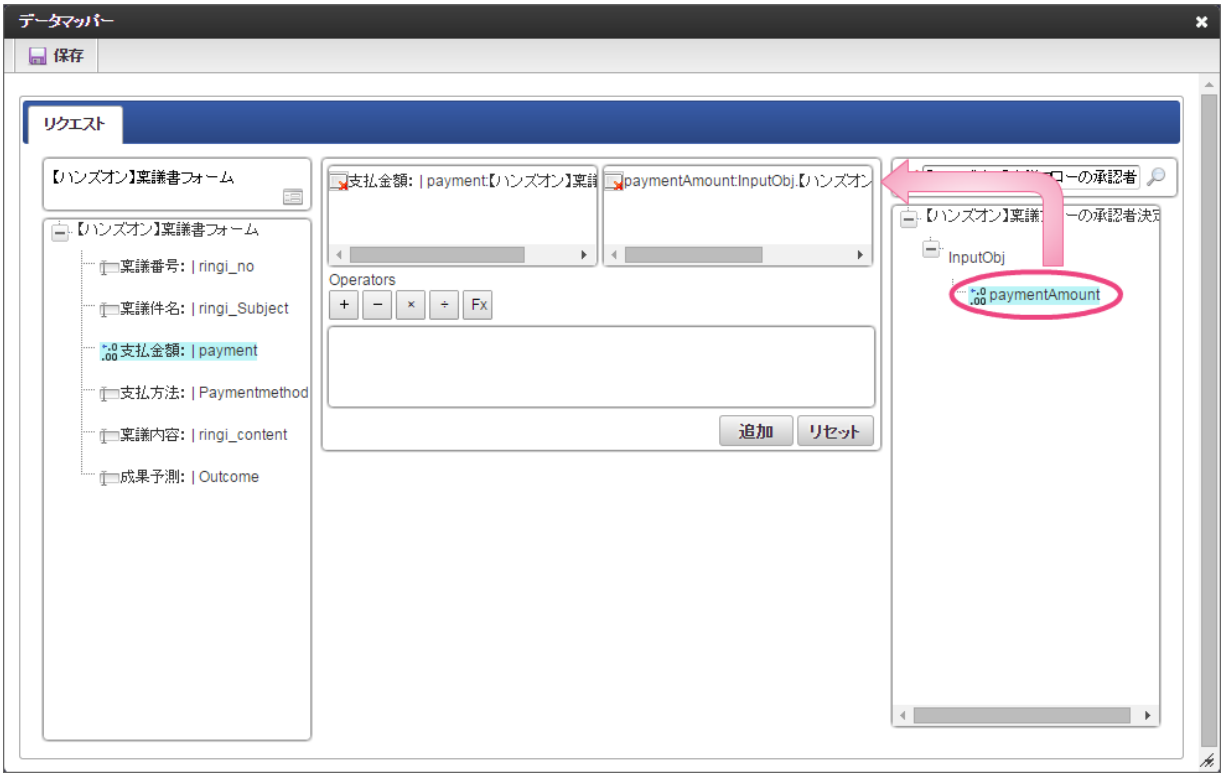


7. データソース選択から作成したルール「【ハンズオン】稟議フローの承認者決定」をクリックしてください。

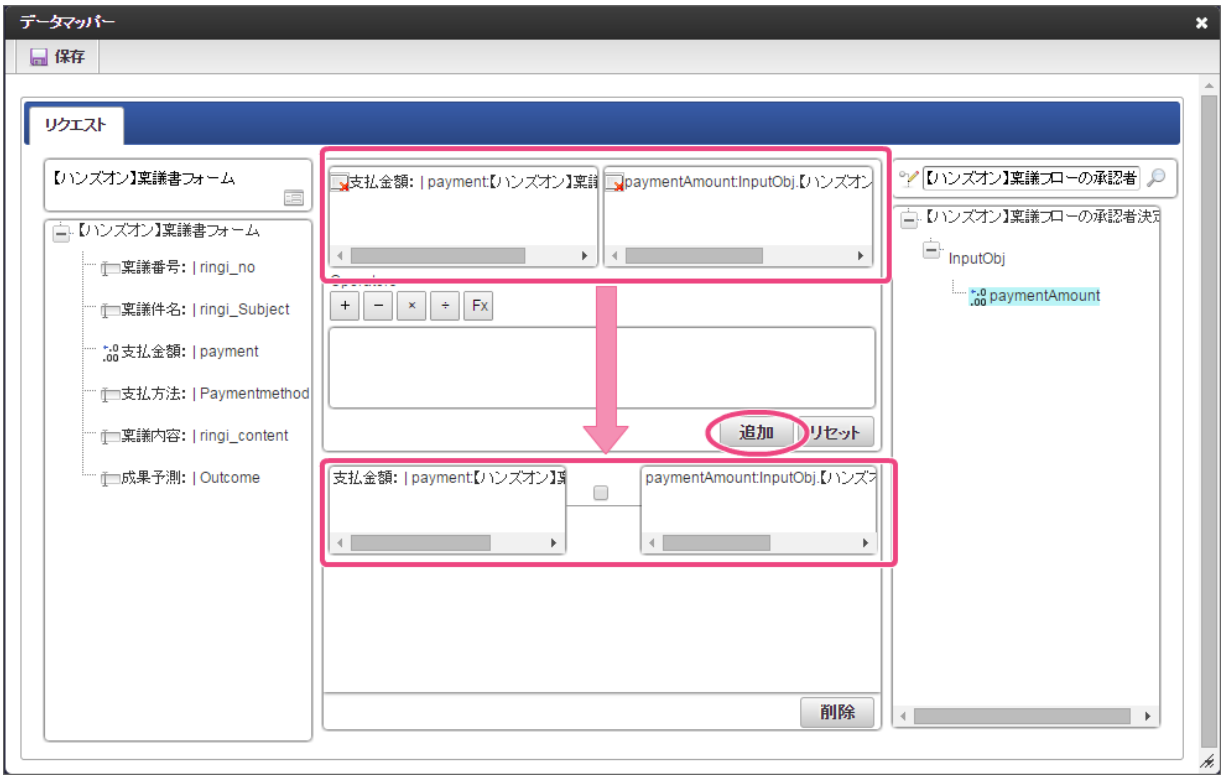
8. 左の欄から「支払金額」をクリックしてください。
   クリック後、中央左の欄に「支払金額: | payment【ハンズオン】稟議書フォーム」と表示されます。



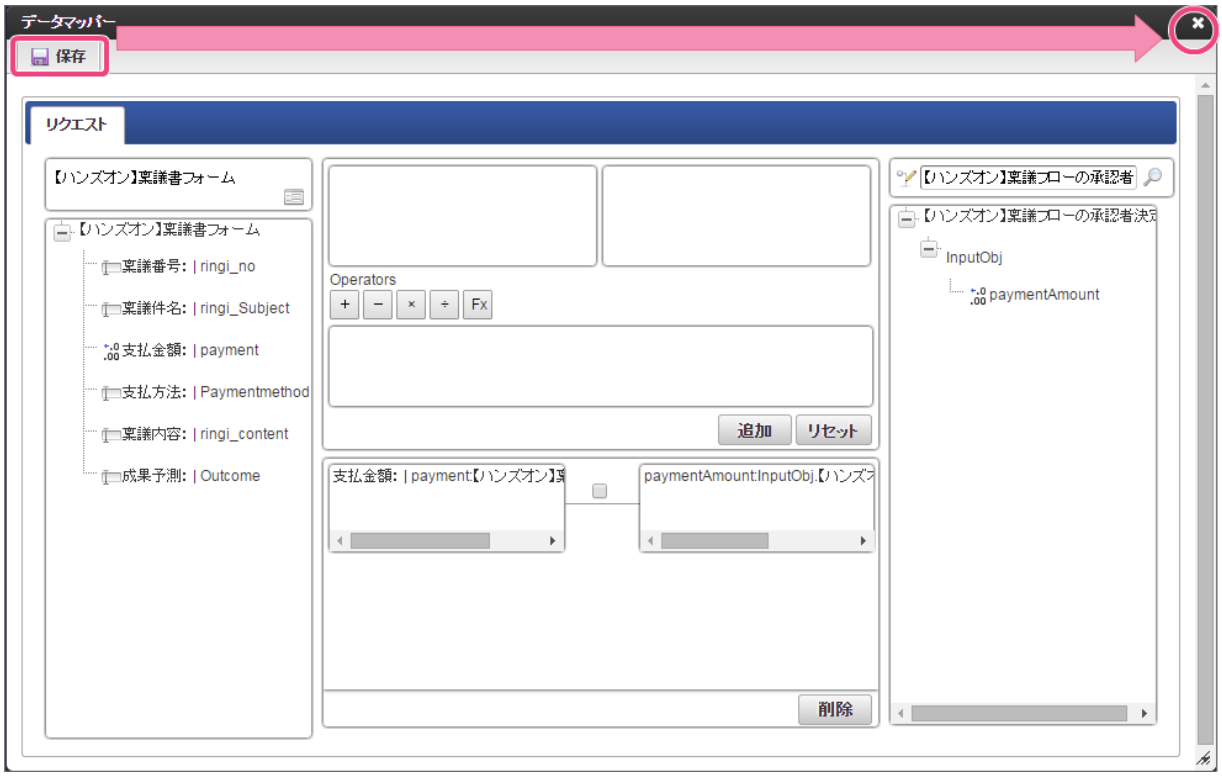
9. 右の欄から「paymentAmount」をクリックしてください。
クリック後、中央右の欄に「paymentAmount:InputObj.【ハンズオン】稟議フローの承認者決定」と表示されます。



10. 「追加」をクリックして、マッピングを追加してください。



11. IM-BIS の動的処理者設定では、レスポンスのマッピングは自動的に行われますので、「保存」をクリックしてください。
正常に保存できたら、「データマッパー」は右上の「 ✖ 」をクリックして閉じてください。



12. 動的処理者設定で「保存」をクリックしてください。

13. 最後に「定義の反映」をクリックして、フローの実行準備を行ってください。

14. これで、 OpenRules の結果に基づいて処理対象者が決定するワークフローを作成することができました。
次の手順で、実行して承認者がどのように設定されるのかを確認してみましょう。

作成したルールを実行する

これまでのシナリオで作成した IM-BIS のフローを使って、ルールを実行し、承認者がどのように設定されるのかを確認してみましょう。

ルールと連携したフローを実行する手順

- フローを実行するための事前準備
- 稟議書のワークフローを申請する
- 青柳辰巳の承認
- 部長（円山益男）の承認
- 社長（原田浩二）の承認

フローを実行するための事前準備

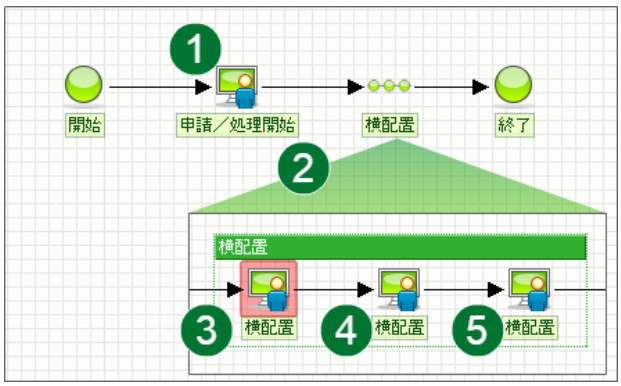
フローを実行（申請・承認）するために、対象のユーザに「BIS担当者」ロールを付与してください。
本書では、以下のように設定します。

- 申請者：林政義（ユーザコード：hayashi）
- 承認者1：青柳辰巳（ユーザコード：aoyagi）
- 承認者2：円山益男（ユーザコード：maruyama）→役職の「部長」
- 承認者3：原田浩二（ユーザコード：harada）→役職の「社長」

> **コラム**
> 今回のハンズオンのフローの「申請／処理開始」ノードの処理対象者は、「BIS担当者」を設定しておりますので、該当のロールを持ったユーザであれば申請できます。

今回は、横配置ノードが OpenRules での結果に基づいて、3つのノードに展開される形で実行される流れを以下の図のように確認します。

実行手順



1. BIS担当者（林）が申請します。
2. 申請内容に基づいて、横配置ノードを展開されます。
   ハンズオンでは3つのノードに展開されます。
3. 横配置ノードの1番目のノードには、ユーザ：青柳辰巳が設定されます。
4. 横配置ノードの2番目のノードには、役職：部長が設定されます。
5. 横配置ノードの3番目のノードには、役職：社長が設定されます。

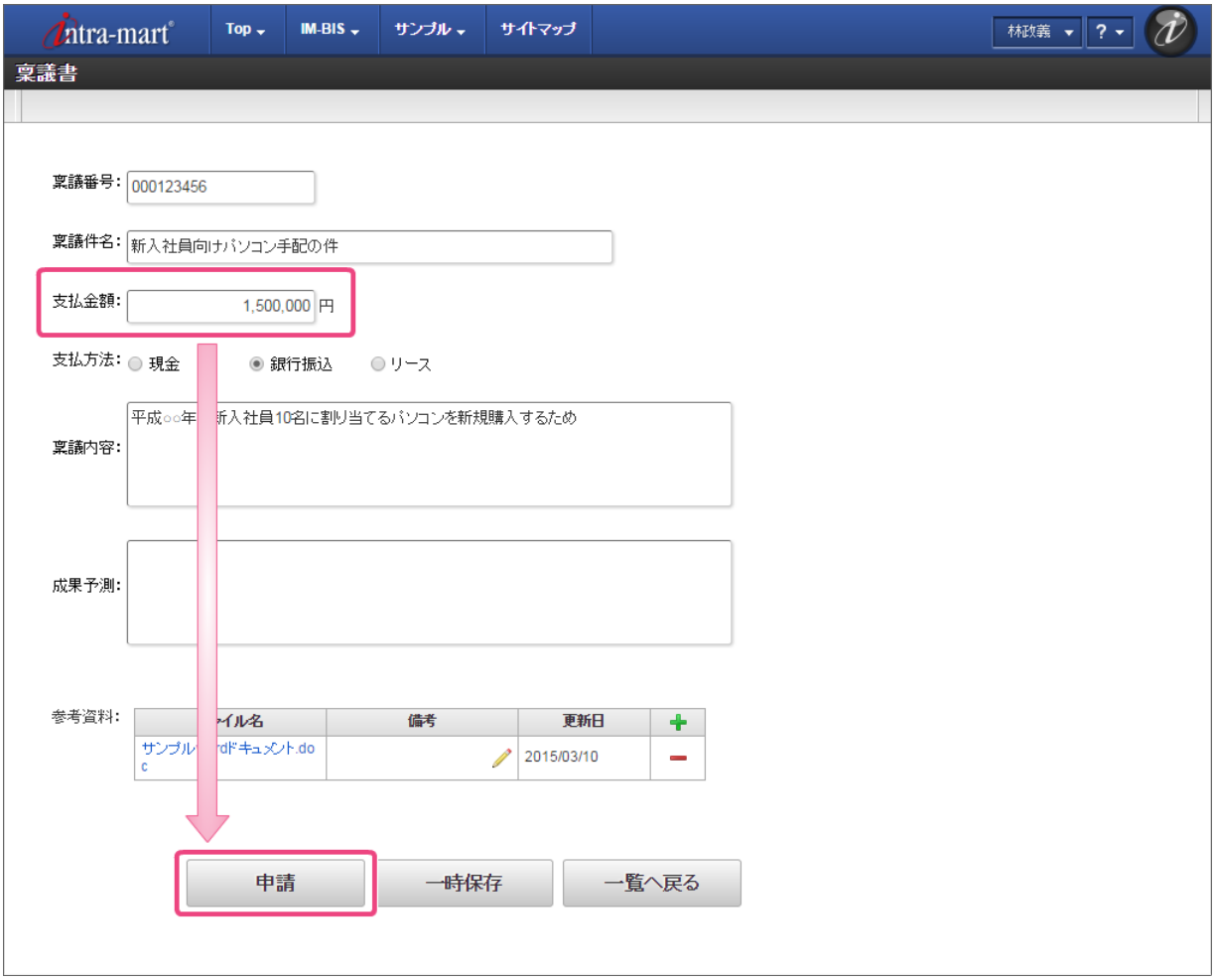
稟議書のワークフローを申請する

1. 申請者のユーザ（例では、「林政義」）でログインしましょう。
2. 上部のメニューの「 IM-BIS 」→「ワークフロー」の順にマウスを重ねてから「申請」をクリックしてください。

3. 「【ハンズオン】稟議書」の「申請／処理開始」をクリックしてください。

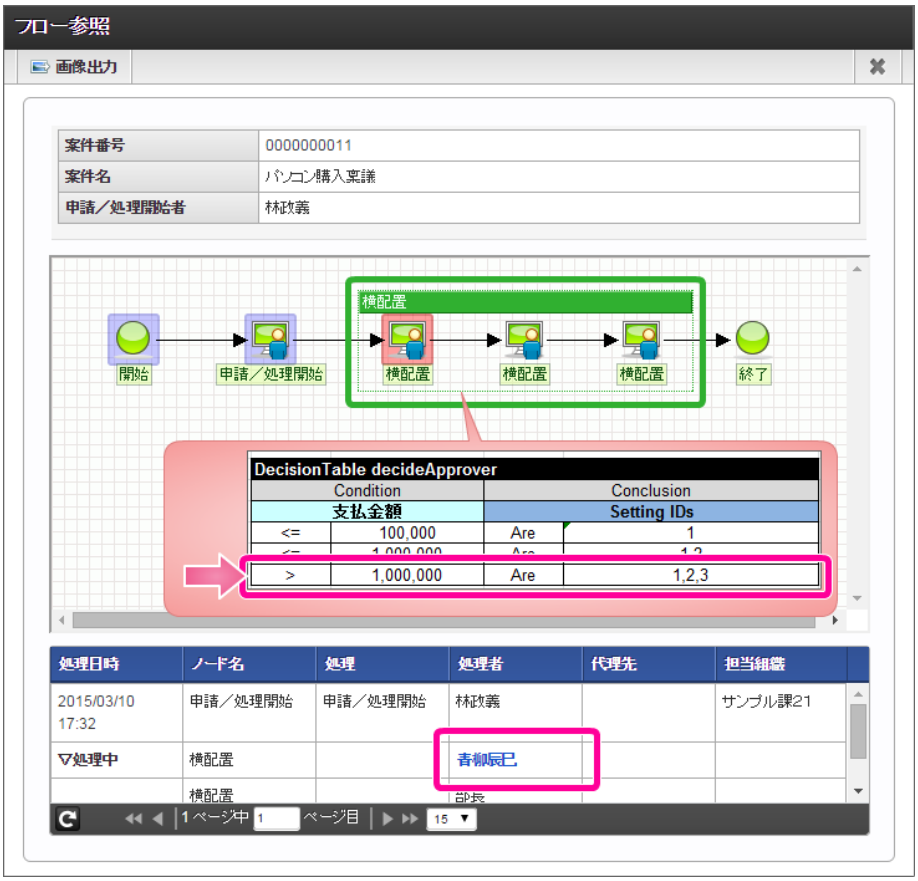
4. 画面の各項目の内容を入力します。
   今回は、次の処理対象者を社長まで対象とするルートにするために、支払金額には「1,500,000」と入力しましょう。

必要な項目の入力が終わったら、「申請」をクリックしてください。



5. 案件名を入力したら、「申請／処理開始」をクリックしてください。
   これで、林さんの申請は完了です。

6. この時点のフローを確認すると、申請画面に入力した支払金額に基づいて、横配置ノードが3つのノードとして展開されています。
   展開されたノードのうち、1番目のノードの処理対象者の青柳さんに切り替えて操作しましょう。

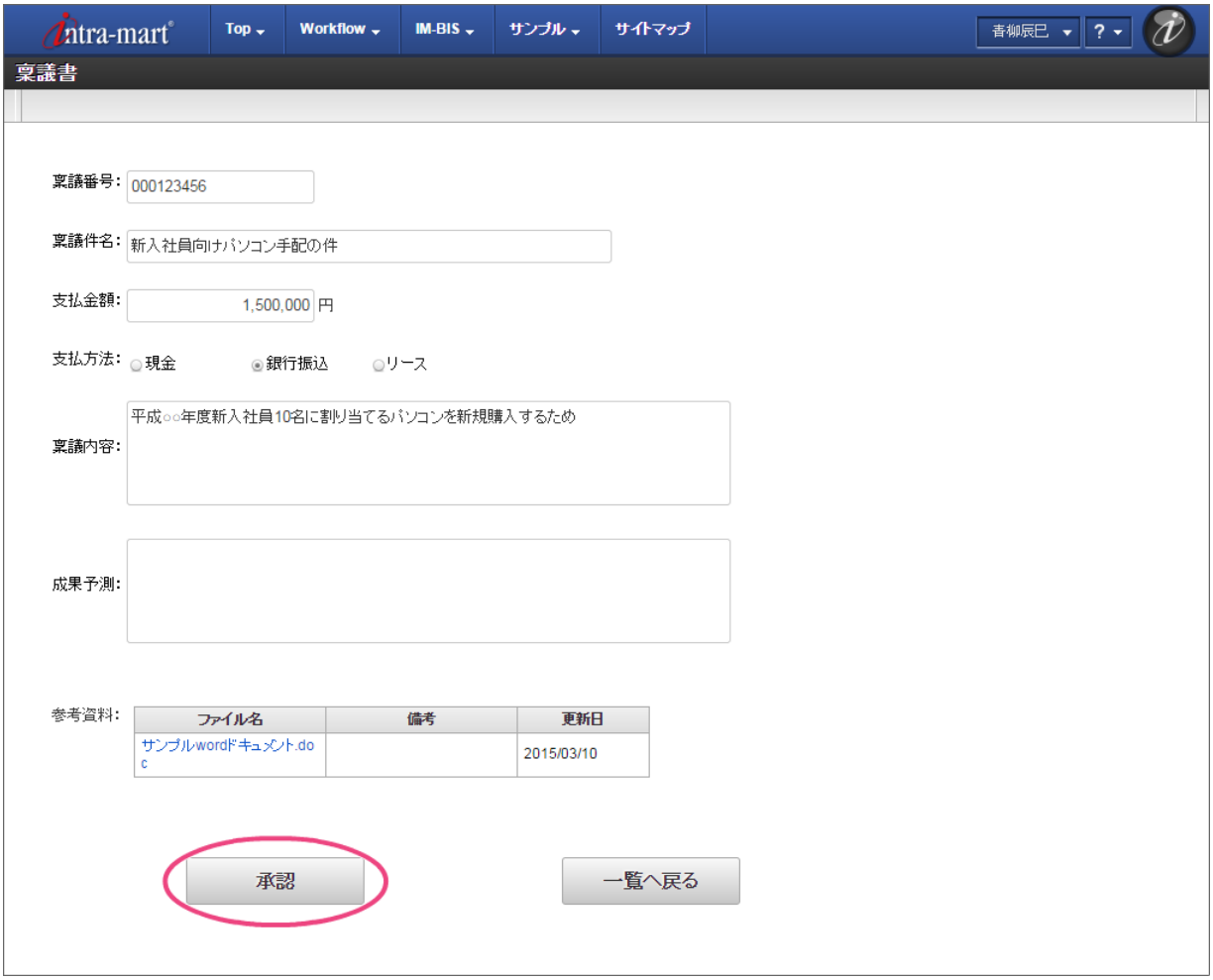


青柳辰巳の承認

1. OpenRules での結果に基づいて、次の処理対象者に設定された「青柳辰巳」でログインしましょう。
2. 上部のメニューの「 IM-BIS 」→「ワークフロー」の順にマウスを重ねてから「未処理」をクリックしてください。

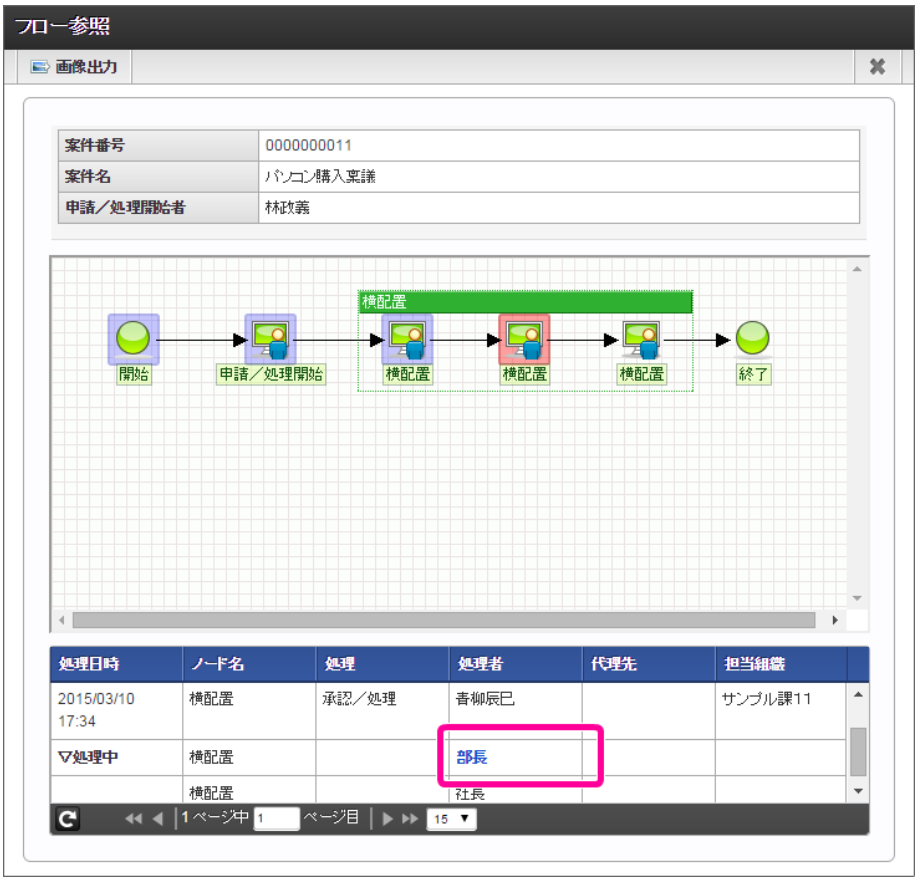
3. 林さんが申請した案件の「処理」をクリックしてください。

4. 内容を確認し、「承認」をクリックしてください。



5. 「承認／処理」をクリックしてください。
   これで青柳さんの承認が完了します。

6. この時点のフローを確認すると、以下のように表示されます。
   次の処理対象者の部長の円山さんに切り替えて操作しましょう。

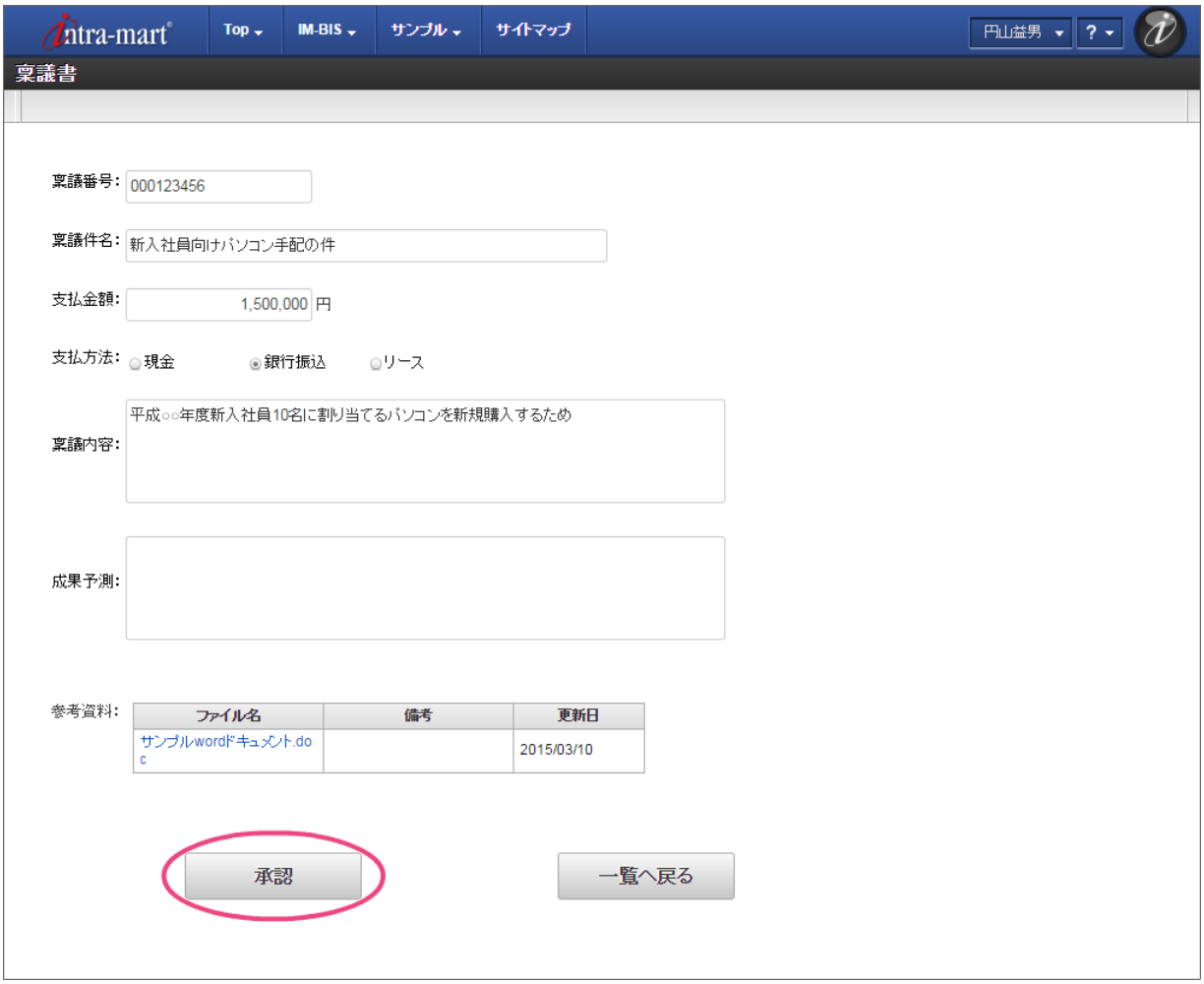


部長（円山益男）の承認

1. OpenRules での結果に基づいて、次の処理対象者に設定された役職の部長の「円山益男」でログインしましょう。
2. 上部のメニューの「 IM-BIS 」→「ワークフロー」の順にマウスを重ねてから「未処理」をクリックしてください。

3. 林さんが申請した案件の「処理」をクリックしてください。

4. 内容を確認し、「承認」をクリックしてください。



5. 「承認／処理」をクリックしてください。
   これで部長の円山さんの承認が完了します。

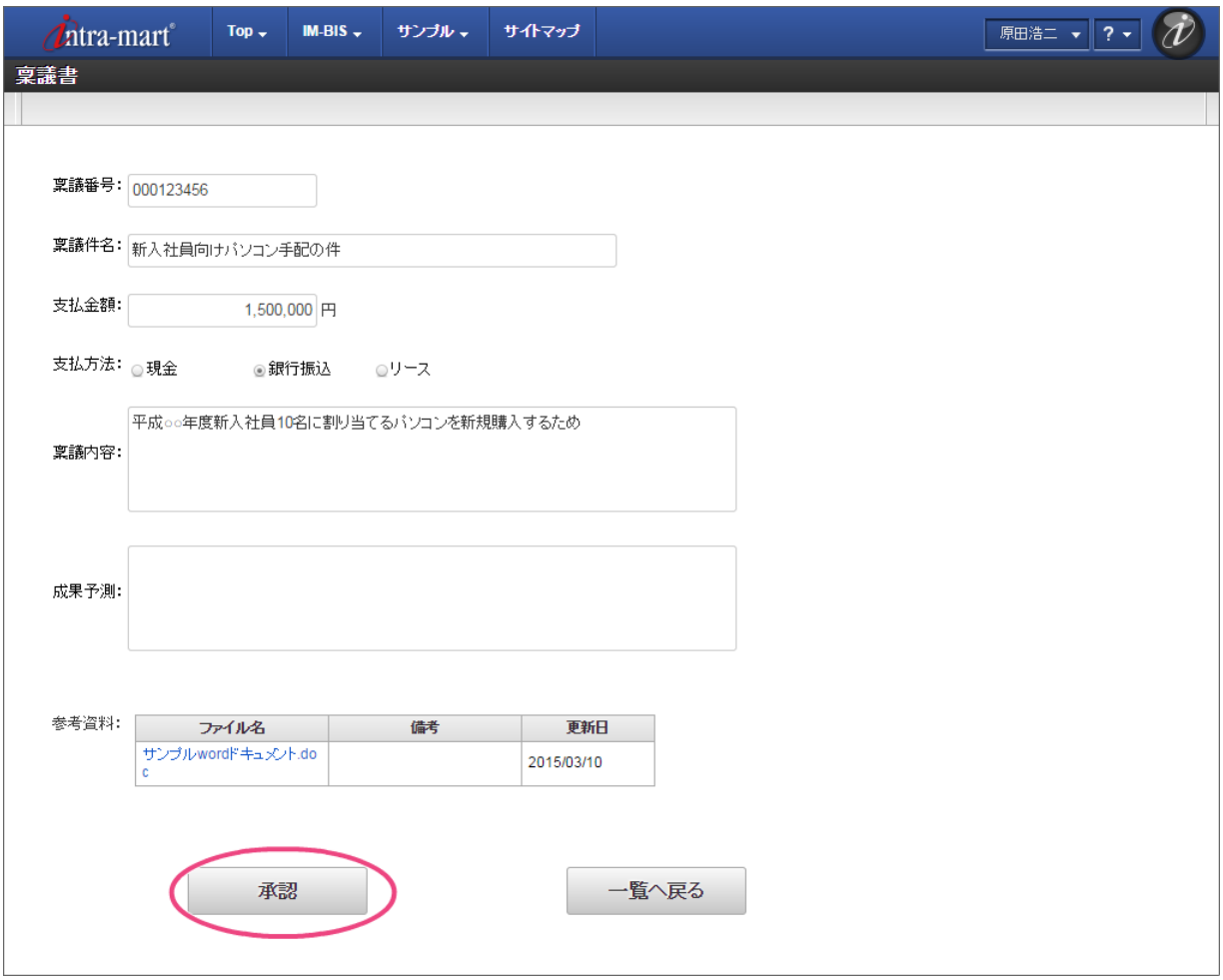
6. この時点のフローを確認すると、以下のように表示されます。
   次の処理対象者の社長の原田さんに切り替えて操作しましょう。

社長（原田浩二）の承認

1. OpenRules での結果に基づいて、最後の処理対象者に設定された役職の社長の「原田浩二」でログインしましょう。
2. 上部のメニューの「 IM-BIS 」→「ワークフロー」の順にマウスを重ねてから「未処理」をクリックしてください。

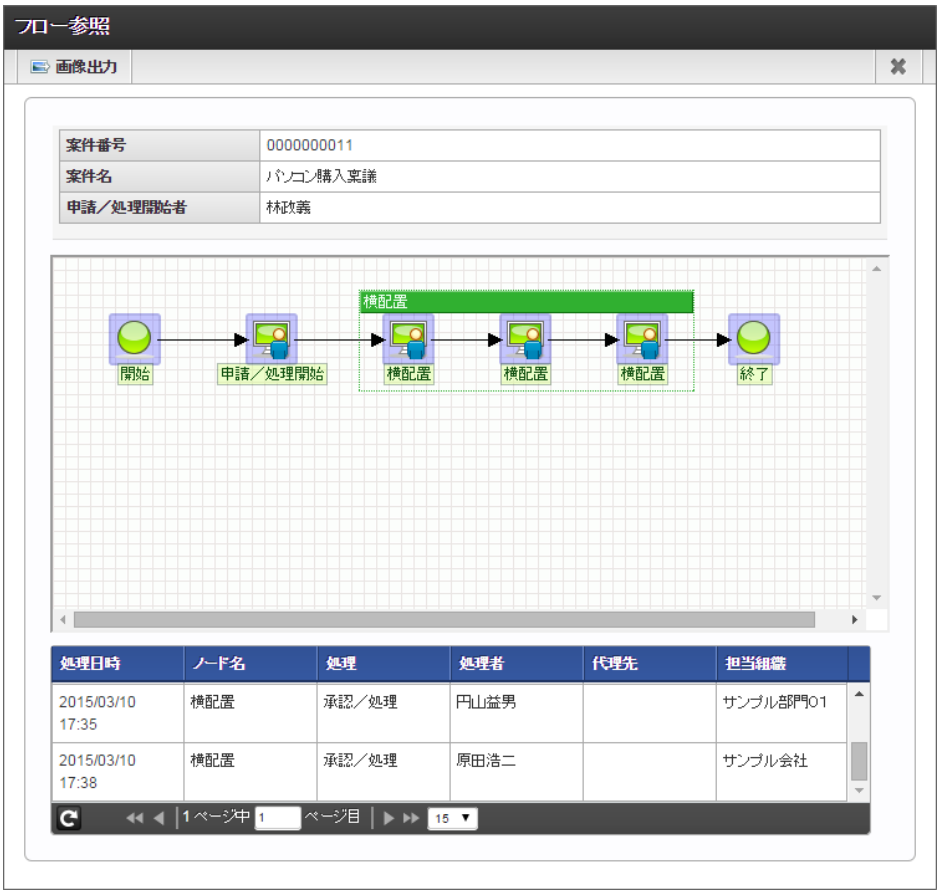
3. 林さんが申請した案件の「処理」をクリックしてください。

4. 内容を確認し、「承認」をクリックしてください。



5. 「承認／処理」をクリックすると、原田さんの承認が完了します。
原田さんがこのフローの最終処理対象者となっているため、これで案件が完了します。

6. 完了時点のフローを確認すると、以下のように表示されます。



7. 今回は、3つの処理対象者が設定されるパターンを確認できましたが、支払金額を変更すると処理対象者が OpenRules の条件で変わります。
支払金額を変更して他のパターンも確認してみてください。

OpenRules for IM-BIS 連携の運用

OpenRules for IM-BIS 連携開発の運用に関する必要事項をまとめています。

データソース定義の登録・更新でエラーが発生する

OpenRules をデータソース定義に登録する際に発生するエラーをまとめています。

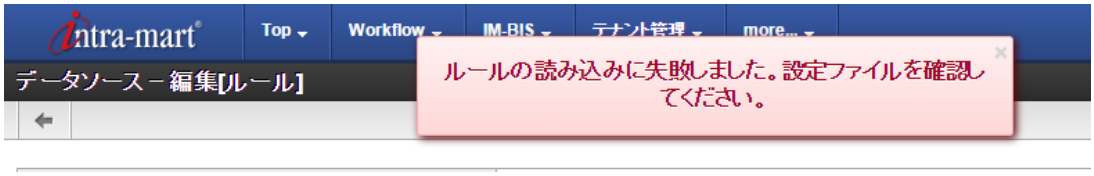
データソース定義で発生するエラー
- OpenRules 内の項目やキーワードの綴りが誤っている
- *DecisionObject* が利用する変数名が誤っている
- *DecisionTable* の演算子を全角で定義している

OpenRules 内の項目やキーワードの綴りが誤っている

現象

データソース定義にExcelファイルをアップロードし、「登録」または「更新」をクリックしたときに下記のエラーメッセージが表示される。

- 「ルールの読み込みに失敗しました。設定ファイルを確認してください。」



条件

- bis.logやコンソールに下記のようなスタックトレースが出力されている。

```
[ERROR] BIS_LOG - [] ルールの読み込みに失敗しました。設定ファイルを確認してください。
Error: Function label Conclution is not found in template table : java.lang.Exception
at file:/C:/resin/resin-pro-4.0.40_forma/webapps/imart/WEB-INF/im_bis/datasource/rule/5iemi011wqmvepd/xxxxx.xls?
sheet=DecisionTable&range=B15:K18&openl=
java.lang.Exception: Function label Conclution is not found in template table
```

原因

Excelファイル内に下記の定義誤りが存在する場合に発生します。

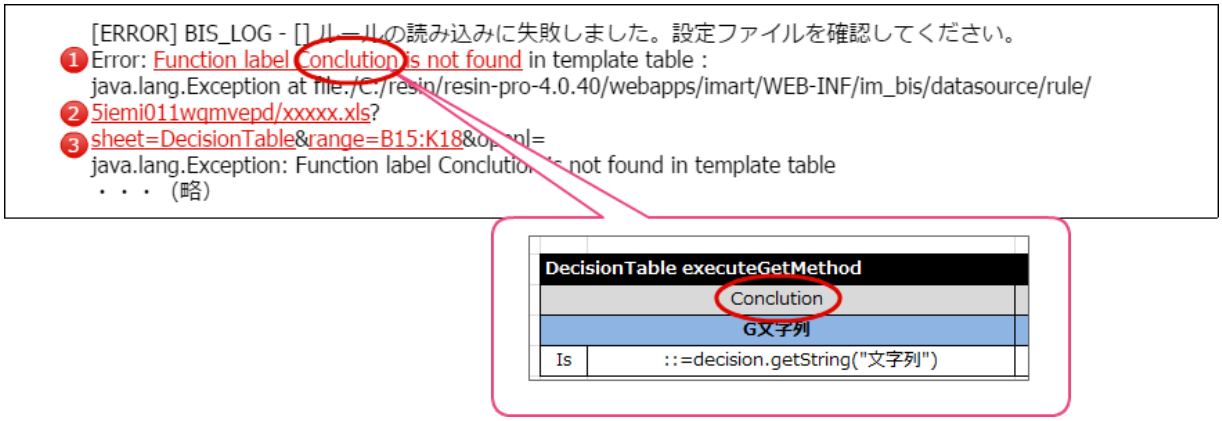
- OpenRules のキーワード（*Condition* など）の綴りが間違っている。

解決方法

スタックトレース中に、Excelファイルの誤っている可能性のあるセルが書いてありますので、そのセルの内容と関連する定義を確認してください。

上のスタックトレースの場合の読み方は、次の通りです。
下記の読み方に基づいて、Excelの定義ファイルを修正し、再度データソース定義にアップロードしてください。

- *DecisionTable* に定義する列タイプを「*Conclusion* 」とすべきところ、綴りミスで「Conclution」と記述した例



1 Function label <キーワード・関数名> is not found

- Function label ・・・ 誤りの箇所が OpenRules のキーワードや *Method* で定義した関数名を記述するセルである可能性を表します。
- is not found in template table・・・ルールの実行時に参照した OpenRules の製品の定義ファイルやデータソース定義にて納したExcelの定義ファイルに定義が存在しないことを表します。

2 <データソース定義ID>/<対象のデータソース定義で参照しているExcelファイル名>.xls

- Excelファイル名・・・データソース定義のどのExcelの定義ファイルにエラーが発生しているかを表します。

3 sheet=<シート名>&range=<セル範囲>

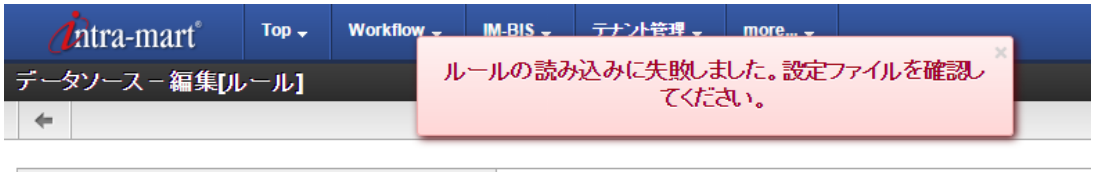
- シート名・・・Excelの定義ファイル内の対象のシート名です。
- セル範囲・・・Excelの定義ファイル内の対象のシートの対象のセル範囲です。

DecisionObject が利用する変数名が誤っている

現象

データソース定義にExcelファイルをアップロードし、「登録」または「更新」をクリックしたときに下記のエラーメッセージが表示される。

- 「ルールの読み込みに失敗しました。設定ファイルを確認してください。」



条件

- bis.logやコンソールに下記のようなスタックトレースが出力されている。

```
[2015-03-23 22:02:41.241] ERROR - [resin-port-8080-95] - [default] - [jp.co.intra_mart.system.bis.soa.connector.rule.RuleEngineContainer]
ルールの読み込みに失敗しました。設定ファイルを確認してください。
Error: Field not found: requestObject
Invalid Code Fragment:
=========================================
 (RequestObject) getInputData(decision, requestObject)
                              ^^^^^^^^^^^^^^
=========================================
   at file:/C:/resin/resin-pro-4.0.40_forma/webapps/forma40_bis/WEB-INF/im_bis/datasource/rule/5iembl44yljdcpd/xxxxx.xls?
sheet=Main&cell=E17&start=43&end=55&openl=


org.openl.syntax.SyntaxErrorException: Error: Field not found: requestObject
Invalid Code Fragment:
=========================================
 (RequestObject) getInputData(decision, requestObject)
                              ^^^^^^^^^^^^^^
=========================================
   at file:/C:/resin/resin-pro-4.0.40_forma/webapps/forma40_bis/WEB-INF/im_bis/datasource/rule/5iembl44yljdcpd/xxxxx.xls?
sheet=Main&cell=E17&start=43&end=55&openl=
```

原因

Excelファイル内に下記の定義誤りが存在する場合に発生します。

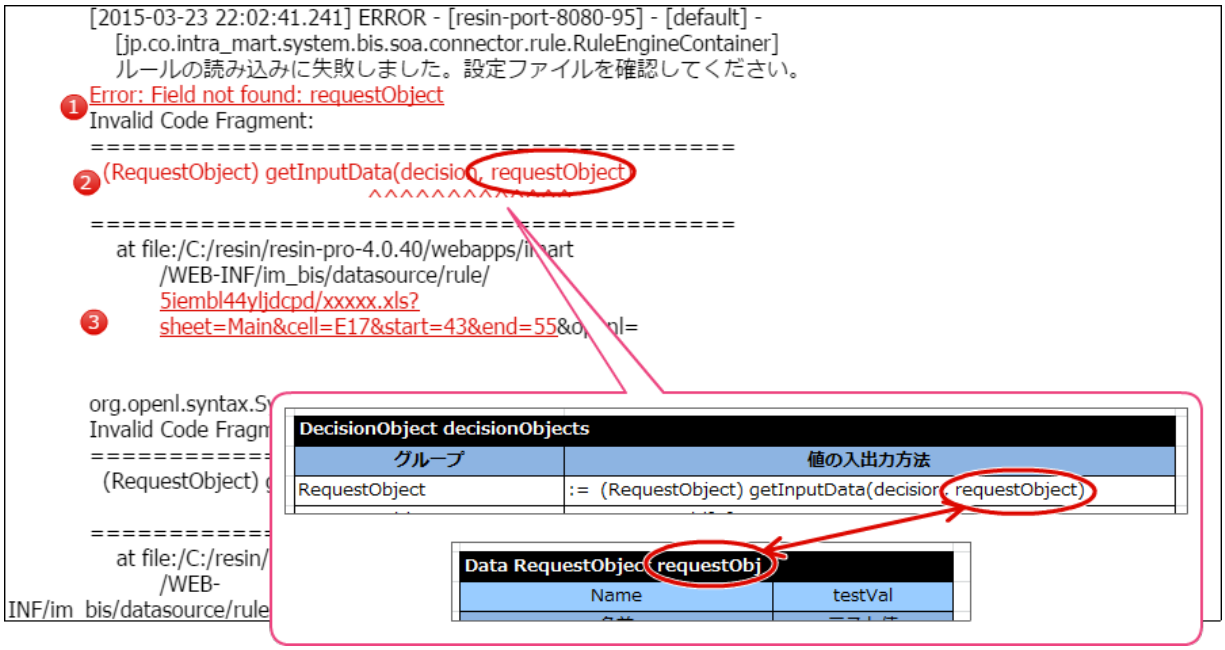
- *DecisionObject* が参照している変数がExcelファイルで定義されていない、または間違っている。

解決方法

スタックトレース中に、Excelファイルの誤っている可能性のあるセルが書いてありますので、そのセルの内容と関連する定義を確認してください。

上のスタックトレースの場合の読み方は、次の通りです。
下記の読み方に基づいて、Excelの定義ファイルを修正し、再度データソース定義にアップロードしてください。

- *DecisionObject* で定義したインスタンス名と *Data/Variable* で定義したインスタンス名に不整合が発生している例



1 Error: Field not found: <変数名>

- <変数名>で定義されたインスタンスが、 *Data/Variable* に定義されていないことを表します。

2 (RequestObject) getInputData(decision, requestObject)

- エラーが発生しているExcel内の式の箇所を表します。

3 xxxxx.xls?sheet=<シート名>&cell=< 2 の式の箇所>

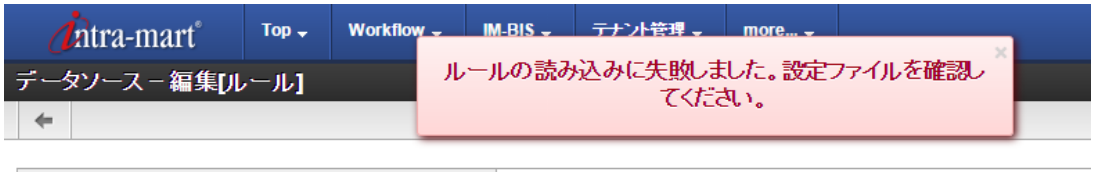
- エラーが発生しているExcelの定義ファイル内のシートやセルを表します。

DecisionTable の演算子を全角で定義している

現象

データソース定義にExcelファイルをアップロードし、「登録」または「更新」をクリックしたときに下記のエラーメッセージが表示される。

- 「ルールの読み込みに失敗しました。設定ファイルを確認してください。」



条件

- bis.logやコンソールに下記のようなスタックトレースが出力されている。

```
[2015-03-23 22:22:33.453] ERROR - [resin-port-8080-97] - [default] -
 [jp.co.intra_mart.system.bis.soa.connector.rule.RuleEngineContainer]
 ルールの読み込みに失敗しました。設定ファイルを確認してください。
Error: Oper = is not defined : java.lang.RuntimeException
 at file:/C:/resin/resin-pro-4.0.40_forma/webapps/imart/WEB-INF/im_bis/datasource/rule/5iembl44yljdcpd/
 xxxxxx.xls?sheet=DecisionTable&cell=B6&openl=
・・・
Caused by: java.lang.RuntimeException: Oper = is not defined
   at com.openrules.types.Oper.<init>(Oper.java:189)
 ... 117 more
・・・
org.openl.syntax.SyntaxErrorException: Error: Oper = is not defined : java.lang.RuntimeException
    at file:/C:/resin/resin-pro-4.0.40_forma/webapps/imart/WEB-INF/im_bis/datasource/rule/5iembl44yljdcpd/
    xxxxxx.xls?sheet=DecisionTable&cell=B6&openl=
・・・
```

原因

Excelファイル内に下記の定義誤りが存在する場合に発生します。

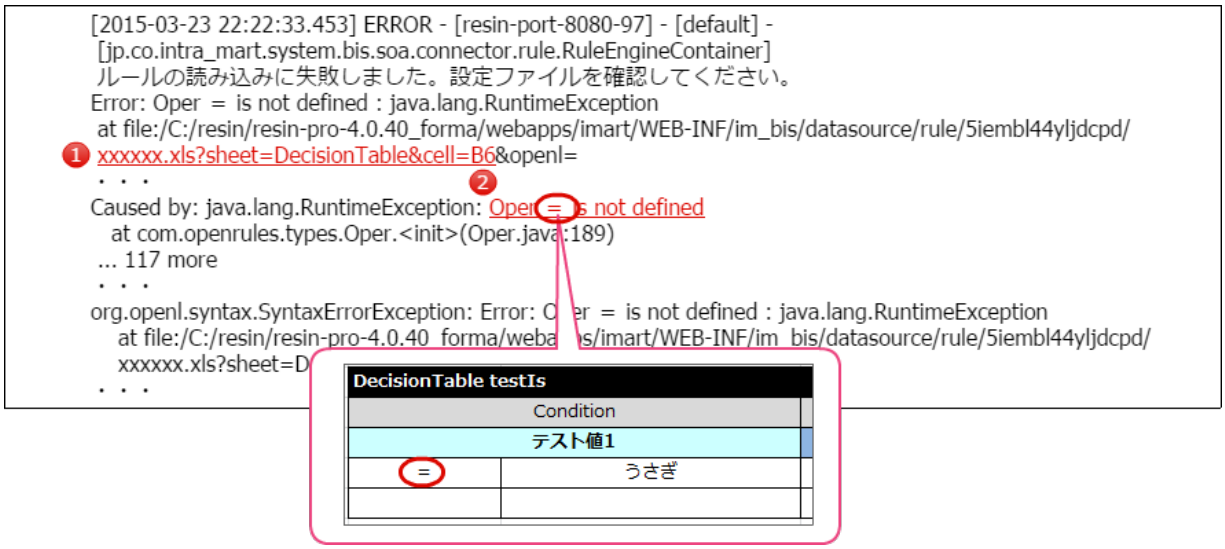
- *DecisionTable* や *Decision* で使っている演算子が、半角ではなく全角になっている
- *DecisionTable* や *Decision* で使っている演算子が、利用している OpenRules のバージョンで利用できない演算子となっている

解決方法

スタックトレース中に、Excelファイルの誤っている可能性のあるセルが書いてありますので、そのセルの内容と関連する定義を確認してください。

上のスタックトレースの場合の読み方は、次の通りです。
下記の読み方に基づいて、Excelの定義ファイルを修正し、再度データソース定義にアップロードしてください。

- *DecisionTable* で利用している演算子を全角で記述した例



1 xxxxxx.xls?sheet=<シート名>&cell=< 2 の箇所>

- エラーが発生しているExcelの定義ファイル内のシートやセルを表します。

2 Oper <演算子> is not defined

- 演算子で利用している記号やキーワードが、 OpenRules で提供されているものと異なっている、全角など不適切な値になっていることを表します。

実行している OpenRules のバージョンは、ルールが実行されたタイミングでコンソール上から確認できます。
下記のように実行する直前数行上のエンジン初期化のログにバージョン名（例：OPENRULES ENGINE 6.3.2 Alpha Evaluation Version）が出力されます。

```
[INFO] o.o.u.Log - [] INITIALIZE OPENRULES ENGINE 6.3.2 Alpha Evaluation Version (build 08112014) for [file:C:\resin\resin-pro-4.0.40\webapps\imart\WEB-INF\・・・
[INFO] o.o.u.Log - [] INCLUDE=../lib/openrules.config/IntramartTemplate.xls
・・・
[INFO] o.o.u.Log - [] *** Decision <実行しているDecision名> ***
[INFO] o.o.u.Log - [] Decision has been initialized
[INFO] o.o.u.Log - [] Decision Run has been initialized
[INFO] o.o.u.Log - [] Decision <実行しているDecision名>:
```

エンジンのバージョンによる演算子の利用可否については、「*OpenRules のバージョンによる記法の差異*」も確認してください。

ルールの実行時にエラーが発生する、正しく動作しない

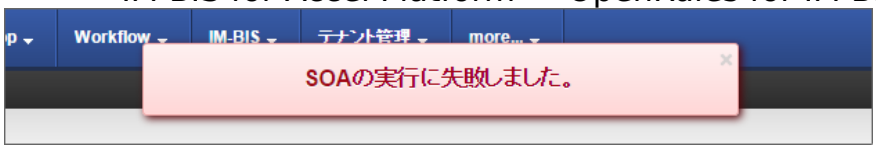
ルールの実行で発生するエラー

- *DecisionTable* で利用している項目が *Glossary* に存在しない
- 「SOAの実行に失敗しました」と表示され、ログにNullPointerExceptionが出力される

DecisionTable で利用している項目が *Glossary* に存在しない

現象

ワークフローの申請画面などの実行画面上で、ルールを実行するイベントを発生させたときに下記のエラーメッセージが表示される。



条件

- bis.logやコンソールに下記のようなスタックトレースが出力されている。

```
[2015-03-23 21:10:15.028] ERROR - [resin-port-8080-99] - [default] - [bis.common.user_program.soa_executor] SOAを実行している最中に例外が発生しました。システム管理者に問い合わせてください。
[2015-03-23 21:17:32.762] ERROR - [resin-port-8080-200] - [default] - [jp.co.intra_mart.system.bis.soa.connector.rule.RuleConnector] ルールを実行している際にエラーが発生しました。
ERROR in Glossary: cannot find element 'テスト値1'
org.openl.binding.OpenLRuntimeException: ERROR in Glossary: cannot find element 'テスト値1'
・・・
Caused by: org.apache.commons.lang.exception.NestableRuntimeException: ERROR in Glossary: cannot find element 'テスト値1'
・・・
[2015-03-23 21:17:32.889] ERROR - [resin-port-8080-200] - [default] - [bis.common.user_program.soa_executor] SOAを実行している最中に例外が発生しました。システム管理者に問い合わせてください。
```

原因

Excelファイル内に下記の定義誤りが存在する場合に発生します。

- *DecisionTable* で *Condition* や *Conclusion* などに使用している項目が *Glossary* に定義されていない、または、項目名が間違っている。

解決方法

スタックトレース中に、定義されていない項目名が出ていますので、その項目名と *Glossary* の定義を照らし合わせてください。

上のスタックトレースの場合の読み方は、下記の通りです。
下記の読み方に基づいて、Excelの定義ファイルを修正し、再度データソース定義にアップロードしてください。

```
[2015-03-23 21:10:15.028] ERROR - [resin-port-8080-99] - [default] -
[bis.common.user_program.soa_executor] SOAを実行している最中に例外が発生しました。
システム管理者に問い合わせてください。
[2015-03-23 21:17:32.762] ERROR - [resin-port-8080-200] - [default] -
[jp.co.intra_mart.system.bis.soa.connector.rule.RuleConnector] ルールを実行している際にエラーが発生しました。
1 ERROR in Glossary: cannot find element 'テスト値1'
org.openl.binding.OpenLRuntimeException: ERROR in Glossary: cannot find element 'テスト値1'
・・・
Caused by: org.apache.commons.lang.exception.NestableRuntimeException:
ERROR in Glossary: cannot find element 'テスト値1'
・・・
[2015-03-23 21:17:32.889] ERROR - [resin-port-8080-200] - [default] -
[bis.common.user_program.soa_executor] SOAを実行している最中に例外が発生しました。
システム管理者に問い合わせてください。
```

- 1 ERROR in Glossary: cannot find element '対象の項目名'
  - *DecisionTable* で'対象の項目名' が *Glossary* で見つけられないことを表しています。

「SOAの実行に失敗しました」と表示され、ログにNullPointerExceptionが出力される

現象

ワークフローの申請画面などの実行画面上で、ルールを実行するイベントを発生させたときに下記のエラーメッセージが表示される。

条件

- bis.logやコンソールに下記のようなスタックトレースが出力されている。
  ログの内容には、 OpenRules で処理ができているかのような内容とNullPointerExceptionが出力されている。

```
[INFO] o.o.u.Log - [] *** Decision myRule ***
[INFO] o.o.u.Log - [] Decision has been initialized
```

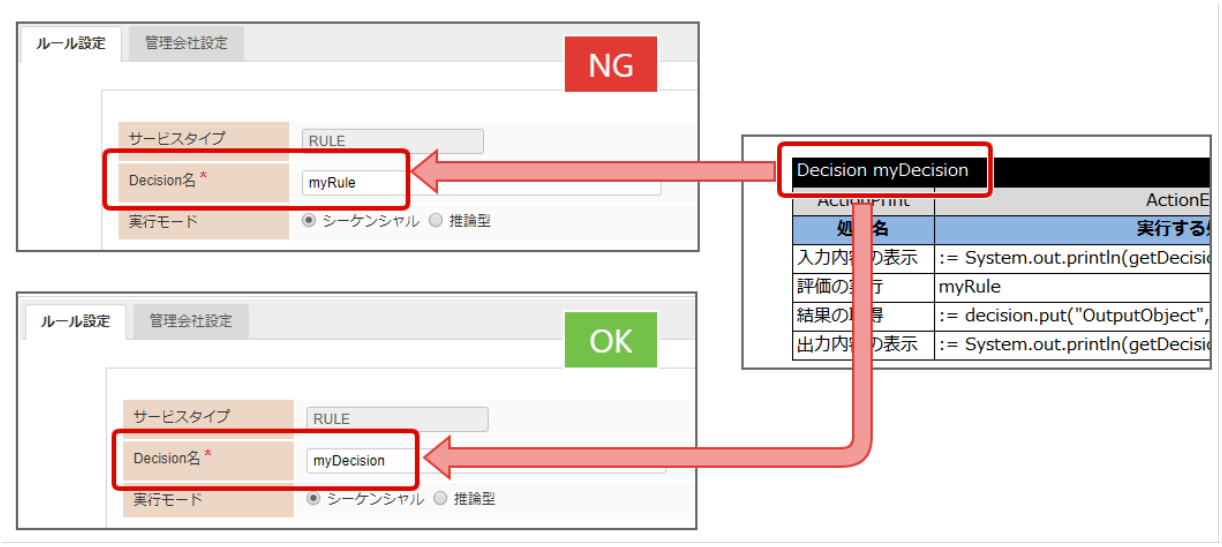
```
[INFO] o.o.u.Log - [] Decision Run has been initialized
[INFO] o.o.u.Log - [] Conclusion: メッセージ = 中学生 [中学生]
[INFO] o.o.u.Log - [] Decision has been finalized
java.lang.NullPointerException
	at jp.co.intra_mart.system.bis.soa.mapping.RuleObjectMapper.getMap(RuleObjectMapper.java:357)
	at jp.co.intra_mart.system.bis.soa.mapping.RuleObjectMapper.mapping(RuleObjectMapper.java:78)
	at jp.co.intra_mart.system.bis.soa.connector.rule.RuleConnector.executeResponseDef(RuleConnector.java:225)
	at jp.co.intra_mart.system.bis.internal.SoaClientManager.execute(SoaClientManager.java:93)
	at jp.co.intra_mart.system.bis.javascript.api.SoaClientManagerObject.execute(SoaClientManagerObject.java:111)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:497)
	at jp.co.intra_mart.system.javascript.MemberBox.invoke(MemberBox.java:126)
	at jp.co.intra_mart.system.javascript.FunctionObject.call(FunctionObject.java:442)
	at jp.co.intra_mart.system.javascript.optimizer.OptRuntime.callN(OptRuntime.java:52)
	at _forma._common._soa._soa_95_client_46_js._c_execute_1(C:\Resin\resin-pro-4.0.53\webapps\imart\WEB-
INF\jssp\product\src\forma\common\soa\soa_client.js:37)
	at _forma._common._soa._soa_95_client_46_js.call(C:\Resin\resin-pro-4.0.53\webapps\imart\WEB-
INF\jssp\product\src\forma\common\soa\soa_client.js)
	at jp.co.intra_mart.system.display.ScriptScope.call(ScriptScope.java:156)
	at jp.co.intra_mart.system.display.ScriptScope.call(ScriptScope.java:142)
	at jp.co.intra_mart.system.display.Content.executeFunction(Content.java:188)
	at jp.co.intra_mart.system.javascript.imapi.ContentObject.jsStaticFunction_executeFunction(ContentObject.java:170)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:497)
	at jp.co.intra_mart.system.javascript.MemberBox.invoke(MemberBox.java:126)
	at jp.co.intra_mart.system.javascript.FunctionObject.call(FunctionObject.java:379)
	at jp.co.intra_mart.system.javascript.optimizer.OptRuntime.callN(OptRuntime.java:52)
	at _forma._common._util._datasource_95_utils_46_js._c_execute_2(C:\Resin\resin-pro-4.0.53\webapps\imart\WEB-
INF\jssp\product\src\forma\common\util\datasource_utils.js:30)
	at _forma._common._util._datasource_95_utils_46_js.call(C:\Resin\resin-pro-4.0.53\webapps\imart\WEB-
INF\jssp\product\src\forma\common\util\datasource_utils.js)
	at jp.co.intra_mart.system.javascript.optimizer.OptRuntime.callN(OptRuntime.java:52)
	at _bis._common._user_95_program._soa_95_executor_46_js._c_execute_1(C:\Resin\resin-pro-4.0.53\webapps\imart\WEB-
INF\jssp\product\src\bis\common\user_program\soa_executor.js:41)
	at _bis._common._user_95_program._soa_95_executor_46_js.call(C:\Resin\resin-pro-4.0.53\webapps\imart\WEB-
INF\jssp\product\src\bis\common\user_program\soa_executor.js)
	at jp.co.intra_mart.system.display.ScriptScope.call(ScriptScope.java:156)
	at jp.co.intra_mart.system.display.ScriptScope.call(ScriptScope.java:142)
	at jp.co.intra_mart.system.display.Content.executeFunction(Content.java:188)
	at jp.co.intra_mart.system.javascript.imapi.ContentObject.jsStaticFunction_executeFunction(ContentObject.java:170)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:497)
	at jp.co.intra_mart.system.javascript.MemberBox.invoke(MemberBox.java:126)
	at jp.co.intra_mart.system.javascript.FunctionObject.call(FunctionObject.java:379)
	at jp.co.intra_mart.system.javascript.optimizer.OptRuntime.callN(OptRuntime.java:52)
	at _forma._designer._event_95_setting._reference_46_js._c_executeSoa_13(C:\Resin\resin-pro-4.0.53\webapps\imart\WEB-
INF\jssp\product\src\forma\designer\event_setting\reference.js:566)
	at _forma._designer._event_95_setting._reference_46_js.call(C:\Resin\resin-pro-4.0.53\webapps\imart\WEB-
INF\jssp\product\src\forma\designer\event_setting\reference.js)
	at jp.co.intra_mart.system.javascript.ContextFactory.doTopCall(ContextFactory.java:394)
	at jp.co.intra_mart.system.javascript.ScriptRuntime.doTopCall(ScriptRuntime.java:3101)
	at _forma._designer._event_95_setting._reference_46_js.call(C:\Resin\resin-pro-4.0.53\webapps\imart\WEB-
INF\jssp\product\src\forma\designer\event_setting\reference.js)
	at jp.co.intra_mart.system.display.ScriptScope.call(ScriptScope.java:156)
	at jp.co.intra_mart.system.display.ScriptScope.call(ScriptScope.java:142)
	at jp.co.intra_mart.system.servlet.jssp.JsspRpcServlet.invoke(JsspRpcServlet.java:177)
	at jp.co.intra_mart.system.servlet.jssp.JsspRpcServlet.execute(JsspRpcServlet.java:141)
	at jp.co.intra_mart.system.servlet.jssp.JsspRpcServlet.doPost(JsspRpcServlet.java:125)
	at javax.servlet.http.HttpServlet.service(HttpServlet.java:159)
	at javax.servlet.http.HttpServlet.service(HttpServlet.java:97)
	at com.caucho.server.dispatch.ServletFilterChain.doFilter(ServletFilterChain.java:109)
	at jp.co.intra_mart.system.servlet.jssp.JSSPContextFilter.doFilter(JSSPContextFilter.java:63)
	at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
	at
jp.co.intra_mart.common.aid.jsdk.javax.servlet.filter.impl.HTTPContextHandlingFilterImpl.doFilter(HTTPContextHandlingFilterImpl.java:53)
	at jp.co.intra_mart.common.aid.jsdk.javax.servlet.filter.HTTPContextHandlingFilter.doFilter(HTTPContextHandlingFilter.java:94)
	at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
	at jp.co.intra_mart.system.servlet.filter.RequestScopeLockReleaseFilter.doFilter(RequestScopeLockReleaseFilter.java:44)
	at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
	at jp.co.intra_mart.system.secure.filter.ApplicationPermissionFilter.doFilter(ApplicationPermissionFilter.java:65)
	at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
	at jp.co.intra_mart.system.secure.filter.SystemPermissionFilter.doFilter(SystemPermissionFilter.java:68)
	at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
	at jp.co.intra_mart.foundation.router.RoutingFilter.doFilter(RoutingFilter.java:41)
```

```
at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
at jp.co.intra_mart.foundation.security.filter.SessionFilter.doFilter(SessionFilter.java:70)
at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
at jp.co.intra_mart.system.log.transition.TransitionLogFilter.doFilter(TransitionLogFilter.java:73)
at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
at jp.co.intra_mart.system.servlet.filter.RequestLogFilter.doFilter(RequestLogFilter.java:110)
at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
at jp.co.intra_mart.system.servlet.filter.ResponseHeaderFilter.doFilter(ResponseHeaderFilter.java:154)
at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
at jp.co.intra_mart.system.context.web.impl.ContextFilter.doContextFilter(ContextFilter.java:137)
at jp.co.intra_mart.system.context.web.impl.PreContextFilterChain.doFilter(PreContextFilterChain.java:47)
at jp.co.intra_mart.system.context.web.impl.ContextFilter.doFilter(ContextFilter.java:78)
at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
at jp.co.intra_mart.foundation.security.filter.ResponseCharacterEncodingFilter.doFilter(ResponseCharacterEncodingFilter.java:90)
at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
at jp.co.intra_mart.foundation.security.filter.RequestCharacterEncodingFilter.doFilter(RequestCharacterEncodingFilter.java:47)
at com.caucho.server.dispatch.FilterFilterChain.doFilter(FilterFilterChain.java:89)
at com.caucho.server.webapp.WebAppFilterChain.doFilter(WebAppFilterChain.java:156)
at com.caucho.server.webapp.AccessLogFilterChain.doFilter(AccessLogFilterChain.java:95)
at com.caucho.server.dispatch.ServletInvocation.service(ServletInvocation.java:290)
at com.caucho.server.http.HttpRequest.handleRequest(HttpRequest.java:838)
at com.caucho.network.listen.TcpSocketLink.dispatchRequest(TcpSocketLink.java:1353)
at com.caucho.network.listen.TcpSocketLink.handleRequest(TcpSocketLink.java:1309)
at com.caucho.network.listen.TcpSocketLink.handleRequestsImpl(TcpSocketLink.java:1293)
at com.caucho.network.listen.TcpSocketLink.handleRequests(TcpSocketLink.java:1201)
at com.caucho.network.listen.TcpSocketLink.handleAcceptTaskImpl(TcpSocketLink.java:997)
at com.caucho.network.listen.ConnectionTask.runThread(ConnectionTask.java:117)
at com.caucho.network.listen.ConnectionTask.run(ConnectionTask.java:93)
at com.caucho.network.listen.SocketLinkThreadLauncher.handleTasks(SocketLinkThreadLauncher.java:169)
at com.caucho.network.listen.TcpSocketAcceptThread.run(TcpSocketAcceptThread.java:61)
at com.caucho.env.thread2.ResinThread2.runTasks(ResinThread2.java:173)
at com.caucho.env.thread2.ResinThread2.run(ResinThread2.java:118)
```

原因

Excelファイル内に下記の定義誤りが存在する場合に発生します。

- *Decision* に定義したDecision名が、データソース定義のDecision名と一致していない



解決方法

データソース定義のDecision名を *Decision* で定義したDecision名と一致させてください。

OpenRules でデバッグをするための設定

OpenRules をデータソース定義に登録して利用する場合に、デバッグを行うための方法です。

OpenRules でのデバッグ方法
- OpenRules のログをログファイルに出力するための設定
- OpenRules の入出力内容をコンソールに出力するための設定

OpenRules のログをログファイルに出力するための設定

OpenRules のログを出力するためのロガーを設定します。
デバッグ時には、「trace」レベルのログまで出力すると、値の受け渡しの処理などを確認できます。
運用時にはログファイルサイズが大きくなりすぎるため、「WARN」以上とすることを推奨します。

設定ファイルの設定

設定ファイルに以下のように記述すると、 OpenRules の実行時のログをすべて出力することができます。
ファイルの詳細は「 設定ファイルリファレンス 」で確認してください。

```
<logger name="org.openl.util.Log">
  <level value="all" />
</logger>
```

上記の設定後には、サーバを再起動すると変更内容が反映されます。

Excelの定義ファイルの設定

Excelの定義ファイル上で、 *Decision* に以下のように記述すると、各処理のタイミングで渡されたオブジェクトの値をログファイルに出力します。

Decision sampleDailyAllowance	
ActionPrint	ActionExecute
Decisions	**Execute Decision Tables**
入力内容の表示	:= Log.trace(getDecisionObject("RequestObject"))
場所の変換	convertArea
出張区分の変換	convertTripClass
日当の計算	setDailyAllowance
結果の取得	:= decision.put("ResponseObject", responseObj)
出力内容の表示	:= Log.trace(getDecisionObject("ResponseObject"))

出力されるログイメージ

上記のように設定を行った後、ルールを実行すると、以下のような形で出力されますので、ログを確認し、エラーの解決等にご利用ください。

```
[2015-03-24 11:48:01.580] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Execute setDecisionVar
[2015-03-24 11:48:01.580] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Execute initializeDecision
[2015-03-24 11:48:01.581] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] *** Decision sampleDailyAllowance ***
[2015-03-24 11:48:01.581] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Decision has been initialized
[2015-03-24 11:48:01.581] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Execute decisionObjects
[2015-03-24 11:48:01.581] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Invoke DecisionObjectTemplate
[2015-03-24 11:48:01.581] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] []
[2015-03-24 11:48:01.581] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Invoke decisionObjects
[2015-03-24 11:48:01.581] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] [TTT]
[2015-03-24 11:48:01.581] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Execute initializeDecisionRun
[2015-03-24 11:48:01.581] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Decision Run has been initialized
[2015-03-24 11:48:01.581] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Execute sampleDailyAllowance
[2015-03-24 11:48:01.582] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Invoke DecisionTemplate
[2015-03-24 11:48:01.582] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Invoke sampleDailyAllowance
[2015-03-24 11:48:01.582] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Decision sampleDailyAllowance: 入力内
容の表示
[2015-03-24 11:48:01.582] [resin-port-8080-87] TRACE org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] RequestObject(id=0) {
  areaClassCode=overseas
  tripClassCode=single-day
}
[2015-03-24 11:48:01.582] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Decision sampleDailyAllowance: 場所の
変換
[2015-03-24 11:48:01.582] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Execute convertArea
[2015-03-24 11:48:01.582] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Invoke convertArea
[2015-03-24 11:48:01.582] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] [fT]
[2015-03-24 11:48:01.582] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Conclusion: 場所 = 国外
[2015-03-24 11:48:01.582] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Decision sampleDailyAllowance: 出張区
分の変換
[2015-03-24 11:48:01.583] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Execute convertTripClass
[2015-03-24 11:48:01.583] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Invoke convertTripClass
[2015-03-24 11:48:01.583] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] [fT]
[2015-03-24 11:48:01.583] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Conclusion: 出張区分 = 日帰り
[2015-03-24 11:48:01.583] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Decision sampleDailyAllowance: 日当の
計算
[2015-03-24 11:48:01.583] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Execute setDailyAllowance
[2015-03-24 11:48:01.584] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Invoke setDailyAllowance
[2015-03-24 11:48:01.584] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] [ffT]
[2015-03-24 11:48:01.584] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Conclusion: 日当 = 2000
[2015-03-24 11:48:01.584] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Decision sampleDailyAllowance: 結果の
取得
[2015-03-24 11:48:01.584] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Decision sampleDailyAllowance: 出力内
容の表示
[2015-03-24 11:48:01.584] [resin-port-8080-87] TRACE org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] ResponseObject(id=0) {
  dailyAllowance=2000
  dummy=ダミー内容
}
[2015-03-24 11:48:01.584] [resin-port-8080-87] DEBUG org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Execute finalizeDecision
[2015-03-24 11:48:01.584] [resin-port-8080-87] INFO  org.openl.util.Log   tenant1    5ieoz3s029tyzpd   -  [] Decision has been finalized
```

OpenRules の入出力内容をコンソールに出力するための設定

OpenRules の実行時に入力・出力のオブジェクトが保持している値をコンソールに出力するための設定方法です。
ログファイルの出力時より出力される内容は少なくなりますが、実行前後の入出力の内容がコンソールに出力されますので、開発環境などコンソールで確認が行いやすい場合には簡単に利用できます。

Excelの定義ファイルの設定

Excelの定義ファイル上で、 *Decision* に以下のように記述すると、各処理のタイミングで渡されたオブジェクトの値をコンソールに出力します。

Decision sampleDailyAllowance	
ActionPrint	ActionExecute
Decisions	**Execute Decision Tables**
入力内容の表示	:= System.out.println(getDecisionObject("RequestObject"))
場所の変換	convertArea
出張区分の変換	convertTripClass
日当の計算	setDailyAllowance
結果の取得	:= decision_put("ResponseObject", responseObj)
出力内容の表示	:= System.out.println(getDecisionObject("ResponseObject"))

出力されるコンソールのイメージ

上記のように設定を行った後、ルールを実行すると、以下のような形で出力されますので、内容を確認し、エラーの解決等にご利用ください。

```
[INFO] o.o.u.Log - [] *** Decision sampleDailyAllowance ***
[INFO] o.o.u.Log - [] Decision has been initialized
[INFO] o.o.u.Log - [] Decision Run has been initialized
[INFO] o.o.u.Log - [] Decision sampleDailyAllowance: 入力内容の表示
RequestObject(id=0) {
  areaClassCode=overseas
  tripClassCode=single-day
}
[INFO] o.o.u.Log - [] Decision sampleDailyAllowance: 場所の変換
[INFO] o.o.u.Log - [] Conclusion: 場所 = 国外
[INFO] o.o.u.Log - [] Decision sampleDailyAllowance: 出張区分の変換
[INFO] o.o.u.Log - [] Conclusion: 出張区分 = 日帰り
[INFO] o.o.u.Log - [] Decision sampleDailyAllowance: 日当の計算
[INFO] o.o.u.Log - [] Conclusion: 日当 = 2000
[INFO] o.o.u.Log - [] Decision sampleDailyAllowance: 結果の取得
[INFO] o.o.u.Log - [] Decision sampleDailyAllowance: 出力内容の表示
ResponseObject(id=0) {
  dailyAllowance=2000
  dummy=ダミー内容
}
[INFO] o.o.u.Log - [] Decision has been finalized
```

OpenRules のキーワードリファレンス

OpenRules for IM-BIS 連携開発でのルールを記述する際に利用できるキーワードをまとめています。

テーブルタイプ（TableType）

OpenRules で定義する各種テーブルタイプです。

- 条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）
- マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）
- マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）
- 変数への値割り当て（DecisionTableAssign）
- ルールの反復処理（DecisionTableIterate）
- スコアに基づくソート処理（DecisionTableSort）
- 処理順の設定（Decision）
- 項目名のマッピング（Glossary）
- 項目とデータ型の定義（Datatype）
- 項目の初期値の定義（Data / Variable）
- オブジェクトのインスタンスの設定（DecisionObject）
- Javaのコードの定義（Method）
- 環境設定情報（Environment）

条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）

Decision に基づいて、Excelのシートで上から書いてある順に条件を評価します。
条件が合致したら、合致した行の結果・アクションの設定内容を実行し、処理が終了します。
従って、合致した行より下の行は評価されません。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	○

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
サブヘッダ部は、条件や評価（処理）単位でセルを結合します。

(1)	**DecisionTable** sampleDecision			
(2)	a Condition		b Conclusion	
	c PurchasePrice		c ApproveFromManager	
(3)	>	100000	=	TRUE

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下のいずれかの方法で記述します。

- DecisionTable %テーブル名%
- DT %テーブル名%
- DecisionTableSingleHit %テーブル名%
- RuleFamily %テーブル名%
  - 各キーワードの後に、半角スペースを入れてテーブル名を記述します。

(2) サブヘッダに利用できるキーワード

サブヘッダには、「結果」に利用できるキーワード、または、「条件」・「結果」のキーワードの組み合わせを1つ以上含める必要があります。

a. 条件に利用できるキーワード

キーワード	利用可否
OnOff	○
Condition	○
ConditionBetween	○
ConditionVarOperValue	○
ConditionIntOperInt	○
ConditionRealOperReal	○
ConditionDateOperDate	○
ConditionAny	○
ConditionMap	×
If	○

コラム

条件のセルの条件値を記入しない場合、 OpenRules では無条件と判断されます。
すべての条件に合致しない場合の処理を記述する際などに利用します。

コラム

OpenRules 6.3.4で列タイプ「OnOff」が追加されました。
利用方法は以下を参照してください。

- *OnOff*

b. 結果に利用できるキーワード

キーワード	利用可否
Conclusion	○
ConclusionVarOperValue	○
ActionAny	○
ActionMap	×
Action	○
ActionPrint	×
Then	○
ActionExecute	○
Message	○
ActionIterate	×
ActionRulesOnArray	×
ActionSort	×

c. 項目の論理名

Glossary で定義した項目の論理名を入力してください。
この項目にはひらがなや漢字などのマルチバイト文字も利用できます。

(3) 明細

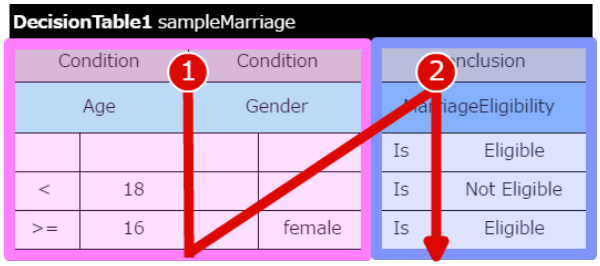
(2)サブヘッダのキーワードに合わせて記述します。
条件や処理の記述は以下を参照してください。

- *条件として利用できるキーワード*
- *結果・処理として利用できるキーワード*

マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）

Decision に基づいて、Excelのシートで上から書いてある順に条件を評価します。
条件に合致した、しないに関わらずすべての条件を評価します。
合致したすべての条件に設定された結果（Actionなど）はすべて実行されます。

このテーブルタイプでは、以下の順に評価・実行します。

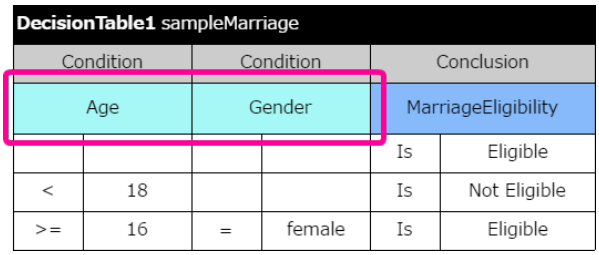


1. すべての条件を評価します。
2. 条件が合致した行に対応する結果（サブヘッダに Conclusion/Then/Action/ActionAny/Messageを設定した列）を、上から順に実行します。
   値を返却する場合、最後に実行された行の値が返却されます。

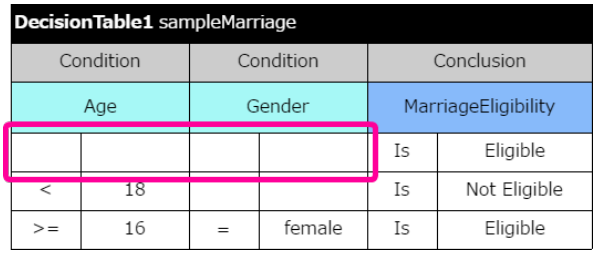
以下の図は、処理の流れの例です。

1. Conditionの入力値として以下の表の値が設定された場合の処理を例に説明します。

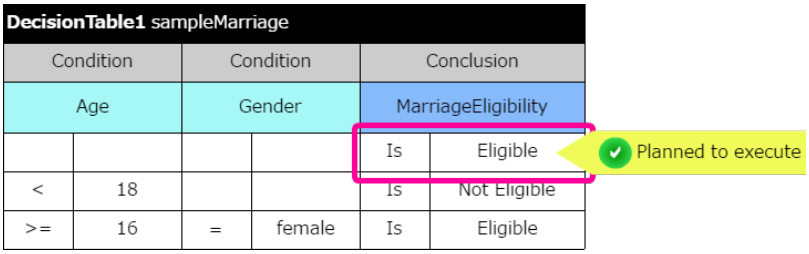
Age （年齢）	Gender （性別）
17	female （女性）



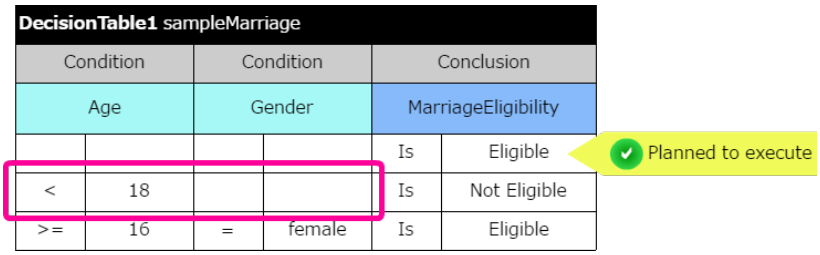
2. 1行目の条件が評価されます。
   Condition（条件）が空欄のため、必ずTrue（Marriage Eligibility（結婚の可否）＝”Eligible”（結婚可能））と返却されます。



3. この時点では結果の変数「MarriageEligibility」への値の設定は行われず、「実行予定」のフラグが設定されます。
   結果の変数「Marriage Eligibility」の値は初期値の「未設定」のまま変わりません。

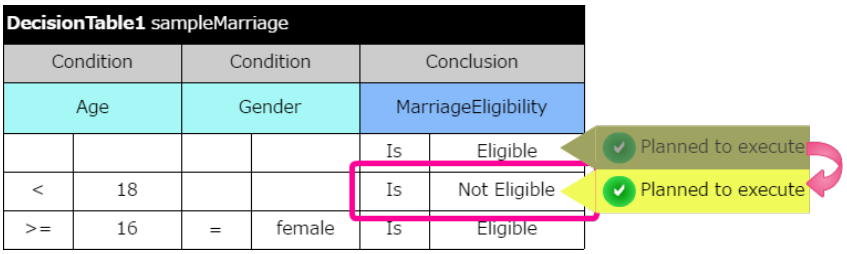


4. 2行目の条件が評価されます。
   入力値の年齢「17」が条件に合致するため、Trueと評価されます。

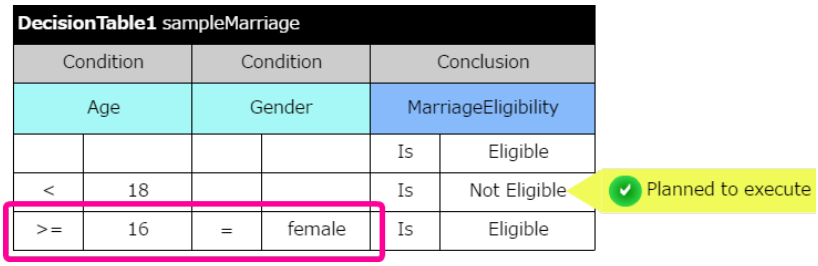


5. この時点でも結果の変数「MarriageEligibility」への値の設定は行われません。

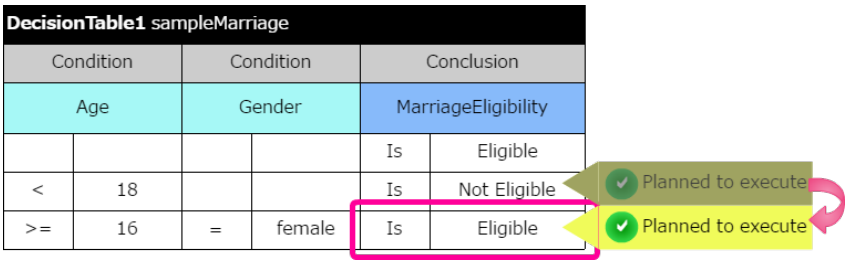
「実行予定」のフラグが1行目の結果から2行目に変更されます。
結果の変数「Marriage Eligibility」の値は初期値の「未設定」のまま変わりません。



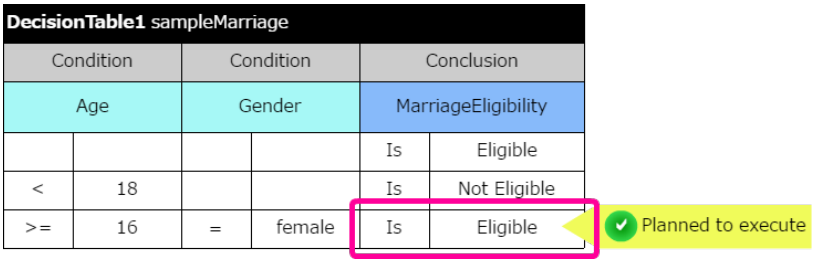
6. 3行目の条件が評価されます。
入力値のAge「17」、Gender「女性」の両方が条件に合致するため、Trueと評価されます。



7. 「実行予定」のフラグが2行目の結果から3行目に変更されます。



8. 最終的にすべての条件の評価後、「実行予定」のフラグが設定されている3行目の「Eligible」が結果の変数「Marriage Eligibility」に設定されます。



DecisionTable1では、以下の点が重要なポイントです。

- それぞれの条件の結果（actionなど）は、他の条件には影響しません。
  DecisionTable1は、対象のDecisionTableのすべての条件の評価後に結果を設定します。
  そのため、先に評価された条件の結果は、後続の条件の評価には影響しません。
- 先に評価された条件の結果（actionなど）を上書きすることができます。
  DecisionTable1では、先に「実行予定」となった条件の結果を、後で評価した条件の結果に変更する（上書きする）ことができます。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	×

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
サブヘッダ部は、条件や評価（処理）単位でセルを結合します。

(1)	DecisionTable1 sampleMarriage					
(2)	a Condition		a Condition		b Conclusion	
	c Age		c Gender		c MarriageEligibility	
(3)	>=	16	=	female	Is	Eligible

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下のいずれかの方法で記述します。

- DecisionTable1 %テーブル名%
- DT1 %テーブル名%
- DecisionTableMultiHit %テーブル名%
  - 「DecisionTable1」、「DT1」、「DecisionTableMultiHit」の後に、半角スペースを入れてテーブル名を記述します。

(2) サブヘッダに利用できるキーワード

サブヘッダには、「結果」に利用できるキーワード、または「条件」・「結果」のキーワードの組み合わせを1つ以上含める必要があります。

a. 条件に利用できるキーワード

キーワード	利用可否
OnOff	○
Condition	○
ConditionBetween	×
ConditionVarOperValue	○
ConditionIntOperInt	○
ConditionRealOperReal	○
ConditionDateOperDate	○
ConditionAny	○
ConditionMap	×
If	○

b. 結果に利用できるキーワード

キーワード	利用可否
Conclusion	○
ConclusionVarOperValue	○
ActionAny	○
ActionMap	×
Action	○
ActionPrint	×
Then	○
ActionExecute	○
Message	○
ActionIterate	×
ActionRulesOnArray	×
ActionSort	×

コラム

OpenRules 6.3.4で列タイプ「OnOff」が追加されました。
利用方法は以下を参照してください。

- *OnOff*

c. 項目の論理名

Glossary で定義した項目の論理名を入力してください。
この項目にはひらがなや漢字などのマルチバイト文字も利用できます。

(3) 明細

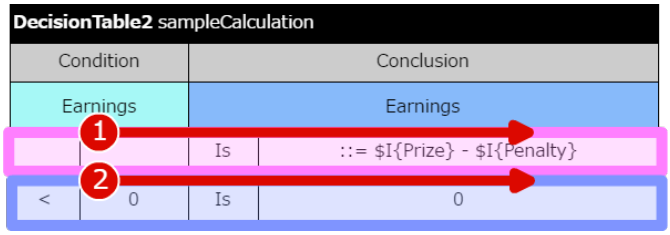
(2)サブヘッダのキーワードに合わせて記述します。
条件や処理の記述は以下を参照してください。

- *条件として利用できるキーワード*
- *結果・処理として利用できるキーワード*

マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）

Decision に基づいて、Excelのシートで上から書いてある順に条件を評価します。
条件に合致した、しないに関わらずすべての条件を評価します。
合致したすべての条件に設定された結果（Actionなど）はすべて実行されます。

このテーブルタイプでは、以下の順に評価・実行します。



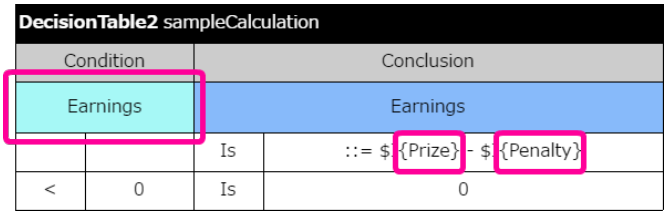
1. 上から順に条件を評価し、条件に合致したらすぐに結果列（サブヘッダに Conclusion/Then/Action/ActionAny/Messageを設定した列）を実行します。

2. 先に評価・実行された条件の結果に基づいて、後続の条件の評価・実行を同様に継続していきます。
   後続で実行された処理が計算の場合、その前に実行された処理によって設定された値に計算を行った結果を返却します。

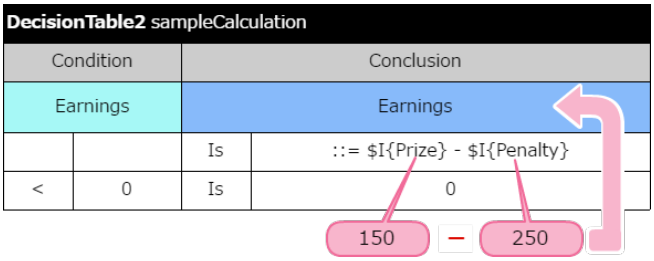
以下の図は、処理の流れの例です。

1. Condition / Conclusionの値として以下の表の値が設定された場合の処理を例に説明します。

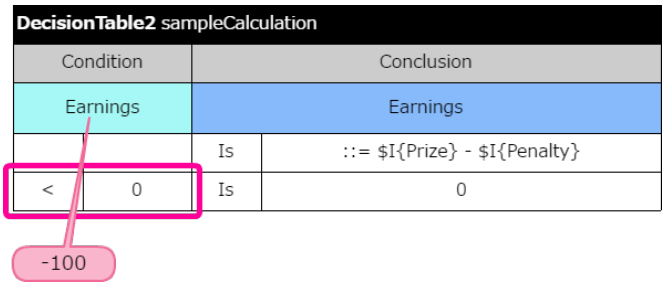
Prize（賞金）	Penalty（罰金）	Earnings（報酬）
150	250	0（初期値）



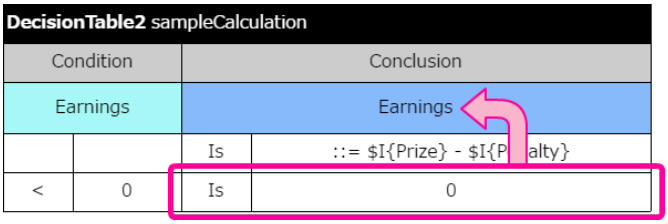
2. 1行目の条件が評価されます。
   Condition（条件）が空欄のため、Conclusionの計算を実行します。
   計算結果の「-100」の値はそのまま「Earnings」にセットされます。



3. 2行目の条件が評価されます。
   このときは、1行目の計算結果を反映した状態で評価されます。
   従って、Earningsは計算結果の「-100」を対象に評価されるため、条件に合致した「True」と評価されます。



4. 2行目の条件の評価に基づいて、Earningsには「0」が設定された上で処理結果として「0」が返却されます。



DecisionTable2では、以下の点が重要なポイントです。

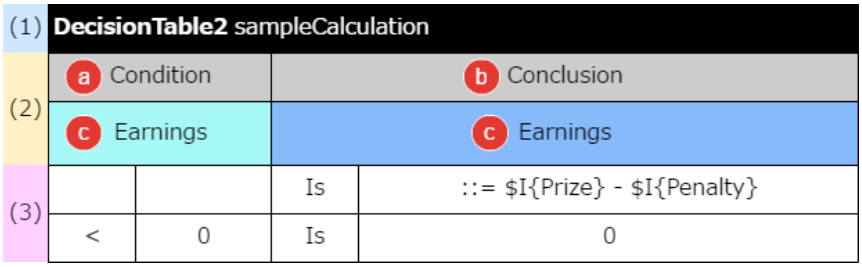
- それぞれの条件の結果（actionなど）が、後続の条件の評価に影響を及ぼします。
  DecisionTable2は、対象のDecisionTableのそれぞれの条件の評価・実行を順次行います。
  そのため、条件・結果の両方に同じ項目を設定している場合、先に評価された条件の結果で変更された項目の値に基づいて、後続の条件の評価が行われます。
- 後続の条件・結果（actionなど）が、先に評価・実行された条件・結果を上書きすることができます。
  DecisionTable2では、先に評価・実行された条件・結果を、後続の評価・実行する条件・結果で上書きする（変更する）ことができます。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	×

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
サブヘッダ部は、条件や評価（処理）単位でセルを結合します。



(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下のいずれかの方法で記述します。

- DecisionTable2 %テーブル名%
- DT2 %テーブル名%
- DecisionTableSequence %テーブル名%
  - 「DecisionTable2」、「DT2」、「DecisionTableSequence」の後に、半角スペースを入れてテーブル名を記述します。

(2) サブヘッダに利用できるキーワード

サブヘッダには、「結果」に利用できるキーワード、または「条件」・「結果」のキーワードの組み合わせを1つ以上含める必要があります。

a. 条件に利用できるキーワード

キーワード	利用可否
OnOff	○

キーワード	利用可否
Condition	○
ConditionBetween	×
ConditionVarOperValue	○
ConditionIntOperInt	○
ConditionRealOperReal	○
ConditionDateOperDate	○
ConditionAny	○
ConditionMap	×
If	○

b. 結果に利用できるキーワード

キーワード	利用可否
Conclusion	○
ConclusionVarOperValue	○
ActionAny	○
ActionMap	×
Action	○
ActionPrint	×
Then	○
ActionExecute	○
Message	○
ActionIterate	×
ActionRulesOnArray	×
ActionSort	×

コラム

OpenRules 6.3.4で列タイプ「OnOff」が追加されました。
利用方法は以下を参照してください。

- *OnOff*

c. 項目の論理名

Glossary で定義した項目の論理名を入力してください。
この項目にはひらがなや漢字などのマルチバイト文字も利用できます。

(3) 明細

(2)サブヘッダのキーワードに合わせて記述します。
条件や処理の記述は以下を参照してください。

- *条件として利用できるキーワード*
- *結果・処理として利用できるキーワード*

変数への値割り当て（DecisionTableAssign）

変数同士の計算結果を他の変数に代入する処理を簡潔に記述できるようにしたテーブルです。
OpenRules 内で変数同士の計算が複数行われる場合に見やすい形式でまとめることができます。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	×

タイプ	利用可否

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
サブヘッダ部は、条件や評価（処理）単位でセルを結合します。

(1)	**DecisionTableAssign** sampleDecision	
(2)	a **Variable**	b **Value**
(3)	計算結果1（CalculationResult1）	::= $R{項目1（Item1）}*$R{項目2（Item2）}
	計算結果2（CalculationResult2）	100

c　d

上記の内容は *DecisionTable* の指定内容と同義です。

DecisionTable sampleDecision	
Action	Action
計算結果1(CalculationResult1)	**計算結果2(CalculationResult2)**
::= $R{項目1（Item1）}*$R{項目2（Item2）}	::= $R{項目3（Item3）}+$R{項目4（Item4）}

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

- DecisionTableAssign %テーブル名%
  - 各キーワードの後に、半角スペースを入れてテーブル名を記述します。

(2) サブヘッダに利用できるキーワード

サブヘッダは、値の割り当て対象の項目の論理名の列ラベルと割り当てる値、または式の列ラベルのみ指定できます。

a. 項目の論理名

Glossary で定義した項目の論理名の列であることを識別しやすい文言を指定してください。
Conclusion などの OpenRules で利用する予約語は指定できません。
この項目にはひらがなや漢字などのマルチバイト文字も利用できます。

b. 列ラベル（値、割り当ての式の列）

(a) で指定した項目に割り当てる値や割り当てるための式を指定する列を表すラベルとしての文言を指定してください。
Conclusion などの OpenRules で利用する予約語は指定できません。
この項目にはひらがなや漢字などのマルチバイト文字も利用できます。

(3) 明細

このテーブルでは、以下の通りに指定してください。

c. 左の列（項目の論理名）
Glossary で定義した項目の論理名を指定してください。
(d)右の列で指定する内容に基づいて値を割り当てる項目が対象です。

d. 右の列（値・式）
(c)左の列で指定した項目に割り当てる値、または式を指定してください。
直接値を割り当てる場合は、割り当てる値をそのまま記述します。
他の項目の計算結果など式の結果に基づいて割り当てる場合は、”::=”をつけて計算式を指定してください。

ルールの反復処理（DecisionTableIterate）

オブジェクトの配列を受け取り、オブジェクト単位に別の *DecisionTable* を繰り返し配列の個数分実行します。
OpenRules 6.4.2から追加され、 *ActionRulesOnArray* 、 *ActionIterate* と同様の役割を果たします。
IM-BIS との連携では利用できません。

スコアに基づくソート処理（DecisionTableSort）

オブジェクトの配列を受け取り、オブジェクト単位に別の *DecisionTable* に基づいてスコアを算出し、スコア順にソートします。
OpenRules 6.4.2から追加され、 *ActionSort* と同様の役割を果たします。
IM-BIS との連携では利用できません。

処理順の設定（Decision）

DataMapperとの値の入出力、実行するDecisionTableの順序などを設定します。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	○

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
サブヘッダ部は、条件や評価（処理）単位でセルを結合します。

(1)	**Decision** sampleRules	
(2)	a ActionPrint	b ActionExecute
	c ProcessName	c ExecutionTargetProcess
(3)	Execute Evaluation	sampleDecision
	Execute Method	:= sampleMethod()

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下のいずれかの方法で記述します。

- Decision %テーブル名%
  - 「Decision」の後に、半角スペースを入れてテーブル名を記述します。

(2) サブヘッダに利用できるキーワード

サブヘッダは、「ActionPrint」、「ActionExecute」が必須です。
その他のキーワードは任意です。

a. 条件に利用できるキーワード

キーワード	利用可否
Condition	○
ConditionBetween	×
ConditionVarOperValue	×
ConditionIntOperInt	×
ConditionRealOperReal	×
ConditionDateOperDate	×
ConditionAny	○
ConditionMap	×
If	×

b. 結果に利用できるキーワード

キーワード	利用可否
Conclusion	○
ConclusionVarOperValue	○
ActionAny	○
ActionMap	×

キーワード	利用可否
Action	○
ActionPrint	○
Then	○
ActionExecute	○
Message	○
ActionIterate	×
ActionRulesOnArray	×
ActionSort	×

c. 列ラベル
明細部に入力する内容を識別するための名称を入力します。
処理には使用しない値のため、 *Glossary* への登録は不要です。
サブヘッダとして設定したキーワードが「ActionPrint」の列は「処理名」、「ActionExecute」は「実行対象のデシジョンテーブル」のように設定できます。
この項目にはひらがな、漢字などのマルチバイト文字が利用できます。

(3) 明細

(2)サブヘッダのキーワードに合わせて記述します。
条件や処理の記述は以下を参照してください。

- *条件として利用できるキーワード*
- *結果・処理として利用できるキーワード*

項目名のマッピング（Glossary）

ルールを定義するExcelファイルで利用している項目名の論理名と物理名をマッピングします。
OpenRules で利用するすべての項目を定義してください。
IM-BIS のデータソース定義のリクエスト・レスポンスにはここで定義された項目を設定してください。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	○

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
サブヘッダ部は、セルを結合する必要はありません。
明細部は、Business Concept単位に結合する必要があります。

(1)	**Glossary** glossary			
(2)	Variable	Business Concept	Attribute	Domain
(3)	name	RequestObject	Name	Jim, Jane
	age		Age	0-100
	message	ResponseObject	resultMessage	Baby, Error!

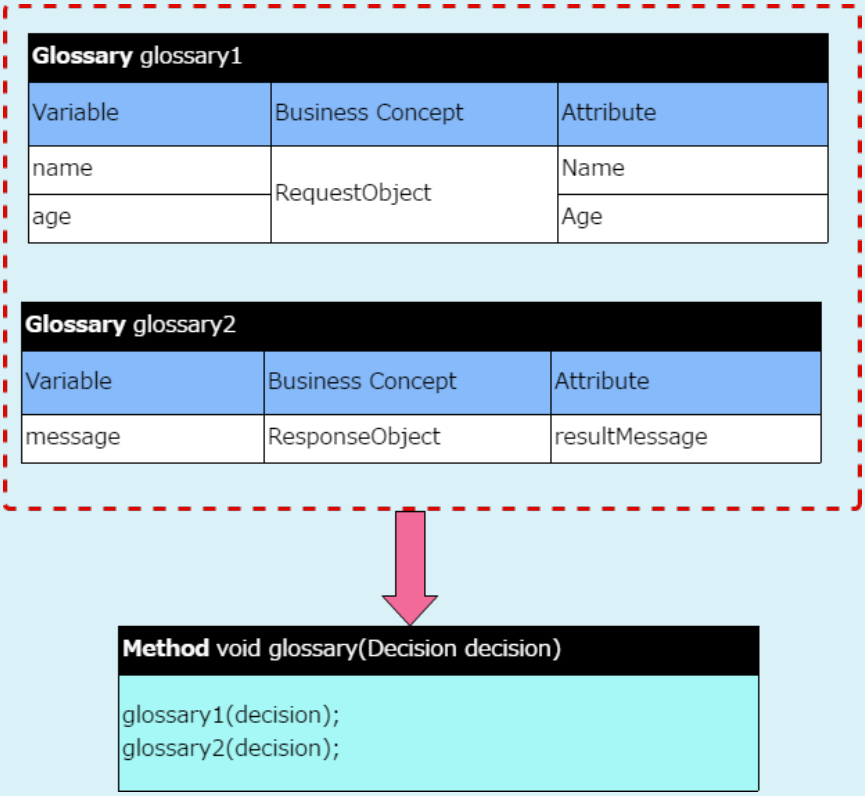
(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

- Glossary %テーブル名%
  - 「Glossary」の後に、半角スペースを入れてテーブル名を記述します。

コラム

Glossaryは、テーブル名を「glossary」と命名しない場合には、Glossary型のテーブルを結合するメソッドを記述する必要があります。
例えば、Glossary型のテーブルを「glossary1」「glossary2」と命名して利用したい場合には、以下の図のように記載してください。
なお、下記の場合、テーブル名を”glossary”としたGlossary型のテーブルの定義は不要です。



(2) サブヘッダに利用できるキーワード

サブヘッダには、「論理名（Variable）」・「グループ（Business Concept）」・「物理名（Attribute）」が必須項目です。
「値の範囲（Domain）」は任意項目です。
Glossaryでのサブヘッダは、漢字を含めて自由に変更することができますが、列の並び順（Variable → Business Concept → Attribute）は変更できません。
「論理名（Variable）」は *DecisionTable* で利用する項目名のため、ユーザにわかりやすい名称を設定できます。ひらがな、漢字などのマルチバイト文字も利用可能です。

- Glossaryに利用できるキーワード

キーワード	利用可否
Variable（Glossary）	○
Business Concept	○
Attribute	○
Domain	○

項目とデータ型の定義（Datatype）

Glossary で設定したグループ（Business Concept）、項目のデータ型を指定します。
この表は、グループ（Business Concept）単位に作成します。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	○

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。

(1)	**Datatype** RequestObject	
(2)	String	Name
	int	Age

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

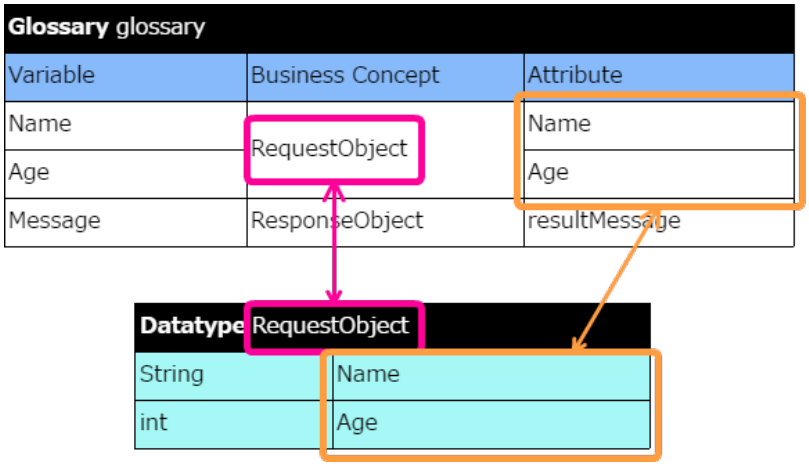
- Datatype %テーブル名%
  - 「Datatype」の後に、半角スペースを入れてテーブル名を記述します。
  - Datatypeのテーブル名は、 *Glossary* のBusiness Conceptとして定義した内容と一致させる必要があります。

サブヘッダに利用できるキーワード

Datatypeでは、サブヘッダはありません。

(2) 明細の記述方法

Datatypeは、以下の方法で *Glossary* の「Business Concept」（左から2番目の項目）の単位で記述します。
Datatypeに設定する項目は、 *Glossary* の「Attribute」（左から3番目の項目）です。



データ型に配列を含めたい場合には、以下のように記述します。

Datatype ResponseObject	
String	Name
String[]	Comments

- 利用できるデータ型
  Javaの基本データ型とユーザ定義のデータ型が利用できます。
  - Javaの基本データ型

タイプ
boolean
char
int
double
long
String (java.lang.String)
Date (java.util.Date)

  - ユーザ定義のデータ型

タイプ
Excelファイル上に設定したオブジェクト
String型の単一パラメータのPublicコンストラクタのあるJavaクラス

タイプ
上記のデータ型の1次元配列

> **注意**
>
> **技術的な制約事項**
>
> - Datatypeでの定義では、最初の項目はString、またはユーザ定義のデータ型にする必要があります。
> （String以外のJavaの基本のデータ型は設定できません。）
> ただし、この制約はDatatypeの表に限定した事項となるため、例えば数値項目しかない場合であっても、DatatypeやData(Variable)で代替の項目を設定します。
> この代替の項目は、Excelファイルでの制約となるため、データソース定義のパラメータには含めなくても問題ありません。

項目の初期値の定義（Data / Variable）

項目とデータ型の定義（Datatype） で定義した項目の初期値を指定します。
ここで設定する値は IM-BIS との連携時のインスタンス化に必要となるため、すべての項目に値を設定する必要があります。

> **注意**
>
> **処理対象者設定に利用する場合の注意**
>
> IM-BIS の動的処理対象者設定に利用する場合には、レスポンスのオブジェクトの定義で、キーワードを必ず「Variable」としてください。
> 「Data」とした場合には、エラーが発生するため、データソース定義として登録することができません。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	○

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
結合するセルの単位が同じであれば、列・行を入れ替えて記述することもできます。

- 横方向に項目を並べるパターン

(1)	**Data** RequestObject requestObj	
(2)	Name	Age
	Name	Age
(3)	John	10

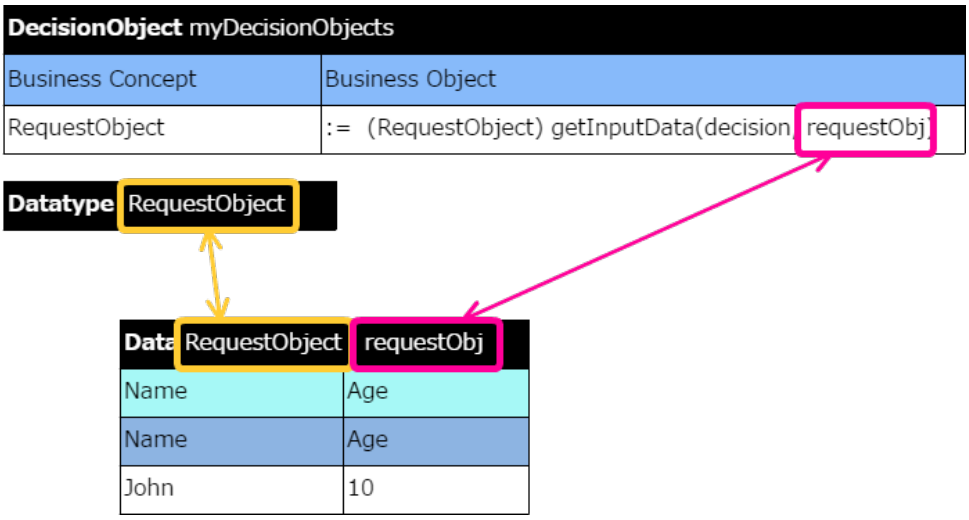
- 縦方向に項目を並べるパターン

(1)		
Data RequestObject requestObj		
Name	Name	John
Age	Age	10
(2)		(3)

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

- Data %テーブル名%
  - 「Data」または「Variable」の後に、半角スペースで区切って、BusinessConcept名、インスタンス名を記述します。
  - BusinessConcept名は、 *Glossary* の「Business Concept」（左から2番目の項目）にします。
  - インスタンス名は、 *DecisionObject* の入力値にします。
    - 入力のBusiness Conceptでは、getInputDataの2番目の引数を指定します。
    - 出力や内部処理のBusiness Conceptでは、記述した式の配列番号を除いた部分が該当します。
      （式が":= responseObj[0]"の場合、"responseObj"とします。）



(2) サブヘッダに利用できるキーワード

Dataのサブヘッダには、 *Glossary* の「Variable」と「Attribute」を記述します。

(3) 明細の記述方法

Dataの明細の値は、インスタンスを作成する処理の初期値に利用されますので、データ型にあわせて、適切な値を設定します。
セルを空にしたままの場合、インスタンスが正しく生成されないため、評価結果が返却されません。

メインヘッダを「Data」とした場合、1件以上入力します。
メインヘッダを「Variable」とした場合、1件のみ入力します。

オブジェクトのインスタンスの設定（DecisionObject）

Glossary で定義したオブジェクト（Business Concept）と IM-BIS （データマッパー）との値のマッピングを行います。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	○

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。

(1)	**DecisionObject** decisionObjects	
(2)	a Business Concept	b Business Object
(3)	RequestObject	:= (RequestObject) getInputData(decision, requestObj)
	ResponseObject	:= responseObj[0]

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

- DecisionObject decisionObjects
  - メインヘッダは、必ず上記のように記載してください。
    名前が異なる等定義に誤りがある場合には、エラーが発生します。

(2) サブヘッダに利用できるキーワード

DecisionObjectのサブヘッダには、「Business Concept」と「Business Object」を記述します。
Glossary と同様にヘッダの内容は、漢字を含めてわかりやすい名称に変更することができます。

a. Business Concept
   Glossary に定義した項目のグループを表す「Business Concept」の列です。

b. Business Object
「Business Concept」に対する値の入出力の式を定義するための列です。

(3) 明細の記述方法

明細の左の列は、 *Glossary* でグループ（Business Concept）として定義した名称を記述します。
明細の右の列は、値の受け渡し方法に応じて式を記述します。

- IM-BIS （データマッパー）から入力値を受け取る場合

```
:= ({Business Conceptの型})getInputData(decision, {Business Conceptのインスタンス名})
```

記述例

```
:= (RequestObject) getInputData(decision, requestObj)
```

- 入力以外の値の受け渡しを行う場合

```
:= {Business Conceptのインスタンス名}[0]
```

記述例

```
:= responseObj[0]
```

Javaのコードの定義（Method）

Javaのコードを利用したメソッドや関数を定義できる表です。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	○

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
結合するセルの単位が同じであれば、列・行を入れ替えて記述することもできます。

(1)	**Method** void DefineCurrentHour()
(2)	long hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY); setReal("Time", hour);

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

- Method %返却型% %メソッド名%
  - 「Method」の後に、半角スペースで区切って、返却型とメソッド名を記述します。
  - 返却型は、以下のようなパターンで使い分けます。
    - void
      - *Decision* で直接Methodを呼び出して実行する
      - Methodのコード内でAPIを利用して、variable（*Glossary* で定義している項目）に値を設定する
    - データ型（intやStringなど）
      - Methodのコード内でreturnを利用して、variable（*Glossary* で定義している項目）に値を設定する
      - *ConditionAny* などの条件で、演算子「Is True」「Is False」を利用している
        （この場合には、必ずboolean型で値を返却します。）

サブヘッダに利用できるキーワード

Methodには、サブヘッダはありません。

(2) 明細の記述方法

Methodでは、自由にJavaのコードを記述します。
Javaのパッケージに含まれるメソッドなども利用できますが、その場合には対象のパッケージをimportするための情報を *Environment* で定義してください。

環境設定情報（Environment）

ルールの実行に必要な他のExcelファイルやJavaのパッケージ等の情報を管理します。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	○

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。

(1)	**Environment**	
(2)	include	../lib/openrules.config/IntramartTemplate.xls
		Rule.xls
	import.java	java.util.Calendar

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

- Environment

サブヘッダに利用できるキーワード

Environmentには、サブヘッダはありません。

(2) 明細の記述方法

Environmentの明細は、参照するファイルやJavaのパッケージ等に合わせて、参照するためのキーワード（includeなど）と対象のファイルやパッケージ名などを記述します。

- 参照するためのキーワード
  - include
    Excelファイルで定義された内容を参照する場合のキーワードです。
  - import.java
    Javaで開発したパッケージを参照する場合のキーワードです。

    > **コラム**
    >
    > IM-BIS では、標準のテンプレート（IntramartTemplate.xls）で、以下のパッケージについては定義済みです。
    >
    > - java.util.ArrayList
    > - java.util.List

- 参照するためのファイルやパッケージ

  データソース定義に一緒にアップロードしたファイルの場合、ファイル名と拡張子を記述します。
  Javaパッケージの場合、プログラムのimport文と同様にパッケージ名を記述します。
  （パッケージ名の最後に、”;”の記述は不要です。）

条件として利用できるキーワード

Decision や *DecisionTable* で条件の設定に利用するキーワードです。

- Condition
- If
- ConditionBetween
- ConditionVarOperValue
- ConditionIntOperInt
- ConditionRealOperReal
- ConditionDateOperDate
- ConditionAny
- ConditionMap

Condition

演算子と比較する値を組み合わせて条件に設定することができます。

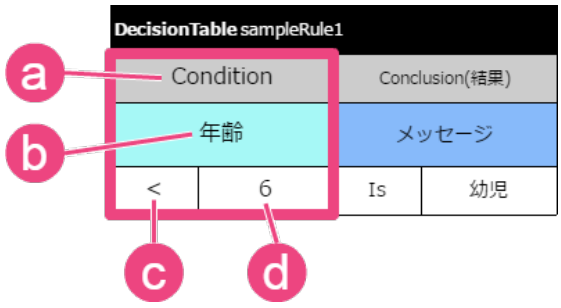
利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	○
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

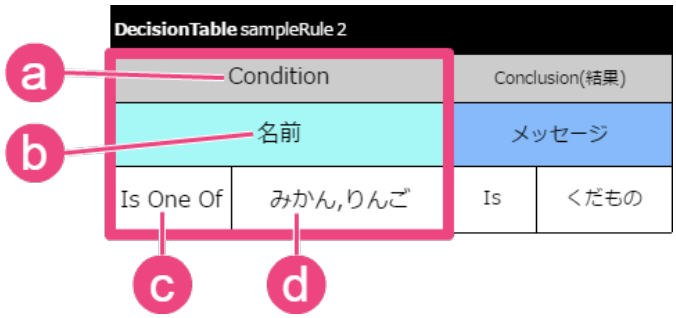
利用するTableTypeのサブヘッダ部に記述します。

- 単一の値を設定する例



  a. キーワード
  条件を表す OpenRules のキーワードです。
  b. 論理名
  Glossary で定義した項目の論理名です。
  c. 演算子
  利用できる演算子 です。
  キーワードやデータ型によって利用可否が異なります。
  d. 基準値
  演算子と組み合わせて右辺に設定される式や値です。

- 複数の値（配列）を設定する例



a. キーワード
　条件を表す OpenRules のキーワードです。
b. 論理名
　Glossary で定義した項目の論理名です。
c. 演算子
　利用できる演算子 です。
　キーワードやデータ型によって利用可否が異なります。
d. 基準値
　演算子と組み合わせて右辺に設定される式や値です。

利用できる演算子

演算子	別の記法	説明
Is	▪ = ▪ ==	条件では、単一の値と比較し、「～に等しい」を表します。
Is Not	▪ != ▪ isnot ▪ Is Not Equal To ▪ Not ▪ Not Equal ▪ Not Equal To	条件では、単一の値と比較し、「～に等しくない」を表します。
>	▪ Is More ▪ More ▪ Is More Than ▪ Is Greater ▪ Greater ▪ Is Greater Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より大きい」（設定した値を含まない）を表します。
>=	▪ Is More Or Equal ▪ Is More Or Equal To ▪ Is More Than Or Equal To ▪ Is Greater Or Equal To ▪ Is Greater Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以上」（設定した値を含む）を表します。
<=	▪ Is Less Or Equal ▪ Is Less Or Equal To ▪ Is Less Than Or Equal To ▪ Is Smaller Or Equal To ▪ Is Smaller Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以下」（設定した値を含む）を表します。
<	▪ Is Less ▪ Less ▪ Is Less Than ▪ Is Smaller ▪ Smaller ▪ Is Smaller Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より小さい」（設定した値を含まない）を表します。
Is Empty	なし	条件では、単一の値と比較し、「空の値」（値が「null」、または空白を含む値）を表します。
Contains	▪ Contain	条件では、文字型（String）の単一の値と比較し、「～を含む」（部分一致）を表します。 大文字・小文字は区別せずに比較します。

演算子	別の記法	説明
Does Not Contain	▪ DoesNotContain	条件では、文字型（String）の単一の値と比較し、「～を含まない」（部分一致）を表します。 大文字・小文字は区別せずに比較します。
Starts With	▪ Start with ▪ Start	条件では、文字型（String）の単一の値と比較し、「～から始まる」（前方一致）を表します。 大文字・小文字は区別せずに比較します。
Match	▪ Matches ▪ Is Like ▪ Like	条件では、文字型（String）の単一の値と比較し、正規表現で表したパターンと一致するかを表します。
No Match	▪ Not Matches ▪ Does Not Match ▪ Not Like ▪ Is Not Like ▪ Different ▪ Different From	条件では、文字型（String）の単一の値と比較し、正規表現で表したパターンと一致しないかを表します。
Within	▪ Inside ▪ Inside Interval ▪ Interval	条件では、整数型（integer）・実数（浮動小数点）型（real）の単一の値と比較し、「範囲内」（～から～の間）を表します。 下限値・上限値は、[0;9], (0;9], 0–9, between 5 and 10, more than 5 and less or equals 10 などの形式で記述します。 [0;9]、または0-9と書いた場合には、「0以上9以下」、[0;9)と書いた場合には、「0以上9未満」を表します。
Is One Of[1]	▪ Is One ▪ Is One Of Many ▪ Is Among ▪ Among	条件では、文字型（String）・整数型（integer）・実数（浮動小数点）型（real）の単一の値と比較し、カンマ区切りで表現した値のいずれかと一致するかを表します。
Is Not One Of[2]	▪ Is Not Among ▪ Not Among	条件では、文字型（String）・整数型（integer）・実数（浮動小数点）型（real）の単一の値と比較し、カンマ区切りで表現した値のいずれとも一致していないかを表します。
Include	▪ Include All	IM-BIS との連携では、利用できません。
Exclude	▪ Do Not Include ▪ Exclude One Of	IM-BIS との連携では、利用できません。
Does Not Include	▪ Include Not All	IM-BIS との連携では、利用できません。
Intersect	▪ Intersect With ▪ Intersects	IM-BIS との連携では、利用できません。

[1] 「Is One Of」は、 OpenRules の不具合のために6.4.0より下位バージョンでは、文字型（String）以外のデータ型には利用できません。

[2] 「Is Not One Of」は、 OpenRules の不具合のために6.4.0より下位バージョンでは、文字型（String）以外のデータ型には利用できません。

コラム

- **Condition / Within利用時の演算子の扱い**

  1つのDecision Tableについて、すべての条件が、”Is” または “Within” となる場合には、演算子列なしで定義することもできます。

- **数値を比較条件に利用する場合の記述方法**

  数値を比較条件に利用する場合には、記述方法によって合致する範囲が変わります。
  特定の数値を基準とした条件を記述する場合には、以下の表を参考にして記述してください。

記述方法	設定時に条件と合致したと評価される値
5	5と等しい場合のみ
[5,10]	5,6,7,8,9,10のいずれかの場合
5;10	5,6,7,8,9,10のいずれかの場合
[5;10)	5,6,7,8,9のいずれかの場合。10は含まない
5-10	5,6,7,8,9,10のいずれかの場合
5- 10	5,6,7,8,9,10のいずれかの場合
-5 - 20	-5以上、かつ20以下の場合
-5 - -20	範囲指定の左辺(-5)が右辺(-20)より大きくなるため、エラー
-5 - -2	-5以上、かつ-2以下の場合
from 5 to 20	5以上、かつ20以下の場合
less 5	5未満の場合（5を含まない）
less than 5	5未満の場合（5を含まない）
less or equals 5	5以下の場合（5を含む）
less or equal 5	5以下の場合（5を含む）
less or equals to 5	5以下の場合（5を含む）
smaller than 5	5未満の場合（5を含まない）
more 10	10より大きい場合（10を含まない）
more than 10	10より大きい場合（10を含まない）
10+	10以上の場合
> 10	10より大きい場合（10を含まない）
>= 10	10以上の場合（10を含む）
between 5 and 10	5,6,7,8,9,10のいずれかの場合
no less than 10	10以上の場合（10を含む）
no more than 5	5以下の場合（5を含む）
equals to 5	5と等しい場合
greater or equal than 5 and less than 10	5以上10未満の場合（5は含むが、10は含まない）
more than 5 less or equal than 10	5より大きく10以下の場合（5は含まないが、10を含む）
more than 5,111,111 and less or equal than 10,222,222	5,111,111 より大きく10,222,222以下の場合（5,111,111は含まないが、10,222,222を含む）
[5‘000;10‘000‘000)	5,000以上10,000,000未満の場合（5,000は含むが、10,000,000は含まない）
[5,000;10,000,000)	5,000以上10,000,000未満の場合（5,000は含むが、10,000,000は含まない）
(5;100,000,000]	5,000以上10,000,000以下の場合（5,000、10,000,000の両方とも含む）

浮動小数点型（double）の範囲を指定する場合には、「FromToDouble」という型を用いて記述します。
この型を利用した場合も、整数の範囲と同様に[2.7; 3.14)のように表します。

If

「If」は、If（Condition）に演算子を記述しなくてもよい特殊なキーワードです。

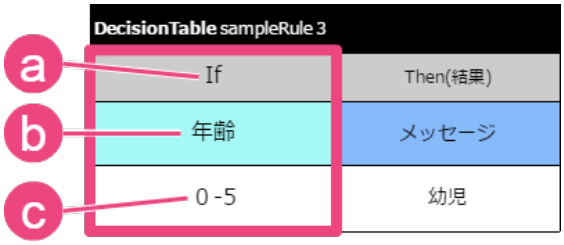
このキーワードは、 *Condition* と演算子”=”や”Within”を利用した場合と同義です。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

If を利用したい場合、以下の図のように記述します。



a. キーワード
　条件を表す OpenRules のキーワードです。

b. 論理名
　Glossary で定義した項目の論理名です。

c. 基準値
　演算子と組み合わせて右辺に設定される式や値です。
　単一の値の場合は *Condition* と”=”を組み合わせた条件と同義です。
　値の範囲の場合は *Condition* と”Within”の条件と同義です。

利用できる演算子

このキーワードでは、演算子を利用しません。

ConditionBetween

範囲内を表す条件として、下限値・上限値を組み合わせて条件に設定することができます。
ConditionBetweenを使う場合には、「下限値」 以上、「上限値」 以下 として評価します。

Condition で、演算子に「Within」を設定した場合と同義です。

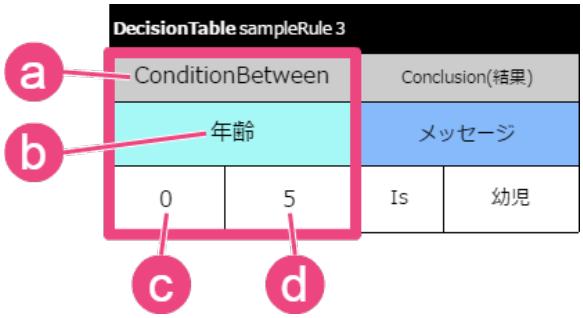
利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	×

テーブルタイプ	利用可否
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	×
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。



a. キーワード
　条件を表す OpenRules のキーワードです。

b. 論理名
　Glossary で定義した項目の論理名です。

c. 下限値
　条件の対象範囲の下限値です。

d. 上限値
　条件の対象範囲の上限値です。

利用できる演算子

このキーワードでは、演算子を利用しません。

ConditionVarOperValue

Glossaryに定義した項目と、特定の値の比較条件に設定することができます。

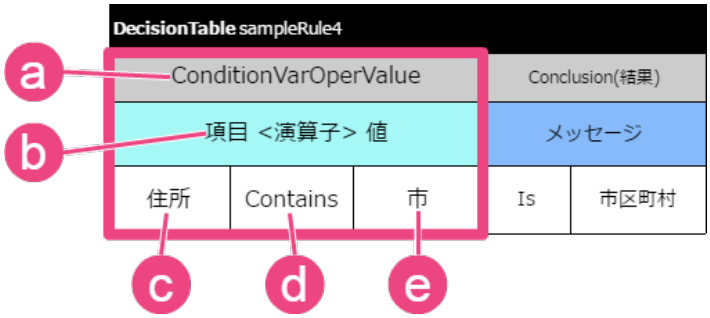
利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×

テーブルタイプ	利用可否
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。



a. キーワード
　条件を表す OpenRules のキーワードです。

b. 列ラベル
　条件式の説明など列を識別するための情報です。

c. 論理名
　Glossary で定義した項目の論理名です。

d. 演算子
　利用できる演算子 です。
　キーワードやデータ型によって利用可否が異なります。

e. 基準値
　演算子と組み合わせて右辺に設定される式や値です。

また、応用的な利用方法として、複数の項目に対応する条件を表すこともできます。
以下の例では、異なる項目の条件を1つの条件として記述しています。

DecisionTable sampleRule5

ConditionVarOperValue			Conclusion(結果)	
項目 <演算子> 値			保険プラン	
積載量	＞	2	Is	貨物：積載量(多)プラン
積載量	＜=	2	Is	貨物：積載量(少)プラン
排気量	＞	250	Is	二輪：大型プラン
排気量	Within	125-250	Is	二輪：中型プラン
排気量	＜	125	Is	二輪：小型プラン

利用できる演算子

このキーワードでは、 *Condition* と同様に演算子を利用することができます。

演算子	別の記法	説明

演算子	別の記法	説明
Is	▪ = ▪ ==	条件では、単一の値と比較し、「～に等しい」を表します。
Is Not	▪ != ▪ isnot ▪ Is Not Equal To ▪ Not ▪ Not Equal ▪ Not Equal To	条件では、単一の値と比較し、「～に等しくない」を表します。
>	▪ Is More ▪ More ▪ Is More Than ▪ Is Greater ▪ Greater ▪ Is Greater Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より大きい」（設定した値を含まない）を表します。
>=	▪ Is More Or Equal ▪ Is More Or Equal To ▪ Is More Than Or Equal To ▪ Is Greater Or Equal To ▪ Is Greater Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以上」（設定した値を含む）を表します。
<=	▪ Is Less Or Equal ▪ Is Less Or Equal To ▪ Is Less Than Or Equal To ▪ Is Smaller Or Equal To ▪ Is Smaller Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以下」（設定した値を含む）を表します。
<	▪ Is Less ▪ Less ▪ Is Less Than ▪ Is Smaller ▪ Smaller ▪ Is Smaller Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より小さい」（設定した値を含まない）を表します。
Is Empty	なし	条件では、単一の値と比較し、「空の値」（値が「null」、または空白を含む値）を表します。
Contains	▪ Contain	条件では、文字型（String）の単一の値と比較し、「～を含む」（部分一致）を表します。 大文字・小文字は区別せずに比較します。
Starts With	▪ Start with ▪ Start	条件では、文字型（String）の単一の値と比較し、「～から始まる」（前方一致）を表します。 大文字・小文字は区別せずに比較します。
Match	▪ Matches ▪ Is Like ▪ Like	条件では、文字型（String）の単一の値と比較し、正規表現で表したパターンと一致するかを表します。
No Match	▪ Not Matches ▪ Does Not Match ▪ Not Like ▪ Is Not Like ▪ Different ▪ Different From	条件では、文字型（String）の単一の値と比較し、正規表現で表したパターンと一致しないかを表します。
Within	▪ Inside ▪ Inside Interval ▪ Interval	条件では、整数型（integer）・実数（浮動小数点）型（real）の単一の値と比較し、「範囲内」（～から～の間）を表します。 下限値・上限値は、[0;9], (0;9], 0–9, between 5 and 10, more than 5 and less or equals 10 などの形式で記述します。 [0;9]、または0-9と書いた場合には、「0以上9以下」、[0;9)と書いた場合には、「0以上9未満」を表します。

演算子	別の記法	説明
Is One Of[3]	▪ Is One ▪ Is One Of Many ▪ Is Among ▪ Among	条件では、文字型（String）・整数型（integer）・実数（浮動小数点）型（real）の単一の値と比較し、カンマ区切りで表現した値のいずれかと一致するかを表します。
Is Not One Of	▪ Is Not Among ▪ Not Among	条件では、文字型（String）・整数型（integer）・実数（浮動小数点）型（real）の単一の値と比較し、カンマ区切りで表現した値のいずれとも一致していないかを表します。
Include	▪ Include All	IM-BIS との連携では、利用できません。
Exclude	▪ Do Not Include ▪ Exclude One Of	IM-BIS との連携では、利用できません。
Does Not Include	▪ Include Not All	IM-BIS との連携では、利用できません。
Intersect	▪ Intersect With ▪ Intersects	IM-BIS との連携では、利用できません。

[3] 「Is One Of」は、 OpenRules の不具合のために6.4.0より下位バージョンでは、文字型（String）以外のデータ型には利用できません。

[4] 「Is Not One Of」は、 OpenRules の不具合のために6.4.0より下位バージョンでは、文字型（String）以外のデータ型には利用できません。

ConditionIntOperInt

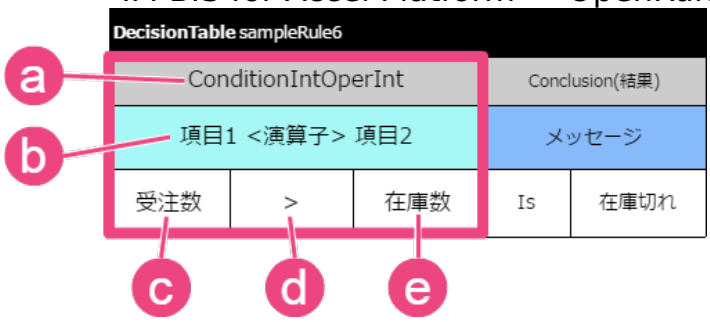
Glossaryに定義した整数型（int）の項目同士の比較条件に設定することができます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。



a. キーワード
条件を表す OpenRules のキーワードです。

b. 列ラベル
条件式の説明など列を識別するための情報です。

c. 論理名（左辺）
Glossary で定義した項目の論理名です。
データ型が整数型（integer）の項目に利用できます。
条件式の左辺に設定されます。

d. 演算子
利用できる演算子 です。
キーワードやデータ型によって利用可否が異なります。

e. 論理名（右辺）
Glossary で定義した項目の論理名です。
データ型が整数型（integer）の項目に利用できます。
条件式の右辺に設定されます。

利用できる演算子

このキーワードでは、 *Condition* で利用できる演算子のうち、整数型（int）に対応している演算子を利用することができます。

演算子	別の記法	説明
Is	▪ = ▪ ==	条件では、単一の値と比較し、「～に等しい」を表します。
Is Not	▪ != ▪ isnot ▪ Is Not Equal To ▪ Not ▪ Not Equal ▪ Not Equal To	条件では、単一の値と比較し、「～に等しくない」を表します。
>	▪ Is More ▪ More ▪ Is More Than ▪ Is Greater ▪ Greater ▪ Is Greater Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より大きい」（設定した値を含まない）を表します。
>=	▪ Is More Or Equal ▪ Is More Or Equal To ▪ Is More Than Or Equal To ▪ Is Greater Or Equal To ▪ Is Greater Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以上」（設定した値を含む）を表します。
<=	▪ Is Less Or Equal ▪ Is Less Or Equal To ▪ Is Less Than Or Equal To ▪ Is Smaller Or Equal To ▪ Is Smaller Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以下」（設定した値を含む）を表します。

演算子	別の記法	説明
<	▪ Is Less ▪ Less ▪ Is Less Than ▪ Is Smaller ▪ Smaller ▪ Is Smaller Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より小さい」（設定した値を含まない）を表します。
Within	▪ Inside ▪ Inside Interval ▪ Interval	条件では、整数型（integer）・実数（浮動小数点）型（real）の単一の値と比較し、「範囲内」（～から～の間）を表します。 下限値・上限値は、[0;9], (0;9], 0–9, between 5 and 10, more than 5 and less or equals 10 などの形式で記述します。 [0;9]、または0-9と書いた場合には、「0以上9以下」、[0;9)と書いた場合には、「0以上9未満」を表します。

ConditionRealOperReal

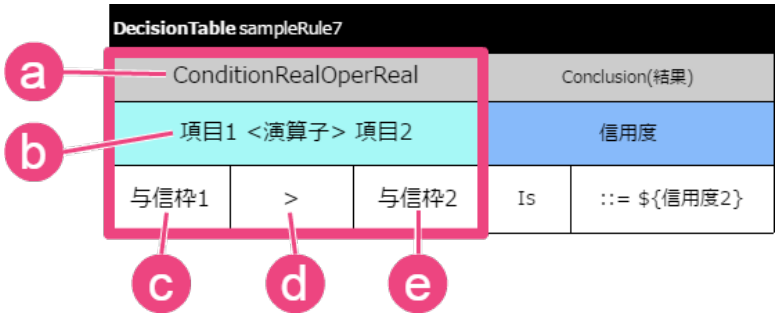
Glossaryに定義した実数型（real）の項目同士の比較条件に設定することができます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。



a. キーワード
　条件を表す OpenRules のキーワードです。

b. 列ラベル
　条件式の説明など列を識別するための情報です。

c. 論理名（左辺）
　Glossary で定義した項目の論理名です。
　データ型が実数（浮動小数点）型（real）の項目に利用できます。

条件式の左辺に設定されます。

d. 演算子
利用できる演算子 です。
キーワードやデータ型によって利用可否が異なります。

e. 論理名（右辺）
Glossary で定義した項目の論理名です。
データ型が実数（浮動小数点）型（real）の項目に利用できます。
条件式の右辺に設定されます。

利用できる演算子

このキーワードでは、 *Condition* で利用できる演算子のうち、実数型（real）に対応している演算子を利用することができます。

演算子	別の記法	説明
Is	▪ = ▪ ==	条件では、単一の値と比較し、「～に等しい」を表します。
Is Not	▪ != ▪ isnot ▪ Is Not Equal To ▪ Not ▪ Not Equal ▪ Not Equal To	条件では、単一の値と比較し、「～に等しくない」を表します。
>	▪ Is More ▪ More ▪ Is More Than ▪ Is Greater ▪ Greater ▪ Is Greater Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より大きい」（設定した値を含まない）を表します。
>=	▪ Is More Or Equal ▪ Is More Or Equal To ▪ Is More Than Or Equal To ▪ Is Greater Or Equal To ▪ Is Greater Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以上」（設定した値を含む）を表します。
<=	▪ Is Less Or Equal ▪ Is Less Or Equal To ▪ Is Less Than Or Equal To ▪ Is Smaller Or Equal To ▪ Is Smaller Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以下」（設定した値を含む）を表します。
<	▪ Is Less ▪ Less ▪ Is Less Than ▪ Is Smaller ▪ Smaller ▪ Is Smaller Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より小さい」（設定した値を含まない）を表します。
Within	▪ Inside ▪ Inside Interval ▪ Interval	条件では、整数型（integer）・実数（浮動小数点）型（real）の単一の値と比較し、「範囲内」（～から～の間）を表します。 下限値・上限値は、[0;9], (0;9], 0–9, between 5 and 10, more than 5 and less or equals 10 などの形式で記述します。 [0;9]、または0-9と書いた場合には、「0以上9以下」、[0;9)と書いた場合には、「0以上9未満」を表します。

ConditionDateOperDate

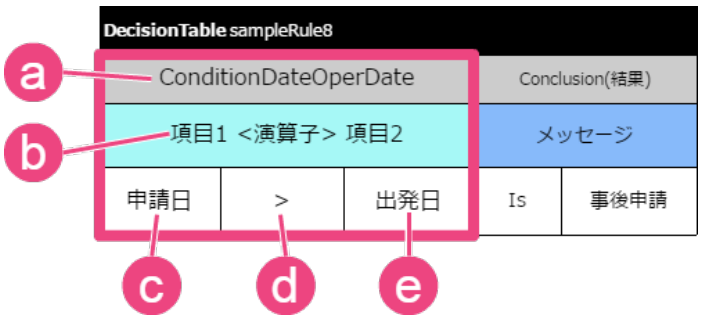
Glossaryに定義した日付型（Date）の項目同士の比較条件に設定することができます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。



a. キーワード
　条件を表す OpenRules のキーワードです。

b. 列ラベル
　条件式の説明など列を識別するための情報です。

c. 論理名（左辺）
　Glossary で定義した項目の論理名です。
　データ型が日付型（Date）の項目に利用できます。
　条件式の左辺に設定されます。

d. 演算子
　利用できる演算子 です。
　キーワードやデータ型によって利用可否が異なります。

e. 論理名（右辺）
　Glossary で定義した項目の論理名です。
　データ型が日付型（Date）の項目に利用できます。
　条件式の右辺に設定されます。

利用できる演算子

このキーワードでは、 *Condition* で利用できる演算子のうち、日付型（Date）に対応している演算子を利用することができます。

演算子	別の記法	説明
Is	▪ = ▪ ==	条件では、単一の値と比較し、「～に等しい」を表します。

演算子	別の記法	説明
Is Not	▪ != ▪ isnot ▪ Is Not Equal To ▪ Not ▪ Not Equal ▪ Not Equal To	条件では、単一の値と比較し、「～に等しくない」を表します。
>	▪ Is More ▪ More ▪ Is More Than ▪ Is Greater ▪ Greater ▪ Is Greater Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より大きい」（設定した値を含まない）を表します。
>=	▪ Is More Or Equal ▪ Is More Or Equal To ▪ Is More Than Or Equal To ▪ Is Greater Or Equal To ▪ Is Greater Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以上」（設定した値を含む）を表します。
<=	▪ Is Less Or Equal ▪ Is Less Or Equal To ▪ Is Less Than Or Equal To ▪ Is Smaller Or Equal To ▪ Is Smaller Than Or Equal To	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～以下」（設定した値を含む）を表します。
<	▪ Is Less ▪ Less ▪ Is Less Than ▪ Is Smaller ▪ Smaller ▪ Is Smaller Than	条件では、整数型（integer）・実数（浮動小数点）型（real）・日付型（Date）の単一の値と比較し、「～より小さい」（設定した値を含まない）を表します。

ConditionAny

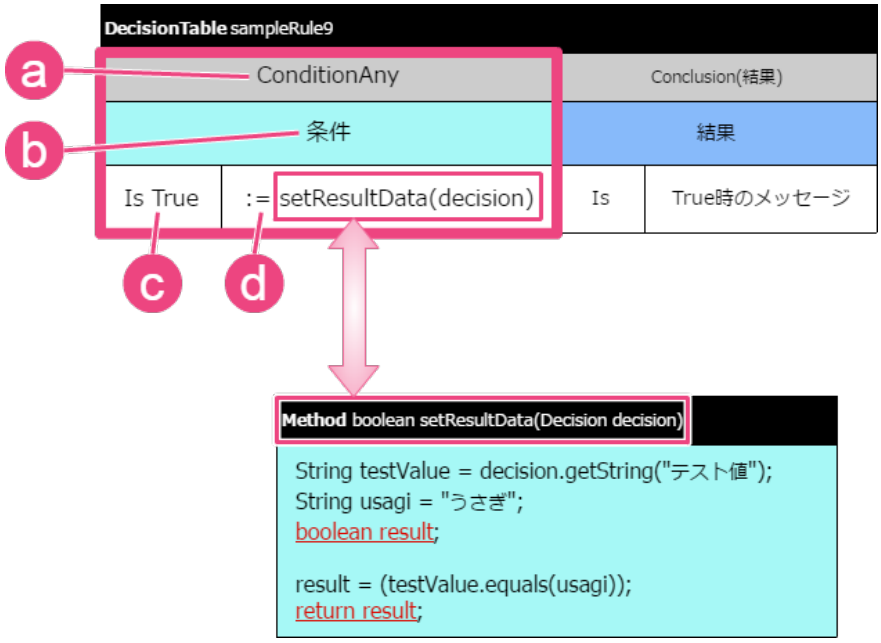
条件に式や *Method* の処理結果を直接指定できます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	○
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。
演算子として記述する「Is True（真の場合）」「Is False（偽の場合）」にそれぞれ実行したい処理を記述します。



a. キーワード
　条件を表す OpenRules のキーワードです。

b. 列ラベル
　条件式の説明など列を識別するための情報です。

c. 演算子
　利用できる演算子 です。
　“Is True”または”Is False”のみ利用できます。

d. 式
　条件を評価するための式です。
　処理内容は *Method* で定義しますが、 *Method* から返却される値はboolean型にする必要があります。
　（図中のmethodの赤下線部を参照してください。）

利用できる演算子

演算子	別の記法	説明
Is True	なし	条件では、記述した式の結果と比較し、「真の場合の処理」を表します。
Is False	なし	条件では、記述した式の結果と比較し、「偽の場合の処理」を表します。

ConditionMap

HashMap型の項目に対する条件を記述することができます。
IM-BIS との値の受け渡しでは、HashMap型を利用することはできません。

結果・処理として利用できるキーワード

Decision や *DecisionTable* で評価（処理）の設定に利用するキーワードです。

- Conclusion
- Then
- ConclusionVarOperValue
- ActionAny
- ActionMap
- ActionPrint
- ActionExecute
- Action
- Message
- ActionIterate
- ActionRulesOnArray
- ActionSort

Conclusion

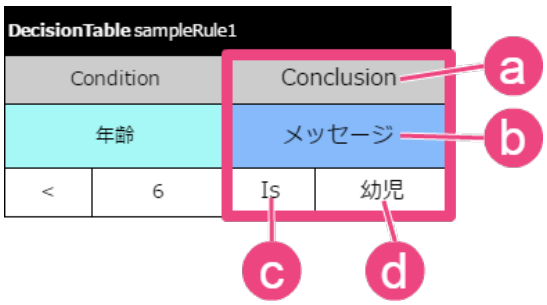
演算子と比較する値を組み合わせて返却する値（評価結果）に設定することができます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。



a. キーワード
結果を表す OpenRules のキーワードです。

b. 論理名
Glossary で定義した項目の論理名です。

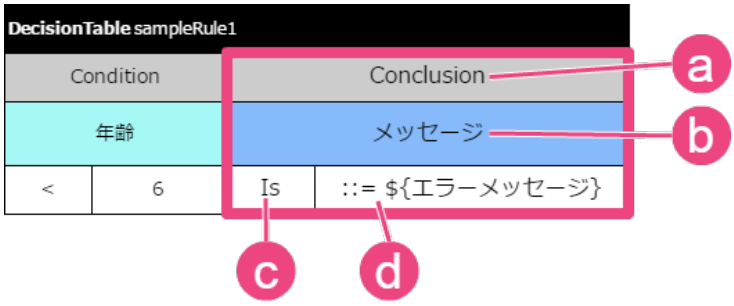
c. 演算子
利用できる演算子 です。
キーワードやデータ型によって利用可否が異なります。

d. 返却値

演算子と組み合わせて条件に合致した場合に返却する値です。

Isと組み合わせて、 *Methodで利用できるキーワード（API）* を記述することもできます。

- 式を設定する例



a. キーワード
結果を表す OpenRules のキーワードです。

b. 論理名
Glossary で定義した項目の論理名です。

c. 演算子
利用できる演算子 です。
キーワードやデータ型によって利用可否が異なります。

d. 式
条件に合致した場合に実行するマクロやメソッドなどを表す式です。
最初に”::=”をつける必要があります。

- カンマ区切りで設定する例

a. キーワード
結果を表す OpenRules のキーワードです。

b. 論理名
Glossary で定義した項目の論理名です。

c. 演算子
利用できる演算子 です。
カンマ区切りで複数の値を返却する場合は”Are”にする必要があります。

d. 返却値
演算子と組み合わせて条件に合致した場合に返却する値です。
カンマ区切りで返却する値を複数記述し、グリッドテーブルに返却した場合には、テーブルの各行の値にセットされます。

利用できる演算子

演算子	別の記法	説明
Is	= ==	対象の項目の値に1つの値を設定します。
Are	なし	カンマ区切りで複数の値を配列型で、対象の項目の値に代入します。 テーブル系アイテムに利用した場合、複数の行に返却することができます。
Add	なし	IM-BIS との連携では、利用できません。
Assign Plus	+=	IM-BIS との連携では、利用できません。

演算子	別の記法	説明
Assign Minus	-=	IM-BIS との連携では、利用できません。
Assign Multiply	*=	IM-BIS との連携では、利用できません。
Assign Divide	/=	IM-BIS との連携では、利用できません。

コラム

- **結果の値のセルに設定する内容について**

  結果（Conclusion）の値に設定する値は、”:=”などを付与せずに記述した場合、書かれている文字列（数値）をそのまま対象の項目の値にセットします。
  他の項目の値をセットする場合には、以下のように記述します。

  ```
  ::= ${項目の論理名}
  ```

  “$”は OpenRules のマクロです。
  データ型によって使い分ける必要がありますので、詳細は *Methodで利用できるキーワード（API）* を参照してください。

Then

「Then」は、Then（Conclusion）に演算子を記述しなくてもよい特殊なキーワードです。
このキーワードは、 *Conclusion* と演算子”=”を利用した場合と同義です。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

Then を利用したい場合、以下の図のように記述します。

a. キーワード
   結果を表す OpenRules のキーワードです。

b. 論理名
   Glossary で定義した項目の論理名です。

c. 返却値

条件に合致した場合に返却する値です。

利用できる演算子

このキーワードでは、演算子を利用しません。

ConclusionVarOperValue

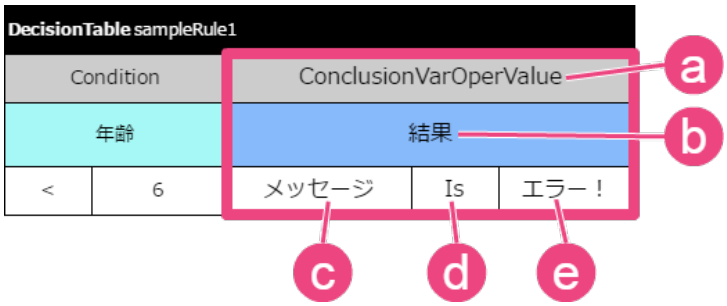
演算子と比較する値を組み合わせて返却する値（評価結果）に設定することができます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。



a. キーワード
結果を表す OpenRules のキーワードです。

b. 列ラベル
返却内容の説明など列を識別するための情報です。

c. 論理名
Glossary で定義した項目の論理名です。
(e) で設定した値のセット対象です。

d. 演算子
利用できる演算子 です。
キーワードやデータ型によって利用可否が異なります。

e. 返却値
(e) で設定した項目にセットする値です。

利用できる演算子

演算子	別の記法	説明
Is	= ==	対象の項目の値に1つの値を設定します。
Are	なし	カンマ区切りで複数の値を配列型で、対象の項目の値に代入します。 テーブル系アイテムに利用した場合、複数の行に返却することができます。
Add	なし	IM-BIS との連携では、利用できません。
Assign Plus	+=	IM-BIS との連携では、利用できません。
Assign Minus	-=	IM-BIS との連携では、利用できません。
Assign Multiply	*=	IM-BIS との連携では、利用できません。
Assign Divide	/=	IM-BIS との連携では、利用できません。

ActionAny

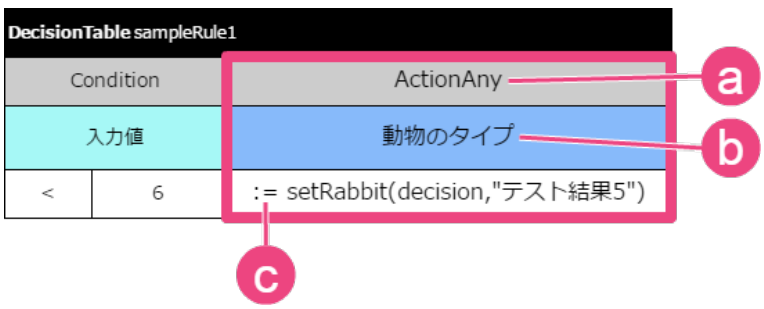
定義済みのJavaやExcelの関数やメソッドを呼び出して実行することができます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	○
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。



a. キーワード
　結果を表す OpenRules のキーワードです。

b. 列ラベル
　返却内容の説明など列を識別するための情報です。

c. 式
　条件に合致した場合に実行するマクロやメソッドなどを表す式です。
　最初に":="をつける必要があります。

利用できる演算子

演算子は記述できません。

ActionMap

HashMap型の項目に対する評価（処理）を記述することができます。
IM-BIS との値の受け渡しでは、HashMap型を利用することはできません。

ActionPrint

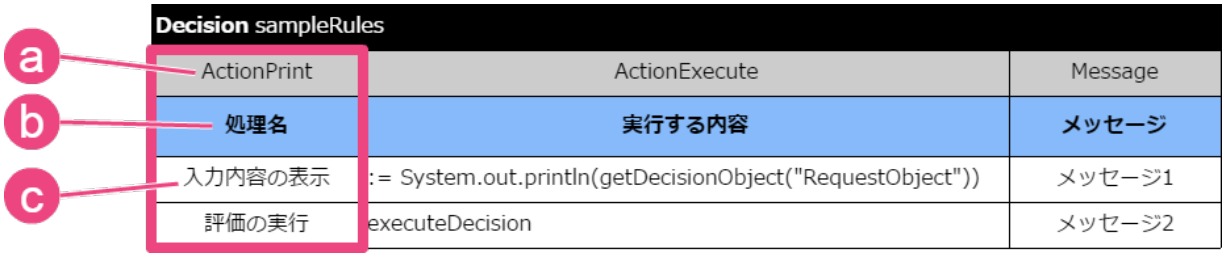
実行する処理名をコンソールに出力します。
ログには出力されません。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	×
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	×
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	×
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	○
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。
ActionPrintは *Decision* でのみ記述できます。（必須項目）



Decision sampleRules		
ActionPrint	ActionExecute	Message
処理名	**実行する内容**	**メッセージ**
入力内容の表示	:= System.out.println(getDecisionObject("RequestObject"))	メッセージ1
評価の実行	executeDecision	メッセージ2

a. キーワード
　コンソールへの出力を表す OpenRules のキーワードです。

b. 列ラベル
　条件式の説明など列を識別するための情報です。

c. 出力内容
　コンソールに出力する内容です。

利用できる演算子

演算子は記述できません。

実行イメージ

ActionPrintで設定した内容は、以下のような形でコンソールに出力されます。

Decision でActionPrintを利用した場合のコンソールへの出力例は以下の通りです。

- *Decision* では、ActionPrintに処理名を設定します。

Decision sampleRules

ActionPrint	ActionExecute	Message
処理名	**実行する内容**	**メッセージ**
入力内容の表示	:= System.out.println(getDecisionObject("RequestObject"))	Message from Decision 1
評価の実行	executeDecision	Message from Decision 2
結果の取得	:= decision().put("ResponseObject", responseObj)	Message from Decision 3
出力内容の表示	:= System.out.println(getDecisionObject("ResponseObject"))	Message from Decision 4

- コンソールの出力内容です。

```
[INFO] o.o.u.Log - [] IMPORT.JAVA=org.openl.types.impl.DynamicObject
[INFO] o.o.u.Log - [] *** Decision sampleRules ***
[INFO] o.o.u.Log - [] Decision has been initialized
[INFO] o.o.u.Log - [] Decision Run has been initialized
[INFO] o.o.u.Log - [] Decision sampleRules: 入力内容の表示
RequestObject(id=0) {
  Age=10
  Name=太郎
}
[INFO] o.o.u.Log - [] sampleRules: Message from Decision 1
[INFO] o.o.u.Log - [] Decision sampleRules: 評価の実行
[INFO] o.o.u.Log - [] Conclusion: 年齢区分 Is 小学生
[INFO] o.o.u.Log - [] Message from DT 3 [produced by executeDecision]
[INFO] o.o.u.Log - [] sampleRules: Message from Decision 2
[INFO] o.o.u.Log - [] Decision sampleRules: 結果の取得
[INFO] o.o.u.Log - [] sampleRules: Message from Decision 3
[INFO] o.o.u.Log - [] Decision sampleRules: 出力内容の表示
ResponseObject(id=0) {
  responseString=小学生
}
[INFO] o.o.u.Log - [] sampleRules: Message from Decision 4
[INFO] o.o.u.Log - [] Decision has been finalized
```

1. *Decision* の明細1行目で設定したメッセージです。
2. *Decision* の明細2行目で設定したメッセージです。
3. *Decision* の明細3行目で設定したメッセージです。
4. *Decision* の明細4行目で設定したメッセージです。

ActionExecute

実行する *Decision* や *DecisionTable* の名前を記述します。
上から順に *Decision* や *DecisionTable* を実行します。
Decision から *Decision* をサブデシジョンとして呼び出すこともできます。

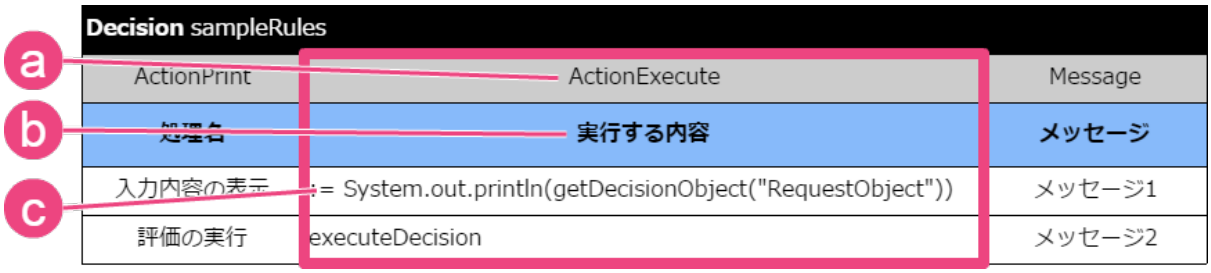
利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	○（必須）
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×

テーブルタイプ	利用可否
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。
ActionExecuteは *Decision* の必須項目です。

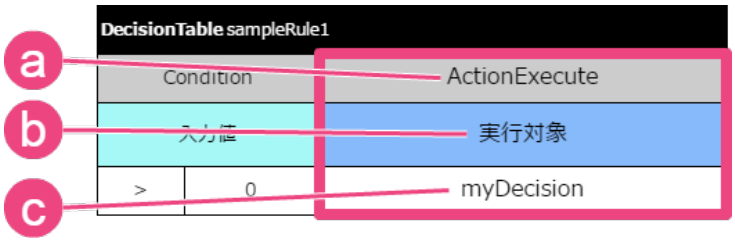


a. キーワード
Decision によるルールの実行などの処理を表す OpenRules のキーワードです。

b. 列ラベル
各処理の説明など列を識別するための情報です。

c. 実行内容
実行する *Decision* や *Method* など実際の処理と順番を表します。
上から順に設定された処理を実行します。

DecisionTable での利用例です。
以下の例では *DecisionTable* からサブデシジョンの呼び出しを行っています。



a. キーワード
Decision によるルールの実行などの処理を表す OpenRules のキーワードです。

b. 列ラベル
各処理の説明など列を識別するための情報です。

c. 実行内容（サブデシジョンなど）
DecisionTable から呼び出す *Decision* 、または他の *DecisionTable* の名称です。

利用できる演算子

演算子は記述できません。

Action

特定の項目に返却する値（評価結果）を設定することができます。

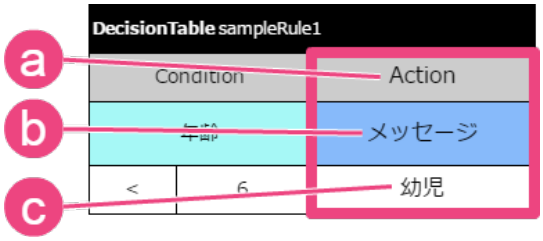
利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×

テーブルタイプ	利用可否
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。



a. キーワード
(b) 論理名の項目への値の代入を表す OpenRules のキーワードです。

b. 論理名
Glossary で定義した項目の論理名です。

c. 返却値
(b) 論理名の項目に代入する値です。

利用できる演算子

演算子は記述できません。

Message

任意に設定した文字列をコンソールに出力します。
ログには出力されません。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
変数への値割り当て（DecisionTableAssign）	×
ルールの反復処理（DecisionTableIterate）	×
スコアに基づくソート処理（DecisionTableSort）	×
処理順の設定（Decision）	○
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×

テーブルタイプ	利用可否
環境設定情報（Environment）	×

記述方法

利用するTableTypeのサブヘッダ部に記述します。

- *Decision* の場合

Decision sampleRules		
ActionPrint	ActionExecute	Message
処理名	**実行する内容**	**メッセージ**
入力内容の表示	:= System.out.println(getDecisionObject("RequestObject"))	メッセージ1

a. キーワード
　コンソールへのメッセージの出力を表す OpenRules のキーワードです。
b. 列ラベル
　条件式の説明など列を識別するための情報です。
c. メッセージ内容
　該当の処理実行時にコンソールに出力する内容です。

- *DecisionTable* の場合

DecisionTable executeDecision				
Condition		Conclusion		Message
年齢		メッセージ		コンソール出力
<	0	=	エラー！	DTメッセージ1
<	6	=	幼児	DTメッセージ2
<	13	=	小学生	DTメッセージ3

a. キーワード
　コンソールへのメッセージの出力を表す OpenRules のキーワードです。
b. 列ラベル
　条件式の説明など列を識別するための情報です。
c. メッセージ内容
　該当の処理実行時にコンソールに出力する内容です。

利用できる演算子

演算子は記述できません。

実行イメージ

Messageで設定した内容は、コンソールに出力されます。

Decision と *DecisionTable* でMessageを利用した場合のコンソールへの出力例は以下の通りです。

- *Decision* では、Messageに「Message from Decision」と設定します。

Decision sampleRules		
ActionPrint	ActionExecute	Message
処理名	**実行する内容**	**メッセージ**
入力内容の表示	:= System.out.println(getDecisionObject("RequestObject"))	Message from Decision 1
評価の実行	executeDecision	Message from Decision 2
結果の取得	:= decision().put("ResponseObject", responseObj)	Message from Decision 3
出力内容の表示	:= System.out.println(getDecisionObject("ResponseObject"))	Message from Decision 4

- *DecisionTable* では、Messageに「Message from DT」と設定します。

DecisionTable executeDecision

Condition		Conclusion		Message
年齢		メッセージ		コンソール出力
<	0	=	エラー！	Message from DT 1
<	6	=	幼児	Message from DT 2
<	13	=	小学生	Message from DT 3

- コンソールの出力内容です。

```
[INFO] o.o.u.Log - [] IMPORT.JAVA=org.openl.types.impl.DynamicObject
[INFO] o.o.u.Log - [] *** Decision sampleRules ***
[INFO] o.o.u.Log - [] Decision has been initialized
[INFO] o.o.u.Log - [] Decision Run has been initialized
[INFO] o.o.u.Log - [] Decision sampleRules: 入力内容の表示
RequestObject(id=0) {
  Age=10
  Name=太郎
}
(1) [INFO] o.o.u.Log - [] sampleRules: Message from Decision 1
[INFO] o.o.u.Log - [] Decision sampleRules: 評価の実行
[INFO] o.o.u.Log - [] Conclusion: 年齢区分 Is 小学生
(2) [INFO] o.o.u.Log - [] Message from DT 3 [produced by executeDecision]
(3) [INFO] o.o.u.Log - [] sampleRules: Message from Decision 2
[INFO] o.o.u.Log - [] Decision sampleRules: 結果の取得
(4) [INFO] o.o.u.Log - [] sampleRules: Message from Decision 3
[INFO] o.o.u.Log - [] Decision sampleRules: 出力内容の表示
ResponseObject(id=0) {
  responseString=小学生
}
(5) [INFO] o.o.u.Log - [] sampleRules: Message from Decision 4
[INFO] o.o.u.Log - [] Decision has been finalized
```

1. *Decision* の明細1行目で設定したメッセージです。
2. *DecisionTable* に設定したメッセージのうち、条件に合致した明細3行目のメッセージです。
3. *Decision* の明細2行目で設定したメッセージです。
4. *Decision* の明細3行目で設定したメッセージです。
5. *Decision* の明細4行目で設定したメッセージです。

Decision で設定した場合には、「（decision名）：（Messageの内容）」という形式で出力されます。
DecisionTable で設定した場合には、条件に合致した行のMessageが「（Messageの内容）[produced by （Decision Table名）]」という形式で出力されます。

コラム

上の例で *DecisionTable* で設定したメッセージは”Rule ID”と組み合わせて、以下の形式で記述することもできます。
（$RULE_IDは、 OpenRules 6.3.0以降で利用できます。）

- *DecisionTable* に「Rule ID」列を追加し、 *Message* は1つの結合セルでまとめます。
  キーワードとして”#”を指定すると、 *DecisionTable* の各行を識別するIDの列として扱われます。

DecisionTable executeDecision

#	Condition		Conclusion		Message
Rule Id	年齢		メッセージ		コンソール出力
Rule 1	<	0	=	エラー！	<$RULE_ID> was executed.
Rule 2	<	6	=	幼児	
Rule 3	<	13	=	小学生	

上記の *DecisionTable* を実行すると”$RULE_ID”に”Rule ID”を代入してコンソールに出力されます。
例えば、2行目の条件（Rule 2）に合致した場合は以下のように出力されます。

```
[INFO] o.o.u.Log - [] IMPORT.JAVA=org.openl.types.impl.DynamicObject
[INFO] o.o.u.Log - [] *** Decision sampleRules ***
[INFO] o.o.u.Log - [] Decision has been initialized
[INFO] o.o.u.Log - [] Decision Run has been initialized
[INFO] o.o.u.Log - [] Decision sampleRules: 入力内容の表示
RequestObject(id=0) {
  Age=10
  Name=太郎
}
[INFO] o.o.u.Log - [] sampleRules: Message from Decision 1
[INFO] o.o.u.Log - [] Decision sampleRules: 評価の実行
[INFO] o.o.u.Log - [] Conclusion: 年齢区分 Is 小学生
[INFO] o.o.u.Log - [] <Rule 2> was executed. [produced by executeDecision]
[INFO] o.o.u.Log - [] sampleRules: Message from Decision 2
[INFO] o.o.u.Log - [] Decision sampleRules: 結果の取得
[INFO] o.o.u.Log - [] sampleRules: Message from Decision 3
[INFO] o.o.u.Log - [] Decision sampleRules: 出力内容の表示
ResponseObject(id=0) {
  responseString=小学生
}
[INFO] o.o.u.Log - [] sampleRules: Message from Decision 4
[INFO] o.o.u.Log - [] Decision has been finalized
```

ActionIterate

オブジェクトの配列を受け取って配列中のオブジェクト1件ずつに対して、別の *DecisionTable* で定義したルールの評価を繰り返し実行することができます。
このキーワードは、 IM-BIS との連携では利用できません。

ActionRulesOnArray

配列型のデータを受け取って配列中のデータ1件ずつに対して、 *DecisionTable* などで定義したルールの評価を実行することができます。
このキーワードは、 IM-BIS との連携では利用できません。

ActionSort

オブジェクトの配列を受け取って配列中のオブジェクト1件ずつに対して、別の *DecisionTable* で定義したルールの評価に基づいたスコアを算出し、スコア順にソートすることができます。
このキーワードは、 IM-BIS との連携では利用できません。

動的処理対象者設定時に利用できるキーワード

動的処理対象者のルールの設定時に利用するキーワードです。
動的処理対象者設定と画面での値の受け渡しなどの連携の設定で差異がある部分を中心にまとめてありますので、記載のない要素は該当のテーブルタイプの説明などを参照してください。

- 【動的処理者設定】条件評価（DecisionTable）
- 【動的処理者設定】処理対象者プラグイン設定一覧（Data IMDynamicUser）
- 【動的処理者設定】返却する処理対象者の設定 ResponseObject（Datatype / Variable）
- 【動的処理者設定】オブジェクトのインスタンスの設定（DecisionObject）
- 【動的処理者設定】処理順の設定（Decision）

【動的処理者設定】条件評価（DecisionTable）

Decision に基づいて、Excelのシートで上から書いてある順に条件を評価します。
条件が合致したら、合致した行の *Conclusion* のSetting IDの値に基づく処理対象者を返却します。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	×

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
サブヘッダ部は、条件や評価（処理）単位でセルを結合します。

(1)	**DecisionTable** sampleDecision			
(2)	Condition		Conclusion	
	購入金額		Setting IDs	
(3)	>	100000	Are	1,2

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下のいずれかの方法で記述します。

- DecisionTable %テーブル名%
- DT %テーブル名%
- DecisionTableSingleHit %テーブル名%
- RuleFamily %テーブル名%
  - 各キーワードの後に、半角スペースを入れてテーブル名を記述します。

(2) サブヘッダに利用できるキーワード

動的承認者設定のサブヘッダには、*DecisionTable* と同様の「条件」を設定できますが、「結果」には、 *Conclusion* と組み合わせて「Setting IDs」に値を設定するようにしてください。

a. 条件に利用できるキーワード

キーワード	利用可否
Condition	○
ConditionBetween	○
ConditionVarOperValue	○
ConditionIntOperInt	○
ConditionRealOperReal	○
ConditionDateOperDate	○
ConditionAny	○
ConditionMap	×
If	○

> **コラム**
>
> 条件のセルの条件値を記入しない場合、 OpenRules では無条件と判断されます。
> すべての条件に合致しない場合の処理を記述する際などに利用します。

b. 結果に利用できるキーワード

キーワード	利用可否
Conclusion	○
ConclusionVarOperValue	×
ActionAny	×
ActionMap	×
Action	×
ActionPrint	×
Then	×
ActionExecute	×
Message	○
ActionRulesOnArray	×

(3) 明細

(2)サブヘッダのキーワードに合わせて記述します。
処理対象者を設定する場合は、結果 *Conclusion* と組み合わせて、 *【動的処理者設定】処理対象者プラグイン設定一覧（Data IMDynamicUser）* のIDの値を記述してください。

【動的処理者設定】処理対象者プラグイン設定一覧（Data IMDynamicUser）

IM-BIS のワークフローの動的承認・縦配置・横配置ノードの処理対象者設定を、 OpenRules の結果に基づいて決定するための処理対象者プラグインの種類と種類に応じた設定値をまとめた一覧です。
この一覧のIDを *【動的処理者設定】条件評価（DecisionTable）* の結果にセットします。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	×

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
結合するセルの単位が同じであれば、 *Data/Variable* と同様に列・行を入れ替えて記述することもできます。

(1)	Data **IMDynamicUser** settings											
(2)	id	process SetNo	code	company Cd	department SetCd	userCd	department Cd	compare	postCd	publicGroupSetCd	publicGroupCd	roleCd
	ID	処理設定No	判定用コード	会社コード	会社組織セットコード	ユーザコード	組織コード	比較	役職コード	パブリックグループセットコード	パブリックグループコード	役割コード
(3)	1	1	1			ueda						
	2	1	2	comp_sample_01	comp_sample_01		dept_sample_11	ge				
	3	2	2	comp_sample_01	comp_sample_01		dept_sample_21	eq				
		a	b	c	d	e		f		g		

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

- Data IMDynamicUser settings
  - 動的承認者設定の場合には、このメインヘッダの記述方法は固定です。

- *Data/Variable* と異なり、Variableをキーワードとして使うことはできません。

(2) サブヘッダに利用できるキーワード

このテーブルのサブヘッダは、「IMDynamicUser」の定義に合わせて設定します。
「IMDynamicUser」については、「 IM-BIS 仕様書 」の「 動的処理対象者設定に関する仕様 」を参照してください。

上段が設定項目の物理名、下段が設定項目の論理名です。

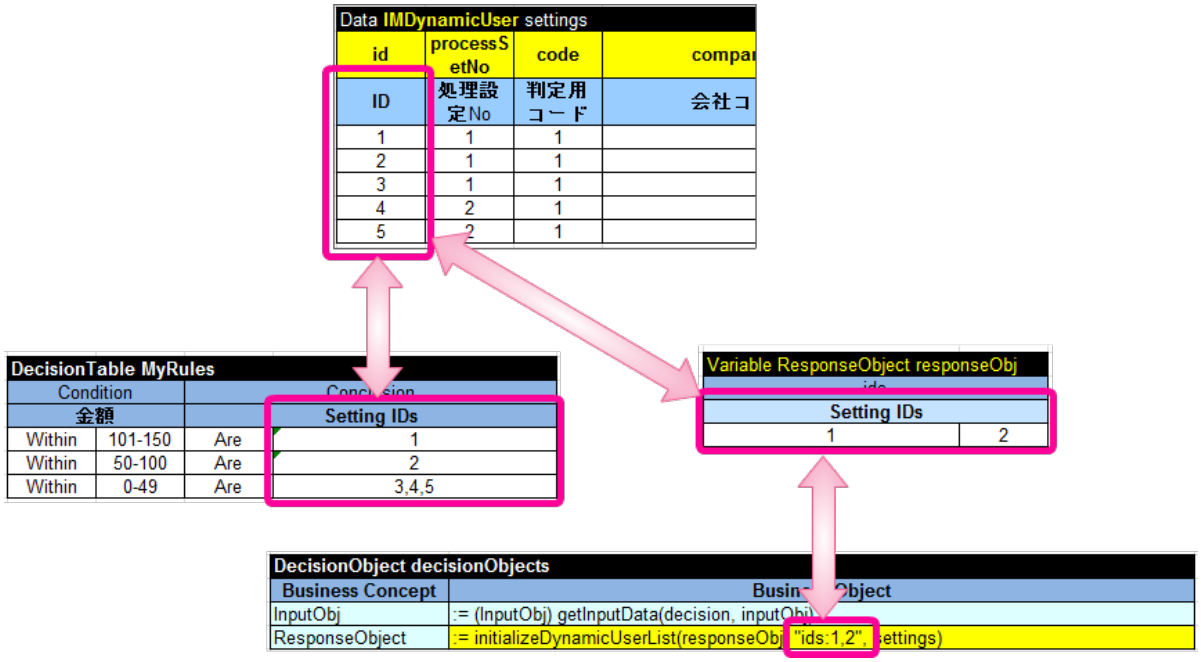
(3) 明細の記述方法

このテーブルの明細には、 *【動的処理者設定】条件評価（DecisionTable）* の結果に設定したい **すべての処理対象者プラグインと設定値** を記述します。

(a) 対象者ID

【動的処理者設定】条件評価（DecisionTable） の結果として返却する「Setting ID」や *ResponseObject* で定義するSetting IDsの初期値などに設定するときの処理対象者を表す識別IDです。
動的処理対象者設定が実行されたときに、このIDに紐づく処理対象者プラグインの種類やプラグインに合わせた組織コードなどの情報をワークフローに受け渡します。

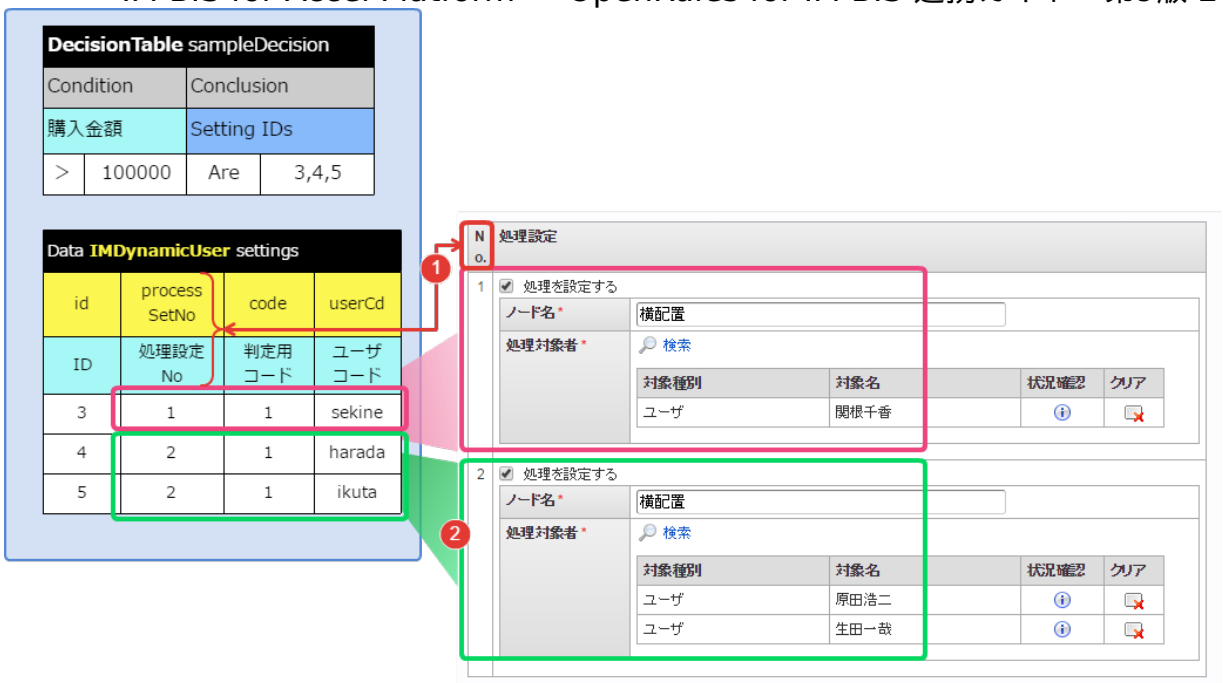
この対象者IDは、以下の図の通り *DecisionTable* 、 *Data/Variable* 、 *DecisionObject* の3つのテーブルで関連付けが必要です。
漏れなく設定してください。



(b) 処理設定No

横配置・縦配置に処理対象者を設定する場合に、展開したノードの何番目に設定するかを設定します。
動的承認ノードには、1を指定します。
横配置・縦配置ノードで複数のノードに展開する場合、1つのノードに対して番号に抜けがないようにしてください。

以下は、*DecisionTable* のSetting IDsに複数の処理対象者を設定した例です。



1. 縦配置ノード・横配置ノードでは、処理設定No（processSetNo）の設定値がワークフローの処理画面の処理対象者設定画面の「No.」に対応しています。

2. 上の例では、処理設定Noの2に対して、2種類の処理対象者（ユーザの原田浩二、生田一哉）が設定されています。
この場合、処理対象者設定の2番目のノードとして、2種類の処理対象者をそのまま設定します。

(c) 判定用コード

返却する処理対象者プラグインの種類を表します。

- 1：ユーザ
- 2：組織
- 3：パブリックグループ
- 4：役職
- 5：役割
- 6：組織＋役職
- 7：パブリックグループ＋役割

(d) 会社・組織関連の項目

判定用コードの「2」（組織）、「4」（役職）、「6」（組織＋役職）を選択したときの設定項目です。

(e) ユーザ関連の項目

判定用コードの「1」（ユーザ）を選択したときの設定項目です。

(f) 比較演算子

組織やパブリックグループとの比較方法を指定します。
以下の3種類の演算子のみが利用できます。

- ge
「～以上」を表します。
- eq
「～に等しい」を表します。
- le
「～以下」を表します。

(g) パブリックグループ関連の項目

判定用コードの「3」（パブリックグループ）、「5」（役割）、「7」（パブリックグループ＋役割）を選択したときの設定項目です。

【動的処理者設定】返却する処理対象者の設定 ResponseObject（Datatype / Variable）

Datatype 、 *Data/Variable* で、ワークフローに返却する処理対象者の設定を「ResponseObject」として定義します。
この定義については、 IM-BIS の動的処理者設定の仕様に基づく内容となるため、そのまま利用してください。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	×

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。

- Datatypeの定義

(1)	**Datatype** ResponseObject	
(3)	String[]	ids
	IMDynamicUser[]	settings

- Variableの定義

(1)	**Variable** ResponseObject responseObj		
(2)	ids		
	Setting IDs		
(3)	1	2	3

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

- Datatype ResponseObject
- Variable ResponseObject responseObj

(2) サブヘッダに利用できるキーワード

Datatypeには、サブヘッダはありません。
Variableで定義するサブヘッダは、処理対象者の設定値を格納する項目の物理名「ids」、論理名「Setting IDs」です。

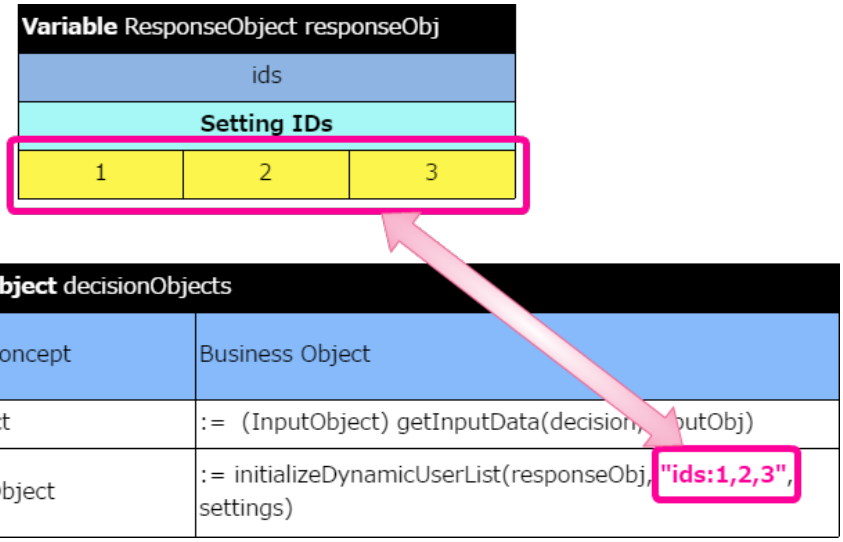
(3) 明細の記述方法

Datatype

Datatypeは、テーブルの基本構造の通りのデータ型・項目で設定します。
データ型「IMDynamicUser」は、 *Environment* で参照している「IntramartTemplate.xls」で定義しているため、別途定義しないようにしてください。

Variable

Variableの明細の値は、 *【動的処理者設定】処理対象者プラグイン設定一覧（Data IMDynamicUser）* に記載しているIDを入力します。
値の個数（セルの数）については、 *【動的処理者設定】オブジェクトのインスタンスの設定（DecisionObject）* のResponseObjectを初期化するための式の2番目のパラメータと一致させてください。



【動的処理者設定】オブジェクトのインスタンスの設定（DecisionObject）

Glossary で定義したオブジェクト（Business Concept）と IM-BIS （データマッパー）との値のマッピングを行います。
動的処理者設定では、返却するオブジェクトは *ResponseObject* です。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	×

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。

(1)	**DecisionObject** decisionObjects	
(2)	Business Concept	Business Object
(3)	InputObject	:= (InputObject) getInputData(decision, inputObj)
	ResponseObject	:= initializeDynamicUserList(responseObj, "ids:1,2,3", settings)

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下の方法で記述します。

- DecisionObject decisionObjects
  - メインヘッダは、必ず上記のように記載してください。
    名前が異なる等定義に誤りがある場合には、エラーが発生します。

(2) サブヘッダに利用できるキーワード

DecisionObjectのサブヘッダには、「Business Concept」と「Business Object」を記述します。
Glossary と同様にヘッダの内容は、漢字を含めてわかりやすい名称に変更することができます。

(3) 明細の記述方法

明細の左の列は、 *Glossary* でグループ（Business Concept）として定義した名称を記述します。
明細の右の列は、値の受け渡し方法に応じて式を記述します。
処理対象者の設定内容を受け渡し式は、記述例の通りに設定してください。

- IM-BIS （データマッパー）から入力値を受け取る場合
  画面上の値の受け渡しと同様に設定できます。

  ```
  := ({Business Conceptの型})getInputData(decision, {Business Conceptのインスタンス名})
  ```

  記述例

  ```
  := (RequestObject) getInputData(decision, requestObj)
  ```

- 動的承認ノードなどの絞込設定や処理対象者の自動設定の場合

  ```
  := initializeDynamicUserList(返却するオブジェクトのインスタンス名, 処理対象者設定のIDを格納する変数と初期値, 処理対象者設定の設定内容を格納する配列名)
  ```

  記述例

  ```
  := initializeDynamicUserList(responseObj, "ids:1,2,3", settings)
  ```

【動的処理者設定】処理順の設定（Decision）

DataMapperとの値の入出力、実行するDecisionTableの順序などを設定します。
動的処理対象者設定の場合には、返却するオブジェクトへの式の記述方法が変わります。

利用できる OpenRules のタイプ

タイプ	利用可否
Rule Engine	○
Rule Solver	×

テーブルの基本構造

メインヘッダ部は、構成する列分のセルを結合する必要があります。
サブヘッダ部は、条件や評価（処理）単位でセルを結合します。

(1)	**Decision** sampleRules	
(2)	ActionPrint	ActionExecute
	処理名	実行する処理
(3)	評価の実行	sampleDecision (DecisionTable名)
	処理対象者の返却	:= decision.put("ResponseObject", resultantDynamicUsers(responseObj, "ids", settings))

(1) メインヘッダの記述方法

メインヘッダは、1セルに結合し、以下のいずれかの方法で記述します。

- Decision %テーブル名%
  - 「Decision」の後に、半角スペースを入れてテーブル名を記述します。

(2) サブヘッダに利用できるキーワード

サブヘッダは、「ActionPrint」、「ActionExecute」が必須です。
その他のキーワードは任意です。
明細部はサブヘッダのキーワードに合わせて記述します。

a. 条件に利用できるキーワード

キーワード	利用可否
Condition	○
ConditionBetween	×
ConditionVarOperValue	×
ConditionIntOperInt	×
ConditionRealOperReal	×
ConditionDateOperDate	×
ConditionAny	○
ConditionMap	×
If	×

b. 結果に利用できるキーワード

結果では、最後に処理対象者を設定するメソッド（resultantDynamicUsers()）を実行してください。

キーワード	利用可否
Conclusion	○
ConclusionVarOperValue	○
ActionAny	○
ActionMap	×
Action	○
ActionPrint	○
Then	○
ActionExecute	○

キーワード	利用可否
Message	○
ActionRulesOnArray	×

(3) 明細の記述方法

明細の左の列は、 *Decision* を参考にして、コンソールなどに出力する内容を記述します。
明細の右の列のうち、 *ResponseObject* に結果を渡すための式については、動的処理者設定向けの記述方法に従ってください。

- Sub-Decisionや *DecisionTable* の実行

  Decision を参照してください。

- *ResponseObject* に評価結果として処理対象者の情報を受け渡す場合

```
:= initializeDynamicUserList(返却するオブジェクトの型名, resultantDynamicUsers(返却するオブジェクトのインスタンス名, 返却する処理対象者の対象IDを格納する項目の物理名, settings))
```

記述例

```
:= decision.put("ResponseObject", resultantDynamicUsers(responseObj, "ids", settings))
```

特殊なキーワード / テーブルタイプ固有のキーワード

この章で扱うキーワードは、テーブルタイプや用途が限定されている特殊なキーワードです。

- Variable（Glossary）
- Business Concept
- Attribute
- Domain
- OnOff

Variable（Glossary）

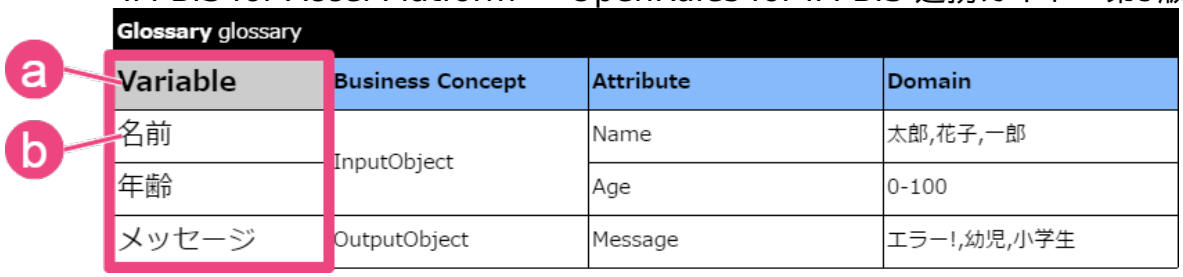
「Variable」は、 *Glossary* で *DecisionTable* のユーザ向けの項目名を定義するためのキーワードです。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	×
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	×
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	○
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

Variableを利用する場合、以下の図のように記述します。



Glossary glossary

Variable	Business Concept	Attribute	Domain
名前	InputObject	Name	太郎,花子,一郎
年齢		Age	0-100
メッセージ	OutputObject	Message	エラー!,幼児,小学生

a. キーワード
Glossary での項目の論理名の定義を表す OpenRules のキーワードです。

b. 論理名
Glossary で定義する対象の項目の論理名です。
論理名は *DecisionTable* での条件や処理対象の指定時に利用します。
論理名にはひらがな・漢字を含むマルチバイト文字が利用できます。

Business Concept

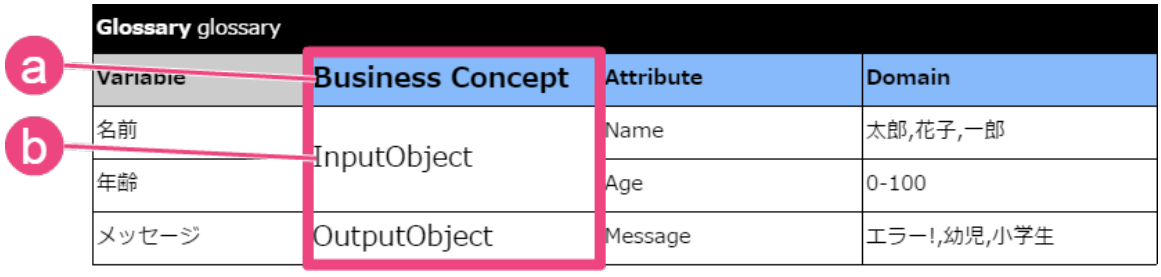
「Business Concept」は、 *Glossary* の複数の *Attribute* を、入出力などの単位で特定のオブジェクト（グループ）でまとめるためのキーワードです。
ただし、このキーワードのセルはラベルとして利用されるため、漢字などを含む任意の文字列に変更することができます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	×
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	×
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	○
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

Business Conceptを利用する場合、以下の図のように記述します。



Glossary glossary

Variable	Business Concept	Attribute	Domain
名前	InputObject	Name	太郎,花子,一郎
年齢		Age	0-100
メッセージ	OutputObject	Message	エラー!,幼児,小学生

a. キーワード
Glossary でのBusiness Concept（複数の項目をまとめるグループやオブジェクト）の定義を表す OpenRules のキーワードです。

b. Business Concept（複数の項目をまとめるグループやオブジェクト）
Glossary で定義する複数の項目をまとめるBusiness Concept（複数の項目をまとめるグループやオブジェクト）です。
OpenRules ではBusiness Concept単位でインスタンスの生成を行います。

Attribute

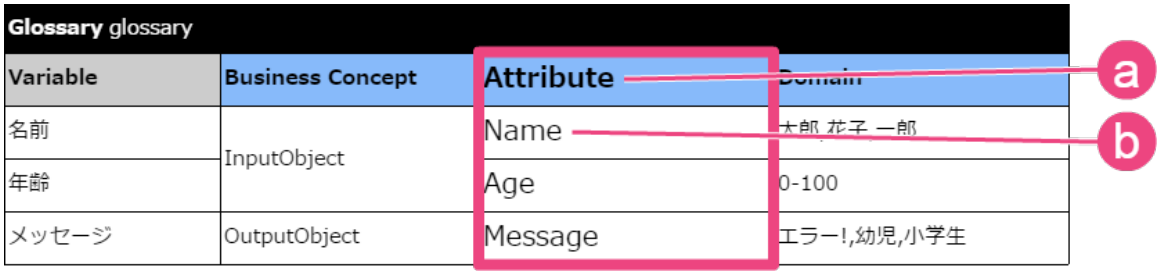
「Attribute」は、 *Glossary* で *DecisionTable* の *Variable（Glossary）* を、ルールの実行時にプログラムに変換する際の名称とのマッピングをするためのキーワードです。
ただし、このキーワードのセルはラベルとして利用されるため、漢字などを含む任意の文字列に変更することができます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	×
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	×
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	○
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

Attributeを利用したい場合、以下の図のように記述します。



a. キーワード
Glossary での項目の物理名の定義を表す OpenRules のキーワードです。

b. 物理名
Glossary で定義する対象の項目の物理名です。
物理名は OpenRules との連携先のJavaプログラムで定義したメンバ名やデータソース定義でのリクエストパラメータ・レスポンスフィールドに該当します。

Domain

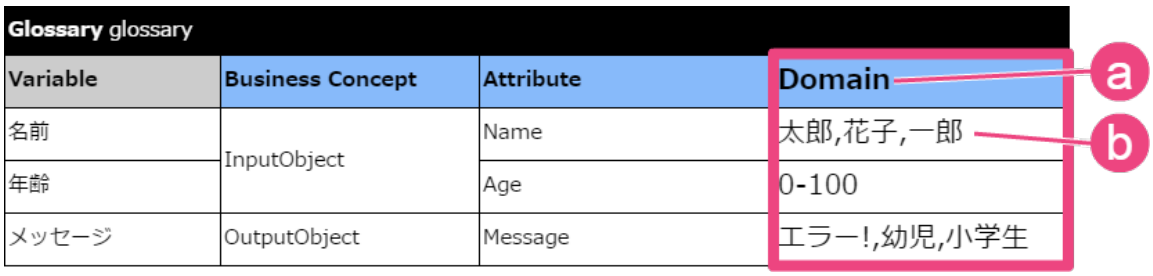
「Domain」は、 *Glossary* で *Variable（Glossary）* の値の例を表示する列のキーワードです。
ただし、このキーワードのセルはラベルとして利用されるため、漢字などを含む任意の文字列に変更することができます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	×
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	×
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	×
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	○
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

Domainを利用したい場合、以下の図のように記述します。



Glossary glossary			
Variable	Business Concept	Attribute	Domain
名前	InputObject	Name	太郎,花子,一郎
年齢		Age	0-100
メッセージ	OutputObject	Message	エラー!,幼児,小学生

a. キーワード
Glossary での項目の値の範囲の定義を表す OpenRules のキーワードです。
この列自体の指定はオプションであるため、定義しなくても問題ありません。

b. ドメイン
Glossary で定義する対象の項目の値の範囲です。
IM-BIS との連携時には特にこの設定内容は利用されません。

補足事項

この項目で設定した値の範囲は、 *compareDomain* を利用した場合の条件にも利用されます。

OnOff

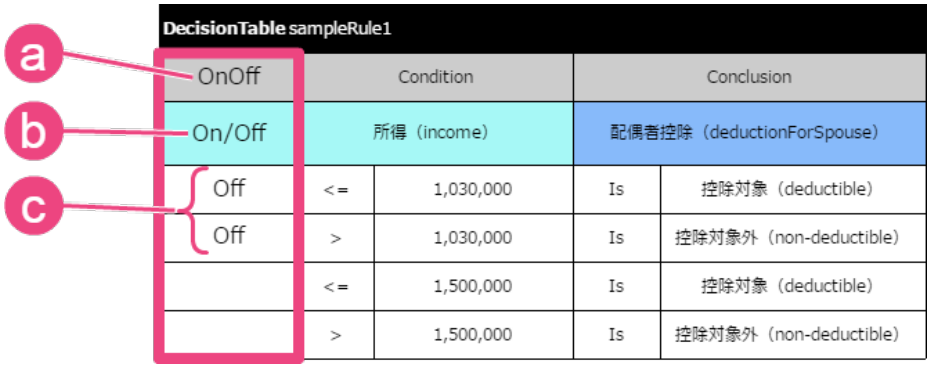
「OnOff」は、 *DecisionTable* 等で該当の行の評価の有効・無効を切り替えるためのキーワードです。
同じ *DecisionTable* 内でルールを履歴として残しておきたい場合に利用できます。

利用できるテーブルタイプ

テーブルタイプ	利用可否
条件評価（DecisionTable / DT / DecisionTableSingleHit / RuleFamily）	○
マルチヒット型の条件評価1（DecisionTable1 / DT1 / DecisionTableMultiHit）	○
マルチヒット型の条件評価2（DecisionTable2 / DT2 / DecisionTableSequence）	○
処理順の設定（Decision）	×
項目名のマッピング（Glossary）	×
項目とデータ型の定義（Datatype）	×
項目の初期値の定義（Data / Variable）	×
オブジェクトのインスタンスの設定（DecisionObject）	×
Javaのコードの定義（Method）	×
環境設定情報（Environment）	×

記述方法

OnOffを利用したい場合、以下の図のように記述します。



DecisionTable sampleRule1				
OnOff	Condition		Conclusion	
On/Off	所得（income）		配偶者控除（deductionForSpouse）	
Off	<=	1,030,000	Is	控除対象（deductible）
Off	>	1,030,000	Is	控除対象外（non-deductible）
	<=	1,500,000	Is	控除対象（deductible）
	>	1,500,000	Is	控除対象外（non-deductible）

a. キーワード
DecisionTable でルールの実行可否を表す OpenRules のキーワードです。
この列は *DecisionTable* の1番左に定義する必要があります。

b. 列ラベル
この列を識別する名称です。
ユーザにわかりやすい名称を自由に設定できます。

c. 設定値
OnOffでは、無効にしたいルールの行に”Off”と入力してください。
大文字・小文字は区別せず、”Off”と入力された行のルールは評価されずに次の処理に遷移します。

補足事項

この列タイプは、 OpenRules 6.3.4以降で利用できます。

Methodで利用できるキーワード（API）

Decision や *DecisionTable* の明細部で”::=”で始まる式を書く場合や *Method* で記述できるキーワードです。
”::=”のないセルでは、 *$(getString)* のみ記述できます。

各種メソッドの記述方法については、 OpenRules が提供するAPIリストやJavaDocを参照してください。

- OpenRules JavaDoc（English）
- OpenRules API [T]ページ（English）

修飾子とタイプ	メソッドと説明
java.lang.String	*$(getString)* / *getString* (java.lang.String name)
int	*$I(getInt)* / *getInt* getInt(java.lang.String name)
long	*$L(getLong)* / getLong(java.lang.String name)
double	*$R(getReal)* / getReal(java.lang.String name)
java.util.Date	*$D(getDate)* / getDate(java.lang.String name)
boolean	*$B(getBool)* / getBool(java.lang.String name)
BigDecimal	*$G(getBigDecimal)* / getBigDecimal(java.lang.String name)
Var	【Solver】 *$V(getVar)*
java.lang.Object	【Solver】 *$O(getBusinessObject)* / getBusinessObject(java.lang.String businessConcept)
void	*setString* (java.lang.String name, java.lang.String value)
void	*setInt* (java.lang.String name, int value)
void	*setReal* (java.lang.String name, double value) *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目に実数(Real)型の値をセットします。
void	*setDate* (java.lang.String name, java.util.Date date) *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目に日付(Date)型の値をセットします。
void	*setBool* (java.lang.String name, boolean value) *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目に論理値(Boolean)型の値をセットします。
boolean	*compareString* (java.lang.String name, java.lang.String op, java.lang.String value) *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目の値を演算子を用いて比較します。
boolean	*compareInt* (java.lang.String name, java.lang.String op, int value) *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目の値を演算子を用いて比較します。
boolean	*compareInt* (java.lang.String name1, java.lang.String op, java.lang.String name2) ２つの *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目の値を比較します。
boolean	*compareReal* (java.lang.String name, java.lang.String op, double value) *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目の値を演算子を用いて比較します。
boolean	*compareReal* (java.lang.String name1, java.lang.String op, java.lang.String name2) ２つの *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目の値を比較します。
boolean	*compareDate* (java.lang.String name, java.lang.String op, java.util.Date date) *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目の値を演算子を用いて比較します。
boolean	*compareDate* (java.lang.String name1, java.lang.String op, java.lang.String name2) ２つの *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目の値を比較します。
boolean	*compareBool* (java.lang.String name, java.lang.String op, boolean value) *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目の値を演算子を用いて比較します。

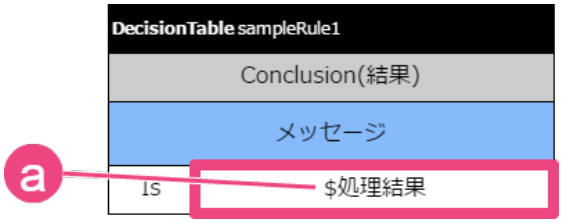
修飾子とタイプ	メソッドと説明
boolean	*compareBool* (java.lang.String name1, java.lang.String op, java.lang.String name2) ２つの *Glossary* に定義した *Variable（Glossary）* の名前に合致した項目の値を演算子を用いて比較します。
boolean	*compareDomain* (java.lang.String name, java.lang.String op, java.lang.String domain)
void	*iterate* (java.lang.String arrayName, java.lang.String rules) 配列の各要素に対して *DecisionTable* を繰り返し実行します。
void	*iterate* (java.lang.String arrayName, java.lang.String arrayType, java.lang.String rules) 配列の各要素に対して *DecisionTable* を繰り返し実行します。
void	*sort* (java.lang.String arrayName) 指定した名前の配列を *DecisionTable* の結果に基づいてソートします。
void	*sort* (java.lang.String arrayName, java.lang.String rules) 指定した名前の配列を特定の *DecisionTable* の結果に基づいてソートします。
void	*sort* (com.openrules.ruleengine.ComparableDecisionVariable[] array) 指定した配列を *DecisionTable* の結果に基づいてソートします。
void	*sort* (com.openrules.ruleengine.ComparableDecisionVariable[] array, java.lang.String rules) 指定した配列を特定の *DecisionTable* の結果に基づいてソートします。

$(getString)

文字列型の項目の値を取得することができるマクロ / メソッドです。
$から始まるマクロで記述した場合には、実行時に OpenRules の処理内で自動的にメソッドの形式に展開されます。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **$%項目の論理名%**
先頭に”$”を追加し、取得した値の代入先の項目の論理名を記述します。
getString()のみ、以下のどちらの記述方法も利用できます。
- $%項目の論理名%
- ::=${%項目の論理名%}

```
//マクロを利用する場合

 ::= ${項目の論理名}

//実行時に展開される形式、メソッドとして記述する場合

 ::= decision.getString("項目の論理名")
```

項目（文字列型）の値の取得のみを行う場合には、::=と{}なしで記述することもできます。

```
//マクロを利用する場合（項目の論理名のみ）

 $項目の論理名
```

メソッドの概要

public java.lang.String getString(java.lang.String name)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名

戻り値

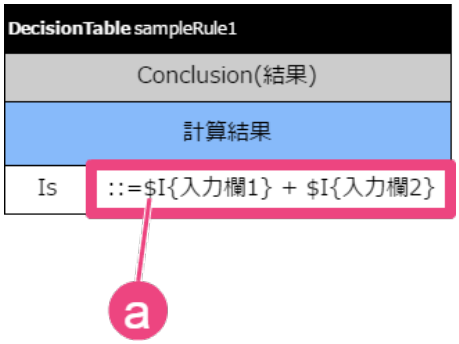
文字列（String）型の *Variable（Glossary）* の値

$I(getInt)

整数型の項目の値を取得することができるマクロ / メソッドです。
$から始まるマクロで記述した場合には、実行時に OpenRules の処理内で自動的にメソッドの形式に展開されます。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **$I%項目の論理名%**
先頭に”$I”を追加し、取得した値の代入先の項目の論理名を記述します。

```
//マクロを利用する場合

 ::= $I{項目の論理名}

//実行時に展開される形式、メソッドとして記述する場合

 ::= decision.getInt("項目の論理名")
```

メソッドの概要

public int getInt(java.lang.String name)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名

戻り値

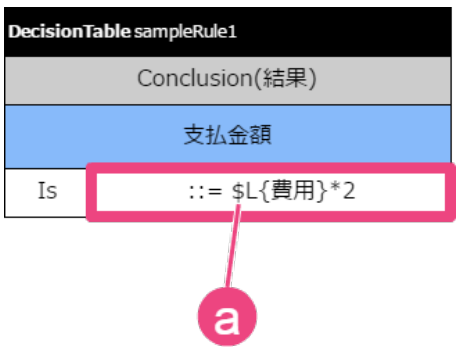
整数（int）型の *Variable（Glossary）* の値

$L(getLong)

整数（long）型の項目の値を取得することができるマクロ / メソッドです。
$から始まるマクロで記述した場合には、実行時に OpenRules の処理内で自動的にメソッドの形式に展開されます。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **$L%項目の論理名%**
先頭に”$L”を追加し、取得した値の代入先の項目の論理名を記述します。

```
//マクロを利用する場合（項目）

 ::= $L{項目の論理名}

//実行時に展開される形式、メソッドとして記述する場合

 ::= decision.getLong("項目の論理名")
```

メソッドの概要

public long getLong(java.lang.String name)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名

戻り値

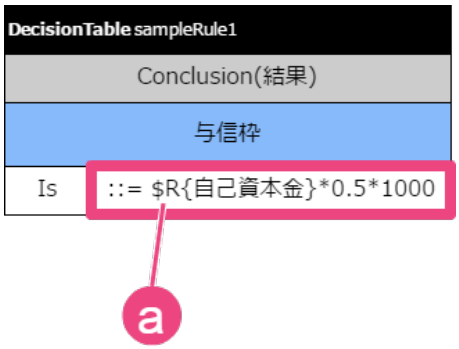
64ビット整数（long）型の *Variable（Glossary）* の値

$R(getReal)

実数（浮動小数点実数：double）型の項目の値を取得することができるマクロ / メソッドです。
$から始まるマクロで記述した場合には、実行時に OpenRules の処理内で自動的にメソッドの形式に展開されます。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **$R%項目の論理名%**
先頭に”$R”を追加し、取得した値の代入先の項目の論理名を記述します。

```
//マクロを利用する場合

 ::= $R{項目の論理名}

//実行時に展開される形式、メソッドとして記述する場合

 ::= decision.getReal("項目の論理名")
```

メソッドの概要

public double getReal(java.lang.String name)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名

戻り値

実数（浮動小数点実数：double）型の *Variable（Glossary）* の値

$D(getDate)

日付（Date）型の用語の値を取得することができるマクロ / メソッドです。
$から始まるマクロで記述した場合には、実行時に OpenRules の処理内で自動的にメソッドの形式に展開されます。

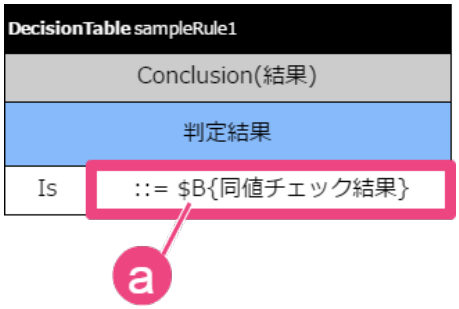
IM-BIS との連携では利用できません。

$B(getBool)

論理（boolean）型の用語の値を取得することができるマクロ / メソッドです。
$から始まるマクロで記述した場合には、実行時に OpenRules の処理内で自動的にメソッドの形式に展開されます。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **$B%項目の論理名%**
先頭に”$B”を追加し、取得した値の代入先の項目の論理名を記述します。

```
//マクロを利用する場合

 ::= $B{項目の論理名}

//実行時に展開される形式、メソッドとして記述する場合

 ::= decision.getBool("項目の論理名")
```

メソッドの概要

public boolean getBool(java.lang.String name)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名

戻り値

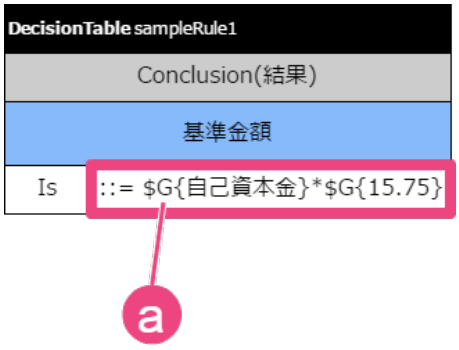
論理値（Boolean）型の *Variable（Glossary）* の値

$G(getBigDecimal)

任意精度の符号付き10進数（BigDecimal）型の項目の値を取得することができるマクロ / メソッドです。
$から始まるマクロで記述した場合には、実行時に OpenRules の処理内で自動的にメソッドの形式に展開されます。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **$G%項目の論理名%**
先頭に”$G”を追加し、取得した値の代入先の項目の論理名を記述します。
または、項目の論理名の代わりに直接値を指定することもできます。

```
//マクロを利用する場合（項目）

 ::= $G{項目の論理名}

//マクロを利用する場合（値）

 ::= $G{値}

//実行時に展開される形式、メソッドとして記述する場合

 ::= decision.getBigDecimal("項目の論理名")
```

メソッドの概要

public BigDecimal getBigDecimal(java.lang.String name)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名、またはBigDecimal型で扱いたい数値

戻り値

任意精度の符号付き10進数（BigDecimal）型の *Variable（Glossary）* の値

$V(getVar)

用語を取得することができるマクロです。
このメソッドは、Rule Solverで利用されています。
$から始まるマクロで記述した場合には、実行時に OpenRules の処理内で自動的にメソッドの形式に展開されます。
IM-BIS との連携では利用できません。

$O(getBusinessObject)

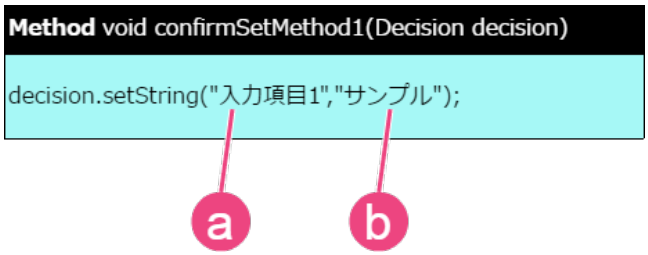
Glossaryに定義したBusinessObjectを取得することができるマクロ / メソッドです。
$から始まるマクロで記述した場合には、実行時に OpenRules の処理内で自動的にメソッドの形式に展開されます。
IM-BIS との連携では利用できません。

setString

文字列型の用語に、任意の値を設定することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
   値をセットする対象の項目の論理名です。

b. **値**
   (a) の項目にセットする値です。

```
//メソッドとして記述する場合

 ::= decision.setString("項目の論理名","設定する値")
```

メソッドの概要

public void setString(java.lang.String name,java.lang.String value)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
value- 設定する値

戻り値

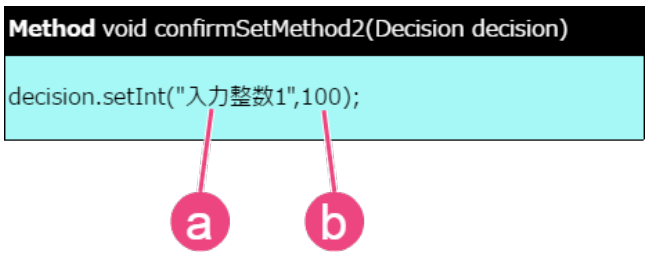
なし

setInt

整数型の用語に、任意の値を設定することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
値をセットする対象の項目の論理名です。

b. **値**
(a) の項目にセットする値です。

```
//メソッドとして記述する場合

 ::= decision.setInt("項目の論理名","設定する値")
```

メソッドの概要

public void setInt(java.lang.String name,int value)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
value- 設定したい値

戻り値

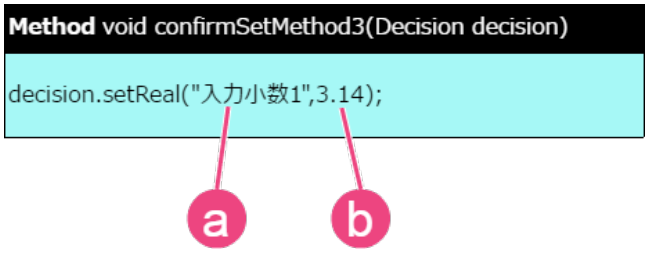
なし

setReal

実数型の用語に、任意の値を設定することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
値をセットする対象の項目の論理名です。

b. **値**
(a) の項目にセットする値です。

```
//メソッドとして記述する場合

 ::= decision.setReal("項目の論理名","設定する値")
```

メソッドの概要

public void setReal(java.lang.String name, double value)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
value- 設定したい値

戻り値

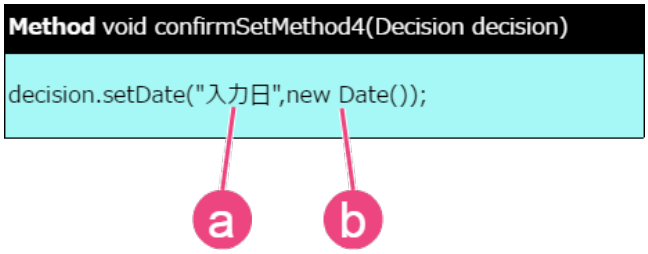
なし

setDate

日付型の用語に、任意の値を設定することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
値をセットする対象の項目の論理名です。

b. **値**
(a) の項目にセットする値です。

```
//メソッドとして記述する場合

 ::= decision.setDate("項目の論理名","設定する値")
```

メソッドの概要

public void setDate(java.lang.String name, java.util.Date date)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
value- 設定したい値

戻り値

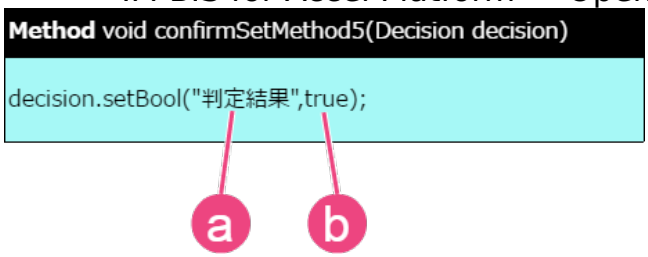
なし

setBool

論理値型の用語に、任意の値を設定することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
値をセットする対象の項目の論理名です。

b. **値**
(a) の項目にセットする値です。

```
//メソッドとして記述する場合

 ::= decision.setBool("項目の論理名","設定する値")
```

メソッドの概要

public void setBool(java.lang.String name, boolean value)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
value- 設定したい値

戻り値

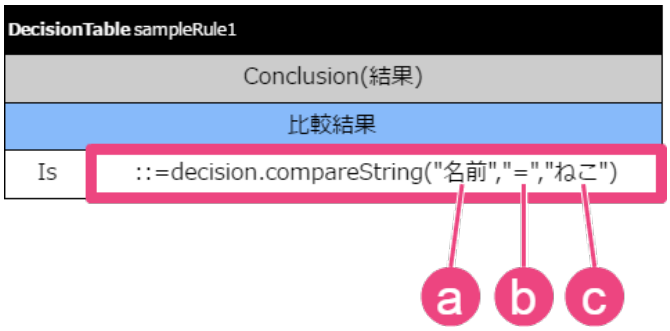
なし

compareString

文字列型の項目の値を特定の値と比較することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
値の比較を行う左辺の項目の論理名です。

b. **比較演算子**
比較方法を表す演算子です。

c. **値**
(a) の項目と比較を行う値です。

```
//メソッドとして記述する場合

 ::= decision.compareString("項目の論理名","比較演算子","比較する値")
```

メソッドの概要

public boolean compareString(java.lang.String name, java.lang.String op, java.lang.String value)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
op- *利用できる演算子* にある文字列型に利用できる比較演算子
value- 比較する値

戻り値

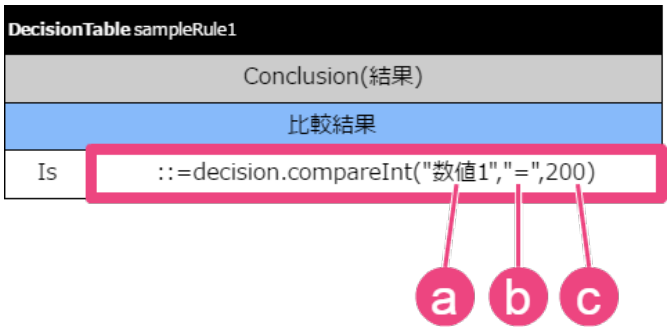
論理値（boolean）型の値

compareInt(name, op, value)

整数型の項目の値を特定の値と比較することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
値の比較を行う左辺の項目の論理名です。

b. **比較演算子**
比較方法を表す演算子です。

c. **値**
(a) の項目と比較を行う値です。

```
//メソッドとして記述する場合

 ::= decision.compareInt("項目の論理名","比較演算子","比較する値")
```

メソッドの概要

public boolean compareInt(java.lang.String name, java.lang.String op, int value)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
op- *利用できる演算子* にある整数型に利用できる比較演算子
value- 比較する値

戻り値

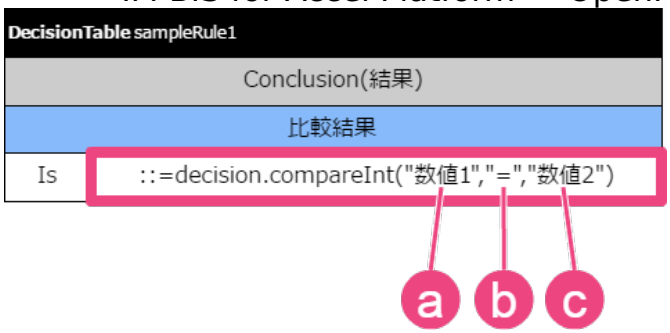
論理値（boolean）型の値

compareInt(name1, op, name2)

２つの整数型の項目の値を比較することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名（左辺）**
値の比較を行う左辺の項目の論理名です。

b. **比較演算子**
比較方法を表す演算子です。

c. **項目の論理名（右辺）**
値の比較を行う右辺の項目の論理名です。

```
//メソッドとして記述する場合

 ::= decision.compareInt("比較する1つめの項目の論理名","比較演算子","比較する2つめの項目の論理名")
```

メソッドの概要

public boolean compareInt(java.lang.String name1, java.lang.String op, java.lang.String name2)

パラメータ

name1- *Glossary* に定義した *Variable（Glossary）* の論理名
op- *利用できる演算子* にある整数型に利用できる比較演算子
name2- *Glossary* に定義した *Variable（Glossary）* の論理名

戻り値

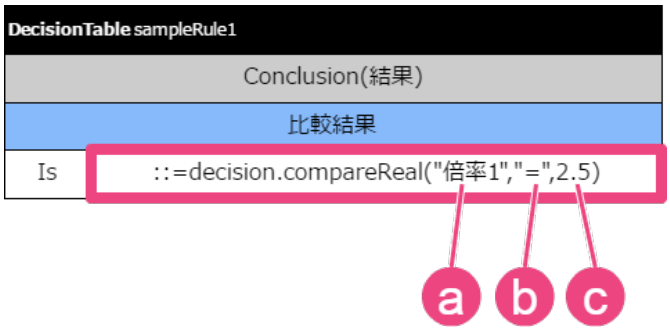
論理値（boolean）型の値

compareReal(name, op, value)

実数型の項目の値を特定の値と比較することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
値の比較を行う左辺の項目の論理名です。

b. **比較演算子**
比較方法を表す演算子です。

c. **値**
(a) の項目と比較を行う値です。

```
//メソッドとして記述する場合

 ::= decision.compareReal("項目の論理名","比較演算子","比較する値")
```

メソッドの概要

public boolean compareReal(java.lang.String name, java.lang.String op, double value)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
op- *利用できる演算子* にある実数型に利用できる比較演算子
value- 比較する値

戻り値

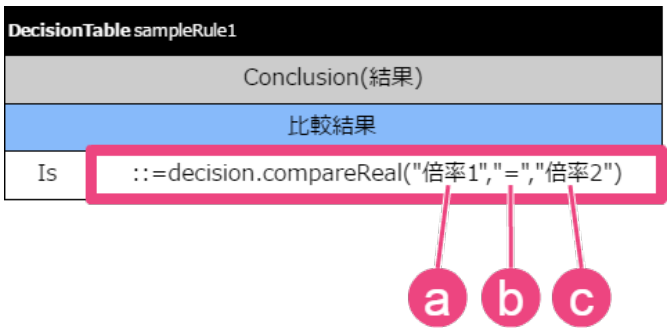
論理値（boolean）型の値

compareReal(name1, op, name2)

２つの実数型の項目の値を比較することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名（左辺）**
値の比較を行う左辺の項目の論理名です。

b. **比較演算子**
比較方法を表す演算子です。

c. **項目の論理名（右辺）**
値の比較を行う右辺の項目の論理名です。

```
//メソッドとして記述する場合

 ::= decision.compareReal("比較する1つめの項目の論理名","比較演算子","比較する2つめの項目の論理名")
```

メソッドの概要

public boolean compareReal(java.lang.String name1, java.lang.String op, java.lang.String name2)

パラメータ

name1- *Glossary* に定義した *Variable（Glossary）* の論理名
op- *利用できる演算子* にある実数型に利用できる比較演算子
name2- *Glossary* に定義した *Variable（Glossary）* の論理名

戻り値

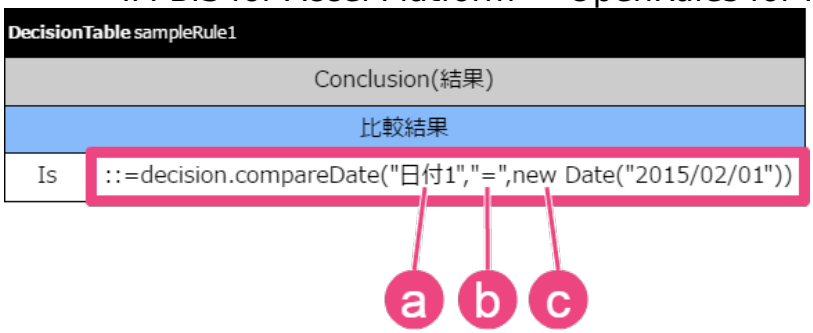
論理値（boolean）型の値

compareDate(name, op, date)

日付型の項目の値を特定の値と比較することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
値の比較を行う左辺の項目の論理名です。

b. **比較演算子**
比較方法を表す演算子です。

c. **値**
(a) の項目と比較を行う値です。

```
//メソッドとして記述する場合

 ::= decision.compareDate("項目の論理名","比較演算子","比較する値")
```

メソッドの概要

public boolean compareDate(java.lang.String name, java.lang.String op, java.util.Date date)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
op- *利用できる演算子* にある日付型に利用できる比較演算子
date- 比較する値（日付）

戻り値

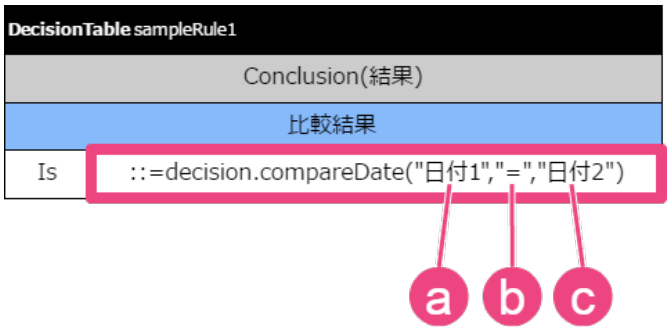
論理値（boolean）型の値

compareDate(name1, op, name2)

２つの日付型の項目の値を比較することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名（左辺）**
値の比較を行う左辺の項目の論理名です。

b. **比較演算子**
比較方法を表す演算子です。

c. **項目の論理名（右辺）**
値の比較を行う右辺の項目の論理名です。

```
//メソッドとして記述する場合

 ::= decision.compareDate("比較する1つめの項目の論理名","比較演算子","比較する2つめの項目の論理名")
```

メソッドの概要

public boolean compareDate(java.lang.String name1, java.lang.String op, java.lang.String name2)

パラメータ

name1- *Glossary* に定義した *Variable（Glossary）* の論理名
op- *利用できる演算子* にある日付型に利用できる比較演算子
name2- *Glossary* に定義した *Variable（Glossary）* の論理名

戻り値

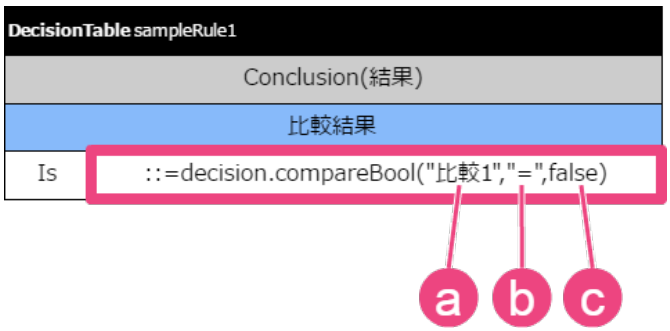
論理値（boolean）型の値

compareBool(name, op, value)

論理値型の項目の値を特定の値と比較することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名**
値の比較を行う左辺の項目の論理名です。

b. **比較演算子**
比較方法を表す演算子です。

c. **値**
(a) の項目と比較を行う値です。

```
//メソッドとして記述する場合

 ::= decision.compareBool("項目の論理名","比較演算子","比較する値")
```

メソッドの概要

public boolean compareBool(java.lang.String name, java.lang.String op, boolean value)

パラメータ

name- *Glossary* に定義した *Variable（Glossary）* の論理名
op- *利用できる演算子* にある論理値型に利用できる比較演算子
value- 比較する値

戻り値

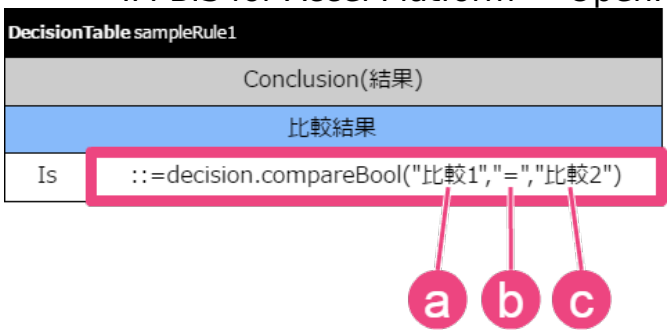
論理値（boolean）型の値

compareBool(name1, op, name2)

２つの論理値型の項目の値を比較することができるメソッドです。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **項目の論理名（左辺）**
値の比較を行う左辺の項目の論理名です。

b. **比較演算子**
比較方法を表す演算子です。

c. **項目の論理名（右辺）**
値の比較を行う右辺の項目の論理名です。

```
//メソッドとして記述する場合

::= decision.compareBool("比較する1つめの項目の論理名","比較演算子","比較する2つめの項目の論理名")
```

メソッドの概要

public boolean compareBool(java.lang.String name1, java.lang.String op, java.lang.String name2)

パラメータ

name1- *Glossary* に定義した *Variable（Glossary）* の論理名
op- *利用できる演算子* にある論理値型に利用できる比較演算子
name2- *Glossary* に定義した *Variable（Glossary）* の論理名

戻り値

論理値（boolean）型の値

compareDomain

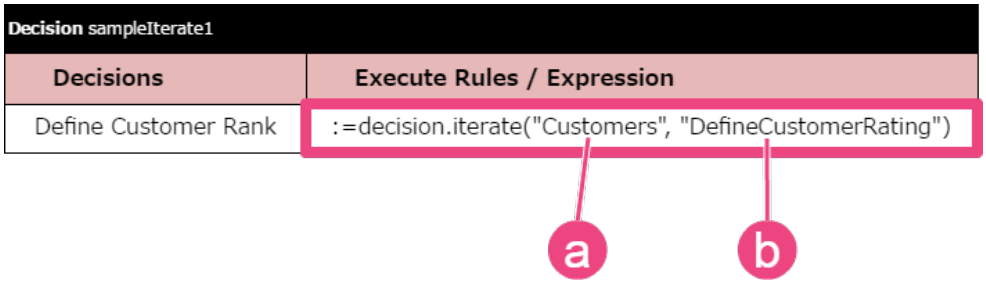
項目の値が *Glossary* に定義している *Domain* の範囲に含まれるかを確認できるメソッドです。
IM-BIS との連携では利用できません。

iterate(arrayName, rules)

パラメータに指定した名前の配列に含まれる要素1つ1つに対して、 *DecisionTable* による評価を繰り返し実行するためのメソッドです。
テーブルタイプ *DecisionTableIterate* 、列タイプの *ActionRulesOnArray* や *ActionIterate* でも同様の処理を実装できます。
IM-BIS との連携では利用できません。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **配列の名前**
(b)の *DecisionTable* の実行対象の要素を含む配列の名前です。

b. **DecisionTableの名前**
各要素に対して実行する *DecisionTable* の名前です。

以下の例は上の図と同義です。

- *DecisionTableIterate*

DecisionTableIterate sampleDTIterate	
Array of Objects	**Rules**
Customers	DefineCustomerRating

- *ActionRulesOnArray*
  DecisionTable1 は”DecisionTableMultiHit”と記述しても問題ありません。

DecisionTable1 sampleDTIterate		
ActionRulesOnArray		
Array of Objects	**Array Type**	**Rules**
Customers	Customer	DefineCustomerRating

- *ActionIterate*
  DecisionTable1 は”DecisionTableMultiHit”と記述しても問題ありません。

DecisionTable1 sampleDTIterate	
ActionIterate	
Array of Objects	**Rules**
Customers	DefineCustomerRating

メソッドの概要

public void iterate(java.lang.String arrayName, java.lang.String rules)

パラメータ

arrayName- *DecisionTable* の繰り返し処理の実行対象の配列の名前
rules- 指定した配列の要素に対して実行する *DecisionTable* の名前

戻り値

なし

iterate(arrayName, arrayType, rules)

パラメータに指定した名前の配列に含まれる要素1つ1つに対して、 *DecisionTable* による評価を繰り返し実行するためのメソッドです。
テーブルタイプ *DecisionTableIterate* 、列タイプの *ActionRulesOnArray* や *ActionIterate* でも同様の処理を実装できます。
IM-BIS との連携では利用できません。

記述方法

利用するTableTypeの明細部の値に記述します。

Decision sampleIterate1	
Decisions	**Execute Rules / Expression**
Define Customer Rank	:=decision.iterate("Customers", "Customer", "DefineCustomerRating")

a. **配列の名前**
   (c)の *DecisionTable* の実行対象の要素を含む配列の名前です。

b. **配列の要素の型**
   配列中の各要素の型です。

c. **DecisionTableの名前**
   各要素に対して実行する *DecisionTable* の名前です。

メソッドの概要

public void iterate(java.lang.String arrayName, java.lang.String arrayType, java.lang.String rules)

パラメータ

arrayName- *DecisionTable* の繰り返し処理の実行対象の配列の名前
arrayType- arrayNameで指定した配列の要素の型
rules- 指定した配列の要素に対して実行する *DecisionTable* の名前

戻り値

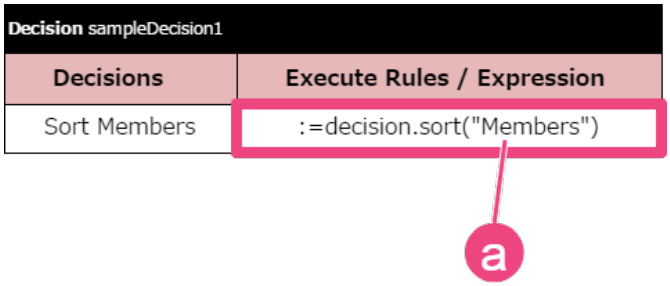
なし

sort(arrayName)

パラメータに指定した名前の配列を *DecisionTable* の結果に基づいてソートするためのメソッドです。
テーブルタイプ *DecisionTableSort* でも同様の処理を実装できます。
IM-BIS との連携では利用できません。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **配列の名前**
ソート対象の配列の名前です。
配列に含まれるオブジェクトは OpenRules が提供するクラス「ComparableDecisionVariable」を継承している必要があります。
ソートを行う際の *DecisionTable* の名前は自動的に “Compare”+配列名で定義されるため、この名前の *DecisionTable* でソートの条件を指定してください。

配列をソートするための要素は、 *Glossary* で”オブジェクト名”+”1”、”オブジェクト名”+”2”と定義してください。
上の図の例の場合、 *Glossary* の定義は以下の通りです。

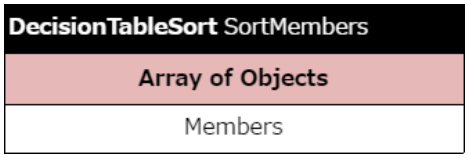
Glossary glossary

Variable	Business Concept	Attribute
TeamMembers（チームメンバー）	Team	members
Member1_Status（ステータス）	Menber1	status
Member1_Age（年齢）		age
Member1_Score（スコア）		score
Member2_Status（ステータス）	Menber2	status
Member2_Age（年齢）		age
Member2_Score（スコア）		score

DecisionTable では、 *Glossary* の番号を付けたオブジェクト同士の評価結果を”Score”に点数としてセットすると、結果に基づいたソートが行えます。

以下の例は上の図と同義です。

- *DecisionTableSort*
  - arrayNameのみを指定する場合



- arrayNameと適用する *DecisionTable* を指定する場合

DecisionTableSortUsingRules SortMembers	
Array of Objects	**Comparison Rules**
Members	CompareMembers

- *sort(arrayName, rules)*

Decision sampleDecision1	
Decisions	**Execute Rules / Expression**
Sort Members	:=decision.sort("Members", "CompareMembers")

メソッドの概要

public void sort(java.lang.String arrayName)

パラメータ

arrayName- ソート対象の配列の名前

戻り値

なし

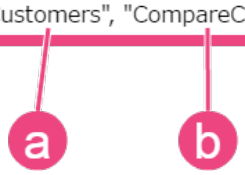
sort(arrayName, rules)

パラメータに指定した名前の配列を *DecisionTable* の結果に基づいてソートするためのメソッドです。
テーブルタイプ *DecisionTableSort* でも同様の処理を実装できます。
IM-BIS との連携では利用できません。

記述方法

利用するTableTypeの明細部の値に記述します。

Decision sampleDecision2	
Decisions	**Execute Rules / Expression**
Sort Customers	:=decision.sort("Customers", "CompareCustomers")



a. **配列の名前**
ソート対象の配列の名前です。
配列に含まれるオブジェクトは OpenRules が提供するクラス「ComparableDecisionVariable」を継承している必要があります。

b. **DecisionTableの名前**
ソートするためのスコアを算出する *DecisionTable* の名前です。

メソッドの概要

public void sort(java.lang.String arrayName, java.lang.String rules)

パラメータ

arrayName- ソート対象の配列の名前
rules- ソートの基準とするスコアを算出する *DecisionTable* の名前

戻り値

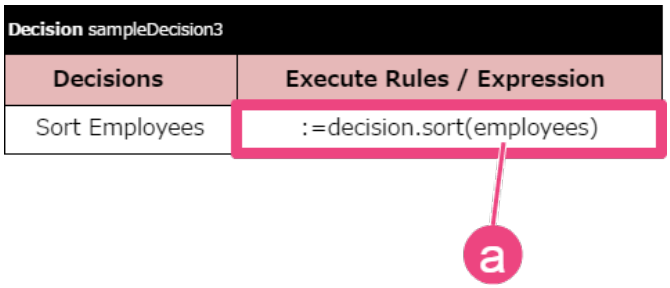
なし

sort(array)

パラメータに指定した配列を *DecisionTable* の結果に基づいてソートするためのメソッドです。
テーブルタイプ *DecisionTableSort* でも同様の処理を実装できます。
IM-BIS との連携では利用できません。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **配列**
ソート対象の配列です。
配列に含まれるオブジェクトは OpenRules が提供するクラス「ComparableDecisionVariable」を継承している必要があります。
ソートを行う際の *DecisionTable* の名前は自動的に “Compare”+配列の要素の型<type>で定義されるため、この名前の *DecisionTable* でソートの条件を指定してください。

メソッドの概要

public void sort(com.openrules.ruleengine.ComparableDecisionVariable[] array)

パラメータ

array- ソート対象の配列

戻り値

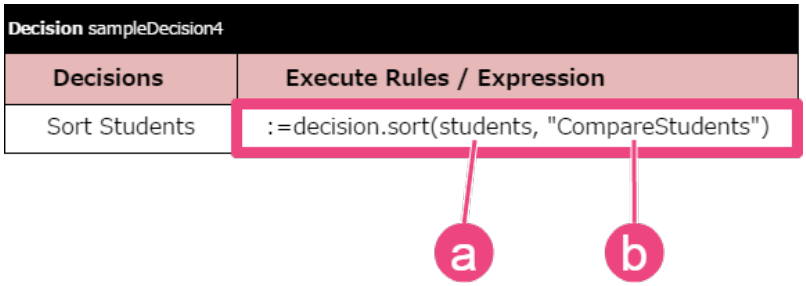
なし

sort(array, rules)

パラメータに指定した配列を *DecisionTable* の結果に基づいてソートするためのメソッドです。
テーブルタイプ *DecisionTableSort* でも同様の処理を実装できます。
IM-BIS との連携では利用できません。

記述方法

利用するTableTypeの明細部の値に記述します。



a. **配列**
ソート対象の配列です。
配列に含まれるオブジェクトは OpenRules が提供するクラス「ComparableDecisionVariable」を継承している必要があります。
ソートを行う際の *DecisionTable* の名前は自動的に “Compare”+配列の要素の型<type>で定義されるため、この名前の *DecisionTable* でソートの条件を指定してください。

b. **DecisionTableの名前**
ソートするためのスコアを算出する *DecisionTable* の名前です。

メソッドの概要

public void sort(com.openrules.ruleengine.ComparableDecisionVariable[] array, java.lang.String rules)

パラメータ

array- ソート対象の配列
rules- ソートの基準とするスコアを算出する *DecisionTable* の名前

戻り値

なし

OpenRules が提供するメソッド

DecisionTable の *Condition* や *Conclusion* で記述できるキーワードです。
OpenRules が提供するJavaクラス等をインポートすると利用できます。

- 日付型項目向けメソッド（ com.openrules.tools.Dates ）
  - Dates.yearsToday
  - Dates.years
  - Dates.monthsToday
  - Dates.months
  - Dates.daysToday
  - Dates.days

日付型項目向けメソッド（ com.openrules.tools.Dates ）

OpenRules 6.4.0で追加された日付計算を行うメソッドです。
利用する場合、 *Environment* に以下のimportを指定してください。

Environment	
include	../lib/openrules.config/IntramartTemplate.xls
import.java	com.openrules.tools.Dates

Dates.yearsToday

日付型の項目の値と現在日付を比較した差を年単位で返却するメソッドです。
条件、結果のいずれにも利用できます。

記述方法

利用するTableTypeの明細部の値に記述します。

- *ConditionAny* の例
  以下は年齢で条件判断を行う例です。

DecisionTable defineAdult

ConditionAny		Conclusion	
条件式（Conditional Expression）		メッセージ	
Is True	:= Dates.yearsToday($D{生年月日（Date of Birth）}) >= 20	Is	成人(Adult)
Is False		Is	こども(Child)

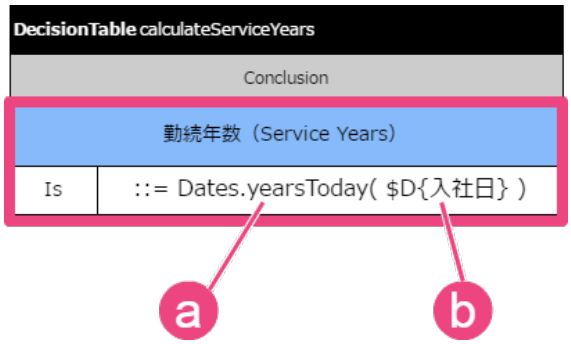
a

b

a. **クラス、メソッド名**
   日付計算のクラス、年数を返却するメソッドです。

b. **日付型の値**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

- *Conclusion* の例
  以下は入力された日付（入社日）から勤続年数を計算する例です。



a. **クラス、メソッド名**
   日付計算のクラス、年数を返却するメソッドです。

b. **日付型の値**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

パラメータ

日付（Date）型の値

論理名で指定する場合、 *$D(getDate)* のマクロを利用できます。

戻り値

整数の値

1年に満たない場合は0を返却します。
パラメータの日付が過去日付の場合、年数を正の整数で返却します。
パラメータの日付が未来日付の場合、年数を負の整数で返却します。

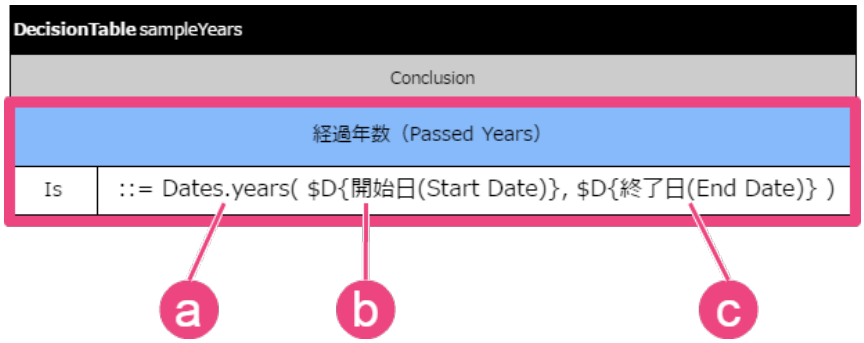
Dates.years

2つの日付型の項目の値を比較した差を年単位で返却するメソッドです。
条件、結果のいずれにも利用できます。

記述方法

利用するTableTypeの明細部の値に記述します。

このメソッドでは(c)日付型の値2から(b)日付型の値1を引いた結果を返却するため、(b)日付型の値1 > (c)日付型の値2の場合には負数を返却します。



a. **クラス、メソッド名**
   日付計算のクラス、年数を返却するメソッドです。

b. **日付型の値1**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

c. **日付型の値2**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

パラメータ

日付（Date）型の値
論理名で指定する場合、 *$D(getDate)* のマクロを利用できます。

戻り値

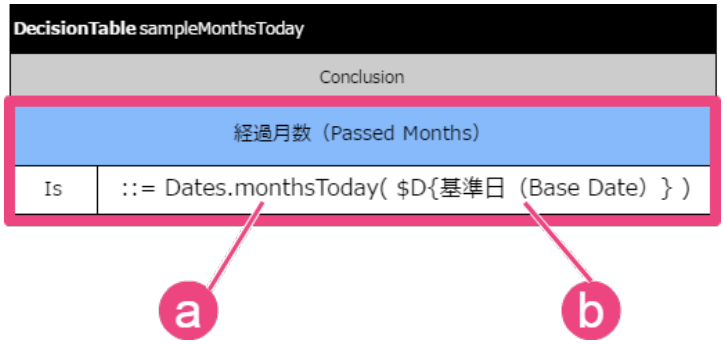
整数の値
1年に満たない場合は0を返却します。

Dates.monthsToday

日付型の項目の値と現在日付を比較した差を月単位で返却するメソッドです。
条件、結果のいずれにも利用できます。

記述方法

利用するTableTypeの明細部の値に記述します。
ConditionAny と組み合わせて条件への利用、*Conclusion* と組み合わせて結果への利用のいずれも可能です。



a. **クラス、メソッド名**
   日付計算のクラス、月数を返却するメソッドです。

b. **日付型の値**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

パラメータ

日付（Date）型の値

論理名で指定する場合、 *$D(getDate)* のマクロを利用できます。

戻り値

整数の値

1ヶ月に満たない場合は0を返却します。
パラメータの日付が過去日付の場合、月数を正の整数で返却します。
パラメータの日付が未来日付の場合、月数を負の整数で返却します。

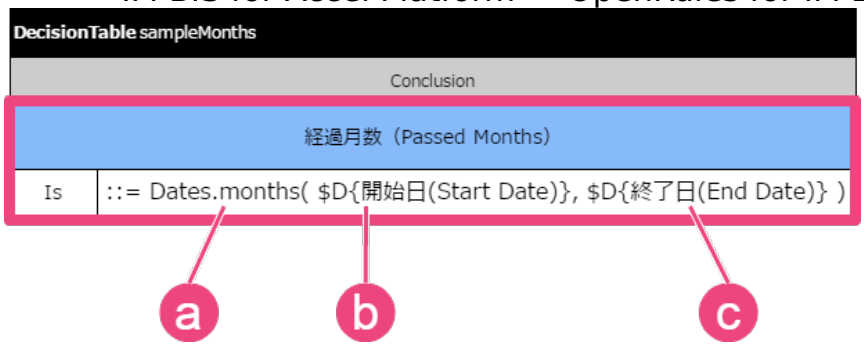
Dates.months

2つの日付型の項目の値を比較した差を月単位で返却するメソッドです。
条件、結果のいずれにも利用できます。

記述方法

利用するTableTypeの明細部の値に記述します。

このメソッドでは(c)日付型の値2から(b)日付型の値1を引いた結果を返却するため、(b)日付型の値1 > (c)日付型の値2の場合には負数を返却します。



a. **クラス、メソッド名**
   日付計算のクラス、月数を返却するメソッドです。

b. **日付型の値1**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

c. **日付型の値2**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

パラメータ

日付（Date）型の値
論理名で指定する場合、 *$D(getDate)* のマクロを利用できます。

戻り値

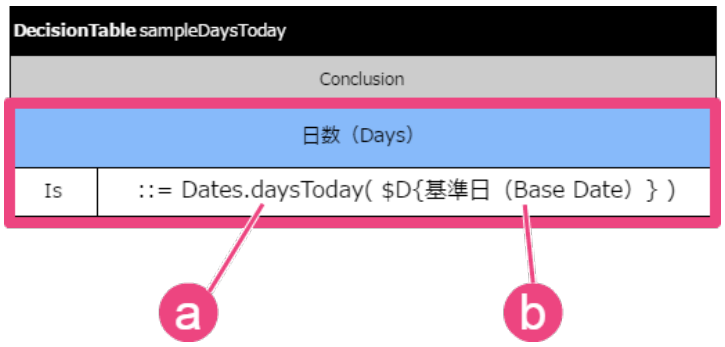
整数の値
1ヶ月に満たない場合は0を返却します。

Dates.daysToday

日付型の項目の値と現在日付を比較した差を日単位で返却するメソッドです。
条件、結果のいずれにも利用できます。

記述方法

利用するTableTypeの明細部の値に記述します。
ConditionAny と組み合わせて条件への利用、*Conclusion* と組み合わせて結果への利用のいずれも可能です。



a. **クラス、メソッド名**
   日付計算のクラス、日数を返却するメソッドです。

b. **日付型の値**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

パラメータ

日付（Date）型の値

論理名で指定する場合、 *$D(getDate)* のマクロを利用できます。

戻り値

整数の値

1ヶ月に満たない場合は0を返却します。
パラメータの日付が過去日付の場合、日数を正の整数で返却します。
パラメータの日付が未来日付の場合、日数を負の整数で返却します。

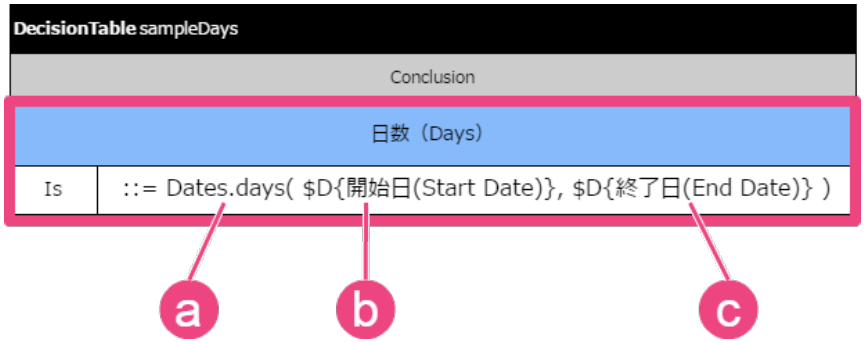
Dates.days

2つの日付型の項目の値を比較した差を日単位で返却するメソッドです。
条件、結果のいずれにも利用できます。

記述方法

利用するTableTypeの明細部の値に記述します。

このメソッドでは(c)日付型の値2から(b)日付型の値1を引いた結果を返却するため、(b)日付型の値1 > (c)日付型の値2の場合には負数を返却します。



a. **クラス、メソッド名**
   日付計算のクラス、日数を返却するメソッドです。

b. **日付型の値1**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

c. **日付型の値2**
   日付（Date）型の値です。
   $D(getDate) を利用した項目、または日付（Date）型の値を指定できます。

パラメータ

日付（Date）型の値
論理名で指定する場合、 *$D(getDate)* のマクロを利用できます。

戻り値

整数の値
1日に満たない場合は0を返却します。

OpenRules のバージョンによる記法の差異

IM-BIS と OpenRules をアップデートする場合、 OpenRules のバージョンによって追加・変更が発生する事項をまとめています。
アップデートを行う際には、本項の内容に基づいて必要な対応を実施してください。

なお、各項目の「利用できる OpenRules のバージョン」は、ユーザモジュールに含まれるバージョンに基づいて記載しているため、 OpenRules 本体で追加・変更されたバージョンと差異がある場合があります。

> **コラム**
>
> 本項の内容は OpenRules 本体の仕様に関する内容のため、以下のリンク先のリリースノートやユーザマニュアルで詳細を参照してください。
>
> - OpenRules （公式サイト）
> - OpenRules User Manual

- テーブルタイプ
  - DecisionTable
  - DecisionTable1
  - DecisionTable2
  - DecisionTableAssign
- 条件として利用できるキーワード
  - *Condition*
  - *ConditionIntOperInt*
  - *ConditionRealOperReal*
  - *ConditionDateOperDate*
- 結果・処理として利用できるキーワード
  - *Conclusion*
  - *ActionExecute*
- 特殊なキーワード / テーブルタイプ固有のキーワード
- Methodで利用できるキーワード（API）
  - 全般

テーブルタイプ

DecisionTable

キーワードで「DecisionTableSingleHit」が利用可能

以下の表で ✔ のある OpenRules のバージョンでは、 *DecisionTable* で「DecisionTableSingleHit」を利用できます。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.4	✖
OpenRules 6.2.6 以降	✔

列タイプ「OnOff」が利用可能

以下の表で ✔ のある OpenRules のバージョンでは、 *DecisionTable* の列タイプとして「OnOff」を利用できます。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.3 以前	✖
OpenRules 6.4.2 以降	✔

DecisionTable1

キーワードで「DecisionTableMultiHit」が利用可能

以下の表で ✔ のある OpenRules のバージョンでは、 *DecisionTable1* で「DecisionTableMultiHit」を利用できます。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.4	✖
OpenRules 6.2.6 以降	✔

列タイプ「OnOff」が利用可能

以下の表で ✔ のある OpenRules のバージョンでは、 *DecisionTable1* の列タイプとして「OnOff」を利用できます。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.3 以前	✖

OpenRules	利用可否
OpenRules 6.4.2 以降	✔

DecisionTable2

列タイプ「OnOff」が利用可能

以下の表で ✔ のある OpenRules のバージョンでは、 *DecisionTable2* の列タイプとして「OnOff」を利用できます。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.3 以前	✖
OpenRules 6.4.2 以降	✔

DecisionTableAssign

テーブルタイプ「DecisionTableAssign」が利用可能

以下の表で ✔ のある OpenRules のバージョンでは、テーブルタイプとして *DecisionTableAssign* が利用できます。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.3 以前	✖
OpenRules 6.4.2 以降	✔

> **コラム**
>
> OpenRules 6.4.2では、配列型のオブジェクトを扱う下記のテーブルタイプも追加されていますが、 IM-BIS との連携では利用できません。
>
> - DecisionTableIterate
> - DecisionTableSort

条件として利用できるキーワード

Condition

数値範囲の記載方法に[1..n]形式を追加

OpenRules 6.3.0では、数値範囲の表現方法で[5,15]（5～15の範囲を表す）と同義の形式として[5..15]の形式でも記載できます。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.6 以前	✖
OpenRules 6.3.0 以降	✔

数値比較の演算子に「Outside / Outside Interval」を追加

OpenRules 6.3.2で提供された演算子「Outside / Outside Interval」が利用できます。
詳細については該当のキーワードのリファレンスを参照してください。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.1 以前	✖
OpenRules 6.3.3 以降	✔

文字列比較の演算子として「Does Not Contain」が利用可能

OpenRules 6.3.3で修正された演算子「Does Not Contain」が利用できます。

当該バージョンより前のバージョンで定義した場合、利用できません。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.1 以前	✖
OpenRules 6.3.3 以降	✔

数値比較の演算子として「Is One Of」「 Is Not One Of」が利用可能

OpenRules 6.4.0で修正された演算子「Is One Of」「Is Not One Of」が利用できます。
当該バージョンより前のバージョンで定義した場合、利用できません。
詳細については該当のキーワードのリファレンスを参照してください。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.3 以前	✖
OpenRules 6.4.2 以降	✔

ConditionIntOperInt

特定のデータ型の項目（*Data/Variable*）同士を比較するためのキーワードの追加

OpenRules 6.3.0では、特定のデータ型の項目同士を簡単に比較するためのキーワードとして、「*ConditionIntOperInt*」を追加しました。
詳細については該当のキーワードのリファレンスを参照してください。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.6 以前	✖
OpenRules 6.3.0 以降	✔

ConditionRealOperReal

特定のデータ型の項目（*Data/Variable*）同士を比較するためのキーワードの追加

OpenRules 6.3.0では、特定のデータ型の項目同士を簡単に比較するためのキーワードとして、「*ConditionRealOperReal*」を追加しました。
詳細については該当のキーワードのリファレンスを参照してください。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.6 以前	✖
OpenRules 6.3.0 以降	✔

ConditionDateOperDate

特定のデータ型の項目（*Data/Variable*）同士を比較するためのキーワードの追加

OpenRules 6.3.0では、特定のデータ型の項目同士を簡単に比較するためのキーワードとして、「*ConditionDateOperDate*」を追加しました。
詳細については該当のキーワードのリファレンスを参照してください。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.6 以前	✖
OpenRules 6.3.0 以降	✔

結果・処理として利用できるキーワード

Conclusion

式の記述時の（）の扱いが変更

OpenRules 6.3.0では、::=で始まる式を *Conclusion* などと組み合わせて記載する場合、式を（）で囲わなくてもよい形に変更になりました。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.6 以前	✖
OpenRules 6.3.0 以降	✔

日付クラス（ com.openrules.tools.Dates ）の追加

OpenRules 6.4.0で日付型項目を扱うためのクラスが追加されました。
このクラスには日数計算などのメソッドが含まれています。
詳細については以下のリファレンスを参照してください。

- *日付型項目向けメソッド（ com.openrules.tools.Dates ）*
- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.3 以前	✖
OpenRules 6.4.2 以降	✔

ActionExecute

Decision からの Sub-Decision(*Decision*)、*DecisionTable* の呼び出し方法の変更

OpenRules 6.2.6では、 *Decision* からの呼び出し方法が変更されました。
OpenRules 6.2.6以降のバージョンの環境で、OpenRules 6.2.4以前に対応した記述でのルールを実行すると、シンタックスエラーが発生します。

- OpenRules 6.2.4以前での記述方法

Decision sampleRules	
ActionPrint	ActionExecute
処理名	実行する処理
sub-decisionの実行	:= mySubDecision(decision)
DecisionTableの実行	:= sampleDecisionTable()
結果の返却	:= decision().put("ResponseObject", responseObj)

```
// Sub-Decision（Decisionから別のDecisionの呼び出し）
 := DecisionName(decision)

// DecisionTable（DecisionからのDecisionの呼び出し）
 := DecisionTableName()
```

  - 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.4	✔
OpenRules 6.2.6 以降	✖

- OpenRules 6.2.6以降での記述方法

Decision sampleRules	
ActionPrint	ActionExecute
処理名	実行する処理
sub-decisionの実行	mySubDecision
DecisionTableの実行	sampleDecisionTable
結果の返却	:= decision().put("ResponseObject", responseObj)

```
// Sub-Decision（Decisionからの別のDecisionの呼び出し）
 DecisionName

// DecisionTable（DecisionからのDecisionの呼び出し）
 DecisionTableName
```

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.4	✖
OpenRules 6.2.6 以降	✔

特殊なキーワード / テーブルタイプ固有のキーワード

現時点では、差異はありません。

Methodで利用できるキーワード（API）

全般

すべてのget/set/compareメソッドの呼び出し変更、getメソッドの代替マクロの追加

すべてのget/set/compareメソッドについて、実行されるメソッドがExcelファイル（DecisionTemplates.xls、DecisionTableExecuteTemplates.xls）からJavaに移行されたことに伴い、呼び出し方法の書き方が変わります。
また、getメソッドの代替の記述方法としてマクロが追加されました。

- OpenRules 6.2.6以前での記述方法

  以下の記述方法は、下位互換を保持するために OpenRules 6.3.0でも動作しますが、非推奨メソッドとして扱います。

DecisionTable sampleRule1	
Conclusion(結果)	
計算結果	
Is	::= getInt("項目A")

```
// get()の例

 getInt("項目の論理名")
```

  - 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.6 以前	✔
OpenRules 6.3.0 以降	△（非推奨）

- OpenRules 6.3.0以降での記述方法

DecisionTable sampleRule1	
Conclusion(結果)	
計算結果	
Is	::= decision.getInt("項目A")
Is	::= $I("項目A")

- OpenRules 6.3.0以降のバージョンでは、上の形式でget/set/compareメソッドを記述してください。
  $で始まるマクロはgetメソッドのみに対応しています。

```
// get()の例

  decision.getInt("項目の論理名")
  $I{項目の論理名}
```

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.6 以前	✖
OpenRules 6.3.0 以降	✔

- 追加されたマクロ（$始まりでget()を実行する表現形式）
  - *$(getString)*
  - *$I(getInt)*
  - *$R(getReal)*
  - *$D(getDate)*
  - *$B(getBool)*
  - *$V(getVar)*

Method からの *Decision* インスタンスに含まれるパラメータへのアクセス方法の変更

Glossary で定義した *Business Concept* や *Variable（Glossary）* を *Method* で扱う場合には *Decision* インスタンスからアクセスします。
OpenRules 6.3.0では、この *Decision* インスタンスへのアクセス方法の記述が変更されました。
OpenRules 6.2.6以前での記述方法を利用している場合、複数のスレッドで同時にルールを実行する際に正しく対象のインスタンスにアクセスできないなどの問題がありますので、注意してください。

- OpenRules 6.2.6以前での記述方法

```
Method  Customer customer()

return (Customer) decision().get("customer";)
```

- OpenRules 6.3.0以降での記述方法

```
Method Customer customer(Decision decision)

return (Customer) decision.get("customer";)
```

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.2.6 以前	✖
OpenRules 6.3.0 以降	✔

BigDecimal、longに対するマクロを追加

OpenRules 6.3.4からBigDecimal、 6.4.1からlongのデータ型に対するgetメソッドのマクロが提供されました。
ただし、 IM-BIS が対応していないため、 IM-BIS との連携では利用できません。

- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.3 以前	✖

OpenRules	利用可否
OpenRules 6.4.2 以降	✔

配列に対する反復処理、ソート処理に関するメソッドを追加

OpenRules 6.4.2から配列に対する反復処理、ソート処理を行うためのメソッドが提供されました。
ただし、 IM-BIS が対応していないため、 IM-BIS との連携では利用できません。

- iterate(String arrayName, String rules)
- iterate(String arrayName, String arrayType, String rules)
- sort(String arrayName)
- sort(String arrayName, String rules)
- sort(ComparableDecisionVariable[] array)
- sort(ComparableDecisionVariable[] array, String rules)
- 利用できる OpenRules のバージョン

OpenRules	利用可否
OpenRules 6.3.3 以前	✖
OpenRules 6.4.2 以降	✔

IM-BIS / OpenRules のバージョン対応表

IM-BIS と対応する OpenRules のバージョンは、本章の通りです。
（ IM-Juggling でのユーザモジュールに含まれる OpenRules のバージョンを記載しています。）

IM-BIS	OpenRules	ユーザモジュール
2013 Summer（8.0.2）	OpenRules 6.2.4	OpenRules 機能強化モジュール
2013 Winter（8.0.3）	OpenRules 6.2.6	OpenRules 機能強化モジュール
2014 Spring（8.0.4）	OpenRules 6.3.0	jp.co.intra_mart.oss_linkage.openrules-8.0.0.imm
2014 Summer（8.0.5） 2014 Winter（8.0.6） 2015 Spring（8.0.7） 2015 Summer（8.0.8）	OpenRules 6.3.1	jp.co.intra_mart.oss_linkage.openrules-8.0.1.imm
2015 Winter（8.0.9）以降 [1]	OpenRules 6.3.3	jp.co.intra_mart.oss_linkage.openrules-8.0.2.imm
	OpenRules 6.4.2	jp.co.intra_mart.oss_linkage.openrules-8.0.3.imm
	OpenRules 7.0.0	jp.co.intra_mart.oss_linkage.openrules-8.0.4.imm

コラム

IM-BIS 8.0.9 以降をご利用中で、 OpenRules 6.3.3から OpenRules をアップデートしたい場合には、「 プロダクトファイルダウンロード 」で提供しているIMMファイルを IM-Juggling 上で差し替えてから再デプロイしてください。
「 プロダクトファイルダウンロード 」からのIMMファイルのダウンロードにはライセンスキーが必要です。

[1] IM-BIS 8.0.9以降の場合、ユーザモジュールのバージョンに基づいて OpenRules のバージョンが異なります。

OpenRules 7.0.0 注意事項

OpenRules 7.0.0 では「 *Decision* 」内でdecisionインスタンスのputメソッドを実行できなくなりました。
そのため、putメソッドを別メソッド内で実行する必要があります。
本ドキュメント内のハンズオンのサンプルモジュールは、下記修正が反映されています。
また、各章のハンズオンは、OpenRules 7.0.0 を利用する場合、以下の記述部分は置き換えて参照してください。

- OpenRules 6.4.2以前での記述方法

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
入力項目の内容の表示	:= System.out.println(getDecisionObject("InputObject"))
評価の実行	myRule
結果の取得	:= decision().put("OutputObject", outputObj)
出力項目の内容の表示	:= System.out.println(getDecisionObject("OutputObject"))

- OpenRules 7.0.0以降での記述方法

Decision myDecision	
ActionPrint	ActionExecute
処理名	実行する処理内容
入力項目の内容の表示	:= System.out.println(getDecisionObject("InputObject"))
評価の実行	myRule
結果の取得	DefineOutput
出力項目の内容の表示	:= System.out.println(getDecisionObject("OutputObject"))

Method void DefineOutput(Decision decision)
decision.put("OutputObject", outputObj[0]);

ダウンロード

付録では、ルールを定義するExcelのテンプレートや各章のハンズオンの完成版のモジュールをダウンロードすることができます。
下記の各リンクからダウンロードしたファイルは、「 IM-BIS システム管理者操作ガイド 」を参考にしてインポートしてからご利用ください。

ハンズオンのサンプルモジュール（完成版）

各ハンズオンの完成版の各種定義ファイルをダウンロードすることができます。

> 注意
>
> - ハンズオンの手順のページに掲載されているハンズオン用の定義ファイルから設定を追加した定義ファイルのため、同一環境に同じハンズオンの実践用定義と完成版の定義をインポートしないでください。
> - 本項のサンプルモジュールは、 IM-BIS 2014 Winter-PATCH_002以降のバージョンの環境で利用することができます。
> - 本項のサンプルモジュールはデータベースが PostgreSQL の環境で作成されているため、その他のデータベースの場合にはアプリケーション定義のインポート時にエラーが発生する可能性があります。
>   エラーが発生した場合には、以下の手順でテーブルを再作成してください。
>   1. インポート後に、「 IM-BIS - 一覧 」の画面から「アプリ」をクリックしてください。
>   2. 「テーブル設定」をクリックしてください。
>   3. 「削除」をクリックして不正な状態になっているテーブルを削除してください。
>   4. 「 IM-BIS - 一覧 」に戻り、「編集」をクリックしてください。
>   5. 「定義の反映」を実行してください。
>
>   以上でエラーが解消し、ご利用いただくことができます。

まずは IM-BIS 連携を実行してみよう

- BIS定義
  bis_hello_openrules.zip
- Formaアプリケーション定義
  forma_hello_openrules.zip
- IM-Workflow 定義
  imw_hello_openrules.zip
- データソース定義ファイル（Excelファイルが含まれています。）
  datasource_hello_openrules.zip

複合条件（AND/OR）を利用して、旅費精算の日当を計算してみよう

- BIS定義ファイル
  bis_travel_expenses.zip
- Formaアプリファイル
  forma_travel_expenses.zip
- IM-Workflow の定義ファイル
  imw_travel_expenses.zip
- データソース定義ファイル
  datasource_travel_expenses.zip

自動車保険の計算と入力チェックをしてみよう

- BIS定義ファイル
  bis_auto_insurance.zip
- Formaアプリファイル
  forma_auto_insurance.zip
- IM-Workflow の定義ファイル
  imw_auto_insurance.zip
- データソース定義ファイル（Excelファイルが含まれています。）
  datasource_auto_insurance.zip

複雑な条件の実行や計算を行うルールを作成してみよう

- BIS定義ファイル
  bis_credit_management.zip
- Formaアプリファイル
  forma_credit_management.zip

- IM-Workflow の定義ファイル
  imw_credit_management.zip
- データソース定義ファイル（Excelファイルが含まれています。）
  datasource_credit_management.zip

入力チェックされた金額に応じて、ルートの形を変えてみよう

- BIS定義ファイル
  bis_request_for_approval.zip
- Formaアプリファイル
  forma_request_for_approval.zip
- IM-Workflow の定義ファイル
  imw_request_for_approval.zip
- データソース定義ファイル（Excelファイルが含まれています。）
  datasource_request_for_approval.zip

OpenRules のテンプレート

汎用テンプレート

sampleTemplate.zip

動的処理者設定テンプレート

Zipファイル形式で圧縮しておりますので、解凍してからご利用ください。

BIS_DynamicUser_SampleRule.zip

各種定義ファイルのインポートの手順

当ガイドに添付されている各種サンプルなどを利用するためには、必要な定義ファイルをインポートしてください。
インポート実行後には、ハンズオンシナリオやフローの実行など必要な作業を実施してください。

インポートの手順

- IM-Workflow 定義ファイル
- IM-BIS 定義ファイル
- Formaのアプリケーションの定義ファイル
- データソース定義ファイル

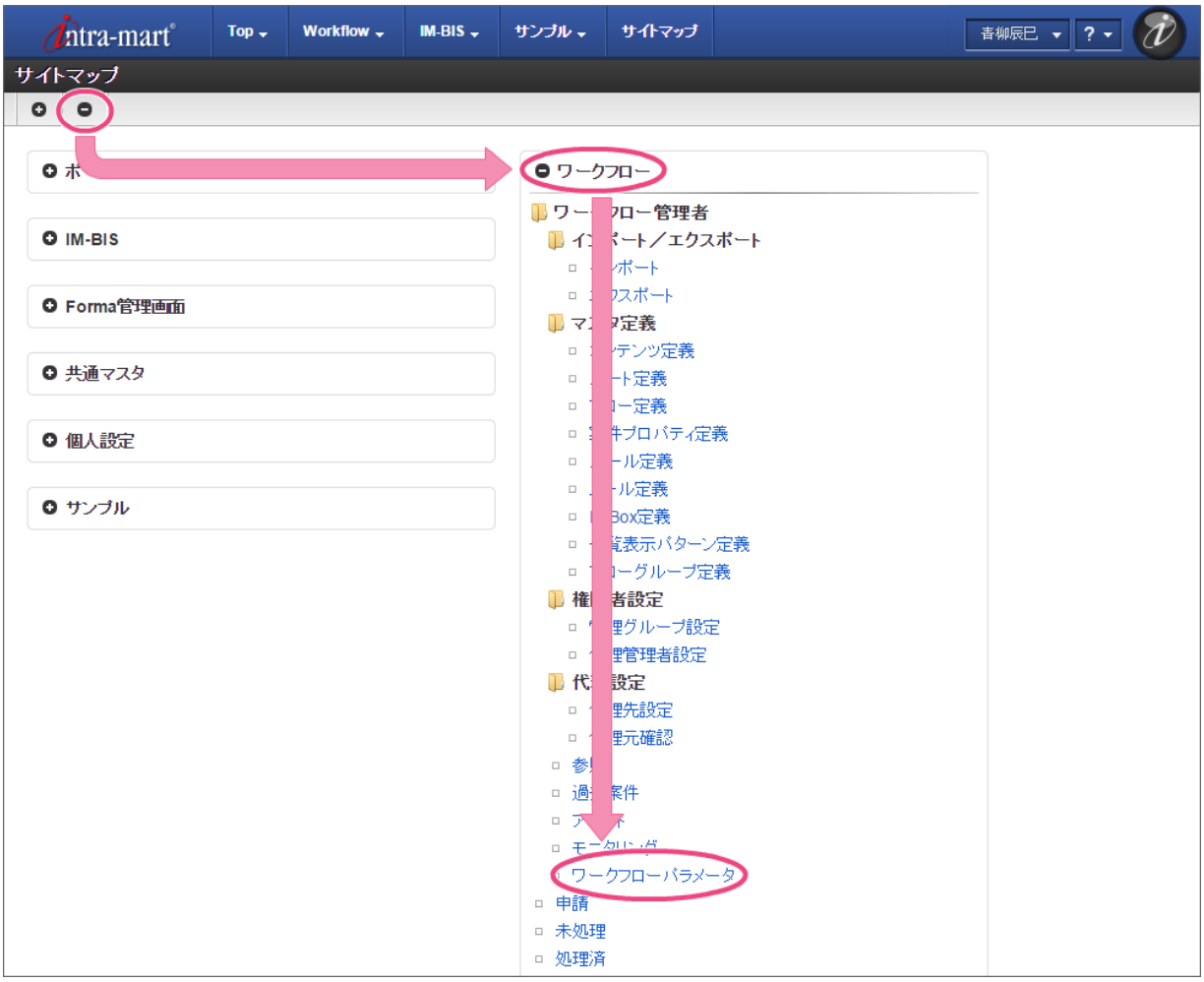
IM-Workflow 定義ファイル

IM-Workflow の定義ファイルをローカルからインポートするための設定を変更する

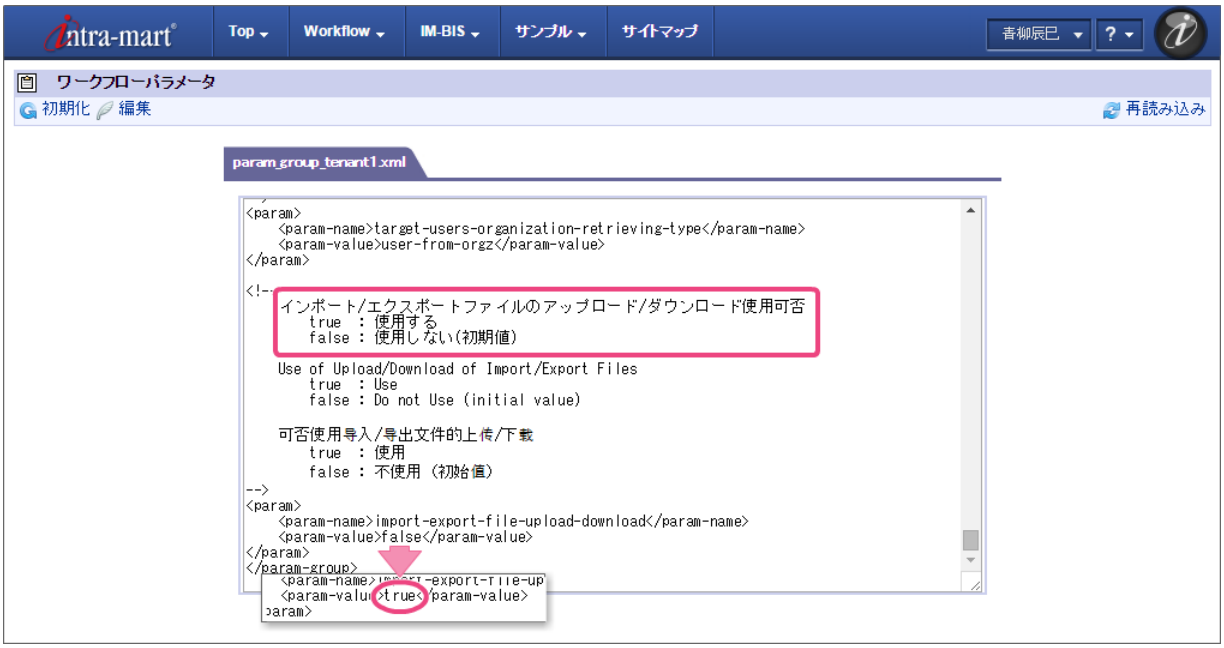
IM-Workflow の定義ファイルをインポートする場合には、対象のファイルをパブリックストレージに格納する必要があります。
ワークフローパラメータを変更すると、ローカルからパブリックストレージにアップロードした上でインポートできます。

> **コラム**
>
> ワークフローパラメータを変更しない場合には、以下のリンク先を参照して、「 IM-Workflow 定義」のファイルをパブリックストレージの規定の場所に格納してください。
>
> - 「 IM-BIS システム管理者操作ガイド 」-「 インポート・エクスポートを行う 」

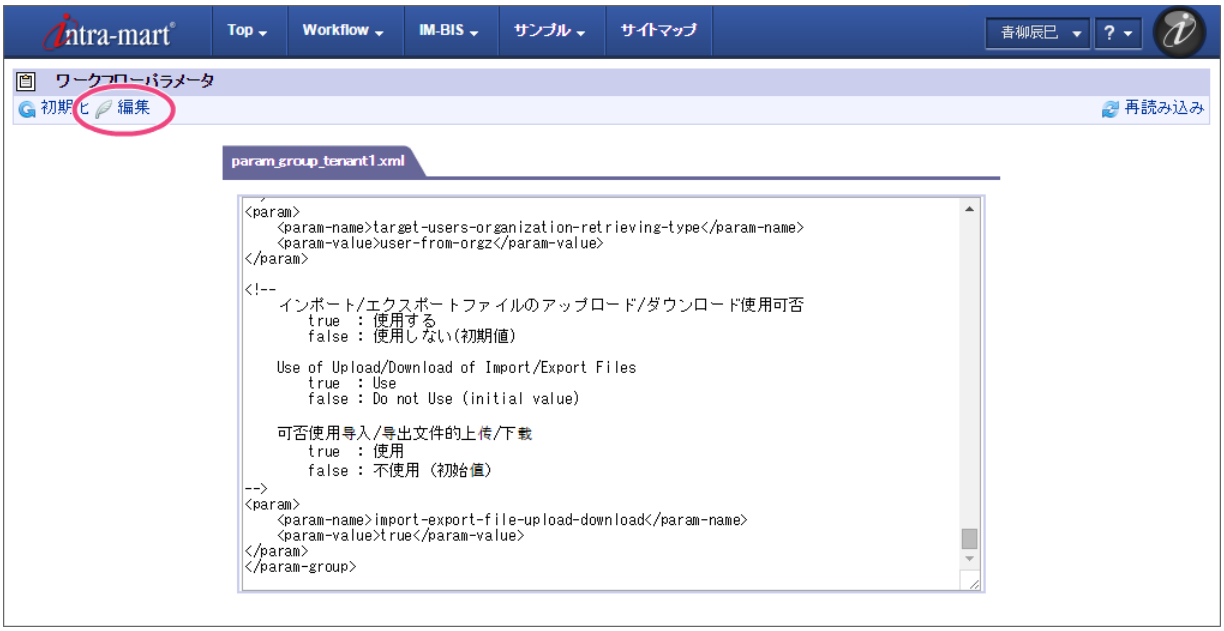
1. 「BIS管理者ロール」を持ったユーザでログインしてください。
2. サイトマップの「ワークフロー」から「ワークフローパラメータ」をクリックしてください。



3. 「ワークフローパラメータ」で「インポート/エクスポートファイルのアップロード/ダウンロード使用可否」の設定値を「false」から「true」に変更してください。



4. 「編集」をクリックして変更内容を保存してください。



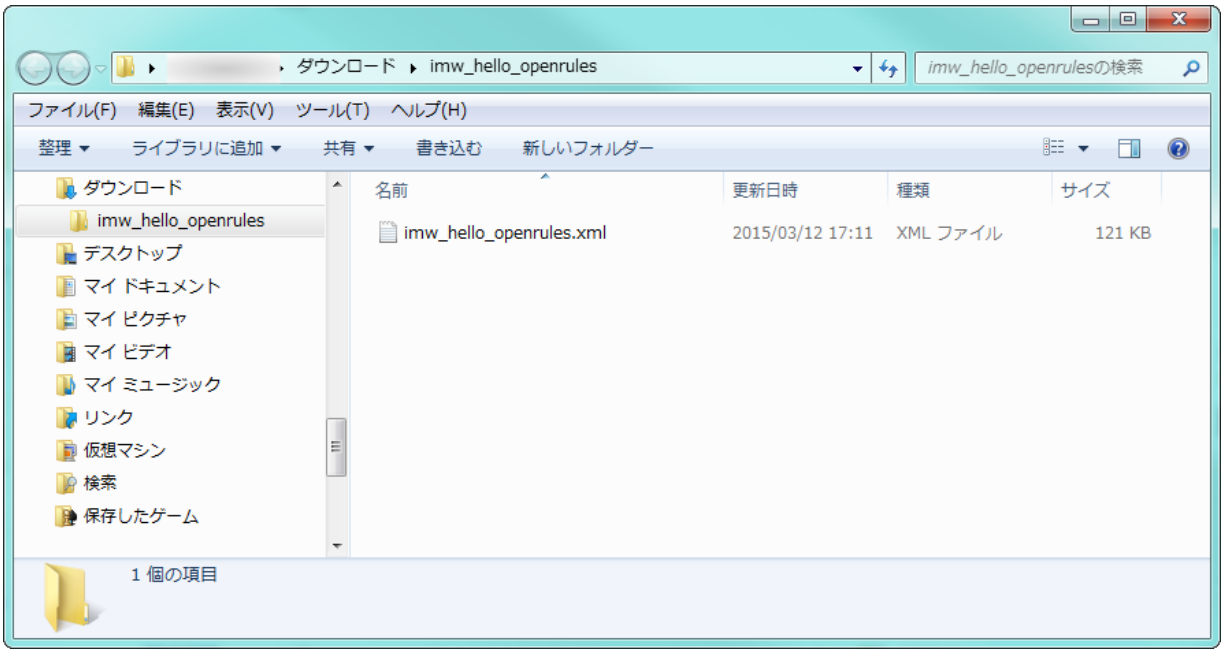
5. これで、ローカルから IM-Workflow の定義ファイルをインポートする準備が整いました。

IM-Workflow の定義ファイルをローカルからインポートする

IM-Workflow の定義ファイルをインポートします。

1. 当ガイドから対象のファイルをダウンロードしてください。

2. ダウンロードしたZipファイルを解凍してください。



3. サイトマップの「 IM-BIS 」から「 IM-Workflowインポート」をクリックしてください。

4. 「ファイルを選択」をクリックし、先ほど解凍したファイルに含まれているXMLファイルを指定してください。

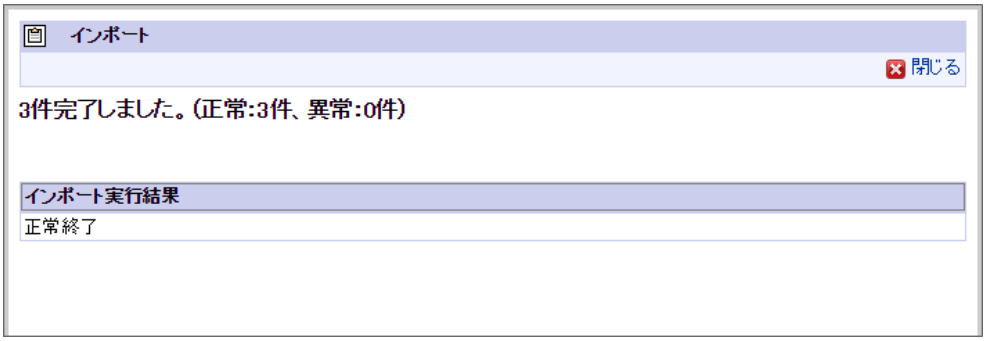
5. 追加したXMLファイル名が表示されていることを確認し、「追加」をクリックしてください。

6. 追加したファイルのチェックボックスをオンにし、「データ選択へ」をクリックしてください。

7. 「全選択」をクリックしてください。

8. 「インポート」をクリックしてください。

9. 以下のように表示されたら、 IM-Workflow の定義ファイルが正常にインポートできました。



IM-BIS 定義ファイル

IM-BIS の定義ファイルをインポートします。

1. 当ガイドから対象のファイルをダウンロードしてください。

2. サイトマップの「 IM-BIS 」から「 IM-BISインポート」をクリックしてください。

3. 「ファイルを選択」をクリックし、先ほどダウンロードしたファイルを選択してください。

4. 「インポート実行」をクリックしてください。

5. 以下のように表示されたら、 IM-BIS の定義ファイルが正常にインポートできました。

Formaのアプリケーションの定義ファイル

Formaのアプリケーションの定義ファイルをインポートします。

1. 当ガイドから対象のファイルをダウンロードしてください。

2. サイトマップの「 IM-BIS 」から「 IM-Formaアプリインポート」をクリックしてください。

3. 「ファイルを選択」をクリックし、インポートファイル（ローカル）に、先ほどダウンロードしたFormaのアプリケーションの定義ファイルを選択してください。

4. 「インポート実行」をクリックしてください。

5. 以下のように表示されたら、Formaのアプリケーションの定義ファイルが正常にインポートできました。

データソース定義ファイル

データソース定義のファイルをインポートします。

1. 当ガイドから対象のファイルをダウンロードしてください。

2. サイトマップの「 IM-BIS 」から「データソース定義インポート」をクリックしてください。

3. 「ファイルを選択」をクリックし、インポートファイル（ローカル）に、先にダウンロードしたデータソース定義のファイルを選択してください。

4. 「インポート実行」をクリックしてください。

5. 以下のように表示されたら、データソース定義のファイルが正常にインポートできました。