Red Hat Decision Manager 7.2

DMN モデルを使用したデシジョンサービスの作成

ガイド

Last Updated: 2023-02-21

Red Hat Decision Manager 7.2 DMN モデルを使用したデシジョンサービスの作成

ガイド

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書は、Red Hat Decision Manager 7.2 のデシジョンサービスに DMN (Decision Model and Notation) モデルを実装する方法を説明します。

目次

はじめに

ビジネスアナリストやルール作成者は、DMN (Decision Model and Notation) を使用して、意思決定要件ダイアグラム (DRD) でデシジョンサービスを視覚的にモデル化できます。このダイアグラムは、1 つまたは複数の意思決定要件ダイアグラム (DRD) で設定されており、デシジョンテーブルなど、DMN ボックス式で定義されたロジックを使用したデシジョンノードで、ビジネスデシジョンを追跡します。

Red Hat Decision Manager は、適合レベル 3 で DMN 1.2 モデルの設計およびランタイムをサポートし、適合レベル 3 で DMN 1.1 モデルはランタイムのみサポートします。Decision Central で直接 DMN モデルを設計したり、既存の DMN モデルを Red Hat Decision Manager プロジェクトにインポートしたりして、デプロイメントや実行が可能です。Decision Central にインポートした DMN 1.1 はすべて、DMN デザイナーで開かれ、保存時に DMN 1.2 モデルに変換されます。

DMN に関する詳細は、Object Management Group (OMG) の Decision Model and Notation specification を参照してください。

重要

Red Hat Decision Manager 7.2 の DMN デザイナーはテクノロジープレビュー機能で、デフォルトでは Decision Central で無効になっています。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) ではサポートされておらず、機能的に完全ではない可能性があるため、実稼働環境での使用は推奨されません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Decision Central で DMN デザイナーを有効にするには、ウィンドウ右上で **Settings → Roles** とクリックし、左側のパネルからロールを選択し、**Editors → DMN Designer → Read** とクリックしてから **Save** をクリックして変更を保存します。

Red Hat テクノロジープレビュー機能の詳細は テクノロジープレビュー機能のサポート範囲 を参照してください。

第1章 DMN (DECISION MODEL AND NOTATION)

DMN (Decision Model and Notation) は、業務的意思決定を説明してモデル化するために、OMG (Object Management Group) が確立している規格です。DMN は XML スキーマを定義して、DMN モデルを DMN 準拠のプラットフォーム間や組織間で共有し、ビジネスアナリストやビジネスルール開発者が DMN デシジョンサービスの設計と実装で協力できるようにするものです。DMN 規格は、ビジネスプロセスを開発してモデル化する BPMN (Business Process Model and Notation) 規格と類似しており、一緒に使用できます。

DMN の背景およびアプリケーションの詳細は、OMG の Decision Model and Notation specification を参照してください。

1.1. DMN 適合レベル

DMN 仕様は、ソフトウェア実装における増分の適合レベルを 3 つ定義します。特定のレベルの準拠を主張する製品は、その前の適合レベルにも準拠する必要があります。たとえば、適合レベル 3 を実装するには、適合レベル 1 および 2 でサポートされるコンポーネントにも対応する必要があります。各適合レベルの公式な定義は OMG の Decision Model and Notation specification を参照してください。

以下は、3 つの DMN 適合レベルの概要です。

適合レベル 1

DMN 適合レベル 1 の実装は、意思決定要件ダイアグラム (DRD)、デシジョンロジック、デシジョンテーブルをサポートしますが、デシジョンモデルは実行可能ではありません。式の定義には、自然言語、非体系化言語を含むすべての言語を使用できます。

適合レベル 2

DMN 適合レベル 2 の実装には、適合レベル 1 の要件のほかに、S-FEEL (Simplified Friendly Enough Expression Language) 式と、完全に実行可能なデシジョンモデルをサポートします。

適合レベル 3

DMN 適合レベル 3 の実装には、適合レベル 1 および 2 の要件のほかに、FEEL (Friendly Enough Expression Language) 式、ボックス式の完全セット、完全に実行可能なデシジョンモデルをサポートします。

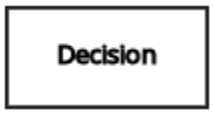
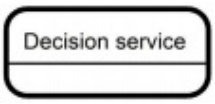
Red Hat Decision Manager は、適合レベル 3 で DMN 1.2 モデルの設計およびランタイムをサポートし、適合レベル 3 で DMN 1.1 モデルはランタイムのみサポートします。Decision Central で直接 DMN モデルを設計したり、既存の DMN モデルを Red Hat Decision Manager プロジェクトにインポートしたりして、デプロイメントや実行が可能です。Decision Central にインポートした DMN 1.1 はすべて、DMN デザイナーで開かれ、保存時に DMN 1.2 モデルに変換されます。

1.2. DMN 意思決定要件ダイアグラム (DRD) のコンポーネント

デシジョン要件ダイアグラム (DRD) は、DMN モデルを視覚的にしたものです。このダイアグラムは、DRD 全体の特定のドメインを表す 1 つ以上の意思決定要件グラフ (DRG) で設定されます。DRG は、デシジョンノード、ビジネスナレッジモデル、ビジネスナレッジのソース、入力データ、およびデシジョンサービスを使用して、ビジネスデシジョンを追跡します。

以下の表では、DRD のコンポーネントについてまとめています。

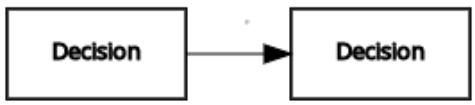
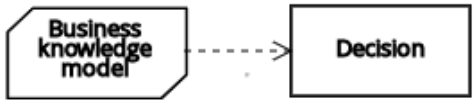
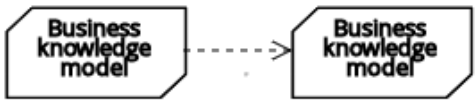
表1.1 DRD コンポーネント

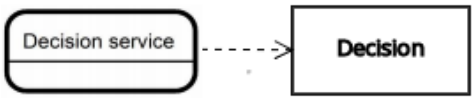
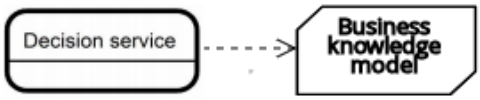
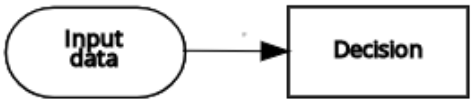
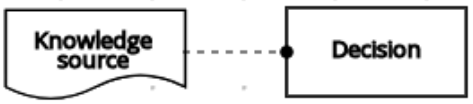
コンポーネント		説明	表記
要素	デシジョン	1つ以上の要素が定義したデシジョンロジックをもとに出力を決定するノード。	
	ビジネスナレッジモデル	1つまたは複数のデシジョン要素が含まれる再利用可能な関数。同じロジックですが、サブの入力または決定が異なるため、ビジネスナレッジモデルを使用してどの手順に従うかを決定します。	
	ナレッジソース	デシジョンまたはビジネスナレッジモデルを規定する外部の機関、ドキュメント、委員会またはポリシー。ナレッジソースは、実行可能なビジネスルールではなく、実際の要因への参照となります。	
	入力データ	デシジョンノードまたはビジネスナレッジモデルで使用する情報。入力データには通常、融資戦略で使用するローン申請データなど、ビジネスに関連するビジネスレベルのコンセプトまたはオブジェクトが含まれます。	
	デシジョンサービス	呼び出しのサービスとして公開される、再利用可能なデシジョンセットを含むトップレベルのデシジョン。デシジョンサービスは、外部アプリケーションまたは BPMN ビジネスプロセスから呼び出し可能です。  **注記** デシジョンサービスノードは、現在 Red Hat Decision Manager 7.2 DMN デザイナーではサポートされていません。このサポートは今後のリリースで提供されます。	
要件コネクター	情報要件	情報を必要とする別のデシジョンノードへの入力データノードまたはデシジョンノードからの接続	

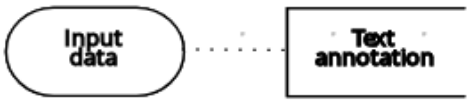
コンポーネント	説明		表記
	ナレッジ要件	デシジョンロジックを呼び出す別のビジネスナレッジモデルまたはデシジョンノードへのビジネスナレッジモデルからの接続	
	認証局の要件	入力データノードまたはデシジョンノードから従属するナレッジソース、またはナレッジソースからデシジョンノード、ビジネスナレッジモデル、または別のナレッジソースへの接続	
アーティファクト	テキストのアノテーション	入力データノード、デシジョンノード、ビジネスナレッジモデル、またはナレッジソースに関連する注釈	
	関連付け	入力データノード、デシジョンノード、ビジネスナレッジモデル、またはナレッジソースからテキストアノテーションへの接続	

以下の表では、DRD 要素間で使用可能なコネクターについてまとめています。

表1.2 DRD コネクタールール

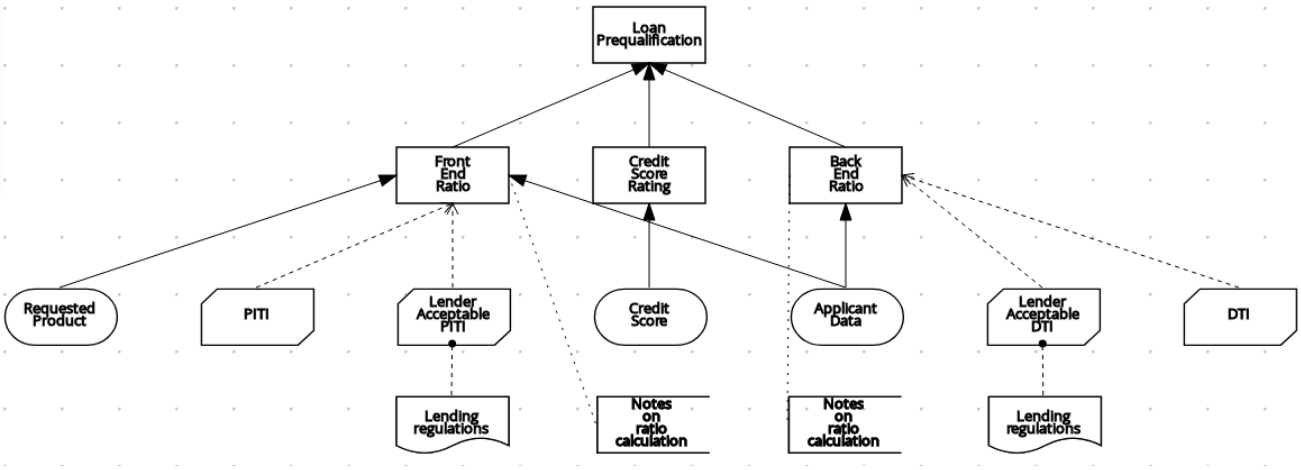
接続元	接続先	接続の種類	例
デシジョン	デシジョン	情報要件	
ビジネスナレッジモデル	デシジョン	ナレッジ要件	
	ビジネスナレッジモデル		

接続元	接続先	接続の種類	例
デシジョンサービス **(現時点では、Red Hat Decision Manager 7.2 DMN デザイナーではサポートされていません。サポートは今後のリリースで提供されます。)**	デシジョン	ナレッジ要件	
	ビジネスナレッジモデル		
入力データ	デシジョン	情報要件	
	ナレッジソース	認証局の要件	
ナレッジソース	デシジョン	認証局の要件	
	ビジネスナレッジモデル		
	ナレッジソース		
デシジョン	テキストのアノテーション	関連付け	
ビジネスナレッジモデル			

接続元	接続先	接続の種類	例
ナレッジソース			
入力データ			

以下の DRD は、これらのコンポーネントの実際の使用例です。

図1.1 DRD 例: ローンの事前審査



1.3. FEEL を使用したルール表現

FEEL (Friendly Enough Expression Language) は、オブジェクトマネージメントグループ (OMG: Object Management Group) の DMN 仕様が定義する式言語です。FEEL 式は DMN モデルを使用して、意思決定のロジックを定義します。FEEL は、デシジョンモデル設定概念にセマンティクスを割り当てて、意思決定のモデル化および実行を容易にすることを目的としています。意思決定要件ダイアグラム (DRD) の FEEL 式は、デシジョンノードおよびビジネスナレッジモデルのボックス式のテーブルセルで使用されます。

DMN における FEEL の詳細は、OMG の Decision Model and Notation specification を参照してください。

1.3.1. FEEL の変数および関数名

従来の多くの式言語と異なり、FEEL (Friendly Enough Expression Language) は、変数および関数名でスペースと少数の特殊文字をサポートします。FEEL 名は **文字**、**?**、または _ の要素で始める必要があります。ユニコード文字も使用できます。変数名は、言語キーワード (**and**、**true**、**every** など) で開始することはできません。先頭以外には **(複数桁の) 数値**、空白文字、特殊文字 (**+**、**-**、/、*、'、**.** など) を使用できます。

たとえば、以下の名前はすべて有効な FEEL 名です。

- Age

- Birth Date
- Flight 234 pre-check procedure

FEEL の変数名および関数名には、いくつかの制約が適用されます。

曖昧性 (多義性)

名前の一部に、スペース、キーワード、およびその他の特殊文字を使用して FEEL に多義性を持たせることができます。この多義性は、式のコンテキストで左から右に名前を一致させて解決されます。パーサーは、変数名を、その範囲に一致する中で一番長い名前に解決します。必要に応じて、**( )** を使用して名前の多義性を排除できます。

名前で使用されるスペース

DMN 仕様では、FEEL 名におけるスペース使用を制限します。DMN 仕様によると、名前には複数のスペースを使用できますが、連続して使用することはできません。
言語を使いやすく、スペース使用に関するよくある誤りを回避するために、Red Hat Decision Manager では、スペースを連続して使用する制限が取り除かれています。Red Hat Decision Manager では、スペースを連続して使用する変数名がサポートされますが、スペースを連続して使用してもスペースの数は 1 つに正規化されます。Red Hat Decision Manager の変数参照では、**First Name** および **First Name** の両方が使用できます。

また、Red Hat Decision Manager では、Web ページ、タブ、改行でよく見られる分割できない空白文字などの使用を正規化します。Red Hat Decision Manager の FEEL エンジンの観点では、このような文字はすべて、処理される前に 1 つの空白文字に正規化されます。

キーワードの in

キーワードの **in** は、この言語の中で、唯一変数名に使用できないキーワードです。仕様では、変数名にキーワードを使用できますが、変数名に **in** を使用すると、**for**、**every**、**some** の各表現概念と矛盾します。

1.3.2. FEEL のデータ型

FEEL (Friendly Enough Expression Language) では、以下のデータ型がサポートされます。

- 数値
- 文字列
- ブール値
- 日付
- 時間
- 日時
- 日時で指定する期間
- 年および月で指定する期間
- 関数
- コンテキスト
- 範囲 (または間隔)

- リスト

注記

FEEL では、DMN 仕様で変数を **function**、**context**、**range**、または **list** として宣言する明示的な方法はありませんが、Red Hat Decision Manager では、これらの種類の変数をサポートするように DMN 型が拡張されています。

以下は、各データ型の説明です。

数値

数値は、FEEL では IEEE 754-2008 の 10 進法の 128 形式 (34 桁) に基づいています。内部的には、数値は Java の **MathContext DECIMAL128** を持つ **BigDecimals** として表されます。FEEL でサポートされる数値データ型は 1 つしかないため、整数と浮動小数点には同じ型が使用されます。FEEL では、小数点の記号にドット (.) が使用されます。**-INF**、**+INF**、または **NaN** はサポートされません。FEEL では、**null** を使用して、無効な数字を表します。

Red Hat Decision Manager では、DMN 仕様が拡張され、以下の数値表記法もサポートされます。

- **科学的記数法:** 接尾辞 **e<exp>** または **E<exp>** を付けて、科学的記数法を使用できます。たとえば、**1.2e3** は **1.2*10**3** と表記するのと同じですが、式ではなくリテラルを使用しています。
- **16 進数:** プリフィックス **0x** を付けて、16 進数を使用できます。たとえば、**0xff** は、10 進数の **255** と同じです。大文字および小文字いずれもサポートされます。たとえば、**0XFF** は **0xff** と同じです。
- **型の接尾辞:** 型の接尾辞として **f**、**F**、**d**、**D**、**l**、**L** を使用できます。この接尾辞は無視されます。

文字列

FEEL では、二重引用符で区切った文字が文字列として解釈されます。
以下に例を示します。

```
"John Doe"
```

ブール値

FEEL は、3 値ブール論理を使用するため、ブール論理式には **true**、**false**、または **null** を使用できます。

日付

FEEL では日付リテラルがサポートされていませんが、組み込みの **date()** 関数を使用して日付の値を構築できます。時間文字列は、FEEL では XML Schema Part 2: Datatypes ドキュメントに定義されている形式に準拠します。形式は、**"YYYY-MM-DD"** で、**YYYY** は 4 桁の年数、**MM** は 2 桁の月数、**DD** は日数に置き換えます。
以下に例を示します。

```
date( "2017-06-23" )
```

日付オブジェクトには、真夜中を表す **"00:00:00"** と同一の時間があります。日付には、タイムゾーンがなく、ローカルであると見なされます。

時間

FEEL では時間リテラルがサポートされていませんが、組み込みの **time()** 関数を使用して時間の値を構築できます。時間文字列は、FEEL では XML Schema Part 2: Datatypes ドキュメントで定義されている形式に準拠します。形式は **"hh:mm:ss[.uuu][(+-)hh:mm]"** です。ここで、**hh** は時間 (**00** から **23**)、**mm** は分、**ss** は秒です。任意で、ミリ秒 (**uuu**) を定義でき、UTC 時間の正 (**+**) または負 (**-**) のオフセットを追加してタイムゾーンを定義できます。オフセットを使用する代わりに、**z** 文字を使用して UTC 時間を表すことができますが、これは **-00:00** のオフセットと同じです。オフセットが定義されていない場合には、時間はローカルとみなされます。
例 :

```
time( "04:25:12" )
time( "14:10:00+02:00" )
time( "22:35:40.345-05:00" )
time( "15:00:30z" )
```

オフセットまたはタイムゾーンを定義する時間値は、オフセットまたはタイムゾーンを定義しないローカル時間と比較できません。

日時

FEEL では日時リテラルがサポートされていませんが、組み込みの **date and time()** 関数を使用して値を構築できます。日時文字列は、FEEL では XML Schema Part 2: Datatypes ドキュメントに定義された形式に準拠します。形式は **"<date>T<time>"** です。**<date>** および **<time>** は規定の XML スキーマ形式に準拠し、**T** で結合されます。
例 :

```
date and time( "2017-10-22T23:59:00" )
date and time( "2017-06-13T14:10:00+02:00" )
date and time( "2017-02-05T22:35:40.345-05:00" )
date and time( "2017-06-13T15:00:30z" )
```

オフセットまたはタイムゾーンを定義する日時の値と、オフセットまたはタイムゾーンを定義しないローカルの日時の値を比較することはできません。

重要

DMN 仕様の実装が、XML スキーマでスペースをサポートしない場合は、キーワード **dateTime** を **date and time** の同義語として使用してください。

日時で指定する期間

FEEL では日と時間で指定する期間を表すリテラルがサポートされていませんが、組み込みの **duration()** 関数を使用して値を構築できます。FEEL では、XML Schema Part 2: Datatypes ドキュメントに定義されている形式に準拠しますが、日、時間、分、および秒にしか適用されません。月および年はサポートされません。
例 :

```
duration( "P1DT23H12M30S" )
duration( "P23D" )
duration( "PT12H" )
duration( "PT35M" )
```

重要

DMN 仕様の実装が、XML スキーマでスペースをサポートしない場合は、キーワード **dayTimeDuration** を **days and time duration** の同義語として使用してください。

年および月で指定する期間

FEEL では年と月で指定する期間リテラルがサポートされていませんが、組み込みの **duration()** 関数を使用して値を構築できます。FEEL では XML Schema Part 2: Datatypes ドキュメントに定義されている形式に準拠しますが、年と月にしか適用されません。日、時間、分、または秒はサポートされません。
例 :

```
duration( "P3Y5M" )
duration( "P2Y" )
duration( "P10M" )
duration( "P25M" )
```

重要

DMN 仕様の実装が、XML スキーマでスペースをサポートしない場合は、キーワード **yearMonthDuration** を **years and months duration** の同義語として使用してください。

関数

FEEL には、関数を作成するのに使用する **function** リテラル (または無名関数) があります。DMN 仕様で変数を **function** として宣言する明示的な方法が提供されていませんが、Red Hat Decision Manager では、関数をサポートするように DMN 型が拡張されています。
以下に例を示します。

```
function(a, b) a + b
```

この例では、FEEL 式は、パラメーター **a** および **b** を追加して結果を返す関数を作成します。

コンテキスト

FEEL には、コンテキストを作成するのに使用する **context** リテラルがあります。**context** は、FEEL ではキーと値のペアのリストとなり、Java などの言語におけるマッピングに似ています。DMN 仕様で変数を **context** として宣言する明示的な方法が提供されていませんが、Red Hat Decision Manager では、コンテキストをサポートするように組み込みの DMN 型が拡張されています。
以下に例を示します。

```
{ x : 5, y : 3 }
```

この式では、チャート内の等位を示す 2 つのエントリー (**x** および **y**) を持つコンテキストが作成されます。

DMN 1.2 では、コンテキスト作成に、キーの一覧を属性として含めてアイテム定義を作成し、そのアイテム定義型を含めて変数を宣言する方法も使用できます。

Red Hat Decision Manager の DMN API では、**DMNContext** の DMN **ItemDefinition** 構造様式として、次の 2 つがサポートされます。

- ユーザー定義の Java タイプ: DMN の **ItemDefinition** で各コンポーネントのプロパティーとゲッターを定義する有効な JavaBeans オブジェクト。必要に応じて、無効な Java 識別子になるコンポーネント名を示すゲッターに対して **@FEELProperty** アノテーションを使用することもできます。

- **java.util.Map** インターフェイス: DMN の **ItemDefinition** でコンポーネント名に対応するキーで、適切なエンティティーを定義する必要があります。

範囲 (または間隔)

FEEL には、範囲または間隔を作成するのに使用する **range** リテラルがあります。FEEL の **range** は、下方境界および上方境界を定義する値で、開区間または閉区間のいずれかにできます。DMN 仕様には (別の式で使用する以外に) 変数を **range** と宣言する明示的な方法はありませんが、Red Hat Decision Manager では、範囲をサポートするように DMN 型が拡張されています。
範囲の構文は以下の形式で定義されます。

```
range          := interval_start endpoint '..' endpoint interval_end
interval_start := open_start | closed_start
open_start     := '(' | ']'
closed_start   := '['
interval_end   := open_end | closed_end
open_end       := ')' | '['
closed_end     := ']'
endpoint       := expression
```

エンドポイントの式は比較可能な値を返す必要があり、下方エンドポイントは上方エンドポイントよりも低くなる必要があります。

たとえば、以下のリテラル式は、**1** から **10** まで (いずれも閉区間) の間隔を定義します。

```
[ 1 .. 10 ]
```

以下のリテラル式は 1 時間から 12 時間までの間隔を定義します。下方境界は含まれます (閉区間) が、上方境界は含まれません (開区間)。

```
[ duration("PT1H") .. duration("PT12H") )
```

デシジョンテーブルの範囲を使用して値の範囲をテストしたり、単純なリテラル式で範囲を使用したりできます。たとえば、以下のリテラル式は、変数 **x** が **0** から **100** の間にある場合は **true** を返します。

```
x in [ 1 .. 100 ]
```

リスト

FEEL には、アイテムの一覧を作成するのに使用する **list** リテラルがあります。FEEL の **list** は、値のコンマ区切りの一覧を角カッコで囲んで表現できます。DMN 仕様には (別の式で使用する以外に) 変数を **list** と宣言する明示的な方法はありませんが、Red Hat Decision Manager では、範囲をサポートするように DMN 型が拡張されています。
以下に例を示します。

```
[ 2, 3, 4, 5 ]
```

FEEL のリストはすべて同じ型の要素を含み、変更できません。リストの要素はインデックスでアクセスでき、最初の要素が **1** になります。負のインデックスは、リストの末尾から数えた要素を表します。たとえば、**-1** は最後の要素にアクセスできることを示します。

たとえば、以下の式は、リスト **x** の 2 番目の要素を返します。

```
x[2]
```

以下の式は、リスト **x** の、最後から 2 番目の要素を返します。

```
x[-2]
```

1.4. ボックス式の DMN デシジョンロジック

DMN のボックス式は、意思決定要件ダイアグラム (DRD) または意思決定要件グラフ (DRG) でデシジョンノードの基盤ロジックを定義するのに使用するテーブルです。ボックス式には他のボックス式が含まれる場合がありますが、トップレベルのボックス式は単一の DRD アーティファクトのデシジョンロジックに対応します。1 つまたは複数の DRG が含まれる DRD は、DMN デシジョンモデルのフローを表現し、反対にボックス式は個別ノードの実際のデシジョンロジックを定義します。DRD とボックス式は、完全で機能的な DMN デシジョンモデルを形成します。

以下は、DMN のボックス式の種類です。

- デシジョンテーブル
- リテラル式
- コンテキスト
- 関係
- 関数
- 呼び出し
- リスト

注記

Red Hat Decision Manager では、Decision Central にボックスリスト式が含まれていませんが、FEEL の **list** のデータ型が含まれているためボックスリテラル式で使用できます。Red Hat Decision Manager の **list** データ型およびその他の FEEL データ型の詳細については、「FEEL のデータ型」 を参照してください。

ボックス式で使用する Friendly Enough Expression Language (FEEL) 式はすべて、OMG の Decision Model and Notation specification に記載されている FEEL 構文の要件に準拠する必要があります。

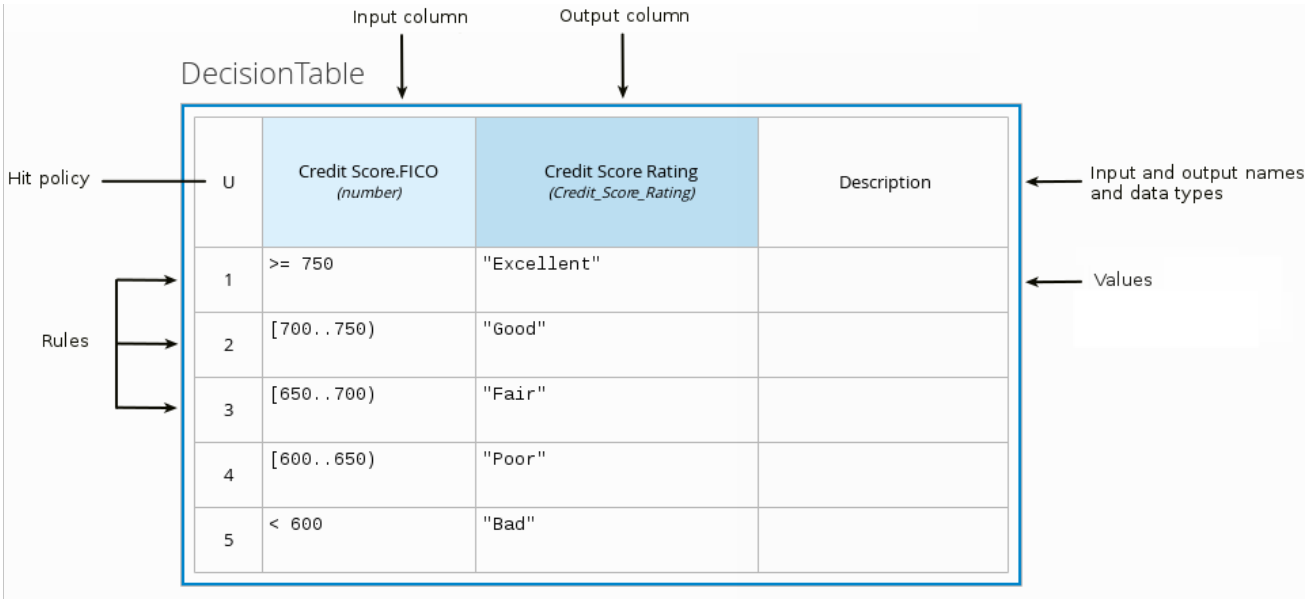
1.4.1. DMN デシジョンテーブル

DMN のデシジョンテーブルは、1 つ以上のビジネスルールをテーブル形式で視覚的に表します。デシジョンテーブルを使用して、デシジョンモデルの特定の地点でこれらのルールを適用するデシジョンノードのルールを定義します。テーブルの各行はルール 1 つで設定されており、その特定行に対する条

件 (入力) と結果 (出力) を定義する列が含まれます。各行の定義は、条件の値を使用して結果を取得できるほど正確です。入力と出力の値には、FEEL 式または定義済みのデータ型の値を指定できます。

たとえば、以下のデシジョンテーブルでは、ローン申請者のクレジットスコアの定義範囲に基づき、クレジットスコアを評価します。

図1.2 クレジットスコア評価のデシジョンテーブル



以下のデシジョンテーブルでは、申請者の借り入れ資格や Berueu Call Type に従い、申請者の融資戦略における次のステップを決定します。

図1.3 融資戦略のデシジョンテーブル

DecisionTable

U	Eligibility *(string)*	BureauCallType *(string)*	Strategy *(tStrategy)*	Description
1	"INELIGIBLE"	-	"DECLINE"	Disregard BureauCallType when ineligible.
2	"ELIGIBLE"	"FULL","MINI"	"BUREAU"	
3	"ELIGIBLE"	"NONE"	"THROUGH"	

以下のデシジョンテーブルでは、ローン事前審査のデシジョンモデルで終端デシジョンノードとして、申請者のローン適正を決定します。

図1.4 ローン事前審査のデシジョンテーブル

DecisionTable

F	Credit Score Rating *(Credit_Score_Rating)*	Back End Ratio *(Back_End_Ratio)*	Front End Ratio *(Front_End_Ratio)*	Loan Pre-Qualification *(Loan_Qualification)*		Description
				Qualification *(string)*	Reason *(string)*	
1	"Poor", "Bad"	-	-	"Not Qualified"	"Credit Score too low."	
2	-	"Insufficient"	"Sufficient"	"Not Qualified"	"Debt to income ratio is too high."	
3	-	"Sufficient"	"Insufficient"	"Not Qualified"	"Mortgage payment to income ratio is too high."	
4	-	"Insufficient"	"Insufficient"	"Not Qualified"	"Debt to income ratio is too high AND mortgage payment to income ratio is too high."	
5	"Fair", "Good", "Excellent"	"Sufficient"	"Sufficient"	"Qualified"	"The borrower has been successfully prequalified for the requested loan."	

デシジョンテーブルは、ルールとデシジョンロジックのモデル化の方法として一般的で、多くの方法論 (DMN など) や実装フレームワーク (Drools など) で使用されます。

重要

Red Hat Decision Manager は DMN デシジョンテーブルおよび Drools ネイティブのデシジョンテーブルの両方をサポートしますが、アセットのタイプが異なると構文の要件も異なり、それぞれを置き換えて使用できません。Red Hat Decision Manager の Drools ネイティブのデシジョンテーブルに関する情報は、**アップロードしたデシジョンテーブルを使用したデシジョンサービスの設計** を参照してください。

1.4.1.1. DMN デシジョンテーブルのヒットポリシー

ヒットポリシーは、デシジョンテーブルにある複数のルールが指定の入力値と一致する場合に、どのように結果に到達するかを決定します。たとえば、デシジョンテーブルの中の 1 つのルールでは、軍人に価格の割引を適用し、別のルールでは学生に割引を適用する場合に、学生であり軍人である顧客には、デシジョンテーブルのヒットポリシーに割引を 1 つだけ適用するのか (**Unique**、**First**) または両方の割引を適用するのか (**Collect Sum**) 指定しておく必要があります。ヒットポリシーの 1 文字 (**U**、**F**、**C+**) をデシジョンテーブルの左上隅に指定します。

以下は、サポートされている DMN デシジョンテーブルのヒットポリシーです。

- **Unique (U):** 一致するルールを 1 つだけ許可します。重複はエラーとなります。
- **Any (A):** 複数のルールが一致するのを許可しますが、出力は同じである必要があります。一致している複数のルールで出力が同じでないと、エラーが発生します。
- **Priority (P):** 複数のルールが一致し、結果が異なるのを許可します。出力値リストで最初に出力されるものが選択されます。
- **First (F):** ルールの順番に従い、最初に一致するのを使用します。
- **Collect (C+、C>、C<、C#):** 集約関数に基づいて、複数のルールから出力を集めます。
  - **Collect (C):** 任意のリストで値を集めます。
  - **Collect Sum (C+):** 集計したすべての値の合計を出力します。値は数値でなければなりません。

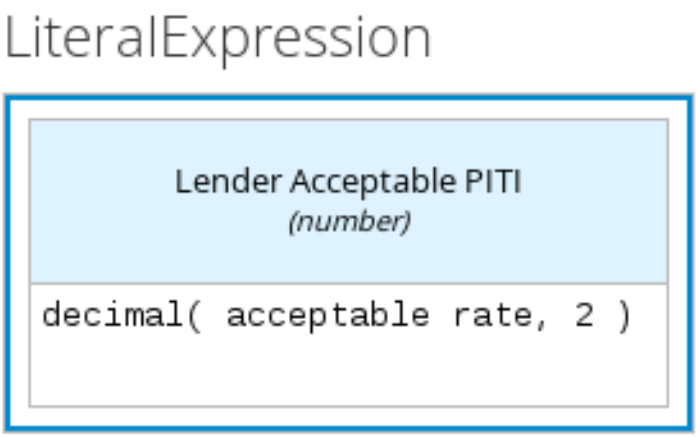
- **Collect Min (C<):** 一致する中で最小の値を出力します。結果の値は、数値、日付、またはテキスト (辞書的順序) など、比較可能な値である必要があります。
- **Collect Max (C>):** 一致する中で最高の値を出力します。結果の値は、数値、日付、またはテキスト (辞書的順序) など、比較可能な値である必要があります。
- **Collect Count (C#):** 一致するルールの数を出力します。

1.4.2. ボックスリテラル式

DMN のボックスリテラル式は、テーブルのセル内のテキストとして使用するリテラル FEEL 式で、通常ラベル付きの列およびデータタイプが割り当てられています。ボックスリテラル式を使用して、デシジョンの特定のノードに対して FEEL で直接、単純または複雑なノードロジックまたはデシジョンデータを定義できます。リテラル FEEL 式は OMG の Decision Model and Notation specification の FEEL 構文要件に準拠する必要があります。

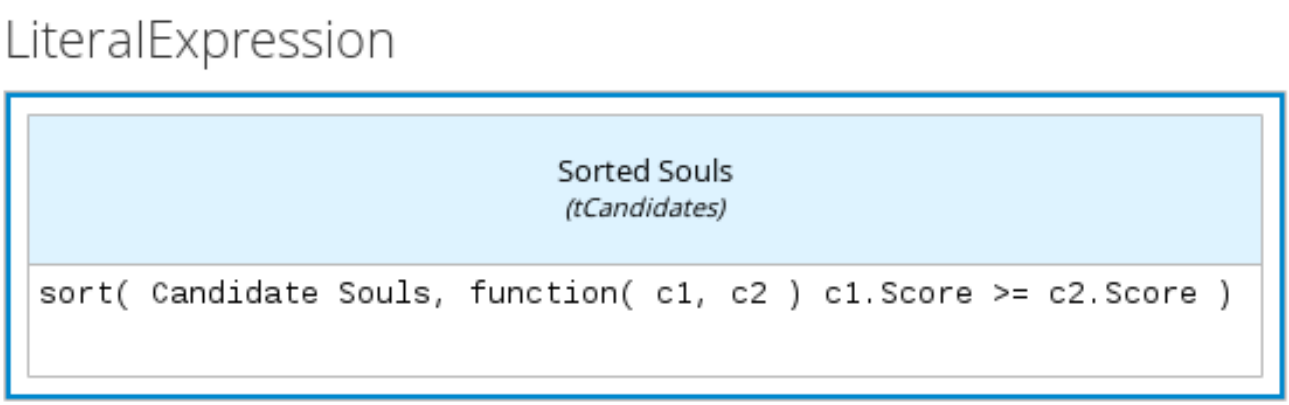
たとえば、以下のボックスリテラル式では、融資のデシジョンにおいて最低限許容できる PITI 計算 (元金 (Principal)、利子 (Interest)、税金 (Tax)、保険 (Insurance)) を定義します。ここでの **acceptable rate** は、DMN モデルで定義した変数です。

図1.5 PITI の最小値のボックスリテラル式



以下のボックスリテラル式は、年齢、場所、趣味などの基準のスコアをもとに、オンラインの出会い系アプリでデート相手の候補 (ソウルメイト) 一覧をソートします。

図1.6 オンラインでデート相手の候補者をマッチングするボックスリテラル式



1.4.3. ボックスコンテキスト式

DMN のボックスコンテキスト式は、結果の値が含まれる、値と変数名のセットです。名前と値のペア

はそれぞれ、コンテキストエントリーとなっています。コンテキスト式を使用して、デシジョンロジックでデータの定義を表現し、DMN デシジョンモデル内で任意のデシジョン要素の値を設定します。ボックスコンテキスト式の値は、データ型の値または FEEL 式を指定でき、デシジョンテーブル、リテラル式、または別のコンテキスト式など、どの型でもサブ式をネスト化させることができます。

たとえば、以下のボックスコンテキスト式では、定義したデータ型 (**tPassengerTable**, **tFlightNumberList**) をもとに、飛行機の再予約を行うデシジョンモデルで遅延客をソートする要素を定義します。

図1.7 航空機利用客のウェイティングリストのボックスコンテキスト式

Context

#	Prioritized Waiting List *(tPassengerTable)*	
1	Cancelled Flights *(tFlightNumberList)*	`Flight List[ Status = "cancelled" ].Flight Number`
2	Waiting List *(tPassengerTable)*	`Passenger List[ list contains( Cancelled Flights, Flight Number ) ]`
	<result>	`sort( Waiting List, passenger priority )`

以下のボックスコンテキスト式では、サブコンテキスト式が含まれるフロントエンドの割合計算として表現されている PITI (元金 (Principal)、利子 (Interest)、税金 (Tax)、保険 (Insurance)) をもとに、ローンの申請者が最小限必要とされるローンの支払いをしているかを決定する要素を定義します。

図1.8 フロントエンドクライアント PITI 割合のボックスコンテキスト式

Context

#	Front End Ratio *(Front_End_Ratio)*			
1	Client PITI *(number)*	#	PITI	
		1	pmt *(<Undefined>)*	`(Requested Product.Amount*((Requested Product.Rate/100)/12))/ (1-(1/(1+(Requested Product.Rate/100)/12)**-Requested Product.Term))`
		2	tax *(<Undefined>)*	`Applicant Data.Monthly.Tax`
		3	insurance *(<Undefined>)*	`Applicant Data.Monthly.Insurance`
		4	income *(<Undefined>)*	`Applicant Data.Monthly.Income`
	<result>	`if Client PITI <= Lender Acceptable PITI() then "Sufficient" else "Insufficient"`		

1.4.4. ボックスリレーション式

DMN のボックスリレーション式は、指定のエンティティーに関する情報 (行として記載) が含まれる従来のデータテーブルです。ボックスリレーションテーブルを使用して、特定のノードでのデシジョンで関連するエンティティーのデシジョンデータを定義します。ボックスリレーション式は、変数名と値を設定する点ではコンテキスト式に似ていますが、リレーション式には結果の値が含まれておらず、定義した変数を 1 つをもとに全変数値を列ごとにリストします。

たとえば、以下のボックスリレーション式は、従業員の勤務表デシジョンで従業員に関する情報を提供します。

図1.9 従業員の情報を含むボックスリレーション式

Relation

#	Name *(string)*	Dept *(string)*	Salary *(number)*
1	"John"	"Sales"	100000
2	"Mary"	"Finances"	120000

1.4.5. ボックス関数式

DMN のボックス関数式は、リテラル FEEL 式、外部の JAVA または PMML 関数のネスト化されたコンテキスト式、あらゆる型のネスト化されたボックス式を含む、パラメーターを使用するボックス式です。デフォルトでは、全ビジネスナレッジモデルは、ボックス関数式として定義されます。ボックス関数式を使用して、デシジョンロジックで関数を呼び出し、全ビジネスナレッジモデルを定義します。

たとえば、以下のボックス関数式では、フライトの予約変更デシジョンモデルで、航空機の定員を決定します。

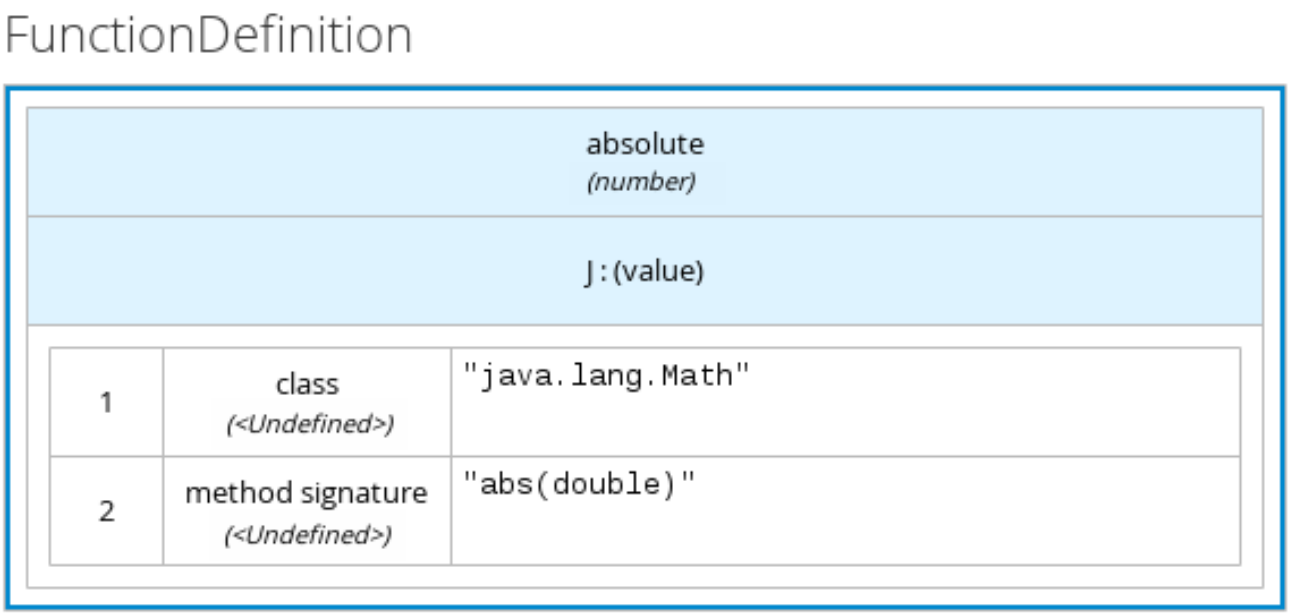
図1.10 フライトの定員に使用するボックス関数式

FunctionDefinition

has capacity *(boolean)*
F : (flight, rebooked list)
`flight.Capacity > count( rebooked list[ Flight Number = flight.Flight Number ] )`

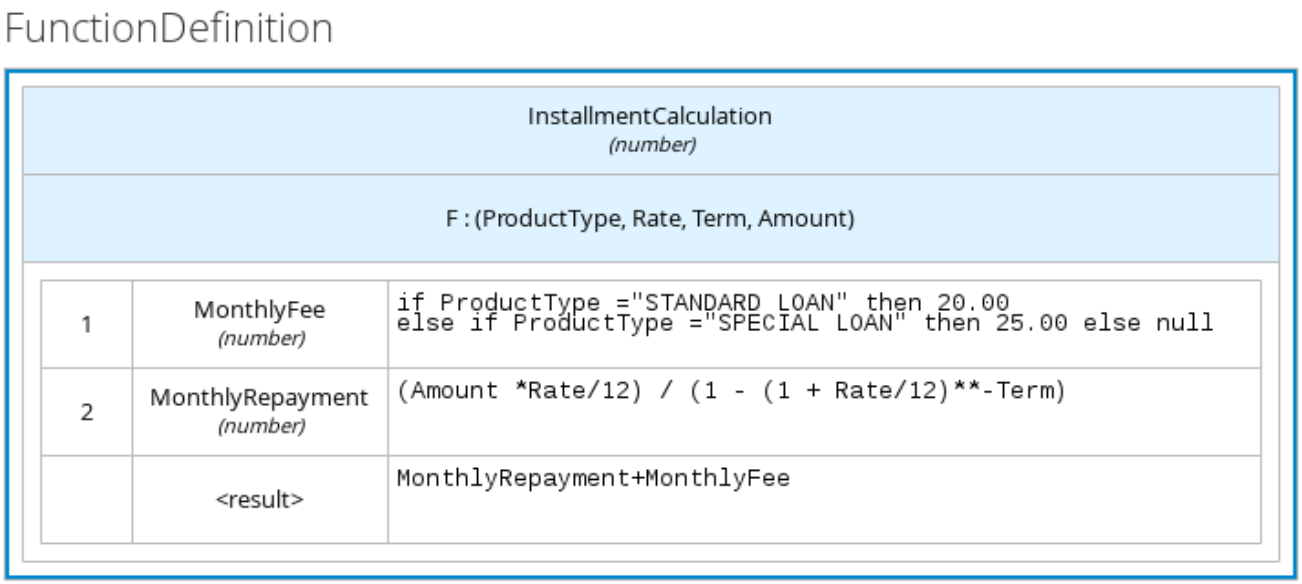
以下のボックス関数式には、デシジョンモデルの計算で絶対値を判断するコンテキスト式として使用する基本的な Java 関数が含まれています。

図1.11 絶対値のボックス関数式



以下のボックス関数式では、ネスト化されたコンテキスト式として定義された関数値を使用し、融資のデシジョンのビジネスナレッジモデルとして、住宅ローンの月額を決定します。

図1.12 ビジネスナレッジモデルのローン計算で使用するボックス関数式



1.4.6. ボックス呼び出し式

DMN のボックス呼び出し式は、ビジネスナレッジモデルを呼び出すボックス式です。ボックス呼び出し式には、呼び出すビジネスナレッジモデルの名前と、パラメーターバインディングのリストが含まれています。各バインディングは、1行に 2 つのボックス式を入れることで表現します。左のボックスにはパラメーターの名前、右のボックスには呼び出したビジネスナレッジモデルを評価するパラメーターに割り当てられる値のバインディング式が含まれます。ボックス式を使用して、デシジョンモデルで定義されているビジネスナレッジモデルを特定のデシジョンノードで呼び出します。

たとえば、以下のボックス呼び出し式では、フライト予約変更のデシジョンモデルで終端デシジョンノードとして **reassign next passenger** ビジネスナレッジモデルを呼び出します。

図1.13 フライトの乗客を再割り当てするボックス呼び出し式

Invocation

#	Rebooked Passengers *(tPassengerTable)*	
	reassign next passenger	
1	Waiting List *(<Undefined>)*	`Prioritized Waiting List`
2	Reassigned Passengers List *(<Undefined>)*	`[]`
3	Flights *(<Undefined>)*	`Flight List`

以下のボックス呼び出し式では、**InstallmentCalculation** ビジネスナレッジモデルを呼び出し、ローンを負担できるかどうか決定する前に、ローンの月額を計算します。

図1.14 必要な月額を判断するボックス呼び出し式

Invocation

#	RequiredMonthlyInstallment *(number)*	
	InstallmentCalculation	
1	ProductType *(<Undefined>)*	`RequestedProduct.ProductType`
2	Rate *(<Undefined>)*	`RequestedProduct.Rate`
3	Term *(<Undefined>)*	`RequestedProduct.Term`
4	Amount *(<Undefined>)*	`RequestedProduct.Amount`

1.5. DMN モデルの例

以下は、入力データ、状況、企業のガイドラインをもとに、デシジョンモデルをどのように使用して決断に至るかを判断する実際の DMN モデル例です。以下のシナリオでは、サンディエゴからニューヨークへのフライトがキャンセルされ、欠航となってしまったフライトの航空会社は、このフライトの乗客に対して、別のフライトを手配する必要があります。

まずは、乗客を目的地に運ぶ最適な方法を決めるのに必要な情報を集めます。

入力データ

- フライトリスト
- 乗客リスト

決定

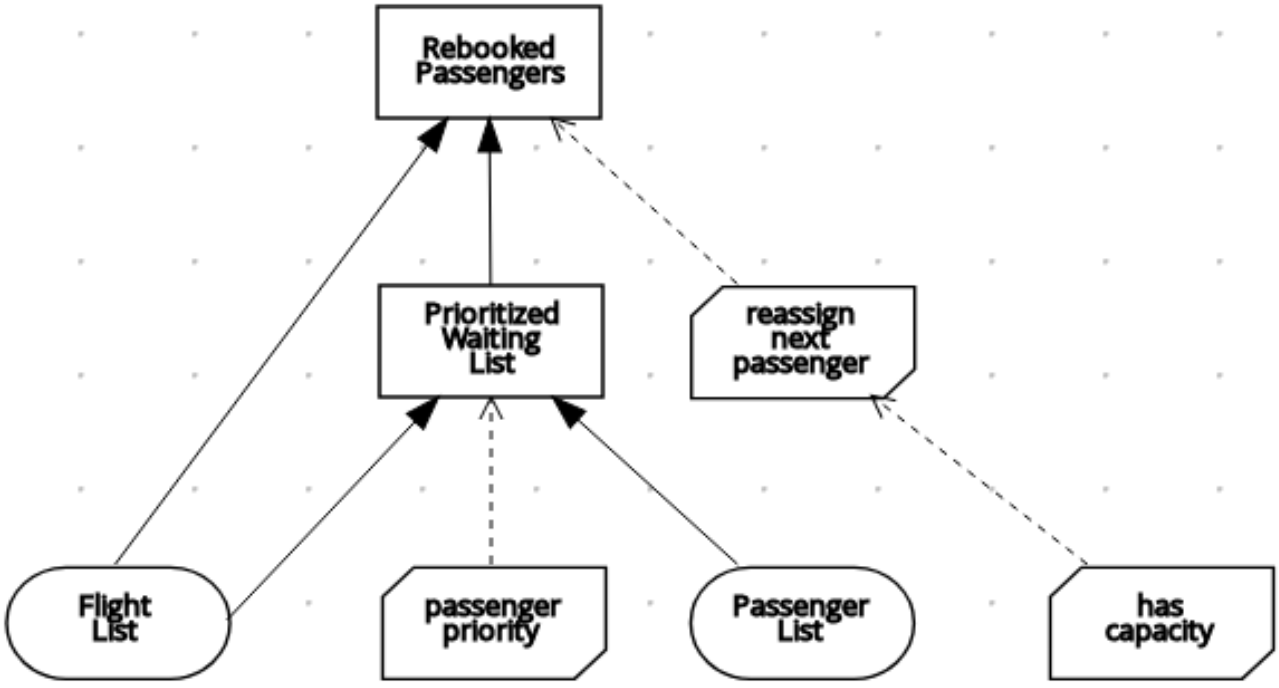
- 新しいフライトで席を確保する乗客の優先順位をつける
- 乗客に提示するフライトを決定する

ビジネスナレッジモデル

- 乗客の優先順位を決定する企業のプロセス
- 席に余裕があるフライト
- フライトをキャンセルされた乗客を再割り当てするのに最適な方法を決定する会社のルール

次に、航空会社は、DMN 仕様を使用して、以下の意思決定要件ダイアグラム (DRD) でそのデシジョンプロセスをモデル化し、予約変更の最適解を決める以下のダイアグラムを作成します。

図1.15 フライト予約変更の DRD



DRD では、フローチャートのように、プロセスの各要素に異なる形状を使用します。楕円形には必要な入力データが 2 つ、長方形にはモデルでのデシジョンポイントを含み、端が欠けた長方形 (ビジネスナレッジモデル) には、繰り返し呼び出せる再利用可能なロジックが含まれます。

DRD は、FEEL 式またはデータ型の値を使用して変数定義を提供するボックス式から各要素のロジックを引き出します。

ウィティングリストの優先順位を確立する以下のデシジョンなど、ボックス式には基本的なものもあります。

図1.16 ウェイティングリストの優先順位に関するボックスコンテキスト式のサンプル

Context

#	Prioritized Waiting List *(tPassengerTable)*	
1	Cancelled Flights *(tFlightNumberList)*	`Flight List[ Status = "cancelled" ].Flight Number`
2	Waiting List *(tPassengerTable)*	`Passenger List[ list contains( Cancelled Flights, Flight Number ) ]`
	<result>	`sort( Waiting List, passenger priority )`

ボックス式には、次の遅延客を再割り当てするための以下のビジネスナレッジモデルなど、詳細にわたる情報や計算が含まれ、さらに複雑なものもあります。

図1.17 乗客再割り当てのボックス関数式

FunctionDefinition

reassign next passenger *(tPassengerTable)*

F : (Waiting List, Reassigned Passengers List, Flights)

1	Next Passenger *(tPassenger)*	`Waiting List[1]`
2	Original Flight *(tFlight)*	`Flights[ Flight Number = Next Passenger.Flight Number ][1]`
3	Best Alternate Flight *(tFlight)*	`Flights[ From = Original Flight.From and To = Original Flight.To and Departure > Original Flight.Departure and Status = "scheduled" and`
4	Reassigned Passenger *(tPassenger)*	1 Name *(string)* `Next Passenger.Name` 2 Status *(string)* `Next Passenger.Status` 3 Miles *(number)* `Next Passenger.Miles` 4 Flight Number *(string)* `Best Alternate Flight.Flight Number` <result> --
5	Remaining Waiting List *(tPassengerTable)*	`remove( Waiting List, 1 )`
6	Updated Reassigned Passengers List *(tPassengerTable)*	`append( Reassigned Passengers List, Reassigned Passenger )`
	<result>	`if count( Remaining Waiting List ) > 0 then reassign next passenger( Remaining Waiting List, Updated Reassigned Passengers List`

以下は、このデシジョンモデルの DMN ソースファイルです。

```
<definitions xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd"
xmlns:kie="https://www.drools.org/kie-dmn" xmlns:feel="http://www.omg.org/spec/FEEL/20140401"
```

```
id="_0019_flight_rebooking" name="0019-flight-rebooking" namespace="https://www.drools.org/kie-dmn">
  <itemDefinition id="_tFlight" name="tFlight">
    <itemComponent id="_tFlight_Flight" name="Flight Number">
      <typeRef>feel:string</typeRef>
    </itemComponent>
    <itemComponent id="_tFlight_From" name="From">
      <typeRef>feel:string</typeRef>
    </itemComponent>
    <itemComponent id="_tFlight_To" name="To">
      <typeRef>feel:string</typeRef>
    </itemComponent>
    <itemComponent id="_tFlight_Dep" name="Departure">
      <typeRef>feel:dateTime</typeRef>
    </itemComponent>
    <itemComponent id="_tFlight_Arr" name="Arrival">
      <typeRef>feel:dateTime</typeRef>
    </itemComponent>
    <itemComponent id="_tFlight_Capacity" name="Capacity">
      <typeRef>feel:number</typeRef>
    </itemComponent>
    <itemComponent id="_tFlight_Status" name="Status">
      <typeRef>feel:string</typeRef>
    </itemComponent>
  </itemDefinition>
  <itemDefinition id="_tFlightTable" isCollection="true" name="tFlightTable">
    <typeRef>kie:tFlight</typeRef>
  </itemDefinition>
  <itemDefinition id="_tPassenger" name="tPassenger">
    <itemComponent id="_tPassenger_Name" name="Name">
      <typeRef>feel:string</typeRef>
    </itemComponent>
    <itemComponent id="_tPassenger_Status" name="Status">
      <typeRef>feel:string</typeRef>
    </itemComponent>
    <itemComponent id="_tPassenger_Miles" name="Miles">
      <typeRef>feel:number</typeRef>
    </itemComponent>
    <itemComponent id="_tPassenger_Flight" name="Flight Number">
      <typeRef>feel:string</typeRef>
    </itemComponent>
  </itemDefinition>
  <itemDefinition id="_tPassengerTable" isCollection="true" name="tPassengerTable">
    <typeRef>kie:tPassenger</typeRef>
  </itemDefinition>
  <itemDefinition id="_tFlightNumberList" isCollection="true" name="tFlightNumberList">
    <typeRef>feel:string</typeRef>
  </itemDefinition>
  <inputData id="i_Flight_List" name="Flight List">
    <variable name="Flight List" typeRef="kie:tFlightTable"/>
  </inputData>
  <inputData id="i_Passenger_List" name="Passenger List">
    <variable name="Passenger List" typeRef="kie:tPassengerTable"/>
  </inputData>
  <decision name="Prioritized Waiting List" id="d_PrioritizedWaitingList">
    <variable name="Prioritized Waiting List" typeRef="kie:tPassengerTable"/>
```

```
    <informationRequirement>
      <requiredInput href="#i_Passenger_List"/>
    </informationRequirement>
    <informationRequirement>
      <requiredInput href="#i_Flight_List"/>
    </informationRequirement>
    <knowledgeRequirement>
      <requiredKnowledge href="#b_PassengerPriority"/>
    </knowledgeRequirement>
    <context>
      <contextEntry>
        <variable name="Cancelled Flights" typeRef="kie:tFlightNumberList"/>
        <literalExpression>
          <text>Flight List[ Status = "cancelled" ].Flight Number</text>
        </literalExpression>
      </contextEntry>
      <contextEntry>
        <variable name="Waiting List" typeRef="kie:tPassengerTable"/>
        <literalExpression>
          <text>Passenger List[ list contains( Cancelled Flights, Flight Number ) ]</text>
        </literalExpression>
      </contextEntry>
      <contextEntry>
        <literalExpression>
          <text>sort( Waiting List, passenger priority )</text>
        </literalExpression>
      </contextEntry>
    </context>
  </decision>
  <decision name="Rebooked Passengers" id="d_RebookedPassengers">
    <variable name="Rebooked Passengers" typeRef="kie:tPassengerTable"/>
    <informationRequirement>
      <requiredDecision href="#d_PrioritizedWaitingList"/>
    </informationRequirement>
    <informationRequirement>
      <requiredInput href="#i_Flight_List"/>
    </informationRequirement>
    <knowledgeRequirement>
      <requiredKnowledge href="#b_ReassignNextPassenger"/>
    </knowledgeRequirement>
    <invocation>
      <literalExpression>
        <text>reassign next passenger</text>
      </literalExpression>
      <binding>
        <parameter name="Waiting List"/>
        <literalExpression>
          <text>Prioritized Waiting List</text>
        </literalExpression>
      </binding>
      <binding>
        <parameter name="Reassigned Passengers List"/>
        <literalExpression>
          <text>[]</text>
        </literalExpression>
      </binding>
```

```
    <binding>
      <parameter name="Flights"/>
      <literalExpression>
        <text>Flight List</text>
      </literalExpression>
    </binding>
  </invocation>
</decision>
<businessKnowledgeModel id="b_PassengerPriority" name="passenger priority">
  <encapsulatedLogic>
    <formalParameter name="Passenger1" typeRef="kie:tPassenger"/>
    <formalParameter name="Passenger2" typeRef="kie:tPassenger"/>
    <decisionTable hitPolicy="UNIQUE">
      <input id="b_Passenger_Priority_dt_i_P1_Status" label="Passenger1.Status">
        <inputExpression typeRef="feel:string">
          <text>Passenger1.Status</text>
        </inputExpression>
        <inputValues>
          <text>"gold", "silver", "bronze"</text>
        </inputValues>
      </input>
      <input id="b_Passenger_Priority_dt_i_P2_Status" label="Passenger2.Status">
        <inputExpression typeRef="feel:string">
          <text>Passenger2.Status</text>
        </inputExpression>
        <inputValues>
          <text>"gold", "silver", "bronze"</text>
        </inputValues>
      </input>
      <input id="b_Passenger_Priority_dt_i_P1_Miles" label="Passenger1.Miles">
        <inputExpression typeRef="feel:string">
          <text>Passenger1.Miles</text>
        </inputExpression>
      </input>
      <output id="b_Status_Priority_dt_o" label="Passenger1 has priority">
        <outputValues>
          <text>true, false</text>
        </outputValues>
        <defaultOutputEntry>
          <text>false</text>
        </defaultOutputEntry>
      </output>
      <rule id="b_Passenger_Priority_dt_r1">
        <inputEntry id="b_Passenger_Priority_dt_r1_i1">
          <text>"gold"</text>
        </inputEntry>
        <inputEntry id="b_Passenger_Priority_dt_r1_i2">
          <text>"gold"</text>
        </inputEntry>
        <inputEntry id="b_Passenger_Priority_dt_r1_i3">
          <text>>= Passenger2.Miles</text>
        </inputEntry>
        <outputEntry id="b_Passenger_Priority_dt_r1_o1">
          <text>true</text>
        </outputEntry>
      </rule>
```

```
<rule id="b_Passenger_Priority_dt_r2">
  <inputEntry id="b_Passenger_Priority_dt_r2_i1">
    <text>"gold"</text>
  </inputEntry>
  <inputEntry id="b_Passenger_Priority_dt_r2_i2">
    <text>"silver","bronze"</text>
  </inputEntry>
  <inputEntry id="b_Passenger_Priority_dt_r2_i3">
    <text>-</text>
  </inputEntry>
  <outputEntry id="b_Passenger_Priority_dt_r2_o1">
    <text>true</text>
  </outputEntry>
</rule>
<rule id="b_Passenger_Priority_dt_r3">
  <inputEntry id="b_Passenger_Priority_dt_r3_i1">
    <text>"silver"</text>
  </inputEntry>
  <inputEntry id="b_Passenger_Priority_dt_r3_i2">
    <text>"silver"</text>
  </inputEntry>
  <inputEntry id="b_Passenger_Priority_dt_r3_i3">
    <text>>= Passenger2.Miles</text>
  </inputEntry>
  <outputEntry id="b_Passenger_Priority_dt_r3_o1">
    <text>true</text>
  </outputEntry>
</rule>
<rule id="b_Passenger_Priority_dt_r4">
  <inputEntry id="b_Passenger_Priority_dt_r4_i1">
    <text>"silver"</text>
  </inputEntry>
  <inputEntry id="b_Passenger_Priority_dt_r4_i2">
    <text>"bronze"</text>
  </inputEntry>
  <inputEntry id="b_Passenger_Priority_dt_r4_i3">
    <text>-</text>
  </inputEntry>
  <outputEntry id="b_Passenger_Priority_dt_r4_o1">
    <text>true</text>
  </outputEntry>
</rule>
<rule id="b_Passenger_Priority_dt_r5">
  <inputEntry id="b_Passenger_Priority_dt_r5_i1">
    <text>"bronze"</text>
  </inputEntry>
  <inputEntry id="b_Passenger_Priority_dt_r5_i2">
    <text>"bronze"</text>
  </inputEntry>
  <inputEntry id="b_Passenger_Priority_dt_r5_i3">
    <text>>= Passenger2.Miles</text>
  </inputEntry>
  <outputEntry id="b_Passenger_Priority_dt_r5_o1">
    <text>true</text>
  </outputEntry>
</rule>
```

```
      </decisionTable>
    </encapsulatedLogic>
    <variable name="passenger priority" typeRef="feel:boolean"/>
  </businessKnowledgeModel>
  <businessKnowledgeModel id="b_ReassignNextPassenger" name="reassign next passenger">
    <encapsulatedLogic>
      <formalParameter name="Waiting List" typeRef="kie:tPassengerTable"/>
      <formalParameter name="Reassigned Passengers List" typeRef="kie:tPassengerTable"/>
      <formalParameter name="Flights" typeRef="kie:tFlightTable"/>
      <context>
        <contextEntry>
          <variable name="Next Passenger" typeRef="kie:tPassenger"/>
          <literalExpression>
            <text>Waiting List[1]</text>
          </literalExpression>
        </contextEntry>
        <contextEntry>
          <variable name="Original Flight" typeRef="kie:tFlight"/>
          <literalExpression>
            <text>Flights[ Flight Number = Next Passenger.Flight Number ][1]</text>
          </literalExpression>
        </contextEntry>
        <contextEntry>
          <variable name="Best Alternate Flight" typeRef="kie:tFlight"/>
          <literalExpression>
            <text>Flights[ From = Original Flight.From and To = Original Flight.To and Departure >
Original Flight.Departure and Status = "scheduled" and has capacity( item, Reassigned Passengers
List ) ][1]</text>
          </literalExpression>
        </contextEntry>
        <contextEntry>
          <variable name="Reassigned Passenger" typeRef="kie:tPassenger"/>
          <context>
            <contextEntry>
              <variable name="Name" typeRef="feel:string"/>
              <literalExpression>
                <text>Next Passenger.Name</text>
              </literalExpression>
            </contextEntry>
            <contextEntry>
              <variable name="Status" typeRef="feel:string"/>
              <literalExpression>
                <text>Next Passenger.Status</text>
              </literalExpression>
            </contextEntry>
            <contextEntry>
              <variable name="Miles" typeRef="feel:number"/>
              <literalExpression>
                <text>Next Passenger.Miles</text>
              </literalExpression>
            </contextEntry>
            <contextEntry>
              <variable name="Flight Number" typeRef="feel:string"/>
              <literalExpression>
                <text>Best Alternate Flight.Flight Number</text>
              </literalExpression>
```

```
          </contextEntry>
        </context>
      </contextEntry>
      <contextEntry>
        <variable name="Remaining Waiting List" typeRef="kie:tPassengerTable"/>
        <literalExpression>
          <text>remove( Waiting List, 1 )</text>
        </literalExpression>
      </contextEntry>
      <contextEntry>
        <variable name="Updated Reassigned Passengers List" typeRef="kie:tPassengerTable"/>
        <literalExpression>
          <text>append( Reassigned Passengers List, Reassigned Passenger )</text>
        </literalExpression>
      </contextEntry>
      <contextEntry>
        <literalExpression>
          <text>if count( Remaining Waiting List ) > 0 then reassign next passenger( Remaining Waiting
List, Updated Reassigned Passengers List, Flights ) else Updated Reassigned Passengers
List</text>
        </literalExpression>
      </contextEntry>
    </context>
  </encapsulatedLogic>
  <variable name="reassign next passenger" typeRef="kie:tPassengerTable"/>
  <knowledgeRequirement>
    <requiredKnowledge href="#b_HasCapacity"/>
  </knowledgeRequirement>
 </businessKnowledgeModel>
 <businessKnowledgeModel id="b_HasCapacity" name="has capacity">
  <encapsulatedLogic>
    <formalParameter name="flight" typeRef="kie:tFlight"/>
    <formalParameter name="rebooked list" typeRef="kie:tPassengerTable"/>
    <literalExpression>
      <text>flight.Capacity > count( rebooked list[ Flight Number = flight.Flight Number ] )</text>
    </literalExpression>
  </encapsulatedLogic>
  <variable name="has capacity" typeRef="feel:boolean"/>
 </businessKnowledgeModel>
</definitions>
```

第2章 RED HAT DECISION MANAGER における DMN サポート

Red Hat Decision Manager は、適合レベル 3 で DMN 1.2 モデルの設計およびランタイムをサポートし、適合レベル 3 で DMN 1.1 モデルはランタイムのみサポートします。DMN モデルは、お使いの Red Hat Decision Manager デシジョンサービスと複数の方法で統合できます。

- DMN デザイナーを使用して Decision Central で直接 DMN モデルを設計します。
- Decision Central で プロジェクトに DMN ファイルをインポートします (**Menu → Design → Projects → Import Asset**)。Decision Central にインポートした DMN 1.1 はすべて、DMN デザイナーで開かれ、保存時に DMN 1.2 モデルに変換されます。
- Decision Central を使用せずにプロジェクトのナレッジ JAR (KJAR) ファイルの一部として DMN ファイルをパッケージ化します。

全 DMN 適合レベル 3 の要件に加え、Red Hat Decision Manager には FEEL および DMN モデルコンポーネントに機能拡張および修正が含まれており、Red Hat Decision Manager での DMN デシジョンサービスの実装体験を最適化します。DMN モデルは、プラットフォームの観点からすると、Red Hat Decision Manager プロジェクトに追加したり、DMN デシジョンサービスを起動するために Decision Server をデプロイしたりできるので、DRL ファイルやアップロードしたデシジョンテーブルなど、Red Hat Decision Manager の他のビジネスアセットとよく似ています。

Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイメントの方法を使用して外部 DMN ファイルを追加する方法は、**Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイ** を参照してください。

2.1. RED HAT DECISION MANAGER における設定可能な DMN プロパティー

Red Hat Decision Manager は、クライアントアプリケーションの Decision Server で DMN モデルを実行する際に設定できる以下の DMN プロパティーを提供します。

org.kie.dmn.strictConformance

このプロパティーを有効にすると、一部の helper 関数や、DMN 1.1 にバックポートされた DMN 1.2 の機能強化など、DMN 規定以外に提供された拡張機能やプロファイルをデフォルトで無効にします。このプロパティーを使用して、DMN Technology Compatibility Kit (TCK) を実行するなど、純粋な DMN 機能だけをサポートするデシジョンエンジンを設定できます。
デフォルト値は **false** です。

```
-Dorg.kie.dmn.strictConformance=true
```

org.kie.dmn.runtime.typecheck

このプロパティーを有効にすると、DRD 要素の入力または出力として、DMN モデルに宣言した型に従う実際の値を検証できるようになります。このプロパティーを使用して、DMN モデルに提供されたデータ、または DMN モデルが生成したデータが、モデルに指定したものに準拠するかどうかを検証できます。
デフォルト値は **false** です。

```
-Dorg.kie.dmn.runtime.typecheck=true
```

org.kie.dmn.decisionservice.coercesingleton

このプロパティーは、デフォルトで、1 つの出力デシジョンを定義するデシジョンサービスの結果

を、1つの出力デシジョン値にします。このプロパティーを無効にすると、出力されたデシジョンを定義するデシジョンサービスの結果を、その意思決定のエントリーを1つ持つ **context** にします。このプロパティーを使用して、プロジェクト要件に従ってデシジョンサービスを調整できます。
デフォルト値: **true**

```
-Dorg.kie.dmn.decisionservice.coercesingleton=false
```

org.kie.dmn.profiles.$PROFILE_NAME

このプロパティーは、Java の完全修飾名で設定された場合に、起動時にデシジョンエンジンに DMN プロファイルを読み込みます。このプロパティーを使用して、DMN 規定とは異なるサポート機能、またはそれ以外のサポート機能を使用する事前定義した DMN プロファイルを実装できます。Signavio DMN モデラーを使用して DMN モデルを作成する場合は、このプロパティーを使用して Signavio DMN プロファイルからお使いの DMN デシジョンサービスに機能を実装します。

```
-Dorg.kie.dmn.profiles.signavio=org.kie.dmn.signavio.KieDMNSignavioProfile
```

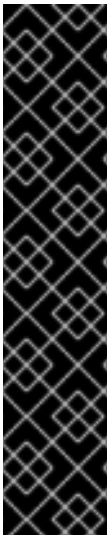
org.kie.dmn.compiler.execmodel

このプロパティーが有効な場合には、ランタイムに実行可能なルールモデルに DMN デシジョンテーブルロジックをコンパイルできます。このプロパティーを使用して、DMN デシジョンテーブルのロジックをより効率的に評価できます。このプロパティーは、実行可能なモデルのコンパイルがプロジェクトのコンパイル時に実行されなかった場合に有用です。このプロパティーを有効にすると、デシジョンエンジンにより最初の評価時のコンパイル時間が増加してしまいますが、その後のコンパイルがより効率的になります。
デフォルト値は **false** です。

```
-Dorg.kie.dmn.compiler.execmodel=true
```

第3章 DECISION CENTRAL での DMN モデルの作成および編集

Decision Central の DMN デザイナーを使用すると、DMN 意思決定要件ダイアグラム (DRD) を設計し、完全で機能的な DMN 意思決定モデルの意思決定論理を定義できます。Red Hat Decision Manager は、適合レベル 3 の DMN 1.2 モデルに対する設計とランタイムの両方のサポートを提供し、FEEL と DMN モデルコンポーネントの機能拡張と修正が含まれており、Red Hat Decision Manager での DMN 設計サービスの実装が最適化されます。Red Hat Decision Manager では、適合レベル 3 の DMN 1.1 に対してランタイムのみサポートしますが、Decision Central にインポートした DMN 1.1 はすべて、DMN デザイナーで開かれ、保存時に DMN 1.2 モデルに変換されます。



重要

Red Hat Decision Manager 7.2 の DMN デザイナーはテクノロジープレビュー機能で、デフォルトでは Decision Central で無効になっています。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) ではサポートされておらず、機能的に完全ではない可能性があるため、実稼働環境での使用は推奨されません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Decision Central で DMN デザイナーを有効にするには、ウィンドウ右上で **Settings → Roles** とクリックし、左側のパネルからロールを選択し、**Editors → DMN Designer → Read** とクリックしてから **Save** をクリックして変更を保存します。

Red Hat テクノロジープレビュー機能の詳細は テクノロジープレビュー機能のサポート範囲 を参照してください。

手順

1. Decision Central で、**Menu → Design → Projects** に移動して、プロジェクト名をクリックします。
2. Decision Central プロジェクトで DMN ファイルを作成するか、インポートします。
   DMN ファイルを作成するには、**Add Asset → DMN** をクリックし、わかりやすい DMN モデル名を入力して、適切な **Package** を選択してから、**Ok** をクリックします。

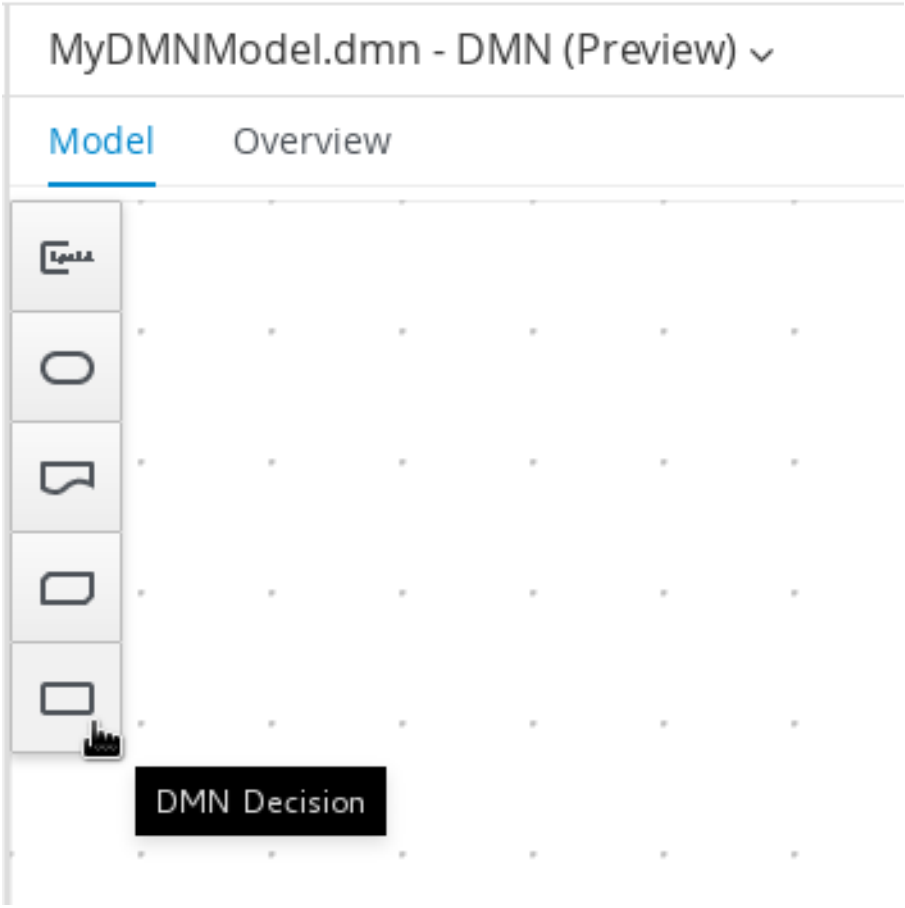
   既存の DMN ファイルをインポートするには、**Import** をクリックし、DMN モデル名を入力して、適切な **Package** を選択し、アップロードする DMN ファイルを選択してから **Ok** をクリックします。

   新しい DMN ファイルが **Project Explorer** の **DMN** パネルに表示され、DMN 意思決定要件ダイアグラム (DRD) のキャンバスが表示されます。

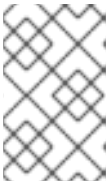
   注記

   レイアウトの情報が含まれていない DMN ファイルをインポートした場合は、インポートした意思決定要件ダイアグラム (DRD) は DMN デザイナーで自動的にフォーマットされます。DMN デザイナーで **Save** をクリックして、DRD レイアウトを保存します。
3. 左側のツールバーから DMN ノードの 1 つをクリックしてドラッグし、新規またはインポートした DMN 意思決定要件ダイアグラム (DRD) にコンポーネントを追加しはじめてください。



以下の DRD コンポーネントを利用できます。

- **デシジョン**: DMN ディジョンにこのノードを使用します。1 つ以上の要素が定義したデシジョンロジックをもとに出力を決定するノード。
- **ビジネスナレッジモデル**:1 つまたは複数のデシジョン要素が含まれる再利用可能な関数には、このノードを使用します。同じロジックですが、サブの入力または決定が異なるため、ビジネスナレッジモデルを使用してどの手順に従うかを決定します。
- **ナレッジソース**: デシジョンまたはビジネスナレッジモデルを規定する外部の機関、ドキュメント、委員会またはポリシーにはこのノードを使用します。ナレッジソースは、実行可能なビジネスルールではなく、実際の要因への参照となります。
- **入力データ**: デシジョンノードまたはビジネスナレッジモデルで使用する情報にはこのノードを使用します。入力データには通常、融資戦略で使用するローン申請データなど、ビジネスに関連するビジネスレベルのコンセプトまたはオブジェクトが含まれます。
- **テキストの注釈**: 入力データノード、デシジョンノード、ビジネスナレッジモデル、またはナレッジソースに関連する注釈にはこのノードを使用します。



注記

デシジョンサービスノードは、現在 Red Hat Decision Manager 7.2 DMN デザイナーではサポートされていません。このサポートは今後のリリースで提供されます。

4. DMN デザイナーキャバスで、新規の DRD ノードをダブルクリックして情報ノード名を入力します。

5. ノードがデシジョンまたはビジネスナレッジモデルの場合は、ノードオプションを表示するノードを選択して **Edit** アイコンをクリックし、DMN ボックス式を開き、ノードのデシジョンロジックを定義します。

図3.1 新規デシジョンノードのボックス式の表示

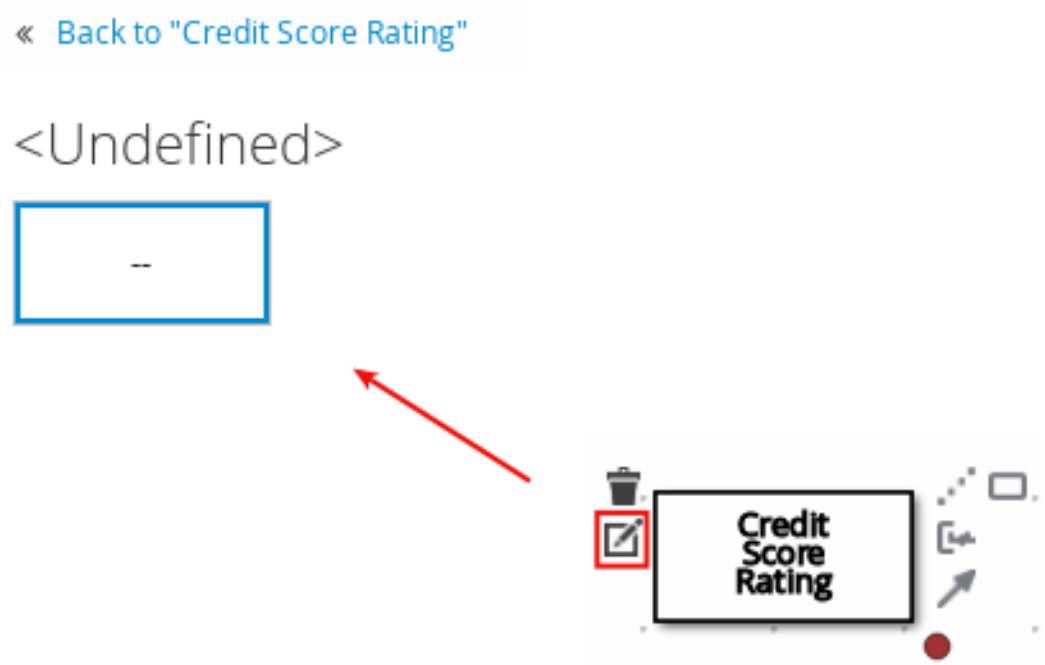
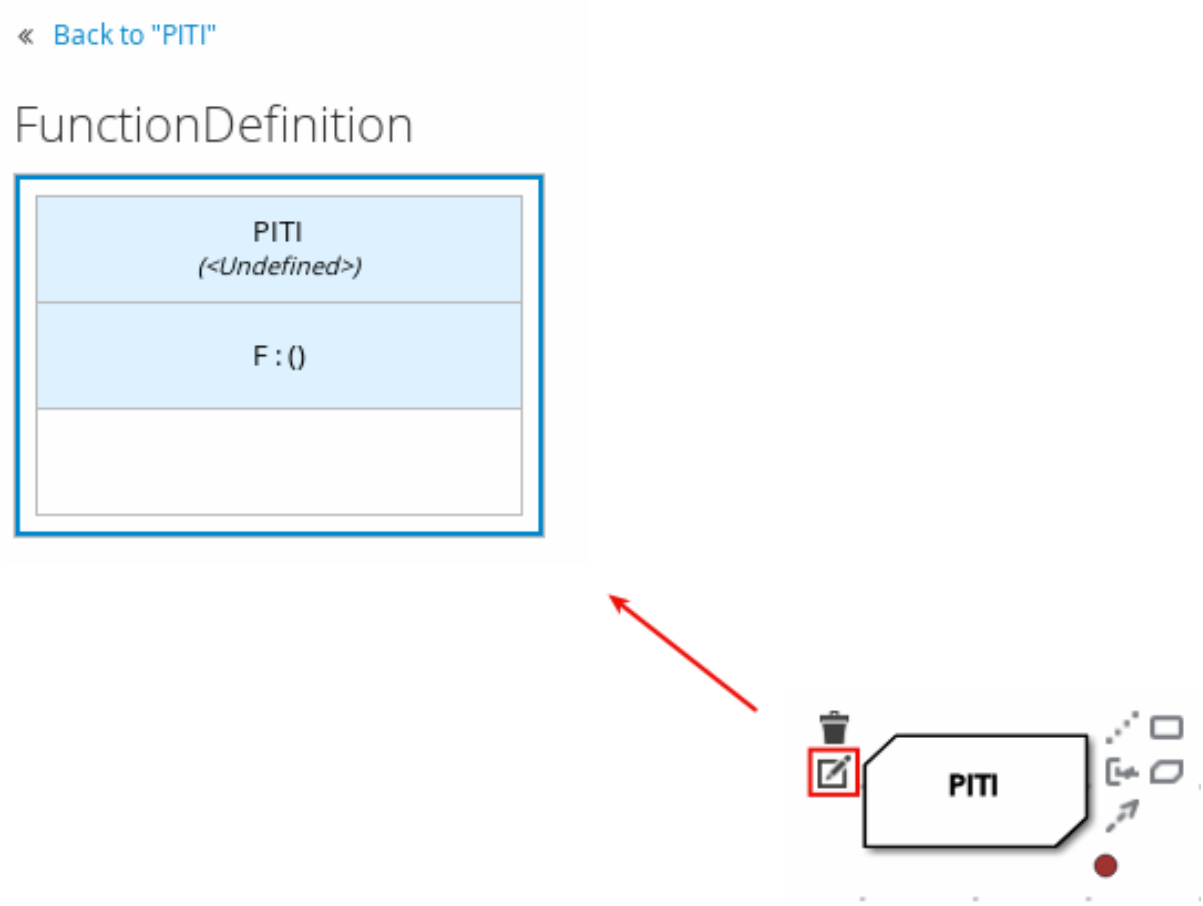
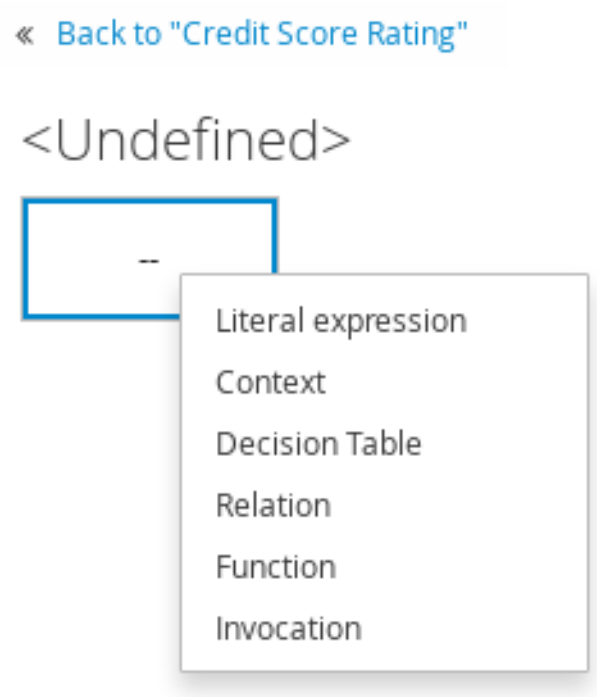


図3.2 新規ビジネスナレッジモデルのボックス式の表示

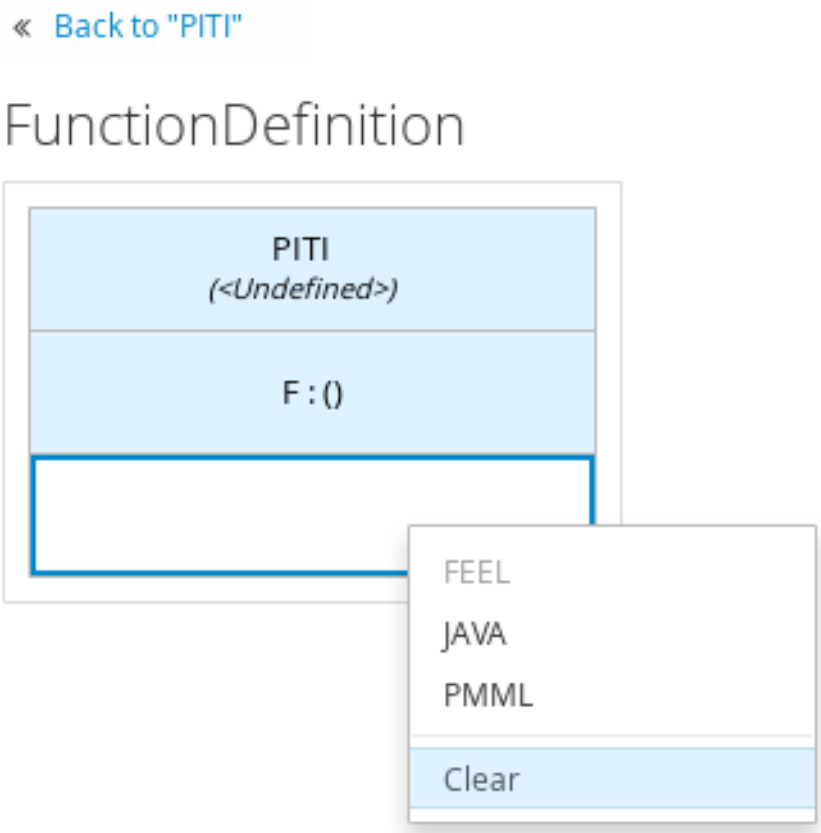


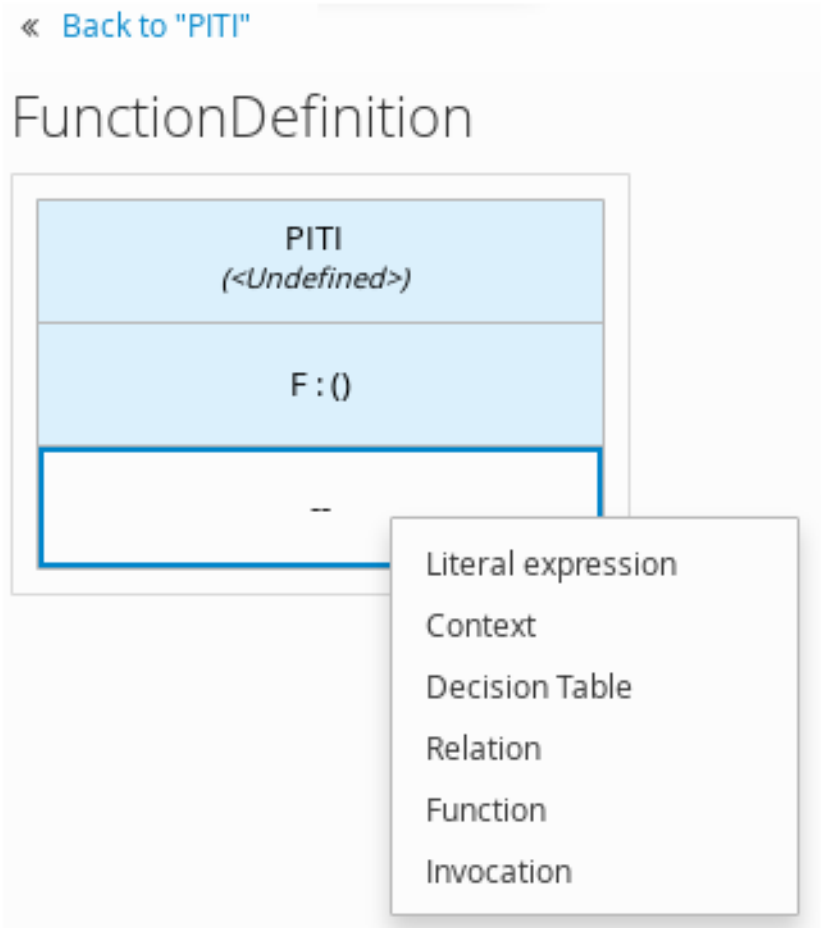
デフォルトでは、ビジネスナレッジモデルはすべて、リテラル FEEL 式、外部の JAVA または PMML 関数のネスト化されたコンテキスト式、またはあらゆる型のネスト化されたボックス式を含む、ボックス関数式として定義されます。

デシジョンノードの場合は、定義されていないテーブルをダブルクリックし、ボックスリテラル式、ボックスコンテキスト式、デシジョンテーブル、またはその他の DMN ボックスコンテキスト式など、使用するボックス式のタイプを選択します。



ビジネスナレッジモデルの場合は、値セルを右クリックして使用する式の種類を選択するか、**Clear** をクリックし、クリアしたセルをダブルクリックして、別の型のボックス式を選択します。





6. デザインノード (任意の式タイプ) またはビジネスナレッジモデル (関数式) のいずれかに対して選択したボックス式デザイナーで、該当するテーブルセルをダブルクリックして、デシジョンロジックに含めるテーブル名、変数データ型、変数名と値、関数パラメーターとバインディング、または FEEL 式を定義します。

セルを右クリックして、テーブルの行および列の挿入または削除、テーブルのコンテンツの消去など、随時、追加のアクションを実行します。

以下は、ローン申請者のクレジットスコアの定義範囲をもとに、クレジットスコアの評価を決定するデシジョンノードのデシジョンテーブルの一例です。

図3.3 クレジットスコア評価のデシジョンノードのデシジョンテーブル

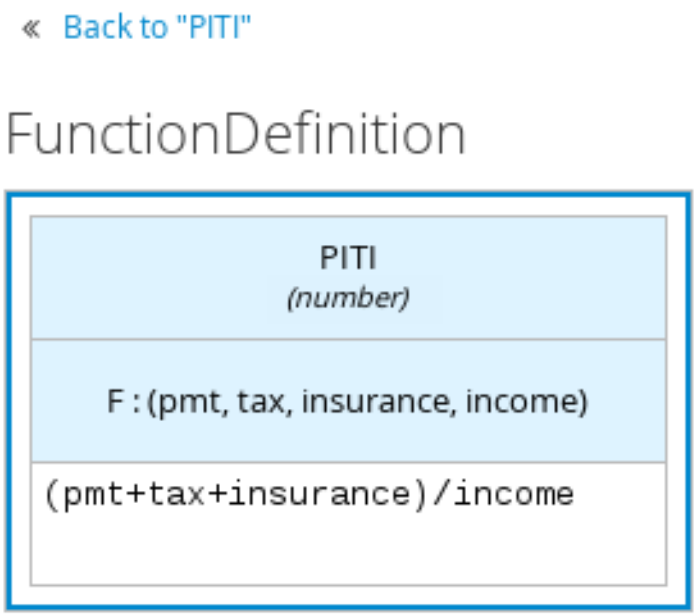
« Back to "Credit Score Rating"

DecisionTable

U	Credit Score.FICO *(number)*	Credit Score Rating *(Credit_Score_Rating)*	Description
1	>= 750	"Excellent"	
2	[700..750)	"Good"	
3	[650..700)	"Fair"	
4	[600..650)	"Poor"	
5	< 600	"Bad"	

以下は、元金、利子、税金、保険 (PITI) をもとに、リテラル式として住宅ローンの支払額を計算するビジネスナレッジモデルのボックス関数式の一例です。

図3.4 PITI 計算のビジネスナレッジモデルの関数



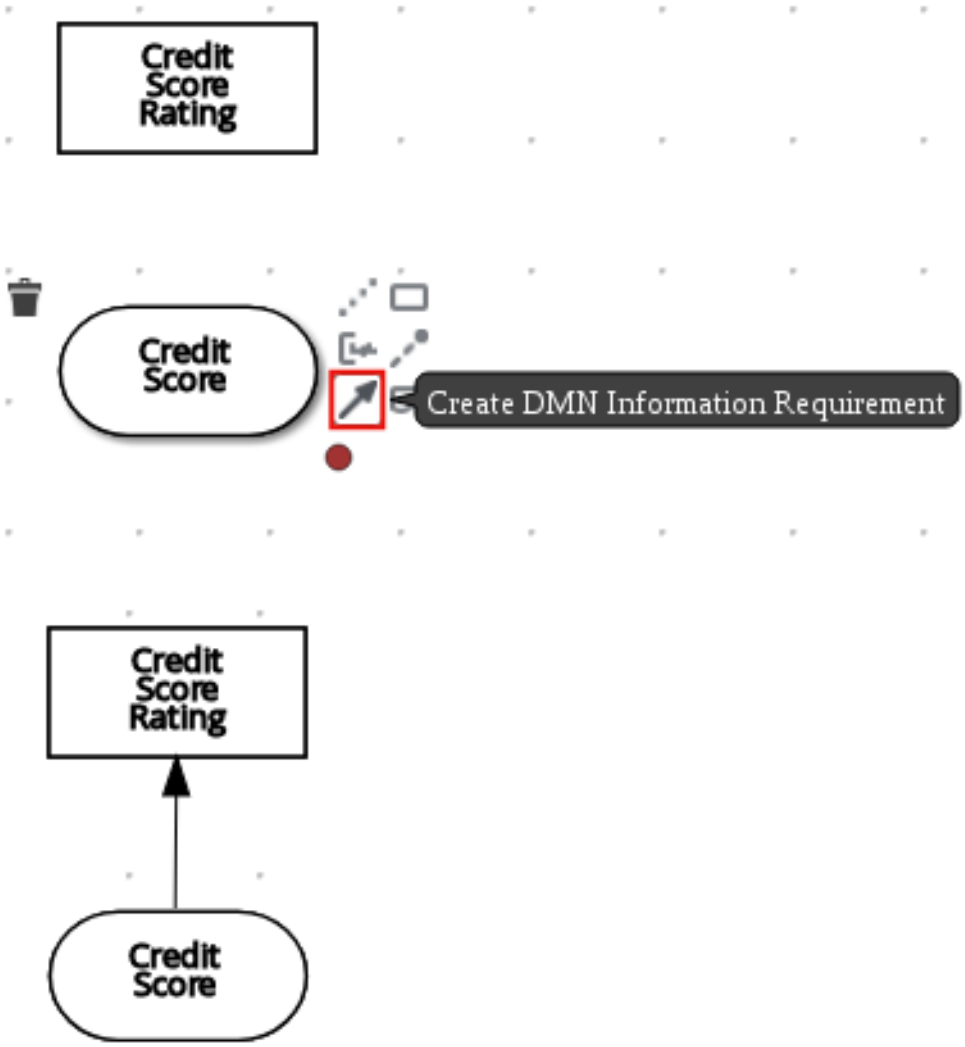
7. 選択したノードのデシジョンロジックを定義した後に、**Back to "<NODE_NAME>"** をクリックして DRD ビューに戻ります。

8. 選択した DRD ノードについては、利用可能な接続オプションを使用して、DRD の次のノードを作成して接続するか、左のツールバーから DRD キャンバスに新規ノードをクリックしてドラッグします。
ノードタイプで、どの接続オプションがサポートされているかが決まります。たとえば、**入力データ** ノードは、アプリケーションの接続タイプを使用してデシジョンオード、ナレッジソース、またはテキストの注釈を接続できますが、**ナレッジソース** ノードは、どの DRD 要素にでも接続できます。**デシジョン** ノードは、別のデシジョンまたはテキスト注釈にだけ接続できます。

   以下の接続タイプは、ノードの種類に応じて利用できます。

   - **情報要件**: 入力データノードまたはデシジョンノードから、情報を必要とする別のデシジョンノードに移動するにはこの接続を使用します。
   - **ナレッジ要件**: ビジネスナレッジモデルからデシジョンロジックを呼び出す別のビジネスナレッジモデルまたはデシジョンノードに移動するにはこの接続を使用します。
   - **認証局の要件**: 入力データノードまたはデシジョンノードから従属するナレッジソース、またはナレッジソースからデシジョンノード、ビジネスナレッジモデルまたは別のナレッジソースに移動するにはこの接続を使用します。
   - **関連付け**: 入力データノード、デシジョンノード、ビジネスナレッジモデル、またはナレッジソースからテキストアノテーションに移動するにはこの接続を使用します。

図3.5 クレジットスコアの入力からクレジットスコア評価のデシジョンへの接続



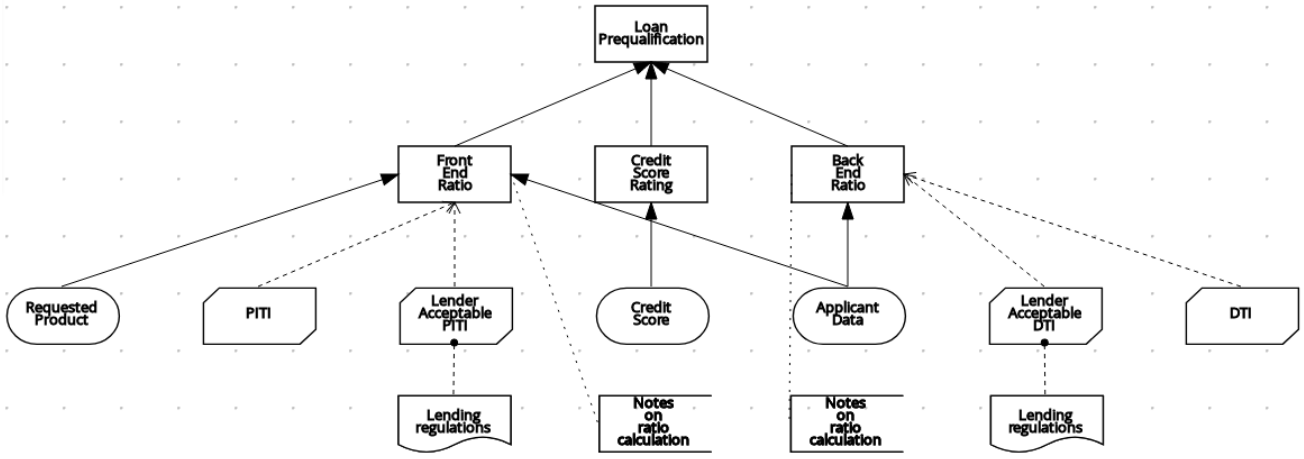
9. 継続して、デシジョンモデルの残りの DRD コンポーネントを追加し、定義します。DMN デザイナーで定期的に **Save** をクリックして作業を保存します。

10. DRD の全コンポーネントを追加して定義した後に、**Save** をクリックし、完了した DRD を保存して検証します。

+

以下は、ローンの事前審査デシジョンモデルの DRD の一例です。

図3.6 ローンの事前審査の完全な DRD



3.1. DECISION CENTRAL でボックス式を使用した DMN デシジョンロジックの定義

DMN のボックス式は、意思決定要件ダイアグラム (DRD) または意思決定要件グラフ (DRG) でデシジョンノードの基盤ロジックを定義するのに使用するテーブルです。ボックス式には他のボックス式が含まれる場合がありますが、トップレベルのボックス式は単一の DRD アーティファクトのデシジョンロジックに対応します。1 つまたは複数の DRG が含まれる DRD は、DMN デシジョンモデルのフローを表現し、反対にボックス式は個別ノードの実際のデシジョンロジックを定義します。DRD とボックス式は、完全で機能的な DMN デシジョンモデルを形成します。

Decision Central で DMN デザイナーを使用して、同梱のボックス式で DRD コンポーネントのデシジョンロジックを定義できます。

前提条件

- Decision Central で DMN ファイルを作成しているか、インポートしている。

手順

1. Decision Central で **Menu → Design → Projects** に移動して、プロジェクト名をクリックし、変更する DMN ファイルを選択します。
2. DMN デザイナーのキャンバスで、定義するデシジョンノードまたはビジネスナレッジモデルを選択し、**Edit** アイコンをクリックして DMN ボックス式デザイナーを開きます。

図3.7 新規デシジョンノードのボックス式の表示

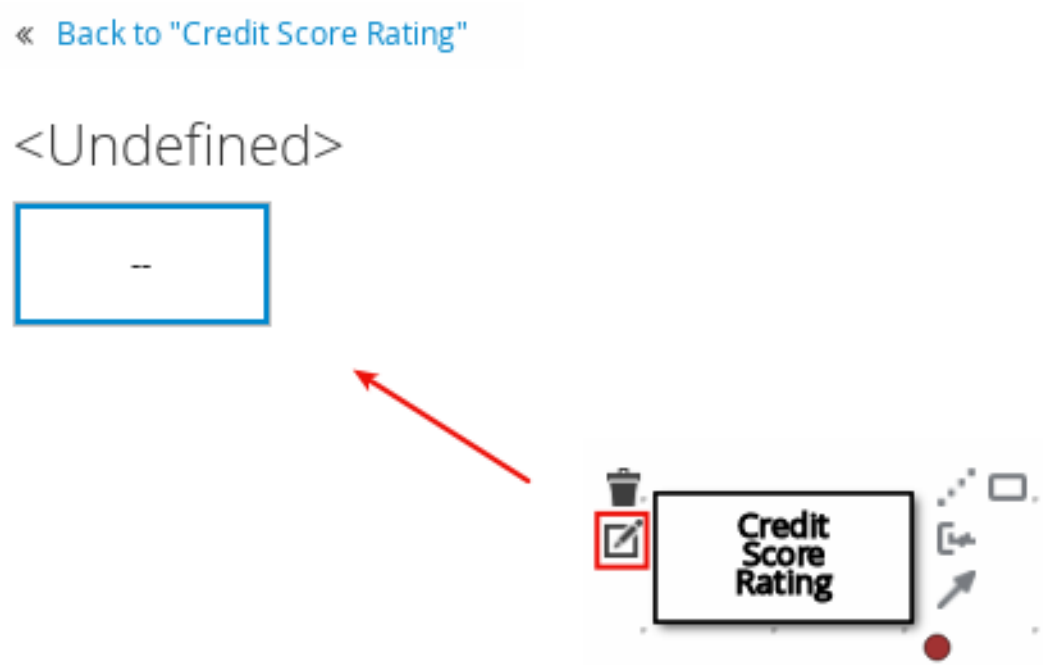
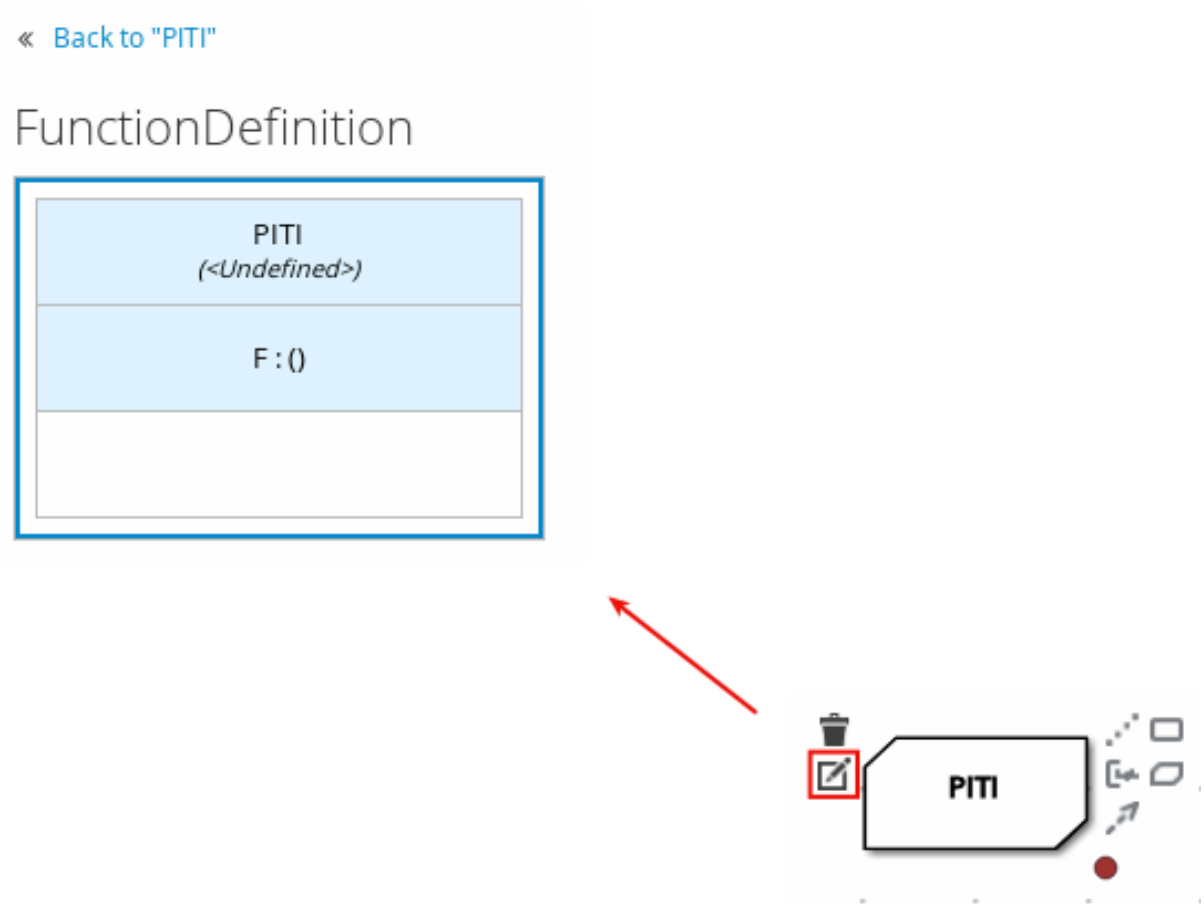
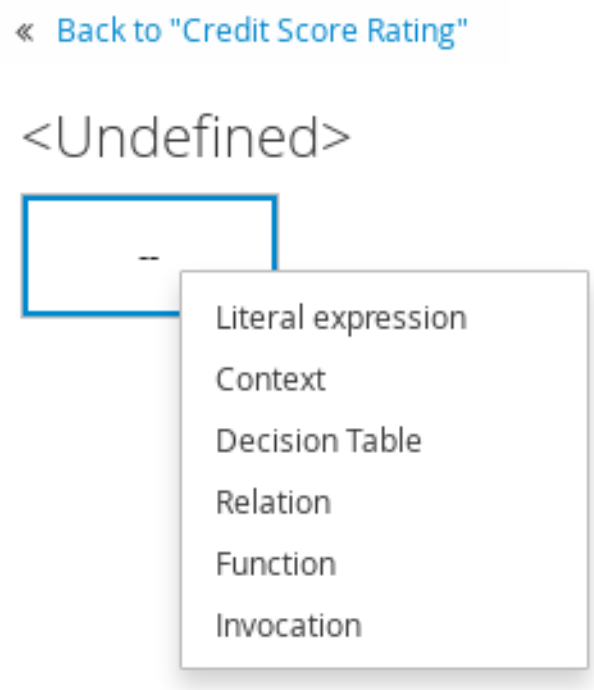


図3.8 新規ビジネスナレッジモデルのボックス式の表示

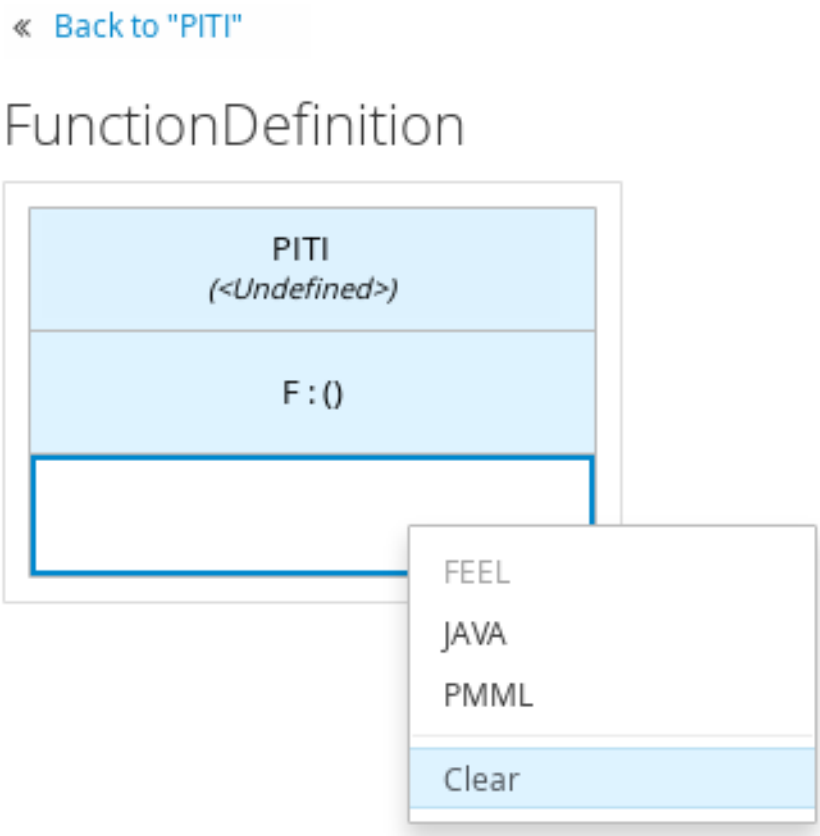


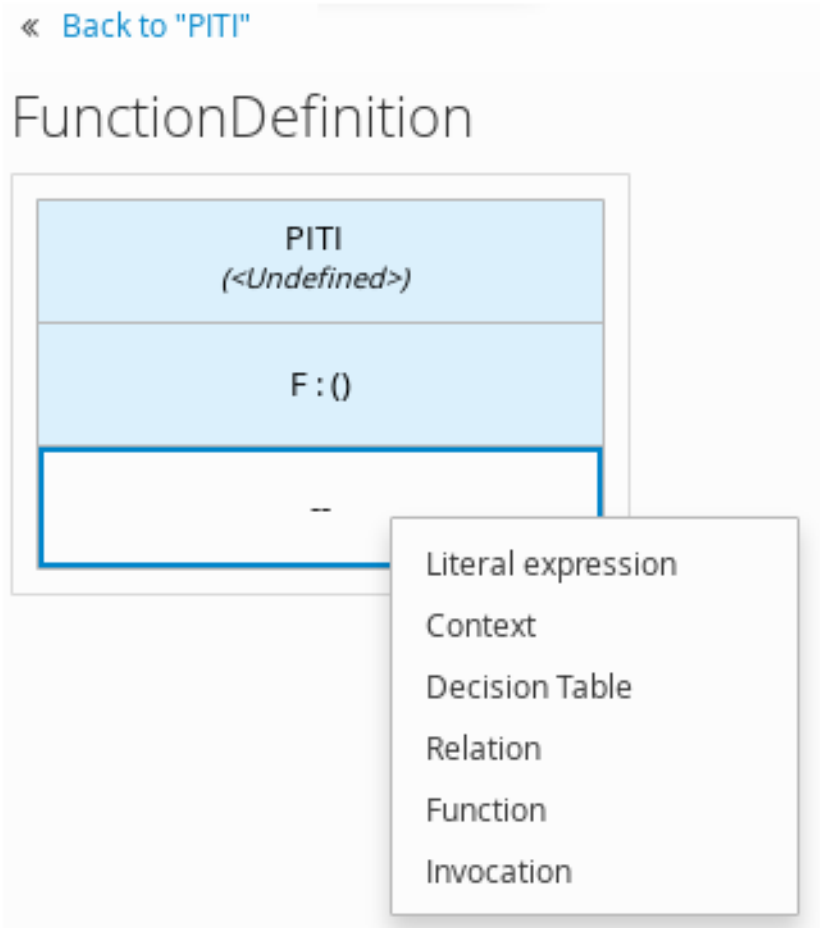
デフォルトでは、ビジネスナレッジモデルはすべて、リテラル FEEL 式、外部の JAVA または PMML 関数のネスト化されたコンテキスト式、またはあらゆる型のネスト化されたボックス式を含む、ボックス関数式として定義されます。

デシジョンノードの場合は、定義されていないテーブルをダブルクリックし、ボックスリテラル式、ボックスコンテキスト式、デシジョンテーブル、またはその他の DMN ボックスコンテキスト式など、使用するボックス式のタイプを選択します。



ビジネスナレッジモデルの場合は、値セルを右クリックして使用する式の種類を選択するか、**Clear** をクリックし、クリアしたセルをダブルクリックして、別の型のボックス式を選択します。

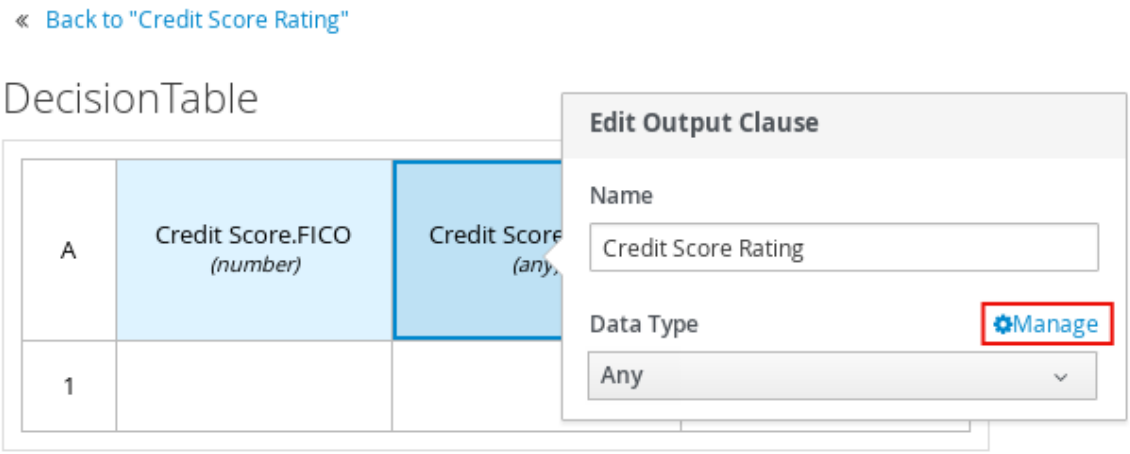




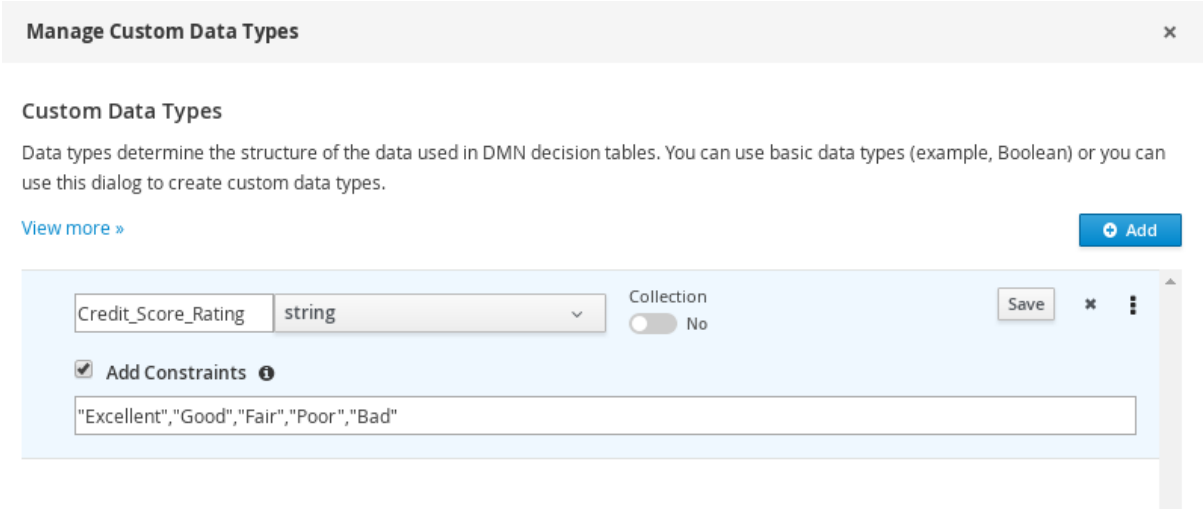
3. この例では、デシジョンノードを使用して、ボックス式タイプとして **デシジョンテーブル** を使用します。
DMN のデシジョンテーブルは、1 つ以上のルールをテーブル形式で視覚的に表します。テーブルの各行はルール 1 つで設定されており、その特定行に対する条件 (入力) と結果 (出力) を定義する列が含まれます。

4. 入力列ヘッダーをダブルクリックして、入力条件の名前とデータ型を定義します。たとえば、入力列に **Credit Score.FICO** という名前で、**number** のデータ型を指定します。この列は、数字のクレジットスコア値または、各種ローン申請者を指定します。

5. 出力列ヘッダーをダブルクリックして、出力値の名前とデータ型を定義します。たとえば、出力列に **Credit Score Rating** という名前を指定して、**Data Type** オプションの横で **Manage** をクリックし、スコア評価を制約として、カスタムのデータ型を作成します。



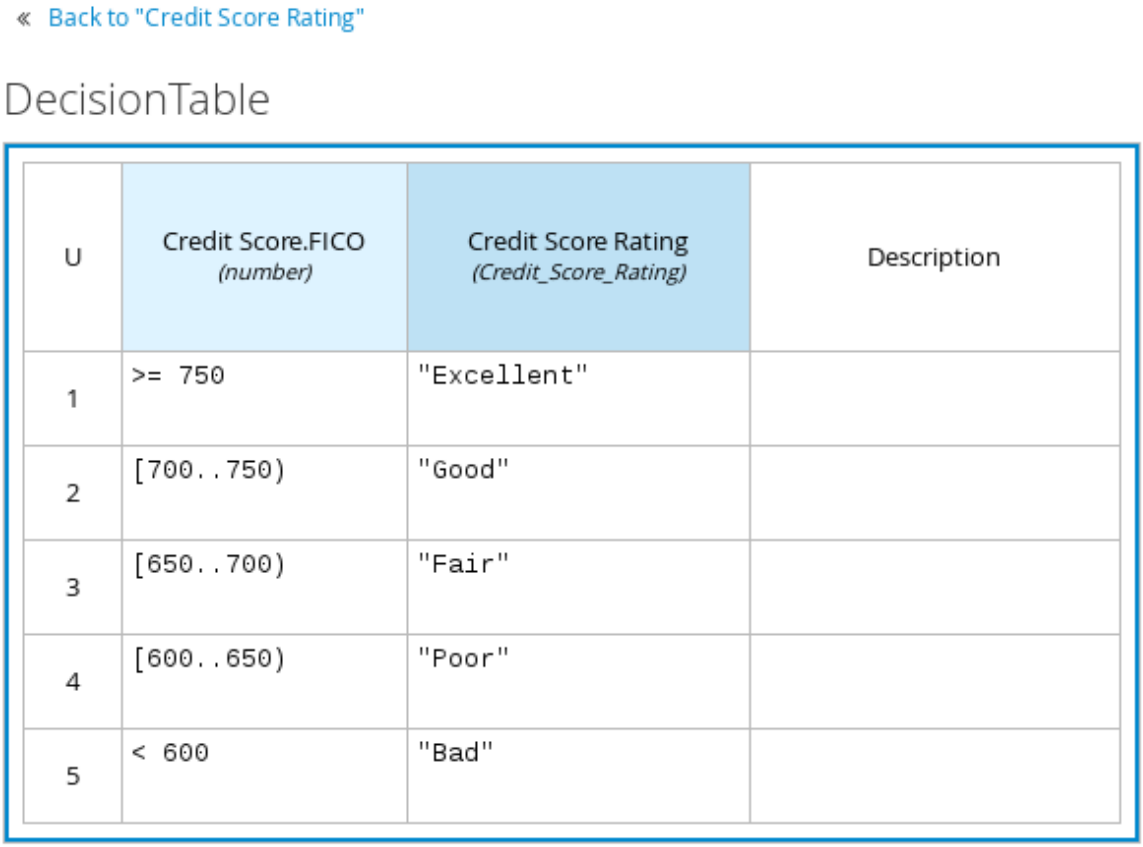
6. **Manage Custom Data Types** ウィンドウで、**Add** をクリックして、**"Excellent","Good","Fair","Poor","Bad"** の制約で **string** として **Credit_Score_Rating** のデータ型を作成します。



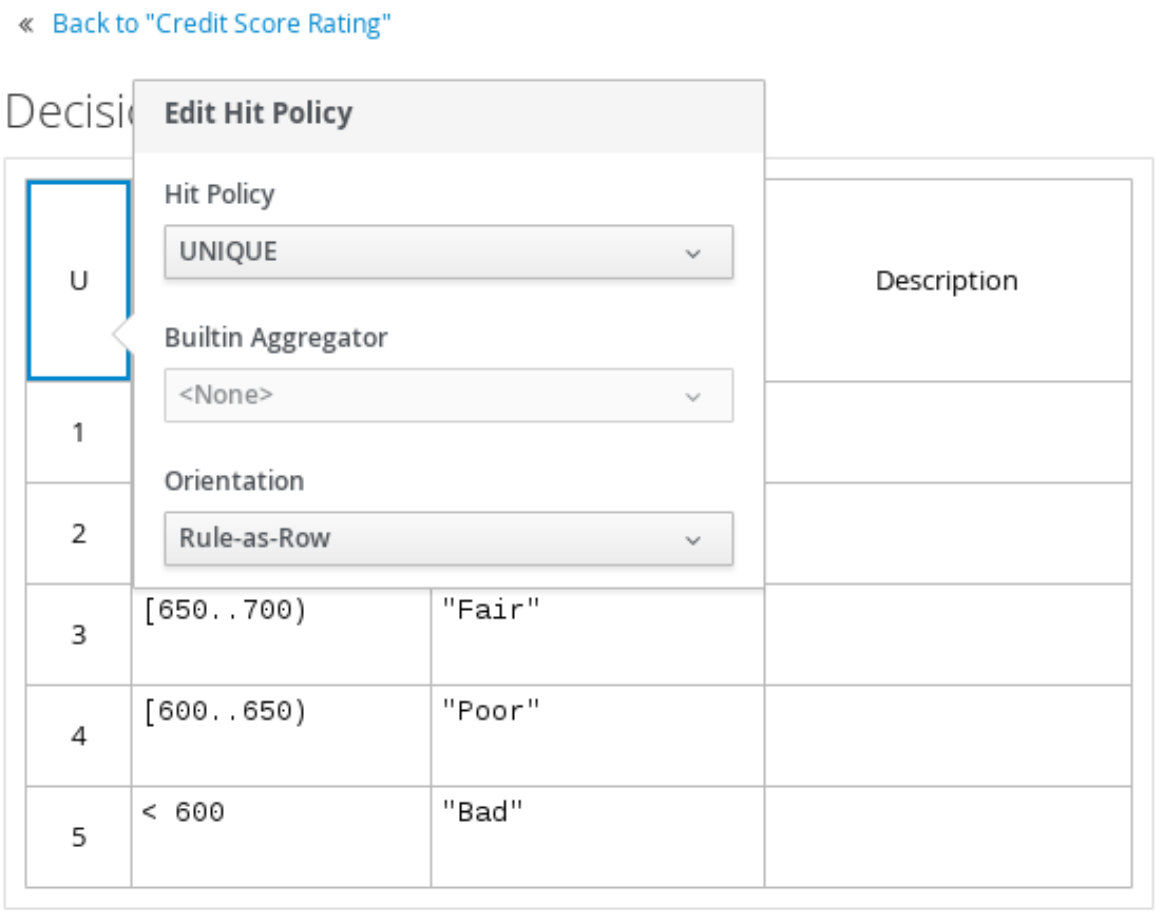
7. **Save** をクリックしてデータ型を保存し、デシジョンテーブルで **Credit Score Rating** 列ヘッダーをダブルクリックして、保存したデータ型をこの新規カスタムデータ型に設定します。

8. **Credit Score.FICO** の入力列を使用して、クレジットスコアの値またはクレジットの範囲を定義し、**Credit Score Rating** 列を使用して、**Credit_Score_Rating** のデータ型で定義した対応する評価の 1 つを指定します。
値のセルを右クリックして、行 (ルール) および列 (句) を挿入または削除します。

図3.9 クレジットスコア評価のデシジョンノードのデシジョンテーブル



9. 全ルールを定義した後に、デシジョンテーブルの左上隅をクリックして、**ヒットポリシー** と **組み込みアグリゲーター** (**COLLECT** ヒットポリシーのみ) のルールを定義します。
ヒットポリシーは、デシジョンテーブルにある複数のルールが指定の入力値とマッチした場合にどのように結果に到達するのかを決定します。組み込みアグリゲーターは、**COLLECT** ヒットポリシーを使用する場合にどのようにルール値を累積するかを決定します。



注記

Orientation **オプション** は、Red Hat Decision Manager 7.2 DMN デザイナーでサポートされず、今後のリリースで Decision Central ユーザーインターフェイスから削除されます。このオプションは、行、列、または行と列(**CrossTable**)の両方が DMN デシジョンテーブルのルールを表すかどうかを決定します。

以下のデシジョンテーブルは、より複雑なデシジョンテーブルで、ローン事前審査のデシジョンモデルで終端デシジョンノードとして、申請者のローン適正を決定します。

図3.10 ローン事前審査のデシジョンテーブル

DecisionTable

F	Credit Score Rating *(Credit_Score_Rating)*	Back End Ratio *(Back_End_Ratio)*	Front End Ratio *(Front_End_Ratio)*	Loan Pre-Qualification *(Loan_Qualification)*		Description
				Qualification *(string)*	Reason *(string)*	
1	"Poor", "Bad"	-	-	"Not Qualified"	"Credit Score too low."	
2	-	"Insufficient"	"Sufficient"	"Not Qualified"	"Debt to income ratio is too high."	
3	-	"Sufficient"	"Insufficient"	"Not Qualified"	"Mortgage payment to income ratio is too high."	
4	-	"Insufficient"	"Insufficient"	"Not Qualified"	"Debt to income ratio is too high AND mortgage payment to income ratio is too high."	
5	"Fair", "Good", "Excellent"	"Sufficient"	"Sufficient"	"Qualified"	"The borrower has been successfully prequalified for the requested loan."	

デシジョンテーブル以外のボックス式タイプの場合は、ボックス式のテーブルを移動して、デシジョン

ロジックの変数とパラメーターを定義するのと同様にこれらのガイドラインに従いますが、ボックス式のタイプの要件に従うようにしてください。ボックスリテラル式など、ボックス式の一部では列が 1 つのテーブルの場合や、関数、テキスト、呼び出し式などのボックス式は、他のタイプのボックス式がネスト化された、列が複数あるテーブルの場合もあります。

たとえば、以下のボックスコンテキスト式では、サブコンテキスト式が含まれるフロントエンドの割合計算として表現されている PITI (元金 (Principal)、利子 (Interest)、税金 (Tax)、保険 (Insurance)) をもとに、ローンの申請者が最小限必要とされるローンの支払いをしているかを決定するパラメーターを定義します。

図3.11 フロントエンドクライアント PITI 割合のボックスコンテキスト式

Context

#	Front End Ratio *(Front_End_Ratio)*	
1	Client PITI *(number)*	# | PITI 1 | pmt *(<Undefined>)* | `(Requested Product.Amount*((Requested Product.Rate/100)/12))/ (1-(1/(1+(Requested Product.Rate/100)/12)**-Requested Product.Term))` 2 | tax *(<Undefined>)* | `Applicant Data.Monthly.Tax` 3 | insurance *(<Undefined>)* | `Applicant Data.Monthly.Insurance` 4 | income *(<Undefined>)* | `Applicant Data.Monthly.Income`
	<result>	`if Client PITI <= Lender Acceptable PITI() then "Sufficient" else "Insufficient"`

以下のボックス関数式では、ネスト化されたコンテキスト式として定義された関数値を使用し、融資のデシジョンのビジネスナレッジモデルとして、住宅ローンの月額を決定します。

図3.12 ビジネスナレッジモデルのローン計算で使用するボックス関数式

FunctionDefinition

InstallmentCalculation *(number)*		
F : (ProductType, Rate, Term, Amount)		
1	MonthlyFee *(number)*	`if ProductType ="STANDARD LOAN" then 20.00 else if ProductType ="SPECIAL LOAN" then 25.00 else null`
2	MonthlyRepayment *(number)*	`(Amount *Rate/12) / (1 - (1 + Rate/12)**-Term)`
	<result>	`MonthlyRepayment+MonthlyFee`

各ボックス式のタイプの例および詳細は、「ボックス式の DMN デシジョンロジック」 を参照してください。

3.2. DECISION CENTRAL での DMN ボックス式のカスタムデータ型の作成

Decision Central の DMN のボックス式では、データ型により、ボックス式の関連のテーブル、列、またはフィールド内で使用するデータの構造を決定します。デフォルトの DMN データ型 (文字列、数字、ブール値など) を使用するか、独自のデータ型を作成して、ボックス式の値に実装する新たなフィールドや制限を指定することもできます。

ボックス式のカスタムのデータ型として、単純なデータ型または構造化されたデータ型のいずれかを作成できます。

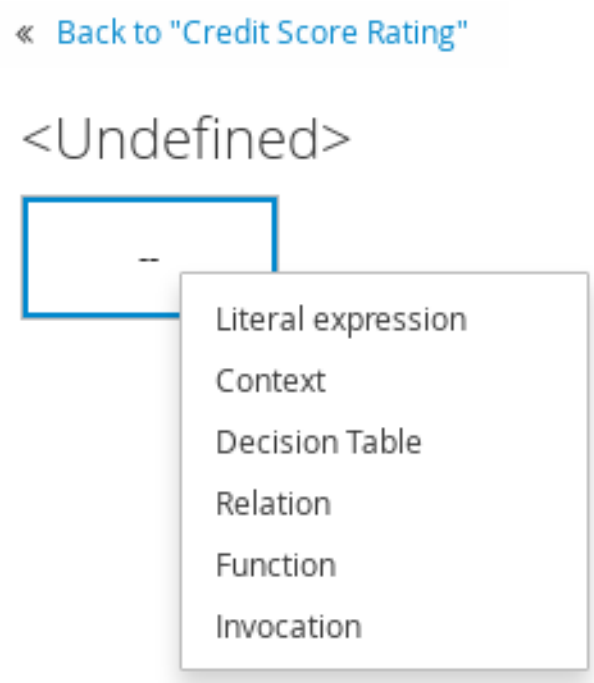
- **単純な** データ型では、名前とタイプの割当のみを指定できます。例: **Age (number)**
- **構造化された** データ型には、親データ型に関連する複数のフィールドが含まれます。例: **Name (string)**、**Age (number)**、**email (string)** のフィールドが含まれる単一の型 **Person**

前提条件

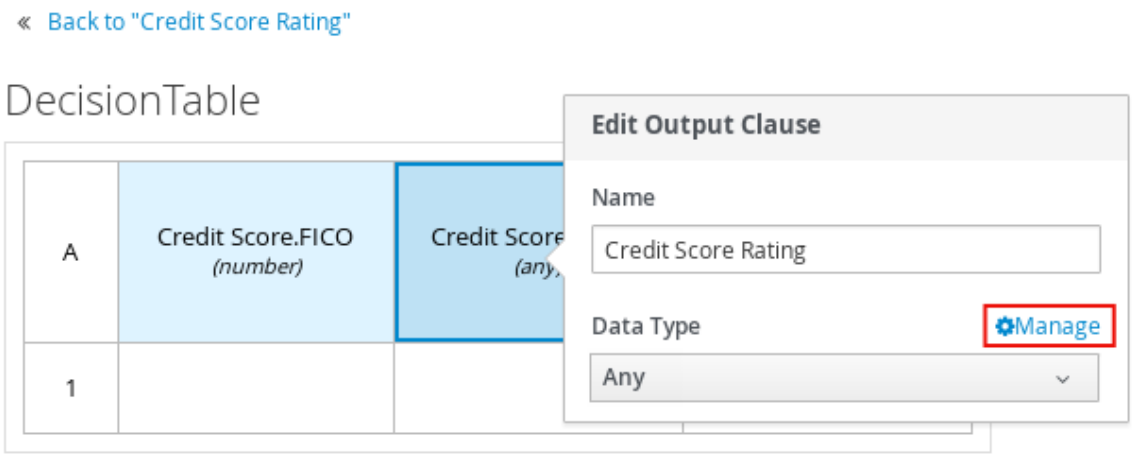
- Decision Central で DMN ファイルを作成しているか、インポートしている。

手順

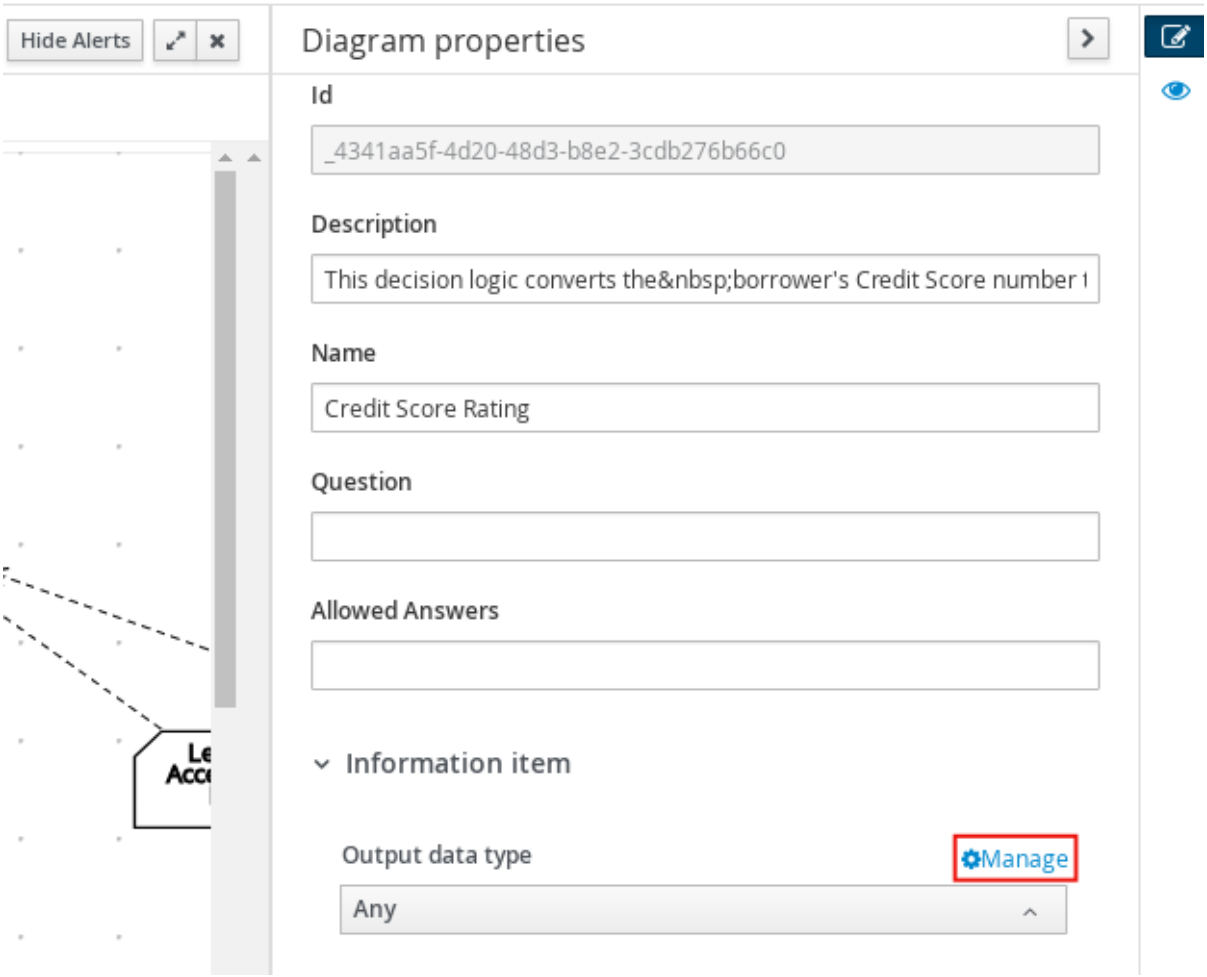
1. Decision Central で **Menu → Design → Projects** に移動して、プロジェクト名をクリックし、変更する DMN ファイルを選択します。
2. DMN デザイナーのキャンバスで、データ型を定義するデシジョンノードまたはビジネスナレッジモデルを選択し、**Edit** アイコンをクリックして DMN ボックス式デザイナーを開きます。
3. ボックス式が定義されていないデシジョンノードの場合は、定義されていないテーブルをダブルクリックし、ボックスリテラル式、ボックスコンテキスト式、デシジョンテーブル、またはその他の DMN ボックスコンテキスト式など、使用するボックス式のタイプを選択します。



4. データ型を定義するテーブルヘッダー、列ヘッダー、またはパラメーターフィールド (ボックス式の種類による) のセルをダブルクリックし、**Manage** をクリックして、カスタムのデータ型を作成します。



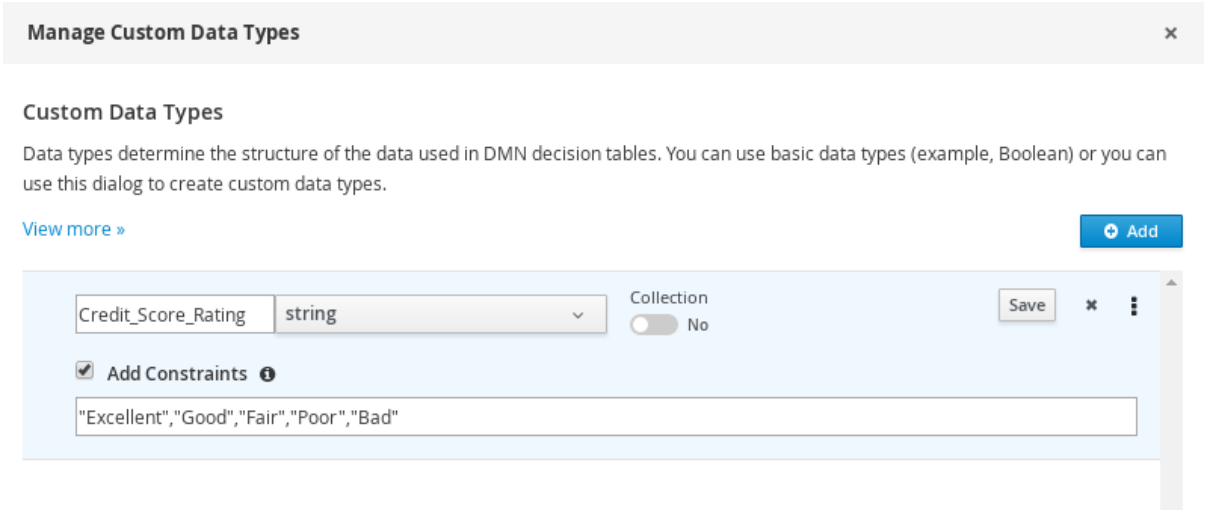
DMN デザイナーの右上隅にある **Diagram properties** アイコンを選択して、カスタムのデータ型を管理することもできます。



ボックス式で指定のセルに定義するデータ型により、ボックス式の関連のテーブル、列、フィールド内で使用するデータの構造を決定します。

この例では、DMN デシジョンテーブルの出力列 **クレジットスコア評価** は、申請者のクレジットスコアをもとにカスタムのクレジットスコア評価を定義します。

5. **Manage Custom Data Types**ウィンドウで、**Add** をクリックして、**"Excellent","Good","Fair","Poor","Bad"** の制約で **string** として Credit_Score_Rating のデータ型を作成します。



データ型がアイテムの一覧を必要とする場合は、**Collection** 設定を有効にします。

6. **Save** をクリックしてデータ型を保存し、デシジョンテーブルで **Credit Score Rating** 列ヘッダーをダブルクリックして、保存したデータ型をこの新規カスタムデータ型に設定し、指定した評価制約で、対象の列のルール値を定義します。

図3.13 クレジットスコア評価のデシジョンテーブル

« Back to "Credit Score Rating"

DecisionTable

U	Credit Score.FICO *(number)*	Credit Score Rating *(Credit_Score_Rating)*	Description
1	>= 750	"Excellent"	
2	[700..750)	"Good"	
3	[650..700)	"Fair"	
4	[600..650)	"Poor"	
5	< 600	"Bad"	

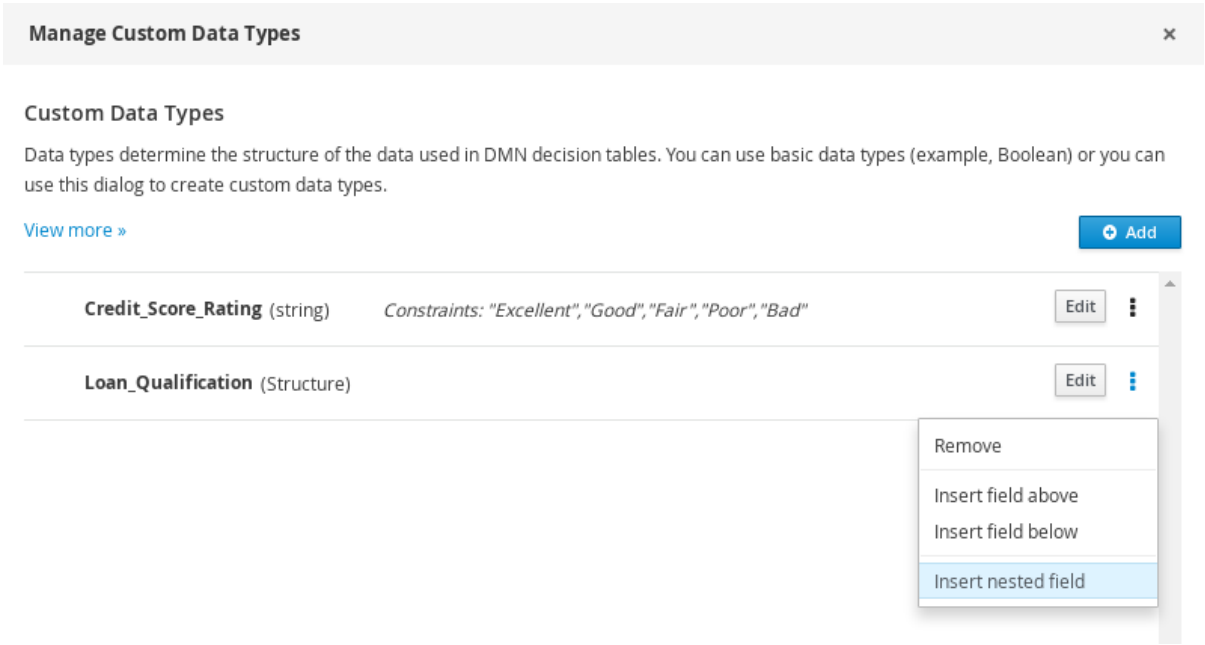
シナリオの DMN デシジョンモデルで、**Credit Score Rating** デシジョンが、以下の **Loan Prequalification** デシジョンに流れ、このデシジョンではカスタムのデータ型が必要です。

DecisionTable

F	Credit Score Rating *(<Undefined>)*	Back End Ratio *(<Undefined>)*	Front End Ratio *(<Undefined>)*	Loan Pre-Qualification *(<Undefined>)*		Description
				Qualification *(string)*	Reason *(string)*	
1						

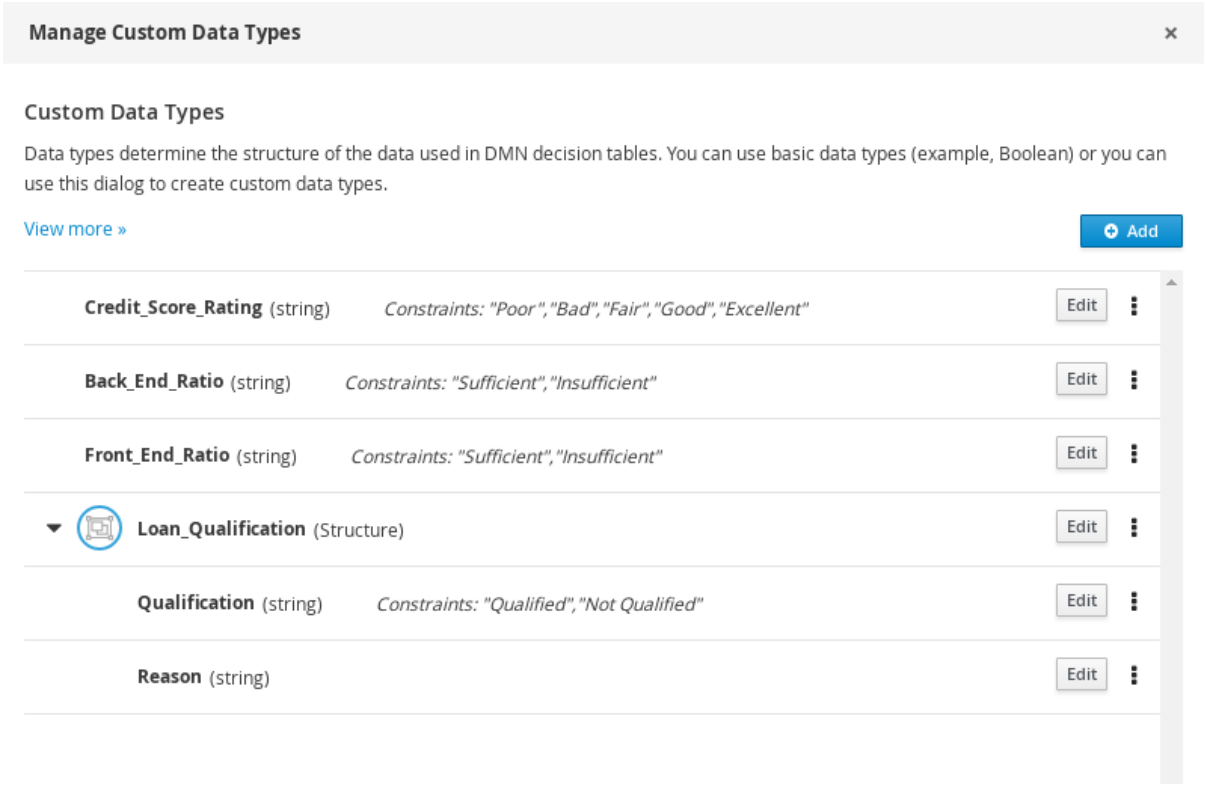
7. この例をそのまま使用し、列ヘッダーをダブルクリックして **Manage Custom Data Types** ウィンドウに戻り、**Add** をクリックして、**Loan_Qualification** データ型を制約なしの **Structure** として作成します。

8. **Loan_Qualification** のデータ型の横にある、設定アイコン (3 つの点) を選び、 **Insert nested field** を選択して、この親データ型のサブフィールドを挿入します。



デシジョンテーブル内にネスト化された列ヘッダー、コンテキストまたは関数式でネスト化されたパラメーターなど、ボックス式の構造化された親データ型と関連するこれらのサブフィールドを使用できます。

9. この例では、構造化された **Loan_Qualification** データ型に、**"Qualified","Not Qualified"** の制約がある **Qualification** フィールドと、制約のない **Reason** フィールドを追加します。また、**"Sufficient","Insufficient"** の制約がある、単純なデータ型 **Back_End_Ratio** と **Front_End_Ratio** を追加します。
データ型を作成するたびに、**Save** をクリックします。



10. デシジョンテーブルに戻り、列ごとに、列ヘッダーセルをダブルクリックし、対応する新規のカスタムデータ型に、このデータ型を設定して、(該当する場合には) 指定した制約で必要に応じて列のルール値を定義します。

図3.14 ローン事前審査のデシジョンテーブル

DecisionTable

F	Credit Score Rating *(Credit_Score_Rating)*	Back End Ratio *(Back_End_Ratio)*	Front End Ratio *(Front_End_Ratio)*	Loan Pre-Qualification *(Loan_Qualification)*		Description
				Qualification *(string)*	Reason *(string)*	
1	"Poor", "Bad"	-	-	"Not Qualified"	"Credit Score too low."	
2	-	"Insufficient"	"Sufficient"	"Not Qualified"	"Debt to income ratio is too high."	
3	-	"Sufficient"	"Insufficient"	"Not Qualified"	"Mortgage payment to income ratio is too high."	
4	-	"Insufficient"	"Insufficient"	"Not Qualified"	"Debt to income ratio is too high AND mortgage payment to income ratio is too high."	
5	"Fair", "Good", "Excellent"	"Sufficient"	"Sufficient"	"Qualified"	"The borrower has been successfully prequalified for the requested loan."	

デシジョンテーブル以外のボックス式タイプの場合は、ボックス式のテーブルを移動して、必要に応じてカスタムのデータ型を定義するのと同様に、これらのガイドラインに従うようにしてください。

たとえば、以下のボックス関数式はカスタムの **tCandidate** と **tProfile** の構造化データ型を使用して、データを関連付けてオンライン出会い系でのデート相手としての適合性を判断します。

図3.15 オンライン出会い系でデート相手としての適合性に使用するボックス関数式

FunctionDefinition

Evaluate Match *(tCandidate)*		
F : (Lonely Soul, Candidate)		
1	Profile1 *(tProfile)*	Lonely Soul
2	Profile2 *(tProfile)*	Candidate
3	Is Match *(boolean)*	Is Soul a Match(Lonely Soul, Candidate) and Is Soul a Match(Candidate, Lonely Soul)
4	Score *(number)*	Number of Matching Interests(Lonely Soul, Candidate) - absolute(Lonely Soul.Age - Candidate.Age)
	<result>	--

図3.16 オンライン出会い系でデート相手としての適合性に使用するカスタムデータ型の定義

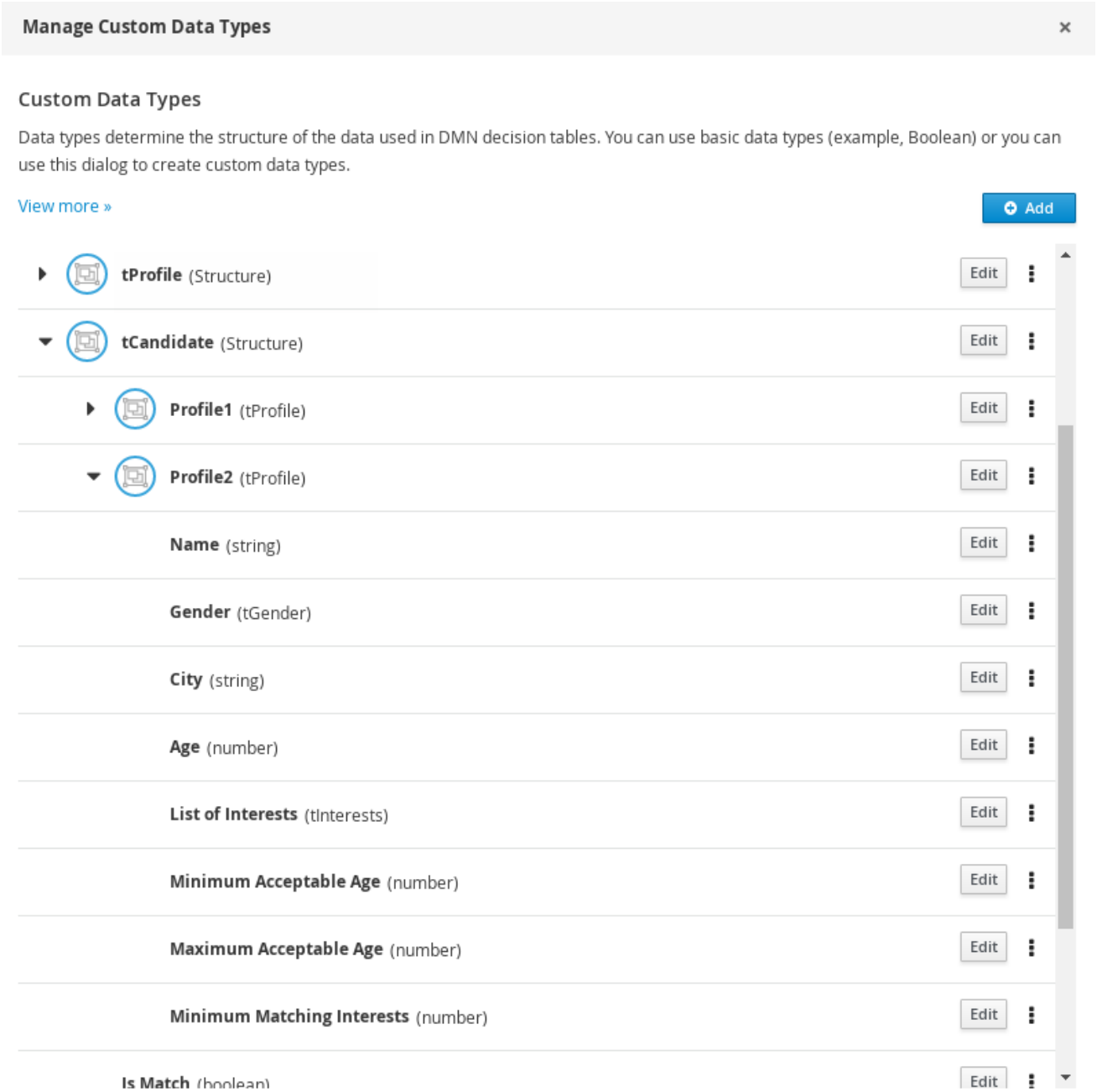
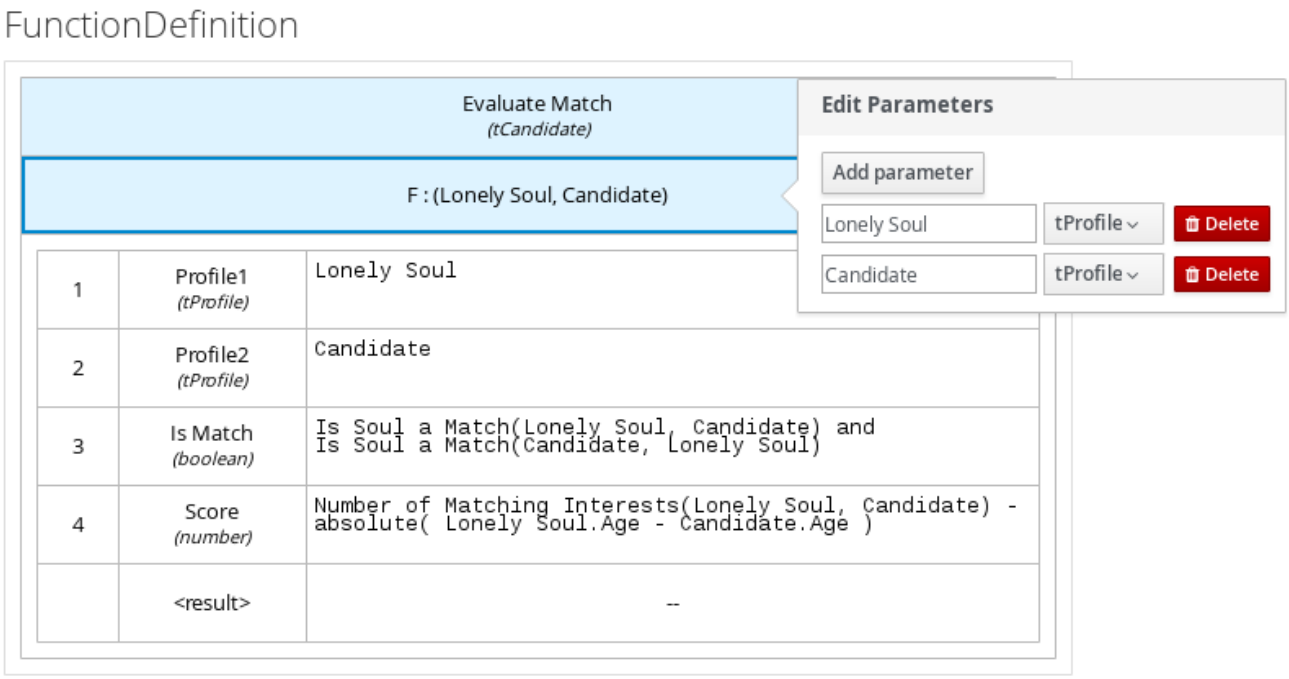


図3.17 オンライン出会い系でデート相手としての適合性に使用するカスタムデータ型を含むパラメーター定義



3.3. DECISION CENTRAL での DMN ナビゲーションとプロパティー

DMN デザイナーには、以下の追加機能が含まれており、意思決定要件ダイアグラム (DRD) のコンポーネントやプロパティーの移動を容易にします。

DMN ファイルとダイアグラムのビュー

DMN デザイナーの左上隅で、**Project Explorer** ビューを選択して、全 DMN と他のファイルを移動するか、**Decision Navigator** ビューを選択して、ノードと、選択した DRD のボックス式間を移動します。

図3.18 Project Explorer のビュー

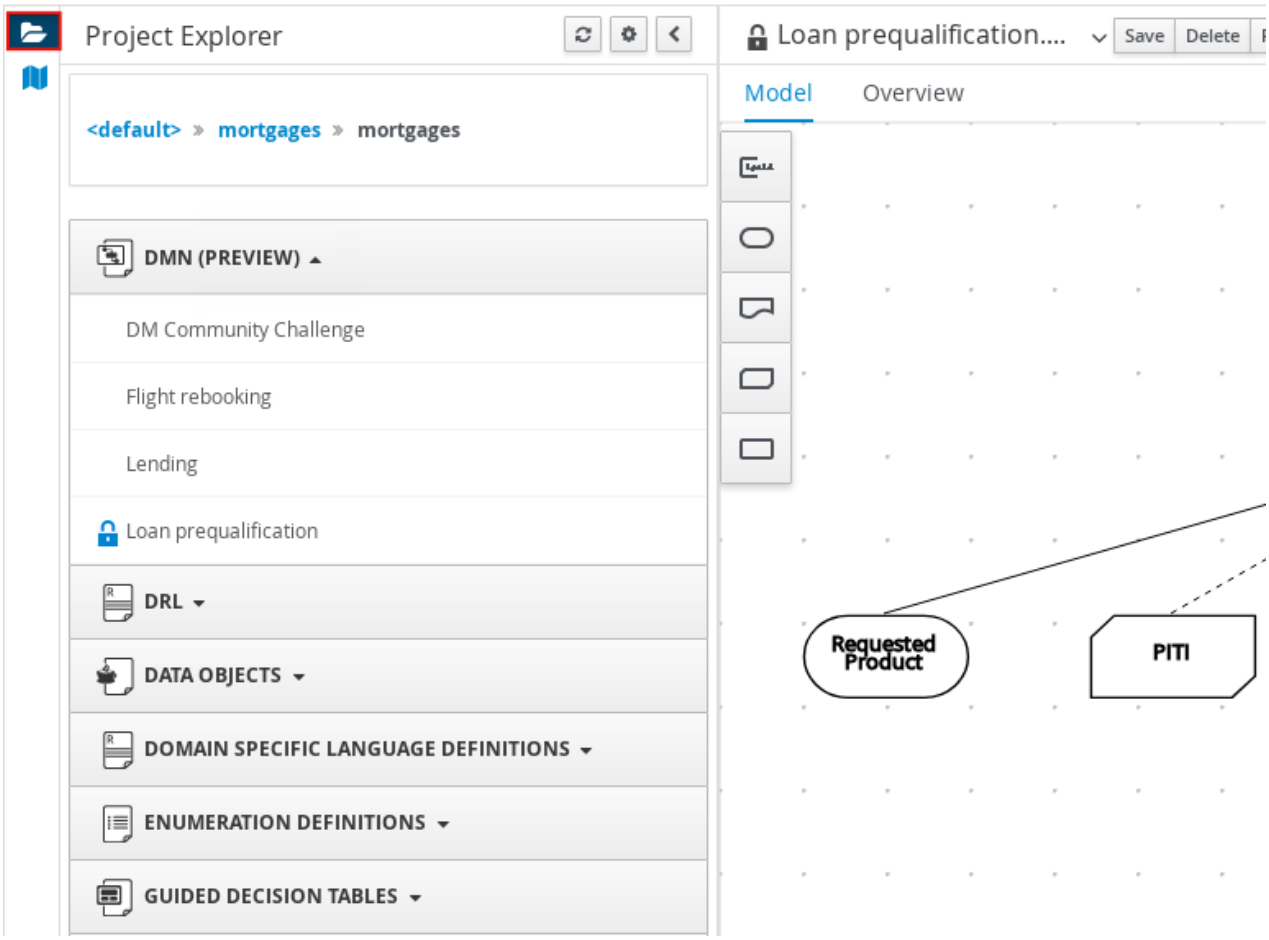
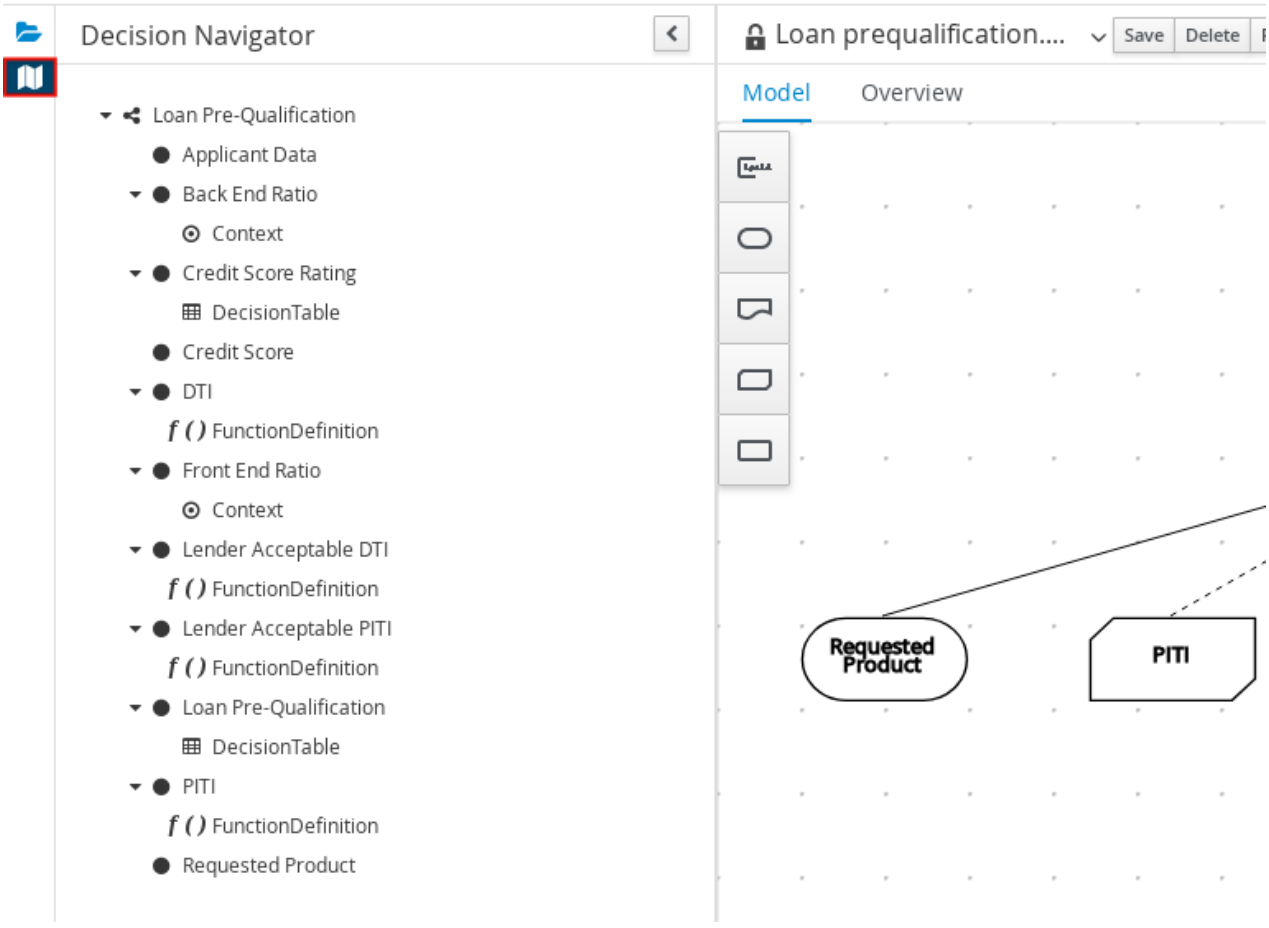
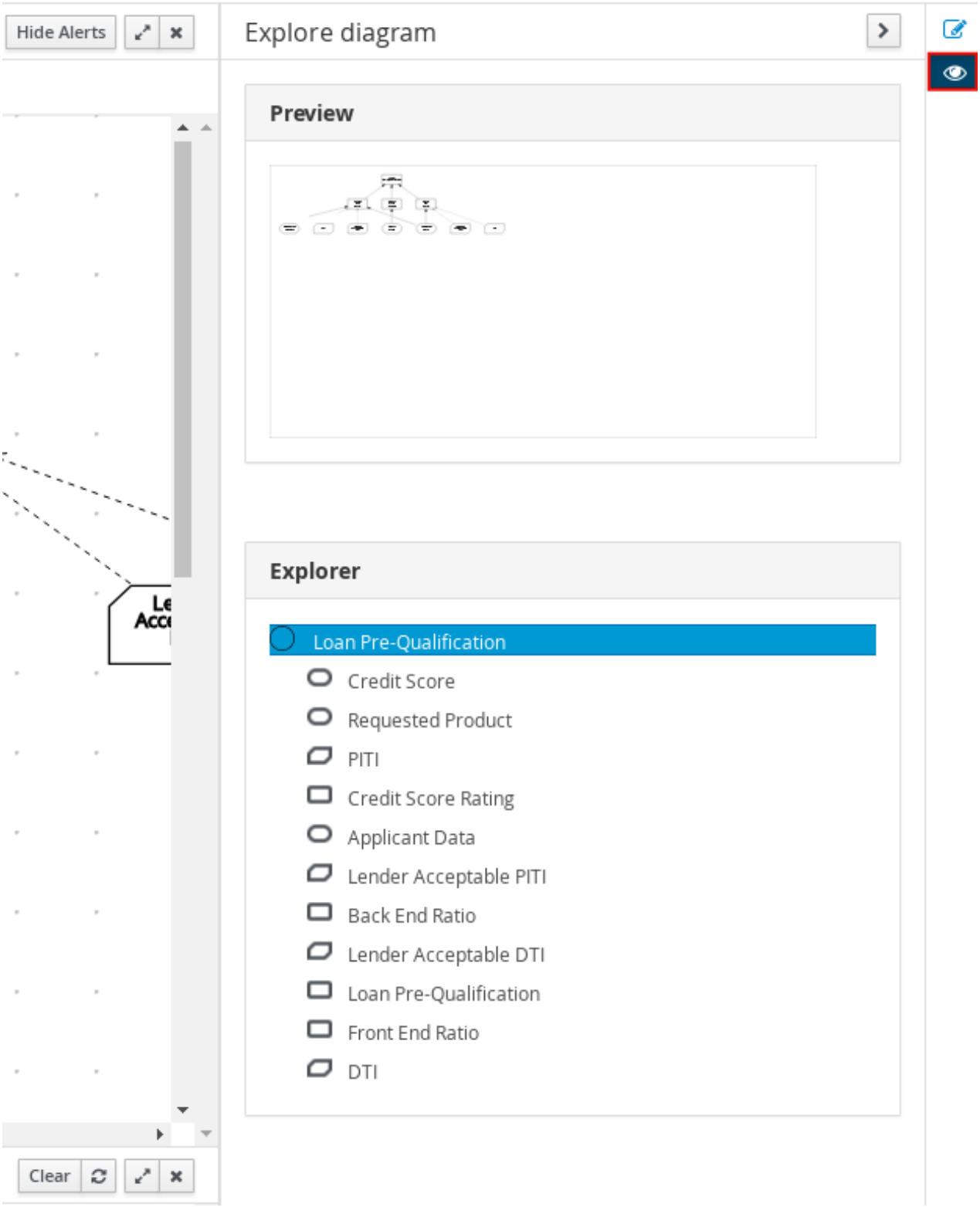


図3.19 Decision Navigator のビュー



DMN デザイナーの右上隅で、**Explore diagram** アイコンを選択して、選択した DRD で俯瞰プレビューを表示して、選択した DRD のノード間を移動します。

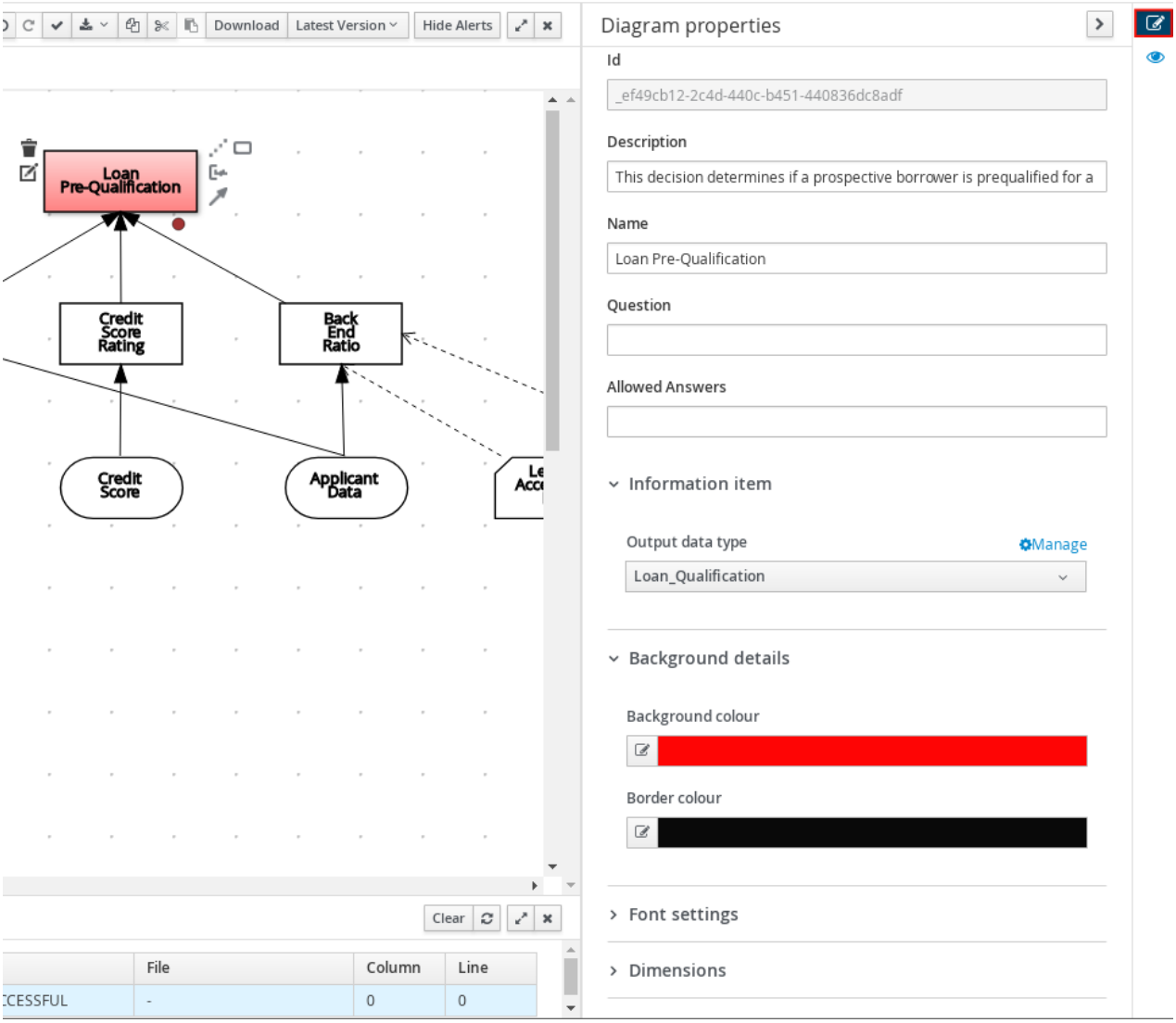
図3.20 Exploer Diagram ビュー



DRD プロパティーと設計

DMN デザイナーの右上隅で、**Diagram properties** アイコンを選択して、情報、データ型、選択した DRD、DRD ノード、ボックス式セルの外観を変更できます。

図3.21 DRD ノードのプロパティー



全 DRD のプロパティーを表示するには、特定のノードではなく、DRD キャンバスの背景をクリックします。

第4章 DMN モデルの実行

Decision Central を使用して Red Hat Decision Manager のプロジェクトに DMN ファイルを作成またはインポートするか、Decision Central を使用しないプロジェクトのナレッジ JAR (KJAR) ファイルの一部として DMN ファイルをパッケージ化できます。Red Hat Decision Manager プロジェクトに DMN ファイルに実装したあと、リモートアクセスの Business Server にそれを含む KIE コンテナーをデプロイするか、呼び出すアプリケーションの依存関係として KIE コンテナーを直接操作することで、DMN デシジョンサービスを実行できます。DMN ナレッジパッケージを作成しデプロイするその他のオプションも利用できますが、そのほとんどはナレッジアセットの全タイプ (DRL ファイルやプロセス定義など) と類似しています。

プロジェクトのパッケージングおよびデプロイメントの方法に外部 DMN アセットを含める方法は、**Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイ** を参照してください。

4.1. DMN コールの JAVA アプリケーションへの直接組み込み

KIE コンテナーは、呼び出しプログラムにナレッジアセットを直接組み込む場合や、KJAR 用 Maven 依存関係を使用して物理的にプルする場合は、ローカルとみなされます。コードのバージョンと、DML 定義のバージョンとの間に密接な関係がある場合は、通常、ナレッジアセットをプロジェクトに直接組み込みます。意思決定への変更は、アプリケーションを更新して再デプロイしないと有効になりません。このアプローチに対する利点は、適切なオペレーションがランタイムへの外部の依存関係に依存していないことですが、ロックされた環境の場合は制限になる可能性があります。

Maven の依存関係を使用すると、システムプロパティーを使用して、更新を定期的にスキャンして自動的に更新するなど、特定バージョンの意思決定が動的に変更するため、柔軟性が高まります。これにより、外部の依存関係がサービスのデプロイ時間に影響を及ぼしますが、意思決定はローカルで実行されるため、ランタイム時に利用可能な外部サービスに対する信頼が低くなります。

前提条件

- KIE コンテナーが、DMN モデルを含む KJAR の形式で、Decision Server でデプロイされている。実行可能モデルとしてコンパイルされていれば尚可。

  ```
  mvn clean install -DgenerateDMNModel=yes
  ```

  プロジェクトのパッケージ化およびデプロイメント、ならびに実行可能モデルに関する詳細は、**Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイ** を参照してください。

手順

1. クライアントアプリケーションで、Java プロジェクトの関連クラスパスに以下の依存関係を追加します。

```
<!-- Required for the DMN runtime API -->
<dependency>
  <groupId>org.kie</groupId>
  <artifactId>kie-dmn-core</artifactId>
  <version>${rhdm.version}</version>
</dependency>

<!-- Required if not using classpath KIE container -->
<dependency>
  <groupId>org.kie</groupId>
```

```
  <artifactId>kie-ci</artifactId>
  <version>${rhdm.version}</version>
</dependency>
```

<version> は、プロジェクトで現在使用する Red Hat Decision Manager の Maven アーティファクトバージョンです (例: 7.14.0.Final-redhat-00002)。

注記

個別の依存関係に対して Red Hat Decision Manager **<version>** を指定するのではなく、Red Hat Business Automation 部品表 (BOM) の依存関係をプロジェクトの **pom.xml** ファイルに追加することを検討してください。Red Hat Business Automation BOM は、Red Hat Decision Manager と Red Hat Process Automation Manager の両方に適用されます。BOM ファイルを追加すると、提供される Maven リポジトリーから、推移的依存関係の適切なバージョンがプロジェクトに含められます。

BOM 依存関係の例:

```
<dependency>
  <groupId>com.redhat.ba</groupId>
  <artifactId>ba-platform-bom</artifactId>
  <version>7.2.0.GA-redhat-00002</version>
  <scope>import</scope>
  <type>pom</type>
</dependency>
```

Red Hat Business Automation BOM (Bill of Materials) についての詳細情報は、What is the mapping between RHDM product and maven library version? を参照してください。

2. **classpath** または **ReleaseId** から KIE コンテナーを作成します。

```
KieServices kieServices = KieServices.Factory.get();

ReleaseId releaseId = kieServices.newReleaseId( "org.acme", "my-kjar", "1.0.0" );
KieContainer kieContainer = kieServices.newKieContainer( releaseId );
```

または、以下のオプションを使用します。

```
KieServices kieServices = KieServices.Factory.get();

KieContainer kieContainer = kieServices.getKieClasspathContainer();
```

3. **namespace** モデルおよび **modelName** モデルを使用して、KIE コンテナーの **DMNRuntime** と、評価する DMN モデルへの参照を取得します。

```
DMNRuntime dmnRuntime =
kieContainer.newKieSession().getKieRuntime(DMNRuntime.class);

String namespace = "http://www.redhat.com/_c7328033-c355-43cd-b616-0aceef80e52a";
String modelName = "dmn-movieticket-ageclassification";

DMNModel dmnModel = dmnRuntime.getModel(namespace, modelName);
```

4. 希望するモデルに対してデシジョンサービスを実行します。

```
DMNContext dmnContext = dmnRuntime.newContext(); (1)

for (Integer age : Arrays.asList(1,12,13,64,65,66)) {
   dmnContext.set("Age", age); (2)
   DMNResult dmnResult =
      dmnRuntime.evaluateAll(dmnModel, dmnContext); (3)

   for (DMNDecisionResult dr : dmnResult.getDecisionResults()) { (4)
      log.info("Age: " + age + ", " +
            "Decision: '" + dr.getDecisionName() + "', " +
            "Result: " + dr.getResult());
  }
}
```

(1) モデル評価に対する入力として使用する、新しい DMN コンテキストをインスタンス化します。この例では、Age Classification の意思決定を複数回ループさせています。

(2) 入力の DMN コンテキストに入力変数を割り当てます。

(3) DMN モデルに定義した DMN デシジョンをすべて評価します。

(4) 1 回の評価で結果が 1 つ以上になることがあり、ループを作成します。

この例では、以下の結果を出力します。

```
Age 1 Decision 'AgeClassification' : Child
Age 12 Decision 'AgeClassification' : Child
Age 13 Decision 'AgeClassification' : Adult
Age 64 Decision 'AgeClassification' : Adult
Age 65 Decision 'AgeClassification' : Senior
Age 66 Decision 'AgeClassification' : Senior
```

DMN モデルがより効率性の高い実行を行えるように、実行可能なモデルとしてこれまでにコンパイルされていない場合は、DMN モデルの実行時に、以下のプロパティーを有効化してください。

```
-Dorg.kie.dmn.compiler.execmodel=true
```

4.2. DECISION SERVER JAVA クライアント API を使った DMN サービスの実行

Decision Server Java クライアント API は、Decision Server の REST または JMS インターフェイスを通してリモートの DMN サービスを呼び出す軽量なアプローチを提供します。これにより、KIE ベースと相互に作用するのに必要なランタイムの依存関係の数が減ります。これを有効にして適切なペースで個別に相互作用するようにし、意思決定の定義から呼び出しコードを切り離すと、柔軟性が上がります。

Decision Server Java クライアント API の詳細は、**KIE API を使った Red Hat Decision Manager の操作**を参照してください。

前提条件

- Decision Server がインストールされ、設定されている (**kie-server** ロールが割り当てられているユーザーの既知のユーザー名と認証情報を含む)。インストールオプションは **Red Hat Decision Manager インストールの計画** を参照してください。
- KIE コンテナーが、DMN モデルを含む KJAR の形式で、Decision Server でデプロイされている。実行可能モデルとしてコンパイルされていれば尚可。

```
mvn clean install -DgenerateDMNModel=yes
```

プロジェクトのパッケージ化およびデプロイメント、ならびに実行可能モデルに関する詳細は、**Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイ** を参照してください。

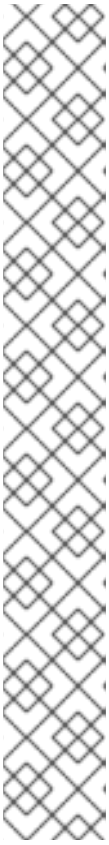
- KIE コンテナーのコンテナー ID に DMN モデルを含んでいる。1 つ以上のモデルが存在する場合は、そのモデルの名前空間およびモデル名が必要です。

手順

1. クライアントアプリケーションで、Java プロジェクトの関連クラスパスに以下の依存関係を追加します。

```
<dependency>
  <groupId>org.kie.server</groupId>
  <artifactId>kie-server-client</artifactId>
  <version>${rhdm.version}</version>
</dependency>
```

<version> は、プロジェクトで現在使用する Red Hat Decision Manager の Maven アーティファクトバージョンです (例: 7.14.0.Final-redhat-00002)。



注記

個別の依存関係に対して Red Hat Decision Manager **<version>** を指定するのではなく、Red Hat Business Automation 部品表 (BOM) の依存関係をプロジェクトの **pom.xml** ファイルに追加することを検討してください。Red Hat Business Automation BOM は、Red Hat Decision Manager と Red Hat Process Automation Manager の両方に適用されます。BOM ファイルを追加すると、提供される Maven リポジトリーから、推移的依存関係の適切なバージョンがプロジェクトに含められます。

BOM 依存関係の例:

```
<dependency>
  <groupId>com.redhat.ba</groupId>
  <artifactId>ba-platform-bom</artifactId>
  <version>7.2.0.GA-redhat-00002</version>
  <scope>import</scope>
  <type>pom</type>
</dependency>
```

Red Hat Business Automation BOM (Bill of Materials) についての詳細情報は、What is the mapping between RHDM product and maven library version? を参照してください。

2. 適切な接続情報で **KieServicesClient** をインスタンス化します。
以下に例を示します。

```
KieServicesConfiguration conf =
    KieServicesFactory.newRestConfiguration(URL, USER, PASSWORD); (1)

conf.setMarshallingFormat(MarshallingFormat.JSON); (2)

KieServicesClient kieServicesClient = KieServicesFactory.newKieServicesClient(conf);
```

(1) 接続情報:

- サンプル URL: **http://localhost:8080/kie-server/services/rest/server**
- この認証情報は、**kie-server** ロールを持つユーザーを参照します。

(2) マーシャリングの形式は、**org.kie.server.api.marshalling.MarshallingFormat** のインスタンスです。これは、メッセージが JSON であるか XML であるかを制御します。マーシャリング形式のオプションは、JSON、JAXB、 XSTREAM です。

3. KIE サーバーの Java クライアントインスタンスで **getServicesClient()** メソッドを呼び出すことで、関連する Decision Server に接続した KIE サーバーの Java クライアントから **DMNServicesClient** を取得します。

```
DMNServicesClient dmnClient =
kieServicesClient.getServicesClient(DMNServicesClient.class );
```

これで、**dmnClient** が、Decision Server でデシジョンサービスを実行できるようになりました。

4. 希望するモデルに対してデシジョンサービスを実行します。
以下に例を示します。

```
for (Integer age : Arrays.asList(1,12,13,64,65,66)) {
    DMNContext dmnContext = dmnClient.newContext(); (1)
    dmnContext.set("Age", age); (2)
    ServiceResponse<DMNResult> serverResp = (3)
        dmnClient.evaluateAll($kieContainerId,
                              $modelNamespace,
                              $modelName,
                              dmnContext);

    DMNResult dmnResult = serverResp.getResult(); (4)
    for (DMNDecisionResult dr : dmnResult.getDecisionResults()) {
        log.info("Age: " + age + ", " +
                 "Decision: '" + dr.getDecisionName() + "', " +
                 "Result: " + dr.getResult());
    }
}
```

(1) モデル評価に対する入力として使用する、新しい DMN コンテキストをインスタンス化します。この例では、Age Classification の意思決定を複数回ループさせています。

2 入力 DMN コンテキストに入力変数を割り当てます。

3 DMN モデルに定義したすべての DMN の意思決定を評価します。

- **$kieContainerId** は、DMN モデルを含む KJAR がデプロイされているコンテナーの ID です。
- **$modelNamespace** は、モデルの名前空間です。
- **$modelName** は、モデルの名前です。

4 DMN の結果オブジェクトは、サーバーの応答から利用できます。

この時点では、**dmnResult** には、評価した DMN モデルから得た意思決定の結果がすべて含まれます。

DMNServicesClient で利用可能なメソッドを使用して、モデルで特定の DMN 意思決定だけを実行することもできます。

注記

KIE コンテナーに DMN モデルが 1 つだけ含まれる場合は、Decision Server API によってデフォルトで選択されるため、**$modelNamespace** と **$modelName** を除外できます。

4.3. DECISION SERVER REST API を使った DMN サービスの実行

Decision Server の REST エンドポイントで直接対話することで、呼び出しコードと、意思決定ロジックの定義の分離が最大になります。呼び出しコードに直接の依存関係がないため、**node.js**、**.net** など、完全に異なる開発プラットフォームに実装できます。このセクションの例では、Nix スタイルの curl コマンドを示しますが、REST クライアントに適用するための関連情報を提供します。

Decision Server REST API についての詳細は、**KIE API を使った Red Hat Decision Manager の操作**を参照してください。

前提条件

- Decision Server がインストールされ、設定されている (**kie-server** ロールが割り当てられているユーザーの既知のユーザー名と認証情報を含む)。インストールオプションは **Red Hat Decision Manager インストールの計画** を参照してください。
- KIE コンテナーが、DMN モデルを含む KJAR の形式で、Decision Server でデプロイされている。実行可能モデルとしてコンパイルされていれば尚可。

```
mvn clean install -DgenerateDMNModel=yes
```

プロジェクトのパッケージ化およびデプロイメント、ならびに実行可能モデルに関する詳細は、**Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイ** を参照してください。

- KIE コンテナーのコンテナー ID に DMN モデルを含んでいる。1 つ以上のモデルが存在する場合は、そのモデルの名前空間およびモデル名が必要です。

手順

1. Decision Server REST API エンドポイントにアクセスするためのベース URL を決定します。これには、以下の値が必要です (例ではローカルデプロイメントのデフォルト値を使用しています)。

   - ホスト (**localhost**)
   - ポート (**8080**)
   - ルートコンテキスト (**kie-server**)
   - ベース REST パス (**services/rest/**)

   ローカルデプロイメントにおけるベース URL の例:

   http://localhost:8080/kie-server/services/rest/

2. ユーザー認証要件を決定します。
   ユーザーを Decision Server 設定に直接定義すると、ユーザー名およびパスワードを要求する HTTP Basic 認証 が使用されます。要求を成功させるには、ユーザーに **kie-server** ルールが必要です。

   以下の例は、curl 要求に認証情報を追加する方法を示します。

   ```
   curl -u username:password <request>
   ```

   Red Hat シングルサインオンを使用して Decision Server を設定している場合は、要求にベアラートークンが必要です。

   ```
   curl -H "Authorization: bearer $TOKEN" <request>
   ```

3. 要求と応答の形式を指定します。REST API エンドポイントには JSON と XML の両方の書式が利用でき、要求ヘッダーを使用して設定されます。

   JSON

   ```
   curl -H "accept: application/json" -H "content-type: application/json"
   ```

   XML

   ```
   curl -H "accept: application/xml" -H "content-type: application/xml"
   ```

4. (任意) デプロイしたデシジョンモデルのリストに対するコンテナーのクエリーです。
   [GET] server/containers/{containerId}/dmn

   curl 要求例:

   ```
   curl -u krisv:krisv -H "accept: application/xml" -X GET "http://localhost:8080/kie-server/services/rest/server/containers/MovieDMNContainer/dmn"
   ```

   サンプルの XML 出力:

   ```
   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <response type="SUCCESS" msg="OK models successfully retrieved from container 'MovieDMNContainer'">
   ```

```
    <dmn-model-info-list>
      <model>
        <model-namespace>http://www.redhat.com/_c7328033-c355-43cd-b616-
0aceef80e52a</model-namespace>
        <model-name>dmn-movieticket-ageclassification</model-name>
        <model-id>_99</model-id>
        <decisions>
          <dmn-decision-info>
            <decision-id>_3</decision-id>
            <decision-name>AgeClassification</decision-name>
          </dmn-decision-info>
        </decisions>
      </model>
    </dmn-model-info-list>
</response>
```

サンプルの JSON 出力:

```
{
  "type" : "SUCCESS",
  "msg" : "OK models successfully retrieved from container 'MovieDMNContainer'",
  "result" : {
    "dmn-model-info-list" : {
      "models" : [ {
        "model-namespace" : "http://www.redhat.com/_c7328033-c355-43cd-b616-
0aceef80e52a",
        "model-name" : "dmn-movieticket-ageclassification",
        "model-id" : "_99",
        "decisions" : [ {
          "decision-id" : "_3",
          "decision-name" : "AgeClassification"
        } ]
      } ]
    }
  }
}
```

5. モデルを実行します。
[POST] server/containers/{containerId}/dmn

curl 要求例:

```
curl -u krisv:krisv -H "accept: application/json" -H "content-type: application/json" -X POST
"http://localhost:8080/kie-server/services/rest/server/containers/MovieDMNContainer/dmn" -d
"{ \"model-namespace\" : \"http://www.redhat.com/_c7328033-c355-43cd-b616-
0aceef80e52a\", \"model-name\" : \"dmn-movieticket-ageclassification\", \"decision-name\" : [
], \"decision-id\" : [ ], \"dmn-context\" : {\"Age\" : 66}}"
```

JSON 要求例:

```
{
  "model-namespace" : "http://www.redhat.com/_c7328033-c355-43cd-b616-0aceef80e52a",
  "model-name" : "dmn-movieticket-ageclassification",
  "decision-name" : [ ],
```

```
  "decision-id" : [ ],
  "dmn-context" : {"Age" : 66}
}
```

XML 要求例 (JAXB 形式):

```
<?xml version="1.0" encoding="UTF-8"?>
<dmn-evaluation-context>
    <model-namespace>http://www.redhat.com/_c7328033-c355-43cd-b616-
0aceef80e52a</model-namespace>
    <model-name>dmn-movieticket-ageclassification</model-name>
    <dmn-context xsi:type="jaxbListWrapper" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
        <type>MAP</type>
        <element xsi:type="jaxbStringObjectPair" key="Age">
            <value xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema">66</value>
        </element>
    </dmn-context>
</dmn-evaluation-context>
```

注記

要求には、その形式にかかわらず、以下の要素が必要です。

- モデルの名前空間
- モデル名
- 入力値を含むコンテキストオブジェクト

JSON 応答例:

```
{
  "type" : "SUCCESS",
  "msg" : "OK from container 'MovieDMNContainer'",
  "result" : {
    "dmn-evaluation-result" : {
      "messages" : [ ],
      "model-namespace" : "http://www.redhat.com/_c7328033-c355-43cd-b616-
0aceef80e52a",
      "model-name" : "dmn-movieticket-ageclassification",
      "decision-name" : [ ],
      "dmn-context" : {
        "Age" : 66,
        "AgeClassification" : "Senior"
      },
      "decision-results" : {
        "_3" : {
          "messages" : [ ],
          "decision-id" : "_3",
          "decision-name" : "AgeClassification",
          "result" : "Senior",
          "status" : "SUCCEEDED"
        }
      }
```

```
    }
  }
}
```

XML (JAXB 形式) 応答例:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response type="SUCCESS" msg="OK from container 'MovieDMNContainer'">
    <dmn-evaluation-result>
        <model-namespace>http://www.redhat.com/_c7328033-c355-43cd-b616-
0aceef80e52a</model-namespace>
        <model-name>dmn-movieticket-ageclassification</model-name>
        <dmn-context xsi:type="jaxbListWrapper"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <type>MAP</type>
            <element xsi:type="jaxbStringObjectPair" key="Age">
                <value xsi:type="xs:int"
xmlns:xs="http://www.w3.org/2001/XMLSchema">66</value>
            </element>
            <element xsi:type="jaxbStringObjectPair" key="AgeClassification">
                <value xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema">Senior</value>
            </element>
        </dmn-context>
        <messages/>
        <decisionResults>
            <entry>
                <key>_3</key>
                <value>
                    <decision-id>_3</decision-id>
                    <decision-name>AgeClassification</decision-name>
                    <result xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Senior</result>
                    <messages/>
                    <status>SUCCEEDED</status>
                </value>
            </entry>
        </decisionResults>
    </dmn-evaluation-result>
</response>
```

第5章 関連情報

- Decision Model and Notation specification
- DMN Technology Compatibility Kit
- **Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイ**

付録A バージョン情報

本書の最終更新日: 2021 年 11 月 15 日 (月)