



Red Hat Decision Manager 7.0

DMN モデルを使用したデシジョンサービスの作成

Red Hat Decision Manager 7.0 DMN モデルを使用したデシジョンサービスの作成

Red Hat Customer Content Services
brms-docs@redhat.com

法律上の通知

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書は、Red Hat Decision Manager 7.0 で DMN (Decision Model and Notation) モデルを使用して デシジョンサービスを作成する方法を説明します。

目次

前書き	3
第1章 DMN 要素	4
1.1. FEEL を使用したルール表現	4
1.2. デシジョンテーブル	5
1.2.1. ヒットポリシー	5
1.3. ボックス式	6
第2章 DMN ユースケース	7
第3章 DMN モデルの例	10
第4章 DMN モデルの呼び出しオプション	14
4.1. DMN コールを JAVA アプリケーションに直接組み込み	14
4.2. DECISION SERVER (JAVA) で DMN サービスをリモートに実行	16
4.3. REST API を使用してリモートサーバーで DMN サービスの呼び出し	18
第5章 関連情報	23
付録A バージョン情報	24

前書き

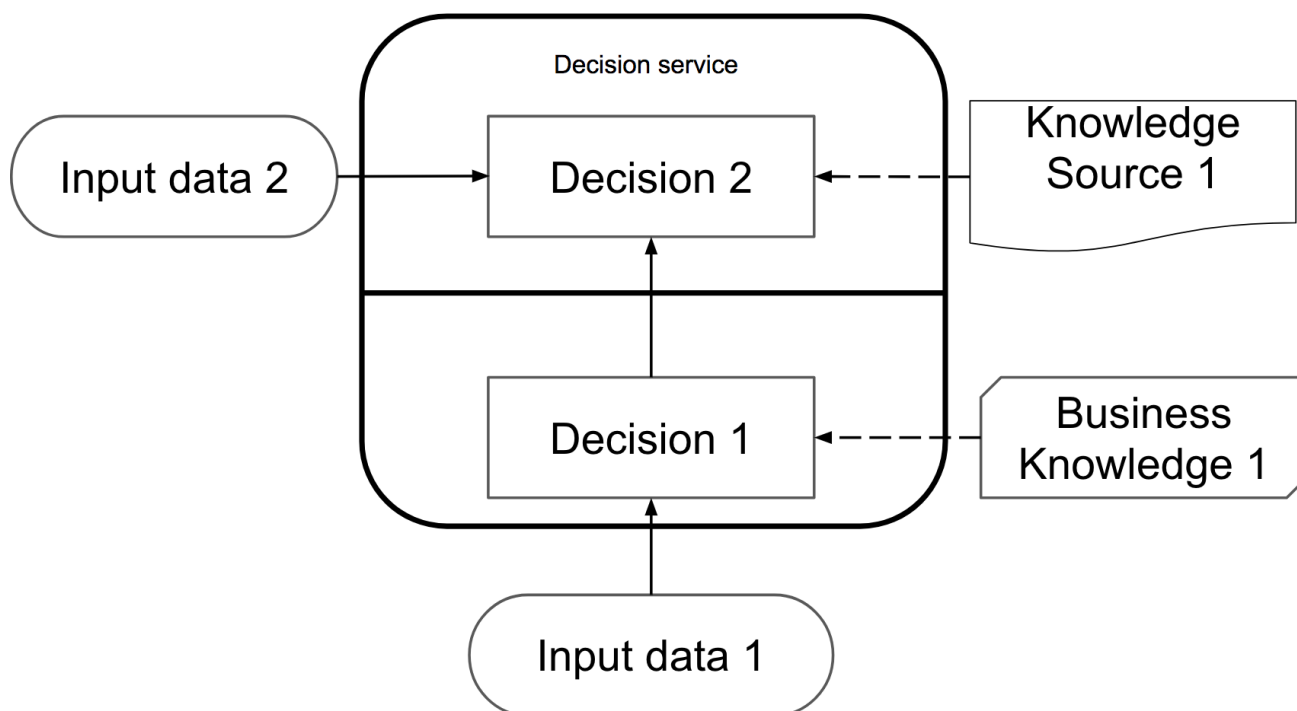
ビジネスアナリストやルール作成者は、DMN (Decision Model and Notation) を使用して、DRD (decision requirements diagram) でデシジョンサービスを視覚的にモデル化できます。このダイアグラムは、デシジョンテーブルなど、DMN モデルの意思決定要素のロジックを描く各決定ノードを使用して、経営上の意思決定を初めから終わりまで明示します。

第1章 DMN 要素

DMN モデルは、以下の 5 つの要素で構成されます。

- **意思決定:** 意思決定ロジックに基づいて、1 つ以上の入力が出力を決定するモデルのノード。
- **入力データ:** 意思決定に必要な情報。この情報には、通常、レストランのピークの営業時間や、スタッフのアベイラビリティなど、ビジネスレベルの概念やビジネス関連のオブジェクトが含まれます。
- **ビジネスナレッジモデル:** 再利用可能な意思決定ロジック。意思決定が使用するロジックは同じですが、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:** 意思決定ロジックを具体化するなどの外部規則、ドキュメント、コミティー、ポリシーなど。ナレッジソースは、実行可能なビジネスルールというより、現実世界の要因への参照となります。
- **デシジョンサービス:** デシジョンサービスは、入力が十分に定義された、トップレベルの決定で、呼び出しサービスとして公開されます。このダイアグラムでは、角が丸いオーバーレイの長方形になります。デシジョンサービスは、外部アプリケーションまたはビジネスプロセス (BPMN) から呼び出せます。詳細は、DMN 仕様ドキュメントの 36 ページを参照してください。

図1.1 意思決定の基本要件のダイアグラム



1.1. FEEL を使用したルール表現

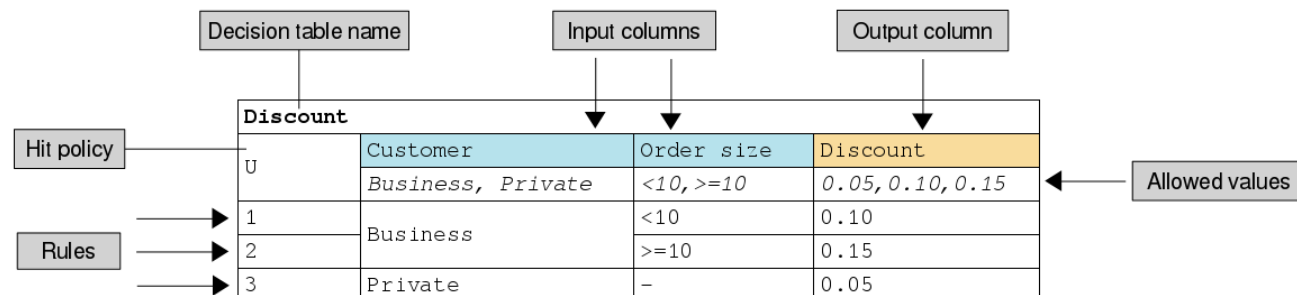
FEEL (Friendly Enough Expression Language) は、DMN 仕様に定義された新しい式言語です。これは、デシジョンモデルの構成にセマンティクスを割り当てて、意思決定のモデリングと、実行のギャップを埋めることを目的としています。DRD (decision requirements diagrams) の FEEL 式は、デシジョンテーブルまたはデシジョンノードのいずれかのテーブルセルに記述されます。FEEL 式は、デシジョンの論理を定義します。

DMN の FEEL の詳細は「[OMG DMN 仕様](#)」を参照してください。

1.2. デシジョンテーブル

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。テーブルの各行が 1 つのルールになり、条件を定義する列と、その特定行に対する結果が含まれます。各行では、結果を正確に取得できるように条件の値を定義します。読みやすくするために、テーブルを表示する際に、より技術的な詳細を一部非表示にします。

図1.2 決定テーブルのサンプル



デシジョンテーブルは、ルールと決定のモデル化を行うのによく使われる方法で、(DMN などの) 多くの方法論で使用され、フレームワークを実装します。



重要

DMN と Drools では、デシジョンテーブルの概要が似ていますが、DMN デシジョンテーブルの構文とレイアウトは DMN 仕様によって定義され、Drools ネイティブのデシジョンテーブルは Drools プロジェクトによって定義されます。Red Hat Decision Manager は、デシジョンテーブルの両方をサポートしますが、取り換えることはできません。Drools デシジョンテーブルの詳細は『[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作成](#)』を参照してください。

1.2.1. ヒットポリシー

ヒットポリシーは、複数のルールが 1 つのデシジョンテーブルに一致したときに、どのように結果に到達するかを定義します。結果に到達するためのポリシーを 5 つの中から 1 つ選択し、テーブルの右上にインジケータを配置してポリシーを指定します。以下で、インジケータのタイプの後に表記される括弧の中身が、インジケータとなります。

- **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 リストで最初にくる出力が選択されます。
- **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。

1.3. ボックス式

ボックス式は、DMN モデルについて、コンテキスト、関数の定義、関数の呼び出し、その他の式で構成される表形式です。たとえば、以下のボックス式は、4 つのパラメーター(**Product**、**Rate**、**Term**、および **Amount**) を使用して、毎月の分割払いを計算する **Installment calculation** 関数を定義します。

図1.3 ボックス式の例

Installment calculation	
(Product, Rate, Term, Amount)	
Monthly Fee	if Product Type = "standard loan" then 30.00 else if Product Type = "special loan" then 20.00 else null
Monthly Repayment	PMT(Rate, Term, Amount)
Monthly Repayment + Monthly Fee	

第2章 DMN ユースケース

この DMN 例では、実際の例にデシジョンモデリングを使用して、入力、状況、企業のガイドラインに基づいてどのように意思決定が行われるかを示しています。このセクションのプロセスは、このコンポーネントの一部がどのように連携するかを示します。このシナリオでは、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、その影響を受ける航空会社が、フライトがキャンセルされた乗客に代替の手配を行います。

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

- フライトリスト
- 乗客リスト

意思決定

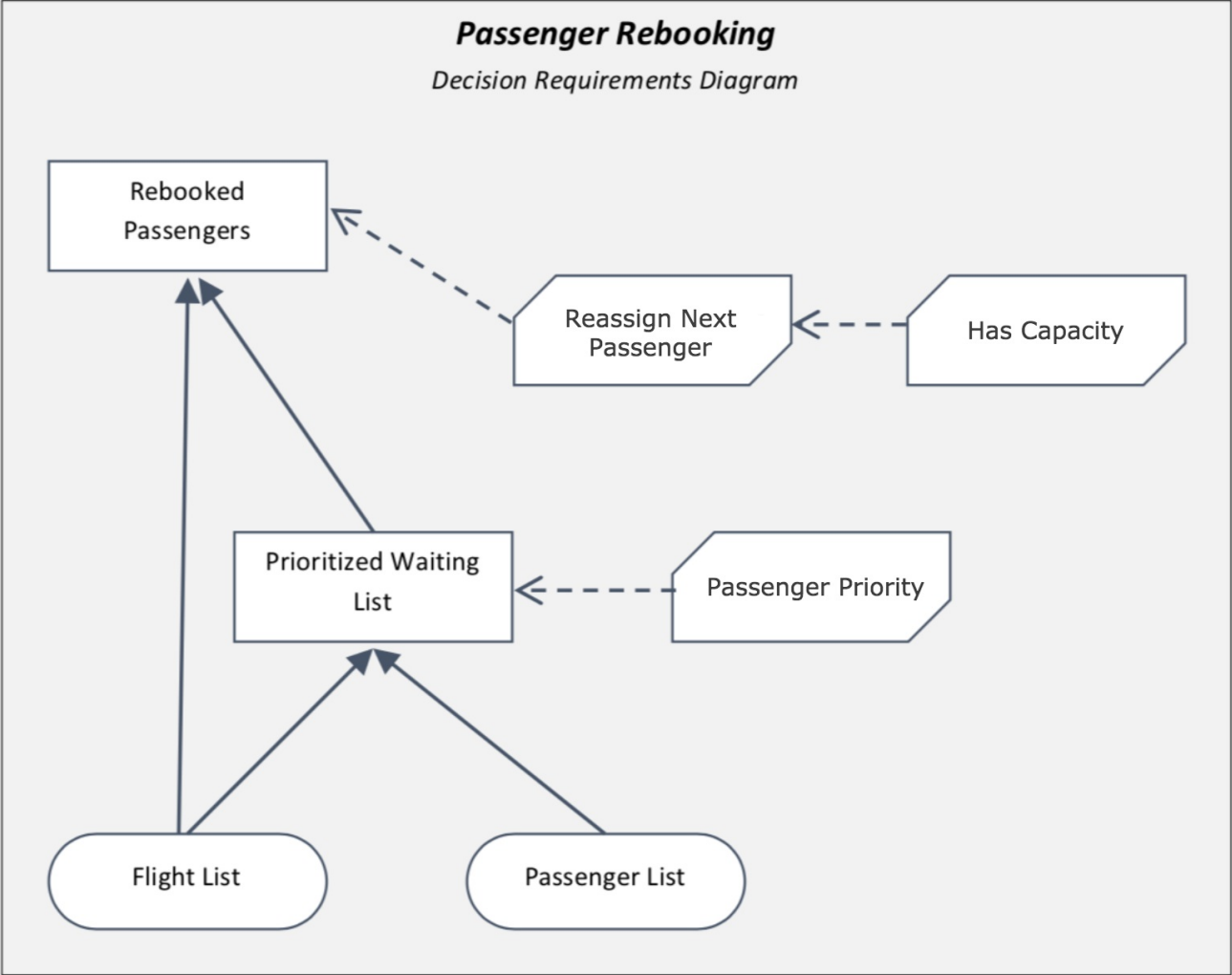
- 新しいフライトで席を確保する乗客の優先順位をつける
- 乗客に提示するフライトを決定する

ビジネスナレッジ

- 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、DRD (decision requirements diagram) 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2.1 乗客の予約変更例における意思決定要件ダイアグラム



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、端が切り取られた長方形には、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。

さらに、DRD は、各要素に対する詳細をボックスに追加し、再度 FEEL 式を使用して変数を定義します。式は、ウェイティングリストの優先順位を確立する航空会社のデシジョンプロセスのように、簡単なものもあります。

図2.2 ウェイティングリストの優先順位に関するボックス式のサンプル

Prioritized Waiting List	
Cancelled Flights	Flight List[Status = "cancelled"].Flight Number
Waiting List	Passenger List[list contains(Cancelled Flights, Flight Number)]
sort(Waiting List, passenger priority)	

要素によっては、詳細や計算が非常に細くなる場合があります。次の乗客を再度割り当てる以下のビジネスナレッジモデルについて考えてみましょう。

図2.3 次の乗客を再割り当てする決定例

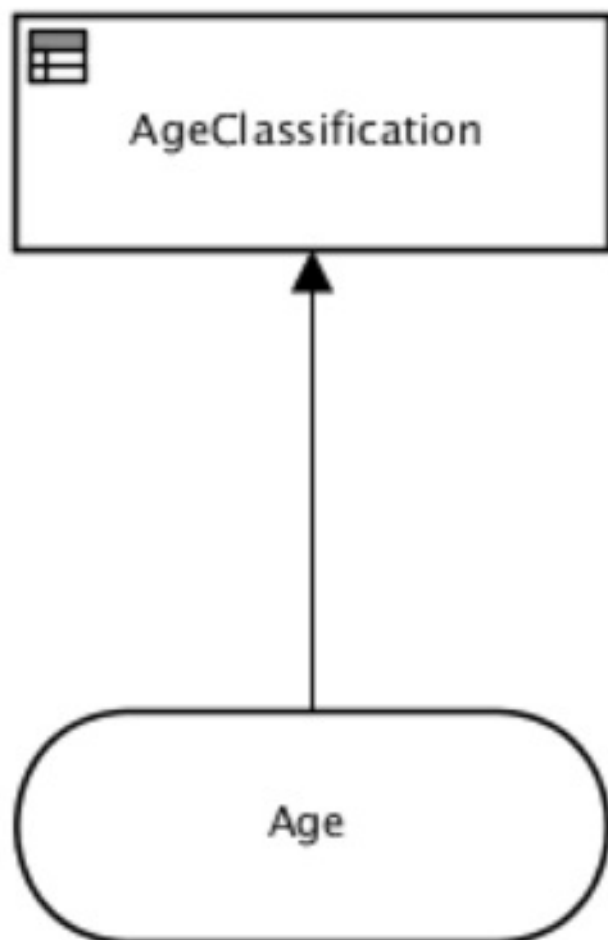
reassign next passenger									
(Waiting List, Reassigned Passengers List, Flights)									
Next Passenger	Waiting List[1]								
Original Flight	Flights[Flight Number = Next Passenger.Flight Number][1]								
Best Alternate Flight	Flights[From = Original Flight.From and To = Original Flight.To and Departure > Original Flight.Departure and Status = "scheduled" and has capacity(item, Reassigned Passengers List)][1]								
Reassigned Passenger	<table><tr><td>Name</td><td>Next Passenger.Name</td></tr><tr><td>Status</td><td>Next Passenger.Status</td></tr><tr><td>Miles</td><td>Next Passenger.Miles</td></tr><tr><td>Flight Number</td><td>Best Alternate Flight.Flight Number</td></tr></table>	Name	Next Passenger.Name	Status	Next Passenger.Status	Miles	Next Passenger.Miles	Flight Number	Best Alternate Flight.Flight Number
Name	Next Passenger.Name								
Status	Next Passenger.Status								
Miles	Next Passenger.Miles								
Flight Number	Best Alternate Flight.Flight Number								
Remaining Waiting List	remove(Waiting List, 1)								
Updated Reassigned Passenger List	append(Reassigned Passengers List, Reassigned Passenger)								
<pre>if count(Remaining Waiting List) > 0 then reassign next passenger(Remaining Waiting List, Updated Reassigned Passengers List, Flights) else Updated Reassigned Passengers List</pre>									

第3章 DMN モデルの例

DMN は、異なる DMN 作成プラットフォーム間で使用される DMN モデルを有効にする XML スキーマを定義します。DMN 仕様は、作成、テスト、および実稼働環境での実行に対して、同じファイルと連携するソフトウェアプラットフォームを複数有効にします。視覚的な作成機能が必要な場合は、Trisotech や Signavio など、サードパーティーの作成プラットフォームを使用する必要があります。

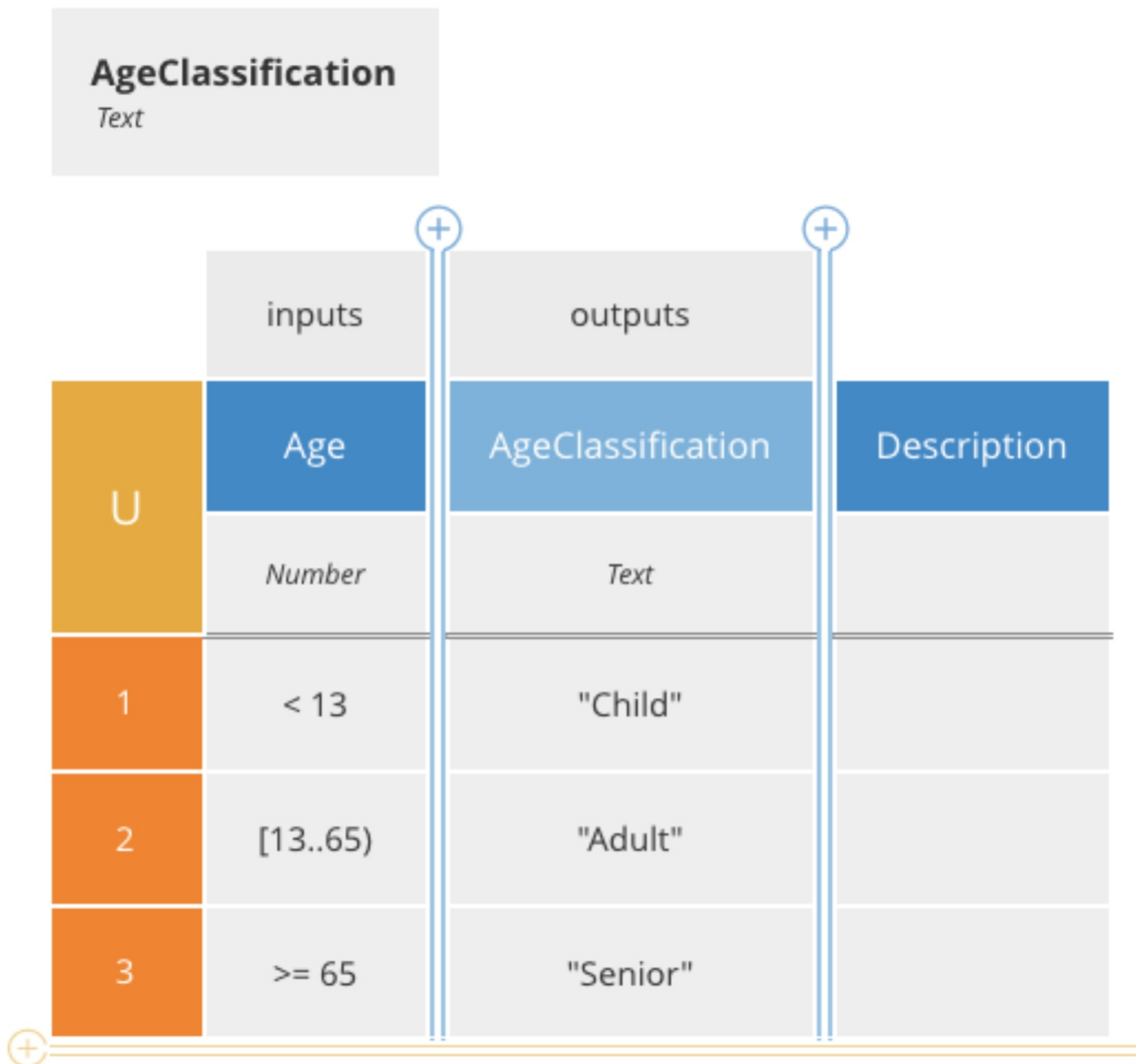
以下の意思決定の要件ダイアグラムは、映画チケットの購入時における、年齢を対象にした分類タイプの決定を示します。この基本例では、この簡単な決定を別のデシジョンの入力にし、計算が繰り返されるのを避ける分類を作成する方法を示します。

図3.1 年齢の分類決定に対する意思決定要件ダイアグラム



この例では、構成する数値の入力値が1つ (**Age**) だけとなり、正しい文字列出力を生成します (**AgeClassification**)。 **AgeClassification** の意思決定の内部機能が基本的なテーブルになります。

図3.2 年齢の分類決定に対するデシジョンテーブル



このテーブルは、年齢の範囲を決定する簡単な FEEL 式を使用して、値を **AgeClassification** 出力値に割り当てます。このデシジョンモデルは、Trisotech DMN Authoring 環境で作成されました。

以下の出力は、このデシジョンモデルの XML ソースとなります。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<semantic:definitions
  xmlns:semantic="http://www.omg.org/spec/DMN/20151101/dmn.xsd"
  xmlns:feel="http://www.omg.org/spec/FEEL/20140401"

  xmlns:tc="http://www.omg.org/spec/DMN/20160719/testcase"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"

  namespace="http://www.redhat.com/_c7328033-c355-
43cd-b616-0aceef80e52a" ①
  name="dmn-movieticket-ageclassification" ②
  id="_99">
  <semantic:extensionElements/>
```

```

<semantic:inputData displayName="Age" id="_1" name="Age">
  <semantic:variable id="_2" name="Age" typeRef="feel:number"/>
</semantic:inputData>
<semantic:decision displayName="AgeClassification" id="_3"
name="AgeClassification">
  <semantic:variable id="_4" name="AgeClassification"
typeRef="feel:string"/>
  <semantic:informationRequirement>
    <semantic:requiredInput href="#_1"/>
  </semantic:informationRequirement>
  <semantic:decisionTable hitPolicy="UNIQUE" id="_5"
outputLabel="AgeClassification">
    <semantic:input id="_6">
      <semantic:inputExpression typeRef="feel:number">
        <semantic:text>Age</semantic:text>
      </semantic:inputExpression>
    </semantic:input>
    <semantic:output id="_7"/>
    <semantic:rule id="_8">
      <semantic:inputEntry id="_9">
        <semantic:text>&lt; 13</semantic:text>
      </semantic:inputEntry>
      <semantic:outputEntry id="_10">
        <semantic:text>"Child"</semantic:text>
      </semantic:outputEntry>
    </semantic:rule>
    <semantic:rule id="_11">
      <semantic:inputEntry id="_12">
        <semantic:text>[13..65)</semantic:text>
      </semantic:inputEntry>
      <semantic:outputEntry id="_13">
        <semantic:text>"Adult"</semantic:text>
      </semantic:outputEntry>
    </semantic:rule>
    <semantic:rule id="_14">
      <semantic:inputEntry id="_15">
        <semantic:text>&gt;= 65</semantic:text>
      </semantic:inputEntry>
      <semantic:outputEntry id="_16">
        <semantic:text>"Senior"</semantic:text>
      </semantic:outputEntry>
    </semantic:rule>
  </semantic:decisionTable>
</semantic:decision>
</semantic:definitions>

```

❶ モデルの名前空間

❷ モデル名

この基本ファイルは、ビジネスロジック、および意思決定全体の入力と出力を入れるのに十分な情報と、一貫して関係を視覚的に表示するためのソフトウェアツールを有効にするのに十分な詳細を取得します。

ルートの **definitions** タグの **namespace** 属性および **name** 属性は、このデシジョンモデルを一意に識別します。多くの XML のように、**namespace** 値は、ドキュメントを作成する組織または個人に関連付けられた一意の URL として表示されます。

第4章 DMN モデルの呼び出しオプション

DMN ファイルに定義したデシジョンを呼び出すには、最初にファイルを KIE コンテナにパッケージ化する必要があります。ナレッジコンポーネントの特定のバージョンは、リモートアクセス用に Decision Server にデプロイするか、呼び出しアプリケーションの依存関係として直接操作できます。このドキュメントでは、このナレッジパッケージを作成およびデプロイするためのオプションをすべて説明しているわけではありませんが、多くの場合は (Drools ルールファイルまたは JBPM プロセス定義などの) その他のアセットに類似します。デシジョンをパッケージ化するか、デプロイしてから、それを呼び出すオプションはいくつかあります。

4.1. DMN コールを JAVA アプリケーションに直接組み込み

KIE コンテナは、ナレッジアセットを呼び出しプログラムに直接組み込む際、または KJAR 用 Maven 依存関係を使用して物理的にプルする際にローカルになります。コードのバージョンと、DMN 定義のバージョンとの間に密接な関係がある場合に、ナレッジアセットをプロジェクトに直接組み込む必要があります。意思決定への変更は、アプリケーションをアップデート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潜在的な利点の 1 つは、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おらず、ロックされた環境への制限になるということです。

Maven の依存関係を使用すると、特定バージョンの意思決定が動的に変更するため、システムプロパティを使用したり、アップデートを定期的にスキャンして自動的にアップデートするなどの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

- 実行する DMN モデルを含む KJAR が作成されている。
- 以下の依存関係が、プロジェクトの **pom.xml** ファイルに追加されている。

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

<!-- Required if not using classpath kie container -->
<dependency>
  <groupId>org.kie</groupId>
  <artifactId>kie-ci</artifactId>
  <version>${drools-version}</version>
</dependency>
```



注記

\${drools-version} は Maven プロパティで、ランタイム時に、その他の KIE / Drools 依存関係に使用されている正確なバージョンを決定する必要があります。

手順

1. **classpath** または **ReleaseId** から KIE コンテナを作成します。

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

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

代替オプション:

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

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

2. **namespace** モデルおよび **modelName** モデルを使用して、KIE コンテナの **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);
```

3. 希望のモデルでデシジョンサービスを実行します。

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

for (Integer age : Arrays.asList(1,12,13,64,65,66)) {
    dmnContext.set("Age", age); ❷
    DMNResult dmnResult = dmnRuntime.evaluateAll(dmnModel,
dmnContext); ❸
    for (DMNDecisionResult dr : dmnResult.getDecisionResults()) { ❹
        log.info("Age " + age + " Decision '" + dr.getDecisionName() +
"'" : " + dr.getResult());
    }
}
```

- ❶ 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。
- ❷ 入力の DMN コンテキストに対して入力変数を割り当てます。
- ❸ DMN モデルに定義した DMN デシジョンをすべて評価します。
- ❹ 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

4.2. DECISION SERVER (JAVA) で DMN サービスをリモートに実行

KIE のリモート API クライアントは、Decision Server の REST または JMS インターフェー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、ナレッジ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。

前提条件

- Decision Server がインストールされ設定されている (**kie-server** ロールを持つユーザーの、既知のユーザー名と認証情報を含む)。
- KIE コンテナが、DMN モデルを含む KJAR の形式で、Decision Server でデプロイされている。
- KIE コンテナのコンテナ 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。
- 以下の依存関係が、プロジェクトの **pom.xml** ファイルに追加されている。

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



注記

\${drools-version} は Maven プロパティで、ランタイム時に、その他の KIE / Drools 依存関係に使用されている正確なバージョンを決定する必要があります。

手順

1. 適切な接続情報で **KieServicesClient** インスタンスをインスタンス化します。
例:

```
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。

2. 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。

3. 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。

例:

```
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() + "'
: " + 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 コンテキストに対して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。



注記

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。

ヒント

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

4.3. REST API を使用してリモートサーバーで DMN サービスの呼び出し

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。

前提条件

- Decision Server がインストールされ、設定されている (サービスユーザーアカウントのアクセスが許可されている)。
- KIE コンテナが、DMN モデルを含む KJAR の形式で、Decision Server でデプロイされている。
- KIE コンテナのコンテナ 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. REST 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 (例ではローカルのデプロイメン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/server**)

ローカルデプロイメントのサンプル URL:

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

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 BasicAuth 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を含む必要があります。

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

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

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のクエリーです。

[GET] /containers/CONTAINER_ID/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] /containers/CONTAINER_ID/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 スタイル):

```

<?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

デシジョンサービスのパッケージングおよびデプロイメント

付録A バージョン情報

本ドキュメントの最終更新日: 2018 年 7 月 3 日