**Lab Guide**

IBM Decision Manger Open Edition

Nigel Crowther – ncrowther@uk.ibm.com

Hands-on Guide

DMN Beyond the Basics



NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive, MD-NC119

Armonk, NY 10504-1785

United States of America

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

**TRADEMARKS**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

© Copyright International Business Machines Corporation 2020.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

**Table of Contents**

[1 Introduction 5](#_Toc131237393)

[2 Prerequisites 6](#_Toc131237394)

[Lab 1 - Data Types 7](#_Toc131237395)

[Introduction 7](#_Toc131237396)

[Instructions 8](#_Toc131237397)

[Conclusion 12](#_Toc131237398)

[Lab 2 – The Divide and Conquer Pattern 13](#_Toc131237399)

[Introduction 13](#_Toc131237400)

[Instructions 13](#_Toc131237401)

[Conclusion 18](#_Toc131237402)

[Lab 3 – The Tiered Service Pattern 19](#_Toc131237403)

[Introduction 19](#_Toc131237404)

[Instructions 21](#_Toc131237405)

[Conclusion 24](#_Toc131237406)

[Lab 4 – The Index Pattern 25](#_Toc131237407)

[Introduction 25](#_Toc131237408)

[Instructions 27](#_Toc131237409)

[Conclusion 27](#_Toc131237410)

[Lab 5 - Hit Policies for Decision Tables 29](#_Toc131237411)

[Introduction 29](#_Toc131237412)

[Instructions 31](#_Toc131237413)

[Unique Policy 31](#_Toc131237414)

[First Policy 31](#_Toc131237415)

[String Collection policy 31](#_Toc131237416)

[Numeric Collection policy 31](#_Toc131237417)

[Lab 6 - Advanced FEEL 32](#_Toc131237418)

[Introduction 32](#_Toc131237419)

[Instructions 32](#_Toc131237420)

[Import DMN project using Kie Sandbox 32](#_Toc131237421)

[Understanding the service 34](#_Toc131237422)

[Data Types 35](#_Toc131237423)

[Decisions 36](#_Toc131237424)

[Functions 37](#_Toc131237425)

[Feel Functions 39](#_Toc131237426)

[Run the service 40](#_Toc131237427)

[Extend the diagram 44](#_Toc131237428)

[3 Conclusion 45](#_Toc131237429)

# Introduction

In this guide we go beyond basics of DMN to build real-world DMN.

The following topics are presented:

* **Structuring Input data**
* **Applying patterns for large projects**
* **Hit Policies**
* **Advanced Feel**

By the end, you will be able to apply these real-world tips and techniques to your DMN projects.

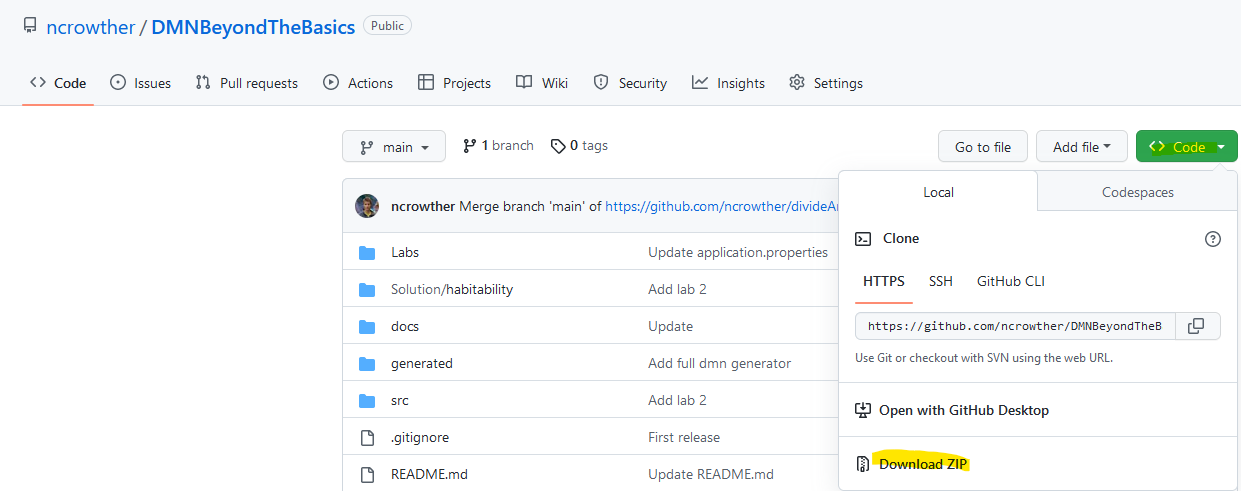
# Prerequisites

For this lab you need **Kie Sandbox** with *Kie Sandbox Extended Services* running. If you have not already done so, download and install the *Kie Sandbox Extended Services*.

You will also need a local copy of the following Git repo.

<https://github.com/ncrowther/DMNBeyondTheBasics>

Click on the link above and then click the *Code* button and then *Download ZIP*:

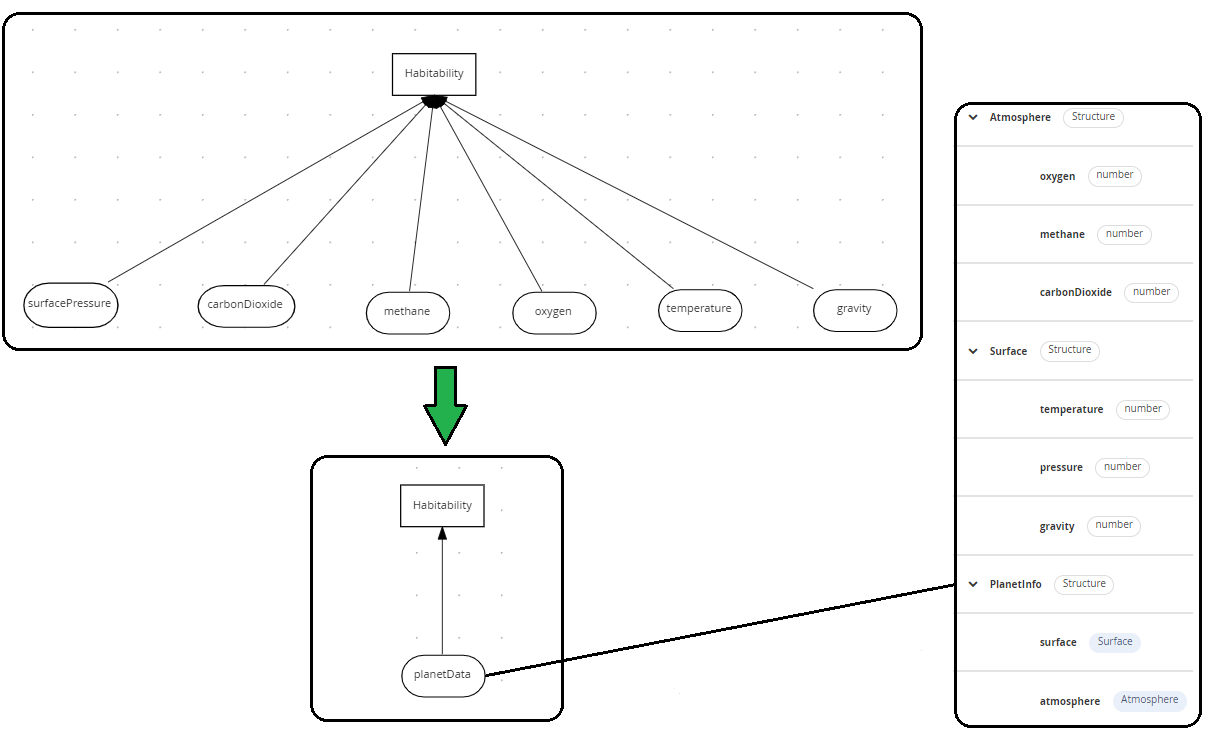


Unpack the zip to a local directory and note the location.

# Lab 1 - Data Types

## Introduction

In this lab you will learn how to simplify DMN diagrams by moving primitive attributes to data structures. See below:

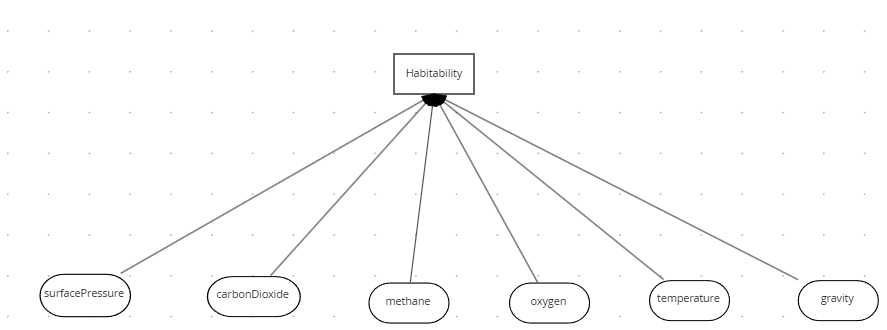


Applying a Data Model to Unstructured Data

## Instructions

1. In browsers Chrome or Safari open the web site <https://sandbox.kie.org/>  
     
   Graphical user interface, application, website

   Description automatically generated
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file*  and then U*pload…*
4. Select the file: *lab01*/*HabitabilityStart.dmn*
5. You should see this:

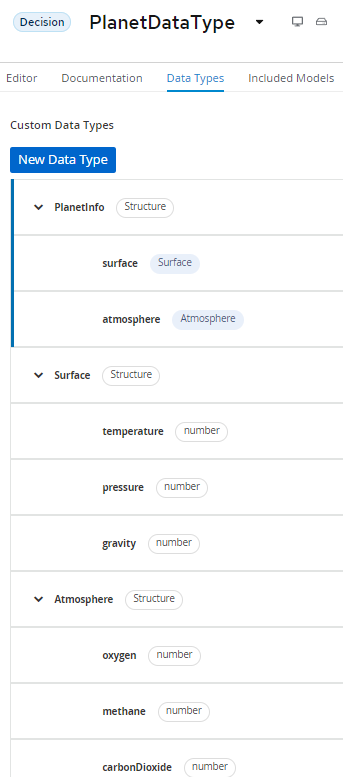


The inputs are simple data types and there are lots of them! This is an antipattern. We will show how to apply a data type to make it more readable.

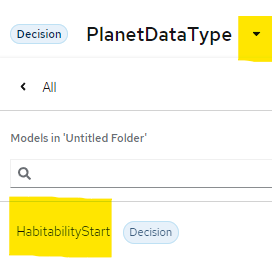
1. Click *New file*  and then Upload…
2. Select the file: *lab01/PlanetDataType.dmn*
3. Click on the *Data Types* tab:



1. You should now see the structure below. This is a data structure containing the attributes from the original diagram. See how it is structured into sub classes of *surface* and *atmosphere*.



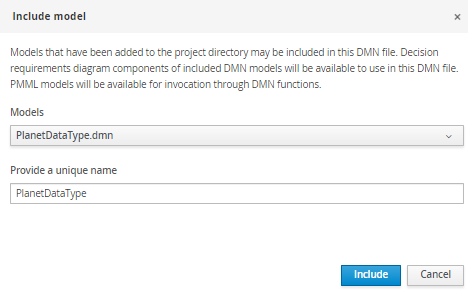
1. Go back to *HabitabilityStart* by clicking the drop-down arrow next to *PlanetDataType*:



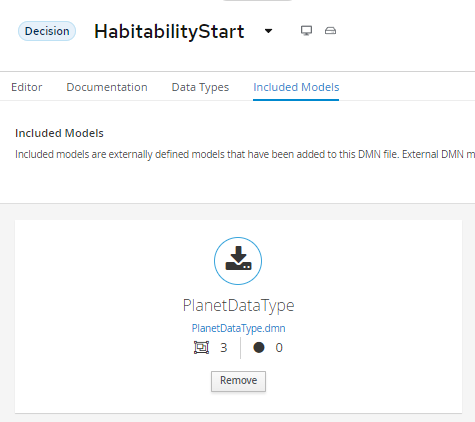
1. Within *HabitabilityStart*, Select **Included Models tab**, and then click **Include Model**



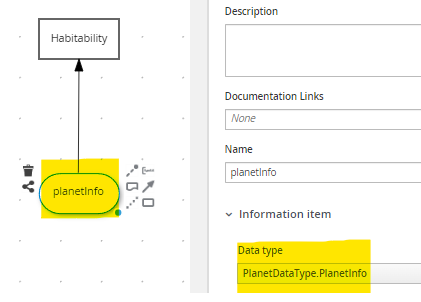
1. Add the *PlanetDataType* model and give it the same name of *PlanetDataType:*



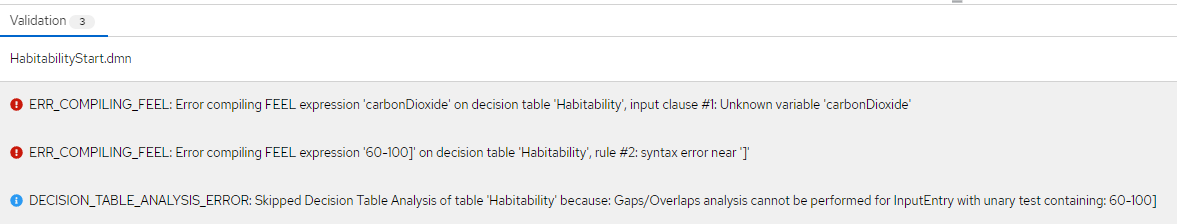
1. Click *Include.* You should see the model has been imported:



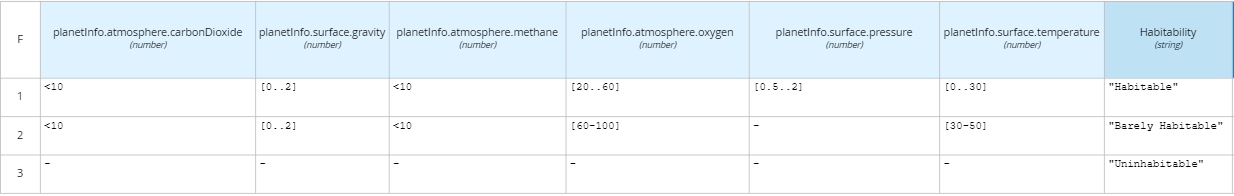
1. Switch to the **Editor** tab.
2. Back in the DMN Diagram, delete all the inputs and replace with one input called *PlanetInfo* assigning it a type of *PlanetDataType.PlanetInfo:*



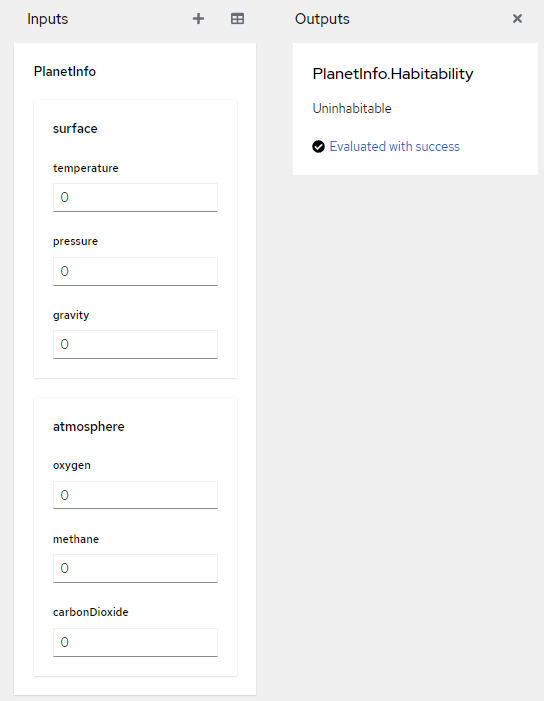
1. Click in the Problems button at the bottom right[[1]](#footnote-2). You should see errors:



The *Habitability* decision table is still referencing the primitive inputs. We need to fix this. Edit the table so that p*lanetInfo* is referenced instead of the primitives. See below:



1. Now there should be no errors and two warnings. We will review the warnings in a later lab. Test the model by pressing *Run*. If you have not installed the *Kie Server Extended Services*, now is the time to do so.
2. Enter zero values for all attributes and the expected result is *Uninhabitable :*



## Conclusion

In this lab we refactored a decision with many inputs into a decision with one input associated to a data structure. Moving primitives to data structures simplifies DMN.

# Lab 2 – The Divide and Conquer Pattern

## Introduction

The decision table in the previous lab will not scale. If all rule combinations were entered the table would be huge. To reduce the size can divide and conquer it into smaller parts.

To do this, we create a decision table for each planet attribute and reduce to an enumerated value: *Optimal, Bearable* or *Deadly.* Then we apply the enumerated values in a decision further up in the model. Using this pattern, we can split a wide table into several smaller manageable tables.

The tables are linked in the DMN diagram shown below:



Divide and Conquer Pattern for Habitability

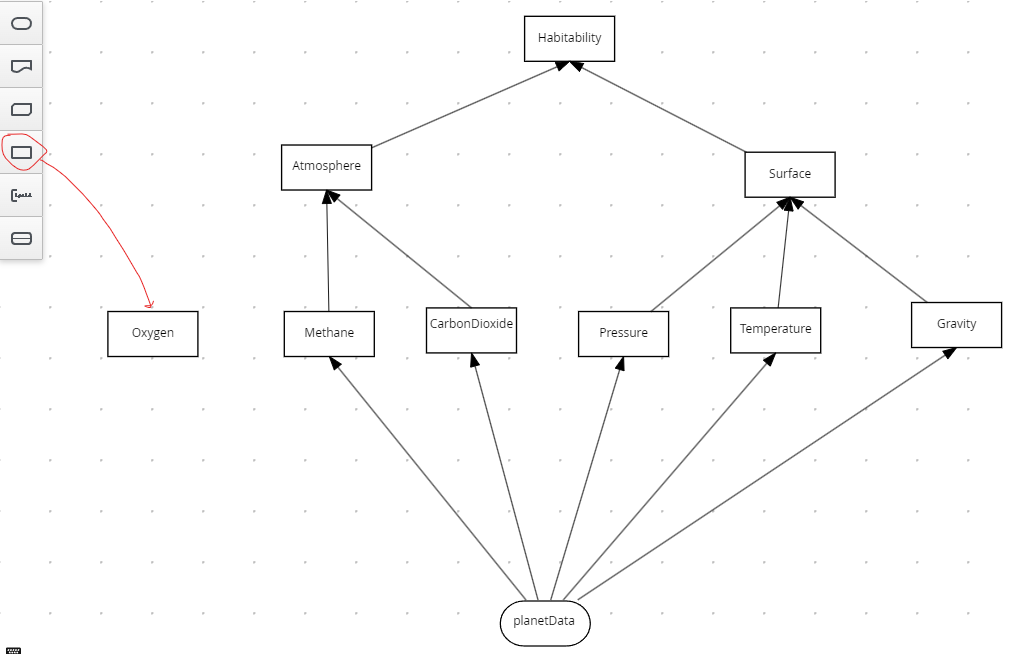
The **habitability** table has the same behavior as the original design, with the benefit that it is easier to maintain. Each attribute has its own table making it easier to change.

## Instructions

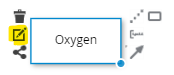
1. In browsers Chrome or Safari open the web site <https://sandbox.kie.org/>
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file*  and then U*pload…*
4. Select the file: *lab02*/*HabitabilityStart.dmn*
5. You should see this:



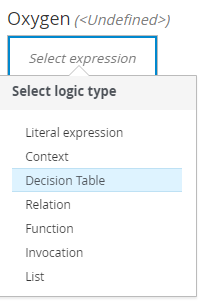
The model is nearly complete but is missing the *oxygen* attribute. Create a new Decision node and call it *Oxygen*.



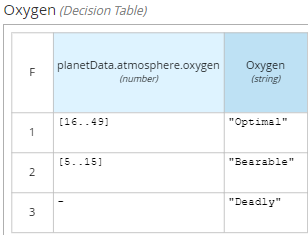
Select the decision and then click the *edit*  button:



Select *Decision Table* as the expression type:



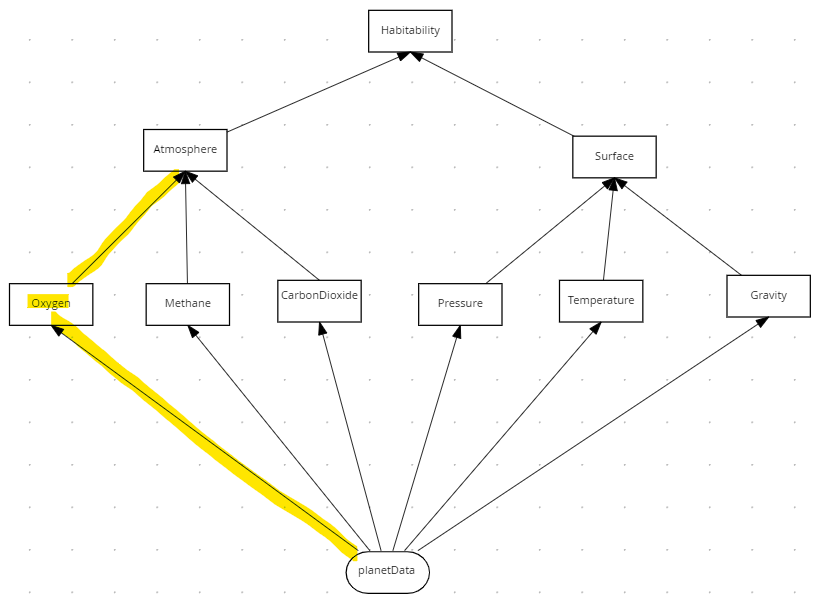
Create the table below.



Verify the following:

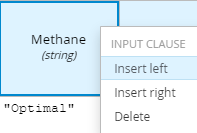
* planetData.atmosphere.oxygen is the condition variable
* Numeric ranges [16..49] and [5..15] are the conditions.
* Enumerated values are in quotes.
* The Hit policy is First (F)

Now plug the *Oxygen* Decision node to the diagram. You should have the following connections:

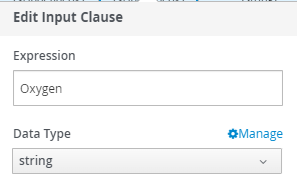


Edit the Atmosphere table so that it handles the Oxygen decision.

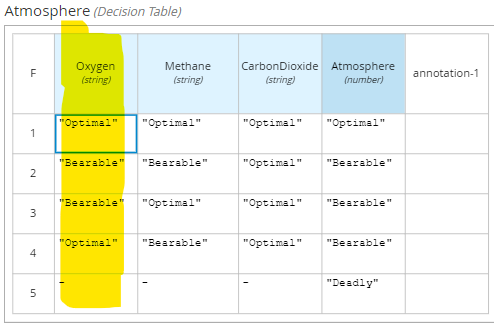
Right-click on the *Methane* column to add a new column to the left:



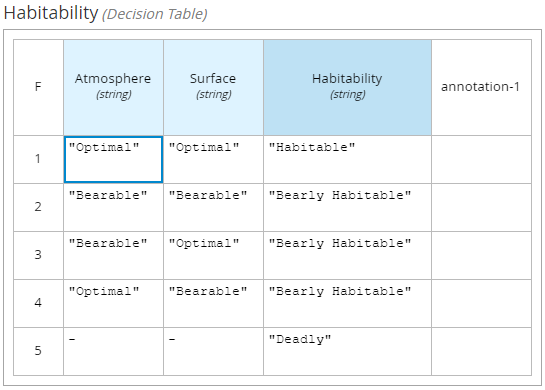
For the new column, set the Expression as *Oxygen* (case important) and the Data type as *string*.



Add four enumerated values (in quotes) as shown below:

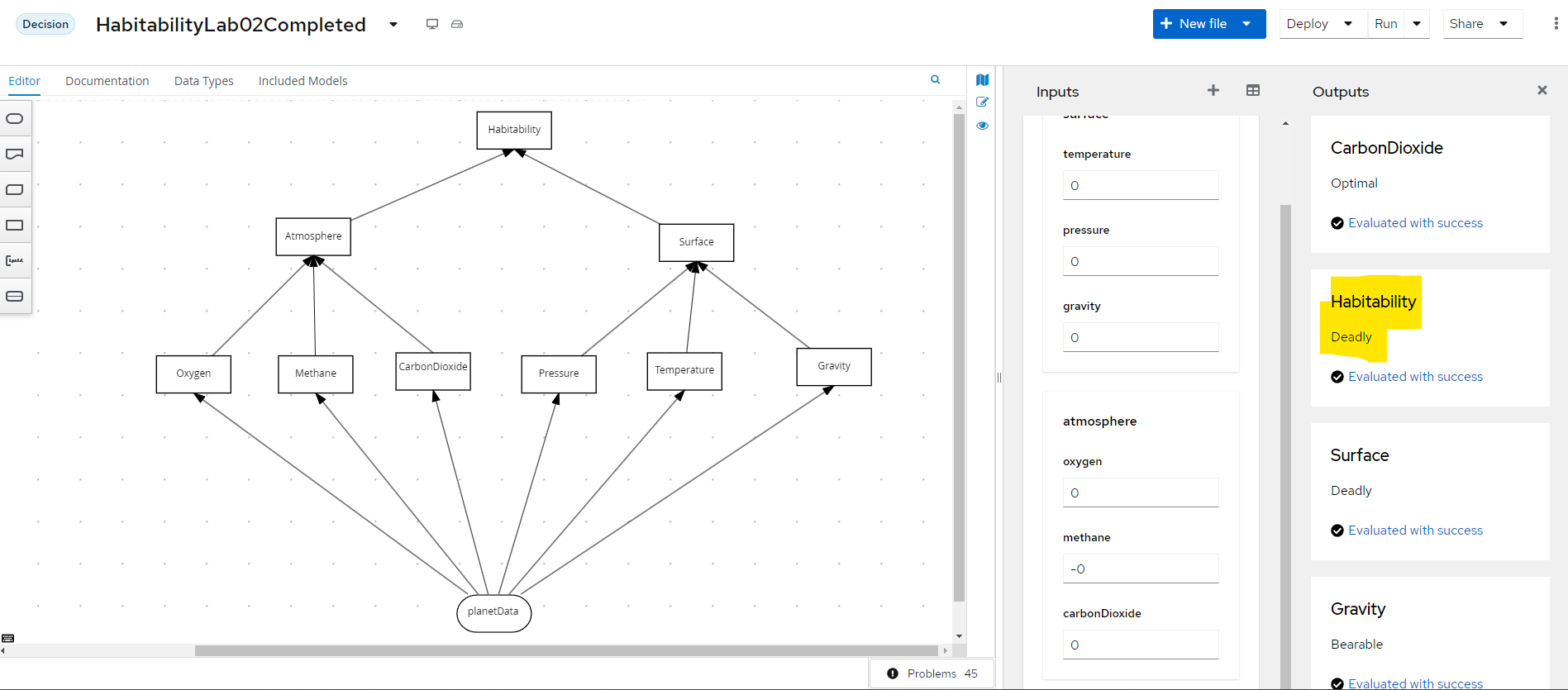


Now look at the root decision node, *Habitablilty*. It combines the decision in the lower tables to make the final decision as to the planet habitability:



We have now implemented the Divide and Conquer pattern by splitting up the wide decision table in lab 01 into several smaller ones! Let’s test the service.

1. Press the *Run* button
2. Enter zero values for all inputs and the expected result is *Deadly*:



## Conclusion

In this lab we refactored the Habitability rules into separate decision services that build up the overall decision. Separating a single decision table in this way helps manage complex decisions with many inputs.

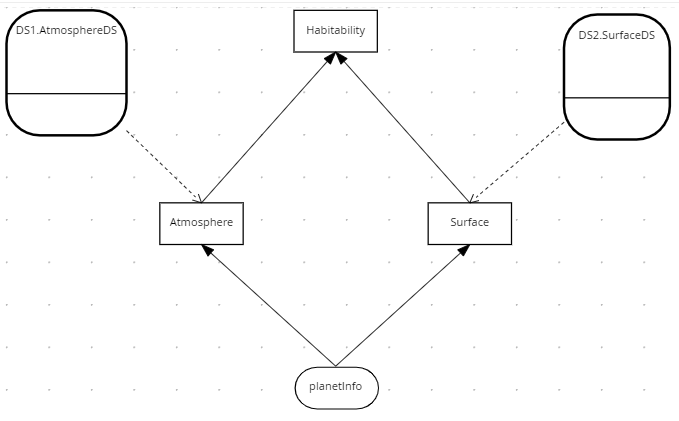
# Lab 3 – The Tiered Service Pattern

## Introduction

The DMN defined in the previous lab is confined to a single file. This is a problem:

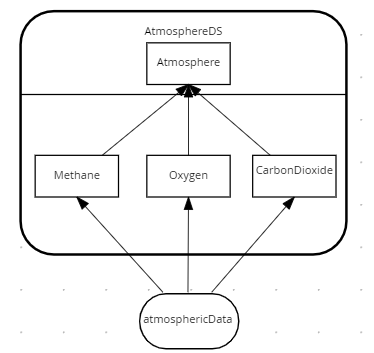
* The DMN cannot be edited simultaneously by separate people
* The DMN cannot be reused in other decisions.

To improve, we move the second-tier decisions into their own decision services and then invoke these decisions from the top tier. See below, where the second-tier decision services are *AtmosphereDS* and *SurfaceDS*:

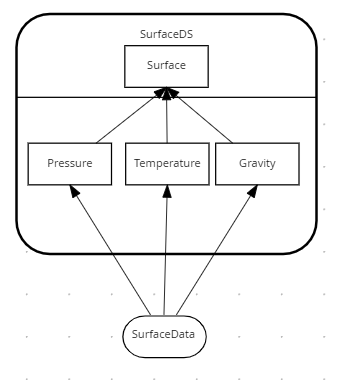


Tiered Service Pattern

The two second tier decision services are *AtmosphereDS* and *SurfaceDS*:



Atmosphere Decision Service



Surface Decision Service

In the Habitability example we have only two tiers – but in practice the pattern could be applied to multiple tiers, with second tier decision services calling third tier services, and so on. The number of tiers depends on project complexity.

## Instructions

1. In browsers Chrome or Safari open the web site <https://sandbox.kie.org/>
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file*  and then U*pload…*
4. Select file: *lab03*/*SurfaceDS.dmn*
5. Repeat steps 3 and 4 for a further two files:

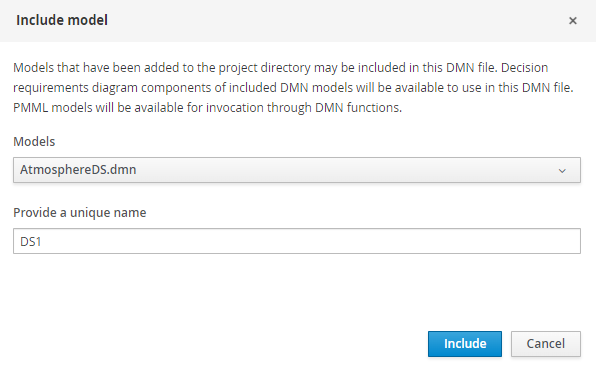
*lab03*/*AtmosphereDS.dmn*

*lab03*/*HabitabilityStart.dmn*

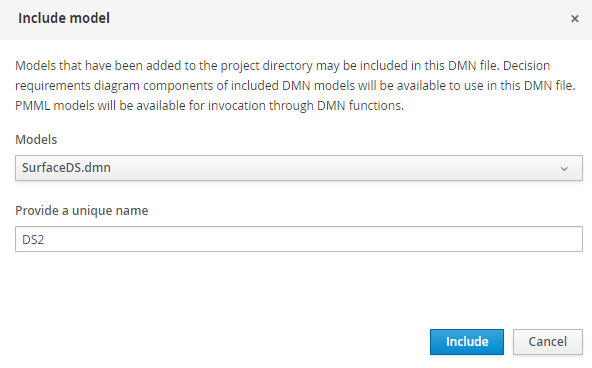
1. Within *HabitabilityStart*, Select **Included Models tab**, and then click **Include Model**



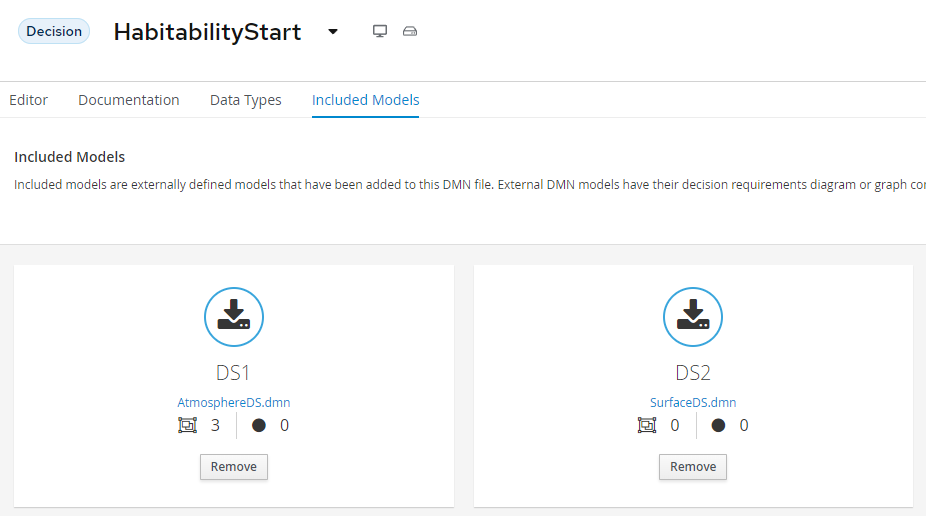
1. Add the *AtomosphereDS.dmn* and give it the name of *DS1:*



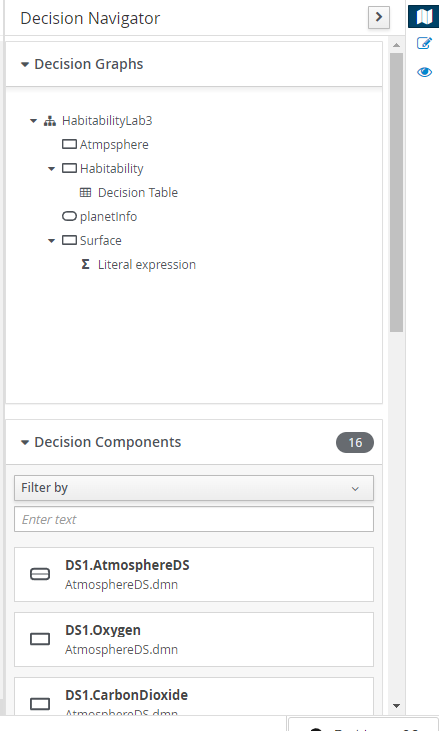
1. Click *Include.*
2. Click *Include Model* againand import *SurfaceDS.dmn.* Give it a name of *DS2:*

**

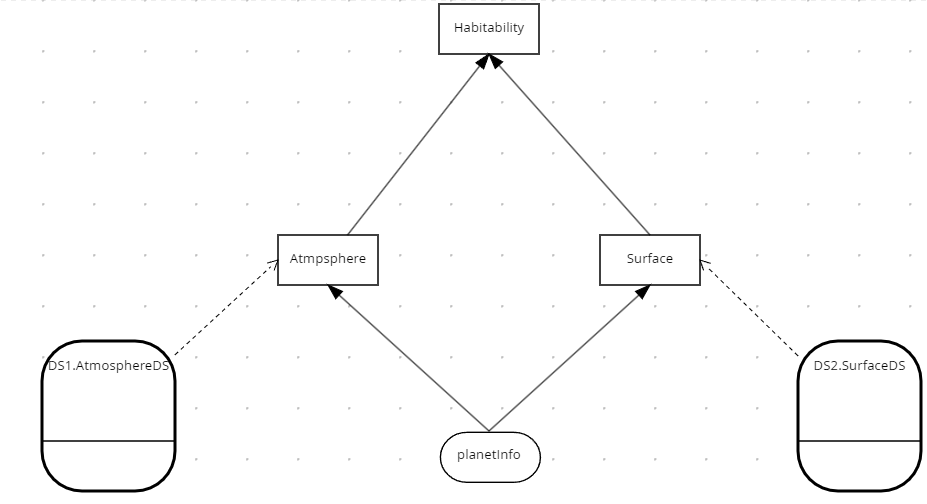
1. You should see both decision services have been imported:



1. Select the *Editor tab.*
2. Select the Decision Navigator button on the far right . You should see the following:

**

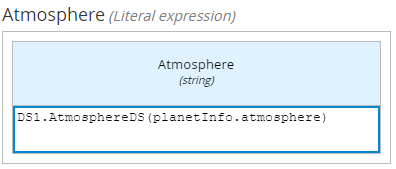
1. Drag and drop *DS1.AtmosphereDS* and *DS2.SurfaceDS* to the diagram
2. Using the arrow connector, connect *DS1.AtmosphereDS* to *Atmosphere* and *DS2.SurfaceDS* to *Surface*. You should now see the following:



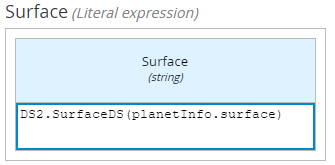
1. Edit the Atmosphere Decision Node. Create a Literal Expression and the FEEL expression:

DS1.AtmosphereDS(planetInfo.atmosphere)

1. Set the return type of *Atmosphere* to *string*. You should see this.



1. Edit *Surface* Decision. You should see it has been predefined to call the *DS2.SurfaceDS* decision service:



1. Press the *Run* button
2. Enter zero values for all inputs and the expected result is *Deadly*

## Conclusion

In this lab we refactored the Habitability rules to call separate decision services. Separating a single decision into multiple services helps scale your DMN projects.

# Lab 4 – The Index Pattern

## Introduction

The Divide and Conquer Pattern does not work with many rules and few conditions. Instead, we can apply the Index Pattern. An example is SWIFT interchange rules. A simplified SWIFT message is:

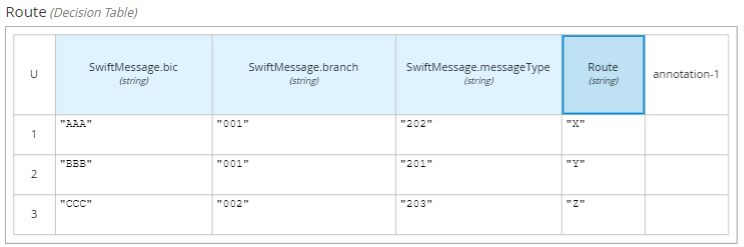
**BIC** – A Bank Identifier Code

**Receiving Branch** – The bank branch

**Message type** – The type of payment

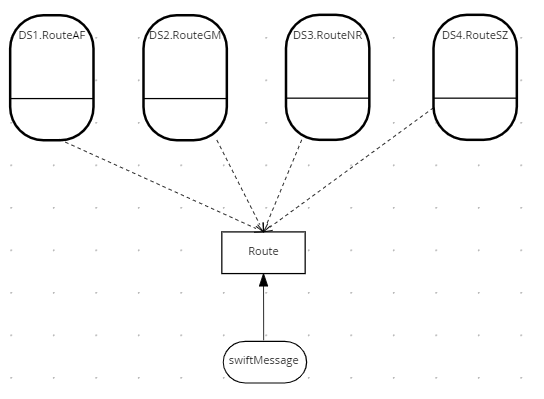
**Route** – The route the message is sent

The rules below decide where to route a SWIFT message within the banking network:



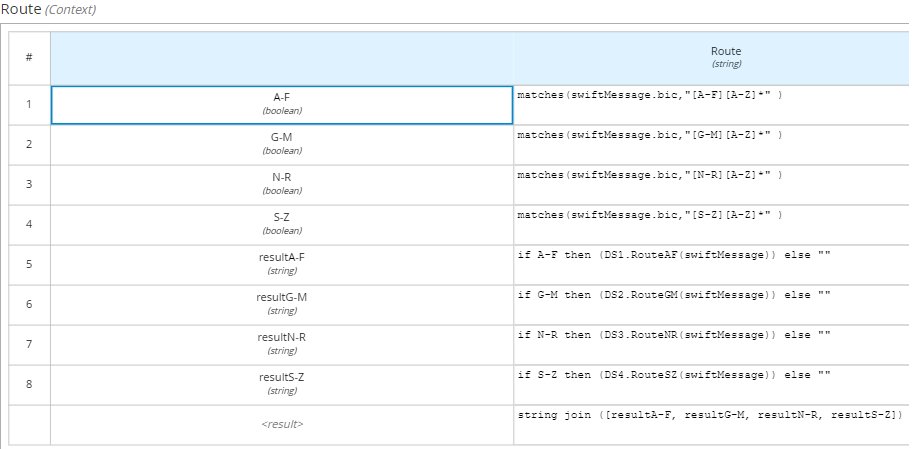
There could be thousands of these rules based on combinations of *Bic, Branch* and *Message Type*.

In DMN this we could model the decision using the Index pattern:

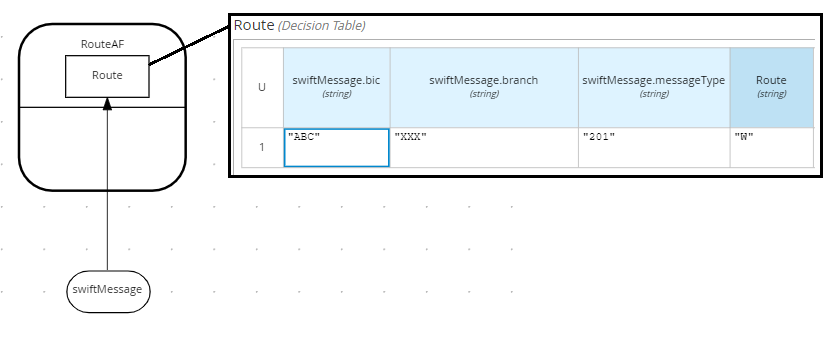


The key logic is the *Route* decision. It routes the message to an indexed decision service based on the first letter of the BIC. This splits the table into four smaller tables, each alphabetically indexed. There could be further indexing within each of these decision services to split the decision table further

Here is the logic for the Route decision:



The indexed decision service caters for each part of allocated the alphabet. The service handling BICs starting A-F is shown below.



The decision services handling BICs G-M, N-R and S-Z follow the same pattern.

## Instructions

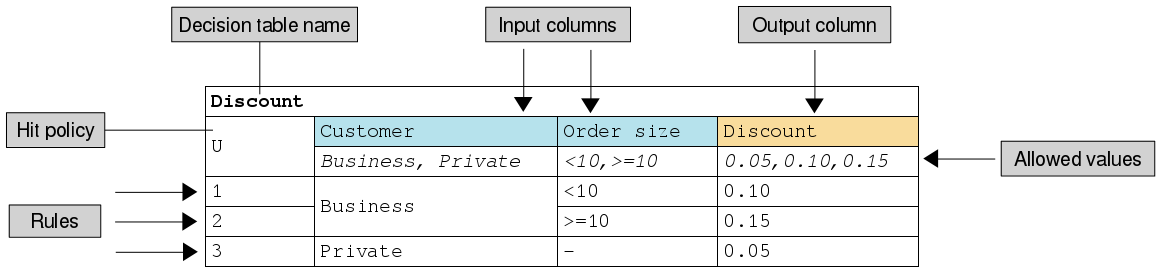
## Conclusion

In this lab we split a big decision table with few inputs into several smaller tables using the Index Pattern.

# Lab 5 - Hit Policies for Decision Tables

## Introduction

Not all decisions are the same! Some count outcomes, some require a go / no go decision and others require exact reasoning. You need a different hit policy for each of these. In DMN the hit policy is specified in the top left of the decision table. See below:



Decision Table

Hit policies change the evaluation order of a decision table. Each hit policy requires different ways of reasoning over the table. Therefore, it is important to be able to select the right hit policy for your application.

The commonly used hit policies are:

|  |  |  |
| --- | --- | --- |
| Hit Policy | Description | When to use |
| ****Unique (U)**** | Permits only one rule to match. Any overlap raises an error. | For detailed reasoning. Ensures your rules cover all cases and are complete |
| First (F) | Rules are evaluated from top to bottom. Rules may overlap, but only the first match counts. | For concise decision tables where a go / no go decision is needed rather than complete reasoning. |
| ****Collect (C)**** | Aggregates values in an arbitrary list. | For multiple row satisfaction.  **String** aggregator:   * <None> - matching instances returned as list * Count – Matching instances counted   **Number** aggregator:   * SUM – matching instances added * Count – matching instances counted * Min – minimum value * Max – maximum value |

Commonly Used Decision Table Hist Policies

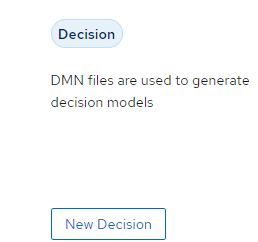
Other less commonly used hit policies are **Any** which has similar behaviour to **Unique**, and **Priority**, **Rule Order** and **Output Order** which have similar behaviour to **Collect.**

## Instructions

Open **Kie Sandbox** in Chrome:

<https://sandbox.kie.org/#/>

Create a new decision:



Click *New File,* thenimport the following DMN as a URL. For each hit policy, run the rules and examine the results. Which algorithm is most appropriate for this use case?

### Unique Policy

<https://github.com/ncrowther/divideAndConquerDMN/blob/main/Labs/Lab01/src/main/resources/org/acme/habitability/Lab01/HabitabilityLab00-Unique.dmn>

### First Policy

<https://github.com/ncrowther/divideAndConquerDMN/blob/main/Labs/Lab01/src/main/resources/org/acme/habitability/Lab01/HabitabilityLab00-First.dmn>

### String Collection policy

<https://github.com/ncrowther/divideAndConquerDMN/blob/main/Labs/Lab01/src/main/resources/org/acme/habitability/Lab01/HabitabilityLab00-Collect.dmn>

### Numeric Collection policy

<https://github.com/ncrowther/divideAndConquerDMN/blob/main/Labs/Lab01/src/main/resources/org/acme/habitability/Lab01/HabitabilityLab00-Count.dmn>

# Lab 6 - Advanced FEEL

## Introduction

In the final lab we put together what we have learnt so far and examine an advanced DMN example. The example has additional documentation here:

<https://access.redhat.com/documentation/en-us/red_hat_process_automation_manager/7.12/html/developing_decision_services_in_red_hat_process_automation_manager/dmn-con_dmn-models>

## Instructions

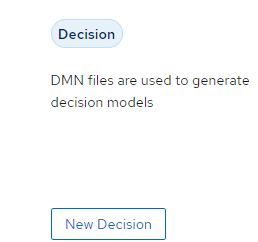
In this lab we will examine an advanced DMN decision service that reschedules flights.

### Import DMN project using Kie Sandbox

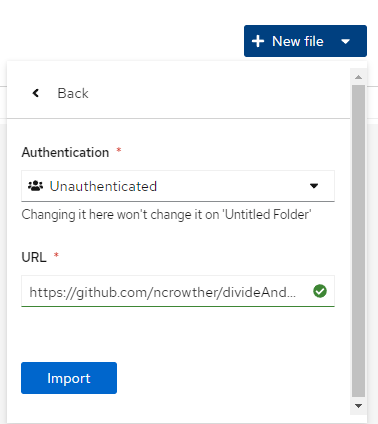
1. Open **Kie Sandbox** in Chrome:

<https://sandbox.kie.org/#/>

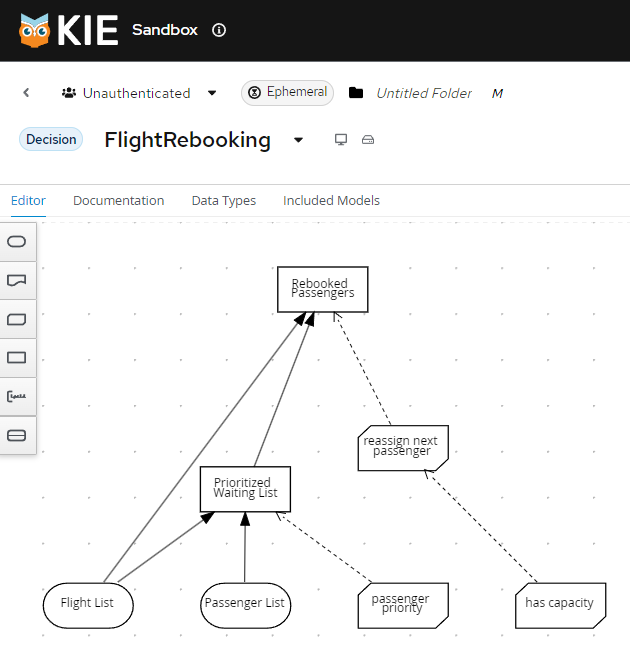
1. Create a new decision:



1. Click *New File* **
2. Select *From URL*.
3. Paste the text below into the URL: <https://github.com/ncrowther/divideAndConquerDMN/blob/main/Labs/Lab00/src/main/resources/org/acme/habitability/Lab00/FlightRebooking.dmn>
4. Click Import 



You should see the following:

­

Passenger Rebooking Service

### Understanding the service

The DMN has the following components:

**Data Types**

A list of flights

A list of passengers

**Decisions**

Prioritizing the passengers who will get seats on a new flight

Determining which flights those passengers will be offered

**Business knowledge models**

The company process for determining passenger priority

Any flights that have space available

Company rules for determining how best to reassign inconvenienced customers

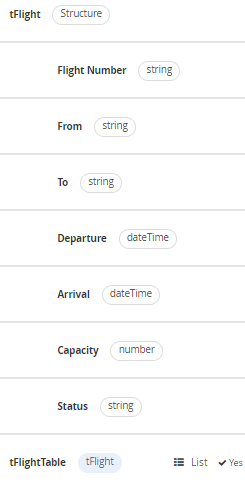
Then, the airline uses the DMN standard to model its decision process in a decision requirements diagram (DRD), and creates the following diagram for determining the best rebooking solution:

**Feel Functions**

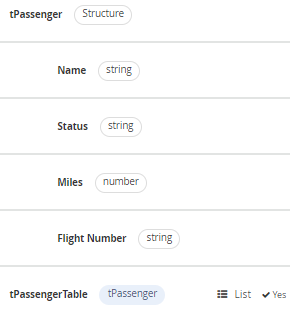
* Sort
* List contains

Let’s examine each of these components in detail, starting with data types.

### Data Types

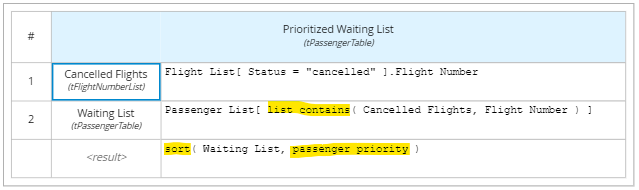


Flight

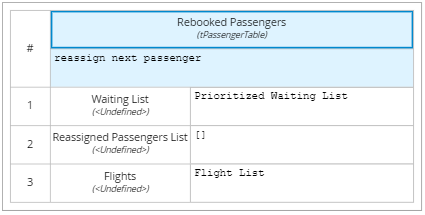


Passenger

### Decisions

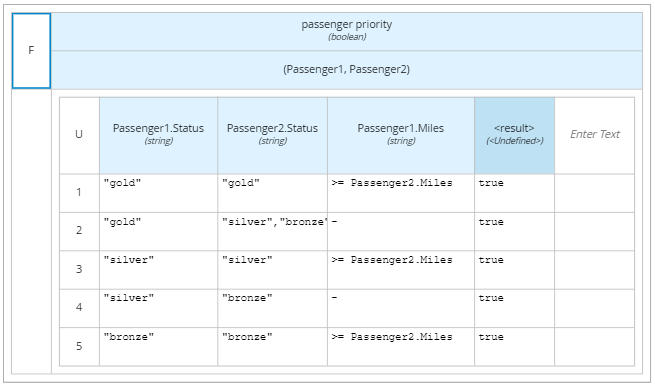


Prioritized Waiting List

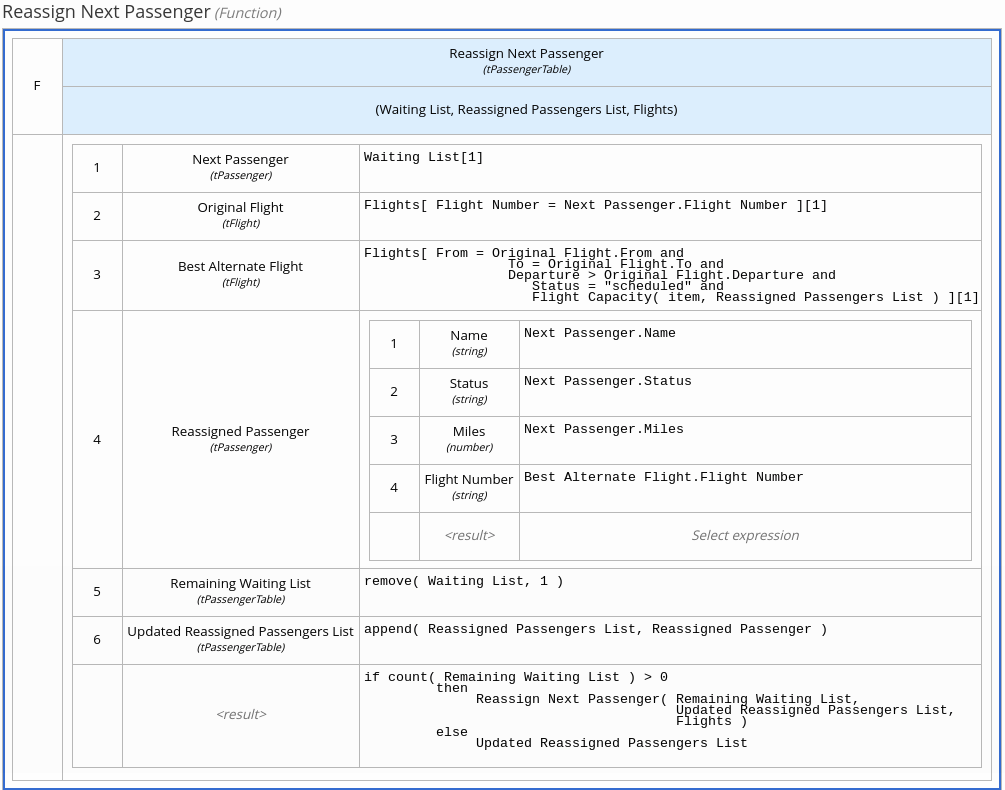


Rebooked Passengers

### Functions



Passenger Priority Function



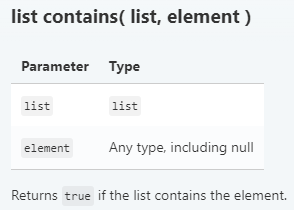
Reassign Next Passenger Function



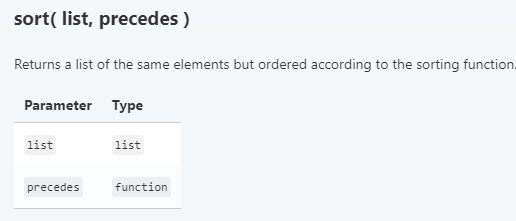
Has Capacity Function

### Feel Functions

<https://kiegroup.github.io/dmn-feel-handbook/#sort-functions>



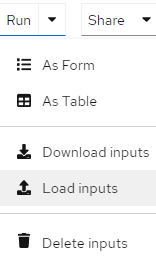
List Contains Function



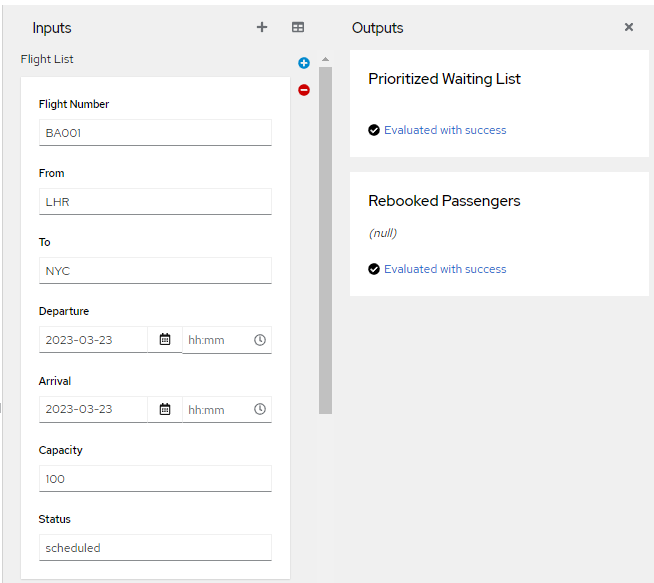
Sort Function

### Run the service

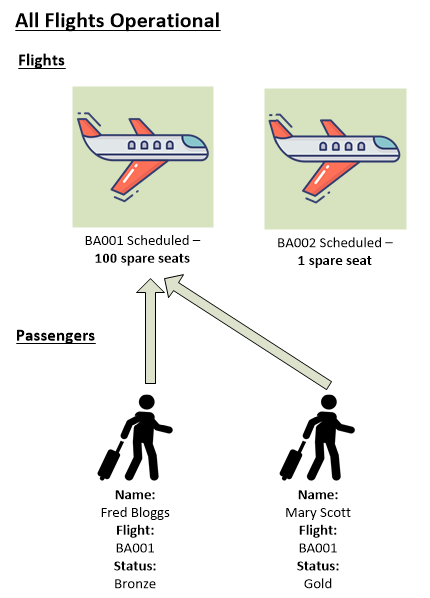
1. Now add some test data. Click *Run->Load Inputs*



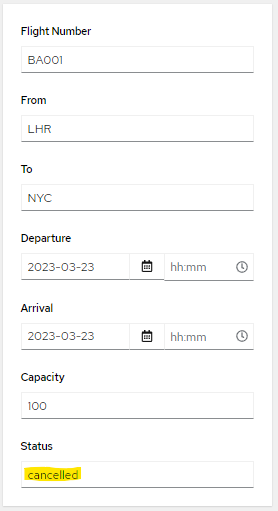
1. Open */Labs/Lab00/src/test/resources/FlightRebooking.json*
2. Click 
3. The service will run using the test data. You should see the test data in the *Inputs* panel and the results in the *Outputs* panel:



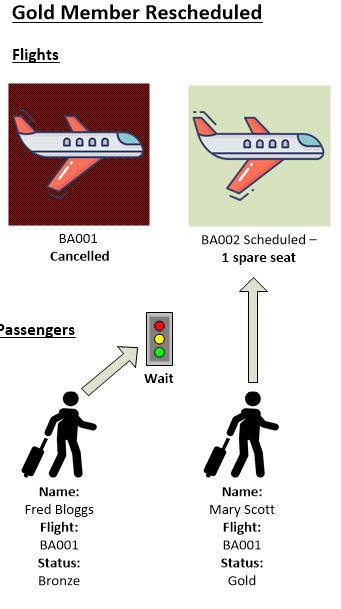
The output contains no waiting list or rebooked passengers all flights are operational. This is represented by the diagram below:



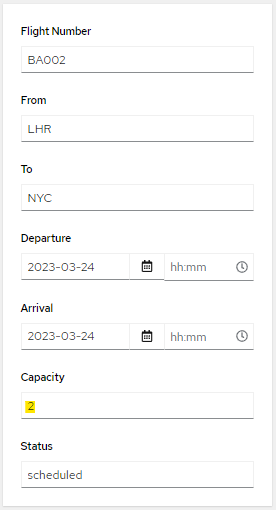
Now change the status of flight BA001 to *cancelled* (all lower case):



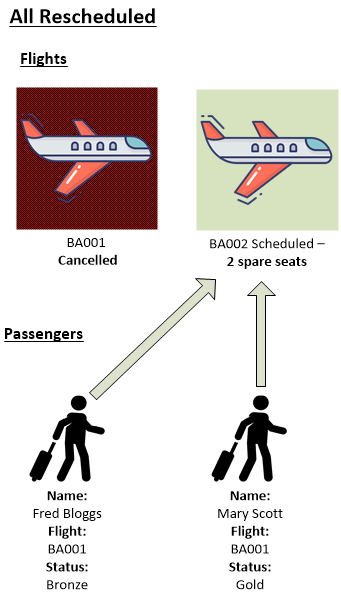
Check that *Mary Scott* is rebooked onto flight *BA002*. *Fred Bloggs* must wait as there is only one spare seat and Mary is a gold member.



Now change the spare seat capacity on flight *BA002* to *2*:



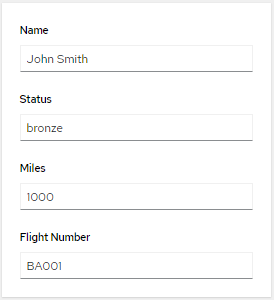
You should see that both Mary and Fred are scheduled to fly on *BA002*:



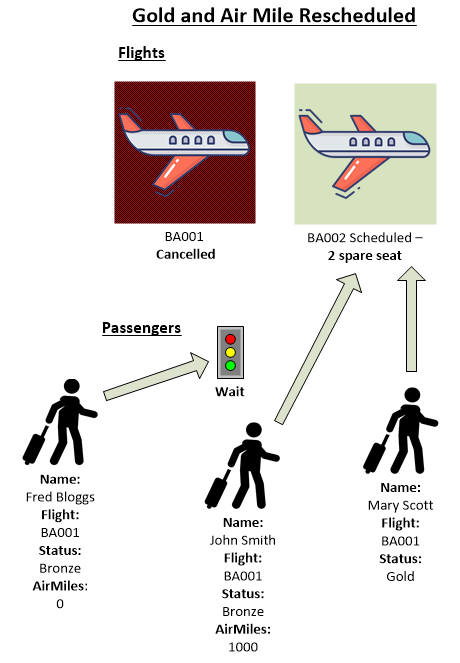
Add another passenger, *John Smith*. To do this, click on the plus button next to *Passenger List* in the *Inputs* Panel:



Add the following data:



After you enter the data, the test should automatically run. Verify that *Fred Bloggs* is put on hold again as *John Smith* has 1000 air miles and *Fred* Bloggs has 0



### Extend the diagram

Modify the DRG so that is formatted as shown in Figure 1. (to be expanded)

Modify the service (to be expanded)

# Conclusion

In this lab we went beyond the basics of DMN to give ideas and techniques for building real world DMN projects.

1. You will need *Kie Sandbox Extended Services* installed. Hit the *Run* button and follow the instructions. [↑](#footnote-ref-2)