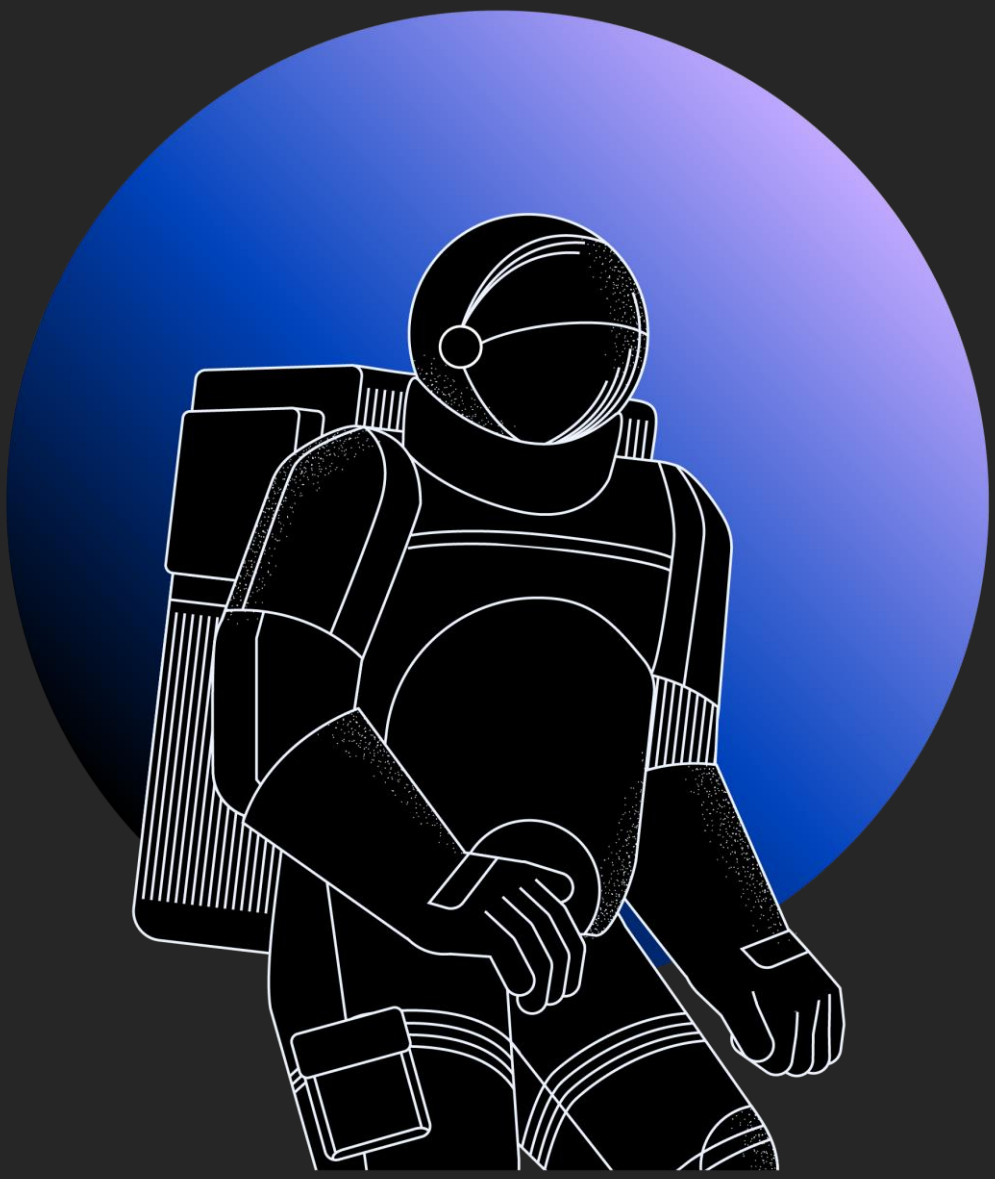
Lab Guide

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 t he 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 o r 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 compl etely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any sim ilarity 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 count ries.

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 *4*

2 Prerequisites *5*

Lab 1 - Data Types *6*

Introduction **6**

Instructions **7**

Conclusion **11**

Lab 2 – The Divide and Conquer Pattern *12*

Introduction **12**

Instructions **12**

Conclusion **17**

Lab 3 – The Tiered Service Pattern *18*

Introduction **18**

Instructions **20**

Conclusion **23**

Lab 4 – The Index Pattern *24*

Introduction **24**

Instructions **26**

Conclusion **30**

Lab 5 - Hit Policies *31*

Introduction **31**

Instructions **33**

Unique Policy 33

Any Policy 36

First Policy 38

String Collection policy 39

Numeric Collection policy 40

Conclusion **41**

Lab 6 - Advanced DMN *42*

Introduction **42**

Instructions **42**

A Quick Tour of the Flight Rebooking Service 43

Extend the diagram 48

3 Conclusion *50*

1 Introduction

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

The following topics are presented:

- **Data Types**
- **Applying patterns for large projects**
- **Hit Policies**
- **Advanced DMN**

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

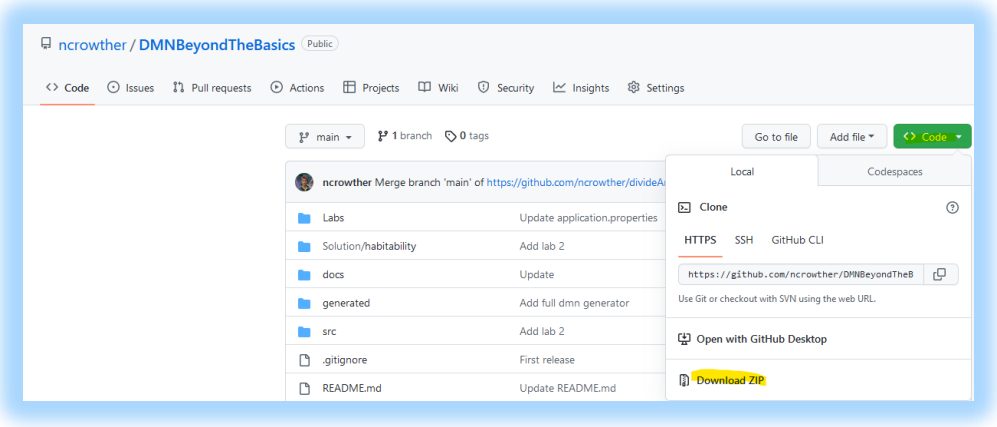
2 Prerequisites

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

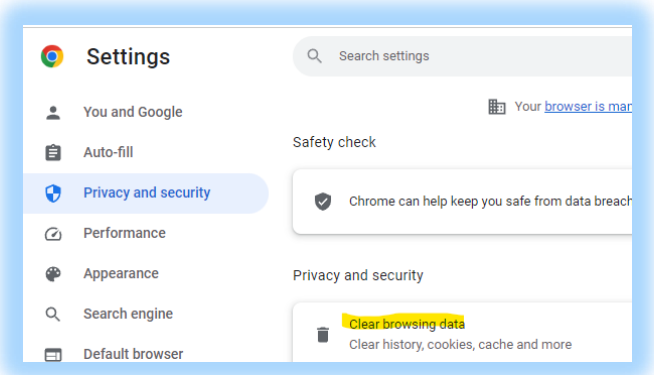
Click on the link above and then click the *Code* button and then *Download ZIP*:



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

Clearing Browser Data

After each lab clear your cache. In Chrome, click ⋮ then select *Settings*. Select *Privacy and security* and press *Clear browsing data:*



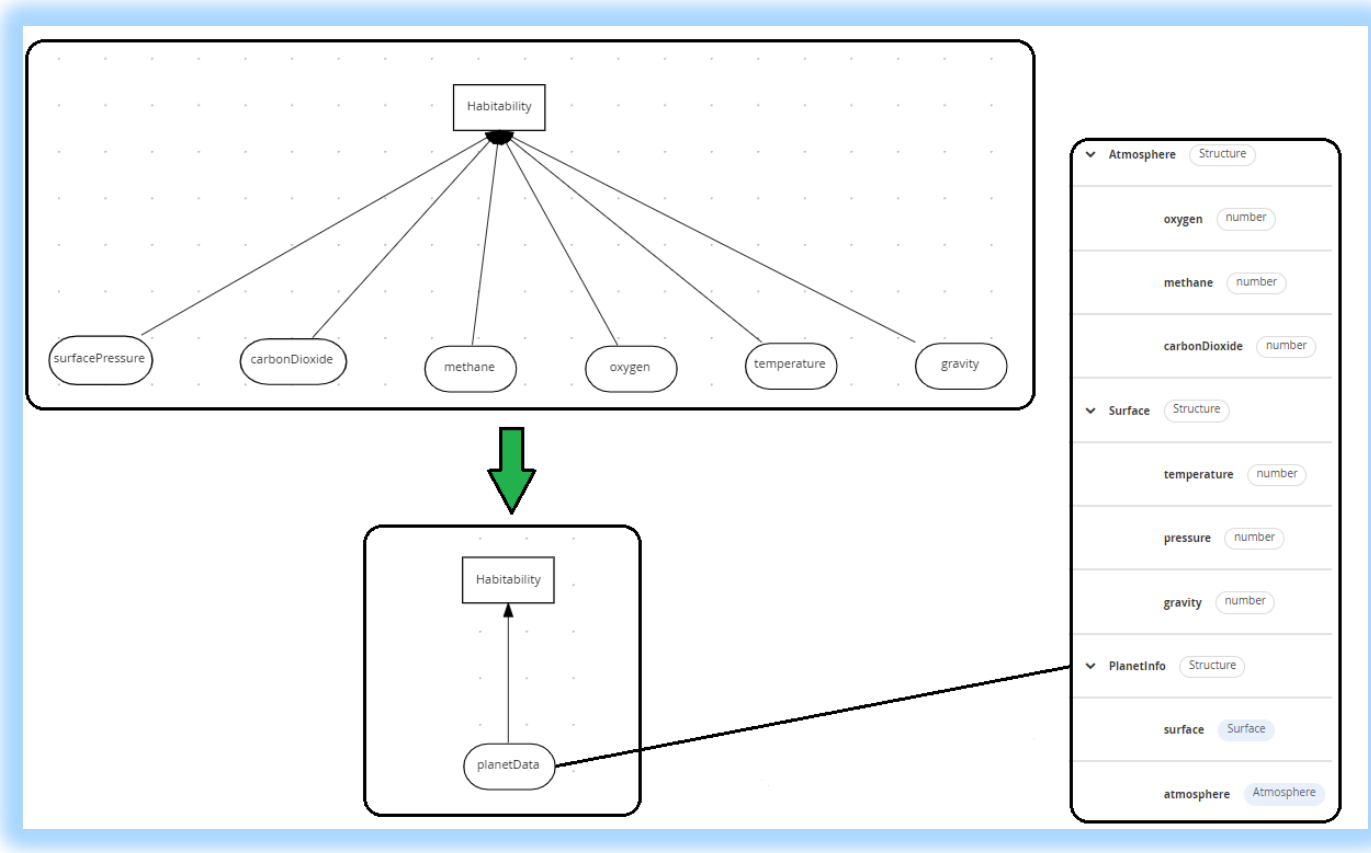
Additional Documentation

Additional documentation on DMN is found here:
https://access.redhat.com/documentation/en-us/red_hat_decision_manager/7.8/html/designing_a_decision_service_using_dmn_models

Lab 1 - Data Types

Introduction

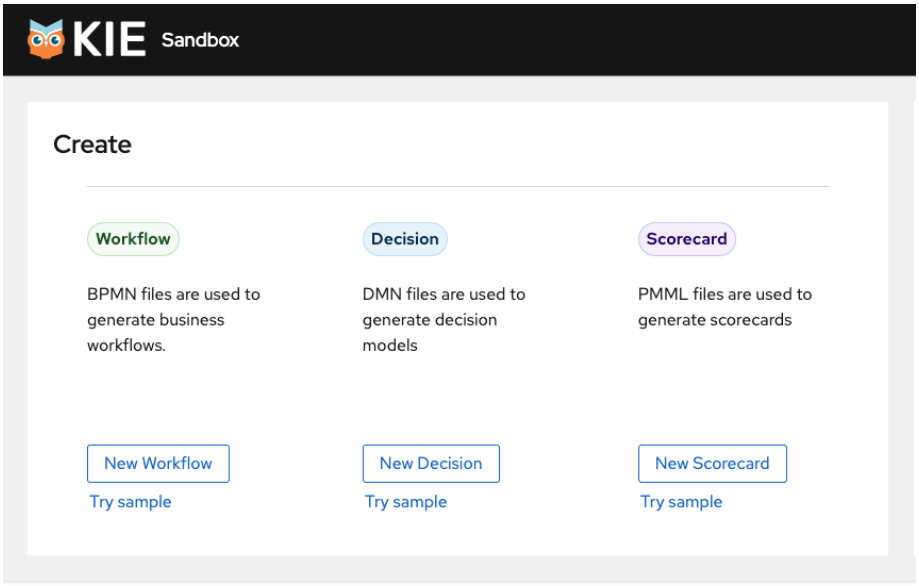
In this lab you will learn how to simplify DMN diagrams by moving primitive attributes to data structures. See below:

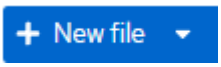


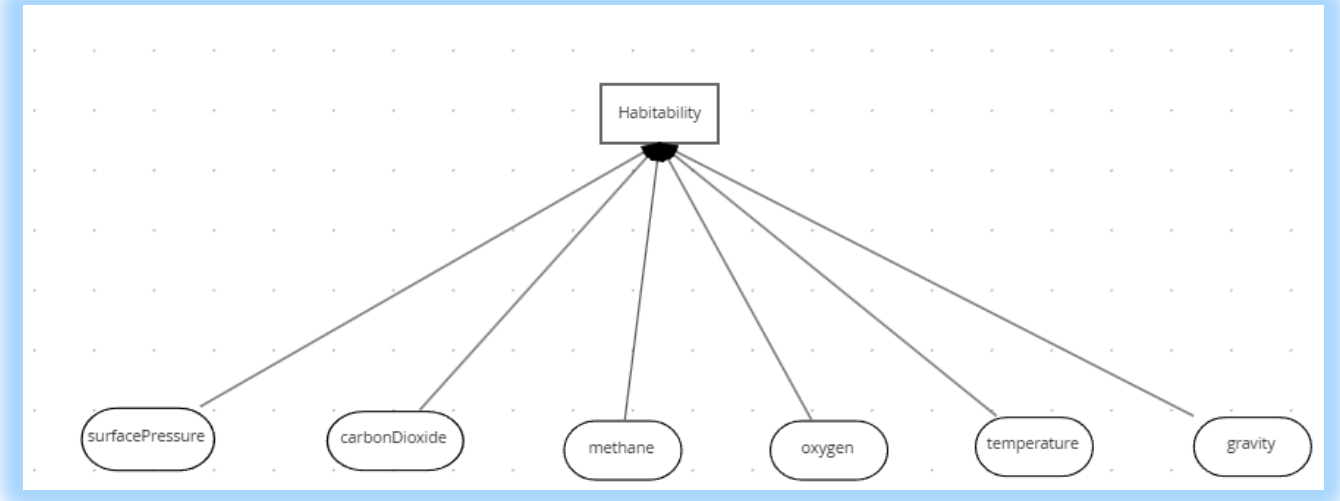
Applying a Data Model to Unstructured Data

Instructions

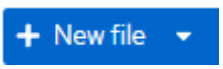
1. In browsers Chrome or Safari open the web site https://sandbox.kie.org/



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:



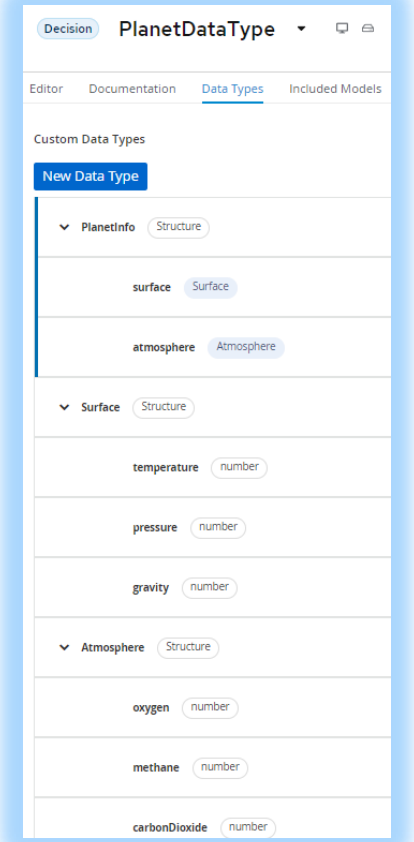
The inputs are simple data types and there are lots of them! This is an antipattern. We will show how to apply a data type to make it more readable.

6. Click *New file*  and then Upload...

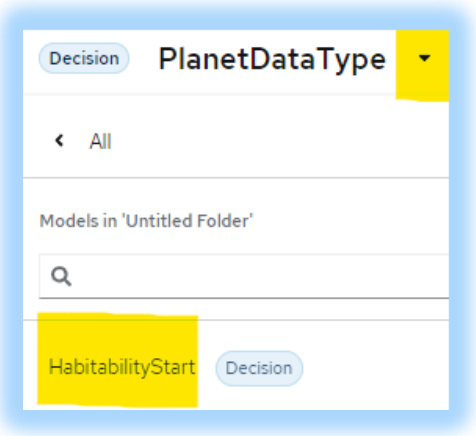
7. Select the file: *lab01/PlanetDataType.dmn*

8. Click on the *Data Types* tab:

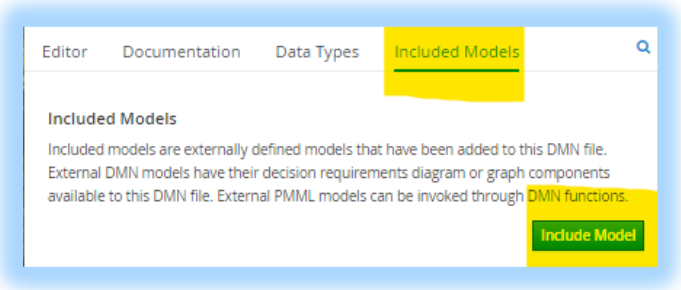
9. You should now see the structure below. This is a data structure containing the attributes from the original diagram. See how it is structured into sub classes of *surface* and *atmosphere*.



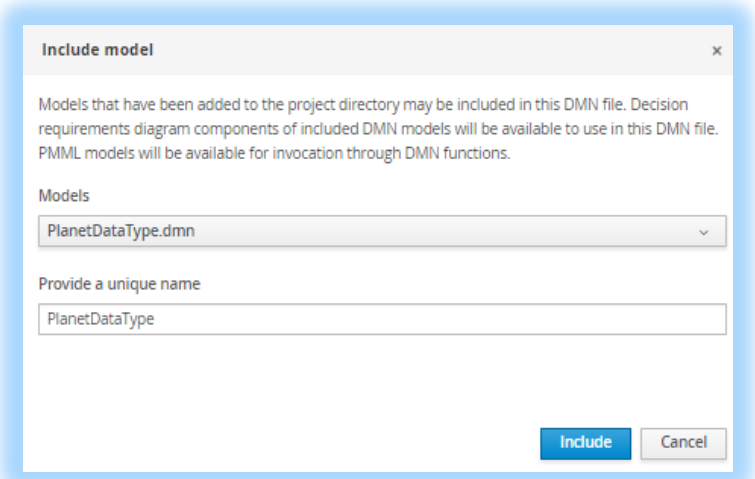
10. Go back to *HabitabilityStart* by clicking the drop-down arrow next to *PlanetDataType*:



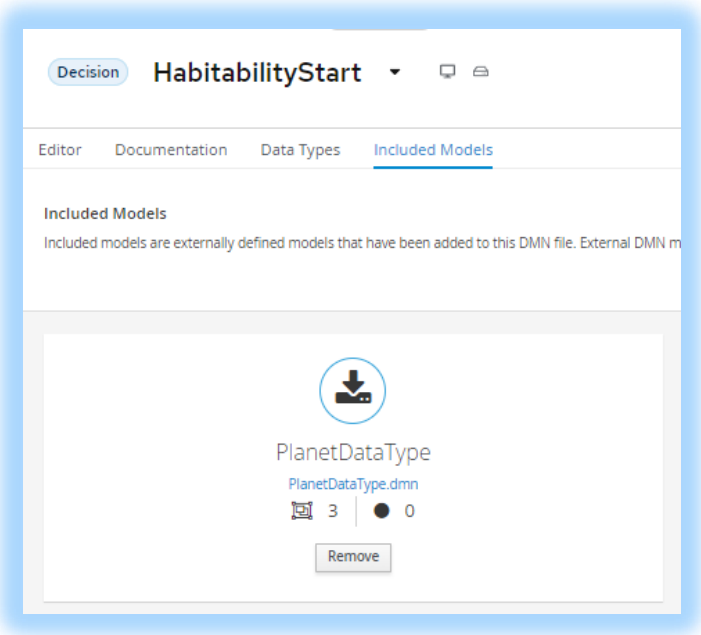
11. Within *HabitabilityStart*, Select **Included Models** tab, and then click **Include Model**



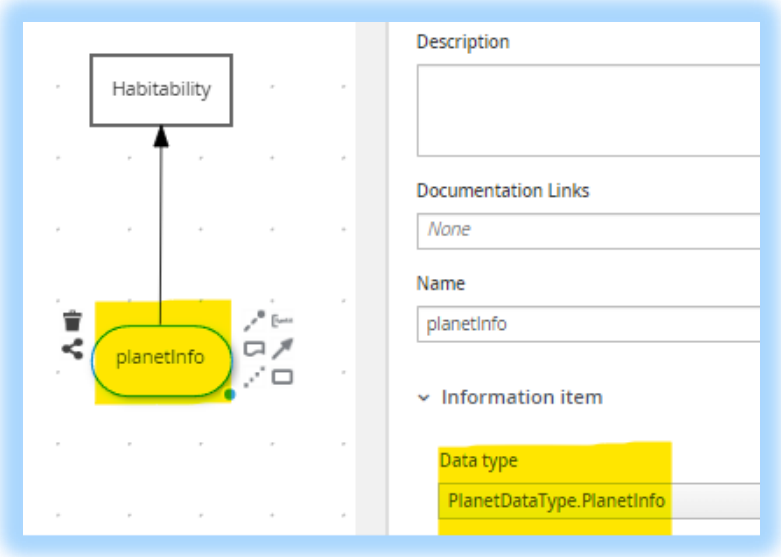
12. Add the *PlanetDataType* model and give it the same name of *PlanetDataType:*



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

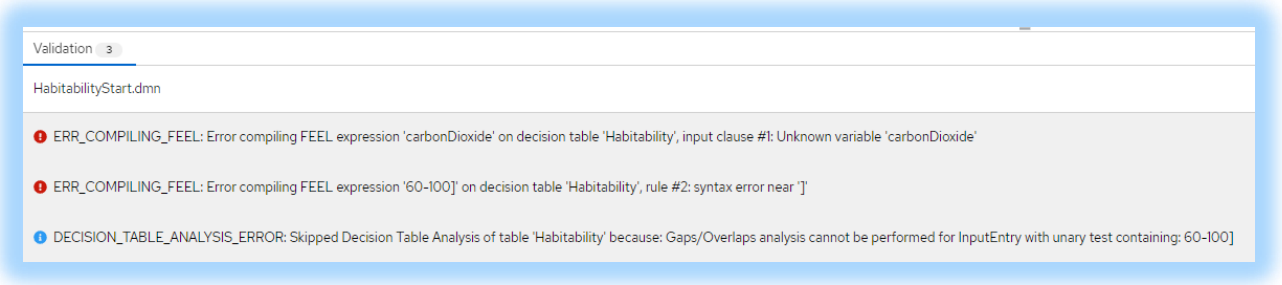


14. Switch to the **Editor** tab.
15. Back in the DMN Diagram, delete all the inputs and replace with one input called *PlanetInfo* assigning it a type of *PlanetDataType.PlanetInfo:*

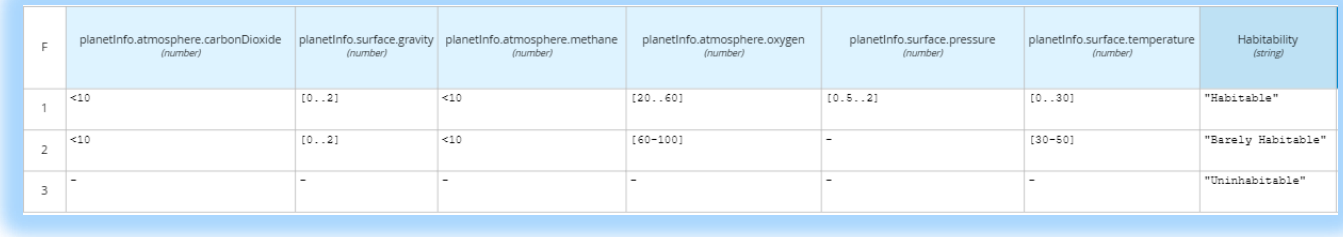


16. Click in the Problems button at the bottom right[1]. You should see errors:

[1] You will need *Kie Sandbox Extended Services* installed. Hit the *Run* button and follow the instructions.



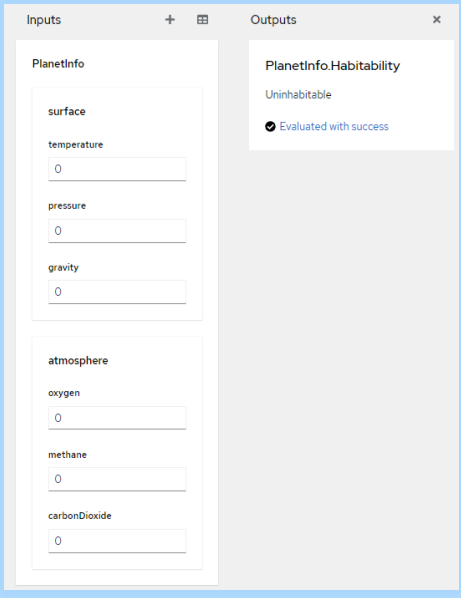
The *Habitability* decision table is still referencing the primitive inputs. We need to fix this. Edit the table so that p*lanetInfo* is referenced instead of the primitives. See below:



F	planetInfo.atmosphere.carbonDioxide *(number)*	planetInfo.surface.gravity *(number)*	planetInfo.atmosphere.methane *(number)*	planetInfo.atmosphere.oxygen *(number)*	planetInfo.surface.pressure *(number)*	planetInfo.surface.temperature *(number)*	Habitability *(string)*
1	<10	[0..2]	<10	[20..60]	[0.5..2]	[0..30]	"Habitable"
2	<10	[0..2]	<10	[60-100]	-	[30-50]	"Barely Habitable"
3	-	-	-	-	-	-	"Uninhabitable"

17. 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.

18. 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.

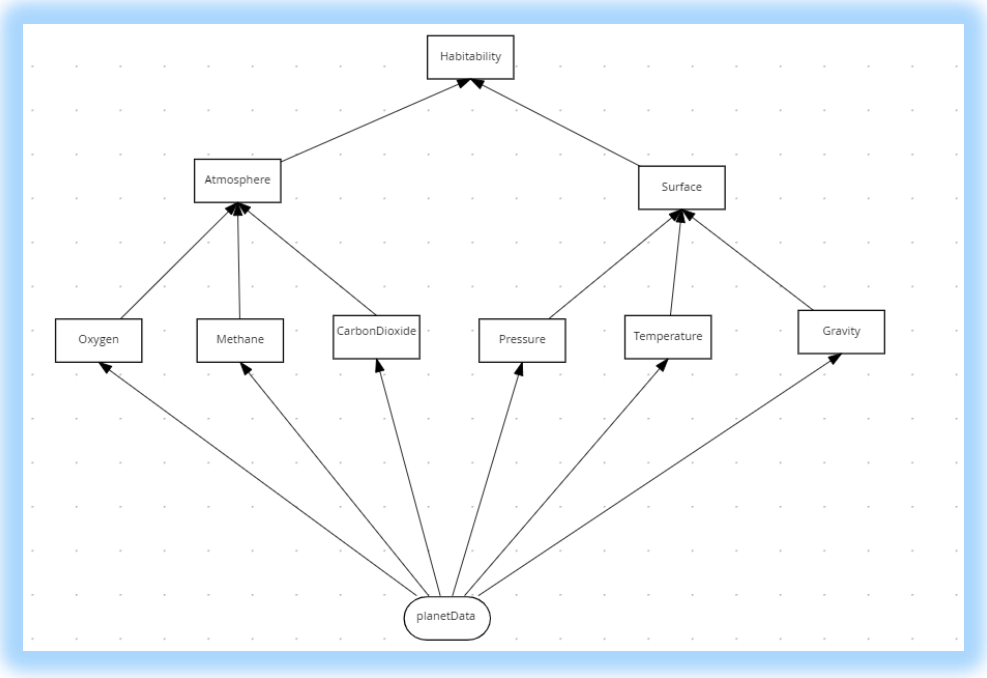
Lab 2 – The Divide and Conquer Pattern

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.

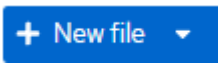
To do this, we create a decision table for each planet attribute and reduce to an enumerated value: *Optimal, Bearable* or *Deadly*. Then we apply the enumerated values in a decision further up in the model. Using this pattern, we can split a wide table into several smaller manageable tables.

The tables are linked in the DMN diagram shown below:



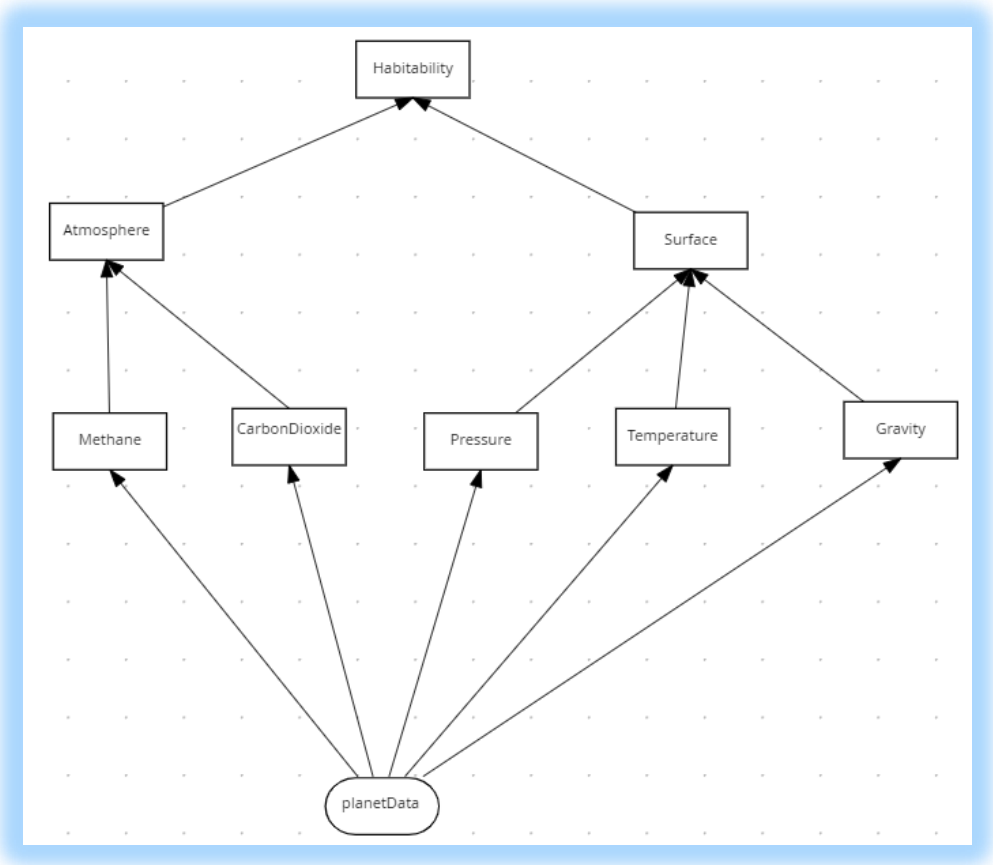
The **habitability** table has the same behavior as the original design, with the benefit that it is easier to maintain. Each attribute has its own table making it easier to change.

Instructions

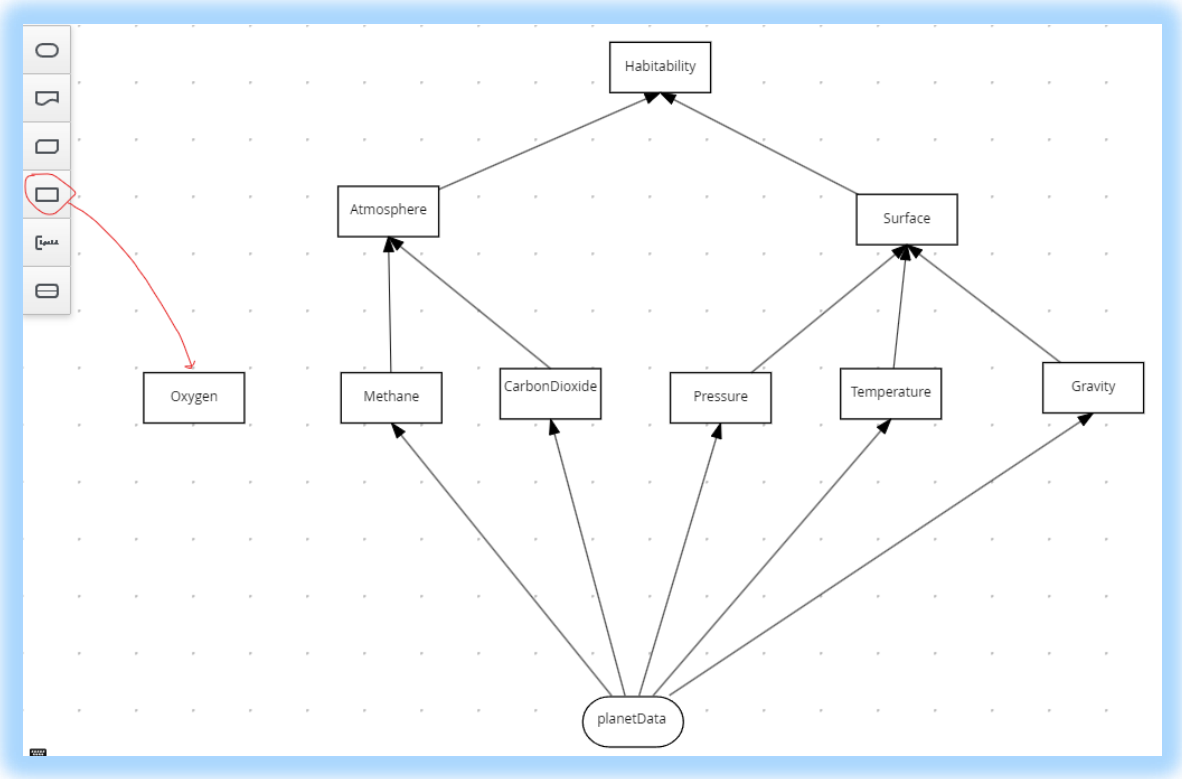
1. In browsers Chrome or Safari open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file* 

and then U*pload...*
4. Select the file: *lab02/HabitabilityStart.dmn*

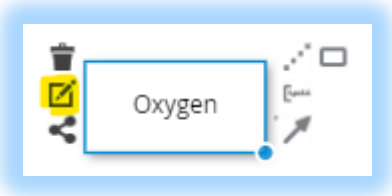
5. You should see this:



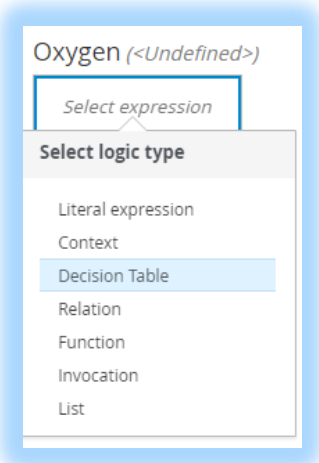
The model is nearly complete but is missing the *oxygen* attribute. Create a new Decision node and call it *Oxygen*.



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



Select *Decision Table* as the expression type:



Create the table below.

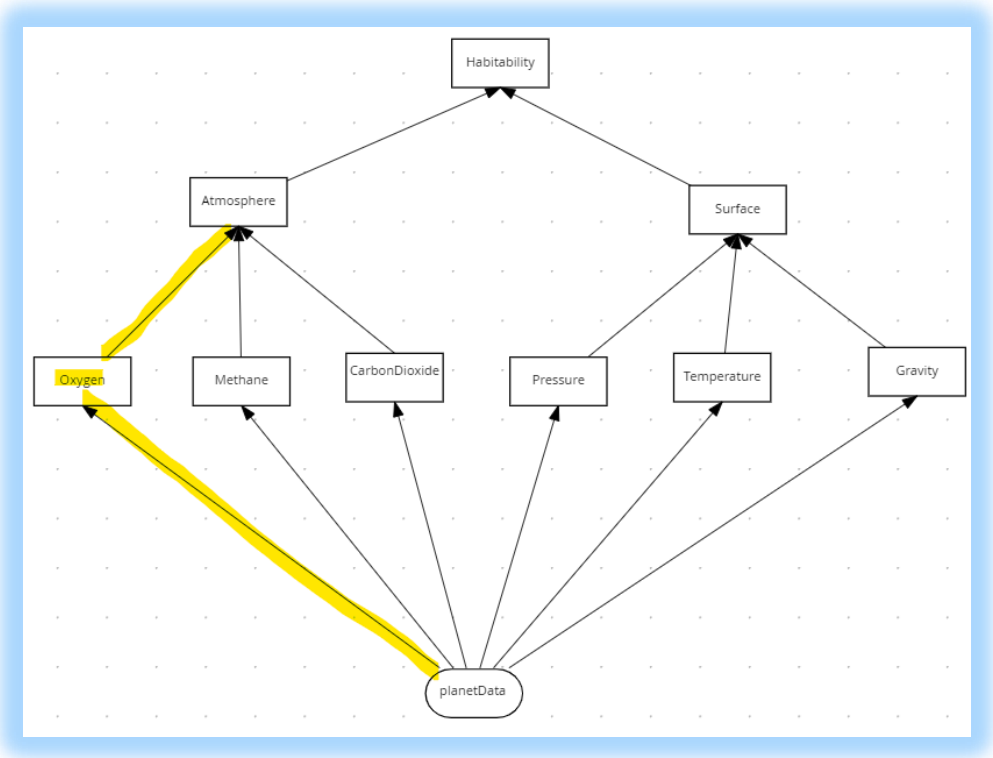
Oxygen *(Decision Table)*

F	planetData.atmosphere.oxygen *(number)*	Oxygen *(string)*
1	`[16..49]`	`"Optimal"`
2	`[5..15]`	`"Bearable"`
3	-	`"Deadly"`

Verify the following:

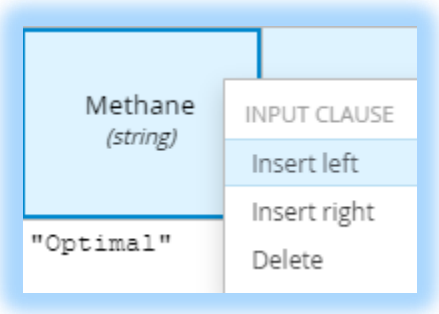
- `planetData.atmosphere.oxygen` is the condition variable
- Numeric ranges `[16..49]` and `[5..15]` are the conditions.
- Enumerated values are in quotes.
- The Hit policy is First (F)

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

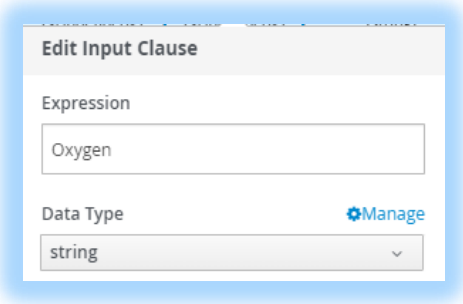


Edit the Atmosphere table so that it handles the Oxygen decision.

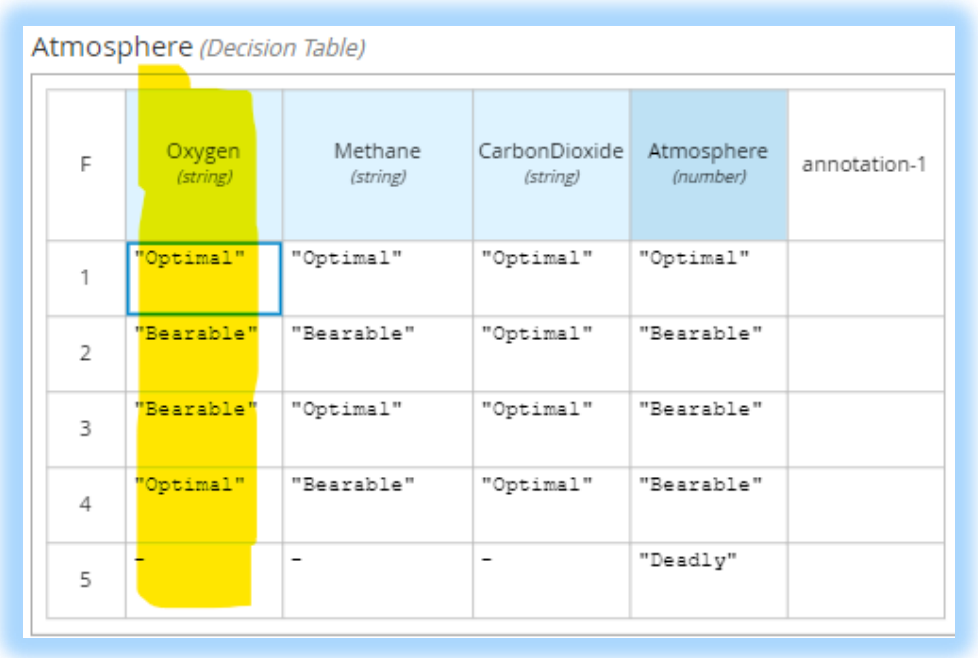
Right-click on the *Methane* column to add a new column to the left:



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:



F	Oxygen (string)	Methane (string)	CarbonDioxide (string)	Atmosphere (number)	annotation-1
1	"Optimal"	"Optimal"	"Optimal"	"Optimal"	
2	"Bearable"	"Bearable"	"Optimal"	"Bearable"	
3	"Bearable"	"Optimal"	"Optimal"	"Bearable"	
4	"Optimal"	"Bearable"	"Optimal"	"Bearable"	
5	-	-	-	"Deadly"	

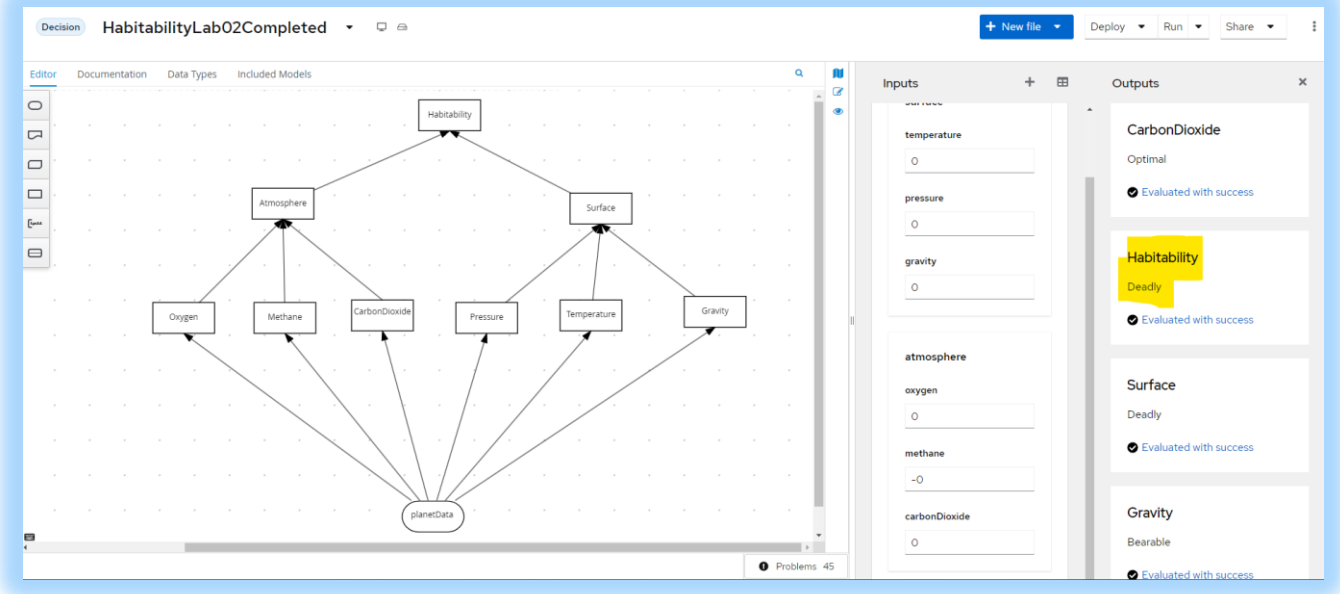
Now look at the root decision node, *Habitablilty*. It combines the decision in the lower tables to make the final decision as to the planet habitability:

Habitability *(Decision Table)*

F	Atmosphere *(string)*	Surface *(string)*	Habitability *(string)*	annotation-1
1	"Optimal"	"Optimal"	"Habitable"	
2	"Bearable"	"Bearable"	"Bearly Habitable"	
3	"Bearable"	"Optimal"	"Bearly Habitable"	
4	"Optimal"	"Bearable"	"Bearly Habitable"	
5	-	-	"Deadly"	

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.

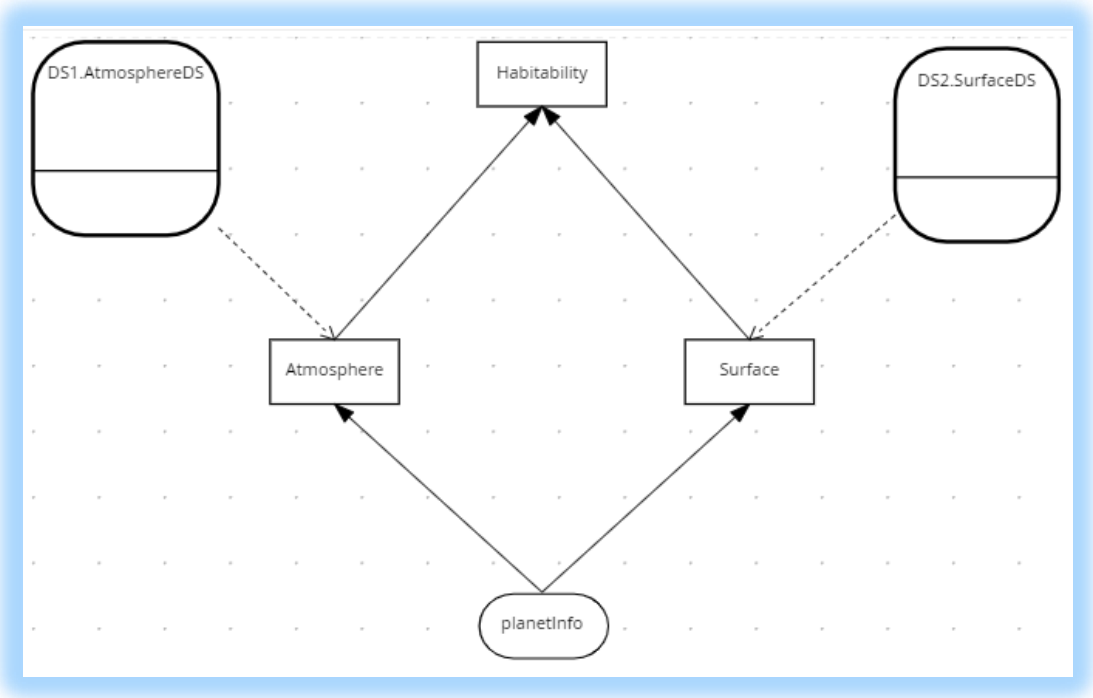
Lab 3 – The Tiered Service Pattern

Introduction

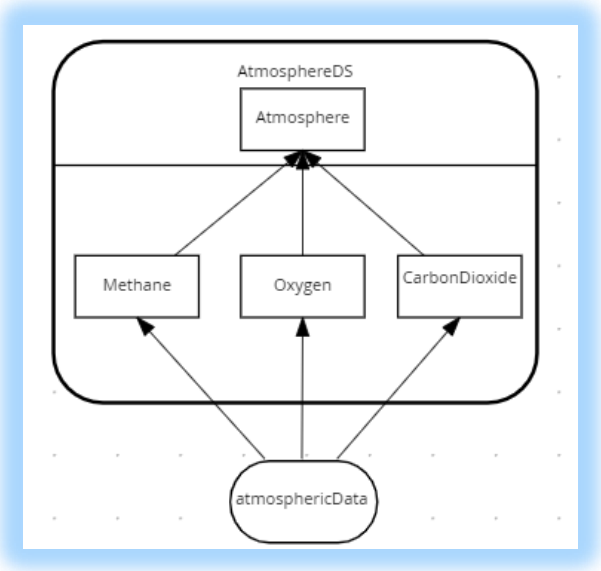
The DMN defined in the previous lab is confined to a single file. This is a problem:

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

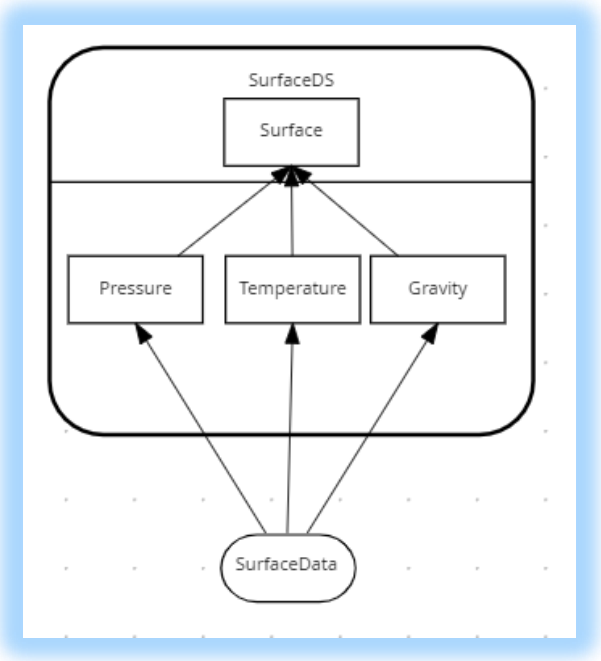
To improve, we move the second-tier decisions into their own decision services and then invoke these decisions from the top tier. See below, where the second-tier decision services are *AtmosphereDS* and *SurfaceDS*:



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



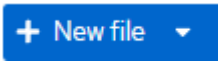
Atmosphere Decision Service



Surface Decision Service

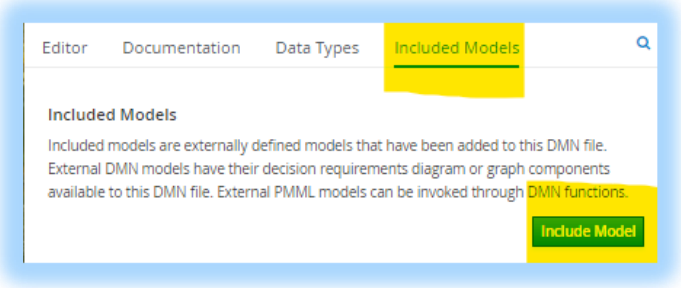
In the Habitability example we have only two tiers – but in practice the pattern could be applied to multiple tiers, with second tier decision services calling third tier services, and so on. The number of tiers depends on project complexity.

Instructions

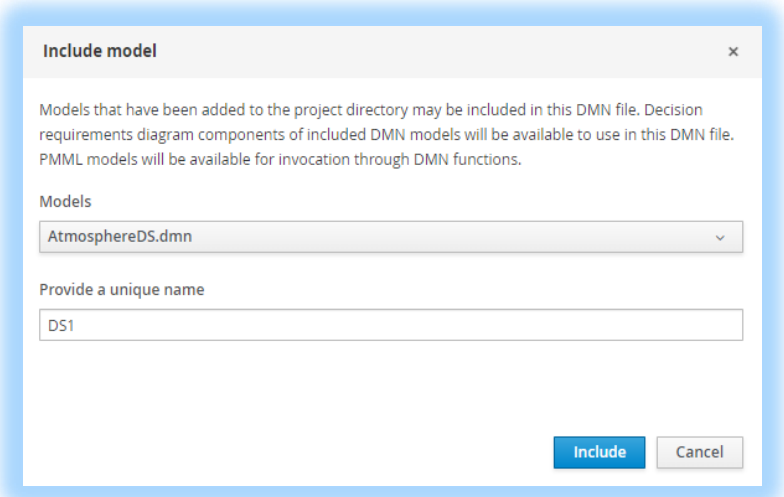
1. In browsers Chrome or Safari open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file* 

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

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

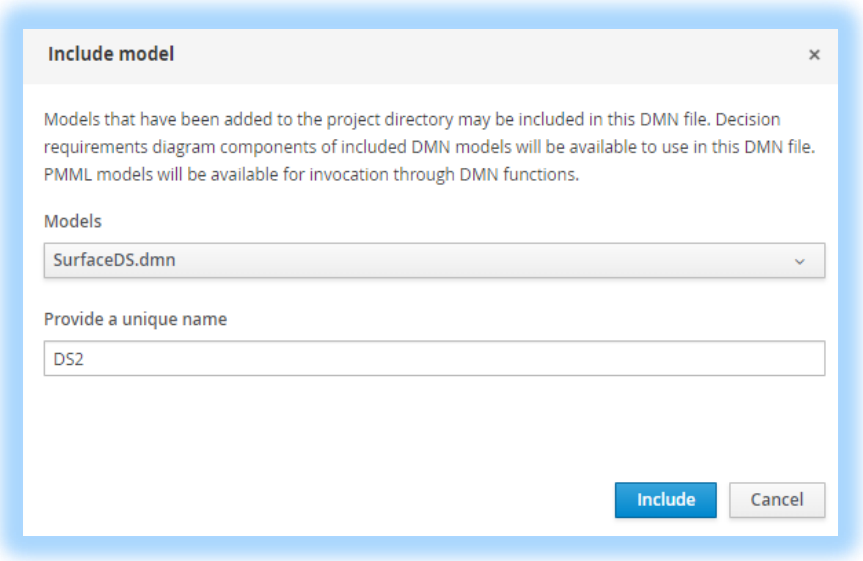


20. Add the *AtomosphereDS.dmn* and give it the name of *DS1:*

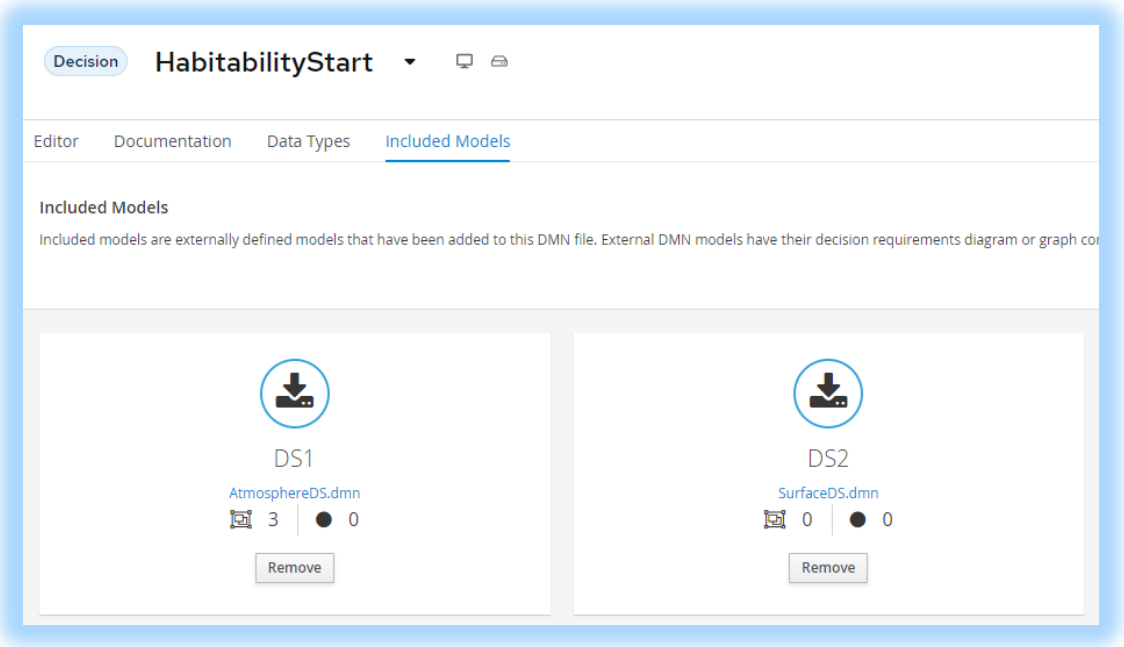


21. Click *Include.*

22. Click *Include Model* again and import *SurfaceDS.dmn.* Give it a name of *DS2:*

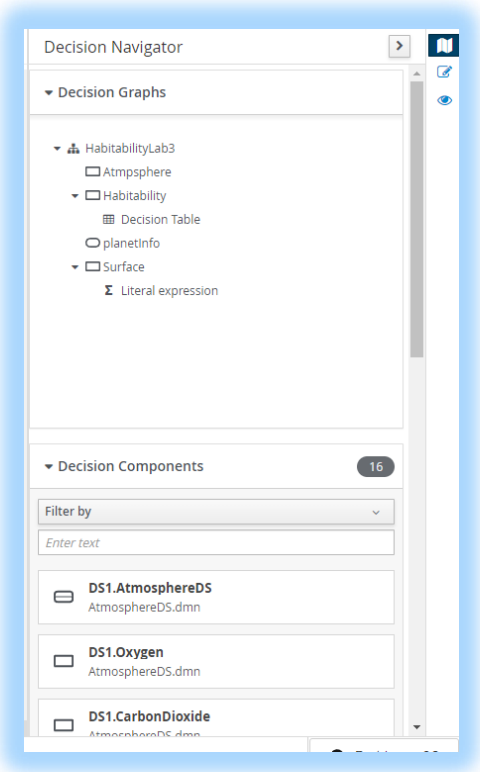


23. You should see both decision services have been imported:



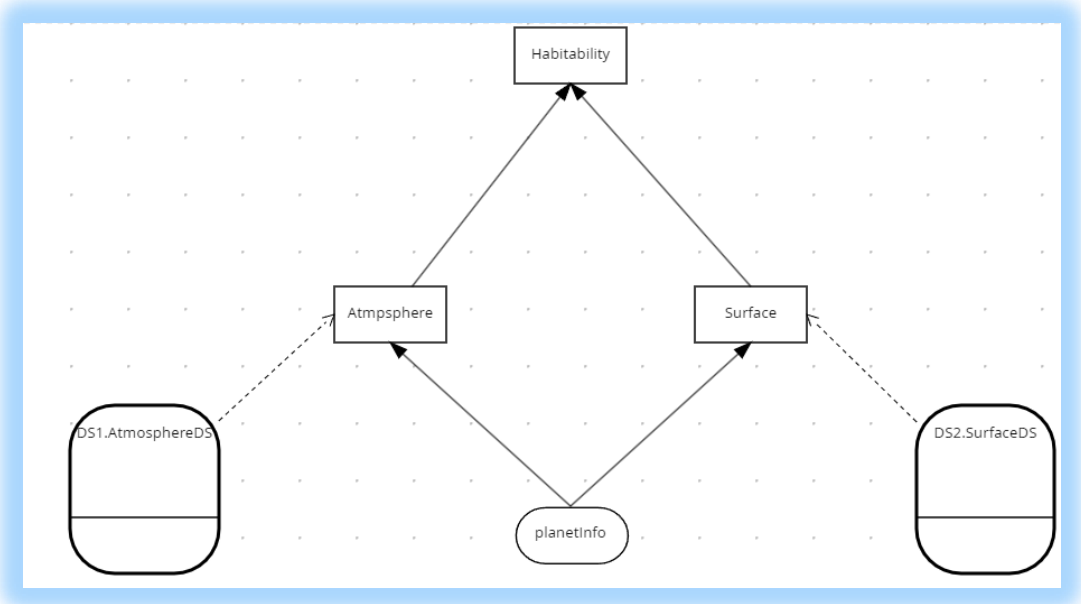
24. Select the *Editor tab.*

25. Select the Decision Navigator button on the far right . You should see the following:



26. Drag and drop *DS1.AtmosphereDS* and *DS2.SurfaceDS* to the diagram

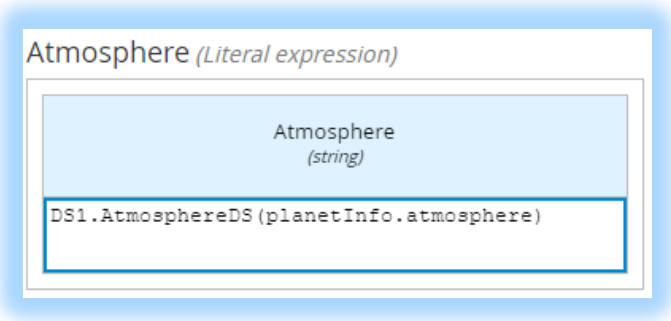
27. Using the arrow connector, connect *DS1.AtmosphereDS* to *Atmosphere* and *DS2.SurfaceDS* to *Surface*. You should now see the following:



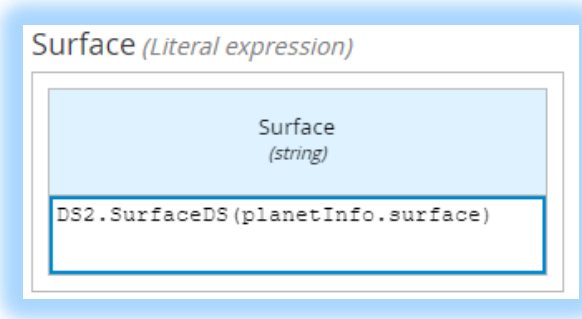
28. Edit the Atmosphere Decision Node. Create a Literal Expression and the FEEL expression:

```
DS1.AtmosphereDS(planetInfo.atmosphere)
```

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



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



31. Press the *Run* button

32. 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

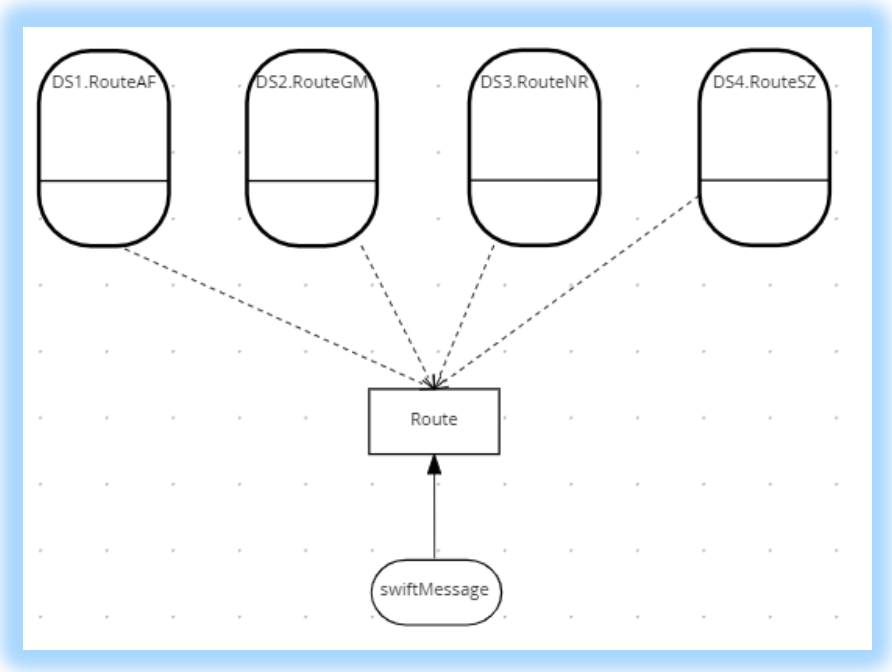
The rules below decide where to route a SWIFT message within the banking network:

Route *(Decision Table)*

U	SwiftMessage.bic *(string)*	SwiftMessage.branch *(string)*	SwiftMessage.messageType *(string)*	Route *(string)*	annotation-1
1	"AAA"	"001"	"202"	"X"	
2	"BBB"	"001"	"201"	"Y"	
3	"CCC"	"002"	"203"	"Z"	

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:



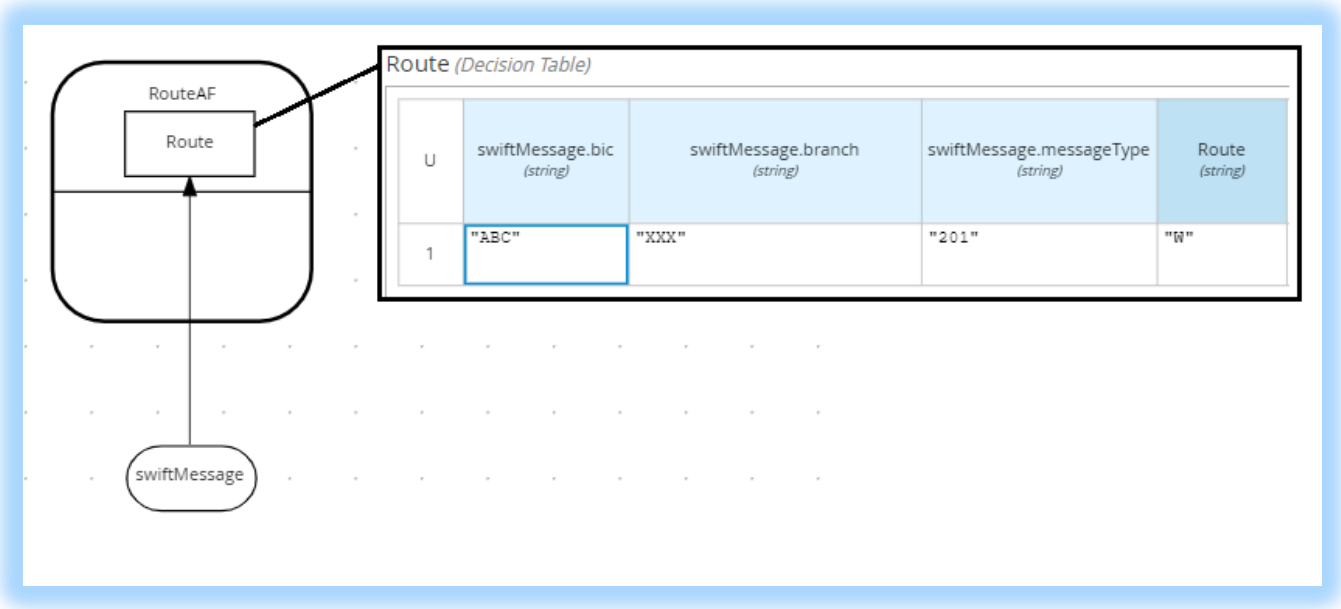
The key logic is the *Route* decision. It routes the message to an indexed decision service based on the first letter of the BIC. This splits the table into four smaller tables, each alphabetically indexed. There could be further indexing within each of these decision services to split the decision table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

Here is the logic for the Route decision:

Route *(Context)*

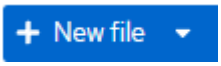
#		Route *(string)*
1	A-F *(boolean)*	`matches(swiftMessage.bic,"[A-F][A-Z]+" )`
2	G-M *(boolean)*	`matches(swiftMessage.bic,"[G-M][A-Z]+" )`
3	N-R *(boolean)*	`matches(swiftMessage.bic,"[N-R][A-Z]+" )`
4	S-Z *(boolean)*	`matches(swiftMessage.bic,"[S-Z][A-Z]+" )`
5	resultA-F *(string)*	`if A-F then (DS1.RouteAF(swiftMessage)) else ""`
6	resultG-M *(string)*	`if G-M then (DS2.RouteGM(swiftMessage)) else ""`
7	resultN-R *(string)*	`if N-R then (DS3.RouteNR(swiftMessage)) else ""`
8	resultS-Z *(string)*	`if S-Z then (DS4.RouteSZ(swiftMessage)) else ""`
	<result>	`string join ([resultA-F, resultG-M, resultN-R, resultS-Z])`

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.

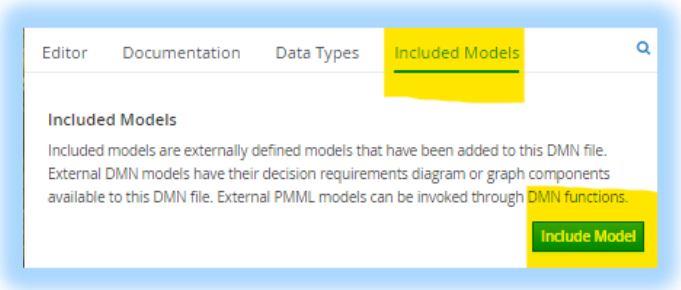
Instructions

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. Perform a multiple select on the Lab04 files:

   lab04/SwiftDataStructure.dmn
   lab04/RouteAF.dmn
   lab04/RouteGM.dmn
   lab04/RouteNR.dmn
   lab04/RouteSZ.dmn
   lab04/SwiftRoutingRules.dmn

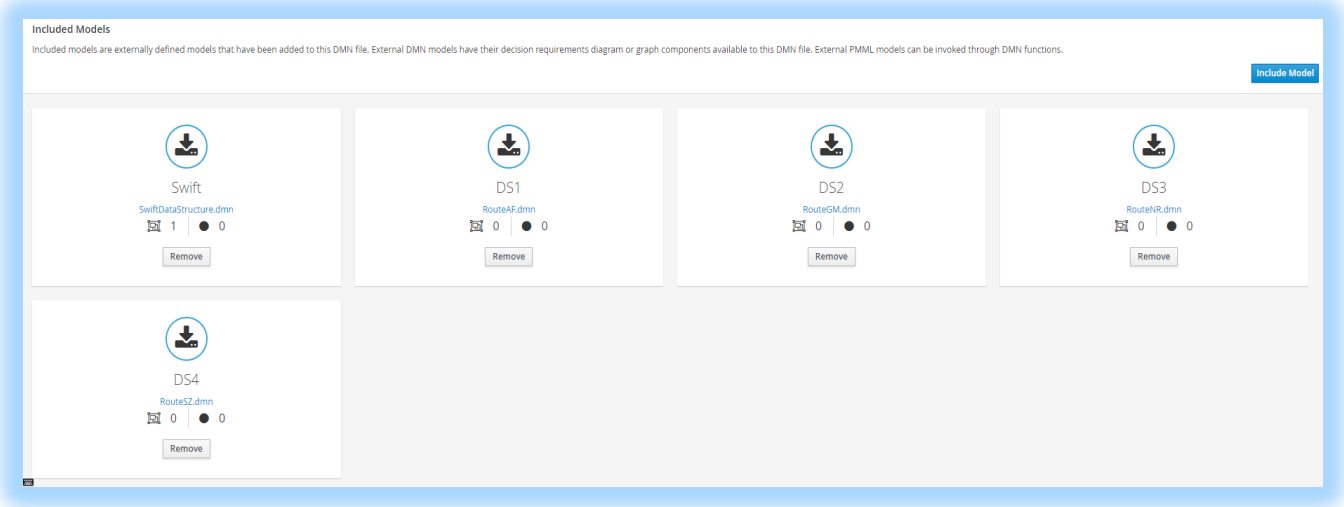
5. Within *SwiftRoutingRulesStart*, select **Included Models** tab, and then click **Include Model**



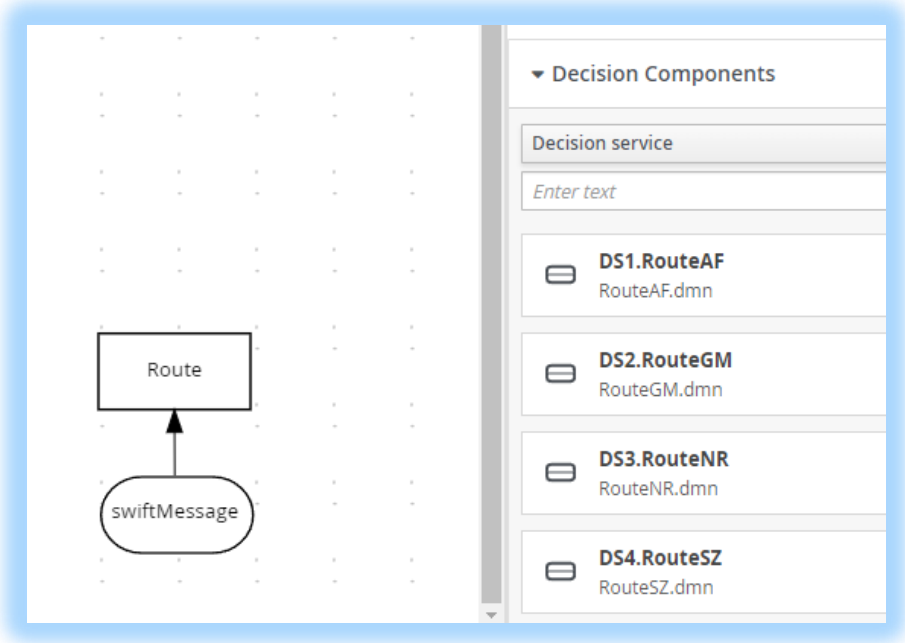
6. Include the following DMN Files as model:

DMN File	Name
SwiftDataStructure.dmn	**Swift**
RouteAF.dmn	**DS1**
RouteGM.dmn	**DS2**
RouteNR.dmn	**DS3**
RouteSZ.dmn	**DS4**

7. Once you have done this, you should see the following models included:

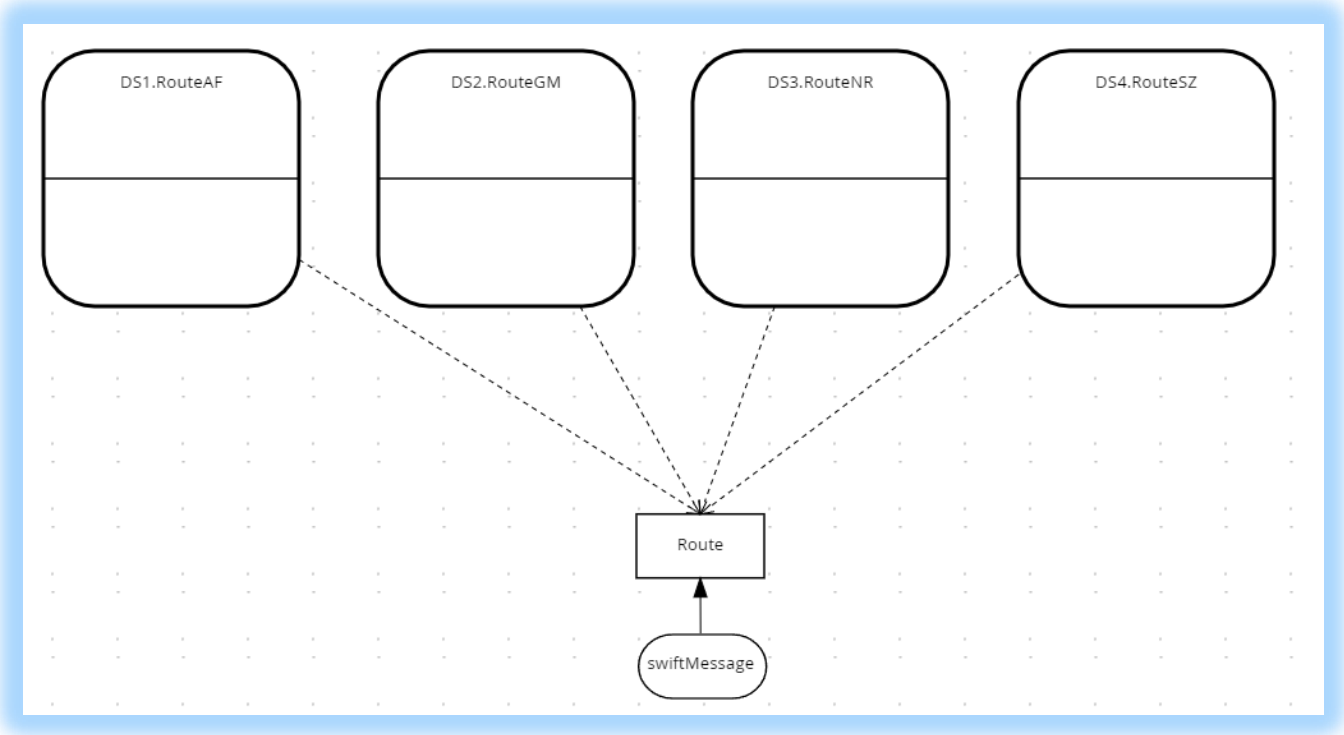


8. Select the *Editor tab.*
9. Select the Decision Navigator button on the far right .
10. Filter by Decision Service. You should see the following:

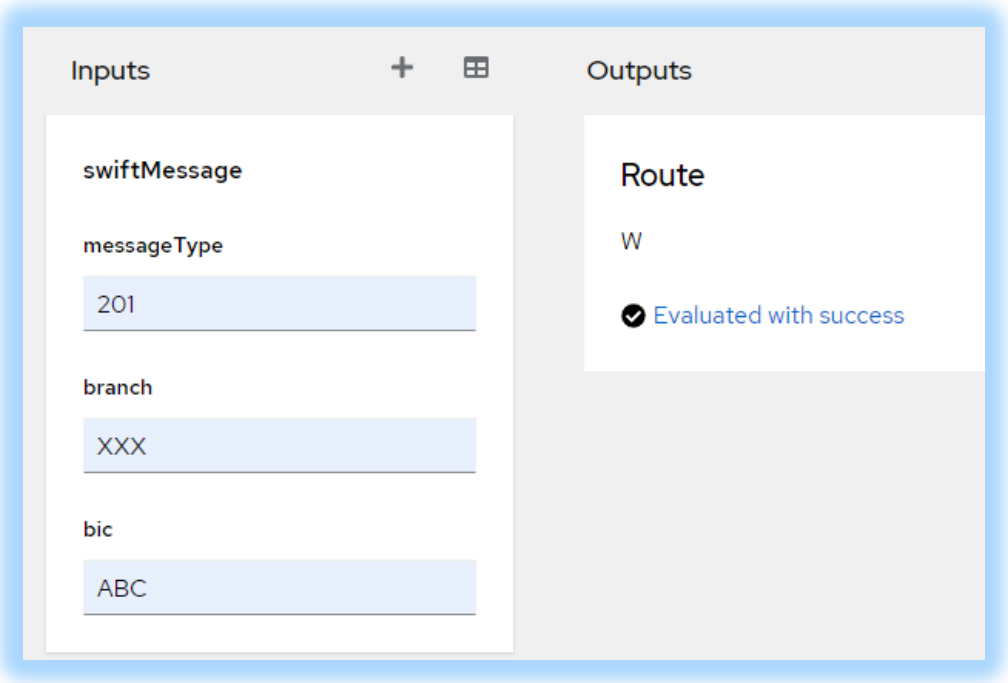


11. Drag and drop all four decision services to the diagram

12. Using the arrow connector, connect *the decision services to Route*. You should see the following:

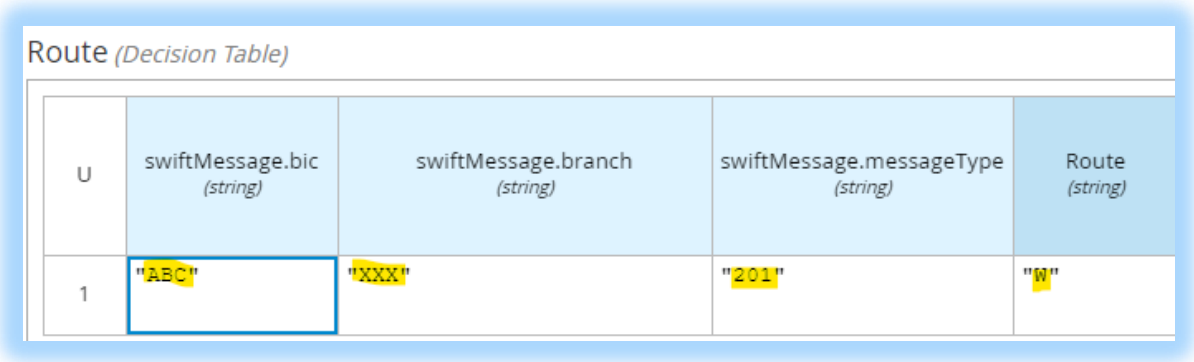


13. Test the decision service by pressing *Run*.
14. Enter the following values:



The output should be W. The Route decision determined the BIC started with 'A' and routed it to RoutAF. This decision service determines that a Swift message of ABC/XXX/201 is routed to W.

Verify this by selecting the RouteAF decision service and viewing the decision table.



Now test the following Swift messages:

Swift Message	Route
GBC/XXX/201	**X**
NBC/XXX/201	**Y**
SBC/XXX/201	**Z**

You should see that each message is routed to a different decision service.

Now add a new message: **ZBC/XXX/201** with route **Z1**. Where would you put this rule?

Conclusion

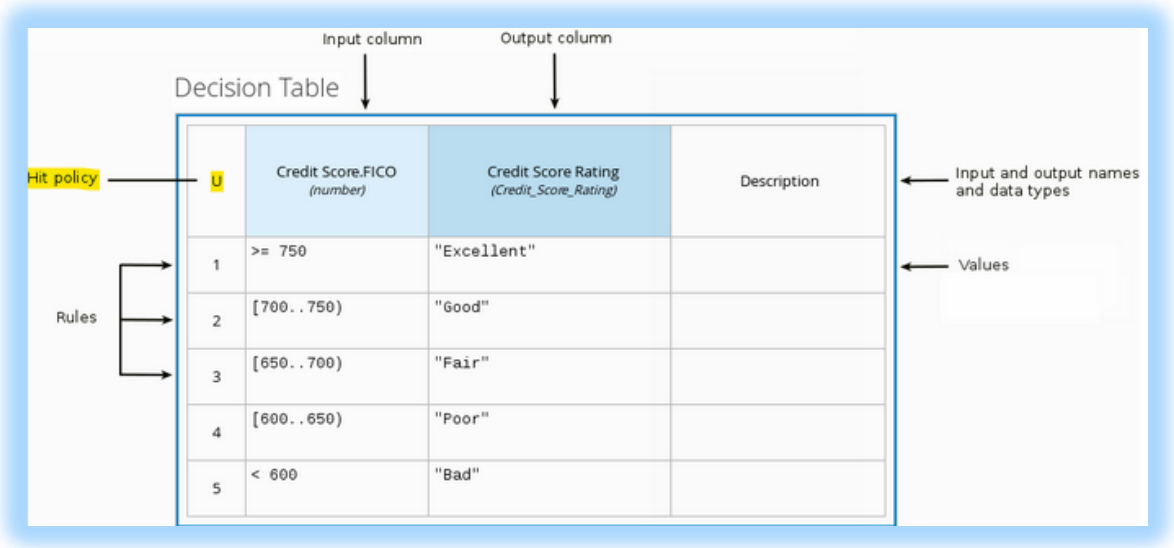
In this lab we split a tall narrow decision table into more manageable chunks using the Index Pattern.

Lab 5 - Hit Policies

Introduction

Decision tables do not always behave the same way. Some tables count outcomes, some are go / no go decisions and others require precise reasoning. Different hit policies not only lead to different results, but also require different ways of thinking about the entire table.

In DMN the hit policy is specified in the top left of the decision table. See below:



It is important to select the appropriate hit policy before entering rows into your decision table. The commonly used policies are:

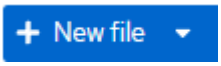
Hit Policy	Description	When to use
Unique (U)	Permits only one rule to match. Any overlap raises an error.	For detailed reasoning. Ensures your rules cover all cases and are complete
Any (A)	Permits only one rule to match. But allows overlaps.	As above but less strict enforcement of overlaps
First (F)	Rules are evaluated from top to bottom. Rules may overlap, but only the first match counts.	For concise decision tables where a simple go / no go decision is needed rather than complete reasoning.
Collect (C)	Aggregates values in an arbitrary list.	For multiple row matches. **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

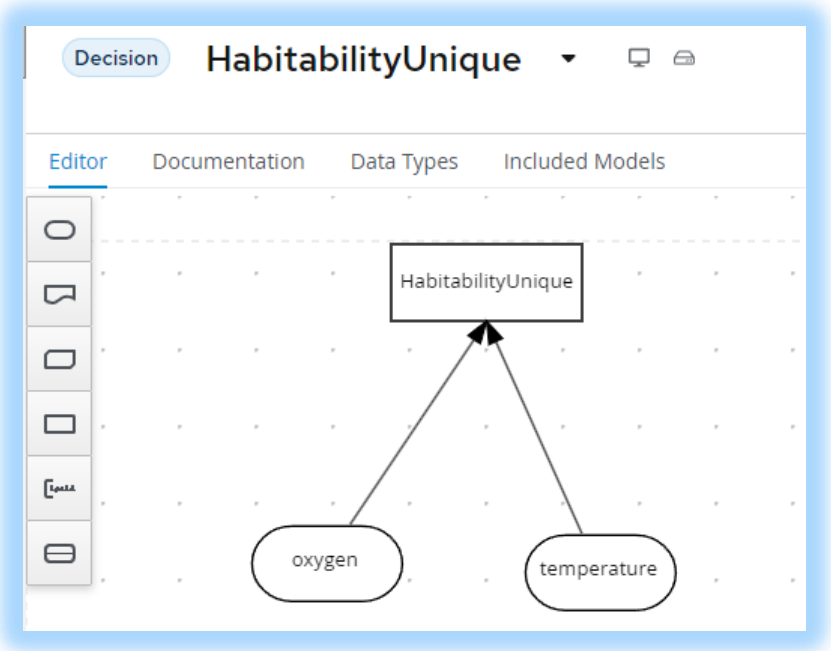
There are additional hit policies not covered in this lab. **Priority** which behaves like **Any**, and **Rule Order** and **Output Order** which behave like **Collect.**

Instructions

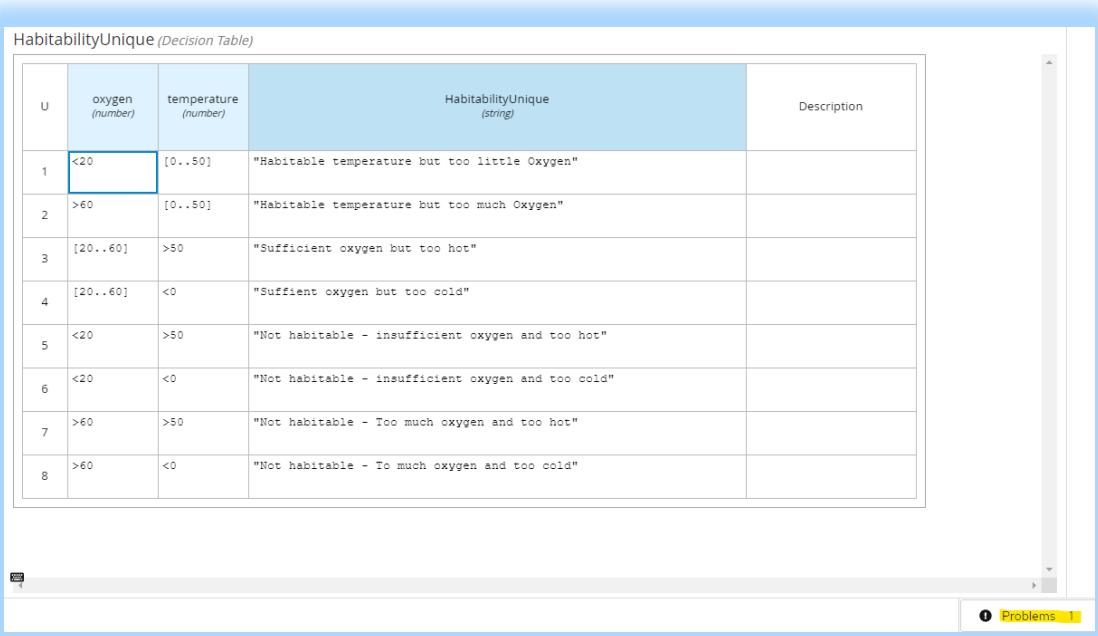
Unique Policy

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* 

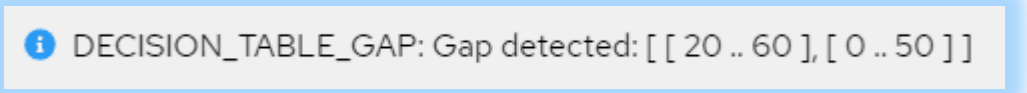
and then U*pload...*
4. Select the file: *lab05/HabitabilityUnique.dmn*. You should see this:



5. Edit the *HabitabilityUnique* decision by clicking it and selecting Edit .
6. Delete the first row by right clicking row 1 and select *Delete*.
7. After deleting the first row you should see this:

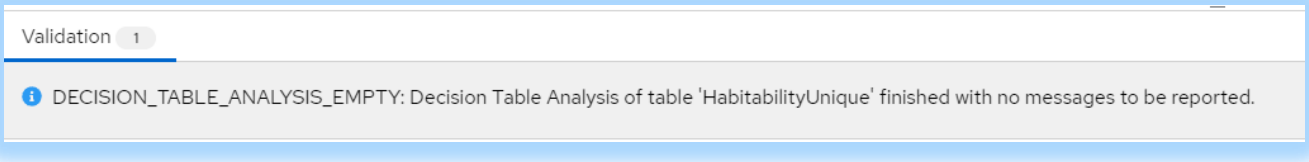


8. Click in the ❗ *Problems* button in the bottom right. There is a gap warning:

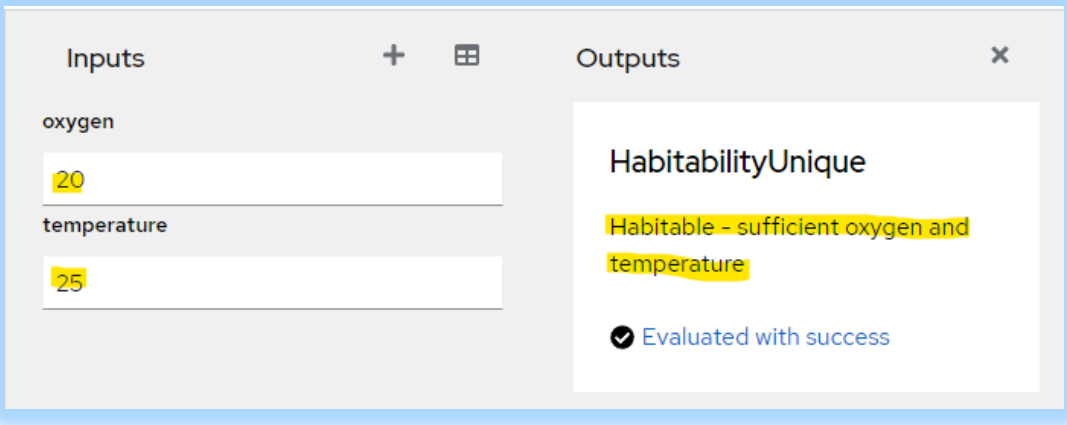


9. The gap created from the deleted row has been detected.

10. Fix this error by hitting **Ctrl-Z** to undo the change. The deleted row should reappear, and the analysis warning should disappear. You should see:



11. Test by pressing R*un*. Enter Oxygen *20* and Temperature *25*:



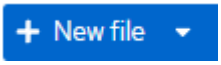
12. The decision correctly evaluates to "*Habitable - sufficient oxygen and temperature*".

Conclusion

The *Unique* policy reasons over every possible input. This is useful for decisions requiring traceability. For example, an applicant my want to know the decision behind their rejected mortgage application. A second advantage of the Unique policy is that you can order rows in any order to get the same result.

The disadvantage of the Unique policy is that it can force the table to be more verbose than it needs to be.

Any Policy

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: lab05/*HabitabilityAny.dmn*
5. Edit the *HabitabilityAny* decision by clicking it and selecting Edit .
6. You should see this:

HabitabilityAny *(Decision Table)*

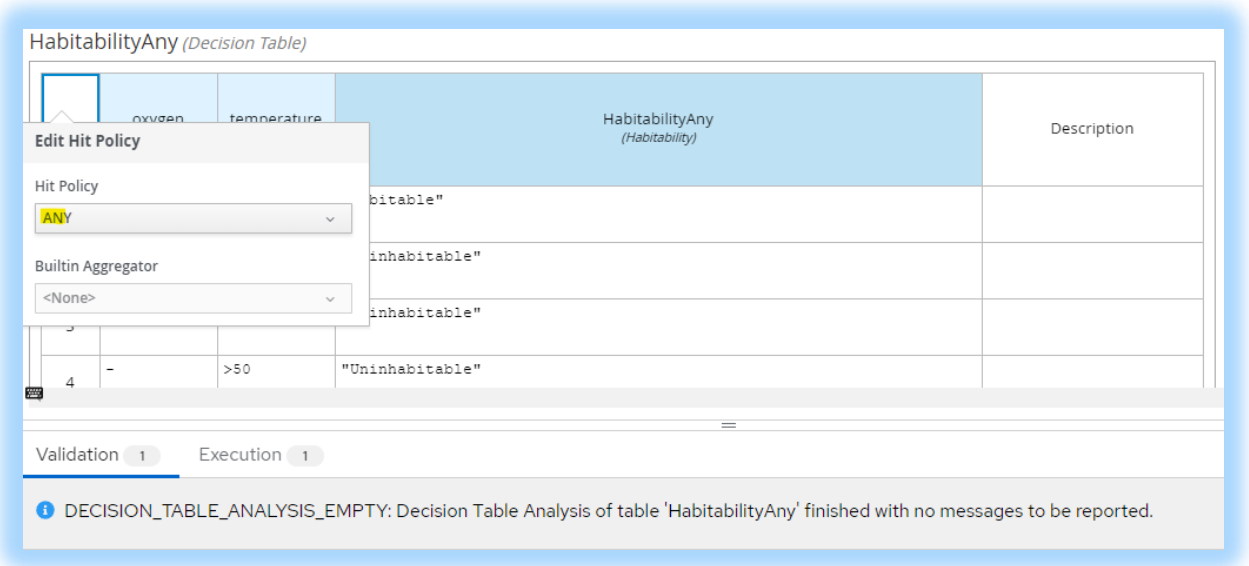
U	oxygen *(number)*	temperature *(number)*	HabitabilityAny *(Habitability)*
1	[20..60]	[0..50]	"Habitable"
2	<20	-	"Uninhabitable"
3	>60	-	"Uninhabitable"
4	-	>50	"Uninhabitable"
5	-	<0	"Uninhabitable"

7. Click in the *Problems* button in the bottom right. There are several gap warnings:

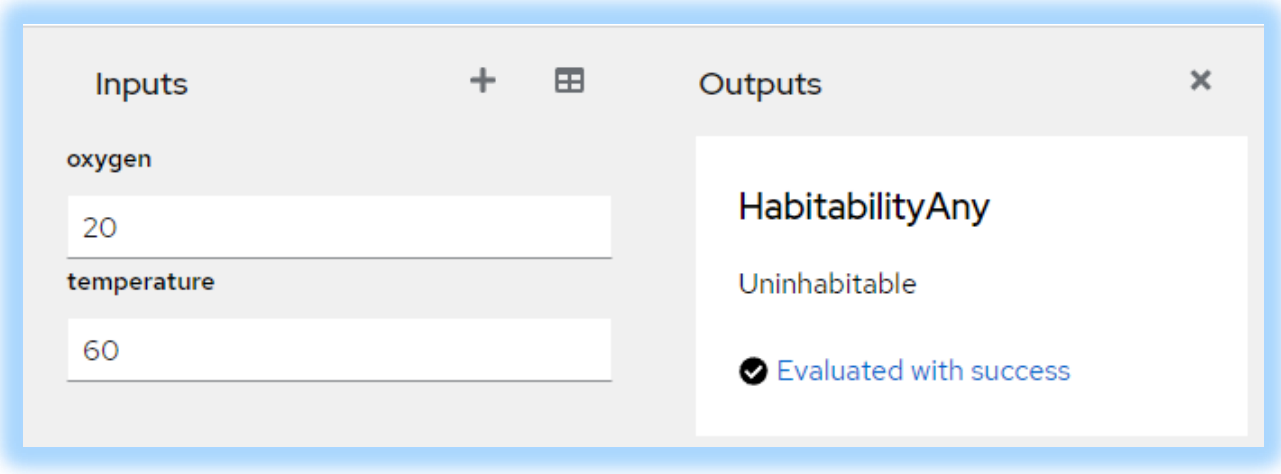
Validation 5

- DECISION_TABLE_HITPOLICY_RECOMMENDER: Overlapping rules have the same output value, so the HitPolicy for decision table 'HabitabilityAny' should be ANY
- DECISION_TABLE_OVERLAP_HITPOLICY_UNIQUE: Overlap detected: Overlap values: [<20, <0] for rules: [2, 5]. UNIQUE hit policy decision tables can only have one matching rule.
- DECISION_TABLE_OVERLAP_HITPOLICY_UNIQUE: Overlap detected: Overlap values: [<20, >50] for rules: [2, 4]. UNIQUE hit policy decision tables can only have one matching rule.
- DECISION_TABLE_OVERLAP_HITPOLICY_UNIQUE: Overlap detected: Overlap values: [>60, <0] for rules: [3, 5]. UNIQUE hit policy decision tables can only have one matching rule.
- DECISION_TABLE_OVERLAP_HITPOLICY_UNIQUE: Overlap detected: Overlap values: [>60, >50] for rules: [3, 4]. UNIQUE hit policy decision tables can only have one matching rule.

8. These errors a due to the Hit policy being **Unique** and there are overlapping rows. Fix these errors by changing the Hit Policy to **Any**. You should see the policy symbol change to **A** and all errors disappear:



9. Test by pressing R*un*. Enter Oxygen *20* and Temperature *60*:



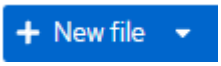
10. This tests row 4 of the decision table.

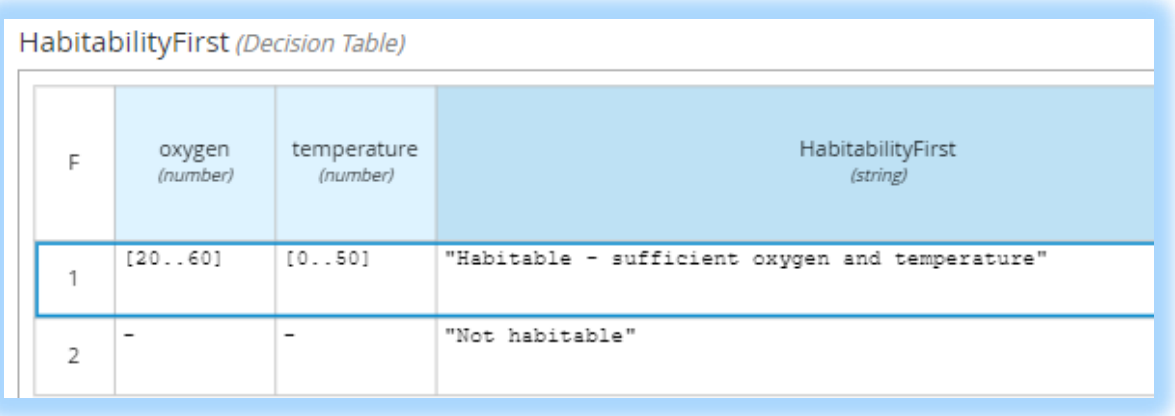
Conclusion

The advantage of the **Any** policy is you can create overlapping rules using "don't care" (-) values to make it more compact.

This advantage is also a disadvantage of imprecision; row 4 is hit whether oxygen is *habitable* or *uninhabitable*. This may be a problem if you need to know the exact reason as to why this planet is uninhabitable.

First Policy

11. In browsers Chrome or Safari open the web site https://sandbox.kie.org/
12. Click on **New Decision**.
13. An empty canvas opens. Click *New file*  and then U*pload...*
14. Select the file: lab05/*HabitabilityFirst.dmn*
15. Edit the *HabitabilityFirst* decision by clicking it and selecting Edit
16. You should see this:

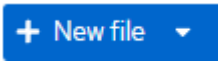


Test the decision by pressing *run*. Enter Oxygen *20* and Temperature *25* to check that row 1 fires.

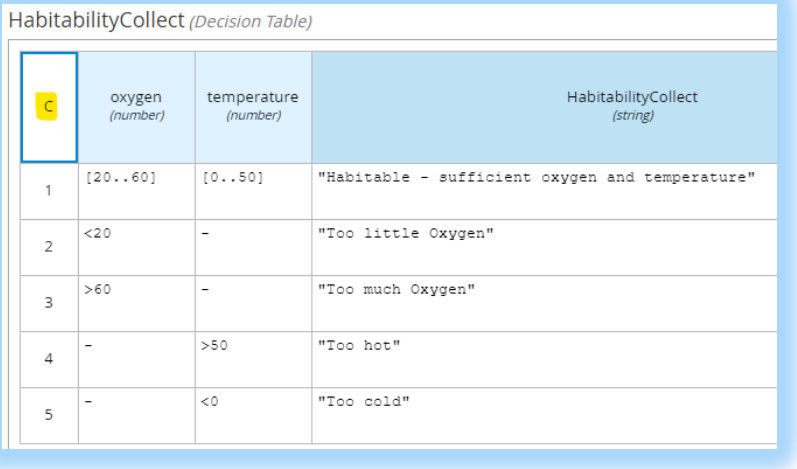
Conclusion

Using the **First** hit policy you get the same result as the *Unique* and *Any* Policies, but with fewer rows. There are just two rows, one for a habitable planet, and another for an uninhabitable planet. The advantage is conciseness, but there are disadvantages. The first is that order matters – you cannot move row 2 to row 1. The **First** policy also has similar problems as the **Any** policy in that the decision has no detailed reasoning. Finally, the policy is the least strict, allowing overlaps and missing rows which could cause problems at run time.

String Collection policy

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...*

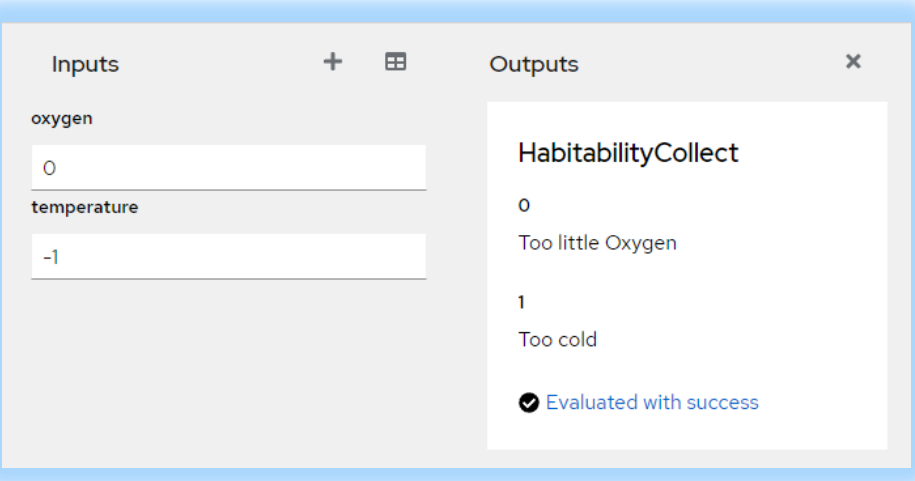
4. Select the file: lab05/*HabitabilityStringCollect.dmn*
5. Edit the *HabitabilityCollect* decision by clicking it and selecting Edit
6. You should see this:



HabitabilityCollect *(Decision Table)*

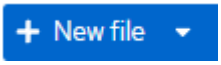
C	oxygen *(number)*	temperature *(number)*	HabitabilityCollect *(string)*
1	[20..60]	[0..50]	"Habitable - sufficient oxygen and temperature"
2	<20	-	"Too little Oxygen"
3	>60	-	"Too much Oxygen"
4	-	>50	"Too hot"
5	-	<0	"Too cold"

7. Test the decision by pressing *run*. Enter Oxygen *0* and Temperature *-1*. This time two rows are fired: row 2 and row 5. Both outputs are passed out of the decision as a list. Row 2 is passed as position 0 and row 5 as position 1 in the list.



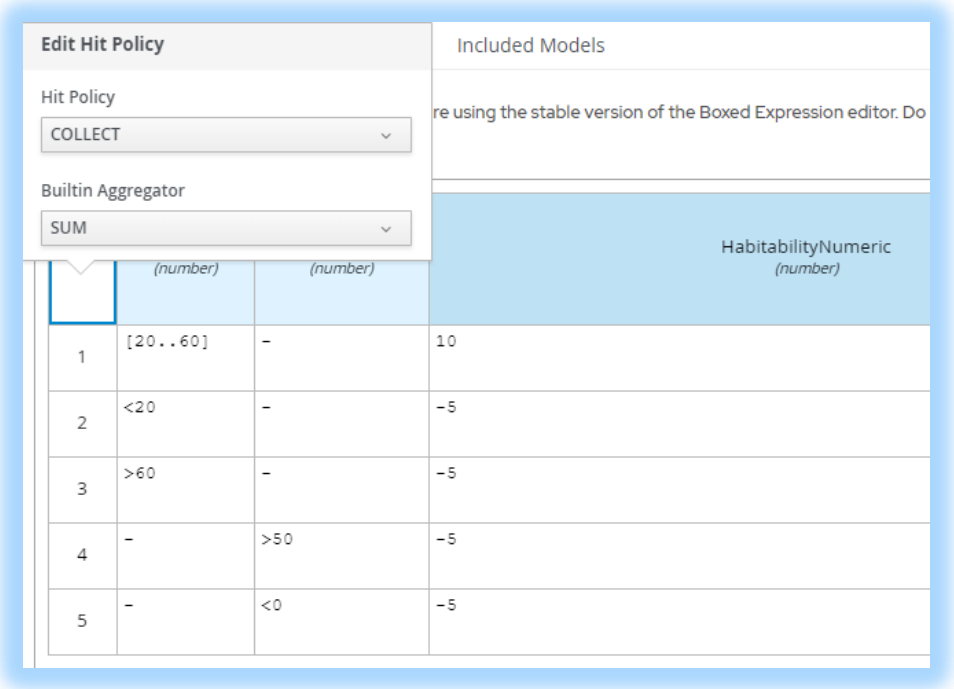
Conclusion
The Collection policy combines rows to make amalgamated decisions. This is useful when you require multiple rows in a decision table to be fired.

Numeric Collection policy

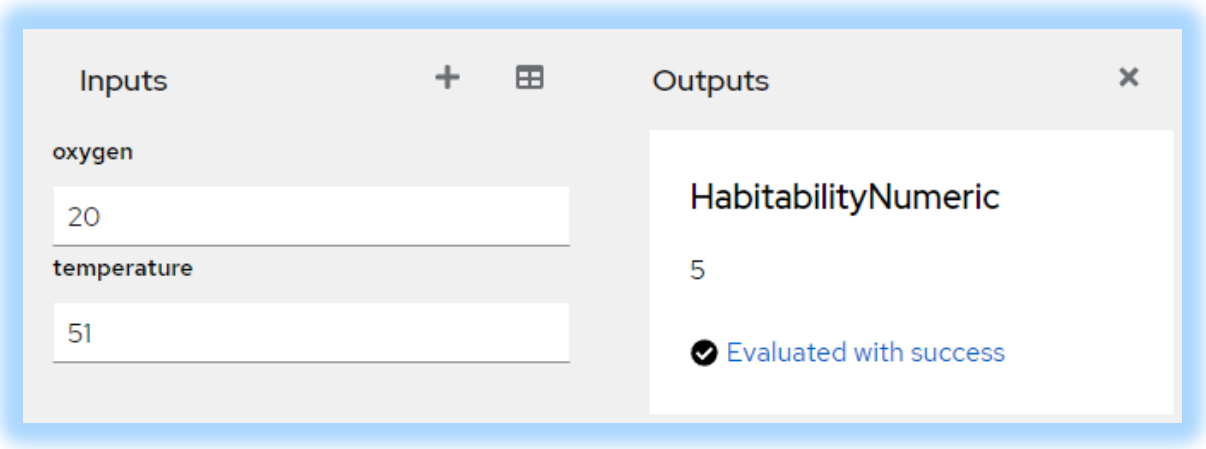
1. In browsers Chrome or Safari open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file*  and then U*pload…*

4. Select the file: lab05/*HabitabilityNumericCollect.dmn*

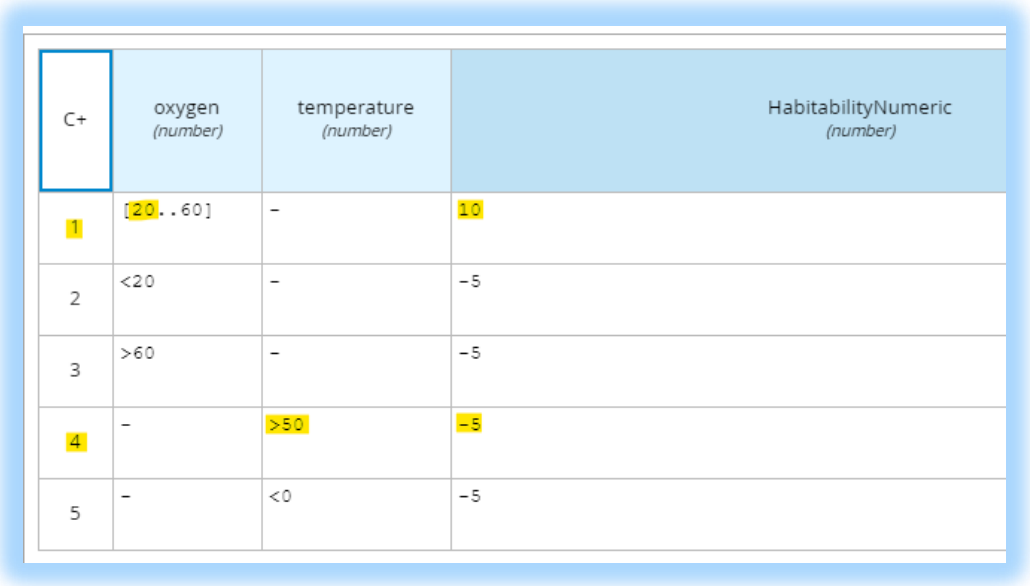
8. Edit the *HabitabilityNumeric* decision by clicking it and selecting Edit
9. You should see this:



10. In the top left of the table, select the hit policy. Click the *C+* symbol. This symbol expands to show the Hit Policy *Collect SUM*:
11. The *Collect SUM* hit Policy collects all the rows satisfying the input conditions and sums them together. Let's test this. Press *Run*. Enter Oxygen *20* and Temperature *51*:



The *collect sum hit policy* collects rows 1 and 4 and then adds them together resulting in 5. See workings highlighted below:



C+	oxygen *(number)*	temperature *(number)*	HabitabilityNumeric *(number)*
1	[20..60]	-	10
2	<20	-	-5
3	>60	-	-5
4	-	>50	-5
5	-	<0	-5

Now run the test with following Hit Policies:

Symbol	Hit Policy / Aggregator	Result	**Comment**
C#	Collect/Count	**2**	**Counts the hits**
C<	Collect/Min	**-5**	**Returns the lowest value**
C>	Collect/Max	**10**	**Returns the highest value**

See the test results change for each aggregator. What are the applications for each aggregator?

Conclusion

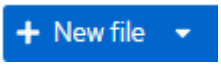
In this lab we looked at hit policies. The choice of hit policy depends on whether you need simple *go/no go* decisions, or comprehensive reasoning.

Lab 6 - Advanced DMN

Introduction

Decision Requirements Diagrams and Decision Tables are something business users can understand. The problem is that it is difficult to do anything useful with these, except provide requirements to developers. The real power of DMN comes with FEEL (Friendly Enough Expression Language). In this lab we will explore an advanced DMN example that uses FEEL to reschedule flights.

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: *Labs**Lab06**src**main**resources**Lab06*\FlightBooking.*dmn*
5. You should see the following:

A Quick Tour of the Flight Rebooking Service

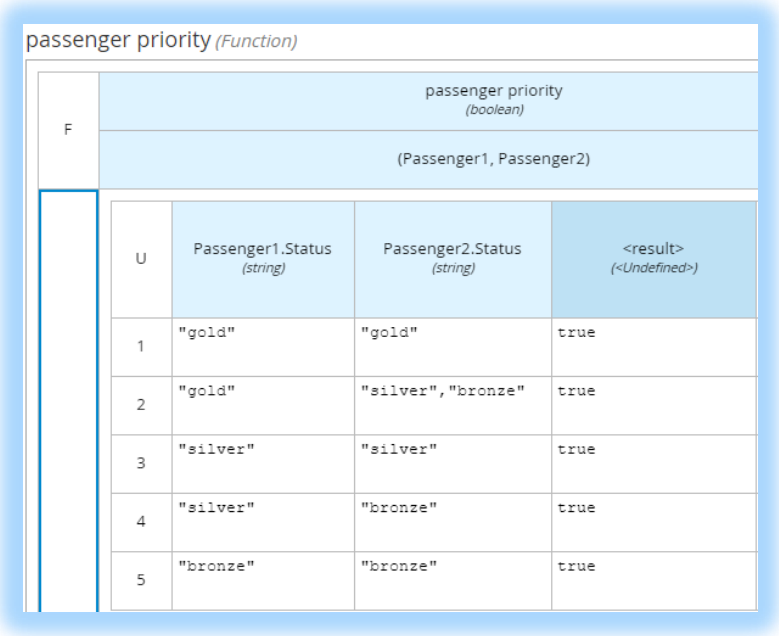
We will briefly examine the main components of the Flight Rebooking service.

Click on the *data types* tab. You should see:

- **tFlightTable** – This will list of flights with status of *cancelled* or *scheduled*
- **tPassengerTable** - A list of passengers with status of *gold, silver* or *bronze*

12. Back in the DMN diagram, edit the *passenger priority* by clicking it and selecting Edit

13. You should see this:



passenger priority *(Function)*

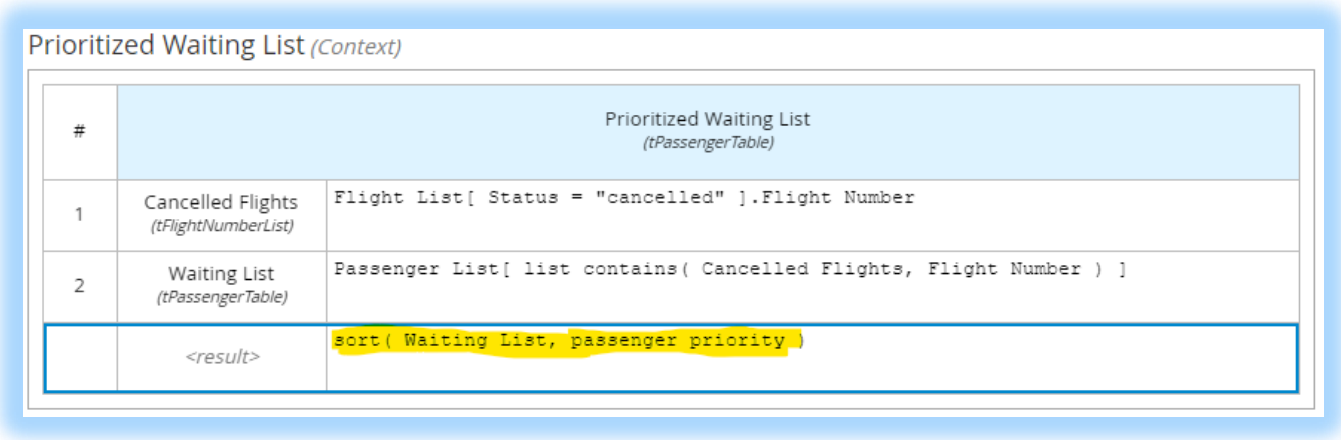
F	passenger priority *(boolean)*	
	(Passenger1, Passenger2)	

U	Passenger1.Status *(string)*	Passenger2.Status *(string)*	<result> *(<Undefined>)*
1	"gold"	"gold"	true
2	"gold"	"silver","bronze"	true
3	"silver"	"silver"	true
4	"silver"	"bronze"	true
5	"bronze"	"bronze"	true

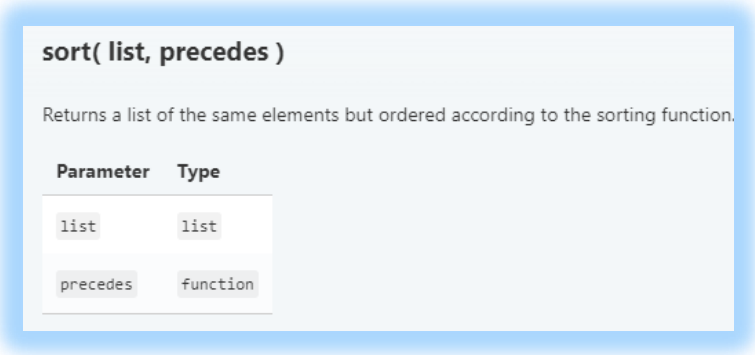
You will notice that it looks different to a standard decision table. It is a *Business Knowledge Model* which is a function invoked from a Decision. It takes parameters *Passenger1* and *Passenger2*.

1. Back in the DMN diagram, edit the *Prioritized Waiting List* by clicking it and selecting Edit

2. You should see this:



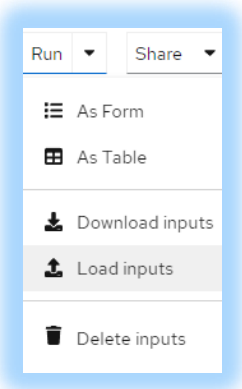
Note the highlighted *sort* function, which takes a list of waiting passengers and the *passenger priority function* described above. The *sort* function is a built in FEEL function and is shown below:



Full details found here https://kiegroup.github.io/dmn-feel-handbook/#sort-functions

Test the Service

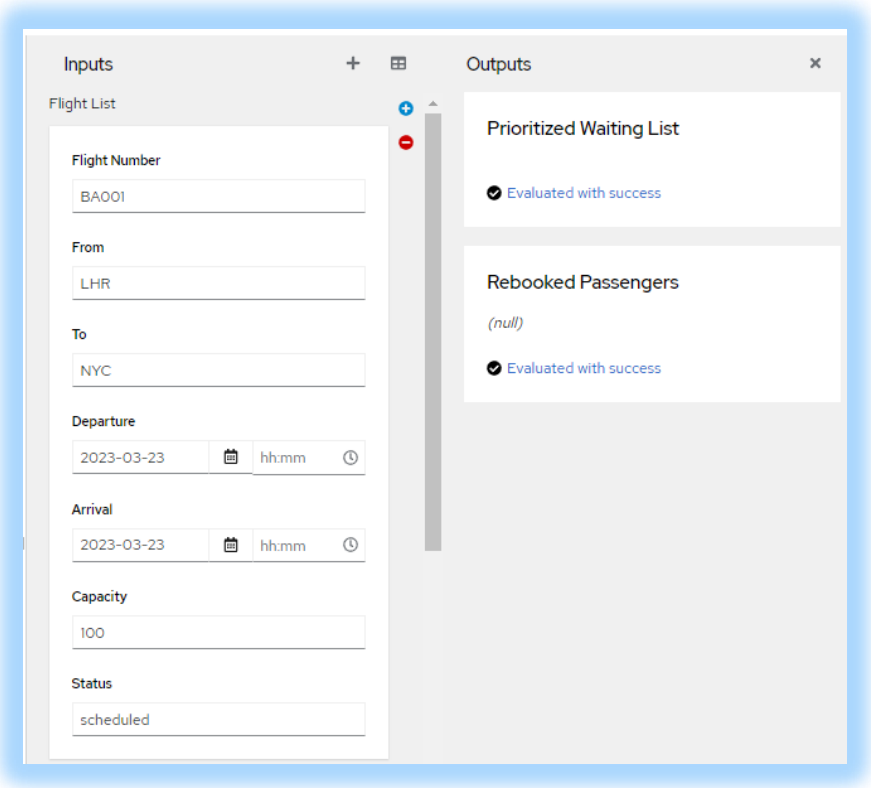
6. Load and run the provided test scenario. Click *Run->Load Inputs*



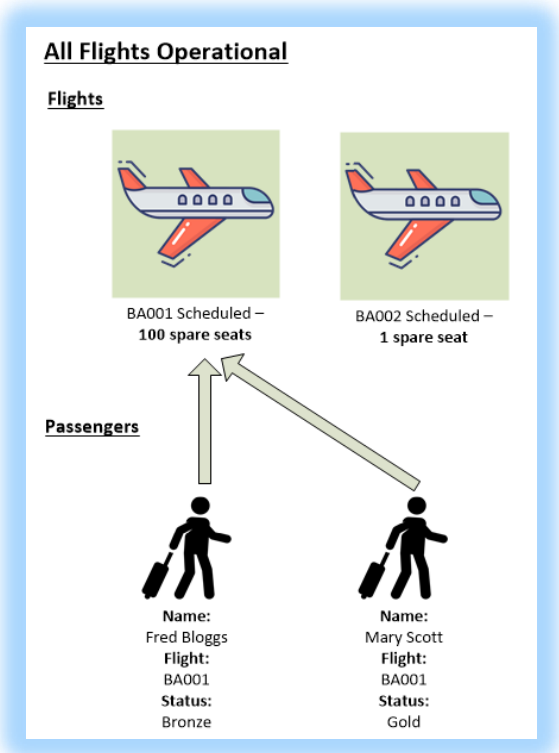
7. Open */Labs/Lab06/src/test/resources/FlightRebooking.json*

8. Press *Run*

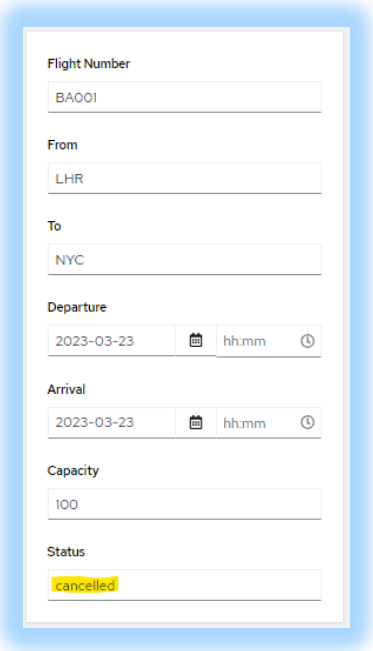
9. The service will run using the test data. You should see the data in the *Inputs* panel and the empty results in the *Outputs* panel:



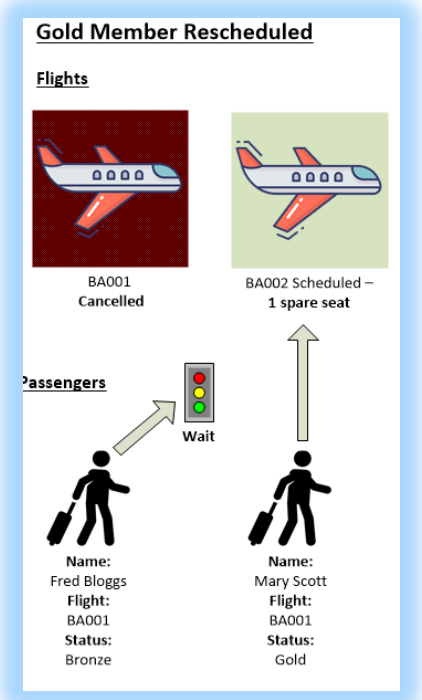
10. This is correct. The output contains no waiting list or rebooked passengers as all flights are operational. This is represented below:



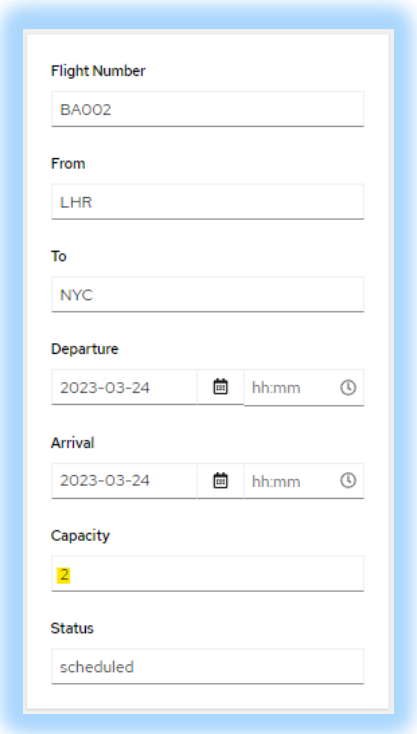
11. In the *Flight List*, change the status of flight BA001 to *cancelled* (all lower case):



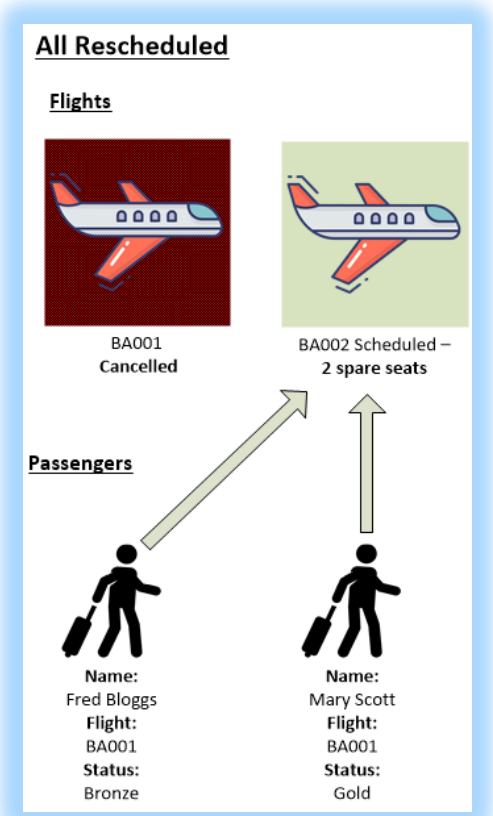
12. The results change so show gold member *Mary Scott* is rebooked onto flight *BA002*, and *bronze* member *Fred Bloggs* must wait:



13. In the *Flight List*, change the spare seat capacity on flight *BA002* to *2*:



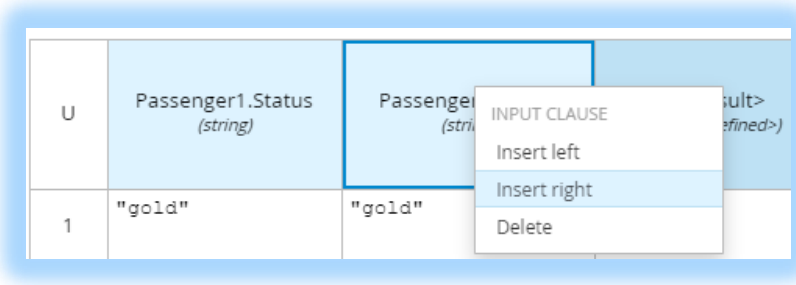
14. Now both Mary and Fred are scheduled to fly on *BA002*:



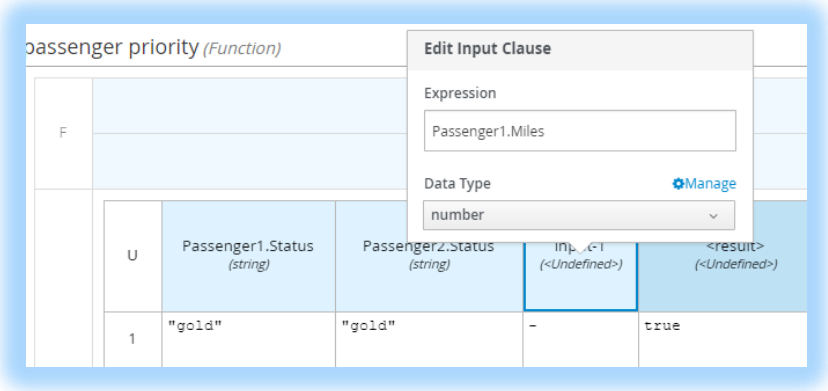
Extend the diagram

15. Extend the diagram so that it handles air miles as well as Gold/Silver/Bronze status. Select and then edit *passenger priority*.

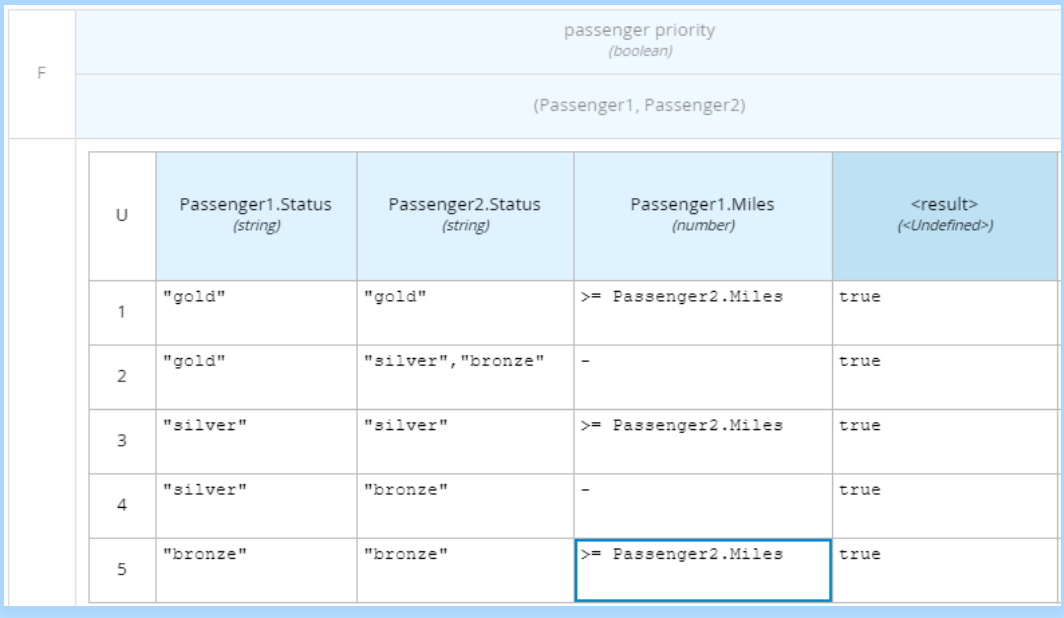
16. Add a new column in the table. Right-click the *Passenger2.Status* column and then select *Insert right* to insert a decision column to the right.



17. Rename the column *Passenger1.Miles* and select data type *number*:

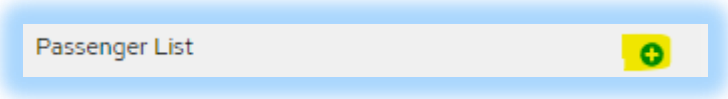


18. Add logic so that if passengers have the same status, they are prioritized on air miles:

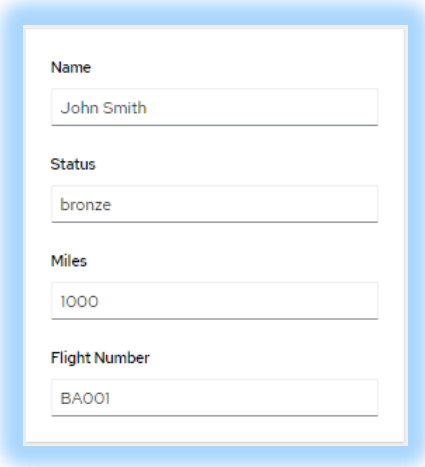


Test New Business Logic

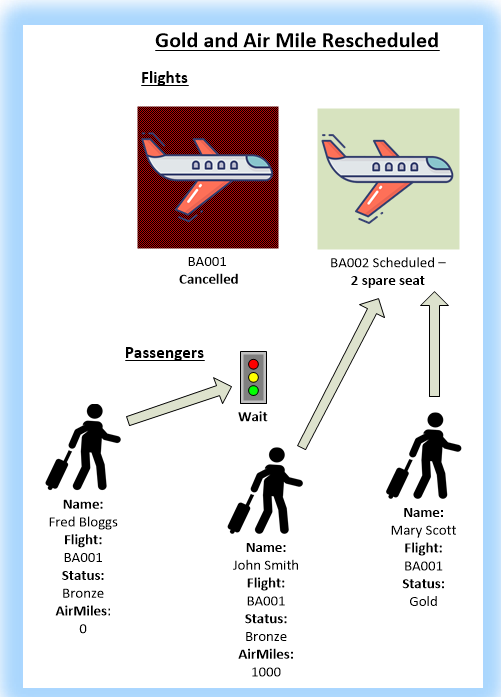
19. Add another passenger. Click on the *Plus* symbol next to *Passenger List* in the *Inputs* Panel, then scroll to the bottom of the list to find the new entry. **Tip:** You may need to drag out the test panel to see the *Plus* symbol.



20. Scroll to the bottom of *Passenger List* to find the newly created entry. Add passenger *John Smith* with the following data:



21. John has *Bronze* status the same as *Fred Bloggs*, but John has more air miles, so using the new logic, he will have priority. Verify that *Fred Bloggs* is put on hold and *John Smith* is rescheduled to flight BA002



3 Conclusion

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