



Cross-Sell on the Web

Extended

STUDENT GUIDE

© Copyright 2023
Pegasystems Inc., Cambridge, MA
All rights reserved.

This document describes products and services of Pegasystems Inc. It may contain trade secrets and proprietary information. The document and product are protected by copyright and distributed under licenses restricting their use, copying, distribution, or transmittal in any form without prior written authorization of Pegasystems Inc.

This document is current as of the date of publication only. Changes in the document may be made from time to time at the discretion of Pegasystems. This document remains the property of Pegasystems and must be returned to it upon request. This document does not imply any commitment to offer or deliver the products or services provided.

This document may include references to Pegasystems product features that have not been licensed by your company. If you have questions about whether a particular capability is included in your installation, please consult your Pegasystems service consultant.

PegaRULES, Process Commander, SmartBPM® and the Pegasystems logo are trademarks or registered trademarks of Pegasystems Inc. All other product names, logos and symbols may be registered trademarks of their respective owners.

Although Pegasystems Inc. strives for accuracy in its publications, any publication may contain inaccuracies or typographical errors. This document or Help System could contain technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Pegasystems Inc. may make improvements and/or changes in the information described herein at any time.

This document is the property of:
Pegasystems Inc.
1 Rogers Street
Cambridge, MA 02142
Phone: (617) 374-9600
Fax: (617) 374-9620
www.pega.com

Mission: Cross-Sell on the Web Extended

Product: Pega Customer Decision Hub™ '23

URL: <https://academy.pega.com/mission/cross-sell-web-extended/v6>

Date: 22 August 2023

Contents

Business use case: Cross-sell on the web extended.....	4
Business use case: Cross-sell on the web extended.....	5
Creating and understanding decision strategies.....	10
Decision strategies.....	11
Creating a decision strategy.....	17
Decision strategy execution.....	23
Creating engagement strategies using customer credit score.....	34
Customer credit score using a scorecard.....	35
Building a scorecard to calculate the creditscore.....	40
Using a Scorecard in a Decision Strategy.....	48
Creating eligibility rules using customer risk segments.....	54
Creating customer risk segments using a decision table.....	55
Using predictive models.....	65
Predictive models drive predictions.....	66
Arbitrating across business issues.....	70
Using predictions in engagement strategies.....	72
Creating parameterized predictors.....	78
Defining and creating customer journeys.....	83
Pega Customer Journeys.....	84
Defining Customer Journeys.....	90

Business use case: Cross-sell on the web extended

Description

Next-Best-Action Designer guides you through the creation of a core Next-Best-Action strategy for your business. Learn how to customize the core strategy by creating decision strategies from scratch that extend Next-Best-Action Designer capabilities.

Learning objectives

- Explain when to consider creating decision strategies from scratch

Business use case: Cross-sell on the web extended

Introduction

This video describes several use cases where decision strategies are used to extend Next-Best-Action Designer capabilities.

Transcript

U+ is a retail bank. The bank is leveraging its website as a marketing channel to improve 1-to-1 customer engagement, drive sales, and deliver Next-Best-Actions in real-time.

The bank is using the Pega Customer Decision Hub™ to recommend more relevant banner ads to its customers when they visit their personal portal. In the start-up phase, U+ successfully implemented all their use cases using Next-Best-Action Designer.

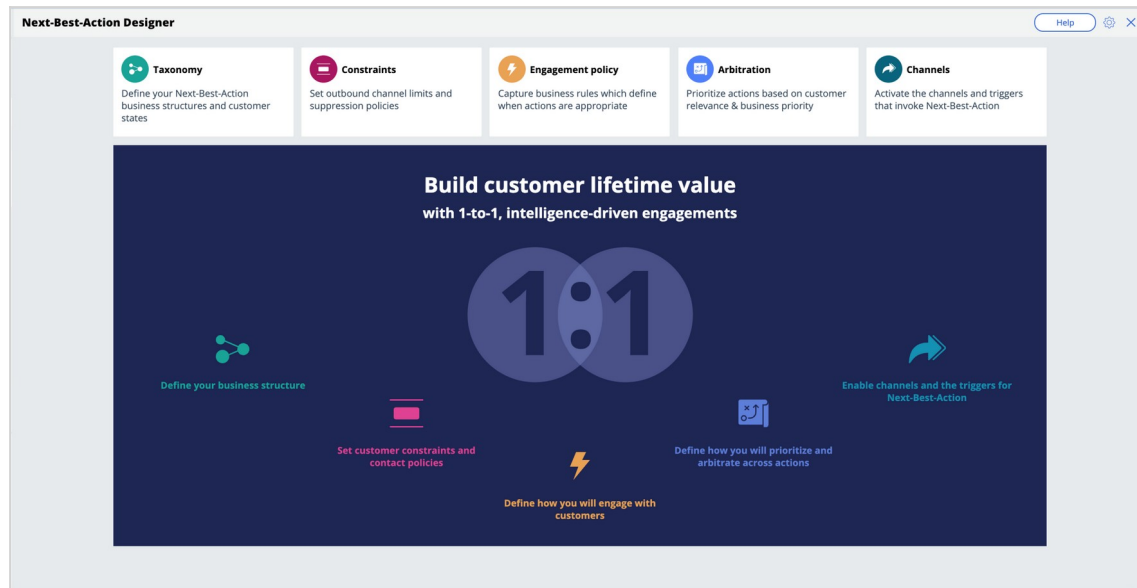
Next-Best-Action Designer guides you through the creation of a Next-Best-Action strategy for your business.

With the Next-Best-Action Designer user interface you define high-level business rules and AI controls, which the system uses to configure the underlying Next-Best-Action strategy framework.

This framework leverages best practices to automatically generate Next-Best-Action decision strategies at the enterprise level.

These decision strategies are a combination of the business rules and AI models that form the core of the Pega Customer Decision Hub, which determines the personalized set of

Next-Best-Actions for each customer.



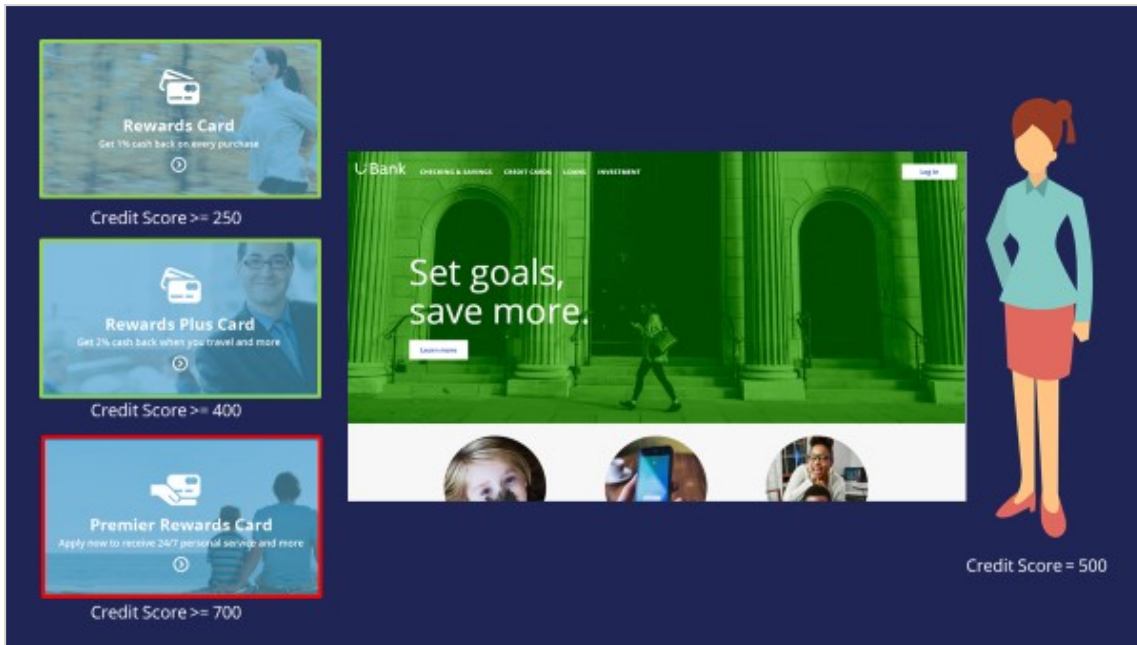
U+ Bank wants to implement additional use cases to meet new business requirements. These use cases require U+ bank to extend Next-Best-Action Designer capabilities.



The first use case is to create suitability rules based on a customer's credit score. The bank wants to determine whether or not a customer is suitable for an offer based on their credit score. In this scenario, the credit score is not available, it needs to be computed in real-time based on customer profile information.

For example, when customer Barbara logs in, U+ bank only wants to present her with the Rewards and Rewards Plus offers, not the Premier Rewards offer. Barbara's credit score is 500. This makes her unsuitable for Premier Rewards, which is only suitable for customers

with a credit score over 700.



To determine suitability, the bank wants to calculate a customer's credit score using a scorecard. A scorecard is used to assign importance to pieces of data for use in a calculation. A scorecard uses a subset of customer property values divided into ranges and assigns scores to each range to compute a final score.

Predictor	Condition	Score
CLV_Value	<100	83
	<400	176
	Otherwise	221
Age	<20	21
	<40	117
	Otherwise	55
Account	N	10
	Y	82

Your Credit Score is: **241**

To implement this, you use a special Suitability condition in Next-Best-Action Designer. The Suitability condition uses a decision strategy that references a Scorecard rule. The Scorecard rule is used to determine the customer's credit score.

In the next use case, the bank has introduced some strict regulations for which they need new eligibility rules. U+ does not want to offer credit cards to customers whom they classify as 'high risk'. Customers are divided into risk segments from AAA to CCC based on their

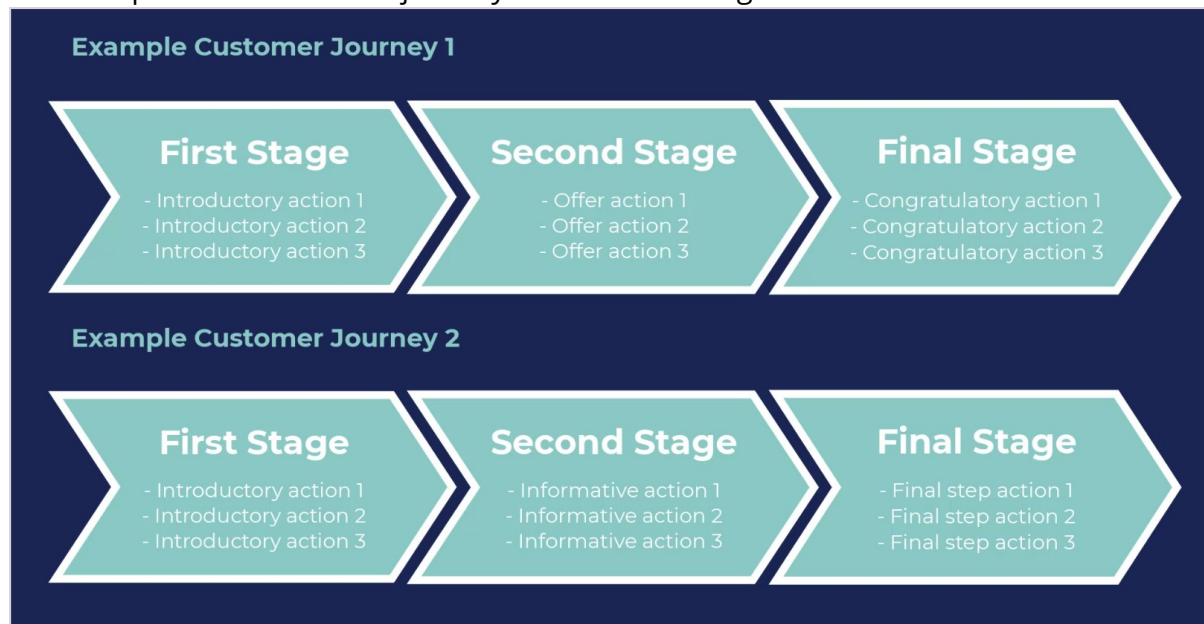
outstanding loan amount and credit score. In the beginning, only customers in the risk segments BBB and CCC will be eligible for credit cards.

Risk Segmentation

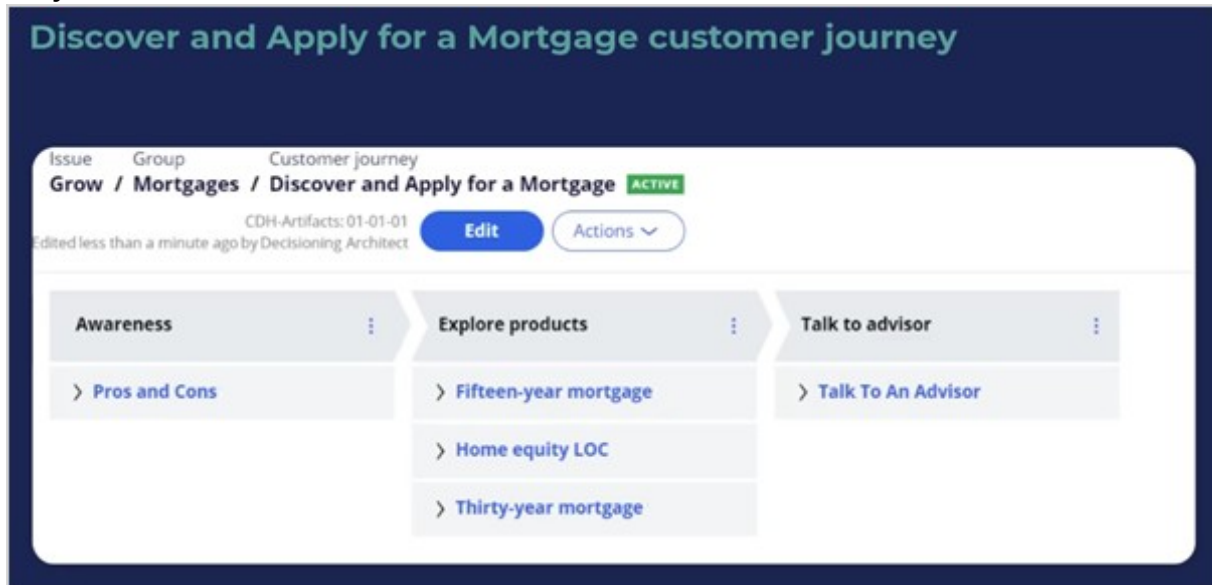
Condition	Risk segment
If Outstanding loan amount >= \$50000	AAA
If Outstanding loan amount is between \$10000 and \$25000 AND Credit score is between 600 and 800	BBB
If Outstanding loan amount is less than \$10000 AND Credit score is between 100 and 200	BBB
If Outstanding loan amount is less than \$10000 AND Credit score is more than 200	CCC
If Outstanding loan amount AND Credit score falls in any other range	AAA

To implement this, you use a special Eligibility condition in Next-Best-Action Designer. The Eligibility condition leverages a decision strategy that uses the scorecard to determine the customer's credit score, which it passes as an argument to a decision table. The decision table then determines which risk segment the customer is in.

In the next use case, the bank wants to regulate and document the full customer experience. To implement this business requirement, you use Pega Customer Journeys. You can set up various customer journeys divided into stages that contain a number of actions.



The customers can advance the journey by moving to the next stage when they meet the required entry criteria. Customer journeys provide a good mental model for managing content and influencing next-best-action decisions to present suitable actions seamlessly for your customers.



In summary, this video has shown you the various use cases U+ Bank would like to implement by extending Next-Best-Action Designer capabilities in this phase of the project.

Creating and understanding decision strategies

Description

Next-Best-Action Designer provides a guided and intuitive UI to bootstrap your application development with proven best practices that generate the underlying strategies for you. These strategies can be customized using designated extension points or by building decision strategies from scratch, depending on the business requirement.

Learning objectives

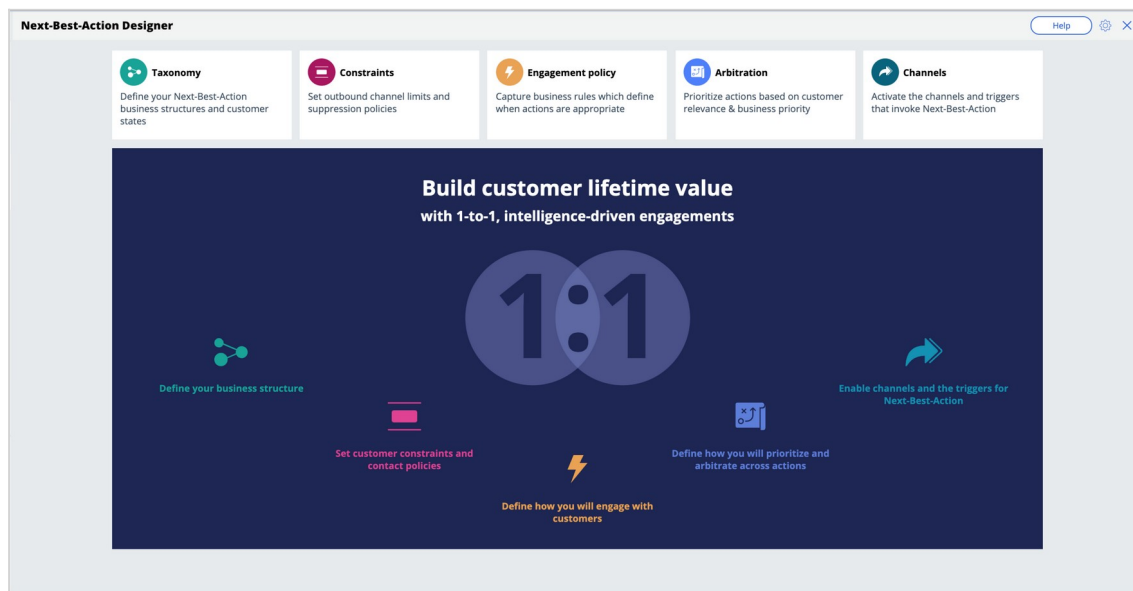
- Describe how decision strategies are used in the Next-Best-Action strategy framework
- Explain the decision strategy canvas and its building blocks
- Create decision strategies from scratch
- Explain what's going on inside each component when a decision strategy is executed

Decision strategies

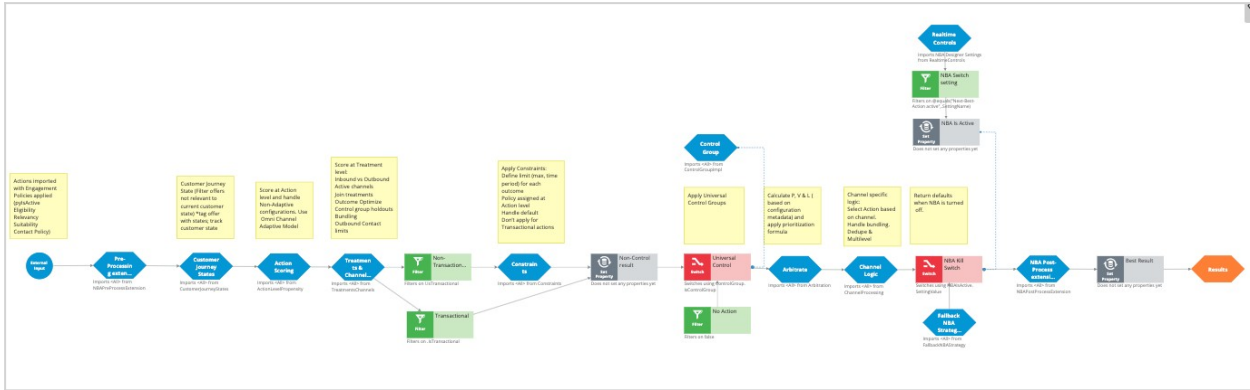
Transcript

Next-Best-Action Designer guides you through the creation of a Next-Best-Action strategy for your business. Its intuitive interface, proven best practices and sophisticated underlying decisioning technology enable you to automatically deliver personalized customer experiences across inbound, outbound, and paid channels.

The Next-Best-Action Designer user interface allows you to easily define, manage and monitor Next-Best-Actions.



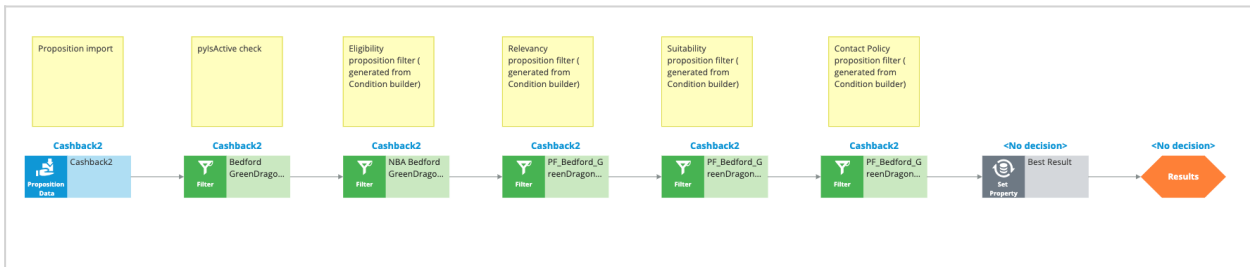
As you use the Next-Best-Action Designer user interface to define strategy criteria, the system uses these criteria to create the Next-Best-Action Strategy framework. This framework leverages best practices to generate Next-Best-Action decision strategies at the enterprise level. These decision strategies are a combination of the business rules and AI models that form the core of the Pega Centralized Decision Hub, which determines the personalized set of Next-Best-Actions for each customer.



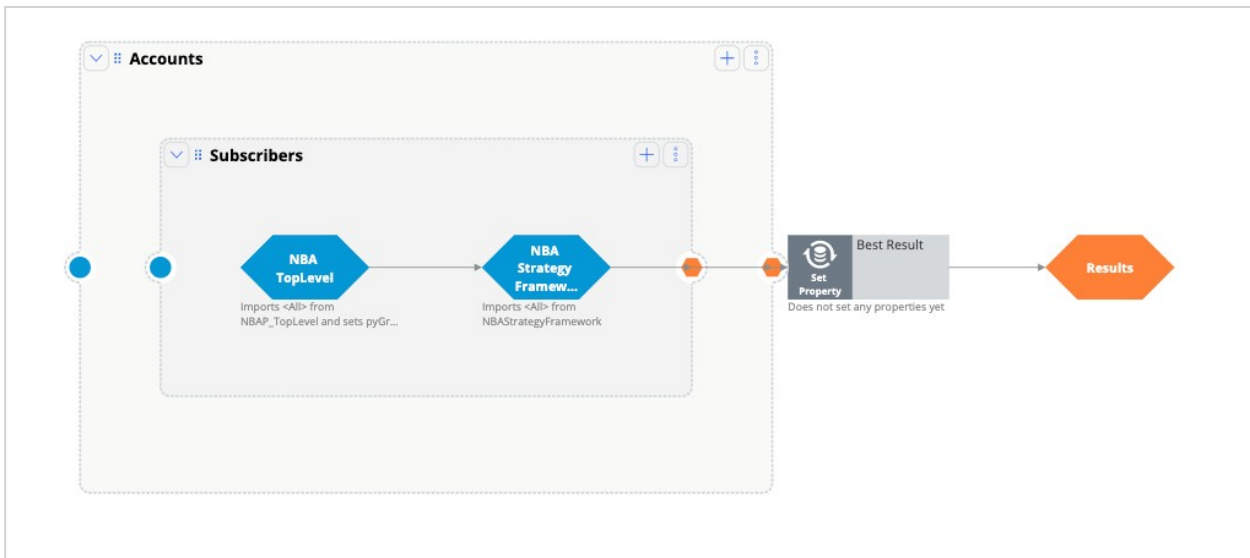
If you want to modify the strategy later, you can do that from Next-Best-Action Designer's simple and transparent interface.

The strategy framework is applied to all relevant Actions and Treatments after you define a Trigger in the Next-Best-Action Designer **Channels** tab.

Each Trigger generates a strategy that first imports the Actions from the appropriate level of the business structure and then applies the Eligibility, Applicability and Suitability rules.

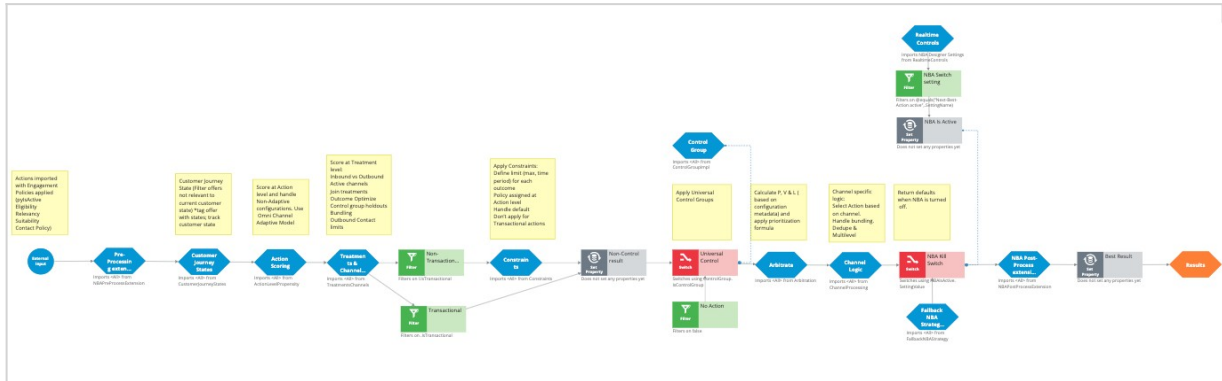


The strategy then passes these results to the strategy framework for processing.

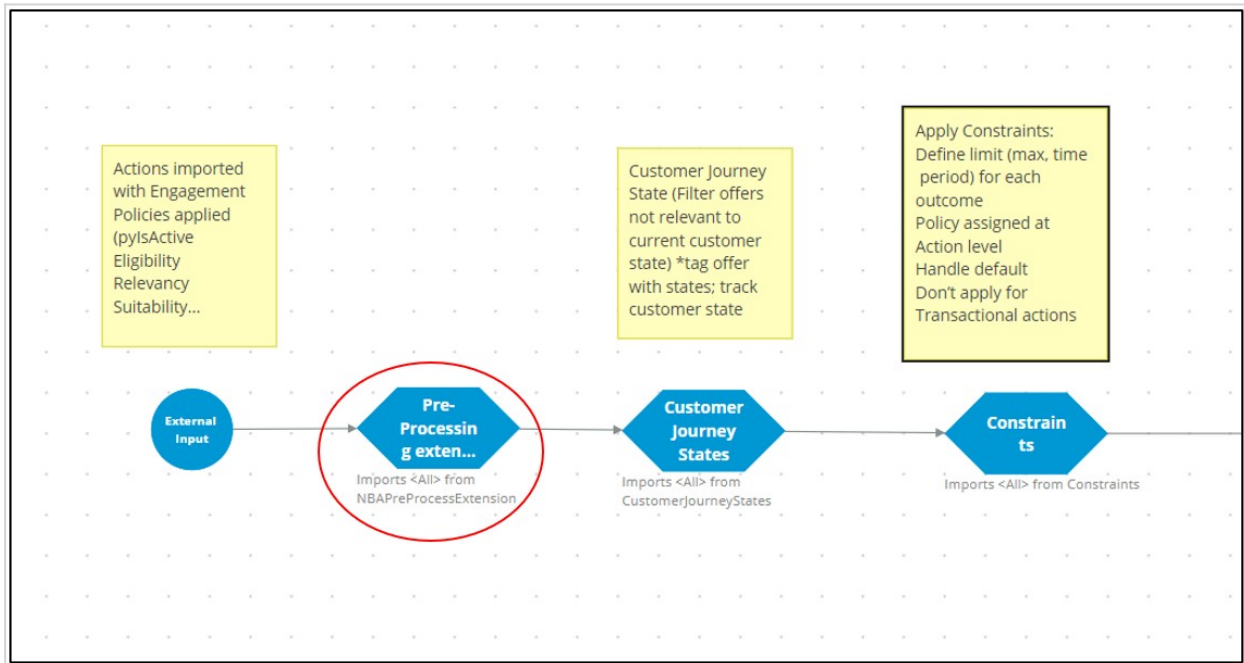


There are several extension points within the framework. An extension point is an empty rule or activity that is intended to be overridden to meet the specific needs of the application. When building an implementation of the current framework, the decision strategy designers must override the empty activity with a functioning interface to their customer master file.

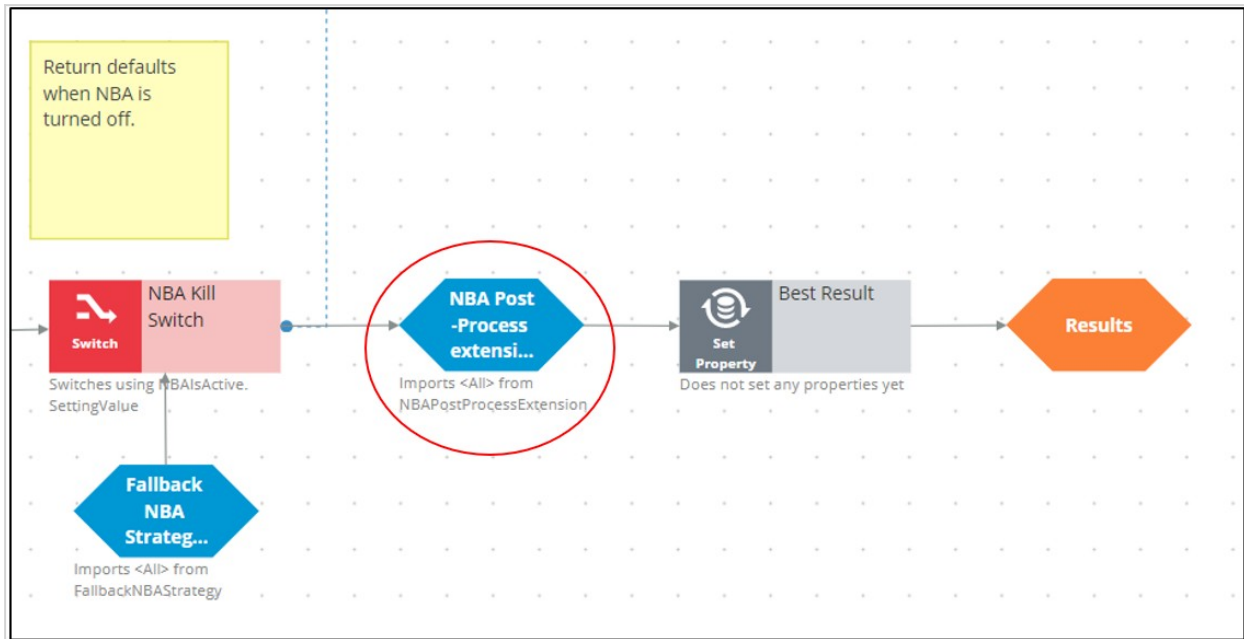
This is the NBA framework strategy when applied to each of the Actions.



The first component within the strategy framework is an extension point for any Action pre-processing you might need to perform.



The last functional component within the strategy framework is policy another extension point for any post-processing that must be performed.



Similarly, there are many other extension points such as the outbound limits extension points and business value extension points.

To ensure upgradeability, avoid overriding any part of the framework that is not a designated extension point.

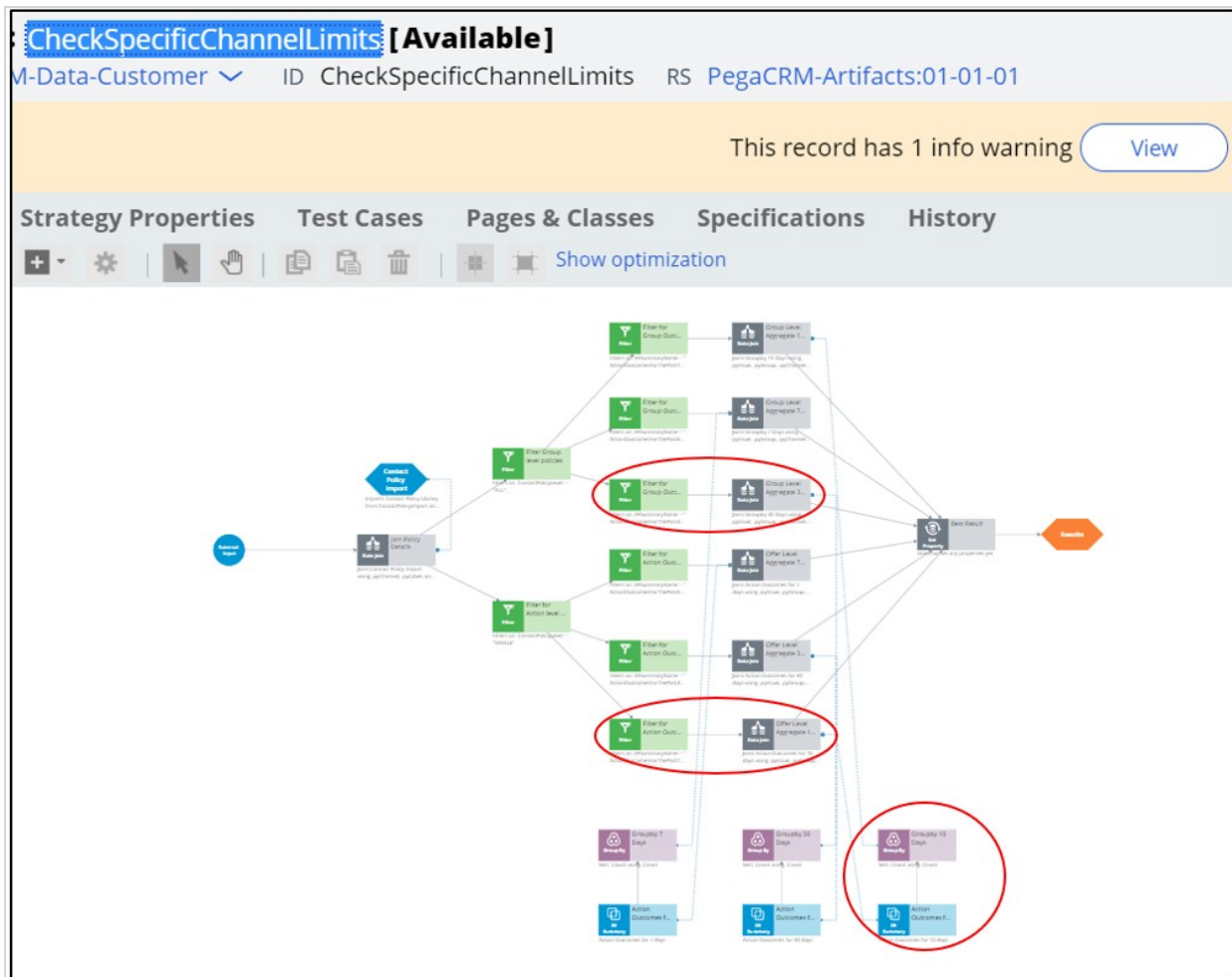
Also, the generated framework has some extension points where you can create strategies.

For example, while configuring values for Arbitration, you can specify a business value for an Action, or you can use a strategy to calculate the value. This can be done by adding a strategy to the existing framework.

Similarly, in defining the engagement rules, you can use a new strategy as a definition instead of an existing condition. Strategy designers can create such strategies from scratch using the decision strategy canvas.

Or, while defining the suppression rules, you can add a strategy to define new suppression rule limits instead of the existing 7 or 30 days.

For example, in the screenshot below, the CheckSpecificChannelLimits rule has been extended to have a 15-day limit:



In conclusion, the NBA Designer provides a guided and intuitive UI to bootstrap your application development with proven best practices. NBA designer generates the underlying strategies for you, which can be extended using existing values in the designated extension points or by building decision strategies from scratch, depending on the business requirement.

Content in 3 different ppts in artifacts folder

Creating a decision strategy

Introduction

Decision Strategies drive Next-Best-Action. They comprise a unit of reasoning represented by decision components. How these components combine determines which action will be selected for a customer: the Next-Best-Action. Learn the type of decision components and how they are used to create decision strategies. Gain hands-on experience designing and executing your own Next-Best-Action decision strategy.

Transcript

This demo will show you how to create a new decision strategy.

It will also describe three important decision components and the types of properties available for use in expressions during strategy building.

In this demo you will build a Next-Best-Label strategy. The Next-Best-Label strategy is a sample strategy, used to illustrate the mechanics of a decision strategy.

Start by creating a new strategy from scratch.

Decision strategies output actions, utilizing the so-called Strategy-Results class.

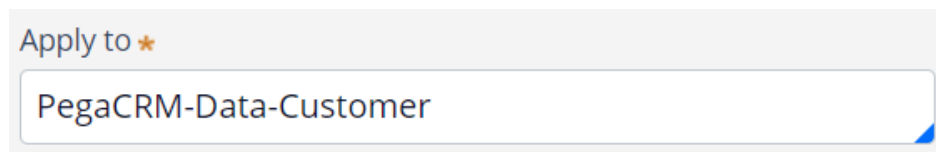
The Strategy-Results class limits the output of the strategy to the actions contained in the Business issue and Group.

The strategy you build will select a Label action from a set of predefined actions. The Label action selected will be the one with the lowest printing cost.

Notice that the complete definition of the Next-Best-Label strategy needs to include a reference to the PegaCRM-Data-Customer class.

This is the 'Apply to' class and it indicates the context of the strategy.

It ensures that from within the strategy, you have access to customer-related properties such as Age, Income, Address, Name, etc.

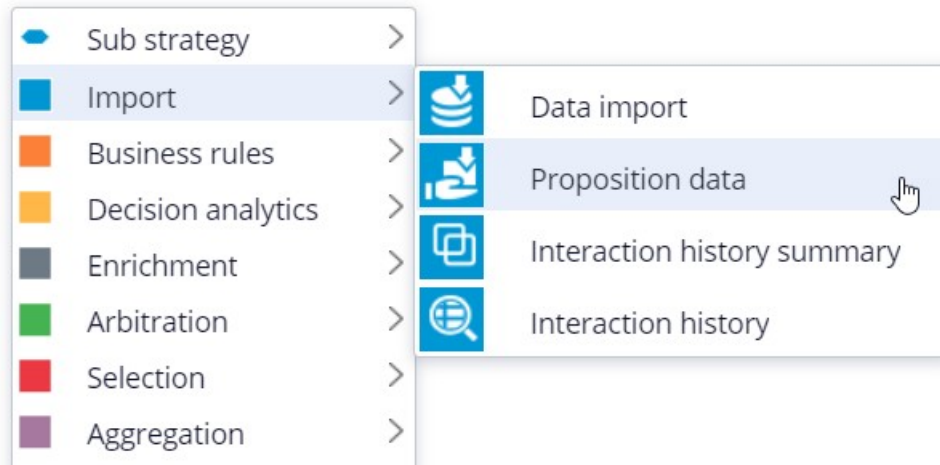


You can now start building the strategy. Right-click on the canvas to get the Context menu, which shows all component categories.

The first component to add is an Import component.

By expanding the Import category, you can see the Import component types available.

In this case you need a Proposition Data component to define the actions that will be considered by the strategy.



Now you need to configure the component. First, right-click to open the Proposition Data properties panel.

Notice that the Business issue and Group are grayed out.

This cannot be changed because the Enablement Business issue and Labels Group have already been selected for this decision strategy.

By default, the strategy will import all actions within that Group, unless you select a specific action.

For this component, you only want to import the Green Label, so let's select that.

Selecting the action from the drop-down menu automatically gives the component the appropriate name.

The description, which will appear under the component on the canvas, will also be generated automatically.

If you want to create your own description, you can do so by clicking the 'Use custom' radio button.

Now you want to import a second action into the strategy. You can use the Copy and Paste buttons to quickly add more Proposition Data components to the canvas.

You can use Alignment Snapping and Grid Snapping for easy placement of the components.

By turning these off, you can place a component anywhere on the canvas, but it makes it more difficult to align the shapes.



Now you need to add the next component in the strategy, which is an Enrichment component called Set Property.

You can add this component to the canvas by selecting it from the component menu.

Next, connect it to the Proposition Data components.

Ultimately, the result of this strategy should be the Label action with the lowest printing cost.

This printing cost is the sum of a base printing cost, which is specific to each label, and a variable cost, which depends on the number of letters.

The Set Property component is where you will calculate the printing cost for each of the actions.

The information in the 'Source components' tab is populated automatically by the Proposition Data components connected to this component.

Notice that the Black Label action is in the first row.

On the Target tab you can add properties for which values need to be calculated.

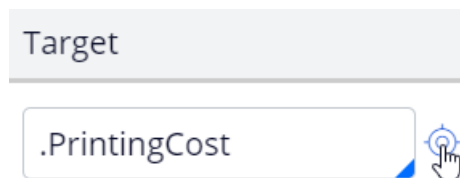
Click 'Add Item' to create the equation that will calculate the printing cost for each of the components.

Begin by setting the Target property to 'dot' PrintingCost.

In Pega, all inputs begin with a dot. This is called the dot-operator and it means that you are going to use a strategy property.

The PrintingCost property is a new strategy property that does not yet exist.

To create the new PrintingCost strategy property, click on the icon next to the Target field.



By default, the property type is Text. In Pega, there are various types supported. In this case, the PrintingCost is a numeric value, so change its type to Decimal.

Next, you need to make PrintingCost equal to the calculation you create. To create the calculation, click on the icon next to the Source field.

Using the Expression builder, you can create all sorts of complex calculations, but in this use case, the computation is very basic.

PrintingCost should equal $\text{BaseCost} + 5 * \text{LetterCount}$.

To access the BaseCost you type a dot. Notice that when you type the dot, a list of available and relevant strategy properties appears.

This not only makes it easy to quickly find the property names you're looking for; it also avoids spelling mistakes.

In a decision strategy, you have two categories of properties available to use in Expressions.

The first category contains the strategy properties, which can be one of two types.

An Action property is defined in the Action form. Examples are the BaseCost and LetterCount properties you are using here.

These properties have a value defined in the Action form and are available in the decision strategy via the Proposition Data component.

The property values can be overridden in the decision strategy but will often be used as read only.

The second type of strategy property is a calculation like the one you just created, PrintingCost. Such calculations are often created and set in the decision strategy.

These types of properties are either used as transient properties, for temporary calculations, or for additional information you want the strategy to output.

The second category contains properties from the strategy context, also called customer properties.

Suppose you want to use a customer property in your Expression, such as Age or Income.

In that case, you would have to type the prefix 'Customer dot', instead of just dot.

This is the list of available properties from the strategy context, also known as Customer properties.

For now, you calculate the printing cost for each action that does not use customer properties.

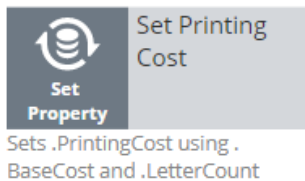
Finalize the Expression.



Even though you used the dot-operator to build your Expression, it's best practice to validate it, so click Test.

If the Expression isn't valid, you will receive an error message on screen.

On the canvas, you can see the automatically generated description for the component: Sets PrintingCost using BaseCost and LetterCount.




Now you want to ensure that the actions will be prioritized based on the lowest printing cost. So, you need to add the Prioritize component from the Arbitration category.

The prioritization can either be based on an existing property, or it can be based on an equation. Let's select an existing property using the dot construct.

Here you can select the order in which the top actions are presented. Since you are interested in the lowest printing costs, configure it accordingly.

You can also select the number of actions that will be returned by the strategy.

If you want to output only one label, select Top 1 here.

Expression* 

Order by

Highest first (9 to 1)

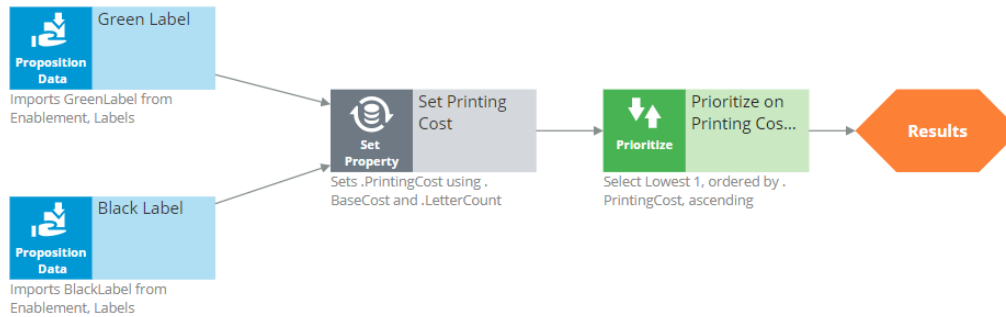
Lowest first (1 to 9)

Output

Top

All

Now you can connect the components and save the strategy.



To test the strategy, first check it out. Then, expand the right-hand side test panel and click 'Save & Run' to examine the results.

You can view results for any of the components by selecting that component.

If more than one action is present, each one is presented as a Page.

For the Set Property component, the Results contain a page for the Black Label and one for the Green Label.

For the Black Label the PrintingCost is 70.

For the Green Label the PrintingCost is 60.

On the canvas, you can show values for strategy properties such as Printing Cost.

For this exercise, you execute this strategy against a Data Transform called UseCase1.

If you open UseCase1, you can see the customer data the strategy uses when you run it.

To test the strategy on a different use case, you can create a Data Transform with different properties.

You can also select a Data Set that points to an actual live database table.

This demo has concluded. What did it show you?

- How to create a decision strategy from scratch.
- How to configure Proposition Data, Set Property and Prioritize decision components.
- How to build expressions in strategies.
- The two categories of properties available for expressions.
- How to test a decision strategy using a use case stored in a data transform.

Decision strategy execution

Introduction

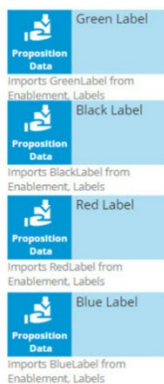
Using Pega Decision Management, you do not need to be an expert in programming, math or data science to design and execute sophisticated decision strategies that engage your customers throughout the customer journey. With its highly intuitive graphical canvas, Pega Decision Management enables you to easily embed Pega or third-party predictive models into your decision strategies. The result is customer-centric interactions that improve the customer experience while increasing customer value, retention and response rates.

Transcript

This demo explains what's going on inside each component when a Decision Strategy is executed.

For example, what happens 'under the covers' when a Filter component is executed, and how does it interact with the components around it?

In the interest of keeping it simple, this example is limited to four actions. In reality, decision strategies will involve many more actions than that.



Here are our 4 actions: 'Green Label', 'Black Label', 'Red Label' and 'Blue Label'; they are represented by a Data Import or, more specifically, a Proposition Data component.

In this example, the Proposition Data components import three data properties for each action: Name, BaseCost and LetterCount.

 Green Label

Imports GreenLabel from Enablement, Labels

 Black Label

Imports BlackLabel from Enablement, Labels

 Red Label

Imports RedLabel from Enablement, Labels

 Blue Label

Imports BlueLabel from Enablement, Labels

Rank	Name	BaseCost	LetterCount
1	Green Label	10	10

The first action's Name is Green Label, its BaseCost is 10, and its LetterCount is 10.

Likewise, the other actions have a Name, BaseCost and LetterCount.

 Green Label

Imports GreenLabel from Enablement, Labels

 Black Label

Imports BlackLabel from Enablement, Labels

 Red Label

Imports RedLabel from Enablement, Labels

 Blue Label

Imports BlueLabel from Enablement, Labels

Rank	Name	BaseCost	LetterCount
1	Green Label	10	10

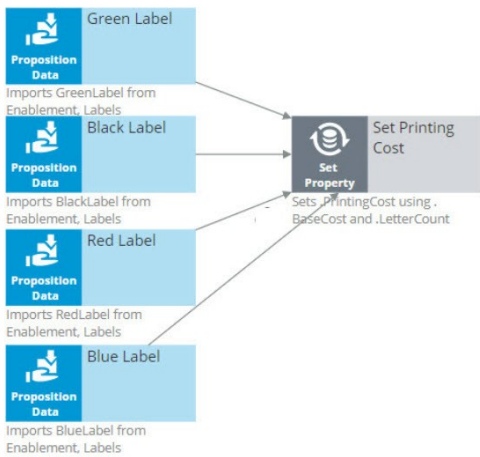
Rank	Name	BaseCost	LetterCount
1	Black Label	20	10

Rank	Name	BaseCost	LetterCount
1	Red Label	30	8

Rank	Name	BaseCost	LetterCount
1	Blue Label	40	9

One property is automatically populated for you; this is the Rank. We will come back to this later, but notice that, as separate components, each action has a Rank of 1.

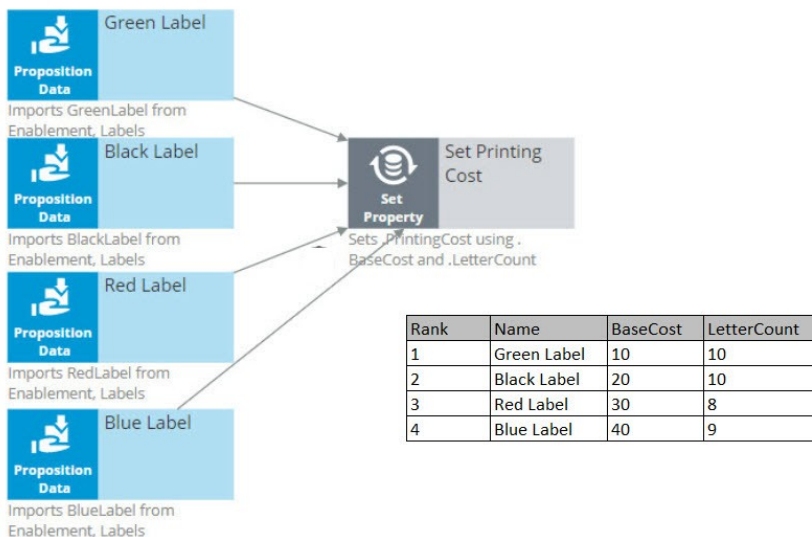
On the strategy canvas, components are connected by drawing arrows from component to component. So, what do these arrows mean exactly?



Well, when you draw an arrow, what happens is that, at runtime, all information in the component you're drawing the arrow from is available as a data source to the component you're drawing the arrow to.

So now, the Name, BaseCost and LetterCount for all of the actions are available in a single Set Property component.

The only data element that changes is the row number, or as we call it in the strategies, the Rank. In each decision component, the Rank value is automatically computed.



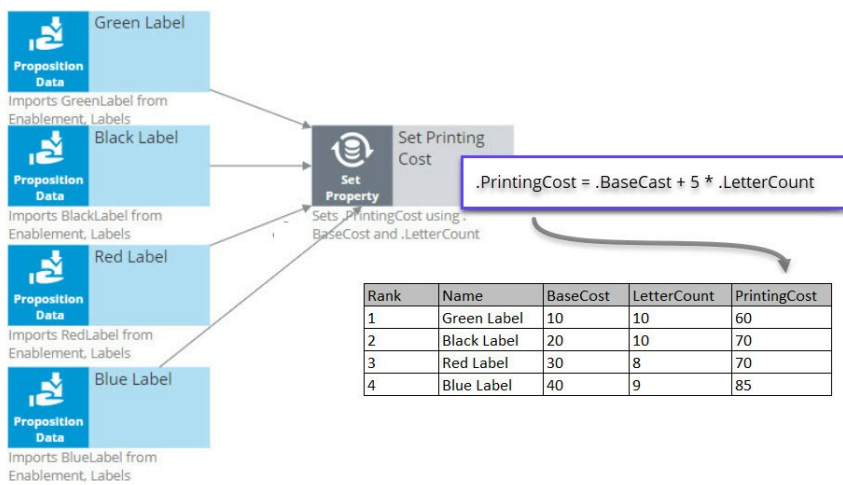
In the Set Property component, the Rank is determined by the order in which the actions are received by the component.

As a result, in this instance, the Green Label action has a Rank of 1, Black has a Rank of 2, Red has a Rank of 3, and Blue has a Rank of 4.

Ultimately, you want to select the best Label action. That is the Label with the lowest printing cost.

The printing cost of a Label is the sum of the BaseCost and a variable cost based on the LetterCount.

You configure the Set Property component to compute the printing cost of each Label action.



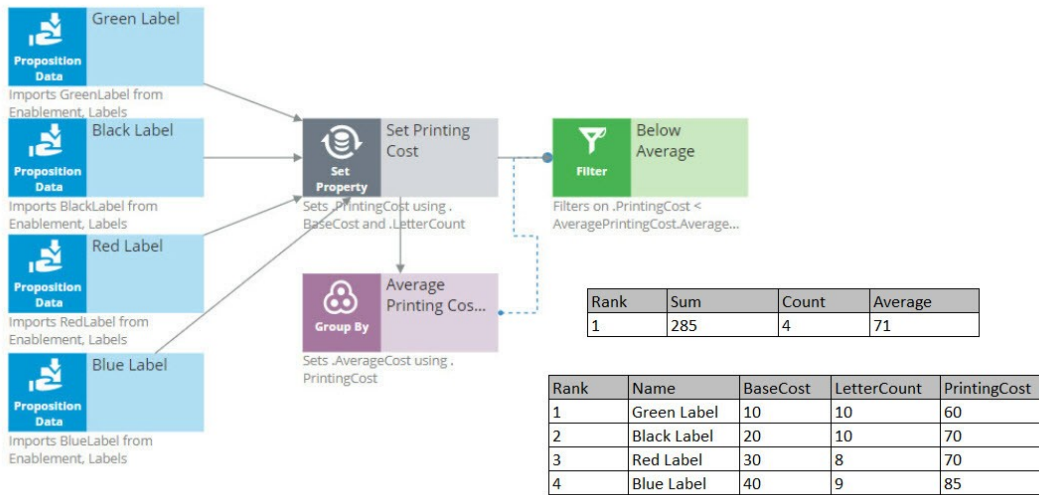
Because we are combining the data in our four Proposition Data components into one Set Property component, we only need to add one PrintingCost property to the new component, and it automatically computes the printing cost for all four actions.

For the Green Label action, PrintingCost equals a BaseCost of 10 plus 5 times the LetterCount of 10 which equals 60.

Similarly, the PrintingCost for the Black and Red Label actions is 70, and for the Blue Label action is 85.

Now, let's say the business rule is to select only Label actions with a printing cost lower than the average printing cost of all labels. For this requirement we use a 'Group by//Filter' component combination.

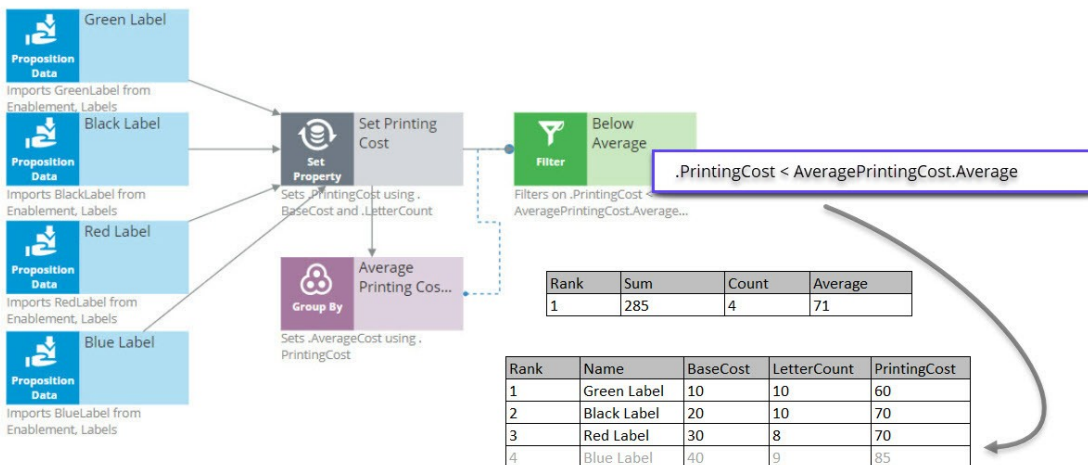
A 'Group by' component offers essential aggregation capabilities, like Sum and Count, that are used in many decision strategies. We will use it to calculate the average printing cost.



Again, we have our set of actions, each with their own specific PrintingCost value. The 'Group by' component combines all actions into one row. How does that work?

Well, it sums the PrintingCost values for all the actions, it counts the actions, and it calculates the average printing cost by dividing the summed printing cost by the count.

In this example, the sum of the PrintingCost values is 285, and the count of the actions is 4, so the average printing cost is 71.



Now that you have calculated the average printing price using a 'Group by' component, configure the Filter component to filter out actions that have a printing cost equal to or higher than this average.

So far in this strategy, we've seen only the solid line arrows, which copy information from one component to another. But now we also see a dotted line arrow.

This tells us that a component refers to information in another component.

Here, the Filter component is referencing the average printing cost that exists inside the Aggregation component. This is an important capability to understand.

The Filter component filters out actions when the printing cost for that action is equal to or above the average printing cost and propagates the other actions.

First, via the solid arrow, the filter looks at the actions sourced from the Set Property component.

Then, it applies the filter condition, which references the average printing cost in the 'Group by' component via the dotted arrow.

The Filter Condition in the Filter component is the Expression: 'dot PrintingCost is smaller than AveragePrintingCost dot Average'.

By using this ComponentName dot Property construct, any decision component can be referenced by any other component by name.

Important to note that the Filter component lets actions through when the condition Expression evaluates to **true** and filters out actions when the condition Expression is not met.

When you refer to a component, you always refer to the first element in the component, the one with Rank 1.

In this case, you are referring to the one and only row in the 'Group by' component, which naturally has Rank 1.

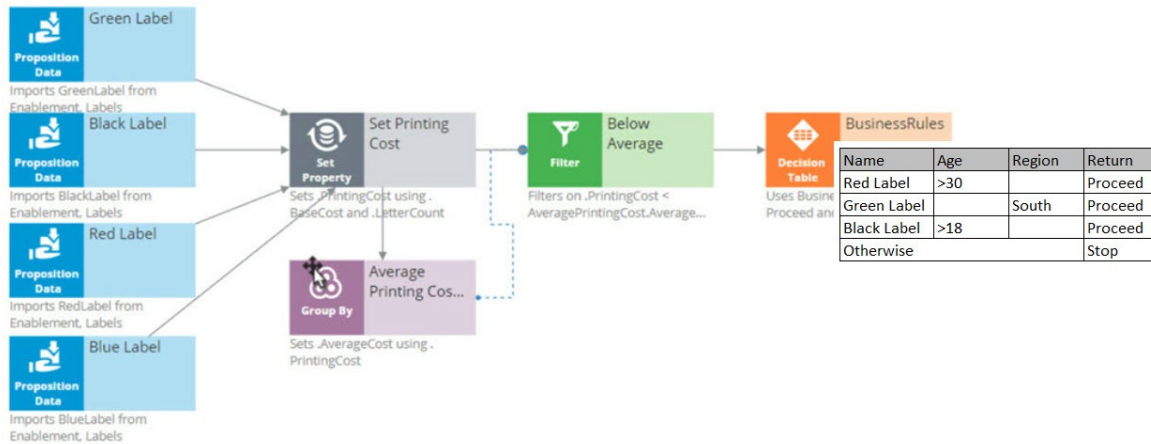
The Rank 1 average equals 71 in the 'Group by' component. This means that the filter will allow Label actions through that have a printing cost lower than 71.

By this standard, the printing cost of the Blue Label action is too high, so it is filtered out. The printing cost of the other Label actions are below 71, so they survive.

The result is that the table contains three surviving actions: Green Label with Rank 1, Black Label with Rank 2, and Red Label with Rank 3.

The next component is a Decision Table. A Decision Table in Pega is an artifact that can be used to implement business requirements in table format.

In a Decision Table, the business rules are represented by a set of conditions and a set of Return values.



The Decision Table receives information about the remaining actions via the solid arrow from the Filter component.

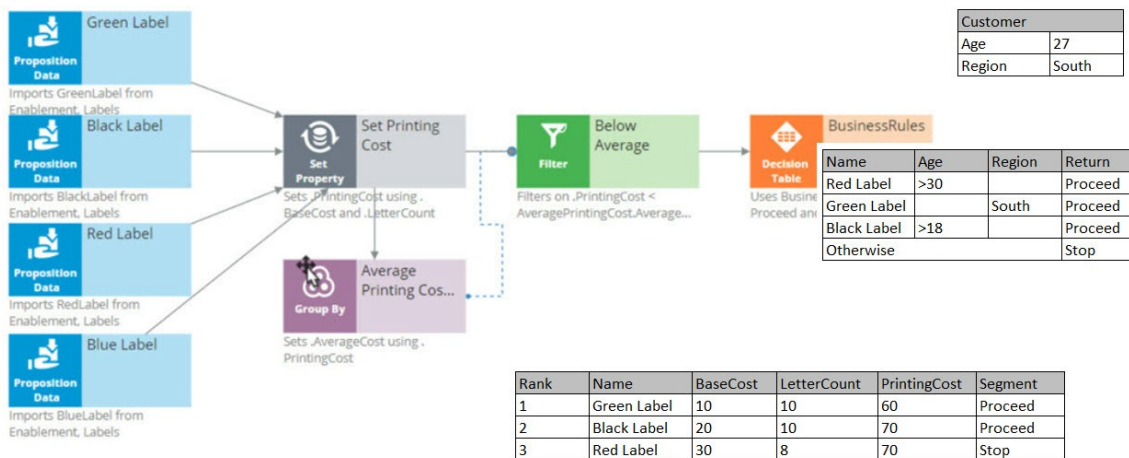
The business criteria say that the Red Label action can be offered if the customer's age is over 30 and they are from any region. If these criteria are met, the Return value is 'Proceed'.

The Decision Table also says that the Green Label action can be offered to anyone in the Southern region. So, if the Region value is South, the Return value for Green is 'Proceed'.

The Black Label action can be offered to anyone over the age of 18.

But in all other cases, or, Otherwise, no Label action meets the criteria, and the Return value is 'Stop'.

As an example, consider a customer with Age 27 and Region South.



Now, the Decision Table applies the business criteria for each action against the customer information and returns a value. The value returned by a Decision Table is also called a Segment.

The Decision Table checks the Green Label action with Rank 1 first, and in this case, it can proceed because the customer's Region is South.

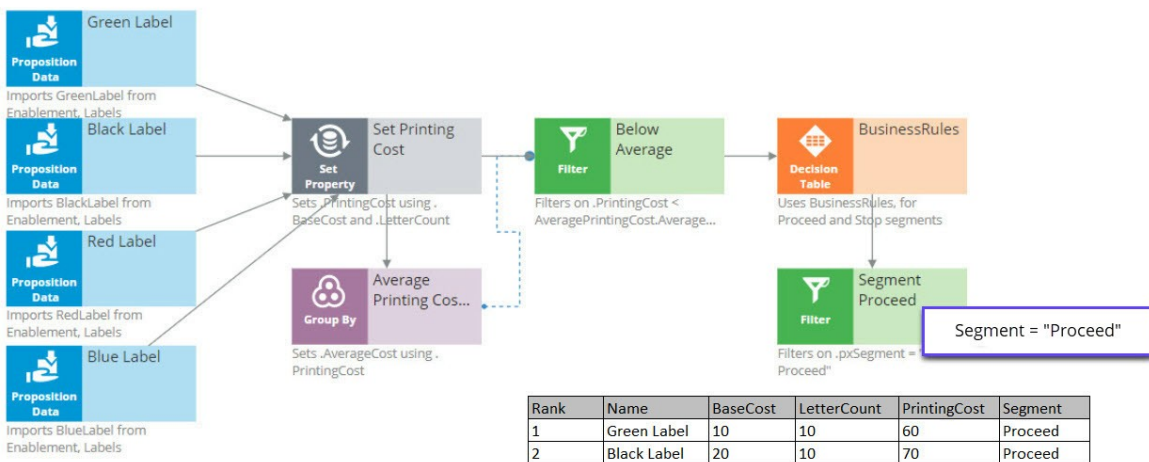
Next, it looks at the Black action and sees that the criteria for Black is that the customer's age is greater than 18. This customer is 27.

Black doesn't care about the Region, so the Segment value for the Black action is 'Proceed'.

Finally, it looks at the Red action, and the Age criteria don't match up, so the Segment value for Red is 'Stop'.

The result of the component is that you get a new segmentation column that flags which of the actions comply with the business rules.

You're now going to filter out the actions that do not match the business rules. This happens in the 'Segment Proceed' Filter component.



Again, via the solid arrow, the strategy copies the data over from the Decision Table component into the Filter component.

Now each action has a Rank, Name, BaseCost, LetterCount, PrintingCost and Segment. The filter condition is applied to this data.

The filter condition says: allow this action through if the Segment value equals 'Proceed'.

What this Filter component now does is go through the list of actions to find the actions with value 'Proceed' in their Segment property.

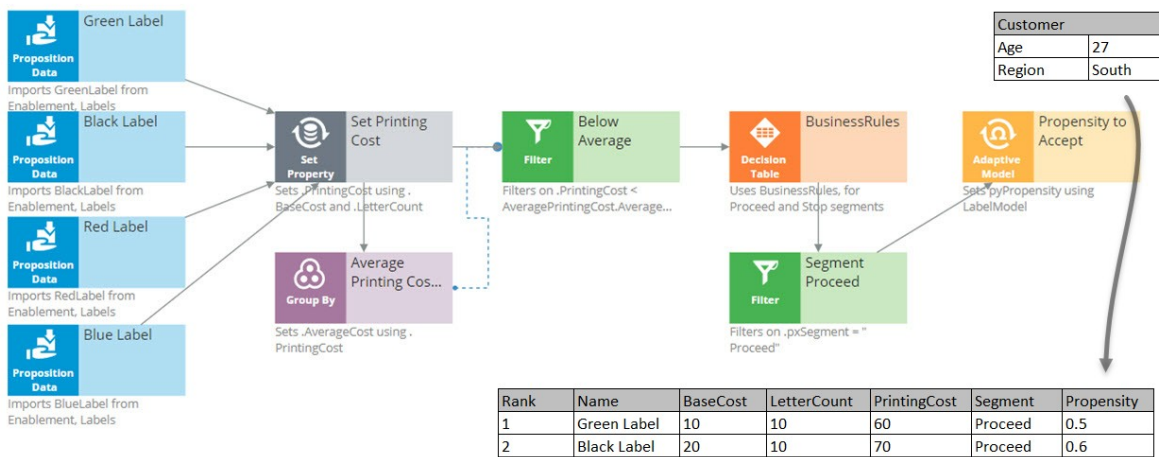
First is the Green Label. Green is allowed through, which means its properties will be available in the new component.

Then the Black Label. It is also allowed through because it also has 'Proceed' in its Segment property.

But the Red Label action is not allowed through, because Red has 'Stop' in its Segment property. Therefore, Red is not part of the output.

The strategy so far has selected two of our original actions, Green and Black.

Now, in the Adaptive Model component, you will use predictive analytics to determine the propensity of each of the remaining actions.



Propensity is the probability that a customer will accept an action, or their likelihood of interest in it.

In order to calculate the propensity, we use an Adaptive Model component. The referenced model is configured to monitor customer characteristics such as Age and Region.

In this case our test customer has an Age of 27 and is from the South Region.

Again, just to keep it simple, we are using a model that makes predictions based on only this information. In reality, models will take into account many more properties.

The Adaptive Model determines the propensity.

First, we supply the action and the customer profile to the Adaptive Model, and the model says: 'Oh, it's the Green Label action; we have some evidence that young people like the Green Label action, but people from the South don't like it.'

Combining both factors, we get an overall propensity of 0.5 for the Green Label action.

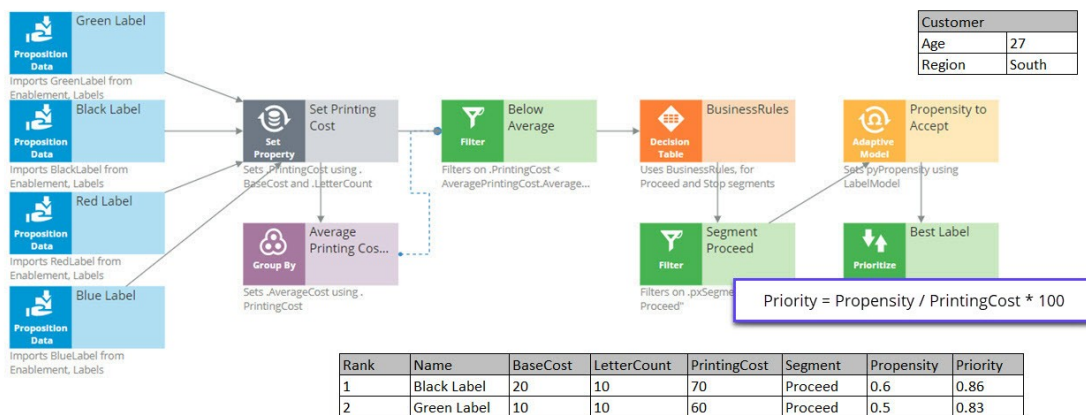
For the Black Label action, the likelihood turns out to be 0.6.

After consulting the Adaptive Model, the Propensity to Accept component sets the Propensity property value for each action.

Remember, the propensity is always a number between zero and 1.

It shows something along the lines of, half of the customers that are like this customer accepted the Green Label action in the past, and 3 out of 5 customers like this customer accepted the Black action last month.

The next component in our chain, called Best Label, is the Prioritize component. This component determines the priority of each action and ranks them. Let's see how this works.



A key element of this component is the priority Expression, which calculates a priority value for each action. According to this Expression, the higher the value, the higher the priority and rank.

In this case, the priority calculation weighs likelihood of acceptance in its equation: 'Propensity divided by PrintingCost times 100'.

When performing this calculation on the Black Label action, we can see that it has a PrintingCost of 70 and a Propensity of 0.6, therefore its Priority is 0.86.

The Green Label action has a lower PrintingCost and a lower Propensity, resulting in a Priority of 0.83.

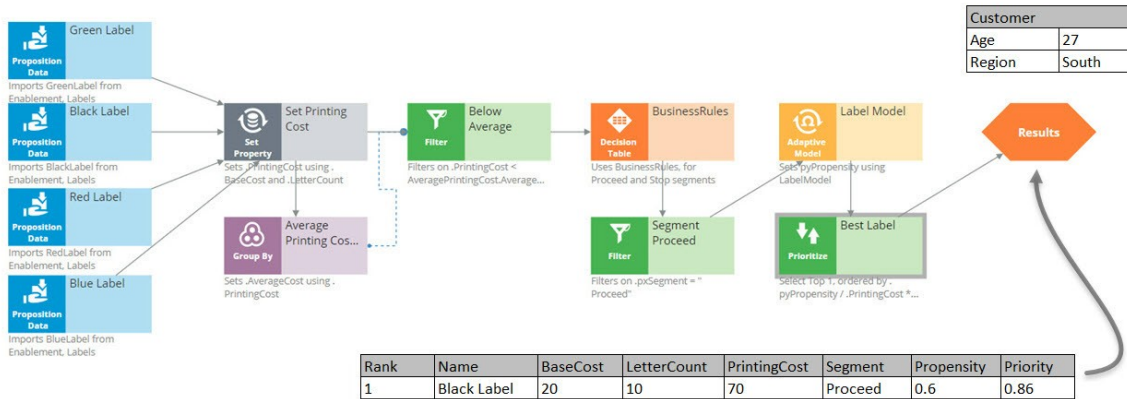
Because 0.86 is higher than 0.83, the Black Label action is now ranked number one.

So, even though the printing cost of the Black Label action is higher than that of the Green Label action, the Black Label action still comes out on top.

In this case, the Priority component reversed the Ranks of the two actions. Black is now the primary action and Green is the secondary action.

The same Prioritization component is also configured to output only the top action.

Therefore, it filters out the Green action altogether, and at the end of our strategy chain, the Black Label is left as our best action.



Creating engagement strategies using customer credit score

Description

A scorecard is a mathematical model that determines a score value based on a set of conditions and a combiner function. The conditions either use customer properties directly, or an expression based on them. The output of a scorecard is the raw score plus a segment, which is obtained by defining a set of cut off values that create a set of score ranges. Learn how a scorecard can be used to determine the customer credit score, and how the credit score can be used in a decision strategy to define suitability rules.

Learning Objectives

- Create a scorecard
- Use the segmentation result and the score value of a scorecard in a decision strategy
- Use a scorecard to determine group-level and action-level suitability

Customer credit score using a scorecard

Introduction

A scorecard is a mathematical model that determines a score value based on a set of conditions and a combiner function. These conditions can either use customer properties directly, or an expression based on them. The output of a scorecard is the raw score and a segment, which is obtained by defining a set of cut-off values that create a set of score ranges.

Transcript

This video explains how a scorecard can be used to determine the customer credit score.

Let's consider a typical loan application scenario for U+ Bank.

The bank wants to automate the home loan requests process by analyzing a customer's credit score based on customer profile information and categorizing it into the right credit score level using a scorecard.

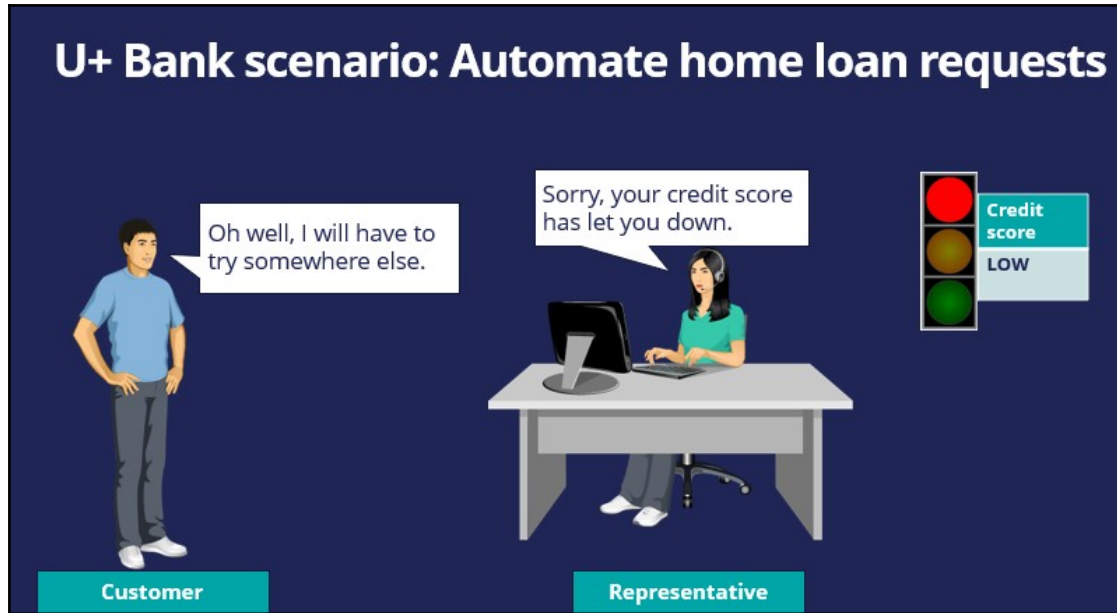
Tom is a U+ Bank customer who visits the bank to apply for a home loan. Iris is a U+ bank loan representative who is responsible for processing loan requests.

Iris gets Tom's profile information. Based on his customer data, Iris calculates his credit score using a scorecard.

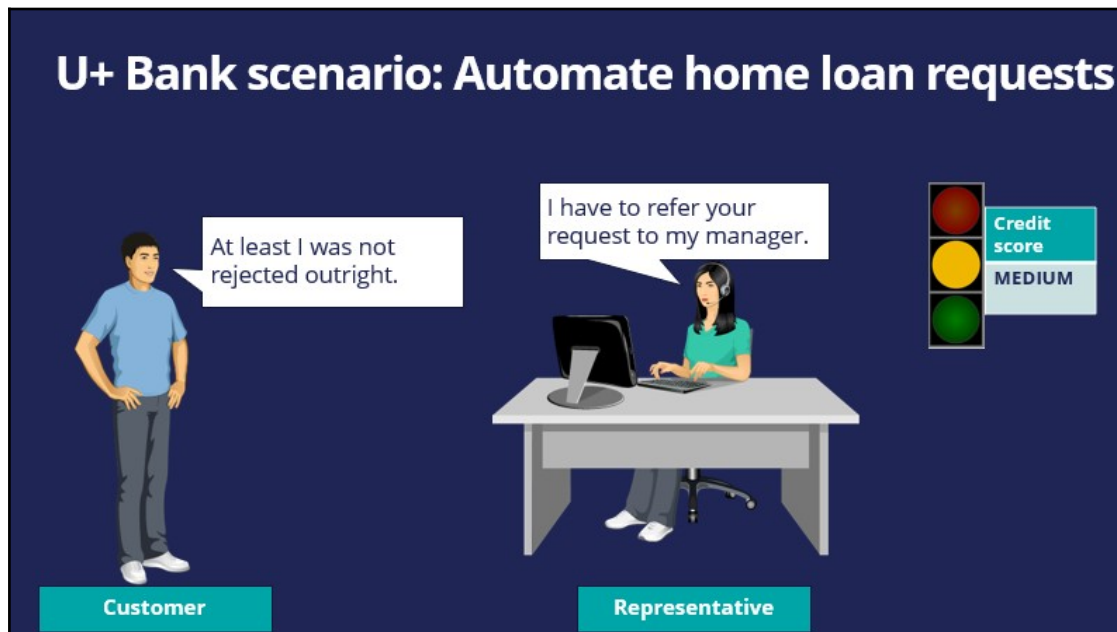
Depending on Tom's credit score, a decision is made on his loan application.

The bank has already defined three credit score levels: **Low**, **Medium** and **High**.

For example, if Tom's credit score is **LOW**, he is not qualified to get a loan, and his application is automatically refused.

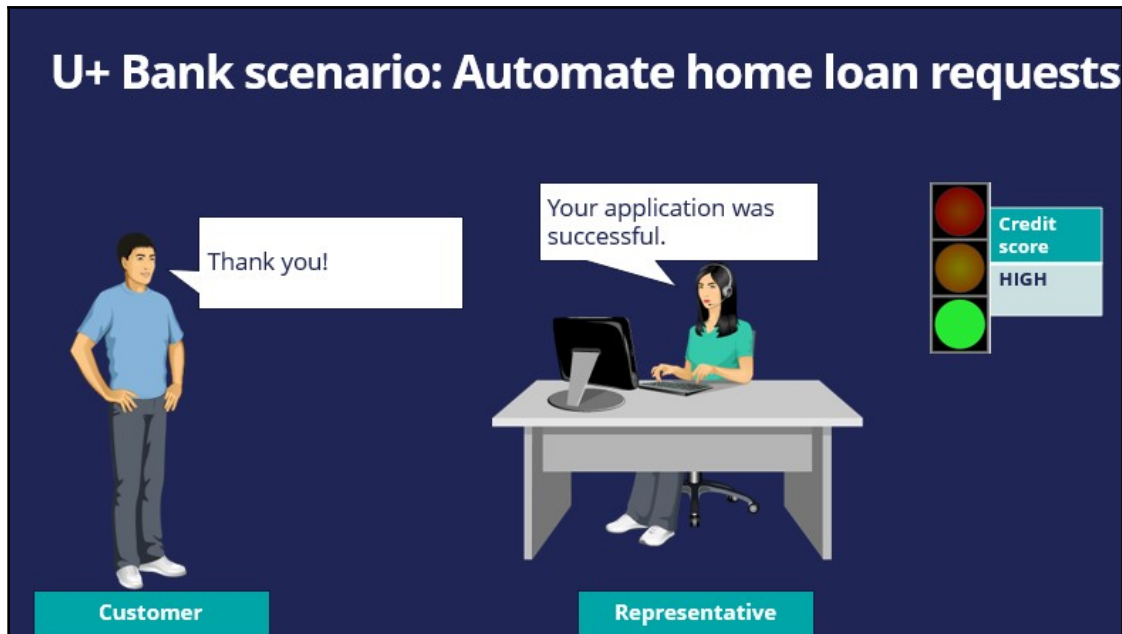


If Tom's credit score is **MEDIUM**, then his application needs to be reviewed further by manager.

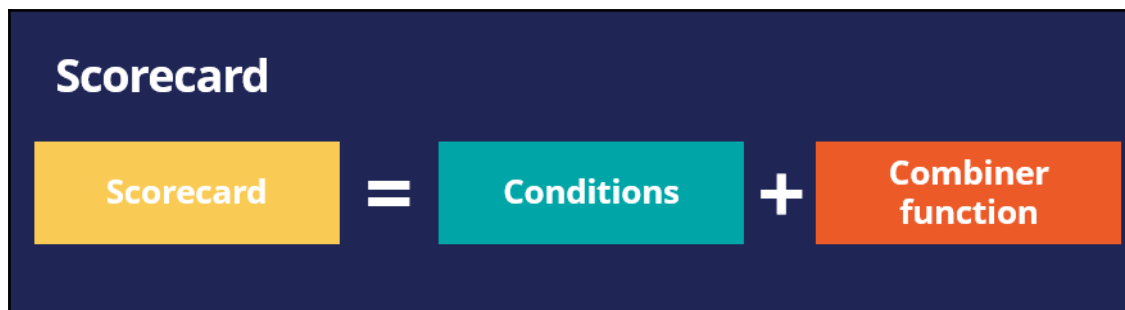


If Tom's credit score is **HIGH**, then he is automatically eligible to get the loan right away.

Let's now learn about the scorecard and how it works by considering the U+ Bank loan application scenario.



A scorecard is a mathematical model that computes a score and outputs a segmentation result based on one or more conditions and a combiner function.



Let's take a look at an example of a scorecard U+ bank could use. Assume that in order to process home loan requests, the bank has identified **three predictors** that can be used in the scorecard calculation. The predictors are **Customer lifetime value, Age, and Account**.

Each predictor has some **conditions** and scores associated with those conditions.

The scorecard enables the bank to assign a **score** to a value or a range of values the predictor can have.

For example, If the **Customer lifetime value** is less than 100, a score of 83 is assigned. If the value is between 100 and 399, a score of 221 is assigned; otherwise a score of 350 is assigned.

If the customers' Age is less than 20, a score of 21 is assigned. If their Age is between 20 and 39, a score of 217 is assigned; otherwise a score of 55 is assigned.

In a similar way, if a customer has an account with U+ bank, a score of 82 is assigned. Otherwise, a score of 10 is assigned.

The combiner function is used to combine the total sum of score values between conditions.

Now that the bank has identified **predictors** and assigned a **score** to a range of values the predictor has, let's see the results of the scorecard.

The output of a scorecard is a **score** and a **segment result**.

First, you will see how to calculate a customer's credit score using the scorecard.

Let's consider an example in which the customer lifetime value is 350, the customer's age is 35, and the customer has a U+ Bank account.

Since the customer lifetime value is 350, he gets a score of 221. For his age, he gets a score of 217, and since he has a U+ bank account, he gets a score of 82. The sum of these three individual scores is 520, representing his final credit score.

The scorecard outputs a **score** as well as a **segment result**.

To process loan requests, the bank has defined three result values. If the credit score is less than or equal to 400, the result is **Not Approved**. If the credit score value is between 401 and 600, the result is **Refer to Manager**. If the credit score is higher than 600, the result is **Approved**. So, the higher a customer's credit score, the better his/her chances of getting a loan approved.

Credit score	Result
< = 400	Not Approved
< = 600	Refer to Manager
Otherwise	Approved

Let's consider the profiles of three customers applying for home loans.

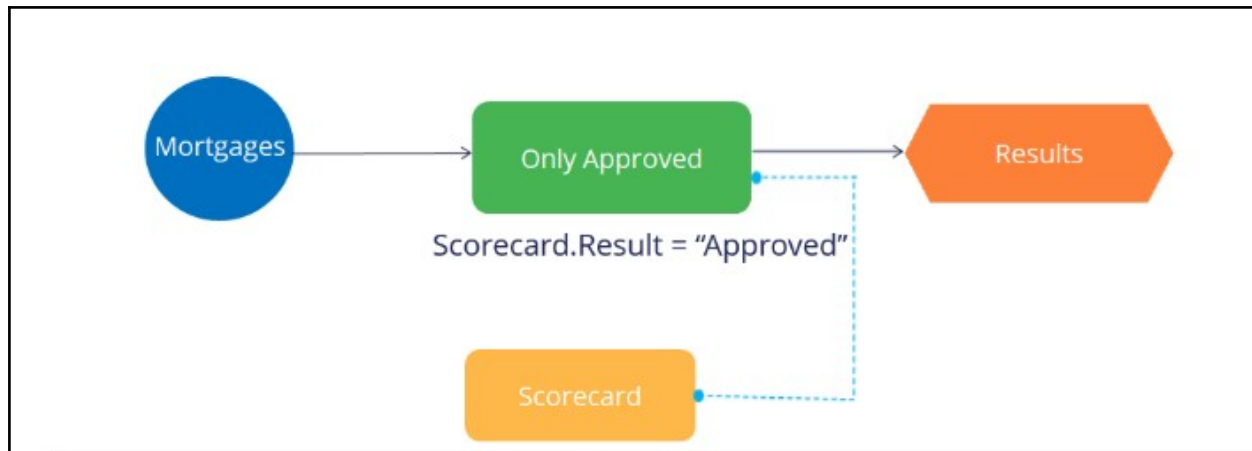
Customer A has a customer lifetime value of 150, their age is 18, and they don't have a U+ Bank account. Therefore, their credit score is 252, and the segment result is **Not Approved**, because their credit score value is less than 400. This means they do not qualify for a loan.

Customer B has a different profile, and their credit score is 415, so the segment result is **Refer to Manager**, because their credit score value is between 401 and 600. This means their application needs further review.

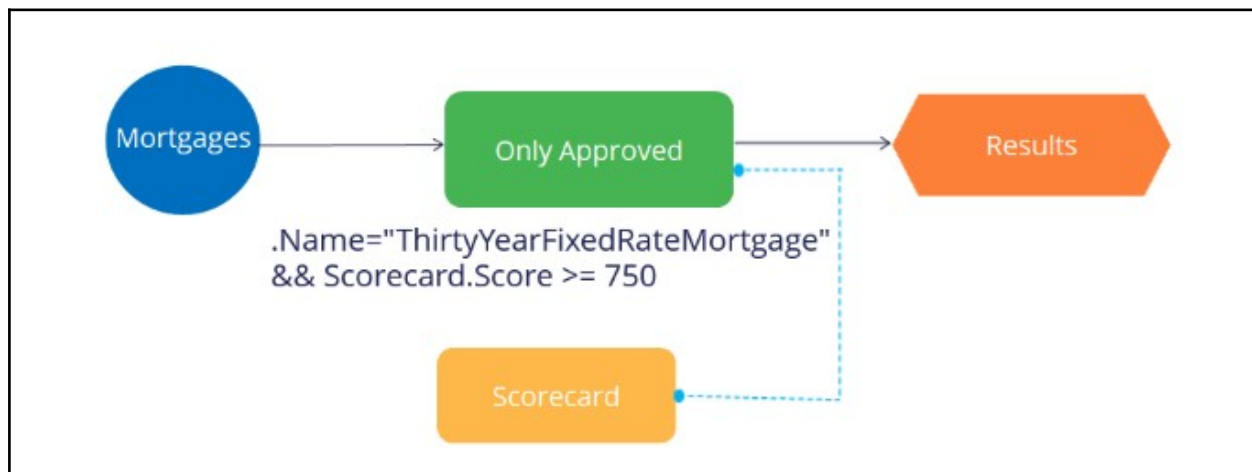
Customer C has a credit score of 649, which is higher than 600, so he is eligible to get a loan right away.

Now that you learned about the scorecard and its results, let's see how to use the scorecard segmentation in a decision strategy.

To use a Scorecard in a decision strategy, use the **Scorecard** component. The **Scorecard** component outputs the result, in this case **Not Approved**, **Refer to Manager**, or **Approved**, which you can use in your decision strategy.



The scorecard also outputs the **raw score value**, this score value can be used directly in the decision strategy.



In summary, the scorecard is a mathematical model that computes a score and outputs a segmentation result based on a set of conditions and a combiner function. The scorecard's score and segmentation result can be used directly in a decision strategy using the Scorecard decision component.

Building a scorecard to calculate the creditscore

Introduction

Scorecard rules are referenced in decision strategies through the scorecard component. Learn how to create scorecard-specific rules to derive decision results from several factors.

Get detailed insight into how scores are calculated by testing scorecard logic using the rule form. Use the test results to view score explanations for all predictors used in the calculation so you can validate and refine the current scorecard design or troubleshoot potential issues.

Transcript

U+ bank wants to build a scorecard that calculates the credit score of a customer to determine if the customer is suitable for a mortgage or not.

The bank has identified three customer properties to use in the scorecard calculation. These are the HasCards (a property that indicates if a customer already has a credit card or not), Income, and Age.

To build a scorecard, navigate to the scorecards landing page and create a scorecard.

Enter a short description for the scorecard.

The Combiner function enables you to select a method for combining scores.



In this scenario, the credit score is the sum of scores attributed to each customer property, so use the Sum method.

The scorecard enables you to assign a score to a value or a range of values the property can have.

For example, the credit card indicator, HasCards property can have the values "Y" or "N".

If a customer has a credit card with U+ bank, assign a score of 100. Otherwise, assign a score of 0.

The screenshot shows a configuration interface for a scorecard rule. It has four columns: Predictor expression, Condition, Score, and Weight. The first row is for the predictor expression ".HasCards". The condition is set to "Y", the score is "100", and the weight is "1". Below this, there is an "Otherwise" section with a score of "0".

Predictor expression*	Condition	Score	Weight*
.HasCards	Y	100	1
	Otherwise	0	

You may also sub-divide values of numeric properties into ranges and assign a score to each range.

The next property is Income. In this case you want to split the values for yearly income into three ranges and assign a score to each range.

The data model contains an Income property, which contains the monthly income of the customer. The scorecard allows you to use this property to build the expression to compute the customer's yearly income.

Once the expression is created, you can start assigning scores to each range. If the customer's yearly income is less than or equal to 25,000, assign a score of 50.

If their yearly income is between 25,001 and 50,000, assign a score of 100; and if their income is between 50,001 and 75,000, assign a score of 150.

If their income is higher than 75,000, assign a score of 200, using the **Otherwise** setting.

The screenshot shows a configuration interface for a scorecard rule. The predictor expression is ".Income*12". There are three rows for conditions: the first row has condition "<= 25000" with score "50"; the second row has condition "<= 50000" with score "100"; the third row has condition "<= 75000" with score "150". Below these is an "Otherwise" section with a score of "200".

Predictor expression*	Condition	Score	Weight*
.Income*12	<= 25000	50	1
	<= 50000	100	
	<= 75000	150	
	Otherwise	200	

In a similar way, split the values for Age into five ranges and assign a score to each range. The higher the range, the higher the score you assign.

Once the Scorecard is defined, you can define the results of the scorecard calculation.

When you refresh, the scorecard calculates the Minimum and Maximum scores.

You may define two or more Result values based on the score. In this scenario, you want the scorecard to return a Result value of 'Not Suitable' if the score is less than 250. Otherwise, the Result value is 'Suitable'.

Result*	Cutoff value	Interval
Not Suitable	250	< 250
Suitable	Otherwise	>= 250

Save the configuration.

Now, run the scorecard and verify the results.

You can either fill in the property values, or you can test the result for a predefined test case using a data transform. For example, select the customer Troy.

The end result for Troy is that he is Not Suitable for a mortgage, as his credit score is 200, which is lower than the set threshold.

Execution results		
Result	Score	Combiner function
Not Suitable	200.0	SUM
Interval	Minimum score	Maximum score
< 250	50.0	500.0

You can see the **Execution details** for Troy, which show how his credit score is computed. Since he has no cards, he gets a score of 0; for his income, he gets a score of 150; and for his age, he gets a score of 50. That's 200 in total.

Execution details							
Predictor expression	Value	Operator	Condition	Score	Weight	Points	Lost points
.HasCards	N		Otherwise	0	1	0	100.0
.Income*12	54000.0	<=	75000	150	1	150	50.0
.Age	26.0	<=	35	50	1	50	150.0

Now, test the results for customer Robert.

Robert is suitable for a mortgage, as his credit score is 250.

Execution results		
Result	Score	Combiner function
Suitable	250.0	SUM
Interval	Minimum score	Maximum score
>= 250	50.0	500.0

The newly created scorecard is now available on the Scorecards landing page.

This demo has concluded. What did it show you?

- How to create a scorecard.
- How to assign a score to categorical and numerical property values.
- How to use expressions in scorecards.
- How to define scorecard outputs.
- How to test scorecards.

Creating an engagement strategy

This demo will show you how to build a decision strategy that can be used in Next-Best-Action Designer to implement more complex Engagement Policy rules.

Engagement Policy rules are business rules that describe which Actions are appropriate for a customer, expressed in the form of either Eligibility, Applicability, or Suitability rules.

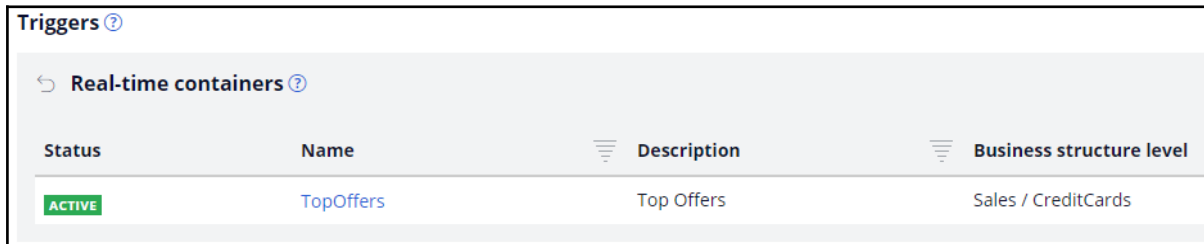
In this video, we will demonstrate the Engagement Policy capability without altering the use case.

Currently, U+ Bank is doing cross-sell on the web by showing various credit cards to its customers. Every time Troy logs in to his accounts page, a credit card offer is shown.

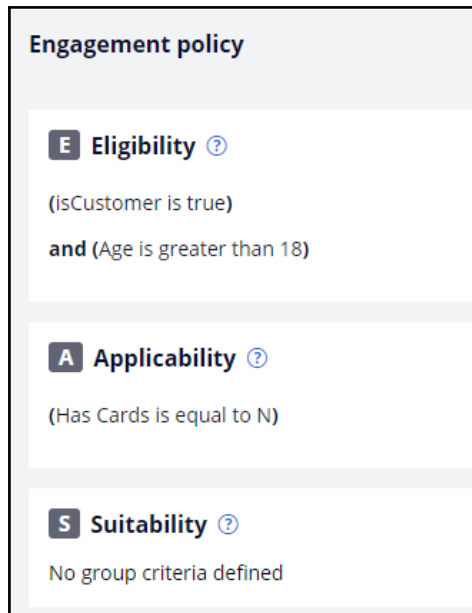
U+ has a new set of Eligibility rules. As a result, Troy only qualifies for the Standard Card offer.

To see the existing configuration, navigate to Next-Best-Action Designer -> Channels.

As you can see, U+ Bank's website invokes the TopOffers real-time container to present offers from the Sales->CreditCards group.



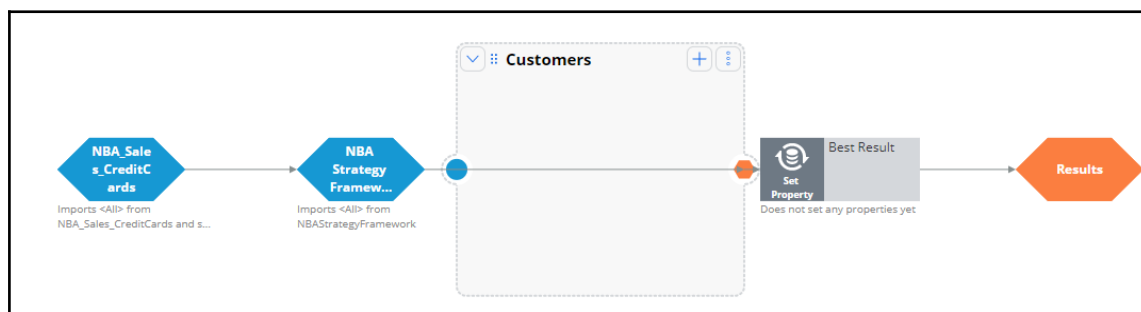
Triggers ?			
Real-time containers ?			
Status	Name	Description	Business structure level
ACTIVE	TopOffers	Top Offers	Sales / CreditCards



In the Engagement Policy, the Eligibility and Applicability rules have been defined to reflect the bank’s current requirements using properties in the criteria.

Sometimes, implementing a specific use case requires a decision strategy.

To understand how the decision strategy is used to enforce Engagement Policy rules, open the Trigger_NBA_Sales_CreditCards strategy. This strategy is associated with the TopOffers real-time container. Every time a decision is requested from the U+ Bank website, this strategy is executed.

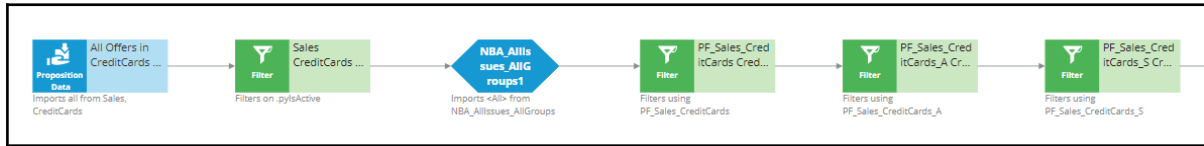


The first component of the strategy is responsible for all Engagement Policy rules.

If you open the strategy, notice that it imports the Actions from the Sales->CreditCards group and then applies the Engagement Policy configuration rules: Eligibility, Applicability, and Suitability.

Each Engagement Policy corresponds to a Filter decision component in the NBA_Sales_CreditCards strategy, and each of these components uses a Proposition Filter rule to implement the rules created in NBA Designer.

A Proposition Filter rule contains the conditions that make certain customers eligible for certain offers.



The decision strategy that implements the new eligibility requirements will be used in this Filter component.

The purpose of this demo is to show the mechanics and required steps for creating an Engagement Strategy, rather than focusing on a concrete use case.

To create a new Engagement Strategy, navigate to the Intelligence -> Strategies and create a strategy with a new canvas.

Enter a short description for the new strategy.

Select the business Issue and Group, and 'Apply to' class. Note that the 'Apply to' class is the Customer class from the Primary Context.

Now, open the strategy.

Enable the External Input component on the canvas to represent all Strategy Results (SR) records that are input into the strategy. The strategy will basically be used as a When rule in a Proposition Filter.

Connect the External Input component to the Results component.



Save and test the strategy with the Troy data transform.

For external input you can use the AllCreditCards strategy, as it imports all Sales->Credit Cards. The test shows that all credit card Actions will reach the Eligibility Strategy, nothing is filtered out by the framework.

Notice that the strategy outputs various credit card offers for Troy.

Complex eligibility rules can be implemented in this decision strategy.

For now, to keep it simple and show the mechanics of the Engagement Strategy, consider that customers qualify only for the Standard Card. Everything else is filtered out.

To implement this, add a Filter component to filter out all credit cards except the Standard Card.

Now, configure the Filter component.

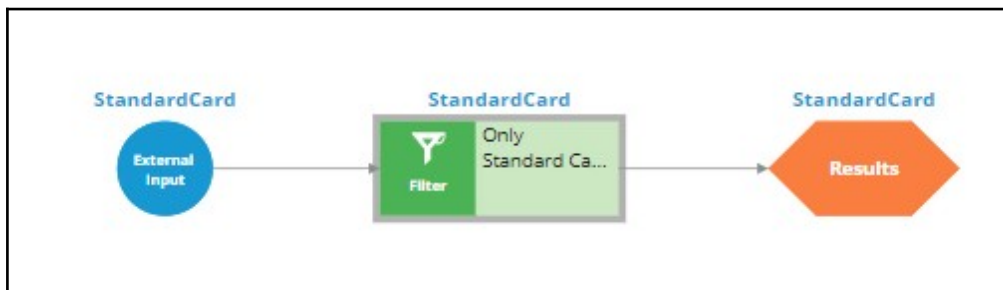
Start by naming the Filter component.

Click the gear icon to open the Expression Editor.

The Filter component should select the Action for which the pyName property is equal to "StandardCard".

Connect the components on the canvas and re-test the strategy.

Examine the output of the Filter component. Now the Filter component outputs only the Standard Card. All other Actions are filtered out.



Make this strategy accessible in Next-Best-Action Designer as an Engagement Strategy.

Once the Record is added, navigate to NBA Designer and open the Engagement Policy to configure this strategy as an Eligibility Strategy for the Group-level Eligibility rules.

The screenshot displays the 'Engagement policy' configuration screen. Under the 'Eligibility' section, three conditions are defined:

- Condition 1: `isCustomer` is true.
- Condition 2: `Age` is greater than 18. A [Select values](#) link is present next to the value field.
- Condition 3: `Engagement Stra...` has results for `Only Standard Card`.

The condition is: Engagement Strategy has results for Only Standard Card.

Saving this completes the required configurations.

Back on the U+ bank website, log in as Troy to see that the Standard Card offer is displayed. Everything else is filtered out by the Engagement Strategy you just created.

This demo has concluded. What did it show you?

- How to build a decision strategy that can be used as an Engagement Policy rule.
- How to define Eligibility rules using a decision strategy in Next-Best-Action Designer.

Using a Scorecard in a Decision Strategy

Introduction

Learn how to use the segmentation result and the score value from a scorecard in a decision strategy. Learn how suitability rules can be defined to reflect the bank's requirements using a decision strategy.

Transcript

Currently, U+ Bank is doing cross-sell on the web by showing various credit cards to its customers.

The bank wants to implement a new requirement: All credit cards are suitable only for customers who have a credit score greater than or equal to 250.

To implement this requirement, use an Engagement Strategy.



U+ already created a Scorecard that computes the customer credit score. To use the Scorecard in the decision strategy, add a Scorecard component to the canvas and configure it.

Select the Scorecard model **DetermineCreditScore**, which U+ already created.

If you open this Scorecard, you can see how the credit score is computed. Note that the 3 customer properties are used and a score is assigned to the value or range of values the property can have.

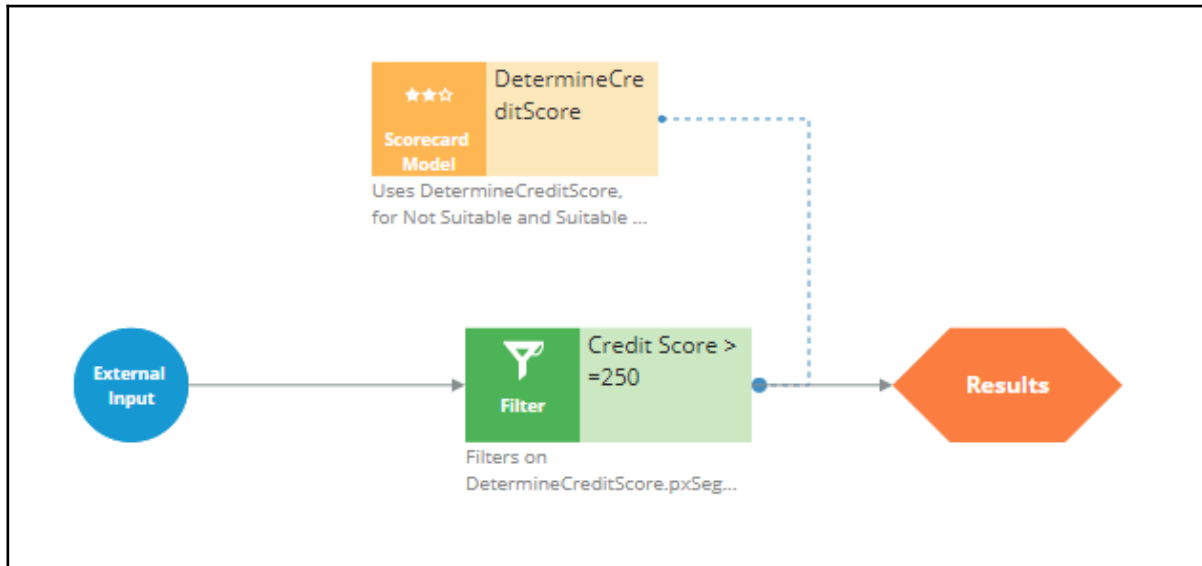
You can also see the segment results. The Scorecard returns value **Not Suitable** if the credit score is less than 250. Otherwise, it outputs the result **Suitable**.

To implement the new requirement, use a Filter component to express the condition under which the Actions are suitable.

You want this strategy to output something only if the result of the scorecard is "Suitable". The result of the Scorecard is stored in the pxSegment property. Therefore, set the Filter condition to `DetermineCreditScore.pxSegment=="Suitable"`.

If the result of the Scorecard is Not Suitable, this strategy has no results.

Note that to reference a property from a component that is not connected to the Filter component input, use the <ComponentName>.<PropertyName> construct.



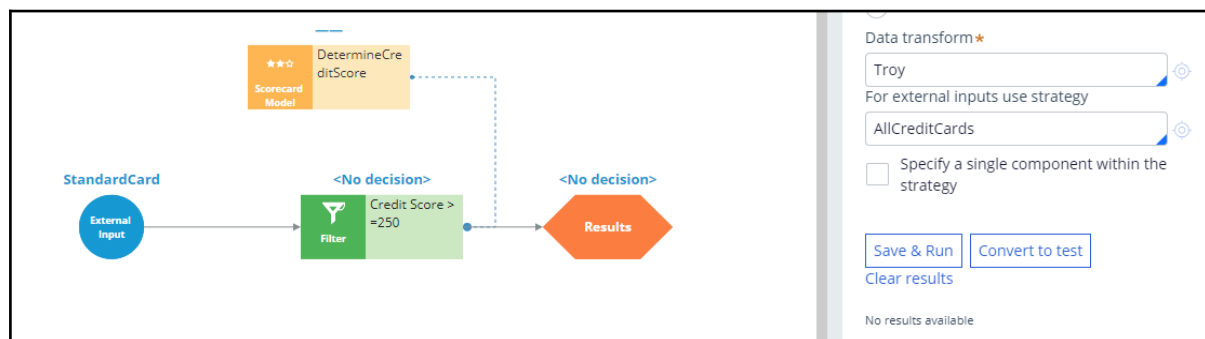
Note that the usage of the External Inputs component also gives you the ability to create more complex conditions, where Action attributes are also used.

Save the configurations.

Now, test the strategy using the customer profiles, Troy and Robert.

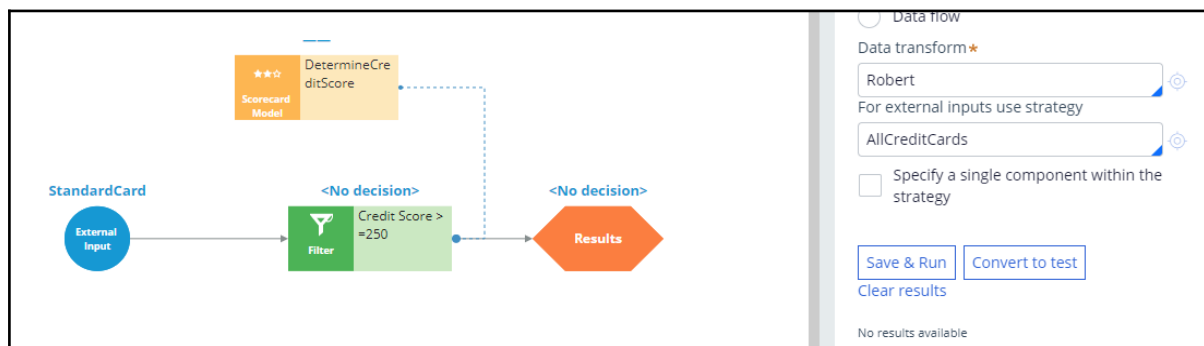
For external inputs, consider all credit cards available.

The result of the Scorecard is: **Not Suitable**. Therefore, the strategy did not output any results for Troy.



Field	Value
Segment	Not Suitable
ApplyAnalytics	---
Benefits	---
Bundle Parent	---
BundleType	---
Component	DetermineCreditScore

Now, repeat the test to verify results for the Robert data transform.

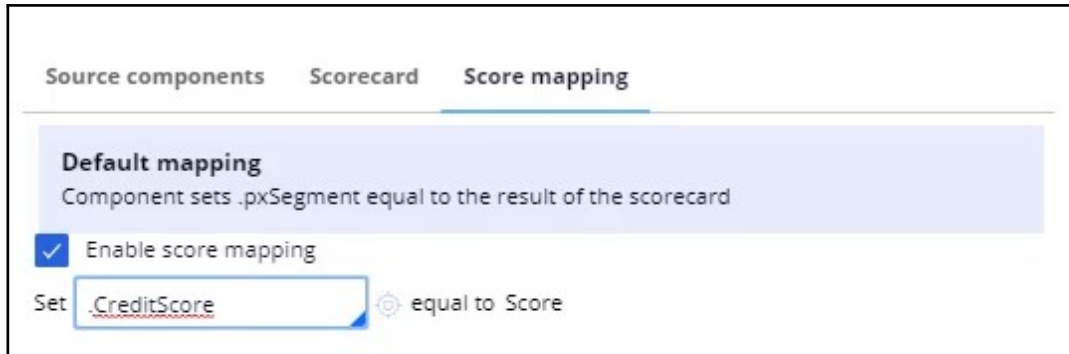


Field	Value
Segment	Not Suitable
ApplyAnalytics	---
Benefits	---
Bundle Parent	---
BundleType	---
Component	DetermineCreditScore

The strategy is also not outputting any results for Robert, as the Segment value is **Not Suitable**.

Now, assume the credit score value for customers is already computed and presented in the Customer data model. However, this value is not set for certain customers, in which case you want to use the credit score determined by the Scorecard itself.

To make these adjustments, start by opening the Scorecard component and mapping the score computed by the Scorecard to the CreditScore property.

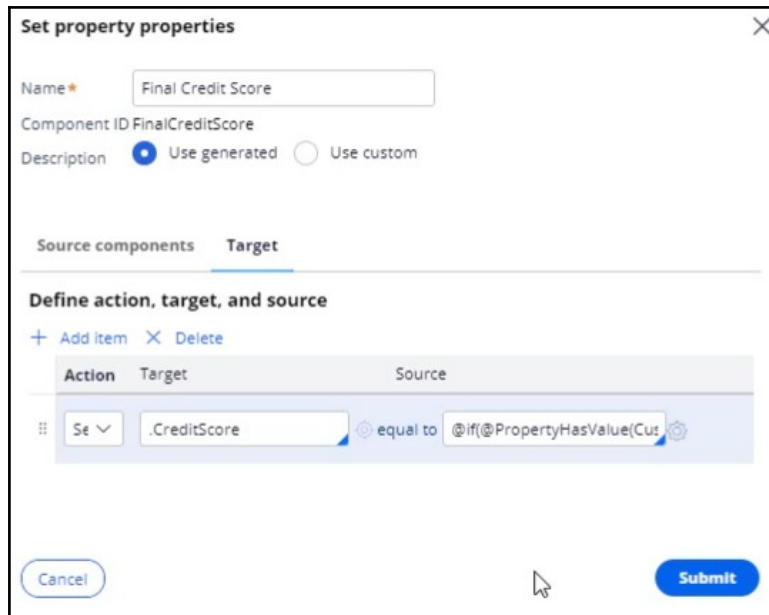


Then, add a Set Property component to determine the actual value of the credit score, given that the credit score is already available for certain customers.

Use 'Add item' to set the CreditScore to either the available credit score value from the Customer data model, or to the value computed by the Scorecard.

Define the Expression as *if(@PropertyHasValue(Customer.CreditScore), Customer.CreditScore, .CreditScore)*.

If set, this Expression will result in the credit score from the Customer data model. If not, the credit score value will be computed by the Scorecard.

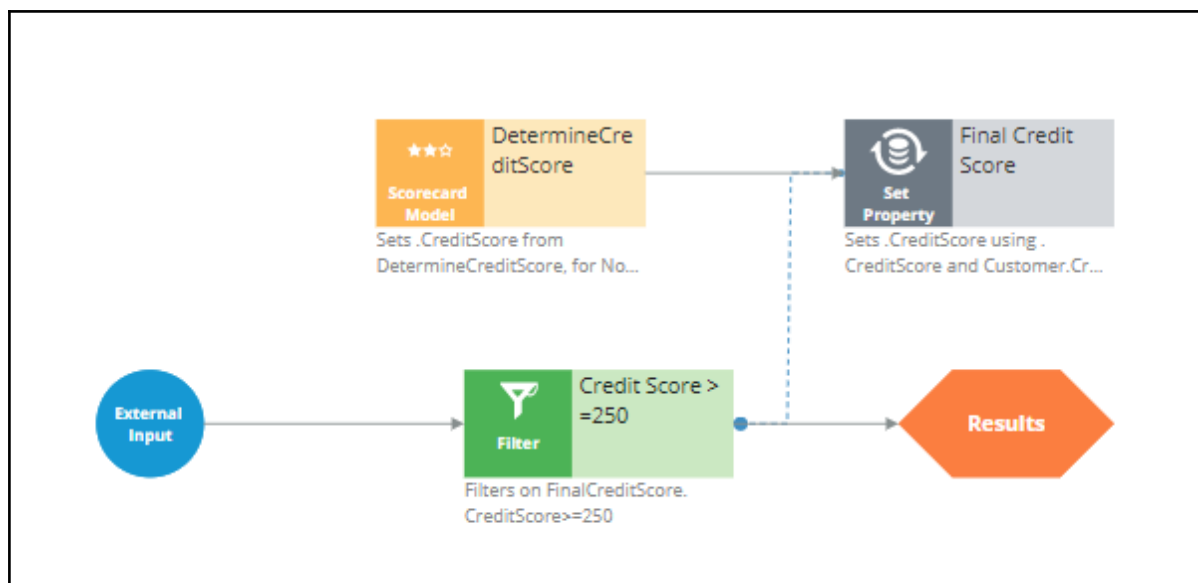


Once the Set Property component is configured, open the **Filter** component properties to modify the Filter condition. In this scenario, the bank has decided to present various credit cards to suitable customers who have credit scores greater than or equal to 250.

So, open the 'Expression builder' to modify the condition as *FinalCreditScore.CreditScore>=250*.

Now, connect the Scorecard Model component to the Set Property component to ensure the CreditScore value determined by the Scorecard is copied to the Set Property component.

Save the configurations.



Re-test the strategy for the Troy and Robert data transforms.

Note that the strategy is not outputting any results for Troy, as his credit score, 200, is less than 250.

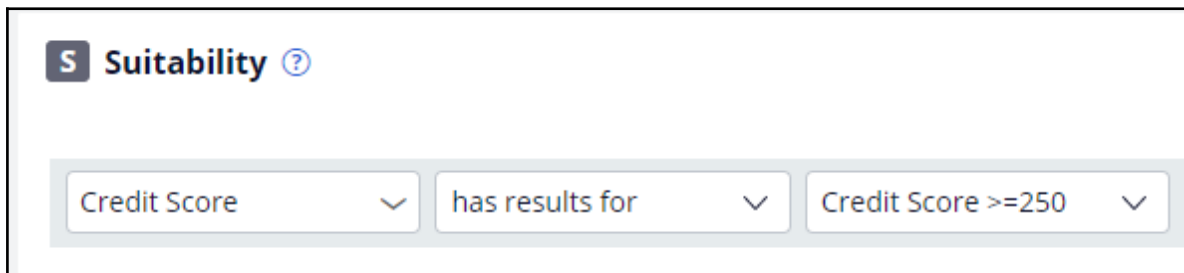
For Robert, the strategy is outputting results.

If you open the Robert data transform, note that the CreditScore in the data model is set to 600.

The Set Property component picks up the credit score value available in the data model (that is, CreditScore = 600) and not the value computed by the Scorecard (CreditScore = 200). That is why the strategy is outputting results for Robert.

Since you have completed configuring the strategy, check it in, so that the strategy will be available to the U+ bank website.

You can now configure this strategy in the Next-Best-Action Designer Engagement Policy as a suitability condition for the Group-level Suitability rules.



Select the strategy. The condition is: *Credit Score has results for the Credit Score >=250.*

Saving this completes all the required configurations.

On the U+ bank website, if you log in as Troy, notice that no credit card offer is displayed. This is because no credit cards are suitable for Troy.

Now, if you log in as Robert, notice that the credit card offer is displayed.

This is because credit cards are suitable for Robert.

This demo has concluded. What did it show you?

- How to use the segmentation result and the score value of a Scorecard in a decision strategy.
- How to use the *PropertyHasValue* function to check if a Customer property has value or not.

Creating eligibility rules using customer risk segments

Description

Decision tables enable you to better adjust to the variables in your business processes. Learn to create decision tables that help you derive a value that has one of a few possible outcomes, where each outcome can be detected by a test condition. Decision tables list two or more rows, each containing test conditions, optional actions, and a result.

Learning Objectives

- Create a decision table
- Use the outcome of a decision table in a decision strategy

Creating customer risk segments using a decision table

Business scenario

U+ Bank is currently doing cross-sell on the web by showing various credit cards to its customers. The bank already uses a customer's credit score to determine their suitability for a credit card. The new eligibility rules require customers to be divided into risk segments varying from AAA to CCC. To start, customers in the risk category BB- and CCC are not eligible for credit cards. Customers from all other risk segments are eligible.

The risk segments are determined by two parameters: **Outstanding loan amount** and the customer **Credit score**.

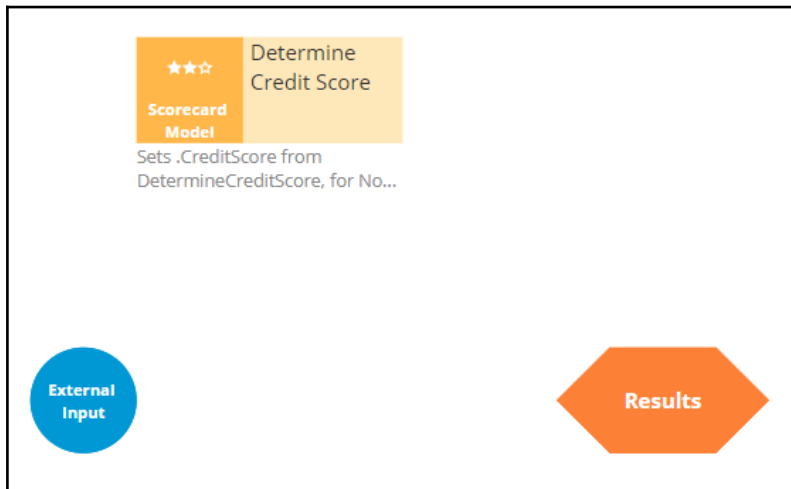
Condition	Risk Segment
If Outstanding loan amount < \$10000 AND Credit score is > 600	AAA
If Outstanding loan amount between \$10000 and \$25000 AND Credit score is > 600	AAA-
If Outstanding loan amount between \$25000 and \$50000 AND Credit score is > 600	AA
If Outstanding loan amount > \$50000 AND Credit score is > 600	AA-
If Outstanding loan amount < \$25000 AND Credit score is between 400 and 600	BBB
If Outstanding loan amount between \$25000 and \$50000 AND Credit score is between 400 and 600	BBB-
If Outstanding loan amount > \$50000 AND Credit score is between 400 and 600	BB-
If Credit score is between 200 and 400	CCC
If Outstanding loan amount AND Credit score falls in any other range	CCC

To implement the new bank regulations, use an Engagement Strategy to:

1. Calculate credit score.
2. Define risk segment.
3. Filter credit cards based on risk segment.

Calculate credit score

U+ has already created a Scorecard, **DetermineCreditScore**, which computes the customer's credit score. You can use the Scorecard in the decision strategy by adding a Scorecard component to the canvas.



The dialog box is titled "Scorecard model properties" and has a close button (X) in the top right corner. It contains the following fields and options:

- Name*: Determine Credit Score
- Component ID: DetermineCreditScore
- Description: Use generated Use custom
- Source components: Scorecard (selected), Score mapping
- Defined on: Applies to Strategy result
- Scorecard model: DetermineCreditScore (with a search icon)

At the bottom of the dialog are "Cancel" and "Submit" buttons.

Since you need the raw score, enable **Score mapping** and set the Score value in the Credit Score property.

Scorecard model properties ✕

Name *

Component ID DetermineCreditScore

Description Use generated Use custom

Source components Scorecard **Score mapping**

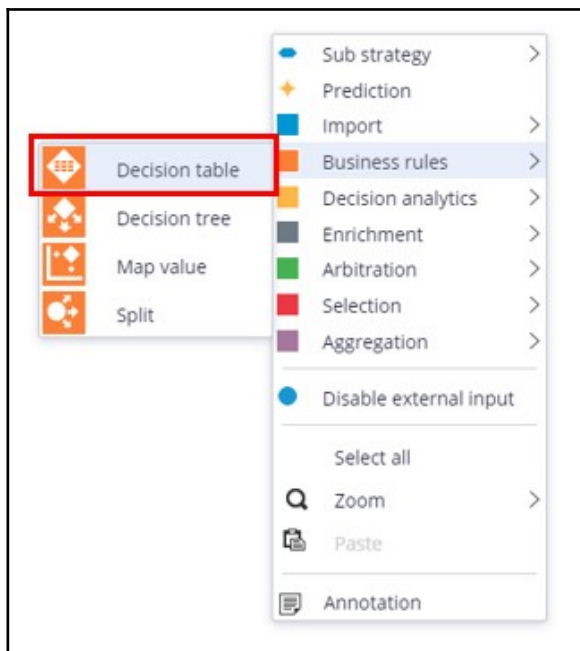
Default mapping
Component sets .pxSegment equal to the result of the scorecard

Enable score mapping

Set equal to Score

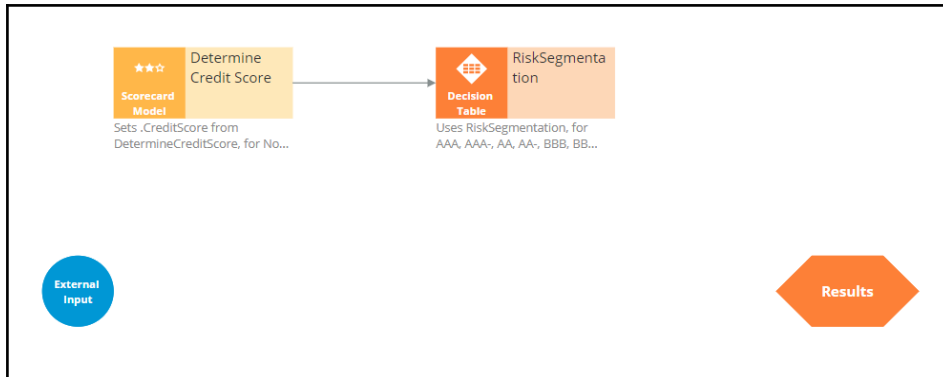
Define risk segment

To implement the risk segmentation requirements, use a **Decision table** component from the **Business rules** category, which outputs the customer risk segment.



The Decision Table can be defined on the Strategy Result, SR class, or a Customer class. The choice often depends on where the properties used in the Decision Table are located. In

this case the outstanding loan amount is a Customer property and the Credit Score calculation is an SR property, so both options are valid.



Decision Table Properties

Name*

Component ID DecisionTable

Description Use generated Use custom

Source components **Decision table**

Default mapping
Component sets .pxSegment equal to the result of the decision table.

Defined on Applies to Strategy result

Decision table

To use Credit Score as a parameter, you first need to define it on the **Parameters** tab. For this scenario, define the Customer class and reference it from the DT component.

Table		Results	<u>Parameters</u>	Pages & Classes	Test cases	Specifications
Name	Description	Data type				
<input type="text" value="CreditScore"/>	<input type="text" value="Customer's credit score"/>	<input type="text" value="Integer"/> <input type="text" value="v"/> <input type="text" value="🗑️"/>				

Creating the table requires two condition columns. First is the **Principal Loan**, which is the property in the data model representing the outstanding loan amount.

The Operator represents the condition applied to the column. In this case, **greater than**, which allows you to express the **Outstanding loan amount** condition from the requirement.

Select a Property

Property:

Label:

Use Range:

Start Range:

End Range:

The second condition column is the Credit Score property. The Credit Score is a parameter, so you need to use the **Param.PropertyName** construct.

The **Use Range** checkbox groups the credit scores together.

Decision Table property chooser

Select a Property

Property:

Label:

Use Range:

Start Range:

End Range:

Next, you need to specify the possible outputs of the Decision Table in the **Results** tab.

Results

Results defined by property

> **Additional Allowed Results.** When selected, each target property will be assigned the value specified

The image shows a vertical list of five configuration rows. Each row consists of a 'Result' input field on the left and a 'Target property' input field on the right. The 'Result' fields contain the following values from top to bottom: 'AAA', 'AAA-', 'AA', 'AA-', and 'BBB'. Below each 'Result' field is a trash can icon and a plus sign in a circle. The 'Target property' fields are currently empty.

After adding the **Target properties** and **Results**, the Table will look like the following.

Conditions		Actions
<ul style="list-style-type: none"> Outstanding Loan Amount 	<ul style="list-style-type: none"> Credit Score 	Return
>=	<=	
if		→
otherwise		→ AAA

Fill in the bank's requirements and specify a **Return** value for each row in the table.

	Conditions				Actions
	Outstanding Loan Amount		Credit Score		Return
	>=	<=	>=	<=	
if		10000	600		→ AAA
else if	10000	25000	600		→ AAA-
else if	25000	50000	600		→ AA
else if	50000		600		→ AA-
else if		25000	400	600	→ BBB
else if	25000	50000	400	600	→ BBB-
else if	50000		400	600	→ BB-
else if			200	400	→ CCC
otherwise					→ CCC

Note, if you leave a value blank it is ignored by the Decision Table. For example, leaving the Credit Score value blank, means that credit score comparison always returns a value of True.

If none of the conditions are met, for example, if the loan amount is zero and the credit score is 50, the Otherwise path is taken; in this example the result will be CCC.

It is important to note that once a Decision Table condition is satisfied, the processing stops. For example, if the loan amount is 30,000, and the credit score is 650, the processing stops at row three returning the result "AA". The other rows are not processed.

After the Table is configured, go back to the property panel of the Decision Table component and map the Decision Table parameter to a Strategy property.

Decision table properties

Source components **Decision table**

Default mapping
Component sets .pxSegment equal to the result of the decision table

Defined on Applies to Strategy result
PegaCRM-Data-Customer

Decision table

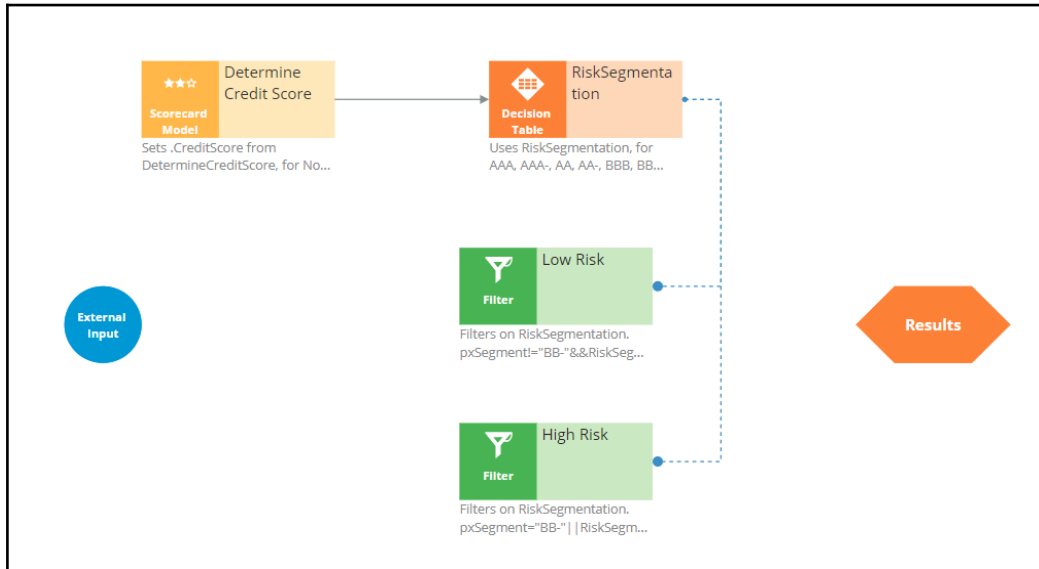
Supply data via

Parameters

CreditScore

Filter credit cards based on the calculated risk segment

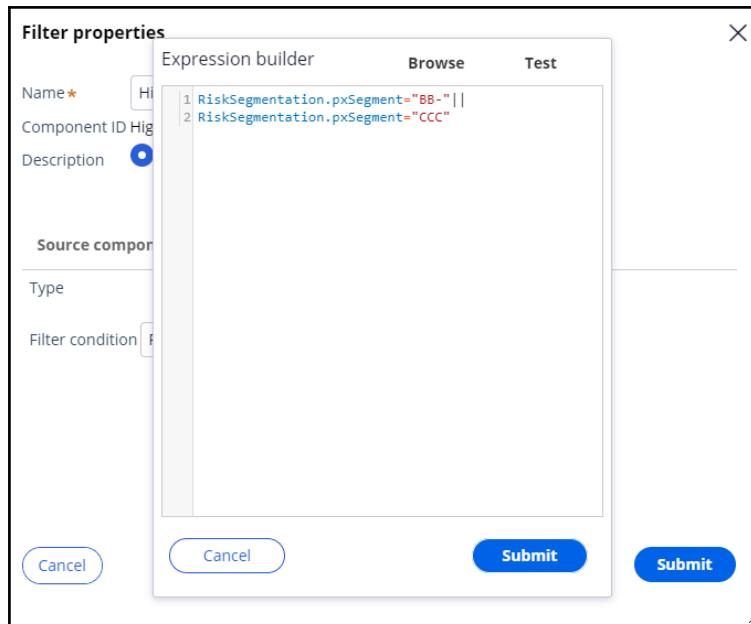
Configure two Filter components to ensure that you identify High Risk and Low Risk customers.



Customers in the risk category BB- and CCC are not eligible for credit cards, but all other customers are eligible. The **pxSegment** Strategy property contains the output of the Decision Table. So, create relevant expressions.

Low Risk Filter: *RiskSegmentation.pxSegment!="BB-"&&
RiskSegmentation.pxSegment!="CCC"*

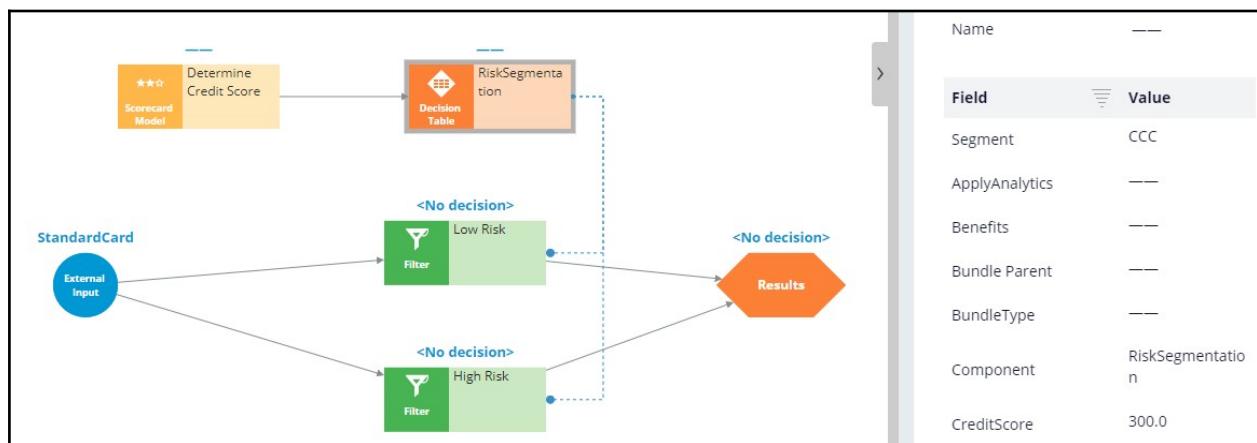
High Risk Filter: *RiskSegmentation.pxSegment="BB-" | |
RiskSegmentation.pxSegment="CCC"*



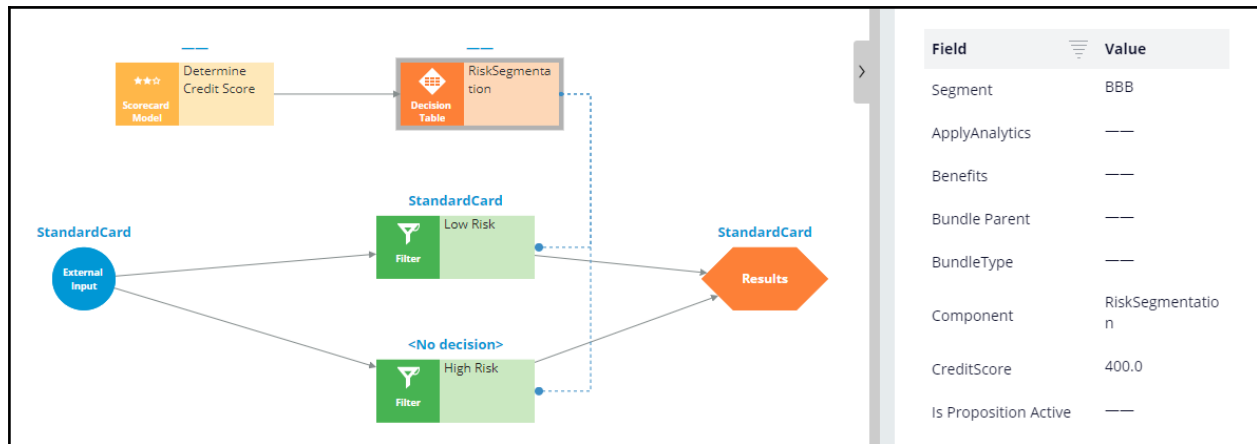
When you test the strategy using the Robert data transform, you will see that Robert falls under the category of high-risk customers, as he has a huge outstanding loan amount, over 50,000.



Therefore, you can expect his risk segment to be CCC.



When you test the strategy for Arnold, who has an outstanding loan of 8000 and a credit score of 400, the Decision Table correctly classifies Arnold in the BBB risk segment and allows all credit cards for him.



Define the Eligibility criteria

Once the decision strategy is ready, you can complete the Eligibility criteria definition in Next-Best-Action Designer.

Risk Segmentation has results for Low Risk

E Eligibility ?

(isCustomer is true)
and (Age is greater than 18)
and (Risk Segmentation has results for Low Risk)

Using predictive models

Description

Predictive analytics requires historical data that contains the customer behavior you want to predict. A predictive model is used to predict an outcome based on customer behavior such as offer acceptance and churn based on customer characteristics such as credit risk, income, product subscriptions, etc. A predictive model component references a predictive model, which in turn predicts customer behavior.

Learning objectives

Describe predictive analytics

Describe how to use a predictive model in a decision strategy

Arbitrate between business issues using applicability rules in Next-Best-Action Designer

Predictive models drive predictions

With the decision management capability of Pega Platform™, you can enhance applications to help optimize business processes, predict customer behavior, analyze natural language, and make informed decisions to better meet customers' needs and to achieve positive business outcomes.

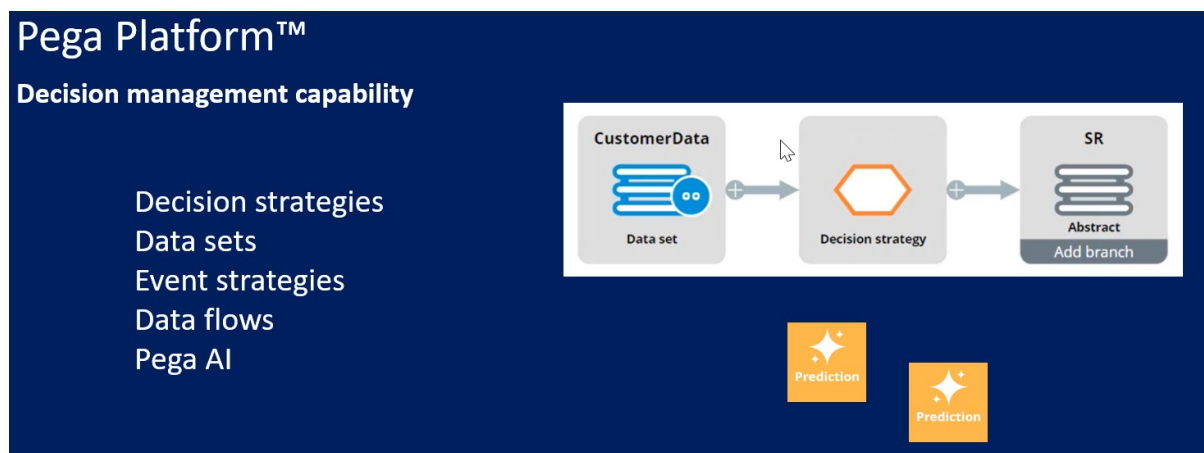
Transcript

This video introduces you to Pega AI, a feature of the decision management capability of Pega Platform™.

Other decisioning features of the Pega Platform include:

- Decision strategies to improve customer experience and deploy intelligent processes based on behavioral and operational data and data sets to read and write the data used in the decision strategies.
- You can use event strategies to detect patterns in data streams and react to them.
- And to ingest, process, and move data from one or more sources to one or more destinations, you can configure data flows as scalable and resilient data pipelines.

Decision management uses Pega AI to make predictions about customer behavior, successful case completion, the topic of an incoming message, or other subjects to make the decisions more relevant.



Decision management is a Pega Platform capability. You can apply decision management to any application that is built on Pega Platform.

Predictions differ to suit the domain they are used in, but one or more predictive models drive them all.

A data scientist can create a predictive model in Pega Platform or an external environment that can export the model as a PMML or H2O file. Another option is to connect to a machine learning service such as Google ML or AWS SageMaker.



If an insurance company wants to use Pega Process AI™ to route incoming claims that might be fraudulent to an expert based on the outcome of a predictive model ...

... the data scientist creates a fraud model to drive a new case management prediction in Prediction Studio.



Prediction Studio is the dedicated workspace where you manage the life cycle of predictive models and the predictions they drive.

A prediction is a hand-off to an application developer, who can then use the prediction in a decision step in the case type to route cases more accurately. This strengthens the separation of concerns.

You can use Pega Customer Decision Hub™ to make next-best-action decisions for your customers.

Customer Decision Hub predictions can predict customer behavior, such as which customer is about to churn ...

... or predict the likelihood that a customer clicks on a web banner to support the decision on which banner to show to a customer.

Pega Adaptive Decision Manager (ADM), a key component of the decision management capability ...

... allows a data scientist to configure self-learning, adaptive models that continuously improve predictions about business processes and customer behavior.

An adaptive model rule typically represents many adaptive model instances because each unique combination of the model context generates a model.

In Customer Decision Hub, adaptive models drive many predictions that come with the product out of the box, such as the Predict Web Propensity prediction that predicts the likelihood that a customer clicks a web banner.

Customer Decision Hub predictions have several features specific for the domain ...

... such as a control group for which the prediction outputs a random propensity instead of the propensity that is generated by the adaptive model instance.

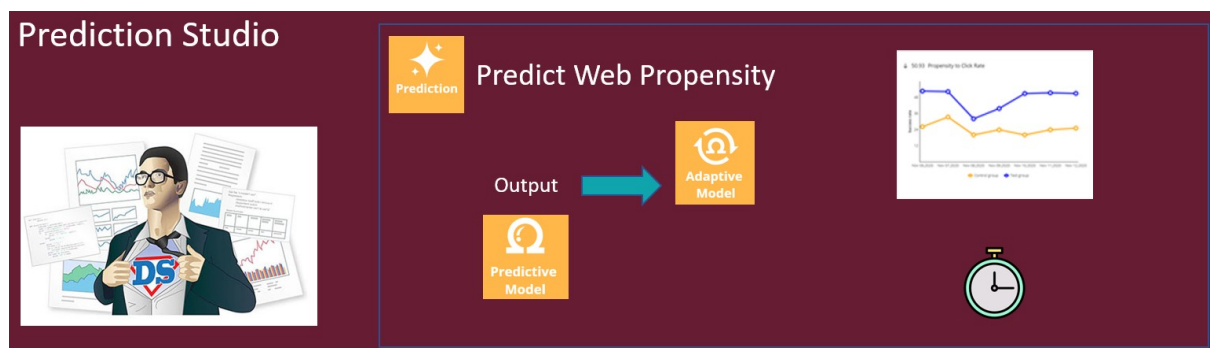
Comparison of the control group and the model propensity-based group allows you to measure the lift in a success rate that the AI generates, an important business metric.

Also, Customer Decision Hub predictions feature a response timeout setting. After the timeout expires, a negative response is recorded.

The response timeout setting depends on the use case. For example, in a web use case, several minutes suffice ...

... while in an outbound email campaign, the response timeout is set to several days to allow customers enough time to respond.

You can further enhance the prediction by using the output of a predictive model as a predictor in the adaptive model.



The Pega Customer Service™ application uses the natural language processing capability of decision management to analyze incoming text and route the messages based on the topics and entities detected.

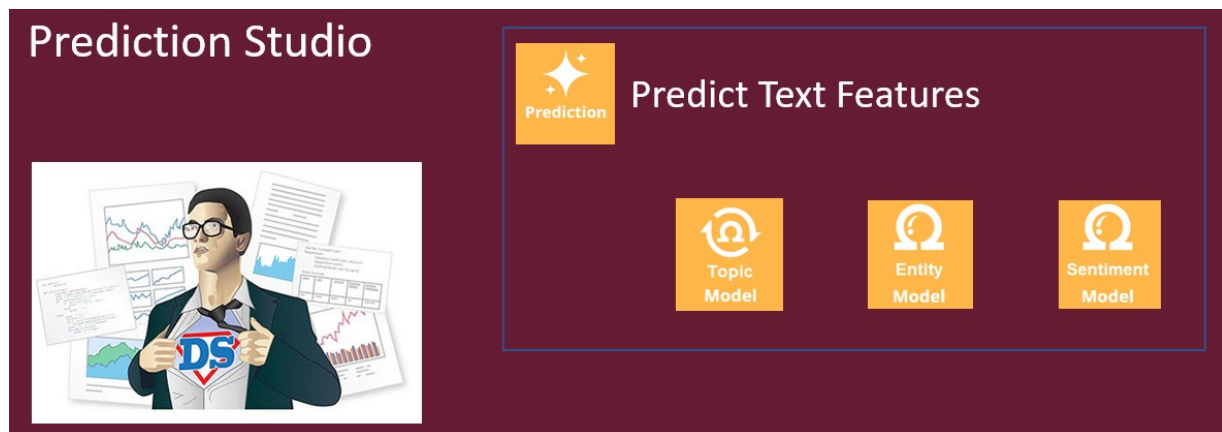
Pega Customer Service uses Text analytics predictions that are distinctly different from both case management predictions and Customer Decision Hub predictions.

Text analytics predictions use predictive models to detect the topic of an incoming message that the application can use to optimize the routing of the message to the relevant department.

Secondly, text analytics predictions use entity extraction models that qualify text as, for example, an account number, a postal ZIP code, or an address.

The application can use this information to fill relevant fields in a case automatically.

Finally, the text analytics predictions come with a sentiment model that can route or prioritize negative messages to improve the customer experience.



Feedback on the detected topics, entities, and sentiment by CSRs improves the performance of the text analytics prediction over time.

This video has concluded. What did it show you?

- Pega AI allows you to improve business processes and customer engagement by using predictions.
- Predictive models drive the predictions.
- The predictive models can be static or adaptive.
- Predictions are managed in Prediction Studio.

Arbitrating across business issues

Pega Customer Decision Hub™ combines analytics, business rules and customer data to make intelligent decisions. Every next best action weighs customer needs and business objectives to optimize decisions.

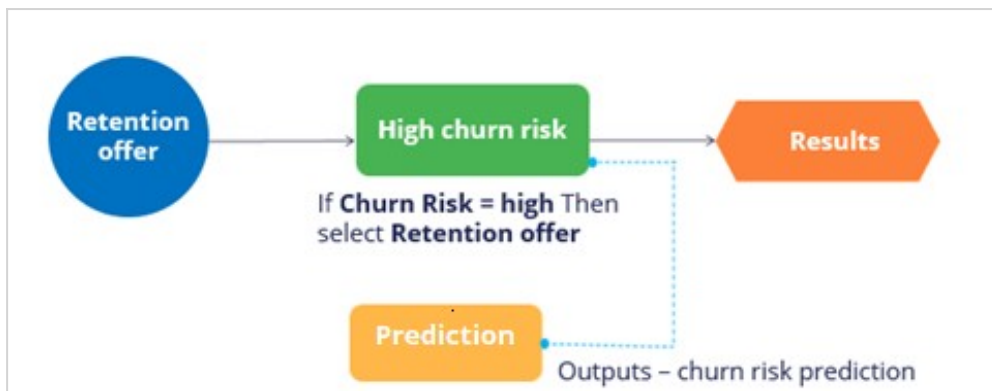


To arbitrate across multiple business issues, you create a decision strategy and use this strategy to define applicability conditions in Next Best Action Designer.

For example, consider a bank who is already doing sales. Now it wants to proactively offer retention offers for customers with high churn risk.

To predict the churn risk, a data scientist creates a churn prediction that calculates the likelihood that a customer will soon leave the bank.

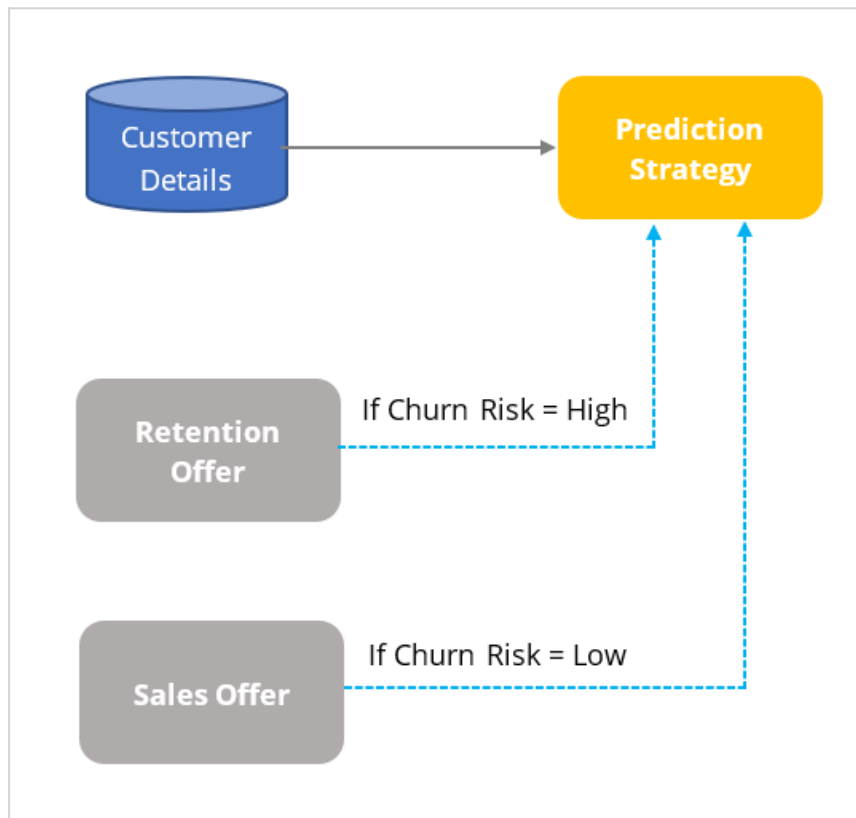
You create a decision strategy that differentiates between high risk and low risk customers that references the churn prediction.



In Next Best Action Designer, you use the decision strategy to define applicability rules for the sales offers and the retention offers.

When the customer has a low churn risk, sales offers are applicable.

Conversely, when the customer has a high churn risk, retention offers are applicable.



Using predictions in engagement strategies

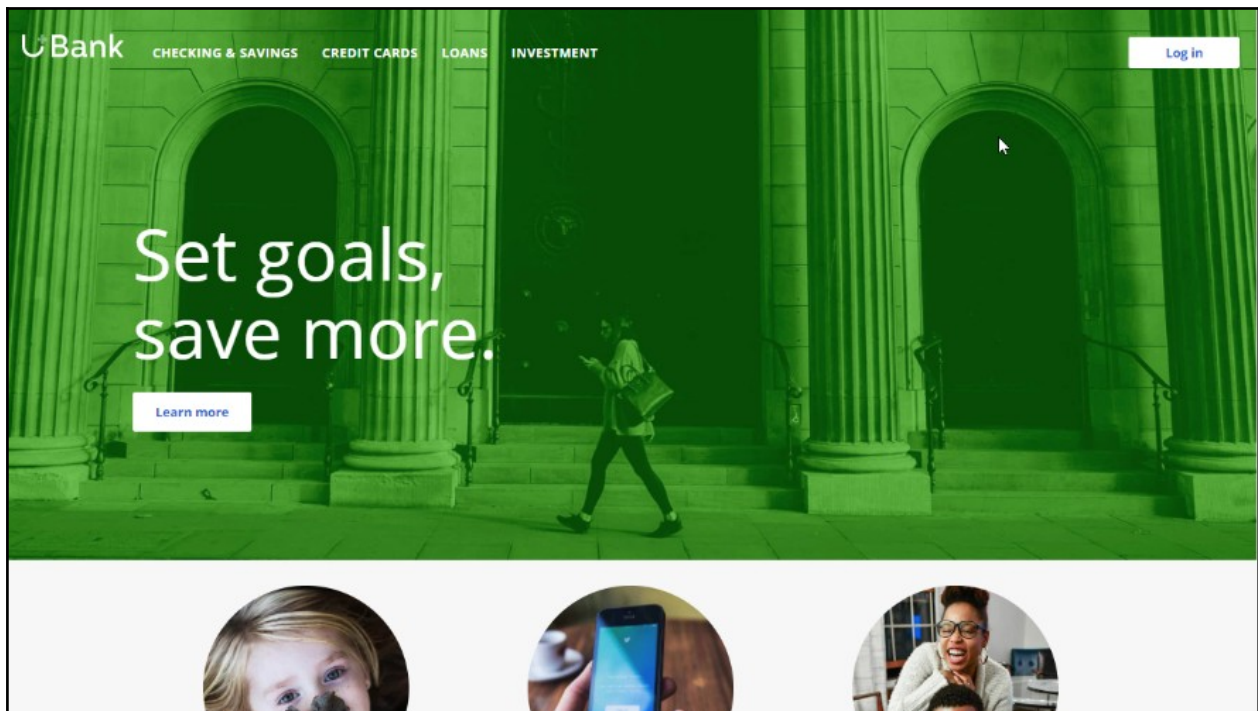
Introduction

A prediction is used to predict customer behavior such as offer acceptance and churn based on characteristics such as credit risk, income, and product subscriptions. Learn how to arbitrate between different groups of actions to display more relevant offers to customers. Gain experience using a prediction in a decision strategy and learn how applicability rules can be defined to reflect the bank's requirements in a decision strategy.

Transcript

This demo shows you how to use a prediction in an engagement strategy to determine customer applicability for a retention offer.

Currently, U+ Bank is cross-selling on the web by showing various credit cards to eligible customers who log in to its website. The bank now wants to show a retention offer, instead of a credit card offer, to customers who are likely to churn in the near future. The credit card offers are shown only to loyal customers.



To meet this business requirement, a decisioning administrator has already set up the taxonomy by defining a new business issue called Retention, and an offer group.


Taxonomy

Business structure

Business structure

Issues / Groups

Retention

 ExtraMiles

Sales

 CreditCards

This ExtraMiles group contains a retention offer, Extra Miles 5K.

Actions

Search

by name or description

Issue / Group

Retention / ExtraMiles ▼

Showing 1 of 1 results

[Extra miles 5K](#)

ExtraMiles5K

The next step is to create an applicability condition that makes a customer qualify for a retention offer when there is a high likelihood that the customer might churn. A data scientist has created a prediction that identifies these high-risk customers. When you open the prediction in Prediction Studio, notice that the possible response labels are **Churn** and

Loyal to predict customer behavior. The result of the prediction is stored in the pxSegment property.

Response labels

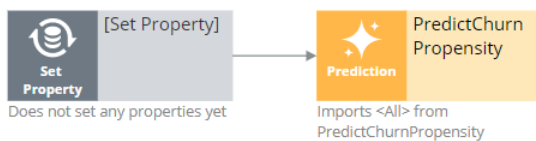
Labels for the possible values of the responses.

Churn

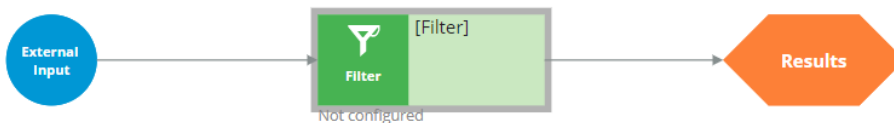
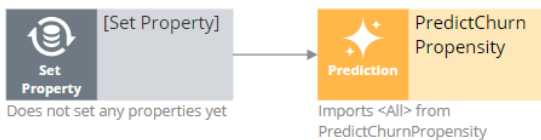
Target label Alternative label

Churn Loyal

To define the applicability condition, you create a decision strategy to output a retention offer only if the response label of the prediction is **Churn**. Add a **Prediction** component to the canvas and configure it to reference the churn prediction. Add a **Set Property** component and connect it to the Prediction component. You can configure the Set Property component at a later point to accommodate parameterized fields.



Next, add a filter component to filter out the loyal customers and pass retention offers to high churn risk customers only.



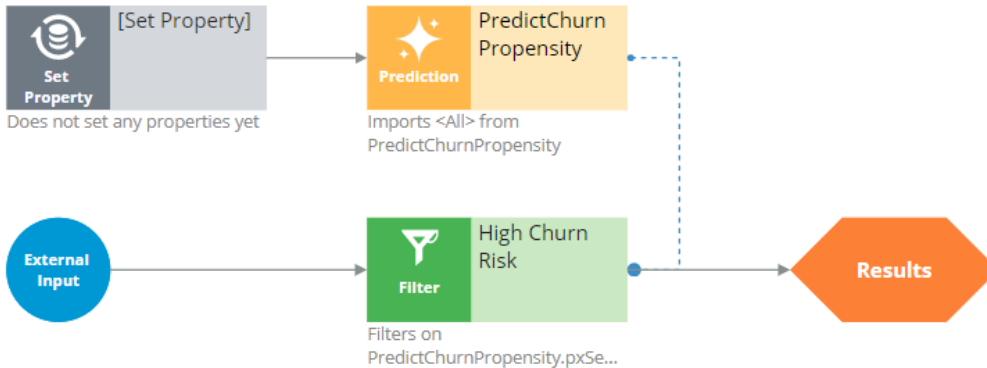
The filter condition is defined to output a retention offer when the pxSegment property of the prediction is equal to **Churn**.

Expression builder

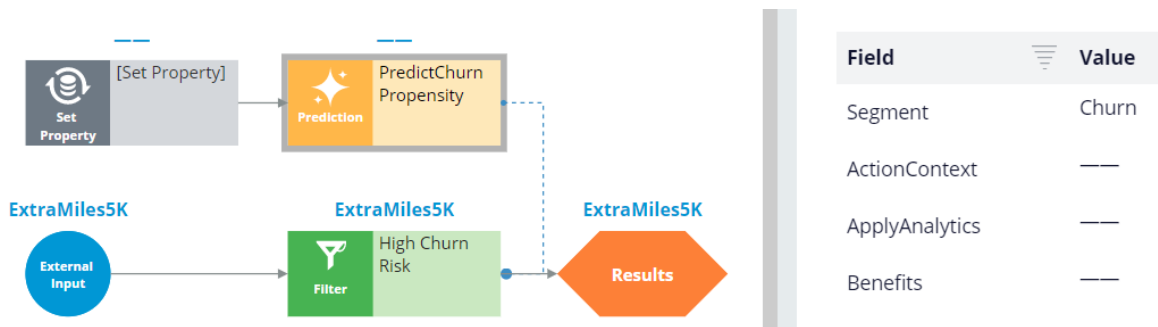
Browse

Test

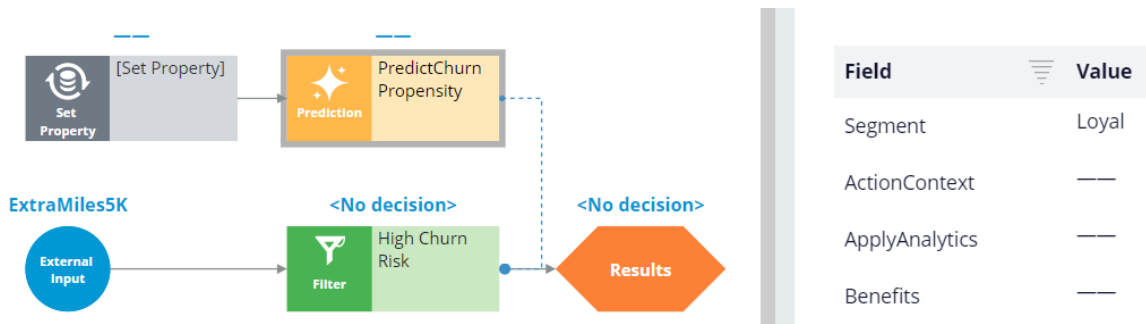
```
1 PredictChurnPropensity.pxSegment="Churn"
```



Next, test the strategy using two customer profiles, **Troy** and **Barbara**. For external inputs, consider all available retention offers. The strategy outputs a result for **Troy** because the result of the prediction is **Churn**.



The strategy does not have a result for **Barbara**, because the Segment value is **Loyal**.



By checking in the strategy, you commit your changes so that they go into effect. You can now use this strategy in the Next-Best-Action Designer engagement policy as an applicability condition.

The first business rule you need to implement is that the **ExtraMiles** group is applicable only to high churn risk customers. To implement this rule, in the **Applicability** section, define a condition for the customer field. Select the **RetentionStrategy**. The condition is: the RetentionStrategy has results for the High Churn Risk component.

The second business rule you need to implement is: U+ Bank wants to show credit card offers to low-risk customers only; meaning the **CreditCards** group is not applicable for high-risk customers. To implement this rule, modify the **Applicability** section of the **CreditCards** group. The condition is: the RetentionStrategy doesn't have results for the High Churn Risk component.

Once the applicability conditions are defined, you need to amend the **Channels** configuration. Because U+ Bank introduced a new group, **ExtraMiles**, which belongs to a new business issue, **Retention**, you need to select the results from the appropriate business structure level. In this case, the bank wants to arbitrate between two different business issues: Sales and Retention. Therefore, select All Issues/All Groups from the business structure level. Saving the configuration implements the business requirement.

On the U+ Bank website, when you log in as **Troy**, notice that the retention offer is displayed because **Troy** is predicted to churn in the near future.

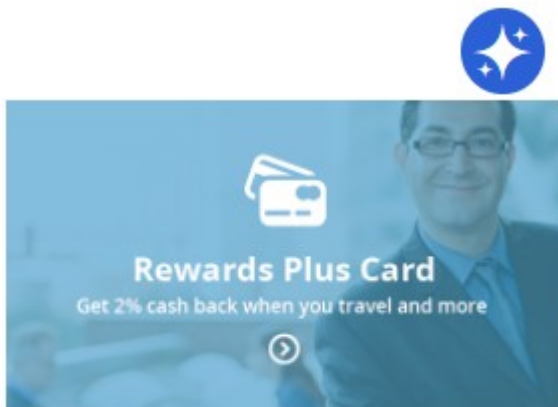


Extra miles 5K

5,000 extra miles

[Learn more](#)

Now, when you log in as **Barbara**, notice that the credit card offer is displayed because she is predicted to remain loyal for now.



Rewards Plus card

Get 2% cash back when you travel and more

[Learn more](#)

You have reached the end of this demo. What did it show you?

- How to use a prediction in a decision strategy
- How to arbitrate between different groups of actions to display more relevant offers to customers
- How to define applicability rules using a decision strategy in Next-Best-Action Designer

Creating parameterized predictors

Introduction

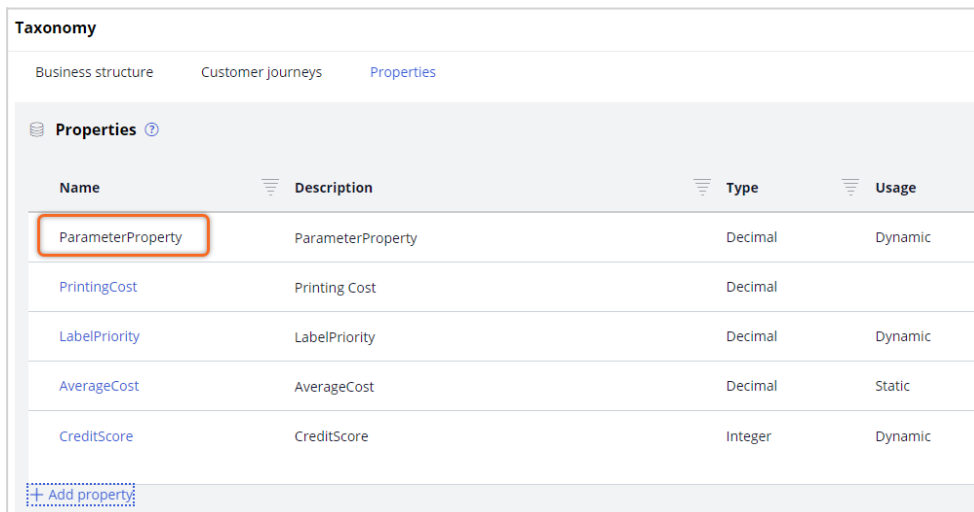
Input fields that are not directly available in the customer data model can be made accessible to the models by configuring these fields as parameterized predictors.

This topic covers the creation of parameterized predictors for expressions, offline model scores, and online model scores.

In all three cases, the Data Scientist creates an SR property to store the parameter value, adds the property as a parameterized predictor to the prediction, and configures the pre-processing extension strategy.

Property to store a parameter

When you create a new property in NBA Designer, the system automatically maps it to the SR class.



Name	Description	Type	Usage
ParameterProperty	ParameterProperty	Decimal	Dynamic
PrintingCost	Printing Cost	Decimal	
LabelPriority	LabelPriority	Decimal	Dynamic
AverageCost	AverageCost	Decimal	Static
CreditScore	CreditScore	Integer	Dynamic

In Prediction Studio, you make sure that the results of the prediction are saved to the SR class.

Prediction properties ✕

Prediction name

Outcome

Prediction output
 Predicting Propensity to Click for each combination of

	<input type="text" value=".pyIssue"/>	
and	<input type="text" value=".pyGroup"/>	
and	<input type="text" value=".pyName"/>	
and	<input type="text" value=".pyDirection"/>	
and	<input type="text" value=".pyChannel"/>	
and	<input type="text" value=".pyTreatment"/>	

[+ Add](#)

Save results to

CDH-SR

Data-pxStrategyResult

Mapping of the parameter

You then map a new parameterized predictor to the SR property you created in the prediction.

Edit Parameters ✕

Parameterized predictors (5) **Pages (1)**

Map fields to parameterized predictors to supply this data to this model. < 1 2

Predictor	Data type	Predictor type	Field
<input type="text" value="NewParameter"/>	<input type="text" value="Decimal"/> ▼	<input type="text" value="numeric"/> ▼	<input type="text" value=".ParameterProperty"/>

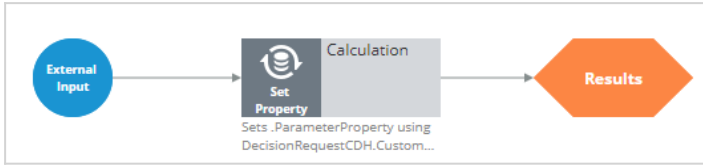
[+ Add parameterized predictor](#)

Configuring the pre-processing extension strategy

To populate the parameter, you configure the pre-processing extension strategy. This strategy runs before the prediction strategy so that the output of the extension strategy is available as input to the prediction.

The first example of a parameterized predictor is an expression.

You add a Set Property component to set the parameter propensity equal to the ratio of customer visits to a web page in the last 30 days and in the last 90 days.



A high value may indicate the increasing interest of the customer in the content of this page.

Set property properties

Name *

Component ID Calculation

Description Use generated Use custom

Source components **Target**

Define action, target, and source

+ Add item X Delete

Action	Target	Source
Se	.ParameterProperty	equal to @if(DecisionRequestCDH.Cl

Expression builder Browse Test

```
1 @if(DecisionRequestCDH.Customer.FSClickstream.I
nvestmentPageVisitsLast90Days=0,0, divide(Decis
ionRequestCDH.Customer.FSClickstream.Investment
PageVisitsLast30Days, DecisionRequestCDH.Custome
r.FSClickstream.InvestmentPageVisitsLast90Day
s))
```

The second example of a parameterized predictor is offline model scores. The data scientist team may produce many product group scores for each customer and each channel.

To make these scores available to the adaptive models as candidate predictors, a Decisioning Architect creates a database table to accommodate the data.

To make the model scores available to the adaptive models, you add a Data Join component to the extension strategy.



The component joins the Offlinescores page to the available data.

Data join properties

Source components **Join** Properties mapping

Join source components with

Type: Pages

Name*: Offlinescores

Class: CDH-SR

To output only relevant scores, you specify the channel and the product category to match the prediction.

Join when all conditions below are met

+ Add item × Delete

	Offline scores		Offlinescores
When	.pyChannel	is equal to	"Web"
and	When		
	.Category	is equal to	.Category

Exclude source component results that do not match join condition.

Configure the component to populate the parameter property.

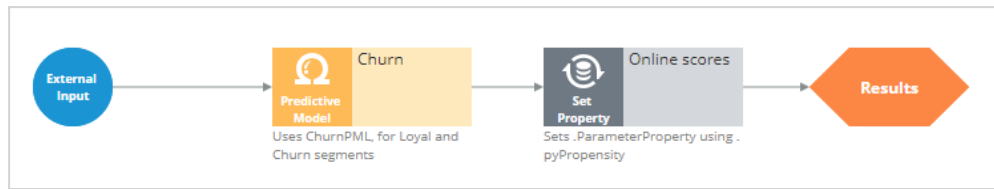
Define mapping

+ Add item × Delete

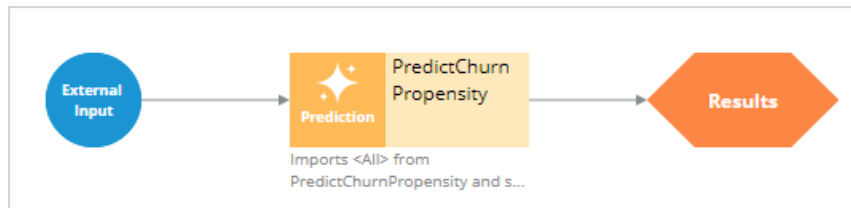
	Target		Source (Offlinescores)
:: Set	.ParameterProperty	equal to	Offlinescores.Score

The third example of a parameterized predictor is the score of a churn model running in Customer Decision Hub. The data scientist team develops churn models based on the latest insights.

In Customer Decision Hub, the models drive a churn prediction. To populate the parameterized predictor, add a Set property component to map the parameter property to the model outcome.



To make the churn score available to the models as a parameterized predictor, add a prediction component that references the churn prediction to the extension strategy.



To summarize, creating parameterized predictors for adaptive models involves:

- Creating an SR property to store the parameter value
- Adding the property as a parameterized predictor to the prediction
- Configuring the pre-processing extension strategy

Defining and creating customer journeys

Description

Customer journeys are the sum of the experiences that your customers go through when interacting with your organization. Rather than looking at just part of an experience or transaction, the customer journey documents the full spectrum of the customer experience. Customer journeys give you a familiar mental model to manage and work with their creative content. You can create customer journeys directly in Next-Best-Action Designer and map customer actions to different journey stages.

Learning objectives

- Define customer journeys in Next-Best-Action Designer.
- Assemble and manage customer journeys.
- Define stage entry criteria for customer journeys.
- Test and monitor customer progress through customer journeys.

Pega Customer Journeys

Customer journeys are a sum of the experiences that your customers go through while interacting with your enterprise. Rather than reviewing only part of an experience or transaction, the customer journey documents the full customer experience.

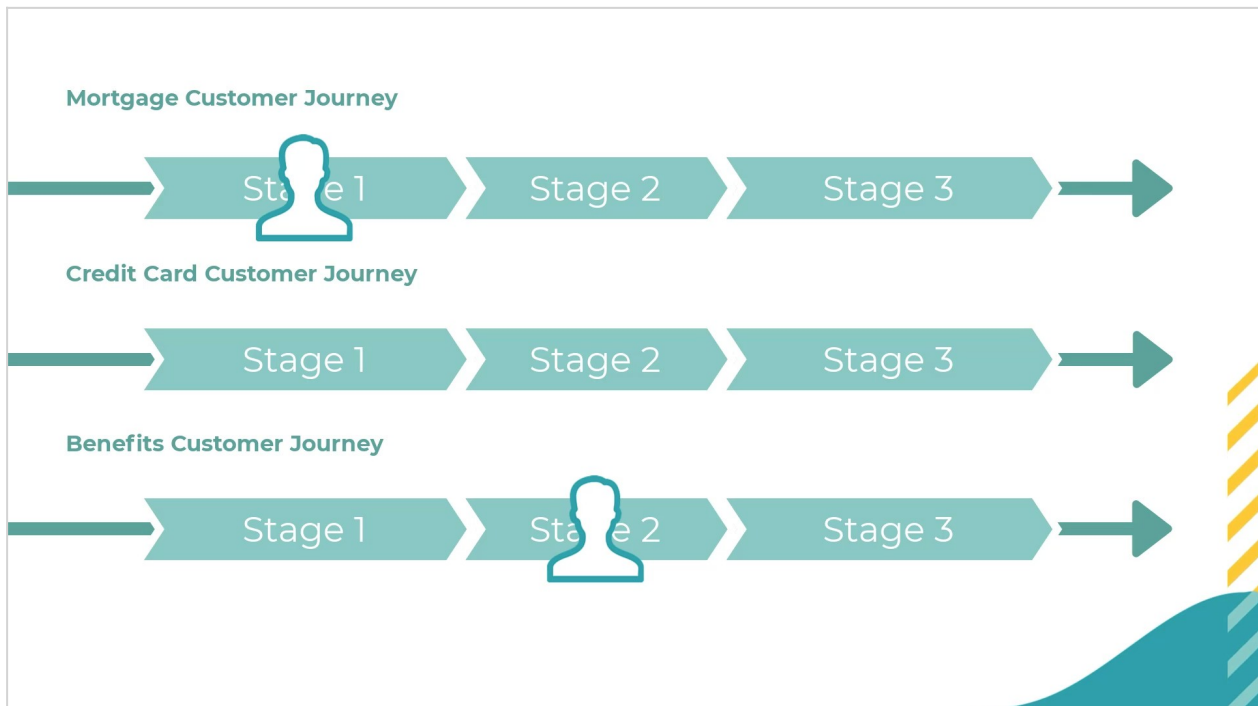
Transcript

Industry standard customer journeys are used to script the customer experience, which often results in a customer getting stuck on a path of offers that does not lead to a relevant outcome. Pega Next-Best-Action Customer journeys take a transformational approach to the industry standard tool that are the customer journeys.

Pega customer journeys and actions in them are prioritized using real-time AI, which provides you the following benefits:

- Allows you to build meaningful customer journeys
- Stays true to the Next-Best-Action paradigm
- Delivers a more relevant customer experience

With Pega customer journeys, customers might find themselves in multiple journeys or switching from one to another depending on what is most relevant to them at that point in time.

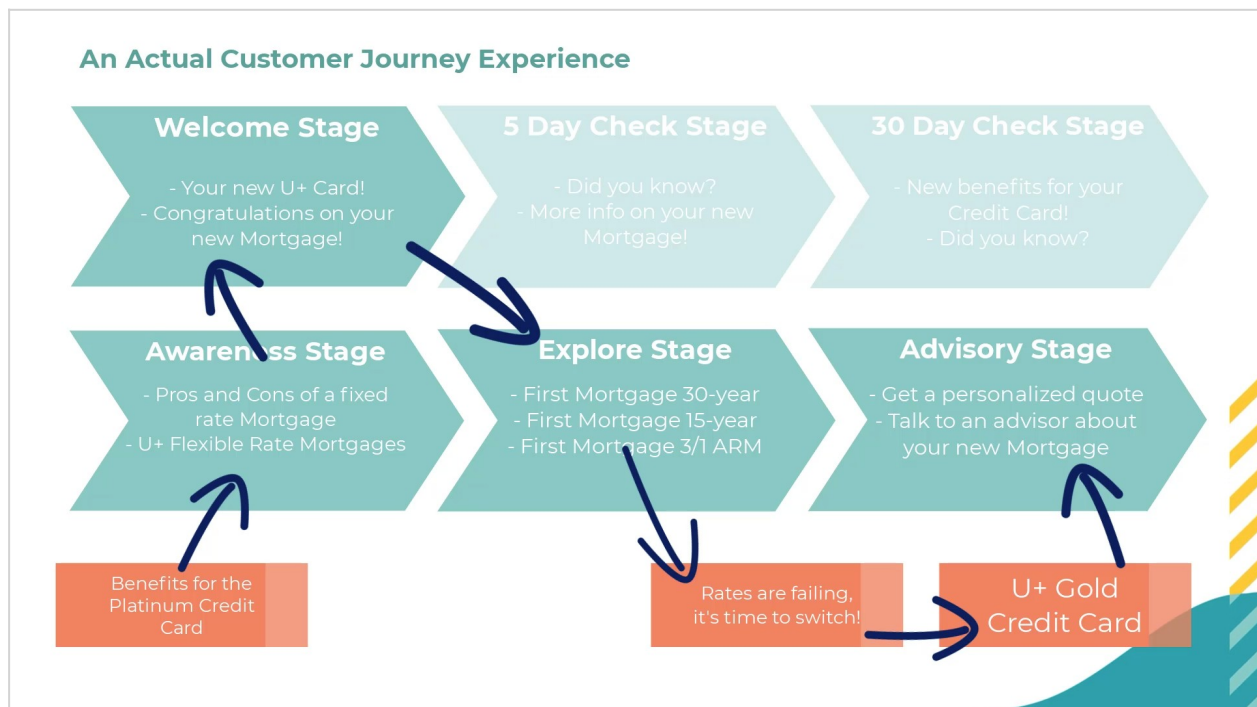


Customer journeys give you a helpful mental model and allow you to influence the customer experience while remaining true to the Next-Best-Action paradigm. Instead of scripting the entire customer experience offer-by-offer, you can add multiple journeys and stages with actions to your system and let AI leverage real-time data to determine which journey or action in a stage is most relevant to the customer at that time.

You can apply customer journeys to multiple business issues, for example:

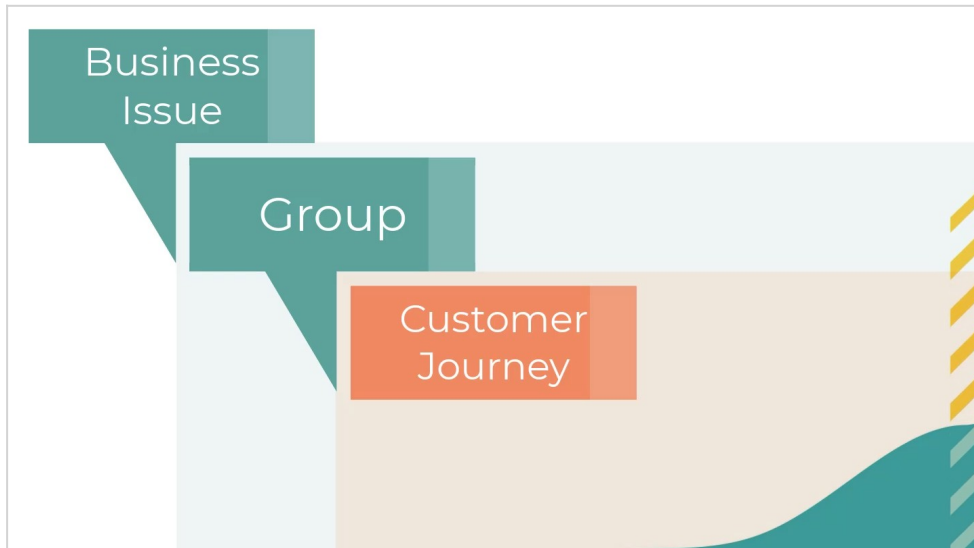
- Periodic check-ins during the onboarding period. You can have an initial welcome stage, followed by a five-day check stage, and then finally, a 30-day check stage.
- Familiarizing and leading the customer to a new mortgage. In this journey, we first raise awareness about mortgages, and then once familiar with the concept, we invite the customer to explore various mortgage offers. Finally, once engaged in the explore phase, we can provide an advisor to help the customer to develop a detailed mortgage offer.

Customers can be a part of many journeys, which are continuously prioritized. Customers can be in one stage of each of given journey, and then progress through the stages sequentially. The system ensures that the customer receives the most relevant action, even if an action is not a part of any journey, which means the actual customer journey might look very different from what you originally had in mind.



You define customer journeys through Next-Best-Action Designer. There, you can define a customer journey and the business issue and group that it will be nested under, which means that this journey inherits the Engagement Policies of that business issue and group.

You can also set your journey to cover all issues and all groups for a broader scope of actions.

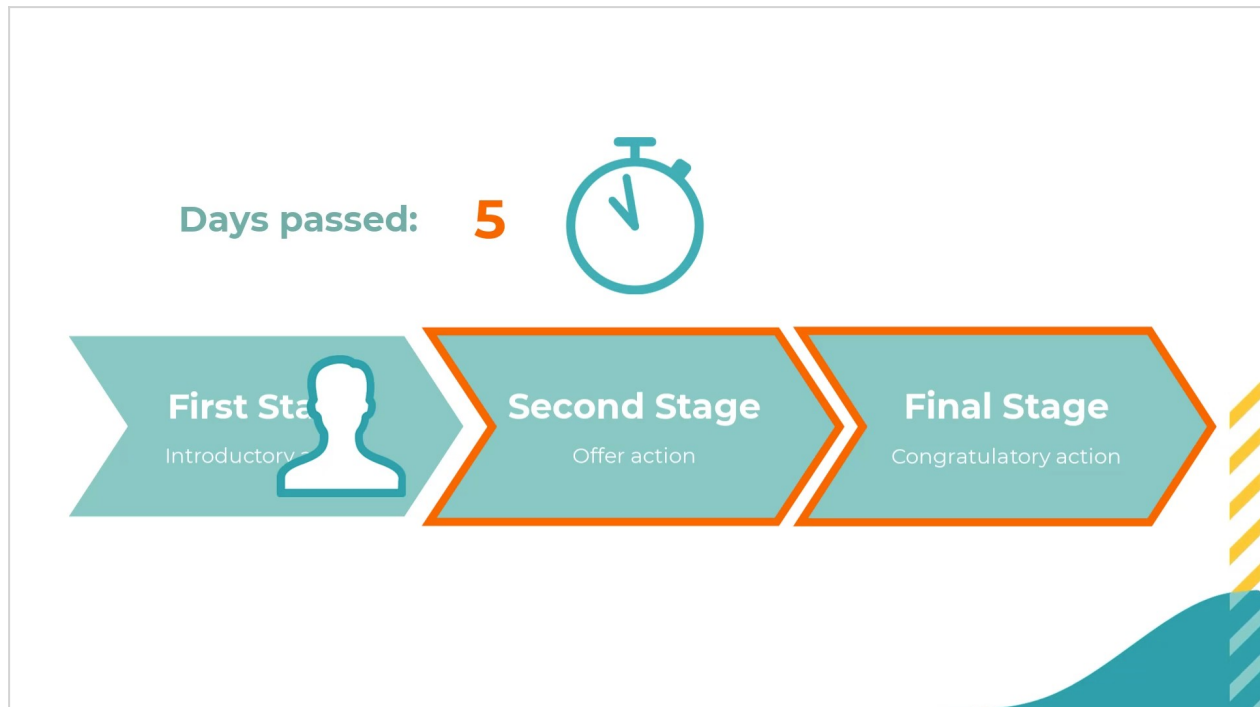


After setting up the journey, you define the number of stages and their names. The customer will progress through these stages in a sequential order. You then add relevant actions to each stage. The actions that you have added come with their available treatments. Customers in each stage see and interact with these actions, and the system recognizes these actions as a part of a particular stage in a customer journey.

If you add an action to a customer journey, you can decide if you present the action as only part of the journey, or both as a part of the journey and as a standalone action. The system records interactions for the action separately.

Before customer progresses from one stage to another, they need to satisfy the entry criteria of the next stage. It is not recommended to use very rigid entry criteria with multiple conditions, as the AI can guide the customer through different journeys and standalone actions.

Entry criteria can be as simple as 30 days passing since the customer has joined the bank and as complex as a customer having engaged with a set of mortgage offers in the last week, showing interest in talking to an advisor.



Take a look at an example of how entry criteria work and where they start to take effect in the prioritization. Say your system has 30 actions, and it is time to present the best three of them to a particular customer. The 12 orange actions are a part of various customer journeys.

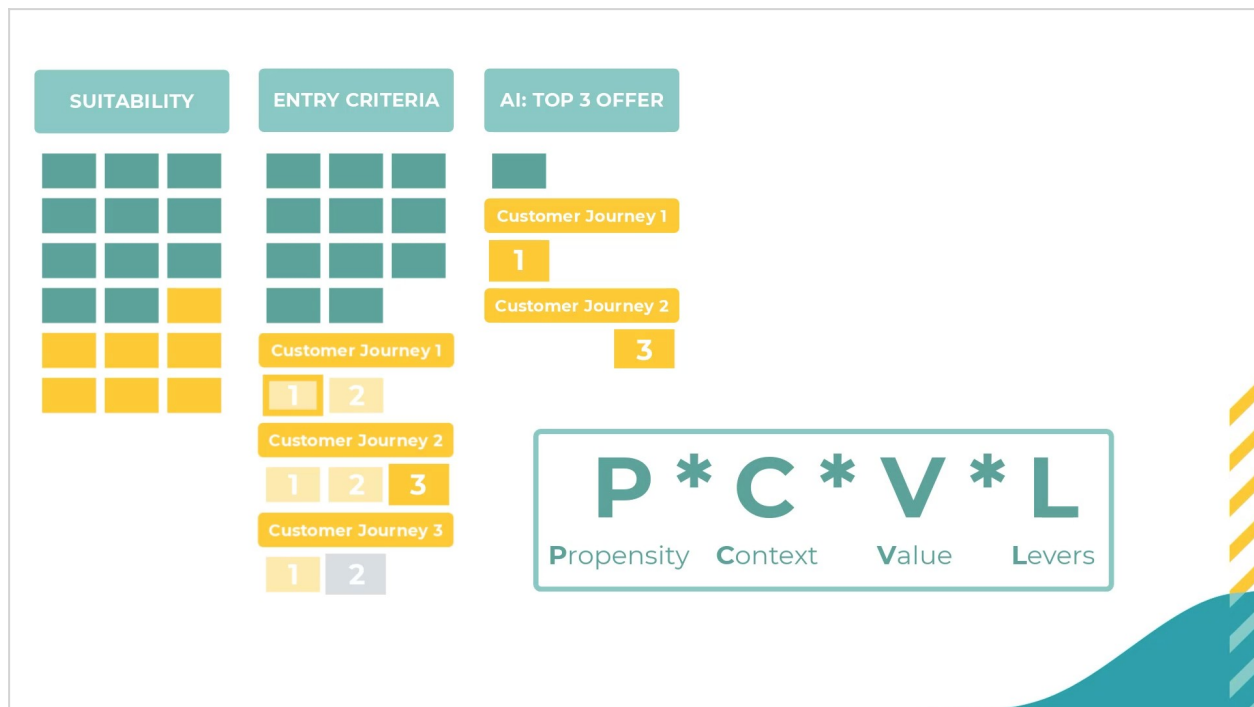
First, the system applies the standard engagement policies. Consequently, the eligibility, applicability, and suitability policies of the business issue, group, and actions are applied, which leaves us with 18 actions, out of which, seven are part of various customer journeys.

Now let's expand to see that the customer journey actions are a part of different journeys, and that each action is a part of a different stage. As you can see, different actions are relevant for the customer in the various stages due to the entry criteria set up by the Decisioning Architect. In this example, the customer:

- Passes the entry criteria to the first stage in the first journey
- Is in the third stage of the second journey, and it passes its entry criteria
- Is in the second stage of the third journey, but it does not pass its entry criteria

From the seven actions, which progressed through the suitability and are part of a journey, the customer actually qualifies for only the two highlighted actions.

At this point, AI selects among the 13 actions the best three actions to show to the customer based on the PCVL calculation (propensity * context * value * levers).



In summary, each component of the policy filters the initial set of 30 actions:

- 24 eligible
- 21 applicable
- 18 suitable

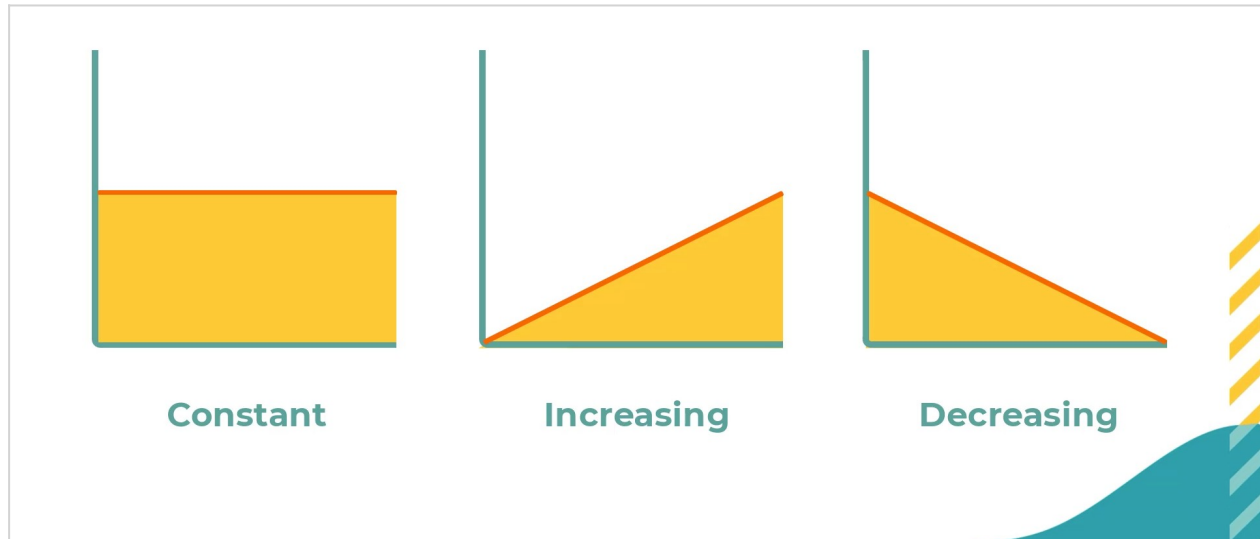
From the 18 suitable actions, 11 are standalone, not part of any journey and two actions from the remaining seven pass the entry criteria from the various journeys and stages defined. The remaining 13 actions, which pass the conditions are prioritized by the AI, and then, the best three are selected. The selected actions can be one of the standalone actions, an action from a journey, or an action from a journey that is already in progress.

Customer journeys can influence the PCVL formula in two ways.

The first way to influence the propensity occurs passively as you add customer journeys to your system. Customer journeys come with predictors that begin to function as your customers interact with the journeys that you have designed for them. These predictors will influence the propensity directly.

The second way is to use stage upweighting. Stage upweighting applies a numerical upweight value to actions in that stage and is used to increase or decrease actions promotion level. You can apply it to any stage, and it is a part of the Business levers in the Arbitration.

Upweighting can be constant, increasing or decreasing over time. For example, you might want to upweight actions of a stage for the first 10 days that the customer is in this new stage. During the first 10 days, the upweight will decrease steadily to avoid annoying the customer with a single offer.



In summary, customer journeys provide a good mental model for managing content and influencing next-best-action decisions to present suitable actions seamlessly for your customers.

Defining Customer Journeys

Define and maintain customer journeys through the Next-Best-Action Designer to let the NBA Designers shape and monitor the experience that customers have with an entire process, not just one action or treatment. For example, the customer journey can guide a customer to first gain awareness of a product, see the offers themselves, and then finally speak to a representative. This sequential offer structure creates a streamlined and more thoughtful approach to presenting a product. Other examples of such processes include applying for a new credit card, refinancing a loan, or onboarding a new customer.

Transcript

This video demonstrates how to define and assemble a fully functioning customer journey in Customer Decision Hub.

U+ Bank wants to use customer journeys for its mortgage offering. The bank plans to first make customers aware of applying for a mortgage and its benefits. After the customer shows interest, U+ Bank wants to enable customers to explore some of the mortgage offers directly. Finally, U+ Bank wants to connect customers to an advisor to make a personalized offer and to ensure that customers make an informed decision on their mortgage. By using customer journeys, U+ Bank puts actions and treatments into stages in a sequential format that guides customers toward mutually beneficial outcomes.



To implement this use case, you first define the customer journey on the **Taxonomy** tab of **Next-Best-Action Designer**.

Add customer journey ✕

Name * Customer journey identifier

Discover and Apply for a Mortgage DiscoverAndApplyForAMortgage

Examples: Discover & Apply for the Platinum Card; Learn about Family Plans

Description

Issue / Group ?

Grow / Mortgages

Availability

Always


Never

Within a defined time period

Cancel
Submit


On the **Taxonomy** tab, you name the customer journey, and then specify the business issue and group to which your customer journey belongs. This designation is important because customer journeys inherit the engagement policies of the business issue and group that they occupy. In the following example, the actions that are assigned to this journey inherit the engagement policies of the **Grow** business issue and **Mortgages** group.

Next-Best-Action Designer




Taxonomy

Define your Next-Best-Action business structures and customer journeys




Constraints

Set outbound channel limits and suppression policies



Engagement policy

Capture business rules which define when actions are appropriate



Blurred

Taxonomy

Business structure Customer journeys Properties

+ **Customer journeys** ?

v **Grow / Mortgages**

Discover and Apply for a Mortgage ACTIVE

Awareness

Explore products

Talk to advisor

After you create your journey, you can add stages to it. At this point, you define only the names of the stages and the sequence in which the customer progresses through them. In this example, you create stages that represent the three engagement stages with the customer in the process of getting a new mortgage:

- **Awareness**
- **Explore products**
- **Talk to advisor**

Business structure	Issue	Group	Customer journey
	Grow	Mortgages	Discover and Apply for a Mortgage ACTIVE
<ul style="list-style-type: none"> ▼ All issues ☰ All groups ▼ Acquire <ul style="list-style-type: none"> ☰ Credit cards ☰ Auto loans ☰ Deposit accounts ☰ Mortgages ▼ Grow <ul style="list-style-type: none"> ☰ Credit cards ☰ Deposit accounts ☰ Mortgages 	<div style="display: flex; justify-content: space-between;"> <div style="width: 30%;"> <p>Awareness ⋮</p> <ul style="list-style-type: none"> > Pros and Cons tile </div> <div style="width: 30%;"> <p>Explore products ⋮</p> <ul style="list-style-type: none"> > Thirty-year mortgage tile > Home equity LOC tile > Fifteen-year mortgage tile </div> <div style="width: 30%;"> <p>Talk to advisor ⋮</p> <ul style="list-style-type: none"> > Talk To An Advisor tile </div> </div>		

After you define the journey and the stages, you then add actions to the corresponding stages. When you add actions to each stage, the application adds all their available treatments. The order in which you add actions to the stage is irrelevant. The AI uses the propensity of the customer to decide which action is the best to display in the stage or out of the stage. The customer journeys do not interfere with the next-best-action paradigm.

Edit Action: Pros and Cons tile [Available]
 Grow • Mortgages • ProsAndConsTile CDH-Artifacts:01-01-01

Details Engagement policy Treatments Flow Test History

Action context [?] Intended recipient [?]
 Customer Eligible contact

Customer journey stage [?]
 Discover and Apply for a Mortgage: Awareness This action is only applicable when in a customer journey stage [?]

+ Add customer journey stage

Eligibility [?]

Inherited from Mortgages Apply [?]

All actions
 (Customer: Relationship length in days is greater than 1
 and Customer: Credit score is greater than 600
 and Customer: Is in collections is false)

U+ Bank wants to ensure that the application displays a given action to a customer only as part of a journey. Configure this in the action itself to prevent it from displaying as a standalone action.

Next, you set the entry criteria. Entry criteria determine which conditions customers must satisfy to move to the next stage. Each stage can have its own criteria that check for various business conditions to match. For example, criteria can check the following conditions:

- The maturity of the customer relationship with U+ Bank.
- The level of interest that a customer shows in a specific category of products.
- Whether it is appropriate to connect customers with an advisor.

To implement entry criteria, use simple conditions that apply properties or use furthermore complex conditions that apply decision strategies.

For the **Discover and Apply for a Mortgage** journey, U+ Bank decides to use the following conditions:

The **Awareness** stage is relevant for customers that currently are not part of this journey.

The **Explore products** stage is relevant only if customers engage in the **Awareness** stage.

The **Talk to advisor** stage is relevant only if customers engage with an offer in the **Explore products** stage.

Edit entry criteria: Awareness

Current stage details

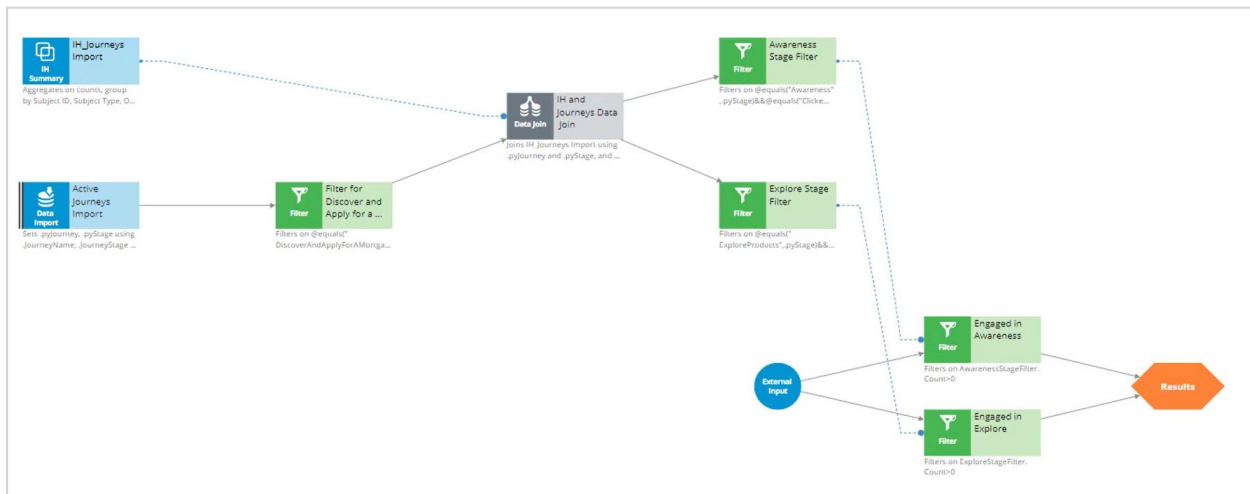
Journey ID	Stage ID
DiscoverAndApplyForAMortgage	Awareness

Customer is in this stage when: ?

+ Add criteria

Cancel Submit

Start with the first stage, **Awareness**. U+ Bank wants to configure the **Awareness** stage to ensure that customers qualify because the goal of this stage is to bring attention to the mortgage offer. As a result, you leave the entry criteria for this stage blank. Customers who qualify for actions from the **Mortgages** group are in this stage of the journey.



Next, set the entry criteria for the two remaining stages by using the decision strategy. The strategy that you use here is customized for this journey and is not a part of the out-of-the-box experience.

The entry criteria expression uses two filter components:

- **Engaged in Awareness**
- **Engaged in Explore**

Therefore, they have to be connected to the Results component.

The **Engaged in Awareness** filter component outputs a record if the customer clicks at least one action from the **Awareness** stage. To implement it, use an interaction history summary component, which imports journeys and stages that a customer has ever initiated. This data is joined with the current journey data to find out the current active stages of this journey. Next, you find out if any click events are registered in the awareness stage. Finally, the U+ Bank definition of engaged in awareness: if at least one or more click events are detected on any action from the awareness stage. The Engaged in explore condition follows a similar pattern. The strategy can implement more complex conditions, but for this scenario, U+ Bank uses this strategy to define its customer engagement in a stage.

Edit entry criteria: Explore products [X]

Current stage details

Journey ID: DiscoverAndApplyForAMortgage Stage ID: ExploreProducts

Customer is in this stage when: [?]

Group ANDs [v] [⋮]

Customer [v] Entry Criteria Discov... [v] has results for [v] Engaged in Awareness [v] [↗] [+] [🗑️]

[Cancel] [Submit]

With this decision strategy, in the **Explore products** stage, you set the condition to check if the strategy has results for **Engaged in Awareness**. The condition determines whether the customer engaged positively with one of the offers in the **Awareness** stage, and the strategy returns a result for the **Engaged in Awareness** filter.

Edit entry criteria: Talk to advisor [X]

Current stage details

Journey ID: DiscoverAndApplyForAMortgage Stage ID: Complete

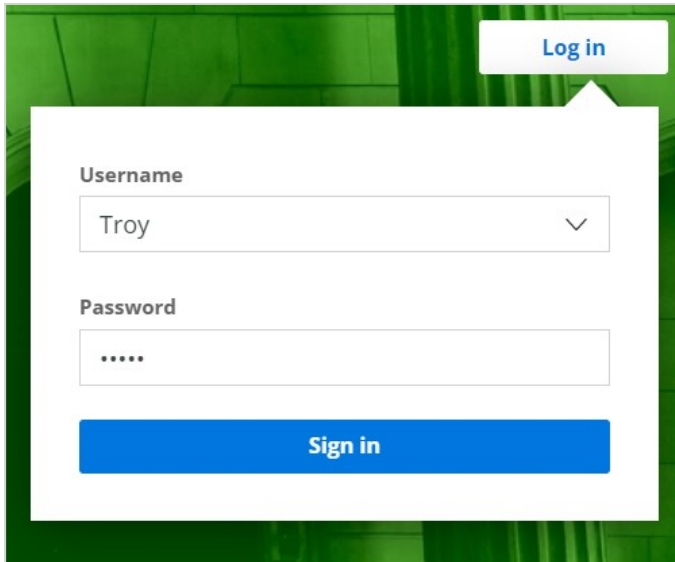
Customer is in this stage when: [?]

Group ANDs [v] [⋮]

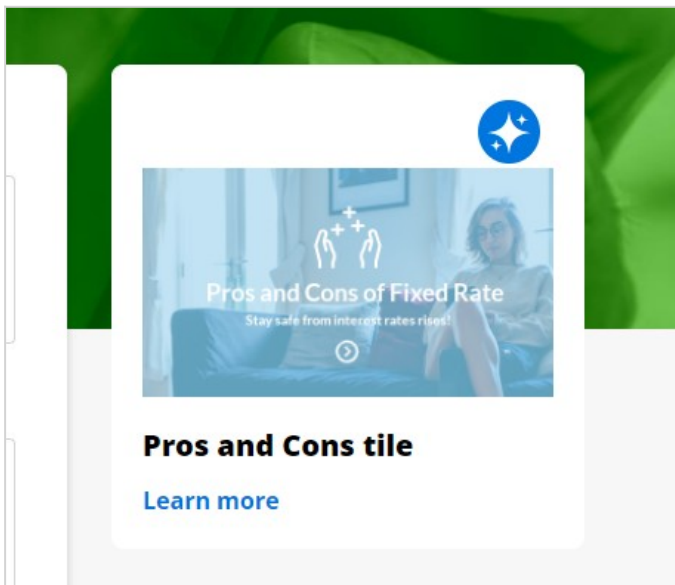
Customer [v] Entry Criteria Discov... [v] has results for [v] Engaged in Explore [v] [↗] [+] [🗑️]

[Cancel] [Submit]

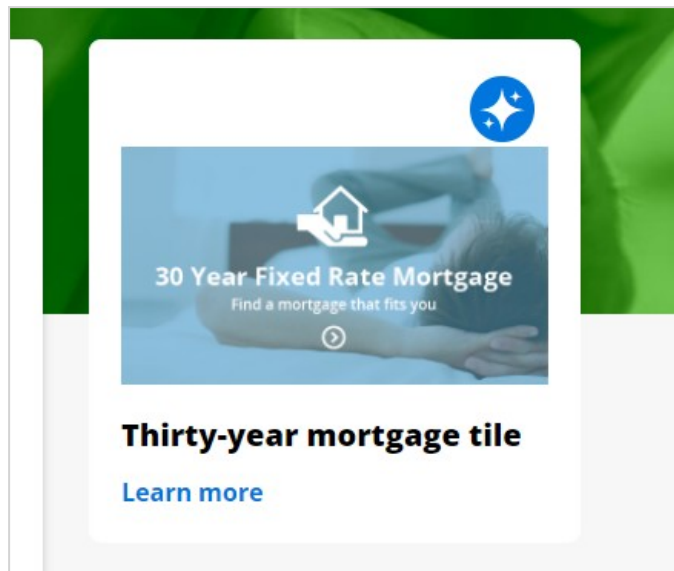
Next, you specify an analogous condition for the **Talk to advisor** stage to check whether the customer engages positively with one of the offers in the **Explore products** stage. After you set up these entry criteria and save them, you can test the customer journey.



Test the first customer journey for U+ Bank on the bank website. You can follow the progress of the U+ customers in the Customer Profile Viewer, as shown:



Follow Troy as he first logs in to the U+ Bank website and begins to engage with the Discover and Apply for a Mortgage customer journey. Troy sees the first web tile for one of the **Awareness** stage actions. This triggers the recording of the impression. As you can see in the **Customer journey** section of the Customer Profile Viewer, Troy is now in the **Awareness** stage.




Troy clicks the banner, meaning he is engaging the awareness stage. This action makes Troy eligible for the next stage of the journey. As a result, the next time he logs in, he can see a mortgages action, as prioritized by the AI, from the possible actions from the current Explore stage. You can see the results of this interaction in Customer Profile Viewer that shows that Troy is now in the **Explore Products** stage.

Overview	Next best actions	Decision history	Interaction history
Demographics Gender: M Age: 56 Income: —			
Customer journeys ⓘ Discover and Apply for a Mortgage ✓ Awareness > Explore product > Talk to advisor Entered stage: Explore product on 4/12/22 6:13 AM			
Suppressed actions No suppressed actions for this customer			
Recent interactions All outcomes ▾ Last 7 days ▾ Tue Apr 12, 2022			
06:13 AM	📄	ThirtyYearMortgageTile ThirtyYearFixedRateMortgage	IMPRESSION
06:13 AM	📄	ProsAndConsTile ProsAndConsOfFixedRate	CLICKED
06:13 AM	📄	ProsAndConsTile ProsAndConsOfFixedRate	IMPRESSION
06:12 AM	📄	FallbackOffer Treatment	IMPRESSION

Later, Troy logs back into the U+ Bank website and shows interest by clicking the offer. As a result, he advances to the **Talk to advisor** stage and the next time he logs in, he sees the **Talk to Advisor** tile. You can see the progress in Customer Profile Viewer.

Customer Profile Viewer Actions

Type: Customer ID View

 **Troy Murphy** 14
 Email address: customer1@enablement.com
 Lifetime value: 900 (3 green dots)
 Churn risk: Medium (2 orange, 1 grey dot)
 Credit risk: High (3 red dots)

[Overview](#) | [Next best actions](#) | [Decision history](#) | [Interaction history](#)

Demographics

Gender	Age
M	26
Income	—

Customer journeys ?

Discover and Apply for a Mortgage

[✓ Awareness](#) > [Explore products](#) > Talk to advisor
 Entered stage: Explore products on 5/18/22 9:52 AM









Suppressed actions

No suppressed actions for this customer

Recent interactions

All outcomes | Last 7 days

Wed May 18, 2022

- 09:52 AM   ThirtyYearMortga... IMPRESSION
 ThirtyYearFixedR...
- 09:51 AM   ProsAndConsTile CLICKED
 ProsAndConsOffi...
- 09:51 AM   ProsAndConsTile IMPRESSION
 ProsAndConsOffi...
- 09:50 AM   ProsAndConsTile IMPRESSION
 ProsAndConsOffi...

Troy moves through the entire journey and can now make an informed decision on which mortgage he wants to purchase after the advisor contacts him. You see now how the underlying decision strategy works in the background for entry criteria to trigger the customer progress through the **Discover and Apply for a Mortgage** customer journey.

You have reached the end of this video. What did it show you?

- How to define and assemble a customer journey.
- How to set stage entry criteria.
- How to test customer journeys.
- How to monitor customer journey progress in Customer Profile Viewer.