



# Cross-Sell on the Web Extended

## STUDENT GUIDE

© Copyright 2020  
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.pegacom](http://www.pegacom)

**Document Name:** MIS\_24076

**Date:** 04 January 2021

# Business use case: Cross-sell on the web extended

## Introduction

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.

## Video



A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

## Transcript

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

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.

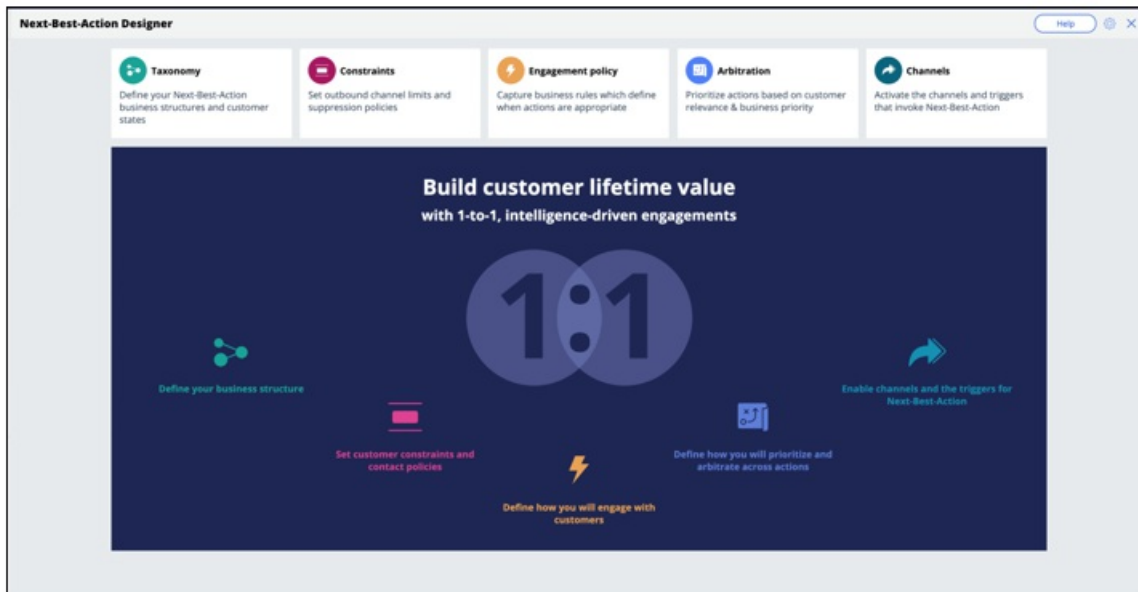
The bank is using the Pega Customer Decision Hub™ to recommend more relevant banner ads to its customers when they visit their personal portal. In the start-up phase, U+ successfully implemented all their use cases using Next-Best-Action Designer.

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.

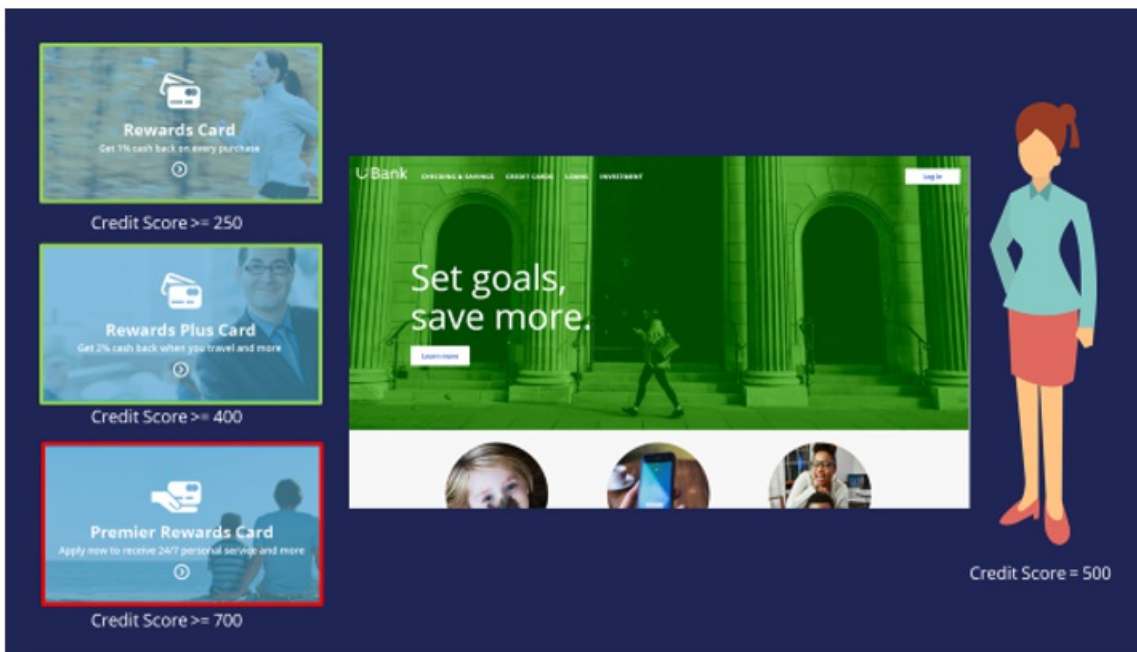


U+ Bank wants to implement additional use cases to meet new business requirements. These use cases require U+ bank to extend Next-Best-Action Designer capabilities.

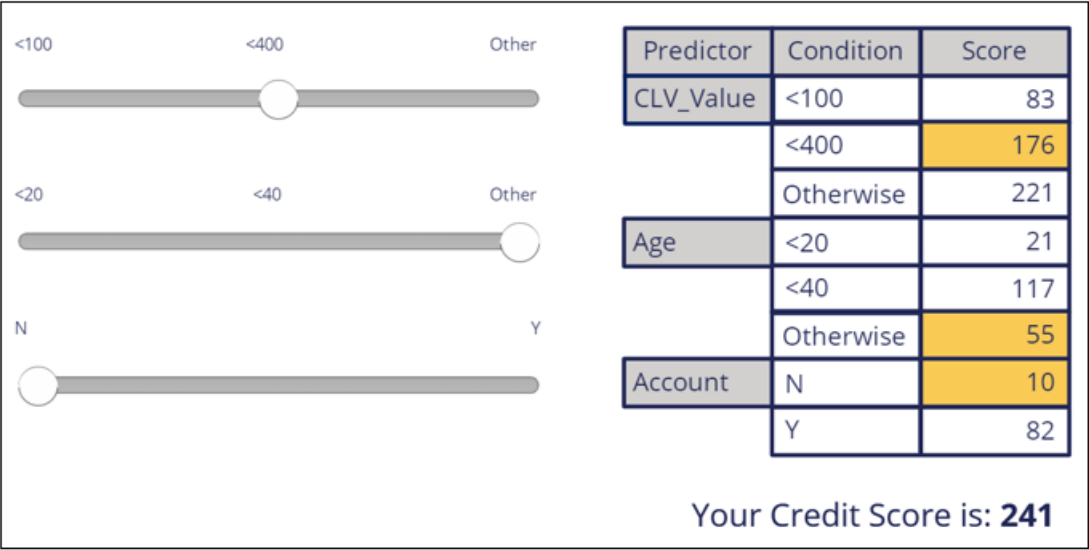


The first use case is to create suitability rules based on a customer's credit score. The bank wants to determine whether or not a customer is suitable for an offer based on their credit score. In this scenario, the credit score is not available, it needs to be computed in real-time based on customer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



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.

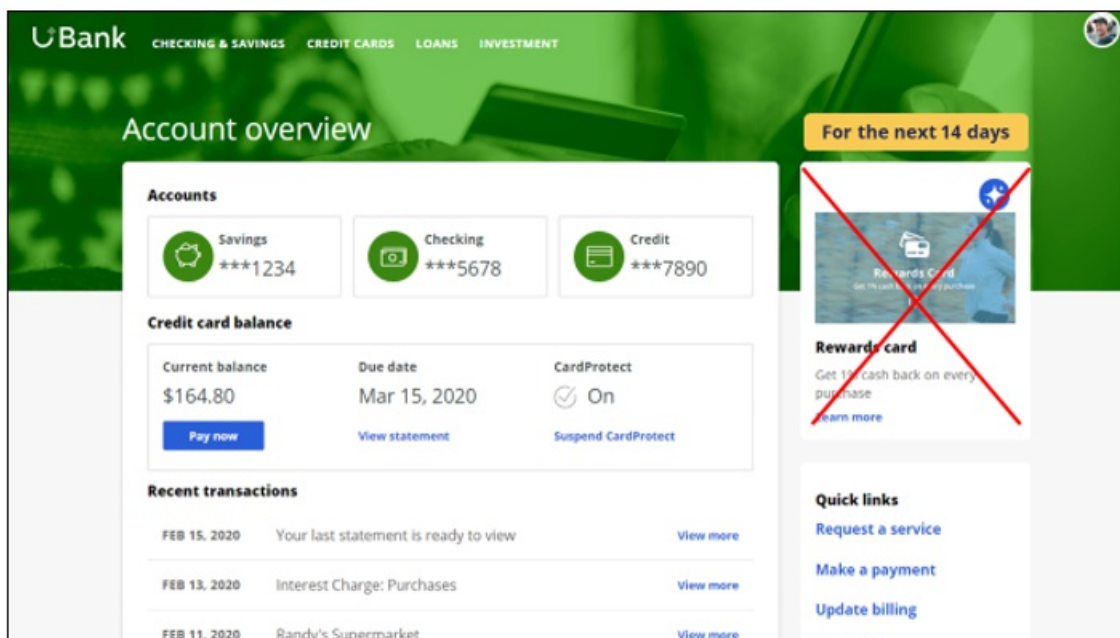
## Risk Segmentation

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.

The next use case is about adding more time periods for tracking customer behavior.

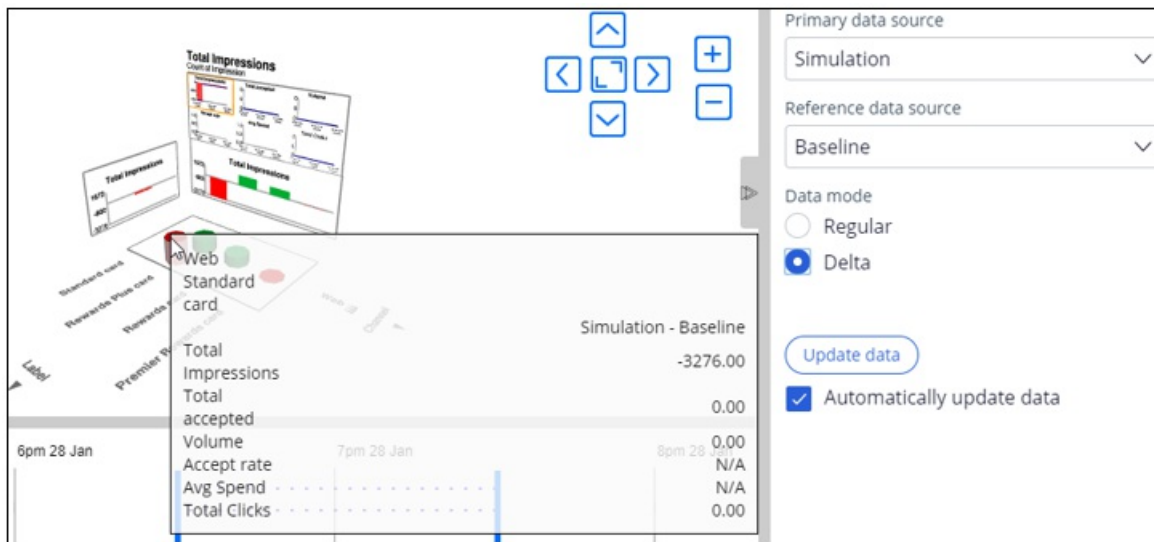
In this use case, the bank does not want to show the same offer to customers who have clicked on it three times in the last 14 days. By default, the Pega Customer Decision Hub tracks customer responses for a period of 7 or 30 days.



This use case requires an additional tracking time period of 14 days.

To add a new tracking time period you have to extend the decision strategies that are used to implement contact policies. This requires creating a new Interaction History Summary rule which tracks customer responses for 14 days.

Finally, while the bank is implementing these various changes, they want to understand what the impact will be on their Next-Best-Actions. Therefore, U+ Bank would like to run some simulations to test the effects of the changes. The results of the simulations can be analyzed using Visual Business Director.



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.

Business use case: Cross-sell on the web extended -- Fri, 07/24/2020 - 06:17  
 To get the full experience of this content, please visit <https://academy.pega.com>

## Next-Best-Action Designer

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

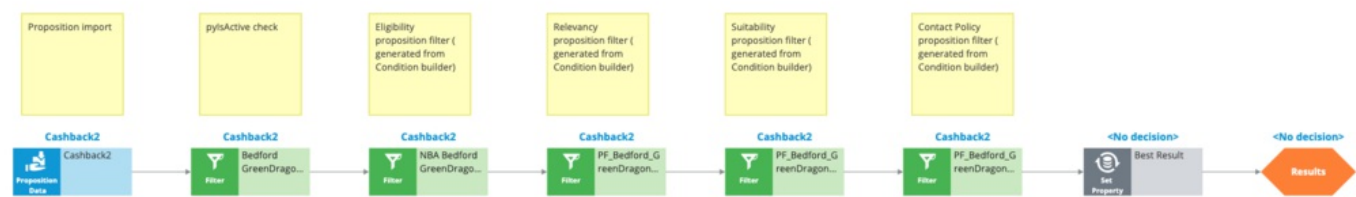




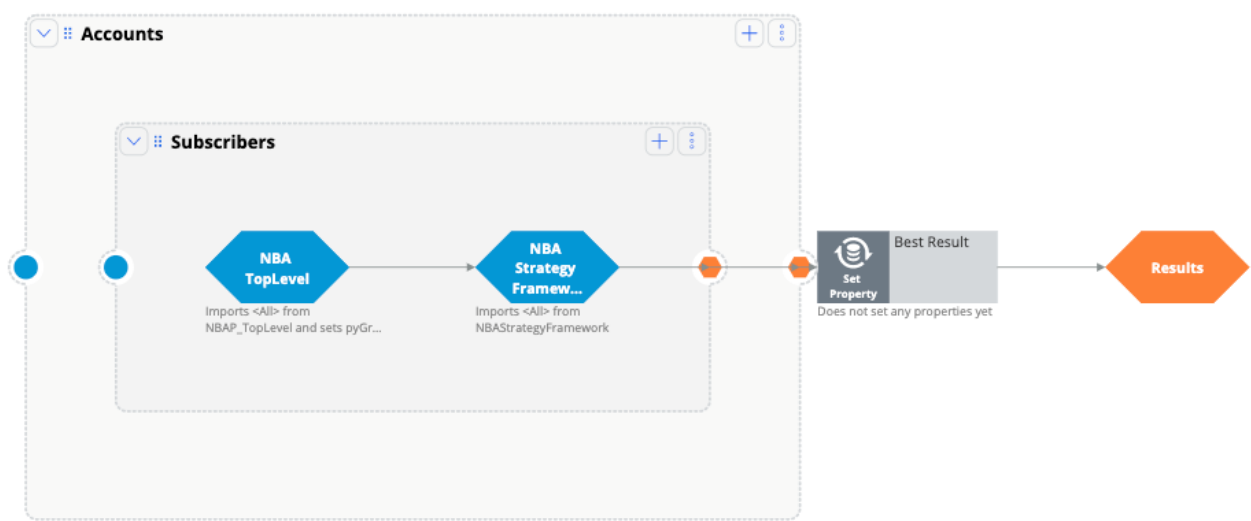
If you want to modify the strategy later, you can do that from Next-Best-Action Designer’s simple and transparent interface.

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, Relevance and Suitability rules.



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

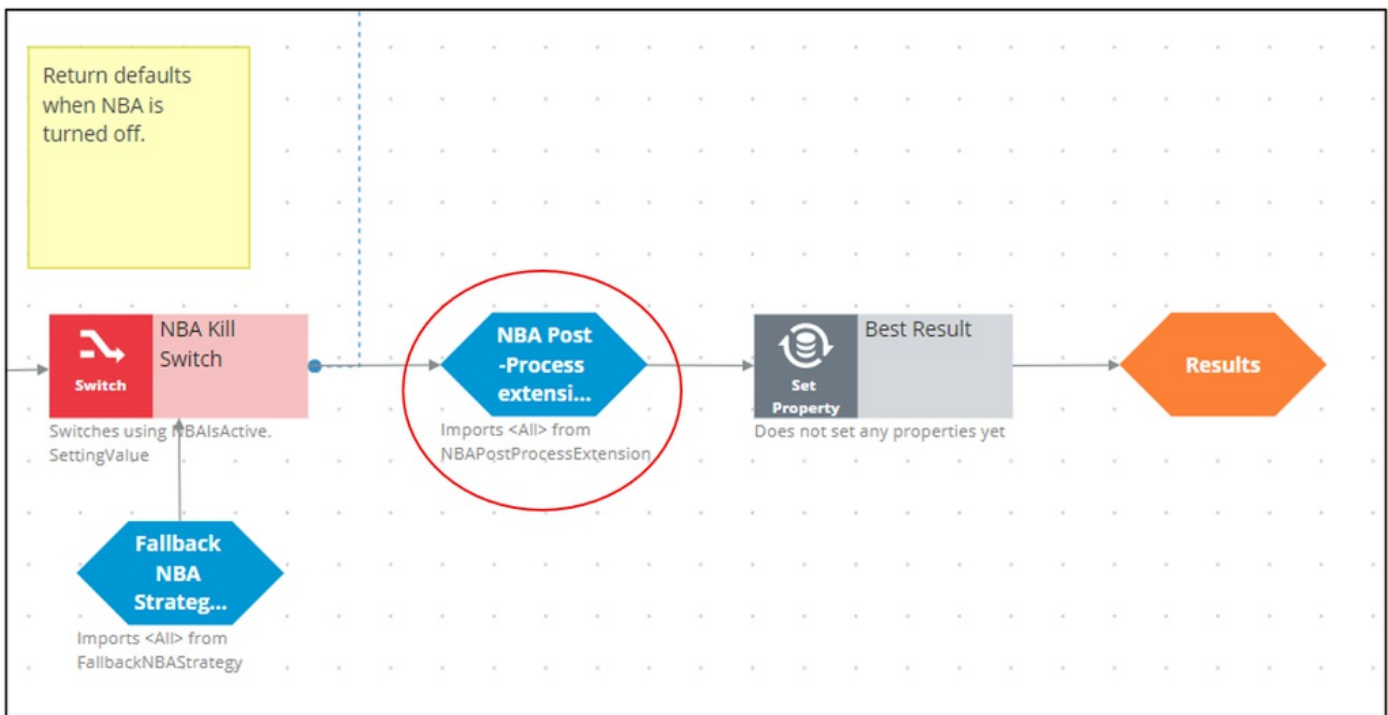


## Strategy framework extension points

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.





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:

## CheckSpecificChannelLimits [Available]

M-Data-Customer ▾

ID CheckSpecificChannelLimits

RS PegaCRM-Artifacts:01-01-01

This record has 1 info warning

[View](#)

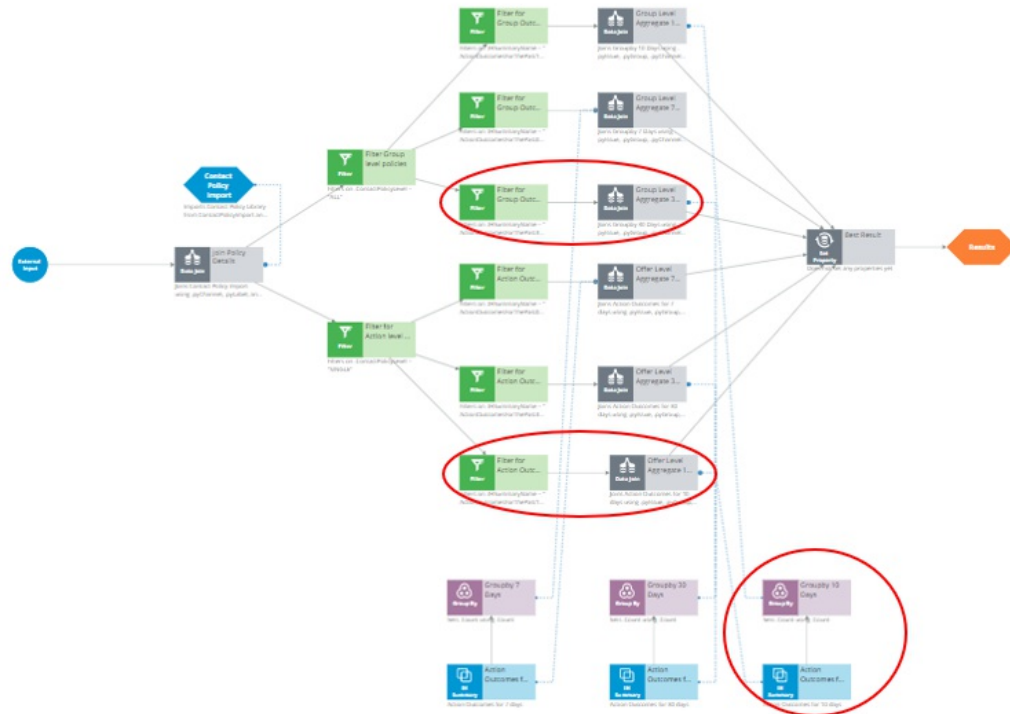
## Strategy Properties

## Test Cases

## Pages & Classes

## Specifications

## History



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.

Decision strategies -- Fri, 07/24/2020 - 06:19

To get the full experience of this content, please visit <https://academy.pega.com>

# Introduction

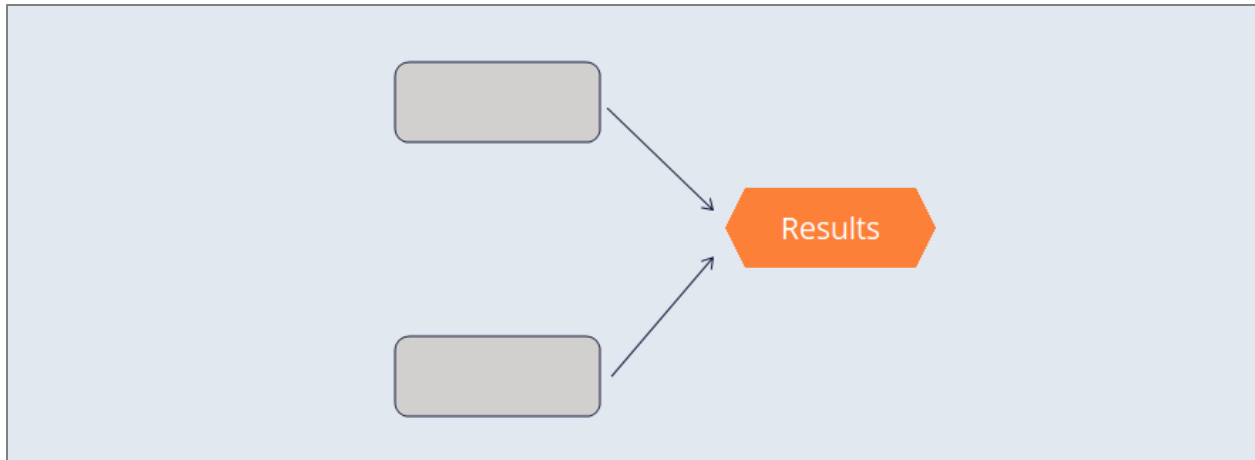
Decision strategies drive the next best action and comprise a unit of reasoning represented by decision components. You use the Proposition Data component to import actions into a strategy canvas. The sequence of the components in the canvas determines which action is selected for a customer.

Click the **Play** button to learn more about decision strategies.



## Screen1: U+ business scenario

U+, a telecom organization, wants to promote two new phones in the contact center: iPhone and Galaxy. Click the + icons to learn more about the elements of a decision strategy that is created for this requirement.



**Decision Components:** A decision strategy is comprised of building blocks called decision components. You can add and connect components to implement the business requirements.

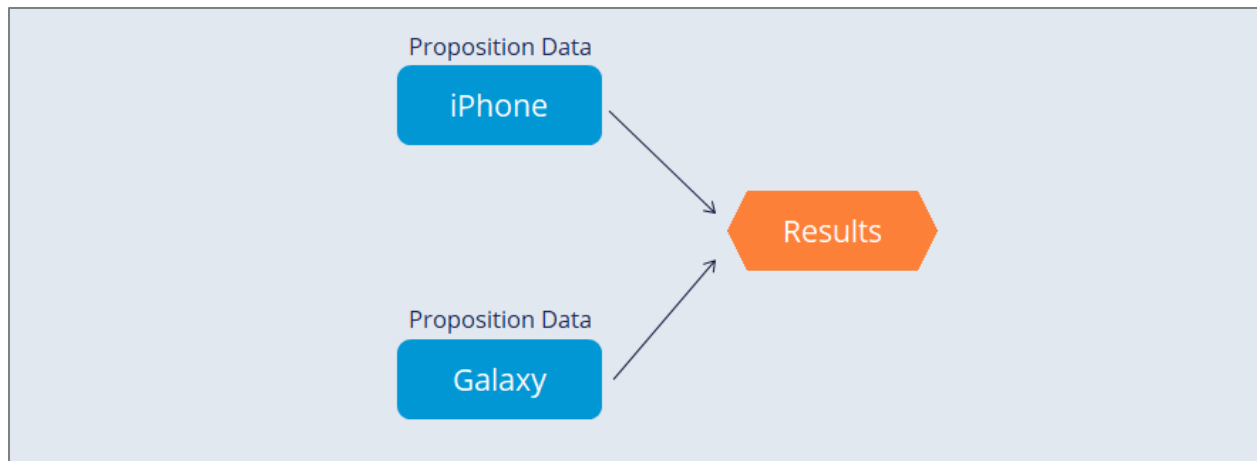
**Arrows:** An important element of the strategy canvas is the arrow. An arrow connects two decision components. A solid line means the data is copied from one component to another.

**Strategy Canvas:** In Pega, business users visually design decision strategies on what is known as a strategy canvas.

### **Screen2: Proposition Data component**

The Proposition Data decision component imports the properties of an action. The result of this component is a flat list of all the properties.

Click the + icons on the proposition components to examine the components' results.



**iPhone:** This Proposition Data component outputs the Price and the Cost properties of the iPhone action.

Name: iPhone

Price: 150

Cost: 100

**Galaxy:** This Proposition Data component outputs the Price and the Cost properties of the Galaxy action.

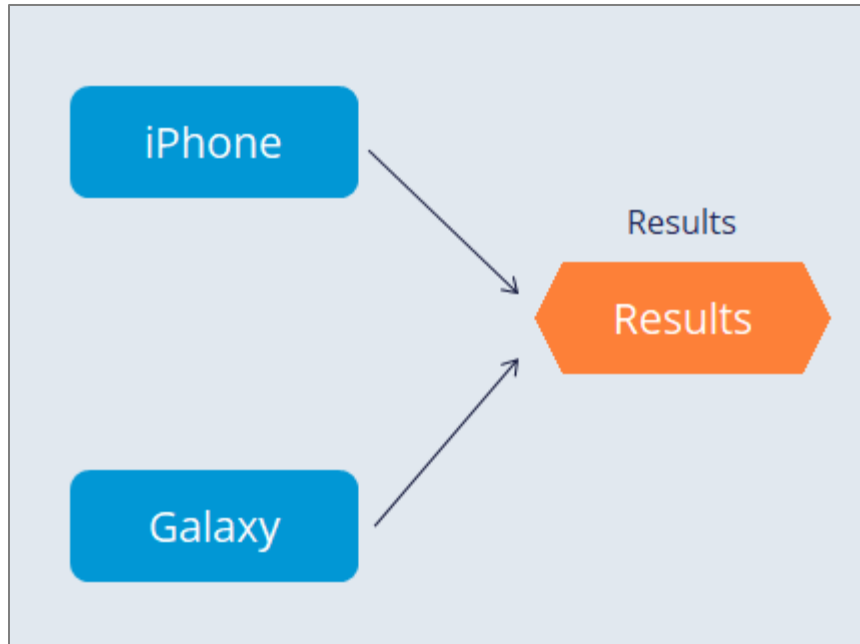
Name: Galaxy

Price: 250

Cost: 150

### **Screen3: Results component**

Another decision component is the Results component. Each strategy always contains one Results component, which defines the output of the decision strategy.



How many actions do you think this strategy outputs?

- A. 0
- B. 1
- C. 2

**Feedback:** Because both iPhone and Galaxy are connected to the Results component with a solid arrow line, this decision strategy outputs two actions.

## Dynamic pricing

U+ Bank wants a dynamic Price for all offered actions. If the Customer value of a customer is higher than 60, the bank wants to offer a 10% discount to the customer.

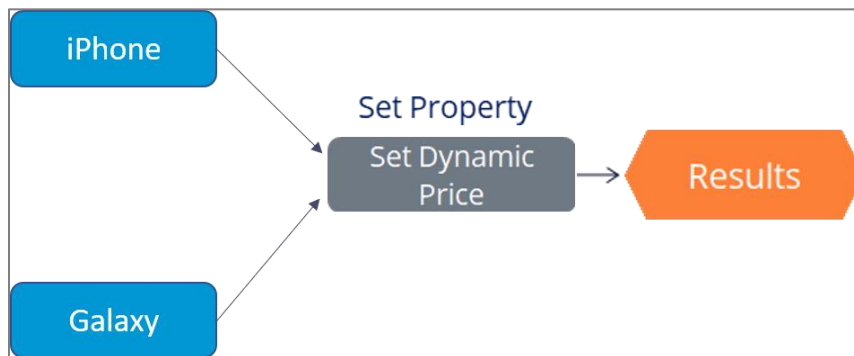
To meet the new requirement, you must enhance the existing strategy to set the value of the Price based on Customer value. Changing the Price dynamically based on the Customer value makes the pricing customer-centric.

### Set Property component



The Set Property component is used to dynamically alter the value of an action property based on a customer property. You use this component to set values to properties that are output by the strategy.

You can set properties to a **constant** or **calculated** value.



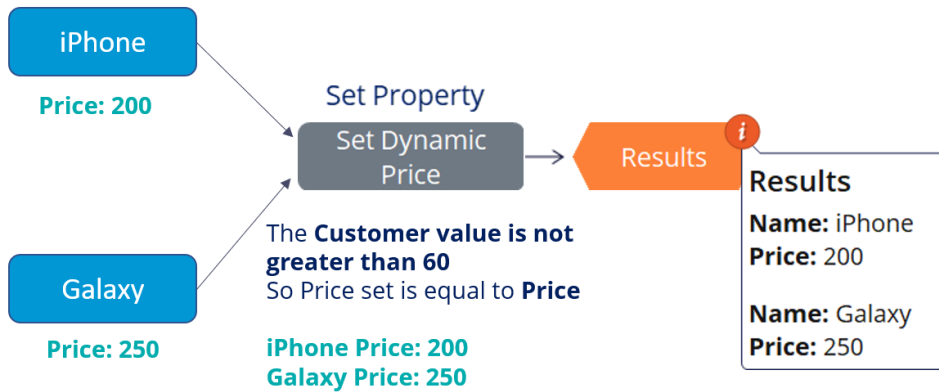
### Example

Consider two customers: Sofie and Lily with customers value 35 and 65 respectively.

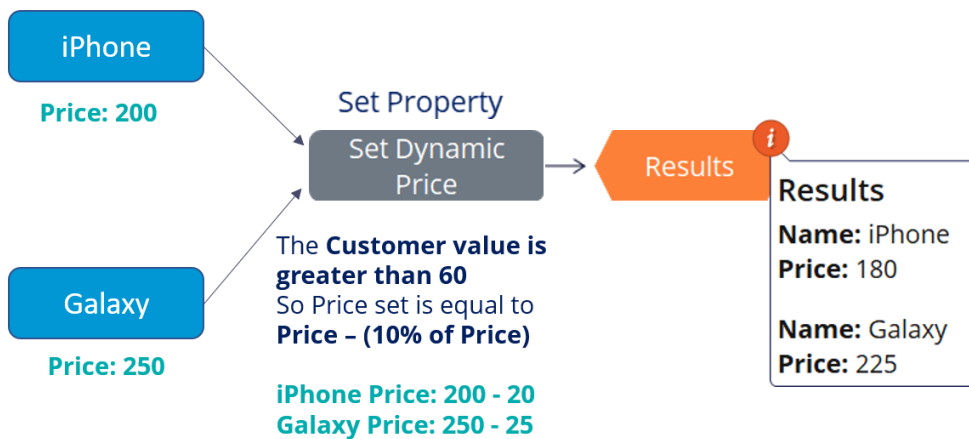
In the center of the following image, slide the vertical line to see how Sofie and Lily's **Customer value** affects the **Price** of the action offered to them.



First name: Sofie  
Customer value: 35



First name: Lily  
Customer value: 65



# Action ranking

U+ wants to offer the most profitable action to its customers.

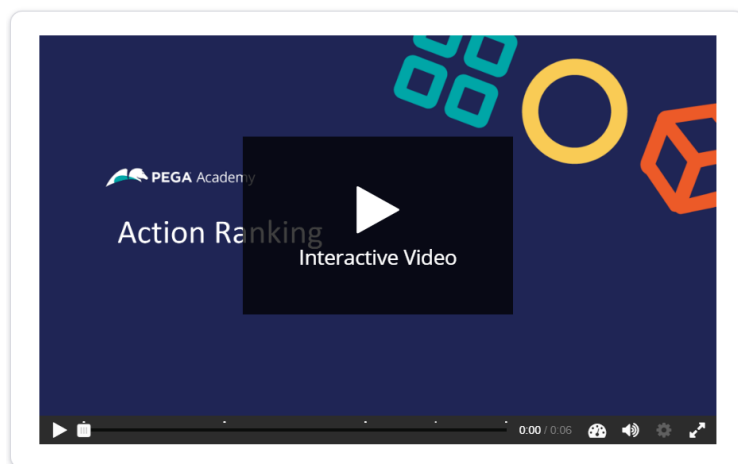
To enhance this strategy based on the new requirement, you need a new decision component that can rank the actions based on Profit and select the **highest ranked** action.

**Profit** is calculated based on **Price** and **Cost** action properties.

## Prioritize component

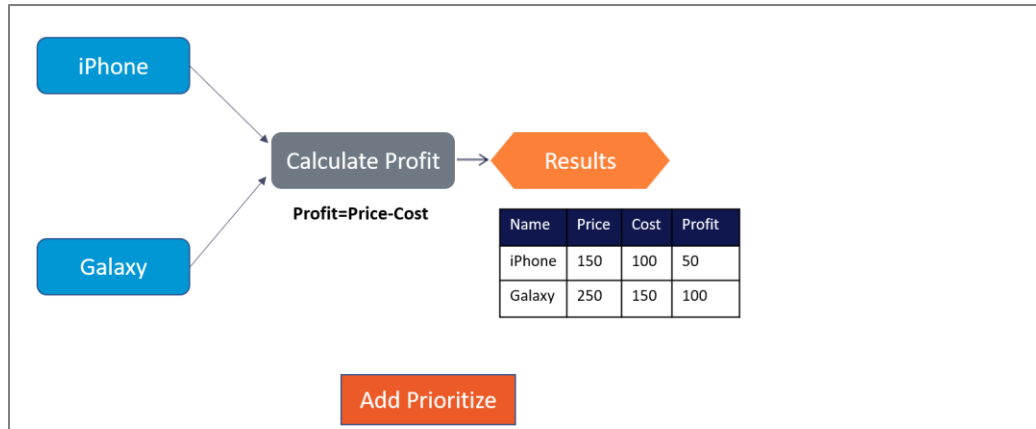
The Prioritize component is a decision strategy component used to rank actions. The Prioritize component is also used to select the top 1, top 2, or arbitrary top- $n$  actions.

Click the Play icon to learn more about action ranking in detail with the help of a sample strategy.



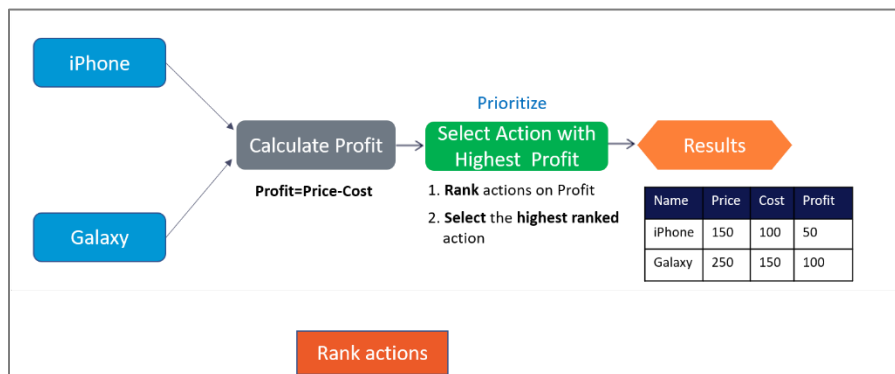
### Screen 1:

The initial strategy outputs price, cost and the profit calculated by the Set Property component. Click **Add Prioritize** to add the Prioritize component to the strategy.



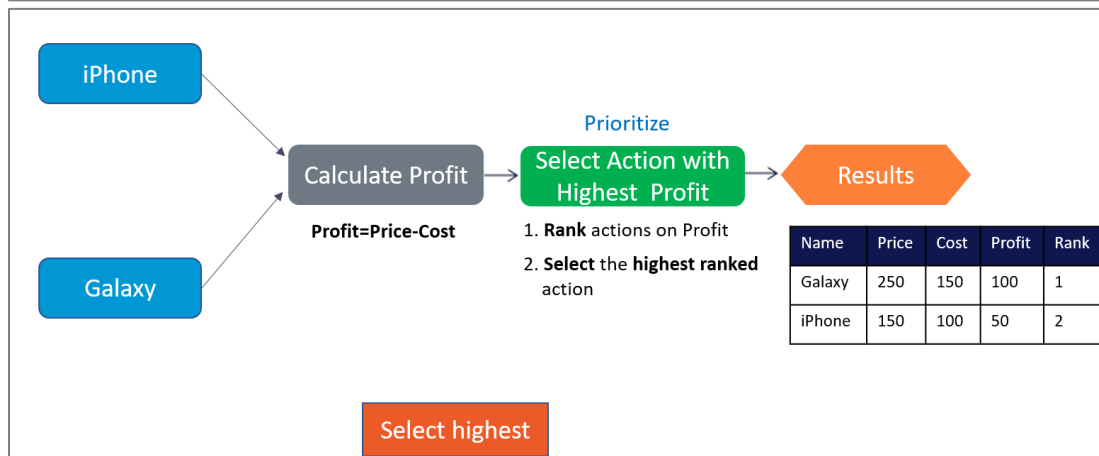
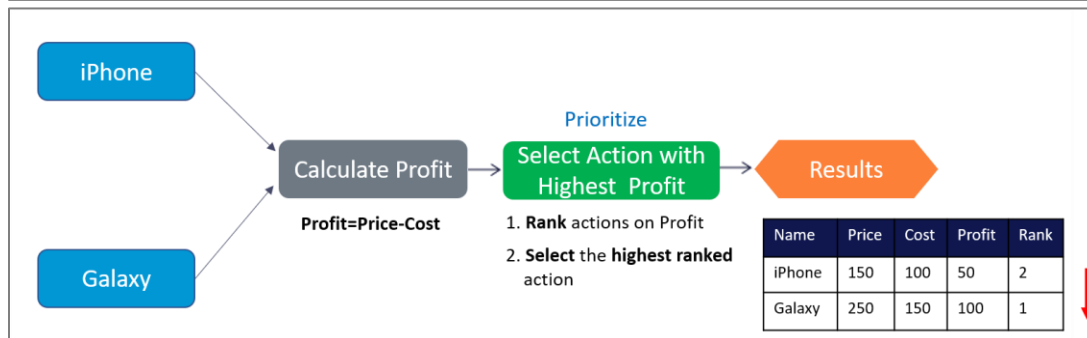
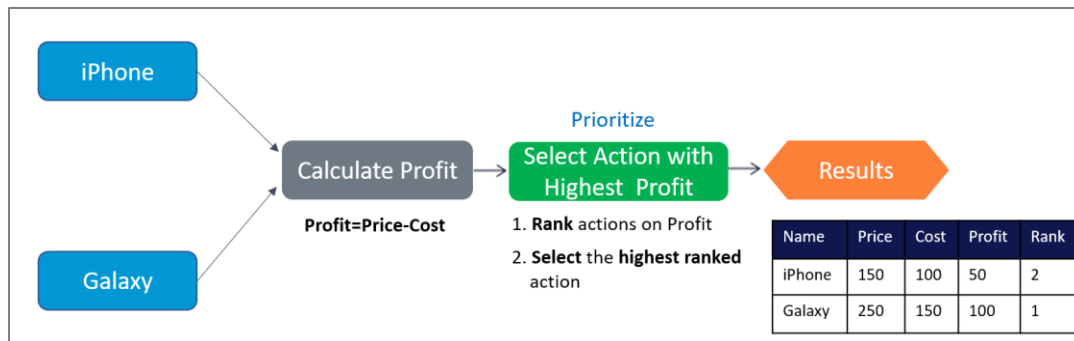
## Screen 2:

The **Prioritize** decision component can perform two operations: **rank** actions based on an expression, and select the **highest** ranked action. Click **Rank actions** to see how the component rank the actions.



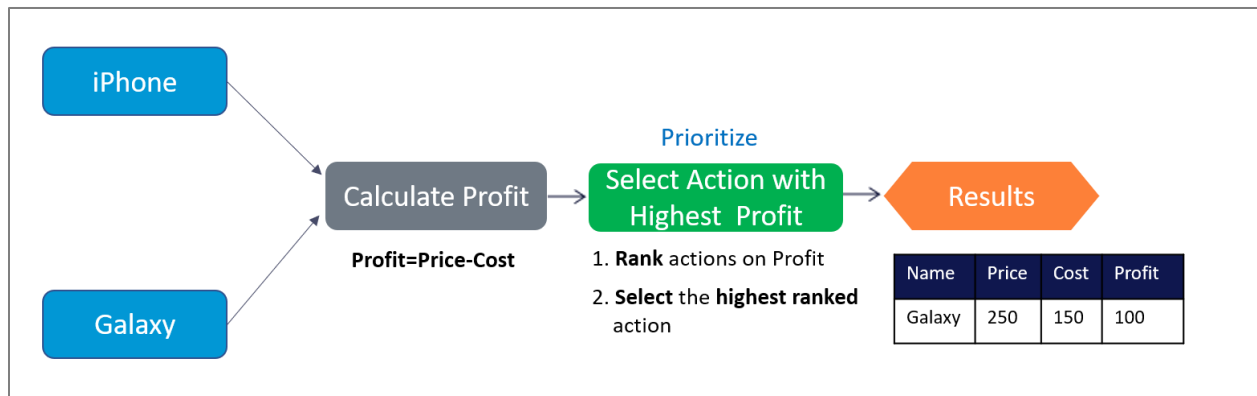
## Screen 3:

Once the actions are ranked, the prioritize component selects the highest ranked action. Click **Select highest** to see the action selected.



#### Screen 4:

You can see that the output of the result component is the highest selected action: Galaxy with a profit of 100.



## Quiz

Examine this strategy and then answer the following question to check your knowledge on action ranking.



What does the **Results** component of the strategy contain?

- ☐ Sony with profit 150
- ☐ LG with profit 100
- ☐ Panasonic with profit 50

**Feedback:** The Prioritize decision component ranks the actions and selects the highest ranked action. Hence, the Results component of the strategy contains Sony with a profit of 150.

# Creating a decision strategy

## Introduction

Decision strategies drive the Next-Best-Action. Each strategy comprises a unit of reasoning represented by decision components. How these components combine determines which action (the Next-Best-Action) will be selected for a customer. Learn the types 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.

## Video



A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

## 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.

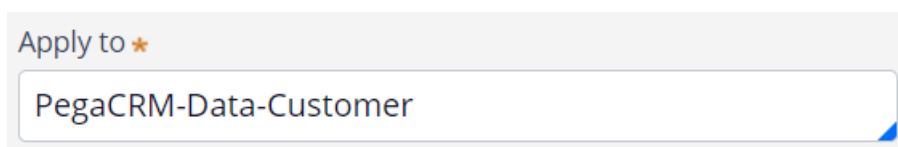
The Strategy-Results class limits the output of the strategy to the actions contained in the Business issue and Group.

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.

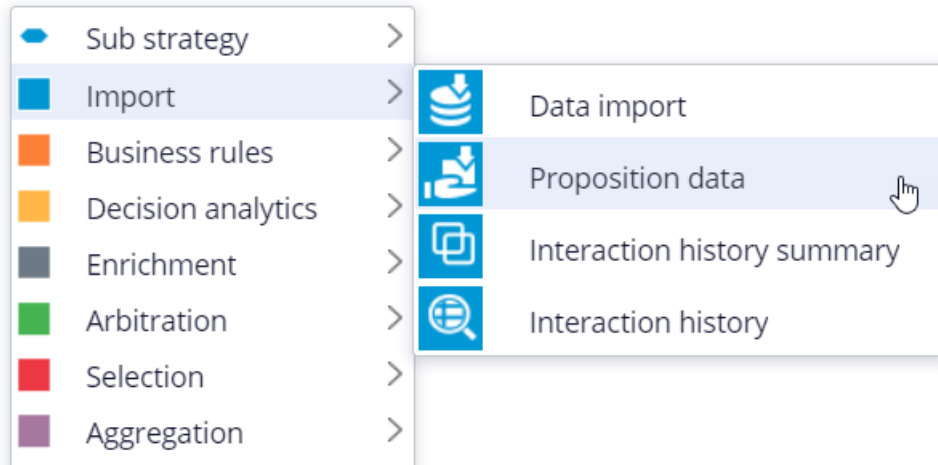


You can now start building the strategy. Right-click on the canvas to get the Context menu, which shows all component categories.

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.

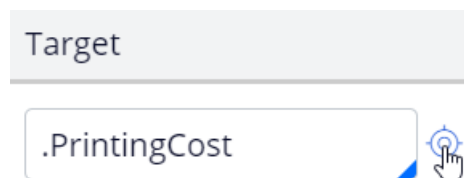
Click 'Add Item' to create the equation that will calculate the printing cost for each of the components.

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.

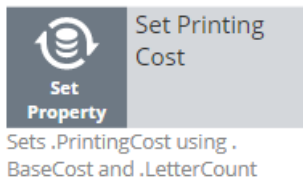
Finalize the Expression.

```
1 .BaseCost + 5 * .LetterCount
```

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   

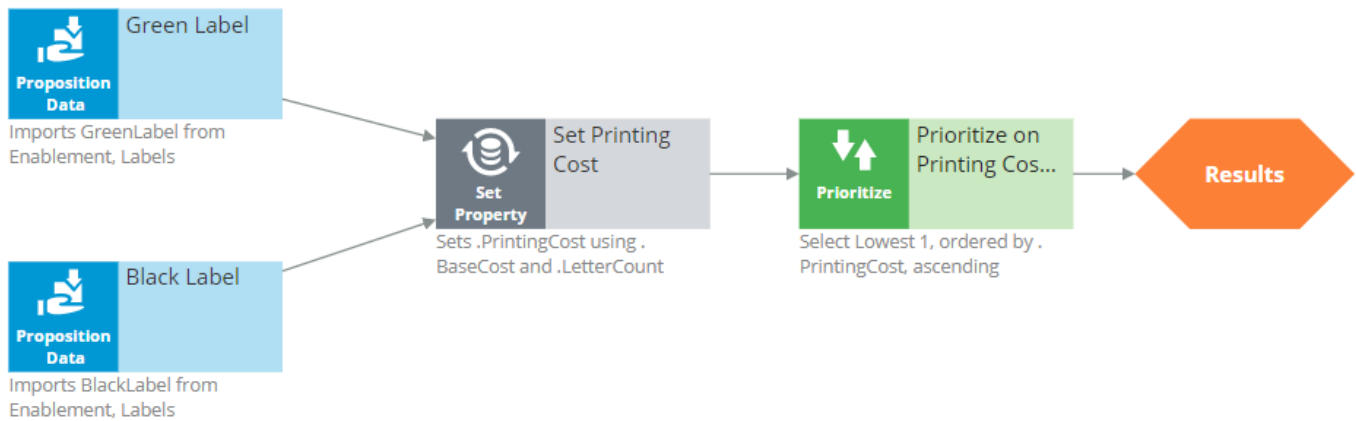
### Order by

- ☐ 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.

Creating a decision strategy -- Fri, 07/24/2020 - 06:21

To get the full experience of this content, please visit <https://academy.pega.com>

# 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.

## Video



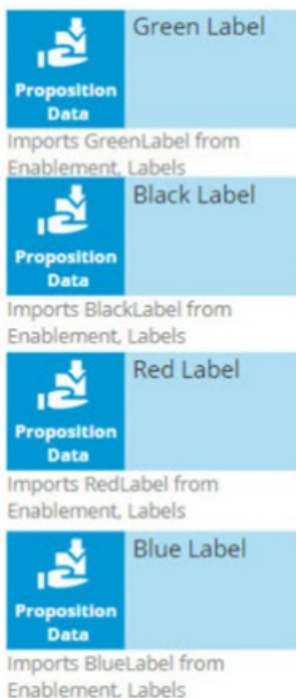
A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

## 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

Rank	Name	BaseCost	LetterCount
1	Green Label	10	10



Black Label

Imports BlackLabel from Enablement, Labels



Red Label

Imports RedLabel from Enablement, Labels



Blue Label

Imports BlueLabel from Enablement, Labels

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

Rank	Name	BaseCost	LetterCount
1	Green Label	10	10



Black Label

Imports BlackLabel from Enablement, Labels

Rank	Name	BaseCost	LetterCount
1	Black Label	20	10



Red Label

Imports RedLabel from Enablement, Labels

Rank	Name	BaseCost	LetterCount
1	Red Label	30	8



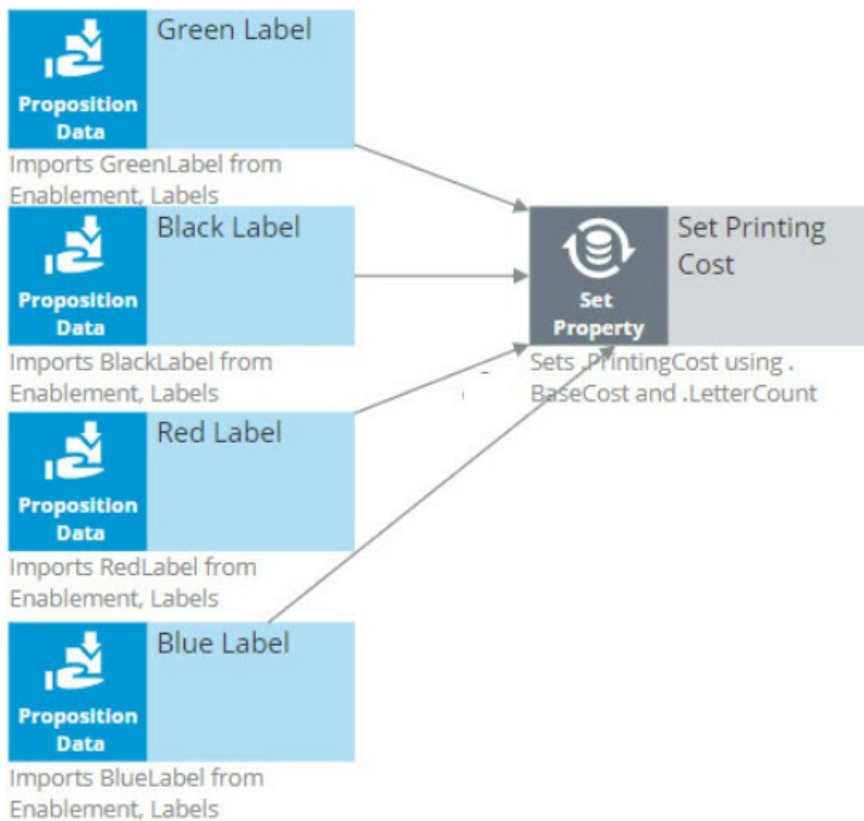
Blue Label

Imports BlueLabel from Enablement, Labels

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.

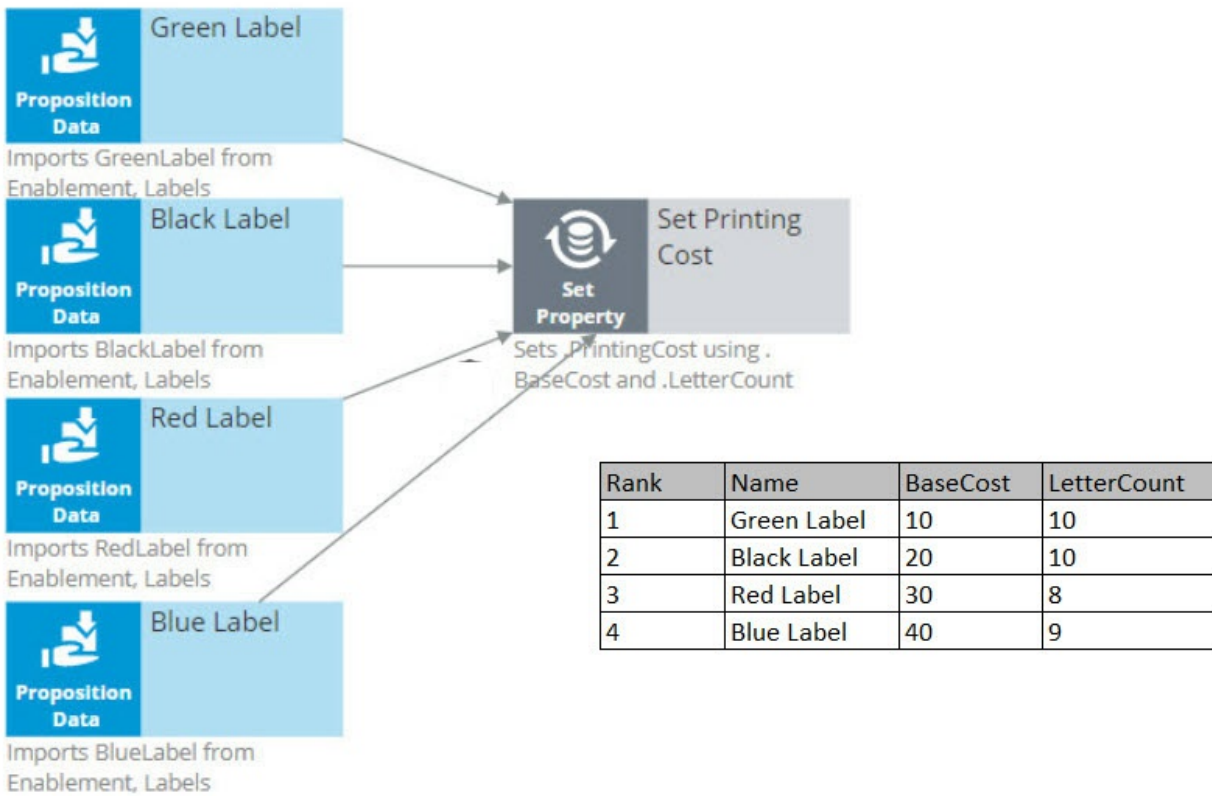
On the strategy canvas, components are connected by drawing arrows from component to component. So, what do these arrows mean exactly?



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.

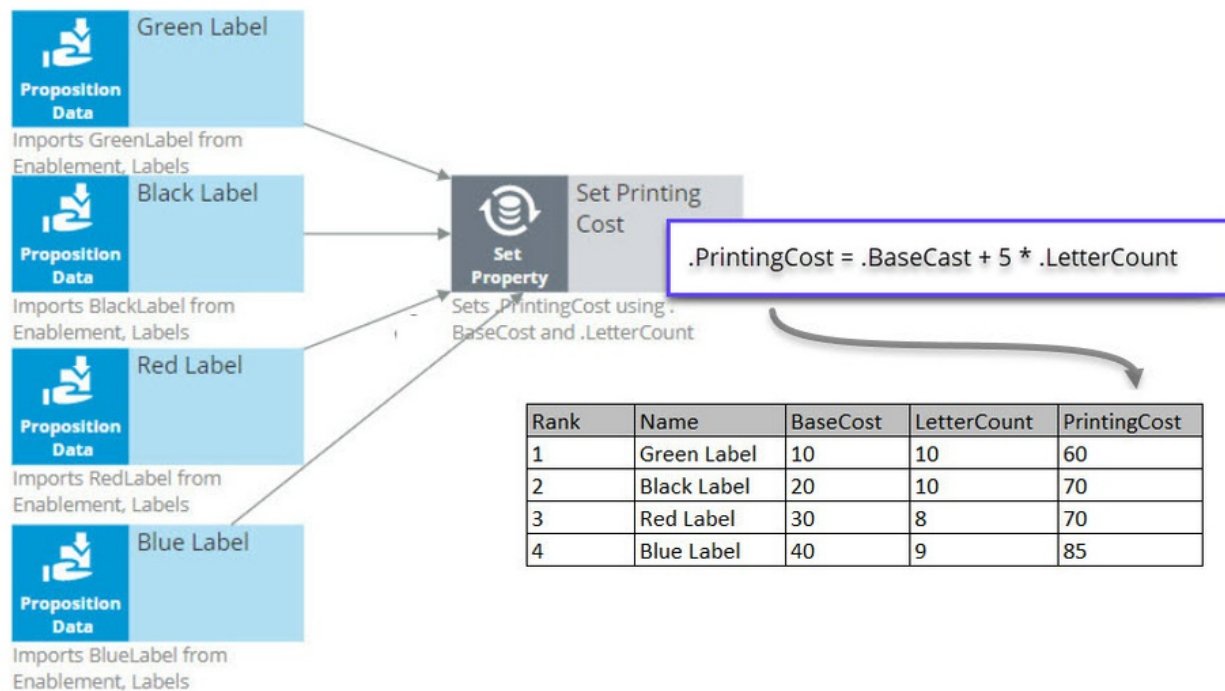
As a result, in this instance, the Green Label action has a Rank of 1, Black has a Rank of 2, Red has a Rank of 3, and Blue has a Rank of 4.

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.





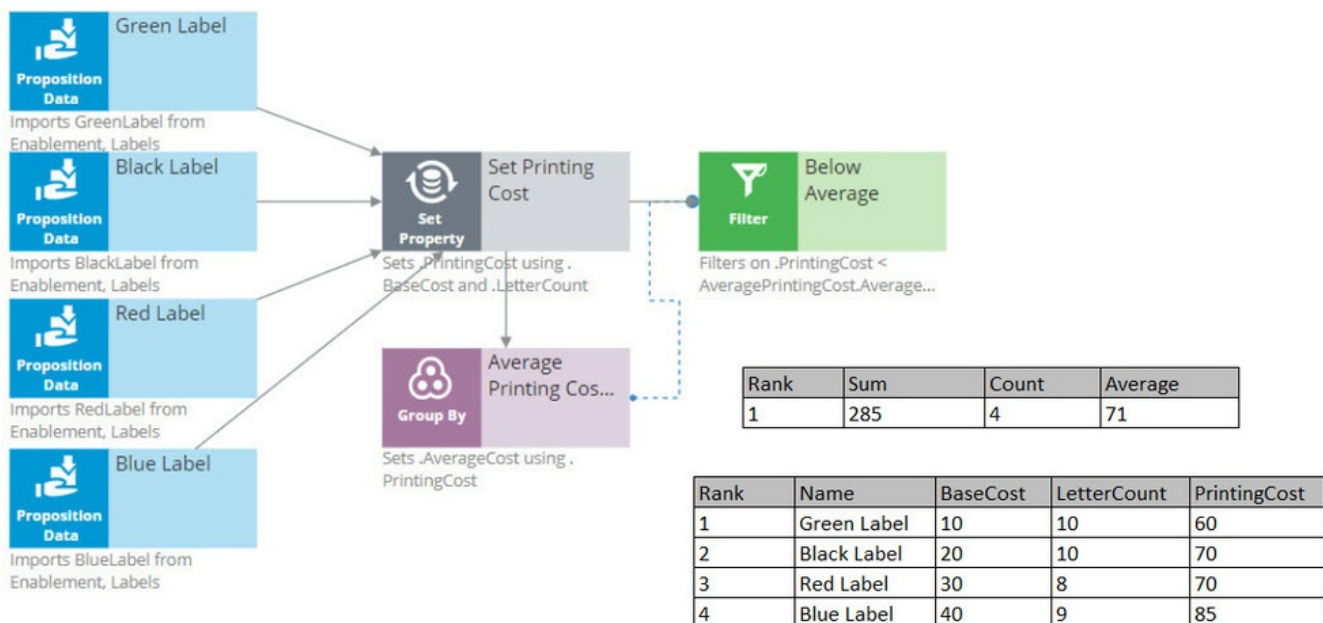
Because we are combining the data in our four Proposition Data components into one Set Property component, we only need to add one PrintingCost property to the new component, and it automatically computes the printing cost for all four actions.

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.

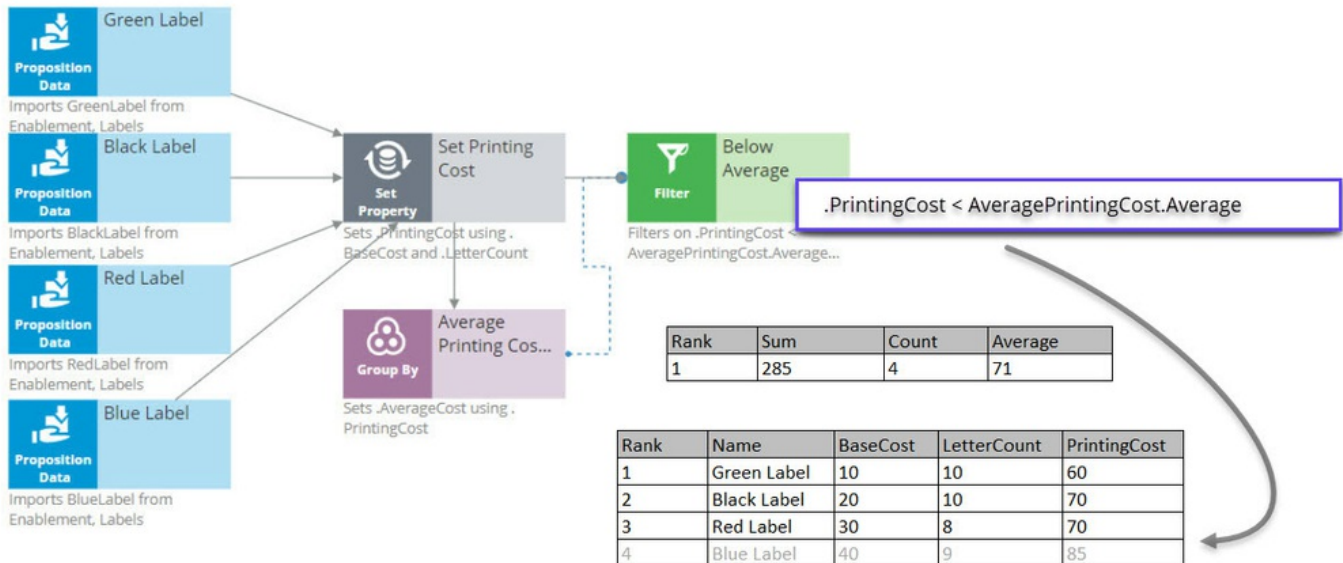




Again, we have our set of actions, each with their own specific PrintingCost value. The 'Group by' component combines all actions into one row. How does that work?

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 lower 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.

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 AverageCost'.

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.

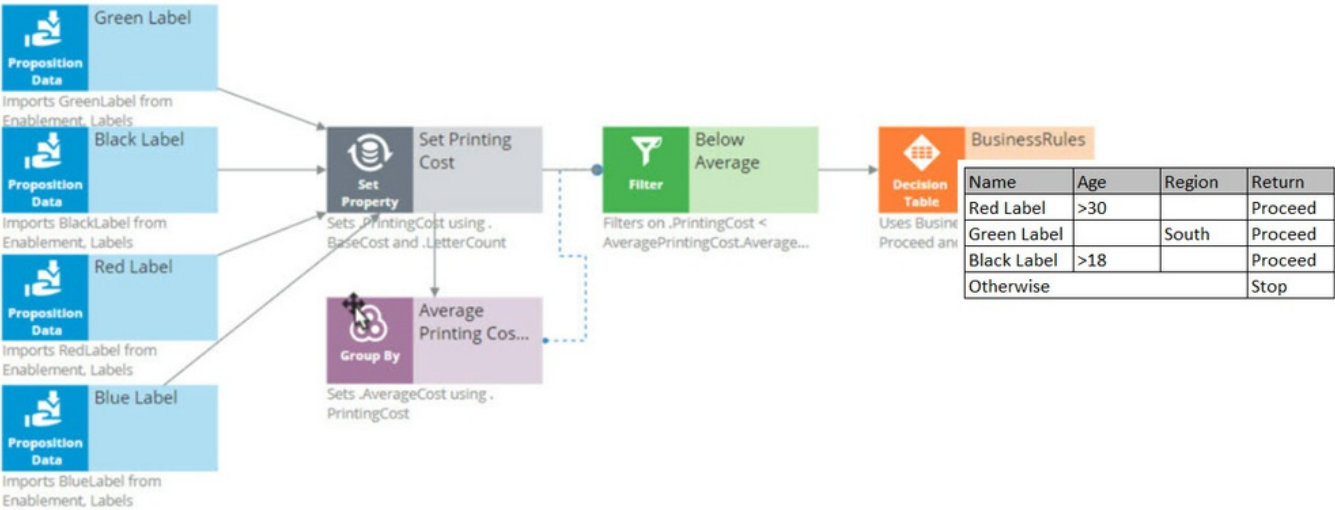
The Rank 1 average equals 71 in the 'Group by' component. This means that the filter will allow Label actions through that have a printing cost lower than 71.

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.

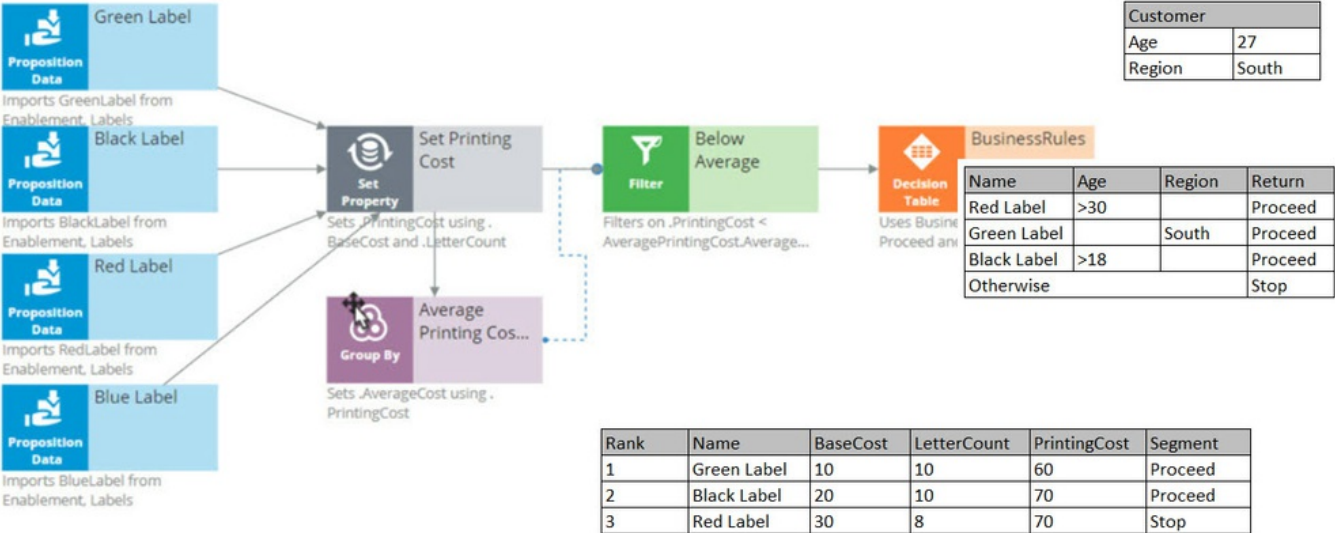
The business criteria say that the Red Label action can be offered if the customer’s age is over 30 and they are from any region. If these criteria are met, the Return value is ‘Proceed’.

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.

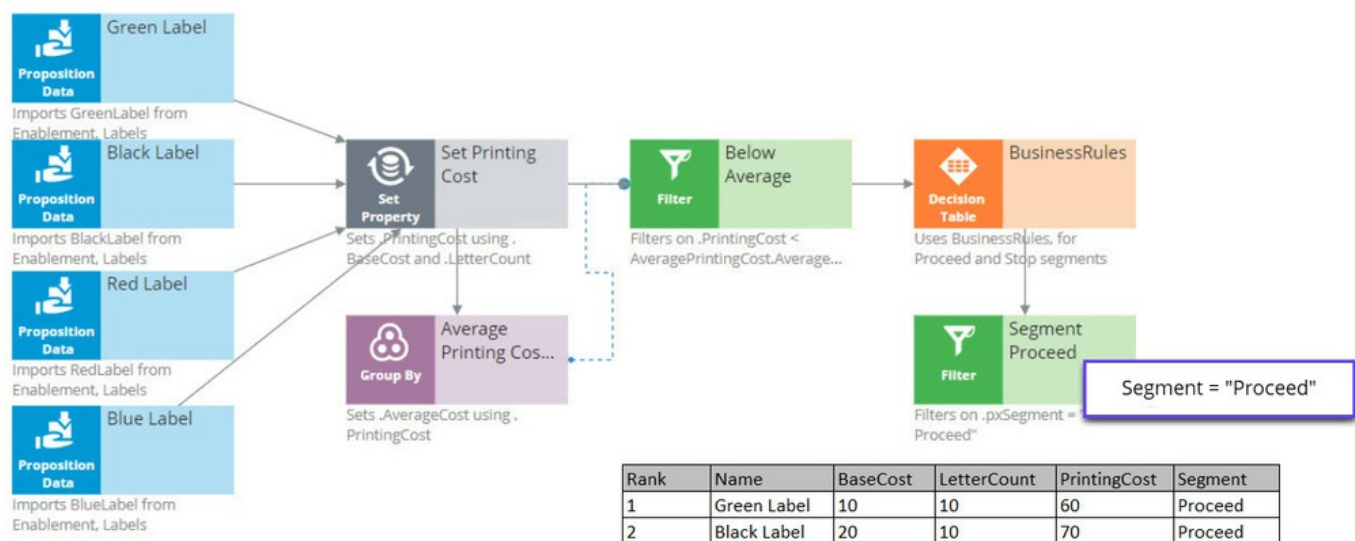
Next, it looks at the Black action and sees that the criteria for Black is that the customer’s age is greater than 18. This customer is 27.

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.

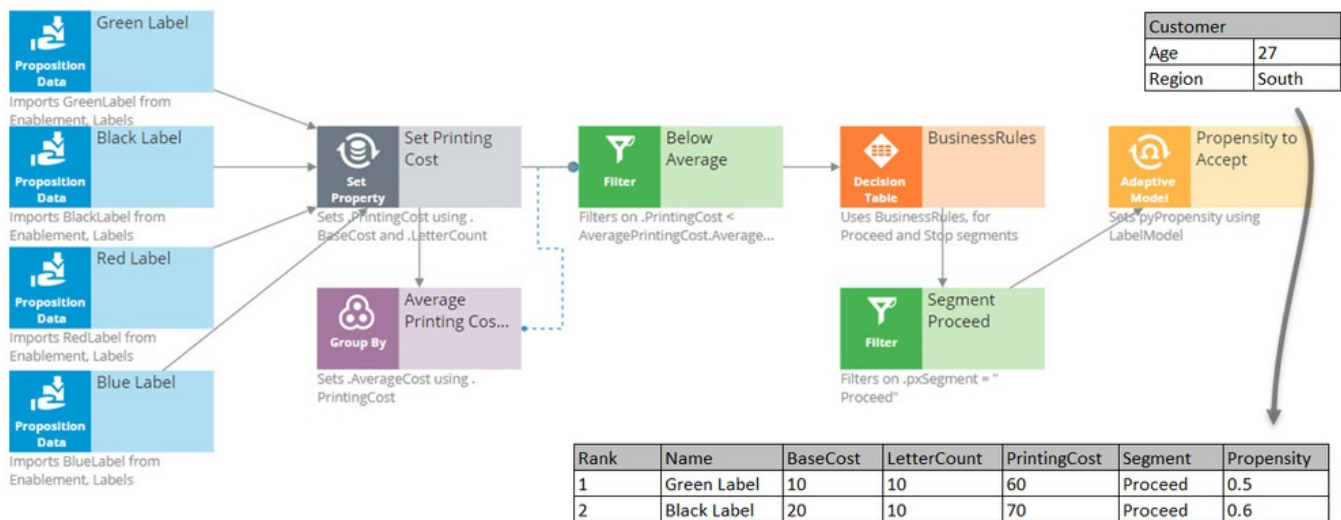
First is the Green Label. Green is allowed through, which means its properties will be available in the new component.

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.

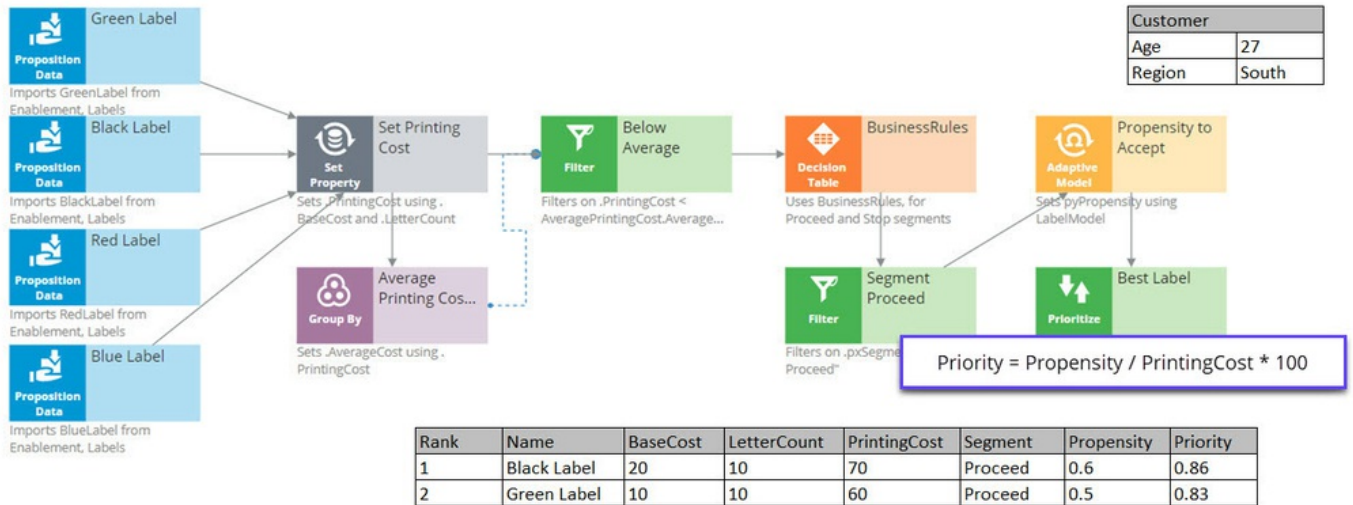
For the Black Label action, the likelihood turns out to be 0.6.

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.

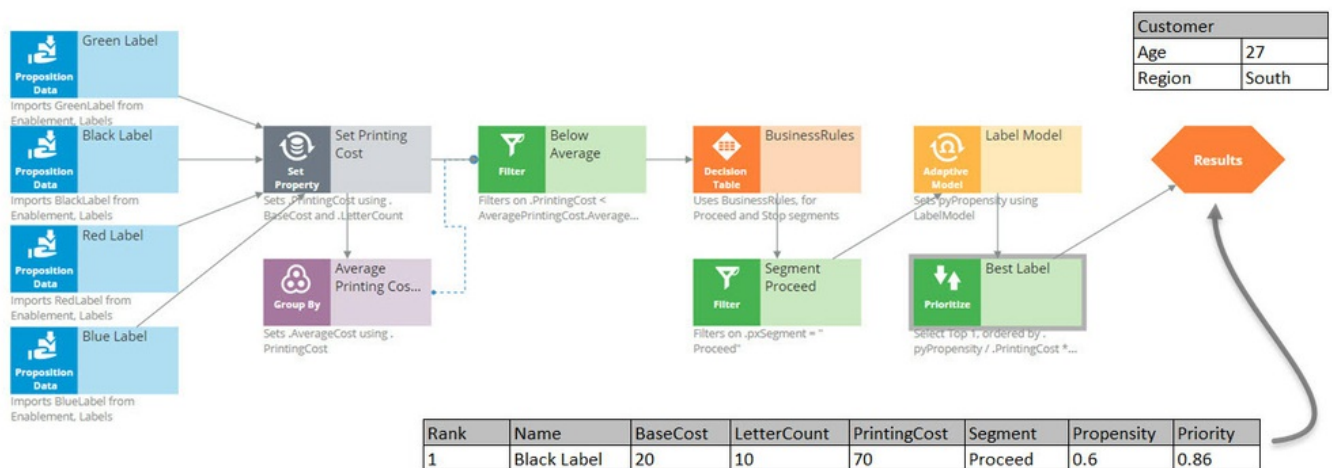
Because 0.86 is higher than 0.83, the Black Label action is now ranked number one.

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.





Decision strategy execution -- Fri, 07/24/2020 - 06:23  
To get the full experience of this content, please visit <https://academy.pega.com>

# Building a scorecard to calculate the credit score

## Duration

10 mins

## 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.

## Video



A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

## Transcript

This demo will show you how to build a scorecard.

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.



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.



Scorecard configuration for **.Income\*12**:

Condition	Score
<= 25000	50
<= 50000	100
<= 75000	150
Otherwise	200

Total Score: 1

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.

Scorecard configuration for **.Age**:

Condition	Score
<= 18	0
<= 25	30
<= 35	50
<= 45	100
<= 55	150
Otherwise	200

Total Score: 1

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.

**Refresh**

Minimum score: 50.0      Maximum score: 500.0

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.

Building a scorecard to calculate the credit score -- Tue, 06/23/2020 - 02:09

To get the full experience of this content, please visit <https://academy.pega.com>

# Creating an engagement strategy

## Duration

10 mins

## Introduction

Scorecard results can be used to describe which actions are appropriate for a customer in the form of eligibility, applicability, or suitability rules. Learn how to build a decision strategy that leverages a scorecard to define engagement policy rules. Gain experience defining eligibility rules using a decision strategy in Next-Best-Action Designer.

## Video



A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

## Transcript

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

### Engagement policy

E Eligibility ?

(isCustomer is true)

and (Age is greater than 18)

A Applicability ?

(Has Cards is equal to N)

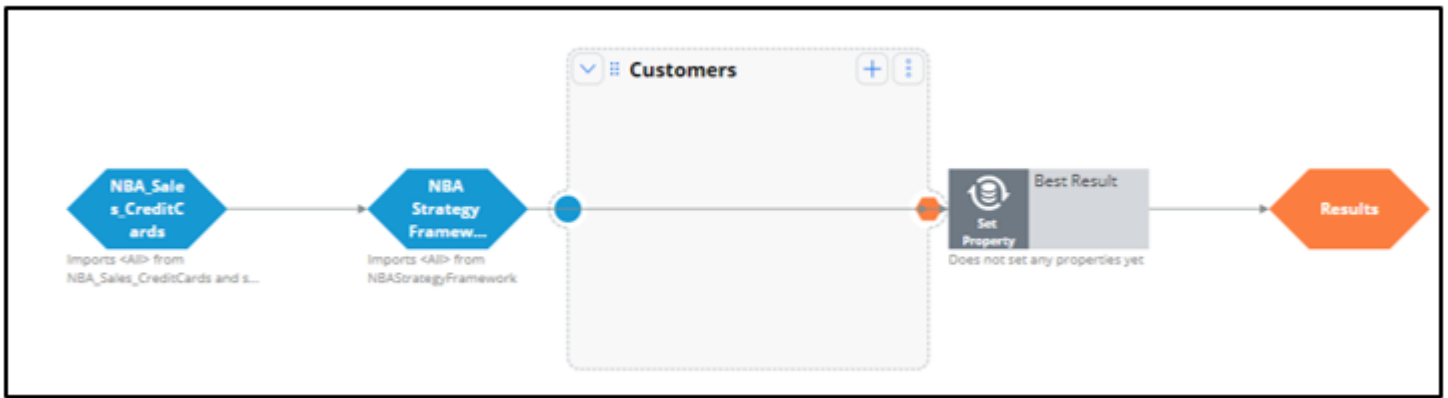
S Suitability ?

No group criteria defined

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.

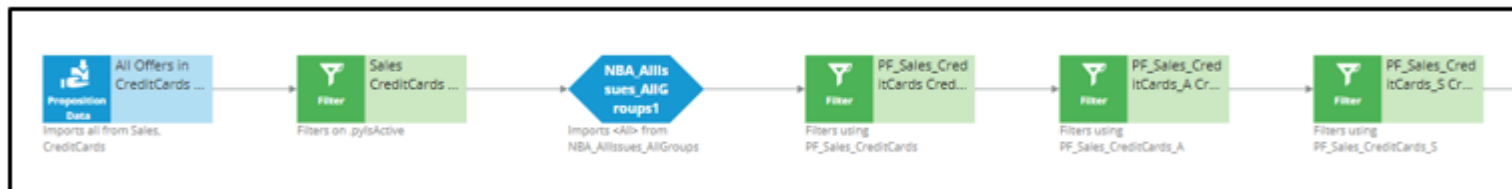


The first component of the strategy is responsible for all Engagement Policy rules.

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.

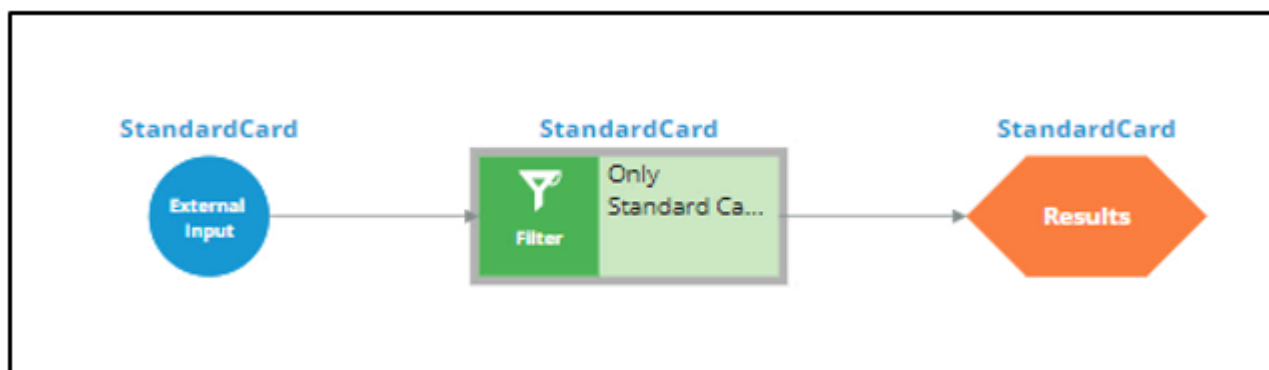
Start by naming the Filter component.

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.

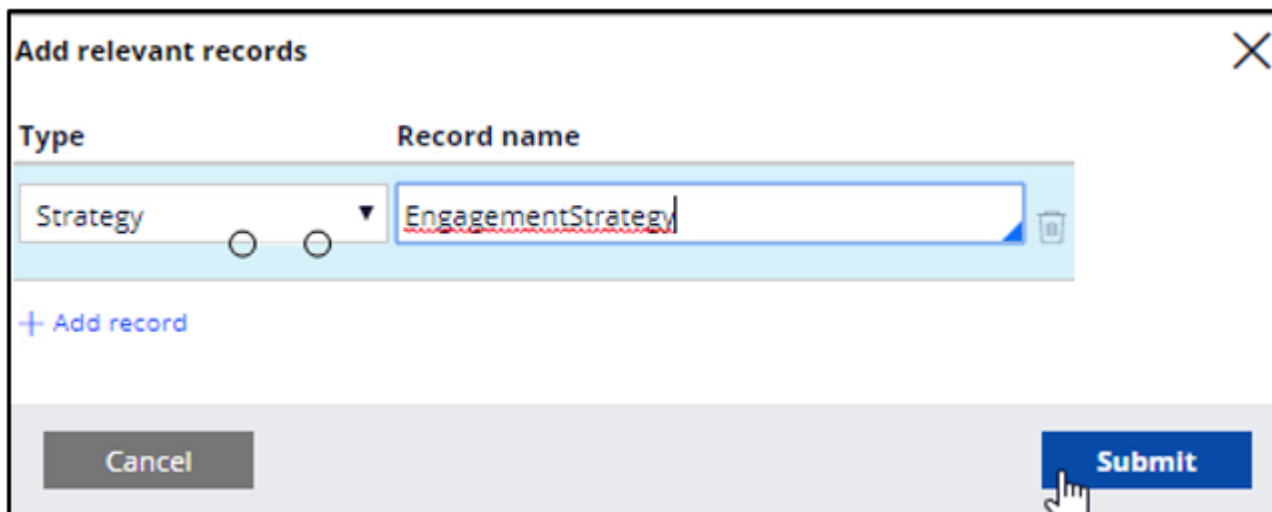


To make this strategy accessible in Next-Best-Action Designer as an Engagement Strategy, log in to the DEV STUDIO.

Navigate to the Relevant Records section.

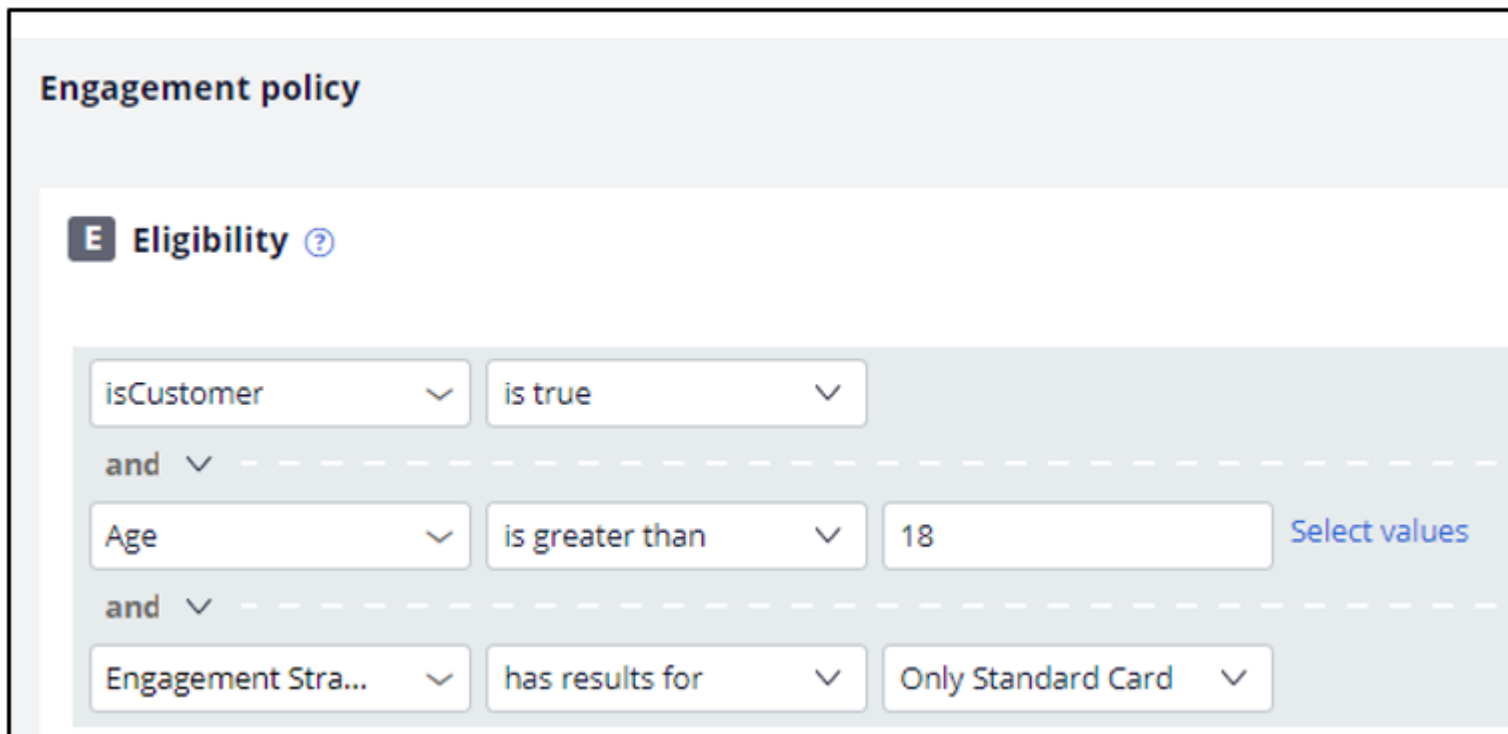
Select the appropriate Class name for adding a Record.

Select the appropriate Type and Record name. In this case, you want to make the strategy accessible in NBA Designer, so set the Type to Strategy and the Record name to EngagementStrategy, the actual name of decision strategy.



The screenshot shows a dialog box titled "Add relevant records" with a close button (X) in the top right corner. It contains two columns: "Type" and "Record name". In the "Type" column, a dropdown menu is open showing "Strategy" as the selected option. In the "Record name" column, a text input field contains "EngagementStrategy". Below the input fields is a blue "+ Add record" button. At the bottom of the dialog are two buttons: "Cancel" on the left and "Submit" on the right, with a mouse cursor pointing at the "Submit" button.

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 shows the "Engagement policy" configuration screen. It has a header "Engagement policy" and a sub-section "E Eligibility" with a help icon. Below this, there are three rows of conditions separated by "and" connectors. The first row has "isCustomer" (dropdown) and "is true" (dropdown). The second row has "Age" (dropdown), "is greater than" (dropdown), and "18" (text input), with a "Select values" link to the right. The third row has "Engagement Stra..." (dropdown), "has results for" (dropdown), and "Only Standard Card" (dropdown).

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.

Creating an engagement strategy -- Tue, 06/23/2020 - 02:11

To get the full experience of this content, please visit <https://academy.pega.com>

# Using a scorecard in a decision strategy

## Duration

10 mins

## 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.

## Video



A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

## Transcript

This demo will show you how to use a Scorecard in a decision strategy to determine customer suitability for a credit card.

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.

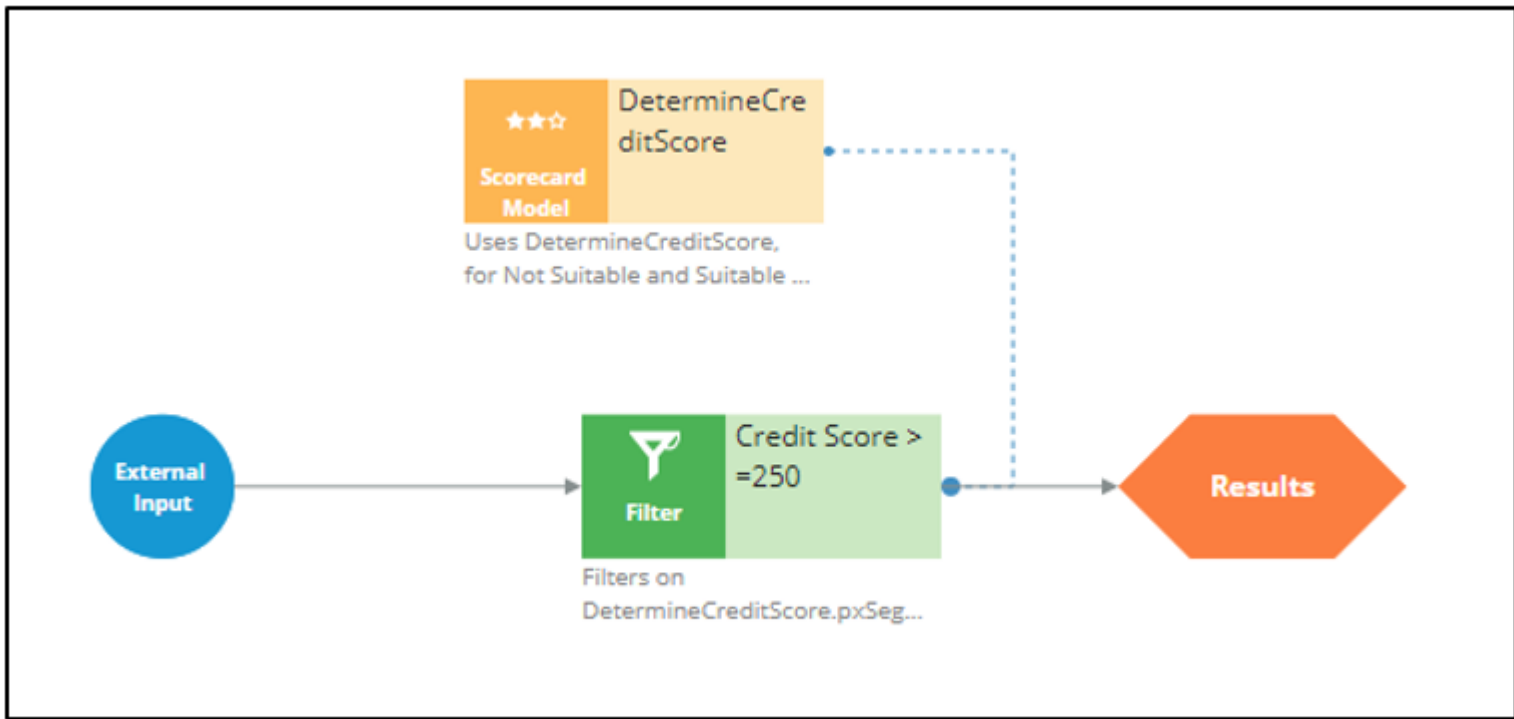
You can also see the segment results. The Scorecard returns value **Not Suitable** if the credit score is less than 250. Otherwise, it outputs the result **Suitable**.

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, use the `<ComponentName>.<PropertyName>` construct.

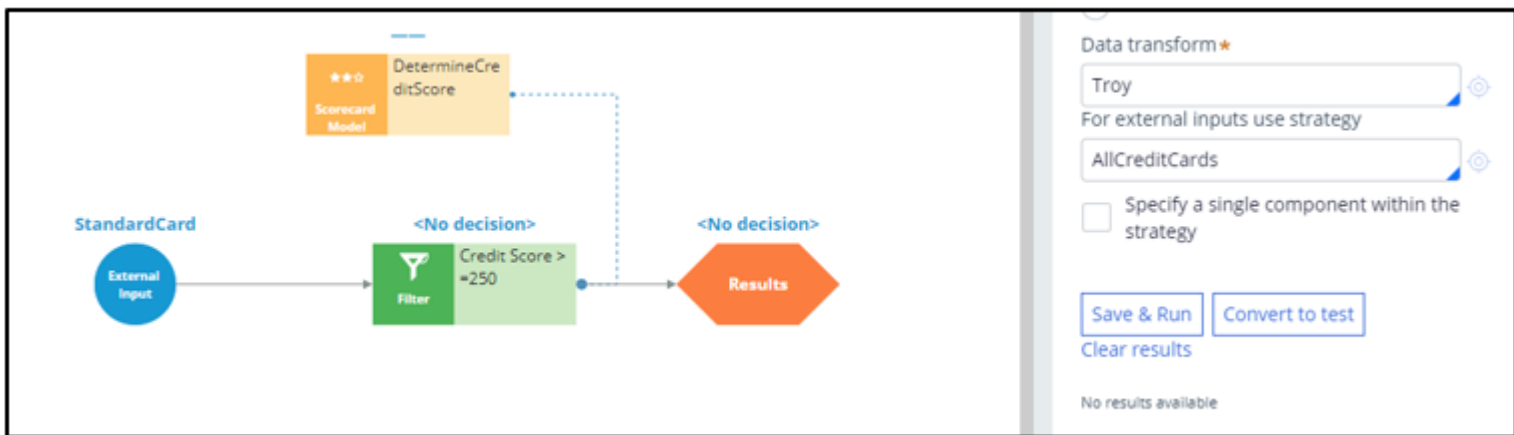


Note that the usage of the External Inputs component also gives you the ability to create more complex conditions, where Action attributes are also used.

Save the configurations.

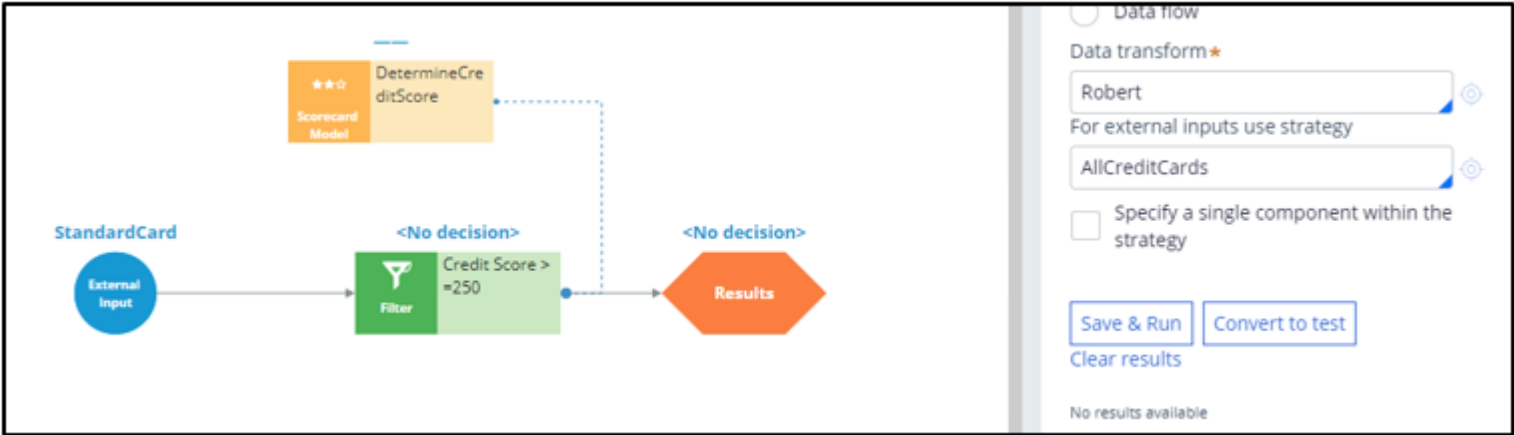
Now, test the strategy using the customer profiles, Troy and Robert.  
For external inputs, consider all credit cards available.

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	DetermineCredit Score

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.

Source components   Scorecard   **Score mapping**

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

☒ Enable score mapping

Set  equal to Score

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.

**Set property properties**

Name \*

Component ID

Description ☒ Use generated ☐ Use custom

Source components **Target**

**Define action, target, and source**

+ Add item X Delete

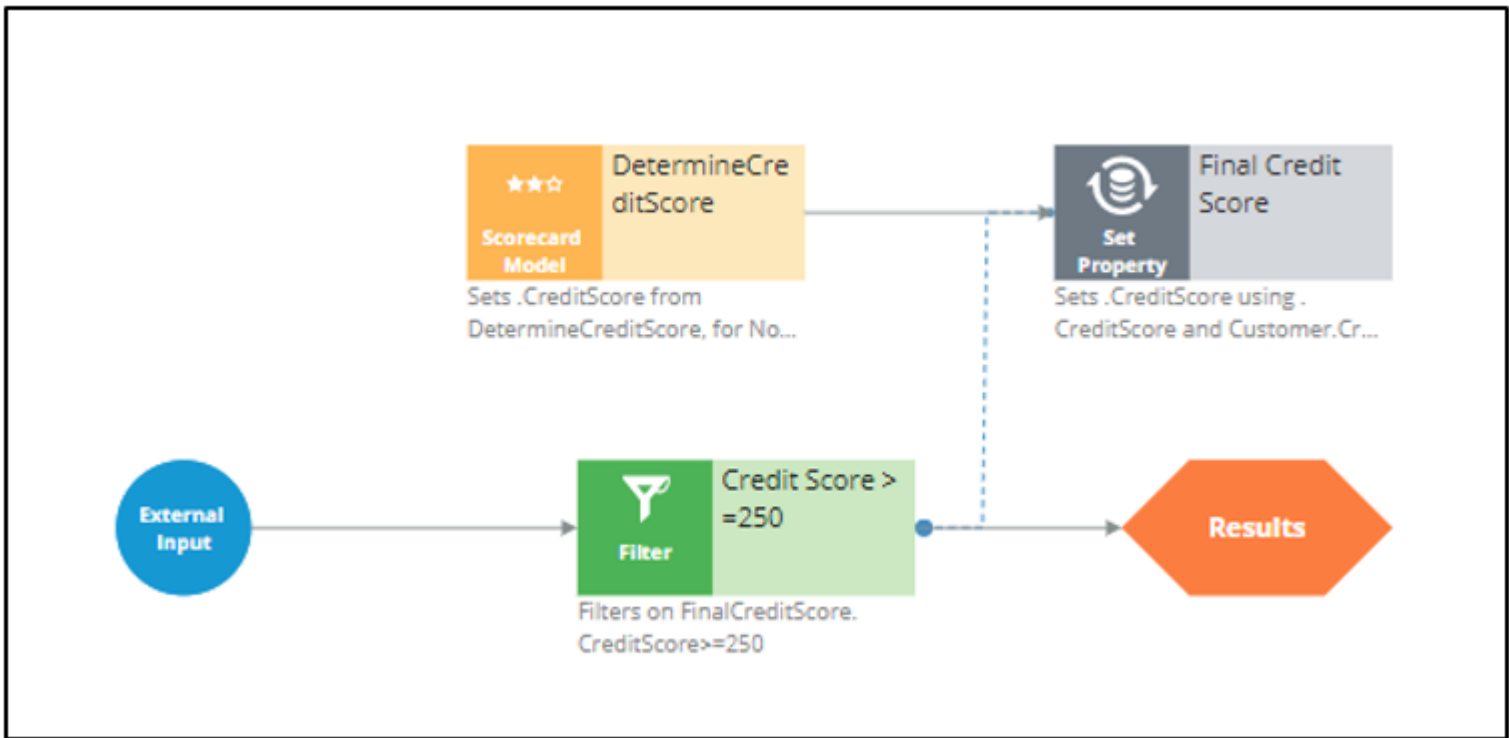
Action	Target	Source
Se ▾	<input type="text" value=".CreditScore"/>	equal to <input type="text" value="@if(@PropertyHasValue(Cus..."/>

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.

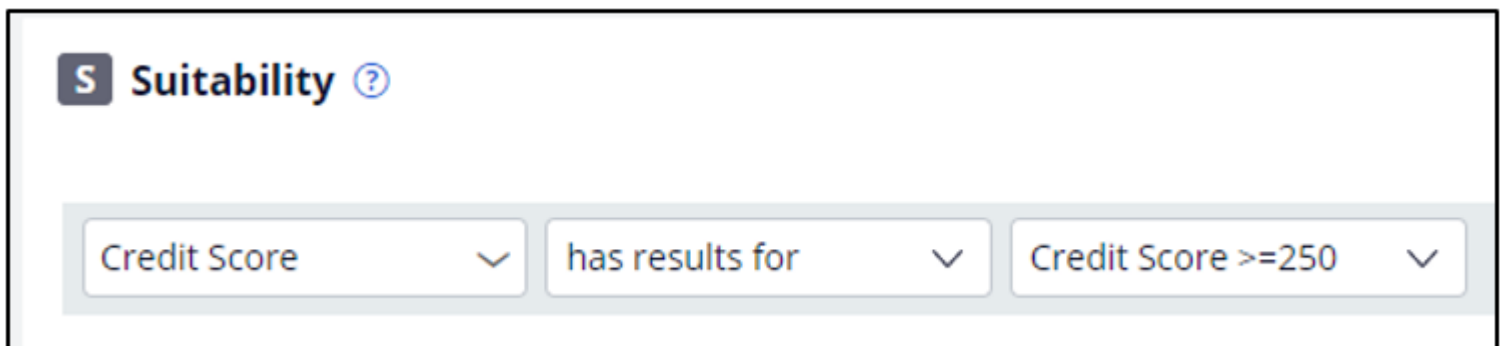
For Robert, the strategy is outputting results.

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  $\geq 250$* .

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.

Using a scorecard in a decision strategy -- Tue, 06/23/2020 - 02:11

To get the full experience of this content, please visit <https://academy.pega.com>

# 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**.

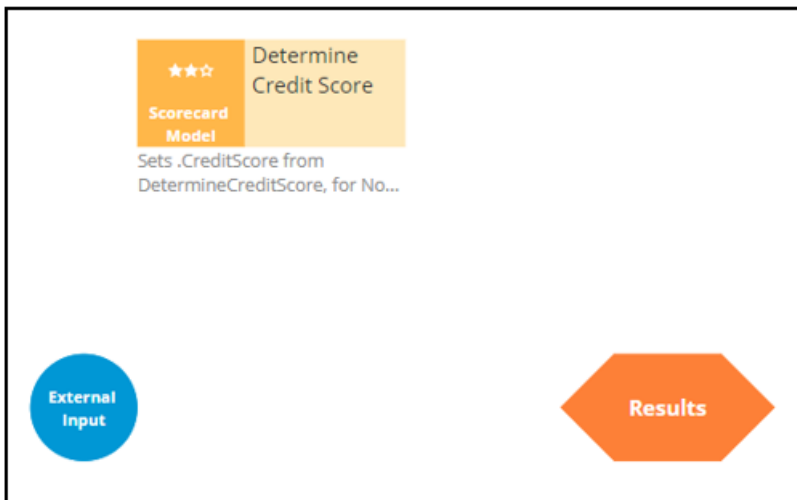
Condition	Risk Segment
If <b>Outstanding loan amount</b> < \$10000 AND <b>Credit score</b> is > 600	AAA
If <b>Outstanding loan amount</b> between \$10000 and \$25000 AND <b>Credit score</b> is > 600	AAA-
If <b>Outstanding loan amount</b> between \$25000 and \$50000 AND <b>Credit score</b> is > 600	AA
If <b>Outstanding loan amount</b> > \$50000 AND <b>Credit score</b> is > 600	AA-
If <b>Outstanding loan amount</b> < \$25000 AND <b>Credit score</b> is between 400 and 600	BBB
If <b>Outstanding loan amount</b> between \$25000 and \$50000 AND <b>Credit score</b> is between 400 and 600	BBB-
If <b>Outstanding loan amount</b> > \$50000 AND <b>Credit score</b> is between 400 and 600	BB-
If <b>Credit score</b> is between 200 and 400	CCC
If <b>Outstanding loan amount</b> AND <b>Credit score</b> 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.



**Scorecard model properties** ✕

Name \*

Component ID DetermineCreditScore

Description ☒ Use generated ☐ Use custom

Source components **Scorecard** Score mapping

Defined on ☒ Applies to ☐ Strategy result

PegaCRM-Data-Customer

Scorecard model

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**

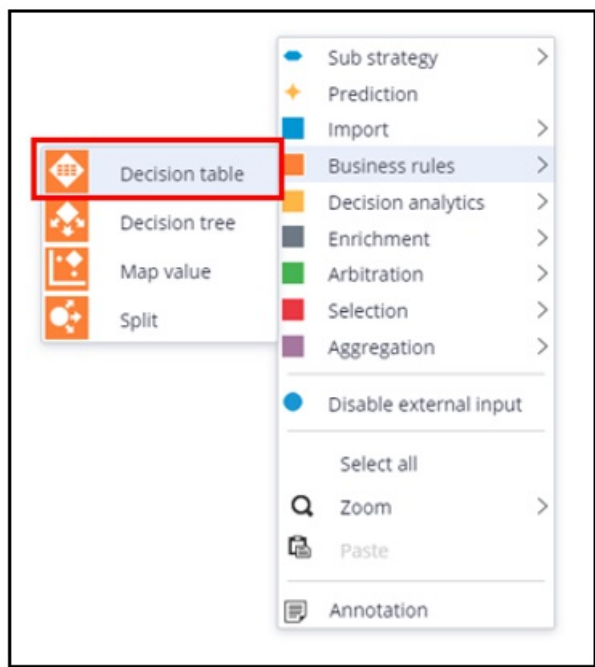
**Default mapping**  
Component sets .pxSegment equal to the result of the scorecard

☒ Enable score mapping

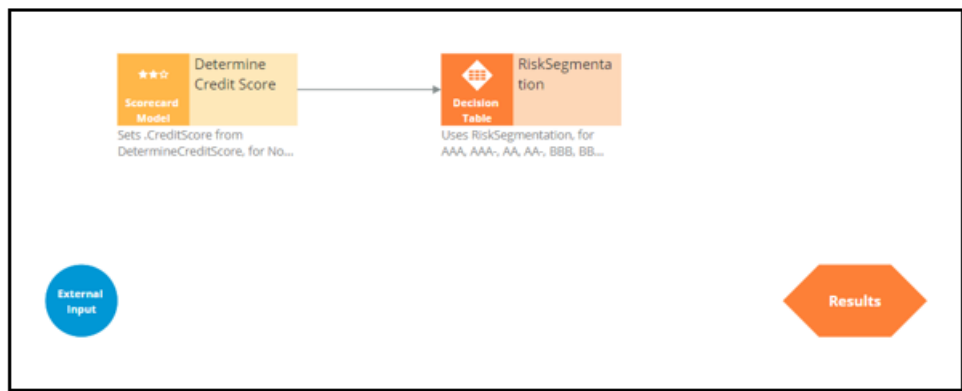
Set  equal to Score

## 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	Parameters	Pages & Classes	Test cases	Specifications
Name	Description	Data type			
CreditScore	Customer's credit score	Integer	▼		

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

Save Cancel

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

Save Cancel

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

Result
 

AAA

Target property

+

Result
 

AAA-

Target property

+

Result
 

AA

Target property

+

Result
 

AA-

Target property

+

Result
 

BBB

Target property

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

	Conditions		Actions	
	◦ Outstanding Loan Amount	◦ 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

RiskSegment

Supply data via

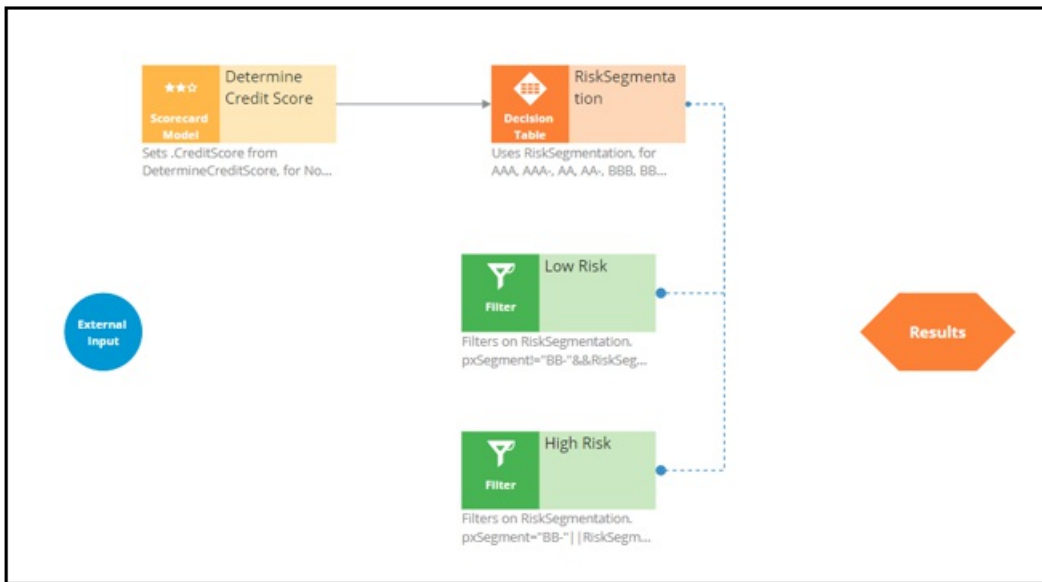
Parameters

CreditScore

.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.



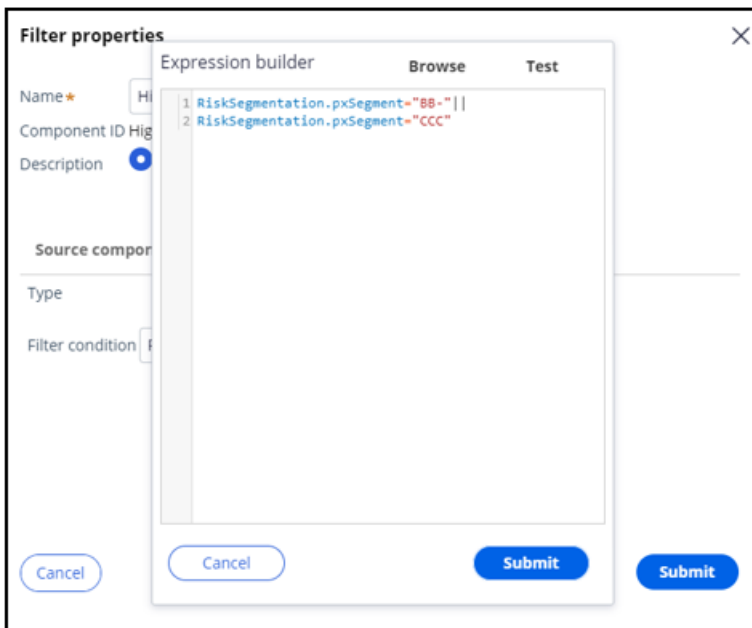
Customers in the risk category BB- and CCC are not eligible for credit cards, but all other customers are eligible. The **pxSegment** Strategy property contains the output of the Decision Table. So, create relevant expressions.

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

*RiskSegmentation.pxSegment != "CCC"*

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

*RiskSegmentation.pxSegment = "CCC"*

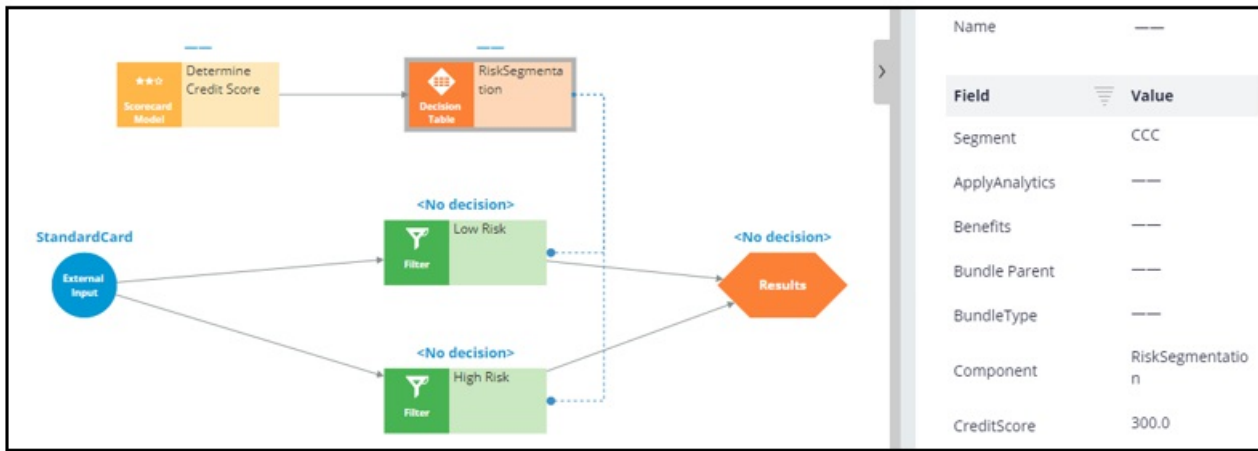


When you test the strategy using the Robert data transform, you will see that Robert falls under the category of high-risk customers, as he has a huge outstanding loan amount, over 50,000.

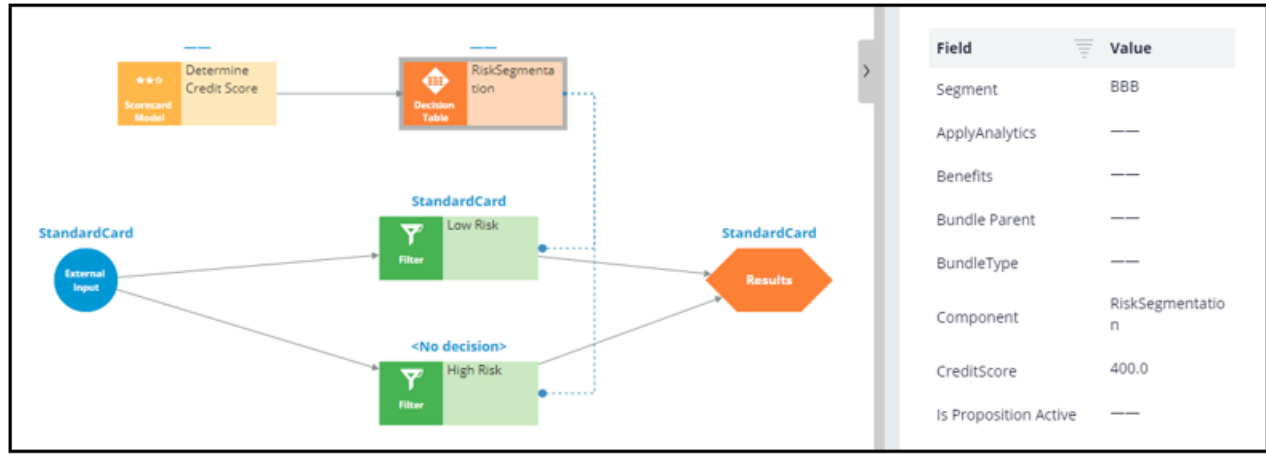


Therefore, you can expect his risk segment to be CCC.





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 in engagement strategies

## Introduction

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

## Video



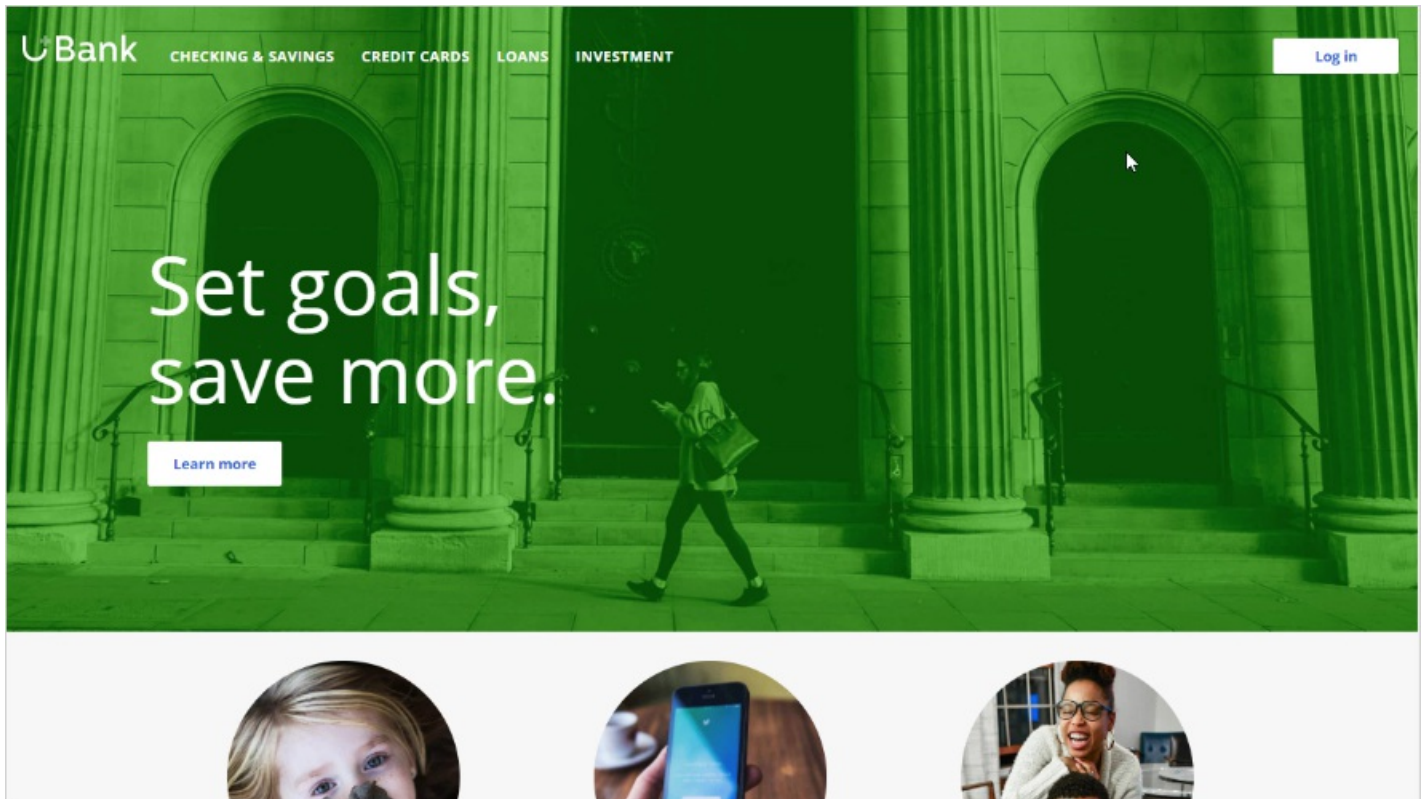
A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

## Transcript

This demo will show you how to use a predictive model in an engagement strategy to determine customer applicability for a retention offer.


Currently, U+ Bank is doing cross-sell on the web by showing various credit cards to all customers who log in to their 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 will only be shown to loyal customers.




To meet this business requirement, a decisioning administrator has already set up the new business structure by defining a

new Issue/Group, the Retention/Loyalty offers under Taxonomy.

 **Taxonomy**  
Define your Next-Best-Action business structures and customer states


---

**Taxonomy**


 **Business structure**

**Issues / Groups**


Retention

 Loyalty offers

Sales

 CreditCards

The retention offer, Extra Miles 5K, has also been created for this issue/group.

 **Offer**

**1 Offer (0 with specialized policies)**

Name	Specialized policies
Extra miles 5K	

The next step is to create an applicability condition that makes a customer qualify for a retention offer when there is a churn risk. Since the churn risk is predicted using a predictive model, you need to use a decision strategy to define this condition.

A decisioning administrator has already created the RetentionStrategy. You can use this strategy as basis for the applicability condition.

A data scientist has already created the predictive model using Pega's built-in machine learning capabilities to identify the customers who are likely to churn in the near future.

To use the predictive model in the decision strategy, add a predictive model component to the canvas and configure it to reference the model. Note that this list contains all the predictive models that are available in the system. These include models created using Pega machine learning, imported PMML or H2O.ai models, and externally executed models built in Google ML or Amazon SageMaker.

Select the desired predictive model from the list.

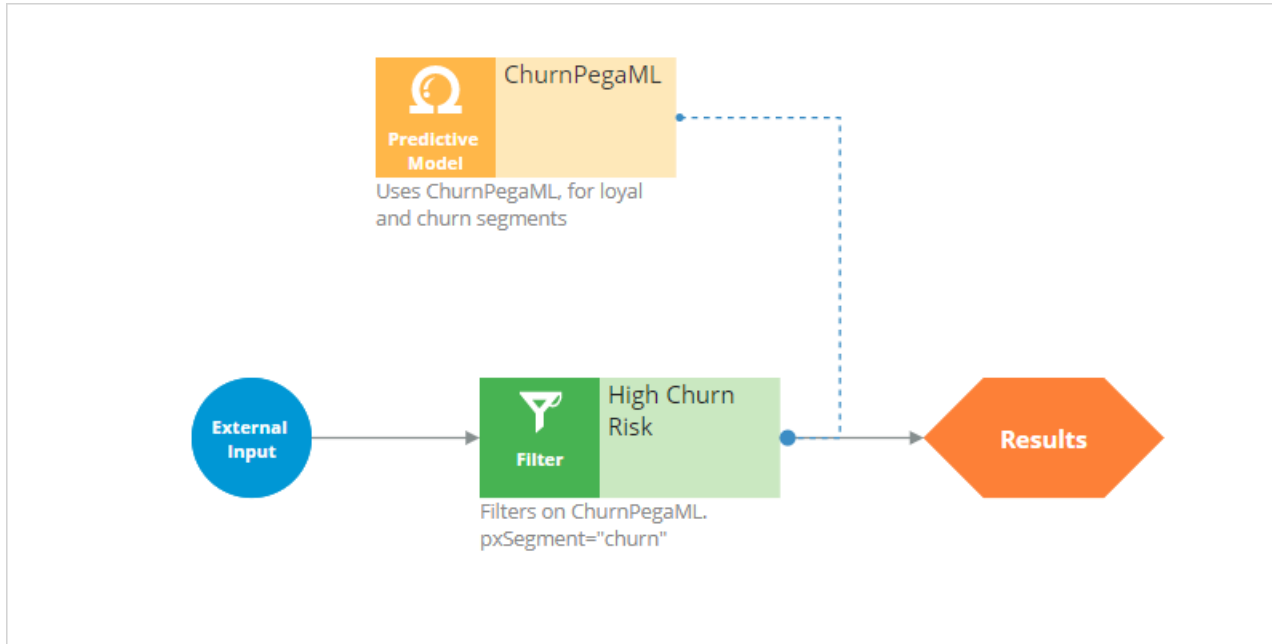
When you open the predictive model, notice that in the **Classification groups** section, the classes are divided into two groups, and their results are labeled "loyal" and "churn" to predict customer behavior.

You can now use a filter component to express the condition under which the retention offer is applicable.

You want this strategy to output a retention offer only if the result of the predictive model is “churn”. The result of the predictive model is stored in the pxSegment property.

Therefore, define the filter condition to output a retention offer for which the pxSegment property is equal to “churn”.

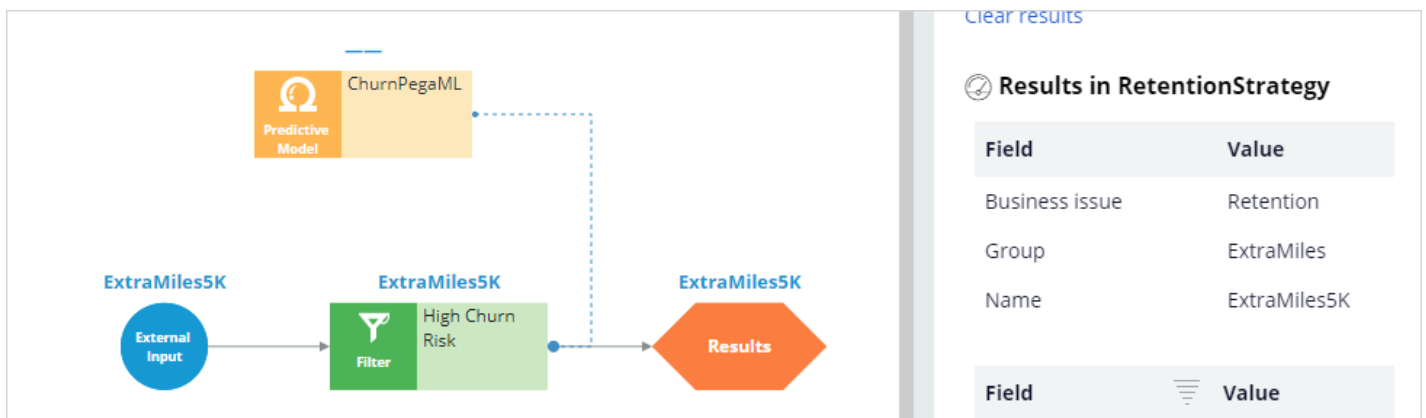
Save the strategy.



Now, test the strategy using the customer profiles, **Troy** and **Barbara**.

For external inputs, consider all available retention offers.

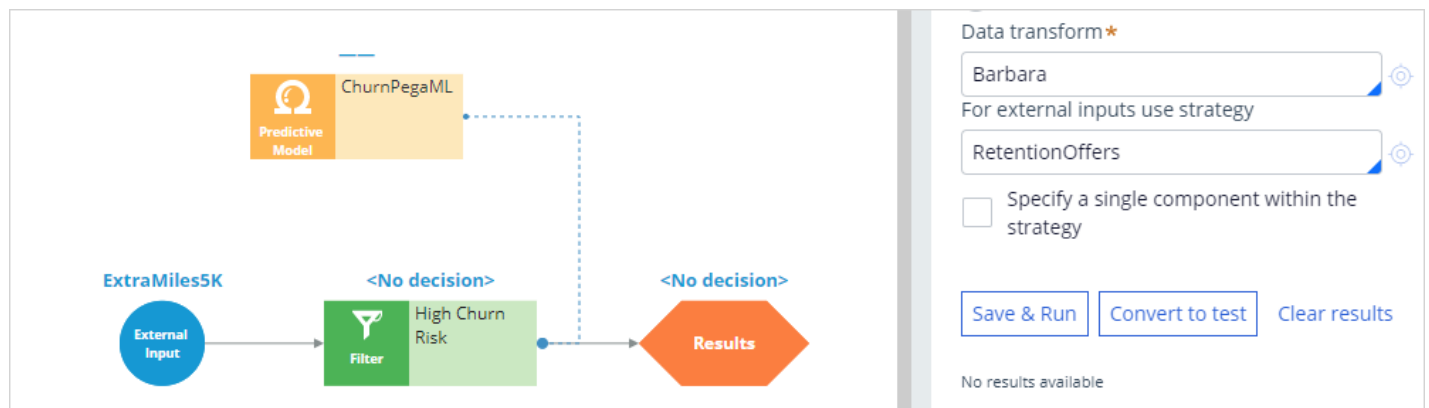
The strategy outputs a result for **Troy** because the result of the predictive model is "churn".



Field	Value
Segment	churn
ActionContext	---
ApplyAnalytics	---
Benefits	---
Bundle Parent	---

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

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



Field	Value
Segment	loyal
ActionContext	---
ApplyAnalytics	---
Benefits	---
Bundle Parent	---

The strategy is not available to the U+ website. By checking it in, you are committing your changes, so they will be put into effect.

You can now use this strategy in the Next-Best-Action Designer engagement policy as an applicability condition for the **Loyalty offers** and **CreditCards** groups.

The first business rule you need to implement is: the **Loyalty offers** group is applicable only for high risk customers. To implement this rule, in the Applicability section, define a condition for the customer field.

Select Strategy and then select RetentionStrategy. The condition is: RetentionStrategy has results for the High Churn Risk.

Retention	<b>E Eligibility</b> ?
Loyalty offers <b>EDITING</b>	<b>A Applicability</b> ?
Sales	Customer ▼ RetentionStrategy ▼ has results for ▼ High Churn Risk ▼
CreditCards	

The second business rule you need to implement is: U+ Bank wants to show credit card offers only to customers who remain loyal for now; meaning the **CreditCards** group is not applicable for high risk customers.

To implement this rule, modify the Applicability section of the CreditCards group.

Select the **RetentionStrategy**. The condition is: RetentionStrategy doesn't have results for the High Churn Risk.


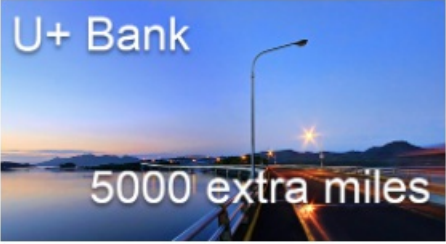
Retention	<b>E Eligibility</b> ?
Loyalty offers	Customer ▼ isCustomer ▼ is true ▼
Sales	and ▼
CreditCards <b>EDITING</b>	Customer ▼ Age ▼ is greater than ▼ 18
	<b>A Applicability</b> ?
	Customer ▼ Has Cards ▼ is equal to ▼ N
	and ▼
	Customer ▼ RetentionStrategy ▼ doesn't have results for ▼ High Churn Risk ▼

Save the configurations.

Once the applicability conditions are defined, you need to amend the channels configuration. Since U+ Bank introduced a new group, **Loyalty offers**, 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 this completes the required configurations.

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

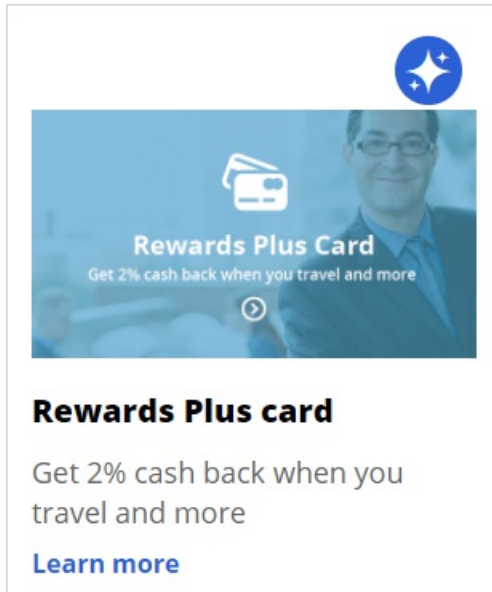



**Extra miles 5K**

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.



This demo has concluded. What did it show you?

- How to use a predictive model 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.

Using predictive models in engagement strategies -- Mon, 10/12/2020 - 03:24  
To get the full experience of this content, please visit <https://academy.pega.com>

# Contact policies

## Understanding contact policy requirements

Too many contact attempts over a short period of time can have a negative impact on a customer's attitude toward further actions by your company. To maximize the lifetime value of every customer relationship, organizations must prevent outreach fatigue by optimizing the number of actions taken.

In the Pega Customer Decision Hub, contact policies allow you to suppress actions after a specific number of outcomes.

Suppressing or pausing an action prevents oversaturation by limiting the number of times a customer is exposed to the same action.

## Defining contact policies

Contact policies determine when and for how long an Action or group of Actions should be shown to a customer. Contact policies track responses to Actions over a specific period of time, allowing you to implement rules such as the following:

- Do not show an ad to a customer for two weeks if the customer ignores the ad five times in a one-week timeframe.

If outcomes are tracked for an individual action, then the action is not shown once the suppression criteria are met.

- Do not show a group of ads for six months if a customer clicks on any ad in the group 3 times over a period of 30 days.

If outcomes are tracked for all actions in the group, then all of these actions are not shown once the suppression criteria are met.

An Interaction History Summary rule is used to determine the number of impressions and clicks generated by a customer over a period of time. The default time periods are 7 and 30 days. There might be business requirements to track a customer's response to an offer over different time periods, for example, 14 days.

You can add more tracking periods by creating a new Interaction History Summary rule for the required time period and then updating the part of the Next-Best-Action strategy that references it.

Contact policies -- Thu, 10/15/2020 - 07:27

To get the full experience of this content, please visit <https://academy.pega.com>



# Adding more tracking time periods for contact policies

## Introduction

Suppression rules determine when and for how long an action or group of actions can be shown to a customer. These rules put an action on hold after a specific number of outcomes are recorded for some or all channels. Learn how to use suppression rules to track customer behavior for time periods that you define.

## Video




A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

## Transcript

This demo illustrates how to add more time periods to contact policies to allow customer behavior tracking for periods other than the default 7 or 30 days.

U+, a retail bank, is using the Pega Customer Decision Hub™ to display marketing offers to customers on its website.

Currently, U+ Bank displays multiple credit card offers to each customer who logs in to the website. For example, if customer Troy logs in to his accounts page, the Standard card and Rewards card offers are displayed based on eligibility criteria defined by the business. If Troy clicks on the banner and then returns to his accounts page, the same set of offers is still displayed.




CHECKING & SAVINGS

CREDIT CARDS


LOANS

INVESTMENT




# Account overview


## Accounts



Savings  
\*\*\*1234




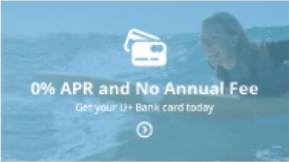

Checking  
\*\*\*5678



Credit  
\*\*\*7890

## Credit card balance

Current balance	Due date	CardProtect
\$164.80	Feb 15, 2020	 On
<a href="#">Pay now</a>	<a href="#">View statement</a>	<a href="#">Suspend CardProtect</a>



## Standard card

0% APR and no annual fee

[Learn more](#)

## Quick links

The bank does not want to show the same offer to customers who have clicked on it three times in the last 14 days.

This is the Pega Customer Decision Hub portal. Navigate to Next-Best-Action Designer to create a new contact policy to track customer responses for a period of 14 days. Notice that by default, responses are tracked for a period of 7 or 30 days.

The values in this drop-down come from an artifact called Field Values in Pega. These are a set of valid values for a field displayed in the application. In this case, these values represent the existing IH Summary rules.

You must now add an additional tracking period of 14 days based on your business requirement.

For this, you need to create a new Interaction History Summary rule that determines a customer's responses over a 14-day period. The easiest way to do this is by creating a copy of an existing rule, such as "Action Outcomes for the past 7 Days". Modify the rule to aggregate the customer responses over 14 days.

←

Data Set : Action Outcomes for the past 14 Days

Actions ▾

Run

Save

Aggregates

Group by

Time Period

Last ▾

14

Days ▾

from source data set

☐ Start aggregating as of

Dimension

For each Subject ID, Subject Type, and for each

Action

Channel

☐ BundleHead
☐ channel group

☐ BundleName
☐ channel sub group

☒ Group
☐ DeviceType

Details

Name

Action Outcomes for the past 14 Days

ID

ActionOutcomesForThePast14Days

Status

Available ▾

Class

Data-pxStrategyResult

Ruleset

PegaCRM-Artifacts:01-01-01

History

Then, create a copy of an existing field value. This will represent the newly created IH Summary rule in a selection list that will be presented to the user when they create the new contact policy.

Field Value: ActionOutcomesForThePast14Days [Available]

CL PegaMKT-Data-NBA-SR ▾

ID IHSummaryName • ActionOutcomesForThePast14Days

RS PegaCRM-Artifacts:01-01-01

🔒

Save as ▾

Delete

Actions ▾

Check out

Localized label

Specifications

History

Translate from

ActionOutcomesForThePast14Days

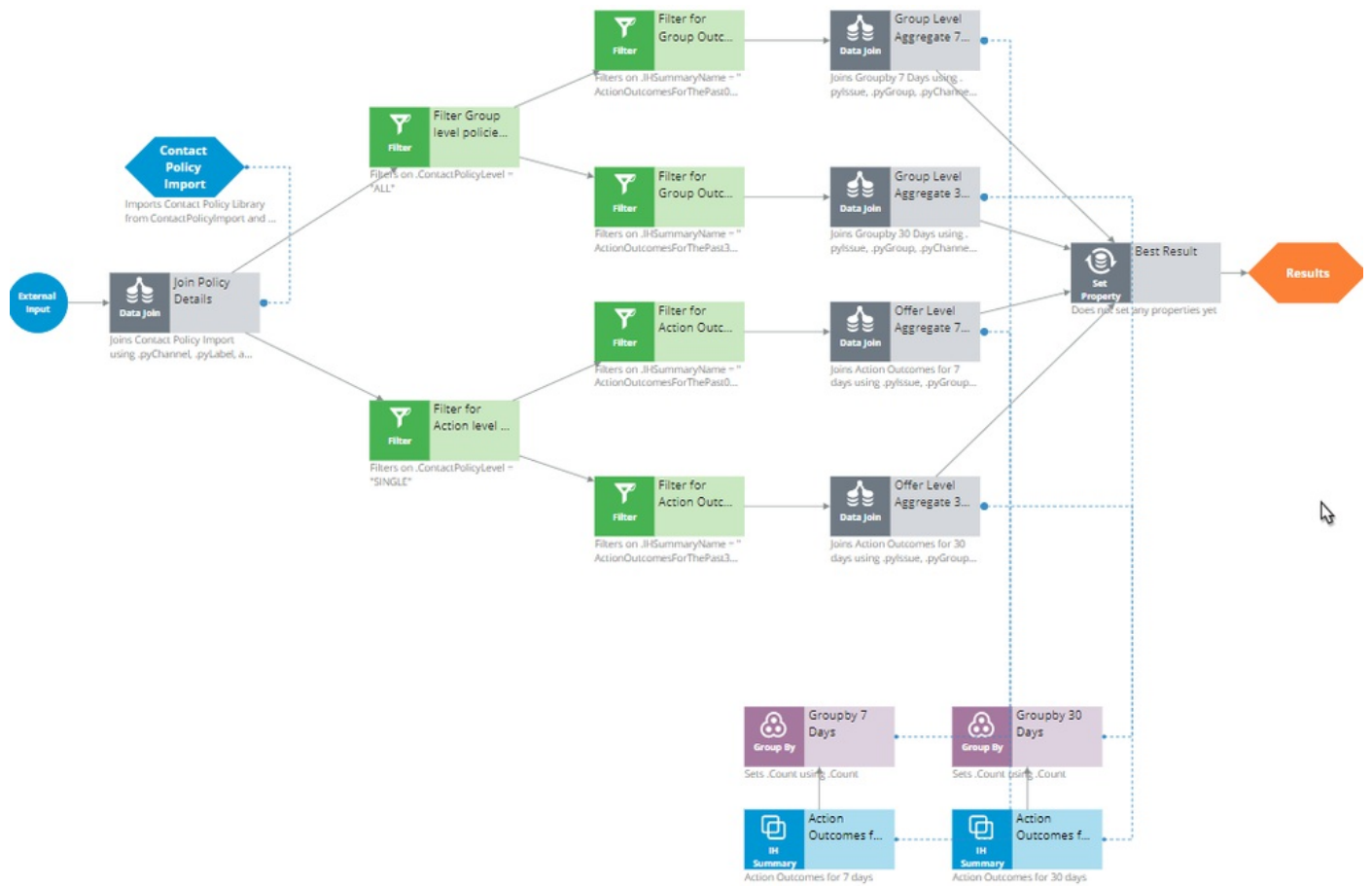
To

14

Extra whitespaces will be collapsed to single space by browser

Field Value Parameters

The CheckSpecificChannelLimits decision strategy is used to implement the contact policies. This strategy takes into account the number of responses from customers and suppresses Actions based on the contact policy configuration. It also checks the contact policy threshold for specific channels, such as web, email, or SMS. It is a sub-strategy of the BehavioralLimits strategy.



Now, let's see how to extend the strategy to incorporate the new tracking period. First, add an Interaction History Summary component to retrieve the customer's interactions with the company over the past 14 days. This IH Summary component must include the IH Summary rule to track the customer's responses over the 14-day period.

## Interaction history summary properties



Name \*

Action Outcomes for 14 days

Component ID ActionOutcomesfor14days

Description



Use generated



Use custom

Action Outcomes for 14

### Select Aggregate Dataset

ActionOutcomesForTheF 

Group by properties: Subject ID, Subject Type, Group, Business issue, Name, Channel, Direction and Outcome

#### Aggregated fields

.Count

is Count

Cancel

Submit

Then, add a Group By component to calculate the sum of customer responses for a particular Issue/Group as received from the IH Summary component.

## Group by properties



Name \*

Groupby 14 Days

Component ID Groupby14Days

Description



Use generated



Use custom

Source components

Properties

### Group output rows by

+ Add item

X Delete

.pyIssue



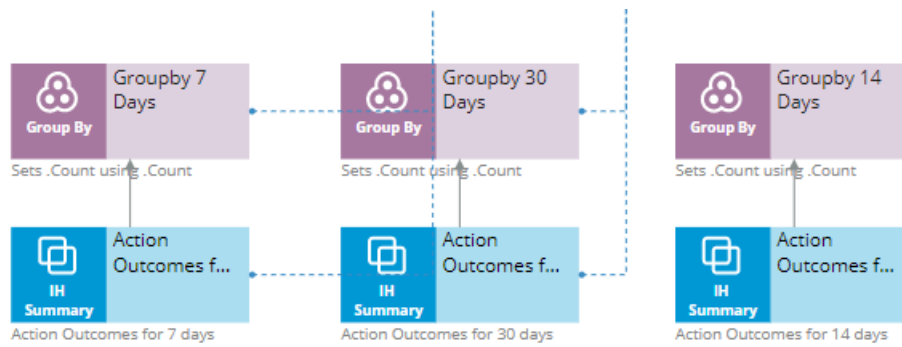
and by

.pyGroup



Cancel

Submit



Next, add a Data Join component to join the data received from the IH Summary component to the strategy. That is, the data from the IH Summary component is referred to and used in the strategy via the Data Join component.

**Data join properties**✕

Name★Offer Level Aggregate 14 Days

Component IDOfferLevelAggregate14Days

Description

☒ Use generated

☐ Use custom

Source components

Join

Properties mapping

**Join source components with**

TypeComponent ▾

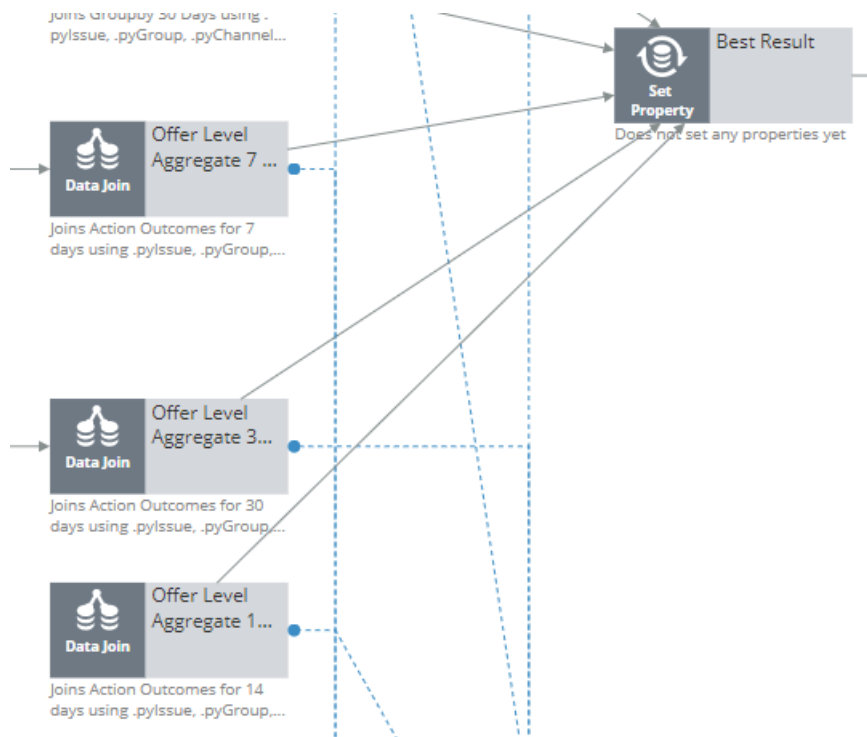
NameAction Outcomes for 14 days ▾

ClassCRM-SR

☐ Exclude model results from Action Outcomes for 10 days 1 ?

Cancel

Submit



Then, add a Group-level Data Join component to join the data received from the Group By component to the strategy. That is, the data from the Group By component is referred to and used in the strategy via the Data Join component.

## Data join properties

Name ★

Component ID GroupLevelAggregate14Days

Description ☒ Use generated ☐ Use custom

### Source components Join Properties mapping

#### Join source components with

Type  ▼

Name  ▼

Class CRM-SR

Then, add a Filter component to filter Action-level outcomes for a particular channel.

## Filter properties



Name Filter for Action Outcomes in past 14 d

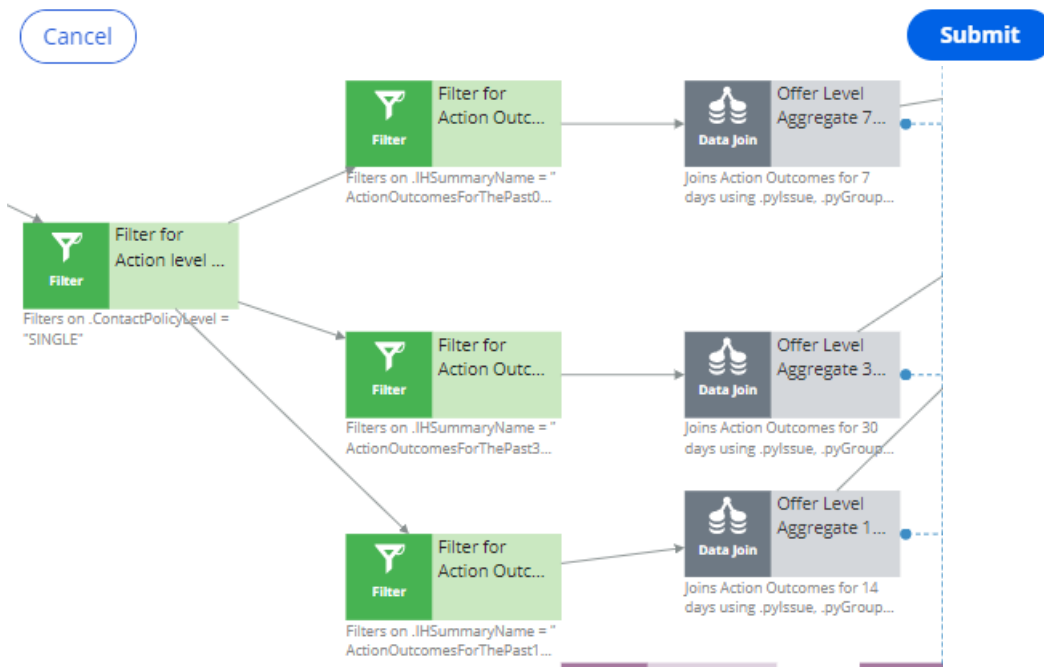
Component ID FilterforActionOutcomesinpast14days

Description ☒ Use generated ☐ Use custom

### Source components **Filter**

Type ☒ Filter condition ☐ Proposition filter

Filter condition .IHSummaryName = "Act"



Finally, add a Filter component to filter Group-level outcomes for all channels.



Filter properties

Name★

Filter for Group Outcomes in past 14 days

Component ID

FilterforGroupOutcomesinpast14days

Description

☒ Use generated

☐ Use custom

Source components

Filter

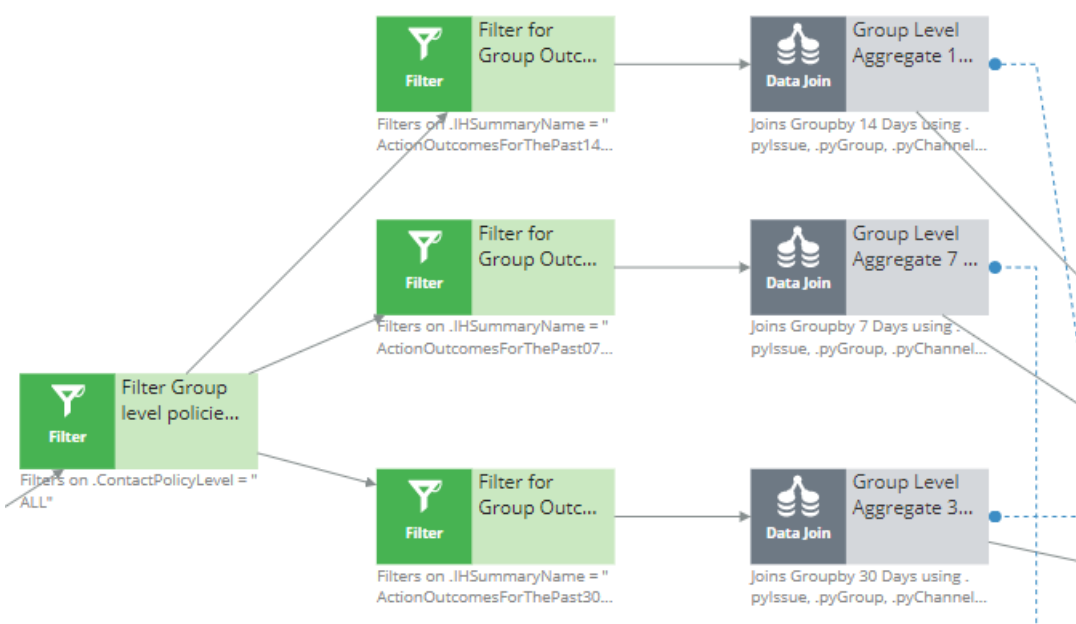
Type

☒ Filter condition

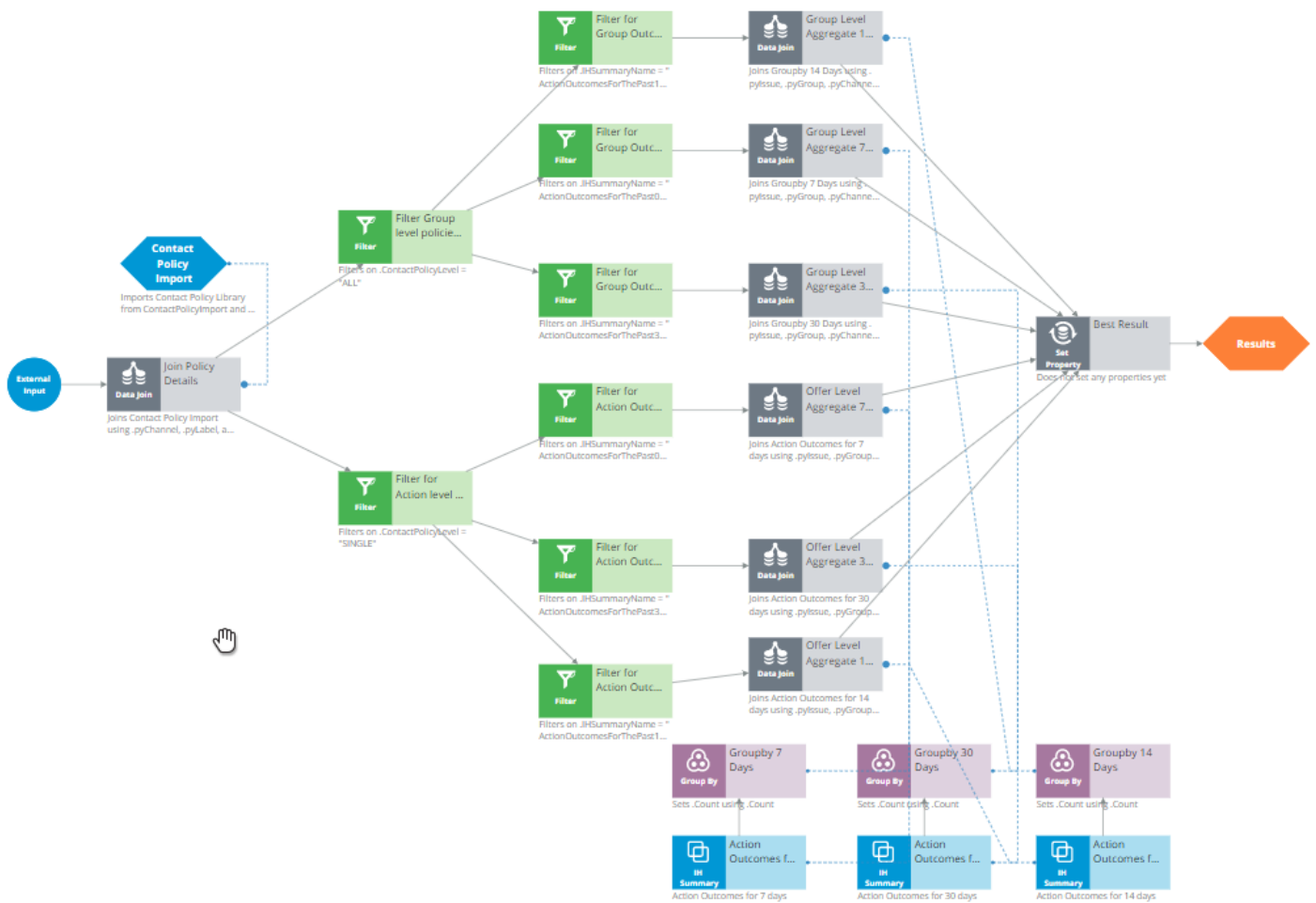
☐ Proposition filter

Filter condition

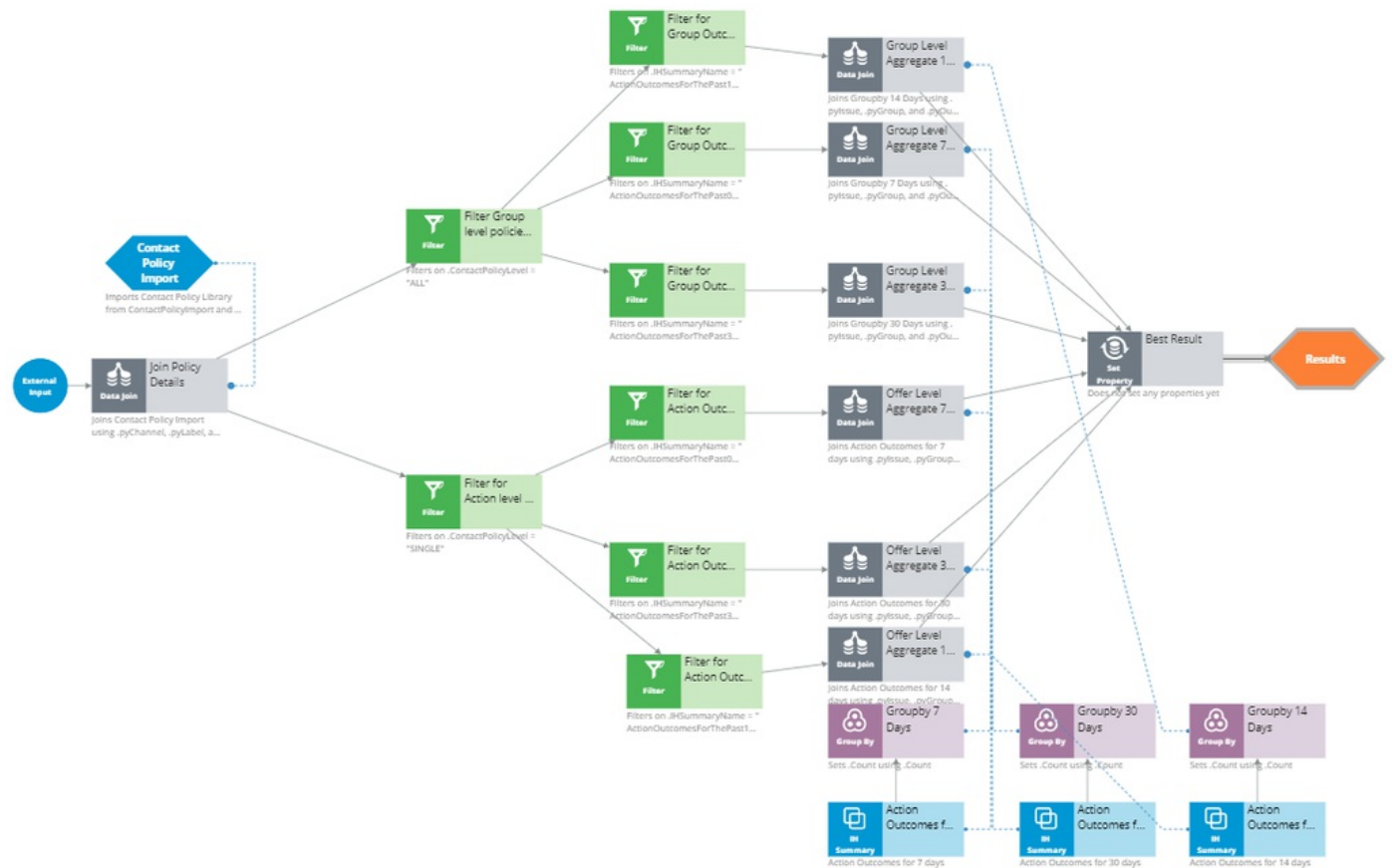
.IHSummaryName = "ActionOutcomesForThePast14Days"



The extended strategy must look like this.



Now you must extend the **CheckAllChannelLimits** strategy in the exact same way. This strategy checks the contact policy threshold for all channels. It is a sub-strategy of the **BehavioralLimits** strategy.



Now, navigate to Next-Best-Action Designer to verify that the new tracking time period for the contact policy is available.

Add contact policy

×

Name \*

Scope

Track

Accepts

▼

for

all actions in the group

▼

within the past

14

▼

days

Cancel

7

14

30

t

⌵

This demo has concluded. What did it show you?

- How to use an IH Summary rule to track time periods in a contact policy.
- How to extend the strategy for a specific channel to add a new tracking time period.
- How to extend the strategy for all channels to add a new tracking time period.
- How to use the new tracking time period in Next-Best-Action Designer.

Adding more tracking time periods for contact policies -- Fri, 07/24/2020 - 06:26  
To get the full experience of this content, please visit <https://academy.pega.com>

# Simulation testing

## Simulation types

By running simulation tests in Pega Customer Decision Hub™, you can understand the effect of business changes on your next-best-action strategy framework. The Pega Customer Decision Hub portal offers a large variety of simulations. The simulation capability ranges from simulations that help identify under-served customers to simulation tests that allow you to investigate how an introduction of a new engagement policy might affect actions offered across a segment of customers.

The different types of simulation testing available in Pega Customer Decision Hub are: Value Finder, Audience Simulation, Distribution Test, Ethical Bias, and Scenario Planner.

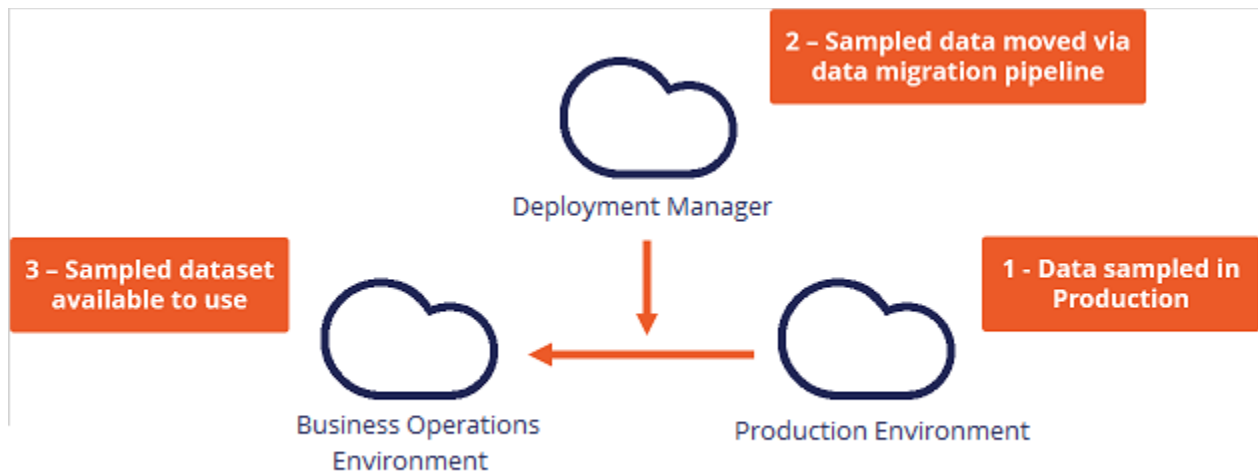
Click on the hotspots to learn more about each of the simulation types.



1. Value Finder allows clients to engage more empathetically by identifying and profiling under-served customers, then suggesting actions for improvement.
2. Audience Simulation tests the configuration of engagement policies against a segment of the audience.
3. Distribution Test enables you to unit test a decision strategy.
4. Ethical Bias tests the engagement policies for unwanted bias.
5. Scenario Planner enables users to easily simulate “what-if” scenarios so they can more accurately forecast results, optimize strategies to hit specific goals, and explore the potential tradeoffs of each option.

## Business-as-usual

Simulations are run in a Business Operations Environment (BOE) that is specifically designed to build, simulate and optimize changes. A sample dataset, which includes interaction history and adaptive models from the production environment, is created via a pipeline into the BOE. This dataset is used as the basis for the simulations.



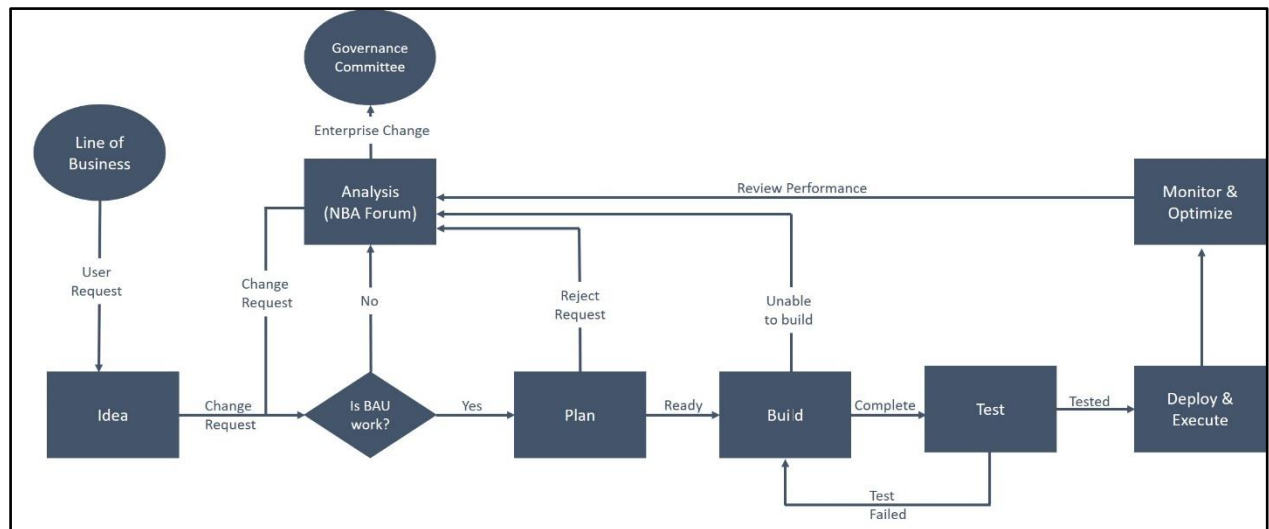
Let's see how each of these simulations can be used in a business-as-usual use case.

U+, a retail bank, has recently implemented a project in which credit card offers are presented to qualified customers when they log in to the web self-service portal. Now U+ would like to leverage the simulation testing capabilities of Pega Customer Decision Hub to:

- Market credit card offers to qualified customers
- Verify if their engagement policy conditions are presenting offers to customers as expected
- Check the offer distribution and prioritization ranges
- Check if business policies and regulations are being violated
- Show improvements in projected value

Each of these simulation types plays a specific role in a business operating model. This is U+ bank's operating model. An operating model supports businesses in the planning, development, testing, monitoring, and optimization of changes. The simulation capability can be utilized throughout the lifecycle but has greater significance when run at certain points.

Click the hotspots to see in which phase of the lifecycle U+ bank can use each of these simulation types.



1. In the ideation and analysis phase, Scenario Planner and Value Finder help the business come up with new ideas and engagement policies to improve offer marketing.
2. During the build phase, Audience Simulation provides detailed information about the pass rates of the engagement policies. It also helps you understand if you need to tighten or loosen any criteria during the build stage. In this way, audience simulation is similar to unit testing.
3. During the testing phase, the Distribution Test, Ethical Bias, and Scenario Planner simulations help you in performing system or acceptance testing. These simulations give you an idea of the distribution of offers across a sample customer base, help identify bias, and look for what is in target and what is not.
4. After deploying, you can use Scenario Planner and Value finder to look for more ideas and opportunities.

# Pega Value Finder

## Pega Value Finder

A value finder simulation allows you to engage more empathetically by identifying and profiling “under-served” customers, then suggesting actions for improvement – like adjusting engagement policies or creating new actions and treatments.

Using Pega Value Finder, you can discover areas in which you can improve the next-best-action strategy by monitoring scenarios in which customers are presented with no actions or only low-propensity actions. This is particularly useful when planning new changes or optimizing existing parameters.

Value Finder identifies and profiles “under-served” customers. These are customers that either do not receive actions, or only receive actions they have a low propensity to accept. It analyzes what happens at every stage of the next-best-action decision funnel, enabling you to:

- View offer distribution to well-engaged, under-engaged, or not-engaged customers
- Identify top opportunities for improvement
- Review details of under-served groups at each level of arbitration

Consider an example in which a bank runs a value finder simulation and identifies groups of not-engaged and under-engaged customers. The details of the groups and how to address the findings are as follows:

Not-engaged customers: There are 7000 under-served customers, none of whom own a credit card. They all have credit scores over 650 but are blocked because the eligibility rules in place keep them from seeing specific offers they have a high propensity for. In this case, you might want to tweak the engagement policy to present them with more appropriate offers.

Under-engaged customers: There are 5000 customers that have good credit scores and own a credit card but have no propensity scores higher than 5%. You might need to create a new offer with different terms and test it to see if you can capture their attention.

## Analyzing customer distribution using a value finder simulation



A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

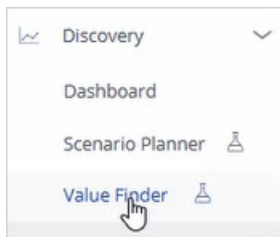
### Transcript

This demo will show you how to analyze customer distribution using a value finder simulation.

U+, a retail bank, has recently implemented a project in which credit card offers are presented to qualified customers when they log in to the self-service web portal. The bank would like to check if there are any un-served or under-served customers and find ways to serve them better.

This is the Pega Customer Decision Hub™ portal.

In **Discovery**, you can create a new value finder simulation.



To create a value finder simulation run, select the issue and group in which you would like to find opportunities. Then select an audience on which you would like to do the simulation run. The audience is a list of potential target customers. You can modify the simulation name as required to easily identify the specific runs.

### Create simulation

Find opportunities for

Sales / CreditCards

Audience \*

SampledCustomers

Simulation ID prefix

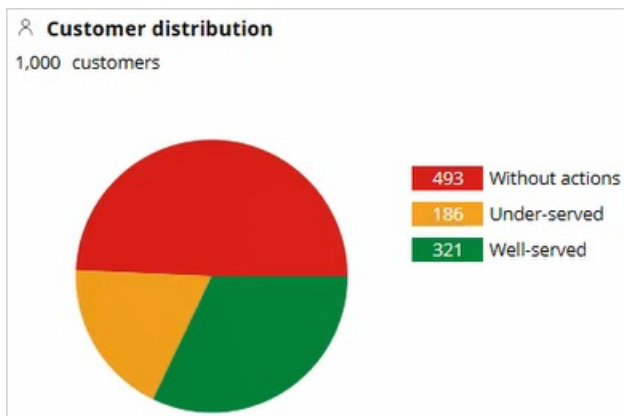
ValueFinder

Population

999

Once the simulation run is complete, Value Finder indicates several opportunities.

The pie chart displays the cumulative numbers of customers without actions, under-served customers, and well-served customers. The **Without actions** category indicates the number of customers who received no actions. The **Under-served** category indicates the number of customers who received low propensity actions. The **Well-served** category indicates the customers who received high propensity actions.



Value Finder identifies a number of opportunities for improving the Next-Best-Action Strategy. However, it displays the top three opportunities only.

The first opportunity represents a group of customers that are under-served due to eligibility conditions. The second opportunity indicates the stage of your Next-Best-Action strategy at which customers are prevented from receiving actions. In this example, it is after suitability. The third opportunity shows the number of customers who are under-served due to low propensity actions.

In this case, notice that there are 217 customers who are under-served because they have no relevant actions or treatments, and 327 customers who have no actions because the suitability condition may be too strict.



🔍 <b>Top opportunities</b>		
Under-served customers after eligibility	UNDER-SERVED	217
Create more relevant actions or treatments for the identified customer groups.		
Customers with out actions because of suitability	WITHOUT ACTIONS	327
Review your suitability rules, as they might be too strict.		
Under-served customers because of arbitration	UNDER-SERVED	1
Review the arbitration, as it might be non-empathetic for your customers.		

Value Finder identifies customers as under-served if the propensity of every action and treatment available to the customer is below the threshold propensity. To provide a convenient starting point, the propensity threshold value is initially chosen in such a way that 1 out of 20 customers is defined as under-served. In this case, this results in an under-served threshold of 6.3%.

Under-served threshold  
6.3%

If the business would like to present offers with a higher propensity, this threshold value can be changed.

Configure under-served threshold

Set the maximum propensity that determines under-served customers.  
A customer is considered as under-served if they only receive actions with a propensity below the threshold.

Propensity threshold

6.3%

At this propensity threshold, approximately 5% of the customers will be defined as under-served.

Cancel

Submit

Details are provided for 'Under-served groups after eligibility' to show which eligibility conditions are contributing to the issue.

Value Finder provides the following information about each group:

Description: Group characteristics such as Age or LifeCycle Period.

Under-served customers: The number of under-served customers in the group.

Accuracy: The number of under-served customers in the group divided by the total number of customers in the group. If the accuracy is 100%, this means that all customers in the group are under-served. If the accuracy is lower, for example, 91.4%, this means that 91.4% of the customers in the group are under-served. The remaining 8.6% have at least one action above the propensity threshold.

🔍 <b>Customer group A</b>		
Description	Under-served customers	Accuracy
AverageSpent is less than or equal to 1.4k ⓘ	85	91.4%
LifeCyclePeriod is not Onboard		
Age is greater than 60		
CreditScore is greater than 80.1		

You can also manage the group by removing a field from the description. Typically, the system lists all potential fields available for the Next-Best-Action Strategy. You can decide to remove a field from the list if it's not a required field. For example, if the business does not want to categorize customers based on age to avoid discrimination, they can remove the

Age field from the list. If a field is removed, Value Finder then recalculates the values for **Under-served customers** and **Accuracy**.

Manage descriptive fields

Manage the fields that are used to identify and describe under-engaged customer groups.

Used fields	
Age	X
AverageBalance	X
AverageSpent	X
CreditScore	X
Customer Lifetime Value	X
LifeCyclePeriod	X
Net Promoter Score	X

Ignored fields
No fields are ignored

Also, you can save or export the customer groups identified as under-served as audiences.

Accuracy

- Save as audience
- Export as CSV
- View details

You can then run distribution tests to get more insight into the current actions that these audiences receive. Use the Value Finder recommendations and your distribution test results as feedback for business stakeholders to inspire them to create new actions and treatments that will be relevant to these customers.

Let's now look at the **WITHOUT ACTIONS** customer category to understand how the customers are filtered. The filtration that happens in this simulation is similar to a funnel filtration for every engagement policy condition type. In this case, the total number of customers in the audience is 1000. Thus, the input population considered for the eligibility conditions is 1000. When the eligibility conditions are applied, 11 customers do not receive any actions. Thus,  $1000 - 11 = 989$  customers who pass through the eligibility level. The output population of the eligibility level is passed on as the input to the next level.

The input population for the applicability conditions is therefore 989. When the applicability condition is applied, 155 customers do not receive an action. Thus,  $989 - 155 = 834$  customers who pass through the applicability level.

The input population for the suitability conditions is therefore 834. When the suitability condition is applied, 327 customers do not receive an action.

WITHOUT ACTIONS	
Eligibility	11
Applicability	155
Suitability	327

In this specific scenario, 327 customers are presented with no actions due to the suitability condition. When you click on **Suitability** under the **WITHOUT ACTIONS** customer category, you can view the suggested recommendation for providing more customers with actions. Use this information from the Value Finder to run a funnel filtration on an audience simulation to detect at which level the suitability condition is preventing customers from receiving an action. That is, whether it is the suitability condition at the group level or the action level.

This demo has concluded. What did it show you?

- How to configure and run a value finder simulation.
- What are the top three opportunities identified by the Value Finder.
- How to interpret the customer group in the under-served customer category.
- How to interpret the without actions customer category.
- How to interpret the under-served customer category.

Pega Value Finder -- Wed, 10/14/2020 - 05:34

To get the full experience of this content, please visit <https://academy.pega.com>

# Audience simulation

## Introduction

You can improve the performance of your next-best-action strategy by testing the configuration of engagement policies against a set of customers. In this way, you can check how many potential actions are filtered out by each component of the policy and discover if a particular criterion is too broad or too narrow for your requirements.

## Video



A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

## Transcript

This video explains what an audience simulation is and how it improves the performance of your Next-Best-Action strategy.

Audience simulation tests the configuration of engagement policies against a set of customers.

Consider an example in which an audience comprises 1071 customers. Let's see how the audience will be filtered based on the engagement policies.

When the eligibility conditions are applied, only 350 out of 1071 pass through. These 350 customers become the input for the next stage, applicability conditions. Only 99 of the 350 pass through the second stage. In the final stage, suitability conditions, only 65 pass through. Of the original 350, only 65 customers qualify to receive at least one offer.

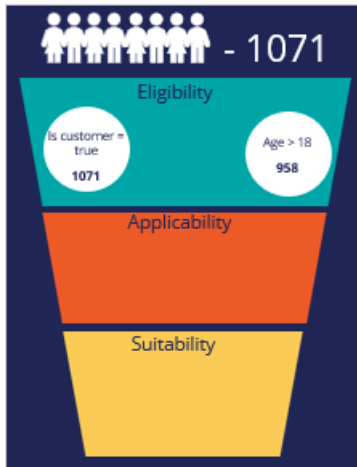


The filtering that happens in this simulation is similar to a funnel filtration for every engagement policy condition type. However, within an engagement policy condition type, each of the criterion is applied separately to the corresponding input audience.

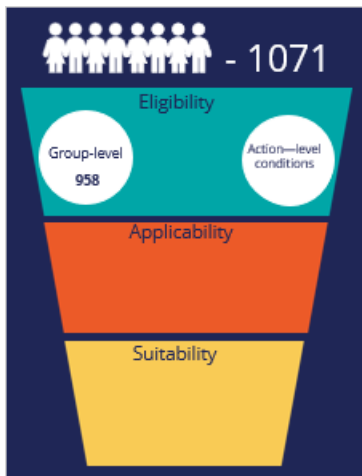
Let's now look at the same example in detail to see how the funnel filtration happens within each engagement policy type.

The engagement policies can be defined for a specific group within an issue and/or for individual actions. Group-level conditions are applicable to all the actions within that group while the action-level conditions are applied to that specific action only.

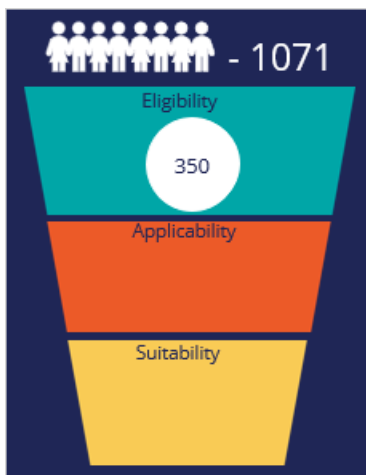
In this case, the total number of the customers in the audience is 1071. Within the eligibility criteria, the first condition does not filter any customers, as all qualify. When the second condition is applied, 958 customers pass through.



Since both conditions need to be met, the intersection of these two conditions, 958, is the final number of customers who pass through at the group level. Now the system checks for individual action-level eligibility conditions that also need to be met.

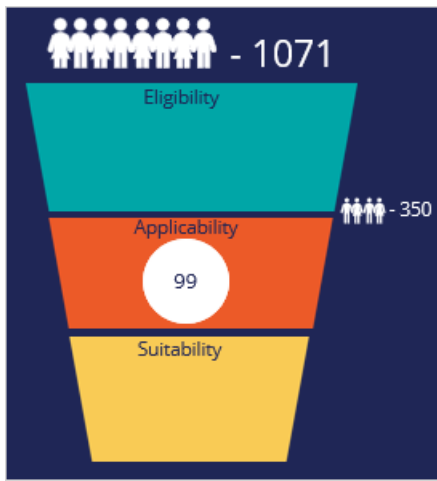


The intersection of the group-level and action-level conditions form the final eligibility-level output population of 350.



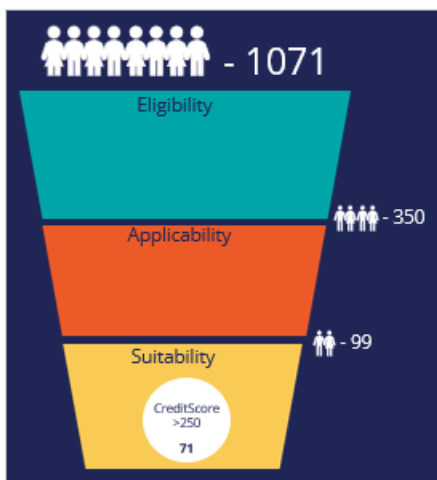
The output population of the eligibility level is passed on as the input for the next level.

The input population that will be filtered by the applicability conditions is therefore 350. When the applicability condition is applied, 99 of the 350 customers qualify for at least one offer. In this case, let's assume there are no action-level conditions. Thus, the result from the individual group-level condition is the same as the overall applicability level, 99.

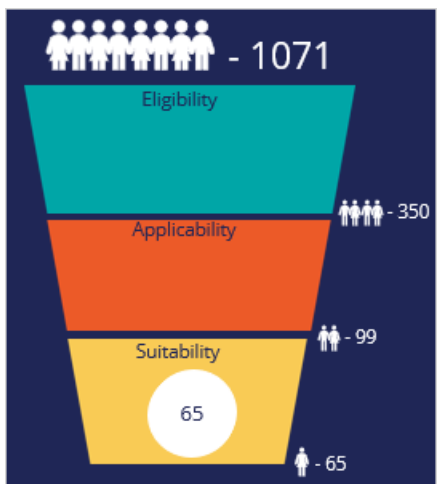


The output population of the applicability level, 99 customers, is passed on as the input to the suitability level.

When the suitability group-level condition is applied, 71 of the 99 customers qualify for the offers.



Now the system checks for individual action-level suitability conditions, and the intersection of both group-level and action-level conditions forms the final suitability-level output population, which is 65.



In summary, with the audience simulation test you can check how many potential actions are filtered out by each component of the engagement policy and discover if a particular criterion is too broad or too narrow for your requirements.

Audience simulation -- Wed, 10/14/2020 - 05:39

To get the full experience of this content, please visit <https://academy.pega.com>

# Running an audience simulation test

You can improve the performance of your next-best-action strategy by testing the engagement policy configuration against a set of customers. In this way, you can check how many potential actions are filtered out by each component of the policy and discover how a criterion might affect your requirements.

## Audience simulation

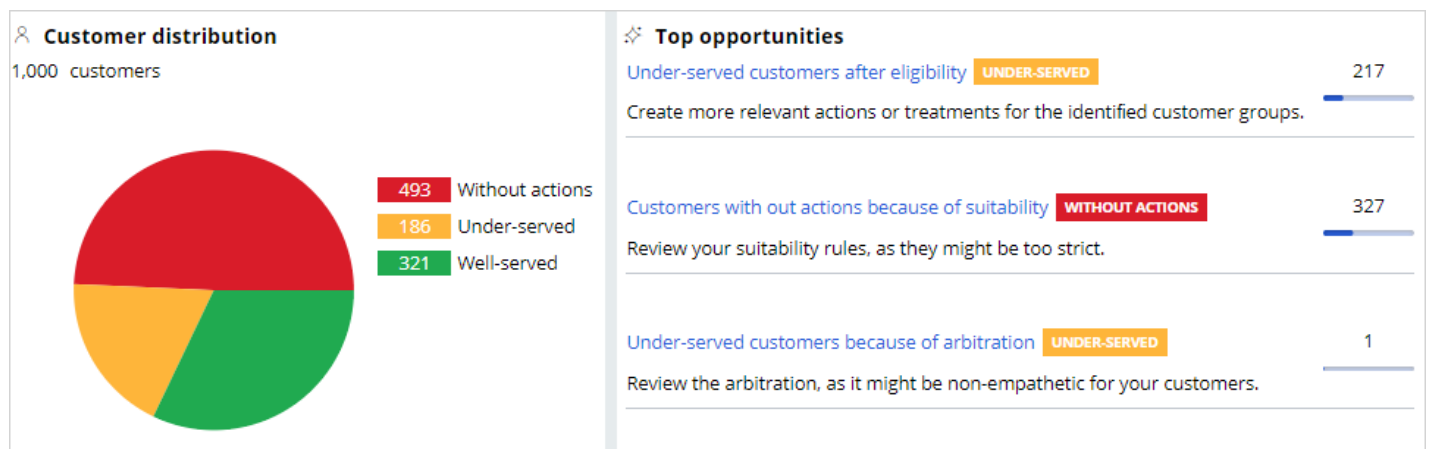


A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

### Transcript

This demo will show you how to run an audience simulation test. It will also explain how many potential actions are filtered out by each section of the engagement policy and reveal if a particular criterion is too broad or too narrow for your requirements.

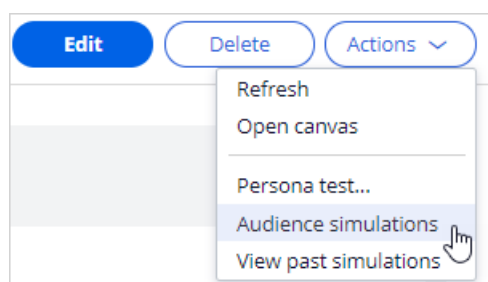
U+, a retail bank, has recently implemented a project in which credit card offers are presented to qualified customers when they log in to the web self-service portal. The bank ran a value finder simulation to find under-engaged customers. In that simulation run, a set of customers is identified with no actions after engagement policy conditions are applied.



The bank would now like to run an audience simulation to investigate why these customers have no actions presented to them, and correct the condition that causes the deviation.

This is the Pega Customer Decision Hub™ portal. First, take a look at the set of eligibility criteria that has already been configured by U+.

In **Audience simulations**, you can create a new simulation run.



Sales / CreditCards

Persona test Audience simulation

Simulation

Select... or Create simulation

To create an audience simulation run, select an audience on which you would like to do the simulation. The audience is a list of potential target customers. You can modify the simulation name as required to easily identify the specific runs. Then, choose the scope of the run. That is, you can choose to simulate only the engagement policies to validate the eligibility, applicability and suitability conditions. Or, you can simulate on both engagement policies and arbitration to understand how the conditions work when arbitration across all actions is also considered.

Create simulation

Audience \* Simulation ID prefix

SampledCustomers EngagementPolicyTest

Population 1000

Next-Best-Action scope ?

☒ Engagement policies only

☐ Engagement policies and arbitration

Cancel Run

Once the simulation run is complete, you can view the details of how the audience is filtered at the group level based on the configured engagement policy conditions.

Persona test Audience simulation

Simulation

EngagementPolicyTest-1 or Create simulation

Show population that passed as

Counts

Completed (Run on 9/2/20 2:47 AM by CDH Analyst) Processed: 1,000

For each component of the engagement policy, the simulation test shows a numerical or percentage value of the audience that will receive the action based on current criteria. For example, if the result of a criterion that checks if the action is active returns 100%, the component did not filter out any audience members.

The filtering process that happens in this simulation is similar to a funnel filtration for every engagement policy condition type. However, within an engagement policy condition type, each of the criterion is applied separately to the corresponding input audience.

In this case, the total number of customers in the audience is 1000. For this audience, the number of customers who received offers is 182. This simulation run also provides details of the audience filtration that happens with each engagement policy condition. When the eligibility condition is applied, 979 customers qualify for the offers. Within the eligibility criteria, the first condition does not filter any customers, as all qualify. When the second condition is applied, only 979 customers pass through. Thus, the intersection of these two conditions, 979, is the final number of customers who pass through to the eligibility level.

In this case, there are no eligibility conditions defined at the action level. If there are, the final result is the result of the



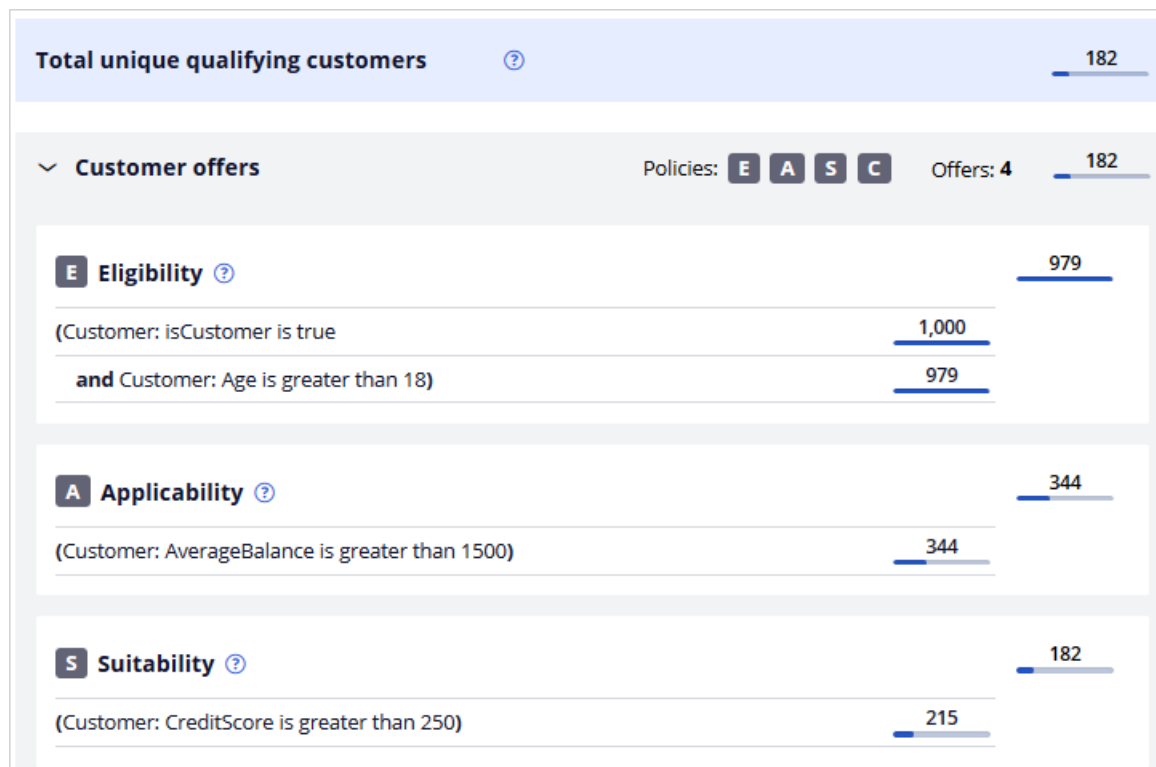
eligibility conditions at the group level, combined with the results at the action level.

The output population of the eligibility level is passed on as the input to the next level.

The input population considered for the applicability condition is therefore 979. When the applicability condition is applied, 344 customers qualify for at least one offer out of the 979 customers. Note, since there are no specific action-level applicability conditions, the result of the individual group-level condition is the same as that of the overall applicability component level, 344.

The output population of the applicability level, 344 customers, is passed on as the input to the suitability level.

When the Suitability group-level condition is applied, 215 customers qualify for the offers out of the 344 customers. If you check the engagement policies at the action level, notice that there are additional suitability conditions. When all of these conditions apply, the total number of customers who qualify for at least one offer is 182.



At the bottom, you can also view the number of customers who qualify for the offers. When only engagement policy conditions are considered in the simulation run, the number you see here is the number of customers who qualify for at least one action. Notice that the Premier Rewards card offer is rarely presented. Let's try to find out why this is so.

Open the Premier Rewards card to view the audience simulation filtering at the action level.

Offers		
4 Offers (4 with specialized policies)		
Name	Specialized policies	
Rewards card	E	57
Rewards Plus card	S	168
Premier Rewards card	S	8
Standard card	E	57

You can choose the same audience simulation to view the action-level filtration details and investigate which engagement policy condition is causing the current outcome.

**Offer: Premier Rewards card [Available]**  
Sales • CreditCards • PremierRewardsCard    PegaCRM-Artifacts:01-01-01

Save as    Delete    Actions

Details    **Engagement policy**    Treatments    Flow    Test    History

A Engagement policy

**Offer: Premier Rewards card [Available]**  
Sales • CreditCards • PremierRewardsCard    PegaCRM-Artifacts:01-01-01

Details    **Engagement policy**    Treatments    Flow    Test    History

**Audience simulation**  
Simulation  
Select... or Create simulation  
EngagementPolicyTest-1  
Eligibility

Open stored flow image  
Refresh  
Audience simulation...  
View past simulations

These numbers show how the engagement policies that are inherited from the CreditCards group are filtered. At the bottom, you can also view the final number of customers who qualify for the specific offers once the action-level engagement policy is applied.

Notice that at the suitability-level, only eight customers qualify for the Premier Rewards card offer.

**Suitability**
8

**Inherited from** CreditCards    215    ☒ Apply

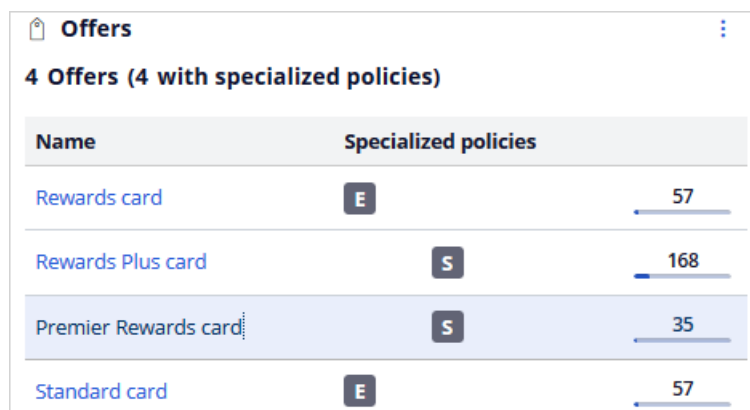
Customer offers  
(Customer: CreditScore is greater than 250)    215

**and**  
(Customer: CreditScore is greater than 825)    8

This is because the configured engagement policy criterion is too narrow. The results of this simulation show us that from the 215 customers who passed the eligibility and applicability conditions, all qualify for the group-level condition, but only eight customers qualify for the action-level condition. As the action-level suitability condition is too narrow, a very small number of customers are presented with the offer. The bank then decides to reduce the CreditScore from 825 to 750 to ensure the

Premier Rewards card offer is also presented to the customers.

Now, modify the credit score condition to reflect the correct value. Then, rerun the same audience simulation at the group level. Notice that now, 35 customers are presented with the Premier Rewards card offer.



Name	Specialized policies	
Rewards card	E	57
Rewards Plus card	S	168
Premier Rewards card	S	35
Standard card	E	57

Let's now run an audience simulation with engagement policies and arbitration in scope.

Note that the simulation results are different. This is because the simulation includes arbitration, adaptive analytics, treatment and channel processing, and constraints. That is, this simulation result shows the number of customers who will receive an action as their top action. Thus, the numbers at the group level and the action level tally.

That is, the sum of all customers (12+118+22+30) is 182.

This demo has concluded. What did it show you?

- How to configure an audience simulation.
- How to view the simulation filtering details at the group and action level.

## Decision funnel explanation simulation

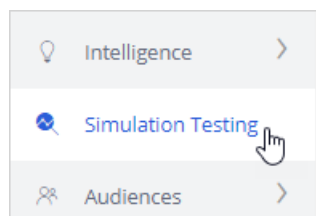


A transcript for this video is not available. Please visit <https://academy.pega.com> to watch.

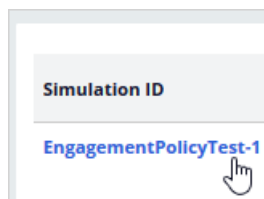
### Transcript

This demo will show you how a decision funnel explanation simulation is auto-created for every audience simulation run at the Next-Best-Action Designer level.

Every time an audience simulation is run, in the background, a decision funnel explanation simulation is created. Now, navigate to the **Simulations** landing page to view the auto-created funnel simulation.



This is the auto-created funnel simulation. Alternatively, you can also create a simulation run from here and show the simulation results in the Engagement policies->Audience Simulation.



To view the funnel simulation, open the simulation. When a funnel simulation is created, a set of reports is added to the run results.

Assigned reports		
Output	Report category	Report
ExplainDetails	Simulations	<a href="#">Top level view</a>
ExplainDetails	Simulations	<a href="#">Champion Challenger - Drill down view</a>
ExplainDetails	Simulations	<a href="#">Prioritization - Drill down view</a>
ExplainDetails	Simulations	<a href="#">Switch - Drill down view</a>
ExplainDetails	Simulations	<a href="#">Proposition filter - Drill down view</a>

Notice that the **NBA\_Sales\_CreditCards** strategy is used to run the simulation. This is because you selected 'Engagement only' in the audience simulation, without the arbitration.

Setup details

Run simulation on revision version

Strategy

NBA\_Sales\_CreditCards

Context

Data-Decision-Request-Customer

Results in

CRM-SR-Sales-CreditCards

Audience

Data set

Simulation ID

EngagementPolicyTest-1

Rule name

SampledCustomers

Purpose

Explain

The simulation reports provide detail about the filtering that occurred. You can download these reports offline to analyze the outcome.

Top level view									
Filtered by: ( Report Sub Type = toplevel or ( Report Sub Type = Overview and Rule Object Class = Rule-Decision-Strategy ) ) and Any Subject Type and Any Name									
Id	Group	Name	Strategy	Component	Type	#	#	P...	
SalesCreditCardsPremierRewardsCardNBA_AllIssues_AllGroups_CustomerResults					Output	10001,000	100.00		
SalesCreditCardsPremierRewardsCardNBA_Sales_CreditCards				Customer Actions	Filter	10001,000	100.00		
SalesCreditCardsPremierRewardsCardNBA_Sales_CreditCards				NBA_Sales_CreditCards_AllActions_PrimarySub Strategy		10001,000	100.00		
SalesCreditCardsPremierRewardsCardNBA_Sales_CreditCards_AllActions				NBA_Sales_CreditCards_A_All	Proposition Filter	10001,000	100.00		
SalesCreditCardsPremierRewardsCardNBA_Sales_CreditCards_AllActions				NBA_Sales_CreditCards_E_All	Proposition Filter	10001,000	100.00		

This demo has concluded. What did it show you?

- How to view the auto-generated decision funnel explanation simulation.

Running an audience simulation test -- Wed, 10/14/2020 - 05:39

To get the full experience of this content, please visit <https://academy.pega.com>