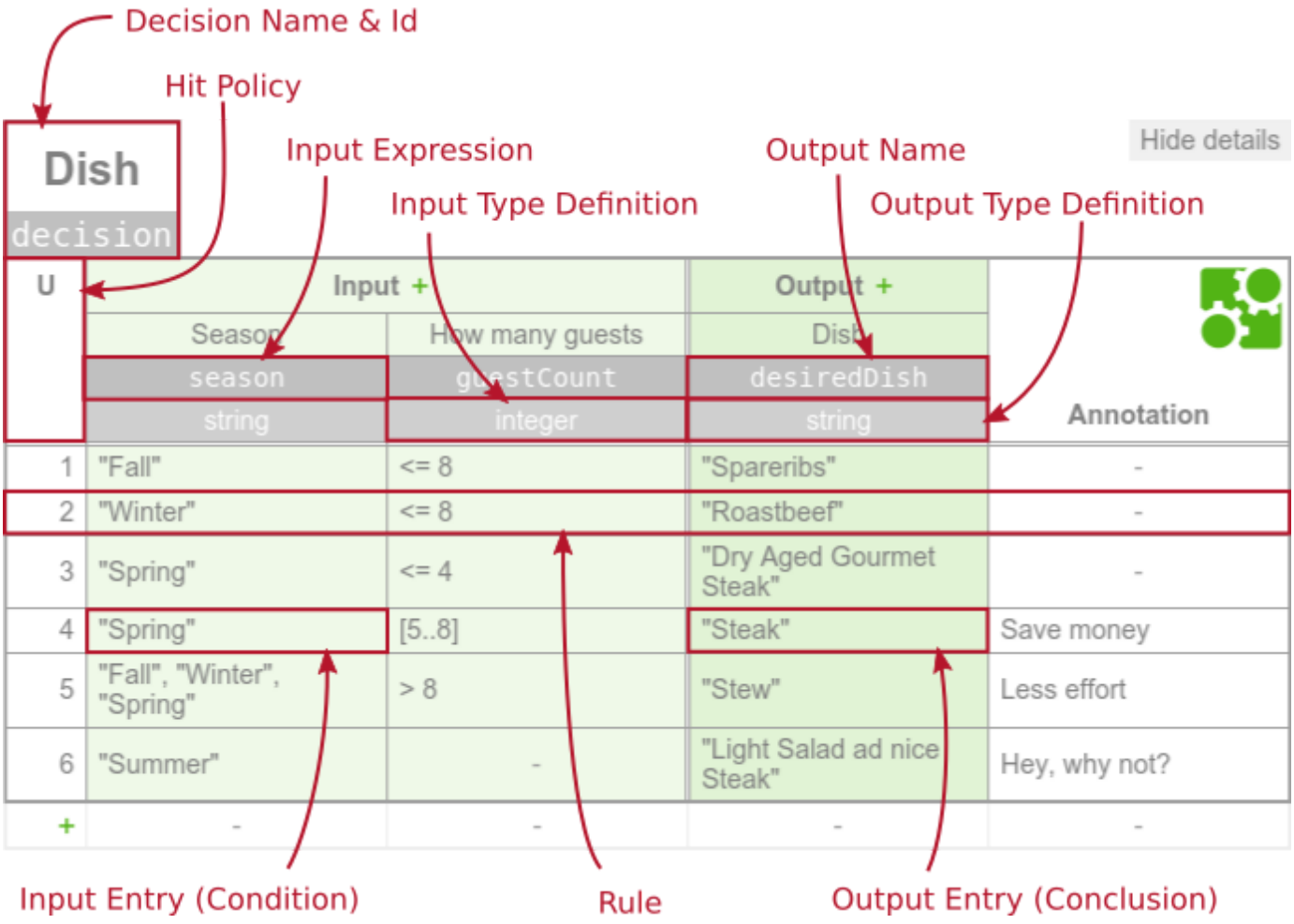
DMN Decision Table

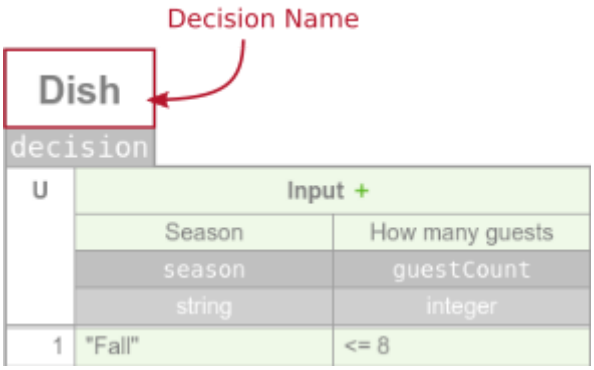


A decision table represents decision logic which can be depicted as a table in DMN 1.1. It consists of inputs (/manual/latest/reference/dmn11/decision-table/input/), outputs (/manual/latest/reference/dmn11/decision-table/output/) and rules (/manual/latest/reference/dmn11/decision-table/rule/).

A decision table is represented by a `decisionTable` element inside a `decision` XML element.

```
<definitions xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd" id="definitions" nam
  <decision id="dish" name="Dish">
    <decisionTable id="decisionTable">
    <!-- ... -->
    </decisionTable>
  </decision>
</definitions>
```

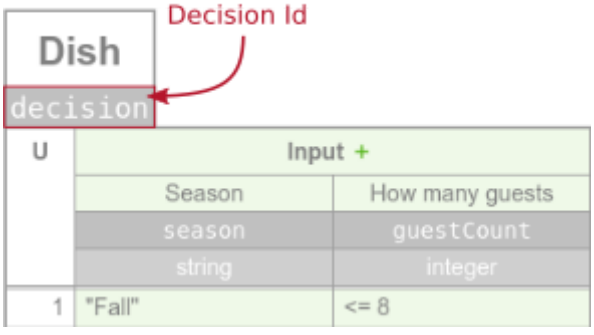
Decision Name



The name describes the decision for which the decision table provides the decision logic. It is set as the `name` attribute on the `decision` element.

```
<decision id="dish" name="Dish">
  <decisionTable id="decisionTable">
  <!-- ... -->
  </decisionTable>
</decision>
```

Decision Id

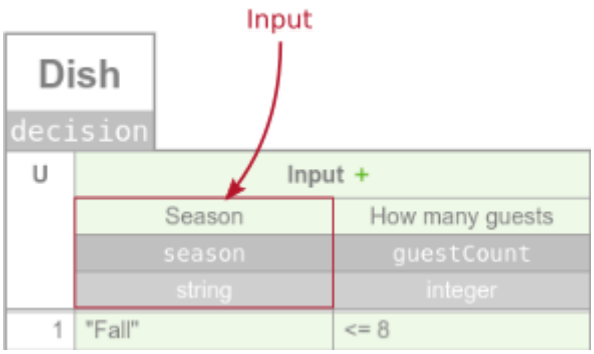


The id is the technical identifier of the decision. It is set in the `id` attribute on the `decision` element.

Each decision should have an unique id when it is deployed (/manual/latest/user-guide/process-engine/decisions/repository/) to the Camunda BPM platform. The engine uses the id as the decision key of the deployed `DecisionDefinition`.

```
<decision id="dish" name="Dish">
  <decisionTable id="decisionTable">
  <!-- ... -->
  </decisionTable>
</decision>
```

DMN Decision Table Input



A decision table can have one or more inputs, also called input clauses. An input clause defines the id, label, expression and type of a decision table input.

An input clause is represented by an `input` element inside a `decisionTable` XML element.

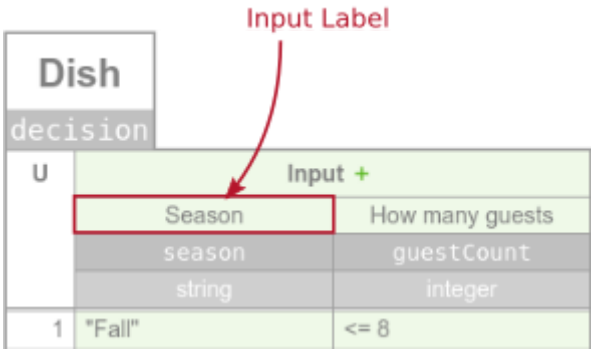
```
<definitions xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd" id="definitions" nam
  <decision id="dish" name="Dish">
    <decisionTable id="decisionTable">
      <input id="input1" label="Season">
        <inputExpression id="inputExpression1" typeRef="string">
          <text>season</text>
        </inputExpression>
      </input>
      <!-- ... -->
    </decisionTable>
  </decision>
</definitions>
```

Input Id

The input id is a unique identifier of the decision table input. It is used by the Camunda BPMN platform to reference the input in the history of evaluated decisions. Therefore, it is required by the Camunda DMN engine. It is set as the `id` attribute of the `input` XML element.

```
<input id="input1" label="Season">
  <inputExpression id="inputExpression1" typeRef="string">
    <text>season</text>
  </inputExpression>
</input>
```

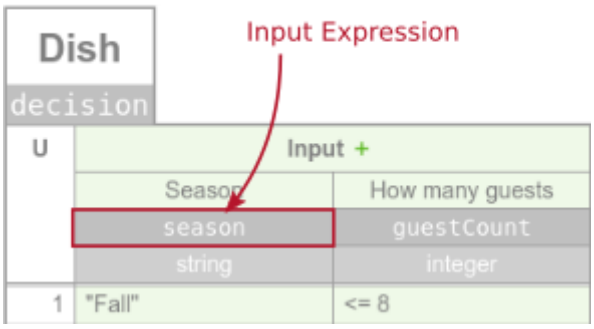
Input Label



An input label is a short description of the input. It is set on the `input` XML element in the `label` attribute. Note that the label is not required but recommended, since it helps to understand the decision.

```
<input id="input1" label="Season">
  <inputExpression id="inputExpression1" typeRef="string">
    <text>season</text>
  </inputExpression>
</input>
```

Input Expression

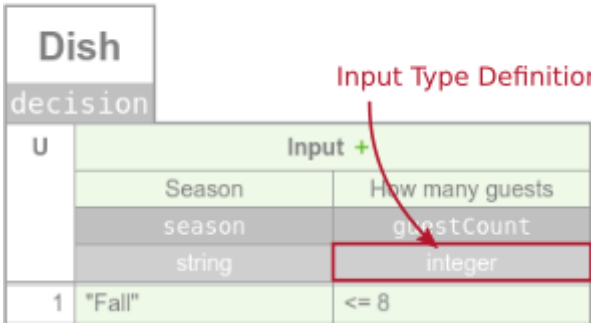


An input expression specifies how the value of the input clause is generated. It is an expression which will be evaluated by the DMN engine. It is usually simple and references a variable which is available during the evaluation. The expression is set inside a `text` element

that is a child of the `inputExpression` XML element.

```
<input id="input1" label="Season">
  <inputExpression id="inputExpression1" typeRef="string">
    <text>season</text>
  </inputExpression>
</input>
```

Input Type Definition



The type of the input clause can be specified by the `typeRef` attribute on the `inputExpression` XML element. After the input expression is evaluated by the DMN engine, it converts the result to the specified type. The supported types are listed in the User Guide (/manual/latest/user-guide/dmn-engine/data-types/#supported-data-types).

```
<input id="input1" label="Season">
  <inputExpression id="inputExpression1" typeRef="string">
    <text>season</text>
  </inputExpression>
</input>
```

Note that the type is not required but recommended, since it helps to understand the possible input values and provides a type safety to be aware of unexpected input values.

Input Expression Language

The expression language of the input expression can be specified by the `expressionLanguage` attribute on the `inputExpression` XML element. The supported expression languages are listed in the User Guide (/manual/latest/user-guide/dmn-engine/expressions-and-scripts/#supported-expression-languages).

```
<input id="input1" label="Season">
  <inputExpression id="inputExpression1" typeRef="string" expressionLanguage="groovy">
    <text>season</text>
  </inputExpression>
</input>
```

If no expression language is set then the global expression language, which is set on the `definitions` XML element, is used.

```
<definitions id="definitions"
             name="definitions"
             xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd"
             expressionLanguage="groovy"
             namespace="http://camunda.org/schema/1.0/dmn">
  <!-- ... -->
</definitions>
```

In case no global expression language is set, the default expression language is used instead. The default expression language for input expressions is JUEL. Please refer to the User Guide (/manual/latest/user-guide/dmn-engine/expressions-and-scripts/#default-expression-languages) to read more about expression languages.

Input Variable Name

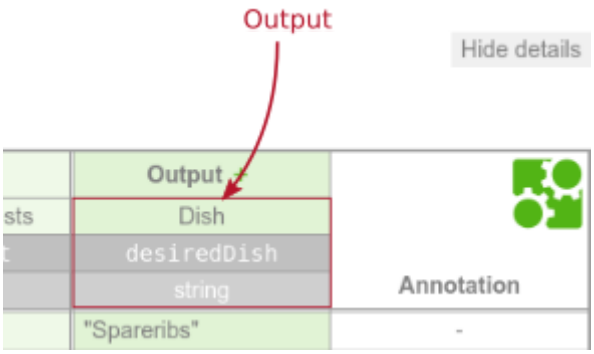
When the input expression is evaluated then the return value is stored in a variable. The name of the variable can be specified by the `camunda:inputVariable` extension attribute (/manual/latest/reference/dmn11/custom-extensions/camunda-attributes/#inputvariable) on the `input` element. By default, the name is `cellInput`.

To use the attribute you have to define the Camunda DMN namespace `xmlns:camunda="http://camunda.org/schema/1.0/dmn` in the XML.

```
<definitions id="definitions"
             name="definitions"
             xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd"
             xmlns:camunda="http://camunda.org/schema/1.0/dmn"
             namespace="http://camunda.org/schema/1.0/dmn">
  <decision id="dish" name="Dish">
    <decisionTable id="decisionTable">
      <input id="input1"
             label="Season"
             camunda:inputVariable="currentSeason">
        <!-- ... -->
      </input>
      <!-- ... -->
    </decisionTable>
  </decision>
</definitions>
```

The variable can be used in an expression of an input entry (/manual/latest/reference/dmn11/decision-table/rule/#input-entry-condition). For example, the JUEL expression `currentSeason != "Fall"` checks if the season input is not `"Fall"`.

DMN Decision Table Output



A decision table can have one or more outputs, also called output clauses. An output clause defines the id, label, name and type of a decision table output.

An output clause is represented by an `output` element inside a `decisionTable` XML element.

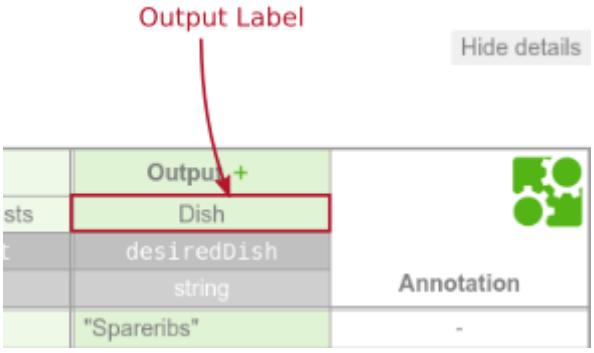
```
<definitions xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd" id="definitions" nam
  <decision id="dish" name="Dish">
    <decisionTable id="decisionTable">
      <!-- ... -->
      <output id="output1" label="Dish" name="desiredDish" typeRef="string" />
      <!-- ... -->
    </decisionTable>
  </decision>
</definitions>
```

Output Id

The output id is a unique identifier of the decision table output. It is used by the Camunda BPMN platform to reference the output in the history of evaluated decisions. Therefore, it is required by the Camunda DMN engine. It is set as the `id` attribute of the `output` XML element.

```
<output id="output1" label="Dish" name="desiredDish" typeRef="string" />
```

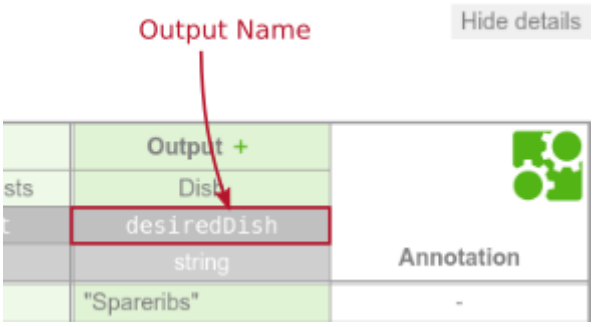
Output Label



An output label is a short description of the output. It is set on the `output` XML element in the `label` attribute. Note that the label is not required but recommended, since it helps to understand the decision.

```
<output id="output1" label="Dish" name="desiredDish" typeRef="string" />
```

Output Name

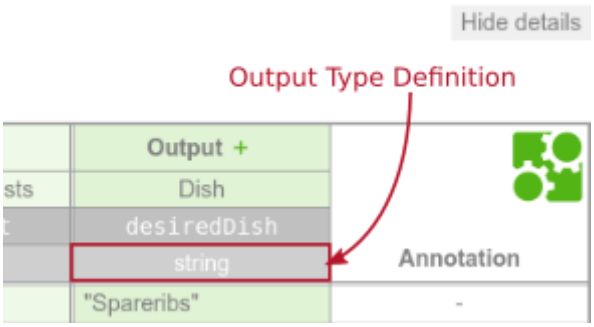


The name of the output is used to reference the value of the output in the decision table result (/manual/latest/user-guide/dmn-engine/evaluate-decisions/#interpret-the-dmndecisiontableresult). It is specified by the `name` attribute on the `output` XML element.

If the decision table has more than one output, then all outputs must have a unique name.

```
<output id="output1" label="Dish" name="desiredDish" typeRef="string" />
```

Output Type Definition



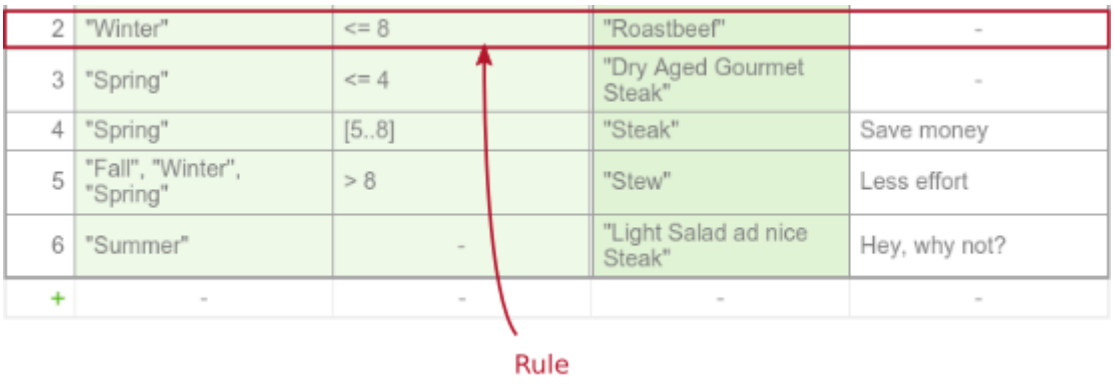
The type of the output clause can be specified by the `typeRef` attribute on the `output` XML element. After an output entry (/manual/latest/reference/dmn11/decision-table/rule/#output-entry-conclusion) is evaluated by the DMN engine, it converts the result to the specified type. The supported types are listed in the User Guide (/manual/latest/user-guide/dmn-engine/data-types/#supported-data-types).

```
<output id="output1" label="Dish" name="desiredDish" typeRef="string" />
```

Note that the type is not required but recommended, since it provides a type safety of the output values.

Additionally, the type can be used to transform the output value into another type. For example, transform the output value `80%` of type String into a Double using a custom data type (/manual/latest/user-guide/dmn-engine/data-types/#implement-a-custom-data-type).

DMN Decision Table Rule

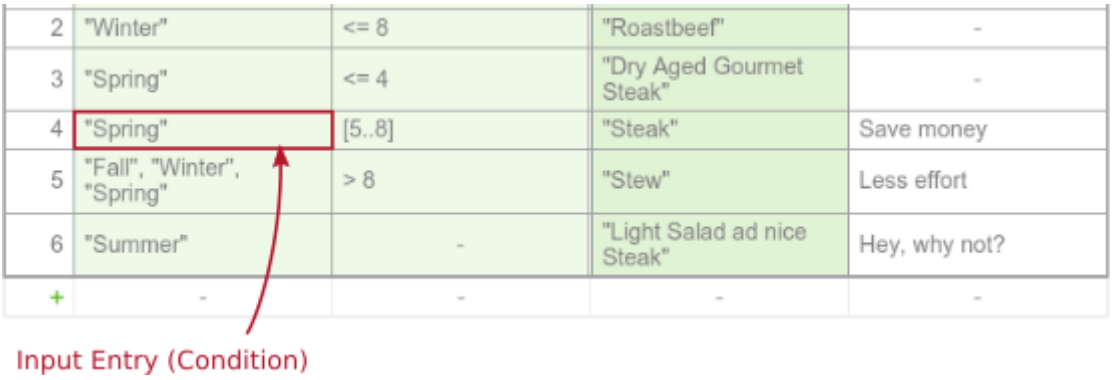


A decision table can have one or more rules. Each rule contains input and output entries. The input entries are the condition and the output entries the conclusion of the rule. If each input entry (condition) is satisfied, then the rule is satisfied and the decision result contains the output entries (conclusion) of this rule.

A rule is represented by a `rule` element inside a `decisionTable` XML element.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd" id="definitions" nam
  <decision id="dish" name="Dish">
    <decisionTable id="decisionTable">
      <!-- ... -->
      <rule id="rule2-950612891-2">
        <inputEntry id="inputEntry21">
          <text>"Winter"</text>
        </inputEntry>
        <inputEntry id="inputEntry22">
          <text><![CDATA[<= 8]]></text>
        </inputEntry>
        <outputEntry id="outputEntry2">
          <text>"Roastbeef"</text>
        </outputEntry>
      </rule>
      <!-- ... -->
    </decisionTable>
  </decision>
</definitions>
```

Input Entry (Condition)



A rule can have one or more input entries, which are the conditions of the rule. Each input entry contains an expression in a `text` element as child of an `inputEntry` XML element.

The input entry is satisfied when the evaluated expression returns `true`.

```
<inputEntry id="inputEntry41">
  <text>"Spring"</text>
</inputEntry>
```

Empty Input Entry

In case an input entry is irrelevant for a rule, the expression is empty, which is always satisfied.

```
<inputEntry id="inputEntry41">
  <text/>
</inputEntry>
```

If FEEL is used as expression language, then an empty input entry is represented by a `-`. Otherwise, the expression is empty.

Expression Language of an Input Entry

The expression language of the input entry can be specified by the `expressionLanguage` attribute on the `inputEntry` XML element. The supported expression languages are listed in the User Guide (/manual/latest/user-guide/dmn-engine/expressions-and-scripts/#supported-expression-languages).

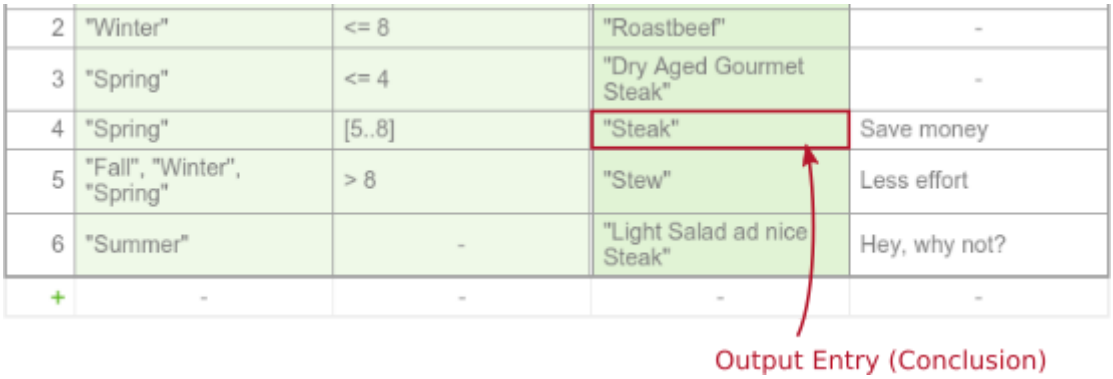
```
<inputEntry id="inputEntry41"
            expressionLanguage="juel">
  <text>cellInput == "Spring"</text>
</inputEntry>
```

If no expression language is set then the global expression language, which is set on the `definitions` XML element, is used.

```
<definitions id="definitions"
             name="definitions"
             xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd"
             expressionLanguage="groovy"
             namespace="http://camunda.org/schema/1.0/dmn">
  <!-- ... -->
</definitions>
```

In case no global expression language is set, the default expression language is used instead. The default expression language for input entries is FEEL (/manual/latest/reference/dmn11/feel/). Please refer to the User Guide (/manual/latest/user-guide/dmn-engine/expressions-and-scripts/#default-expression-languages) to read more about expression languages.

Output Entry (Conclusion)



A rule can have one or more output entries, which are the conclusions of the rule. Each output entry contains an expression in a `text` element as child of an `outputEntry` XML element.

```
<outputEntry id="outputEntry4">
  <text>"Steak"</text>
</outputEntry>
```

Empty Output Entry

If the output entry is empty, then the output is ignored and not part of the decision table result.

```
<outputEntry id="outputEntry4">
  <text/>
</outputEntry>
```

Expression Language of an Output Entry

The expression language of the expression can be specified by the `expressionLanguage` attribute on the `outputEntry` XML element. The supported expression languages are listed in the User Guide (/manual/latest/user-guide/dmn-engine/expressions-and-scripts/#supported-expression-languages).

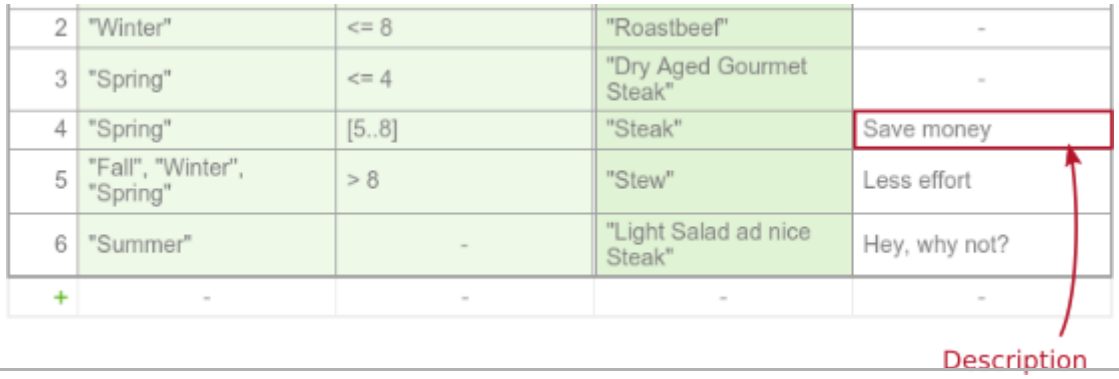
```
<outputEntry id="outputEntry4"
             expressionLanguage="groovy">
  <text>"Steak"</text>
</outputEntry>
```

If no expression language is set then the global expression language, which is set on the `definitions` XML element, is used.

```
<definitions id="definitions"
             name="definitions"
             xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd"
             expressionLanguage="groovy"
             namespace="http://camunda.org/schema/1.0/dmn">
  <!-- ... -->
</definitions>
```

In case no global expression language is set, the default expression language is used instead. The default expression language for output entries is JUEL. Please refer to the User Guide (/manual/latest/user-guide/dmn-engine/expressions-and-scripts/#default-expression-languages) to read more about expression languages.

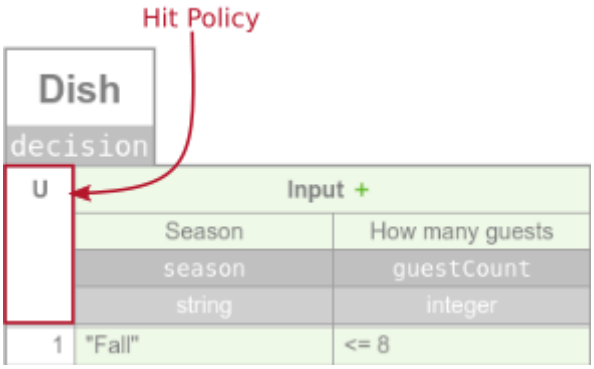
Description



A rule can be annotated with a description that provides additional information. The description text is set inside the `description` XML element.

```
<rule id="rule4">
  <description>Save money</description>
  <!-- ... -->
</rule>
```

DMN Hit Policy



A decision table has a hit policy that specifies what the results of the evaluation of a decision table consist of.

The hit policy is set in the `hitPolicy` attribute on the `decisionTable` XML element. If no hit policy is set, then the default hit policy `UNIQUE` is used.

```
<definitions xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd" id="definitions" nam
  <decision id="dish" name="Dish">
    <decisionTable id="decisionTable" hitPolicy="RULE ORDER">
      <!-- .. -->
    </decisionTable>
  </decision>
</definitions>
```

In the visual representation of the decision table, the hit policy is specified by the initial letter of the hit policy. The following hit policies are supported by the Camunda DMN engine:

Visual representation	XML representation
U	UNIQUE
A	ANY
F	FIRST
R	RULE ORDER
C	COLLECT

The Role of a Hit Policy

A hit policy specifies how many rules of a decision table can be satisfied and which of the satisfied rules are included in the decision table result (/manual/latest/user-guide/dmn-engine/evaluate-decisions/#interpret-the-dmndecisiontableresult). The hit policies Unique, Any and First will always return a maximum of one satisfied rule. The hit policies Rule Order and Collect can return multiple satisfied rules.

Unique Hit Policy

Only a single rule can be satisfied. The decision table result contains the output entries of the satisfied rule.

If more than one rule is satisfied, the Unique hit policy is violated.

See the following decision table.

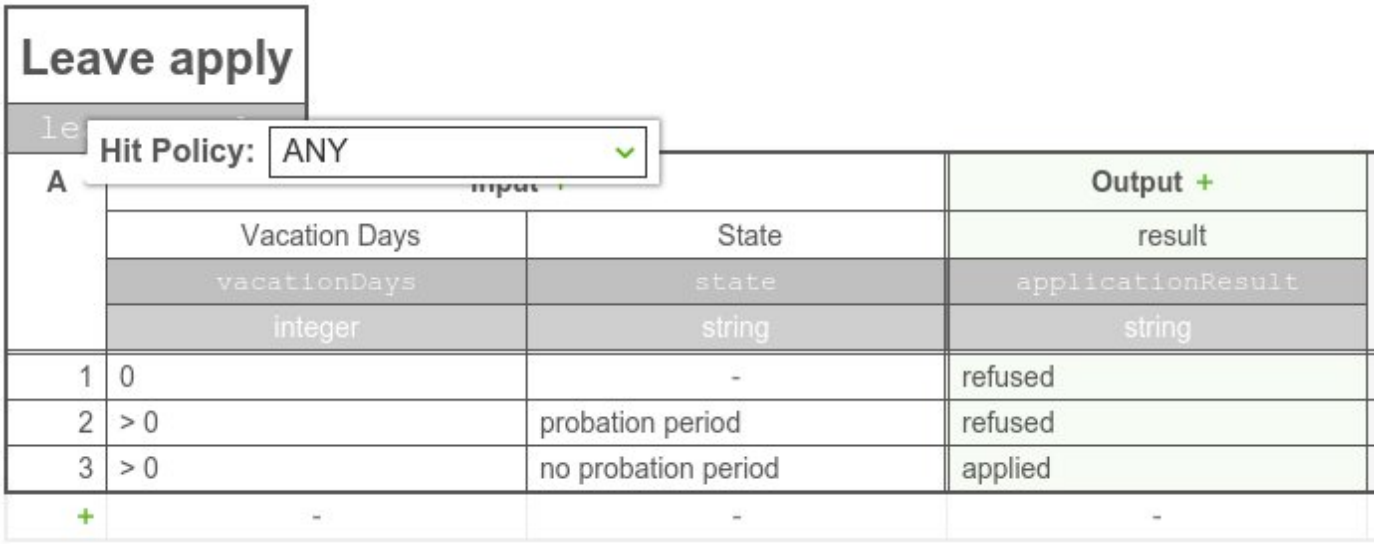
Depending on the current season the dish should be chosen. Only one dish can be chosen, since only one season can exist at the same time.

Any Hit Policy

Multiple rules can be satisfied. However, all satisfied rules must generate the same output. The decision table result contains only the output of one of the satisfied rules.

If multiple rules are satisfied which generate different outputs, the hit policy is violated.

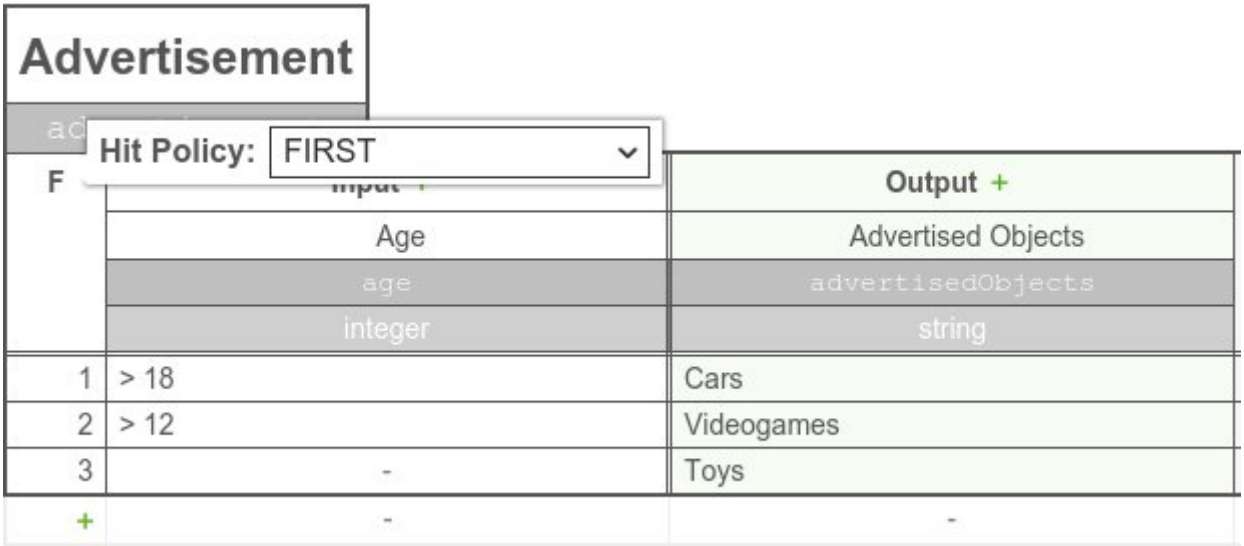
See the following example:



This is a decision table for the leave application. If the applier has no vacation days left or is currently in the probation period, the application will be refused. Otherwise the application is applied.

First Hit Policy

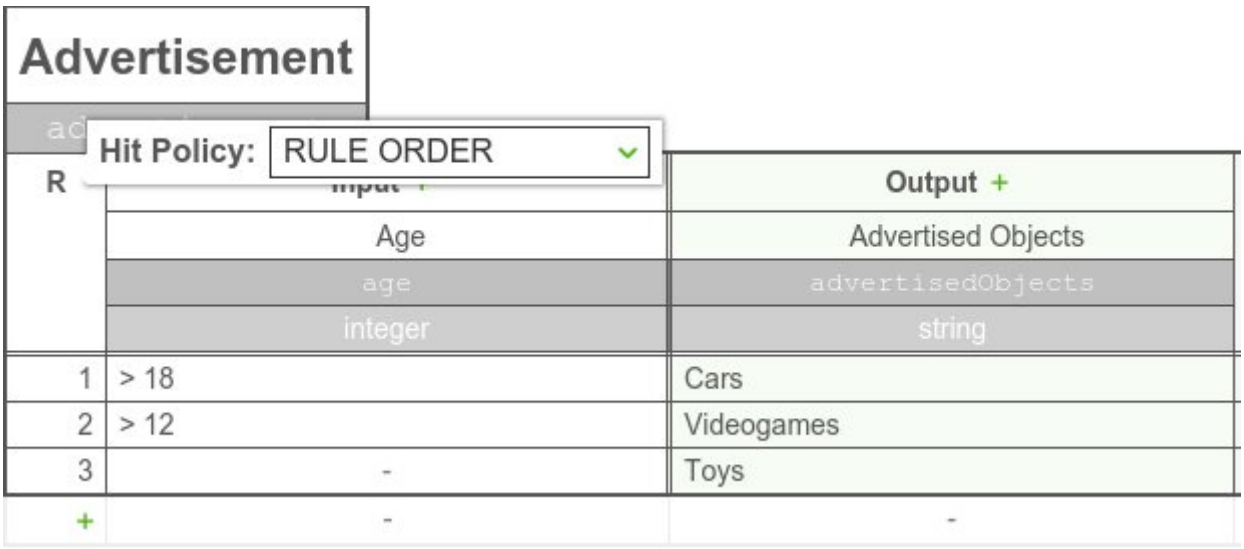
Multiple rules can be satisfied. The decision table result contains only the output of the first satisfied rule.



See the above decistion table for advertisement. Regarding the current age of the user, which advertisement should be shown is decided. For example, the user is 19 years old. All the rules will match, but since the hit policy is set to first only, the advertisement for Cars is used.

Rule Order Hit Policy

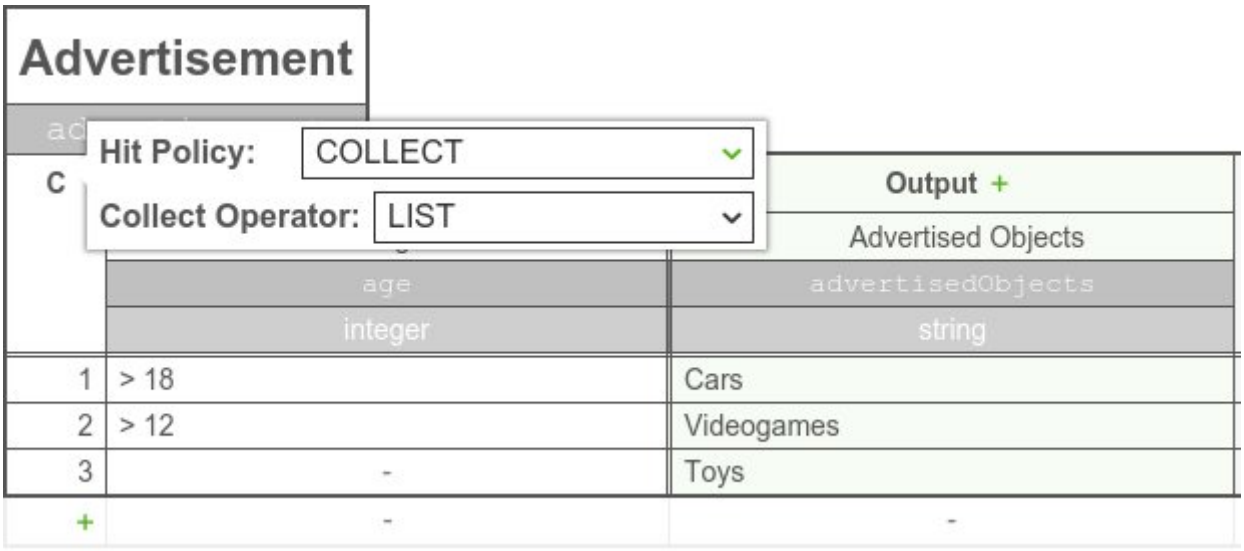
Multiple rules can be satisfied. The decision table result contains the output of all satisfied rules in the order of the rules in the decision table.



Again, see the advertisement example with the rule order policy. Say we have a user at the age of 19 again. All rules are satisfied so all outputs are given, ordered by the rule ordering. It can perhaps be used to indicate the priority of the displayed advertisements.

Collect Hit Policy

Multiple rules can be satisfied. The decision table result contains the output of all satisfied rules in an arbitrary order as a list.



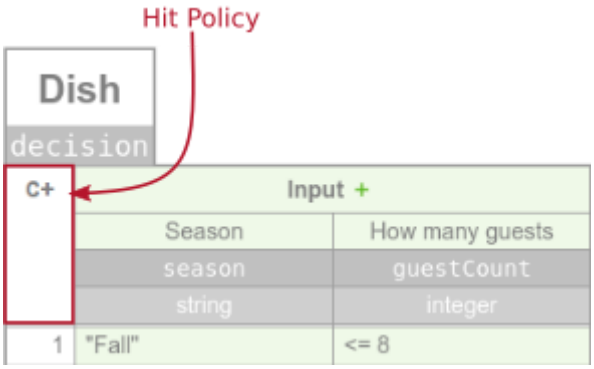
With this hit policy, the output list has no ordering. So the advertisement will be arbitrary if, for example, the age is 19.

Additionally, an aggregator can be specified for the Collect hit policy. If an aggregator is specified, the decision table result will only contain a single output entry. The aggregator will generate the output entry from all satisfied rules. **Note** if the Collect hit policy is used with an aggregator, the decision table can only have one output.

The aggregator is set as the `aggregation` attribute of the `decisionTable` XML element.

```
<decisionTable id="decisionTable" hitPolicy="COLLECT" aggregation="SUM">
  <!-- .. -->
</decisionTable>
```

Aggregators for Collect Hit Policy

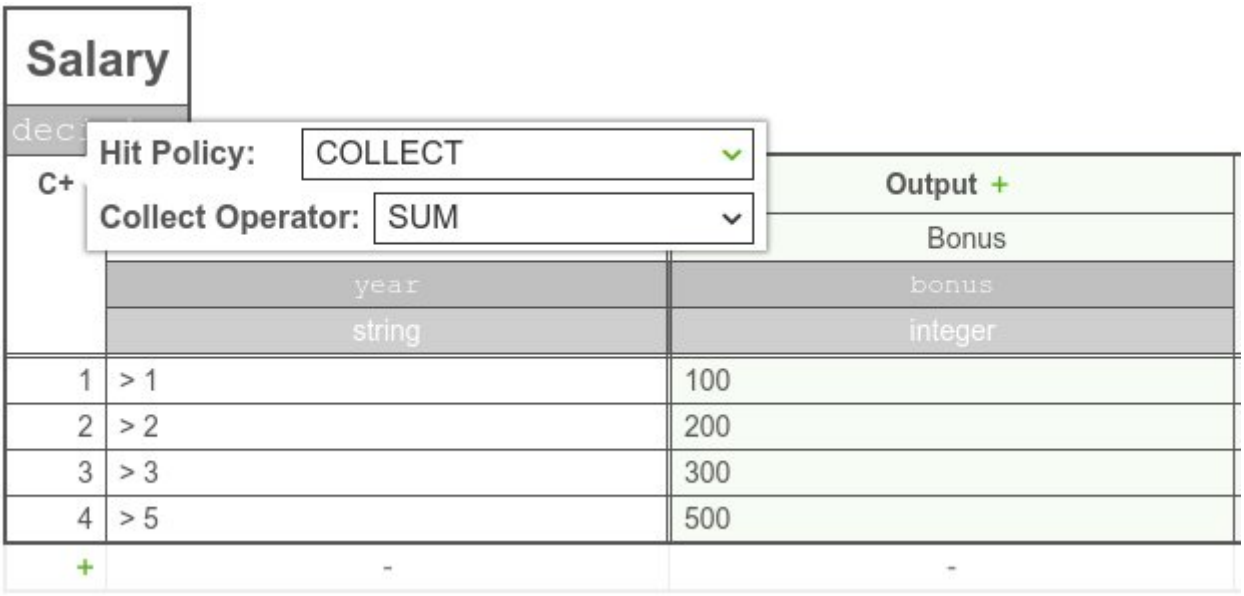


In the visual representation of the decision table the aggregator is specified by a marker after the hit policy. The following aggregators are supported by the Camunda DMN engine:

Visual representation	XML representation	Result of the aggregation
+	SUM	the sum of all output values
<	MIN	the smallest value of all output values
>	MAX	the largest value of all output values
#	COUNT	the number of output values

SUM aggregator

The SUM aggregator sums up all outputs from the satisfied rules.



The showed decision table can be used to sum up the salary bonus for an employee. For example, the employee has been working in the company for 3.5 years. So the first, second and third rule will match and the result of the decision table is 600, since the output is summed up.

MIN aggregator

The MIN aggregator can be used to return the smallest output value of all satisfied rules. See the following example of a car insurance. After years without a car crash the insurance fee will be reduced.

Car insurance

Hit Policy: COLLECT
Collect Operator: MIN

	C<	Output +
		Insurance fee
	year	fee
	integer	double
1	-	205.43
2	> 2	150.21
3	> 3	98.83
4	> 4	64.32
+	-	-

For example, if the input for the decision table is 3.5 years, the result will be 98.83, since the first three rules match but the third rule has the minimal output.

MAX aggregator

The MAX aggregator can be used to return the largest output value of all satisfied rules.

Pocket Money

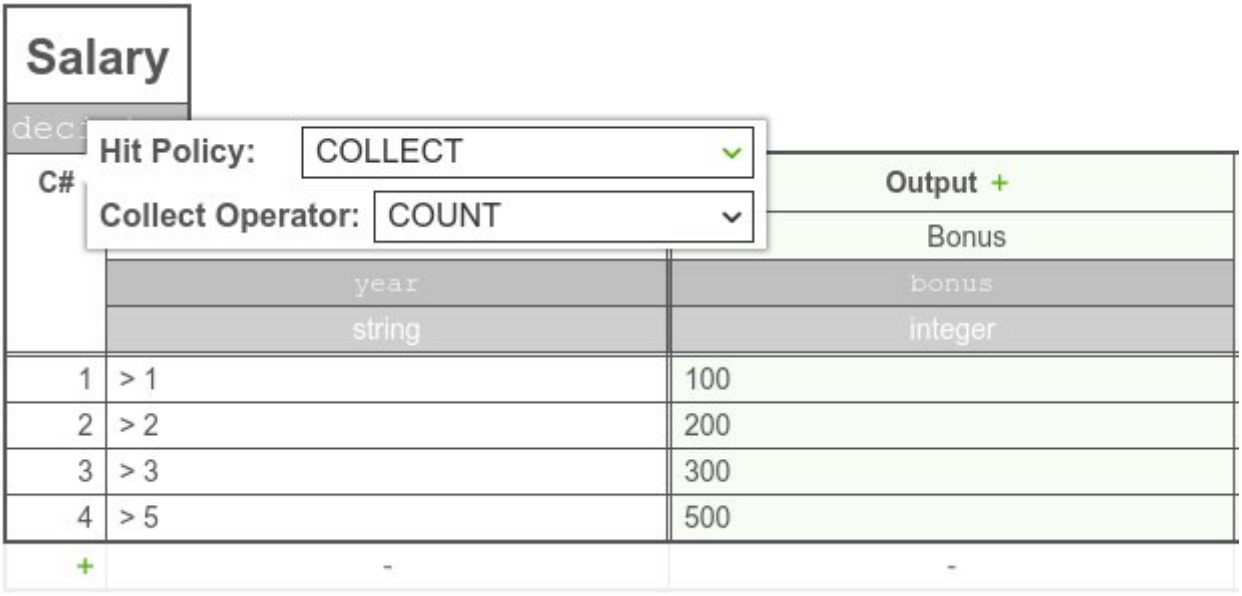
Hit Policy: COLLECT
Collect Operator: MAX

	C>	Output +
		Amount
	age	amount
	integer	integer
1	> 5	2
2	> 8	5
3	> 14	20
4	> 16	50
+	-	-

This decision table represents the decision for the amount of pocket money for a child. Depending of the age, the amount grows. For example, an input of 9 will satisfy the first and second rules. The output of the second rule is larger then the output of the first rule, so the output will be 5. A child at the age of 9 will get 5 as pocket money.

COUNT aggregator

The COUNT aggregator can be use to return the count of satisfied rules.



For example, see the salary bonus decision table again, this time with the COUNT aggregator. With an input of 4, the first three rules will be satisfied. Therefore, the result from the decision table will be 3, which means that after 4 years the result of the decision table is 3 salary bonuses.