Oracle® Cloud

Using Decisions in Oracle Integration 3

G28051-02
May 2025

Oracle Cloud Using Decisions in Oracle Integration 3,

G28051-02

Primary Author: Oracle Corporation

Contents

Preface

Audience vi
Documentation Accessibility vi
Diversity and Inclusion vi
Related Resources vi
Conventions vii

1 Learn About Decision Modeling

What Are Decisions? 1-1
About Decision Model Notation 1-1
About Friendly Enough Expression Language 1-2
- Data Types 1-2
- Grammar Rules 1-3
- Built-In Functions 1-4
  - String Functions 1-4
  - List Functions 1-5
  - Numeric Functions 1-6
  - Boolean Functions 1-6
  - Conversion Functions 1-6
- List Iteration Expressions Using Keywords 1-7
- Date, Time, and Duration Functions 1-7
  - Conversion Examples 1-7
  - Arithmetic Operation Examples 1-8
  - Comparison Operation Examples 1-9

2 Learn About Decisions in Oracle Integration

How to Model Decisions in Oracle Integration? 2-1
Understand the Decision Designer 2-2
- Decision Requirement Diagrams 2-2
- Decision Designer Components 2-2

Workflow for Using Decision Models in Oracle Integration 2-5

Best Practices for Modeling Decisions 2-6

3 Design Decision Models

Create Decision Models 3-1
Create a New Decision Model 3-1
Import a Decision Model 3-2
Add Decisions 3-2
Define Decision Input and Type 3-4
Create Input Data 3-4
Define Custom Data Types 3-5
Construct a Data Type 3-6
Import a JSON Schema 3-7
Connect Nodes on the Canvas 3-8
Configure a Decision's Logic 3-8
Configure Empty Decisions 3-8
Configure Expressions 3-9
Configure Decision Tables 3-10
About Decision Table Elements 3-10
Specify a Decision Table's Logic 3-25
Configure If / Else Statements 3-26
Configure Functions 3-28
Configure Lists 3-30
Configure Contexts 3-32
Configure Relations 3-34
Configure Loops 3-36
Review and Fix Errors in a Decision 3-39

4 Test and Activate Decision Models

Test a Decision Model 4-1
Expose Decisions as Services 4-1
Activate a Decision Model 4-2
Add a Decision Model to an Integration 4-3

5 Manage Decision Models

Update a Decision Model 5-1
Semantic Versioning Rules 5-2
Clone a Decision Model 5-2
Deactivate a Decision Model 5-2
Edit or View a Decision Model 5-3

Delete a Decision Model 5-3

6 Troubleshoot Decisions

The Option to Activate a Decision Model Is Not Displayed 6-1
Unable to Add a Decision Model to an Integration 6-1
An Input Used in a Decision Is Not Recognized 6-2

Preface

This guide describes how to model decisions in Oracle Integration.

Topics:

- Audience
- Documentation Accessibility
- Diversity and Inclusion
- Related Resources
- Conventions

Audience

This guide is intended for users who want to model and manage decisions in Oracle Integration.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://support.oracle.com/portal/ or visit Oracle Accessibility Learning and Support if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Resources

For more information, see these Oracle resources:

- Oracle Integration documentation on the Oracle Help Center.

- Oracle Cloud at http://cloud.oracle.com.

Conventions

The following text conventions are used in this document.

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Learn About Decision Modeling

In business workflows, you'll often be required to create decisions that enable you to automate policies, computations, and reasoning. Using Oracle Integration, you can model the decisions for your workflows and drive better outcomes.

Discover what decisions are, the supported decision modeling standard, and the syntax for defining them.

Topics:

- What Are Decisions?
- About Decision Model Notation
- About Friendly Enough Expression Language

What Are Decisions?

In business contexts, decisions are the fundamental building blocks of operations and strategy.

Business operations are driven by a constant stream of decisions. Whether it's approving loan applications, deciding on document changes, dispatching emergency services, or calculating bonus shopping points, each choice is crucial to the operation's success. These decisions can greatly impact an organization's efficiency and customer satisfaction, making it essential to have a structured approach to decision-making.

A decision model framework provides such a structure. By employing a decision model framework, you can articulate a wide spectrum of automated decisions. You can model decisions as a hierarchical tree of simpler sub-decisions, each easily managed with decision tables and straightforward expressions, rather than complex production rules.

Oracle Integration utilizes the Decision Model and Notation (DMN) framework to provide a robust and intuitive decision-modeling capability.

About Decision Model Notation

Decision Model and Notation (DMN), developed by the Object Management Group (OMG), is a standard for modeling operational decisions.

It enables organizations to visualize, document, and automate complex business logic. DMN provides a clear representation of decision-making processes based on various inputs, rules, and considerations. This visual approach facilitates improved understanding, resulting in more accurate decisions.

Additionally, DMN's XML-based schema provides a standardized format for exchanging decision models. This enables seamless interoperability between DMN-compliant platforms and organizations, promoting consistency, reducing errors, and accelerating the development and deployment of decision services.

Oracle Integration supports the Decision Model and Notation (DMN) standard, version 1.1.

For more information on DMN, see the DMN Specification Document by OMG.

About Friendly Enough Expression Language

Decision Model and Notation (DMN) defines Friendly Enough Expression Language (FEEL) to provide standard executable semantics to all expressions used within a decision model.

In Oracle Integration, you use FEEL to define expressions in the logic of all decision types, including decision tables.

Explore the FEEL data types, syntax, and various functions supported in Oracle Integration.

- Data Types
- Grammar Rules
- Built-In Functions
- List Iteration Expressions Using Keywords
- Date, Time, and Duration Functions

Data Types

Oracle Integration supports the following FEEL data types for decision input data, expression values, function arguments, and return values.

FEEL Data Type	Notation in Oracle Integration	Description
number	Number	FEEL numbers are based on the IEEE 754-2008 Decimal128 format, with 34 decimal digits of precision and rounding toward the nearest neighbor with ties favoring the even neighbor. Numbers are a restriction of the XML Schema type `precisionDecimal`, and are equivalent to Java `BigDecimal` with `MathContext DECIMAL128` .
string	Text	Variable-length sequence of characters italicized or encapsulated in double quotes.
boolean	True or False	Logical Boolean (true/false).
date-time	Date and Time	Calendar date and time combination.
duration	Duration	Duration in the ISO 8601 date-time format.

> **Note:**
>
> You can extend these basic data types by defining custom data types. See Define Custom Data Types.

Grammar Rules

Learn about the syntax for commonly-used FEEL expressions through simple examples. For the complete definition of FEEL syntax, see *Decision Model and Notation (DMN), v1.1*.

Arithmetic Expressions

Name	FEEL Expression	Return Value
Addition (+)	0.15+30	30.15
Subtraction (-)	15-30	-15
Multiplication (*)	.20*40.02	8.004
Division (/)	1/50	0.02
Exponentiation (**)	2**3	8

Interval Expressions

Start Value	End Value	FEEL Expression	Return Value
Inclusive	Inclusive	15 in [15..30]	true
Exclusive	Exclusive	15 in (15..30)	false
Exclusive	Inclusive	30 in (15..30]	true
Inclusive	Exclusive	30 in [15..30)	false

> **Note:**
>
> In decision table input entry and input/output allowed value cells, you can use intervals or list of intervals to test against the input data.

Comparison Expressions

Name	FEEL Expression	Return Value
Less than (<)	8<2**3	false
Less than or equal to (<=)	15 in (<=15)	true
Equal (=)	8=2**3	true
Greater than (>)	30 in (>30)	false
Greater than or equal to (>=)	1/5>=0.20	true
Not equal to (!=)	8!=2**3	false

> **Note:**
>
> In decision table input entry and input/output allowed value cells, you can use comparison operators to define unary expressions.

Other Expressions

Name	FEEL Expression	Return Value
Disjunction	(2*2=2**2) or (3*2=3**2)	true
Conjunction	(2*2=2**2) and (3*2=3**2)	false
Negation	not(2*2=2**2)	false

> **Note:**
>
> In decision table input entry and input/output allowed value cells, you can use comma-separated list of values to specify disjunction.

Built-In Functions

FEEL includes a library of built-in functions that you can use to define expressions.

Oracle Integration supports the following types of built-in functions in decisions. Additionally, to assist with expression creation, a list of built-in functions is presented as suggestions when you click an empty expression field.

- String Functions
- List Functions
- Numeric Functions
- Boolean Functions
- Conversion Functions

String Functions

Name(parameters)	Parameter Domain	Description	Example
string(*from*)	non-null	Convert *from* to a string.	string(1.1) = "1.1" string(null) = null
substring(*string, start, length*)	string, number1	Return *length* (or all) characters in *string*, starting at *start position*. The first position is 1, last position is -1.	• substring("redwood",3) = "dwood" • substring("redwood",3,3) = "dwo" • substring("redwood", -2, 1) = "o"
string length(*string*)	string	Return length of *string*.	string length("red") = 3
upper case(*string*)	string	Return uppercased *string*.	upper case("aBc4") = "ABC4"
lower case(*string*)	string	Return lowercased *string*.	lower case("aBc4") = "abc4"
substring before (*string, match*)	string, string	Return substring of *string* before the *match* in *string*.	• substring before("redwood", "wood") = "red" • substring before("redwood", "xyz") = ""
substring after (*string, match*)	string, string	Return substring of *string* after the *match* in *string*.	• substring after("redwood", "dw") = "ood" • substring after("", "o") = ""

Name(parameters)	Parameter Domain	Description	Example
replace(*input, pattern, replacement, flags?*)	string2	Regular expression pattern matching and replacement.	replace("abcd", "(ab)|(a)", "[1=$1] [2=$2]") = "[1=ab] [2=]cd"
contains(*string, match*)	string	Does the *string* contain the *match*?	contains("redwood", "de") = false
starts with(*string, match*)	string	Does the *string* start with the *match*?	starts with("redwood", "re") = true
ends with(*string, match*)	string	Does the *string* end with the *match*?	ends with("redwood", "d") = true
matches(*input, pattern, flags?*)	string2	Does the *input* match the regexp *pattern?*	matches("redwood", "^re*w") = true

List Functions

Name(parameters)	Parameter Domain	Description	Example
list contains(*list, element*)	list, any element of the semantic domain including **null**	Does the *list* contain the *element*?	list contains([1,2,3], 2) = true
count(*list*)	list	Return size of *list*.	count([1,2,3]) = 3
minimum(*list*)	(list of) comparable items	Return minimum item.	minimum([1,2,3]) = 1
maximum(*list*)	(list of) comparable items	Return maximum item.	maximum([1,2,3]) = 3
sublist(*list, start position, length?*)	list, number1, number2	Return list of *length* (or all) elements of *list*, starting with *list[start position]*. The first position is 1, last position is -1.	sublist([1,2,3], 1, 2) = [2]
append(*list, item…*)	list, any element including **null**	Return new *list* with *items* appended.	append([1], 2, 3) = [1,2,3]
concatenate(*list…*)	list	Return new *list* that is a concatenation of the arguments.	concatenate([1,2],[3]) = [1,2,3] concatenate(1,2,3) = [1,2,3] concatenate([1,2],3) = [1,2,3]
insert before(*list, position, newItem*)	list, number1, any element including **null**	Return new *list* with *newItem* inserted at *position*.	insert before([1,3],1,2) = [1,2,3]
remove(*list, position*)	list, number1	The *list* with item at *position* removed.	remove([1,2,3], 2) = [1,3]
reverse(*list*)	list	Reverse the *list*.	reverse([1,2,3]) = [3,2,1]
index of(*list, match*)	list, any element including **null**	Return ascending list of *list* positions containing *match*.	index of([1,2,3,2],2) = [2,4]
union(*list…*)	list	Concatenate with duplicate removal.	union([1,2],[2,3]) = [1,2,3]
distinct values(*list*)	list	Duplicate removal.	distinct values([1,2,3,2,1]) = [1,2,3]
flatten(*list*)	list	Flatten nested lists.	flatten([[1,2],[[3]], 4]) = [1,2,3,4]
sum(*list*)	(list of) numbers	Return sum of numbers.	sum([1,2,3]) = 6
mean(*list*)	(list of) numbers	Return arithmetic mean (average) of numbers.	mean([1,2,3]) = 2

Numeric Functions

Name(parameters)	Parameter Domain	Description	Example
number(*from, grouping separator, decimal separator*)	number, separator, decimal notation	Convert *from* to a number.	number("1 000,0", " ", ",") = number("1,000.0", ",", ".")
decimal(*n, scale*)	number, number1	Return *n* with given *scale*.	• decimal(1/3, 2) = .33 • decimal(1.5, 0) = 2 • decimal(2.5, 0) = 2
floor(n)	number	Return greatest integer <= n.	• floor(1.5) = 1 • floor(-1.5) = -2
ceiling(n)	number	Return smallest integer >= n.	• ceiling(1.5) = 2 • ceiling(-1.5) = -1

Boolean Functions

Name(parameters)	Parameter Domain	Description	Example
not(*negand*)	boolean	Logical negation.	• not(true) = false • not(null) = null

Conversion Functions

Name(parameters)	Parameter Domain	Description	Example
date(*from*)	date string	Convert *from* to a date.	date("2012-12-25") - date("2012-12-24") = duration("P1D")
date(*from)*	date and time	Convert *from* to a date (set time components to null).	date(date and time("2012-12-25T11:00:00Z")) = date("2012-12-25")
date and time(*from*)	date time string	Convert *from* to a date and time.	date and time("2012-12-24T23:59:00") + duration("PT1M") = date and time("2012-12-25T00:00:00")
time(*from*)	time string	Convert *from* to time.	time("23:59:00") + duration("PT2M") = time("00:01:00")
time(*from*)	time, date and time	Convert *from* to time (ignoring date components).	time(date and time("2012-12-25T11:00:00Z")) = time("11:00:00")
duration(*from*)	duration string	Convert *from* to a date and time or years and months duration.	• date and time("2012-12-24T23:59:00") - date and time("2012-12-22T03:45:00") = duration("P2DT20H14M") • duration("P2Y2M") = duration("P26M")
years and months duration(*from, to*)	both are date and time	Return years and months duration between *from* and *to*.	years and months duration(date("2011-12-22"), date("2013-08-24")) = duration("P1Y8M")

List Iteration Expressions Using Keywords

Common list iteration expressions, demonstrating the usage of *for*, *some*, *every*, *in*, *return*, and *satisfies* keywords.

To assist with expression creation, a list of keywords is presented as suggestions when you click an empty expression field.

Name(parameters)	Description	Example
for [item] in [list] return [expression]	Iterate over a list.	for *i* in [1,2,3,4] return *i*i* = [1,4,9,16]
sum (for [item] in [list] return [expression])	Iterate over a list and return the sum of iterations.	sum(for *i* in [1,2,3,4] return *i*i*) = 30
every [item] in [list] satisfies [expression]	Test if every item in the list satisfies the test condition described by the expression.	• every *n* in [12,50,51] satisfies *n* > *5* = true • every *n* in [12,50,51] satisfies *n* < *50* = false
some [item] in [list] satisfies [expression]	Test if at least one of the items in the list satisfies the test condition described by the expression.	• some *n* in [12,50,51] satisfies *n* > *50* = true • some *n* in [12,50,51] satisfies *n* > *51* = false

Date, Time, and Duration Functions

Because FEEL does not support literal representation of date, time, or duration values, you must use a combination of built-in functions and string or number literals to express these values.

Built-in functions extract date, time, and duration data from strings or numbers. The following sections provide examples of common date and time operations using built-in functions:

- Conversion Examples
- Arithmetic Operation Examples
- Comparison Operation Examples

Conversion Examples

Examples in this section demonstrate conversion of number or string literals into date, time, or duration data types.

Date and Time Data Type Examples

The following examples convert string literals to date or date-time values.

Example	Description
date("2012-12-25")	Represents the date 2012-12-25 in the *YYYY-MM-DD* format.
time("23:59:00")	Represents the time 23:59:00 in the *hh:mm:ss* format.
time("23:59:00-08:00")	Represents the local time 23:59:00 in the *hh:mm:ss* format; the local time is 8 hours behind UTC.
date and time("2012-12-25T11:00:00")	Represents the date 2012-12-25 and time 11:00:00 in *YYYY-MM-DD* and *hh:mm:ss* formats, respectively.

Example	Description
date and time("2012-12-25T11:00:00Z")	Represents the date 2012-12-25 and time 11:00:00 in *YYYY-MM-DD* and *hh:mm:ss* formats, respectively; "Z" represents UTC time.

Duration Data Type Examples

A few conversion examples using the duration function are listed in the following table.

Example	Description
duration("P1DT12H30M")	Represents a duration of 1 day, 12 hours, and 30 minutes.
duration("-P120D")	Represents a duration of minus 120 days.
duration("PT2000H")	Represents a duration of 2000 hours.
duration("P1Y2M3DT10H30M")	Represents a duration of 1 year, 2 months, 3 days, 10 hours, and 30 minutes.

Arithmetic Operation Examples

Examples in this section demonstrate arithmetic operations that can be performed on date, time, or duration data types.

Operator	Example	Result	Description
(+)	date("2016-12-25") + duration("P3Y2M6D")	date("2020-03-02")	Returns the date 3 years 2 months and 6 days after 2016-12-25.
(+)	date and time("2016-12-25T12:30:00Z") + duration("P1Y2M3DT10H30M")	date and time("2018-02-28T23:00:00Z")	Returns the date and time 1 year 2 months 3 days and 10 hours 30 minutes after the date 2016-12-25 and time 12:30:00.
(+)	date("2016-12-25") + duration("-P3Y2M6D")	date("2013-10-19")	Returns the date 3 years 2 months and 6 days before 2016-12-25.
(-)	date("2015-12-25") - date("2012-12-25")	duration("P1095DT0H0M0S")	Returns a duration indicating number of days and time.
(-)	date and time("2012-12-25T12:00:00") - date and time("2015-12-25T11:12:00")	duration("-P1094DT23H12M0S")	Returns a duration indicating number of days and time.
(-)	date and time("2012-12-25T12:00:00-08:00") - date and time("2015-12-25T11:12:00Z")	duration("-P1094DT15H12M0S")	Returns a duration indicating the number of days and time between date and time of two different time zones.
(-)	date("2016-12-25") - duration("P3Y2M6D")	date("2013-10-19")	Returns the date 3 years 2 months and 6 days before 2016-12-25.
(/)	(date("2015-12-25") - date("2012-12-25"))/ duration("P1D")	1095	Returns the number of days between two dates.
(/)	(date and time("2015-12-25T17:00:00") - date and time("2015-12-25T09:12:00"))/ duration("PT1H")	7.8	Returns the number of hours between two date and time values.
(/)	(date("2015-12-25") - date("2015-12-24"))/ duration("P1Y")	0.00273972602739726 03	Returns the number of years between two dates.
None	years and months duration(date("2012-12-23") , date("2015-12-25"))	duration("-P3Y0M")	Returns the years and months duration between two dates.

Comparison Operation Examples

Examples in this section demonstrate comparison operations that can be performed on date or time data types.

Operator	Example	Result	Description
>	date("2012-12-25") > date("2015-12-25")	false	Determines if Date A occurs after Date B.
>	((date("2015-12-25") - date("2015-11-25")) > duration("P1Y")	false	Determines if Duration A is greater than Duration B.

2

Learn About Decisions in Oracle Integration

The Decision feature in Oracle Integration allows you to create decision models that make business workflows less complex, easier to manage, and more robust in the face of change.

Using the Decision feature in Oracle Integration, you can:

- Create decision models.
- Create decisions and sub-decisions in the models.
- Define the input data and type for decisions.
- Create associated services to use decision models in integrations.

Learn about the decision designer, workflow for using decisions in integrations, and best practices for modeling decisions.

Topics:

- How to Model Decisions in Oracle Integration?
- Understand the Decision Designer
- Workflow for Using Decision Models in Oracle Integration
- Best Practices for Modeling Decisions

How to Model Decisions in Oracle Integration?

In Oracle Integration, you can model decisions in two ways: within a project or externally.

Prerequisites

To use decisions in Oracle Integration, you must enable Process Automation for your Oracle Integration instance. Additionally, you must assign the predefined, Process Automation roles (namely, *ServiceDeveloper* or *ServiceAdministrator*) to the required users or groups so that they can access the Decision feature on your instance. For detailed instructions, see Enable Process Automation with Oracle Integration 3 in *Administering Oracle Cloud Infrastructure Process Automation*.

Ways to Use Decisions

After completing the prerequisites, you can model decisions in Oracle Integration within these two contexts:

- **Projects:** You can model decisions in the context of projects using the Decision feature. A project serves as the central hub for all your automation work, including designing decisions and integrations. Each project targets a specific business objective. It offers convenient deployment and unified observability, facilitating teamwork in building, deploying, and monitoring integrations and decisions. To learn more about projects, see Get Started with Projects in *Using Integrations in Oracle Integration 3*.
  Use project-based decisions when calling decisions from an integration or external client application. This guide focuses on how to model decisions in projects. See Design Decision Models.

- **Process:** You can create decision applications in Process. The Process feature in Oracle Integration enables you to rapidly design, automate, and manage business processes in the cloud.
  Use decision applications when calling decisions from a process application. To create decision applications, see Model Decisions in *Using Oracle Cloud Infrastructure Process Automation*.

Understand the Decision Designer

Learn about the components of the decision designer for building powerful decision models.

Using the decision designer, you can:

- Add decisions to a decision model.
- Define the input data and type for decisions.
- Configure the logic for decisions.
- Create services to use a decision model in integrations.

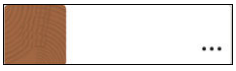
Topics:

- Decision Requirement Diagrams
- Decision Designer Components

Decision Requirement Diagrams

In the decision designer, you can create decision requirement diagrams (DRD)—in accordance with DMN standards—to visually represent your decisions.

Oracle Integration currently supports one DRD per decision. The following table lists all the available DRD components, which you can use to create your decision model.

DRD Component		Description	Icons
Elements	Decision	Denotes a node that determines an output based on its inputs and the logic it contains.	
	Input Data	Denotes information used as input by one or more decision elements.	
Requirements	Information Requirement	Denotes the flow of information from an input data or a decision element to another decision element.	

Decision Designer Components

The decision designer features a toolbar, sidebar, canvas, and overview panel to assist you in creating decisions.

- Toolbar
- Sidebar
- Canvas
- Canvas Overview

- A Snapshot of the Designer

Toolbar

The toolbar offers basic editing and designer-resizing controls.

Toolbar Icon	Name	Description
	Undo	Reverts the last action.
	Redo	Repeats the last action.
	Designer toggle	Expands or retracts the decision designer.

Sidebar

The sidebar offers controls to access different aspects of decision modeling. Each of these controls functions as a toggle, enabling you to view or hide the corresponding panel.

Sidebar Icon	Name	Description
	Palette	Opens the diagram palette with the input data element and all decision types. From the palette, drag and drop elements onto the canvas to use them in your diagram. See Add Decisions and Create Input Data.
	Services	Opens the Services panel, where you can create decision services. See Expose Decisions as Services.
	Types	Opens the Types panel, where you can define a new type definition. See Define Custom Data Types.
	Test decision model	Opens the Test Decision Model panel. See Test a Decision Model.
	Properties	Opens the Properties panel, where you can edit a node's properties. **Note:** This icon appears on the sidebar when you select an input or decision node on the canvas, and it opens the corresponding properties panel when clicked. See Add Decisions and Create Input Data.

Sidebar Icon	Name	Description
	View or edit decision logic	Opens the decision logic page, where you can view or edit a decision's logic. **Note:** This icon appears on the sidebar when you select a decision node on the canvas, and it opens the corresponding logic page when clicked. See Configure a Decision's Logic.

Canvas

The decision canvas is the central area where you can create a diagram that represents your decision model, using the elements in the palette.

Additionally, the canvas provides the following two functions:

Canvas Icon	Name	Description
	Search	Searches for a decision node on the canvas using the name you enter.
	Enable highlight mode toggle	When enabled, highlights the node you select on the canvas.

Canvas Overview

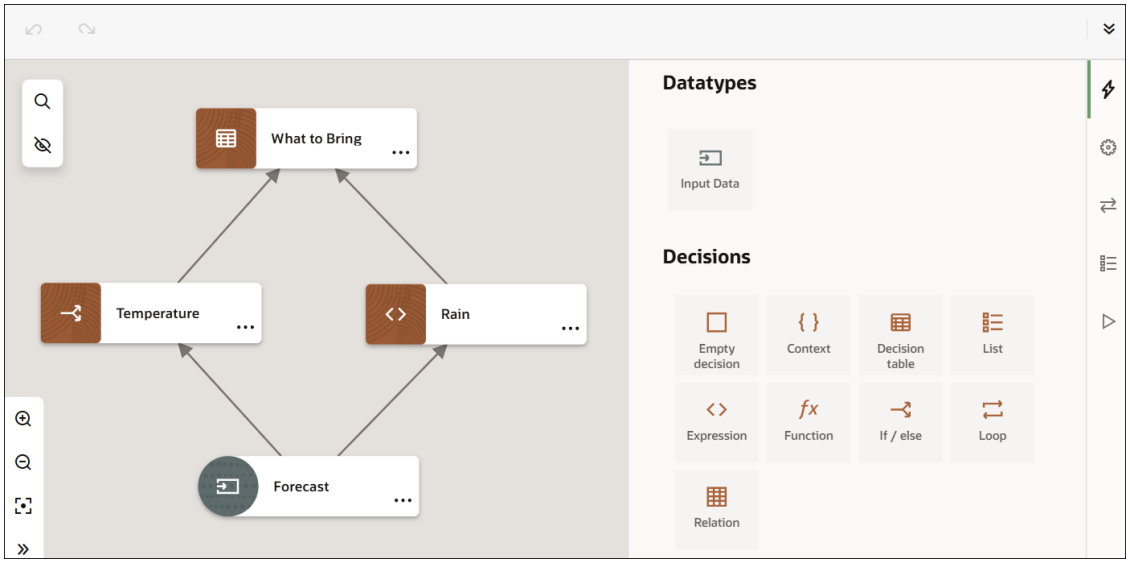
The canvas overview panel, located at the bottom-left corner, provides a quick overview of your entire canvas. You can collapse or expand it as needed.

It contains the following controls:

Canvas Overview Icon	Name	Description
	Zoom in	Zooms in on the canvas.
	Zoom out	Zooms out on the canvas.
	Reset view	Resets the canvas to its default view.
	Collapse overview toggle	Collapses or expands the Canvas Overview panel.

A Snapshot of the Designer

The following image shows a decision model in the designer, with the Palette panel open.



Workflow for Using Decision Models in Oracle Integration

Use the following approach as a general guideline to create and use a decision model in Oracle Integration. After creating a decision model, you must develop an integration to use it.

You may choose to complete some of these steps in any order. Iteratively refine your decision model as you build it to suit your requirements.

Workflow Table

Order	Step	More information
1	Create a project	To create a decision model, you must create a project in Oracle Integration. Alternatively, you can use an existing project. To create a new project, see Create or Import a Project in *Using Integrations in Oracle Integration 3*.
2	Create a decision model	Create a decision model, the container for decisions, input data, and decision services. See Create Decision Models.
3	Add decisions	Add decisions within your model. See Add Decisions.
4	Define input variables	Define the input data and type for your decisions. See Define Decision Input and Type.
5	Connect nodes on the canvas	Connect all nodes on the canvas to visually represent the decision flow. Connect Nodes on the Canvas.

Order	Step	More information
6	Configure the logic for decisions	Configure the logic for each decision node, defining how its output is derived from its input. You can use Friendly Enough Expression Language (FEEL) to define expressions across the decision model. See Configure a Decision's Logic. **Note:** Add and connect the nodes on the canvas before configuring the decision logic. This provides all the necessary data as suggestions during the configuration process.
7	Fix errors in decisions	Check each decision for errors and warnings, and resolve them. Review and Fix Errors in a Decision.
8	Test the decision model	Test the decision model and its decisions to ensure they work as expected. See Test a Decision Model.
9	Create decision services	Create decision services to complete the configuration of your decision model. See Expose Decisions as Services.
10	Activate the decision model	Activate the decision model to use it in an integration. See Activate a Decision Model.
11	Create an integration	Create an integration to call the decision model. See Create Integrations in *Using Integrations in Oracle Integration 3*.
12	Add the decision model to the integration	Using the decision services you created, add the decision model to the integration. See Add a Decision Model to an Integration.

Best Practices for Modeling Decisions

For optimal decision model development in Oracle Integration, adhere to these best practices and recommendations. They improve development, maintenance, and readability of decision models.

- Adopt a bottom-up approach to develop a decision model. Start by creating the lowest-level decisions, which provide input to other decisions. Finally, create the main output decision, which provides the model's result.
- Ensure that only one user edits a decision model at a time to avoid conflicts and preserve data integrity.
- Add and connect all the nodes on the canvas before configuring the logic within decisions. This provides all the necessary data as suggestions during the configuration process. See Connect Nodes on the Canvas.

- Use decision tables where possible; they're the preferred form of logic. See Configure Decision Tables.
- In decision tables, avoid using the First (F) hit policy. This hit policy makes the decision logic overly reliant on the order of the rules. See About Hit Policies.
- Use functions to apply a decision logic multiple times, for example, while applying a logic to each element of a list. See Configure Functions.
- Avoid using functions unnecessarily; they make decision models difficult to test and debug.
- Break down a complex, nested expression into simpler expressions.
- Use boxed expressions instead of FEEL expressions, wherever possible, to improve readability.
- Use nouns or short noun phrases to name each decision and input data (for example, Student Days, Total Days, Person). Do not use verbs in names.
- Provide a description for each decision to indicate what it accomplishes, for example, *Calculates total days by adding vacation and working days.* Make all descriptions consistent with each other.

3

Design Decision Models

Learn how to create a decision model, add decisions, define inputs, and configure logic for decisions.

Topics:

- Create Decision Models
- Add Decisions
- Define Decision Input and Type
- Connect Nodes on the Canvas
- Configure a Decision's Logic
- Review and Fix Errors in a Decision

Create Decision Models

A decision model is a container for decisions, input data, and decision services.

You can create a new decision model or import one from your local file system.

- Create a New Decision Model
- Import a Decision Model

> **Note:**
>
> To create a decision model, you must first create a project in Oracle Integration. Alternatively, you can use an existing project. To create a project, see Create or Import a Project in *Using Integrations in Oracle Integration 3*.

Create a New Decision Model

Create a decision model from scratch to use in your integrations.

1. Open a project.
   a. In the navigation pane, click **Projects**.
   b. Click a project's name.
2. In the left toolbar, click **Decision** .
3. Create a decision model.
   a. In the Decisions box, click **Add** (if no decision models have been created) or **+** (if one or more decision models have been created).
   b. In the Add decision panel, click **Create**.

c. In the Create decision panel, enter the following information for the decision model, and click **Create**.

Field	Description
Name	Enter a name for the decision model. Make sure the name conveys the purpose of the model. Note that you can't change the name after you create the decision model.
Identifier	Oracle Integration generates this value using the Name value.
Version	For a new decision model, the default version is 01.00.0000. You can revise the version number for subsequent updates.
Description	Provide additional information about the decision model.

The decision designer appears, with an **Input Data** node added to the canvas. See Understand the Decision Designer.

In the designer, you can:

- Add decisions
- Define input data
- Configure the logic for decisions
- Test your decision model
- Create decision services to implement your decision model

Import a Decision Model

You can import a decision model, saved as a DMN (`.dmnx`) file, into a project in Oracle Integration.

1. Open a project.
   a. In the navigation pane, click **Projects**.
   b. Click a project's name.
2. In the left toolbar, click **Decision**  .
3. Import a decision model.
   a. In the Decisions box, click **Add** (if no decision models have been created) or **+** (if one or more decision models have been created).
   b. In the Add decision panel, click **Import**.
   c. In the Import decision panel, click the field to browse for a file or drag and drop a file into the field. Click **Import**.

   The decision designer appears, where you can edit the imported model. See Understand the Decision Designer.

Add Decisions

A decision has logical notations, such as decision tables, expressions, if-then-else rules, and so on. Within your decision model, you can add a decision by selecting the logic type you plan to use to obtain a specific output.

Within decision models, decisions are classified into two categories:

- Main decision: The output of the main decision provides the result of the decision model.
- Supporting decision: One or more supporting decisions provide input to the main decision.

You can add a decision to your model by dragging one of the decision types from the Palette panel onto the canvas. After you add a decision, you define its properties and logic.

In the canvas, you can order decisions in several ways. For example, you can create a bottom-up sequence flow, with the main decision at the top and supporting decisions below it; or a left-to-right flow, and so on.

To add a decision to your model:

1. In your project, click **Decision** in the left toolbar to view your decision model.
2. In the Decisions box, point to the decision model you created, click •••, and click **Edit**.
3. In the decision designer, click **Palette** on the sidebar.
4. On the Palette panel, choose one of the following decision types under **Decisions**.
   - Empty decision
   - Context
   - Decision table
   - List
   - Expression
   - Function
   - If / else
   - Loop
   - Relation
5. Drag the decision type onto the canvas and drop it at the required spot.
6. In the Decision Properties panel that appears, enter the following information for the decision.

Field	Description
Name	Enter a unique name for the decision node. **Note:** You can also assign a decision node the same name as a previously-added decision node. Choose an existing name from the drop-down list next to the **Name** field. However, if you add a particular decision type to the canvas and select the name of a different type of decision node from the drop-down list, the node's logic type changes.
Edit	Click to edit the decision node's logic. You can also double-click the node on the canvas to view or edit its logic.

Field	Description
Delete	Click to delete either the decision node or both the node and its associated logic. Alternatively, click ••• on a decision node, and select **Delete** to remove it from the canvas.
Logic	Change the decision node's logic type if required. For example, you can add an empty decision to the canvas and change its logic from the properties panel. **Note:** If you select a previously-implemented decision node in the **Name** field and change the type in the **Logic** field, the content of the previously-implemented node is overwritten.
Description	Provide additional information, if any, for the decision node.
Question	Add questions for the decision node.
Allowed Answers	Add the possible answers for the decision node.

7. Click **Close**, or click to return to the Palette panel and add another decision.

   Changes you make within the decision model are automatically saved and validated.

Define Decision Input and Type

An input data variable is a placeholder for information that is to be supplied to a decision model when the model is invoked.

The supported data types for input data are text, number, boolean (true or false), date and time, and duration. See Data Types. You can extend the built-in data types to define custom, complex data types.

The following topics explain how to create input variables and custom data types.

- Create Input Data
- Define Custom Data Types

Create Input Data

Use built-in or custom data types to create input variables for a decision model.

To create an input data variable:

1. In the decision designer, click **Palette** on the sidebar.
2. On the Palette panel, select the **Input Data** element and drag it onto the canvas.
3. In the Input Properties panel that appears, enter the following information for the input data node.

Field	Description
Name	Enter a unique name for the input node. **Note:** You can also assign an input data node the same name as a previously-added node. Choose an existing name from the drop-down list next to the **Name** field. However, if you add a particular input data type to the canvas and select the name of a different type of input from the drop-down list, the input node's type changes.
Delete	Click to delete either the input node or both the node and its associated logic. Alternatively, click ••• on the input node, and select **Delete** to remove it from the canvas.
Mode	Specify the data type for the input node. The default is set to **Text**. You can select one of the built-in data types or choose a custom data type. To identify the data type as a list, select the **Make a list** check box. **Note:** If you select a previously-defined variable in the **Name** field: • The mode and other details are auto-populated. • If you change the type in the **Mode** field, the content of the previous-defined variable is overwritten.
Allowed Values	You can optionally define allowed values to restrict the input variable to enumerated values or ranges.
Type	If you select **Other Type** in the **Mode** field, select a previously-defined custom data type in this field. Click **Show type definition list** to view all custom data types defined in the decision model, or to create a new custom data type. See Define Custom Data Types.

4. Click **Close**, or click to return to the Palette panel and add another input node.

   Changes you make within the decision model are automatically saved and validated.

Define Custom Data Types

If built-in data types aren't suitable for an input node in your decision model, you can create a custom data type.

A custom data type can be a new complex data type or an alias of a built-in data type. Further, to create a complex data type, you can use a combination of built-in data types or other complex data types. While configuring an input node, you can assign a custom data type to it by selecting **Other Type** in the **Mode** field. See Create Input Data.

Further, you can define a custom data type in one of the following ways:

- Manually define each custom data type. See Construct a Data Type.

- Import a JSON sample or schema containing custom data types. See Import a JSON Schema.

Construct a Data Type

In the decision designer, you can create a custom data type from scratch.

1. Click **Types** on the sidebar.
2. In the Types panel, click **Create new type**.
3. In the New Type panel, enter the following information for the type definition.

Field	Description
Name	Enter a unique name for the type definition.
Mode	Specify the data type for the definition. The default is set to **Text**. You can select one of the built-in data types or choose to define a complex data type. To identify the data type as a list, select the **Make a list** check box.
Allowed Values	If you select the mode as **Text**, **Number**, **Date and Time**, or **Duration**, you can optionally define allowed values to restrict the data type definition to enumerated values or ranges.
Define Type Attributes	If you select the mode as **Complex**, define attributes within the type definition. See Define Attributes for a Complex Data Type.

The new data type is now displayed in the Types panel, with options to edit or delete it.

4. If required, repeat the steps to add more data types.

Define Attributes for a Complex Data Type

To define a complex data type, you must specify its attributes.

While defining a custom data type, if you select **Complex** in the **Mode** field, the Define Type Attributes section appears.

Define the attributes as follows.

1. Click in the Define Type Attributes section.

   A default attribute is added to the table.

2. Click the attribute to select it, and then click **Edit**.
3. In the Edit Type Attribute panel, enter the following information for the attribute.

Field	Description
Name	Enter a unique name for the attribute.
Mode	Specify the data type for the attribute. The default is set to **Text**. You can select one of the built-in data types or choose a custom data type. To identify the attribute as a list, select the **Make a list** check box.
Allowed Values	You can optionally define allowed values to restrict the attribute definition to enumerated values or ranges.
Type	If you select **Other Type** in the **Mode** field, select a previously-defined custom data type in this field.

4. Click **Back** to return to the type-definition panel.
5. To delete an attribute, click the attribute's row, and then click **Delete** .

Import a JSON Schema

To quickly create a custom data type, import a JSON sample or schema into your decision model.

> **Note:**
>
> There are certain constraints associated with importing JSON schema files. See JSON Schema Import Restrictions.

To import a JSON sample or schema containing a custom data type:

1. In the decision designer, click **Types** on the sidebar.
2. In the Types panel, click **Import type** .
3. From the drop-down menu, select one of the following options:
   - **Import from sample**: Select to import a JSON sample.
   - **Import from schema**: Select to import a JSON schema.
4. If you select **Import from sample**, perform these actions in the Create new type panel.
   a. Enter a unique name for the type definition.
   b. In the Sample section, enter a JSON sample containing the data type. Click **Next**.
   c. Review the generated schema, and click **Create**.
5. If you select **Import from schema**, perform these actions in the Create new type panel.
   a. Enter a unique name for the type definition.
   b. In the Schema section, enter a valid JSON schema containing the data type. Click **Create**.

The new data type is now displayed in the Types panel, with options to edit or delete it.

JSON Schema Import Restrictions

Review these restrictions before importing a JSON schema to create custom data types in your decision model.

- The size of the JSON schema to import cannot exceed 200 KB.
- The JSON schema cannot have fields that exceed 20 levels of nesting.
- The field names in the JSON schema must meet these criteria:
  - Start with a letter, underscore, or colon.
  - Contain only letters, digits, underscores, hyphens, or periods.
- Oracle Integration supports the creation of data types exclusively from the following keywords in a JSON schema:
  - type

- format
- items
- enum
- definitions
- properties
- $ref

Keywords not in the list won't work, for example, anyOf, allOf, oneOf, pattern, required, minLength, maxLength, and regex.

Connect Nodes on the Canvas

After adding decisions and defining input data, connect the nodes on the canvas to visually represent the decision flow.

An input node can connect to multiple decision nodes. Decision nodes can have one-to-many and many-to-one relationships between them.

To connect a node to another:

1. Click a node to display its connection arrow  .
2. Drag the arrow to the node you want to connect.
3. Repeat these steps to connect all nodes, establishing the complete node graph for your decision flow.
4. To delete a connection, double-click the arrow.

Configure a Decision's Logic

Configure the logic for a decision node by specifying how its output is derived from its input.

The following topics explain how to configure the logic for each decision type.

- Configure Empty Decisions
- Configure Expressions
- Configure Decision Tables
- Configure If / Else Statements
- Configure Functions
- Configure Lists
- Configure Contexts
- Configure Relations
- Configure Loops

Configure Empty Decisions

While planning your decision model, you can add empty decisions as placeholders.

To add an empty decision to the canvas, see Add Decisions.

Follow these steps to configure the decision's logic:

1. Click the **Empty decision** node on the canvas, and then click **Decision Properties** on the sidebar.
2. In the Decision Properties panel, update the **Logic** field to the required logic type.
3. Based on your selection, use the respective configuration procedure to define the logic. See Configure a Decision's Logic.

Configure Expressions

An expression is a logical notation, defined according to the syntax of FEEL, that evaluates to a single value. It may consist of one or more operands (such as literals, constants, or variables) and zero or more operators.

In Oracle Integration, you can use input variables, outputs of other decisions, or built-in functions to define an expression.

To add an expression decision to the canvas and define its properties, see Add Decisions.

Follow these steps to configure the decision's logic:

1. Double-click the expression node to access its logic editor.

   Alternatively, select the node, and click <> on the sidebar.
2. Click the **Enter Expression** field to view a suggestion list, containing decision outputs, variables, functions, and keywords.
3. Define an expression using the suggestions or write your own. Use the FEEL syntax. See About Friendly Enough Expression Language.
4. To reuse the entire logic definition in another decision, cut or copy it and paste it into the desired decision. Click ••• in the header and select the required action.

Changes you make within the decision model are automatically saved and validated. Errors and warnings, if any, are displayed in the editor. Click the error or warning icon to review and fix them. See Review and Fix Errors in a Decision.

Here are a few examples of simple expressions:

- This expression evaluates an applicant's age. If the *age* property of the variable *applicant* is less than 70, the output of the decision is *true*, else it is *false*.

- This expression calculates the area of a circle using a constant and an input variable, *radius*.

Configure Decision Tables

Decision tables are the notation of choice to model complex logic. Their tabular layout helps you effectively document all the possible conditions and results of a problem.

To add a decision table to the canvas and define its properties, see Add Decisions.

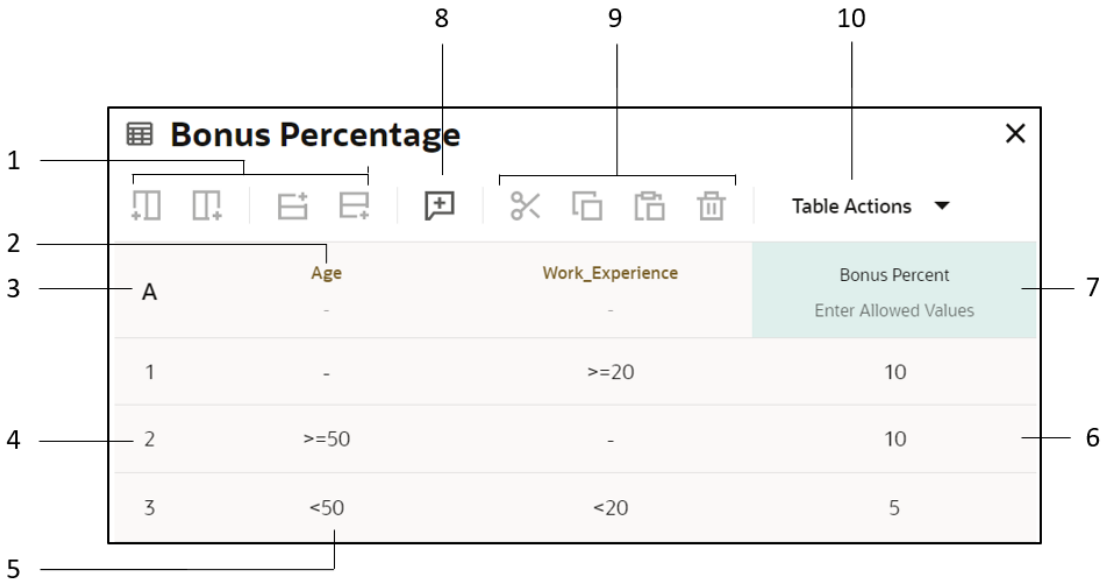
To learn about different components of a decision table and how to configure them, see the following topics:

- About Decision Table Elements
- Specify a Decision Table's Logic

About Decision Table Elements

Familiarize yourself with all the elements that constitute a decision table.

The following figure shows an example decision table with all its elements noted:



1. **Row and column controls:** Add rows and columns.
2. **Input header cell:** Contains the expression associated with a particular input column, and lets you specify the allowed values for the column.
3. **Hit policy cell:** Displays the hit policy selected for the table.
4. **Rule:** A row within a table.
5. **Input entry cell:** Contains an input entry.

6. **Output entry cell:** Contains an output entry.
7. **Output header cell:** Contains the name of the output column, and lets you specify the allowed values for the column.
8. **Add Annotation button:** Adds a column for documenting or annotating decision rules. Annotations aren't considered as part of the decision logic; they serve as explanatory notes for designers.
9. **Cut, Copy, Paste, and Delete buttons:** Cut, copy, paste, or delete rows and columns within a decision table. However, you cannot:
   - Copy a row into a column, and vice versa.
   - Copy an input column into an output column or an annotation column, and vice versa.
10. **Table Actions menu:** Contains options to copy and paste an entire decision table.

Learn about each element in detail and how to use them to configure different parts of a decision table. See the following topics:

- About Decision Table Input
- About Decision Table Output
- About Rules
- About Hit Policies

About Decision Table Input

An input (also referred to as input clause) to a decision table is represented as a column within the table. It consists of an input header cell and several input entry cells. A decision table may have multiple inputs.

- Input Header Cell
- Input Entry Cells

Input Header Cell

The input header cell contains the following two components:

Input expression

In combination with input entries, an input expression determines the value of a particular input column. It can be a simple test expression, for example, *Age>50*. You can use input variables, outputs of other decisions, or built-in functions to define input expressions.

In the decision designer, the expression language used for all expressions, including input expressions, is friendly enough expression language (FEEL). See About Friendly Enough Expression Language.

Allowed values

Using this component, you can specify the input entry mode (or type) and the associated allowed-value constraints for an input column. Click the **Mode** icon in a header cell to select a mode or specify allowed values.

However, for an input column, the mode of allowed values (or specific allowed values) are automatically populated when you specify the input expression for the column. For example, if an input expression returns Boolean values, allowed values for that column are populated as *true*, *false*.

The options you can choose from for changing the mode of allowed values are listed in the following table. Note that, after you specify the input expression for a column, the type of allowed values you can toggle between are limited.

Header Cell Modes	Description
Auto	Use this mode to determine allowed entries for the column based on the input expression. This is the default selection for a new input column.
Text	Use this mode to restrict entries to text strings. Optionally, you can specify a particular string or list of strings.
Number	Use this mode to restrict entries to numbers. Optionally, you can specify a particular number, limit, or range.
Date and Time	Use this mode to restrict entries to date and time values. Optionally, you can specify a particular date and time value, limit, or range.
Duration	Use this mode to restrict entries to time duration values.
True or False	Use this mode to permit only Boolean entries.
Other Type	Use this mode to restrict entries to a given custom data type.
Any	Use this mode to specify that there are no restrictions on the data type of entries. Optionally, you can restrict entries to a given value or list of values. This is the default selection for an output column.
Advanced	Use this mode to permit FEEL expressions and *null* values as entries. Optionally, you can specify constraints using FEEL expressions.

> **Note:**
>
> The data type of input entry cells is determined by the data type of the input expression. Make sure that the type of allowed values you supply is consistent with the data type of the input column.

Input Entry Cells

In the input entry cells, you enter strings, numbers, Boolean values, date and time values, and so on, based on the mode selected. If your input expression returns a finite set of values or if you've provided specific allowed values, an auto-suggest menu appears when you click an empty input entry cell. Use the suggestions to specify the input entries.

> **Note:**
>
> If the data type of an input entry does not match the data type of the column, or if the input entry is not among the allowed values, an error is displayed in the decision table editor.

Mode for input entries

Based on the input expression or allowed values you specify in the header cell, the mode for entry cells is automatically selected. If required, click the **Mode** icon in a cell to switch to a

different mode. The mode options available to switch are dependent on the data type of the input expression or the allowed values you've specified.

The following table details all available modes for an input entry cell. Optionally, use the constraint options available in the Mode editor.

Entry Cell Modes	Description
Text	Use this mode to enter strings. In this mode, you can enter a string as a plain literal without double quotes.
Number	Use this mode to enter numbers.
Date and Time	Use this mode to enter date and time values.
Duration	Use this mode to enter time duration values.
True or False	Use this mode to enter Boolean values.
Any	Use this mode to mark an entry as irrelevant (-).
Advanced	Use this mode to enter advanced FEEL expressions and *null* values. To learn more about the syntax and examples, see Grammar Rules.

About Decision Table Output

An output (also referred to as output clause) of a decision table is represented as a column within the table. It consists of an output header cell and several output entry cells. A decision table may have multiple outputs.

When you create a new decision table, a table with a single output column appears. To add additional outputs, click the header cell of the existing output column, and click **Add Column After**.

- Output Header Cell
- Output Entry Cells

Output Header Cell

The output header cell contains the following two components:

Output label

A decision table's output column is initially titled **Output**. You can change it to a name of your choice.

Allowed values

Using this component, you can specify the output entry mode (or type) and the associated allowed-value constraints for an output column. Click the **Mode** icon in a header cell to select a mode or specify allowed values.

The modes available in the output header cell are identical to those available in the input header cell. See Input Header Cell.

Output Entry Cells

In the output entry cells, you enter the decision table's results for different combinations of input data.

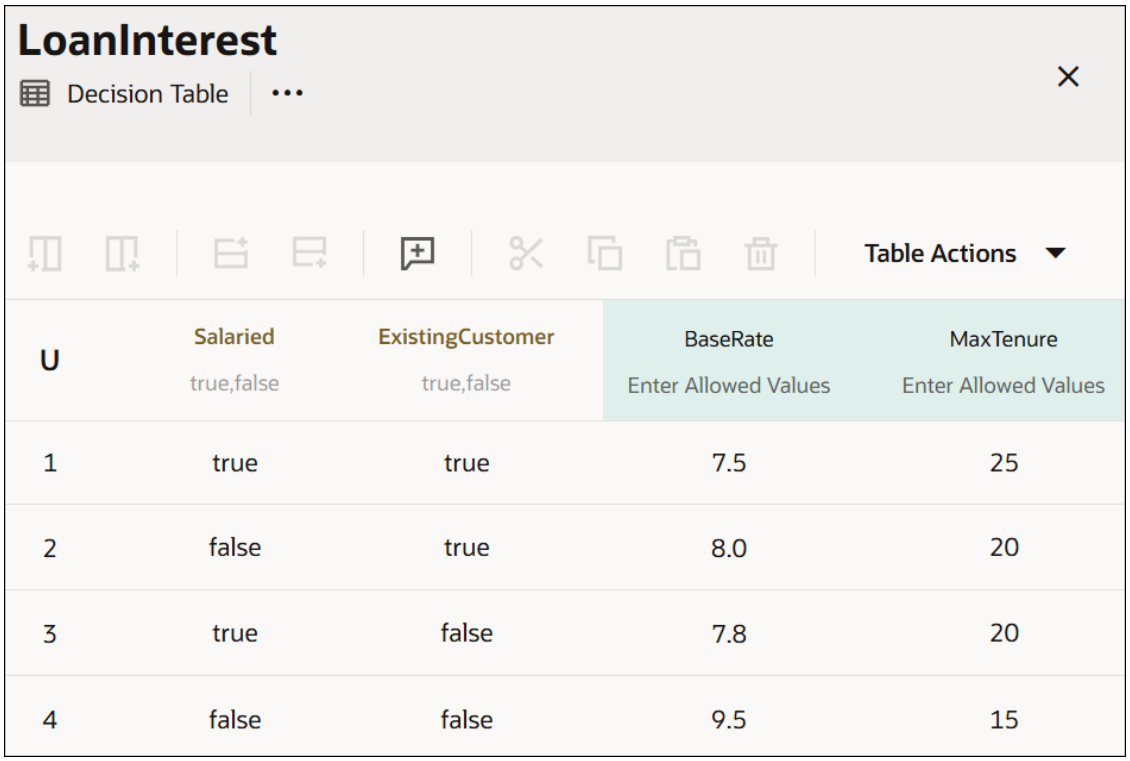
Based on the mode and allowed values you specify in the header cell, an auto-suggest menu appears when you click an empty output entry cell. Use the suggestions to specify the output entries.

Mode for output entries

Based on the mode or allowed values you specify in the output header cell, the mode for entry cells is automatically selected. If required, click the **Mode** icon in a cell to switch to a different mode. The mode options available to switch are dependent on the allowed values you've specified.

An output entry cell offers the same modes as an input entry cell, excluding the **Any** mode. For all available modes, see Input Entry Cells. Optionally, use the constraint options available in the Mode editor.

The following figure shows an example decision table with two output columns. It determines the home loan interest rates and maximum loan tenures based on whether the customer is salaried and an existing client of the bank.



To reference a particular output of this multi-output table from another decision, use the following format: *DecisionName.OutputLabel*; for example, *LoanInterest.BaseRate*.

About Rules

Rules are expressed as rows within a decision table. Every rule consists of one or more input entries and a corresponding output entry.

Generally, a decision table consists of multiple rules. When the input data to the decision table matches the input entries of a rule, the table's result contains the output entry of the rule.

About Hit Policies

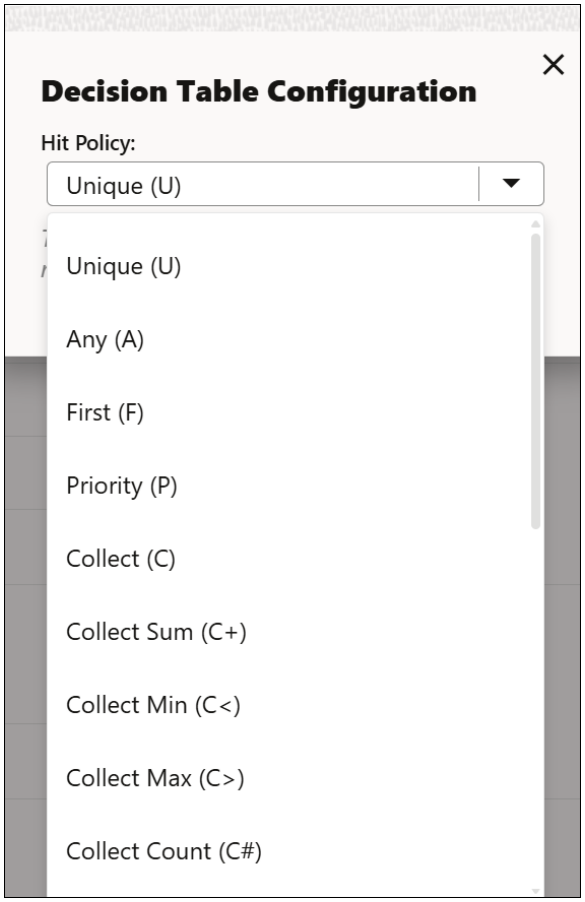
The hit policy of a decision table determines the table's output from the output entry cells of matched rules. A rule is matched when the input data to the decision table matches the input entries of a rule.

The hit policy cell displays the selected policy for the table. To locate the hit policy cell, see About Decision Table Elements.

Based on the hit policy, decision tables are classified into the following categories:

1. **Single Hit**: A single-hit table returns the output of only one rule. In the single-hit category, Oracle Integration supports the following hit policies:
   - **Unique (U)**: Only one of the rules can match.
   - **Any (A)**: Multiple rules can match, but all matching rules must have the same output.
   - **First (F)**: Multiple rules can match; the output of the first rule that matches is returned.
   - **Priority (P)**: Multiple rules can match; the output value that has the highest priority is returned.
2. **Multiple Hit**: A multiple-hit table returns the output of multiple rules. In this category, Oracle Integration supports the following hit policies:
   - **Collect (C)**: Multiple rules can match; outputs are returned in a list.
   - **Collect Sum (C+)**: Multiple rules can match; the sum of outputs is returned.
   - **Collect Min (C<)**: Multiple rules can match; the smallest output value is returned.
   - **Collect Max (C>)**: Multiple rules can match; the largest output value is returned.
   - **Collect Count (C#)**: Multiple rules can match; the count is returned.

When you create a new table, the Unique (U) hit policy is selected by default. To change the policy, click the hit policy cell and choose from the available options in the **Hit Policy** drop-down list.

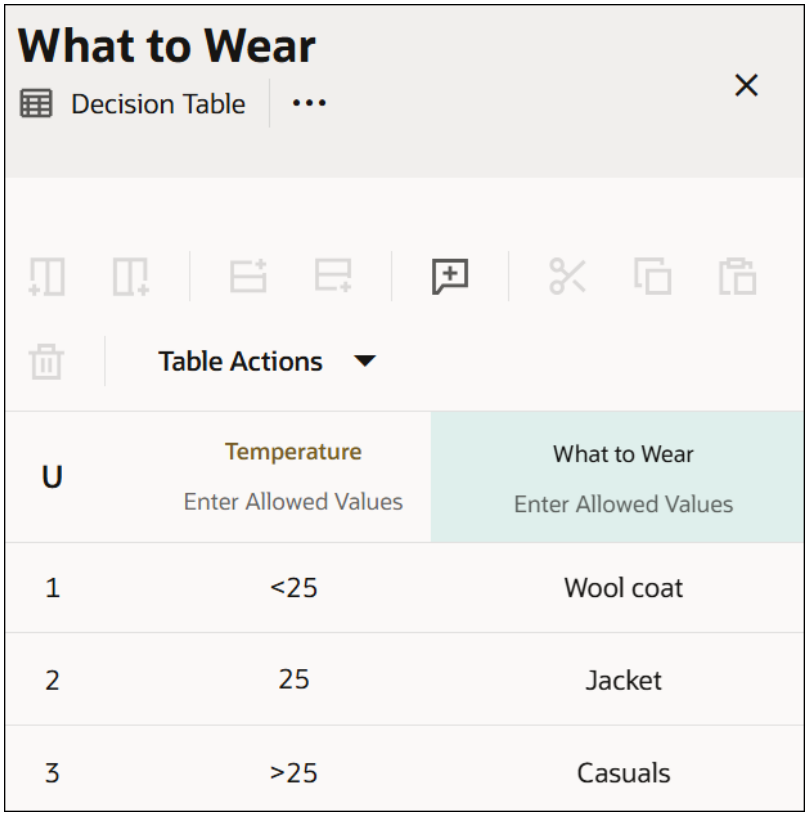


> **Note:**
>
> If rules within the table do not conform to the selected hit policy, a warning is displayed in the decision table editor.

Hit Policy Examples

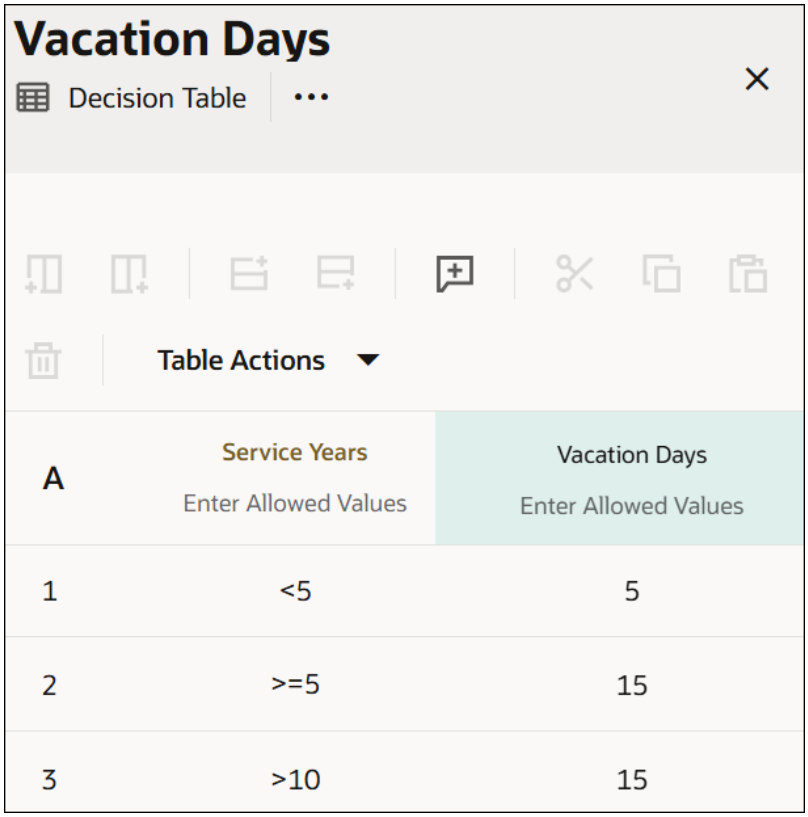
- **Single Hit Unique**
  In a decision table with Unique hit policy, only one rule can match. All rules are independent of each other, and no overlap is permitted. The decision table returns the output of the rule that matches.

  Here is a decision table created with the Unique hit policy. In this example, for any input value of temperature, only one rule can match.



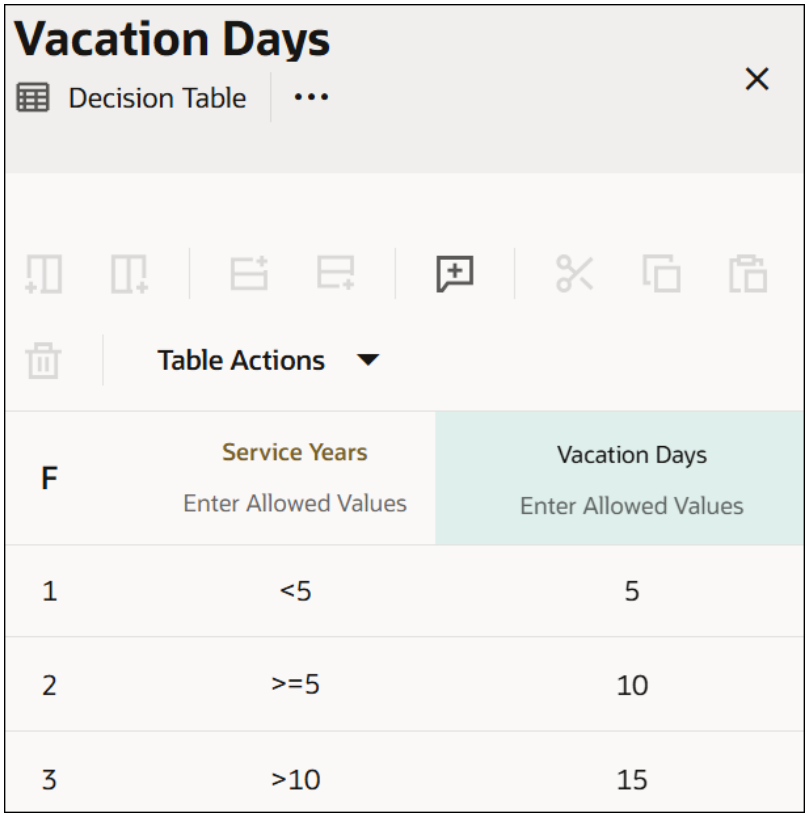
- **Single Hit Any**
  In a decision table with Any hit policy, multiple rules can match. Overlaps are permitted only if the matching rules have the same output. The decision table returns the output of any one of the matching rules. The hit policy is breached if matching rules have different outputs.

  Here is a decision table created with the Any hit policy. In this example, for service years of 11, the second and third rules match. This overlap is allowed because these rules have the same output. The decision table returns the output of any one of these rules.



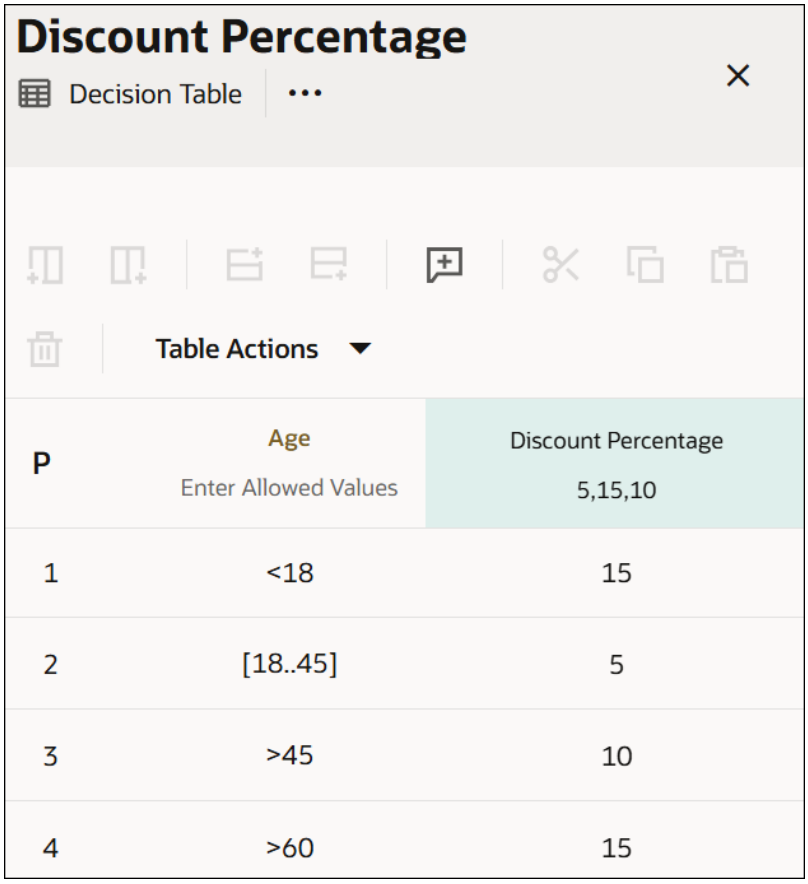
- **Single Hit First**
  In a decision table with First hit policy, multiple rules with different output entries can match. The output of the lowest-numbered matching rule is the result of the table.

  Here is a decision table created with the First hit policy. In this example, for service years of 11, the second and third rules match. The decision table returns only the second rule's output.



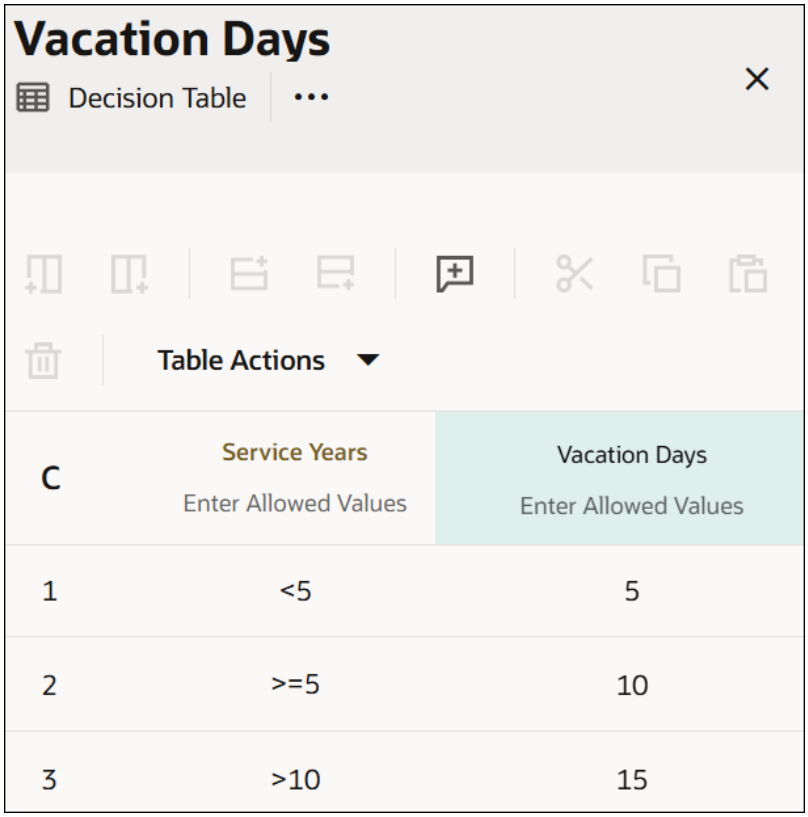
- **Single Hit Priority**
  In a decision table with Priority hit policy, multiple rules with different output entries can match. The priority of output values (in descending order) is specified as a list in the Allowed Values cell of the output column. The decision table returns the output value that has the highest priority among outputs of all matching rules.

  Here is a decision table created with the Priority hit policy. In this example, the last two rules match for an input age of 61. The decision table returns the output value that has the highest priority among these rules, that is, 15. The priority order is defined in the Allowed Values cell.



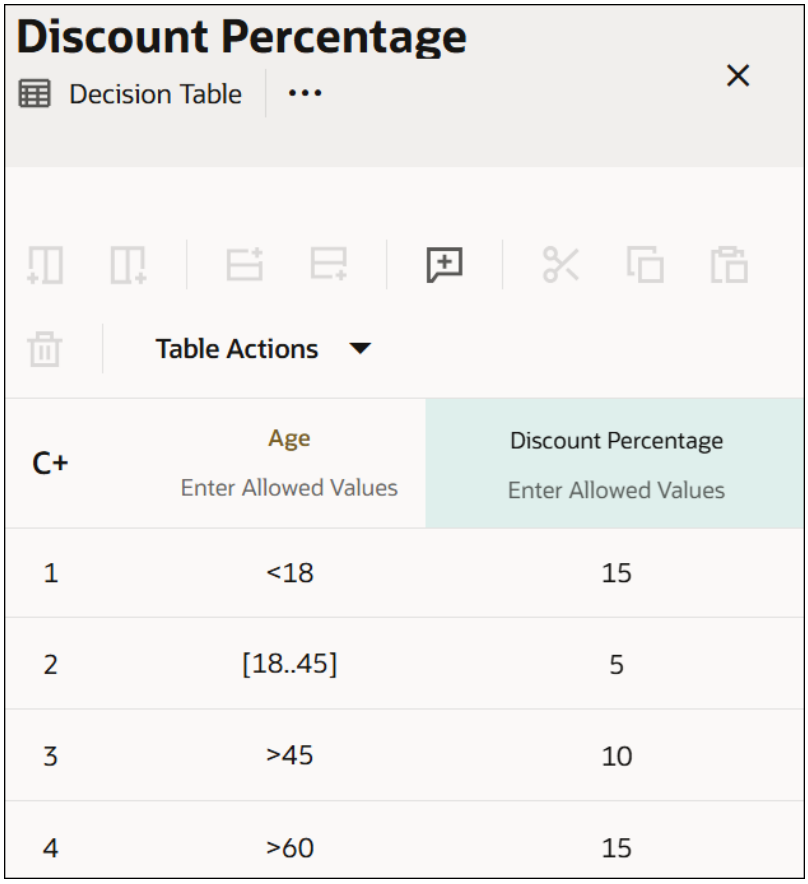
- **Multiple Hit Collect**
  In a decision table with Collect hit policy, multiple rules with different output entries can match. The decision table returns outputs of all matching rules in a list.

  Here is a decision table created with the Collect hit policy. In this example, two rules match for service years of 11. The decision table returns output values of these rules in a list, that is, 10 and 15.



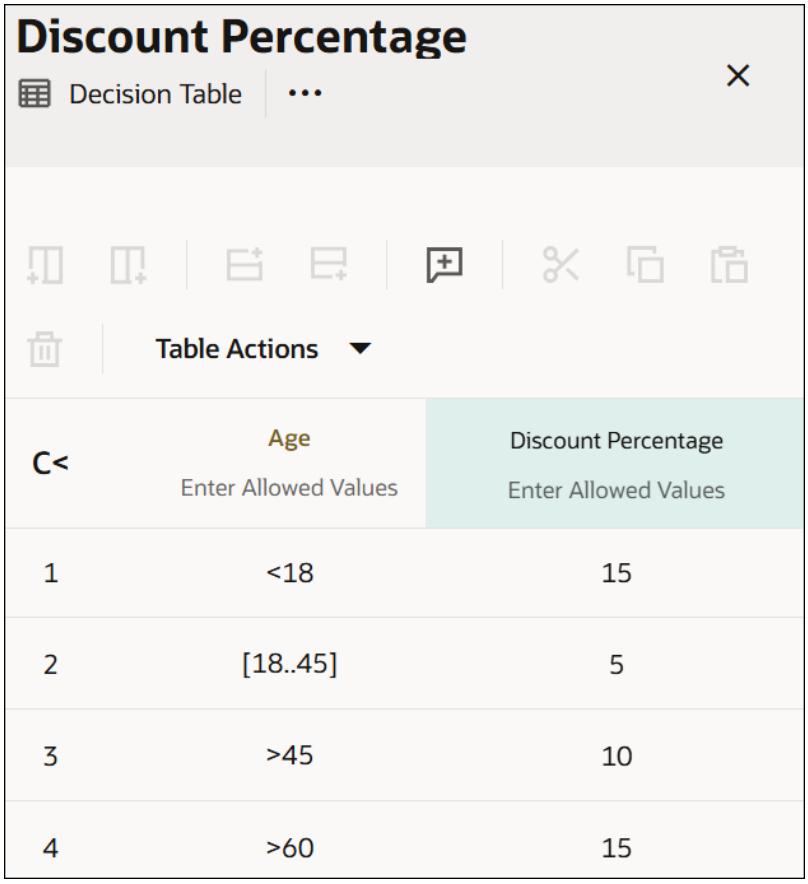
- **Multiple Hit Collect (Sum)**
  In a decision table with Collect (Sum) hit policy, multiple rules with different output entries can match. The decision table returns the sum of outputs of all matching rules.

  Here is a decision table created with the Collect (Sum) hit policy. In this example, the last two rules match for an input age of 61. The decision table returns the sum of output values of these rules, that is, 25.



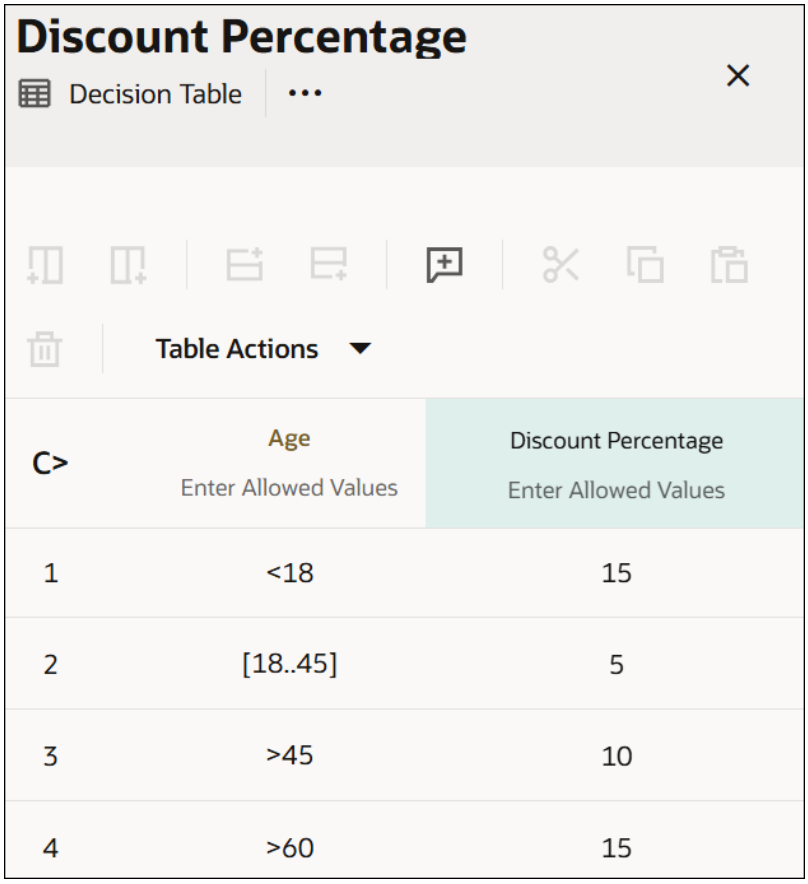
- **Multiple Hit Collect (Min)**
  In a decision table with Collect (Min) hit policy, multiple rules with different output entries can match. The decision table returns the smallest output value among all matching rules.

  Here is a decision table created with the Collect (Min) hit policy. In this example, the last two rules match for an input age of 61. The decision table returns the smallest output value among these rules, that is, 10.



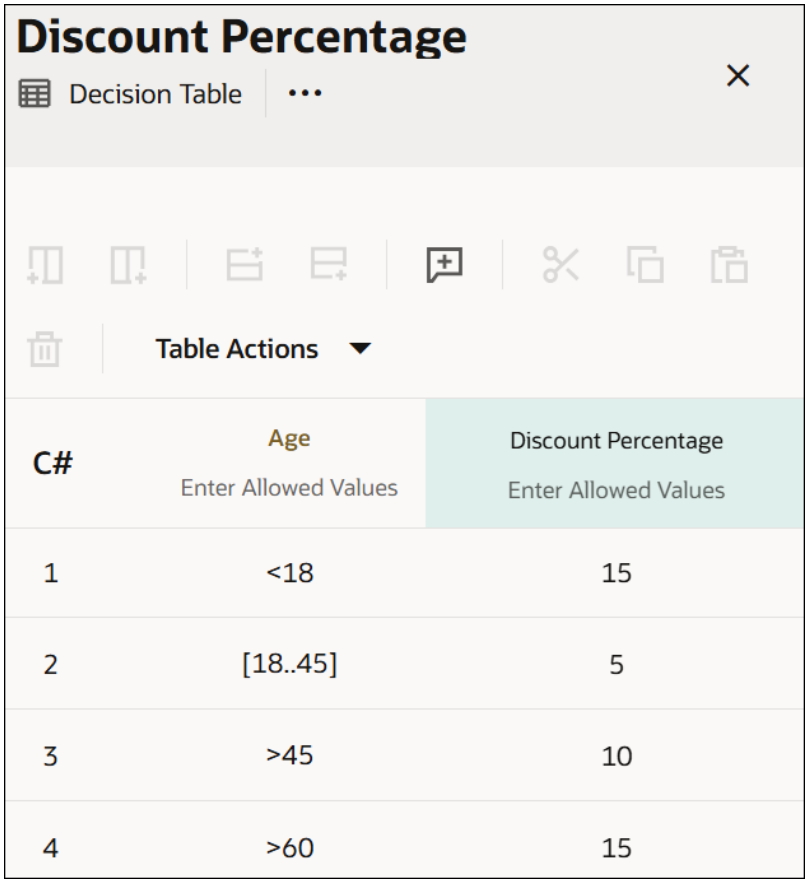
- **Multiple Hit Collect (Max)**
  In a decision table with Collect (Max) hit policy, multiple rules with different output entries can match. The decision table returns the largest output value among all matching rules.

  Here is a decision table created with the Collect (Max) hit policy. In this example, the last two rules match for an input age of 61. The decision table returns the largest output value among these rules, that is, 15.



- **Multiple Hit Collect (Count)**
  In a decision table with Collect (Count) hit policy, multiple rules with different output entries can match. The decision table returns the count of matching rules.

  Here is a decision table created with the Collect (Count) hit policy. In this example, the last two rules match for an input age of 61. The decision table returns the count of matching rules, that is, 2.



Specify a Decision Table's Logic

Follow these steps to configure logic for a decision table.

1. Double-click a decision table node on the canvas to access its logic editor.

   Alternatively, select the node, and click <> on the sidebar.
2. Specify the hit policy for the table.
   a. Click the hit policy cell.
   b. In the dialog box that appears, select the required policy. Click **Close**.
3. Configure expressions for the input columns.
   a. In the first input column, click the **Enter Expression** field to view a suggestion list, containing decision outputs, input variables, functions, and keywords.
   b. Define the expression for the column using the suggestions or write your own.

      The Allowed Value cell is automatically populated based on the input expression.
   c. Add additional input columns if necessary, and repeat the steps to enter expressions for all input columns.

      To add a new input column, click the header cell of an existing input column and then click **Add Column After**.
4. Optionally, enter a name for the output column. It is initially titled **Output**. You can change it to a name of your choice.

5. Configure allowed values for the output column. Initially, the *mode* for allowed values is set to **Any**.
   a. In the output column's header cell, click **Any**.
   b. In the dialog box that appears, perform the following actions:
      i. In the **Mode** field, select the desired allowed value type.
      ii. Indicate whether the allowed values should be a single value, a list, or a range.
      iii. Specify the permitted values for cell entries of the output column.
      iv. Click **Close**.
6. Add and configure rules (that is, rows of the table).
   a. In row 1, double-click the first input entry cell, and select a value from the suggestion list that appears.

      The suggestion list contains allowed values for the column.
   b. Similarly, enter values for all cells of the table.
   c. To add additional rules (rows), click **Add Row After**.
7. For other actions that you can perform in the decision table editor, see About Decision Table Elements.

Changes you make within the decision model are automatically saved and validated. Errors and warnings, if any, are displayed in the editor. Click the error or warning icon to review and fix them. See Review and Fix Errors in a Decision.

Configure If / Else Statements

An If / else expression is a logical notation that evaluates a test statement. It executes a primary expression if the test is *true* and a secondary expression if the test is not *true*. You can also introduce additional test statements using the **Add Else If** option.

To add an if / else decision to the canvas and define its properties, see Add Decisions.

Follow these steps to configure the decision's logic:

1. Double-click the if-else node to access its logic editor.

   Alternatively, select the node, and click < > on the sidebar.
2. In the **if** expression field, enter a test expression. Click the field to view a suggestion list, containing decision outputs, variables, functions, and keywords. You can use these suggestions to define expressions or write your own using the FEEL syntax. See About Friendly Enough Expression Language.
3. Configure the logic for the **then** and **else** fields. These fields have the expression notation selected by default.
   a. To change the logical notation for a field, click ••• in its row, select **Change Value**, then select a different notation from the available options.
   b. Configure the logic for the selected notation.
4. Optionally, add additional test statements.
   a. Click **Add Else If**.

      New fields labeled **else if** and **then** are added above the **else** field.

b. Configure the logic for the new fields. You can change the logical notation for the newly-added **then** field.

c. To delete the **else if** field, click ••• in its row, and click **Delete**.

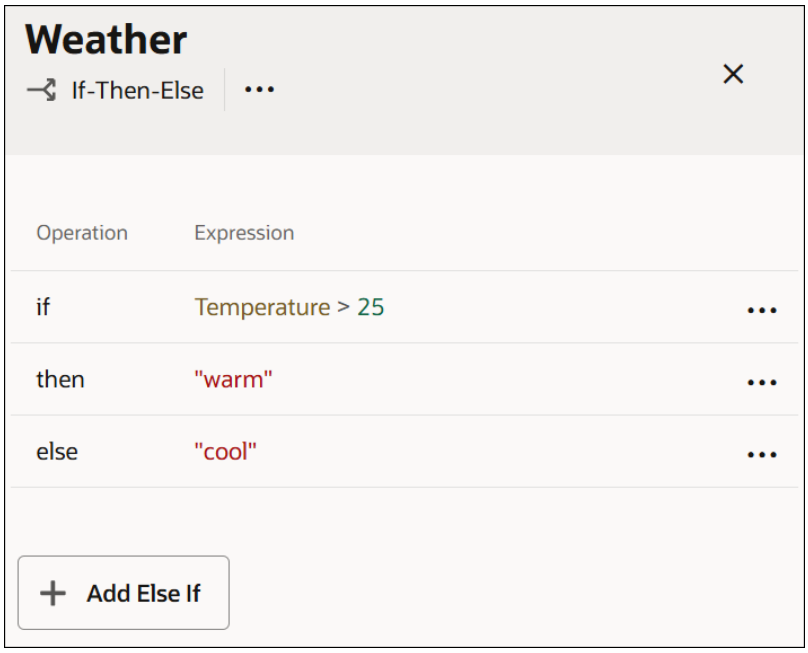
5. To copy and paste data to or from a field, click ••• in a field and select the required action.

6. To reuse the entire logic definition in another decision, cut or copy it and paste it into the desired decision. Click ••• in the header and select the required action.

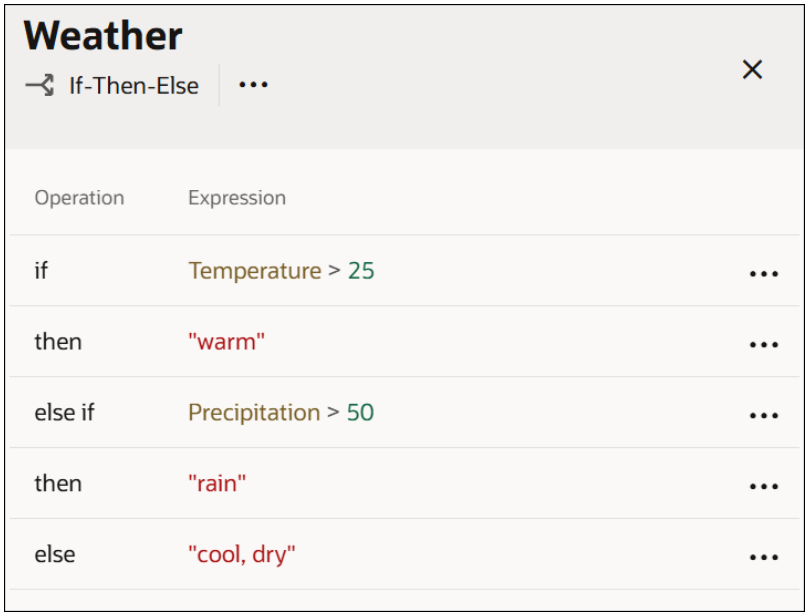
Changes you make within the decision model are automatically saved and validated. Errors and warnings, if any, are displayed in the editor. Click the error or warning icon to review and fix them. See Review and Fix Errors in a Decision.

Here are a few examples of If / else decisions:

- In the following example, the input value of temperature determines the output of the If / else decision:



- The following example uses an additional test statement, *precipitation > 50*, through the **else if** field to determine the final output:



Configure Functions

You can create functions to define specific operations that aren't available through built-in functions. In Oracle Integration, decisions created using the function notation return a value only when invoked from another decision.

To successfully invoke a function from another decision, the number and type of parameters in the function invocation must match those in the function definition.

> **Note:**
>
> Because a function decision by itself doesn't return a result, it's not an output decision. Therefore, you can't add function decisions to a decision service.

To add a function decision to the canvas and define its properties, see Add Decisions.

Follow these steps to configure the decision's logic:

1. Double-click the function node to access its logic editor.

   Alternatively, select the node, and click < > on the sidebar.
2. Add parameters to the function.
   a. Click **Parameters ( )**, and then click **Add Parameter** +.
   b. Enter a name for the parameter and select a data type for it. See Data Types.
   c. Similarly, add as many parameters to your function as needed.
3. In the **Body** field, define the function's logic using the parameters you added.
   a. Select the required notation for the **Body** field. Click •••, select **Change Value**, then select a notation from the available options.

      The field has the expression notation selected by default.
   b. Configure the logic for the selected notation using the parameters defined.

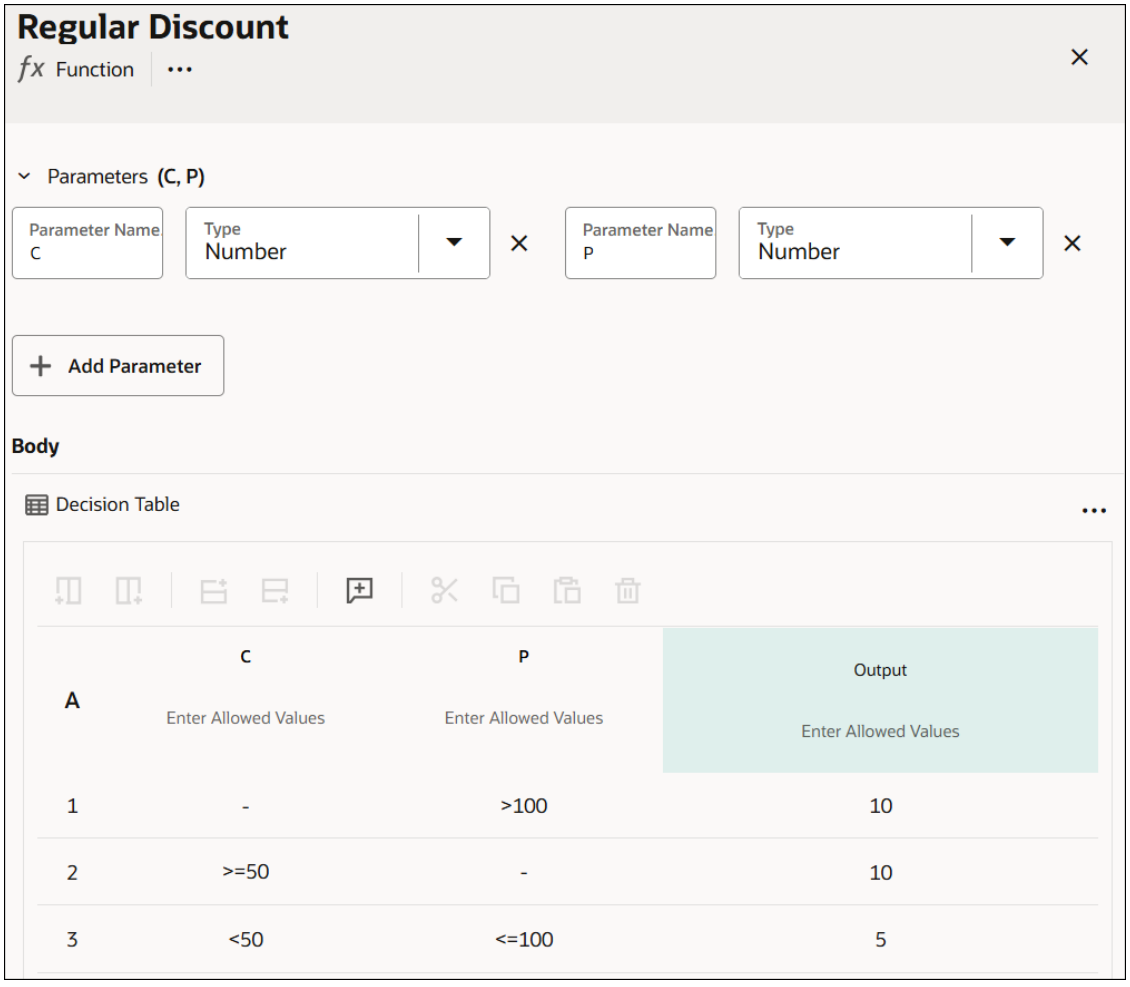
c. Additionally, you can move or replicate the **Body** field's contents between decisions. Click ••• and select the required action.

4. To reuse the entire logic definition (including parameters) in another decision, cut or copy it and paste it into the desired decision. Click ••• in the header and select the required action.

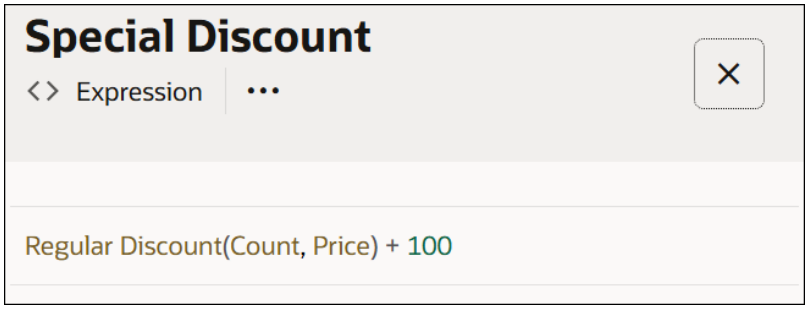
Changes you make within the decision model are automatically saved and validated. Errors and warnings, if any, are displayed in the editor. Click the error or warning icon to review and fix them. See Review and Fix Errors in a Decision.

The following example demonstrates a function implementation in Oracle Integration. Here, the function decision contains the logic for regular discounts in the form of a decision table. An output decision invokes the function decision to calculate the special discount percentage.

A function decision



An output decision invoking the function



Configure Lists

A list notation is a vertical list of elements, where each element is an independent logical notation. The output of a list notation contains outputs of all its elements. You can also invoke the output of a particular list element from another decision.

To add a list decision to the canvas and define its properties, see Add Decisions.

Follow these steps to configure the decision's logic:

1. Double-click the list node to access its logic editor.

   Alternatively, select the node, and click <> on the sidebar.

2. Click **Add item** + to create a new list item.

   An item is created with the expression notation selected by default.

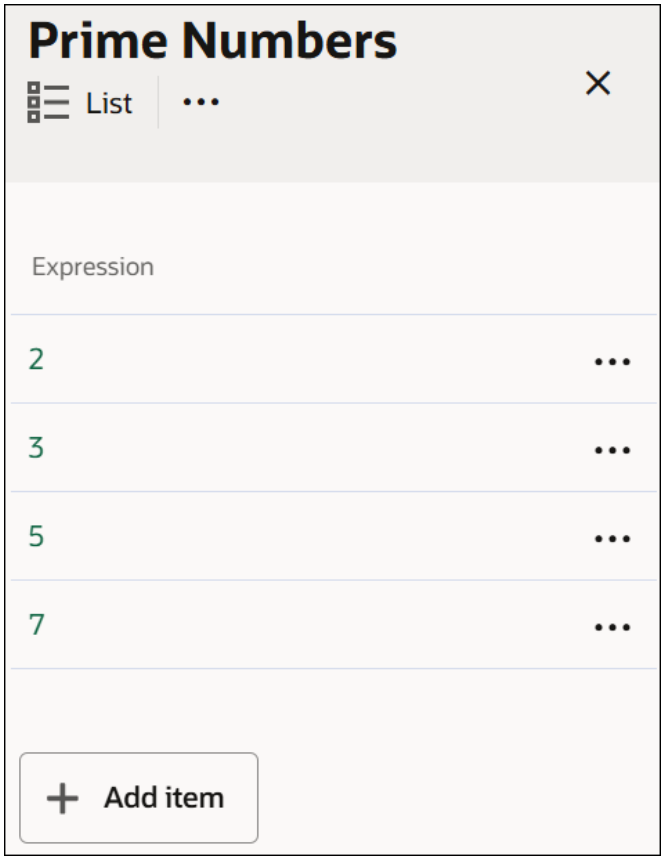
3. To change the logical notation for the list item, click •••, select **Change Value**, then select a different notation from the available options.

   > **Note:**
   >
   > If you add a function as one of the list items, the list notation as a whole doesn't return an output. However, you can invoke results of individual list items throughout the decision model.

4. Configure the logic for the list item according to the selected notation. You can use input variables or built-in functions to define the logic.
5. Similarly, add as many list items to your decision as needed.
6. Additionally, you can move or replicate a logical notation and its contents between list items. Click ••• in a list item's row and select the required action.
7. To reuse the entire logic definition in another decision, cut or copy it and paste it into the desired decision. Click ••• in the header and select the required action.
8. To delete a list item, click ••• in its row and click **Delete**.

Changes you make within the decision model are automatically saved and validated. Errors and warnings, if any, are displayed in the editor. Click the error or warning icon to review and fix them. See Review and Fix Errors in a Decision.

The following example is a list of simple expressions, containing prime numbers that are less than 10.



According to the FEEL syntax, you can also define horizontal lists in *expression* fields across all decision types. For example, a list of all prime numbers less than 10 can be defined as: [2,3,5,7].

In a list of *n* elements, use *<list_name>[n]* to invoke the n^{th} element from the beginning of the list, and use *<list_name>[-n]* to invoke the n^{th} element from the end of the list. In this example, to invoke the list entry of 2, you can either use *Prime Numbers[1]* or *Prime Numbers[-4]*.

Additionally, you can use built-in functions on a list decision within other decisions. For example, the following expression decision returns the sum of all items in the *Prime Numbers* decision.

Configure Contexts

A context is a collection of one or more key-value pairs with an optional result field. Each pair is called a context entry. The key attribute within a context entry acts as an identifier to its corresponding value attribute.

You can use a context to collectively document all decision logic related to a particular scenario or entity. Say you need to determine the loan eligibility of an applicant, based on the applicant's net monthly income and expense. For this purpose, you can create a decision named *Loan Eligibility* using the context notation and add expressions or logic for gross monthly income, monthly expense, and net monthly income. Then, you can add a result field (within the context) that evaluates the net income and expense for the loan eligibility, or you can choose to evaluate these within another decision.

Without a result field, a context decision returns multiple key-value pairs as output. In this case, you can invoke any context entry from another decision. If you add a result field, the output of this field is displayed as the context's output. Here, you can only invoke the context's result from another decision.

To add a context decision to the canvas and define its properties, see Add Decisions.

Follow these steps to configure the decision's logic:

1. Double-click the context node to access its logic editor.

   Alternatively, select the node, and click <> on the sidebar.
2. Add entries to the context.
   a. Click **Add Entry** to create a new context entry. A key-value pair is created.
   b. In the **Key** field, enter a unique name or label.
   c. Configure the logic for the **Expression** (value) field. This field has the expression notation selected by default.
      i. To change the logical notation for the value field, click ••• in its row, select **Change Value**, then select a different notation from the available options.

      > **Note:**
      >
      > If you add a function as one of the context entries, the context as a whole doesn't return a result. However, you can invoke results of individual context entries throughout the decision model.

      ii. Configure the logic for the selected notation. You can use input variables or built-in functions to define the logic.
   d. Similarly, add as many entries to your context as needed.
   e. Use the **Drag** ::: control to reorder the context entries.
3. Optionally, add a result for the context.
   a. Click **Add Result**.

      A result field is created with the expression notation selected by default.
   b. To change the logical notation for the result field, click ••• in its row, click **Change Value**, then select a different notation from the available options.

c. Configure the logic for the selected notation.

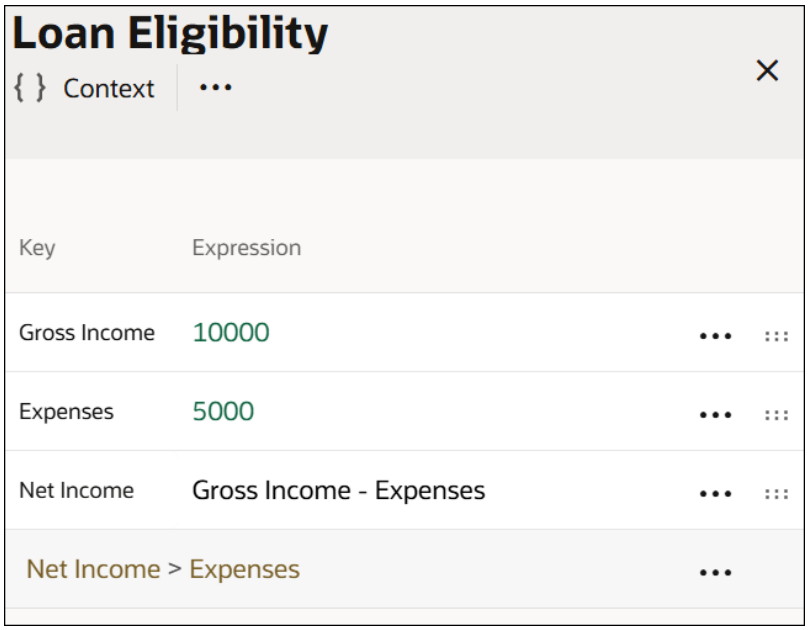
4. Additionally, you can move or replicate a logical notation and its contents between fields. Click ••• in a field and select the required action.
5. To reuse the entire logic definition in another decision, cut or copy it and paste it into the desired decision. Click ••• in the header and select the required action.
6. To delete a field, click ••• in its row and click **Delete**.

Changes you make within the decision model are automatically saved and validated. Errors and warnings, if any, are displayed in the editor. Click the error or warning icon to review and fix them. See Review and Fix Errors in a Decision.

A context decision with a result

The following image shows a context with a result field that determines the loan eligibility of applicants.

The output of the result field is the context's output. In this case, the context returns a true or false regarding an applicant's loan eligibility. You can reference the context's result in other decisions within the model using the context name (for example, *Loan Eligibility*).

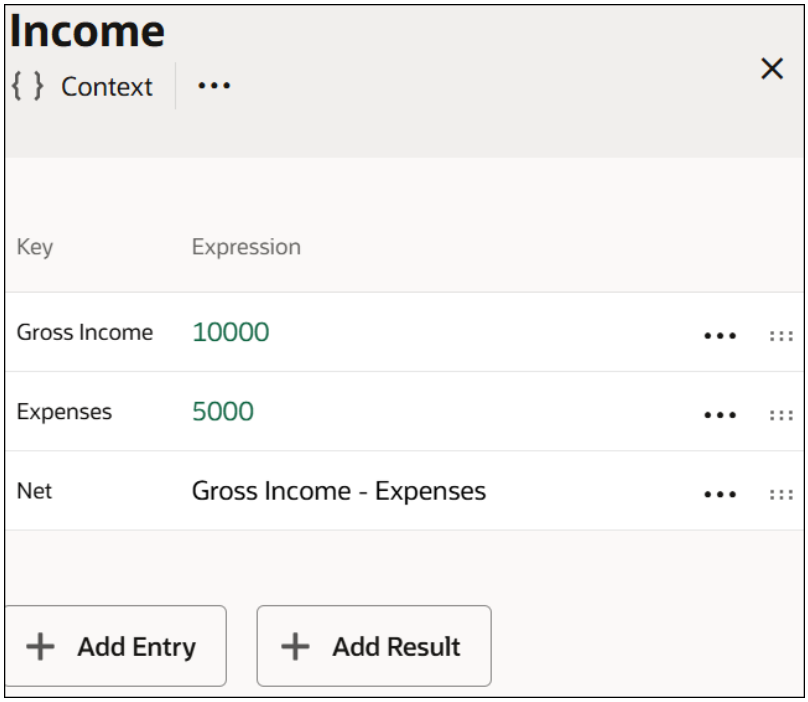


A context decision without a result

The following image shows a context without a result field and an expression decision referencing multiple context entries to determine the loan eligibility of applicants.

The output of this context is a list containing results of all three context entries. To reference a particular context entry from another decision, use the format *ContextName.EntryKey* (for example, *Income.Expenses*). Within a context, an entry can only reference entries that are above it.

- A context without a result:



- An output decision calling context entries:

Configure Relations

You can use a relation notation as a convenient shorthand to represent multiple contexts.

A relation decision is a list of similar contexts in a pivoted or transposed layout. In other words, each column name is the common *key* attribute for all cell entries under it, which essentially are the *value* attributes. For details about contexts and *key-value* pairs, see Configure Contexts. In a relation decision, each cell entry is an independent logical notation.

In the output of a relation decision, outputs of all contexts within it are clearly distinguished. You can also invoke the output of a particular context or context entry from another decision.

To add a relation decision to the canvas and define its properties, see Add Decisions.

Follow these steps to configure the decision's logic:

1. Double-click the relation node to access its logic editor.

   Alternatively, select the node, and click <> on the sidebar.

2. Click **Add Column After** or **Add Row After** to add additional rows or columns.

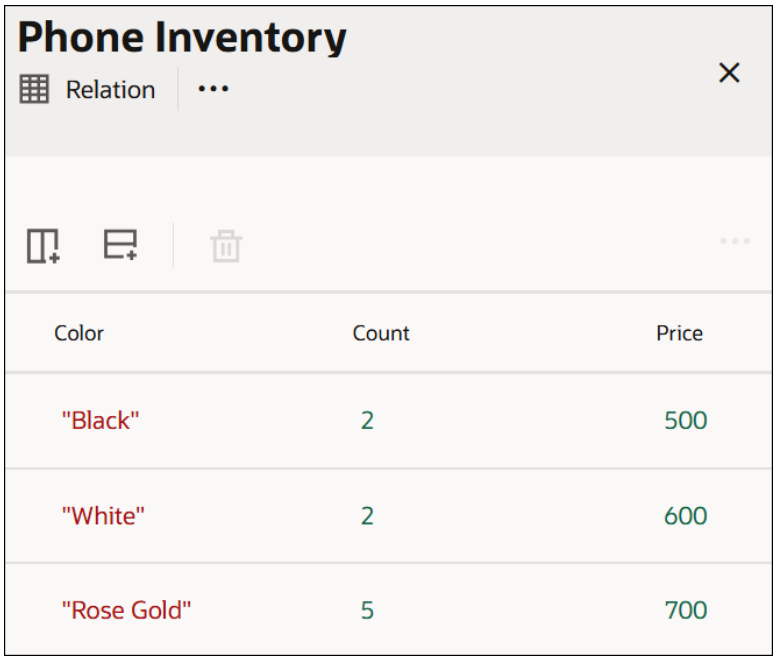
3. Name each column in the header row.
4. Enter the data in the cells of the rows below. All cells have the expression notation selected by default.
   a. To change the logical notation for a cell, click the cell, click ••• above the table on the right, then **Change Value**, and then select a different notation from the available options.

      > **Note:**
      >
      > - You cannot insert a decision table within a relation.
      > - If you add a function as one of the cell entries, the relation as a whole doesn't return a result. However, you can invoke results of individual cell entries or contexts throughout the decision model.

   b. Configure the logic for the selected notation within each cell.
5. Additionally, you can move or replicate the data of an entire row to another row, or a column to another column. Click to select a row or column, click ••• above the table on the right, and select the required action.
6. To reuse the entire logic definition in another decision, cut or copy it and paste it into the desired decision. Click ••• in the header and select the required action.
7. To delete a row or column, click it, and then click **Delete** .

Changes you make within the decision model are automatically saved and validated. Errors and warnings, if any, are displayed in the editor. Click the error or warning icon to review and fix them. See Review and Fix Errors in a Decision.

The following image shows a relation decision that contains the inventory information for a phone brand:



Similar to list notations, use either *Phone Inventory[1]* or *Phone Inventory[-3]* to access the entire context related to black-colored phones. To access all cell entries of a particular column,

use the relation name in combination with the column name, for example, *Phone Inventory.Price* returns all entries of the Price column. To access a particular context entry (for example, “700”), use *Phone Inventory.Price[3]*, *Phone Inventory.Price[-1]*, or *Phone Inventory[Color="Rose Gold"].Price*.

The following image shows the result of the entire relation decision, which has each context listed separately:

Configure Loops

Create loops to iterate over lists or arrays. Using the loop logical notation, you can create three different types of loops, namely For, Some, and Every.

- **For**: Iterates over a list and returns an array or a list containing the results.
- **Some**: Checks if at least one list item satisfies the test condition defined by an expression and returns a Boolean value.
- **Every**: Checks if every list item satisfies the test condition defined by an expression and returns a Boolean value.

To add a loop decision to the canvas and define its properties, see Add Decisions.

Follow these steps to configure the decision's logic:

1. Double-click the loop node to access its logic editor.

   Alternatively, select the node, and click < > on the sidebar.

2. Configure a For loop.

   a. In the **Operation** column, choose **For** in the drop-down menu. In the corresponding expression field, enter the loop variable.

b. In the expression field of the **in** row, enter the list or array over which to iterate. Click the field to view a suggestion list, containing decision outputs, variables, functions, and keywords. You can use these suggestions to define expressions or write your own using the FEEL syntax. See About Friendly Enough Expression Language.

c. Configure the logic for the **return** field. The field has the expression notation selected by default.

   i. To change the logical notation for the field, click ••• in its row, select **Change Value**, then select a different notation from the available options.

   ii. Configure the logic that calculates and returns the result.

d. Optionally, add an additional condition.

   i. Click **Add Condition**.
   A new field labeled **where** is added above the **return** field.

   ii. Enter the expression for the **where** field.

   iii. To delete the **where** field, click ••• in its row, and click **Delete**.

3. Configure a Some or Every loop.

   a. In the **Operation** column, choose **Some** or **Every** from the drop-down menu. In the corresponding expression field, enter the loop variable.

   b. In the expression field of the **in** row, enter the list or array over which to iterate. Click the field to view a suggestion list, containing decision outputs, variables, functions, and keywords. You can use these suggestions to define expressions or write your own using the FEEL syntax. See About Friendly Enough Expression Language.

   c. Configure the logic for the **satisfies** field. The field has the expression notation selected by default.

      i. To change the logical notation for the field, click ••• in its row, select **Change Value**, then select a different notation from the available options.

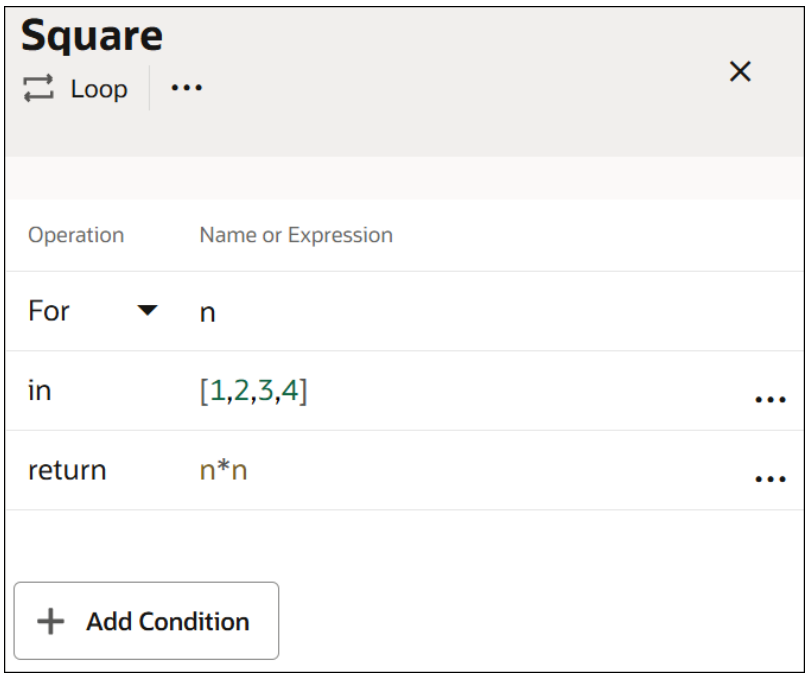
      ii. Enter a test expression or logic.

4. To copy and paste data to or from a field, click ••• in a field and select the required action.

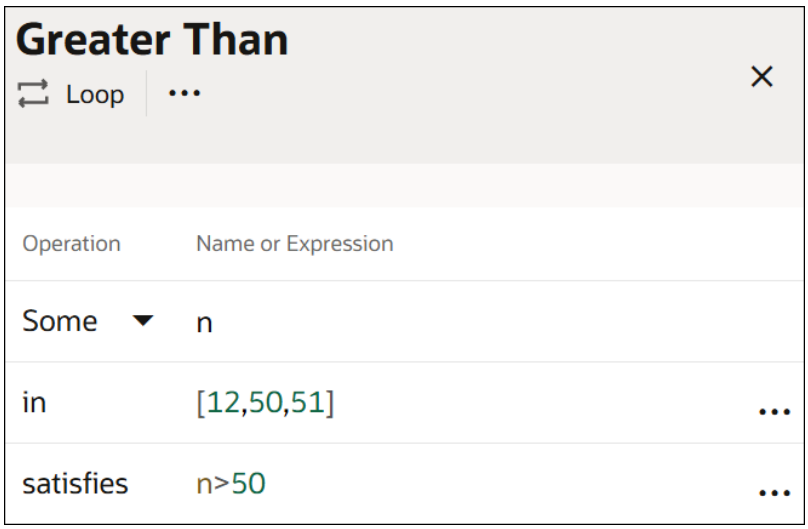
5. To reuse the entire logic definition in another decision, cut or copy it and paste it into the desired decision. Click ••• in the header and select the required action.

Changes you make within the decision model are automatically saved and validated. Errors and warnings, if any, are displayed in the editor. Click the error or warning icon to review and fix them. See Review and Fix Errors in a Decision.

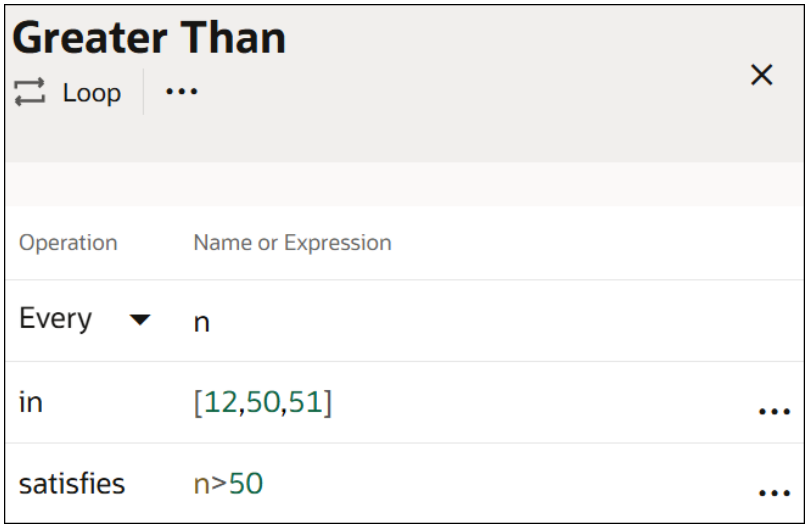
- The following example shows a simple For loop that returns the squares of list items:



- The following decision contains a Some loop that checks if at least one element in the list is greater than 50:



- The following decision contains an Every loop that checks if every element in the list is greater than 50:



Review and Fix Errors in a Decision

Check each decision in your decision model for errors and warnings. Resolve all issues before testing and activating the model.

If a decision node contains errors, an error icon appears on the node.

To view and resolve the errors or warnings in a decision:

1. Double-click the node to access its logic editor. Alternatively, select the node, and click </> on the sidebar.

   In the editor, the total number of errors and warnings are displayed in the top-right corner.

2. To view all errors and warnings, click the error or warning icon. Additionally, for decision tables, you can see where the errors or warnings occur in the table.

   Here's an image showing the errors and warnings for an empty decision table.

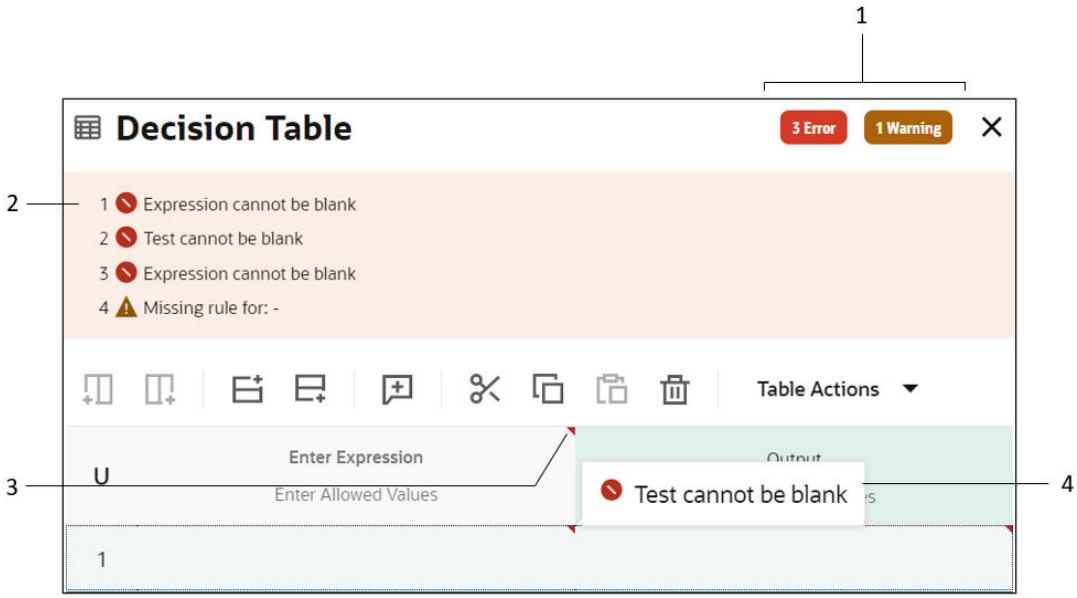


Image annotation details:

1: The error and warning icons, showing their respective counts.

2: The list of errors and warnings that appears when you click the error or warning icon.

3: The red marker that indicates an error in a specific cell of a decision table.

4: The error information that appears when you mouse over the red marker.

3. Review and fix all errors in each decision of your model. Afterward, you can test and activate the model.

4

Test and Activate Decision Models

After configuring a decision model, test and activate it. You can then add it to an integration.

Topics:

- Test a Decision Model
- Expose Decisions as Services
- Activate a Decision Model
- Add a Decision Model to an Integration

Test a Decision Model

Test your decision model to ensure it works as expected.

1. In the decision designer, click **Test Decision Model** ▷ on the sidebar.
2. In the Test Decision Model panel:
   a. Enter the input data manually.
   b. Or, use the **Mode** toggle to switch modes, and enter the input data as a JSON sample.
   c. Click **Start Test**.

   The panel displays the decision model's result.
3. To view the outcome of a specific decision in the model, click the decision's name on the left.
4. To repeat the test, click **Go Back**.

Expose Decisions as Services

To call your decision model in integrations, you must add at least one decision service in the model. A decision service exposes one or more decisions of your model as public REST APIs.

A decision service consists of a set of input data and a set of decisions from a decision model.

> **Note:**
>
> Make sure your decisions are error-free before creating a decision service. Errors in the decisions propagate to the service you create, and appear in the Services panel. To fix errors, see Review and Fix Errors in a Decision.

Follow these steps to add a service to your decision model.

1. In the decision designer, click **Services** ⇄ on the sidebar.

2. In the Services panel, click **Add new service** +.
3. In the Add Decision Service panel, enter a name for the decision service and click **OK**.

   The service is created and added to the Services panel.
4. Click the service's name to expand it and enter the necessary data.
   a. Click the **Output Decisions** field, and select the decision to expose through the service.
   b. Similarly, click the **Input Data** field, and select the input data to expose through the service.

   You can add multiple decisions and input data items to a service.
5. To delete a service, click ⋮ next to the service name and select **Delete**.

   Here's an example of a decision service's endpoint URL. You can view it in the integration designer's configuration wizard. See Add a Decision Model to an Integration.

   ```
   https://<base_url>/decision/api/v1/decision-models/<project_name>-
   <decision_model_name>/versions/<decision_model_version>/active/definition/
   decision-services/<decision_service_name>
   ```

   Where:

   - `base_url`: is the base URL of your Oracle Integration instance.
   - `project_name`: is the name of your project in Oracle Integration.
   - `decision_model_name`: is the name of your decision model.
   - `decision_model_version`: is the version of your decision model.
   - `decision_service_name`: is the name of your decision service.

Activate a Decision Model

After configuring a decision model, activate it to use it in an integration.

Prerequisites

Before you activate a decision model, complete the following tasks:

- Fix all errors in your decision model. See Review and Fix Errors in a Decision.
- Create at least one decision service. See Expose Decisions as Services.

> **Note:**
>
> After you complete these tasks, the decision model's state changes from **Draft** to **Configured**. You can verify this on the Decision page of your project. You can activate a decision model only when it is in the **Configured** state.

Activate a Decision Model

1. In your project, click **Decision** ▦ in the left toolbar to view the list of decision models.

2. In the Decisions box, point to the configured decision model you want to activate, click •••, and click **Activate**.

A notification message confirms the successful activation of the decision model. The model's state changes to **Active**.

> **Note:**
>
> Within a major version family (for example, 1.x.x), only one decision model can be active at a time. If version 1.0.0 is active and you activate version 1.0.1, the system automatically deactivates 1.0.0 and sets 1.0.1 as active. However, multiple versions of a model can be active simultaneously, provided they belong to different major-version families (for example, 2.x.x or 3.x.x). For more information on versioning, see Update a Decision Model.

Add a Decision Model to an Integration

After activating a decision model, add it to your integration at the appropriate point.

You must have created an integration in your project before executing the following steps. To create an integration, see Create Integrations in *Using Integrations in Oracle Integration 3*.

1. In your project, click **Integration** in the left toolbar to view the list of integrations.
2. In the Integrations box, point to the required integration, click •••, and click **Edit**.
3. In the integration designer, add the decision model at the required point in your integration.
   a. Click **Actions** to open the Actions panel.
   b. From the Call section on the panel, drag the **Decision** element and drop it onto the sign that appears when you hover over a connection arrow between elements.

   Alternatively, hover over a connection arrow, click the sign that appears, and select the **Decision** element from the dialog box.

   > **Note:**
   >
   > If you have not enabled Process Automation for your Oracle Integration instance, the **Decision** element is disabled in the integration designer. To enable Process Automation for your service instance, see How to Model Decisions in Oracle Integration?.

4. In the Decision Service Configuration Wizard that appears, configure the decision element.
   a. On the Basic Info page, provide a unique name and description for the element. Click **Continue**.
   b. On the Configuration page, enter the following data:
      i. Select the decision model to associate with the element.

> **Note:**
>
> A decision model's name appears in the options only if you have activated it previously.

ii. Specify the decision model version to use.

iii. Specify the decision service to use.

iv. Click **Continue**.

c. On the Summary page, review the data you have entered, and click **Finish**.

You've successfully added and configured a decision element. Now, a corresponding map element appears on the canvas. To continue configuring your integration, see Map Data in *Using the Oracle Mapper with Oracle Integration 3* and Create Application Integrations in *Using Integrations in Oracle Integration 3*.

5

Manage Decision Models

Effectively manage your existing decision models in your project. Version, clone, edit, or delete models to keep your workspace up-to-date and organized.

Topics:

- Update a Decision Model
- Clone a Decision Model
- Edit or View a Decision Model
- Deactivate a Decision Model
- Delete a Decision Model

Update a Decision Model

You can update a decision model for various purposes, according to the semantic versioning rules.

To understand semantic versionining, see Semantic Versioning Rules.

To ensure seamless integration, updating the minor or patch versions of a decision model does not disrupt the integration flow that references the model. The integration automatically incorporates these updates when you activate the updated model.

However, for a major version revision of a decision model, you must manually update the integration to use the new version.

> **Note:**
>
> After versioning a decision model, you must activate the new version for the integration to pick up the changes. There are limitations regarding the concurrent activation of versions. See Activate a Decision Model.

Versioning decision models serves the following purposes:

- **Change tracking**: Allows maintaining a history of changes to your decision model to review previous versions or revert when necessary.
- **Support for multiple scenarios**: Enables using different versions to address varying business automation scenarios.
- **Incremental development**: Facilitates updating your decision model incrementally as your business requirements evolve over time.

To version a decision model:

1. In your project, click **Decision** in the left toolbar to view the list of decision models.
2. In the Decisions box, point to the decision model you want to version, click •••, and click **Create new version**.

3. In the Create new version panel:
   a. Update the **Version** field with an appropriate number, and add a description if necessary.
   b. Click **Version**.

   A new version of the decision model is created and listed in the Decisions box.

Semantic Versioning Rules

Oracle enforces semantic versioning rules, also known as SemVer, for all decision models.

With semantic versioning, you increment the version numbers in a meaningful and controlled way. A version number provides a quick explanation of the types of changes that are in the version.

Oracle Integration supports the following types of updates to decision models.

Type of update	Example	Guidance
Major version revision	1.2.3 to 2.0.0	The change adds a major new feature and has the potential to break backward compatibility.
Minor version revision	1.2.3 to 1.3.0	The change adds minor new features while maintaining backward compatibility.
Patch version revision	1.2.3 to 1.2.4	The change improves quality while maintaining backward compatibility.

Clone a Decision Model

Clone a decision model to quickly create a base for another use case or to try out changes.

1. In your project, click **Decision** in the left toolbar to view the list of decision models.
2. In the Decisions box, point to the decision model you want to clone, click •••, and click **Clone**.
3. In the Clone decision panel:
   a. Enter a name and description for the model. Update the version if necessary.
   b. Click **Clone**.

   A new decision model is created and listed in the Decisions box.

Deactivate a Decision Model

In some cases, the system automatically deactivates decision models, but in other cases, you need to deactivate them manually when required.

Within a major version family (for example, 1.x.x), only one decision model can be active at a time. If version 1.0.0 is active and you activate version 1.0.1, the system automatically deactivates 1.0.0 and sets 1.0.1 as active.

However, you may require to manually deactivate a decision model in specific scenarios, such as:

- **Error mitigation**: In case of unforeseen errors, deactivation is a quick way to prevent a faulty model from impacting business processes. You can edit the model to correct errors and reactivate it.

- **Model replacement**: When a new, major version of a decision model is ready for production, you may need to retire the older version.
- **Maintenance**: During system maintenance or troubleshooting, you may pause a decision model to prevent errors from system unavailability.

To deactivate a decision model:

1. In your project, click **Decision** in the left toolbar to view the list of decision models.
2. In the Decisions box, point to the decision model you want to deactivate, click •••, and click **Deactivate**.
3. In the Deactivate decision dialog, click **Deactivate**.

A confirmation message appears, indicating that the decision model has been submitted for deactivation.

Edit or View a Decision Model

You can edit a decision model at any time when it's in the **Draft** or **Configured** state. However, a decision model in the **Active** state is view-only. To edit an active decision model, you must either deactivate or version it.

To edit or view a decision model:

1. In your project, click **Decision** in the left toolbar to view the list of decision models.
2. In the Decisions box, point to the decision model you want to edit or view, click •••, and click **Edit** or **View**.

   The decision designer opens in the edit or view mode, based on what you chose.

Delete a Decision Model

If you don't need a decision model anymore, you can delete it from your project.

You may require to delete a decision model in the following scenarios:

- **End of lifecycle**: Delete models that are outdated and no longer relevant to current business processes or technology.
- **Project organization and cleanup**: Clear out unused models to maintain a clean and efficient project workspace.
- **Presence of duplicate or incorrect models**: Delete incorrectly setup models or redundant models to optimize model management.

To delete a decision model:

1. In your project, click **Decision** in the left toolbar to view the list of decision models.
2. In the Decisions box, point to the decision model you want to delete, click •••, and click **Delete**.
3. In the Delete decision dialog, click **Delete**.

> **Note:**
>
> Deleting a decision model that is currently used in one or more integrations does not delete its references within those integrations. This may result in runtime errors.

6

Troubleshoot Decisions

Find solutions for problems with decision models, decision logic, or services.

Topics:

- The Option to Activate a Decision Model Is Not Displayed
- Unable to Add a Decision Model to an Integration
- An Input Used in a Decision Is Not Recognized

The Option to Activate a Decision Model Is Not Displayed

If you are unable to view the option to activate your decision model, you haven't fully configured it.

The Problem

On the Decision page of your project, when you click ••• on a decision model, the **Activate** option is not available.

Possible Causes

The decision model is in the **Draft** state due to either of the following reasons:

- There are validation errors in the model.
- No decision service exists in the model.

Solutions

Perform the following tasks to change the decision model's state from **Draft** to **Configured** and view the **Activate** option.

- Fix all errors in your decision model. See Review and Fix Errors in a Decision.
- Create at least one decision service. See Expose Decisions as Services.

Unable to Add a Decision Model to an Integration

If you are unable to add a decision model to an integration, you may have not activated the model.

The Problem

In the integration designer, when you add a **Decision** element at the required point in your integration, the Decision Service Configuration Wizard opens. In the wizard, the required decision model is not available to select.

Possible Cause

The decision model you want to add to your integration is not active.

Solution

Activate the decision model to make it available for selection in the configuration wizard of the integration designer. See Activate a Decision Model.

An Input Used in a Decision Is Not Recognized

If you encounter a **Name not found** error within a decision, you may be using an undefined input.

The Problem

Within a decision, when you use an input variable or an output of another decision in a field, the **Name not found** error is displayed.

Possible Causes

- You have not created the required input variable.
- You have not connected the nodes on the canvas to represent the decision flow.

Solutions

- Create the necessary variable or decision you intend to use as an input. See Create Input Data or Add Decisions.
- Connect the nodes on the canvas to establish the complete node graph for your decision flow. See Connect Nodes on the Canvas.