Corticon Basic Guided Journey

Table of Contents

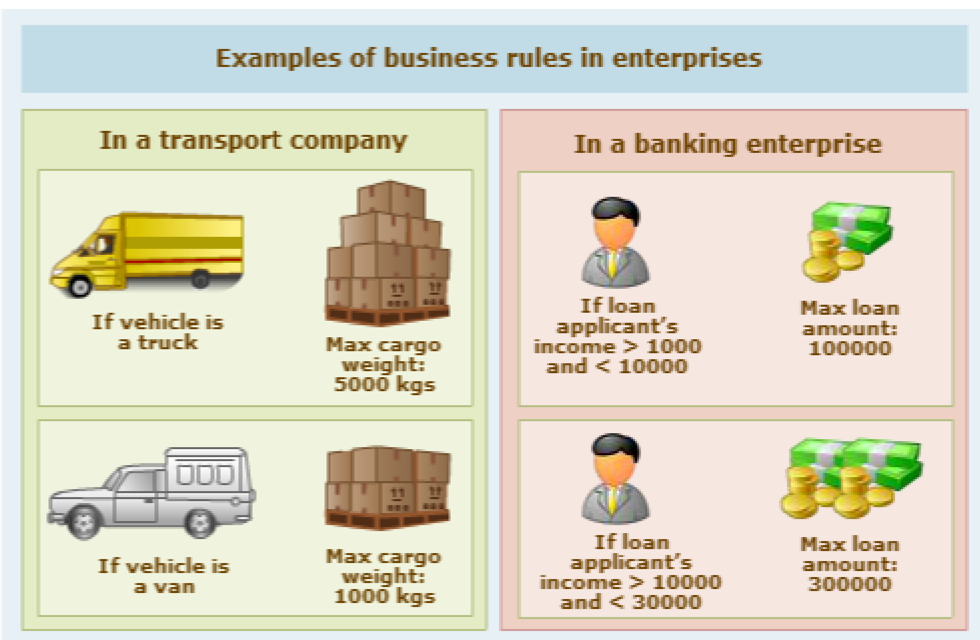
Copyright

Last updated: Corticon 7.2

1

Introduction to BRMS

Every enterprise has rules, policies, and regulations that govern different aspects of its business operations.



For example, a transport carrier company may have rules about how much cargo each of its vehicles can carry, or a bank may have rules about how much money it can lend to an individual.

In the past, enterprises implemented business rules manually, through employees. This approach tended to be inefficient, prone to errors, and lacked transparency. To eliminate these problems, enterprises began to automate business rules, enabling employees and systems to make business decisions more quickly and with fewer errors.

Before the advent of BRMS, automating business rules meant developing your own rules application, which was time consuming, resource-intensive, and costly. Often, the application would fall short of realizing the business analyst's requirements. Additionally, any changes required modifying and testing the application code.

A better approach is to use a BRMS such as Corticon that can enable enterprises to automate business rules quickly, transparently, and flexibly.

For details, see the following topics:

What is a BRMS?

A Business Rule Management System (BRMS) like Corticon enables enterprises to model, test, and automate complex business rules without writing code.

A BRMS has several advantages:

- **Increased agility**—Enterprises using a BRMS show greater responsiveness to change and reduced time-to-market in implementing new rules or modifying existing rules.
- **Reduced cost**—Automated decision-making is faster and more efficient.
- **Better quality**—Logical rule models reduce human errors, ensure that there is consistency in decision-making, and encourage the use of best practices.
- **Increased visibility into the application development process**—Requirements that are captured as rule models by business analysts become the implementation, which reduces surprises during user-acceptance testing.

Internal or external systems can not only access the business rules but also invoke them as part of an automated business process.

Introduction to Progress Corticon

Progress Corticon is a BRMS that enables enterprises to model, test, and automate business rules. It offers the following components:

- **Progress Corticon.js Studio**—The development environment to create and test rule models for JavaScript. Corticon.js Studio offers the ability to package rules as JavaScript to be run wherever JavaScript runs.
- **Progress Corticon Studio**—The development environment to create and test rule models for Java.
- **Progress Corticon Server**—The runtime environment that hosts deployed rule models (created in Corticon Studio), exposing them as Decision Services to the external world. Decision Services can be exposed as Web services (SOAP or REST), Java services, or .NET services.

Roles in the lifecycle of a Decision Service

There are two roles in the lifecycle of a Decision Service:

- **Rule modelers**—Create rule models in Corticon Studio or Corticon.js Studio depending on their deployment strategy. They define business terms, the rule logic, and the order of execution of rules.
- **Integration developers**—Package and deploy rules as Decision Services to Corticon Server or for JavaScript deployment and make them accessible to external systems.

This content provides an entry point into Corticon for rule modelers and integration developers.

Overview of Corticon Studio

Corticon Studio and Corticon.js are integrated development environments for developing business rules. They have the following features:

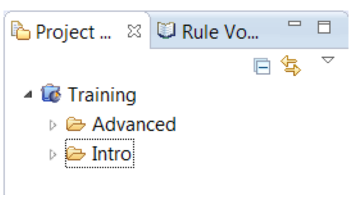
- Workspaces, projects, and folders to organize rule-modeling components
- Capabilities to create rule-modeling components
- Views and perspectives to see and open rule-modeling components
- Editors to develop and modify rule-modeling components

Start Corticon Studio by choosing **Progress > Corticon Studio**.

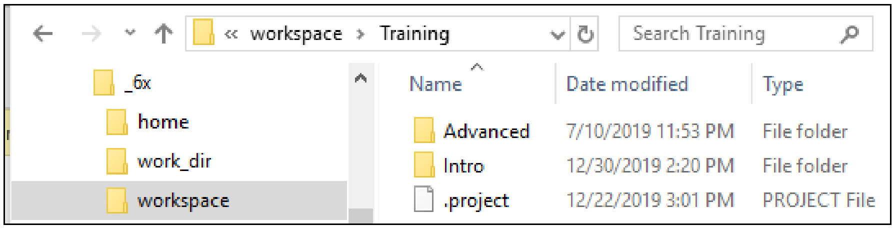
Start Corticon.js Studio by choosing **Progress > Corticon.js Studio**.

Workspaces, projects, and folders

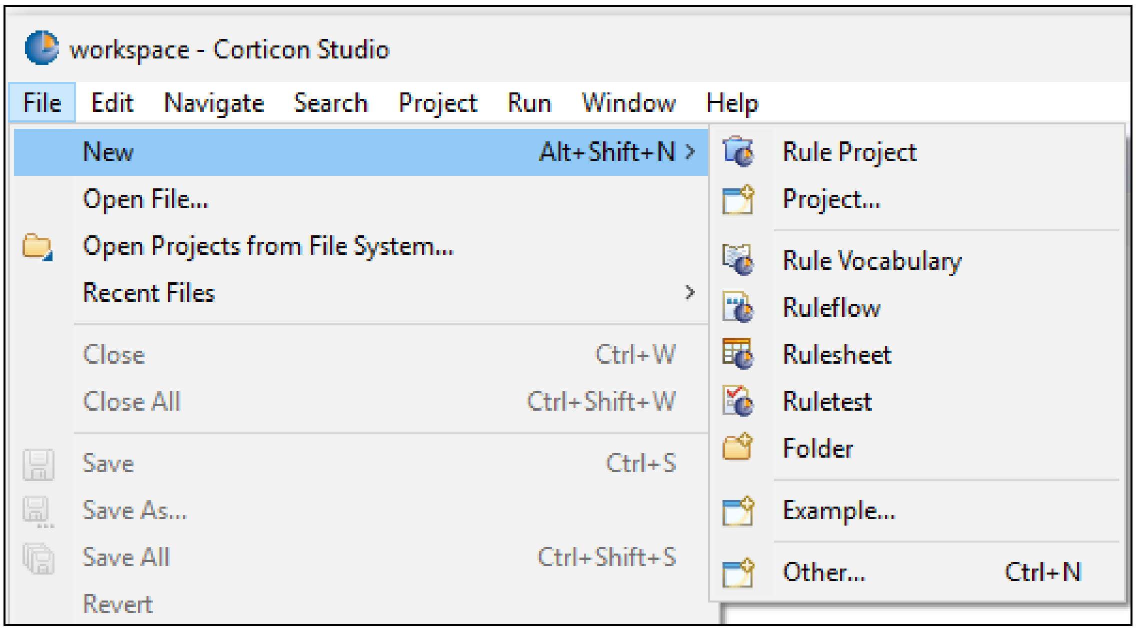
Workspaces, projects, and folders provide a way to organize rule-modeling components. A workspace is a root directory that acts as a container for rule projects created in Corticon Studio.



A Corticon rule project such as the Training project is a container for rule-modeling components. A Corticon rule project can further contain folders to separate rule-modeling components, which can be further separated by folders. When you create workspaces, rule projects, or folders, the relevant directory and folder structures are created as locations on your computer.



Rule-modeling components

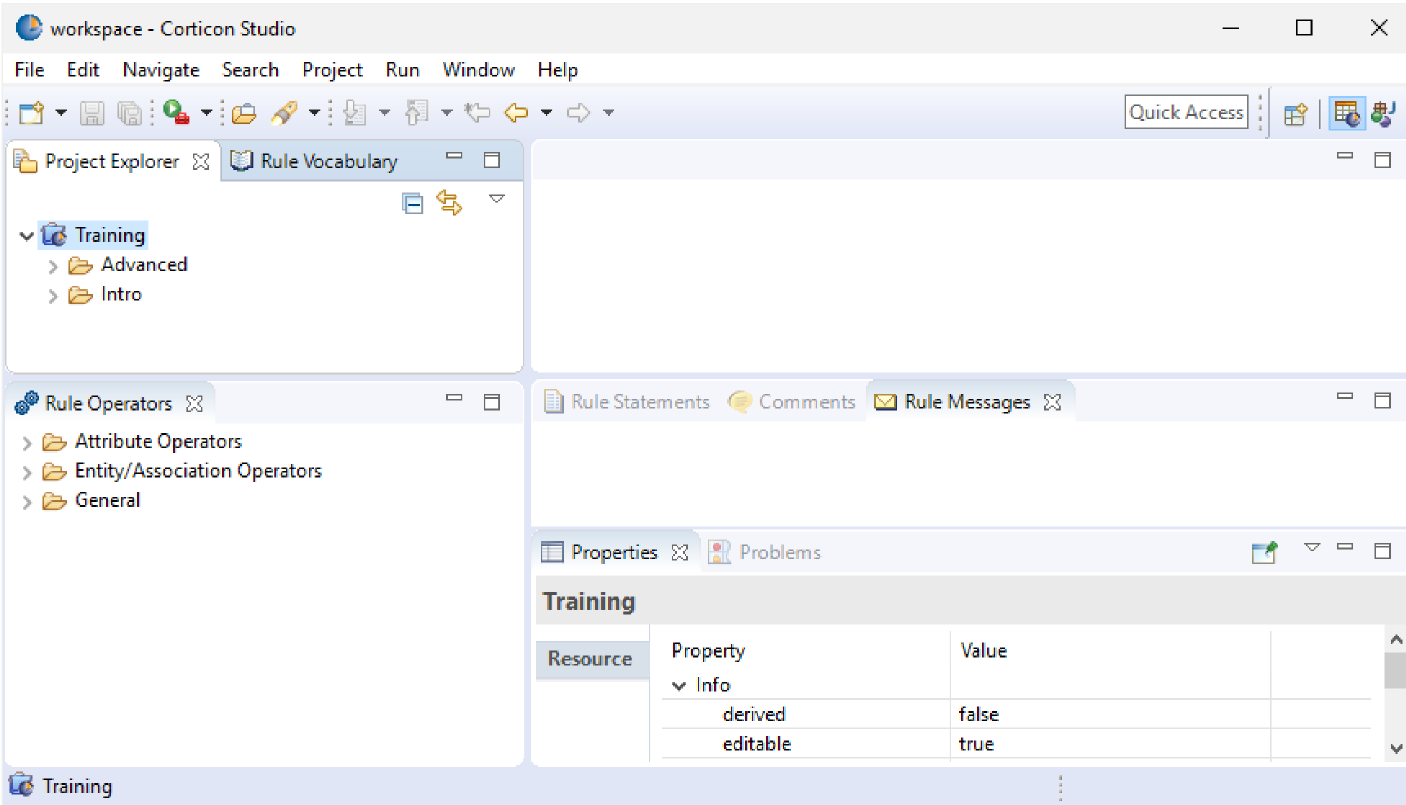


Corticon Studio provides capabilities to create the following rule-modeling components:

- **Rule Vocabulary**—Enables you to define business terms used by the rule model.
- **Rulesheet**—Enables you to model the rule logic using terms from the Vocabulary.
- **Ruleflow**—Enables you to organize one or more Rulesheets into a sequence that can be deployed as a Decision Service on the Corticon Server.
- **Ruletest**—Enables you to test Rulesheets and Ruleflows.

Together, the Vocabulary, Rulesheets, and Ruleflows form a rule model in Corticon.

Views and perspectives



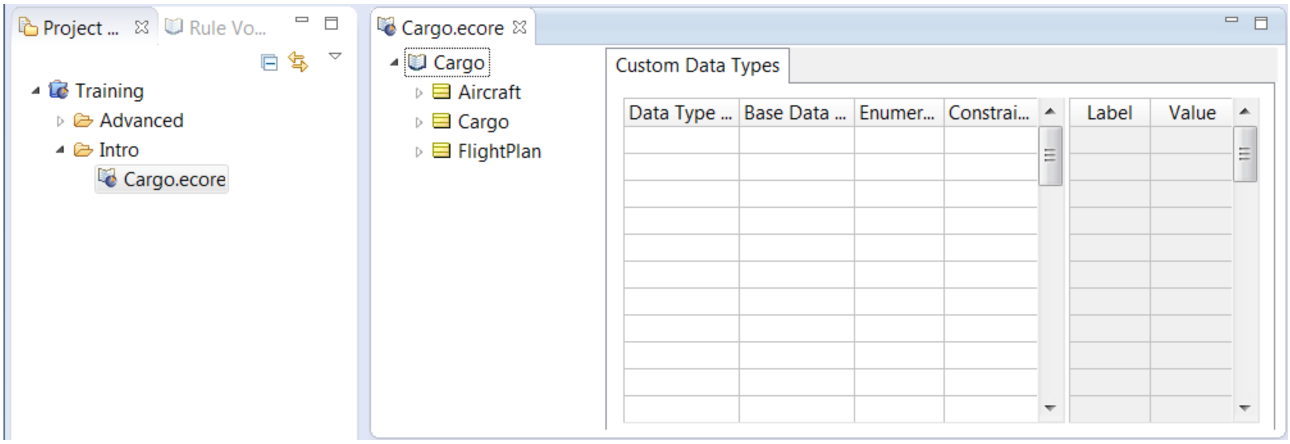
The user interface of Corticon Studio is composed of several panels, each with one or more **Views** as tabs. Each of these **Views** displays rule components, details of rule components, or other features used in rule modeling. They provide an easy way to access and use rule components.

For example, the **Project Explorer** view in this image enables you to access rule components contained in rule projects. To open a rule component, you simply double-click it from the **Project Explorer** view.

A perspective is an arrangement or layout of panels and views. By default, Corticon Studio opens in the Corticon Designer perspective, which provides all the views you require to model your rules.

You can customize a perspective by clicking and dragging views within Corticon Studio. You can close views by clicking the Close button on the view.

Editors



When you open a rule-modeling component (by creating it, or by double-clicking it in the **Project Explorer** view), it opens in its own, unique, editor, which enables you to define and develop the rule component.

In this example, a Vocabulary file (`Cargo.ecore`) is opened in a Vocabulary editor.

AI Assistant

AI Tip **Meet your AI Assistant**

Corticon Studio features a powerful AI Assistant designed to help you create, understand, and optimize your rule projects. Whether you're new to Corticon or an experienced user, the AI Assistant can be a valuable partner as you work in Corticon Studio.

Throughout this guide, you'll find AI Tips—practical suggestions for making the most of the AI Assistant. We encourage you to explore and experiment to discover how it can best support your rule modeling tasks.

For step-by-step instructions on enabling and using the Corticon AI Assistant, click AI Tip wherever you see it:

Tasks to create and test rule models

To model and test rules in Corticon Studio, you must do the following:

- Import or create a Vocabulary that contains business terms.
- Model your rule logic in Rulesheets.
- Test your rule logic using Ruletests.
- Create a Ruleflow that can be deployed as a Decision Service.

2

What is a Vocabulary?

A Corticon Vocabulary is a rule-modeling component that enables you to define all the business terms required in your rules.

For example, a transport company may have a rule that determines how much cargo each type of vehicle can carry. Two key business terms used in this rule are cargo and vehicle. You can define these terms as **Entities** in your Vocabulary.

A Vocabulary is similar to a data model such as a Unified Modeling Language (UML) model or an Entity-Relationship model. The terms in the Vocabulary can come from a number of sources—database tables, forms used in business operations, policy and procedure documents, and so on.

When you build a Vocabulary, you define not only the terms, but also the relationships between them. For example, a single vehicle can carry many cargo containers, implying a one-to-many relationship. You define this as an association in your Vocabulary.

For details, see the following topics:

- How business terms are represented in the Corticon Vocabulary
- Create a Vocabulary file
- Add an entity
- Add attributes to an entity
- Types of associations
- Add associations to an entity
- Rule Operators

How business terms are represented in the Corticon Vocabulary

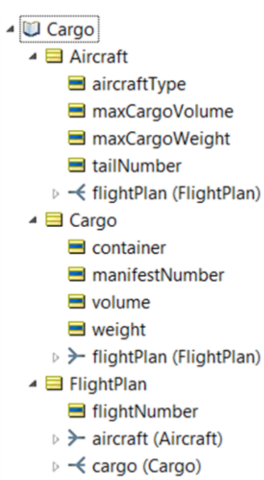
Business terms are represented in the Vocabulary as either entities or attributes.

An attribute is like a data field that holds a value. An entity is a collection of attributes. For example, the term cargo may have data fields such as cargoID, weight, volume, and so on. So, cargo should be defined as an entity in a Vocabulary and the terms cargo ID, weight, and volume as attributes that belong to the entity cargo.

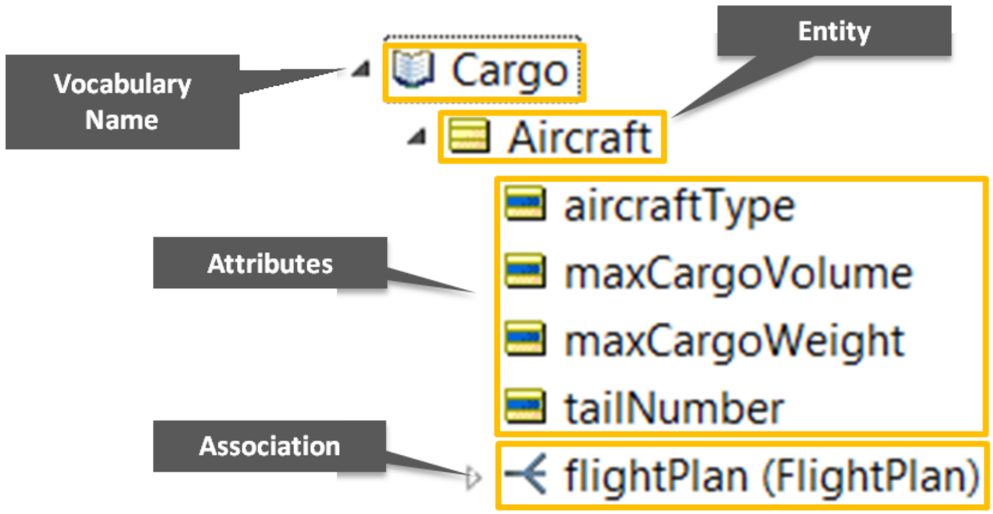
Relationships between entities are represented as associations in a Vocabulary. For example, since a vehicle can carry multiple cargo containers, you can create a one-to-many association between the vehicle entity and the cargo entity in a Vocabulary.

The Vocabulary tree

A Vocabulary is represented as a hierarchical tree:



The top level—the root node of the tree—is the **Vocabulary Name**. Its icon is an open book icon. In this example, **Cargo** is the Vocabulary Name:



One level under the Vocabulary name are its **entities**. Entities have a golden box icon. Each entity name must be unique in the Vocabulary.

Indented under each entity are its **attributes**. Each attribute has a name, a data type, and a few other properties. Each attribute has a golden box icon with a green bar.

Finally, **associations** are at the same hierarchical level as attributes. The associated entity is displayed in parentheses. The association icon changes based on the type of association. Because the `Cargo` entity has a many-to-one relationship with the `FlightPlan` entity, the icon is multiple bars converging into a single horizontal bar. The `FlightPlan` to `Cargo` association shows a reverse of the icon—a single bar diverging into multiple bars.

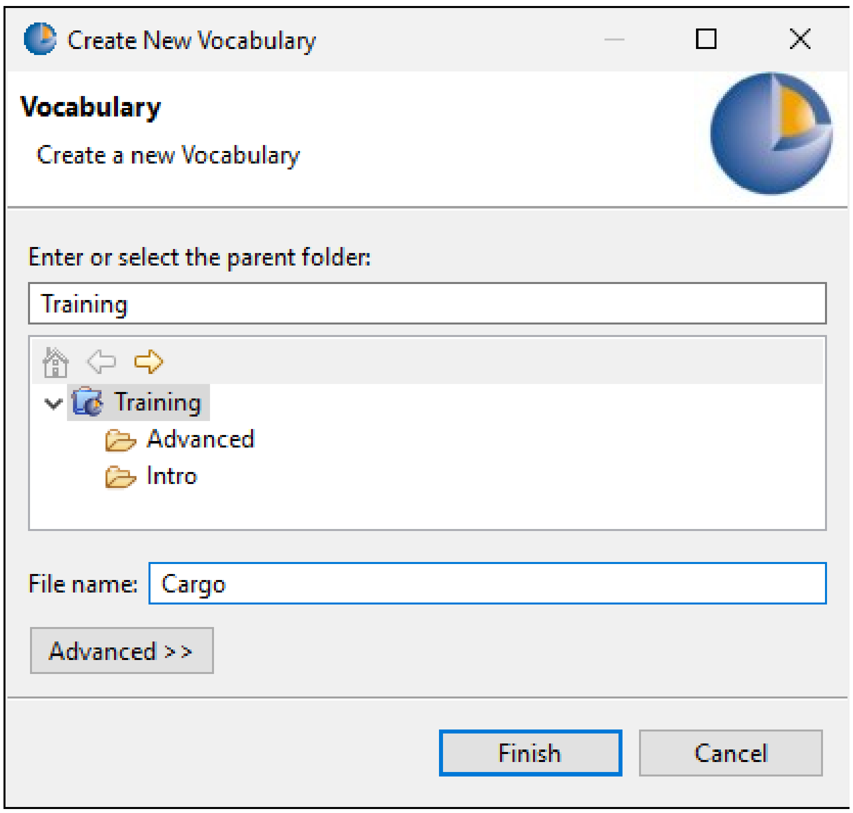
Tasks to define a Vocabulary

To define a Vocabulary:

- Create a new Vocabulary or import an existing Vocabulary.
- Add entities to the Vocabulary.
- Add attributes to entities.
- Add associations to entities.

Create a Vocabulary file

AI Tip **Learn About Vocabulary**
Use the AI Assistant to explore and clarify the vocabulary elements in your project. This can help ensure consistency and clear understanding across your team. Try prompts like: • “Summarize the purpose of this vocabulary.” • “List all the bi-directional entity associations in this vocabulary.”



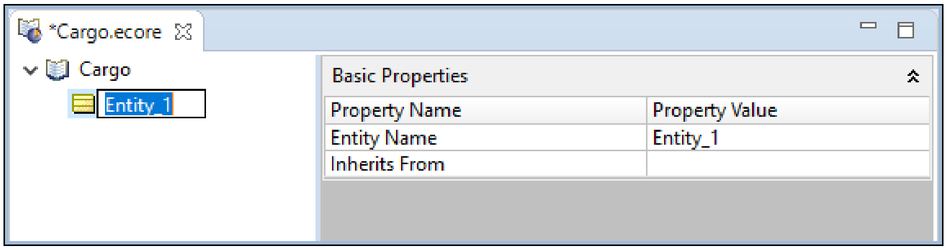
You must create a Vocabulary as part of a rule project. The Vocabulary file opens in the Vocabulary editor and has the extension `.ecore`.

To create a Vocabulary file:

1. Select **File > New > Rule Vocabulary**.
2. In the **Create New Vocabulary** wizard:
   a. In the **Enter or select the parent folder** area, select the rule project and optionally, a folder within it.
   b. In the **File name** field, enter a name for the Vocabulary file.
   c. Click **Finish**.

The Vocabulary file is created and appears under the rule project in the **Rule Project Explorer** view. It is represented by a folder icon image and automatically opens in the Vocabulary editor.

Add an entity



Before you add an entity, you must determine what you want to name it. Every entity name must be unique within the Vocabulary and cannot contain any spaces.

Note: Certain terms such as size, sum, and now are part of a predefined set of Rule Operators. The Operator Vocabulary contains predefined operators such as =, >, and <, that are required to build business rules. They cannot be used as entity names. Refer to the “Vocabularies” section of the Quick Reference Guide for more information on restricted names.

Follow these steps to add an entity in the Vocabulary editor:

1. In the Vocabulary editor, right-click the Vocabulary name and select **Add Entity**.

   Note: An entity with the default name `Entity_1` is created. The entity name is automatically editable. If it is not editable, double-click the default name `Entity_1`.

   ---

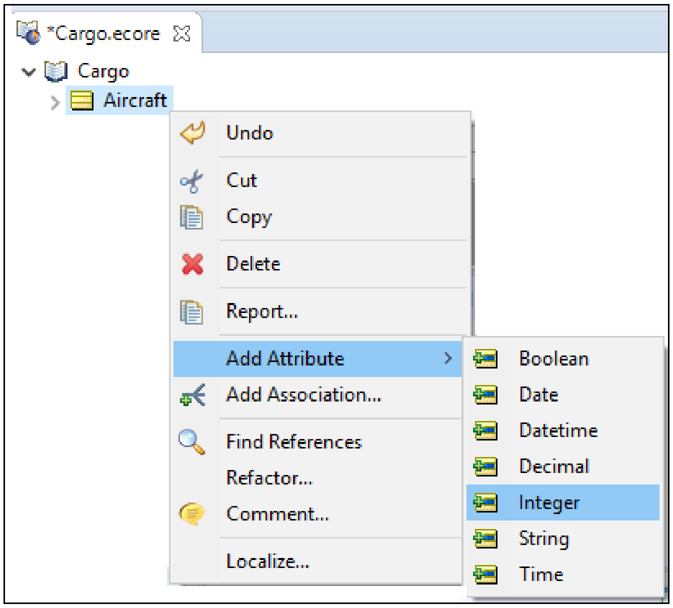
2. Delete `Entity_1` and enter the name of the business term `Aircraft` as the entity name.

Add attributes to an entity

After you create an entity, you can add attributes to it. An attribute is similar to a data field or a variable, whose values are populated at runtime.

Follow these steps to add attributes to an entity in the Vocabulary editor:

1. Right-click the entity name and then select **Add Attribute**.
2. Choose from seven data types displayed in the side list—Boolean (True or False), Decimal, DateTime, Date, Integer, String, Time. For this example, choose **Integer**.

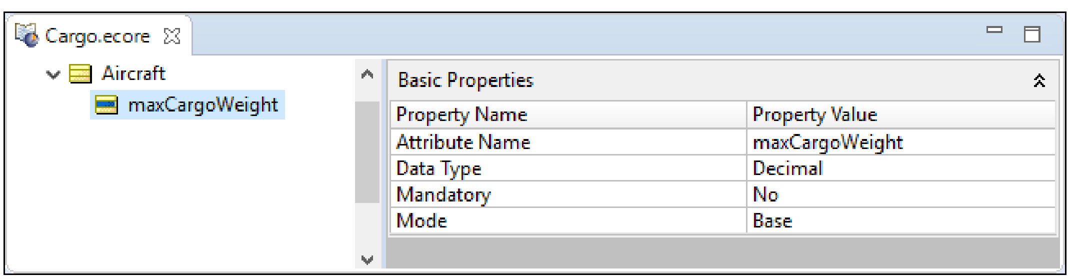


Note: An attribute with the default name `Attribute_1` is created. The attribute name is automatically editable. If it is not editable, double-click the default name `Attribute_1`.

---

3. Delete `Attribute_1` and enter the attribute name. Attribute names must be unique for each entity. For this example, type in `MaxCargoWeight`.

After you add an attribute, the property editor panel of the Vocabulary editor displays all attribute's properties with default values.



AI Tip **Refine Your Vocabulary**

The AI Assistant can help spot redundant or unclear attributes in your vocabulary and suggest ways to improve clarity. Try prompts like:

- “Are there any redundant attributes in this vocabulary?”
- “Suggest ways to make the elements of this vocabulary more clear.”

Each attribute has the following properties:

- **Attribute Name**—Automatically populated when you define the attribute.
- **Data Type**—Assigned when you created the attribute, you can change the data type in the drop-down list—Boolean (True or False), Decimal, DateTime, Date, Integer, String, Time. You can also create custom data types using the **Custom Data Types** tab.
- **Mandatory**—Set to **No** by default. Select **Yes** if the attribute value cannot be null and must be populated at runtime. Retain **No** if the attribute value can be null.
- Mode—Set to **Base** by default. This setting enables the attribute to be used by systems outside Corticon. In most cases, you should retain the default setting. For more information about this property, refer to the Corticon documentation.

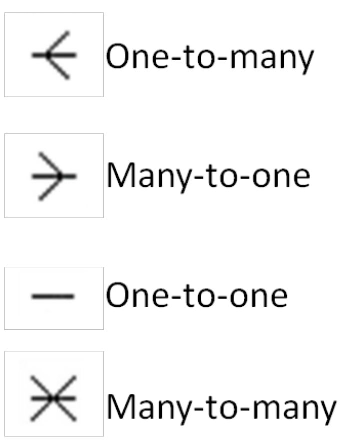
You can add a comment to any Vocabulary attribute, entity, association, and domain. Comments are listed by date and can be set as Note, TODO, ChangeLog, or NeedsReview.

Try adding a comment to **Cargo.weight**:

1. Right-click on **weight** in the **Cargo** entity, and then choose **Comment**.
2. In the **Comment** window, type `Weight in whole kilograms.` and click **OK**.

Types of associations

An association describes the relationship between two entities. In a Corticon Vocabulary, you can define the association under either entity. By default, the association that you define appears under both entities in the Vocabulary tree.



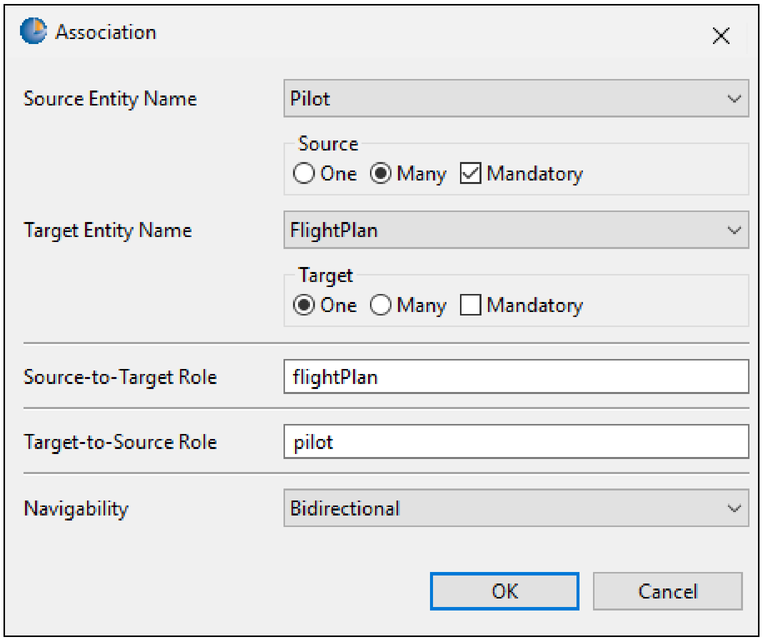
There are four types of associations:

- **One-to-many**—If a vehicle can contain many cargo containers, from the perspective of the Vehicle (source) entity, the relationship with the Cargo (target) entity is one-to-many. A one-to-many association's icon is a single bar diverging into multiple branches.
- **Many-to-one**—From the perspective of the Cargo (source) entity, the relationship with the Vehicle (target) entity is many-to-one. A many-to-one association's icon shows multiple branches converging into a single bar.
- **One-to-one**—If a vehicle can have only one driver, the relationship is one-to-one. This is true from both perspectives. A one-to-one association's icon is a single bar.

- **Many-to-many**—Multiple workers may be assigned to load multiple vehicles through the workday. Each worker may load multiple vehicles, and each vehicle may have multiple workers assigned to load it. A many-to-many association's icon shows multiple branches converging to a point from both sides.

Add associations to an entity

When you define an association, you can add it under either of the two entities in the relationship. You add an association to an entity just as you add an attribute. You must right-click the entity and select **Add Association**.



The Association dialog box opens in which you set the association's properties. In most cases, you only need to look at these properties:

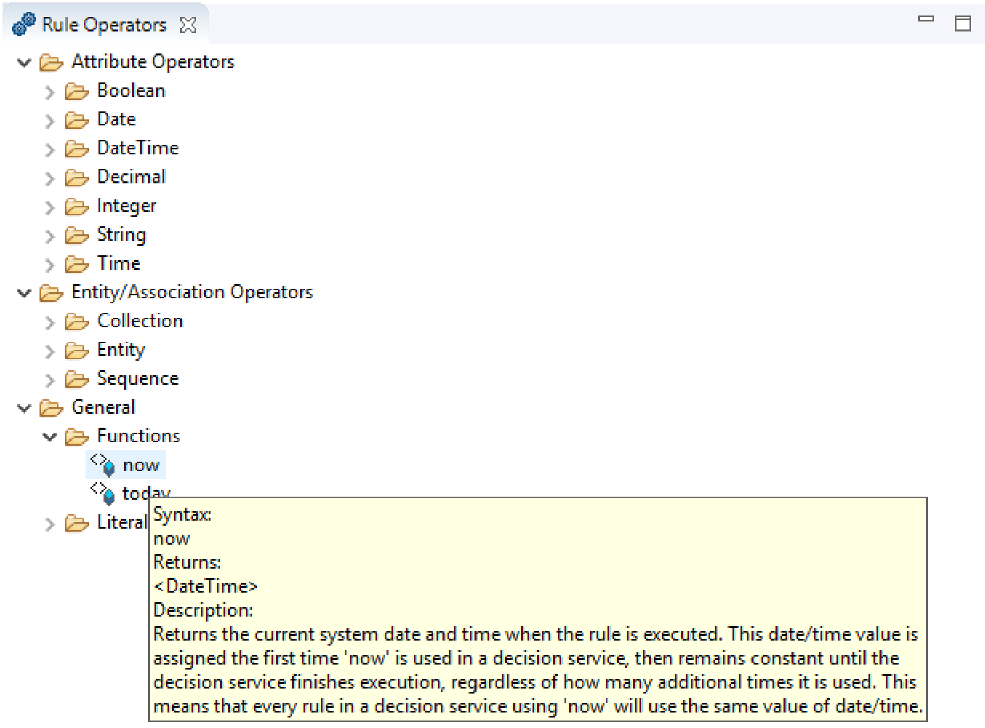
- **Source Entity Name**—By default, the entity to which you are adding the association is selected as the source entity. You should retain this default setting.
- **Source**—Select **One** or **Many** depending on the role you want the source entity to play in the relationship.
- **Target Entity Name**—Select the target entity from the drop-down list.
- **Target**—Select **One** or **Many** depending on the role you want the target entity to play in the relationship.

When you complete setting the properties, click **OK**.

The association is displayed in the Vocabulary tree under both entities with the relevant association icon.

Rule Operators

When you use the Vocabulary to build rules, you also make use of rule operators such as =, <, and >. Corticon Studio provides a rich set of predefined rule operators in the **Rule Operators** view. By default, this view is located at the bottom-left of the Corticon Designer perspective.



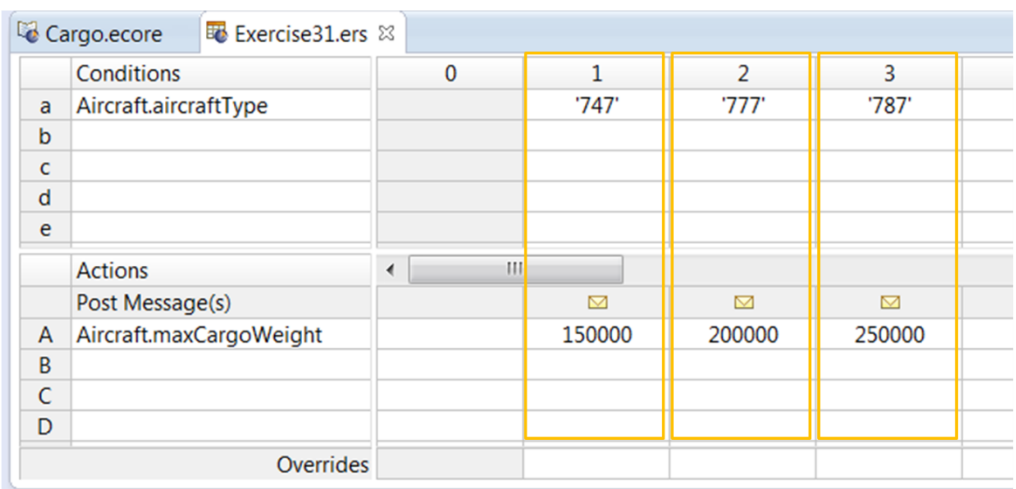
The Rule Operators view organizes operators into folders based on their function and purpose. For example, the Attribute Operators folder contains subfolders for different data types. These subfolders data type contain rule operators that you can use when defining rules, which you can when defining rules of that data type.

You can learn about each rule operator in the **Rule Operators** view by hovering the mouse pointer over it. A tooltip appears, describing the rule operator, as shown in this image. For details about all the available rule operators, see *the Rule Language Guide in the documentation set*.

3

What is a Rulesheet?

You define your rule logic in a Corticon Rulesheet. A rule is like an ‘if-then’ statement. Each rule consists of one or more conditions (if) that are associated with one or more actions (then).



This Rulesheet has three rules. The Rulesheet editor has the following parts:

- **Conditions**—Where you define the conditions for each rule. For example, `Aircraft.aircraftType = 747`. The condition value could be a single value (`747`), a set of values (`747, 777, 787`), or a range of values (`weight=100000..200000`).
- **Actions**—Where you define the actions that need to be triggered when the conditions are satisfied. For example, `Aircraft.maxCargoWeight=150000`.

- **Rule columns**—They represent a rule, for example, the highlighted columns in this image. Each column represents a rule. These columns associate a set of conditions with a set of actions. For example, column 1 defines the rule—if the aircraft is a 747, then its maximum cargo weight is 150,000.

The terms `Aircraft.aircraftType` and `Aircraft.maxCargoWeight` come from the Rule Vocabulary. Each Rulesheet must be linked to a Rule Vocabulary.

Corticon evaluates all the conditions in each rule. If all the conditions in the rule are satisfied, the actions in the rule are triggered.

Note: If an action does not execute for some reason, Corticon still tries to execute the other actions in the rule.

AI Tip **Understand Rulesheets**

The AI Assistant can help you quickly understand the purpose and structure of any rulesheet, making it easier to work with both new and existing projects. Try prompts like:

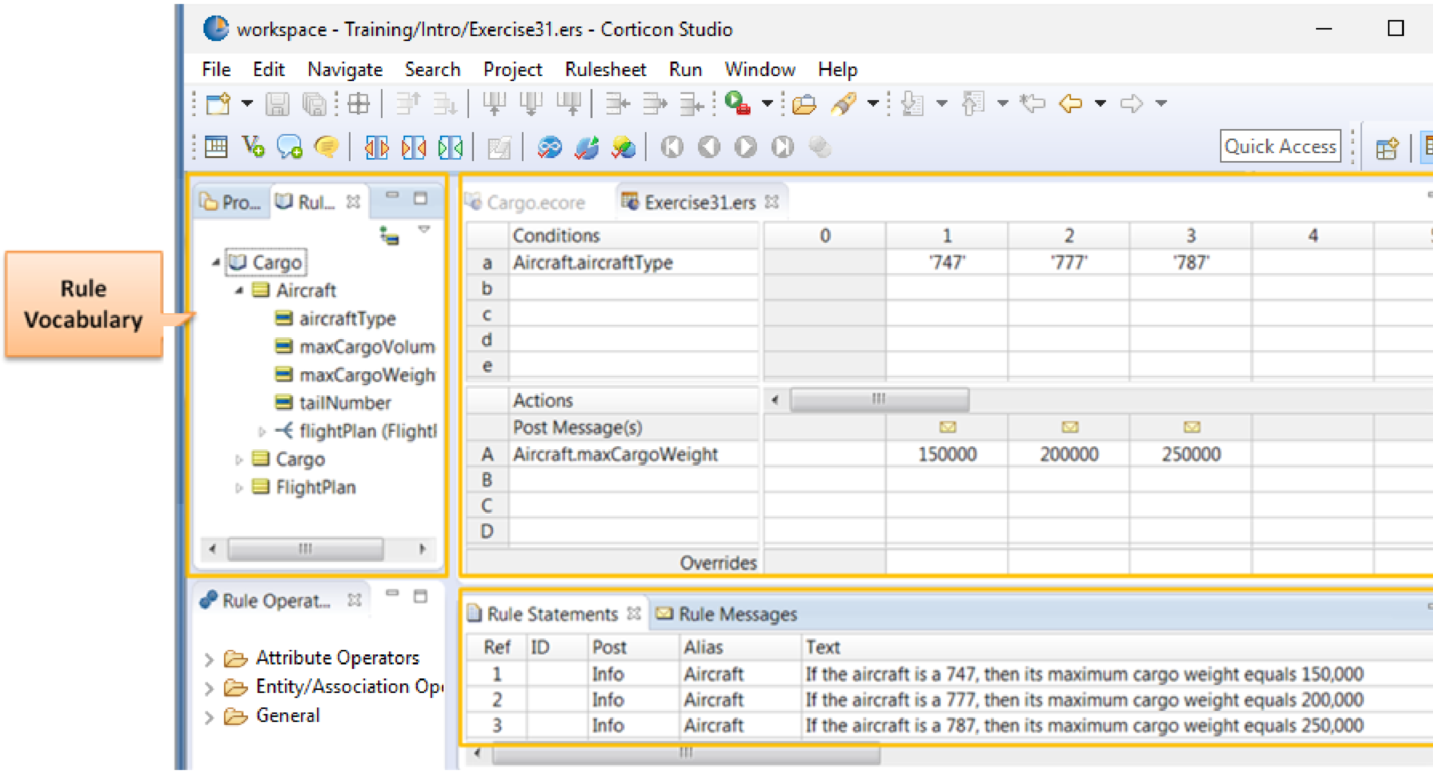
- “Summarize the purpose of this rulesheet.”
- “List each rule in this rulesheet, with a brief explanation of what it does.”

For details, see the following topics:

- Tasks to define rules in a Rulesheet
- Create a Rulesheet file
- Define a rule statement
- Define rules in the Rulesheet editor
- Link a rule with a rule statement
- Analyze rules

Tasks to define rules in a Rulesheet

To define a Rulesheet you will use the following views:



Perform the following tasks to define rules in a Rulesheet:

1. Create a Rulesheet file.
2. Describe each rule informally in a rule statement. Rule statements help document the purpose of a rule. They can also be posted as messages to a client application when the rule fires.
3. Define the rules by dragging terms from the **Rule Vocabulary** view to the **Conditions** and **Actions** sections in the Rulesheet editor and defining values for each rule in the rule columns.
4. Link each rule statement with its rule.

Create a Rulesheet file

You create a Rulesheet file just like you create a Vocabulary file. A Rulesheet file has the extension `.ers`.

Note: A Rulesheet file must be linked to a Vocabulary file. When you create a Rulesheet file, you create it for a rule project. If a Vocabulary exists in the same rule project (before you create the Rulesheet file), Corticon Studio and Corticon.js links it to the Rulesheet, by default.

Follow these steps to create a Rulesheet file:

1. In Corticon Studio, select **File > New > Rulesheet**.
2. In the **Create New Rulesheet** wizard:
   a. In the **File name** field, type a name for the Rulesheet.
   b. In the **Enter or select the parent folder** field, specify the rule project or folder in which the Rulesheet must be created.
   c. Click **Finish**.

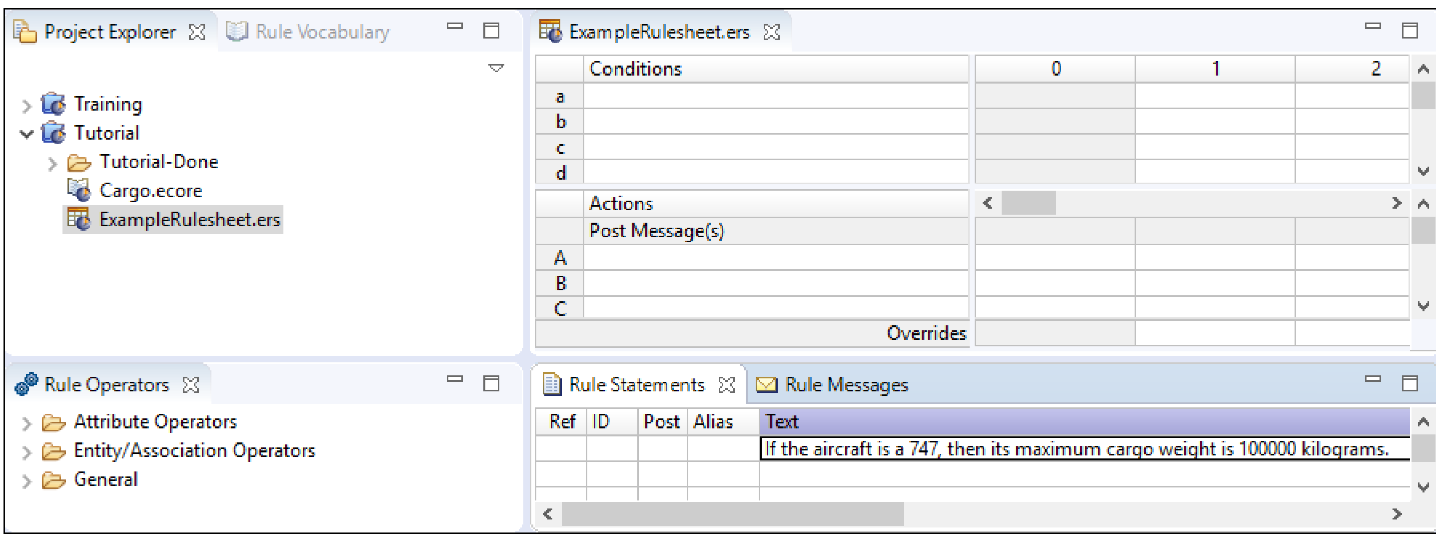
The newly created Rulesheet file appears in the **Project Explorer** view. The Rulesheet also opens by default in the Rulesheet editor. In addition, the **Rule Vocabulary** view and the **Rule Statements** view become active.

Define a rule statement

A rule statement serves the following needs:

- Writing the rule in plain or informal language enables you to articulate the rule quickly, prepares you for defining the rule formally, and helps you identify Vocabulary terms needed in the rule.
- The rule statement describes the rule and can be made a part of the output message sent by Corticon when the rule fires. So by reading the output message users can understand the all the actions triggered by the rule.
- Finally, a rule statement is a way to document a rule so that stakeholders can understand its purpose and logic by reading the rule statement.

Rule statements appear in the Rule Statements view.



Note: You can define multiple rule statements for a single rule. For example, one rule statement can document the rule, while another is sent as part of the output message when the rule fires.

Follow these steps to create a rule statement:

1. Open the Rulesheet. The Rule Statements view becomes active. The Rule Statements view comprises several rows and columns.
2. To create the rule statement, double-click a cell in the column named **Text** and type the rule statement.

Define rules in the Rulesheet editor

A Rulesheet typically contains multiple rules. This Rulesheet has two rules.

	Conditions	0	1	2	3	4
a	Aircraft.aircraftType		'747'	'777'	-	
b						
c						
d						
	Actions					
	Post Message(s)					
A	Aircraft.maxCargoWeight		150000	200000		
B						
C						
	Overrides					

To define rules in the Rulesheet:

• Select and drag the Vocabulary term from the **Rule Vocabulary** view and drop it in the next empty cell in the **Conditions** or **Actions** pane. Drop each Vocabulary term into a separate row.

Or

1. Double-click the cell that corresponds with the Vocabulary term.

2. Enter the condition and action value.

Note: The values that you enter must conform to the syntax rules enforced by Corticon.

When you specify a value in a rule column cell, the equality operator is implied. For example, when you enter `747` in column 1 as shown here, it means (if) `Aircraft.aircraftType = '747'`.

AI Tip Get More from Your Prompts

The AI Assistant can provide responses of varying length and detail to your prompts. The more specific your request, the more likely you are to get the result you want. Try prompts like:

- “Generate a one-sentence summary of this rulesheet.”
- “Produce a formal, detailed specification of this rulesheet, including all rules and vocabulary elements used.”

Syntax for values in rule column cells

The values that you specify must conform to the following syntax:

- If the Vocabulary term’s data type is String, Date, DateTime, or Time, the corresponding value in the rule column must be enclosed in plain single quotes.

- CORRECT: 'apple'
- INCORRECT: “apple”, ‘apple’, apple

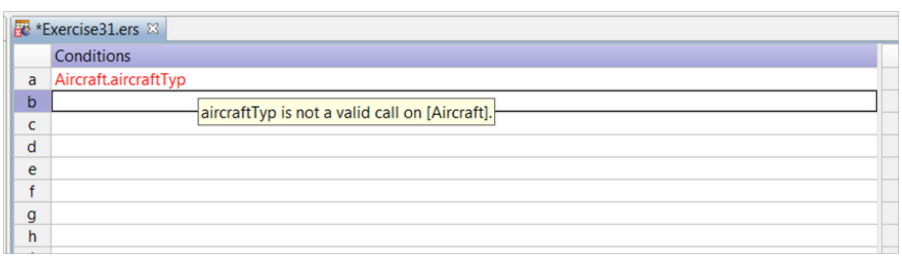
- If the Vocabulary term’s data type is Integer, the corresponding value must neither be enclosed in quotes, contain decimal points or delimiters.
  - CORRECT: 4, -4
  - INCORRECT: 4.0, '4'

- If the Vocabulary term is Decimal, the corresponding value can optionally contain a decimal point, but not commas. Also the value must not be enclosed in quotes.
  - CORRECT: 10, -10.0, 10.5, 25145
  - INCORRECT: 10,25 or 1,025.00

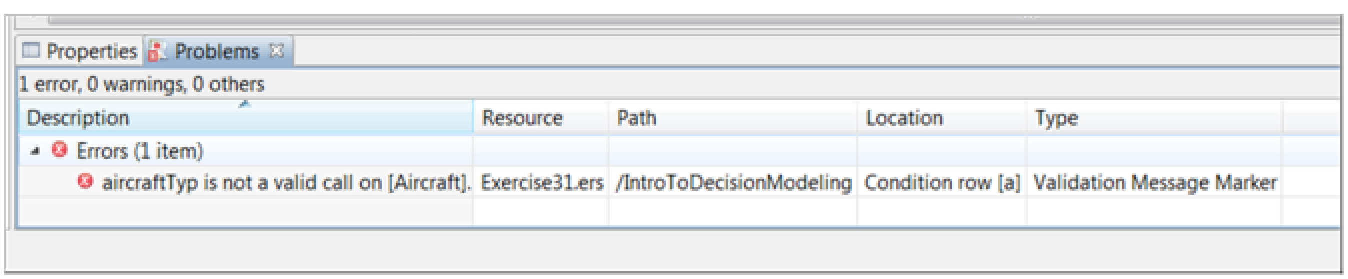
- If the Vocabulary term is Boolean, the corresponding value can only be T, F, true, or false. The case does not matter but the value must not be enclosed in quotes.
  - CORRECT: t, TRUE, F, FALSE
  - INCORRECT: 'false'

Syntax errors in a Rulesheet

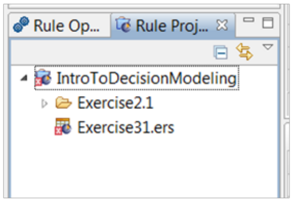
If you make any syntax errors while defining a rule, Corticon Studio detects the error and highlights it in red. If you hover your mouse pointer over the error, a tooltip appears, describing the problem and enabling you to understand the cause of the error and resolve it.



A description of the error is also displayed in the `Problems` view. It describes the causes of errors in all rule-modeling components that are open.

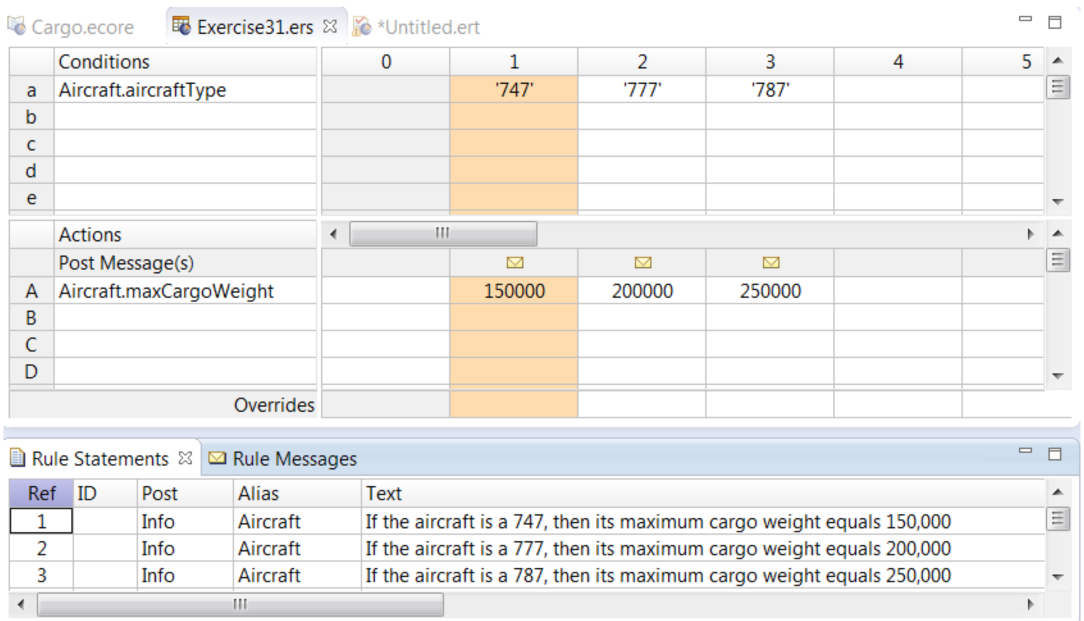


The project folder and the Rulesheet in the Rule Project Explorer view are also marked with error symbols.



Link a rule with a rule statement

You must explicitly link each rule statement with its rule. To link a rule statement with its rule, enter the number of the rule column in the rule statement's Ref cell. If you only want to document the rule, you do not need to configure any other properties.



If you want to post the rule statement as part of the output message, configure the following properties:

- **Post**—Enables you to indicate the severity of the rule violation. You can select from three levels—Info, Warning, and Violation. Note that these severity levels are user-defined and have no special meaning in Corticon. You can define them to match your requirements.
- **Alias**—Enables the Vocabulary entity in the rule condition that you want displayed with the output message when there is a rule violation.

Note: The Ref, ID, Post, and alias columns enable many optional ways to set up Rule Statements. *For more information, see "Rule statements window" in the Quick Reference Guide.*

AI Tip **Quickly Document Rulesheets**

Use the AI Assistant to generate documentation for your rulesheets to share with stakeholders. Try prompts like:

- “Generate a high-level specification for this rulesheet.”
- “Create a detailed specification for this rulesheet, including a listing of each rule and the vocabulary elements used in conditions and actions.”

After you link a rule statement with a rule, when you select the rule statement in the Rule Statements view, the corresponding rule column is highlighted in orange. Similarly, if you select a cell or a group of cells within the rule column, the corresponding rule statement is highlighted in orange, indicating that the rule statement and the rule are linked.

Analyze rules

When you have finished modeling rules, analyze the rules for logical errors. Often, initial business rule specifications are:

- **Ambiguous**—The rules conflict under certain scenarios.
- **Incomplete**—The rules fail to address all possible scenarios.
- **Looping**—The rules form circular logic or loops.

Before automating the rules, it is critical to eliminate logical errors to ensure that the decision service provides correct and consistent results. Corticon Studio provides unique and powerful features to help you ensure that rules are complete and consistent.

AI Tip **Discover Ways to Improve Your Rules**

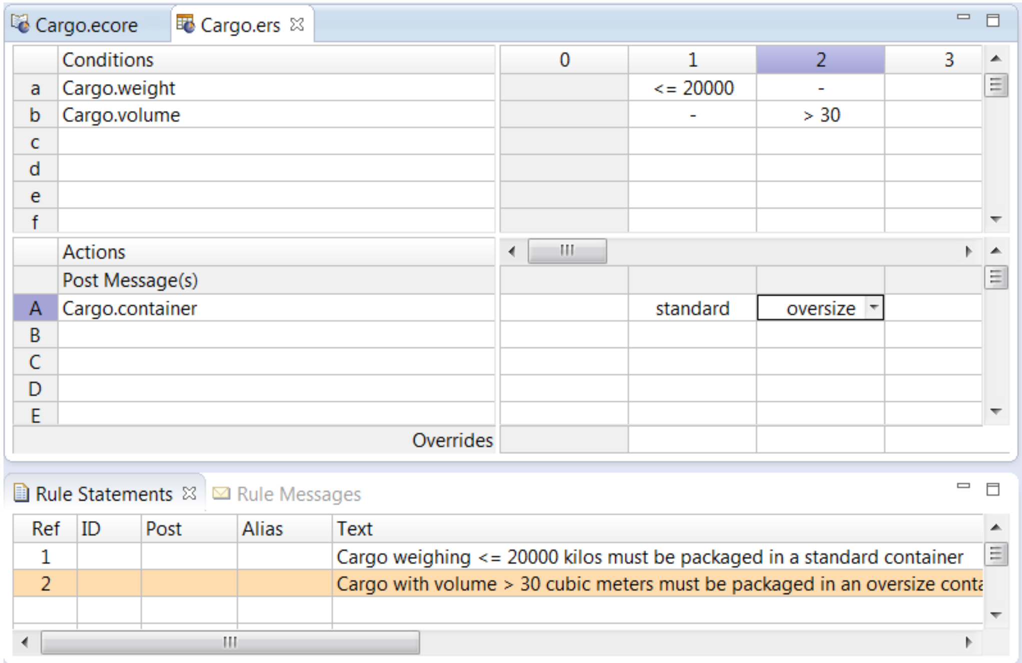
The AI Assistant can help identify potential improvements in your rulesheets, such as redundancies or inefficiencies. Try prompts like:

- “How could this rulesheet be improved?”
- “What other conditions might this rulesheet need to cover?”

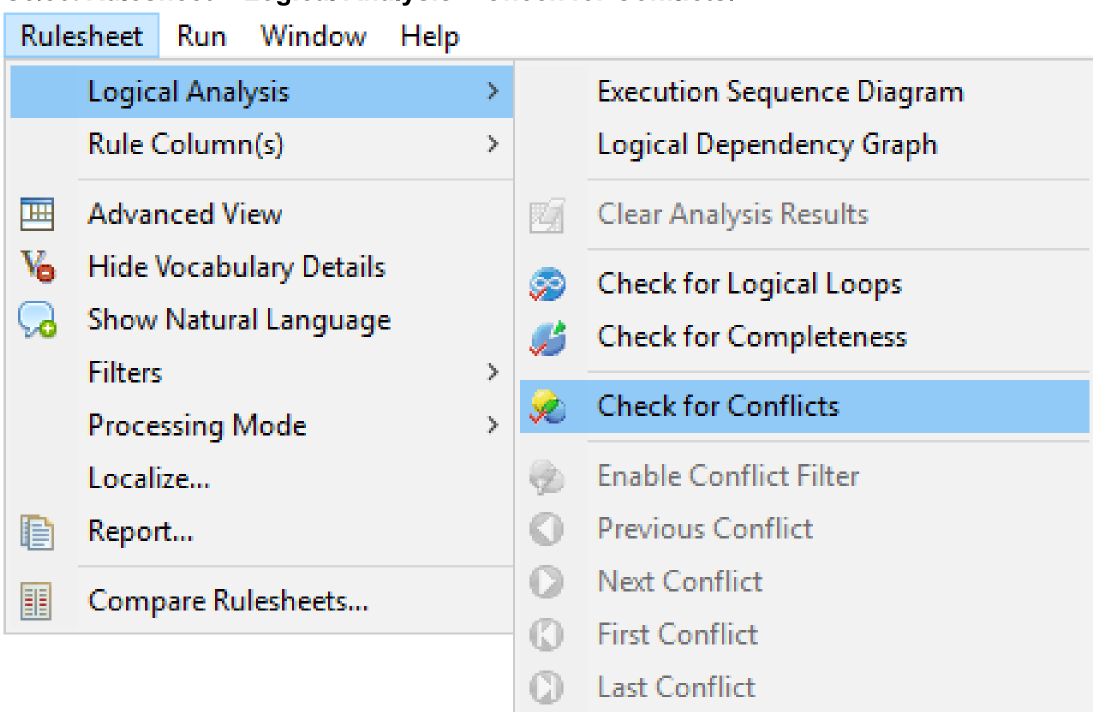
Check for conflicts

Begin by checking for conflicts in the rules:

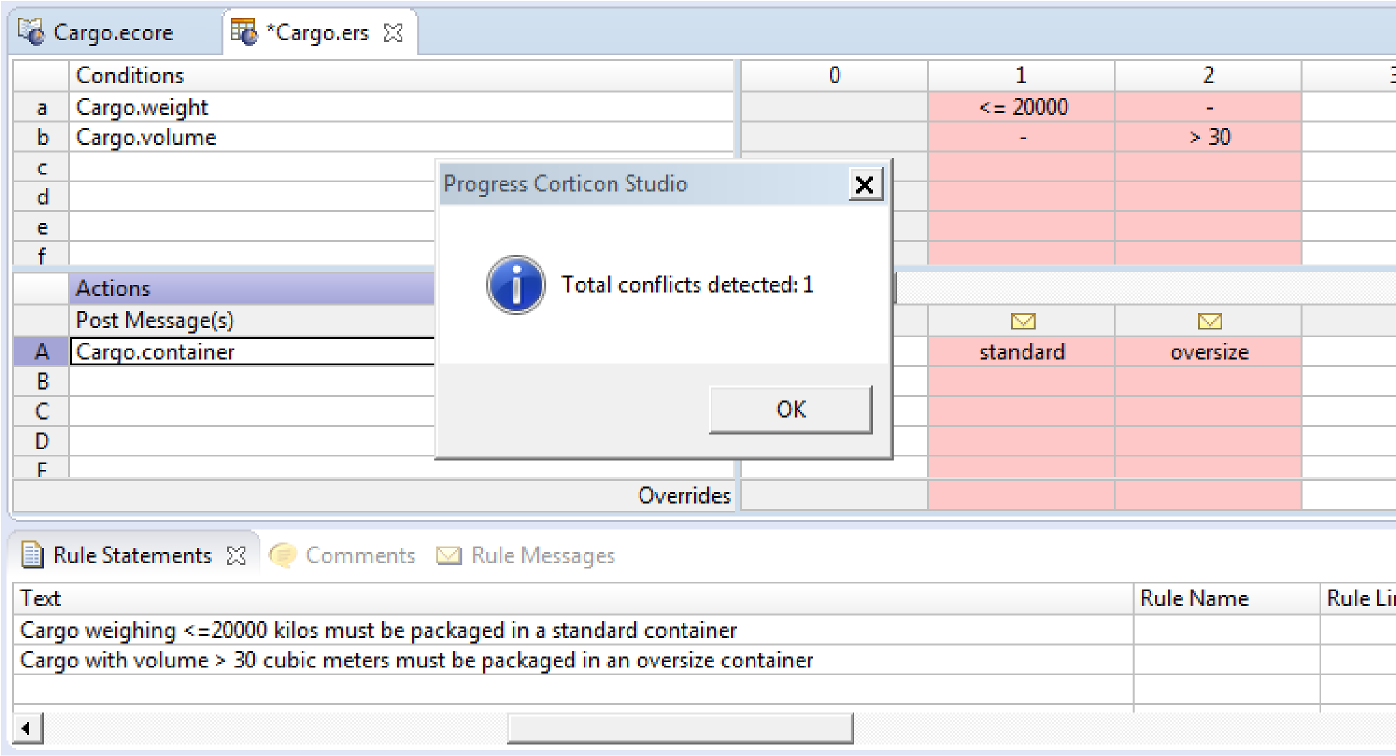
1. Open the Rulesheet and select the **Cargo.ers** tab.



2. Select **Rulesheet > Logical Analysis > Check for Conflicts**.



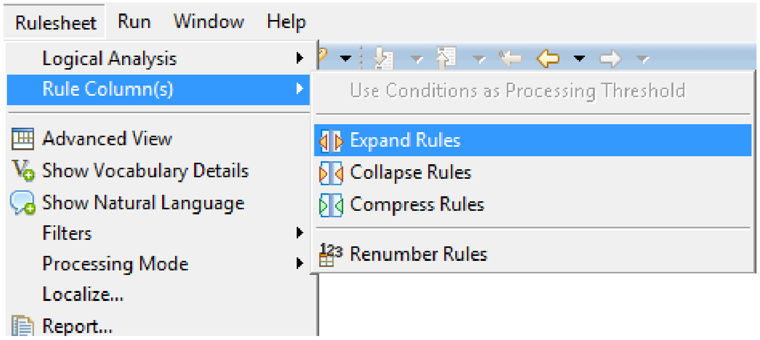
If one or more columns contain conflicting rules, Corticon detects them and highlights the columns in pink. The total number of conflicts is also displayed in a dialog box.



3. Click **OK** to dismiss the dialog box.

Resolve conflicts

Sometimes, conflicts may not be immediately visible just by looking at the rules because each rule is actually made up of sub-rules (rules without dashes) and it is the sub-rules that are in conflict. To see these sub-rules, select **Rulesheet > Rule Column(s) > Expand Rules**.



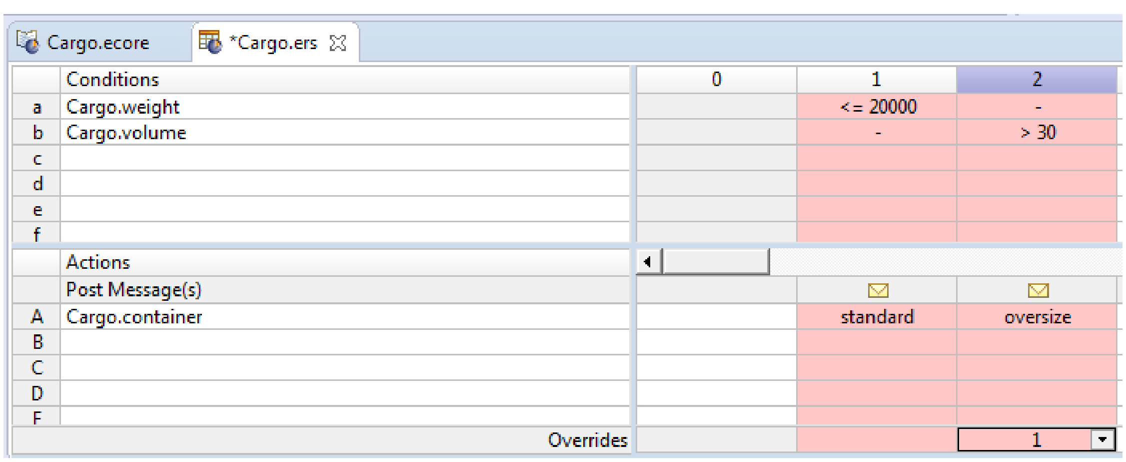
It helps you pinpoint the source of the conflict.

Rule 1 is expanded into three columns, 1.1, 1.2, and 1.3, and rule 2 is expanded into three columns 2.1, 2.2, and 2.3. The expansion shows all of the logical possibilities for each rule. Rule 1 states Cargo weighing <= 20,000 kilos, regardless of volume, must be packaged in a standard container. Corticon Studio recognizes three possible ranges for Cargo.volume (<=30, >30, and null), as seen in the expanded rules.

With the rules expanded, the source of the conflict becomes obvious. Scenarios with Cargo.weight <=20000 and Cargo.volume > 30 are in conflict, because they define mutually exclusive actions (rule 1.2 assigns a standard container while rule 2.1 assigns an oversize container). To get your rules right, this conflict must be addressed.

To resolve the conflict, you can either change your original rules, or decide that one rule should override the other. To implement the override:

1. Collapse your rules back to the original state by selecting **Rulesheet > Rule Column(s) > Collapse Rules**.
2. Override Rule 1 with Rule 2. In the **Overrides** cell in Rule 2, select the column number of the rule that you want Rule 2 to override—in this case, Rule 1.

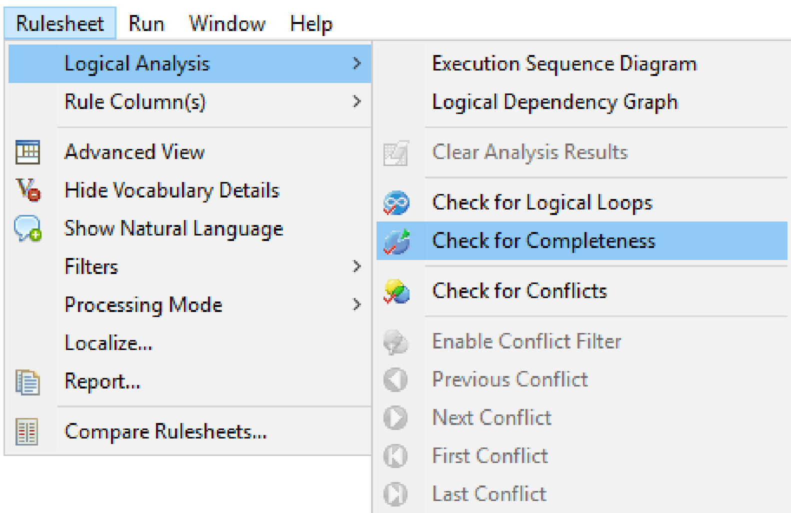


3. Check for conflicts again by selecting **Rulesheet > Logical Analysis > Check for Conflicts**. You see that the conflict has been resolved. With the override, Rule 2 now means “Use oversized containers when volume is >30, **even when** weight is <=20000.”
4. Dismiss the dialog box by clicking **OK**.
5. Save your Rulesheet by clicking on the **Save** icon on the toolbar or by choosing **File>Save**.

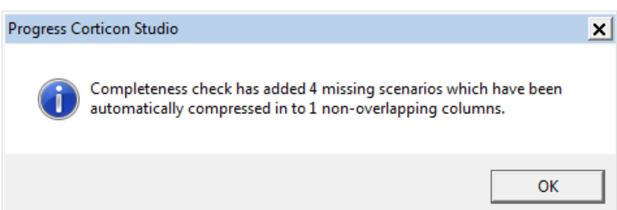
Check for completeness

Conflict is one form of logical error. Another form is incompleteness in the logic.

To see if the rules are complete, select **Rulesheet > Logical Analysis > Check for Completeness**.



A message window opens informing you that the rules are incomplete. You missed some scenarios.



The completeness checking algorithm calculates the set of all possible combinations of values in all conditions. The algorithm then compares this set of possible combinations to those already specified in the Rulesheet and automatically inserts missing combinations of conditions as new columns. These new columns are highlighted in green.

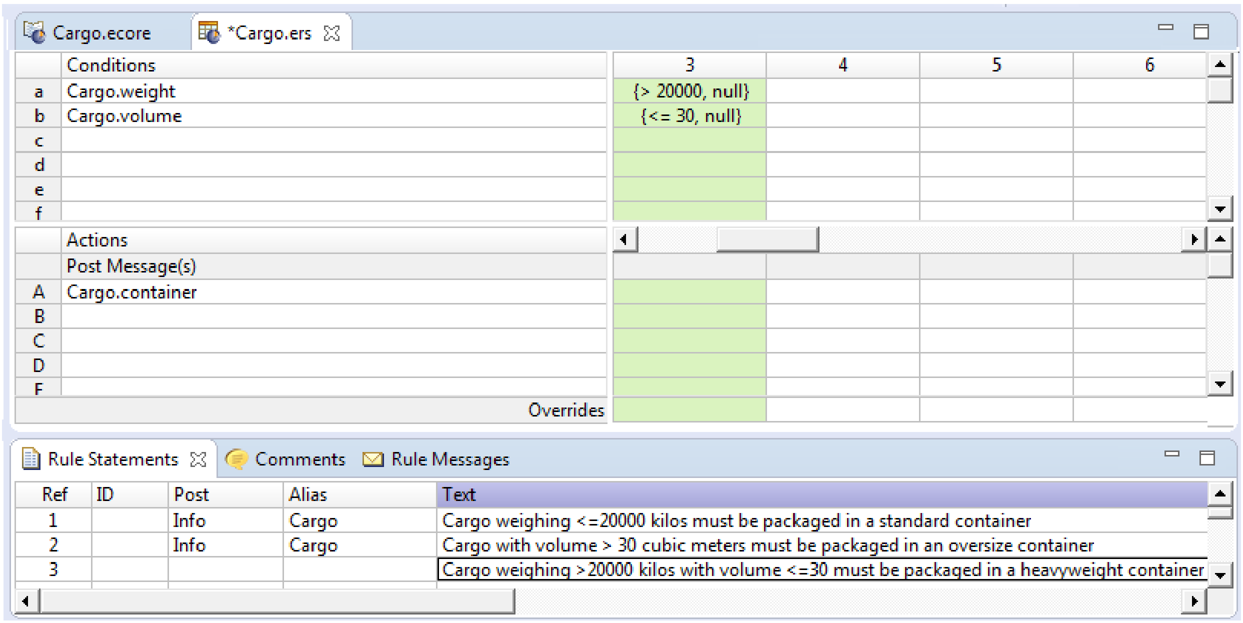
In this case, Corticon Studio has added a new rule in column 3—where the cargo weighs > 20000 and the cargo volume is less than or equal to 30. The completeness check adds condition values, but does not choose actions—leaving it to the rule modeler.

Click **OK** to dismiss the window.

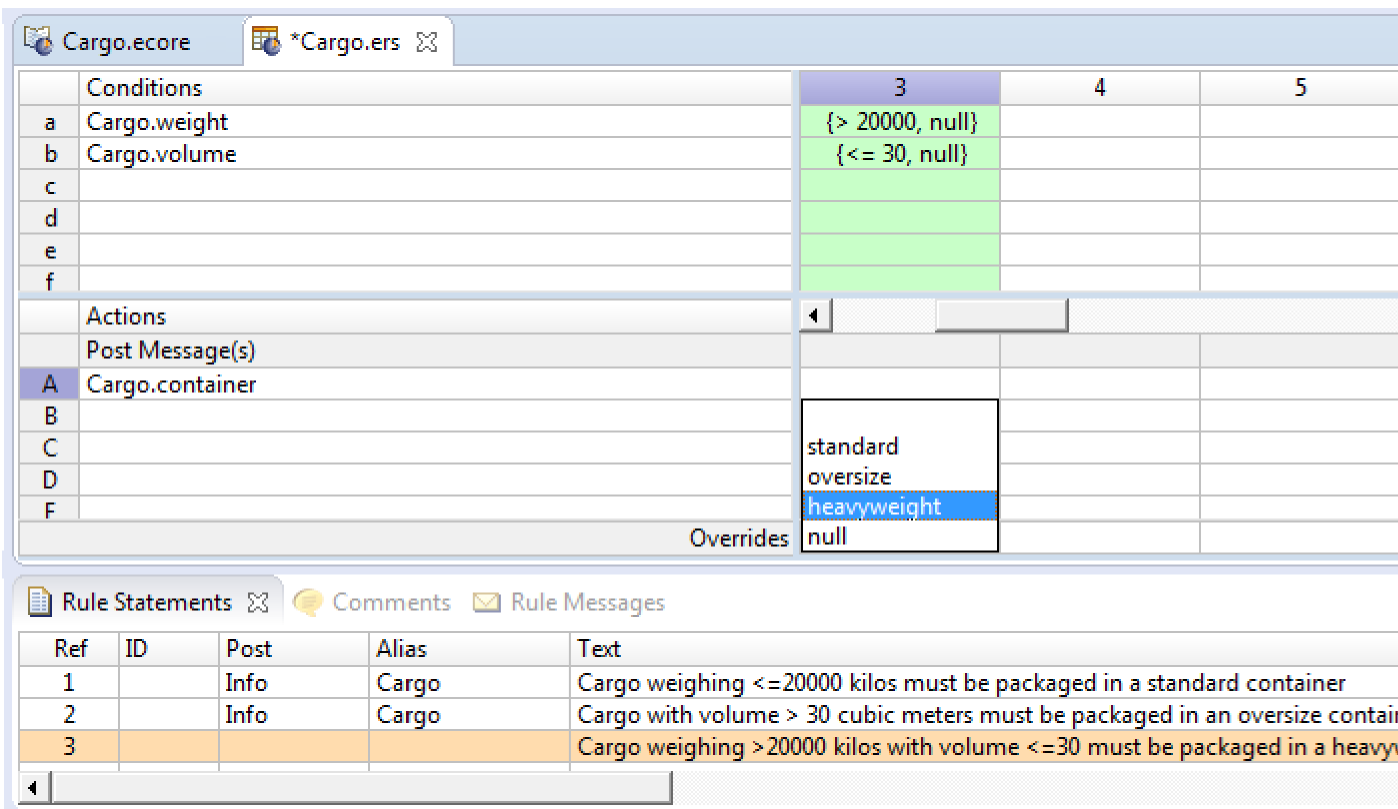
Resolve completeness errors

1. Add a new rule statement for Rule 3: **Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be packaged in a heavyweight container.**

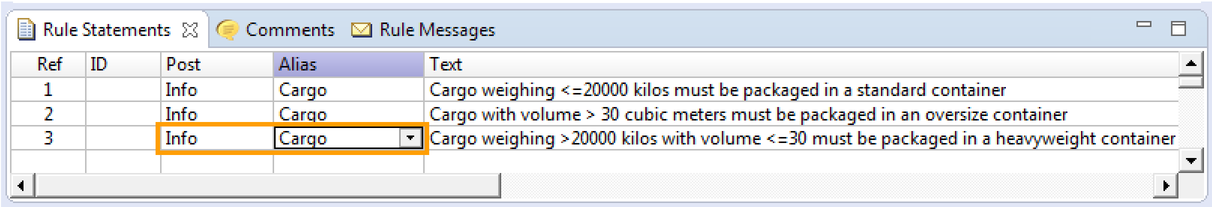
Note: Do not forget to link the rule statement with the corresponding column.



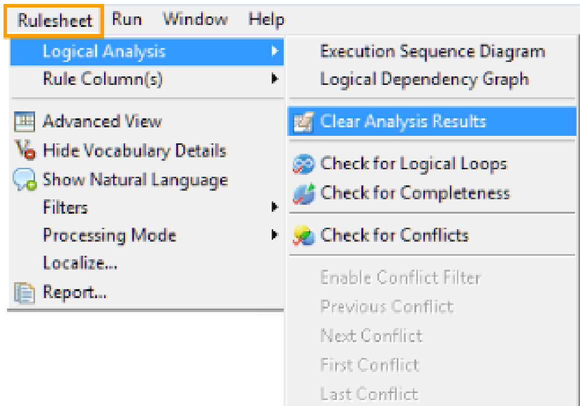
2. Define an action in rule cell 3A. In this case, select **heavyweight** as the container option.



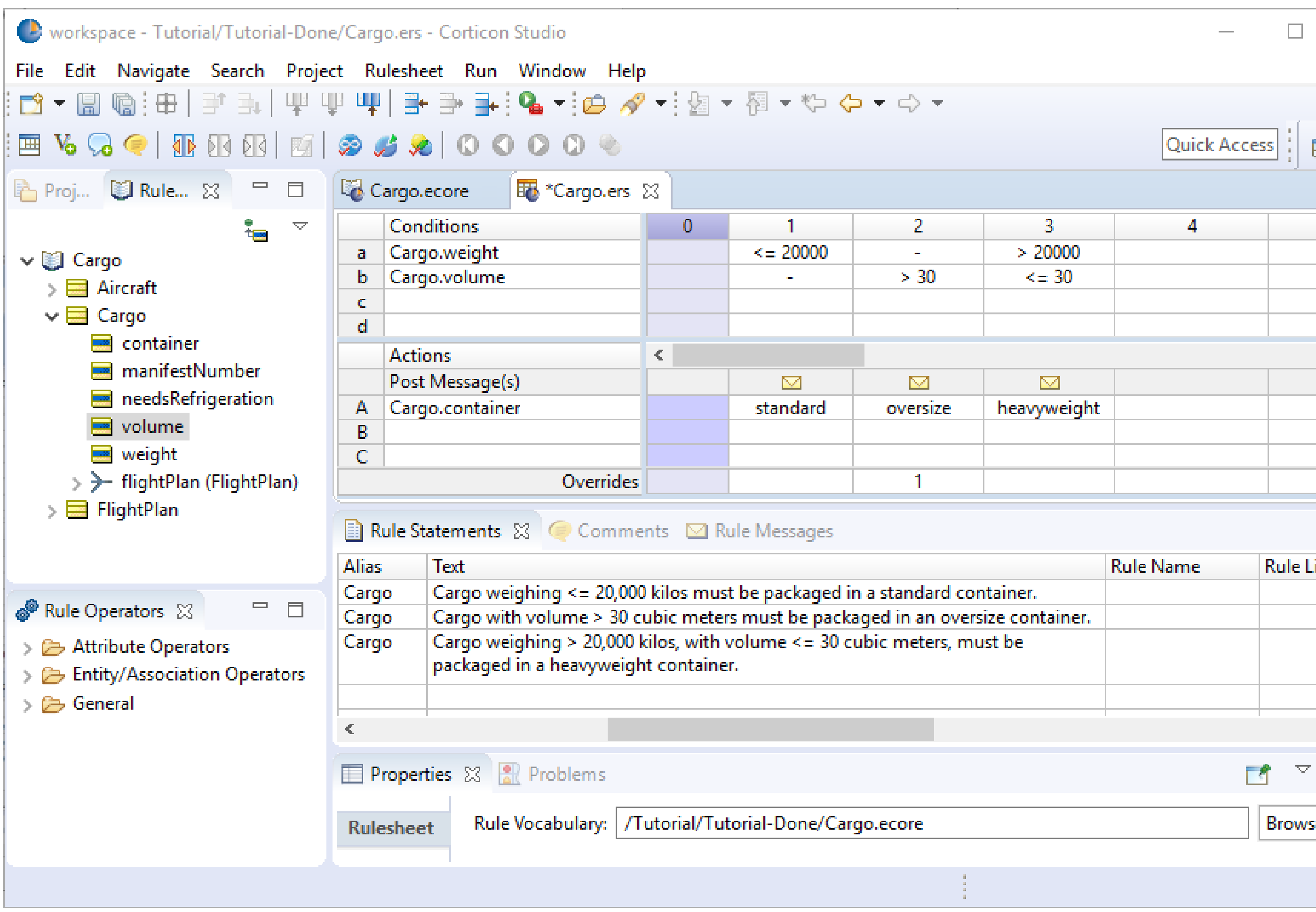
3. Post an Info message to the Cargo entity as you did for the first two rules in the Rulesheet.



4. Select **Rulesheet > Logical Analysis > Clear Analysis Results** to remove the highlighting in Rule 3.



After you clear analysis results, your Rulesheet looks like this.



5. Run the completeness check again. The dialog box should indicate that the Rulesheet is complete.
6. Click **OK** in the dialog box.

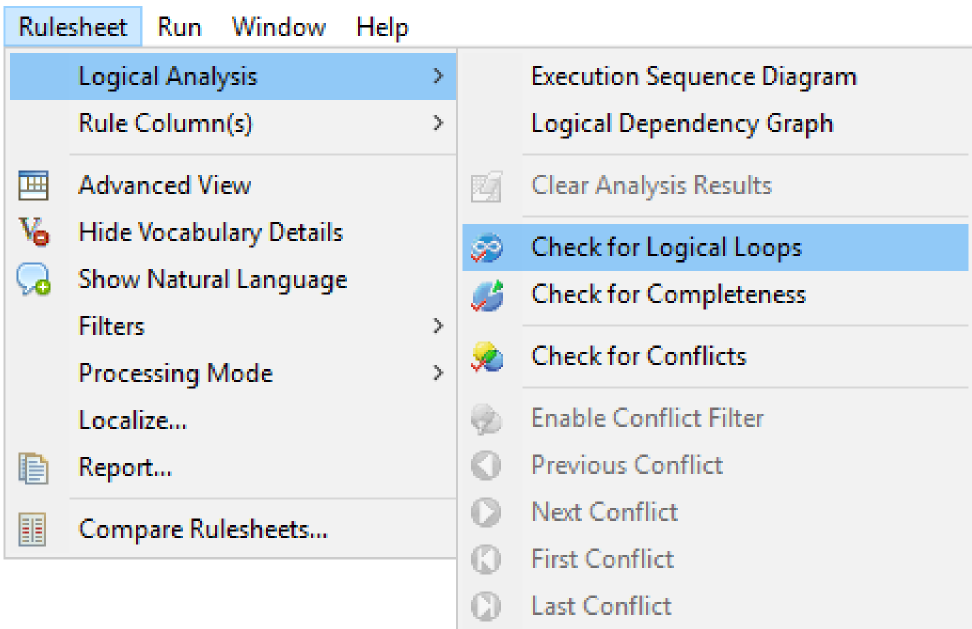
Note: Although checking for completeness can identify rules that you should include in your Decision Service, there may be situations where you do not want a newly-added rule. In this case, you can just delete the rule.

---

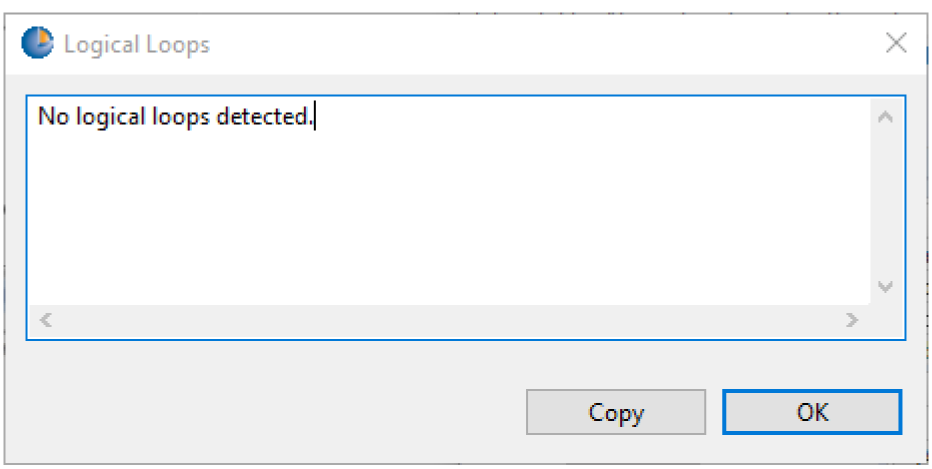
7. Save your Rulesheet.

Check for logical loops

A third form of logical error is circular logic or loops. To check for this, select **Rulesheet > Logical Analysis > Check for Logical Loops**.



You see the following result.



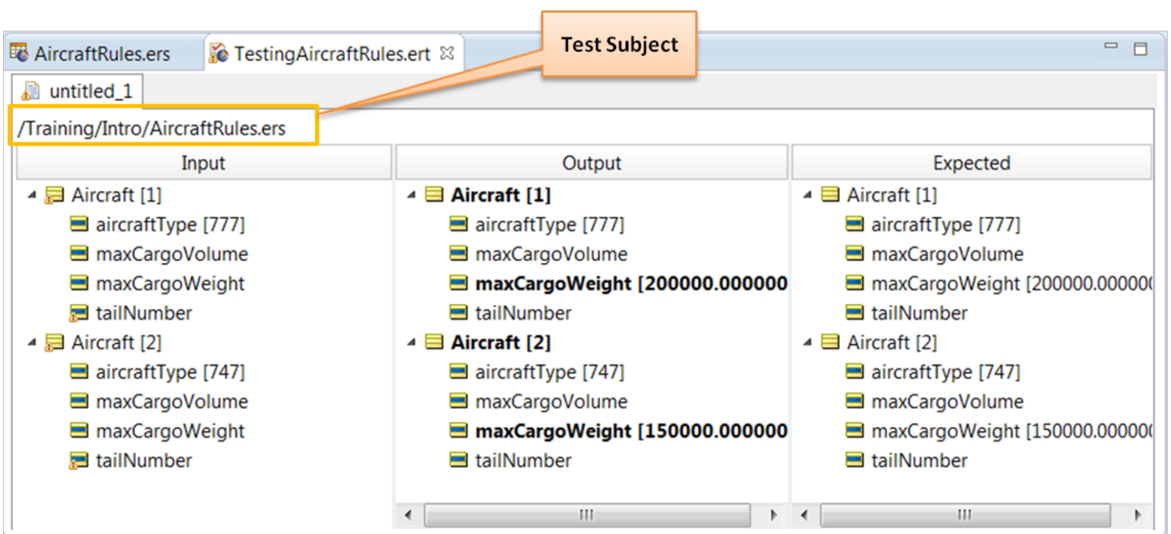
This Rulesheet is very simple and contains no logical loops. Click **OK** in the dialog box to dismiss it.

Note: While unintended logical loops should be fixed, sometimes logical loops are a useful technique for implementing rule logic that requires recursive reasoning.

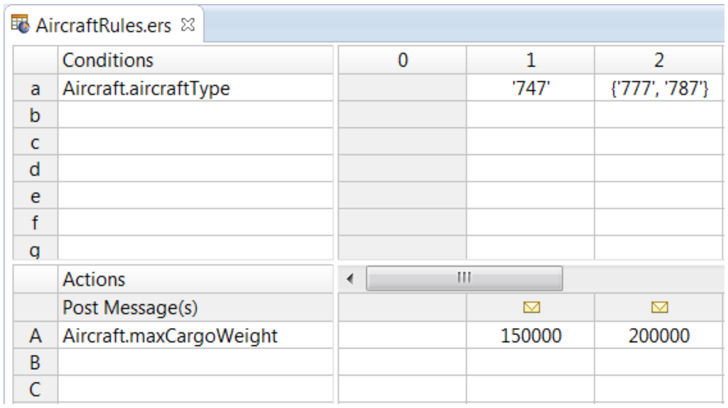
4

What is a Ruletest?

A Ruletest simulates a business scenario.



The Ruletest input data is evaluated by the rules.



If the data satisfies all the conditions in a rule, the rule fires and some output containing the results of the rule execution is produced.

You can define different sets of input data to test how the rules behave in different scenarios. You can also use a Ruletest to compare the output of a rule execution with expected results.

A Ruletest stores this information in a Ruletest file, enabling you to save use-cases that are of interest, change rules, and run the test again to see how the modified rules behave when applied to the same use-cases.

The images show an example of a Rulesheet (`AircraftRules.ers`) with two rules and a Ruletest (`TestingAircraftRules.ert`) that has been executed.

The Rulesheet contains rules that define a value for `Aircraft.maxCargoWeight` based on the value of `Aircraft.aircraftType` received in input data.

The Ruletest tests the Rulesheet. The Ruletest editor has four parts:

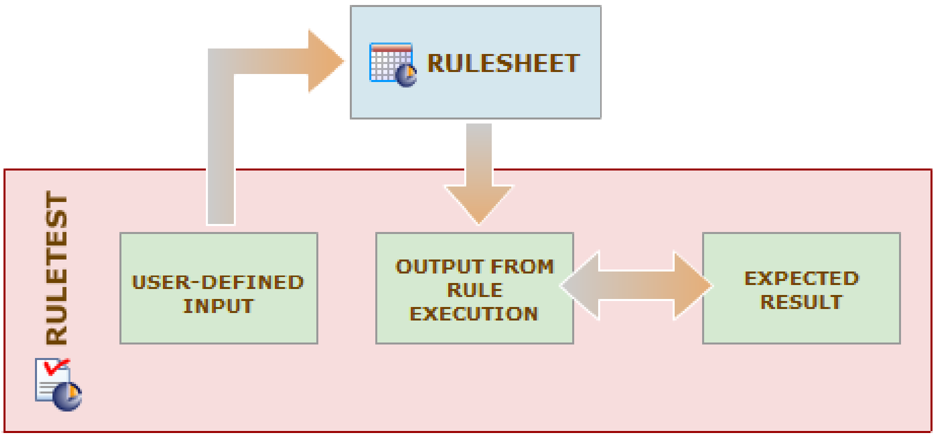
- **Test Subject**—Specifies which Rulesheet or Ruleflow is being tested (in this case AircraftRules.ers).
- **Input**—Where you define input data to be processed by the rules in the Rulesheet. This example has two instances of the **Aircraft** entity with different values for **Aircraft.aircraftType**.
- **Output**—Where Corticon Studio displays the result of a Ruletest execution. A value for `Aircraft.maxCargoWeight` has been assigned based on the rules in the **AircraftRules.ers** Rulesheet.
- **Expected**—where you can optionally define your expected result.

For details, see the following topics:

- How a Ruletest works
- Tasks to test rules in a Rulesheet using a Ruletest
- Create a Ruletest file
- Specify input for a Ruletest
- How to specify expected results
- How to run a Ruletest
- Compare the output with expected results
- Embed dynamic data in a rule statement
- How to link a rule statement to multiple rules

How a Ruletest works

When you run a Ruletest, the input values are used to generate output that you can compare with your expected results.



These steps details how the rules generate the output values:

1. Input data is processed by the rules in the Rulesheet.
2. If the input data satisfies all the conditions in one or more rules, those rules fire. The Ruletest then displays output that could include the same data as the input but with changed values, additional data and values. or both.
3. If the input data does not trigger any rules, the Ruletest displays the unchanged input data as the output.
4. If you specify expected results for a test, the Ruletest automatically compares the output with the expected results and displays any differences through color-coding.

Tasks to test rules in a Rulesheet using a Ruletest

Perform the following tasks to test rules using a Ruletest:

1. Create a Ruletest file.
2. Specify inputs for the test.
3. Optionally, specify expected results to compare with the output of a test execution.
4. Run the test.
5. Run the ruletst with Rule Trace view to see te

Create a Ruletest file

A Ruletest file must be created under a rule project and must refer to a Rulesheet. A Ruletest file has the extension `.ert`.

Follow these steps to create a Ruletest file:

1. In Corticon Studio, select **File > New > Ruletest**.
2. In the **Create New Ruletest** wizard:
   a. Ensure that the **Enter or select parent folder** field contains the path where you want the Ruletest created.
   b. In the **File name** field, enter the name of the Ruletest .
   c. Click **Next**.
3. In the **Select Test Subject** page:
   a. Ensure that the correct Rulesheet (or Ruleflow) is selected on the **Run in Studio** tab.
   b. Click **Finish**.

The newly created Ruletest appears in the **Rule Project Explorer** view, and opens in the Ruletest editor, by default.

Specify input for a Ruletest

In the business world, a Corticon Decision Service may receive input in different formats such as JSON and XML. However, in a Ruletest, you must specify input data in the Corticon data format.

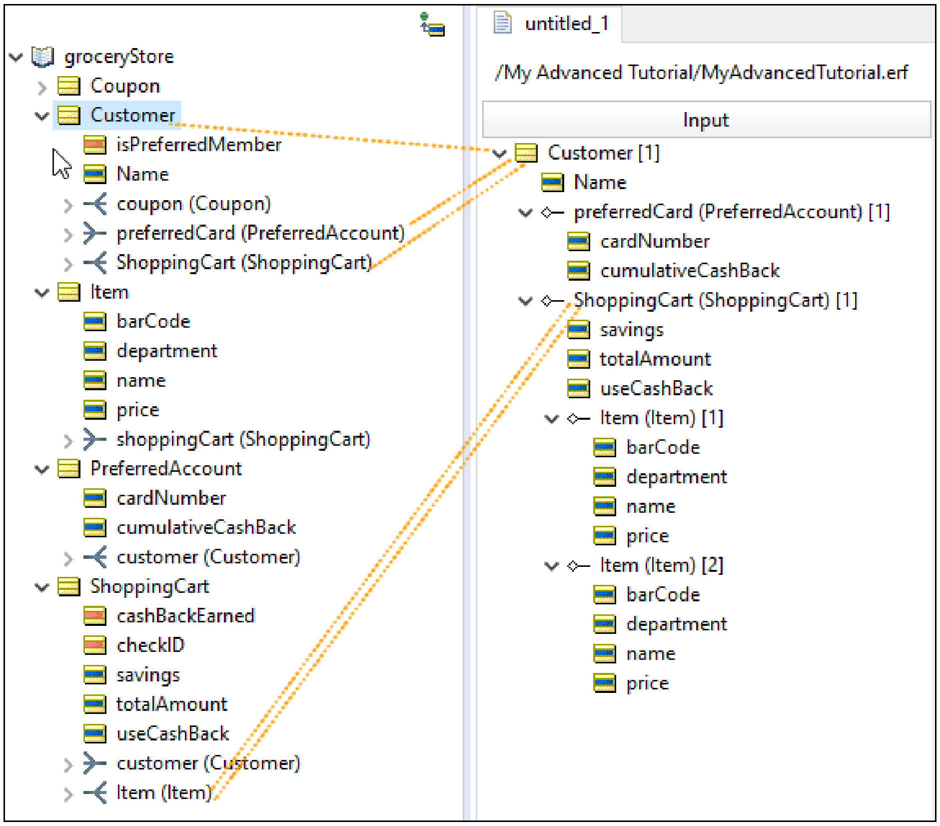
You specify inputs for a Ruletest by:

1. Dragging entities from the **Rule Vocabulary** view, and dropping them into the **Input** pane. Each time you drag and drop an entity, you create an instance of the entity, adding all the attributes in the entity to the **Input** pane. However, you can delete attributes that you do not require.
2. Specifying values for the attributes by double-clicking name and entering its value. The syntax for specifying these values is similar to that of specifying attribute values in Rulesheets, with one difference—you must not enter any values in quotes, even if the data type of the attribute is String, DateTime, Date or Time. If you enter quotes, the Ruletest treats it as part of the value.

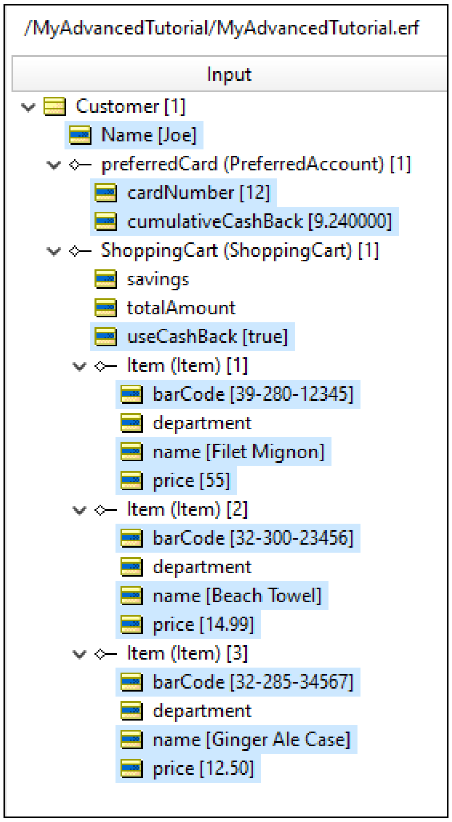
   Note: If you break any of these rules while specifying attribute values in the **Input** pane, Corticon Studio indicates a mismatch with a warning icon.

You can specify different entities as well as multiple instances of the same entity. When you specify multiple instances of the same entity, each entity is assigned an identity number. The identity number is 1 for the first instance of the entity, and is incremented by one for each additional instance that you drag and drop.

In the following example, several entities form the request from associations. The customer is the primary entity. The customer's association with a preferred account and with a shopping cart are what you want, and then multiple associations between items that are in the customer's shopping cart:

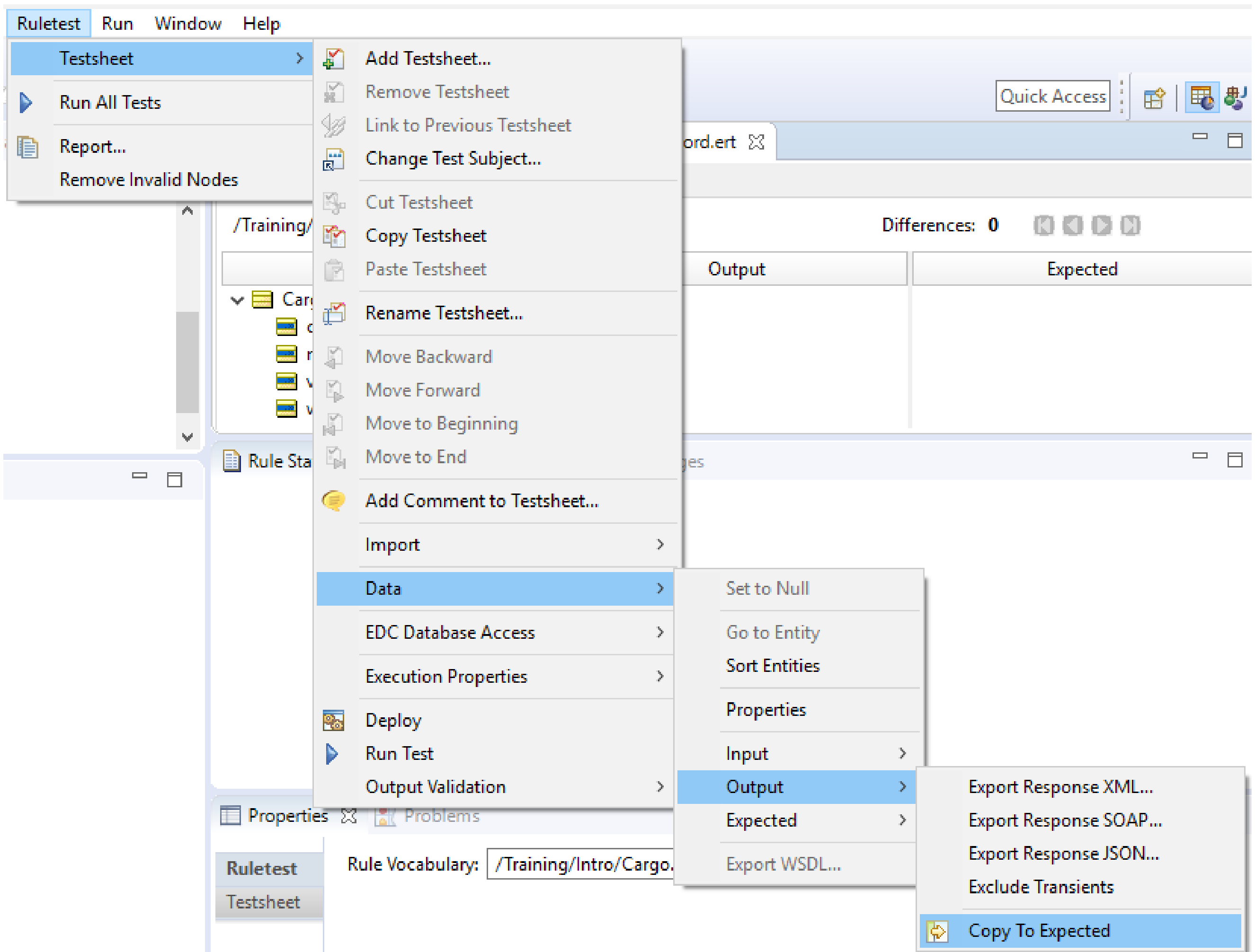


When you enter values for the Ruletest, it is ready to test:



How to specify expected results

You can optionally define expected results in the **Expected** pane by dragging elements from the **Rule Vocabulary** view, dropping them in the **Expected** pane, then specifying values, like you do while defining input data.



Note: After you run the Ruletest, you can copy data from the **Output** pane to the **Expected** pane if you want to make some changes to the rules and see if the output differs from the output of the previous test. To do it, select **Ruletest > Testsheet > Data > Output > Copy To Expected** or click the **Copy To Expected** button.

How to run a Ruletest

When you run a Ruletest:

- If the Ruletest is being run on the Rulesheet for the first time, or if the Rulesheet has been modified since the last test execution, Corticon compiles the Rulesheet.
- Corticon applies the rules in the Rulesheet to the input data.

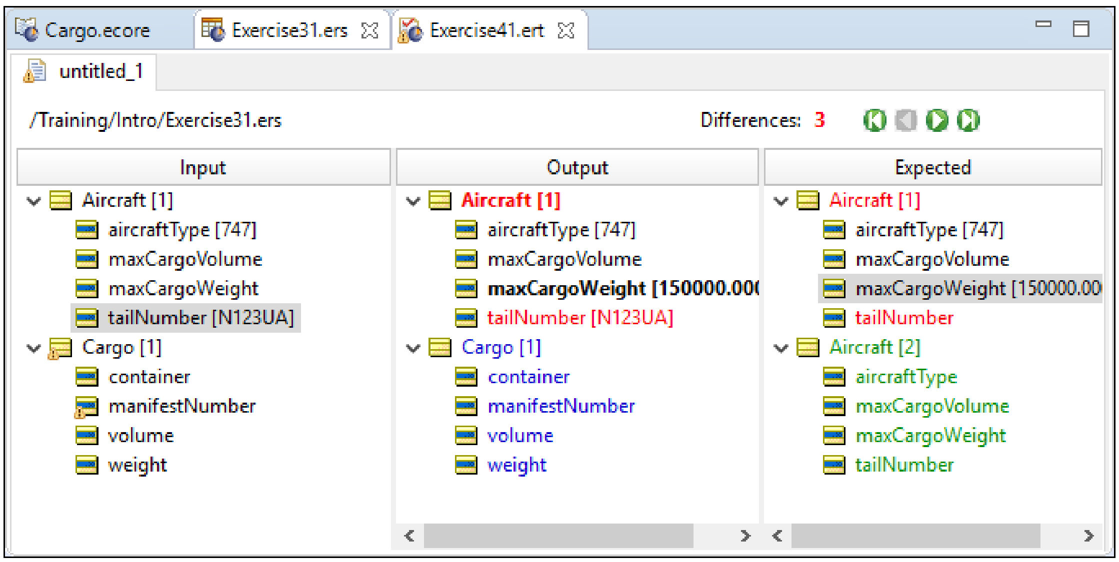
To run the Ruletest:

1. Ensure that the Ruletest is open.
2. Click the **Run Test** button.

The **Output** pane displays the result of the Ruletest execution. Any elements that are modified or added (as a result of the actions in the rule) are highlighted in bold. Any rule statements linked to the rule and configured for posting are displayed in the **Rule Messages** view.

Note: If no rule fires, the **Output** pane displays the same elements and values as in the **Input** panel, without any highlights. The **Rule Messages** view remains blank.

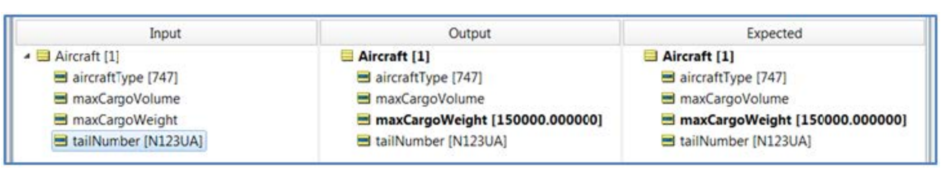
Any differences between the **Output** pane and the **Expected** pane are highlighted through color codes as shown in this image.



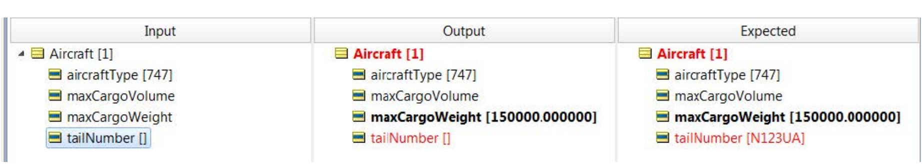
Compare the output with expected results

The Ruletest displays differences between the Output pane and the Expected pane as follows:

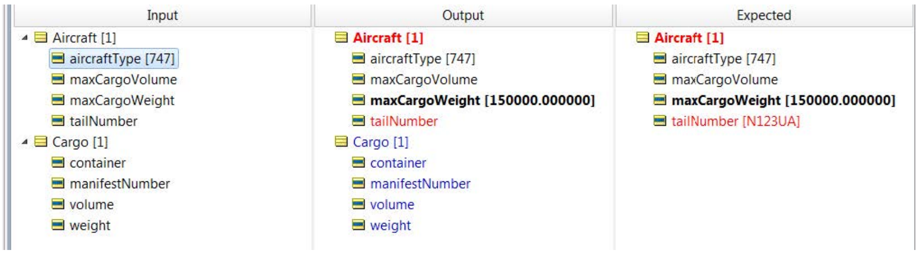
- If there are no differences, both sets of data are shown in black, in both of the panes.



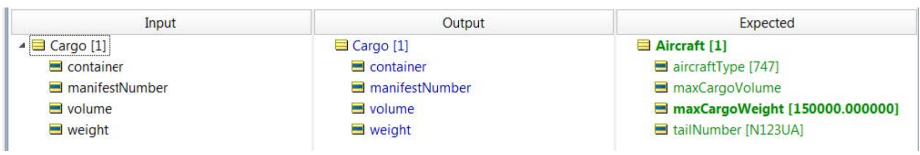
- If one or more attribute values differ for the same entity instance, the entity and the attributes are shown in red, in both panes.



- If additional entity instances are produced in the output (that do not exist in the **Expected** pane), they are shown in blue in the **Output** pane.

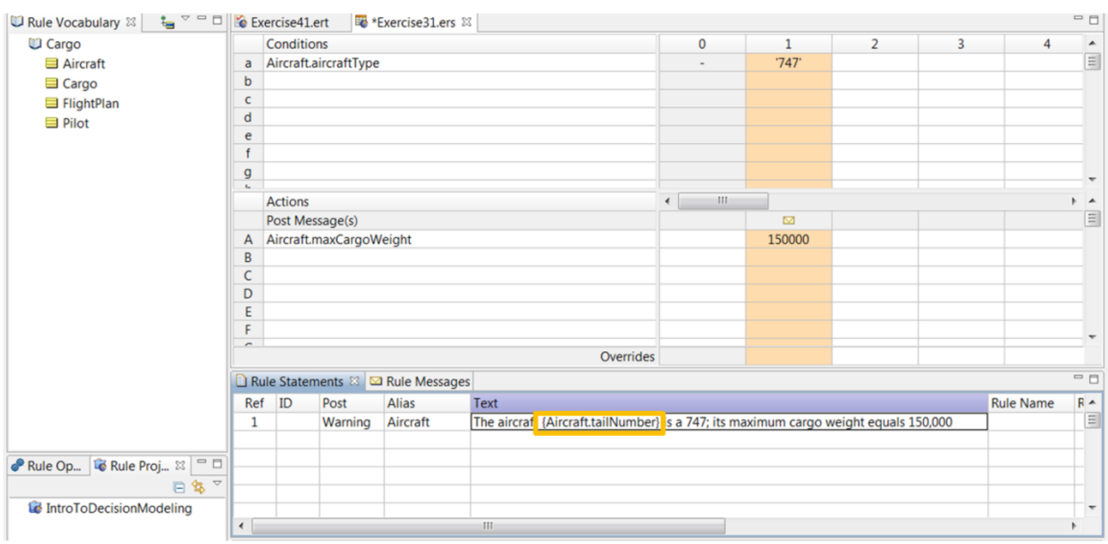


- If the **Expected** pane contains entity instances that are not produced in the **Output** pane, they are displayed in green in the **Expected** pane.
-



Embed dynamic data in a rule statement

Rule statements in a Rulesheet can be used to generate rule messages. Rule messages need not contain only static information. They can also contain dynamic data such as the values of attributes (in request messages received by Corticon, or created or modified as the result of a rule execution at runtime).



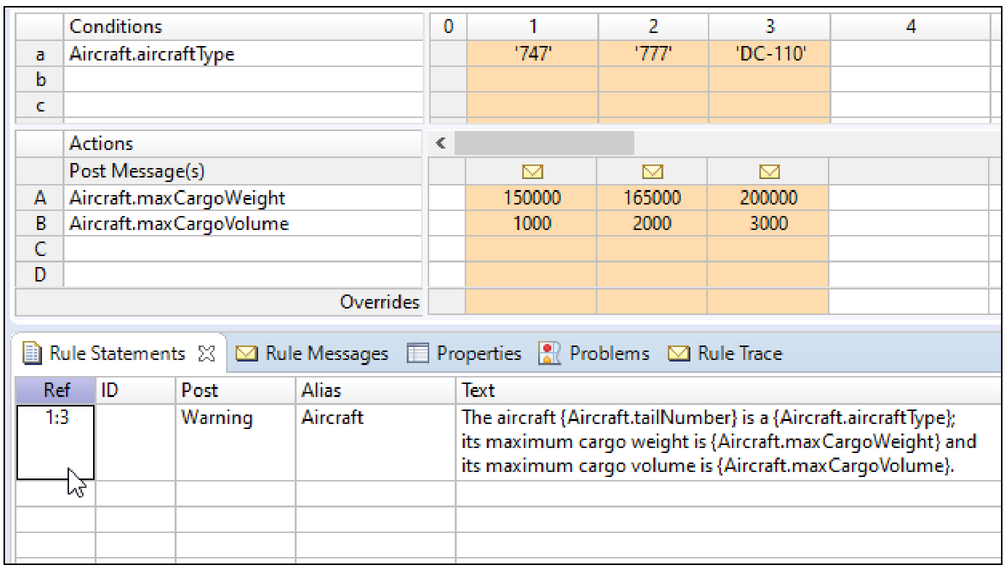
For example, the rule statement *If an aircraft is of the type 747, the maximum cargo weight is 150,000* could be modified to include the aircraft's tail number when a rule message is generated: *The aircraft N123UA is of the type 747; its maximum cargo weight is 150,000.* Here, `N123UA` is the value of the attribute **tailNumber** in a request message received by Corticon at runtime.

You can embed dynamic data in rule messages by specifying the attribute name within the rule statement. To specify an attribute name in a rule statement:

1. Open the Rulesheet.
2. In the **Rule Statement**s view, specify the attribute name by:
   a. Dragging it from the **Rule Vocabulary** view and dropping it to the appropriate place within the rule statement.
   b. Enclosing it in curly braces to indicate that its value will be populated at runtime.

How to link a rule statement to multiple rules

Using dynamic data, you can create a single rule statement that describes multiple rules. You can then link the rule statement to each of the rules. When any of the rules fire, attribute names is replaced by actual values in the rule message.



For example, if you have three rules, each specifying the maximum cargo weight for a certain type of aircraft, then you can use a a single rule statement that embeds the aircraft tail number, the aircraft type, and the maximum cargo weight to cover all three rules.

Here are two common methods:

- <ColumnNumber>:<ColumnNumber>—Specifies a range of rule columns, for example 1:3, where 1 and 3 are inclusive. When any of these rules fires, the rule statement is posted as a rule message, as shown in the previous image above.
- {<ColumnNumber>, <ColumnNumber>,...}—Specifies a list of rule columns, for example {1, 3}. When any of the rules in the list fires, the rule statement is posted as a rule message.

5

Enhance rules

Comprehensive business rules require a rule modeling tool that supports the necessary comparison operators, equations, calculations, and logic operation required to capture the complexity of your business scenarios. In this content, you explore the following ways to enhance your rules.

For details, see the following topics:

Action-only rules

As you develop rule models in Corticon, you may encounter situations where you require a rule that always fires regardless of conditions, such as a rule that sets a default value for an attribute. For example, a transportation company may require a rule that sets the maximum cargo weight for an aircraft, regardless of the aircraft type. You can address situations like this by defining action-only rules.

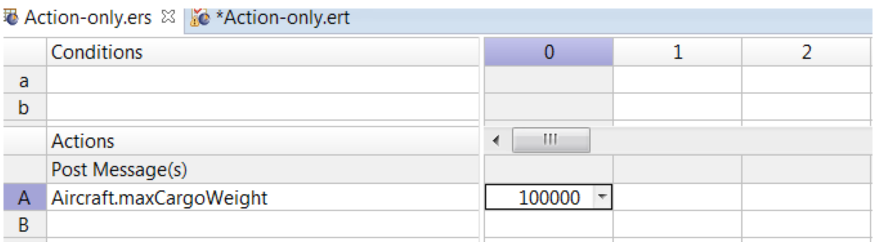
You define an action-only rule in column **0** of a Rulesheet. The cells that correspond to the Conditions panel in column 0 are grayed out. This indicates that you cannot specify any conditions in column **0**. However, the cells that correspond to the **Actions** pane in column **0** are editable, enabling you to specify actions that always fire, regardless of conditions.

You can use column **0** to:

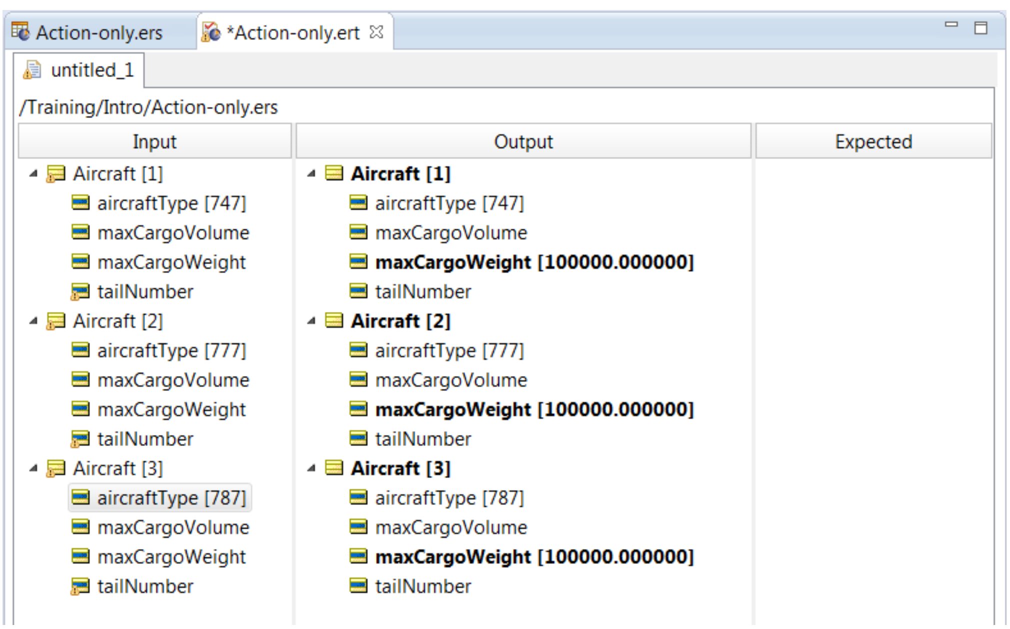
- Specify equations and calculations.
- Assign default values to attributes.
- Specify any action that you want performed whenever the Rulesheet is processed.

Note: Each **Action** row in column **0** is a separate rule and can have its own rule statement.

To define an action-only rule, drag the required attribute to an action row, and drop it in an **Action** row.. Then, enter the default value for the attribute in corresponding cell in column **0**.



An action-only rule always fires as the following Rulesheet demonstrates:



Comparison operators

You define rule conditions by specifying a Vocabulary term in a conditions row and entering a value in a corresponding rule column cell. When you do it, the condition automatically assumes the equality **=** operator.

	Conditions	0	1	2
a	Aircraft.aircraftType	-	'747'	
b				
c				
	Actions			
	Post Message(s)			
A	Aircraft.maxCargoWeight		150000	
B				
C				
	Overrides			

For example, when you specify `Aircraft.aircraftType` in a cell in the **Conditions** pane and enter `747` in a corresponding rule column cell, you define the condition `Aircraft.aircraftType = '747'`.

Note: When you enter a numeric value into a cell for String data type, the value is automatically expressed in single quotation marks.

However, as you model rules, you may require different types of operators. When you specify a condition that checks if an attribute value is greater than or less than a certain value—for example, `cargo volume > 1000` or `cargo weight < 100000`—use a comparison operator from the library of rule operators that Corticon provides.

	Conditions	0	1
a	Cargo.volume		>= 1000
b	Cargo.weight		< 100000

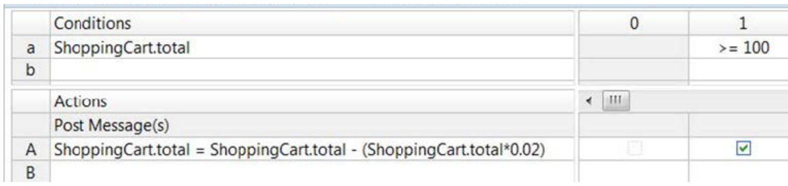
The types of comparison operators that you can use in a condition depend on the data type of the attributes in the condition. For instance, for an Integer data type, you can use operators such as >, <, >=, and <=. To check which operators are applicable for each data type, refer to the **Attribute Operators** category in the **Rule Operators** view in Corticon Studio.

You can use a comparison operator in a rule by either:

- Double-clicking the rule column cell, and then entering the operator followed by the value.
- Dragging the operator to the rule column cell, and then entering the operator followed by the value.

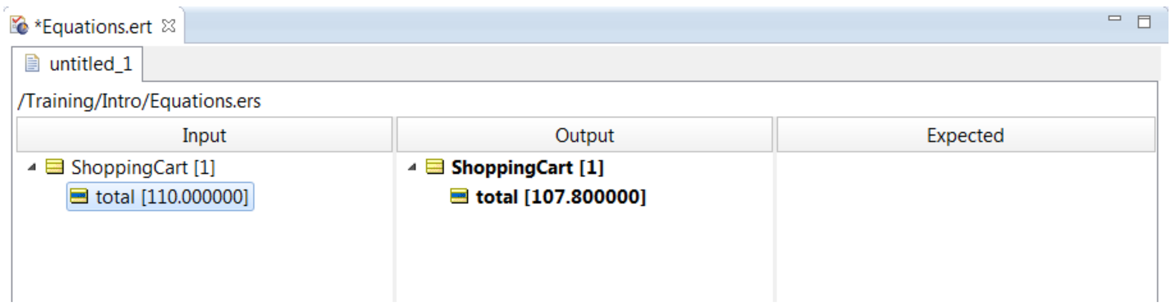
Equations and calculations

As you model rules, you may need to derive the value of an attribute from a calculation. For example, in a supermarket where a customer gets a discount of 2% if the total price of items in their shopping cart is greater than or equal to 100. To facilitate it, you would need a rule that checks if the total price of all items is greater than 100 and if so, deduct the discount from it to arrive at a new total. This rule would look like:



When you define an equation in an action row , the corresponding cells in rule columns turn into checkboxes. You must select the checkbox in the appropriate column (in this example, column **1**) for the action to be triggered.

When the Rulesheet is processed at runtime, Corticon populates the attribute values on the right side of the equation, performs the calculation, and assigns the result of the expression to the attribute on the left side of the equation.



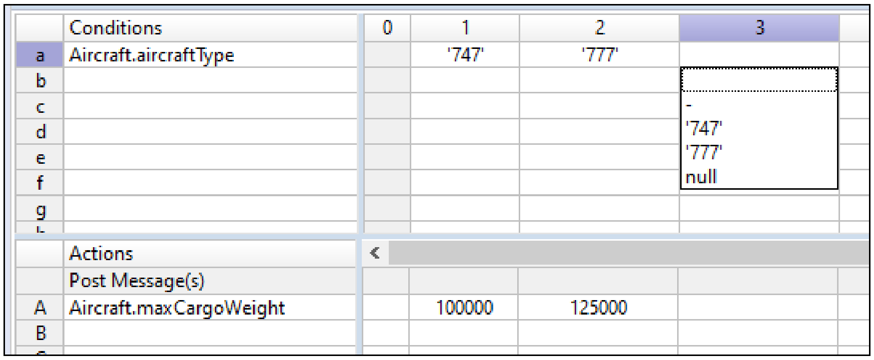
Value sets and ranges

You can use values in rule columns in two ways:

- Value sets
- Value ranges

Value sets

When you define values for a condition or an action spanning multiple rule columns, you build a value set. A value set is a set of distinct values for a condition or an action. For example, in the following image, the condition `Aircraft.aircraftType` has two values in its value set. The action `Aircraft.maxCargoWeight` also has two values in its value set. As you build a value set by defining values across different rule columns, each cell in the same row becomes a drop-down list when you click it, displaying all the values that you enter in the value set.



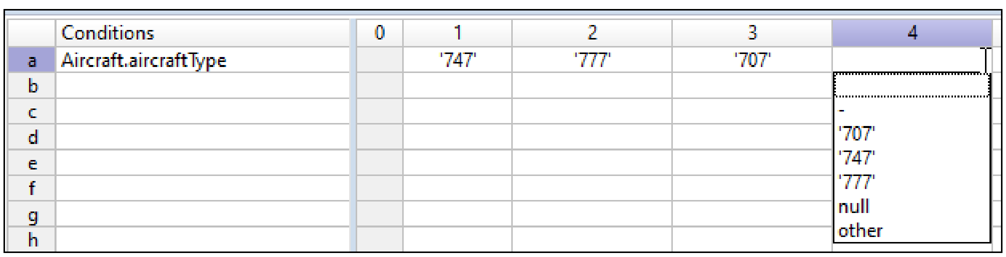
Note: The value null in the drop-down is added when the properties of a attribute are not mandatory—there might be no value entered.

How to use other in a condition value set

Other is a special type of operator that helps you recognize that other values may be unaccounted for, and gives you a simple way to detect them in rules. You can think of it as the operator that provides the ELSE clause of an IF-THEN-ELSE rule. It is available only for value sets that you build in **Condition** rows.

You can enter `other` as a cell value. It is not a String value, so single quotation marks are not required.

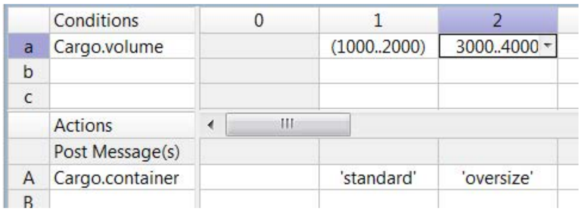
After you enter three values for a condition's value set, a value named **other** is added to the list of choices in the drop-down list, as shown in this image.



In this example, the value set for `Aircraft.aircraftType` has three values—`'747'`, `'777'`, and `'787'`. However, if data is received with an aircraft type outside this value set, you can choose **other** in a rule that sets the maximum cargo weight of any aircraft that is not a 747, 777, or a 787.

Value ranges

A common requirement while defining rules is to specify conditions that check for a range of values in a single rule column cell. For example, a rule that checks if the cargo volume is between 1000 and 2000 (a range of values), and assigns a container type. You specify a range of values by entering the starting value, followed by two dots or periods, followed by the ending value—for example `1000..2000`—as shown in the image.



Enclose the value range in parentheses or square brackets as follows:

Syntax example	Description
`1000..2000`	No square brackets or parentheses specifies a range between 1000 and 2000 and includes the values 1000 and 2000.
`(1000..2000)`	Enclosed in parentheses specifies a range between 1000 and 2000, but excludes the values 1000 and 2000.
`[1000..2000]`	Enclosed in square brackets specifies a range between 1000 and 2000 and includes the values 1000 and 2000.
`(1000..2000]`	An opening parenthesis and a closing square bracket specifies a range between 1000 and 2000, excludes 1000, but includes 2000.
`[1000..2000)`	An opening square bracket and a closing parenthesis specifies a range between 1000 and 2000, includes 1000, but excludes 2000.

Note: You must use a square bracket and a parenthesis together. You cannot start or finish a range with a square bracket or a parenthesis and leave the other end of the range empty. For instance, specifying `1000..2000)` results in an error.

Boolean conditions

A Boolean condition is any condition that returns true or false—or, if data for the attribute is not mandatory, **null**. For example, you may want to specify a condition that checks if the weight of a cargo container is less than 5000. To define a Boolean condition using a comparison operator, specify the condition expression in the **Conditions** row.

	Conditions	0	1	2
a	Cargo.weight < 5000		T	-
b	Cargo.volume > 10000		-	T
c				
	Actions			
	Post Message(s)			
A	Cargo.container		'standard'	'large'
B				
C				

All the cells in that row (across all rule columns) display three choices in a drop-down list—**T** (true), **F** (false), and hyphen. Corticon automatically inserts a hyphen in other cells indicating that they do not play a role in the Boolean condition.

-
T
F
null

Boolean conditions versus value sets

Some conditions can be expressed as Boolean conditions or through value sets. For example, a condition that checks if the aircraft type is 747 can be expressed as a Boolean condition or through a value set with a single value, as shown in this image.

	Conditions	0	1
a	Aircraft.aircraftType		'747'
b			
	Actions		
	Post Message(s)		
A	Aircraft.maxCargoWeight		100000
B			

=

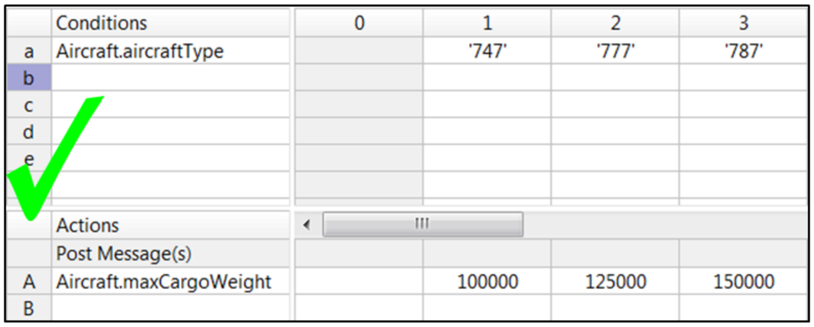
	Conditions	0	1
a	Aircraft.aircraftType='747'		T
b			
	Actions		
	Post Message(s)		
A	Aircraft.maxCargoWeight		100000
B			

So when should you use a Boolean condition and when should you build a value set?

If you only have to define a rule that checks for a single value, as in the first example, you can use either approach.

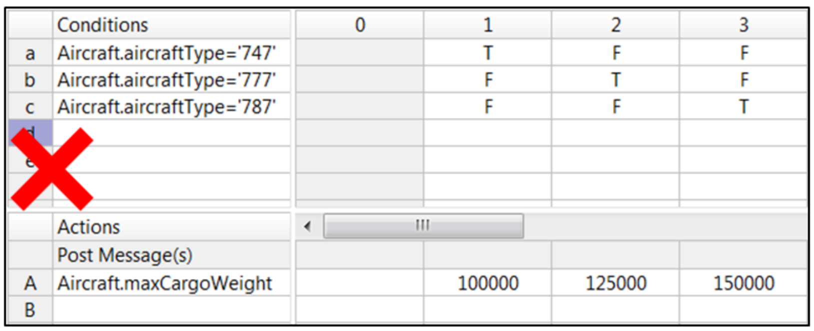
However, if you have to define rules that share a common pattern and check for multiple values for an attribute, it is better to use value sets.

The following image has a good, readable pattern.



	Conditions	0	1	2	3
a	Aircraft.aircraftType		'747'	'777'	'787'
b					
c					
d					
e					
	Actions				
	Post Message(s)				
A	Aircraft.maxCargoWeight		100000	125000	150000
B					

The following image shows a pattern that should be discouraged. You need to parse all the values to understand the logic that was clear in the first example.



	Conditions	0	1	2	3
a	Aircraft.aircraftType='747'		T	F	F
b	Aircraft.aircraftType='777'		F	T	F
c	Aircraft.aircraftType='787'		F	F	T
d					
e					
	Actions				
	Post Message(s)				
A	Aircraft.maxCargoWeight		100000	125000	150000
B					

Logical structures

You can define logical structures in rules:

- Logical AND on page 56
- Logical OR on page 57
- Logical NOT on page 58

Logical AND

A common requirement when you define rules is to specify AND logic. For example, consider a situation where a rule must assign a maximum cargo weight for an aircraft only when the aircraft type is a 747 AND the cargo volume is 4000. You achieve it by defining two conditions and entering their values in the same rule column, as shown in this image. The rule fires only when all the conditions are satisfied.

	Conditions	0	1
a	Aircraft.aircraftType	-	'747'
b	Cargo.volume		4000
c			
	Actions		
	Post Message(s)		
A	Aircraft.maxCargoWeight		150000
B			
	Overrides		

Logical OR

Just like AND, a common requirement is to use OR logic in a Rulesheet.

Specify OR logic in either of two ways:

- **Through multiple rules**—Used when the OR logic requires two or more conditions that are different, as shown here:

  OR_Logic_MultipleRules.ers

	Conditions	0	1	2
a	Cargo.volume		> 2000	-
b	Cargo.weight		-	> 100000
c				
	Actions			
	Post Message(s)			
A	Cargo.container		'oversize'	'oversize'
B				

  After defining the rules, when you save the file, Corticon Studio automatically detects the OR logic and adds hyphens in any empty cells that correspond to the defined conditions.

- **Through OR value sets**—Use when the OR logic needs to check for a list of values for a single attribute, you can enclose those values in curly braces in a rule column cell as shown here:

  OR_Logic_ValueSets.ers

	Conditions	0	1	2
a	Aircraft.aircraftType		{'747', '777', '787'}	{'DC-10', 'MD-10'}
b				
	Actions			
	Post Message(s)			
A	Aircraft.maxCargoWeight		100000	150000
B				

- **Comparison operators in an OR value set**—Use multiple values. For example if you specify `{<200, >600}`, any value that is less than `200` or greater than `600` satisfies the condition.

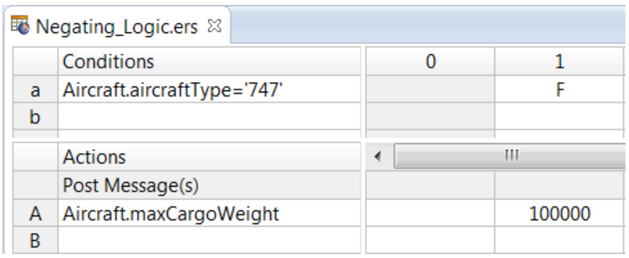
Note: You can specify an OR value set either by double-clicking a rule column cell and entering the value set or by pressing the `CTRL` key and selecting multiple values from the drop-down list (in which case, Corticon Studio automatically formats the OR value set).

Logical NOT

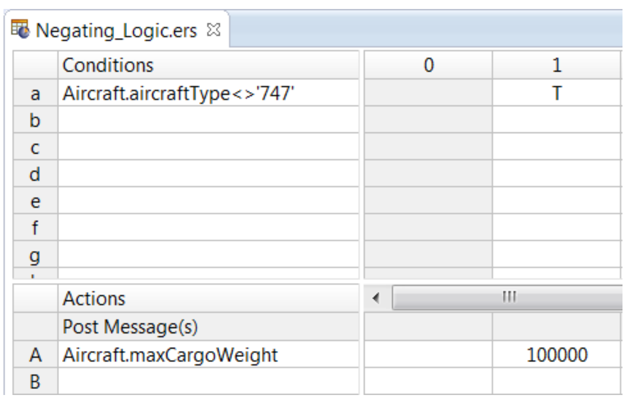
You can also use negating logic in Rulesheets when you want to define a rule that fires when the attribute is **NOT** a specified value. For example, if a rule assigns a maximum cargo weight to any aircraft that is not a 747, then the condition must be defined in such a way that the rule checks for cases where the value of the aircraft type attribute is not 747.

There are several ways to use logical NOT:

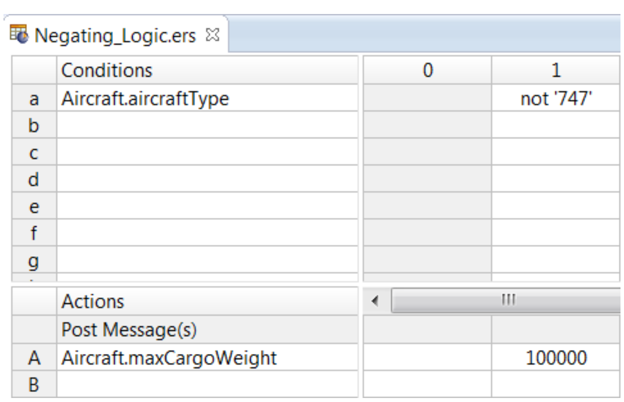
- Specify a Boolean condition expression with an equality operator (`Aircraft.aircraftType='747'`) that returns `False`.



- Specify a Boolean condition expression with an inequality operator (`Aircraft.aircraftType < >'747'`) that returns `True`. The inequality operator (`< >`) is available for all data types.



- Specify a condition (Boolean or value-set based) that checks if the value is `not` something. To do this, you must use the `not` operator, which is a Boolean operator available in the Attribute Operators > Boolean subcategory in the Rule Operators view.



In these examples, negating logic is expressed in different ways. You should choose whichever way is most intuitive to you and any stakeholders who may want to read and understand the rules in your Rulesheet.

Avoid multiple negatives

While a Corticon Rulesheet enables you to define double and triple negatives, defining multiple negatives is not a good practice. As in natural language, double and triple negatives are difficult for human readers to grasp, so any stakeholder who reads your rules may find them hard to understand.

For example, as shown in the image, you can create a rule in Corticon that checks if the aircraft type is a 747 using two negatives—`Aircraft.aircraftType <> '747'` and `F`. While this works the same way as `Aircraft.aircraftType = '747'` and `T`, it is harder to read.

	Conditions	0	1
a	Aircraft.aircraftType<>'747'	-	F
b			
c			
	Actions		
	Post Message(s)		
A	Aircraft.maxCargoWeight		100000
B			

Check for null values

You can specify a condition that checks for null in attribute values. Null means that no value has been provided for the attribute in the input data. Null is different from the value `0` which has significance in mathematical calculations, or an empty string that does not contain any characters, but is still treated as a value.

Check for a null value when you want to define a rule that assigns a default value if the value is null. You can also send back information in the response rule message if the value received is null.

You check for a null value by specifying null without any quotes in the condition expression, as shown in this image.

	Conditions	0	1
a	Aircraft.aircraftType		null
b			

	Conditions	0	1
a	Aircraft.aircraftType = null		T
b			

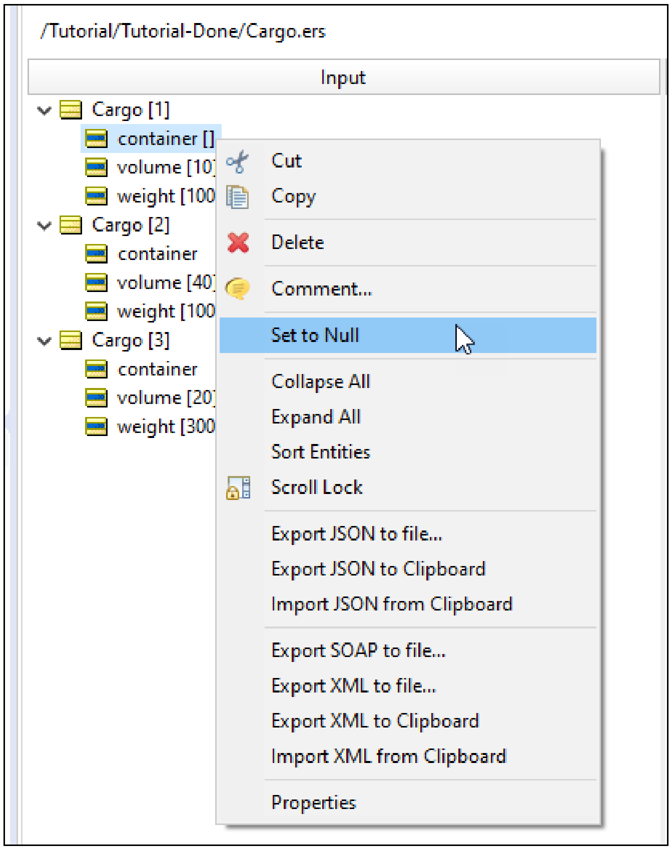
When you create an attribute, you can specify whether the attribute value is mandatory or a null value is acceptable. If you attempt to define a condition that checks for null when the attribute value is mandatory, you see a warning message.

Nulls in Ruletests

If you have to check for null in a rule condition, you must also understand how to specify null in Ruletests because you may want to test the rule using the Ruletest.

By default when you drag and drop an entity into the **Input** pane in the **Ruletest** editor, all the attributes in the entity are null. If you double-click a String attribute, the attribute value changes from null to an empty string (denoted by empty square brackets) even if you do not type anything.

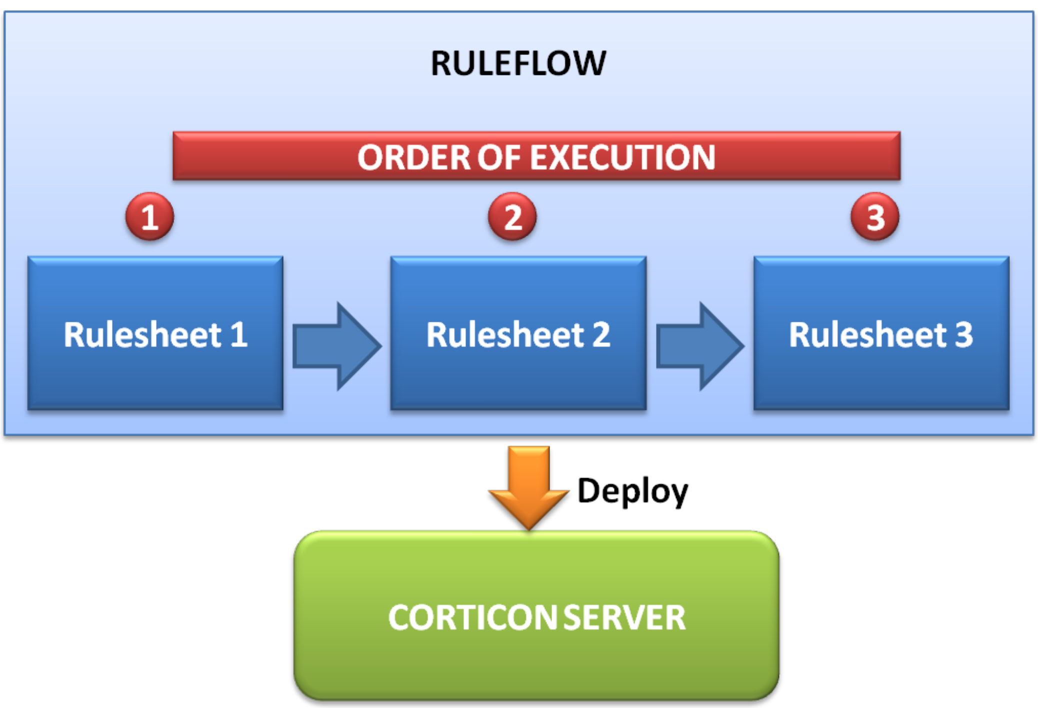
To ensure that the attribute value is null, right-click the attribute and select **Set to Null**:



6

What is a Ruleflow?

A Ruleflow enables you to connect two or more Rulesheets in a sequence.



When the Ruleflow is processed at runtime, the Rulesheets are executed one by one in that sequence. The output of one Rulesheet becomes the input of the next Rulesheet.

Note that you can use a Rulesheet in multiple Ruleflows as reusable modules of rule logic. Any change in the rule logic has to be made only once in the Rulesheet and it is propagated across all Ruleflows that refer to it.

A Ruleflow is deployed to Corticon Server as a Decision Service. You cannot deploy Rulesheets directly. If you need to deploy only one Rulesheet, you must add it to a Ruleflow and then deploy the Ruleflow.

For Corticon.js, a Ruleflow is packaged and deployed wherever the JavaScript application is running.

AI Tip **Multilingual Responses**
The Corticon AI Assistant can respond in your preferred language. Try prompts like: • “What is the purpose of this Ruleflow? Respond in French!” • “Quel est le but de ce flux de règles?”

For details, see the following topics:

- Create Ruleflows
- Test Ruleflows using Ruletests
- Use Rule Trace View to drill into a Ruletest
- Advanced Ruleflow tips and tricks

Create Ruleflows

Perform the following tasks to create Ruleflows:

- Create a Ruleflow file.
- Add Rulesheets to the Ruleflow using the Ruleflow editor.
- Connect Rulesheets in the Ruleflow editor.
- Set Ruleflow properties.

Create a Ruleflow file

You must create a Ruleflow file under a rule project and associate it with a Vocabulary. All Rulesheets to be added to a Ruleflow must be associated with the Ruleflow’s Vocabulary. A Ruleflow file has the extension `.erf`.

To create a Ruleflow file:

1. In Corticon Studio, click **File > New > Ruleflow**.
2. In the **Create New Rule Flow** wizard:
    a. Ensure that the **Enter or select parent folder** field contains the path of the location where you want the Ruleflow created.
    b. In the **File name** field, type a name for the **Ruleflow**.

c. Click **Next**.

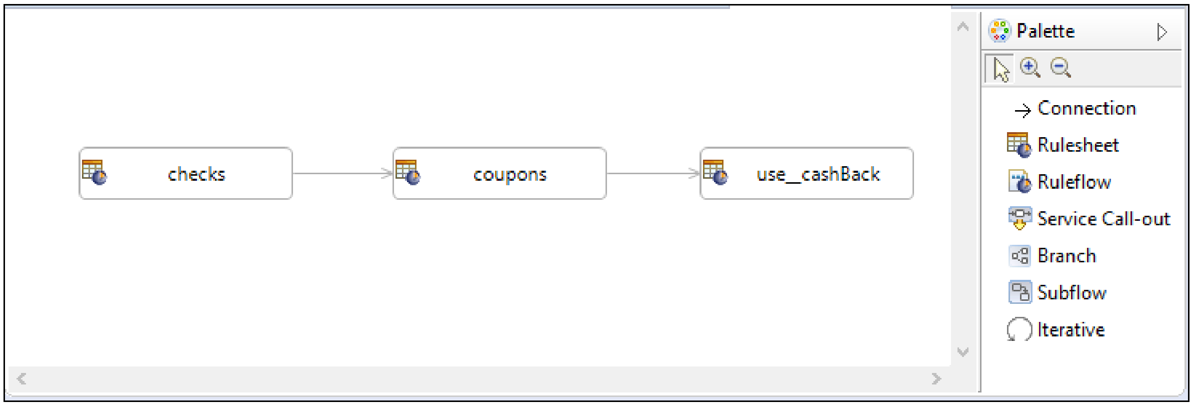
d. Ensure that the Vocabulary with which you want to associate the Ruleflow is selected, and click **Finish**.

The newly created Ruleflow displays in the **Project Explorer** view, and the **Ruleflow** editor.

Overview of the Ruleflow editor

The Ruleflow editor consists of two sections:

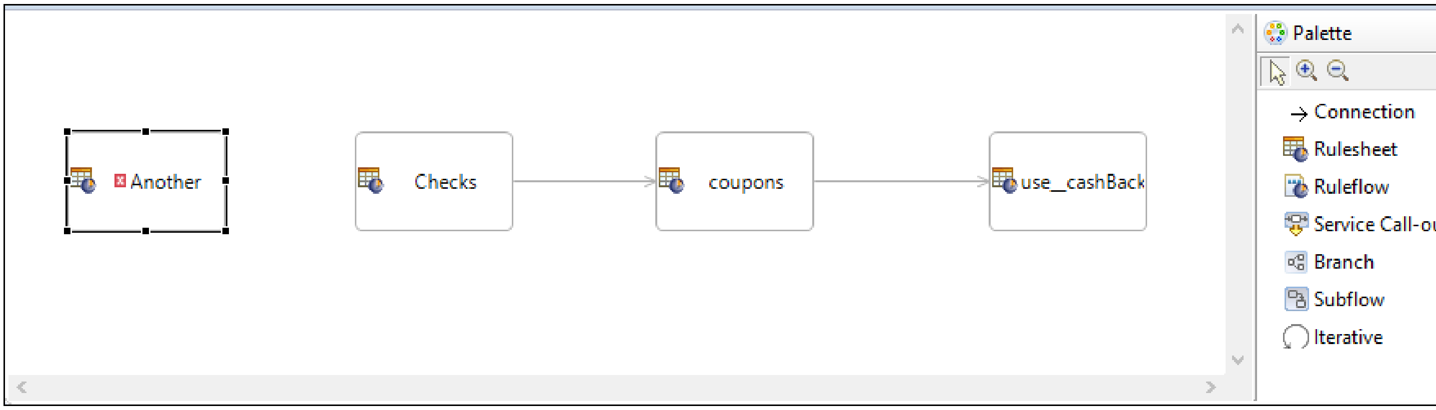
- **Modeling area**—Where you model a flowchart-like diagram of connected Rulesheets. The flowchart displays the sequence in which Rulesheets execute.
- **Palette**—Sets the model to perform actions.



Add Rulesheets to the Ruleflow

Before you start working with the **Ruleflow** editor, you must identify which Rulesheets you want to add, and the sequence in which you want the Rulesheets to execute at runtime.

To add Rulesheets to the Ruleflow, you drag each Rulesheet from the **Rule Project Explorer** view to the modeling area in the **Ruleflow** editor. After you add a Rulesheet to the **Ruleflow** editor, it is represented by a box, as shown for **Another**.



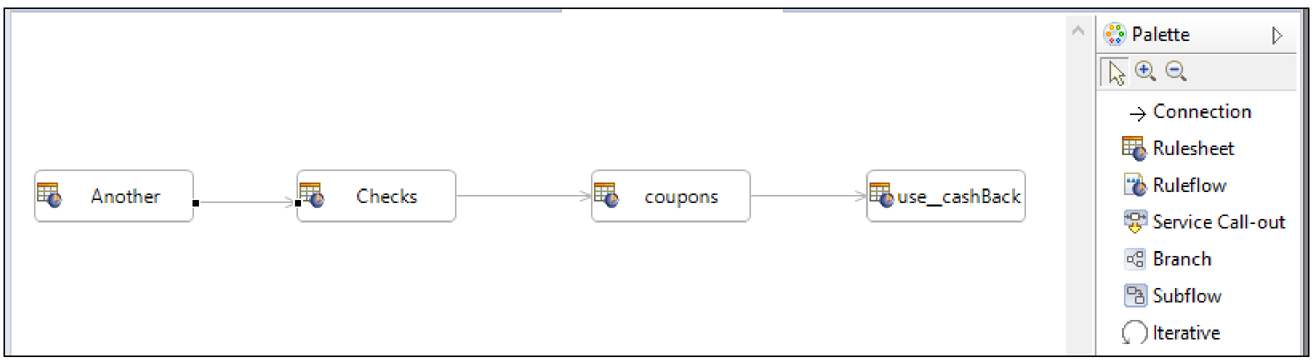
Since the Rulesheets are sequenced in a flowchart-like diagram, you can choose to arrange the Rulesheet boxes vertically, horizontally, or in a combination.. However,Progress recommends that you drag and position the first Rulesheet in the sequence first, the second next, and so on to make it easier to connect them.

Note: Each Rulesheet box bears the Rulesheet's filename, by default. However, you can change the name by triple-clicking the Rulesheet box and modifying it.

AI Tip **Multilingual Responses**
The Corticon AI Assistant can respond in your preferred language. Try prompts like: • "What is the purpose of this Ruleflow? Respond in French!" • "Quel est le but de ce flux de règles?"

Connect Rulesheets in the Ruleflow editor

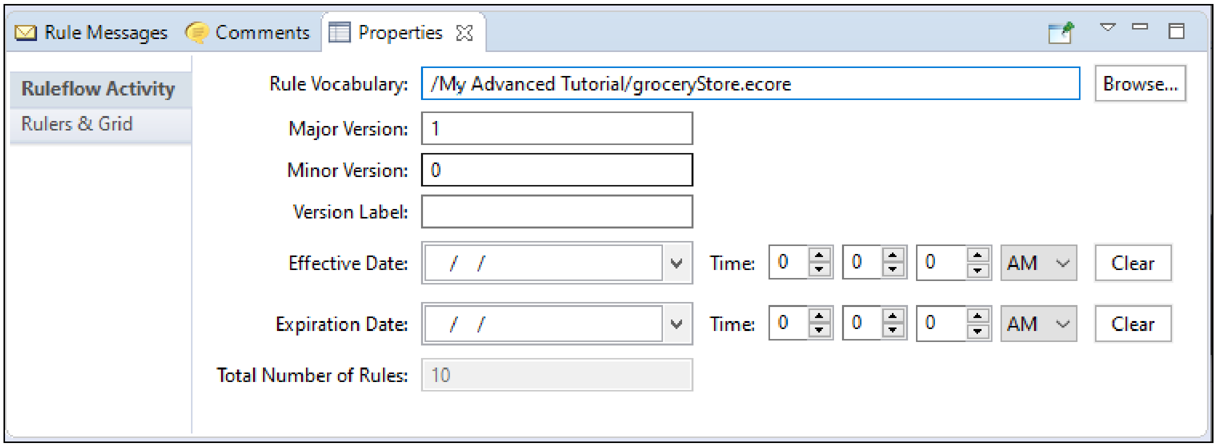
After you add all the Rulesheets that you require, you must connect them to define the order of execution.



You connect one Rulesheet to another by clicking **Connection** in the **Palette** and then dragging the connection arrow from one Rulesheet box to the next.

Set Ruleflow properties

When you open a Ruleflow file, the Properties view in Corticon is populated with different Ruleflow properties.



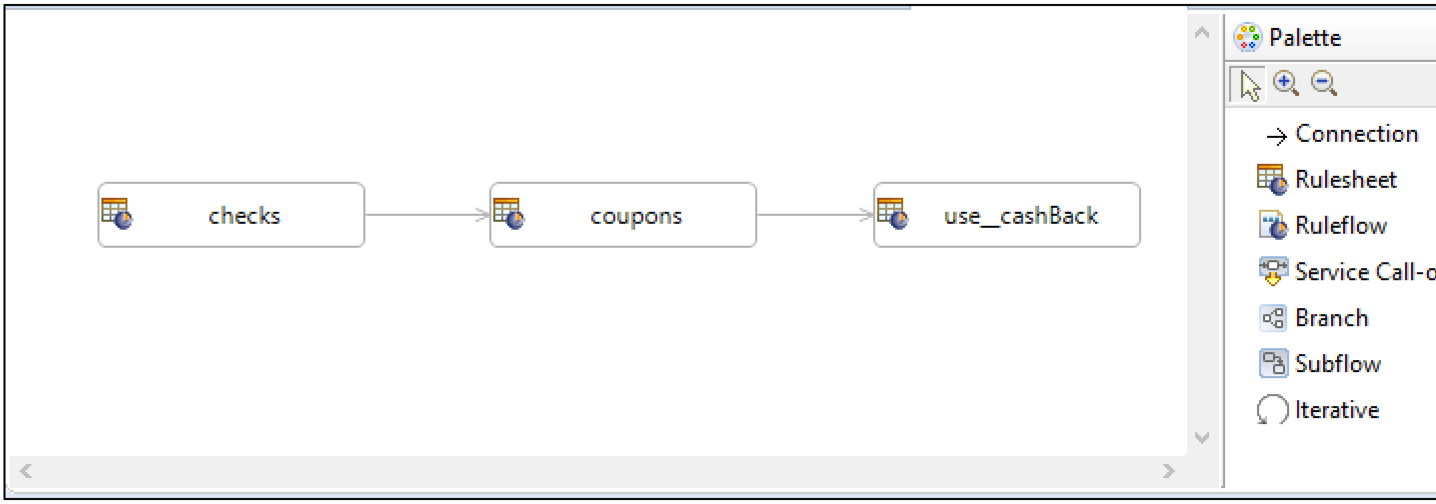
Ruleflow properties are categorized in two tabs:

Tab	Description
Ruleflow Activity	The properties in this tab are: • **Rule Vocabulary**—the Vocabulary file associated with the Ruleflow. • **Major Version, Minor Version, Version Label, Effective Date, Expiration Date**—enables you to maintain and manage multiple versions of the Ruleflow. • **Total Number of Rules**—the number of rules included in the Ruleflow. This is a non-editable field; it is the sum of the number of rule columns and action-only rules in each Rulesheet that is added to the Ruleflow.
Rulers & Grid	Provides various options to display rulers and a grid in the modeling area of the **Ruleflow** editor.

Test Ruleflows using Ruletests

You can test a Ruleflow using a Ruletest just as you use it to test a Rulesheet. When you test a Ruleflow in a Ruletest, the input data that you define is processed by the first Rulesheet in the Ruleflow's sequence. After the first Rulesheet completes executing, its output becomes the input to the next Rulesheet in the sequence, and so on, until the last Rulesheet completes executing.

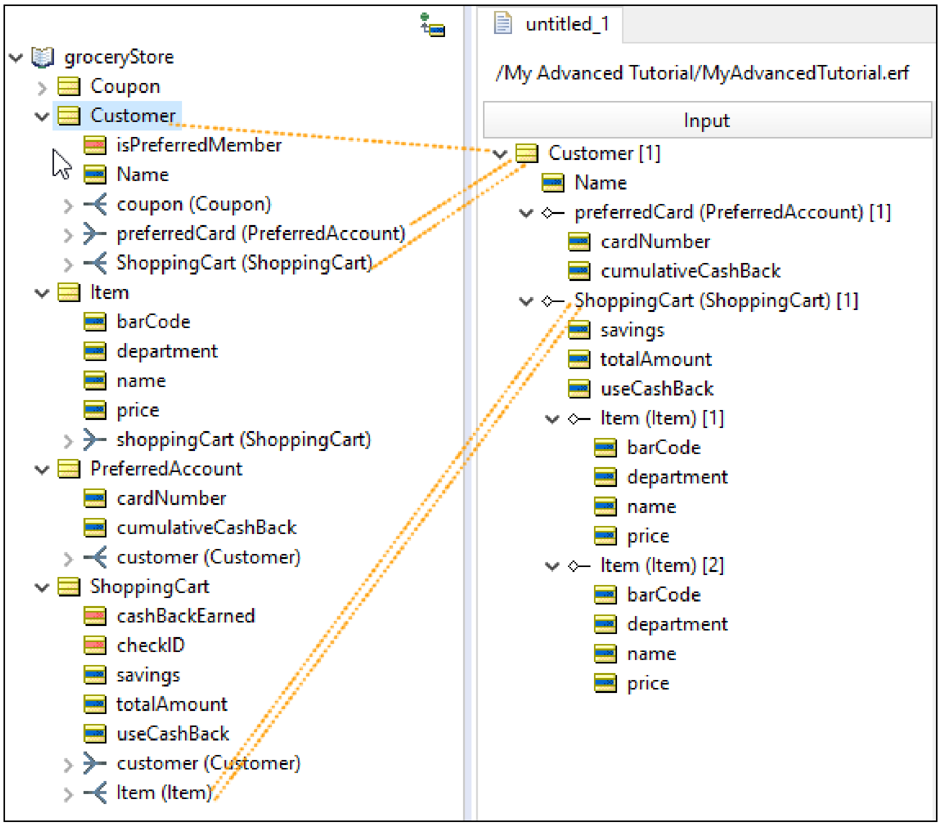
To test a Ruleflow using a Ruletest, you must select the Ruleflow as the Ruletest's test subject either while creating the Ruletest or by modifying the **Test Subject** property in the Ruletest editor.



You specify inputs for a Ruletest by:

1. In a Ruletest editor, drag entities from the **Rule Vocabulary** view, and dropping them into the **Input** pane. Each time you drag and drop an entity, you create an instance of the entity, adding all the attributes in the entity to the **Input** pane. You can delete attributes that you do not require.

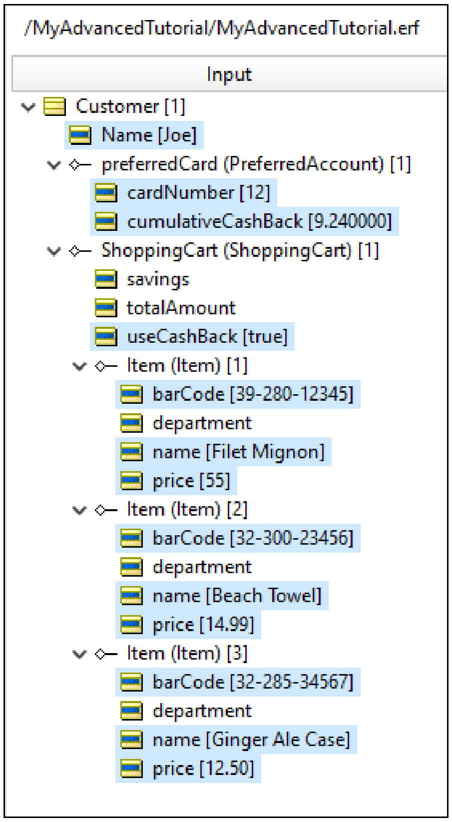
   In the following example, several entities form the request from associations. The customer is the primary entity. The customer's association with a preferred account and with a shopping cart are what you want, and then multiple associations between items that are in the customer's shopping cart:



Note: You can specify different entities as well as multiple instances of the same entity. When you specify multiple instances of the same entity, each entity is assigned an identity number. The identity number is 1 for the first instance of the entity, and is incremented by one for each additional instance that you drag and drop.

2. Specify values for the attributes by double-clicking name and entering its value. The syntax for specifying these values is similar to that of specifying attribute values in Rulesheet expressions, with one difference—you must not enter any values in quotes, even if the data type of the attribute is String, DateTime, Date or Time. If you enter quotes, the Ruletest treats it as part of the value.

   When you enter values for the Ruletest, it is ready to test:



Note: If you break any of these rules while specifying attribute values in the **Input** pane, Corticon Studio indicates a mismatch with a warning icon.

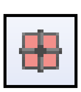
3. Select **Ruletest > Run All Tests**.

The **Output** pane then displays the final output of the Ruleflow. The **Rule Messages** view displays all rule messages that are created as a result of rules firing in each Rulesheet.

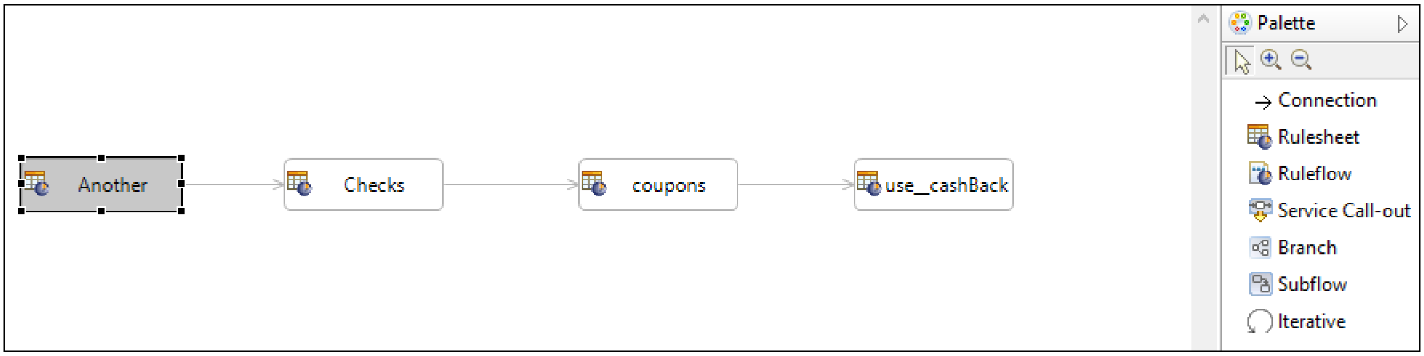
Enable and disable Rulesheets in a Ruleflow

As you test Ruleflows, you may find it necessary to omit specific Rulesheets from the Ruleflow to see how the Ruleflow performs in different use cases.

To omit a Rulesheet from the sequence of execution, select the Rulesheet box in the **Ruleflow** editor and click on the toolbar **Disable** button:



The disabled Rulesheet box turns grey, indicating that it has been disabled, as shown in this image.



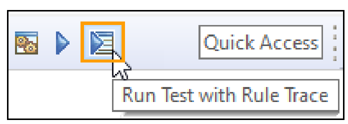
When you omit a Rulesheet from the sequence, the order of execution skips a step—the output from the previous Rulesheet becomes the input to the next Rulesheet.

To re-enable a disabled Ruleflow object, click on the object and click the Disable button.

Use Rule Trace View to drill into a Ruletest

You can get a dynamic view of all the actions and rules in a ruletest by running the test with rule trace. It reveals every rule that fired in a way that lets you sort, highlight, and locate every action and rule. You can even make changes to rulesheets and then the rerun the tests.

To run a Ruletest with Rule Trace View:

1. Open a Ruletest file in the Ruletest editor.
2. Click the toolbar button **Run Test with Rule Trace**: 

3. In the **Rule Trace** pane, examine the output, as shown here for the testsheet in *"Model the third Rulesheet" in the Advanced Tutorial Guide*.

Rule Statements | Rule Messages | Rule Trace

Sequence	Action	Element	Old Value	New Value	Assoc...	Location
1	Update Attribute	Item (Item) [3]/department		285		checks : A0
2	Update Attribute	Item (Item) [2]/department		300		checks : A0
3	Update Attribute	Item (Item) [1]/department		280		checks : A0
4	Update Attribute	ShoppingCart (ShoppingCart) [1]/totalAmount		82.490000		checks : D0
5	Update Attribute	Customer [1]/isPreferredMember		true		:
6	Update Attribute	ShoppingCart (ShoppingCart) [1]/cashBackEarned		1.649800		coupons : A0
7	Add Entity	Coupon [1]				coupons : 3
8	Update Attribute	Coupon [1]/expirationDate		05/07/22		coupons : 3
9	Update Attribute	Coupon [1]/description		10% off next gas ...		coupons : 3
10	Update Attribute	preferredCard (PreferredAccount) [1]/cumulativeCashBack	9.240000	10.889800		coupons : B0
11	Update Attribute	ShoppingCart (ShoppingCart) [1]/totalAmount	82.490000	71.600200		use_cashBack : 1
12	Update Attribute	ShoppingCart (ShoppingCart) [1]/savings		10.889800		use_cashBack : 1
13	Update Attribute	preferredCard (PreferredAccount) [1]/cumulativeCashBack	10.889800	0.000000		use_cashBack : 1

The results of a rule trace are dynamic:

- **Highlight**—Click anywhere on a line to highlight that element in the Testsheet output. Click on any item in the Ruletest to see all the rules related to that element highlighted in the Rule Trace Viewer.
- **Locate**—Double-click on any line to open the related Rulesheet positioned at the Action line and rule. The Rulesheet is in editable form so you can quickly make changes and rerun to see the effect of the change.
- **Sort**—Click any column header on the **Rule Trace** tab to sort the tab content in ascending order. Click again to sort into descending order.

This dynamic feature is best presented in this video:

https://youtu.be/H7S8U0gIl6k

For more information, see *"Trace rule execution" in the Corticon Rule Modeling Guide*.

Note: See how the rule trace view helped in a large project in the blog Fast Rules Diagnostics and Root Cause Analysis with the New Rule Trace Viewer.

Note:

- If the results are overwhelming, try changing the test subject to just one Rulesheet or disabling some rules.
- The Rule Trace Viewer is based on JSON. If you have the Studio property `com.corticon.tester.ccserver.execute.format` set to `XML` (instead of the default, `JSON`), the Rule Trace Viewer function is inoperative.

Advanced Ruleflow tips and tricks

Ruleflows provide a number of powerful features that are not in the scope of this course, including:

- **Nested Ruleflows**—Reduces the complexity of large Ruleflows by breaking them into smaller, child Ruleflows and adding them to the parent Ruleflow, referred to as a **Ruleflow in a Ruleflow**.
- **Conditional Branching**—Creates branches in a Ruleflow, where the value of an attribute determines which branch the data is routed to.
- **Subflows**—Configures an Iteration for a Subflow to enable looping behavior.
- **Versioning**—Assigns Major and Minor Version number so that the Corticon Server responds correctly to requests for the different versions.
- **Effective Dates**—Lets you name Ruleflows with identically named Ruleflows with slight variations so that they respond to requests only when they are in the specified date range.
- **Service Call-outs**—Accesses Datasources to enrich your rules, and update databases.

To learn more rule modeling techniques, see the *Advanced Rule Modeling Tutorial*.

AI Tip **Gain Insights into Ruleflows**
The AI Assistant can help you analyze how specific values are set or used in a ruleflow. For example, if a ruleflow calculates insurance rates. Try prompts like: • "How is the insurance rate calculated in this ruleflow?" • "List the age brackets used for rate calculations."