PROGRESS® Corticon®

Corticon Studio Tutorial: Basic Rule Modeling

PROGRESS

Notices

Copyright agreement

Table of Contents

Preface.....7

Progress Corticon documentation.....7

Overview of Progress Corticon.....9

Chapter 1: Introduction.....11

Chapter 2: Business process and rules.....13

Discovering the business problem.....13

The business process and rules.....14

Chapter 3: Setting up the Tutorial.....15

Tutorial set up with Corticon Studio for Analysts.....16

Tutorial set up with full Corticon Studio.....19

Chapter 4: Getting started with the Corticon Basic Tutorial.....23

Opening the Tutorial's Vocabulary.....24

Creating a Rulesheet.....26

About Corticon Rulesheets.....28

Explore the Rules Operators.....29

Chapter 5: Steps that define Rules in the Model.....31

Chapter 6: Conflicts and completeness.....35

Check for conflicts.....35

- Identify conflicts.....36
- Expand the rules.....36
- Resolve conflict errors.....37

Check for completeness.....38

- Resolve completeness errors – step 1.....39
- Resolve completeness errors – step 2.....40

Check for logical loops.....41

Save the Rulesheet.....42

Chapter 7: Testing rules.....43

Defining a test case....43
Creating a Ruletest....44
Setting the Test Subject....46
Set up the test scenario – step 1....47
Set up the test scenario – step 2....48
Set up the test scenario – step 3....50
Execute the Ruletest....50
Verify the test results....51
Changing the test....52
Saving the Ruletest....52

Chapter 8: Modifying the Corticon resources....55

Discover new business rules....56
Add new rule statements....56
Open and edit the Vocabulary....57
Add an attribute – step 1....58
Add an attribute – step 2....59
Modify a custom data type....60
Save the Vocabulary changes....60
Model the new rule....61
Check new rules for conflicts....62
Setting overrides....62
Checking a new rule for completeness....63
Modify the Ruletest....64
Summary....65

Appendix A: Third party acknowledgments 67

Preface

For details, see the following topics:

- Progress Corticon documentation
- Overview of Progress Corticon

Progress Corticon documentation

The following documentation, as well as a *What's New in Corticon* document, is included with this Progress Corticon release:

Corticon Tutorials	
Corticon Studio Tutorial: Basic Rule Modeling	Introduces modeling, analyzing, and testing rules and decisions in Corticon Studio. Recommended for evaluators and users getting started. *See also the PowerPoint-as-PDF version of this document that is accessed from the Studio for Analysts' **Help** menu.*
Corticon Studio Tutorial: Advanced Rule Modeling	Provides a deeper look into Corticon Studio's capabilities by defining and testing vocabularies, scope, collections, messages, filters, conditions, transient data, and calculations in multiple rulesheets that are assembled into a Ruleflow. *See also the PowerPoint-as-PDF version of this document that is accessed from the Studio for Analysts' **Help** menu.*
Corticon Tutorial: Using Enterprise Data Connector (EDC)	Introduces Corticon's direct database access with a detailed walkthrough from development in Studio to deployment on Server. Uses Microsoft SQL Server to demonstrate database read-only and read-update functions.

Corticon Studio Documentation: Defining and Modeling Business Rules	
Corticon Studio: Installation Guide	Step-by-step procedures for installing Corticon Studio and Corticon Studio for Analysts on computers running Microsoft Windows. Also shows how use other supported Eclipse installations for integrated development. Shows how to enable internationalization on Windows.
Corticon Studio: Rule Modeling Guide	Presents the concepts and purposes the Corticon Vocabulary, then shows how to work with it in Rulesheets by using scope, filters, conditions, collections, and calculations. Discusses chaining, looping, dependencies, filters and preconditions in rules. Presents the Enterprise Data Connector from a rules viewpoint, and then shows how database queries work. Provides information on versioning, natural language, reporting, and localizing. Provides troubleshooting and many *Test Yourself* exercises.
Corticon Studio: Quick Reference Guide	Reference guide to the Corticon Studio user interface and its mechanics, including descriptions of all menu options, buttons, and actions.
Corticon Studio: Rule Language Guide	Reference information for all operators available in the Corticon Studio Vocabulary. A Rulesheet example is provided for many of the operators. Includes special syntax issues, handling arithmetic and character precedence issues.
Corticon Studio: Extensions Guide	Detailed technical information about the Corticon extension framework for extended operators and service call-outs. Describes several types of operator extensions, and how to create a custom extension plug-in.
Corticon Server Documentation: Deploying Rules as Decision Services	
Corticon Server: Deploying Web Services with Java	Details installing the Corticon Server as a Web Services Server, and then deploying and exposing Decision Services as Web Services on Tomcat and other Java-based servers. Presents the features and functions of the browser-based Server Console.
Corticon Server: Deploying Web Services with .NET	Details installing the Corticon Server as a Web Services Server, and then deploying and exposing decisions as Web Services with .NET. Provides installation and configuration information for the .NET Framework and Internet Information Services (IIS) on various supported Windows platforms.
Corticon Server: Integration & Deployment Guide	An in-depth, technical description of Corticon Server deployment methods, including preparation and deployment of Decision Services and Service Contracts through the Deployment Console tool. Discusses relational database concepts and implementation of the Enterprise Data Connector. Goes deep into the server to discuss state, persistence, and invocations by version or effective date. Includes samples, server monitoring techniques, and recommendations for performance tuning.

Overview of Progress Corticon

Progress® Corticon® is the Business Rules Management System with the patented "no-coding" rules engine that automates sophisticated decision processes.

Progress Corticon products

Progress Corticon distinguishes its development toolsets from its server deployment environments.

- **Corticon Studios** are the Windows-based development environment for creating and testing business rules:
  - **Corticon Studio for Analysts.** is a standalone application, a lightweight installation that focuses exclusively on Corticon.
  - **Corticon Studio** is the *Corticon Designer* perspective in the **Progress Developer Studio** (PDS), an industry-standard Eclipse and Java development environment. The PDS enables development of applications integrated with other products, such as Progress OpenEdge.

  The functionality of the two Studios is virtually identical, and the documentation is appropriate to either product. Documentation of features that are only in the *Corticon Designer* (such as on integrated application development and Java compilation) will note that requirement. Refer to the *Corticon Studio: Installation Guide* to access, prepare, and install each of the Corticon Studio packages.

  Studio Licensing - Corticon embeds a time-delimited evaluation license that enables development of both rule modeling and Enterprise Data Connector (EDC) projects, as well as testing of the projects in an embedded Axis test server. You must obtain studio development licenses from your Progress representative.

- **Corticon Servers** implement web services for business rules defined in Corticon Studios:
  - **Corticon Server for deploying web services with Java** is supported on various application servers, and client web browsers. After installation on a supported Windows platform, that server installation's deployment artifacts can be redeployed on various UNIX and Linux web service platforms as Corticon Decision Services. The guide *Corticon Server: Deploying web services with Java* provides details on the full set of platforms and web service software that it supports, as well as installation instructions in a tutorial format for typical usage.
  - **Corticon Server for deploying web services with .NET** facilitates deployment of Corticon Decision Services on Windows .NET Framework and Microsoft Internet Information Services (IIS). The guide *Corticon Server: Deploying web services with .NET* provides details on the platforms and web service software that it supports, as well as installation instructions in a tutorial format for typical usage.

  Server Licensing - Corticon embeds a time-delimited evaluation license that enables evaluation and testing of rule modeling projects on supported platform configurations. You must obtain server deployment licenses and server licenses that enable the Enterprise Data Connector (EDC) from your Progress representative.

1

Introduction

This *Basic Rule Modeling Tutorial* provides an introduction to the Progress Corticon Studio.

You will learn how to capture rules from business specifications, model the rules, analyze them for logical errors, and test the execution of your rules; all without programming.

Your goal is to create a Decision Service: A group of rules that captures the logic of a single decision-making step completely and unambiguously. In one sense, a Decision Service (when managed with Progress Corticon Studio) is a business-friendly model of your rules (i.e., your decision-making logic). In another sense, a Decision Service is a powerful asset, allowing you to automatically process the rules as a part of business transactions.

The Decision Services that you build using Corticon Studio may be deployed as executable, standards-based services available to other software applications via Java, .NET, or SOAP messaging. Corticon Decision Services are in use today across the globe, automating many high-volume decision-intensive processes.

After you complete this tutorial, see the other Corticon Tutorials:

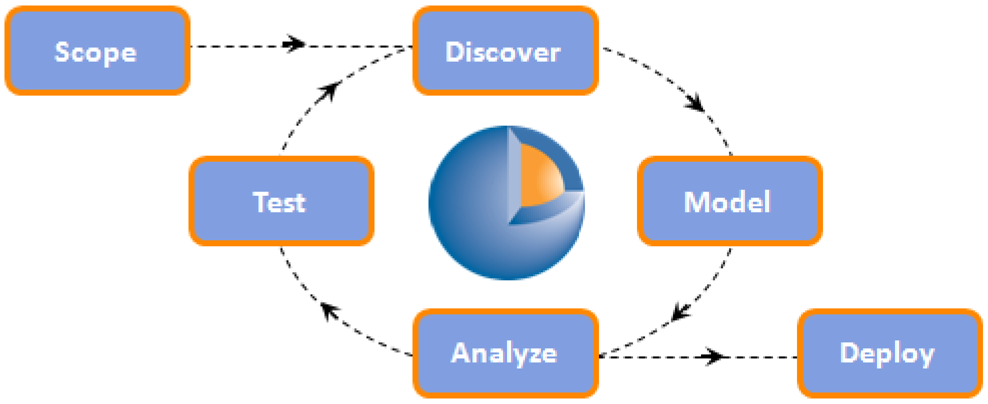
- *Corticon Studio Tutorial: Advanced Rule Modeling*
- *Corticon Tutorial: Using Enterprise Data Connector (EDC)*
- *Corticon Server: Deploying Web Services with Java*
- *Corticon Server: Deploying Web Services with .NET*

Note: A Corticon installation contains a time-limited evaluation license. Once your evaluation period ends, you need to get a new license from Progress Software. See "Registering your Corticon license in Studio" in the *Corticon Studio: Installation Guide* for instructions.

This Tutorial is designed for hands-on use. We recommend that you type along with the instructions and illustrations presented.

The Corticon Decision Services methodology

Corticon provides a simple yet powerful methodology for modeling business rules called the "Corticon Decision Services Methodology". This methodology follows the lifecycle of your decision service, and involves the following steps:



1. Scope the business problem. Ask the question: *What decisions are we trying to make?*
2. Discover the business rules. Ask the question: *How do we make these decisions?*
3. Model the business rules. Using Progress Corticon, logically express the rules.
4. Analyze the business rules. Using Progress Corticon, ensure that the rules are complete, consistent and free of logical errors.
5. Test the business rules. Using Progress Corticon, test rule execution to ensure expected business results.
6. Deploy the business rules. Using Progress Corticon Server, automate the rules across enterprise applications.

Steps 2-6 are repeated as rules change. This tutorial includes two iterations through the lifecycle, including an initial implementation and subsequent change cycle. Deployment is covered in a separate tutorial (see *Server Deployment Tutorial* and *Server Integration & Deployment Guide*).

2

Business process and rules

This chapter describes the business process scenario and rules.

For details, see the following topics:

- Discovering the business problem
- The business process and rules

Discovering the business problem

The example used in this Tutorial is derived from a business problem in which an air cargo company loads cargo of various sizes and weights into containers, then onto its fleet of aircraft prior to shipment.

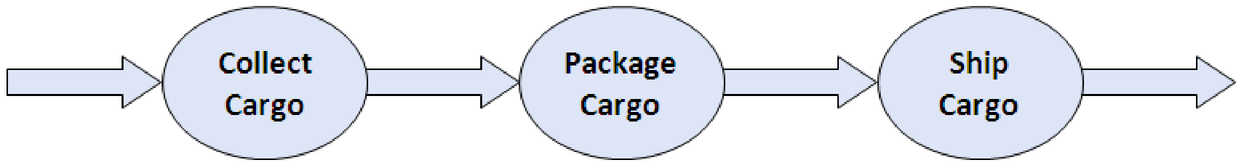
To operate safely, the company must ensure that an aircraft is never loaded with cargo that exceeds an aircraft's capabilities. Flight plans are created by the company that assign cargo shipments to containers, then containers to aircraft. Part of the flight plan creation process involves verifying that no plan violates any safety or operational rule.

The air cargo company wants to improve the quality and efficiency of the flight planning process by modeling and automating business rules using Corticon's Business Rule Management System.

The business process and rules

Complex problems such as flight planning are better described in their component parts. The best way to do this is by describing the business process for this problem. From a process diagram, we can easily identify the decision-making activities, which in turn are described by business rules.

First, define your business process as a sequence of activities or steps:



Next, determine which process steps involve decisions. Any step involving a decision is a candidate for automation using Corticon.

In this process, all three steps involve decisions, in addition to physical labor. The **scope** of this tutorial is the "Package Cargo" step, which involves the decision about what container to use for various cargo, based upon such criteria as the cargo's weight, volume and contents.

Today, the packaging decision is made by our shipping personnel based upon their experience. The problem is that some people make better decisions than others, which leads to inconsistent practices. We want to use Corticon to standardize and automate the packaging decision.

Now that we have scoped our problem, we need to **discover** our business rules.

To discover our business rules, we just ask: "How do we make this decision?"

For this case, we ask "How do we package cargo?" We ask this question to the people who perform and manage this step in the process.

They often provide the answer in the form of a policy or procedure manual, or simply as a set of rules that they follow. Sometimes the rules are embedded in the code of our legacy systems. In other cases, the rules are not documented and found only in the heads of our people.

In all cases, we will capture the discovered rules directly into Corticon Studio.

Package Cargo

- Cargo weighing <= 20,000 kilos must be packaged in a standard container.
- Cargo with volume > 30 cubic meters must be packaged in an oversize container.

3

Setting up the Tutorial

Note: See the *Corticon Studio: Installation Guide* for instructions on downloading and installing Corticon Studios.

Whether you install the Corticon Studio for Analysts or the full Corticon Studio, the tutorial is fundamentally the same. However, the nature of the two Studios makes the setup of the tutorial a bit different.

Choose the brief procedure appropriate to your Studio type. Once you have it setup, the look and feel of the demo will be consistent for both Studios.

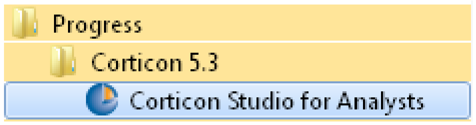
Note: About the Corticon license -- Corticon embeds a timed evaluation license in each Corticon Studio that lets you evaluate Corticon Studio features. Typically, you do not need to do anything to get started, But, when you start Studio, a **License Warning** alert means that the license file is invalid, corrupted, or expired. Then, when you create or modify any Corticon files, you get an **Asset Locked Warning**, indicating that you can just review existing files. Contact your Progress Corticon representative to obtain a workable license. Place the license file on your Studio machine, then launch Studio. Choose **Window > Preferences**, then expand **Progress Corticon > Rule Modeling**. Click **Browse** and then navigate to choose your new, valid, unexpired license. When you click **OK**, and restart Studio the license update process is complete.

For details, see the following topics:

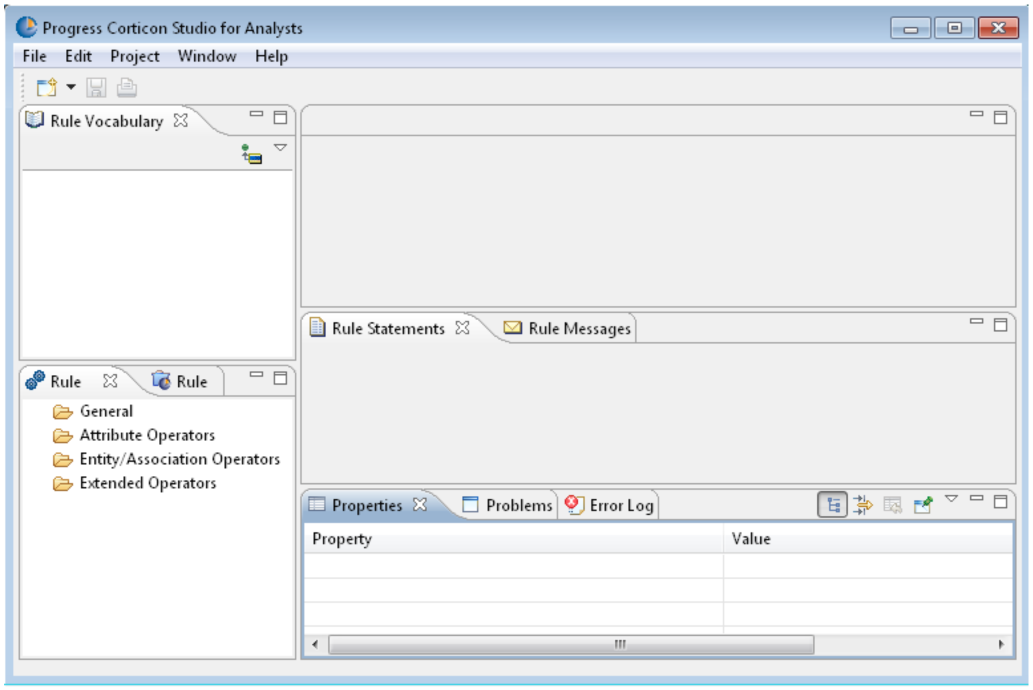
- Tutorial set up with Corticon Studio for Analysts
- Tutorial set up with full Corticon Studio

Tutorial set up with Corticon Studio for Analysts

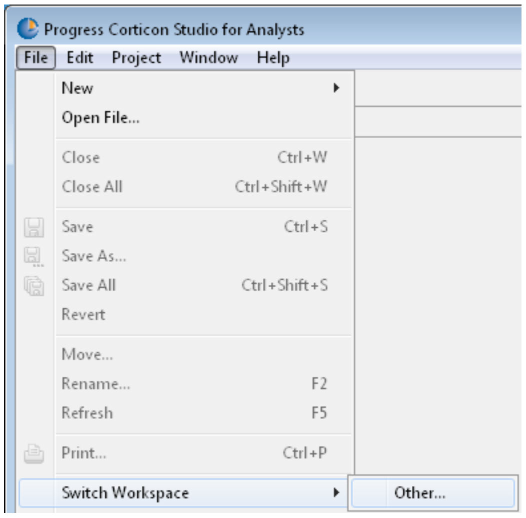
1. After installing Corticon Studio for Analysts successfully, start the Studio by choosing the **Start** menu path **All Programs > Progress > Corticon 5.3 > Corticon Studio for Analysts**, as shown:



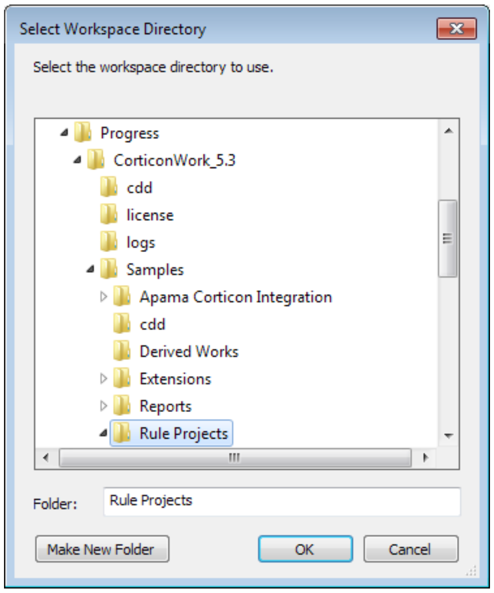
2. The initial workspace is empty, showing only the various views in Studio, as shown:



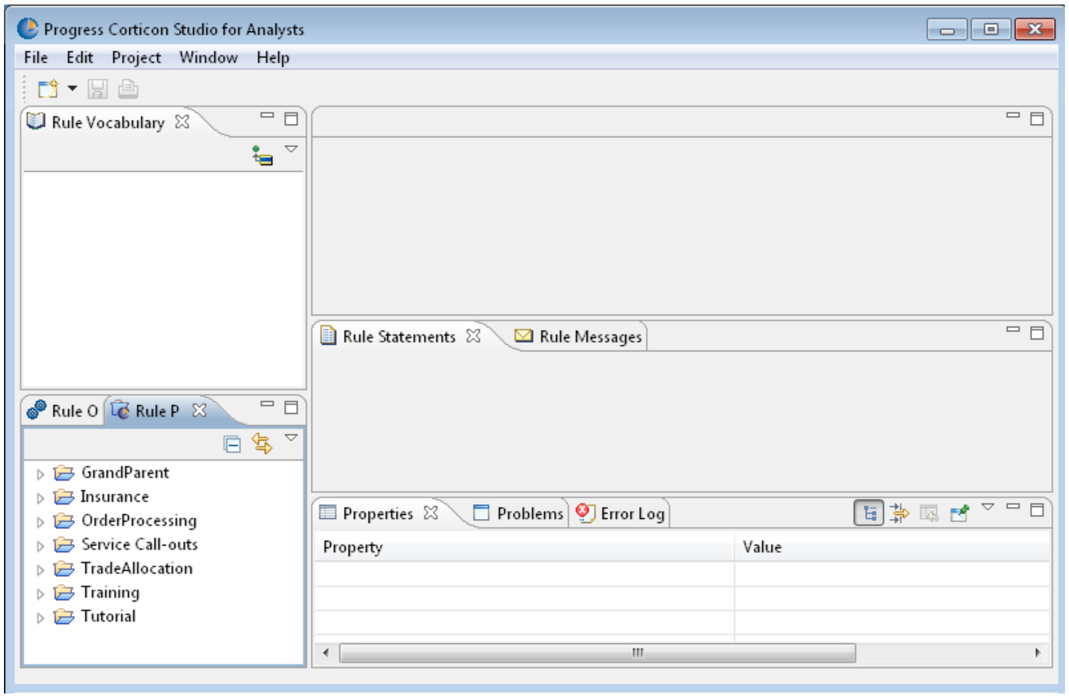
3. Switch to the workspace where this Tutorial's sample files are located by choosing **File > Switch Workspace > Other**, as shown:



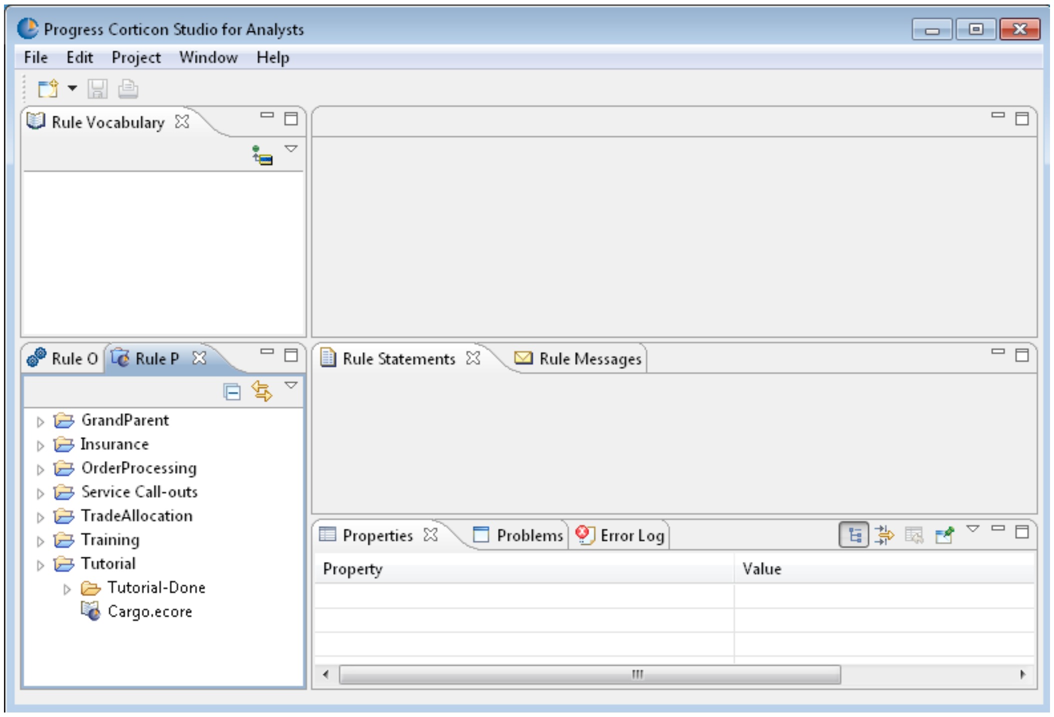
4. In the **Workspace Launcher** dialog box, click **Browse**. Navigate to your Corticon installation directory's `\Samples\Rule Projects`, and then click on `Rule Projects` to select it, as shown:



5. Click **OK**. Corticon Studio closes and restarts, with the new workspace open in the **Rule Project Explorer** view in the lower left, as shown:



6. Click on the arrow to the left of **Tutorial** to expand its list, as shown:



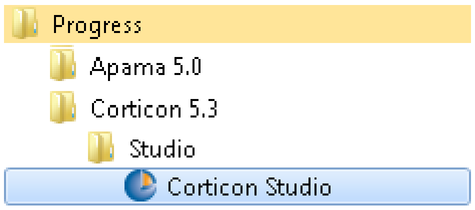
The Basic Rule Modeling Tutorial is now set up in your Corticon Studio for Analysts.

You can move ahead to the beginning of the tutorial at Getting started with the Corticon Basic Tutorial on page 23.

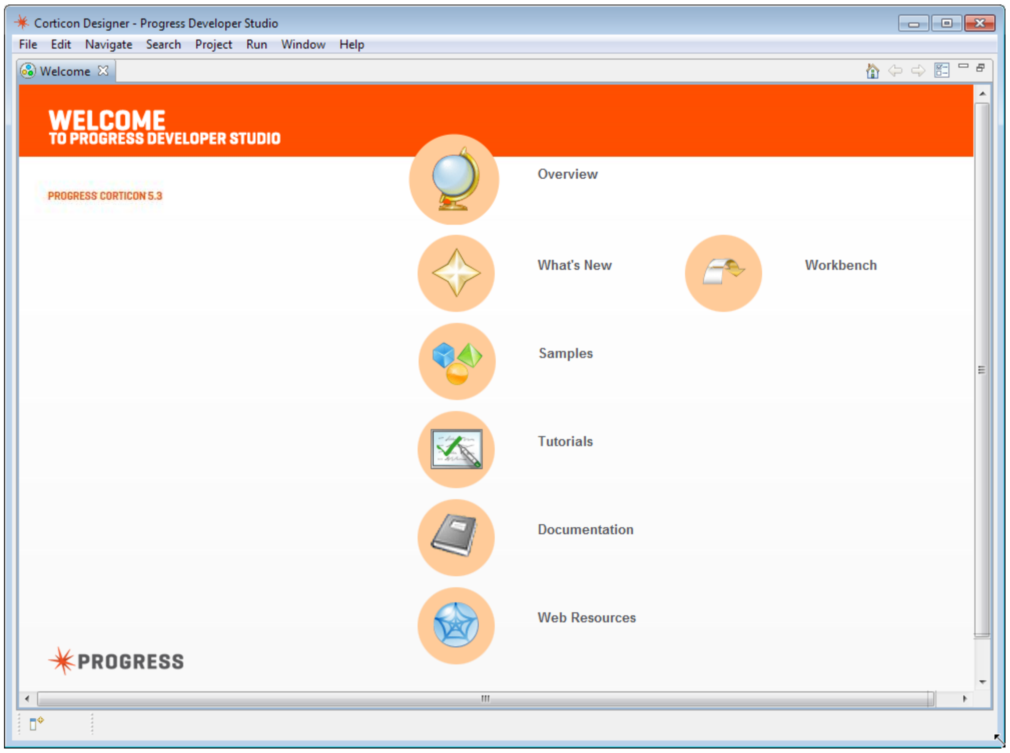
Note: You might want to open or print this guide as well as other books in the Corticon documentation set. Choose **Help > Contents** on the Studio menubar to open the Corticon help PDF portfolio. (Requires a PDF reader product.)

Tutorial set up with full Corticon Studio

1. After successfully installing the full Corticon Studio (the *Corticon Designer* perspective in the Progress Developer Studio), start the Studio by choosing the **Start** menu path **All Programs > Progress > Corticon 5.3 > Studio > Corticon Studio**, as shown:

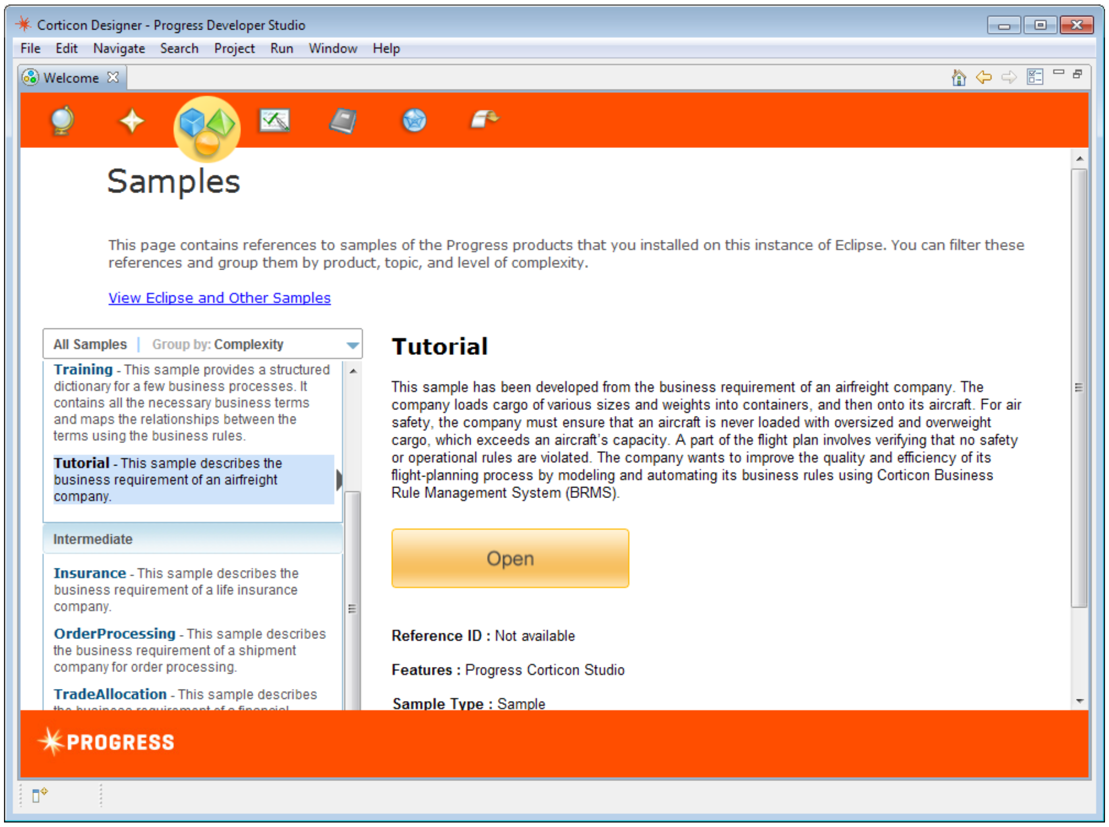


2. The Progress Developer Studio opens for Corticon 5.3 to its Welcome page, as shown:


Corticon Designer - Progress Developer Studio
File Edit Navigate Search Project Run Window Help
Welcome
WELCOME
TO PROGRESS DEVELOPER STUDIO
PROGRESS CORTICON 5.3
Overview
What's New
Workbench
Samples
Tutorials
Documentation
Web Resources
PROGRESS

Note: If the page does not display as shown, you might need to re-open it by choosing **Help > Welcome**, and, if necessary, clicking the **Restore Welcome** link.

3. Click **Samples** to open the Samples page, and then scroll through the list of samples to select **Tutorial**, as shown:


Corticon Designer - Progress Developer Studio
File Edit Navigate Search Project Run Window Help
Welcome
All Samples | Group by: Complexity
Intermediate
Open
PROGRESS

Samples

This page contains references to samples of the Progress products that you installed on this instance of Eclipse. You can filter these references and group them by product, topic, and level of complexity.

View Eclipse and Other Samples

Training - This sample provides a structured dictionary for a few business processes. It contains all the necessary business terms and maps the relationships between the terms using the business rules.

Tutorial - This sample describes the business requirement of an airfreight company.

Insurance - This sample describes the business requirement of a life insurance company.

OrderProcessing - This sample describes the business requirement of a shipment company for order processing.

TradeAllocation - This sample describes

Tutorial

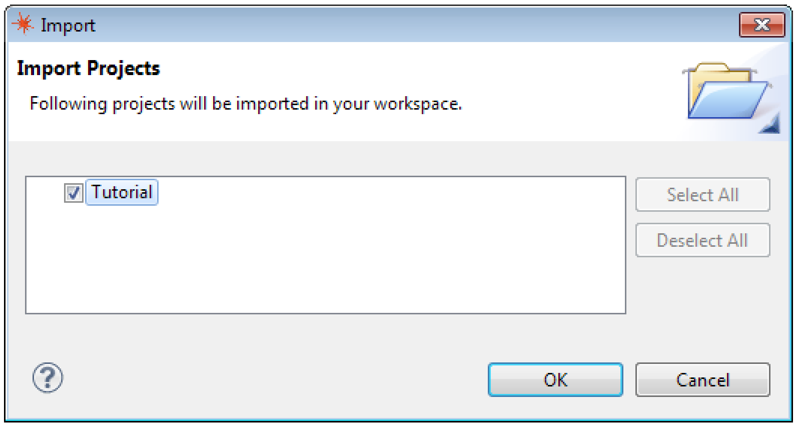
This sample has been developed from the business requirement of an airfreight company. The company loads cargo of various sizes and weights into containers, and then onto its aircraft. For air safety, the company must ensure that an aircraft is never loaded with oversized and overweight cargo, which exceeds an aircraft's capacity. A part of the flight plan involves verifying that no safety or operational rules are violated. The company wants to improve the quality and efficiency of its flight-planning process by modeling and automating its business rules using Corticon Business Rule Management System (BRMS).

Reference ID : Not available

Features : Progress Corticon Studio

Sample Type : Sample

4. Click **Open**. The **Import** dialog box opens. Select the **Tutorial**, as shown:

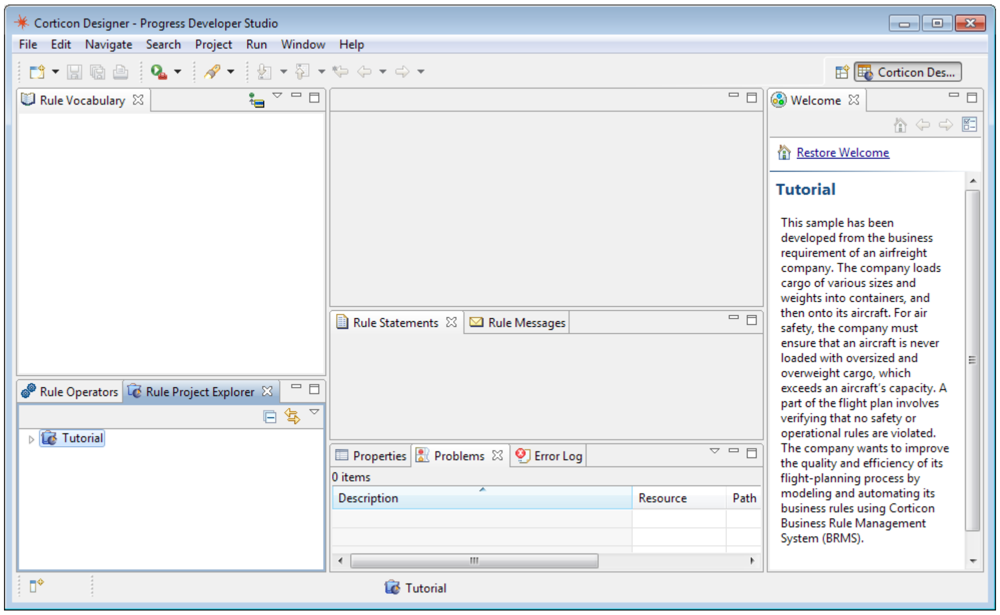


Import Projects

Following projects will be imported in your workspace.

☑ Tutorial

5. Click **OK** to load the project into the workspace while displaying the Welcome info in the right panel, as shown:



Tutorial

This sample has been developed from the business requirement of an airfreight company. The company loads cargo of various sizes and weights into containers, and then onto its aircraft. For air safety, the company must ensure that an aircraft is never loaded with oversized and overweight cargo, which exceeds an aircraft's capacity. A part of the flight plan involves verifying that no safety or operational rules are violated. The company wants to improve the quality and efficiency of its flight-planning process by modeling and automating its business rules using Corticon Business Rule Management System (BRMS).

The Basic Rule Modeling Tutorial is now set up in your Corticon Studio.

You can move ahead to the beginning of the tutorial at Getting started with the Corticon Basic Tutorial on page 23, just ahead.

Note: You might want to open or print this guide as well as other sections of the Corticon documentation set as online help. Choose **Help > Help Contents** on the Studio menubar to open the Corticon online help.

4

Getting started with the Corticon Basic Tutorial

This chapter shows how to start using the tutorial project to jumpstart your learning experience.

Note: The basic tutorial uses a pre-defined Vocabulary which you review and then apply to making business rules. In the advanced tutorial, you start with a clean project to create the Vocabulary and all the rules.

Remember: The folder **Tutorial-Done** provides the results you will get from doing all the tasks described in this tutorial. Copy from that folder if you must, yet keep it pristine - it can help clarify things that you can't seem to get right, as well as provide you a fast-path to the answers.

For details, see the following topics:

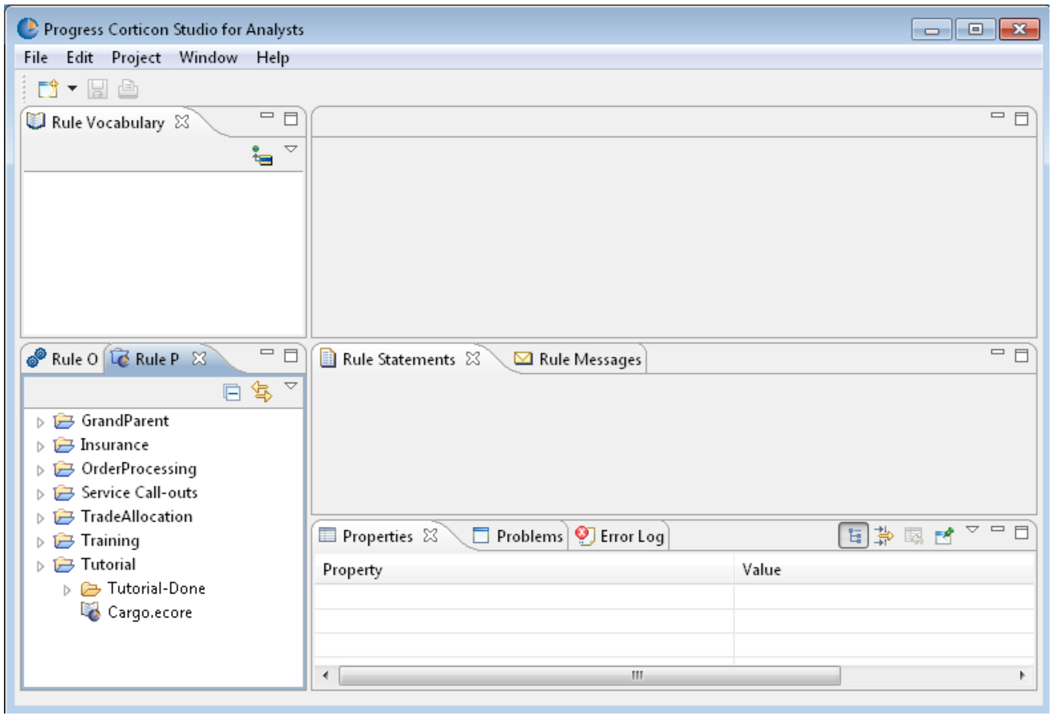
- Opening the Tutorial's Vocabulary
- Creating a Rulesheet
- About Corticon Rulesheets
- Explore the Rules Operators

Opening the Tutorial's Vocabulary

When you describe business process rules, you need a Vocabulary. The Vocabulary uses the terms referenced by our business rules (such as ‘aircraft’ and ‘flight plan’) to express each item's characteristics and its relationship with different items.

We'll create the Vocabulary for the advanced tutorial but for this tutorial we prepared a Vocabulary in advance.

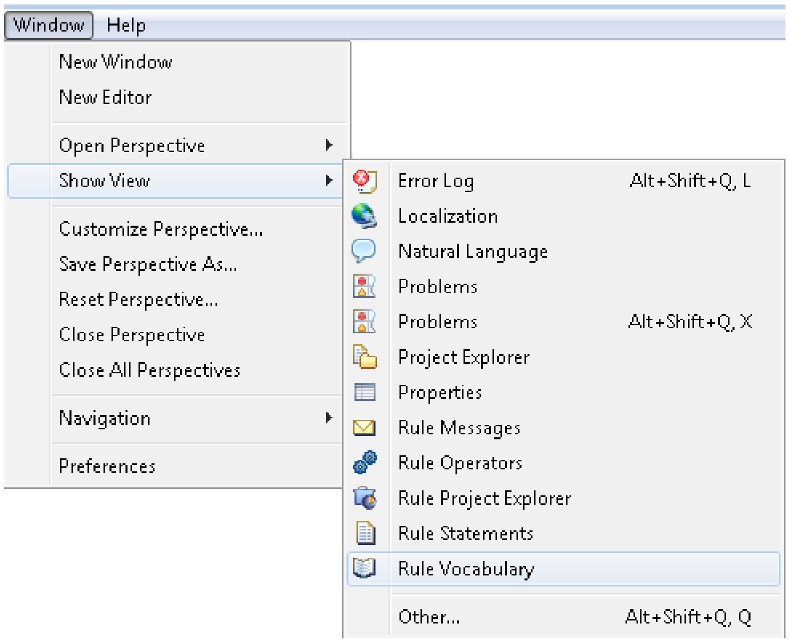
The setup tasks resulted in a page similar to this in the Corticon Studio for Analysts:



Navigating the Studio page

The tabs and views on a Studio page are dynamic. You might want to open some, rearrange some, and close others. Let's navigate around a bit to see how that works:

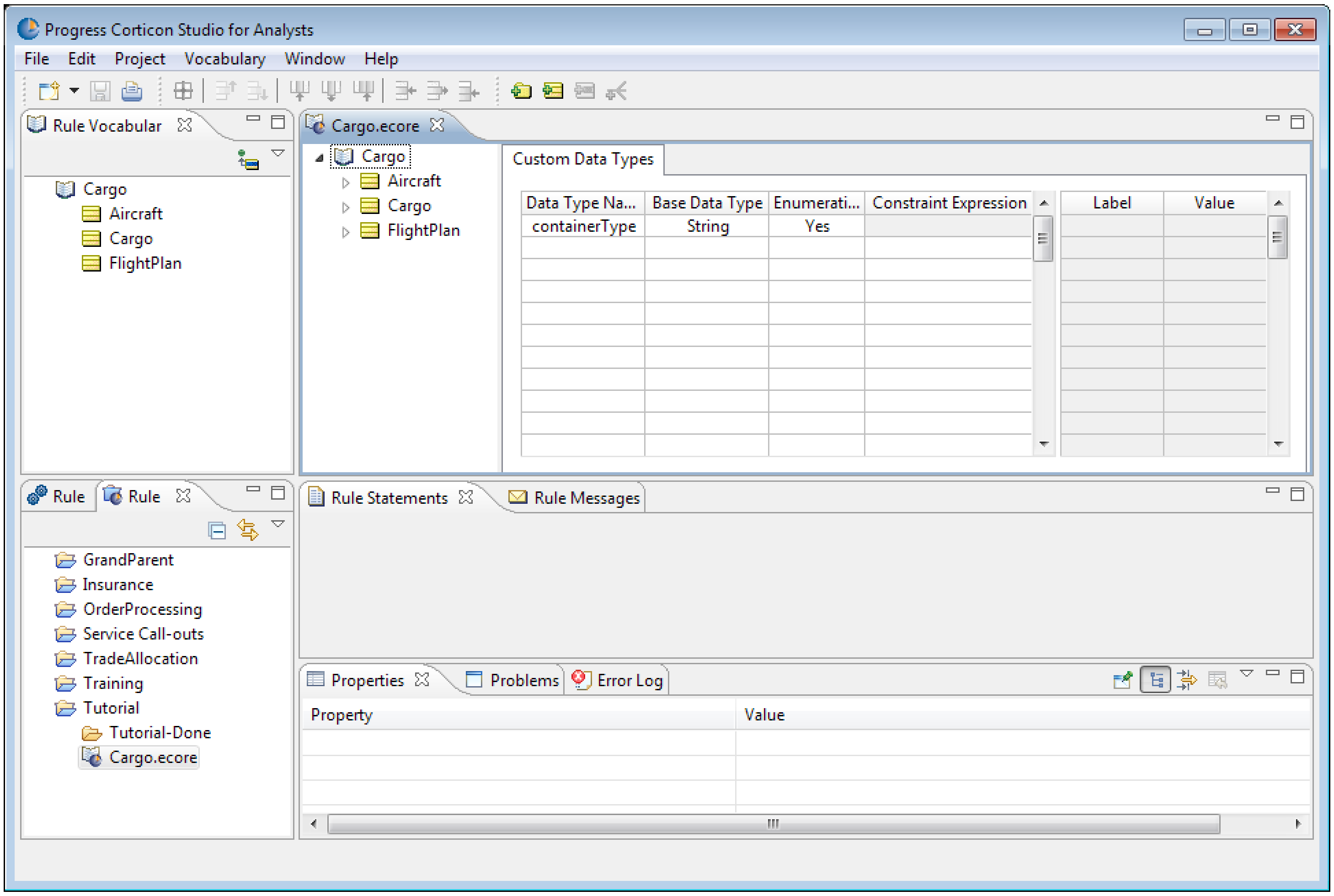
- If the **Rule Project Explorer** tab is not active, click on its tab to bring it to the front. This view is provided in the Studio for Analysts while the full Studio's *Corticon Designer* perspective provides the same functionality in the standard Eclipse IDE **Project Explorer** view. As you proceed through the Studio documents, you might find both these terms used to convey the same view functions.
- If the window seems crowded, click on a section's **Minimize** to pin the minimized window on the window's margin. Try that if the **Welcome** view is open.
- If a tab is in one section of the screen and you would rather it were in another, drag the tab to that section.
- If you do not see a view - such as **Rule Vocabulary** - choose the view from the **Window > Show View** menu, as shown:



To open the tutorial's Vocabulary:

* Double-click on the file `Cargo.ecore` in the **Rules Project** view.

It opens in the **Rule Vocabulary** view, and it also opens as a tab in the **Vocabulary** editor editing area, as shown:



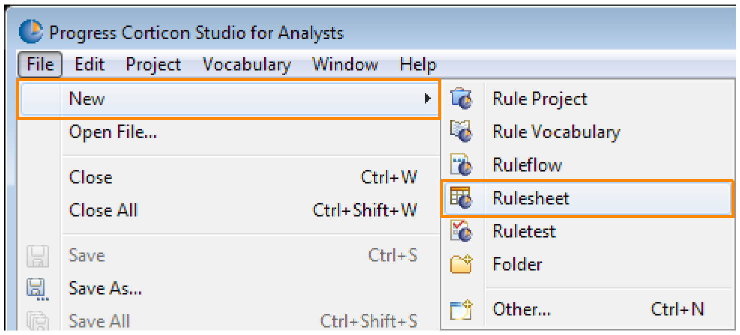
Data Type Na...	Base Data Type	Enumerati...	Constraint Expression
containerType	String	Yes	

Let's start building a rule model by using this Vocabulary to create our first Rulesheet.

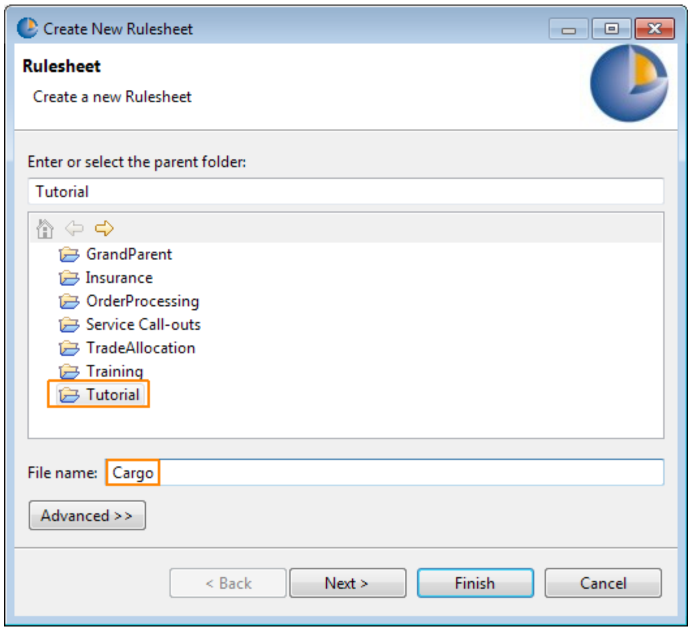
Creating a Rulesheet

To create a rulesheet:

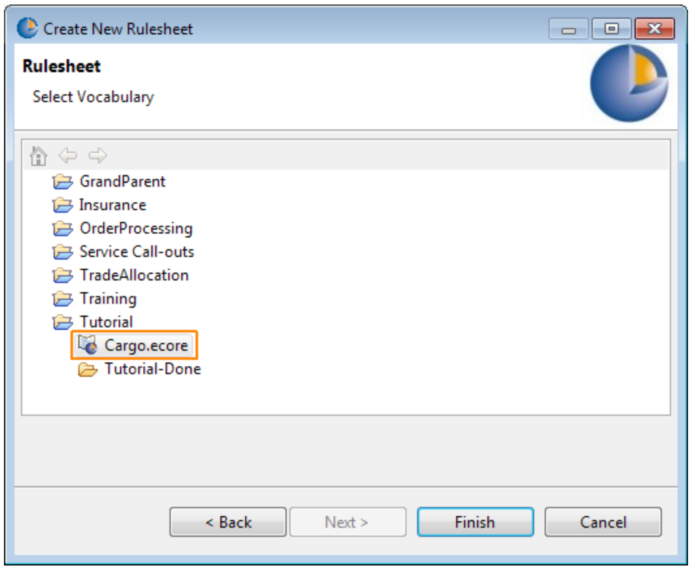
1. To create a new Rulesheet, select the menu item **File > New > Rulesheet**, as shown:



The **Create New Rulesheet** dialog box opens, as shown:

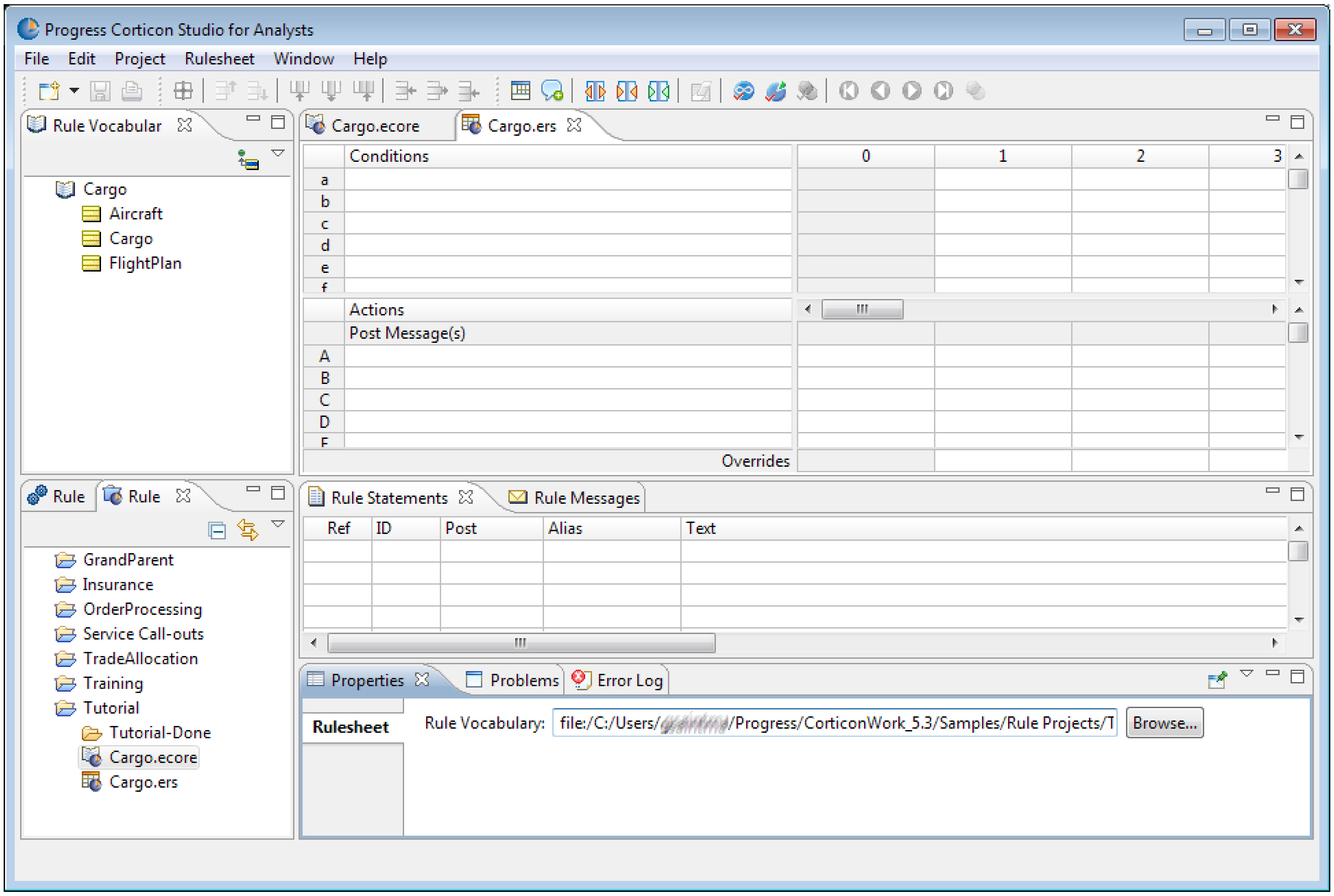


2. Click on **Tutorial** as the **Parent Folder**, enter `Cargo` as the **File name**, and then click **Next**.



3. Select `Cargo.ecore` as the Vocabulary to associate with your new Rulesheet, and then click **Finish**.

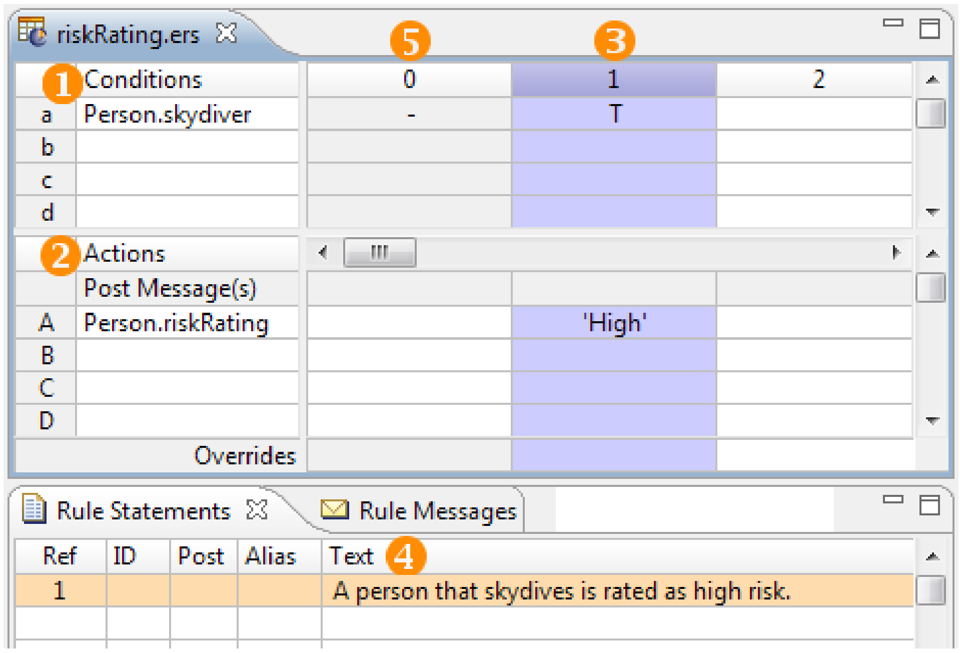
A `Cargo.ers` tab opens in the editing area with the Rulesheet layout, as shown:



Rule Vocabulary: file:/C:/Users/[redacted]/Progress/CorticonWork_5.3/Samples/Rule Projects/T

About Corticon Rulesheets

The Corticon Rulesheet is the metaphor you use to model your business rules. A Corticon Rulesheet is extended from a basic decision table, so that you can model business rules logic from simple tests to complex inferencing problems. See the *Advanced Rule Modeling Tutorial* and *Rule Modeling Guide* for some complex examples.



	Conditions	0	1	2
a	Person.skydiver	-	T	
b				
c				
d				
	Actions			
	Post Message(s)			
A	Person.riskRating		'High'	
B				
C				
D				
	Overrides			

Ref	ID	Post	Alias	Text
1				A person that skydives is rated as high risk.

Rulesheets contain sections for specific parts of rules. Sets of Conditions (label 1) and Actions (label 2) are tied together by the vertical columns on the right to form rules. For example, rule column 1 (label 3) is read as “if a Person's skydiver value is true, then assign ‘High’ to that Person's riskRating. We say that this column provides the model or implementation of the rule statement (label 4). Column 0 (label 5) is used to define calculation rules that contain Actions, but no Conditions (see *Rule Modeling Guide* for examples).

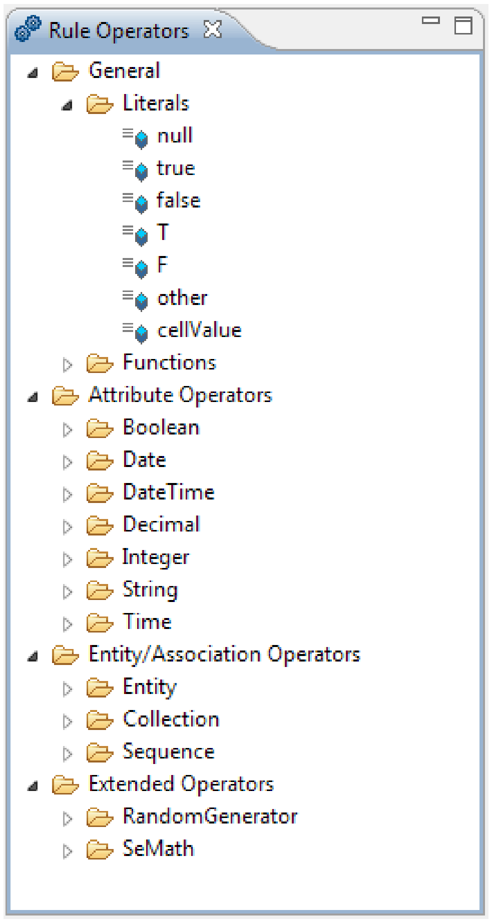
This basic tutorial weaves together a few inter-related rules from our Vocabulary to get you familiar with fundamental modeling tasks.

Explore the Rules Operators

While the **Rule Vocabulary** contains our nouns (the *things* we reference in our business rules), the **Rule Operators** contain our verbs (the *actions* we take in our business rules).

Corticon Studio has a rich set of operators for manipulating data, similar to the Excel function library. In addition, programmers can add new Extended Operators through an API (for example, to perform a new statistical function or to make a call out). See the *Rule Language Guide* for more information.

The **Rule Operators** view categorizes the operators, as shown:


Rule Operators
General
Literals
null
true
false
T
F
other
cellValue
Functions
Attribute Operators
Boolean
Date
DateTime
Decimal
Integer
String
Time
Entity/Association Operators
Entity
Collection
Sequence
Extended Operators
RandomGenerator
SeMath

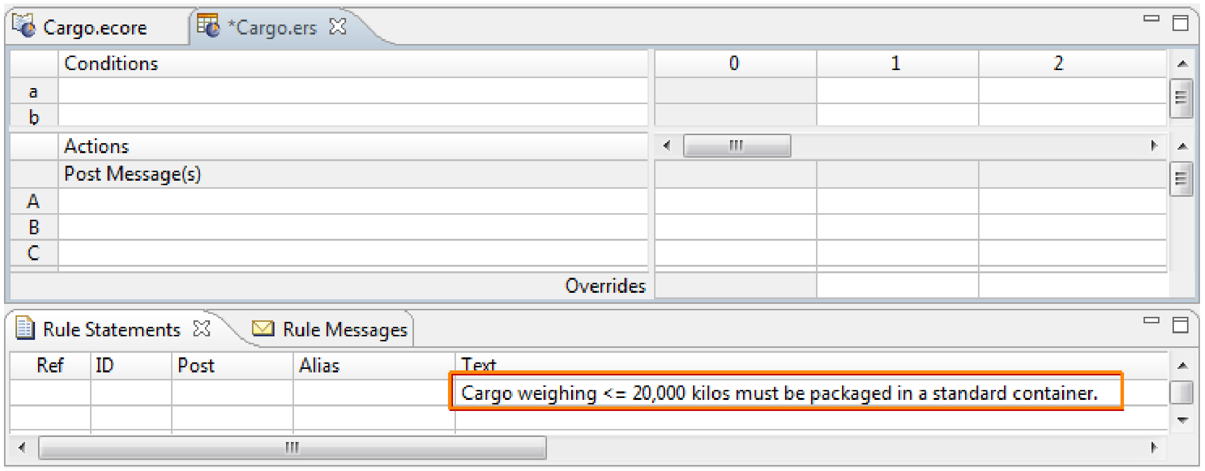
5

Steps that define Rules in the Model

To model business rules:

1. Enter the plain-language business rule into the **Rule Statements** section of the Rulesheet as shown. You can copy and paste the text:

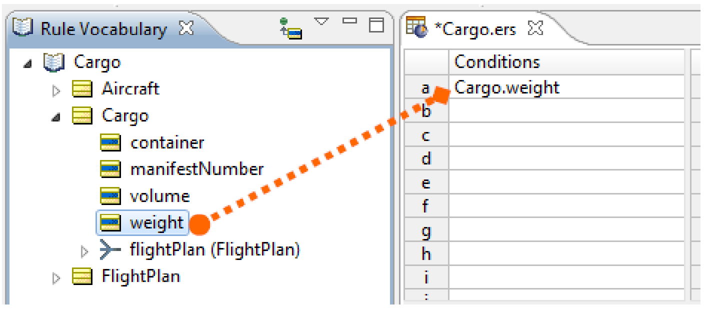
   `Cargo weighing <= 20,000 kilos must be packaged in a standard container.`



Cargo weighing <= 20,000 kilos must be packaged in a standard container.

   The condition of our first rule: `Cargo weighing <= 20000` can be restated as `Cargo.weight <= 20000`. Let's model this logic in Corticon Studio.

2. Drag the `Cargo` attribute `weight` from the Vocabulary and drop it in Row **a** of the Conditions pane, as shown:

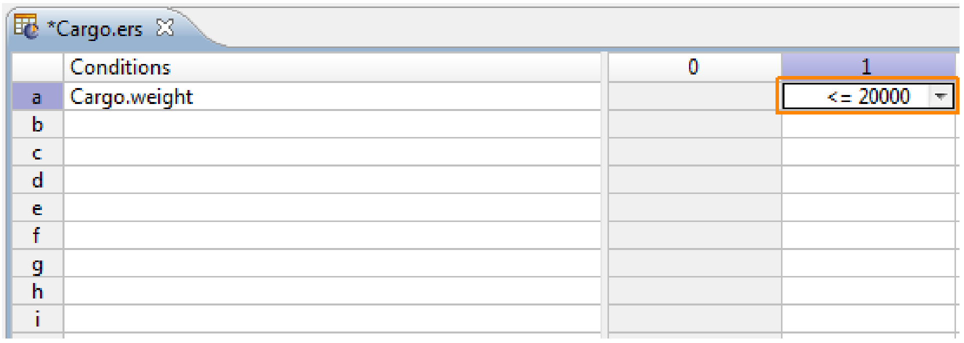


Note: We use an orange dotted line to indicate dragging and dropping. Drag *from* the orange circle, and drop *on* the orange diamond.

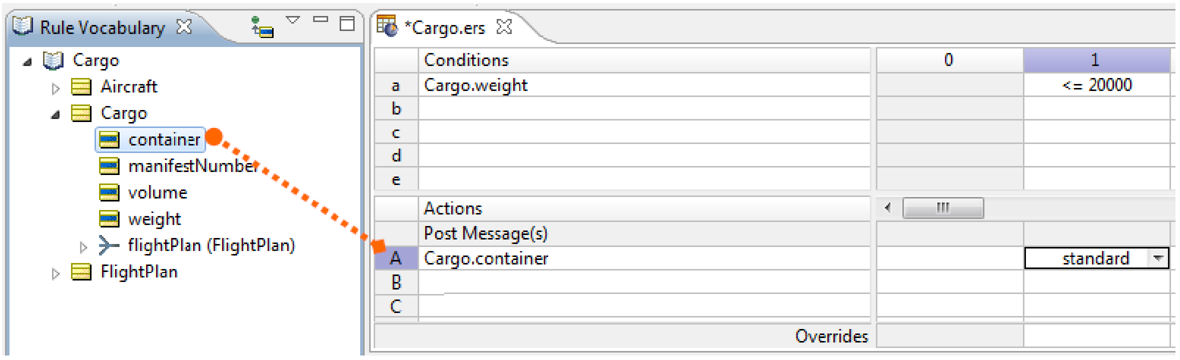
Our first rule evaluates a condition and takes an action if -- and only if -- the condition is satisfied. Therefore, we will use the **Conditions** and **Actions** panes to model the rule.

3. Type the value for cargo weight in cell 1a (row a, column 1): `<=20000`.

Note: Do not use commas in values; commas are separators for multiple values.



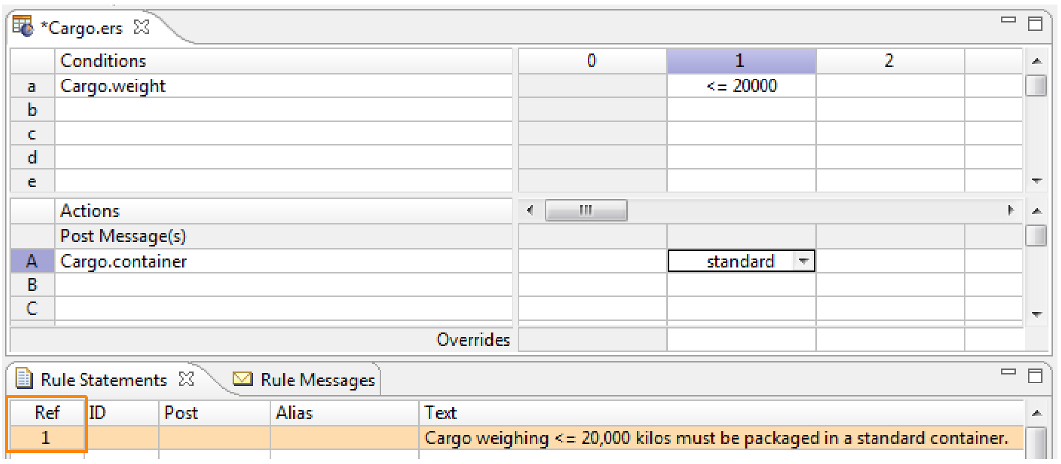
4. Drag the appropriate term from the Vocabulary and drop it in row A of the **Actions** section.



5. Define the action for rule 1 by selecting "standard" in the drop-down menu within **Actions** row A of column (rule) 1. The drop-down menu options were defined in the Vocabulary, which we will edit later.

   This action assigns the value of "standard" to the **Cargo.container** attribute.

6. Enter the Rule Statement's **Reference** number. This connects the Rule Statement to the corresponding model in a column in the Rulesheet above.



*Cargo.ers

	Conditions	0	1	2
a	Cargo.weight		<= 20000	
b				
c				
d				
e				
	Actions			
	Post Message(s)			
A	Cargo.container		standard	
B				
C				
	Overrides			

Rule Statements | Rule Messages

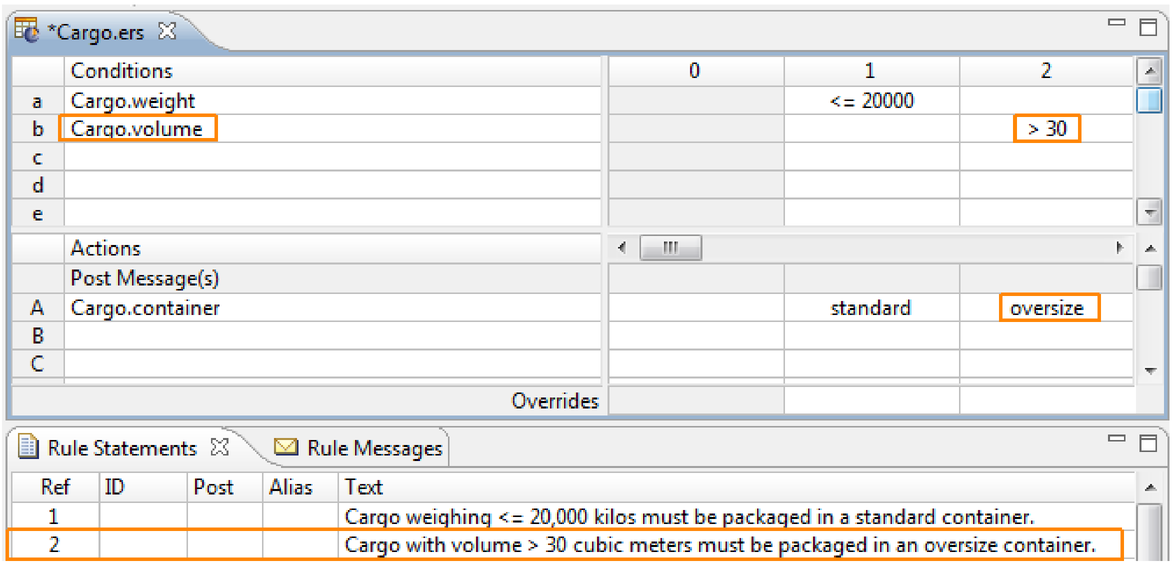
Ref	ID	Post	Alias	Text
1				Cargo weighing <= 20,000 kilos must be packaged in a standard container.

Reference numbers logically link rows in the **Rule Statements** section to columns in the **Conditions/Actions** section – notice how clicking on Rule Statement Ref 1 causes column 1 to highlight. The highlighting reminds you that column 1 is the *model* of the rule (expressed in plain-language) in Rule Statement Ref 1.

7. Now model the second rule. Its plain-language business rule is:

```
Cargo with volume > 30 cubic meters must be packaged in an oversize
container.
```

The steps are similar to our first rule, as highlighted:



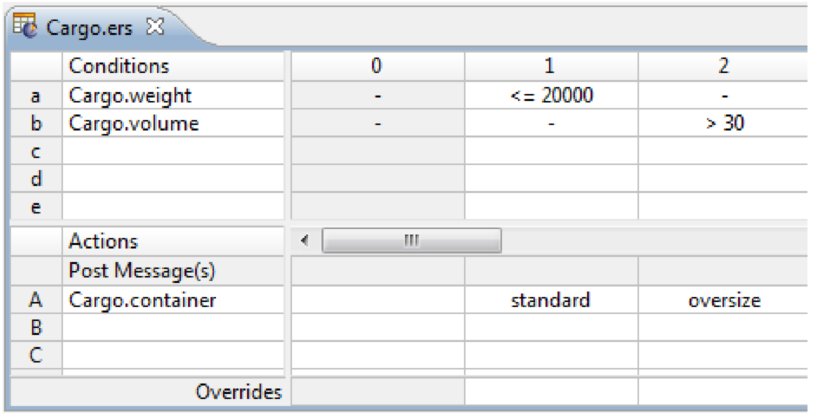
*Cargo.ers

	Conditions	0	1	2
a	Cargo.weight		<= 20000	
b	Cargo.volume			> 30
c				
d				
e				
	Actions			
	Post Message(s)			
A	Cargo.container		standard	oversize
B				
C				
	Overrides			

Rule Statements | Rule Messages

Ref	ID	Post	Alias	Text
1				Cargo weighing <= 20,000 kilos must be packaged in a standard container.
2				Cargo with volume > 30 cubic meters must be packaged in an oversize container.

8. Save your Rulesheet by selecting **File** > **Save** on the menubar (or clicking the **Save** button).



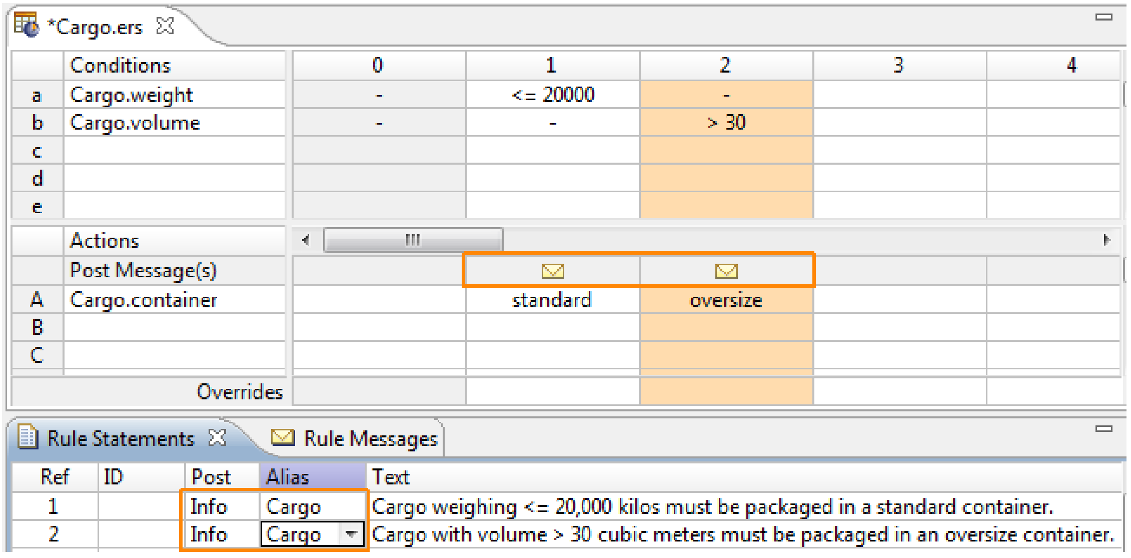
	Conditions	0	1	2
a	Cargo.weight	-	<= 20000	-
b	Cargo.volume	-	-	> 30
c				
d				
e				
	Actions			
	Post Message(s)			
A	Cargo.container		standard	oversize
B				
C				
	Overrides			

When saved, Corticon Studio places a dash in the empty cells, meaning that the condition is ignored.

The conditions and actions in a column are "anded" together -- the conditions and actions in a column are joined together to complete a rule. For example, rule 1 reads: "Cargo weighing less than or equal to 20,000 kilos, *ignoring volume*, must be packaged in a standard container."

Next, we will "Post" the Rule Statements to the "Cargo" entity. This will provide an audit trail during rule execution, which you will see during rule testing.

9. For each of the rule statements, click in the **Post** column to select the appropriate Severity Level, `Info` on the drop-down menu. Then click in the **Alias** column select the one option, `Cargo`.



	Conditions	0	1	2	3	4
a	Cargo.weight	-	<= 20000	-		
b	Cargo.volume	-	-	> 30		
c						
d						
e						
	Actions					
	Post Message(s)		✉	✉		
A	Cargo.container		standard	oversize		
B						
C						
	Overrides					

Ref	ID	Post	Alias	Text
1		Info	Cargo	Cargo weighing <= 20,000 kilos must be packaged in a standard container.
2		Info	Cargo	Cargo with volume > 30 cubic meters must be packaged in an oversize container.

Once you define a Post, an icon is displayed in the **Post Message(s)** row of the Rulesheet in the referenced column.

6

Conflicts and completeness

This chapter shows how you can check, identify, and resolve errors related to conflicts and errors.

For details, see the following topics:

- Check for conflicts
- Check for completeness
- Check for logical loops
- Save the Rulesheet

Check for conflicts

Now that we have finished modeling our rules, it is time to analyze our rules for logical errors. Very often, initial business rule specifications are **ambiguous** and **incomplete**. By **ambiguous**, we mean that the rules are **conflicting** under certain scenarios. By **incomplete**, we mean that the rules fail to address all possible scenarios.

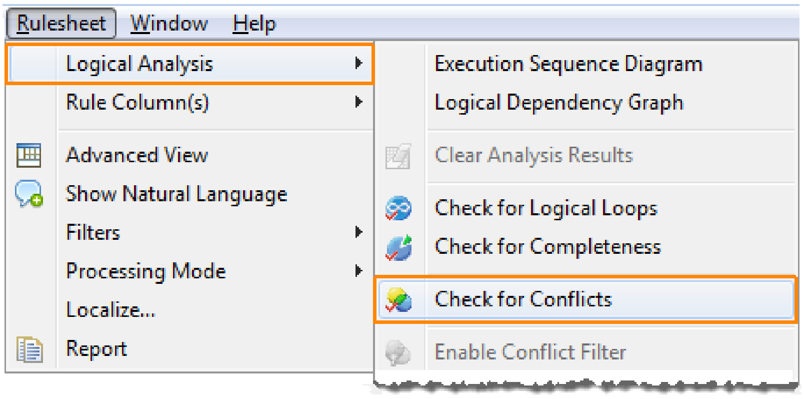
Prior to automating your rules, it is critical to eliminate logical errors in order to ensure that our decision service provides correct, consistent results. We call this "Rule Referential Integrity". Corticon Studio provides unique and powerful features to help you ensure Rule Referential Integrity. These features will be explored in this Analyze phase of the rule management lifecycle.

Be sure the **Cargo.ers** tab is selected, then Check for Conflicts.

To check for conflicts, either:

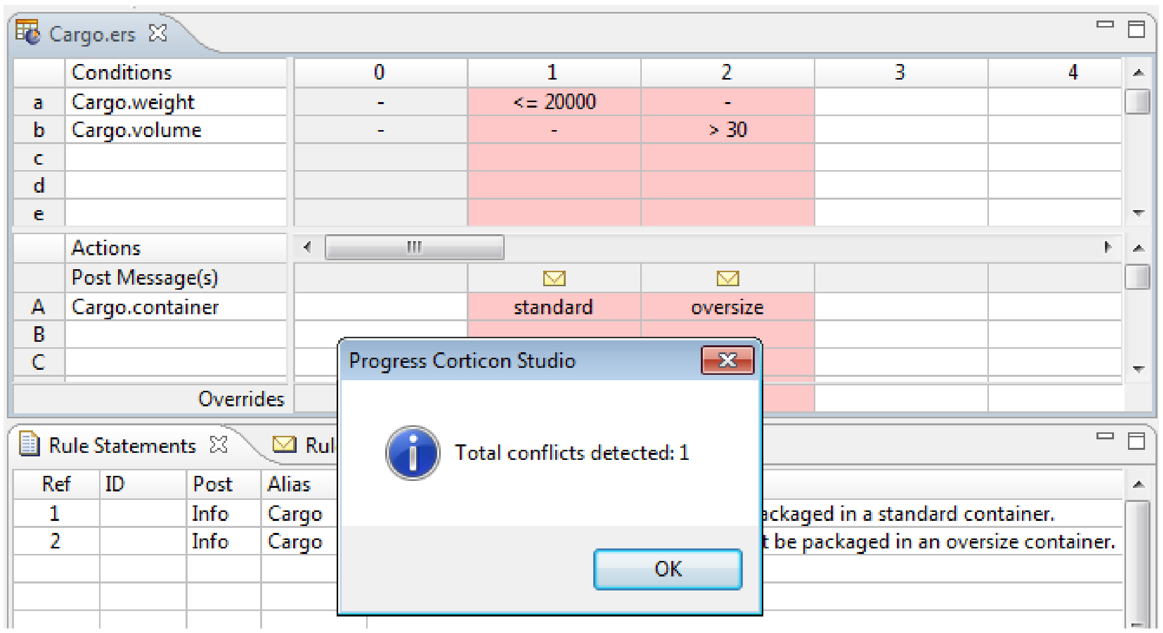
- Click the **Check for Conflicts** button on the Corticon Studio toolbar.

- Choose the menu command **Rulesheet > Logical Analysis > Check for Conflicts**



Identify conflicts

Columns containing conflicts are shaded in pink. This indicates that one or more conflicts exist between the pink rules.



	Conditions	0	1	2	3	4
a	Cargo.weight	-	<= 20000	-		
b	Cargo.volume	-	-	> 30		
c						
d						
e						

	Actions	0	1	2	3	4
	Post Message(s)		✉	✉		
A	Cargo.container		standard	oversize		
B						
C						
	Overrides					

Progress Corticon Studio

Total conflicts detected: 1

OK

Ref	ID	Post	Alias	
1		Info	Cargo	ackaged in a standard container.
2		Info	Cargo	t be packaged in an oversize container.

The total number of conflicts detected is displayed.

Click **OK**.

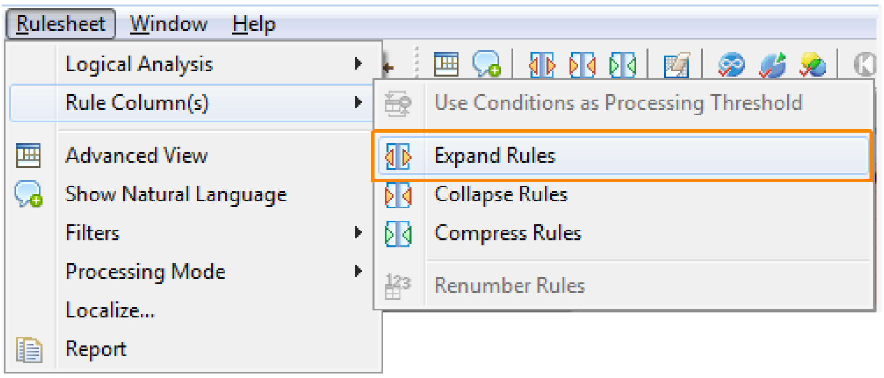
Expand the rules

We need to expand the Rules to reveal sub-rules (rules without dashes) inside the general rules (rules with at least one dash value). This accurately pinpoints the source of the ambiguity.

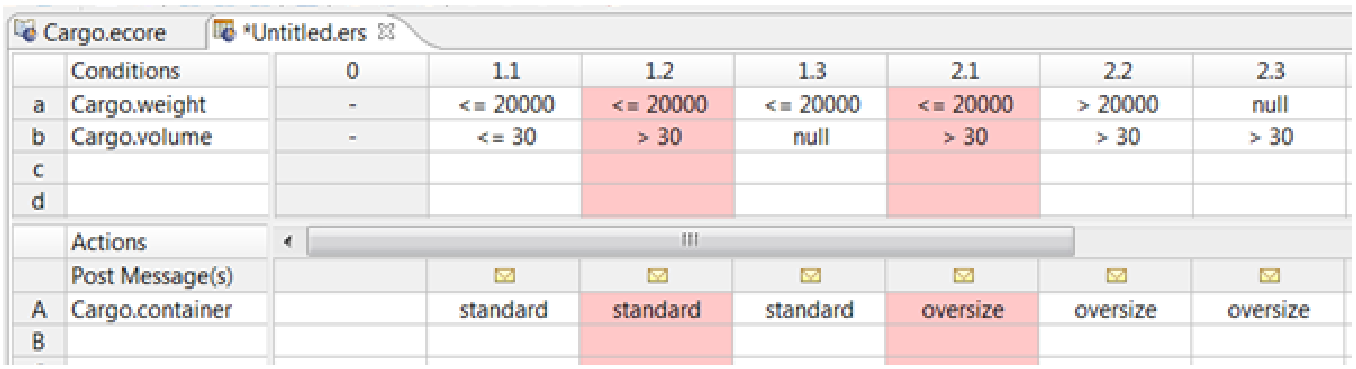
To expand the rules, either:

- Click the **Expand Rules** button on the Corticon Studio toolbar.

- Choose the menu command **Rulesheet > Rule Column(s) > Expand Rules**.



The rules expand by adding columns, as shown:



Cargo.ecore | *Untitled.ers

	Conditions	0	1.1	1.2	1.3	2.1	2.2	2.3
a	Cargo.weight	-	<= 20000	<= 20000	<= 20000	<= 20000	> 20000	null
b	Cargo.volume	-	<= 30	> 30	null	> 30	> 30	> 30
c								
d								

	Actions							
	Post Message(s)		✉	✉	✉	✉	✉	✉
A	Cargo.container		standard	standard	standard	oversize	oversize	oversize
B								

When expanded, rule 1 is represented as three columns (1.1, 1.2 and 1.3), and rule 2 is represented as three columns (2.1, 2.2 and 2.3). Expansion shows all of the logical possibilities for each rule. Rule 1 states Cargo weighing <= 20,000 kilograms, *regardless of volume*, must be packaged in a standard container, unless no volume was ever stated (a *null* value). Corticon Studio recognizes that there are three possible ranges for volume (<=30, >30, and *null*), which we see in the expanded state.

Note:

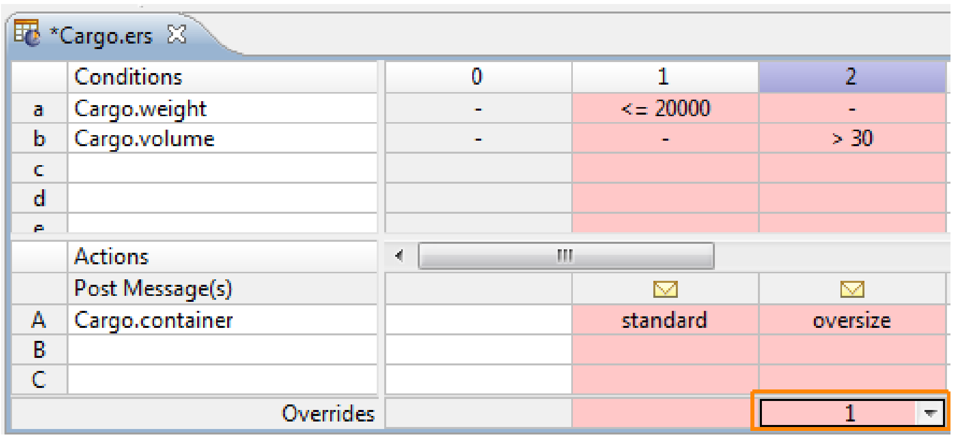
Looking at the expanded rules, the source of the conflict becomes apparent: In scenarios where cargo weight <=20000 **and** cargo volume > 30, our rules are in conflict, defining mutually exclusive actions (assigning both standard *and* oversize containers).

To ensure well-formed rules, this scenario must be addressed!

Resolve conflict errors

To resolve the conflict, you can either change your original rules, or decide that one rule should override the other. Let's implement the override. We'll override Rule 1 with Rule 2.

1. Collapse your rules back to the original state by clicking the **Collapse Rules** button on the Corticon Studio toolbar.
2. In the Overrides row, in the *overriding* rule's column, select the column number of the rule to *be overridden*. Multiple selections can be made by holding the **CTRL** key while you click.

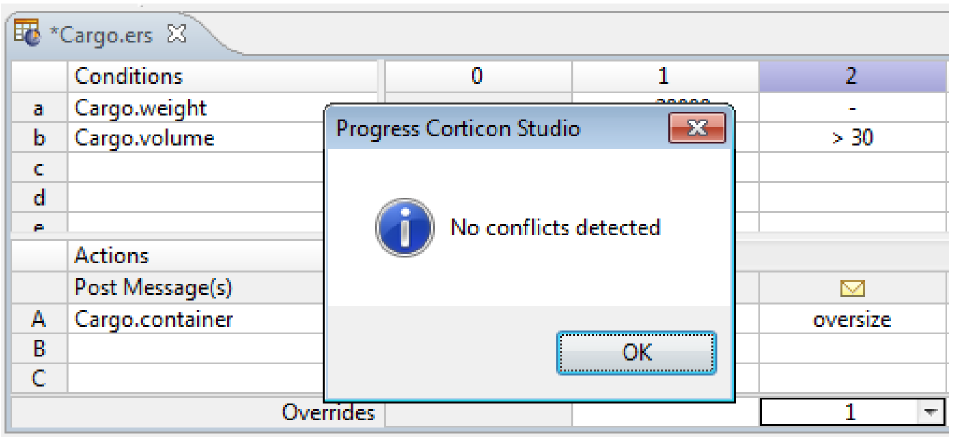


*Cargo.ers

	Conditions	0	1	2
a	Cargo.weight	-	<= 20000	-
b	Cargo.volume	-	-	> 30
c				
d				
e				
	Actions			
	Post Message(s)		✉	✉
A	Cargo.container		standard	oversize
B				
C				
	Overrides			1

3. Click the **Check for Conflicts** button again, and you will 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 <=20,000."



*Cargo.ers

	Conditions	0	1	2
a	Cargo.weight			-
b	Cargo.volume			> 30
c				
d				
e				
	Actions			
	Post Message(s)			✉
A	Cargo.container			oversize
B				
C				
	Overrides			1

Progress Corticon Studio

No conflicts detected

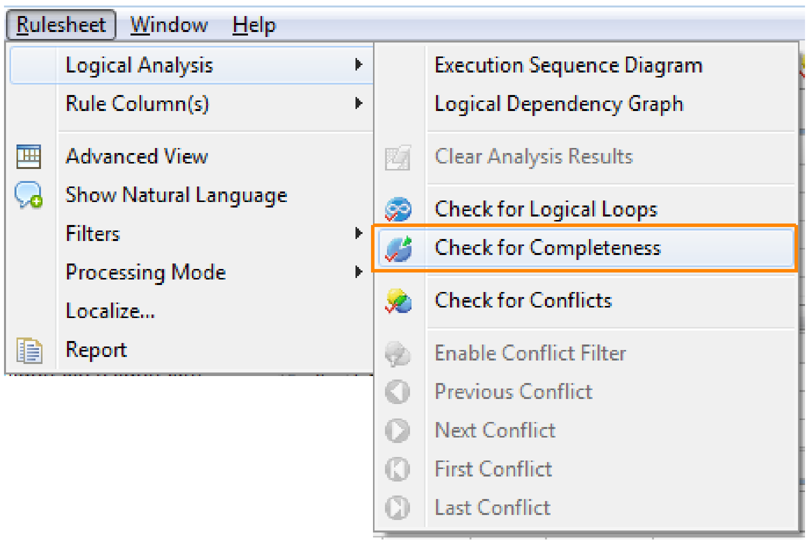
OK

Check for completeness

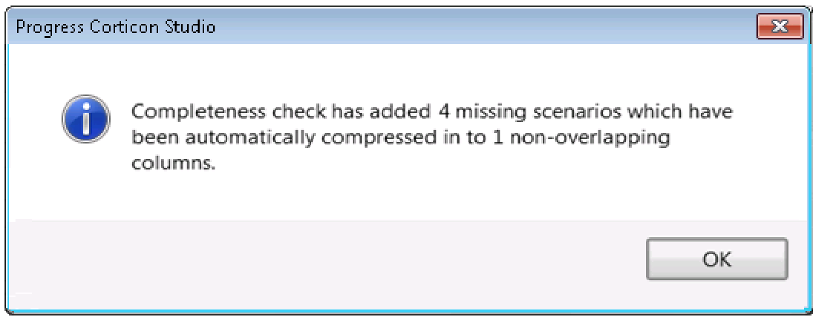
Conflict is one form of logical error. Another form is incompleteness, or loopholes in our logic.

To check for logical errors in the Rulesheet, either:

- Click the **Check for Completeness** button on the Corticon Studio toolbar.
- Choose the menu command **Rulesheet > Logical Analysis > Check for Completeness**



A message window informs us that our rules are incomplete. We missed some scenarios.



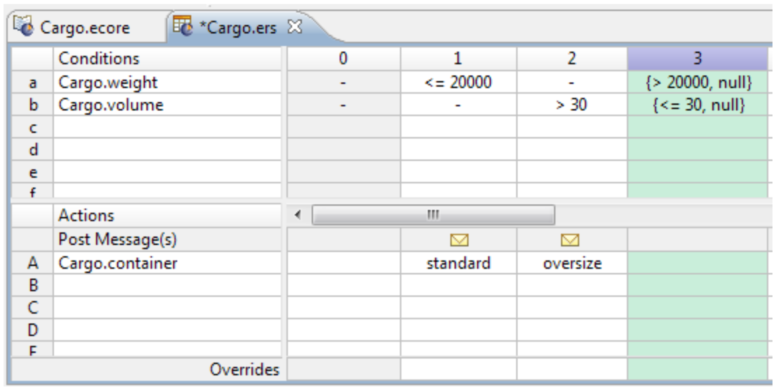
Progress Corticon Studio

Completeness check has added 4 missing scenarios which have been automatically compressed in to 1 non-overlapping columns.

Click **OK** to dismiss the window.

Resolve completeness errors – step 1

The Completeness Checking algorithm calculates the set of all possible mathematical combinations of all 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 shaded in green.



	Conditions	0	1	2	3
a	Cargo.weight	-	<= 20000	-	{> 20000, null}
b	Cargo.volume	-	-	> 30	{<= 30, null}
c					
d					
e					
f					
	Actions				
	Post Message(s)		✉	✉	
A	Cargo.container		standard	oversize	
B					
C					
D					
E					
	Overrides				

The Completeness Check adds condition values, but does not choose actions – that's our job as rule modelers.

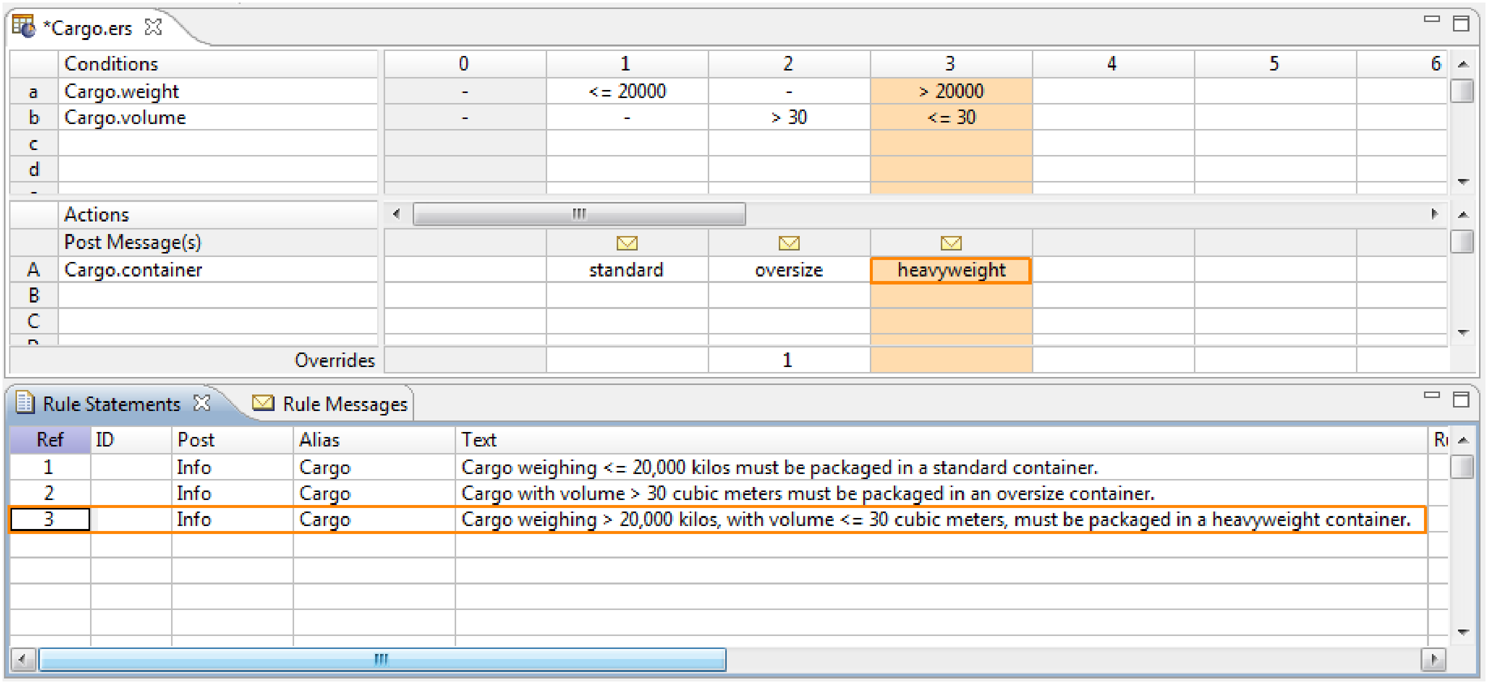
Let's define a new Rule Statement for this (formerly) missing scenario:

```
Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be
packaged in a heavyweight container.
```

Resolve completeness errors – step 2

In this example, incompleteness will be resolved by specifying an action for column 3, and then adding a new Rule Statement.

First, add the new Rule Statement for rule 3. Next, define an action in rule cell 3A. In this case, select "heavyweight" as the container option. Last, post an **Info** message to the Cargo entity as we did for the first two rules, and use its **Reference** to link it with the corresponding column. Your Rulesheet will look like this:



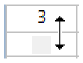
	Conditions	0	1	2	3	4	5	6
a	Cargo.weight	-	<= 20000	-	> 20000			
b	Cargo.volume	-	-	> 30	<= 30			
c								
d								

	Actions							
	Post Message(s)		✉	✉	✉			
A	Cargo.container		standard	oversize	heavyweight			
B								
C								
	Overrides			1				

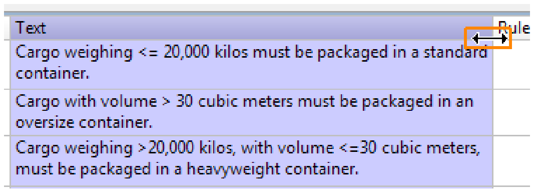
Ref	ID	Post	Alias	Text
1		Info	Cargo	Cargo weighing <= 20,000 kilos must be packaged in a standard container.
2		Info	Cargo	Cargo with volume > 30 cubic meters must be packaged in an oversize container.
3		Info	Cargo	Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be packaged in a heavyweight container.

Click the **Clear Analysis Results** button on the Corticon Studio toolbar to remove the highlighting.

Note: When rule statements get long, you can do line wrapping by first clicking and dragging to a larger row height:



Then, as you drag the **Text** area narrower, the statement text wraps into the assigned row height:



Cargo weighing <= 20,000 kilos must be packaged in a standard container.
Cargo with volume > 30 cubic meters must be packaged in an oversize container.
Cargo weighing >20,000 kilos, with volume <=30 cubic meters, must be packaged in a heavyweight container.

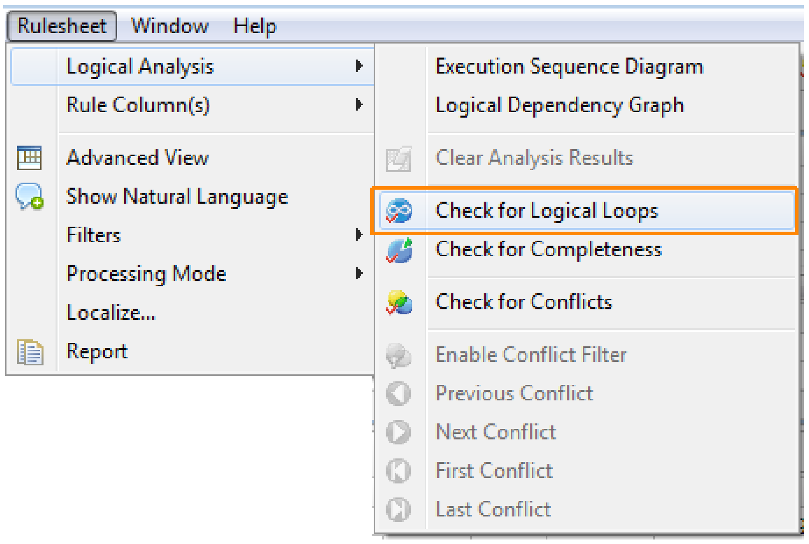
Check for logical loops

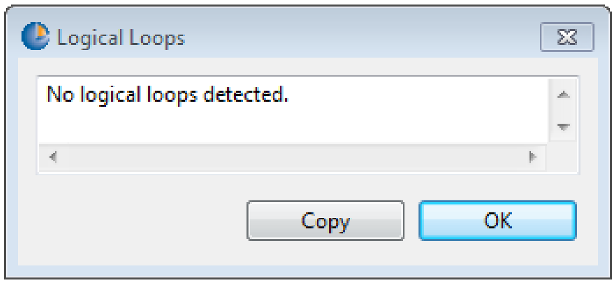
A third form of logical error is circular logic.

To check for logical errors in the Rulesheet.

Perform one of the following:

- Click the **Check for Logical Loops** button on the toolbar.
- Choose **Rulesheet** > **Logical Analysis** > **Check for Logical Loops** from the Corticon Studio menubar.





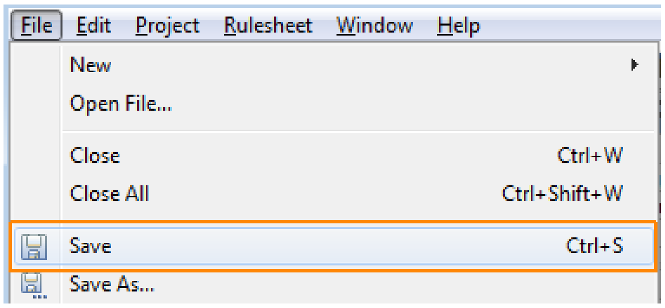
No logical loops detected.

This very simple Rulesheet contains no logical loops. While unintended logical loops are problematic, sometimes logical loops are a useful technique for solving complex inferencing problems that require recursive reasoning. You can learn more in the *Rule Modeling Guide*.

Save the Rulesheet

To save your Rulesheet, either:

- Click the **Save** button on the toolbar.
- Choose the menu command **File > Save**.



7

Testing rules

This chapter shows how you can create, setup, and verify a ruletest.

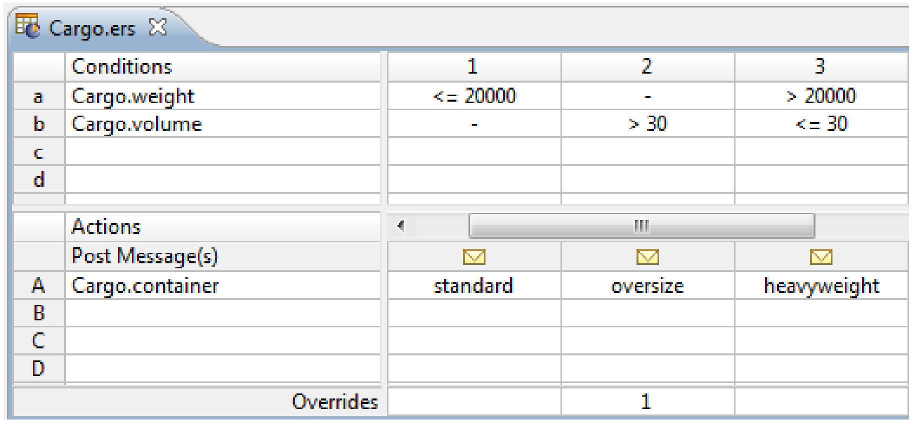
For details, see the following topics:

- Defining a test case
- Creating a Ruletest
- Setting the Test Subject
- Set up the test scenario – step 1
- Set up the test scenario – step 2
- Set up the test scenario – step 3
- Execute the Ruletest
- Verify the test results
- Changing the test
- Saving the Ruletest

Defining a test case

The Analyze phase helped to ensure the logical integrity of our rules. The Test phase helps to ensure that our rules give us the correct business results. Let's move on to testing.

First, we're going to define a test case for each one of our rules below by defining some input values and expected results. Corticon Studio allows us to pre-define our expected results and then highlights any variances in our test results. The table below defines one test case for each rule.



	Conditions	1	2	3
a	Cargo.weight	<= 20000	-	> 20000
b	Cargo.volume	-	> 30	<= 30
c				
d				
	Actions			
	Post Message(s)	✉	✉	✉
A	Cargo.container	standard	oversize	heavyweight
B				
C				
D				
	Overrides		1	

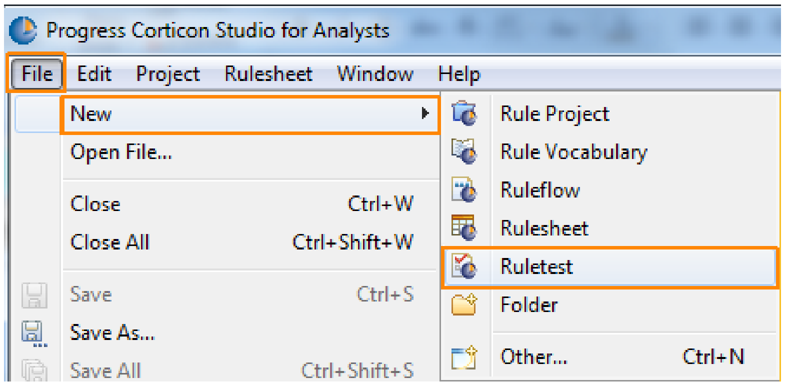
	Test Rule 1	Test Rule 2	Test Rule 3
Input Values			
Cargo.weight	1000	1000	30000
Cargo.volume	10	40	20
Expected Results			
Cargo.container	standard	oversize	heavyweight

For test 2, we expect rule 2 to override rule 1.

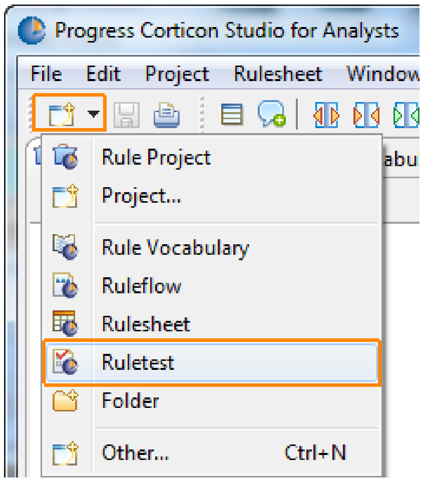
Creating a Ruletest

To create a Ruletest:

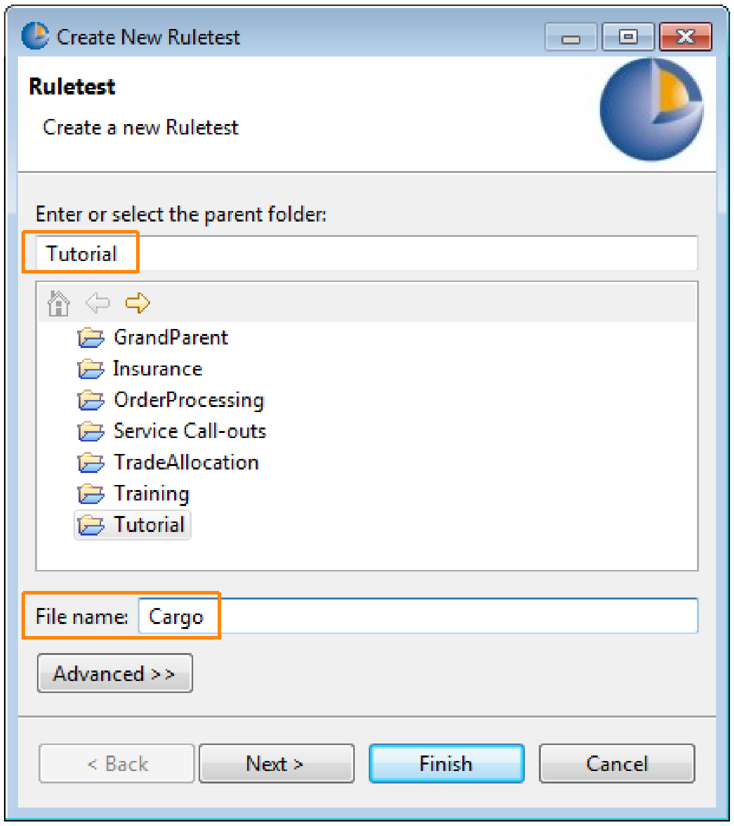
1. Perform one of the following:
    - Select the menu command **File > New > Ruletest**.



- Click to dropdown the toolbar's **New** options, and then select **Ruletest**.



2. Select **Tutorial** as the Parent Folder and name the file `Cargo`.

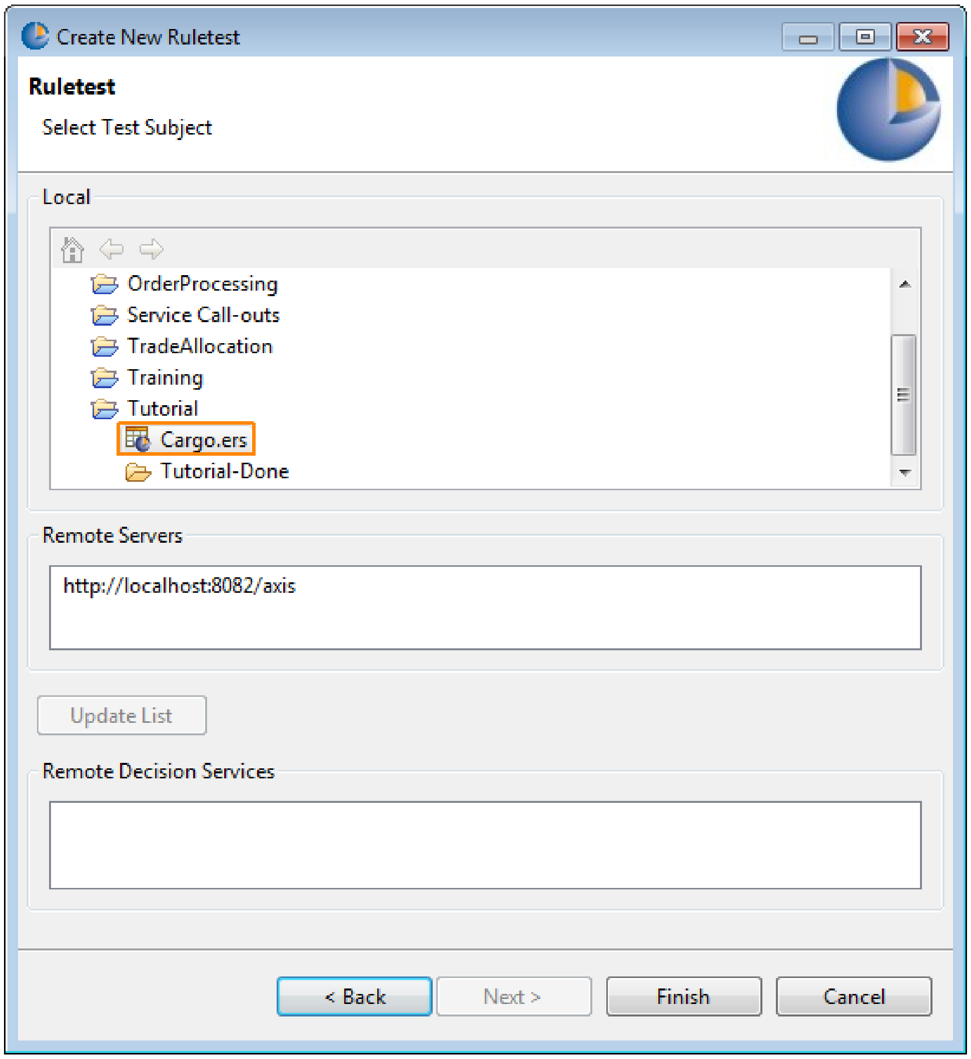


All test files are assigned a file extension of `.ert`.

3. Click **Next >** to continue.

Setting the Test Subject

Select the appropriate Rulesheet as your **Test Subject**. For our example, we want to use the `Cargo` Rulesheet within our `Tutorial` project directory.



Click **Finish** to display the new Ruletest.

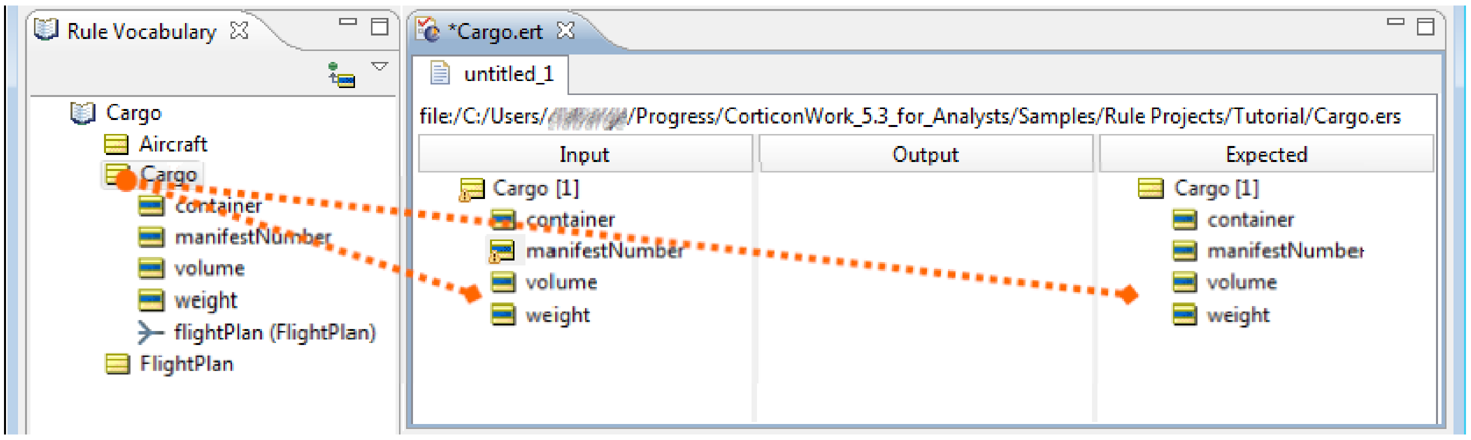
Note: You can disregard the **Remote Servers** and **Remote Decision Service** panes - if you are interested in those, refer to the *Server: Deploying Web Services* guides for more information.

Set up the test scenario – step 1

Drag the `Cargo` entity from the Vocabulary and drop it anywhere in the Ruletest's **Input** and **Expected** panes, as shown. `Cargo[1]` on the Ruletest's **Input** and **Expected** panes represents one ‘instance’ or ‘example’ of the Cargo entity.

As you can see, your new Ruletest is associated with the appropriate Rulesheet. This ensures that your Ruletest scenario will be tested by the right rules.

Note: When you make changes to a Corticon file, an asterisk is placed in the file's tab as a reminder to save the file.



file:/C:/Users/[redacted]/Progress/CorticonWork_5.3_for_Analysts/Samples/Rule Projects/Tutorial/Cargo.ers

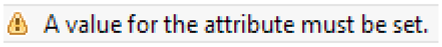
Set up the test scenario – step 2

	Test Rule 1	Test Rule 2	Test Rule 3
Input Values			
Cargo.weight	1000	1000	30000
Cargo.volume	10	40	20
Expected Results			
Cargo.container	standard	oversize	heavyweight

First, remove unneeded attributes by selecting them and pressing the **Delete** key. **Shift**-click and **Ctrl**-click are also supported for multi-selecting contiguous and non-contiguous attributes, respectively.

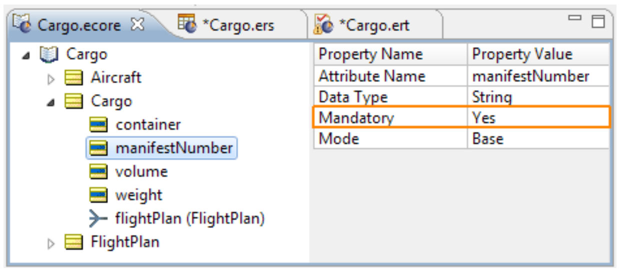
We will remove the `manifestNumber` attribute as it is not needed to test these rules.

Note: You may have noticed in the **Input** column that the `Cargo` entity and the `manifestNumber` attribute both have a warning 'decoration' with the problem description:



A value for the attribute must be set.

That's because that attribute was preset as ***mandatory*** in the Vocabulary:

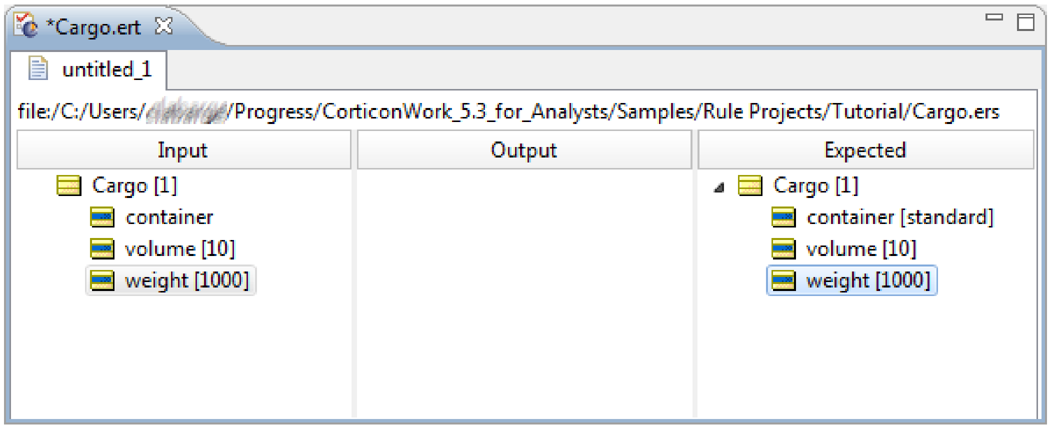


- Cargo
  - Aircraft
  - Cargo
    - container
    - manifestNumber
    - volume
    - weight
    - flightPlan (FlightPlan)
  - FlightPlan

Property Name	Property Value
Attribute Name	manifestNumber
Data Type	String
Mandatory	Yes
Mode	Base

You can see in the Vocabulary that an Aircraft's `tailNumber` and a FlightPlan's `flightNumber` are also set as manadatory -- they are each the key to a unique item so they must have a value. See the *Rule Modeling Guide* for more about how mandatory attributes can be used.

Next, add test data for test #1 to the Ruletest, per the table above. Double-clicking on the term you want opens a data entry box. When complete, your Ruletest should look like this.



file:/C:/Users/[redacted]/Progress/CorticonWork_5.3_for_Analysts/Samples/Rule Projects/Tutorial/Cargo.ers

Input	Output	Expected
Cargo [1]		Cargo [1]
container		container [standard]
volume [10]		volume [10]
weight [1000]		weight [1000]

Set up the test scenario – step 3

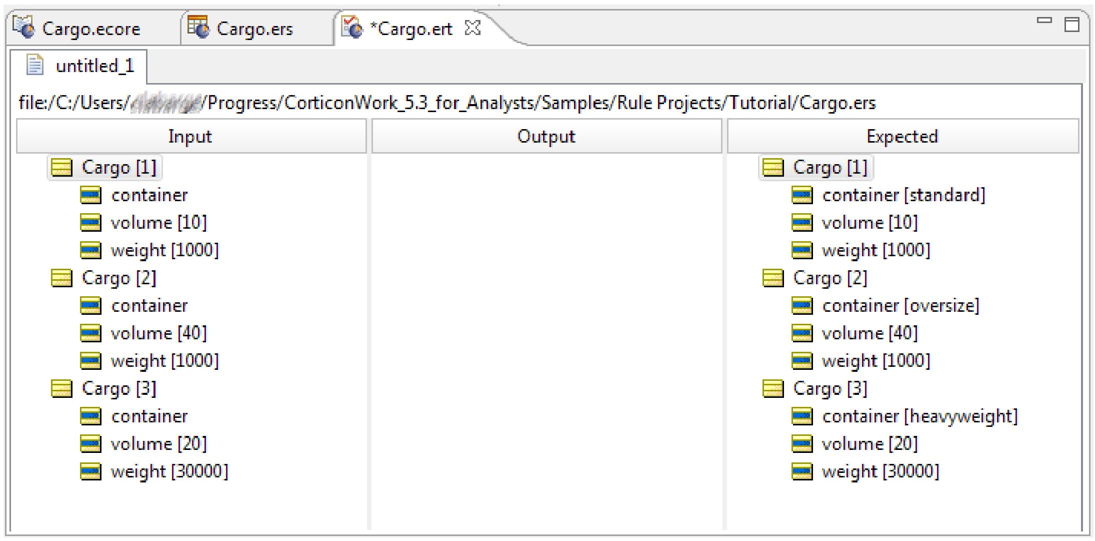
	Test Rule 1	Test Rule 2	Test Rule 3
Input Values			
Cargo.weight	1000	1000	30000
Cargo.volume	10	40	20
Expected Results			
Cargo.container	standard	oversize	heavyweight

Enter the remaining test data as shown to complete the Input pane of the Ruletest.

You can easily duplicate the first test scenario by selecting Cargo [1], copying it, then pasting it below. Repeat for both the Input and Expected panes

Also, select the *expected* container values from the drop-downs that appear when the container attribute is clicked.

Your Ruletest should look like the one below.



Cargo.ecore | Cargo.ers | *Cargo.ert

untitled_1

file:/C:/Users/[redacted]/Progress/CorticonWork_5.3_for_Analysts/Samples/Rule Projects/Tutorial/Cargo.ers

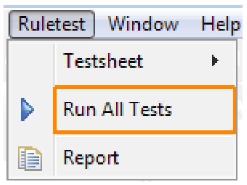
Input	Output	Expected
Cargo [1]		Cargo [1]
container		container [standard]
volume [10]		volume [10]
weight [1000]		weight [1000]
Cargo [2]		Cargo [2]
container		container [oversize]
volume [40]		volume [40]
weight [1000]		weight [1000]
Cargo [3]		Cargo [3]
container		container [heavyweight]
volume [20]		volume [20]
weight [30000]		weight [30000]

Execute the Ruletest

To run the ruletest, either:

- Click the **Run Test** . button on the toolbar.

- Choose the menu command **Ruletest** > **Run All Tests**

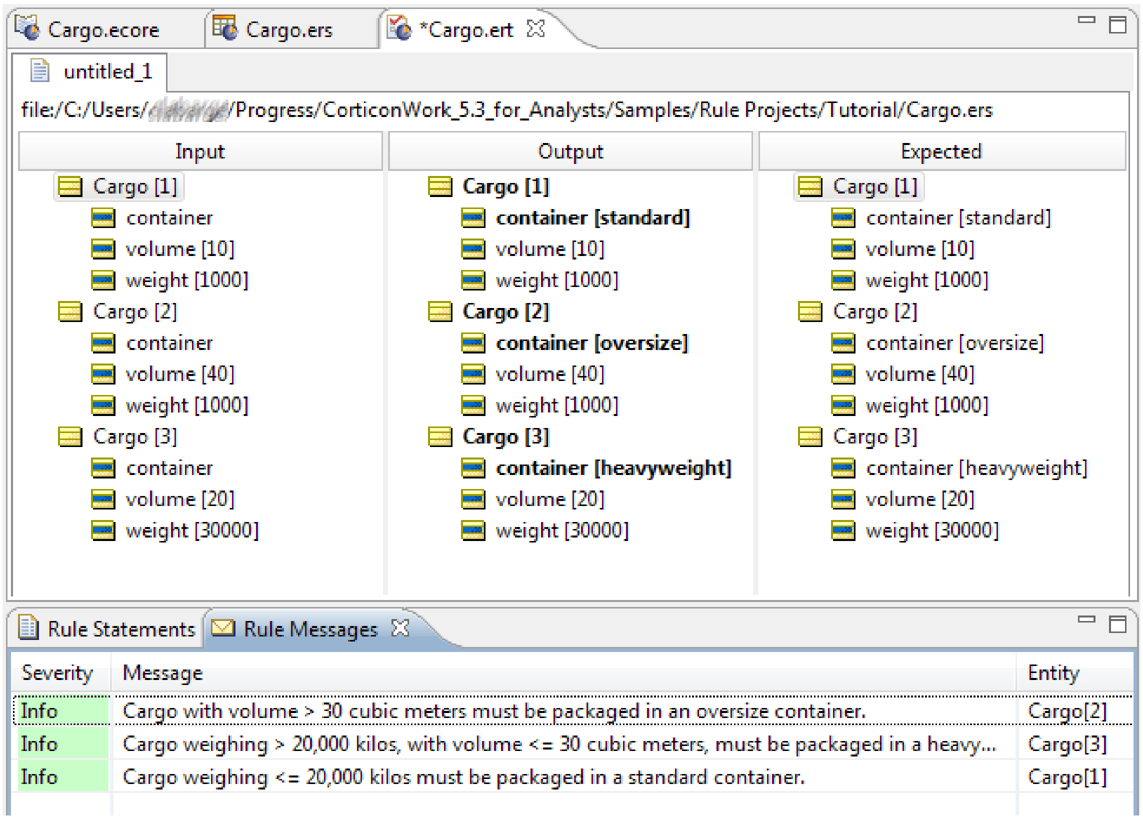


- Choose the menu command **Ruletest** > **Testsheet** > **Run Test**

Note: The first time rules are executed, they are automatically compiled from the rule model into an optimized executable form, then deployed into the engine, which may take a few seconds. Once deployed, the rules will execute much faster on subsequent tests.

Verify the test results

Check the outcome of your test in the Ruletest **Output** pane.



file:/C:/Users/[redacted]/Progress/CorticonWork_5.3_for_Analysts/Samples/Rule Projects/Tutorial/Cargo.ers

Input	Output	Expected
Cargo [1]	**Cargo [1]**	Cargo [1]
container	**container [standard]**	container [standard]
volume [10]	volume [10]	volume [10]
weight [1000]	weight [1000]	weight [1000]
Cargo [2]	**Cargo [2]**	Cargo [2]
container	**container [oversize]**	container [oversize]
volume [40]	volume [40]	volume [40]
weight [1000]	weight [1000]	weight [1000]
Cargo [3]	**Cargo [3]**	Cargo [3]
container	**container [heavyweight]**	container [heavyweight]
volume [20]	volume [20]	volume [20]
weight [30000]	weight [30000]	weight [30000]

Severity	Message	Entity
Info	Cargo with volume > 30 cubic meters must be packaged in an oversize container.	Cargo[2]
Info	Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be packaged in a heavy...	Cargo[3]
Info	Cargo weighing <= 20,000 kilos must be packaged in a standard container.	Cargo[1]

All *unchanged* terms and values in the Ruletest's **Output** pane are displayed in normal style.

Any terms altered by rules, including their parent entities, are shown in **bold** text.

Messages shown in the Rule Messages pane are produced by using the Post command in your Rule Statements, as follows:

- **Severity** indicates whether a message contains information, warnings or violations.

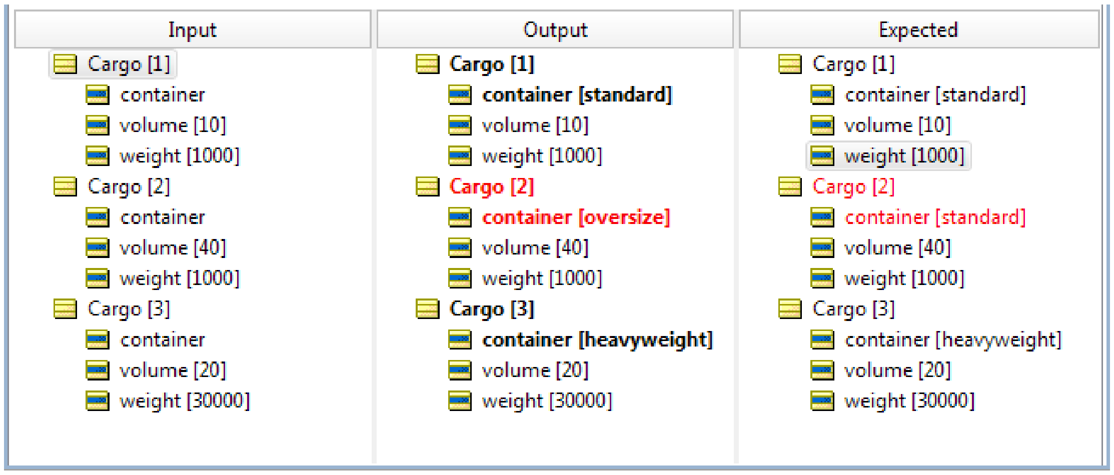
- **Message** contains the Rule Statement text.
- **Entity** shows the entity to which this message is bound or linked. When we select the Cargo[1] entity within the Output pane, the third rule message is highlighted, showing the audit trail of rules that fired for that entity (in this case only one rule fired for Cargo[1]).

Changing the test

Next, let's change one of the test cases in the Expected pane to illustrate how variance testing works.

In the **Expected** pane, change the container value in Cargo[2] to **standard**. Then, rerun the test.

Your Ruletest should look like this:



Input	Output	Expected
Cargo [1]	**Cargo [1]**	Cargo [1]
container	**container [standard]**	container [standard]
volume [10]	volume [10]	volume [10]
weight [1000]	weight [1000]	weight [1000]
Cargo [2]	**Cargo [2]**	Cargo [2]
container	**container [oversize]**	container [standard]
volume [40]	volume [40]	volume [40]
weight [1000]	weight [1000]	weight [1000]
Cargo [3]	**Cargo [3]**	Cargo [3]
container	**container [heavyweight]**	container [heavyweight]
volume [20]	volume [20]	volume [20]
weight [30000]	weight [30000]	weight [30000]

Here we see that the `Cargo[2]` entity and the container attribute are colored red in both the Output and Expected panes, indicating that our Output results differ from our Expected results. Change `Cargo[2]` Expected container value back to **oversize** before saving the Ruletest on the next page.

Corticon Studio automatically highlights differences in values in red in both the Output and Expected panes.

Note: Be sure to click on the sheet away from the edit box so that all the highlights are rendered for the sheet.

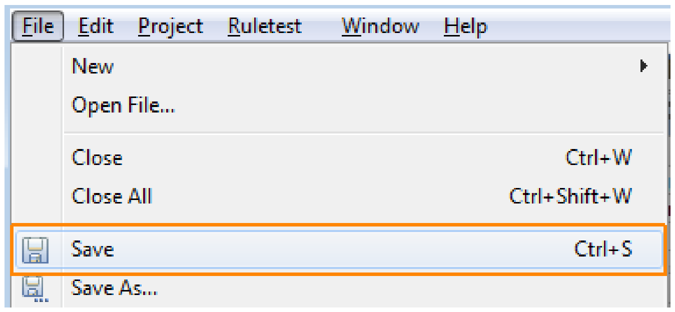
Other types of differences are also highlighted. Unexpected entities in the Output pane are highlighted in blue. And missing entities are highlighted green within the Expected pane. More details about color codes are included in the *Corticon Studio: Quick Reference Guide*.

Saving the Ruletest

To save the Test, either:

- Click the **Save** button on the toolbar.

- Choose **File** > **Save** from the menubar.



8

Modifying the Corticon resources

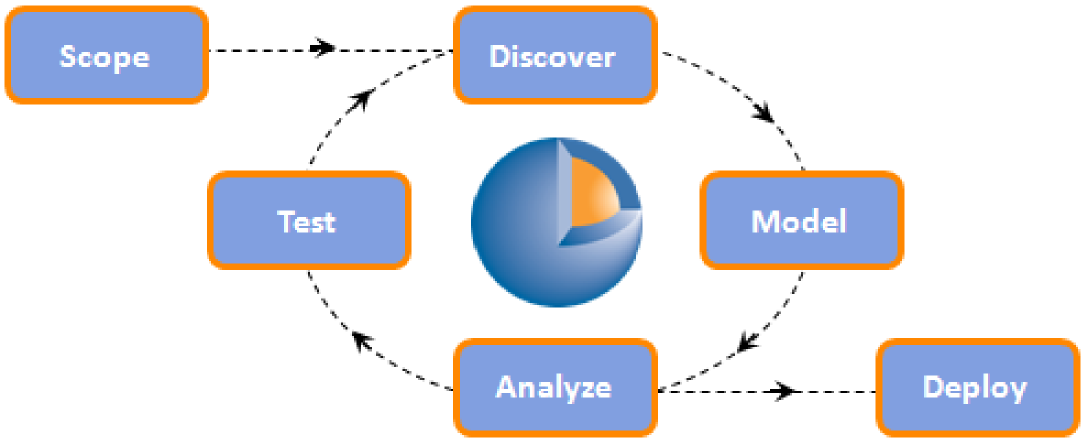
This chapter shows how you can modify the existing Corticon resources.

For details, see the following topics:

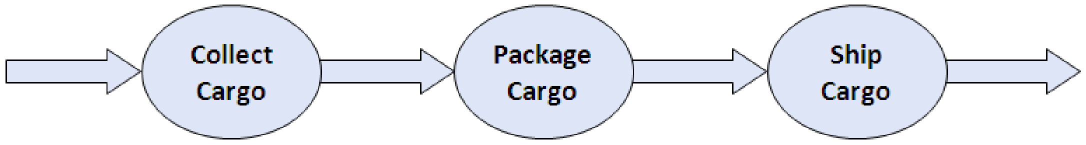
- Discover new business rules
- Add new rule statements
- Open and edit the Vocabulary
- Add an attribute – step 1
- Add an attribute – step 2
- Modify a custom data type
- Save the Vocabulary changes
- Model the new rule
- Check new rules for conflicts
- Setting overrides
- Checking a new rule for completeness
- Modify the Ruletest
- Summary

Discover new business rules

When the time comes to change your rules, you will be glad you decided to model and manage them in Corticon Studio.



Let's go through another development lifecycle, starting with the discovery of a new business rule.



Package Cargo

Initial Rules:

- Cargo weighing <= 20,000 kilos must be packaged in a standard container.
- Cargo with volume > 30 cubic meters must be packaged in an oversize container.

Added after Completeness Check:

- Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be packaged in a heavyweight container.

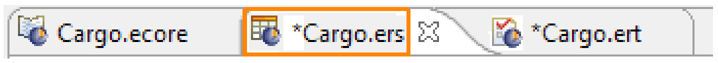
New Rule (to add now):

- Cargo requiring refrigeration must be packaged in a reefer container.

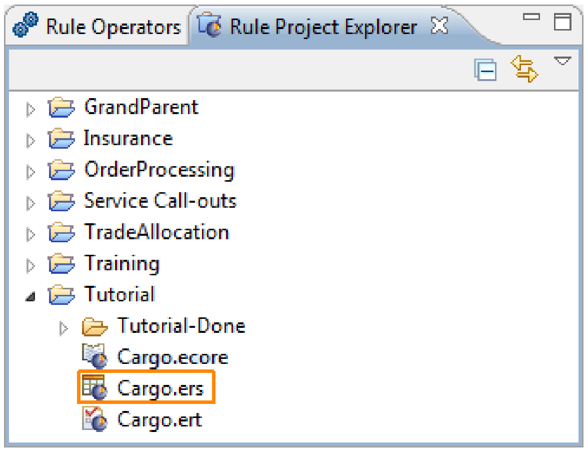
Add new rule statements

To add a new rule statement:

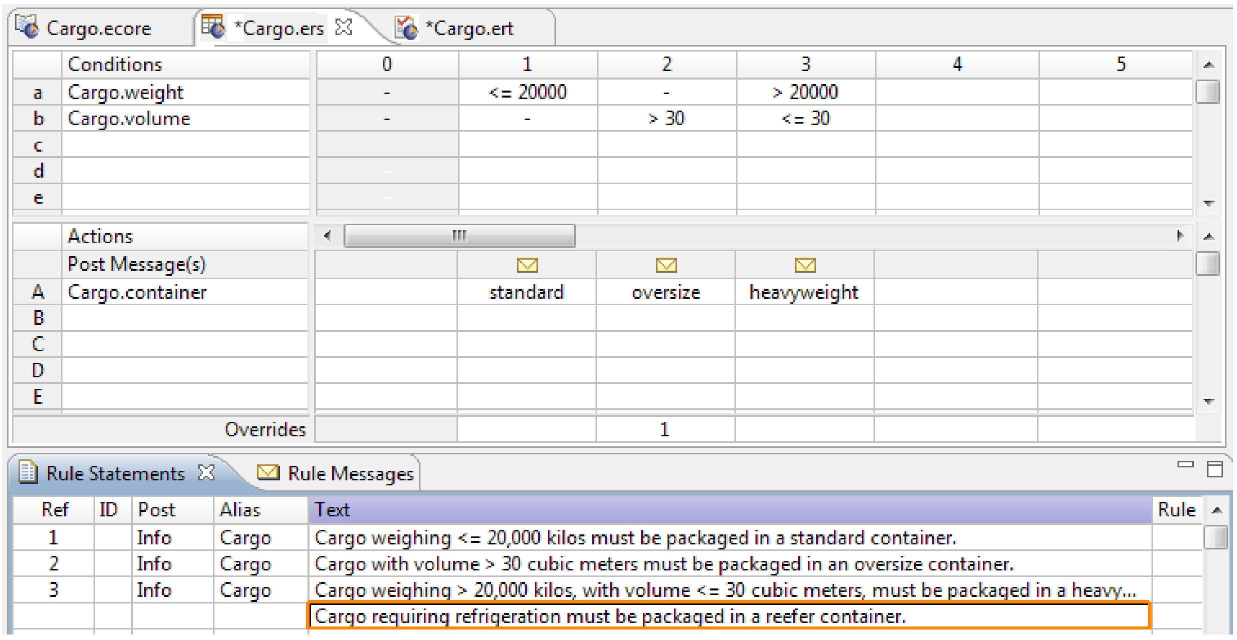
Return to our Rulesheet by clicking the `Cargo.ers` tab:



If you closed the Rulesheet, reopen it from the **Open** menu, or double-click the file inside the **Project Explorer** (or **Rule Project Explorer**) window:



In the rulesheet, enter the new Rule Statement from the previous page. This will guide us when we model the new rule.



	Conditions	0	1	2	3	4	5
a	Cargo.weight	-	<= 20000	-	> 20000		
b	Cargo.volume	-	-	> 30	<= 30		
c							
d							
e							
	Actions						
	Post Message(s)		✉	✉	✉		
A	Cargo.container		standard	oversize	heavyweight		
B							
C							
D							
E							
	Overrides			1			

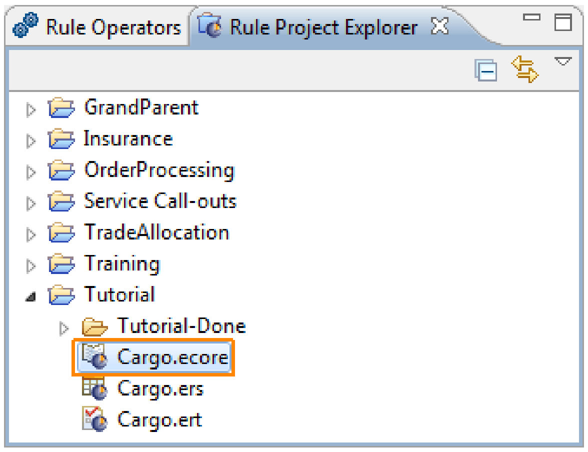
Ref	ID	Post	Alias	Text	Rule
1		Info	Cargo	Cargo weighing <= 20,000 kilos must be packaged in a standard container.	
2		Info	Cargo	Cargo with volume > 30 cubic meters must be packaged in an oversize container.	
3		Info	Cargo	Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be packaged in a heavy...	
				Cargo requiring refrigeration must be packaged in a reefer container.	

Open and edit the Vocabulary

To model this rule, we need to work in the model's Vocabulary, so we'll select the **Cargo.ecore** tab.

If you need to open **Cargo.ecore** into its editor, use any of the following techniques:

- Select **Cargo.ecore** from the **Recent File** list at the bottom of the **File** menu
- Double-click on the file in the **Rule Project Explorer** (or **Project Explorer**) window:

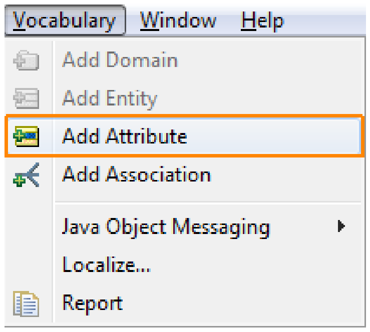


Add an attribute – step 1

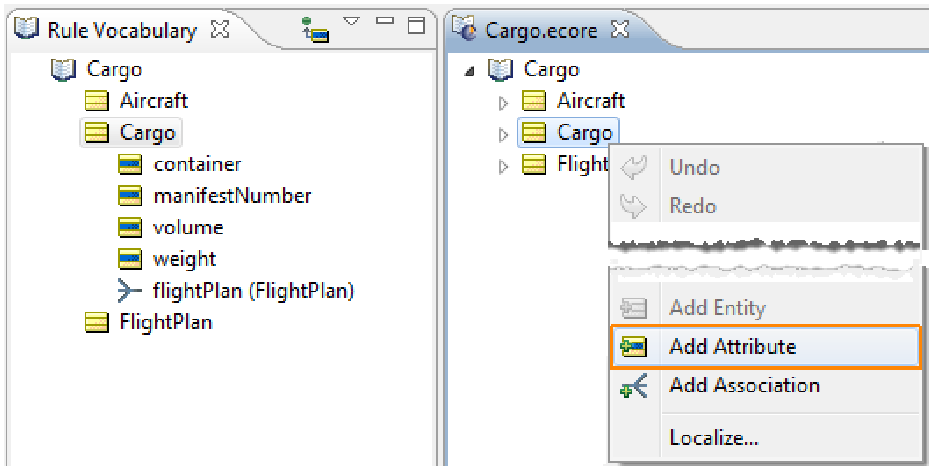
First, we need to add a new attribute to the `Cargo` entity to define whether the Cargo requires refrigeration. We will call this **needsRefrigeration**).

To add a new attribute, either:

- Highlight the `Cargo` entity, and then click the **Add Attribute** button on the toolbar.
- Highlight the `Cargo` entity and choose the menu command **Vocabulary** > **Add Attribute**.



- Within the Vocabulary editor window in the upper right pane (not the Vocabulary window in the upper left pane), right click on the Cargo entity and choose **Add Attribute** on the popup menu.

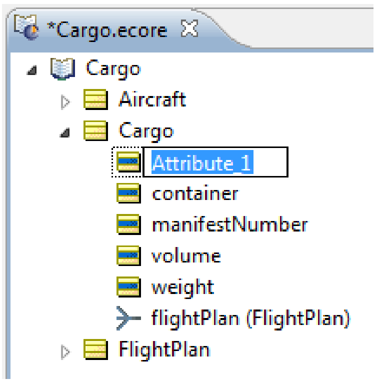


Add an attribute – step 2

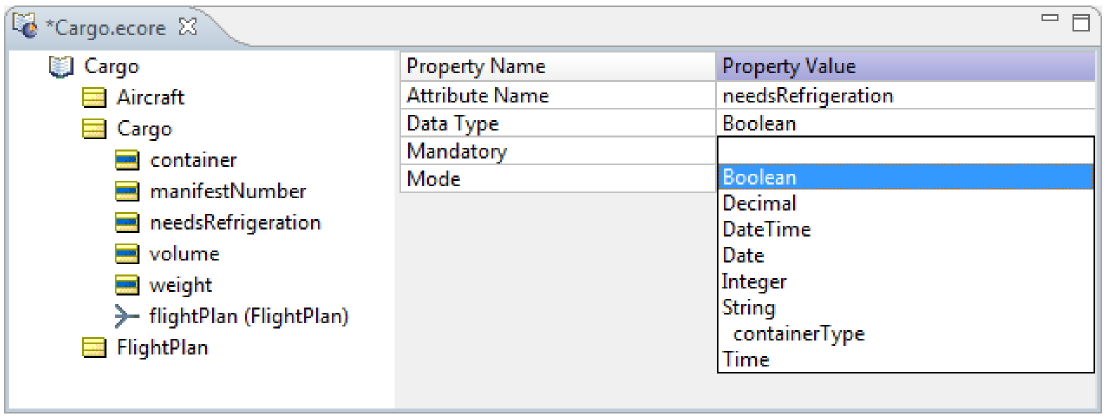
Second, we need to add "reefer" as another possible selection option for the container attribute.

To change an attribute's name and datatype:

1. Double-click on the attribute name to type over its initial default value.



2. Type `needsRefrigeration`.
3. Click on its **Data Type**'s Property Value, then use the pulldown to choose **Boolean** -- it is either true or false.



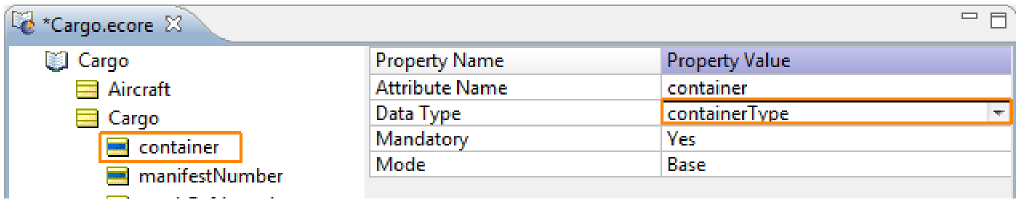
Property Name	Property Value
Attribute Name	needsRefrigeration
Data Type	Boolean
Mandatory	
Mode	

Modify a custom data type

Now we need to add "reefer" as another allowable type of container. It is a *custom datatype*.

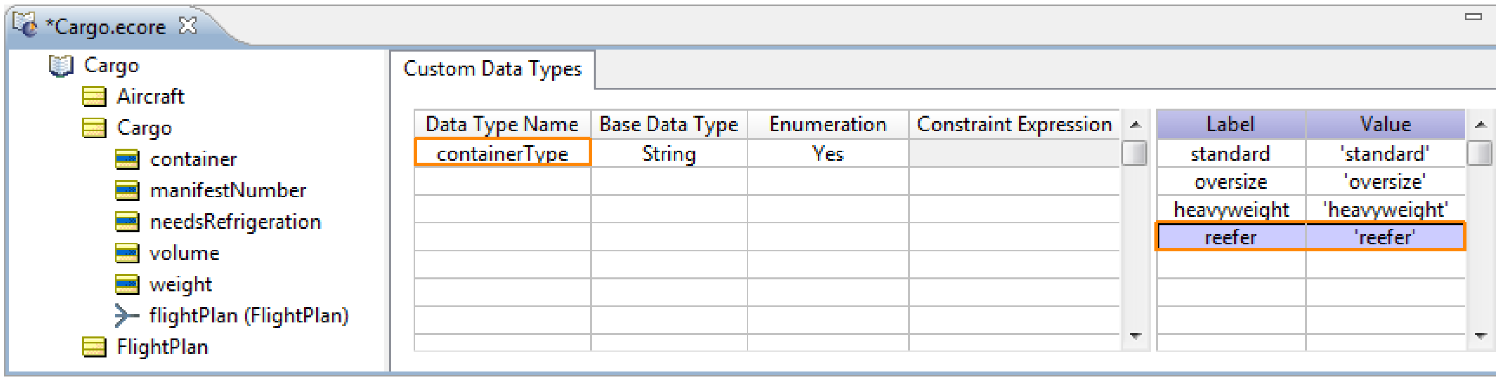
To modify a custom datatype:

1. Select the **container** attribute and note the Data Type: **containerType**. This custom data type defines a set of allowable values for **container** (an *enumerated set*).



Property Name	Property Value
Attribute Name	container
Data Type	containerType
Mandatory	Yes
Mode	Base

2. To define custom data types, click on the **Cargo** in the Vocabulary editor -- the "root" node of the Vocabulary.
3. Click on the **containerType** entry. This will display the enumerated set in the rightmost table, if available.
4. Add `reefer` as both a **Label** and a single-quoted **Value**, as shown:



Data Type Name	Base Data Type	Enumeration	Constraint Expression
containerType	String	Yes	

Label	Value
standard	'standard'
oversize	'oversize'
heavyweight	'heavyweight'
reefer	'reefer'

Custom data types are a powerful capability that let you define rules at the level of the Vocabulary that are reused wherever Vocabulary terms are used. Learn more about custom data types in the *Rule Modeling Guide*.

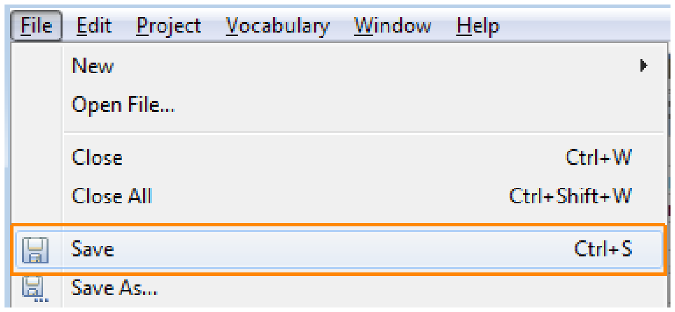
Save the Vocabulary changes

now we can save the Vocabulary and exit the **Vocabulary Editor** pane.

To save the Vocabulary:

Either:

- Click the **Save** button on the Corticon Studio toolbar.
- Choose the menu command **File > Save**, as shown:



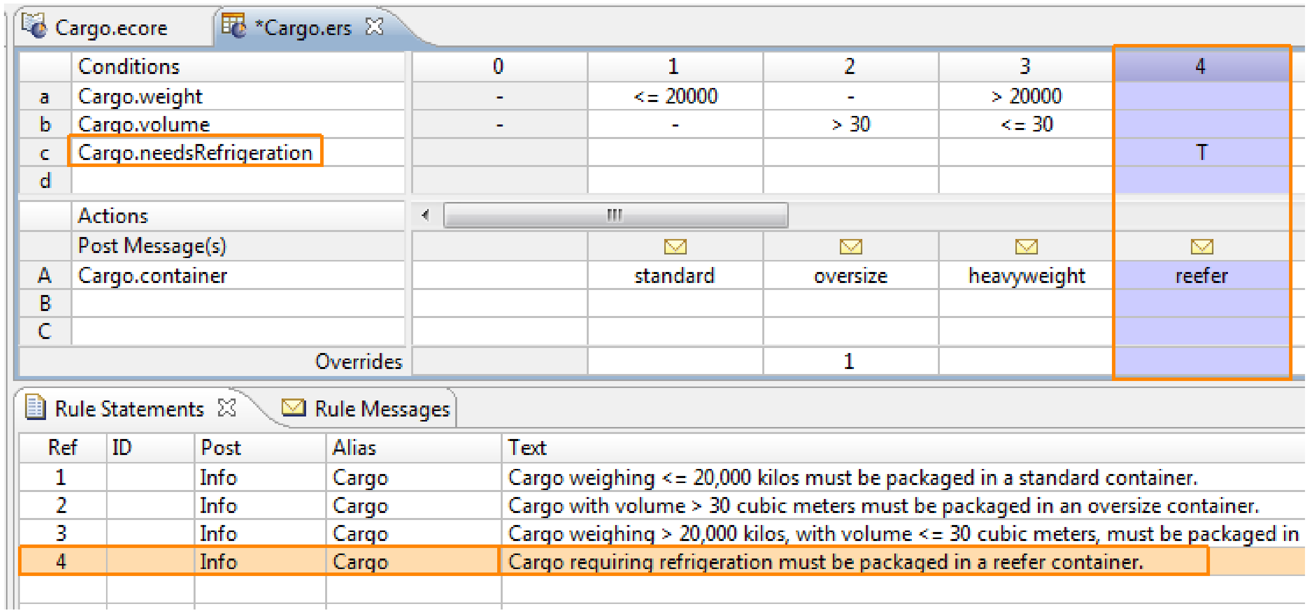
Model the new rule

Now that the Vocabulary contains the new terms required by the new rule, complete the model as shown below.

To model the new rule:

1. Drag `needsRefrigeration` onto Conditions line c.
2. Click in **Conditions** line c, column 4 to choose `T` from its menu.
3. Click in **Actions** line A, column 4 to choose `reefer` from its menu.
4. Add **Reference** line 4, then:
   a. In the **Post** column, select `Info`.
   b. In the **Alias** column, select `Cargo`.
   c. In the **Text** column, Type `Cargo requiring refrigeration must be packaged in a reefer container.`

Here is how it looks:



	Conditions	0	1	2	3	4
a	Cargo.weight	-	<= 20000	-	> 20000	
b	Cargo.volume	-	-	> 30	<= 30	
c	Cargo.needsRefrigeration					T
d						
	Actions					
	Post Message(s)					
A	Cargo.container		standard	oversize	heavyweight	reefer
B						
C						
	Overrides			1		

Ref	ID	Post	Alias	Text
1		Info	Cargo	Cargo weighing <= 20,000 kilos must be packaged in a standard container.
2		Info	Cargo	Cargo with volume > 30 cubic meters must be packaged in an oversize container.
3		Info	Cargo	Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be packaged in
4		Info	Cargo	Cargo requiring refrigeration must be packaged in a reefer container.

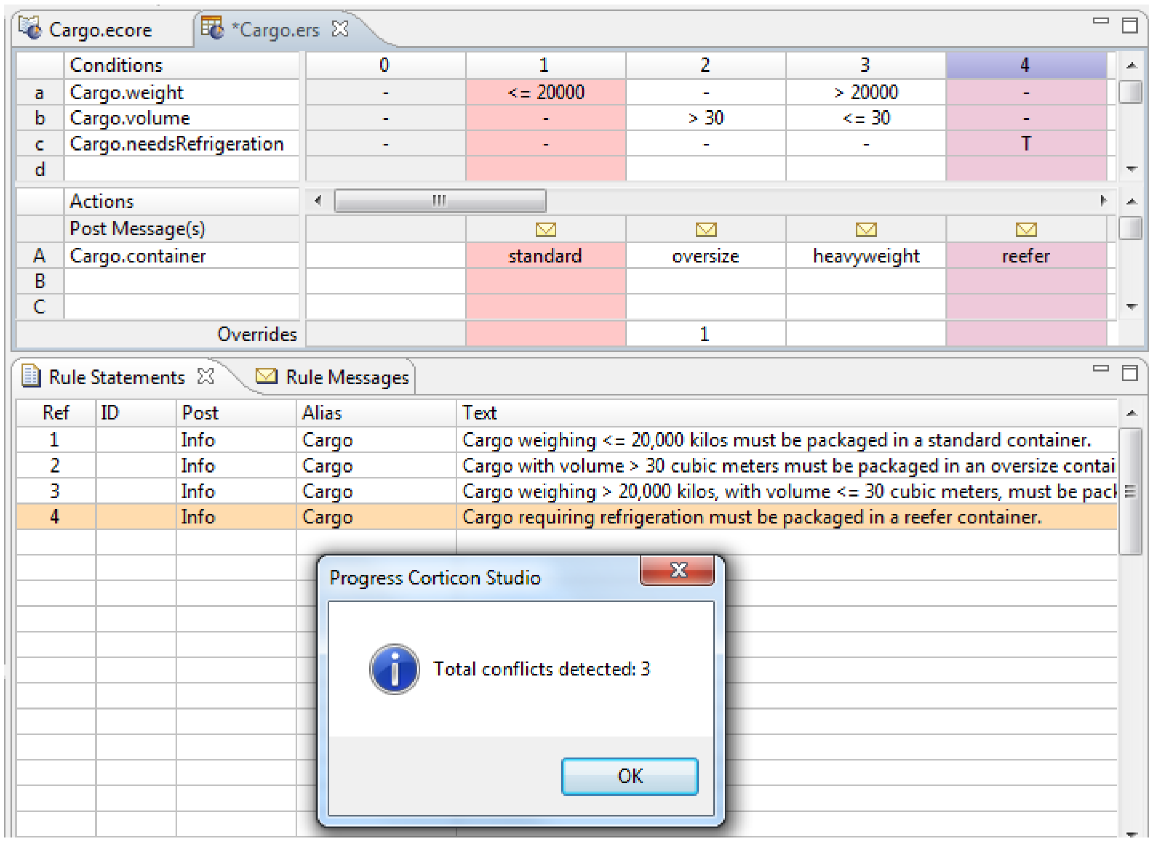
The rule is complete!

Check new rules for conflicts

Now we'll check for logical errors in the Rulesheet.

(To review this procedure, refer to Check for conflicts that we did earlier.)

Rule analysis is an iterative process. Whenever we add or change our rules, it is necessary to re-analyze. In this case, adding a single new rule has caused three new conflicts, one with each of our three existing rules.



	Conditions	0	1	2	3	4
a	Cargo.weight	-	<= 20000	-	> 20000	-
b	Cargo.volume	-	-	> 30	<= 30	-
c	Cargo.needsRefrigeration	-	-	-	-	T
d						
	Actions					
	Post Message(s)		✉	✉	✉	✉
A	Cargo.container		standard	oversize	heavyweight	reefer
B						
C						
	Overrides			1		

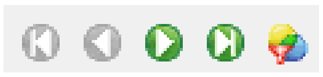
Ref	ID	Post	Alias	Text
1		Info	Cargo	Cargo weighing <= 20,000 kilos must be packaged in a standard container.
2		Info	Cargo	Cargo with volume > 30 cubic meters must be packaged in an oversize contai
3		Info	Cargo	Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be pack
4		Info	Cargo	Cargo requiring refrigeration must be packaged in a reefer container.

Progress Corticon Studio

Total conflicts detected: 3

OK

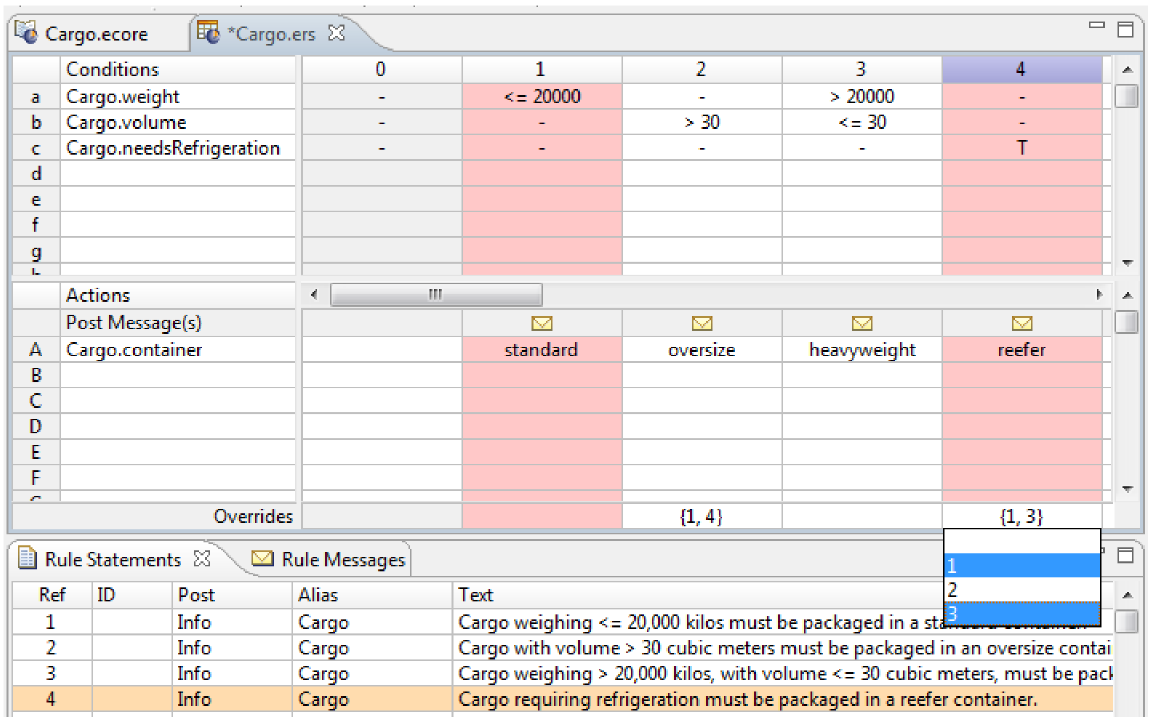
You can step through multiple conflicts, and filter the Rulesheet to view only the conflicting rules, by using the buttons to the right of the **Conflict Check** button:



Setting overrides

Reefer containers are only available for standard and heavyweight, but not for oversize loads. Thus, let's set rule 4 to override rules 1 & 3, and rule 2 to override rules 1 & 4. You can select multiple values in the override cell by holding down the **Ctrl** key while you make your selections.

When complete, your Rulesheet should look like this:



Cargo.ecore | *Cargo.ers

	Conditions	0	1	2	3	4
a	Cargo.weight	-	<= 20000	-	> 20000	-
b	Cargo.volume	-	-	> 30	<= 30	-
c	Cargo.needsRefrigeration	-	-	-	-	T
d						
e						
f						
g						

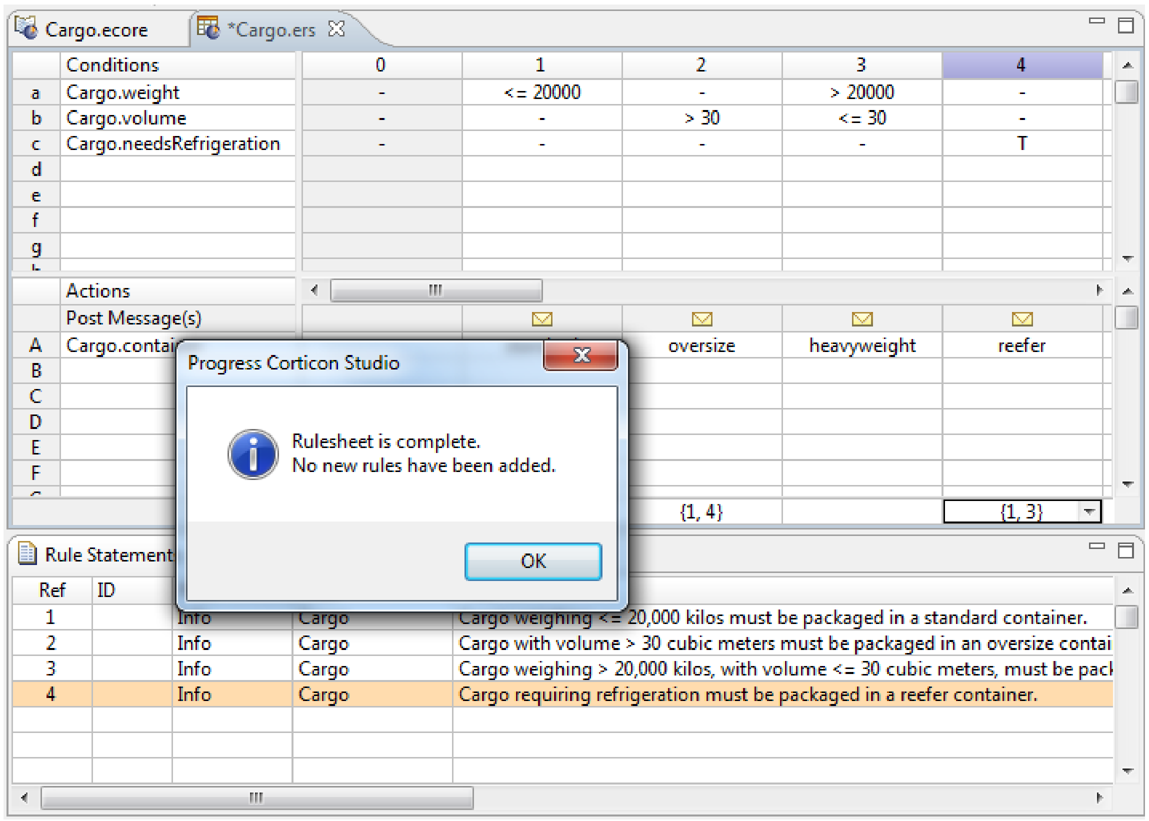
	Actions					
	Post Message(s)		✉	✉	✉	✉
A	Cargo.container		standard	oversize	heavyweight	reefer
B						
C						
D						
E						
F						
	Overrides			{1, 4}		{1, 3}

Rule Statements | Rule Messages

Ref	ID	Post	Alias	Text
1		Info	Cargo	Cargo weighing <= 20,000 kilos must be packaged in a sta
2		Info	Cargo	Cargo with volume > 30 cubic meters must be packaged in an oversize contai
3		Info	Cargo	Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be pack
4		Info	Cargo	Cargo requiring refrigeration must be packaged in a reefer container.

Checking a new rule for completeness

Click the **Check for Completeness** button again



Cargo.ecore | *Cargo.ers

	Conditions	0	1	2	3	4
a	Cargo.weight	-	<= 20000	-	> 20000	-
b	Cargo.volume	-	-	> 30	<= 30	-
c	Cargo.needsRefrigeration	-	-	-	-	T
d						
e						
f						
g						

	Actions					
	Post Message(s)					
A	Cargo.contai			oversize	heavyweight	reefer
B						
C						
D						
E						
F						
				{1, 4}		{1, 3}

Progress Corticon Studio

Rulesheet is complete.
No new rules have been added.

OK

Rule Statement

Ref	ID			
1		Info	Cargo	Cargo weighing <= 20,000 kilos must be packaged in a standard container.
2		Info	Cargo	Cargo with volume > 30 cubic meters must be packaged in an oversize contai
3		Info	Cargo	Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be pack
4		Info	Cargo	Cargo requiring refrigeration must be packaged in a reefer container.

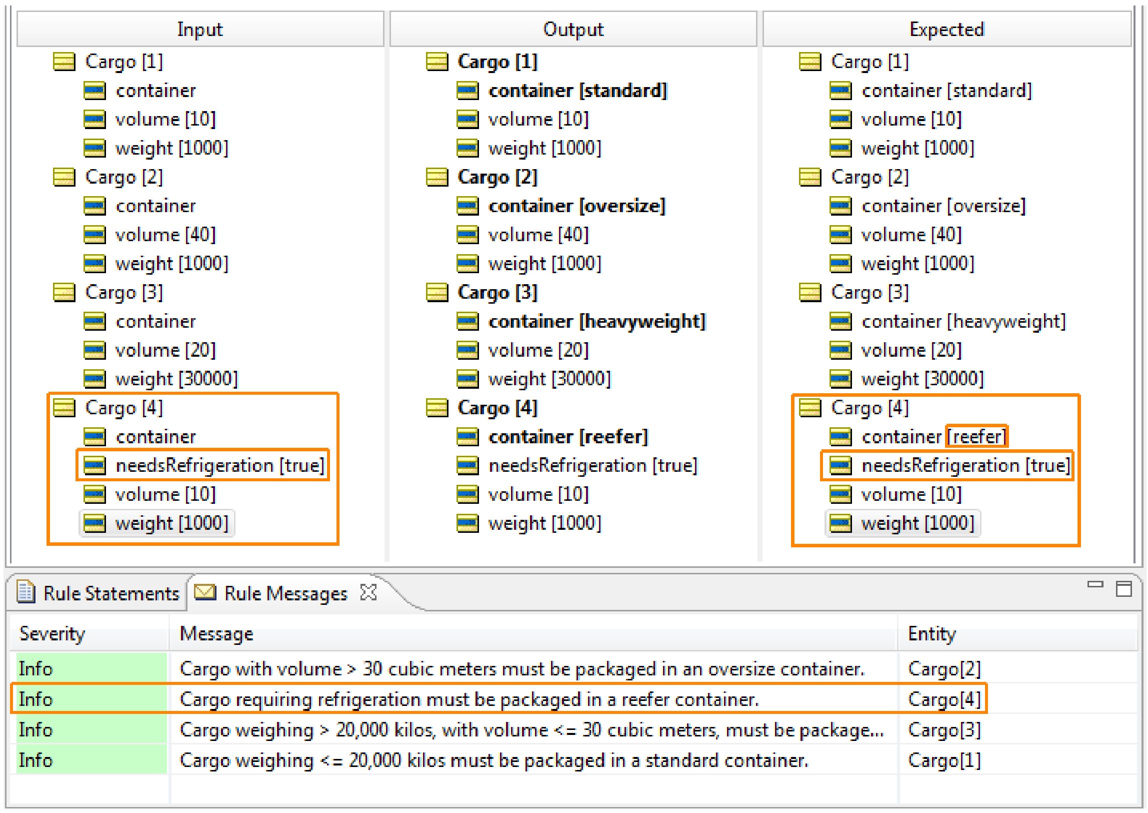
Save the Rulesheet.

Modify the Ruletest

Your last step is to modify your test case to test the new rule.

1. Save your rules.
2. In Ruletest `Cargo.ert`, copy and paste `Cargo[1]` to create `Cargo[4]`, in both the Input and Expected panes.
3. Drag and drop the **needsRefrigeration** attribute from your Vocabulary onto both `Cargo[4]` entities.
4. In both panes, set the value of **needsRefrigeration** to **true**.
5. In only the Expected pane, change the value of container to **reefer**.
6. Run the test.

Your Ruletest should look like this:



Input	Output	Expected
Cargo [1]	**Cargo [1]**	Cargo [1]
container	**container [standard]**	container [standard]
volume [10]	volume [10]	volume [10]
weight [1000]	weight [1000]	weight [1000]
Cargo [2]	**Cargo [2]**	Cargo [2]
container	**container [oversize]**	container [oversize]
volume [40]	volume [40]	volume [40]
weight [1000]	weight [1000]	weight [1000]
Cargo [3]	**Cargo [3]**	Cargo [3]
container	**container [heavyweight]**	container [heavyweight]
volume [20]	volume [20]	volume [20]
weight [30000]	weight [30000]	weight [30000]
Cargo [4]	**Cargo [4]**	Cargo [4]
container	**container [reefer]**	container [reefer]
needsRefrigeration [true]	needsRefrigeration [true]	needsRefrigeration [true]
volume [10]	volume [10]	volume [10]
weight [1000]	weight [1000]	weight [1000]

Severity	Message	Entity
Info	Cargo with volume > 30 cubic meters must be packaged in an oversize container.	Cargo[2]
Info	Cargo requiring refrigeration must be packaged in a reefer container.	Cargo[4]
Info	Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be package...	Cargo[3]
Info	Cargo weighing <= 20,000 kilos must be packaged in a standard container.	Cargo[1]

Summary

Congratulations! We have now completed two full iterations of the Corticon development lifecycle, and have learned the basic concepts of rule modeling, analysis, and testing in Progress Corticon.

To learn advanced rules modeling techniques, see the *Corticon Studio Tutorial: Advanced Rule Modeling* .

To learn how to deploy rules, see the *Corticon Server: Deploying Web Services* tutorials.

To extend this tutorial experience and see how the Cargo assets we created are transformed into rules that interact with a relational database, work through the tutorial *Using the Enterprise Data Connector*.

A

Third party acknowledgments