

PROGRESS[®] CORTICON[®]

Corticon Studio:
Quick Reference Guide

Notices

Copyright agreement

© 2013 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Apama, Business Empowerment, Business Making Progress, Corticon, Corticon (and design), DataDirect (and design), DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, Empowerment Center, Fathom, Making Software Work Together, OpenEdge, Powered by Progress, PowerTier, Progress, Progress Control Tower, Progress Dynamics, Progress Business Empowerment, Progress Empowerment Center, Progress Empowerment Program, Progress OpenEdge, Progress Profiles, Progress Results, Progress RPM, Progress Software Business Making Progress, Progress Software Developers Network, ProVision, PS Select, RulesCloud, RulesWorld, SequeLink, SpeedScript, Stylus Studio, Technical Empowerment, WebSpeed, Xcalia (and design), and Your Software, Our Technology—Experience the Connection are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, Apama Dashboard Studio, Apama Event Manager, Apama Event Modeler, Apama Event Store, Apama Risk Firewall, AppsAlive, AppServer, BusinessEdge, Cache-Forward, DataDirect Spy, DataDirect SupportLink, Future Proof, High Performance Integration, OpenAccess, ProDataSet, Progress Arcade, Progress ESP Event Manager, Progress ESP Event Modeler, Progress Event Engine, Progress RFID, Progress Responsive Process Management, Progress Software, PSE Pro, SectorAlliance, SeeThinkAct, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

See Table of Contents for location of Third party acknowledgements within this documentation.

Table of Contents

Preface.....	9
Progress Corticon documentation.....	9
Overview of Progress Corticon.....	11
 Chapter 1: Using Progress Corticon Studio.....	13
Progress Corticon Studio components.....	13
Initial menu commands.....	14
Initial toolbar.....	17
The file tab.....	18
The active Corticon Studio window.....	19
File naming restrictions.....	20
 Chapter 2: Rule Projects.....	21
Creating a Rule Project.....	21
The Rule Project explorer window.....	24
 Chapter 3: Vocabularies.....	25
Creating a Vocabulary.....	25
The Vocabulary window.....	26
Vocabulary menu commands.....	27
Vocabulary toolbar.....	27
Context-sensitive right-click pop-up menu.....	28
Populating a new Vocabulary.....	29
The Vocabulary tree view.....	29
Vocabulary properties.....	29
Vocabulary node naming restrictions.....	29
Adding nodes to the Vocabulary tree view.....	30
Domain nodes: Adding and editing domains and their properties.....	30
Entity nodes: Adding and editing entities and their properties.....	31
Attribute nodes: Adding and editing attributes and their properties.....	35
Enumerated values.....	38
Association Nodes: Adding and editing associations and their properties.....	39
Editing an association.....	43
Renaming a vocabulary node.....	43
Saving a new Vocabulary.....	43
Opening an existing Vocabulary.....	44
Modifying a Vocabulary.....	45

Creating a Vocabulary report.....	45
-----------------------------------	----

Chapter 4: Rulesheets.....47

Creating a new Rulesheet.....	48
Rulesheet menu commands.....	49
Rulesheet toolbar.....	51
Context-sensitive right-click pop-up menus.....	52
Rulesheet sections.....	55
Rule statements window.....	57
Using the business vocabulary to build rules.....	59
Using the operator vocabulary to build rules.....	60
Naming Rulesheets.....	61
Deleting Rulesheets.....	61
Saving a new Rulesheet.....	61
Validating and optimizing a Rulesheet.....	62
Creating a Rulesheet report.....	62
Closing a Rulesheet.....	62
Saving a modified Rulesheet.....	63

Chapter 5: Ruleflows.....65

Ruleflow window.....	66
Creating a new Ruleflow.....	66
Naming Ruleflows.....	67
Adding Ruleflows.....	67
Deleting Ruleflows.....	67
Saving a new Ruleflow.....	68
Ruleflow menu commands.....	68
Ruleflow toolbar.....	68
Context-sensitive right-click pop-up menus in Ruleflows.....	69
Ruleflow window.....	72
Iteration.....	73
Enabling and disabling iteration for Rulesheets.....	73
Enabling and disabling iteration for Subflows.....	73
Ruleflow tools palette.....	74
Renaming a Ruleflow and/or saving a Ruleflow to a different location.....	75
Creating a Ruleflow report.....	75
Closing a Ruleflow.....	76
Ruleflow properties tab.....	76
Ruleflow sub-tab.....	76
Comments sub-tab.....	78
Rulers & grid sub-tab.....	78

Chapter 6: Ruletests.....79

Ruletest window.....	79
Creating a new Ruletest.....	80
Ruletest menu commands.....	82
Ruletest toolbar.....	84
Context-sensitive right-click pop-up menus.....	85
Testsheet tabs.....	87
Populating the input panel.....	87
Adding entities to the test tree.....	87
Creating associations in the test tree.....	88
Assigning attribute values in the test tree.....	89
Automatically generating a test tree.....	89
Executing tests.....	90
Sequence of message posting.....	91
Sorting messages.....	91
Using the expected panel.....	91
Expected panel – output results match expected exactly.....	91
Expected panel – different values output than expected.....	92
Expected panel – fewer values output than expected.....	92
Expected panel - more values output than expected.....	93
Expected panel – all problems.....	93
Format of output data.....	94
Creating multiple test scenarios on the same testsheet.....	94
Creating multiple test scenarios as a set of testsheets.....	95
Creating a sequential test using multiple testsheets.....	95
Naming testsheets.....	96
Adding testsheets.....	97
Associating one child entity with more than one parent.....	97
Saving ruletests.....	98
Importing an XML or SOAP document to a testsheet.....	98
Exporting a testsheet to an XML document.....	99
Exporting a testsheet to a SOAP message.....	99
Creating a Ruletest report.....	100
 Chapter 7: Exiting Corticon Studio.....	 101
 Appendix A: Keyboard shortcuts	 103
 Appendix B: Third party acknowledgments	 105

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	
<i>Corticon Studio Tutorial: Basic Rule Modeling</i>	Introduces modeling, analyzing, and testing rules and decisions in Corticon Studio. Recommended for evaluators and users getting started. <i>See also the PowerPoint-as-PDF version of this document that is accessed from the Studio for Analysts' Help menu.</i>
<i>Corticon Studio Tutorial: Advanced Rule Modeling</i>	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. <i>See also the PowerPoint-as-PDF version of this document that is accessed from the Studio for Analysts' Help menu.</i>
<i>Corticon Tutorial: Using Enterprise Data Connector (EDC)</i>	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	
<i>Corticon Studio: Installation Guide</i>	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.
<i>Corticon Studio: Rule Modeling Guide</i>	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 <i>Test Yourself</i> exercises.
<i>Corticon Studio: Quick Reference Guide</i>	Reference guide to the Corticon Studio user interface and its mechanics, including descriptions of all menu options, buttons, and actions.
<i>Corticon Studio: Rule Language Guide</i>	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.
<i>Corticon Studio: Extensions Guide</i>	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	
<i>Corticon Server: Deploying Web Services with Java</i>	Details installing the Corticon Server as a Web Services Server, and then and 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.
<i>Corticon Server: Deploying Web Services with .NET</i>	Details installing the Corticon Server as a Web Services Server and 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.
<i>Corticon Server: Integration & Deployment Guide</i>	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 3.7.1 and Java 7 development environment. The PDS enables integrated applications with other products such as Progress OpenEdge and Progress Apama.

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 4.0 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.

Using Progress Corticon Studio

For details, see the following topics:

- [Progress Corticon Studio components](#)
- [Initial menu commands](#)
- [Initial toolbar](#)
- [The file tab](#)
- [The active Corticon Studio window](#)
- [File naming restrictions](#)

Progress Corticon Studio components

Corticon Studio files consist of four major types:

1. **Vocabulary:** a structured dictionary containing all necessary business terms and relationships between them used by the business rules.
2. **Rulesheet:** a set of conditions and actions and plain language statements written from a common business Vocabulary.
3. **Ruleflow:** a set of one or more *Rulesheets* organized for sequential execution. Once a *Ruleflow* has been saved and deployed to the *Corticon Server*, it is called a Decision Service.
4. **Ruletest:** a set of one or more Testsheets containing sample data used for testing *Rulesheets* and *Ruleflows*. Like *Rulesheets*, *Ruletests* also use a common Vocabulary model.

A **Rule Project** may contain any number of *Vocabularies*, *Rulesheets*, *Ruleflows* or *Ruletests*.

Following construction and testing within Corticon Studio, a *Rulesheet* must be saved before it can be tested by a *Ruletest*. *Ruleflows* must be saved before they can be deployed to *Corticon Server*. Deployment and integration is covered in greater detail in the *Server Integration & Deployment Guide*. A quick introduction to this topic is provided in the *Server Deployment Tutorial*.

File Suffixes

The four types of files listed above have the following file types or file suffixes:

- *.ecore* — the *Vocabulary*.
- *.ers* — the *Rulesheet*. Multiple *Rulesheets* may use the same Vocabulary, but each *Rulesheet* has only one associated Vocabulary.
- *.erf* — the *Ruleflow*. One *Ruleflow* may contain multiple *Rulesheets*, each of which has only one associated Vocabulary.
- *.ert* — the *Ruletest*. A *Ruletest* only) or a *Ruleflow*.

Initial menu commands

The following actions are available when Corticon Studio first opens and no files are open (a command is grayed out when no files are open):

Many actions are standard behaviors in the Eclipse development environment. See the Eclipse Help's *Workbench User Guide* for details on standard functionality.

File Menu

The actions accessed on the File menu are:

- **New** - Creates the specified Corticon Studio resource.
- **New > Rule Project** - Creates a new Rule Project.
- **New > Rule Vocabulary** - Creates a new Rule Vocabulary file (*.ecore*).
- **New > Ruleflow** - Creates a new Ruleflow file (*.erf*).
- **New > Rulesheet** - Creates a new Rulesheet file (*.ers*).
- **New > Ruletest** - Creates a new Ruletest file (*.ert*).
- **New > Folder** - Creates a new folder.

The other File menu options perform standard Eclipse functions.

Edit Menu

The actions accessed on the Edit menu are standard Eclipse functions with the following notes and additions:

- **Select All** - Applies only to *Rulesheets* and *Ruleflow*s. See [Ruleflow](#) section.
- **Find/Replace** - Not enabled in the Corticon Designer.
- **Add Task** - Not enabled in the Corticon Designer.
- **Insert Row** - Inserts a new Row into the active file.
- **Remove Row** - Removes the selected Row from the active file.
- **Add Rows to End** - Inserts 10 Rows at the end of the active file.
- **Insert Column** - Inserts a new Column into the active file.
- **Remove Column** - Removes the selected Column(s) from the active file.
- **Add Columns to End** - Inserts 10 Columns at the end of the active file.
- **Move Up** - Moves a Custom Data Type (CDT) enumerated Label/Value row up in the list. See the *Rule Modeling Guide*'s "Building the Vocabulary" chapter for more info on CDTs.
- **Move Down** - Moves a Custom Data Type (CDT) enumerated Label/Value row down in the list.
- **Enable / Disable** - Toggles enabling and disabling of a column, row, cell or shape. (This feature lets the rule builder experiment with temporarily removing logic without removing its information from the work.) The hierarchy of granularity is as follows:
 1. Rulesheet columns and rows
 2. Individual THEN cells (these are the cells in the lower-right quadrant of the Rulesheet)
 3. Rule Statements
 4. Shapes on the Ruleflow

Navigate Menu

The actions accessed on the Navigate menu are standard Eclipse functions.

Search Menu

The actions accessed on the Search menu are standard Eclipse functions.

Project Menu

The actions accessed on the Project menu are standard Eclipse functions.

Run Menu

The actions accessed on the Run menu are standard Eclipse functions.

Vocabulary, Rulesheet, Ruleflow, Ruletest Menus

Note: When a Corticon file is active, the corresponding menu is added to the toolbar. Each of these is described in later sections of this guide, as follows:

- [Vocabulary menu commands](#) on page 27
- [Rulesheet menu commands](#) on page 49
- [Ruleflow menu commands](#) on page 68
- [Ruletest menu commands](#) on page 82

Window Menu

The actions accessed on the Window menu are standard Eclipse functions with the following notes and additions:

- **Open Perspective** - Opens a defined set of views and editors. If already open, returns to that instance.
- **Open Perspective > Corticon Designer** - Resets the layout to the views commonly used in Corticon modeling.
- **Open Perspective > Other** - Opens the perspective dialog where you can choose an available perspective.
- **Show View** - Adds a perspective-related view as a tab in the perspective window. Views in the Corticon Designer include **Error Log**, **Localization**, **Natural Language**, **Problems**, **Properties**, **Rule Message**, **Rule Operations**, **Rule Project Explorer**, **Rule Statements**, and **Rule Vocabulary**. **Other** lists all available views, whether or not directly relevant in the current perspective.
- **Preferences** - Opens the Preferences dialog for the installation. Corticon preferences include setting the user role, setting the path to your license, and a variety of settings for the appearance of View Ruleflow diagrams. The user role is typically **Rule Modeling**. When changed to **Integration & Deployment**, additional functionality is added that enables you to map Vocabularies to external data such as XML and Java business objects.

Help Menu

The actions accessed on the Help menu are standard Eclipse functions with the following notes and additions:

- **Welcome** - Opens the Welcome page with access to What's New, Samples, Tutorials, Documentation, and Web Resources.
- **Help Contents** - Opens the help system where you can search for terms and browse the help structure of Eclipse, third-party, and Progress help, including Corticon-specific rendering of the documentation in the help format.
- **Dynamic Help Contents** - Opens context-sensitive Corticon help plus access to related topics.
- **Check for Updates** - Accesses Progress and related sites to determine whether updates are available.
- **About...** - Details versions and licenses of installed Progress and third-party products. Also provides access to installation details.

Initial toolbar

The initial Corticon Studio toolbar displays the basic tools that you will need to create a new Vocabulary, *Rulesheet*, *Ruleflow*, or *Ruletest* files, or open existing files.

At this point, many of the menubar options and toolbar buttons have limited functionality, but they will be described here because they are options which are commonly available to all open files.



The icons on the toolbar initiate the following actions:

-  (Opens dropdown **New**menu) Creates the specified Corticon *resource* (also referred to as an *asset*).
- **New >**  Creates a new Rule Project.
- **New >**  Creates a new Rule Vocabulary file (*.ecore*).
- **New >**  Creates a new Ruleflow file (*.erf*).
- **New >**  Creates a new Rulesheet file (*.ers*).
- **New >**  Creates a new Ruletest file (*.ert*).
- **New >**  Creates a new folder.
-  Saves changes to the active file.
-  Prints the active file.
-  Disables the selection.
-  Moves a Custom Data Type (CDT) enumerated Label/Value row up in the list. See the *Rule Modeling Guide's* "Building the Vocabulary" chapter for more info on CDTs.
-  Moves a Custom Data Type (CDT) enumerated Label/Value row down in the list.

-  Inserts a new Column into the active file.
-  Removes the selected Column(s) from the active file.
-  Inserts 10 Columns at the end of the active file.
-  Inserts a new Row into the active file.
-  Removes the selected Row from the active file.
-  Inserts 10 Rows at the end of the active file.

Note: Vocabulary, Rulesheet, Ruleflow, Ruletest Icons

Note: When a Corticon file is active, the corresponding menu is added to the toolbar. Each of these is described in later sections of this guide, as follows:

- [Vocabulary toolbar](#) on page 27
 - [Rulesheet toolbar](#) on page 51
 - [Ruleflow toolbar](#) on page 68
 - [Ruletest toolbar](#) on page 84
-

The file tab

When you create a new or open an existing Vocabulary, *Rulesheet*, *Ruleflow*, or *Ruletest*, a tab is displayed at the top of each window. Multiple tabs will be arranged horizontally underneath the Corticon Studio toolbar when multiple files are open at once.

Shown is the name of the window and the window's controls.



1. File type and name
2. Minimize the file's window
3. Maximize file's window
4. Close the file's window

The active Corticon Studio window

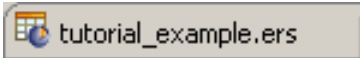
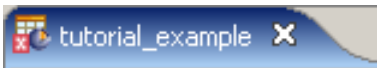
Throughout this manual, reference will be made to the "active" Corticon Studio window. Any single open window in Corticon Studio may be active at a given time. Indication of active status is provided by the window's tab color. As shown below, the active window's tab is colored **blue**, while inactive tabs are colored **gray**. "Shifting focus" or, in other words, activating a different window, is accomplished simply by clicking anywhere within an inactive window or on the window's tab.

Below, the tabs of two Corticon Studio windows, `tutorial_example.erf` (a *Ruleflow*) and `Untitled.ers` (a *Rulesheet*) are shown tiled horizontally. `tutorial_example.erf` is the active window because its tab is **blue**.



Window Tabs

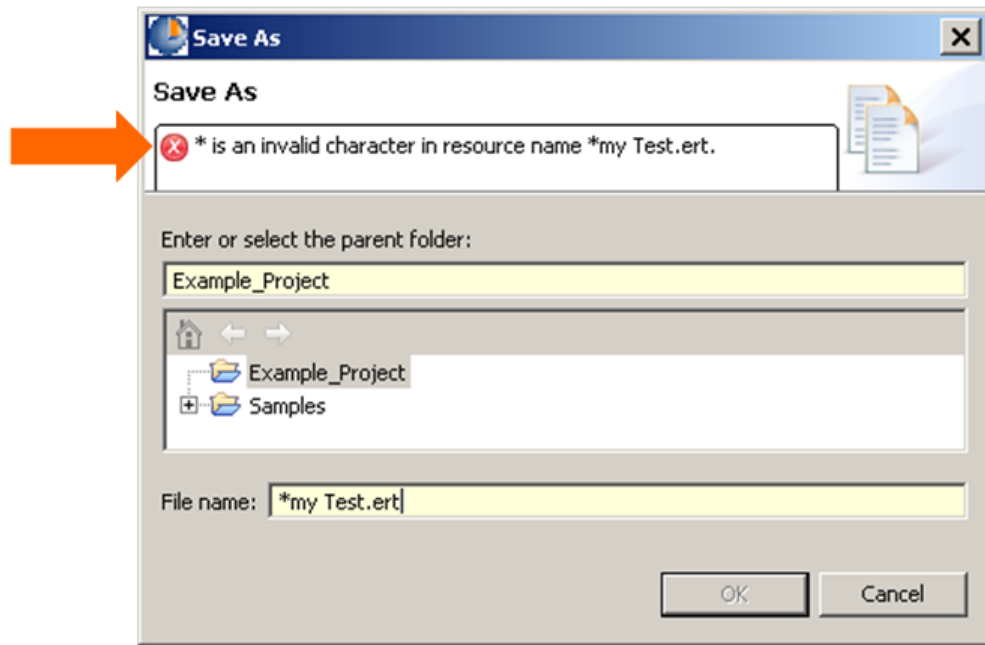
Each window tab contains information about the window's status:

	This window is the active Corticon Studio file, and its save status is up-to-date
	This window is not active. Clicking on it will cause it to become active
	This window is the active Corticon Studio file, but recent changes have not yet been saved. As soon as the file is saved, the asterisk before the window's tab name will disappear.
	The small red square overlaying the window's file type icon indicates there is an error somewhere in the window. Look in the Problems window for more information (Window > Show View > Problems in the Corticon Studio menubar)
	The small yellow triangle overlaying the window's file type icon indicates there is a warning somewhere in the window. Look in the Problems window for more information (Window > Show View > Problems in the Corticon Studio menubar)

File naming restrictions

File names must comply with the following rules:

- File names may contain or start with almost any alphanumeric character and most special characters - the **Save** or **Save As** dialog will inform you if you choose a prohibited character, as shown below.
- File names may not begin with a space, but may contain spaces in subsequent character positions.



Rule Projects

In Corticon Studio, a Vocabulary, as well as any *Ruleflows*, *Rulesheets* and *Ruletests* associated with that Vocabulary, must be stored in a Rule Project. By default, Corticon Studio provides you with a new directory, called *workspace*, where you store your Projects.

Although you can create Rule Project folders linked to directories anywhere on your local file system and view them in Corticon Studio, we recommend that you use the default path provided: `[CORTICON_WORK_DIR]\workspace`. When you upgrade to later versions of Progress Corticon Studio, your Rule Projects will remain and survive the upgrade process intact. Other directory locations in the file system may need to be re-linked to a Rule Project when new versions of Progress Corticon Studio are installed.

For details, see the following topics:

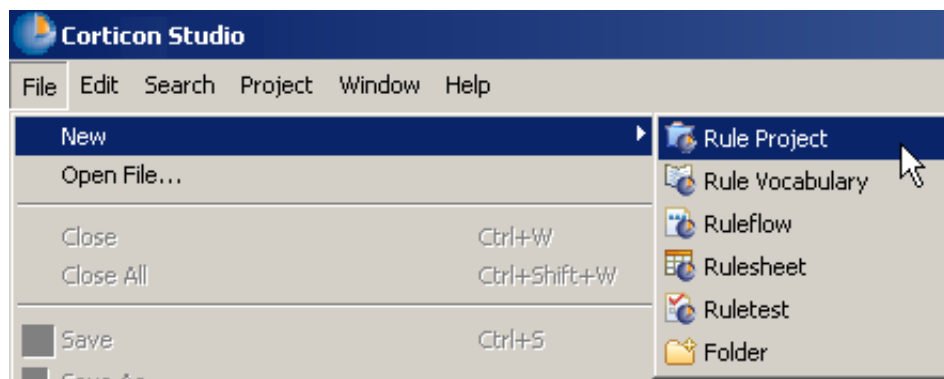
- [Creating a Rule Project](#)

Creating a Rule Project

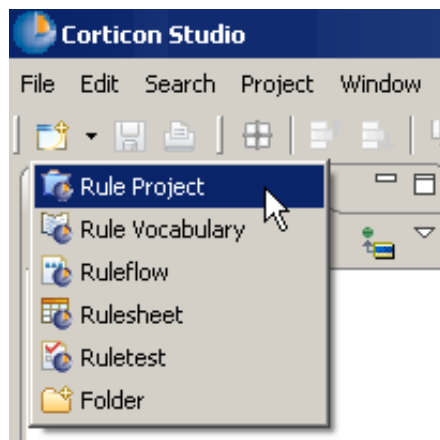
Follow these steps to create a new Rule Project:

Do one of the following:

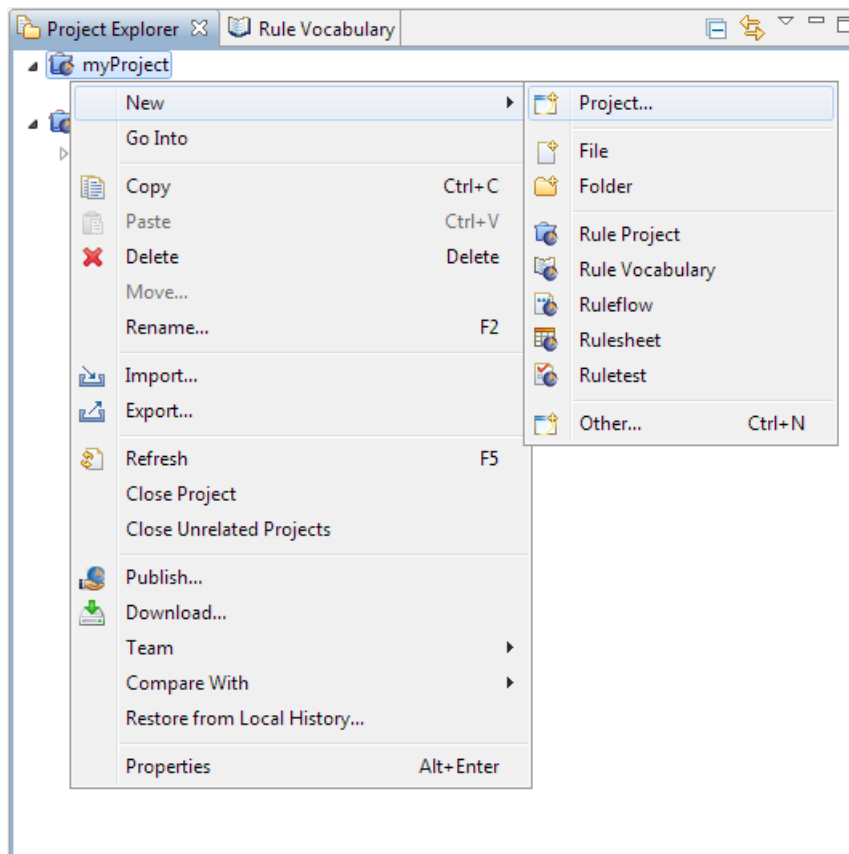
- Select **File > New > Rule Project** from the menubar



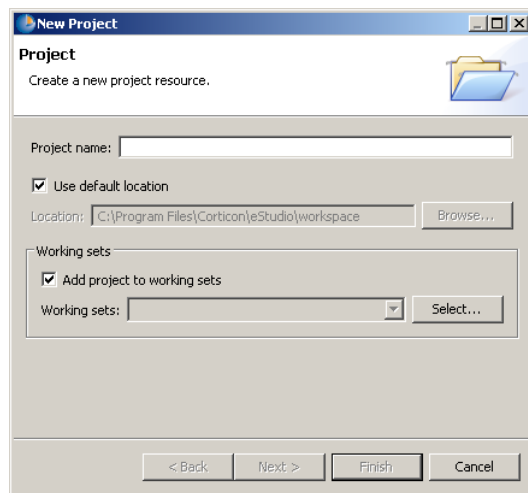
- Click the down arrow to the right of the **New** icon  on the toolbar and select **Rule Project**.



- Right-click in the **Project Explorer** to open its menu, and then choose **New > Project**.



These techniques all launch the same **New Project** wizard, as shown:



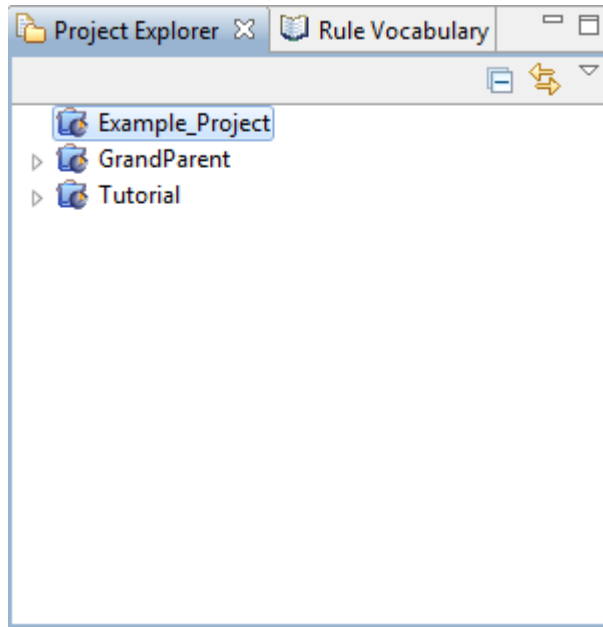
Make sure the **Use default location** checkbox is checked and enter a name for your project in the **Project name** field. Notice that the path to your new Rule Project is automatically appended to Corticon Studio's default workspace folder to define its **Location**. Click **Finish** to create your new Rule Project.

You can also choose to create your new Project in a different Workspace by using the **Browse...** button to locate and select it.

The Rule Project explorer window

The Rule Project you just created should now be displayed in the **Project Explorer** window within Corticon Studio for Analysts.

The **Project Explorer** window provides a convenient way of navigating between your Corticon Studio for Analysts files, including Vocabularies, *Rulesheets*, *Ruleflows*, and *Ruletests*. Double-clicking on a highlighted file in the list shown in the **Project Explorer** window below will cause the file to open and become the active window in Corticon Studio for Analysts. Our new `Example_Project`, shown below, is empty, so there are no files available to open yet.



Now that we have created a Rule Project, we can turn our attention to creating a Vocabulary that we will use to model our Business Rules.

Vocabularies

A Vocabulary is used to build rule models in a *Rulesheet* or test cases in a *Ruletest* (see the [Rulesheet](#) and [Ruletest](#) sections of this manual). Once a Vocabulary is open, other Corticon Studio options, including a Vocabulary option in the menubar and relevant toolbar icons, become available.

Vocabulary files have the file suffix `.ecore`.

Important: The "Creating a Vocabulary" chapter in the *Rule Modeling Guide* describes the Vocabulary modeling process, including the use of these options.

For details, see the following topics:

- [Creating a Vocabulary](#)
- [The Vocabulary window](#)
- [Populating a new Vocabulary](#)

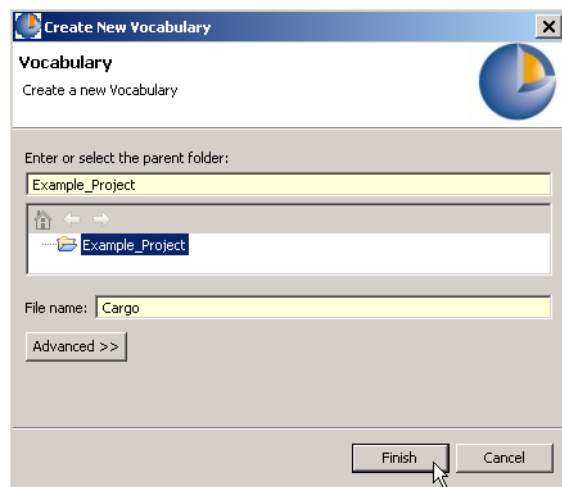
Creating a Vocabulary

To create a new Vocabulary:

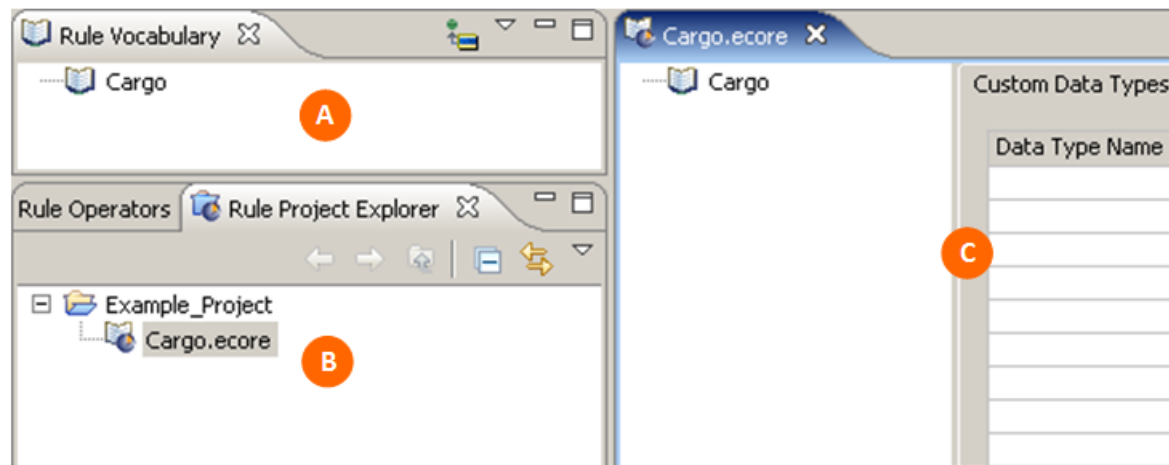
1. Do one of the following:
 - Select **File > New > Rule Vocabulary** from the Corticon Studio menubar
 - Click the down arrow to the right of the **New** icon

- Right-click in the **Project Explorer** to open its menu, and then choose **New > Rule Vocabulary**.

These techniques all launch the same **Create a New Vocabulary** wizard, illustrated below.



2. Select the parent Rule Project for the new Vocabulary by highlighting the `Example_Project` folder we just created.
3. Enter a name for the new Vocabulary in the **File name** field. It is not necessary to type the file extension `.ecore` (we used `Cargo` here).
4. Click **Finish** to create your new Vocabulary. It is now displayed in the new **Rule Vocabulary** window (A), in the **Rule Project Explorer** window (B), and as the open file and active tab in the **Cargo.ecore** window (C). The window sizes have been adjusted to fit on the page.



The Vocabulary window

The **Vocabulary** window is the window with a tab containing the name of your Vocabulary file ([Cargo.ecore window \(C\)](#)), provides all the tools you need to create a new or modify an existing Vocabulary.

Vocabulary menu commands

The following menubar commands are available when a Vocabulary window is the active window, as shown above.

Note: The Database Access section is available only when you have set your user role preference to **Integration & Deployment**.

Vocabulary Menu

The following actions are accessible only when the active file is a vocabulary (.ecore) file:

- **Add Domain** - Adds a Domain to the Vocabulary. Domains are discussed in the *Rule Modeling Guide*.
- **Add Entity** - Adds an Entity to the Vocabulary.
- **Add Attribute** - Adds an Attribute to the selected Entity.
- **Add Association** - Adds an Association between two existing Entities where the selected entity is set as the Source entity.
- **Database Access > Import Database Metadata** - Imports database metadata.
- **Database Access > Clear Database Metadata** - Remove any database metadata previously imported.
- **Database Access > Create/Update Database Schema** - Creates or updates the database schema.

Note: This feature is supported only on certain database brands. When you choose a Database Server, the feature will be enabled only if it is supported for that brand.

- **Database Access > Export Database Access Properties** - Outputs database access connection information for use in creation of deployment descriptors.
- **Database Access > Validate Mappings** - Tests the database mapping, and displays any issues that are detected.
- **Java Object Messaging > Import Java Class Metadata** - Imports Java object metadata from a specified jar file.
- **Java Object Messaging > Clear Java Class Metadata** - Removes imported Java class metadata from memory.
- **Localize** - Lets you set the Language parameter (Locale) for the Vocabulary and displays the Vocabulary's elements in a list.
- **Report** - Creates an HTML report and launches your browser for viewing. See [Creating a Vocabulary Report](#).

Vocabulary toolbar

When a Vocabulary window is active, commands are added to the toolbar, as shown:



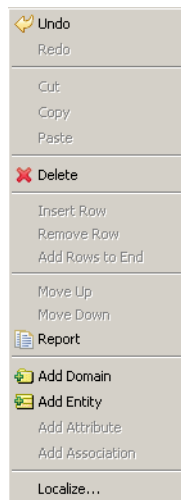
The Vocabulary icons provide the same functions as the corresponding menu commands:

-  **Vocabulary > Add Domain.**
-  **Vocabulary > Add Entity.**
-  **Vocabulary > Add Attribute.**
-  **Vocabulary > Add Association.**

Context-sensitive right-click pop-up menu

In addition to the menubar and toolbar functions described above, Corticon Studio also provides many Vocabulary functions within a convenient right-click pop-up menu.

A sample Vocabulary Editor Pop-up menu is displayed below.



Important: The right-click pop-up menus are context-sensitive, meaning not all options are available for all Vocabulary nodes. For example, the "Add Attribute" option is only available when an Entity is selected in the Vocabulary tree.

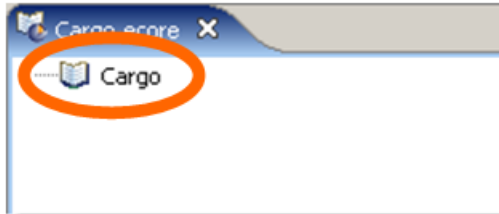
Populating a new Vocabulary

Important: The discussion in this section assumes Vocabulary creation and editing is conducted while in Corticon Studio's **Rule Modeling** mode. Features such as XML and Java Object mapping and the Enterprise Data Connector require **Integration & Deployment** mode, a preference described in detail in the *Integration & Deployment Guide*.

The Vocabulary tree view

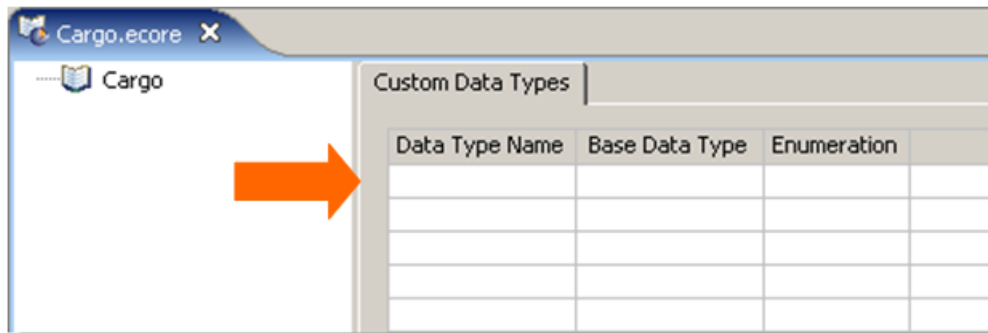
All elements of a Vocabulary are referred to generically as nodes, and these **nodes** are arranged in the Vocabulary window in a hierarchical, or "tree" view.

When a new Vocabulary is displayed in its window, the tree view is empty with the exception of the Vocabulary's "name" node. The name node contains the name of the Vocabulary and a "open book" icon to its left, as show below:



Vocabulary properties

Each type of node in the Vocabulary (including the name) has its own types of properties. These properties are displayed immediately to the right of the node simply by clicking on the node name or icon.



The Vocabulary's name node properties consist of **Custom Data Types** definitions. Refer to the "Building the Vocabulary" chapter of the *Rule Modeling Guide* for complete details regarding the creation and use of Custom Data Types.

Vocabulary node naming restrictions

Following are some guidelines that you should use when creating new nodes in the Vocabulary:

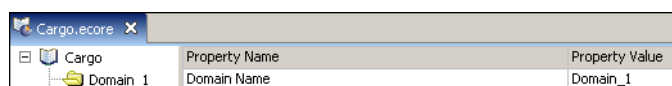
- Domain, Entity, Attribute, and Association Role node names may only contain the following alphanumeric characters: letters, numbers, and underscores. However:
 - Domain, Entity, Attribute, and Association Role node names may **not** begin with a number.
 - Spaces are **not** allowed in Domain, Entity, Attribute or Association node names. If the preferred name is composed of multiple words, then underscore characters may be used to combine them in a single-word name.
 - Domain, Entity, Attribute, and Association Role node names may begin with an underscore.
 - Domain, Entity, Attribute, and Association Role node names may begin with either an upper or lower case letter.
- No two Entities in the same Domain may have the same node name (case-insensitive).
- Names of Operators (such as `sum`, `size`, `month`, `now`, `today`) or Extended Operators (if any) may **not** be used as Entity or Domain node names. They *may* be used as Attribute or Association Role node names.
- Custom Data Type names may not use the names of any of the base data types (such as `string`, `decimal`, `boolean`). See the *Rule Modeling Guide*'s chapter "Creating the Vocabulary" for more information about Custom Data Types.
- Names of aliases in the Scope section may not match (case-sensitive) the node names of Entities in the Vocabulary.
- No two Attributes/Association Role nodes in the same Entity may have the same name (case-insensitive).
- No two Domains within another Domain may have the same node name (case-insensitive).
- No two root-level Domains may have the same node name(case-insensitive).

Adding nodes to the Vocabulary tree view

Domain nodes: Adding and editing domains and their properties

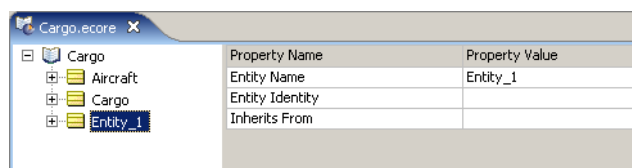
The purpose of Domains is explained in more detail in the *Rule Modeling Guide*, "Creating the Vocabulary" chapter. Generally, if all your Entity names are unique, you won't need to use Domains in your Vocabulary. To add a Domain node:

1. Select the name node in the Vocabulary tree view, which is the one with the  icon.
2. Do one of the following:
 - Right-click to display the pop-up menu and choose **Add Domain**
 - Choose **Vocabulary > Add Domain** from the menubar.
3. Select the new Domain node in the Vocabulary tree to display its properties.
4. Assign a Name for the new Domain in the Property Value column to the right, as shown below. Or, double-click the new Domain node in the tree view to edit the default `Domain_1` value. Your changes in either location will be updated in both.



Entity nodes: Adding and editing entities and their properties

1. Do one of the following:
 - Select the name node in the Vocabulary tree to add a new Entity node to the "root level" of the tree.
 - Select the newly created Domain node to add a new Entity to the Domain.
2. Do one of the following:
 - Right-click to display the pop-up menu and choose **Add Entity**
 - Choose **Vocabulary > Add Entity** from the menubar.
 - Choose  from the toolbar
3. Select the Entity in the Vocabulary tree to display its properties.
4. Assign property values as shown below:



The window shown here lists the basic properties common to every Entity.

Table 1: Basic Entity Properties

Property	Value
Entity Name	Assign a name to the entity. By default, this will be pre-filled as <code>Entity_x</code> , where <code>x</code> is an automatically determined unique number. As with Domains, double-clicking the node in the tree view will also open an editing box. Name changes made in either the node or the property will update in both places
Entity Identity	Used only when using EDC. Specifies which attributes (if any) act as its Entity's primary key. Chose multiple attributes on the pulldown list by opening the list then holding SHIFT while clicking the selections.
Inherits From	Optional. The Vocabulary model contains extensive support for inheritance concepts. For more information on using this feature, refer to the <i>Rule Modeling Guide</i> , "Creating the Vocabulary" chapter.

When set to Integration & Deployment mode, additional Entity properties are available. These properties and their usage are discussed in the *Integration & Deployment Guide's* chapter "Preparing Studio files for deployment." >

Table 2: XML Mapping and Java Object Mapping Entity Properties

Property	Value
XML Namespace	Specifies the full namespace of XML Element Name when there is no exact match.
XML Element Name	Specifies the XML Element Name when there is no exact match.
Java Package	Specifies the package to be used for Java class metadata mapping.
Java Class Name	Maps the entity to the specified class when no class exists with the the entity name.

When set to Integration & Deployment mode, additional Entity database properties are available.

Table 3: Enterprise Data Connector (EDC) Entity Properties

Property	Description	Values	Applicability
Datastore Persistent	Indicates whether this entity will be database bound.	Yes, No	Required, defaults to No.
Table Name	Name of database table, chosen from a drop-down list of all database table names, fully-qualified with catalog and schema if applicable.		Optional, if not specified system will infer best matching table from database metadata.
Datastore Caching	Indicates whether instances of this entity will be subject to caching.	Yes, No	Required, defaults to No.
Identity Strategy	Strategy to generate unique identity for this entity.	Native, Table, Identity Sequence, UUID (These are described in the following table.)	Optional. Only enabled if Entity Identity is not specified and DataStore Persistent is set to Yes.
Identity Column Name	Name of the identity column, chosen from a drop-down list consisting of all column names associated with the table.		Optional. Only enabled if Entity Identity is unspecified. If not specified, system will attempt to find match, namely <entity>_ID.
Identity Sequence	The fully-qualified name of the sequence to be used.		Only applicable if Entity Identity is unspecified and Identity Strategy is Sequence. Required if enabled.
Identity Table Name	The fully-qualified name of the identity table to be used, chosen from a drop-down list of all table names and sequence names.		Only applicable if Entity Identity is unspecified and Identity Strategy is Table. Optional. If not specified, the value will default to SEQUENCE_TABLE.

Property	Description	Values	Applicability
Identity Table Name Column Name	The name of the column in the identity table that is used as the key (this column will contain the name of the entity). Chosen from a drop-down list of all columns in the table selected in the Table Name field.		Only applicable if Entity Identity is unspecified and Identity Strategy is Table. If not specified the value will default to SEQUENCE_NAME (String).
Identity Table Value Column Name	The name of the column which holds the identity value. Chosen from a drop-down list of all columns in the table selected in the Table Name field.		Only applicable if Entity Identity is unspecified and Identity Strategy is Table. If not specified the value will default to NEXT_VAL (Big Integer).
Version Strategy	Strategy to control optimistic concurrency.	Version Number Timestamp	Optional.
Version Column Name	Name of column that contains either version number or timestamp.		Only applicable if Version Strategy is specified. Required if enabled.

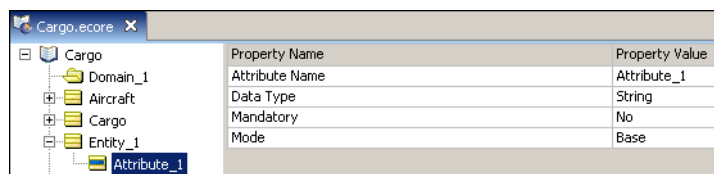
Table 4: Description of the Identity Strategy values

Strategy	Description
Native	Allows database to choose best possible identity strategy.
Table	Uses a database table, whose name may be specified in Identity Table Name. This table will have two columns: a name column and a value column. (Previously referenced as "increment".)
Identity	Uses the native identity capability in DB2, SQL Server (the column is defined as an "identity" column in the database schema).
Sequence	Uses sequence capability in DB2, PostgreSQL and Oracle.
UUID	Generates 128-bit UUID string of 32 hex digits. (Previously referenced as "uuid-hex".)

Note: Databases mentioned in the Identity Strategy table do not imply that they are currently supported RDBMS brands. Contact Progress Support for details on the latest databases supported.

Attribute nodes: Adding and editing attributes and their properties

1. Select any Entity node in the Vocabulary tree view.
2. Do one of the following:
 - Right-click to display the pop-up menu and choose **Add Attribute**
 - Choose **Vocabulary > Add Attribute** from the menubar.
 - Choose  from the toolbar
3. Select the Entity in the Vocabulary tree to display its properties.
4. Assign property values as described in the following table.



The screenshot shows a software window titled 'Cargo.ecore'. On the left is a tree view with a hierarchy: Cargo (folder) -> Domain_1 (folder) -> Aircraft (folder) -> Cargo (folder) -> Entity_1 (folder) -> Attribute_1 (selected). On the right is a table with two columns: 'Property Name' and 'Property Value'.

Property Name	Property Value
Attribute Name	Attribute_1
Data Type	String
Mandatory	No
Mode	Base

Note:

The window shown here lists the basic properties common to every Attribute.

Table 5: Basic Attribute Properties

Property	Value
Attribute Name	Assign a name to the new Attribute. As with Domain and Entity nodes, double-clicking the node in the tree view will also open an editing box. Name changes made in either the node or the property will update in both places
Data Type	Enter the data type of the attribute. The default value is String. Other available data types: Boolean, Decimal, DateTime, Date, Integer and Time. You can also have a custom data type. All types are described in the <i>Rule Language Guide</i> and in the <i>Rule Modeling Guide</i> 's "Creating the Vocabulary" chapter.
Mandatory	A mandatory attribute cannot have a value of null. This setting affects the members of the values sets shown in <i>Rulesheet</i> drop-downs. For example, an attribute whose Mandatory value is No will always include a null value selection in its <i>Rulesheet</i> drop-downs.
Mode	Choose the attribute's Mode from the drop-down list. <i>Base</i> attributes exist or are used by systems outside <i>Corticon</i>, and are included in the XML schemas and contracts generated by the Deployment Console. <i>Base</i> attributes map directly to an element in the XML CorticonRequest/Response documents or object properties processed by the <i>Server</i> in production. <i>Extended Transient</i> attributes are <i>derived</i> fields; they exist only during rule execution and are not part of XML schemas generated by the Deployment Console. They do not need to be included in the XML CorticonRequest documents or objects, and they are not included in the XML CorticonResponse documents or objects produced by the <i>Server</i> in deployment. Likewise, they cannot be dragged into a <i>Ruletest</i>'s Input column because they are created as output.

When set to Integration & Deployment mode, additional Attribute properties are available. These properties and their usage are discussed in the *Server Integration & Deployment Guide*'s chapter "Preparing Studio files for deployment."

Table 6: XML Mapping and Java Object Mapping Attribute Properties

Property	Value
XML Namespace	Specifies the full namespace of XML Element Name when there is no exact match.
XML Element Name	Specifies the XML Element Name when there is no exact match.
Java Object Get Method	Manually specifies the GET method of a class property that does not conform to naming conventions of the auto-mapper.
Java Object Set Method	Manually specifies the SET method of a class property that does not conform to naming conventions of the auto-mapper.
Java Object Field Name	Manually specifies a public instance variable name.

When set to Integration & Deployment mode, additional Attribute database properties are available.

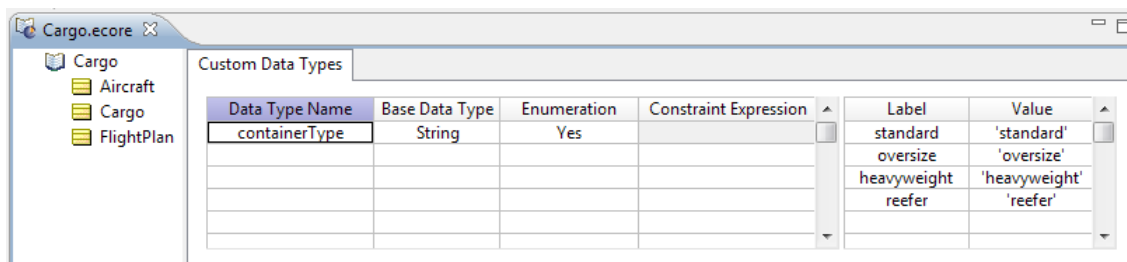
Table 7: Enterprise Data Connector (EDC) Attribute Properties

Property	Description	Values	Applicability
Column Name	Name of the database column, chosen from a drop-down list consisting of all column names associated with the Entity Table Name (see Entity Properties).		Optional, if not specified, system will infer best match from database metadata.
Value Strategy	Strategy to use to generate unique value for this column.	Native, Table, Identity, Sequence, UUID	Optional. Only enabled if attribute is an element of the Entity Identity set. Only value strategies that make sense with respect to the attribute data type will be presented in the drop-down list.
Value Sequence	The fully-qualified name of the sequence to be used.		Only enabled if attribute is an element of the Entity Identity set and Identity Strategy is Sequence. Required if enabled.
Value Table Name	The fully-qualified name of the identity table to be used, chosen from a drop-down list of all table names and sequence names.		Only enabled if attribute is an element of the Entity Identity set and Identity Strategy is Table. Optional. If not specified, the value will default to SEQUENCE_TABLE.

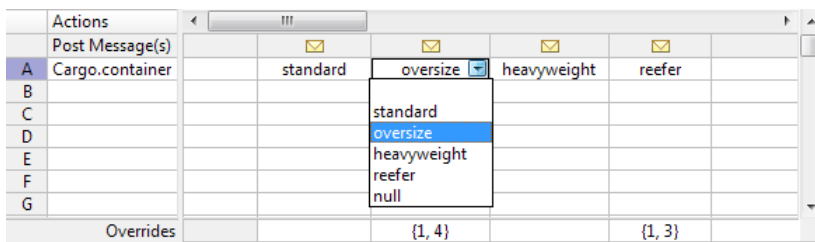
Property	Description	Values	Applicability
Value Table Name Column Name	The name of the column in the identity table that is used as the key (this column will contain the name of the entity). Chosen from a drop-down list of all columns in the table selected in the Table Name field.		Only enabled if attribute is an element of the Entity Identity set and Identity Strategy is Table. If not specified the value will default to SEQUENCE_NAME (String).
Value Table Value Column Name	The name of the column which holds the identity value. Chosen from a drop-down list of all columns in the table selected in the Table Name field.		Only enabled if attribute is an element of the Entity Identity set and Identity Strategy is Table. If not specified the value will default to NEXT_VAL (Big Integer).

Enumerated values

You can define lists of values that are the set of allowable values associated with a Vocabulary attribute. In the *Basic Tutorial*, you saw how we could delimit the options for a `containerType` by defining labels and their respective values:

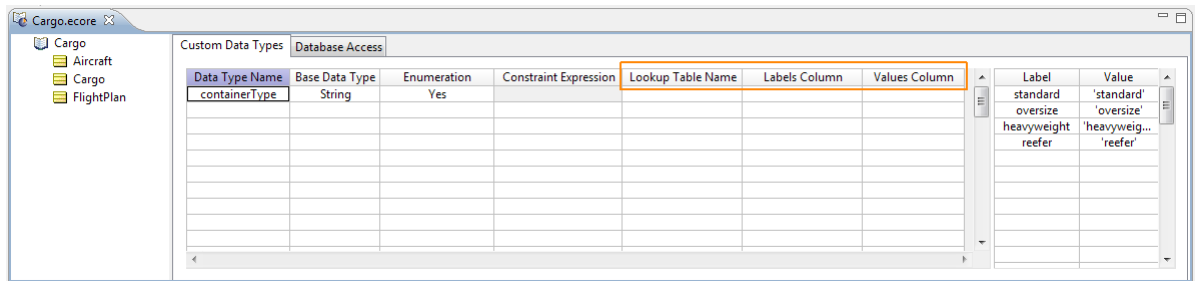


Then, when you are in the Rulesheet, the defined values -- as well as `null` and `blank` -- were offered.



Importing enumerated values from a database

When you use the Enterprise Data Connector and establish connection to a database, additional functionality is added to the **DataTypes** tab:



You can specify a column within a table of the connected database to retrieve and import the name and values (or just the values) to populate the selections to the specified attribute.

Note: For more information about enumerations and retrieving values from databases, see:

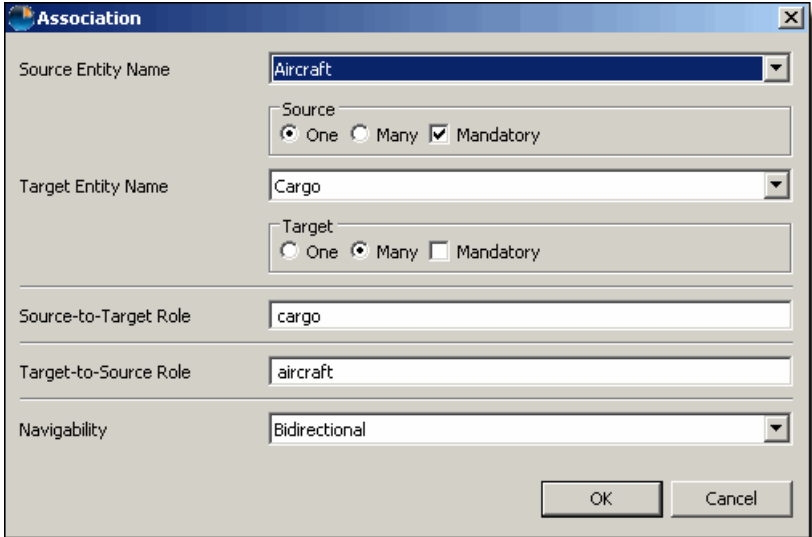
- *"Enumerations defined in the Vocabulary" in the Rule Modeling Guide*
 - *"Enumerations retrieved from a database" in the Rule Modeling Guide*
 - *"Importing an attribute's possible values from database tables" in the Using EDC Guide*
 - *"Mapping database tables to Vocabulary Entities" in the Integration and Deployment Guide*
-

Association Nodes: Adding and editing associations and their properties

To add a new Association node:

1. Select an Entity in the Vocabulary tree view.
2. Perform one of the following:
 - Right-click to display the pop-up menu and choose **Add Association**.
 - Choose **Vocabulary > Add Association** from the menubar.

- Choose  from the toolbar.
3. Complete the Association window as shown in the following table:



Note: This window is shown as it is in Rule Modeling mode.

Property	Value
Source Entity Name	An association relates two entities. Corticon Studio refers to one entity as the "source" and the other as the "target." Choose the name of the supplier from the drop-down list, which includes the entity names already defined in this Vocabulary.
Target Entity Name	Choose the name of the 2 nd entity – the "target" entity – from the drop down list.

Property	Value
Cardinality Radio Buttons	Select the combination of radio buttons that describe the relationship you want between the two entities. The possible combinations are:
	One-To-Many. This is the default and is shown as 1>* in the properties. This means that a given instance of the supplier entity may be related to multiple instances of the target entity. Displays as: ↵
	One-To-One. Shown as 1>1, it means that a given instance of the supplier entity may be related to a single instance of the target entity. Displays as: —
	Many-To-One. Shown as *>1, it means that multiple instances of the supplier entity may be related to a single instance of the target entity. Displays as: ➤
	Many-To-Many. Shown as *>*, it means that multiple instances of the supplier entity may be related to multiple instances of the target entity. Displays as: ✕
	Mandatory. Also known as optionality. Select this box if <i>at least one</i> instance of the source or target MUST be present in data sent to the Corticon Server to be processed by rules using this Vocabulary.
Role Names	Role names are useful when two entities share more than one association. Custom role names can give each association a unique, descriptive name. Source-To-Target provides a name for the association from the perspective of the source entity. Target-To-Source provides a name for the association from the perspective of the target entity.
Navigability	Bidirectional. Select Bidirectional if you want the association to be traversable (visible) in both directions within the Vocabulary. This means that associations between two entities are visible from each entity in the Vocabulary tree view. This option provides the most flexibility when writing rules. Source Entity < Target Entity. Use this option to prevent the association <i>from</i> the Target entity <i>to</i> the Source entity from being displayed in the Vocabulary tree view. This option prevents a rule from being written using the scope <code>Target_entity.source_entity.attribute</code>. Source Entity > Target Entity. Use this option to prevent the association <i>from</i> the Source entity <i>to</i> the Target entity from being displayed in the Vocabulary tree view. This prevents a rule from being written using the scope <code>Source_entity.target_entity.attribute</code>. See the <i>Rule Modeling Guide</i> for more information on using associations in rule modeling.

When set to Integration and Deployment mode, additional Association properties are available. These properties and their usage are discussed in the *Server Integration & Deployment Guide's* chapter "Preparing Studio files for deployment".

Table 8: XML Mapping and Java Object Mapping Association Properties

Property	Value
XML Property Name	Specifies the XML Element Name when there is no exact match to the Vocabulary association name.
Java Object Get Method	Manually specifies the GET method of a class property that does not conform to naming conventions of the auto-mapper.
Java Object Set Method	Manually specifies the SET method of a class property that does not conform to naming conventions of the auto-mapper.
Java Object Field Name	Manually specifies a public instance variable name.

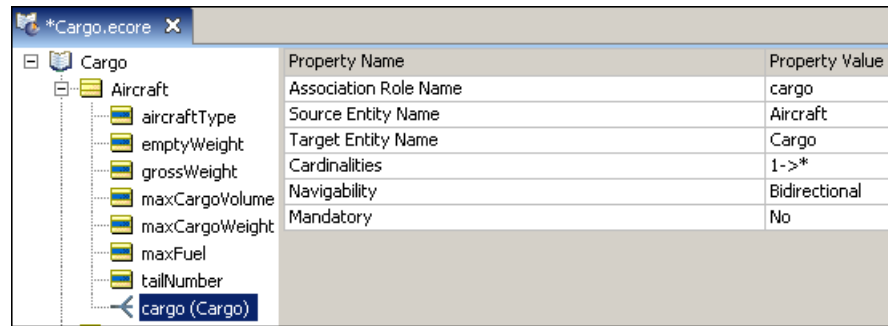
When set to Integration & Deployment mode, additional Association database properties are available.

Table 9: Enterprise Data Connector (EDC) Attribute Properties

Property	Description	Applicability
Join Expression	Expression that defines the relationships between foreign key columns in the database	Required for all database-mapped associations. Inferred in most instances from database metadata (exceptions: unary associations and certain many-to-many associations).

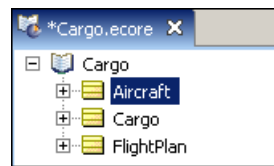
Editing an association

1. Select an association on the Vocabulary tree to display its properties.
2. Edit properties as described in [Association nodes: Adding and editing associations and their properties](#).



Renaming a vocabulary node

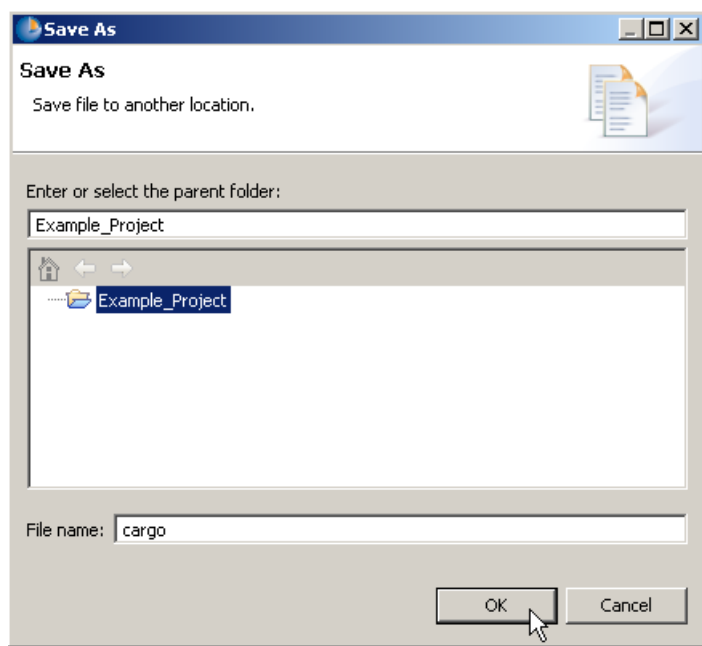
1. Double-click on the node name or icon to display an editing box.
2. Type the preferred node name and press **Enter**.



Saving a new Vocabulary

1. Do one of the following:
 - Save the Vocabulary from the toolbar or choose **File > Save** from the menubar to save any changes you have made.

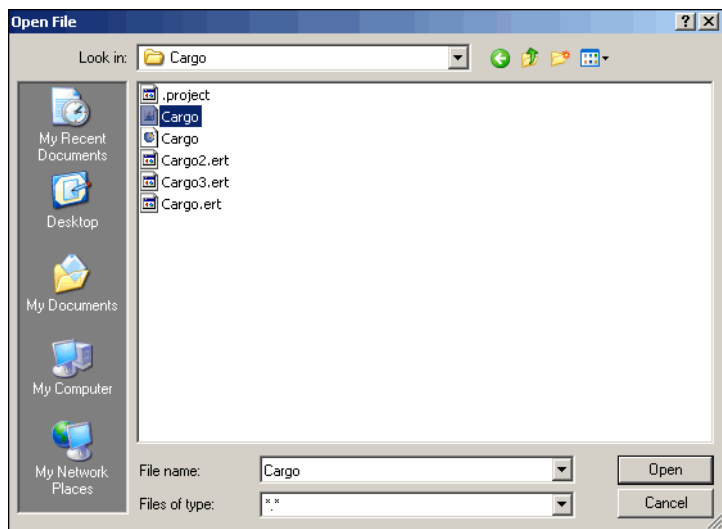
- Use **File > Save As** on the menubar and Navigate to the directory where you want to store the Vocabulary, or click  to create a new directory.



2. Type any name, including embedded blanks (max. of 36 characters) in length, and click **OK**.
3. See [File Naming Restrictions](#) for naming restrictions.

Opening an existing Vocabulary

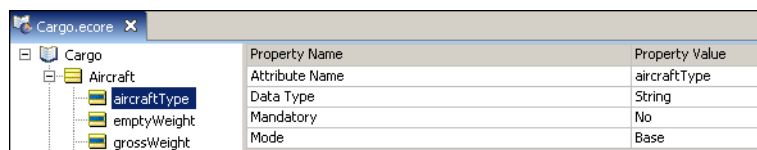
1. Choose **File > Open File** from the Corticon Studio menubar or use the toolbar to display the **Open** window.



2. Navigate to the directory where the Vocabulary you want is stored.
3. Select the appropriate `.ecore` file and click **Open**.

Modifying a Vocabulary

1. Choose the **Vocabulary Editor** tab to enter the Vocabulary editor.
2. Select any node to display its properties.



3. Modify node properties as described above.

Creating a Vocabulary report

1. Choose **Vocabulary > Report** from the menubar.
2. If your local machine has a web browser installed, the HTML report opens as a new web page.
3. Corticon Studio saves the report file to `/Users/<your username>/AppData/Local/Temp/` using the name format `CorticonVocabularyXXXXX.html`, where XXXXX is a unique auto-generated number. You can save the HTML to a different name or location from the browser.
4. For information about creating custom reports or saving them to custom locations, see the "Corticon Reporting Framework" chapter of the *Rule Modeling Guide*.

Rulesheets

A *Rulesheet* is a file that contains a set of business rules. By organizing these rules, it becomes a self-contained, independent "unit" of automated decision-making. A Rule Project can include any number of *Rulesheets*.

Following test and validation, one or more *Rulesheets* may be packaged in a *Ruleflow* (see [The Ruleflow](#) chapter) and deployed into a production environment (described in the *Server Integration & Deployment Guide*). Once packaged in a *Ruleflow* and deployed and available to other IT systems, we refer to the *Ruleflow* as a **Decision Service**.

Because a Rule Project may contain multiple *Rulesheets*, there are two methods of navigating between open *Rulesheets* within Corticon Studio:

- Double-clicking on any *Rulesheet* name in the **Rule Project Explorer** window opens that *Rulesheet* and makes it active.
- Clicking on any *Rulesheet* tab makes that *Rulesheet* active.

Multiple *Rulesheets* may be open in Corticon Studio simultaneously, although only one may be active at any given time.

A *Rulesheet* file has the suffix `.ers`.

Rulesheet Window

Opening a *Rulesheet* or making a *Rulesheet* the active Corticon Studio window causes the Studio menubar and toolbar to display the tools needed to begin working with rules.

For details, see the following topics:

- [Creating a new Rulesheet](#)
- [Rulesheet menu commands](#)

- [Rulesheet toolbar](#)
- [Context-sensitive right-click pop-up menus](#)
- [Rulesheet sections](#)
- [Rule statements window](#)
- [Using the business vocabulary to build rules](#)
- [Using the operator vocabulary to build rules](#)
- [Naming Rulesheets](#)
- [Deleting Rulesheets](#)
- [Saving a new Rulesheet](#)
- [Validating and optimizing a Rulesheet](#)
- [Creating a Rulesheet report](#)
- [Closing a Rulesheet](#)
- [Saving a modified Rulesheet](#)

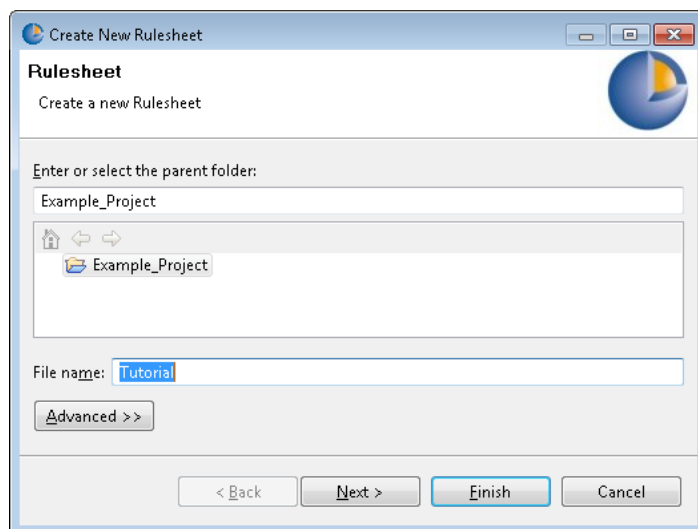
Creating a new Rulesheet

Creating *Rulesheets* involves the same general process as described in [Creating a Rule Project](#) and [Creating a Vocabulary](#). Follow these steps to create a new *Rulesheet*:

1. Choose **File > New > Rulesheet** from the menubar

OR

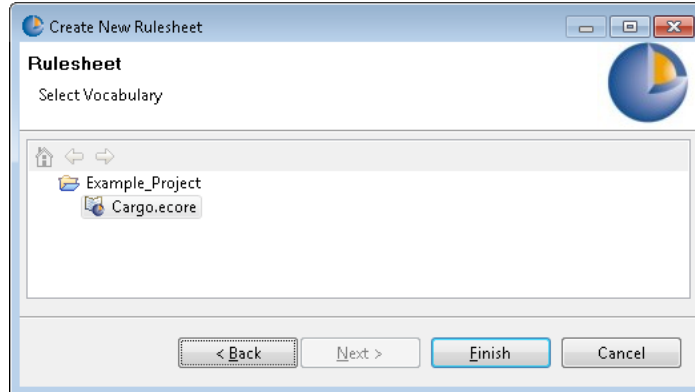
2. Click the down arrow next to the **New** icon  on the toolbar and select **Rulesheet**. Either method launches the **Create New Rulesheet** wizard.



3. Highlight the Rule Project you would like to associate the new *Rulesheet* with in the list (or enter it manually in the **Enter or select parent folder:** field).

4. Enter a file name for the *Rulesheet* in the **File name:** field. Click **Next** to continue.

Figure 1: The Create New Rulesheet Wizard window

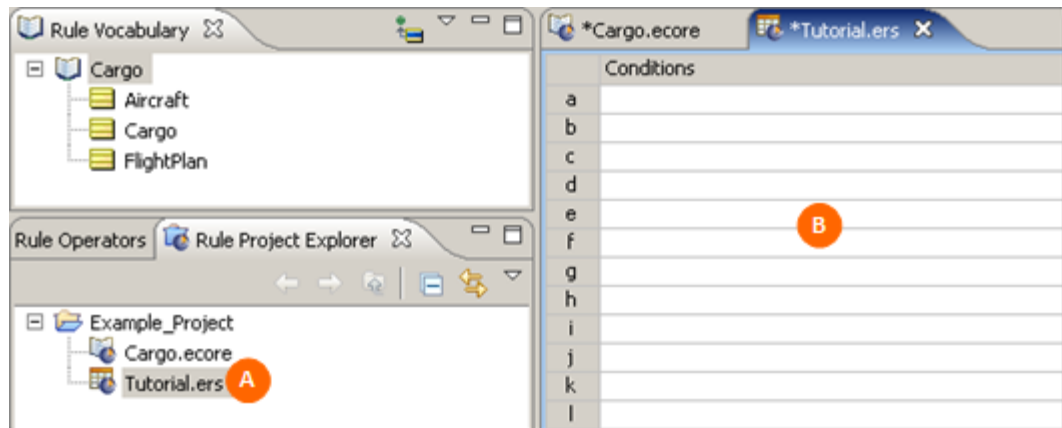


5. Select the Vocabulary with which to associate the *Rulesheet*. A *Rulesheet* can only be associated with one Vocabulary. In this example, we expanded the *Example_Project* folder and chose the *Cargo.ecore* file we created earlier. As you create additional Vocabulary files within a Rule Project, they become available for selection from this list.

You need to have a Vocabulary within a Rule Project (either imported samples or newly created) in order to create a Rulesheet.

6. Click **Finish**. Your new *Rulesheet* is now displayed in the **Rule Project Explorer** window (A) and in a new *Rulesheet* window with a tab of the given name (B).

Figure 2: A new Rulesheet as it appears in Corticon Studio



A *Rulesheet* menubar and toolbar replace the Vocabulary menubar and toolbar and you are now ready to begin modeling your rules.

Rulesheet menu commands

The following menubar options are available when a Rulesheet window is the active window, as shown above.

Note: This is an an additional menu. Other menus are unchanged.

Rulesheet Menu

The following actions are accessible only when the active file is a rulesheet (.ers) file:

- **Logical Analysis > Execution Sequence Diagram** - Creates a graphic that shows the execution sequence of the rules in the active *Rulesheet*. Your default graphic viewer launches to display the diagram, and a copy of the file is saved to [CORTICON_HOME]\Studio\plugins\com.corticon.services_n.n.n\DepGraph.
- **Logical Analysis > Logical Dependency Graph** - Creates a graphic that shows the logical dependencies of the data created in the *Rulesheet*. This is different than an Execution Sequence Diagram in that it does not show the sequence of rule execution. Your default graphic viewer launches to display the diagram, and a copy of the file is saved to [CORTICON_HOME]\Studio\plugins\com.corticon.services_n.n.n\DepGraph.
- **Logical Analysis > Clear Analysis Results** - Removes highlighting that resulted from checking for conflicts and completeness.
- **Logical Analysis > Check for Logical Loops** - Performs logical loop detection across all rules in all *Rulesheets* in the active Project.
- **Logical Analysis > Check for Completeness** - Highlights incompleteness in the active *Rulesheet*.
- **Logical Analysis > Check for Conflicts** - Highlights conflict between two or more rules in the active *Rulesheet*.
- **Logical Analysis > Enable Conflict Filter** - A toggle that either hides or shows all rule columns not part of the current conflict.
- **Logical Analysis > Previous Conflict** - Highlights the previous conflict in the sequence. This option is grayed out until you perform a conflict check.
- **Logical Analysis > Next Conflict** - Highlights the next conflict in the sequence. This option is grayed out until you perform a conflict check.
- **Logical Analysis > First Conflict** - Highlights the first conflict in the sequence. This option is grayed out until you perform a conflict check.
- **Logical Analysis > Last Conflict** - Highlights the last conflict in the sequence. This option is grayed out until you perform a conflict check.
- **Rule Columns > Use Conditions as Processing Threshold** - Advanced functionality described in the *Rule Modeling Guide*, Dependency & Looping chapter
- **Rule Columns > Expand Rules** - Creates multiple simple rules from each complex rule.
- **Rule Columns > Collapse Rules** - Reverses the Expand Rules function.
- **Rule Columns > Compress Rules** - Detects overlapping conditions between rules and combines columns to produce a smaller number, but logically equivalent, set of rule columns.
- **Rule Columns > Renumber Rules** - Causes rule columns that have been expanded into component rules (also called sub-rules) to be renumbered from left to right.
- **Advanced View** - A toggle that switches between the advanced and simple *Rulesheet* views.
- **Show Natural Language** - Displays in natural language.
- **Filters > Database Filter** - When EDC is enabled, this is a toggle that -- when cleared -- is a filter that is applied to the data currently in working memory. When checked, the filter is a database query that can retrieve data from the database, and add that data to working memory.

- **Filters > Precondition** - Preconditions are Filter expressions that stop *Rulesheet* execution if not satisfied by at least one piece of data. For more information, see the Filters chapter of the *Rule Modeling Guide*.
- **Processing Mode > Optimized Inferencing** - Default selection. This option affects the active *Rulesheet* only. Causes rules to execute without re-evaluation or re-execution (no looping). For more information about this processing mode, see the *Rule Modeling Guide*.
- **Processing Mode > Advanced Inferencing** - Allows only multi-rule inter-dependencies to re-evaluate and re-execute. This option affects the active *Rulesheet* only. For more information about this processing mode, see the *Rule Modeling Guide*.
- **Processing Mode > Advanced Inferencing with Self-Triggering** - Allows all rule dependencies (including self-dependencies) to re-evaluate and re-execute. This option affects the active *Rulesheet* only. For more information about this processing mode, see the *Rule Modeling Guide*.
- **Localize** - Opens the Rulesheet localization window. See the Localization section of the *Rule Modeling Guide* for more information.
- **Report** - Creates an HTML report and launches your browser for viewing. See [Creating a Rulesheet Report](#).

Rulesheet toolbar

When a Rulesheet window is active, commands are added to the toolbar, as shown:

Many of the functions provided by the toolbar are also accessible on the Rulesheet menu.



The Rulesheet icons provide the same functions as the corresponding menu commands:

-  Simple  Advanced - Toggles to the advanced or simple *Rulesheet* views. New *Rulesheets* are shown in Simple view.
-  - Switches the view of the active *Rulesheet* to Natural Language view. See the *Rule Modeling Guide* for more information about this feature.
-  - Expands conditions and actions to display all component rules (those containing no dashes). Same as menu command **Rulesheet > Rule Column(s) > Expand Rules**.
-  - Collapses conditions and actions to hide component rules and show only general rules (those containing dashes). Same as menubar option **Rulesheet > Rule Column(s) > Collapse Rules**.
-  - Compresses conditions and actions to eliminate redundancies within general and component rules. Same as menu command **Rulesheet > Rule Column(s) > Compress Rules**.

-  - Clears the colored highlights that appear when an conflict or completeness check is performed. Same as menu command **Rulesheet > Logical Analysis > Clear Analysis Results**.
-  - Performs logical loop detection. Same as menu command **Rulesheet > Logical Analysis > Check for Logical Loops**.
-  - Performs completeness check. Same as menu command **Rulesheet > Logical Analysis > Check for Completeness**.
-  - Performs conflict check. Same as menu command **Rulesheet > Logical Analysis > Check for Conflicts**.
-  - Highlights the first conflict in a set. This button is grayed out until you perform a conflict check. Same as menu command **Rulesheet > Logical Analysis > First Conflict**.
-  - Highlights the previous conflict in a set. This button is grayed out until you perform a conflict check. Same as menubar option **Rulesheet > Logical Analysis > Previous Conflict**.
-  - Highlights the next conflict in a set. This button is grayed out until you perform a conflict check. Same as menubar option **Rulesheet > Logical Analysis > Next Conflict**.
-  - Highlights the last conflict in a set. This button is grayed out until you perform a conflict check. Same as menubar option **Rulesheet > Logical Analysis > Last Conflict**.
-  - Enables the conflict Filter, which hides all rule columns not part of the current conflict. This feature is a toggle – the filter is *disabled* when the red funnel icon is *visible*. Same as menubar option **Rulesheet > Logical Analysis > Conflict Filter**.
-  - Once enabled, the conflict Filter icon appears like this. Same as menubar option **Rulesheet > Logical Analysis > Conflict Filter**. Click to disable.

Context-sensitive right-click pop-up menus

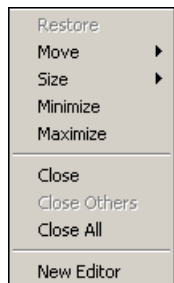
In addition to the menubar and toolbar functions described above, Corticon Studio also provides many functions in convenient right-click pop-up menus. There are two types of right-click pop-up menus that appear when working with *Rulesheets*:

- *Rulesheet* Tab Pop-up Menu appears when a *Rulesheet*'s tab is right-clicked. This menu provides quick access to many of the most frequently used *Rulesheet*-level functions.
- *Rulesheet* Pop-up Menu appears when a section within a *Rulesheet* is right-clicked. This menu provides quick access to many of the most frequently used *Rulesheet*-level functions.

The options available in these pop-up menus depend on which *Rulesheet* section or section the mouse is over when it is right-clicked. Not all options are available for all sections.

Rulesheet Tab Pop-up Menu

The *Rulesheet* Tab Pop-up menu, appearing when a *Rulesheet* tab is right-clicked, is shown below.



Important: The right-click pop-up menus are context-sensitive, meaning not all options will be available at all times.

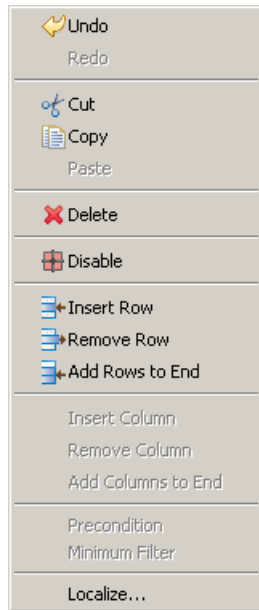
Restore	
Move	Makes the entire <i>Rulesheet</i> window moveable for relocating into other sections of Corticon Studio.
Size	Assists in resizing the <i>Rulesheet</i> window by providing highlighted bars for dragging.
Minimize	Collapses the entire <i>Rulesheet</i> window. Clicking the <i>Rulesheet</i> tab returns it to its original size and position
Maximize	Expands the <i>Rulesheet</i> window to fill the entire screen
Close	Closes the <i>Rulesheet</i> . You will be asked to save any changes
Close Others	Closes any other tabs that may be open. You will be asked to save any changes
Close All	Closes all open Vocabulary, <i>Rulesheet</i> , <i>Ruleflow</i> , and <i>Ruletest</i> windows
New Editor	Opens a copy of the existing <i>Rulesheet</i> window

Many of the options in the *Rulesheet* Tab Pop-up Menu rearrange the Corticon Studio view, or "perspective". If you want to revert to the original, default view, select **Window > Reset Perspective** from the *Studio* menubar.

Rulesheet Pop-up Menu - Row

Right-clicking in a row within a *Rulesheet* causes a pop-up menu to appear with the following options:

A sample *Rulesheet* Pop-up menu when you right-click within a row is shown below.



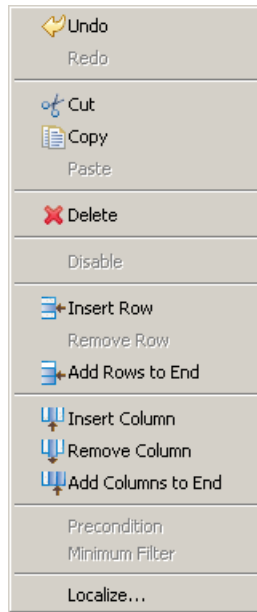
These options repeat several of those found in the *Studio* menubar and toolbar, as described previously.

Important: The right-click pop-up menus are context-sensitive, meaning not all options are available for all active sections of the *Rulesheet*. For example, this pop-up menu displays options available for condition and action rows.

Rulesheet Pop-up Menu - Column

Right-clicking in a column within a *Rulesheet* causes a pop-up menu to appear with the following options:

A sample *Rulesheet* Pop-up menu when you right-click within a column is shown below.



These options repeat several of those found in the *Studio* menubar and toolbar, as described previously.

Important: The right-click pop-up menus are context-sensitive, meaning not all options are available for all active sections of the *Rulesheet*. For example, this pop-up menu displays options available for condition and action columns.

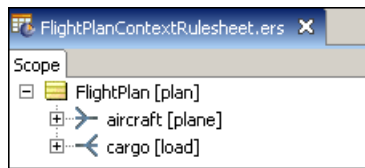
Rulesheet sections

The *Rulesheet* window is divided into several sections, each with a specific use in creating business rules.

Scope

This section is visible only when the **Advanced View** is toggled (the button  on the Corticon Studio toolbar.)

Figure 3: Scope section, showing an Entity and its alias



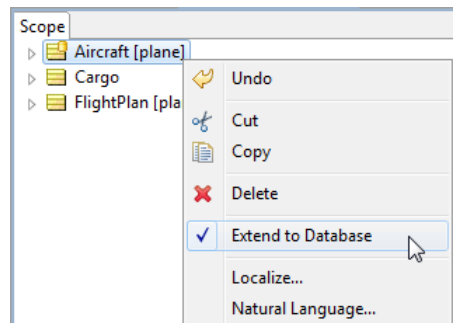
The **Scope** section provides a reduced set of Vocabulary terms with which to build a **Rulesheet**.

The Scope section can be populated *directly* by dragging and dropping Vocabulary elements into it, or *indirectly*, by dragging and dropping Vocabulary nodes onto other *Rulesheet* sections.

Once a node is visible in the Scope section, it can be dragged and dropped to other *Rulesheet* sections henceforth.

For those Entities in the Scope section, aliases and associations can be defined by double-clicking the Entity to open an edit box. An alias replaces the fully qualified entity name wherever it is used in the *Rulesheet*, and serves as a sort of "local" name for use in that *Rulesheet* only. When you are in Integration & Deployment mode, EDC lets an alias be set to **Extend to Database**, as shown:

Figure 4: Setting an alias in scope to Extend to Database

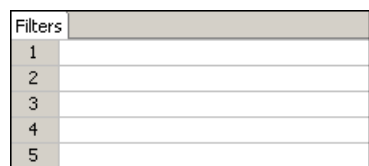


Note: For more information about **Extend to Database**, see the "Writing Rules to access external data" chapter of the *Rule Modeling Guide* and the "Using EDC" chapter of the *Integration and Deployment Guide*.

See the *Rule Modeling Guide* for details on scope, context and *Rulesheet* aliases.

Filters

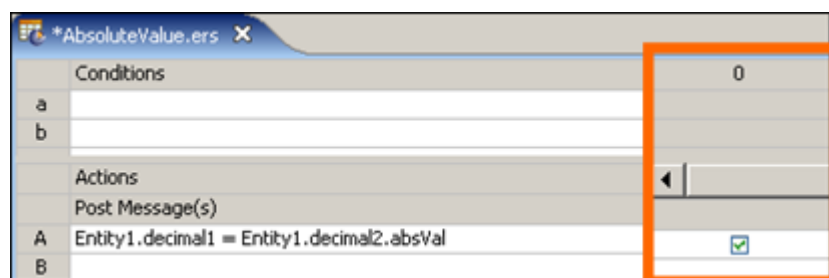
This section is visible only when the **Advanced View** is toggled (the button  on the Corticon Studio toolbar.)



Filters are boolean expressions - they are either **TRUE** or **FALSE** - that are applied to data before rule columns are evaluated. Before using this section, we recommend reading the "Filters" chapter of the *Rule Modeling Guide*.

Nonconditional Rules

This section is visible in all *Rulesheet* views, both **Simple** and **Advanced**



Column 0 in the Conditions/Actions section of the *Rulesheet* is special because the Conditions rows are grayed-out and unusable. Expressions may only be modeled in the Action rows.

Logically, this means that all Actions modeled in Column 0 are unconditionally executed. In other words, no Conditions must be satisfied in order for these Actions to execute.

Each Action row in Column 0 acts as a separate, independent rule.

Important: In versions of Corticon Studio prior to 5.2, a separate Nonconditional Rules section contained this type of logic. Starting in version 5.2, Nonconditional rules appear as Actions in Column 0. Each Action row constitutes a separate Nonconditional rule.

Rules

This section of the *Rulesheet* organizes Conditions, Values and Actions in a table format.

The screenshot shows the 'Cargo.ers' window with a decision table. The table is divided into two main sections: 'Conditions' and 'Actions'. The 'Conditions' section has columns 0, 1, and 2. Column 0 is for unconditional actions, column 1 for actions based on condition 1, and column 2 for actions based on condition 2. The 'Actions' section has columns 0, 1, and 2. Column 0 is for unconditional actions, column 1 for actions based on condition 1, and column 2 for actions based on condition 2. The table is organized into four quadrants: 1 (Conditions), 2 (Actions), 3 (Conditions), and 4 (Actions). The cells in a vertical rule column (highlighted in blue) specify the Condition rows which must be "satisfied" (determined by the cell values in quadrant 3) necessary to execute or "fire" the Action rows (determined by the cell values in quadrant 4).

Conditions		0	1	2
a	Cargo.weight	-	150000 .. 200000	-
b	Cargo.volume	-	-	< 300
c				
d				
e				

Actions		0	1	2
A	Cargo.packaging		Container	Pallet
B				
C				
D				
E				

Conditions (quadrant 1) and **Actions** (quadrant 2) are organized in the decision table shown above. Sets of Conditions and Actions are tied together to form rules by the vertical columns spanning the two right quadrants (3 & 4) of this section.

The cells in a vertical rule column (highlighted in **blue**) specify the Condition rows which must be "satisfied" (determined by the cell values in quadrant 3) necessary to execute or "fire" the Action rows (determined by the cell values in quadrant 4).

The illustration above contains the model of the sample business rule: *if* a cargo's weight is not between 150000 and 200000, and that cargo's volume is less than 300, then assign its packaging a value of `Pallet`.

Rule statements window

The **Rule Statements** window displays a list of natural language statements which describe the rules modeled in the *Rulesheet*. The Rule Statements window has several columns:

The screenshot shows the 'Rule Statements' window with a table of natural language statements. The table has columns: Ref, ID, Post, Alias, and Text. The first row (Ref 1) describes a rule for cargo weighing between 150,000 and 200,000 pounds must be packaged in a container. The second row (Ref 2) describes a rule for cargo with a total volume less than 300 cubic yards must be packaged on a pallet.

Ref	ID	Post	Alias	Text
1		Info	Cargo	Cargo weighing between 150,000 and 200,000 pounds must be packaged in a container
2		Info	Cargo	Cargo with a total volume less than 300 cubic yards must be packaged on a pallet

Rule Statements serve as documentation or elaboration on the meaning and intent of business rules modeled in the *Rulesheet*. Linking the plain-language *description* of a business rule in the Rule Statement section to its formal *model* in the other sections can provide important insight into rule operation during testing and deployment. Here, rule statement #2 is an informal description of the rule logic modeled in the *Rulesheet* column #2, as shown above.

When a Rule Statement row is clicked, the corresponding Columns or Action rows will highlight in **orange** in the *Rulesheet*. When a Rulesheet column is selected, the corresponding Rule Statement will highlight in **orange**.

Ref

This field is mandatory when linking Rule Statements to their *Rulesheet* columns (the rule models).

Rule Statements have a many-to-many relationship with *Rulesheet* columns. In other words, a Rule Statement may be reused for multiple Columns, and multiple Rule Statements may be expressed for a single Column. Entries in the **Ref** column serve to establish the relationship between the Rule Statements and *Rulesheet* columns. The various options are summarized below:

Ref	The Rule Statement is linked or connected to:
1	Column 1
1:3	Columns 1,2 and 3
{ 1, 3 }	Columns 1 and 3
0	Column 0
A0	Action Row A in Column 0
B1	Action Row B in Column 1
C1	Action Row C in Column 1
A1:B2	Action Rows A and B in Columns 1 and 2
{ A1, B2 }	Action Row A in Column 1 and Action Row B in Column 2
1:B2	invalid
A:2	invalid

When a colon (:) character is used to indicate that a range of Action cells is linked to the Rule Statement, then the left-hand and right-hand sides of the : must have the same form: both [column][row], both [column], or both [row] values. When they do not, the values will turn **red**, as shown in the last two rows above.

ID

Allows you to link Rule Statements to external source documentation using a code or ID number. This column is optional.

Post

This field is mandatory if you want the Rule Statement to appear as a Rule Message during *Rulesheet* execution (called "posting" the Rule Statement). Three "severity" levels are available: **Info**, **Warning**, and **Violation**. These severity levels have no intrinsic meaning - you can use them however you want. In a Corticon Studio *Ruletest*, Rule Messages with Info severity are color-coded **green**, Warnings **yellow**, and Violations **red**.

Alias

This field is mandatory if you want the Rule Statement posted during *Rulesheet* execution. All posted Rule Messages must be "attached" or "linked" to a Vocabulary entity. The choice of entity to "post to" is usually based on the entity being tested or acted upon in the associated rule.

Text

Technically, this field is optional, but posting a Rule Statement with no text results in an empty Rule Message. In order to have a meaningful posted Rule Message, we recommend entering plain language business rule statement in this field. Even when you do not plan to post messages during Rulesheet execution, creating a clear, concise version of the rule model is considered a best practice in rule modeling.

Rule Name

Allows you to assign custom names to the Rule Statements and the rule models they link to. This field is optional.

Rule Link

When a rule model has external documentation, you can enter an absolute path to a file on your local system to which it can link.. This field is optional.

Source Name

Allows you to reference the name of the source material the rule from which it originates. This field is optional.

Source Link

When a rule model has external documentation, you can enter an absolute path to a file on your local system to which it can link. This field is optional.

Category

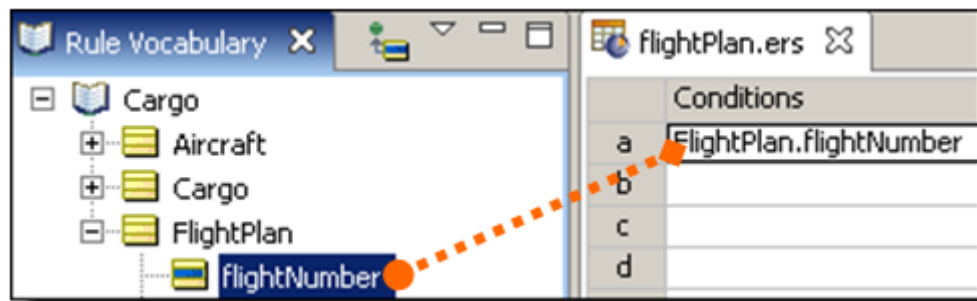
Allows you to assign a category name to each Rule Statement and the rule model to which it links. This field is optional.

Comments

Allows you to enter any comments for this Rule Statement and rule model to which it links. This field is optional.

Using the business vocabulary to build rules

Select the **Rule Vocabulary** tab; click the  icon beside an entity to expand the node & view its tree structure. Then, drag the specified term and drop it onto the *Rulesheet*.



Important: You can use either the **TAB** key to move from cell to cell within a row or the **ARROW** keys to move between rows within the *Rulesheet* grid. To move to another section in the *Rulesheet*, click on a cell within that section.

Important: Dragging and dropping is shown in this Guide as a dotted orange line, with an orange circle indicating the drag *origin*, and an orange diamond indicating the drag *destination*.

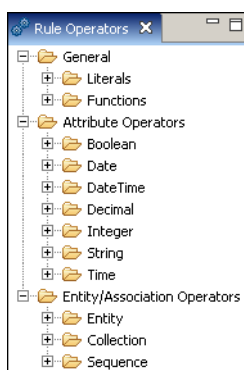
Using the operator vocabulary to build rules

Literal Terms, Functions & Operations

Corticon Studio's built-in library of operators is located (in the default Rule Modeling perspective) in the **Rule Operators** window in the lower left-hand corner of the Corticon Studio window.

If this window tab is not visible, select **Window > Show View > Rule Operators** from the Corticon Studio menubar.

Not all operators may be used in all types of rules or in all parts of the *Rulesheet* – refer to the *Rule Language Guide* for complete details.



1. Click the **General** or **Attribute Operators** folders, or click the plus sign beside a category, to expand them.
2. Select the operator you want, and then drag and drop it onto the *Rulesheet*.

Naming Rulesheets

Rulesheets may be named when created and renamed by:

Clicking **File > Save As...**

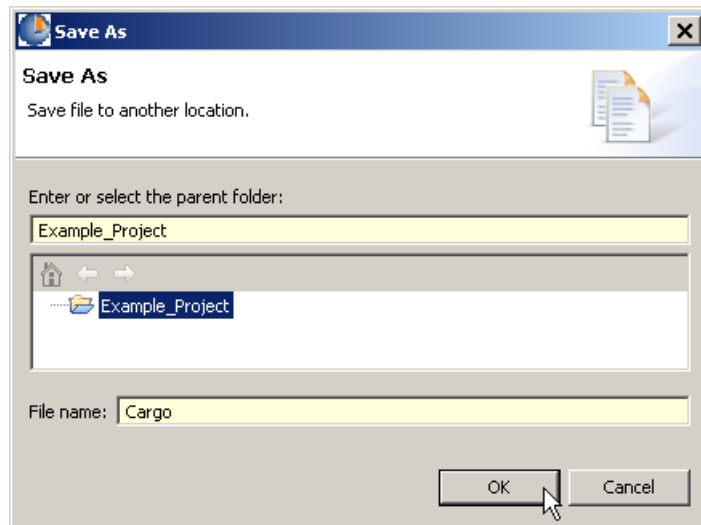
Deleting Rulesheets

Deleting *Rulesheets* is performed using one of the following methods:

- Right-click the *Rulesheet* in the **Project Explorer** window and select **Delete**  from the pop-up menu
- Select the *Rulesheet* in the **Project Explorer** window and hit the **Delete** key.

Saving a new Rulesheet

1. Save the *Rulesheet* using the menubar or toolbar as described previously.
2. Navigate to the **Project** folder where you want to store the *Rulesheet*.
3. The same file naming conventions apply to *Rulesheets* as apply to Vocabularies. See [File Naming Restrictions](#).



Validating and optimizing a Rulesheet

Conflict, Completeness, and Logical Loop Checkers, as well as the Compress Rules feature, are collectively known as validation and optimization functions. These topics are addressed briefly in the *Corticon Studio Tutorial: Basic Rule Modeling*, and in more depth in the *Corticon Studio: Rule Modeling Guide*.

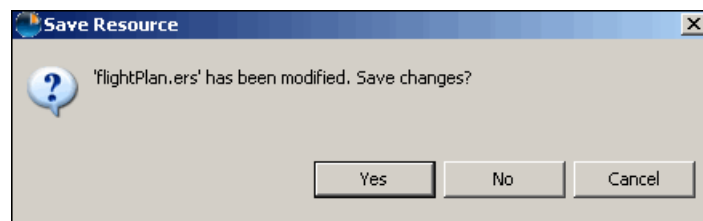
On very rare occasions, expanding, compressing or completing very large Rulesheets may cause Corticon Studio to run out of memory. If this occurs, try increasing Corticon Studio's memory allotment by modifying the shortcut you used to launch Corticon Studio. Details on this procedure are included in the *Corticon Studio: Installation Guide*, "Changing Corticon Studio Memory Allocation" section.

Creating a Rulesheet report

1. Choose **Rulesheet > Report** from the menubar.
2. If your local machine has a web browser installed, the HTML report should open as a new web page.
3. Corticon Studio will save the report file to `/Users/<your username>/AppData/Local/Temp/` using the name format `Corticon Rulesheet XXXXX.html`, where `XXXXX` is a unique auto-generated number. You can save the HTML to a different name or location from the browser.
4. For information about creating custom reports or saving them to custom locations, see the "Corticon Studio Reporting Framework" chapter of the *Rule Modeling Guide*.

Closing a Rulesheet

1. Close the *Rulesheet* using  on its tab or the **File > Close** in the menubar.
2. Choose **Yes** to save any changes and close the file.



Saving a modified Rulesheet

When you save a modified *Rulesheet*, the Corticon Studio automatically saves it to the current *Rulesheet* name. To save the *Rulesheet* to another name:

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

Ruleflows

A *Ruleflow* is a way of aggregating and organizing *Rulesheets* into a single unit of automated decision-making. It is a "flow diagram" depicting a set of one or more *Rulesheets* that have been validated using tools introduced earlier in this manual.

A *Ruleflow* may be assembled from as many *Rulesheets* as you want, provided that all the *Rulesheets* use the same Vocabulary file. In other words, a *Ruleflow* can have only one associated Vocabulary.

Following test and validation, a *Ruleflow* may be deployed into a production environment (described in the *Server Integration & Deployment Guide*). Once deployed and available to other IT systems, we refer to a *Ruleflow* as a **Decision Service**.

Important: In Corticon Studio, an individual *Rulesheet* may be tested using a *Ruletest*. But in order to deploy a *Rulesheet* to *Corticon Server*, it must be "packaged" as a *Ruleflow*. Only *Ruleflows* are deployable to *Corticon Server* and invocable as Decision Services.

The *Ruleflow* concept, introduced in Corticon Studio 5.3, is intended to facilitate *Rulesheet* reuse - any *Rulesheet* can be included in as many *Ruleflows* as you want. In earlier versions of Corticon Studio, reusing a *Rulesheet* required cutting and pasting, or recreating it, into multiple Rule Set files.

Ruleflow files have the extension `.erf`.

For details, see the following topics:

- [Ruleflow window](#)
- [Ruleflow properties tab](#)

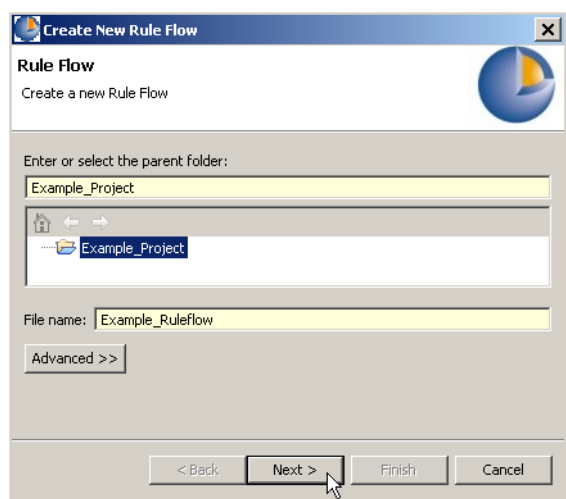
Ruleflow window

Opening a *Ruleflow* or making a *Ruleflow* the active Corticon Studio window causes the Corticon Studio menubar and toolbar to display the tools needed to begin working with *Ruleflows*.

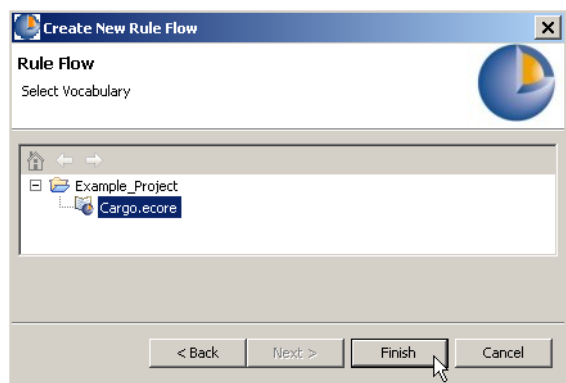
Creating a new Ruleflow

Creating *Ruleflows* is a simple process described in this section. Follow these steps to create a new *Ruleflow*.

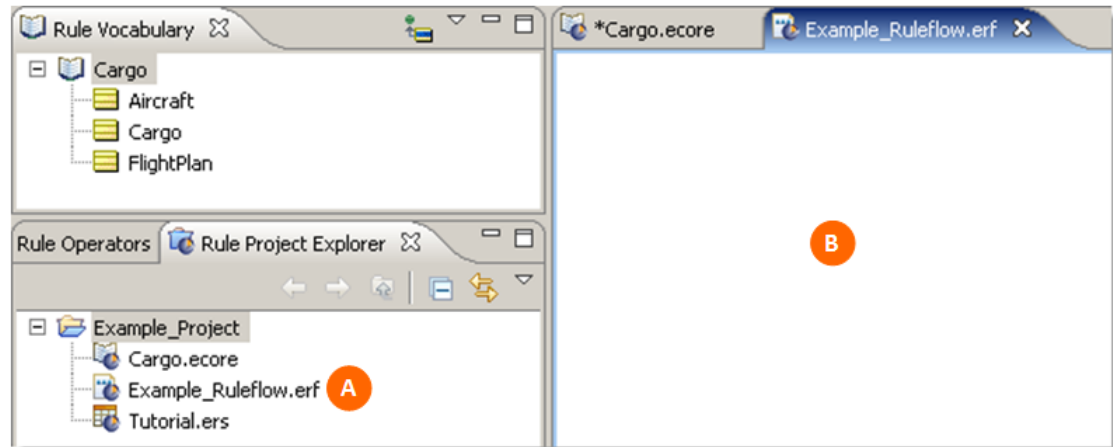
1. Choose **File > New > Ruleflow** from the menubar or click the down arrow next to the **New** icon on the toolbar and select **Ruleflow**. Either method will launch the **Create a New Ruleflow** wizard.



2. Highlight the Project you would like to associate the new *Ruleflow* with on the **Project** list (or enter it manually in the **Enter or select parent folder:** field).
3. Enter a file name for the *Ruleflow* in the **File name:** field. Click **Next >** to continue.



4. Select the Vocabulary to associate the *Ruleflow* to. A *Ruleflow* can only be associated with one Vocabulary. Here, we simply expand the *Example_Project* folder and highlight the *Cargo.ecore* file, which was created earlier in the [Creating a Vocabulary](#) section. As you create additional Vocabulary files within a Project they become available for selection from this list for use with this or other *Ruleflows* that you create.



5. Your new *Ruleflow* is now displayed in the **Rule Project Explorer** window (A) and in a new *Ruleflow* window with tab of the given name (B).

A *Rulesheet* menubar and toolbar replace the Vocabulary menubar and toolbar and you are now ready to begin modeling your rules.

Naming Ruleflows

As noted earlier, you name your *Ruleflows* during the *Ruleflow* creation process. As always, *Ruleflow* names, just like any other Corticon Studio, must adhere to and comply with the guidelines shown in the [File Naming Restrictions](#) section of this document.

Adding Ruleflows

To add additional *Ruleflows* to a Rule Project

Select **File>New > Ruleflow** from the menubar and repeat the steps recounted earlier in the [Ruleflow creation](#) process.

Alternatively, you can select **New > Ruleflow** from the toolbar.

Deleting Ruleflows

- A *Ruleflow* can be deleted by:
 - highlighting it in the **Project Explorer** file list and selecting **Edit > Delete** from the menubar
 - right-clicking the file in the **Project Explorer** window and selecting **Delete** from the pop-up menu.

Saving a new Ruleflow

- Select **File > Save** from the menubar or click the **Save** icon  on the toolbar to save a new or modified *Ruleflow*.

Refer to the [Ruleflow menubar](#) section for details regarding saving, or renaming, a *Ruleflow* with a different name or to a different file location.

Ruleflow menu commands

The following menubar options are available when a Ruleflow window is the active window, as shown above.

Note: This is an an additional menu. Other menus are unchanged.

Ruleflow Menu

The following actions are accessible only when the active file is a ruleflow (.erf) file:

- **Properties** - Opens the Ruleflow properties window.
- **Report** - reates an HTML report and launches your browser for viewing. See [Creating a Ruleflow Report](#).

Ruleflow toolbar

A few special toolbar icons, shown below inside orange boxes, are inserted into the standard Corticon Studio toolbar when a *Ruleflow* is the active window.

Many of the functions provided by the toolbar are also accessible through the menubar.



	Opens the <i>Ruleflow</i> properties window, usually at the bottom of the Corticon Studio window
	Zooms the <i>Ruleflow</i> window to magnification selected

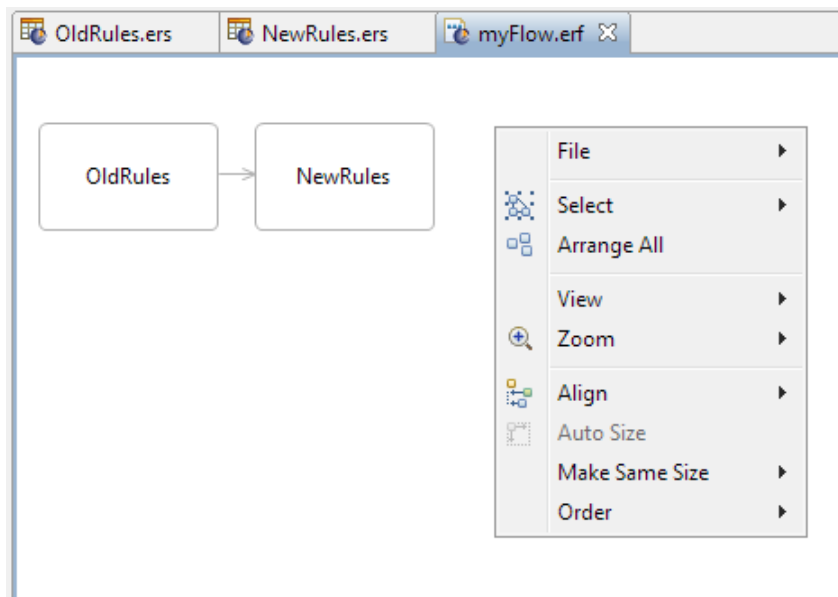
Context-sensitive right-click pop-up menus in Ruleflows

Ruleflow Tab Pop-up Menu Options

Note: The *Ruleflow* tab pop-up menu opens when a Ruleflow's tab is right-clicked. This menu provides quick access to many of the most frequently used generic, window-level functions. It has the same options as the other tab pop-ups, including the *Rulesheet* and *Ruletest*.

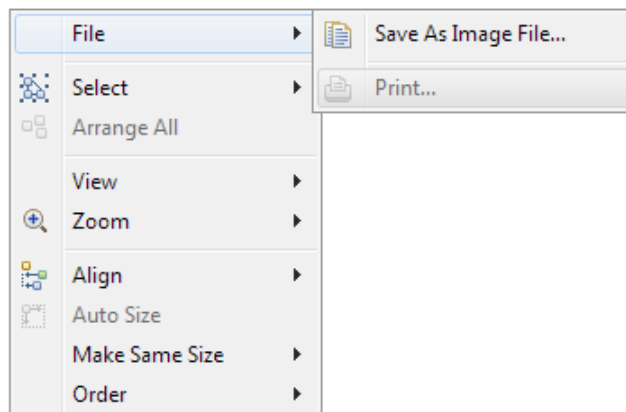
Ruleflow Window Pop-up Menu Options

The Ruleflow Window Pop-up Menu opens when you right-click within the Ruleflow window. This menu provides quick access to many of the most frequently used Ruleflow-specific functions.

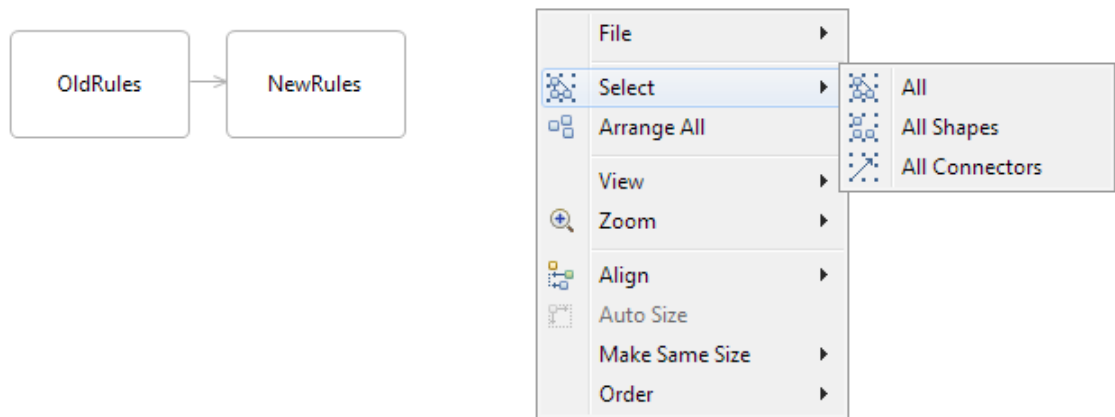


The following are the menu and submenu items (with descriptions where not apparent). Items with adjacent objects are object-oriented, others are page oriented:

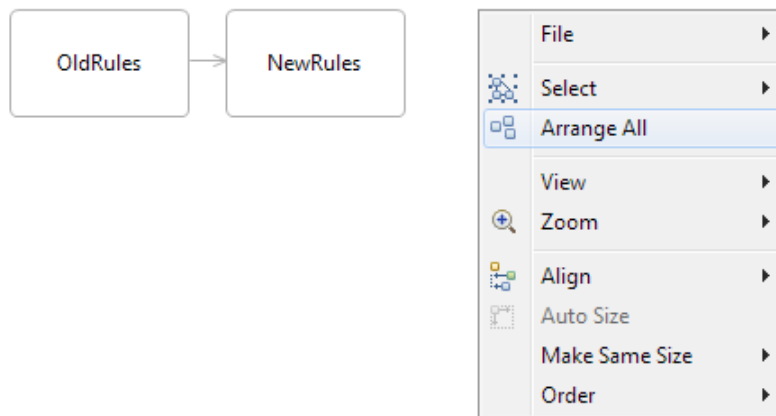
File



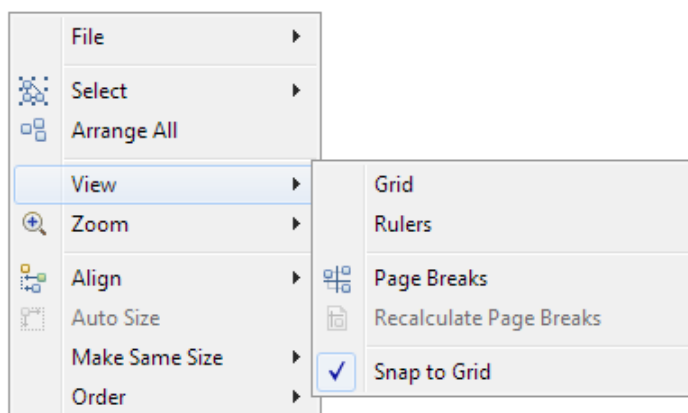
Select



Arrange All



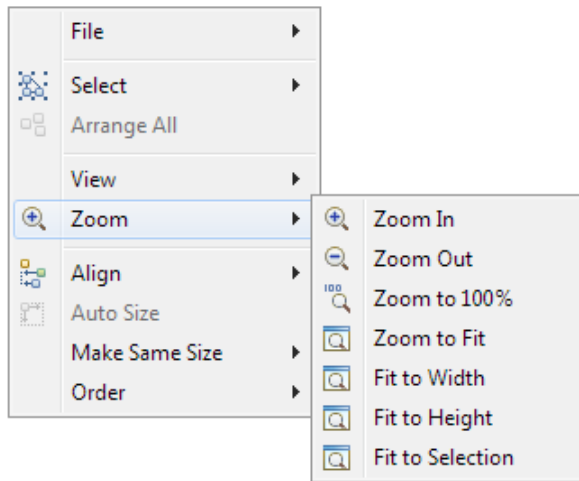
View



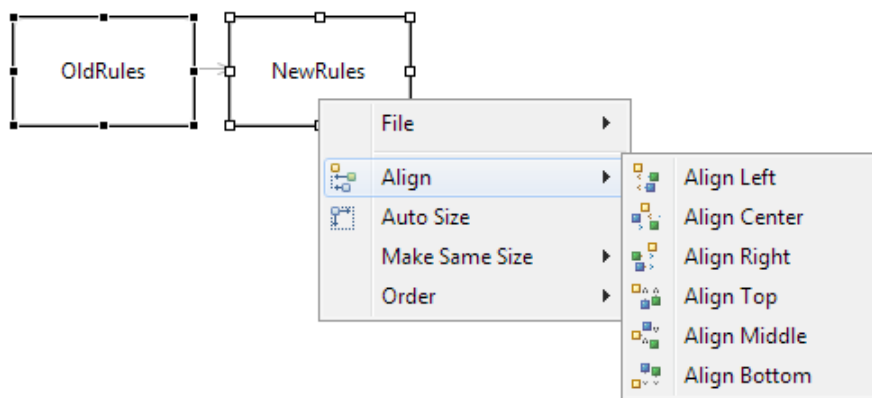
- **Grid** displays a light gray grid overlay in the Ruleflow window
- **Ruler** displays rulers on the X and Y axes of the Ruleflow window
- **Page Breaks** displays breaks in the Ruleflow window when the window is larger than the size of the page

- **Recalculate Page Breaks** updates breaks when changes are made to the Ruleflow window
- **Snap to Grid** assists in aligning Ruleflow shapes to the overlay grid (whether visible or not)

Zoom



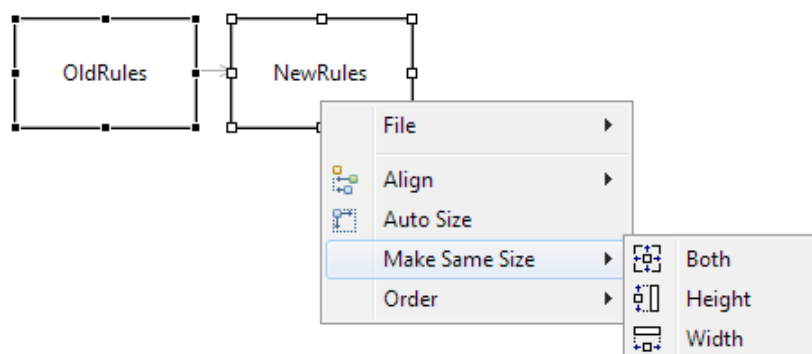
Align



Auto Size

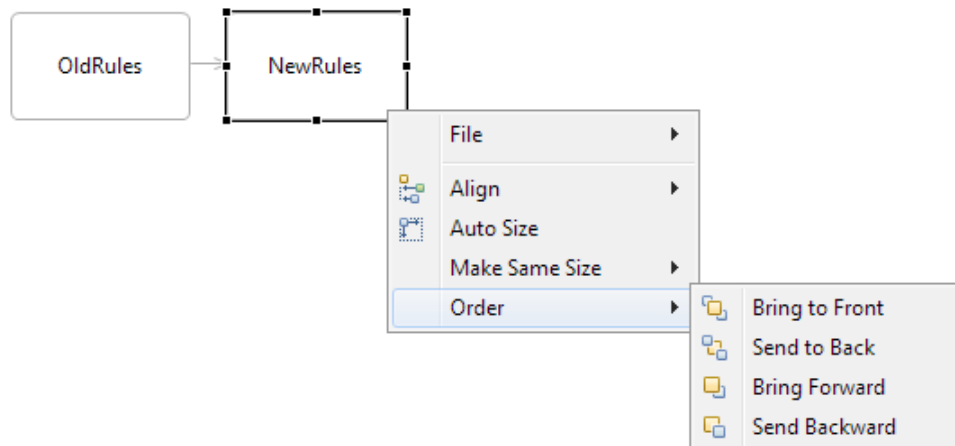
Resizes the selected object if it is not the preferred size.

Make Same Size



- **Both** resizes selected Rulesheet block icons to be the same size
- **Height** resizes selected Rulesheet block icons to be the same height
- **Width** resizes selected Rulesheet block icons to be the same width

Order



Ruleflow window

Ruleflow diagrams are displayed within the *Ruleflow* window. The objects that comprise or "populate" a *Ruleflow* window include:

Rulesheets

Rulesheets are the core objects in a *Ruleflow*. By arranging and organizing *Rulesheets* in the *Ruleflow*, you can specify execution sequence of the *Rulesheets* and offer control of the iteration of the *Rulesheets*.

Ruleflows serve as a kind of deployable "container" for *Rulesheets*. *Rulesheets* remain the basic unit of decision making, but they become much more reusable when combined in different ways in different *Ruleflows*.

For example, *Rulesheet sample.ers* can be included in many *Ruleflows*, each of which may have its own use in different business processes or applications. This is a "service-oriented approach" (SOA) to using *Rulesheets*, which is compatible with the SOA design practices commonly in use in modern IT system development.

Rulesheets may be designated for iteration by clicking the iteration icon which appears when you hover the mouse cursor over a *Rulesheet* object in the *Ruleflow* window.

Connectors

Connectors are the objects that serve to connect or "stitch" *Rulesheets* together and control their sequence of execution. If a connector is drawn *from* *Rulesheet sample1.ers* *to* *sample2.ers*, then when a deployed *Ruleflow* is invoked, it will execute the rules in *sample1.ers* first, followed by the rules in *sample2.ers*.

Subflows

Subflows provide yet another level of reusability of *Ruleflow* objects. A Subflow may contain many *Rulesheets* and connectors - it essentially becomes a "package" of these objects that can be copied and pasted and re-used within the *Ruleflow*.

Subflows may also be designated for iteration, so that all the objects in the Subflow are re-executed until the values derived by their constituent rules cease changing.

Service Call-Outs

Service Call-outs are a type of extension to a Ruleflow, and require significant Java development expertise. The method for building Service Call-outs is documented in detail in the *Extensions User Guide*.

Service Call-outs are enabled by your *Corticon* license. If you do not see the Service Call-out icon in your Corticon Studio's *Ruleflow* editor palette (see below), then your license does not enable Service Call-outs.

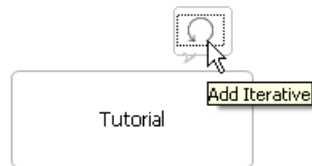
Iteration

Iteration may be enabled for *Rulesheet* and Subflow objects in a *Ruleflow*. When an object is set to iterate, it will repeatedly re-execute until the values derived by the object's rules cease to change. Once values in the object cease changing, the iteration stops and execution continues to the next (as determined by the Connectors) object, which may be either another *Rulesheet* or Subflow.

Enabling and disabling iteration for Rulesheets

To enable iteration for *Rulesheets*:

1. Click on the Rulesheet you want to have iterate.
2. Drag the **Iterative** icon from the palette and drop it on the Rulesheet block.



3. The iterative symbol attaches to the Rulesheet, as shown:



4. To disable iteration, click on the iterative icon within a Rulesheet, and then press the **Delete** key.

Enabling and disabling iteration for Subflows

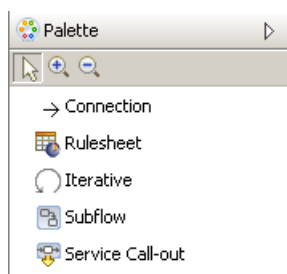
To enable iteration for Subflows:

1. Hover your pointer over the Subflow block's title section. You will see a circular icon appear in a "call-out" pop-up window. Move your mouse to click the circular icon, and the icon will open near the bottom of the Subflow block icon.
2. To disable iteration, click the circular icon and press the **Delete** key.

Important: Logical loop processing options within a *Rulesheet* do not result in a circular icon appearing on the *Rulesheet* object within the *Ruleflow*.

Ruleflow tools palette

The far right-hand column of the *Ruleflow* window contains the *Ruleflow* tools palette. The *Ruleflow* tools palette contains five tools:



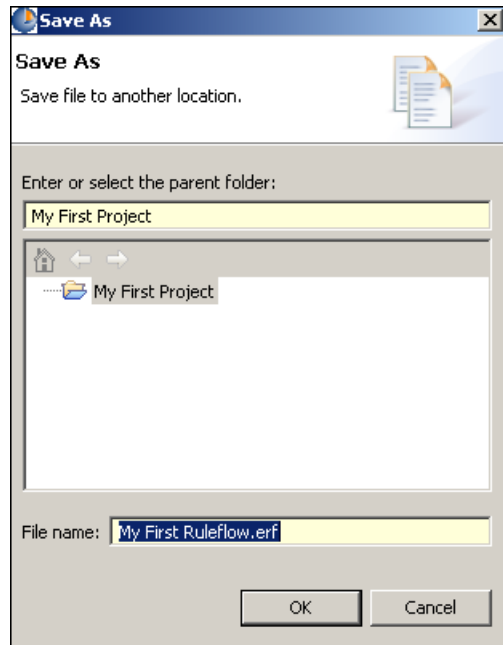
- **Select** – Allows you to select one or more diagram objects. You can select multiple objects by click-hold-dragging to surround the objects with the box that appears.
- **Zoom** – Allows you to zoom in or out on a portion of the *Ruleflow* window.
- **Connection** – Creates a flow link between two *Rulesheets* in a sequence. Select this tool and then use your mouse to drag a line from one *Rulesheet* rectangle to another, then release.
- **Rulesheet** – Adds a *Rulesheet* to the *Ruleflow* window. Select this tool and use your mouse to drag a rectangular shape in the *Ruleflow* window.
- **Iterative** – Allows you to designate individual or multiple *Rulesheets* or Subflows as iterative.
- **Subflow** – Allows you to create Subflows within a *Ruleflow* window. Use your mouse to drag a rectangular shape in the *Ruleflow* window. If you want to include *Rulesheets* in your Subflow, be sure to drag and drop the *Rulesheets* into the Subflow rectangle. Dragging a new Subflow rectangle to surround existing *Rulesheets* will **not** include them in the Subflow.
- **Service Call-out** – Service Call-outs are a type of extension to a *Ruleflow*, and require significant Java development expertise. The method for building Service Call-outs is documented in detail in the *Extensions User Guide*.

Service Call-outs are enabled by your *Corticon* license. If you do not see the Service Call-out icon in your Corticon Studio's *Ruleflow* editor palette, then your license does not enable Service Call-outs.

Once a tool has been selected, it remains selected until you perform the action. To perform the same action again, re-select the tool.

Renaming a Ruleflow and/or saving a Ruleflow to a different location

Selecting **File > Save As...** from the menubar displays the **Save As** dialog, allowing you to assign the *Ruleflow* a different name or save it to another location.



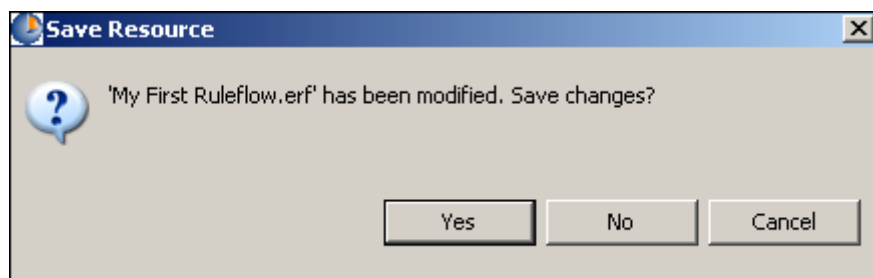
1. To rename the *Ruleflow*, modify the name in the **File name:** field with the **Project** folder highlighted in the **Project** list.
2. To save the *Ruleflow* to a different location, whether you are renaming it or not, select that **Project** folder in the **Project** list or enter the parent folder name manually in the **Enter or select the parent folder:** text field.
3. The same file naming conventions apply to *Rulesheets* as apply to *Rulesheets*. See [File Naming Restrictions](#).

Creating a Ruleflow report

1. Choose **Ruleflow > Report** from the menubar.
2. If your local machine has a web browser installed, the HTML report should open as a new web page.
3. Corticon Studio will save the report file to `/Users/<your username>/AppData/Local/Temp` using the name format `CorticonRuleflowXXXXX.html`, where `XXXXX` is a unique auto-generated number. You can save the HTML to a different name or location from the browser.
4. For information about creating custom reports or saving them to custom locations, see the "Corticon Reporting Framework" chapter of the *Rule Modeling Guide*.

Closing a Ruleflow

1. Close a *Ruleflow* using the  on the *Ruleflow* tab, by right-clicking the tab and selecting **Close** from the pop-up menu or by selecting **File > Close** on the menubar.

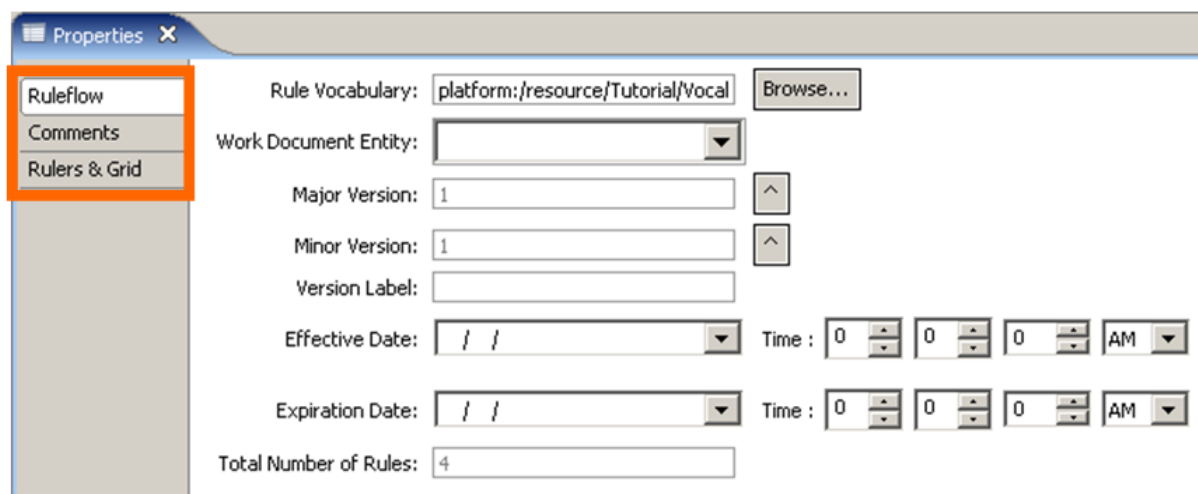


2. Choose **Yes** to save any changes if you have modified the *Ruleflow* in any way prior to closing it.

Ruleflow properties tab

Ruleflow properties are set in the **Properties** tab at the bottom of the Corticon Studio window when a *Ruleflow* is the active file. This window is comprised of several "sub-tabs" arranged vertically at the left-hand side of the window. These sub-tabs are shown below in the orange box.

Figure 5: Assigning a Version Number to a Ruleflow



Ruleflow sub-tab

Rule Vocabulary

Use this field to change the Vocabulary referenced by this *Ruleflow*. Notice that the location of the Rule Vocabulary associated with the *Ruleflow* is automatically entered for you in the **Rule Vocabulary** field. You can change this value by selecting **Browse** and choosing a different Vocabulary, if necessary.

Work Document Entity

The **Work Document Entity** drop-down menu allows you to choose from among the existing entities in your Vocabulary and designate one of them the Work Document Entity.

The Work Document Entity provides the root for XML schemas generated in the Deployment Console. Rule Modelers generally need not be concerned with this setting, as it relates to the *Ruleflow* deployment on *Corticon Server*.

Major and Minor Version Numbers

Major Version numbers for *Ruleflows* are optional and are assigned manually. **Minor Version** numbers are automatically incremented each time a *Ruleflow* is saved. A Minor version number may also be incremented manually.

When we use different version numbers to label identically named *Ruleflows*, the *Server* keeps them straight in memory, and responds correctly to requests for the different (Major) versions. In other words, an application or process can use (or "call") different (Major) versions of the same *Ruleflow* simply by including the (Major) version number in the request message. The details of how this works at the *Server* level are technical in nature and are described in the *Server Integration & Deployment Guide*.

A plain-text description of this version can be added in the *Ruleflow* Version Label field, immediately below the *Ruleflow* **Minor Version** field. Version numbers may be manually raised anytime but never lowered.

Version Label

Optional. Give the version a unique text label.

Effective and Expiration Dates

Effective and Expiration dates are optional for *Ruleflows* and can be assigned singly or in pairs. When we use different Effective and Expiration dates to describe identically named *Ruleflows*, the *Server* keeps them straight in memory, and responds correctly to requests for the different dates. In other words, an application or process can use different versions of the same *Ruleflow* depending on date criteria. The details of how this works at the *Server* level are technical in nature and are described in the *Server Integration & Deployment Guide*.

Effective and Expiration Dates may be assigned using the same window as above. Clicking on the **Effective Date** or **Expiration Date** drop-down displays a calendar:

Figure 6: Setting Effective and Expiration Dates

The screenshot shows the 'Properties' dialog box for a 'Ruleflow'. The 'Rule Vocabulary' is set to 'platform:/resource/Tutorial/Vocal'. The 'Work Document Entity' is a dropdown menu. The 'Major Version' is 2, 'Minor Version' is 1, and 'Version Label' is empty. The 'Effective Date' is a dropdown menu showing '/ /'. The 'Expiration Date' is a dropdown menu. The 'Total Number of Rules' is 4. A calendar pop-up is displayed for February 2009, with the 6th of the month highlighted.

Effective Date and Expiration Date Times are entered in their respective **Time** fields.

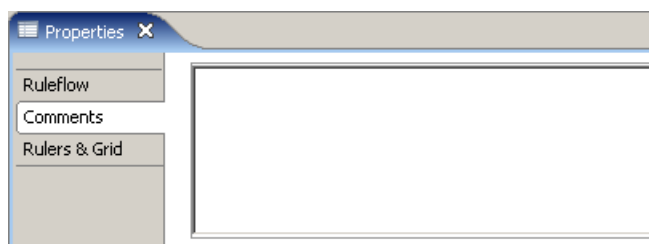
Total Number of Rules

Counts the total number of rules in all the *Rulesheets* included in this *Ruleflow*. For Column numbers 1 and higher, each *enabled* column is counted as one rule. For Column 0, each *enabled* Action row is counted as one rule.

Comments sub-tab

Selecting the **Comments** link within the **Properties** window allows you to add comments to your *Ruleflow*s, as shown below:

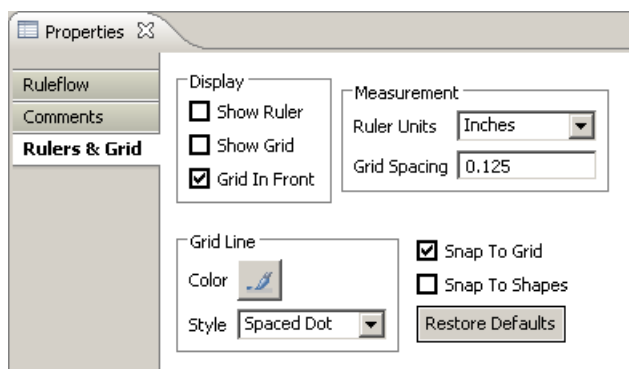
Figure 7: The Ruleflow Comments Sub-tab



Rulers & grid sub-tab

Selecting the **Rulers & Grid** sub-tab lets you display or hide the **Ruler** and **Grid** as well as format the color and style of the **Grid Line**. You can also adjust the unit of measure for the Ruler, the spacing within the Grid and **Snap** objects within the *Ruleflow* diagram to the Grid (or to align with other shapes). Clicking **Restore Defaults** will re-set these properties to the original default settings. The **Rulers & Grid** sub-tab is shown below:

Figure 8: The Ruleflow Rulers and Grid Sub-tab



Ruletests

For details, see the following topics:

- [Ruletest window](#)

Ruletest window

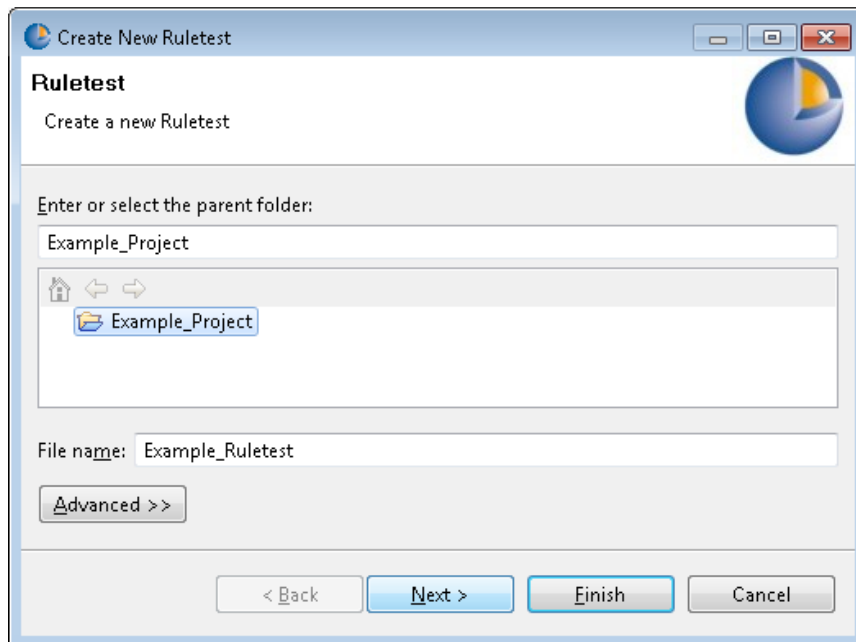
A *Ruletest* is the mechanism within Corticon Studio for creating use cases or test scenarios of sample data and sending them to a *Rulesheet* or *Ruleflow* for processing. Ruletests consist of one or more Testsheets which test independent *Rulesheets* or *Ruleflows*, or can be linked together to test a succession of *Rulesheets* or *Ruleflows* to simulate a process sequence.

Opening a *Ruletest* or making a *Ruletest* the active Corticon Studio window causes the Studio menubar and toolbar to display the tools needed to test *Rulesheets* and *Ruleflow*.

Rulesheets, as individual files, may only be tested using a Corticon Studio *Ruletest*. In order to deploy and test using *Corticon Server*, *Rulesheets* must be packaged as *Ruleflows* before they can be executed. *Ruleflows* may be tested both in Corticon Studio using *Ruletests* and on *Server* using standard request messages.

Creating a new Ruletest

1. Choose **File > New > Ruletest** from the Corticon Studio menubar, or click  from the toolbar, to display a new **Create New Ruletest** dialog box.

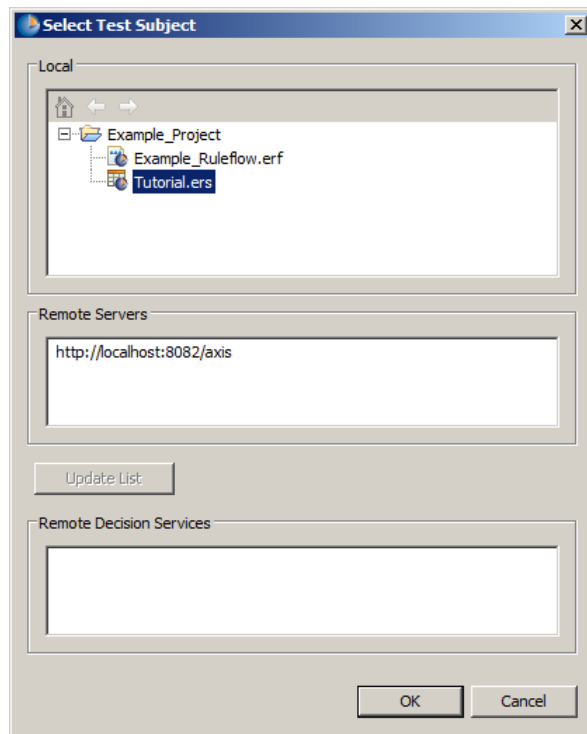


2. Select the **parent folder** for the new *Ruletest*, in our example `Example_Project`. This will associate the *Ruletest* with the `Example_Project` project and display it in the **Project Explorer** window along with the other files in that project.
3. Enter a file name for the new *Ruletest*, in our example `Example_Ruletest`.
4. Click **Next >** to continue and display the **Select Test Subject** dialog box, shown below. Here, you select the *Rulesheet* or *Ruleflow* that is to be tested. You will be presented with 2 lists of available test subjects, both **Local** and **Remote**.
 - The Local test subjects are the *Rulesheets* and *Ruleflows* available in your active Project.
 - The Remote section contains a list of remote Corticon *Server* instances that Corticon Studio 'knows'.

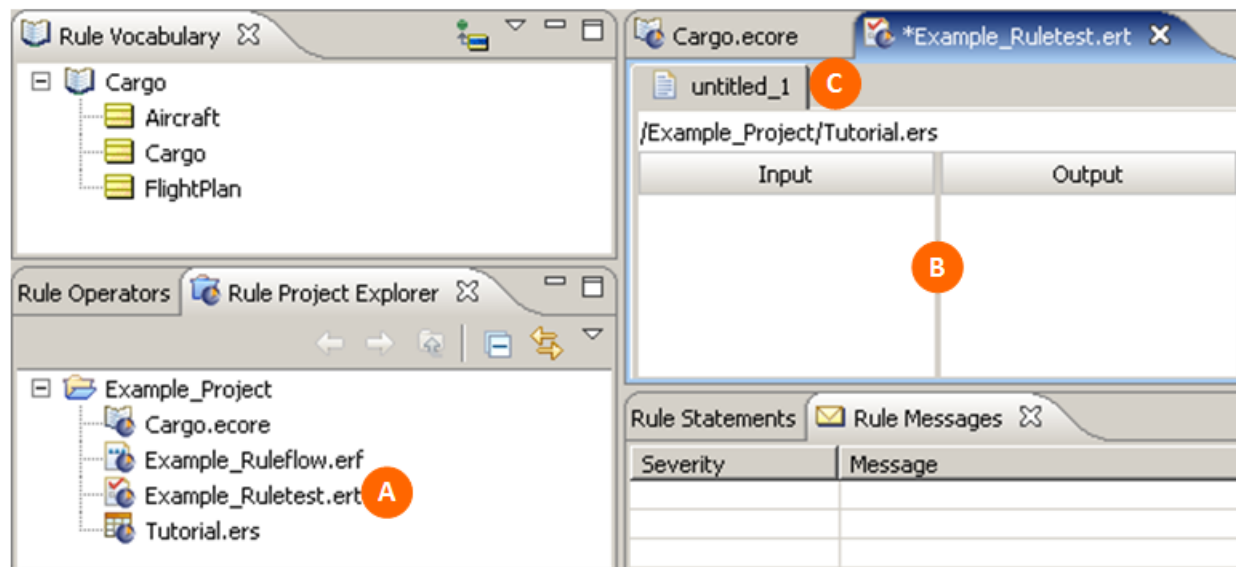
Note: The list of remote services Corticon Studio 'knows' is configured in Corticon Studio's properties. See the `CcDeployment.properties` section of the *Server Integration & Deployment Guide* for more details.

If these remote *Servers* are running, and if Corticon Studio has network access to them, then clicking **Update list** will tell these remote *Servers* to report which *Ruleflows* are currently deployed to them and available for execution. You can select one from the list.

5. In the screenshot below, we chose local *Rulesheet* `Tutorial.ers`.



6. Click **Finish** to display your new *Ruletest* in Corticon Studio:



7. The new *Ruletest* should appear as a new node in the **Project Explorer** tree (A), a new window with the chosen *Ruletest* name as its tab name (B). The new *Ruletest* will also have a Testsheet tab (C). A Testsheet tab is divided into 3 vertical "panels": **Input**, **Output**, and **Expected**. In the graphic above, **Expected** is not shown.
8. You can easily change the *Ruletest*'s test subject by double-clicking on the current test subject, immediately below the Testsheet tab (C). This will re-open the **Select Test Subject** window, where you can change your selection.

Ruletest menu commands

The following menu commands are available when a Ruletest window is the active window, as shown above.

Note: This is an additional menu. Other menus are unchanged.

Ruletest Menu

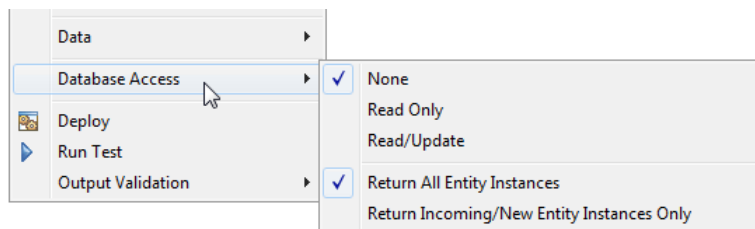
The following actions are accessible only when the active file is a ruletest (`.ert`) file:

- **Testsheet > Add Testsheet** - Inserts a new Testsheet.
- **Testsheet > Remove Testsheet** - Deletes the specified Testsheet.
- **Testsheet > Link To Previous Testsheet** - Causes the Input panel of the second Testsheet to be populated with the data from the Output panel of the first.
- **Testsheet > Change Test Subject** - opens a window that lets you select a new *Rulesheet* or *Ruleflow* to test.
- **Testsheet > Cut Testsheet** - Cut the active Testsheet.
- **Testsheet > Copy Testsheet** - Copy the active Testsheet.
- **Testsheet > Paste Testsheet** - Paste the active Testsheet.
- **Testsheet > Rename Testsheet** - Opens an entry window to change the Testsheet name.
- **Testsheet > Move Backward** - Moves the selected Testsheet tab one tab towards the beginning of the *Ruletest*.
- **Testsheet > Move Forward** - Moves the selected Testsheet tab one tab towards the end of the *Ruletest*.
- **Testsheet > Move To Beginning** - Moves the selected Testsheet tab directly to the start of the *Ruletest*.
- **Testsheet > Move To End** - Moves the selected Testsheet tab directly to the end of the *Ruletest*.
- **Testsheet > Import XML/SOAP** - Import a valid CorticonRequest XML document into Corticon Studio as a *Ruletest*.
- **Testsheet > Data > Set to Null** - Resets the selected Testsheet tree node to null.
- **Testsheet > Data > Go to Entity** - Displays an entity when an association tree node is selected.
- **Testsheet > Data > Sort Entities** - Sorts entity nodes alphabetically by name. One entity must be selected.
- **Testsheet > Data > Properties** - Displays the *Ruletest* Properties window.
- **Testsheet > Data > Input > Export Request XML** - Exports the active Testsheet's Input pane as a *CorticonRequest* XML document.
- **Testsheet > Data > Input > Export Request SOAP** - Exports the active Testsheet's Input pane as a *CorticonRequest* XML document with SOAP envelope.
- **Testsheet > Data > Input > Generate Data Tree** - Constructs the minimum Input data structure necessary to test the chosen Rulesheet. Uses the Rulesheet's scope for guidance.

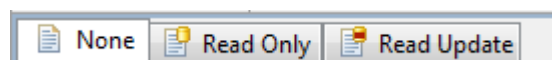
- **Testsheet > Data > Output > Export Response XML** - Exports the active Testsheet's Output pane as a *CorticonResponse* XML document.
- **Testsheet > Data > Output > Export Response SOAP** - Exports the active Testsheet's Output pane as a *CorticonResponse* XML document with SOAP envelope.
- **Testsheet > Data > Output > Copy to Expected** - Copies the data in the Output panel to the Expected panel.
- **Testsheet > Data > Expected > Export Response XML** - Exports the active Testsheet's Expected pane as a *CorticonResponse* XML document.
- **Testsheet > Data > Expected > Export Response SOAP** - Exports the active Testsheet's Expected pane as a *CorticonResponse* XML document with SOAP envelope.
- **Testsheet > Database Access** - Available when in Integration & Deployment mode. See the following section for details.
- **Testsheet > Export WSDL** - Create a WSDL directly from the Input pane of a *Ruletest*. For more information about using WSDLs to integrate Decision Services, see *Server Integration & Deployment Guide*
- **Testsheet > Deploy** - Compiles the Ruletest target without executing it.
- **Testsheet > Run Test** - Compiles (if needed) and executes Ruletest.
- **Testsheet > Output Validation > Validate** - Reruns the color-coded validation of the Output and Expected data. See [Using the Expected Panel](#).
- **Run All Tests** - Executes all Testsheets in the *Ruletest*.
- **Report** - Creates an HTML report and launches your browser for viewing. See [Creating a Ruletest Report](#).

Database Access

When set to Integration & Deployment mode, the Enterprise Data Connector provides the following menu group:



Each testsheet can elect to use the Vocabulary's database connection to provide Read Only or Read/Update access to the database. The chosen option is indicated in the testsheet's tab as stylized here where the sheet name is appropriate to its database access selection:



The second option selection determines what is returned in response messages. It does not apply when Database Access is **None**

- **Return All Entity Instances** - Instructs Corticon Server to return all entities (queried during the course of rule execution) in the response message.

- **Return Incoming/New Entity Instances Only** - Instructs Corticon Server to return only entities which were directly used in the rules, present in the request message, and/or generated by the rules (if any).

See the *Integration and Deployment Guide* for more about these options.

Ruletest toolbar

When a Ruletest window is active, commands are added to the toolbar, as shown:

Many of the functions provided by the toolbar are also accessible on the Ruletest menu.



The Ruletest icons provide the same functions as the corresponding menu commands:



- **Ruletest > Testsheet > Add Testsheet**



- Ruletest > Testsheet > Remove Testsheet



-  **Ruletest > Testsheet > Link to Previous Testsheet**



- Ruletest > Testsheet > Change Test Subject



- **Ruletest > Testsheet > Cut Testsheet**



- **Ruletest > Testsheet > Copy Testsheet**



- **Ruletest > Testsheet > Paste Testsheet**

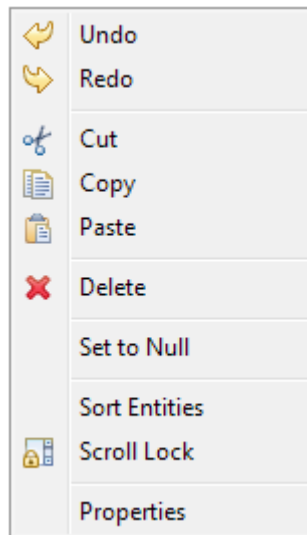


- Ruletest > Testsheet > Rename Testsheet

-  Ruletest > Testsheet > Move Backward
-  Ruletest > Testsheet > Move Forward
-  Ruletest > Testsheet > Move to Beginning
-  Ruletest > Testsheet > Move to End
-  Ruletest > Testsheet > Data > Output > Copy to Expected
-  Ruletest > Testsheet > Output Validation > Validate
-  *Pop-up menu in testsheet* > Scroll Lock
-  Ruletest > Testsheet > Deploy
-  Ruletest > Testsheet > Run Test

Context-sensitive right-click pop-up menus

When you right-click in a Testsheet, the *Ruletest* pop-up menu displays as shown:



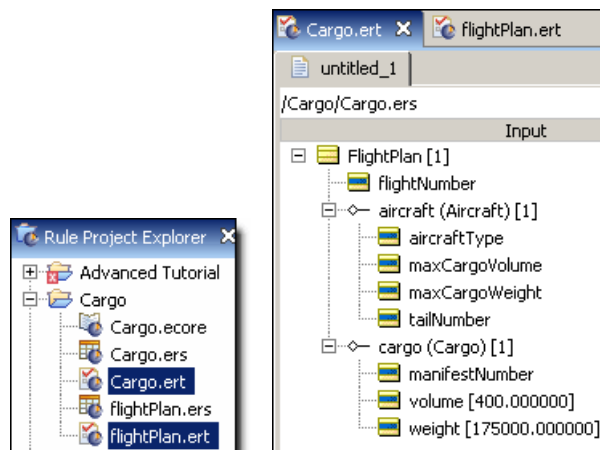
The commands in the Ruletest popup menu are:

Undo		Remove the last action.
Redo		Redo the last action that was undone. (exposed only after you have done Undo)
Cut		Copy and then remove the currently selected entity or attribute.
Copy		Copy the currently selected entity or attribute.
Paste		Paste the recently copied entity or attribute. (exposed only after you have done Cut or Copy)
Delete		Deletes the selected entity or attribute.
Set to Null		Resets the value of the selected node(s) to null.
Scroll Lock		Synchronizes scrolling of the three testsheet columns.
Go to Entity		Locates the parent entity of a selected collection element.

Sort Entities		Sorts entities alphabetically and by element ID number
Properties		Opens the <i>Ruletest's</i> Properties window. The <i>Ruletest</i> sub-tab displays the location of the Vocabulary associated with the <i>Ruletest</i> , the Testsheet sub-tab displays the location of the Rulesheet associated with the <i>Ruletest</i>

Testsheet tabs

A *Ruletest* is comprised of **Input**, **Output** and **Expected** panels. *Ruletests* you create are displayed in the **Rule Project Explorer** window. You can navigate between open Testsheets by clicking the Testsheet's tab to make it active.

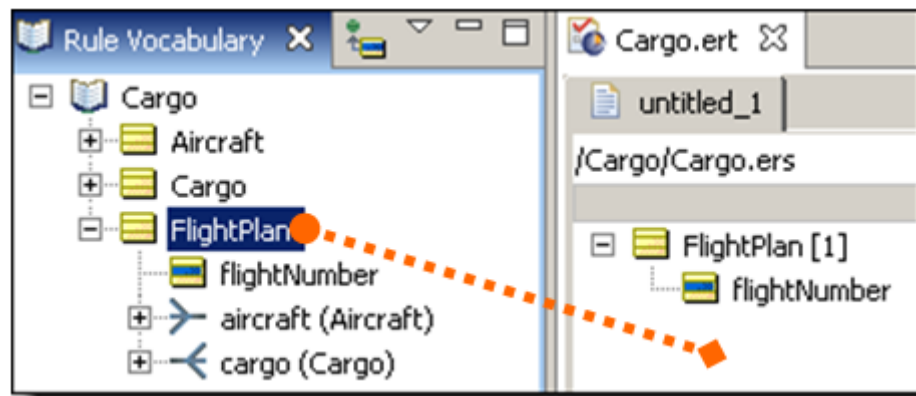


Populating the input panel

Creating a set of test data in the **Input** panel is also called creating the "test tree", since the Input data structure uses the same "nodes" as the **Rule Vocabulary** window and arranges them in the same "tree view".

Adding entities to the test tree

Drag and drop entity nodes from the **Rule Vocabulary** window onto the Testsheet as shown to the right. When creating new root-level entities, the nodes may be dropped anywhere on the Input panel of the Testsheet.



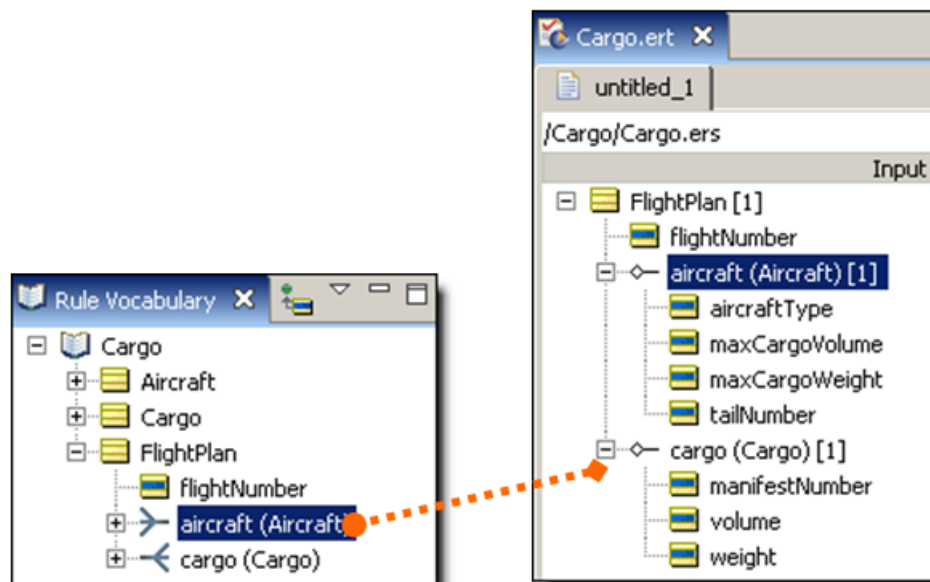
Important: Entities may be deleted by pressing the **Delete** key or by selecting **Delete** from the right-click pop-up menu.

Creating associations in the test tree

Creating an association adds an instance of the selected entity in the Testsheet, but this is different from creating an instance of a root-level entity by itself (without forming the association to the "parent" entity). An association creates a link between entities; if this did not exist, there would be no direct relationship between them.

To create an association:

1. Drag and drop the association for the specified "child" entity **directly on top of** the "parent" entity, as indicated by the orange line in this illustration, from the **Rule Vocabulary** window into the Input panel.

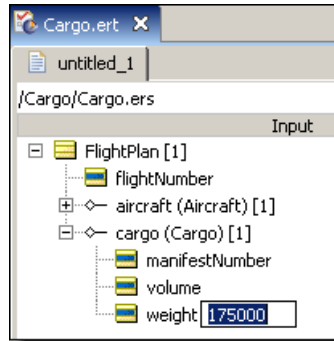


2. Verify the child entity has the appropriate "indented" structure underneath the parent entity, indicating the association has been created correctly.

Important: Be careful to follow precisely the orange line in these illustrations, which shows that the association must be dragged from the Vocabulary and dropped **directly onto** the yellow entity icon to which it is being associated. Dropping the association on an empty or incorrect portion of the Testsheet will produce a message informing you that the association request was ignored.

Assigning attribute values in the test tree

1. Double-clicking the term you want opens a data entry box



2. Do any one of the following:
 - Type test data in the box and press **Return** to enter the data
 - Press **Tab** to advance the entry box to the next attribute, and press **Return** to open its data entry box.

Important: Attributes not used in a scenario may be deleted by selecting them and pressing the **Delete** key or by selecting **Delete** from the right-click pop-up menu. Multiple attributes may be selected using **Shift-click** and **Ctrl-click** to select contiguous and non-contiguous groups, respectively. In the illustration above, the attributes of aircraft[1] have not been deleted, just collapsed and hidden from view.

Automatically generating a test tree

When creating a *Ruletest*, it is important to provide the input data required by the *Rulesheet* being tested. Without the necessary input data, the rules cannot create the expected output.

Creating a test tree manually can be tedious for very large Rulesheet test subjects. Corticon Studio provides a way to simplify and speed up the process.

Generate Test Tree is an option in Corticon Studio menu **Ruletest > Testsheet > Data > Input** that automatically analyzes the Vocabulary terms used by the *Rulesheet* and generates the corresponding tree structure in the Input pane of the Testsheet. The input of specific data *values* for the attributes still must be performed manually, but the tree *structure* is created automatically.

When a Vocabulary association between two entities is one-to-many, then the **Generate Test Tree** feature will create 2 instances (copies) in the tree. You can add more to test a larger collection, or delete them to test a smaller collection. You can always edit the resulting test tree by adding or removing terms.

Executing tests

Before a *Ruletest* can be executed, the corresponding *Rulesheets* (whether being tested directly or as contained in *Ruleflows*) must be compiled. This compilation process does not occur until the *Ruletest* is run. Depending on how many *Rulesheets* and rules must be compiled, this compilation step may take a few seconds.

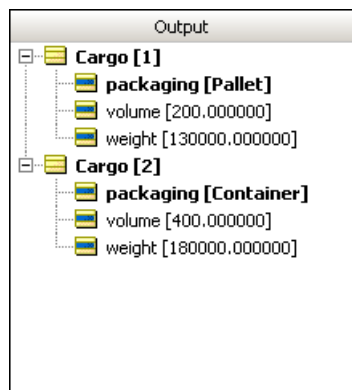
Each time a change is made to a rule in a *Rulesheet* referenced by the *Ruletest*, the rules will need to be re-compiled. This recompilation will occur automatically when **Run Test**  is selected. This will cause a brief delay in execution while the new compilation is performed.

Once a *Ruletest* has been executed, it will rerun without recompiling as long as the rules have not changed since the last execution. Changing the Input test data does not require recompilation.

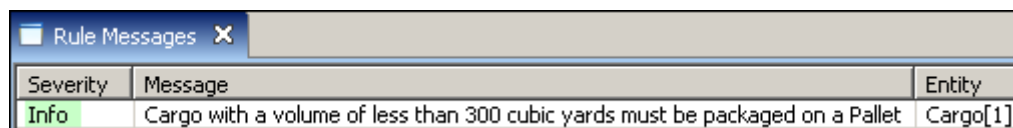
To compile rules *without* running them, a special **Deploy** toolbar option is provided: .

Important: This is a different sequence of events than that performed by earlier versions of Corticon Studio. In versions 4.x and earlier, the compilation step occurred whenever a Rule Set was saved, so rules were already compiled when a *Ruletest* was executed. This resulted in longer save times, but shorter test execution times. The sequence in this Corticon Studio results in shorter save times, but longer test execution times. The *overall sum* of both save and test execution steps remains the same, however.

1. Execute the test by choosing **Ruletest > Testsheet > Run Test** or  on the toolbar.
2. Check the outcome of the test in the **Results** panel.



3. Test results are displayed in regular type style. Parts of the test tree ("nodes") modified by the *Server* are shown in **bold** black text. Any errors are shown in red.



4. Posted rule statements appear in the **Rule Messages** window at the bottom of the Testsheet. In addition to the **Message** and **Severity** columns, the box also displays the **Entity** node to which the rule statement has been posted. In the example above, the Rule Message displayed was posted, or "linked" to Cargo[1]. This means that it was the data contained in Cargo[1] which caused the rule corresponding to the posted Rule Statement to fire.
5. Make any necessary adjustments to the *Rulesheet*, save and re-test by repeating steps 1-5.

Sequence of message posting

Message posting normally occurs at the tail end of a rule's execution – a sort of "final action" even though not technically an action like expressions in Action rows.

An exception to this behavior occurs for Column 0. A rule statement with **Ref** = 0 posts at the *beginning* of Rulesheet execution. If you want to force your rule statements in Column 0 to post after the Action rows execute, use **Ref** values of A0, B0, and so on (rather than 0.)

Sorting messages

Messages posted by rules and displayed in the **Rule Messages** window are a useful tool for understanding and troubleshooting *Rulesheet* execution. Messages displayed can be sorted in the following ways:

- When the **Severity** header is clicked, the messages sort by severity level first (Info then Warning then Violation), then in order of generation, with earlier messages posted higher in the table.
- When the **Message** header is clicked, the messages sort in order of generation, with the first message generated displayed in the first row.
- When the **Entity** header is clicked, the messages sort in ascending order by entity name/number, then in order of generation.

When any Rule Message row is clicked, the corresponding Entity highlights in the **Output** panel above. This provides a convenient way of navigating among the data in a large data set, rather than vertically scrolling.

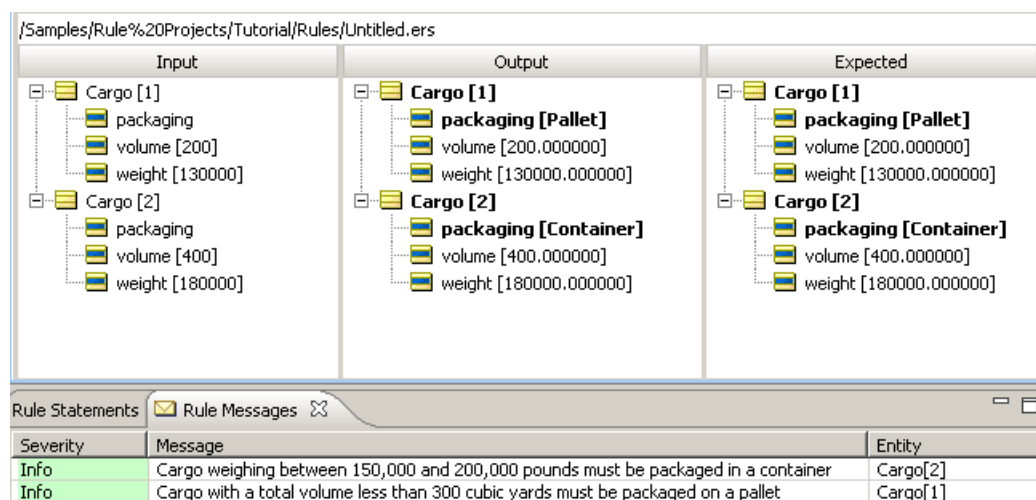
Using the expected panel

If you use the *Ruletest*'s **Expected** panel to create a set of expected test results, then Corticon Studio will automatically compare the actual **Output** data to your **Expected** data, and color code the differences for easy review and analysis.

The **Expected** panel can be populated in the same manner as the **Input** panel - by dragging and dropping nodes from the **Rule Vocabulary** window to create an identical tree structure. A simpler method is provided by the **Ruletest > Testsheet > Data > Output > Copy to Expected** option in the Corticon Studio menubar, or the  button in the Corticon Studio toolbar.

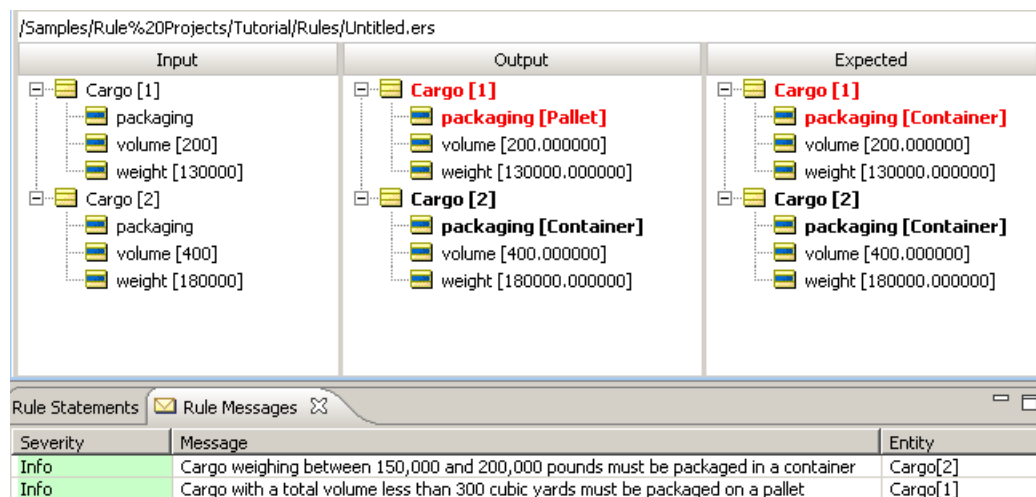
Expected panel – output results match expected exactly

In the example below, both `packaging` values are shown in **bold** text, indicating that these values were changed by the rules. Because all colors are black, the **Output** data is consistent with the **Expected** data.



Expected panel – different values output than expected

In the example shown below, the expected value of `Cargo[1]` packaging value is `Container`, but the *Ruletest* produced an actual value of `Pallet`. Since the **Output** does not match the **Expected** data, the text is colored red.



Expected panel – fewer values output than expected

In the example below, `Cargo[2]` is included in the **Input** and **Expected** trees, but did not appear in the **Output** tree. Because data was expected but not produced, the difference is colored green.

/Samples/Rule%20Projects/Tutorial/Rules/Untitled.ers

Input	Output	Expected
- Cargo [1] - packaging - volume [200] - weight [130000] - Cargo [2] - packaging - volume [400] - weight [180000]	- Cargo [1] - packaging [Pallet] - volume [200.000000] - weight [130000.000000]	- Cargo [1] - packaging [Pallet] - volume [200.000000] - weight [130000.000000] - Cargo [2] - packaging [Container] - volume [400.000000] - weight [180000.000000]

Rule Statements Rule Messages

Severity	Message	Entity
Info	Cargo weighing between 150,000 and 200,000 pounds must be packaged in a container	Cargo[2]
Info	Cargo with a total volume less than 300 cubic yards must be packaged on a pallet	Cargo[1]

Expected panel - more values output than expected

In the example below, *Cargo [2]* was produced in the **Output**, but was not anticipated by the **Expected** panel. For this reason, the difference is colored blue.

/Samples/Rule%20Projects/Tutorial/Rules/Untitled.ers

Input	Output	Expected
- Cargo [1] - packaging - volume [200] - weight [130000] - Cargo [2] - packaging - volume [400] - weight [180000]	- Cargo [1] - packaging [Pallet] - volume [200.000000] - weight [130000.000000] - Cargo [2] - packaging [Container] - volume [400.000000] - weight [180000.000000]	- Cargo [1] - packaging [Pallet] - volume [200.000000] - weight [130000.000000]

Rule Statements Rule Messages

Severity	Message	Entity
Info	Cargo weighing between 150,000 and 200,000 pounds must be packaged in a container	Cargo[2]
Info	Cargo with a total volume less than 300 cubic yards must be packaged on a pallet	Cargo[1]

Expected panel – all problems

All the problems and differences described individually above are combined below to show how a real *Ruletest* (with many problems) may appear.

/Samples/Rule%20Projects/Tutorial/Rules/Untitled.ers

Input	Output	Expected
- Cargo [1] - packaging - volume [200] - weight [130000] - Cargo [2] - packaging - volume [400] - weight [180000]	- Cargo [1] - packaging [Pallet] - volume [200.000000] - weight [130000.000000] - Cargo [2] - packaging [Container] - volume [400.000000] - weight [180000.000000]	- Cargo [1] - packaging [Container] - volume [200.000000] - weight [130000.000000] - Cargo [3] - packaging - volume - weight

Format of output data

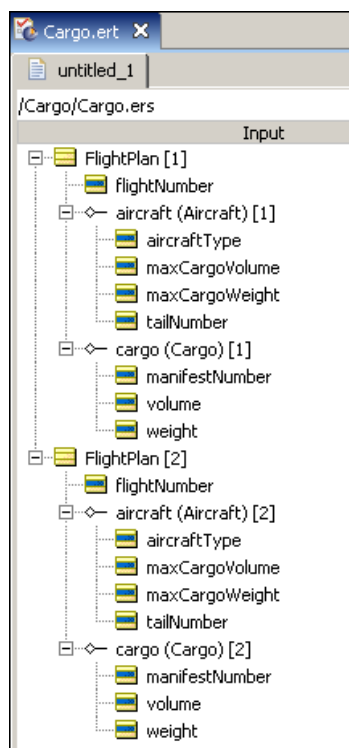
By default, the data in the Output pane will be displayed in the same format as the Input. In other words, if the Input data is flat, then the so will be the Output. If the Input is hierarchical, then so will be the Output.

You can change this default behavior to force full-time hierarchical or flat Output, regardless of Input structure. See the *Server Integration & Deployment Guide* for more information about the `xmlmessagingstyle` property in `CcStudio.properties`.

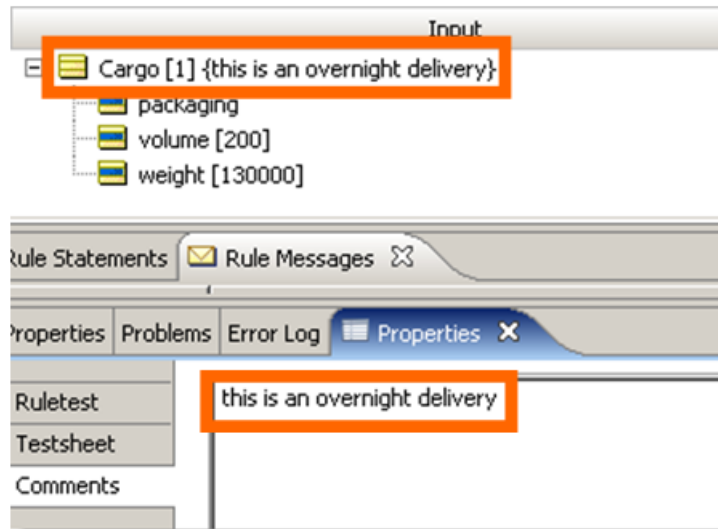
For more information about flat and hierarchical data formats, including examples, see the Integrating Decision Services chapter of the *Server Integration & Deployment Guide*.

Creating multiple test scenarios on the same testsheet

1. To test more than one scenario with the same Input panel, drag as many entity nodes from the Rule Vocabulary as you need. You can drag and drop as per standard procedure, or copy & paste elements already in the Input panel.



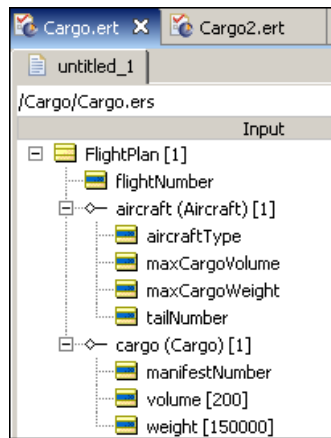
2. New elements will be automatically numbered to ensure a unique ID for each. These unique ID numbers are the same referenced by posted messages in the Results panel. See step 5 in [Executing Tests](#).



3. Additional descriptive text may be added to any node by selecting **Properties** from its right-click pop-up menu, and typing in the **Comments** sub-tab of the **Properties** window.

Creating multiple test scenarios as a set of testsheets

1. It may be more convenient to organize multiple scenarios as a set of Testsheets – each scenario with its own set of **Input**, **Output**, and **Expected** panels.

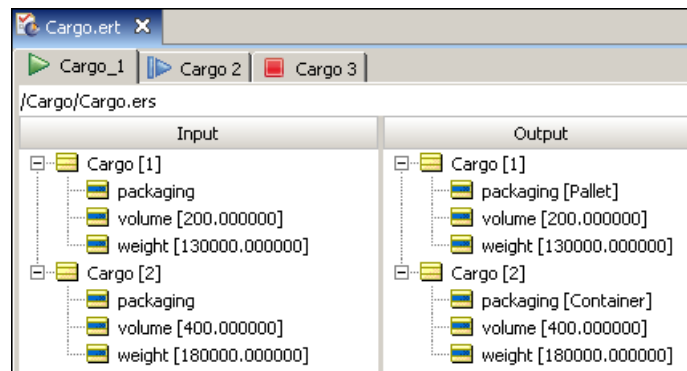


2. Each **Input** panel must be linked to a *Rulesheet* or *Ruleflow*. See step 1 of [Executing Tests](#).
3. Each **Input** panel will generate its own **Results** panel. All of these Testsheet tabs may be [renamed](#) for better organization.

Creating a sequential test using multiple testsheets

1. Multiple *Ruleflows* may be tested in an integrated scenario by linking them together in a single *Ruletest*. This simulates a complex series of decisions or a part of a business process.

Note: *Rulesheets* may also be linked together like this, but *Ruleflows* are more commonly used to test sequences of *Rulesheets*.

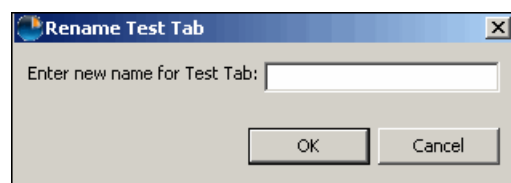


2. Assume we want to test 3 *Ruleflows* (named *Cargo*, *Cargo2* and *Cargo3*) in sequence.
3. Create a new *Ruletest* as before and link it to *Ruleflow cargo.ert* as usual. See step 1 of [Executing Tests](#).
4. Execute the *Ruletest* to generate data in the Output panel.
5. Add a new Testsheet using  and link it to *Ruleflow cargo2.ert*.
6. Link this new Testsheet to the previous Testsheet using . The data from the Output panel of the previous Testsheet should appear in the Input panel of the new Testsheet.
7. Repeat for as many new Testsheets as needed to model the sequence. You can execute one sheet at a time by selecting  from the toolbar, or execute all at once by selecting **Ruletest > Run All Tests** from the menubar

Naming testsheets

When you add a new Testsheet, it is assigned a default name of *Untitled_X*, where *X* is a sequential number. You can rename each Testsheet tab for easy reference. Unnamed Testsheet tabs will be saved with the default name.

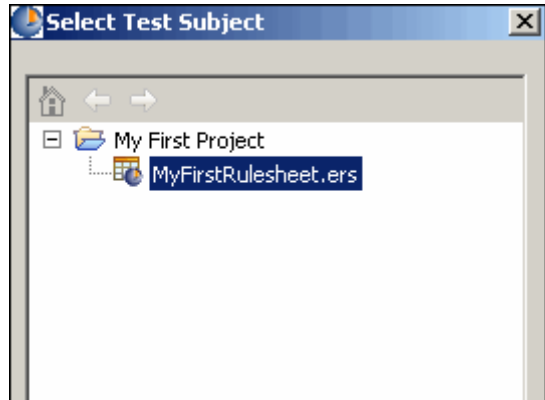
1. You can either select **Ruletest > Testsheet > Rename Testsheet** from the menubar or the right-click pop-up menu, or select the **Rename Testsheet** icon from the *Ruletest* toolbar to display the **Rename Test Tab** dialog box.



2. Type the new name in the space provided and press or click **Enter**.
3. Like *Rulesheets* and Vocabularies, Test and Testsheet names must comply with the guidelines described in [File Naming Restrictions](#).

Adding testsheets

1. You can either select **Ruletest > Testsheet > Add Testsheet** from the menubar or the right-click pop-up menu, or select the **Add Testsheet** icon from the *Ruletest* toolbar to display the **Select Test Subject** dialog box.



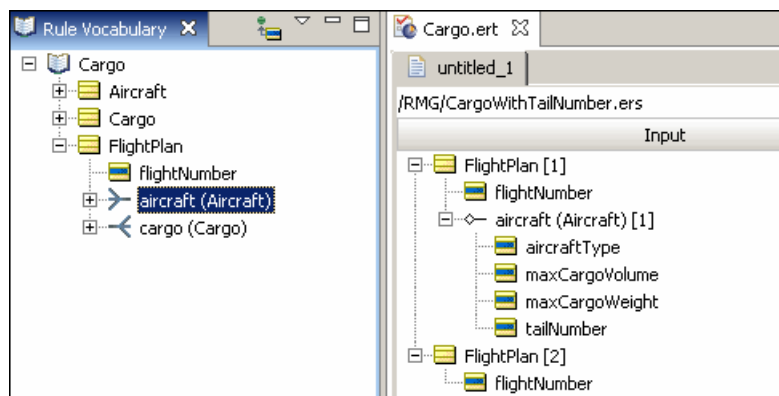
2. Select the *Rulesheet* to associate with the new *Ruletest* and click **OK**.

Important: Corticon Studio creates a new Testsheet tab at the end of the current set of tabs, assigns the new tab a system-generated name, and automatically selects the new tab.

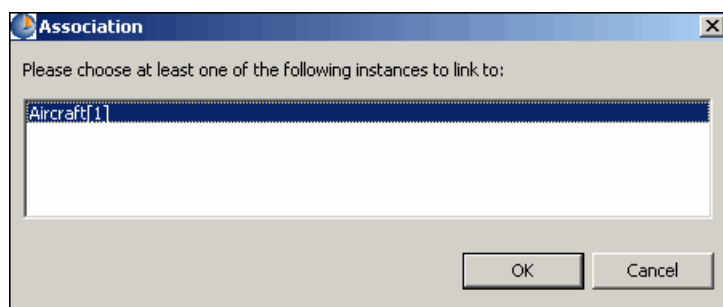
Associating one child entity with more than one parent

To create associations between a child entity and more than one parent:

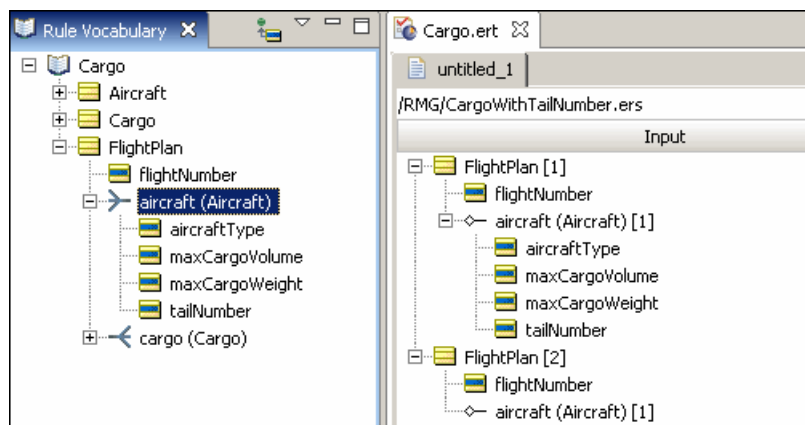
1. First, ensure the parent-child (source-target) association supports a multiplicity of many-to-many or many-to-one.
2. Drag and drop the parent entities (here, *FlightPlan*) as usual.
3. Create an association between *aircraft* and *FlightPlan[1]* as described in **Creating Associations**.
4. To associate *aircraft[1]* to *FlightPlan[2]* in addition to *FlightPlan[1]*, drag *aircraft* as shown to the right, and press the **CTRL** key as you drop it.



5. A pop-up window will ask you to select the Aircraft you are linking to. In this case, select `Aircraft[1]` and click **OK**.



6. Verify the child entity has the appropriate "indented" structure underneath both parent entities, indicating the two associations have been created correctly.
7. Notice that data for `aircraft[1]` is only entered once, under the `FlightPlan[1]` entity, even though it is associated with both `FlightPlans`.



8. In large Testsheets, it may be difficult to locate the original instance of `aircraft[1]`. Clicking on any copy and choosing **Ruletest > Go To Entity** from the menubar will automatically locate the original instance.

Saving ruletests

When you save a new test, the **Save** Window displays. Follow the instructions shown in [Saving a New Rulesheet](#) to save your test scenario.

When you save an existing test, the system automatically saves it to the existing name (`*.ert`). To save the *Ruletest* to another name:

- Choose **File > Save As** from the menu.

Importing an XML or SOAP document to a testsheet

In some cases, it may be more convenient to import data into the Tester rather than recreate it. If this data can be structured in an XML document that adheres to the structure defined by the Vocabulary, then this document can be directly imported into Corticon Studio using **Test > Import XML/SOAP**. Corticon Studio can provide a sample XML or SOAP document to use as a template by choosing **Test > Export XML** or **Test > Export SOAP**.

Exporting a testsheet to an XML document

Exporting Testsheets into XML documents can be useful for a few reasons:

- For use as a template for structuring much larger data sets for subsequent XML Import (as described above).
- To generate an XML data payload to test a deployed *Rulesheet* (a Decision Service on the *Server*).

A Sample Testsheet Exported to XML

The sample XML file shown below is a direct export of the Input Testsheet created in [Populating the Input Testsheet](#) section. Notice the hierarchical structure.

Consult the *Server Integration & Deployment Guide* for more information about the sections of the XML document, including the important `decisionServiceName` parameter shown in line 2 of the `CorticonRequest` tag, below.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <CorticonRequest xmlns="urn:decision:insertDecisionServiceName"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   decisionServiceName="InsertDecisionServiceName">
3    <WorkDocuments>
4      <FlightPlan id="FlightPlan_id_1">
5        <flightNumber xsi:nil="true" />
6        <aircraft id="Aircraft_id_1">
7          <aircraftType xsi:nil="true" />
8          <maxCargoVolume xsi:nil="true" />
9          <maxCargoWeight xsi:nil="true" />
10         <tailNumber xsi:nil="true" />
11       </aircraft>
12       <cargo id="Cargo_id_1">
13         <manifestNumber xsi:nil="true" />
14         <volume>400</volume>
15         <weight>175000</weight>
16       </cargo>
17     </FlightPlan>
18   </WorkDocuments>
19 </CorticonRequest>

```

Exporting a testsheet to a SOAP message

This feature is similar to **Export to XML**, except that it encloses the XML payload within SOAP-compliant envelope information. The XML payload is identical to the XML generated by the **Export to XML** option, as shown above.

Consult the *Server Integration & Deployment Guide* for more information about the sections of the SOAP request document, including the important `decisionServiceName` parameter shown in the first line of the `CorticonRequest` tag, below.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <SOAP-ENV:Envelope xmlns:SOAP-ENV=
   "http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC=
   "http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd=
   "http://www.w3.org/2001/XMLSchema">
3  <SOAP-ENV:Body>
4  <CorticonRequest xmlns=
   "urn:decision:insertDecisionServiceName" xmlns:xsi=
   "http://www.w3.org/2001/XMLSchema-instance"
   decisionServiceName="InsertDecisionServiceName">
5      <WorkDocuments>
6          <FlightPlan id="FlightPlan_id_1">
7              <flightNumber xsi:nil="true" />
8              <aircraft id="Aircraft_id_1">
9                  <aircraftType xsi:nil="true" />
10                 <maxCargoVolume xsi:nil="true" />
11                 <maxCargoWeight xsi:nil="true" />
12                 <tailNumber xsi:nil="true" />
13             </aircraft>
14             <cargo id="Cargo_id_1">
15                 <manifestNumber xsi:nil="true" />
16                 <volume>400</volume>
17                 <weight>175000</weight>
18             </cargo>
19         </FlightPlan>
20     </WorkDocuments>
21 </CorticonRequest>
22 </SOAP-ENV:Body>
23 </SOAP-ENV:Envelope>

```

Creating a Ruletest report

1. Choose **Ruletest > Report** from the menubar.
2. If your local machine has a web browser installed, the HTML report should open as a new web page.
3. Corticon Studio will save the report file to `/Users/<your username>/AppData/Local/Temp` using the name format `Corticon Ruletest XXXXX.html`, where `XXXXX` is a unique auto-generated number. You can save the HTML to a different name or location from the browser.
4. For information about creating custom reports or saving them to custom locations, see the "Corticon Reporting Framework" chapter of the *Rule Modeling Guide*.

Exiting Corticon Studio

When you close the Corticon Studio (select **File > Exit** from the menu), you are prompted to save all open Vocabulary, *Rulesheet*, *Ruleflow*, and *Ruletest* files.

Important: If you choose to exit without first saving your files, any changes you made since opening them will be lost.

A

Keyboard shortcuts

Several menu commands have keyboard shortcuts, as noted next to the corresponding menu command. In addition to these, the following keyboard shortcuts are available:

Activate Editor	F12
Backward History	Alt+Left
Build All	Ctrl+B
Close	Ctrl+W
Close All	Ctrl+Shift+W
Collapse All	Ctrl+Shift+Numpad_Divide
Content Assist	Ctrl+Space
Context Information	Ctrl+Shift+Space
Copy	Ctrl+C
Cut	Ctrl+X
Delete	Delete
Find and Replace	Ctrl+F
Forward History	Alt+Right
Last Edit Location	Ctrl+Q
Maximize Active View or Editor	Ctrl+M
New	Ctrl+N
New menu	Alt+Shift+N
Next	Ctrl+.
Next Editor	Ctrl+F6
Next Page	Alt+F7
Next Perspective	Ctrl+F8
Next Sub-Tab	Alt+PageDown
Next View	Ctrl+F7
Open Resource	Ctrl+Shift+R
Paste	Ctrl+V
Previous	Ctrl+,
Previous Editor	Ctrl+Shift+F6
Previous Page	Alt+Shift+F7
Previous Perspective	Ctrl+Shift+F8
Previous Sub-Tab	Alt+PageUp

Previous View	Ctrl+Shift+F7
Print	Ctrl+P
Properties	Alt+Enter
Quick Access	Ctrl+3
Quick Fix	Ctrl+1
Quick Switch Editor	Ctrl+E
Redo	Ctrl+Y
Refresh	F5
Rename	F2
Save	Ctrl+S
Save All	Ctrl+Shift+S
Select All	Ctrl+A
Show In...	Alt+Shift+W
Show Key Assist	Ctrl+Shift+L
Show System Menu	Alt+-
Show View	Alt+Shift+Q, Q
Show View (View: Console)	Alt+Shift+Q, C
Show View (View: Error Log)	Alt+Shift+Q, L
Show View (View: Outline)	Alt+Shift+Q, O
Show View (View: Problems)	Alt+Shift+Q, X
Show View Menu	Ctrl+F10
Switch to Editor	Ctrl+Shift+E
Undo	Ctrl+Z
Zoom In	Ctrl+=
Zoom Out	Ctrl+-

You can access this list online by choosing the menu command **Help > Key Assist** or pressing **Ctrl+Shift+L**.

Third party acknowledgments

One or more products in the Progress Corticon v5.3.3 release includes third party components covered by licenses that require that the following documentation notices be provided:

Progress Corticon v5.3.3 incorporates Apache Commons Discovery v0.2 from The Apache Software Foundation. Such technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 - Copyright (c) 1999-2001 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgement: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).\" Alternately, this acknowledgement may appear in the software itself, if and wherever such third-party acknowledgements normally appear.
4. The names "The Jakarta Project", "Commons", and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.
5. Products derived from this software may not be called "Apache" nor may "Apache" appear in their names without prior written permission of the Apache Group.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Corticon v5.3.3 incorporates Apache SOAP v2.3.1 from The Apache Software Foundation. Such technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 Copyright (c) 1999 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "SOAP" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>.

Progress Corticon v5.3.3 incorporates DOM4J v1.6.1. Such technology is subject to the following terms and conditions: Project License BSD style license Copyright 2001-2005 (C) MetaStuff, Ltd. All Rights Reserved.

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "DOM4J" must not be used to endorse or promote products derived from this Software without prior written permission of MetaStuff, Ltd. For written permission, please contact dom4j-info@metastuff.com.

4. Products derived from this Software may not be called "DOM4J" nor may "DOM4J" appear in their names without prior written permission of MetaStuff, Ltd. DOM4J is a registered trademark of MetaStuff, Ltd.

5. Due credit should be given to the DOM4J Project - <http://www.dom4j.org>

THIS SOFTWARE IS PROVIDED BY METASTUFF, LTD. AND CONTRIBUTORS ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL METASTUFF, LTD. OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT,

INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Corticon v5.3.3 incorporates Jaxen v1.0. Such technology is subject to the following terms and conditions: JAXEN License - \$Id: LICENSE,v 1.3 2002/04/22 11:38:45 jstrachan Exp \$ - Copyright (C) 2000-2002 bob mcwhirter and James Strachan. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.

3. The name "Jaxen" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jaxen.org.

4. Products derived from this software may not be called "Jaxen", nor may "Jaxen" appear in their name, without prior written permission from the Jaxen Project Management (pm@jaxen.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the Jaxen Project (<http://www.jaxen.org/>)."

Alternatively, the acknowledgment may be graphical using the logos available at <http://www.jaxen.org/>. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE Jaxen AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Jaxen Project and was originally created by bob mcwhirter <bob@werken.com> and James Strachan <jstrachan@apache.org>. For more information on the Jaxen Project, please see <<http://www.jaxen.org/>>.

Progress Corticon v5.3.3 incorporates JDOM v1.0 GA. Such technology is subject to the following terms and conditions: \$Id: LICENSE.txt,v 1.11 2004/02/06 09:32:57 jhunter Exp \$ - Copyright (C) 2000-2004 Jason Hunter and Brett McLaughlin. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.
3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact <request_AT_jdom_DOT_org>.
4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management <request_AT_jdom_DOT_org>.

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (<http://www.jdom.org/>)."

Alternatively, the acknowledgment may be graphical using the logos available at <http://www.jdom.org/images/logos>. THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the JDOM Project and was originally created by Jason Hunter <jhunter_AT_jdom_DOT_org> and Brett McLaughlin <brett_AT_jdom_DOT_org>. For more information on the JDOM Project, please see <<http://www.jdom.org/>>.

Progress Corticon v5.3.3 incorporates Saxpath v1.0. Such technology is subject to the following terms and conditions: Copyright (C) 2000-2002 werken digital. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.
3. The name "SAXPath" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@saxpath.org.
4. Products derived from this software may not be called "SAXPath", nor may "SAXPath" appear in their name, without prior written permission from the SAXPath Project Management (pm@saxpath.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the SAXPath Project (<http://www.saxpath.org/>)."

Alternatively, the acknowledgment may be graphical using the logos available at <http://www.saxpath.org/>

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE SAXPath AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the SAXPath Project and was originally created by bob mcwhirter <bob@werken.com> and James Strachan <jstrachan@apache.org>. For more information on the SAXPath Project, please see <<http://www.saxpath.org/>>.

