

Corticon Studio: Quick Reference Guide

Notices

Copyright agreement

© 2016 Progress Software Corporation and/or one of 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.

Business Making Progress, Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, Deliver More Than Expected, Icenium, Kendo UI, Making Software Work Together, NativeScript, OpenEdge, Powered by Progress, Progress, Progress Software Business Making Progress, Progress Software Developers Network, Rollbase, RulesCloud, RulesWorld, SequeLink, Sitefinity (and Design), SpeedScript, Stylus Studio, TeamPulse, Telerik, Telerik (and Design), Test Studio, and WebSpeed are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, AppsAlive, AppServer, BravePoint, BusinessEdge, DataDirect Spy, DataDirect SupportLink, Future Proof, High Performance Integration, OpenAccess, ProDataSet, Progress Arcade, Progress Profiles, Progress Results, Progress RFID, Progress Software, ProVision, PSE Pro, SectorAlliance, Sitefinity, 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.

Please refer to the Release Notes applicable to the particular Progress product release for any third-party acknowledgements required to be provided in the documentation associated with the Progress product.

Table of Contents

Chapter 1: Using Progress Corticon Studio.....	9
Progress Corticon Studio components.....	9
Initial menu commands.....	10
Keyboard shortcuts	15
Initial toolbar.....	17
The file tab.....	18
The active Corticon Studio window.....	19
File naming restrictions.....	20
Renaming and relocating assets in the Project Explorer.....	21
 Chapter 2: Rule Projects.....	 23
Creating a Rule Project.....	23
Downloading a Project from a deployed Decision Service.....	26
The Project Explorer window.....	27
 Chapter 3: Vocabularies.....	 31
Creating a Vocabulary.....	31
The Vocabulary window.....	33
Vocabulary menu commands.....	33
Vocabulary toolbar.....	34
Commands on the Vocabulary context-sensitive menu.....	34
Populating a new Vocabulary.....	35
The Vocabulary tree view.....	35
Vocabulary custom data types and database access.....	35
Adding nodes to the Vocabulary tree view.....	36
Vocabulary node naming restrictions.....	36
Domain nodes: Adding and editing domains and their properties.....	37
Entity nodes: Adding and editing entities and their properties.....	37
Attribute nodes: Adding and editing attributes and their properties.....	42
Enumerated values.....	45
Association Nodes: Adding and editing associations and their properties.....	46
Editing an association.....	50
Saving a new Vocabulary.....	50
Opening an existing Vocabulary.....	52
Modifying a Vocabulary.....	52
Creating a Vocabulary report.....	53

Chapter 4: Rulesheets.....55

Creating a new Rulesheet.....	56
Rulesheet menu commands.....	58
Rulesheet toolbar.....	59
Commands on the Rulesheet context-sensitive menu.....	61
Rulesheet sections.....	62
Rule statements window.....	64
Rulesheet properties.....	66
Using the business vocabulary to build rules.....	67
Using the operator vocabulary to build rules.....	67
Naming Rulesheets.....	68
Deleting Rulesheets.....	68
Saving a new Rulesheet.....	69
Validating and optimizing a Rulesheet.....	69
Creating a Rulesheet report.....	70
Closing a Rulesheet.....	70
Saving a modified Rulesheet.....	70

Chapter 5: Ruleflows.....71

Creating a new Ruleflow.....	72
Ruleflow window.....	73
Naming Ruleflows.....	74
Adding Ruleflows.....	74
Deleting Ruleflows.....	74
Saving a new Ruleflow.....	74
Ruleflow menu commands.....	74
Ruleflow toolbar.....	74
Editor commands on the Ruleflow context-sensitive menu.....	75
Renaming a Ruleflow and/or saving a Ruleflow to a different location.....	77
Creating a Ruleflow report.....	78
Closing a Ruleflow.....	79
Ruleflow properties.....	79
Ruleflow	80
Comments.....	81
Rulers and grid.....	82
Ruleflow preferences.....	82

Chapter 6: Objects on a Ruleflow canvas.....87

Ruleflow canvas tools.....	89
Object commands on the Ruleflow context-sensitive menu.....	89
Properties of Ruleflow objects on a Ruleflow canvas.....	92
Iteration.....	92

Adding colors and comments to Ruleflow objects.....	94
---	----

Chapter 7: Ruletests.....97

Creating a new Ruletest.....	98
Choosing a test subject in the Studio workspace.....	100
Choosing a test subject that is a deployed Decision Service.....	100
Specifying server URLs for access to test subjects.....	102
Ruletest window.....	103
Ruletest menu commands.....	103
Ruletest toolbar.....	107
Commands on a Testsheet context-sensitive menu.....	108
Testsheet tabs.....	110
Populating the input panel.....	111
Adding entities to the test tree.....	111
Creating associations in the test tree.....	111
Assigning attribute values in the test tree.....	112
Automatically generating a test tree.....	113
Executing tests.....	113
Sequence of message posting.....	114
Sorting messages.....	114
Using the expected panel.....	114
Expected panel: Output results match expected exactly.....	115
Expected panel: Different values output than expected.....	115
Expected panel: Fewer values output than expected.....	116
Expected panel: More values output than expected.....	116
Expected panel: All problems.....	117
Expected panel: Setting selected attributes to ignore validation.....	117
Format of output data.....	119
Creating multiple test scenarios on the same testsheet.....	120
Creating multiple test scenarios as a set of testsheets.....	121
Creating a sequential test using multiple testsheets.....	121
Naming testsheets.....	122
Adding testsheets.....	123
Associating one child entity with more than one parent.....	123
Saving a Ruletest.....	124
Importing an XML or SOAP document to a testsheet.....	124
Importing a JSON document to a testsheet.....	125
Exporting a testsheet to an XML document.....	125
Exporting a testsheet to a SOAP message.....	126
Exporting a testsheet to a JSON document.....	127
Creating a Ruletest report.....	128

Chapter 8: Exiting Corticon Studio.....129

Appendix A: Access to Corticon knowledge resources.....	131
--	------------

Using Progress Corticon Studio

For details, see the following topics:

- [Progress Corticon Studio components](#)
- [Initial menu commands](#)
- [Keyboard shortcuts](#)
- [Initial toolbar](#)
- [The file tab](#)
- [The active Corticon Studio window](#)
- [File naming restrictions](#)
- [Renaming and relocating assets in the Project Explorer](#)

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** contains 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 *Corticon Server: Integration & Deployment Guide*. A quick introduction to this topic is provided in the two *Corticon Server: Deploying Web Services* guides.

File Suffixes

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

- `.ecore` — the Vocabulary.
- `.ers` — a Rulesheet. A Rulesheet has only one associated Vocabulary. Multiple Rulesheets can use the same Vocabulary
- `.erf` — a Ruleflow. One Ruleflow can contain multiple Rulesheets, Service Callouts, or other Ruleflows.
- `.ert` — a Ruletest. A Ruletest's test subject is either a Rulesheet or a Ruleflow.

Initial menu commands

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

The following actions are available when Corticon Studio opens:

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 Ruleflows.
- **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, except the following:

- **Package and Deploy Decision Services** - Enables quick deployment of a Ruleflow as a Decision Service to a Corticon Server. For details, see the topic *"Using Studio to compile and deploy Decision Services" in the "Packaging and deploying Decision Services" section of the Integration and Deployment Guide.*
- **Download Decision Services** - Enables retrieval of selected project assets from a Corticon Server. For details, see the topic .
- **Upgrade Rule Assets** - Opens the **Upgrade Corticon Assets** dialog. For a discussion of this feature, see *"Upgrading projects coming forward from a prior release" in the Corticon Installation Guide.*

Run Menu

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

Vocabulary, Rulesheet, Ruleflow, Ruletest Menus

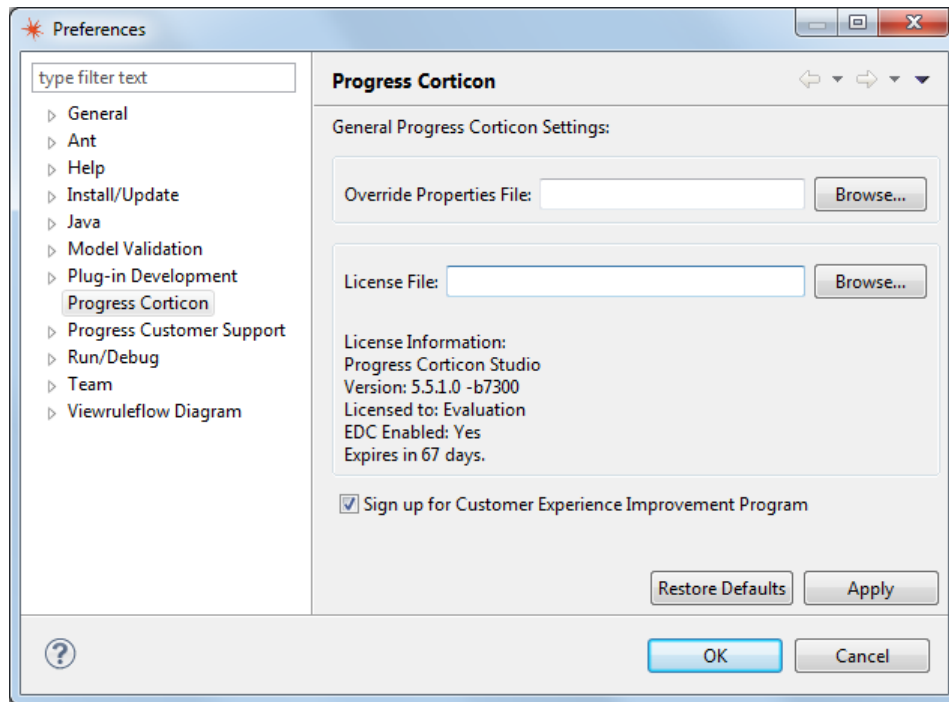
Note: When a Corticon file is active, the corresponding menu is added to the toolbar. Each of these is described in later sections of this guide, as follows:

- [Vocabulary menu commands](#) on page 33
- [Rulesheet menu commands](#) on page 58
- [Ruleflow menu commands](#) on page 74
- [Ruletest menu commands](#) on page 103

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:
 - **Progress Corticon:**



- **Setting the override properties file** - The basic override file, `brms.properties`, is installed at the work directory root. You can choose to point to a file that exists at your preferred location that will be the last set of overrides that are loaded.

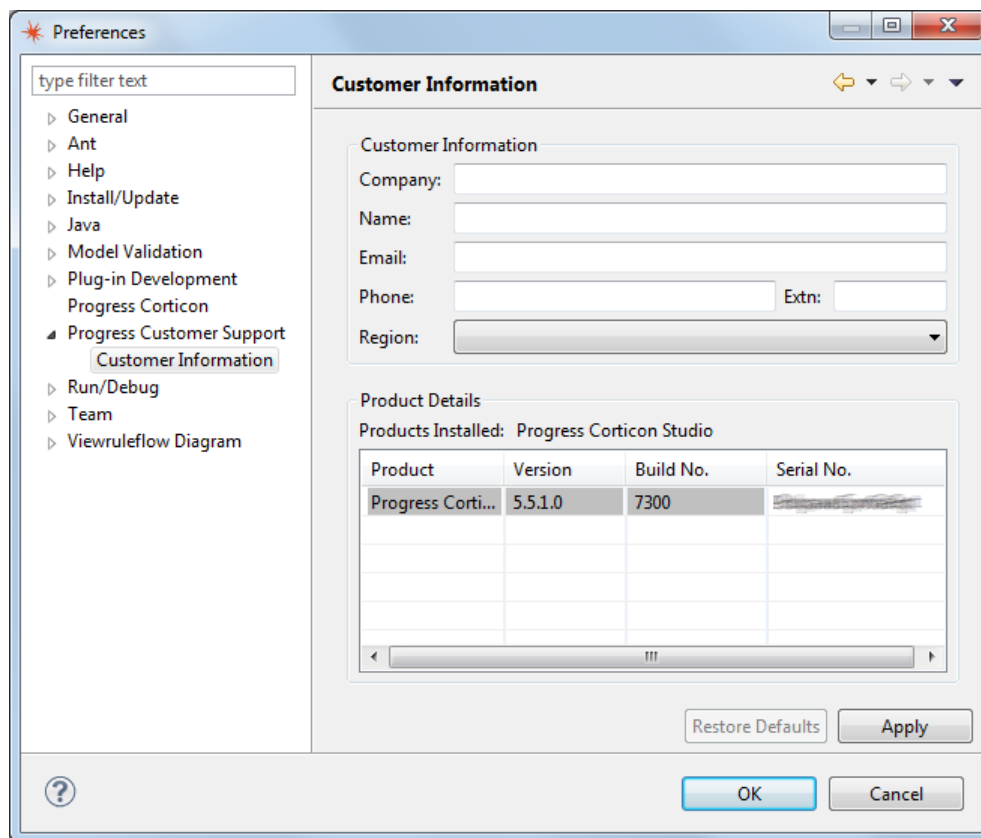
Note: Changing the override properties file -- and changes within an override file --- are not applied until you restart the Studio.

- **License file path and information** - The current license file information is displayed. You can point a preferred license file or license file location.
- **Sign up for the Customer Experience Improvement Program** - Reflects the option selected when you opened the current workspace. When the option is cleared, you are opted out.

Note: Changing the license file location is applied as soon as you click **Apply** or **OK** -- you do not need to restart the Studio.

Click **Apply** to record your selections. Click **OK** to close the dialog.

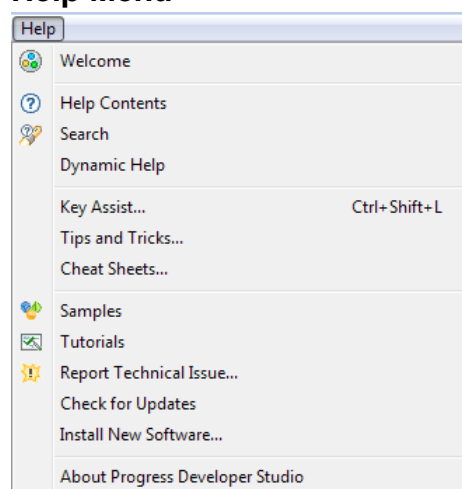
- **Progress Customer Support > Customer Information:**



Click **Apply** to record your entries. Click **OK** to close the dialog.

Note: The setting for **User Role** as **Rule Modeler** or **Integration and Deployment** that was a **Preference** setting in releases prior to 5.5.1 has moved to the **Vocabulary** menu as the **Simple View** (Rule Modeler) or **Advanced View** (Integration and Deployment).

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. Use the **Home** button in the upper right if the page the opens is not the top page.
- **Help Contents** - Opens the help system window where you can search for terms and browse the help structure of Eclipse, third-party, and Progress help, including Progress Corticon Documentation, the complete documentation for this product release.
- **Search** - Implements the help system on a tab instead of in a separate window.
- **Dynamic Help Contents** - Opens context-sensitive Corticon help plus access to related topics.
- **Key Assist** - Keyboard shortcuts are detailed in the next topic. You can press **Ctrl+Shift+L** to open the **Key Assist** panel.
- **Samples** - Opens the **Welcome** page's **Samples** panel to access and unpack Corticon sample projects.
- **Tutorials** - Opens the **Welcome** page's **Tutorials** panel to access the Basic and Advanced tutorial documents.
- **Report Technical Issues** - Enables submission of issues with Corticon and other Progress products. You are guided to completing the Customer Information if you have not already done so. For detailed instructions, see the Help section **Progress Technical Issue Reporting Guide**.
- **Check for Updates** - Accesses Progress and related sites to determine whether updates are available.
- **About Progress Developer Studio** - Details versions and licenses of installed the Progress Developer Studio, Corticon Studio, Eclipse, and other installed products. Also provides access to installation details.

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+-







Initial toolbar

The initial Corticon Studio toolbar displays basic tools that you will apply in most of the Corticon editor windows.



The tools 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.
-  Saves changes to all open files.
-  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.

Standard Eclipse tools are then appended to the toolbar:



and added at the right end of the toolbar:



For more information about these tools and their relevance to Corticon functions, refer to the Eclipse and Workbench help topics in the online help.

When a Corticon Studio editor is active, its tools are added to the toolbar. See the following topics for more information:

- [Vocabulary toolbar](#) on page 34
- [Rulesheet toolbar](#) on page 59
- [Ruleflow toolbar](#) on page 74
- [Ruletest toolbar](#) on page 107

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


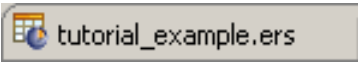



Throughout this manual, reference will be made to the "active" Corticon Studio window. Any single open window in Corticon Studio may be active at a given time. Indication of active status is provided by the window's tab color. As shown below, the active window's tab is colored **blue**, while inactive 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

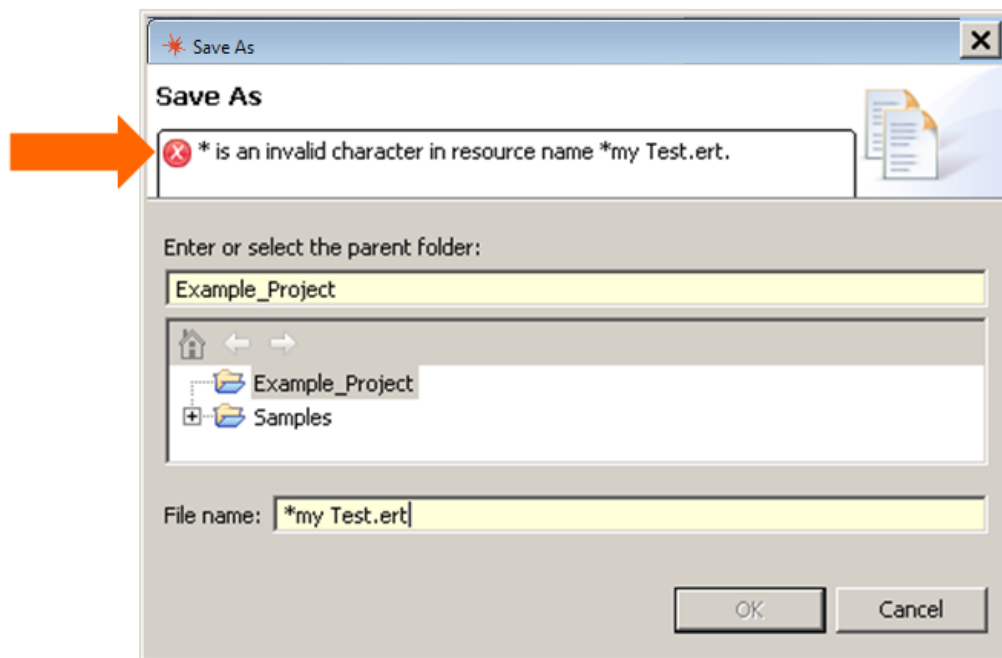
Each window tab contains information about the window's status:

	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 must not begin with a space, but can contain spaces in subsequent character positions.
- File names can 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:



Renaming and relocating assets in the Project Explorer

When your project becomes large, you might want to revise some of the file names, and then sort them into various folders. Within Studio, you can rename Corticon *rule asset files* -- Vocabularies (*.ecore*), Rulesheets (*.ers*), Ruleflows (*.erf*), and Ruletests (*.ert*) -- and folders within their project. You can also move files to other new or existing folder hierarchies in the project. The Studio *refactors* these changes so names and paths within the project are correspondingly adjusted in all the related rule asset files in the project.

Examples:

- You relocate the project's Vocabulary to a new *Vocab* folder. Rulesheets, Ruletests, and Ruleflows that reference the Vocabulary are automatically updated.
- You rename a Rulesheet that is referenced in several Ruleflows. The Rulesheet and the Ruleflows that include that Rulesheet are automatically updated.
- You rename a Ruleflow that is the test subject of a Ruletest. The Ruleflow and the Ruletests that reference that Ruleflow are automatically updated.

Note: Close all files that are active in editors before attempting to change or move files in the Project Explorer. If your change impacts a file that is open in a Studio editor -- whether directly or by reference -- the Studio will close the file without saving any changes you have made since the last save action.

Note: Perform renaming and relocation changes *within* Corticon Studio's Project Explorer. Other techniques for file name and path changes, such as Windows Explorer or command line interface, will result in dependent file references becoming invalid.

Note: This functionality relates only to Corticon rule asset names and paths. Renaming a Vocabulary's elements -- domains, entities, attributes, and associations -- is advisable only before you create assets based on the Vocabulary.

Note: Deployment oriented files (and any other non-rule asset files) are unchanged. You might need to regenerate schema files (.xsd and .wsdl). Decision Service files (.esd) are each complete package, but Deployment Descriptor files (.cdd) files will require manual editing to update rule asset names and paths.

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:

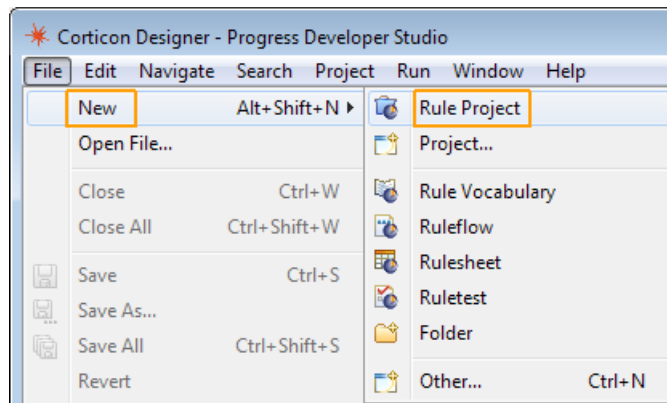
- [Creating a Rule Project](#)
- [Downloading a Project from a deployed Decision Service](#)
- [The Project Explorer window](#)

Creating a Rule Project

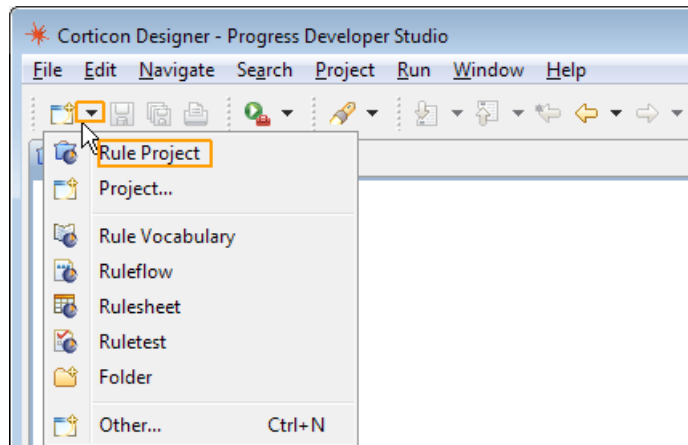
Follow these steps to create a new Rule Project:

Do one of the following:

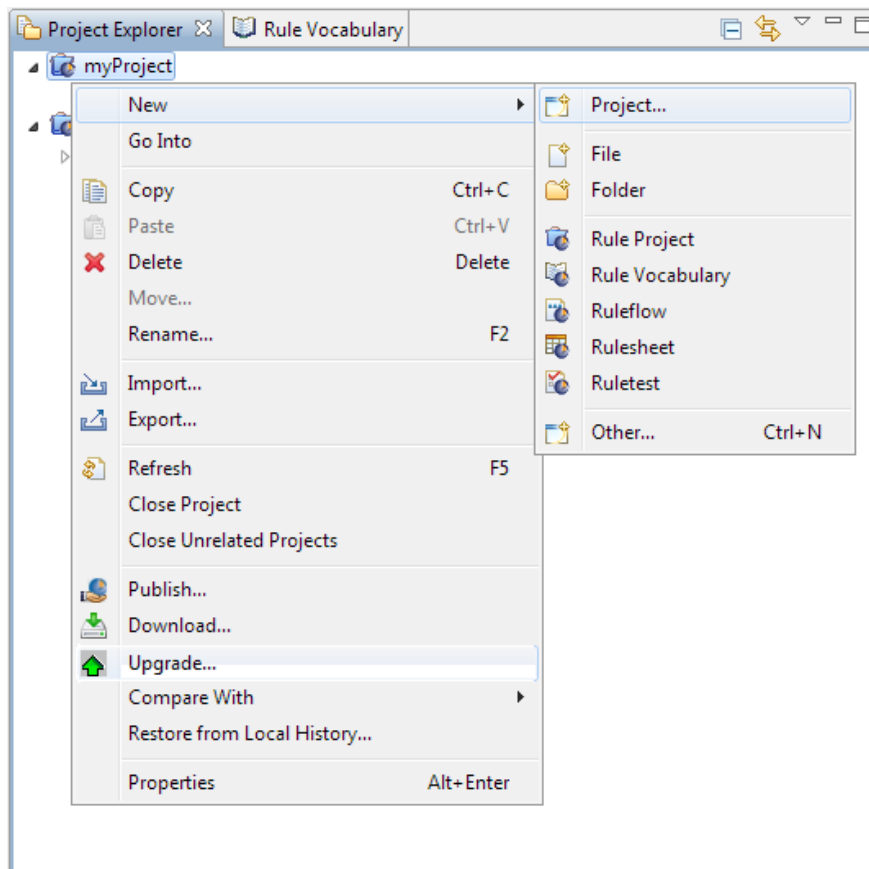
- Select **File > New > Rule Project** from the menubar



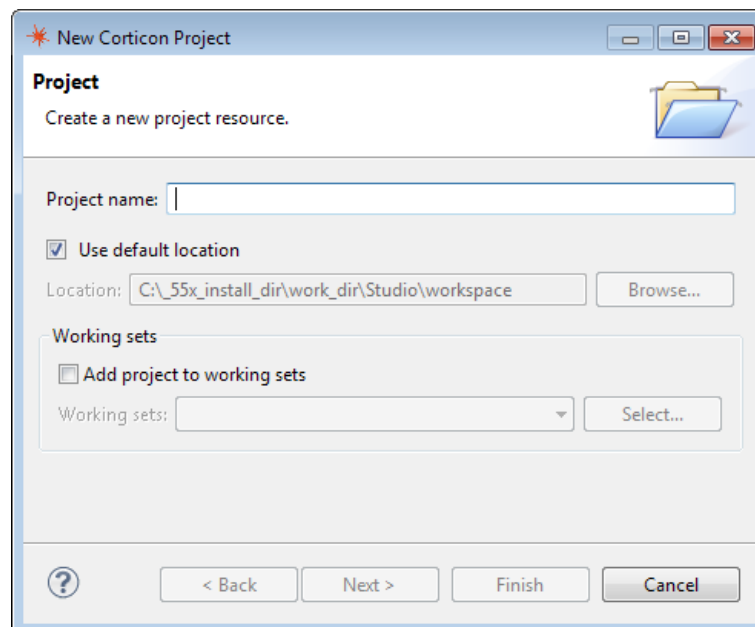
- Click the down arrow to the right of the **New** icon  on the toolbar and select **Rule Project**.



- 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.

Downloading a Project from a deployed Decision Service

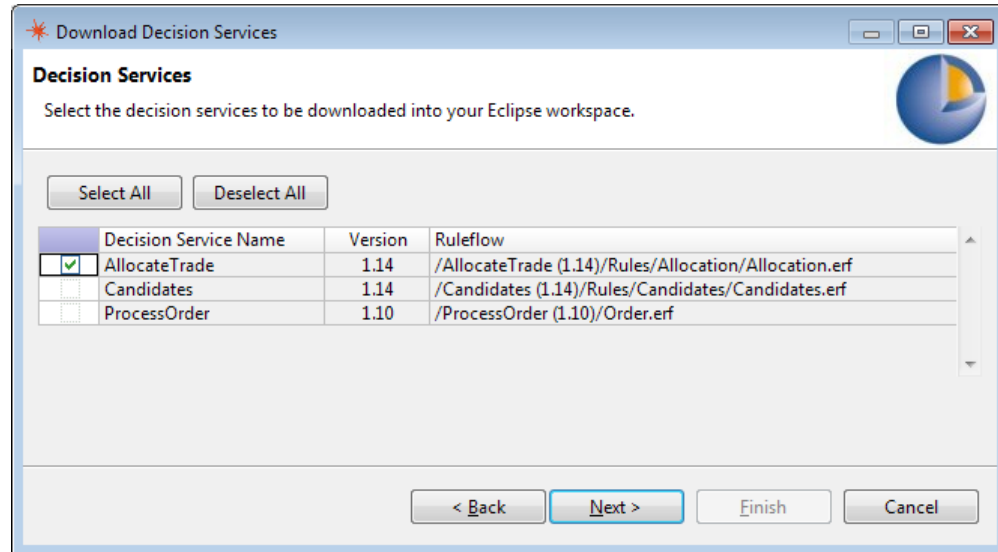
The Download wizard lets you connect to a Corticon Server to download Decision Services to the Corticon Studio location. You can then modify and save these files to create a new version of the Decision services to a Server using *"Using Studio to compile and deploy Decision Services" in the Integration and Deployment Guide*.

To download a Decision Service:

1. Select **Project > Download**.
2. Enter the **Server URL** of the Corticon server.
3. Enter **Username** and **Password** (Use the standard credentials `admin` and `admin`).
4. Click **Test Connection**.
 - For successful connection, the system displays: **Server connection test was successful**.
 - For invalid username or password, the system displays: **User does not have rights to upload/download content to/from the server**.
 - For commonplace errors such as the server being down or unreachable, the system displays: **Server connection test failed. Server may be off-line, unreachable or not listening on specified port**.
5. Click **Next**.

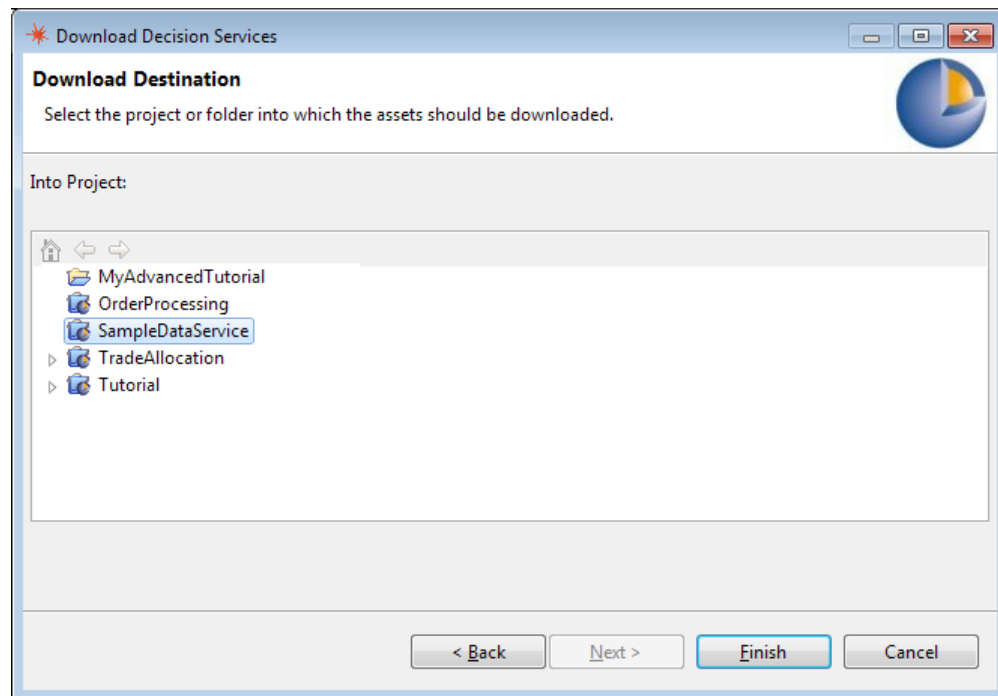
The available Decision Services are listed.
6. Select the Decision Service to be downloaded, as shown in [Available Decision Services](#), and click **Next**.

Figure 1: Available Decision Services



7. Select the destination project or a folder, as shown in [Download Destination for the Selected Decision Services](#), in which the Decision Service is to be downloaded and click **Finish**.

Figure 2: Download Destination for the Selected Decision Services

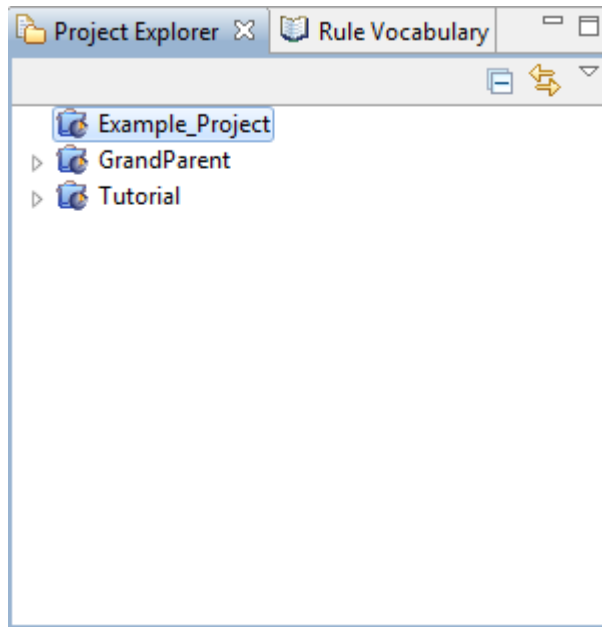


When you click **Finish**, the system downloads the selected assets into the selected project or folder.

The Project Explorer window

The Rule Project you just created is now listed in the **Project Explorer** window in Corticon Studio.

The **Project Explorer** window provides a convenient way of navigating between your Corticon Studio 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. Our new `Example_Project`, shown below, is empty, so there are no files available to open yet.

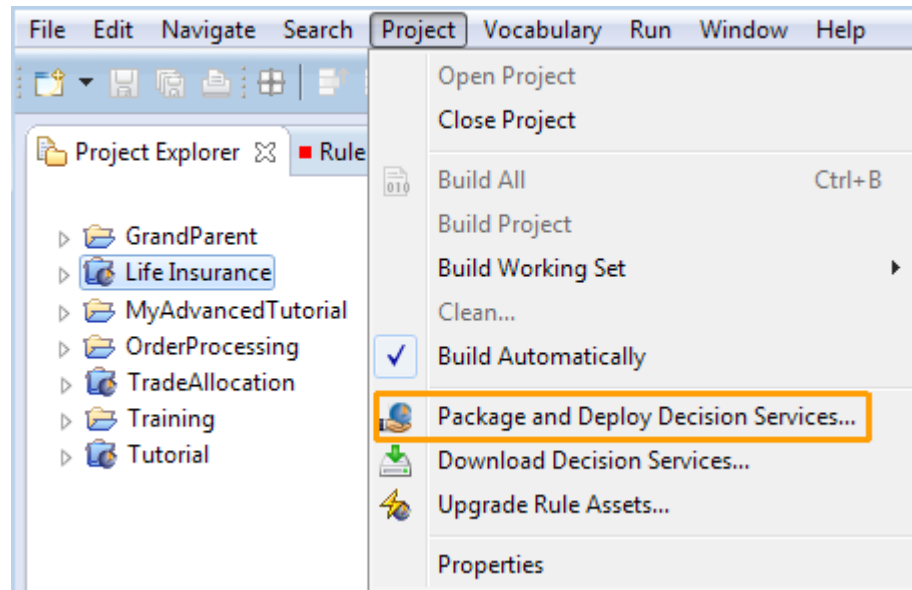


Project Menu

As Rule Projects do not have an editor, their project-level actions are appended to the standard **Project** menu to apply to the currently selected project. The actions accessed on the **Project** menu are standard Eclipse functions, except the following:

- **Package and Deploy Decision Services** - Enables quick deployment of a Ruleflow as a Decision Service to a Corticon Server. For details, see the topic *"Using Studio to compile and deploy Decision Services"* section of the *Integration and Deployment Guide*.
- **Download Decision Services** - Enables retrieval of selected project assets from a Corticon Server. For details, see the topic .
- **Upgrade Rule Assets** - Opens the **Upgrade Corticon Assets** dialog. For a discussion of this feature, see *"Upgrading projects coming forward from a prior release"* in the *Corticon Installation Guide*.

These commands are also accessible on the dropdown menu that opens when you right-click on a project, as shown in this excerpted image where the intent is to publish selected Ruleflows that are in the **Life Insurance** project:



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](#)

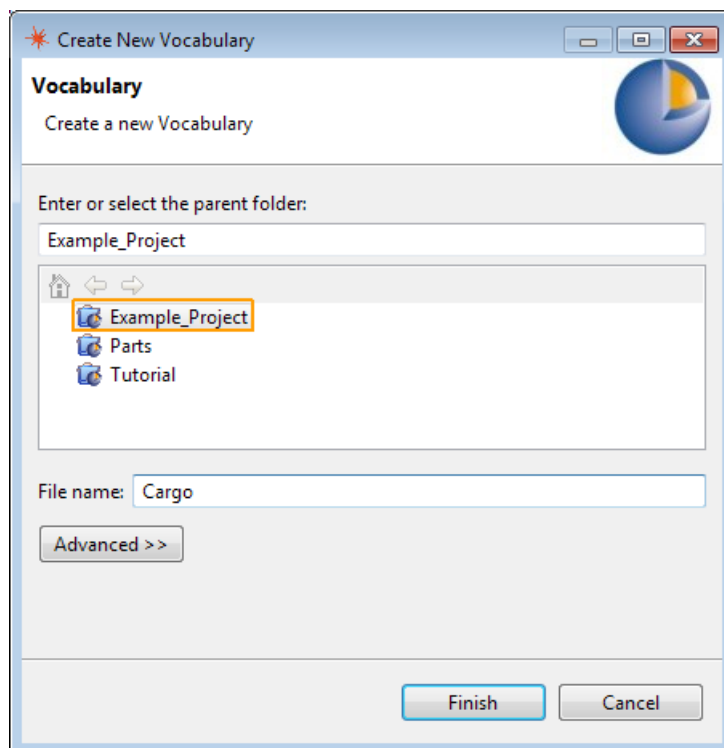
Creating a Vocabulary

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**.

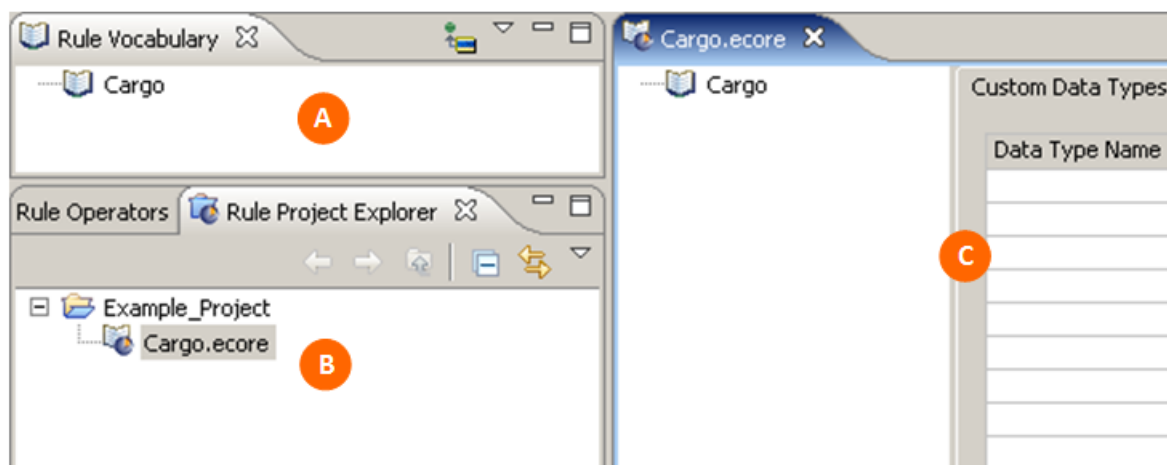
These techniques all launch the same **Create a New Vocabulary** wizard, illustrated below.



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** entry area. It is not necessary to type the file extension `.ecore` (we used `Cargo` here).

Note: The **Advanced** options are not relevant to Corticon and should not be used.

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 menu is available when the Vocabulary editor is the active window.

The **Vocabulary** menu has the following items:

- **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 from the connected data source.
 - **Clear Database Metadata** - Removes any database metadata previously imported.

Note: For more information, see the section "*Mapping and validating database metadata in the Integration and Deployment Guide*".

- **Clear Database Mappings** - Removes specific database mappings whether imported or manually defined.

Note: For more information, see the topic "*Clearing database mapping in the vocabulary in the Rule Modeling Guide*".

- **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 disabled if it is not supported for that brand.

- **Import Enumeration Elements** - Imports enumeration elements from each data type's **Lookup Table Name** in the connected data source.
- **Export Database Access Properties** - Outputs database access connection information for use in creation of deployment descriptors.
- **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.







- **Clear Java Class Metadata** - Removes imported Java class metadata from memory.
- **Set to Read Only | Set to Read/Write** - Toggles the Vocabulary to limit it from making any changes and to free it to add/modify/delete elements, database access, and custom data types.
- **Show Vocabulary Details | Hide Vocabulary Details** - Toggles the Vocabulary to show or hide details.
- **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 the Vocabulary editor is active, its tools are added to the toolbar, as shown:



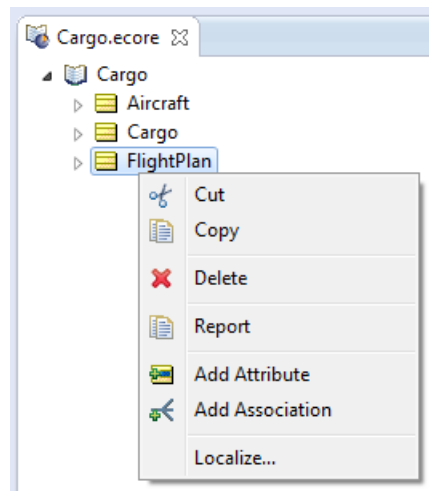
The Vocabulary tools provide the same functions as the corresponding **Vocabulary** menu commands:

-  **Vocabulary > Add Domain.**
-  **Vocabulary > Add Entity.**
-  **Vocabulary > Add Attribute.**
-  **Vocabulary > Add Association.**
-  **Vocabulary > Show Vocabulary Details**  **Vocabulary > Hide Vocabulary Details**
- Toggles the associated Vocabulary to show or hide details.

Commands on the Vocabulary context-sensitive 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.



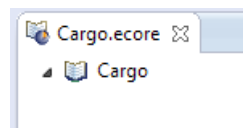
Important: The right-click pop-up menus are context-sensitive, meaning not all options are available for all Vocabulary nodes. For example, the "Add Attribute" option is only available when an Entity is selected in the Vocabulary tree.

Populating a new Vocabulary

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 an "open book" icon to its left, as shown:



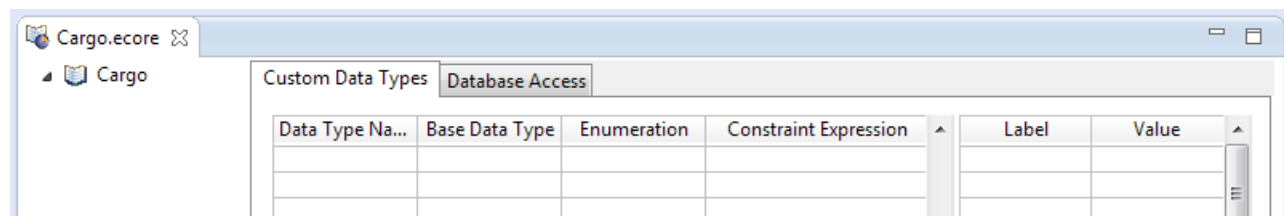
Vocabulary custom data types and database access

Each type of node in the Vocabulary (including the name) has its specific 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 are:

- **Custom Data Types** - You can define data types to offer lists of values in tests and branches, or to be bound by ranges of values specified in an expression. See [Enumerated values](#) on page 45 for more information. Also, refer to the "Building the Vocabulary" chapter of the *Rule Modeling Guide* more on using Custom Data Types.

- **Database Access** - Corticon Studio enables database access through Corticon's Enterprise Data Connector (EDC), a feature that will require additional licensing on deployment servers. For more information, refer to *Corticon Tutorial: Using Enterprise Data Connector (EDC)*.



Adding nodes to the Vocabulary tree view


Vocabulary node naming restrictions

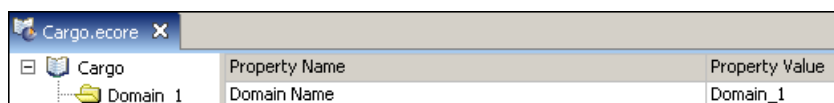
Use the following guidelines when creating new nodes in the Vocabulary:

- Domain, Entity, Attribute, and Association Role node names can contain only alphanumeric characters (letters and numbers), and underscores. However:
 - Domain, Entity, Attribute, and Association Role node names can **not** begin with a number.
 - Spaces are **not** allowed in Domain, Entity, Attribute or Association node names. If the preferred name is a combination of multiple words, then the underscore characters can be used to combine them into a single-word name.
 - Domain, Entity, Attribute, and Association Role node names can begin with an underscore.
 - Domain, Entity, Attribute, and Association Role node names can begin with either an upper or lower case letter.
- No two Entities in the same Domain can have the same node name (case-insensitive).
- Names of Operators (such as `sum`, `size`, `month`, `now`, `today`) or Extended Operators (if any) can **not** be used as Entity or Domain node names. They *can* be used as Attribute or Association Role node names.
- Custom Data Type names cannot 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 cannot match (case-sensitive) the node names of Entities in the Vocabulary.
- No two Attributes/Association Role nodes in the same Entity can have the same name (case-insensitive).
- No two Domains within another Domain can have the same node name (case-insensitive).
- No two root-level Domains can have the same node name(case-insensitive).
- No association can have the same name as an entity.


Domain nodes: Adding and editing domains and their properties

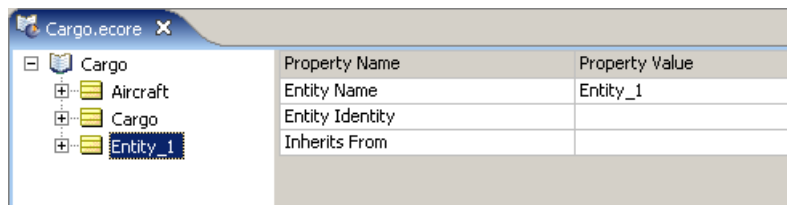
The purpose of Domains is explained in more detail in the *Rule Modeling Guide*, "Creating the Vocabulary" chapter. Generally, if all your Entity names are unique, you won't need to use Domains in your Vocabulary. To add a Domain node:

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 the Vocabulary is showing its details, 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 entity name.

When the Vocabulary is showing its details, 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.

Property	Description	Values	Applicability
Datastore Caching	Indicates whether instances of this entity are subject to caching.	Optional, only active when the entity is Datastore Persistent .	Options are: <ul style="list-style-type: none"> • No Cache or blank (default) - Disable caching. • Read Only - Caches data that is never updated • Read/Write - Caches data that is sometimes updated while maintaining the semantics of "read committed" isolation level. If the database is set to "repeatable read," this concurrency strategy almost maintains the semantics. Repeatable read isolation is compromised in the case of concurrent writes. • Nonstrict Read/Write - Caches data that is sometimes updated without ever locking the cache. If concurrent access to an item is possible, this concurrency strategy makes no guarantee that the item returned from the cache is the latest version available in the database.
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.


Property	Description	Values	Applicability
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.
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.

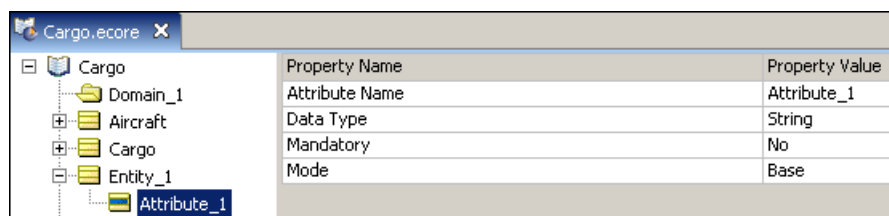
Table 4: Description of the Identity Strategy values

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. For the current list of supported RDBMS brands, access the web location [Progress Corticon 5.5.2 - Supported Platforms Matrix](#).

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.



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's</i> "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 Rulesheet drop-downs. For example, an attribute whose Mandatory value is No will always include a null value selection in its Rulesheet drop-downs.
Mode	<p>Choose the attribute's Mode from the drop-down list. <i>Base</i> attributes exist or are used by systems outside Corticon, 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 Server in production.</p> <p><i>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 Server in deployment.</p> <p>Transient attributes can, however, be dragged into a Ruletest's Input column and provided input values (often zero) so that calculations in tests produce results that can validate rules.</p> <hr/> <p>Note: If you export tests that use transients to XML, and then run them on a deployment Server, the transients are ignored.</p> <hr/>

When the Vocabulary is showing its details, 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 the Vocabulary is showing its details, additional Attribute database properties are available.

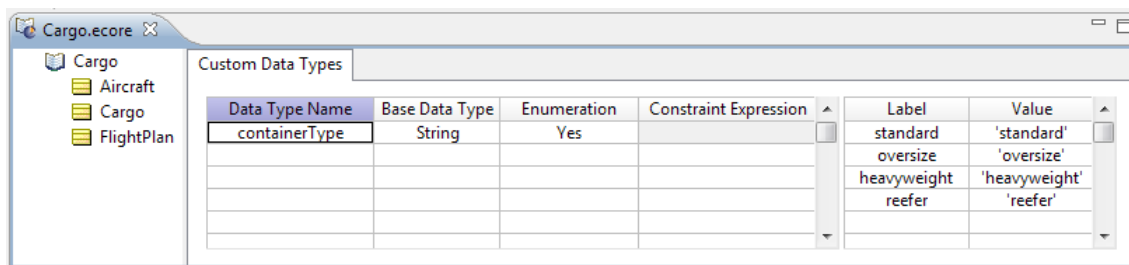
Table 7: Enterprise Data Connector (EDC) Attribute Properties

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.

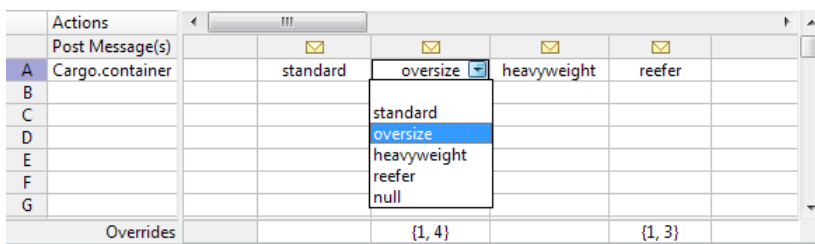
Property	Description	Values	Applicability
Value Table Name Column Name	The name of the column in the identity table that is used as the key (this column will contain the name of the entity). Chosen from a drop-down list of all columns in the table selected in the Table Name field.		Only enabled if attribute 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:

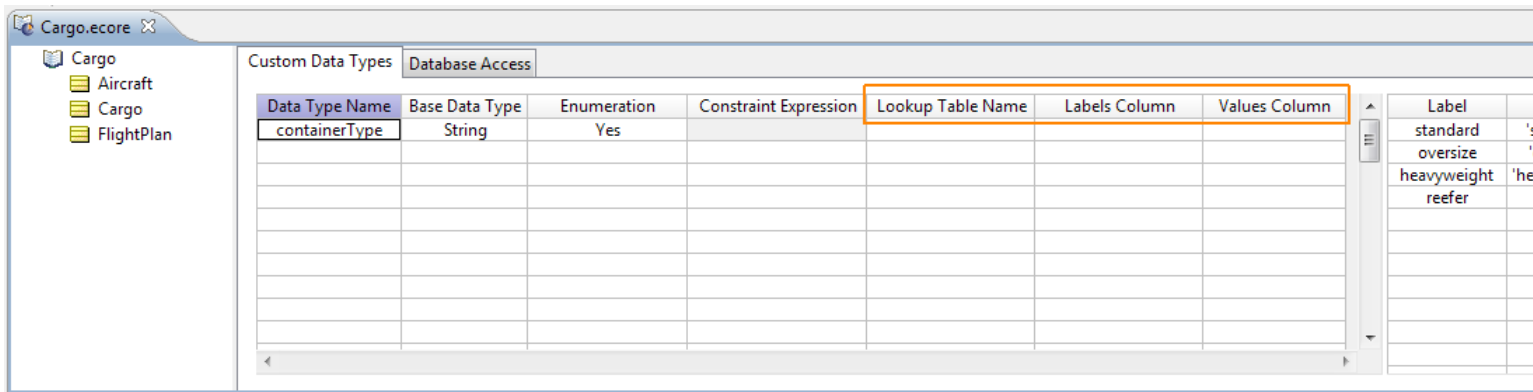


Then, when you are in the Rulesheet, the defined values -- as well as `null` and `blank` -- were offered.



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:

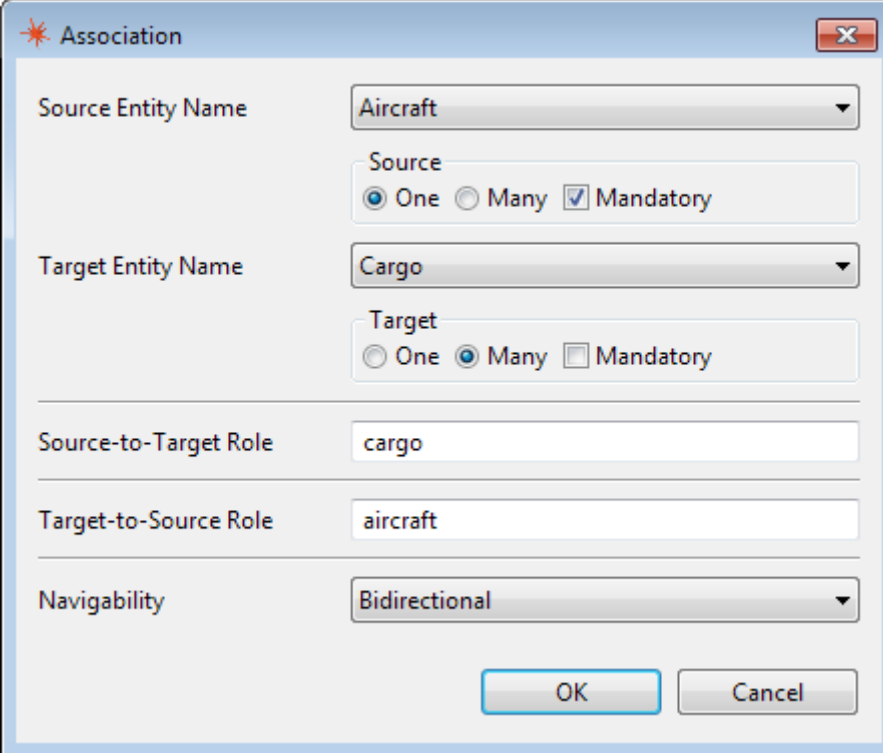
- *"Enumerations defined in the Vocabulary" in the Rule Modeling Guide*
- *"Enumerations retrieved from a database" in the Rule Modeling Guide*
- *"Importing an attribute's possible values from database tables" in the Integration and Deployment Guide*
- *"Mapping database tables to Vocabulary Entities"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



The image shows a dialog box titled "Association" with a close button (X) in the top right corner. The dialog contains the following fields and options:

- Source Entity Name:** A dropdown menu with "Aircraft" selected.
- Source:** Radio buttons for "One" (selected), "Many", and a checked checkbox for "Mandatory".
- Target Entity Name:** A dropdown menu with "Cargo" selected.
- Target:** Radio buttons for "One", "Many" (selected), and an unchecked checkbox for "Mandatory".
- Source-to-Target Role:** A text field containing "cargo".
- Target-to-Source Role:** A text field containing "aircraft".
- Navigability:** A dropdown menu with "Bidirectional" selected.
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

Note: This dialog shows the settings displayed when the Vocabulary is showing its details.

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 second 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	<p>Role names are useful when two entities share more than one association. Custom role names can give each association a unique, descriptive name.</p> <p>Source-To-Target provides a name for the association from the perspective of the source entity.</p> <p>Target-To-Source provides a name for the association from the perspective of the target entity.</p>
Navigability	<p>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.</p> <p>Source Entity < Target Entity. Use this option to prevent the association from the Target entity to 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>.</p> <p>Source Entity < Target Entity. Use this option to prevent the association from the Source entity to 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>.</p> <p>See the <i>Rule Modeling Guide</i> for more information on using associations in rule modeling.</p>

When the Vocabulary is showing its details, additional Association properties are available, as highlighted:

The screenshot shows the 'Association' dialog box with the following fields and values:

- Source Entity Name:** Aircraft
- Source:** ☒ One ☐ Many ☒ Mandatory
- Target Entity Name:** Cargo
- Target:** ☐ One ☒ Many ☐ Mandatory
- Source-to-Target Role:** cargo
- XML Property Name:** (empty)
- Java Object Get Method:** (dropdown)
- Java Object Set Method:** (dropdown)
- Java Object Field Name:** (dropdown)
- Target-to-Source Role:** aircraft
- XML Property Name:** (empty)
- Java Object Get Method:** (dropdown)
- Java Object Set Method:** (dropdown)
- Java Object Field Name:** (dropdown)
- Navigability:** Bidirectional

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.

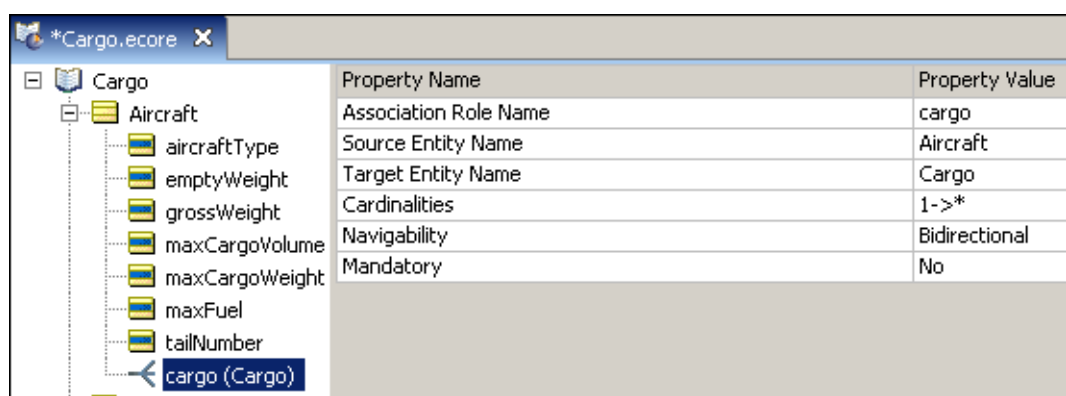
When set to Integration & Deployment mode, additional Association database properties are available.

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

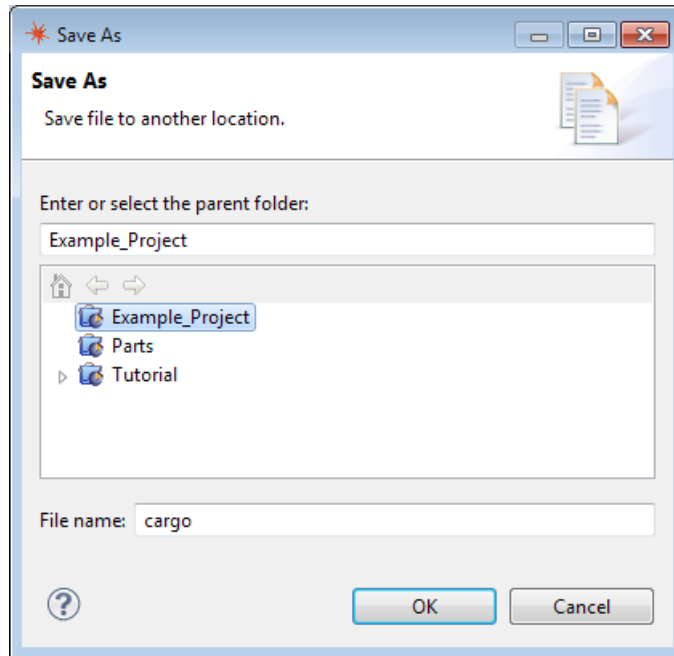
1. Select an association on the Vocabulary tree to display its properties.
2. Edit properties as described in [Association nodes: Adding and editing associations and their properties](#).



Saving a new Vocabulary

1. Do one of the following:

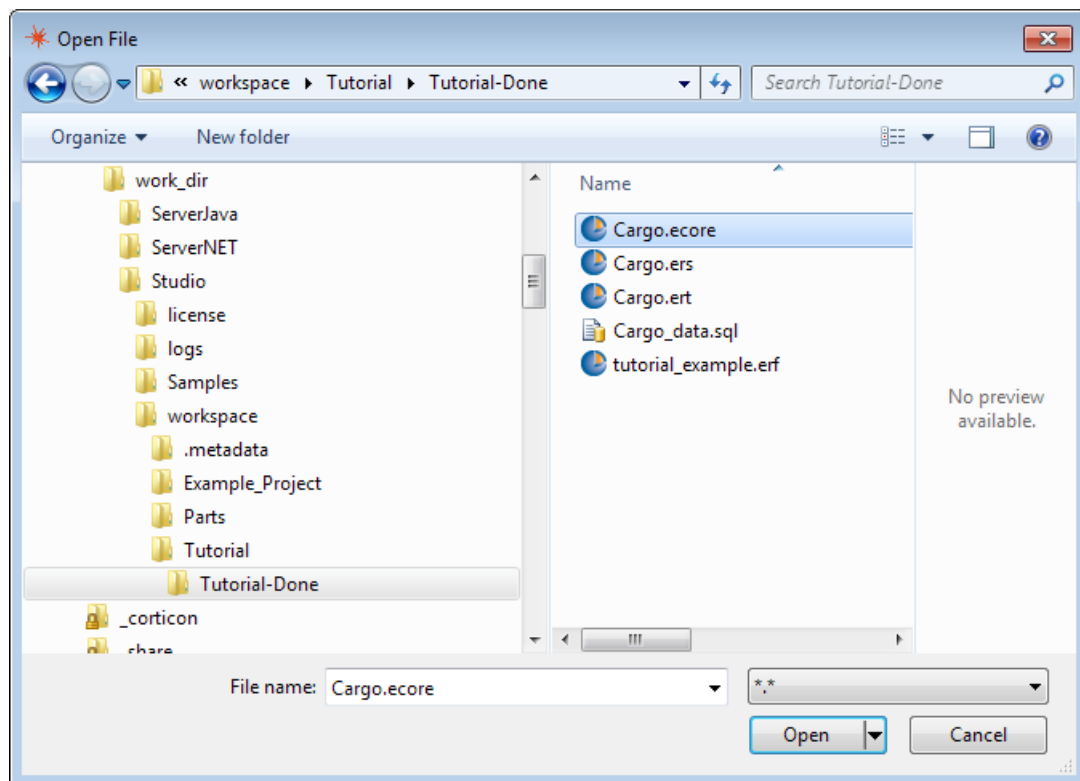
- Save the Vocabulary from the toolbar or choose **File > Save** from the menubar to save any changes you have made.
- Use **File > Save As** on the menubar and Navigate to the directory where you want to store the Vocabulary, or click  to create a new directory.



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

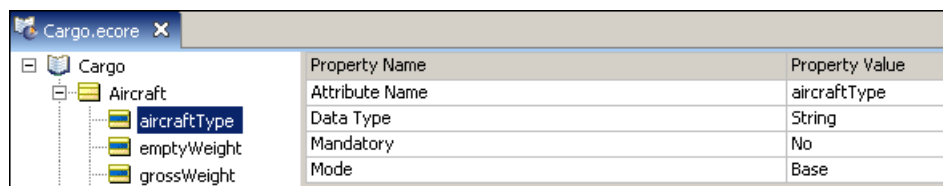
1. Choose **File > Open File** from the Corticon Studio menubar or use the toolbar to display the **Open** window.



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 below.
 - It is a good idea to close any open editors for Rulesheets, Ruleflows and Ruletests in the project before changing the Vocabulary.
 - When modifying an entity, attribute, or association's properties in the Vocabulary -- such as changing the data type of an attribute -- consider the impact in conditions and actions in the project's Rulesheets, as well as in Ruleflows and Ruletests.

- Renaming an entity, attribute, or association does not automatically perpetuate to the project's Rulesheets, Ruleflows and Ruletests; therefore, renaming should be avoided or performed very carefully.
- Deleting an entity, attribute, or association from the Vocabulary can impact conditions and actions in the project's Rulesheets, as well as Ruleflows and Ruletests. When reviewing Rulesheets after a Vocabulary deletion, choose **Advanced** view to reveal any lingering references to the deleted items in the **Scope** section.
- After editing the vocabulary, save it before opening other editors. Then open the project's Rulesheets, Ruleflows and Ruletests to expose errors that might have resulted from the changes made to the Vocabulary.

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](#)

- [Commands on the Rulesheet context-sensitive menu](#)
- [Rulesheet sections](#)
- [Rule statements window](#)
- [Rulesheet properties](#)
- [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](#)

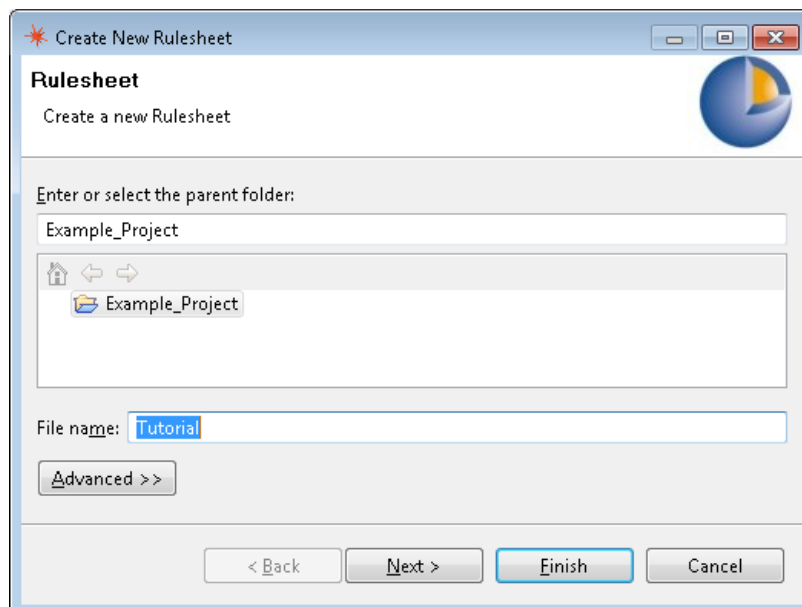
Creating a new Rulesheet

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.

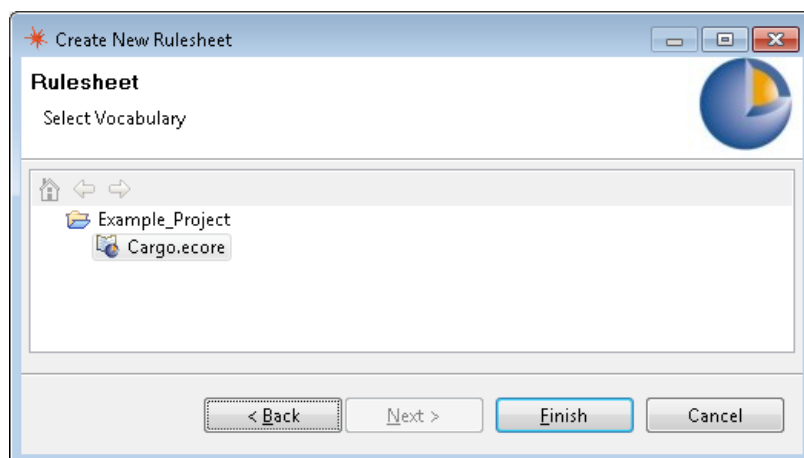


3. Highlight the Rule Project you would like to associate the new Rulesheet with in the list (or enter it manually in the **Enter or select parent folder** entry area.
4. Enter a file name for the Rulesheet in the **File name:** entry area.

Note: The **Advanced** options are not relevant to Corticon and should not be used.

5. Click **Next** to continue.

Figure 3: The Create New Rulesheet Wizard window

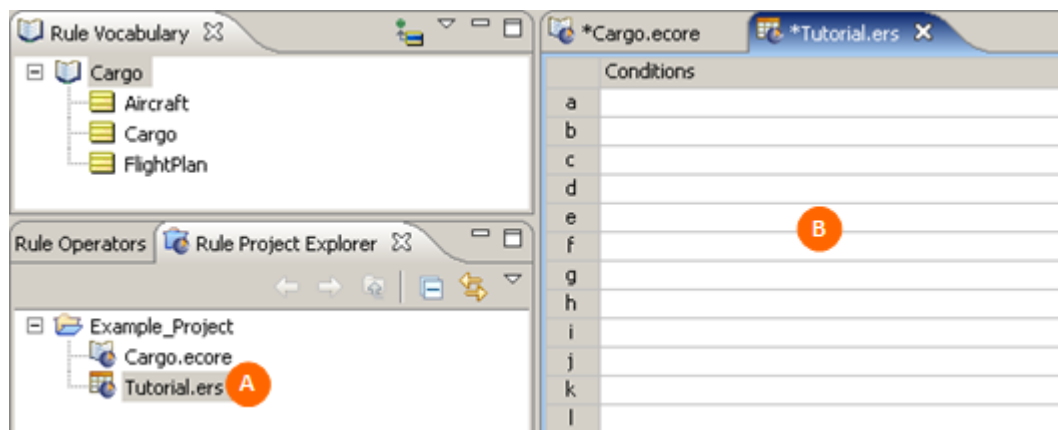


6. Select the Vocabulary with which to associate the Rulesheet. A Rulesheet can only be associated with one Vocabulary. In this example, we expanded the `Example_Project` folder and chose the `Cargo.ecore` file we created earlier. As you create additional Vocabulary files within a Rule Project, they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

7. 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 4: 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 menu is available when the Rulesheet editor is the active window.

The **Rulesheet** menu has the following items:

- **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\eclipse\plugins\com.corticon.services_n.n.n\DepGraph.`
 - **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\eclipse\plugins\com.corticon.services_n.n.n\DepGraph.`
 - **Clear Analysis Results** - Removes highlighting that resulted from checking for conflicts and completeness.
 - **Check for Logical Loops** - Performs logical loop detection across all rules in all Rulesheets in the active Project.
 - **Check for Completeness** - Highlights incompleteness in the active Rulesheet.
 - **Check for Conflicts** - Highlights conflict between two or more rules in the active Rulesheet.
 - **Enable Conflict Filter** - A toggle that either hides or shows all rule columns that are not part of the current conflict.
 - **Previous Conflict** - Highlights the previous conflict in the sequence. This option is grayed out until you perform a conflict check.
 - **Next Conflict** - Highlights the next conflict in the sequence. This option is grayed out until you perform a conflict check.
 - **First Conflict** - Highlights the first conflict in the sequence. This option is grayed out until you perform a conflict check.
 - **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. For more information on this topic, see the section *"Rule dependency: Chaining and looping" in the Rule Modeling Guide*.
 - **Expand Rules** - Creates multiple simple rules from each complex rule.
 - **Collapse Rules** - Reverses the Expand Rules function.
 - **Compress Rules** - Detects overlapping conditions between rules and combines columns to produce a smaller number, but logically equivalent, set of 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.





- **Show Vocabulary Details | Hide Vocabulary Details** - Toggles the associated Vocabulary to show or hide details.
- **Show Natural Language** - Displays the Condition and Action expressions in the Rulesheet that have natural language equivalents.
- **Filters** - For more information on filters and preconditions, see the topics in the section *"Filters and Preconditions" in the Rule Modeling Guide*.
 - **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.
 - **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** - The selected option affects only the active Rulesheet. For more information on the processing mode options, see the topics in the section *"Rule dependency: Chaining and looping" in the Rule Modeling Guide*.
 - **Optimized Inferencing** - Default selection. Causes rules to execute without re-evaluation or re-execution (no looping).
 - **Advanced Inferencing** - Allows only multi-rule inter-dependencies to re-evaluate and re-execute.
 - **Advanced Inferencing with Self-Triggering** - Allows all rule dependencies (including self-dependencies) to re-evaluate and re-execute.
- **Localize** - Opens the **Localization** view. For more information, see the topic *"Localizing Corticon Studio" in the Rule Modeling Guide*.
- **Report** - Creates an HTML report and launches your browser for viewing. See [Creating a Rulesheet Report](#).

Rulesheet toolbar

When a Rulesheet editor is active, its tools are added to the toolbar, as shown:



The Rulesheet tools provide the same functions as the corresponding **Rulesheet** menu commands:

-  **Simple View**  **Advanced View** - Toggles to the advanced or simple Rulesheet views. New Rulesheets are shown in Simple view.
-  **Show Vocabulary Details**  **Hide Vocabulary Details** - Toggles the associated Vocabulary to show or hide details.

-  - 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.

Commands on the Rulesheet context-sensitive menu

In addition to the menubar and toolbar functions described above, Corticon Studio also provides many functions in a right-click pop-up menu. A Rulesheet context-sensitive menu opens when a section in a Rulesheet is right-clicked. Its menu provides access to Rulesheet functions relevant in the context of the sections or items where it was accessed.

Commands	Description
Undo / Redo	Undo reverts from the previous action. Redo restates an action that was undone.
Cut / Copy / Paste	In any section, performs these standard edit functions.
Select All	In all sections except the Scope, selects all active rows (and columns) in the Rulesheet section in which the menu was activated.
Disable / Enable	On any area except Vocabulary elements, Disable greys out one or more selected entries / lines / columns / cells so that they are virtually deleted. Selecting one or more disabled items lets you Enable them again.
Insert Row / Remove Row / Add Rows to End	Adds or deletes rows. Adds ten rows the bottom of the current set in the Rulesheet section in which the menu was activated.
Insert Column / Remove Column / Add Columns to End	Adds or deletes rows. Adds ten columns to the right of the current set.
Localize	Opens the Localization view. The menu command Rulesheet > Localize performs the same operation. For more information, see the topic <i>"Localizing Corticon Studio" in the Rule Modeling Guide</i> .
Natural Language	Opens the Natural Language view. The menu command Rulesheet > Show Hide Natural Language is a toggle that displays the natural language entries in the Rulesheet without opening the Natural Language window. For more information, see the topic <i>"Working with rules in natural language" in the Rule Modeling Guide</i> .

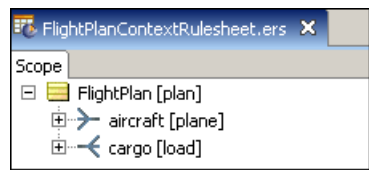
Rulesheet sections

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 5: Scope section, showing an Entity and its alias



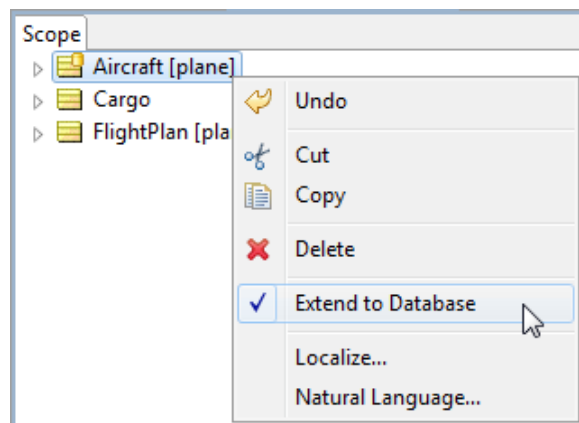
The **Scope** section provides a reduced set of Vocabulary terms with which to build a **Rulesheet**.

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 an entity has been set to **Datastore Persistent** in the Vocabulary's details view, it is decorated with a database symbol. In the Rulesheet, that means you provide an alias and then set the entity to **Extend to Database**, as shown:

Figure 6: 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" section of the *Rule Modeling Guide* and the "Implementing EDC" section of the *Integration and Deployment Guide*.

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

Filters

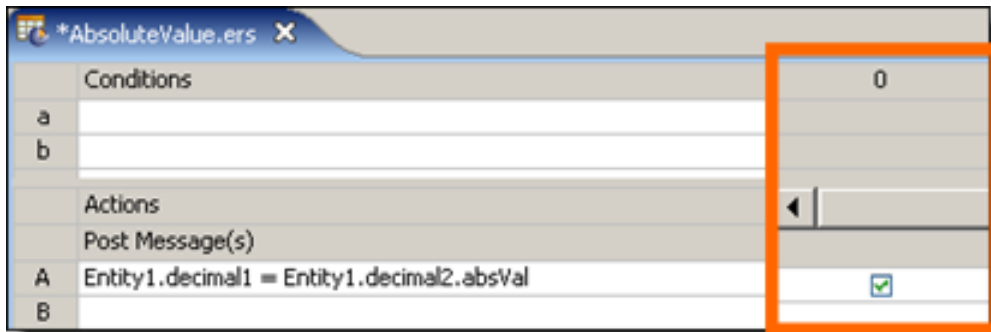
This section is visible only when the **Advanced View** is toggled (the button  on the Corticon Studio toolbar.)

Filters	
1	
2	
3	
4	
5	

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. Each Action row constitutes a separate Nonconditional rule.

Important: In versions of Corticon Studio prior to 5.2, a separate Nonconditional Rules section contained this type of logic.

Rules

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

Conditions		0	1	2
a	Cargo.weight	-	150000 .. 200000	-
b	Cargo.volume	-	-	< 300
c				
d				
e				
Actions				
Post Message(s)				
A	Cargo.packaging		Container	Pallet
B				
C				
D				
E				
Overrides				

Annotations in the image: 1 points to the Conditions section, 2 points to the Actions section, 3 points to the blue-highlighted cells in column 2 of the Conditions table, and 4 points to the blue-highlighted cells in column 2 of the Actions table.

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:

Ref	ID	Post	Alias	Text
1		Info	Cargo	Cargo weighing between 150000 and 200000 kilograms must be packaged in a container
2		Info	Cargo	Cargo with a total volume less than 300 cubic meters 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

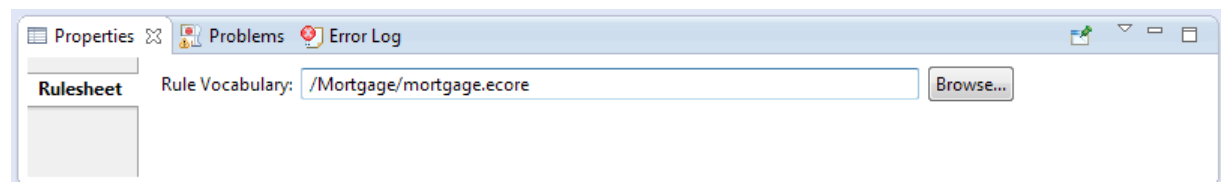
Allows you to assign a category name to each Rule Statement and the rule model to which it links. This field is optional.

Comments

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


Rulesheet properties

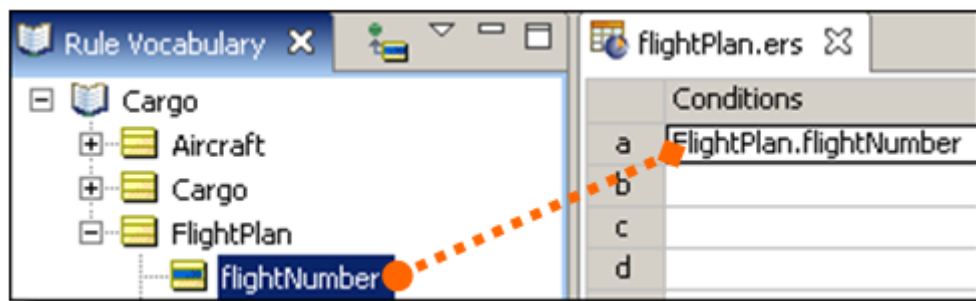
When you are editing a Rulesheet, the **Properties** tab's **Rulesheet** displays the Rule Vocabulary that supports this Rulesheet.



Note: A Rulesheet is tightly bound to its Vocabulary. While you are allowed to browse to select another Vocabulary, that action is a reserved for advanced users who might be applying an updated version of the same Vocabulary, perhaps as carefully changed by a project collaborator.

Using the business vocabulary to build rules

Select the **Rule Vocabulary** tab; click the  button 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*.

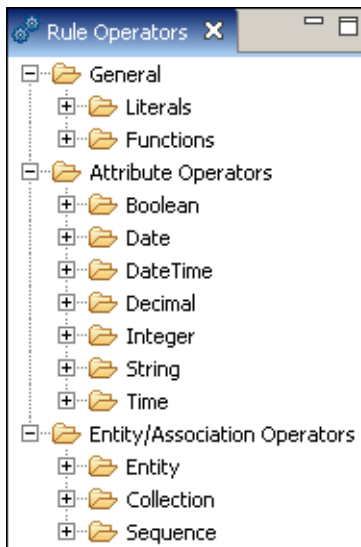
Using the operator vocabulary to build rules

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 can be named when created and renamed later by:

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

Rulesheet names must adhere to and comply with the guidelines shown in the [File naming restrictions](#) on page 20 section of this document.

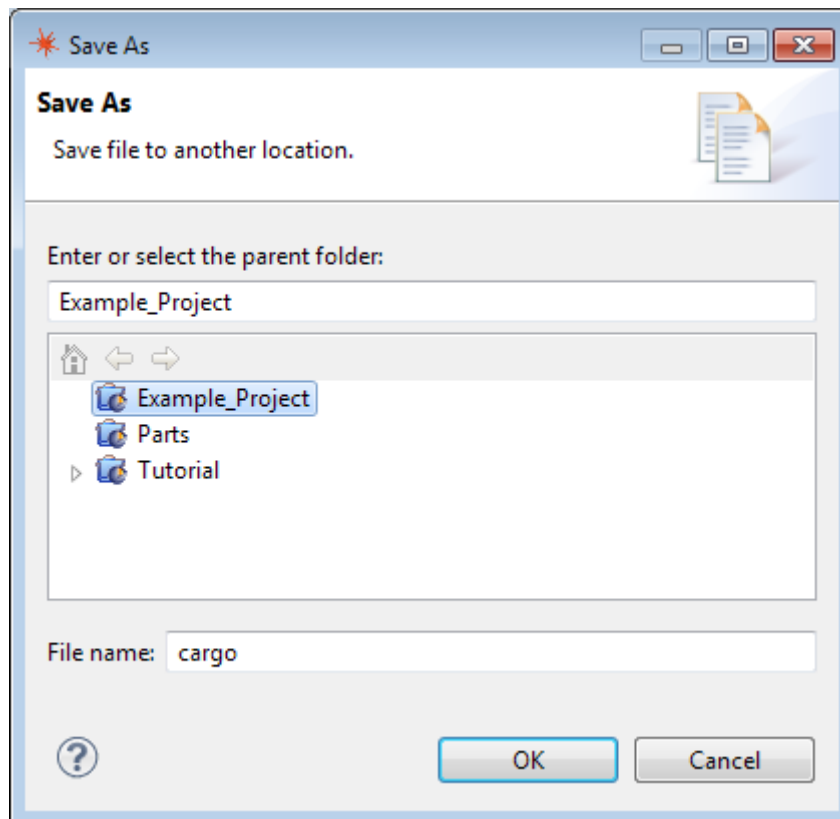
Deleting Rulesheets

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 then press the **Delete** key.

Saving a new Rulesheet

1. With the Rulesheet selected, choose the Save command or click the save button.
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](#) on page 20.



Validating and optimizing a Rulesheet

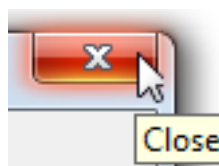
Conflict, Completeness, and Logical Loop Checkers, as well as the Compress Rules feature, are collectively known as validation and optimization functions. These topics are addressed briefly in the *Corticon Studio Tutorial: Basic Rule Modeling*, and in more depth in the *Corticon Studio: Rule Modeling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

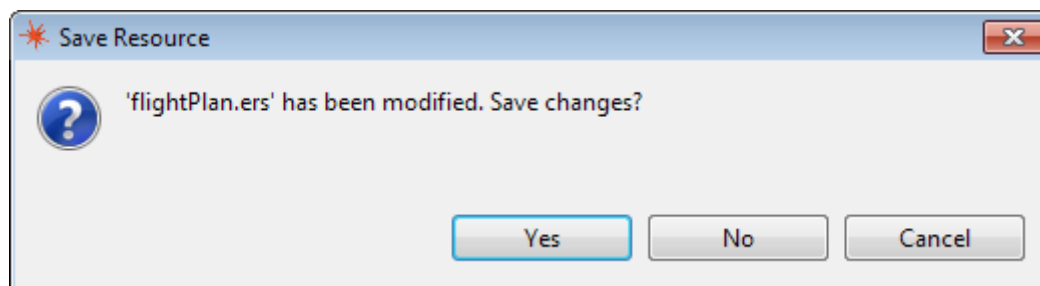
Creating a Rulesheet report

1. Select the menu command **Rulesheet > Report**.
2. The Rulesheet report should open as a new HTML page in your default web browser.
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 file 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. To close a Rulesheet, click its Close button,  , or select the menu command **File > Close**.



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 where they will be invocable as Decision Services.

The Ruleflow concept, introduced in Corticon Studio 5.0, 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`.

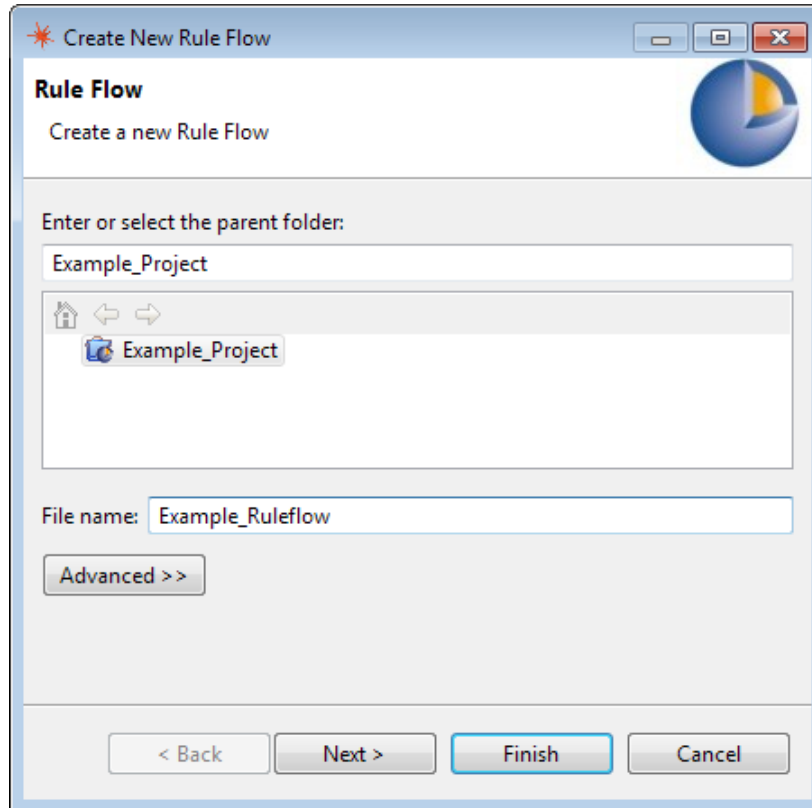
For details, see the following topics:

- [Creating a new Ruleflow](#)
- [Ruleflow window](#)
- [Ruleflow properties](#)
- [Ruleflow preferences](#)

Creating a new Ruleflow

Follow these steps to create a new Ruleflow:

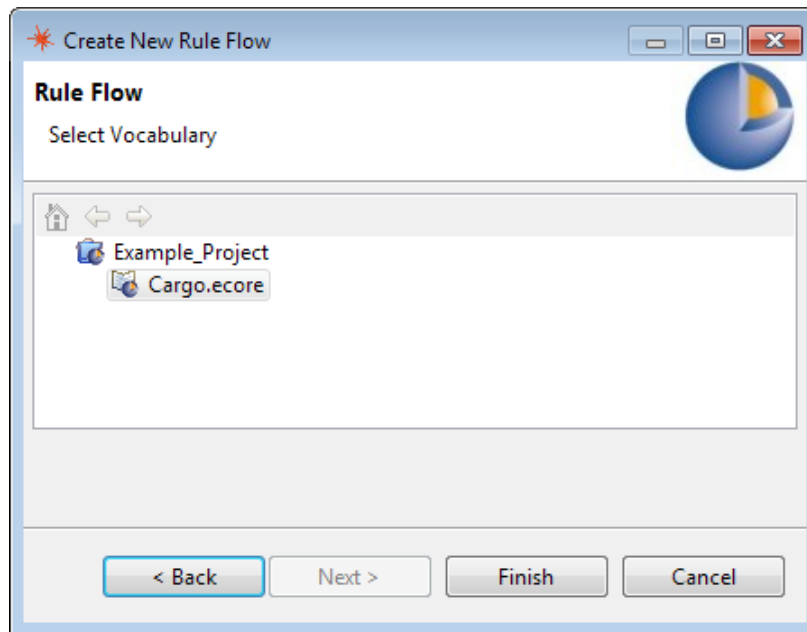
1. Choose **File > New > Ruleflow** from the menubar or click the down arrow next to the **New** icon on the toolbar and select **Ruleflow**. Either method will launch the **Create a New Ruleflow** wizard.



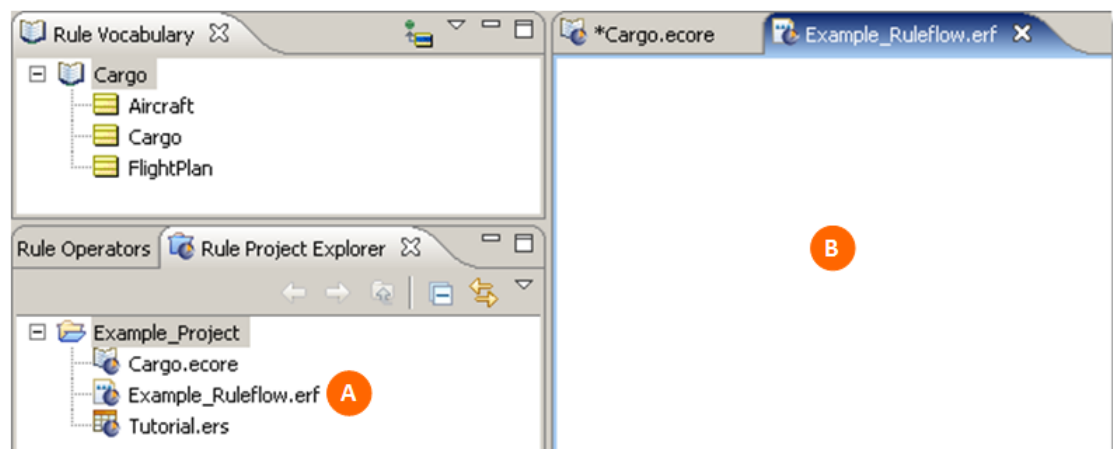
2. On the **Project** list, select the Project you want to associate with the new Ruleflow (or enter it manually in the **Enter or select parent folder** entry area).
3. Enter a file name for the Ruleflow in the **File name** entry area.

Note: The **Advanced** options are not relevant to Corticon and should not be used.

4. Click **Next** to continue.



5. Select the Vocabulary with which you want to associate the Ruleflow. A Ruleflow can only be associated with one Vocabulary. Here, we simply expand the `Tutorial` folder and highlight the `Cargo.ecore` file. 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.



6. 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).

The Ruleflow menubar and toolbar are displayed instead of the Vocabulary menubar and toolbar. You are now ready to begin modeling your sequence of Rulesheets.

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.

Naming Ruleflows

Ruleflow names must adhere to and comply with the guidelines shown in the [File naming restrictions](#) on page 20 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

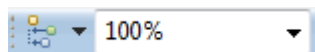
The following menu is available when the Ruleflow editor is the active window.

The **Ruleflow** menu has the following items:

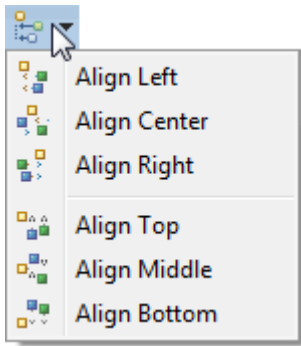

- **Properties** - Opens the Ruleflow properties window.
- **Report** - Creates an HTML report and launches your browser for viewing. See [Creating a Ruleflow Report](#).

Ruleflow toolbar

When the Ruleflow editor is active, a few of its tools are added to the toolbar, as shown:



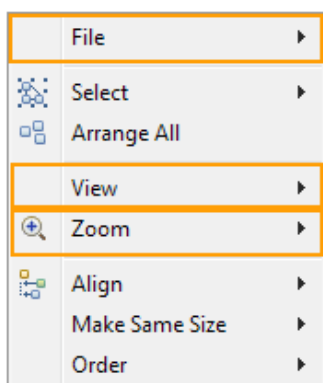
These functions can also be accessed on the context menu that opens when you right-click on the Ruleflow canvas.

	Aligns multiple selected objects on the Ruleflow canvas.
	Zooms the Ruleflow window to the specified magnification.

Editor commands on the Ruleflow context-sensitive menu

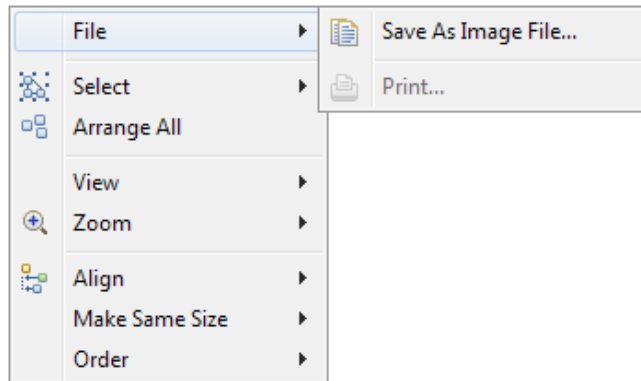
The Ruleflow pop-up menu opens when you right-click on a Ruleflow's canvas. This menu provides quick access to many frequently used functions specific to managing the file and the canvas.

See [Object commands on the Ruleflow context-sensitive menu](#) on page 89 for commands that affect objects on the canvas.



The three sections highlighted above are the file and canvas related menu commands on the Ruleflow pop-up menu.

File



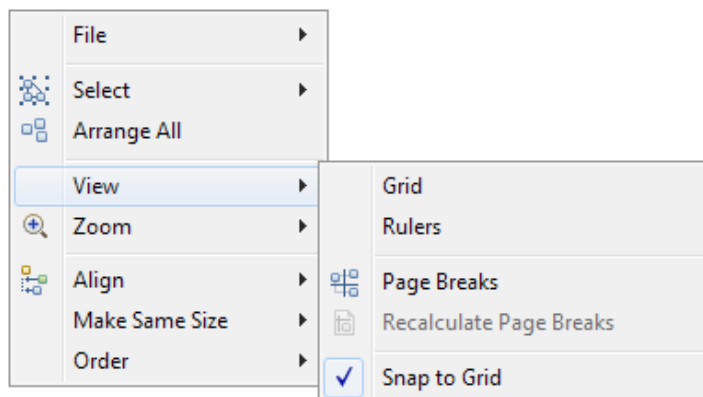
Select

See [Object commands on the Ruleflow context-sensitive menu](#) on page 89.

Arrange All

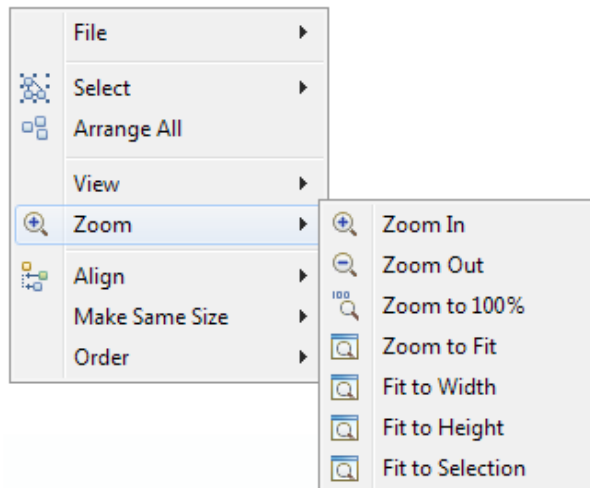
See [Object commands on the Ruleflow context-sensitive menu](#) on page 89.

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

See [Object commands on the Ruleflow context-sensitive menu](#) on page 89.

Make Same Size

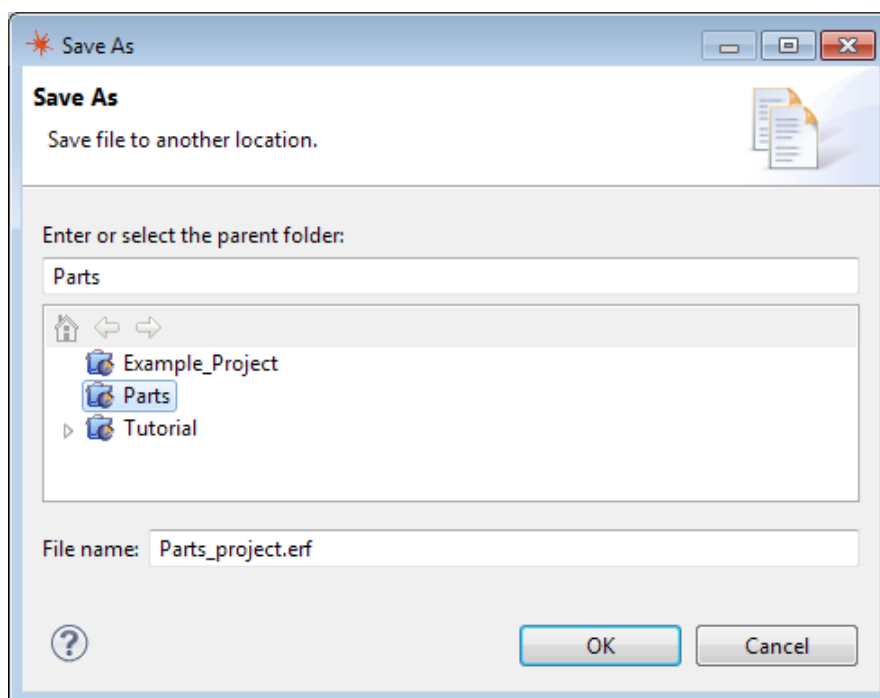
See [Object commands on the Ruleflow context-sensitive menu](#) on page 89.

Order

See [Object commands on the Ruleflow context-sensitive menu](#) on page 89.

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.

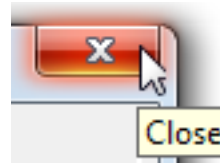


1. To rename the Ruleflow, modify the name in the **File name:** field with the **Project** folder highlighted in the **Project** list.
2. To save the Ruleflow to a different location, whether you are renaming it or not, select that **Project** folder in the **Project** list or enter the parent folder name manually in the **Enter or select the parent folder:** text field.
3. The same file naming conventions apply to Ruleflows as apply to Rulesheets. See the [File naming restrictions](#) on page 20 section of this document.

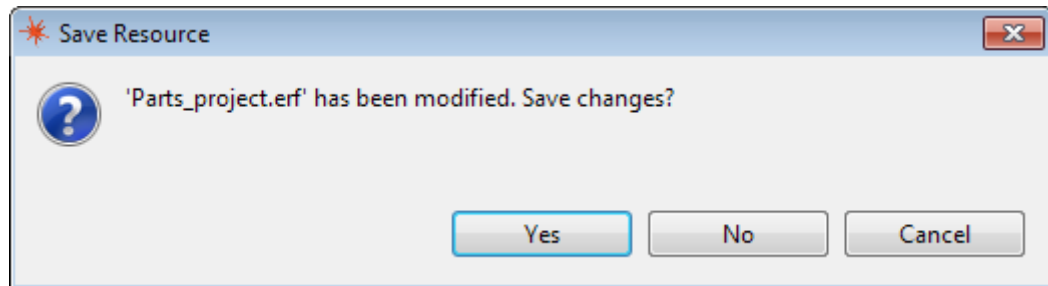
Creating a Ruleflow report

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 `CorticonRuleflowXXXXXX.html`, where `XXXXXX` 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



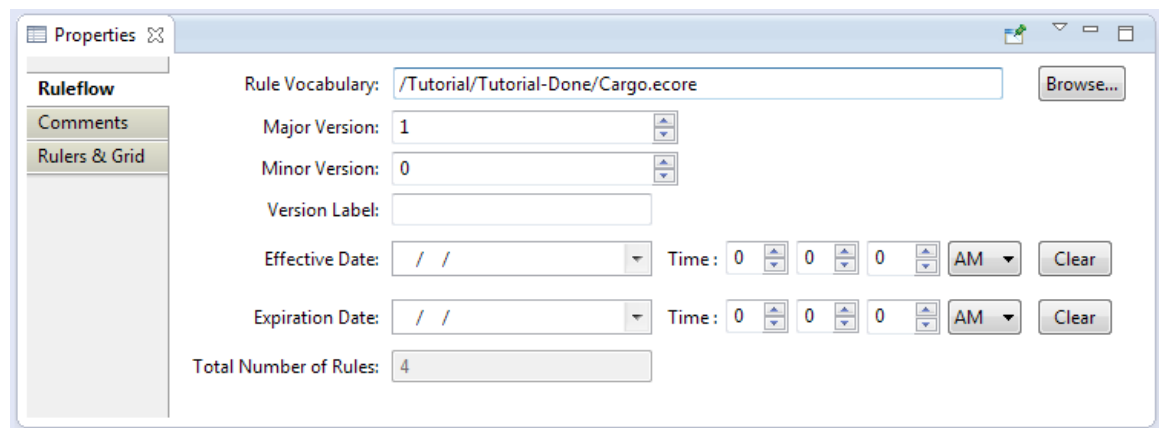
1. To close a Ruleflow, click its Close button, **File > Close**.



2. Choose **Yes** to save any changes you made.

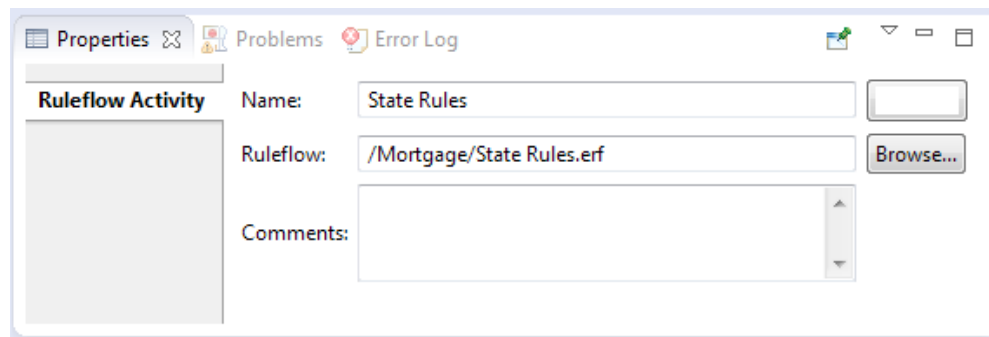
Ruleflow properties

When a Ruleflow is the active file, its properties display in the **Properties** tab at the bottom of the Corticon Studio window. This window provides "sub-tabs" arranged vertically at the left-hand side of the window.



Note:

Ruleflow within a Ruleflow - When a Ruleflow is added as a component of another Ruleflow, its **Properties** reflect that use case, providing only one tab, **Ruleflow Activity**, as illustrated:



In this context, the original file is referenced and you are allowed to browse to choose a different Ruleflow that uses the same Vocabulary. You can change the name of the Ruleflow in this context so that it provides meaning, and you can add comments. None of these actions change the Ruleflow properties of the original Ruleflow.

For more information, see *Conditional Branching in Ruleflows in the Corticon Rule Modeling Guide* and [Properties of Ruleflow objects on a Ruleflow canvas](#) on page 92.

Ruleflow

The Ruleflow tab contains information about the Vocabulary that is used by all Rulesheets, and other Ruleflows on the current Ruleflow canvas.

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.

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 Date and Expiration Date

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 7: Setting the Effective Date on a Ruleflow

The screenshot shows the 'Properties' window for a Ruleflow. The left sidebar has tabs for 'Ruleflow', 'Comments', and 'Rulers & Grid'. The main area contains the following fields:

- Rule Vocabulary:** /Tutorial/Tutorial-Done/Cargo.ecore (with a 'Browse...' button)
- Major Version:** 2
- Minor Version:** 1
- Version Label:** (empty)
- Effective Date:** / / (with a calendar dropdown)
- Expiration Date:** (empty)
- Total Number of Rules:** (empty)
- Time:** 0:00 AM (with 'Clear' button)
- Time:** 0:00 AM (with 'Clear' button)

The calendar dropdown for the Effective Date shows the month of June 2014. The days of the week are S, M, T, W, T, F, S. The dates are arranged in a grid, with the 6th of June highlighted in blue.

Once dates are set, you can specify the effective time on that specified date.

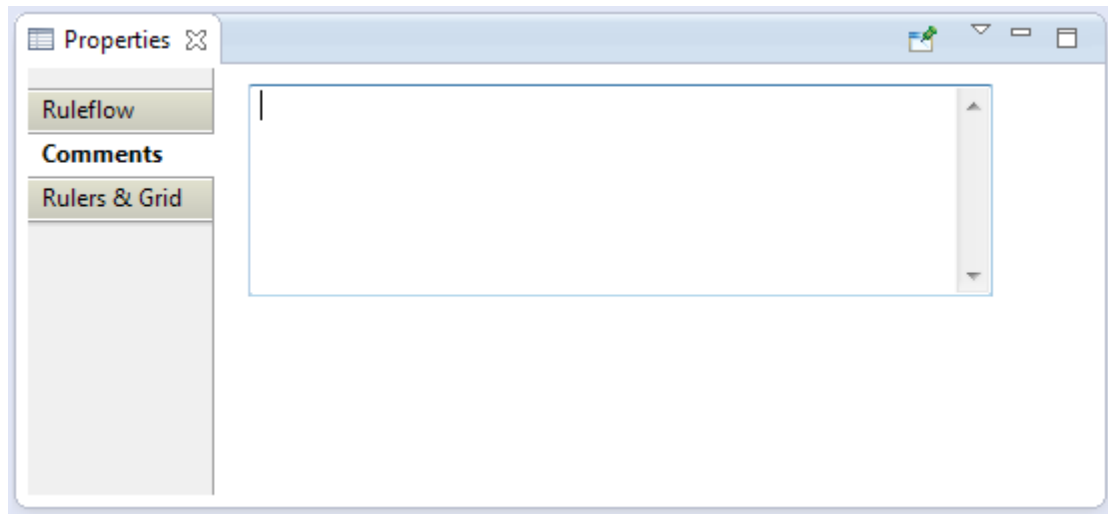
Total Number of Rules

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

The **Comments** sub-tab in the **Properties** window, as shown below, lets you add comments to a Ruleflow.

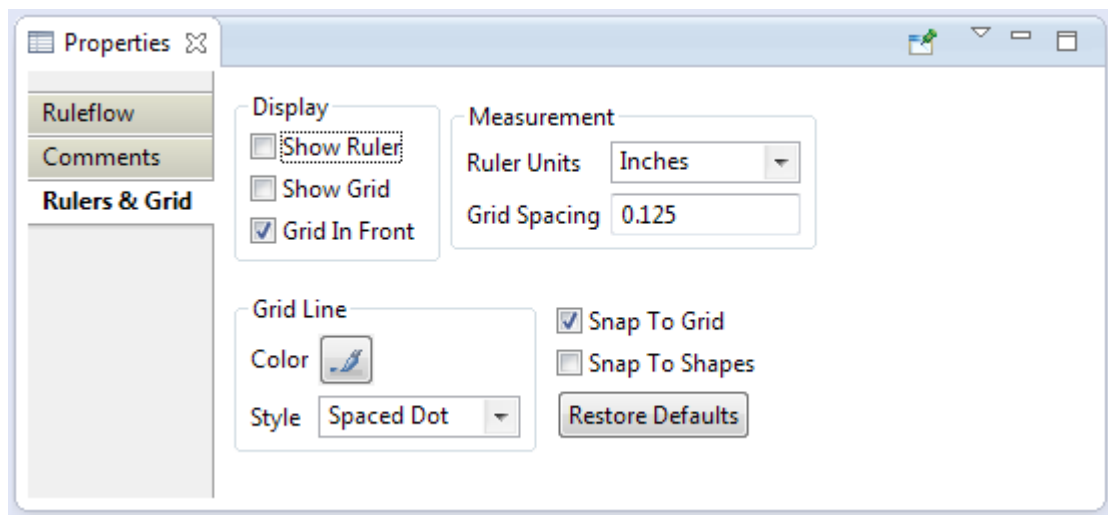
Figure 8: The Ruleflow Comments Sub-tab



Rulers and grid

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** reset the properties to the default settings, as shown:

Figure 9: The Ruleflow Rulers and Grid Sub-tab

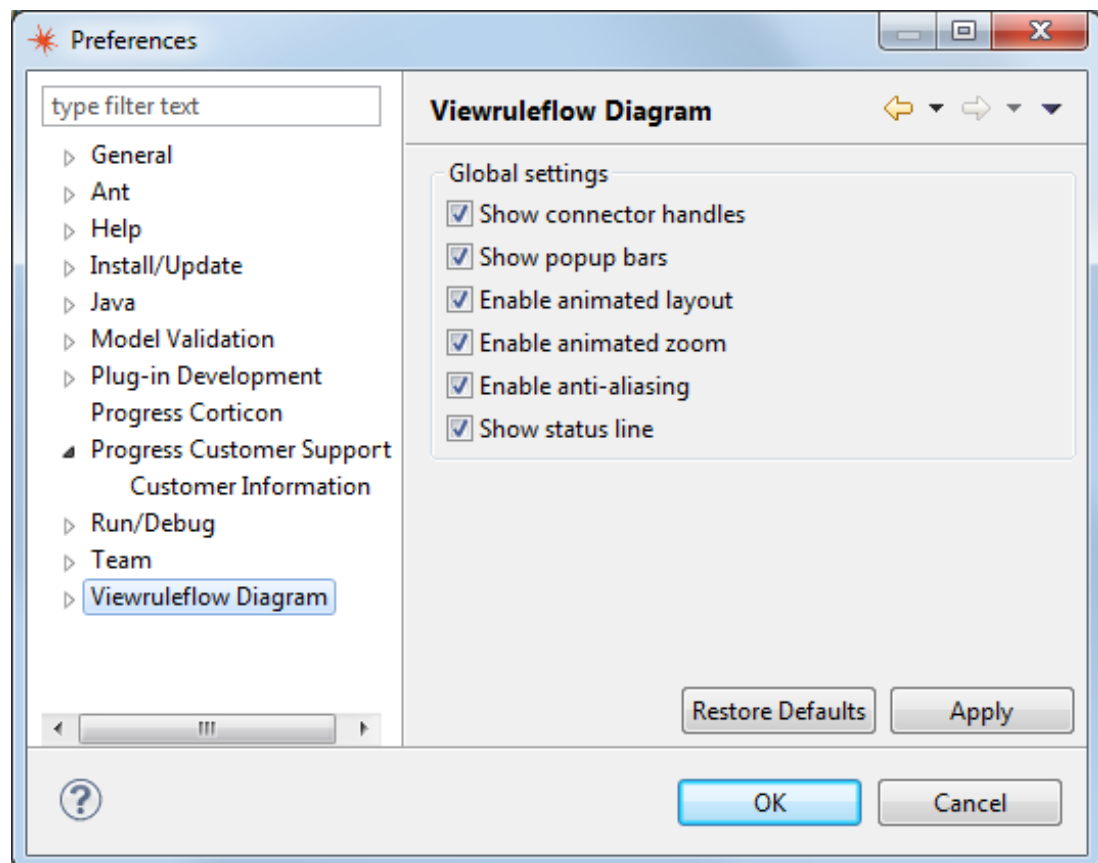


Ruleflow preferences

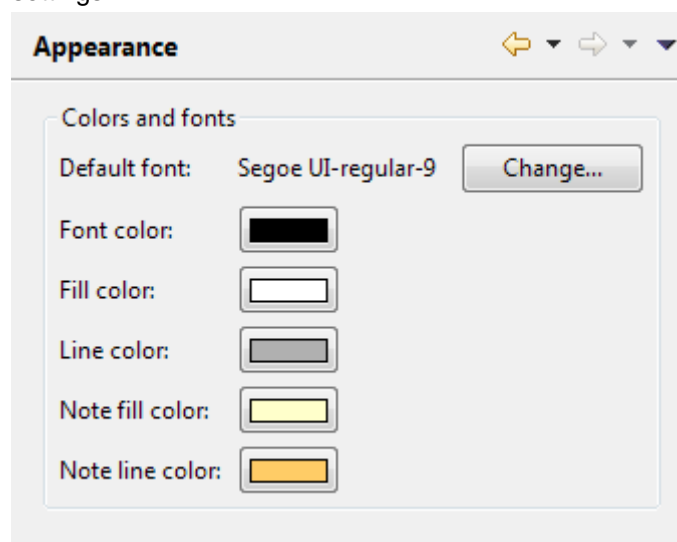
You can set many aspects of a the Ruleflow canvas and its output.

To set Ruleflow Preferences:

1. In Studio, choose the menu item **Window > Preferences**.
2. Click on **Viewruleflow Diagram**. The global settings are available as shown:



3. Expand **Viewruleflow Diagram**, and then click on each of its items to expose the corresponding settings:



The image displays three screenshots of the Progress Corticon configuration interface, arranged vertically.

Connections

Line style:

- Oblique
- Oblique
- Rectilinear

Printing

General printing settings:

Page setup

Orientation

- ☒ Portrait
- ☐ Landscape

Units

- ☒ Inches
- ☐ Millimetres

Size

Size: Letter

Width (in inches): 8.5

Height (in inches): 11

Margins

Top (in inches): 0.5

Bottom (in inches): 0.5

Left (in inches): 0.5

Right (in inches): 0.5

Rulers And Grid

Ruler options

- ☐ Show rulers for new diagram

Ruler units: Inches

Grid options

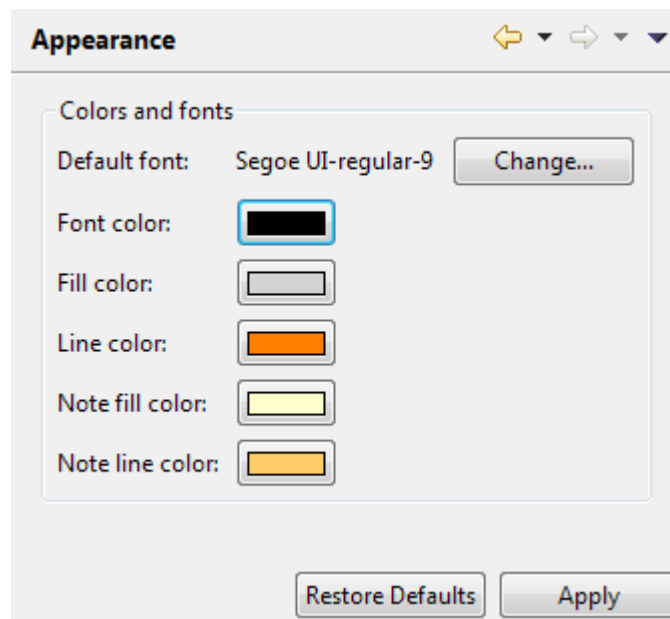
- ☐ Show grid for new diagrams
- ☒ Snap to grid for new diagrams
- ☐ Snap to shapes for new diagrams

Grid spacing (in inches): 0.125

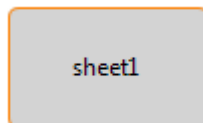
4. Adjust the settings to suit your needs, click **Apply**, and then click **OK**.

The next Ruleflow you create (as well as the next objects you define in an existing Ruleflow) will use your settings.

For example, if you modified and applied the **Appearance** preferences of Full and Line Color as follows...



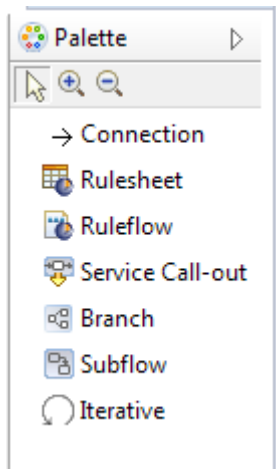
...the next Rulesheet you bring onto a Ruleflow canvas would have the following look:



Note: The color preferences you set here could have the fill color overridden by the color property on the Properties of each object on the palette, as presented in [Adding colors and comments to Ruleflow objects](#) on page 94. When an individual object's color reverts to its default value, white, the underlying preference color is used.

Objects on a Ruleflow canvas

Ruleflow diagrams are displayed on the Ruleflow canvas. The objects that can populate a Ruleflow canvas are displayed in its palette:



These objects are used as follows:

Connections

Connections are the objects that connect or "stitch" assets and objects together to 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`.

Connections are used on a canvas but are not Corticon assets. They have no properties yet can change color when you choose **Preferences > Viewruleflow Diagram > Appearance > Line color** (which also will change the object outlines as well.)

Rulesheets

Rulesheets are the essential Corticon assets 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 that use the same Vocabulary, each of which may have its own use in different business processes or applications.

Ruleflows

Just as set of rules can be defined on several Rulesheets, and then assembled into a Ruleflow, Ruleflows can also be used as modules that are assembled into a 'master' Ruleflow. This capability makes it easier to test components of a large complex solution, as well as to make the 'master' Ruleflow canvas easier to read and understand.

Service Call-Outs

Service Call-outs are a type of extension to a Ruleflow, and require significant Java development expertise. While they are not named *assets* in a project, they are named *operations* that are loaded into your Studio and Server projects. The method for building Service Call-outs is documented in detail in the *Extensions User Guide*.

Branch

Branching lets you chose an attribute that is true/false (Boolean) or defined in a list (enumeration) so that data being processed is channeled to the branch object (Rulesheet, another Ruleflow, a Subflow, or a Service Call-out) that will act on it, as well as the subsequent objects that areconnected to it the branch. Branches are powerful processing structures of Corticon assets, yet are not in themselves Corticon assets.

Subflows

Subflows provide a way to group multiple Corticon assets as a subset of a complete Ruleflow, and can be set to iterate so that all the objects in the subflow are re-executed until the values derived by their constituent rules cease changing. Subflows are a grouping mechanism of Corticon assets used on a canvas, but are not Corticon assets.

Note: While you can have multiple subflows in one Ruleflow, re-using a subflow name that has different Rulesheets and connections will evaluate the last one compiled ("farthest downstream") and use that for all instances.

Iterative

Rulesheets, Ruleflows, and Subflows can be designated for iteration by clicking the **Iterative** icon in the palette, and then clicking on the target object on the canvas. Clicking on an iterative symbol in a object selected for iteration, and then pressing **Delete** removes the iterative assignment. Iteration settings are used on a canvas but are not Corticon assets.

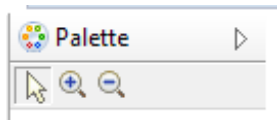
For details, see the following topics:

- [Ruleflow canvas tools](#)
- [Object commands on the Ruleflow context-sensitive menu](#)
- [Properties of Ruleflow objects on a Ruleflow canvas](#)
- [Iteration](#)

- [Adding colors and comments to Ruleflow objects](#)

Ruleflow canvas tools

The Ruleflow objects palette provides tools as well as object types:

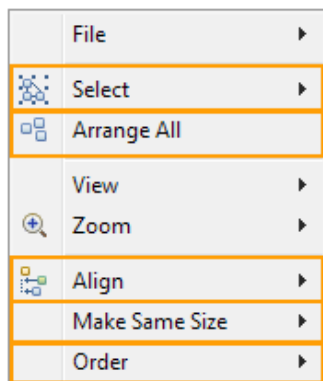


- **Select** – The default tool, the arrow cursor lets you select objects and manipulate one or more objects on the canvas.
- **Zoom** – The (+) button zooms in on the canvas; the (-) button zooms out from the canvas.

Object commands on the Ruleflow context-sensitive menu

The Ruleflow pop-up menu opens when you right-click on a Ruleflow's canvas. This menu provides quick access to many frequently used functions for managing objects on a Ruleflow canvas.

See [Editor commands on the Ruleflow context-sensitive menu](#) on page 75 for the Ruleflow file-level commands.

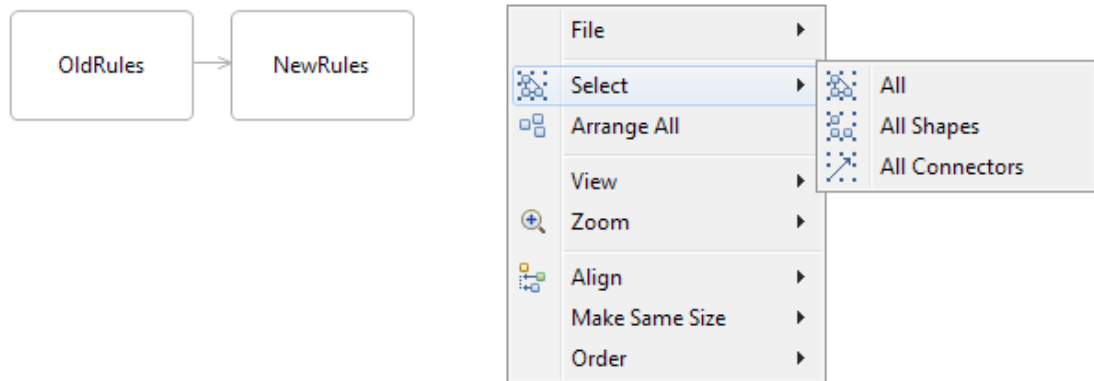


The five sections highlighted above are the object-related menu commands on the Ruleflow pop-up menu.

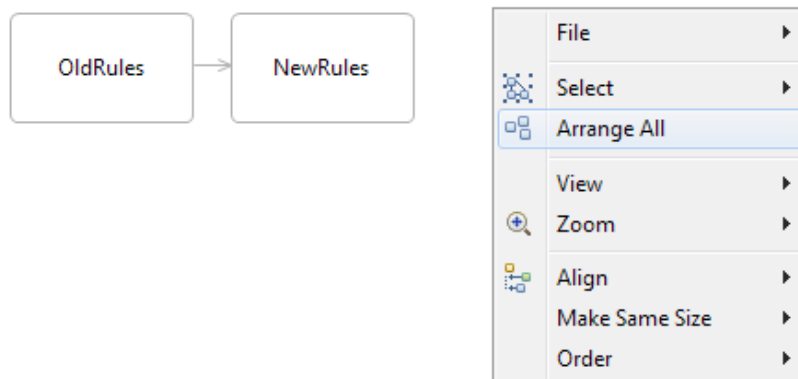
File

See [Editor commands on the Ruleflow context-sensitive menu](#) on page 75.

Select



Arrange All



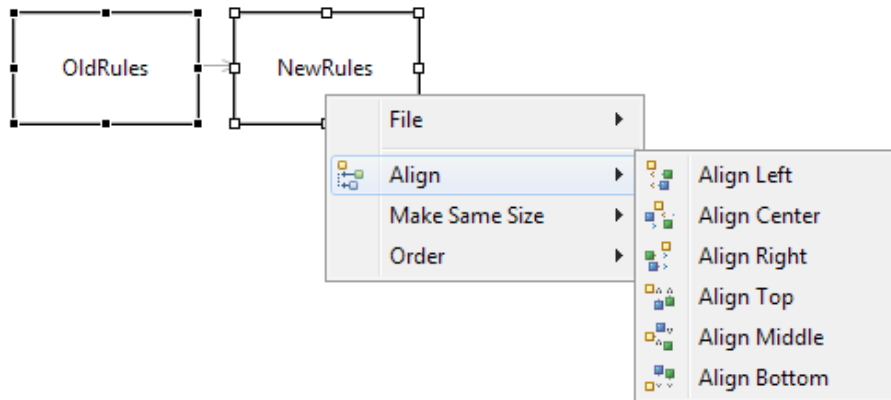
View

See [Editor commands on the Ruleflow context-sensitive menu](#) on page 75.

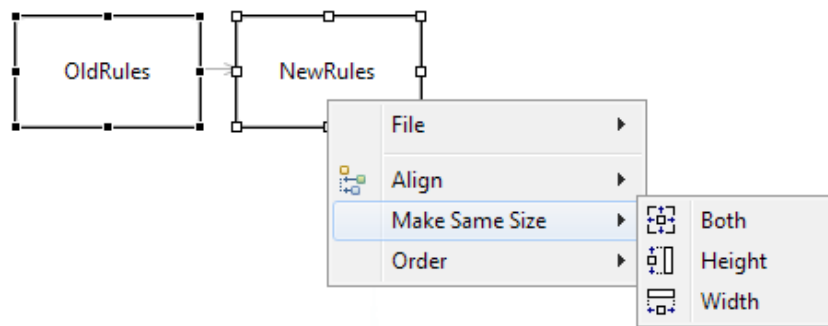
Zoom

See [Editor commands on the Ruleflow context-sensitive menu](#) on page 75.

Align

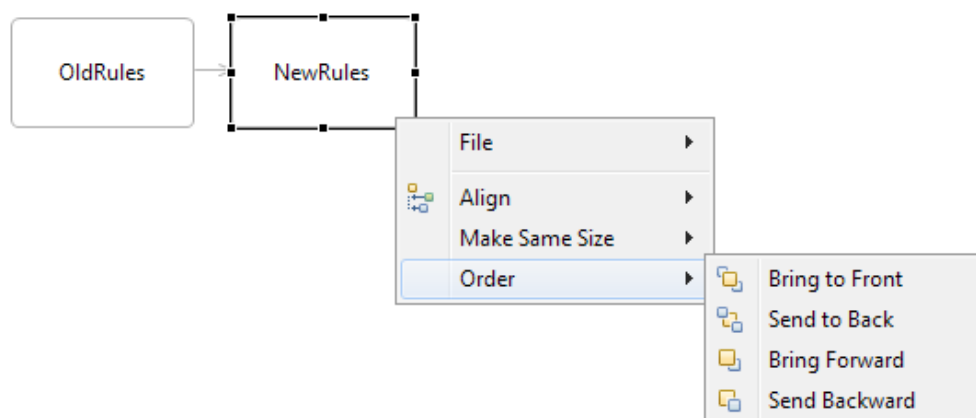


Make Same Size



- **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


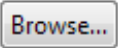


Properties of Ruleflow objects on a Ruleflow canvas

While the Ruleflow that manages the entire file and the current canvas has specific properties (see [Ruleflow properties](#) on page 79, the objects on the canvas have properties of their own in the context of the Ruleflow.

Common properties

The following properties and buttons are common to many of these objects.

- **Name** - The default name from the source file. This name can be changed to whatever will improve comprehension of its context and function on the canvas, with no impact on the source file.
-  - Color lets you choose from a color palette to override the default color that fills the object. See [Adding colors and comments to Ruleflow objects](#) on page 94 for more information. You could also use preferred general colors for all aspects of canvas objects, as discussed in [Ruleflow preferences](#) on page 82
-  - For objects that are Corticon assets, lets you navigate to a different file of the object type that uses the same Vocabulary.
- **Comments** - Provides space for adding text annotations that are reflected in reports and on the canvas when the object is in focus. See [Adding colors and comments to Ruleflow objects](#) on page 94 for more information.

Object specific properties

- **Rulesheet Activity - Rulesheet** is a valid `.ers` file in the project. You can select a different file but you cannot change the path or the name of the original file.
- **Ruleflow Activity - Ruleflow** is valid `.erf` file in the project. You can select a different file but you cannot change the path or the name of the original file.
- **Service Call-out - Service Name** pulldown menu listing service call-outs that are loaded into the workspace. The currently loaded one displays the unmodifiable **Description** specified by its author.
- **Branch Activity** - Defines the branch structure by choosing an appropriate attribute, listing values on which to branch, and then the node for each branch path. See the following topic, [Setting up a Branch on a Ruleflow canvas](#), for more information.

Iteration

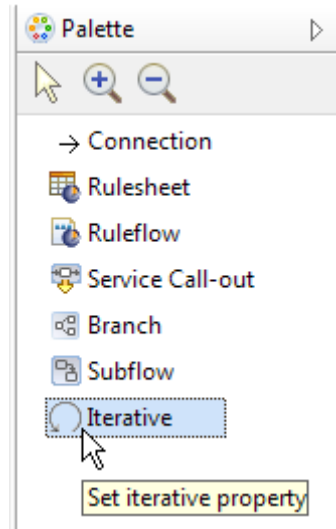
Iteration can be enabled for Rulesheet, Ruleflow, Branch and Subflow objects in a Ruleflow. When an object is set to iterate, it will repeatedly re-execute until the values derived by the object's rules cease to change. Once values in the object cease changing, the iteration stops and execution continues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

Important: Logical loop processing options are not related to these iterative functions.

Enabling and disabling iteration

To enable iteration:

1. Click on the **Iterative** icon on the palette to select it.

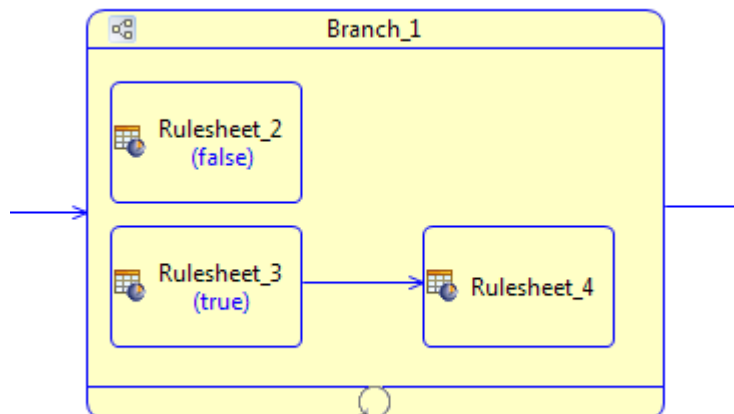


2. Click on the object on which you want to enable iterations.

- On a Rulesheet or Ruleflow object, click anywhere in the object. The iteration icon displays in the lower section of the object:



- On a Branch or Subflow object, click on the container's title bar. An iteration section is created at the bottom of the object with the iteration icon enclosed:

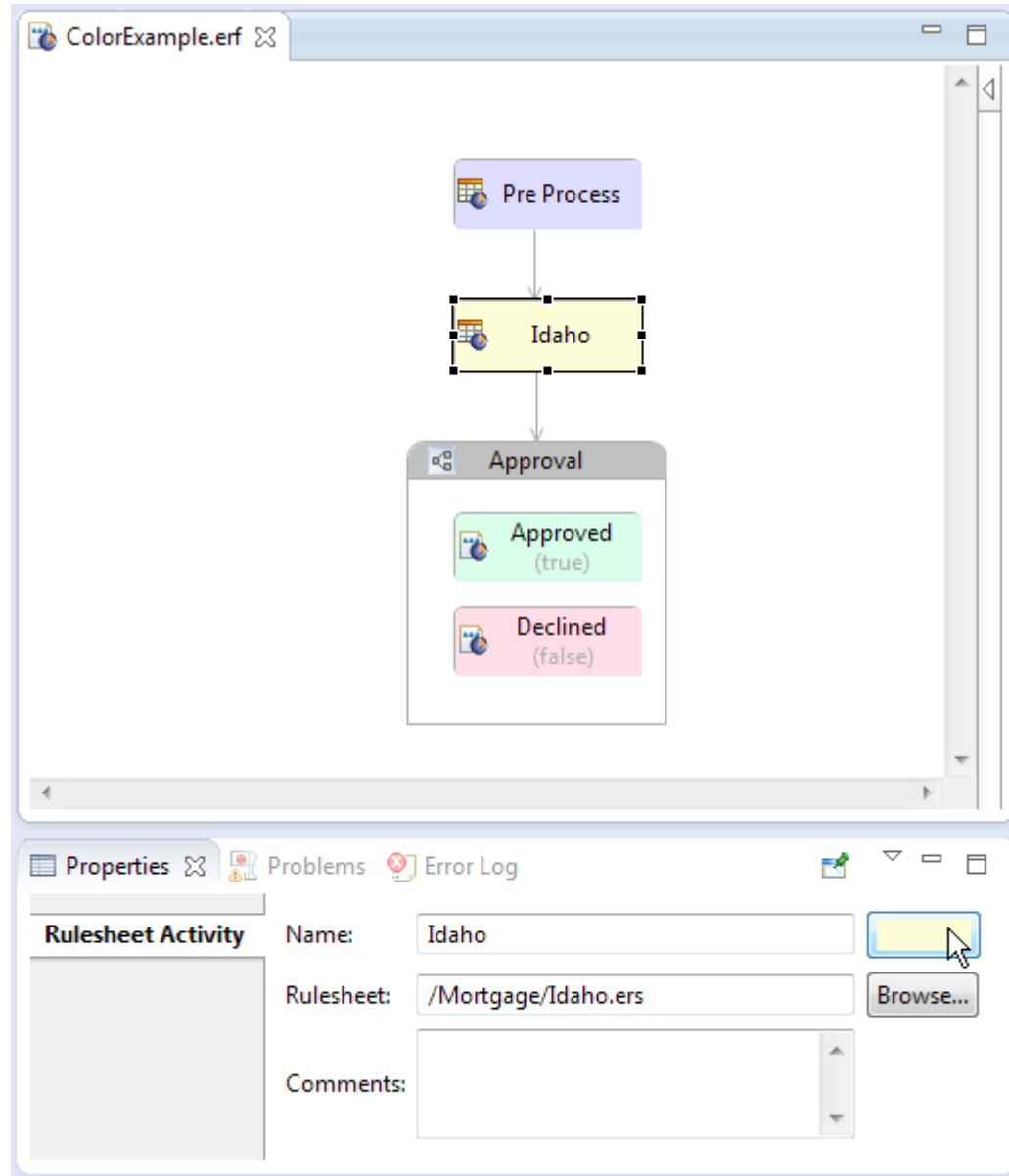


To disable iteration on an object, click on the iterative icon in the object to select it, and then press the **Delete** key.

Adding colors and comments to Ruleflow objects

Coloring Ruleflow objects

You can add color to objects on the Ruleflow canvas to enhance the look and feel of the set of objects. Each object on the canvas provides a color button to set that objects color, as illustrated:

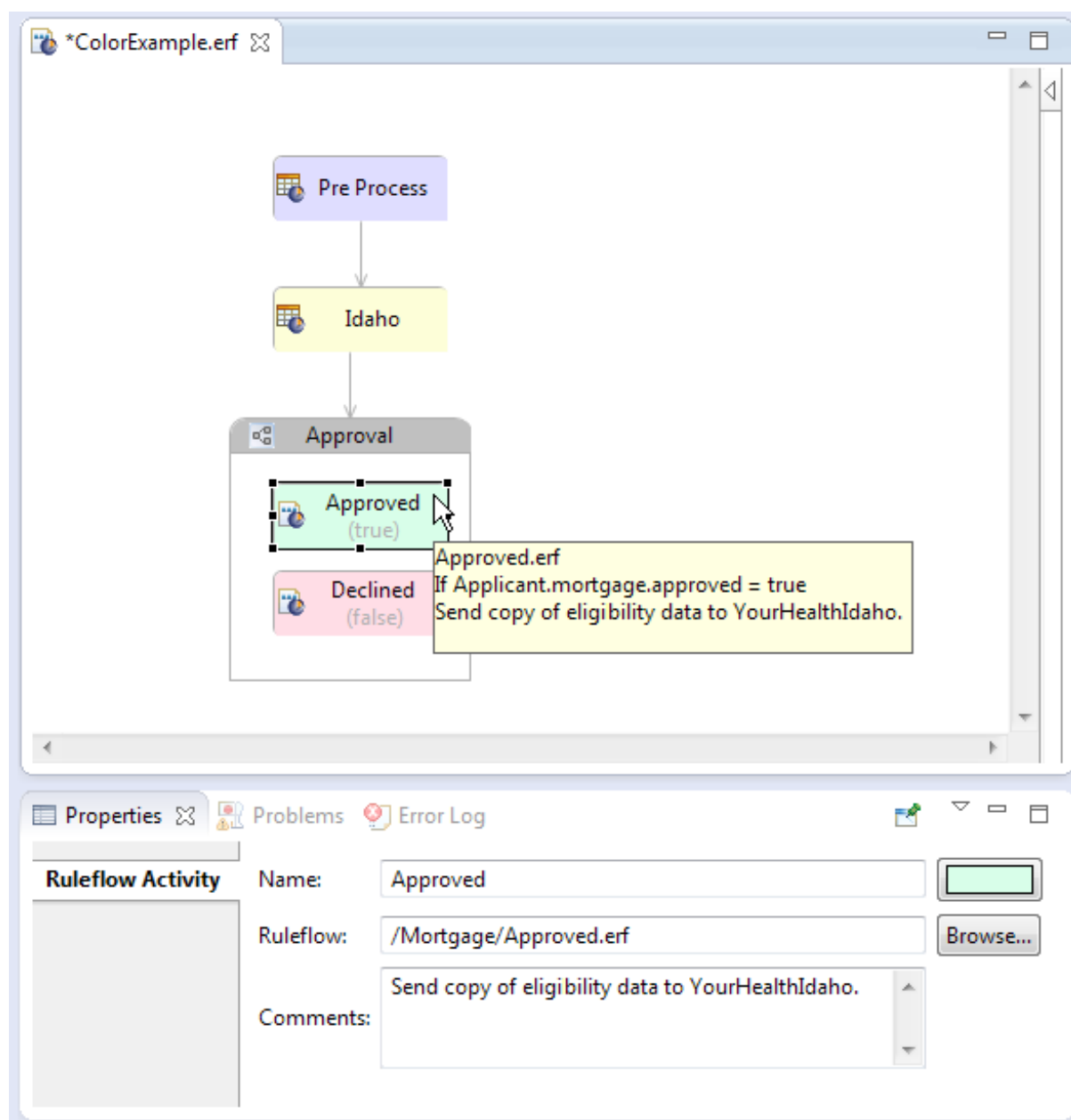


The color palette is the standard Windows basic 48-color palette that lets you choose to create custom colors.

Note: The color preference you set here on an object will override the color property set in **Window > Preferences > Viewruleflow Diagram > Appearance > Fill color**, as presented in [Ruleflow preferences](#) on page 82. When an individual object's color reverts to its default value, white, the underlying preference color is used.

Adding comments to Ruleflow objects

Each object on the Ruleflow palette can also have text that you add to its **Properties** tab, **Comments**. The following illustration shows that the text is displayed when you hover over the object.



In this example, the selected item is a Ruleflow that is the node for a branch activity. The info box first lists the asset name, then the branch attribute condition that assigned it to the node, and then the comment text.

Ruletests

A Ruletest is a Corticon asset in a project yet it is not a deployable asset. Its purpose is to define testsheets, each of which accepts data input and expected results that you want to test against any of the Rulesheets and Ruleflows in the project, or against a corresponding Decision Service deployed on a remote Java or .NET server.

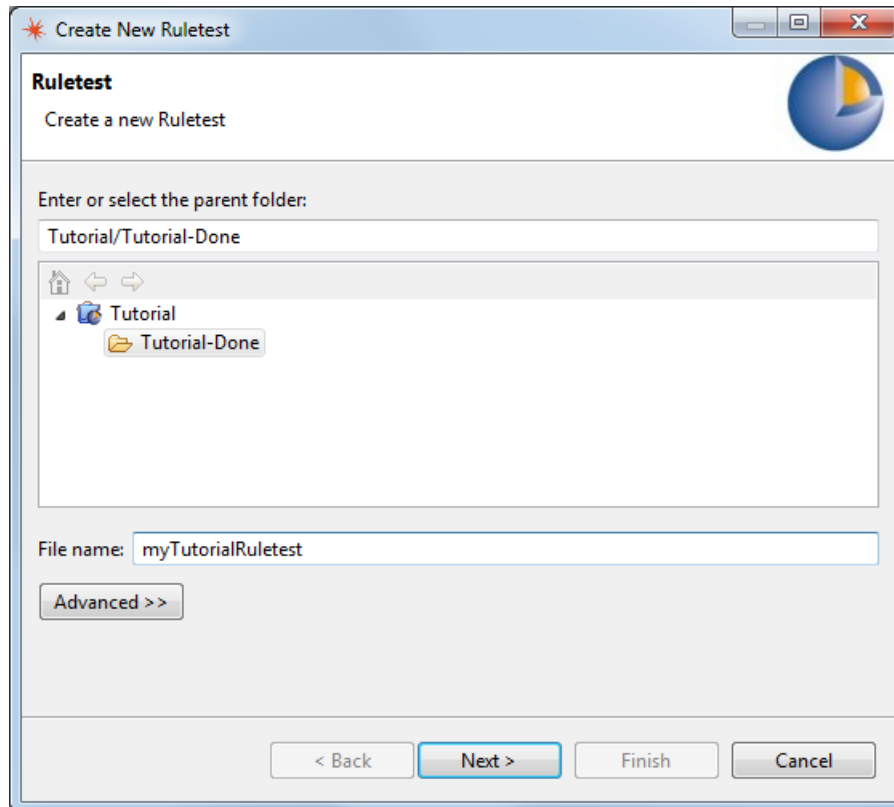
Ruletests provide techniques to import and export requests in SOAP/XML and JSON/REST formats as well to perform all database functions including caching but not including write operations.

For details, see the following topics:

- [Creating a new Ruletest](#)
- [Ruletest window](#)

Creating a new Ruletest

1. In Studio's menu, choose **File > New > Ruletest**, or click the tool button , to open the **Create New Ruletest** dialog box.



2. Select the **parent folder** for the new Ruletest, in our example Tutorial/Tutorial-Done.
3. Enter a file name for the new Ruletest, in our example myTutorialRuletest.

Note: The **Advanced** options are not relevant to Corticon and should not be used.

4. Click **Next** to continue and display the **Select Test Subject** dialog box, as shown:

Select Test Subject

☒ Execute test in Corticon Studio

- MyAdvancedTutorial
- OrderProcessing
- RetirementBenefits
- SampleDataService
- TradeAllocation
- Tutorial

☐ Execute test against a deployed Decision Service

Server URL:

Decision Services:

Name	Major	Minor	Effective Start Date	Effective Stop Date

Optional Overrides

Major Version:

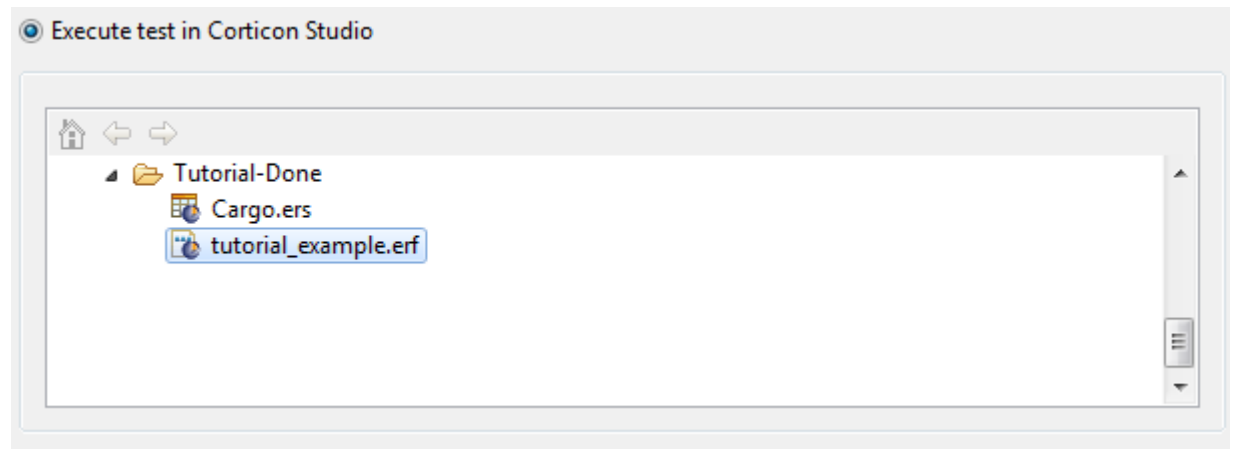
Minor Version:

Effective Target Date: / / Time: 0 0 0 AM

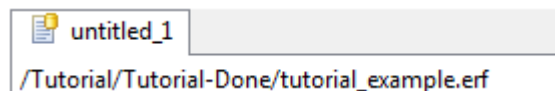
You can either execute the test in Corticon Studio, or execute the test against a deployed Decision Service, as discussed in the following topics.

Choosing a test subject in the Studio workspace

Select **Execute test in Corticon Studio**. In the view of the projects in the current workspace, only Rulesheets and Ruleflows are listed. Choose one of the files in the same project as the Ruletest to select it and enable the **Finish** button.



When you click **Finish**, the dialog closes, and the selected file is listed at the top of the testsheet, as shown:

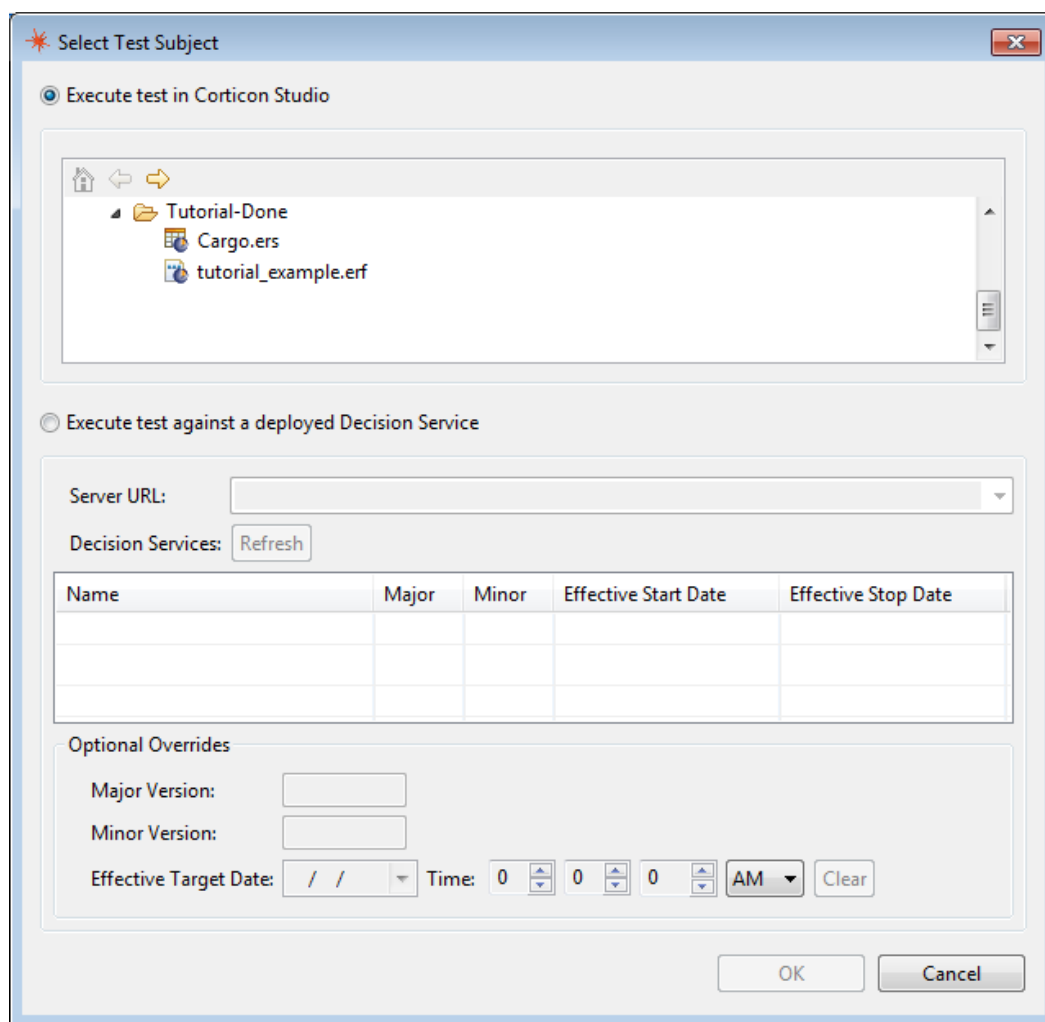


Choosing a test subject that is a deployed Decision Service

A Ruletest can specify a test subject deployed as a Decision Service on a Corticon Server that defines effective date and version properties. From the Studio point of view, such a server is often referred to as a *remote server*. By setting a target date, a test can execute as though it were sent at a specific date and time. Using this feature enables setting the clock back to see how past date ranges would have handled a request, as well as setting the clock forward to test deployed Decision Services in pre-production staging.

To use a Decision Service on a remote server as the test subject:

1. With the Ruletest you want to test open in its Studio editor, choose the menu command **Ruletest > Testsheet > Change Test Subject**. The **Select Test Subject** dialog box opens:



2. Click **Execute test against deployed Decision Service**.
3. For the **Server URL**, enter a URL; for example, a Java Server installed (and running) on the same machine as the Studio: `http://localhost:8850/axis`. Your entry is validated when you click **Refresh**, and persisted in your Studio. Once you have persisted URLs, click on the right side of the Server URL area to open the dropdown menu to make your selection.

Note: Only a few Server URLs are persisted this way. If you have a larger list that you want to edit, see [Specifying server URLs for access to test subjects](#) on page 102

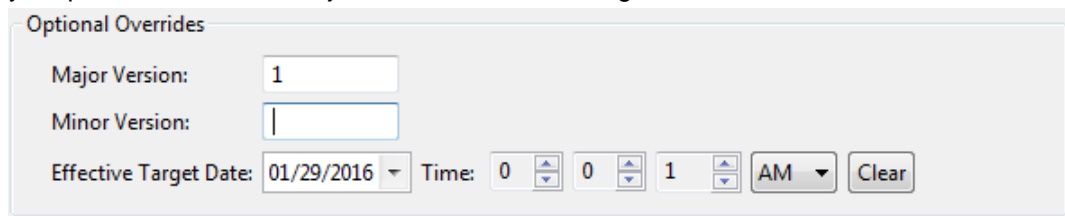
Click **Refresh** to populate the list of deployed Decision Services on that server.

4. Click on an appropriate Decision Service for this Ruletest:

Name	Major	Minor	Effective Start Date	Effective Stop Date
AllocateTrade	1	14		
Candidates	1	14		
Cargo	1	0		
ProcessOrder	1	10		

5. You might want to click **OK** at this point.

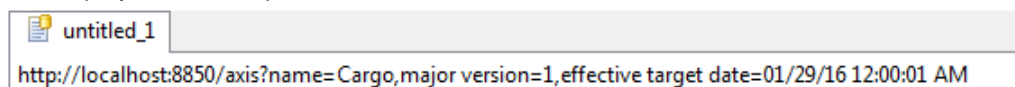
- When the selected Decision Service was deployed with a date range defined -- active from the effective date through the expiration date -- you might want to apply overrides to the test Decision Service's version or to simulate the Ruletest's call as occurring at a specific point in time. Specify your preferred values -- major version + effective target date -- as illustrated:



The dialog box titled "Optional Overrides" contains the following fields:

- Major Version:** A text input field containing the value "1".
- Minor Version:** A text input field containing a single vertical bar "|".
- Effective Target Date:** A date picker showing "01/29/2016".
- Time:** A time picker showing "00:01 AM".
- Clear:** A button to reset the overrides.

- Click **OK**. The dialog closes. The details of the remote server and Decision Service specifications are displayed at the top of the Testsheet:



The Testsheet displays the following URL in a text area:

```
http://localhost:8850/axis?name=Cargo,major version=1,effective target date=01/29/16 12:00:01 AM
```

- Run the Ruletest.

The test executes against the specified Decision Service on the selected server using the overrides you entered.

Specifying server URLs for access to test subjects

You can add a few additional server URLs through direct input, but only a few are persisted before rolling over their list.

When you have more than a few remote servers that you want to access to use their deployed Decision Services as Ruletest test subjects, you can and edit their connection information to the Studio's properties file.

To extend the test subject's Server URL list:

- In Studio's `brms.properties`, add `com.corticon.deployment.soapbindingurl_` lines with unique integer values and the server's URL. For this example, two .NET server applications, and three Java web services were defined in the following lines:

```
com.corticon.deployment.soapbindingurl_1=http://nbbedgsaintma5:8850/axis
com.corticon.deployment.soapbindingurl_2=http://nbbedgsaintma5:8850/Java_UAT
com.corticon.deployment.soapbindingurl_3=http://nbbedgsaintma1:8850/axis
com.corticon.deployment.soapbindingurl_4=http://nbbedgsaintma1:80/axis
com.corticon.deployment.soapbindingurl_5=http://nbbedgsaintma1:80/anotherNET
```

- The **Select Test Subject** dialog's **Server URL** list the various host-port-context URLs you specified:

☒ Execute test against a deployed Decision Service

Server URL:

Decision Services:

Name	URL
	http://nbbedgsaintma5:8850/axis
	http://nbbedgsaintma5:8850/java_UAT
	http://nbbedgsaintma1:8850/axis
	http://nbbedgsaintma1:80/axis
	http://nbbedgsaintma1:80/anotherNET

Optional Overrides

Major Version:

Minor Version:

Effective Target Date: / / Time: : : AM

For information about context URLs, see *"Creating custom context URLs on a web server" in the Installation Guide.*

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.

Ruletest menu commands

The following menu is available when the Ruletest editor is the active window.

The **Ruletest** menu has the following items:

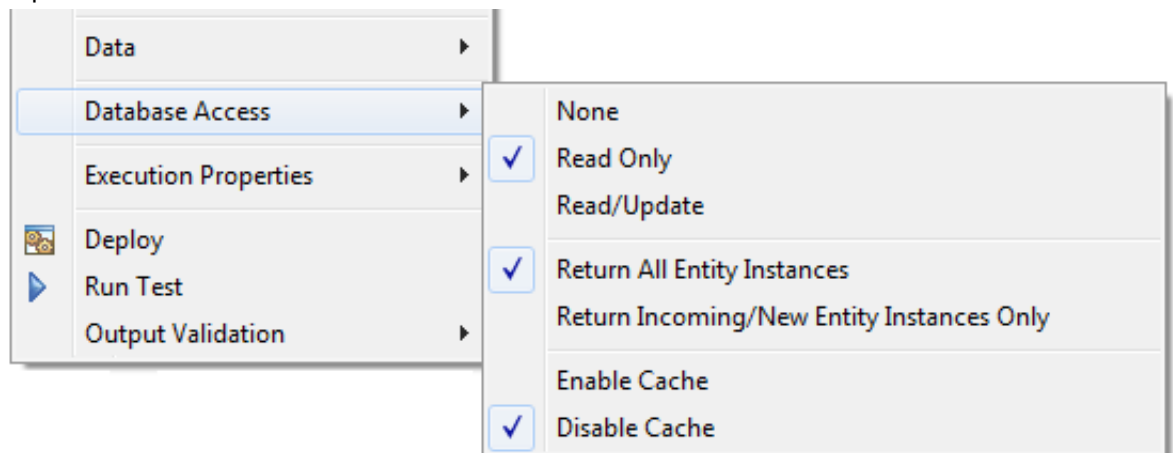
- **Testsheet**
 - **Add Testsheet** - Inserts a new Testsheet.

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

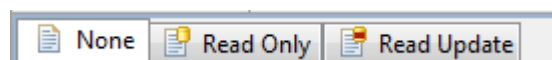
- **Copy to Expected** - Copies the data in the Output panel to the Expected panel.
- **Expected**
 - **Export Response XML** - Exports the active Testsheet's Expected pane as a `CorticonResponse` XML document.
 - **Export Response SOAP XML** - Exports the active Testsheet's Expected pane as a `CorticonResponse` XML document with SOAP envelope.
 - **Export Response JSON** - Exports the active Testsheet's Expected pane as a `CorticonResponse` JSON document.
- **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*
- **Database Access** - See the following section for details.
- **Execution Properties** - Toggles suppression of each of the message types and test output in test results. See the section below for more information.
- **Deploy** - Compiles the Ruletest target without executing it.
- **Run Test** - Compiles (if needed) and executes Ruletest.
- **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 options

The Testsheet options that are available when you are connecting to a database through the Enterprise Data Connector are:



The first option group determines, when not set to **None**, whether the Vocabulary's database connection is intended to provide **Read Only** or **Read/Update** access to the database. The chosen option decorates the testsheet's tab as stylized here where the sheet name is appropriate to its database access selection:



The second option group 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).

For more information, see the topic *"Data Synchronization" in the "Implementing EDC" section of the Integration and Deployment Guide*.

The third option group, **Enable Cache** or **Disable Cache**, determines whether the Testsheet should Enable Cache or Disable Cache when running tests. This features requires that you set Vocabulary and Rulesheets to use caching. It does not apply when Database Access is **None**.

For more information, see the topic *"Working with database caches" in the Integration and Deployment Guide*.

Execution Properties

When you choose to suppress selected level of messages in server output and logs, you might want to see that same behavior in your Ruletests first. These settings restrict each of the three types of Rule Messages (info, warning, and violation) from being posted to the output of an execution. They are test server settings that correspond to the Corticon Server properties:

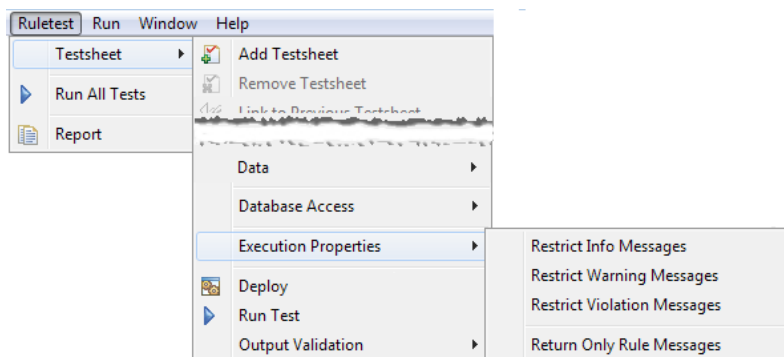
```
com.corticon.server.restrict.rulemessages.info
com.corticon.server.restrict.rulemessages.warning
com.corticon.server.restrict.rulemessages.violation
```

Also, you can choose to suppress the work document output in server output and logs, and you can test that same behavior (suppressing the **Output** pane) in your Ruletests first.

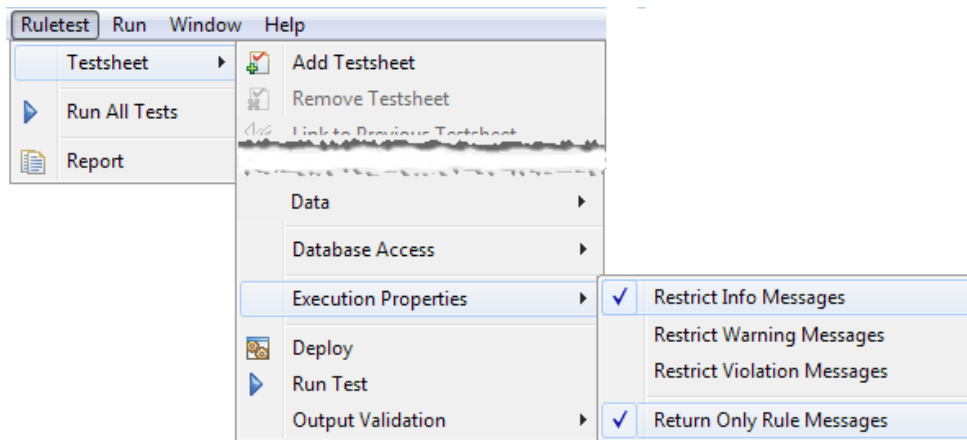
The following execution property suppresses the creation of the output pane, and displays just rule messages:

```
com.corticon.server.restrict.response.rulemessages.only
```

You can see the impact of these restrictions in testsheets by engaging each of the toggles under **Execution Properties** on the **Ruletest > Testsheet** menu as shown:



When an option is checked, messages of that type are suppressed:



Clicking an option again clears it, so that the message type or output is again produced.

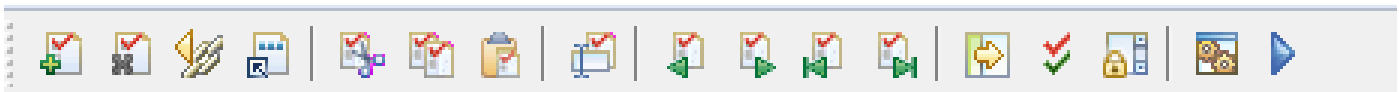
Exporting messages from a Testsheet adds in any execution properties you set. For example, in SOAP/XML:

```
<ExecutionProperties>
  <ExecutionProperty
    value="true"
    name="PROPERTY_EXECUTION_RESTRICT_RULEMESSAGES_INFO"/>
  <ExecutionProperty
    value="true"
    name="PROPERTY_EXECUTION_RESTRICT_RESPONSE_TO_RULEMESSAGES_ONLY"/>
</ExecutionProperties>
```





Importing that message will toggle -- in this example -- the corresponding Testsheet options, **Restrict Info Messages** and **Return Only Rule Messages**.














Ruletest toolbar

When the Ruletest editor is active, its tools are added to the toolbar, as shown:



The Ruletest tools provide the same functions as the corresponding **Ruletest > Testsheet** menu commands:

-  **Add Testsheet**
-  **Remove Testsheet**
-  **Link to Previous Testsheet**
-  **Change Test Subject**

-  **Cut Testsheet**
-  **Copy Testsheet**
-  **Paste Testsheet**
-  **Rename Testsheet**
-  **Move Backward**
-  **Move Forward**
-  **Move to Beginning**
-  **Move to End**
-  **Copy Output to Expected**
-  **Validate**
-  **Scroll Lock**
-  **Deploy**
-  **Run Test**

Commands on a Testsheet context-sensitive menu

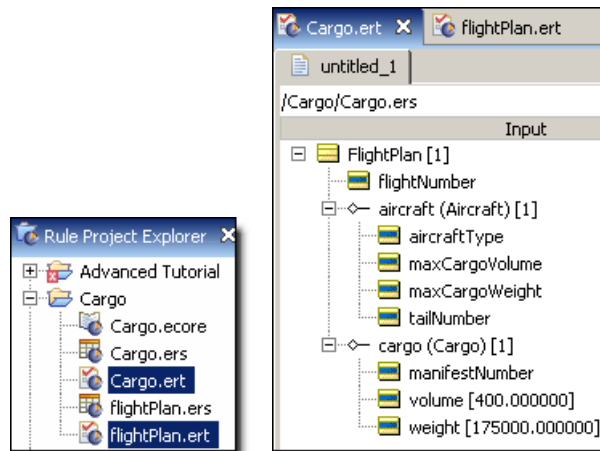
When you right-click in a Testsheet, the Ruletest context-sensitive menu opens, displaying the following commands and toggles as appropriate to certain columns and data types:

Command	Columns	Comments
Cut	Input, Output, or Expected	Any line(s) in a column. Selects the branches under each selected level as well.
Copy	Input, Output, or Expected	Any line(s) in a column. Selects the branches under each selected level as well.
Paste	Input or Expected	Valid target location in column.
Delete	Input, Output, or Expected	Any line in a column. Selects the branches under each selected level as well.
Set to Null	Input or Expected	Applies only to attributes. Use Ctrl+click or Shift+click to choose several attributes.
Ignore/Enable Validation	Expected	Applies only to attributes. Use Ctrl+click or Shift+click to choose several attributes.
Go to Entity	Input or Expected	Selected association.
Sort Entities	Input or Expected	Anywhere in selected column sorts to the entire column.

Command	Columns	Comments
Scroll Lock	Anywhere on sheet	Synchronizes scrolling of all three columns.
Properties	Anywhere on sheet	<p>Opens the Ruletest's Properties panel which provides the following tabs:</p> <ul style="list-style-type: none"> • Ruletest - The Vocabulary file associated with this Ruletest • Testsheet - The Rulesheet or Ruleflow file associated with the Ruletest. • Comments - Each line and each column can enter unique text information that is then displayed adjacent to that item, enclosed in braces. • Datastore ID - When the selected item is an association, allows a datastore identifier to be entered. This is important when using the Enterprise Data Connector. For more information, see the topic <i>"Identity Strategies" in the Relational database concepts chapter of the Integration and Deployment Guide.</i>

Testsheet tabs

A Ruletest contains **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.

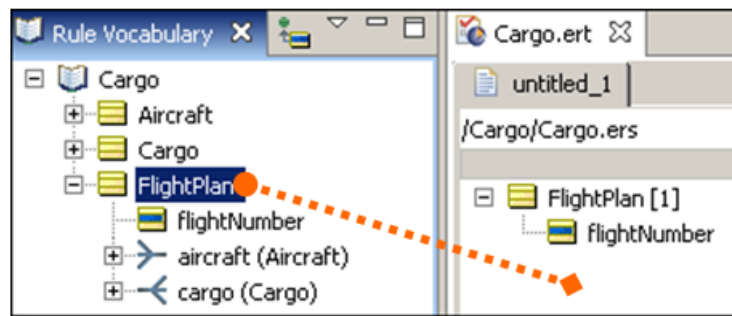


Populating the input panel

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.



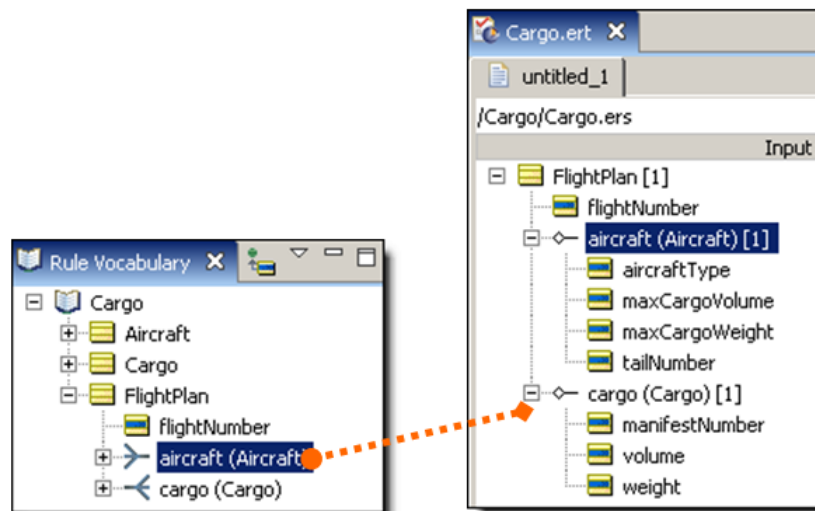
Important: Entities may be deleted by pressing the **Delete** key or by selecting **Delete** from the right-click pop-up menu.

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.

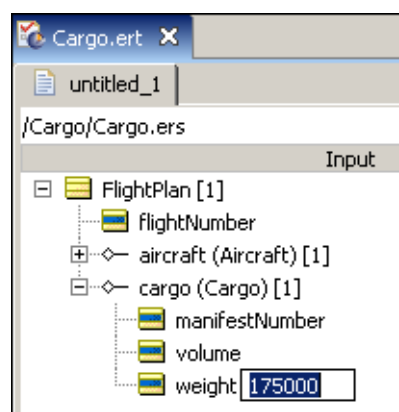


2. Verify that the child entity has the appropriate "indented" structure underneath the parent entity, indicating the association has been created correctly.

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-click on the attribute where you want to enter data. Its data entry box opens.
2. Type the test value for that attribute:



3. Either:
 - Press **Enter** to record the entry
 - Press **Tab** to record that entry, and then advance to the entry box of the next attribute.

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

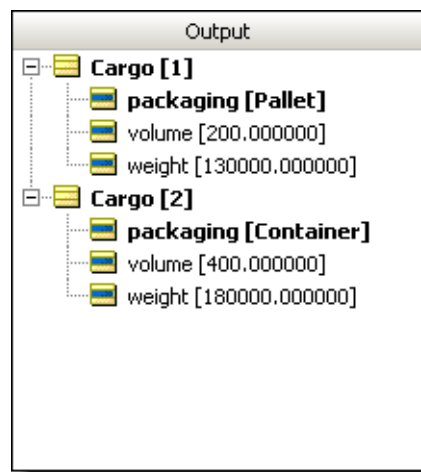
Before a Ruletest can be executed, the corresponding Rulesheets (whether being tested directly or as contained in Ruleflows) must be compiled. This compilation process does not occur until the Ruletest is run. Depending on how many Rulesheets and rules must be compiled, this compilation step may take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

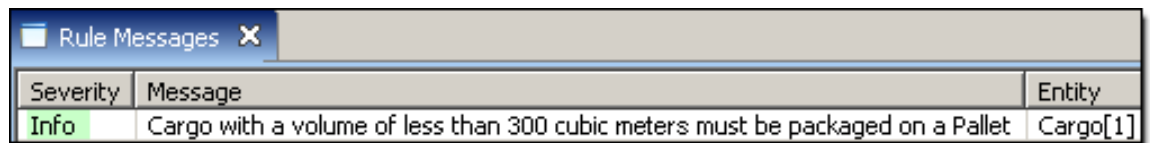
Once a Ruletest has been executed, it will rerun without recompiling as long as the rules have not changed since the last execution. Changing the Input test data does not require recompilation.

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

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.



- 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.



Severity	Message	Entity
Info	Cargo with a volume of less than 300 cubic meters must be packaged on a Pallet	Cargo[1]

- 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.
- 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.

/Tutorial/Rules/Untitled.ers

Input	Output	Expected
<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging volume [200] weight [130000] Cargo [2] <ul style="list-style-type: none"> packaging volume [400] weight [180000] 	<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging [Pallet] volume [200.000000] weight [130000.000000] Cargo [2] <ul style="list-style-type: none"> packaging [Container] volume [400.000000] weight [180000.000000] 	<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging [Pallet] volume [200.000000] weight [130000.000000] Cargo [2] <ul style="list-style-type: none"> packaging [Container] volume [400.000000] weight [180000.000000]

Rule Statements Rule Messages

Severity	Message	Entity
Info	Cargo weighing between 150000 and 200000 kilograms must be packaged in a container	Cargo[2]
Info	Cargo with a total volume less than 300 cubic meters must be packaged on a pallet	Cargo[1]

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.

/Tutorial/Rules/Untitled.ers

Input	Output	Expected
<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging volume [200] weight [130000] Cargo [2] <ul style="list-style-type: none"> packaging volume [400] weight [180000] 	<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging [Pallet] volume [200.000000] weight [130000.000000] Cargo [2] <ul style="list-style-type: none"> packaging [Container] volume [400.000000] weight [180000.000000] 	<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging [Container] volume [200.000000] weight [130000.000000] Cargo [2] <ul style="list-style-type: none"> packaging [Container] volume [400.000000] weight [180000.000000]

Rule Statements Rule Messages

Severity	Message	Entity
Info	Cargo weighing between 150000 and 200000 kilograms must be packaged in a container	Cargo[2]
Info	Cargo with a total volume less than 300 cubic meters must be packaged on a pallet	Cargo[1]

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.

/Tutorial/Rules/Untitled.ers

Input	Output	Expected
<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging volume [200] weight [130000] Cargo [2] <ul style="list-style-type: none"> packaging volume [400] weight [180000] 	<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging [Pallet] volume [200.000000] weight [130000.000000] 	<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging [Pallet] volume [200.000000] weight [130000.000000] Cargo [2] <ul style="list-style-type: none"> packaging [Container] volume [400.000000] weight [180000.000000]

Rule Statements Rule Messages

Severity	Message	Entity
Info	Cargo weighing between 150000 and 200000 kilograms must be packaged in a container	Cargo[2]
Info	Cargo with a total volume less than 300 cubic meters 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.

/Tutorial/Rules/Untitled.ers

Input	Output	Expected
<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging volume [200] weight [130000] Cargo [2] <ul style="list-style-type: none"> packaging volume [400] weight [180000] 	<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging [Pallet] volume [200.000000] weight [130000.000000] Cargo [2] <ul style="list-style-type: none"> packaging [Container] volume [400.000000] weight [180000.000000] 	<ul style="list-style-type: none"> Cargo [1] <ul style="list-style-type: none"> packaging [Pallet] volume [200.000000] weight [130000.000000]

Rule Statements Rule Messages

Severity	Message	Entity
Info	Cargo weighing between 150000 and 200000 kilograms must be packaged in a container	Cargo[2]
Info	Cargo with a total volume less than 300 cubic meters 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) might look.

/Tutorial/Rules/Untitled.ers

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

Expected panel: Setting selected attributes to ignore validation.

When different values are output than what was expected, it could mean that the **Expected** column data created from **Output** data were reflecting dynamic values such as dates and time. If your Rulesheets use `now` or `today`, the **Expected** values will evaluate as errors very soon. To handle that situation, you can choose to ignore validation for selected values in the **Expected** column.

Consider the following example:

The selected attribute in this test has no input value and no expected value

Input	Output	Expected
<div><div>Customer [1]</div><div>age [57]</div><div>smoker [true]</div><div>policy (Policy) [1]</div><div>category [Life]</div><div>coverage</div><div>effective_date</div><div>id</div><div>newlyCreated [true]</div><div>premium [0]</div><div>type [Standard]</div></div>		<div><div>Customer [1]</div><div>age [57]</div><div>smoker [true]</div><div>policy (Policy) [1]</div><div>category [Life]</div><div>coverage</div><div>effective_date</div><div>id</div><div>newlyCreated [true]</div><div>premium [2220.000000]</div><div>type [Standard]</div></div>

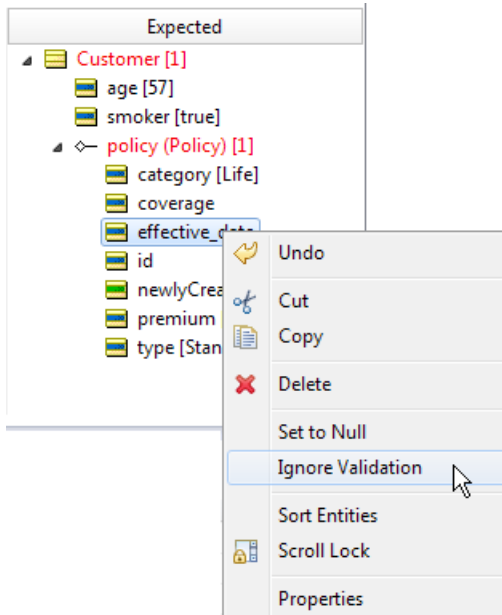
When the test runs, it is valid.

Input	Output	Expected
<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date id newlyCreated [true] premium [0] type [Standard] 	<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date id newlyCreated [true] premium [2220.000000] type [Standard] 	<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date id newlyCreated [true] premium [2220.000000] type [Standard]

But when the input gets a value and the output still has no value (or a different value), the test fails.

Input	Output	Expected
<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date [7/4/2014] id newlyCreated [true] premium [0] type [Standard] 	<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date [7/4/2014] id newlyCreated [true] premium [2220.000000] type [Standard] 	<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date id newlyCreated [true] premium [2220.000000] type [Standard]

Clicking on the expected attribute, you can choose **Ignore Validation**.



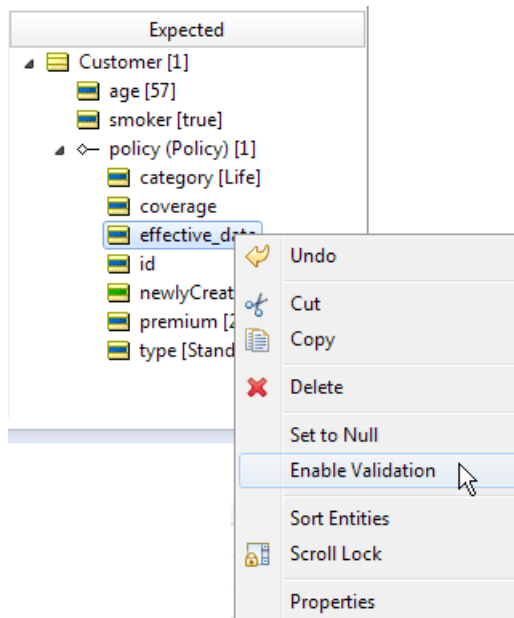
An attribute that will be ignored is greyed out.

Input	Output	Expected
<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date [7/4/2014] id newlyCreated [true] premium [0] type [Standard] 		<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date id newlyCreated [true] premium [2220.000000] type [Standard]

Running the same test, the test passes.

Input	Output	Expected
<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date [7/4/2014] id newlyCreated [true] premium [0] type [Standard] 	<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date [7/4/2014] id newlyCreated [true] premium [2220.000000] type [Standard] 	<ul style="list-style-type: none"> Customer [1] <ul style="list-style-type: none"> age [57] smoker [true] policy (Policy) [1] <ul style="list-style-type: none"> category [Life] coverage effective_date id newlyCreated [true] premium [2220.000000] type [Standard]

The setting can revert by selecting the attribute and then choosing **Enable Validation**.



Format of output data

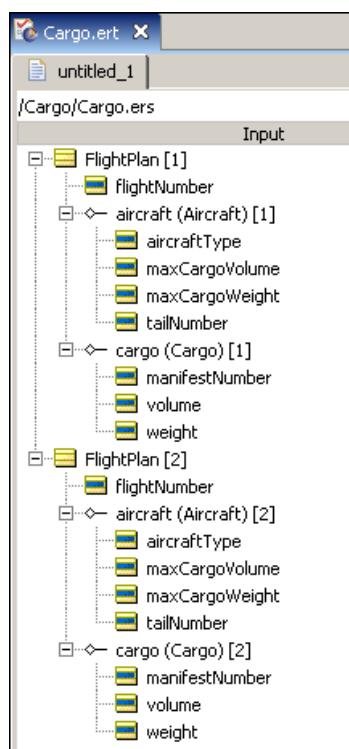
By default, the data in the Output pane will be displayed in the same format as the Input. In other words, if the Input data is flat, then the so will be the Output. If the Input is hierarchical, then so will be the Output.

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`.

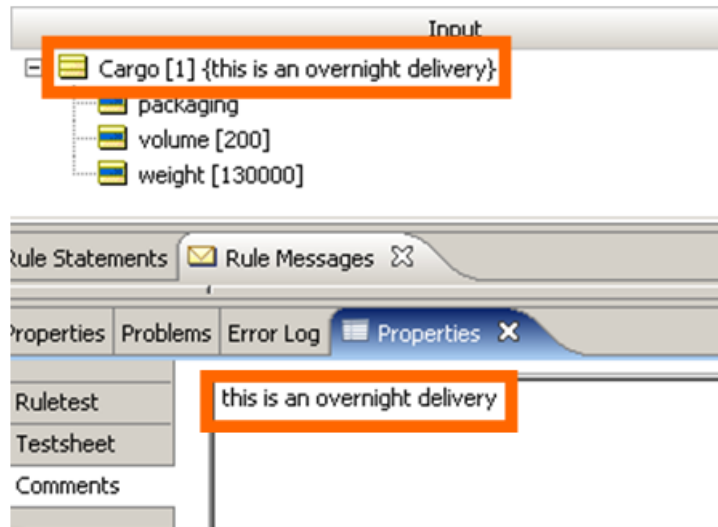
For more information about flat and hierarchical data formats, including examples, see the Integrating Decision Services chapter of the *Server Integration & Deployment Guide*.

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 and paste elements already in the Input panel.



2. New elements are automatically numbered to ensure a unique ID for each.

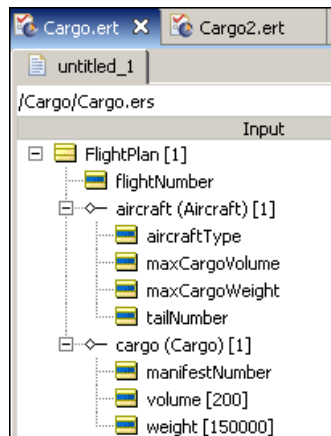


3. Additional descriptive text may be added to any node by selecting **Properties** from its right-click pop-up menu, and typing text in the **Comments** tab of the **Properties** window.

Creating multiple test scenarios as a set of testsheets

You might find it more convenient to organize multiple scenarios into a set of Testsheets, each scenario with its own **Input**, **Output**, and **Expected** panels.

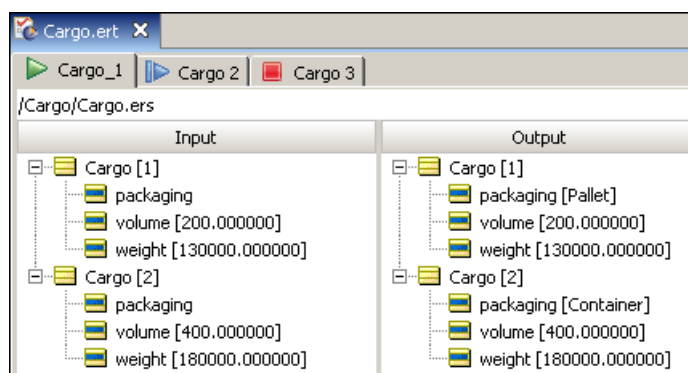
1. Each **Input** panel must be linked to a Rulesheet or Ruleflow.
2. Each **Input** panel will generate its own **Output** panel. Each of these Testsheet tabs can be **renamed** to help you distinguish them.






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.

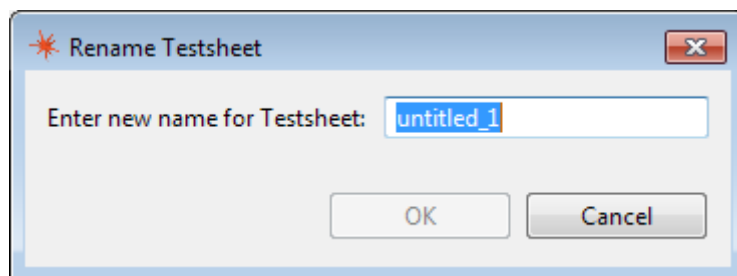


2. Assume we want to test three Ruleflows (named `Cargo`, `Cargo2` and `Cargo3`) in sequence.
3. Create a new Ruletest as before and link it to Ruleflow `cargo.ert` as usual.
4. Execute the Ruletest to generate data in the Output panel.
5. Add a new Testsheet using  and link it to Ruleflow `cargo2.ert`.
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.

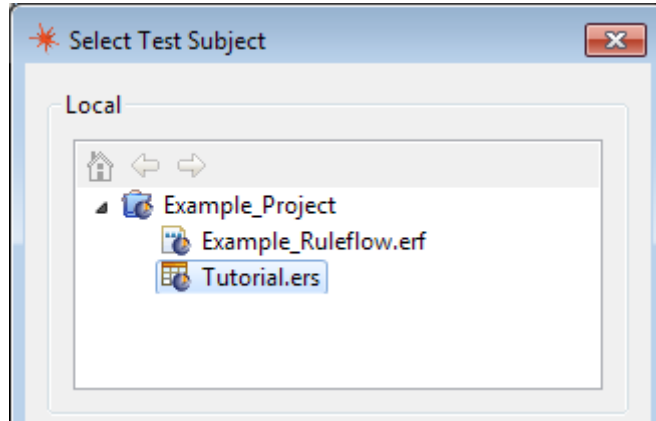
1. You can either select **Ruletest > Testsheet > Rename Testsheet** from the menubar or the right-click pop-up menu, or select the **Rename Testsheet** icon from the Ruletest toolbar to display the **Rename Test Tab** dialog box.



2. Type the new name in the space provided and press or click **Enter**.
3. Like Rulesheets and Vocabularies, Ruletest and Testsheet names must comply with the guidelines described in [File naming restrictions](#) on page 20.

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.



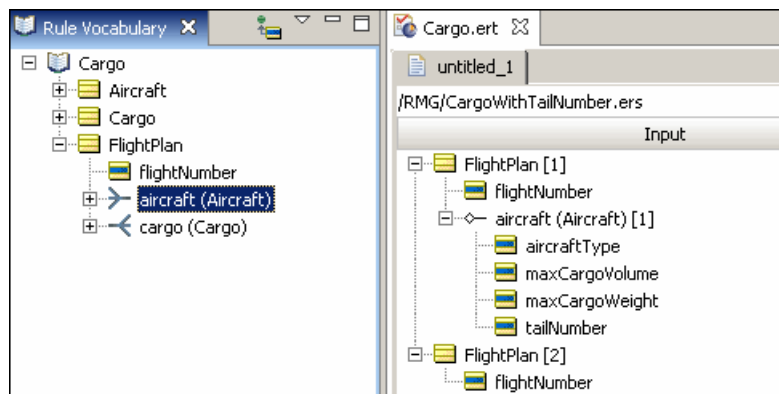
2. Select the Rulesheet to associate with the new Ruletest and click **OK**.

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.

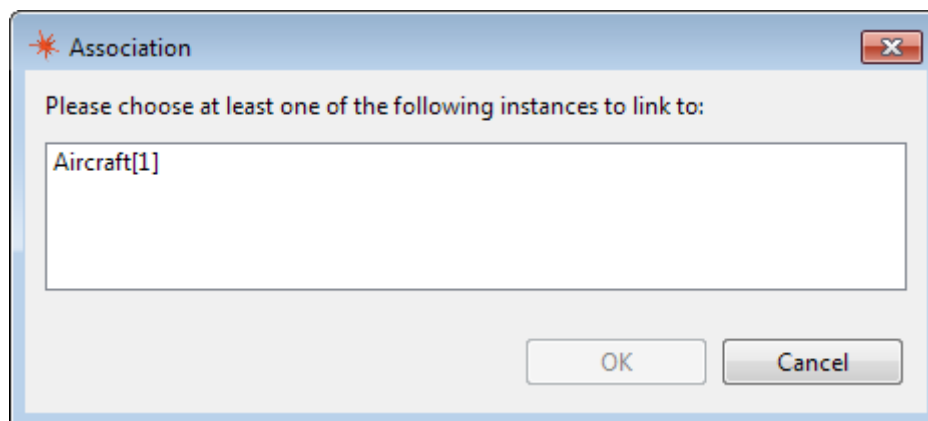
Associating one child entity with more than one parent

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

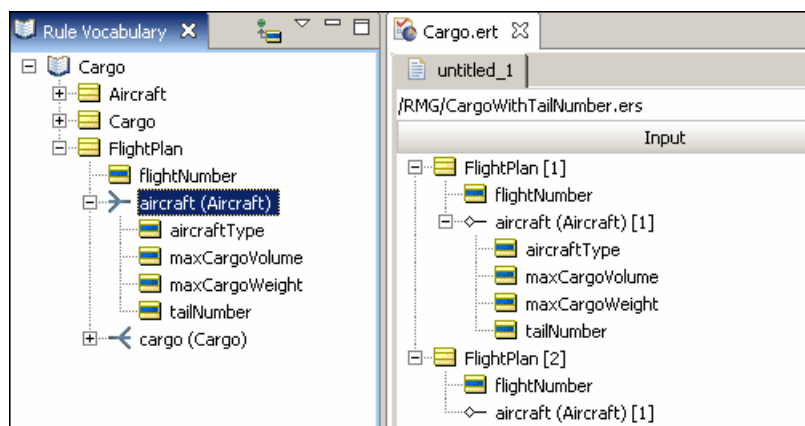
1. First, ensure the parent-child (source-target) association supports a multiplicity of many-to-many or many-to-one.
2. Drag and drop the parent entities (here, *FlightPlan*) as usual.
3. Create an association between aircraft and *FlightPlan[1]* as described in **Creating Associations**.
4. To associate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

You can save a new Ruletest as soon as you enter some data into it. To save the Ruletest with a different name, choose the menu command **File > Save As**.

Importing an XML or SOAP document to a testsheet

You can import XML or SOAP data into the Tester rather than recreate it by using the **Ruletest** menu command **Testsheet > Import > XML/SOAP**.

Note: This is the default setting. If you changed the Studio execution property that controls the import format from its default value, `XML`, to `JSON`, you now need to either change it back to `com.corticon.testster.ccserver.execute.format=XML` (or delete the line) in the Studio's `brms.properties` file, and then restart the Studio.

Corticon Studio can provide a sample XML or SOAP document from a populated testsheet that you can use as a template. Choose the menu command **Testsheet > Data > Input > Export Request XML** or **Testsheet > Data > Input > Export Request SOAP**.

Importing a JSON document to a testsheet

You can import a well-formed JSON-formatted CorticonRequest into the Tester with the **Ruletest** menu command **Testsheet > Import > JSON**.

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).

Input

To generate the Input column of a testsheet to well-formed XML, choose the menu command **Testsheet > Data > Input > Export Request 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:Corticon"
2  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2  decisionServiceName="InsertDecisionServiceName">
3  <WorkDocuments>
4  <Cargo id="Cargo_id_1">
5      <weight>1000</weight>
6      <volume>10</volume>
7      <container xsi:nil="true" />
8  </Cargo>
9  <Cargo id="Cargo_id_4">
10     <weight>1000</weight>
11     <volume>10</volume>
12     <container xsi:nil="true" />
13     <needsRefrigeration>true</needsRefrigeration>
14 </Cargo>
15 </WorkDocuments>
16 </CorticonRequest>

```

Output

To generate the Output column of a testsheet to well-formed XML, choose the menu command **Testsheet > Data > Output > Export Response XML**.

Expected

To generate the Expected column of a testsheet to well-formed XML, choose the menu command **Testsheet > Data > Expected > Export Response XML**.

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 `CorticonRequest` tag, shown in the following image.

Input

To generate the Input column of a testsheet to SOAP, choose the menu command **Testsheet > Data > Input > Export Request SOAP**. The output from a simple `Cargo` test is as shown:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
2  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:Corticon" xmlns:xsi="http://www.w3.org/2001/XMLSchema"
4  decisionServiceName="InsertDecisionServiceName">
5  <WorkDocuments>
6  <Cargo id="Cargo_id_1">
7  <weight>1000</weight>
8  <volume>10</volume>
9  <container xsi:nil="true" />
10 </Cargo>
11 <Cargo id="Cargo_id_4">
12 <weight>1000</weight>
13 <volume>10</volume>
14 <container xsi:nil="true" />
15 <needsRefrigeration>true</needsRefrigeration>
16 </Cargo>
17 </WorkDocuments>
18 </CorticonRequest>
19 </SOAP-ENV:Body>
20 </SOAP-ENV:Envelope>

```

Output

To generate the Output column of a testsheet to well-formed SOAP, choose the menu command **Testsheet > Data > Output > Export Response SOAP**.

Expected

To generate the Expected column of a testsheet to well-formed SOAP, choose the menu command **Testsheet > Data > Expected > Export Response SOAP**.

Exporting a testsheet to a JSON document

You can export a testsheet into well-formed JSON documents. Exporting Testsheets into JSON documents is quite useful:

- As a template for structuring much larger data sets for subsequent JSON Import (as described above)
- As a JSON payload to test a deployed Decision Service on the Server

Input

To generate the Input column of a testsheet to JSON, choose the menu command **Testsheet > Data > Input > Export Request JSON**. The output from a simple `Cargo` test is as shown:

```
1 [{"Objects": [  
2   {  
3     "weight": "1000",  
4     "container": null,  
5     "volume": "10",  
6     "__metadata": {  
7       "#id": "Cargo_id_1",  
8       "#type": "Cargo",  
9     }  
10  },  
11  {  
12    "weight": "1000",  
13    "needsRefrigeration": "true",  
14    "container": null,  
15    "volume": "10",  
16    "__metadata": {  
17      "#id": "Cargo_id_4",  
18      "#type": "Cargo",  
19    }  
20  }  
21 ]}]
```

Output

To generate the Output column of a testsheet to well-formed JSON, choose the menu command **Testsheet > Data > Output > Export Response JSON**.

Expected

To generate the Expected column of a testsheet to well-formed JSON, choose the menu command **Testsheet > Data > Expected > Export Response JSON**.

Creating a Ruletest report

1. Select the menu command **Ruletest > Report**.
2. The Ruletest report should open as a new HTML page in your default web browser.
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.

Access to Corticon knowledge resources

[Complete online documentation for the current release](#)

Corticon online tutorials:

- [Tutorial: Basic Rule Modeling in Corticon Studio](#)
- [Tutorial: Advanced Rule Modeling in Corticon Studio](#)
- [Modeling Progress Corticon Rules to Access a Database using EDC](#)
- [Connecting a Progress Corticon Decision Service to a Database using EDC](#)

Corticon guides (PDF):

- [What's New in Corticon](#)
- [Corticon Installation Guide](#)
- [Corticon Studio: Rule Modeling Guide](#)
- [Corticon Studio: Quick Reference Guide](#)
- [Corticon Studio: Rule Language Guide](#)
- [Corticon Studio: Extensions Guide](#)
- [Corticon Server: Integration and Deployment Guide](#)
- [Corticon Server: Web Console Guide](#)
- [Corticon Server: Deploying Web Services with Java](#)
- [Corticon Server: Deploying Web Services with .NET](#)

Corticon JavaDoc API reference (HTML):

- [Corticon Foundation API](#)
- [Corticon Model API](#)
- [Corticon Server API](#)

See also:

- [Introducing the Progress® Application Server](#)
- Corticon documentation for this release on the [Progress download site](#): What's New Guide (PDF), Installation Guide (PDF), PDF download package, and the online Eclipse help installed with Corticon Studio.