



Corticon.js

Quick Reference

Copyright

© 2021 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

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

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient 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.

Last updated with new content: Corticon.js 1.3

Updated: 2021/11/08

Table of Contents

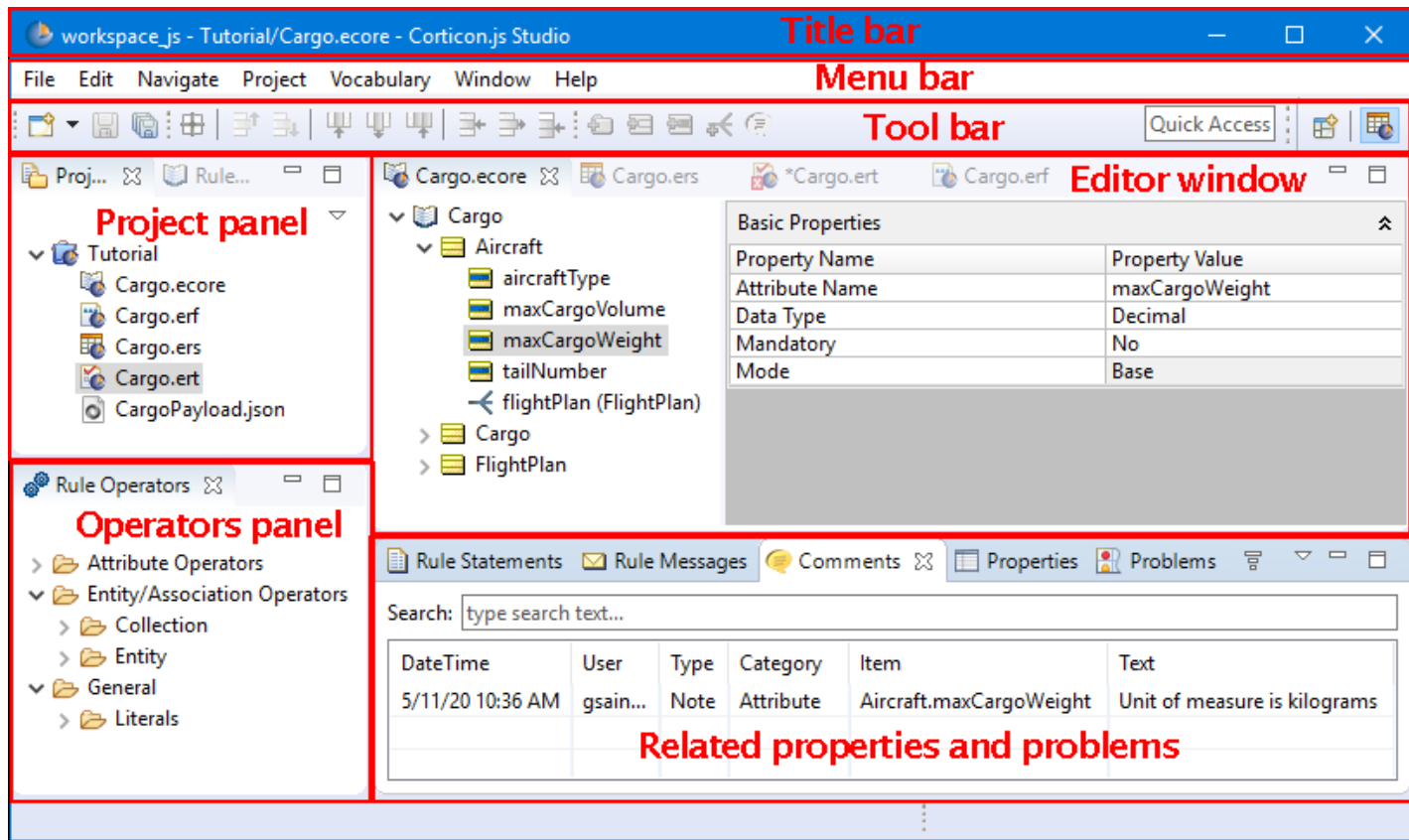
A guide to Progress Corticon.js Studio	9
The active Corticon Studio window.....	14
Alternate settings for Corticon Studio properties.....	15
File naming restrictions.....	16
How to rename and relocate assets in the Project Explorer.....	17
How to use Git to manage project assets.....	18
 Rule Projects.....	 19
How to import or export a Rule Project.....	19
How to create a Rule Project.....	20
The Project Explorer window.....	21
How to package rules for deployment	23
 Vocabularies.....	 27
How to create a Vocabulary.....	27
The Vocabulary window.....	28
How to define a Vocabulary.....	30
Custom data types tab.....	30
Add nodes to the Vocabulary tree view.....	31
How to save a new Vocabulary.....	37
How to open an existing Vocabulary.....	38
How to modify a Vocabulary.....	38
How to create a Vocabulary report.....	39
How to find references to an entity, attribute, or association in a project.....	39
How to refactor entity attribute or association names in a project.....	41
 Rulesheets.....	 45
How to create a Rulesheet.....	46
Rulesheet menu commands.....	47
Rulesheet toolbar.....	49
Commands on the Rulesheet context-sensitive menu.....	50
Rulesheet sections.....	51
Navigation in a Rulesheet.....	54
Rule statements window.....	54
Rulesheet properties.....	56
Use the business vocabulary to build rules.....	56
Use Rule Operators to build rules.....	57

How to name Rulesheets.....	57
How to validate and optimize a Rulesheet.....	57
How to create a Rulesheet report.....	58
How to save a new Rulesheet.....	58
How to save a modified Rulesheet.....	59
How to close a Rulesheet.....	59
How to delete Rulesheets.....	59
Ruleflows.....	61
How to create a Ruleflow.....	61
How to name Ruleflows.....	63
How to rename a Ruleflow or save a Ruleflow to a different location.....	63
How to create a Ruleflow report.....	64
How to save a new Ruleflow.....	65
How to close a Ruleflow.....	65
How to delete Ruleflows.....	65
Ruleflow window.....	65
Ruleflow menu commands.....	65
Ruleflow toolbar.....	66
Editor commands on the Ruleflow context-sensitive menu.....	66
Ruleflow properties.....	69
Ruleflow Activity.....	70
Rulers and grid.....	71
Settings for Ruleflow graph fonts.....	71
Ruleflow preferences.....	71
Objects on a Ruleflow canvas.....	75
Ruleflow canvas tools.....	76
Object commands on the Ruleflow context-sensitive menu.....	76
Properties of Ruleflow objects on a Ruleflow canvas.....	80
How to add colors to Ruleflow objects.....	81
Ruletests.....	83
How to create a Ruletest.....	83
Choose a test subject in the Studio workspace.....	86
Ruletest window.....	86
Ruletest menu commands.....	87
Ruletest toolbar.....	88
Commands on a Testsheet context-sensitive menu.....	89
Testsheet tabs.....	91
Populate the input panel.....	92
Execute tests.....	94
Format of decimal scale.....	95

Create multiple test scenarios on the same testsheet.....	95
Create multiple test scenarios as a set of testsheets.....	96
Create a sequential test using multiple testsheets.....	97
How to add testsheets.....	98
How to rename testsheets.....	99
How to associate one child entity with more than one parent.....	99
How to save a Ruletest.....	101
How to import a JSON document to a testsheet.....	101
How to export a testsheet as a JSON document.....	102
How to create a Ruletest report.....	103
 How to document rule assets.....	105
About comments.....	105
Add asset-level comments.....	108
Add item-level comments.....	109
Use the Comments view.....	109
 The Corticon Studio reporting framework.....	111
 Studio license expiration.....	115
 How to exit Corticon Studio.....	117

A guide to Progress Corticon.js Studio

The Progress Corticon.js Studio is where rules are created, maintained, tested, and packaged for deployment. The Corticon.js Studio workbench comprises several work sections for your interactive rule development and testing.



The highlighted sections in this illustration are:

- **Title bar** - Displays the current workspace, and the active project's active file in the editor.
- **Menu bar** - Provides access to Studio commands. The active file's type adds its menu.
- **Tool bar** - Provides access to most Studio commands. The active file's type adds its bar to the toolset.
- **Project panel** - Lists all the projects that are in the workspace. Expanding a project opens its folders and lists its files.
- **Editor window** - Enables the editing functions for the file's type. When other files open in the editor window, they all are available but only one is active.
- **Operators panel** - Provides hints to the rule language syntax and usage and also lets you drag and drop selected operators into rules.
- **Related properties and problems** - Comments, rule statements/messages, and added information are relative to the active window. Settings for the active window's file are maintained here. Problems are workspace wide listings.

Corticon assets in a Rule Project

In this view of the Corticon Studio, the Cargo [Rule Project](#) is loaded. The editor view has each of the four file types in a Corticon project open:

- **Vocabulary** - An `.ecore` file is structured dictionary containing all necessary business terms and relationships between them used by the business rules. In this illustration, the Vocabulary is the active window.
- **Rulesheet** - An `.ers` file is a set of conditions and actions and plain language statements written from a common business Vocabulary. A Rulesheet is associated with one Vocabulary. Multiple Rulesheets can use the same Vocabulary.
- **Ruleflow** - An `.erf` file is a set of one or more Rulesheets, and other Ruleflows organized on a canvas for sequential execution. Corticon.js generates JavaScript for a Ruleflow when you package rules for deployment.
- **Ruletest** - An `.ert` file is a set of one or more Testsheets containing sample data used for testing Rulesheets and Ruleflows, all based on the same Vocabulary. While optional, they are the essence of Corticon's test-as-you-go structure.

Once you have defined a Vocabulary, you build Rulesheets that you then assemble into Ruleflows that can include additional functions. You can then create tests to run against the Ruleflow right in the Studio as though it were deployed.

The project illustrated also shows that some supporting files that are often placed in the project space; in this case, a JSON test request.

Initial menu commands

Many menu commands are standard behaviors in any development environment. Commands that are key to Corticon Studio are available when Corticon.js Studio opens:

File Menu

The Corticon actions accessed on the File menu are:

- **New** - Creates the specified Corticon.js Studio resource dialogs to create Corticon rule projects, their assets and files in the file system. You generally want to create Rule Projects instead of just Projects to ensure that the Corticon Nature is applied.
- **New > Folder** - Creates a new file folder.
- **Import** - Enables import of archive files, project folders, or loose files from your file system into an existing project. See [Importing or exporting a Rule Project](#).
- **Export** - Enables export of resources of an existing project to archive files, or the file system. See [Importing or exporting a Rule Project](#).
- **Switch Workspace** - Select a different workspace. Lets you choose whether to save each open file in the editors before restarting into the selected workspace. It is a good idea to update rule assets after you choose a workspace that has not been used recently.

Edit Menu

The initial Corticon actions accessed on the Edit menu are the basic cut, copy, paste, delete, and select. You might see that the undo/redo actions are available.

Project Menu

The Corticon actions accessed on the Project menu are described in [The Project Explorer window](#) on page 21. They have limited functionality when there are no open projects.

Vocabulary, Rulesheet, Ruleflow, Ruletest Menus

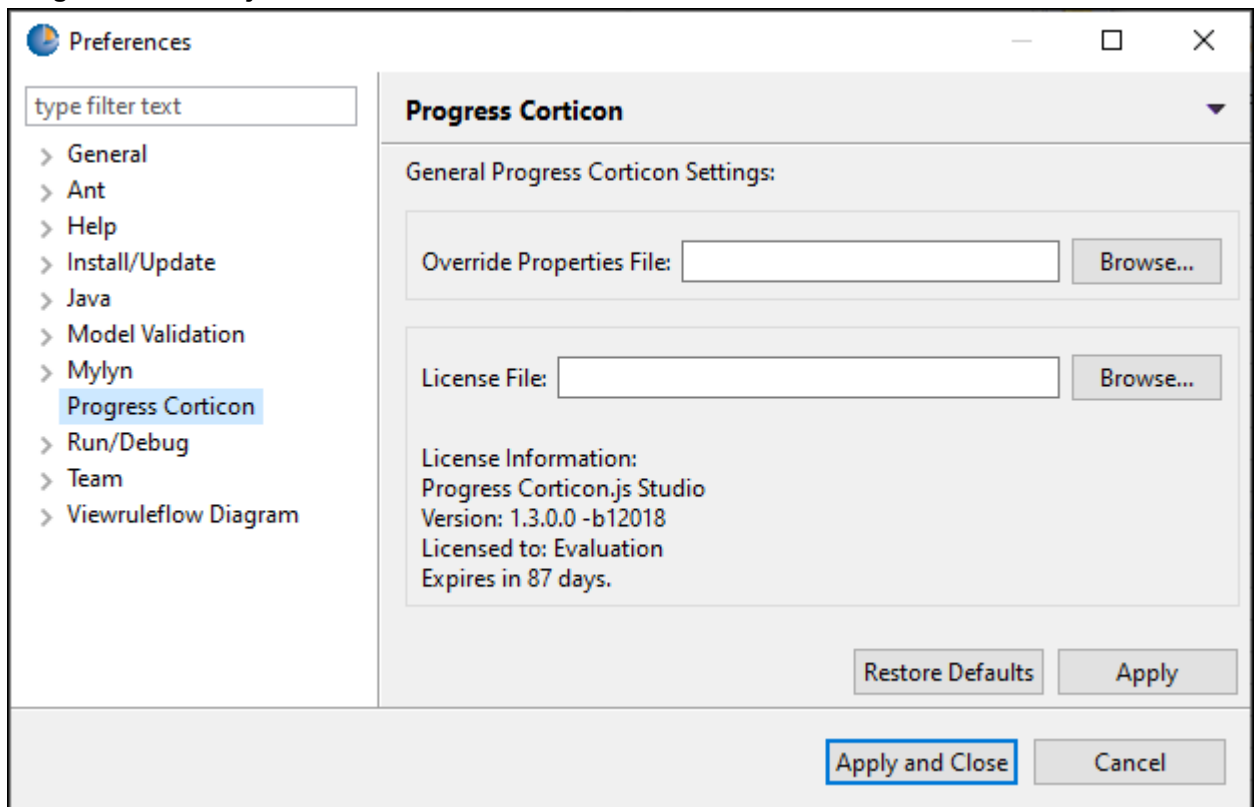
Note: When a Corticon.js file is active in the Editor Window, its file type's menu replaces another type's menu in the menubar. Each of these is described in a separate topic, as follows:

- [Vocabulary menu commands](#) on page 30
- [Rulesheet menu commands](#) on page 47
- [Ruleflow menu commands](#) on page 65
- [Ruletest menu commands](#) on page 87

Window Menu

The Corticon actions accessed on the Window menu are:

- **Show View** - Adds a perspective-related view as a tab in the perspective window. Views in the Corticon.js Designer include **Comments**, **Localization**, **Natural Language**, **Problems**, **Properties**, **Rule Messages**, **Rule Operators**, **Project Explorer**, and **Rule Vocabulary**. **Other** lists all available views, whether or not directly relevant in the current perspective.
- **Open Perspective** - Opens a defined set of views and editors. You might want to switch to Corticon Classic to extend the Studio user interface to include Eclipse tooling that Corticon Studio does not typically use.
 - **Corticon.js Designer** - Resets the layout to the views commonly used default Corticon user interface.
 - **Other** - Opens the perspective dialog where you can choose an available perspective, such as **Corticon Designer Classic**, the perspective that shows a richer set of Eclipse options.
- **Preferences** - Opens the **Preferences** dialog for the installation. Corticon.js preferences include:
 - **Progress Corticon.js:**



-
- **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.

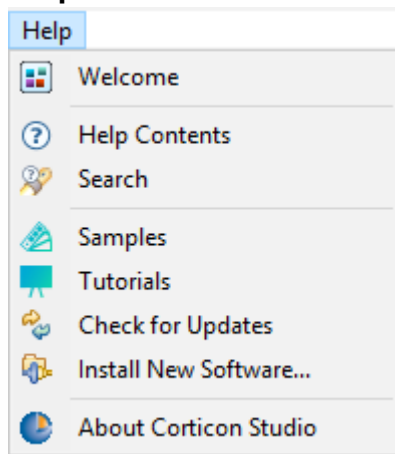
Note: Changing the override properties file -- and changes within an override file --- are not applied until you restart the Studio.

- **License file path and information** - The current license file information is displayed. You can point a preferred license file or license file location.

-
- **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.

Help Menu



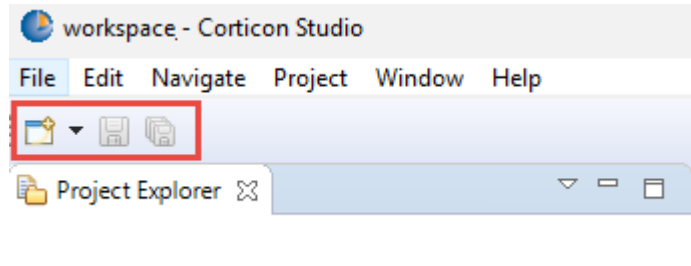
The Corticon actions accessed on the Help menu are:

- **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 that opens is not the top page.
- **Help Contents** - Opens the help system window where you can access the Corticon Information Hub for up-to-date documentation for the latest release of the Studio and its related Corticon tools. You can also search for terms, access documentation archives, and browse the help structure of Eclipse and third-party tools.
- **Samples** - Opens the **Welcome** page's **Samples** panel to access and unpack Corticon.js sample projects.
- **Tutorials** - Opens the **Welcome** page's **Tutorials** panel to access the tutorial documents on the Corticon Information Hub.
- **Check for Updates** - Allows you to access various Eclipse sites to update or add plugins to Studio. The Corticon plugins are not available from an update site.
- **About Corticon Studio** - Details versions and licenses of installed Corticon.js Studio, Eclipse, and other installed products. Also provides access to installation details.




Note: Keyboard shortcuts - Many menu commands have keyboard shortcuts displayed next to their command name. There are also many other general and Eclipse shortcuts that might be relevant in your work. You can open the list of shortcuts by pressing **CTRL+SHIFT+L**. The complete list is accessible at **Window > Preferences**, and then **General > Keys**.

Initial tool bar

The initial Corticon.js Studio toolbar shows tools that create and save assets in open projects. These tools are also accessible as menu commands.



The tools initiate the following actions:

-  > **New** opens the dialogs to create Corticon rule projects, their assets and files in the file system. You generally want to create Rule Projects instead of just Projects to ensure that the Corticon Nature is applied.
-  Saves changes to the active file in the editor window.
-  Saves changes to all open files in the editor window.

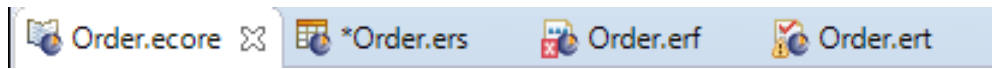
For details, see the following topics:

- [The active Corticon Studio window](#)
- [Alternate settings for Corticon Studio properties](#)
- [File naming restrictions](#)
- [How to rename and relocate assets in the Project Explorer](#)
- [How to use Git to manage project assets](#)

The active Corticon Studio window

Any single open window in Corticon.js Studio can be the *active window*. Indication of active status is provided by the window's tab color. As shown below, the active window's tab is colored **white**, while inactive tabs are colored **gray**. "Shifting focus" (that is, activating a different window), is accomplished simply by clicking anywhere within an inactive window or on the window's tab.

Below, the tabs of four Corticon.js Studio windows are shown in the editor space.



- The active window's tab is white, **Order.ecore**
- The `.ers` file is not the active window. It has been changed but not saved, as indicated by the asterisk preceding the tab's name.
- The `.erf` file is not the active window and has been saved, but the red square overlaying the window's file type icon indicates there is an error somewhere in the file content, as described in the **Problems** window.
- The `.ert` file is not the active window and has been saved, but the yellow triangle overlaying the window's file type icon indicates there is an warning somewhere in the file content, as described in the **Problems** window.

Alternate settings for Corticon Studio properties

Corticon changes many behaviors based on properties that you can customize in

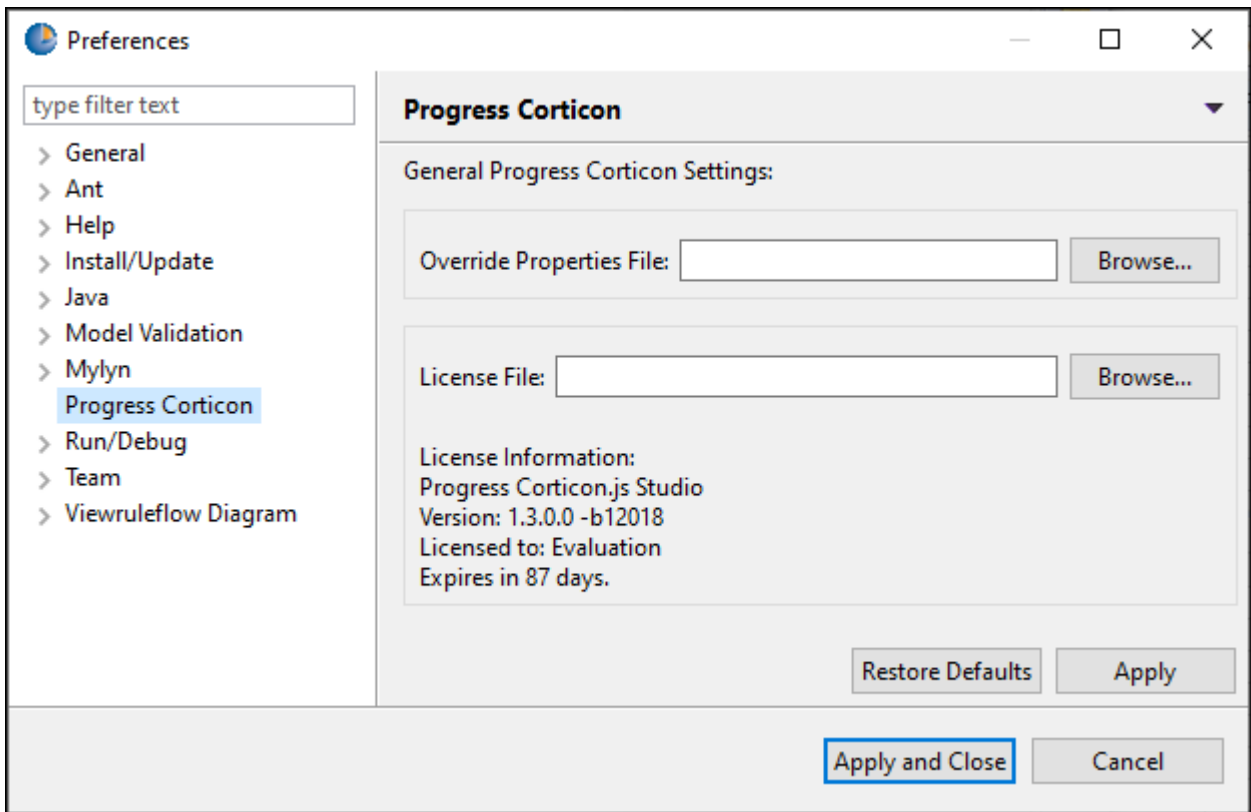
`[CORTICON_WORK_DIR]/brms.properties`. Studio enables you to specify variations of this file that you can locate and name as suits you.

Using an alternate location and name for the Studio's properties file

You can choose to point Corticon.js Studio to a preferred location or name of the `brms.properties` file.

To set an alternate location of the properties settings file in Studio:

1. The file must exist before you can point to it. You can copy the default file located at Studio's installation root, or you can just create a new file and give it your preferred name, such as `my_brms.properties`. Save the file at your preferred location, such as `C:\preferences\`.
2. In Studio, choose **Window > Preferences**, and then choose **Progress Corticon**, as shown:



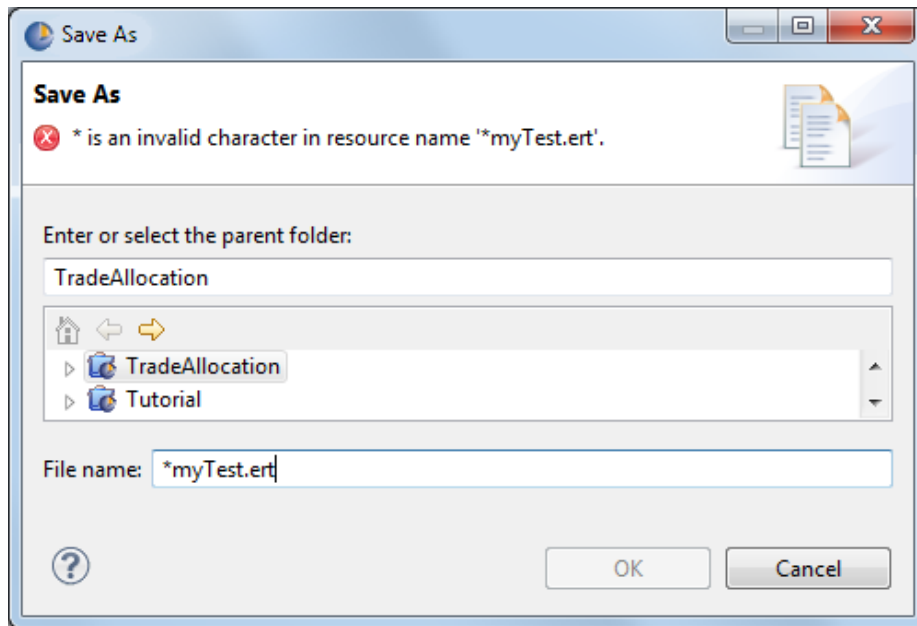
3. Enter or browse to the override properties file you created, and then select it.
4. Click Apply and Close.
5. Restart Corticon.js components for their changes to take effect.

When Corticon.js starts up, it will apply your properties file with your settings.

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:



How to rename and relocate 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'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.

How to use Git to manage project assets

Progress Corticon recommends using a source code repository to manage files in your projects. Source code repositories are software version control systems that manage access to files in a repository, and help maintain current and historical versions of the files.

Corticon Studio bundles a plugin for the popular version control system **Git**.

Use Git, or another source code control repository to perform important file management operations. These systems provide several features and options but typically you will use only a small handful of them on a daily basis for:

- Checking-out a rule project you need to work on
- Adding new rule assets to a project and committing them to the repository to share with others
- Making changes to rule assets and committing them back to the repository
- Changing the organization of rule assets by moving them to different folders
- Adding markers to the repository to identify the version or assets at a release milestone

Corticon rule assets are XML files. You can compare different versions of XML files using a `diff` command; however, the output is in XML format which can be difficult to interpret.

The EGit plugin provides a Git perspective, a **File > Import** function, and **Preferences** in the **Team** section. The EGit plugin bundled with Corticon Studio provides basic command line features. For information about EGit advanced command line features, see <http://www.eclipse.org/egit/documentation/>.

To access Git documentation, downloads, and community, see <https://git-scm.com/doc>

Rule Projects

In Corticon.js Studio, a Vocabulary, as well as any Ruleflows, Rulesheets and Ruletests associated with that Vocabulary, must be stored in a Rule Project.

If you use the default workspace, when you upgrade to later versions of Progress Corticon.js Studio, your Rule Projects will remain and survive the upgrade process intact.

For details, see the following topics:

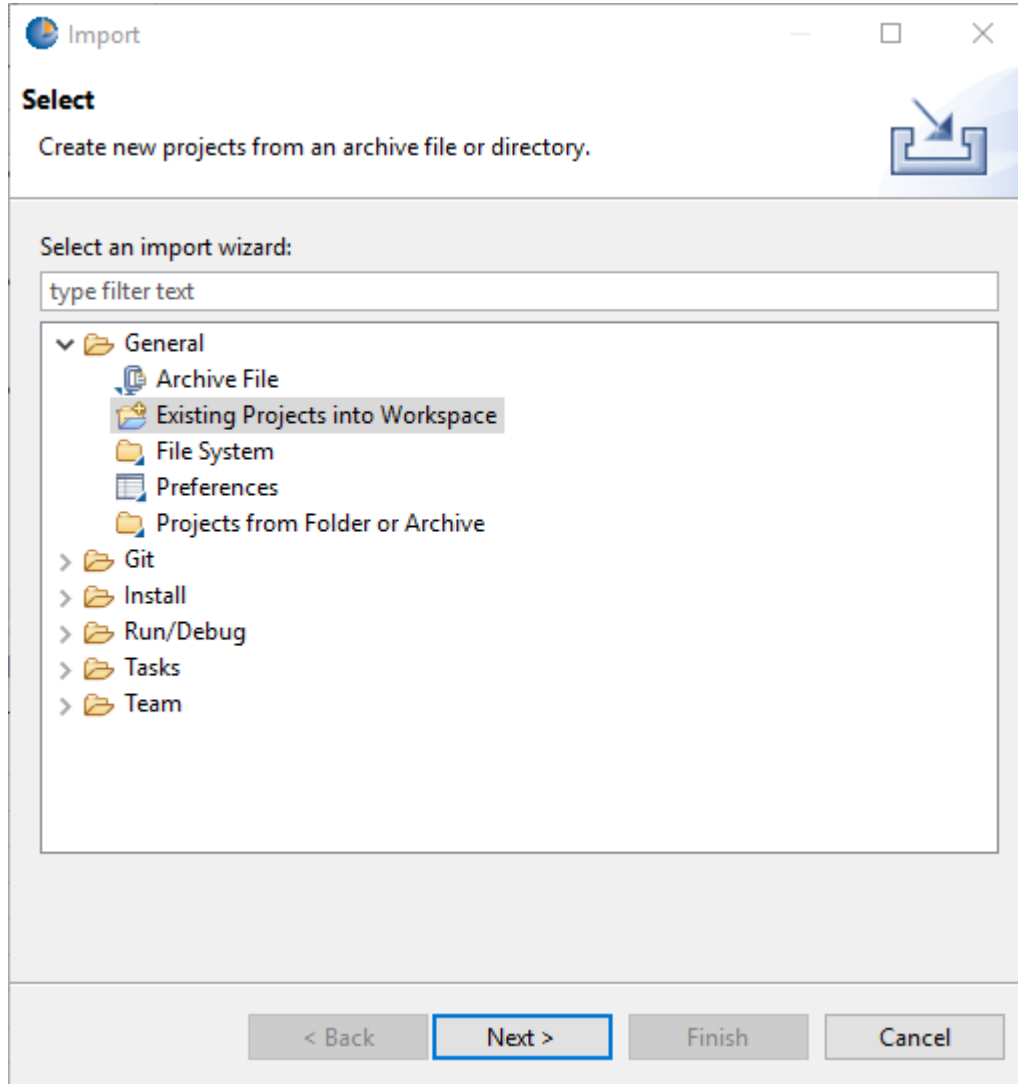
- [How to import or export a Rule Project](#)
- [How to create a Rule Project](#)
- [The Project Explorer window](#)

How to import or export a Rule Project

In collaborative development environments, you might need to bring existing rule projects into your workspace, and then, after adding and changing assets, exporting the rule project assets for other developers to access.

Note: The import and export functions are documented at **Help > Help Contents**, and then **Workbench User Guide > Tasks > Importing** and **Exporting**.

Choose **File > Import**. In the dialog, **General** options let you choose from existing projects, the file system, and archives. Imports assume the current workspace as the target folder.



After importing a project, be sure to upgrade rule assets before making changes to assets. If the source project was created and saved in a higher version of Corticon, unpredictable results can occur.

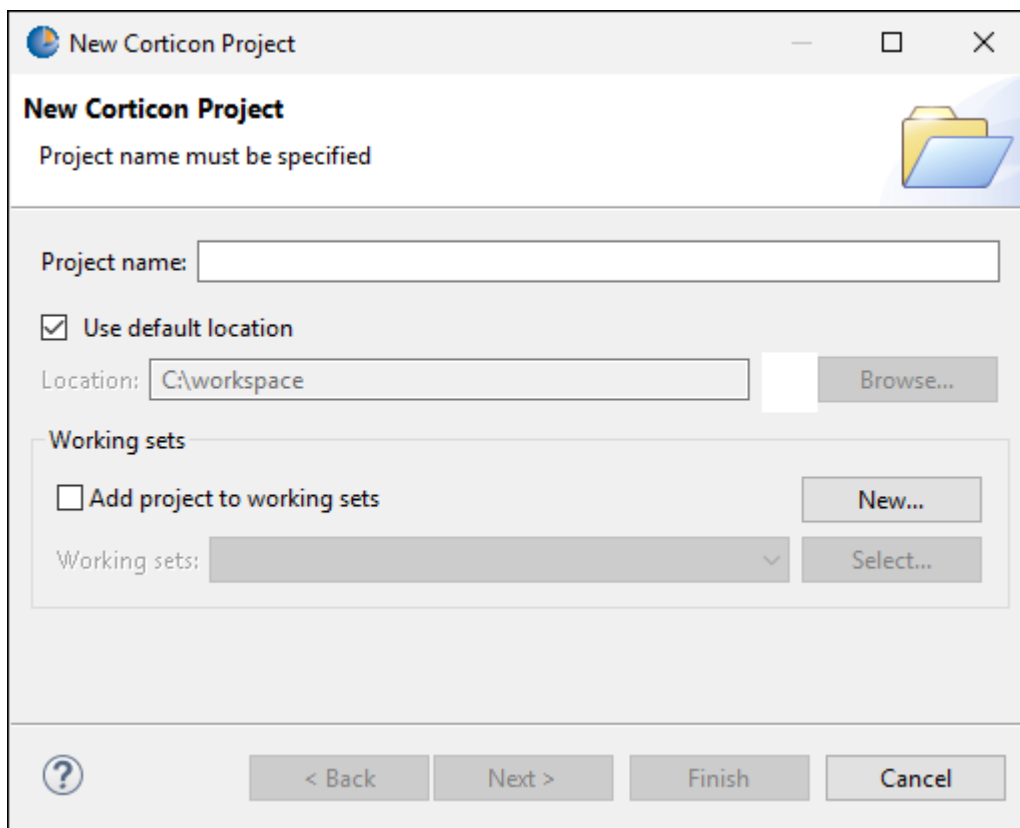
Before performing a project export to the file system or an archive, choose **Save > All**, and then choose **Close > All**.

How to create a Rule Project

To create a new Rule Project, either:

- In Corticon Studio, select **File > New > Rule Project**
- Select **New** on the toolbar, and then select **Rule Project**.

1. The **New Corticon Project** dialog box opens:

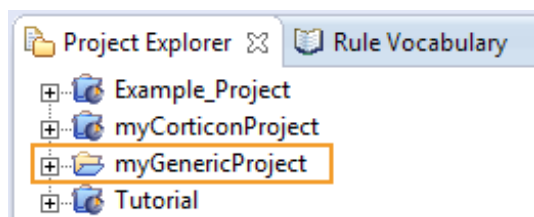


2. Check the **Use default location** option
3. Enter a name for your project in the **Project name** field. Notice that the path to your new Rule Project is automatically appended to Corticon.js Studio's default workspace folder to define its **Location**.
4. Click **Finish** to create your new Rule Project.

Corticon Nature

If you chose Project instead of Rule Project, you miss out on the Corticon Nature.

Corticon.js Studio relies on your project having a "Corticon Nature" that identifies it as a Corticon project and not just a generic Eclipse project. The following illustration shows both Corticon.js and generic Eclipse projects. Projects having a Corticon Nature are distinguishable by the Corticon icon added to the project icon.

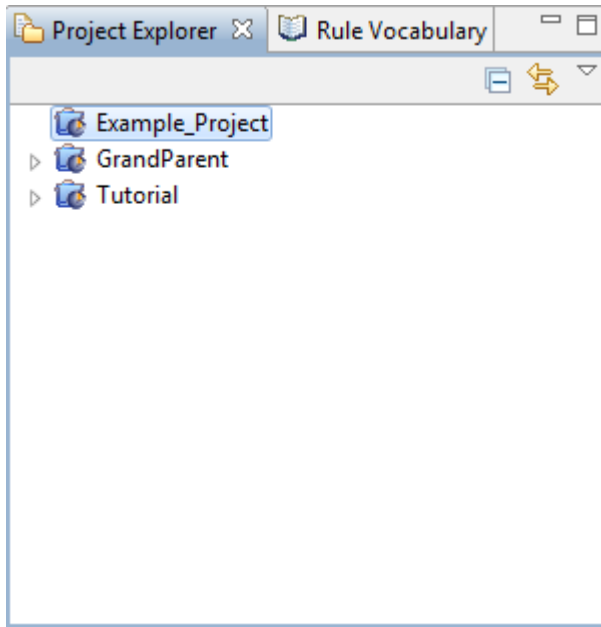


To convert a generic project to a Corticon project, right-click on the project's folder, and then select **Add Corticon Nature**.

The Project Explorer window

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

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



Project Menu

The **Project** menu has standard Workbench functions, as well as the following:

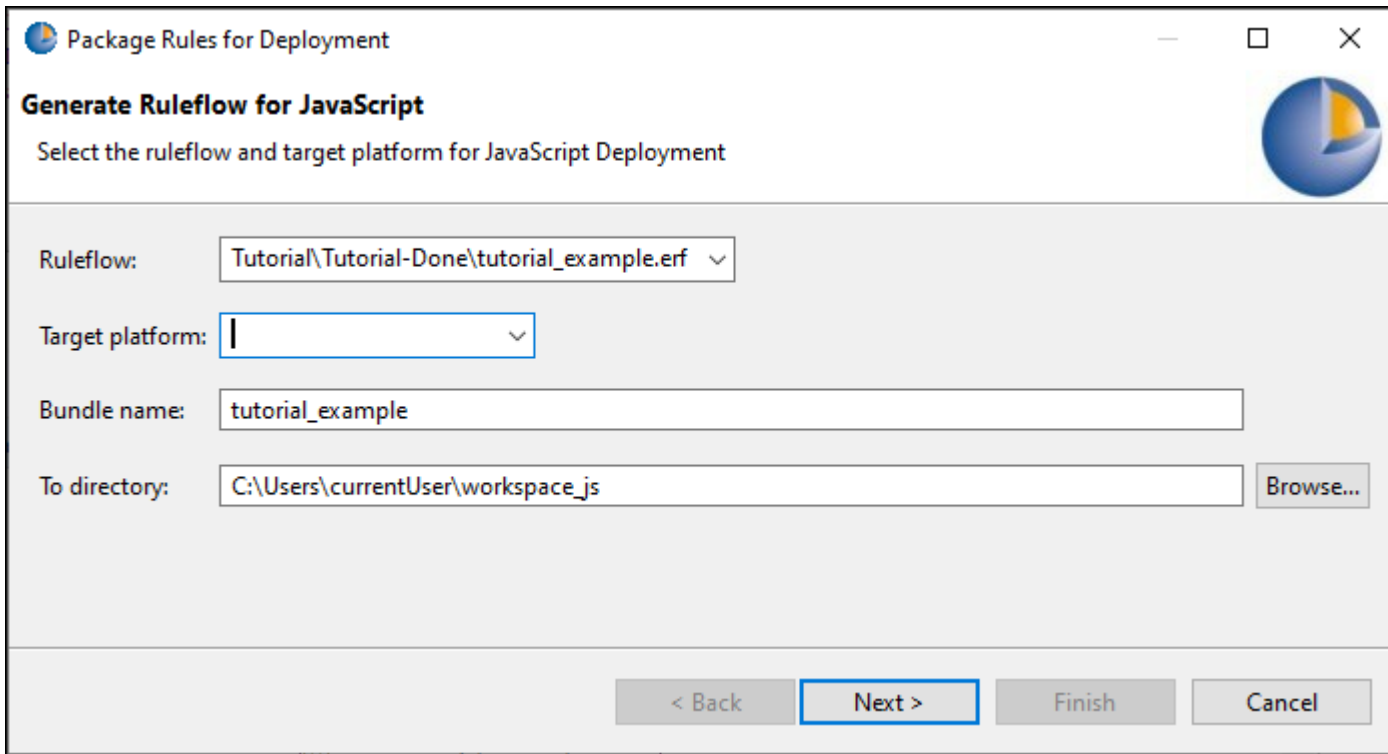
- **Validate Project** - Project validation loads every asset in the project into memory and validates it to identify any errors or warnings. On large projects, project validation can use significant memory and take time to complete. Project validation is done in the background such that you can continue to use Studio while validation is ongoing. The progress of project validation is shown in the footer of the Corticon Studio window. It can also be viewed in the Progress view by selecting **Window > Show View > Other** then **General: Progress**
- **Upgrade Rule Assets** - It is important that you keep project assets in synch with the Studio. If you brought new assets into your workspace, you should perform this action. Opens the **Upgrade Corticon.js Assets** dialog. For a discussion of this feature, see *"How to upgrade projects coming forward from a prior release" in the Corticon.js Installation Guide*.
- **Package rules for deployment** - Lets you choose selected Ruleflows to package for deployment.
- **Properties** - Opens the dialog that provides several advanced features.

These commands are also accessible on the dropdown menu that opens when you right-click on a project.

How to package rules for deployment

To select specific Ruleflows to be packaged for deployment from the Project Explorer view, do the following:

1. Select a project or folder in your Project Explorer view.
2. Right-click and select **Package rules for deployment**. You are forced to save all changes before the packaging dialog opens:



Package Rules for Deployment

Generate Ruleflow for JavaScript

Select the ruleflow and target platform for JavaScript Deployment

Ruleflow: Tutorial\Tutorial-Done\tutorial_example.erf ▾

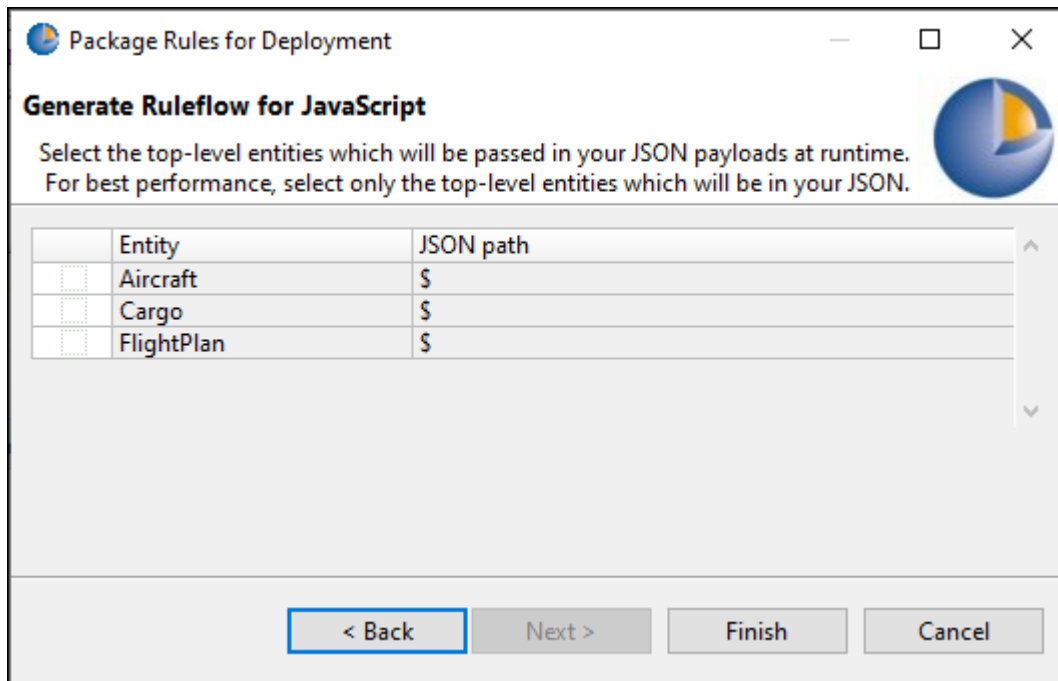
Target platform: ▾

Bundle name: tutorial_example

To directory: C:\Users\currentUser\workspace_js Browse...

< Back Next > Finish Cancel

3. In the **Generate Ruleflow for JavaScript** dialog:
 - a. Select a Ruleflow. All the Ruleflows in the selected project are listed as candidates for deployment.
 - b. Select a Target platform.
 - c. Confirm the bundle name. You can edit the bundle name.
 - d. Confirm the To Directory.
 - e. Click **Next**.
 - f. In the next dialog:



- g. Select the top-level entities that will be passed in your JSON payloads at runtime. Choosing none, selects all.
- h. Click **Finish**

Vocabularies

A Corticon project typically has exactly one Vocabulary. The Vocabulary defines the entities (the things in the model), then attributes of each thing, and finally associations to describe how things relate to each other. It is used to build rule models in [Rulesheets](#) on page 45 and test cases in [Ruletests](#) on page 83.

See *"Build the Vocabulary" in the Rule Modeling topics* for the steps to create a Vocabulary model from scratch.

For details, see the following topics:

- [How to create a Vocabulary](#)
- [How to define a Vocabulary](#)
- [How to find references to an entity, attribute, or association in a project](#)
- [How to refactor entity attribute or association names in a project](#)

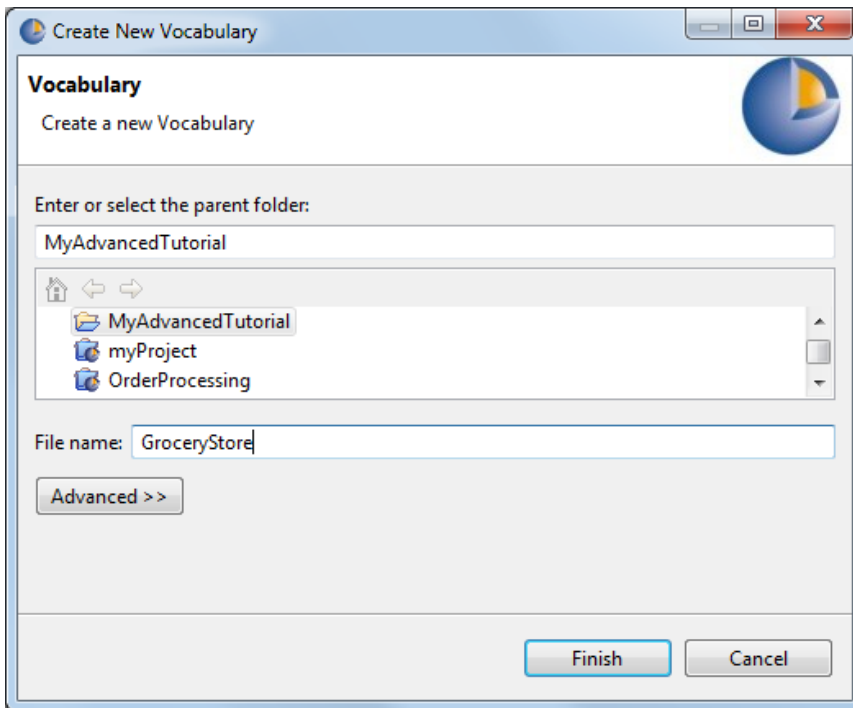
How to create a Vocabulary

To create a new Vocabulary:

1. Do one of the following:

- Select **File > New > Rule Vocabulary** from the Corticon.js 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, as shown:



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.js and should not be used.

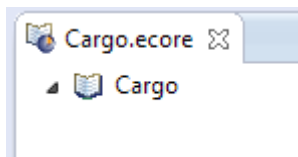
4. Click **Finish** to create your new Vocabulary.

The Vocabulary window

The **Vocabulary** window is the Editor window that displays when its tab is the active window. A Vocabulary file has the extension `.ecore`. As the active window, the Vocabulary menu is available and the toolbar offers Vocabulary tools.

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 toolbar

When the Vocabulary editor is active, its tools are added to the toolbar

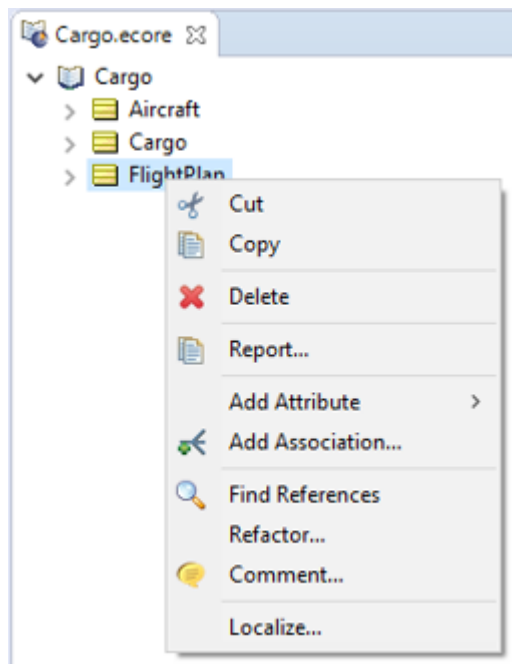
The Vocabulary tools provide the same functions as the corresponding **Vocabulary** menu commands:

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

Commands on the Vocabulary context-sensitive menu

In addition to the menubar and toolbar functions, Corticon.js Studio also provides many Vocabulary functions within a right-click context 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.

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.
- **Add Entity** - Adds an Entity to the Vocabulary.
- **Add Attribute** - Lets you choose the basic data type and then add 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.
- **Find References** - Finds references to an entity, attribute, or association in a project.
- **Refactor** - Refactors entity, attribute, or association names in a project.
- **Report** - Creates an HTML report and launches your browser for viewing. See [Creating a Vocabulary Report](#).

How to define a Vocabulary

The Vocabulary is the essential catalog of elements that define the use case of the project.

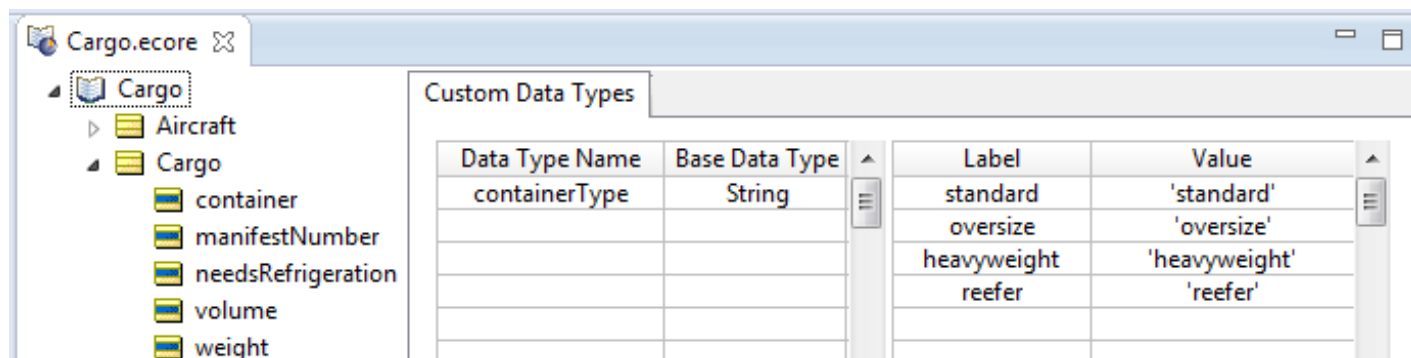
The topics in this section add to what is described in the Corticon.js Rule Modeling guide, specifically:

- *"Generate a Vocabulary"*
 - *"Use JSON to generate a vocabulary"*
 - *"Use JSON Schema to generate a vocabulary"*
- *"Build a Vocabulary by hand"*

Custom data types tab

At the root of every Vocabulary in its editor, the tab **Custom Data Types** displays.

You can define lists of values that are the set of allowable values associated with a Vocabulary attribute. The Tutorial sample demonstrates how to delimit the options for a `containerType` by defining labels and their respective values:



For more information about enumerations, see *"Enumerations defined in the Vocabulary" in the Rule Modeling Guide*.

Add nodes to the Vocabulary tree view

Nodes are the elements that describe the structure of the Vocabulary.


Vocabulary node naming restrictions

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

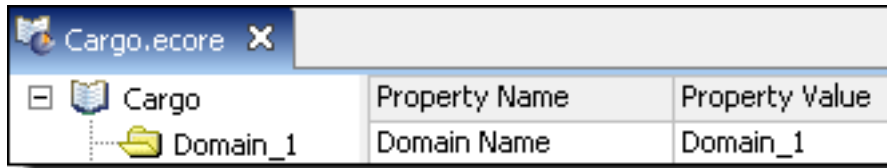
- Domain, Entity, Attribute, and Association Role node names can contain only alphanumeric characters (letters and numbers), and underscores. However:
 - Domain, Entity, Attribute, and Association Role node names can **not** begin with a number.
 - Spaces are **not** allowed in Domain, Entity, Attribute 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`, `floor`, and `month`) 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 *Build the Vocabulary* topics 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.

Add and edit domain nodes and their properties

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

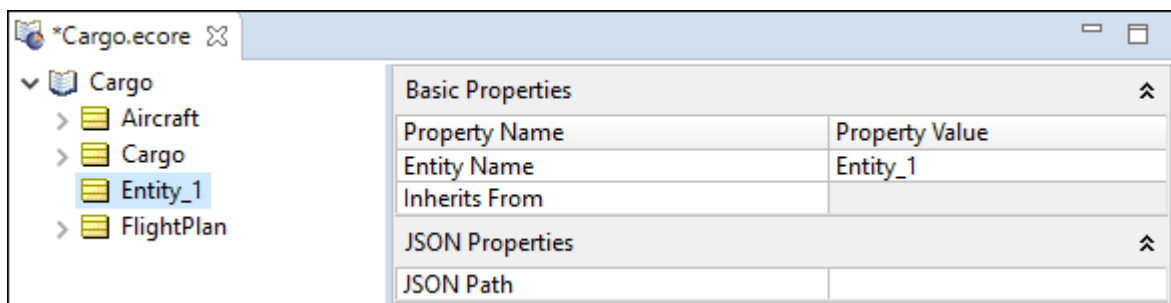
1. Select the name node in the Vocabulary tree view, which is the one with the  icon.
2. Do one of the following:
 - Right-click to display the pop-up menu and choose **Add Domain**

- 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.



Add and edit entity nodes and their properties

1. Select the Vocabulary node in the Vocabulary where you want to add an entity
2. Right-click and then choose **Add Entity**
3. Select the Entity in the Vocabulary tree to display its properties, as shown:



4. Assign property values as listed in the Basic Entity Properties table:

Table 1: Entity: Basic 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
Inherits From	Not used in Corticon.js Studio.

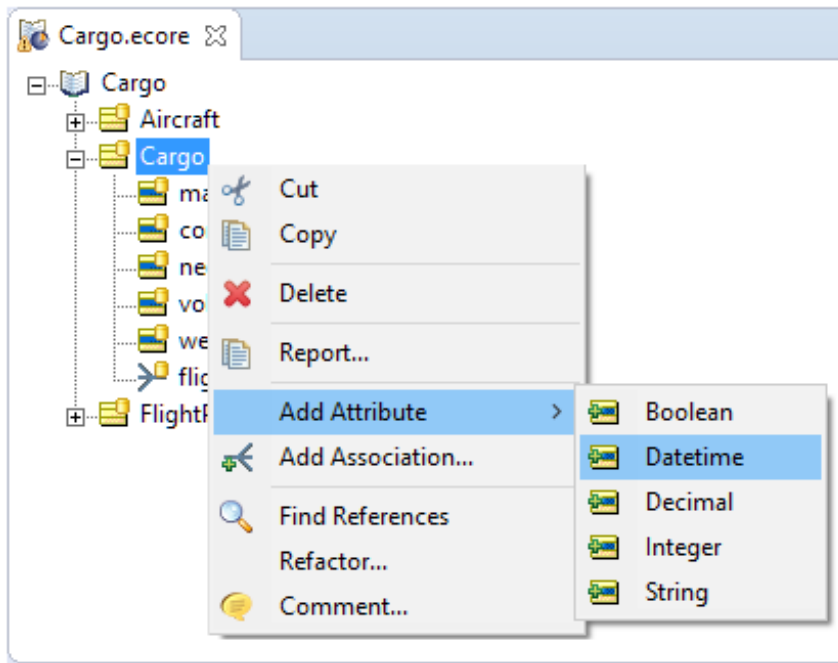
Table 2: Entity: JSON Properties

Property	Value
JSON Path	Blank for manually created Vocabularies. See "Root, paths, and top-level entities" in the Corticon.js Integration Guide for more information.

-->

Add and edit attribute nodes and their properties

1. Select the Entity node in the Vocabulary tree view where you want to add an attribute.
2. Right-click, choose **Add Attribute**, and then use its submenu to specify the datatype of the attribute, as illustrated for a Datetime attribute:

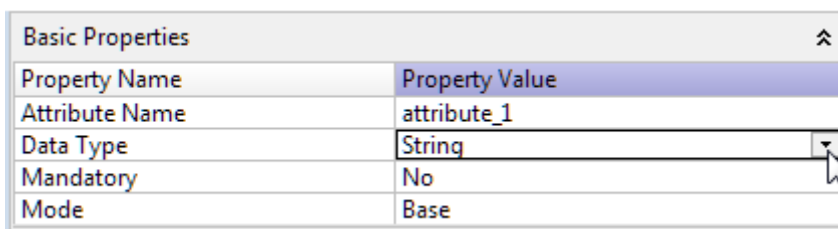


(You could instead choose the menu command **Vocabulary > Add Attribute**.) Note that the toolbar button

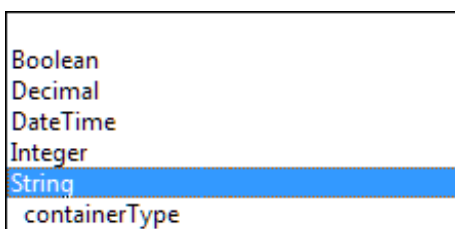


always creates a String attribute.

3. After creating the new attribute, the editor cursor lets you modify the default name from `Attribute1` to your preferred name.
4. If your preferred data type is a Custom Data Type, click on the dropdown menu for the data type:



5. Select the defined Custom Data Type you want to use for this attribute:



6. Assign the other property values as described in the following table.

Note:

The following table lists the basic properties common to every Attribute.

Table 3: Basic Attribute Properties

Property	Value
Attribute Name	Assigns 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; however, you should choose to refactor the name instead to insure that it perpetuates throughout the project.
Data Type	The default value is String. Other available data types: Boolean, Decimal, DateTime, Integer. You can also have a custom data type.
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 <code>No</code> will always include a <code>null</code> value selection in its Rulesheet drop-downs.
Mode	<p>Choose the attribute's Mode from the drop-down list.</p> <p><code>Base</code> attributes map to object properties processed at runtime.</p> <p><code>Transient</code> attributes are <i>derived</i> fields; they exist only during rule execution. They are not included in the objects produced at runtime. 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, the transients are included.</p> <hr/>

Table 4: Entity: JSON Properties

Property	Value
JSON Element name	Blank for manually created Vocabularies. See <i>"JSON payloads and results in Corticon.js"</i> in the <i>Corticon.js Integration Guide</i> for more information.

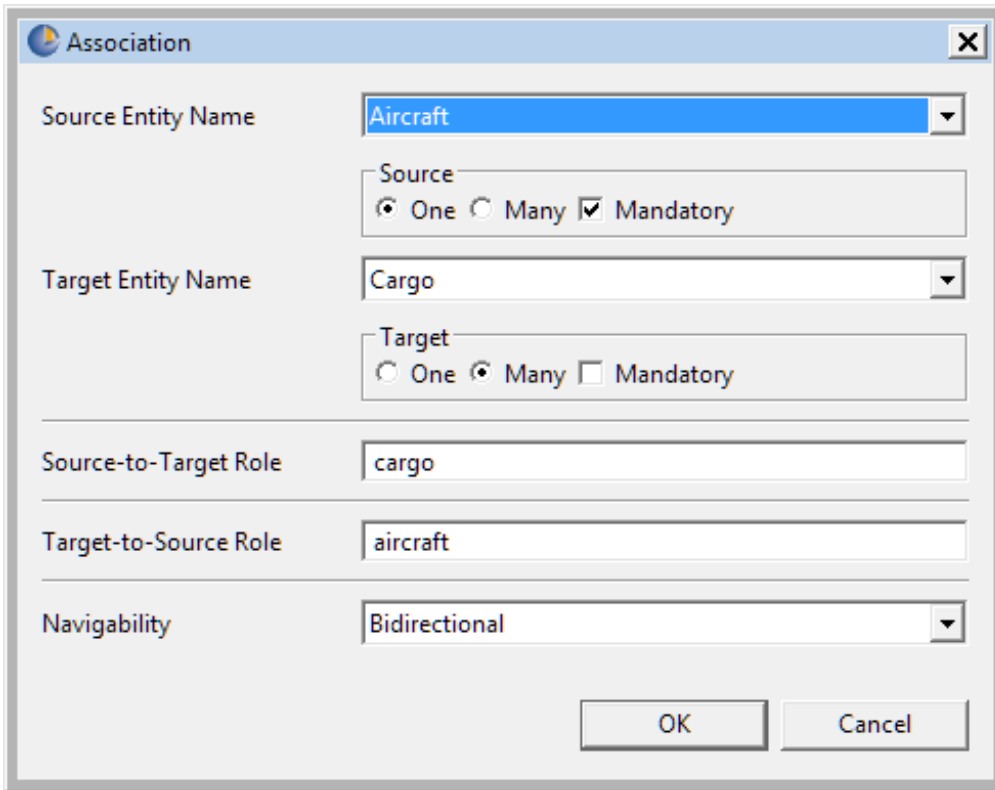
Add and edit association nodes and their properties

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.

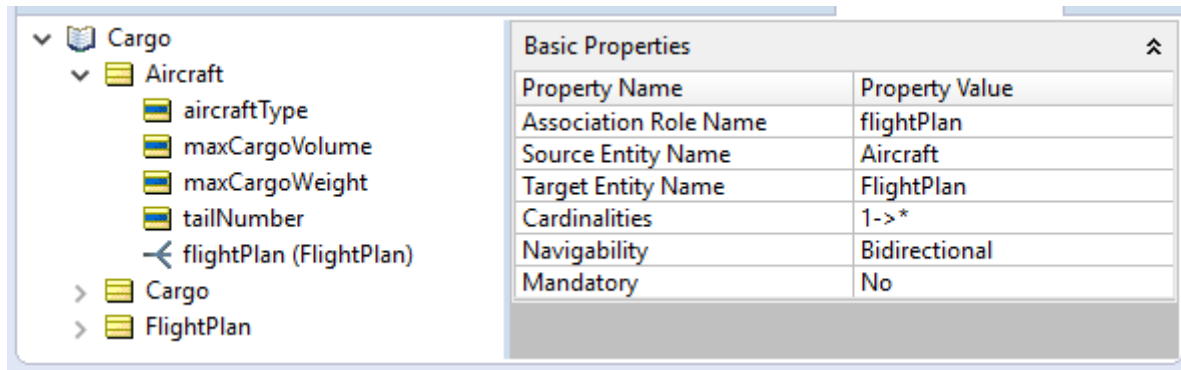
3. Complete the Association window as shown in the following table:




Property	Value
Source Entity Name	An association relates two entities. Corticon.js 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.

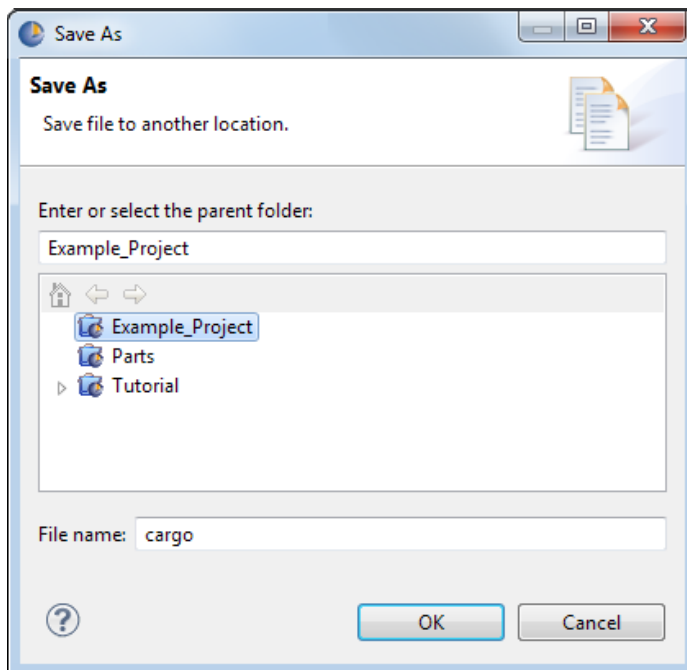
Property	Value
Cardinality Radio Buttons	Select the combination of radio buttons that describe the relationship you want between the two entities. The possible combinations are:
	One-To-Many. This is the default and is shown as 1>* in the properties. This means that a given instance of the supplier entity 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 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 <i>from</i> the Target entity <i>to</i> the Source entity from being displayed in the Vocabulary tree view. This option prevents a rule from being written using the scope <code>Target_entity.source_entity.attribute</code>.</p> <p>Source Entity < Target Entity. Use this option to prevent the association <i>from</i> the Source entity <i>to</i> the Target entity from being displayed in the Vocabulary tree view. This prevents a rule from being written using the scope <code>Source_entity.target_entity.attribute</code>.</p> <p>See the <i>Rule Modeling Guide</i> for more information on using associations in rule modeling.</p>

To edit an association, click on it in the Vocabulary, and then change the property values in the right panel, as illustrated:



How to save a new Vocabulary

- 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.



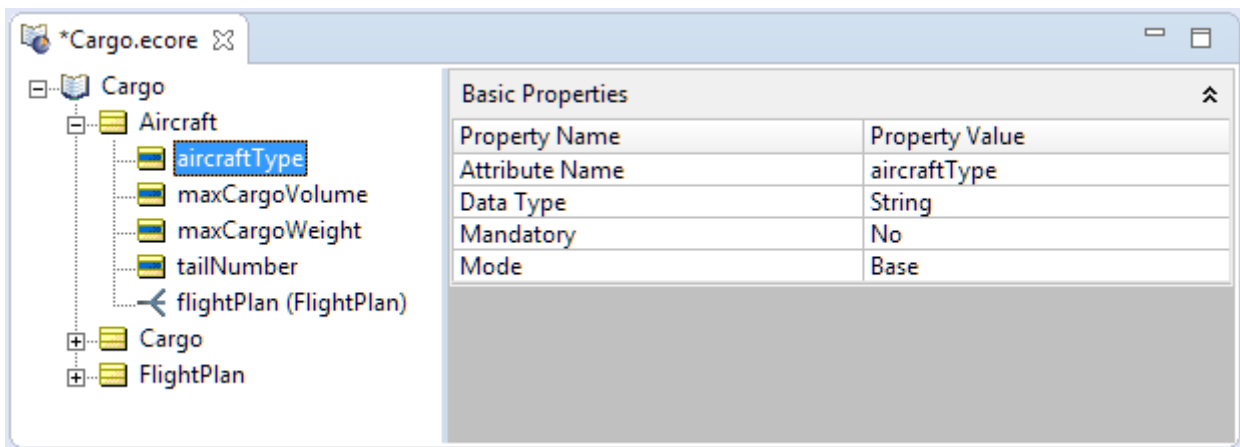
- Type any name, including embedded blanks (max. of 36 characters) in length, and click **OK**.
- See [File Naming Restrictions](#) for naming restrictions.

How to open an existing Vocabulary

1. Choose **File > Open File** from the Corticon.js 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**.

How to modify a Vocabulary

1. Choose the **Vocabulary Editor** tab to enter the Vocabulary editor.
2. Select the node you want to edit to display its properties, as shown:



3. Modify the properties with the following considerations:
 - 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. When you use *refactoring* instead, the changes are evaluated and changed throughout the project.
 - 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.

How to create a Vocabulary report

1. With the Vocabulary you want to report open as the current window, choose the menu command **Vocabulary > Report**.

The **Generate Report** dialog opens.

2. Choose the **Report Type**, **Report Style**, and **Output Folder** for this report.
3. Click **Finish**.
4. The Vocabulary report opens as a new HTML page in your default web browser, and then saves the HTML, XML, and CSS files in the specified output folder.

The default CSS and XSLT files are samples that you can customize to suit your preferences.

How to find references to an entity, attribute, or association in a project

When working on large projects with many assets, it is a real convenience to find everywhere an entity, attribute, or association is referenced in the project. This feature reveals the vocabulary item's usage and impact, and lets you easily navigate to each instance. Just right-click on an entity, attribute, or association name in the Vocabulary editor, and then choose **Find References** to initiate a search.

The search examines all Rulesheets, Ruleflows, and Ruletests in the project, then lists each match within each asset. Double-clicking on a match opens the asset and puts the focus at that location, as shown:

The screenshot shows the Progress Corticon IDE with the 'Cargo.ers' Rulesheet open. The 'Conditions' section is visible, showing a table with columns 0 through 5. The 'Actions' section shows a 'Post Message(s)' action with a 'Cargo.container' message. The 'Overrides' section shows a table with columns 0 through 5. The 'Search' window is open, showing 16 references to 'Cargo.volume' in the project. The search results are listed under the 'Tutorial' folder, with the first match highlighted: 'Condition cell: (b, 2); Expression: > 30'.

	0	1	2	3	4	5
a Cargo.weight		<= 20000	-	> 20000	-	
b Cargo.volume		-	> 30	<= 30	-	
c Cargo.needsRefrigeration		-	-	-	T	

	0	1	2	3	4	5
A Cargo.container		standard	oversize	heavyweight	reefer	

	0	1	2	3	4	5
Overrides			{1, 4}		{1, 3}	

'Cargo.volume' - 16 references in project

- Tutorial
 - Tutorial-Done
 - Cargo.ers
 - Condition cell: (b, 2); Expression: > 30
 - Condition cell: (b, 3); Expression: <= 30
 - Condition row: b; Expression: Cargo.volume
 - Scope variable: Cargo.volume
 - Cargo.ert

Right-clicking on a match presents a menu of navigation options as well as their keyboard shortcuts, as shown:

The screenshot shows the Progress Corticon IDE with the 'Cargo.ers' Rulesheet open. The 'Conditions' section is visible, showing a table with columns 0 through 5. The 'Actions' section shows a 'Post Message(s)' action with a 'Cargo.container' message. The 'Overrides' section shows a table with columns 0 through 5. The 'Search' window is open, showing 16 references to 'Cargo.volume' in the project. The search results are listed under the 'Tutorial' folder, with the first match highlighted: 'Condition cell: (b, 2); Expression: > 30'.

	0	1	2	3	4	5
a Cargo.weight		<= 20000	-	> 20000	-	
b Cargo.volume		-	> 30	<= 30	-	
c Cargo.needsRefrigeration		-	-	-	T	

	0	1	2	3	4	5
A Cargo.container		standard	oversize	heavyweight	reefer	

	0	1	2	3	4	5
Overrides			{1, 4}		{1, 3}	

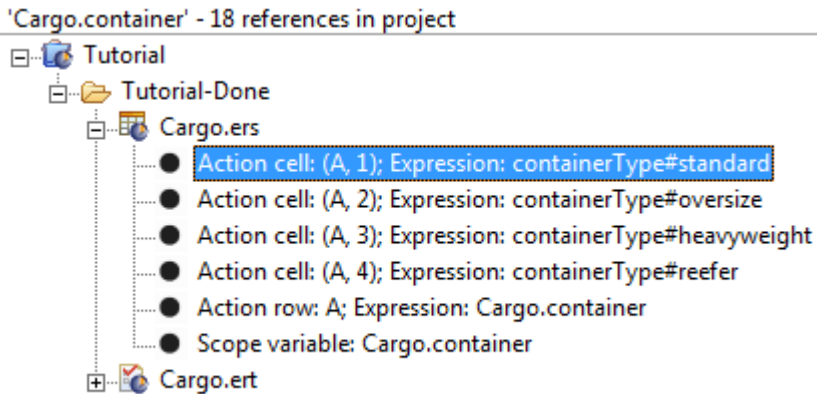
'Cargo.volume' - 16 references in project

- Tutorial
 - Tutorial-Done
 - Cargo.ers
 - Condition cell: (b, 2); Expression: > 30
 - Condition cell: (b, 3); Expression: <= 30
 - Condition row: b; Expression: Cargo.volume
 - Scope variable: Cargo.volume
 - Cargo.ert

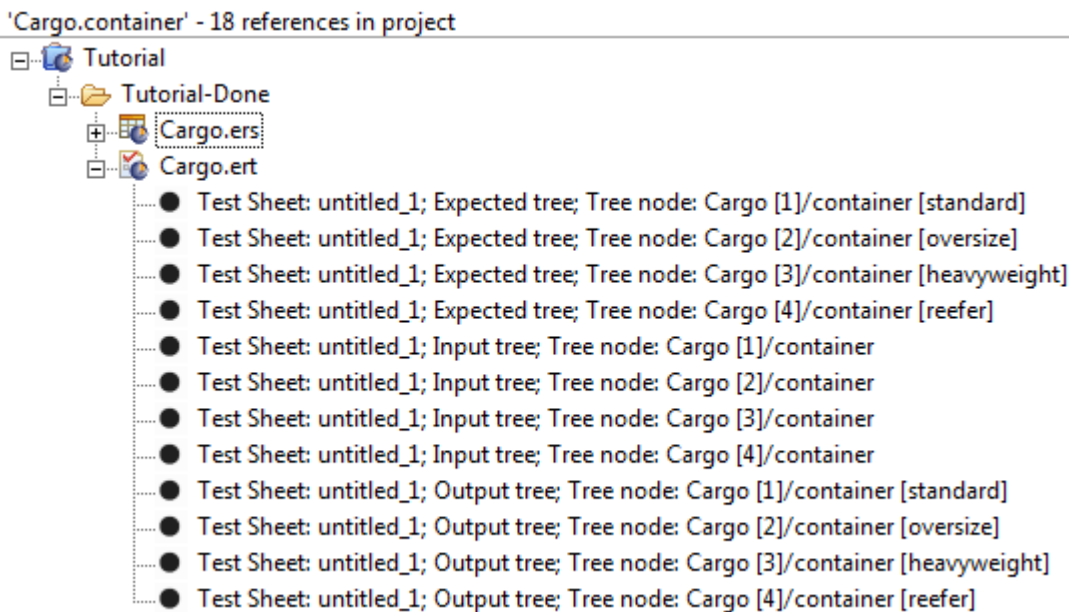
Right-click context menu options:

- Show In: Alt+Shift+W
- Next Match: Ctrl+,
- Previous Match: Ctrl+,
- Expand All
- Remove Selected Matches: Delete
- Remove All Matches
- Copy: Ctrl+C
- Search Again: F5

When the attribute is a custom data type, the search results indicate the Data Type Name and value, as shown:



When results are in Ruletests, the results detail the testsheet, tree, and node, as shown:



How to refactor entity attribute or association names in a project

As your Corticon projects evolve, you might have need to change the name of some entities, attributes, and associations to better describe their meaning.

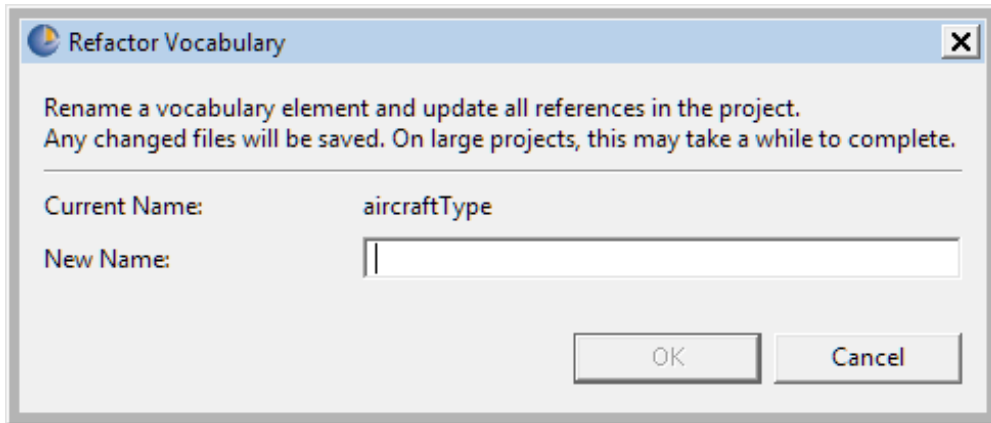
To avoid time consuming manual work and automate a potentially tedious process, Corticon lets you *refactor* the name of an entity, attribute, or association so that each change is automatically applied to all instances of the name in all assets in the project.

How does refactoring differ from renaming? When you *rename* a vocabulary element, only the vocabulary changes while the project assets that use that element are not updated, and will likely become invalid. But when you *refactor* the name of a vocabulary element, Corticon Studio updates all the Rulesheets, Ruleflows, and Ruletests which use that element, and maintains the validity of your project.

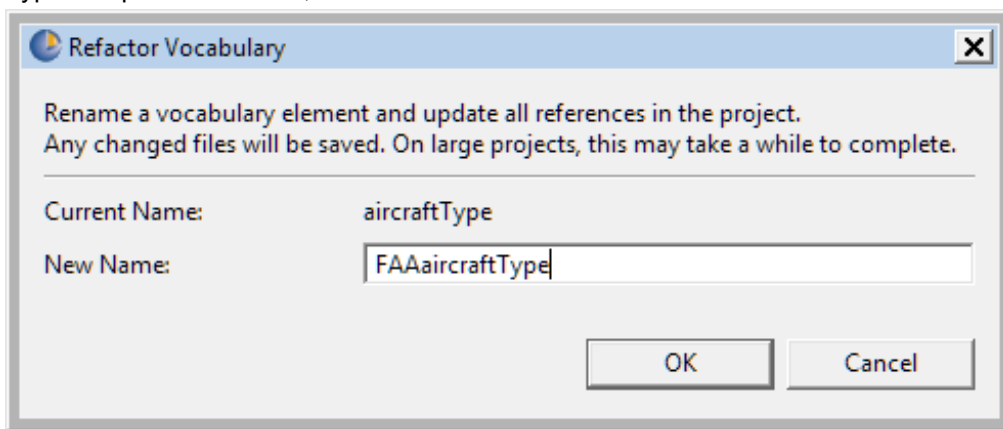
Note: Backup the project's rule assets before refactoring the Vocabulary since refactoring modifies and automatically saves rule assets.

To refactor an entity, attribute, or association name, do the following:

1. Open the project's Vocabulary.
2. Click on the name you want to change and highlight it.
3. Either right-click on the item to choose **Refactor** in the context menu, or choose the menu action **Vocabulary > Refactor**.
4. The **Refactor Vocabulary** dialog box opens for the selected item:



5. Type in a preferred name, and then click **OK**:



The change is applied throughout the project, and then all the changed files in the project are saved.

If **Refactor** determines the new name you have entered is currently used by another attribute in the entity, or currently used in the parent entity name, the name is disallowed and the **OK** button is disabled. When this occurs, type a new unused name in the **Refactor Vocabulary** dialog box and select **OK**.

Note: **CTRL-Z** or **Undo** options are unavailable. To revert a refactored name to the original value, use the **Refactor** feature again and change the name back.

Important: Allow the process to finish

Refactoring on a large project might take a minute or more. Although an automated process, refactoring is a non-trivial operation that requires parsing all dependent assets to properly perform the changes. The refactoring process first searches the assets and locates where entities, attributes, and associations are used. Then it replaces the old term with a new term at each location.

While processing, **Refactor** blocks use of the UI to prevent other edits to be performed. This avoids any possibility of the renaming operation undoing simultaneous edits made by users, or encountering other concurrency issues.

Also, any unsaved changes in other open editors are saved by the refactoring process.

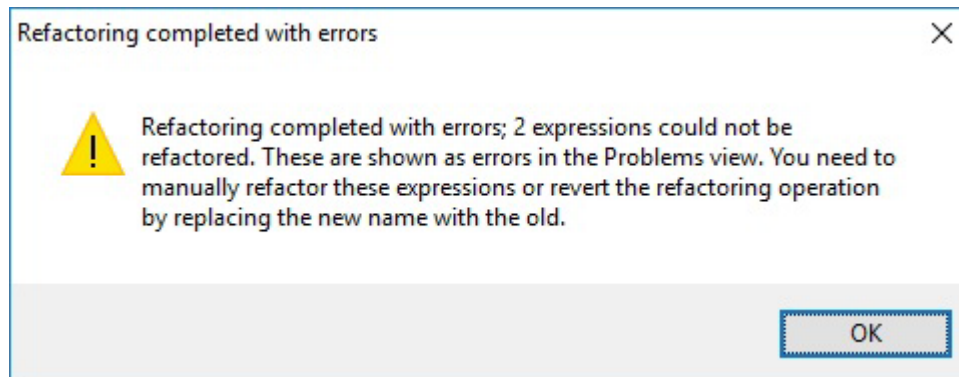
Note: Do not cancel the refactoring process once started, to do so could leave the assets in an invalid state as some of the files would be updated but not all of them.

Important: Expressions that cannot be refactored

Within a Rulesheet, there are a small number of expression types that cannot be refactored automatically:

- When two entities have an attribute with the same name and the expression assigns the value of one attribute to the other. For example: `A.name = B.name`
- When entities and attributes share the same name. For example: `Name.name`
- When an entity would be refactored to the same name as an existing alias in a Rulesheet. **Refactor** detects when this can occur, stops the operation, and displays a message recommending a new name be chosen for the entity, or a Rulesheet alias be changed.

If expressions that cannot be refactored are encountered, the **Refactor** operation still continues to refactor other expressions until the operation is finished. The expressions that could not be refactored are left unchanged and flagged as errors in the **Problems** view, and an alert is given as shown:



To update those expressions correctly, review the list of errors in the **Problems** view, and manually edit the Rulesheets accordingly.

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 assembled in a Ruleflow (see the [Ruleflow](#) chapter), packaged and deployed into a production environment.

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

- Double-clicking on any Rulesheet name in the **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.js 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.js Studio window causes the Studio menubar and toolbar to display the tools needed to begin working with rules.

For details, see the following topics:

- [How to create a Rulesheet](#)
- [Rulesheet menu commands](#)
- [Rulesheet toolbar](#)
- [Commands on the Rulesheet context-sensitive menu](#)

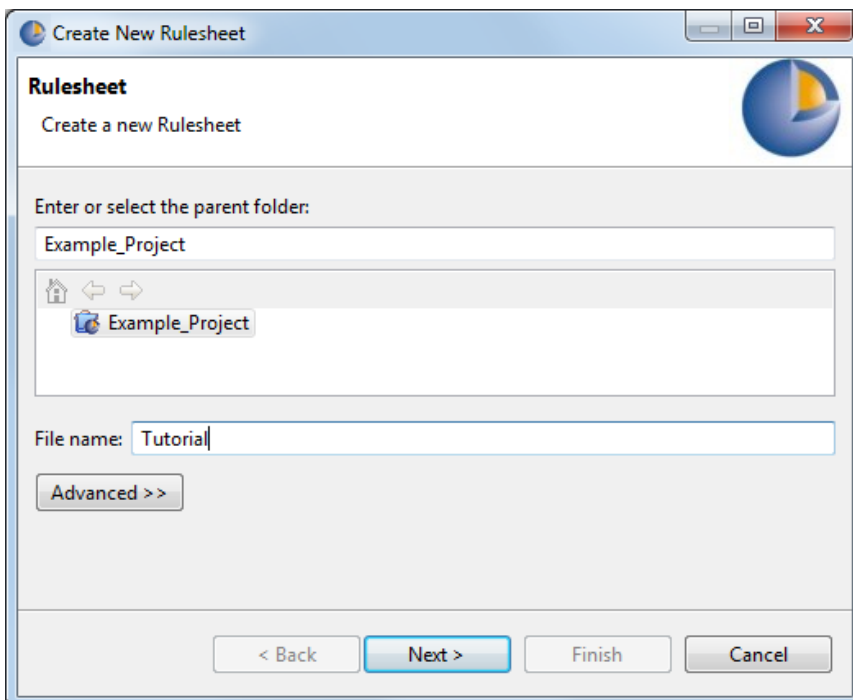
- [Rulesheet sections](#)
- [Navigation in a Rulesheet](#)
- [Rule statements window](#)
- [Rulesheet properties](#)
- [Use the business vocabulary to build rules](#)
- [Use Rule Operators to build rules](#)
- [How to name Rulesheets](#)
- [How to validate and optimize a Rulesheet](#)
- [How to create a Rulesheet report](#)
- [How to save a new Rulesheet](#)
- [How to save a modified Rulesheet](#)
- [How to close a Rulesheet](#)
- [How to delete Rulesheets](#)

How to create a Rulesheet

To create a Rulesheet:

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

The **Create New Rulesheet** dialog opens.

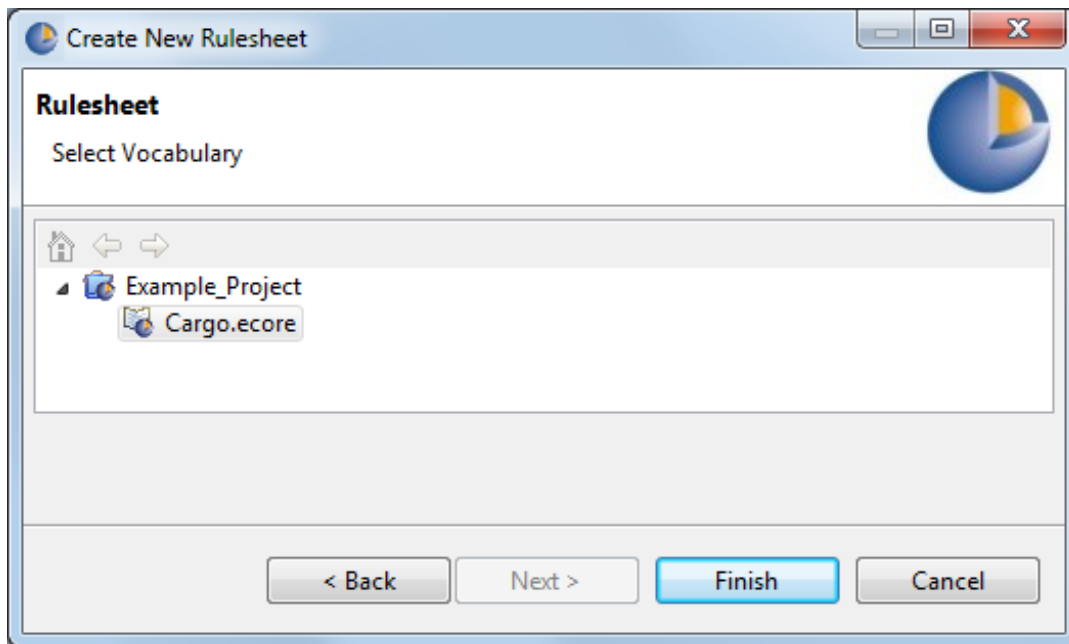


2. Highlight the Rule Project you would like to associate the new Rulesheet with in the list (or enter it manually in the **Enter or select parent folder** entry area.
3. Enter a file name for the Rulesheet in the **File name:** entry area.

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

4. Click **Next** to continue.

Figure 1: The Create New Rulesheet Wizard window



Note: The dialog only lists the Vocabularies for the parent project. This prevents you from creating cross-project assets. Do not create cross-project assets.

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

You must have a Vocabulary within a Rule Project to create a Rulesheet.

6. Click **Finish**.

Your new Rulesheet is now listed in the **Project Explorer** window and in a new Editor window with a tab of the given name and the `.ers` extension. The Rulesheet menubar and toolbar replace the Vocabulary menubar and toolbar while the Rulesheet is the active window. You can now model 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. The graph is saved as a labeled set of files at `[CORTICON_WORK]\Graphs`.
 - **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. The graph is saved as a labeled set of files at `[CORTICON_WORK]\Graphs`.
 - **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 in chaining" 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.
- **Simple View | Advanced View** - Toggles display of the Scope and Filters panels on the left side of the Rulesheet.
- **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" in the Rule Modeling Guide*.
 - **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*.







- **Report** - Creates an HTML report and launches your browser for viewing. See [Creating a Rulesheet Report](#).
- **Compare Rulesheets** - Analyzes a selected Rulesheet in the context of the one currently in the editor to report on differences.

Rulesheet toolbar

When a Rulesheet editor is active, its tools are added to the toolbar.

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.
-  - Switches the view of the active Rulesheet to **Natural Language View**. See the *Rule Modeling Guide* for more information about this feature.
-  - Presents the **Add Comment** dialog. The comments you enter and all associative information are captured and displayed in the **Comments View**. See the *Documenting Rule Assets* section 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.js 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.
Comments ...	Presents the Add Comment dialog. Any comments you enter and all associated information is captured and displayed in the Comments View . See the How to document rule assets on page 105 section for details of this feature.
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>"How to work with rules and filters in natural language"</i> in the <i>Rule Modeling Guide</i> .

Rulesheet sections

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

Rules

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

cargo.ers			
Conditions		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			

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.*

Nonconditional Rules

The first column is a special column reserved for *nonconditional* rules, often referred to as *action-only* rules.

AbsoluteValue.ers				
Conditions				0
a				
b				
Actions				<
Post Message(s)				
A	Entity1.decimal1 = Entity1.decimal2.absVal			✓
B				
Overrides				

Rule Statements				
Ref	ID	Post	Alias	Text
A0				decimal1 equals the absolute value of decimal2

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.

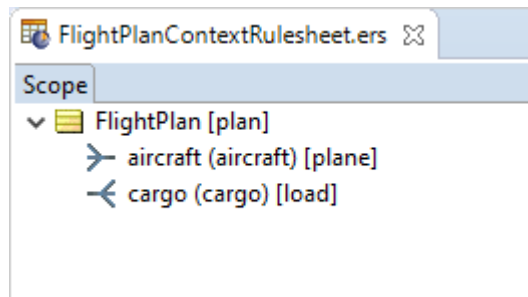
Logically, this means that all Actions modeled in Column 0 are unconditionally executed. In other words, no Conditions must be satisfied in order for these Actions to execute.

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

Scope and Filters in the Advanced View

When the **Advanced View** is selected, the Scope and Filter sections for the current Rulesheet are displayed.

Figure 2: Scope section, showing an Entity and its alias




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

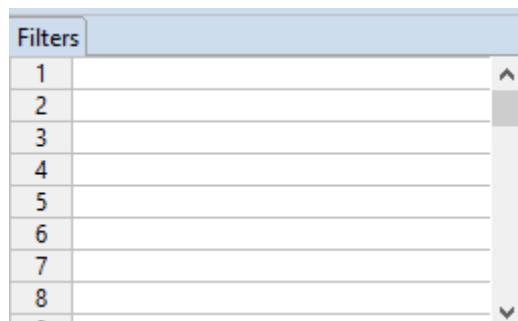
The Scope section can be populated *directly* by dragging and dropping Vocabulary elements into it, or *indirectly*, by dragging and dropping Vocabulary nodes onto other Rulesheet sections.

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.

Filters

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



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

Navigation in a Rulesheet

Mouse navigation

All general mouse functions are supported:

- Click on a cell to select it.
- Double-click on a cell to enter edit mode with the cursor at the location you clicked in the line.
- Click to open a selected cell's dropdown menu, and then click on the preferred value in the list.
- Use the scroll wheel to move up and down the rows in the selected section.

Keyboard navigation

The Rulesheet window enables navigation through several common keyboard shortcuts in its *cell-style* sections -- all the sections except Scope:

- Conditions and their cells
- Actions and their cells
- Filters
- Rule Statements and their columns

Keyboard shortcuts on a Rulesheet:

- Move up: Up-arrow
- Move down: Down-arrow, or Enter
- Move right: Right-arrow, or Tab
- Move left: Left-arrow, or Shift+Tab
- Edit content of the cell in focus: Backspace (places the cursor at end of entry)
- Clear content of the cell in focus, and then add content: Spacebar
- Beginning of line in edit mode: Home
- End of line in edit mode: End

Note: Nonconditional actions - In a column 0 value on an Action line, Backspace and Spacebar both toggle the current selection.

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:

Rule Statements		Rule Messages		
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.js Studio Ruletest, Rule Messages with Info severity are color-coded **green**, Warnings **yellow**, and Violations **red**.

Alias

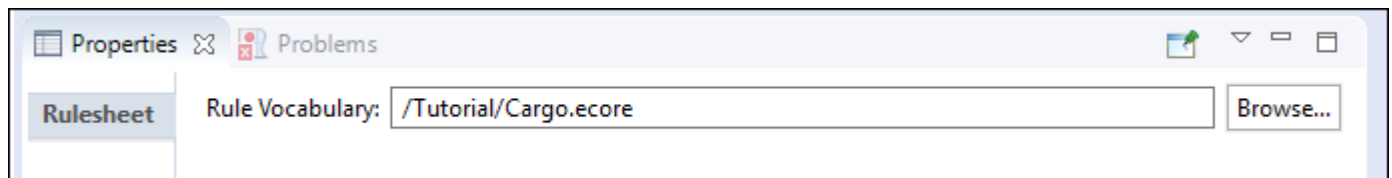
This field is mandatory if you want the Rule Statement posted during Rulesheet execution. All posted Rule Messages must be “attached” or “linked” to a Vocabulary entity. The choice of entity to “post to” is usually based on the entity being tested or acted upon in the associated rule.

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.


Rulesheet properties

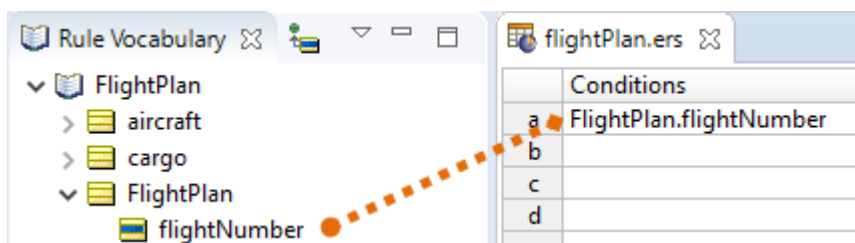
When you are editing a Rulesheet, the **Properties** tab's **Rulesheet** displays the Rule Vocabulary that supports this Rulesheet.



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.

Use 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*.

Use Rule Operators to build rules

Corticon.js Studio's built-in library of operators is located in the **Rule Operators** window in the lower left-hand corner of the Corticon.js Studio window.

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

1. In the Rule Operators windows, expand the tree to locate the operator you want to use.
2. Select the operator you want. You can hover over the operator to display its syntax, data type returned, and description.
3. Either drag and drop the operator onto the Rulesheet, or simply type its text into your expression.

How to name 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 16 section of this document.

How to validate and optimize 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.js Studio Tutorial: Basic Rule Modeling*, and in more depth in the *Corticon.js Studio: Rule Modeling Guide*.

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

How to create a Rulesheet report

1. With the Rulesheet you want to report open as the current window, choose the menu command **Rulesheet > Report**.

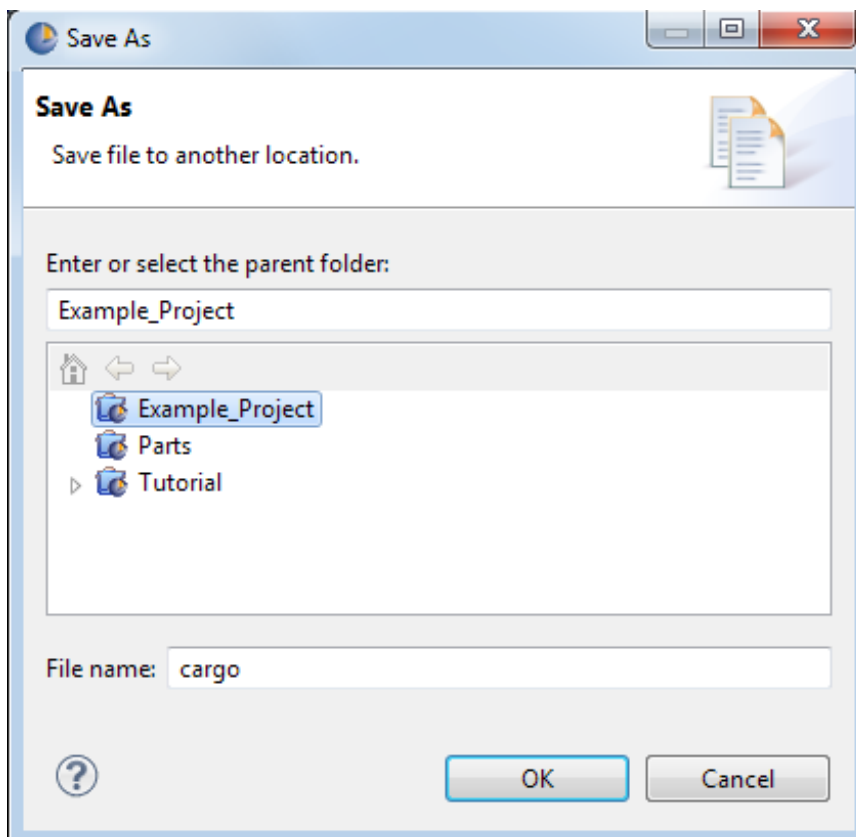
The **Generate Report** dialog opens.

2. Choose the **Report Type**, **Report Style**, and **Output Folder** for this report. Report types focus on either Natural Language or Expressions in the Rulesheet.
3. Click **Finish**.
4. The Rulesheet report opens as a new HTML page in your default web browser, and then saves the HTML, XML, and CSS files in the specified output folder.

The default CSS and XSLT files are samples that you can customize to suit your preferences.

How to save 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 16.

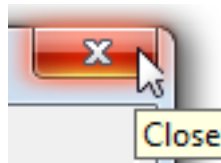


How to save a modified Rulesheet

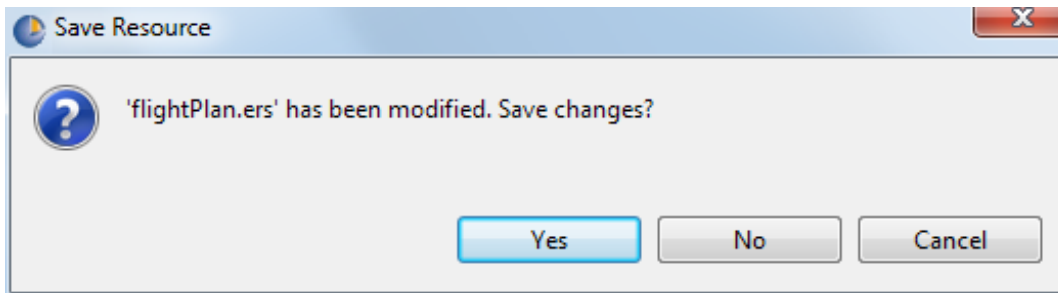
When you save a modified Rulesheet, the Corticon.js Studio automatically saves it to the current Rulesheet name. To save the Rulesheet to another name:

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

How to close 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.

How to delete 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.

Ruleflows

A Ruleflow is how Corticon assembles and organizes the components of a set of rules into a single unit. It is presented as a flow diagram, where Rulesheets, Service Callouts, and even other Ruleflows depict the flow of the rule, allowing it to iterate and to branch.

Once a Ruleflow is defined, it can be tested in Studio, and it can be compiled into a Decision Service for deployment in production environments (as described in the *Deployment Guide*).

Important: Corticon's Ruletester lets you test individual Rulesheets as well as Ruleflows in the Studio. In order to deploy a Rulesheet to it must be part of a Ruleflow. .

The Rulesheets components and the Ruleflow must be all using the same Vocabulary file. Ruleflow files have the extension `.erf`.

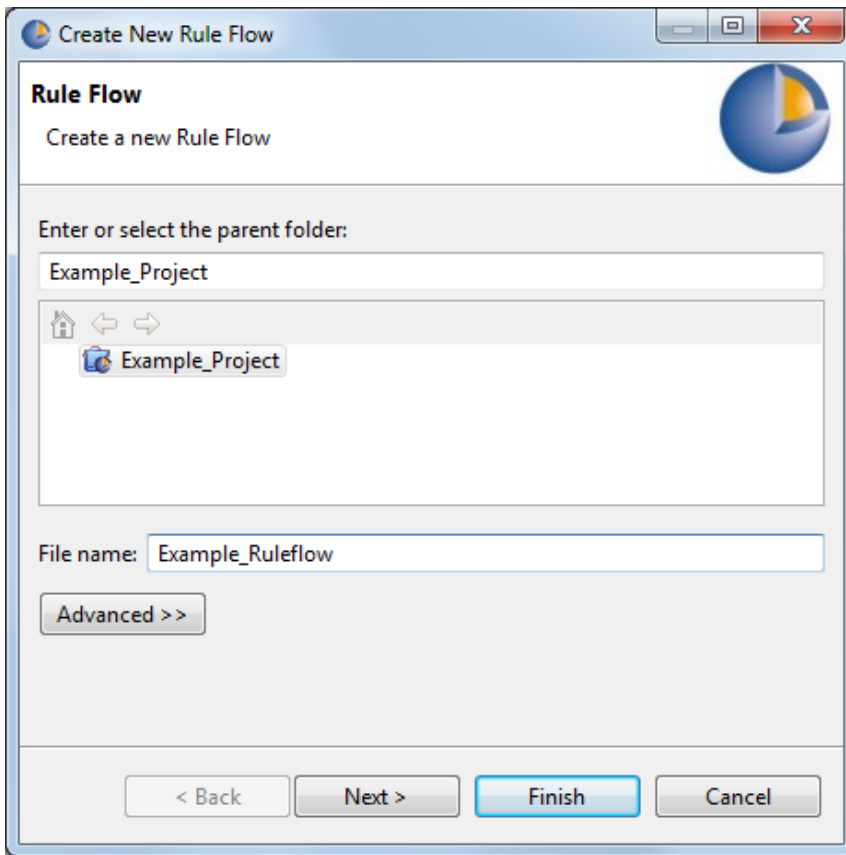
For details, see the following topics:

- [How to create a Ruleflow](#)
- [Ruleflow window](#)
- [Ruleflow properties](#)
- [Ruleflow preferences](#)

How to create a Ruleflow

To create a Ruleflow:

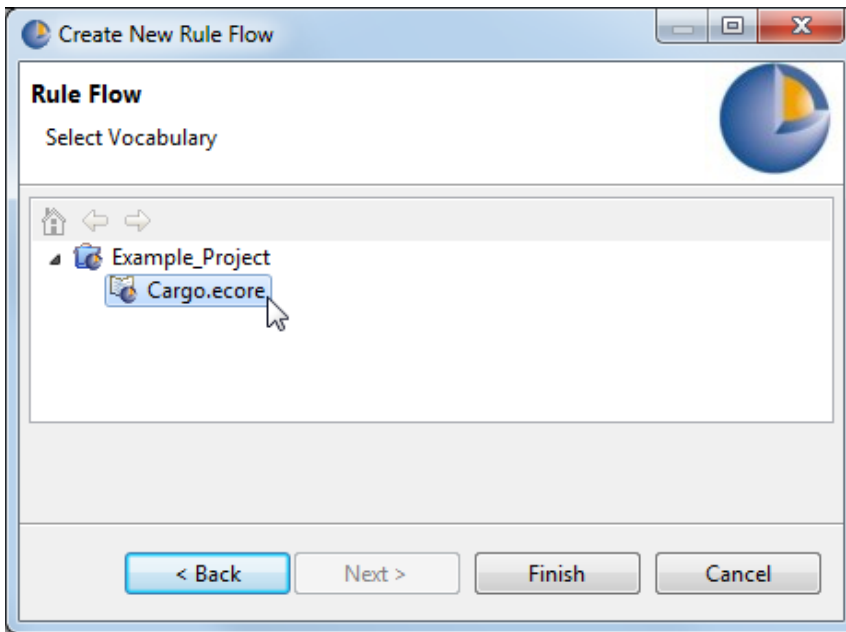
1. Choose **File > New > Ruleflow** from the menubar or click the down arrow next to the **New** icon on the toolbar and select **Ruleflow**. Either method will launch the **Create a New Ruleflow** wizard.



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.js and should not be used.

4. Click **Next** to continue.



Note: The dialog only lists the Vocabularies for the parent project. This prevents you from creating cross-project assets. Do not create cross-project assets.

5. Select the Vocabulary with which you want to associate the Ruleflow. A Ruleflow can only be associated with one Vocabulary. We expanded the `Example_Project` folder to highlight the `Cargo.ecore` file, and then click **Finish**.
6. Your new Ruleflow is added in the **Project Explorer** window with the `.erf` extension, and in a new active window with a tab of the given name. The Ruleflow menubar and toolbar are displayed instead of the Vocabulary menubar and toolbar.

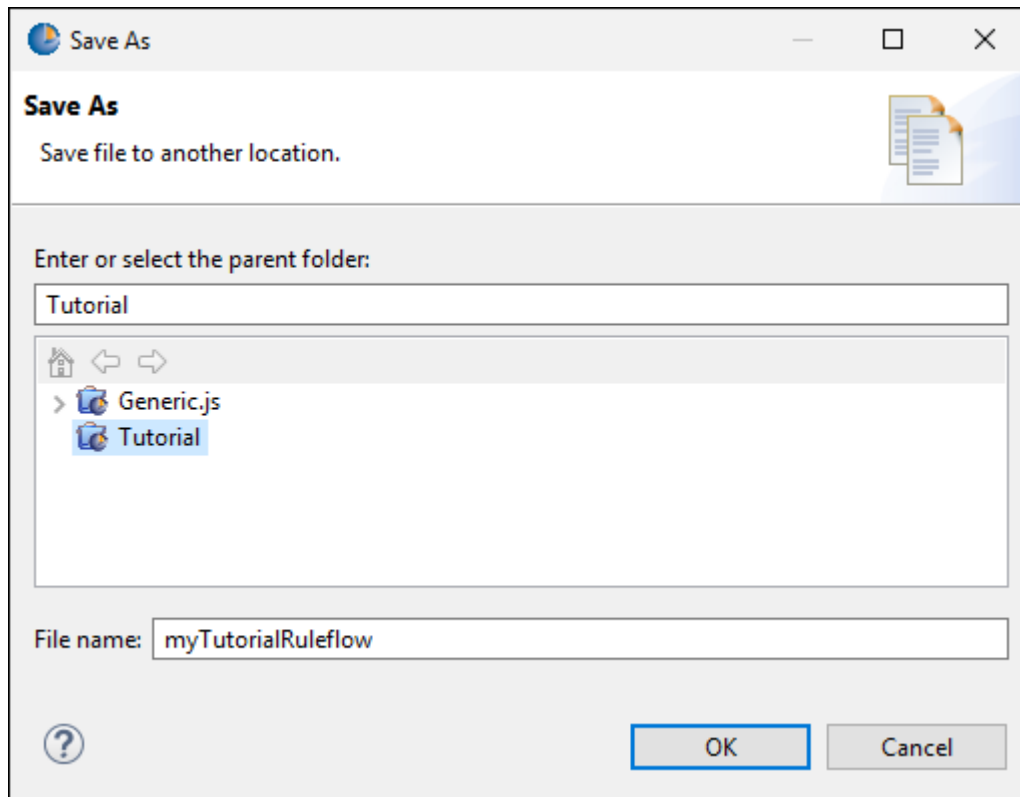
You are now ready to begin assembling the components of your Ruleflow.

How to name Ruleflows

Ruleflow names must adhere to and comply with the guidelines shown in the [File naming restrictions](#) on page 16 section of this document.

How to rename a Ruleflow or save 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 16 section of this document.

How to create a Ruleflow report

1. With the Ruleflow you want to report open as the current window, choose the menu command **Ruleflow > Report**.
2. Choose the **Report Type**, **Report Style**, and **Output Folder** for this report. Report types focus on either Natural Language or Expressions in the Ruleflow's Rulesheets.
3. Click **Finish**.
4. The Ruleflow report opens as a new HTML page in your default web browser, and then saves the HTML, XML, and CSS files in the specified output folder.

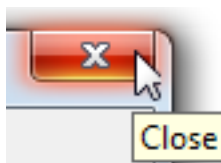
The default CSS and XSLT files are samples that you can customize to suit your preferences.

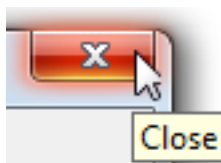
How to save a new Ruleflow

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

Refer to the [Ruleflow menu commands](#) on page 65 topic for details regarding saving, or renaming, a Ruleflow with a different name or to a different file location.

How to close a Ruleflow



1. To close a Ruleflow, click its **Close** button, , or select the menu command **File > Close**.
2. Choose **Yes** to save any changes you made.

How to delete 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.

Ruleflow window

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

Ruleflow menu commands

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

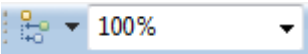
The **Ruleflow** menu has the following commands:

- **Comment** - Opens the Add Comment window. When no object is selected, the comment is a file type for the entire Ruleflow. When one object on the canvas has been selected, the comment applies as one entry to that object and its type. To review comments on this asset and its components, see [Use the Comments view](#) on page 109.
- **Properties** - Opens the Ruleflow properties window.
- **Dependency graph** - Generates attribute and logical dependency graphs based on selected nodes in the active Ruleflow. For more information, see *"How to generate Ruleflow dependency graphs" in the Rule Modeling Guide*.

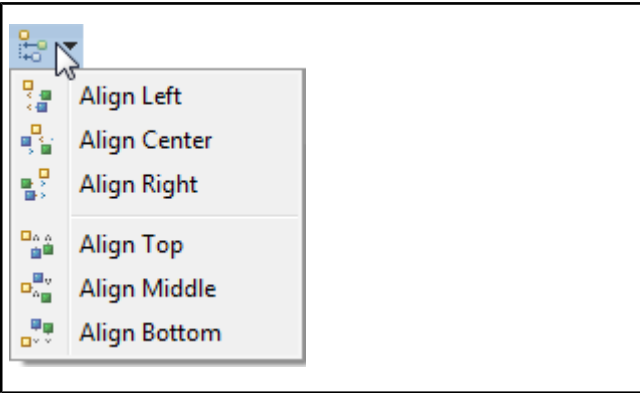

- **Report** - Creates an HTML report and launches your browser for viewing. See [How to create a Ruleflow Report](#).
- **Service Contract** - Creates a CSV file that provides guidelines for creating a service contract for the Ruleflow. See *"Service contract output" in the Deployment Guide*.

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.

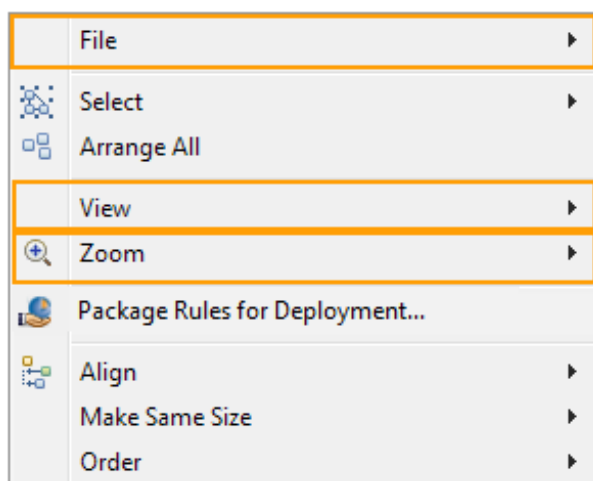
	Aligns multiple selected objects on the Ruleflow canvas.
	Zooms the Ruleflow window to the specified magnification.

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.

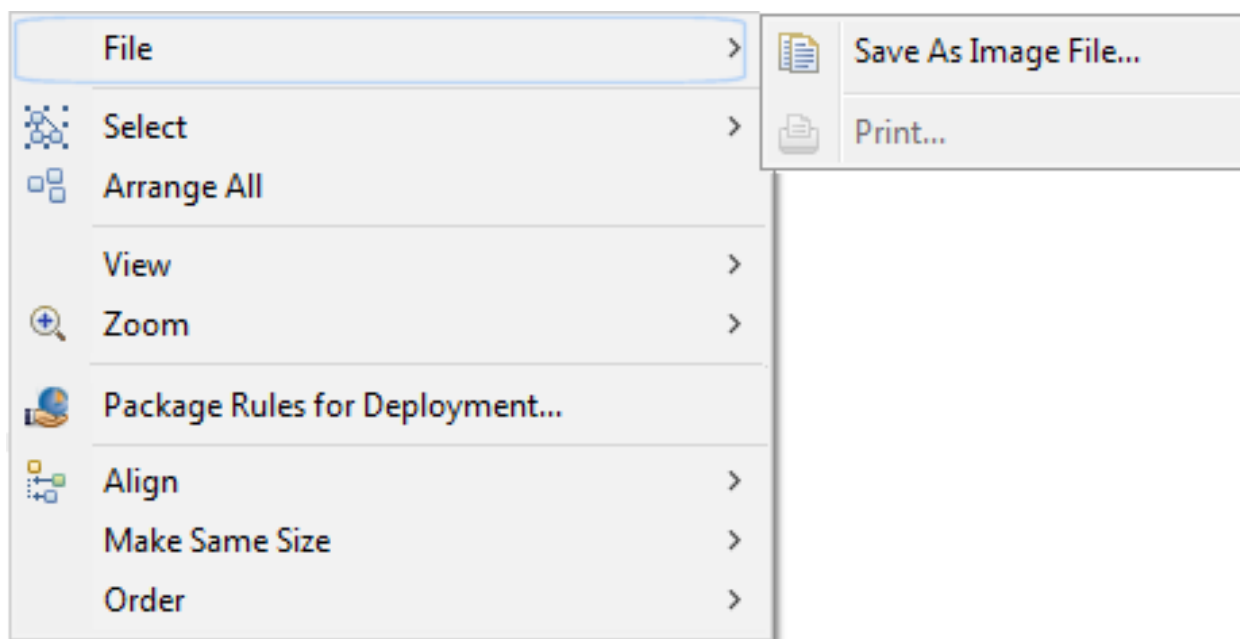
Note: You might see a menu item **Remove from Context**. It is a standard operator yet not used in the Ruleflow editor.

See [Object commands on the Ruleflow context-sensitive menu](#) on page 76 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



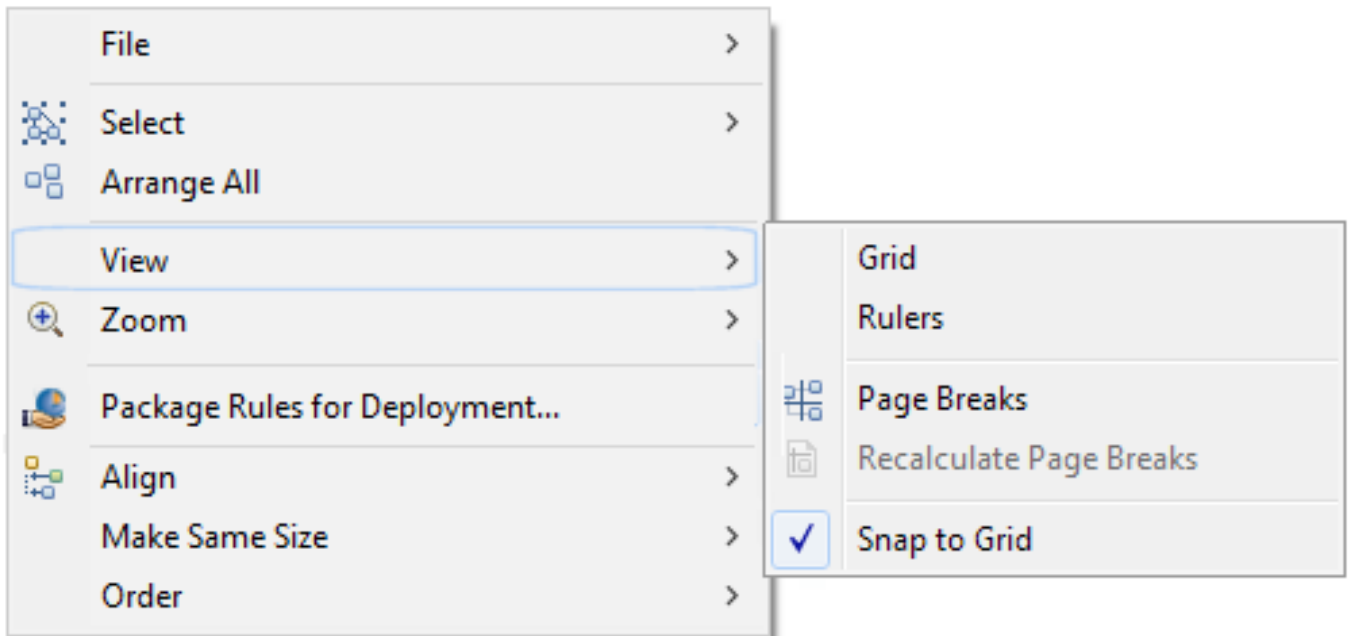
Select

See [Object commands on the Ruleflow context-sensitive menu](#) on page 76.

Arrange All

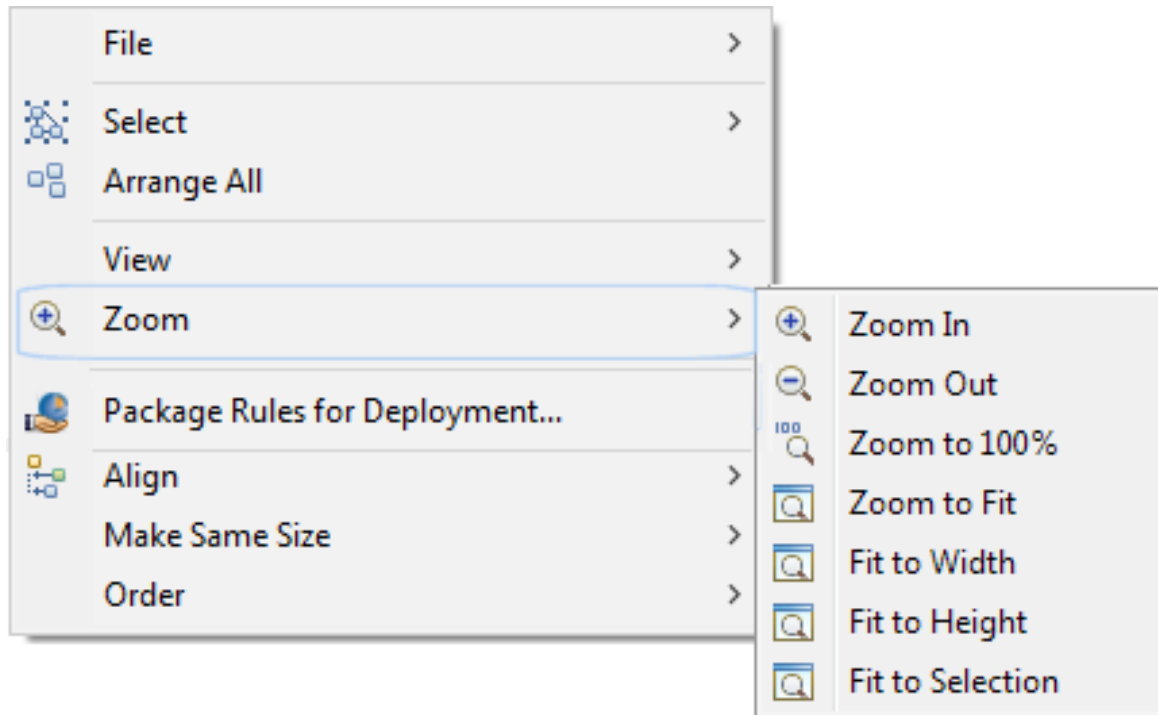
See [Object commands on the Ruleflow context-sensitive menu](#) on page 76.

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



Package Rules for Deployment

Lets you move the Ruleflow immediately to packaging as a Decision Service. Forces you to save all changes, then opens the **Package Rules for Deployment** dialog.

Align

See [Object commands on the Ruleflow context-sensitive menu](#) on page 76.

Make Same Size

See [Object commands on the Ruleflow context-sensitive menu](#) on page 76.

Order

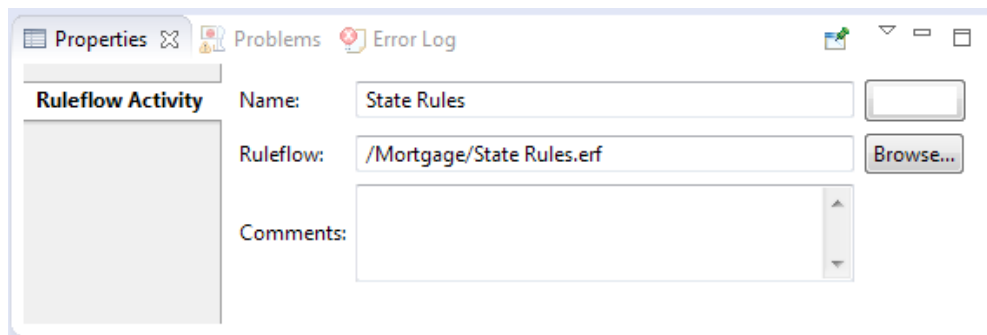
See [Object commands on the Ruleflow context-sensitive menu](#) on page 76.

Ruleflow properties

When a Ruleflow is the active file, its properties display in the **Properties** tab at the bottom of the Corticon.js 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. None of these actions change the Ruleflow properties of the original Ruleflow.

Ruleflow Activity

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.

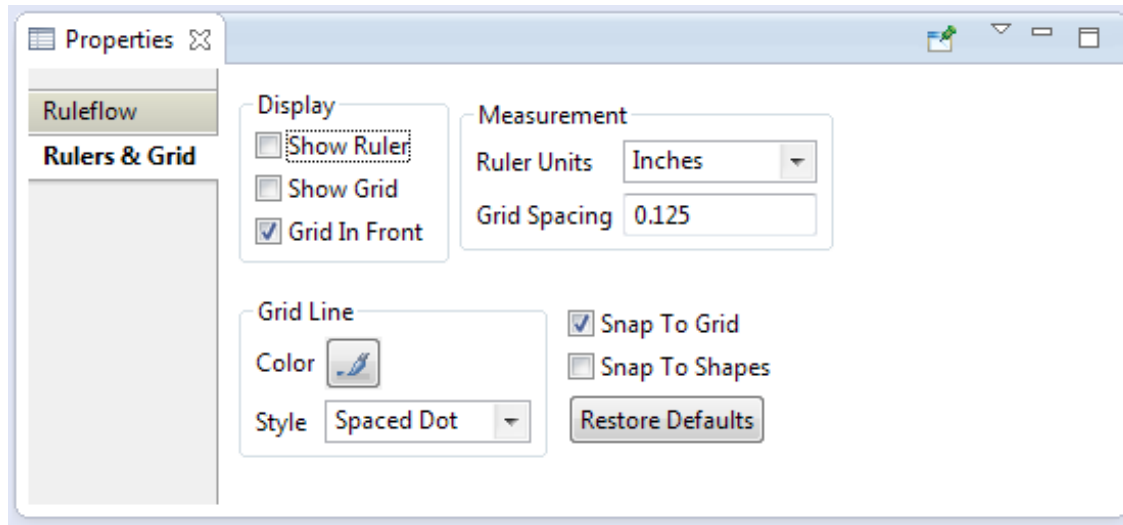
Total Number of Rules

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.

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 3: The Ruleflow Rulers and Grid Sub-tab



Settings for Ruleflow graph fonts

The following settings that you specify in the `brms.properties` file impact all the graphs produced from Ruleflows on the local machine:

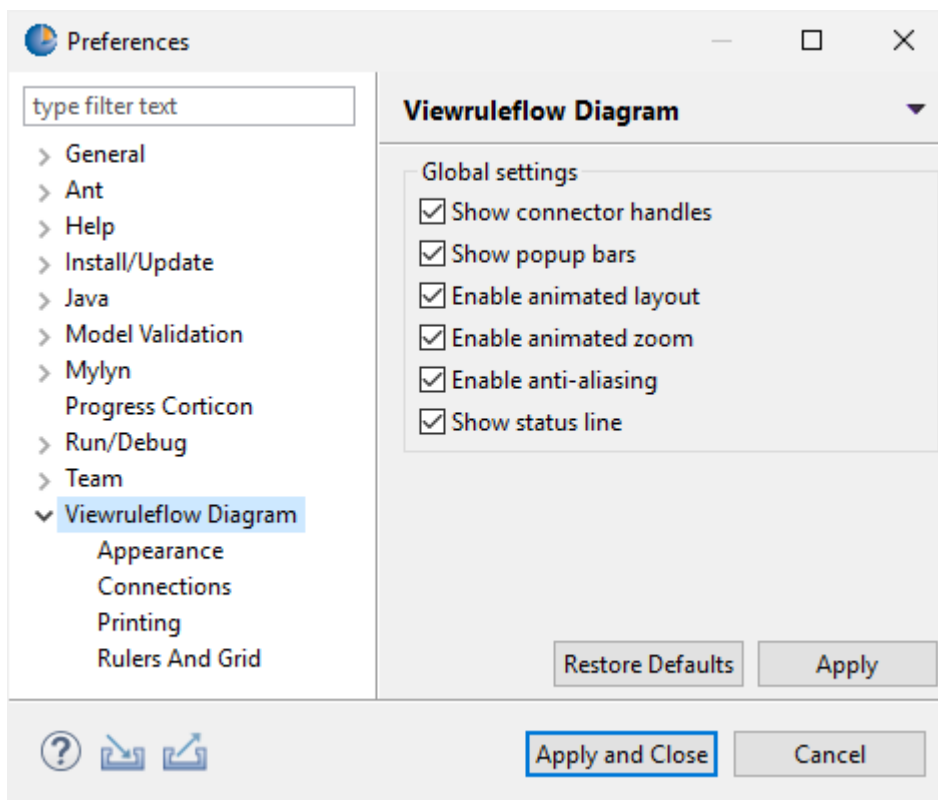
```
These properties set the font type and size used by the graph visualizer
com.corticon.crml.CrmlGraphVisualizer.fontname=Helvetica-Narrow.ttc
com.corticon.crml.CrmlGraphVisualizer.fontname.ja=msgothic.ttc
com.corticon.crml.CrmlGraphVisualizer.fontsize=9
```

Ruleflow preferences

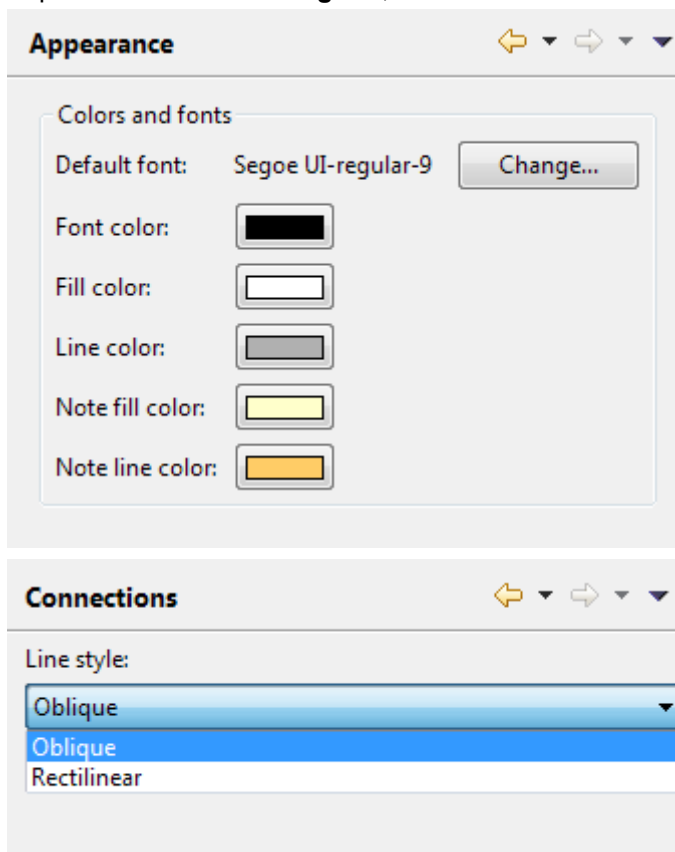
You can set many aspects of a the Ruleflow canvas and its output.

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:



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



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

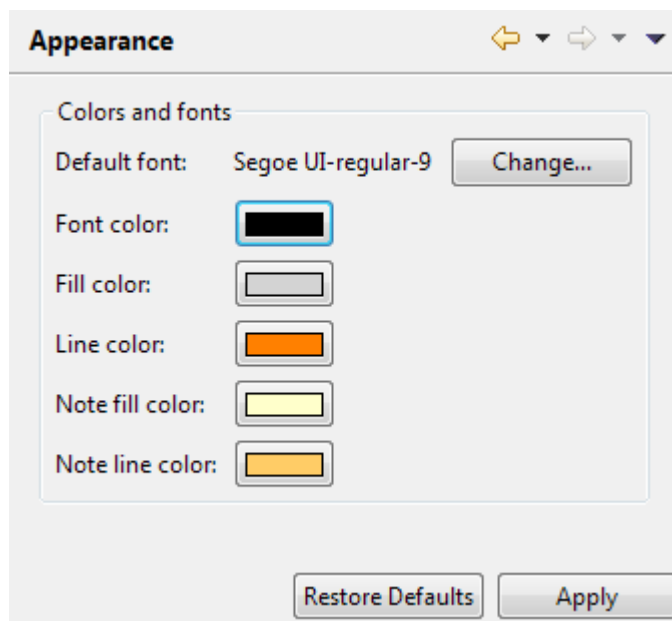
☐ Snap to shapes for new diagrams

Grid spacing (in inches): 0.125

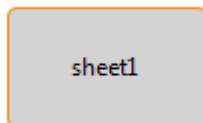
4. Adjust the settings to suit your needs, click **Apply**, and then click **OK**.

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 [How to add colors to Ruleflow objects](#) on page 81. 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.js 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.js 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.

Branch

Branching lets you choose 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, or a Subflow) that will act on it, as well as the subsequent objects that are connected to the branch. Branches are powerful processing structures of Corticon.js assets, yet are not in themselves Corticon.js assets.

Subflows

Subflows provide a way to group multiple Corticon.js 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.js assets used on a canvas, but are not Corticon.js assets.

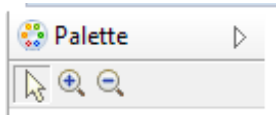
Note: While you can have multiple subflows in one Ruleflow, re-using a subflow name that has different Rulesheets and connections will evaluate the last one compiled ("farthest downstream") and use that for all instances.

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](#)
- [How to add colors to Ruleflow objects](#)

Ruleflow canvas tools

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

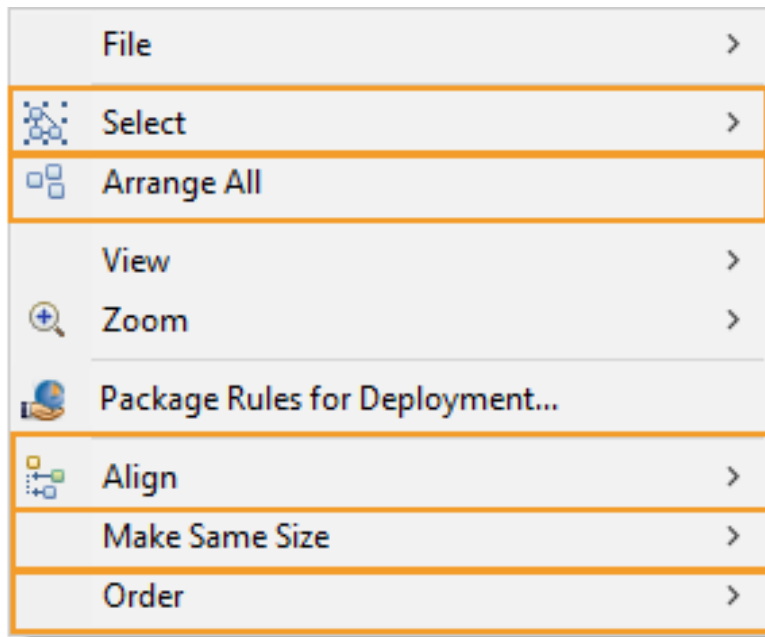


- **Select** – The default tool, the arrow cursor lets you select objects and manipulate one or more objects on the canvas.
- **Zoom** – The (+) button zooms in on the canvas; the (-) button zooms out from the canvas.

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 66 for the Ruleflow file-level commands.

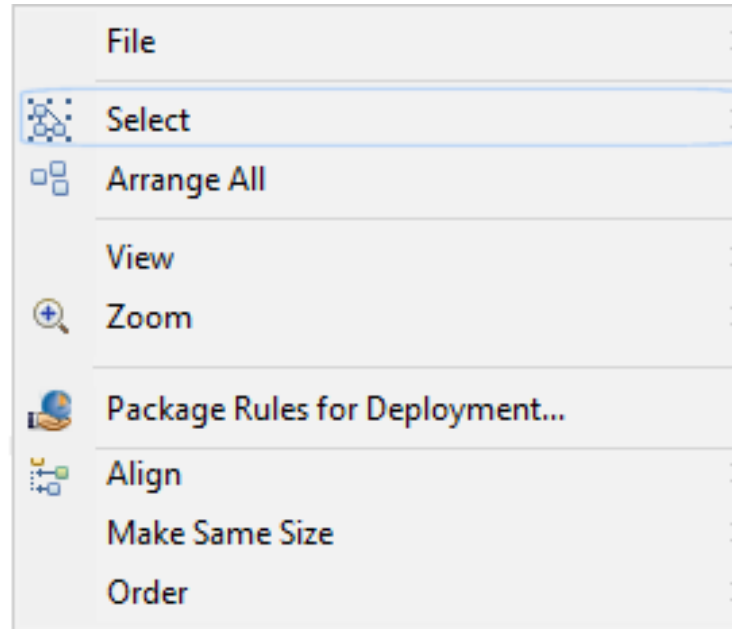


The five sections highlighted above are the object-related menu commands on the Ruleflow pop-up menu.

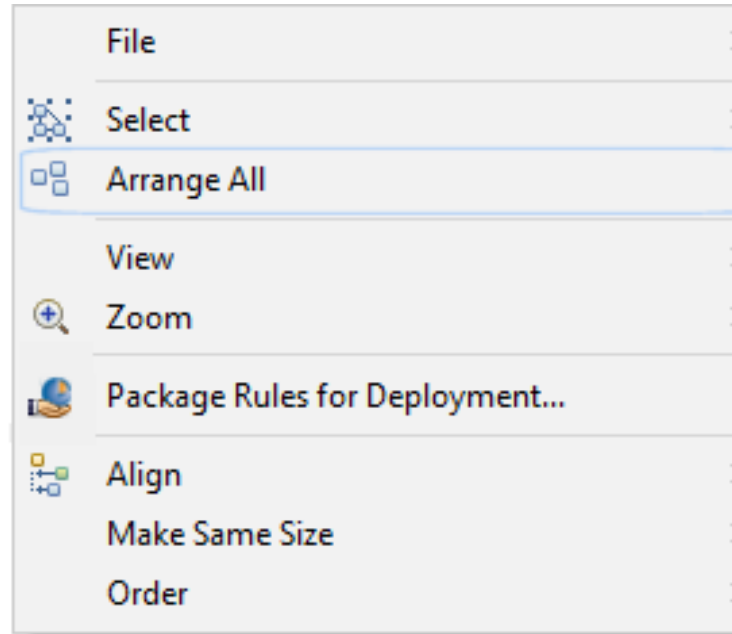
File

See [Editor commands on the Ruleflow context-sensitive menu](#) on page 66.

Select



Arrange All



View

See [Editor commands on the Ruleflow context-sensitive menu](#) on page 66.

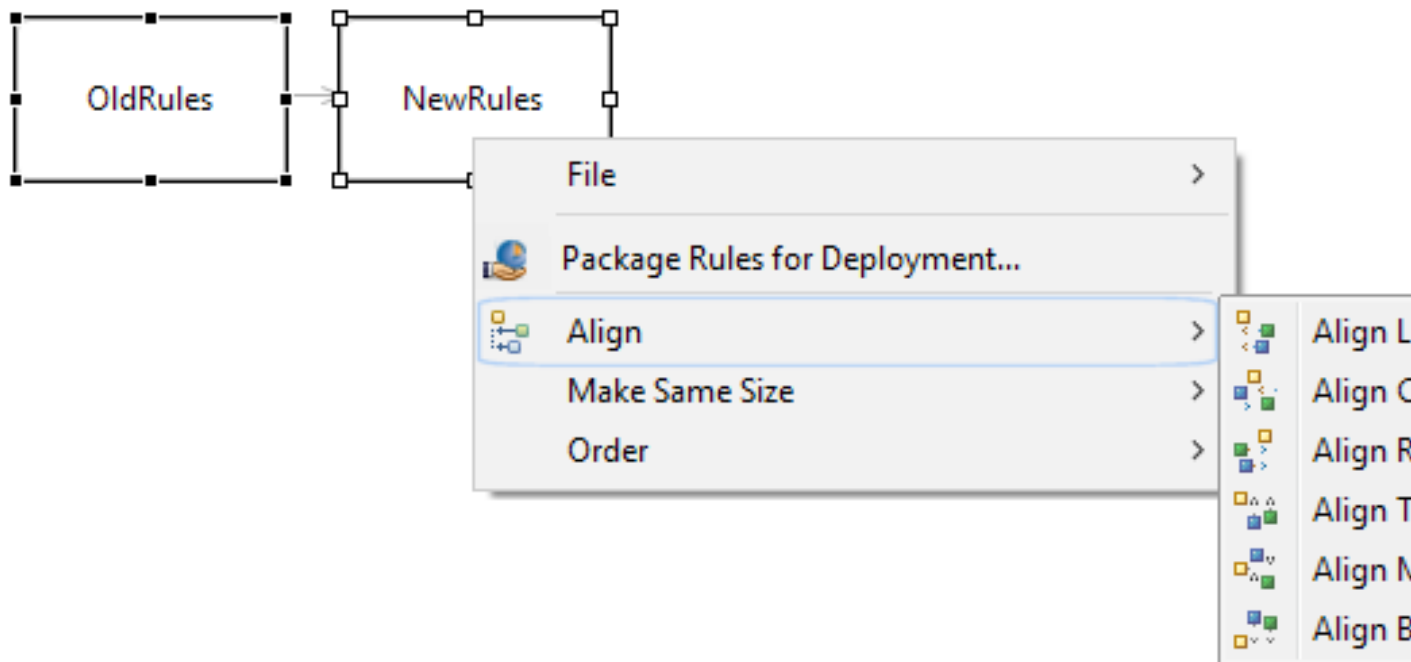
Zoom

See [Editor commands on the Ruleflow context-sensitive menu](#) on page 66.

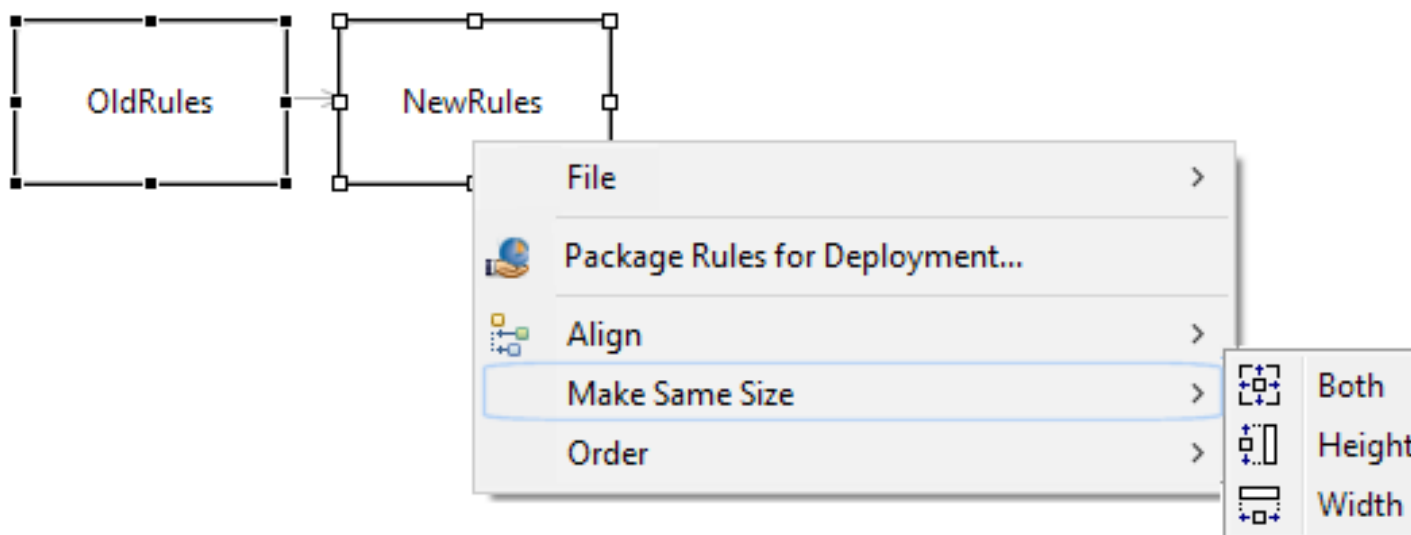
Package Rules for Deployment

Lets you move the Ruleflow immediately to packaging as a Decision Service. Forces you to save all changes, then opens the **Package Rules for Deployment** dialog.

Align

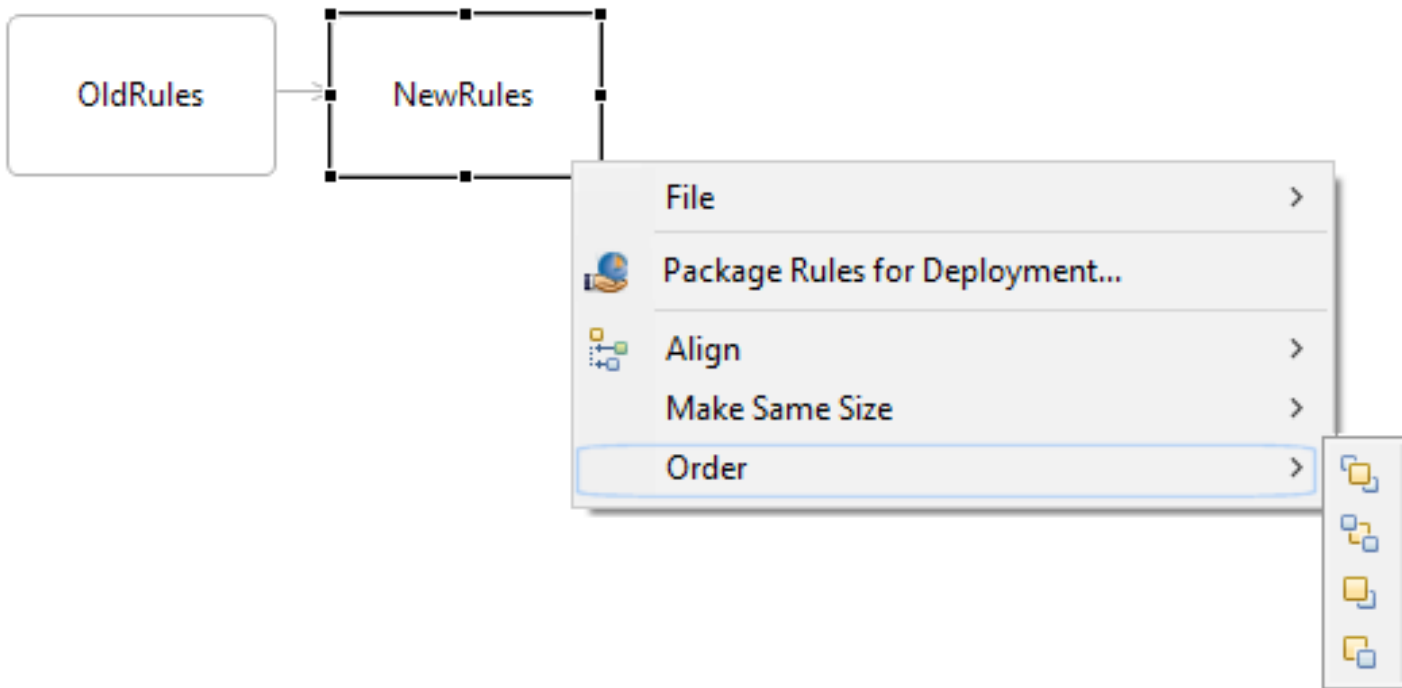


Make Same Size



- **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

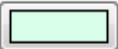
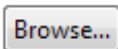


Properties of Ruleflow objects on a Ruleflow canvas

While the Ruleflow that manages the entire file and the current canvas has specific properties (see [Ruleflow properties](#) on page 69), 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. To change the name on the canvas, double-click the instance on the canvas and re-name it.
-  - Color lets you choose from a color palette to override the default color that fills the object. See [How to add colors to Ruleflow objects](#) on page 81 for more information. You could also use preferred general colors for all aspects of canvas objects, as discussed in [Ruleflow preferences](#) on page 71
-  - For objects that are Corticon.js assets, lets you navigate to a different file of the object type that uses the same Vocabulary.

Object specific properties

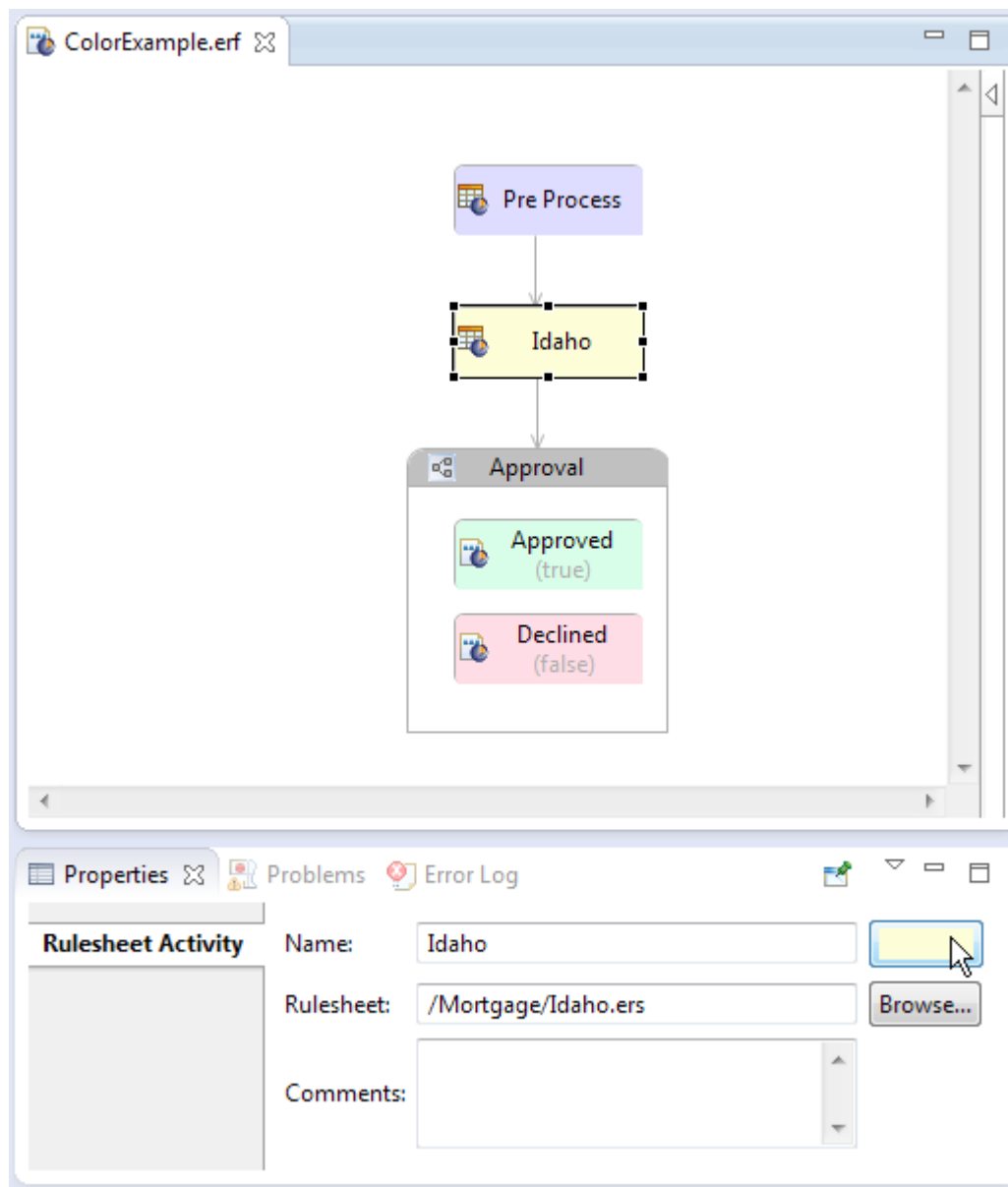
- **Rulesheet Activity - Rulesheet** is a valid `.ers` file in the project. You can select a different file but you cannot change the path or the name of the original file.
- **Ruleflow Activity - Ruleflow** is a valid `.erf` file in the project. You can select a different file but you cannot change the path or the name of the original file.

- **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 section "Conditional branching in Ruleflows" in the Rule Modeling Guide.

Note: Comments on a Ruleflow and on Ruleflow Objects - To add a comment to the Ruleflow or to the object in focus on the canvas, choose the Ruleflow menu command **Comment**. To review comments on this asset and its components, see [Use the Comments view](#) on page 109.

How to add colors to 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 71. When an individual object's color reverts to its default value, white, the underlying preference color is used.

Ruletests

A Ruletest is a Corticon.js 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.

Note: Ruletest settings - Several properties control Ruletest behaviors and limits. For more information, see *"Studio properties and settings" in the Rule Modeling Guide*.

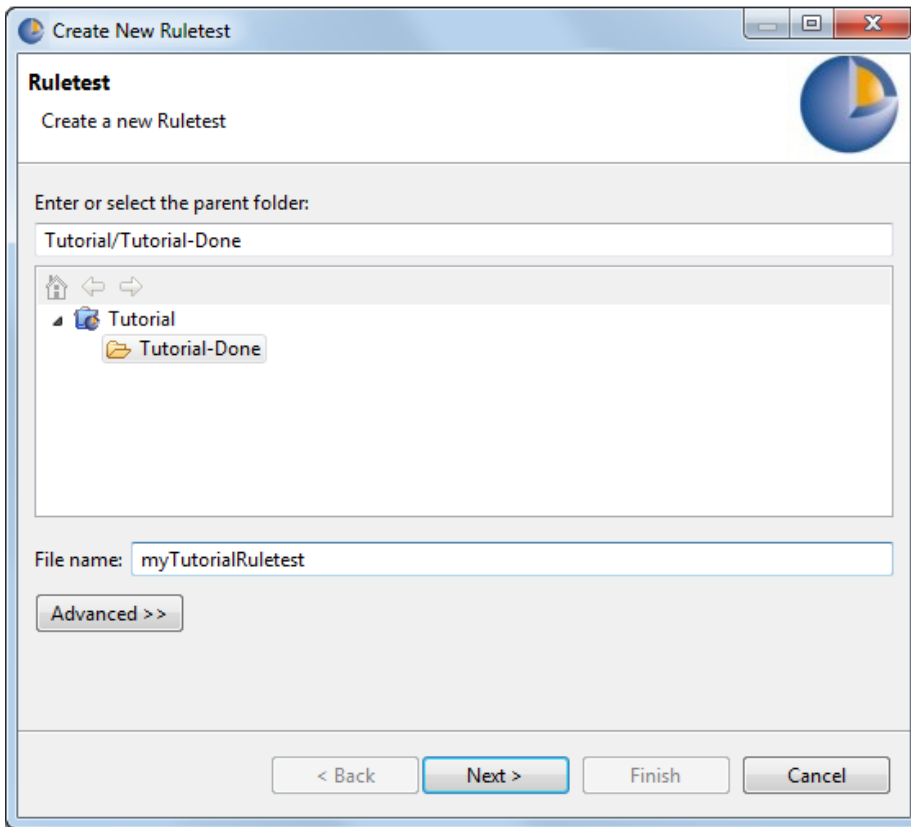
For details, see the following topics:

- [How to create a Ruletest](#)
- [Ruletest window](#)

How to create a Ruletest

To create a new Ruletest:

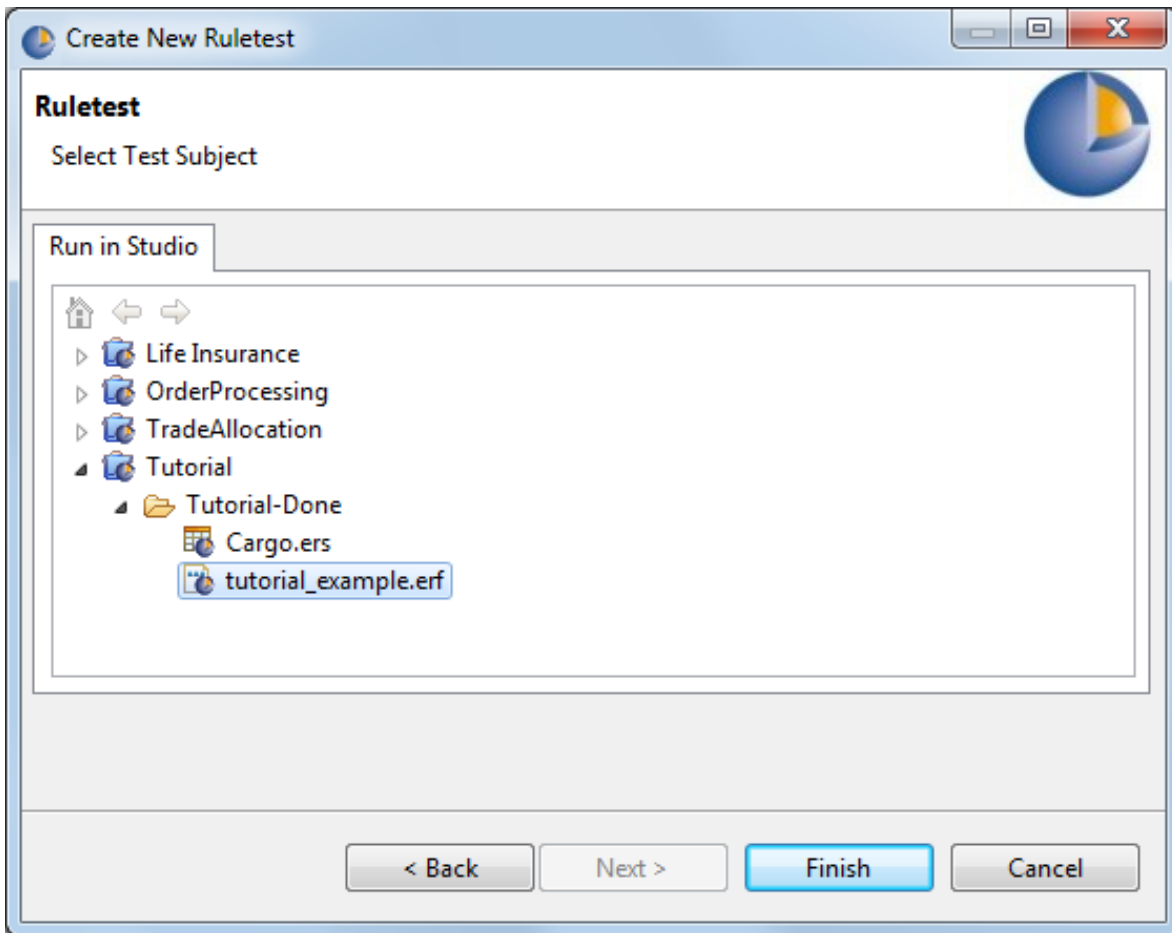
1. Choose **File > New > Ruletest** from the menubar or click the down arrow next to the **New** icon on the toolbar and select **Ruletest**. Either method will launch the **Create a New Ruletest** wizard.



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.js and should not be used.

4. You can simply click **Finish** to accept the first Rulesheet or Ruleflow in the folder as the test subject. If you click **Next**, the **Select Test Subject** panel opens, as shown:



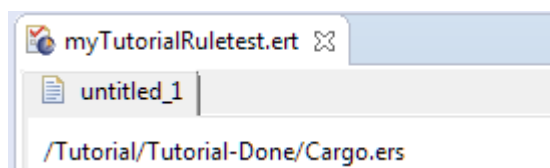
Choose the Rulesheet or Ruleflow file in the project that you want as the test subject, and then click **Finish**

Note: The dialog lists only the Rulesheet and Ruleflows in the parent project to ensure that you do not attempt to create cross-project assets.

You can later change a testsheet's test subject to run against another Rulesheet or Ruleflow file in the same project file in Corticon.js Studio.

Note: You can also directly edit the test subject path on the testsheet. This feature, typically reserved for advanced users, will reject an invalid file but might not catch all the subtleties of the validation in the **Select Test Subject** dialog.

The Ruletest file opens in its editor with an initial Testsheet tab, `untitled_1` assigned to the selected test subject, as illustrated:

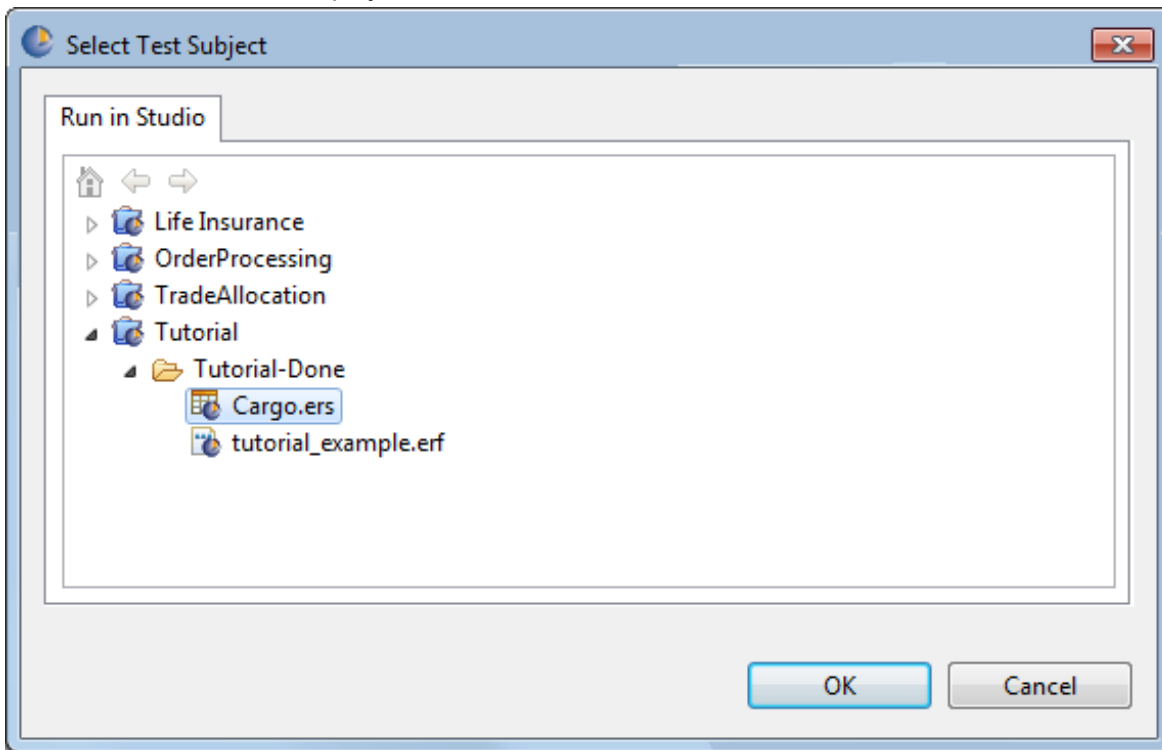


You can double-click on the Testsheet's tab to rename it.

Note: For more about Testsheet tabs, see [How to add testsheets](#) on page 98 and [How to rename testsheets](#) on page 99

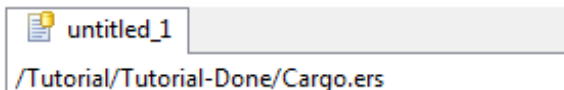
Choose a test subject in the Studio workspace

1. In the Ruletest you want to test, open it in its editor, and then choose the menu command **Ruletest > Testsheet > Change Test Subject**. The **Select Test Subject** dialog box opens on its **Run in Studio** tab. Notice that the view of the projects in the current workspace shows only Rulesheets and Ruleflows. Choose one of the files in the same project as the Ruletest to select it, as shown:



2. Click **OK**.

The dialog closes, and the selected file is listed at the top of the testsheet, as shown:



Ruletest window

A Ruletest is the mechanism within Corticon.js 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.js 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.js Studio Ruletest. In order to deploy and test using , Rulesheets must be packaged as Ruleflows before they can be executed. Ruleflows may be tested both in Corticon.js 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.

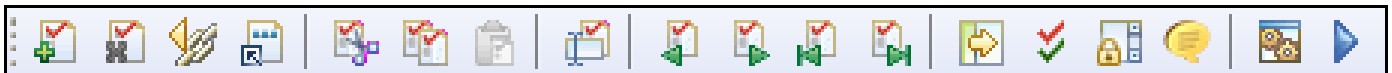
The **Ruletest** menu has the following items:

- **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 date from the Output panel of the first.
 - **Change Test Subject** - Opens a window that lets you select a new Rulesheet or Ruleflow to test in Corticon.js Studio.
 - **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.
 - **Add Comments to Testsheet** - Opens the **Add Comment** window to add and display a comment of up to 200 characters to the one selected item in the Input column. When no item is selected, the comment applies to the Testsheet – it can be of indeterminate length, and will be added to a Ruleflow report.
- **Import**
 - **JSON** - Import a valid CorticonRequest JSON document into Corticon.js 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 JSON** - Exports the active Testsheet's Input pane as a JSON document you name.
 - **Exclude Transients** - Ignores attribute values that are transient data mode in output file. Toggling this setting under one tree toggles it for Input, Output, and Expected.
- **Output**






- **Export Response JSON** - Exports the active Testsheet's Output pane as a JSON document you name.
- **Exclude Transients** - Ignores attribute values that are transient data mode in tests. Toggling this setting under one tree toggles it for Input, Output, and Expected.
- **Copy to Expected** - Copies the data in the Output panel to the Expected panel.
- **Expected**
 - **Export Response JSON** - Exports the active Testsheet's Expected pane as a `CorticonResponse` JSON document.
 - **Exclude Transients** - Ignores attribute values that are transient data mode in tests. Toggling this setting under one tree toggles it for Input, Output, and Expected.
- **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 *"Review test results when using the Expected panel" in the Rule Modeling Guide*.
- **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](#).
- **Remove Invalid Nodes** - Discards all invalid nodes in Input, Output, and Expected columns in all Testsheets in the Ruletest.














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**
-  **Add Comment**
-  **Deploy**
-  **Run Test**

Commands on a Testsheet context-sensitive menu

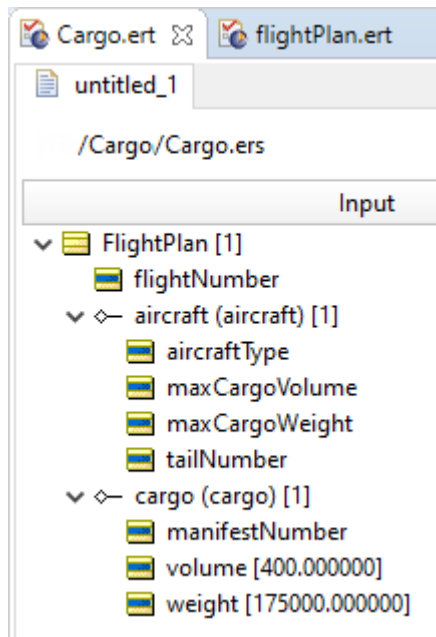
When you right-click in a Testsheet, the Ruletest context-sensitive menu opens, displaying the following commands and toggles as appropriate to certain columns and data types:

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.
Collapse All/Expand All	Input, Output, or Expected	Collapses and Expands the items in a test tree. These menu options operate on each tree separately. The previous expansion state of any children is lost by using these options.
Set to Null	Input or Expected	Applies only to attributes. Use Ctrl+click or Shift+click to choose several attributes.
Ignore Validation	Expected	Right-click on a node and chose Ignore Validation from the pop-up menu, all instances of that node in that view are ignored during the validation stage of testing, not just the single instance you selected. Each occurrence of that node that appears in the view is grayed-out to indicate that all of them are ignored during validation. You can toggle Ignore Validation on or off.
Key Attribute	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.
Scroll Lock	Anywhere on sheet	Synchronizes scrolling of all three columns.
Export JSON	Input, Output, or Expected	Exports the selected pane in the active Testsheet in JSON format.

Command	Columns	Comments
Export JSON to clipboard	Input, Output, or Expected	Exports the selected pane in the active Testsheet in JSON format to the user's clipboard.
Comments	Input, Output, or Expected	Add comments to entities, attributes, and associations. But only one comment per item. Annotations are displayed adjacent to that item, enclosed in braces. All comments and associated information are captured and displayed in the Comments View . See <i>Documenting Rule Assets</i> for more information about this feature.
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. • 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 section of the Data Integration Guide</i>.

Testsheet tabs

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

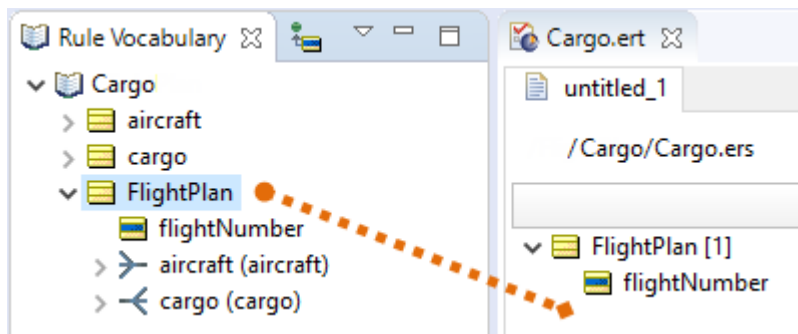


Populate the input panel

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”.

Add 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.



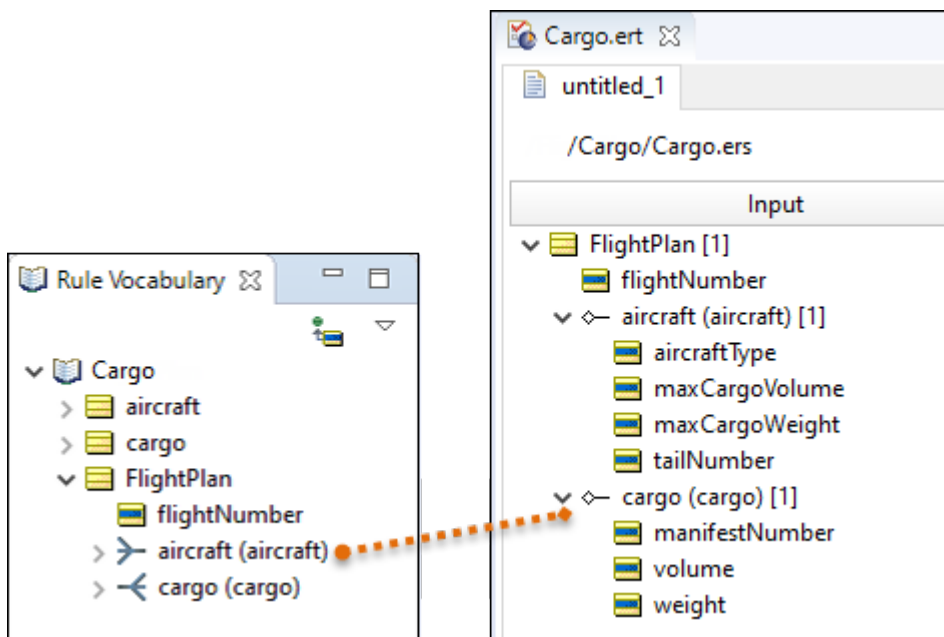
Important: Entities may be deleted by pressing the **Delete** key or by selecting **Delete** from the right-click pop-up menu.

Create 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.

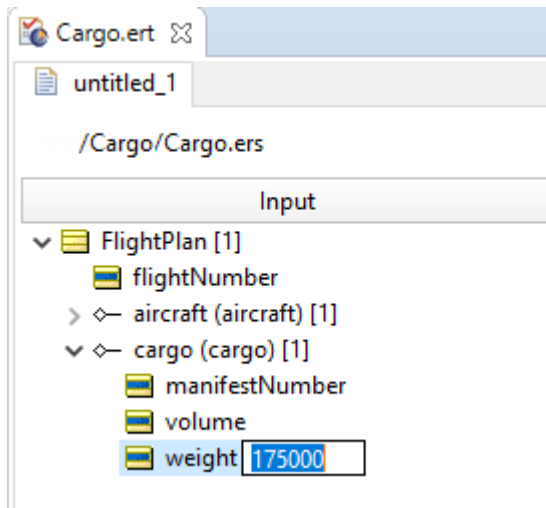


2. Verify that the child entity has the appropriate “indented” structure underneath the parent entity, indicating the association has been created correctly.

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.

Assign 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.


Execute tests

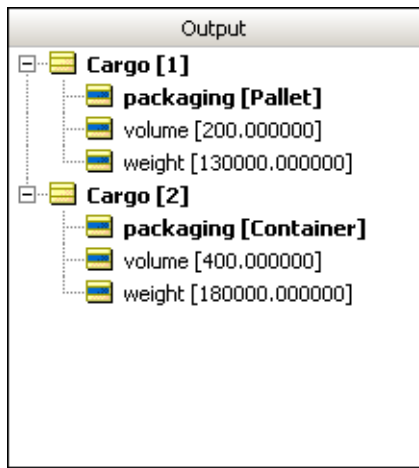
Before a Ruletest can be executed, the corresponding Rulesheets (whether being tested directly or as contained in Ruleflows) must be compiled. This compilation process does not occur until the Ruletest is run. Depending on how many Rulesheets and rules must be compiled, this compilation step may take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



3. Test results are displayed in regular type style. Parts of the test tree ("nodes") that were modified are shown in **bold** black text. Any errors are shown in red.
4. Make any necessary adjustments to the Rulesheet, save and re-test by repeating these steps.

Format of decimal scale

Decimal scale is in the same format as the Input. All Decimal values are rounded to the specified number of decimal places. Default value is 6. For example, 4.6059556 will be rounded, displayed, and/or returned as 4.605957.

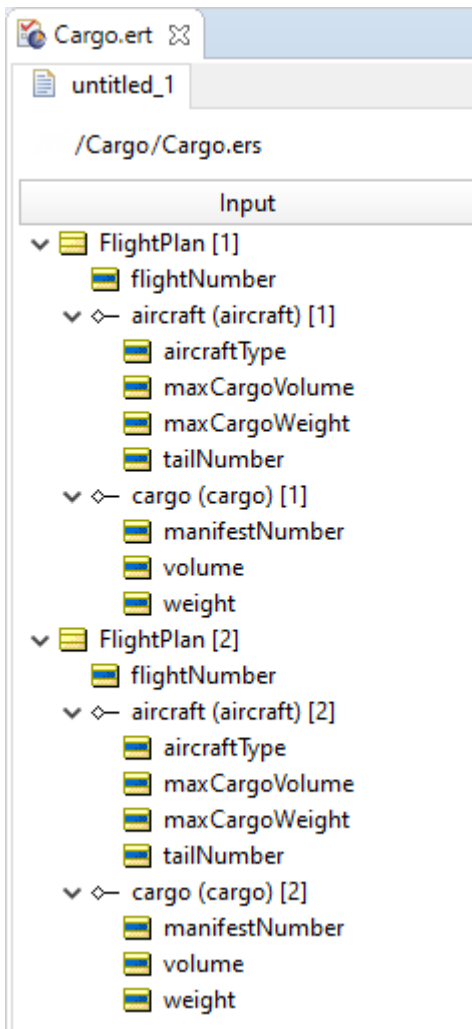
You can adjust the default precision for decimal values. In the `brms.properties` file, set the Studio's decimal scale:

```
decimalscale=2
```

When set to 2, the example's rounded value is 4.61.

Create 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.



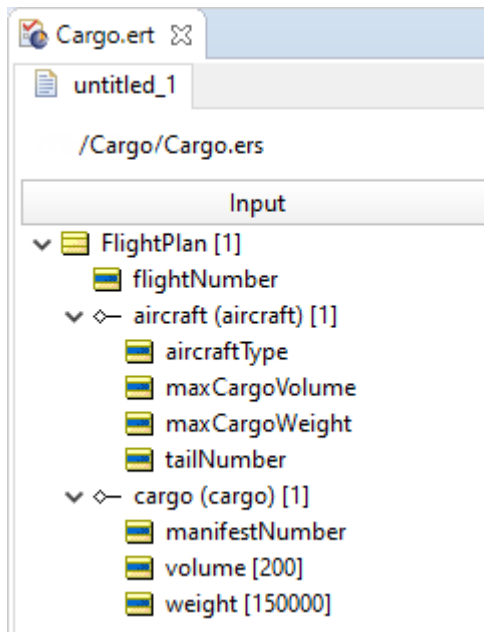
2. New elements are automatically numbered to ensure a unique ID for each.
3. Additional descriptive text can be added to each item in the **Input** column node by right-clicking the item, and then entering the type and text you want. As there can only be one comment per item, once you have added a comment, the menu item is no longer available. You must access and update or extend the existing comment.

The comments in the Input column will propagate to the Results column after you run a test.

Create 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.

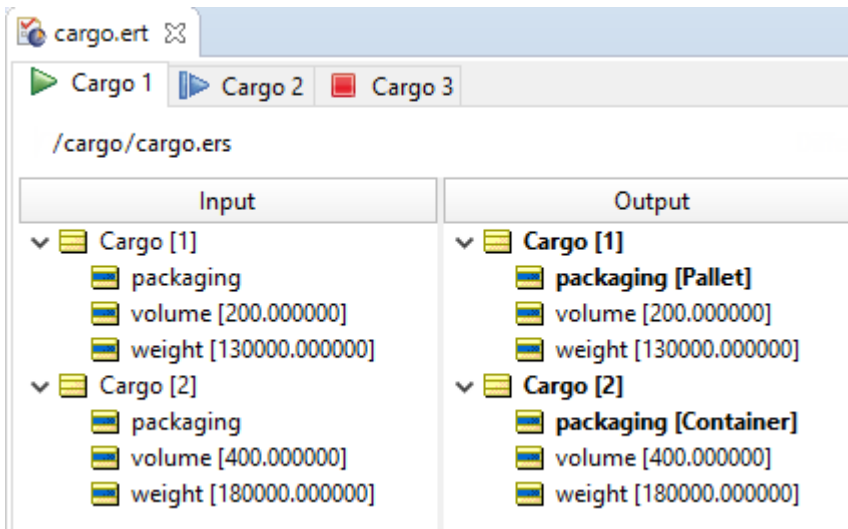
1. Each **Input** panel must be linked to a Rulesheet or Ruleflow.
2. Each **Input** panel will generate its own **Output** panel. Each of these Testsheet tabs can be [renamed](#) to help you distinguish them.




Create a sequential test using multiple testsheets

Multiple Rulesheets and Ruleflows can 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. Assume we want to test a Rulesheet, a Ruleflow, and another Rulesheet in sequence as **Cargo 1**, **Cargo 2** and **Cargo 3**.

1. Create a Ruletest in the project, name it `cargo.ert`, and choose `Cargo.ers` as the test subject.
2. Add data, and then execute the Ruletest to generate data in the Output panel.
3. Add a new Testsheet by choosing **Ruletest > Testsheet > Add Testsheet**, choose the file `cargo2.ersf`, and then name it **Cargo 2**.
4. Add another new Testsheet by choosing **Ruletest > Testsheet > Add Testsheet**, choose the file `flightPlan.ers`, and then name it **Cargo 3**.
5. Click on the **Cargo 3** tab, and then choose **Ruletest > Testsheet > Link to previous testsheet**. The tab is decorated with a red square and the adjacent tab is decorated with a blue arrow. The data from the Output panel of the previous Testsheet displays in the Input panel of the new Testsheet.
6. Click on the **Cargo 2** tab, and then choose **Ruletest > Testsheet > Link to previous testsheet**. The tab's blue arrow is now decorated with a split, and the adjacent tab is decorated with a green arrow. The data from the Output panel of the previous Testsheet displays in the Input panel of the new Testsheet. Here is how the tabs will look:

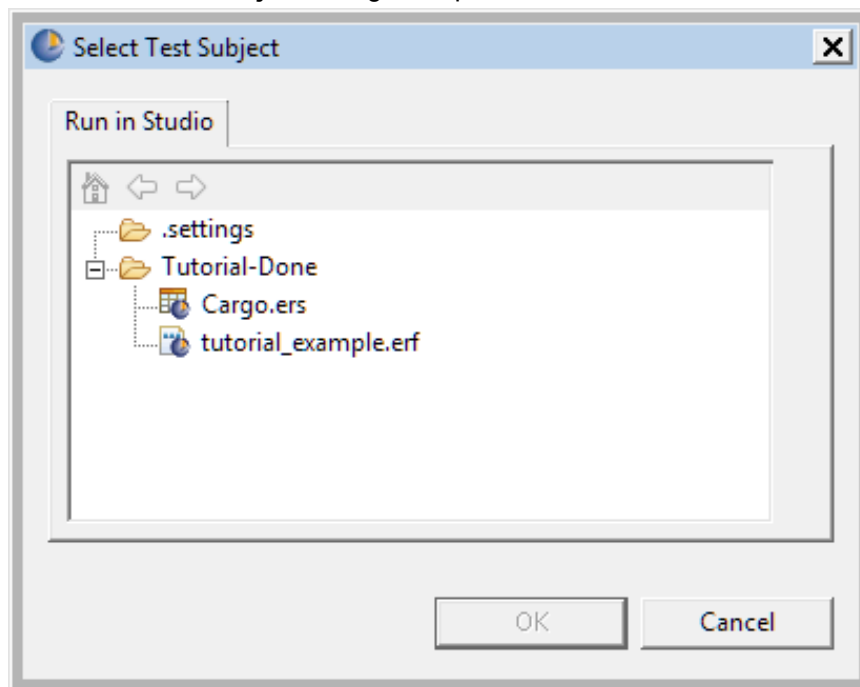


Repeat for as many new Testsheets as needed to model the sequence. You can execute one sheet at a time by selecting  from the toolbar, or execute all at once by selecting **Ruletest > Run All Tests**.

How to add testsheets

To add a Testsheet to the Ruletest in its editor, select the menu command **Ruletest > Testsheet > Add Testsheet**.

The **Select Test Subject** dialog box opens, as illustrated:



For the test subject, click on a listed Rulesheet or Ruleflow, as described in [Choose a test subject in the Studio workspace](#) on page 86

Click **OK**.

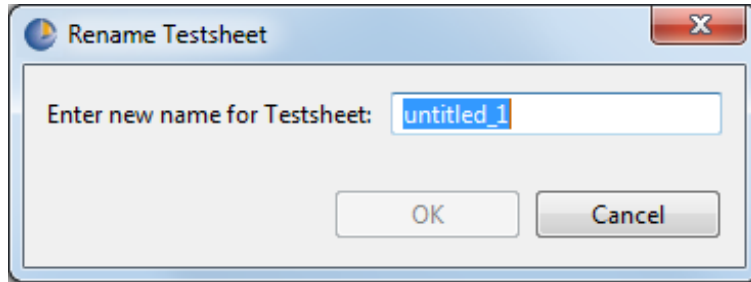
Corticon.js Studio creates a tab at the end of the current set of tabs, assigns the new tab a default name, and then opens the new Testsheet.

How to rename testsheets

When you add a new Testsheet, it is assigned a default name of `untitled_n`, where `n` is a sequential number. You can change each Testsheet name to a preferred name when its Ruletest is open in its editor in these ways:

- Open the Testsheet's tab, and then choose the menu action **Ruletest > Testsheet > Rename Testsheet**
- Open the Testsheet's tab, and then click the **Rename Testsheet** button on the toolbar
- Double-click on a Testsheet's tab

The **Rename Testsheet** dialog box opens, as shown:



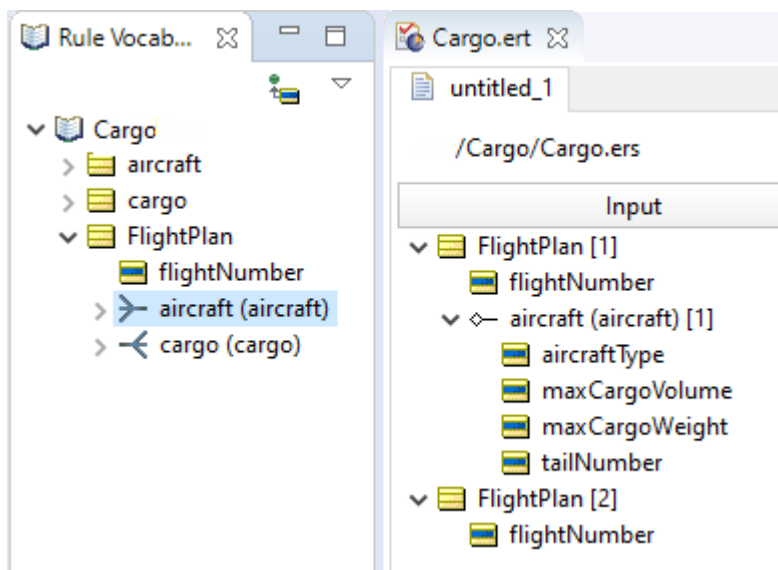
Enter the unique name you want to assign within this Ruletest.

Note: Ruletest and Testsheet names must comply with the guidelines described in [File naming restrictions](#) on page 16.

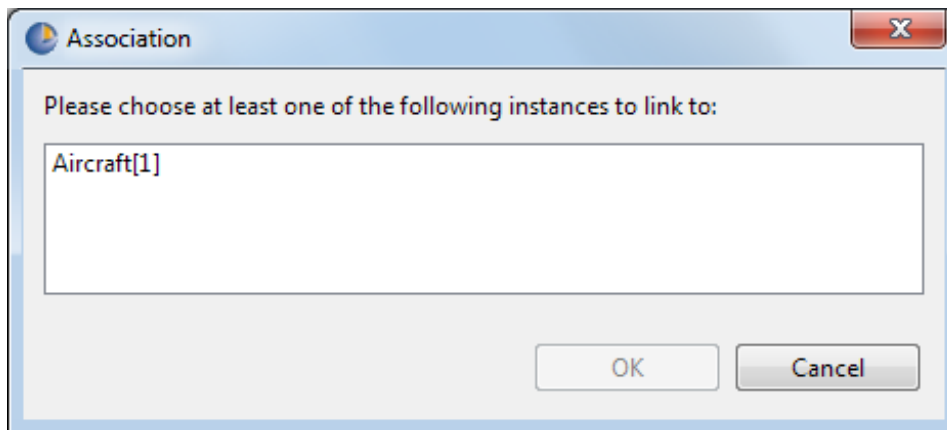
How to associate one child entity with more than one parent

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

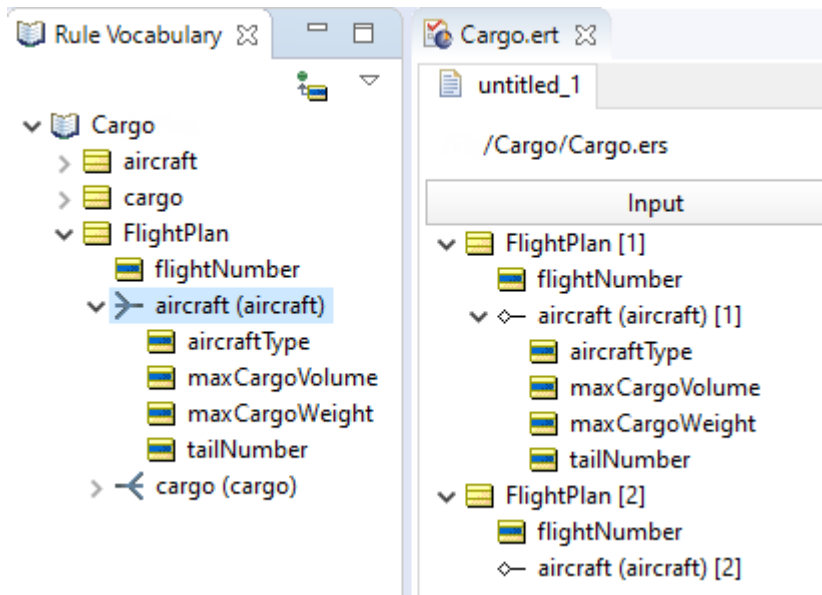
1. First, ensure the parent-child (source-target) association supports a multiplicity of many-to-many or many-to-one.
2. Drag and drop the parent entities (here, `FlightPlan`) as usual.
3. Create an association between `aircraft` and `FlightPlan[1]` as described in **Creating Associations**.
4. To associate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

How to save 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**.

How to import a JSON document to a testsheet

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

You can quickly generate an input document from an existing Testsheet. Click anywhere in the **Input** column, and then choose either **Export JSON** to create a file, or click **Export JSON to clipboard**.

The export file from the `Cargo` sample's `Cargo.ert` **Input** column is as shown:

```
[
  {
    "volume": 10,
    "container": null,
    "weight": 1000
  },
  {
    "volume": 40,
    "container": null,
    "weight": 1000
  },
  {
    "volume": 20,
    "container": null,
    "weight": 30000
  },
  {
    "volume": 10,
    "container": null,
    "needsRefrigeration": true,
    "weight": 1000
  }
]
```

How to export a testsheet as 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.

To generate any column of a testsheet to JSON, click on the column, and then choose either **Export JSON** to create a file, or click **Export JSON to clipboard**. The export file from the `Cargo` sample's **Output** column is as shown:

```
[
  {
    "volume": 10,
    "container": "standard",
    "weight": 1000
  },
  {
    "volume": 40,
    "container": "oversize",
    "weight": 1000
  },
  {
    "volume": 20,
    "container": "heavyweight",
    "weight": 30000
  },
  {
    "volume": 10,
    "container": "reefer",
    "needsRefrigeration": true,
    "weight": 1000
  }
]
```

How to create a Ruletest report

If the test subject of the Ruletest is a Ruleflow that has version identifiers set, the version is shown in the header of the report.

1. Select the menu command **Ruletest > Report**.
2. Choose the **Report Type**, **Report Style**, and **Output Folder** for this report.
3. Click **Finish**.
4. The Ruletest report opens as a new HTML page in your default web browser, and then saves the HTML, XML, and CSS files in the specified output folder.

The default CSS and XSLT files are samples that you can customize to suit your preferences.

How to document rule assets

As you design and develop rule assets, it is a good idea to document essential information about Vocabulary, Rulesheet, Ruleflow, and Ruletest files that you create, as well as the elements within them. For example, a comment can explain an asset's purpose and design, or annotate the use of an item within the asset. This information is particularly helpful to colleagues that maintain or enhance rule assets.

Note: Prior to the introduction of this feature, comments were added in Corticon Studio to an asset's **Properties** tab. When existing projects from earlier releases are opened in their editor, existing comments in the **Properties** tab are migrated to the **Comments** view.

For details, see the following topics:

- [About comments](#)
- [Add asset-level comments](#)
- [Add item-level comments](#)
- [Use the Comments view](#)

About comments

Corticon lets you associate comments with most items in your rule assets. These comments provide helpful information about the assets to current and future stakeholders – a best practice.

Levels of Comments

You can add two levels of comments, asset-level and item-level. Asset-level comments are associated with an asset itself, while Item-level comments are more granular and associated with items such as Vocabulary entities, domains, attributes, and associations.

Multiple Comments

You can add more than one comment to an asset or individual item within an asset. Using multiple comments allows you to create a history or audit trail for the asset or item.

Types of Comments

Each comment is an action **To Do**, a **Change Log**, an informative **Note** (the default type of comment), or a comment that **Needs Review**.

Rule asset specifics

Each type of rule asset has specific commenting behavior, as follows:

Asset	Notes
Vocabularies	<ul style="list-style-type: none"> Comments are added in the Vocabulary editor, not from the Project Explorer or Rule Vocabulary views. Comments can be added at the root level for the file, but not for Custom Data Types or Datasource definitions.
Rulesheets	<ul style="list-style-type: none"> Select a range of items (such as <code>a1 : c2</code>) to add and associate comments to them. If you later inject a new column within that range, the existing comments for that range are now associated to items in the added column too. And the Comments View is auto-updated to reflect the extended range (e.g., <code>a1:c3</code>). When a column or row is removed from the Rulesheet, comments associated with its items are removed. When a column or row is inserted into the Rulesheet, all comments are re-numbered accordingly. When you expand columns, existing comments are associated with the expanded columns. However, you cannot add comments to an expanded column directly. When you collapse or compress a column, associated comments are removed. Therefore, comments of compressed columns are removed. You cannot add comments to Rule Statements.

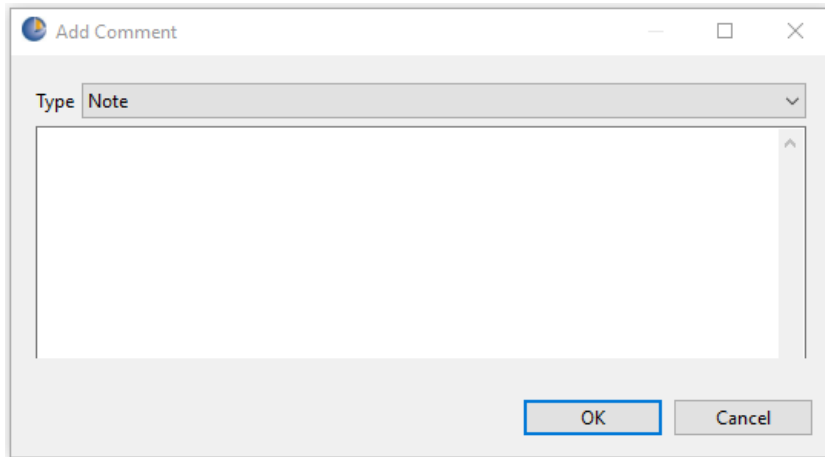
Asset	Notes
Ruleflows	<ul style="list-style-type: none">• To add a comment to a Ruleflow, select the item to comment, and then choose the menu command Ruleflow > Add Comments.• You can add comments to most objects on the canvas: Rulesheet, Ruleflow, Service Call-out, Branch, and Subflow. Connections and Iteratives do not support comments.• When an item is a node is selected, the comment is added to it as an item-level comment. When no object is selected, the comment is added as a file-level comment.• You can add several comments for an item or file.• When multiple objects are selected on the canvas, no comments are allowed.• Unlike other assets, the Ruleflow toolbar provides no Comments button.
Ruletests	<ul style="list-style-type: none">• You can add comments in the Input column to all entities, attributes, and associations.• Only one comment is allowed for each item.• Only one item can be selected to apply comment.• Comments can be added for each Testsheet, but not to the Ruletest file.

Add asset-level comments

To add comments to a rule asset file:

1. In Corticon.js Studio, open the file in its editor.

2. On the toolbar, click the **Comments** button . The **Add Comment** dialog opens:

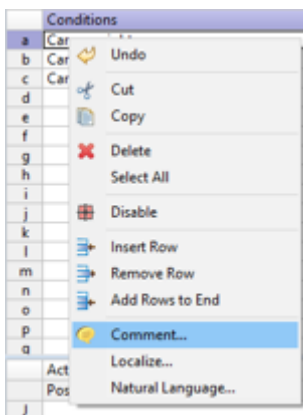


3. On the drop-down menu, select the type of comment you want: **To Do**, **Change Log**, **Note**, or **Needs Review**.
4. Enter a comment in the field provided.
5. Click **OK**. The comment is captured and displayed in the Comments View.

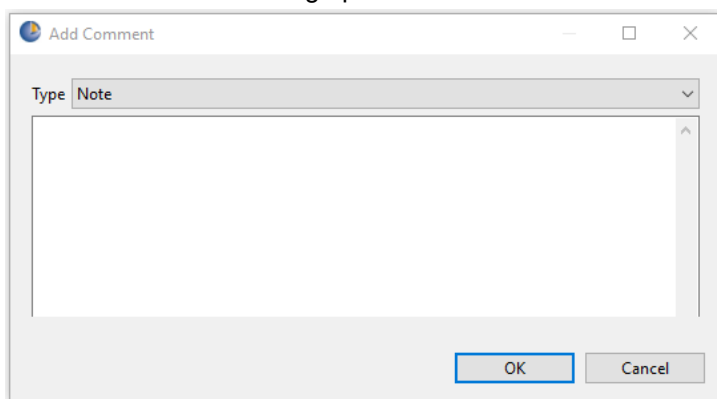
Add item-level comments

To add comments to items in a rule asset:

1. In Corticon.js Studio, open the asset file.
2. Right-click on the item or range of items you want to comment, and then select **Comments ...** on the context menu, as shown in a Rulesheet:



3. The **Add Comment** dialog opens:



4. Click the **Type** field's dropdown button, and then choose the type of comment you want to enter: **To Do**, **Change Log**, **Note**, or **Needs Review**.
5. Enter a description or annotation in the field provided.
6. Click **OK**. The comment is captured and displayed in the Comments View.

Note: Ruletest Testsheets support just one comment per item. Other assets support multiple comments per item.

Use the Comments view

The **Comments** view lists all comments associated with a rule asset and its elements. If the view is not currently showing, select the menu command **Window > Show View > Comments**.

The **Comments** view opens as a tab in the asset's workspace, as illustrated:

Rule Statements Rule Messages Properties Comments					
Search: type search text...					
DateTime	User	Type	Category	Item	Text
10/06/17 9:53:59 AM	cmcguire	TODO	Ers		Print out Rule Canonical Model
10/06/17 9:52:57 AM	cmcguire	Note	Condition Cell	{a1:b3}	Adding a comment to the range a:1 to b:3
10/06/17 9:51:12 AM	cmcguire	ChangeLog	Action	{A}	Reflects latest

where:

- **DateTime:** Displays the date and time when the comment was added or last edited. Read-only.
- **User:** Displays the name of the person that created the comment or last edited it. Read-only.
- **Type:** Identifies the type of comment entered: **Note**, **TODO**, **Change Log**, or **Needs Review**. The type can be changed.
- **Category:** Displays the class of element that was commented. Read-only.
- **Item:** Identifies the location of the item commented. Read-only.
- **Text:** Contains the body of the comment. The text can be edited as appropriate.

Editing a comment

To manage an existing comment:

- **Edit:** Double-click on a row in the Comments view to open that comment in the **Edit Comment** dialog where you can change the Type or Text of the comment.
- **Undo|Redo:** On the menu, select **Edit > Undo** and **Edit > Redo** to modify your changes.
- **Delete:** Select one or more comments in the Comments view, then right-click and choose **Delete**.

Managing the listed comments

The **Comments** view lets you sort, filter, and search the comments list.

- **Filter** : Turn auto filtering on to limit the displayed comments to just those defined for the item selected in the active asset editor. When auto filtering is off, all comments defined in the asset are displayed.
- **Sort:** Click a column header in Comments view to sort displayed comments by that field in ascending or descending order.
- **Search:** Enter search text in the **Search** field of the Comments view to display only comments containing that text in the text area.
- **Navigation:** Click on a comment in the Comments view to highlight in orange the **Item** range of the comment.

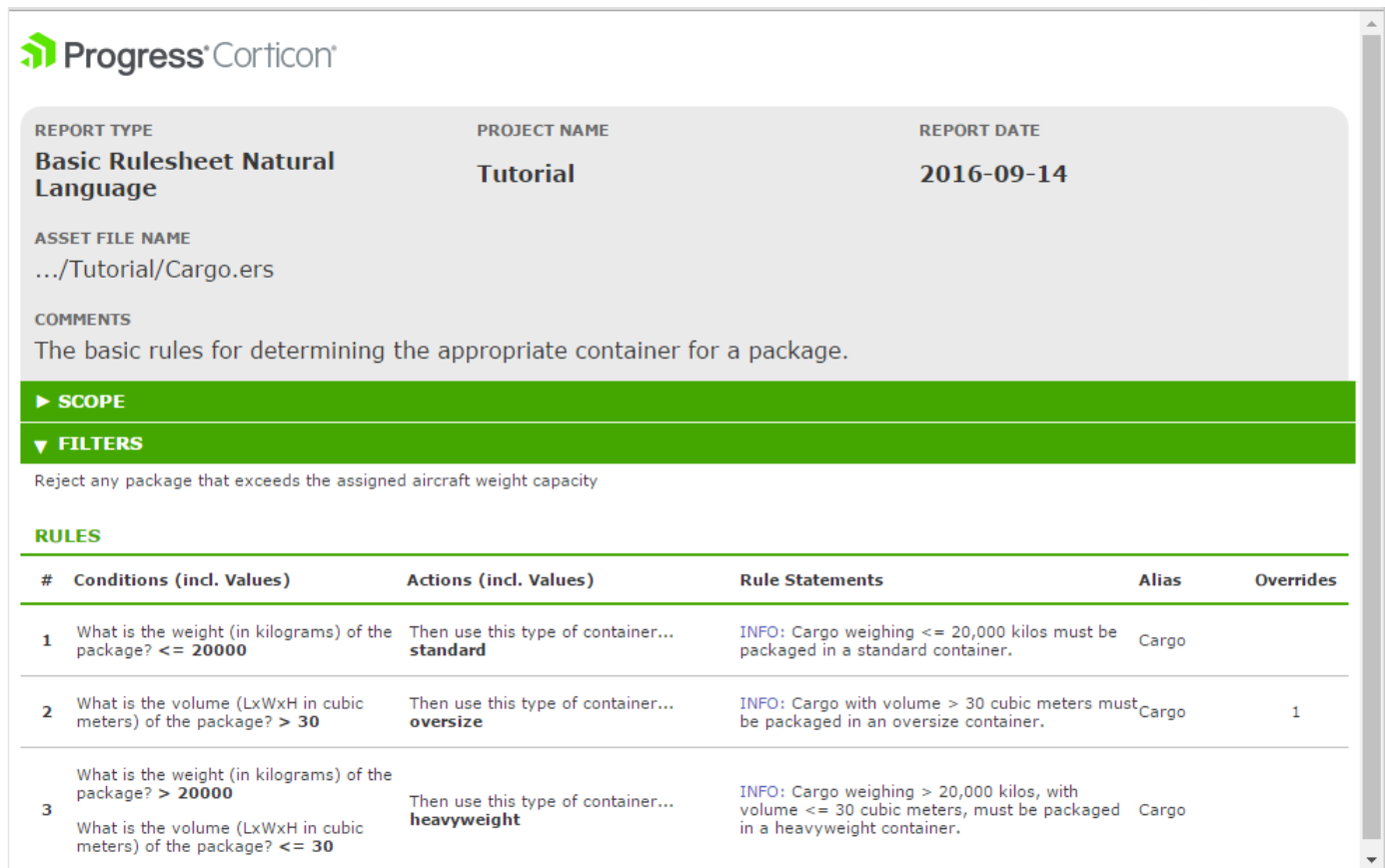
The Corticon Studio reporting framework

Corticon Studio lets you create reports on each of the assets in a project: Vocabulary, Rulesheets, Ruleflows, and Ruletests.

You generate reports from a dialog box that lets you choose from standard types and styles of reports, and then set the output folder.

The standard reports provide summaries at different levels of detail for the different asset types.

For example:



Progress Corticon

REPORT TYPE
Basic Rulesheet Natural Language

PROJECT NAME
Tutorial

REPORT DATE
2016-09-14

ASSET FILE NAME
.../Tutorial/Cargo.ers

COMMENTS
The basic rules for determining the appropriate container for a package.

► **SCOPE**

▼ **FILTERS**
Reject any package that exceeds the assigned aircraft weight capacity

RULES

#	Conditions (incl. Values)	Actions (incl. Values)	Rule Statements	Alias	Overrides
1	What is the weight (in kilograms) of the package? <= 20000	Then use this type of container... standard	INFO: Cargo weighing <= 20,000 kilos must be packaged in a standard container.	Cargo	
2	What is the volume (LxWxH in cubic meters) of the package? > 30	Then use this type of container... oversize	INFO: Cargo with volume > 30 cubic meters must be packaged in an oversize container.	Cargo	1
3	What is the weight (in kilograms) of the package? > 20000 What is the volume (LxWxH in cubic meters) of the package? <= 30	Then use this type of container... heavyweight	INFO: Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be packaged in a heavyweight container.	Cargo	

The standard report types are:

Report Type one of the XSLT files for the asset type:

- **Vocabulary**
 - Basic Vocabulary
 - Detailed Vocabulary
- **Rulesheets**
 - Basic Rulesheet Expressions
 - Basic Rulesheet Natural Language
 - Detailed Rulesheet Expressions
 - Detailed Rulesheet Natural Language
- **Ruleflows**
 - Basic Ruleflow Expressions
 - Basic Ruleflow Natural Language
 - Detailed Ruleflow Expressions
 - Detailed Ruleflow Natural Language
- **Ruletests**
 - Basic Ruletest

The type files are located at `[CORTICON_WORK_DIR]\Studio\Reports\XSLT\` in folders according to the asset types. You can copy the files to use as templates or change them to create report types that are then offered in the **Report Type** dropdown menu for the asset type.

Report Style is the CSS stylesheet to use for the report. The basic stylesheets are:

- Corticon Blue
- Corticon Green

The style files are located at `[CORTICON_WORK_DIR]\Studio\Reports\CSS\`. You can copy a stylesheet file to use as a template to create custom report styles that are then offered in the **Report Style** dropdown menu.

Output Folder is the location where the report will be stored on disk. The default location is `[CORTICON_WORK_DIR]/Studio/Reports`. You can create a root location such as `C:\CorticonStudioReports` and then append subfolder names to sort out your projects, tasks, clients, or versions.

When you have set the parameters of the report, and then click **Finish** the processing takes place as follows:

1. Generates an XML file using its built-in XML template
2. Uses the selected XSLT transformation to move the XML into HTML
3. Applies the selected CSS to render the HTML file in a web browser page
4. Copies the XML, HTML, and CSS files to the specified output folder.

Customizing the XSLT stylesheet transforms

Corticon.js Studio's initial XSLT stylesheets can be modified or copied to create preferred XSLT files to generate custom Studio reports. Name and save your modified XSLT file in the appropriate XSLT subfolder so that it will be listed and callable when you run reports.

Customizing the CSS stylesheets

Corticon.js Studio's initial CSS stylesheet can be modified or copied to create a preferred CSS files to render standard and custom Studio reports. Name and save your file in the CSS folder so that it will be listed and callable when you run reports.

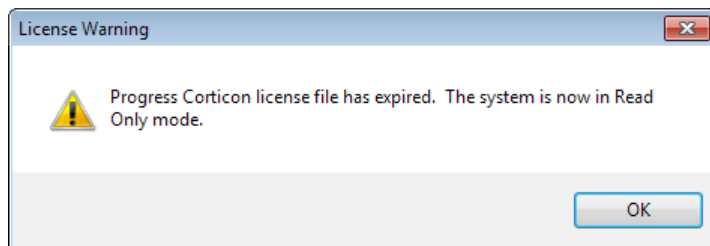
See also:

- [How to create a Vocabulary report](#) on page 39
- [How to create a Rulesheet report](#) on page 58
- [How to create a Ruleflow report](#) on page 64
- [How to create a Ruletest report](#) on page 103

Studio license expiration

If your license indicates that it has expired, contact your Progress Corticon.js representative to obtain an updated license file. Corticon.js Studio alerts the user at startup, and then limits functionality:

Figure 4: License expiration alert at Studio startup



How to exit Corticon Studio

When you close the Corticon.js 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.
