Progress®

Corticon
Quick Reference

Progress® Corticon®

Copyright

Updated: 2019/07/31

Table of Contents

1

A guide to Progress Corticon Studio

The Progress Corticon Studio is where rules are created, maintained, tested, and packaged for deployment. The Studio is a workbench that enables work in Corticon's component editors and files that comprise **Rule Projects**:

1. **Vocabulary**: a structured dictionary containing all necessary business terms and relationships between them used by the business rules.
2. **Rulesheet**: a set of conditions and actions and plain language statements written from a common business Vocabulary.
3. **Ruleflow**: a set of one or more Rulesheets, embedded service call-outs, and other Ruleflows organized for sequential execution. A Ruleflow is compiled into a versioned Decision Service, and then deployed on Servers.
4. **Ruletest**: a set of one or more Testsheets containing sample data used for testing Rulesheets and Ruleflows. Like Rulesheets, Ruletests also use a common Vocabulary model.

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

Following construction and testing within Corticon Studio, a Rulesheet must be saved before it can be tested by a Ruletest. Ruleflows must be compiled before they can be deployed to Corticon Server.

File Suffixes

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

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

For details, see the following topics:

- Initial menu commands
- Keyboard shortcuts
- Initial toolbar
- The file tab
- The active Corticon Studio window
- Alternate settings for Corticon Studio properties
- File naming restrictions
- Renaming and relocating assets in the Project Explorer
- Using SVN or Git to manage project assets

Initial menu commands

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

The following actions are available when Corticon Studio opens:

File Menu

The actions accessed on the File menu are:

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

The other File menu options perform standard Eclipse functions.

Edit Menu

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

- **Select All** - Applies only to Rulesheets and Ruleflows.

- **Find/Replace** - Not enabled in the Corticon Designer.
- **Add Task** - Not enabled in the Corticon Designer.
- **Insert Row** - Inserts a new Row into the active file.
- **Remove Row** - Removes the selected Row from the active file.
- **Add Rows to End** - Inserts 10 Rows at the end of the active file.
- **Insert Column** - Inserts a new Column into the active file.
- **Remove Column** - Removes the selected Column(s) from the active file.
- **Add Columns to End** - Inserts 10 Columns at the end of the active file.
- **Move Up** - Moves a Custom Data Type (CDT) enumerated Label/Value row up in the list. See the *Rule Modeling Guide's* “Building the Vocabulary” chapter for more info on CDTs.
- **Move Down** - Moves a Custom Data Type (CDT) enumerated Label/Value row down in the list.
- **Enable / Disable** - Toggles enabling and disabling of a column, row, cell or shape. (This feature lets the rule builder experiment with temporarily removing logic without removing its information from the work.) The hierarchy of granularity is as follows:

  1. Rulesheet columns and rows
  2. Individual THEN cells (these are the cells in the lower-right quadrant of the Rulesheet)
  3. Rule Statements
  4. Shapes on the Ruleflow

Navigate Menu

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

Search Menu

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

Project Menu

The actions accessed on the Project menu are standard Eclipse functions, except the following:

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

Run Menu

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

Vocabulary, Rulesheet, Ruleflow, Ruletest Menus

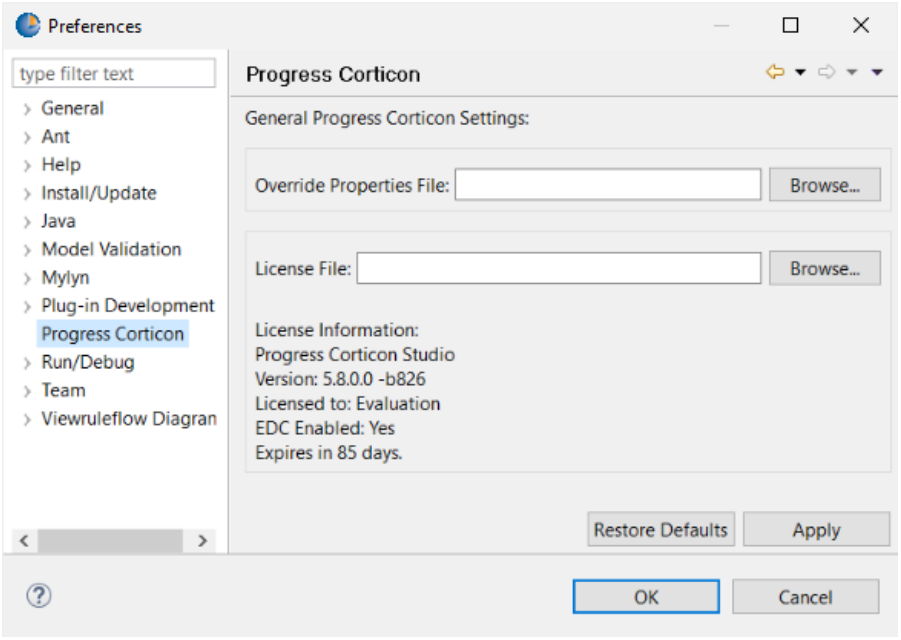
Note: When a Corticon file is active, the corresponding menu is added to the toolbar. Each of these is described in later sections of this guide, as follows:

- Vocabulary menu commands on page 33
- Rulesheet menu commands on page 66
- Ruleflow menu commands on page 92
- Ruletest menu commands on page 121

Window Menu

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

- **Open Perspective** - Opens a defined set of views and editors. If already open, returns to that instance.
- **Open Perspective > Corticon Designer** - Resets the layout to the views commonly used in Corticon modeling.
- **Open Perspective > Other** - Opens the perspective dialog where you can choose an available perspective.
- **Show View** - Adds a perspective-related view as a tab in the perspective window. Views in the Corticon Designer include **Error Log**, **Localization**, **Natural Language**, **Problems**, **Properties**, **Rule Message**, **Rule Operations**, **Project Explorer**, **Rule Statements**, and **Rule Vocabulary**. **Other** lists all available views, whether or not directly relevant in the current perspective.
- **Preferences** - Opens the **Preferences** dialog for the installation. Corticon preferences include:
  - **Progress Corticon**:



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

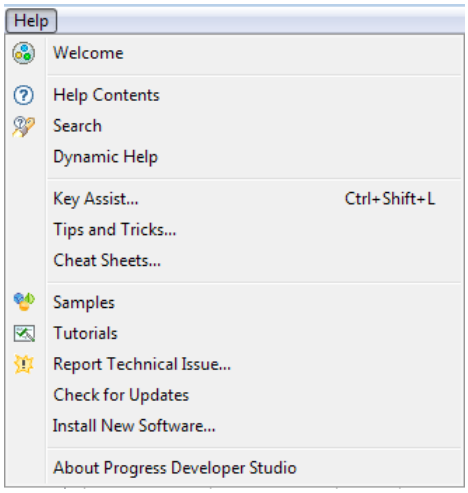
    Note: Changing the override properties file -- and changes within an override file --- are not applied until you restart the Studio.

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

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

- **Welcome** - Opens the Welcome page with access to What's New, Samples, Tutorials, Documentation, and Web Resources. Use the **Home** button in the upper right if the page that opens is not the top page.
- **Help Contents** - Opens the help system window where you can access the up-to-date documentation for this release of the Studio and its related Corticon tools. You can also search for terms and browse the help structure of Eclipse and third-party tools.
- **Search** - Implements the help system on a tab instead of in a separate window.
- **Dynamic Help Contents** - Opens context-sensitive Corticon help plus access to related topics.
- **Key Assist** - Keyboard shortcuts are detailed in the next topic. You can press **Ctrl+Shift+L** to open the **Key Assist** panel.
- **Samples** - Opens the **Welcome** page's **Samples** panel to access and unpack Corticon sample projects.
- **Tutorials** - Opens the **Welcome** page's **Tutorials** panel to access the Basic and Advanced tutorial documents.
- **Check for Updates** - Accesses Progress and related sites to determine whether updates are available.
- **About Progress Corticon Studio** - Details versions and licenses of installed Corticon Studio, Eclipse, and other installed products. Also provides access to installation details.

Keyboard shortcuts

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

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

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

Initial toolbar

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

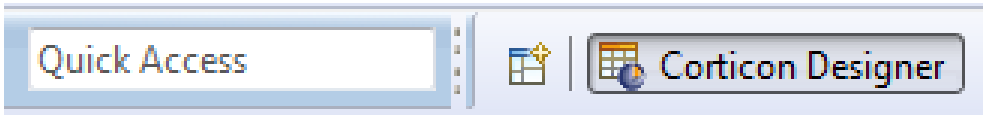
The tools on the toolbar initiate the following actions:

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

-  Removes the selected column(s) from the active file.
-  Inserts 10 columns at the end of the active file.
-  Inserts a new row into the active file.
-  Removes the selected row from the active file.
-  Inserts 10 rows at the end of the active file.

Standard Eclipse tools are then appended to the toolbar:

and added at the right end of the toolbar:



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

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

- Vocabulary toolbar on page 36
- Rulesheet toolbar on page 68
- Ruleflow toolbar on page 93
- Ruletest toolbar on page 125

The file tab

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

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

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

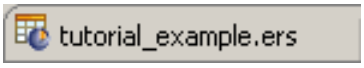
The active Corticon Studio window

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

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

Window Tabs

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

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

Alternate settings for Corticon Studio properties

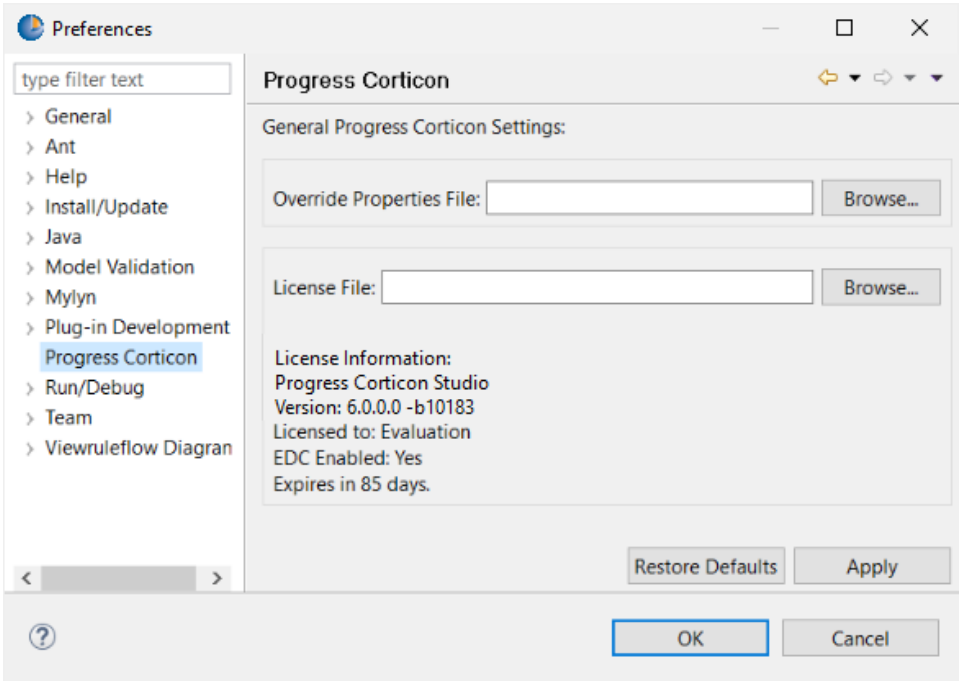
Corticon changes many behaviors based on properties that you can customize in `[CORTICON_WORK_DIR]/brms.properties`. Studio enables you to specify variations of this file that you can locate and name as suits you.

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

You can choose to point Corticon 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 the Server'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 click on **Progress Corticon**.
3. Enter or browse to the override properties file you created, as shown:



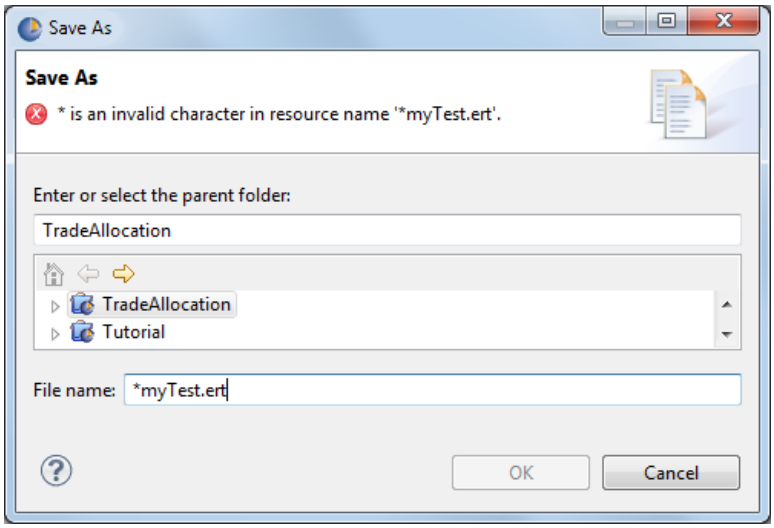
After editing and saving your overrides file, restart Corticon components for their changes to take effect. The properties in these files are described in detail in the following topics. When Corticon starts up, it will apply your properties file with your settings.

For more information, see the topic *"Configuring Corticon properties and settings" in the Server Guide.*

File naming restrictions

File names must comply with the following rules:

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



Renaming and relocating assets in the Project Explorer

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

Examples:

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

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

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

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

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

Using SVN or 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 plugins for two popular software version control systems in use by many developers today, Apache Subversion (**SVN**) and **Git**.

Use SVN, 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 project and creating a local working copy;
- Copying, moving, and renaming files in the repository as necessary;
- Checking-in or committing a file to the repository;
- Recording changes to a file or a set of files over time so that you can recall specific versions later;
- Branching a new line of development from the main line of development, and merging and synchronizing changes between branches.

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.

SVN

The Corticon SVN plugin provides an SVN perspective, a **File > Import** function, and **Preferences** in the **Team** section. However, an SVN Connector is not included with the Subversion SVN plugin bundled with Corticon Studio. You can download a Subversion SVN connector compatible with the Subversion plugin's version from the Eclipse Marketplace at https://marketplace.eclipse.org. For more information about Subversion SVN connectors, see http://www.eclipse.org/subversive/installation-instructions.php.

To access SVN documentation, downloads, and community, see https://subversion.apache.org/docs/

Git

The Corticon 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

2

Rule Projects

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

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

For details, see the following topics:

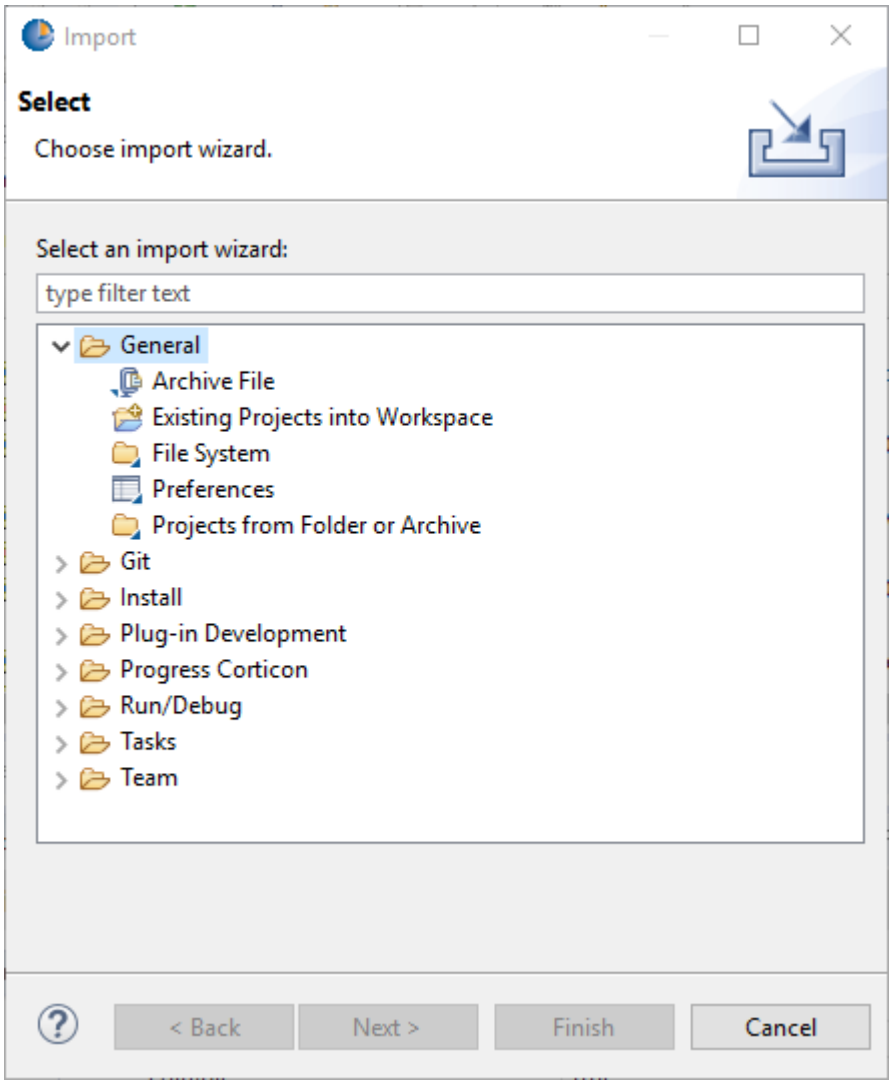
- Importing or exporting a Rule Project
- Creating a Rule Project
- The Project Explorer window

Importing or exporting 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 standard Eclipse functions, documented in the Eclipse help sections **Workbench User Guide > Tasks > Importing** and **Exporting**.

General options let you access archive files, project folders, or loose files. Imports assume the current workspace as the target folder.



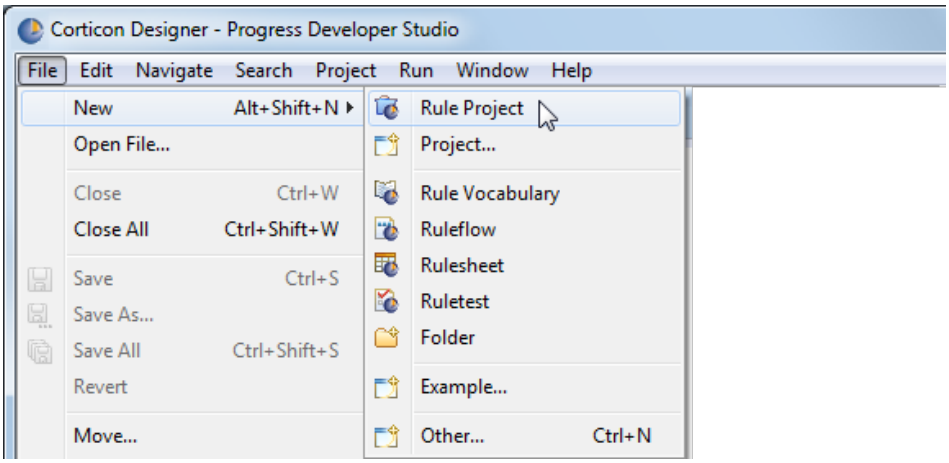
After importing a project, be sure to upgrade rule assets before making changes to assets.

Before performing a project export, choose **Save > All**, and then choose **Close > All**.

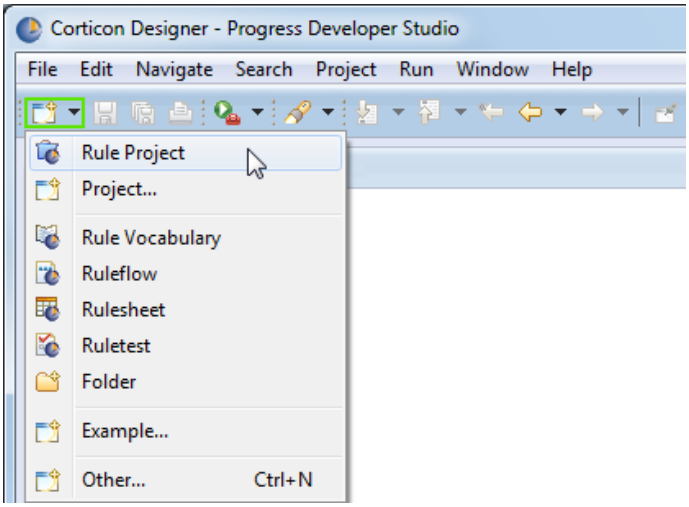
Creating a Rule Project

To create a new Rule Project, either:

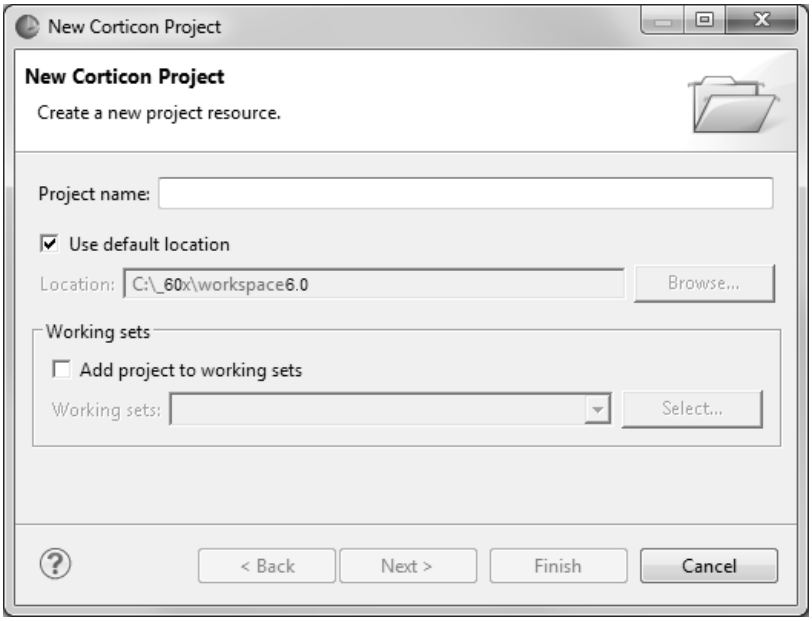
- In Corticon Studio, select **File > New > Rule Project**



- Select **New** on the menu, and then select **Rule Project**.



1. The **New Project** dialog box opens, as shown:



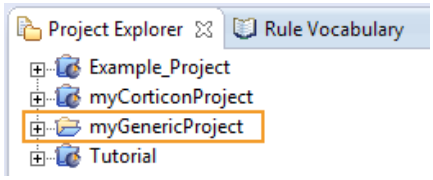
2. Check the **Use default location** option

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 Studio's default workspace folder to define its **Location**.
4. Click **Finish** to create your new Rule Project.

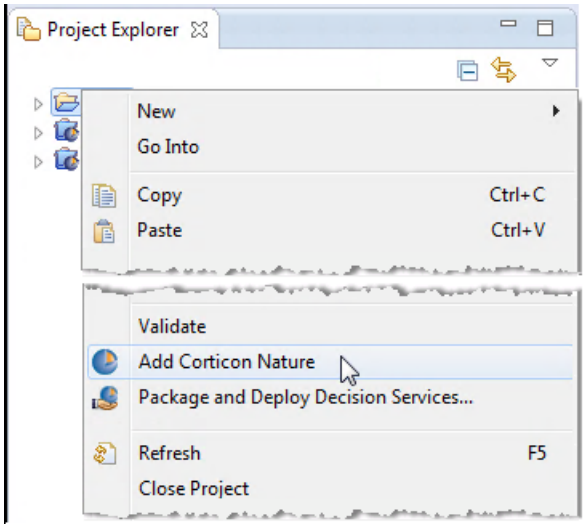
Corticon Nature

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

Corticon 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 and generic Eclipse projects. Projects having a Corticon Nature are distinguishable by the Corticon icon added to the project icon.



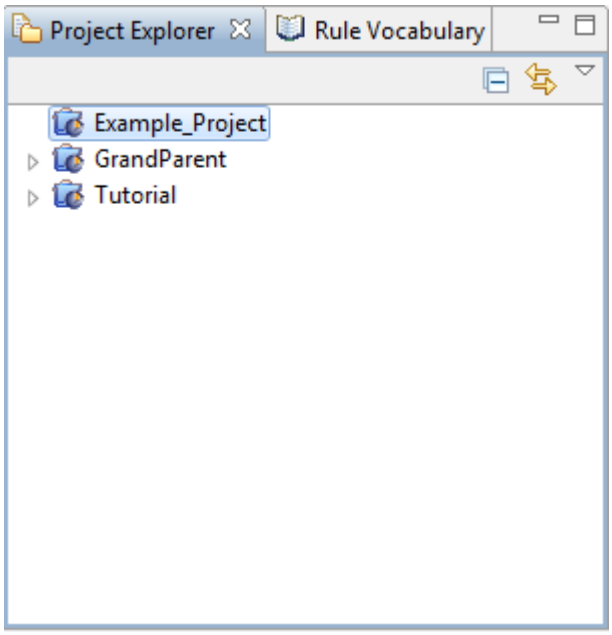
To convert a generic project to a Corticon project, right-click on the project's folder, and then select **Add Corticon Nature**, as illustrated:



The Project Explorer window

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

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

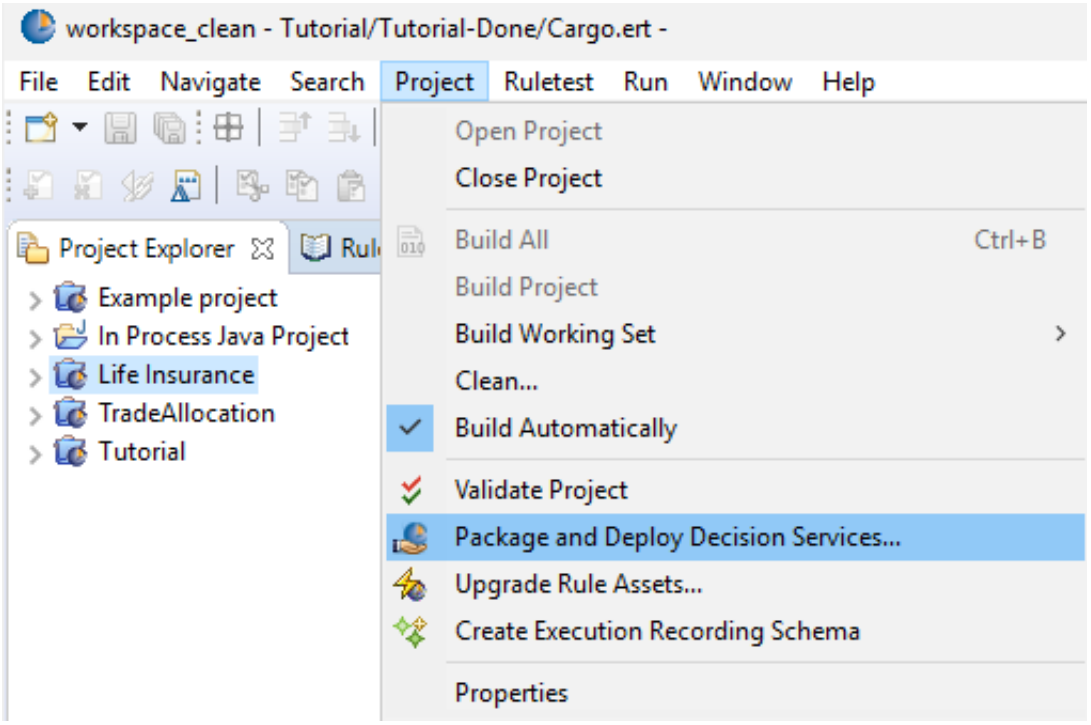


Project Menu

As Rule Projects do not have an editor, their project-level actions are appended to the standard Eclipse Project. The**Project** menu has standard Eclipse functions, as well as the following:

- **Validate Project** - The project validation process has an impact on available memory, and uses background CPU cycles. When validating large projects, follow the build progress in the **Progress** view (exposed from the menu item **Window > Show View > Other** then **General**: **Progress**
- **Package and Deploy Decision Services** - Lets you choose selected Ruleflows to compile as Decision Services. For details, see the topic *"Using Studio to compile and deploy Decision Services" section of the Deployment Guide.*
- **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 Assets** dialog. For a discussion of this feature, see *"Upgrading projects coming forward from a prior release" in the Corticon Installation Guide*.
- **Properties** - Opens the dialog that provides several advanced features, one of which enables Corticon Extensions to add service call-out JAR files that will be packaged with Decision Services created in the project. See the topic *"Building the Java classes and JARs" in the Extensions Guide.*

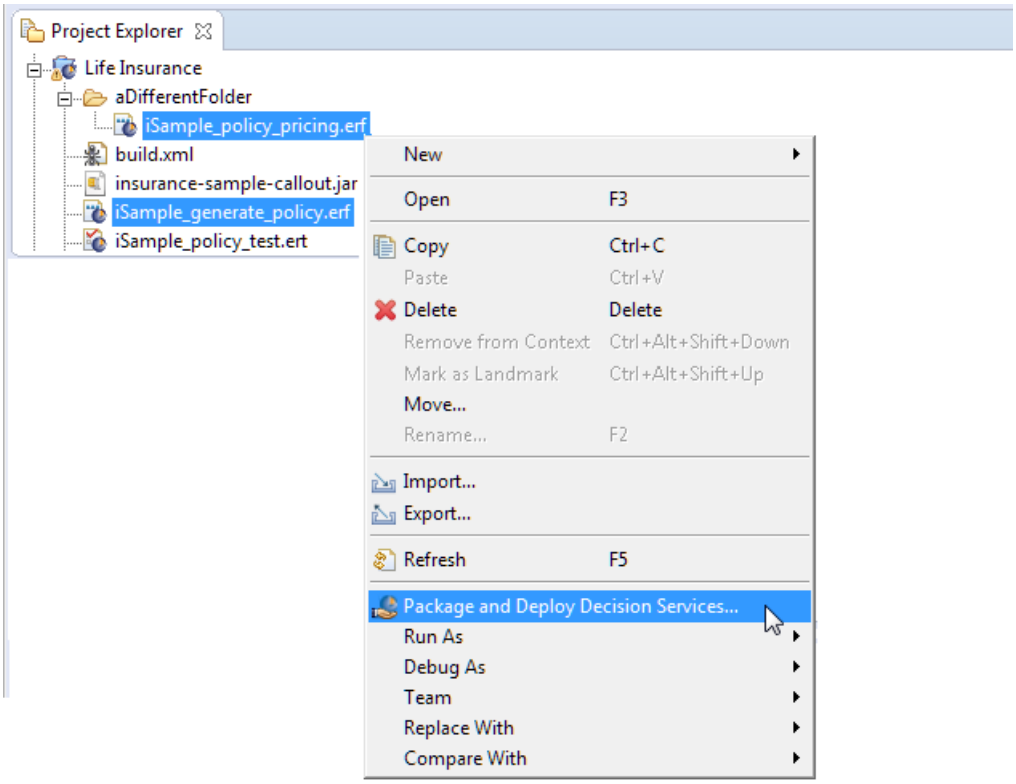
These commands are also accessible on the dropdown menu that opens when you right-click on a project, as shown in this excerpted image where the intent is to package and deploy selected Ruleflows that are in the **Life Insurance** project:



Publishing selected Ruleflows from Project Explorer

You can select specific Ruleflows for packaging and deployment from the Project Explorer view, and then publish only those Ruleflows as a Decision Service. To package and deploy selected Ruleflows from a project, do the following:

1. Select a project or folder in your Project Explorer view.
2. Select one or more Ruleflows across folders in the project.
3. Right-click and select **Package and Deploy Decision Services**, as illustrated:



When you choose your deployment target, the specified Ruleflows have been selected for compilation and deployment as Decision Services. Forces you to save all changes, then opens the **Package and Deploy Decision Services** dialog. See *"Using Studio to compile and deploy Decision Services" in the Deployment Guide.*

Upgrading projects coming forward from a prior release

When you install a new release of Corticon Studio, your Corticon development assets (Vocabularies, Rulesheets, Ruleflows, and Ruletests) might need to be upgraded for use in the Corticon Studio. These could be projects you were working on or samples you imported from another location. When you open an asset, Corticon Studio checks the version of the asset and alerts you if it needs to be upgraded. As you open each asset that needs to be upgraded, you are alerted as to the required action, and encouraged to proceed to upgrade it. The asset upgrade process will force the upgrade of dependent assets that also require a similar upgrade. This task can be tedious if you have a lot of assets and a lot of projects.

The project upgrade utility lets you choose projects or folders of projects that will be upgraded all in one pass. Typically, you would update projects in your workspace but, you can also choose to update assets in folders outside your workspace.

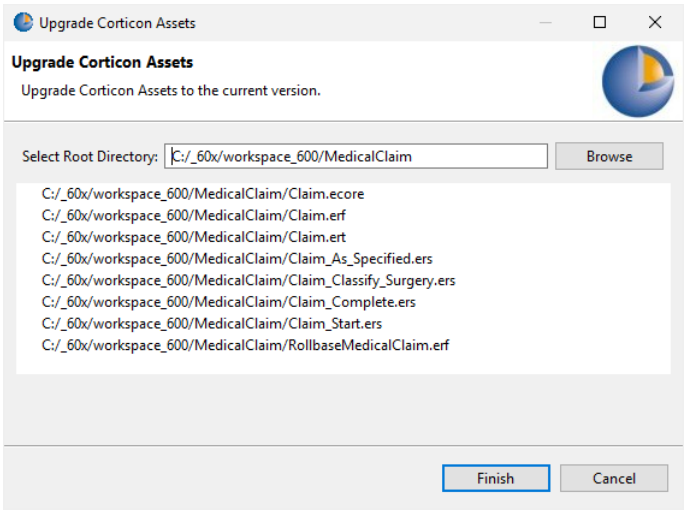
Performing this procedure just after upgrading your Corticon Studio will relieve you of having to do this task as you work.

Note: Once you upgrade your assets, you cannot open them with earlier versions of Corticon Studio.

To upgrade a selected project:

1. Choose the **Upgrade Rule Assets** command, 

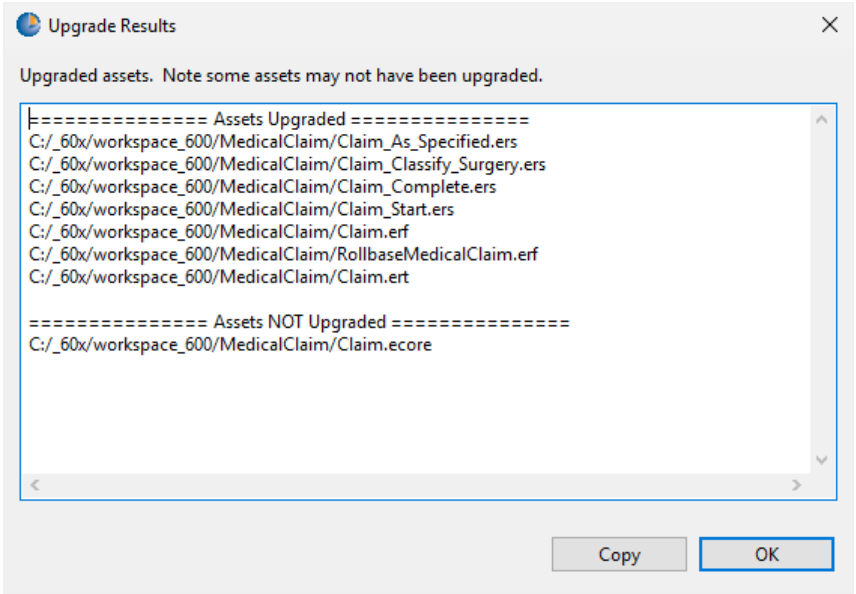
, on the dropdown menu that opens when you right-click on any project name in the **Projects Explorer** view. The **Upgrade Corticon Assets** dialog box opens, as shown:



   In this example, the **MedicalClaim** project was selected so all the assets in that project are selected for upgrade.

2. Click **Finish** to start upgrade processing. You are alerted that this process cannot be undone, so consider whether you should cancel, backup the files as-is, and then run the same upgrade process again to completion.

3. After processing, the **Update Results** window opens, listing the assets that required upgrade and were upgraded, as well as those that did not require an upgrade, as shown:



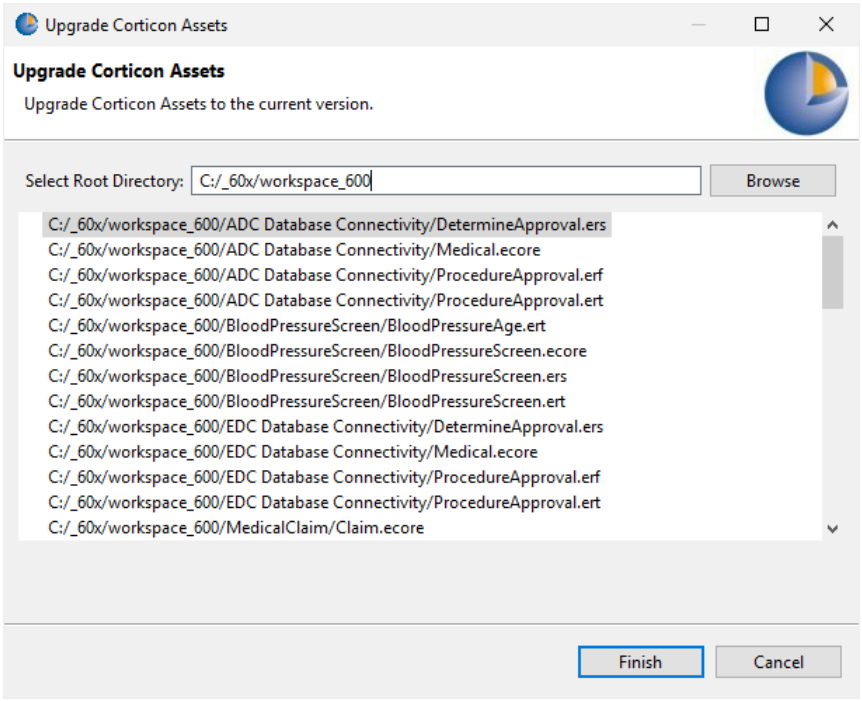
You can choose to click **Copy**, and then paste the text into your preferred text editor.

4. Click **OK** to close the results window and the dialog box.

To upgrade multiple projects not in your workspace:

1. Choose the **Upgrade** command, 

, from the **Project** menu. The **Upgrade Corticon Assets** dialog box opens.
2. Click **Browse** to navigate to the root folder containing Rule Projects are rule assets, and click **OK**. The rule assets are listed in the **Upgrade Corticon Assets** dialog box, as in this example:



3. Click **Finish** to start upgrade processing. You are alerted that this process cannot be undone, so consider whether you should cancel, backup the files as-is, and then run the same upgrade process again to completion.
4. After processing, the **Update Results** window opens, listing the assets that required an upgrade and were upgraded, as well as those that did not require an upgrade. To record this information, simply copy the text from this window and save it elsewhere.
5. Click **OK** to close the results window and the dialog box.

3

Vocabularies

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

Vocabulary files have the file suffix `.ecore`.

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

For details, see the following topics:

- Creating a Vocabulary
- The Vocabulary window
- Populating a new Vocabulary
- Finding references to an entity, attribute, or association in a project
- Refactoring entity attribute or association names in a project

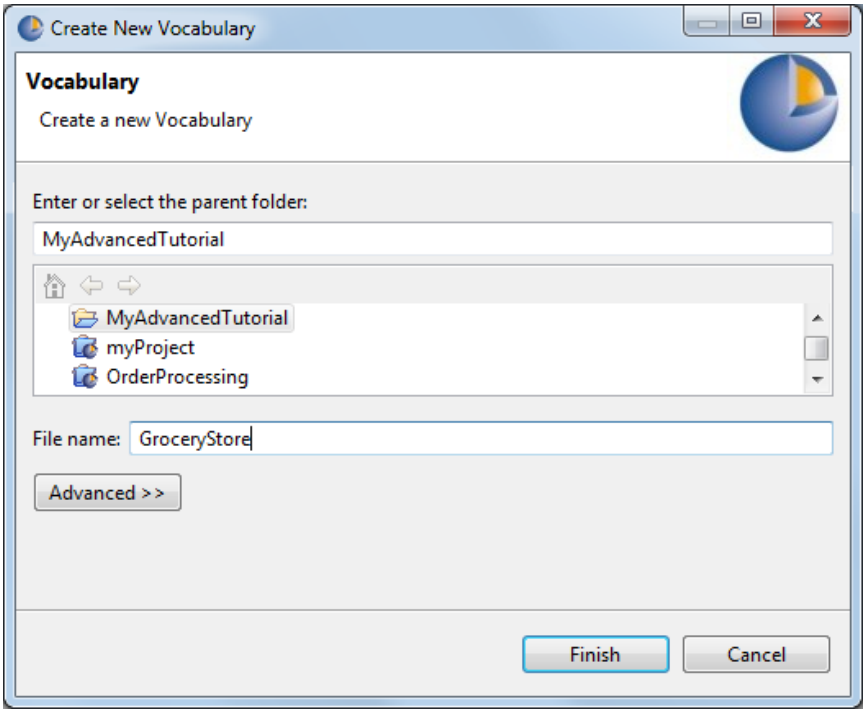
Creating a Vocabulary

To create a new Vocabulary:

1. Do one of the following:

- Select **File > New > Rule Vocabulary** from the Corticon Studio menubar
- Click the down arrow to the right of the **New** icon
- Right-click in the **Project Explorer** to open its menu, and then choose **New > Rule Vocabulary**.

These techniques all launch the same **Create a New Vocabulary** wizard, as shown:

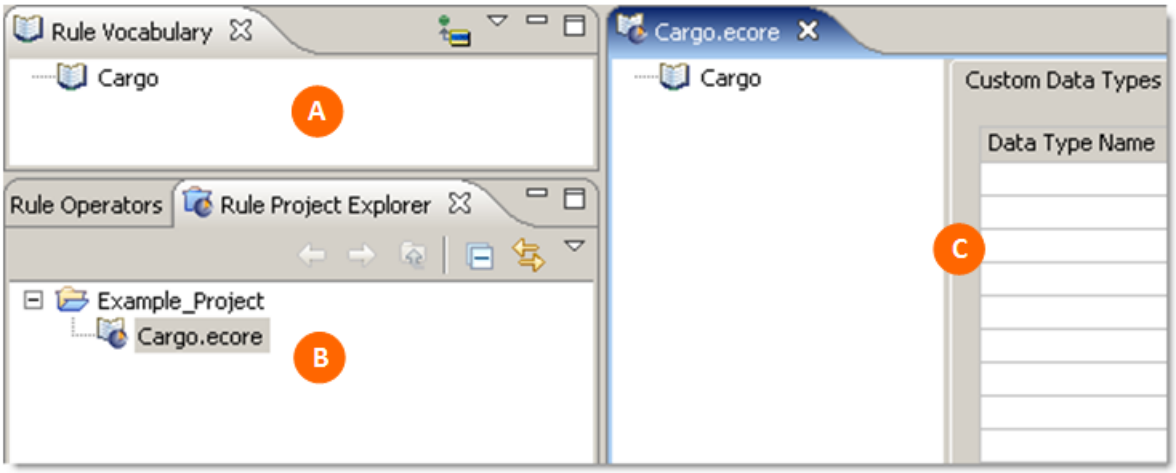


2. Select the parent Rule Project for the new Vocabulary by highlighting the `Example_Project` folder we just created.
3. Enter a name for the new Vocabulary in the **File name** entry area. It is not necessary to type the file extension `.ecore` (we used `Cargo` here).

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

---

4. Click **Finish** to create your new Vocabulary. It is now displayed in the new **Rule Vocabulary** window (**A**), in the **Project Explorer** window (**B**), and as the open file and active tab in the **Cargo.ecore** window (**C**). The window sizes have been adjusted to fit on the page.



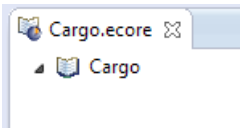
The Vocabulary window

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

The Vocabulary tree view

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

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



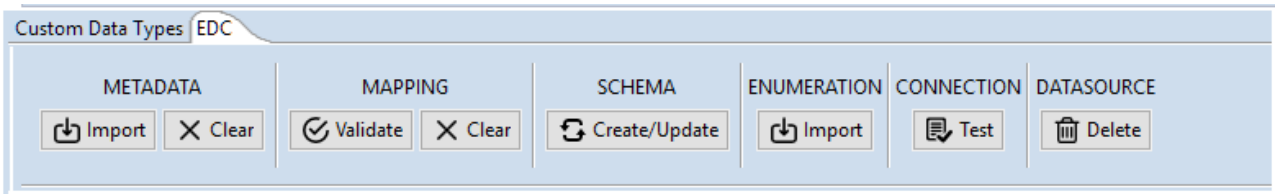
Vocabulary menu commands

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

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

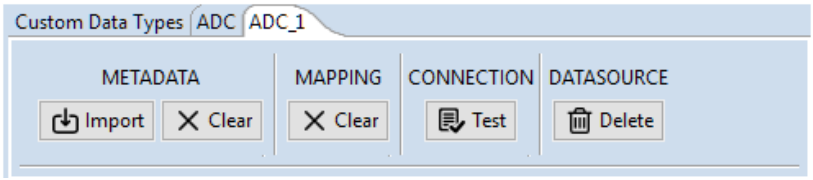
- **Add Domain** - Adds a Domain to the Vocabulary. Domains are discussed in the *Rule Modeling Guide*.
- **Add Entity** - Adds an Entity to the Vocabulary.
- **Add Attribute** - Lets you choose the basic data type and then add an Attribute to the selected Entity.
- **Add Association** - Adds an Association 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.

- **Add Datasource > Add EDC Datasource**



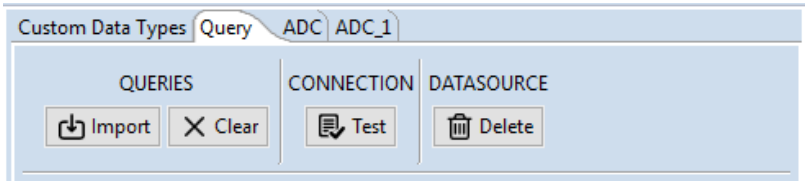
  - **METADATA: Import, Clear** - Gets/clears the schema definition from the database. See *"Importing EDC database metadata into a Vocabulary" in the Data Integration Guide.*
  - **MAPPING: Validate, Clear** - Validates/clears the mappings for this Datasource in the Vocabulary. See *"Importing EDC database metadata into a Vocabulary" in the Data Integration Guide.*
  - **SCHEMA: Create/Update** - Sets up the schema in the defined database. See *"Creating a schema in the database" in the Data Integration Guide.*
  - **ENUMERATION: Import** - Gets Custom Data Type values from the database. See *"Importing an attribute's possible values from database table" in the Data Integration Guide.*
  - **CONNECTION: Test** - Checks the defined connection. See *"Defining the database connection for EDC" in the Data Integration Guide.*
  - **DATASOURCE: Delete** - Removes this Datasource definition from the Vocabulary.

- **Add Datasource > Add ADC Datasource**



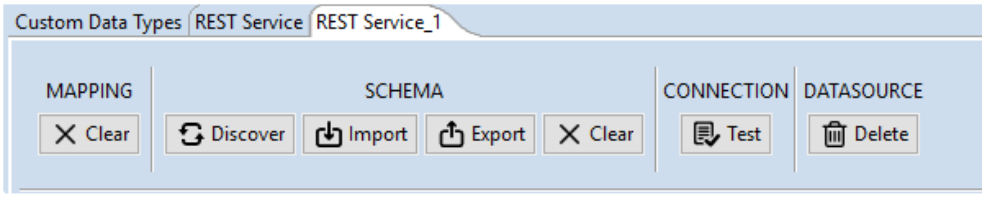
  - **METADATA: Import, Clear** - Gets/clears the schema definition from the database. See *"Importing ADC database metadata into a Vocabulary" in the Data Integration Guide.*
  - **MAPPING: Clear** - Clears the mappings for this Datasource in the Vocabulary. See *"Importing ADC database metadata into a Vocabulary" in the Data Integration Guide.*
  - **CONNECTION: Test** - Checks the defined connection. See *"Defining a database connection for ADC" in the Data Integration Guide.*
  - **DATASOURCE: Delete** - Removes this Datasource definition from the Vocabulary.

- **Add Datasource > Add Query Datasource**



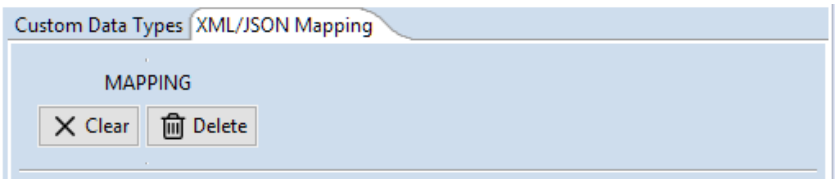
  - **QUERIES: Import, Clear** - Gets/clears the queries from the database that ADC will use, and then caches them locally. See *"Define a Query Datasource for ADC" in the Data Integration Guide.*
  - **CONNECTION: Test** - Checks the defined connection.
  - **DATASOURCE: Delete** - Removes this Datasource definition from the Vocabulary.

- **Add Datasource > Add REST Datasource**



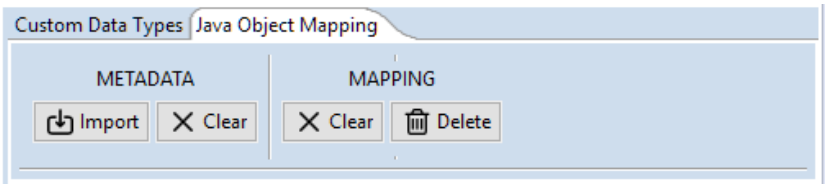
- **MAPPING: Clear***"Importing REST Datasource metadata into a Vocabulary" in the Data Integration Guide.*
- **SCHEMA: Discover, Import, Export, Clear** - Gets the schema definition from the Datasource.
  - *Discover*: Uses the given URL to query the REST Services driver to get the schema, stores it in the Vocabulary, and then uses it to import metadata from the Datasource.
  - *Import*: Opens the **Load REST Schema File** dialog to access a defined JSON schema file. The file is stored in the Vocabulary, and then used it to import metadata from the Datasource. See *"Importing REST Datasource metadata into a Vocabulary" in the Data Integration Guide.*
  - *Export*: Creates a JSON file with the schema at the specified location. Exports the Datasource configuration. See *"Exporting a Datasource configuration" in the Data Integration Guide.*
  - *Clear*: Clears the schema data and the database metadata.
- **CONNECTION: Test** - Checks the defined connection. See *"Define a Datasource connection for REST" in the Data Integration Guide.*
- **DATASOURCE: Delete** - Removes this Datasource definition from the Vocabulary.

- **Datasource Configuration File > Import Database Access Properties** - Imports an EDC configuration defined in a previous release. See the note in *"Define the database connection for EDC" in the Data Integration Guide.*
- **Add Document Mapping > Add XML/JSON Mapping**



  - **MAPPING: Clear** - Clear mappings that match naming convention of the elements in your XML/JSON payload. See *"XML/JSON Mapping" in the Deployment Guide.*
  - **MAPPING: Delete** - Clears and removes the XML/JSON Mapping panel.

- **Add Document Mapping > Add Java Object Mapping**

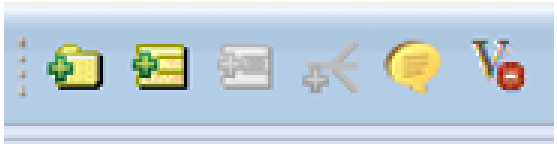


  - **METADATA: Import, Clear** - Configures your Vocabulary when the data payload is a map or collection of Java objects to use specified methods. See *"Java object mapping" in the Deployment Guide.*
  - **MAPPING: Clear, Delete** - Clear removes all the JOM mappings, Delete clears and removes the Java Object Mapping data panel.

- **Set to Read Only | Set to Read/Write** - Toggles the Vocabulary to limit it from making any changes and to free it to add/modify/delete elements, Datasource access, and custom data types.
- **Show Vocabulary Details |Hide Vocabulary Details** - Toggles the Vocabulary to show or hide details.
- **Localize** - Lets you set the Language parameter (Locale) for the Vocabulary and displays the Vocabulary's elements in a list.
- **Report** - Creates an HTML report and launches your browser for viewing. See Creating a Vocabulary Report.
- **Export WSDL** - Opens the **Export Vocabulary WSDL** dialog box for this Vocabulary. WSDL service contracts are discussed in the set of topics in the Deployment section, *Integrating Corticon Decision Services*.
- **Export XSD** - Opens the **Export Vocabulary XSD** dialog box for this Vocabulary. XSD service contracts are discussed in the set of topics in the Deployment section, *Integrating Corticon Decision Services*.

Vocabulary toolbar

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



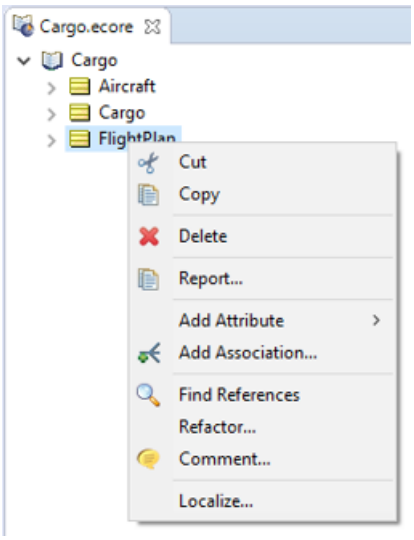
The Vocabulary tools provide the same functions as the corresponding **Vocabulary** menu commands:

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

Commands on the Vocabulary context-sensitive menu

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

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

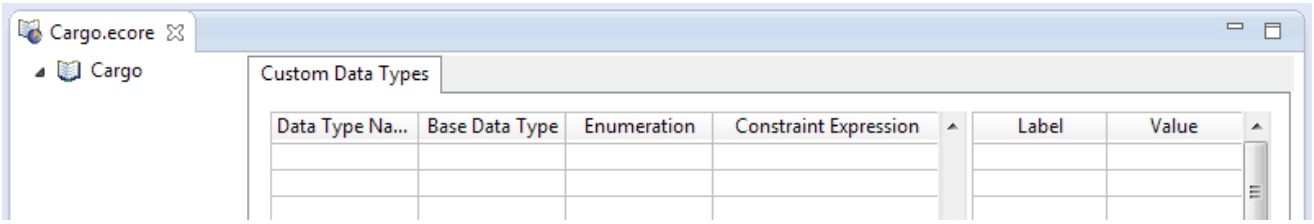


Important: The right-click pop-up menus are context-sensitive, meaning not all options are available for all Vocabulary nodes. For example, the “Add Attribute” option is only available when an Entity is selected in the Vocabulary tree.

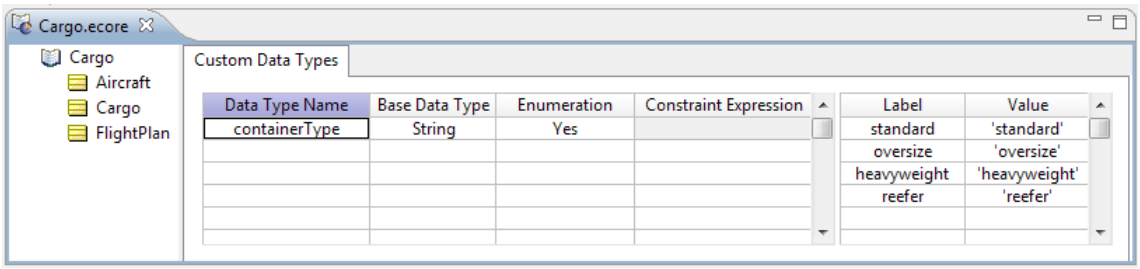
Populating a new Vocabulary

Custom data types tab

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

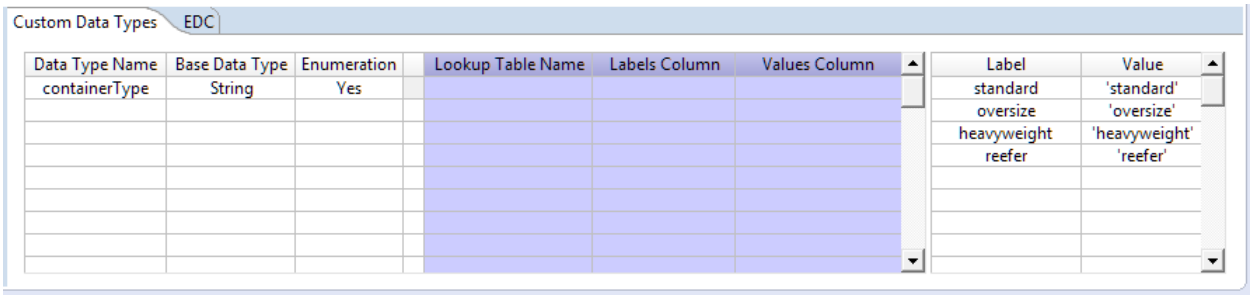


You can define lists of values that are the set of allowable values associated with a Vocabulary attribute. The Tutorial sample demonstrates how to delimit the options for a `containerType` by defining labels and their respective values:



Importing enumerated values from a Datasource

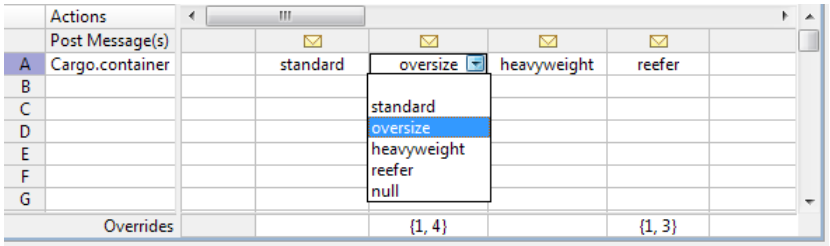
When the Vocabulary connects to a Datasource, additional functionality is added to the **Custom DataTypes** tab:



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

How enumerated values function in a Rulesheet

In a Rulesheet, the defined values for attributes that are assigned the custom data type are offered a list of its specified values -- as well as `null` and blank.



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

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

Datasource tabs

When you choose to define connections to data sources, each instance is defined as a uniquely named Corticon Datasource on a tab of the Vocabulary in its editor. The techniques for Datasource connectivity are:

- **Enterprise Data Connector (EDC) Datasource** - EDC accesses one database per Vocabulary, and enables Corticon to create queries to retrieve and write data as needed. This technique makes for easy data access, and is a good option when modest amounts of data will be accessed with reasonable performance. For more information, see *"Getting Started with EDC" in the Data Integration Guide.*
- **Advanced Data Connector (ADC) Datasource** - One or more ADC connections can be defined, each accessing your store of SQL queries to retrieve and update data in the connected database. ADC is best for requirements where query performance when retrieving large amounts of data is crucial. For more information, see *"Getting Started with ADC" in the Data Integration Guide.*
- **Query Datasource** - The Query Datasource is used to retrieve query names from `CORTICON_ADC_READ` and `CORTICON_ADC_WRITE` tables to be used when defining an ADC Service Callout in a Ruleflow. Exactly

one query Datasource is required for an ADC-enabled Vocabulary. For more information, see *"Getting Started with ADC" in the Data Integration Guide.*

- **REST Datasource** - One or more REST Datasources can be defined, each accessing its own specified connection URL. For more information, see *"Getting Started with REST Services" in the Data Integration Guide.*

Importing Datasource configurations

An alternative to adding data source definitions to your vocabulary is to import the definitions from a Datasource configuration file. This approach allows IT or another person with knowledge of the data sources to define them and make them easy to add to a Vocabulary. To import a Datasource configuration file, do the following:

1. Select **Vocabulary > Datasource Configuration File > Import** from the Vocabulary main menu.
2. Browse to locate the existing Datasource configuration file in the directory structure, and select it to highlight the file name.
3. Click **OK**. The Datasource definitions are added to your Vocabulary.

Note: If you update existing Datasource definitions by importing definitions from a Datasource configuration file, your existing Datasource connection parameters are updated but not your existing Vocabulary mapping.

Adding nodes to the Vocabulary tree view

Vocabulary node naming restrictions

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

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

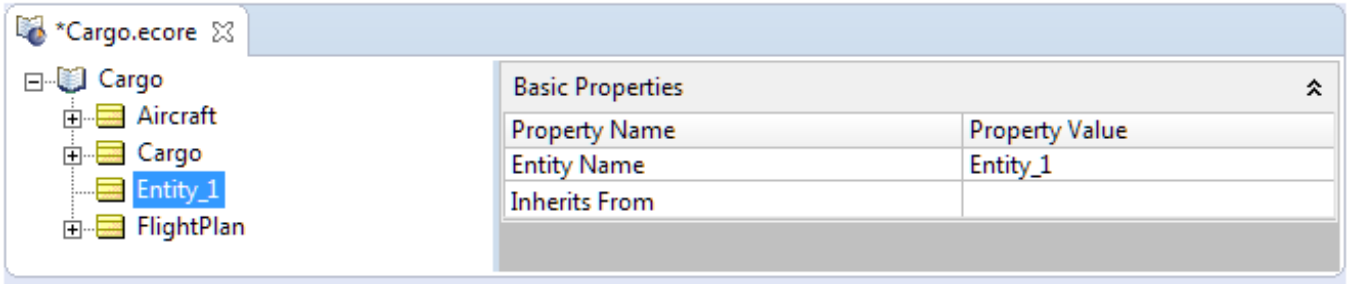
Adding and editing domain nodes and their properties

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

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

Adding and editing 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:



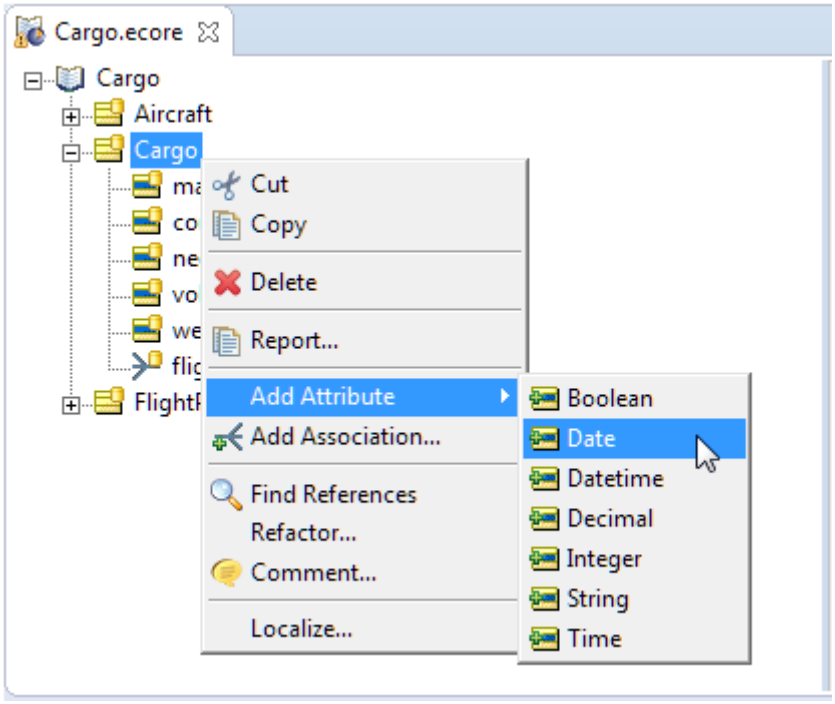
4. Assign property values as listed in the Basic Entity Properties table:

Table 1: Entity: Basic Properties

Property	Value
Entity Name	Assign a name to the entity. By default, this will be pre-filled as `Entity_x`, where `x` 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	Optional. The Vocabulary model contains extensive support for inheritance concepts. For more information on using this feature, refer to the *Rule Modeling Guide*, “Creating the Vocabulary” chapter.

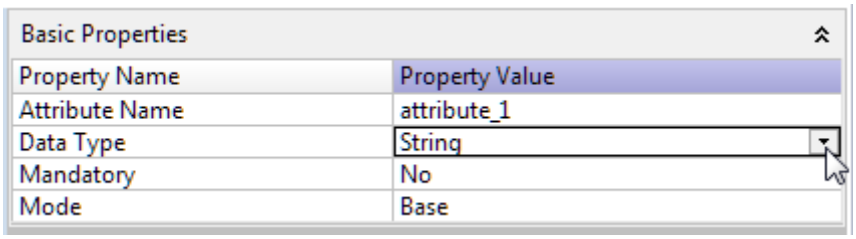
Adding and editing attribute nodes and their properties

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 Date attribute:

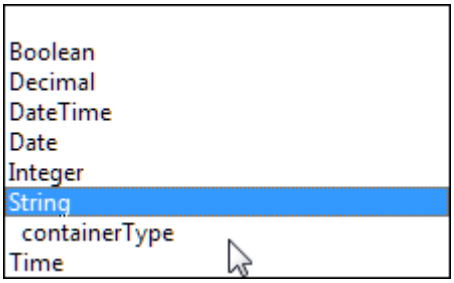


(You could instead choose the menu command **Vocabulary > Add Attribute**.) Note that the toolbar button  always creates a String attribute.

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



...and then select the defined Custom Data Type you want to use for this attribute, as illustrated:



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

Note:

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

Table 2: 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, Date, Integer and Time. You can also have a custom data type. All types are described in the *Rule Language Guide* and in the *Rule Modeling Guide's* “Building the Vocabulary” chapter.
Mandatory	A mandatory attribute cannot have a value of null. This setting affects the members of the values sets shown in Rulesheet drop-downs. For example, an attribute whose Mandatory value is `No` will always include a `null` value selection in its Rulesheet drop-downs.
Mode	Choose the attribute's Mode from the drop-down list. `Base` attributes exist or are used by systems outside Corticon, and are included in the XML schemas and contracts . `Base` attributes map directly to an element in the XML CorticonRequest/Response documents or object properties processed by the Server in production. `Transient` attributes are *derived* fields; they exist only during rule execution and are not part of XML schemas. They do not need to be included in the XML CorticonRequest documents or objects, and they are not included in the XML CorticonResponse documents or objects produced by the Server in deployment. 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. **Note:** If you export tests that use transients to XML, and then run them on a deployment Server, the transients are ignored.

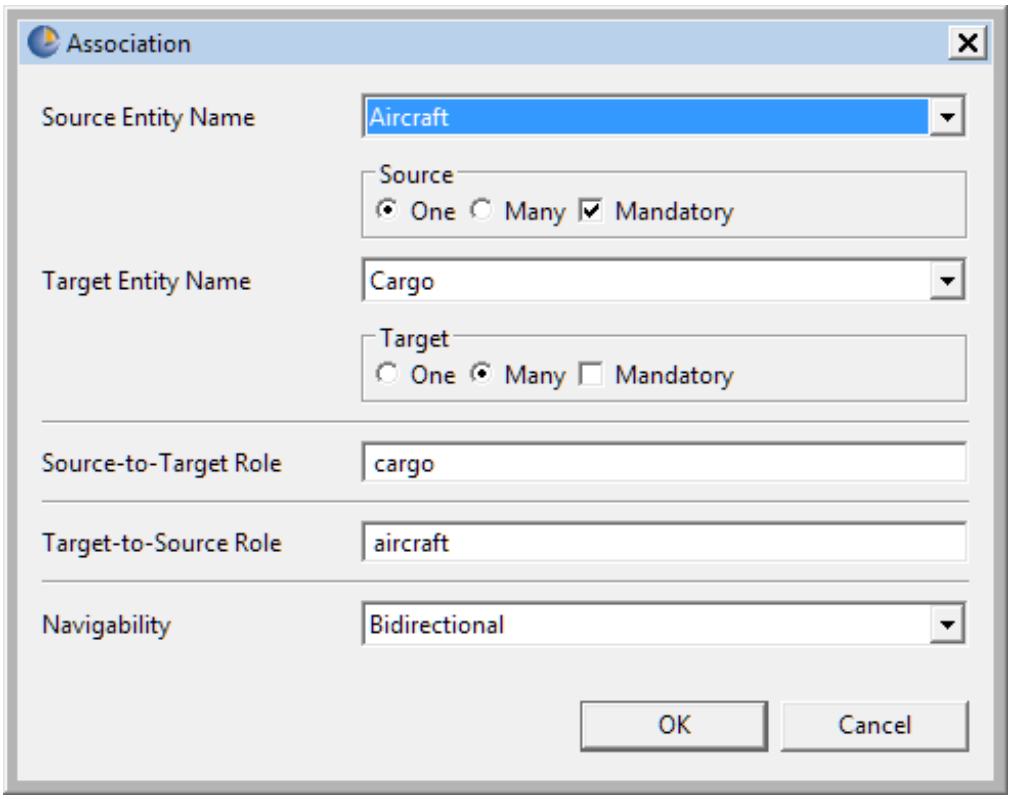
Adding and editing association nodes and their properties

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.

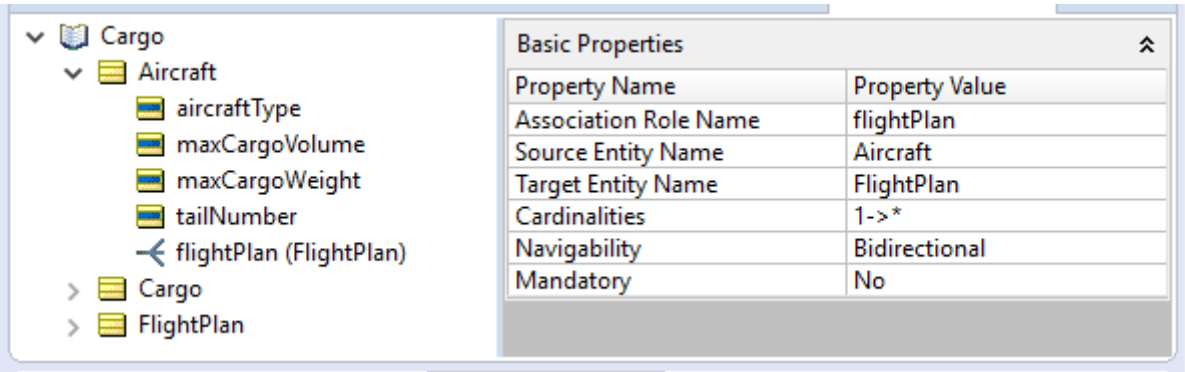
3. Complete the Association window as shown in the following table:



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

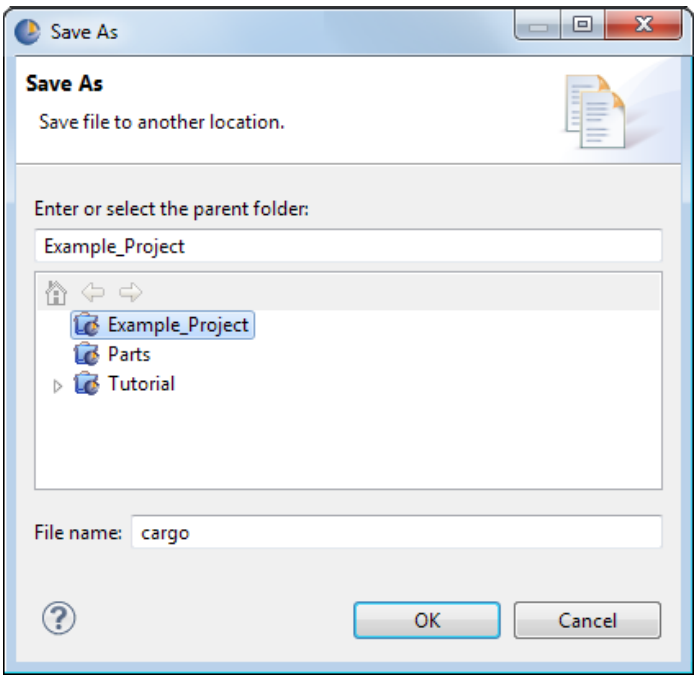
Property	Value
Cardinality Radio Buttons	Select the combination of radio buttons that describe the relationship you want between the two entities. The possible combinations are:
	One-To-Many. This is the default and is shown as `1>*` in the properties. This means that a given instance of the supplier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

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



Saving a new Vocabulary

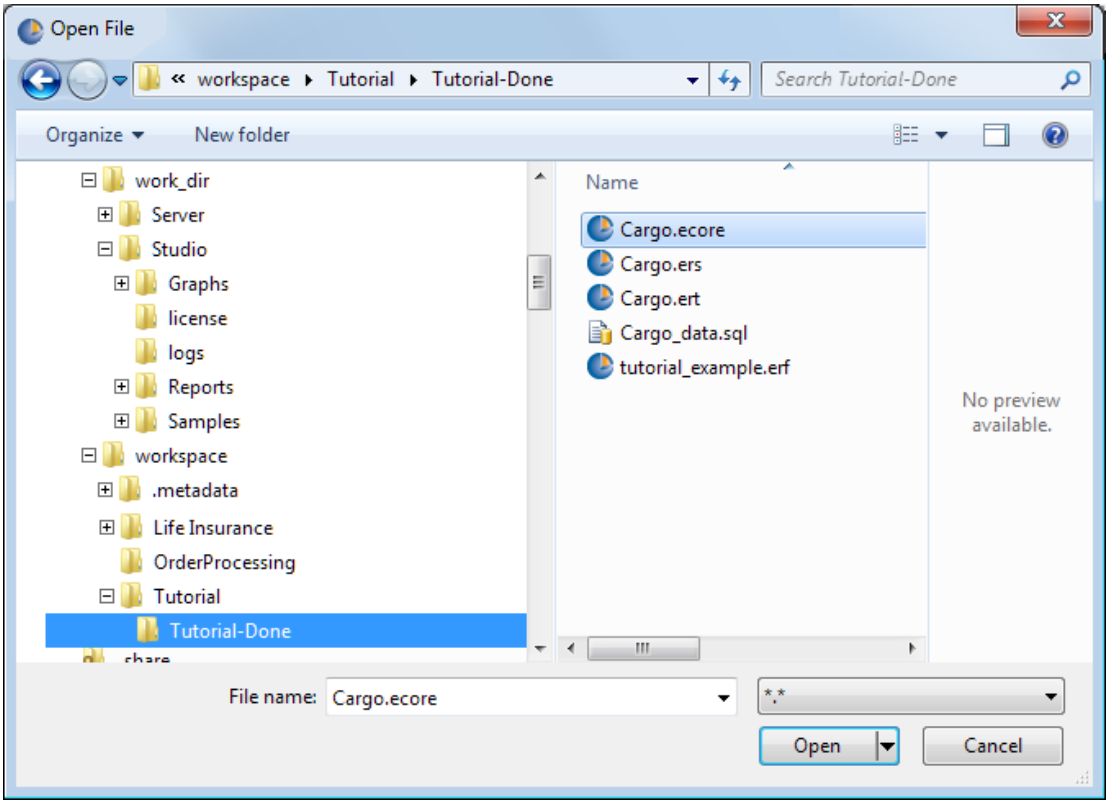
1. Do one of the following:
   - Save the Vocabulary from the toolbar or choose **File** > **Save** from the menubar to save any changes you have made.
   - Use **File** > **Save As** on the menubar and Navigate to the directory where you want to store the Vocabulary, or click to create a new directory.



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

Opening an existing Vocabulary

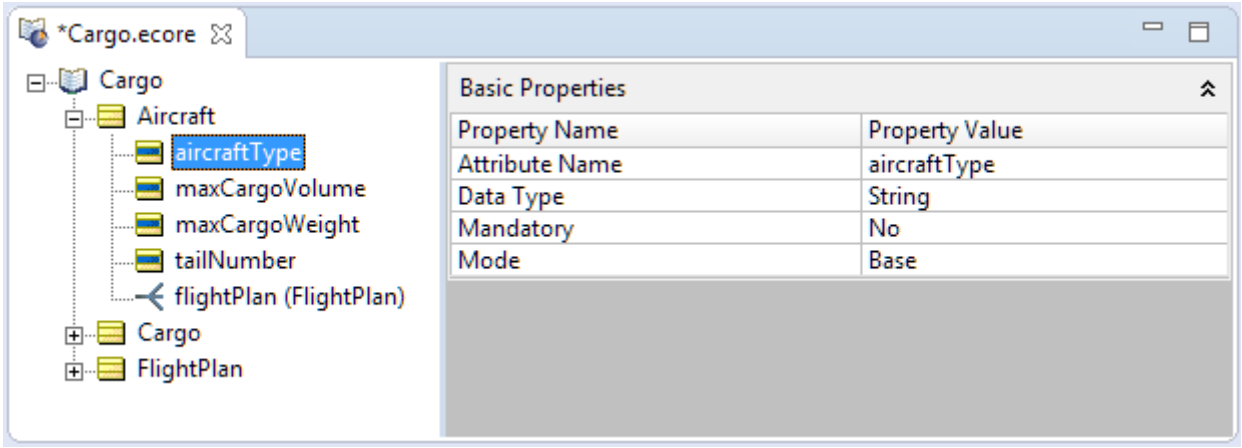
1. Choose **File** > **Open File** from the Corticon Studio menubar or use the toolbar to display the **Open** window.



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

Modifying a Vocabulary

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



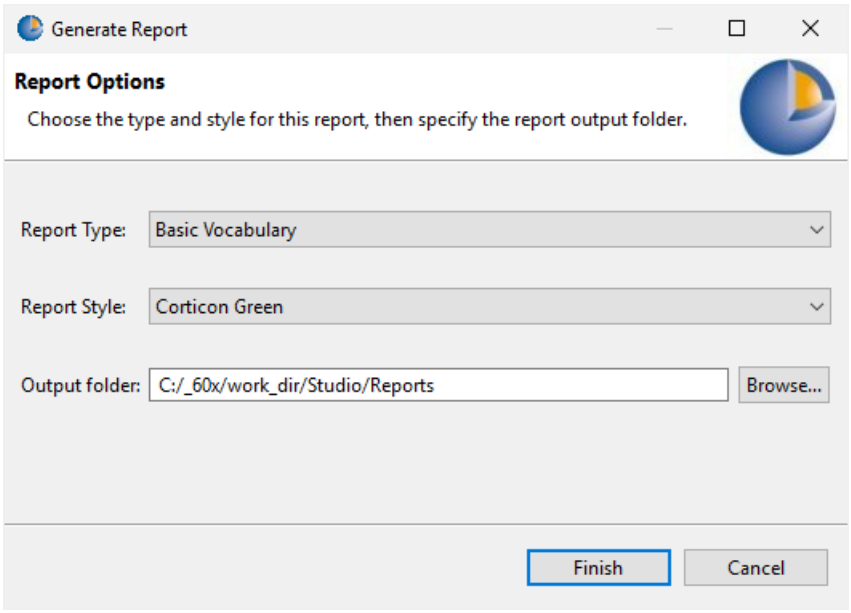
3. Modify the properties with the following considerations:.

- 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.

Creating 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, as shown:



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.

For information about creating custom transformations and stylesheets for reports, see the topic *Reporting Framework chapter of the Rule Modeling Guide.*

Creating Vocabularies for OpenEdge

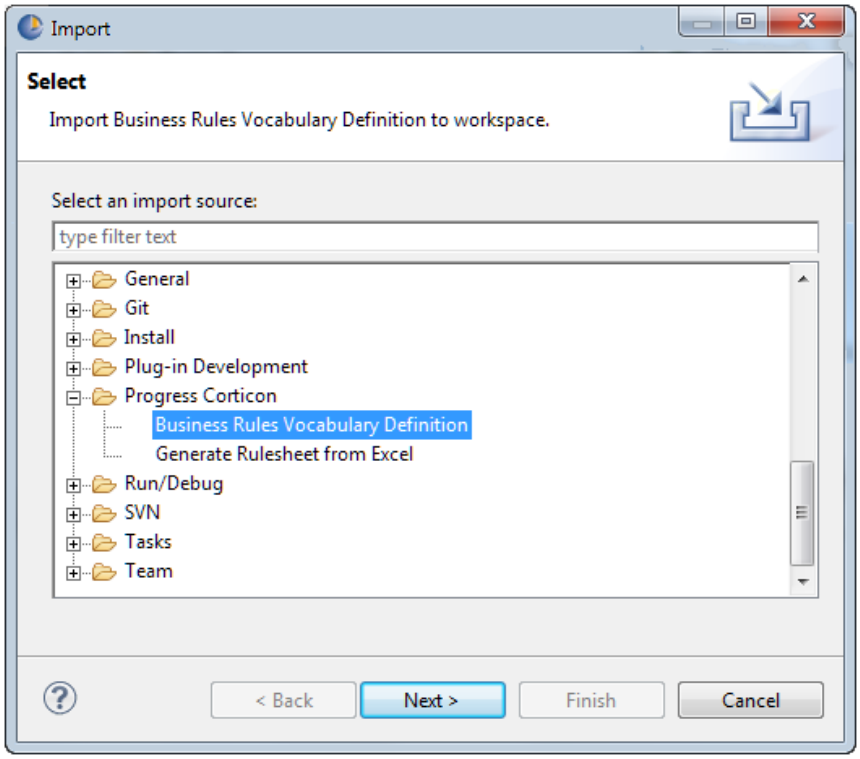
Importing an OpenEdge Business Rules Vocabulary Definition file

OpenEdge developers can use Corticon for their business rules, using Progress Developer Studio to integrate their ABL projects with Corticon Decision Services. A schema exported from Progress OpenEdge can be imported and used as the basis for Vocabulary entities and attributes in Corticon Studio.

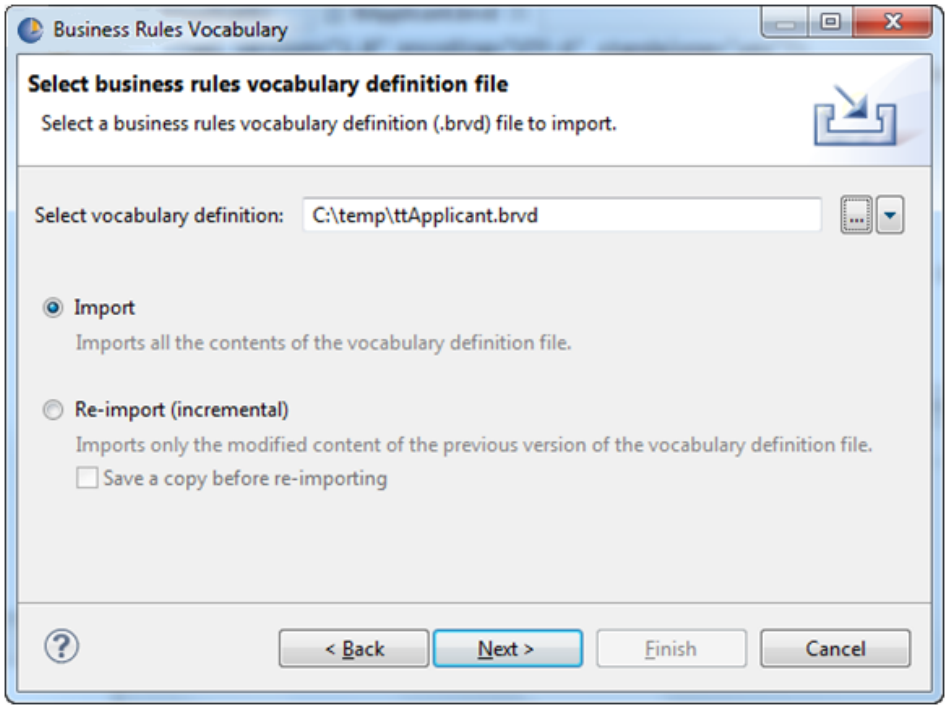
To import a Vocabulary definition created in OpenEdge into the Corticon perspective:

1. In the integrated OpenEdge/Corticon Eclipse development environment, choose the menu command **Window > Open Perspective > Corticon Designer**.
2. Choose the menu command **File > New > Rule Project**, name the project -- in this example, `CorticonProject` -- and then click **Finish**.
3. Choose the menu command **File > New > Progress Corticon > Rule Project**.

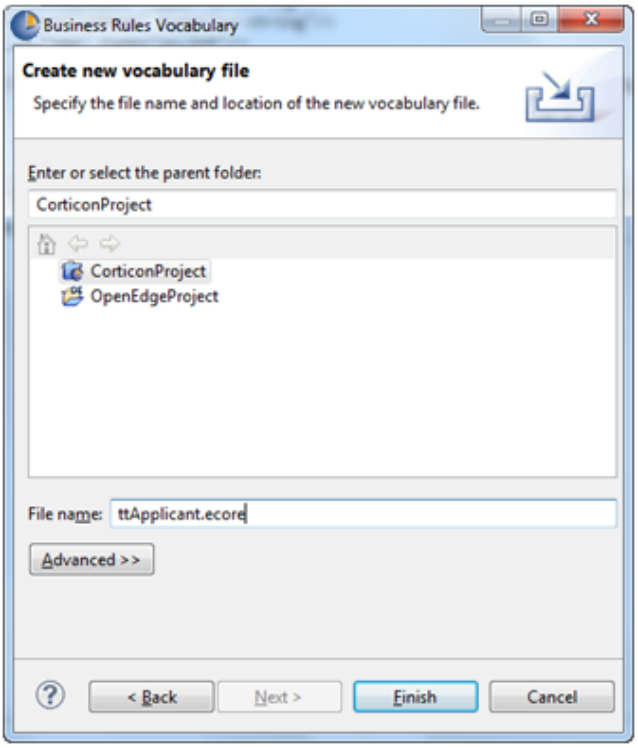
4. In the **New Corticon Project** wizard, specify the name and location of the project.
5. Optionally, add the project to working sets and selected project references.
6. Click **Finish**. The new project is created and displayed in the **Project Explorer** view.
7. Click on the project name, and then choose the menu command **File > Import > Import**
8. In the **Import** dialog, expand **Progress Corticon**, and then click on **Business Rules Vocabulary Definition**, as shown:



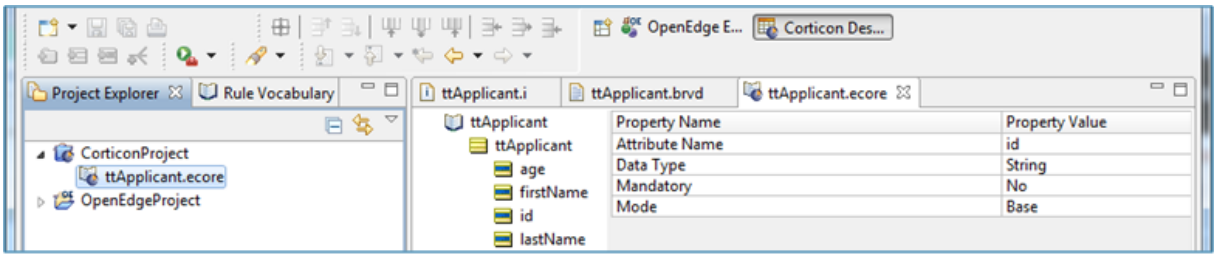
9. In the **Business Rules Vocabulary** dialog, locate the `.brvd` file -- in this example, `ttApplicant.brvd`, that was staged in the `c:\temp` folder -- that was created in OpenEdge.
10. Select **Import**, as shown:



11. Click **Next**.
12. Select the **CorticonProject**, and then enter the Vocabulary **File name** -- in this example, `ttApplicant.ecore`, as shown. (The name must have the `.ecore` extension.)



13. Click **Finish**.
14. Double-click on the `.ecore` file name in the Project Explorer to open it in the Corticon Vocabulary editor. The example looks like this:



15. Review the Vocabulary to ensure that it represents the exported data correctly.

The import processing of the OpenEdge BRVD file into a Corticon Vocabulary is complete.

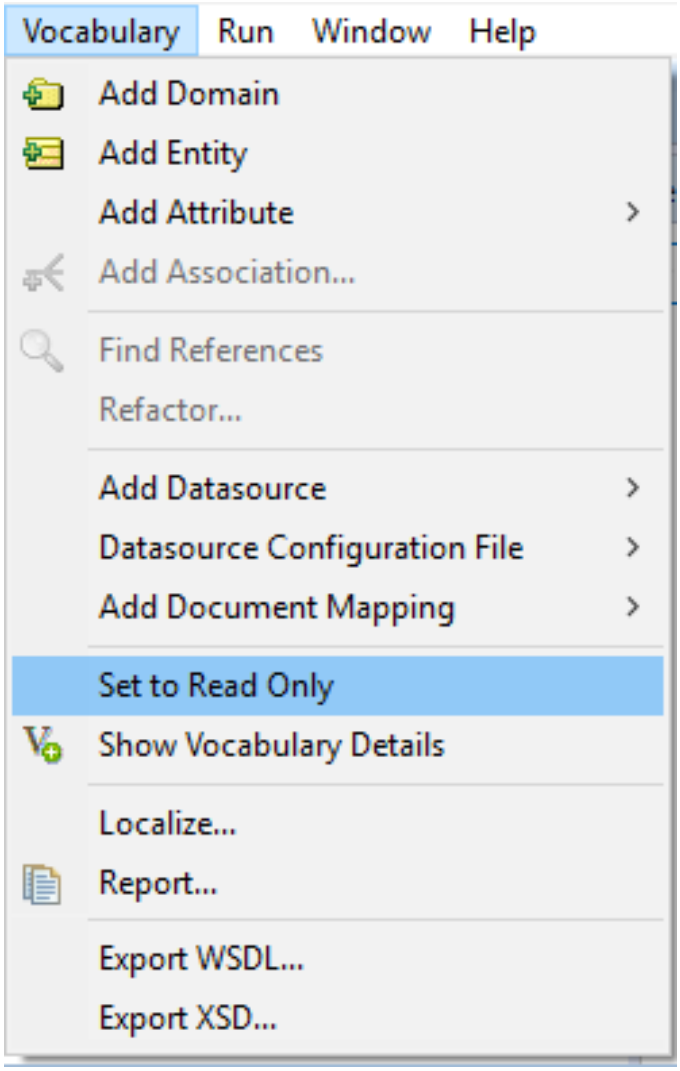
You can now create and test Rulesheets and Ruleflows, and then publish a Decision Service to Corticon Server (such as the one that runs in an OpenEdge Web Server).

Locking and unlocking Corticon Vocabularies

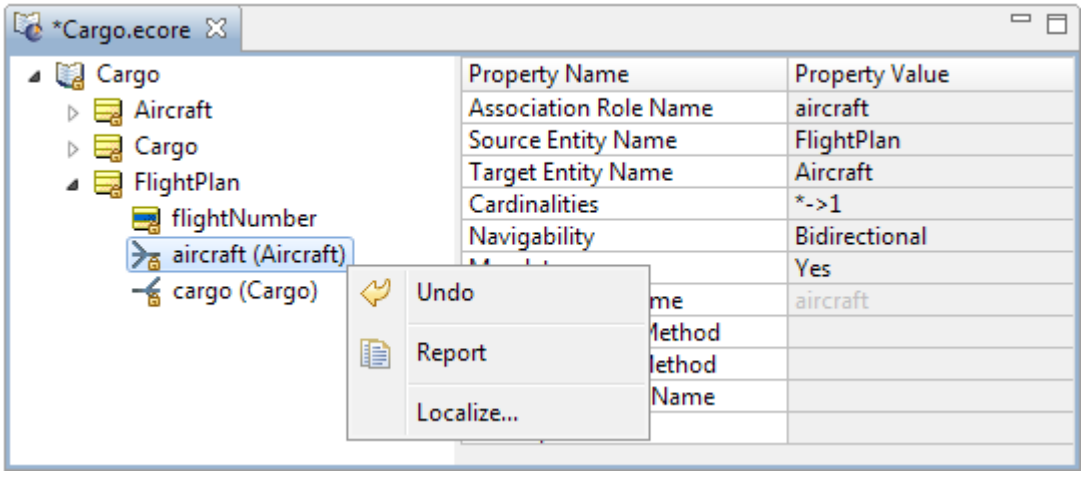
There are advantages to locked and unlocked Vocabularies:

- When your rule development has many Rulesheets and Ruletests, you might want to add a measure of control over Vocabulary changes. Locking the Vocabulary prevents you from accidentally changing an entity or attribute, thereby invalidating rules and calls from databases.
- When OpenEdge developers use Corticon for their business rules, they use Progress Developer Studio to integrate their ABL projects with Corticon Decision Services. During import of a Business Rules Vocabulary Definition (BRVD) created in Open Edge, a Corticon mechanism flags vocabulary entities, attributes, and associations as read-only. This protection keeps you from accidentally modifying the Vocabulary such that it no longer matches its source in OpenEdge. If you want to define transient attributes, constraint expressions, or make other modifications to the vocabulary in Corticon you will need to unlock it. When doing so, you need to be sure that your modifications do not make the vocabulary incompatible with its source in OpenEdge.

You can choose to lock a Vocabulary in Studio by selecting the **Set to Read Only** option, as shown:

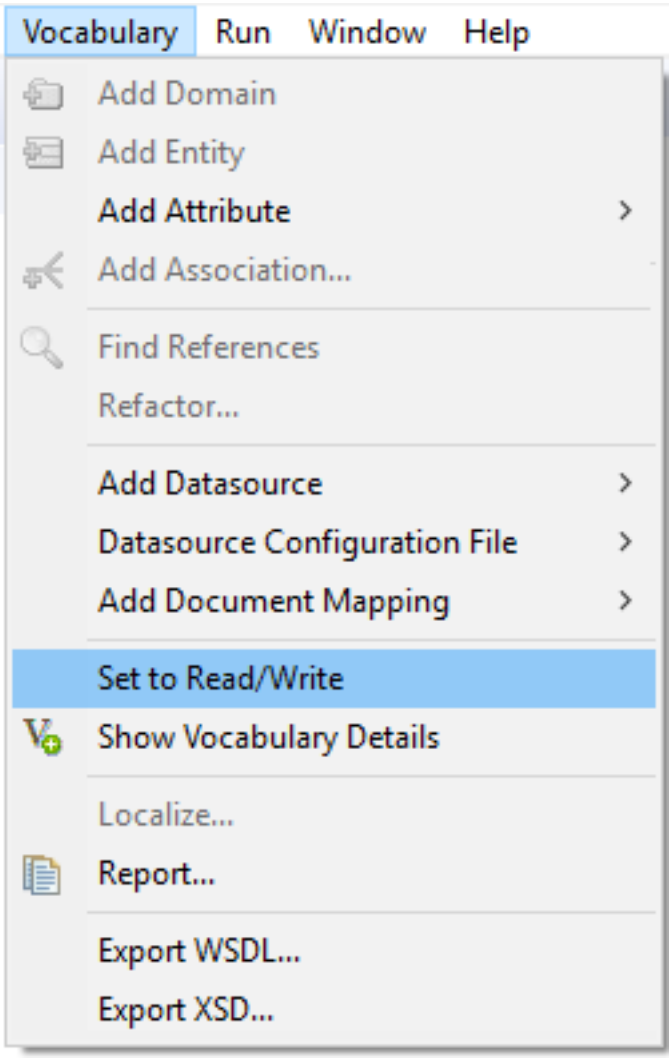


Once locked, all the icons in the Vocabulary display a padlock symbol.

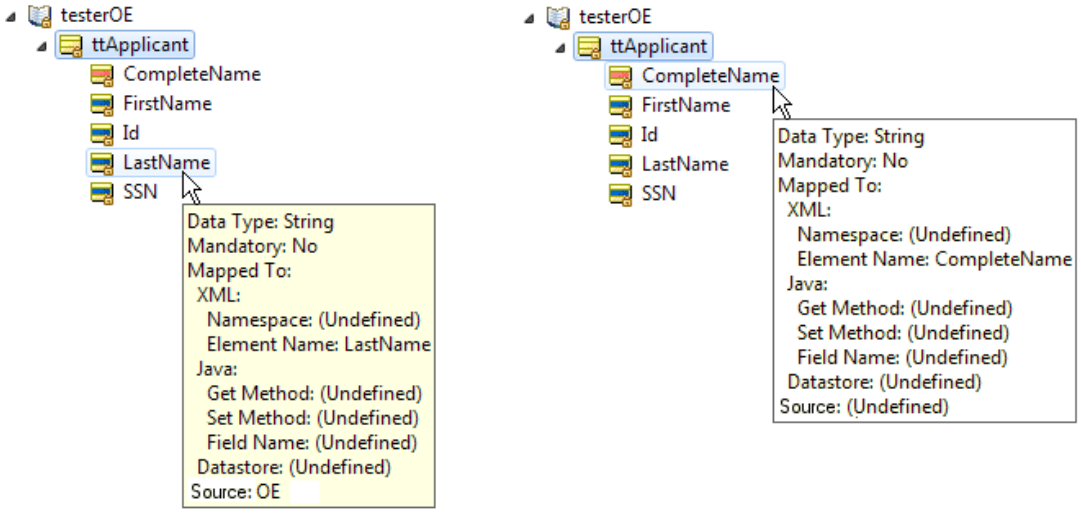


In a locked Vocabulary, all functions in the Vocabulary editor are display-only, including Custom Data Types and Database Access. **Undo** (and **Redo**) options will toggle the mode until the Vocabulary file is saved.

A locked Vocabulary can be unlocked by selecting the **Set to Read/Write** option.



Note: In Vocabularies created through BRVD import, changes that you make when unlocked and then locked again, continue to reflect the OE origin of BRVD imported attributes and entities.



You should never rename attributes or entities that are bound to OpenEdge or EDC resources as they would likely cause data synchronization problems.

Applying an updated BRVD to a Vocabulary

If you make changes to your OpenEdge application, you might need to regenerate the BRVD file, and then re-import it into Corticon to update your Vocabulary.

A re-imported BRVD file impacts any changes you might have made if you set the imported Vocabulary into Read/Write mode. While you cannot change the Vocabulary when it is in Read Only mode, the re-import actions will proceed on a locked or unlocked Vocabulary.

Note: A re-import of a BRVD file changes the current Vocabulary. It is a good practice to choose **Save a copy before re-importing** in the import dialog to backup the existing Vocabulary before applying changes to it, especially if you entered Read/Write mode and made changes.

Before the re-import action is applied, a dialog displays the entities, attributes, and associations in the Vocabulary and how they will be impacted.

The status and intended actions on Vocabulary entities, attributes, and associations during a re-import can be the following:

- **Match**- No action if the imported element and the existing element are identical in the following ways:
  - Entity: Same name and same origination (BRVD import or user-defined action)
  - Attribute: Same name, same datatype, and same origination (BRVD import or user-defined action)
  - Association: Same name, same target entity, and same cardinality
- **UserDefined** - No action if the existing element was created in Corticon.
- **Add** - The imported element is entered into the Vocabulary and marked as originating from the BRVD as follows:
  - Entity: The entity name in the BRVD does not match an existing entity in the Vocabulary

  - Attribute: The attribute name in the BRVD does not match an existing attribute in the corresponding entity in the Vocabulary.
  - Association: The association name in the BRVD does not match an existing association in the Vocabulary

- **Remove** - Deletes the existing element if it was previously created from a BRVD element, but is not in the import file.
- **Remove/Re-add** - Removes the existing element and then recreates it from its definition in the import file, under the following conditions:
  - Attribute: Same name but the datatype is different. (When the datatype is an enumerated list or constraint expression, if that custom data type has the same base data type as the imported attribute in the BRVD, it is a Match.)
  - Association: Same name but with a different Target Entity.

  Note: The Remove/Re-Add action does not apply to an Entity that originated from a BRVD. A re-import can either Add or Remove a BRVD originated entity.

- **Merge** - Revises an existing element marked as created locally that is now a BRVD element in the import file. The existing element will transform to be defined as an imported element (originated from BRVD) and marked as Merge, as follows:
  - Entity: Same name, but created in the Vocabulary editor.
  - Attribute: Same name, same datatype, but created in the Vocabulary editor. If the existing element was defined as transient mode, it is changed to base mode.
  - Association: Same name, same target entity, but different cardinality or created in the Vocabulary editor.

The dialog lets you choose to **Copy to Clipboard** to retain the action list as this information will not be available once you proceed to apply the re-import actions.

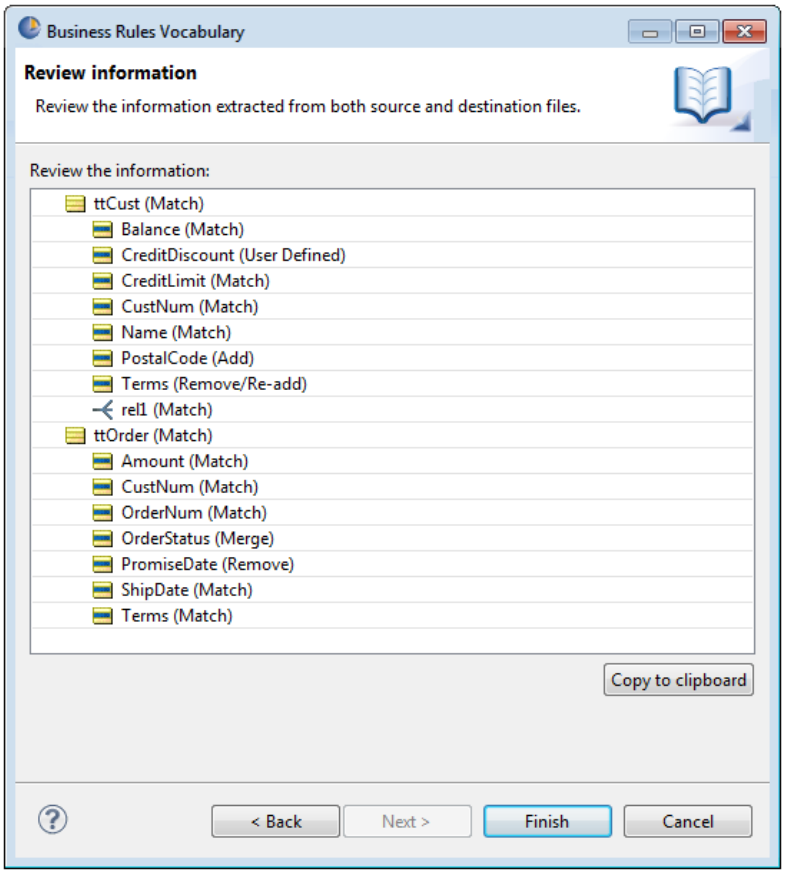
Note: If you choose to **Cancel** at this point, nothing was changed.

See the Progress OpenEdge documentation for details about a complete end-to-end workflow involved in an integrated OpenEdge Business Rules environment.

Example

The following figure shows how some BRVD changes to attributes are displayed for review:

Figure 1: Review information before re-import is applied.

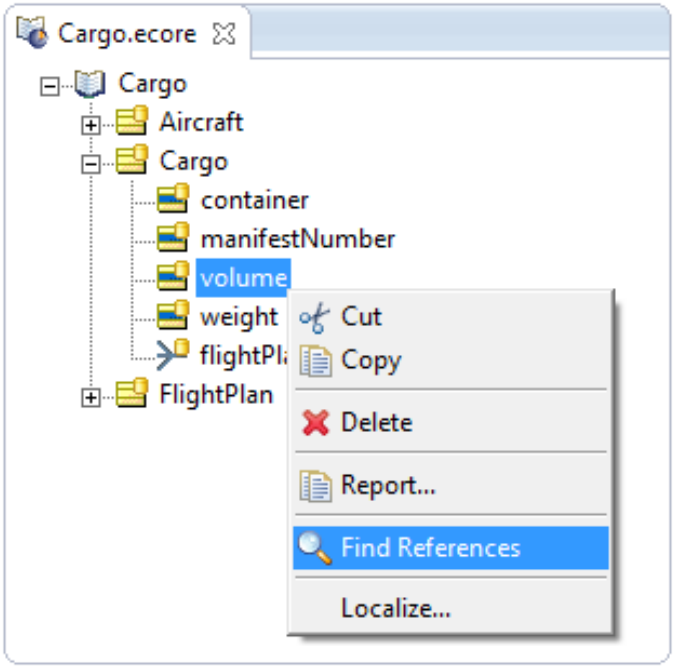


The revised BRVD file is being re-imported into a Vocabulary previously created from a BRVD. The following attribute changes are shown in the Review Information dialog above:

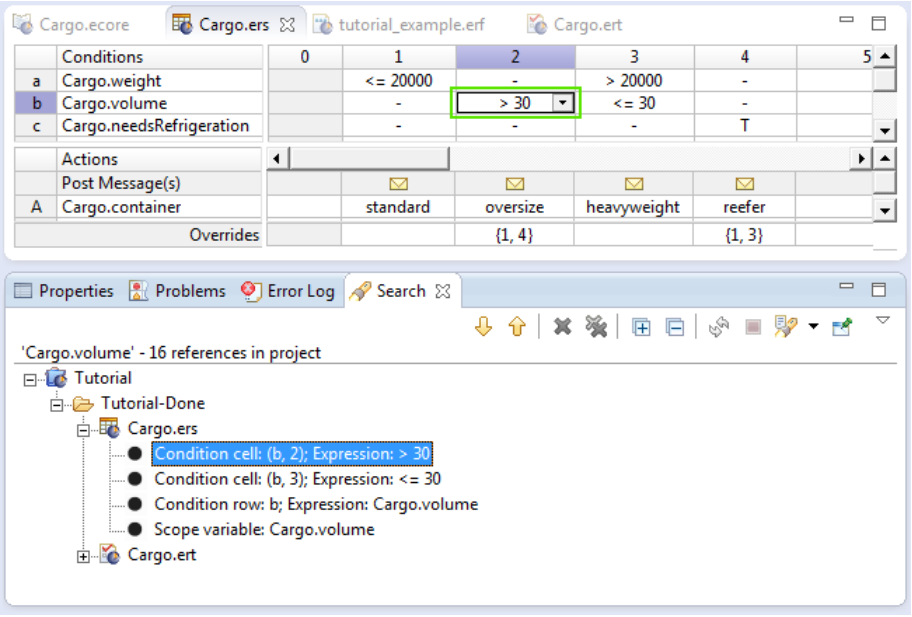
- **Match** – Unchanged attributes that originated in OpenEdge.
- **User Defined** – Added `CreditDiscount` only in Corticon.
- **Add** – `Postal code` added in OpenEdge and then added into the latest BRVD file.
- **Remove/Re-add** – Changed `Terms` to an `integer` only in Corticon, but it is still a `decimal` in OpenEdge and the BRVD. After this action, `Terms` is a `decimal`.
- **Merge** – Added `OrderStatus` to Corticon while awaiting BRVD change from OpenEdge. The new BRVD has `OrderStatus` so it is a merge
- **Remove** - Dropped the `Promise date` in OpenEdge from the latest BRVD file.

Finding references to an entity, attribute, or association in a project

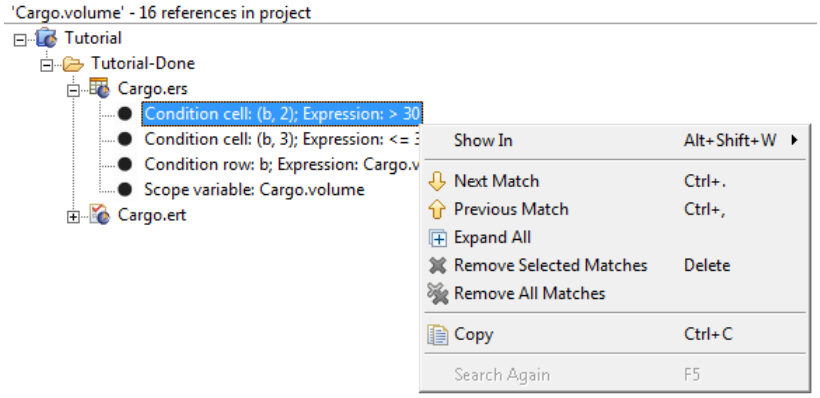
When working on large projects with many assets, it is a real convenience to find everywhere an entity, attribute, or association is referenced in the project. This feature reveals the vocabulary item's usage and impact, and lets you easily navigate to each instance. Just right-click on an entity, attribute,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



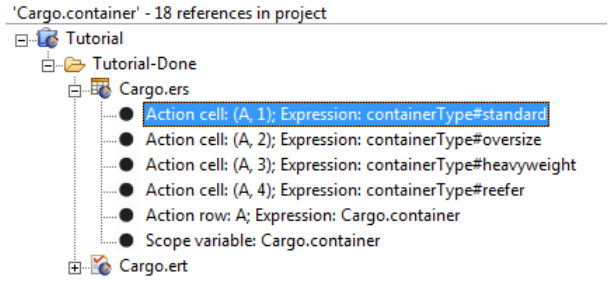
The search examines all Rulesheets, Ruleflows, and Ruletests in the project, then lists each match within each asset. Double-clicking on a match opens the asset and puts the focus at that location, as shown:



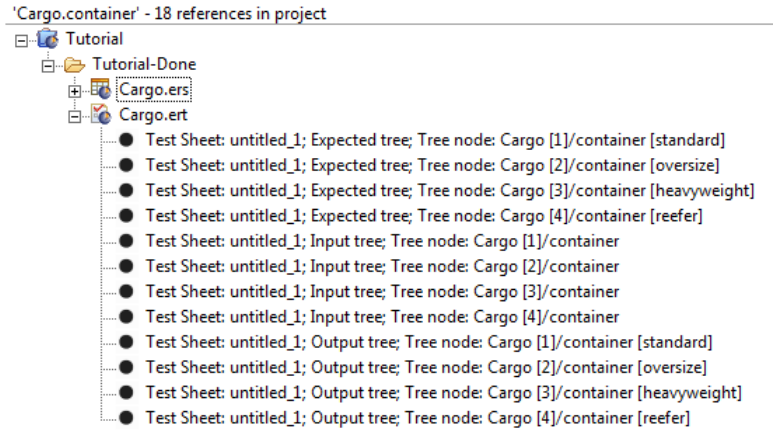
Right-clicking on a match presents a menu of navigation options as well as their keyboard shortcuts, as shown:



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:



Refactoring 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.

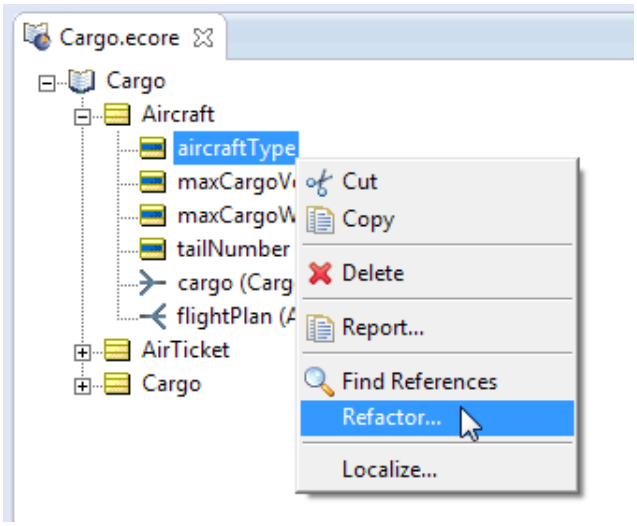
To avoid time consuming manual work and automate a potentially tedious process, Corticon lets you *refactor* the name of an entity, attribute, or association so that each change is automatically applied to all instances of the name in all assets in the project.

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.

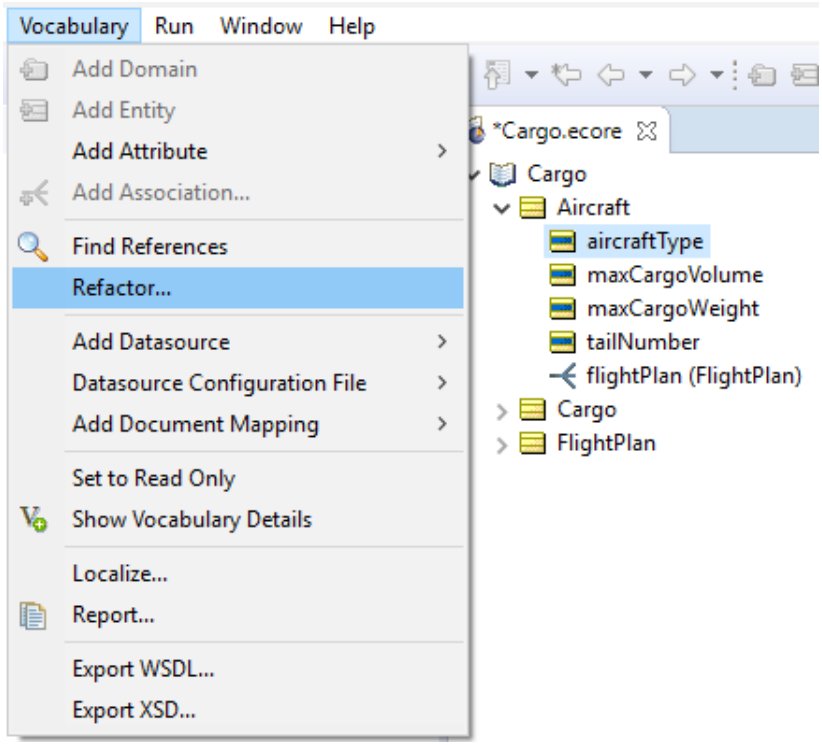
Note: Backup the project's rule assets before refactoring the Vocabulary since refactoring modifies and automatically saves rule assets.

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

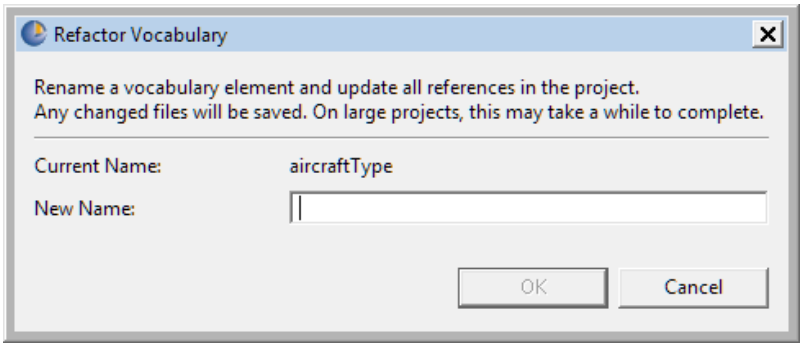
1. Open the project's Vocabulary.
2. Click on the name you want to change and highlight it.
3. Either right-click on the item to choose **Refactor** in the context menu:



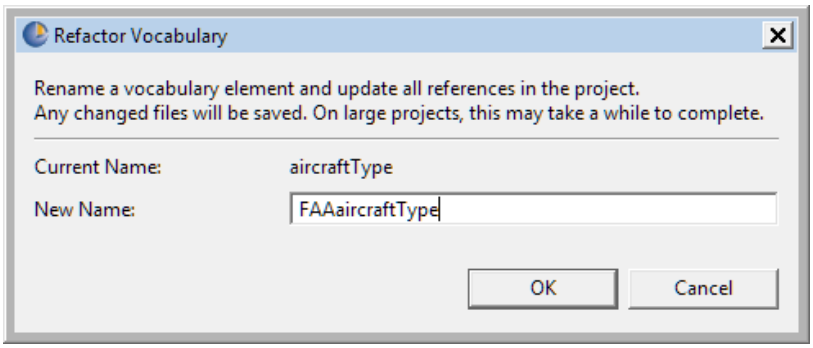
or choose the menu action **Vocabulary > Refactor**:



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.

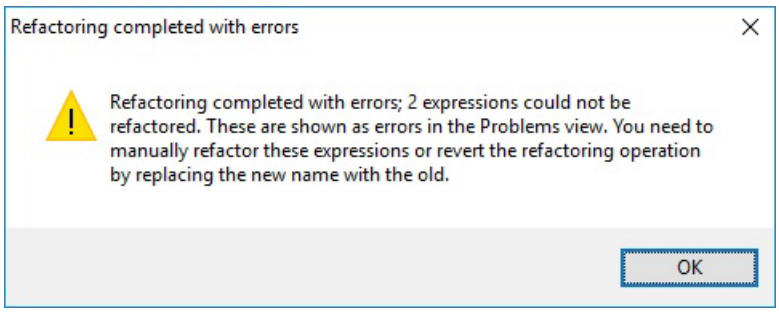
Important: Expressions that cannot be refactored

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.
- Inherited attributes and associations. For more information about inheritance, see *"Support for Inheritance" in the Rule Modeling Guide*.

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.

4

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 as a Decision Service, and deployed into a production environment (described in the *Deployment Guide*).

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

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

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

A Rulesheet file has the suffix `.ers`.

Rulesheet Window

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

For details, see the following topics:

- Creating a new Rulesheet
- Rulesheet menu commands
- Rulesheet toolbar

- Hover help in a Rulesheet
- Commands on the Rulesheet context-sensitive menu
- Rulesheet sections
- Navigation in a Rulesheet
- Rule statements window
- Rulesheet properties
- Using the business vocabulary to build rules
- Using the operator vocabulary to build rules
- Naming Rulesheets
- Deleting Rulesheets
- Saving a new Rulesheet
- Validating and optimizing a Rulesheet
- Creating a Rulesheet report
- Closing a Rulesheet
- Saving a modified Rulesheet
- Creating rules by importing an Excel worksheet

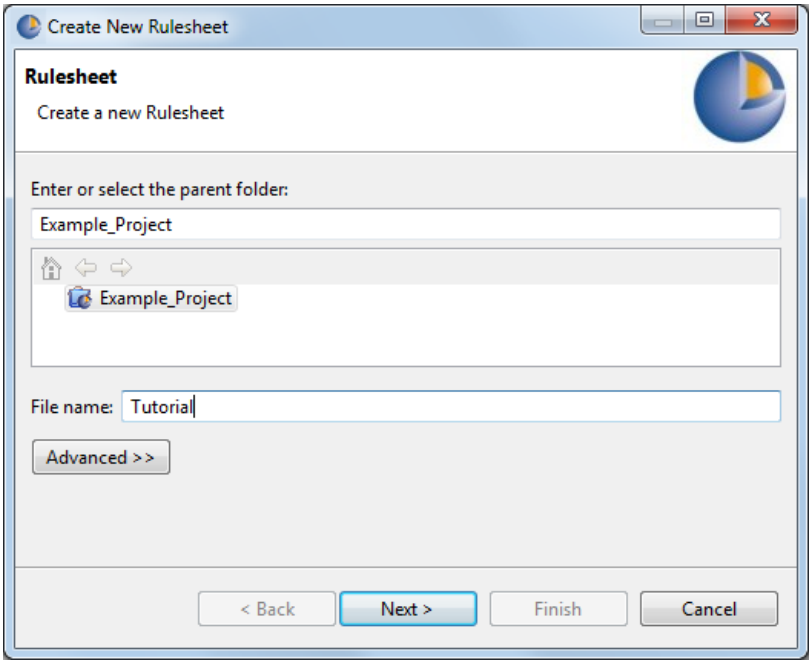
Creating a new Rulesheet

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

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

   OR

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

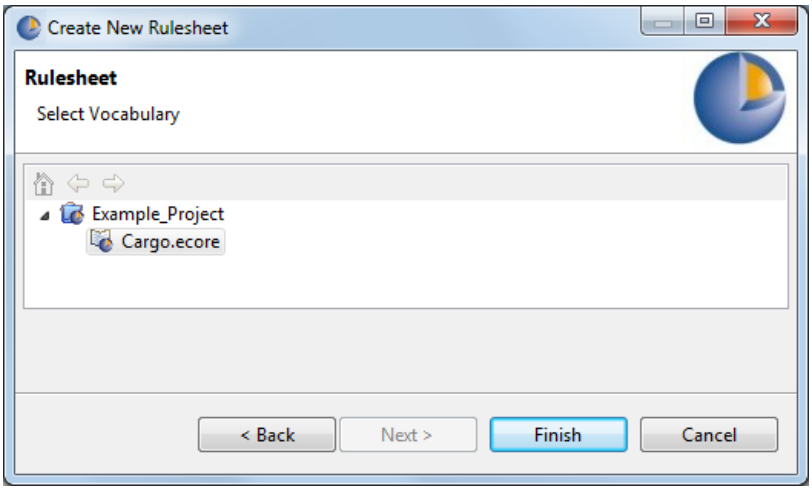


3. Highlight the Rule Project you would like to associate the new Rulesheet with in the list (or enter it manually in the **Enter or select parent folder** entry area.
4. Enter a file name for the Rulesheet in the **File name:** entry area.

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

5. Click **Next** to continue.

Figure 2: The Create New Rulesheet Wizard window



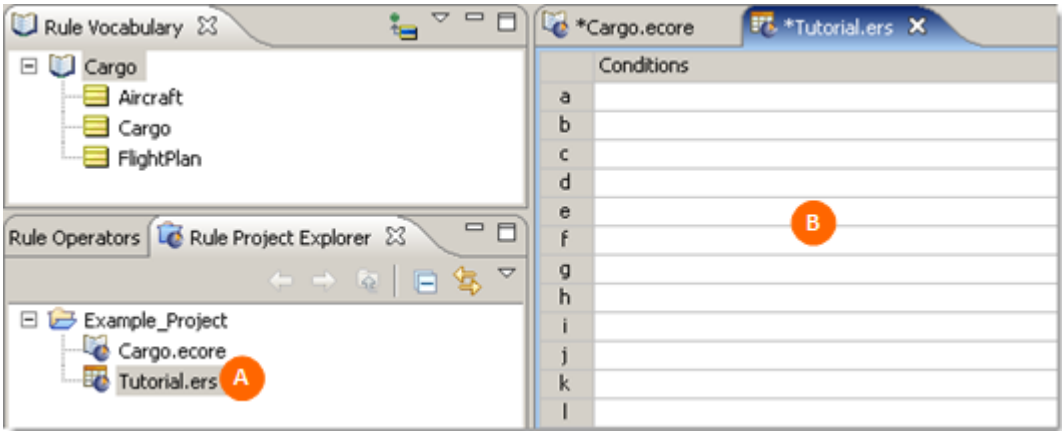
Note: The dialog only lists the Vocabularies for the parent project. This prevents you from creating cross-project assets. Do not create cross-project assets.

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

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

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

Figure 3: A new Rulesheet as it appears in Corticon Studio



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

Rulesheet menu commands

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

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

- **Logical Analysis**
  - **Execution Sequence Diagram** - Creates a graphic that shows the execution sequence of the rules in the active Rulesheet. Your default graphic viewer launches to display the diagram. 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: Chaining and looping" in the Rule Modeling Guide.*
  - **Expand Rules** - Creates multiple simple rules from each complex rule.
  - **Collapse Rules** - Reverses the Expand Rules function.
  - **Compress Rules** - Detects overlapping conditions between rules and combines columns to produce a smaller number, but logically equivalent, set of rule columns.
  - **Renumber Rules** - Causes rule columns that have been expanded into component rules (also called sub-rules) to be renumbered from left to right.

- **Simple View | Advanced View** - Toggles display of the Scope and Filters panels on the left side of the Rulesheet.
- **Show Vocabulary Details** |**Hide Vocabulary Details** - Toggles the associated Vocabulary to show or hide details.
- **Show Natural Language** - Displays the Condition and Action expressions in the Rulesheet that have natural language equivalents.
- **Filters** - For more information on filters and preconditions, see the topics in the section *"Filters and Preconditions" in the Rule Modeling Guide.*
  - **Database Filter** - When EDC is enabled, this is a toggle that -- when cleared -- is a filter that is applied to the data currently in working memory. When checked, the filter is a database query that can retrieve data from the database, and add that data to working memory.
  - **Cache Query** - Enables caching on an Entity or database filter. This is independent of Datastore Caching.
  - **Precondition** - Preconditions are Filter expressions that stop Rulesheet execution if not satisfied by at least one piece of data. For more information, see the Filters chapter of the *Rule Modeling Guide*.

- **Processing Mode** - The selected option affects only the active Rulesheet. For more information on the processing mode options, see the topics in the section *"Rule dependency: Chaining and looping" in the Rule Modeling Guide.*
  - **Optimized Inferencing** - Default selection. Causes rules to execute without re-evaluation or re-execution (no looping).
  - **Advanced Inferencing** - Allows only multi-rule inter-dependencies to re-evaluate and re-execute.
  - **Advanced Inferencing with Self-Triggering** - Allows all rule dependencies (including self-dependencies) to re-evaluate and re-execute.

- **Localize** - Opens the **Localization** view. For more information, see the topic Localizing Corticon Studio on page 151.
- **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, as shown:

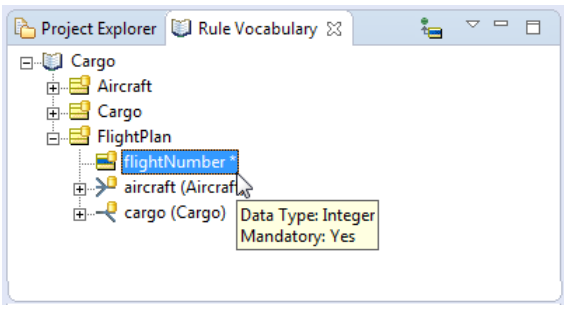
The Rulesheet tools provide the same functions as the corresponding **Rulesheet** menu commands:

-  **Simple View**  **Advanced View** - Toggles to the advanced or simple Rulesheet views. New Rulesheets are shown in Simple view.
-  **Show Vocabulary Details**  **Hide Vocabulary Details** - Toggles the associated Vocabulary to show or hide details.
-  - Switches the view of the active Rulesheet to **Natural Language View**. See the *Rule Modeling Guide* for more information about this feature.
-  - 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**.

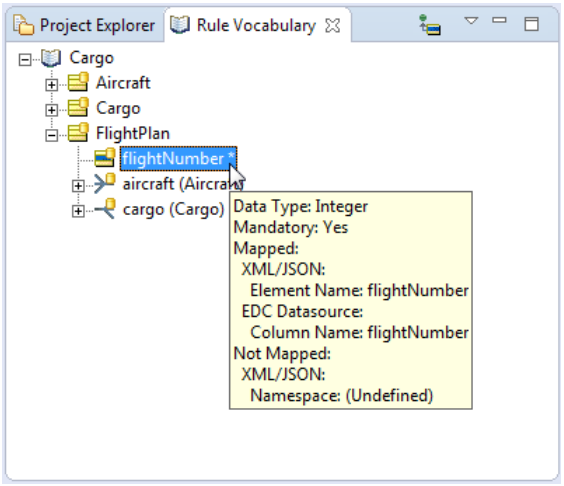
-  - Performs logical loop detection. Same as menu command **Rulesheet** > **Logical Analysis** > **Check for Logical Loops**.
-  - Performs completeness check. Same as menu command **Rulesheet** > **Logical Analysis** > **Check for Completeness**.
-  - Performs conflict check. Same as menu command **Rulesheet** > **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.

Hover help in a Rulesheet

When you hover the cursor over a Vocabulary element, Hover Help provides information about that element, as illustrated:

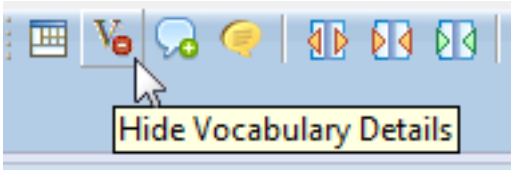


When you add Datasources and document mapping, details about the selected element in each additional mapping can display in the Hover Help, as illustrated showing the details of the XML/JSON mapping option and the EDC Datasource property mapping:



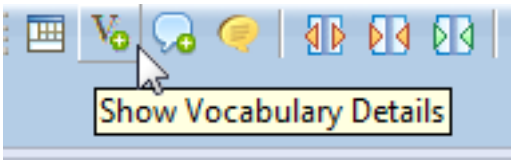
You can control whether the Hover Help displays the defined Datasources and document mapping for the selected element:

- On the currently active menu/toolbar, the **Hide Vocabulary Details** button displays a minus sign:

  When you click it, the Hover Help will show only Basic Properties.

- Then, when you want Hover Help to display details about defined Datasources and document mapping for the selected element, click the **Show Vocabulary Details** button.

Note: If you have not defined any Datasources or document mappings for the selected element, the **Show/Hide** option has no effect.

Commands on the Rulesheet context-sensitive menu

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

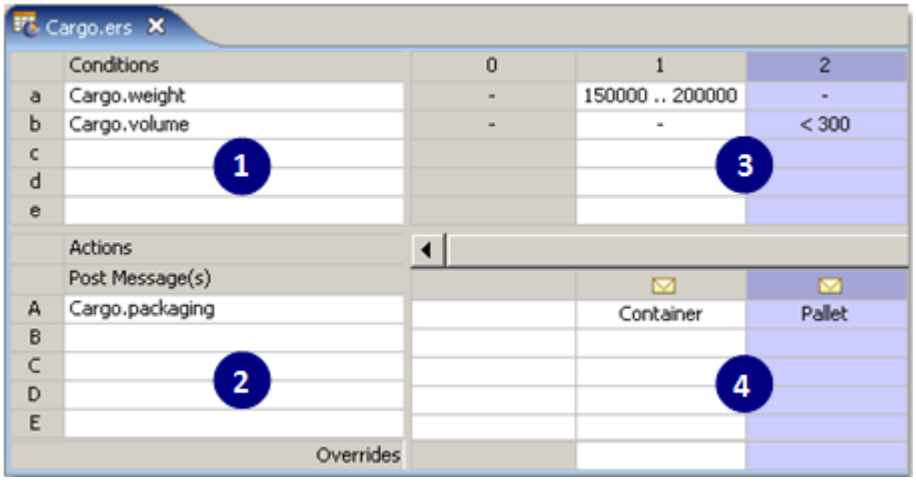
Commands	Description
Undo / Redo	Undo reverts from the previous action. Redo restates an action that was undone.
Cut / Copy / Paste	In any section, performs these standard edit functions.
Select All	In all sections except the Scope, selects all active rows (and columns) in the Rulesheet section in which the menu was activated.
Disable / Enable	On any area except Vocabulary elements, **Disable** greys out one or more selected entries / lines / columns / cells so that they are virtually deleted. Selecting one or more disabled items lets you **Enable** them again.
Insert Row / Remove Row / Add Rows to End	Adds or deletes rows. Adds ten rows the bottom of the current setin 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 Documenting rule assets on page 145 section for details of this feature.
Localize	Opens the **Localization** view. The menu command **Rulesheeet > Localize** performs the same operation. For more information, see the topic Localizing Corticon Studio on page 151.
Natural Language	Opens the **Natural Language** view. The menu command **Rulesheeet > 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 *"Working with rules in natural language" in the Rule Modeling Guide.*

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.



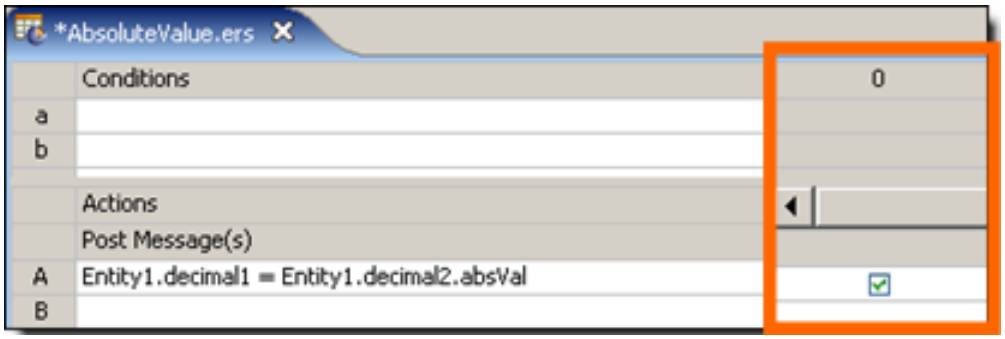
Conditions (quadrant **1**) and **Actions** (quadrant **2**) are organized in the decision table shown above. Sets of Conditions and Actions are tied together to form rules by the vertical columns spanning the two right quadrants (**3** & **4**) of this section.

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.



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.

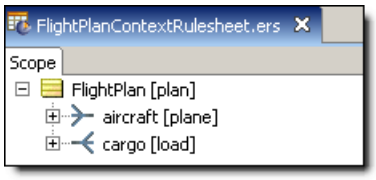
Logically, this means that all Actions modeled in Column 0 are unconditionally executed. In other words, no Conditions must be satisfied in order for these Actions to execute.

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 4: Scope section, showing an Entity and its alias



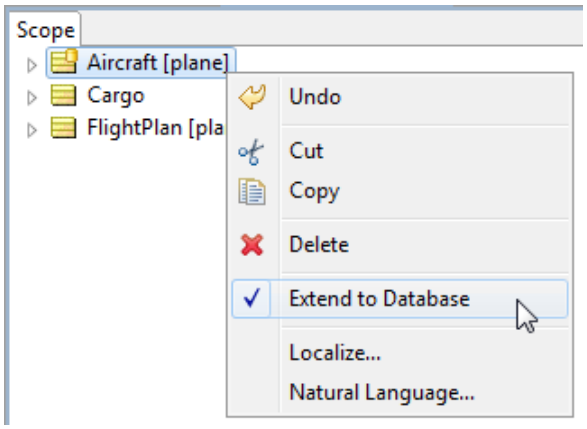
The **Scope** section provides a reduced set of Vocabulary terms with which to build a **Rulesheet**.

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

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

For those Entities in the Scope section, aliases and associations can be defined by double-clicking the Entity to open an edit box. An alias replaces the fully qualified entity name wherever it is used in the Rulesheet, and serves as a sort of local name for use in that Rulesheet only. When an entity has been set to **Datastore Persistent** in the Vocabulary's details view, it is decorated with a database symbol. In the Rulesheet, that means you provide an alias and then set thew entity to **Extend to Database**, as shown:

Figure 5: Setting an alias in scope to Extend to Database

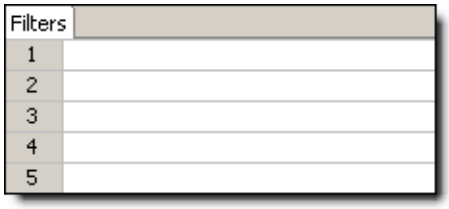


Note: For more information about **Extend to Database**, see the "Writing Rules to access external data" section of the *Rule Modeling Guide* and the "Advanced EDC" topics of the *Data Integration Guide*.

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

Filters

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



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

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

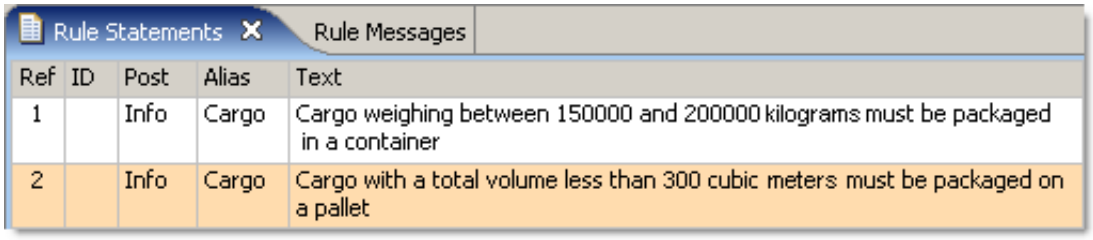
Keyboard shortcuts on a Rulesheet:

- 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:



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

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

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

Ref

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

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

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

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

ID

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

Post

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

Alias

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

Text

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

Rule Name

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

Rule Link

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

Source Name

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

Source Link

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

Category

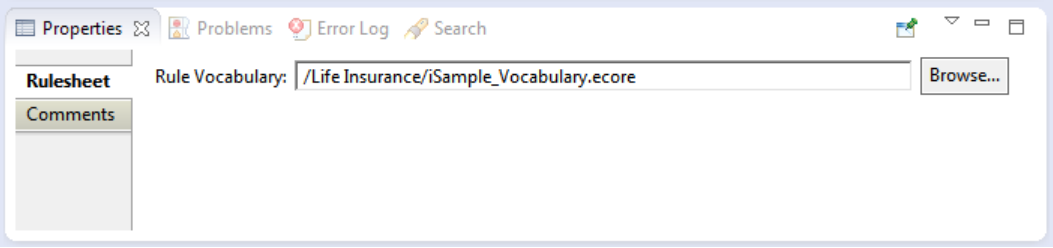
Allows you to assign a category name to each Rule Statement and the rule model to which it links. This field is optional.

Comments

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

Rulesheet properties

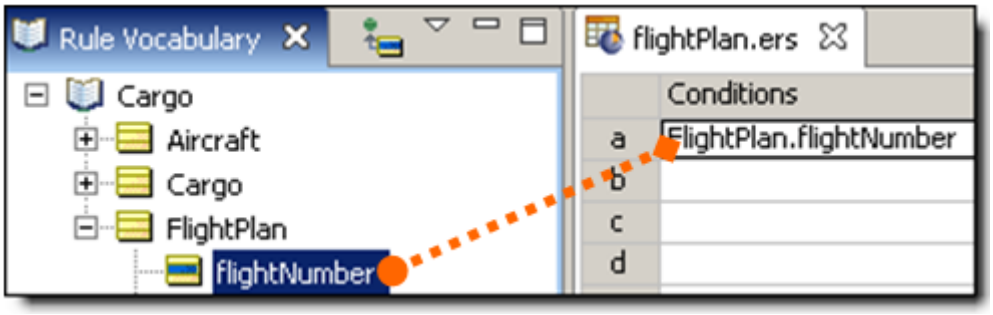
When you are editing a Rulesheet, the **Properties** tab's **Rulesheet** displays the Rule Vocabulary that supports this Rulesheet.



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

Using the business vocabulary to build rules

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



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

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

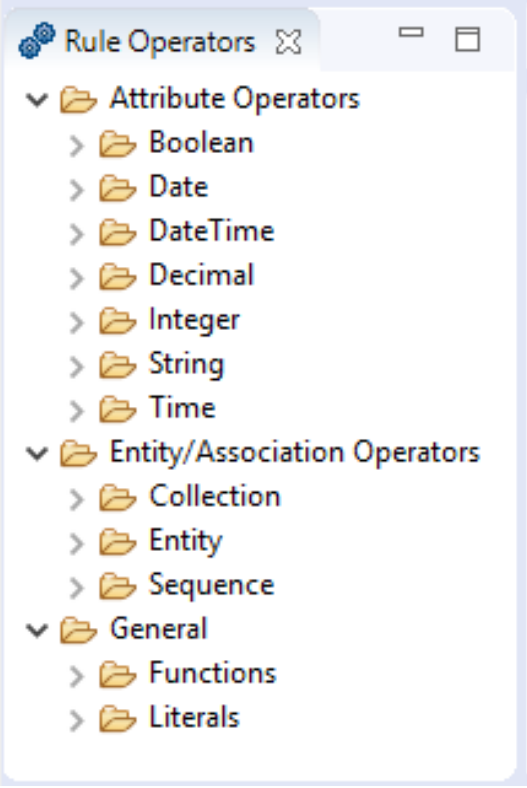
Using the operator vocabulary to build rules

Literal Terms, Functions & Operations

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

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

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



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

Naming Rulesheets

Rulesheets can be named when created and renamed later by:

Clicking **File** > **Save As…**

Rulesheet names must adhere to and comply with the guidelines shown in the File naming restrictions on page 18 section of this document.

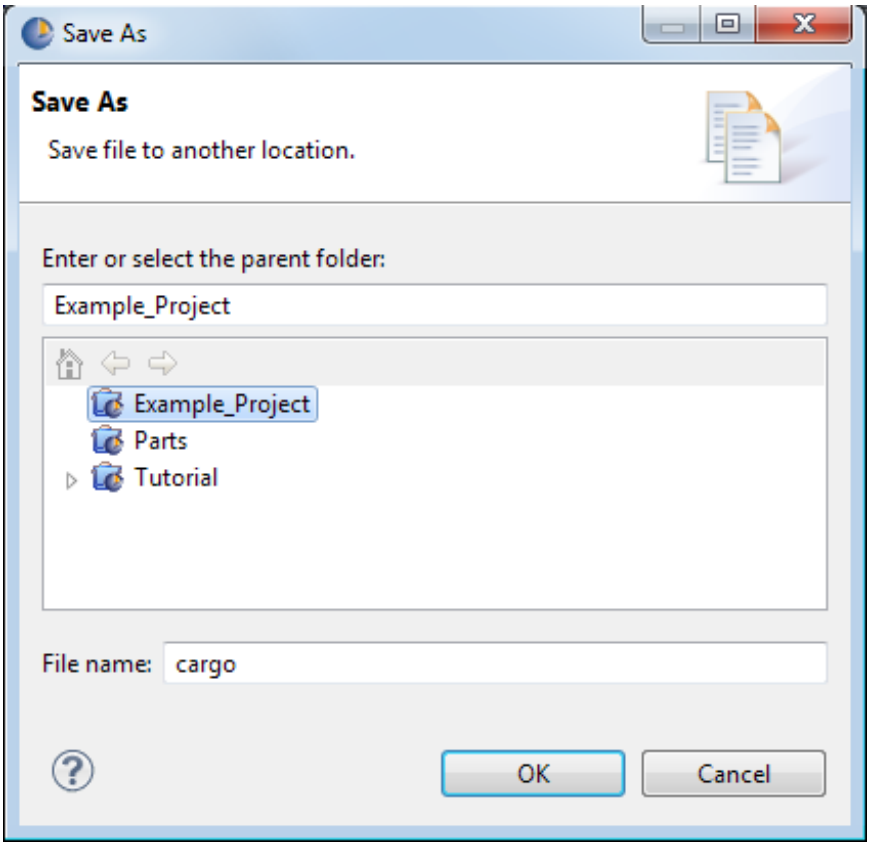
Deleting Rulesheets

Deleting Rulesheets is performed using one of the following methods:

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

Saving a new Rulesheet

1. With the Rulesheet selected, choose the Save command or click the save button.
2. Navigate to the **Project** folder where you want to store the Rulesheet.
3. The same file naming conventions apply to Rulesheets as apply to Vocabularies. See File naming restrictions on page 18.



Validating and optimizing a Rulesheet

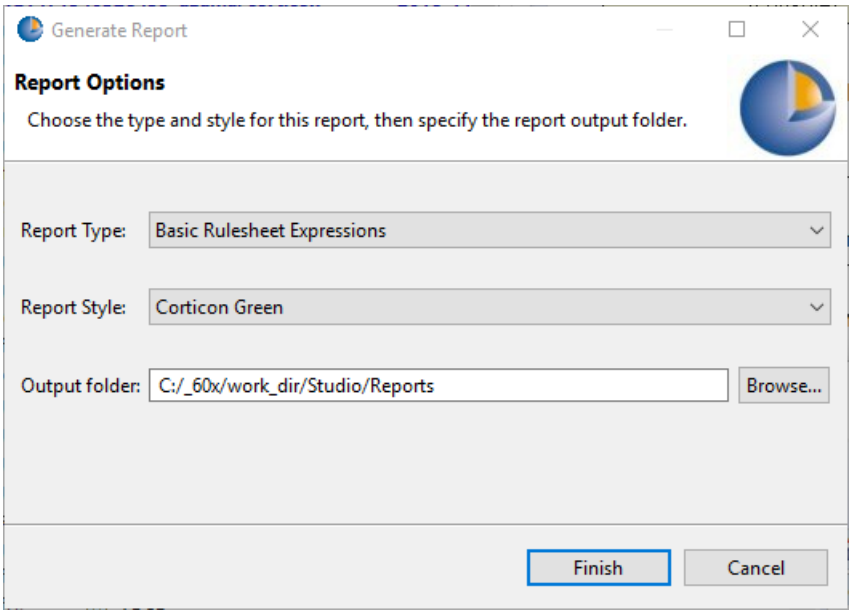
Conflict, Completeness, and Logical Loop Checkers, as well as the Compress Rules feature, are collectively known as validation and optimization functions. These topics are addressed briefly in the *Corticon Studio Tutorial: Basic Rule Modeling* , and in more depth in the *Corticon Studio: Rule Modeling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

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

Creating a Rulesheet report

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

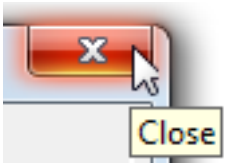
   The **Generate Report** dialog opens, as shown:



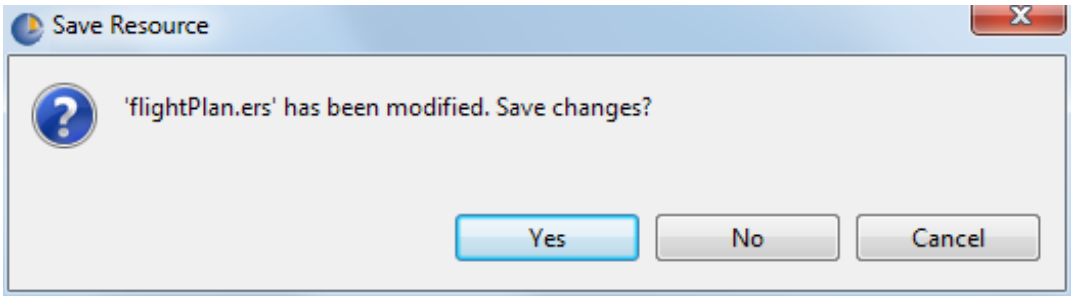
2. Choose the **Report Type**, **Report Style**, and **Output Folder** for this report. Report types focus on either Natural Language or Expressions in the Rulesheet.
3. Click **Finish**.
4. The Rulesheet report opens as a new HTML page in your default web browser, and then saves the HTML, XML, and CSS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

For information about creating custom transformations and stylesheets for reports, see the topic *Reporting Framework chapter of the Rule Modeling Guide.*

Closing a Rulesheet

1. To close a Rulesheet, click its Close button, 

, or select the menu command **File** > **Close**.



2. Choose **Yes** to save any changes, and close the file.

Saving a modified Rulesheet

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

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

Creating rules by importing an Excel worksheet

Often decisions are based on statistical data in tables that are actuarial or real-world data that cannot be derived through a formula. Insurance underwriting, manufacturing tooling, life sciences, and census data have examples of data that is key to decision making yet revised only at regular time intervals.

When used in Corticon, a Rulesheet that provides such data is very useful but could require hours of data entry to create what might be thousands of rules. This feature makes that task fast, easy, and accurate.

Consider the following excerpt of the sample Excel worksheet with data as an example for life insurance underwriting. (In a Studio installation, the file is located at `[CORTICON_WORK_DIR]\Samples\ImportExcel\SampleDimensionalData.xslx`.)

Column A contains headers that identify the applicant's age, and row 1 contains headers that identify the beneficiary's age. The cells are the policy factor that adjusts the policy premium:

	A	B	C	D
1	**Dimensions**	**50**	**51**	**52**
2	**40**	0.9680	0.9698	0.9715
3	**41**	0.9652	0.9670	0.9689
4	**42**	0.9618	0.9640	0.9660

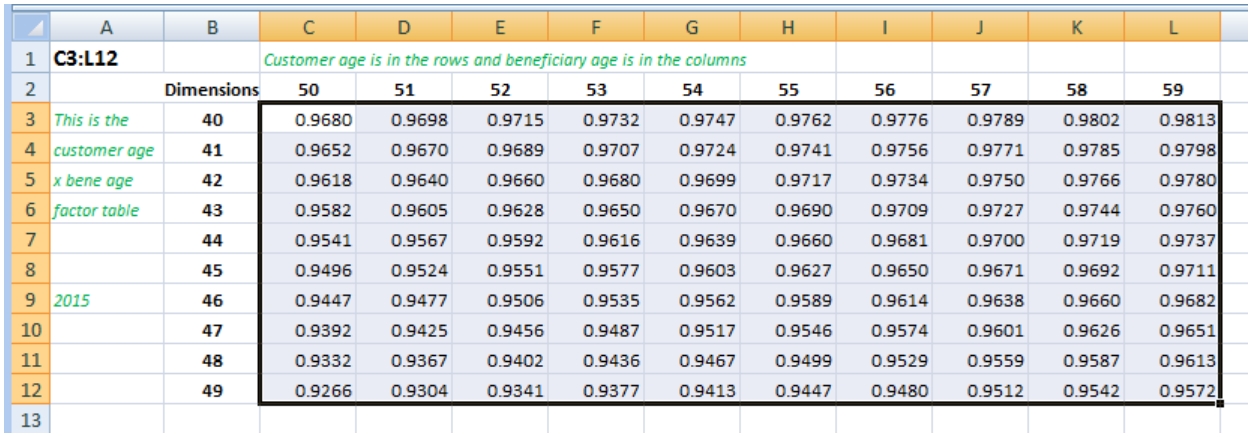
When this two-dimensional information is imported into a Corticon Rulesheet, each of the two dimensions and the data area are assigned to Vocabulary attributes within an entity. Then, each of the two dimensions defines a Condition and the data point specifies an Action, as shown:

	Conditions	0	1	2	3
a	Policy.applicantAge		50	51	52
b	Policy.beneficiaryAge		40	40	40
	Actions				
	Post Message(s)				
A	Policy.factor		0.968	0.9698	0.9715

When the data range expands, there could be thousands of rules in the Rulesheet that were all created by one import action, and later updated by performing the same import from the revised worksheet.

Preparing Excel sheets

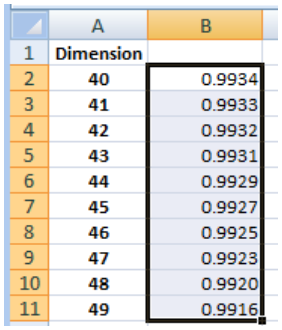
In order to achieve successful import of a dimensional Excel worksheet into Corticon, the sheet must have a specific layout. The sample sheet **Two Dimensions** is modified and offset here to present a few concepts:



	A	B	C	D	E	F	G	H	I	J	K	L
1	**C3:L12**		*Customer age is in the rows and beneficiary age is in the columns*									
2		**Dimensions**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**
3	*This is the*	**40**	0.9680	0.9698	0.9715	0.9732	0.9747	0.9762	0.9776	0.9789	0.9802	0.9813
4	*customer age*	**41**	0.9652	0.9670	0.9689	0.9707	0.9724	0.9741	0.9756	0.9771	0.9785	0.9798
5	*x bene age*	**42**	0.9618	0.9640	0.9660	0.9680	0.9699	0.9717	0.9734	0.9750	0.9766	0.9780
6	*factor table*	**43**	0.9582	0.9605	0.9628	0.9650	0.9670	0.9690	0.9709	0.9727	0.9744	0.9760
7		**44**	0.9541	0.9567	0.9592	0.9616	0.9639	0.9660	0.9681	0.9700	0.9719	0.9737
8		**45**	0.9496	0.9524	0.9551	0.9577	0.9603	0.9627	0.9650	0.9671	0.9692	0.9711
9	*2015*	**46**	0.9447	0.9477	0.9506	0.9535	0.9562	0.9589	0.9614	0.9638	0.9660	0.9682
10		**47**	0.9392	0.9425	0.9456	0.9487	0.9517	0.9546	0.9574	0.9601	0.9626	0.9651
11		**48**	0.9332	0.9367	0.9402	0.9436	0.9467	0.9499	0.9529	0.9559	0.9587	0.9613
12		**49**	0.9266	0.9304	0.9341	0.9377	0.9413	0.9447	0.9480	0.9512	0.9542	0.9572
13												

- The data range, `C3:L12`, has no empty cells, and each cell has the same data type.
- Column `B`, to the left of the start point, is assumed to have the values for each row in the vertical dimension.
- Row `2`, above the start point, is assumed to have the values for each column in the horizontal dimension.
- The dimension values have no empty cells, no duplicate values, and each cell has the same data type.
- The dimension and data cells, `B2:L12`, are the only cells read by the import feature. Data in any other cells have no impact on the import. Conversely, setting a range that includes blank cells, such as `C3:M13`, will not import successfully.

A one dimensional table follows similar rules. The only consideration is to specify only the appropriate dimension,, as in the sample sheet **One Dimension** that uses the vertical attribute:



	A	B
1	**Dimension**	
2	**40**	0.9934
3	**41**	0.9933
4	**42**	0.9932
5	**43**	0.9931
6	**44**	0.9929
7	**45**	0.9927
8	**46**	0.9925
9	**47**	0.9923
10	**48**	0.9920
11	**49**	0.9916

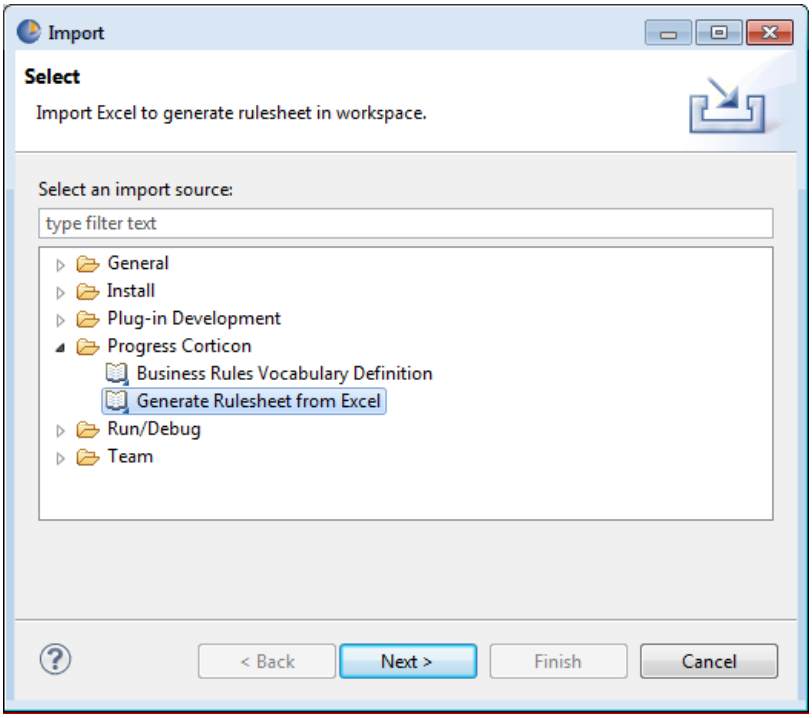
Note: Decimal precision - Some decimal values might seem imprecise between the Excel workbook value and their corresponding Rulesheet action value. A good practice is to set the Excel workbook precision before initiating an import into Corticon. With the Workbook open and the data formatted to the preferred decimal precision, click the **Microsoft Office** Button, click **Excel Options**, and then click **Advanced** category. In the **When calculating this workbook section**, select the workbook that you want, click to select the **Set precision as displayed** check box, and then click **OK**. You are alerted that you are losing precision while in fact you are ensuring that the value displayed is the value that is imported into Corticon. Save and close the Excel workbook, and then start the import process.

Importing the Excel sheet into a Rulesheet format

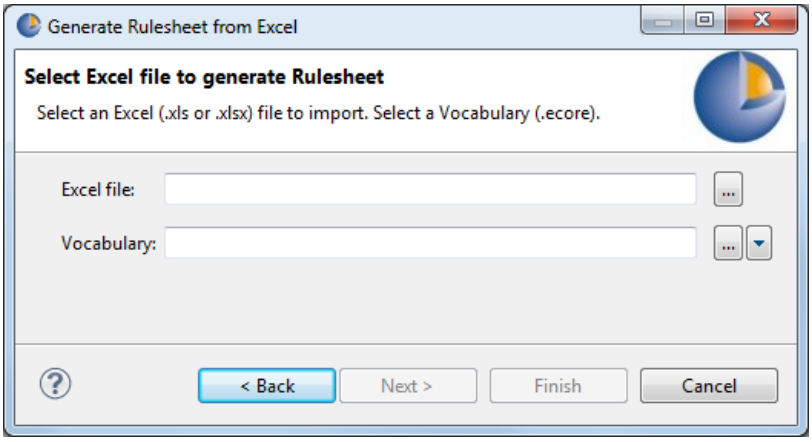
Once your Excel sheet has been prepared, you can import it into a Rulesheet.

To import an Excel sheet into Corticon Studio:

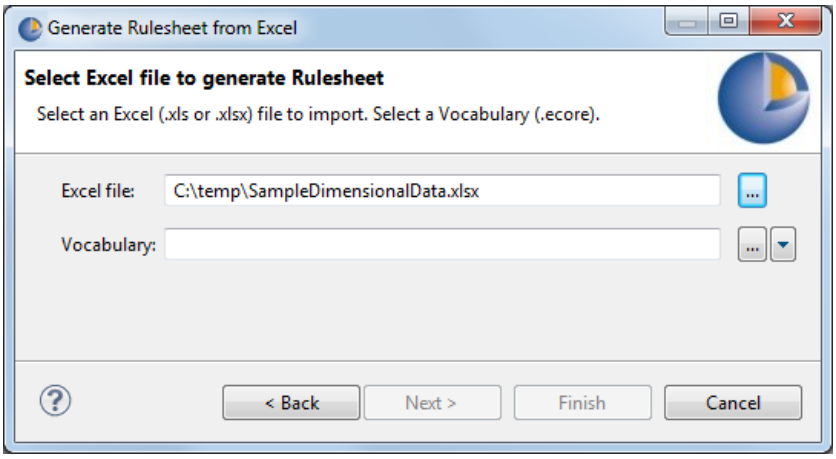
1. Select the menu command **File > Import**. You could instead navigate to the Vocabulary of the project where the Rulesheet will be created to right-click on it, and then select **Import**.



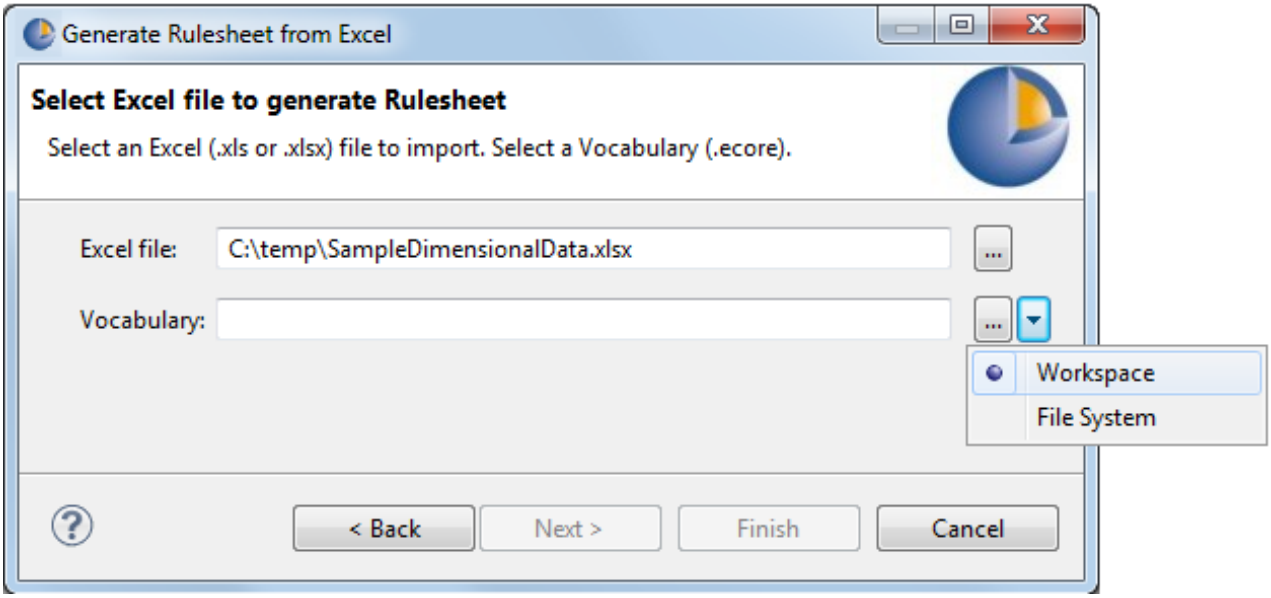
Click **Next**. The **Generate Rulesheet from Excel** dialog box opens:



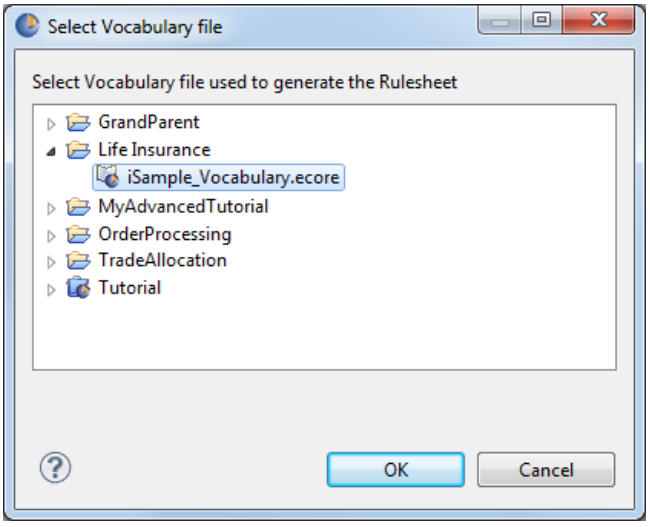
2. Click the **Excel file** search button, and then locate and open the Excel file. For this example, select `SampleDimensionalData.xlsx`.



3. Click the **Vocabulary** search button, and then locate and open the Vocabulary file. As illustrated, you can choose to find it in the workspace or in the file system. (If you launched the **Import** function with the Vocabulary file selected, the Vocabulary file is pre-selected when the dialog opens.)



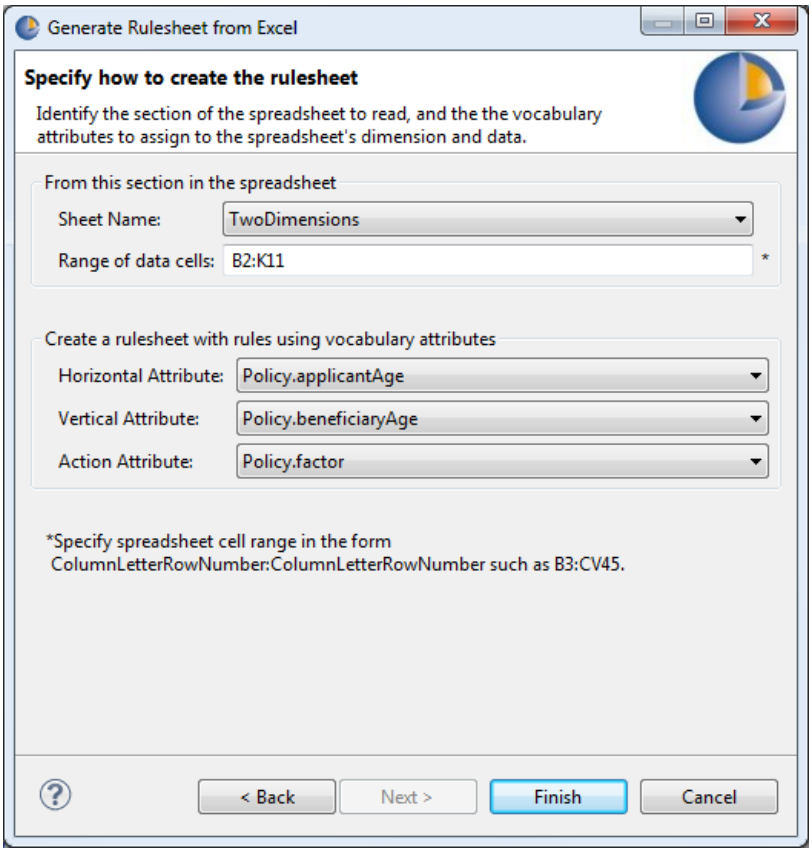
4. To follow the example, choose the Life Insurance sample's Vocabulary:



Click **OK** to close the selection dialog, and then click **Finish** to enter the import parameters.

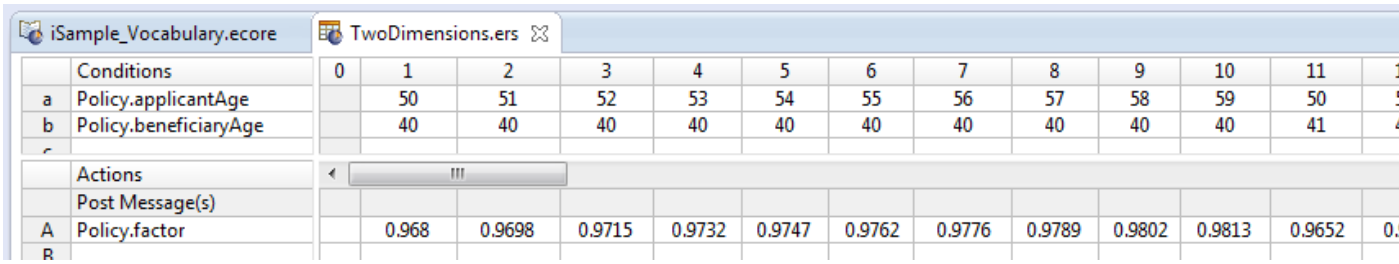
Note: If you are following these steps hands on in Studio, you need to add three attributes to the Life Insurance sample's Vocabulary **Policy** entity, one named `applicantAge` with the datatype `human_age`, one named `beneficiaryAge` with the datatype `human_age`, and another named `factor` with the datatype `decimal`.

5. On the **Specify how to create the Rulesheet** panel, choose or enter the parameters as shown:



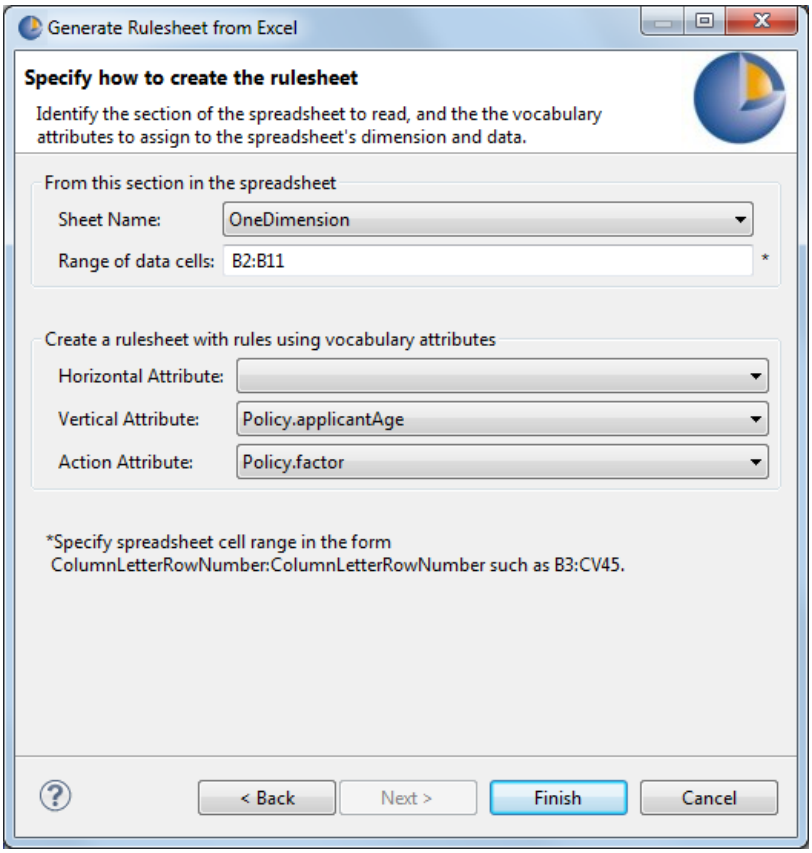
Click **Finish**. The import process creates a Rulesheet in the project with the name of the Excel sheet.

6. Review the results in the Studio:



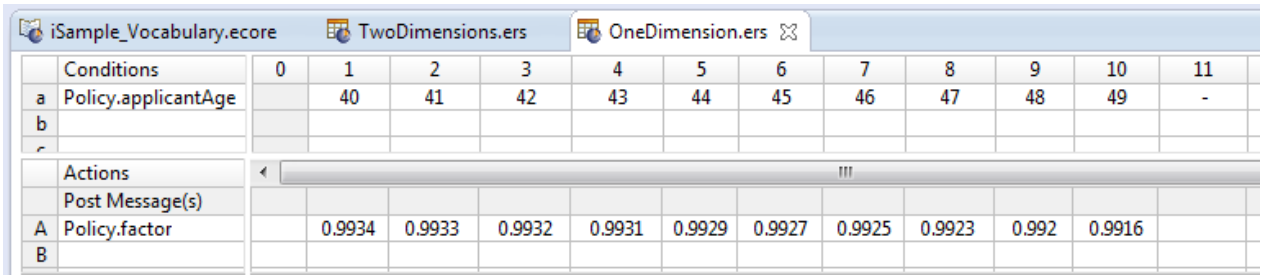
The Rulesheet has 100 columns, one for each rule from the 10x10 matrix that was imported.

7. To follow up with an import of the one dimensional sheet, repeat steps 1 through 4, and then choose or enter the parameters as shown:



Notice that the Horizontal attribute was left blank and the range specifies only one column.

8. Review the results in the Studio:



The Rulesheet has 10 columns, as shown, one for each rule from the 10x1 matrix that was imported.

Updating an imported sheet

When you run these import tasks again, the existing Rulesheet of the same name in the project is replaced.

5

Ruleflows

A Ruleflow is how Corticon assembles and organizes the components of a set of rules into a single unit. It is a 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. But in order to deploy a Rulesheet to it must be part of a Ruleflow. Only Ruleflows can be compiled into Decision Services, and only Decision Services are deployable to Corticon Server .

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

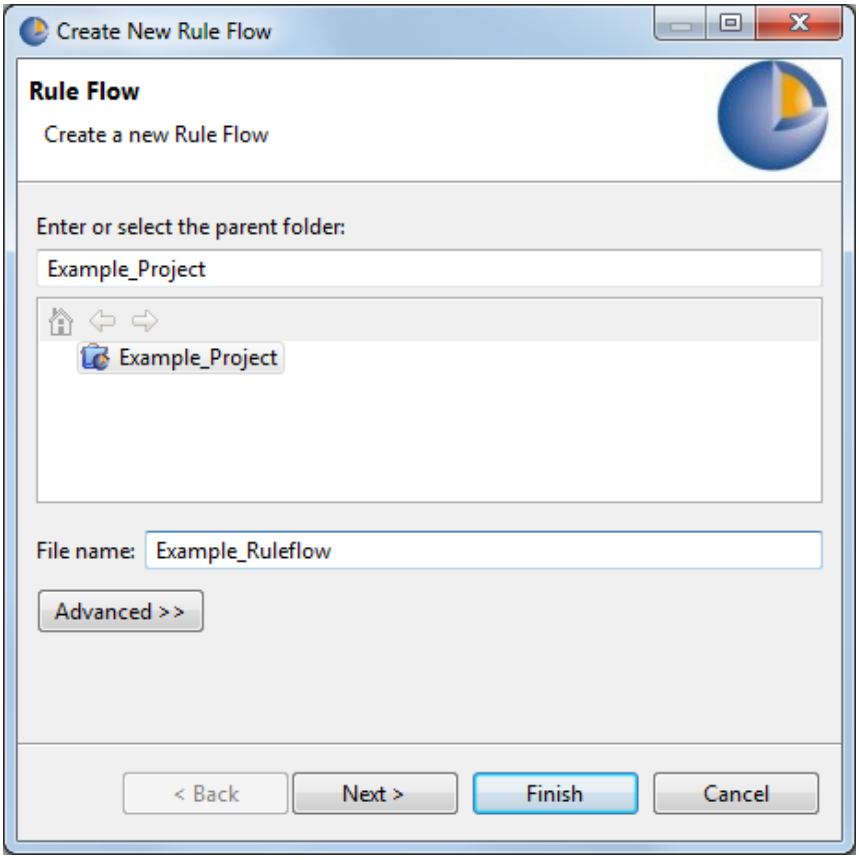
For details, see the following topics:

- Creating a Ruleflow
- Ruleflow window
- Ruleflow properties
- Ruleflow preferences

Creating a Ruleflow

Follow these steps to create a new Ruleflow:

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

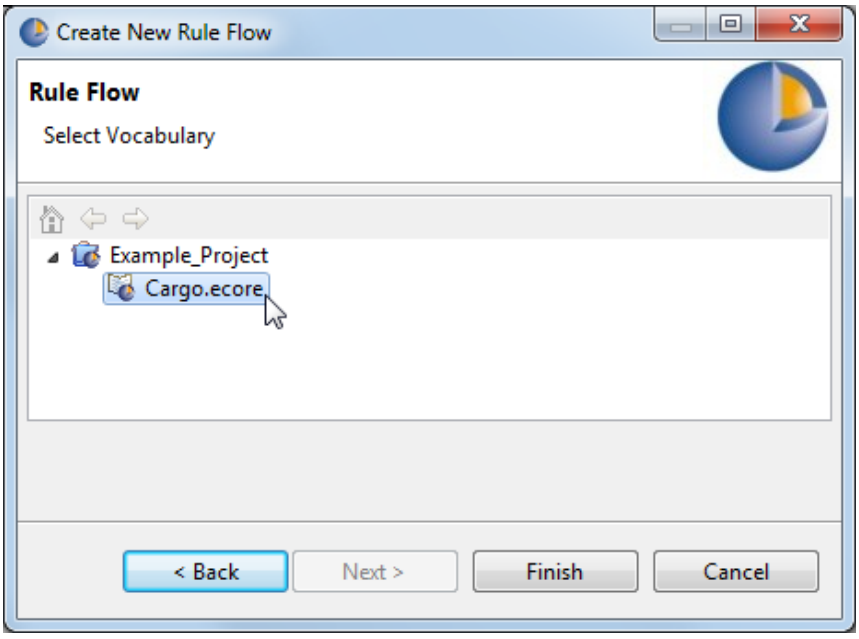


2. On the **Project** list, select the Project you want to associate with the new Ruleflow (or enter it manually in the **Enter or select parent folder** entry area.
3. Enter a file name for the Ruleflow in the **File name** entry area.

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

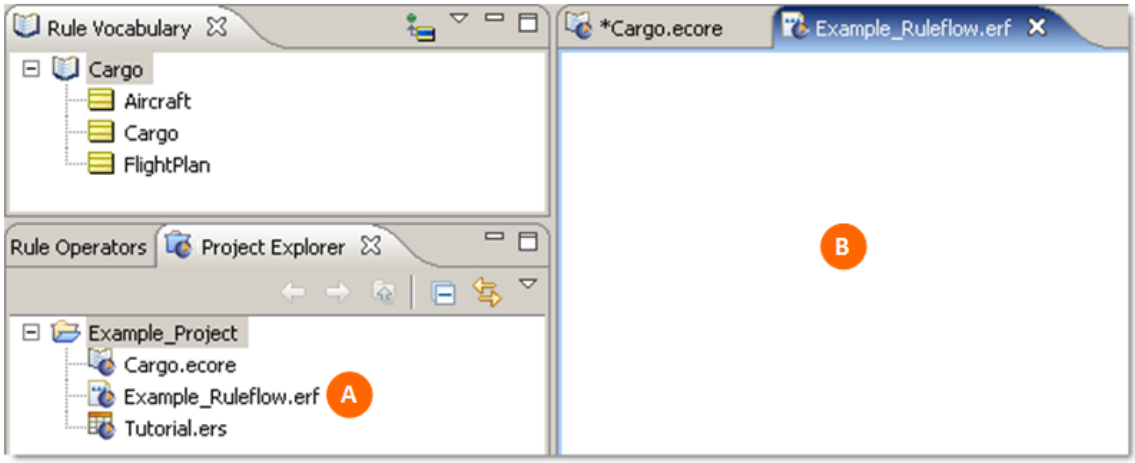
---

4. Click **Next** to continue.



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. Here, we simply expand the `Tutorial` folder and highlight the `Cargo.ecore` file. As you create additional Vocabulary files within a Project they become available for selection from this list for use with this or other Ruleflows that you create.



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

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

Ruleflow window

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

Naming Ruleflows

Ruleflow names must adhere to and comply with the guidelines shown in the File naming restrictions on page 18 section of this document.

Adding Ruleflows

To add additional Ruleflows to a Rule Project

Select **File > New > Ruleflow** from the menubar and repeat the steps recounted earlier in the Ruleflow creation process.

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

Deleting Ruleflows

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

Saving a new Ruleflow

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

  Refer to the Ruleflow menubar section for details regarding saving, or renaming, a Ruleflow with a different name or to a different file location.

Ruleflow menu commands

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

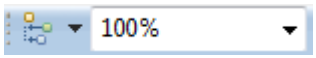
The **Ruleflow** menu has the following commands:

- **Comments** - 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 Using the Comments view on page 149.
- **Properties** - Opens the Ruleflow properties window.

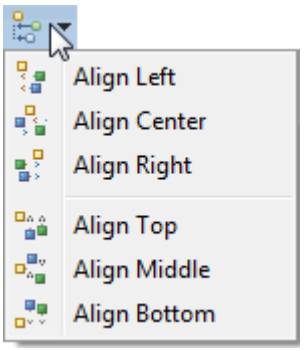
- **Dependency graph** - Generates attribute and logical dependency graphs based on selected nodes in the active Ruleflow. For more information, see *"Generating Ruleflow dependency graphs" in the Rule Modeling Guide.*
- **Report** - Creates an HTML report and launches your browser for viewing. See Creating a Ruleflow Report.
- **Export WSDL** - Opens the **Export Ruleflow WSDL** dialog box for this Ruleflow. WSDL service contracts are discussed in the set of topics in the Deployment section, *Integrating Corticon Decision Services*.
- **Export XSD** - Opens the **Export Ruleflow XSD** dialog box for this Ruleflow. XSD service contracts are discussed in the set of topics in the Deployment section, *Integrating Corticon Decision Services*.
- **Service Contract** - Creates a CSV file that provides guidelines for creating a service contract for the Ruleflow. See *"Examples of service contract reports generated from a Ruleflow" 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.

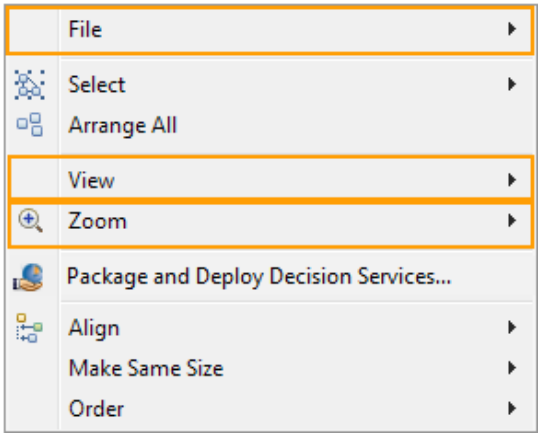
	Aligns multiple selected objects on the Ruleflow canvas.
	Zooms the Ruleflow window to the specified magnification.

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.

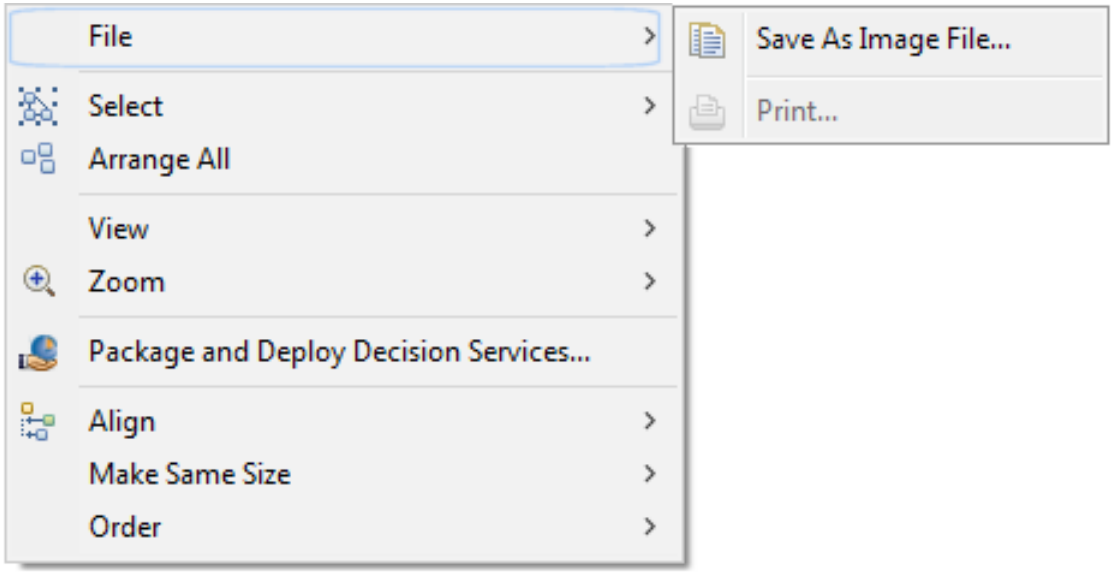
Note: You might see a menu item **Remove from Context**. It is an Eclipse operator that is listed yet inactive in the Ruleflow. It is benign and should be ignored. See Eclipse documentation for more information.

See Object commands on the Ruleflow context-sensitive menu on page 107 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



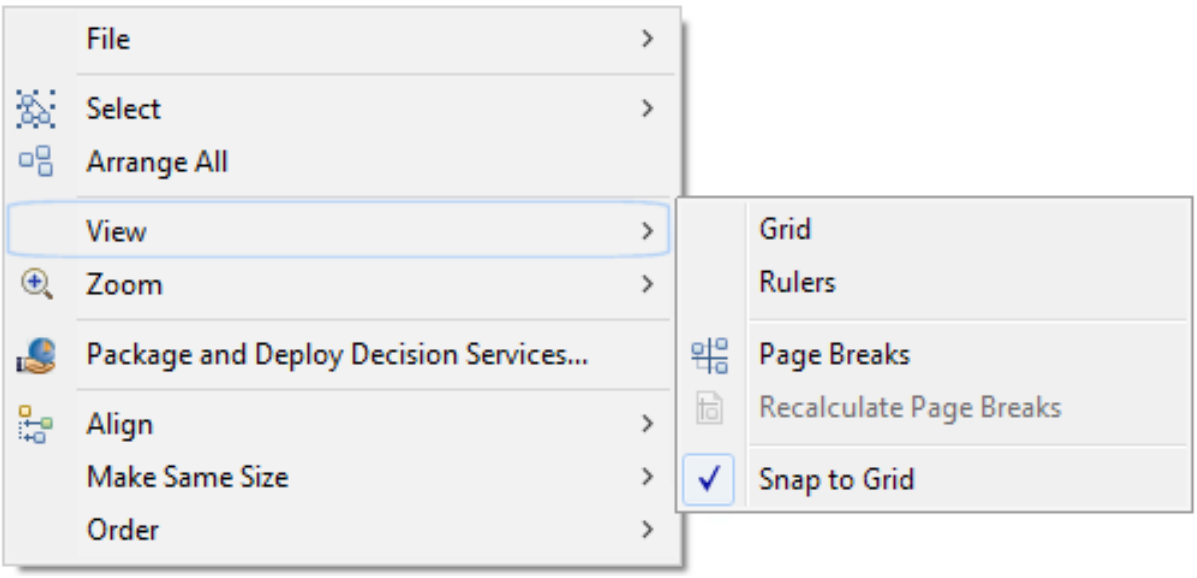
Select

See Object commands on the Ruleflow context-sensitive menu on page 107.

Arrange All

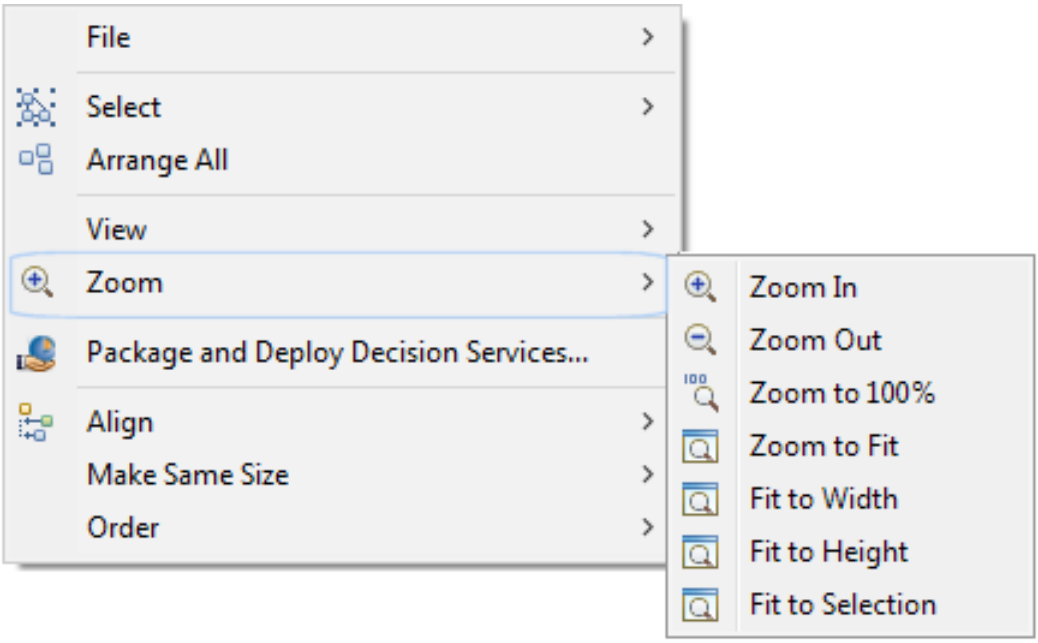
See Object commands on the Ruleflow context-sensitive menu on page 107.

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 and Deploy Decision Services

Lets you move the Ruleflow immediately to compilation and deployment as a Decision Service. Forces you to save all changes, then opens the **Package and Deploy Decision Services** dialog. See *"Using Studio to compile and deploy Decision Services" in the Deployment Guide.*

Align

See Object commands on the Ruleflow context-sensitive menu on page 107.

Make Same Size

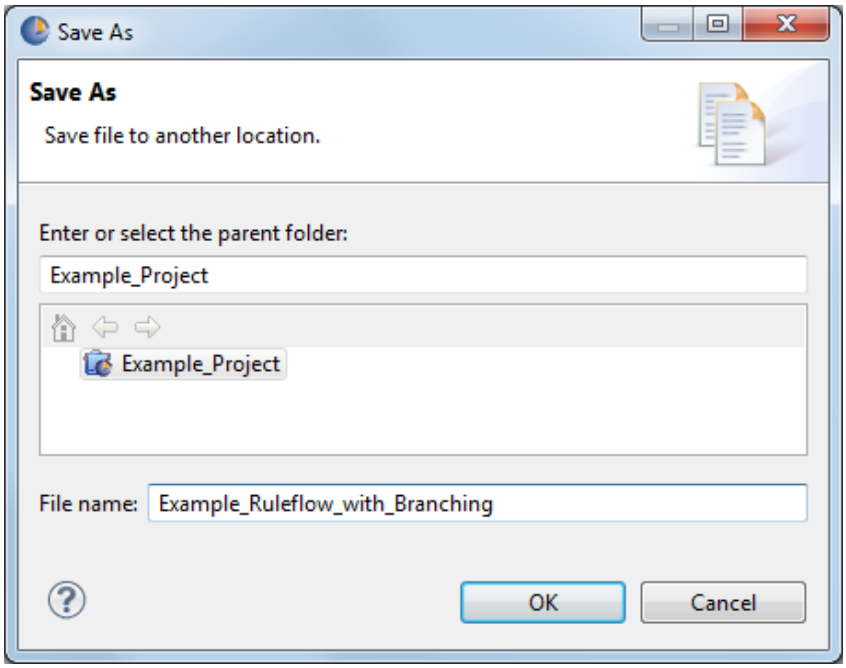
See Object commands on the Ruleflow context-sensitive menu on page 107.

Order

See Object commands on the Ruleflow context-sensitive menu on page 107.

Renaming a Ruleflow or saving a Ruleflow to a different location

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

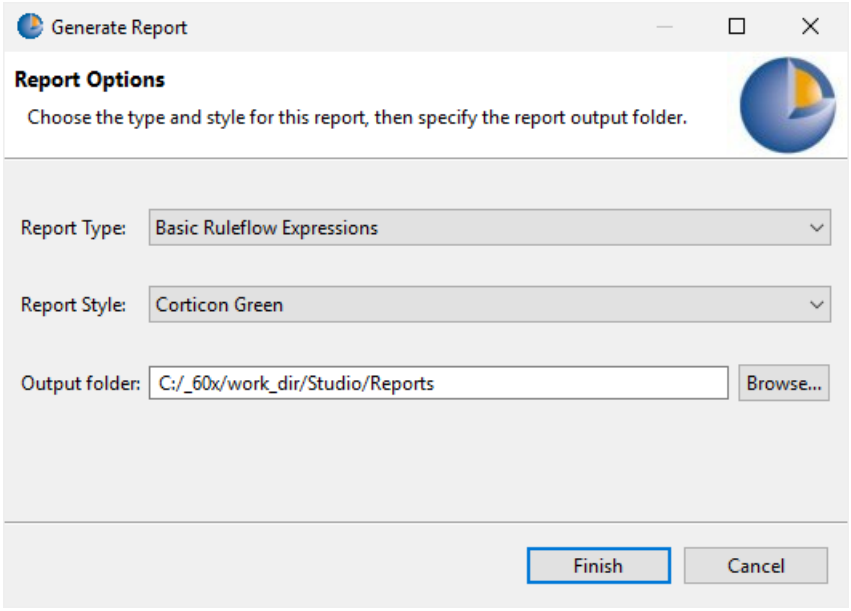


1. To rename the Ruleflow, modify the name in the **File name:** field with the **Project** folder highlighted in the **Project** list.
2. To save the Ruleflow to a different location, whether you are renaming it or not, select that **Project** folder in the **Project** list or enter the parent folder name manually in the **Enter or select the parent folder:** text field.
3. The same file naming conventions apply to Ruleflows as apply to Rulesheets. See the File naming restrictions on page 18 section of this document.

Creating a Ruleflow report

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

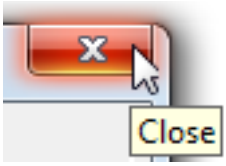
   The **Generate Report** dialog opens, as shown:



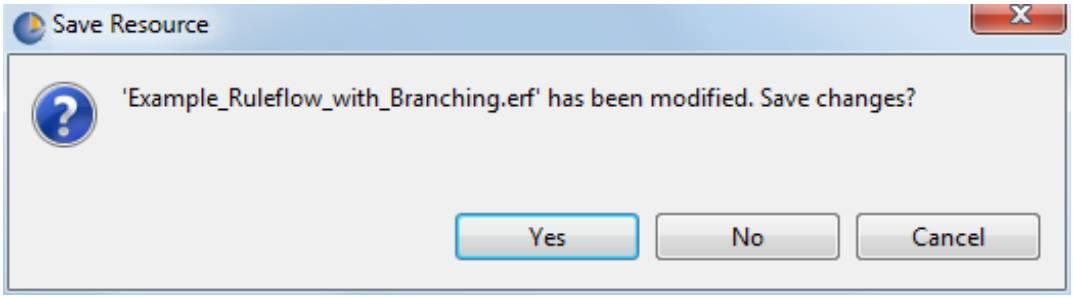
2. Choose the **Report Type**, **Report Style**, and **Output Folder** for this report. Report types focus on either Natural Language or Expressions in the Ruleflow's Rulesheets.
3. Click **Finish**.
4. The Ruleflow report opens as a new HTML page in your default web browser, and then saves the HTML,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

For information about creating custom transformations and stylesheets for reports, see the topic *Reporting Framework chapter of the Rule Modeling Guide.*

Closing a Ruleflow

1. To close a Ruleflow, click its Close button, 

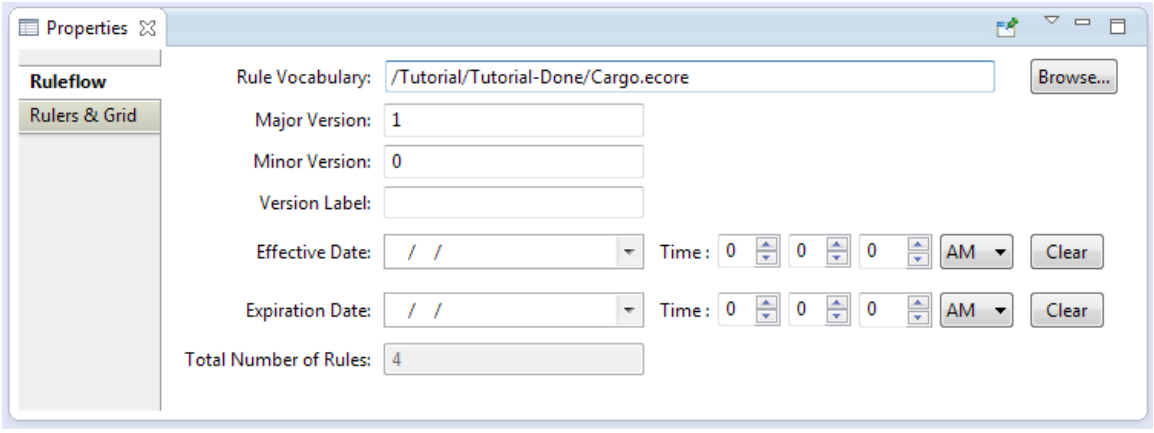
, or select the menu command **File** > **Close**.



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

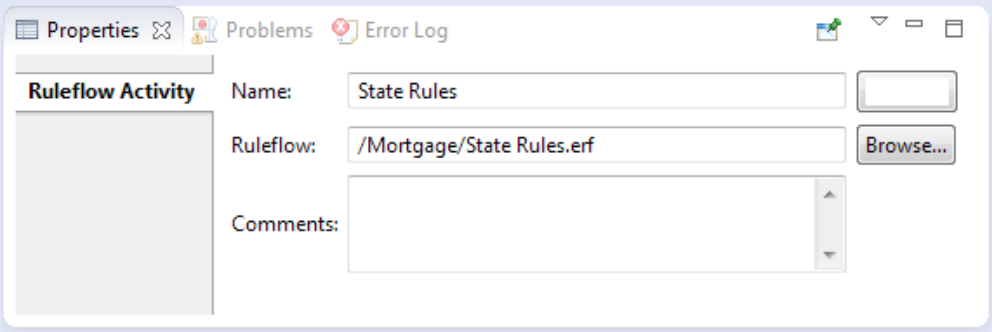
Ruleflow properties

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



Note:

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



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

For more information, see *Conditional Branching in Ruleflows in the Corticon Rule Modeling Guide* and Properties of Ruleflow objects on a Ruleflow canvas on page 110.

Ruleflow

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

Rule Vocabulary

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

Major and Minor Version Numbers

Version numbers for Ruleflows are optional and are assigned manually.

When different version numbers are assigned to a Ruleflow, the Server keeps them distinguished in memory, and responds correctly to requests for the different (Major) versions. In other words, an application or process can use (or call) different (Major) versions of the same Ruleflow simply by including the (Major) version number in the request message. The details of how this works at the Server level are described in the *Server Guide*.

Version Label

A plain-text description of this version can be added in the Ruleflow Version Label field.

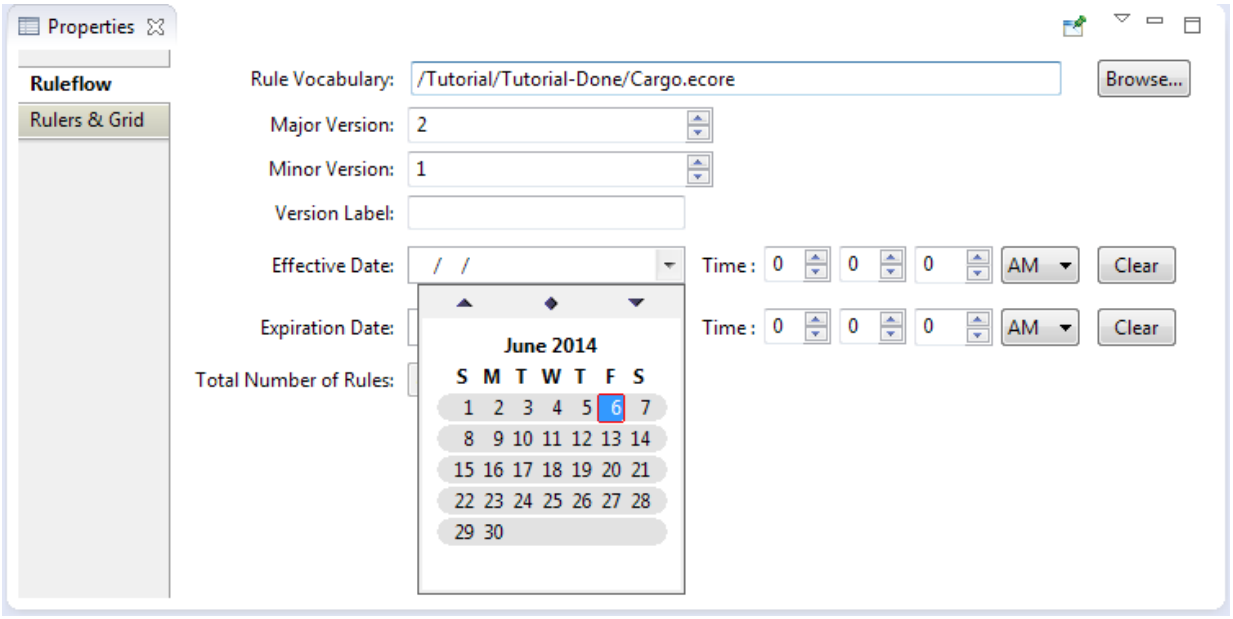
Optional. Give the version a unique text label.

Effective Date and Expiration Date

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

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

Figure 6: Setting the Effective Date on a Ruleflow



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

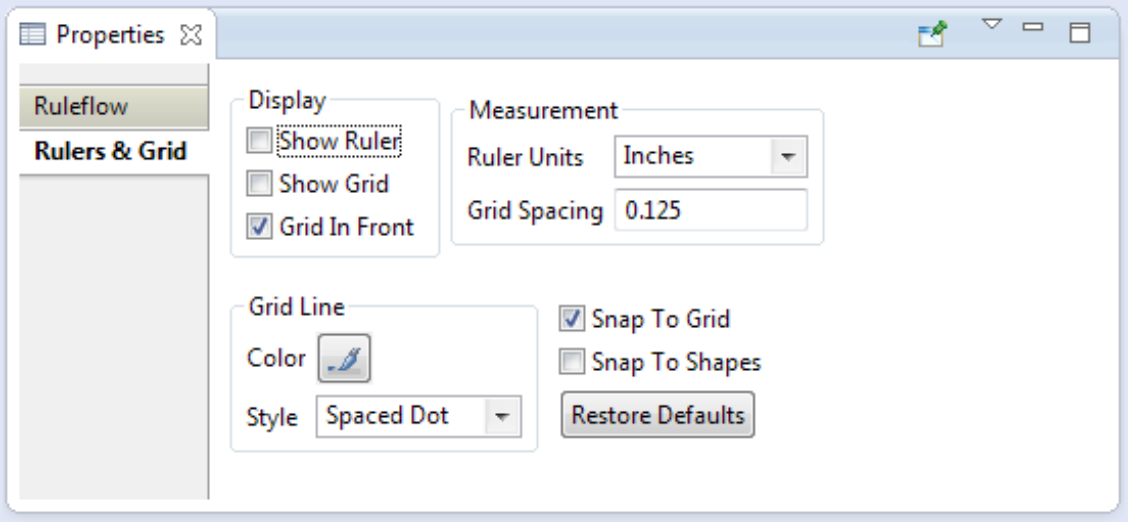
Total Number of Rules

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

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 7: 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:

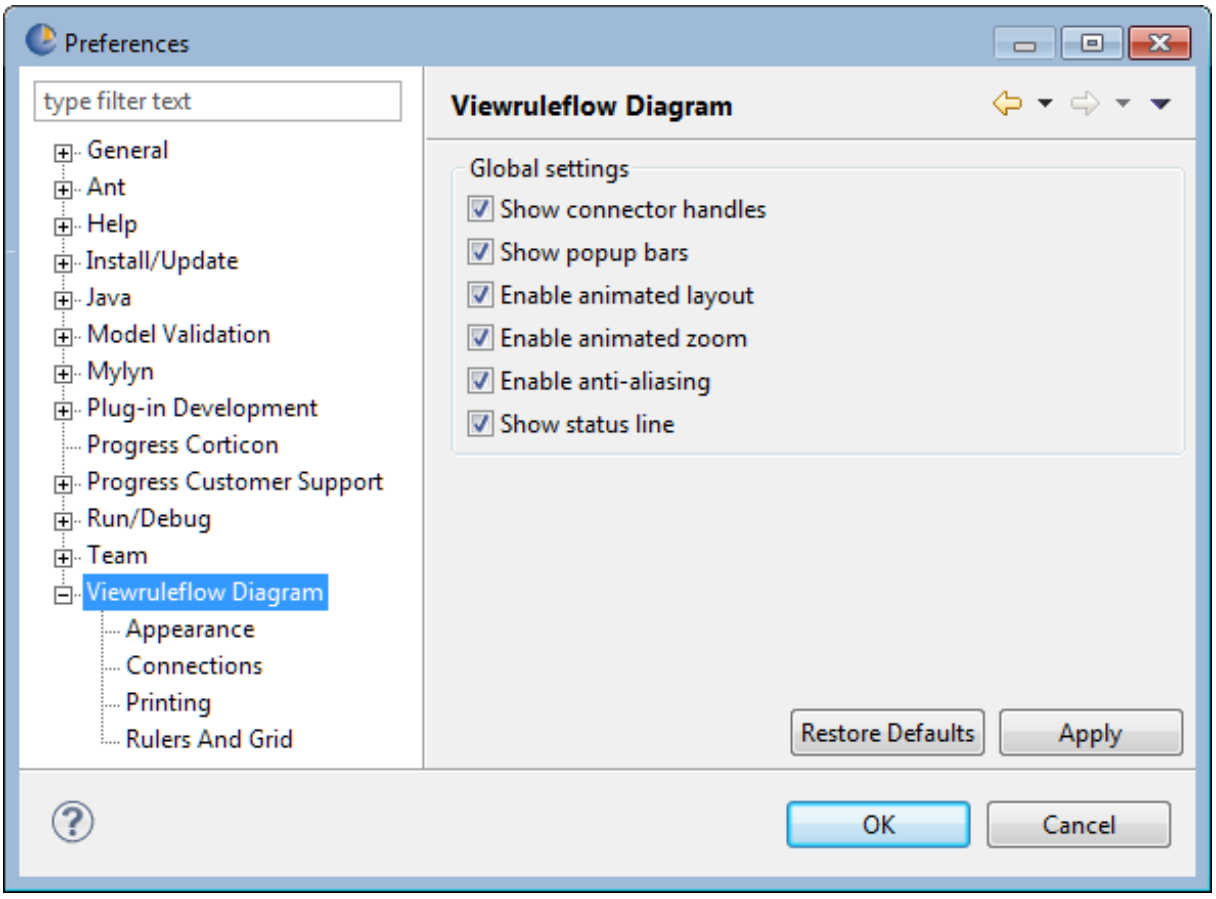
```
These properties set the font type and size used by the graph visualizer
  com.corticon.crml.CrmlGraphVisualizer.fontname=Helvetica-Narrow.ttc
  com.corticon.crml.CrmlGraphVisualizer.fontname.ja=msgothic.ttc
  com.corticon.crml.CrmlGraphVisualizer.fontsize=9
```

Ruleflow preferences

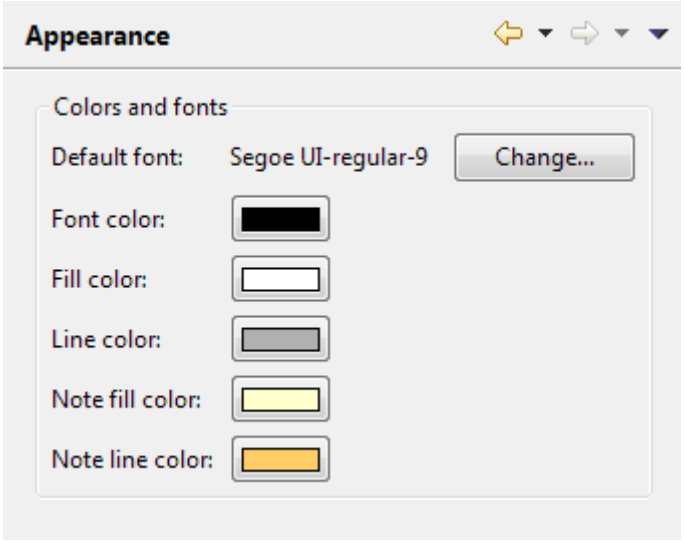
You can set many aspects of a the Ruleflow canvas and its output.

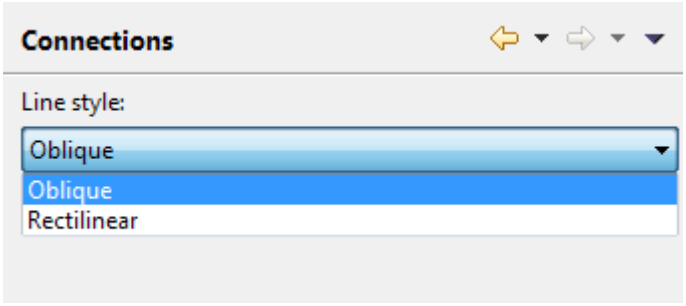
To set Ruleflow Preferences:

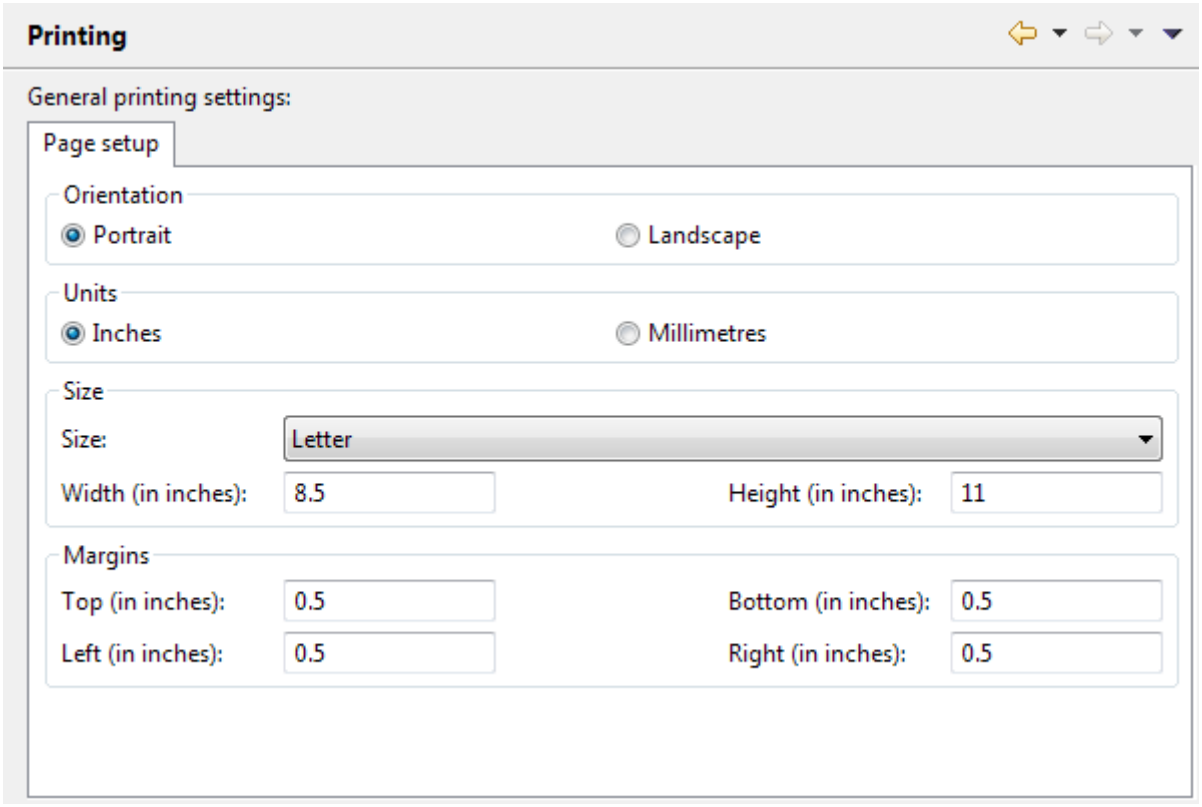
1. In Studio, choose the menu item **Window > Preferences**.
2. Click on **Viewruleflow Diagram**. The global settings are available as shown:

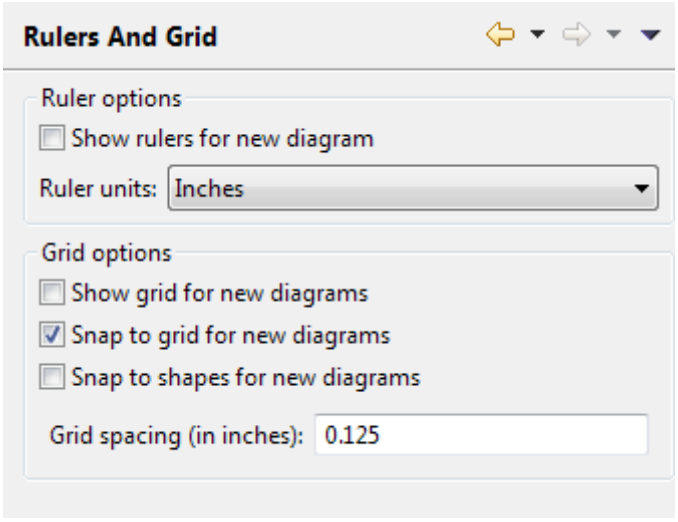


3. Expand **Viewruleflow Diagram**, and then click on each of its items to expose the corresponding settings:





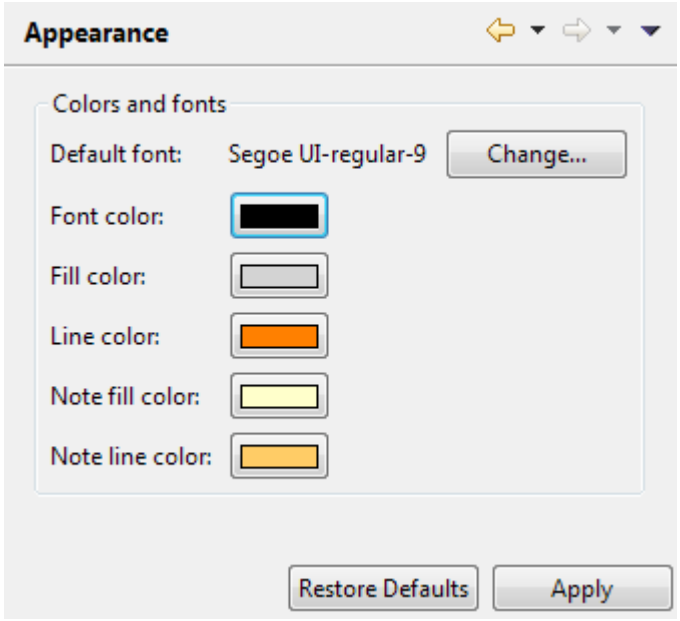




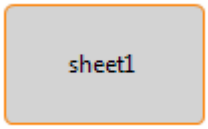
4. Adjust the settings to suit your needs, click **Apply**, and then click **OK**.

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

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



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

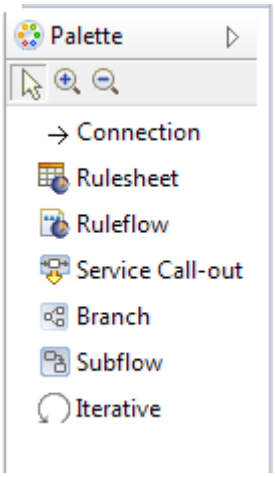


Note: The color preferences you set here could have the fill color overridden by the color property on the Properties of each object on the palette, as presented in Adding colors to Ruleflow objects on page 112. When an individual object's color reverts to its default value, white, the underlying preference color is used.

6

Objects on a Ruleflow canvas

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



These objects are used as follows:

Connections

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

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

Rulesheets

Rulesheets are the essential Corticon assets in a Ruleflow. By arranging and organizing Rulesheets in the Ruleflow, you can specify execution sequence of the Rulesheets and offer control of the iteration of the Rulesheets.

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

For example, Rulesheet `sample.ers` can be included in many Ruleflows that use the same Vocabulary, each of which may have its own use in different business processes or applications.

Ruleflows

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

Service Call-Outs

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

Service Call-outs are also the mechanism for the packaged functionality that enables Advanced Data Connectors to access services and related stored queries. See the *Data Integration Guide* for more information.

Branch

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

Subflows

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

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

Iterative

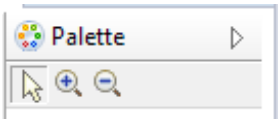
Rulesheets, Ruleflows, and Subflows can be designated for iteration by clicking the **Iterative** button in the palette, and then clicking on the target object on the canvas. To remove an iterative assignment, click on an iterative symbol in a object selected for iteration, and then press **Delete**.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

For details, see the following topics:

- Ruleflow canvas tools
- Object commands on the Ruleflow context-sensitive menu
- Properties of Ruleflow objects on a Ruleflow canvas
- Iteration
- Adding colors to Ruleflow objects

Ruleflow canvas tools

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

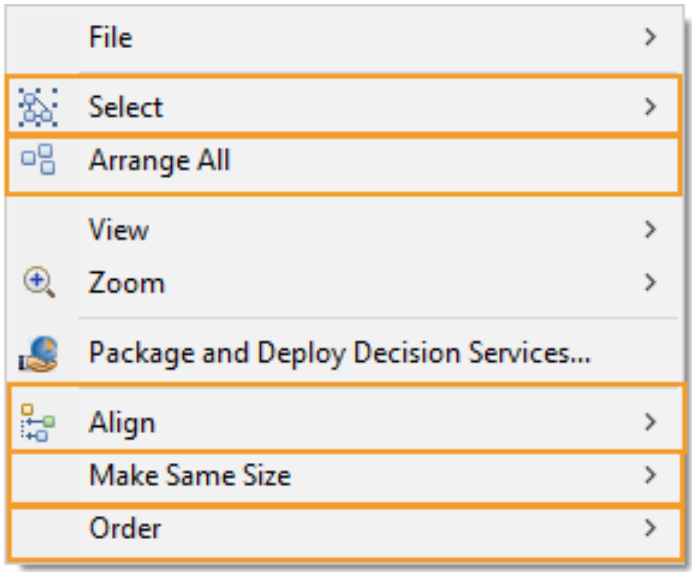


- **Select** – The default tool, the arrow cursor lets you select objects and manipulate one or more objects on the canvas.
- **Zoom** – The (+) button zooms in on the canvas; the (-) button zooms out from the canvas.

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 93 for the Ruleflow file-level commands.

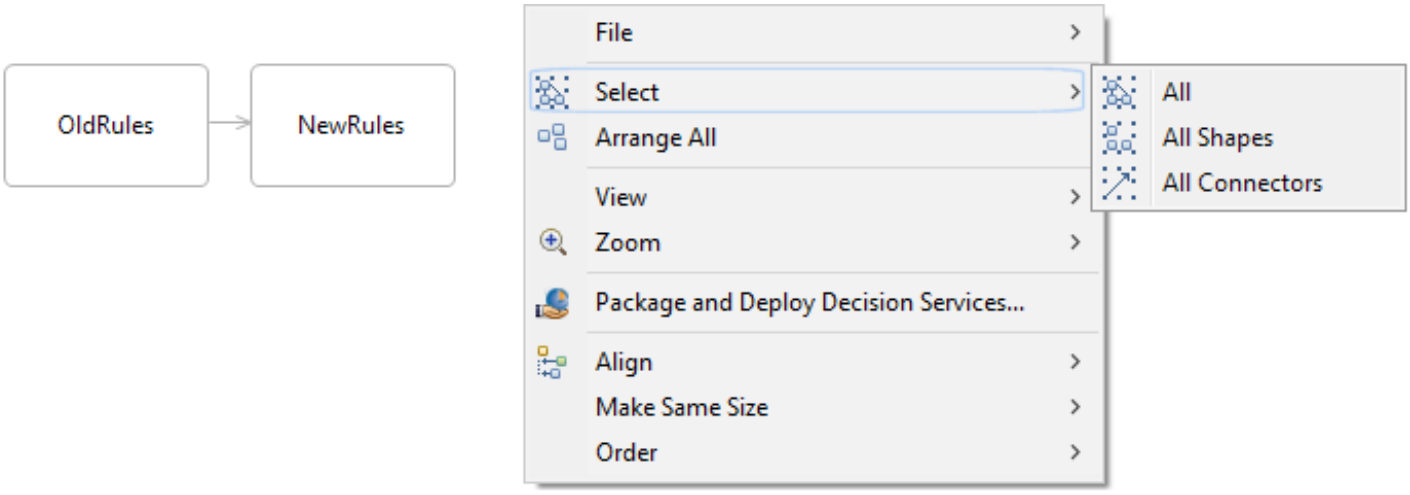


The five sections highlighted above are the object-related menu commands on the Ruleflow pop-up menu.

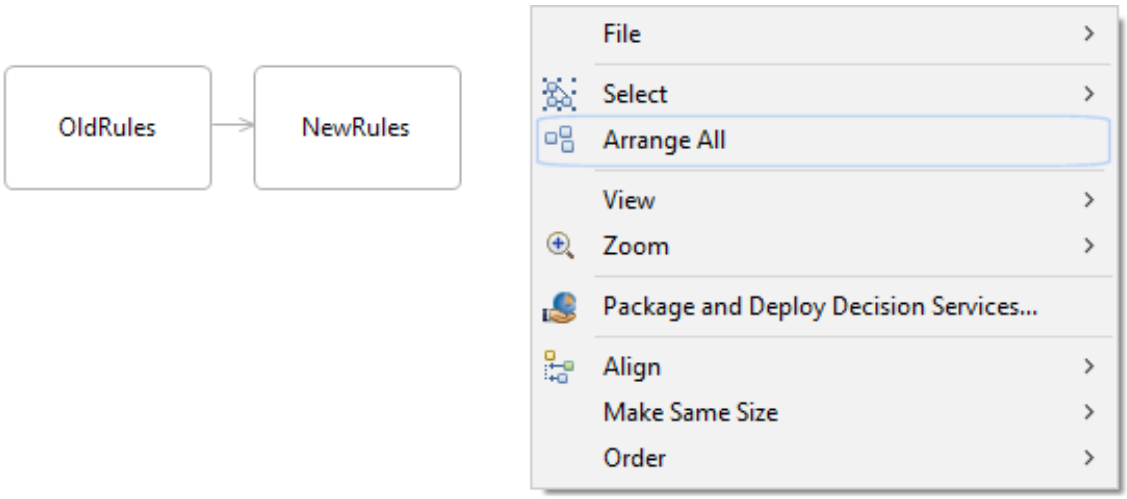
File

See Editor commands on the Ruleflow context-sensitive menu on page 93.

Select



Arrange All



View

See Editor commands on the Ruleflow context-sensitive menu on page 93.

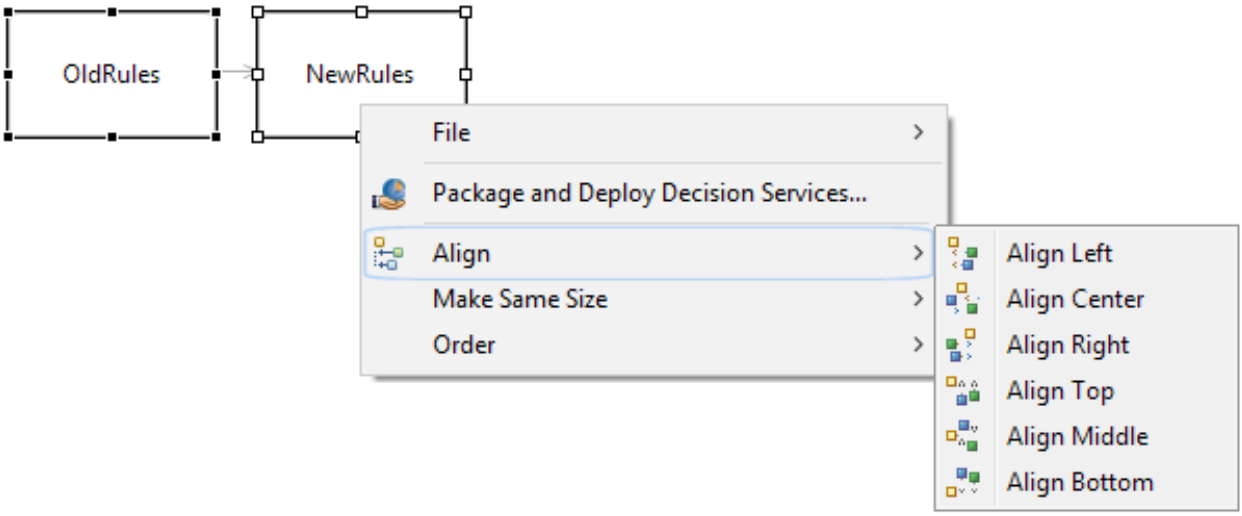
Zoom

See Editor commands on the Ruleflow context-sensitive menu on page 93.

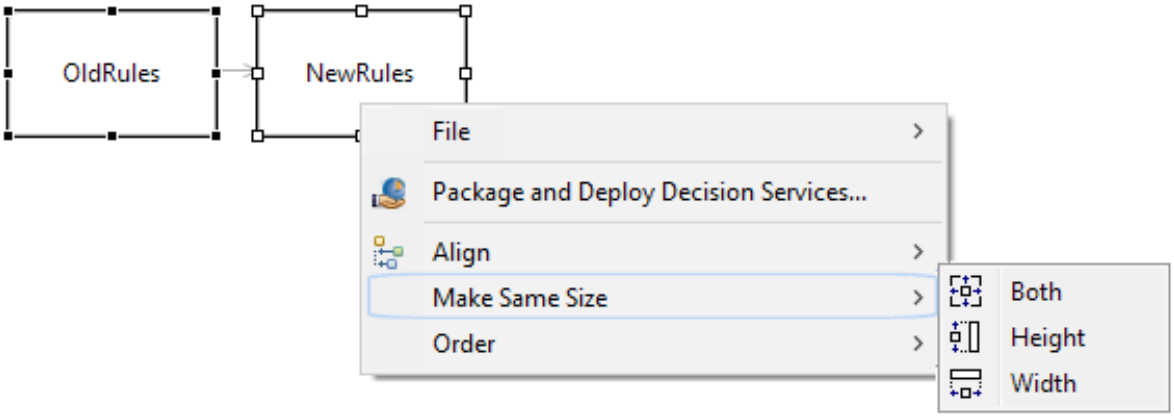
Package and Deploy Decision Services

Lets you move the Ruleflow immediately to compilation and deployment. Forces you to save all changes, then opens the **Package and Deploy Decision Services** dialog. See *"Using Studio to compile and deploy Decision Services" in the Deployment Guide.*

Align

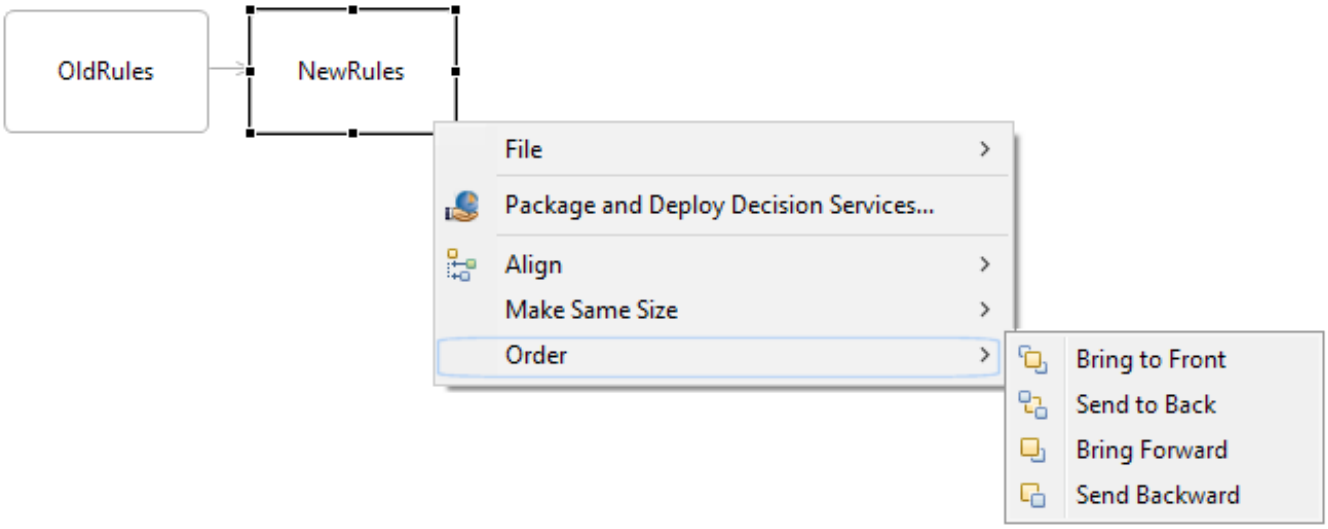


Make Same Size



- **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 98), 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 Adding colors to Ruleflow objects on page 112 for more information. You could also use preferred general colors for all aspects of canvas objects, as discussed in Ruleflow preferences on page 101
- 

- For objects that are Corticon 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 valid `.erf` file in the project. You can select a different file but you cannot change the path or the name of the original file.
- **Service Call-out** - **Service Name** pulldown menu listing service call-outs that are loaded into the workspace. The currently loaded one displays the unmodifiable **Description** specified by its author.
- **Branch Activity** - Defines the branch structure by choosing an appropriate attribute, listing values on which to branch, and then the node for each branch path. See the 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 Using the Comments view on page 149.

Iteration

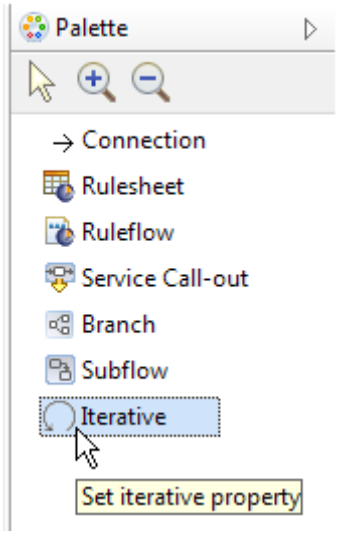
Iteration can be enabled for Rulesheet, Ruleflow, Branch and Subflow objects in a Ruleflow. When an object is set to iterate, it will repeatedly re-execute until the values derived by the object's rules cease to change. Once values in the object cease changing, the iteration stops and execution continues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

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

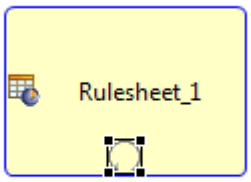
Enabling and disabling iteration

To enable iteration:

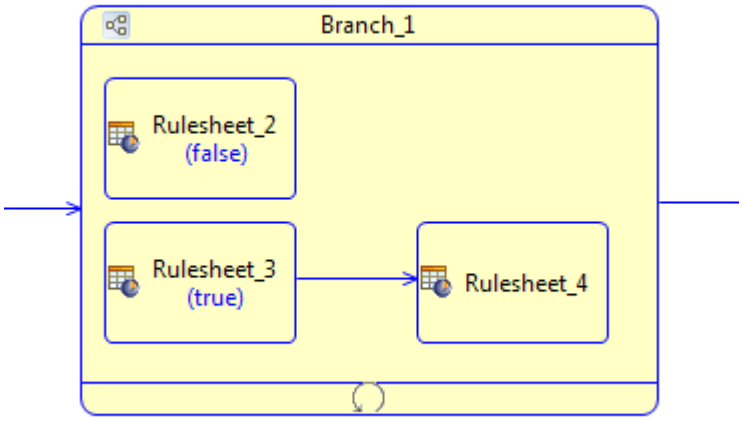
1. Click on the **Iterative** button on the palette to select it.



2. Click on the object where you want iterations enabled.
   - On a Rulesheet or Ruleflow object, click anywhere in the object. The iteration icon displays in the lower section of the object:



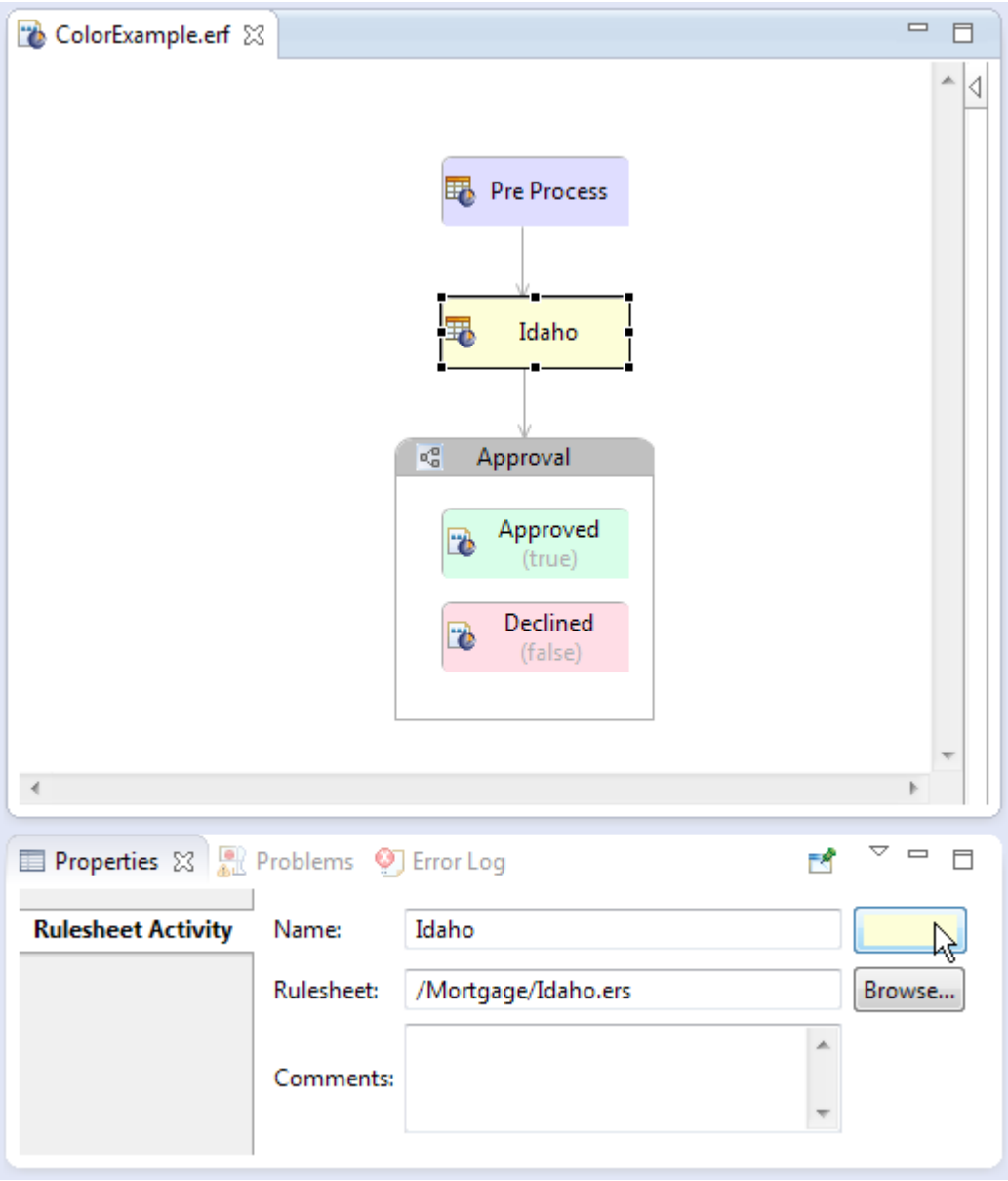
   - On a Branch or Subflow object, click on the container's title bar. An iteration section is created at the bottom of the object with the iteration icon enclosed:



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

Adding colors 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 101. When an individual object's color reverts to its default value, white, the underlying preference color is used.

7

Ruletests

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

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

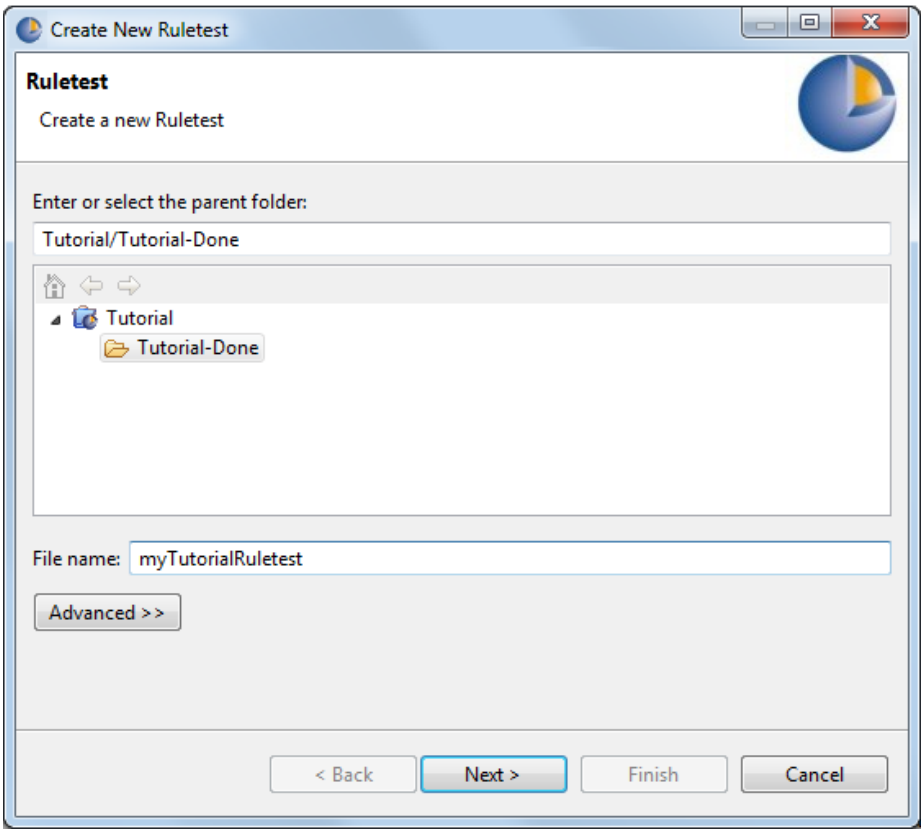
Note: Ruletest settings - Several properties control Ruletest behaviors and limits. For more information, see *"Setting Studio properties" in the Rule Modeling Guide.*

For details, see the following topics:

- Creating a new Ruletest
- Ruletest window

Creating a new Ruletest

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

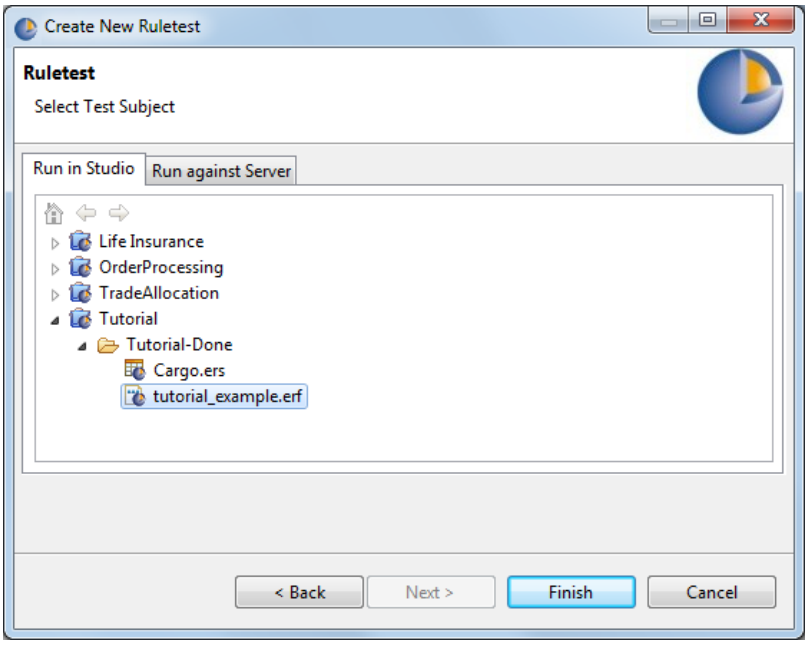


2. Select the **parent folder** for the new Ruletest, in our example `Tutorial/Tutorial-Done`.
3. Enter a file name for the new Ruletest, in our example `myTutorialRuletest`.

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

---

4. 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 on its **Run against Studio** tab, as shown:



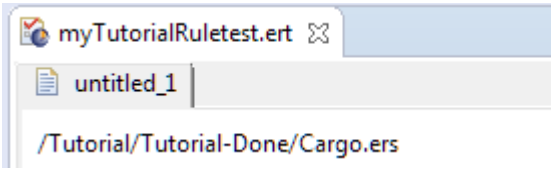
Choose the Rulesheet or Ruleflow file in the project that you want as the test subject, and then click **Finish**

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 Studio, or against an appropriate Decision Service deployed on a Corticon Server, as discussed in the following topics.

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:



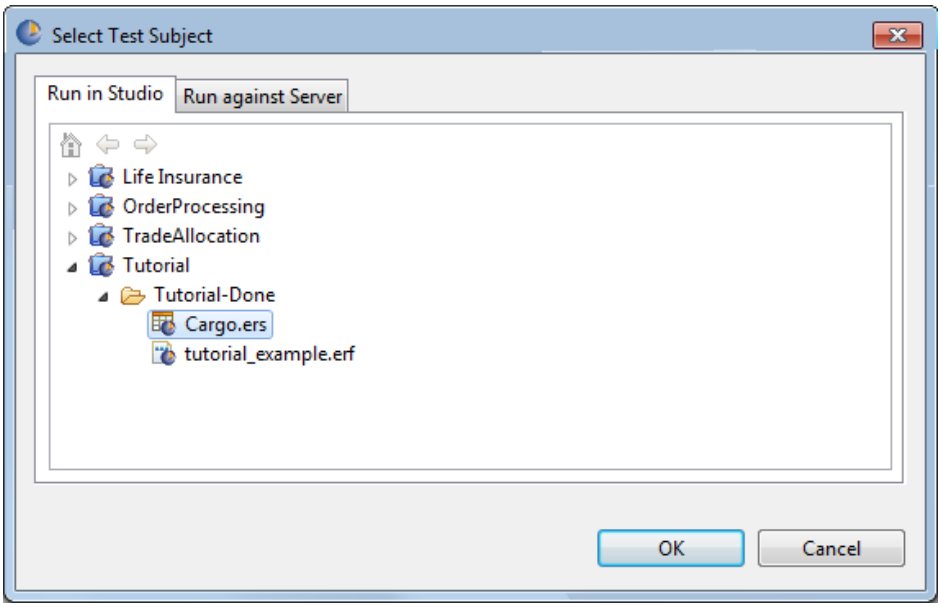
You can double-click on the Testsheet's tab to rename it.

Note: For more about Testsheet tabs, see Adding testsheets on page 135 and Renaming testsheets on page 137

Choosing a test subject in the Studio workspace

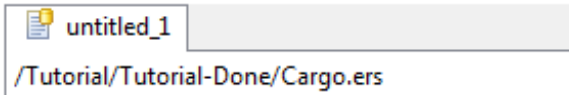
1. In the Ruletest you want to test, open it in its editor, and then choose the menu command **Ruletest > Testsheet > Change Test Subject**. The **Select Test Subject** dialog box opens on its **Run in Studio** tab.

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:

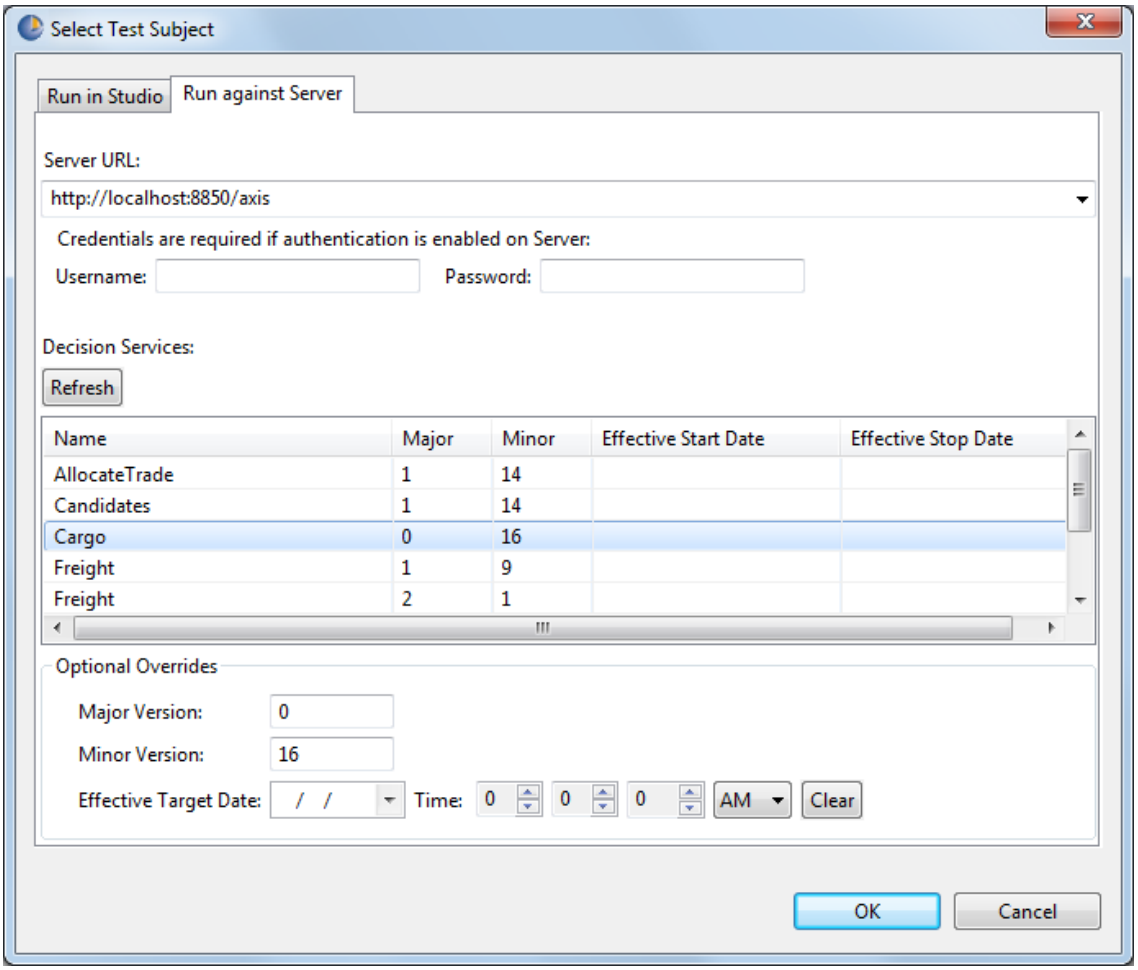


Choosing a test subject that is a deployed Decision Service

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

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

1. In the Ruletest you want to test open in its editor, choose the menu command **Ruletest > Testsheet > Change Test Subject**. The **Select Test Subject** dialog box opens.
2. Click the **Run against Server** tab to open its panel, as shown:

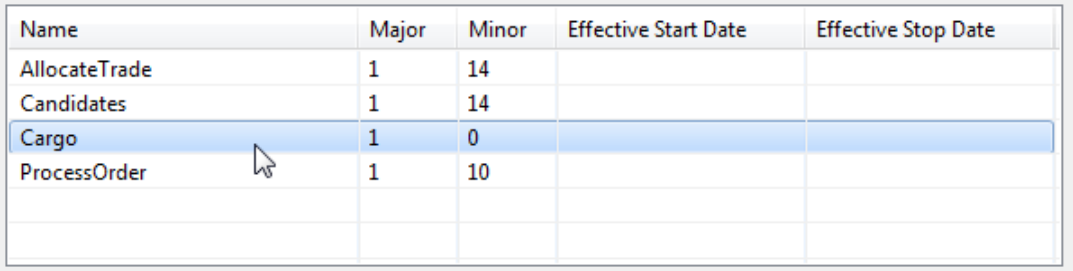


3. For the **Server URL**, enter a URL; for example, a Java Server installed (and running) on the same machine as the Studio: `http://localhost:8850/axis`. Your entry is validated when you click **Refresh**, and persisted in your Studio. Once you have persisted URLs, click on the right side of the Server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

Note: Only a few Server URLs are persisted this way. If you have a larger list that you want to edit, see Specifying server URLs for access to test subjects on page 120

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

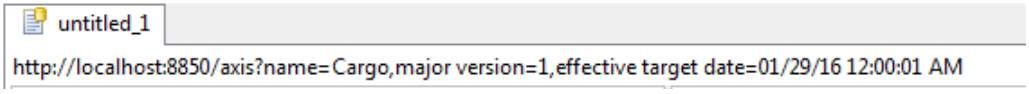
4. Click on an appropriate Decision Service for this Ruletest:



5. You can click **OK** at this point if you do not want to apply the optional overrides.
6. When the selected Decision Service was deployed with a date range defined, it is active from the effective date through the expiration date. You can apply overrides to change the test Decision Service's version or

to simulate the Ruletest's call as occurring at a specific point in time. Specify your preferred values -- major version + effective target date -- as illustrated:

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



8. Run the Ruletest.

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

Specifying server URLs for access to test subjects

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

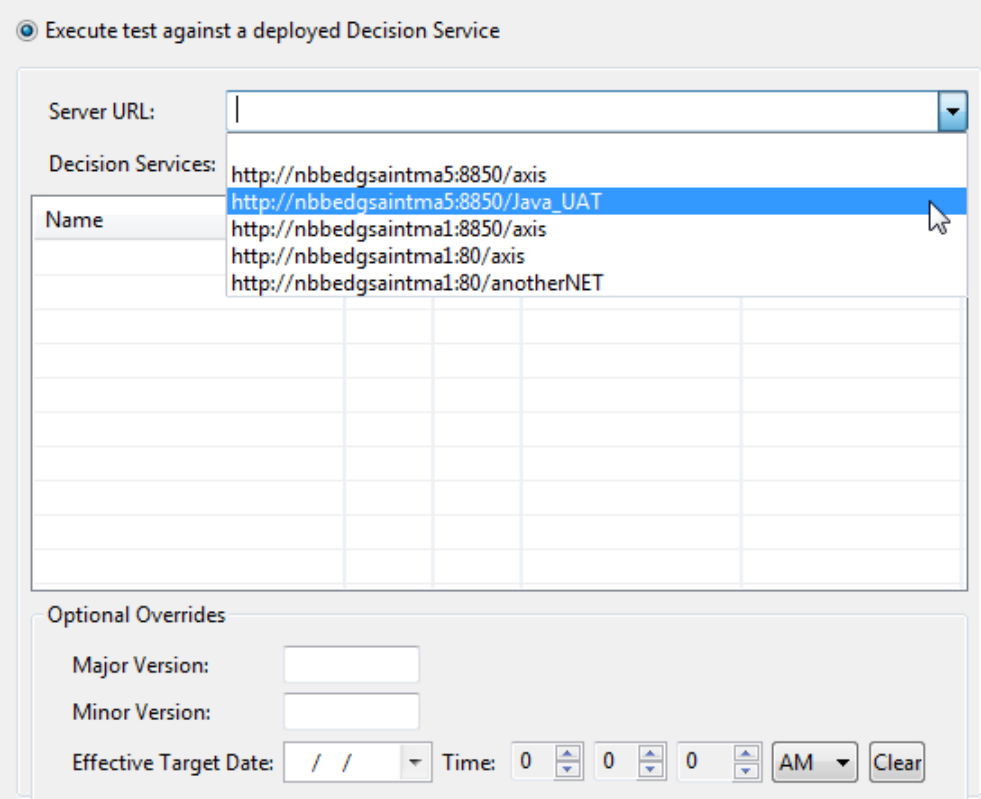
When you have more than a few remote servers that you want to access to use their deployed Decision Services as Ruletest test subjects, you can and edit their connection information to the Studio's properties file.

To extend the test subject's Server URL list:

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

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

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



Ruletest window

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

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

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

Ruletest menu commands

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

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

- **Testsheet**
  - **Add Testsheet** - Inserts a new Testsheet.
  - **Remove Testsheet** - Deletes the specified Testsheet.
  - **Link To Previous Testsheet** - Causes the Input panel of the second Testsheet to be populated with the date from the Output panel of the first.

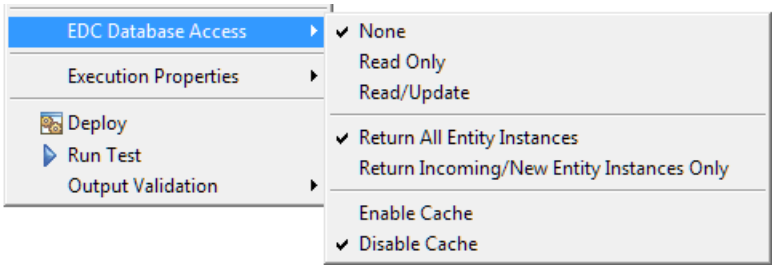
- **Change Test Subject** - Opens a window that lets you select a new Rulesheet or Ruleflow to test in Corticon Studio, or a Decision Service to test where it is deployed on a Corticon Server.
- **Cut Testsheet** - Cut the active Testsheet.
- **Copy Testsheet** - Copy the active Testsheet.
- **Paste 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.
- **Comment** - 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**
  - **XML/SOAP** - Import a valid CorticonRequest XML document into Corticon Studio as a Ruletest.
  - **JSON** - Import a valid CorticonRequest JSON document into Corticon Studio as a Ruletest.
- **Data**
  - **Set to Null** - Resets the selected Testsheet tree node to null.
  - **Go to Entity** - Displays an entity when an association tree node is selected.
  - **Sort Entities** - Sorts entity nodes alphabetically by name. One entity must be selected.
  - **Properties** - Displays the Ruletest Properties window.
  - **Input**
    - **Export Request XML** - Exports the active Testsheet's Input pane as a `CorticonRequest` XML document.
    - **Export Request SOAP XML** - Exports the active Testsheet's Input pane as a `CorticonRequest` XML document with SOAP envelope.
    - **Export Request JSON** - Exports the active Testsheet's Input pane as a `CorticonRequest` JSON document.
    - **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.
    - **Generate Data Tree** - Constructs the minimum Input data structure necessary to test the chosen Rulesheet. Uses the Rulesheet's scope for guidance.
  - **Output**
    - **Export Response XML** - Exports the active Testsheet's Output pane as a `CorticonResponse` XML document.
    - **Export Response SOAP XML** - Exports the active Testsheet's Output pane as a `CorticonResponse` XML document with SOAP envelope.
    - **Export Response JSON** - Exports the active Testsheet's Output pane as a `CorticonResponse` JSON document.

    - **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 XML** - Exports the active Testsheet's Expected pane as a `CorticonResponse` XML document.
    - **Export Response SOAP XML** - Exports the active Testsheet's Expected pane as a `CorticonResponse` XML document with SOAP envelope.
    - **Export Response JSON** - Exports the active Testsheet's Expected pane as a `CorticonResponse` JSON document.
    - **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.
  - **Export WSDL** - Create a WSDL directly from the Input pane of a Ruletest. For more information about using WSDLs to integrate Decision Services, see the *Deployment Guide*
- **EDC Database Access** - *See the following section for details.*
- **Execution Properties** - Toggles suppression of each of the message types and test output in test results. *See the section below for more information.*
- **Deploy** - Compiles the Ruletest target without executing it.
- **Run Test** - Compiles (if needed) and executes Ruletest.
- **Output Validation > Validate** - Reruns the color-coded validation of the Output and Expected data. See *"Using the Expected Panel" in the Rule Modeling Guide.*.

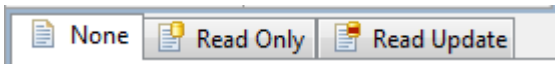
- **Run All Tests** - Executes all Testsheets in the Ruletest.
- **Report** - Creates an HTML report and launches your browser for viewing. See Creating a Ruletest Report.
- **Remove Invalid Nodes** - Discards all invalid nodes in Input, Output, and Expected columns in all Testsheets in the Ruletest.

EDC Database Access options

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



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



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

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

For more information, see the topic *"How data from an EDC Datasource integrates into rule output" in the "Advanced EDC topics" section of the Data Integration Guide.*

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

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

Execution Properties

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

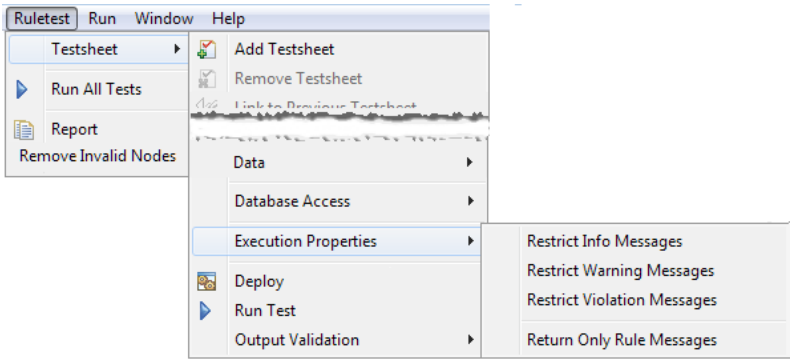
```
com.corticon.server.restrict.rulemessages.info
com.corticon.server.restrict.rulemessages.warning
com.corticon.server.restrict.rulemessages.violation
```

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

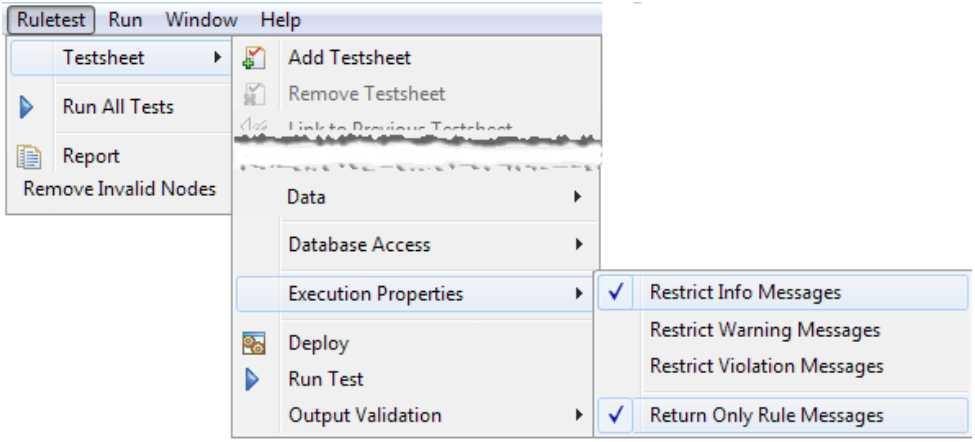
The following execution property suppresses the creation of the output pane, and displays just rule messages:

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

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



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



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

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

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

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

Ruletest toolbar

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

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

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

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

Commands on a Testsheet context-sensitive menu

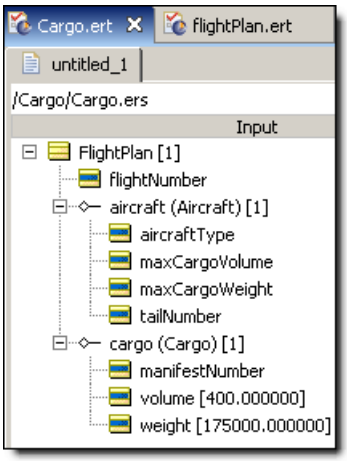
When you right-click in a Testsheet, the Ruletest context-sensitive menu opens, displaying the following commands and toggles as appropriate to certain columns and data types:

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.

Command	Columns	Comments
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 *Documenting Rule Assets* for more information about this feature.
Properties	Anywhere on sheet	Opens the Ruletest's **Properties** panel which provides the following tabs: • **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 *"Identity Strategies" in the Relational database concepts chapter of the Data Integration Guide.*

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.

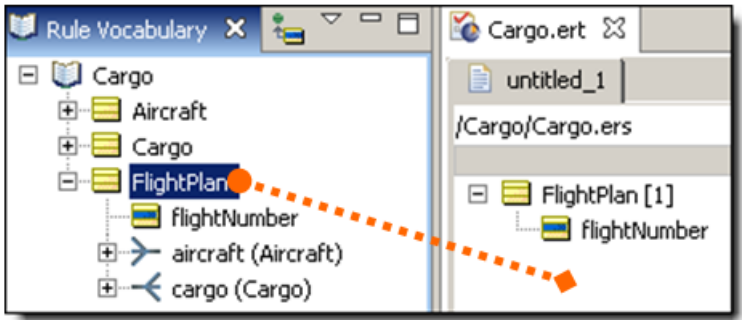


Populating the input panel

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

Adding entities to the test tree

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



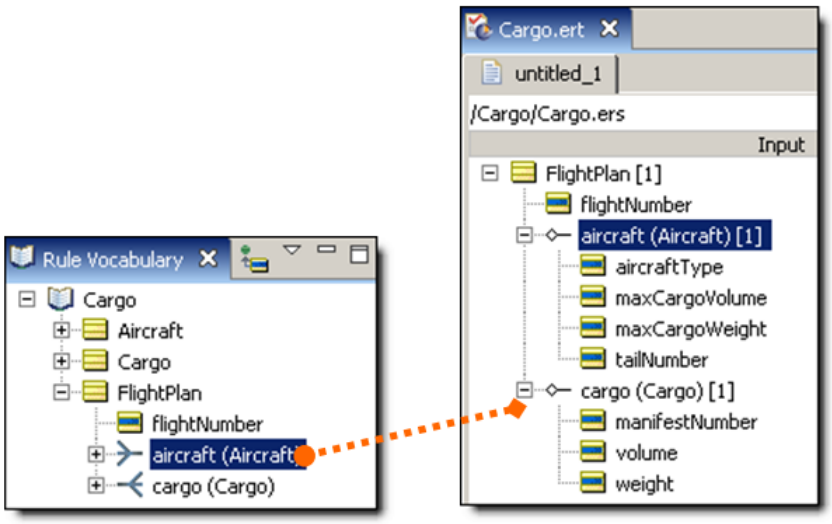
Important: Entities may be deleted by pressing the **Delete** key or by selecting **Delete** from the right-click pop-up menu.

Creating associations in the test tree

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

To create an association:

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

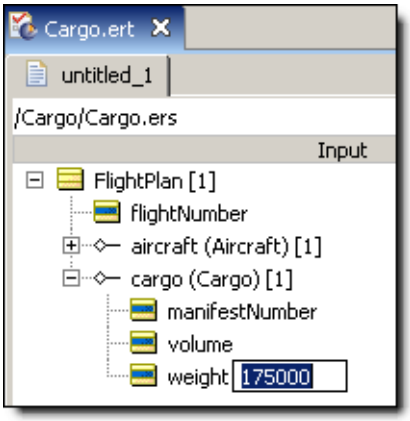


2. Verify that the child entity has the appropriate “indented” structure underneath the parent entity, indicating the association has been created correctly.

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

Assigning attribute values in the test tree

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



3. Either:

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

Automatically generating a test tree

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

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

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

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

Executing tests

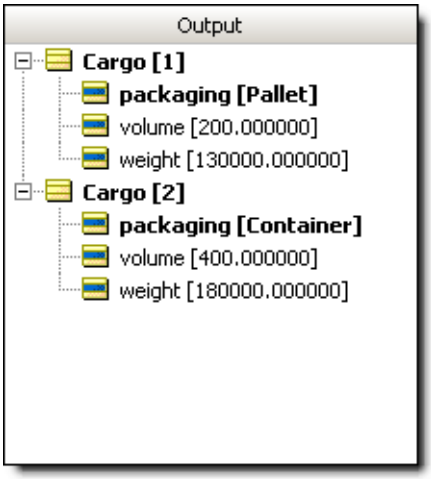
Before a Ruletest can be executed, the corresponding Rulesheets (whether being tested directly or as contained in Ruleflows) must be compiled. This compilation process does not occur until the Ruletest is run. Depending on how many Rulesheets and rules must be compiled, this compilation step may take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

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

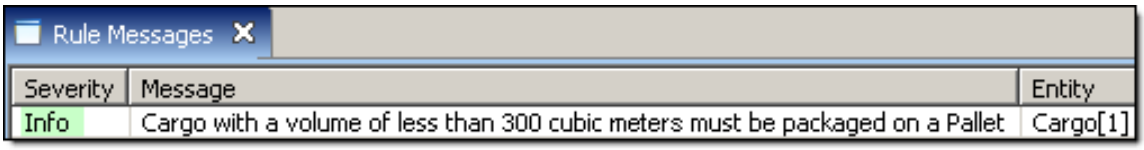
Once a Ruletest has been executed, it will rerun without recompiling as long as the rules have not changed since the last execution. Changing the Input test data does not require recompilation.

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

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



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



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

Sequence of message posting

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

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

Sorting messages

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

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

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

Format of output data

Data in the Output pane is displayed in the same format as the Input. You can adjust settings for some preferences:

- **Decimal scale** - Default precision for Decimal values. 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`. In the `brms.properties` file, set the Studio Test's decimal scale:

  ```
  decimalscale=2
  ```

  When set to `2`, the rounded value is `4.61`.

- **Messaging Style** - You can change the messaging style to force full-time hierarchical or flat Output, regardless of the Input structure.

  In the `brms.properties` file, set the Studio Test's XML messaging style:

  - `Hier` (hierarchical)
  - `Flat`
  - `Autodetect`

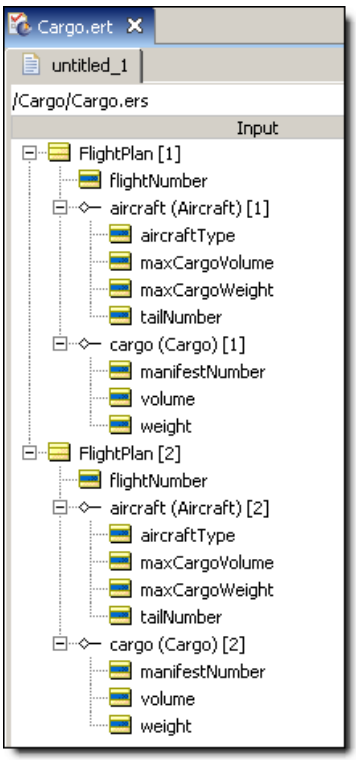
  The default value is `Hier`:

  ```
  com.corticon.designer.tester.xmlmessagingstyle=Hier
  ```

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

Creating multiple test scenarios on the same testsheet

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



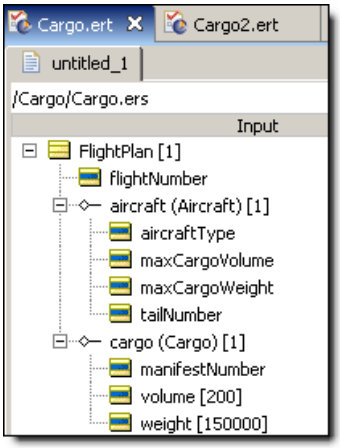
2. New elements are automatically numbered to ensure a unique ID for each.
3. Additional descriptive text can be added to each item in the **Input** column node by right-clicking the item, and then entering the type and text you want. As there can only be one comment per item, once you have added a comment, 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.

Creating multiple test scenarios as a set of testsheets

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

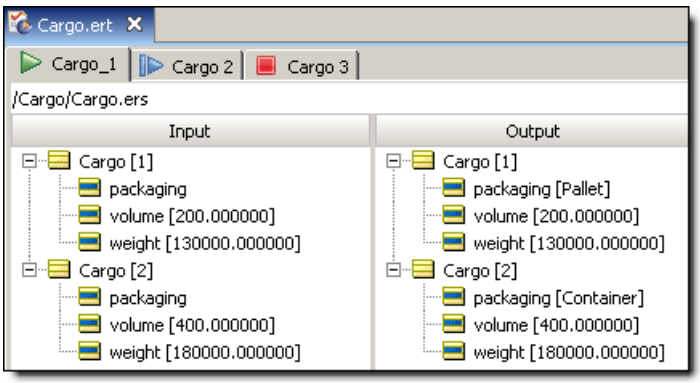
1. Each **Input** panel must be linked to a Rulesheet or Ruleflow.
2. Each **Input** panel will generate its own **Output** panel. Each of these Testsheet tabs can be renamed to help you distinguish them.



Creating a sequential test using multiple testsheets

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

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



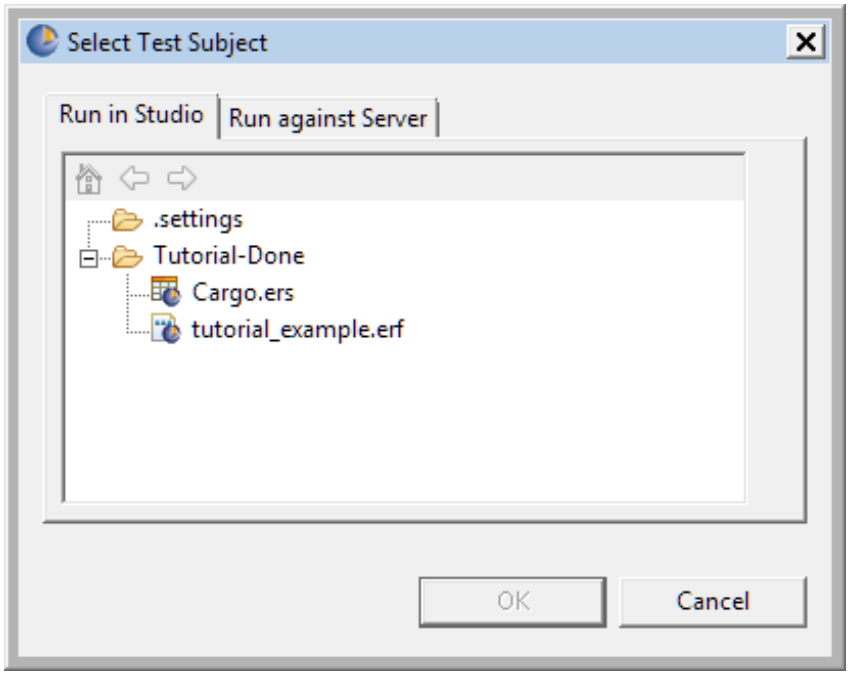
2. Assume we want to test three Ruleflows (named `Cargo`, `Cargo2` and `Cargo3`) in sequence.
3. Create a new Ruletest as before and link it to Ruleflow `cargo.erf` as usual.
4. Execute the Ruletest to generate data in the Output panel.
5. Add a new Testsheet using  and link it to Ruleflow `cargo2.erf`.

6. Link this new Testsheet to the previous Testsheet using . The data from the Output panel of the previous Testsheet should appear in the Input panel of the new Testsheet.
7. Repeat for as many new Testsheets as needed to model the sequence. You can execute one sheet at a time by selecting from the toolbar, or execute all at once by selecting **Ruletest** > **Run All Tests** from the menubar

Adding 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 on its **Run in Studio** tab, as illustrated:



For the test subject, either:

- Click on a listed Rulesheet or Ruleflow, as described in Choosing a test subject in the Studio workspace on page 117
- Choose a target on a server as described in Choosing a test subject that is a deployed Decision Service on page 118

Click **OK**.

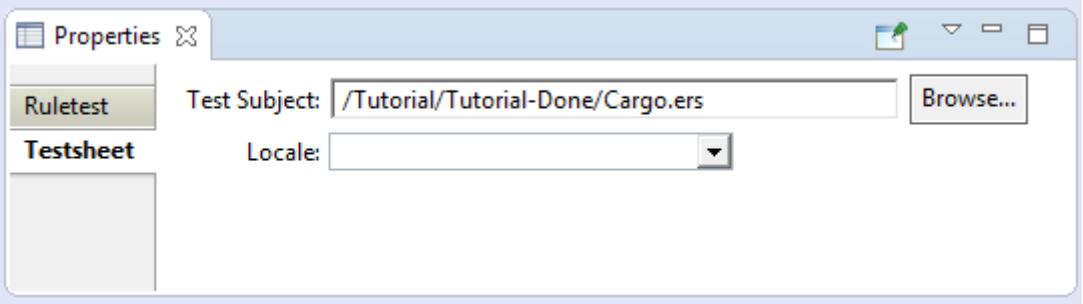
Corticon Studio creates a tab at the end of the current set of tabs, assigns the new tab a default name, and then opens the new Testsheet.

Setting the locale for a testsheet

The Studio enables you to pass the locale for a request into its Tester. The setting applies to tests run in Studio and run against Server. The chosen locale property is then stored with the Testsheet, and any exported XML or JSON files from the Testsheet. If the Locale is not set, then a Locale value is not sent in the request. The default value is the Server default.

To set the locale on a Testsheet:

1. Open the Ruletest in its editor, and then choose a Testsheet on which you want to set the locale.
2. In its Properties section, choose the **Testsheet** tab, as illustrated:



3. On the **Locale** dropdown menu, choose the locale for this Testsheet.
4. Save the Rulestest.

When you run the Testsheet, the selected locale is applied. For example, where the locale is set to `French`, the requests exported from Tester have the following header lines:

XML

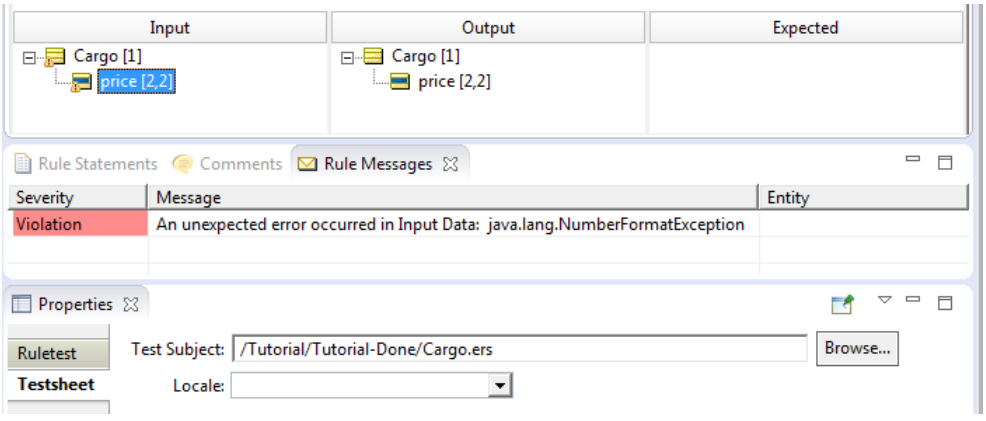
```
...
<ExecutionProperties>
   <ExecutionProperty name="PROPERTY_EXECUTION_LOCALE" value="fr" />
</ExecutionProperties>
```

JSON

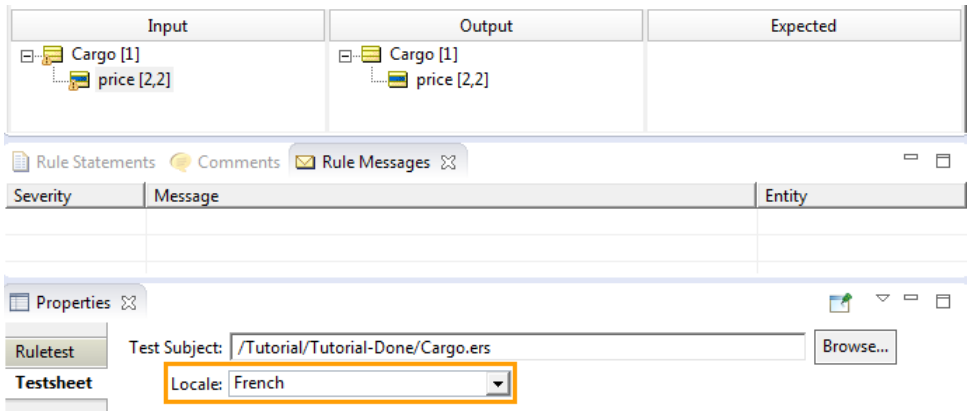
```
{
  "__metadataRoot": {"#locale": "fr"},
  "Objects":
```

Use case in the Ruletest editor

For a given property that is a decimal value, illustrated as `price` in this example, entering the value `2,2` is shown to be an error in the default locale:



Setting the locale to French, the comma delimiter for a decimal value is correct. The value is valid:

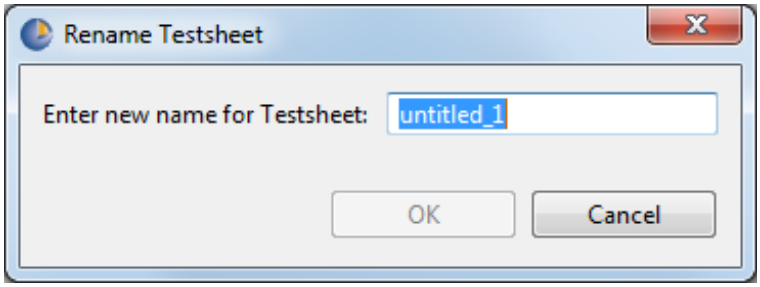


Renaming testsheets

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:



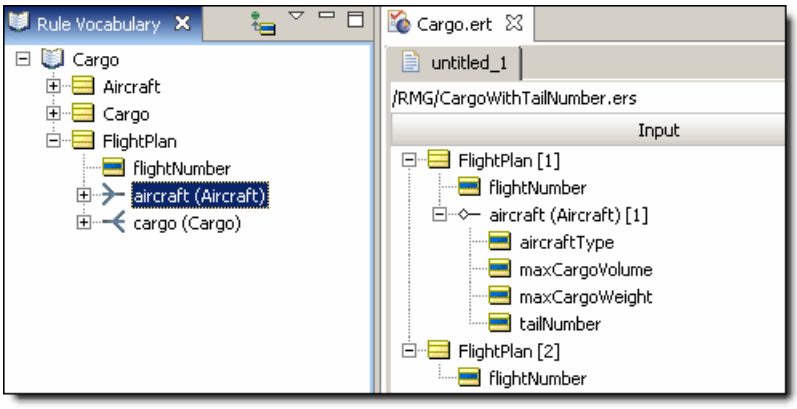
Enter the unique name you want to assign within this Ruletest.

Note: Ruletest and Testsheet names must comply with the guidelines described in File naming restrictions on page 18.

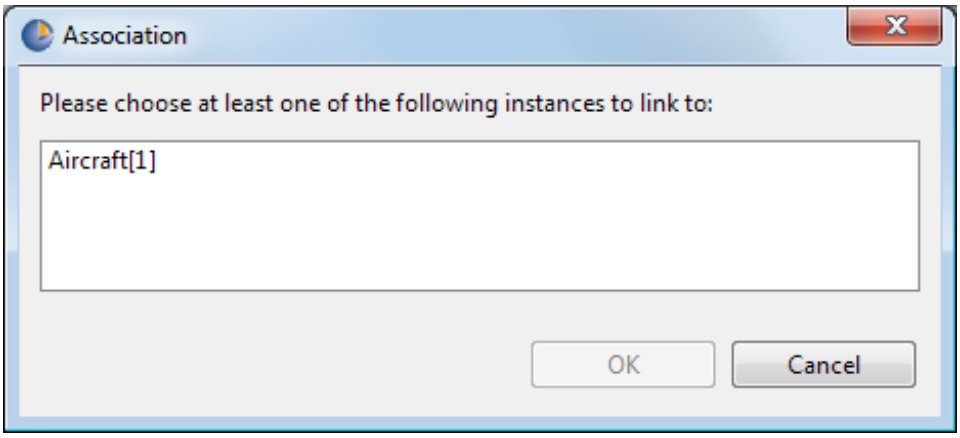
Associating one child entity with more than one parent

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

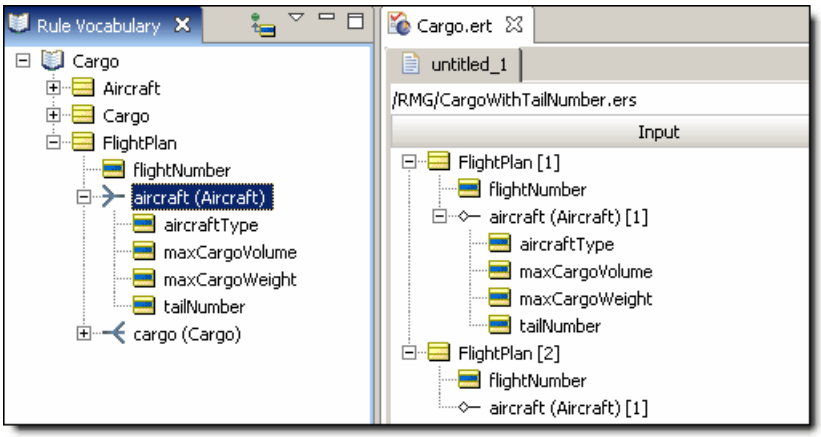
1. First, ensure the parent-child (source-target) association supports a multiplicity of many-to-many or many-to-one.
2. Drag and drop the parent entities (here, `FlightPlan`) as usual.
3. Create an association between aircraft and `FlightPlan[1]` as described in **Creating Associations**.
4. To associate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



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



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



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

Saving a Ruletest

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

Importing an XML or SOAP document to a testsheet

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

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

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

Importing a JSON document to a testsheet

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

Exporting a testsheet to an XML document

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

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

Input

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

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

Note: You can chose to exclude transients when you export Testsheets to XML using the Exclude Transients option. For example, select **Ruletest > Testsheets > Data > Input > Exclude Transients** to set your preference to exclude transients. Toggle the option off to include transients in the exported document.

You must replace the `decisionServiceName` parameter, shown in line 2, with the Decision Service this request will call.

```
<?xml version="1.0" encoding="UTF-8"?>
<CorticonRequest xmlns="urn:Corticon"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
decisionServiceName="InsertDecisionServiceName">
  <WorkDocuments>
    <Cargo id="Cargo_id_1">
      <weight>1000</weight>
      <volume>10</volume>
      <container xsi:nil="true" />
    </Cargo>
    <Cargo id="Cargo_id_4">
      <weight>1000</weight>
      <volume>10</volume>
      <container xsi:nil="true" />
      <needsRefrigeration>true</needsRefrigeration>
    </Cargo>
  </WorkDocuments>
</CorticonRequest>
```

Output

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

Expected

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

Exporting a testsheet to a SOAP message

To produce the request as an XML payload within SOAP-compliant envelope information, open a Ruletest in Studio, and then select **Ruletest > Testsheets > Data > Input > Export Request SOAP** to sexcept that it encloses the XML payload within SOAP-compliant envelope information. The XML payload is identical to the XML generated by the **Export to XML** option, as shown above.

Note: You can chose to exclude transients when you export Testsheets to SOAP using the Exclude Transients option. For example, set your preference to exclude transients. Toggle the option off to include transients in the exported document.

Input

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

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

Note: On line 4, you must replace "`InsertDecisionServiceName`" with the name of the deployed Decision Service that this request will call..

Output

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

Expected

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

Exporting a testsheet to a JSON document

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

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

Note: You can chose to exclude transients when you export Testsheets to JSON using the Exclude Transients option. For example, select **Ruletest > Testsheets > Data > Input > Exclude Transients** to set your preference to exclude transients. Toggle the option off to include transients in the exported document.

Input

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

```
{"Objects": [
    {
        "weight": "1000",
        "container": null,
        "volume": "10",
        "__metadata": {
            "#id": "Cargo_id_1",
            "#type": "Cargo",
        }
    },
    {
        "weight": "1000",
        "needsRefrigeration": "true",
        "container": null,
        "volume": "10",
        "__metadata": {
            "#id": "Cargo_id_4",
            "#type": "Cargo",
        }
    }
]}
```

Output

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

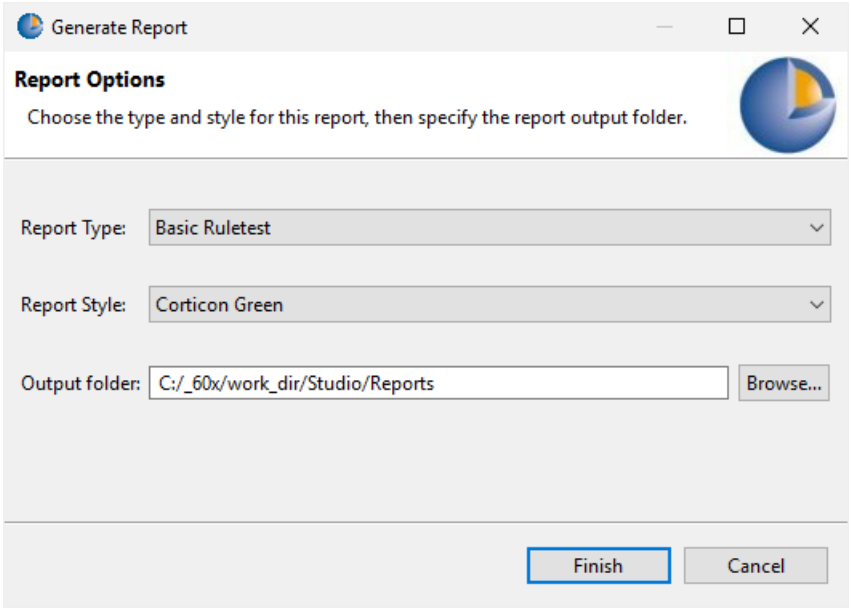
Expected

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

Creating a Ruletest report

1. Select the menu command **Ruletest** > **Report**.

   The **Generate Report** dialog opens, as shown:



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.

For information about creating custom transformations and stylesheets for reports, see the topic *Reporting Framework chapter of the Rule Modeling Guide.*

8

Documenting 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
- Adding asset-level comments
- Adding item-level comments
- Using 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:

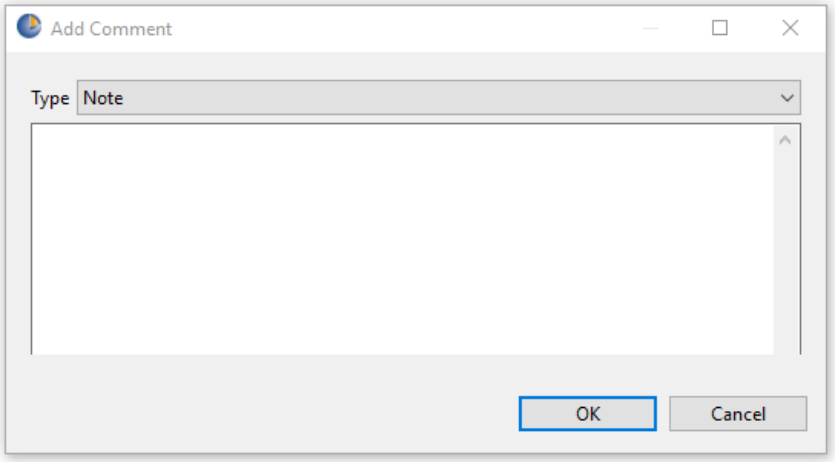
Asset	Notes
Vocabularies	• Comments are added in the Vocabulary editor, not from the Project Explorer or Rule Vocabulary views. • Comments can be added at the root level for the file, but not for Custom Data Types or Datasource definitions.
Rulesheets	• Select a range of items (such as `a1:c2`) to add 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., a1:c3). • 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	• 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	• 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.

Adding asset-level comments

To add comments to a rule asset file:

1. In Corticon Studio, open the file in its editor.
2. On the toolbar, click the **Comments** button  . The **Add Comment** dialog opens:

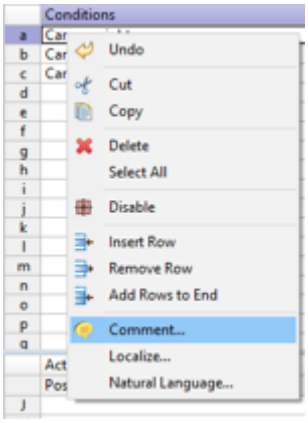


3. On the drop-down menu, select the type of comment you want: **To Do**, **Change Log**, **Note**, or **Needs Review**.
4. Enter a comment in the field provided.
5. Click **OK**. The comment is captured and displayed in the Comments View.

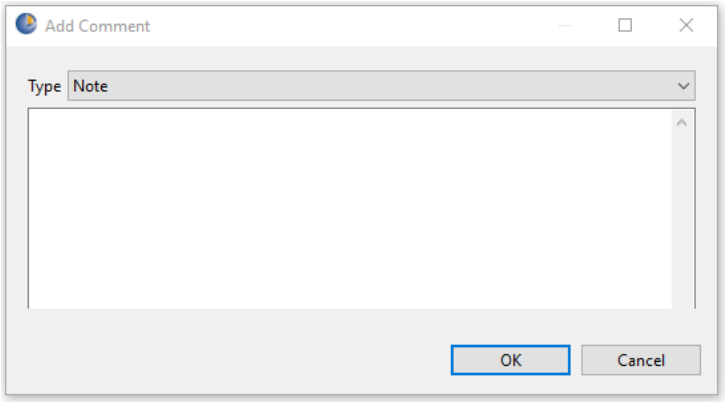
Adding item-level comments

To add comments to items in a rule asset:

1. In Corticon 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:



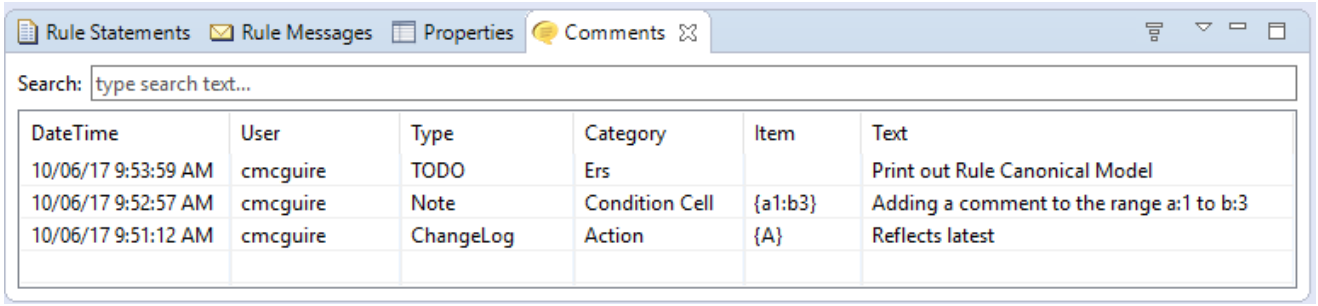
4. Click the **Type** field's dropdown button, and then choose the type of comment you want to enter: **To Do**, **Change Log**, **Note**, or **Needs Review**.
5. Enter a description or annotation in the field provided.
6. Click **OK**. The comment is captured and displayed in the Comments View.

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

Using 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:



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.

9

Localizing Corticon Studio

Localizing your rule modeling and processing environment involves three related functions:

- Displaying the Studio **program** in your locale of choice. This means switching the Corticon Studio user interface (menus, operators, system messages, etc.) to a new language. Consult with Progress Corticon support or your Progress representative to learn more about available and upcoming Corticon localization packages.
- Displaying your Studio **assets** in your locale of choice. This means switching your Vocabularies, Rulesheets, Ruleflows, and Ruletests to a new language. This part is described here.
- Requests submitted to a Corticon Server can specify an execution property that indicates the **locale of the incoming payload** so that the server can transform the payload's decimal and date values to the decimal delimiter and month literal names of the server, run the rules, and return the output formatted to the submitter's preference. See *"Handling requests and replies across locales" in the Deployment Guide*.

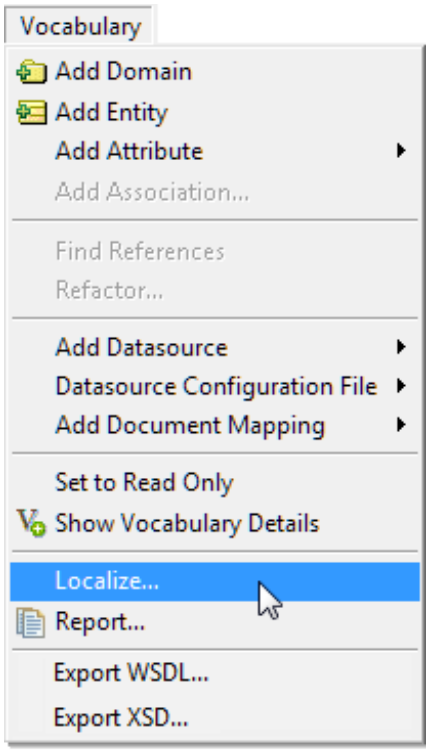
For details, see the following topics:

Localizing the Vocabulary

A Vocabulary can be localized into several different languages. If a Vocabulary includes multiple locale information, then Corticon Studio displays the locale corresponding to the operating system's current locale.

To localize a Vocabulary, select **Vocabulary>Localize…** from Corticon Studio's menubar as shown below:

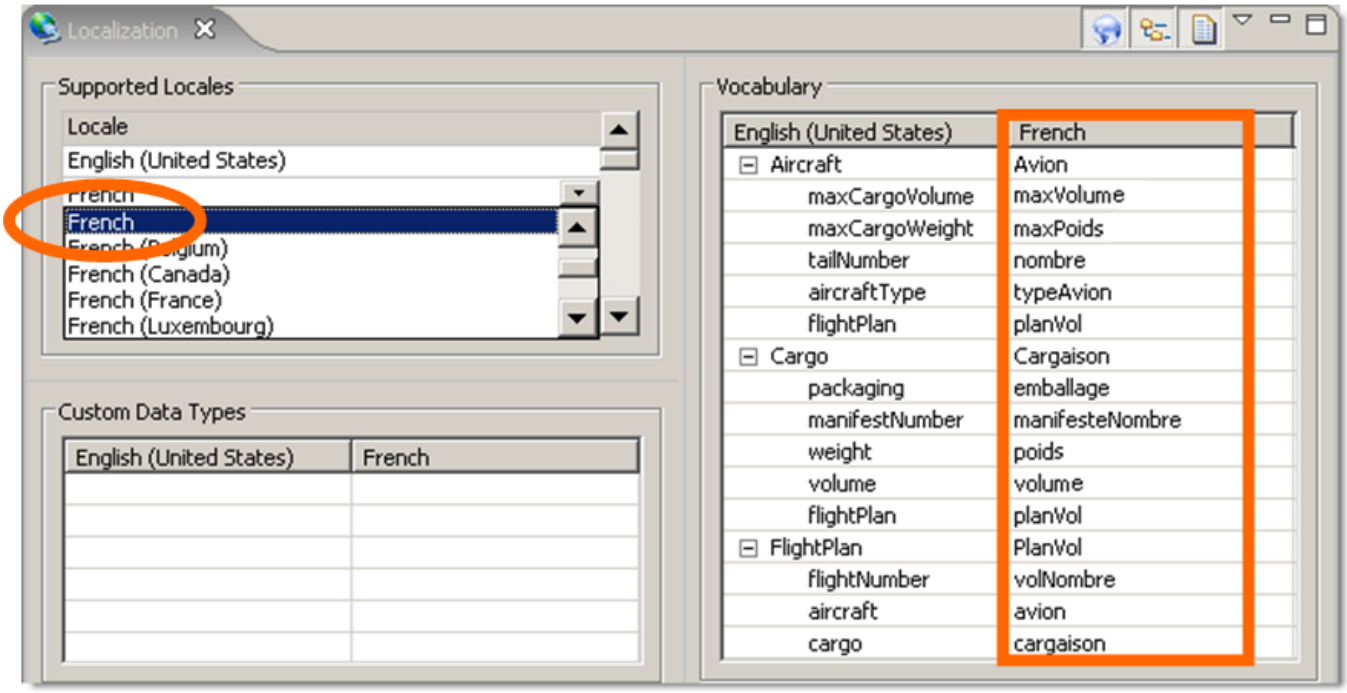
Figure 8: Localize a Vocabulary



Corticon Studio displays the Vocabulary Localization window as shown below.

Notice that we've selected French in the second line of the **Supported Locales** pane, circled below in orange. This choice causes a second column to appear to the right in the **Vocabulary** pane (shown below in an orange rectangle).

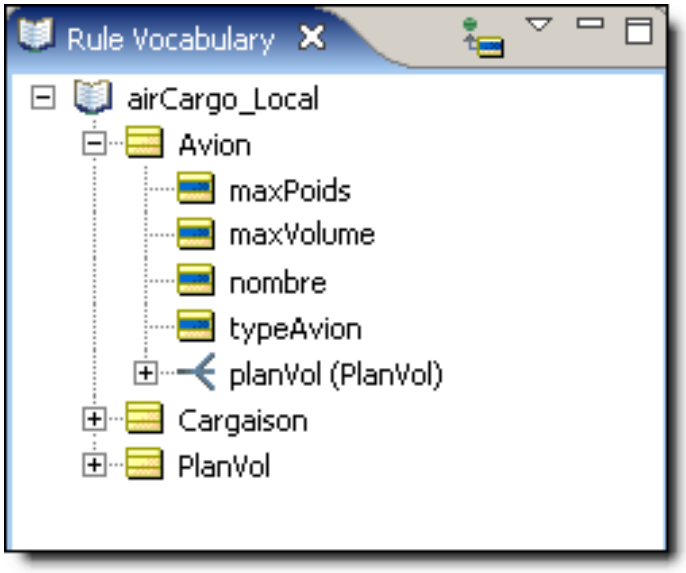
Figure 9: Selecting and populating a second locale



In the **French** column, we must manually add French translations for each Vocabulary term, including association role names. These translated terms must be unique from the English or other base version shown in the left-hand column.

With a localized Vocabulary, now switch your operating system to French locale. Different OS's have different methods for switching locales - consult your OS help or documentation for assistance.

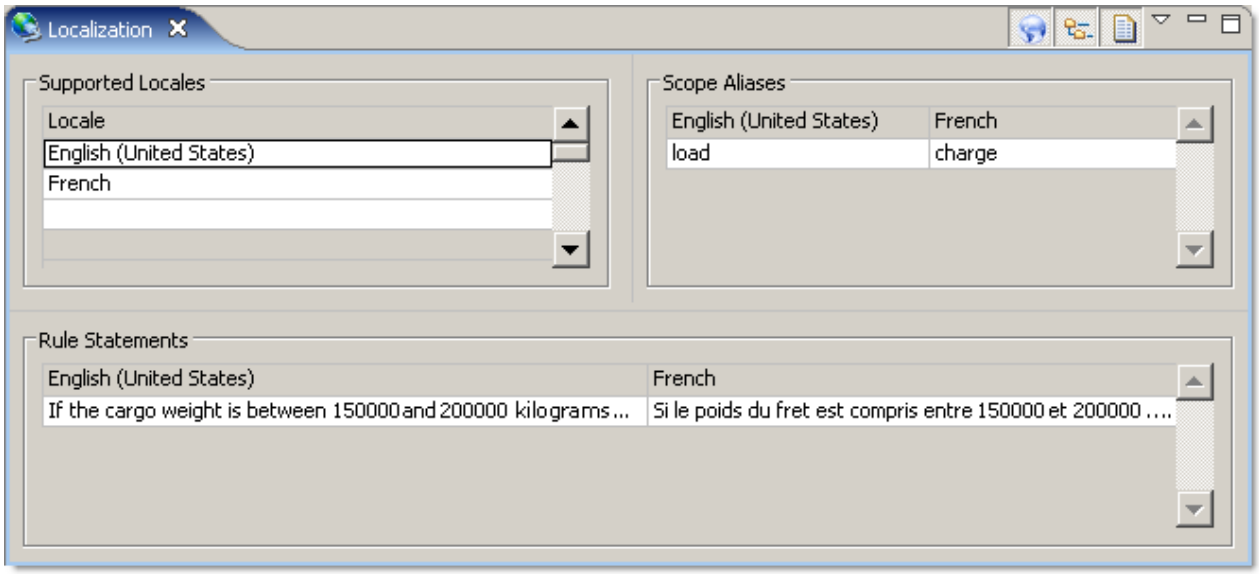
Figure 10: A Vocabulary displaying its French translation



Localizing the Rulesheet

When you create a new Rulesheet or Ruletest using a localized Vocabulary, those assets will be localized too. The **Rulesheet >Localize…** menu selection allows you to further localize the Rulesheet by translating Scope aliases and Rule Statements, as shown below:

Figure 11: A Rulesheet displaying its French translation

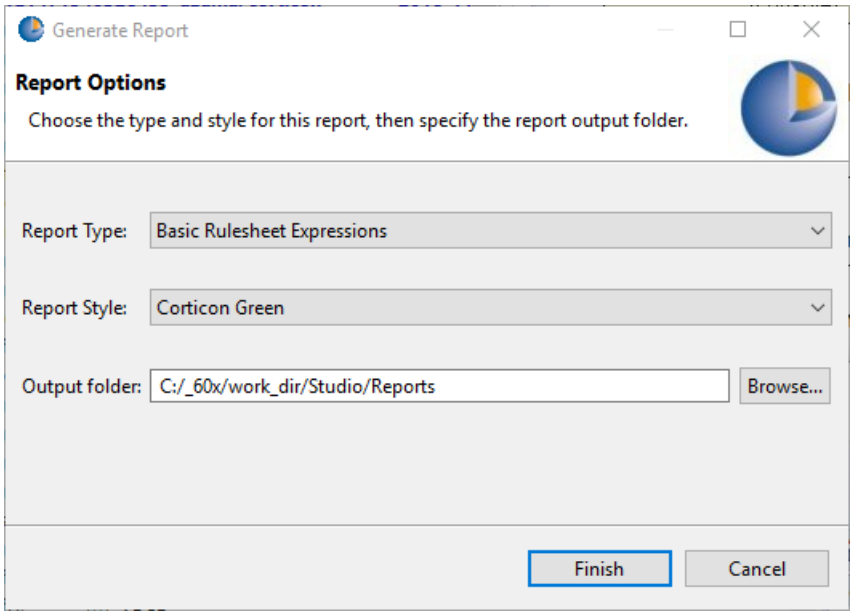


10

The Corticon Studio reporting framework

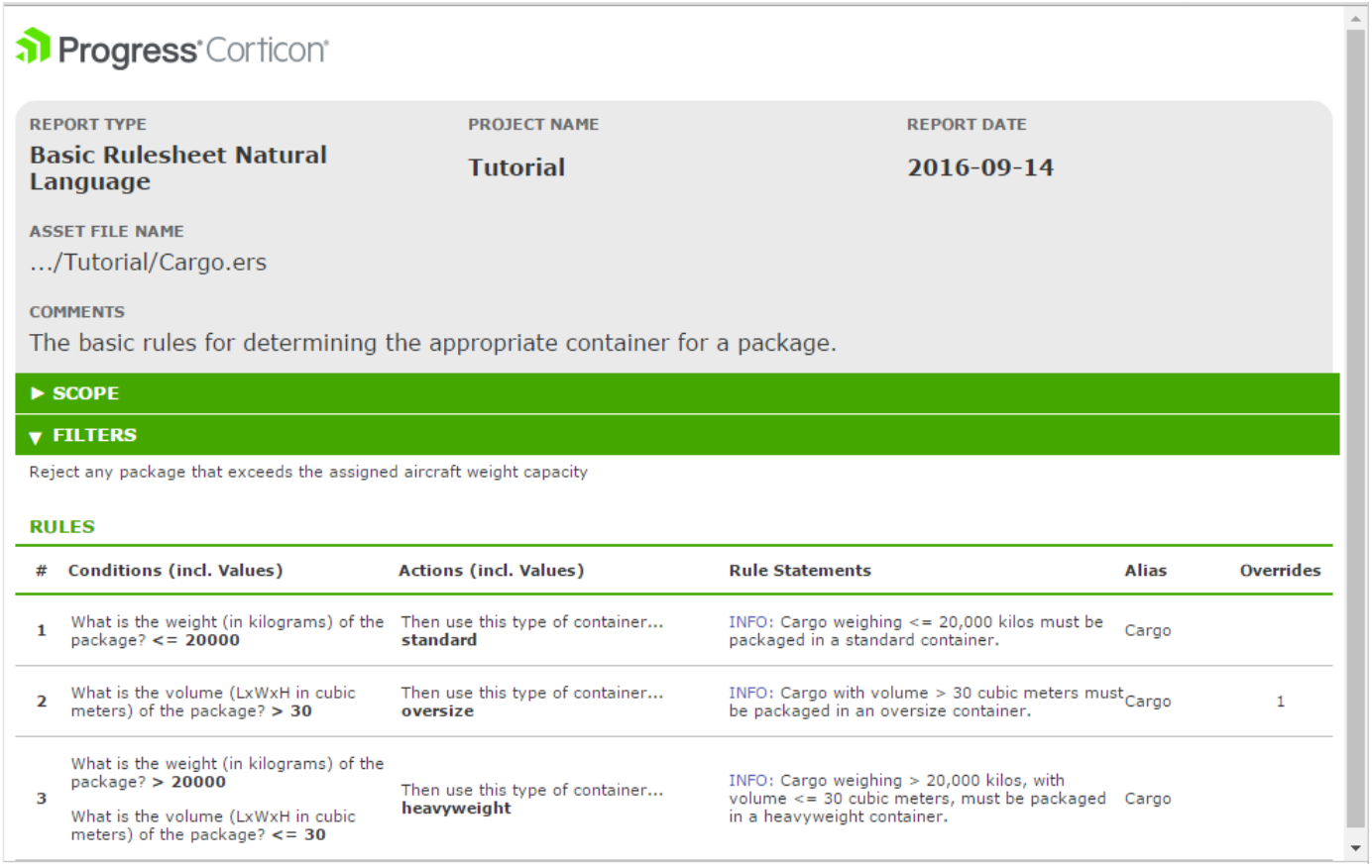
Corticon Studio lets you create reports on each of the assets in a project: Vocabulary, Rulesheets, Ruleflows, and Ruletests.

You generate reports from a dialog box that lets you choose from standard types and styles of reports, and then set the output folder, as illustrated here for a Rulesheet:



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 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 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:

- Creating a Vocabulary report on page 49
- Creating a Rulesheet report on page 81
- Creating a Ruleflow report on page 97
- Creating a Ruletest report on page 143

11

Exiting Corticon Studio

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

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