What's New in Corticon

Copyright

Updated: 2018/09/27

1

Overview of Progress Corticon

Progress® Corticon® is the Business Rules Management System with the patented rules engine that enables you to automate sophisticated decision processes—without having to write code.

Progress Corticon products

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

- **Corticon Studio** is the Windows-based development environment for creating and testing business rules:
  - When installed as a standalone application, Corticon Studio provides a complete Eclipse development environment for Corticon in the **Corticon Designer** perspective. You can use this Eclipse installation as the basis for adding other Eclipse tools.
  - When installed into an appropriate existing Eclipse environment, Corticon integrates with installed Eclipse tools to enable you to develop Corticon applications in the **Corticon Designer** perspective.

  Note: Refer to the *Corticon Installation Guide* for details about integrating Corticon Studio into an existing Eclipse environment.

- **Corticon Servers** implement web services and in-process servers for deploying business rules defined in Corticon Studio:
  - **Corticon Server for Java** is supported on various application servers, and client web browsers. After you install it on a supported Windows platform, its deployment artifacts can be redeployed on various UNIX and Linux web service platforms as Corticon Decision Services.
  - **Corticon Server for .NET** facilitates deployment of Corticon Decision Services on Windows .NET Framework and Microsoft Internet Information Services (IIS).

- **Corticon Web Console** enables administration of multiple remote Corticon Servers. A Web Console server is deployed into a Progress Application Server, and then is accessed by users through authenticated web browser connections.

2

What's new and changed in Corticon 5.7.2

This section summarizes the new, enhanced, and changed features in Progress® Corticon® 5.7.2.

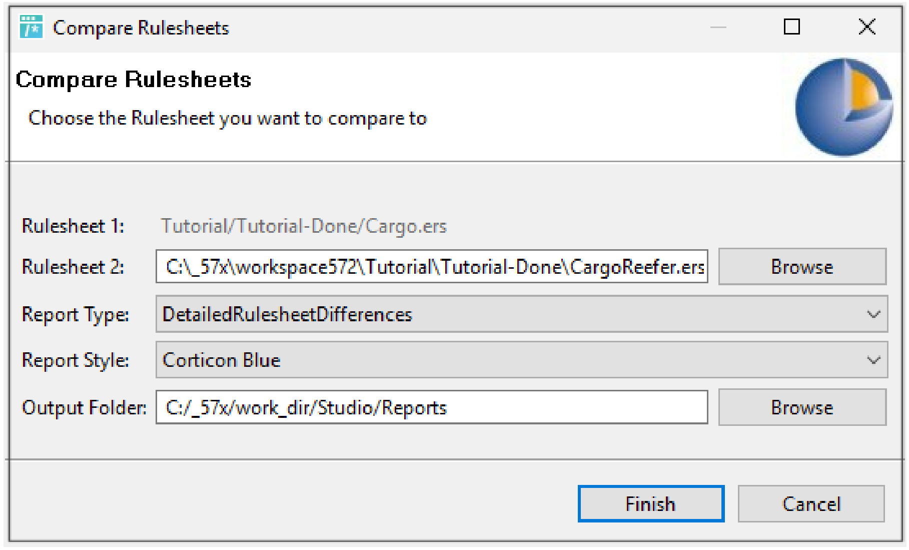
For details, see the following topics:

Reporting on differences between Rulesheet versions

Two versions of a Rulesheet can have modest changes, yet it can be difficult to see all the differences during a visual inspection of the two Rulesheets. Reporting on differences between Rulesheets provides help in debugging mistaken rule changes, and inconsistent rule definitions. For example:

- **Diagnosing a Ruletest failure** - When a Ruletest fails because of changes in newer Rulesheets, you can use Rulesheet difference reports to determine what changed, and then make changes to a Rulesheet to fix bad rules, or to indicate changes to make to your Ruletest expected results.
- **Resolving merge conflicts** - When using a source control system such as git, you may encounter situations where you want to commit a Rulesheet that someone else has changed, and discover a merge conflict. Using Rulesheet difference analysis and reports, you can see what has changed and decide how to manually merge the differences so you can commit your changes.

With one Rulesheet in the editor, choosing the menu command **Compare Rulesheets** initiates the dialog as illustrated:



Compare Rulesheets

Compare Rulesheets

Choose the Rulesheet you want to compare to

Rulesheet 1: Tutorial/Tutorial-Done/Cargo.ers

Rulesheet 2: C:_57x\workspace572\Tutorial\Tutorial-Done\CargoReefer.ers

Report Type: DetailedRulesheetDifferences

Report Style: Corticon Blue

Output Folder: C:/_57x/work_dir/Studio/Reports

The analysis is not simply a text file comparison of the source files -- it is a comparison operation that takes the list of objects in each Rulesheet, and then maps them to find matching criteria unique to specific objects in the input lists. Sets of matching criteria by context are found, and the output is a mapping of matched objects, and notation of unique 'extra' objects.

For more information, see the topic *"Comparing and reporting on Rulesheet differences" in the Rule Modeling Guide.*

3

What was new and changed in Corticon 5.7.1

This section summarizes the new, enhanced, and changed features in Progress® Corticon® 5.7.1.

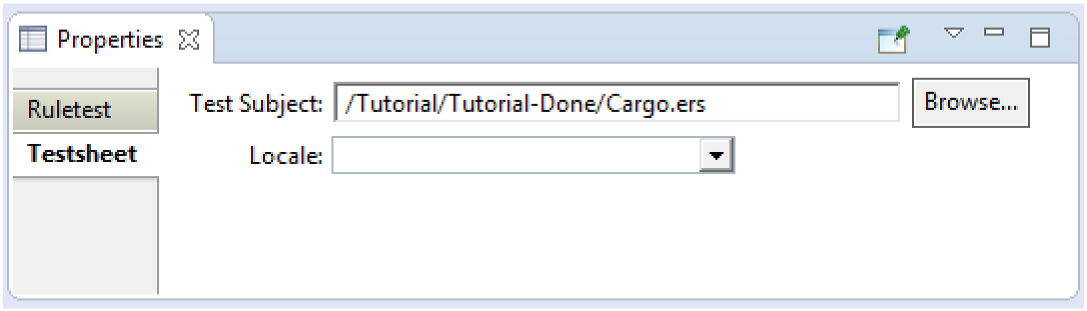
For details, see the following topics:

Testsheets can specify locale

The Studio enables you to pass the locale for a request into its Tester. Specifying locale instructs Corticon how to parse locale sensitive data in a testsheet input such as dates and numeric formats. The setting applies to tests run in Studio and run against Server. The chosen locale property is 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:



Test Subject: /Tutorial/Tutorial-Done/Cargo.ers

Locale:

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

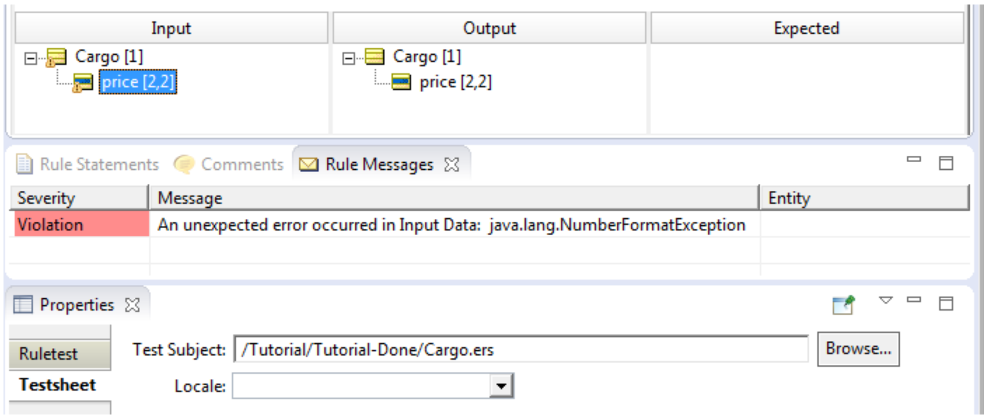
```
 ...
                <ExecutionProperties>
                <ExecutionProperty name="PROPERTY_EXECUTION_LOCALE" value="fr" />
                </ExecutionProperties>
```

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:

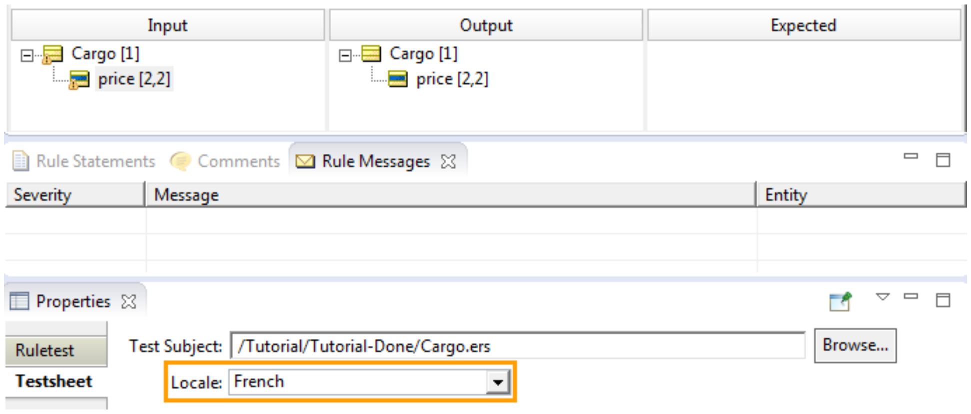


Severity	Message	Entity
Violation	An unexpected error occurred in Input Data: java.lang.NumberFormatException	

Test Subject: /Tutorial/Tutorial-Done/Cargo.ers

Locale:

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



Input	Output	Expected
Cargo [1]	Cargo [1]	
price [2,2]	price [2,2]	

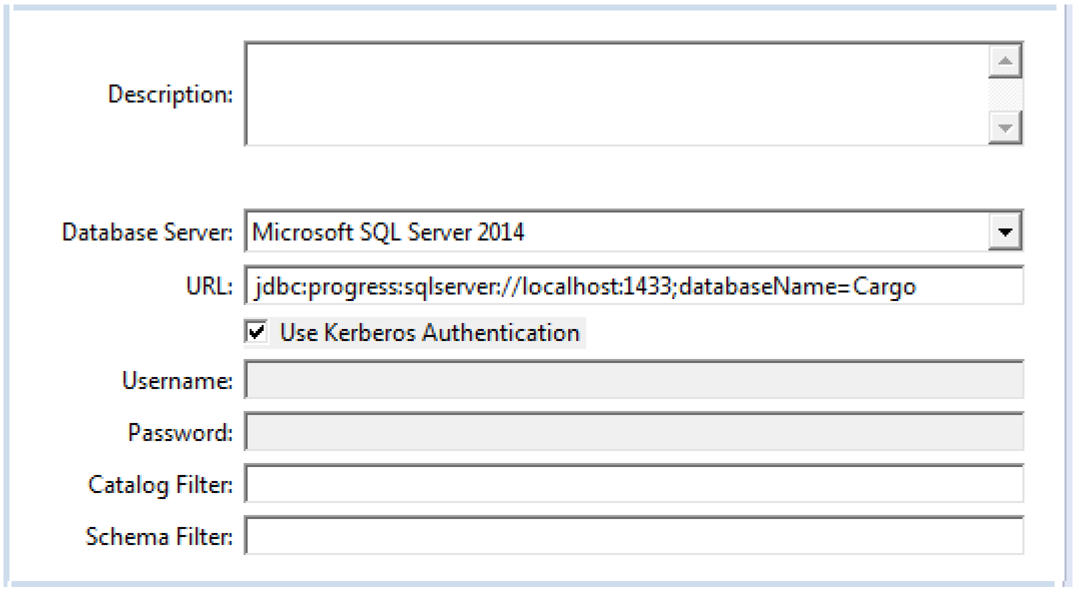
Test Subject: /Tutorial/Tutorial-Done/Cargo.ers

Locale: French

This information was added as the topic *"Setting the locale for a Testsheet" in the Quick Reference Guide.*

Database connections support Kerberos

When using Kerberos for authentication, the identity of the user running Studio or Server is used to authenticate database access. Database connections for EDC and ADC Datasources let you choose to declare that you want to use **Kerberos Authentication**, as illustrated:



Description:

Database Server: Microsoft SQL Server 2014

URL: jdbc:progress:sqlserver://localhost:1433;databaseName=Cargo

☑ Use Kerberos Authentication

Username:

Password:

Catalog Filter:

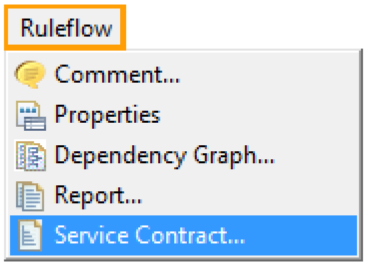
Schema Filter:

Choosing Kerberos authentication suppresses the username and password fields.

Note: Kerberos is supported on Corticon Server for Java; however, it is not supported on Corticon Server for .NET.

Generate Service Contract from a Ruleflow

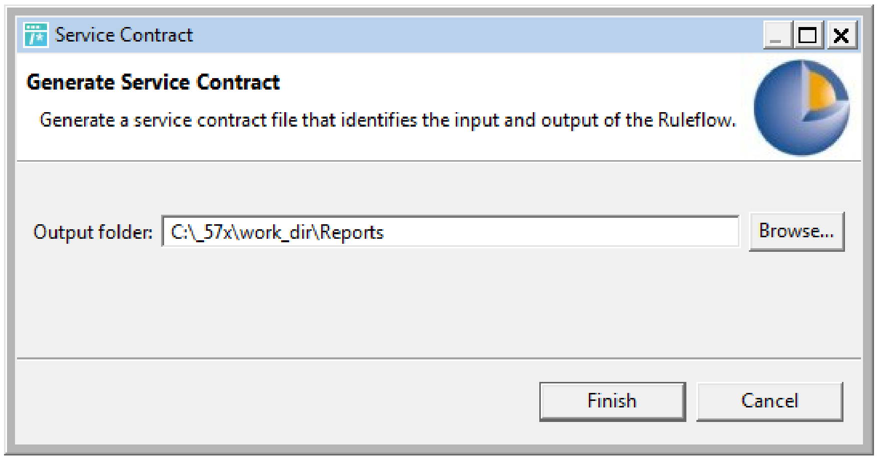
Studio now provides the ability to generate a *service contract* for a Ruleflow. This contract is a CSV file that identifies all the inputs and outputs of the Ruleflow. It is for use by system integrators when accessing a deployed Decision Service generated from the Ruleflow. You can generate a service contract from a Ruleflow by just choosing the menu command Service Contract, as ilustrated:



When you choose the Ruleflow menu command **Service Contract**, a document is produced based on the *RuleflowFileName*:

```
RuleflowFileName_ServiceContract.csv
```

at the default Studio location `[WORK_DIR]\Reports`. For example:


Service Contract

Generate Service Contract

Generate a service contract file that identifies the input and output of the Ruleflow.

Output folder: C:_57x\work_dir\Reports

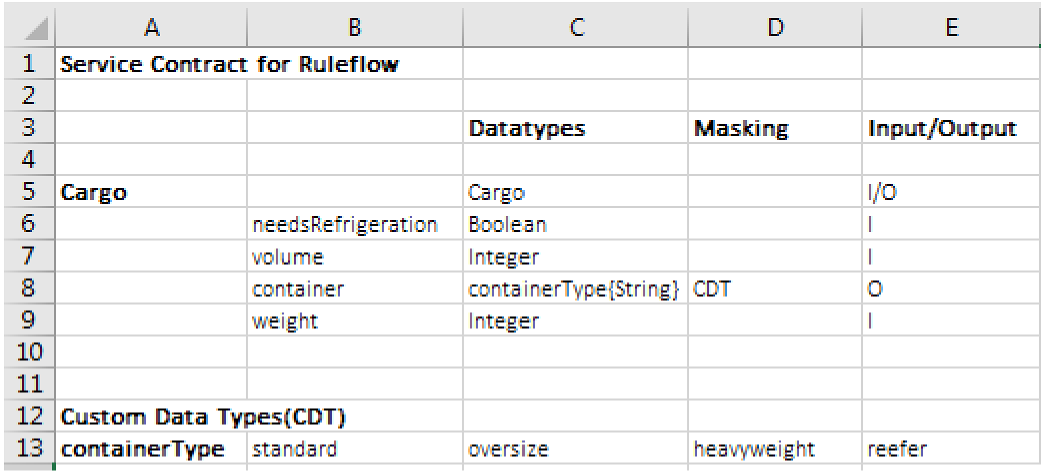
Controlling date and time format masks

You can specify date and time formats that you prefer in a generated service contract by setting properties in the `brms.properties` file, as shown:

```
#Properties for masking attributes in a Service Contract. Default values are:
# com.corticon.serviceContract.date.masking=MM/dd/yy
# com.corticon.serviceContract.dateTime.masking=MM/dd/yy h:mm:ss a
# com.corticon.serviceContract.time.masking=h:mm:ss a
#
#com.corticon.serviceContract.date.masking=
#com.corticon.serviceContract.dateTime.masking=
#com.corticon.serviceContract.time.masking=
```

Uncomment a masking property you want to set, and then provide an appropriate mask value.

When `tutorial_example_ServiceContract.csv` is opened in Excel, the CSV file looks like this:

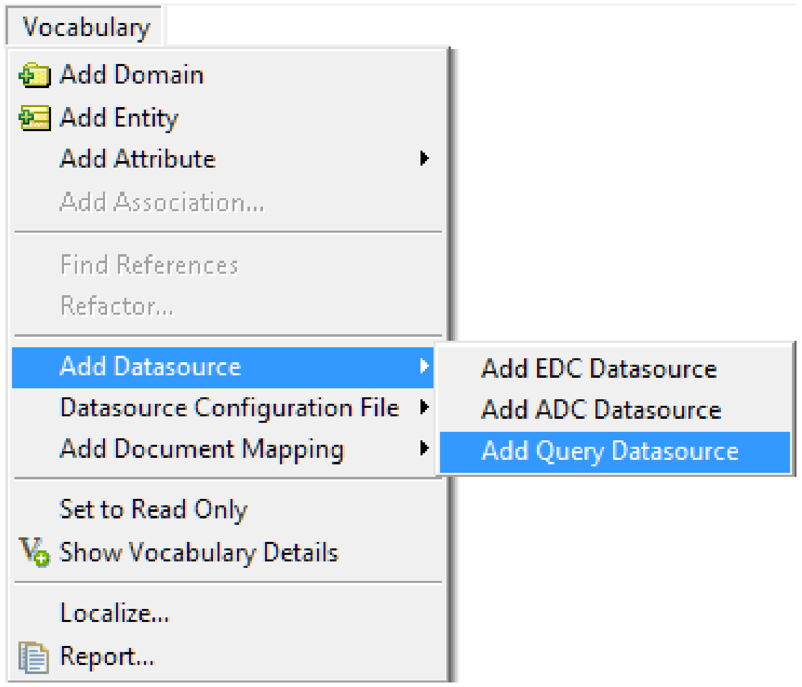


	A	B	C	D	E
1	**Service Contract for Ruleflow**				
2					
3			**Datatypes**	**Masking**	**Input/Output**
4					
5	**Cargo**		Cargo		I/O
6		needsRefrigeration	Boolean		I
7		volume	Integer		I
8		container	containerType{String}	CDT	O
9		weight	Integer		I
10					
11					
12	**Custom Data Types(CDT)**				
13	**containerType**	standard	oversize	heavyweight	reefer

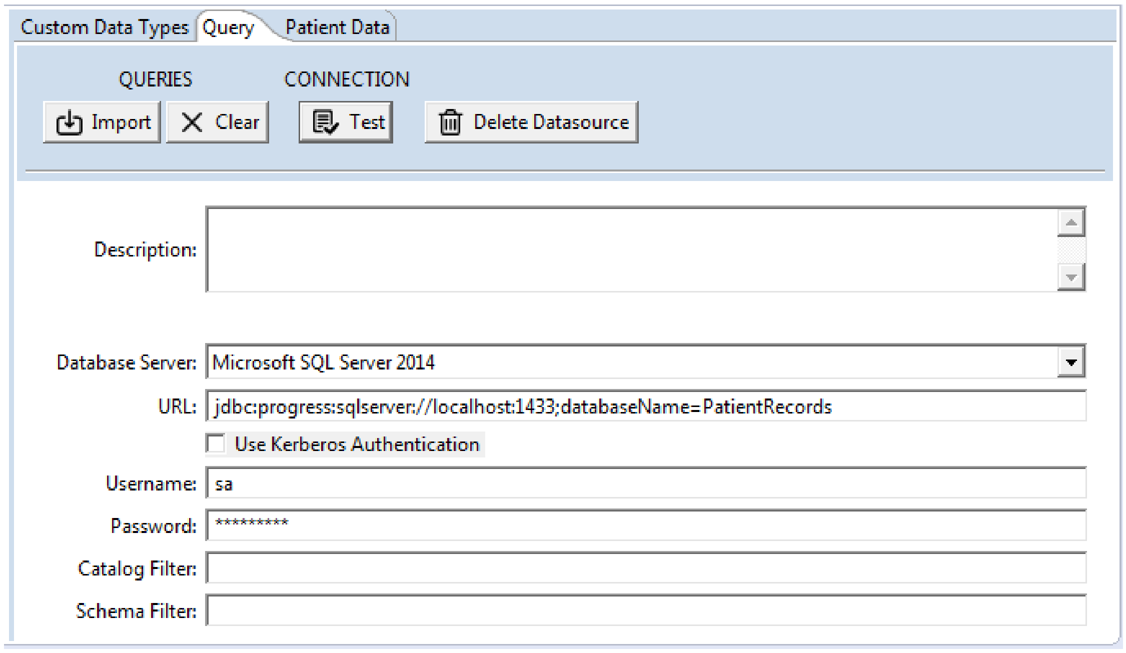
This material was added to the full documentation as the topics *"Examples of service contract reports generated from a Ruleflow"* and *"Guidelines for REST/JSON Service Contracts" in the Integration and Deployment Guide.*

New Query Datasource

In version 5.7.0, one of the ADC Datasources was declared as the repository for the Query Service for ADC and Batch queries. That technique has been improved to allow a separate Datasource just for queries. Choosing **Add Query Datasource**, as shown...



... which opens the **Query** tab, as shown...



Description:

Database Server: Microsoft SQL Server 2014

URL: jdbc:progress:sqlserver://localhost:1433;databaseName=PatientRecords

Use Kerberos Authentication

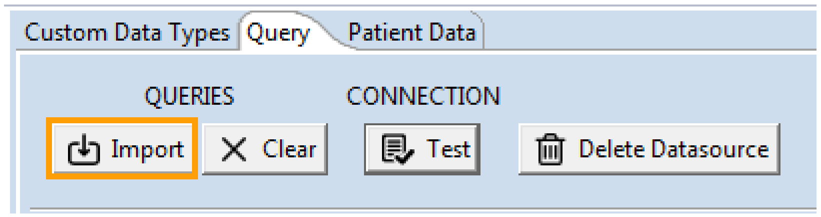
Username: sa

Password: *********

Catalog Filter:

Schema Filter:

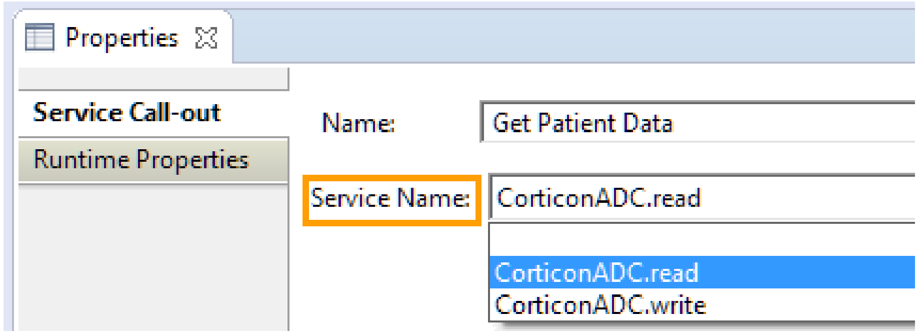
... where you can **Import** the Queries from the Datasource into the project, as shown:



Improved Usability of ADC Queries

In version 5.7.0, you had to type in the name of ADC queries. Now, you can choose queries from a drop down list.

In a Ruleflow that has ADC service callouts, chose one on the canvas, then, on the object's **Properties** tab, choose its **Service Call-out** sidetab, click the **Service Name** pulldown to select the service you want:

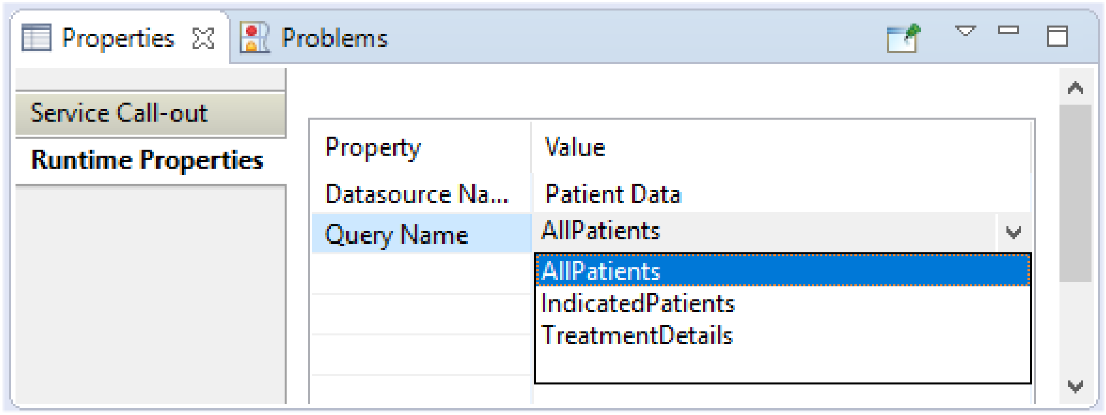


Name: Get Patient Data

Service Name: CorticonADC.read

CorticonADC.read
CorticonADC.write

Choose the **Runtime Properties** tab. Click on the **Datasource Name** property to enable its pulldown that manages all the stored queries of the type you selected.



Property	Value
Datasource Na...	Patient Data
Query Name	AllPatients

Select **Query Name**, and then, for its value, use its dropdown menu to select the appropriate query from the selected Query Datasource.

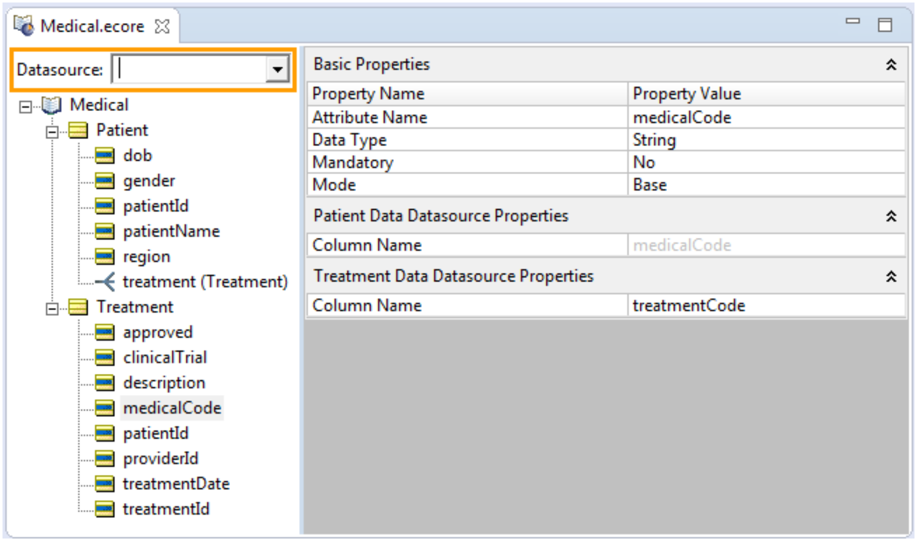
When you revise queries in the Query Datasource, be sure to clear and re-import the queries through their connection.

Improved usability of EDC and ADC Datasources in the Vocabulary

The Vocabulary editor has been enhanced to decorate the vocabulary tree to indicate which elements of the vocabulary are mapped to an ADC Datasource. This decoration has been available for EDC Datasources in previous releases.

When an EDC or ADC Datasource is added to a vocabulary, the Vocabulary editor is modified to place a **Datasource** pulldown menu above the Vocabulary tree, as illustrated:

Figure 1: Datasource not selected



Basic Properties

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

Patient Data Datasource Properties

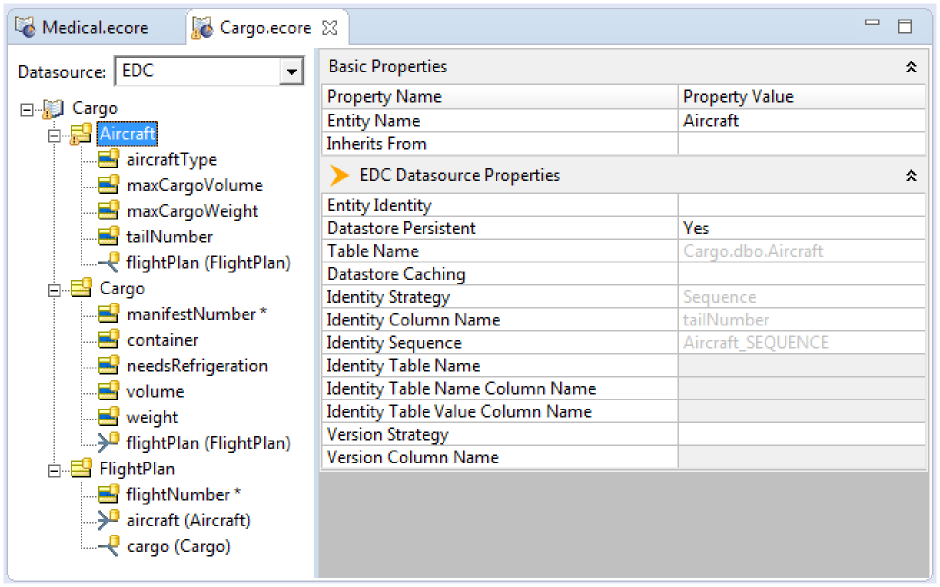
Column Name	medicalCode

Treatment Data Datasource Properties

Column Name	treatmentCode

If an EDC Datasource is selected, the tree icons take on database 'decorations' on each persisted entity and attribute. The list of attributes in each persisted entity is re-arranged such that the one or more attributes that comprise the entity identity, the Primary Key, are at the top of each list, as illustrated:

Figure 2: EDC Datasource selected



Datasource: EDC

```
Cargo
  Aircraft
    aircraftType
    maxCargoVolume
    maxCargoWeight
    tailNumber
    flightPlan (FlightPlan)
  Cargo
    manifestNumber *
    container
    needsRefrigeration
    volume
    weight
    flightPlan (FlightPlan)
  FlightPlan
    flightNumber *
    aircraft (Aircraft)
    cargo (Cargo)
```

Basic Properties

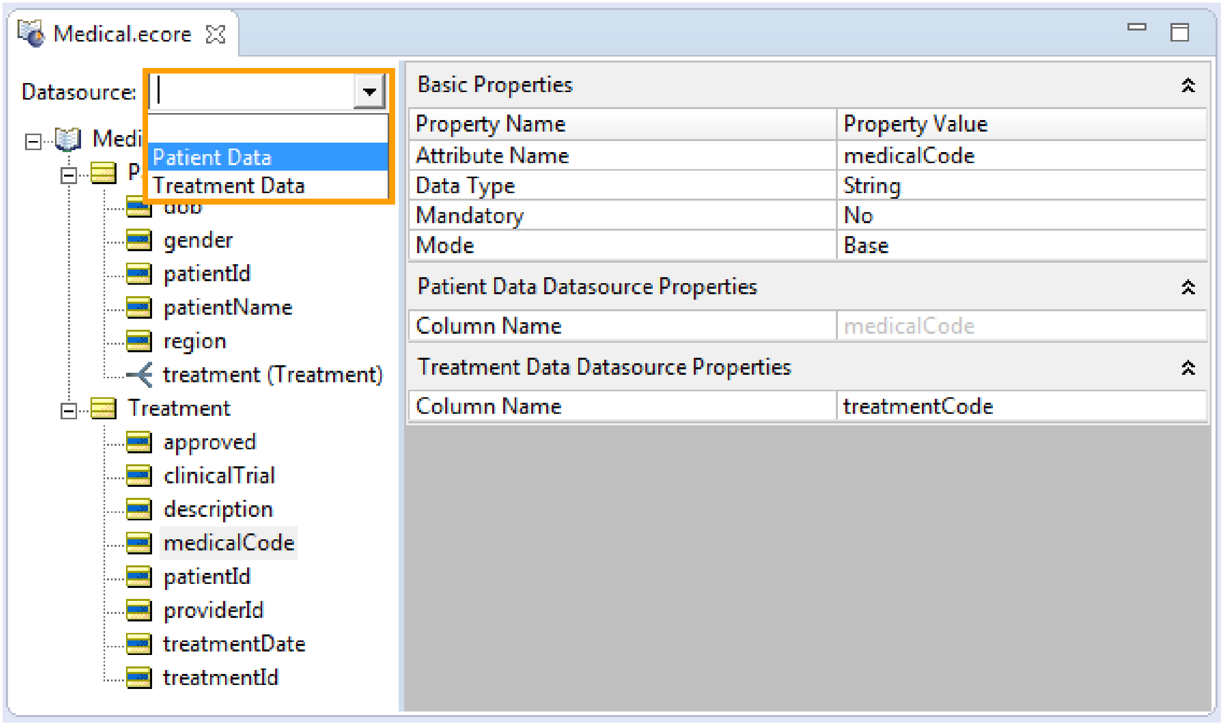
Property Name	Property Value
Entity Name	Aircraft
Inherits From	

EDC Datasource Properties

Entity Identity	
Datastore Persistent	Yes
Table Name	Cargo.dbo.Aircraft
Datastore Caching	
Identity Strategy	Sequence
Identity Column Name	tailNumber
Identity Sequence	Aircraft_SEQUENCE
Identity Table Name	
Identity Table Name Column Name	
Identity Table Value Column Name	
Version Strategy	
Version Column Name	

When more than one Datasource has been defined in a Vocabulary, clicking the pulldown lets you choose which Datasource you want to view, as illustrated:

Figure 3: Selecting from multiple Datasources



Basic Properties

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

Patient Data Datasource Properties

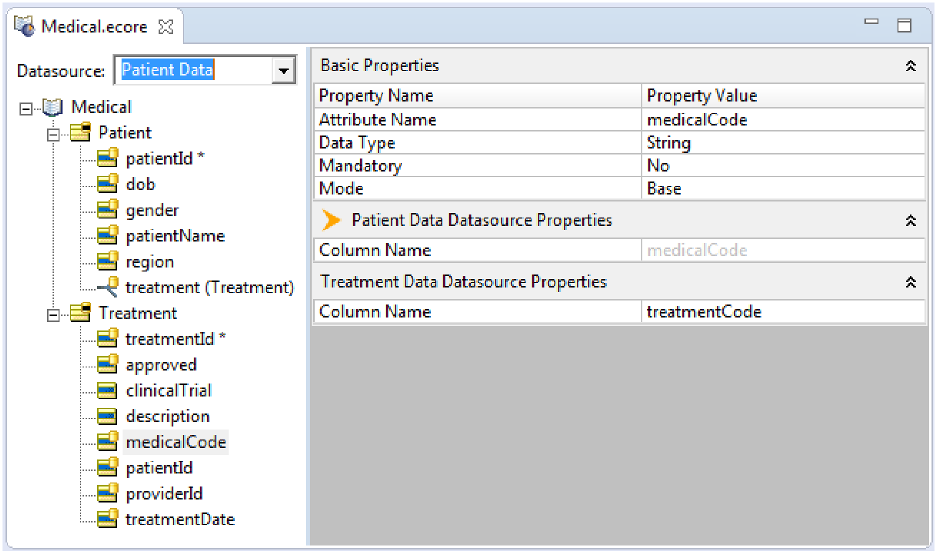
Column Name	medicalCode

Treatment Data Datasource Properties

Column Name	treatmentCode

In the following illustration, the Patient ADC Datasource is showing its persisted elements and keys, and the Datasource's section of the Properties panel is decorated to indicate that it is the Datasource in the current view:

Figure 4: Patient Datasource keys and persistent elements



Basic Properties

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

Patient Data Datasource Properties

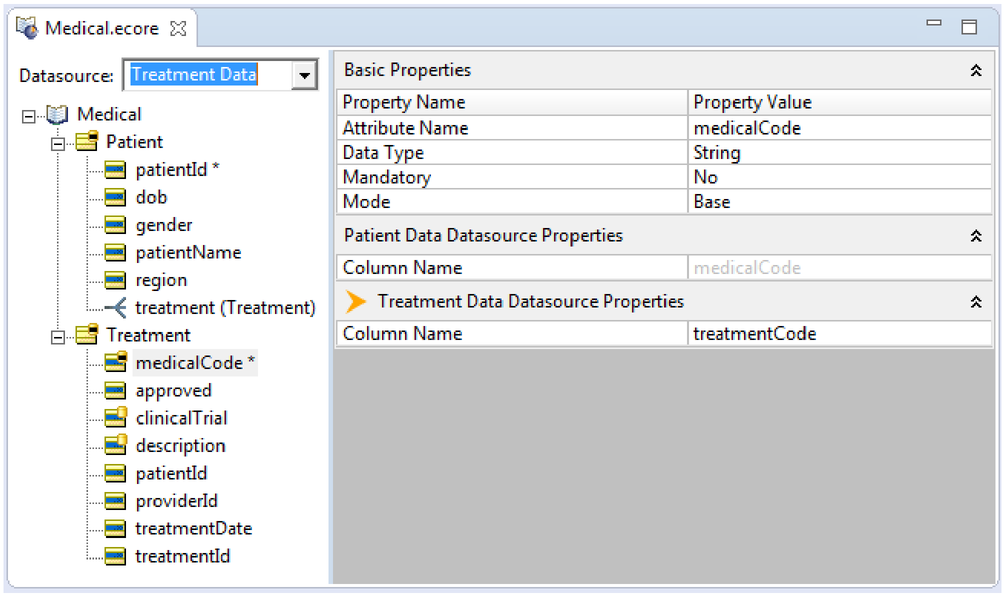
Column Name	medicalCode

Treatment Data Datasource Properties

Column Name	treatmentCode

When you toggle the Datasource selector to the Treatment ADC Datasource, it decorates its persisted elements and keys, and the Datasource's section of the Properties panel is decorated to indicate that it is the Datasource in the current view:

Figure 5: Treatment Datasource keys and persistent elements



Basic Properties

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

Patient Data Datasource Properties

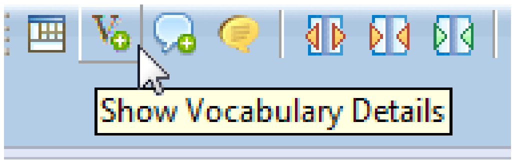
Column Name	medicalCode

Treatment Data Datasource Properties

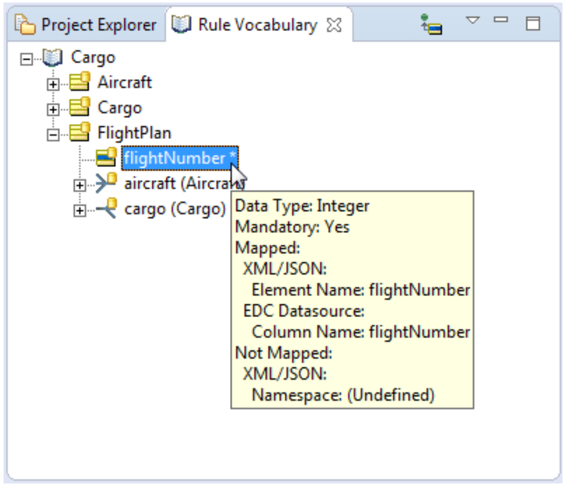
Column Name	treatmentCode

Improved hover help in Vocabulary to display multiple Datasources

When Vocabulary details are selected, hover help in the Vocabulary has been enhanced to clarify what is mapped and what is not mapped:



The following illustration shows the Rule Vocabulary showing the details of the XML/JSON mapping option and the EDC Datasource property mapping:



```
Data Type: Integer
Mandatory: Yes
Mapped:
 XML/JSON:
  Element Name: flightNumber
 EDC Datasource:
  Column Name: flightNumber
Not Mapped:
 XML/JSON:
  Namespace: (Undefined)
```

This information was added in the topic *"How Datasource information is viewed in the Vocabulary" in the Data Integration Guide.*

Set client type for connections to IIS servers

Corticon's SOAP stack requires certain client behavior when passing payloads to an IIS server. Changing the setting in the `brms.properties` file enables the appropriate client type for IIS.

```
com.corticon.studio.client.soap.clienttype=IIS
```

- Value of `JAVA` uses the SOAP client for a Corticon server running inside a Java container.
- Value of `IIS` uses the SOAP client for a Corticon server running on IIS

When this value is not set, the value `JAVA` is assumed.

This setting is required in IIS projects and deployments for the following functions:

- The clients `testServerAxis.bat` (Java) and `Corticon-Api-Remote-Test.exe` (.NET) require this property set to IIS.
- Ruletest Test Subjects use this property set to IIS to enable running the test against an IIS server. Avoid having some Testsheets within a Ruletest connect to IIS while others are Java as running the full Ruletest will fail.
- Packaging and deploying a project to an IIS server requires the client type set to IIS .

To revert to using these functions againist a Java server, reset the property value to JAVA or clear the property in `brms.properties`, and then restart Corticon.

This information was added as the topic *"Setting the Client Type" in the guide to Deploying Web Services on .NET.*

Settable properties described in context

Setting properties that change behaviors in the Corticon products are put into effect as line items in the `brms.properties` file that is installed at the `[WORK_DIR]` root. For information about how that file is handled, see *"Configuring Corticon properties and settings" in the Integration and Deployment Guide.*

The documentation describes the properties you can set in:

Specific locations inline:

- *"Dynamic discovery of new or changed Decision Services" in the Integration and Deployment Guide.*
- *"Turning off server state persistence" in the Integration and Deployment Guide.*
- *"Terminating infinite loops" in the Rule Modeling Guide.*
- *"Single machine configuration (Wrappers and Pools; Maximum Pool Sizes)" in the Integration and Deployment Guide.*
- *"Format of output data" in the Quick Reference Guide.*
- *"Setting Server startup to auto load CDD files" in the Integration and Deployment Guide's Inside Corticon Server section.*
- *"Relaxing enforcement of custom data types" in the Rule Modeling Guide*. Also see *"Running a Ruletest in Corticon Studio" in the Rule Modeling Guide.*

- *"Specifying server URLs to access test subjects" in the Quick Reference Guide.* Also see *"Troubleshooting Java server" in the Java Deployment Guide* and *"Configuring Studio to send a SOAP message to IIS" in the .NET Deployment Guide.*
- *"The Corticon Server Sandbox" in the Java Deployment Guide* and *"The .NET Server Sandbox" in the .NET Deployment Guide.*
- *"Tracing Rule Execution" in the Rule Modeling Guide.*
- *"XML/JSON mapping" in the Integration and Deployment Guide.*
- *"Using the Server Deployment Console " in the Integration and Deployment Guide's Packaging and deploying Decision Services section.*

Several settings in the context of a feature's topic:

- *"Implementing Rule Execution Recording in a database" in the Integration and Deployment Guide.*
- *"Setting Service Contract properties" in the Integration and Deployment Guide.* and *"Extended service contracts" in the Integration and Deployment Guide.*
- *"Diagnosing runtime performance of server and Decision Services " in the Integration and Deployment Guide.*
- *"Configuring log files" in the Integration and Deployment Guide.*
- *"Setting properties in a CDD file" in the Integration and Deployment Guide.*

General settings that are listed as appropriately placed topics in the documentation:

- *"Setting Studio properties" in the Rule Modeling Guide.*
- *"Formatting Date, Time, and DateTime properties" in the Rule Language Guide.*
- *"Setting Server execution properties" in the Integration and Deployment Guide's Performance Tuning section.*
- *"Setting Server build properties" in the Integration and Deployment Guide's Performance Tuning section.*
- *"Setting Web Console server properties" in the Integration and Deployment Guide.*
- *"Setting Server properties that are baked into Decision Services" in the Integration and Deployment Guide.*

Enhancements to Learning Materials

The following changes to learning materials are worthy of note:

- **Using Native Corticon Extensions in .NET** - When creating extensions for .NET use cases, you might prefer to program a DLL in Microsoft Visual Studio to use exclusively on .NET servers. *"Extensions in .NET execution environments" in the Deploying Web Services with .NET Guide.*
- **Improved REST calls** - A JSON request message has a body that describes the parameters for handling the request payload, and the payload.

```
{
  "name": "string",
  "majorVersion": "string",
  "minorVersion": "string",
  "effectiveTimestamp": "string",
  "Objects": [
     {}
  ]
}
```

See the following topic and related topics for more information: *"About creating a JSON request message for a Decision Service" in the Integration and Deployment Guide.*

- **Checklist of steps in an upgrade** - Installing a software upgrade calls for a review of related files and assets in your developmrnt and deployment environments. See *"Procedures for upgrading a Corticon installation" in the Installation Guide.*
- **Undocumented Timezone formats** - While not previously listed in the documentation, Corticon's support of Java time formats does include the general timezone symbol (`z`) as well as the RFC 822 time zone (`Z`), and ISO 8601 (`X`) formats. See *"Formatting Date, Time, and DateTime properties" in the Rule Language Guide.*

  Note: See Oracle's SimpleTimeFormat Javadocs for more detailed information.

- **.NET Servers use brms.properties** - Placing a `brms.properties` file adjacent to the `axis.war` file on a .NET server enables easy changes to properties.
- **Tutorials updated to this release** - All the Corticon online tutorials have been revised to bring them up to date and to reflect the very latest user interfaces and functionality. Watch for additional new tutorials in the coming months !

4

What was new and changed in Corticon 5.7

This section summarizes the new, enhanced, and changed features in Progress® Corticon® 5.7.0.

For details, see the following topics:

Enhanced access to external databases

A Decision Service can now use multiple instances of Corticon's Advanced Data Connector (ADC) to access data in multiple databases. An example would be a Decision Service that reads data from two database and writes to a third. The Corticon Vocabulary editor has been enhanced to allow you to map entities and attributes to more than one database.

See the *Corticon Server: Data Integration Guide* for comprehensive information about access to external databases.

Batch rule processing

Corticon Server now makes it easy to perform batch rule processing of database data. This is for rule processing of large data sets such as healthcare records or financial transactions either on-demand or on a recurring schedule, such as nightly or quarterly.

Corticon Server now accepts requests to start a batch process. It will efficiently retrieve the specified data from a database and pass it to the indicated Decision Service for processing. Batch rule processing is designed for use with Decision Services that use ADC. You can start and manage batch processes through the Corticon Web Console, or use the Corticon Server REST API to integrate batch processing with other scheduling tools you use.

To learn more about batch processing in Corticon, see the Corticon *Data Integration Guide* section *"Getting Started with Batch".*

Datasource Configuration File

A new file has been added for the configuration of Datasources used by a Decision Service, the Datasource Configuration File. This XML file can be created manually or exported from Corticon Studio when editing a Vocabulary. When deploying a Decision Service that accesses external Datasources, you provide the EDS file for the Decision Service and its Datasource Configuration File. By editing the configuration file when you deploy a Decision Service, you can adjust the connections and their credentials so that they use test or production databases.

The Datasource Configuration File replaces the `database.properties` file previously used with EDC Decision Services. While you can still deploy using the `database.properties` file, you are encouraged to move to the using Datasource Configuration Files.

To learn more about Datasource Configuration Files, see the Corticon *Data Integration Guide* section *"Deploying projects that use data integration".*

Ability to richly document rule assets

Commenting files as you design and develop them is considered a professional best practice. It provides essential information to current and future stakeholders of projects, including colleagues that will maintain and enhance them to meet the inevitable changing business requirements of a project.

You can now add comments to all rule assets and the discrete components within them. Example include comments on entities and attributes in a Vocabulary and individual rules and cells in a Rulesheet. A new **Comments** View provides a flexible way to view the comments in a rule asset.

For more information about this feature, see the section *"Documenting Rule Assets" in the Quick Reference Guide.*

Ability to refactor element names in a Vocabulary

Corticon now lets you *refactor* a Vocabulary 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.

Choose the **Refactor** option on the right-click context menu of any entity, attribute, or association name to initiate the process.

For more information about this feature, see *"Refactoring entity, attribute, or association names in a project" in the Quick Reference Guide.*

Usability improvements

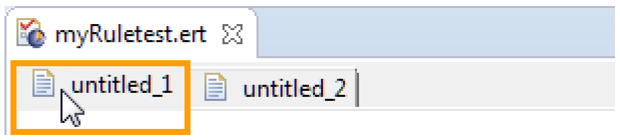
The behavior of several Corticon Studio features has changed to improve their usability. Significant usability improvements in this release include:

Numerical equality applied during validation

When comparing expected results with output results during the validation stage of testing, two values that have a different number of trailing zeros to the right of the decimal place now validate correctly. **Note:** Avoid introducing rounding errors and inconsistent use of big decimal data types. These can still lead to differences during comparisons. This information was added to *Techniques that refine rule testing* in the *Rule Modeling Guide*

Double-click to rename a Testsheet

In the Ruletest editor, you can change the name of a Testsheet by double-clicking on its tab, as illustrated:



The **Rename Testsheet** dialog opens where you enter your preferred name for the Testsheet.

This improvement updates the topic *"Renaming Testsheets" in the Quick Reference Guide.*

Collapse All and Expand All items in Testsheet trees

Right-click in a Ruletest view to select **Expand All** or **Collapse All** 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.

Exclude Transients when exporting Testsheets to XML, SOAP, or JSON

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

Ignore Validation applies to all instances of a node in a view

In Testsheets, when you 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.

File Selection dialogs filter only files from the parent project

To prohibit you from creating cross-project assets and the inevitable confusion associated with them, file selection dialogs filter and display only files from the parent project. For example, the **Select Vocabulary** dialog only lists the Vocabularies for the parent project.

Add Vocabulary attributes by data type

When you use the **Add Attribute >** option to add an attribute to an entity from the vocabulary tree, a list of data types now displays. Select the appropriate data type for the new attribute, and the properties for the new attribute are set to that data type. This speeds the process of building a vocabulary by eliminating the separate step of setting the data type of attributes after being added.

See *the Vocabularies topic "Attribute nodes: Adding adding and editing attributes and their properties" in the Quick Reference Guide* for additional information.

Publish selected Ruleflows from Project Explorer

You can now select specific Ruleflows for packaging and deployment from the Project Explorer view, and then publish only those Ruleflows. In projects with large number of Ruleflows, you can place your main Ruleflows in a separate folder to make it easy to locate them and select them for deployment.

See *"Publishing Ruleflows from Project Explorer" in the Quick Reference Guide* for additional information.

Access to HTTP Headers in Extended Operators and Service Callouts

You can now create extensions (extended operators or service callouts) that access the HTTP headers included on the call to execute a Decision Service deployed as a SOAP or REST web service. One use of this mechanism is to pass a user security token from the client making the request to the Decision Service where an extension can use the token to access an external data source as the client. This is useful when Corticon is behind a SaaS or other multi-tenant service and a Decision Service needs to access external data as a specific tenant.

Corticon only makes the HTTP headers available--their interpretation is dependent on the extension that uses them.

For more information, see the topic *"Accessing HTTP Headers in Extended Operators and Service Callouts" in the Corticon Extensions Guide.*

Bundled plugins that support management of project assets

Corticon Studio bundles plugins for two popular software version control systems in use by many developers today, Apache Subversion (SVN) and Git. Corticon Studio bundles Eclipse plugins for Subversive SVN version 4.0.5. and EGit version 4.0. If you are using Subversion or Git to manage your rule assets, you can now do this directly from Corticon Studio without the need to install additional plugins..

For more information about this feature, see the section *"Using SVN or Git to manage project assets" in the Quick Reference Guide.*

Other changes

The following changes are also noteworthy:

- **Corticon Studio now supports Java 1.8 and Eclipse 4.5**
- **Studio can run in different languages** You can run Corticon Studio in French, Japanese, Brazilian Portuguese, or Spanish. Corticon Studio now bundles the Eclipse language packs for these languages. See *"Setting Studio to run in another language" in the Installation Guide* for more information.
- **Changes to OpenEdge and Corticon Eclipse Environment** - Corticon Studio no longer supports installation directly into Progress Developers Studio for OpenEdge. To jointly develop business applications with OpenEdge and Corticon, use separate Eclipse instances for Progress Developers Studio and Corticon Studio.
- **Ruletests can remove invalid nodes** - The Ruletest menu provides the command **Remove Invalid Nodes** that discards all invalid nodes in Input, Output, and Expected columns in all testsheets in the Ruletest.
- **Changes to Documentation Delivery** - Several changes have been made to how documentation is presented and how it is accessed:
  - A new guide, the *Data Integration Guide*, gathers the information on EDC and ADC from other guides to expand on the ways you can use Corticon features to read and write from remote databases.
  - Corticon Studio no longer includes all documentation content in its installed help. Instead, a single page provides online access to all the 5.7 documentation components in PDF and HTML formats.

Deprecated Features

With the release of Corticon 5.7, we are announcing the deprecation of several features. These features will be removed in a future release of Corticon. If you use any of these features, you are encouraged to begin preparation to use the identified alternatives.

- **CDD deployment of ERF files** - Deployment of uncompiled ERF files to an application server places the requirement on the application server to compile the Ruleflow and associated Rulesheets into a deployable Decision Service. This places a significant load on the application server during compilation and is not supported on all application servers. The best practice is to deploy precompiled EDS files. This eliminates the need for the server to perform compilation and better fits into modern Continuous Integration and Continuous Deployment practices. Recent Corticon features – such as the ability to export EDS files from

Corticon Studio, and to perform compilation in build scripts such as ant -- provide flexible options for producing EDS files, eliminating the need for ERF deployment.

- **Download of Decision Services** - Corticon Studio retains a legacy capability to download a Decision Service from a Corticon Server. If the rule assets used to create the Decision Service were included in the deployed EDS for the Decision Service, this feature could be used to access the rule assets. The best practice for managing your rule assets is to store them in a source code control system such as SVN or Git. Corticon Studio now includes plugins for using both SVN and Git with your rule projects.
- **SOAP Management API** - Corticon Server has both SOAP and REST APIs for managing the Corticon Server. Given the prevalence of REST, no new additions were made to the SOAP APIs in Corticon 5.7. If you use the SOAP API to deploy or manage Decision Services, you will need to migrate to using the equivalent REST APIs. Note that the ability to execute a Decision Service with a SOAP request will not be removed -- only the management APIs are affected. Corticon Server's REST APIs are accessible via Swagger on your deployed Corticon Server at `http://localhost:8850/axis/swagger` (use your appropriate host name and port number).
- **Deployment Console** - The Corticon Deployment Console bundled with Corticon Server provides a GUI mechanism for creating CDD files, compiling Decision Services, and generating WSDL files for Decision Services. These tasks can each be performed in other ways. The CDD file format is a simple text file manifest describing a Decision Service. The most common practice is to copy a sample CDD file bundled with Corticon, and then make modifications to it in a text editor. The compiling of Decision Services can be achieved in a few ways -- export from Corticon Studio, bundled ant macros for scripted builds, and Corticon Server APIs. WSDLs are now generated and bundled with Decision Services on compilation. The WSDL for a Decision Service can be accessed via the Corticon Web Console or the Corticon Server REST API.