PROGRESS EXCHANGE 2013

DISCOVER. DEVELOP. DELIVER.

Corticon Business Rules for OpenEdge

PROGRESS

Contents

Introduction

This document outlines the exercises that you will complete as part of the workshop. It guides you step by step through building a simple Corticon decision service and integrating this with an OpenEdge application. No previous knowledge of Corticon is required. However, a basic understanding of OpenEdge ABL, Temp-Tables and DataSets is advisable.

A basic OpenEdge application is provided as a starting point. The end product of the workshop will be decision logic deployed and running in the Corticon Server and integrated with the OpenEdge application.

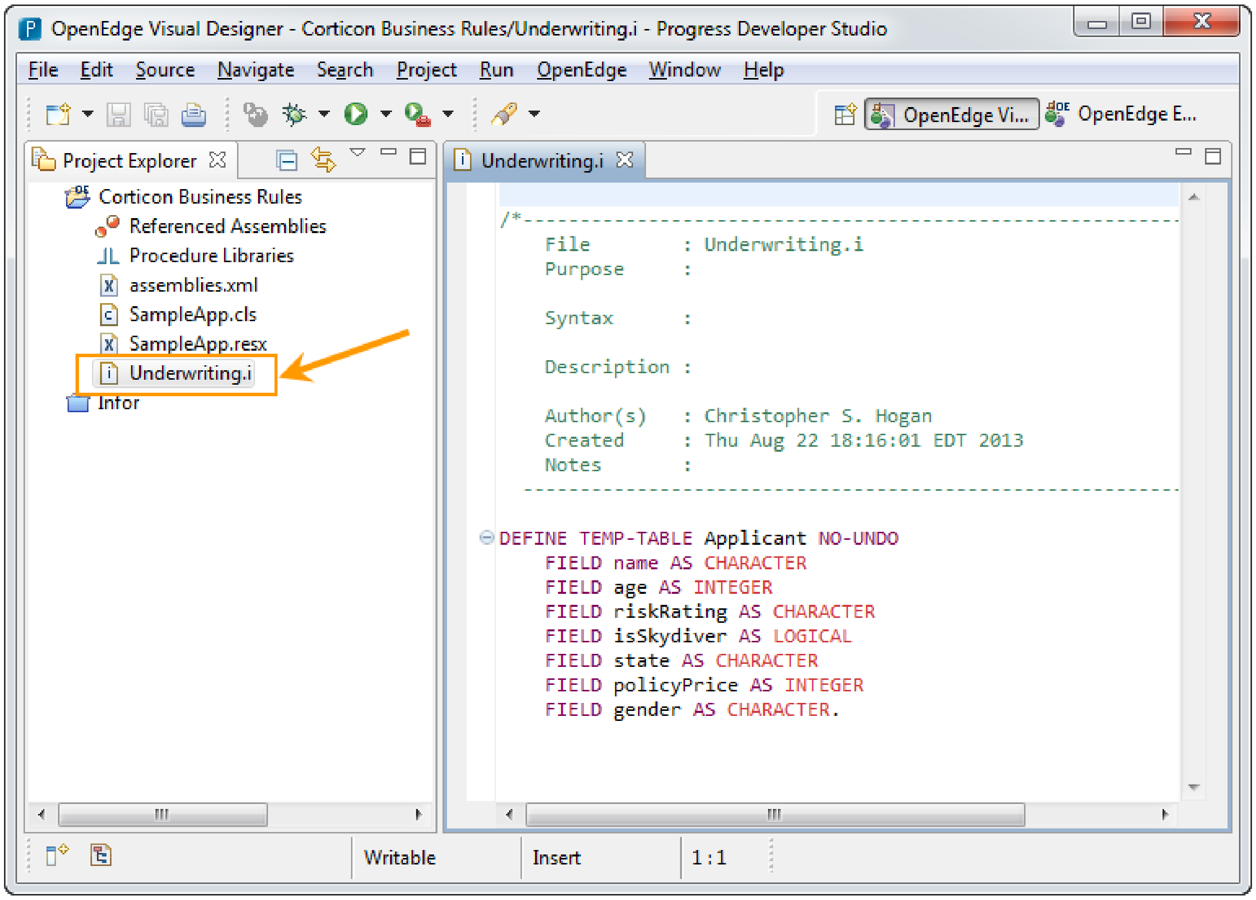
Exercise 1 – Model Corticon Decision

A Corticon decision model is comprised of a Vocabulary (data model), and sets of Rules against that model. The Vocabulary can be built from scratch inside Corticon Studio, but it can also be generated from existing artifacts. OpenEdge 11.3 provides the ability to create a Corticon Vocabulary from existing data structures like Temp-Tables and Pro DataSets. In this exercise we will create the Vocabulary from a Temp-Table within the provided application. Once we have the Vocabulary, we will model the business rules and use the analysis tools in Corticon Studio to ensure accuracy and completeness. If there are any defects, we will resolve them. Finally we will use test data to execute the rules, and see the outcome of a given decision.

When we are satisfied with the decision model, we will use the publish wizard to deploy the service to the Corticon Server. This will make the decision available as a callable service from any application via SOAP, as well as through the built in Corticon integration with OpenEdge 11.3.

Step 1 – Create Vocabulary

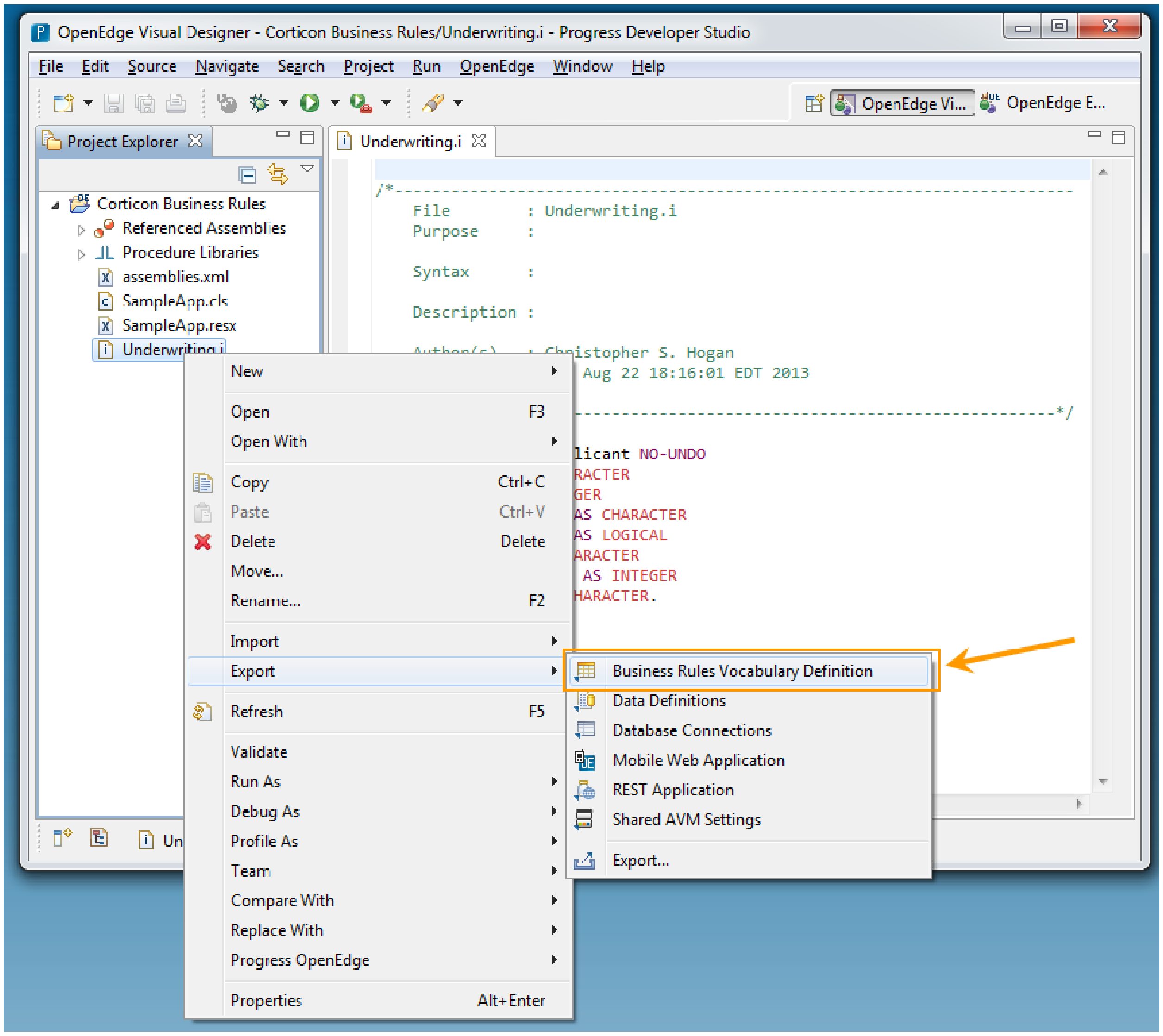
Launch Progress Developer Studio for OpenEdge from the desktop. If you are prompted to select a Workspace, you should use the following location **C:\Progress\Corticon_5.3\OE_workspace.** You will see the "Corticon Business Rules" project opened. Double click **Underwriting.i** to reveal the contents of a pre-defined temp-table (Figure 1)



```
/*------------------------------------------------------------------------
    File        : Underwriting.i
    Purpose     :

    Syntax      :

    Description :

    Author(s)   : Christopher S. Hogan
    Created     : Thu Aug 22 18:16:01 EDT 2013
    Notes       :
  ----------------------------------------------------------------------*/

DEFINE TEMP-TABLE Applicant NO-UNDO
    FIELD name AS CHARACTER
    FIELD age AS INTEGER
    FIELD riskRating AS CHARACTER
    FIELD isSkydiver AS LOGICAL
    FIELD state AS CHARACTER
    FIELD policyPrice AS INTEGER
    FIELD gender AS CHARACTER.
```

Figure 1

Right mouse click on Underwriting.i and select **Export/Business Rules Vocabulary Definition** (Figure 2).

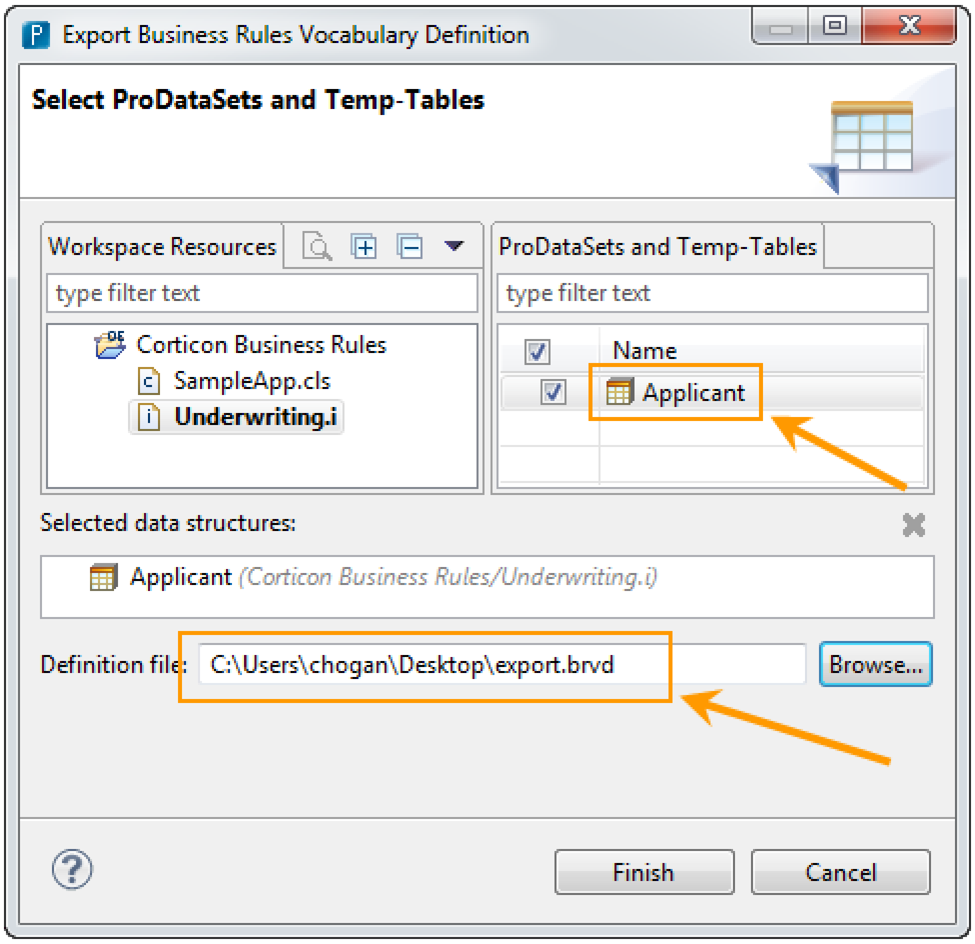


```
/*------------------------------------------------------------------------
    File        : Underwriting.i
    Purpose     :

    Syntax      :

    Description :

    Author(s)   : Christopher S. Hogan
                       Aug 22 18:16:01 EDT 2013
                  ------------------------------------------------*/

                        licant NO-UNDO
                        RACTER
                        GER
                        AS CHARACTER
                        AS LOGICAL
                        ARACTER
                         AS INTEGER
                        HARACTER.
```

Figure 2

This will launch the export wizard where you can specify to temp-tables and datasets you wish to export, as well as the location of the exported file. (Figure 3)

You should choose the **Applicant Temp-Table** for export. You can choose any location and name for the export definition file. Be sure to remember your choices, as you will use this file to import into Corticon Studio in a later step.


Export Business Rules Vocabulary Definition

Select ProDataSets and Temp-Tables

Workspace Resources

type filter text

- Corticon Business Rules
  - SampleApp.cls
  - **Underwriting.i**

ProDataSets and Temp-Tables

type filter text

✓	Name
✓	Applicant

Selected data structures:

Applicant *(Corticon Business Rules/Underwriting.i)*

Definition file: C:\Users\chogan\Desktop\export.brvd

Figure 3

Now Launch Corticon Studio from the desktop. If you are prompted to select a Workspace, you should use the following location **C:\Progress\Corticon_5.3\Corticon_workspace.** You should see the project "Progress Exchange 2013" opened (Figure 4)

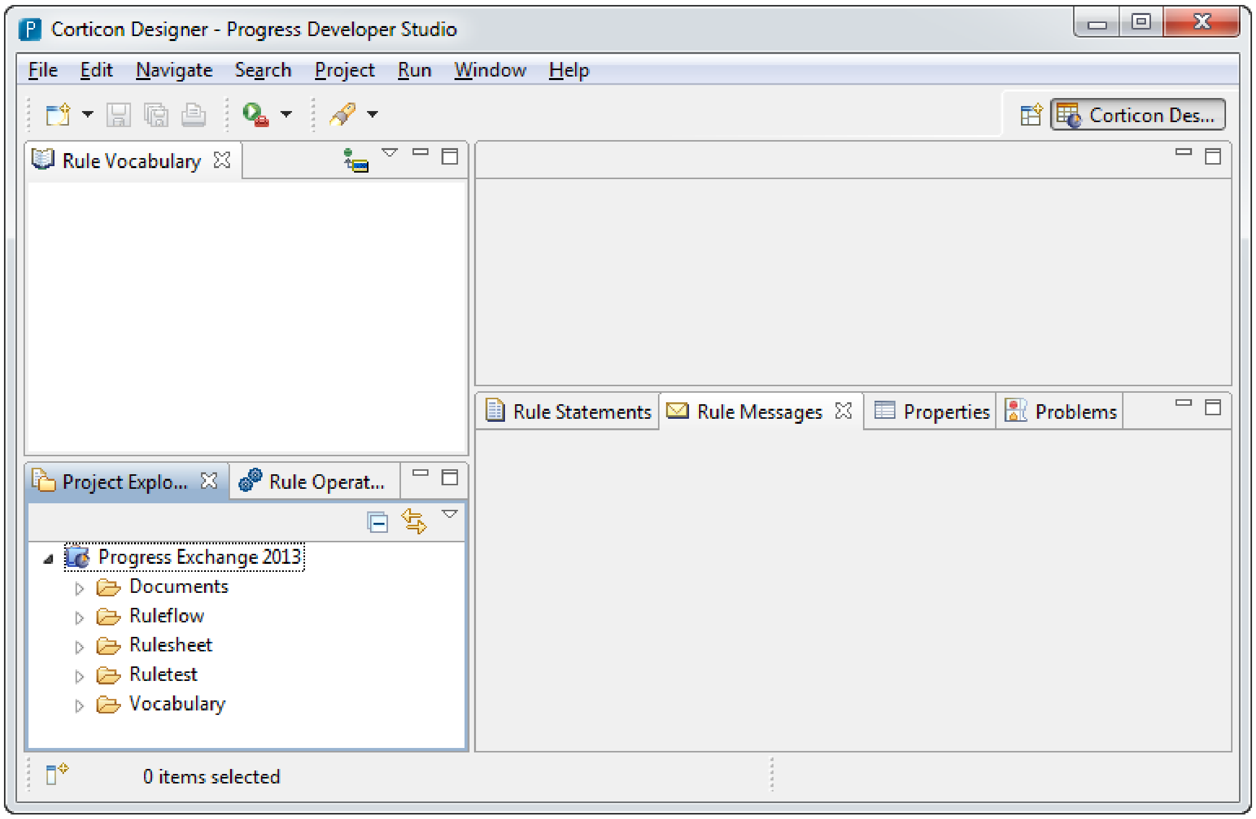


Figure 4

Right mouse click on the project and select Import (Figure 5). This will launch the import wizard. From here you should choose “Business Rule Vocabulary Definition” (Figure 6). Next you should select the file that you exported in the previous steps (Figure 7). And finally you will specify the name and location in the Corticon project for the imported vocabulary file. You should choose the **Vocabulary** folder, and name the file **Underwriting** (Figure 8)**.** You should now see **Underwriting.ecore** in the Project Explorer. (Figure 9).

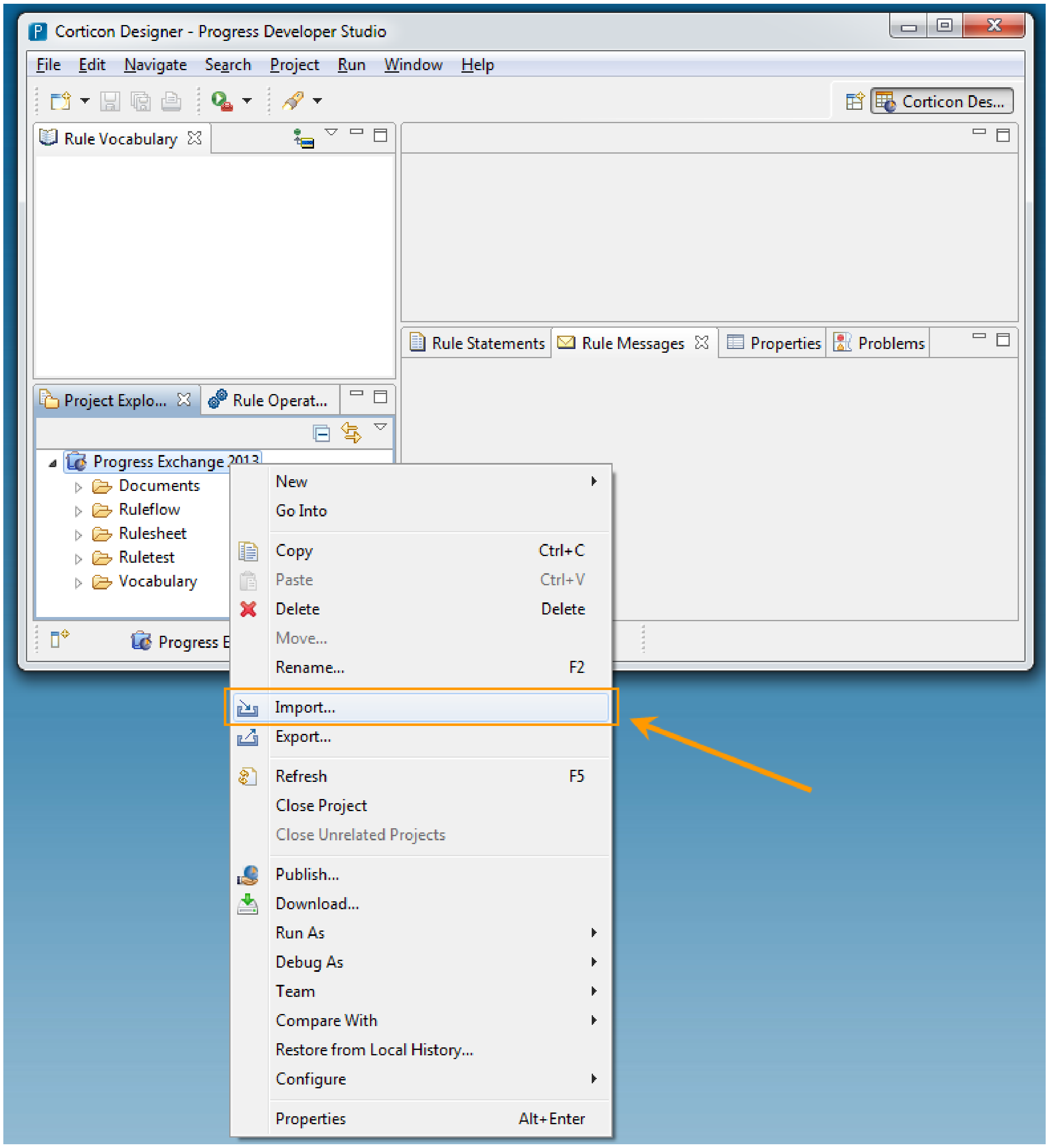

Corticon Designer - Progress Developer Studio
File Edit Navigate Search Project Run Window Help
Corticon Des...
Rule Vocabulary
Rule Statements
Rule Messages
Properties
Problems
Project Explo...
Rule Operat...
Progress Exchange 2013
Documents
Ruleflow
Rulesheet
Ruletest
Vocabulary
Progress E
New
Go Into
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Move...
Rename... F2
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Publish...
Download...
Run As
Debug As
Team
Compare With
Restore from Local History...
Configure
Properties Alt+Enter

Figure 5

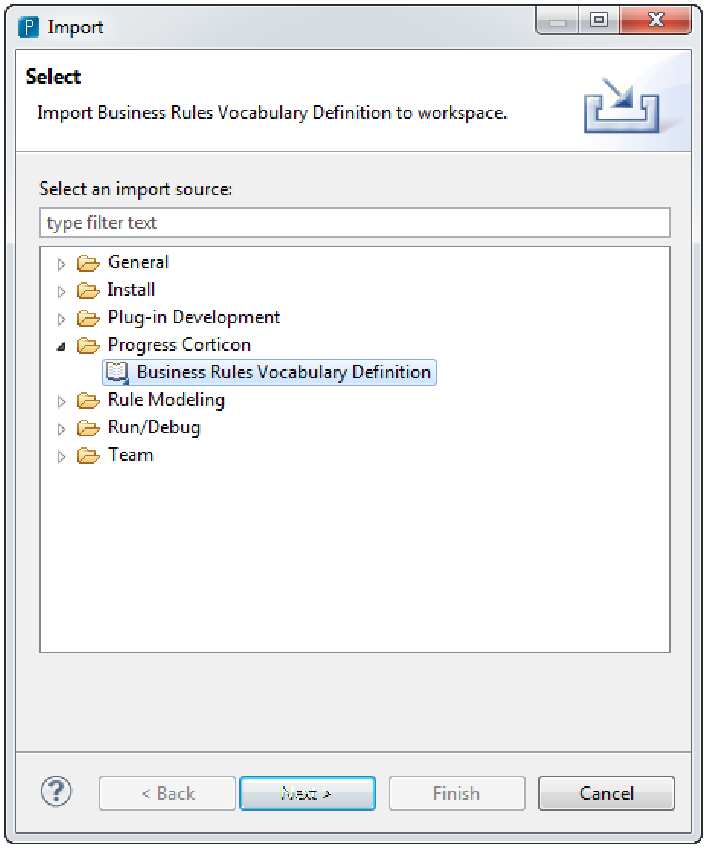


Figure 6

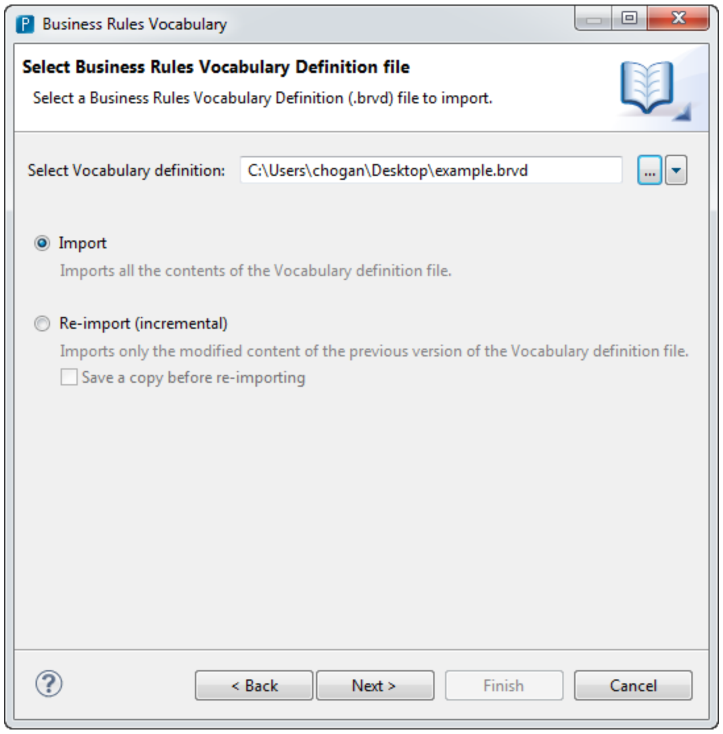
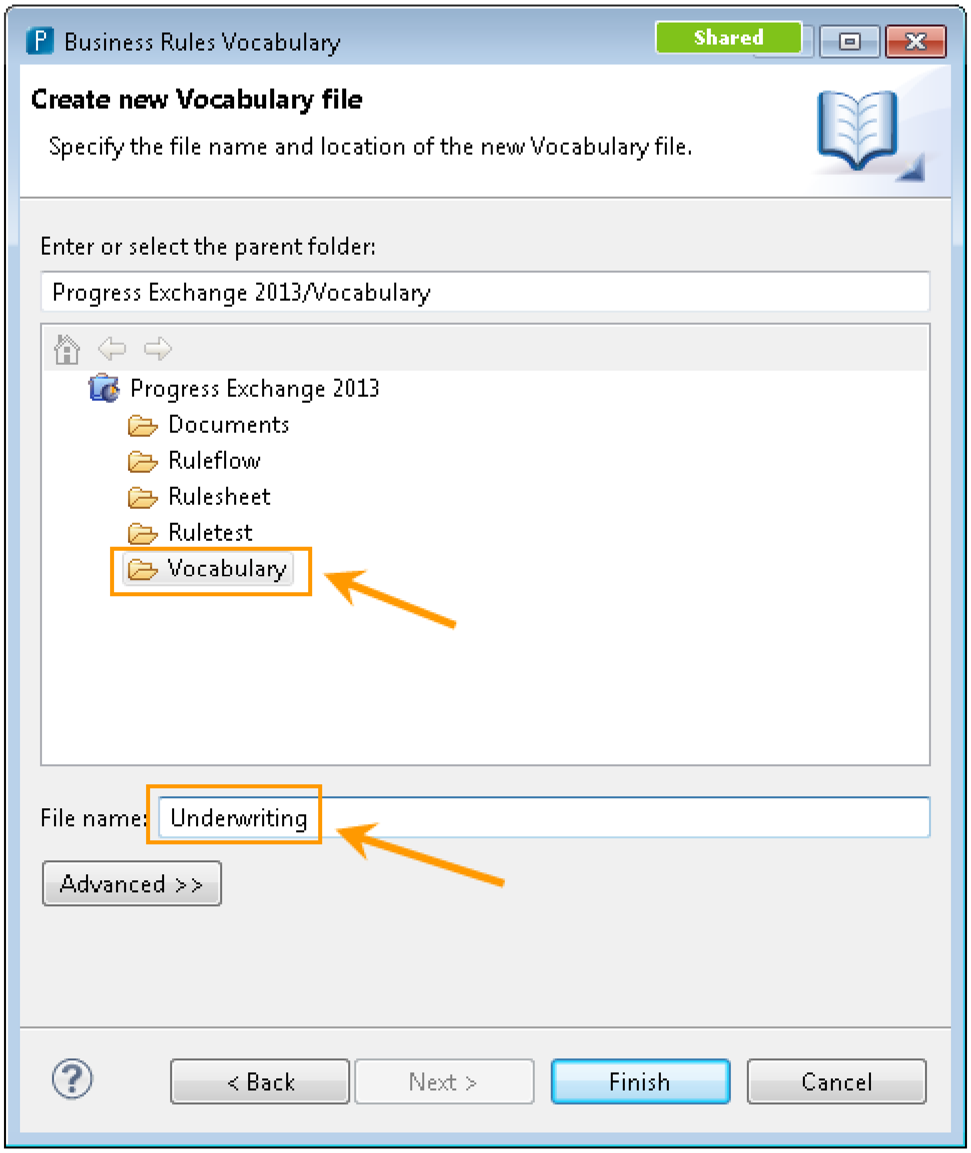

Business Rules Vocabulary
Select Business Rules Vocabulary Definition file
Select a Business Rules Vocabulary Definition (.brvd) file to import.
Select Vocabulary definition: C:\Users\chogan\Desktop\example.brvd
Import
Imports all the contents of the Vocabulary definition file.
Re-import (incremental)
Imports only the modified content of the previous version of the Vocabulary definition file.
Save a copy before re-importing
< Back
Next >
Finish
Cancel

Figure 7



Business Rules Vocabulary

Create new Vocabulary file

Specify the file name and location of the new Vocabulary file.

Enter or select the parent folder:

Progress Exchange 2013/Vocabulary

- Progress Exchange 2013
  - Documents
  - Ruleflow
  - Rulesheet
  - Ruletest
  - Vocabulary

File name: Underwriting

Figure 8

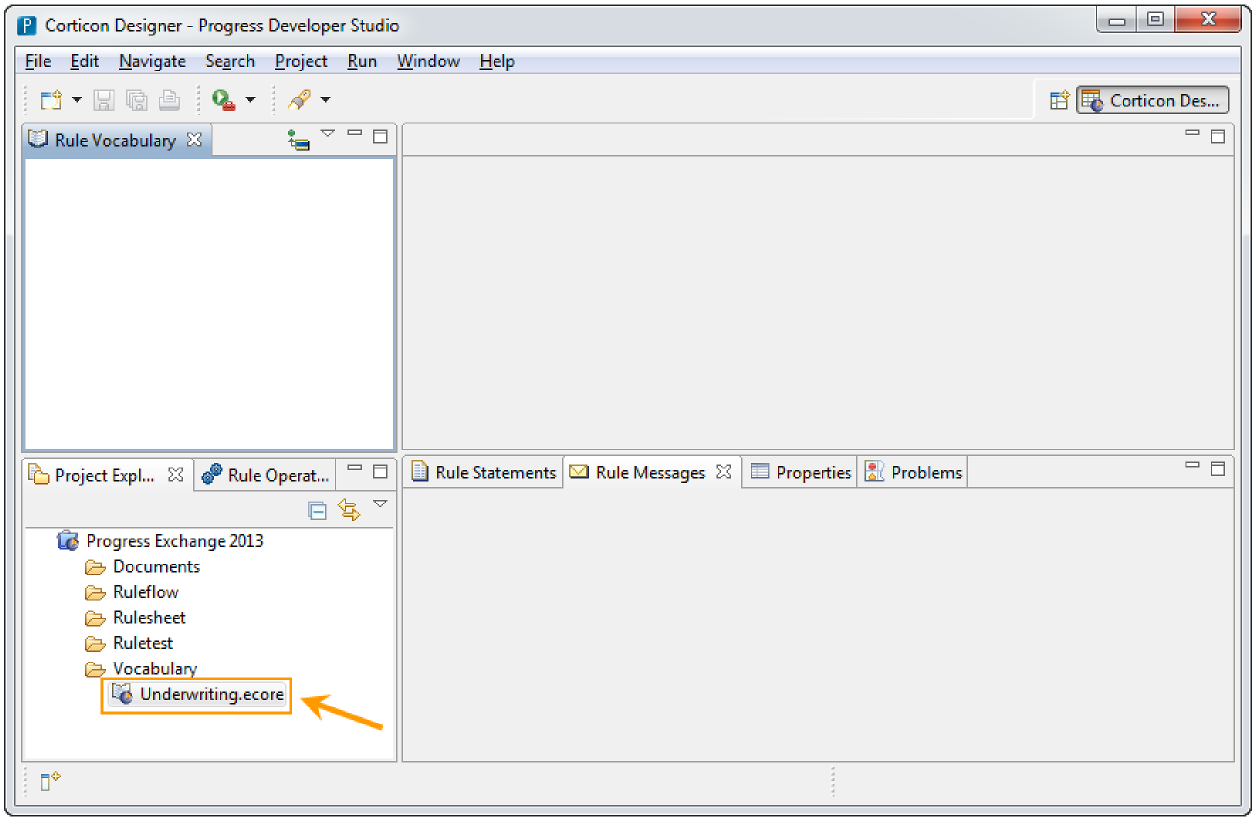


Figure 9

Step 2 – Model Business Rules in Rulesheet Editor

To create a new Rulesheet, right mouse click on the project and select New/Rulesheet (Figure 10). Choose the "Rulesheet" folder and name it **RiskRating** (Figure 11)**.** You are then prompted to specify the Vocabulary you want to use for this Rulesheet. Select the **Underwriting.ecore** file that you previously created (Figure 12). You are now presented with a blank Rulesheet, ready to use for modeling our rules (Figure 13).

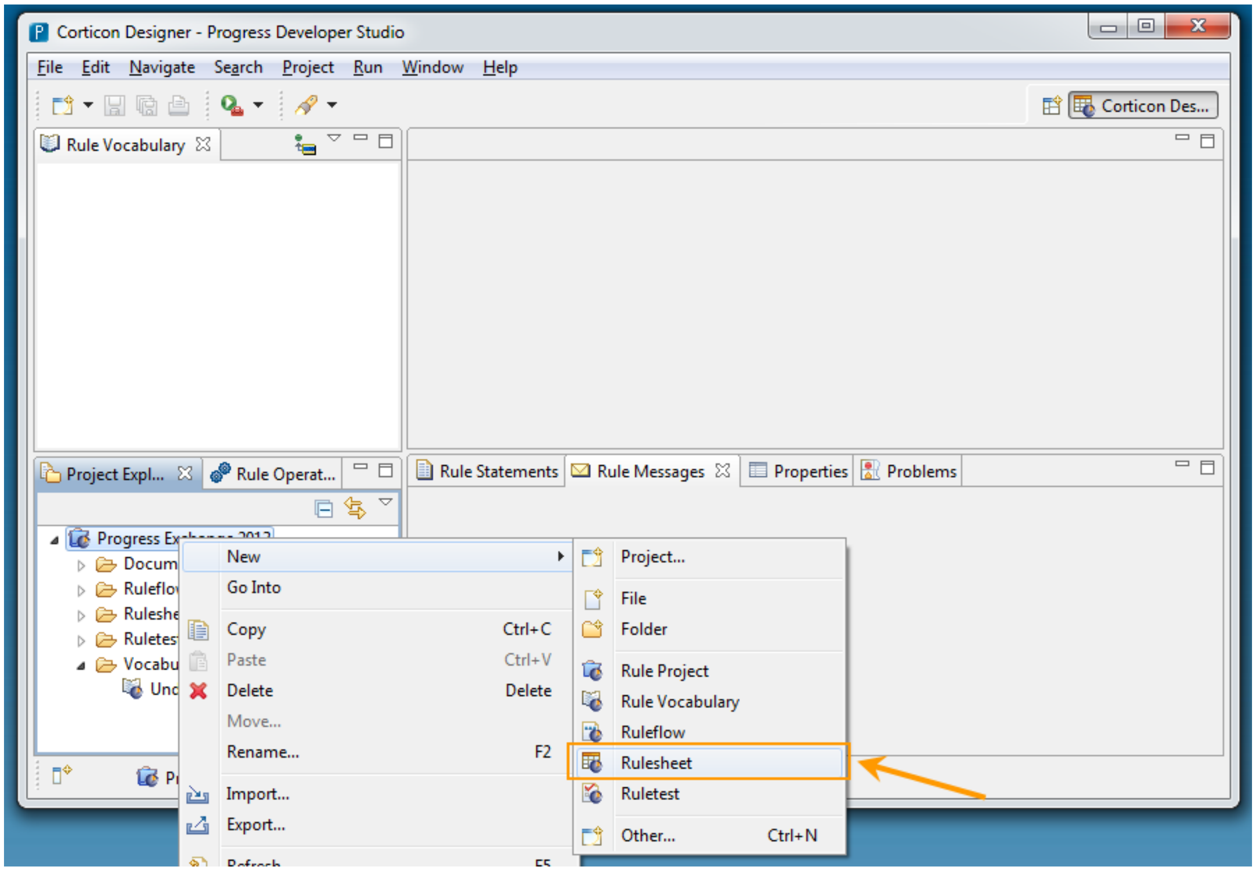
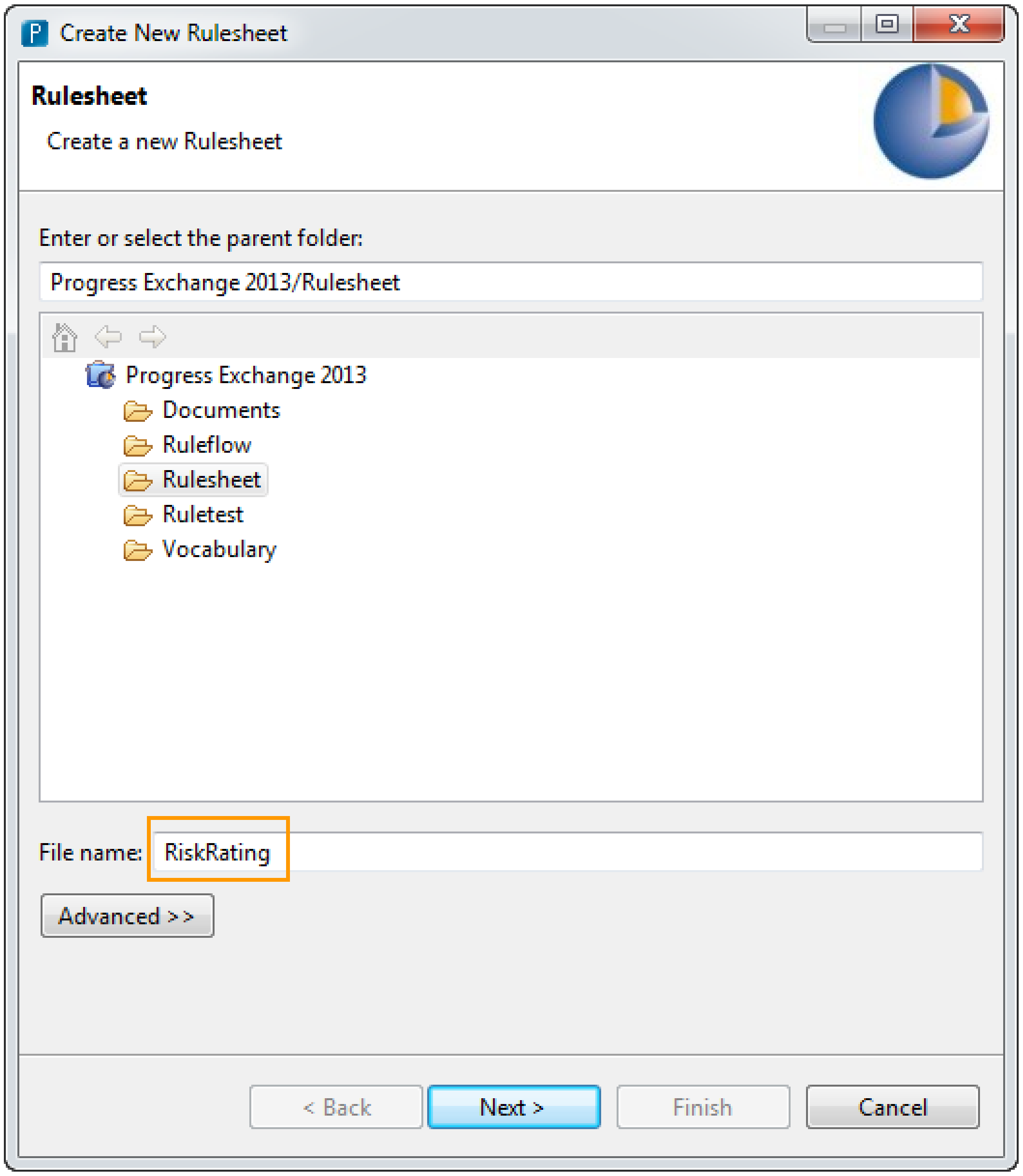

Corticon Designer - Progress Developer Studio
File Edit Navigate Search Project Run Window Help
Corticon Des...
Rule Vocabulary
Project Expl...
Rule Operat...
Rule Statements
Rule Messages
Properties
Problems
Progress Exchange 2013
Docum
Rulefloy
Rulshe
Ruletes
Vocabu
Unc
New
Go Into
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Move...
Rename... F2
Import...
Export...
Refresh F5
Project...
File
Folder
Rule Project
Rule Vocabulary
Ruleflow
Rulesheet
Ruletest
Other... Ctrl+N

Figure 10



Enter or select the parent folder: Progress Exchange 2013/Rulesheet

File name: RiskRating

Figure 11

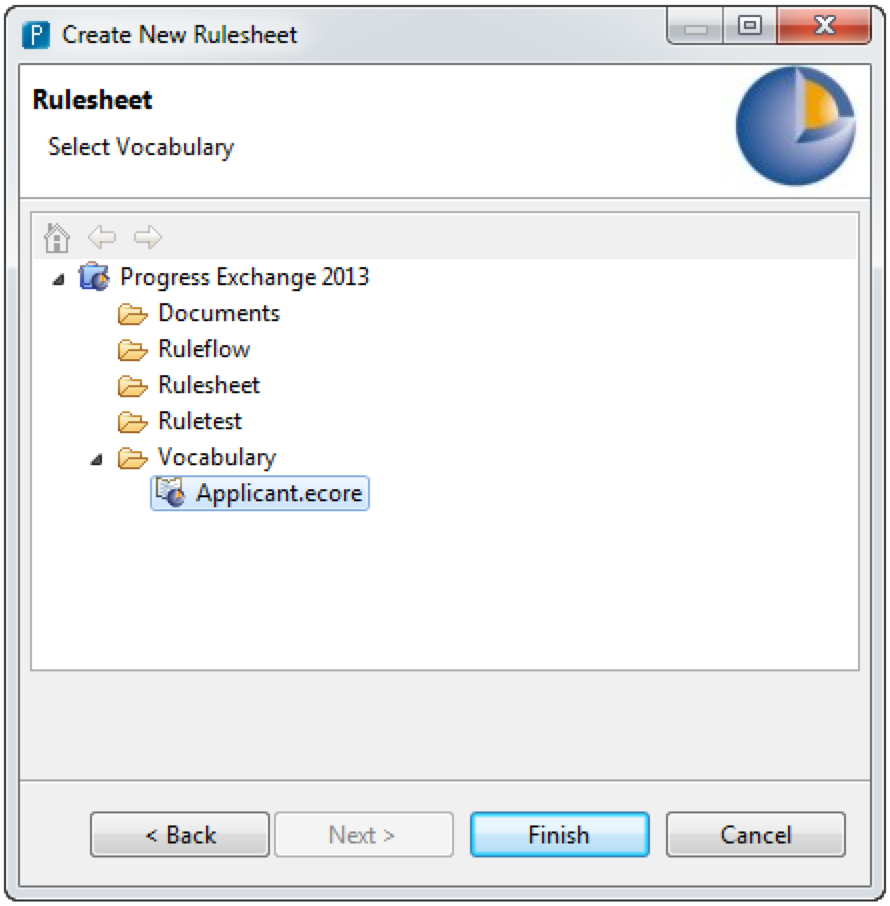

Create New Rulesheet
Rulesheet
Select Vocabulary
Progress Exchange 2013
Documents
Ruleflow
Rulesheet
Ruletest
Vocabulary
Applicant.ecore
< Back
Next >
Finish
Cancel

Figure 12

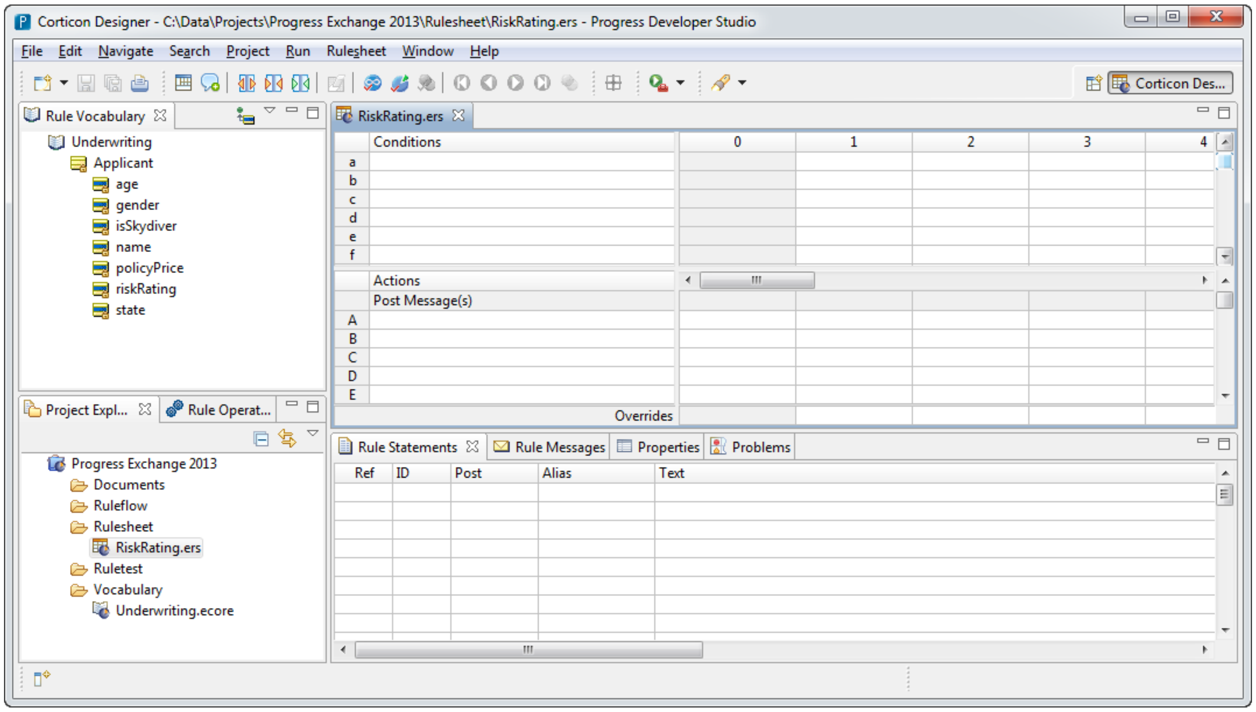
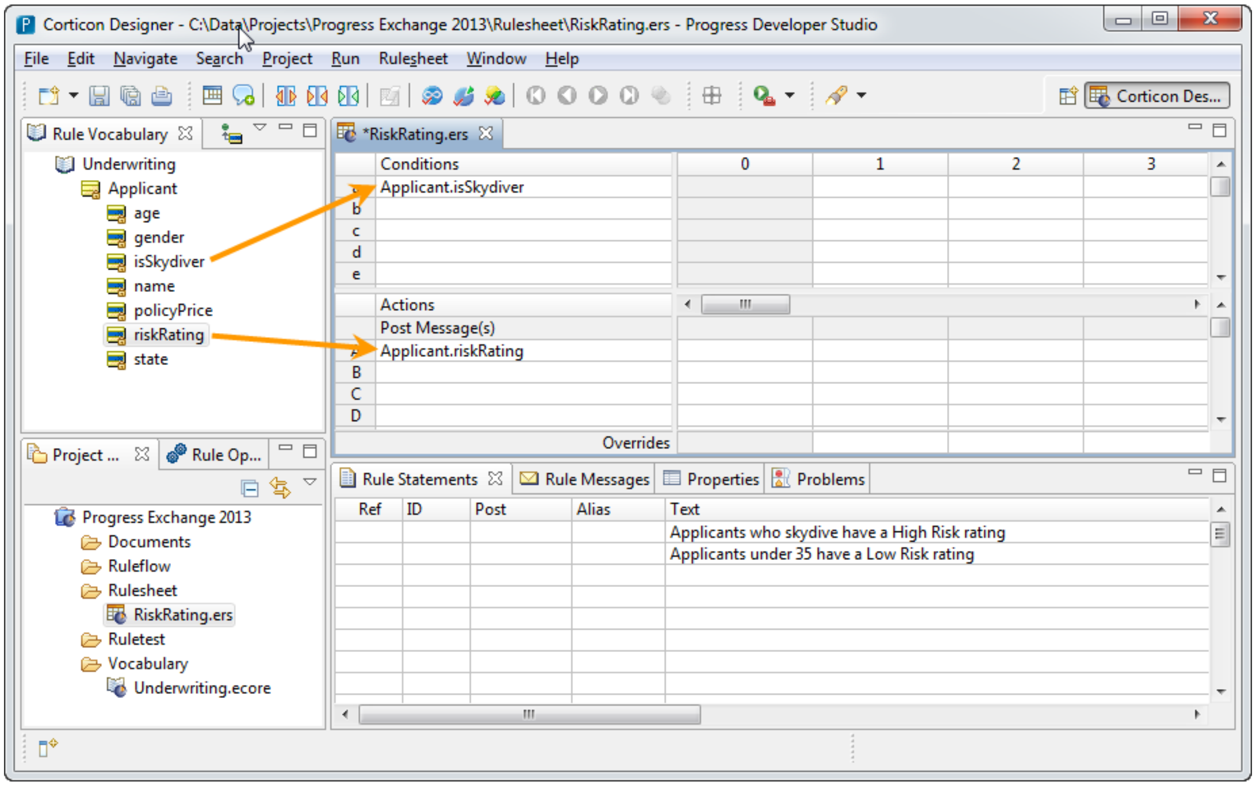


Figure 13

To model the rules, first type in the Rule Statements (Figure 14). Then you assign the conditions and actions that make up the two rules in this example. For the first rule, drag the “isSkydiver” attribute over into the first row of the Conditions area. Then drag “riskRating” to the first row of the Actions area (Figure 14). In Column 1, select “T” as the value for the Condition and type in “High Risk” as the value for the Action. (Figure 15).

For the second rule, drag “age” attribute to the second row in the Condition area. In Colum two type in “<35” for the Condition value, and type in “Low Risk” for the action value (Figure 16).

Finally, link together the Rule Statements with the declarative representation, but typing in the corresponding column number. You can also choose to post the Rule Statements as shown (Figure 17).

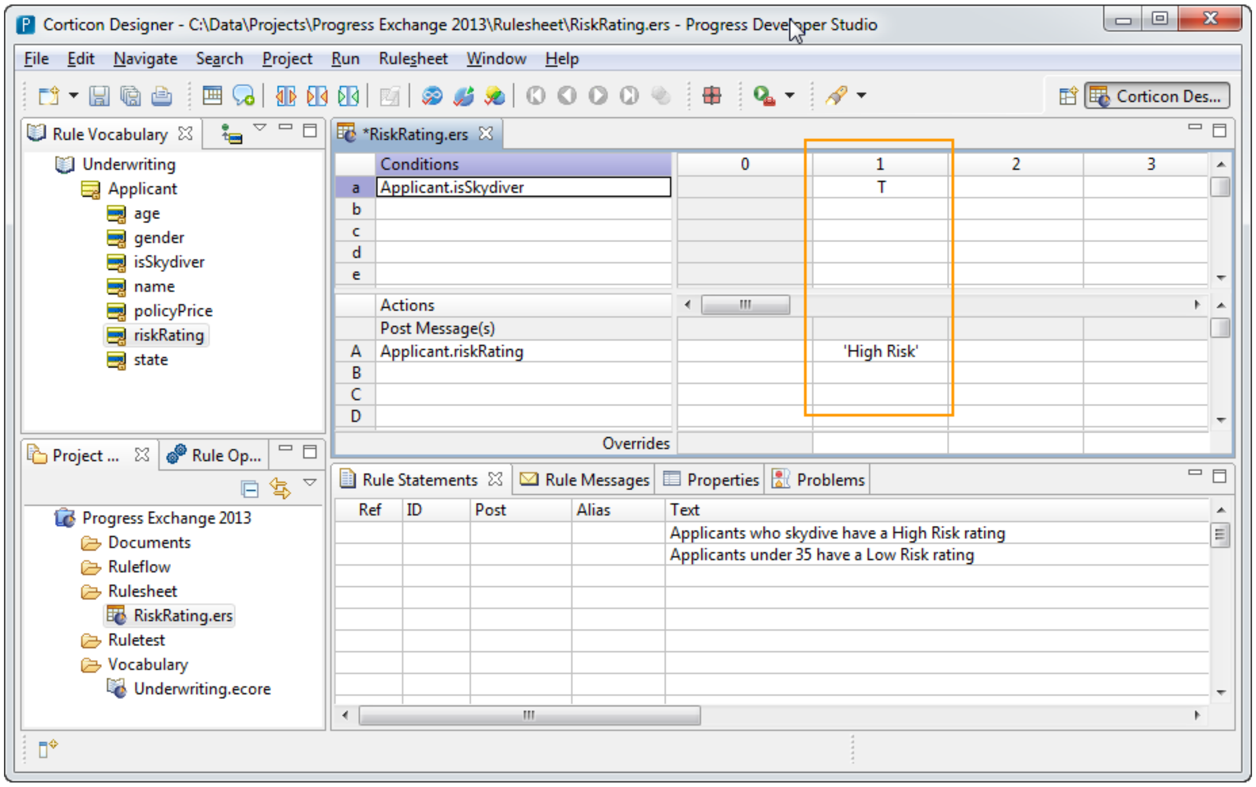


	Conditions	0	1	2	3
a	Applicant.isSkydiver				
b					
c					
d					
e					

	Actions				
	Post Message(s)				
A	Applicant.riskRating				
B					
C					
D					
	Overrides				

Ref	ID	Post	Alias	Text
				Applicants who skydive have a High Risk rating
				Applicants under 35 have a Low Risk rating

Figure 14

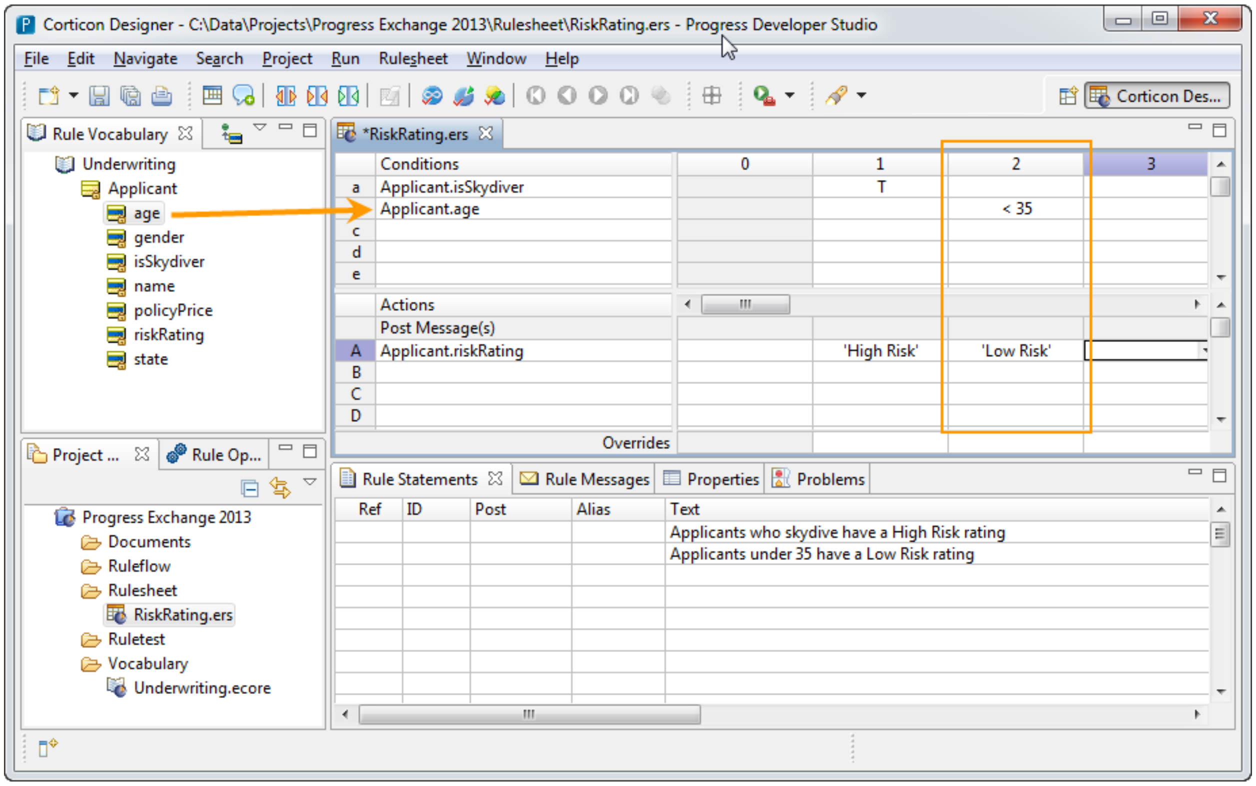


	Conditions	0	1	2	3
a	Applicant.isSkydiver		T		
b					
c					
d					
e					

	Actions				
	Post Message(s)				
A	Applicant.riskRating		'High Risk'		
B					
C					
D					
	Overrides				

Ref	ID	Post	Alias	Text
				Applicants who skydive have a High Risk rating
				Applicants under 35 have a Low Risk rating

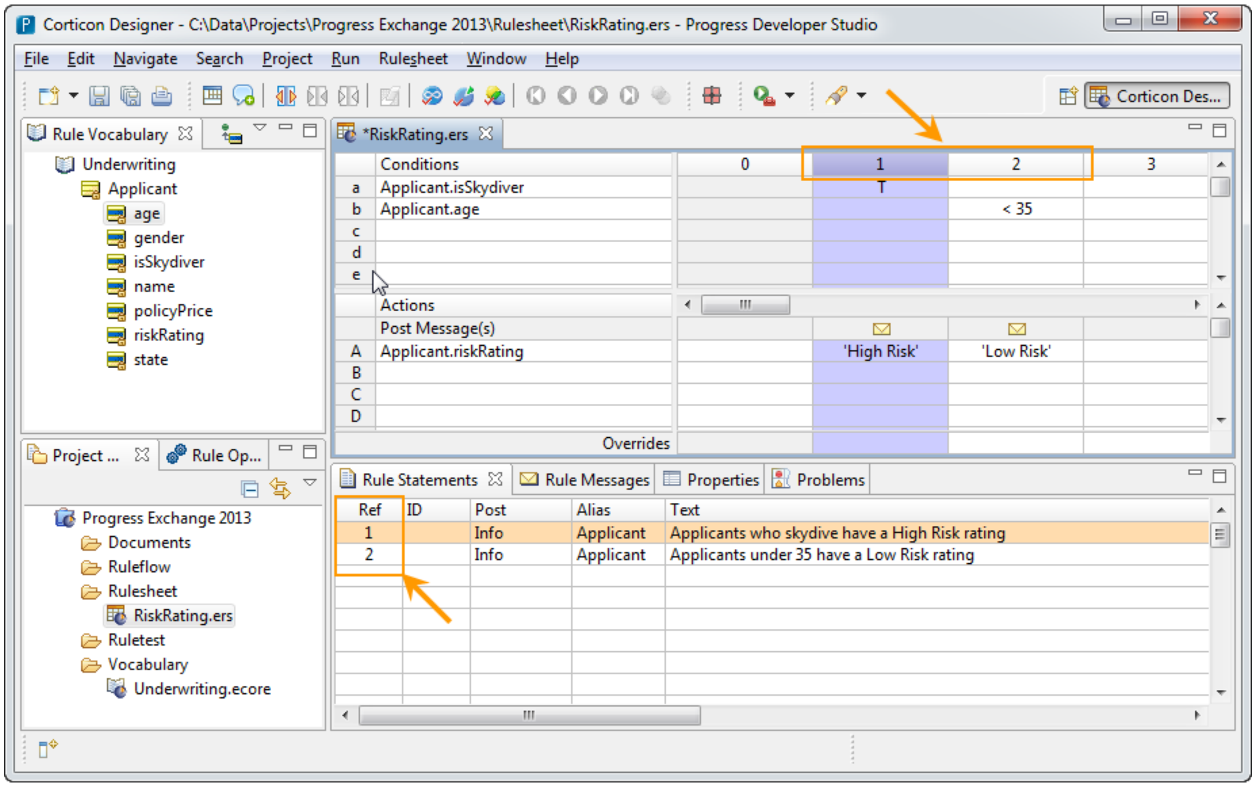
Figure 15



	Conditions	0	1	2	3
a	Applicant.isSkydiver		T		
	Applicant.age			< 35	
c					
d					
e					
	Actions				
	Post Message(s)				
A	Applicant.riskRating		'High Risk'	'Low Risk'	
B					
C					
D					
	Overrides				

Ref	ID	Post	Alias	Text
				Applicants who skydive have a High Risk rating
				Applicants under 35 have a Low Risk rating

Figure 16

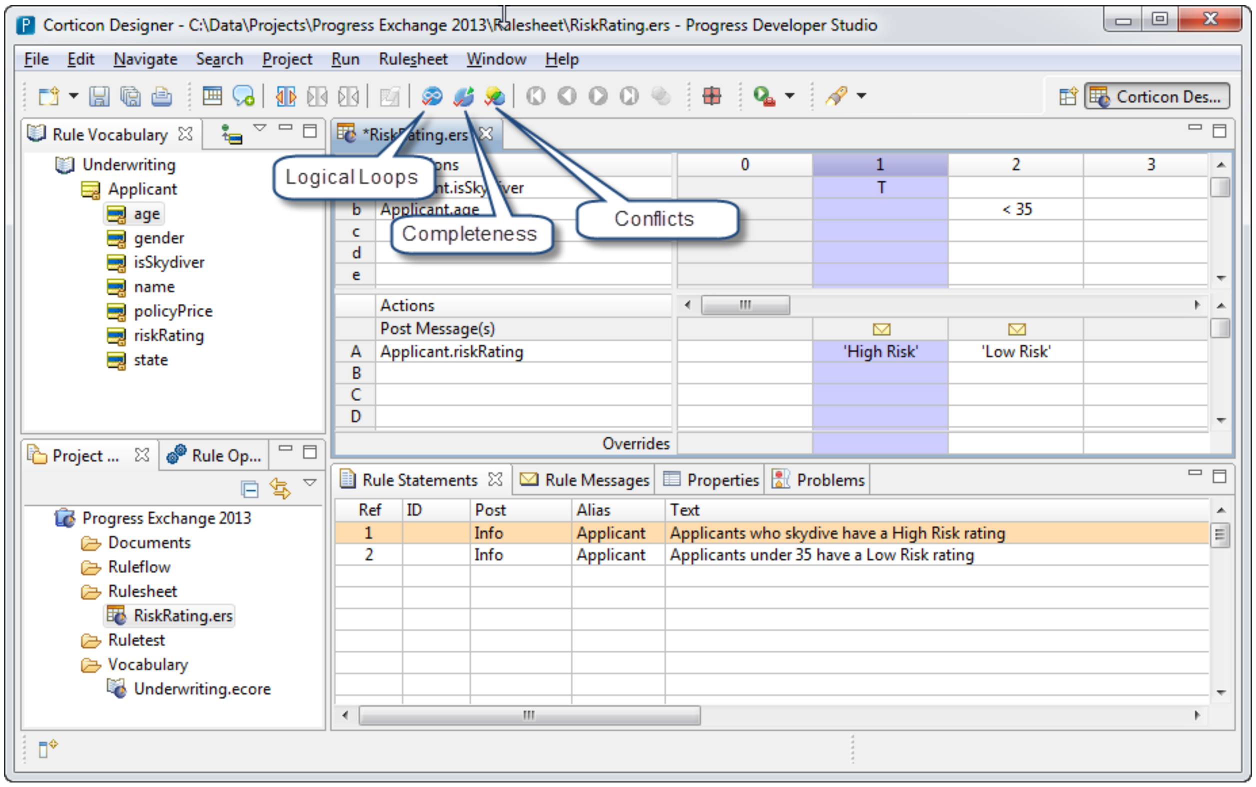


	Conditions	0	1	2	3
a	Applicant.isSkydiver		T		
b	Applicant.age			< 35	
c					
d					
e					
	Actions				
	Post Message(s)		✉	✉	
A	Applicant.riskRating		'High Risk'	'Low Risk'	
B					
C					
D					
	Overrides				

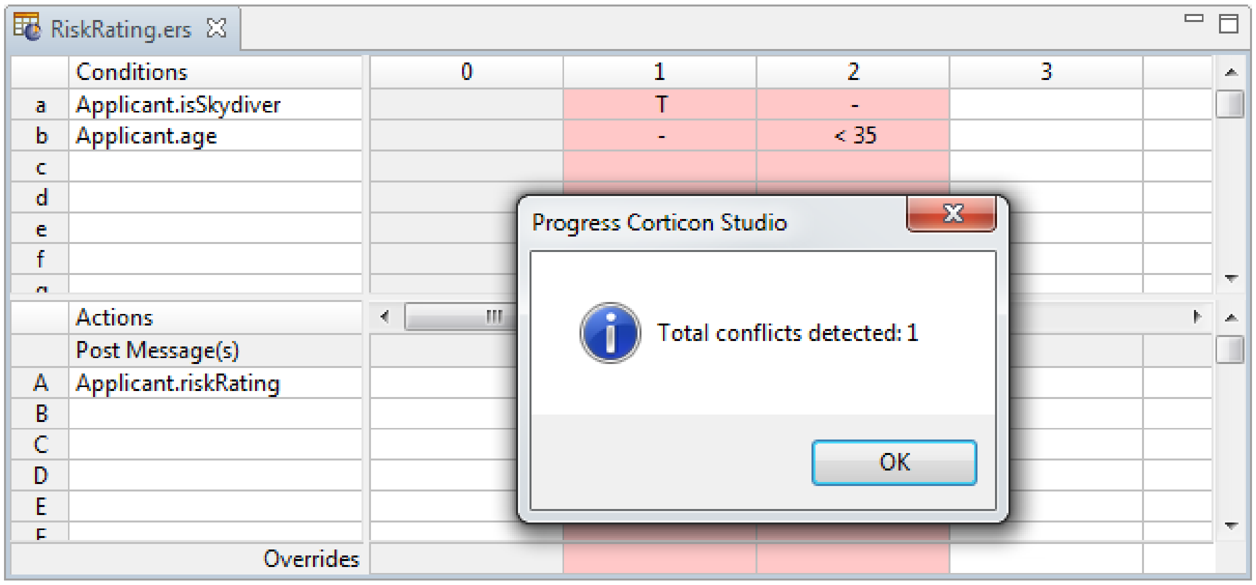
Ref	ID	Post	Alias	Text
1		Info	Applicant	Applicants who skydive have a High Risk rating
2		Info	Applicant	Applicants under 35 have a Low Risk rating

Figure 17

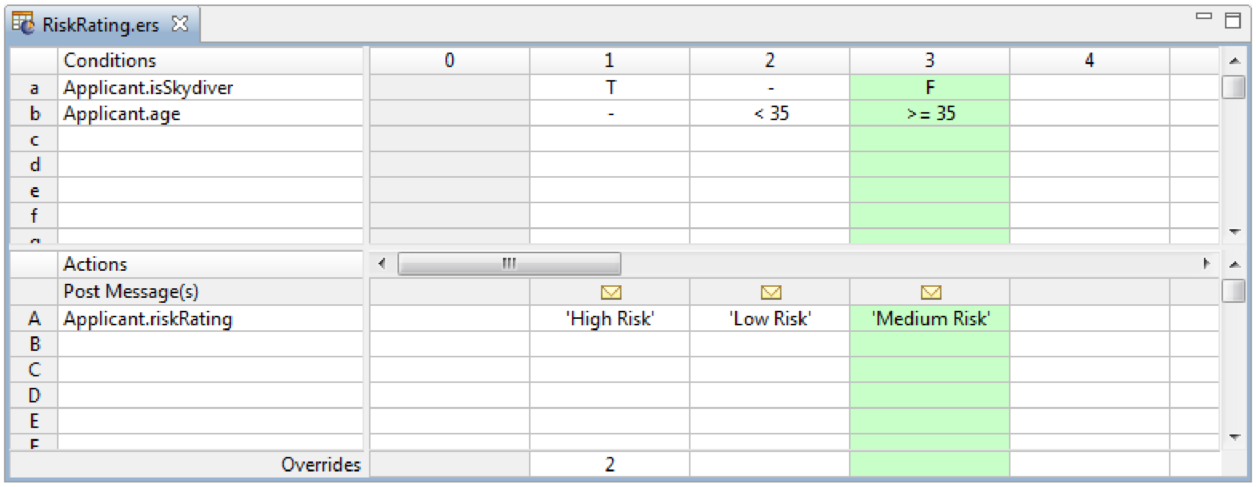
Now we can check for any flaws in the rule model, such as conflicts and incompleteness. This means there are rules that overlap, or there a gaps in the logic and not enough rules to cover all possible scenarios. Conflicts can be resolved by using overrides, and incompleteness is handled by adding additional rules as suggested.



Ref	ID	Post	Alias	Text
1		Info	Applicant	Applicants who skydive have a High Risk rating
2		Info	Applicant	Applicants under 35 have a Low Risk rating



Total conflicts detected: 1


RiskRating.ers

	Conditions	0	1	2	3	4
a	Applicant.isSkydiver		T	-	F	
b	Applicant.age		-	< 35	>= 35	
c						
d						
e						
f						

	Actions					
	Post Message(s)		✉	✉	✉	
A	Applicant.riskRating		'High Risk'	'Low Risk'	'Medium Risk'	
B						
C						
D						
E						
	Overrides		2			

Step 3 – Create Tests

Right click on the Project and select New/Ruletest. Name it **Applicant** and save it in the Ruletest folder. You will then choose the Rulesheet that you want to test against. Select the RiskRating Rulesheet you have just created (Figure 18).

You can now drag the Applicant from the Vocabulary area and drop into the Input section. Double click on the age attribute and type in 25. The double click isSkydiver and choose “false”. You can provide a name to identify this applicant, although it is not necessary for the decision to execute. You may also delete the other attributes for clarity’s sake if desired (Figure 19).

Once complete, press the “Execute” button to run the test and see the outcome of the decision, as well as the corresponding Rule Statements for every rule that fired (Figure 20, 21).

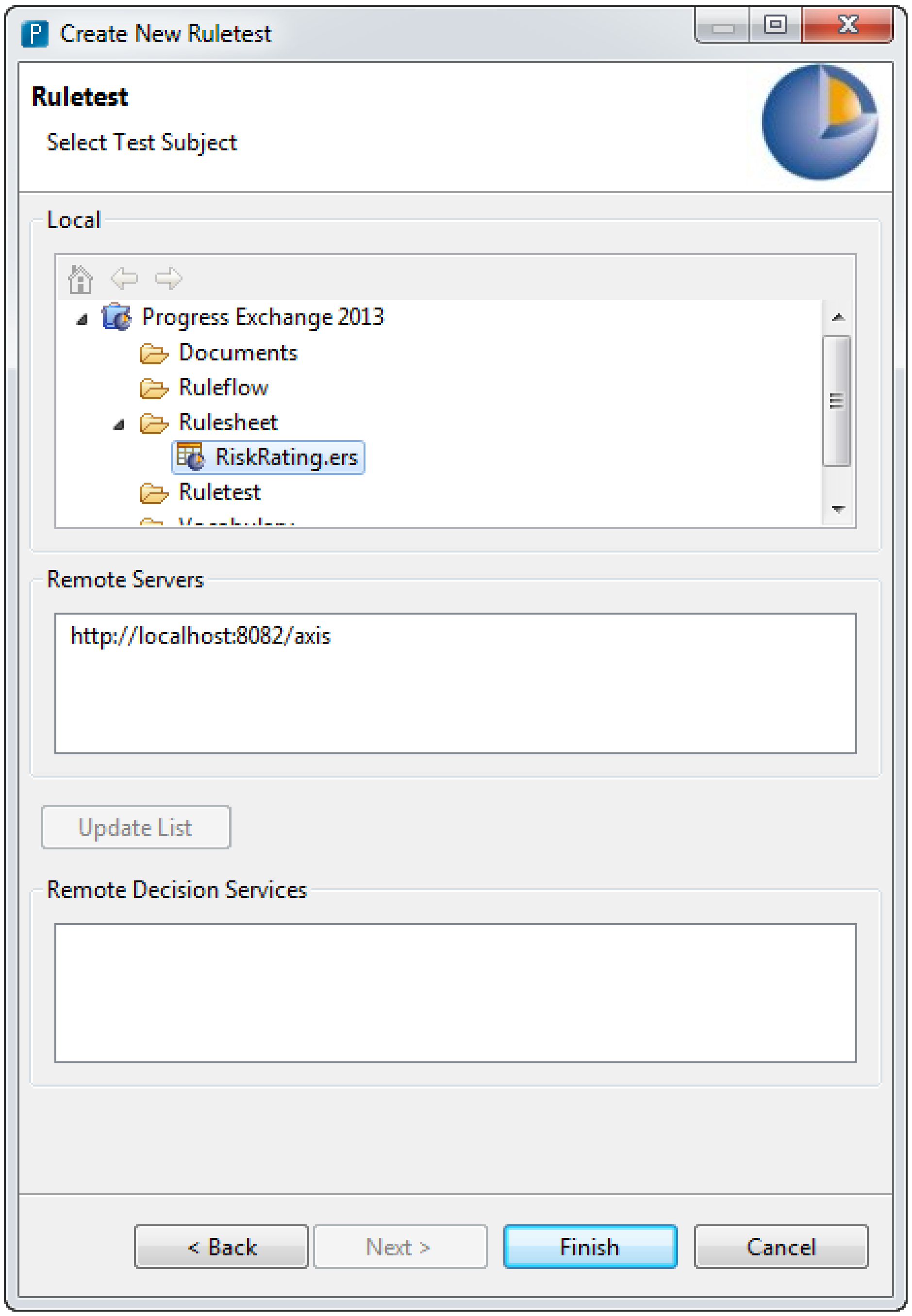

Create New Ruletest
Ruletest
Select Test Subject
Local
Progress Exchange 2013
Documents
Ruleflow
Rulesheet
RiskRating.ers
Ruletest
Remote Servers
http://localhost:8082/axis
Update List
Remote Decision Services
< Back
Next >
Finish
Cancel

Figure 18

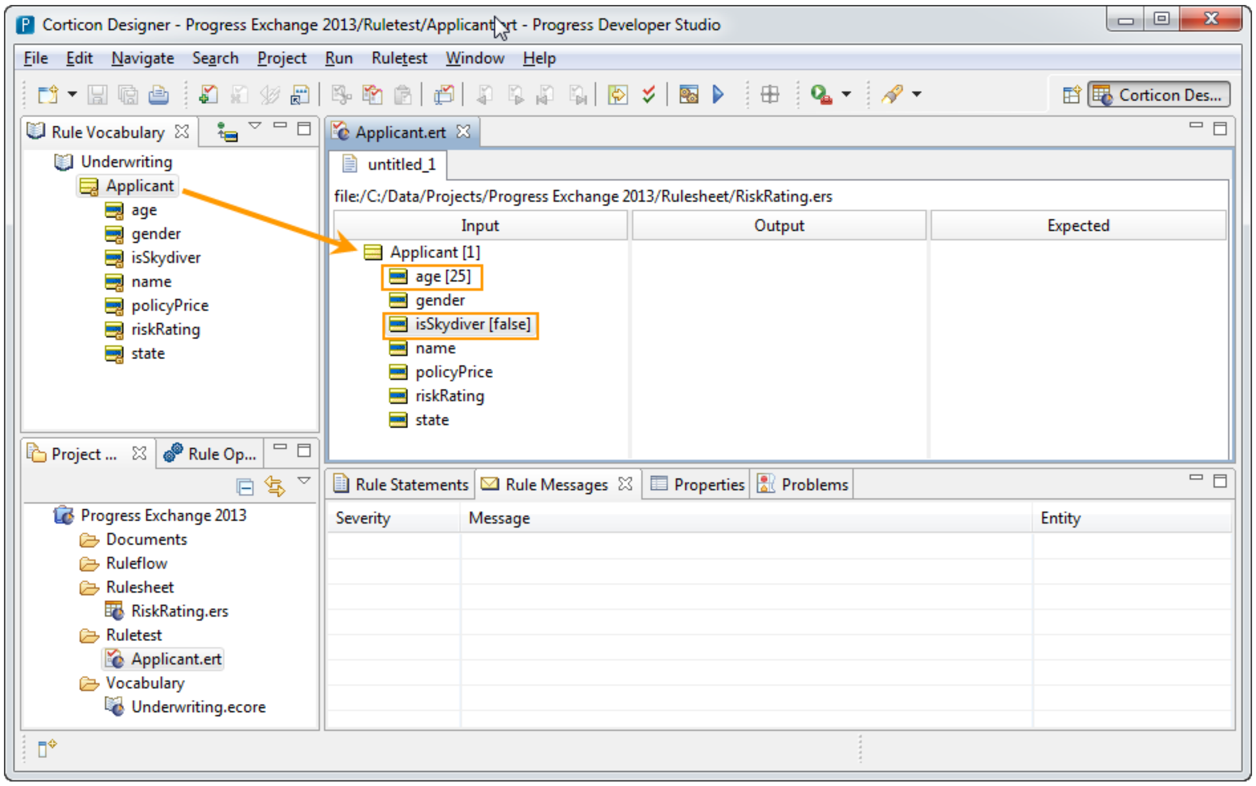
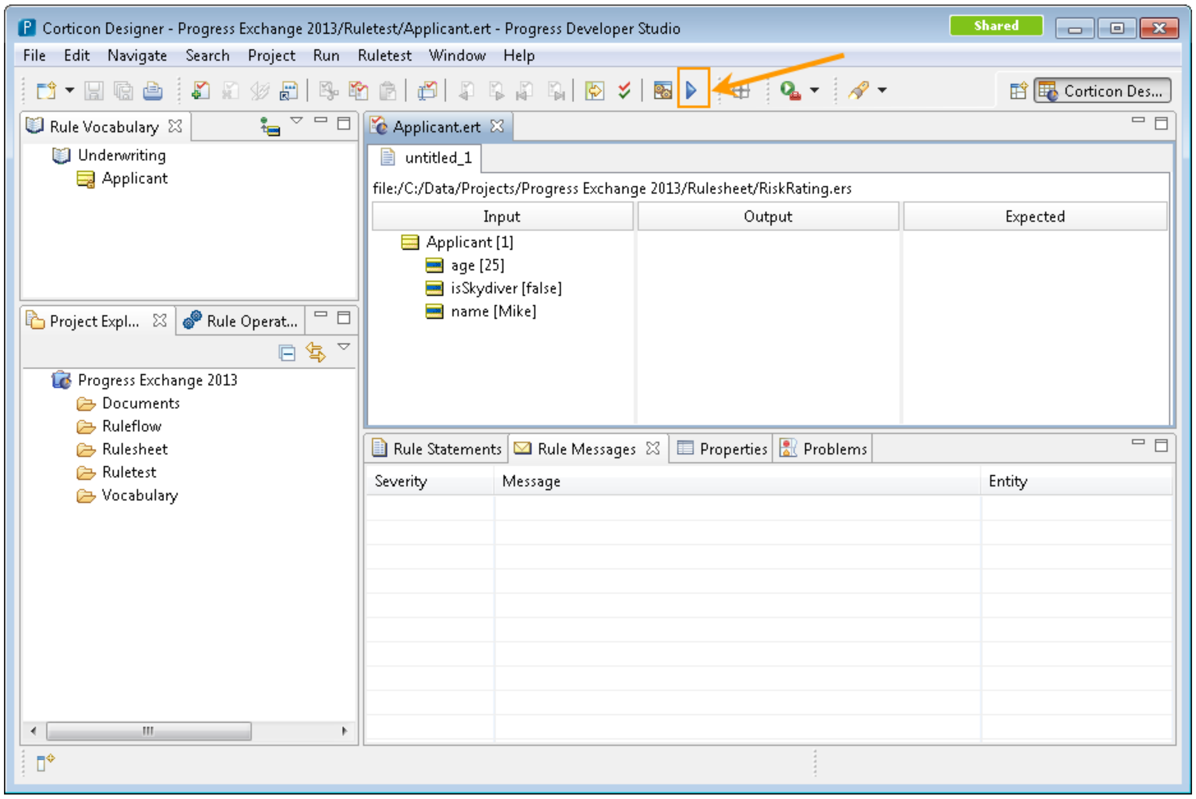

Corticon Designer - Progress Exchange 2013/Ruletest/Applicant.ert - Progress Developer Studio
File Edit Navigate Search Project Run Ruletest Window Help
Corticon Des...
Rule Vocabulary
Underwriting
Applicant
age
gender
isSkydiver
name
policyPrice
riskRating
state
Applicant.ert
untitled_1
file:/C:/Data/Projects/Progress Exchange 2013/Rulesheet/RiskRating.ers
Input
Output
Expected
Applicant [1]
age [25]
gender
isSkydiver [false]
name
policyPrice
riskRating
state
Project ...
Rule Op...
Progress Exchange 2013
Documents
Ruleflow
Rulesheet
RiskRating.ers
Ruletest
Applicant.ert
Vocabulary
Underwriting.ecore
Rule Statements
Rule Messages
Properties
Problems
Severity
Message
Entity

Figure 19

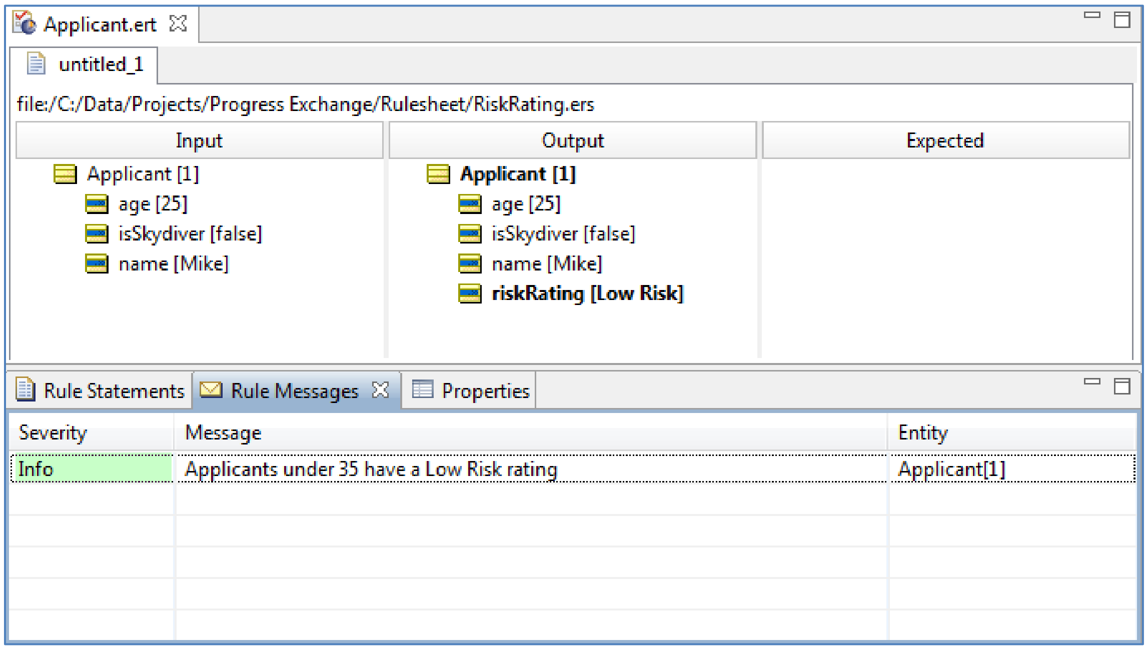


file:/C:/Data/Projects/Progress Exchange 2013/Rulesheet/RiskRating.ers

Input	Output	Expected
Applicant [1]		
age [25]		
isSkydiver [false]		
name [Mike]		

Severity	Message	Entity

Figure 20



file:/C:/Data/Projects/Progress Exchange/Rulesheet/RiskRating.ers

Input	Output	Expected
Applicant [1]	**Applicant [1]**	
age [25]	age [25]	
isSkydiver [false]	isSkydiver [false]	
name [Mike]	name [Mike]	
	riskRating [Low Risk]	

Severity	Message	Entity
Info	Applicants under 35 have a Low Risk rating	Applicant[1]

Figure 21

Step 4 – Deploy Decision Service

A decision service is a collection of one or more Rulesheets orchestrated in a Ruleflow. To deploy a decision service, first create a new Ruleflow by right mouse clicking on the project and selecting New/Ruleflow. Name the Ruleflow **NewPolicyProcessing (**Figure 21)**.** Next select the Vocabulary to use for this Ruleflow. Choose the Underwriting.ecore that you created earlier (Figure 23). Finally, drag the RiskRating.ers Rulesheet from the Project Explorer on to the Ruleflow canvas.

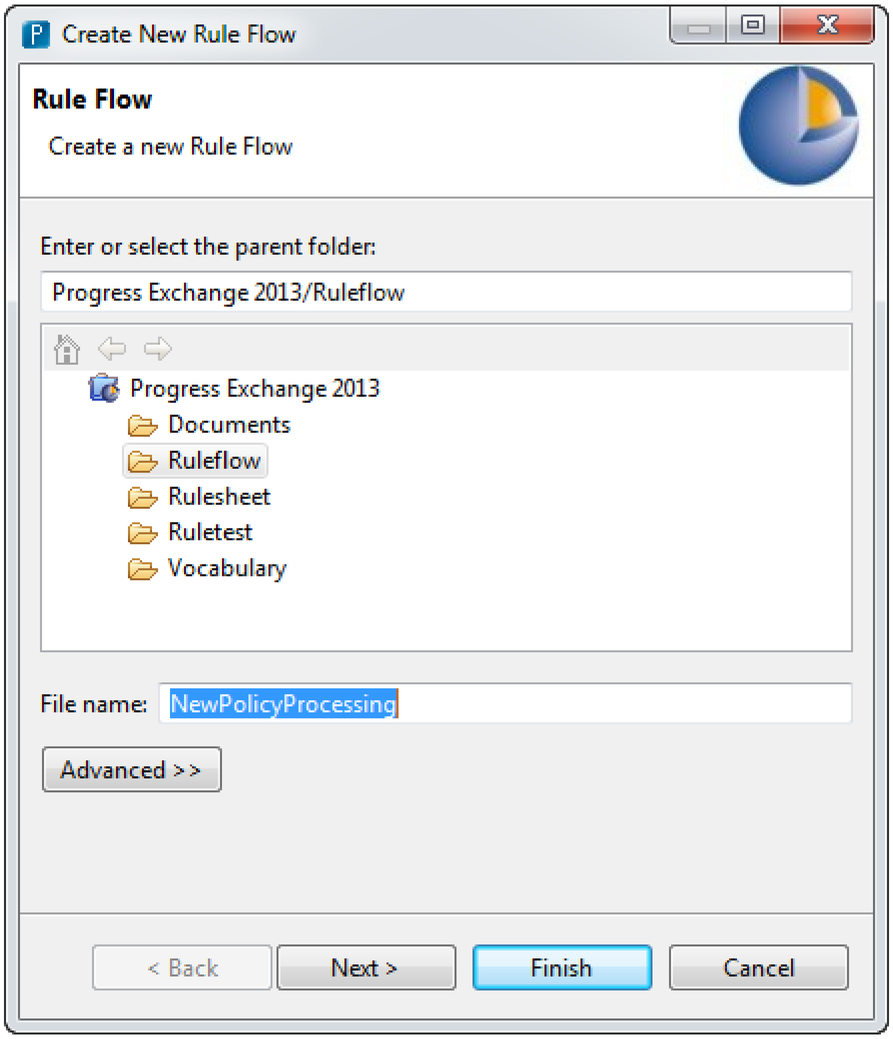


Figure 22

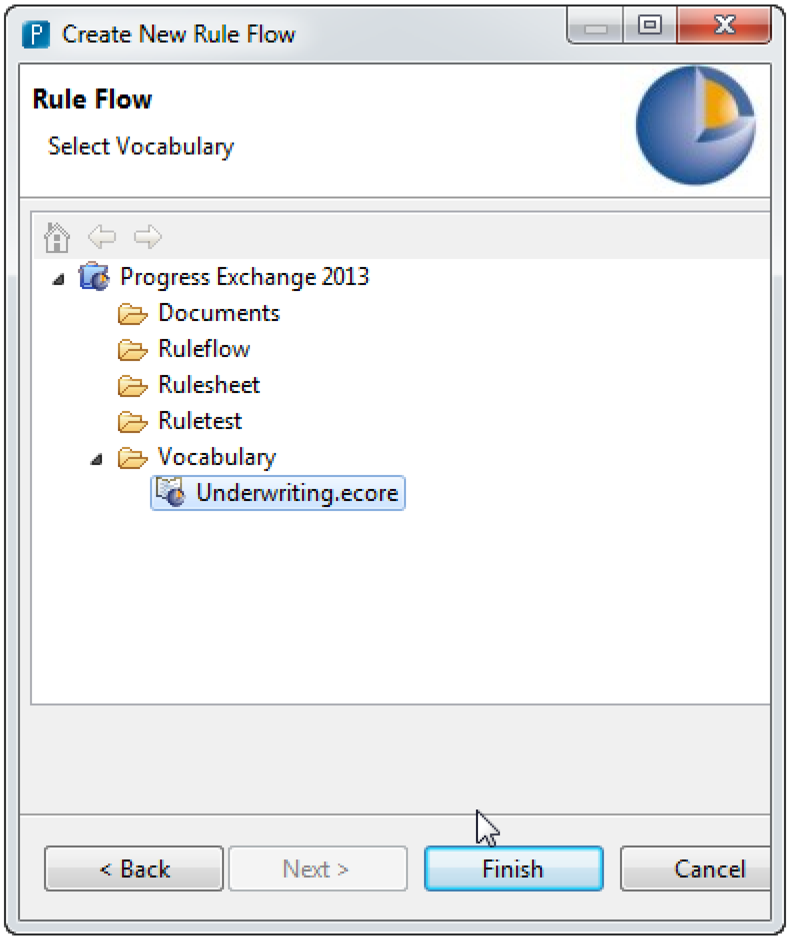

Create New Rule Flow
Rule Flow
Select Vocabulary
Progress Exchange 2013
Documents
Ruleflow
Rulesheet
Ruletest
Vocabulary
Underwriting.ecore
< Back
Next >
Finish
Cancel

Figure 23

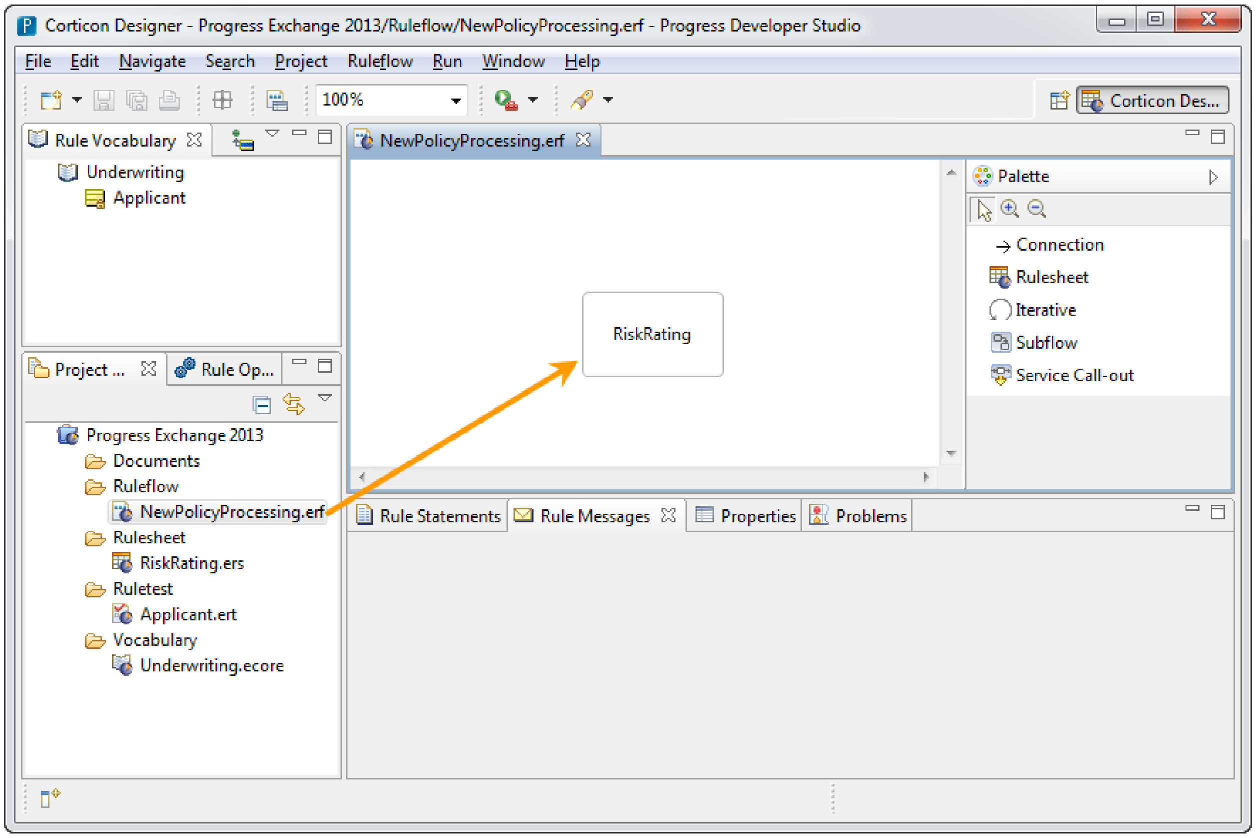


Figure 24

The decision service is completed and ready to be deployed to the Corticon Server. Start the Corticon Server from the desktop, and ensure that the application server is running (Figure 25, 26). In Corticon Studio, right mouse click on the project and select “Publish”. Provide the connection details to the Corticon Server you want to publish to (Figure 28).

URL: http://localhost:8082/axis

User: admin

Password: admin

Select the “NewPolicyProcessing” flow and Finish (Figure 29).

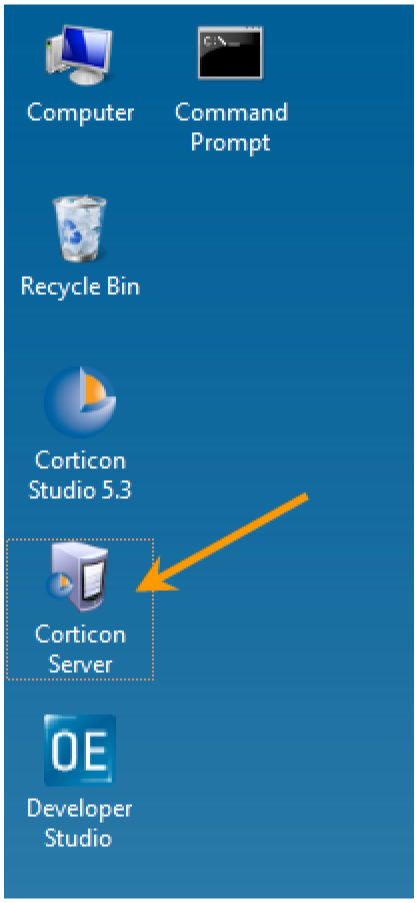
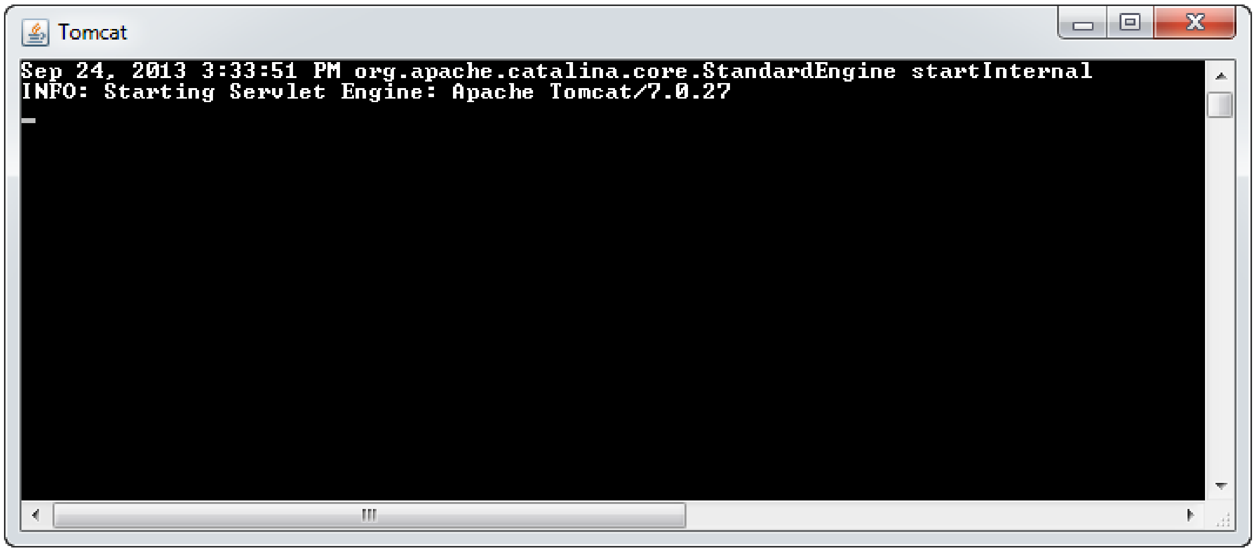
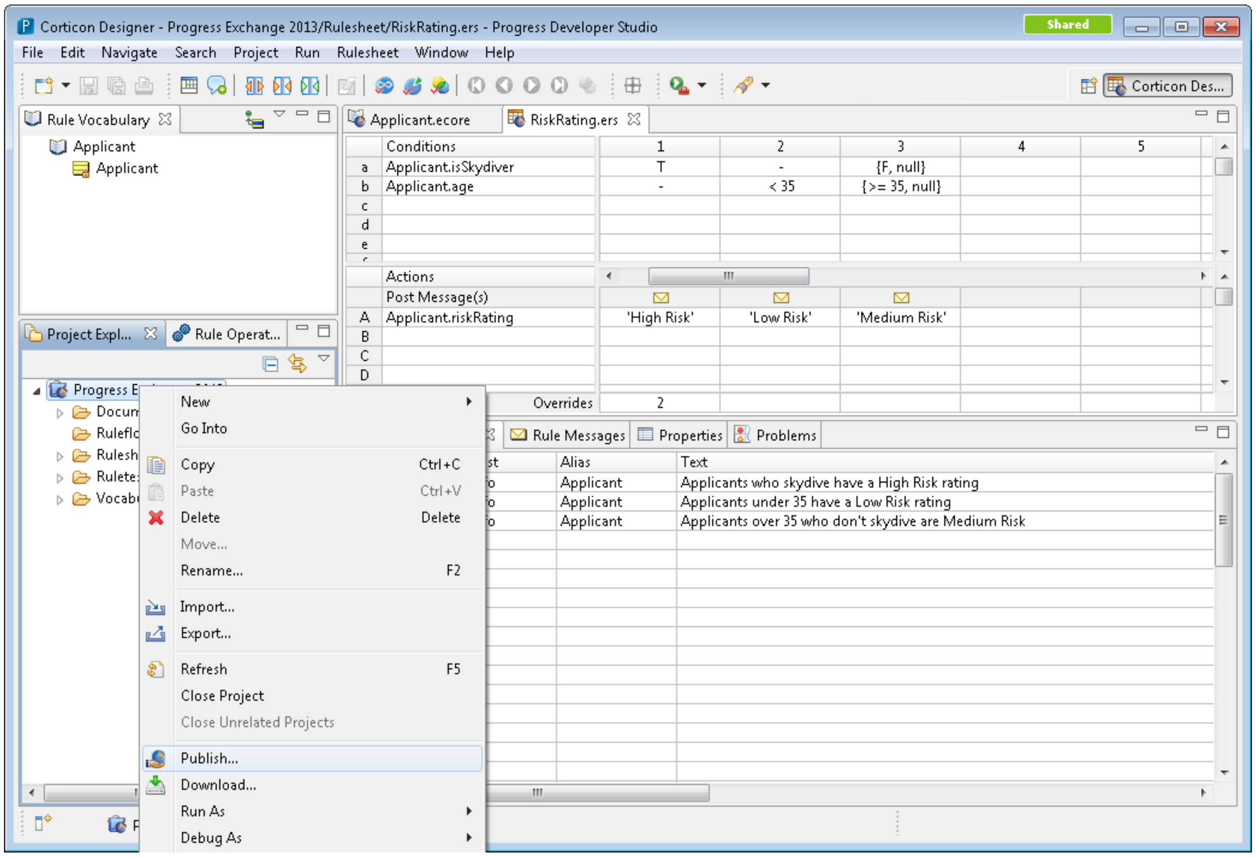


Figure 25



```
Sep 24, 2013 3:33:51 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.27
_
```

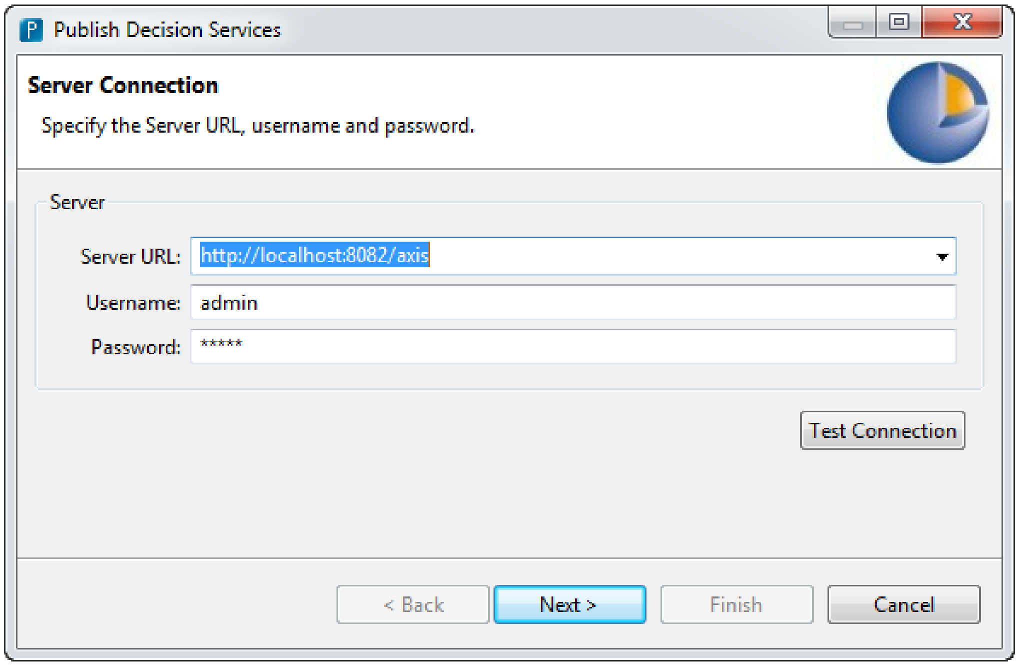
Figure 26



	Conditions	1	2	3	4	5
a	Applicant.isSkydiver	T	-	{F, null}		
b	Applicant.age	-	< 35	{>= 35, null}		
	Actions					
	Post Message(s)					
A	Applicant.riskRating	'High Risk'	'Low Risk'	'Medium Risk'		
	Overrides	2				

Alias	Text
Applicant	Applicants who skydive have a High Risk rating
Applicant	Applicants under 35 have a Low Risk rating
Applicant	Applicants over 35 who don't skydive are Medium Risk

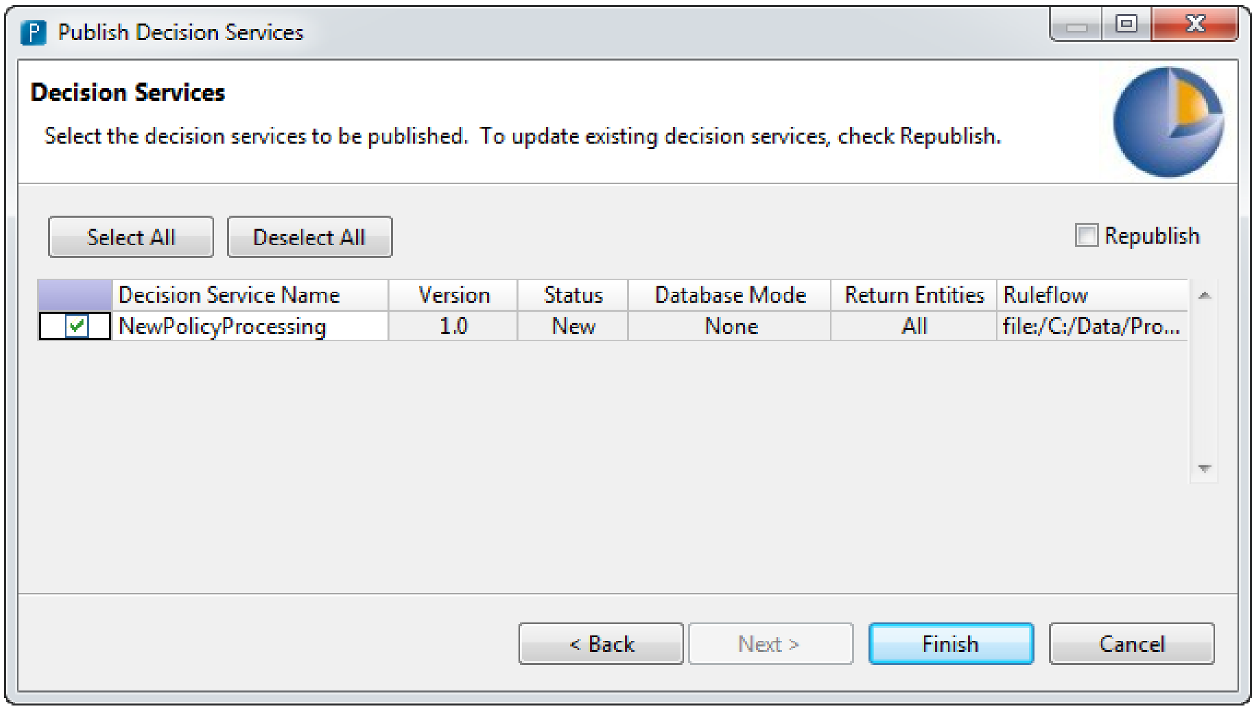
Figure 27



Server Connection
Specify the Server URL, username and password.

Server
Server URL: http://localhost:8082/axis
Username: admin
Password: *****

Figure 28



Publish Decision Services

Decision Services

Select the decision services to be published. To update existing decision services, check Republish.

	Decision Service Name	Version	Status	Database Mode	Return Entities	Ruleflow
✓	NewPolicyProcessing	1.0	New	None	All	file:/C:/Data/Pro...

Figure 29

To access the Corticon Server Console and confirm the service is deployed, launch the web browser from the desktop and go to the URL: http://localhost:8082/axis

User: admin

Password: admin

Once logged in you can access the deployed decision services (Figure 30), and see a list of all decisions and various details. You should see v1.0 of NewPolicyProcessing, along with some other default decision services (Figure 31).

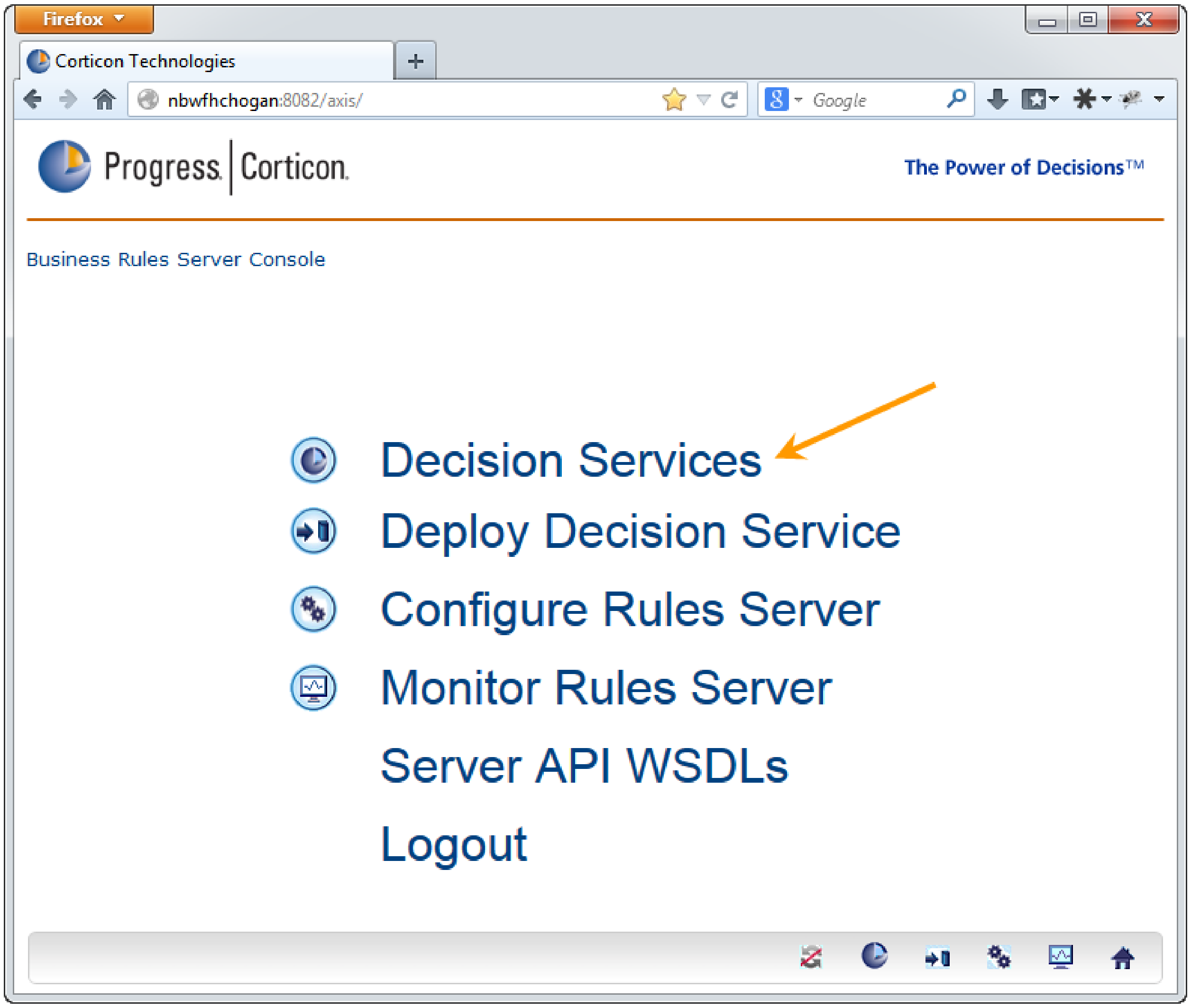
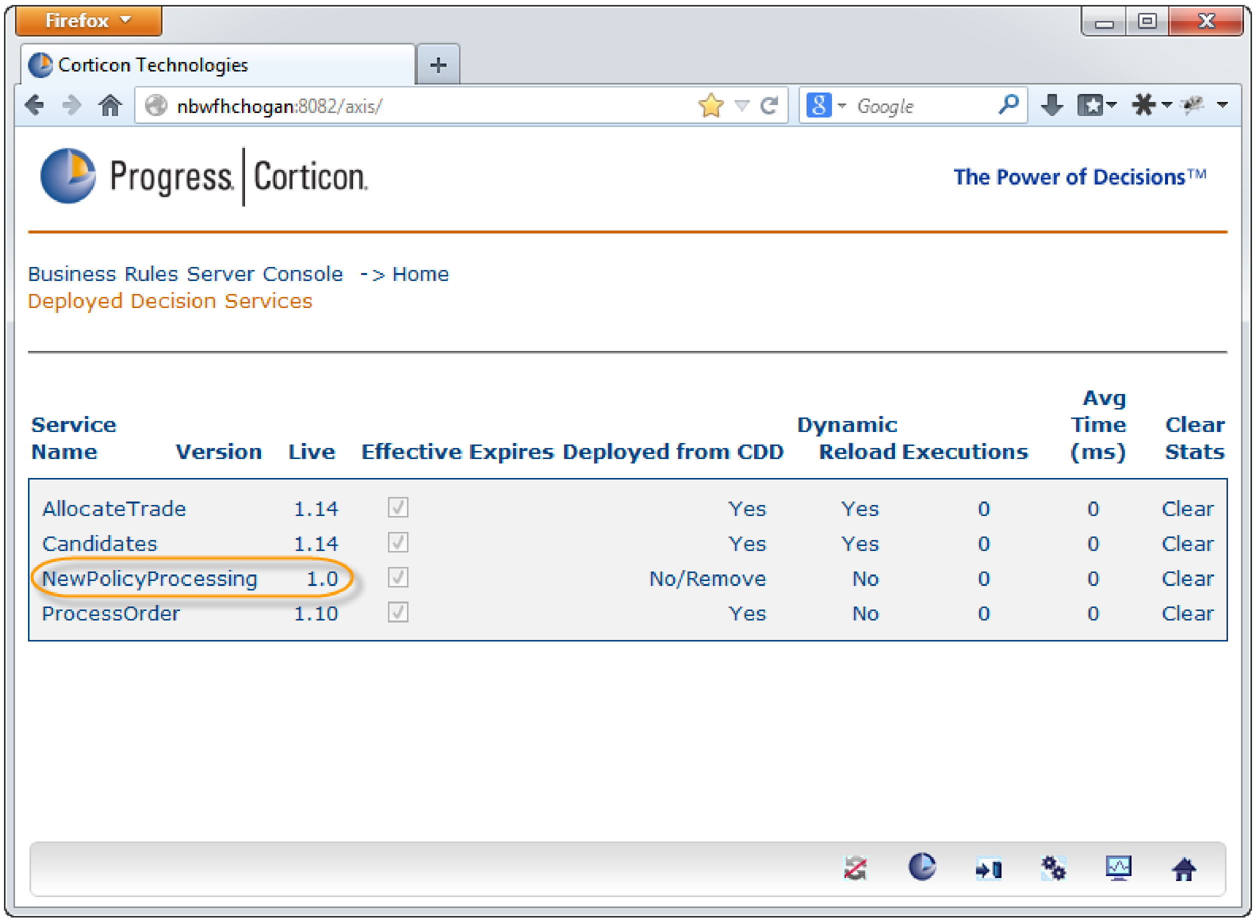

Firefox
Corticon Technologies
nbwfhchogan:8082/axis/
Google
Progress | Corticon
The Power of Decisions™
Business Rules Server Console
Decision Services
Deploy Decision Service
Configure Rules Server
Monitor Rules Server
Server API WSDLs
Logout

Figure 30



Business Rules Server Console -> Home
Deployed Decision Services

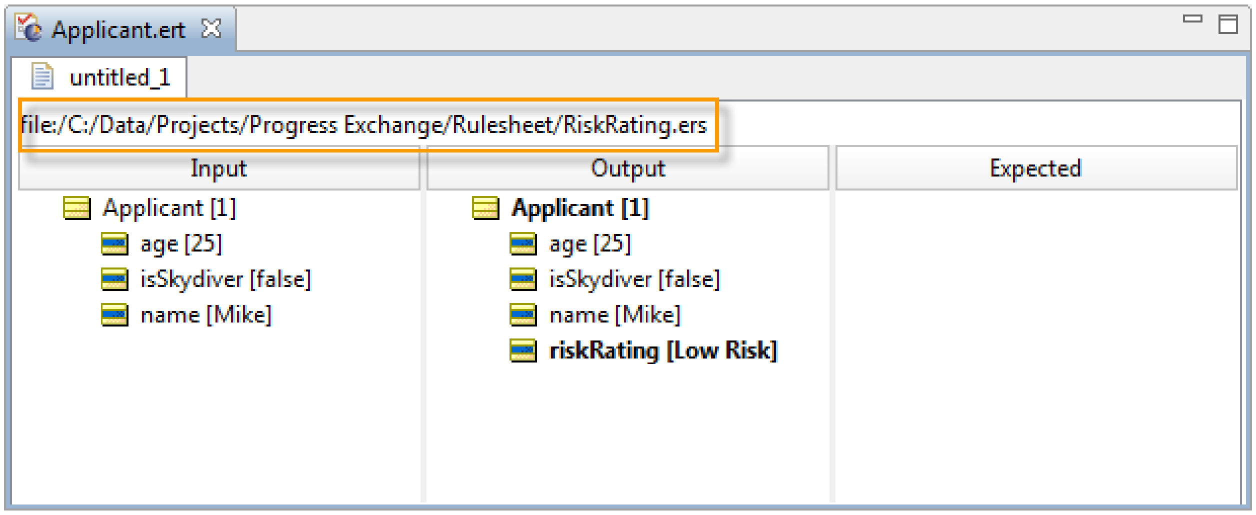
Service Name	Version	Live	Effective	Expires	Deployed from CDD	Dynamic Reload	Executions	Avg Time (ms)	Clear Stats
AllocateTrade	1.14	✓			Yes	Yes	0	0	Clear
Candidates	1.14	✓			Yes	Yes	0	0	Clear
NewPolicyProcessing	1.0	✓			No/Remove	No	0	0	Clear
ProcessOrder	1.10	✓			Yes	No	0	0	Clear

Figure 31

Step 5 – Test Deployed Decision Service

Corticon Studio allows you to test against remote decision services. To do see, return to the Ruletest you created, and update the tested asset. Double click on the file path of the RiskRating.ers file (Figure 32), choose "Remote Servers", click the "Update List" button, and select NewPolicyProcessing in the list (Figure 33).

Now when you execute the test, Corticon Studio will create a SOAP message, post it to the remote URL and display the result payload from the deployed decision service (Figure 34).

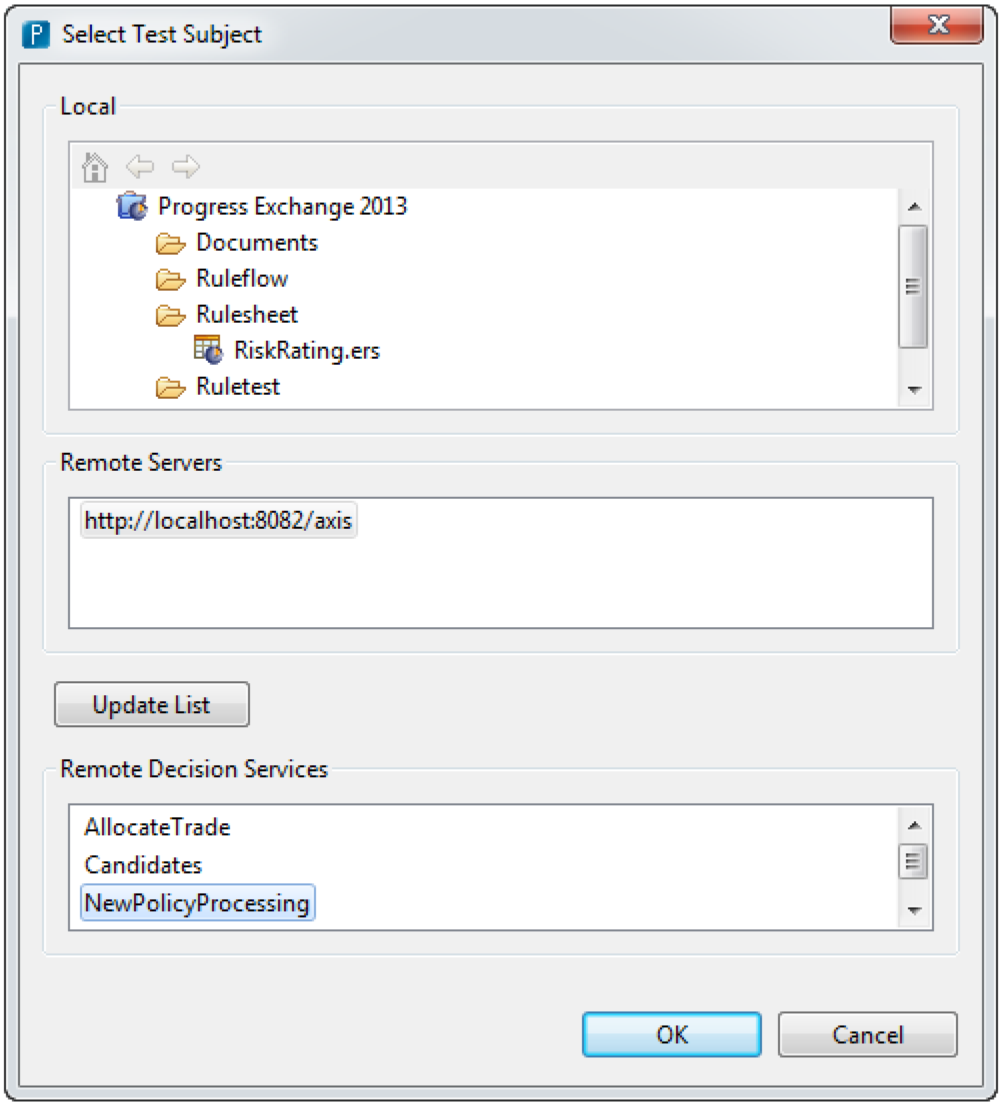

Applicant.ert

untitled_1

file:/C:/Data/Projects/Progress Exchange/Rulesheet/RiskRating.ers

Input	Output	Expected
Applicant [1]	**Applicant [1]**	
age [25]	age [25]	
isSkydiver [false]	isSkydiver [false]	
name [Mike]	name [Mike]	
	riskRating [Low Risk]	

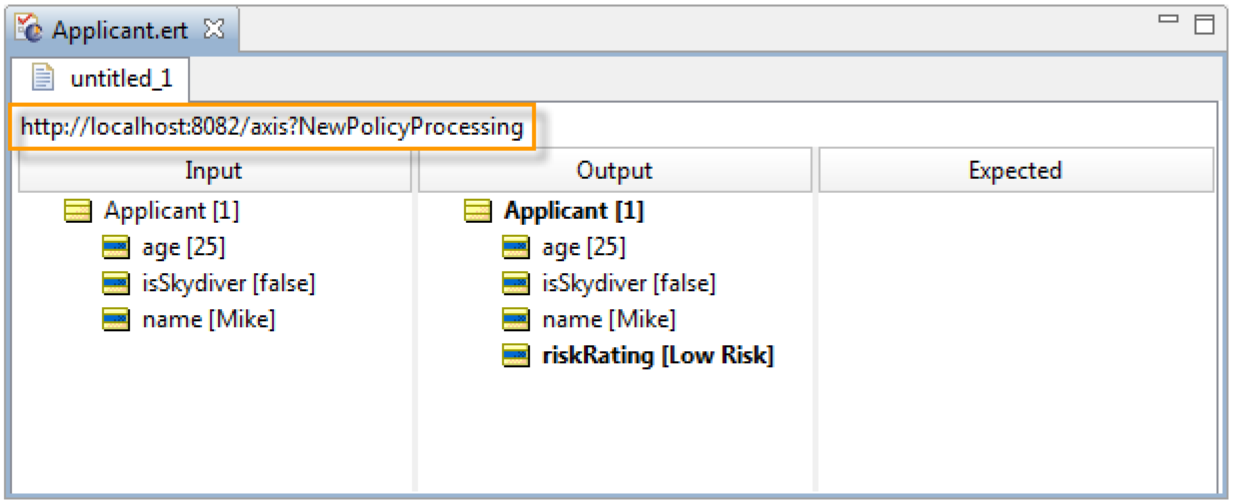
Figure 32



http://localhost:8082/axis

AllocateTrade
Candidates
NewPolicyProcessing

Figure 33



Applicant.ert

untitled_1

http://localhost:8082/axis?NewPolicyProcessing

Input	Output	Expected
Applicant [1]	**Applicant [1]**	
age [25]	age [25]	
isSkydiver [false]	isSkydiver [false]	
name [Mike]	name [Mike]	
	riskRating [Low Risk]	

Figure 34

Exercise 2 – Integrate Corticon Decision Service

In this exercise we will use the new features of OpenEdge 11.3 to integrate the deployed decision service with an existing application. Because the decision service vocabulary was created from existing Temp-Tables, these can be seamlessly used as the data payload, with no mapping or transformation necessary.

Step 1 – Configure the Environment

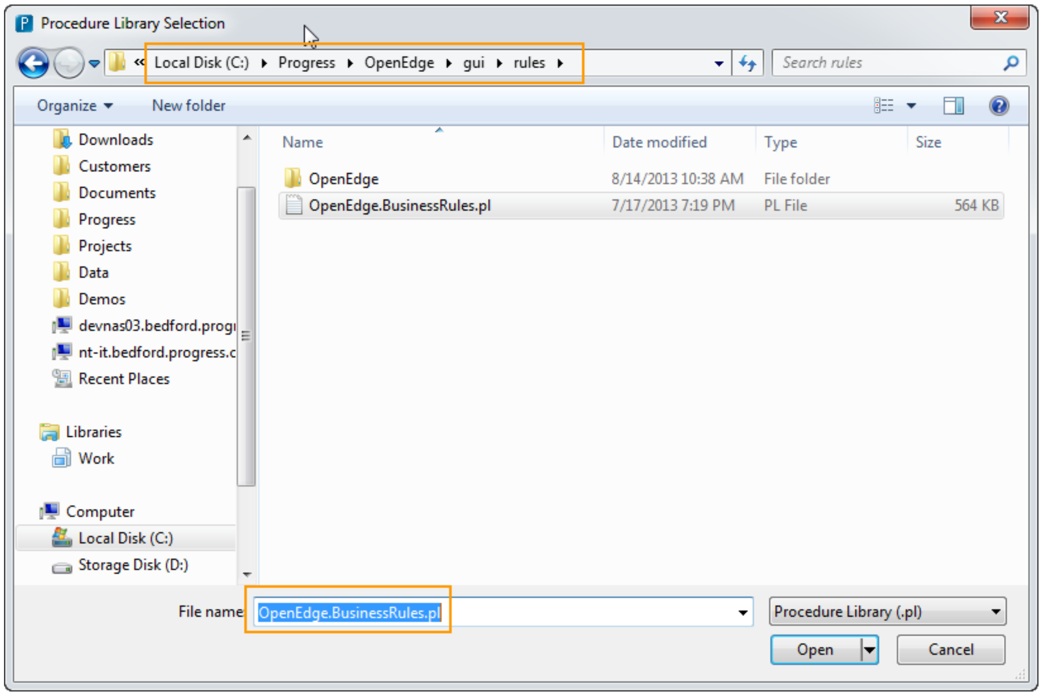
Launch Developer Studio for OpenEdge from the desktop, if it is not already running. To leverage the built in connectivity with Corticon in OE 11.3 you will need to add one External Directory, and one External Library to your project.

Right mouse click on the “Corticon Business Rules” project and select Properties. Navigate to Progress OpenEdge\PROPATH .

Click “Add External Library” and select “C:\Progress\OpenEdge\gui\rules\OpenEdge.BusinessRule.pl” (Figure 35)

Click “Add External Directory” and select “C:\Progress\OpenEdge\gui\rules” (Figure 36)

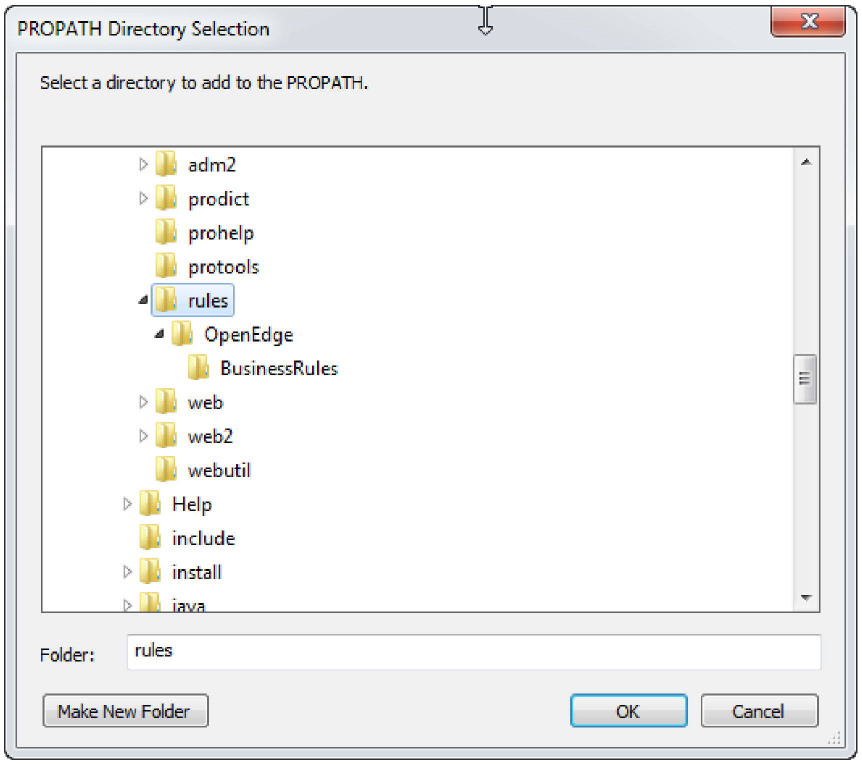
You should now see the 2 components in the Properties view (Figure 37).



Name	Date modified	Type	Size
OpenEdge	8/14/2013 10:38 AM	File folder	
OpenEdge.BusinessRules.pl	7/17/2013 7:19 PM	PL File	564 KB

File name: OpenEdge.BusinessRules.pl

Figure 35

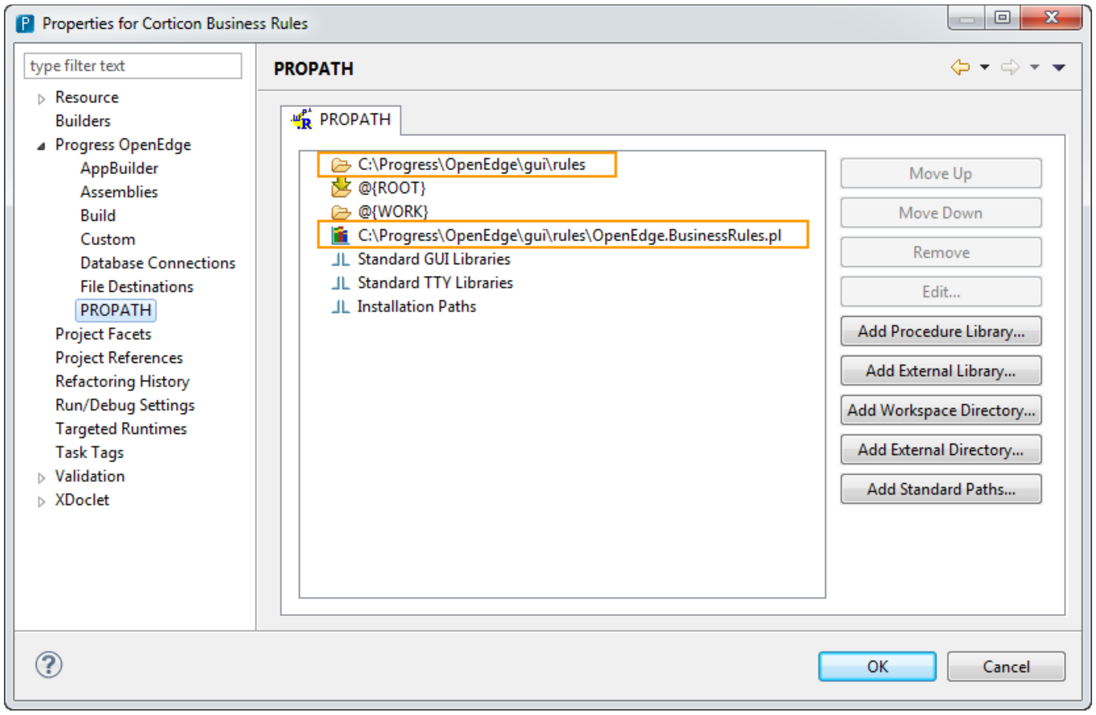

PROPATH Directory Selection

Select a directory to add to the PROPATH.

- adm2
- prodict
- prohelp
- protools
- rules
  - OpenEdge
    - BusinessRules
- web
- web2
- webutil
- Help
- include
- install
- java

Folder: rules

Figure 36


Properties for Corticon Business Rules

PROPATH

C:\Progress\OpenEdge\gui\rules
@{ROOT}
@{WORK}
C:\Progress\OpenEdge\gui\rules\OpenEdge.BusinessRules.pl
Standard GUI Libraries
Standard TTY Libraries
Installation Paths

Figure 37

Step 2 – Add "Using" References

In the Project Explorer, double click on SampleApp.cls to bring up the visual editor for the application form (Figure 38). To view and edit the code associated with the form, right mouse click on the form and select "View Source", or press F9 (Figure 39).

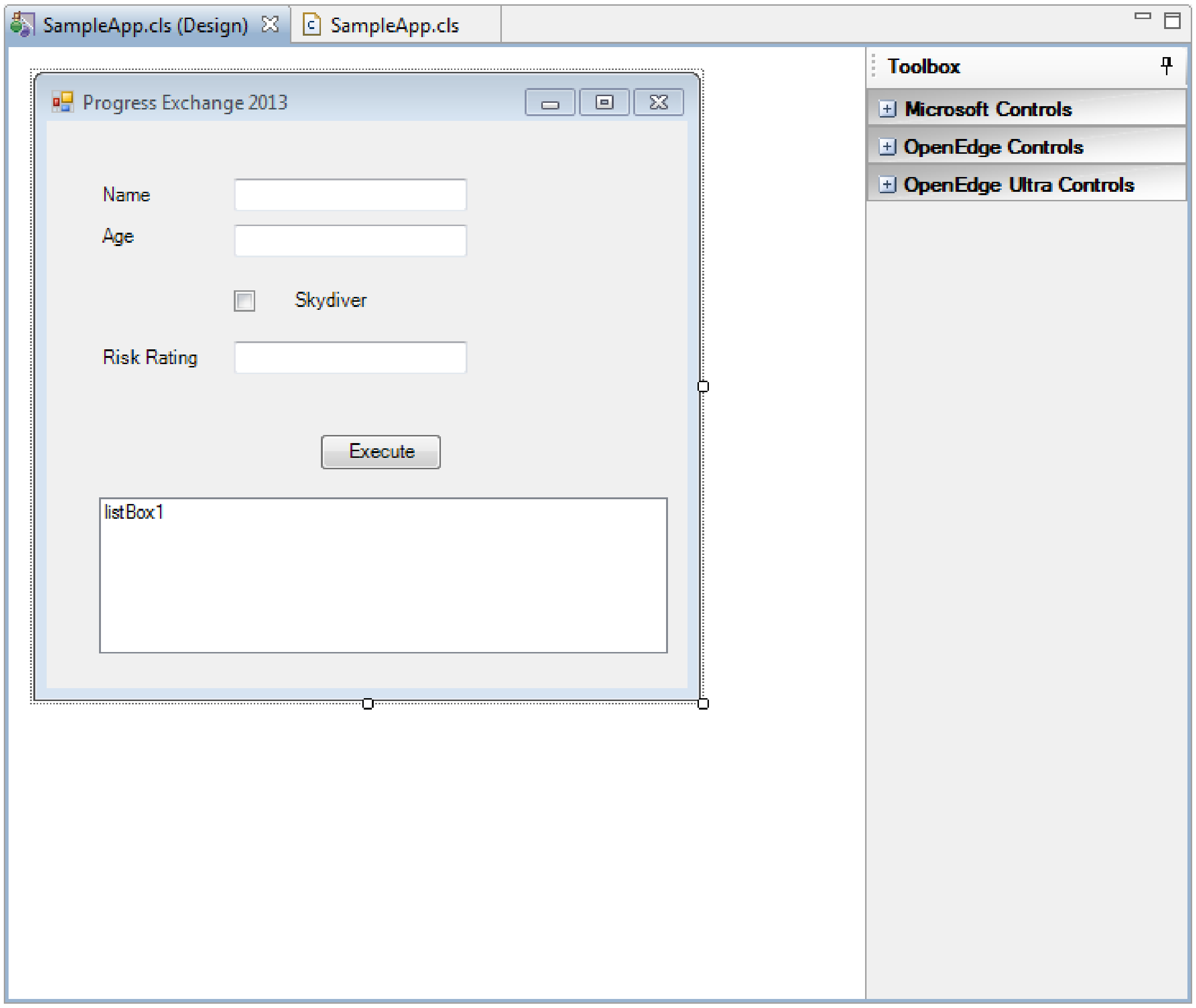
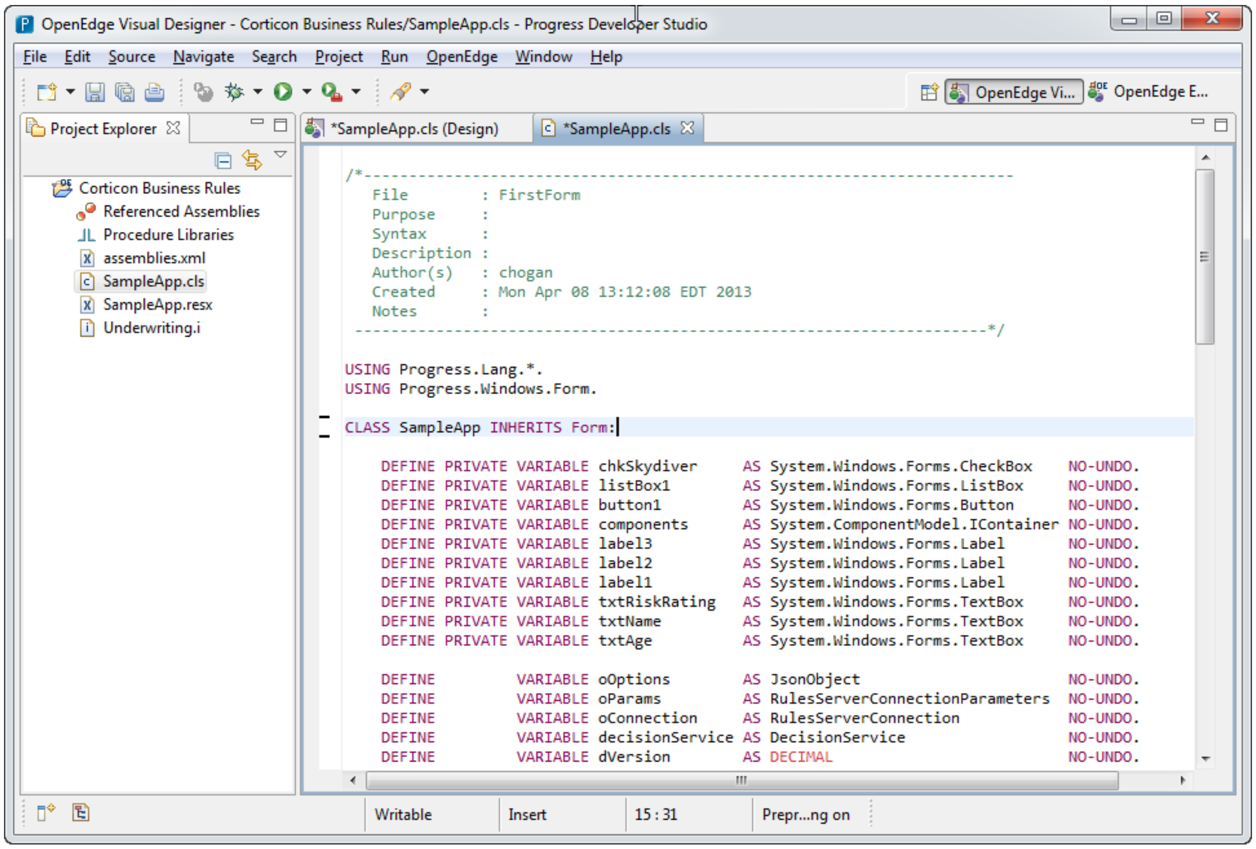


Figure 38

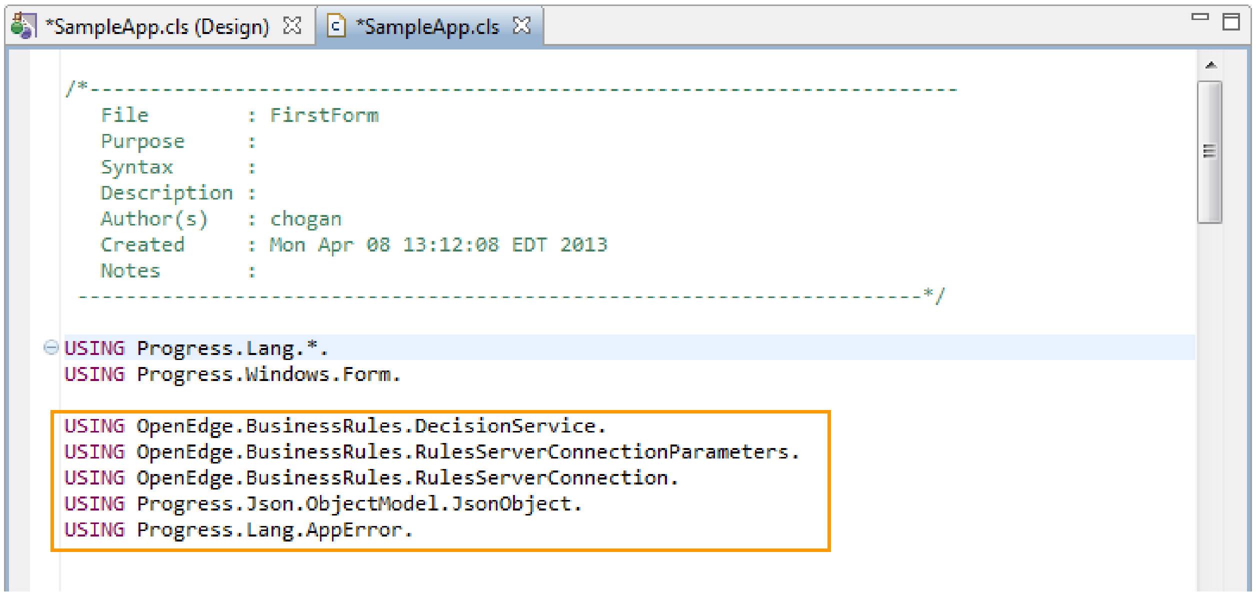


```
/*------------------------------------------------------------------------
   File        : FirstForm
   Purpose     :
   Syntax      :
   Description :
   Author(s)   : chogan
   Created     : Mon Apr 08 13:12:08 EDT 2013
   Notes       :
 ----------------------------------------------------------------------*/

USING Progress.Lang.*.
USING Progress.Windows.Form.

CLASS SampleApp INHERITS Form:

    DEFINE PRIVATE VARIABLE chkSkydiver     AS System.Windows.Forms.CheckBox    NO-UNDO.
    DEFINE PRIVATE VARIABLE listBox1        AS System.Windows.Forms.ListBox     NO-UNDO.
    DEFINE PRIVATE VARIABLE button1         AS System.Windows.Forms.Button      NO-UNDO.
    DEFINE PRIVATE VARIABLE components      AS System.ComponentModel.IContainer NO-UNDO.
    DEFINE PRIVATE VARIABLE label3          AS System.Windows.Forms.Label       NO-UNDO.
    DEFINE PRIVATE VARIABLE label2          AS System.Windows.Forms.Label       NO-UNDO.
    DEFINE PRIVATE VARIABLE label1          AS System.Windows.Forms.Label       NO-UNDO.
    DEFINE PRIVATE VARIABLE txtRiskRating   AS System.Windows.Forms.TextBox     NO-UNDO.
    DEFINE PRIVATE VARIABLE txtName         AS System.Windows.Forms.TextBox     NO-UNDO.
    DEFINE PRIVATE VARIABLE txtAge          AS System.Windows.Forms.TextBox     NO-UNDO.

    DEFINE         VARIABLE oOptions        AS JsonObject                       NO-UNDO.
    DEFINE         VARIABLE oParams         AS RulesServerConnectionParameters  NO-UNDO.
    DEFINE         VARIABLE oConnection     AS RulesServerConnection            NO-UNDO.
    DEFINE         VARIABLE decisionService AS DecisionService                  NO-UNDO.
    DEFINE         VARIABLE dVersion        AS DECIMAL                          NO-UNDO.
```

Figure 39

The first step is to add the USING references as shown in Figure 40. The code snippet is provided so you can copy and paste into your project.

```
/*Begin code snippet*/
USING OpenEdge.BusinessRules.DecisionService.
USING OpenEdge.BusinessRules.RulesServerConnectionParameters.
USING OpenEdge.BusinessRules.RulesServerConnection.
USING Progress.Json.ObjectModel.JsonObject.
USING Progress.Lang.AppError.
/*End code snippet*/
```



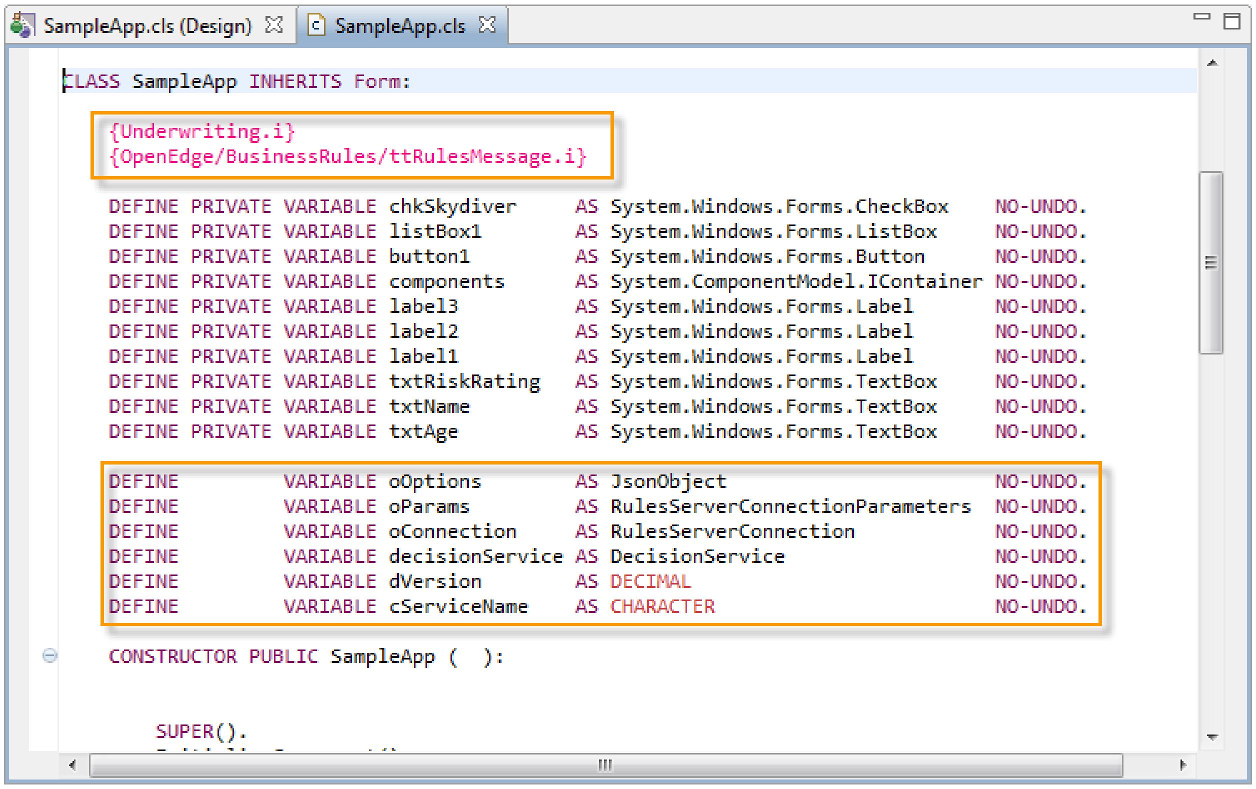
```
/*------------------------------------------------------------------------
   File        : FirstForm
   Purpose     :
   Syntax      :
   Description :
   Author(s)   : chogan
   Created     : Mon Apr 08 13:12:08 EDT 2013
   Notes       :
 ----------------------------------------------------------------------*/

USING Progress.Lang.*.
USING Progress.Windows.Form.

USING OpenEdge.BusinessRules.DecisionService.
USING OpenEdge.BusinessRules.RulesServerConnectionParameters.
USING OpenEdge.BusinessRules.RulesServerConnection.
USING Progress.Json.ObjectModel.JsonObject.
USING Progress.Lang.AppError.
```

Figure 40

Step 3 – Add Variable Definitions

Next you will add the reference to the Include files and Variable definitions as show in Figure 41. The code snippet is provided so you can copy and paste into your project.

```
/*Begin code snippet*/
{Underwriting.i}
{OpenEdge/BusinessRules/ttRulesMessage.i}

DEFINE        VARIABLE oOptions        AS JsonObject                      NO-UNDO.
DEFINE        VARIABLE oParams         AS RulesServerConnectionParameters NO-UNDO.
DEFINE        VARIABLE oConnection     AS RulesServerConnection           NO-UNDO.
DEFINE        VARIABLE decisionService AS DecisionService                 NO-UNDO.
DEFINE        VARIABLE dVersion        AS DECIMAL                         NO-UNDO.
DEFINE        VARIABLE cServiceName    AS CHARACTER                       NO-UNDO.
/*End code snippet*/
```



```
CLASS SampleApp INHERITS Form:

    {Underwriting.i}
    {OpenEdge/BusinessRules/ttRulesMessage.i}

    DEFINE PRIVATE VARIABLE chkSkydiver     AS System.Windows.Forms.CheckBox     NO-UNDO.
    DEFINE PRIVATE VARIABLE listBox1        AS System.Windows.Forms.ListBox      NO-UNDO.
    DEFINE PRIVATE VARIABLE button1         AS System.Windows.Forms.Button       NO-UNDO.
    DEFINE PRIVATE VARIABLE components      AS System.ComponentModel.IContainer  NO-UNDO.
    DEFINE PRIVATE VARIABLE label3          AS System.Windows.Forms.Label        NO-UNDO.
    DEFINE PRIVATE VARIABLE label2          AS System.Windows.Forms.Label        NO-UNDO.
    DEFINE PRIVATE VARIABLE label1          AS System.Windows.Forms.Label        NO-UNDO.
    DEFINE PRIVATE VARIABLE txtRiskRating   AS System.Windows.Forms.TextBox      NO-UNDO.
    DEFINE PRIVATE VARIABLE txtName         AS System.Windows.Forms.TextBox      NO-UNDO.
    DEFINE PRIVATE VARIABLE txtAge          AS System.Windows.Forms.TextBox      NO-UNDO.

    DEFINE          VARIABLE oOptions        AS JsonObject                        NO-UNDO.
    DEFINE          VARIABLE oParams         AS RulesServerConnectionParameters   NO-UNDO.
    DEFINE          VARIABLE oConnection     AS RulesServerConnection             NO-UNDO.
    DEFINE          VARIABLE decisionService AS DecisionService                   NO-UNDO.
    DEFINE          VARIABLE dVersion        AS DECIMAL                           NO-UNDO.
    DEFINE          VARIABLE cServiceName    AS CHARACTER                         NO-UNDO.

    CONSTRUCTOR PUBLIC SampleApp (  ):


        SUPER().
```

Figure 41

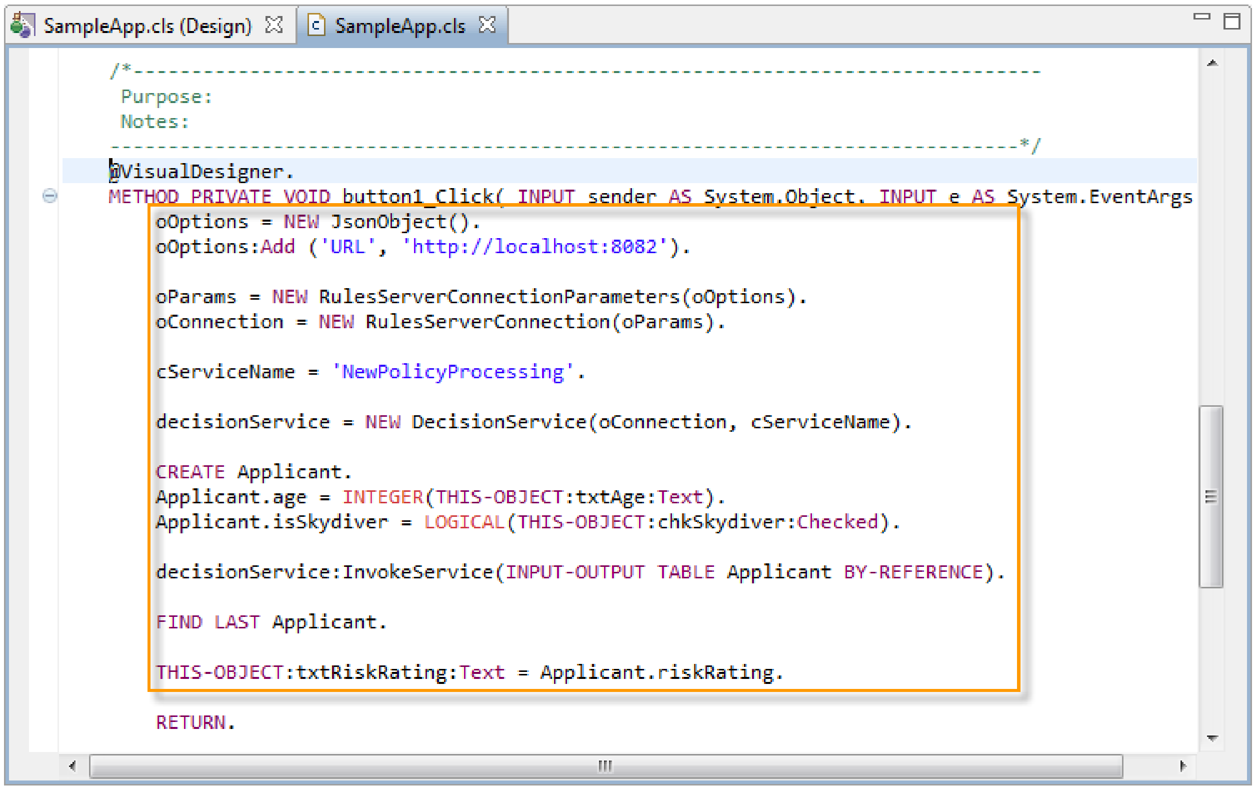
Step 4 – Add Decision Service Execution Code

In this step you will add the code to execute the decision service when the “Execute” button is clicked. The code will create a connection to the Corticon Server. It will create a record in the temp-table based on the values entered in the user interface. It will execute the decision service and the update the form with the results of the decision (Figure 42).

The code snippet is provided so you can copy and paste into your project.

```
/*Begin code snippet*/
oOptions = NEW JsonObject().
oOptions:Add ('URL', 'http://localhost:8082').
oParams = NEW RulesServerConnectionParameters(oOptions).
oConnection = NEW RulesServerConnection(oParams).

cServiceName = 'NewPolicyProcessing'.
decisionService = NEW DecisionService(oConnection, cServiceName).

EMPTY TEMP-TABLE Applicant.
CREATE Applicant.
Applicant.age = INTEGER(THIS-OBJECT:txtAge:Text).
Applicant.isSkydiver = LOGICAL(THIS-OBJECT:chkSkydiver:Checked).

decisionService:InvokeService(INPUT-OUTPUT TABLE Applicant BY-REFERENCE).
```

```
FIND LAST Applicant.
THIS-OBJECT:txtRiskRating:Text = Applicant.riskRating.

/* get the messages */
decisionService:GetMessages(OUTPUT table RulesMessage).

listBox1:Items:Clear().
FOR EACH RulesMessage:
      listBox1:Items:Add(RulesMessage.MessageText).
END.
/*End code snippet*/
```



```
    /*------------------------------------------------------------------------------
     Purpose:
     Notes:
    ------------------------------------------------------------------------------*/
    @VisualDesigner.
    METHOD PRIVATE VOID button1_Click( INPUT sender AS System.Object, INPUT e AS System.EventArgs
        oOptions = NEW JsonObject().
        oOptions:Add ('URL', 'http://localhost:8082').

        oParams = NEW RulesServerConnectionParameters(oOptions).
        oConnection = NEW RulesServerConnection(oParams).

        cServiceName = 'NewPolicyProcessing'.

        decisionService = NEW DecisionService(oConnection, cServiceName).

        CREATE Applicant.
        Applicant.age = INTEGER(THIS-OBJECT:txtAge:Text).
        Applicant.isSkydiver = LOGICAL(THIS-OBJECT:chkSkydiver:Checked).

        decisionService:InvokeService(INPUT-OUTPUT TABLE Applicant BY-REFERENCE).

        FIND LAST Applicant.

        THIS-OBJECT:txtRiskRating:Text = Applicant.riskRating.

        RETURN.
```

Figure 42

Step 5 – Run OpenEdge Application

Now we will run the application. From Developer Studio, click the Run button and select "Run As\Progress OpenEdge Application" (Figure 43). This will execute your app at the form will be shown (Figure 44).

Input values for Name, Age, Skydiver and click the Execute button. Based up the values you provided you will see output similar to Figure 45. You can change the values and Execute again to get various answers from the decision.

PROGRESS

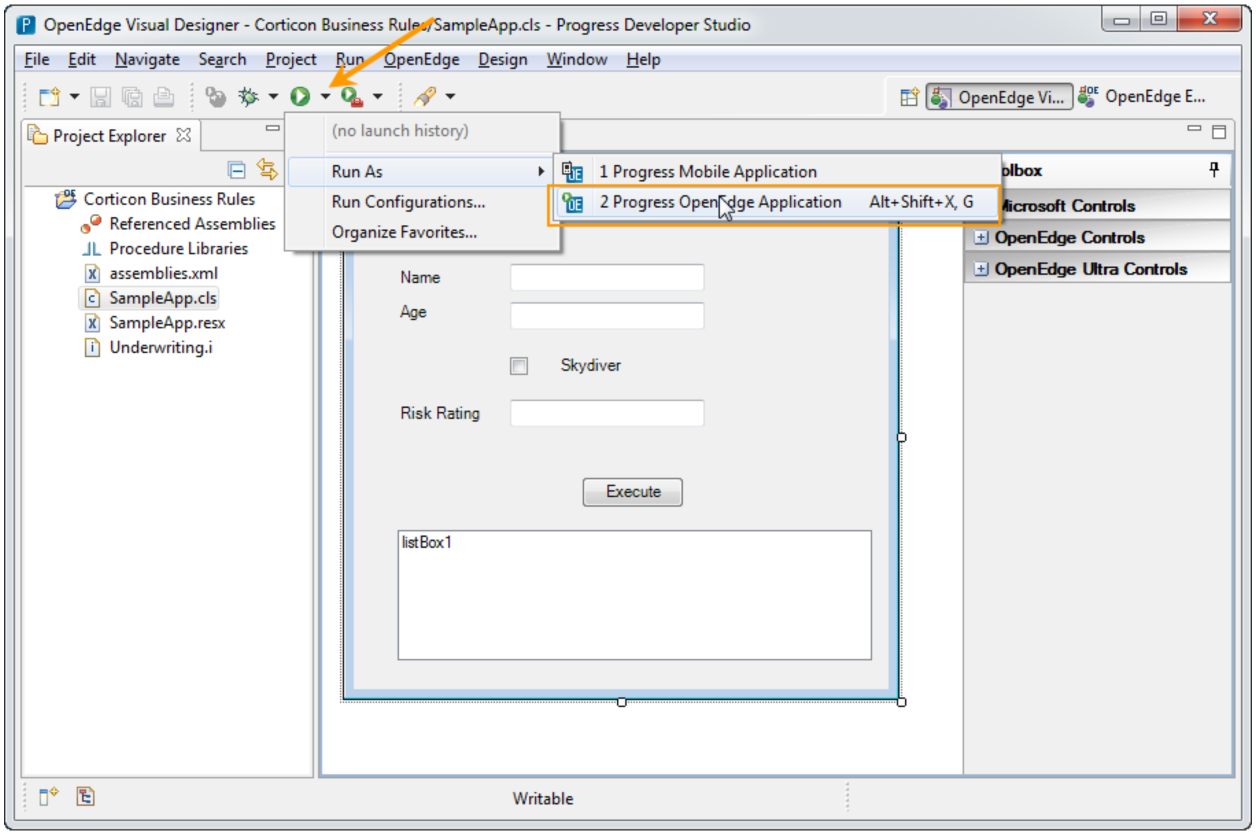


Figure 43

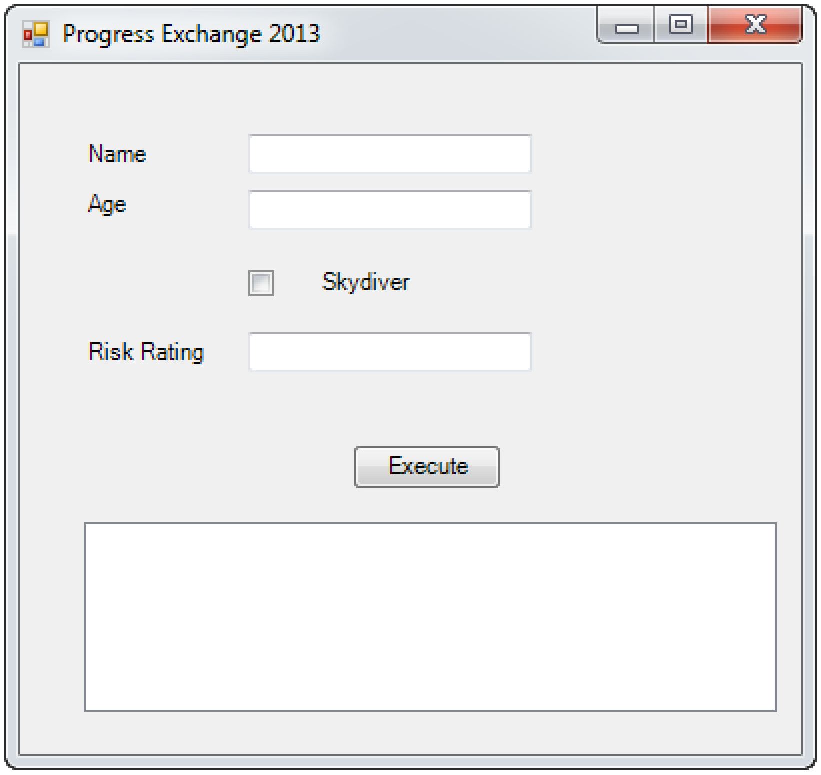
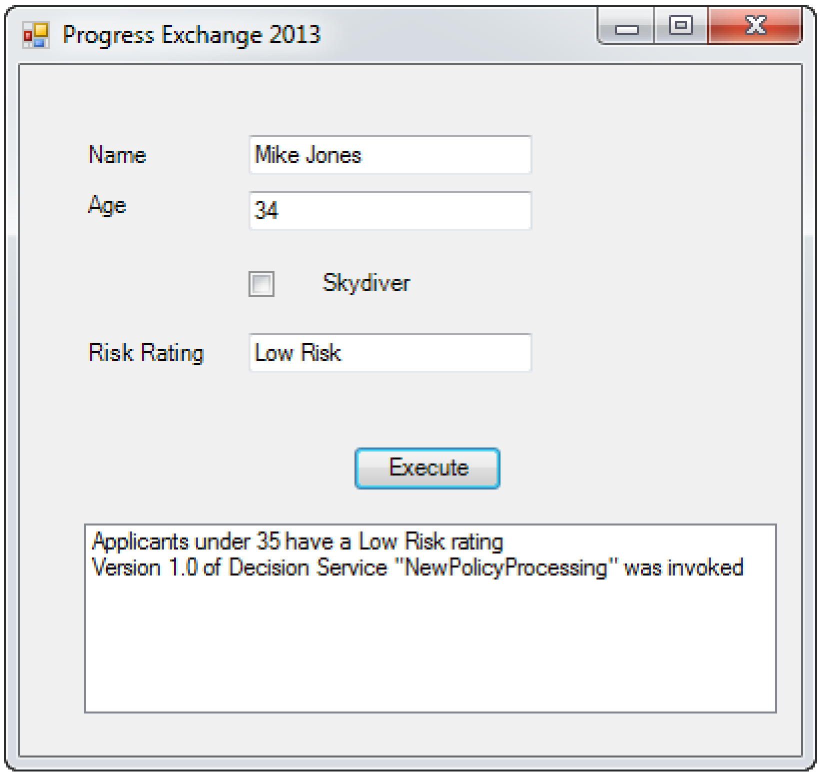


Figure 44



Name: Mike Jones
Age: 34
[ ] Skydiver
Risk Rating: Low Risk

```
Applicants under 35 have a Low Risk rating
Version 1.0 of Decision Service "NewPolicyProcessing" was invoked
```

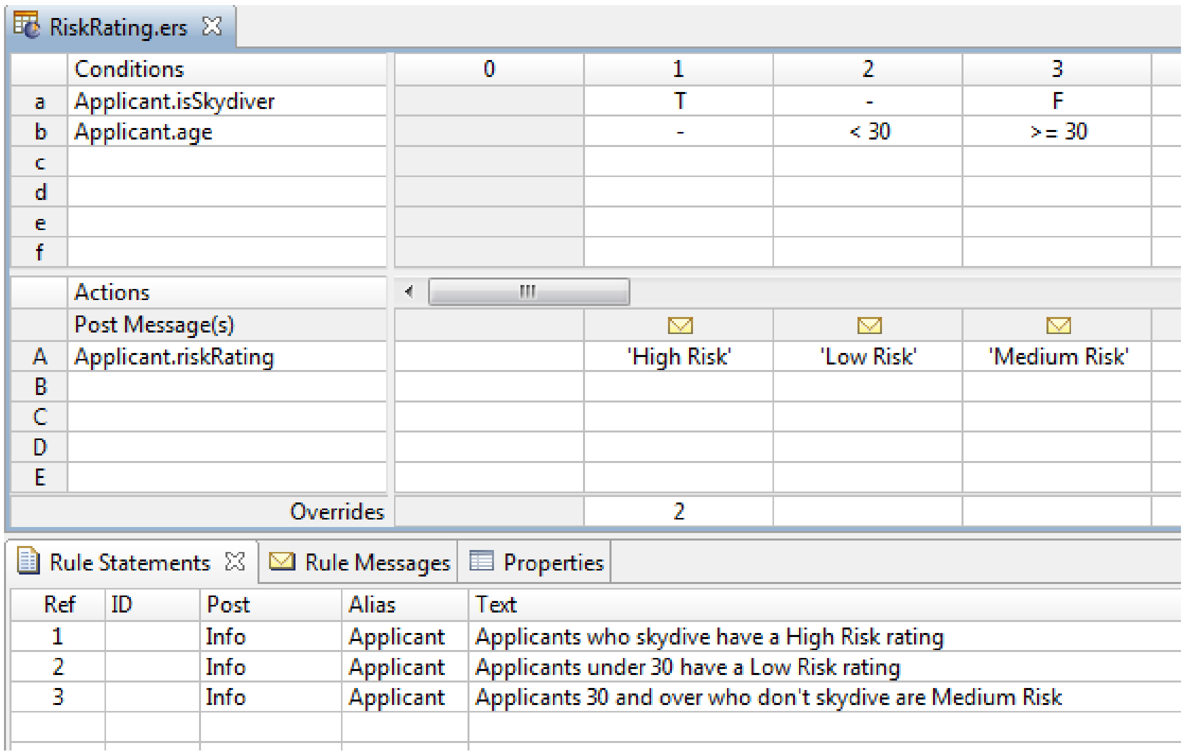
Figure 45

Exercise 3 – Update Deployed Decision Service

In this exercise, while leaving the OpenEdge application running, you make a change the decision service you previously modeled. Once you re-deploy, you will see that for the same input values into the OpenEdge application you will get a different answer.

The first step is to launch Corticon Studio, if it is not already running. Update the rules to use 30 as the threshold age. Make sure to update the Rule Statements as well (Figure 46). Next, right mouse click on the Project and launch the Publish wizard again. Make sure to check the “Republish” option to overwrite an existing version (Figure 47).

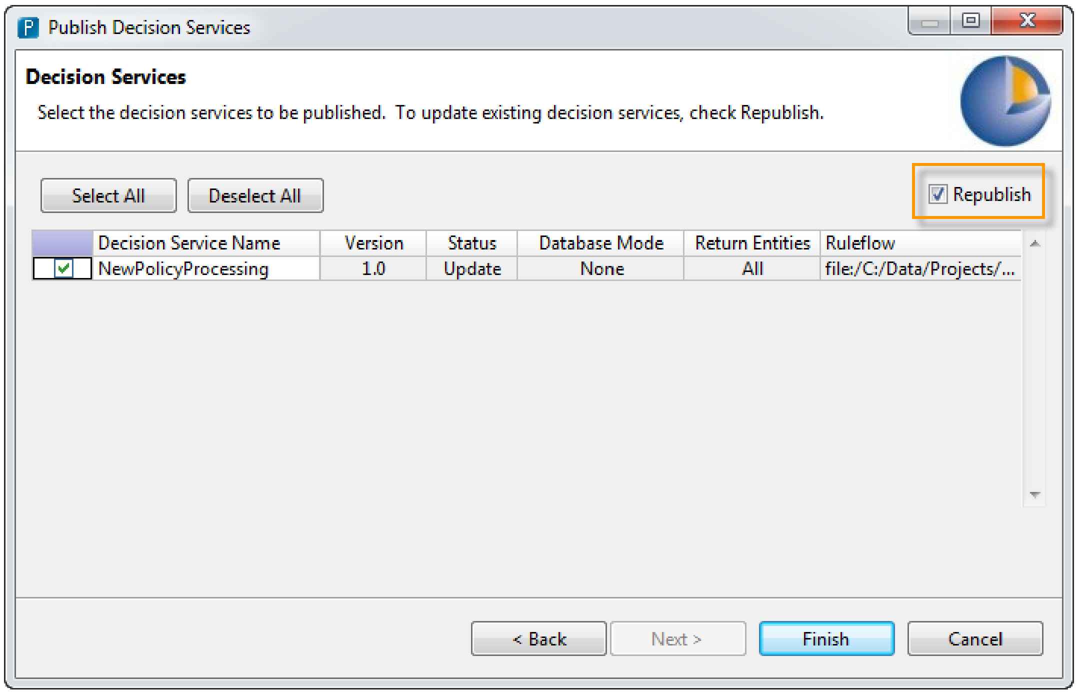
Finally, click the execute button on your OpenEdge app and see the new results (Figure 48).



	Conditions	0	1	2	3
a	Applicant.isSkydiver		T	-	F
b	Applicant.age		-	< 30	>= 30
c					
d					
e					
f					
	Actions				
	Post Message(s)		✉	✉	✉
A	Applicant.riskRating		'High Risk'	'Low Risk'	'Medium Risk'
B					
C					
D					
E					
	Overrides		2		

Ref	ID	Post	Alias	Text
1		Info	Applicant	Applicants who skydive have a High Risk rating
2		Info	Applicant	Applicants under 30 have a Low Risk rating
3		Info	Applicant	Applicants 30 and over who don't skydive are Medium Risk

Figure 46

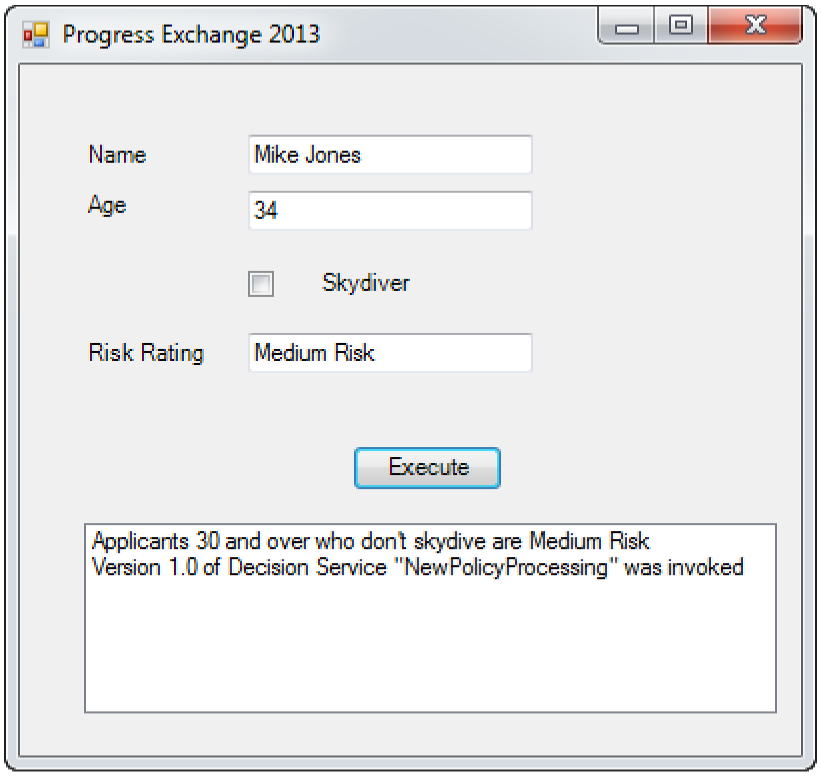

Publish Decision Services

Decision Services

Select the decision services to be published. To update existing decision services, check Republish.

	Decision Service Name	Version	Status	Database Mode	Return Entities	Ruleflow
✓	NewPolicyProcessing	1.0	Update	None	All	file:/C:/Data/Projects/...

Figure 47


Progress Exchange 2013

Name	Mike Jones
Age	34
Skydiver	☐
Risk Rating	Medium Risk

```
Applicants 30 and over who dont skydive are Medium Risk
Version 1.0 of Decision Service "NewPolicyProcessing" was invoked
```

Figure 48

Bonus activities

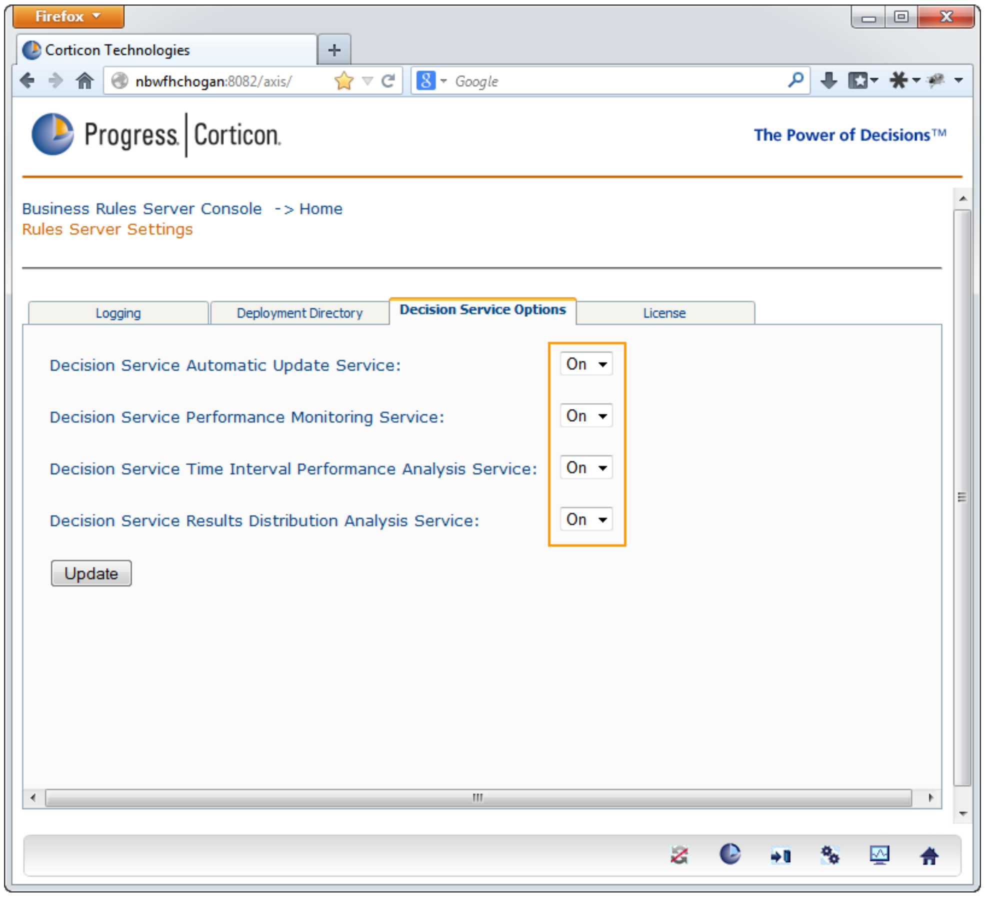
If you complete the basic lab exercises before the allotted time, you may wish to look at some addition examples of functionality.

Exercise 1 – Enhance the Vocabulary

In developer Studio for OE, update Underwriting.i to add new fields the temp-table. You can also create an additional temp-table and define a relationship with a Dataset. Export the .brvd file and re-import into Corticon Studio. You should see the updated vocabulary. If you used a Dataset, you will see an association between the Vocabulary entities

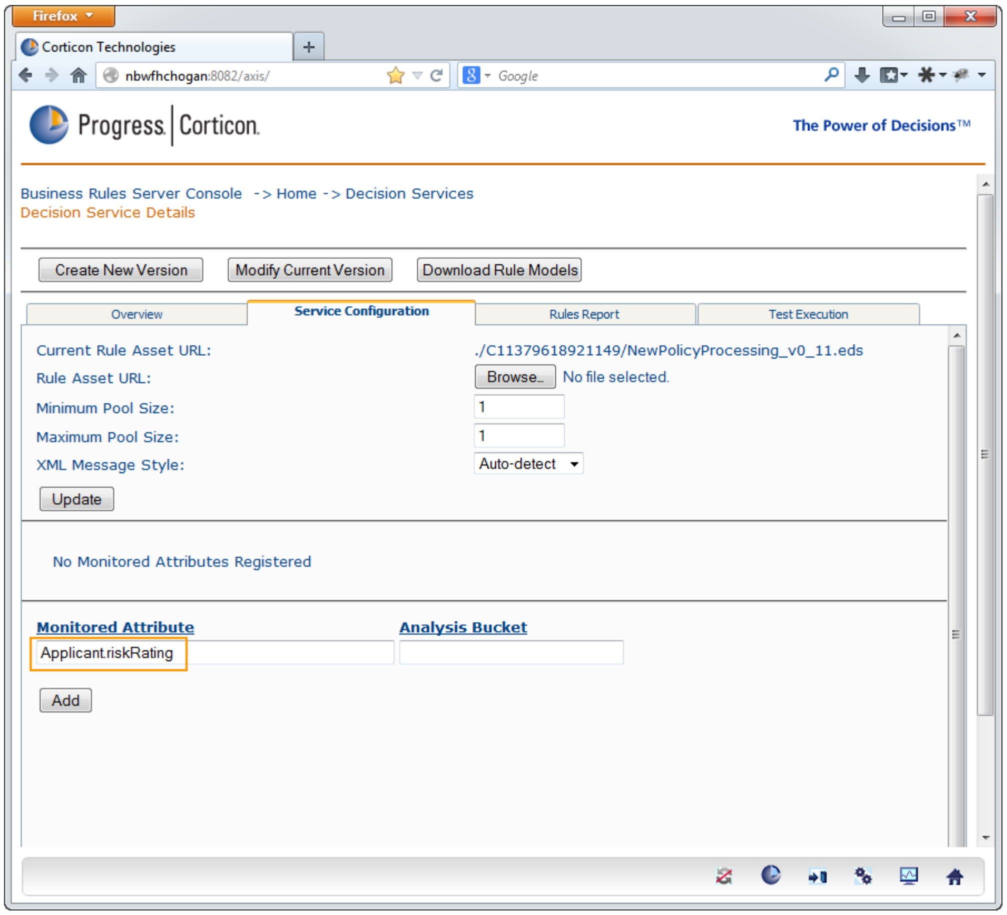
Exercise 1 – Configure Server Console for Monitoring

Click “Configure Rules Server” icon from toolbar. Ensure all options are “Yes” under Decision Service Options (Figure 49). Go to list of Decision Services and select NewPolicyProcessing. Drill down to Service Configuration and add Applicant.riskRating as a Monitored Attribute (Figure 50). For all future executions of this service, Corticon will capture statistics for the attribute that you can review (figure 51, 52).



Decision Service Automatic Update Service:	On
Decision Service Performance Monitoring Service:	On
Decision Service Time Interval Performance Analysis Service:	On
Decision Service Results Distribution Analysis Service:	On

Figure 49



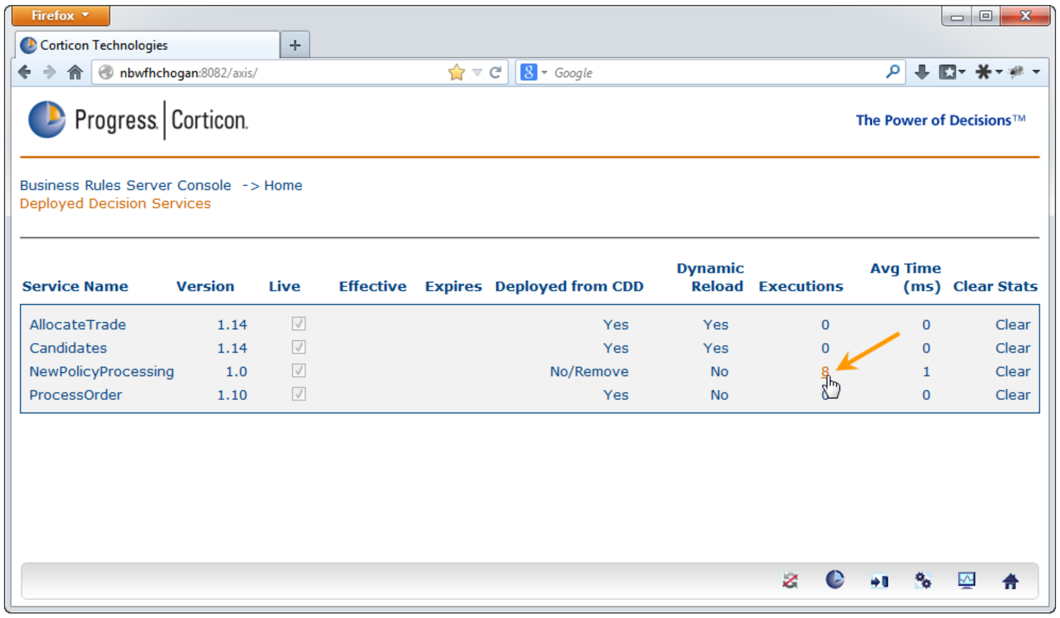
Business Rules Server Console -> Home -> Decision Services
Decision Service Details

Current Rule Asset URL:	./C11379618921149/NewPolicyProcessing_v0_11.eds
Rule Asset URL:	Browse... No file selected.
Minimum Pool Size:	1
Maximum Pool Size:	1
XML Message Style:	Auto-detect

No Monitored Attributes Registered

Monitored Attribute	Analysis Bucket
Applicant.riskRating	

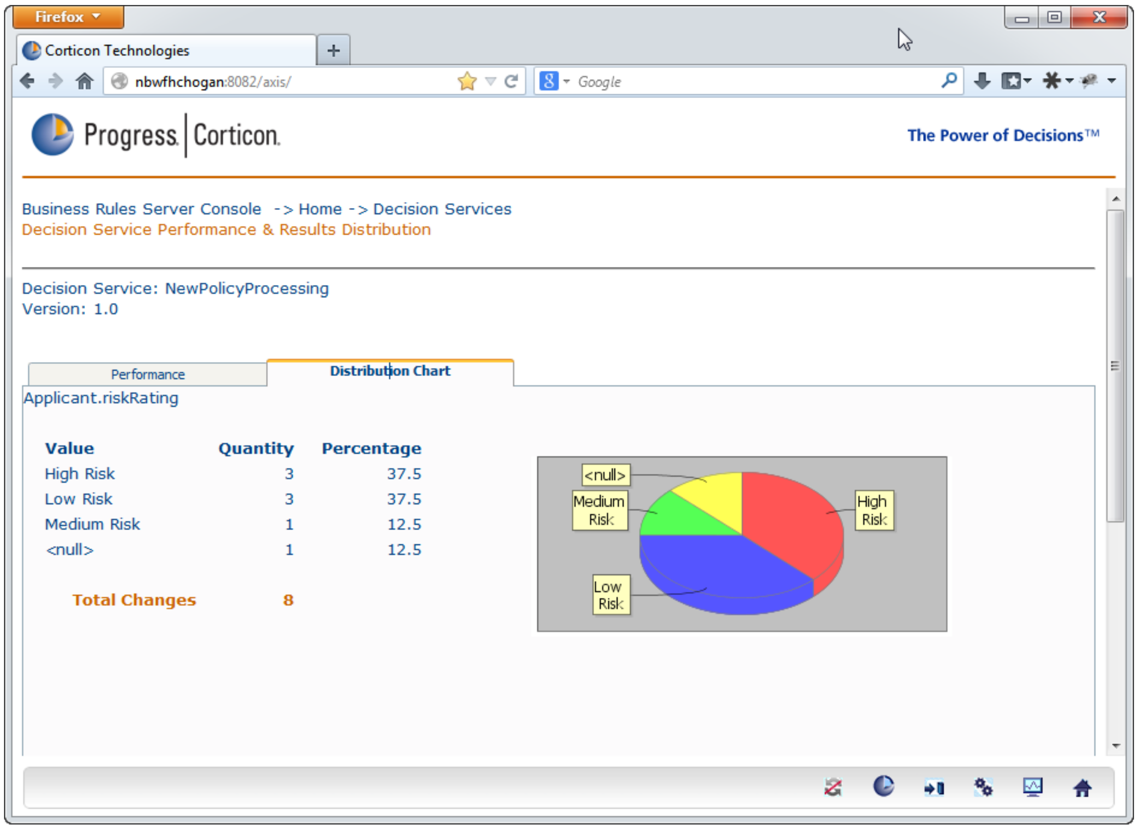
Figure 50



Business Rules Server Console -> Home
Deployed Decision Services

Service Name	Version	Live	Effective	Expires	Deployed from CDD	Dynamic Reload	Executions	Avg Time (ms)	Clear Stats
AllocateTrade	1.14	✓			Yes	Yes	0	0	Clear
Candidates	1.14	✓			Yes	Yes	0	0	Clear
NewPolicyProcessing	1.0	✓			No/Remove	No	8	1	Clear
ProcessOrder	1.10	✓			Yes	No	0	0	Clear

Figure 51



Business Rules Server Console -> Home -> Decision Services
Decision Service Performance & Results Distribution

Decision Service: NewPolicyProcessing
Version: 1.0

Applicant.riskRating

Value	Quantity	Percentage
High Risk	3	37.5
Low Risk	3	37.5
Medium Risk	1	12.5
<null>	1	12.5
Total Changes	8	

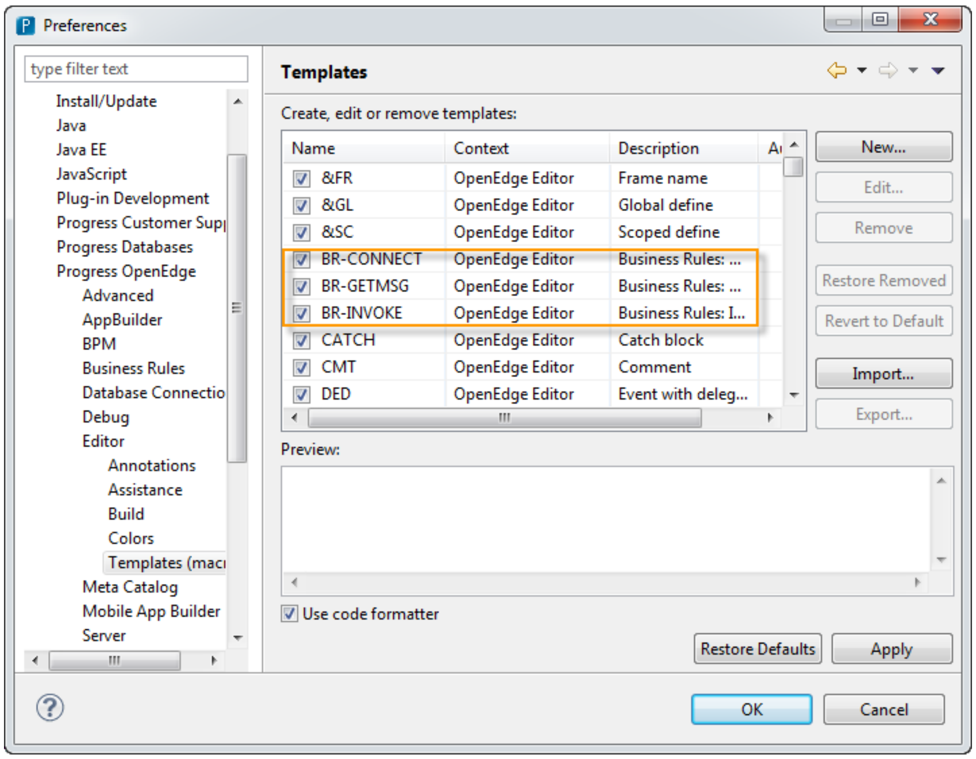
Figure 52

Exercise 2 – Use Code Macros

There are 3 default macros provided to assist with code insertion.

BR-CONNECT
BR-GETMSG
BR-INVOKE

The templates for these are editable and found in System Preferences (Figure 53). You can experiment using these macros to automatically insert the code that you copy/pasted in the previous exercise.



Preferences

type filter text

- Install/Update
- Java
- Java EE
- JavaScript
- Plug-in Development
- Progress Customer Sup
- Progress Databases
- Progress OpenEdge
  - Advanced
  - AppBuilder
  - BPM
  - Business Rules
  - Database Connectio
  - Debug
  - Editor
    - Annotations
    - Assistance
    - Build
    - Colors
    - Templates (mac
  - Meta Catalog
  - Mobile App Builder
  - Server

Templates

Create, edit or remove templates:

	Name	Context	Description
☑	&FR	OpenEdge Editor	Frame name
☑	&GL	OpenEdge Editor	Global define
☑	&SC	OpenEdge Editor	Scoped define
☑	BR-CONNECT	OpenEdge Editor	Business Rules: ...
☑	BR-GETMSG	OpenEdge Editor	Business Rules: ...
☑	BR-INVOKE	OpenEdge Editor	Business Rules: I...
☑	CATCH	OpenEdge Editor	Catch block
☑	CMT	OpenEdge Editor	Comment
☑	DED	OpenEdge Editor	Event with deleg...

Preview:

☑ Use code formatter

Figure 53

Exercise 2 – Externalize to Standalone.p file

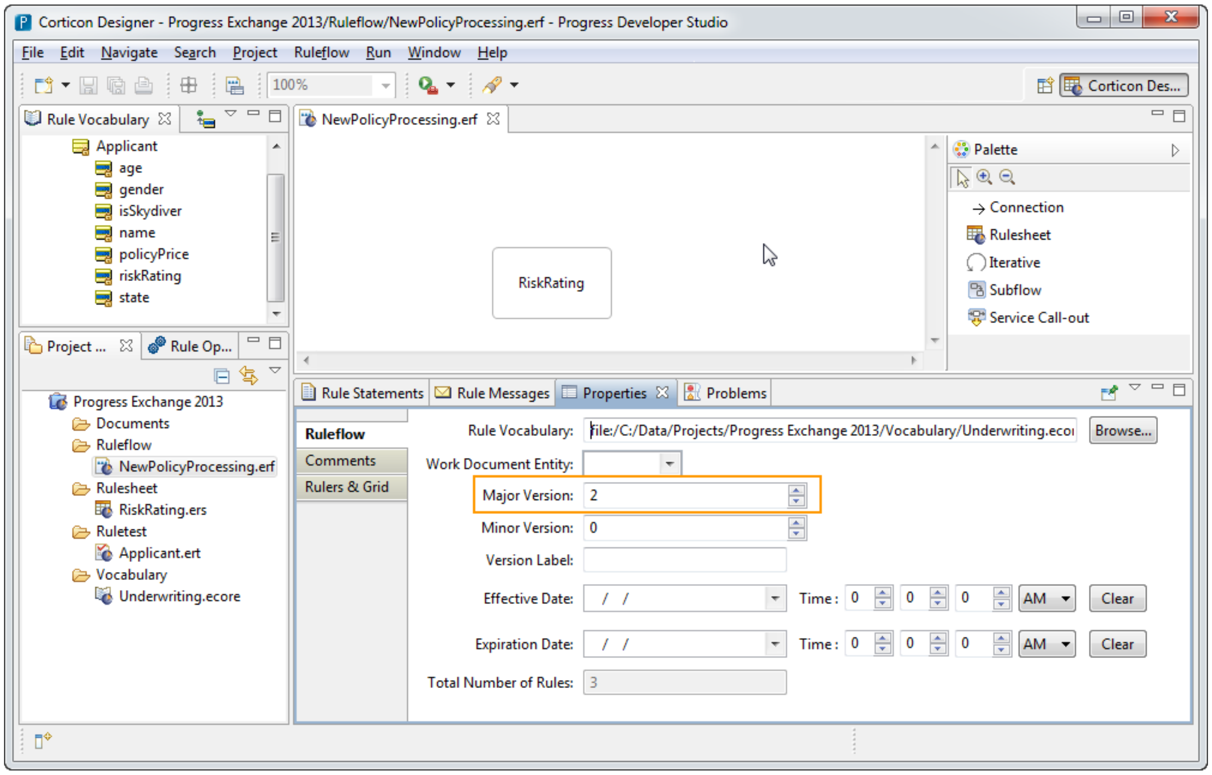
Instead of putting all of the decision service execution code in the Form definition, you can place it in a standalone.p file and then execute that .p file from the Form code. This could be helpful if you want to re-use this code from many locations.

Exercise 3 – Add new Rulesheets to Ruleflow

The current Ruleflow has only one Rulesheet. Real world decisions have many Rulesheets. In this step, create a new Rulesheet called PolicyPricing. Add 3 new rules that set a policy price amount for the three different risk levels High Risk, Low Risk, Medium Risk. You can then add this Rulesheet to your existing NewPolicyProcessing Ruleflow and link the steps together. Now your decision service not only calculates a risk rating, but it also determines the policy price as well.

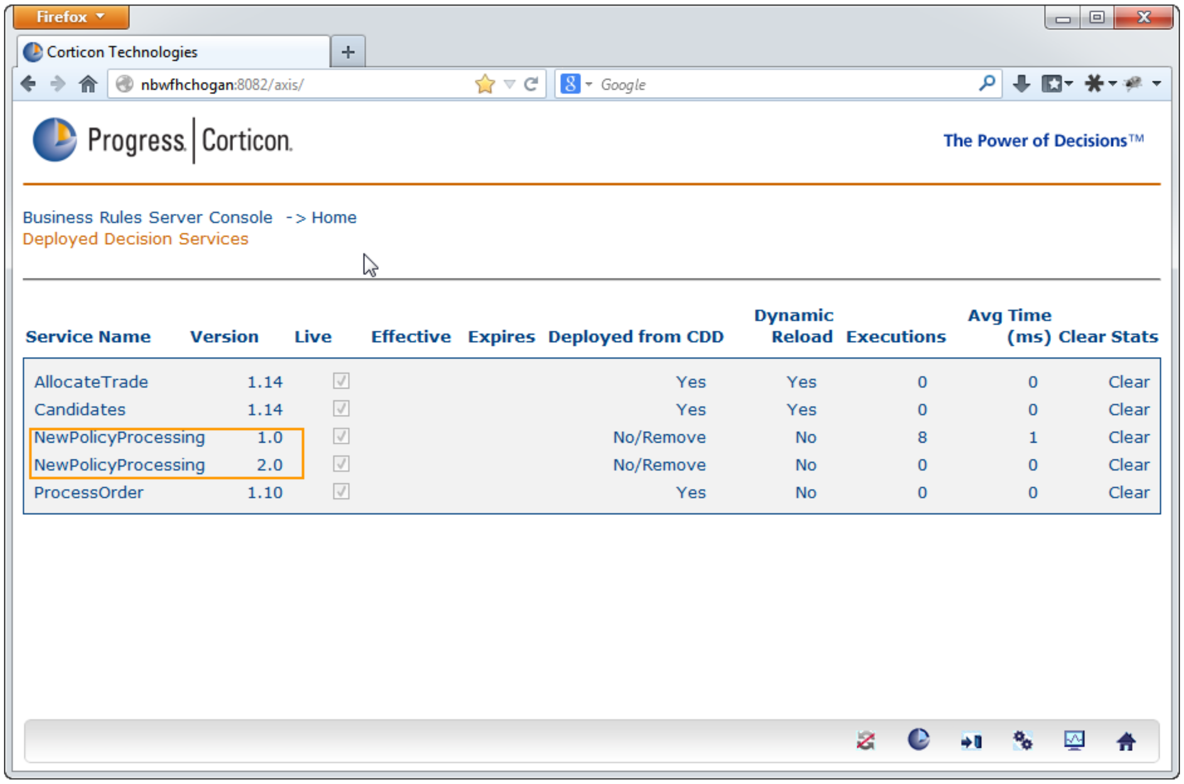
Exercise 3 – Deploy Multiple Versions

Corticon supports deployment of multiple versions of the same decision service. To create new version, navigate to the Properties panel for the Ruleflow (Figure 52). Update the Major Version to 2 and save the Ruleflow. Now when you publish the Ruleflow you will see to versions deployed on the Corticon Server (Figure 55).



Rule Vocabulary: file:/C:/Data/Projects/Progress Exchange 2013/Vocabulary/Underwriting.ecor
Work Document Entity:
Major Version: 2
Minor Version: 0
Version Label:
Effective Date: / / Time : 0 0 0 AM
Expiration Date: / / Time : 0 0 0 AM
Total Number of Rules: 3

Figure 54



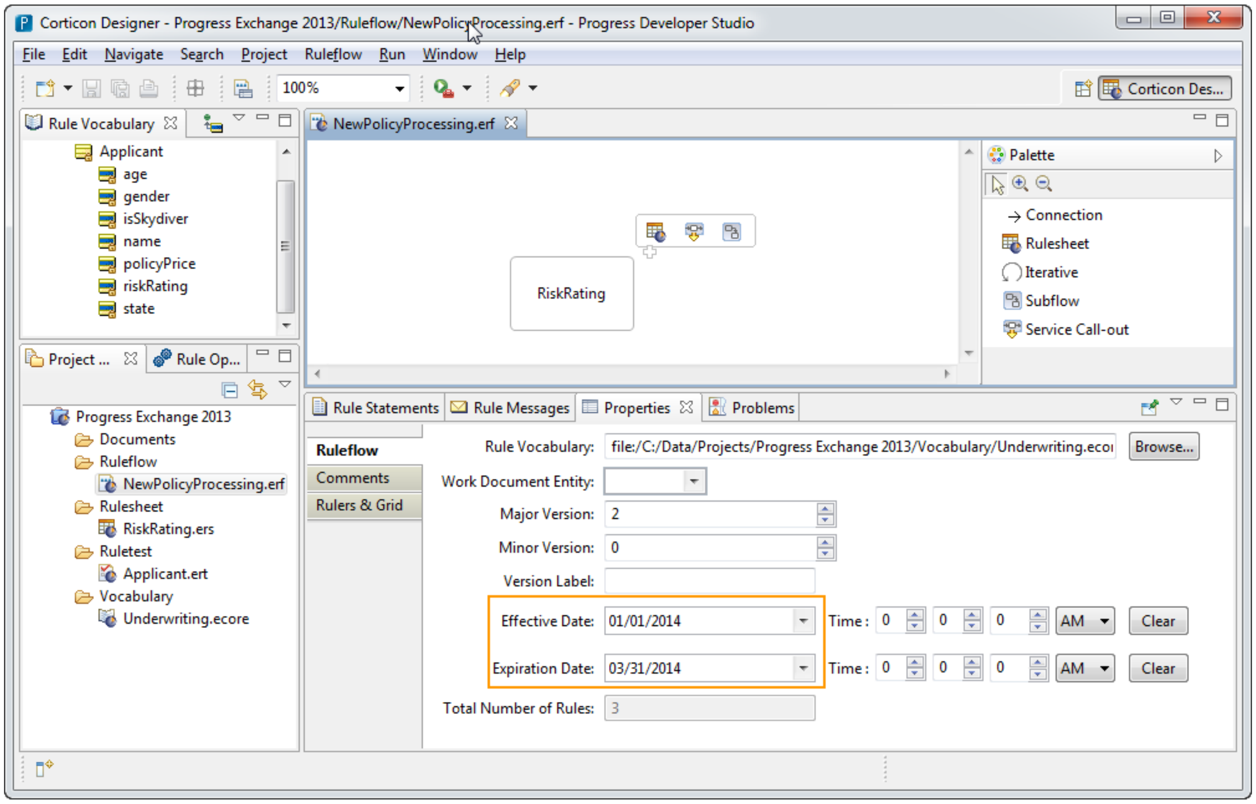
Business Rules Server Console -> Home
Deployed Decision Services

Service Name	Version	Live	Effective	Expires	Deployed from CDD	Dynamic Reload	Executions	Avg Time (ms)	Clear Stats
AllocateTrade	1.14	✓			Yes	Yes	0	0	Clear
Candidates	1.14	✓			Yes	Yes	0	0	Clear
NewPolicyProcessing	1.0	✓			No/Remove	No	8	1	Clear
NewPolicyProcessing	2.0	✓			No/Remove	No	0	0	Clear
ProcessOrder	1.10	✓			Yes	No	0	0	Clear

Figure 55

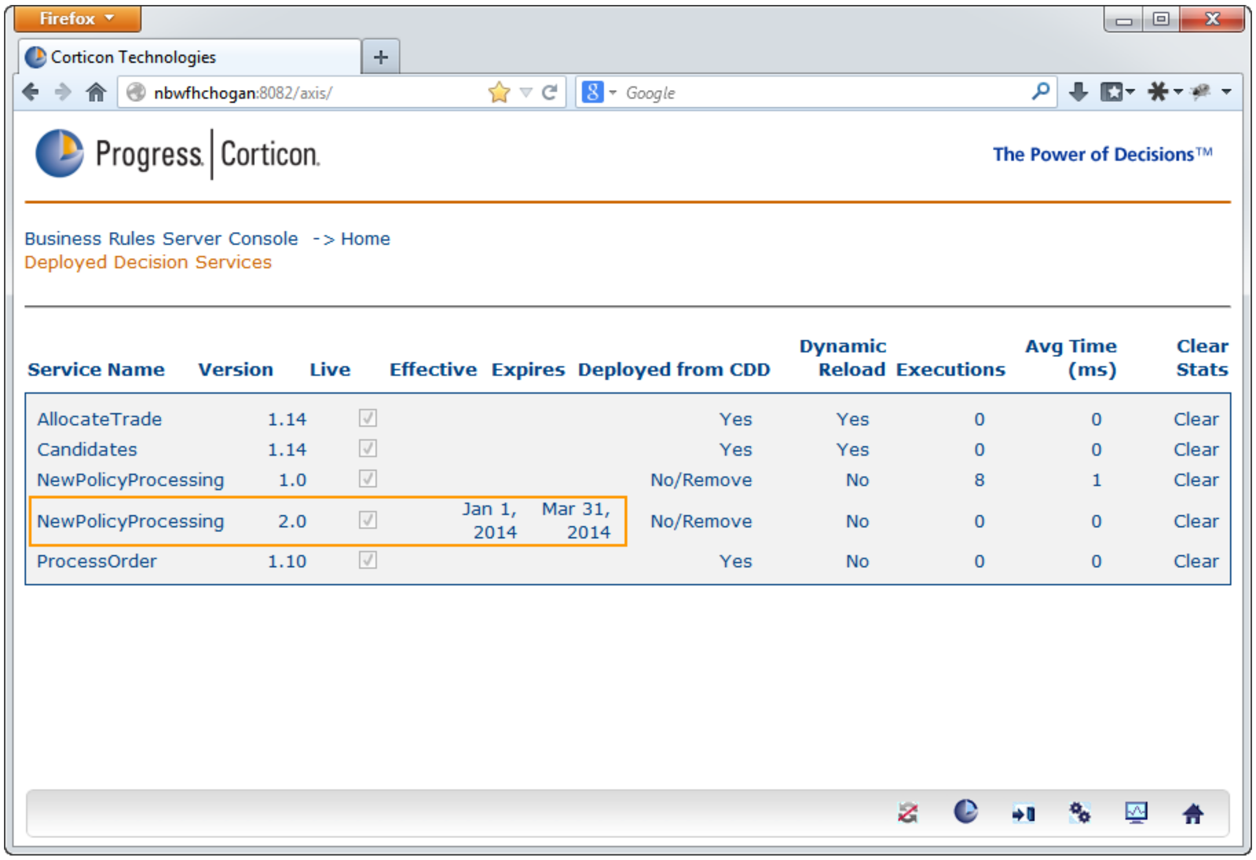
Exercise 3 – Deploy with Effective Dates

Corticon also support effective dates for versions. This means you can specify what date ranges that a certain version is applicable. Navigate to the properties for the Ruleflow and specify a start and stop date/time (Figure 56). Re-publish the decision service, and you will see that there are now effective dates for the decision service (Figure 57).



Rule Vocabulary: file:/C:/Data/Projects/Progress Exchange 2013/Vocabulary/Underwriting.ecore
Work Document Entity:
Major Version: 2
Minor Version: 0
Version Label:
Effective Date: 01/01/2014 Time : 0 0 0 AM
Expiration Date: 03/31/2014 Time : 0 0 0 AM
Total Number of Rules: 3

Figure 56



Business Rules Server Console -> Home
Deployed Decision Services

Service Name	Version	Live	Effective	Expires	Deployed from CDD	Dynamic Reload	Executions	Avg Time (ms)	Clear Stats
AllocateTrade	1.14	✓			Yes	Yes	0	0	Clear
Candidates	1.14	✓			Yes	Yes	0	0	Clear
NewPolicyProcessing	1.0	✓			No/Remove	No	8	1	Clear
NewPolicyProcessing	2.0	✓	Jan 1, 2014	Mar 31, 2014	No/Remove	No	0	0	Clear
ProcessOrder	1.10	✓			Yes	No	0	0	Clear

Figure 57