Agility through Business Rules Management

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.

Use Case

During this workshop you'll be asked to create a decision service and integrate this service with your backend OpenEdge application. You are working for an IT department of a large Insurance company. This company is modernizing its IT infrastructure by externalizing as much as possible all business logic in a services layer. This promotes speed of development, ease of maintenance, reuse of the same services across other applications (such as the web site), traceability and quality of the logic. The decision service we are going to work on is a service that assesses the risk of someone applying for a new insurance policy. Usually "risk" translates to higher or lower premiums to be paid for insurance coverage. Later we'll extend the service to encompass other elements.

Setting up your environment - Getting started with Progress Arcade

We will use a Progress Arcade virtual image in the Cloud with all Progress software required already installed for you to complete the workshop exercises.

WHAT IS PROGRESS ARCADE?

Progress® Arcade™ is a web portal where you can deploy and manage Progress applications in a Cloud-based environment. You can use Arcade to migrate existing Progress applications to a public Cloud environment, run and test the application in the Cloud, and then deploy the application. Deployment includes the ability to host demos or to publish applications that are available to subscribers.

Progress Arcade is also a site where you can interact with other Arcade users, and find out about services offered by various Progress Technology Partners.

HOW DO I ACCESS THE FEATURES OF PROGRESS ARCADE?

The features of Progress Arcade are grouped into functional areas, represented by six panels on the Arcade home page. The panels include:

- **Stage & Test** – Gain experience in running your application in the public cloud.
- **Deploy** - Deploy your application into full production for customer use directly from the public cloud (not currently available).
- **Demo** – Publish your application for demonstration purposes using the public cloud.
- **Expo** – Search for information on complementary products and services offered by the Progress community.
- **Community Café** – Access Progress product resources and network with others in the Progress community.
- **Product Showroom** – Learn about the features and benefits of Progress products. Specify what technology you are interested in, and within minutes get a public cloud-based machine dedicated to your exclusive use.

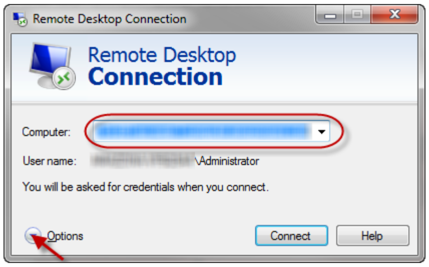
LAUNCHING YOUR OE CORTICON DEMO MACHINE ON PROGRESS ARCADE

To launch your OpenEdge Corticon Demo machine:

1. Several machines have been started for this workshop. Each of you will be given a unique DNS address by the workshop host (see PowerPoint slide on how to do this) at this point which points to a virtual windows server running on Progress Arcade.

2. On your laptop, choose **Start| Accessories| Remote Desktop Connection** and paste your unique DNS into the **Computer** field. Click the **Connect** button.

   You may need to click the button left of Options to change (erase) the domain name.

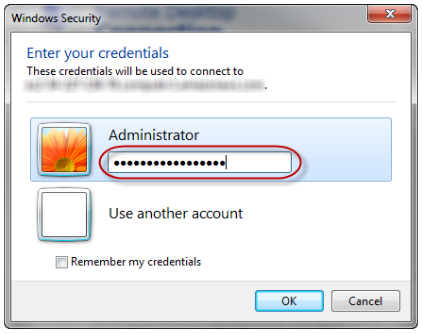


Computer: [redacted]

User name: [redacted]\Administrator

You will be asked for credentials when you connect.

3. Enter **Administrator** as the Username, and **ApjPug2015** as the Password.



Enter your credentials

These credentials will be used to connect to [redacted].

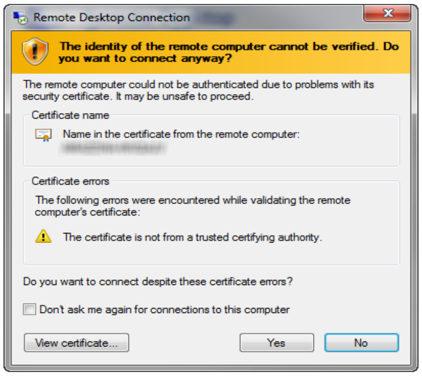
Administrator

•••••••••••••••••

Use another account

□ Remember my credentials

4. Click the **Yes** button to say you trust this remote connection.



The identity of the remote computer cannot be verified. Do you want to connect anyway?

The remote computer could not be authenticated due to problems with its security certificate. It may be unsafe to proceed.

Certificate name

Name in the certificate from the remote computer:
[redacted]

Certificate errors

The following errors were encountered while validating the remote computer's certificate:

⚠ The certificate is not from a trusted certifying authority.

Do you want to connect despite these certificate errors?

□ Don't ask me again for connections to this computer

5. You should now be connected to the demonstration machine running in Progress Arcade.

<u>Note:</u> The machine instance <u>may</u> have a different keyboard setting than what you are using. If your laptop has a different keyboard, you may need to set this up yourself on the machine instance through **Control Panel| Region** and Language settings.

Exercise 1 – Model Corticon Decision

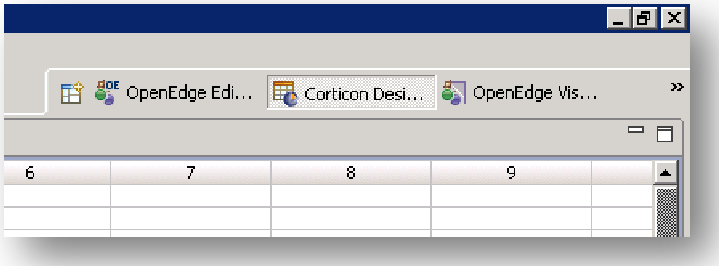


Figure 1: Corticon Designer Perspective

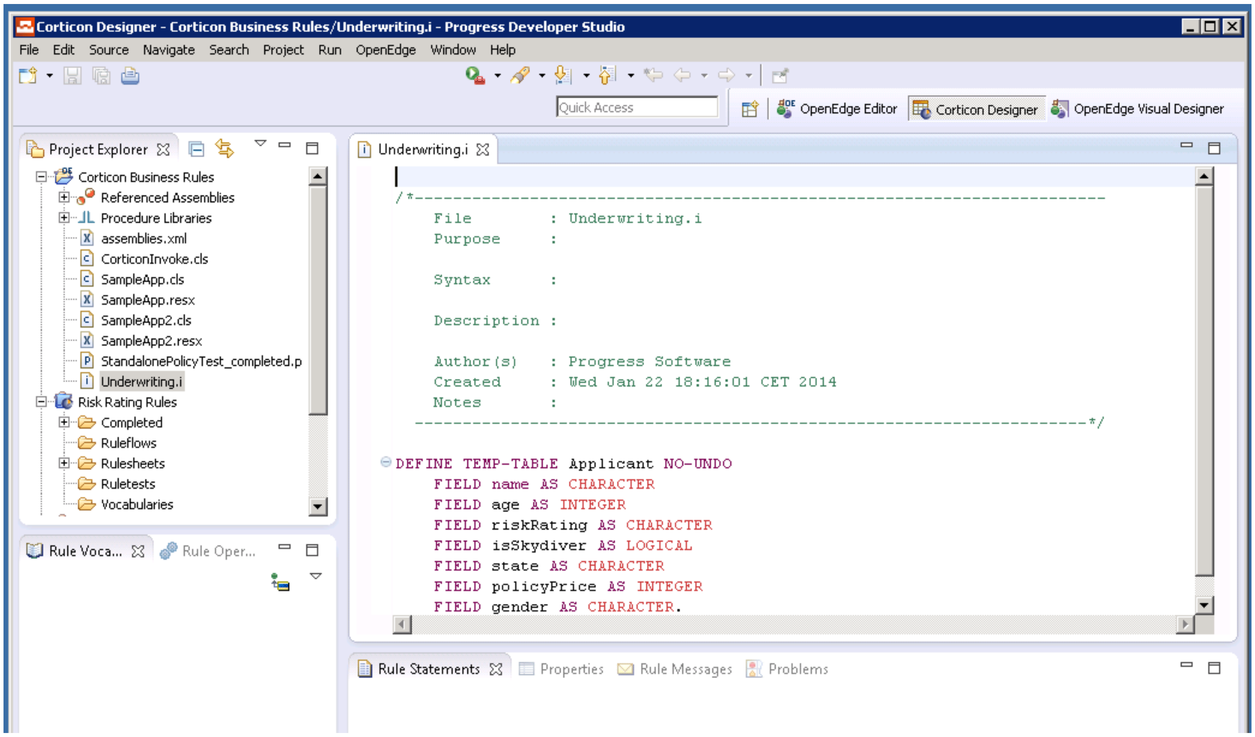
A Corticon decision model is comprised of a Vocabulary (data model), and sets of Rules against that model. The Vocabulary can be built from scratch inside OpenEdge Developer Studio (Corticon perspective), but it can also be generated from existing artifacts. OpenEdge provides the ability to create a 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 OpenEdge Developer 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 running under the Tomcat application server which is part of OpenEdge Developer Studio.

The Corticon Server (axis.war) is bundled with Corticon Studio (installed on this image including the Corticon Eclipse plugin into OpenEdge Developer Studio. It was added explicitly to enable OE developers to only need Corticon Studio for the developing and testing of OE/Corticon applications. 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.

Step 1 – Create Vocabulary

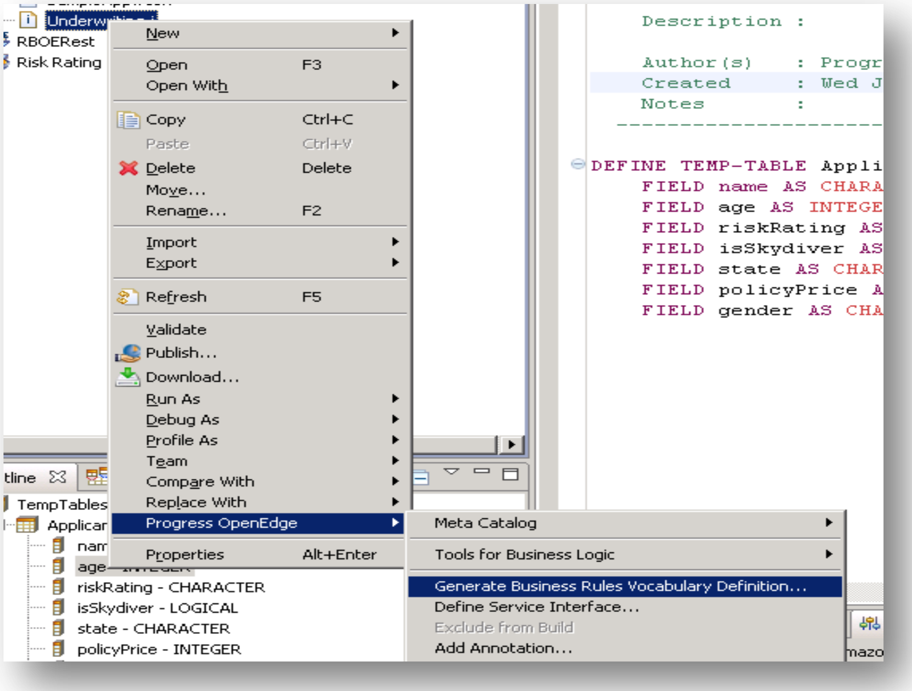
Launch Progress Developer Studio for OpenEdge from the desktop. If you are prompted to select a Workspace, you should use the following location **C:\OpenEdge\workspace.** You will see the "**Corticon Business Rules**", please click on the "+" and open the project. Double click **Underwriting.i** to reveal the contents of a pre-defined temp-table (Figure 2).



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

    Syntax      :

    Description :

    Author(s)   : Progress Software
    Created     : Wed Jan 22 18:16:01 CET 2014
    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 2

Right mouse click on Underwriting.i and select **Export/Business Rules Vocabulary Definition** (Figure 3).

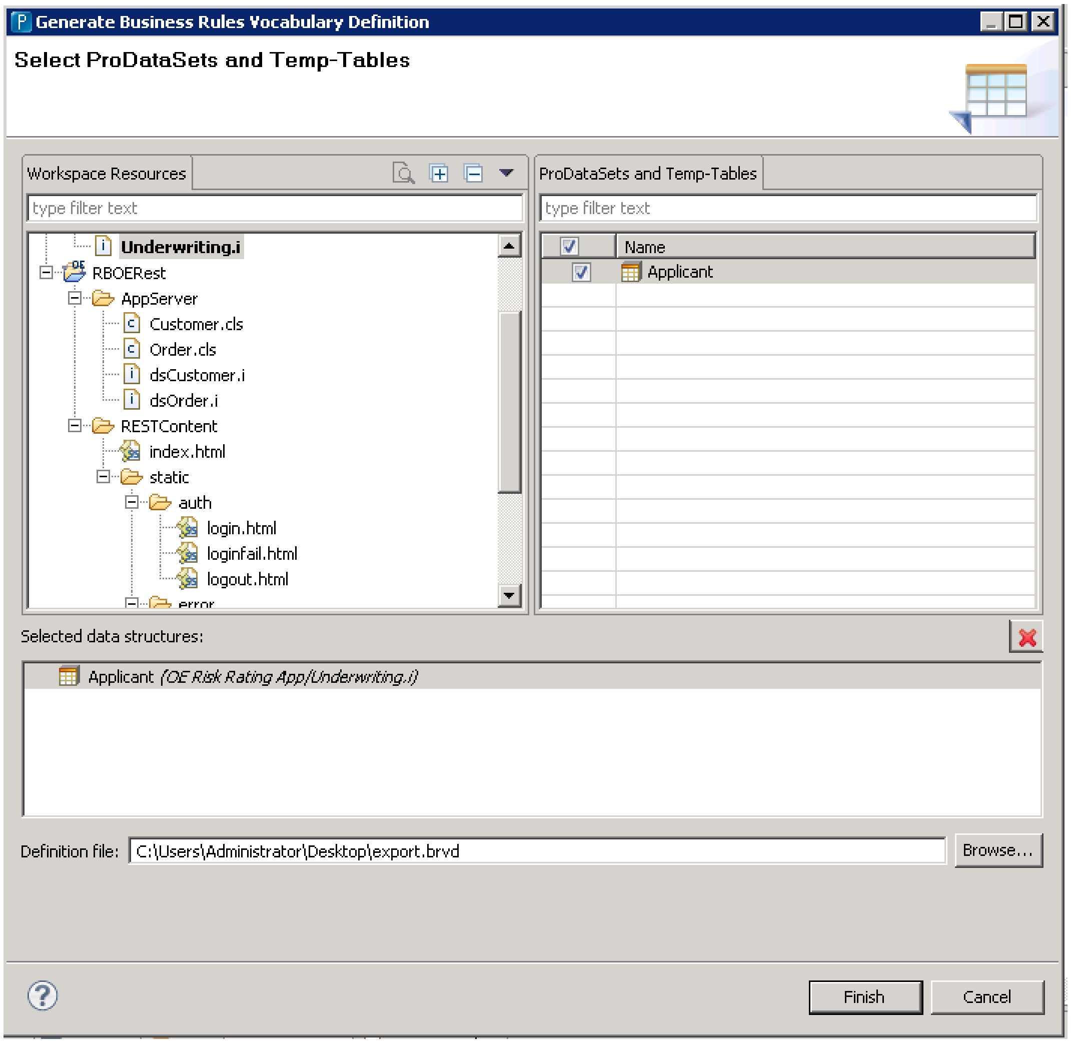


```
    Description :

    Author(s)    : Progr
    Created      : Wed J
    Notes        :
  ----------------------

DEFINE TEMP-TABLE Appli
    FIELD name AS CHARA
    FIELD age AS INTEGE
    FIELD riskRating AS
    FIELD isSkydiver AS
    FIELD state AS CHAR
    FIELD policyPrice A
    FIELD gender AS CHA
```

Figure 3

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 4).

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 OpenEdge Developer Studio in a later step.



Generate Business Rules Vocabulary Definition

Select ProDataSets and Temp-Tables

☑	Name
☑	Applicant

Selected data structures:

Applicant *(OE Risk Rating App/Underwriting.i)*

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

Figure 4

Close the **Underwriting.i** file & Right mouse click on the “Risk Rating Rules” project (Figure 5).

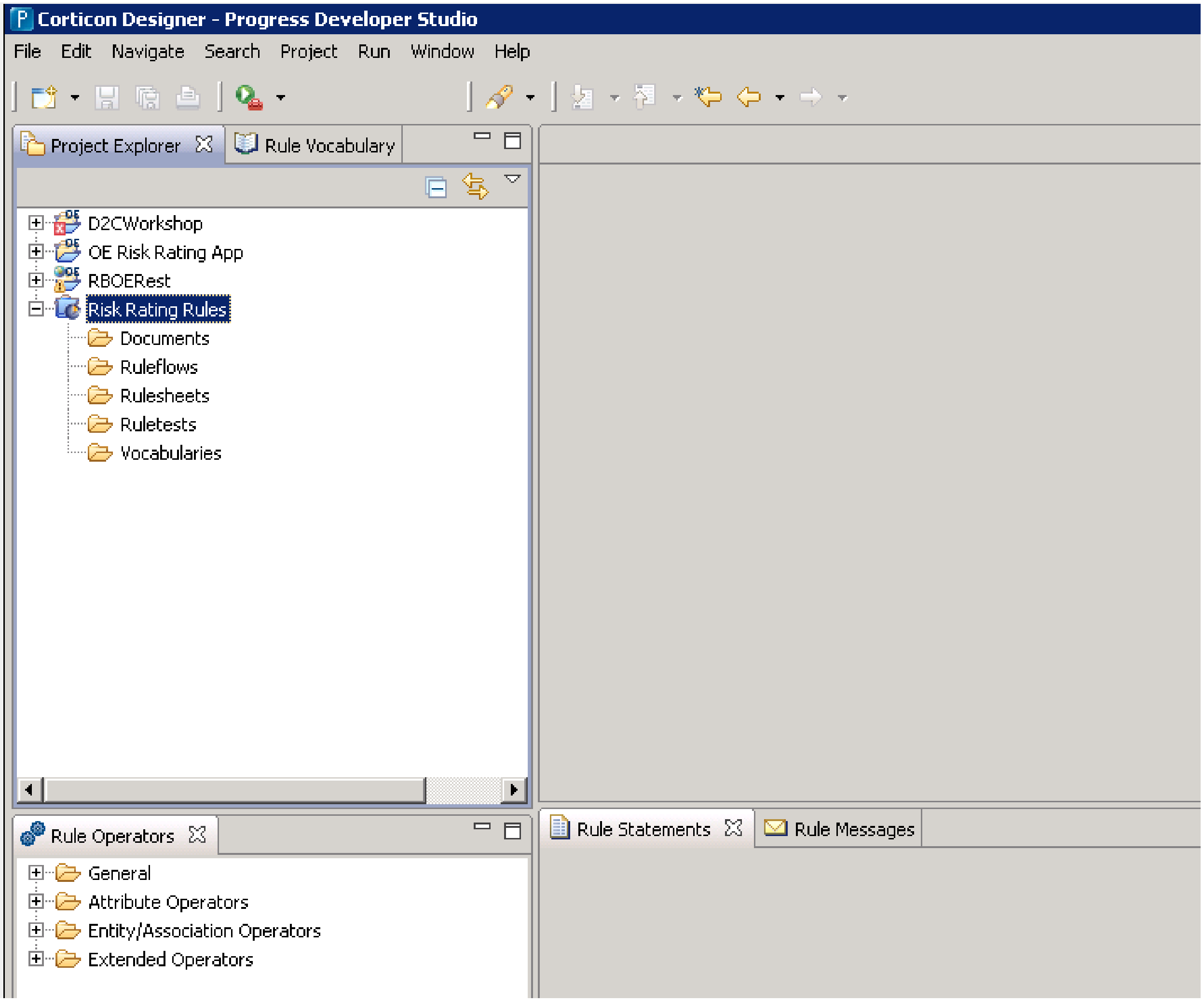


Figure 5

Right mouse click on the and select Import (Figure 6). This will launch the import wizard.

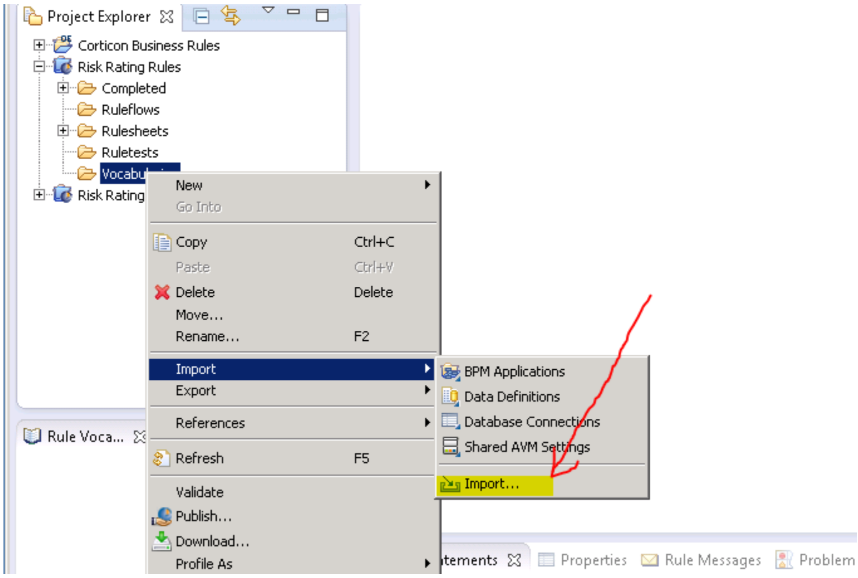
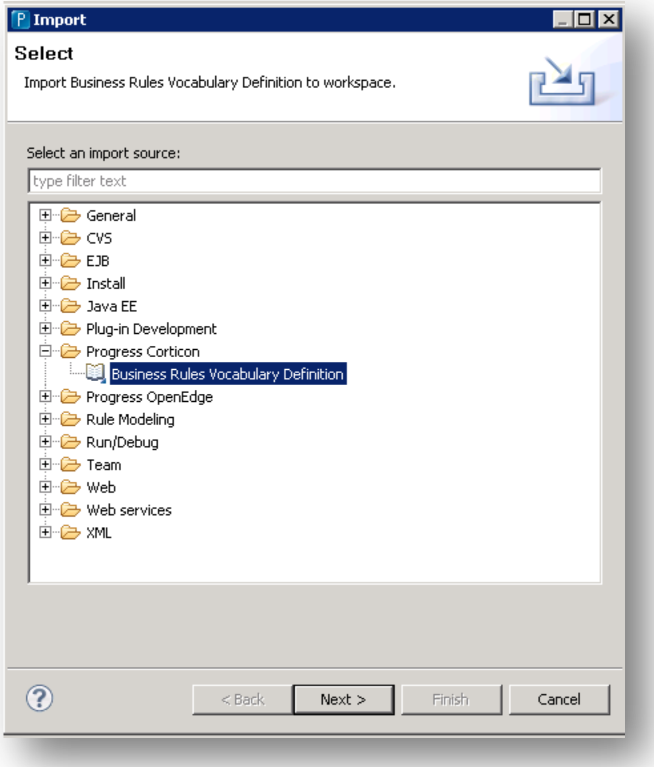


Figure 6

From here you should choose “Business Rule Vocabulary Definition” (Figure 7).

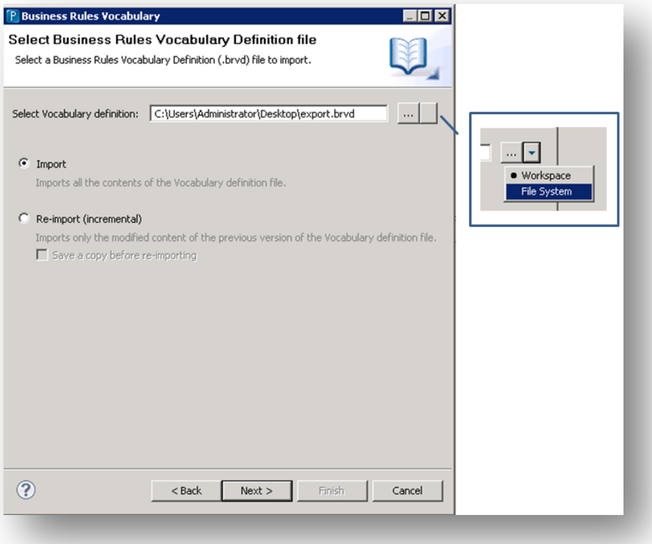

Import

Select

Import Business Rules Vocabulary Definition to workspace.

Figure 7

Next you should select the file that you exported in the previous steps (Figure 8).



Business Rules Vocabulary

Select Business Rules Vocabulary Definition file

Select a Business Rules Vocabulary Definition (.brvd) file to import.

Select Vocabulary definition: C:\Users\Administrator\Desktop\export.brvd

● Import

Imports all the contents of the Vocabulary definition file.

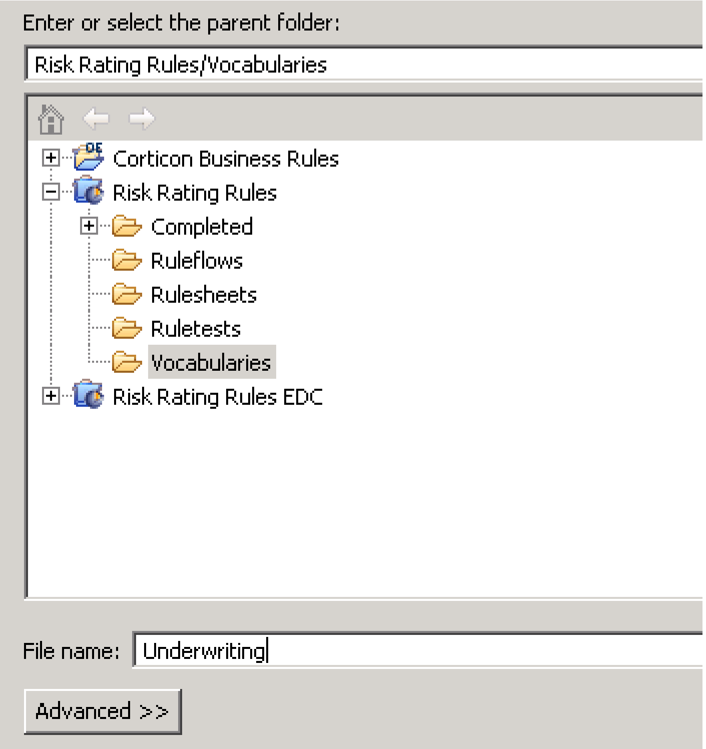
○ Re-import (incremental)

Imports only the modified content of the previous version of the Vocabulary definition file.

□ Save a copy before re-importing

Figure 8

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 9)**.**



Enter or select the parent folder:

Risk Rating Rules/Vocabularies

File name: Underwriting

Figure 9

You should now see **Underwriting.ecore** in the project explorer (Figure 10)**.**

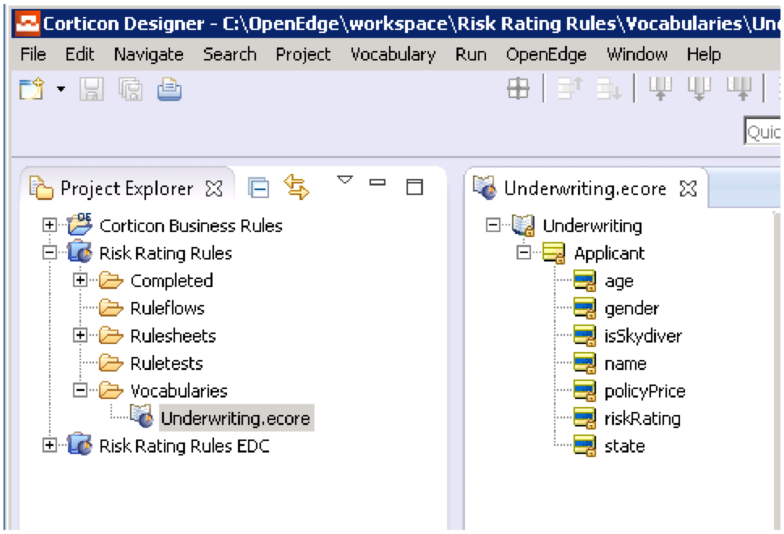
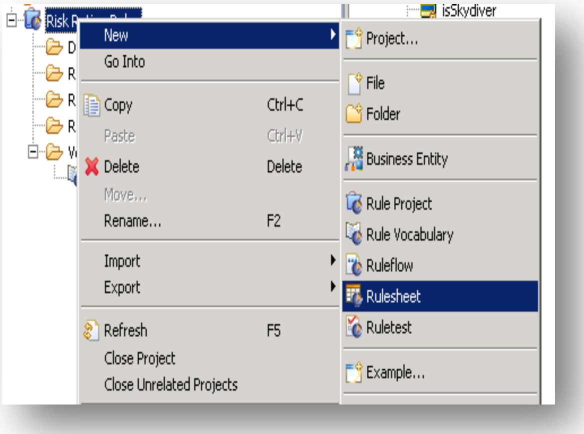


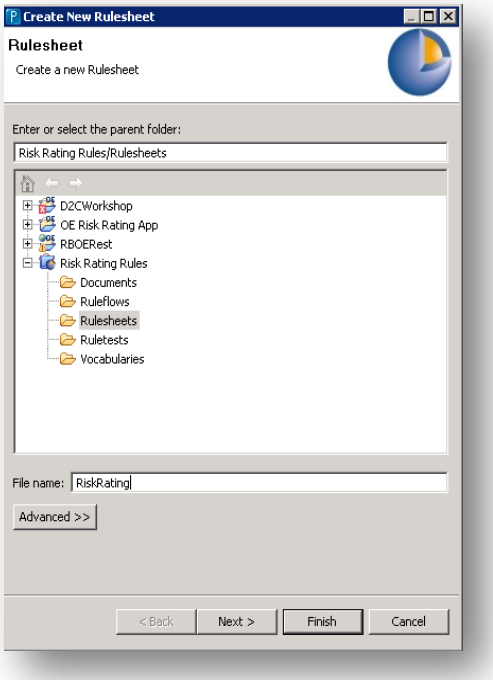
Figure 10

Now you have an **Underwriting.ecore** file. In the right pane, the generated vocabulary will be displayed. Mind the little "locks". This means that the Corticon vocabulary can only be changed when the Temp table definition is changed. To Ensure this stays in sync, you use the the Import wizard and select the RE-import feature.

Step 2 – Model Business Rules in Rulesheet Editor

To create a new Rulesheet, right mouse click on the project and select New/Rulesheet (Figure 11). Choose the “Rulesheet” folder and name it **RiskRating** (Figure 11 & 12**).**





Enter or select the parent folder: Risk Rating Rules/Rulesheets

File name: RiskRating

Figure 11 **Figure 12**

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 13).

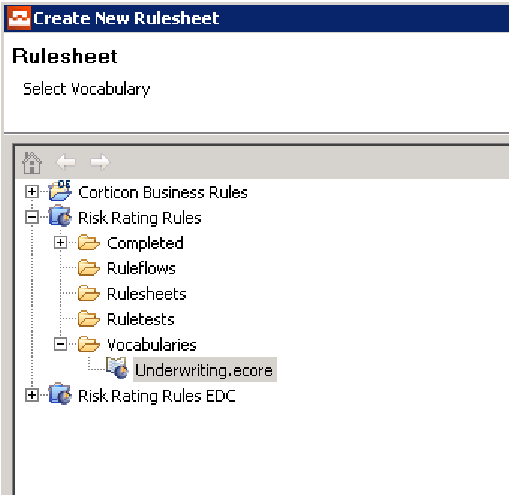


Figure 13

You are now presented with a blank Rulesheet, ready to use for modeling our rules (Figure 14).

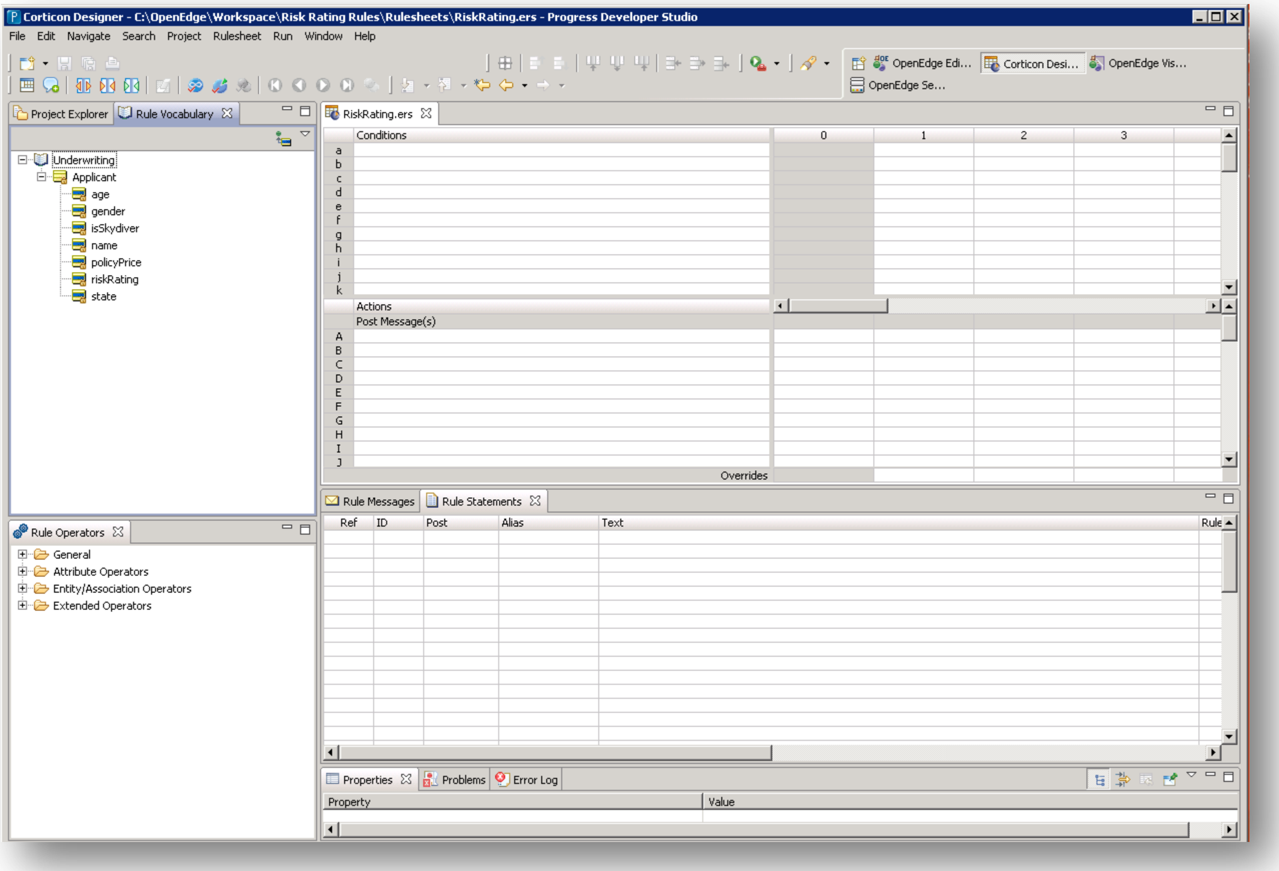


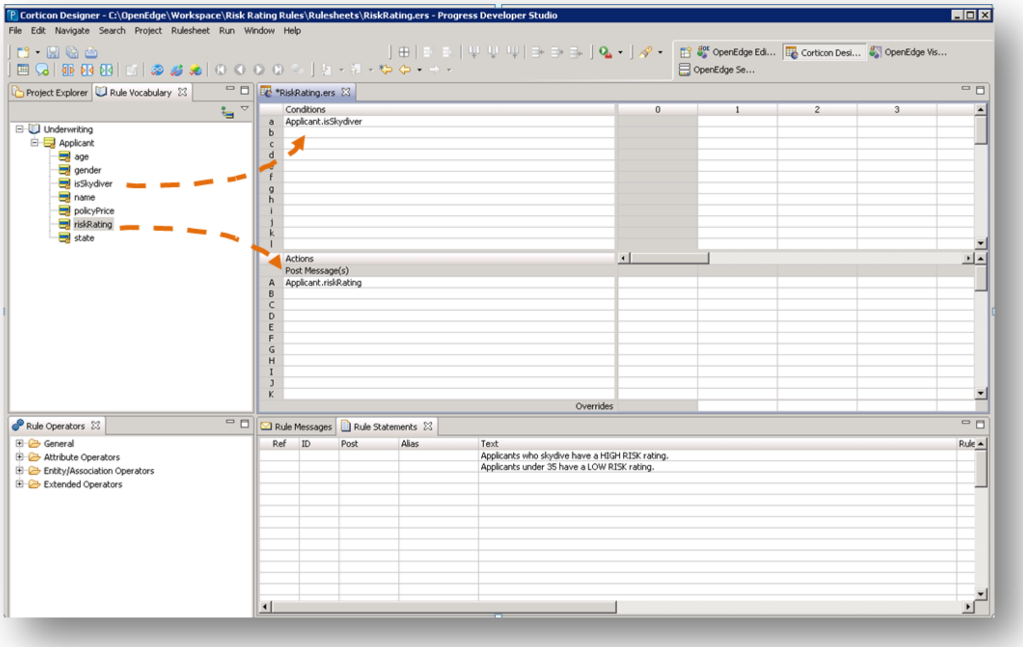
Figure 14

To model the rules, first type in the Business Rule Statements (Figure 15) in the Text field.

Row 1 - Applicants who skydive are a HIGH RISK rating

Row 2 – Applicants under 35 are a LOW RISK rating

Then 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 15).


Corticon Designer - C:\OpenEdge\Workspace\Risk Rating Rules\Rulesheets\RiskRating.ers - Progress Developer Studio
File Edit Navigate Search Project Rulesheet Run Window Help
OpenEdge Edi... Corticon Desi... OpenEdge Vis... OpenEdge Se...
Project Explorer
Rule Vocabulary
Underwriting
Applicant
age
gender
isSkydiver
name
policyPrice
riskRating
state
*RiskRating.ers
Rule Operators
General
Attribute Operators
Entity/Association Operators
Extended Operators
Rule Messages
Rule Statements

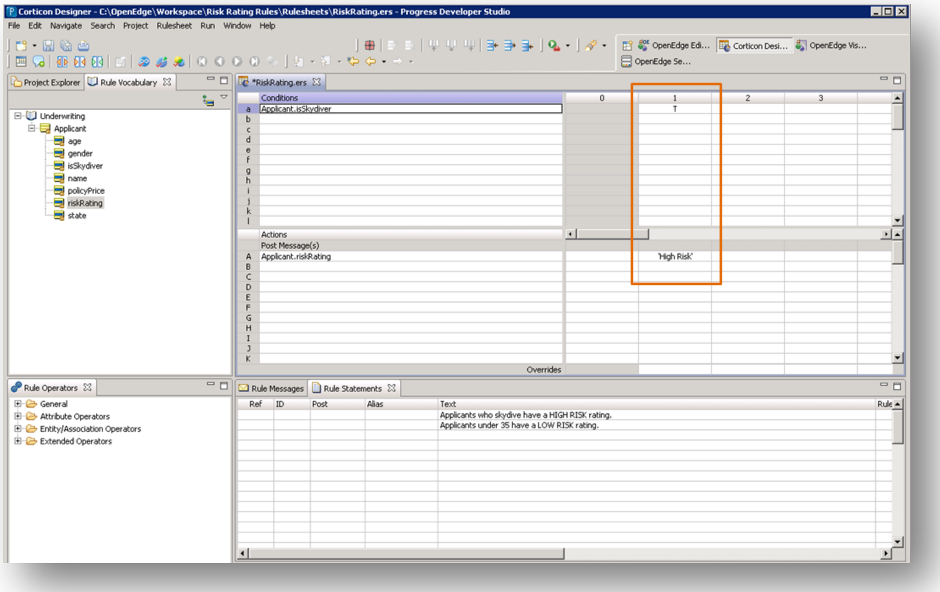
	Conditions	0	1	2	3
a	Applicant.isSkydiver				
b					
c					
d					
e					
f					
g					
h					
i					
j					
k					
l					
	Actions				
	Post Message(s)				
A	Applicant.riskRating				
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
	Overrides				

Ref	ID	Post	Alias	Text	Rule
				Applicants who skydive have a HIGH RISK rating.	
				Applicants under 35 have a LOW RISK rating.	

Figure 15

Now we have the rule, we now need to set the rule **condition** and **Action** to set the required **Decision**

In Column 1, select “T” (for True / Positive) as the value for the Condition and type in “High Risk” as the value for the Action. (Figure 16).



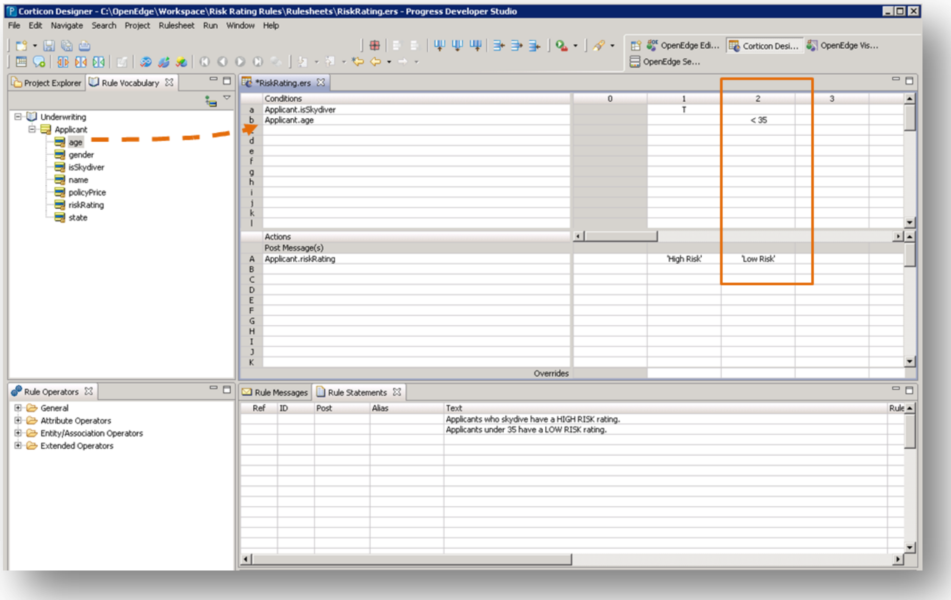
Corticon Designer - C:\OpenEdge\Workspace\Risk Rating Rules\Rulesheets\RiskRating.ers - Progress Developer Studio

	Conditions	0	1	2	3
a	Applicant.isSkydiver		T		
	Actions				
	Post Message(s)				
A	Applicant.riskRating		'High Risk'		
	Overrides				

Ref	ID	Post	Alias	Text	Rule
				Applicants who skydive have a HIGH RISK rating.	
				Applicants under 35 have a LOW RISK rating.	

Figure 16

For the second rule, drag the “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 17).



Corticon Designer - C:\OpenEdge\Workspace\Risk Rating Rules\Rulesheets\RiskRating.ers - Progress Developer Studio

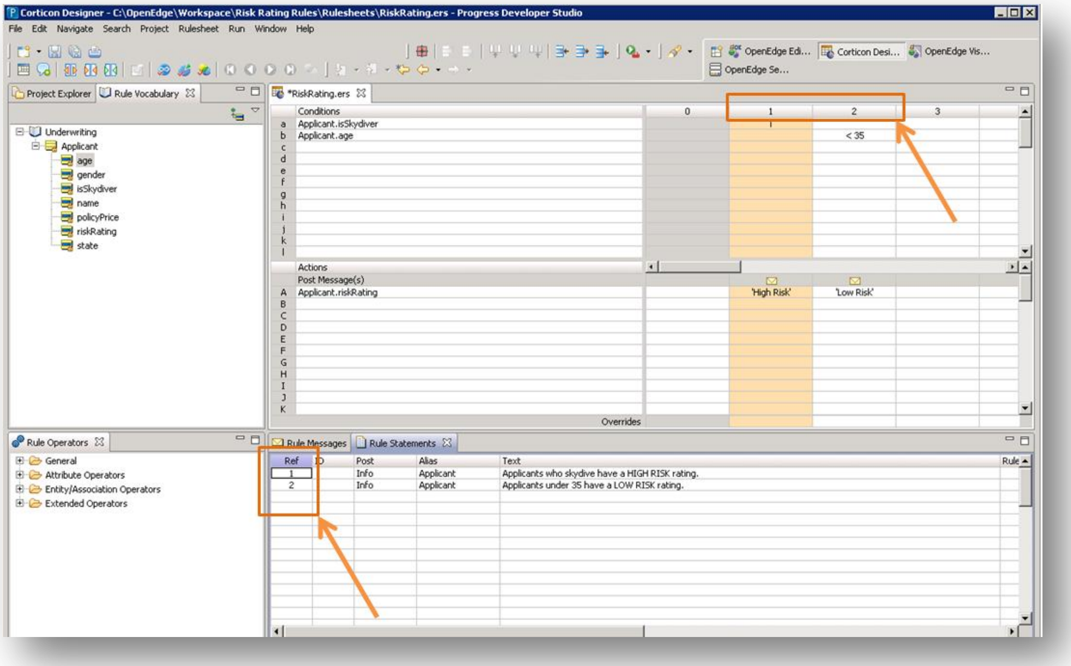
	Conditions	0	1	2	3
a	Applicant.isSkydiver		T		
b	Applicant.age			< 35	
	Actions				
	Post Message(s)				
A	Applicant.riskRating		'High Risk'	'Low Risk'	
	Overrides				

Ref	ID	Post	Alias	Text	Rule
				Applicants who skydive have a HIGH RISK rating.	
				Applicants under 35 have a LOW RISK rating.	

Figure 17

Finally, we need to link the Rule Statements (Business Rule) with the declarative representation (Condictin and Colum) by typing in the corresponding column number in the Rule Statement section.

If you would like to post the Rule Statement as a message to the request, You can choose to **post** (info, Warning or Violation) the Rule Statements as shown (Figure 18).

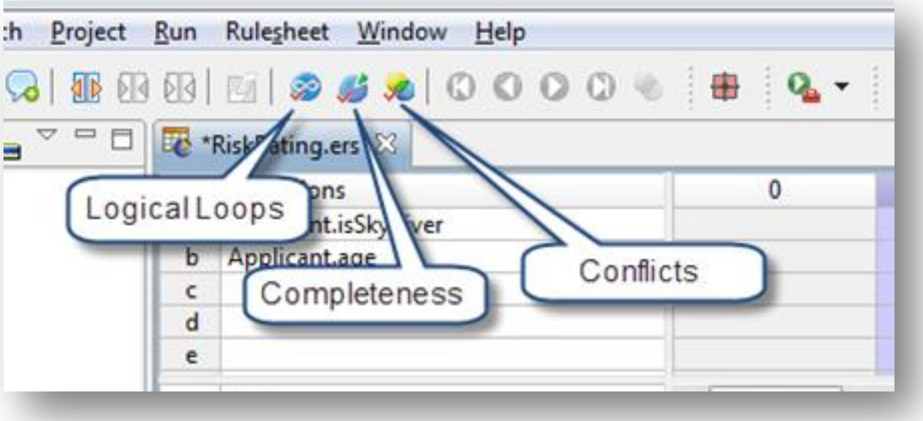


	Conditions	0	1	2	3
a	Applicant.isSkydiver		T		
b	Applicant.age			< 35	
	Actions				
	Post Message(s)		✉	✉	
A	Applicant.riskRating		'High Risk'	'Low Risk'	
	Overrides				

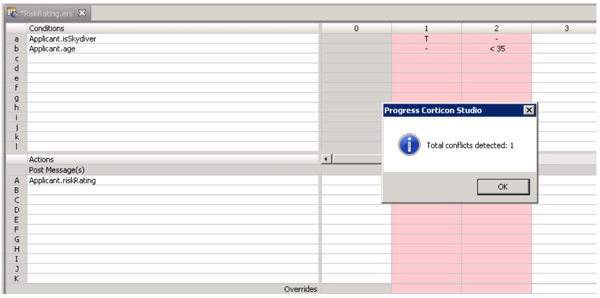
Ref	ID	Post	Alias	Text	Rule
1		Info	Applicant	Applicants who skydive have a HIGH RISK rating.	
2		Info	Applicant	Applicants under 35 have a LOW RISK rating.	

Figure 18

Now we have the defined the ruleset we need to check for any flaws in the rule model, such as conflicts and/or incompleteness. This means there maybe rules that overlap, or there maybe a gap in the logic or there are 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.



Click on each button to see what response you get, it should be something like the 2 screenshots below.

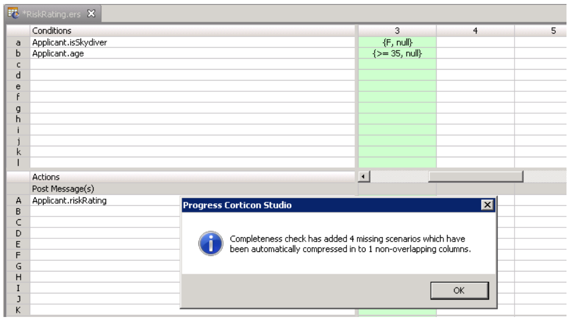


	Conditions	0	1	2	3
a	Applicant.isSkydiver		T	-	
b	Applicant.age		-	< 35	
c					
d					
e					
f					
g					
h					
i					
j					
k					
l					
	Actions				
	Post Message(s)				
A	Applicant.riskRating				
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
	Overrides				

Progress Corticon Studio

Total conflicts detected: 1

OK



	Conditions	3	4	5
a	Applicant.isSkydiver	{F, null}		
b	Applicant.age	{>= 35, null}		
c				
d				
e				
f				
g				
h				
i				
j				
k				
l				
	Actions			
	Post Message(s)			
A	Applicant.riskRating			
B				
C				
D				
E				
F				
G				
H				
I				
J				
K				

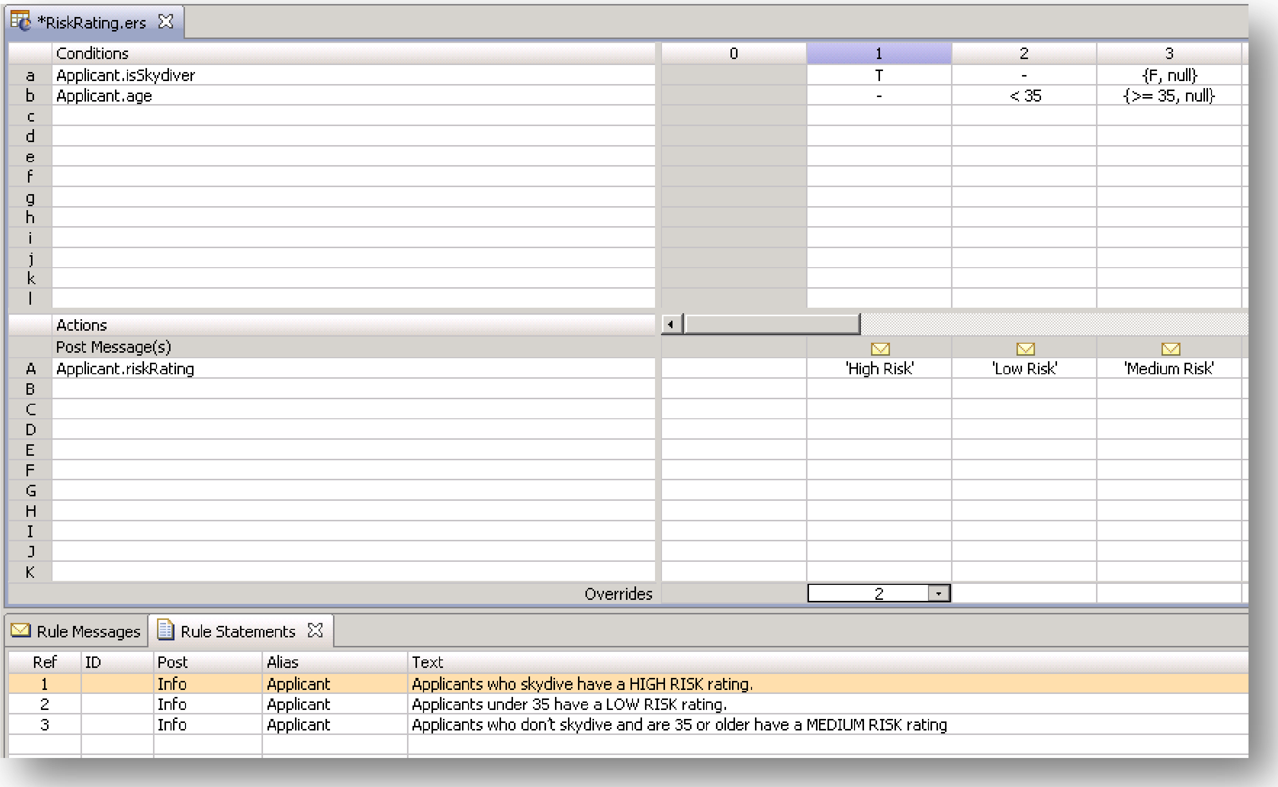
Progress Corticon Studio

Completeness check has added 4 missing scenarios which have been automatically compressed in to 1 non-overlapping columns.

OK

Now we will resolve these rule integrity issues as follows:

1. For the rule conflict, introduce a **rule priority override** for rule 2 in rule column 1. Skydiving is dangerous regardless age, so ignoring rule 2 (the age rule) makes sense in this context.
2. For the missing rule (3), add a risk rating of MEDIUM RISK. Also don't forget to add the missing rule statement text (Applicants who don't skydive and are 35 or older have a MEDIUM RISK rating).



	Conditions	0	1	2	3
a	Applicant.isSkydiver		T	-	{F, null}
b	Applicant.age		-	< 35	{>= 35, null}
c					
d					
e					
f					
g					
h					
i					
j					
k					
l					
	Actions				
	Post Message(s)		✉	✉	✉
A	Applicant.riskRating		'High Risk'	'Low Risk'	'Medium Risk'
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
	Overrides		2		

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.
3		Info	Applicant	Applicants who don't skydive and are 35 or older have a MEDIUM RISK rating

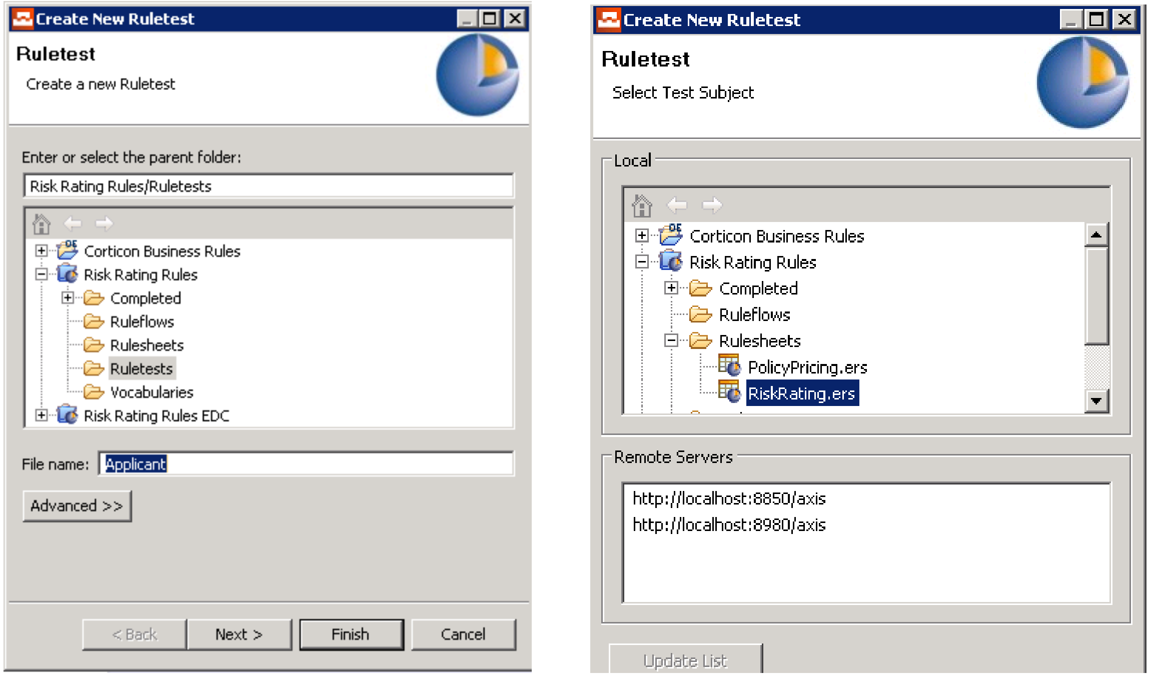
Figure 19: Complete and consistent rulesheet

NB: For simplicity reasons, we have ignored the potential NULL values in rule 3. Corticon will flag the possibility that your input data may not contain values. Usually this situation is prevented by setting the Mandatory flag to TRUE in the vocabulary meaning data presence is enforced at service level. This is not possible with the current level of integration between OE and Corticon. That said, based on customer input, Progress may relax this restriction in the next version of Corticon.

NOW, be careful! **SAVE** your rulesheet. This is the only way we can test the rules!

Step 3 – Create Tests

Right click on the Project and select New/Ruletest. Name it **Applicant** and hit next, You will then choose the Rulesheet that you want to test against. Select the RiskRating Rulesheet you have just created (Figure 20).



Create New Ruletest

Ruletest

Create a new Ruletest

Enter or select the parent folder: Risk Rating Rules/Ruletests

File name: Applicant

Create New Ruletest

Ruletest

Select Test Subject

Local: Risk Rating Rules/Rulesheets/RiskRating.ers

Remote Servers:

http://localhost:8850/axis
http://localhost:8980/axis

Figure 20

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. Then 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 21).

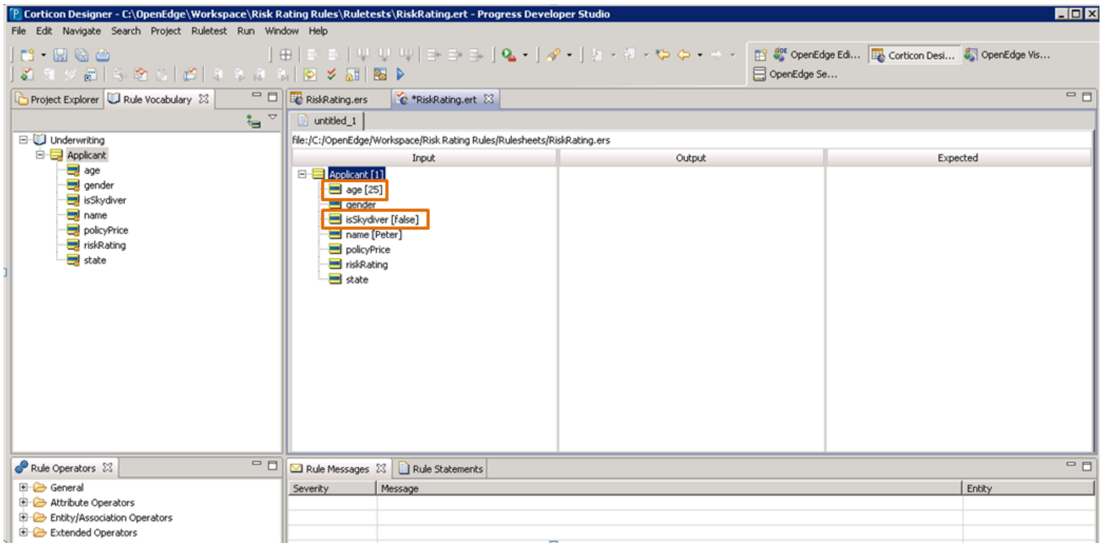
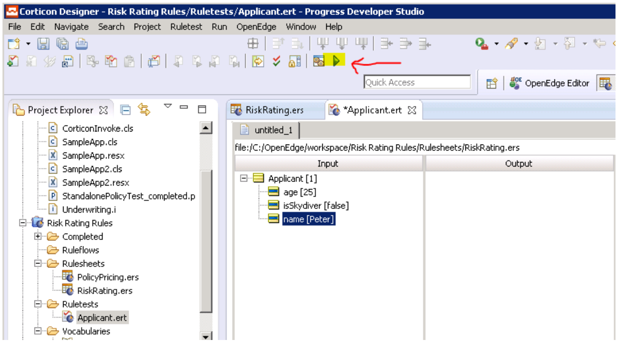


Figure 21

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 22, 23).



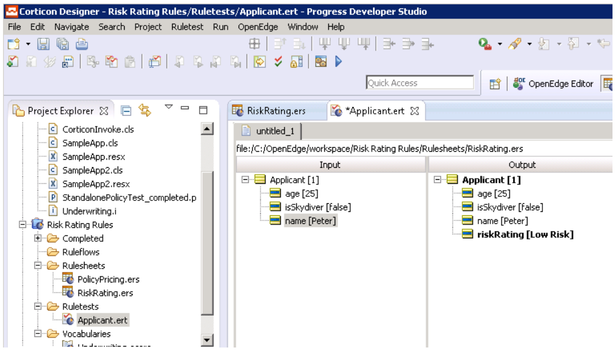
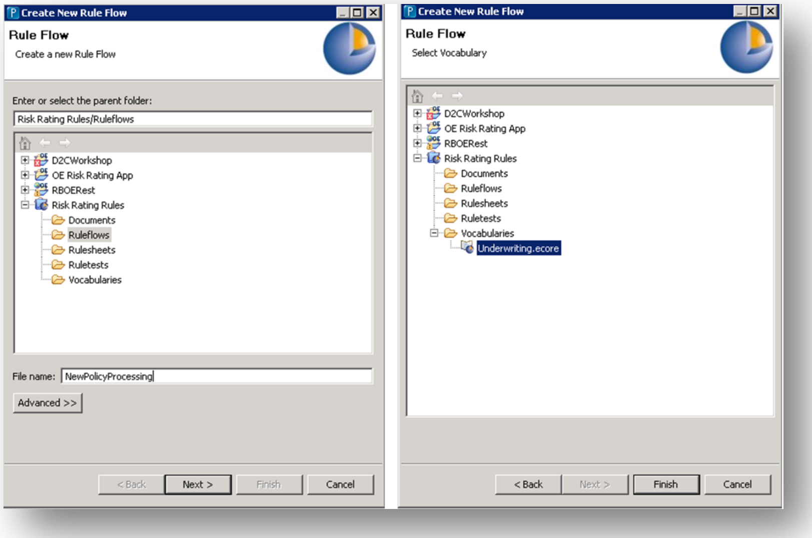


Figure 22

Figure 23

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 24). Next select the Vocabulary to use for this Ruleflow. Choose the Underwriting.ecore that you created earlier.



Create New Rule Flow

Rule Flow

Create a new Rule Flow

Enter or select the parent folder: Risk Rating Rules/Ruleflows

File name: NewPolicyProcessing

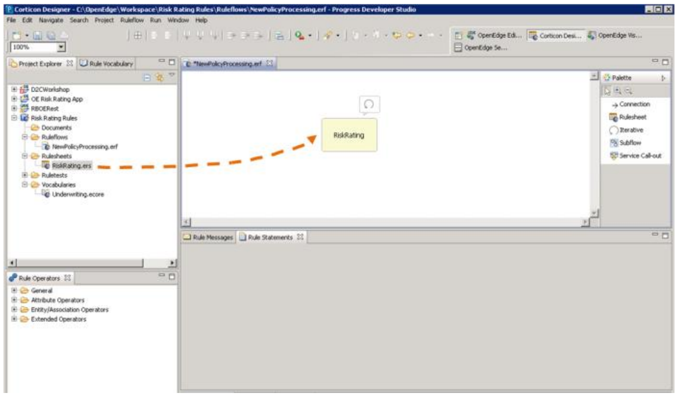
Create New Rule Flow

Rule Flow

Select Vocabulary

Figure 24

Finally, drag the RiskRating.ers Rulesheet (Figure 25) from the Project Explorer on to the Ruleflow canvas and **SAVE**, you now have a rule that you can publish to the Corticon Server.



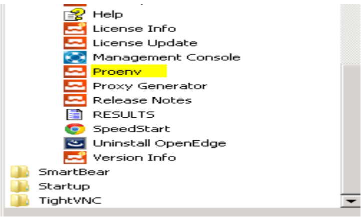
Corticon Designer - C:\OpenEdge\Workspace\Risk Rating Rules\Ruleflows\NewPolicyProcessing.erf - Progress Developer Studio

Figure 25

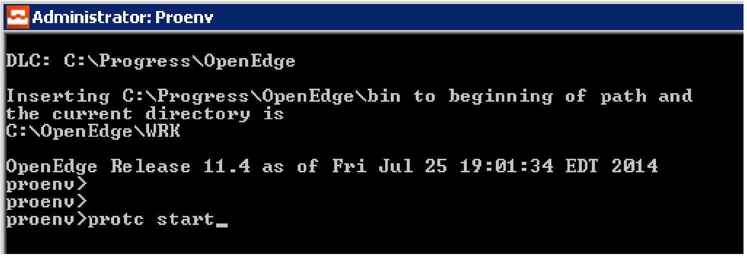
The decision service is completed and ready to be deployed to the Corticon Server.

Start the Corticon Server as follows:

1. Go to the Windows Start Menu → All Programs/Progress/OpenEdge 11.4/Proenv

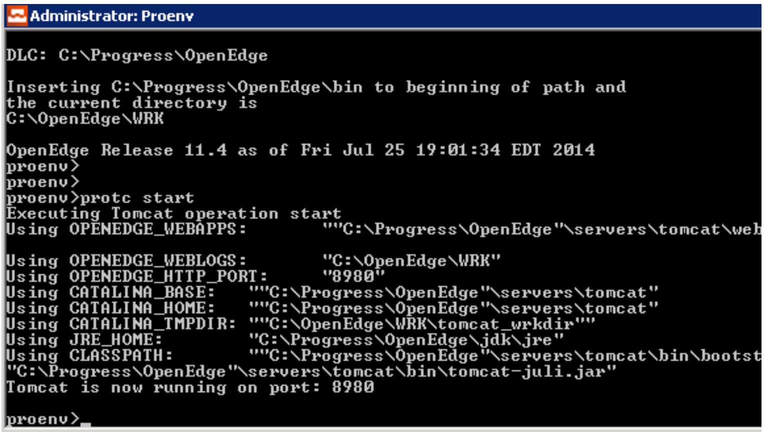


2. Type in the DOS box: **protc start**



```
DLC: C:\Progress\OpenEdge

Inserting C:\Progress\OpenEdge\bin to beginning of path and
the current directory is
C:\OpenEdge\WRK

OpenEdge Release 11.4 as of Fri Jul 25 19:01:34 EDT 2014
proenv>
proenv>
proenv>protc start_
```

3. Hit Enter. The Tomcat application Server with Corticon Server will now start up, once running (see message at the bottom).



```
DLC: C:\Progress\OpenEdge

Inserting C:\Progress\OpenEdge\bin to beginning of path and
the current directory is
C:\OpenEdge\WRK

OpenEdge Release 11.4 as of Fri Jul 25 19:01:34 EDT 2014
proenv>
proenv>
proenv>protc start
Executing Tomcat operation start
Using OPENEDGE_WEBAPPS:        ""C:\Progress\OpenEdge"\servers\tomcat\web

Using OPENEDGE_WEBLOGS:        "C:\OpenEdge\WRK"
Using OPENEDGE_HTTP_PORT:      "8980"
Using CATALINA_BASE:   ""C:\Progress\OpenEdge"\servers\tomcat"
Using CATALINA_HOME:   ""C:\Progress\OpenEdge"\servers\tomcat"
Using CATALINA_TMPDIR: ""C:\OpenEdge\WRK\tomcat_wrkdir""
Using JRE_HOME:        "C:\Progress\OpenEdge\jdk\jre"
Using CLASSPATH:       ""C:\Progress\OpenEdge"\servers\tomcat\bin\bootst
"C:\Progress\OpenEdge"\servers\tomcat\bin\tomcat-juli.jar"
Tomcat is now running on port: 8980

proenv>_
```

You can close the DOS window.

In OpenEdge Developer Studio, right mouse click on the project and select “**Publish**”.

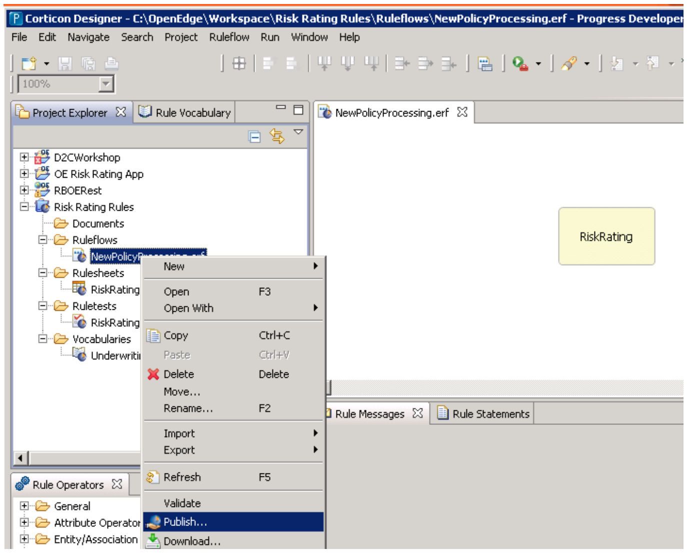
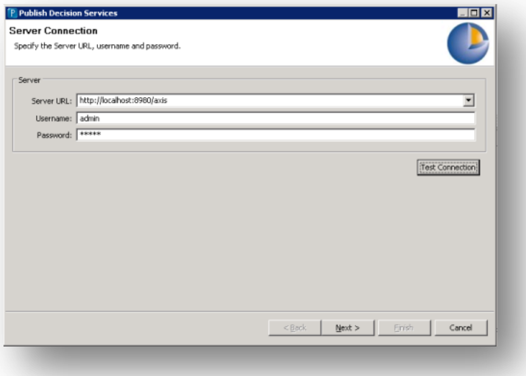


Figure 26

Provide the connection details to the Corticon Server you want to publish to (Figure 27).



Server Connection

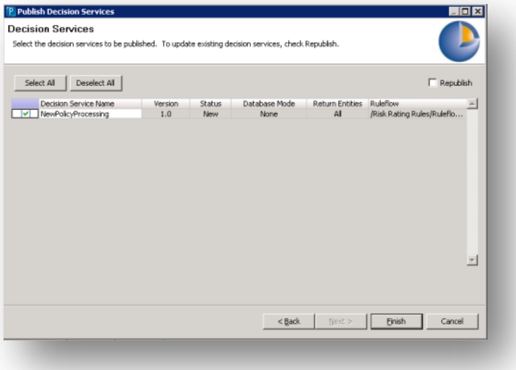
Specify the Server URL, username and password.

Server

Server URL: http://localhost:8980/axis

Username: admin

Password: *****



Decision Services

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

□ Republish

	Decision Service Name	Version	Status	Database Mode	Return Entities	Ruleflow
☑	NewPolicyProcessing	1.0	New	None	All	/Risk Rating Rules/Ruleflo...

Figure 27

Figure 28

URL: http://localhost:8980/axis

User: admin

Password: admin

Select the “NewPolicyProcessing” flow and Finish (Figure 28).

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:8980/axis **(there is a shortcut in the menu bar in Google Chrome).**

User: admin

Password: admin

Once logged in you can access the deployed decision services (Figure 29), and see a list of all decisions and various details. You should see v1.0 of the NewPolicyProcessing service (Figure 30).

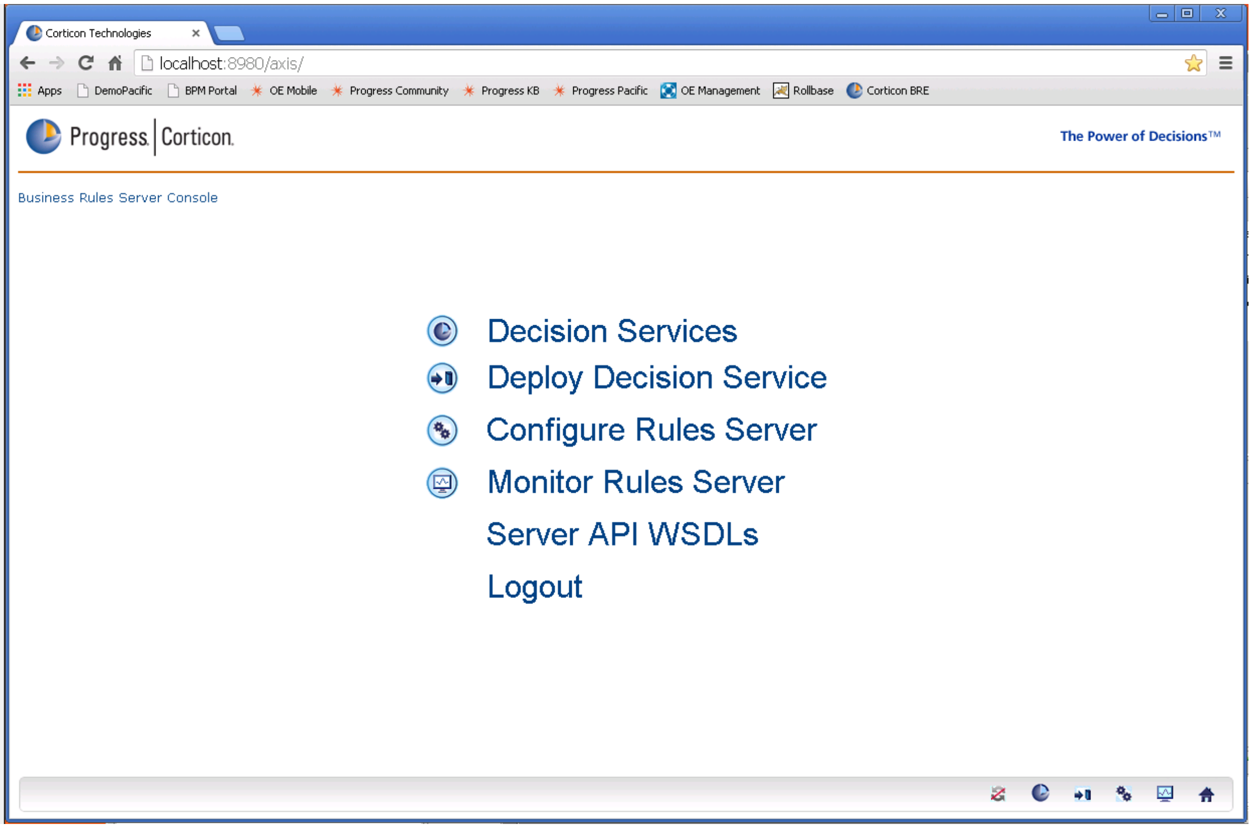
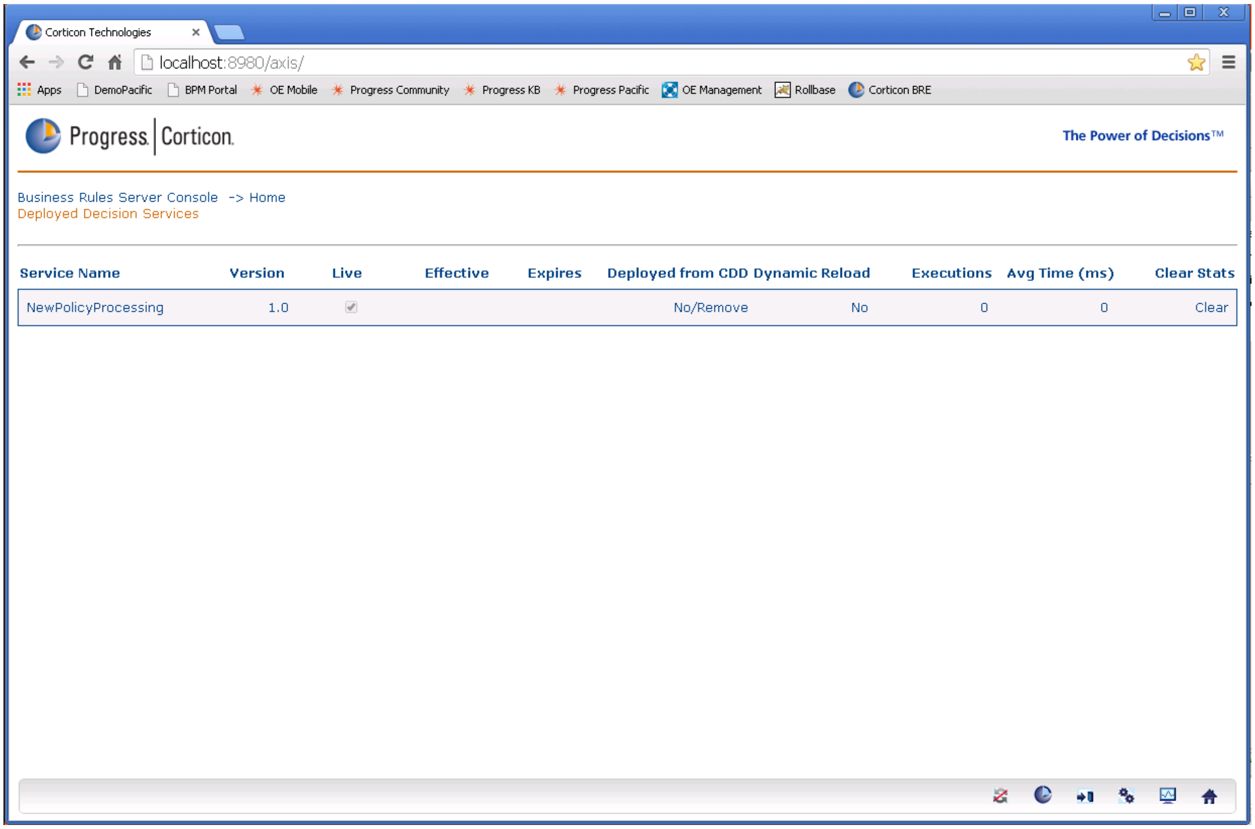


Figure 29



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
NewPolicyProcessing	1.0	☑			No/Remove	No	0	0	Clear

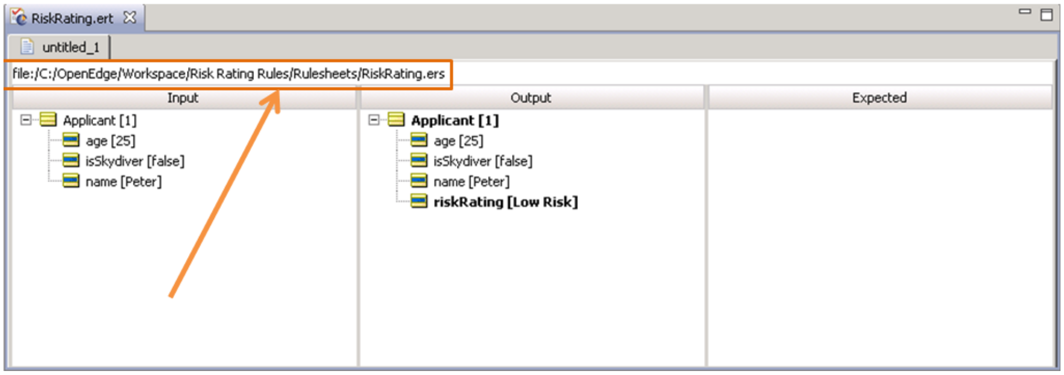
Figure 30

Step 5 – Test Deployed Decision Service

OpenEdge Developer Studio allows you to test against remote decision services deployed under **LOCALHOST**. If Corticon Server is licensed (and probably “lives” on a separate Server), you’d be able to test against any defined URL end point.

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 31), choose “Remote Servers”, click the “Update List” button, and select NewPolicyProcessing in the list (Figure 32).

Now when you execute the test, OpenEdge Developer Studio will create a SOAP message, post it to the remote URL and display the result payload from the deployed decision service (Figure 33).



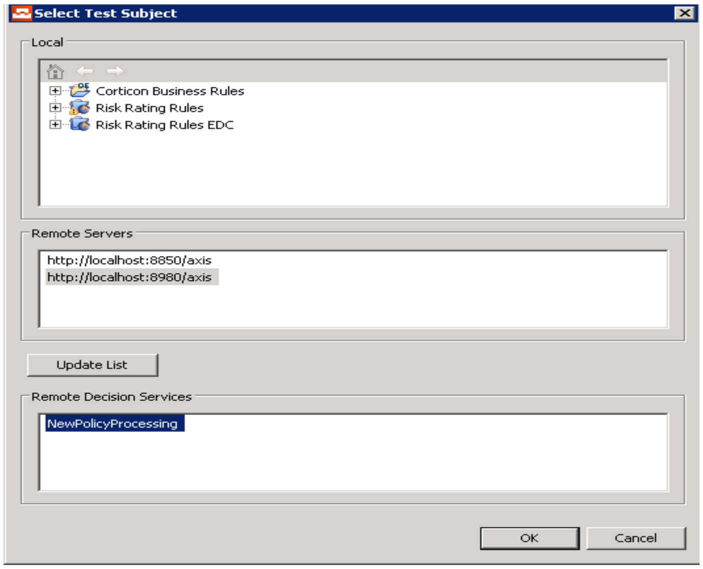
RiskRating.ert

untitled_1

file:/C:/OpenEdge/Workspace/Risk Rating Rules/Rulesheets/RiskRating.ers

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

Figure 31



Select Test Subject

Local

- Corticon Business Rules
- Risk Rating Rules
- Risk Rating Rules EDC

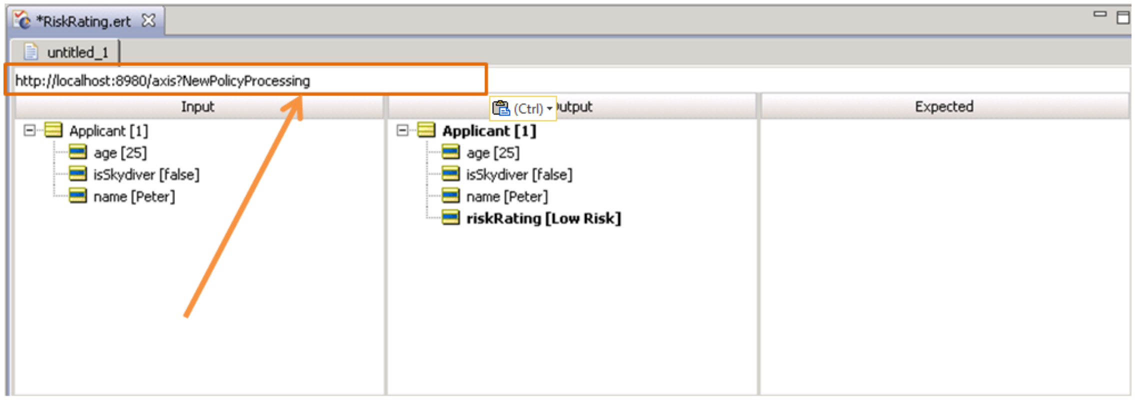
Remote Servers

http://localhost:8850/axis
http://localhost:8980/axis

Remote Decision Services

NewPolicyProcessing

Figure 32



*RiskRating.ert

untitled_1

http://localhost:8980/axis?NewPolicyProcessing

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

Figure 33

Exercise 2 – Integrate Corticon Decision Service

In this exercise we will use the new features of OpenEdge 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.

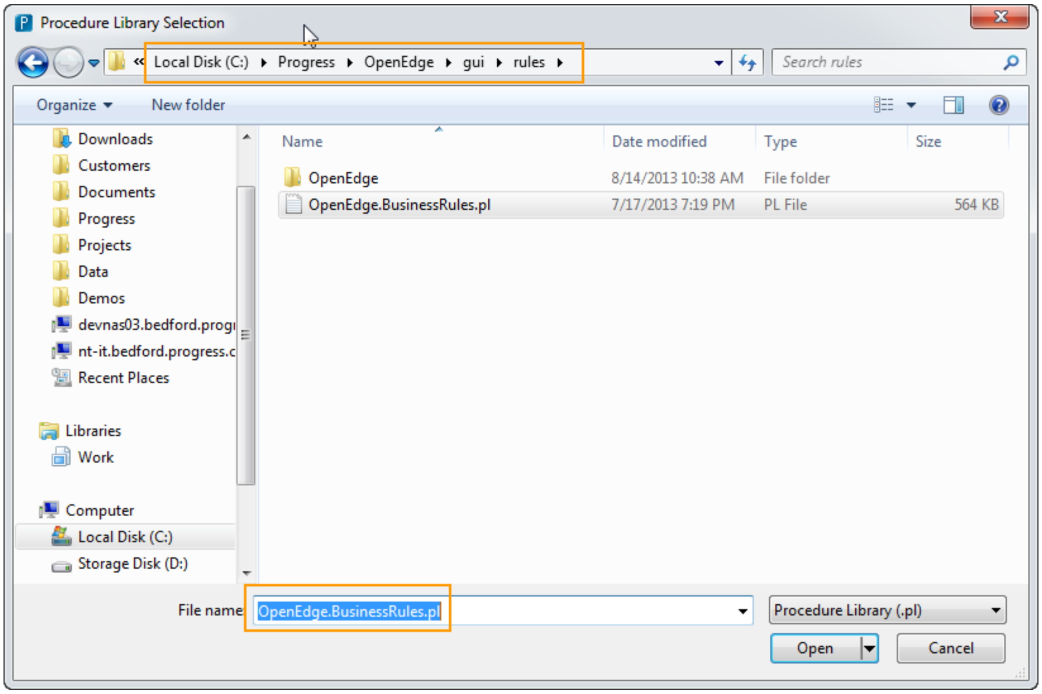
Step 1 – Configure the Environment

Launch Developer Studio for OpenEdge from the desktop, if it is not already running and activate the OpenEdge Editor perspective. To leverage the built in connectivity with Corticon in OpenEdge 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 34)

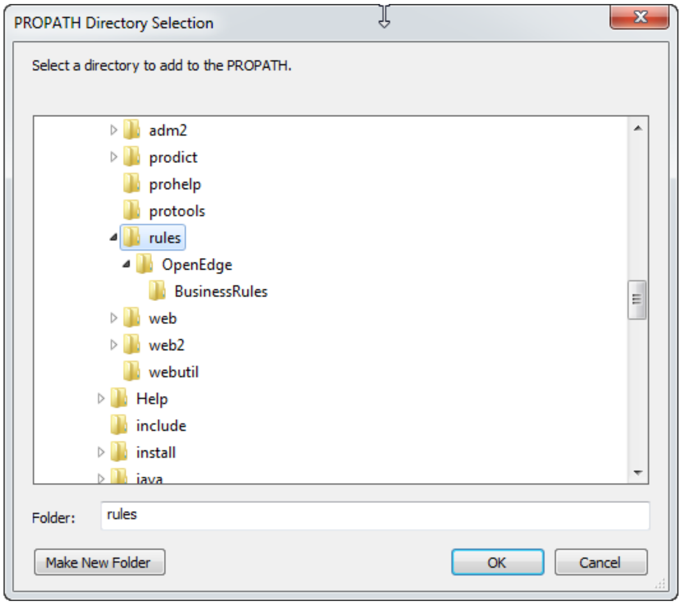


Name	Date modified	Type	Size
OpenEdge	8/14/2013 10:38 AM	File folder	
OpenEdge.BusinessRules.pl	7/17/2013 7:19 PM	PL File	564 KB

File name: OpenEdge.BusinessRules.pl

Figure 34

Click “Add External Directory” and select “C:\Progress\OpenEdge\gui\rules” (Figure 35)

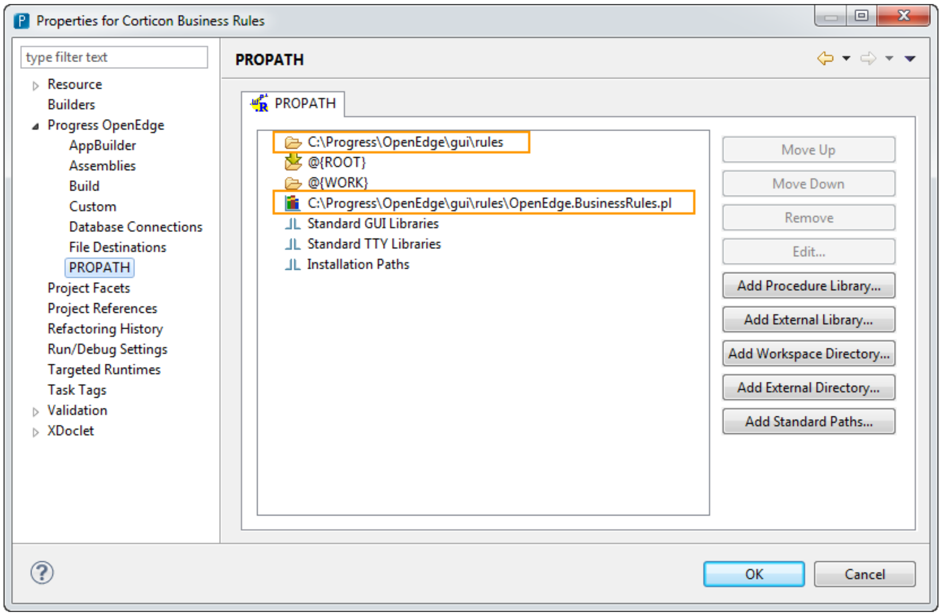

PROPATH Directory Selection

Select a directory to add to the PROPATH.

Folder: rules

Figure 35

You should now see the 2 components in the Properties view (Figure 36).


Properties for Corticon Business Rules

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

Figure 36

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 37). To view and edit the code associated with the form, right mouse click on the form and select “View Source”, or press F9 (Figure 38).

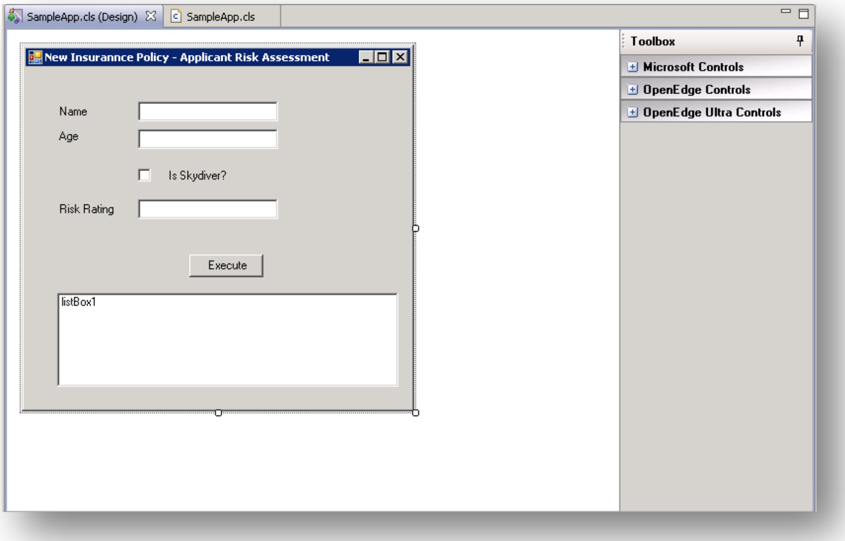
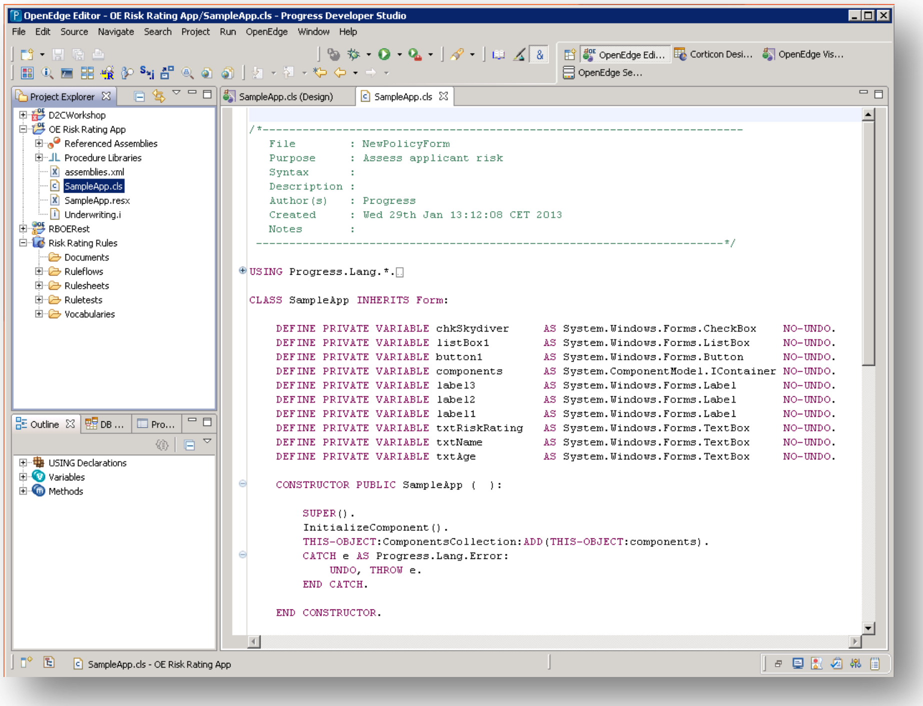


Figure 37

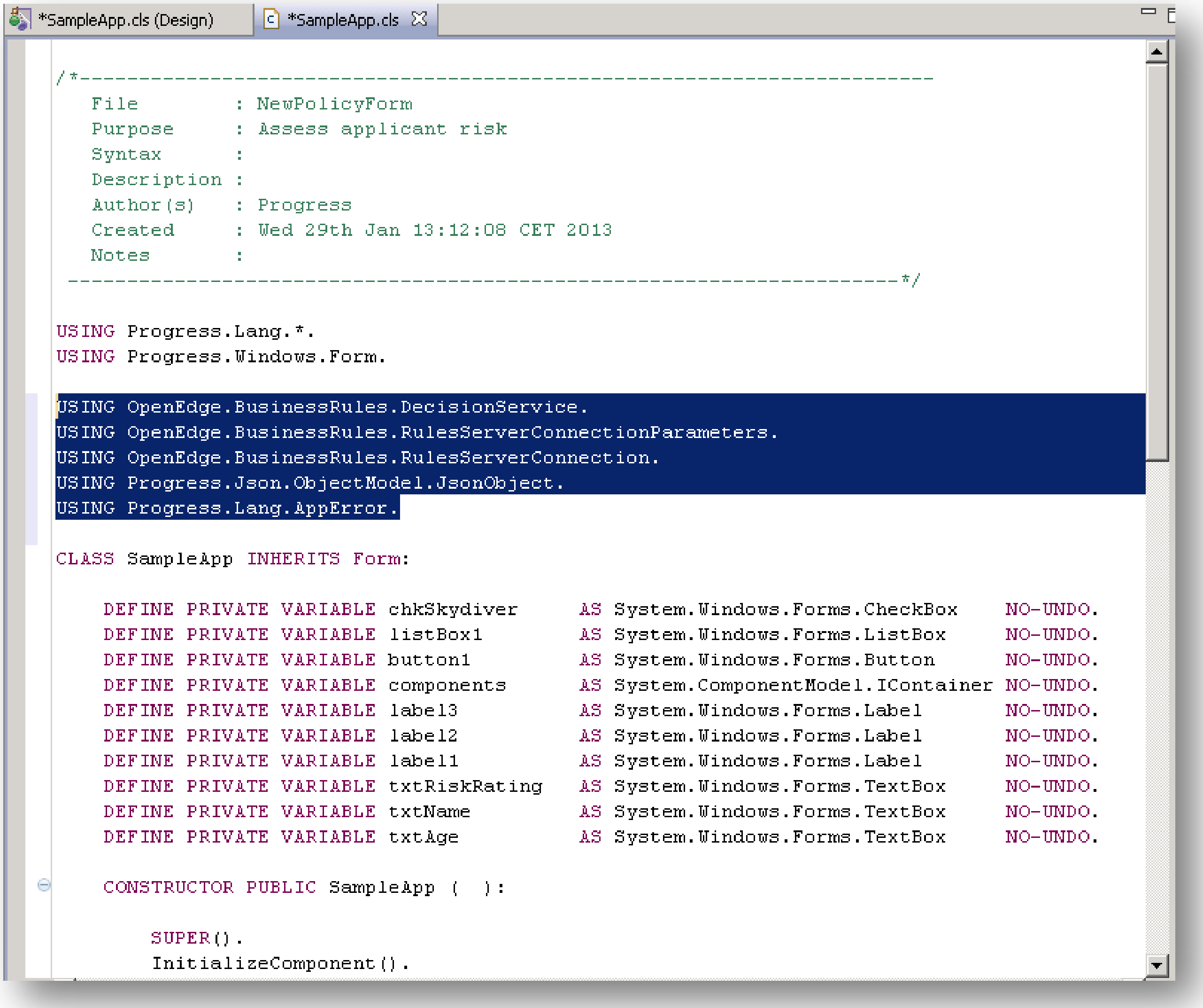


```
/*------------------------------------------------------------------------
   File        : NewPolicyForm
   Purpose     : Assess applicant risk
   Syntax      :
   Description :
   Author(s)   : Progress
   Created     : Wed 29th Jan 13:12:08 CET 2013
   Notes       :
 ----------------------------------------------------------------------*/

USING Progress.Lang.*.

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.

    CONSTRUCTOR PUBLIC SampleApp (  ):

        SUPER().
        InitializeComponent().
        THIS-OBJECT:ComponentsCollection:ADD(THIS-OBJECT:components).
        CATCH e AS Progress.Lang.Error:
            UNDO, THROW e.
        END CATCH.

    END CONSTRUCTOR.
```

Figure 38

The first step is to add the USING references as shown in Figure 39. 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*/
```



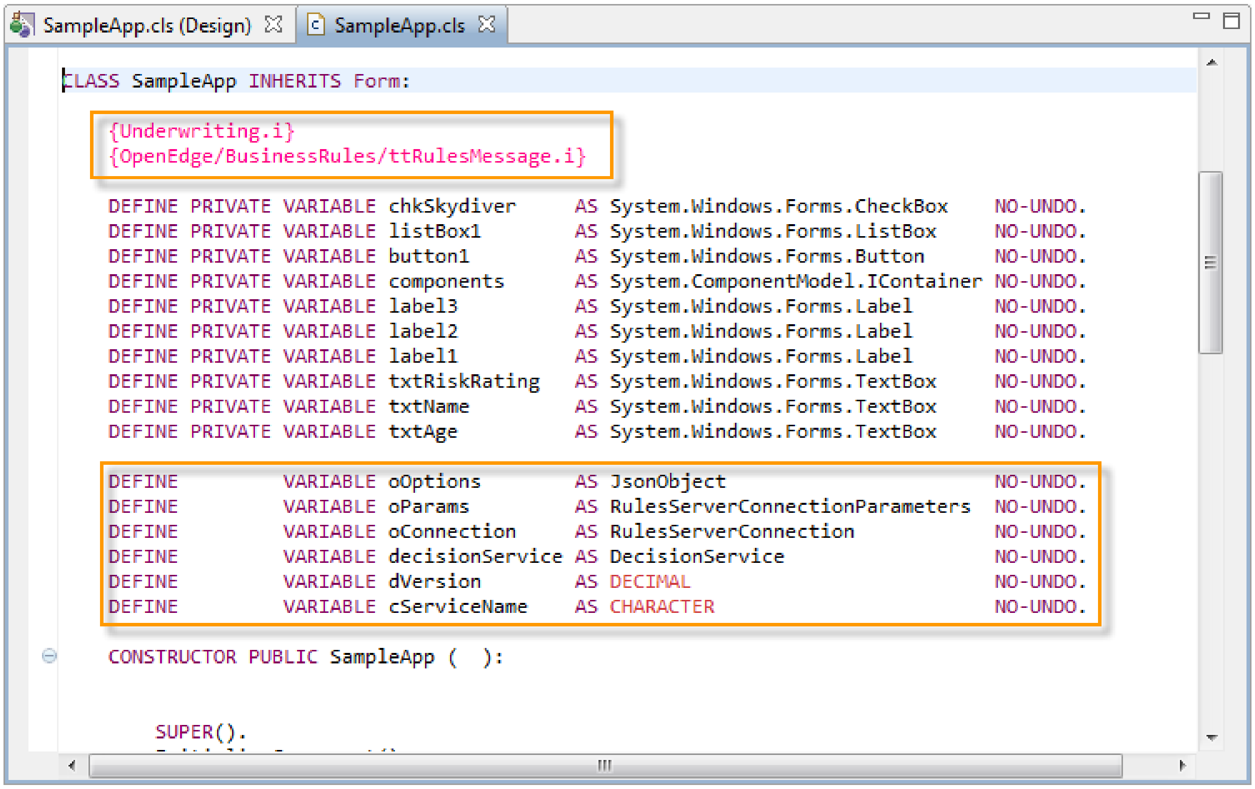
```
/*------------------------------------------------------------------------
   File        : NewPolicyForm
   Purpose     : Assess applicant risk
   Syntax      :
   Description :
   Author(s)   : Progress
   Created     : Wed 29th Jan 13:12:08 CET 2013
   Notes       :
 ----------------------------------------------------------------------*/

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.

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.

    CONSTRUCTOR PUBLIC SampleApp (  ):

        SUPER().
        InitializeComponent().
```

Figure 39

Step 3 – Add Variable Definitions

Next you will add the reference to the Include files and Variable definitions as show in Figure 40. 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().
```

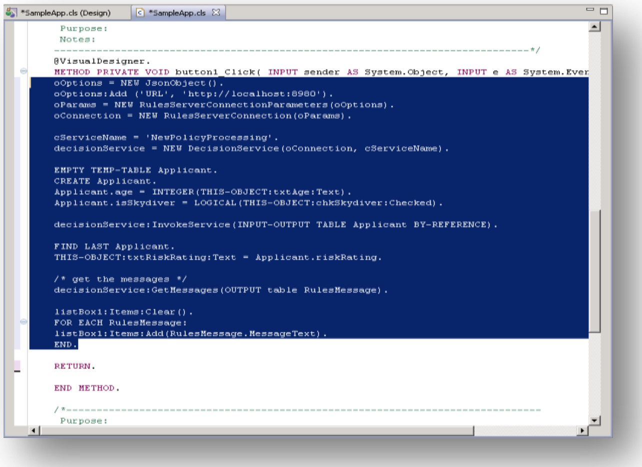
Figure 40

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 41).

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:8980').
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.Ever
   oOptions = NEW JsonObject().
   oOptions:Add ('URL', 'http://localhost:8980').
   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.

   RETURN.

   END METHOD.

   /*------------------------------------------------------------------------------
     Purpose:
```

Figure 41

Don't forget to save!

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 42). This will execute your app at the form will be shown (Figure 43).

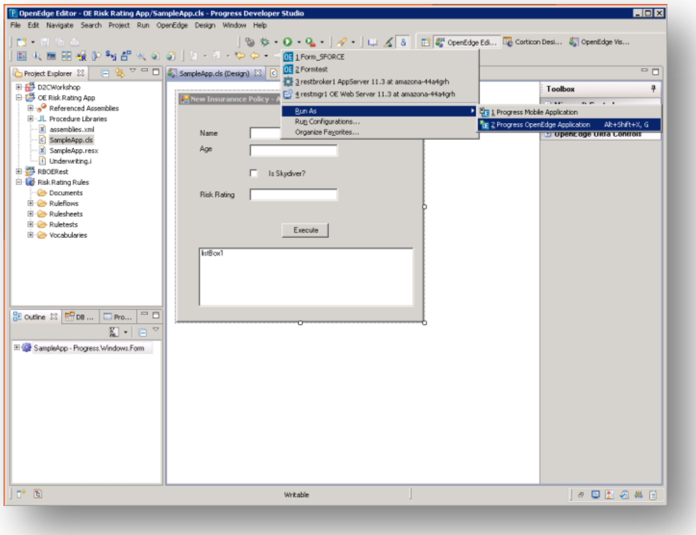
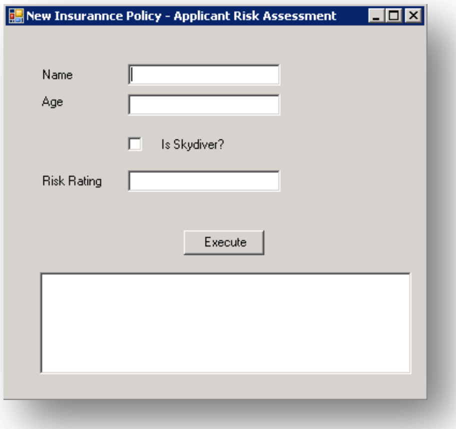


Figure 42

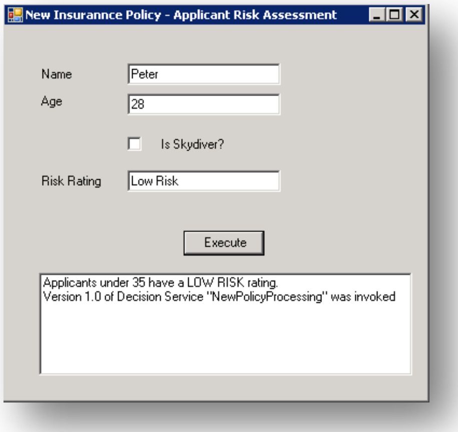
Input values for Name, Age, Skydiver and click the Execute button. Based up the values you provided you will see output similar to Figure 44. You can change the values and Execute again to get various answers from the decision.



New Insurannce Policy - Applicant Risk Assessment

Name
Age
□ Is Skydiver?
Risk Rating

Execute



New Insurannce Policy - Applicant Risk Assessment

Name: Peter
Age: 28
□ Is Skydiver?
Risk Rating: Low Risk

Execute

Applicants under 35 have a LOW RISK rating.
Version 1.0 of Decision Service "NewPolicyProcessing" was invoked

Figure 43 **Figure 44**

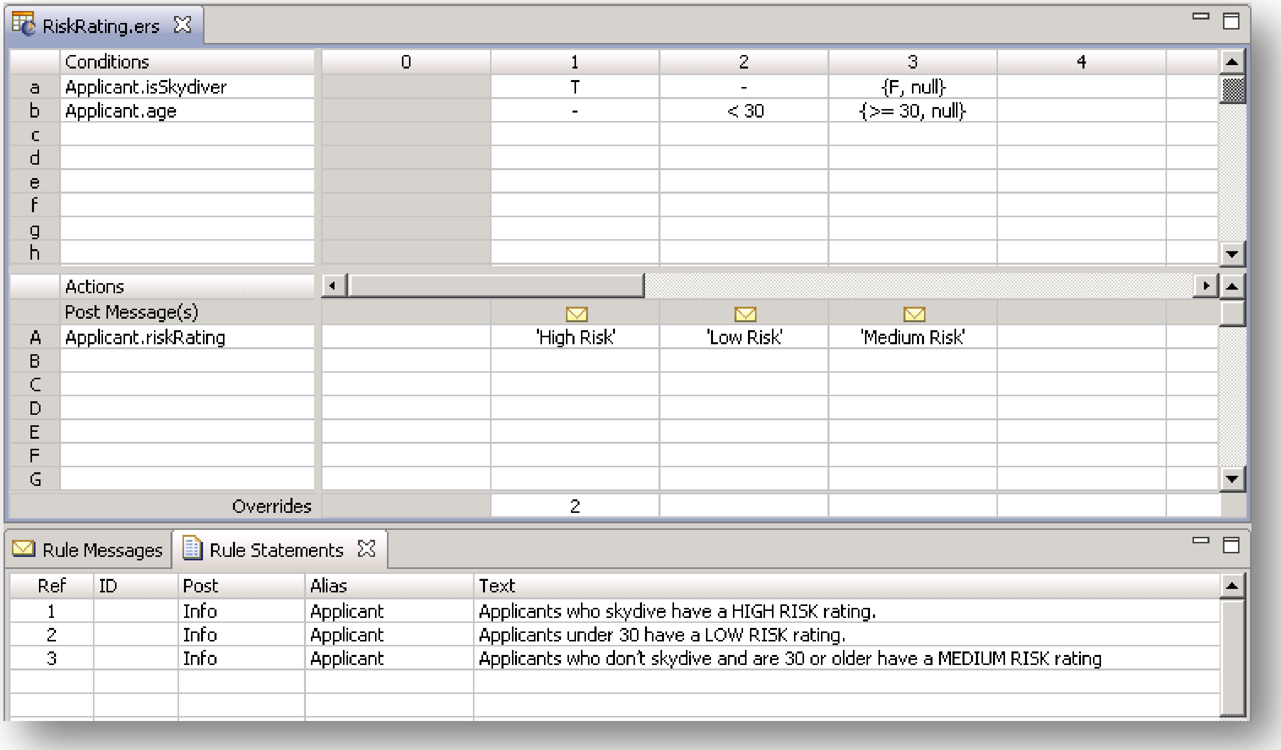
Congratulations! You have a working OE application integrated with the Corticon rules engine!

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 OpenEdge Developer 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 45).

Don't forget to save!!

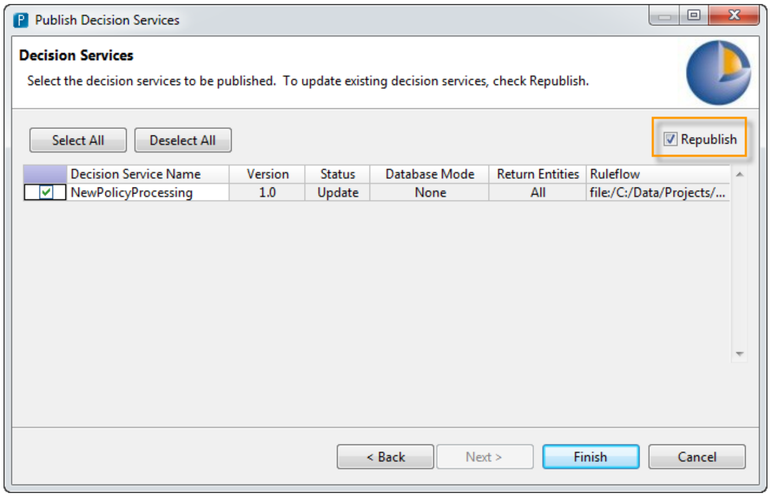


	Conditions	0	1	2	3	4
a	Applicant.isSkydiver		T	-	{F, null}	
b	Applicant.age		-	< 30	{>= 30, null}	
c						
d						
e						
f						
g						
h						
	Actions					
	Post Message(s)		✉	✉	✉	
A	Applicant.riskRating		'High Risk'	'Low Risk'	'Medium Risk'	
B						
C						
D						
E						
F						
G						
	Overrides		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 who don't skydive and are 30 or older have a MEDIUM RISK rating

Figure 45

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 46).



Decision Services

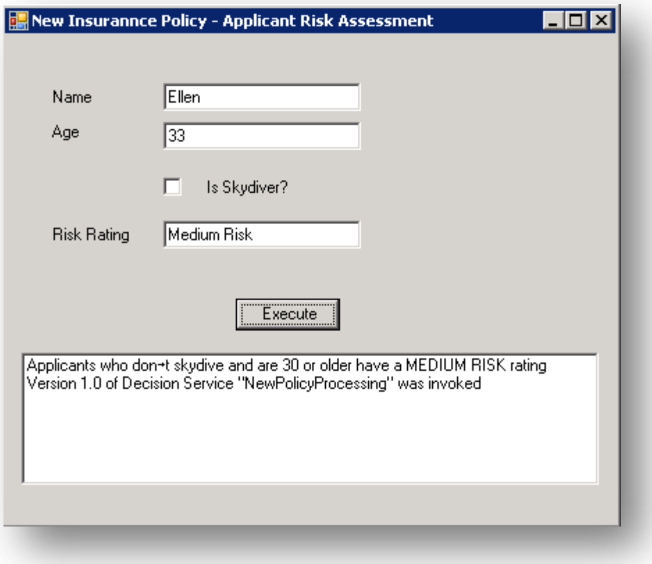
Select the decision services to be published. To update existing decision services, check Republish.

☑ Republish

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

Figure 46

Finally, click the execute button on your OpenEdge app and see the new results (Figure 47).



Name: Ellen

Age: 33

□ Is Skydiver?

Risk Rating: Medium Risk

```
Applicants who don't skydive and are 30 or older have a MEDIUM RISK rating
Version 1.0 of Decision Service "NewPolicyProcessing" was invoked
```

Figure 47

As you can see, changing the rules is really easy and you don’t have to touch a single line of OE code!

Bonus activities

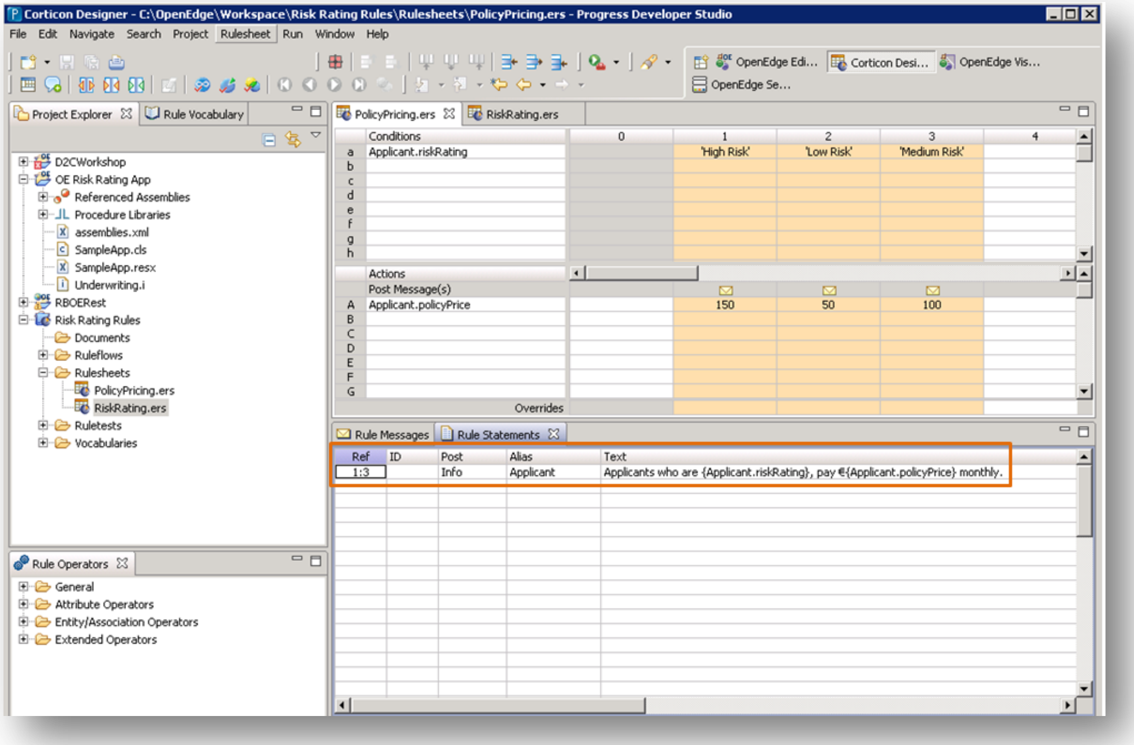
If you complete the **basic lab exercises** before the allotted time, you may wish to look at some addition examples of functionality.

Exercise 1 – Add new Rulesheets to Ruleflow

The current Ruleflow has only one Rulesheet. Real world decisions have many Rulesheets. Your Insurance Company has decided to not only assess the Applicant's risk, but also calculate a monthly premium for the new policy. Let's add this calculation to a new rulesheet. Doing it in a new rulesheet allows us to segregate rules which promote ease of maintenance and re-use of rules in other ruleflows.

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. Note that in the rule statements we have done some interesting things.

1. We have parameterized the rule statements with actual values passed into the rules engine as well as output calculated values. Just drag and drop them from the vocabulary into the rule statement.
2. One rule statement refers to multiple rules using a rule range in the Ref field. You can also refer to individual rules using the comma separator (depending on your locale).

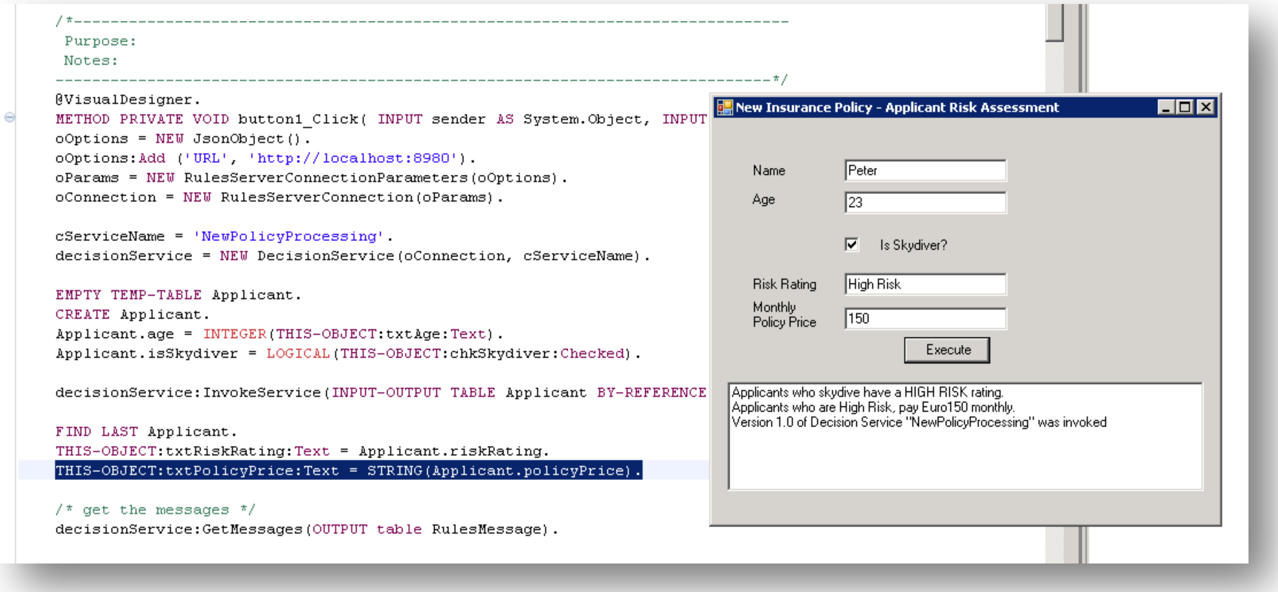


	Conditions	0	1	2	3	4
a	Applicant.riskRating		'High Risk'	'Low Risk'	'Medium Risk'	
b						
c						
d						
e						
f						
g						
h						
	Actions					
	Post Message(s)		✉	✉	✉	
A	Applicant.policyPrice		150	50	100	
B						
C						
D						
E						
F						
G						
	Overrides					

Ref	ID	Post	Alias	Text
1:3		Info	Applicant	Applicants who are {Applicant.riskRating}, pay €{Applicant.policyPrice} monthly.

Figure 48

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. If you make a few adjustments to your OE app, the new policy price will be displayed in the screen (Figure 49).



```
/*------------------------------------------------------------------------------
 Purpose:
 Notes:
----------------------------------------------------------------------------*/
@VisualDesigner.
METHOD PRIVATE VOID button1_Click( INPUT sender AS System.Object, INPUT
oOptions = NEW JsonObject().
oOptions:Add ('URL', 'http://localhost:8980').
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.
THIS-OBJECT:txtPolicyPrice:Text = STRING(Applicant.policyPrice).

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

New Insurance Policy - Applicant Risk Assessment

Name: Peter

Age: 23

☑ Is Skydiver?

Risk Rating: High Risk

Monthly Policy Price: 150

```
Applicants who skydive have a HIGH RISK rating.
Applicants who are High Risk, pay Euro150 monthly.
Version 1.0 of Decision Service "NewPolicyProcessing" was invoked
```

Figure 49

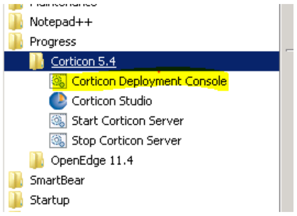
Exercise 2 – Consuming the Decision Service with soapUI

Your company has asked you to integrate the **NewPolicyProcessing** decision service with another business application. In this non OE application, you'd like to use the SOAP web services protocol to send data to Corticon and parse the response. You'll need to make sure that Corticon accepts your SOAP message and you'd like to test this before the actual integration effort. You have selected soapUI, an open source utility, to do your testing. You have already downloaded and installed the free Eclipse plugin into Progress Developer Studio.

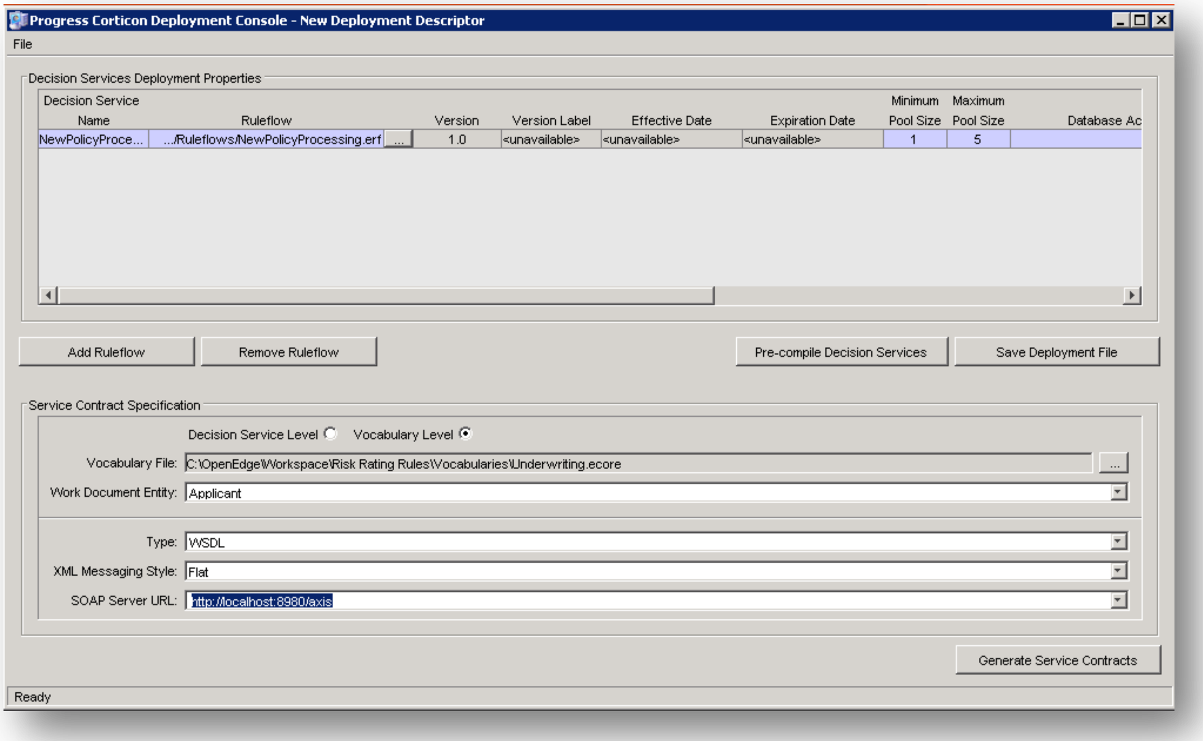
soapUI requires an integration artifact (WSDL) which details the decision service specifications. The Corticon Deployment Console (normally installed with a full, standalone Corticon Server installation) has the ability to create this for you. So let's get started!

Please follow these steps:

1. Open the Corticon Deployment Console. Go to Start Menu → All Programs → Progress → Corticon 5.4 → Deployment Console.



2. Refer to Figure 50 below. Enter in the upper pane (Decision Services Deployment Properties) the decision service name and ruleflow path ("C:\OpenEdge\Workspace\Risk Rating Rules\Ruleflows\NewPolicyProcessing.erf"). In the lower pane (Service Contract Specification):
    a. select "Vocabulary Level" - we'll create a **generic** WSDL for any ruleflow pointing to the used vocabulary
    b. select the Work Document Entity – the driving entity for our service
    c. select Type **WSDL** in the drop down (XML schema file or XSD is not used in this exercise)
    d. make sure the URL is pointing to **http://localhost:8980/axis**
    e. Hit the Generate Service Contracts button and place the file on your desktop



Progress Corticon Deployment Console - New Deployment Descriptor

File

Decision Services Deployment Properties

Decision Service Name	Ruleflow	Version	Version Label	Effective Date	Expiration Date	Minimum Pool Size	Maximum Pool Size	Database Ac
NewPolicyProce...	.../Ruleflows/NewPolicyProcessing.erf	1.0	<unavailable>	<unavailable>	<unavailable>	1	5	

Service Contract Specification

Decision Service Level ○ Vocabulary Level ●

Vocabulary File: C:\OpenEdge\Workspace\Risk Rating Rules\Vocabularies\Underwriting.ecore

Work Document Entity: Applicant

Type: WSDL

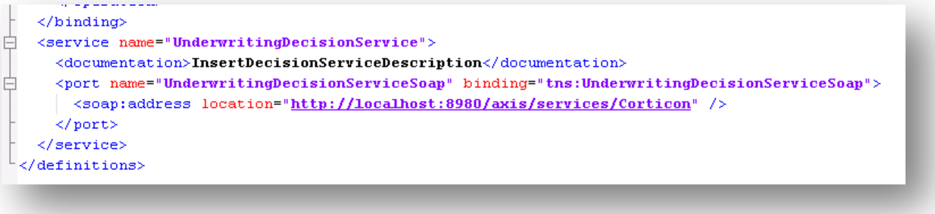
XML Messaging Style: Flat

SOAP Server URL: http://localhost:8980/axis

Ready

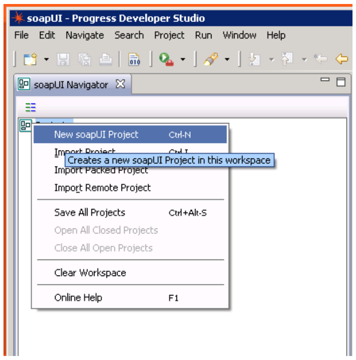
Figure 50: Corticon Deployment Console

3. Open the generated WSDL on your desktop with NotePad++ (just double click it). Insert the proper decision service name at the end and save the file.



```
    </binding>
    <service name="UnderwritingDecisionService">
      <documentation>InsertDecisionServiceDescription</documentation>
      <port name="UnderwritingDecisionServiceSoap" binding="tns:UnderwritingDecisionServiceSoap">
        <soap:address location="http://localhost:8980/axis/services/Corticon" />
      </port>
    </service>
  </definitions>
```

Figure 51: WSDL file - insertion of decision service name

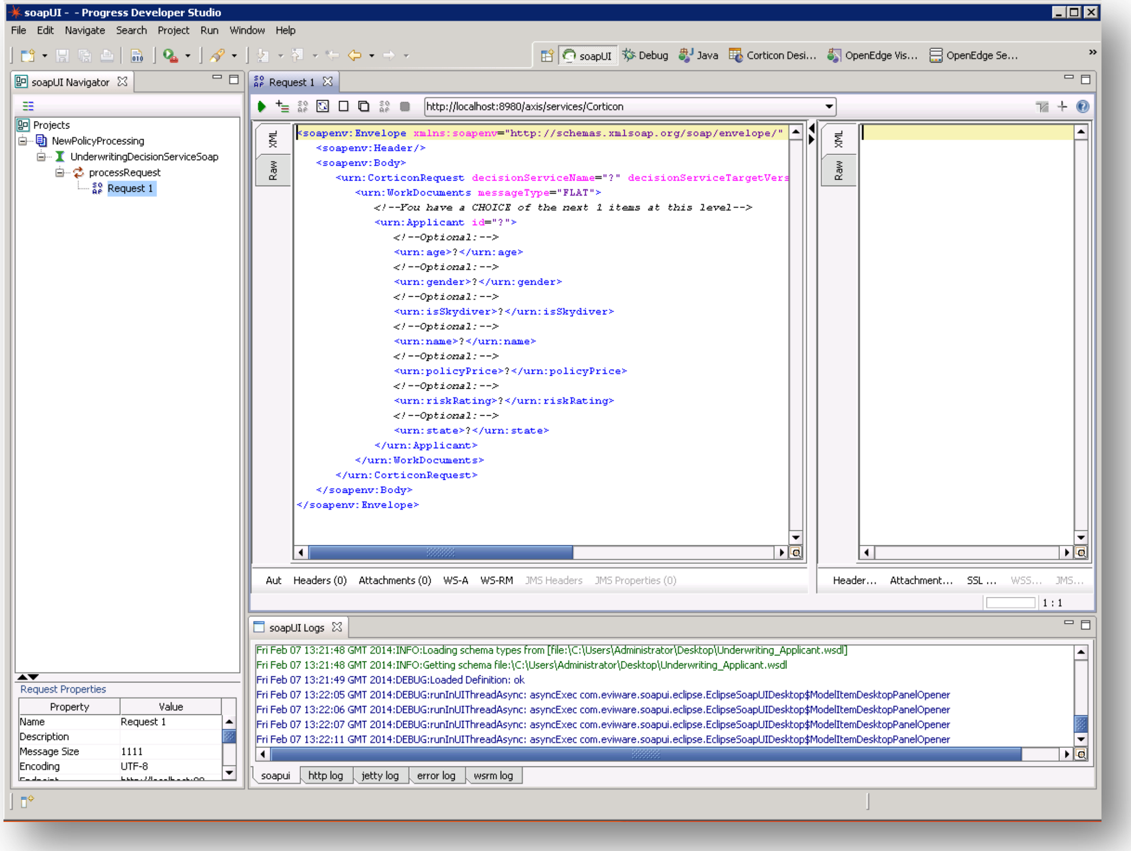


soapUI - Progress Developer Studio
File Edit Navigate Search Project Run Window Help
soapUI Navigator
New soapUI Project Ctrl-N
Import Project Ctrl-I
Creates a new soapUI Project in this workspace
Import Packed Project
Import Remote Project
Save All Projects Ctrl+Alt-S
Open All Closed Projects
Close All Open Projects
Clear Workspace
Online Help F1

4. Open the soapUI perspective in Progress Developer Studio.

5. Create a new soapUI project by right mouse clicking on Projects in the soapUI pane.

Enter the Project Name and Initial WSDL file link and press OK. Open the **Request1** in the tree structure in the left pane (double click **Request1**). The request payload will open. As you can see, some of inputs are missing.


soapUI - - Progress Developer Studio
File Edit Navigate Search Project Run Window Help
soapUI Debug Java Corticon Desi... OpenEdge Vis... OpenEdge Se...
soapUI Navigator
Projects
NewPolicyProcessing
UnderwritingDecisionServiceSoap
processRequest
Request 1
XML
Raw
Aut Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Properties (0)
Header... Attachment... SSL ... WSS... JMS...
1 : 1
soapUI Logs
soapui http log jetty log error log wsrm log

Request 1

http://localhost:8980/axis/services/Corticon

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   <soapenv:Header/>
   <soapenv:Body>
      <urn:CorticonRequest decisionServiceName="?" decisionServiceTargetVers
         <urn:WorkDocuments messageType="FLAT">
            <!--You have a CHOICE of the next 1 items at this level-->
            <urn:Applicant id="?">
               <!--Optional:-->
               <urn:age>?</urn:age>
               <!--Optional:-->
               <urn:gender>?</urn:gender>
               <!--Optional:-->
               <urn:isSkydiver>?</urn:isSkydiver>
               <!--Optional:-->
               <urn:name>?</urn:name>
               <!--Optional:-->
               <urn:policyPrice>?</urn:policyPrice>
               <!--Optional:-->
               <urn:riskRating>?</urn:riskRating>
               <!--Optional:-->
               <urn:state>?</urn:state>
            </urn:Applicant>
         </urn:WorkDocuments>
      </urn:CorticonRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

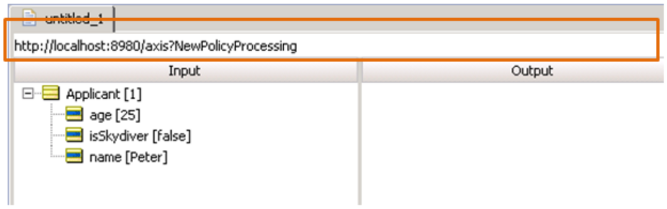
Request Properties

Property	Value
Name	Request 1
Description	
Message Size	1111
Encoding	UTF-8

```
Fri Feb 07 13:21:48 GMT 2014:INFO:Loading schema types from [file:\C:\Users\Administrator\Desktop\Underwriting_Applicant.wsdl]
Fri Feb 07 13:21:48 GMT 2014:INFO:Getting schema file:\C:\Users\Administrator\Desktop\Underwriting_Applicant.wsdl
Fri Feb 07 13:21:49 GMT 2014:DEBUG:Loaded Definition: ok
Fri Feb 07 13:22:05 GMT 2014:DEBUG:runInUIThreadAsync: asyncExec com.eviware.soapui.eclipse.EclipseSoapUIDesktop$ModelItemDesktopPanelOpener
Fri Feb 07 13:22:06 GMT 2014:DEBUG:runInUIThreadAsync: asyncExec com.eviware.soapui.eclipse.EclipseSoapUIDesktop$ModelItemDesktopPanelOpener
Fri Feb 07 13:22:07 GMT 2014:DEBUG:runInUIThreadAsync: asyncExec com.eviware.soapui.eclipse.EclipseSoapUIDesktop$ModelItemDesktopPanelOpener
Fri Feb 07 13:22:11 GMT 2014:DEBUG:runInUIThreadAsync: asyncExec com.eviware.soapui.eclipse.EclipseSoapUIDesktop$ModelItemDesktopPanelOpener
```

Figure 52: soapUI - decision service invocation

6. Instead of filling in the "?" or wiping them out, let's create the payload in Corticon Studio and copy it into soapUI. Reopen your Corticon Designer perspective and make sure that your test subject point to the Corticon Server end point.



http://localhost:8980/axis?NewPolicyProcessing

Applicant [1]
- age [25]
- isSkydiver [false]
- name [Peter]

7. Export the Request SOAP message by saving it to your Desktop. See Figure 53.

8. Open the SOAP Request with Notepad ++ and add the correct decision service name, and save. Copy the content of the entire message into the request message pane of soapUI. Effectively replacing the sample message generated by soapUI.

9. Launch the request by hitting the **green** button in the request pane. Did you see results like in Figure 54?

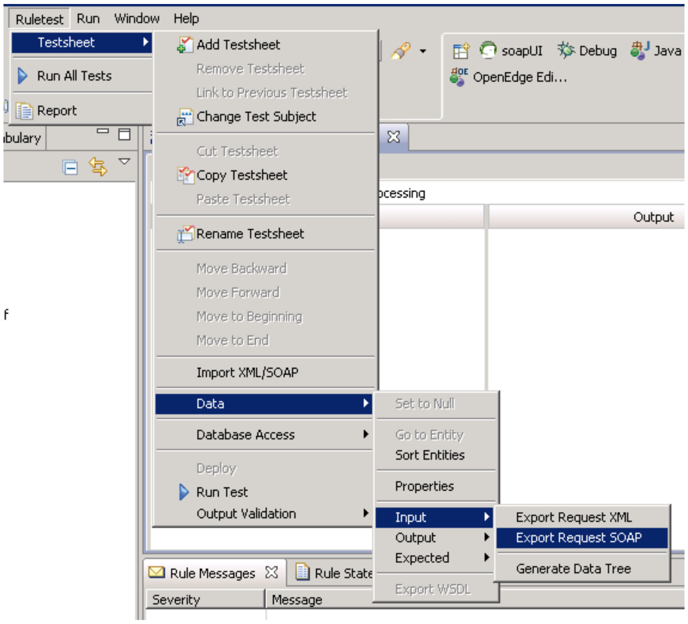
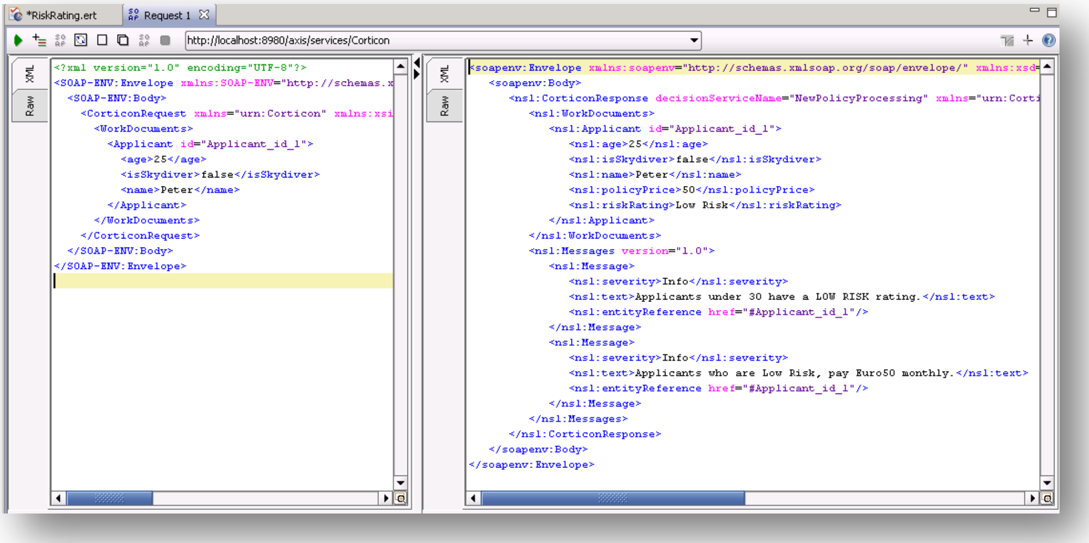


Figure 53: Exporting the request message



```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.x
  <SOAP-ENV:Body>
    <CorticonRequest xmlns="urn:Corticon" xmlns:xsi
      <WorkDocuments>
        <Applicant id="Applicant_id_1">
          <age>25</age>
          <isSkydiver>false</isSkydiver>
          <name>Peter</name>
        </Applicant>
      </WorkDocuments>
    </CorticonRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd=
   <soapenv:Body>
      <nsl:CorticonResponse decisionServiceName="NewPolicyProcessing" xmlns="urn:Corti
         <nsl:WorkDocuments>
            <nsl:Applicant id="Applicant_id_1">
               <nsl:age>25</nsl:age>
               <nsl:isSkydiver>false</nsl:isSkydiver>
               <nsl:name>Peter</nsl:name>
               <nsl:policyPrice>50</nsl:policyPrice>
               <nsl:riskRating>Low Risk</nsl:riskRating>
            </nsl:Applicant>
         </nsl:WorkDocuments>
         <nsl:Messages version="1.0">
            <nsl:Message>
               <nsl:severity>Info</nsl:severity>
               <nsl:text>Applicants under 30 have a LOW RISK rating.</nsl:text>
               <nsl:entityReference href="#Applicant_id_1"/>
            </nsl:Message>
            <nsl:Message>
               <nsl:severity>Info</nsl:severity>
               <nsl:text>Applicants who are Low Risk, pay Euro50 monthly.</nsl:text>
               <nsl:entityReference href="#Applicant_id_1"/>
            </nsl:Message>
         </nsl:Messages>
      </nsl:CorticonResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 54: Results from invoking a decision service in soapUI

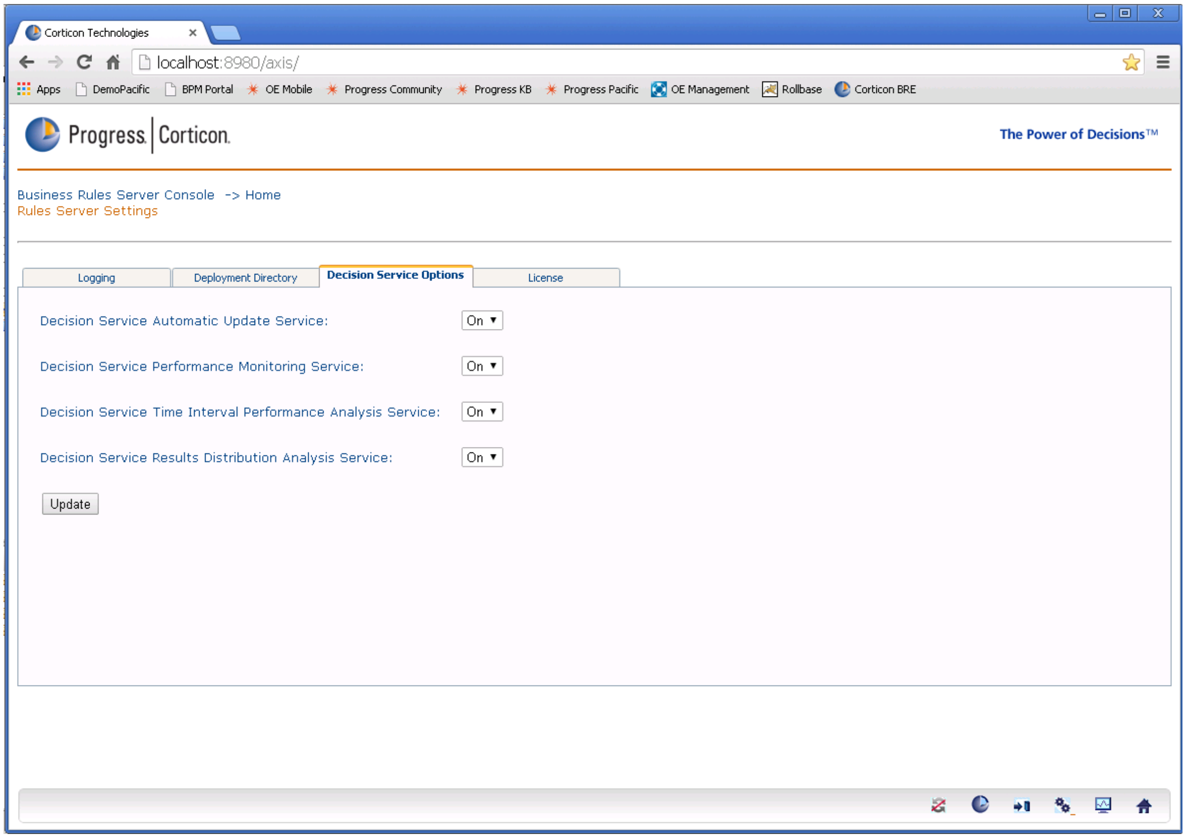
Note 1: Corticon Server returns two objects: the **WorkDocuments** and **Messages**. Both can be used to do all sorts of things (persist the data, display data in a screen, display the rule trace in a screen).

Note 2: soapUI can be used to perform web services testing. But also for load/performance testing it is a great tool.

Exercise 3 – Configure Server Console for Monitoring

Your management has requested better insight in the decisions being taken by the rule server. Corticon Server is able to monitor the values that are passing through the engine or being generated as the result of rules processing. All these statistics can be requested by API for integration with backend monitoring tools or business intelligence solutions. But the information is also displayed in the Corticon Server console!

Click “Configure Rules Server” icon from toolbar. Ensure all options are “Yes” under Decision Service Options (Figure 55). Go to list of Decision Services and select NewPolicyProcessing. Drill down to Service Configuration and add Applicant.riskRating as a Monitored Attribute (Figure 56). For all future executions of this service, Corticon will capture statistics for the attribute that you can review (figure 57, 58).



Business Rules Server Console -> Home
Rules Server Settings

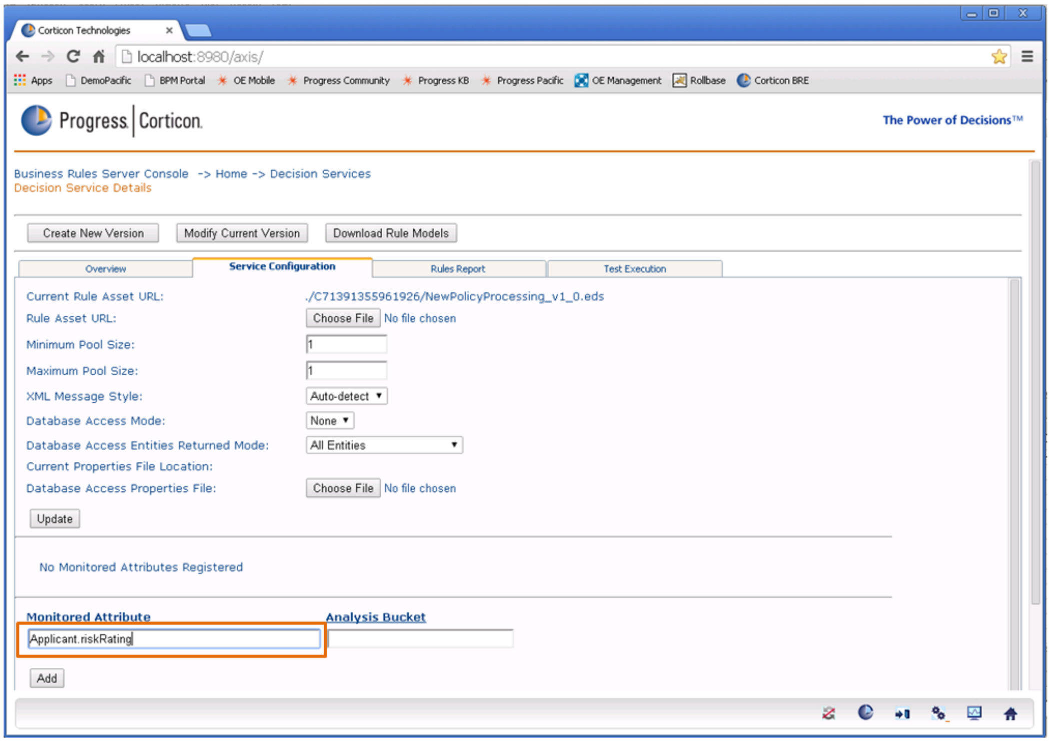
Decision Service Automatic Update Service: On

Decision Service Performance Monitoring Service: On

Decision Service Time Interval Performance Analysis Service: On

Decision Service Results Distribution Analysis Service: On

Figure 55



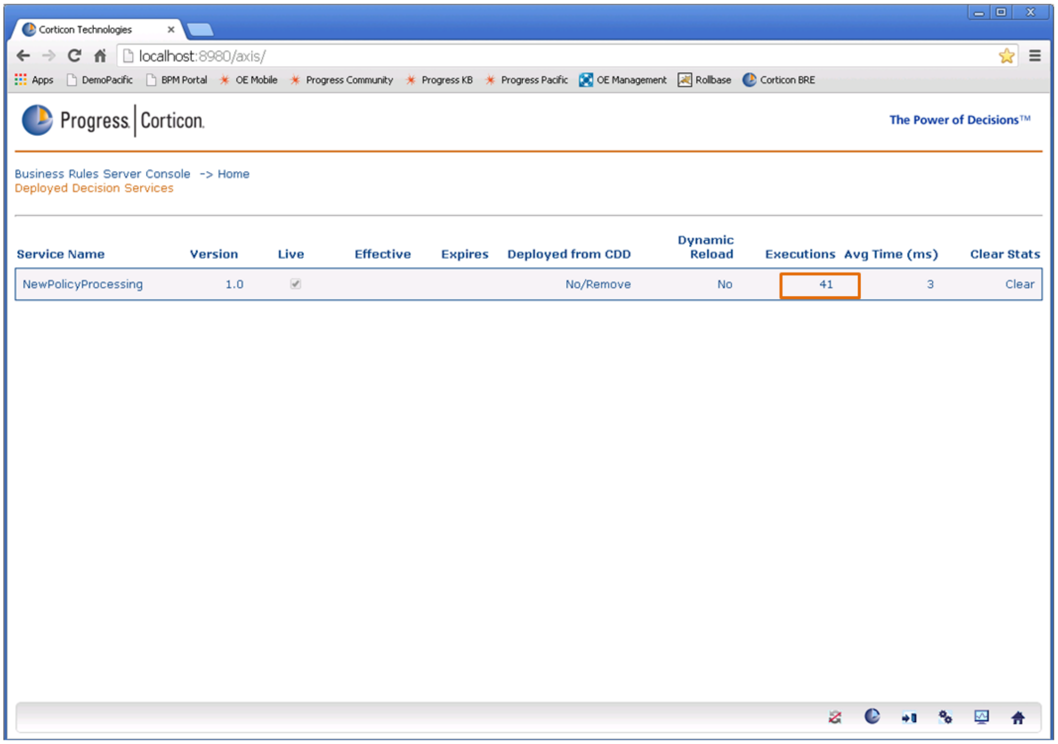
Business Rules Server Console -> Home -> Decision Services
Decision Service Details

Current Rule Asset URL:	./C71391355961926/NewPolicyProcessing_v1_0.eds
Rule Asset URL:	Choose File No file chosen
Minimum Pool Size:	1
Maximum Pool Size:	1
XML Message Style:	Auto-detect ▾
Database Access Mode:	None ▾
Database Access Entities Returned Mode:	All Entities ▾
Current Properties File Location:	
Database Access Properties File:	Choose File No file chosen

No Monitored Attributes Registered

Monitored Attribute	Analysis Bucket
Applicant.riskRating	

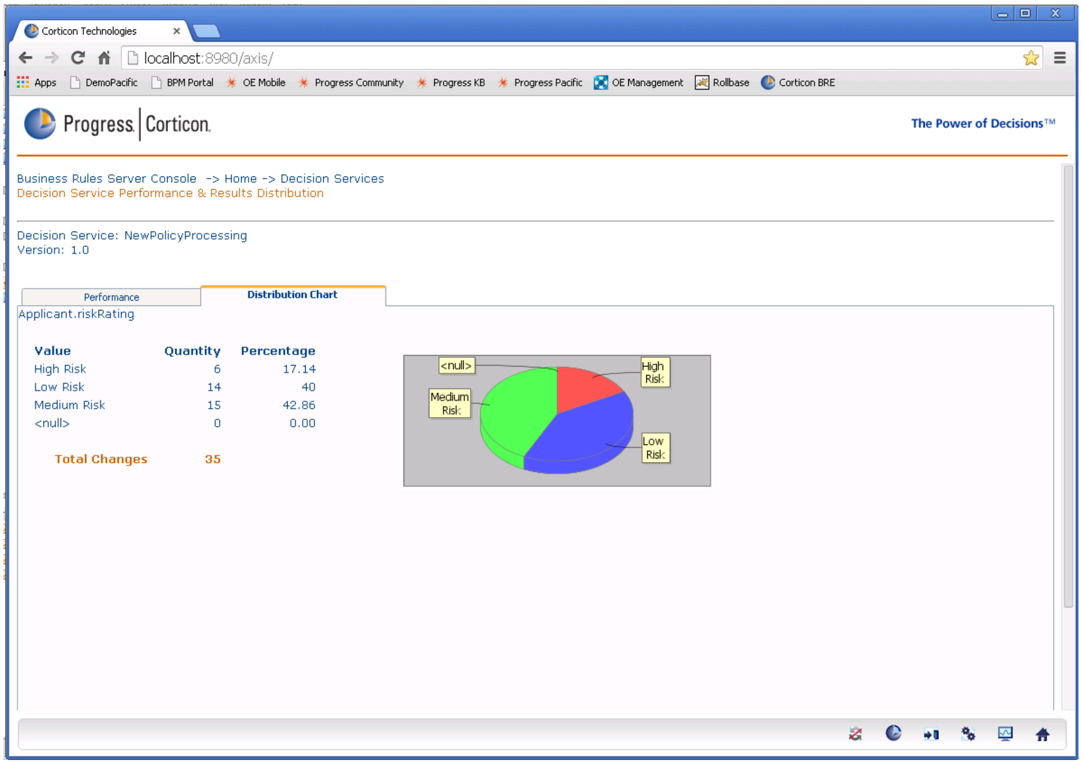
Figure 56



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
NewPolicyProcessing	1.0	✓			No/Remove	No	41	3	Clear

Figure 57



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

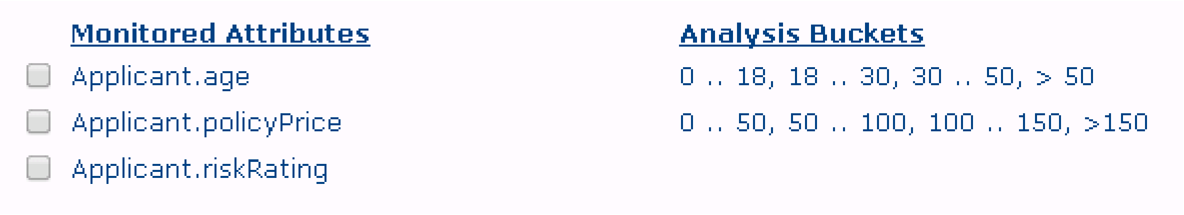
Decision Service: NewPolicyProcessing
Version: 1.0

Applicant.riskRating

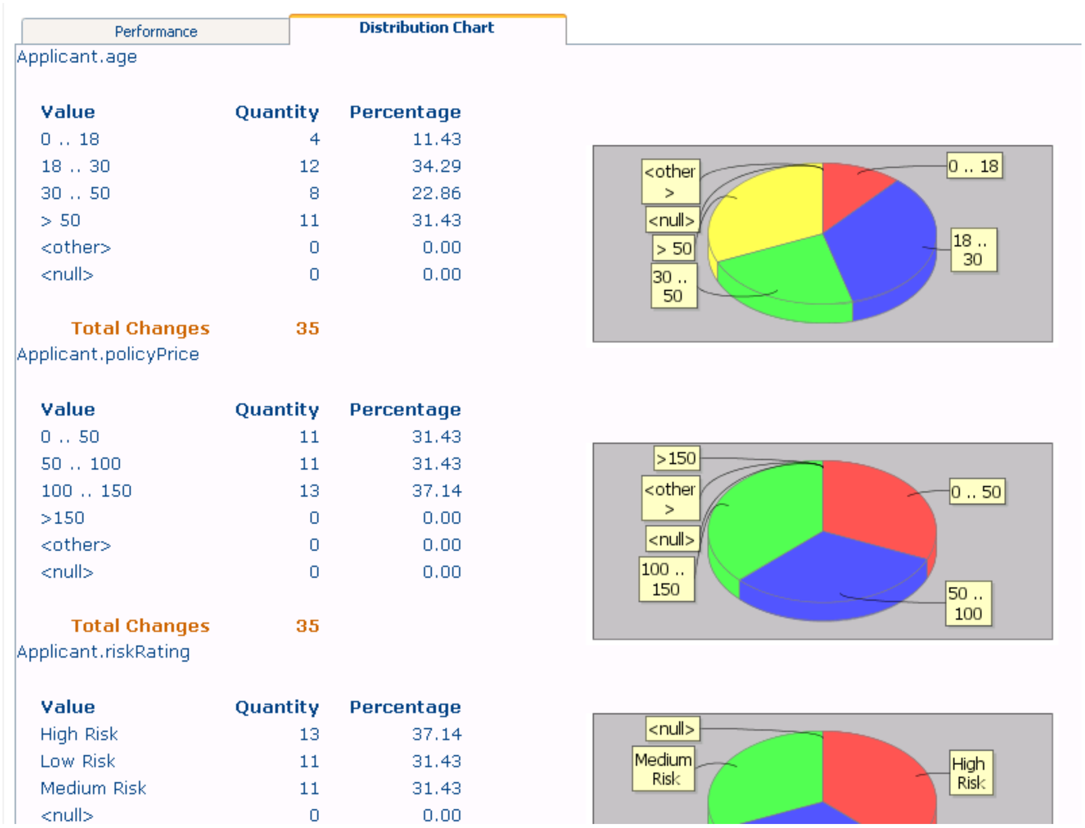
Value	Quantity	Percentage
High Risk	6	17.14
Low Risk	14	40
Medium Risk	15	42.86
<null>	0	0.00
Total Changes	35	

Figure 58

Feel free to insert other monitoring attributes.



	Monitored Attributes	Analysis Buckets
□	Applicant.age	0 .. 18, 18 .. 30, 30 .. 50, > 50
□	Applicant.policyPrice	0 .. 50, 50 .. 100, 100 .. 150, >150
□	Applicant.riskRating	



Applicant.age

Value	Quantity	Percentage
0 .. 18	4	11.43
18 .. 30	12	34.29
30 .. 50	8	22.86
> 50	11	31.43
<other>	0	0.00
<null>	0	0.00
Total Changes	35	

Applicant.policyPrice

Value	Quantity	Percentage
0 .. 50	11	31.43
50 .. 100	11	31.43
100 .. 150	13	37.14
>150	0	0.00
<other>	0	0.00
<null>	0	0.00
Total Changes	35	

Applicant.riskRating

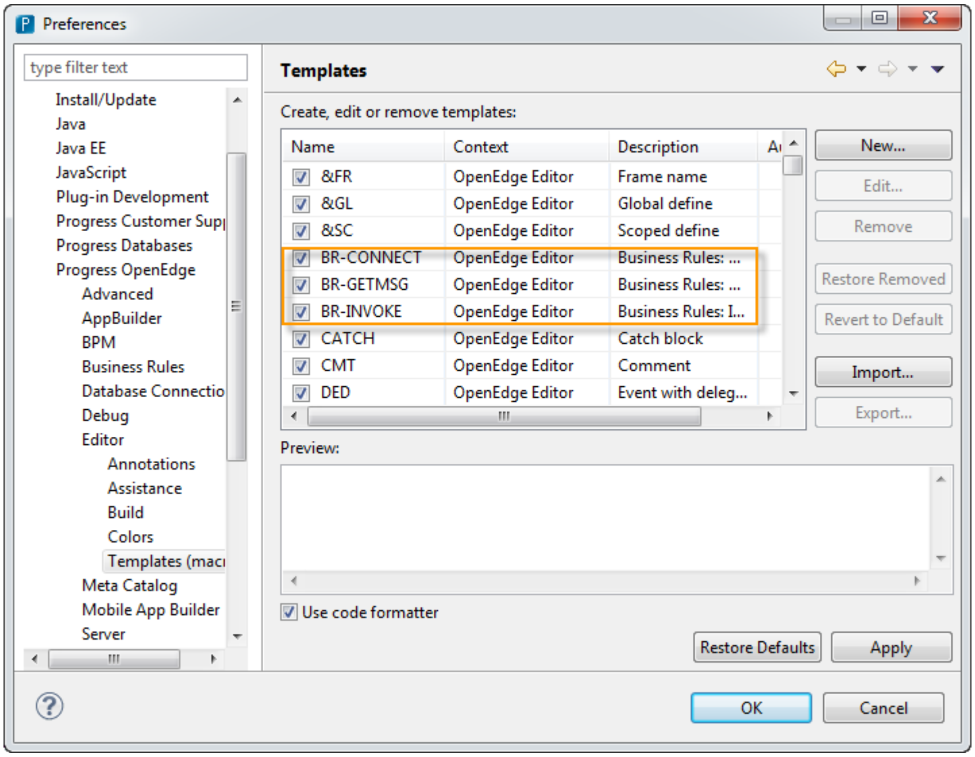
Value	Quantity	Percentage
High Risk	13	37.14
Low Risk	11	31.43
Medium Risk	11	31.43
<null>	0	0.00

Exercise 4 – Use Code Macros

Coding the integration code in Progress Developer Studio requires work. To minimize the coding effort, Progress made a number of macros available. There are 3 default macros provided to assist with code insertion.

1. BR-CONNECT
2. BR-GETMSG
3. BR-INVOKE

The templates for these are editable and found in System Preferences (Figure 59). You can experiment using these macros to automatically insert the code that you copy/pasted in the previous exercise.



Templates

Create, edit or remove templates:

	Name	Context	Description	A
☑	&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 59

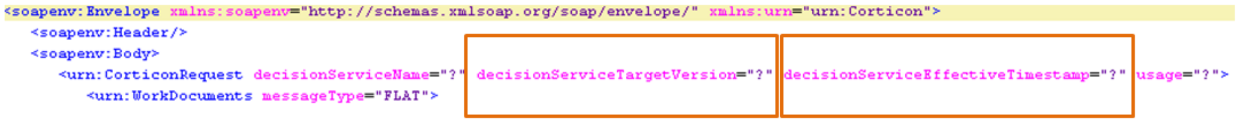
Tip: For non OE developers, just reset your perspective to OpenEdge Editor. Revisit the SampleApp.cls by going to the code window (remember F9). Type the macro name followed by a spaces, and the macro will auto-generate all the template code for you.

Exercise 5 – 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 6 – Deploy Multiple Versions

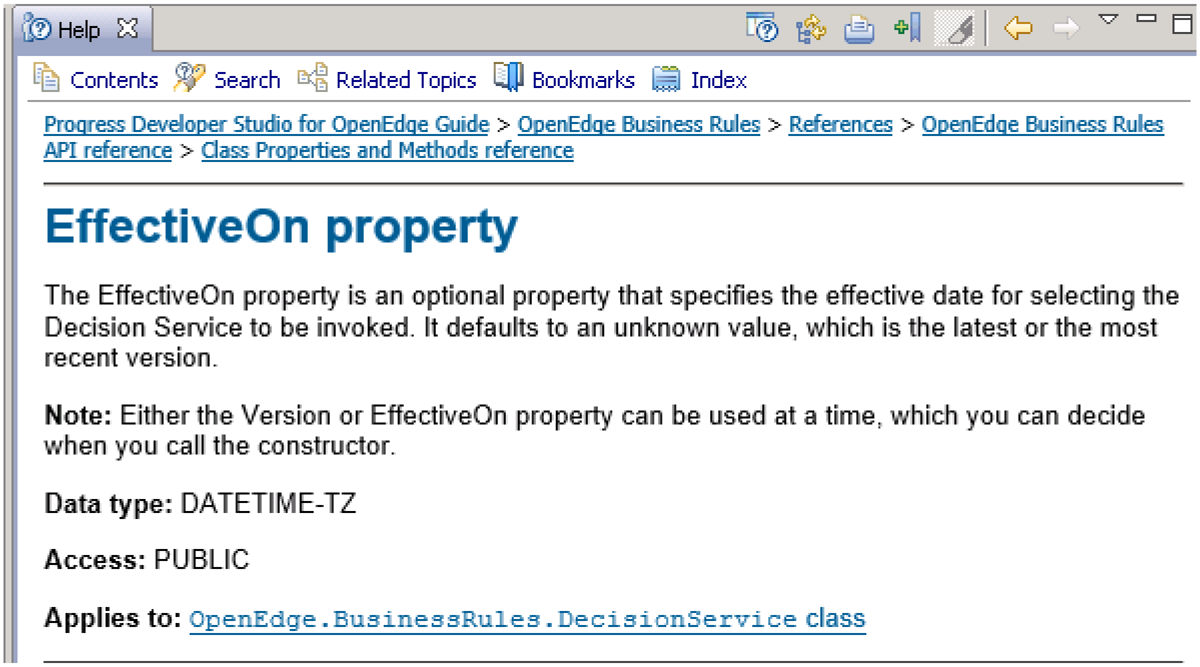
Corticon supports deployment of multiple versions of the same decision service. In practice, most organizations need this. This feature supports back dating or future dating of decision services. For example it helps conducting “what if” exercises like “What if we run against a previous or future version of a decision service?” Multiple version of the same decision service can “live”in Corticon Server. The invocation request payload parameters provide the necessary data to Corticon Server to automatically run the correct version against the request payload.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:Corticon">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:CorticonRequest decisionServiceName="?" decisionServiceTargetVersion="?" decisionServiceEffectiveTimestamp="?" usage="?">
         <urn:WorkDocuments messageType="FLAT">
```

Figure 60: Version and date parameters in the request payload

Your ABL code also supports these decision service invocation parameters.



Progress Developer Studio for OpenEdge Guide > OpenEdge Business Rules > References > OpenEdge Business Rules API reference > Class Properties and Methods reference

EffectiveOn property

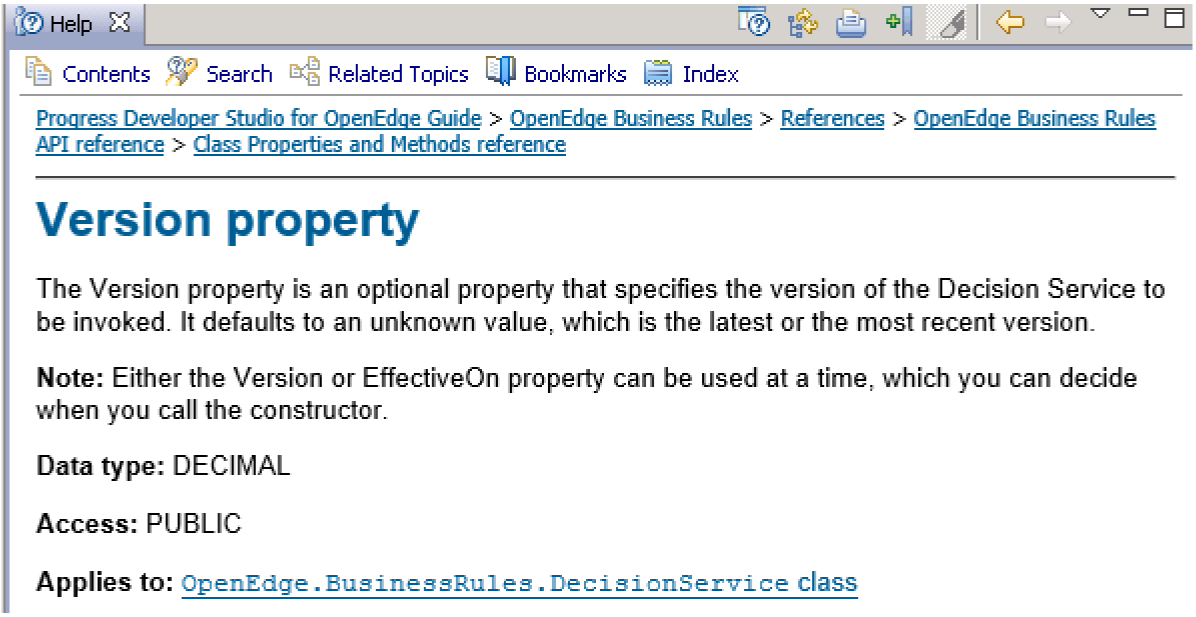
The EffectiveOn property is an optional property that specifies the effective date for selecting the Decision Service to be invoked. It defaults to an unknown value, which is the latest or the most recent version.

Note: Either the Version or EffectiveOn property can be used at a time, which you can decide when you call the constructor.

Data type: DATETIME-TZ

Access: PUBLIC

Applies to: `OpenEdge.BusinessRules.DecisionService` class



Progress Developer Studio for OpenEdge Guide > OpenEdge Business Rules > References > OpenEdge Business Rules API reference > Class Properties and Methods reference

Version property

The Version property is an optional property that specifies the version of the Decision Service to be invoked. It defaults to an unknown value, which is the latest or the most recent version.

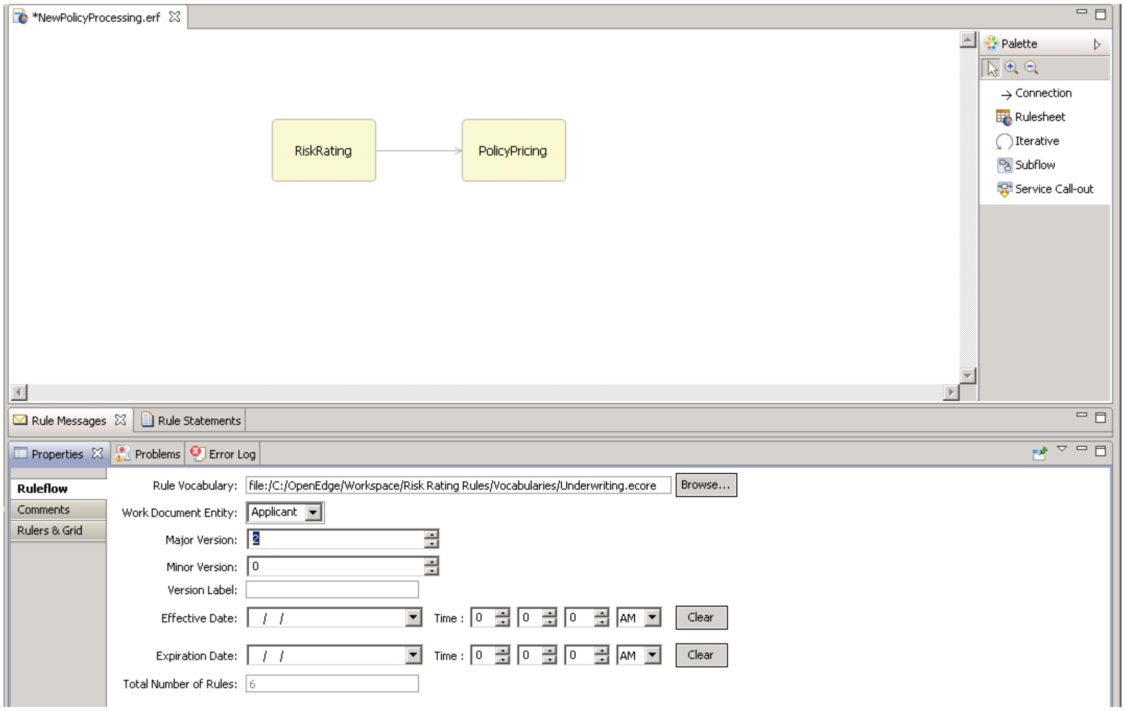
Note: Either the Version or EffectiveOn property can be used at a time, which you can decide when you call the constructor.

Data type: DECIMAL

Access: PUBLIC

Applies to: `OpenEdge.BusinessRules.DecisionService` class

To create a new version, navigate to the Properties panel for the Ruleflow (Figure 60). Update the Major Version to 2 and save the Ruleflow. Now when you publish the Ruleflow, you will see two versions deployed on the Corticon Server (Figure 61).



Rule Vocabulary: file:/C:/OpenEdge/Workspace/Risk Rating Rules/Vocabularies/Underwriting.ecore

Work Document Entity: Applicant

Major Version: 2

Minor Version: 0

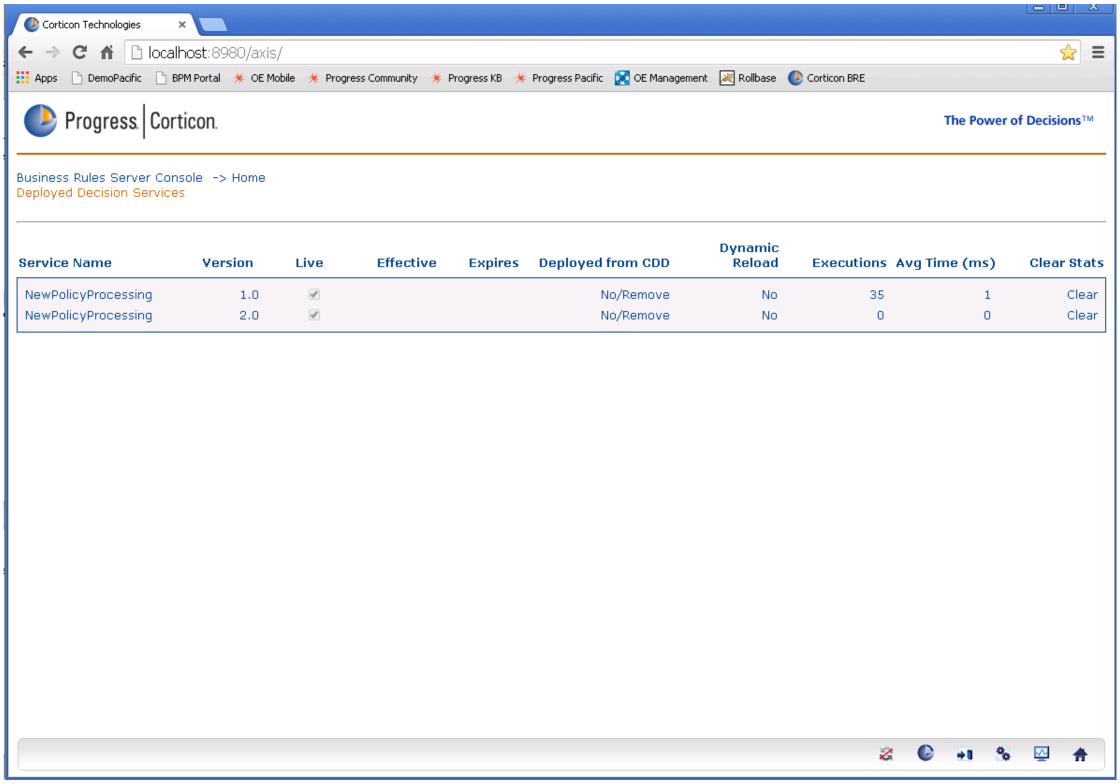
Version Label:

Effective Date: / / Time : 0 0 0 AM

Expiration Date: / / Time : 0 0 0 AM

Total Number of Rules: 6

Figure 61



Business Rules Server Console -> Home
Deployed Decision Services

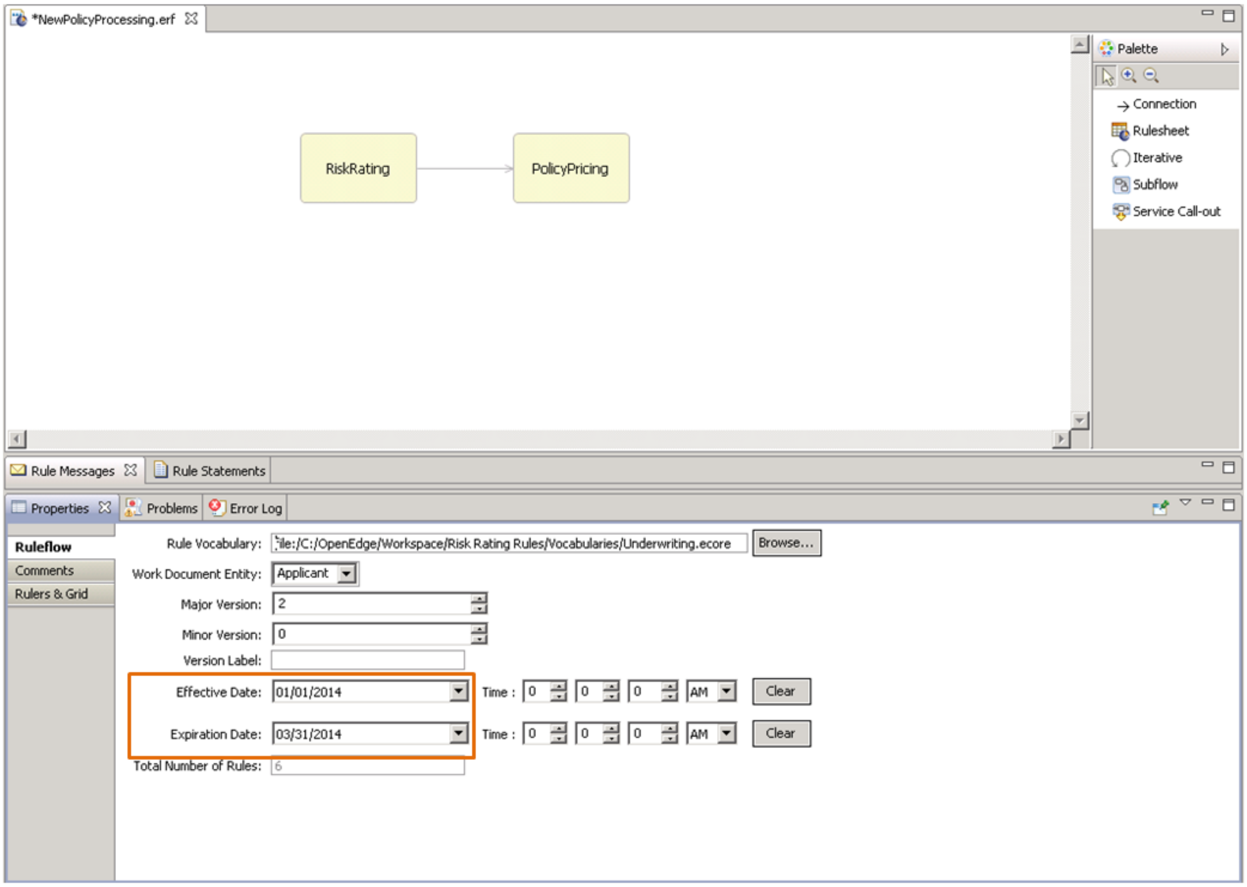
Service Name	Version	Live	Effective	Expires	Deployed from CDD	Dynamic Reload	Executions	Avg Time (ms)	Clear Stats
NewPolicyProcessing	1.0	☑			No/Remove	No	35	1	Clear
NewPolicyProcessing	2.0	☑			No/Remove	No	0	0	Clear

Figure 62

Exercise 7 – Deploy with Effective Dates

Corticon also support effective dates for versions. This means you can specify for what date ranges a certain version is applicable. Navigate to the properties for the Ruleflow and specify a start and stop date/time (Figure 62). Re-publish the decision service, and you will see that there are now effective dates for the decision service (Figure 63).

Note: The date settings in Progress Developer Studio – perspective Corticon Studio are set to US date format (MM/DD/YY). This can be changed in the Corticon configuration settings when the software is installed. We hope you don't mind we left it to a US setting?



Rule Vocabulary: ile:/C:/OpenEdge/Workspace/Risk Rating Rules/Vocabularies/Underwriting.ecore

Work Document Entity: Applicant

Major Version: 2

Minor Version: 0

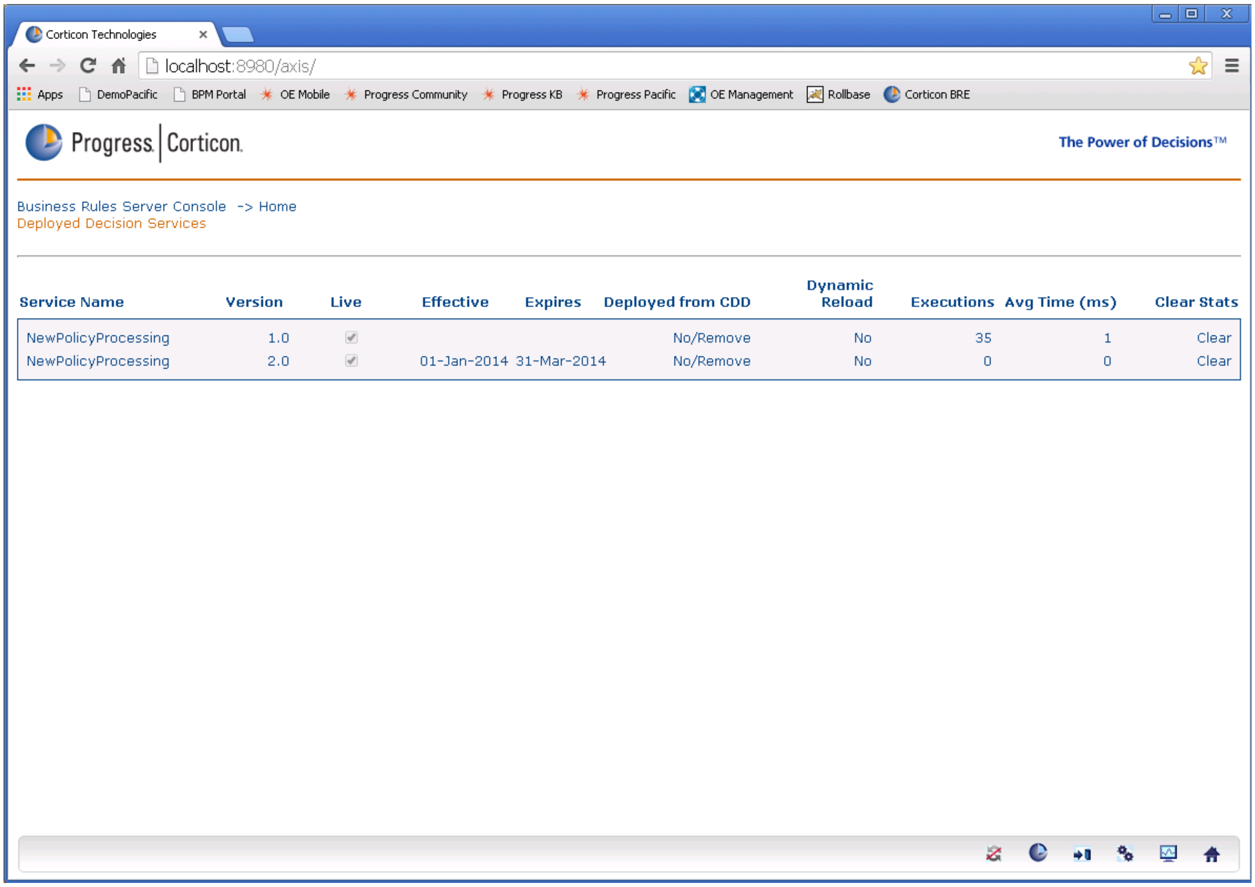
Version Label:

Effective Date: 01/01/2014 Time : 0 0 0 AM

Expiration Date: 03/31/2014 Time : 0 0 0 AM

Total Number of Rules: 6

Figure 63



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
NewPolicyProcessing	1.0	☑			No/Remove	No	35	1	Clear
NewPolicyProcessing	2.0	☑	01-Jan-2014	31-Mar-2014	No/Remove	No	0	0	Clear

Figure 64

Exercise 8 – Dynamic Database Integration with Corticon Enterprise Data Connector

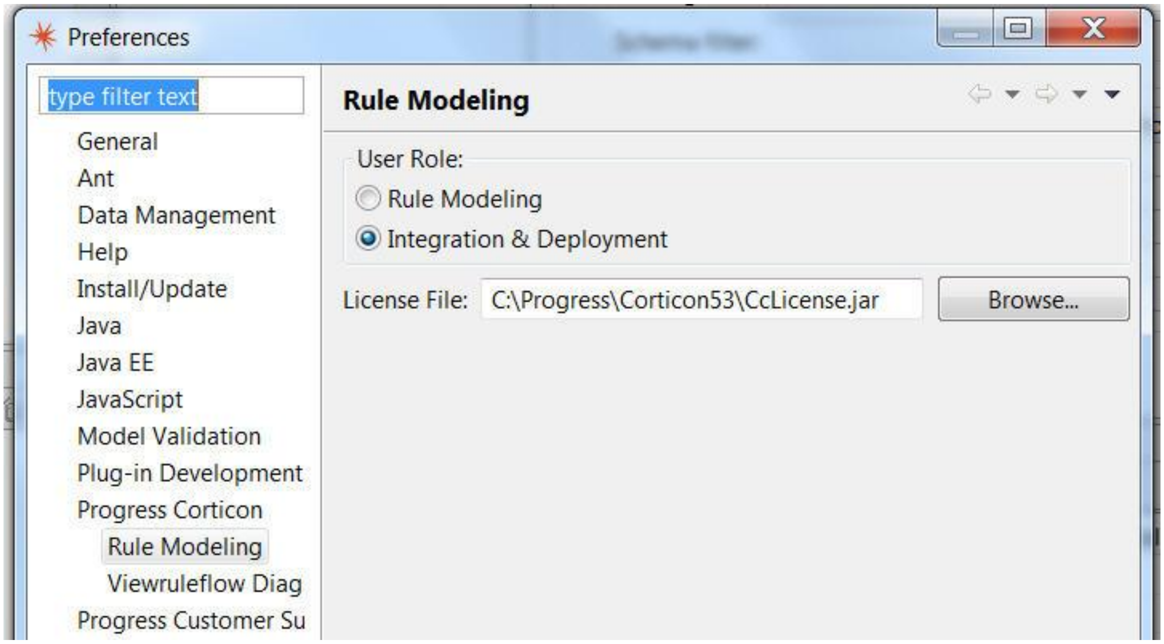
Corticon Enterprise Data Connector (EDC) provides a way for your Corticon Rulesheets to dynamically interact with external data sources during rule execution. Corticon EDC can act in Read-only mode where it is used to enrich the data provided in a request message when a Corticon decision service is invoked, as required in order to evaluate the rules that make up the decision service, but no data is updated in the external datasource and all data is returned the calling application in the response message including all data that was retrieved from an external data source and all changes to any data that were performed by the decision service. Corticon EDC can also be used in Read-Write mode which enables not only dynamic data enrichment, but also for updates to data made by the rule actions within a decision service to be dynamically updated in an external data source. Corticon EDC can also assist with the creation of an appropriate external data source schema to correspond with a Corticon vocabulary, should no suitable schema already exist, and in some situations can auto-map existing relational schema against existing vocabulary entities and attributes.

In this exercise you will:

1. enable EDC in your Progress Developer Studio environment
2. configure a connection from an Corticon vocabulary to an OpenEdge database
3. configure the persistence properties of your vocabulary's entity and attributes
4. have EDC create a new table in the Sports2000 sample OpenEdge database to represent the Applicant entity in the RiskRating Corticon vocabulary
5. populate that Applicant table with some test data
6. run TestSheets to show Read-only and Read-Write mode EDC behavior
7. configure a RuleSheet to enable batch execution
8. run TestSheets to show batch execution of a RuleSheet in Read-only & Read-Write modes

We will use the Risk Rating Rules EDC project for this Exercise. Now would be a good time to close other files and open that project in Progress Developer Studio.

1. To **enable EDC in your Progress Developer Studio environment** select Preferences from the Window menu, then select Progress Corticon->Rule modeling in the resulting dialog. You will see User Role options. Change the User role from Rule Modeling to Integration and Deployment:



Rule Modeling

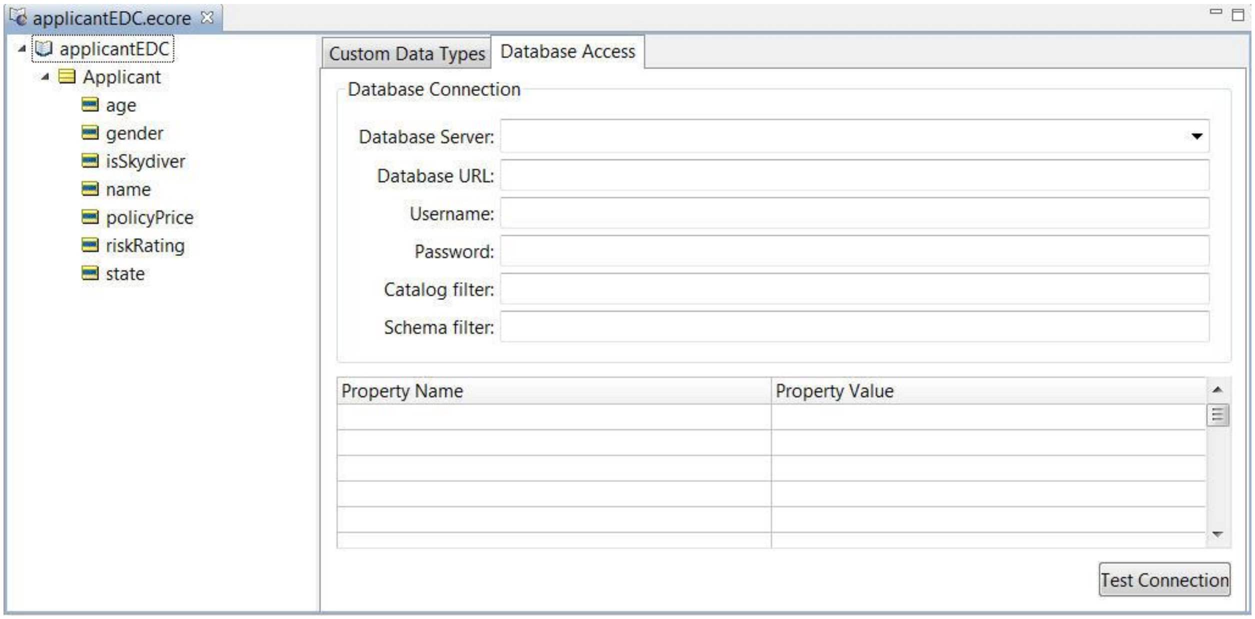
User Role:

○ Rule Modeling

● Integration & Deployment

License File: C:\Progress\Corticon53\CcLicense.jar

You can now confirm that you have access to the Corticon EDC functionality by opening the applicantEDC.ecore vocabulary and clicking on the root level node of the vocabulary. If EDC is enabled you will see an additional Database Access tab to the right of the Custom Data Types tab:



Database Connection

Database Server:

Database URL:

Username:

Password:

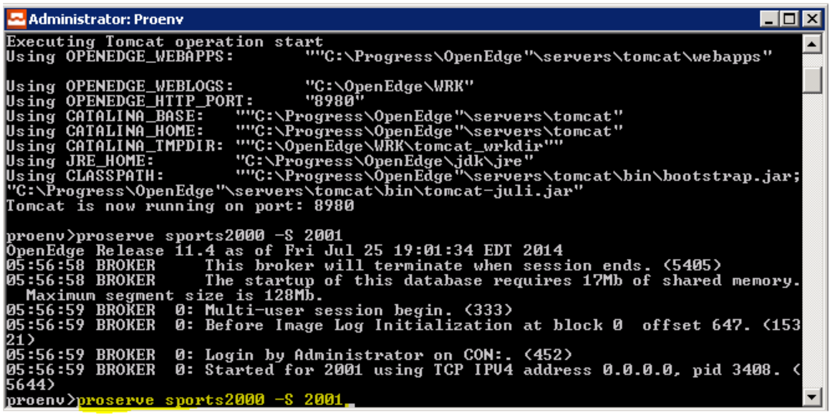
Catalog filter:

Schema filter:

Property Name	Property Value

Note that Corticon EDC is a separately licensed product capability, so you also have to specify a license file that includes the EDC feature in order to see the above Database Access tab in Progress Developer Studio. All evaluation licenses for Corticon Studio include the EDC feature. Corticon Server evaluation licenses do not include EDC by default, so if you request an evaluation license to test EDC functionality be sure that your Progress rep or SE is aware that you need access to that functionality.

2. In order to **configure a connection from an Corticon vocabulary to an OpenEdge database** you need to first start the OE Database by going to the command line and type in “preserve sports2000 –S 2001



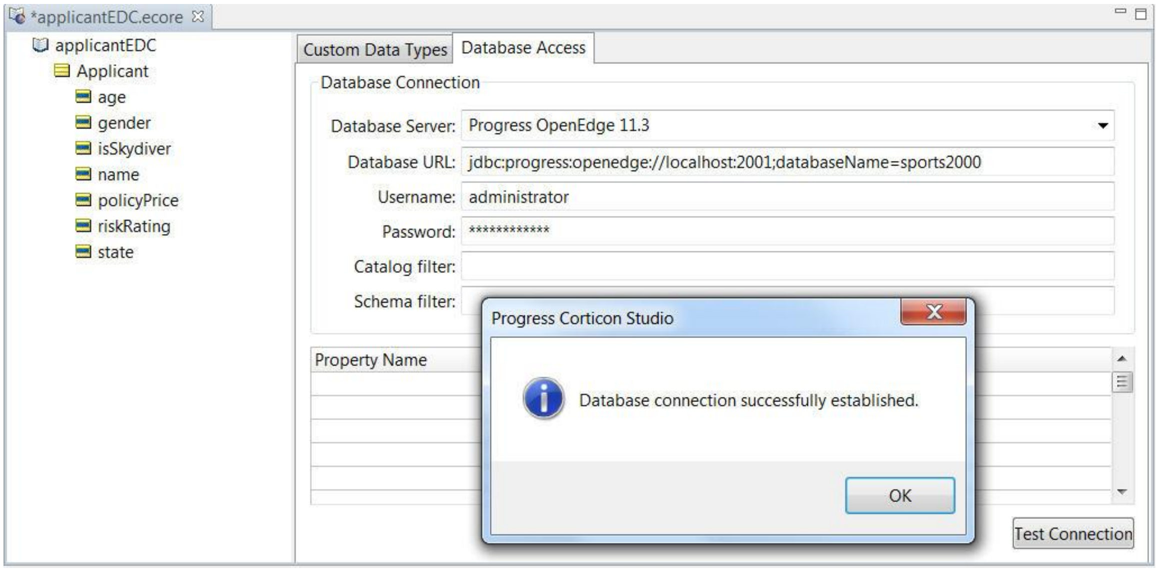
Administrator: Proenv

```
Executing Tomcat operation start
Using OPENEDGE_WEBAPPS:        ""C:\Progress\OpenEdge"\servers\tomcat\webapps"

Using OPENEDGE_WEBLOGS:        "C:\OpenEdge\WRK"
Using OPENEDGE_HTTP_PORT:      "8980"
Using CATALINA_BASE:    ""C:\Progress\OpenEdge"\servers\tomcat"
Using CATALINA_HOME:    ""C:\Progress\OpenEdge"\servers\tomcat"
Using CATALINA_TMPDIR: ""C:\OpenEdge\WRK\tomcat_wrkdir""
Using JRE_HOME:         "C:\Progress\OpenEdge\jdk\jre"
Using CLASSPATH:        ""C:\Progress\OpenEdge"\servers\tomcat\bin\bootstrap.jar;
"C:\Progress\OpenEdge"\servers\tomcat\bin\tomcat-juli.jar"
Tomcat is now running on port: 8980

proenv>proserve sports2000 -S 2001
OpenEdge Release 11.4 as of Fri Jul 25 19:01:34 EDT 2014
05:56:58 BROKER     This broker will terminate when session ends. (5405)
05:56:58 BROKER     The startup of this database requires 17Mb of shared memory.
  Maximum segment size is 128Mb.
05:56:59 BROKER  0: Multi-user session begin. (333)
05:56:59 BROKER  0: Before Image Log Initialization at block 0  offset 647. (153
21)
05:56:59 BROKER  0: Login by Administrator on CON:. (452)
05:56:59 BROKER  0: Started for 2001 using TCP IPV4 address 0.0.0.0, pid 3408. (
5644)
proenv>proserve sports2000 -S 2001_
```

Next we open the Risk Rating Rules EDC project and open the UnderwritingEDC.ecore vocabulary. Select the root node of the vocabulary and select the Database Access tab. Configure a database connection as follows:

- Select Progress OpenEdge 11.3 from the Database Server dropdown.
- In the Database URL replace the <server> with localhost
- In the Database URL change the port number to 2001
- In the Database URL change the database name to sports2000
- Specify your windows username (for this workshop VM: administrator)
- Specify your windows password (for this workshop VM: ApjPug2015)

Click the ‘Test Connection’ button. If you have any connection problems ask an instructor for assistance.



Database Server: Progress OpenEdge 11.3

Database URL: jdbc:progress:openedge://localhost:2001;databaseName=sports2000

Username: administrator

Password: ************

Catalog filter:

Schema filter:

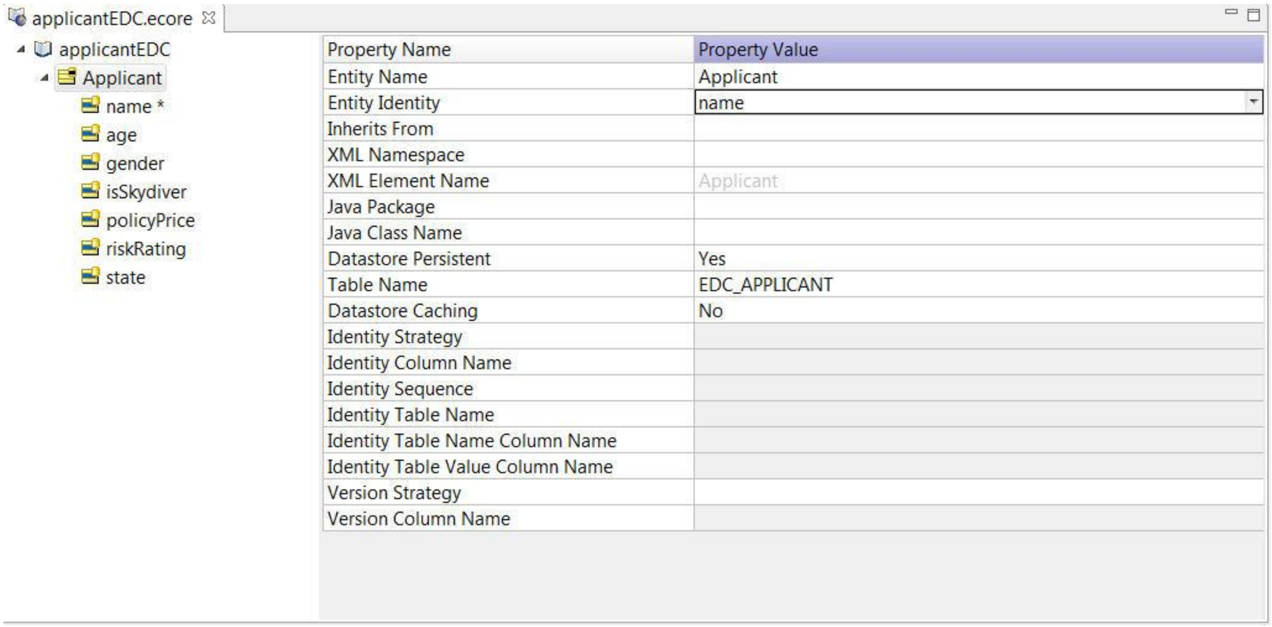
Progress Corticon Studio

Database connection successfully established.

If we wanted to map our vocabulary's entities and attributes to existing tables and columns in the database we would now import a local copy of the database schema metadata which enables Corticon to attempt some name-based automatic mapping, and simplifies manual mapping by populating dropdown selections with valid table and column names. This step is **not necessary for this lab**, but if you do want to import the database metadata, select Database Access -> Import Database Metadata from the Vocabulary menu.

3. Next let's configure **the persistence properties of your vocabulary's entity and attributes.** Select the Applicant node in your vocabulary and then:

- Change the value of the Datastore Persistent property to Yes.
- Set the TableName property to your desired tablename. For this lab specify EDC_APPLICANT to avoid having to edit the loadEDCTestData.sql script that we will use later.
- Select which attribute of Applicant will act as it primary key identifier. Select name for this lab. Normally you would select a unique ID attribute of each entity to be enriched or persisted.



applicantEDC.ecore

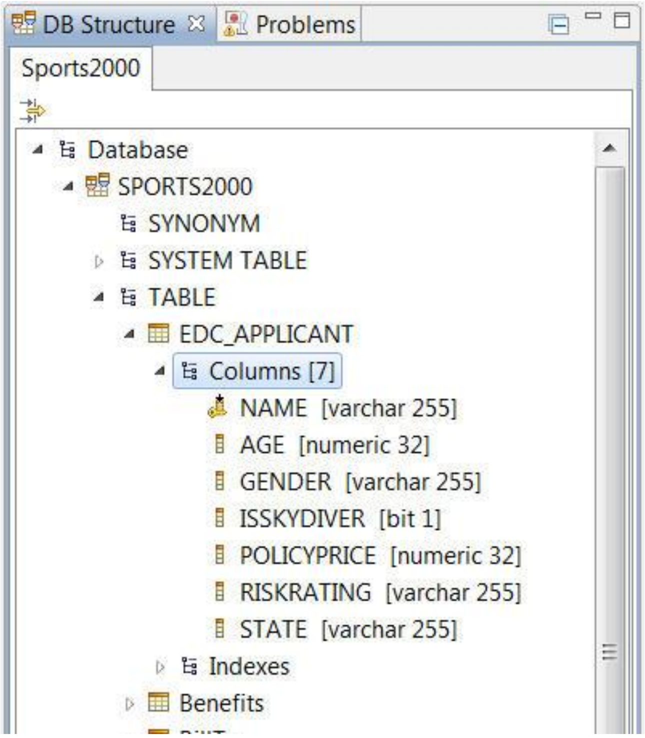
- applicantEDC
  - Applicant
    - name *
    - age
    - gender
    - isSkydiver
    - policyPrice
    - riskRating
    - state

Property Name	Property Value
Entity Name	Applicant
Entity Identity	name
Inherits From	
XML Namespace	
XML Element Name	Applicant
Java Package	
Java Class Name	
Datastore Persistent	Yes
Table Name	EDC_APPLICANT
Datastore Caching	No
Identity Strategy	
Identity Column Name	
Identity Sequence	
Identity Table Name	
Identity Table Name Column Name	
Identity Table Value Column Name	
Version Strategy	
Version Column Name	

Before we instruct Corticon to create the new table in the OE database, let's open the DB Navigator perspective in Progress Developer Studio, and double click the Sports2000 Connection Profile to enable an active connection to the Sports2000 database.

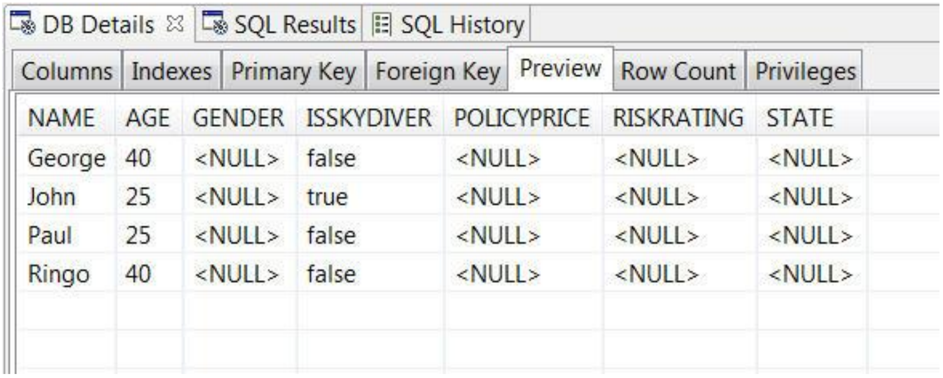
Then in the DB Structure tab you can expand Database->SPORTS2000->TABLE to see the list of tables. Click on a table to see it's properties and preview its data in the DB Details tab. If you want to run some exploratory queries against the database you can right click on Sports2000 in the Active Connection list and select 'New SQL Editor'. At this stage there is no EDC_APPLICANT table in the sports2000 DB corresponding with our vocabulary's Applicant entity.

4. To perform the table creation from the Vocabulary menu select Database Access -> Create/Update Database Schema. In the DB Navigator perspective, right click on the TABLE node of the DB Structure tab and select Refresh to see that the new table has been created to hold applicant data (you might need to click on the active connection in order to get the DB Structure tab to repopulate).



- Database
  - SPORTS2000
    - SYNONYM
    - SYSTEM TABLE
    - TABLE
      - EDC_APPLICANT
        - Columns [7]
          - NAME [varchar 255]
          - AGE [numeric 32]
          - GENDER [varchar 255]
          - ISSKYDIVER [bit 1]
          - POLICYPRICE [numeric 32]
          - RISKRATING [varchar 255]
          - STATE [varchar 255]
        - Indexes
      - Benefits

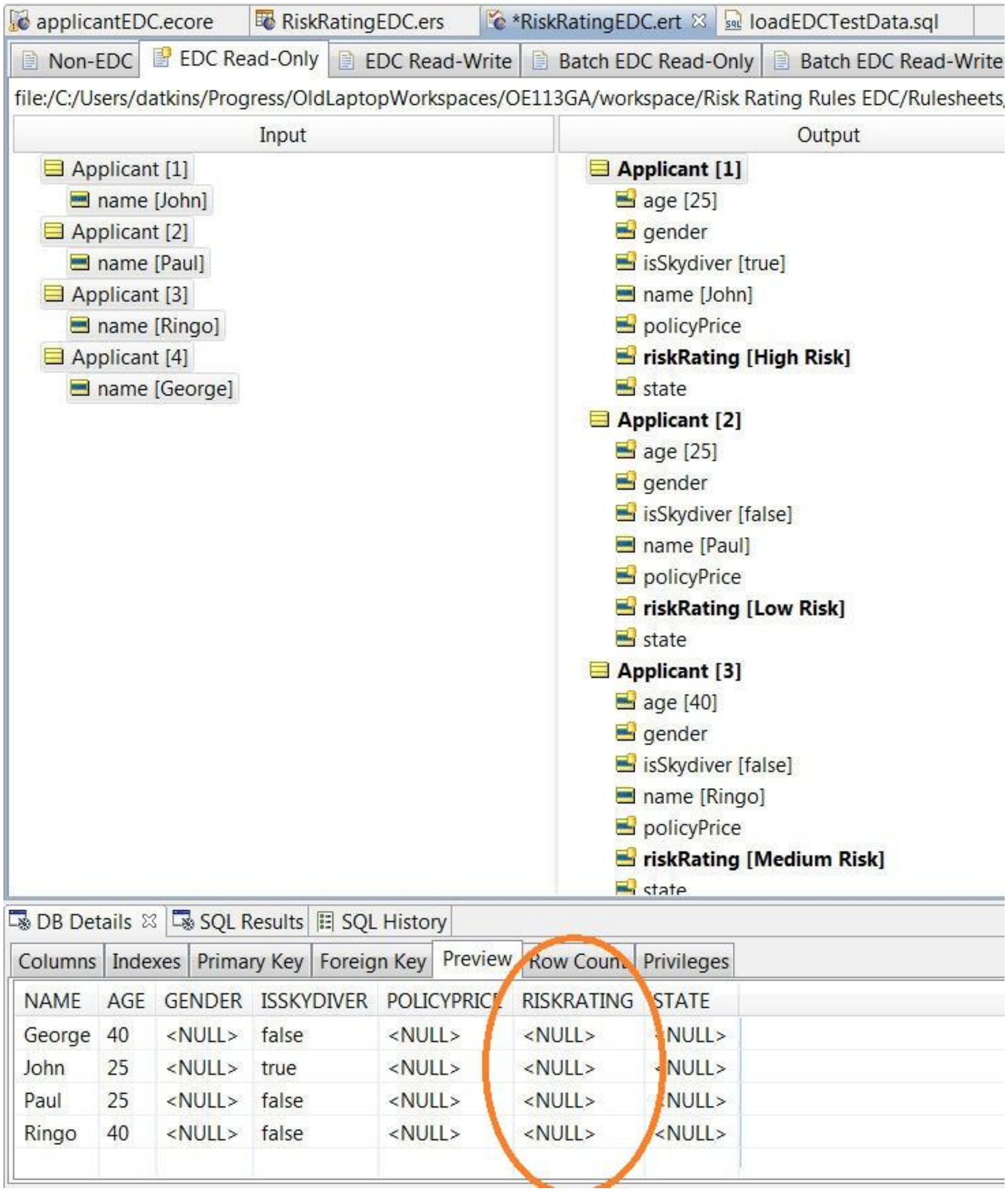
5. Now we have created the table we need to **populate that table with some sample data.** To do this go to the DB Navigator perspective, right click on the Sports2000 Active Connection and select New SQL Editor. Then in the SQL Editor window select the folder icon to open a file selector dialog and select the loadEDCTestData.sql file from the Ruletest folder of the Risk Rating Rules EDC Project in your workspace. Then click the green and white run arrow to execute the SQL script. Assuming you named your table correctly and configured your vocabulary to connect to the database using the DBA1 user, this script should create 4 Applicant records in the new table. Again you can use the DB Navigator view to confirm. Select the table DBA1.EDC_APPLICANT in the DB Structure tab and then select Preview tab in the DB Details area. You should see output like this:



NAME	AGE	GENDER	ISSKYDIVER	POLICYPRICE	RISKRATING	STATE
George	40	<NULL>	false	<NULL>	<NULL>	<NULL>
John	25	<NULL>	true	<NULL>	<NULL>	<NULL>
Paul	25	<NULL>	false	<NULL>	<NULL>	<NULL>
Ringo	40	<NULL>	false	<NULL>	<NULL>	<NULL>

6. In order to **run TestSheets to show Read-only and Read-Write mode EDC behavior** of the RiskRatingEDC.ers RuleSheet, open the RiskRatingEDC.ert Ruletest. There are five TestSheets in this RuleTest: Non-EDC, EDC Read-Only, EDC Read-Write, Batch EDC Read-Only and Batch EDC Read-Write. The Read-Only/Read-Write mode of operation is set for each TestSheet from the RuleTest menu, select TestSheet->DatabaseAccess. For now let's run the first three TestSheets in that order:

- The Non-EDC TestSheet requires all input data (age, isSkydiver, name) to be input in the test case.
- The EDC Read-Only TestSheet also contains 4 test cases, but only passes the name, which you will recall is defined as the primary key of the table we created in the database to store applicant data – yet this TestSheet gets the same output showing that Corticon EDC is dynamically pulling in the required data from the database. However, if we Preview the values in the DBA1.EDC_APPLICANT table again we will see no RISKRATING values have been updated in the database.



file:/C:/Users/datkins/Progress/OldLaptopWorkspaces/OE113GA/workspace/Risk Rating Rules EDC/Rulesheets,

Input

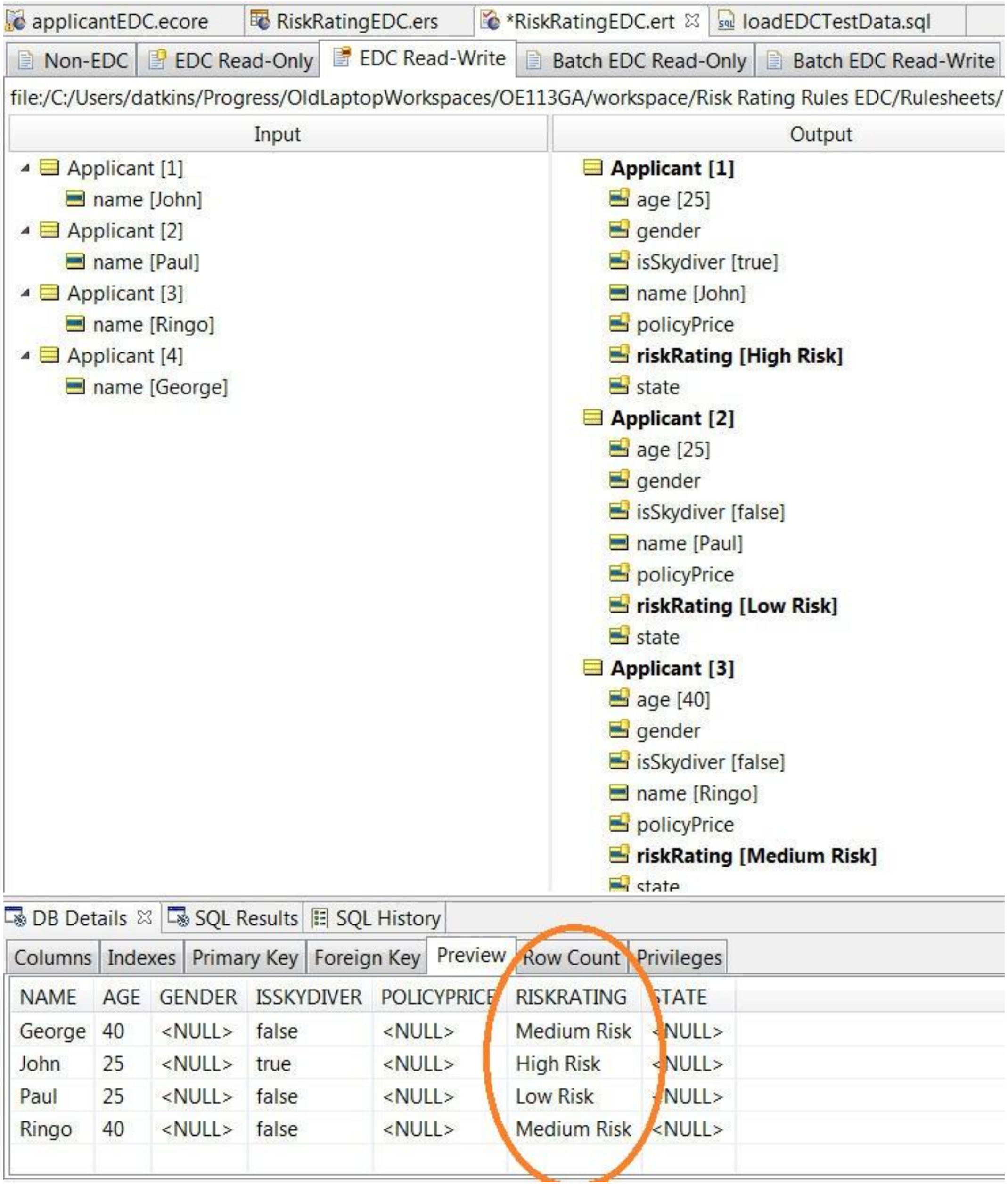
- Applicant [1]
  - name [John]
- Applicant [2]
  - name [Paul]
- Applicant [3]
  - name [Ringo]
- Applicant [4]
  - name [George]

Output

- **Applicant [1]**
  - age [25]
  - gender
  - isSkydiver [true]
  - name [John]
  - policyPrice
  - **riskRating [High Risk]**
  - state
- **Applicant [2]**
  - age [25]
  - gender
  - isSkydiver [false]
  - name [Paul]
  - policyPrice
  - **riskRating [Low Risk]**
  - state
- **Applicant [3]**
  - age [40]
  - gender
  - isSkydiver [false]
  - name [Ringo]
  - policyPrice
  - **riskRating [Medium Risk]**
  - state

NAME	AGE	GENDER	ISSKYDIVER	POLICYPRICE	RISKRATING	STATE
George	40	<NULL>	false	<NULL>	<NULL>	<NULL>
John	25	<NULL>	true	<NULL>	<NULL>	<NULL>
Paul	25	<NULL>	false	<NULL>	<NULL>	<NULL>
Ringo	40	<NULL>	false	<NULL>	<NULL>	<NULL>

- The EDC Read-Write TestSheet contains the same seed data in four test cases as the EDC Read-only TestSheet, but if you Preview the DBA1.EDC_APPLICANT table after executing the EDC Read-Write TestSheet you will see that the RISKRATING values have been updated in the database table as well as being returned to and output by the RuleTest.



file:/C:/Users/datkins/Progress/OldLaptopWorkspaces/OE113GA/workspace/Risk Rating Rules EDC/Rulesheets/

Input

```
Applicant [1]
    name [John]
Applicant [2]
    name [Paul]
Applicant [3]
    name [Ringo]
Applicant [4]
    name [George]
```

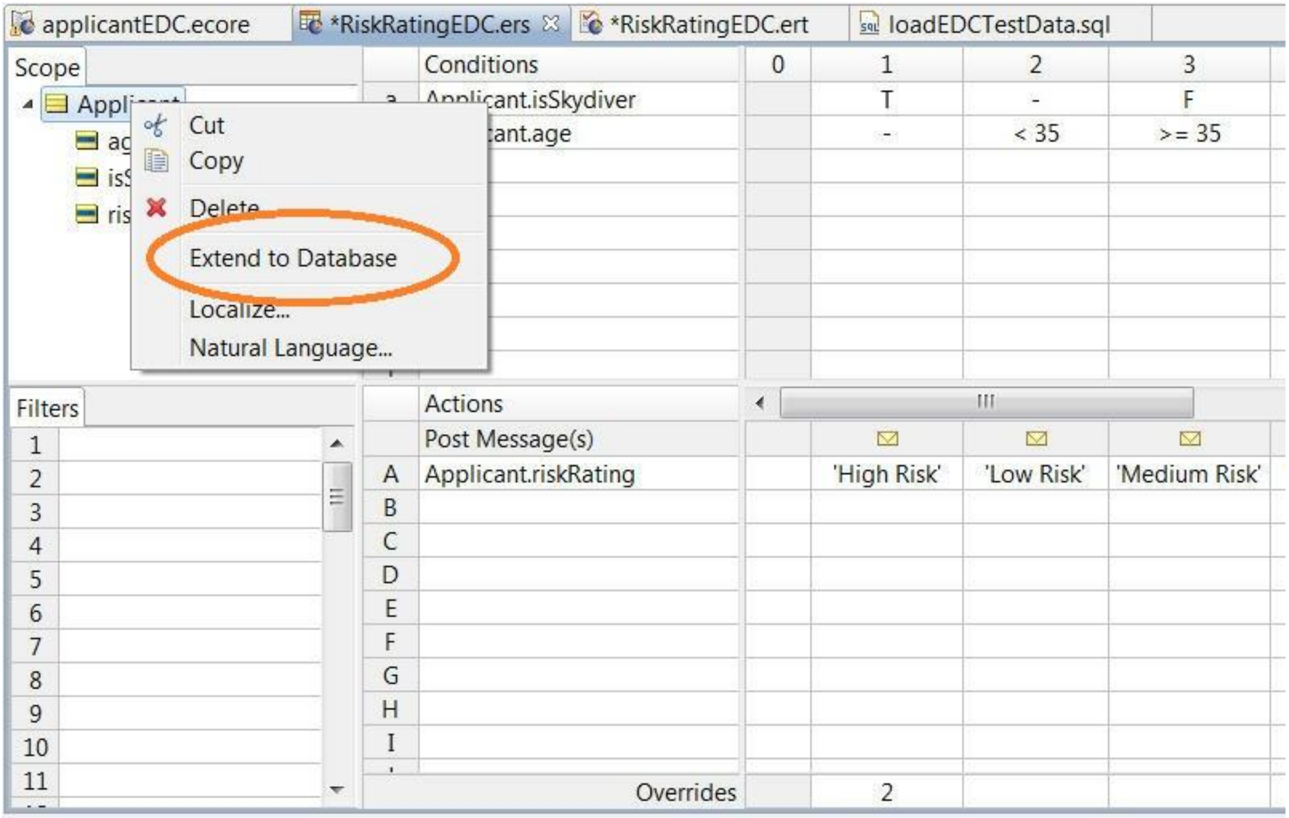
Output

```
Applicant [1]
    age [25]
    gender
    isSkydiver [true]
    name [John]
    policyPrice
    riskRating [High Risk]
    state
Applicant [2]
    age [25]
    gender
    isSkydiver [false]
    name [Paul]
    policyPrice
    riskRating [Low Risk]
    state
Applicant [3]
    age [40]
    gender
    isSkydiver [false]
    name [Ringo]
    policyPrice
    riskRating [Medium Risk]
    state
```

NAME	AGE	GENDER	ISSKYDIVER	POLICYPRICE	RISKRATING	STATE
George	40	<NULL>	false	<NULL>	Medium Risk	<NULL>
John	25	<NULL>	true	<NULL>	High Risk	<NULL>
Paul	25	<NULL>	false	<NULL>	Low Risk	<NULL>
Ringo	40	<NULL>	false	<NULL>	Medium Risk	<NULL>

In some situations it can be useful to **configure a RuleSheet to enable batch execution.** This means that no primary key seed data is required in the request message when executing a decision service. Before we update the RuleSheet to support batch mode operation try running the two Batch TestSheets to see that no output is produced.

To enable batch execution the root entity of your RuleSheet (or of the first RuleSheet in a RuleFlow) needs to be 'extended to the database'. To do this expand the RiskRatingEDC.ers RuleSheet to its Advanced View (from the Rulesheet menu select Advanced View) so you can see the Scope and Filter/Precondition areas. Right click on the root entity in the scope section and select Extend to Database. Save the Rulesheet.

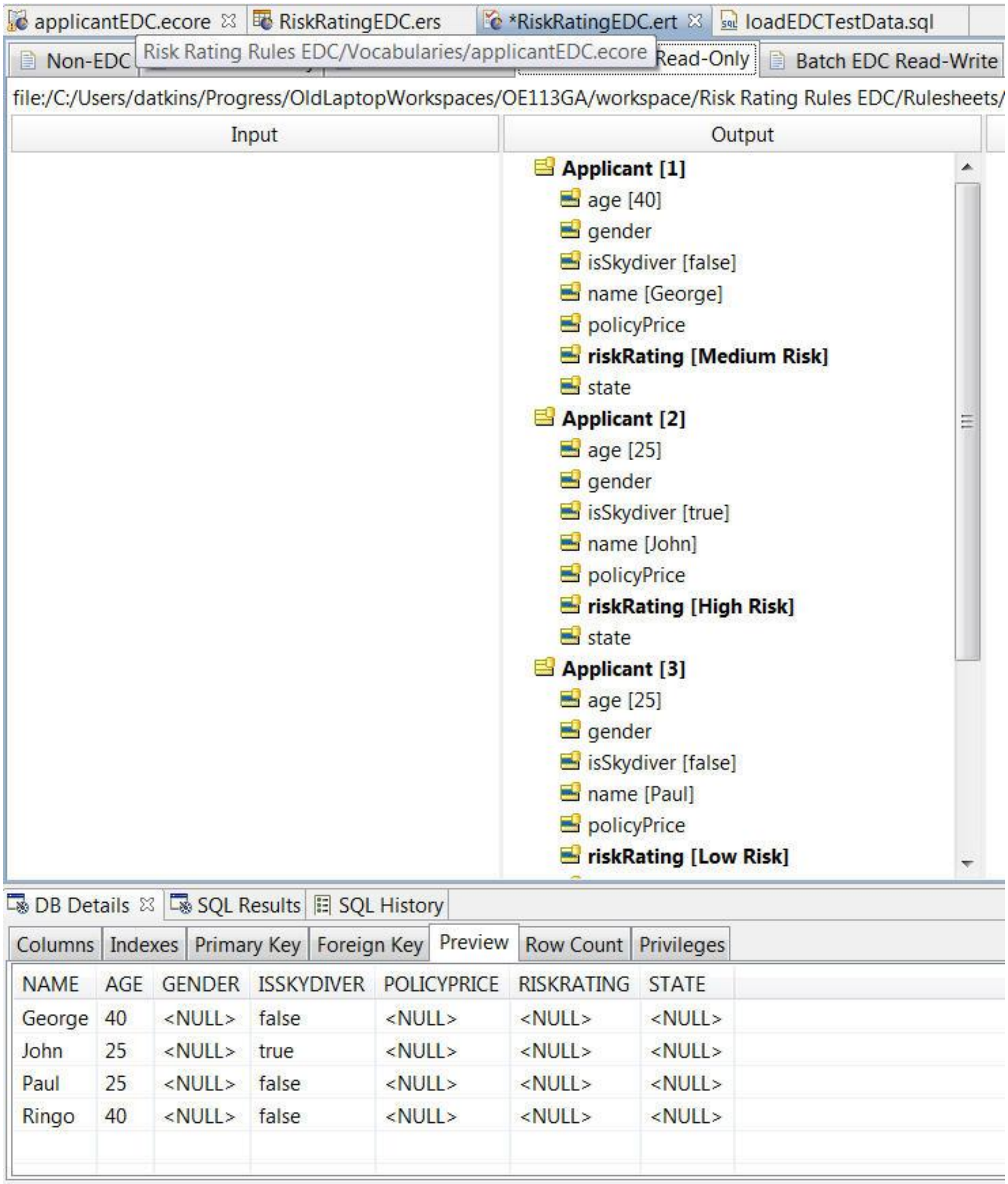


	Conditions	0	1	2	3
a	Applicant.isSkydiver		T	-	F
	cant.age		-	< 35	>= 35

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

7. In order to **run TestSheets to show batch execution of a RuleSheet in Read-only & Read-Write modes** (re)open the RiskRatingEDC.ert Ruletest. Before we proceed, let's reset the test data in our database table by running the resetEDCTestData.sql script in the Vocabularies directory. Now let's run the Batch EDC Read-Only and Batch EDC Read-Write TestSheets:

- Note that the Batch EDC Read-Only TestSheet contains NO input test cases – yet when we run this TestSheet we get the same output showing that Corticon EDC is dynamically pulling in the required data from the database. However, if we Preview the values in the DBA1.EDC_APPLICANT table again we will see no RISKRATING values have been updated in the database.

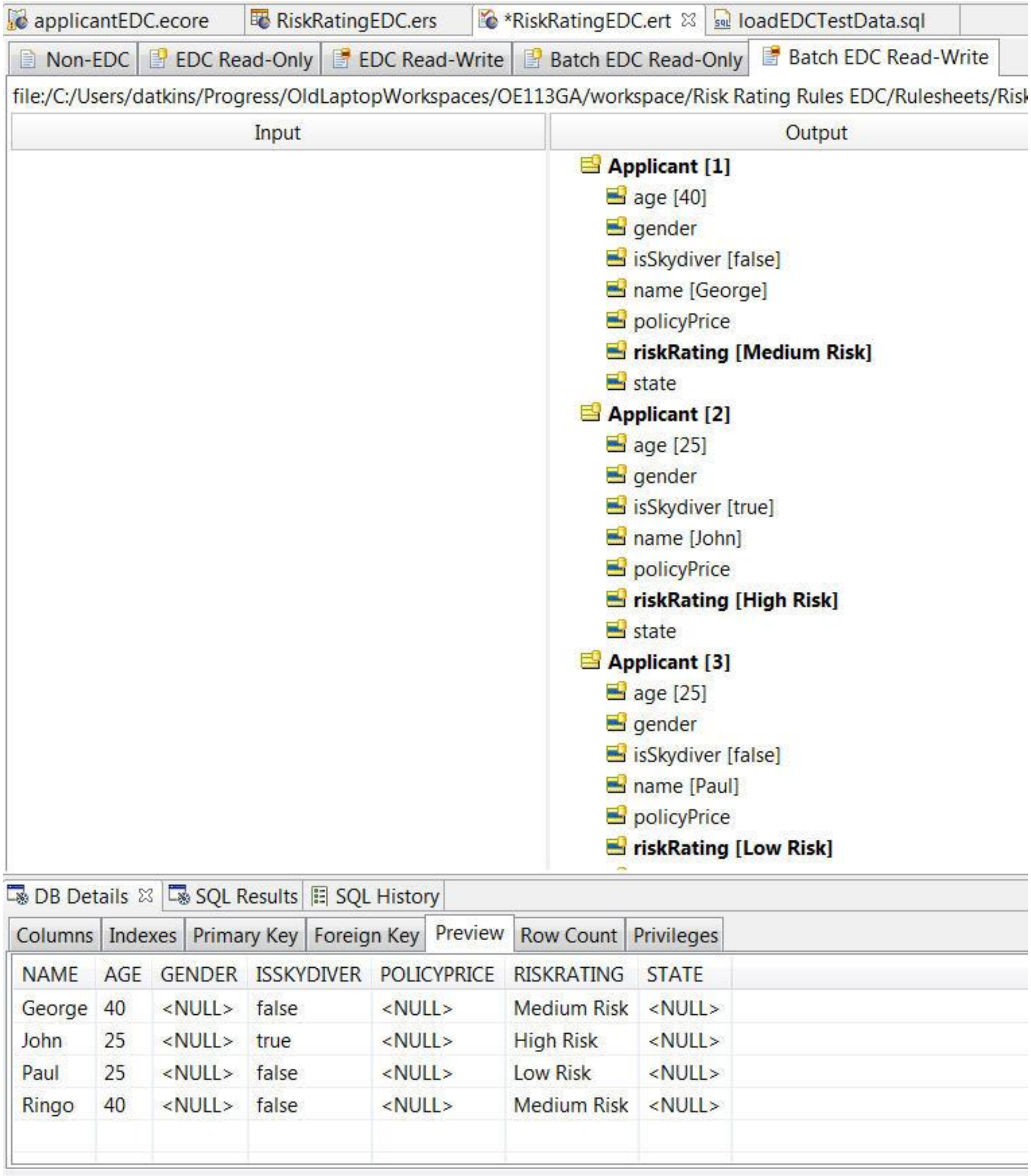


file:/C:/Users/datkins/Progress/OldLaptopWorkspaces/OE113GA/workspace/Risk Rating Rules EDC/Rulesheets/

```
Applicant [1]
    age [40]
    gender
    isSkydiver [false]
    name [George]
    policyPrice
    riskRating [Medium Risk]
    state
Applicant [2]
    age [25]
    gender
    isSkydiver [true]
    name [John]
    policyPrice
    riskRating [High Risk]
    state
Applicant [3]
    age [25]
    gender
    isSkydiver [false]
    name [Paul]
    policyPrice
    riskRating [Low Risk]
```

NAME	AGE	GENDER	ISSKYDIVER	POLICYPRICE	RISKRATING	STATE
George	40	<NULL>	false	<NULL>	<NULL>	<NULL>
John	25	<NULL>	true	<NULL>	<NULL>	<NULL>
Paul	25	<NULL>	false	<NULL>	<NULL>	<NULL>
Ringo	40	<NULL>	false	<NULL>	<NULL>	<NULL>

- The Batch EDC Read-Write TestSheet contains ONE partial input test case, yet when it is executed it still produces 4 output records. This shows that input request data can be combined with batch mode execution, and that input request data will override and overwrite data stored in the database. So, the partial test case for John specifies him to NOT be a skydiver (unlike the pre-populated test data stored in the database) but if you Preview the DBA1.EDC_APPLICANT table after executing the EDC Batch Read-Write TestSheet you will see that John's skydiver status has been updated to false, and also that all four records' RISKRATING values have been updated in the database table as well as being returned to and output by the RuleTest.



file:/C:/Users/datkins/Progress/OldLaptopWorkspaces/OE113GA/workspace/Risk Rating Rules EDC/Rulesheets/Risk

```
Applicant [1]
    age [40]
    gender
    isSkydiver [false]
    name [George]
    policyPrice
    riskRating [Medium Risk]
    state
Applicant [2]
    age [25]
    gender
    isSkydiver [true]
    name [John]
    policyPrice
    riskRating [High Risk]
    state
Applicant [3]
    age [25]
    gender
    isSkydiver [false]
    name [Paul]
    policyPrice
    riskRating [Low Risk]
```

NAME	AGE	GENDER	ISSKYDIVER	POLICYPRICE	RISKRATING	STATE
George	40	<NULL>	false	<NULL>	Medium Risk	<NULL>
John	25	<NULL>	true	<NULL>	High Risk	<NULL>
Paul	25	<NULL>	false	<NULL>	Low Risk	<NULL>
Ringo	40	<NULL>	false	<NULL>	Medium Risk	<NULL>

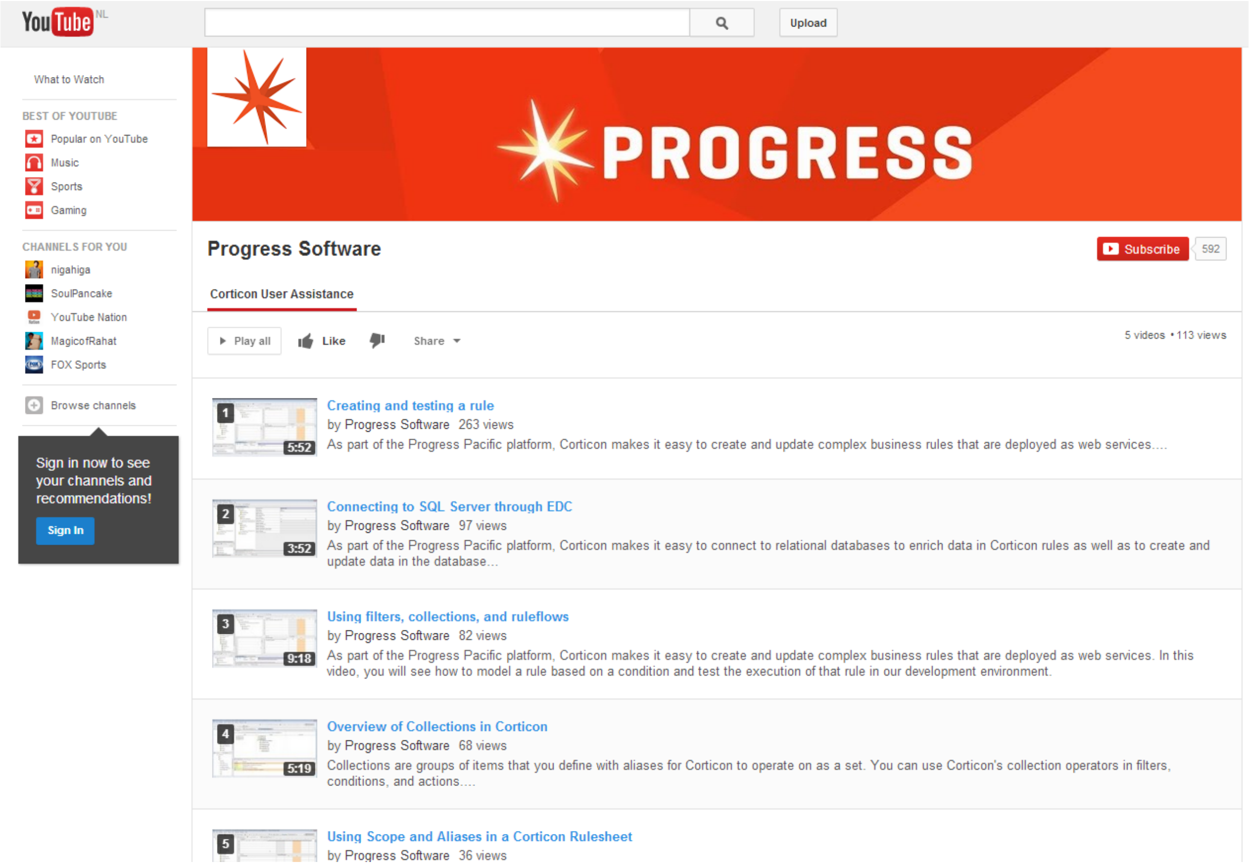
Corticon EDC can also connect to several other external datasources: Oracle 10/11, SQL Server 2005/2008/2012, IBM DB2, OpenEdge v10.2 (as well as v11.3), and can also connect to non-relational data sources (Rollbase and SalesForce.com) using the relational abstraction and connectivity of those data sources provided by the DataDirect Cloud JDBC Driver.

Conclusion

This concludes this workshop. We hoped you enjoyed it. Due to the limited time, we only scratched the surface of what is possible with Corticon for OpenEdge.

What to do next?

If you'd like to learn more about Corticon have a look at the Corticon YouTube channel for some great technical videos: http://www.youtube.com/playlist?list=PLC679RvCan2AFmzCof-8KZV0Mx8WpeE9L



Progress Software

Corticon User Assistance

5 videos • 113 views

1 5:52 Creating and testing a rule
by Progress Software 263 views
As part of the Progress Pacific platform, Corticon makes it easy to create and update complex business rules that are deployed as web services....

2 3:52 Connecting to SQL Server through EDC
by Progress Software 97 views
As part of the Progress Pacific platform, Corticon makes it easy to connect to relational databases to enrich data in Corticon rules as well as to create and update data in the database...

3 9:18 Using filters, collections, and ruleflows
by Progress Software 82 views
As part of the Progress Pacific platform, Corticon makes it easy to create and update complex business rules that are deployed as web services. In this video, you will see how to model a rule based on a condition and test the execution of that rule in our development environment.

4 5:19 Overview of Collections in Corticon
by Progress Software 68 views
Collections are groups of items that you define with aliases for Corticon to operate on as a set. You can use Corticon's collection operators in filters, conditions, and actions....

5 Using Scope and Aliases in a Corticon Rulesheet
by Progress Software 36 views

Further learning? Note there are e-learning courses available on the Progress Education Community. Within 3 days, we'll teach to create the most sophisticated Corticon rule models through the following courses:

1. **Introduction to Decision Modeling with Progress Corticon Studio**
   This course is an introductory course for Corticon. It provides an overview of Business Rule Modeling Systems (BRMS), their purpose and value, and the function of the different components of Progress Corticon. It teaches how to write and test business rules using Progress Corticon Studio and covers the basic concepts of creating Vocabularies, defining rules in Rulesheets, testing rules using Ruletests, and creating Ruleflows.

2. Advanced Decision Modeling with Progress Corticon Studio

This course focuses on advanced features in Corticon Studio. It begins with a review of the following rule modeling components—Vocabulary, Rulesheet, Ruletest, and Ruleflow. You learn about Vocabulary features such as Custom Data Types and Domains. You learn about Rulesheet features such as Scope, Aliases, Collections, Filters, Dependency, and Looping. You then learn about the following Ruleflow features—Subflows, Iteration, and Service Call-outs. Finally, you learn about Ruletest features such as Annotations, generating data trees, and testing multiple Ruleflows from a single Ruletest.

3. Using Corticon Business Rules in a Progress OpenEdge Application

In this course, you will learn how to use Corticon's business rules in an OpenEdge application. First you will learn how to set up an integrated development environment that contains OpenEdge and Corticon. Then you will learn how to export a Business Rules Vocabulary Definition from an OpenEdge application. Finally you will learn how to use a Corticon decision service from an OpenEdge application.

After taking this class, you should be able to:

- Set up an integrated environment that contains Corticon and OpenEdge
- Export a Business Rules Vocabulary Definition file from an OpenEdge application
- Use a Corticon decision service from an OpenEdge application

Lastly, talk to your Progress Account manager

to get your rules projects going!