Business Logic

Exercises

Contents

1. Module 3

1.1 Objectives 3

1.2 Overview 3

2. About Business Logic 4

2.1 Introduction 4

2.2 References 4

3. Exercises 5

3.1 Prerequisites 5

3.2 Configure Design and Deployment Structure 5

3.2.1 Creating Design Folder Structure 5

3.2.2 Creating Deployment Structure 6

3.3 Decision Table on WS-AppServer Classes 6

3.3.1 Creating the Rule Group 6

3.3.2 Creating the Decision Table 7

3.3.3 Setting the Conditions 9

3.3.4 Defining the Actions 14

3.3.5 Save the Decision Table 16

3.3.6 Duplicating a Rule 16

3.3.7 Inserting a Rule 17

3.4 Using Decision Tables 18

3.4.1 Using the Rule Test Tool 18

3.4.2 Configuring Service Container for Rule Logic............19
3.4.3 Decision Tables at Run Time.....................................20
3.5 Using a Business Process as Rule Action.......................23
3.5.1 Prerequisites...23
3.5.2 Qualify & Analyze 23
3.5.3 Creating the Business Process Model.......................23
3.6 Rules on WS-AppServer Classes....................................27
3.6.1 Creating the Rule Group..27
3.6.2 Creating the Rule...27
3.7 Testing the Rule 32
3.7.1 Configure Service Container for Rule logic................32
3.7.2 Using the Rule Test Tool Artifact...............................32
3.7.3 Runtime Action 34
3.8 Decision Tables in a Business Process Models 35
3.8.1 Creating XML Schema 35
3.8.2 Creating Decision Table 37
3.8.3 Testing the Decision Table 41
3.8.4 Creating the Business Process Model.......................42
3.9 Make Changes Available to SCM....................................48
4. Learning Report 49

1. Module

1.1 Objectives

After completing this course module, you will be able to:

- Explain the concept of business logic with Cordys
- Implement business logic using decision tables
- Use a decision table on backend action
- Us a decision table in a Business Process Model

1.2 Overview

An enters functioning is not only described in terms of the structure of data and the organization of the operations that it performs, but also in terms of business logic, i.e. the constraints under which the enterprise operates. Business logic determines how a business operates that is, business rules can prevent, cause, or suggest actions or steps to be executed at several points in a business process.

2. About Business Logic

2.1 Introduction

Business process models describe the flow of the processes or the operations in your business. In addition, your business may require operations to act on certain events in your business or environment as well. This kind of business logic can be implemented by means of rules which enable your business to effectively respond and execute on certain business events.

For example, when a new sales order is received, you want this event to trigger a number of necessary actions as a consequence of the sales order, like assign discounts based on the customer details and order amount. When a car insurance quote is received, the calculation of the insurance can be done automatically based age of the driver, car price etc. and the quote sent back.

As these rules may change frequently based on changing regulations, standards, policies, etc. you want to model business rules in such a way that you can easily update them without embedding the rules in a programming language like java, C, etc.

In Cordys you can use decision tables and business rules to achieve this.

2.2 References

More information about this subject is available

- Cordys Online Documentation
  Working with Business Models → Modeling Business Rules
- http://community.cordys.com

3. Exercises

3.1 Prerequisites

Before you can start with this module please note the following prerequisites, the exercises are written based on their successful completion.

You must have completed the following modules

- Application Management
- Connectivity
- Developing Web Services
- (Developing Process for some exercises)
- (Modeling Data for some exercises)

You must have ONLY the following roles assigned to yourself

- Administrator
- Developer
- Cordys Fundamentals Trainee

3.2 Configure Design and Deployment Structure

In this exercise you will setup the folder structure for the design time and deployment time components for decision tables and business rules.

For detailed information see module Application Management.

3.2.1 Creating Design Folder Structure

Here you will setup the design folder structure for modeling data according to the standard for this training (See Application Management).

1. Open the *Workspace Documents*.
2. Open the *Fundamentals training* workspace.
3. Make sure the *My Application Project* is opened
4. Right click *My Application Project* and select *New → Folder*.
5. Enter the name: **Rules**.

NOTE

The naming convention for folders in your project dictates the document type in plural. However, when implementing business logic, this is a combination of decision tables, rule groups and rules. These will be stored together in one folder called (business) *Rules*.

3.2.2 Creating Deployment Structure

Here you will create the deployment structure.

<u>Decision tables and rules</u>

Rules, decision tables and rule groups are at runtime identified by their name and the application (package) they belong to, so no additional action is required.

3.3 Decision Table on WS-AppServer Classes

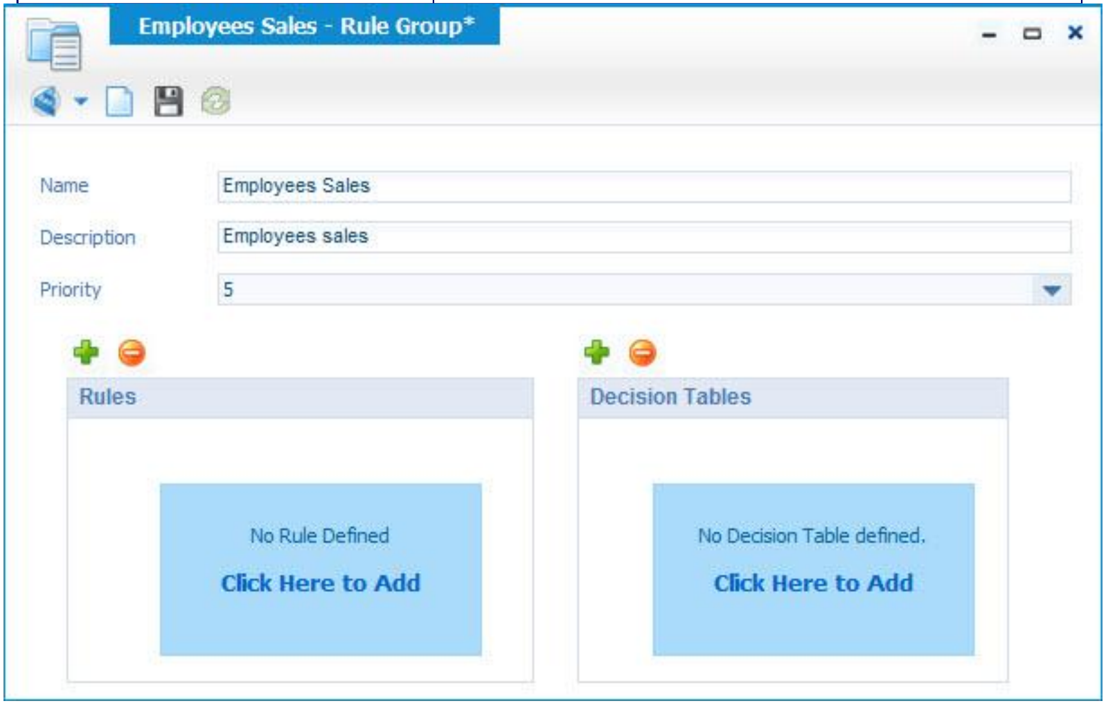
In this exercise you will create a decision table to assign the sales manager to a new sales employee based on the location details of the new sales employee.

3.3.1 Creating the Rule Group

Decision tables and rules need to be combined into (or as part of) a rule group. The rule group details will be used first to decide in which order the rule groups are executed. It also provides the functionality to combine the decision tables and rules based on their logical function.

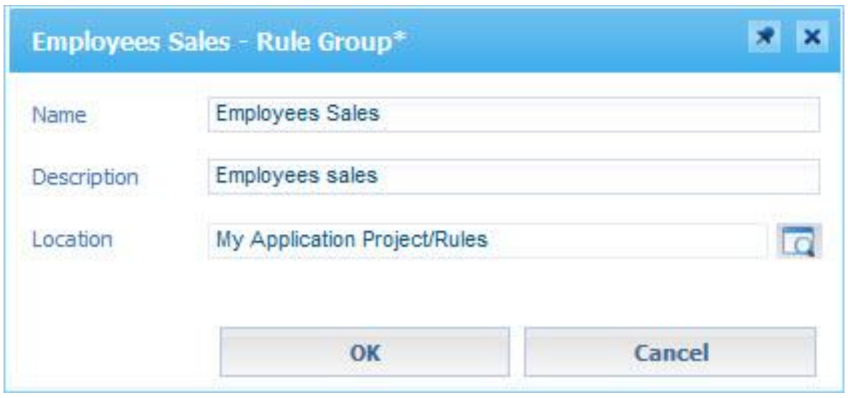
1. In the *Workspace Documents* toolbar, click **New**.
2. Select the *Rule Group* document.
   By default, the priority is 5 in a range of 1(lowest) through 10 (highest).
3. Provide the following details.

Field	Value
Name	Employees Sales
Description	Employees sales
Priority	5



Name: Employees Sales
Description: Employees sales
Priority: 5

4. Save the rule group
Location: *My Application → Rules*.


Employees Sales - Rule Group*

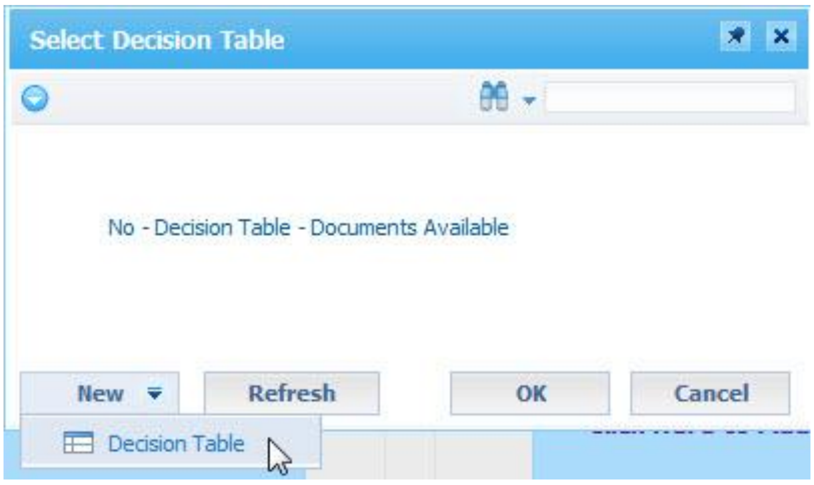
Name	Employees Sales
Description	Employees sales
Location	My Application Project/Rules

5. Leave the rule group open.

3.3.2 Creating the Decision Table

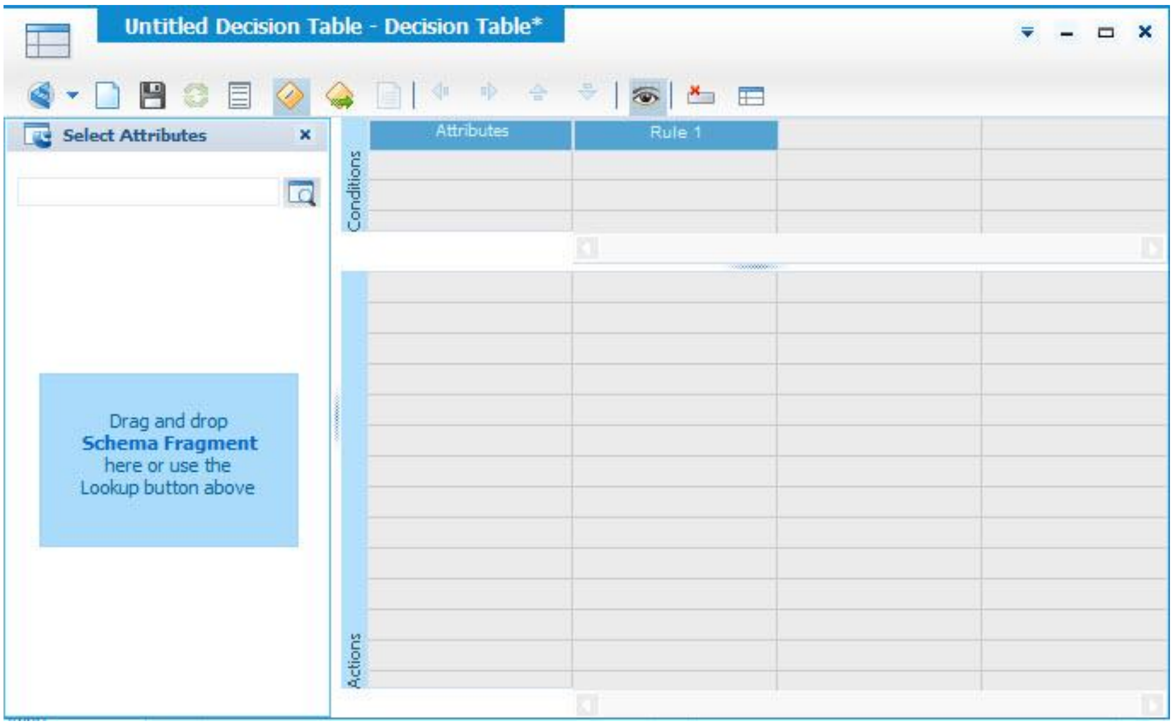
In this part you will create a decision table, defining actions or assignments as a consequence of specific conditions and events.

1. In the *Decision Table* part click **Add** or Click Here to Add.
2. Click **New** and select *Decision Table*.


Select Decision Table

No - Decision Table - Documents Available

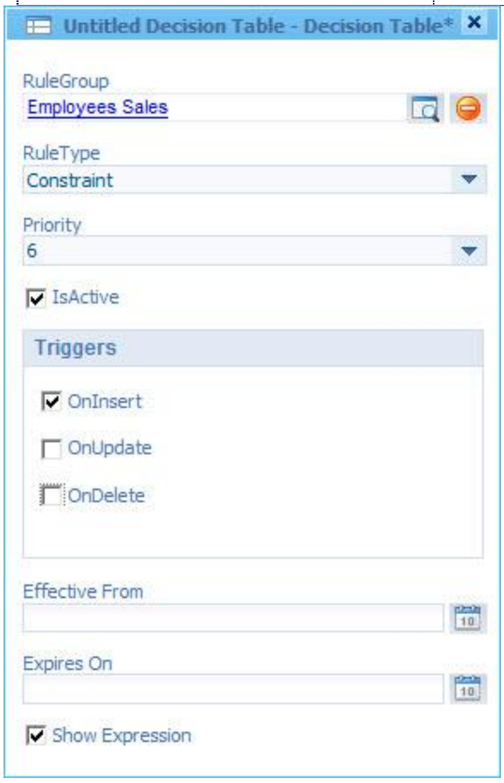
3. The decision table editor opens:



4. In the toolbar, click **Show Decision Table Properties** (▤).

5. Provide the following values:

Field	Value
Rule Group	Employees Sales
Rule Type	Constraint (make change before sending data to the backend)
Priority	6
Triggers	OnInsert - checked OnUpdate – unchecked OnDelete – unchecked (these rules only apply in case of a new sales employees)


Untitled Decision Table - Decision Table*

RuleGroup: Employees Sales

RuleType: Constraint

Priority: 6

☑ IsActive

Triggers

☑ OnInsert

☐ OnUpdate

☐ OnDelete

Effective From

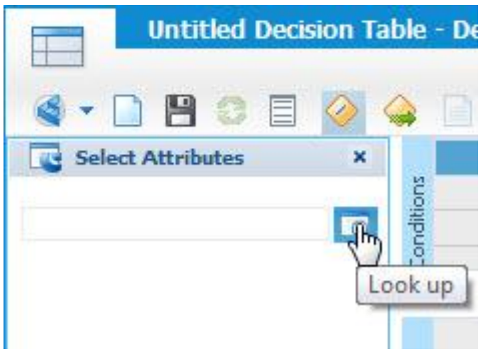
Expires On

☑ Show Expression

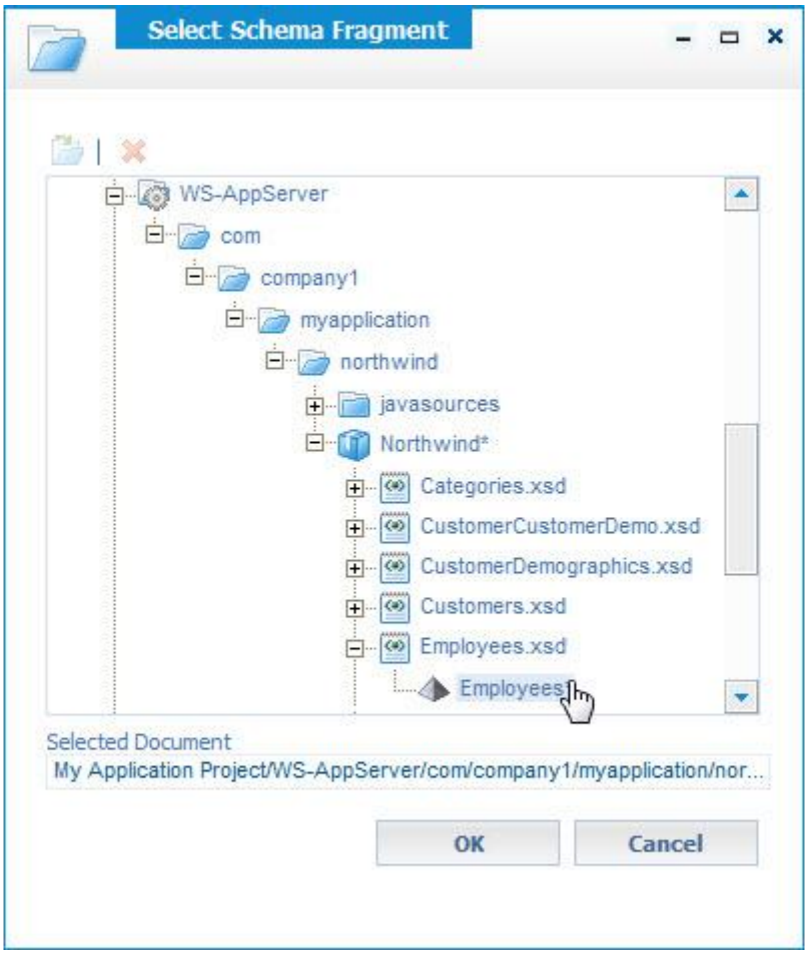
3.3.3 Setting the Conditions

In this part you will create/set the appropriate conditions of the decision table, based on the elements of a WS-AppServer package class.

1. Click the lookup button for the *Select Attributes*.



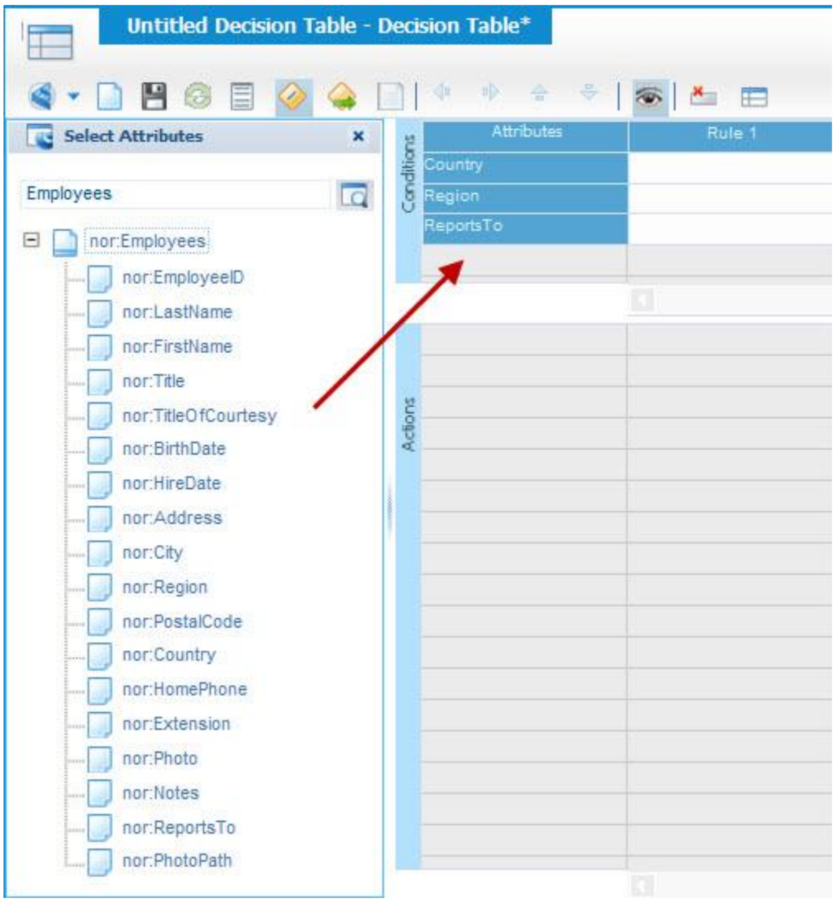
2. Navigate to *My Application project → WS-AppServer → com → companyX → myapplication → northwind → Northwind → Employees.xsd → Employees*.



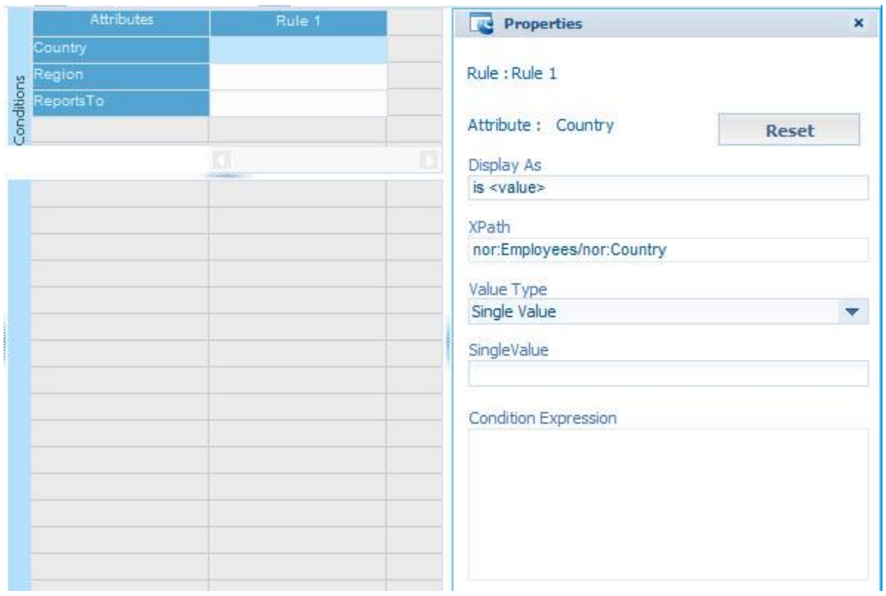
3. Select the *Employees* fragment and click **OK**.

4. Drag and drop the following fields into the *Attributes* column of the decision table:

- *Country*
- *Region*
- *ReportsTo*



5. Select the condition cell of *Rule 1* and attribute *Country*.
You will find the properties in the right hand part of the decision table editor, or click **Show Properties** ( ) in the toolbar to open it.



Rule : Rule 1

Attribute : Country

Display As
is <value>

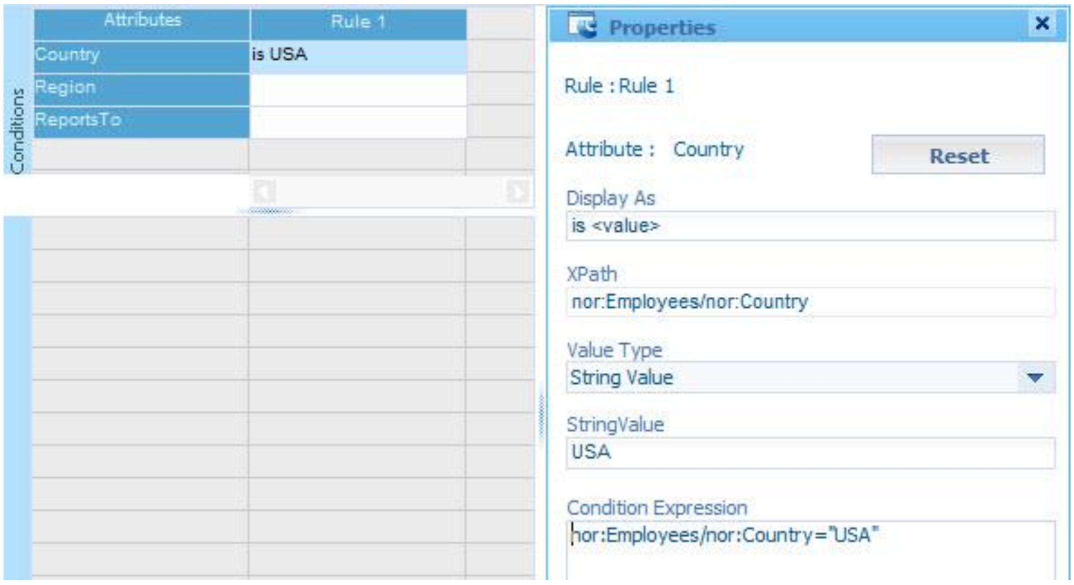
XPath
nor:Employees/nor:Country

Value Type
Single Value

SingleValue

Condition Expression

6. Select *Value Type* **String Value** and at the *Value* field enter: **USA**



Attributes	Rule 1
Country	is USA
Region	
ReportsTo	

Rule : Rule 1

Attribute : Country

Display As: is <value>

XPath: nor:Employees/nor:Country

Value Type: String Value

StringValue: USA

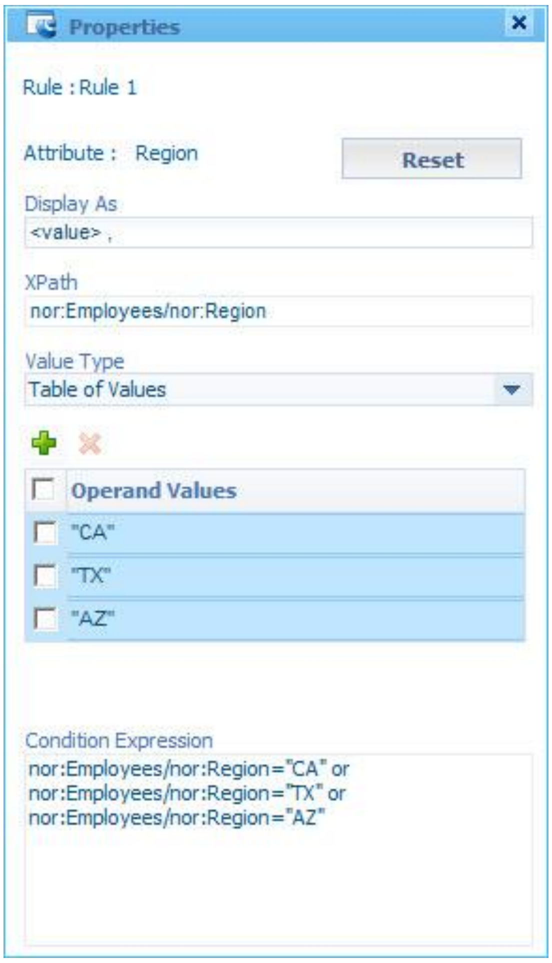
Condition Expression: nor:Employees/nor:Country="USA"

NOTE
When you use the value type: *String Value*, you do not add double quotes around the text to signify that it is a string value.

7. Select the condition cell of *Rule 1* and attribute *Region*.

8. Select *Value Type* **Table of Values** and add three separate entries with the double quotes:

- "CA"
- "TX"
- "AZ"



Rule : Rule 1

Attribute : Region

Display As
<value> ,

XPath
nor:Employees/nor:Region

Value Type
Table of Values

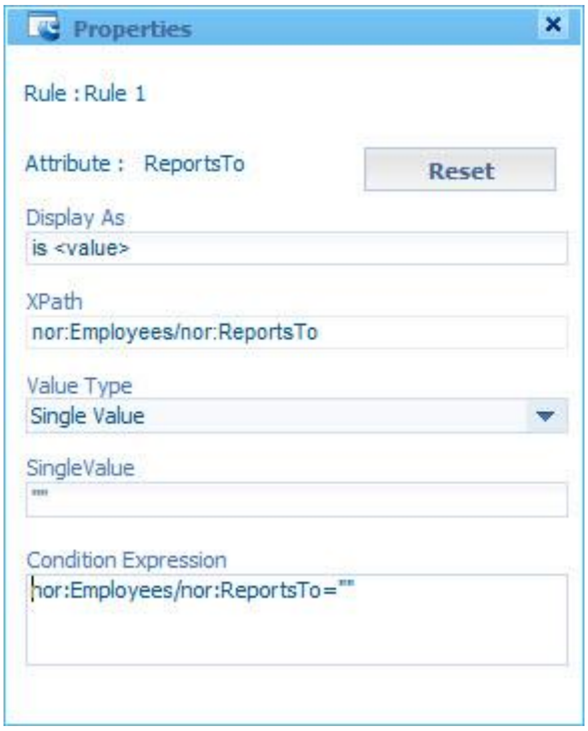
☐	Operand Values
☐	"CA"
☐	"TX"
☐	"AZ"

Condition Expression

```
nor:Employees/nor:Region="CA" or
nor:Employees/nor:Region="TX" or
nor:Employees/nor:Region="AZ"
```

9. Select the condition cell of *Rule 1* and attribute *ReportsTo.*

10. Select *Value Type* **Single Value** and at the *Value* field enter **""**.



Rule : Rule 1

Attribute : ReportsTo

Display As: is <value>

XPath: nor:Employees/nor:ReportsTo

Value Type: Single Value

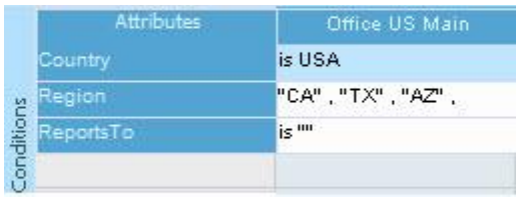
SingleValue: ""

Condition Expression: nor:Employees/nor:ReportsTo=""

11. Select the column header of *Rule 1*.

12. Change the *Display As* field to **Office US Main**.

13. Your *Rule 1* condition set should look like this:



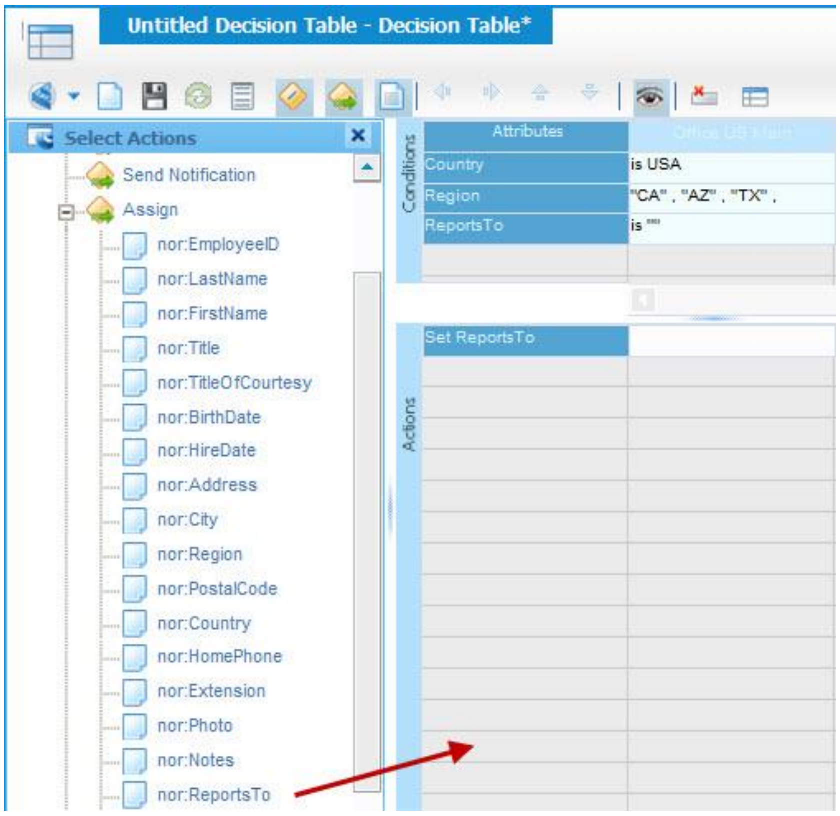
Attributes	Office US Main
Country	is USA
Region	"CA" , "TX" , "AZ" ,
ReportsTo	is ""

3.3.4 Defining the Actions

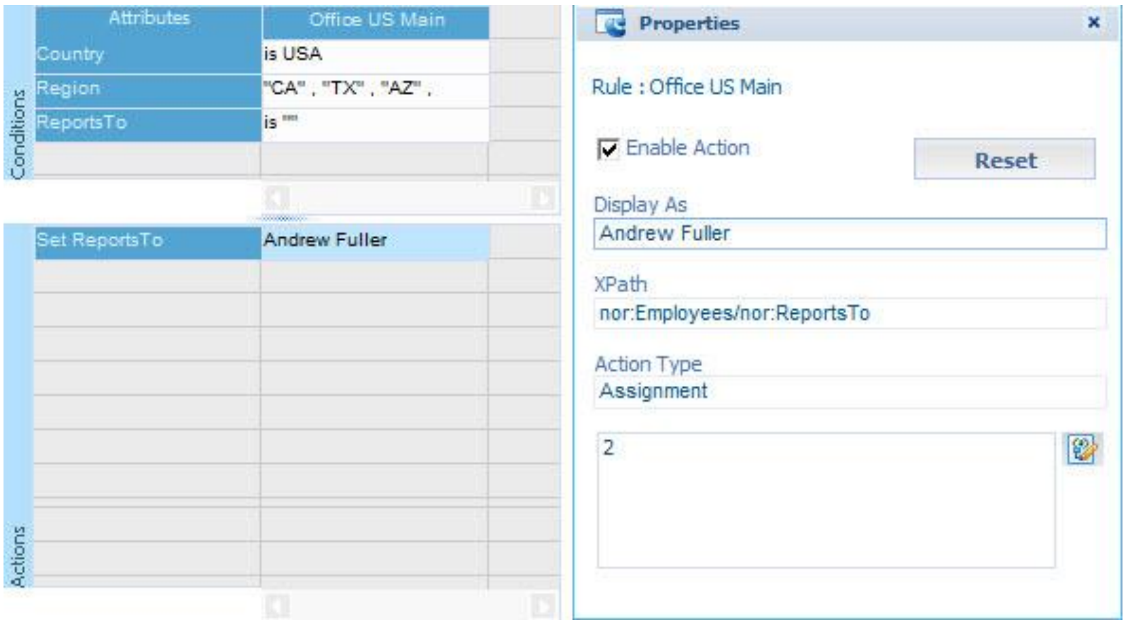
In this part you will define the actions that must be executed when the condition(s) are met.

1. In the toolbar of the decision editor, click **Show Actions** ( ).

2. From the *Select Actions group* box, drag and drop from the *Assign* group, the *ReportsTo* field into the *Actions* area.



3. Select the action cell of *Rule 1* and action *Set ReportsTo*.
4. At the *assignment* area, enter **2** and for the *Display As* enter the value **Andrew Fuller**.

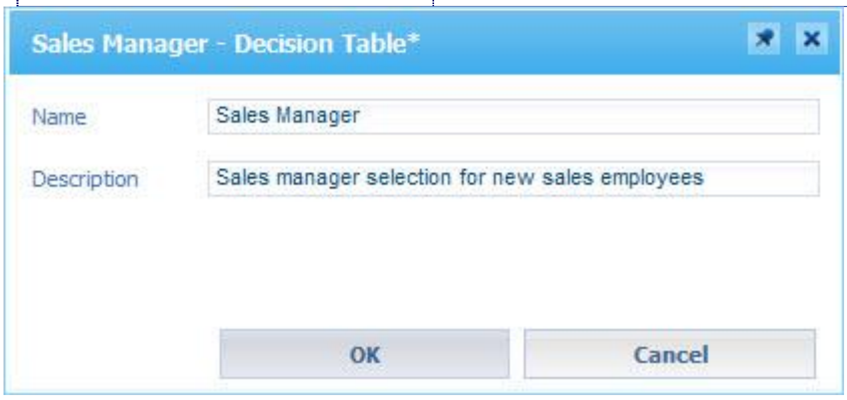


3.3.5 Save the Decision Table

Now that you have a condition and an action you can save the decision table.

1. Save the decision table with the following values:

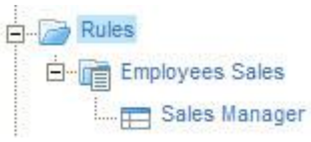
Field	Value
Name	Sales Manager
Description	Sales manager selection for new sales employees



Sales Manager - Decision Table*

Name	Sales Manager
Description	Sales manager selection for new sales employees

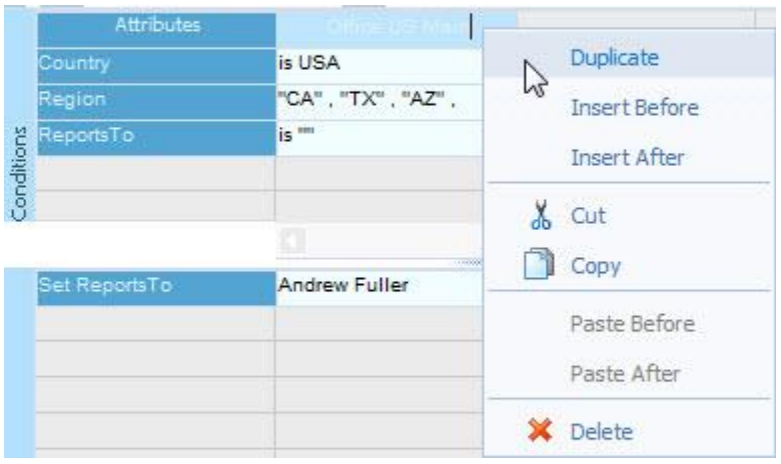
2. Click **Save**.



3.3.6 Duplicating a Rule

1. If closed open the *Sales Manager* decision table.

2. Right click the first rule header and select *Duplicate*.

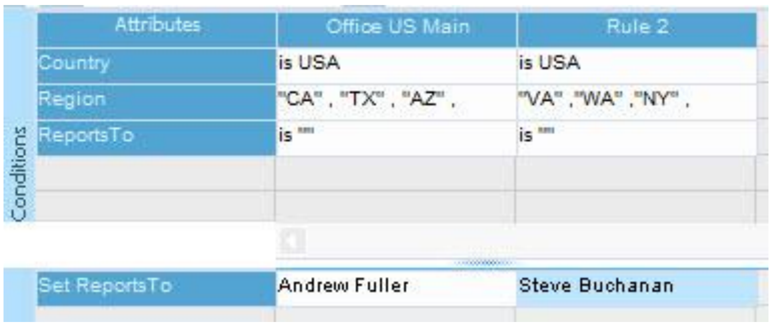


Attributes	
Country	is USA
Region	"CA" , "TX" , "AZ" ,
ReportsTo	is ""
Set ReportsTo	Andrew Fuller

3. For the duplicated rule, change the following values:

Field	Value
Name	Rule 2
Display As	Office US
Regions	"VA" , "WA", "NY"
Set ReportsTo	Display As: Steve Buchanan value: 5

4. Your decision table should be similar to:



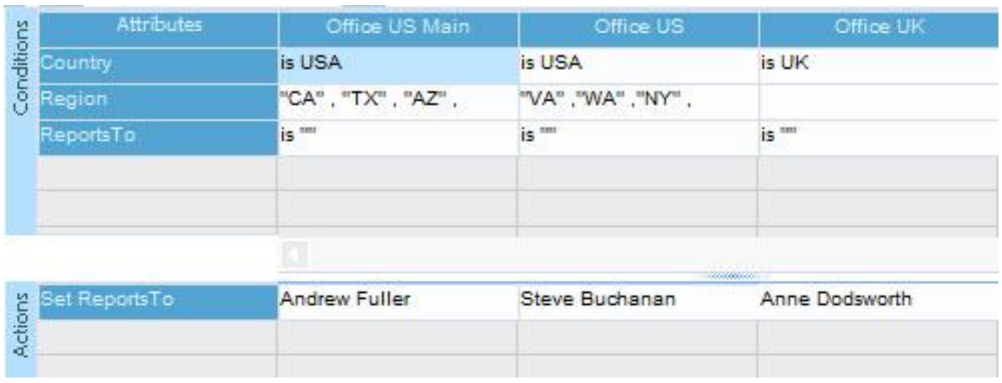
Attributes	Office US Main	Rule 2
Country	is USA	is USA
Region	"CA" , "TX" , "AZ" ,	"VA" ,"WA" ,"NY" ,
ReportsTo	is ""	is ""
Set ReportsTo	Andrew Fuller	Steve Buchanan

3.3.7 Inserting a Rule

1. Right click the second rule header and select *Insert After*.
2. Create rule 3 based on the following configuration:

Field	Value
Name	Rule 3
Display As	Office UK
Country	String value, with value UK.
ReportsTo	Single Value, with value ""
Set ReportsTo	Display As: Anne Dodsworth value: 9

3. The decision table should be similar to:



Attributes	Office US Main	Office US	Office UK
Country	is USA	is USA	is UK
Region	"CA" , "TX" , "AZ" ,	"VA" ,"WA" ,"NY" ,	
ReportsTo	is ""	is ""	is ""
Set ReportsTo	Andrew Fuller	Steve Buchanan	Anne Dodsworth

4. Save your decision table.

Are these rules expected to be changed frequently?

____

____

3.4 Using Decision Tables

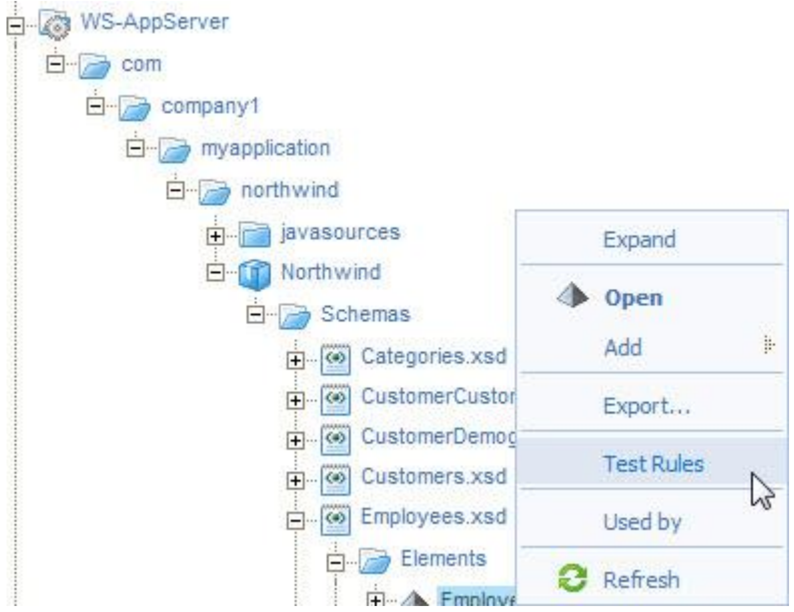
When the decision table is created there are 2 ways to test proper functioning of the modeled conditions and actions:

- Using the *Rule Test Tool*.
- Performing the action that will actually trigger the rule, for example inserting a new employee.

In this exercise you will use both ways to explore the possibilities of the rule test tools as well as see the full behavior of the decision tables at runtime when, for example, a new employee is inserted.

3.4.1 Using the Rule Test Tool

1. In the *Workspace Documents*, navigate to *WS-AppServer → com → companyX → myapplication → northwind → Northwind → Schemas → Employees.xsd → Elements → Employees*.
2. Right click the *Employees* schema fragment and select *Test Rules*:



NOTE
You can also right click the decision table to test only the decision table, by starting from the Employees object all rules and decision tables related to the Employees will be used in the test cycle.

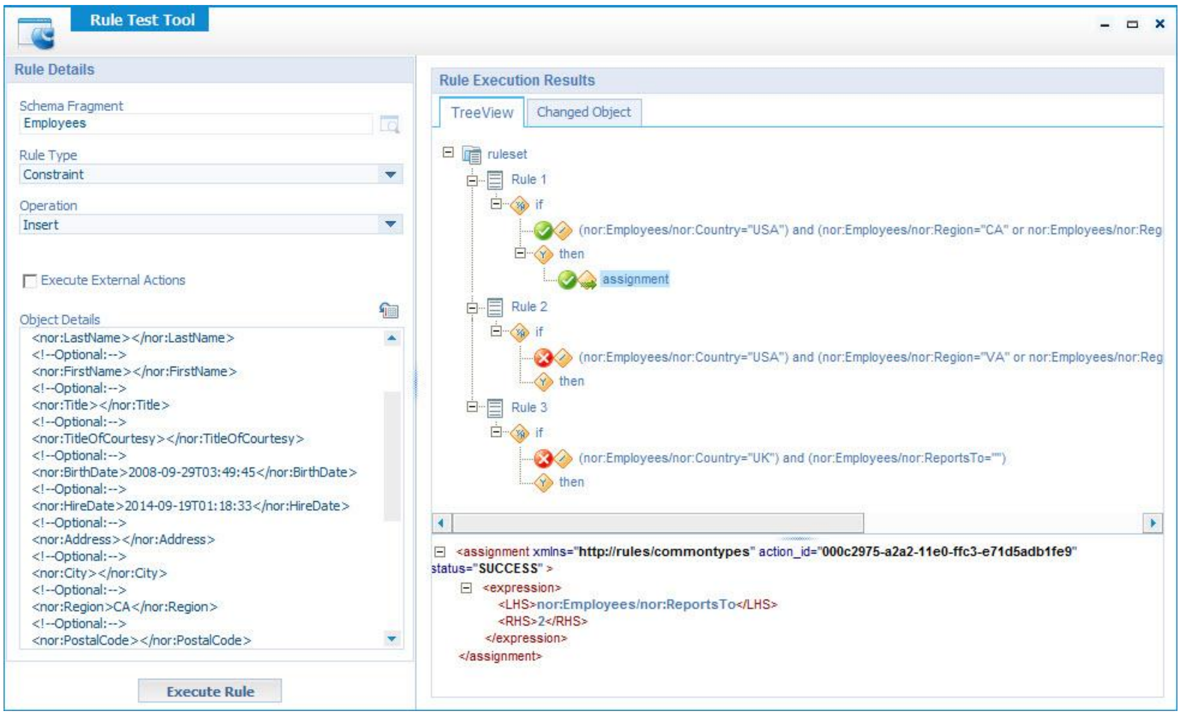
3. Uncheck *Execute External Actions*.

4. In the *Object details*, change the values of the fields:

Field	Value
Country	**USA**
Region	**CA**
ReportsTo	empty, *do not enter a value*

This will qualify with one of the rules in your decision table.

5. Click **Execute Rule**.



```
<nor:LastName></nor:LastName>
<!--Optional:-->
<nor:FirstName></nor:FirstName>
<!--Optional:-->
<nor:Title></nor:Title>
<!--Optional:-->
<nor:TitleOfCourtesy></nor:TitleOfCourtesy>
<!--Optional:-->
<nor:BirthDate>2008-09-29T03:49:45</nor:BirthDate>
<!--Optional:-->
<nor:HireDate>2014-09-19T01:18:33</nor:HireDate>
<!--Optional:-->
<nor:Address></nor:Address>
<!--Optional:-->
<nor:City></nor:City>
<!--Optional:-->
<nor:Region>CA</nor:Region>
<!--Optional:-->
<nor:PostalCode></nor:PostalCode>
```

```
ruleset
  Rule 1
    if
      (nor:Employees/nor:Country="USA") and (nor:Employees/nor:Region="CA" or nor:Employees/nor:Reg
    then
      assignment
  Rule 2
    if
      (nor:Employees/nor:Country="USA") and (nor:Employees/nor:Region="VA" or nor:Employees/nor:Reg
    then
  Rule 3
    if
      (nor:Employees/nor:Country="UK") and (nor:Employees/nor:ReportsTo="")
    then
```

```
<assignment xmlns="http://rules/commontypes" action_id="000c2975-a2a2-11e0-ffc3-e71d5adb1fe9"
status="SUCCESS" >
    <expression>
        <LHS>nor:Employees/nor:ReportsTo</LHS>
        <RHS>2</RHS>
    </expression>
</assignment>
```

6. Click in the tree view on the *conditions* and *assignments* to see more details.
7. Go to the *Changed Object* tab to verify the object after the rule(s) and corresponding assignments are performed.
8. Change the object details to test correct behavior of the other rules.

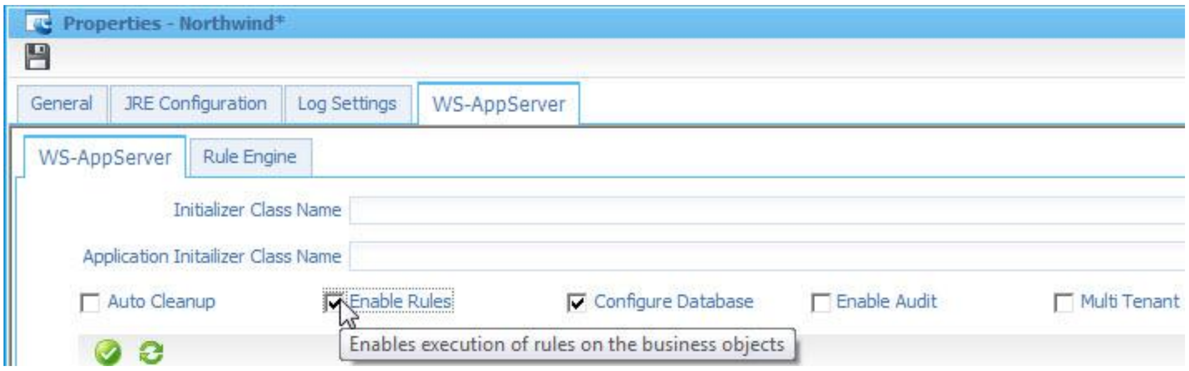
NOTE
When you selected to test rules, the rule is automatically published to the organization, to make it available at runtime.

3.4.2 Configuring Service Container for Rule Logic

The decision table you created before is associated with the employees object of a WS-AppServer package. Subsequently, if you want to make the rules active at runtime, you need to configure the corresponding WS-AppServer service container such that it will enable the rules.

1. Open the *System Resource Manager*.
2. Right click the *Northwind* service container and select *Properties*.
3. Go to the *WS-AppServer* tab.

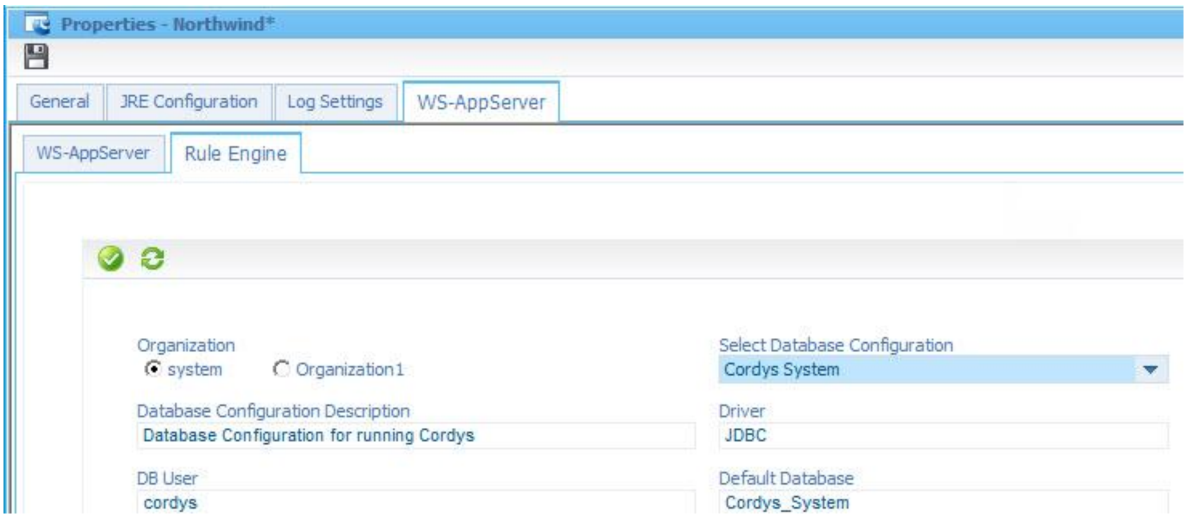
4. Check *Enable Rules*.



Initializer Class Name
Application Initailizer Class Name
☐ Auto Cleanup ☑ Enable Rules ☑ Configure Database ☐ Enable Audit ☐ Multi Tenant

Enables execution of rules on the business objects

5. Go to the *Rule Engine* tab.
6. Select the radio button **system** organization.
7. *Select Database Configuration* **Cordys System** as defined in the system organization.



Organization: [✓ system] Organization1

Select Database Configuration: Cordys System

Database Configuration Description: Database Configuration for running Cordys

Driver: JDBC

DB User: cordys

Default Database: Cordys_System

8. Save the *Northwind* service container properties.
9. Check *Restart Service Container and click* **Yes**.

Which information does the WS-AppServer container need from the Rule Repository?

____

____

3.4.3 Decision Tables at Run Time

In this part you will create a user interface to test runtime behavior of the modeled decision table.

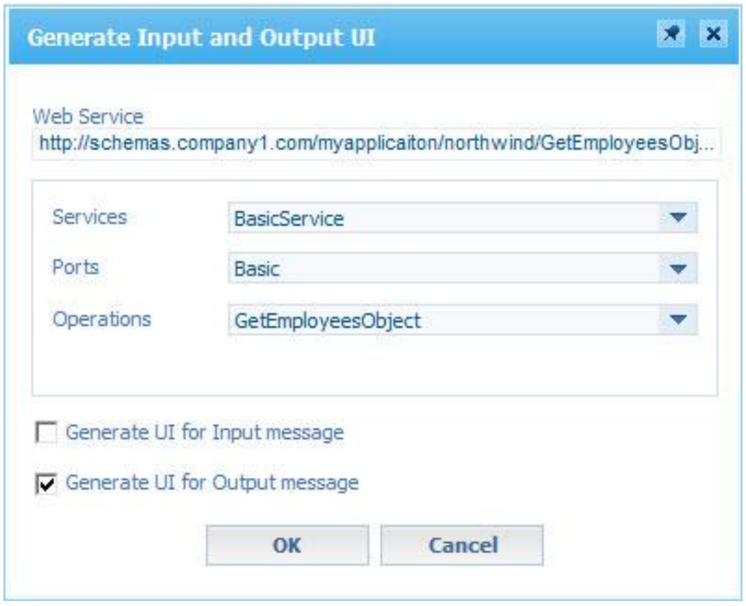
1. If closed, open the *Workspace Documents*.

2. Create a new *User Interface* with the following values:

Field	Value
Name	Maintain Sales Employees
Description	Maintain sales employees
Location	My Application project/User Interfaces/com/companyX/myapplication

3. Add the web services operation *GetEmployeesObject* from the *Northwind → Basic* web service interface.

4. Click **OK**, with only *Generate UI for Output message* checked.



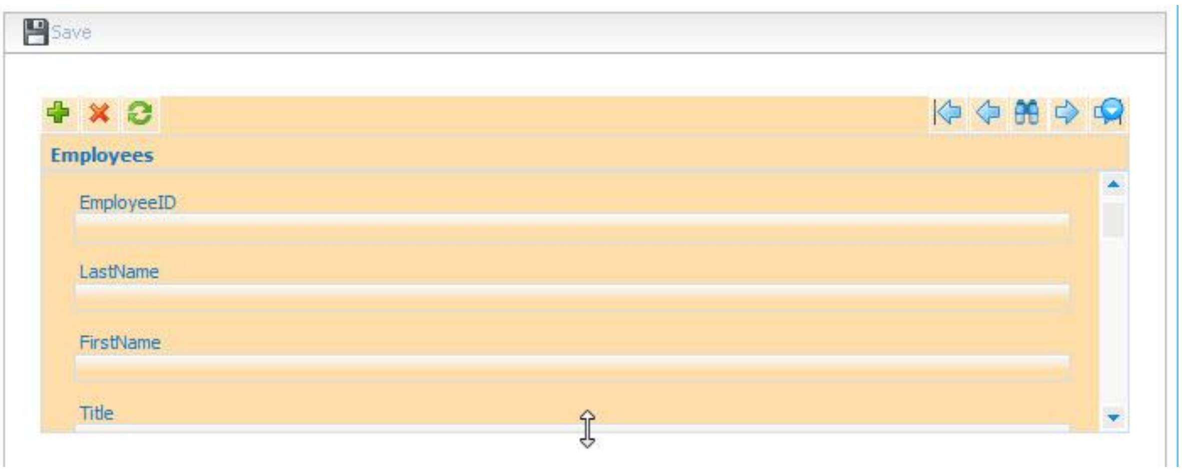
Generate Input and Output UI

Web Service
http://schemas.company1.com/myapplicaiton/northwind/GetEmployeesObj...

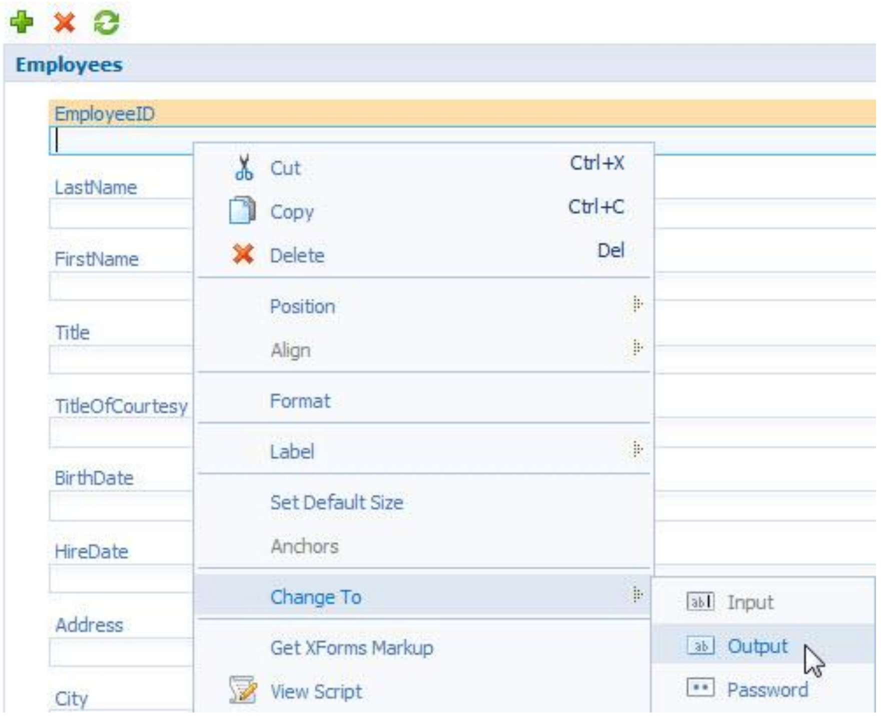
Services: BasicService
Ports: Basic
Operations: GetEmployeesObject

[ ] Generate UI for Input message
[✓] Generate UI for Output message

5. Resize the *Employees* groupbox to your needs.



6. Right click in the *EmployeeID* field and select *Change To → Output*.



7. Save the user interface.

8. Click **Preview** ( ).

NOTE

The preview works the same as a deployed user interface hence you can test the rules from the preview of the user interface.

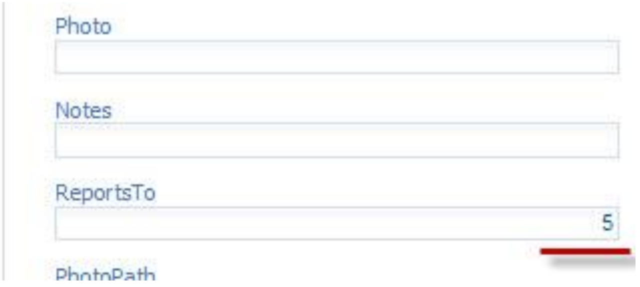
9. Click **Insert**, to insert a new employee.

10. Insert yourself as employee, enter values for the fields:

Field	Value
Country	USA
Region	WA
ReportsTo	empty, *do not enter a value*

11. Click **Save**.

As a result of the decision case (applied when inserting a new employee), the *ReportsTo* field will have the correct manager, in this example 5, filled in.



Would the rules also work when you had used the GetEmployeesObjects operation instead?

3.5 Using a Business Process as Rule Action

In this exercise you will create a simple business process which you will use later as action in a rule. In general, you can use any business process that as action. For example, as a result of inserting a new employee (using the form you created in the preceding exercise), this triggers the new hire process you built before.

3.5.1 Prerequisites

1. This exercise assumes you have finished the module Developing Processes. If you did not perform that module you can safely ignore this exercise.

3.5.2 Qualify & Analyze

In this exercise, we will explain the situation for which you will model and implement a business process.

1. Read the analysis:

   The process is a fragment of a messaging process. In your organization, information about news and events is sent to update the employees. They can subscribe to receive the messages.

   You will use this fragment of the messaging process as an example to automatically inform employees when a new sales employee is engaged.

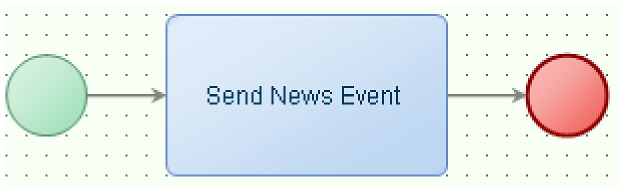
3.5.3 Creating the Business Process Model

1. If closed, open the *Workspace Documents*.
2. Create a new *Business process Model* with the following details:

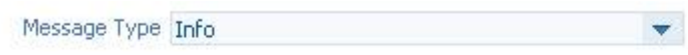
Field	Value
Name	NewsEvent
Description	Send news event to subscribed people
Location	My Application project/Business Process Models/com/companyX/myapplication

3. Open the process properties.
4. Change the namespace to:
   http://schemas.companyX.com/myapplication/salesprocesses

5. Model the process design like this:

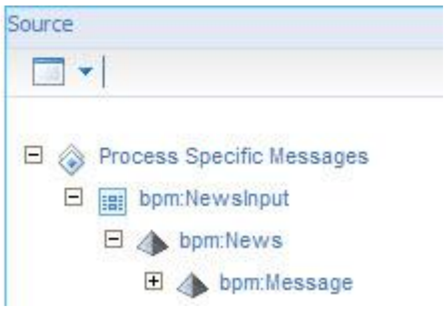


6. Right click the activity and select *Insert → User Interface*.
7. Select the *Notify Application Service*.
8. Open the properties of the activity and set the *Message Type* to *Info*.


Message Type Info

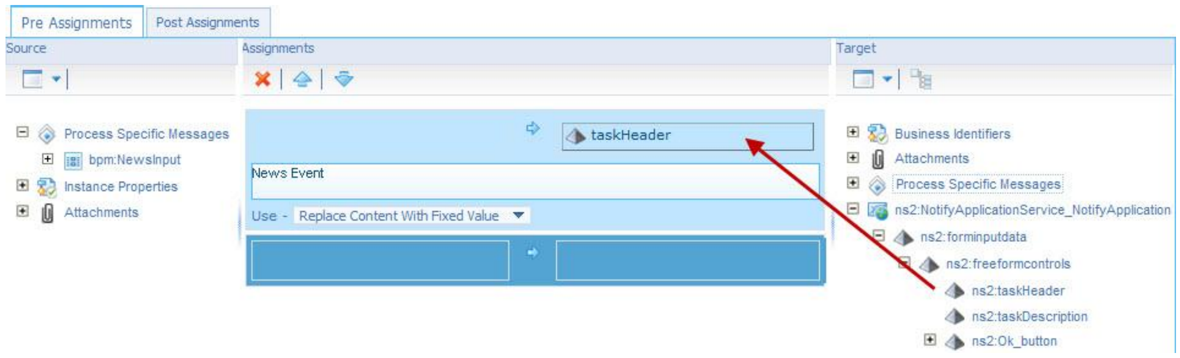
9. Go to the tab page *Workflow Model* and change the *Notification Subject* to **News Event**.

10. Go to the *Message Map* tab.
11. Right click *Process Specific Messages* and select *Create Message*.
12. Enter the name: **NewsInput**.
13. Add an element to it, named: **News**.
14. Add an element to it, named: **Message**.



15. Open the properties of the *Start Event*, select this message as *Input Message*.
16. Return to the *Message Map* tab page.

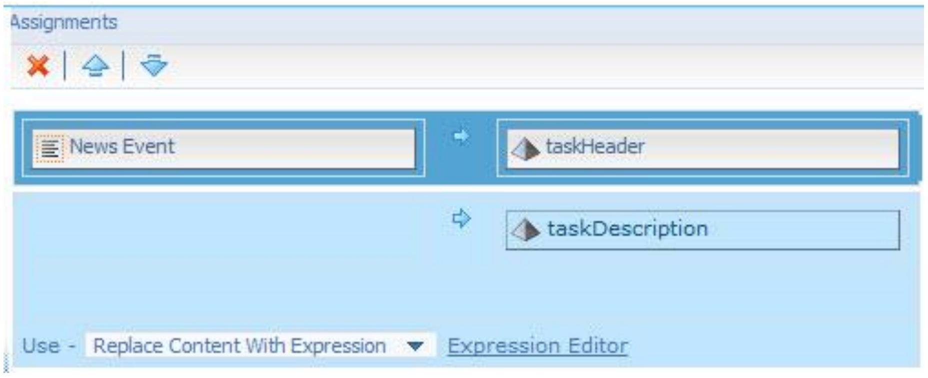
17. Select the *Send News Event* activity.
18. Expand the input message of the activity *NotifyApplicationService_xxxx*.
19. Create an assignment for the element *taskHeader* with the text: **News Event**.



News Event
Use - Replace Content With Fixed Value

20. Right click *taskDescription* and select *Create Assignment*.

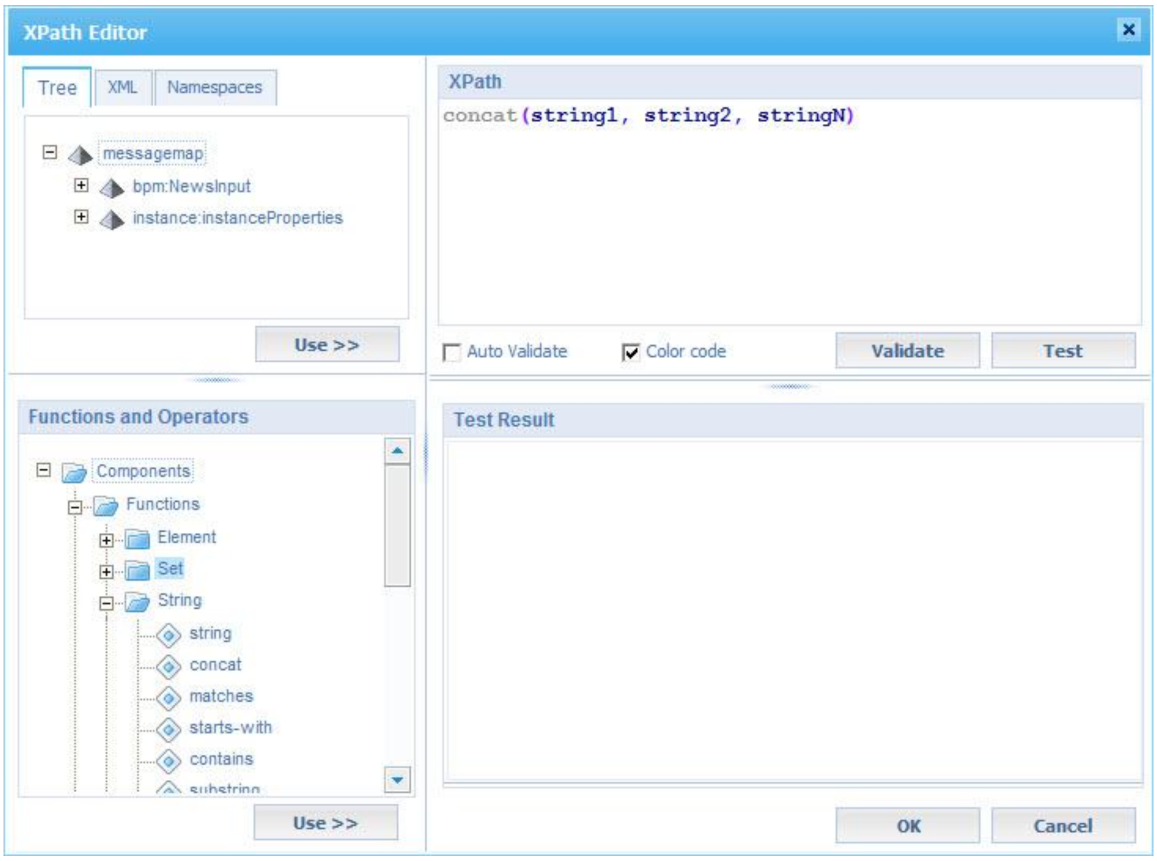
21. In the *Use* select box select *Replace Content With → Expression*.



22. Click Expression Editor, to open the XPath expression editor.

You will model the following news line:
Message 'actual message text' at yyyy-MM-dd (HH:mm)

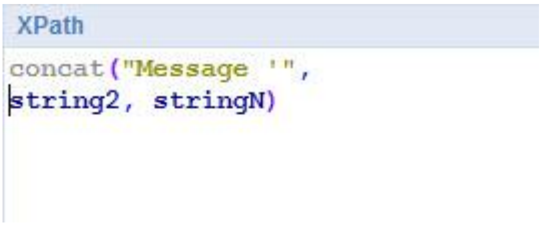
23. Drag and drop the *concat* (string function) from the functions group box into the *XPath* textbox:



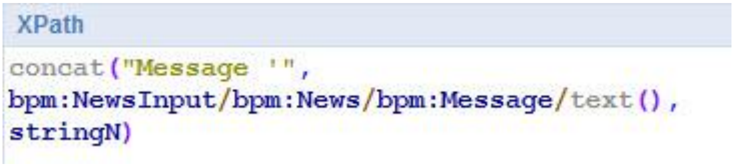
```
concat(string1, string2, stringN)
```

NOTE
For readability, you can add line feeds in the XPath expression. When used outside the string parts, they are not included in the result of the expression. You can use the **Test** button to check the resulting string.

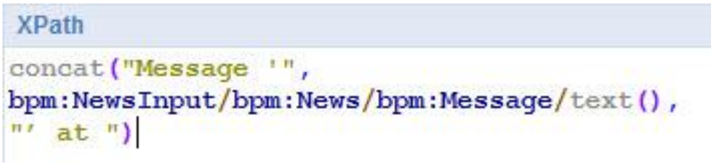
24. Replace *string1* with the text **"Message '"**


XPath
```
concat("Message '",
string2, stringN)
```

25. Replace *string2* with the *bpm:Message* element from Message map tree.

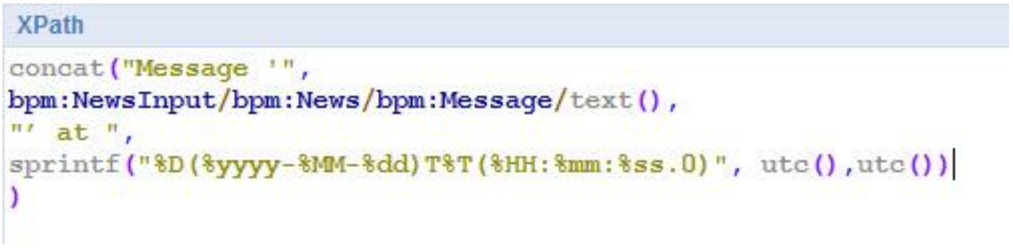

XPath
```
concat("Message '",
bpm:NewsInput/bpm:News/bpm:Message/text(),
stringN)
```

26. Replace *stringN* with **"' at "**.


XPath
```
concat("Message '",
bpm:NewsInput/bpm:News/bpm:Message/text(),
"' at ")
```

27. Add a **comma** at the end of the *concat* function before the closing parentheses.

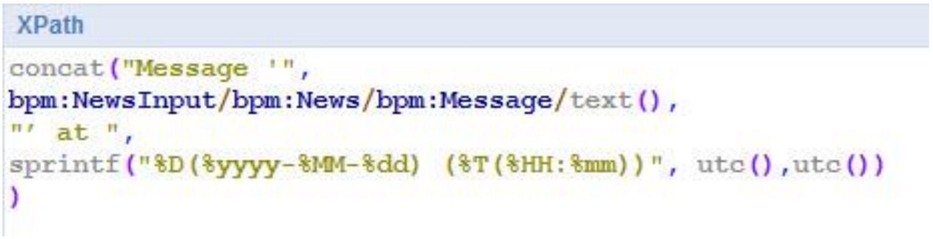
28. Drag and drop the *sprintf* (date-time function) after the *comma*.


XPath
```
concat("Message '",
bpm:NewsInput/bpm:News/bpm:Message/text(),
"' at ",
sprintf("%D(%yyyy-%MM-%dd)T%T(%HH:%mm:%ss.0)", utc(),utc())
)
```

29. In the *sprint* function enter the following changes:

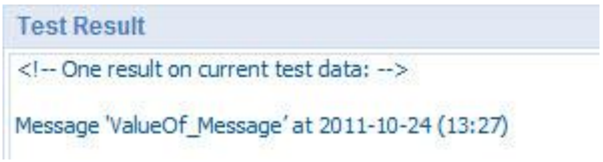
- Change ***)T%T(*** into **) %T(**
- Remove ***:%ss.0***
- Put parentheses around ***%T(%HH:%mm)*** , resulting in (***%T(%HH:%mm)***)

30. Your XPath expression contains four parts to be concatenated:


XPath
```
concat("Message '",
bpm:NewsInput/bpm:News/bpm:Message/text(),
"' at ",
sprintf("%D(%yyyy-%MM-%dd) (%T(%HH:%mm))", utc(),utc())
)
```

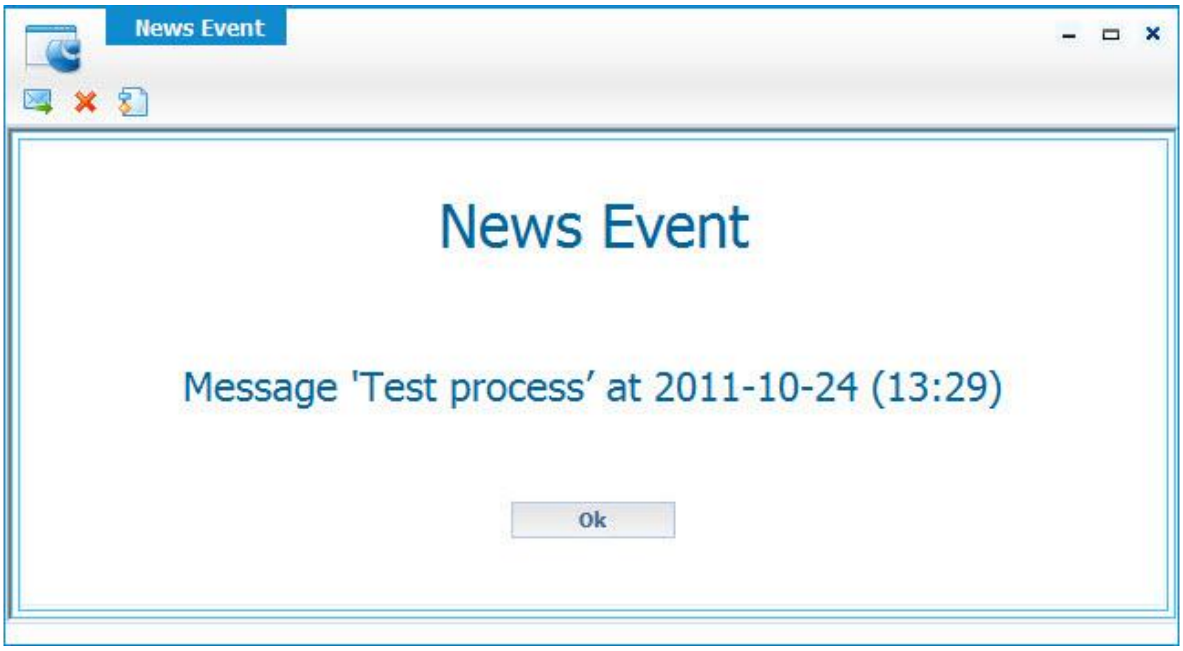
```
concat("Message '",
bpm:NewsInput/bpm:News/bpm:Message/text(),
"' at ",
sprintf("%D(%yyyy-%MM-%dd) (%T(%HH:%mm))", utc(),utc()))
```

31. Click **Test** to check the result is the desired message:


Test Result
```
<!-- One result on current test data: -->

Message 'ValueOf_Message' at 2011-10-24 (13:27)
```

32. Click **Ok**.
33. Save, validate and publish the process.
34. Run the process and provide a message line.
35. Check via the process instances view that your process has completed successfully.
36. Check your inbox, work list *Notifications*, whether you have received the message.



News Event

Message 'Test process' at 2011-10-24 (13:29)

What is the usage of the sprint function (use the documentation)?

____

____

3.6 Rules on WS-AppServer Classes

While a decision table provides you an “easy to use” matrix to define rules, using the Rule documents enables you to create more technical rules including functions and nested structures.

In this exercise you will create a rule that will send a news event when a new sales employee is hired.

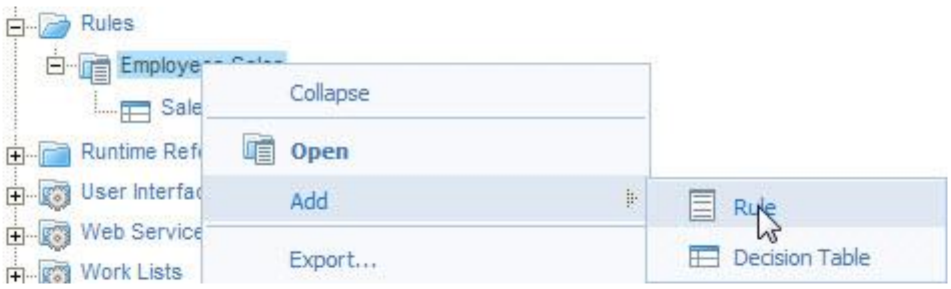
3.6.1 Creating the Rule Group

Similar to the decision table you created before, a rule must be part of a rule group. You will use the same rule group as the decision table. You can find more details about Creating the Rule Group on page 6.

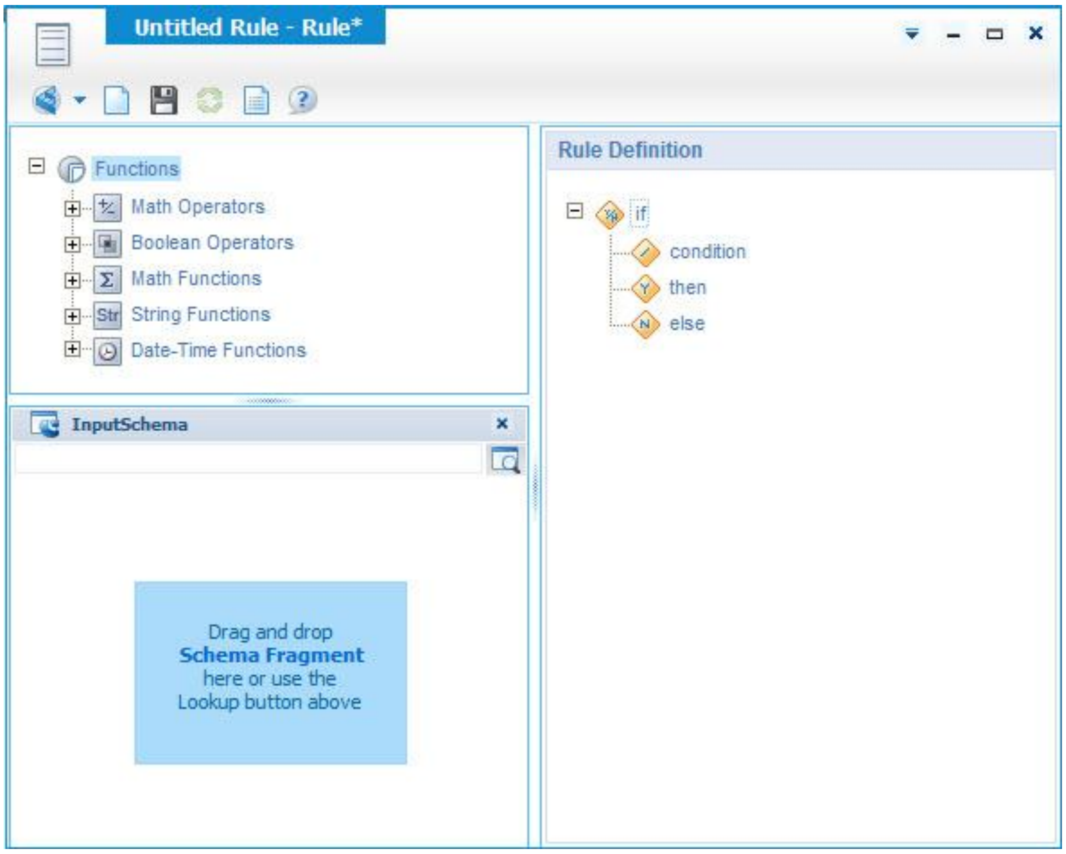
3.6.2 Creating the Rule

1. If closed, open the *Workspace Documents*.
2. Navigate to *My Application project → Rules*.

3. Right click the *Employee Sales* rule group and select *Add → Rule*.

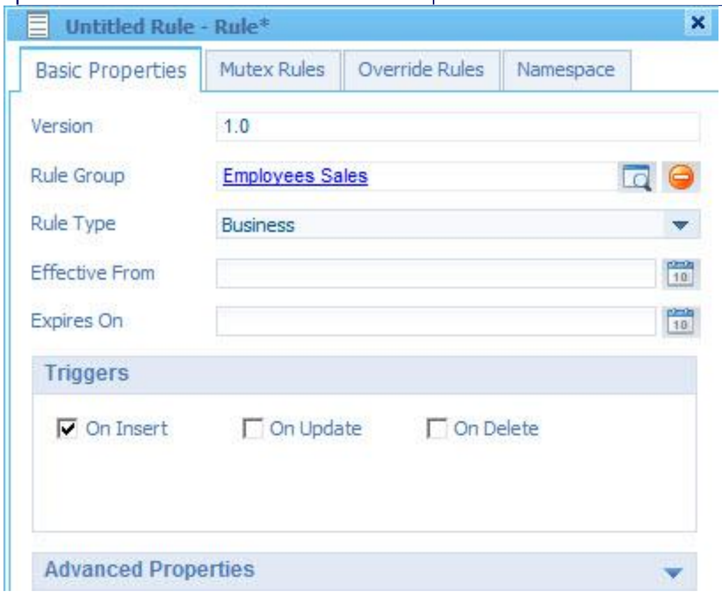


The rule editor opens:



4. In the toolbar click **Show Rule Properties**( ) and provide the following values:

Field	Value
Rule Group	Employees Sales
Rule Type	Business (perform action after the backend is updated, so the employee is actually inserted)
Triggers	OnInsert - checked OnUpdate – unchecked OnDelete – unchecked this rule only applies in case of new sales employees


Untitled Rule - Rule*

Basic Properties | Mutex Rules | Override Rules | Namespace

Version: 1.0
Rule Group: Employees Sales
Rule Type: Business
Effective From:
Expires On:

Triggers
☑ On Insert ☐ On Update ☐ On Delete

Advanced Properties

Give an example of a possible usage of the activation dates.

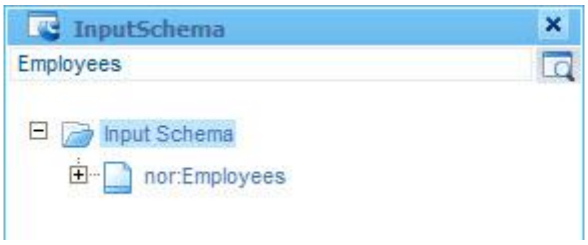
____

____

5. Click the **Look up** button, for the *InputSchema*.

6. Navigate to *WS-AppServer → com → companyX → myapplication → northwind → Northwind → Employees.xsd*.

7. Select the *Employees* (XML schema fragment) and click **OK**:


InputSchema

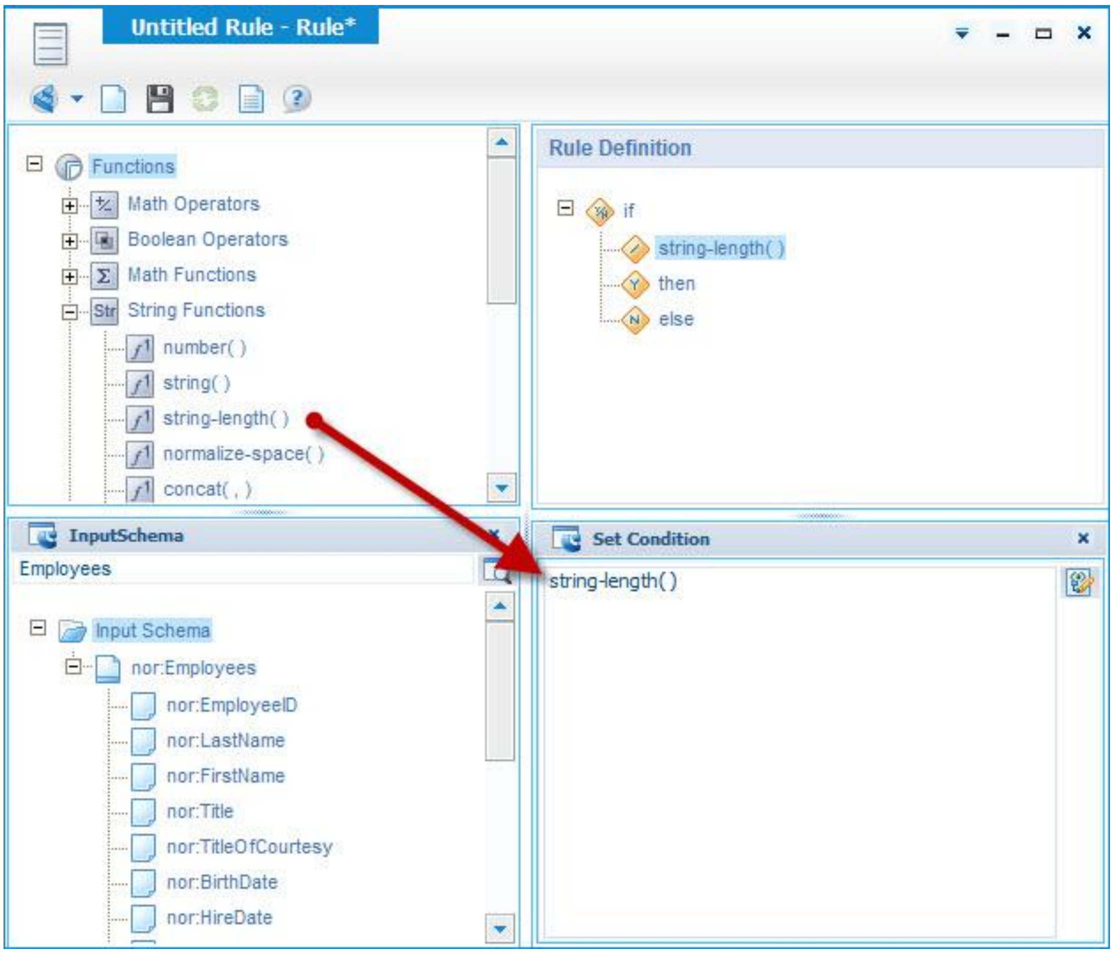
Employees

Input Schema
nor:Employees

You will create a condition which is nonsensical but will illustrate the possibilities of a condition. The condition will check whether the last name has less than six characters.

8. In the *Rule Definition* area click on *Condition* (condition).

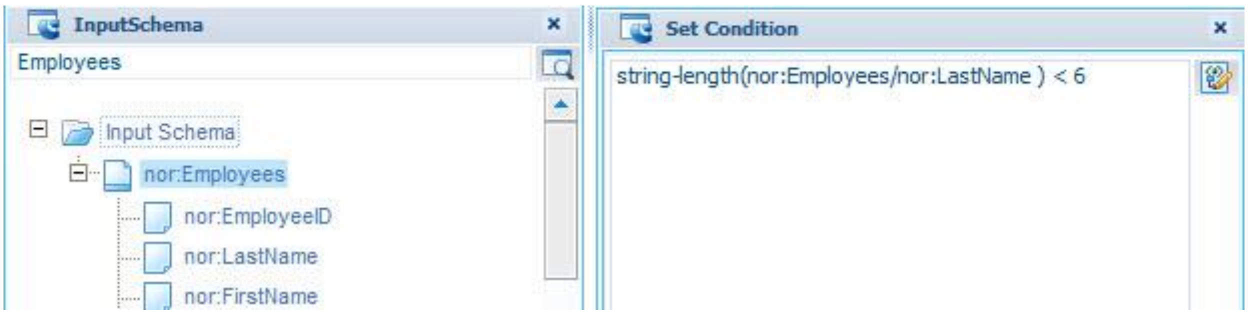
9. From the *Functions* drag and drop the *string-length( )* function into the *Set Condition* textbox in the lower right part of the rules editor.



string-length( )

10. From the *InputSchema* group box, drag and drop the *LastName* field in between the parentheses of the *string-length( )* function.

11. Complete the condition by adding **< 6** at the end.



string-length(nor:Employees/nor:LastName) < 6

You can also use the expression editor to model the expression, which includes having options to test your function.

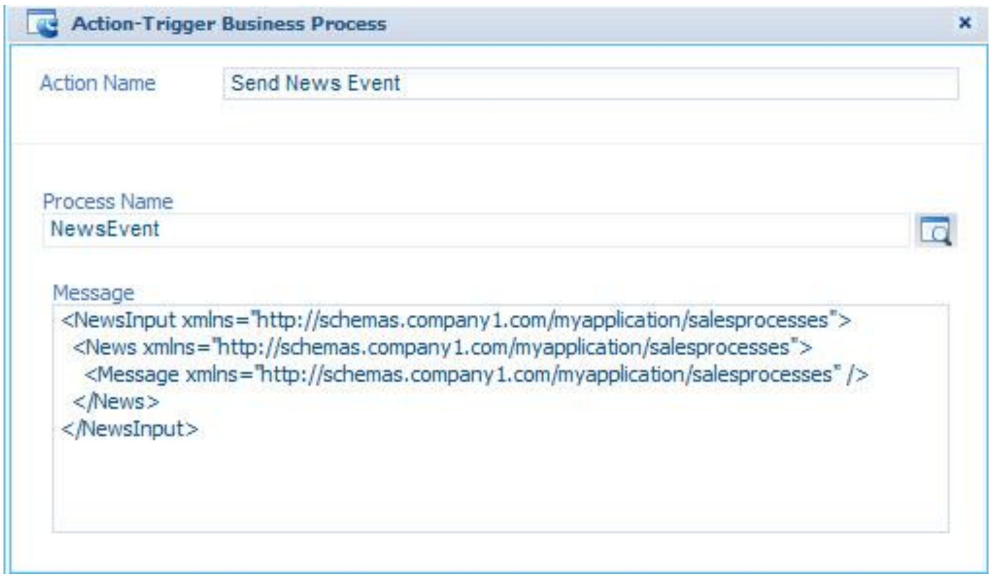
What condition can be used to make the conditions always true?

---

12. In the *Rule Definition* group box, right click *then* (then) and select *Actions → Trigger Business Process*.

13. Provide *Action Name*: **Send News Event**.

14. Use the **Look up** button and select the *NewsEvent* process.



Action Name: Send News Event

Process Name: NewsEvent

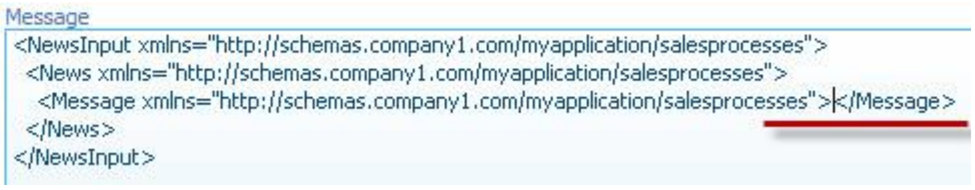
Message

```
<NewsInput xmlns="http://schemas.company1.com/myapplication/salesprocesses">
  <News xmlns="http://schemas.company1.com/myapplication/salesprocesses">
    <Message xmlns="http://schemas.company1.com/myapplication/salesprocesses" />
  </News>
</NewsInput>
```

You will now configure the process start message using the actual values of the employees object.

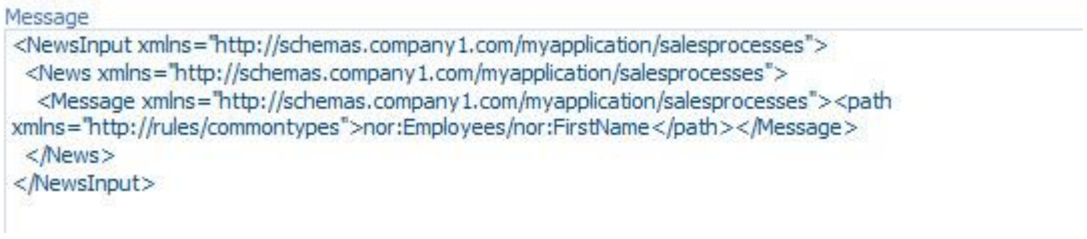
The input message can be modeled using both fixed values and fields of the business object. To distinguish between them the business object fields and functions need to be included in a path element e.g. <path> /actual/field</path>.

15. In the *Message* textbox create a closing **</Message>** tag and make sure to remove the “/” at the end of the opening tag.



```
<NewsInput xmlns="http://schemas.company1.com/myapplication/salesprocesses">
  <News xmlns="http://schemas.company1.com/myapplication/salesprocesses">
    <Message xmlns="http://schemas.company1.com/myapplication/salesprocesses"></Message>
  </News>
</NewsInput>
```

16. Drag and drop the *FirstName* field (InputSchema) into the *Message* element.

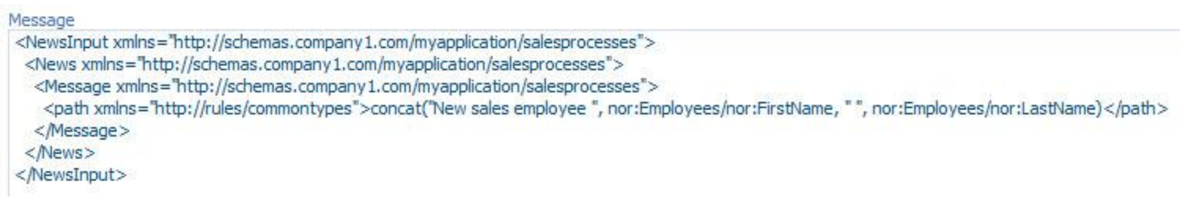


```
<NewsInput xmlns="http://schemas.company1.com/myapplication/salesprocesses">
  <News xmlns="http://schemas.company1.com/myapplication/salesprocesses">
    <Message xmlns="http://schemas.company1.com/myapplication/salesprocesses"><path
xmlns="http://rules/commontypes">nor:Employees/nor:FirstName</path></Message>
  </News>
</NewsInput>
```

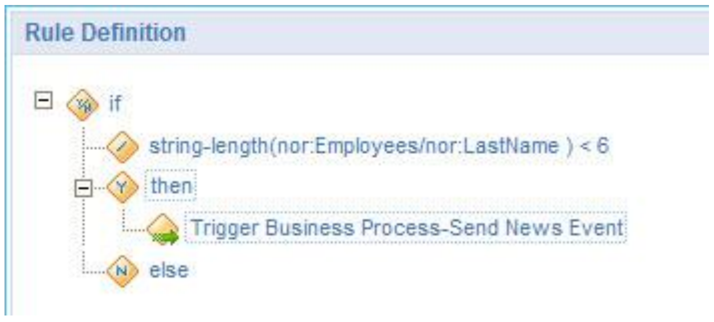
You can see that the element is added in a path element. Therefore when using functions and multiple fields you sometimes need to remove the path elements.

17. The message must be structured like this:

```
<path>concat("New sales employee ", FirstName, " ",
LastName)</path>
```


Message
```
<NewsInput xmlns="http://schemas.company1.com/myapplication/salesprocesses">
  <News xmlns="http://schemas.company1.com/myapplication/salesprocesses">
    <Message xmlns="http://schemas.company1.com/myapplication/salesprocesses">
      <path xmlns="http://rules/commontypes">concat("New sales employee ", nor:Employees/nor:FirstName, " ", nor:Employees/nor:LastName)</path>
    </Message>
  </News>
</NewsInput>
```

18. Your Rule Definition should look like this:


Rule Definition

```
if
  string-length(nor:Employees/nor:LastName ) < 6
  then
    Trigger Business Process-Send News Event
  else
```

19. Save the rule with the following values:

Field	Value
Name	NewSalesEmployee
Description	New sales employee

20. Publish the rule to the organization.

3.7 Testing the Rule

You can “test” your rule similar to testing the decision tables as both are conditions related to business object and trigger certain actions, for more information see Using Decision Table on page 18.

3.7.1 Configure Service Container for Rule logic

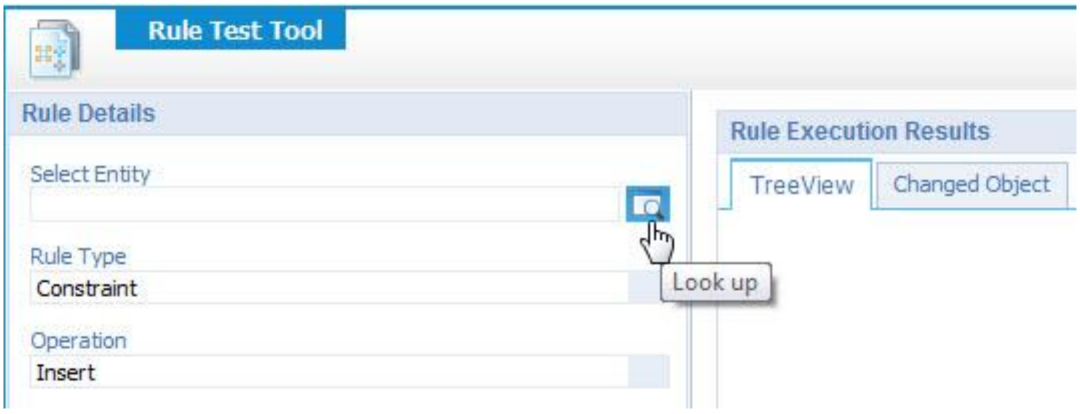
The configuration for a rule is similar to a decision table, for more information see Configuring Service Container for Rule Logic on page 19.

3.7.2 Using the Rule Test Tool Artifact

Rules can be tested directly via the rule test tool available from the My Application App Palette.

1. From the *My Application App Palette* select the *Rule Test Tool* (Rule Test Tool).

2. Click *Select Entity* **Look up** button.



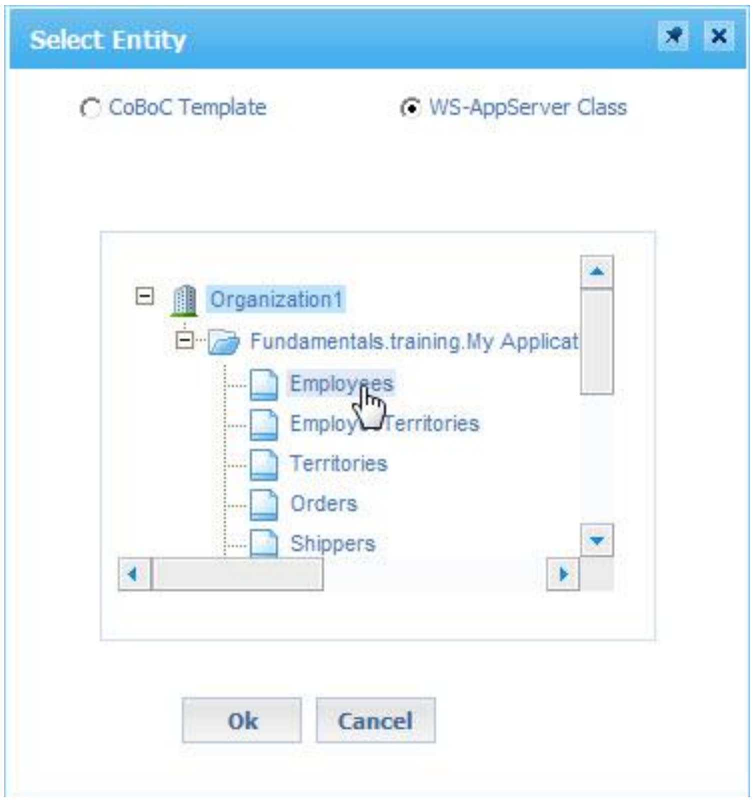
Select Entity

Rule Type: Constraint

Operation: Insert

3. Check *WS-AppServer Class.*

4. Navigate to *Organization X → Fundamentals.training.... → Employees*.



( ) CoBoC Template [✓ WS-AppServer Class]

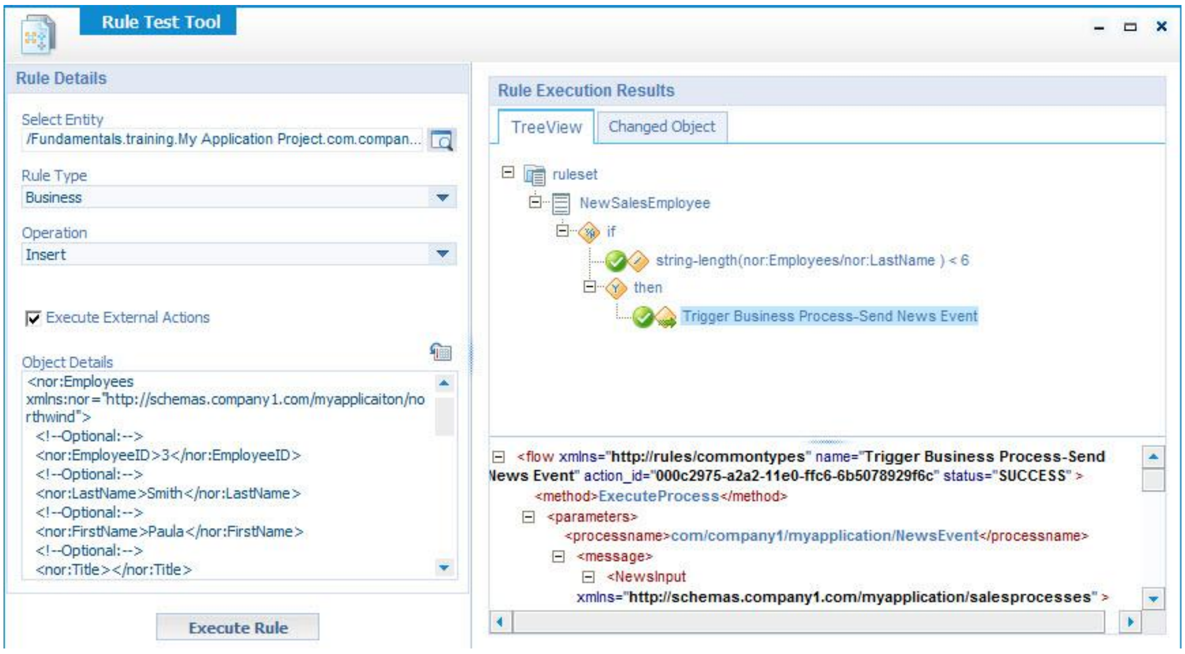
5. Select *Employees* and click **OK**.

6. Select *Rule Type* ***Business***.

7. Make sure *Execute External Actions* is checked, since you want the *News Event* process to be started.

8. Provide a first name and a last name with less than 6 characters.

9. Click **Execute Rule**.



Select Entity: /Fundamentals.training.My Application Project.com.compan...

Rule Type: Business

Operation: Insert

[✓] Execute External Actions

Object Details

```
<nor:Employees
xmlns:nor="http://schemas.company1.com/myapplicaiton/no
rthwind">
  <!--Optional:-->
  <nor:EmployeeID>3</nor:EmployeeID>
  <!--Optional:-->
  <nor:LastName>Smith</nor:LastName>
  <!--Optional:-->
  <nor:FirstName>Paula</nor:FirstName>
  <!--Optional:-->
  <nor:Title></nor:Title>
```

```
ruleset
  NewSalesEmployee
    if
      string-length(nor:Employees/nor:LastName ) < 6
    then
      Trigger Business Process-Send News Event
```

```
<flow xmlns="http://rules/commontypes" name="Trigger Business Process-Send
News Event" action_id="000c2975-a2a2-11e0-ffc6-6b5078929f6c" status="SUCCESS" >
    <method>ExecuteProcess</method>
  <parameters>
      <processname>com/company1/myapplication/NewsEvent</processname>
    <message>
      <NewsInput
      xmlns="http://schemas.company1.com/myapplication/salesprocesses" >
```

Is there a new notification in your inbox?

3.7.3 Runtime Action

In this part you will by means of a user interface insert a new employee and as a result you will see the effect of both the decision table and the rule.

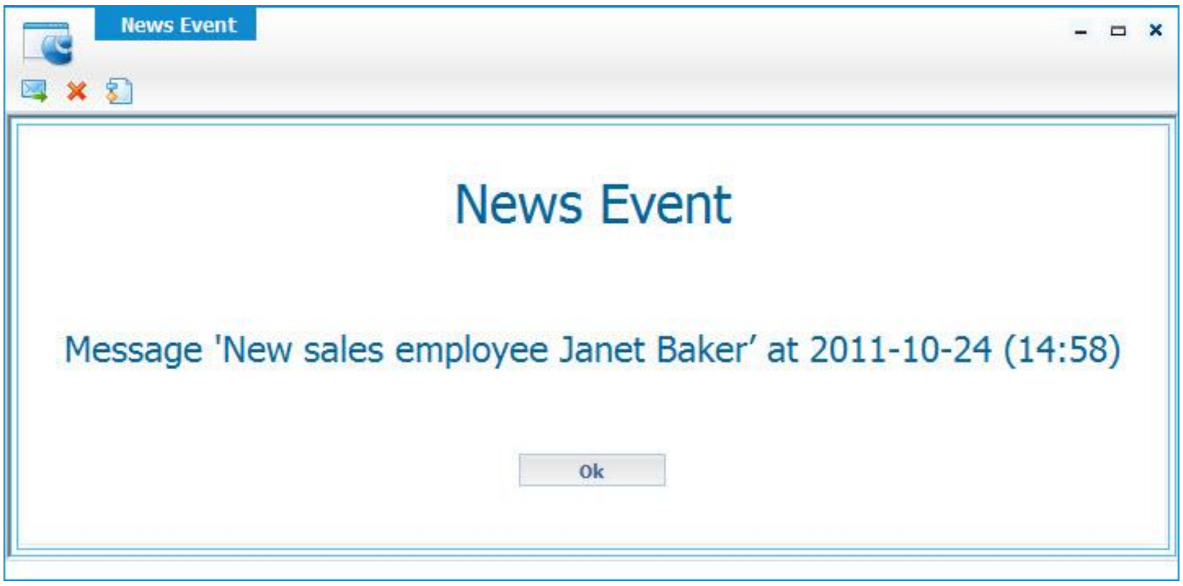
1. In the *Workspace Documents* navigate to *User Interfaces → com → companyX → myapplication*.
2. Open the *Maintain Sales Employee* form.
3. In the toolbar, click Preview (🖥).

4. Click **Insert**, to insert a new employee
5. Insert yourself as employee, enter the following values for the fields:

Field	Value
LastName	Your last name, however less than 6 characters!
FirstName	Your first name
Country	**USA**
Region	**WA**
ReportsTo	empty, *do not enter a value*

6. Click **Save**.

7. The *ReportsTo* field should have the correct manager filled (from the decision table rules) and in your inbox a new News Event message about the new employee should be available.



News Event

Message 'New sales employee Janet Baker' at 2011-10-24 (14:58)

3.8 Decision Tables in a Business Process Models

In this exercise you will study the usages of a decision table in a business process.

You will create a decision table that is not linked to a backend but serves as a standalone decision matrix.

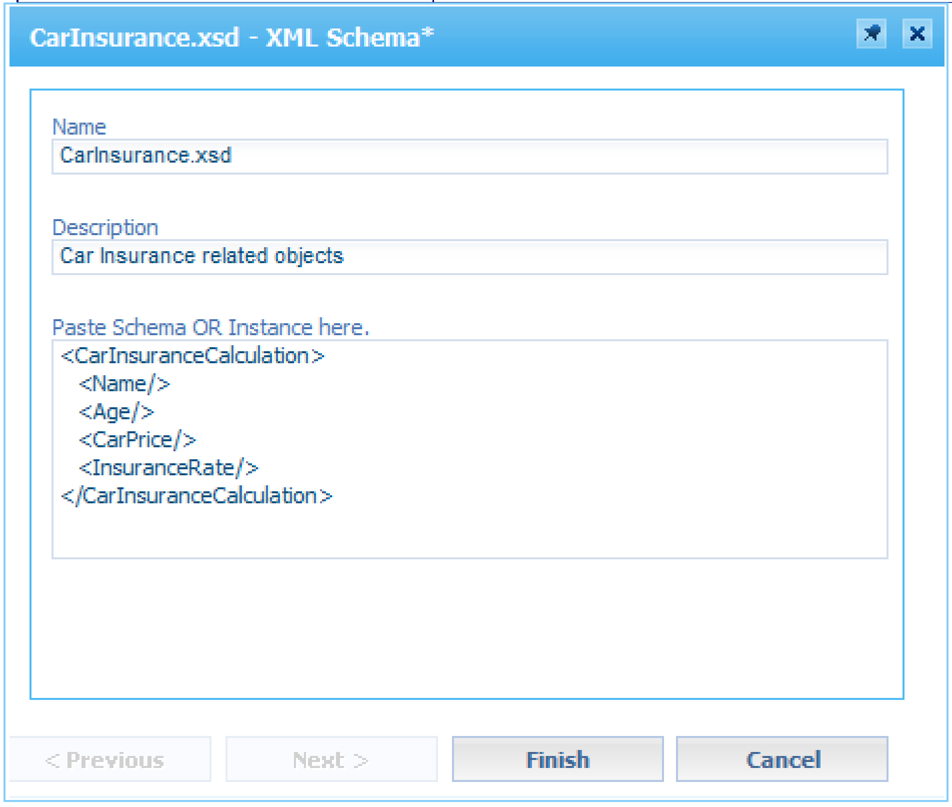
Both the process and the decision table contain only a fragment of the whole process and decision matrix.

3.8.1 Creating XML Schema

1. In the Workspace Documents, navigate to *XML Schemas → Common Objects*.

2. Add a new *XML Schema* to the common folder providing the following details:

Field	Value
Name	CarInsurance.xsd
Description	Car Insurance related objects
Paste Schema	<CarInsuranceCalculation> <Name/> <Age/> <CarPrice/> <InsuranceRate/> </CarInsuranceCalculation>


CarInsurance.xsd - XML Schema*

Name
CarInsurance.xsd

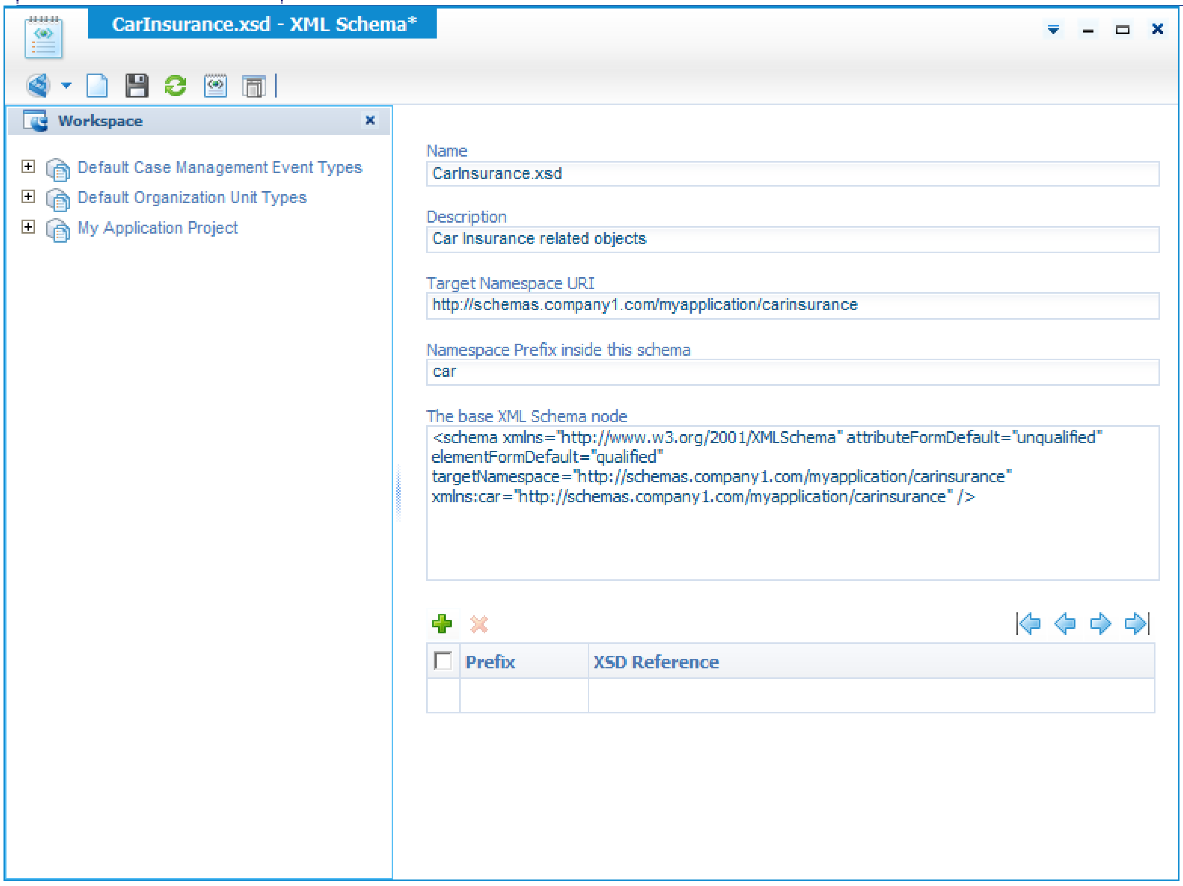
Description
Car Insurance related objects

Paste Schema OR Instance here.
```
<CarInsuranceCalculation>
  <Name/>
  <Age/>
  <CarPrice/>
  <InsuranceRate/>
</CarInsuranceCalculation>
```

3. Click **Finish**.

4. Providing the following details:

Field	Value
Target namespace URI	http://schemas.companyX.com/myapplication/carinsurance
Namespace prefix	car



CarInsurance.xsd - XML Schema*

Workspace

- Default Case Management Event Types
- Default Organization Unit Types
- My Application Project

Name
CarInsurance.xsd

Description
Car Insurance related objects

Target Namespace URI
http://schemas.company1.com/myapplication/carinsurance

Namespace Prefix inside this schema
car

The base XML Schema node
```
<schema xmlns="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://schemas.company1.com/myapplication/carinsurance"
xmlns:car="http://schemas.company1.com/myapplication/carinsurance" />
```

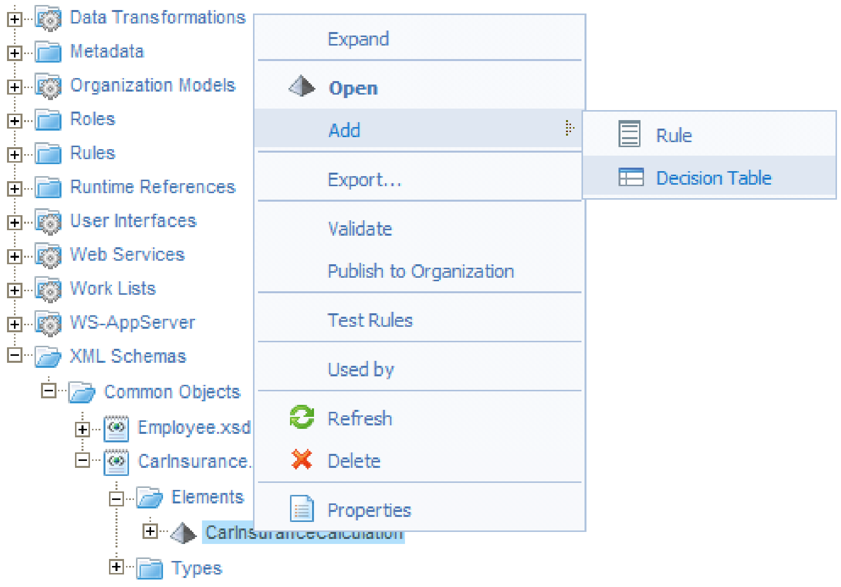
	Prefix	XSD Reference

5. Save and close the XML Schema.

3.8.2 Creating Decision Table

1. In the Workspace Documents, navigate to *XML Schemas → Common Objects → CarInsurance.xsd → Elements → CarInsuranceCalculation*.

2. Right click *CarInsuranceCalculation* and select *Add → Decision table*.



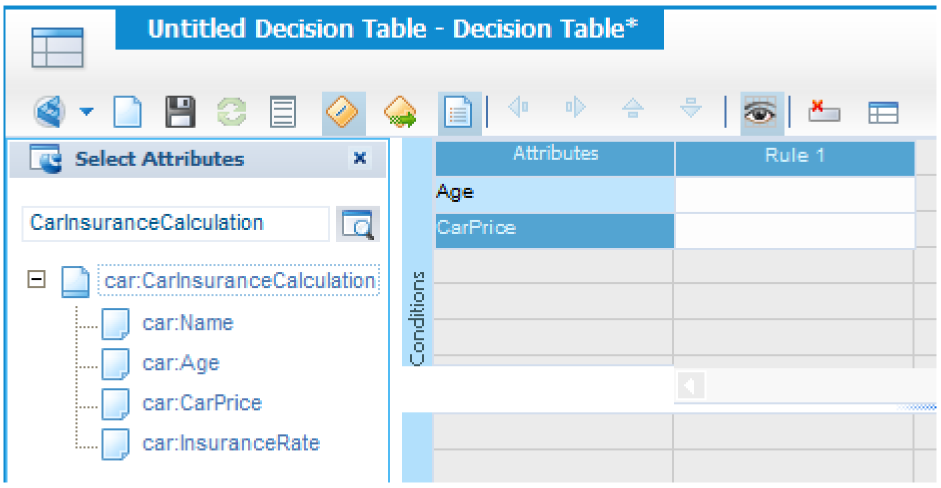
3. Click **Show Decision Table Properties** (), in the toolbar.

4. In the properties window, provide the following values:

Field	Value
Rule Group	Create one with the following details: Name: Car Insurance Location: My Application project/Rules

Are the decision table properties Rule Type and Triggers having any effect?

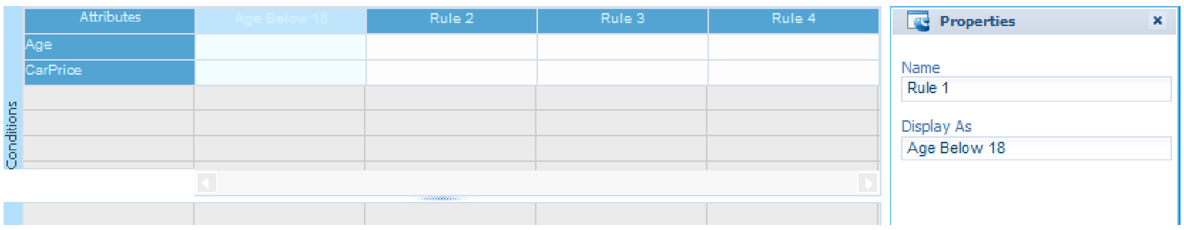
5. Drag and drop the attributes *Age* and *CarPrice* into the *Condition* section



6. Use **Add Rule** () to add 3 more rules.

7. Double click in the column header of *Rule 1*, to open the properties.

8. Change the *Display As* to **Age Below 18**.



Attributes	Age Below 18	Rule 2	Rule 3	Rule 4
Age				
CarPrice				

Conditions

Properties

Name: Rule 1

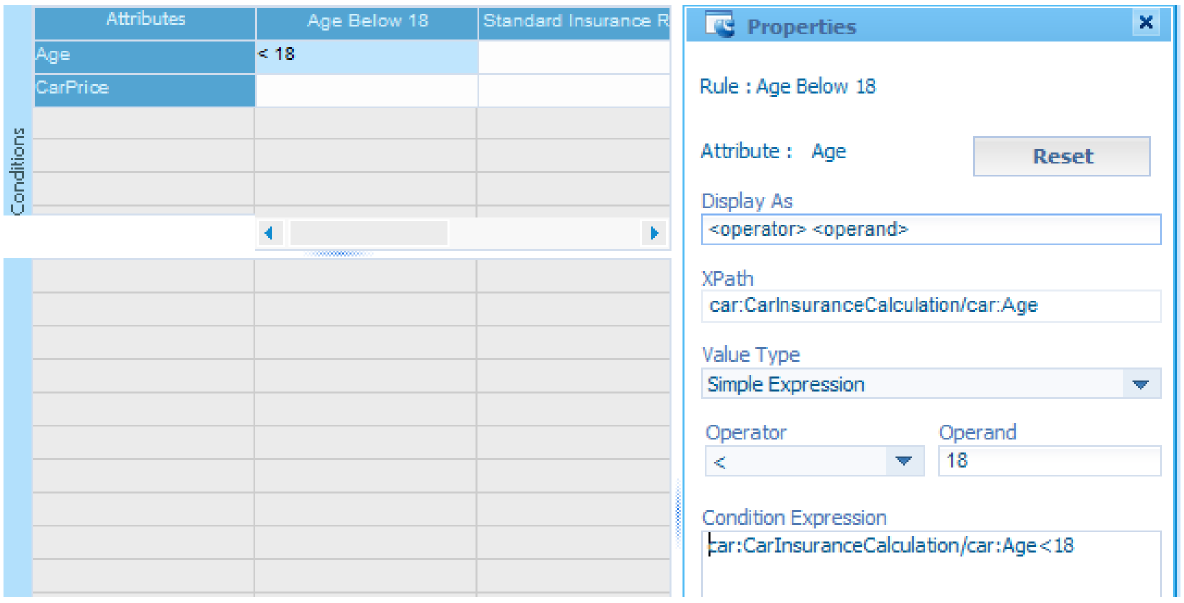
Display As: Age Below 18

9. Change the display as for the other rules:

Rule Name	Display As
Rule 2	Standard Insurance Rate
Rule 3	Luxury Cars
Rule 4	Young Driver

10. Select the condition cell rule *Age Below 18* for the attribute *Age*.

11. Select *Value Type* **Simple Expression**, Select Value **<** and provide value **18**.



Attributes	Age Below 18	Standard Insurance R
Age	< 18	
CarPrice		

Conditions

Properties

Rule : Age Below 18

Attribute : Age

Display As: <operator> <operand>

XPath: car:CarInsuranceCalculation/car:Age

Value Type: Simple Expression

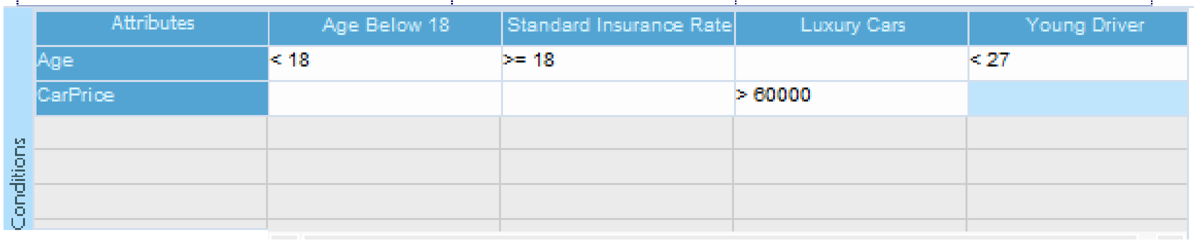
Operator: <

Operand: 18

Condition Expression: car:CarInsuranceCalculation/car:Age<18

12. For the other rules add the following conditions:

Rule Name	Attribute	Simple Expression
Standard Insurance Rate	Age	>= 18
Luxury Cars	CarPrice	> 60000
Young Drivers	Age	< 27

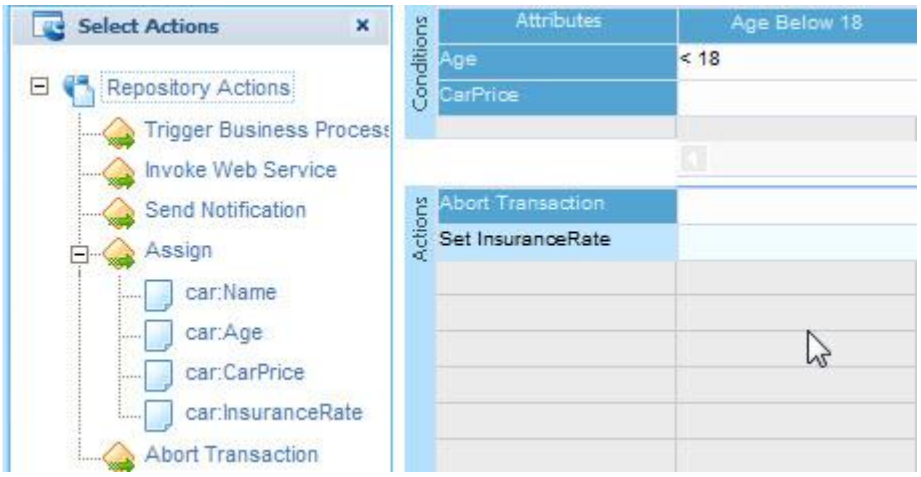


Attributes	Age Below 18	Standard Insurance Rate	Luxury Cars	Young Driver
Age	< 18	>= 18		< 27
CarPrice			> 60000	

Conditions

13. Click **Show Actions** ( ).

14. Add the actions *Abort Transaction* and the *InsuranceRate* (Assign) in the *Actions* section.



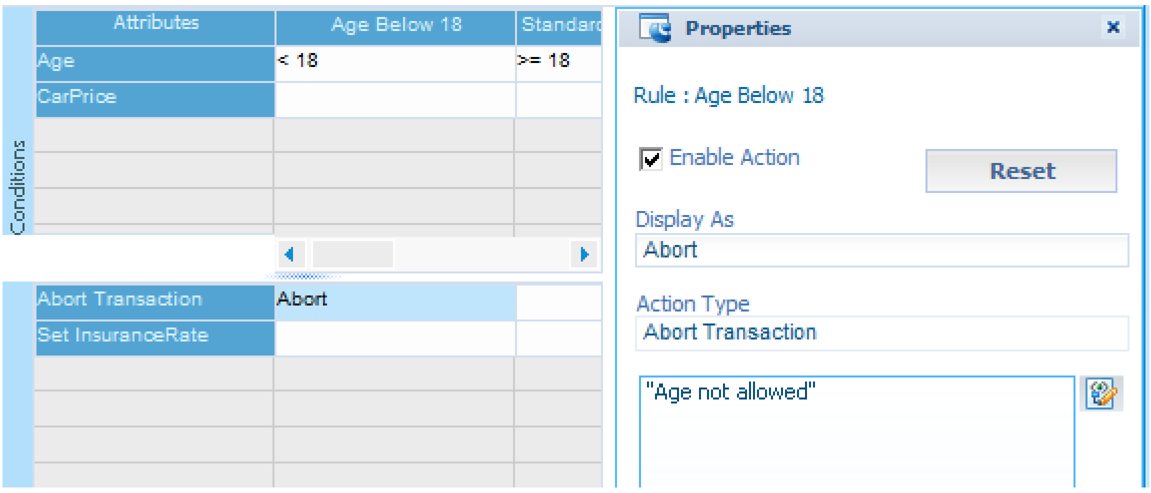
Select Actions

- Repository Actions
  - Trigger Business Process
  - Invoke Web Service
  - Send Notification
  - Assign
    - car:Name
    - car:Age
    - car:CarPrice
    - car:InsuranceRate
  - Abort Transaction

	Attributes	Age Below 18
Conditions	Age	< 18
	CarPrice	
Actions	Abort Transaction	
	Set InsuranceRate	

15. Select the Action cell of rule *Age Below 18* for the action *Abort Transaction*.

16. Provide text **"Age not allowed"**, as abort message.



	Attributes	Age Below 18	Standard
Conditions	Age	< 18	>= 18
	CarPrice		
	Abort Transaction	Abort	
	Set InsuranceRate		

Properties

Rule : Age Below 18

☑ Enable Action

Display As: Abort

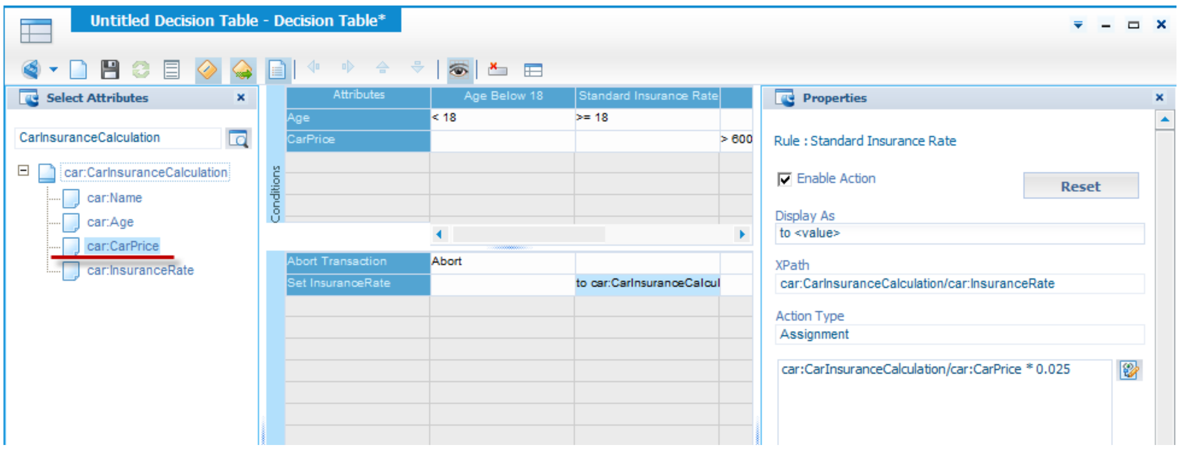
Action Type: Abort Transaction

"Age not allowed"

This action will stop evaluating the other rules in the decision table and returns a fault/abort to the caller e.g. BPM, web service.

17. Select the Action cell of rule *Standard Insurance Rate* for the action *Set InsuranceRate*.

18. Drag and drop the attribute *CarPrice* into the assignment and add *** 0.025** to it. Complete assignment: **car:CarInsuranceCalculation/car:CarPrice * 0.025**



Untitled Decision Table - Decision Table*

Select Attributes

CarInsuranceCalculation

- car:CarInsuranceCalculation
  - car:Name
  - car:Age
  - car:CarPrice
  - car:InsuranceRate

	Attributes	Age Below 18	Standard Insurance Rate	
Conditions	Age	< 18	>= 18	
	CarPrice			> 600
	Abort Transaction	Abort		
	Set InsuranceRate		to car:CarInsuranceCalcul	

Properties

Rule : Standard Insurance Rate

☑ Enable Action

Display As: to <value>

XPath: car:CarInsuranceCalculation/car:InsuranceRate

Action Type: Assignment

car:CarInsuranceCalculation/car:CarPrice * 0.025

This sets the standard insurance price to 2.5% of the car price.

19. Select the Action cell of rule *Luxury Cars* for the action *Set InsuranceRate*.

20. Drag and drop the attribute *InsuranceRate* into the assignment and add *** 1.02** to it. Complete assignment: **car:CarInsuranceCalculation/car:InsuranceRate * 1.02**

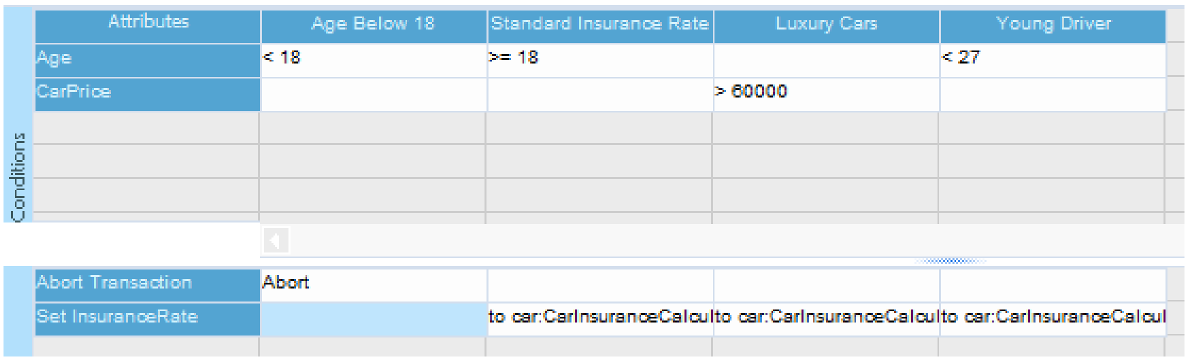
This will increase the insurance price with 2%.

21. Select the Action cell of rule *Young Drivers* for the action *Set InsuranceRate*.

22. Drag and drop the attribute *InsuranceRate* into the assignment and add *** 1.05** to it. Complete assignment: **car:CarInsuranceCalculation/car:InsuranceRate * 1.05**

This will increase the insurance price with 5% (In addition to the 2% for luxury cars if relevant).

23. The complete decision table should look like this:



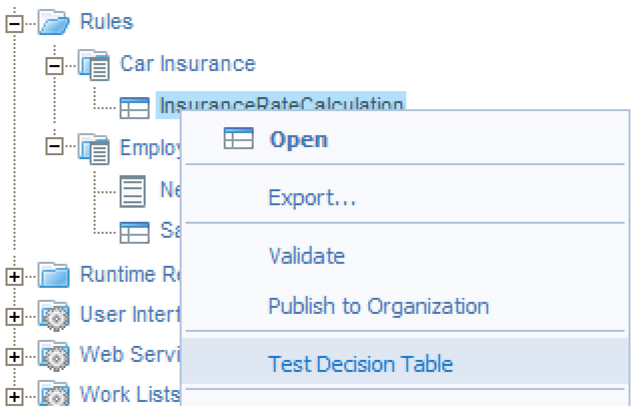
Attributes	Age Below 18	Standard Insurance Rate	Luxury Cars	Young Driver
Age	< 18	>= 18		< 27
CarPrice			> 60000	
Abort Transaction	Abort			
Set InsuranceRate		to car:CarInsuranceCalcul	to car:CarInsuranceCalcul	to car:CarInsuranceCalcul

24. Save the decision table with the following details:

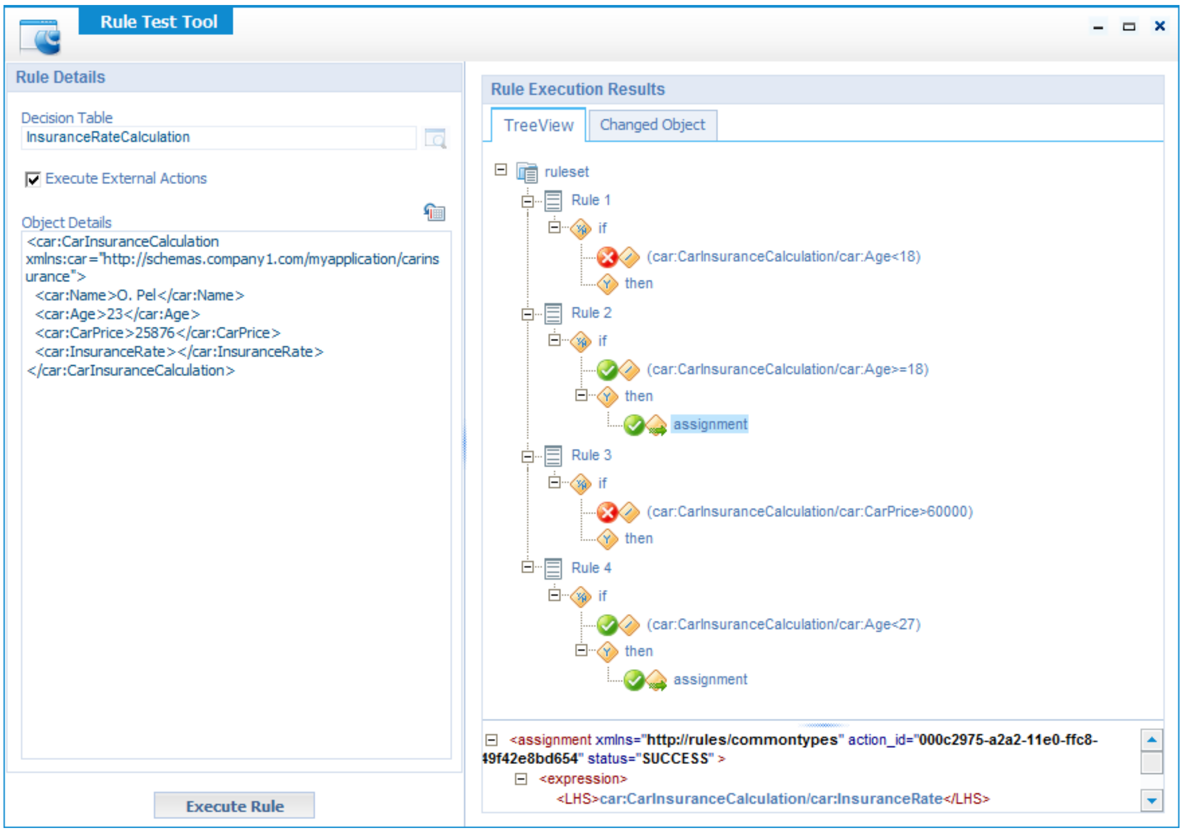
Field	Value
Name	InsuranceRateCalculation
Description	Calculate insurance rate

3.8.3 Testing the Decision Table

1. In the Workspace Documents, right click the *InsuranceRateCalculation* decision table and select *Test Decision Table*.



2. In the object details provide values for **Age** and **CarPrice.**
3. Click **Execute Rule**.



```
<car:CarInsuranceCalculation
xmlns:car="http://schemas.company1.com/myapplication/carins
urance">
  <car:Name>O. Pel</car:Name>
  <car:Age>23</car:Age>
  <car:CarPrice>25876</car:CarPrice>
  <car:InsuranceRate></car:InsuranceRate>
</car:CarInsuranceCalculation>
```

```
ruleset
  Rule 1
    if
      ✗ (car:CarInsuranceCalculation/car:Age<18)
      then
  Rule 2
    if
      ✓ (car:CarInsuranceCalculation/car:Age>=18)
      then
        ✓ assignment
  Rule 3
    if
      ✗ (car:CarInsuranceCalculation/car:CarPrice>60000)
      then
  Rule 4
    if
      ✓ (car:CarInsuranceCalculation/car:Age<27)
      then
        ✓ assignment
```

```
<assignment xmlns="http://rules/commontypes" action_id="000c2975-a2a2-11e0-ffc8-
49f42e8bd654" status="SUCCESS" >
  <expression>
    <LHS>car:CarInsuranceCalculation/car:InsuranceRate</LHS>
```

4. Check whether the correct rules are applied to your values.
5. Check the calculated insurance rate on the tab page *Changed Object*.

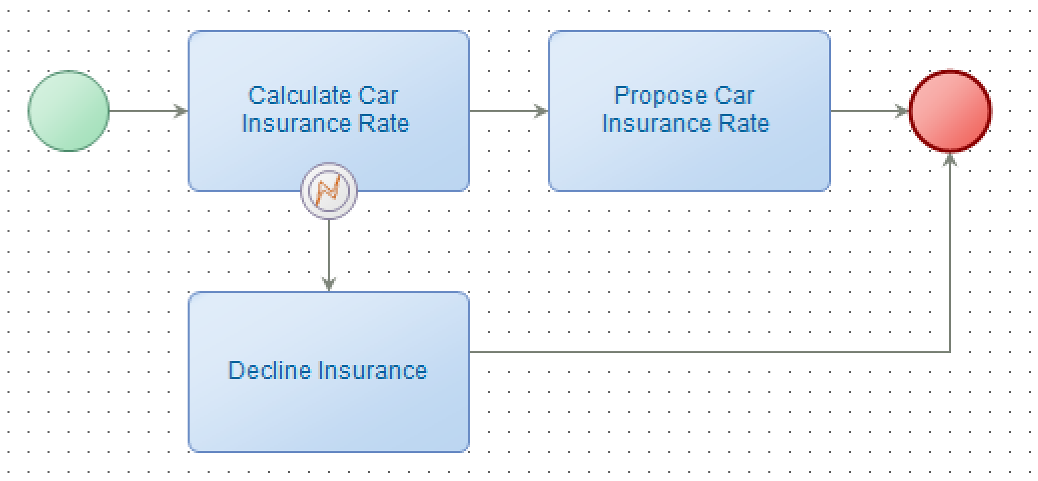
You can use a decision table anywhere by generating a web service for the decision table or use it directly in a business process model, which you will do in the next exercise.

3.8.4 Creating the Business Process Model

1. In the *Workspace Documents* toolbar, click **New**.
2. Select the *Business Process Model* document.
3. Open the process properties.
4. Change the namespace to:
   http://schemas.companyX.com/myapplication/insuranceprocesses
5. Save the process with the following details:

Field	**Value**
Name	CarsInsuranceRequest
Description	Process new car insurance request
Location	My Application project/Business Process Models/com/companyX/myapplication

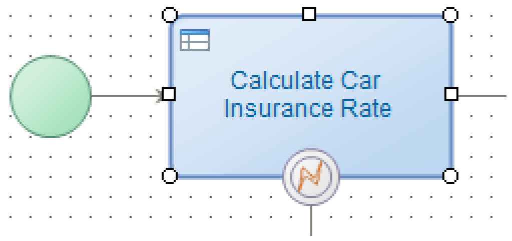
6. Model the process design like this:



The Catch Exception ( ) you place in the border of activity *Calculate Car Insurance Rate*.

7. Right click the activity *Calculate Car Insurance Rate* and select *Insert → Decision Table*.

8. Select the *InsuranceRateCalculation* decision table.



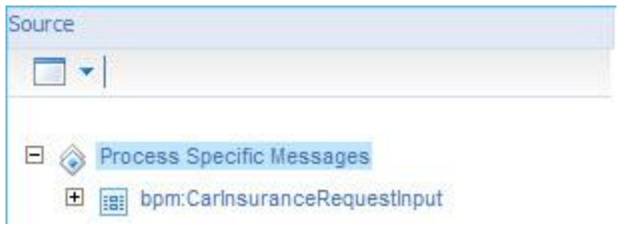
9. Select the *Notify Application Service* (user interface) for the other two activities.

10. Open the properties of the *Decline Insurance* activity and set *Message Type* to **Info**.

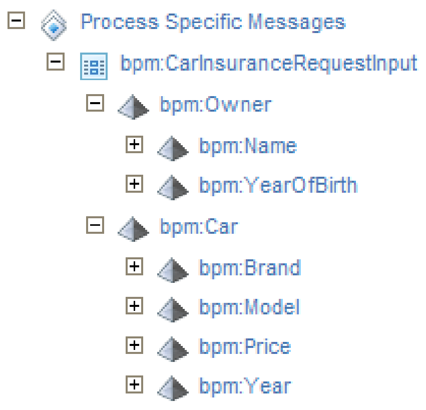
Creating a Start Message

1. Go to the Message Map tab.

2. In the source box, right click Process Specific Message and create a Message **CarInsuranceRequestInput**.



3. Add the following Element structure to the message:

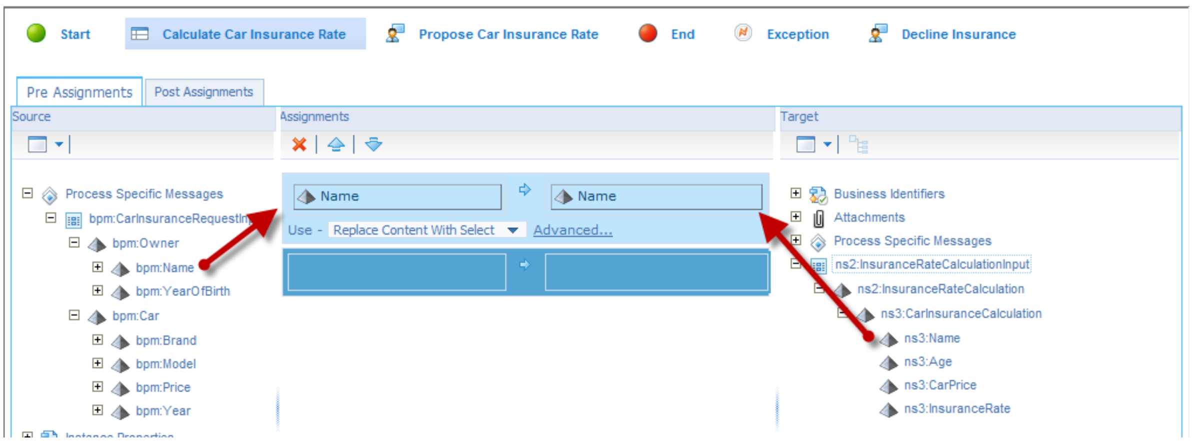

Process Specific Messages
bpm:CarInsuranceRequestInput
bpm:Owner
bpm:Name
bpm:YearOfBirth
bpm:Car
bpm:Brand
bpm:Model
bpm:Price
bpm:Year

Owner
Name
YearOfBirth
Car
Brand
Model
Price
Year

4. Return to the model tab page.
5. Open the properties of the start event.
6. Set *Trigger Type* to *Message* and select the *CarInsuranceRequestInput* message.

Defining the Data flow

1. Go to the *Message Map*.
2. Click the activity *Calculate Car Insurance Rate* at the top.
3. Create an assignment from *Name (start message)* to *Name (Decision table input)*.

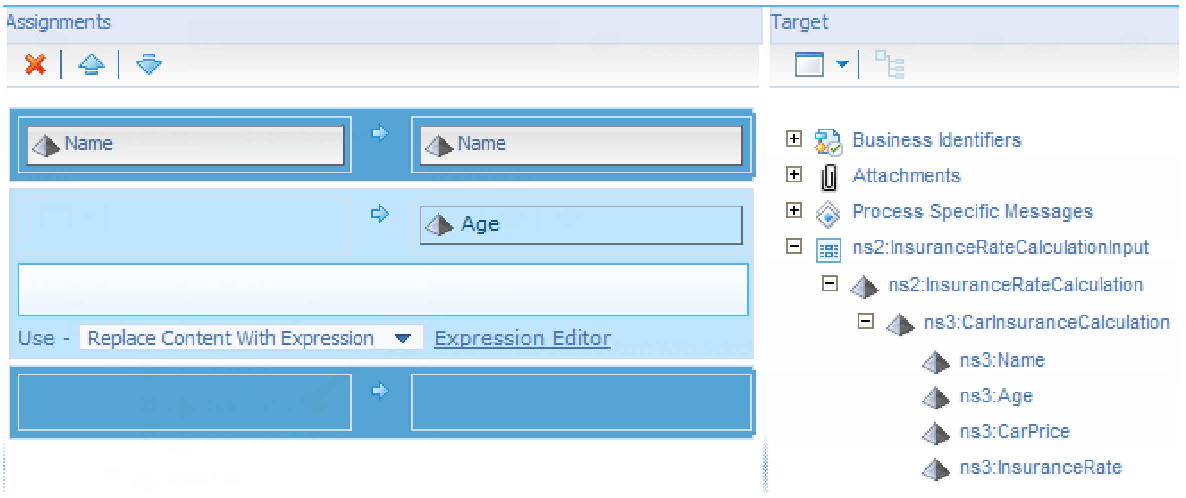


Process Specific Messages
bpm:CarInsuranceRequestIn
bpm:Owner
bpm:Name
bpm:YearOfBirth
bpm:Car
bpm:Brand
bpm:Model
bpm:Price
bpm:Year

Name → Name
Use - Replace Content With Select Advanced...

Business Identifiers
Attachments
Process Specific Messages
ns2:InsuranceRateCalculationInput
ns2:InsuranceRateCalculation
ns3:CarInsuranceCalculation
ns3:Name
ns3:Age
ns3:CarPrice
ns3:InsuranceRate

4. From the target box, drag and drop the element *Age* into the second assignment.
5. In the *Use* select box, select *Replace Content With Expression*.

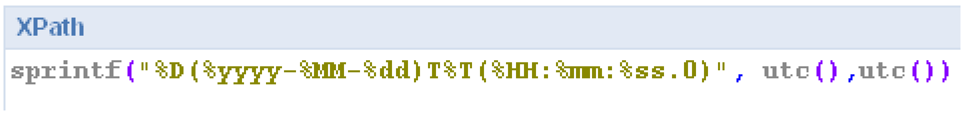


Name → Name
→ Age
Use - Replace Content With Expression Expression Editor

Business Identifiers
Attachments
Process Specific Messages
ns2:InsuranceRateCalculationInput
ns2:InsuranceRateCalculation
ns3:CarInsuranceCalculation
ns3:Name
ns3:Age
ns3:CarPrice
ns3:InsuranceRate

6. Click Expression Editor.

7. In the Functions and Operators box navigate to *Components → Functions → Date-Time*.
8. Drag and drop the *sprintf* function in the XPath text box.



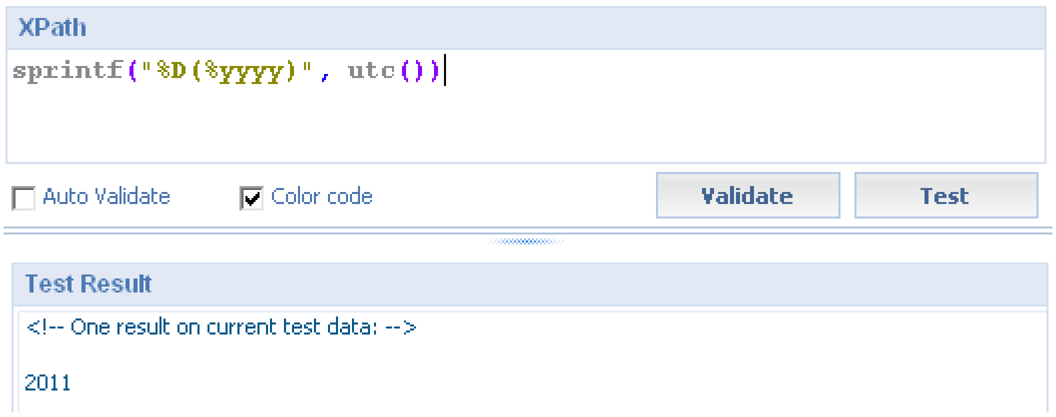
XPath

```
sprintf("%D(%yyyy-%MM-%dd)T%T(%HH:%mm:%ss.0)", utc(),utc())
```

9. Change the sprint function so it looks like this:

sprintf("%D(%yyyy)", utc())

10. When you click **Test** it should return the current year.



XPath

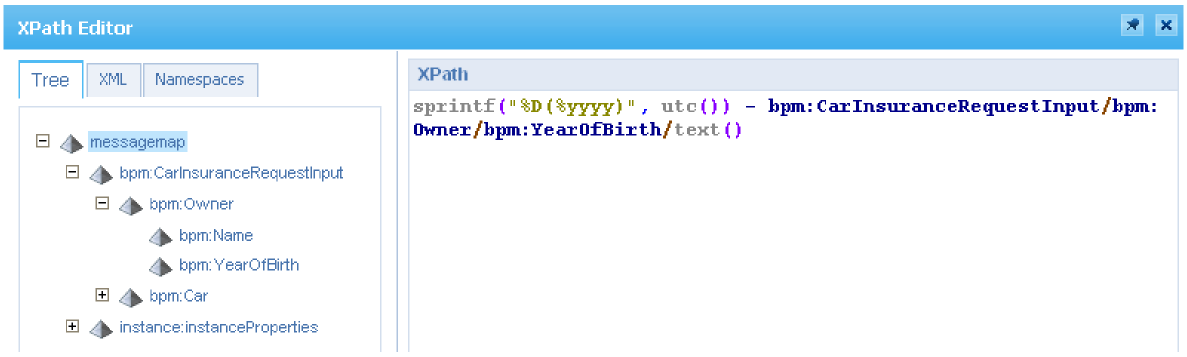
```
sprintf("%D(%yyyy)", utc())
```

Test Result

```
<!-- One result on current test data: -->

2011
```

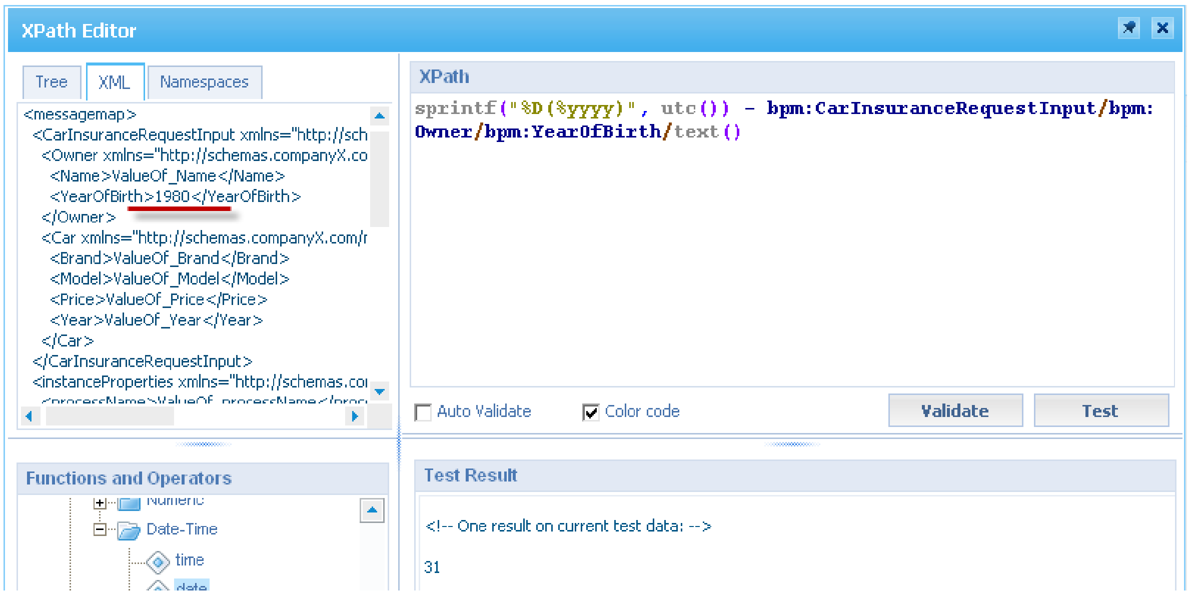
11. Add “ – “ (space minus space) at the end of the expression, without the quotes.
12. From the tree box, drag and drop the *YearOfBirth* (*CarInsuranceRequest → owner*) element behind the minus.



XPath

```
sprintf("%D(%yyyy)", utc()) - bpm:CarInsuranceRequestInput/bpm:
Owner/bpm:YearOfBirth/text()
```

You can test the expression by going to the XML tab page and change the value of *YearOfBirth* from **ValueOf_YearOfBirth** to a 4 digit year and click **Test**.



```
<messagemap>
  <CarInsuranceRequestInput xmlns="http://sch
    <Owner xmlns="http://schemas.companyX.co
      <Name>ValueOf_Name</Name>
      <YearOfBirth>1980</YearOfBirth>
    </Owner>
    <Car xmlns="http://schemas.companyX.com/r
      <Brand>ValueOf_Brand</Brand>
      <Model>ValueOf_Model</Model>
      <Price>ValueOf_Price</Price>
      <Year>ValueOf_Year</Year>
    </Car>
  </CarInsuranceRequestInput>
  <instanceProperties xmlns="http://schemas.co
```

```
sprintf("%D(%yyyy)", utc()) - bpm:CarInsuranceRequestInput/bpm:
Owner/bpm:YearOfBirth/text()
```

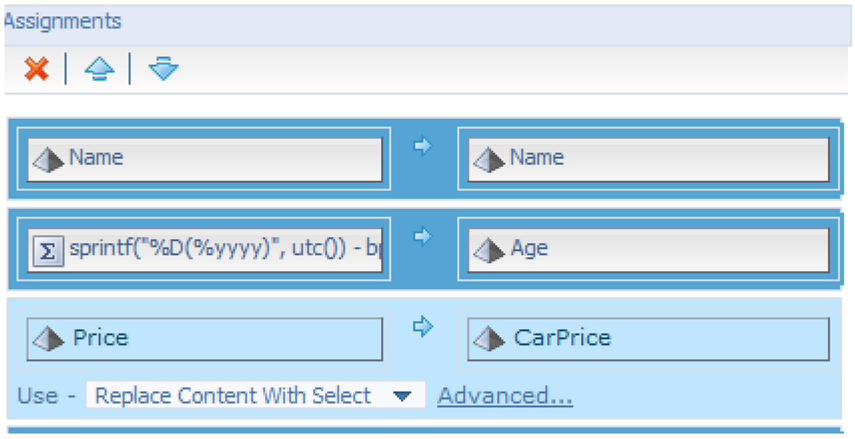
```
<!-- One result on current test data: -->

31
```

13. Test the simplified Age calculation to see if it works.

14. Click **OK**.

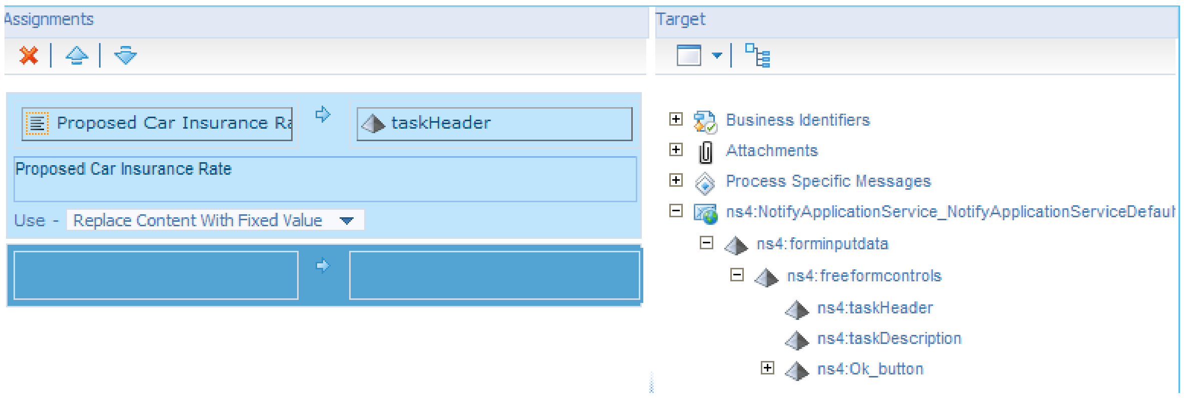
15. Create an assignment from *Price* to *CarPrice*


Assignments
Name
Name
sprintf("%D(%yyyy)", utc()) - b
Age
Price
CarPrice
Use - Replace Content With Select
Advanced...

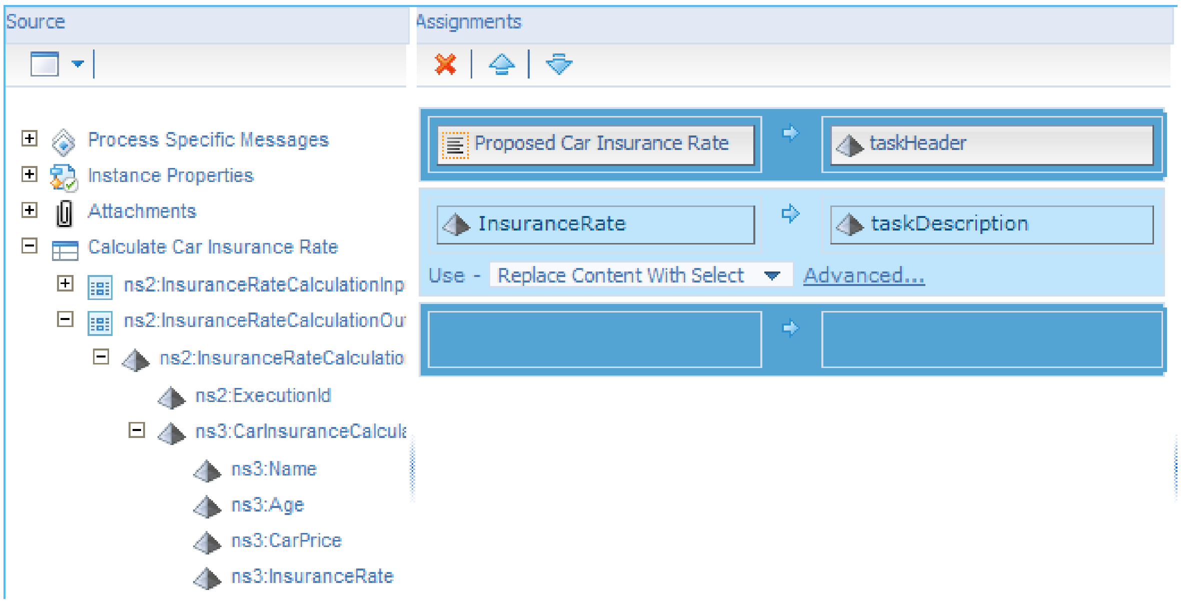
16. Select the *Propose Car Insurance Rate* activity at the top.

17. Expand the input message of the Notify Application.

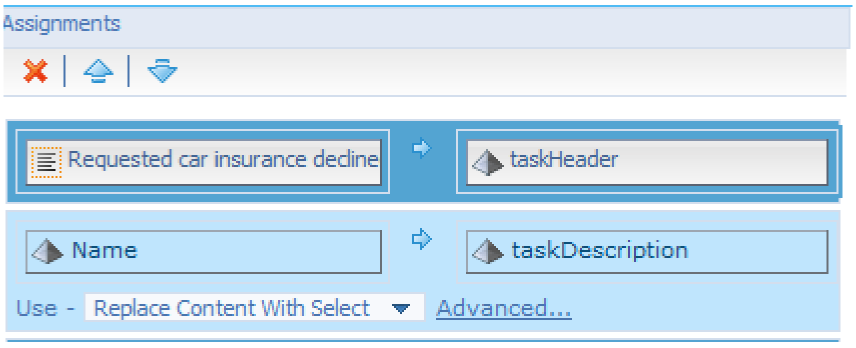
18. Create a fixed value assign for *taskHeader* with the text **Proposed Car Insurance Rate**.


Assignments
Proposed Car Insurance Ra
taskHeader
Proposed Car Insurance Rate
Use - Replace Content With Fixed Value
Target
Business Identifiers
Attachments
Process Specific Messages
ns4:NotifyApplicationService_NotifyApplicationServiceDefaul
ns4:forminputdata
ns4:freeformcontrols
ns4:taskHeader
ns4:taskDescription
ns4:Ok_button

19. Create an assignment from *InsuranceRate* (output message of the decision table) to *taskDescription*

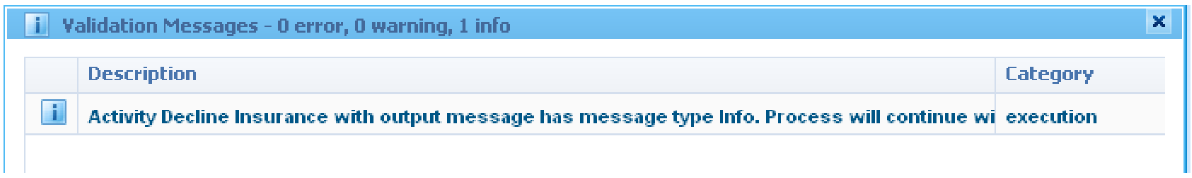

Source
Assignments
Process Specific Messages
Instance Properties
Attachments
Calculate Car Insurance Rate
ns2:InsuranceRateCalculationInp
ns2:InsuranceRateCalculationOu
ns2:InsuranceRateCalculatio
ns2:ExecutionId
ns3:CarInsuranceCalcul
ns3:Name
ns3:Age
ns3:CarPrice
ns3:InsuranceRate
Proposed Car Insurance Rate
taskHeader
InsuranceRate
taskDescription
Use - Replace Content With Select
Advanced...

20. Select the *Decline Insurance* activity at the top.
21. Expand the input message of the Notify Application.
22. Create a fixed value assignment for *taskHeader* with the text **Requested car insurance declined for**.
23. Create an assignment from *Name* (process specific message: *CarInsuranceRequestInput → Owner → Name*) to *taskDescription*


Assignments
Requested car insurance decline
taskHeader
Name
taskDescription
Use - Replace Content With Select
Advanced...

Deploy and Run the process

1. Validate your process


Validation Messages - 0 error, 0 warning, 1 info

Description	Category
Activity Decline Insurance with output message has message type Info. Process will continue wi	execution

2. Publish the process to the organization.
3. Run the process a couple of time, with different values and check if the corresponding message arrives in your inbox.

   Note that proposed rates appear in the personal tasks and decline message in the notification work list.

What is the benefit of using the decision table in a business process?

____

____

3.9 Make Changes Available to SCM

In this exercise you will make your developed content/changes available to your team members by sending the changes to the SCM application.

You should only make changes available after you have tested these to work correctly. This is to ensure that your team members' work is not affected when they incorporate your changes.

NOTE
This only applies when your workspace was created using an SCM application.

1. If closed open the Workspace Documents.
2. Click **Make Changes Available to Others** () in the toolbar.
3. Review the modified content.
4. Provide as comment **Business Logic**.
5. Click **Make Available**.

4. Learning Report

Achievements

- ❑ I know the concept of business logic.
- ❑ I know the advantage of using decision tables and rules.
- ❑ I can create decision cases and rules.
- ❑ I can implement business logic of backend operations.
- ❑ I can implement business logic in business processes.

Notes