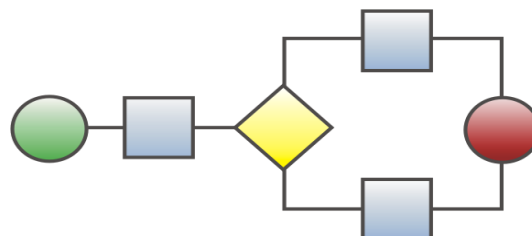


Developing Processes

Exercises



Contents

1. Module	3
1.1 Objectives.....	3
1.2 Overview.....	3
2. About Developing Processes.....	4
2.1 Introduction.....	4
2.2 References	4
3. Exercises	5
3.1 Prerequisites.....	5
3.2 Developing and Deploying the Activities of the BPM	5
3.2.1 Creating Project Design-time Folder Structure	6
3.2.2 Creating Application Deployment Folder Structure	6
3.2.3 Adding Runtime References.....	6
3.3 Developing and Deploying Business Process Models	10
3.3.1 Implementing BPM Activities	10
3.3.2 Defining Data Flow	13
3.3.3 Task Assignment	16
3.3.4 Validating your Business Process Model.....	18
3.3.5 Publishing your Business Process Model.....	18
3.4 Debugging a Process	18
3.5 Adding Process Specific Messages.....	20
3.5.1 Creating a Process Specific Message	20

3.5.2 Adding an Input Message to a Process	21
3.5.3 Applying the Process Input Message	23
3.6 Running and Monitoring Processes	24
3.6.1 Monitoring the Process	24
3.6.2 Task Type Execution	26
3.6.3 Execution of Type Info Messages.....	28
3.6.4 Adding and Assigning the Process Output Message	29
3.6.5 Process Completion	33
3.7 Making a Process Short Lived	33
3.8 Making a Process Web Service Enabled.....	35
3.9 Make Changes Available to SCM	35
4. Sub Processes, Decisions and Loops	36
4.1 Qualify & Analyze	36
4.2 Design & Model	36
4.2.1 Designing the Main Business Process Model.....	36
4.2.2 Adding the Basic Constructs.....	37
4.2.3 Adding a For Each Construct.....	38
4.3 Develop & Deploy	40
4.3.1 Add Runtime References	40
4.3.2 Implementing the constructs.....	40
4.3.3 Using the XPath Editor	43
4.3.4 Implementing the For Each Construct	46
4.3.5 Implementing the Decision Construct	47
4.4 Running and Monitoring the Process.....	49
4.4.1 Running the Process	49
4.4.2 Monitoring Process Execution	49
4.5 Make Changes Available to SCM	50
5. Appendix.....	51
5.1 Adding Constructs to the Process	51
5.2 Adding Connector Lines	51
5.3 Selecting Constructs.....	51
5.4 Validating a Business Process Model.....	52
5.5 Publishing a Business Process Model.....	53
5.6 Running a Process	54
5.7 Monitoring Process Instances.....	54
6. Learning Report	55

1. Module

1.1 Objectives

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

- Understand the concept of the Cordys Closed Loop BPM cycle
- Create executable business process models
- Understand the concept of the Message Map
- Understand and make use of sub processes
- Monitor processes

1.2 Overview

A business process model (BPM) is a graphical representation of a business process. In addition with Cordys, these models also include web services that interact with existing applications, databases, legacy systems and user interfaces to send information or tasks to other users. In this way, you will employ collaboration between people, systems and organizations.

2. About Developing Processes

2.1 Introduction

In this module, you will learn how to develop business processes with Cordys, and make them executable.

In order to make a process executable the included activities must be made executable. In this module you will take the business process model you have made in the *Business Process Management* module as a starting point.

You will (re)use ready-for-use components to implement the given activities to make the model executable. Finally, you will deploy the process and monitor the behavior of the running process instances.

In addition to developing a process, you will also setup and learn more about the Collaborative Workspace with respect to process models.

This module is structured based on the steps of the *Cordys Closed Loop BPM Cycle* described in the module *Business Process Management*.

2.2 References

More information about this subject is available

- Online Cordys Documentation
Working with Business Models → Modeling Business Processes
- <http://community.cordys.com> (here you can find more details on the Cordys Closed Loop BPM cycle as well)

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
- Business Process Management

You must have ONLY the following roles assigned to yourself

- Administrator
- Developer
- Cordys Fundamentals Trainee

Starting the required service container

1. Make sure the *Cordys Services* (service container) in your organization is started.

3.2 Developing and Deploying the Activities of the BPM

In order to make a process executable, you need to make the activities in the process executable. In Cordys activities can be executed both by humans as well as systems. You can divide these activities in two groups:

- 1) Web Services
- 2) User Interfaces

A **Web Service** is an automated action that is performed in a backend system or executed within an available application. Examples are retrieving data from a data store, inserting a new customer into a CRM system, downloading a file, sending a message, start ... etc.

A **User Interface** is a form that allows a user to view information (info) related to the process and/or interact with the process, by performing certain actions (tasks) before the process can continue with the next step. This is also referred to as human interaction. In the module *Workflow* you will learn more about human interaction.

When you need to implement an activity there are two ways to get the implementation for it:

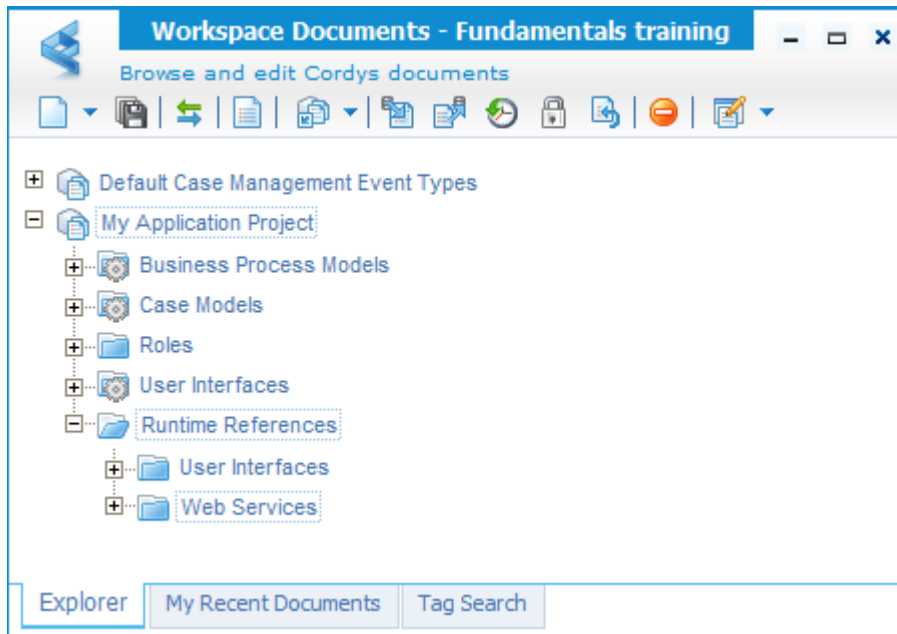
- 1) Build it in this project.
- 2) (Re)Use an existing implementation.

In this module you will use existing implementations (Runtime References). You will learn how to build Web Services in the module *Developing Web Services* and User Interfaces in the module *Developing User Interfaces*.

3.2.1 Creating Project Design-time Folder Structure

In this paragraph you will setup the design-time folder structure for *Runtime References* according to the standard guidelines (See module *Application Management*).

1. If closed, open the *Workspace Documents* and select your *Fundamentals training* workspace.
2. Right click *My Application Project* and Select **New → Folder**.
3. Enter the following name **Runtime References**.
4. Add two subfolders to *Runtime References*:
 - 1) **User Interfaces**
 - 2) **Web Services**



3.2.2 Creating Application Deployment Folder Structure

In this paragraph you will create the deployment structure.

Why are no actions like Start Point of Qualified name required for the deployment structure for the Runtime References?

3.2.3 Adding Runtime References

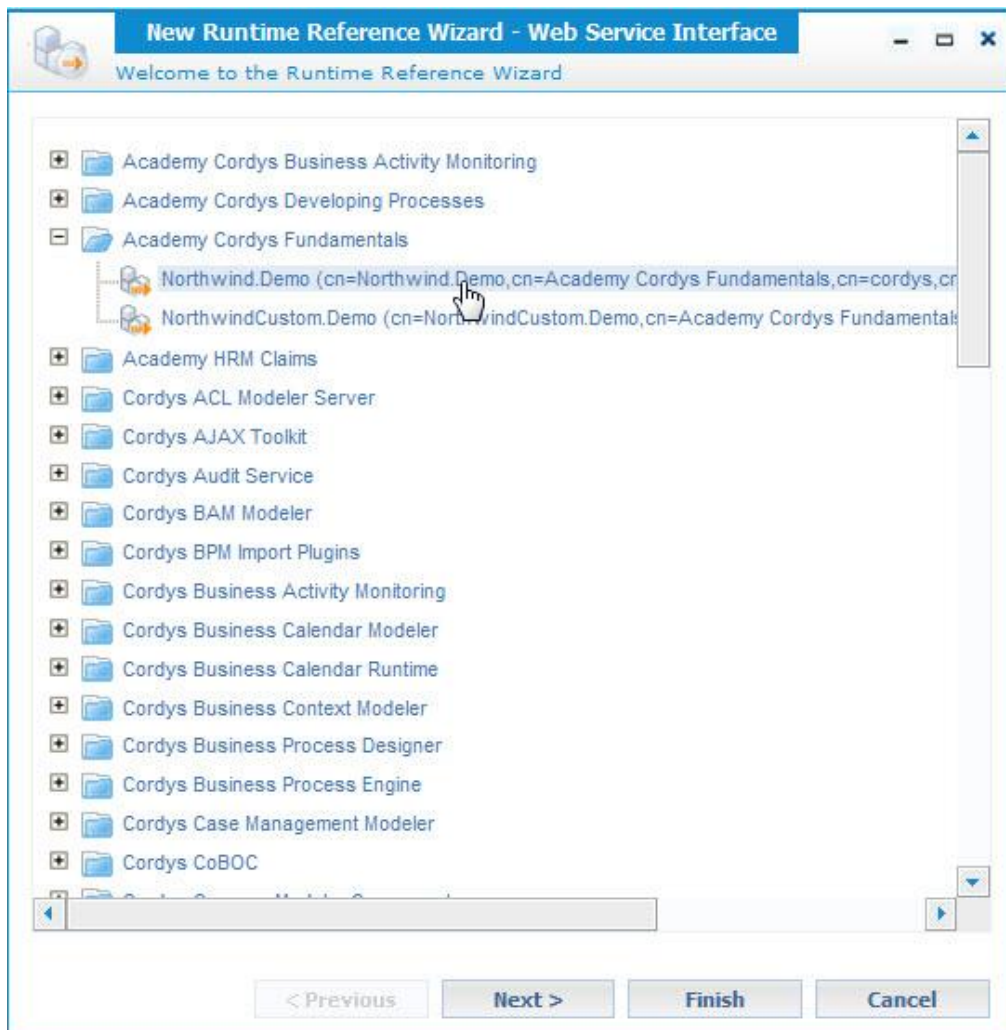
In this exercise you will add an existing web service and user interface as runtime references to your project.

Adding Runtime References to Web Services

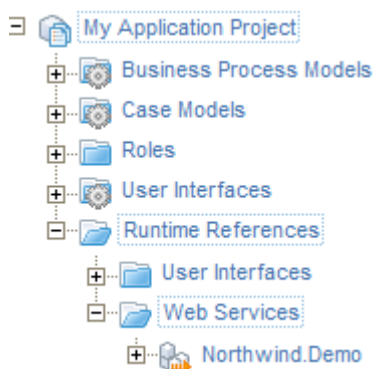
In this paragraph a reference to the web service *GetOrdersObject*, which provides information about a specific order, is made.

1. Navigate to **Runtime References → Web Services**.

2. Right click the *Web Services* folder and select *Add Runtime Reference → Other*.
3. Select **Web Service Interface**.
4. Expand the folder *Academy Cordys Fundamentals*



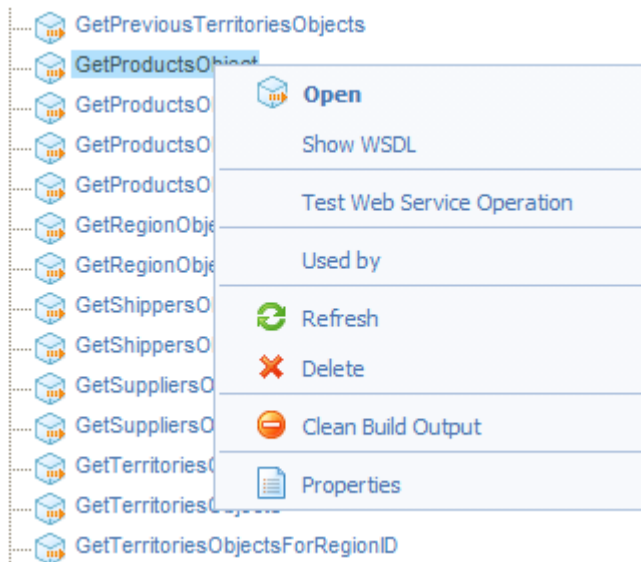
5. Select *Northwind.Demo*.
6. Click **Finish** (**Next** would show the reference details).



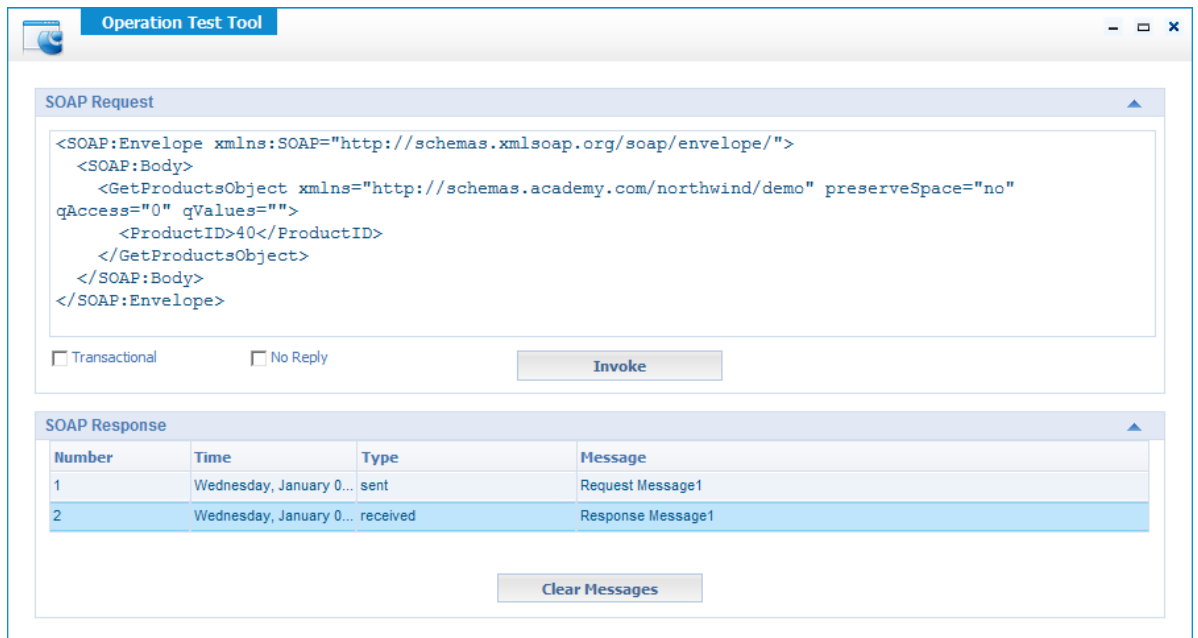
Testing the Web Service

1. Navigate to *Runtime References → Web Services → Northwind.Demo → GetProductsObject*.

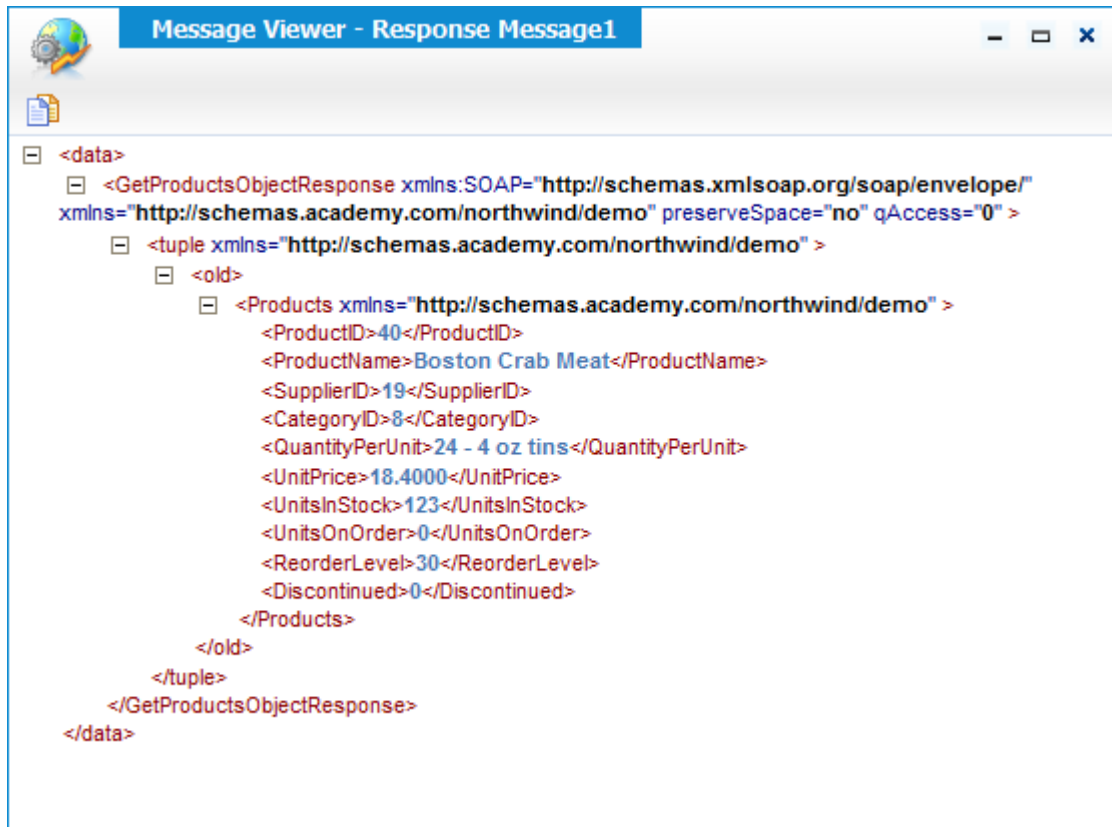
2. Right click *GetProductsObject* and select *Test Web Service Operation*.



3. Double Click the word *PARAMETER* and type **40**. Any number between 1 and 77 will result in a valid product.
4. Click **Invoke**.



5. In the *Soap Response* box click in the row with *received* in the *Type* column.

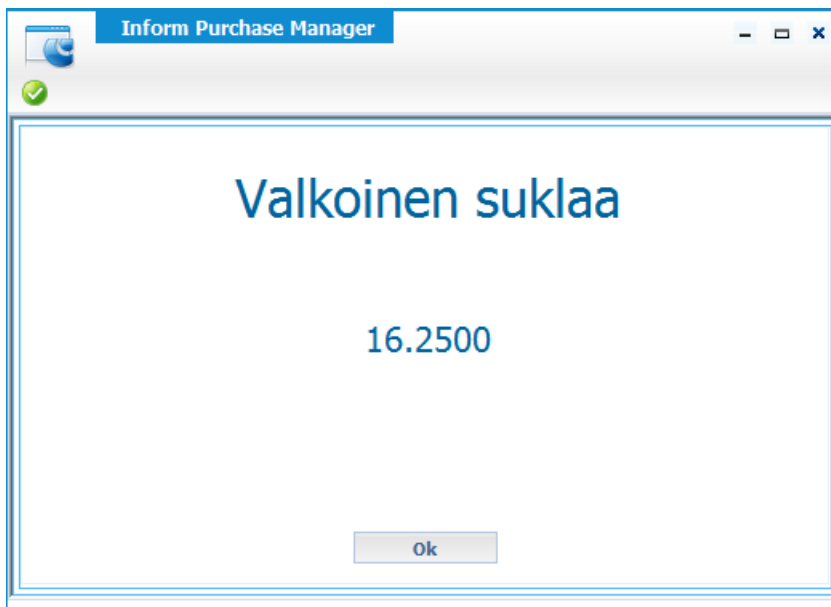


This is the response of the *GetProductsObject* Web Service Operation containing the information about the selected product.

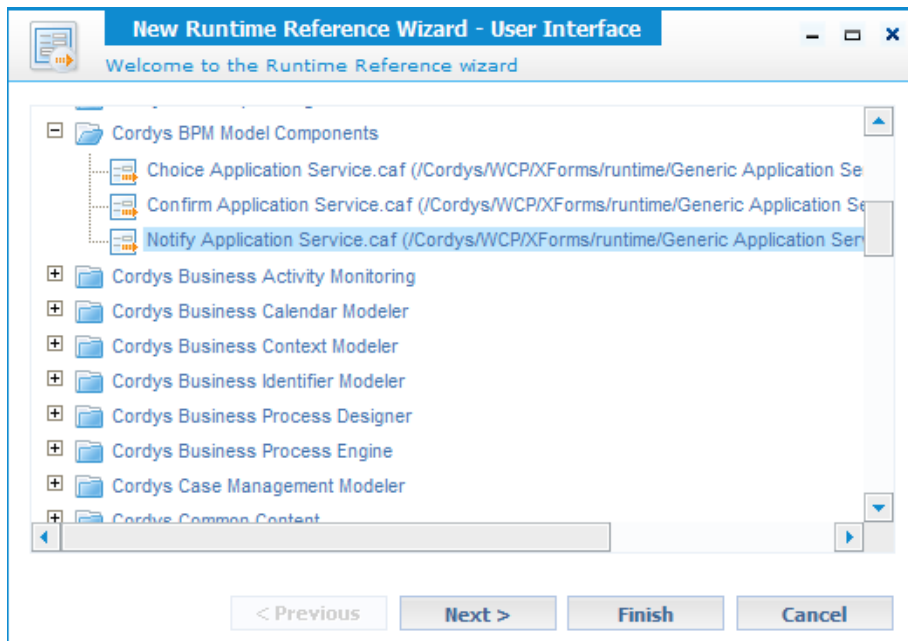
6. Close the *Message Viewer*.
 7. Close the *Operation test Tool*.

Adding Runtime Reference to a User Interface

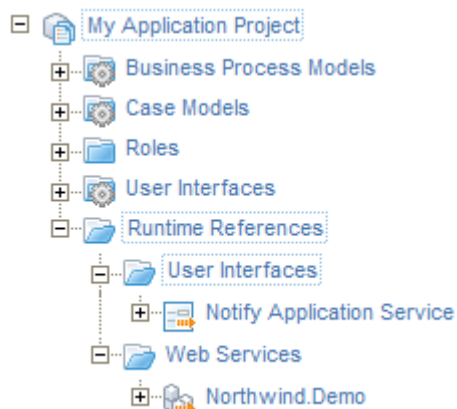
In this paragraph, a reference to an existing user interface is made. This Notify Application Service is a simple user interface which you will use in your process. Later you will build your own user interfaces that can/will meet your process specific requirements. The Notify Application Service can present two lines of information, for example:



1. Navigate to *Runtime References* → *User Interfaces*.
2. Right click the *User Interfaces* folder and select *Add Runtime Reference* → *Other*.
3. Select **User Interface**.
4. Expand the folder *Cordys BPM Model Components*:



5. Select *Notify Application Service.caf*
6. Click **Finish**.



3.3 Developing and Deploying Business Process Models

In this exercise, you will transform your earlier made business process model into an executable process. You will do this by adding the relevant service and user interface to the activities and providing the data flow between the activities.

Once the implementation of the BPM is finished you will validate and publish it to runtime in order to be able to run and test it.

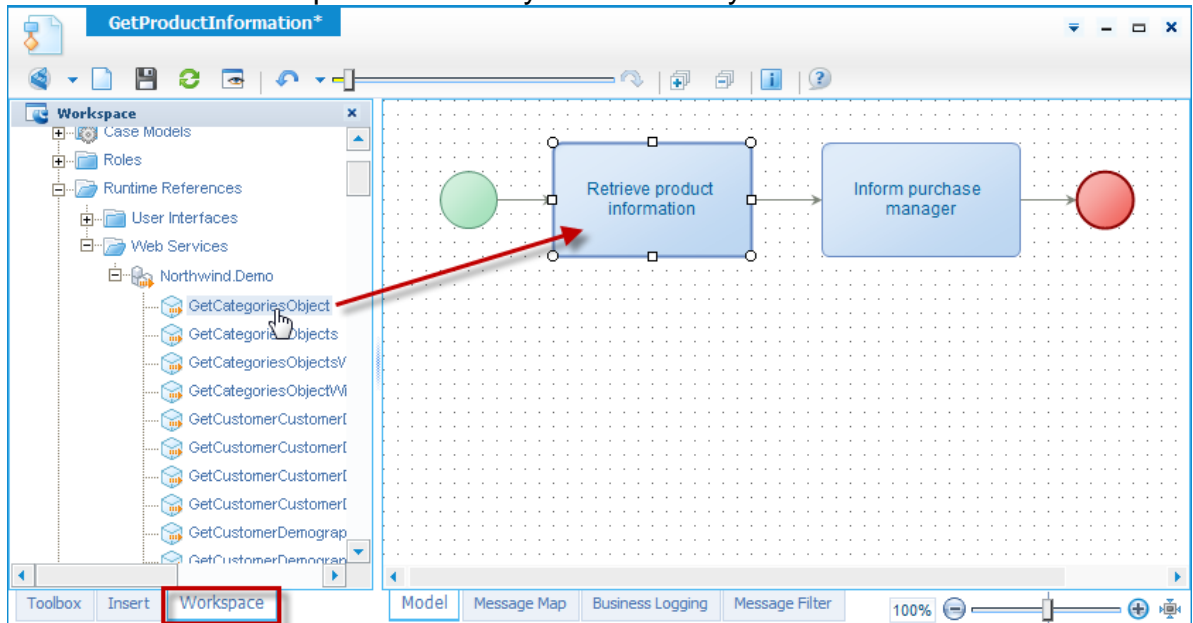
3.3.1 Implementing BPM Activities

Once you have the relevant web services and/or user interfaces available, you need to add these to your business process model.

You can add these executable components to your model using one of the following methods:

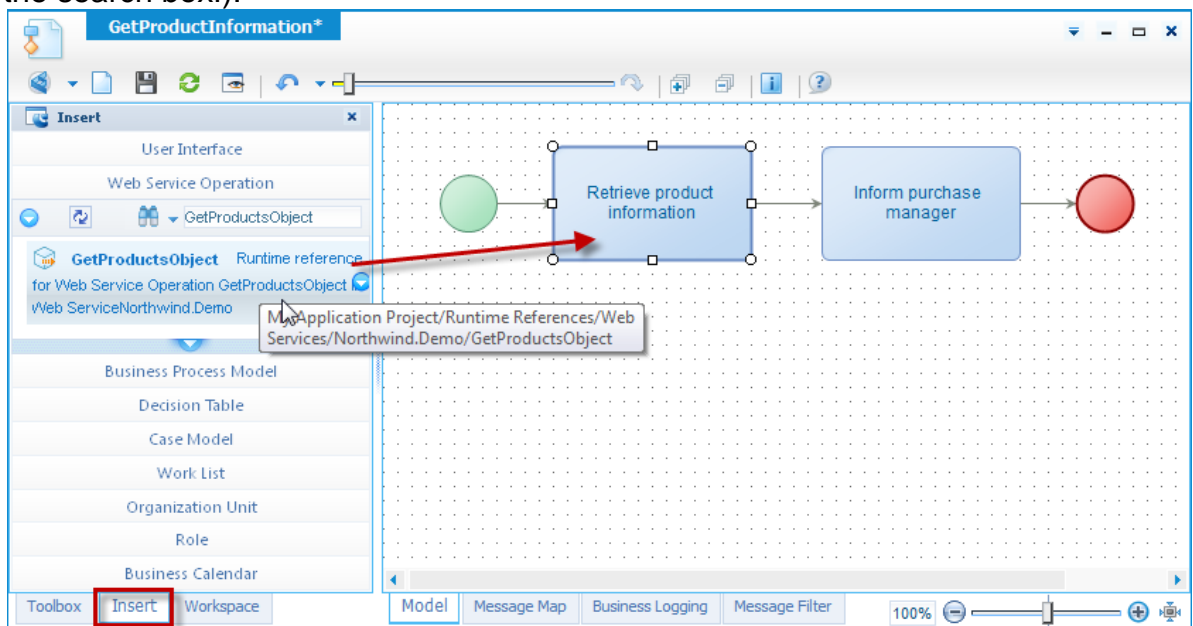
- **Via Workspace tab:**

Select the *Workspace* tab at the bottom left of your BPM and drag and drop the desired web service operation directly on the activity in the model.



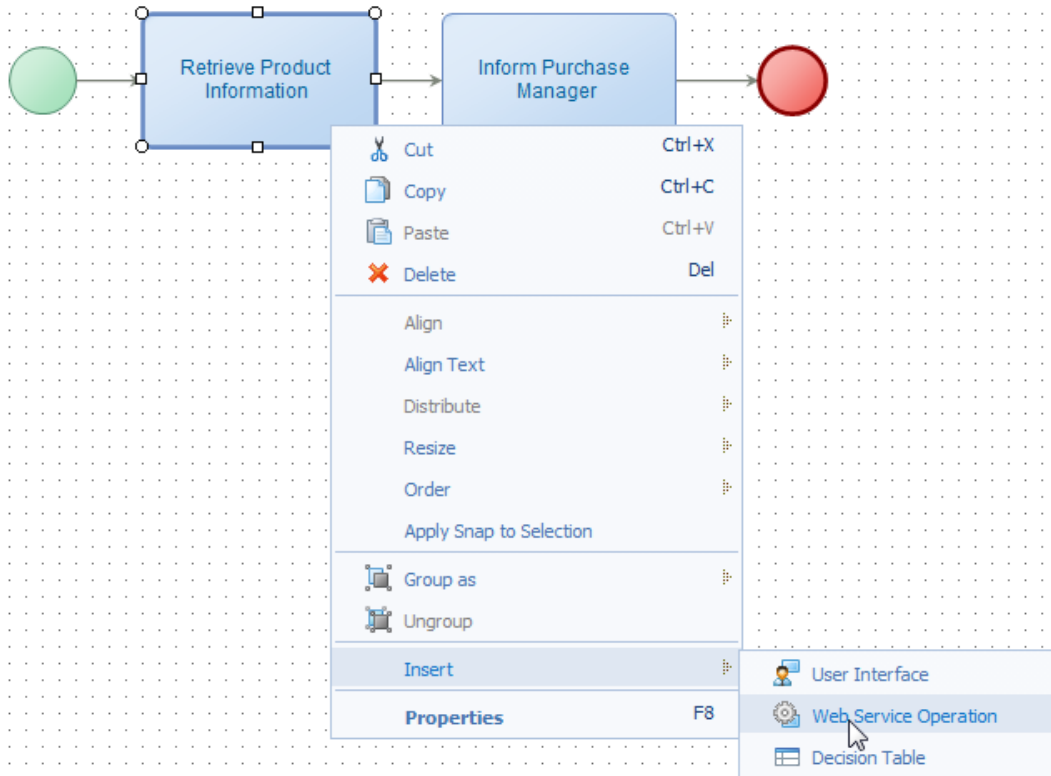
- **Via Insert tab:**

Select the *Insert* tab at the bottom left. Select *Web Service Operation* and drag and drop the desired web service operation directly on the activity in the model (tip: use the search box!).

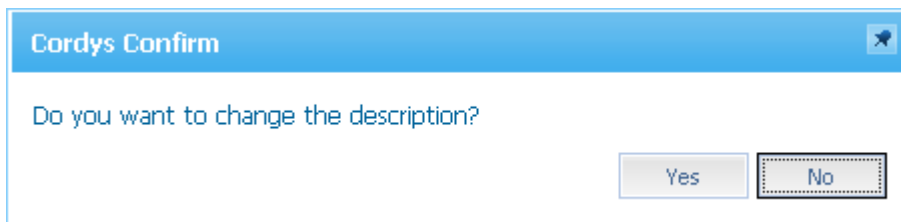


- **Via Insert on Activity**

Right click the activity and select *Insert → Web Service Operation*. Use the search box in the screen that appears to select the desired web service operation.



1. If closed, open your BPM *GetProductInformation*.
2. Use any of the above mentioned methods to drag and drop the *GetProductsObject* on the BPM activity *Retrieve Product Information*.



3. Click **No** as you do not want to rename the description of the activity from *Retrieve Product Information* into *GetProductsObject*.
4. Apply the *Notify Application Service* in the BPM activity: *Inform Purchase Manager*, without changing the description of the activity.

Both activities now have an icon, in the left upper corner, to identify the implementation type of the activity.



5. **Save** your process.

3.3.2 Defining Data Flow

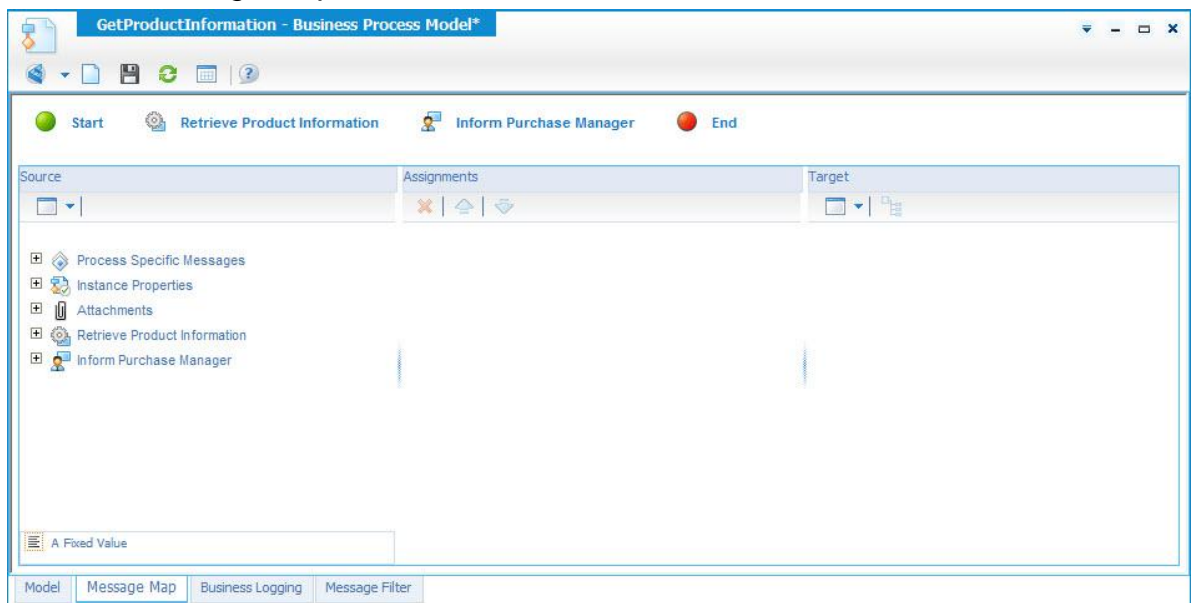
Typically, web services use input and output messages for putting data into the service and returning data from the service. An input message specifies your request (e.g. 40 as ProductID) and the output message contains the response of the web service (e.g. the product details related to the given ProductID). The same applies to user interfaces. User interfaces display data on a form and return the form data back to the process.

In the design-time version of the BPM you can access the input and output messages of the services via the *Message Map*. The message map shows the structure of these messages and enables you to pass data to the different activities in your process. Quite often, you will access the data before or after the given activity is executed. Therefore you can enter the message map at the Pre Assignments or Post Assignments level of an activity). When you run the process, the message map contains the actual data of the services.

Implementing the Retrieve product information activity

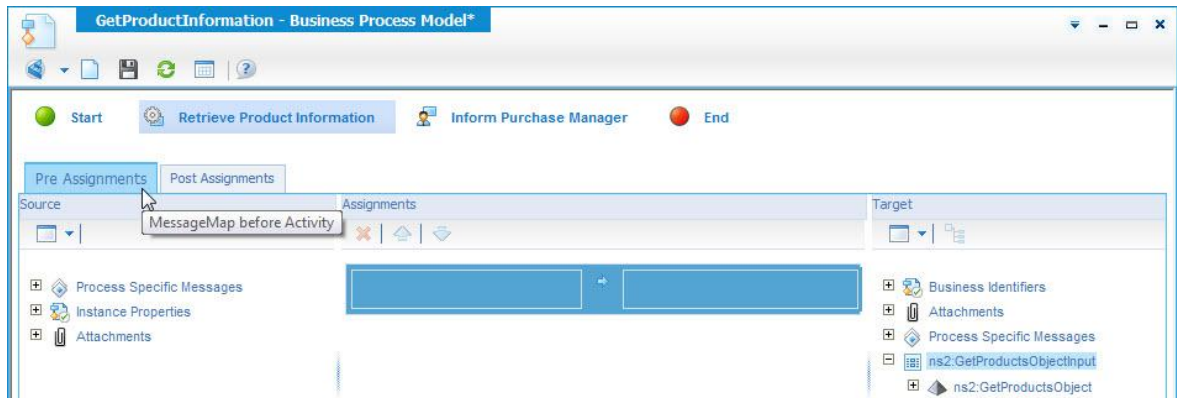
In this paragraph you will implement the *Retrieve product information* activity. You will add an assignment to enter a value into the input message for the *GetProductsObject* web service. When running the process the assignment is executed so the business process engine will pass the value to the web service.

1. If closed, open your BPM *GetProductInformation*.
2. Click the *Message Map* tab at the bottom of the editor screen.



3. Select the *Retrieve Product Information* activity (at the top) to select the point where you want to enter values into the message map.

4. Check whether you are at the Pre Assignments tab page, since you want to enter a value before the activity is being executed. If not, click the *Pre Assignments* tab to make this active:

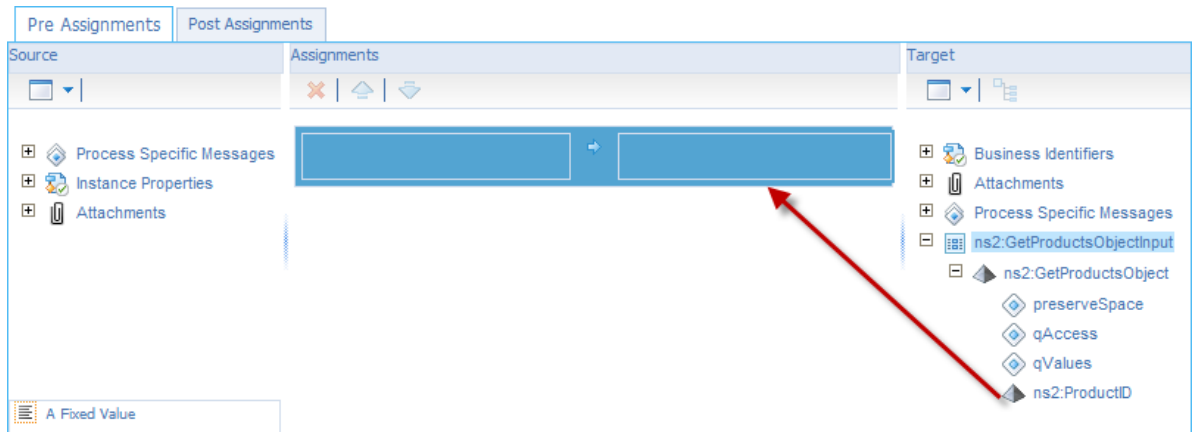


5. In the Target box, at the right, locate the message *ns2.GetProductsObjectInput*.

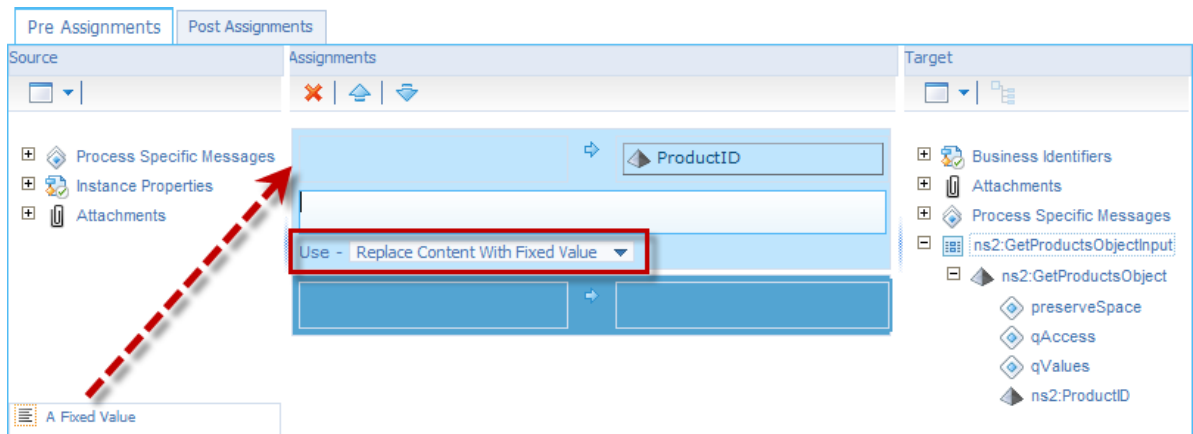
NOTE

In the course materials and related screen shots, we refer to *ns2:GetProductsObjectInput* to make a reference to the associated namespace. However, with your process it might be a different number, e.g. *ns3*, *ns4* or *ns* with any number.

6. Expand the element *ns2:GetProductsObject* below it. Drop the element *ns2:ProductID* into the right textbox in the Assignment area in the middle.



The assignment that is made automatically has *Use – Replace Content with Fixed Value*. It can also be made manually by dragging and dropping the text *A Fixed Value* into the Assignments area



7. In the assignment area, type in **50** as the value for the ProductID.

The screenshot shows the 'Assignments' tab in a BPM editor. At the top, there are three icons: a red 'X', a blue up arrow, and a blue down arrow. Below these is a large light blue box. Inside this box, on the right, is a variable selector showing a diamond icon and the text 'ProductID'. On the left, there is a text input field containing the value '50'. At the bottom of the box, there is a label 'Use -' followed by a dropdown menu currently showing 'Replace Content With Fixed Value'.

The BPM engine will now execute the assignment before the first activity is executed. So effectively it prepares the SOAP request, just as you did when you manually changed the “PARAMETER” value to a valid ProductID while testing this web service operation in exercise Adding Runtime References on page 6.

Implementing the Inform purchase manager

The *Inform purchase manager* activity will now be implemented with the user interface to send the purchase manager the information that is retrieved from the order system.

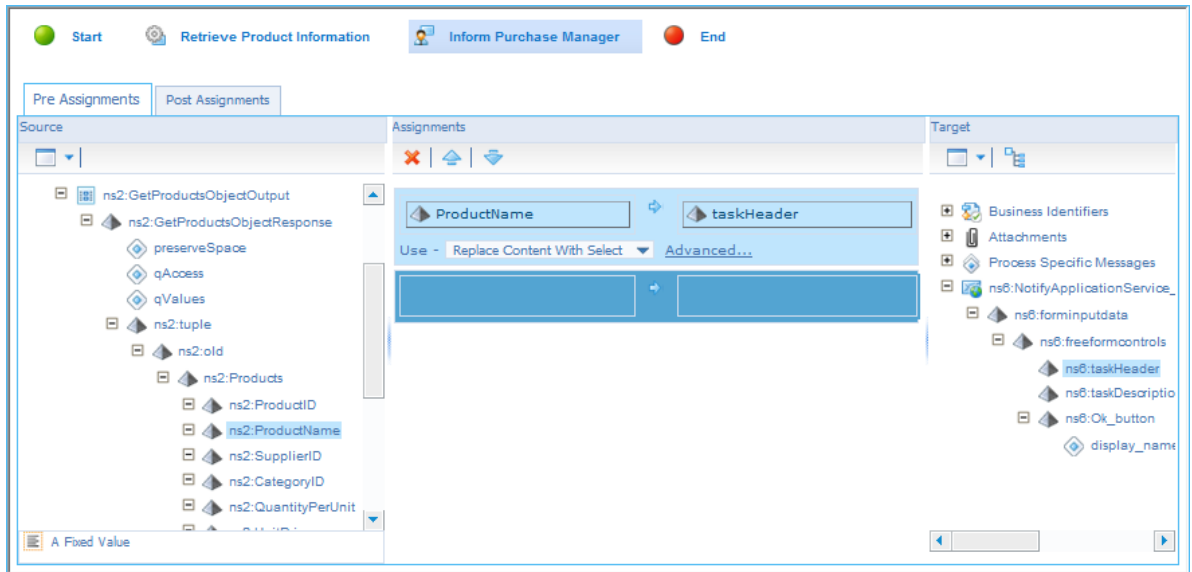
1. In the Message Map tab select the *Inform Purchase Manager* activity.
2. Check whether you are at the *Pre Assignments* tab page.
3. In the *Target* box, right click the element *ns6:forminputdata*, below the message *ns6:NotifyApplicationService* and select *Expand All*.

Which elements can be filled for the input message of the notification form?

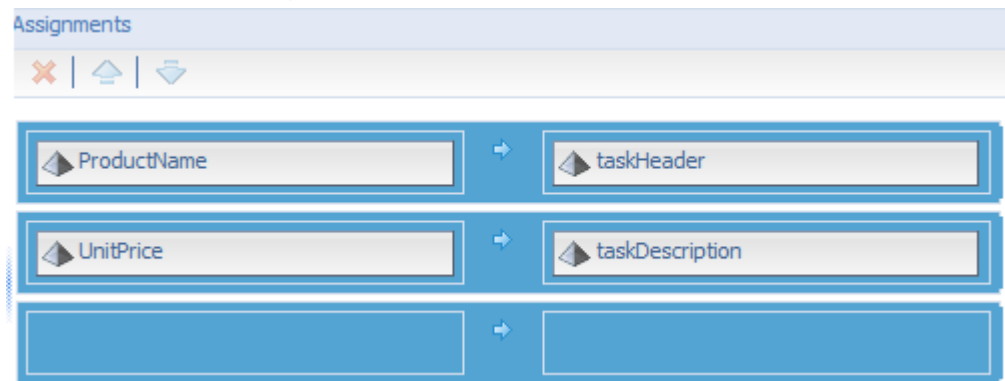
For the input message of the notification you will use the output message of the Retrieve Product Information.

4. In the *Source* box left, locate the activity *Retrieve Product Information* and expand it one level down.
You will notice that this web service operation has three types of messages: input and output and additionally a fault details message.
5. Right click the response message *ns2:GetProductsObjectOutput* and select *Expand All*.
6. Navigate to the Element *ns2:ProductName* (*ns2:GetProductsObjectResponse* → *ns2:tuple* → *ns2:old* → *ns2:Products*).
7. Select the *ns2:ProductName* element

8. Hold down the **CTRL** key and **select** *ns6:taskHeader* in the Target box
You can see that the assignment is created:



9. Create another assignment from *UnitPrice* to *taskDescription*:



10. Save the process.

3.3.3 Task Assignment

A user interface includes human interaction, meaning: it must be sent to a person, whose task it is to look at it, check it or add information to it. Therefore, in your business process model, you need to associate a role (that is assigned to one or more organizational users) to the user interface.

In the module Workflow, you will learn more about workflow so here you will simply create a role and assign yourself this role, and use the role for the activity, without looking at the details.

1. In *My Application Project* navigate to *Roles*.
2. Right click *Roles* and select *New* → *Other*.

3. Select *Role*.

Untitled Role - Role*

Drag the Roles or Tasks from the Workspace Documents or Quick Access Menu to assign them.

Name: Untitled Role

Description: Untitled Role

☒ Is Functional Role

Sub Roles: Drag and drop Roles or use the Add button above

User Interfaces: Drag and drop User Interfaces or use the Add button above

4. Provide the following values:

Field	Value
Name	Purchase Manager
Description	Purchase managers

5. Click **Save**.

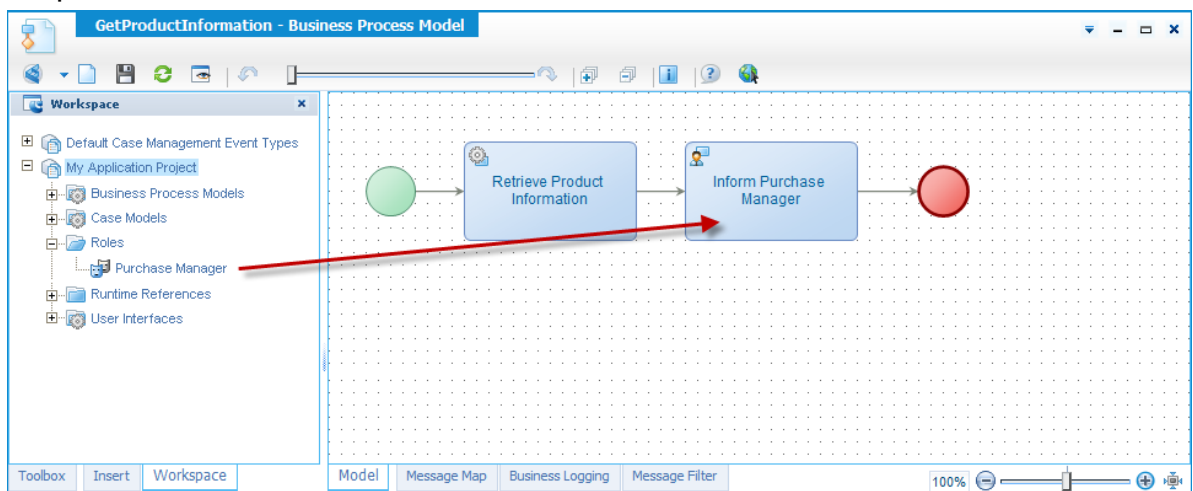
NOTE

You can assign task to a role and sub roles. In this case you will use the role for sending message, so no additional configuration is required. This is similar to using a distribution list for your e-mails.

6. Close the Role editor screen.

7. If closed, open your BPM *GetProductInformation*

8. Drag and drop the created role on top of the *Inform Purchase Manager* activity in the process:



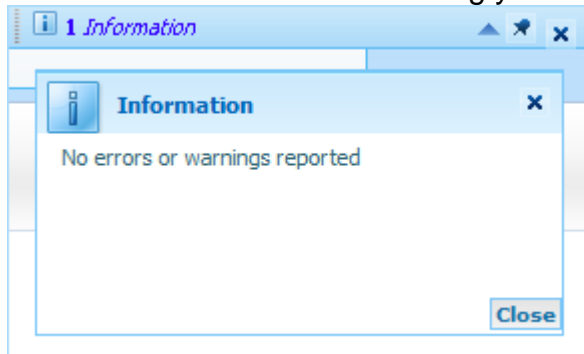
9. Save the process.

3.3.4 Validating your Business Process Model

In this paragraph you will validate the changes you made while implementing the process activities.

1. If closed, open your BPM *GetProductInformation*.
2. Validate your process. (see the Appendix on page 52)

You should have a UFO informing you there are no errors or warnings.



3.3.5 Publishing your Business Process Model

To be able to execute or run a process, you first need to publish your process. At this stage, you are not ready to package and deploy your application, so you will publish the process to your own organization. This enables you to explore the runtime execution of your process.

1. Publish the process to your organization (see the Appendix on page 53).

Why don't you have to publish the role, Web Service and User Interface you use in your process?

3.4 Debugging a Process

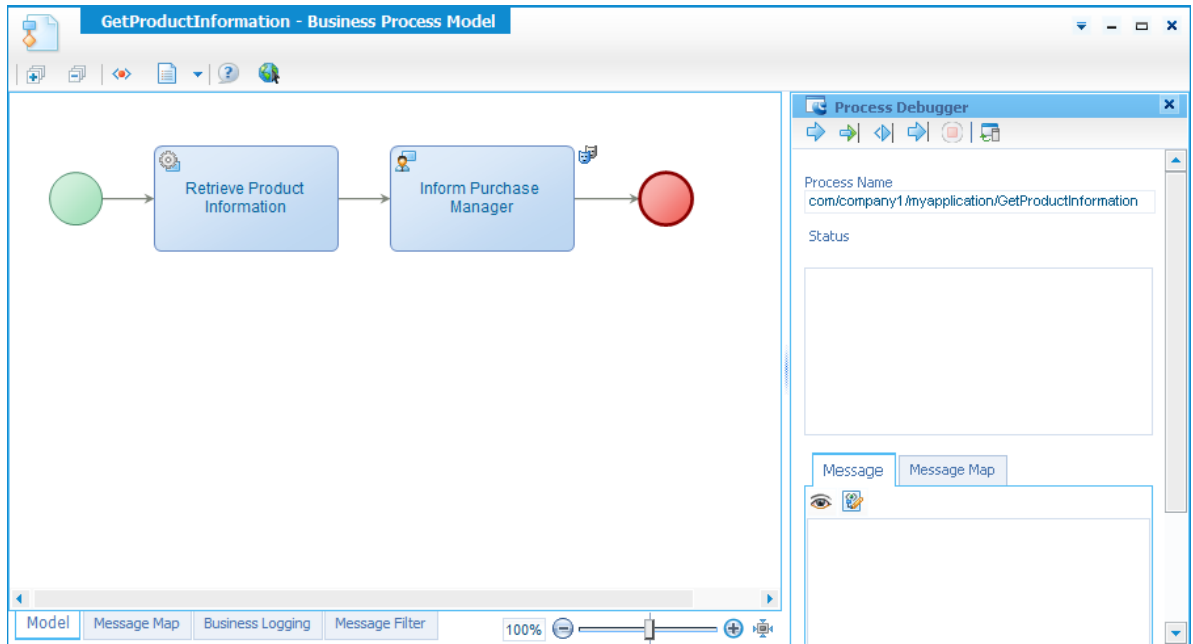
In this exercise, you will use the process debugger to explore the runtime behavior of your process.


While developing a process, you can debug your process to trace the running process not only activity by activity but also assignment by assignment.

You can start a process in debug mode using one of the following methods:

- Right click in the process and select *Execution → Debug* (CTRL + F12).
- Click in the process, to get focus to the editor and press CTRL + F12.
- In the Workspace Documents window, right click the process and select *Execution → Debug*.

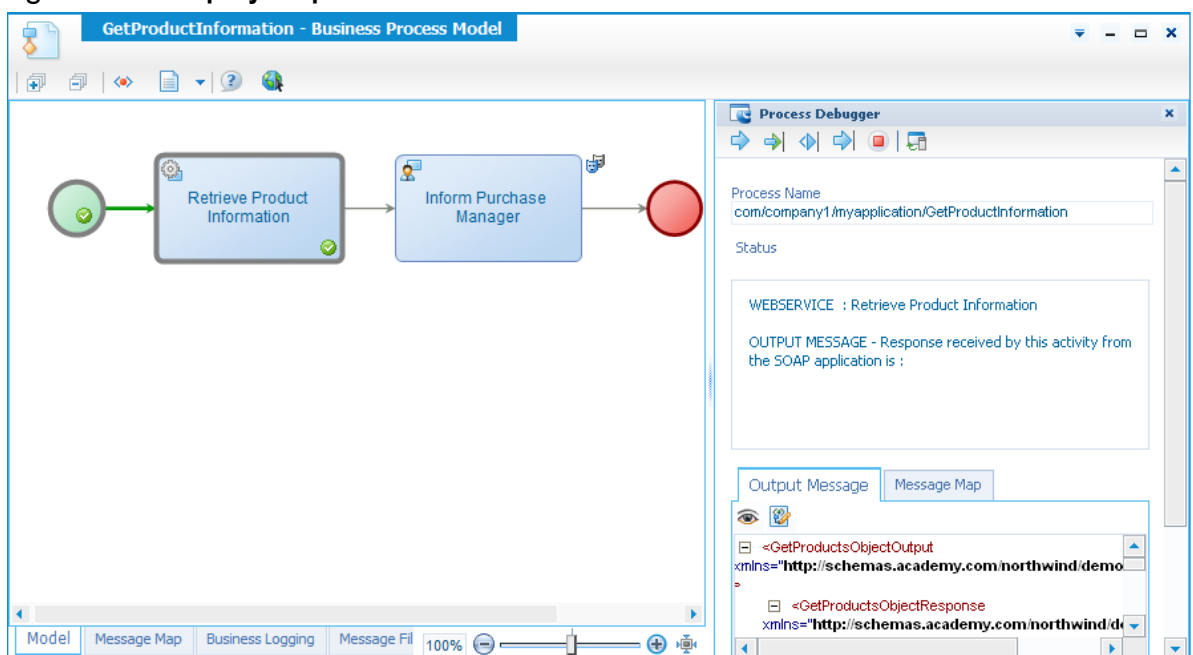
1. Debug the process using one of the before mentioned methods.
This will open the Process Debugger box on your process:



2. Click **Step By Step** () in the *Process Debugger* box, the process is started.
3. Again click **Step By Step**.
In the status field, details of the first assignment are displayed. The first assignment in your process was ProductID = 50.

What is the difference between the tabs Message and Message Map?

4. Again click **Step By Step**.



5. Explore Status and the responses in both the Message and Message Map tabs.

6. Go through the whole process, *Step By Step* and carefully look at the information displayed. The user notification message is displayed as part of debugging your process as well.
7. Close the process debugger.

3.5 Adding Process Specific Messages

In order to persist data you can also create messages or elements in the message map. These messages are called Process Specific Messages.

These specific messages can be used for example to start the process with a certain message, or store original message content before changing it, etc.

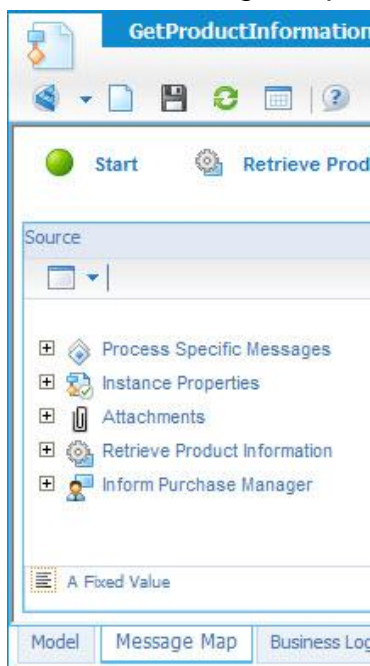
Typically these messages are used as input and output messages of the process.

In this exercise you will learn how to create and apply Process Specific Messages.

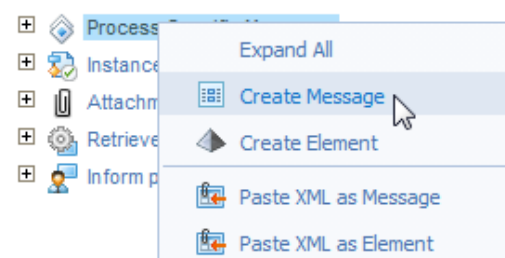
3.5.1 Creating a Process Specific Message

In this part of the exercise, you will create a more informative input message structure to request for a Product Number, instead of using the hard coded product id.

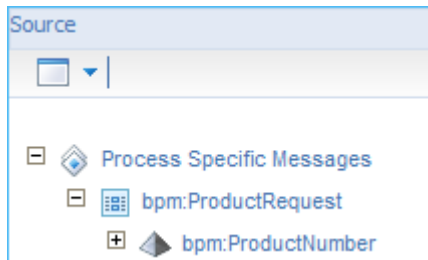
1. If closed, open your BPM *GetProductInformation*.
2. Go to the *Message Map* tab:



3. In the *Source* pane, right click *Process Specific Messages* and select *Create Message*:



4. Enter the name: **ProductRequest**.
5. Right click the created message and select *Create Element*.
6. Enter the name: **ProductNumber**.
7. The process specific message should look like this:



NOTE

There are two types of process specific messages:

Message used to contain a structure of elements and can be used for passing a message into and / or out of a process.

Element can be used only in the process it is defined in. It is often used as a single element but can contain a structure of elements as well.

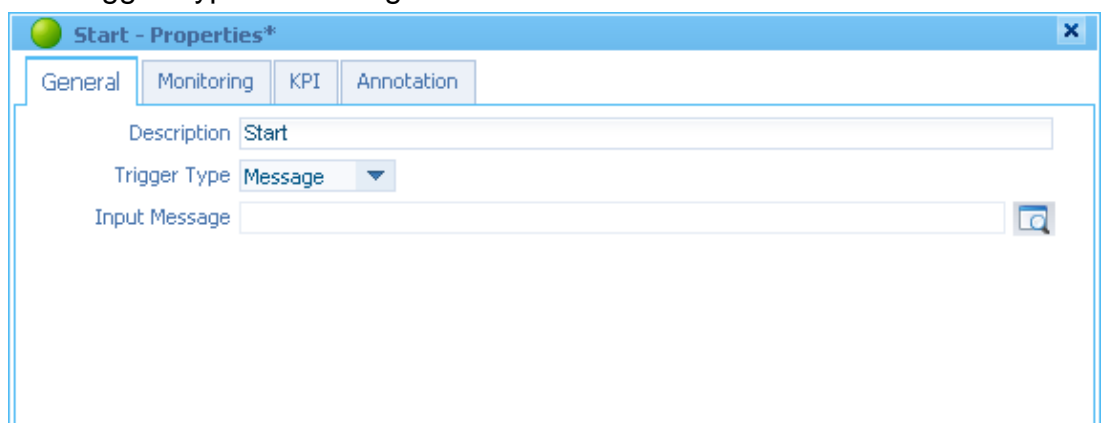
3.5.2 Adding an Input Message to a Process

Next, you will use the created process specific message as the input message for your process.

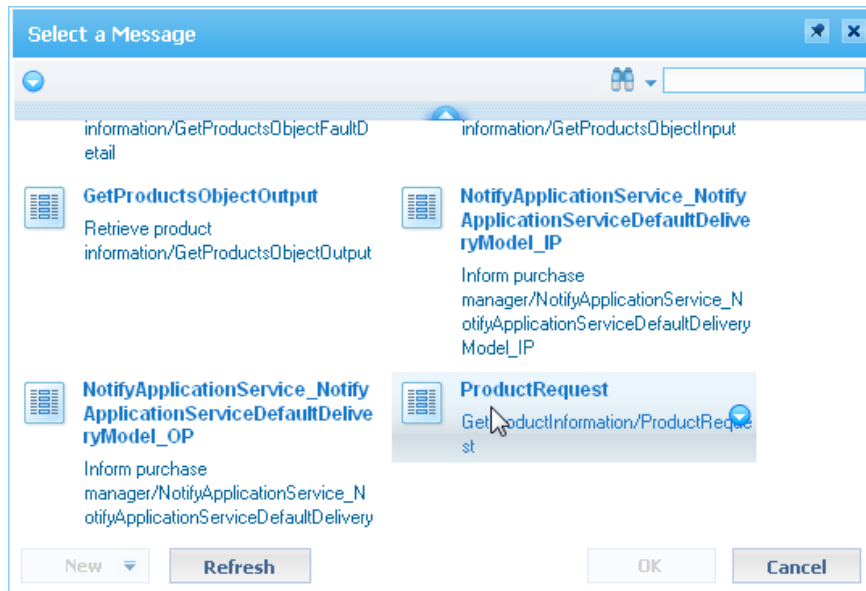
Assign input message to a process

You can assign a message, to a construct like start and end event, using one of the following methods:

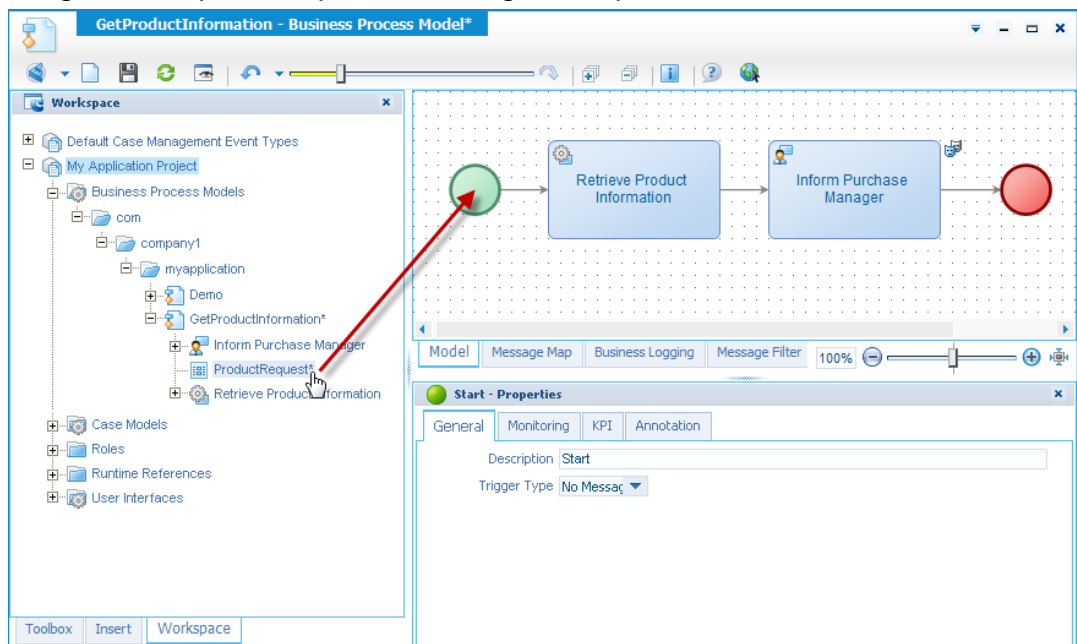
- Via the **Lookup** button
 - Open the properties of the Start construct.
 - Set Trigger Type to Message:



- Click the **Look up** button at the End of Input Message:



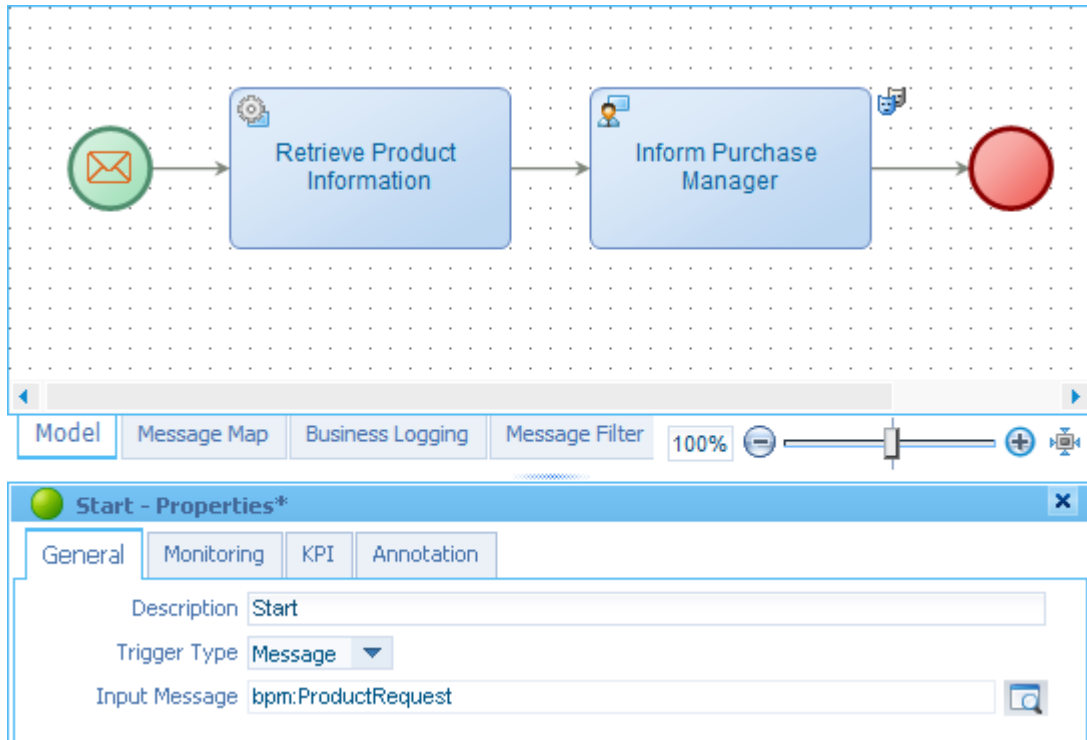
- Select the required message.
- Via Drag and Drop
 - Select the *Workspace* tab in your process.
 - Drag and Drop the required message on top of the start event:



Assign the input message

1. If closed, open your BPM *GetProductInformation*.

- Assign the *ProductRequest* message as input message to your process using one of the before mentioned methods:

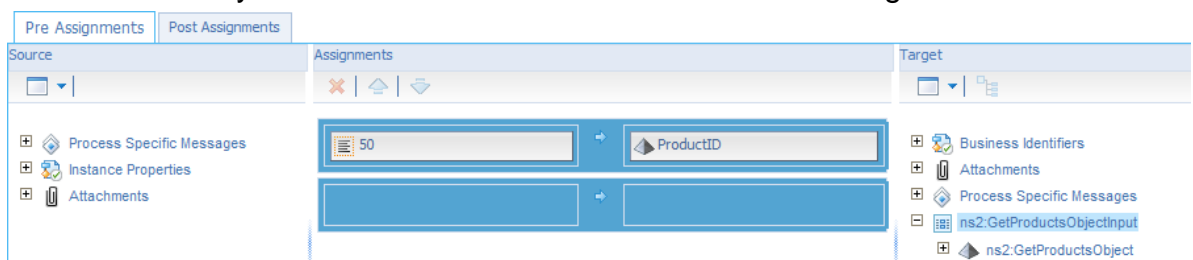


- Save the process.

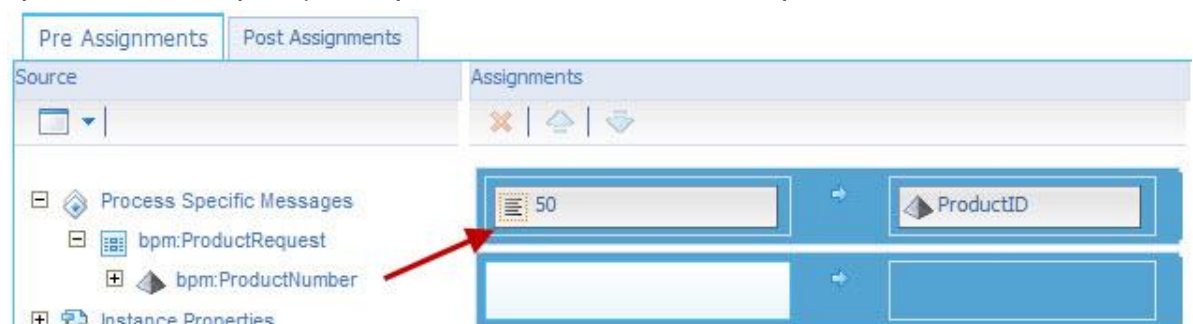
3.5.3 Applying the Process Input Message

Currently you have a fixed value assignment for the *ProductID* with value 50. You will now replace this by using the values from the process input message you have just added.

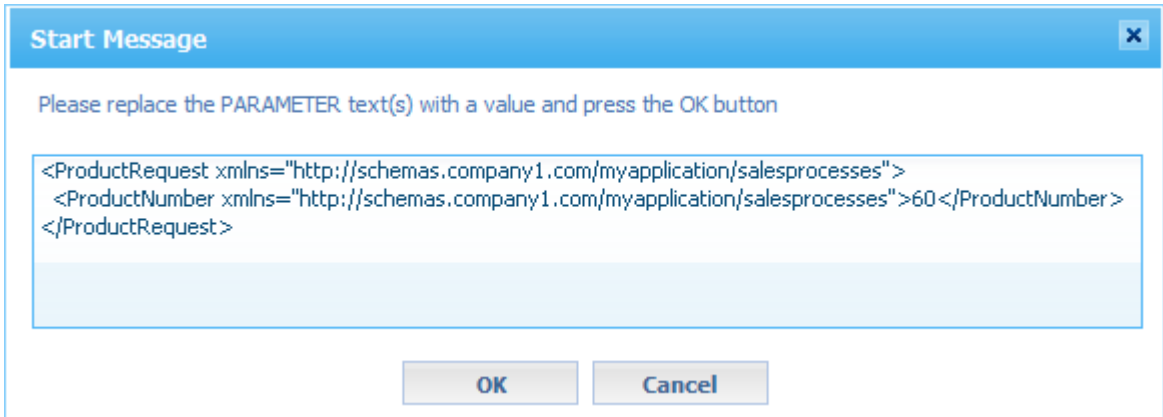
- If closed, open your BPM *GetProductInformation*.
- Go to the *Message Map* tab.
- Select the activity *Retrieve Product Information* to see the assignments.



- Click in the assignment where *ProductID* is set to 50.
- Drag and drop the *bpm:ProductNumber* (Source, process specific message: bpm:ProductRequest) on top of the 50 fixed value to replace it.



6. Go to the Model tab.
7. **Save, Validate and Publish** your process.
8. *Run* the process, in the Start Message box, replace *PARAMETER* by **60**.



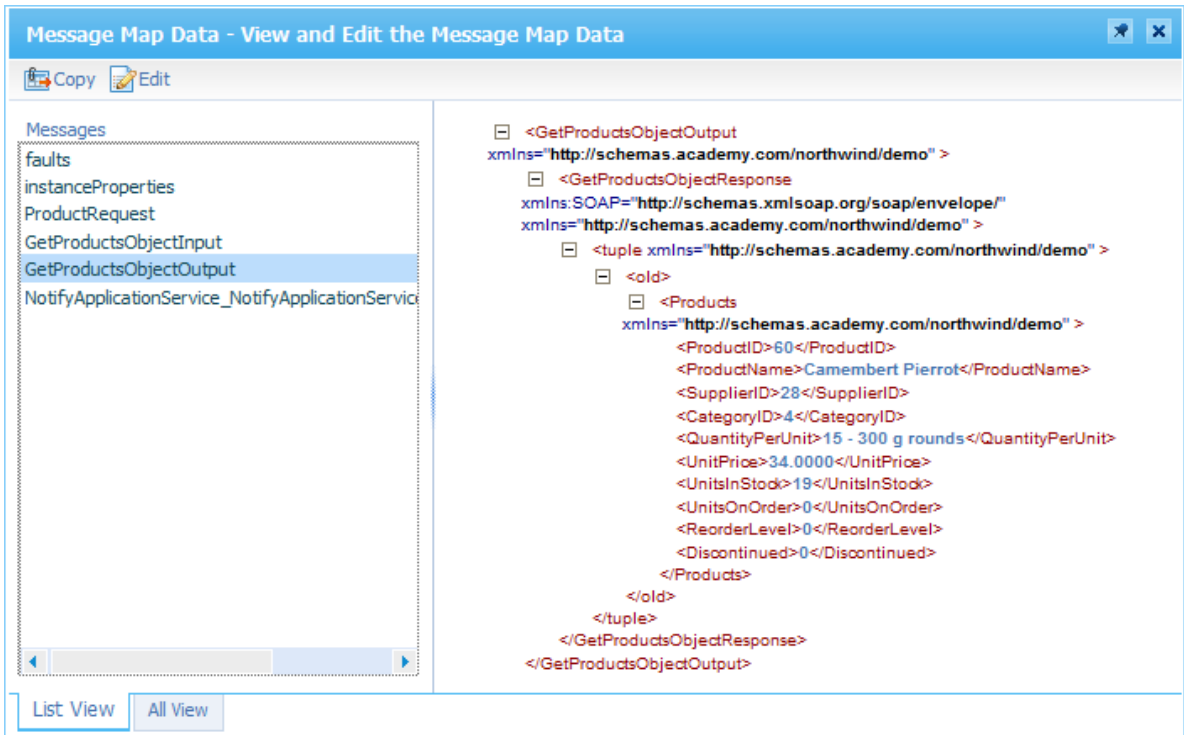
Start Message

Please replace the PARAMETER text(s) with a value and press the OK button

```
<ProductRequest xmlns="http://schemas.company1.com/myapplication/salesprocesses">
  <ProductNumber xmlns="http://schemas.company1.com/myapplication/salesprocesses">60</ProductNumber>
</ProductRequest>
```

OK Cancel

9. Click **OK**.
10. Use the process instance view (See the Appendix on page 54) to check if the correct product Information is retrieved.



Message Map Data - View and Edit the Message Map Data

Copy Edit

Messages

- faults
- instanceProperties
- ProductRequest
- GetProductsObjectInput
- GetProductsObjectOutput**
- NotifyApplicationService_NotifyApplicationService

```
<GetProductsObjectOutput
xmlns="http://schemas.academy.com/northwind/demo" >
  <GetProductsObjectResponse
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.academy.com/northwind/demo" >
    <tuple xmlns="http://schemas.academy.com/northwind/demo" >
      <old>
        <Products
xmlns="http://schemas.academy.com/northwind/demo" >
          <ProductID>60</ProductID>
          <ProductName>Camembert Pierrot</ProductName>
          <SupplierID>28</SupplierID>
          <CategoryID>4</CategoryID>
          <QuantityPerUnit>15 - 300 g rounds</QuantityPerUnit>
          <UnitPrice>34.0000</UnitPrice>
          <UnitsInStock>19</UnitsInStock>
          <UnitsOnOrder>0</UnitsOnOrder>
          <ReorderLevel>0</ReorderLevel>
          <Discontinued>0</Discontinued>
        </Products>
      </old>
    </tuple>
  </GetProductsObjectResponse>
</GetProductsObjectOutput>
```

List View All View

3.6 Running and Monitoring Processes

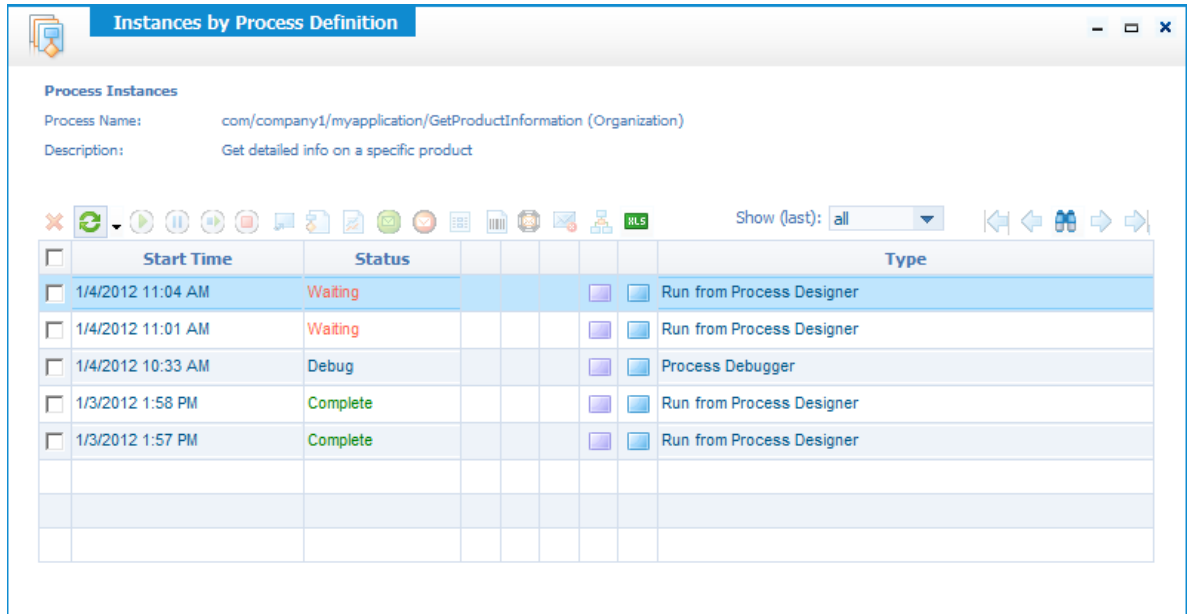
In this chapter you will see how an implemented process is presented in the process monitor.


In case you forgot how to publish, run and monitor processes take a look at the appendix on page 54.

3.6.1 Monitoring the Process

1. Run the *GetProductInformation* process.

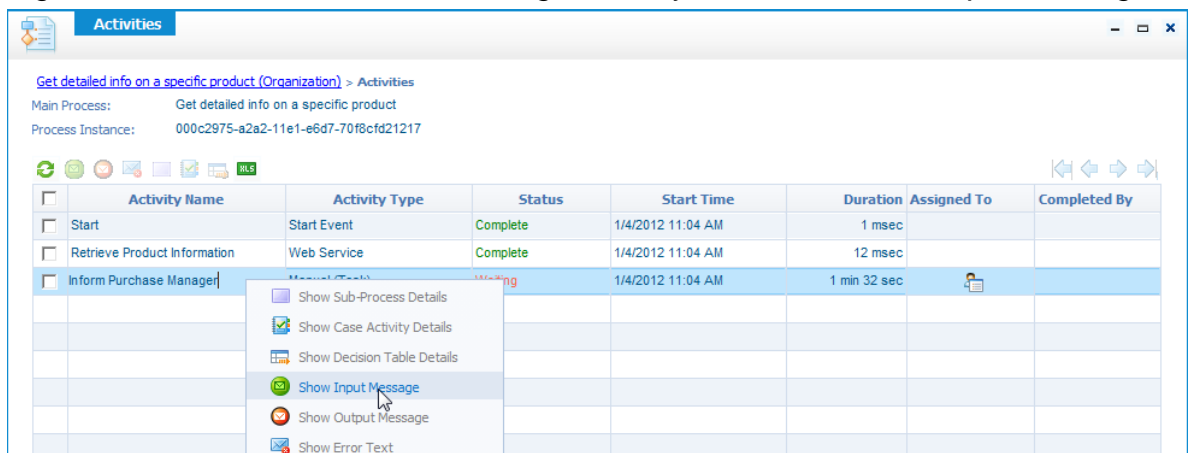
2. In the *Start Message* box change *PARAMETER* to a valid product number and click **OK**.
3. Open the process instance viewer:



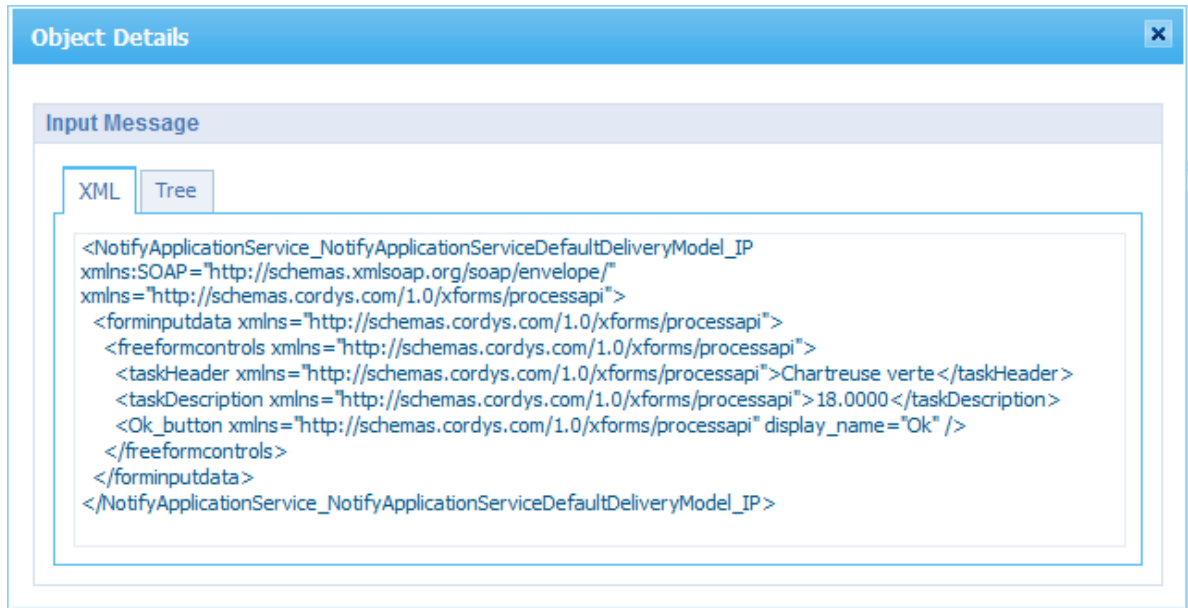
4. Select the **Show Activities** () icon for the latest instance that has the status *Waiting*.

At which activity is the process waiting and for how long?

5. Right click the *Inform Purchase Manager* activity and select *Show Input Message*.



6. The runtime input message, as defined in Defining Data Flow on page 13 is displayed:



Where do the values for *taskHeader* and *taskDescription* come from?

7. Close the *Object Details* window.
8. Right click the activity *Retrieve Product Information* and select *Show Output Message*.
9. Look at the details and close the window.
10. Close the *Activities* window.
11. Leave the *Instances by Process Definition* window open.

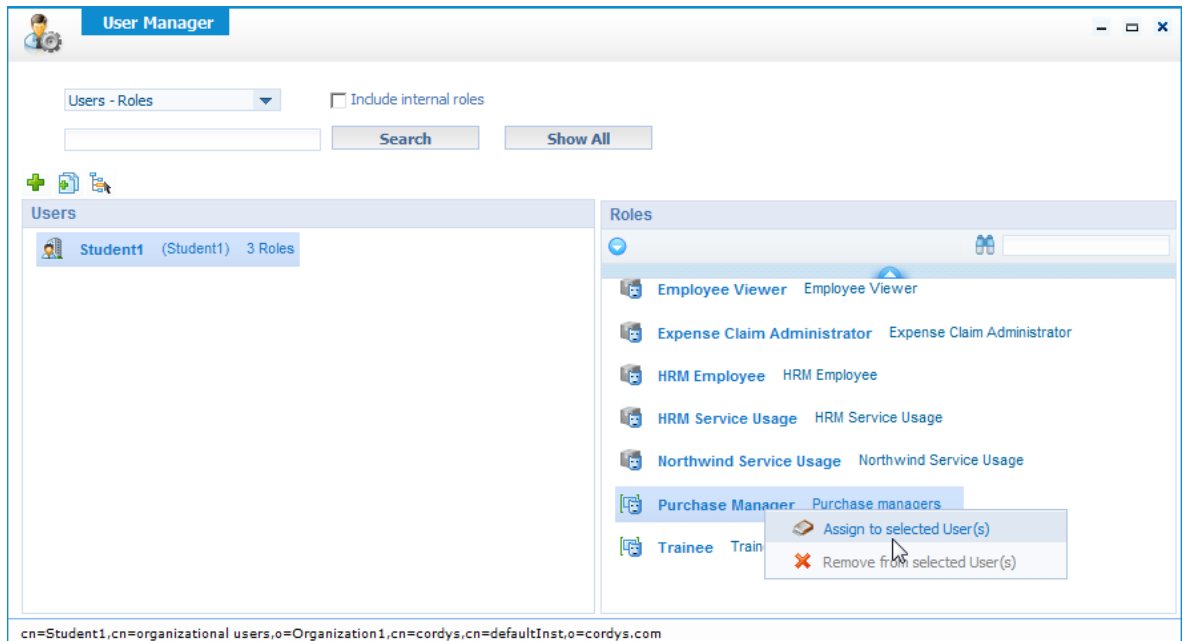
3.6.2 Task Type Execution

When you add User Interfaces to a process they are tasks by default. This means that users have to complete the tasks before the process can pick up the next activity in the flow.

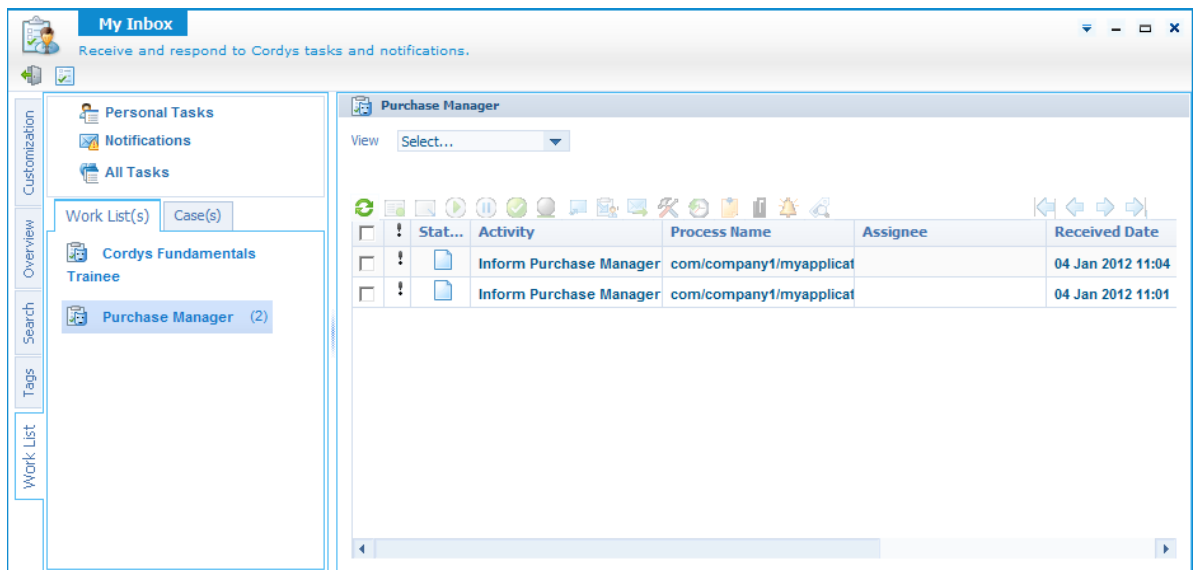
You will now complete the tasks so the waiting process can continue.

1. Open the *User Manager* (artifact).

2. Assign the *Purchase Manager* role to your student account:



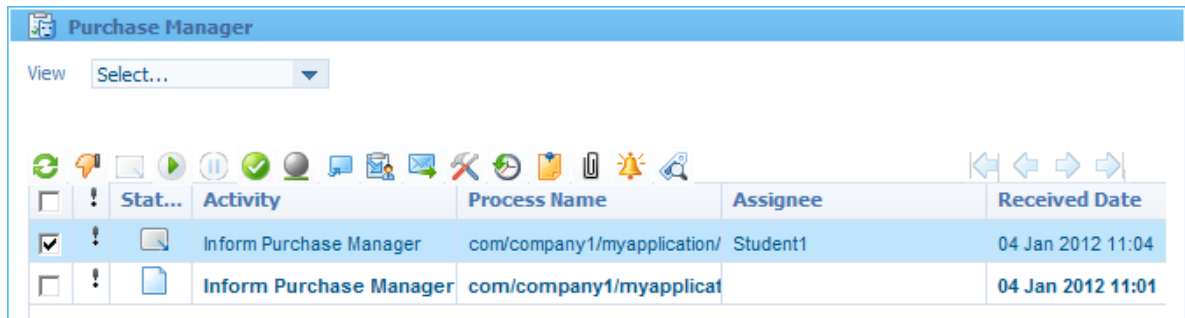
3. Open *My Inbox* (artifact).
 4. In the *Work Lists(s)* tab, select *Purchase Manager*. This will show the tasks for the selected role:



The number (2) at the end of *Purchase Manager* indicates the number of unassigned tasks, not the number of tasks, see column *Assignee*.

5. Check the first message with subject *Inform Purchase Manager*.
 6. Click **Claim Task** (📌) in the toolbar to assign the task to yourself. You have now “locked” this task for other Purchase managers.

7. Check to see that the *Assignee* field is now filled with your user name.



The screenshot shows the 'Purchase Manager' application window. At the top, there is a 'View' dropdown menu set to 'Select...'. Below this is a toolbar with various icons for process management. The main area displays a table with the following columns: 'Stat...', 'Activity', 'Process Name', 'Assignee', and 'Received Date'. The table contains two rows of data.

Stat...	Activity	Process Name	Assignee	Received Date
<input checked="" type="checkbox"/>	Inform Purchase Manager	com/company1/myapplication/	Student1	04 Jan 2012 11:04
<input type="checkbox"/>	Inform Purchase Manager	com/company1/myapplicat		04 Jan 2012 11:01

8. Double click the message to open it.
9. Click **Complete Task** (🟢) in the toolbar.
The process will continue, meaning the process has completed.
10. Close the Inbox.
11. Switch back to the *Instances by Process Definition* window.
12. Check the status and details of the process and its activities; use the *Refresh* icon to update the view.

NOTE

You will learn more details about the inbox in the module Workflow.

3.6.3 Execution of Type Info Messages

By default user interfaces are executed as Type Task activities. This means that the process will wait (status is set to *Waiting*) until the task is completed.

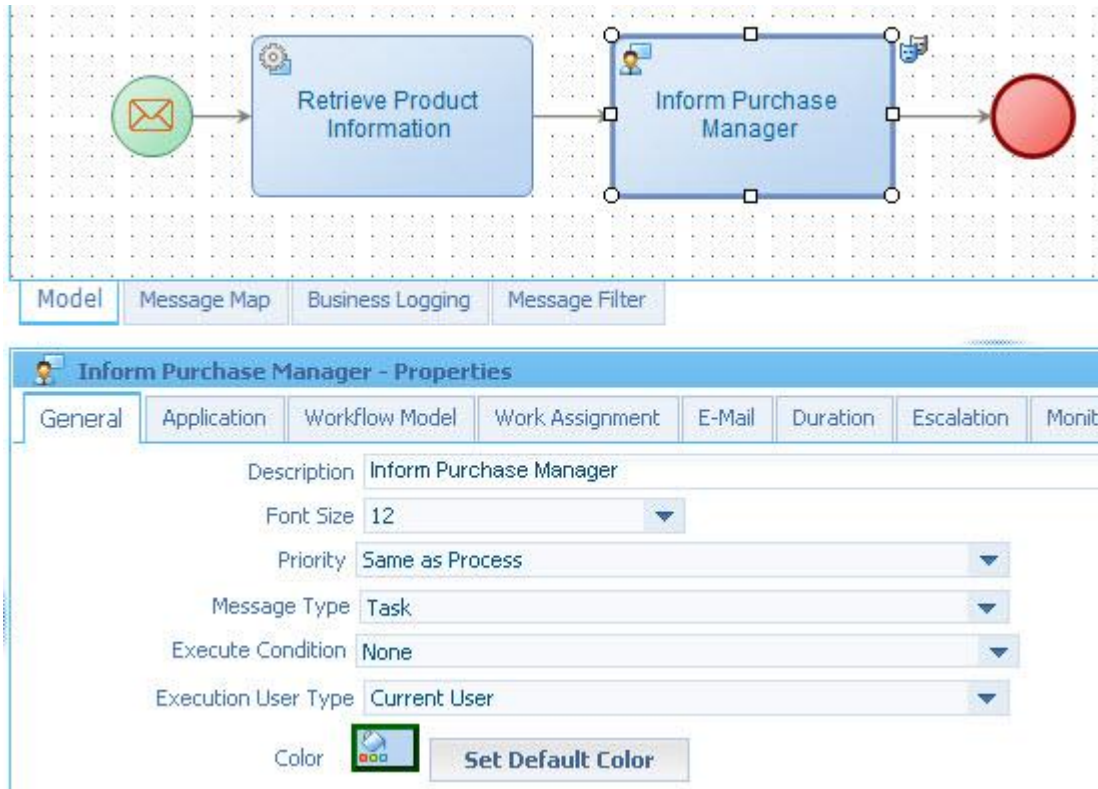
In this case, however, the process should not wait for the purchase managers to read the message and complete the task. So it would improve business performance if the process could continue immediately with the next step.

In this exercise we will change the type of the Inform Purchase manager activity from task to info.

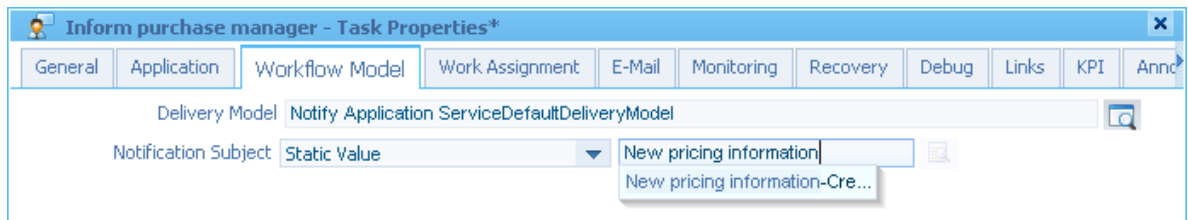
By default the subject of the message for the purchase managers is the same as the activity description. In this exercise you will assign a more informative subject header as well.

1. If closed, open your BPM *GetProductInformation*.

2. Right click the Activity *Inform Purchase Manager* and select *Properties* (or use double click):



3. Change *Message Type* from *Task* to **Info**.
4. Select the tab *Workflow Model*.
5. Change the *Notification Subject* to: **New pricing information**.



6. **Save** the process.
7. Validate, Publish and Run the process.
8. Check whether the process runs to status *Complete* at once.

Where in your inbox can you find this info message?

3.6.4 Adding and Assigning the Process Output Message

Here you will create and set the output message of the process.

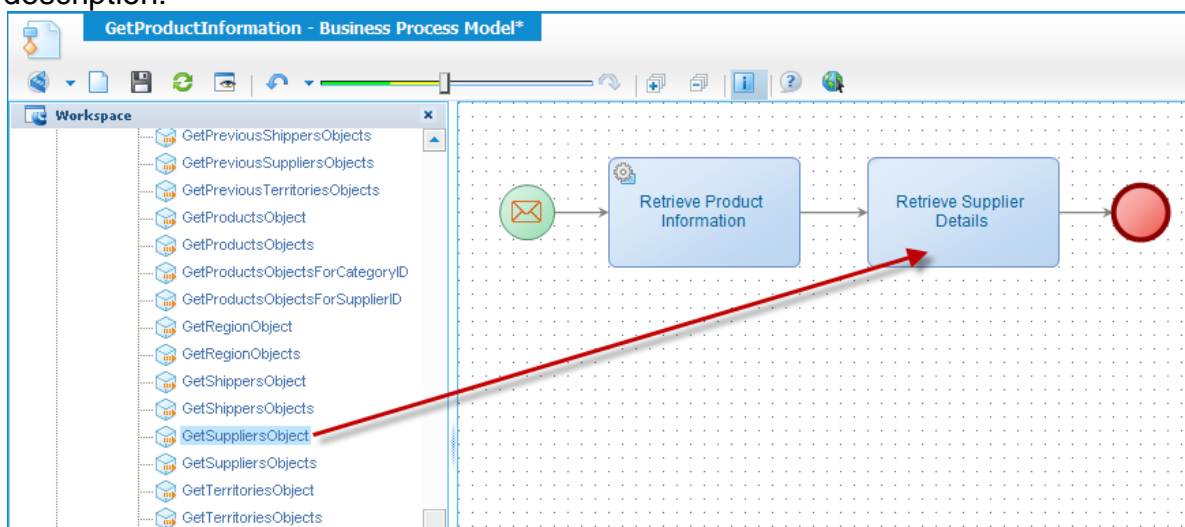
Replacing the notification activity

First you will create the structure of the output message. You will replace the notification with a web service that retrieves supplier details. After that you will copy the values of the product and the supplier to your process specific output message.

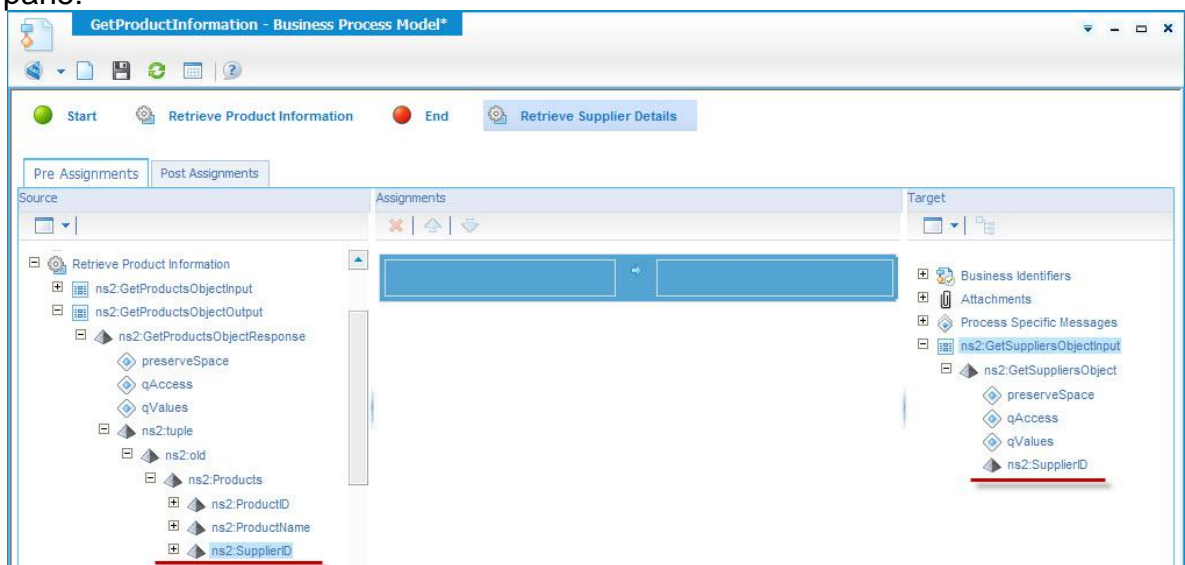
1. If closed, open your BPM *GetProductInformation*.
2. Delete the *Inform Purchase Manager* activity.
3. Select the *Retrieve Product Information* activity and add a new activity named **Retrieve Supplier Details**.
4. Link the activity to the *End* event again:



5. On the left side of the screen, select the *Workspace* tab.
6. Browse to the *GetSuppliersObject* web service in the folder *Runtime References* → *Web Services* → *Northwind.Demo*
7. Drag and drop the web service onto the activity, choose NOT to change the description:

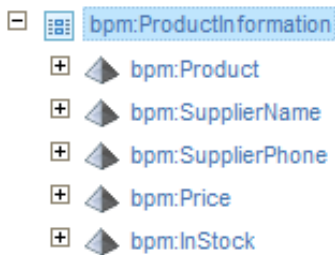


8. Go to the *Message Map* tab.
9. Select the *Retrieve Supplier Details* activity.
10. Make an assignment by selecting the *ns2:SupplierID* from the *GetProductsObjectOutput* in the source pane and holding down the CTRL-key while selecting the *ns2:SupplierID* from the *ns2:GetSuppliersObjectInput* in the target pane:



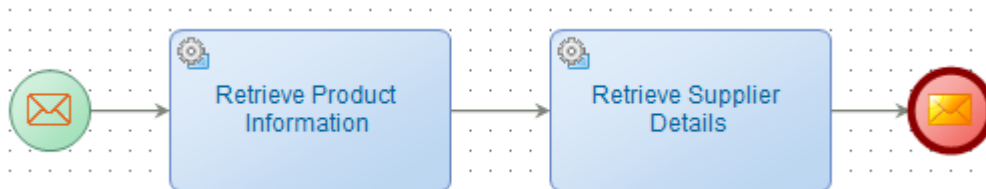
Creating the Process Specific Message

11. Right click *Process Specific Message* in the source pane and select *Create Message*.
12. Enter the name: **ProductInformation**.
13. Add 5 elements to the *ProductsInformation* named: **Product**, **SupplierName**, **SupplierPhone**, **Price** and **InStock**:



Assign the output message

14. Assign the *ProductInformation* message to the End event (See Adding an Input Message on page 21 for instructions).



Assign values to the output message

In the former part you defined the structure of the output message; in this part you will set values to the output message.

15. Go to the Message Map.
16. Select the *End* event.
17. In the *Target* box navigate to:
Process Specific Messages → *bpm:ProductInformation*

18. Drag and drop all elements into the *Assignments* pane

Assignments

✖ | ⬆ | ⬇

Product

SupplierName

SupplierPhone

Price

InStock

Use Replace Content With Fixed Value ▼

19. From the *source* pane, drag and drop the *ns2:ProductName*, *ns2:UnitPrice* and *ns2:UnitsInStock* from the *ns2:GetProductsObjectOutput* message and the *ns2:CompanyName* and *ns2:Phone* from the *ns2:GetSuppliersObjectOutput* into the corresponding areas in the *Assignments* pane:

Assignments

✖ | ⬆ | ⬇

ProductName Product

CompanyName SupplierName

Phone SupplierPhone

UnitPrice Price

UnitsInStock InStock

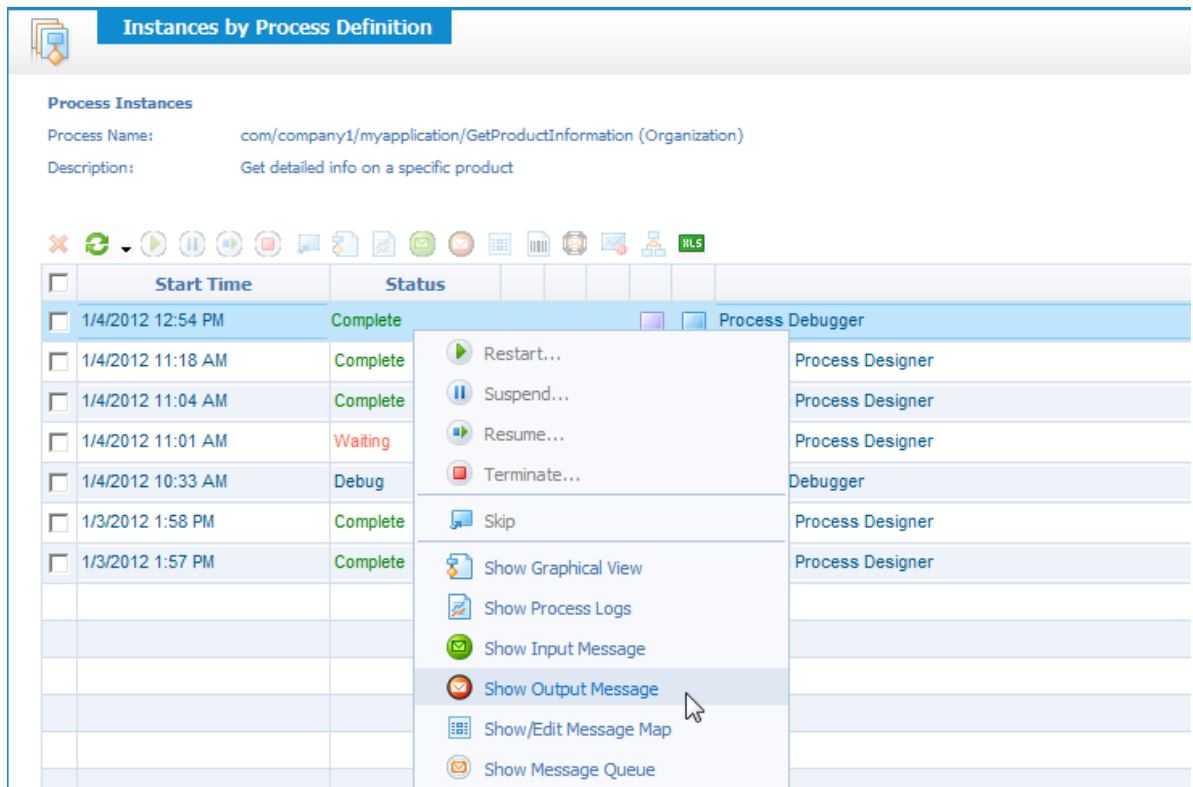
Use Replace Content With Select ▼ [Advanced...](#)

20. Save, validate and publish the process.
21. Debug the process; provide a product number in the range of **1** to **77** as start message.
22. Check that the *ProductInformation* message is created and contains the correct information.

3.6.5 Process Completion

In this paragraph you will take a closer look at the completion of the process.

1. Return to the *Instances by Process Definition* window.
2. Right click the last completed process and select *Show Output Message*.



3. Check that the correct product information is displayed.
4. Close the message box.
5. Close the process instances view.

NOTE

In case you use this process as a sub process in another process (calling process), the output message will be passed on to the calling process as soon as the sub process completes.

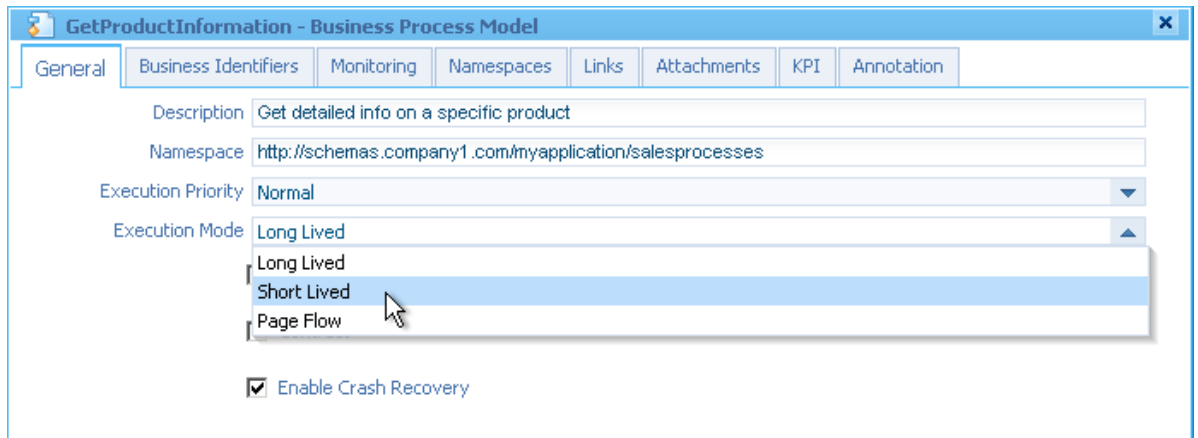
3.7 Making a Process Short Lived

In the previous exercise you have noticed that the output message is not returned to the user that has started the process. This is because any process is Long Lived by default (asynchronous execution). Hence, you will be only informed that the process is started, so no further information from the process is directly sent to you.

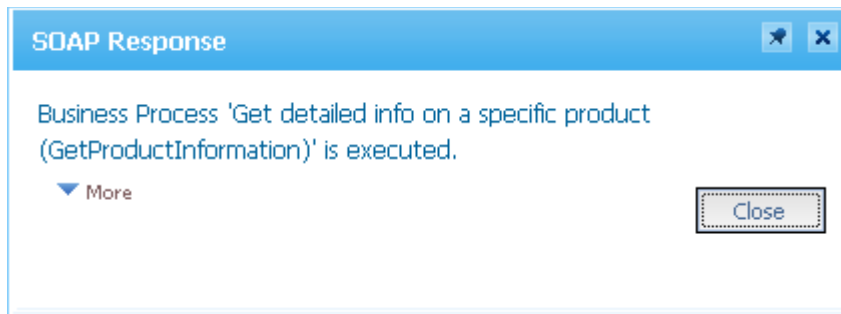
In this exercise you will make the process to be executed synchronously by changing it to a Short Lived process.

1. If closed, open your BPM *GetProductInformation*.

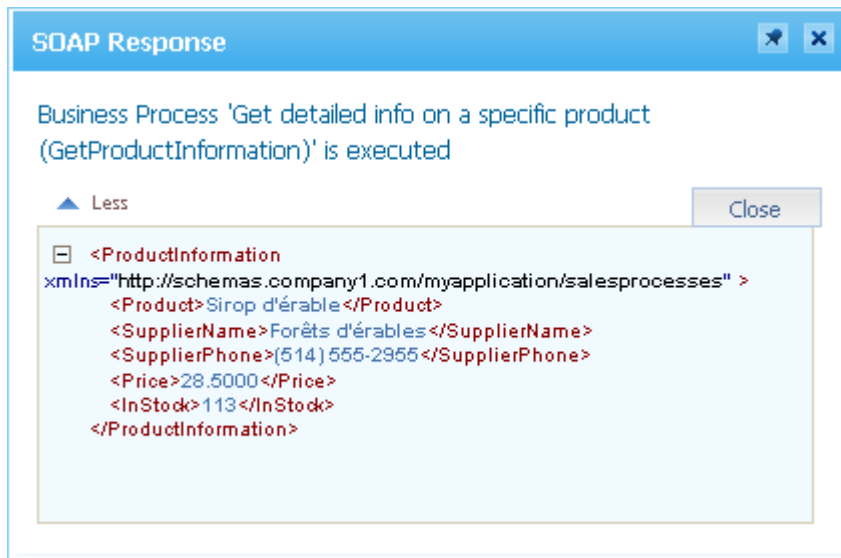
2. Right click in the process and select *Properties* (F8).
3. Change the *Execution Mode* from *Long Lived* to *Short Lived*.



4. Save the process.
 5. Validate and publish the process.
 6. Run the process with *ProductNumber* 61.
- You will get the following SOAP response from your process:



7. Click **More** to see the whole End message.



8. Click **Close**.

Can a short lived process start a long lived process?

3.8 Making a Process Web Service Enabled

In the module *Developing Web Services*, you will learn how to create a Web Service that starts this 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 is 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 **Developing Processes**.
5. Click **Make Available**.

4. Sub Processes, Decisions and Loops

To get familiar with the concepts you have learned until now, you will repeat some steps of the previous exercises. In the former exercises you have created a simple process to explore how to create and implement a business process model; in this exercise you will apply more constructs and the concept of a sub process to extend your knowledge.

The exercise is setup according to the Cordys Closed Loop BPM cycle methodology.

4.1 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 shipping department of ACME inc. often runs into a situation that orders cannot be shipped completely as not all items on the order have enough stock. Moreover, these products have not been ordered yet, so the shipment of the order is delayed or results in many backorders.

What is the target of the improvement?

4.2 Design & Model

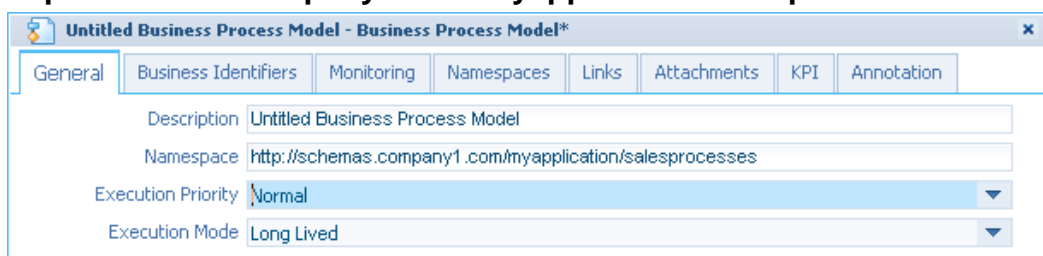
Here you will actually make a model of the stock replenishment process.

4.2.1 Designing the Main Business Process Model

In this part you will model the main process, in fact the generic part.

1. If closed, open *My Application Project*.
2. Navigate to *Business Process Models* → *com* → *companyX* → *myapplication*.
3. Right click the *myapplication* folder and select *New* → *Business Process Model*.
4. Right click in the process and select *Properties*.
5. Change the namespace to the following:

<http://schemas.companyX.com/myapplication/salesprocesses>



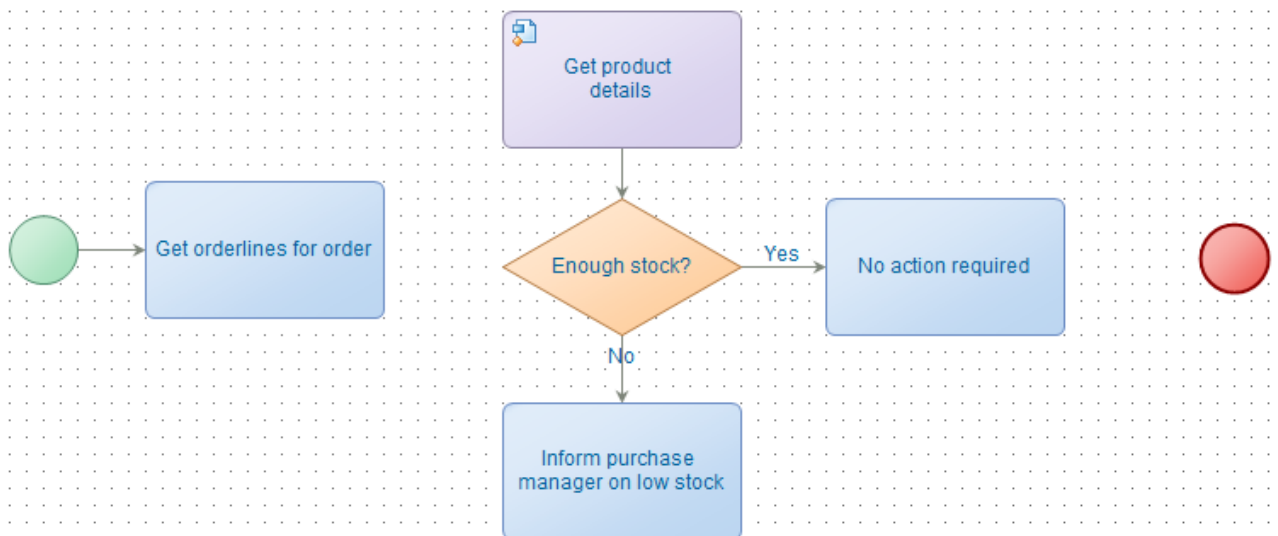
6. Save the process and provide the following values:


Field	Value
Name	StockReplenishment
Description	Order stock for products on order
Location	Prefilled with: My Application Project/Business Process Models/com/companyX/myapplication

7. Click OK.

4.2.2 Adding the Basic Constructs

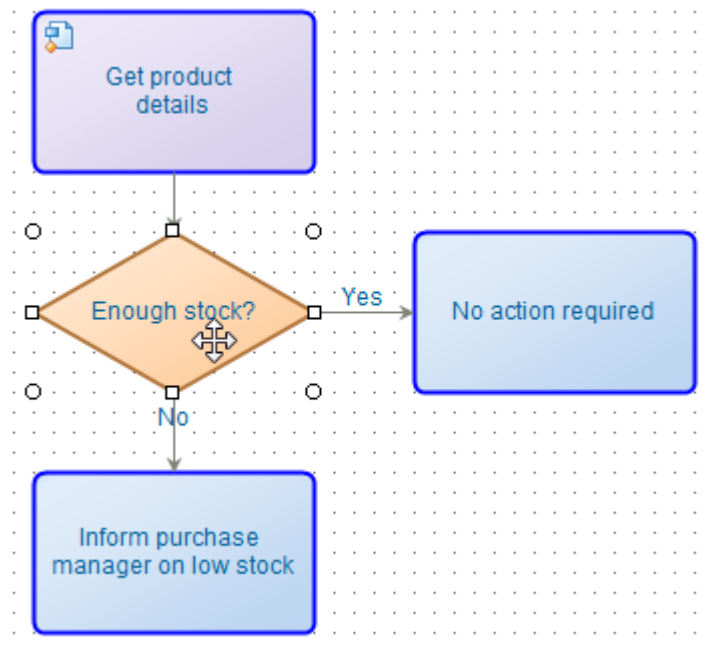
In the **next steps** you will model the process like this, use this screenshot as reference while performing the steps:



- Add the following constructs to the modeling area (refer to the screenshot above):
 - Start Event
 - Activity
 - Independent Sub-Process ()
 - Decision
 - 2 more Activities
 - End event
- Make sure the constructs are aligned properly.
- Add connectors between the different constructs (when not done automatically while designing) like in the screenshot. (see the appendix on page 51)

NOTE

When moving a construct on the model, the activities and sub processes will be highlighted in bright blue to indicate the location where it is properly aligned with the construct(s) that it is attached to:



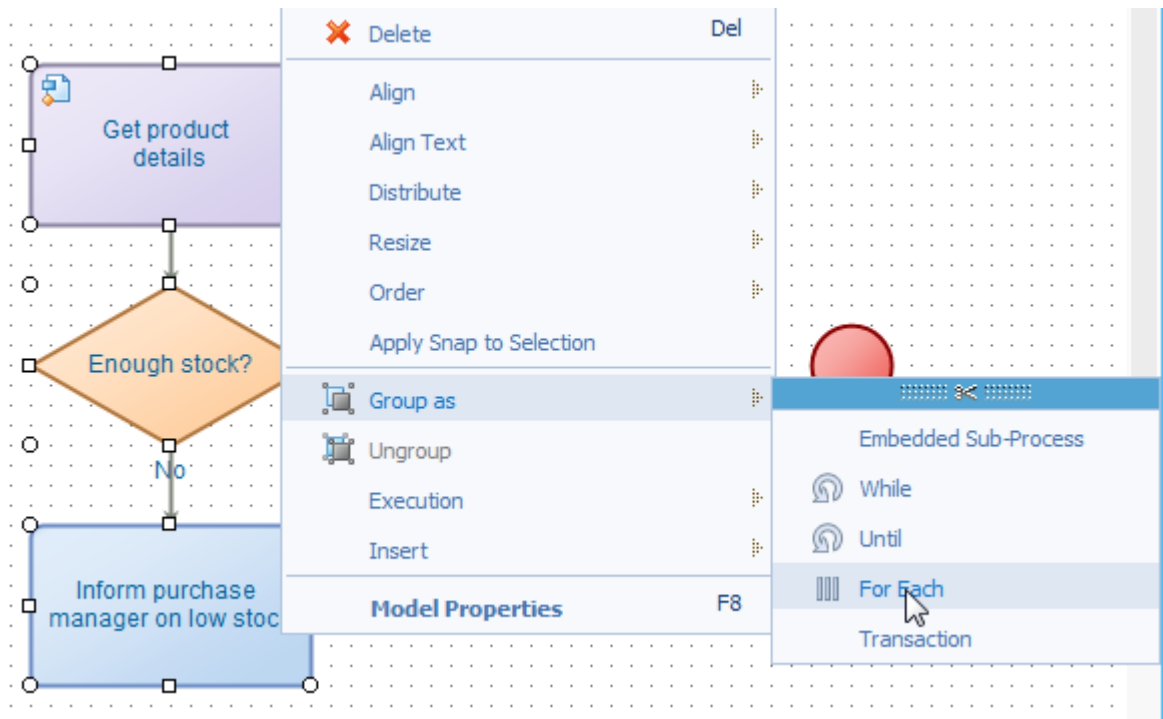
4. Change the description of the constructs to:

Construct	Description
Activity 1	Get orderlines for order
Sub Process	Get product details
Decision	Enough stock?
Activity 2	No action required
Activity 3	Inform purchase manager on low stock
Decision to No action required	Yes
Decision to Inform purchase...	No

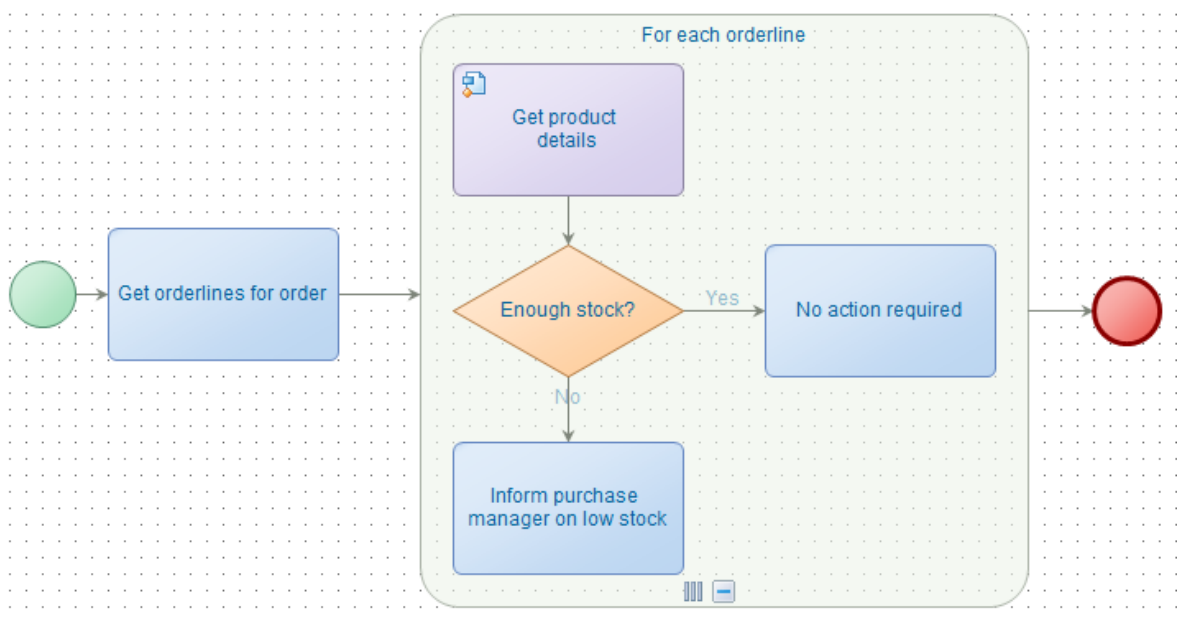
4.2.3 Adding a For Each Construct

1. Make a rectangle around all the constructs *between the Get orderliness for order construct and the End event*, to select them.

2. Right click and select *Group As* → *For Each*.



3. Change the description of the *ForEach* to: **For each orderline**.
4. Add a connector from the *Get orderliness for order* activity to the *For Each*.
5. Add a connector from the *For Each* to the *End* event.
6. The complete process design should look similar to:



7. Save the process.

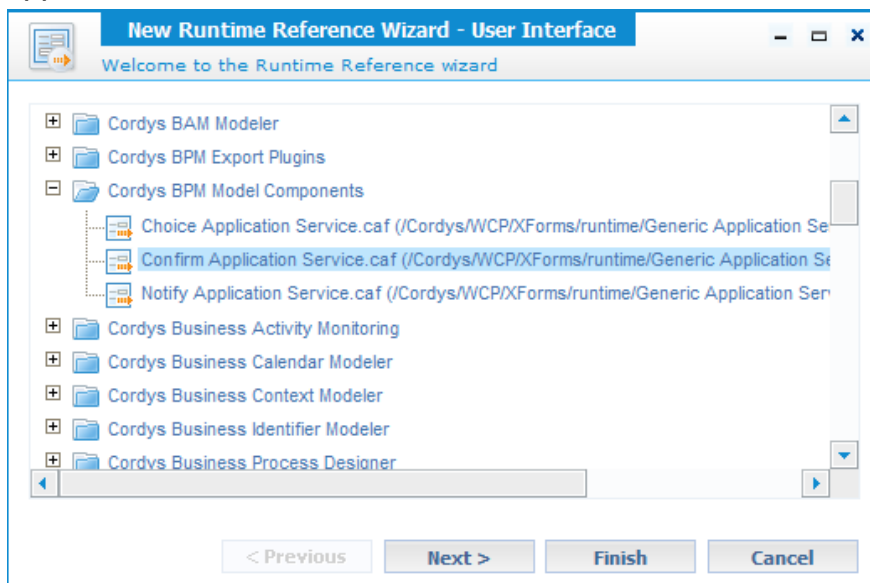
4.3 Develop & Deploy

In this part you will design and implement the activities and sub process in this main process. You will re-use the process that you have designed in previous exercises.

4.3.1 Add Runtime References

For the Stock replenishment process, you will use existing services and user interfaces so you do not have to create them yourself.

1. In the workspace tree navigate to *Runtime References* → *User Interfaces*.
2. Right click the folder *User interfaces*.
3. Select *Add Runtime Reference* and select *User Interface*.
4. Select from the Application *Cordys BPM Model Components*, the *Confirm Application Service*.



5. Click **Finish**.

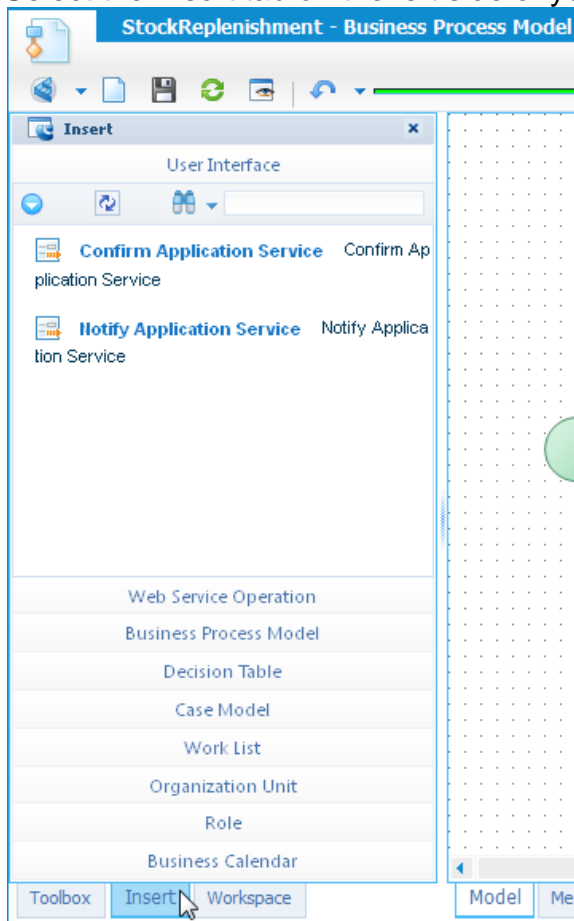
4.3.2 Implementing the constructs

Now that you have done the design of your business process model, you will implement it with existing elements in your workspace and then define the data flow.

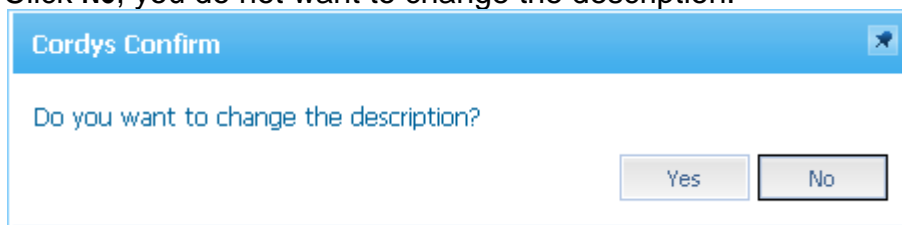
Add the web service, (sub)process, user interface and role

1. If closed, open the *StockReplenishment* process.

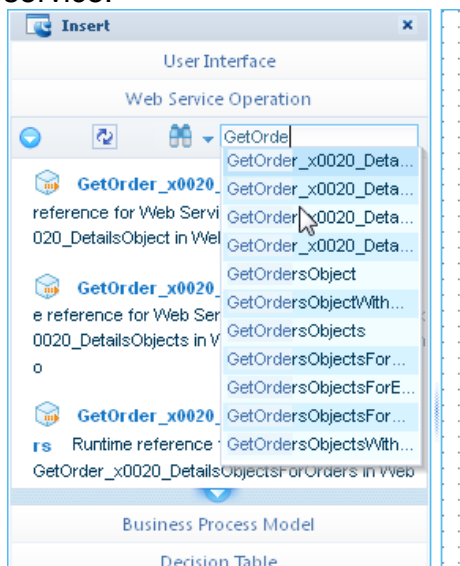
2. Select the *Insert* tab on the left side of your modeler:



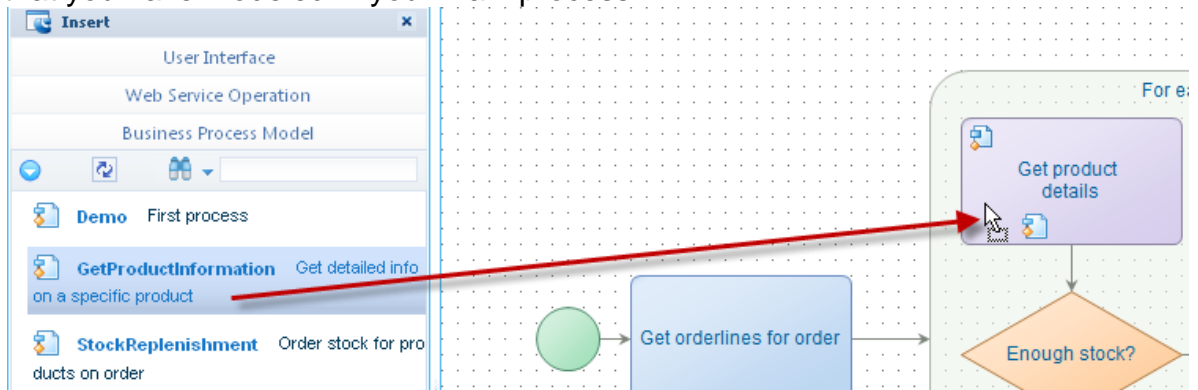
3. Select *User Interface* and drag and drop the *Confirm Application Service* onto the activity *Inform purchase manager on low stock*.
4. Click **No**, you do not want to change the description:



5. In the *Insert* pane, select *Web Service Operation*.
6. Use the search box to find the *GetOrder_x0020_DetailsObjectsForOrderID* web service.



7. Drag and drop this web service onto the activity *Get orderlines for order*, leaving the description as it is.
8. In the *Insert* pane, select *Business Process Model*.
9. Drag and drop the *GetProductInformation* model onto the independent sub process that you have modeled in your main process:



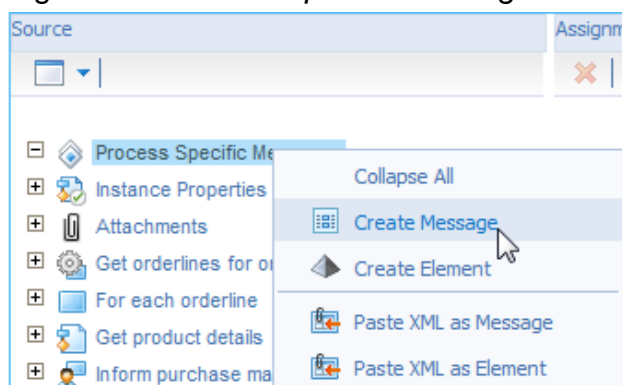
10. Finally, select *Role* in the *Insert* pane.
11. Drag and drop the *Purchase Manager* role onto the *Inform purchase manager on low stock* activity.

NOTE

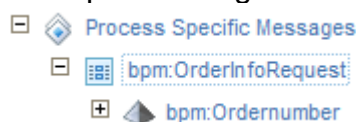
The “No action required” activity will not be implemented with a web service or user interface. This activity is here only for visibility reasons, it can be left out as well: the process will then check only the ‘No’ condition and move to the next set of data.

Create process specific input message

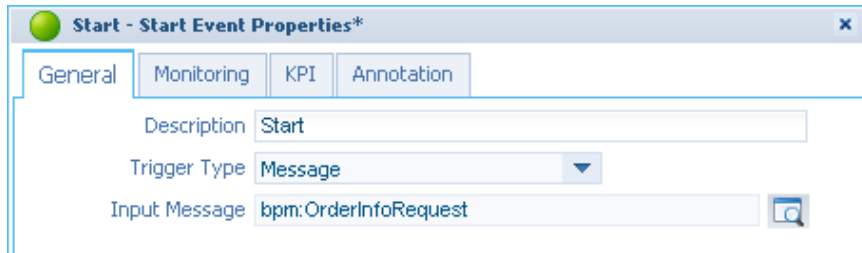
1. Go to the *Message Map*.
2. Right click *Process Specific Messages* and select *Create Message*.



3. Enter the name **OrderInfoRequest**.
4. Right click the created message and select *Create Element*.
5. Enter the name **Ordernumber**.
6. The input message should look like:

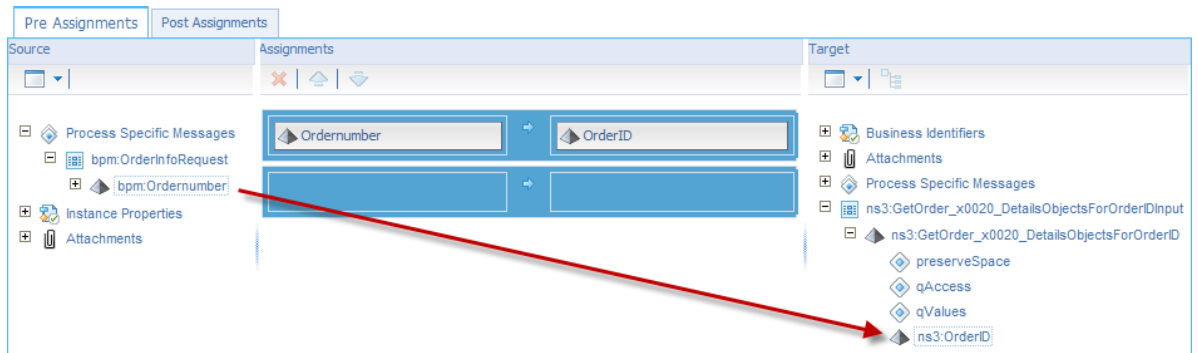


7. Add the *OrderInfoRequest* message to the start Event (see Adding an Input Message on page 21).

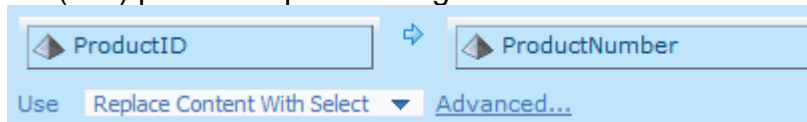


Add Assignments

1. Go to the *Message Map*.
2. Select the *Get orderlines for order* activity.
3. Create an assignment from:
Ordernumber (*OrderInfoRequest* → *Ordernumber*)
 to
OrderID (*GetOrder_x0020_DetailsObjectsForOrderIDInput* → *GetOrder_x0020_DetailsObjectsForOrderID* → *OrderID*)



4. Select the *Get product details* process.
5. Create an assignment from the *ProductID* in the response of the *GetOrder_x0020_DetailsObjectForOrderID* web service to the *Productnumber* of the (sub) process input message:

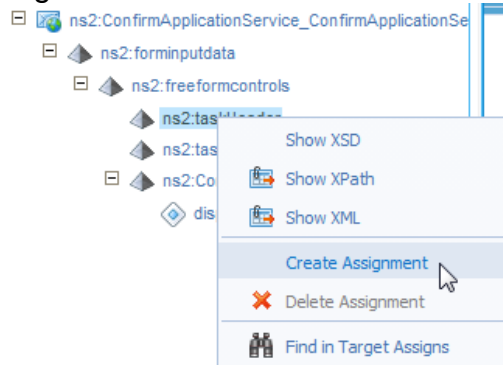


4.3.3 Using the XPath Editor

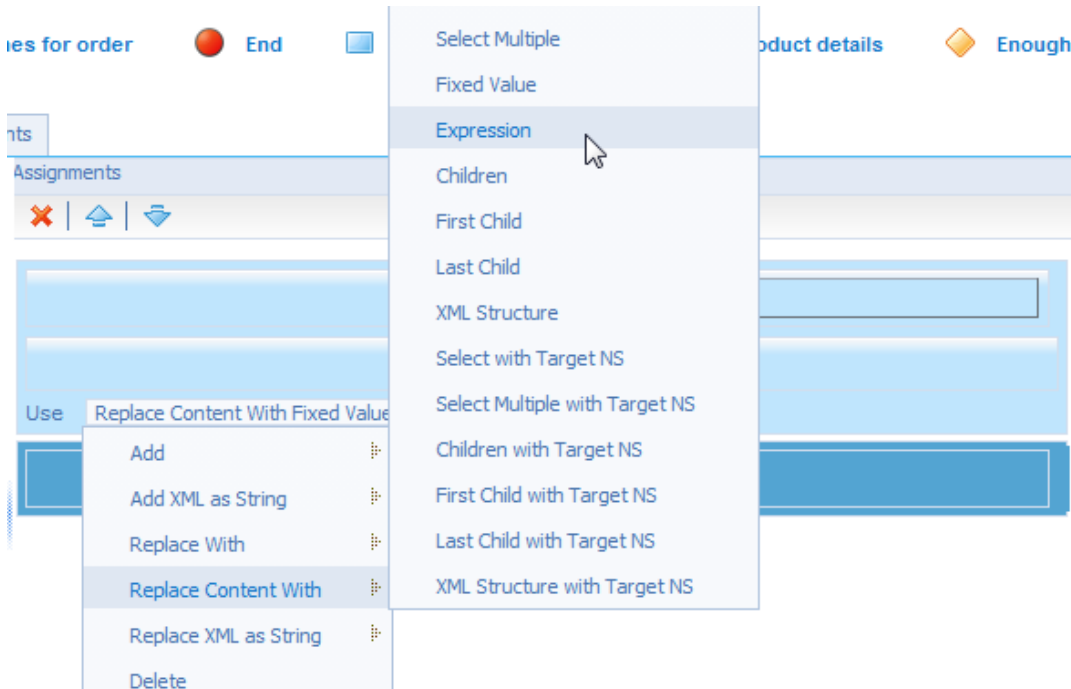
The task that is sent to the purchase manager needs to have a more readable message than just the name of, for example, a product. For this you will use the XPath editor to create your own expressions.

1. Select the *Inform purchase manager on low stock* activity.
2. In the target box, right click the *ns2:forminputdata* element below the *ConfirmApplicationService* message and select *Expand all*.

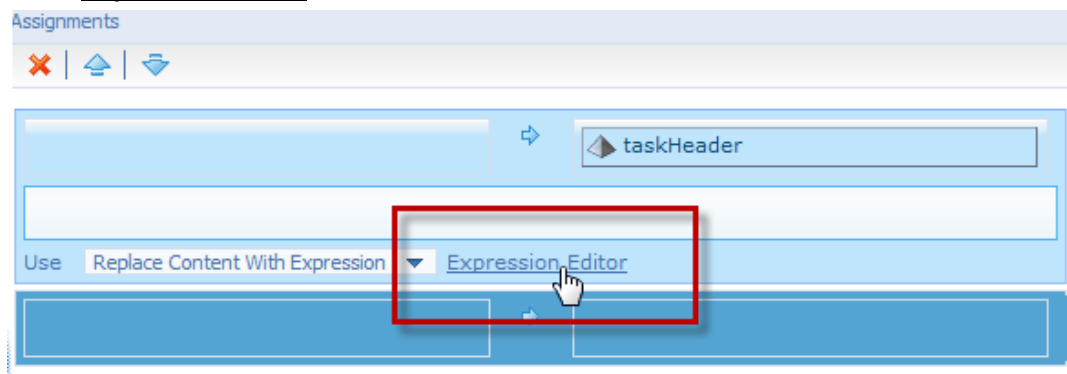
3. Right click *ns2:taskHeader* and select *Create Assignment*



4. From the *Use* select box, of the created assignment, select *Replace Content With* → *Expression*.

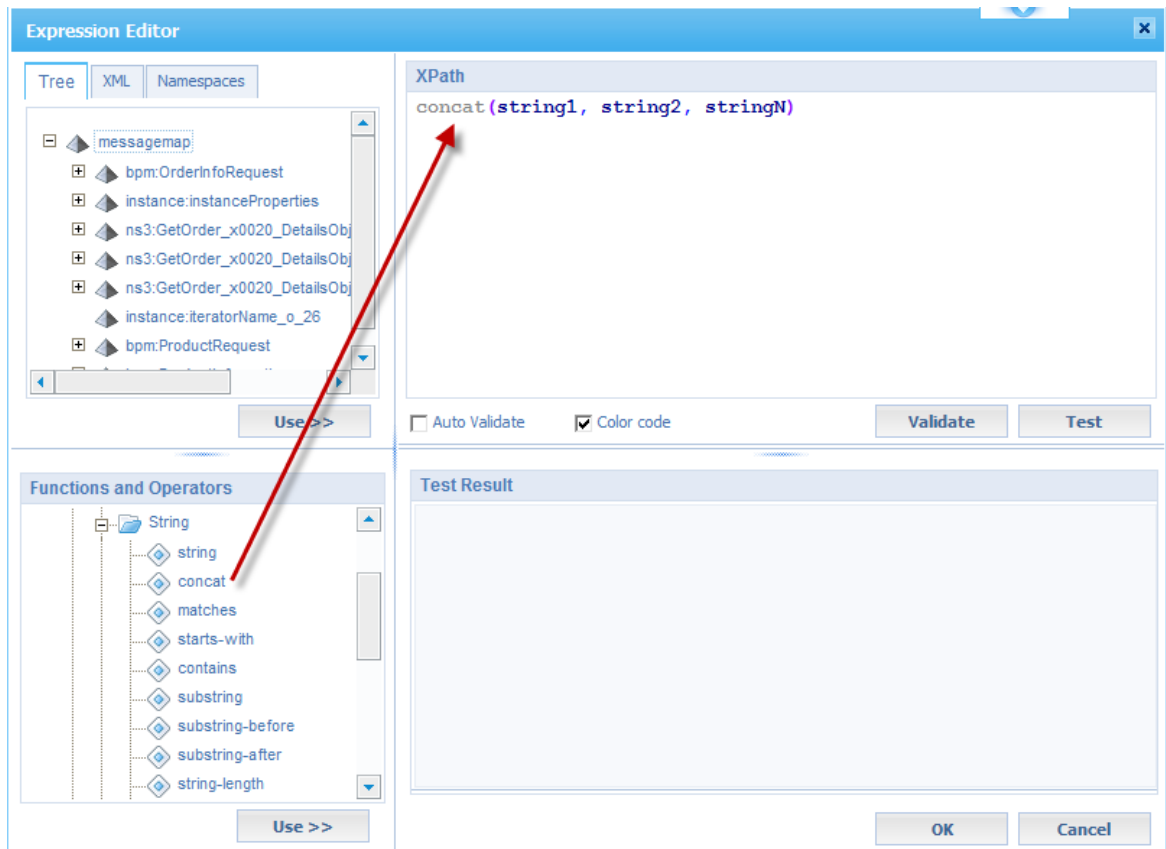


5. Click **Expression Editor**.



The Expression Editor opens. To compose the required XPath expression, you will use the concat function to concatenate strings together into one string.

6. From the *Functions and Operators* box, drag and drop the string function *concat* into the *XPath* box:

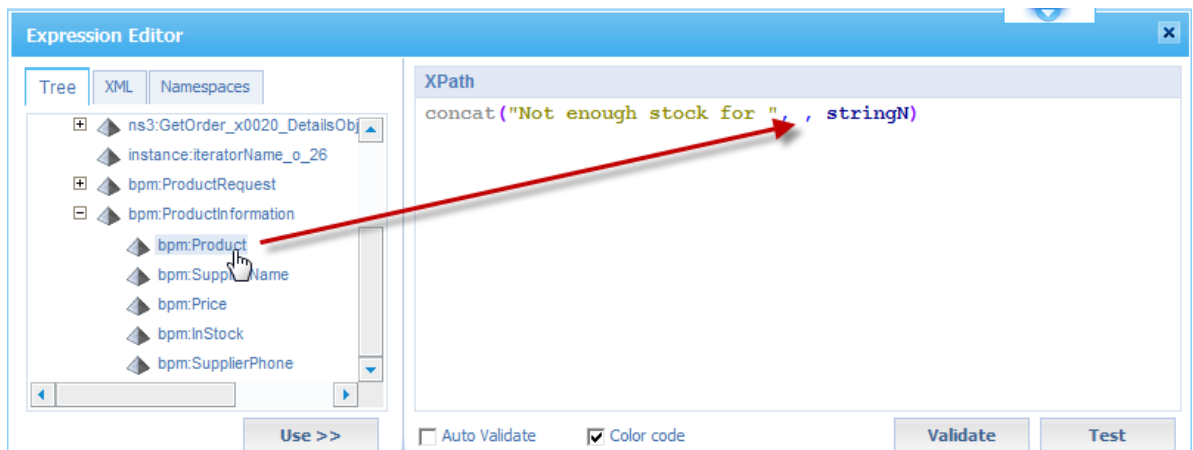


7. Double click *string1* and enter the text: "Not enough stock for ":

XPath

```
concat("Not enough stock for ", string2, stringN)
```

8. Remove the part **string2**.
9. From the *Tree* box, drag and drop *bpm:Product* (*messagemap* → *bpm:ProductInformation* → *bpm:Product*) at the location of the deleted *string2*.



10. Remove the part **", stringN"**.
11. The XPath expression should look like this:

XPath

```
concat("Not enough stock for ", bpm:ProductInformation/
bpm:Product/text())
```

```
concat("Not enough stock for ",
bpm:ProductInformation/bpm:Product/text())
```

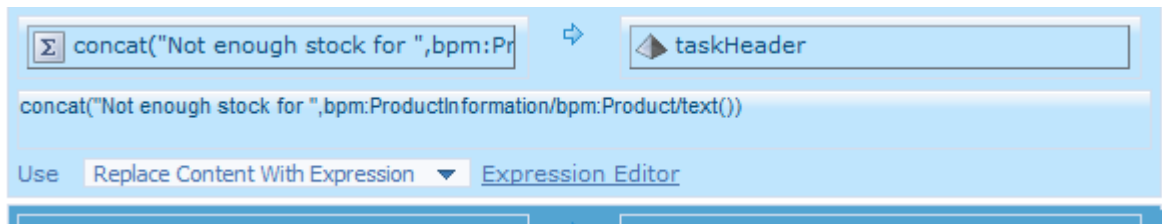
12. Click **Validate** to check the XPath syntax of your expression.
13. Click **Test** to check the result of the expression:

Test Result

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

Not enough stock for ValueOf_Product

14. Click **OK** to save your XPath expression and close the editor.



15. Create another assignment for the *taskDescription*.
16. Change it into an *expression* with the following XPath syntax:

```
concat("Call ", bpm:ProductInformation/bpm:SupplierName/text(), "
on number: ", bpm:ProductInformation/bpm:SupplierPhone/text())
```

17. Click **Test** to check the result of the expression:

Test Result

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

Call ValueOf_SupplierName on number: ValueOf_SupplierPhone

18. Create an assignment for the attribute *display_name* of the *Confirm_button* with a fixed value **Re-ordered**.



19. Save your model.

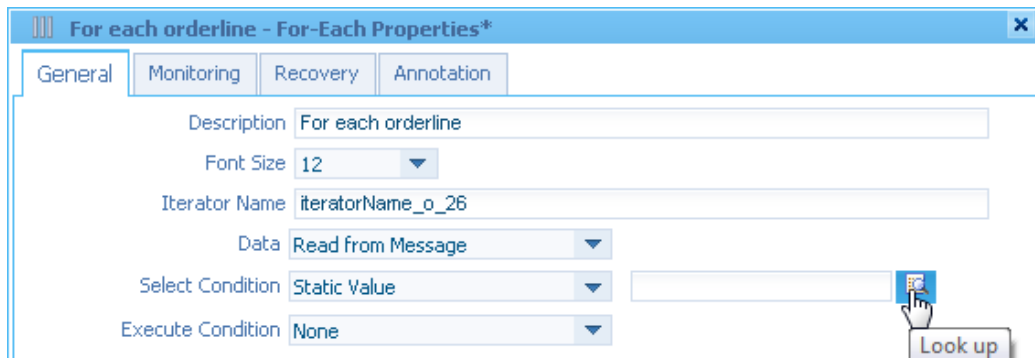
4.3.4 Implementing the For Each Construct

In this exercise the *For Each* construct added earlier is made executable.

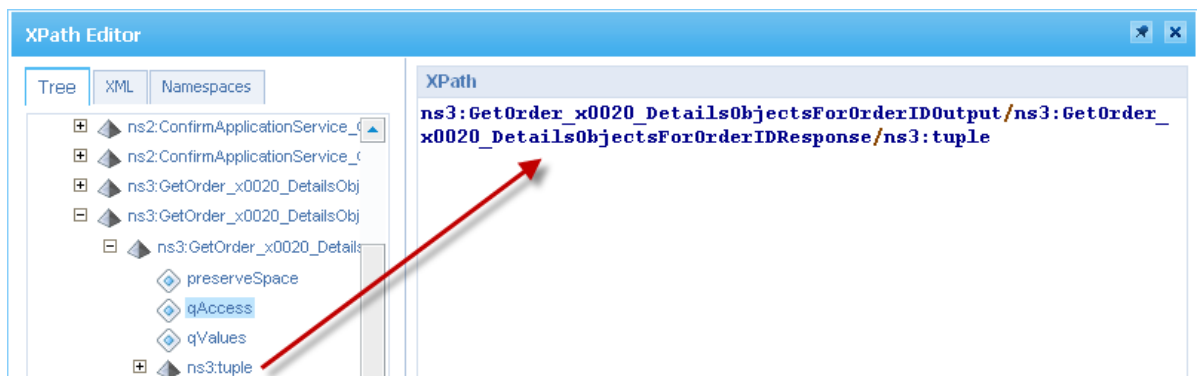
The *Select Condition* property of the construct defines which element will be used to iterate the for each loop.

In this case, for each order line (each `Order_x0020_DetailsObject`) from the output message, the decision will be evaluated.

1. Select the *Model* tab.
2. Open the properties of the *For Each* construct.
3. Click the **Lookup** button for the *Select Condition*.



4. Drag and drop the `ns3:tuple` element (from `ns3:GetOrder_x0020_DetailsObjectsForOrderIDOutput/ns3:GetOrder_x0020_DetailsObjectsForOrderIDResponse`) into the XPath box.



5. Click **OK**.

What is the function of the iterator name?

6. Change the *Iterator Name* to **iterator_orderline**.

4.3.5 Implementing the Decision Construct

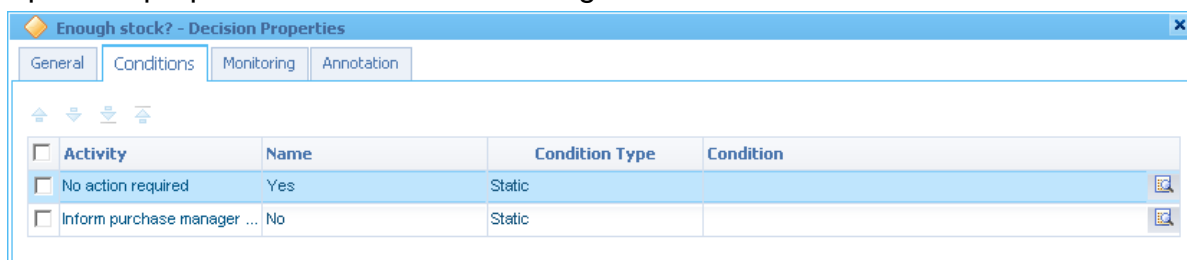
In this part of the exercise the *Decision* construct added earlier will be made executable. On the *Conditions* tab page of the *Decision* construct you will define in which conditions the process should execute in a certain direction.

In this case, for each order line a check will be done whether there are enough products on stock compared to the quantity on the order line.

You can specify the condition by:

- Using the XPath Editor
- Copying the condition and change the values in Conditions tab page directly:

1. Open the properties of the *Decision* and go to the *Conditions* tab:

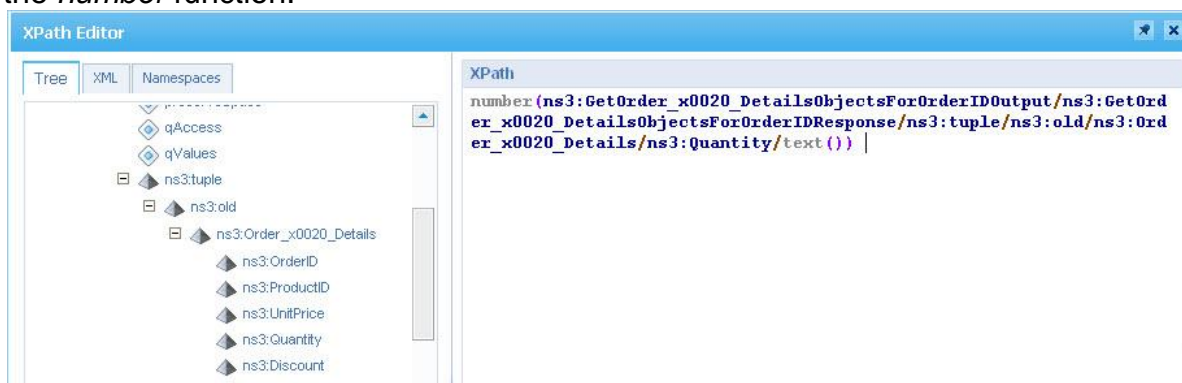


2. Set the *Condition Type* for the activity *No action required* to *Default*.
3. Select the **Look up** button for the activity *Inform purchase manager*.
4. Add the *number* (numeric) function.

XPath

```
number (argumentToCastOptional)
```

5. Place the element *ns3:Quantity* (*ns3:GetOrder_x0020_DetailsObjectsForOrderIDOutput/ns3:GetOrder_x0020_DetailsObjectsForOrderIDResponse/ns3:tuple/ns3:old/ns3:Order_x0020_Details*) into the *number* function.



NOTE

By using the *number* function you force the value to be cast to a number to avoid number string comparison.

6. Type a *greater than* sign **>** at the end.
7. Add a *number* function at the end and use the *bpm:InStock* (from *bpm:ProductInformation*).
8. Your expression should look like this:

XPath

```
number (ns3:GetOrder_x0020_DetailsObjectsForOrderIDOutput/ns3:GetOrder_x0020_DetailsObjectsForOrderIDResponse/ns3:tuple/ns3:old/ns3:Order_x0020_Details/ns3:Quantity/text()) > number (bpm:ProductInformation/bpm:InStock/text())
```

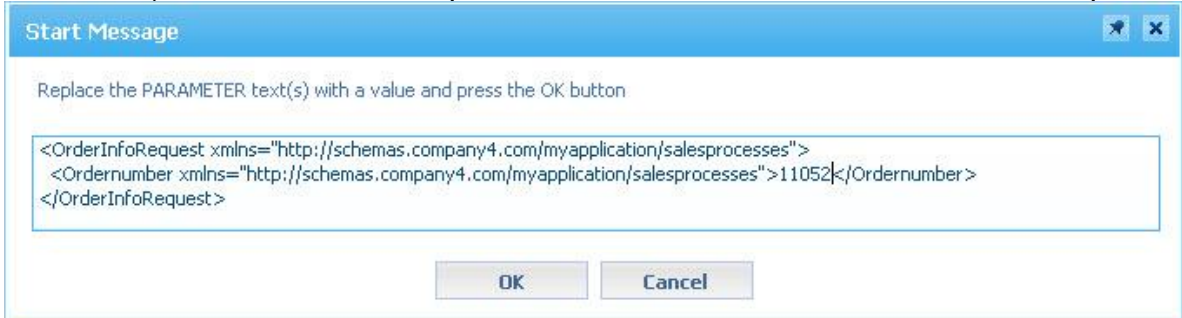
9. Validate and test the XPath expression (you can provide values at the XML tab).
10. Click **OK** to save the expression.
11. Close the properties screen.
12. **Save**, validate and publish the process.

4.4 Running and Monitoring the Process

You will now run your process and monitor whether the process is correctly working.

4.4.1 Running the Process

1. Run the process.
2. Fill the start message with a valid order number (e.g. any number between **11050** and **11060**), order **11052** has 1 product with low stock, order **11056** has multiple.



Start Message

Replace the PARAMETER text(s) with a value and press the OK button

```
<OrderInfoRequest xmlns="http://schemas.company4.com/myapplication/salesprocesses">
  <Ordernumber xmlns="http://schemas.company4.com/myapplication/salesprocesses">11052</Ordernumber>
</OrderInfoRequest>
```

OK Cancel

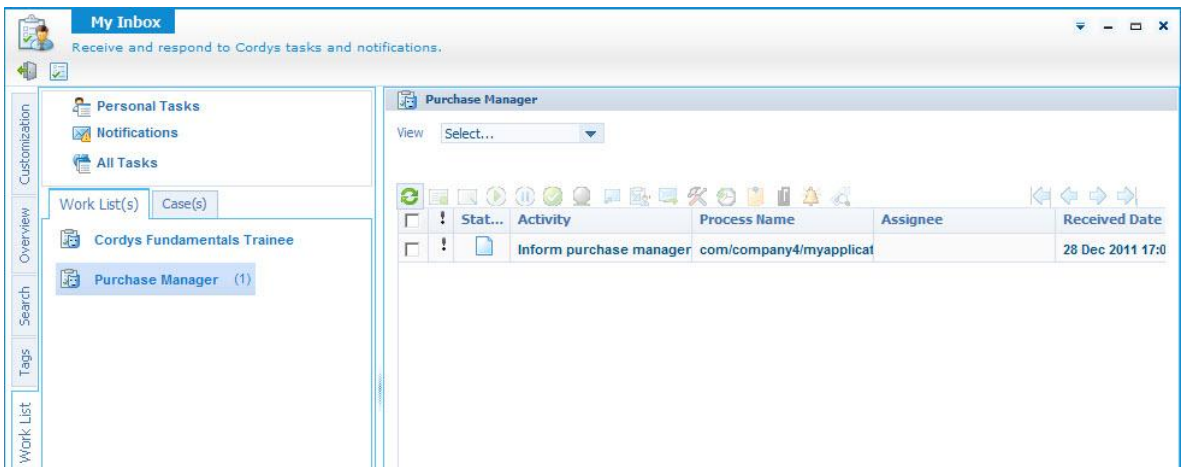
3. Click OK.

4.4.2 Monitoring Process Execution

1. Open the *Process Instance Manager* () from *My Application*.

What is the status for the “sub” process GetProductInformation?

2. Open *My Inbox*.
3. Select the *Work List* of the *Purchase Manager*.



My Inbox
Receive and respond to Cordys tasks and notifications.

Personal Tasks
Notifications
All Tasks


Work List(s) Case(s)

Cordys Fundamentals Trainee
Purchase Manager (1)

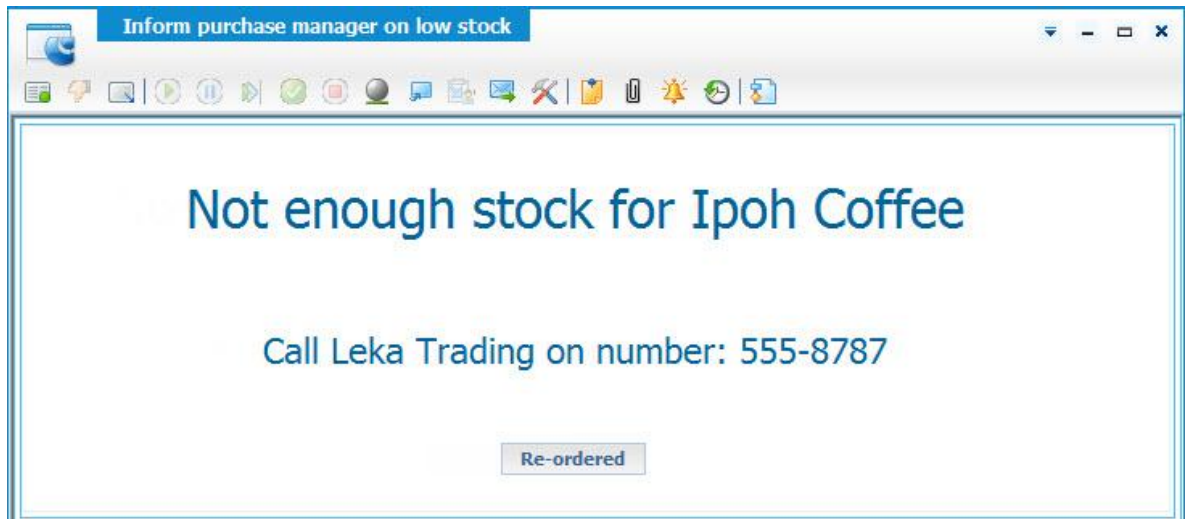
Purchase Manager

View Select...

Stat...	Activity	Process Name	Assignee	Received Date
	Inform purchase manager	com/company4/myapplicat		28 Dec 2011 17:0

4. Double click the message to open it.
5. Claim the task ().

6. Check that the product name, supplier name and phone number values are correct.



7. Click the **Re-ordered** button.
8. Switch to the *Process Instance Manager* window.
9. Refresh the list and check whether the process is completed or if more tasks are waiting.

What improvements can you think of to optimize the StockReplenishment process both in the design model as in the running part?


4.5 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 is 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 **Developing Processes with Sub Process**.
5. Click **Make Available**.

5. Appendix

5.1 Adding Constructs to the Process

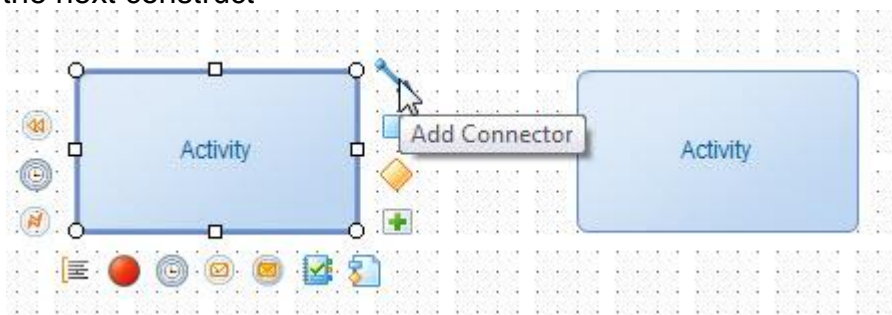
You can add constructs to your process using one of the following methods:

- Select a construct in the Toolbox then click in the modeling area.
- Drag and Drop from the Toolbox into the modeling area.

5.2 Adding Connector Lines

You can add connector lines using one of the following methods:

- Select a construct and use the connector line from intellisense and drag and drop to the next construct

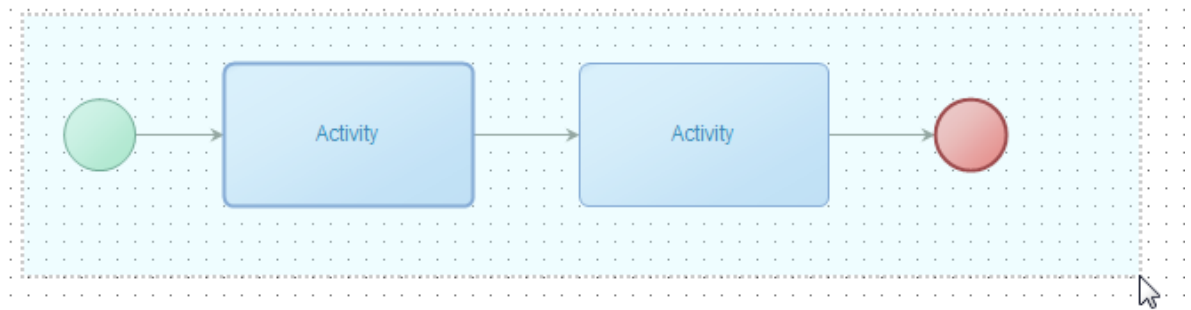


- Select the first construct and hold down the CTRL key while clicking the relevant next construct etc. Release the CTRL key to stop adding connectors.

5.3 Selecting Constructs

You can select multiple construct using one of the following methods:

- Keep left mouse button pressed and draw a dashed rectangle around all constructs you want to select:

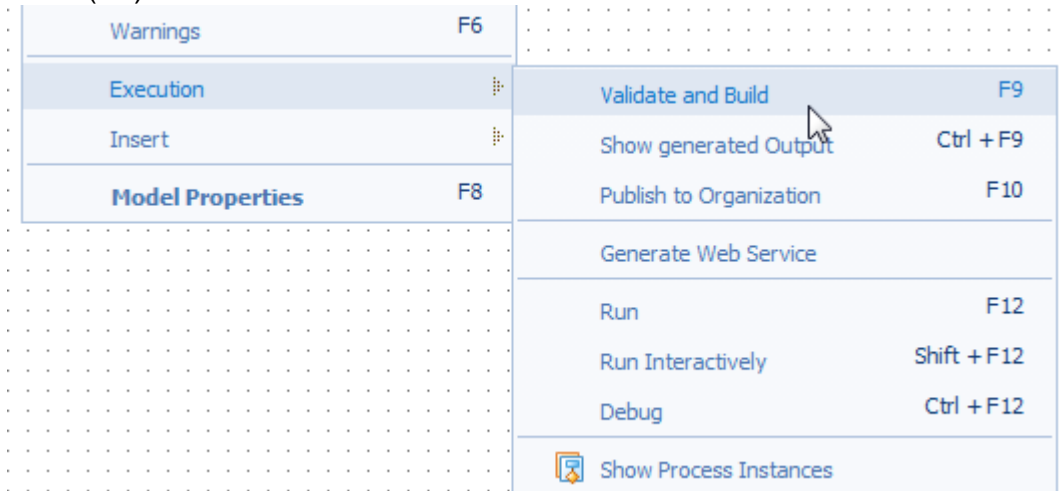




- Select the first construct and the subsequently SHIFT click the next construct
- Click in the process and use CTRL + A (select all constructs of the process).

5.4 Validating a Business Process Model


You can validate the process using one of the following methods:

- Right click the design area of the process and select **Execution** → **Validate and Build** (F9).



- Click the design area of the process, to get focus to the model editor and press **F9**.
- Click **Menu** () in the toolbar, Click **Validate** (), click [here](#) ([Click here to validate](#))
- In the Workspace Documents window, right click the process and select **Validate**.
- In the Workspace Documents window, right click the folder containing the process and select **Validate** (This will validate the content of the complete folder structure).
- In the Workspace Documents window, right click the project and select **Validate** (This will validate the whole project, only choose this method when explicitly instructed to do so).

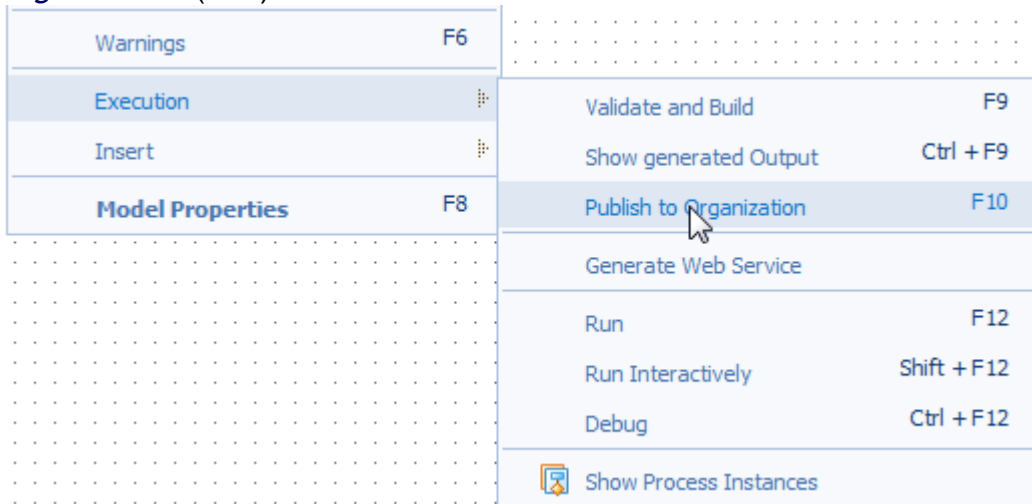
NOTE



You can (re)view the generated warning list at any time by clicking the **Show/Hide warnings** () button or pressing F6.

5.5 Publishing a Business Process Model

You can publish a process to an organization using one of the following methods:

- Right click the design area of the process and select **Execution** → **Publish to Organization** (F10).



- Click the design area of the process, to get focus to the model editor and press **F10**.
- Click **Menu** () in the toolbar, Click **Publish** ( Publish), click [here](#) (Click [here](#) to publish)
- In the Workspace Documents window, right click the process and select **Publish to Organization**.
- In the Workspace Documents window, right click the folder the process exists in and select **Publish to Organization** (This will publish the content of the complete folder structure).
- In the Workspace Documents window, right click the project and select **Publish to Organization** (This will publish the whole project, only choose this methods when explicitly instructed to do so).

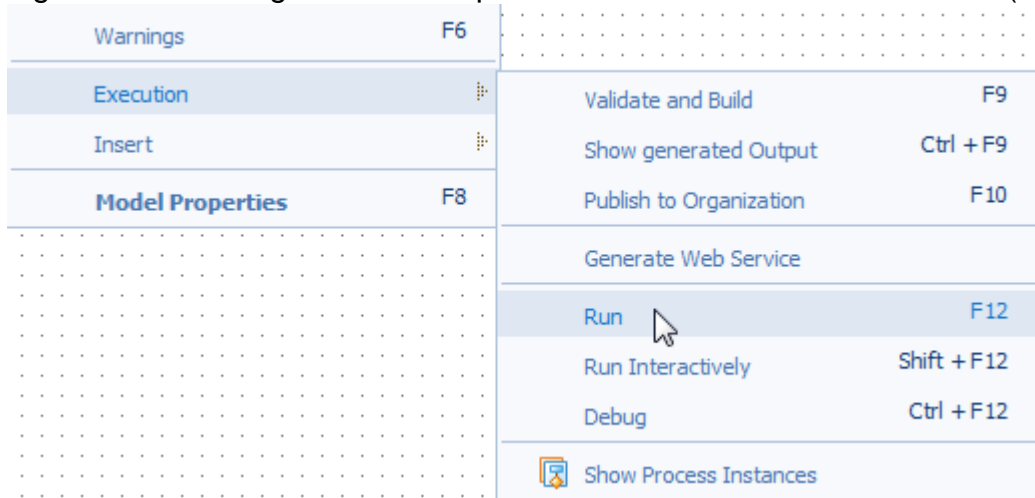
NOTE



Publishing your process will implicitly validate your process as well.

5.6 Running a Process

You can run a process from the design environment, using one of the following methods:

- Right click the design area of the process and select **Execution** → **Run** (F12)



- Click the design area of the process, to get focus to the model editor and press **F12**.
- Click **Menu** () in the toolbar, Click **Test** ().
- In the Workspace Documents windows, right click the process and select **Execution** → **Run**.

5.7 Monitoring Process Instances

You can view the various process instances from the design environment using one of the following methods:

- Right click the design area of the process and select **Execution** → **Show Process Instances**.
- In the Workspace Documents window, right click the process and select **Execution** → **Show Process Instances**.

As a (Process) Administrator, you can monitor all processes using the following artifacts from *My Applications* App palette:

- Process Instance Manager
- Deployed Process Models

6. Learning Report

Achievements

- ☐ I know how a process is designed
- ☐ I know how a process is developed
- ☐ I know how the message map works
- ☐ I can design and model a process
- ☐ I can develop and deploy a process
- ☐ I can monitor instances of started process
- ☐ I can debug a process

Notes