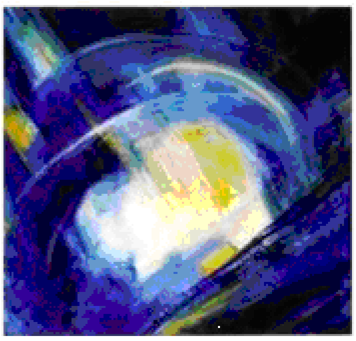
SemTalk 3.0

BPMN Edition

Tutorial



The main purpose of this tutorial is to learn how model business processes with SemTalk BPMN Edition. For this purpose, it is assumed that the reader is well acquainted with SemTalk. For more information about SemTalk and SemTalk Tutorials please visit our website URL: www.semtalk.com

SemTalk BPMN Edition is based mainly on BPMN version 1.0. For a comprehensive description of BPMN and its mapping to the underlying constructs of BPEL4WS refer to the document: BPMN version 1.0 – May 3, 2004 (available at www.bpmi.org)

Email: support@semtalk.com

1 Introduction

Business Process Management (BPM) is about how to describe business processes and their environment, and possibly how to adjust them when their environment changes. As a response to changing environments, the industry has expressed their desire to have adaptive business processes that can be refined and optimized causing that an increasing number of companies are using Web Services and the Service Oriented Architecture (SOA) for addressing the integration requirements for connecting applications.

Using SOA developers are capable of developing distributed systems. The components of these systems are services that perform a particular task. The modularity of these services permit that developers deal with “what” these services do and not “how” they perform a task. Because these services are usually distributed and can be located anywhere (within a local network or but also increasingly on the public Internet as well), it makes sense for large and medium enterprises to model business domains as related services. Instead of investing on technology developed solely for use in one department, these services can be deployed and reused across the enterprise or between enterprises. In other words, SOA is about “Composition” versus “Extension” of software applications.

The Business Process Modeling Notation (BPMN) specification provides a graphical notation for expressing business processes. This notation has been proposed by the Business Process Management Initiative (www.bpmi.org).

The objective of BPMN is to support business process management by both technical users and business users by providing a notation that is intuitive to business users yet able to represent complex process semantics. The main focus of BPMN is on systems that interact and interrupt one another, where there are many deeply nested independent, but coordinated, interacting threads of execution.

Before start adding process elements to a BPMN diagram it is convenient to define the three basic types of models within an end-to-end BPMN model. These are:

- Private (internal) business processes: business processes, which are internal to a specific organization.
- Abstract (public) processes: processes represent the communication between a private business process and another process or participant.
- Collaboration (global) processes: sequence of activities that represent the message exchange pattern between two or more business entities.

In this version of SemTalk BPMN Edition, only the first two types of models can be modeled. The third type, a collaboration process model, needs an extra subset of graphical notations, which are under discussion in the BPMI Notation Working Group. The first two types of process do not require separate graphical notations and thus, both can be modeled in the same BPMN Diagram type.

This BPMN Tutorial is based on use cases where the activities (Tasks) in the process are controlled by an entity (i.e. they are run by a process engine). The supported BPMN methodology in this SemTalk BPMN edition is designed to describe external and/or internal activities in business processes. On the other hand, collaboration activities are not controlled by a definitive entity. They are merely a description of the interactions between business entities.

Furthermore, this version of SemTalk BPMN Edition will proved more useful to those users that want to build a BPMN model and next deploy it in an orchestration engine like Microsoft BizTalk Server 2004. With this aim, the following are some prerequisites:

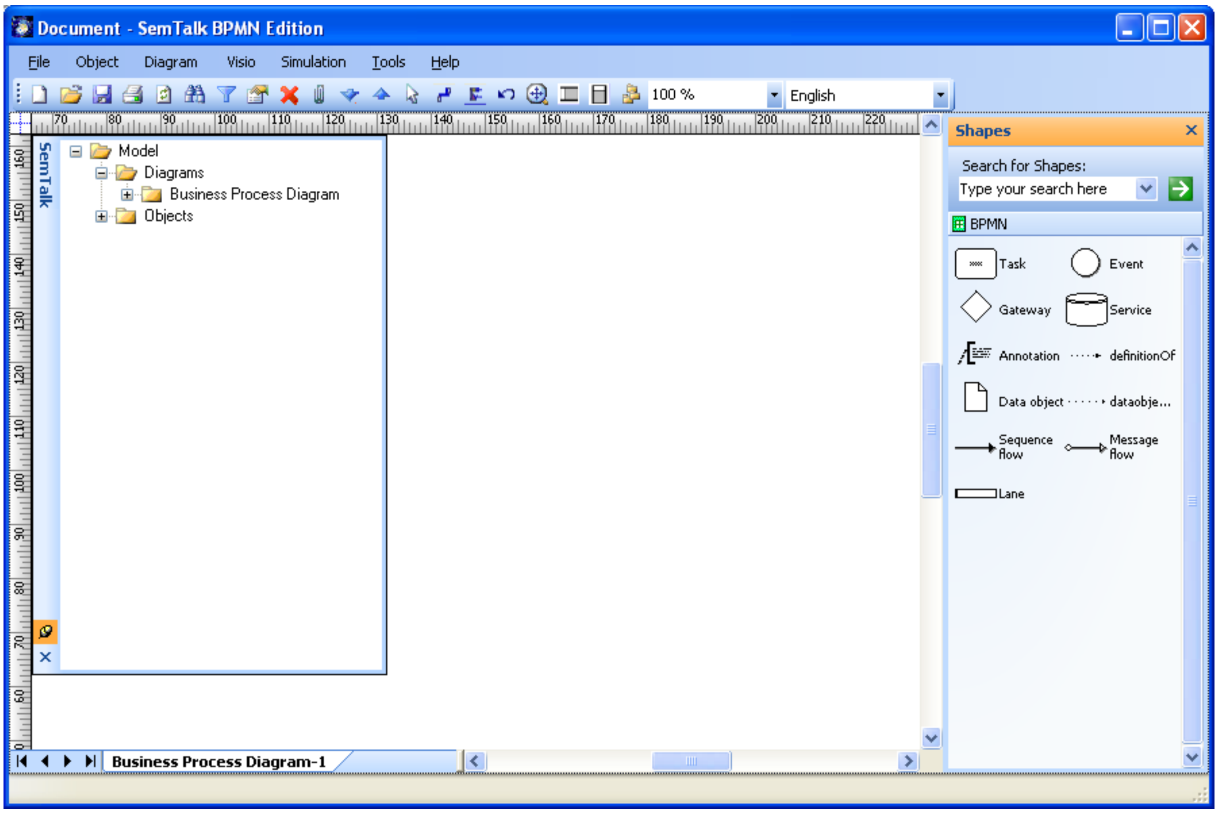
- SemTalk installed in your system
- Internet Explorer 5.5 or later
- Some familiarity with XML Schema, WSDL, XPath, BPEL, and related Web service standards
- wsdl example files found at: www.semtalk.com/download/TravelService.zip

You may want to watch a video clip showing SemTalk BPMN from our website: http://www.semtalk.com/test/bpmn.wmv

2 Start SemTalk BPMN Edition

Start SemTalk with it's icon on the desktop or select Start → Programs → SemTalk → SemTalk3 and begin to work. (In case you did not open a BPMN diagram before, please open a new diagram by selecting File → New and choose the BPMN template (bpmn.vst)).

You should see the following screen:

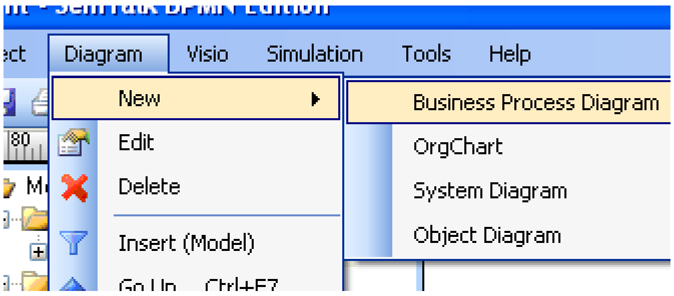


The example in this tutorial is a simple version of a travel booking service.

3 Creating a process

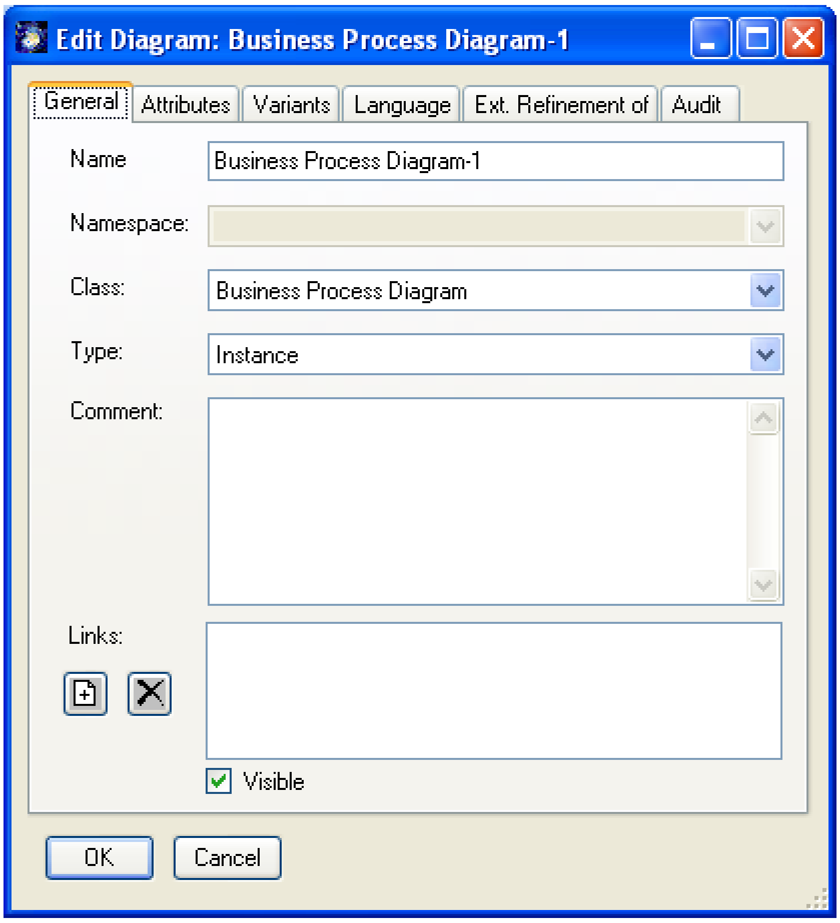
The easiest way to create a new process is to right-click on “Business Process Diagram” in the Explorer and to select “New”.

You may also select under the menu option: Diagram → New → Business Process Diagram.



This will create a new diagram of the class “Business Process Diagram”. You can rename that diagram directly by renaming the diagram tab or a using a SemTalk dialog (Right Click→ Edit) on the background or the menu item “Diagram → Edit”

The diagram dialog shows up as following:


Edit Diagram: Business Process Diagram-1

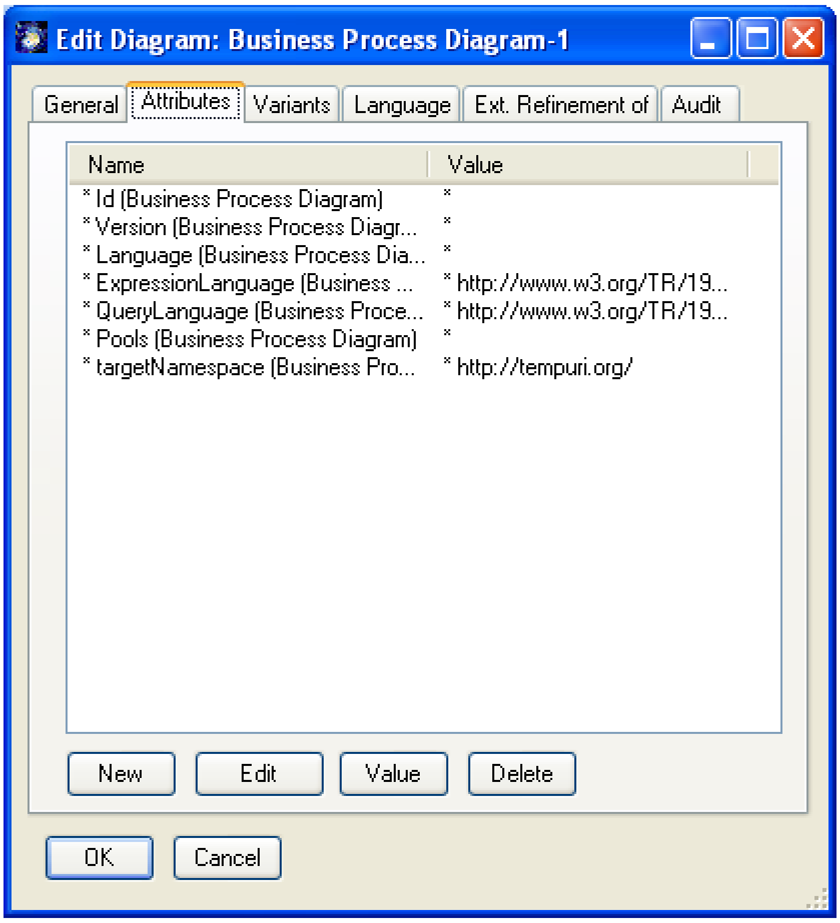
General | Attributes | Variants | Language | Ext. Refinement of | Audit

Name	Business Process Diagram-1
Namespace:	
Class:	Business Process Diagram
Type:	Instance
Comment:	
Links:	

Visible [✓]

OK | Cancel

In the BPMN Diagram “Attributes” tab you can specify important attributes for a BPMN - BPEL mapping of the process.



Name	Value
* Id (Business Process Diagram)	*
* Version (Business Process Diagr...	*
* Language (Business Process Dia...	*
* ExpressionLanguage (Business ...	* http://www.w3.org/TR/19...
* QueryLanguage (Business Proce...	* http://www.w3.org/TR/19...
* Pools (Business Process Diagram)	*
* targetNamespace (Business Pro...	* http://tempuri.org/

The figure above shows different top-level attributes of a BPMN model. Wsdl and variable attribute are set automatically by SemTalk. More important are the attributes: ExpressionLanguage. QueryLanguage, supressJoinFailure and enableInstance. The attributes are determined by the process engine where the BPEL code (generated from the BPMN model) will be implemented.

A brief explanation of top-level attributes is as follows:

- QueryLanguage. This attribute specifies the XML query language used for selection of nodes in assignment, property definition, and other uses. The default for this attribute is XPath 1.0, represented by the URI of the XPath 1.0 specification: http://www.w3.org/TR/1999/REC-xpath-19991116.
- ExpressionLanguage. This attribute specifies the expression language used in the process. The default for this attribute is XPath 1.0, represented by the URI of the XPath specification: http://www.w3.org/TR/1999/REC-xpath-19991116.

- suppressJoinFailure. This attribute determines whether the joinFailure fault will be suppressed for all activities in the process. The effect of the attribute at the process level can be overridden by an activity using a different value for the attribute. The default for this attribute is "false".
- enableInstanceCompensation. This attribute determines whether the process instance as a whole can be compensated by platform-specific means. The default for this attribute is "false".

Please refer to the BPMN v1.0 for more information.

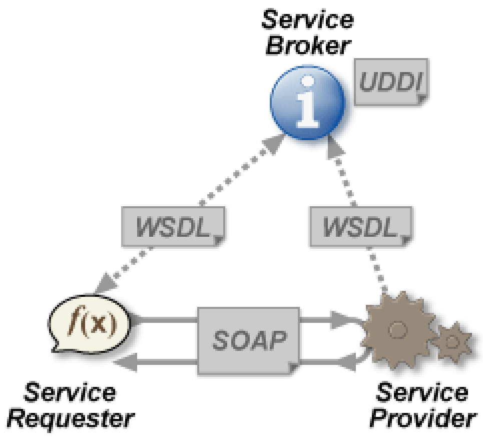
The “External Refinement Of” and “Language” tabs are discussed in SemTalk General Tutorial.

4 Adding Process Elements

You may add now elements to your new process: Simply drag them from the stencil to your process pane.

4.1 Adding Process Participants

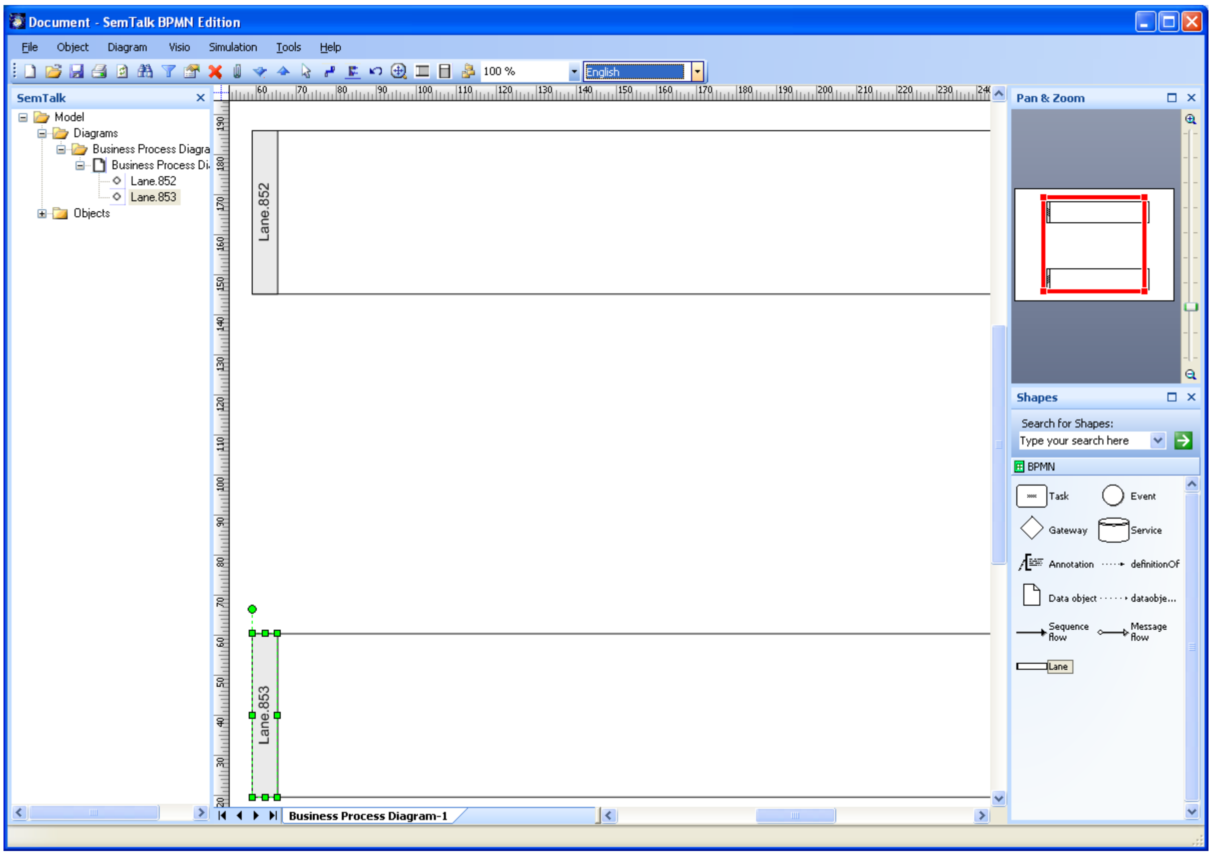
In our example Task's properties are mainly defined by the Participants (Service Requester(s), Service Provider(s)) of the Travel Booking Service. Thus, we should first define the set of Participants and assign them a corresponding Web service.



NOTE: According to the W3C a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface that is described in a machine-processable format such as WSDL. Other systems interact with the Web service in a manner prescribed by its interface using messages, which may be enclosed in a SOAP envelope. These messages are typically conveyed using HTTP, and normally comprise XML in conjunction with other Web-related standards.

NOTE: Before continuing make sure that you have on hand the WSDL files for this tutorial.

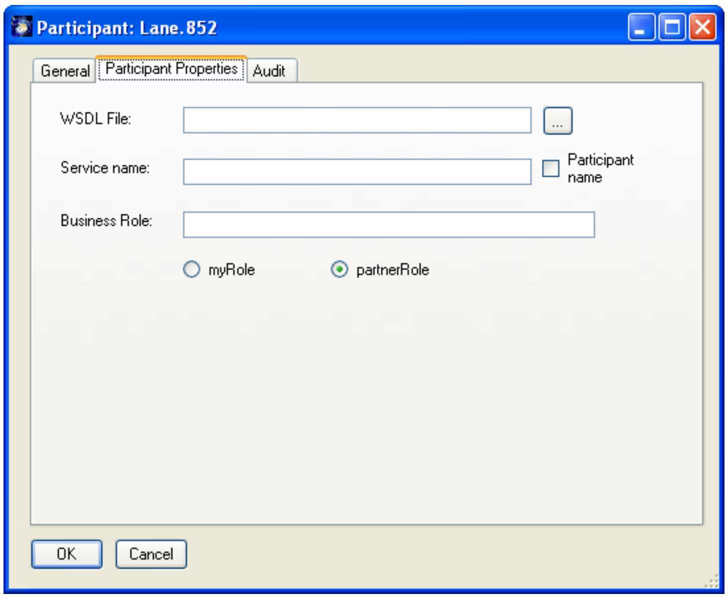
A process Participant is represented by a lane. Please drag 2 lane shapes from the stencil and drop them on the model.



Now you will assign the corresponding WSDL files to each Participant. One of the Participants will be a credit card check service and the other a hotel booking service.

To edit the Participant properties (or any other process object) you can do the following:

1. Double click or
2. Right Click "Edit" or
3. Select & Menu "Object → Edit" or
4. Find it in the explorer as a child of "Diagrams→ BPMN Diagram→ Business Process Diagram-1" or
5. Find it in the explorer as a child of "Objects →".


Participant: Lane.852

General | Participant Properties | Audit

WSDL File:
Service name: [ ] Participant name
Business Role:
myRole [✓ partnerRole]

OK | Cancel

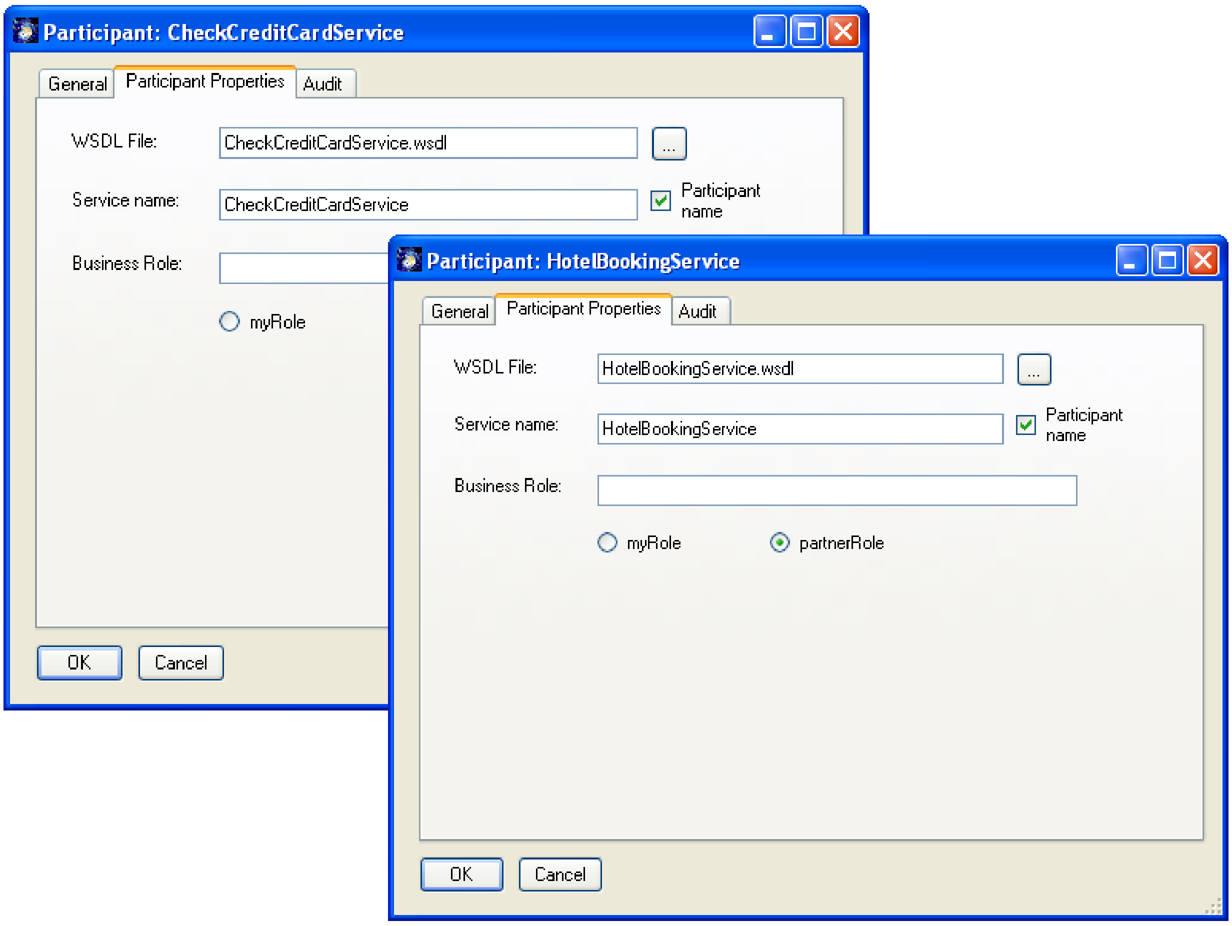
Please click on the Tab named "Participant Properties"

To enter the location of the WSDL file click the button "…" and browse until you find file named "CheckCreditCardService.wsdl".

If you check the checkbox "Participant name" the lane will be renamed with the service name provided in the WSDL file after pressing OK. Please note that the WSDL file may not contain this name and still be valid.

The options "myRole" and "partnerRole" indicate what role this Participant plays in the process. The "myRole" indicates that the Participant is the client and "partnerRole" which is the case of this service. In the textbox "Business Role" you may want to specify the role with another identifier (e.g. "CreditCardCheckRole").

Next repeat this procedure with the other Participant and assign to it the WSDL file for the hotel booking service (HotelReservationService.wsdl).


Participant: CheckCreditCardService

WSDL File: CheckCreditCardService.wsdl
Service name: CheckCreditCardService
[✓ Participant name]
Business Role:
myRole

Participant: HotelBookingService

WSDL File: HotelBookingService.wsdl
Service name: HotelBookingService
[✓ Participant name]
Business Role:
myRole [✓ partnerRole]

Now that the Participants are ready, you can continue with the other process elements.

4.2 Adding Flow Objects

In BPMN a Process is depicted as a network of Flow Objects, which are a set of other activities and the controls that sequence them. There are 3 classes of flow objects: activities (Tasks), Events and Gateways. These flow objects are explained further in detail during the course of this tutorial.

The storyline of this travel booking example says that its user will be able to use web services to book a room in a hotel using his/her credit card. The system should check if the credit card is valid and if the hotel reservation was done. If every is ok a confirmation e-mail will be sent to the user. If there is a problem with the credit card, the user will be notified.

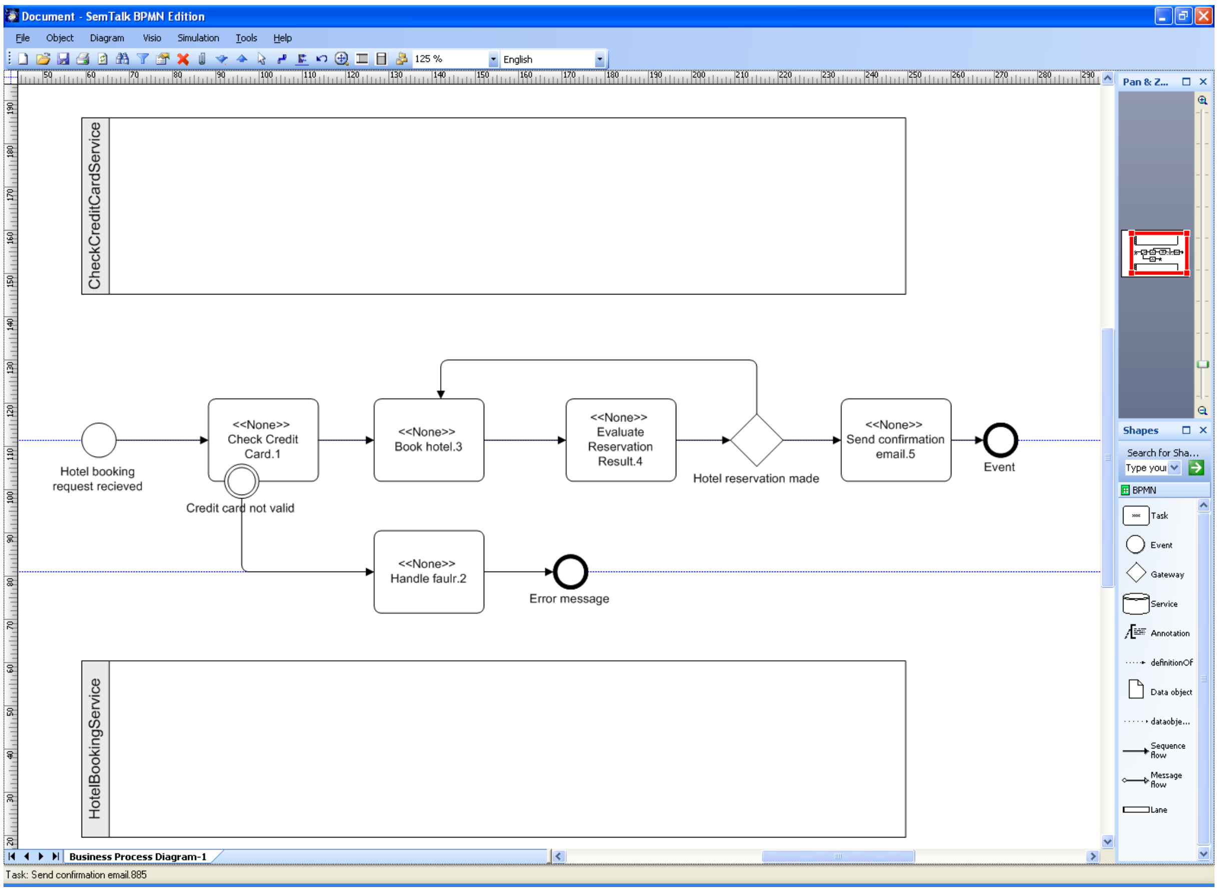
First of all, it is clear that at least 3 Tasks are needed: “check credit card”, “book hotel room” and “send confirmation e-mail”. Additionally a Task will be needed to evaluate if the hotel reservation was made and a Task to evaluate the an error in the process, if any occurs. So you need 5 Tasks in total.

You may write the name of these Tasks just by writing over the shape or you may want to follow an object oriented method named “Compose”.

Furthermore, you need at least two Events, one trigger Event and one end Event. The process should be triggered when a hotel booking request is received and finished when the confirmation has been sent. But because there is the possibility of an process error or fault an extra Task to handle the fault and a third Event are needed.

Last but not least, the system should check first if the credit card is valid and, like mentioned before, if the hotel reservation was made. If the system failed to validate the credit card, then the Task should be interrupted. Therefore an Intermediate Event is attached to the Task. If the reservation was not made, the system should try again. This means that at least 1 Gateway is needed to check and route the control flow after the evaluation of the information.

After connecting the flow objects with a sequence flow the process should like this: (see section 6.1 for more information on how to connect flow objects using sequence flows)



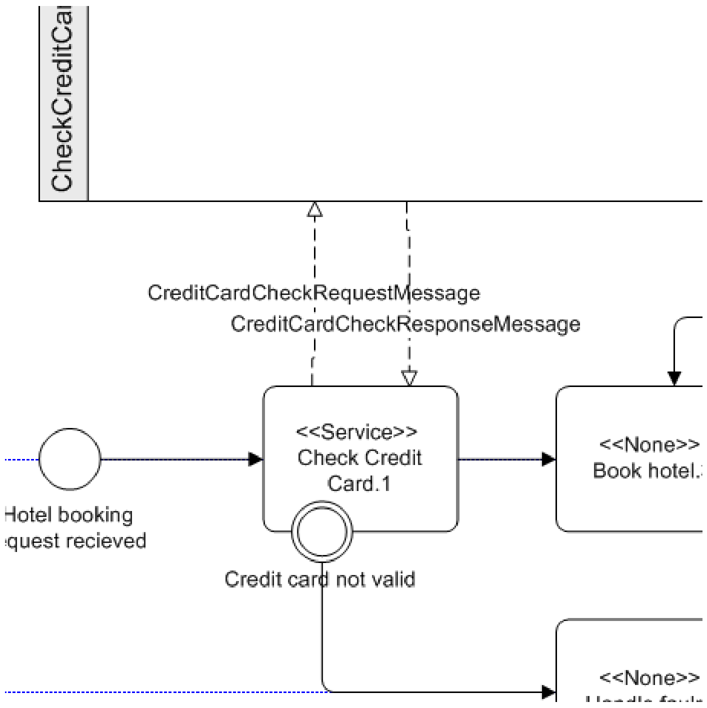
4.3 *Defining Flow Objects Properties*

4.3.1 *Tasks*

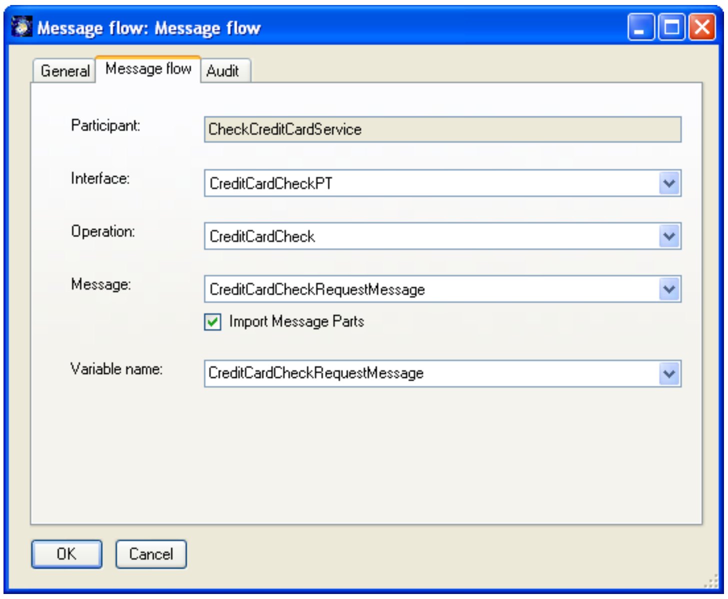
Let us begin by defining the Tasks properties.

As mentioned before, the Tasks properties are greatly influenced by the Participants Web Services. The reason for this has to do with the purpose of our model. Our model describes the process of orchestrating the Web Services of 2 Participants. The sequence flows in the process determine the order of the activities that will coordinate the Web Services. Thus, the activities must communicate with the Web Services and pass the data that the Web Services will consume. This is done using message flows, which flow orthogonally to the sequence flow. See section 6.2 for more information on message flow rules.

The first Task “Check credit card” communicates with the “CheckCreditCardService”. The activity will request the service to validate the credit card and receive the confirmation of the validation. Thus, we need messages flows flowing in both directions. Your model should look like this:


CheckCreditCa
CreditCardCheckRequestMessage
CreditCardCheckResponseMessage
<<Service>> Check Credit Card.1
<<None>> Book hotel.
Hotel booking quest recieved
Credit card not valid
<<None>>

As you notice the task type (<<task type>>) was set automatically to <<Service>>. For more information on Task properties and description of each supported task type see section 5.2.



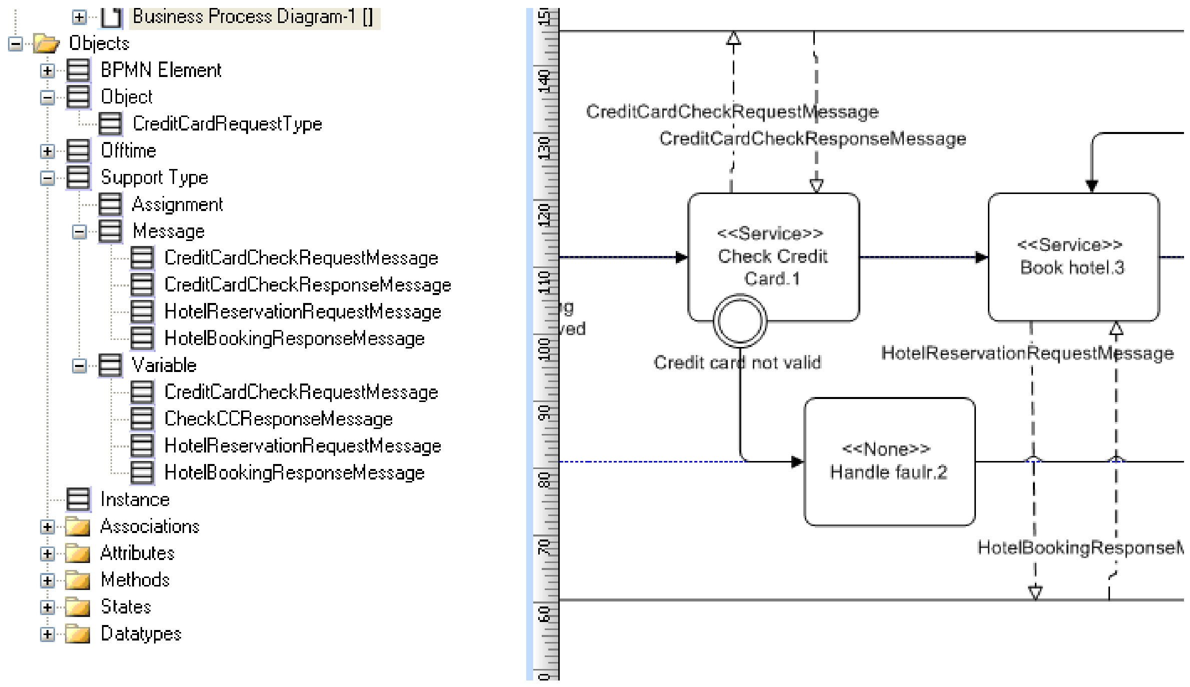
Participant: CheckCreditCardService
Interface: CreditCardCheckPT
Operation: CreditCardCheck
Message: CreditCardCheckRequestMessage
☑ Import Message Parts
Variable name: CreditCardCheckRequestMessage

Please open the edit dialog box for the message flow from Task to Participant. This message flow will carry the request message. As you can see in the dialog box the interface, operation and message has been already set.

Additionally, you can decide to import the message parts as classes. This is especially interesting if you want to examine the structure of the message being sent.

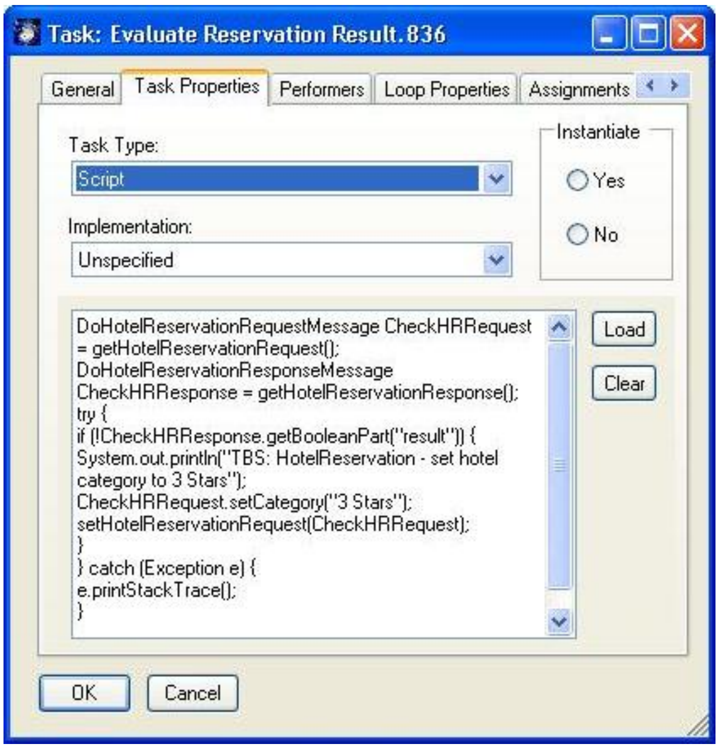
Furthermore, a variable name should be entered. Variables are necessary for the BPEL export. Variables provide the means for holding messages that constitute the state of a business process. The messages held are often those that have been received from partners or are to be sent to partners. Variables can also hold data that are needed for holding state related to the process and never exchanged with partners.

After pressing OK and repeating the same procedure with the response message flow (enter “CheckCCResponse” as variable name), you should see the following:


Business Process Diagram-1 []
Objects
BPMN Element
Object
CreditCardRequestType
Offtime
Support Type
Assignment
Message
CreditCardCheckRequestMessage
CreditCardCheckResponseMessage
HotelReservationRequestMessage
HotelBookingResponseMessage
Variable
CreditCardCheckRequestMessage
CheckCCResponseMessage
HotelReservationRequestMessage
HotelBookingResponseMessage
Instance
Associations
Attributes
Methods
States
Datatypes
CreditCardCheckRequestMessage
CreditCardCheckResponseMessage
<<Service>> Check Credit Card.1
<<Service>> Book hotel.3
Credit card not valid
HotelReservationRequestMessage
<<None>> Handle faulr.2
HotelBookingResponseM

Please repeat this procedure with Task „Book hotel”, which is also a service Task.

The next Task “Evaluate Reservation Result” is a Task different from the previous Tasks. When this Task is reached no service will be called, but instead the engine that is executing the process will run a script. The script will check that the reservation has been successful.



Task Type: Script

Implementation: Unspecified

Instantiate: Yes No

```
DoHotelReservationRequestMessage CheckHRRequest
= getHotelReservationRequest();
DoHotelReservationResponseMessage
CheckHRResponse = getHotelReservationResponse();
try {
if (!CheckHRResponse.getBooleanPart("result")) {
System.out.println("TBS: HotelReservation - set hotel
category to 3 Stars");
CheckHRRequest.setCategory("3 Stars");
setHotelReservationRequest(CheckHRRequest);
}
} catch (Exception e) {
e.printStackTrace();
}
```

To define the properties of this Task, open the edit dialog box and select “Script” as task type in the “Task Properties” tab. To browse your files for a script file press the “Load” button and choose the proper file. The script text will be displayed in the textbox. You may also write a script directly in SemTalk.

The “Send Confirmation” and “Handle fault” Tasks work analogously. The engine executing the process will perform the script when the Task is reached after a positive evaluation or when an error occurs, respectively.

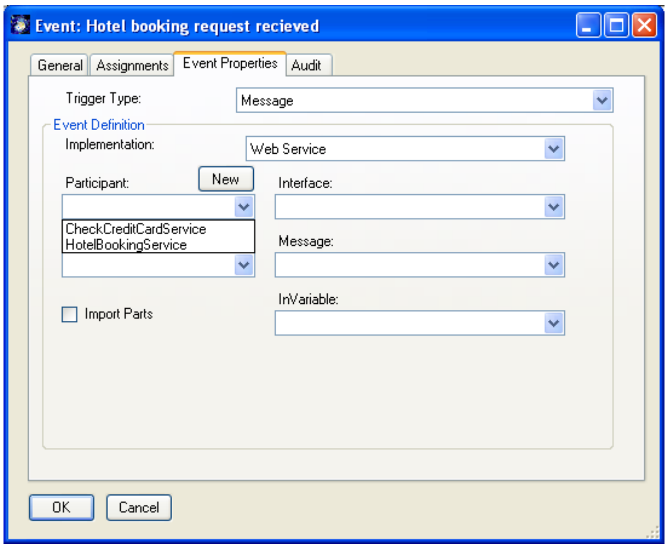
4.3.2 Events

Let us continue with the Events. An Event is something that “happens” during the course of a business process. These Events affect the flow of the Process and usually have a cause or an impact. BPMN has restricted the use of Events to include only those types of Events that will affect the sequence or timing of activities of a process.

In our example, the three types of Events are represented: Start, Intermediate and End Event. The Event type if set automatically by SemTalk depending of the Event’s position in the process sequence.

Like in the case of Tasks, Events properties depend on the Web Services of the process Participants. Events receive or send messages from and to the Participants respectively. See section 6.2 for more information on message flow rules.

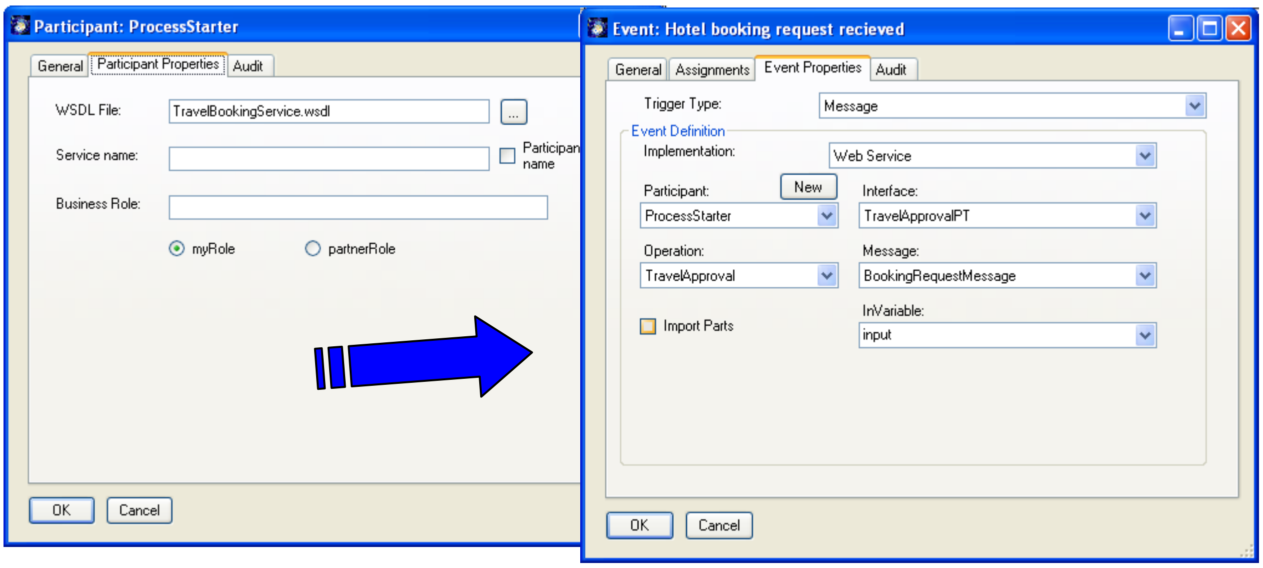
The first Event “Hotel booking request received” is a Start Event. This Event is triggered when an specific message from a Participant is received. The message will contain the information that the other Participants will consume to run their activities. In our example the Participant will be the customer and he/she will send his/her card number, card type and the necessary information to book a hotel (hotel name, hotel category, city, dates (from, to)). The first two are not mandatory, but the rest is. The corresponding WSDL file contains this information. (TravelBookingServiceInterface.wsdl)



Trigger Type: Message
Implementation: Web Service
Participant:
CheckCreditCardService
HotelBookingService
Interface:
Message:
InVariable:

In SemTalk it is not necessary to draw the Participant on the model to be able to define the Participant and its properties. Please open the edit dialog box of the Start Event and click on the “Event Properties” tab. Select “Message” in the “Trigger type” drop-down list and “Web Service” in the Implementation drop-down list. You should see the dialog box to the right.

As you may notice, we must create a new Participant. Click on the "New" button and enter a name for the new participant (e.g. "ProcessStarter"). After clicking "OK" you should see the Participant dialog box. Select the appropriate WSDL file and the option "myRole". Click "OK".


Participant: ProcessStarter

General | Participant Properties | Audit

WSDL File: TravelBookingService.wsdl

Service name:

Participant name

Business Role:

[✓ myRole] partnerRole

OK | Cancel

Event: Hotel booking request recieved

General | Assignments | Event Properties | Audit

Trigger Type: Message

Event Definition

Implementation: Web Service

Participant: ProcessStarter (New)

Interface: TravelApprovalPT

Operation: TravelApproval

Message: BookingRequestMessage

Import Parts

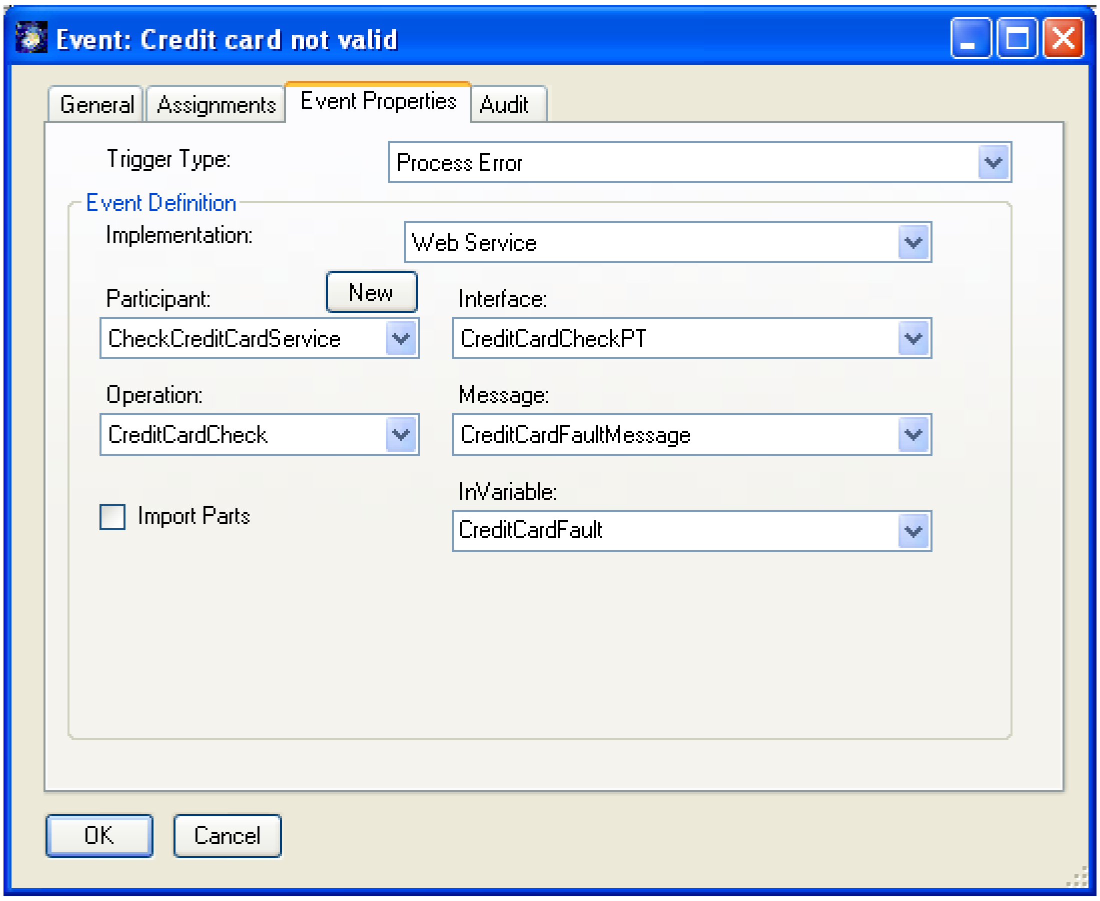
InVariable: input

OK | Cancel

Make sure you select the operation "TravelApproval" and the message "BookingRequestMessage". Please enter as variable name "input".

The End Events "Confirmation email sent" and "Error message" behave analogously. For both of them the same operation "TravelApproval" and the message "BookingRequestMessage" are valid. For both of them you may use the variable "output".

Finally, the Intermediate Event "Credit card not valid" is a "Process Error" Event. This Event will "catch" an exception in case that the credit card is not valid. Open the edit dialog box of this event and set the "Event type" to "Process Error" and "Implementation" to "Web Service". Select the operation "CheckCreditCard" and the message "Exception". Change the variable name to "CreditCardFault".



Trigger Type: Process Error

Event Definition

Implementation: Web Service

Participant: CheckCreditCardService

Interface: CreditCardCheckPT

Operation: CreditCardCheck

Message: CreditCardFaultMessage

Import Parts

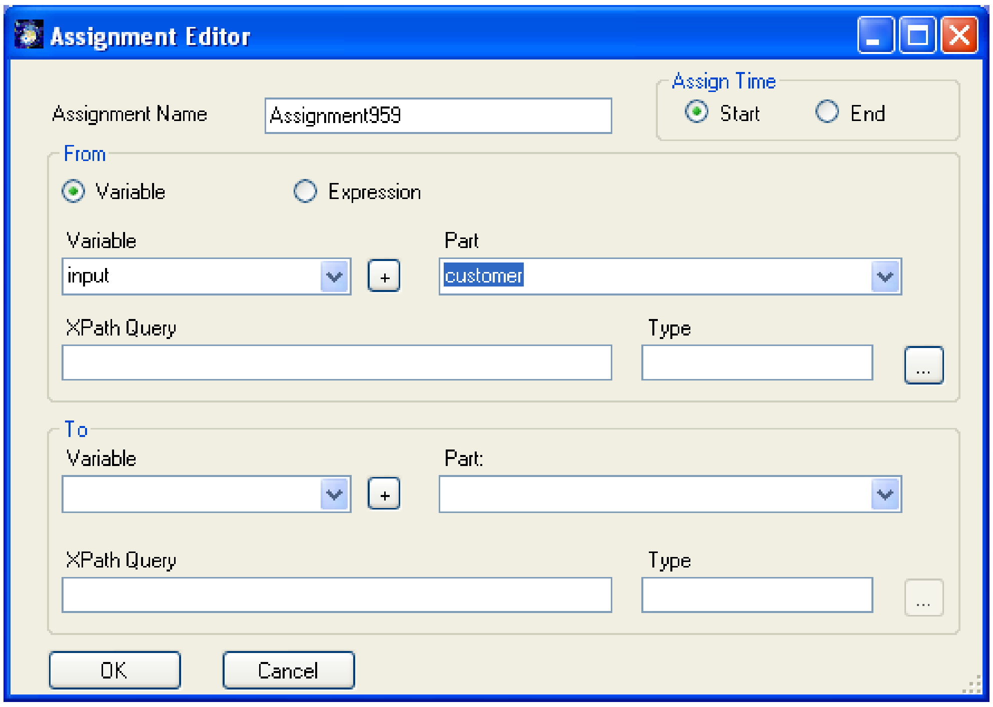
InVariable: CreditCardFault

The following table summarizes the different variables, their corresponding messages and where the message definitions are located.

Variable name	**File containing the message definition**	**Message name**
input	TravelBookingServiceInterface.wsdl	bookRequest
output	TravelBookingServiceInterface.wsdl	bookResponse
CreditCardFault	CheckCreditCardService.wsdl	Exception
CheckCCRequest	CheckCreditCardService.wsdl	CheckCreditCardRequest
CheckCCResponse	CheckCreditCardService.wsdl	CheckCreditCardResponse
HotelReservationRequest	hotelReservationServiceImpl.wsdl	HotelReservationRequest
HotelReservationResponse	hotelReservationServiceImpl.wsdl	HotelReservationResponse

4.3.3 Assignments

Before a Participant's operation is invoked, the data that is to be sent to the Participant has to be copied to a variable that holds this data. Assign activities fulfill this task. In the next steps, you will define the Assignments in the Tasks and Events you added to your process earlier.


Assignment Editor

Assignment Name: Assignment959

Assign Time: [✓ Start] End

From: [✓ Variable] Expression

Variable: input

Part: customer

XPath Query:

Type:

To

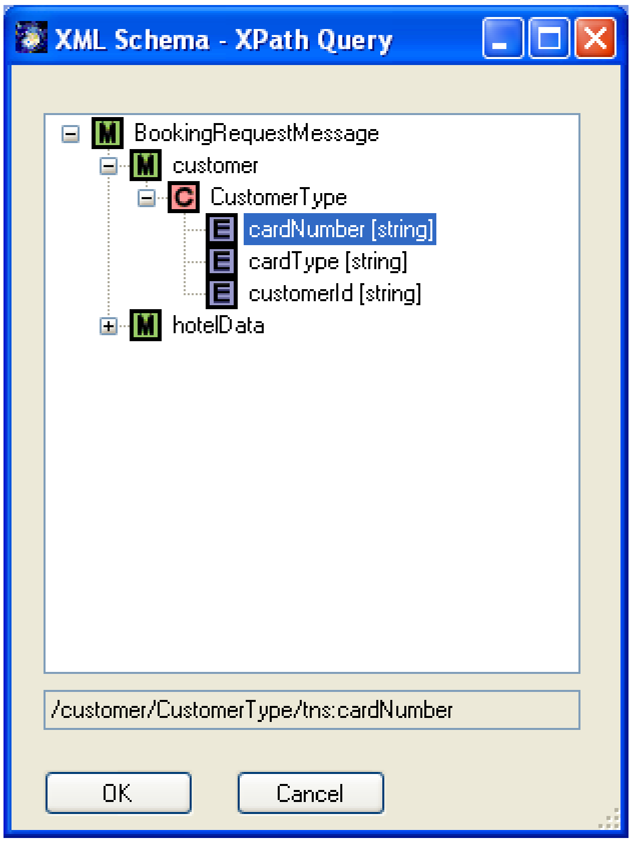
Variable:

Part:

XPath Query:

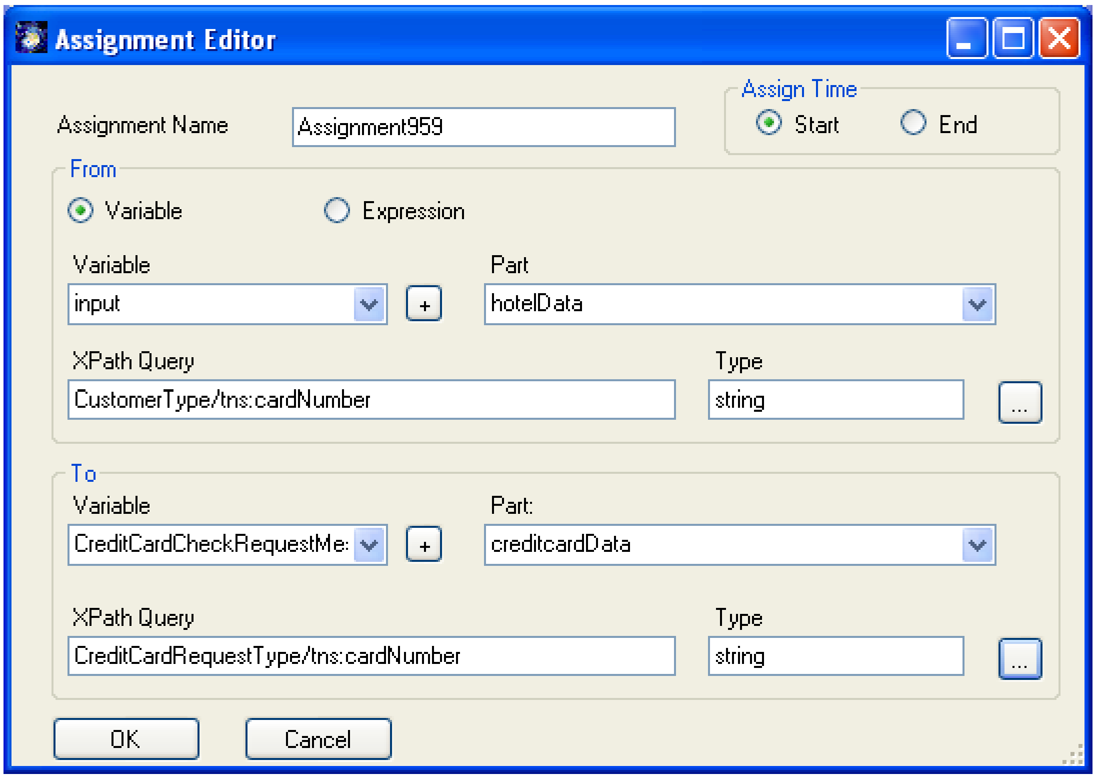
Type:

Open the dialog box of the Task “Check credit card” and go to the “Assignments” tab. Select from the From “Variable” drop-down list “input”. As you can see the message part “customer” and its type are immediately listed in the control boxes. Please notice that the part type is not a XML schema type. That means that probably the part has a complex type. Click on the “…” button to see a tree view of the XML schema.



/customer/CustomerType/tns:cardNumber

As you can see, the part “customer” has more than one attribute. To map these attributes select one at a time, e.g. “cardNumber” and click “OK”. Do the same in the To section, but select now the variable “CheckCCRequest”. Your “Assignments” tab should look similar to the figure below.


Assignment Editor

Assignment Name: Assignment959

Assign Time: [✓ Start] End

From: [✓ Variable] Expression

Variable: input

Part: hotelData

XPath Query: CustomerType/tns:cardNumber

Type: string

To

Variable: CreditCardCheckRequestMe:

Part: creditcardData

XPath Query: CreditCardRequestType/tns:cardNumber

Type: string

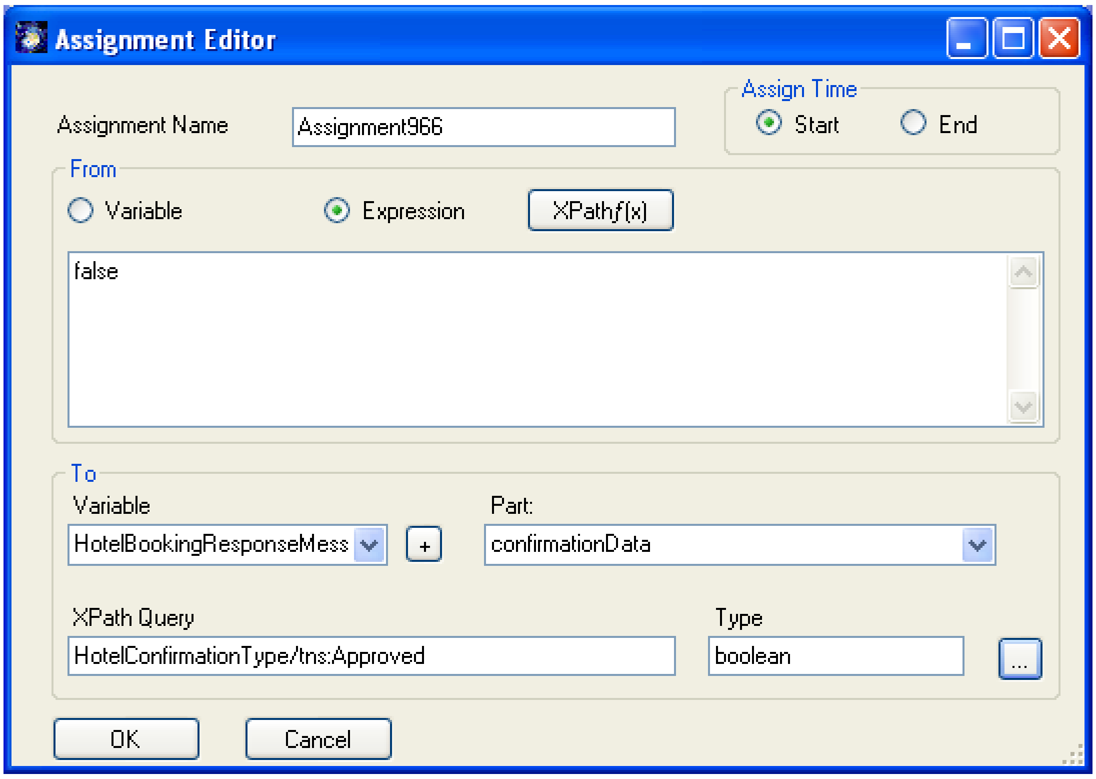
Now, the “cardNumber” part of the input variable has been mapped to the “cardNumber” part of the CheckCCRequest variable. Map the “cardType” parts in the same way now.

You can edit an Assignment by selecting it in the “Assignment” box and clicking on the “New” button. You can delete an Assignment by first selecting it and the clicking on the “Delete“ button.

Complete the assignments for the "Book hotel" using the values provided in the table below:

"from" Variable	"from" Part	"to" Variable	"to" Part
input	hotelCompany	HotelReservationRequest	name
input	target	HotelReservationRequest	city
input	hotelCategory	HotelReservationRequest	category
input	from	HotelReservationRequest	from
input	to	HotelReservationRequest	to
input	cardNumber	HotelReservationRequest	cardNumber
input	cardType	HotelReservationRequest	cardType

Finally map the "false" literal of type xsd:boolean to the result part of the HotelReservationResponse variable. To do this select the option "Expression" in the From section and write "false". In the To section select the "HotelReservationResponse" variable and the part "HotelReservationResponse". Add the Assignment.


Assignment Editor

Assignment Name: Assignment966

Assign Time: [✓ Start] End

From: Variable [✓ Expression]

false

To

Variable: HotelBookingResponseMess

Part: confirmationData

XPath Query: HotelConfirmationType/tns:Approved

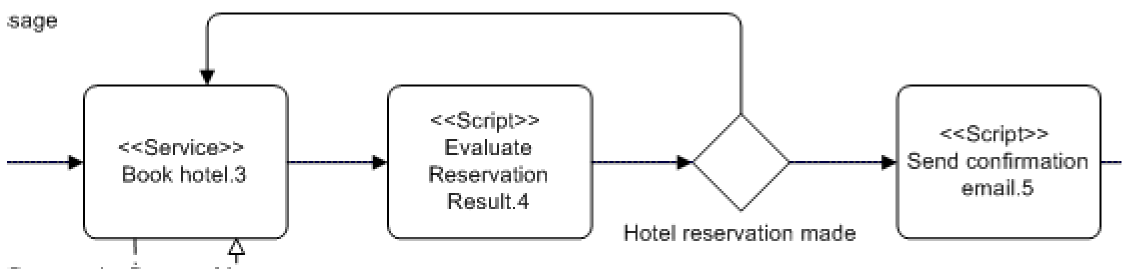
Type: boolean

The property "Assign Time" indicates when the assign element will be carried out. "Start" means that the assign element will proceed the invoke.

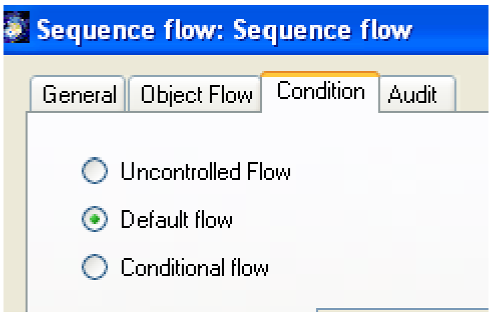
4.3.4 Sequence flow

A Sequence flow is used to show the order those activities will be performed in a Process. To learn more on Sequence flow rules go to section 6.1.

There are 3 types of Sequence flows: Uncontrolled flow, conditional flow and default flow (See section 7 page 55). The Gateway "Hotel reservation made" is a good example to illustrate the 3 cases. As it was mentioned before, the reservation result must be evaluated to continue with its confirmation or try again. To define this open the dialog box of the Sequence flow going from the Gateway to the Script Task.


sage
<<Service>> Book hotel.3
<<Script>> Evaluate Reservation Result.4
Hotel reservation made
<<Script>> Send confirmation email.5

This Sequence flow will be the default flow of the decision point, i.e. this flow will be used only if all the other outgoing conditional flows are not true at runtime.



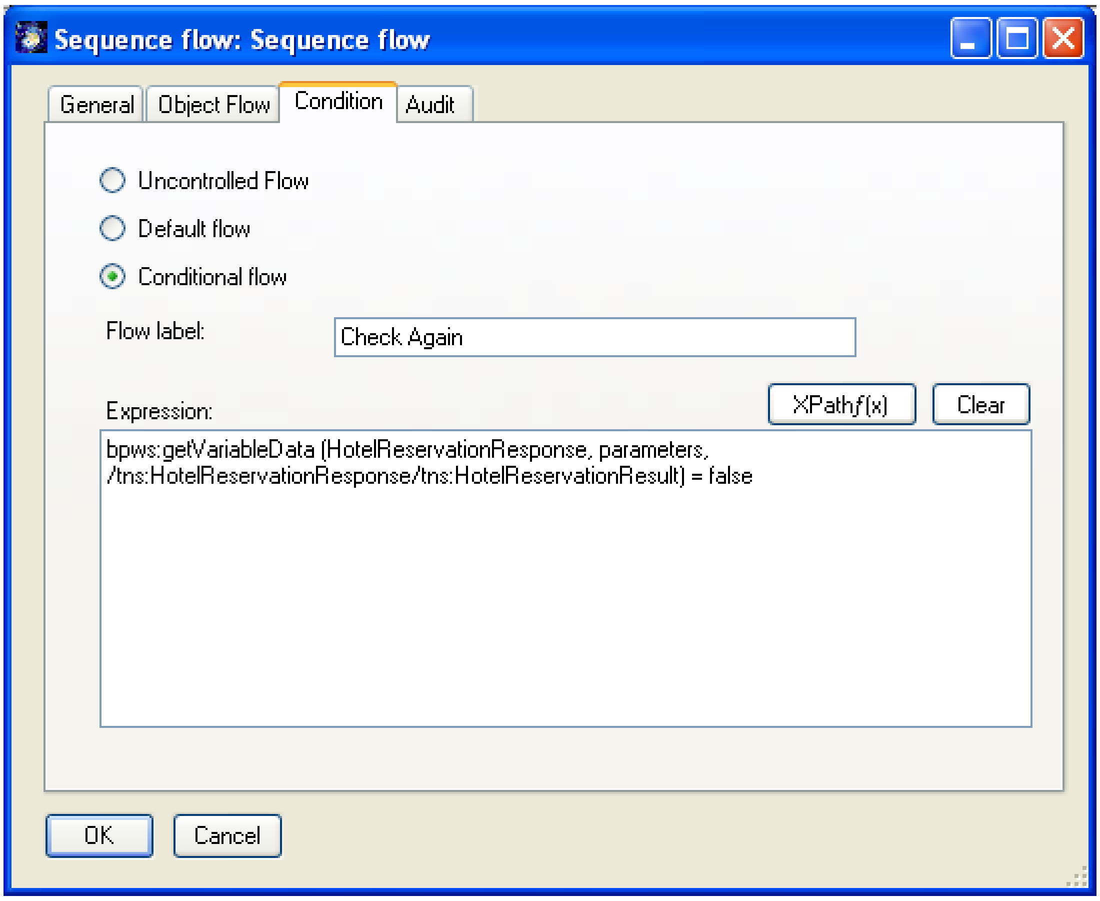
Uncontrolled Flow [✓ Default flow] Conditional flow

Open the dialog box of the sequence flow going from the Gateway back to the Task "Book hotel". In the "Sequence flow Properties" tab select the "Condition Flow" option and write the following BPEL expression:

```
bpws:getVariableData (HotelReservationResponse, parameters,
/tns:HotelReservationResponse/tns:HotelReservationResult) = false
```

The process engine evaluates this expression at runtime to select the next activity is the process sequence. In other words, the HotelReservationResult is "false" then the process engine will try to book again the reservation.

If you check “Free Text” you may write a general label for the Sequence flow. If not the expression will be used a label. Write “Check Again” as label.



Sequence flow: Sequence flow

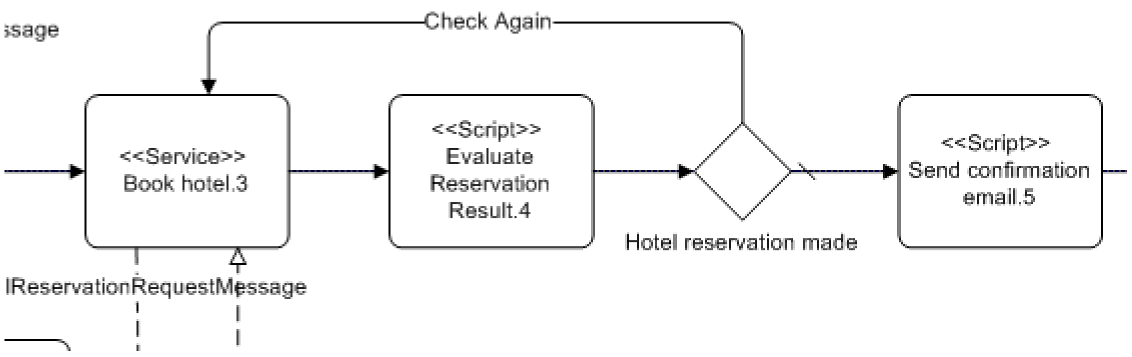
General | Object Flow | Condition | Audit

( ) Uncontrolled Flow
( ) Default flow
[✓ Conditional flow]

Flow label: Check Again

Expression:

```
bpws:getVariableData (HotelReservationResponse, parameters,
/tns:HotelReservationResponse/tns:HotelReservationResult) = false
```


ssage
Check Again
<<Service>> Book hotel.3
<<Script>> Evaluate Reservation Result.4
<<Script>> Send confirmation email.5
Hotel reservation made
lReservationRequestMessage

4.3.5 Gateways

For more information on Gateways go to section 5.6

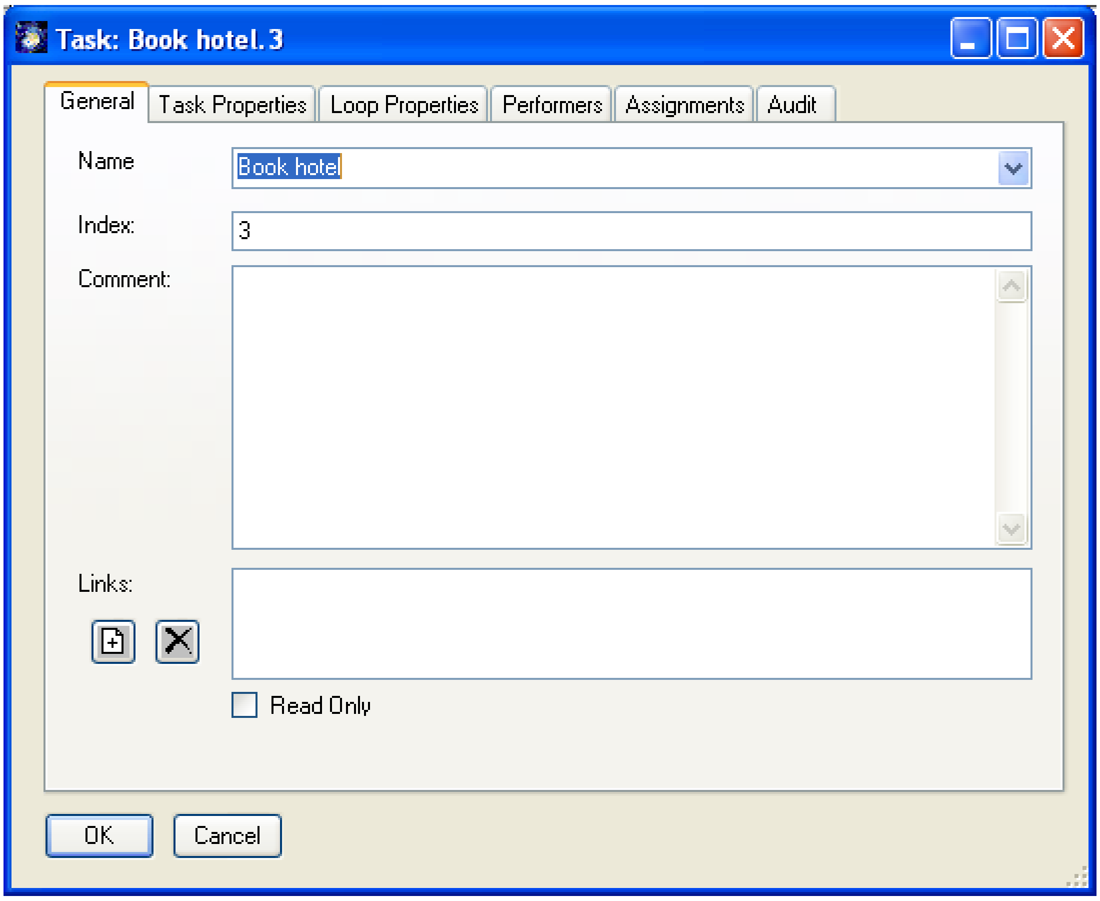
5 General Functionalities

5.1 Task Dialog

You can edit any object

6. By double clicking or
7. Right Click “Edit” or
8. Select & Menu “Object → Edit” or
9. Find it in the explorer as a child of “Diagrams→ BPMN Diagram→ Business Process Diagram-1” or
10. Find it in the explorer as a child of “Objects →”.

The Task dialog looks like this:


Task: Book hotel.3

Name: Book hotel

Index: 3

Comment:

Links:

Note on the BPMN methodology:

We are using the term Task to refer to a step in a process. Each Task is an instance of its *Task* class (we will distinguish the Task instance from the *Task* class with italics). E.g. “Check Hotel Reservation” is the *Task* class and Hotel Reservation Check.785 a Task instance of Check Hotel Reservation in a process.

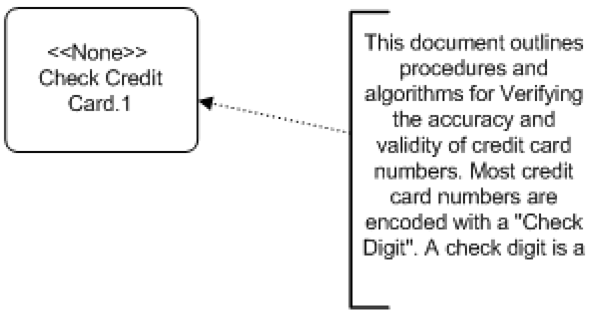
You can not change the name of an already existing Task in the process. You can make it an instance of another *Task* class using the “Class” drop-down list. If you want to change all occurrences of ”Check Hotel Reservation“ you can rename the function class using “Edit Class”. Typing in a new name on the shape directly renames the *Task* class if and only if it has one single instance.

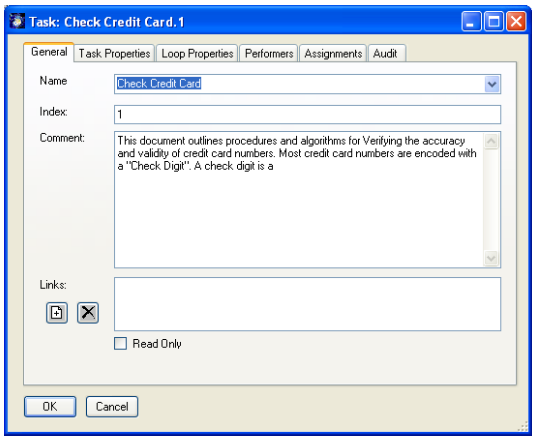
If you click on the “New” button SemTalk pops up an “Add” dialog and you can enter the name of a new class.

Links (hyperlinks, attachments, documents), can be added with the right click menu “New” or the “+” button and removed by the right click menu “Remove” or the “-“ button on the background of the links list. Double clicking the link in the listbox does a shell execute to open that document with the (for windows) associated application. Functions having at least one link will show up with a little marker in the upper left corner.

NOTE: The “General” tab is a generic SemTalk tab and not a BPMN specific one.

In order to display a comment / definition of an object drag & drop the “Annotation” shape in the drawing area and point with the “definition of” connector to one or more objects:


<<None>>
Check Credit Card.1
This document outlines procedures and algorithms for Verifying the accuracy and validity of credit card numbers. Most credit card numbers are encoded with a "Check Digit". A check digit is a



Name: Check Credit Card

Index: 1

Comment: This document outlines procedures and algorithms for Verifying the accuracy and validity of credit card numbers. Most credit card numbers are encoded with a "Check Digit". A check digit is a

Please notice the “Annotation” shape handles. You can use these handles to modify the height and width of the text box, as well as, to rotate the shape and place it where you wish. You can type your text directly on the “Annotation” shape and the changes will be modified automatically in the dialog box the next time you open it.

5.2 Task "Task Properties" Tab

A Task is an atomic activity that is included within a Process. Generally, an end-user and/or an application are used to perform the Task when it is executed.

There are different types of Tasks identified within BPMN to separate the types of inherent behavior that Tasks might represent. These types are the following:

Service Task: is a Task that provides some sort of service, which could be a Web service or an automated application. An In-Message, which indicates that the Message will be sent at the start of the Task must be assigned. An Out-Message which implies the arrival of a message marks the completion of the Task. Implementation specifies the technology that will be used to send and receive the messages. A Web service is the default technology.

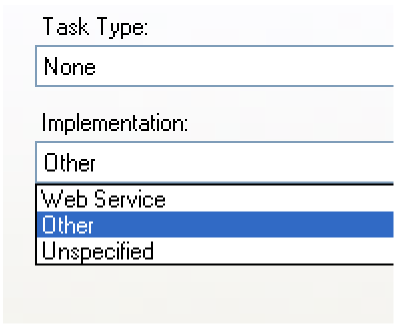
Receive Task: is a simple Task that is designed to wait for a message to arrive from an external Participant (relative to the Process). An In-Message in this context is equivalent to an *in-only* message pattern (Web service). This means that once the message has been received, the Task is completed. A Receive Task is often used to start a Process. If the Task is to be Instantiated, it must have no Incoming Sequence Flow or the Incoming Sequence Flow has a source of a Start Event. A Web service is the default technology.

Send Task: is a simple Task that is designed to send a message to an external Participant (relative to the Process). An Out-Message for must be entered. Once the message has been sent, the Task is completed. The Message in this context is equivalent to an out-only message pattern (Web service). A Web service is the default technology.

User Task: is a typical "workflow" Task where a human performer performs the Task with the assistance of a software application and is scheduled through a Task list manager of some sort. One or more Performers MAY be entered. A Performer(s) defines the human resource that will be performing the User Task. The Performers entry could be in the form of a specific individual, a group, or an organization. (See Task "Performers" Tab). An In-Message and an Out-Message must be entered. The In-Message indicates that the Message will be sent at the start of the Task, and the Out-Message marks the completion of the Task.

Script Task: A Script Task is executed by a business process engine. When the Task is ready to start, the engine will execute the script. When the script is completed, the Task will also be completed. No script will be written in SemTalk, but you can add a file containing an script to the Task shape as an attachment.

Manual Task: is a Task that is expected to be performed without the aid of any business process execution engine or any application. One or more Performers may be entered. The Performers defines the human resource that will be performing the Manual Task. The Performers entry could be in the form of a specific individual, a group, an organization role or position, or an organization.


Task Type:
None
Implementation:
Other

You can define the first five task types (i.e. Service, Receive, Send, User and Script) in the “Task Properties” tab. The Manual task type, but as well as the User task type can be defined in the “Performers” tab.

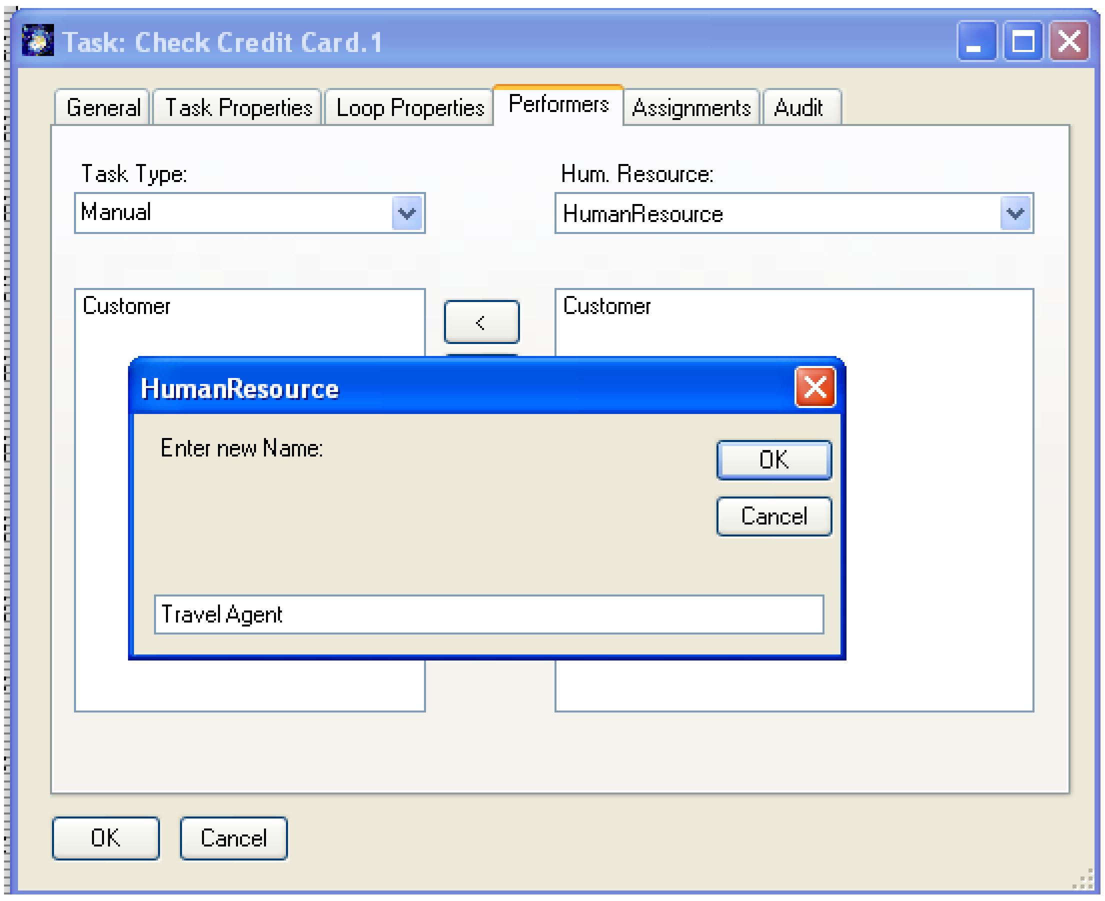
In the **Implementation** drop-down list you have three choices: Other, Unspecified and Web Service. Please choose the most appropriate for the Task.

Choosing one of the **Instantiate** options will determine if this Task will start the business process or not.

In order to quickly identify task types in a process model, a label with the <<task type>> will be placed directly above of the Task name on the shape.

5.3 Task "Performers" Tab

On the "Performers" tab you can assign human resources to the Task. These human resources are grouped under Participants more specifically under Entity and Role.



Task Type: Manual

Hum. Resource: HumanResource

Customer

Customer

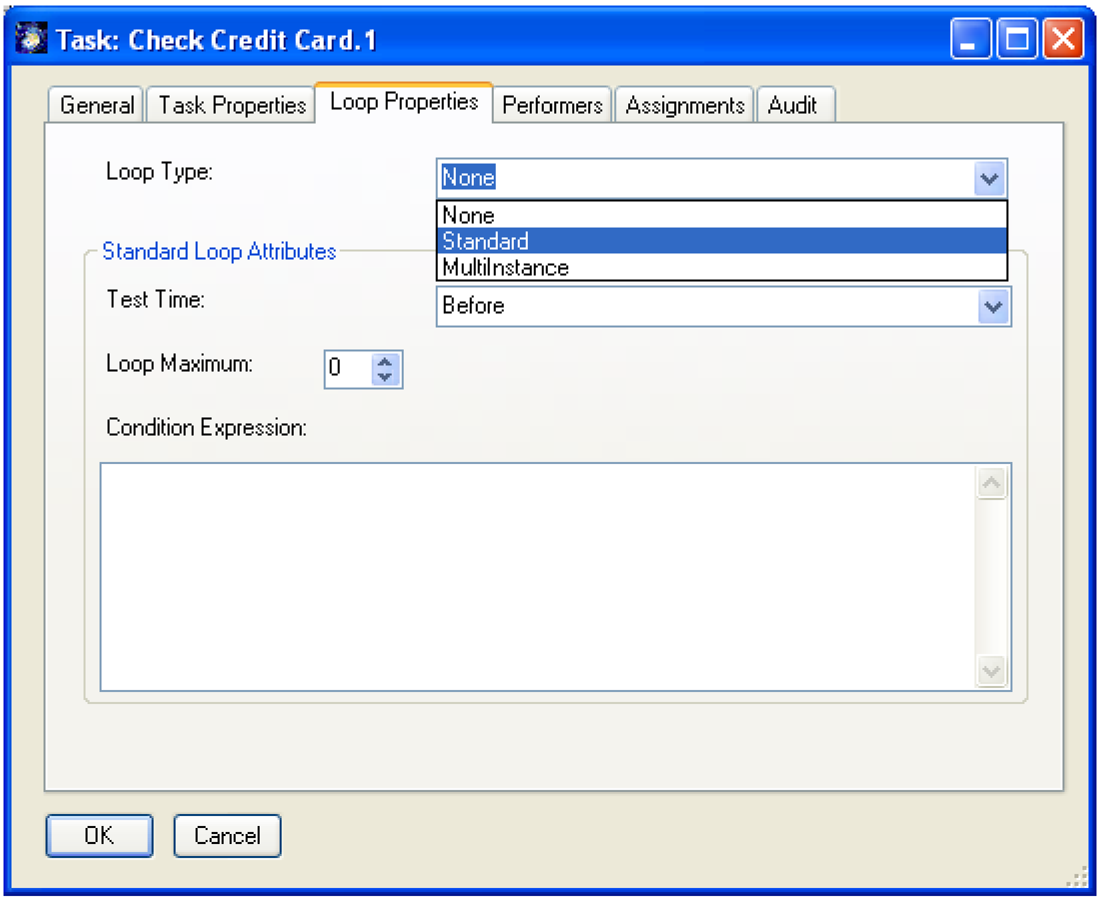
Enter new Name: Travel Agent

These are people (Roles) or organizational units (Entities) which actually execute the Task. The left hand listbox shows the currently assigned resources and the right hand listbox shows the list of all available human resources. You may use double clicking in the listboxes or the arrow buttons to add / remove resources. Using the new button you can add new Participants (in this case, as performers) and new Participant classes. The new instance will automatically get the class which is selected in the drop-down list.

You can see / edit all existing resources in the explorer as child nodes of "Participant"

5.4 Task "Loop Properties" Tab

A Loop can be defined as "A sequence of instructions (in our case a Task) that repeats either a specified number of times or until a particular condition is met". In BPMN you can choose between different types of loops: A Standard Loop and a MultiInstance Loop.


Task: Check Credit Card.1

General | Task Properties | Loop Properties | Performers | Assignments | Audit

Loop Type: None
None
Standard
MultiInstance

Standard Loop Attributes

Test Time: Before

Loop Maximum: 0

Condition Expression:

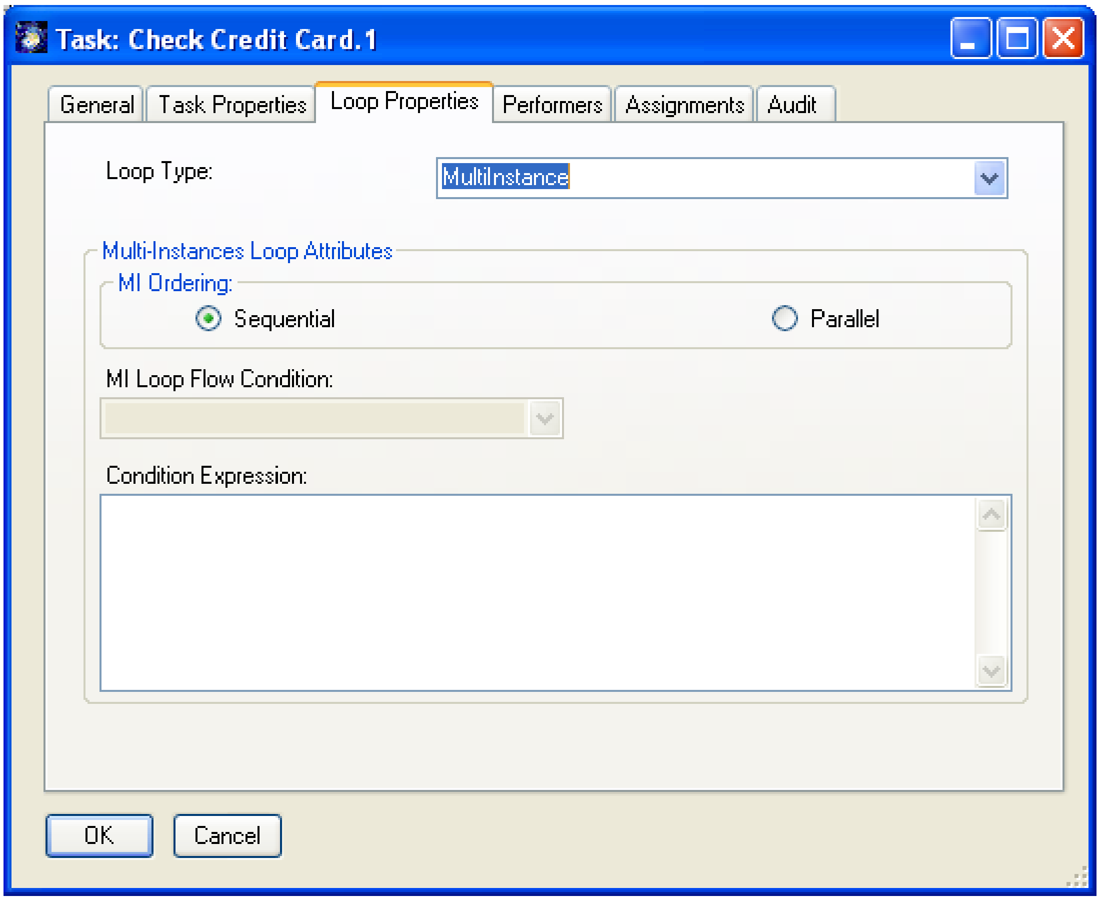
OK | Cancel

A Standard Loop activity will have a boolean **expression** that is evaluated after each cycle of the loop. If the expression is still True, then the loop will continue. There are two variations of the loop, which reflect the programming constructs of while and until. That is, a while loop will evaluate the expression **Before** the activity is performed, which means that the activity may not actually be performed. The until loop will evaluate the expression **After** the activity has been performed, which means that the activity will be performed at least once. In the drop-down list "Test Time" you can choose between both loop variations.

The Loop Maximum is an optional attribute that provides a simple way to add a cap to the number of loops.

In the "Condition Expression" text box write the corresponding expression for this loop.

If you choose a "MultiInstance Loop" you will see the following dialog box:


Task: Check Credit Card.1

Loop Type: MultiInstance

Multi-Instances Loop Attributes

MI Ordering: [✓ Sequential] Parallel

MI Loop Flow Condition:

Condition Expression:

MultiInstance loops reflect the programming construct “foreach”. The loop expression for a MultiInstance loop is a numeric expression evaluated only once before the activity is performed. The result of the expression evaluation will be an integer that will specify the number of times that the activity will be repeated. There are also two variations of the MultiInstance loop to determine whether the instances are performed sequentially or in parallel.

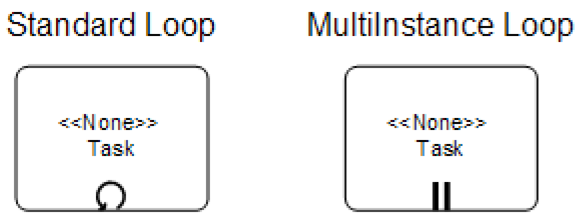
MI Ordering: This applies to only MultiInstance Loops. The MI_Ordering attribute defines whether the loop instances will be performed sequentially or in parallel. Parallel MI_Ordering is equivalent to multi-instance specifications that other notations, such as UML Activity Diagrams use.

MI Loop Flow Condition: This attribute is equivalent to using a Gateway to control the flow past a set of parallel paths. The following are optional values:

- **None**: is the same as uncontrolled flow (no Gateway) and means that all Task instances SHALL generate a token that will continue when that instance is completed.

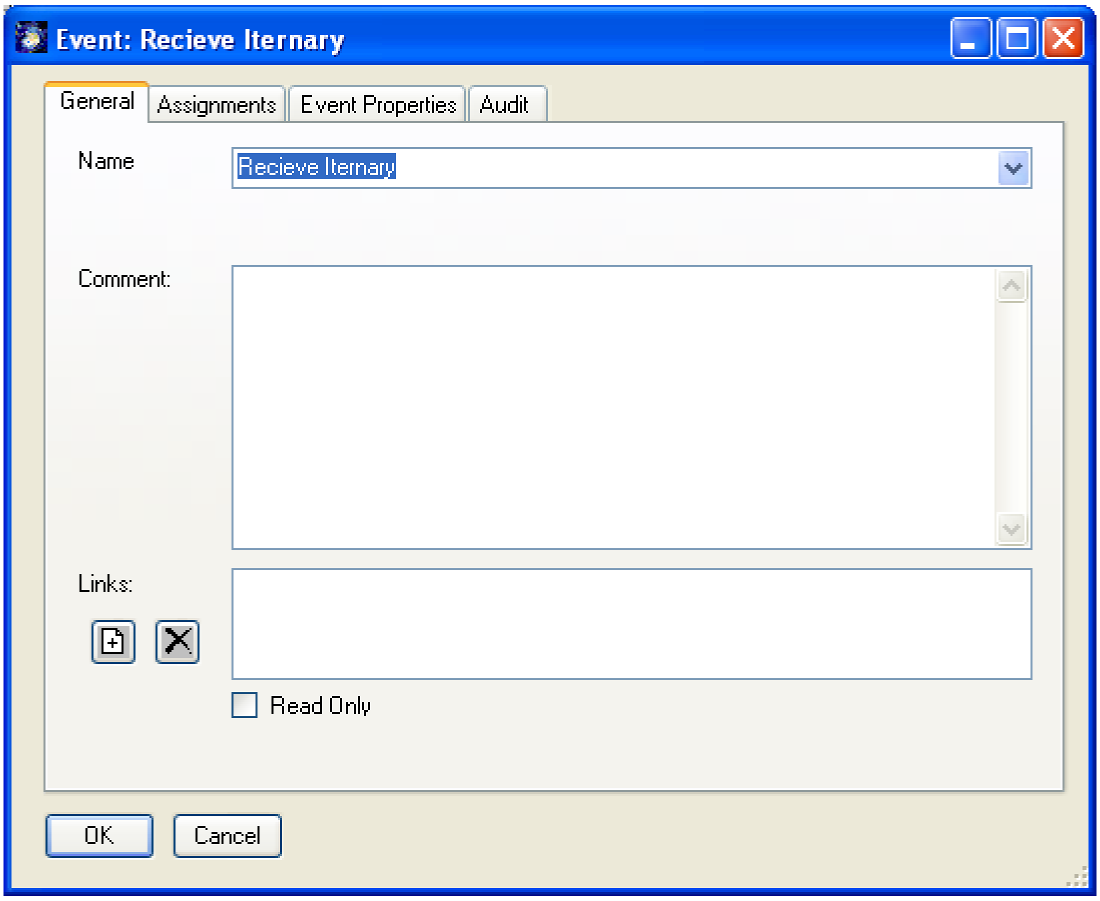
- **One**: is the same as an Exclusive Gateway and means that the Token SHALL continue past the Task after only one of the Task instances has completed. The Task will continue its other instances, but additional Tokens MUST NOT be passed from the Task.
- **All**: is the same as a Parallel Gateway and means that the Token SHALL continue past the Task after all of the Task instances have completed.
- **Complex**: is the same as a Complex Gateway. The Complex MI Flow Condition expression will determine the Token flow.

The marker for a Task that is a standard loop is a small line with an arrow head that curls back upon itself. The marker for a Task that is a MultiInstance loop is a pair of vertical lines in parallel.



5.5 Event Dialog

The Event Dialog will look like this:

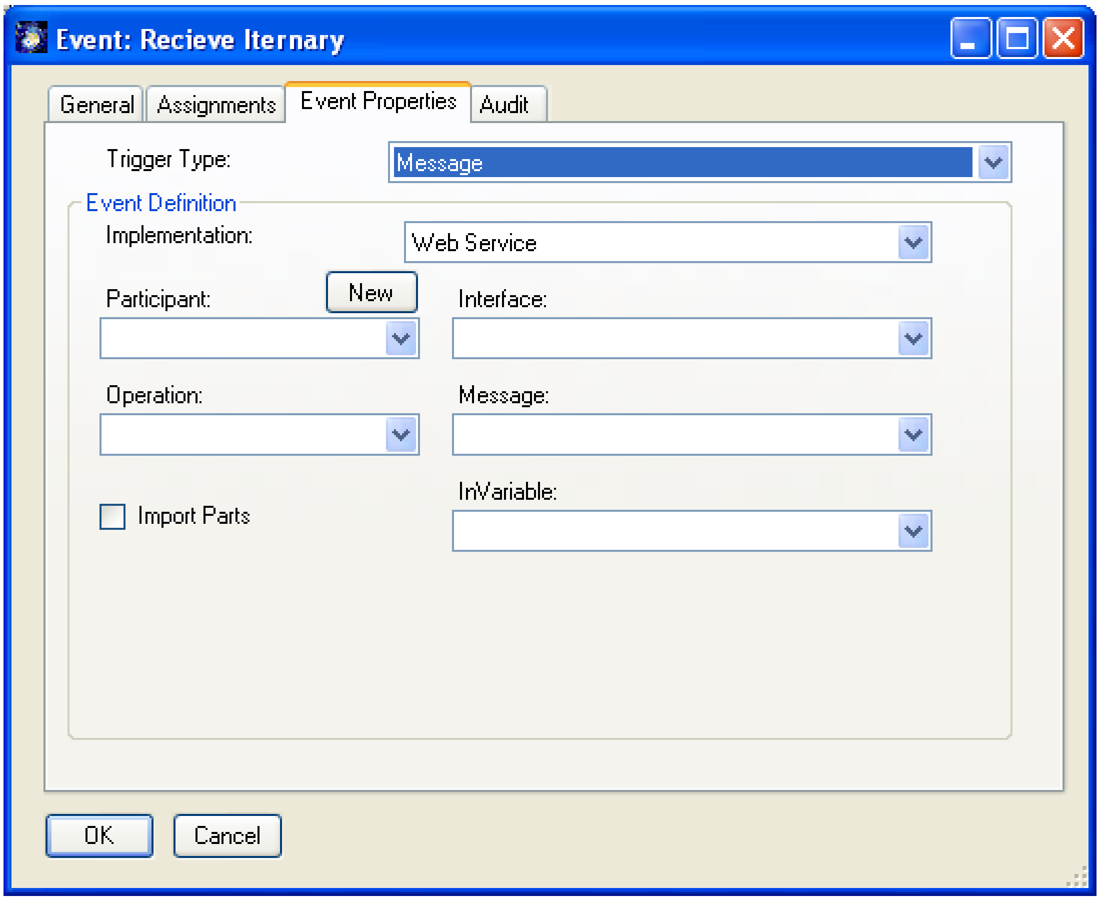


An Event is something that “happens” during the course of a business process. These Events affect the flow of the Process and usually have a cause or an impact. BPMN has restricted the use of Events to include only those types of Events that will affect the sequence or timing of activities of a process.

Please see 4.2 for more information on the “General” Tab.

5.5.1 Event “Event Properties” Tab

An Event has a basic shape in form of a circle, but depending of the type of trigger the shape will acquire a different marker inside the circle (to help identify the Trigger or Result of the Event). Simply drag and drop an Event shape from the BPMN stencil. To define the properties of any kind of Event you will use the “Event Properties” tab. It looks like this:


Event: Recieve Iternary

Trigger Type: Message

Event Definition

Implementation: Web Service

Participant:

Interface:

Operation:

Message:

InVariable:

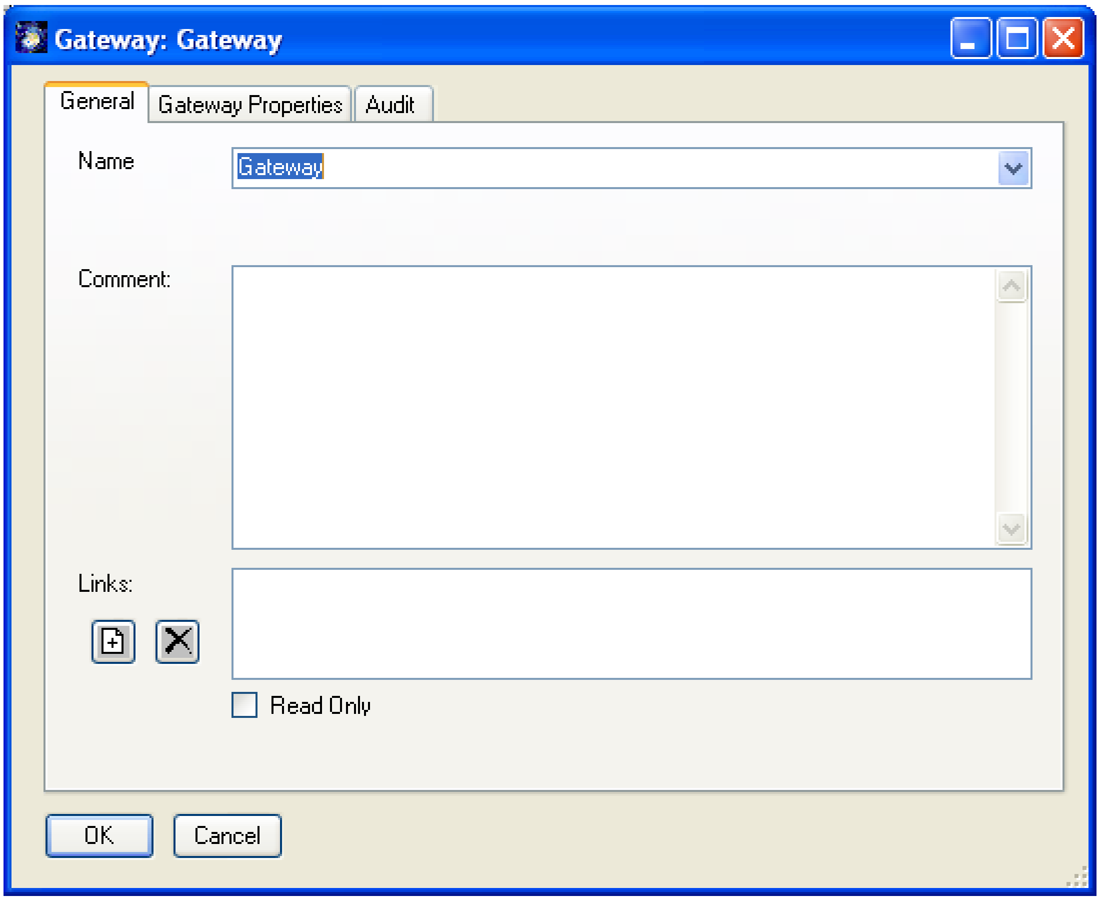
☐ Import Parts

Depending on the position of the Event in the sequence flow you will define a “Trigger” (Start and Intermediate Events) or a “Result” (End Events) in this tab.

The properties of an event depend on its trigger type and it’s Implementation. Please refer to Table 3 for more information about specific event types and their corresponding triggers or results.

5.6 Gateway Dialog

The Gateway dialog looks like this:


Gateway: Gateway

General | Gateway Properties | Audit

Name: Gateway

Comment:

Links:

Read Only

OK | Cancel

Please see section 4.2 for more information on the “General” Tab.

Gateways are modeling elements that are used to control how Sequence Flow interacts as they converge and diverge within a Process. If the flow does not need to be controlled, then a Gateway is not needed. The term “Gateway” implies that as Tokens arrive at a Gateway, they can be merged together on input and/or split apart on output as the Gateway mechanisms are invoked.

There are different types of Gateways (as described in Table 3) and the behavior of each type Gateway will determine how many of the Gates (see Note below) will be available for the continuation of flow.

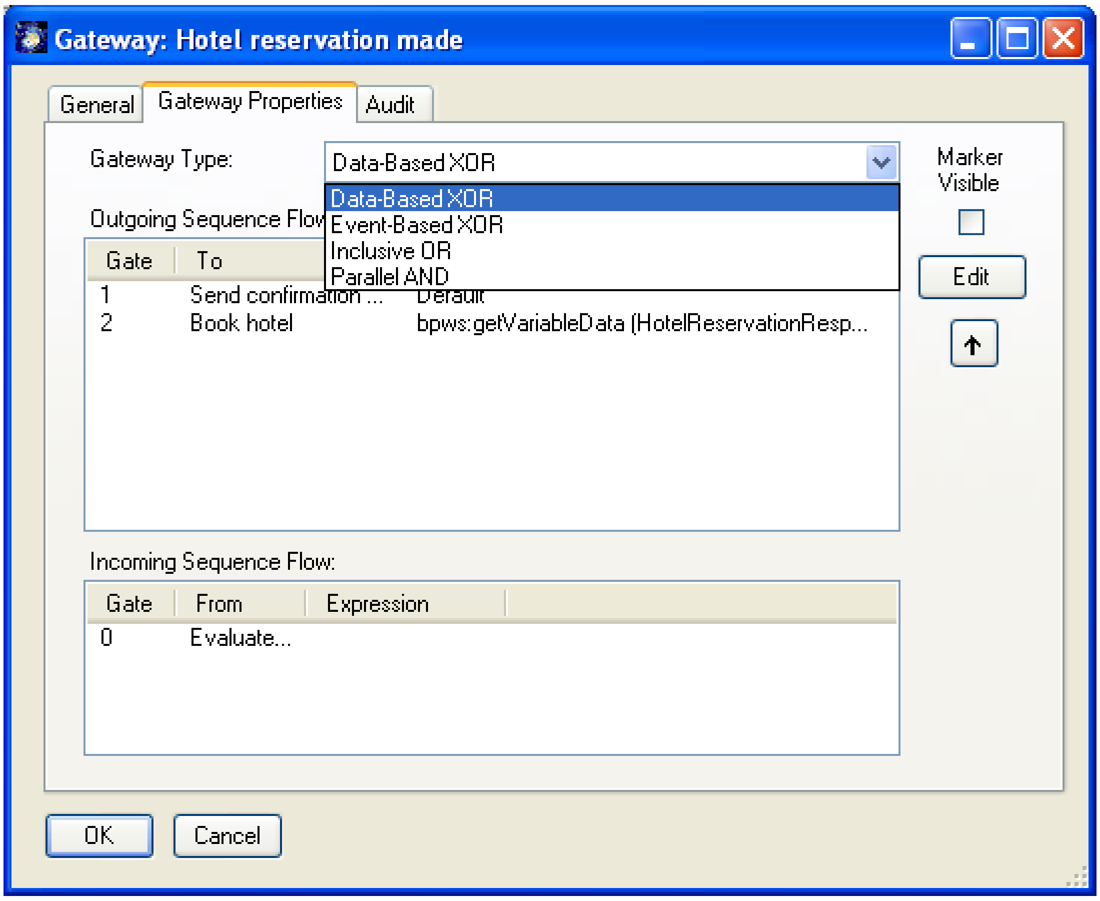
Note on SemTalk BPMN Edition:

> BPMN defines a Gateway as a collection of “Gates.” Each Gate should contain an expression that controls how a Sequence Flow interacts within the Process. Because

there will be one Gate for each outgoing Sequence Flow of the Gateway, in SemTalk BPMN Edition the expression is defined in the Sequence Flow itself and thus, the relationship Gate – Sequence Flow is maintained.

5.6.1 Gateway "Gateway Properties" Tab

Simply drag and drop a Gateway shape from the BPMN stencil. The "Gateway Properties" tab looks like this:


Gateway: Hotel reservation made

General | Gateway Properties | Audit

Gateway Type: Data-Based XOR

- Data-Based XOR
- Event-Based XOR
- Inclusive OR
- Parallel AND

Marker Visible ☐

Outgoing Sequence Flow

Gate	To	
1	Send confirmation ...	Default
2	Book hotel	bpws:getVariableData (HotelReservationResp...

Incoming Sequence Flow:

Gate	From	Expression
0	Evaluate...	

Gateway Type: From the drop-down list you will choose which Gateway type you want to include at this time in your model. A Data-Based XOR Gateway is set by default. The rest of the controls will react differently to every Gateway type you have chosen. The shape appearance will change automatically to correspond to its graphical representation in the BPMN specs.

Marker Visible: This option determines if the XOR Marker is displayed in the center of the Gateway diamond (an "X") for a Data-Based Gateway. The marker is displayed if the check box is checked in. By default, the marker is not displayed.

Edit: The “Edit” button is used to edit the assigned condition expressions from this dialog. You must select the sequence flow to which the expression was assigned and click on the button. This will open first the sequence flow dialog and then you can edit the expression.

🡅: With this button you can change the gates’ evaluation order. The default gate will always have number “0” meaning that it will be the last one to be evaluated.

Outgoing Sequence Flow: The columns show important information about the outgoing sequence flows, their target object, their evaluation position in the evaluation order (only valid for Data-Based Gateways) and their condition expressions.

Incoming Sequence Flow: The columns show important information about the incoming sequence flows, their source object and their condition expressions (only valid for Complex Gateways).

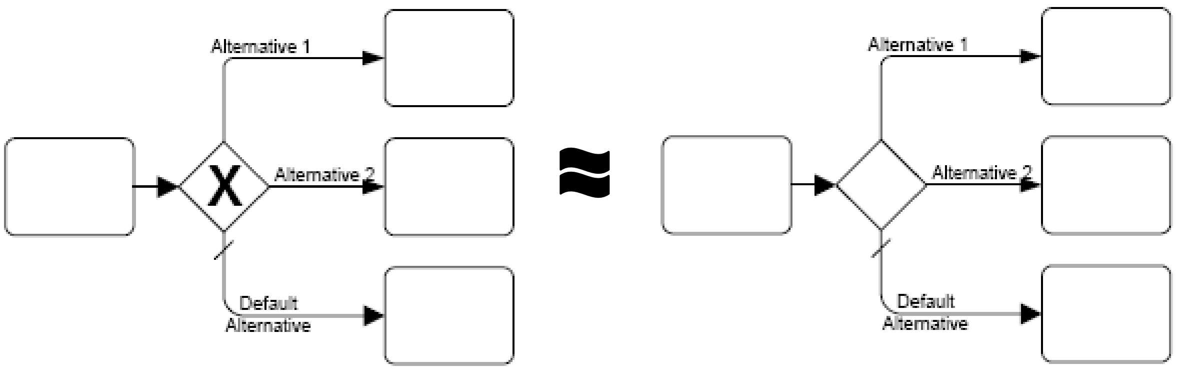
By double-clicking on the header of each column you can sort the columns ascending or descending.

As you can see in Table 3 BPMN includes 5 different types of Gateways. We will discuss each one of them briefly.

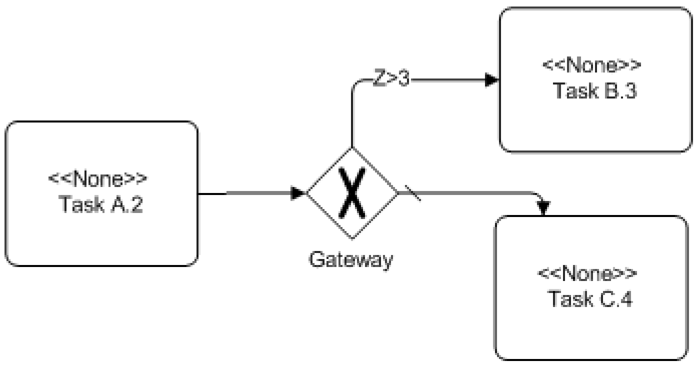
5.6.2 Data-Based Gateway (XOR):

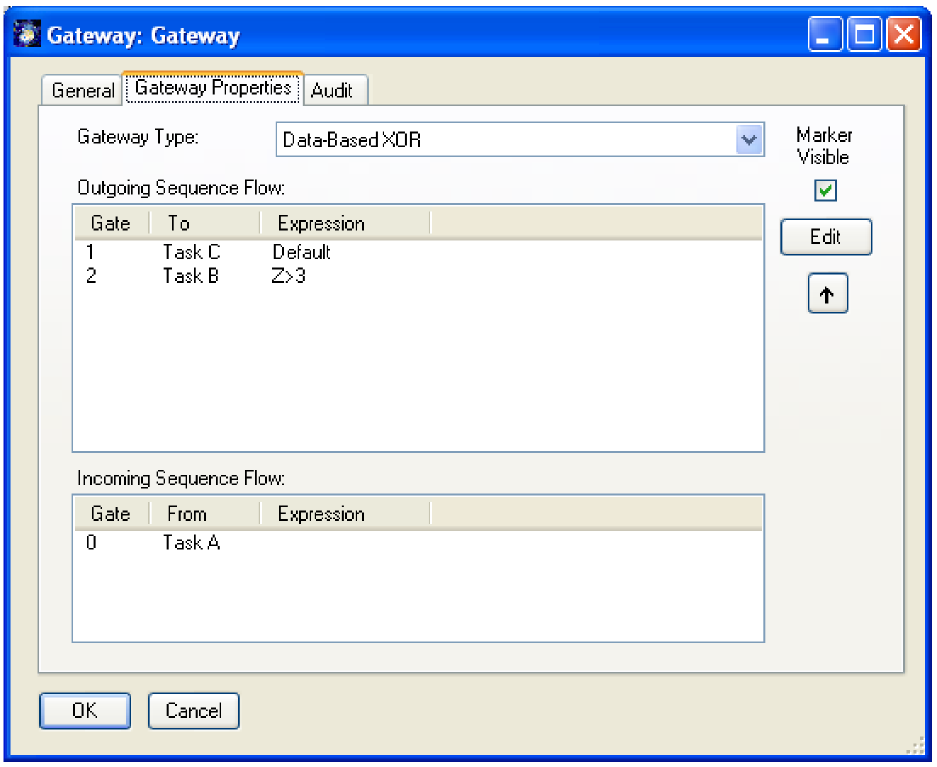
The Data-Based Exclusive Gateways are the most commonly used type of Gateways. The set of Gates for Data-Based Exclusive Decisions is based on the boolean expression contained in the Condition Expression of the outgoing Sequence Flow of the Gateway. These expressions use the values of process data to determine which path should be taken (hence the name Data-Based).

The Data-Based Exclusive Gateway MAY use a marker that is shaped like an “X” and is placed within the Gateway diamond to distinguish it from other Gateways.



The following example illustrates the handling of a Data-Based Gateway dialog:





Gateway Type: Data-Based XOR

Marker Visible ✓

Outgoing Sequence Flow:

Gate	To	Expression
1	Task C	Default
2	Task B	Z>3

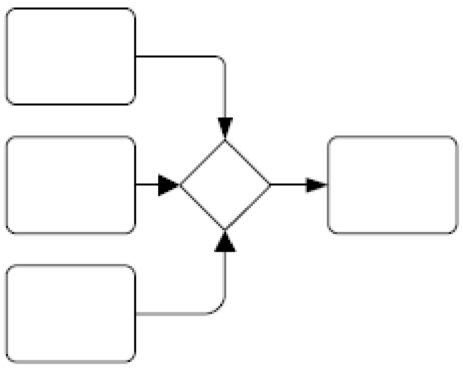
Incoming Sequence Flow:

Gate	From	Expression
0	Task A	

The default Sequence Flow (the Default attribute is defined in the Sequence flow dialog, see section 6.3) is not mandatory for a Gateway. This means that if it is not used, then it is up to the modeler to ensure that at least one Sequence Flow (at least one expression) be valid at runtime.

The condition expressions should be evaluated in a specific order. The order position is shown as the integer on the third of the Outgoing Sequence Flow list. The first one that evaluates as TRUE will determine the Sequence flow that will be taken. Since the behavior of this Gateway is exclusive, any other conditions that may actually be TRUE will be ignored--only one Sequence flow can be chosen. One of the Sequence Flows may be “default” (or otherwise), and is the last Sequence Flow considered (it will always have the evaluation position (Eval Pos) number “0”). This means that if none of the other Sequence Flows are chosen, then the default Sequence Flow will be chosen.

The merging behavior of the Gateway can also be modeled using a Data-based Gateway:



The figure on the right shows an XOR Gateway used to merge three different incoming Sequence Flows. All the incoming Sequence Flows are alternative, i.e. only one of the three Sequence Flows would ever pass a Token at one time.

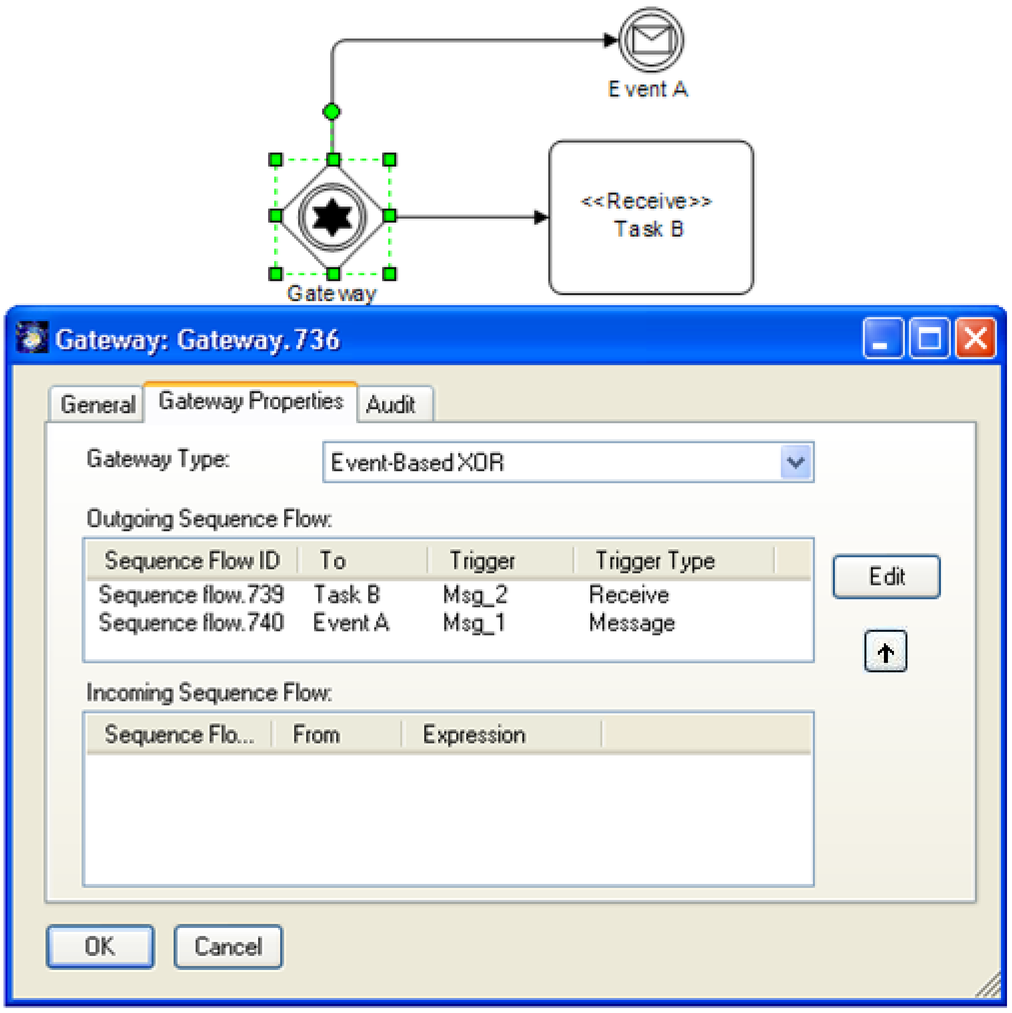
If there are multiple incoming Sequence Flows, all of them will be used to continue the flow of the Process (as if there were no Gateway). That is, Process flow SHALL continue when a signal (a Token) arrives from any of a set of Sequence Flow. Signals from other Sequence Flow within that set may arrive at other times and the flow will continue when they arrive as well, without consideration or synchronization of signals that have arrived from other Sequence Flow.

In simple situations, Exclusive Gateways need not be used for merging Sequence Flow, but there are more complex situations where they are required. Thus, a modeler should always be aware of the behavior of a situation where Sequence Flows are uncontrolled.

5.6.3 Event-Based Gateway (XOR):

On the input side, their behavior is the same as a Data-Based Exclusive Gateway. On the output side, this Decision represents a branching point in the process where the alternatives are based on Events that occurs at that point in the Process, rather than the evaluation of expressions using process data. A specific Event, for example the receipt of a message, determines which of the paths will be taken. The receipt of the message can be modeled with a Task of TaskType Receive or an Intermediate Event with a Message Trigger. In addition to Messages, other Triggers for Intermediate Events can be used, such as Timers and Errors. Furthermore, any outgoing Sequence Flows MUST not contain a condition expression.

Because this Gateway is an Exclusive Gateway, the merging functionality for the Event-Based Exclusive Gateway is the same as the Data-Based Exclusive Gateway described in the previous section.



Gateway: Gateway.736

Gateway Type: Event-Based XOR

Outgoing Sequence Flow:

Sequence Flow ID	To	Trigger	Trigger Type
Sequence flow.739	Task B	Msg_2	Receive
Sequence flow.740	Event A	Msg_1	Message

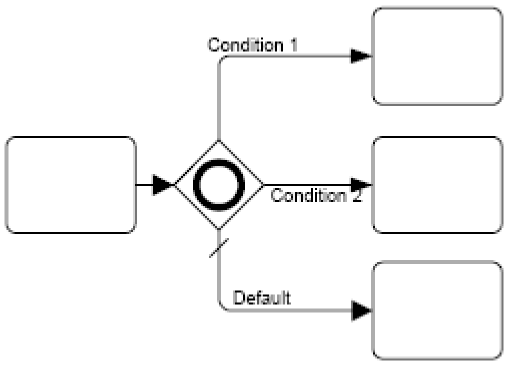
Incoming Sequence Flow:

Sequence Flo...	From	Expression

The Gateway Properties Tab for an Event-Based XOR Gateway looks a little different than in the other cases, because as mentioned before, the Gateway evaluates Events that occur at this point. You see now that the third and fourth columns of the Outgoing Sequence Flow list have the headers “Trigger” and “Trigger Type” respectively. If you select the corresponding Sequence flow and press the button “Edit” you will open the corresponding dialog of the Event or Task.

5.6.4 Inclusive Gateway (OR)

The Inclusive Gateway represents a branching point where alternatives are based on conditional expressions contained within outgoing Sequence Flow. However, in this case, the True evaluation of one condition expression does not exclude the evaluation of other condition expressions. All Sequence Flow with a True evaluation will be traversed by a Token. For a single incoming Token, there may be a Token generated for one to all of the outgoing Sequence Flow.



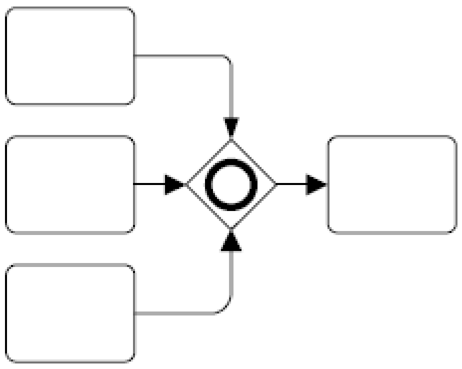
Since each path is independent, all combinations of the paths may be taken, from zero to all. However, it should be designed so that at least one path is taken.

The Inclusive Gateway will use a marker that is in the shape of a circle or an “O” and is placed within the Gateway diamond to distinguish it from other Gateways.

There is no need for a specific order to evaluate the condition expressions. The order position is shown as the integer on the third of the Outgoing Sequence Flow and it will serve only as auxiliary guide to the modeler. The author recommends leaving it as zero.

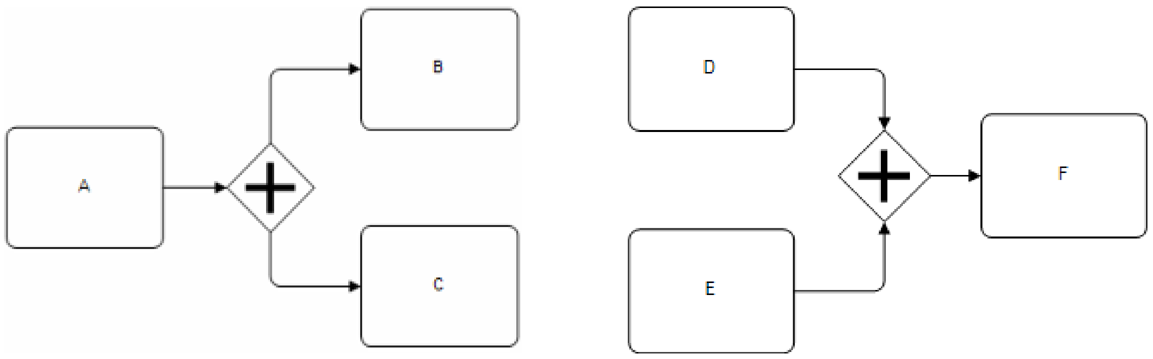
When the Inclusive Gateway is used as a Merge, it will wait for (synchronize) all Tokens that have been produced upstream. It does not require that all incoming Sequence Flow produce a Token (as the Parallel Gateway (AND) does).

If an upstream Inclusive OR produces two out of a possible three Tokens, then a downstream Inclusive OR will synchronize those two Tokens and not wait for another Token, even though there are three incoming Sequence Flows.



5.6.5 Parallel Gateway (AND):

Parallel Gateways can be used to synchronize parallel flow and to create parallel flow.



These Gateways are not required to create parallel flow, but they can be used to clarify the behavior of complex situations where a string of Gateways are used and parallel flow is required. The associated Sequence Flow must have its Condition attribute set to None, i.e. no Condition Expression will be defined for these Sequence Flows.

The Parallel Gateway uses a marker that is in the shape of a plus sign and is placed within the Gateway diamond to distinguish it from other Gateways.

6 Connecting Process Elements

This section defines the graphical objects used to connect two process objects and how the flow progresses through a Process.

There are three ways of connecting process objects in BPMN: the Sequence Flow, the Message Flow and the Association. SemTalk adds an association called “definitionOf” (for more information see page 16). Sequence Flow and Message Flow will affect the performance of activities within a Process. An Association and “definitionOf” will only provide extra information about the process.

The Sequence Flow and the Message Flow represent graphically orthogonal aspects of a business processes. On one hand, a Sequence Flow is used to show the order that activities will be performed in a Process. On the other hand, a Message Flow is used to show the flow of messages between two entities that are prepared to send and receive them. Thus, a Sequence Flow will generally flow in a single direction (either left to right, or top to bottom) and Message Flow will flow at a 90° from the Sequence Flow.

Depending on the object itself and on the purpose of the connecting object (progression flow [Sequence Flow] or communication flow [Message Flow]) BPMN defines a set of flow object connection rules.

6.1 Sequence Flow Rules

Table 1 displays the BPMN flow objects and shows how these objects can connect to one another through Sequence Flow. The ✓symbol indicates that the object listed in the row can connect to the object listed in the column.

Table 1: Sequence Flow Rules

To / From	○	Task	◇	◎	**○**
○		✓	✓	✓	✓
Task		✓	✓	✓	✓
◇		✓	✓	✓	✓
◎		✓	✓	✓	✓
○					

Note: Only those objects that can have incoming and/or outgoing Sequence Flow are shown in the table. Thus, Lane, Data Object, and Annotation are not listed in the table.

6.2 *Message Flow Rules*

Table 2 displays the BPMN modeling objects and shows how these objects can connect to one another through Message Flow. The ✓ symbol indicates that the object listed in the row can connect to the object listed in the column.

All Message Flow must connect two separate entities. They can connect to the Lane boundary or to Flow Objects within the Lane boundary. They cannot connect two objects within the same Lane.

Table 2: Message Flow Rules

To / From	○	Task	▭	◎	⭘
○					
Task	✓	✓	✓	✓	
▭	✓	✓	✓	✓	
◎		*	*	*	
⭘	✓	✓	✓	✓	

* An Intermediate Event of type Message MAY be the target for Message Flow; it can have one incoming Message Flow. If the Event has an incoming Message Flow, then it MAY NOT have an outgoing Message Flow.

* An Intermediate Event MAY be a source for Message Flow; it can have no outgoing Message Flow. If the Event has an outgoing Message Flow, then it MAY NOT have an incoming Message Flow.

Note: Only those objects that can have incoming and/or outgoing Message Flow are shown in the table. Thus, Gateway, Data Object, and Annotation are not listed in the table.

6.3 Sequence Flow

A Sequence Flow is used to show the order that activities will be performed in a Process. Each Flow has only one source and only one target (see Table XXX for allowed sources and targets).

A Sequence Flow cannot be called a “Control Flow” because a Message Flow can also affect the progress of a process (for example by triggering a Message Event). A Sequence Flow can be either a “Uncontrolled Flow” and “Conditional Flow” or a “Default Flow”.

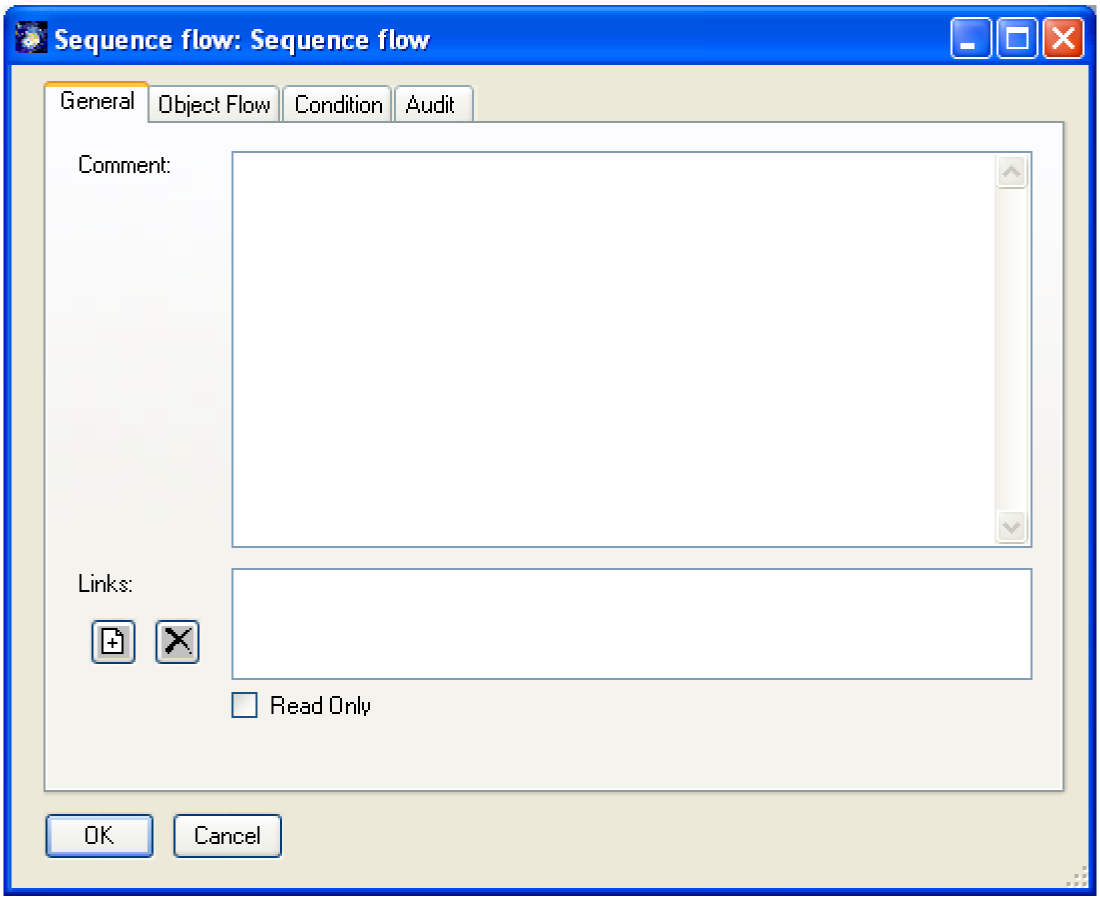
→ An Uncontrolled Flow is a Sequence Flow not affected by conditions or coming out of a Gateway..

◇→ A Conditional Flow is a Sequence Flow controlled by conditions (specified in a condition expression) or coming out of a Gateway. The “mini-diamond” shape is used to differentiate it from a Uncontrolled Flow only when its source is not a Gateway.

⧵→ A Default Flow is a Sequence Flow is a Sequence Flow used to insure that at least one Gate in a Data-Based Exclusive or an Inclusive Gateway be valid at runtime. If no Gate is valid, then the model is invalid.

6.3.1 Sequence Flow Dialog

The Sequence Flow dialog looks like this:

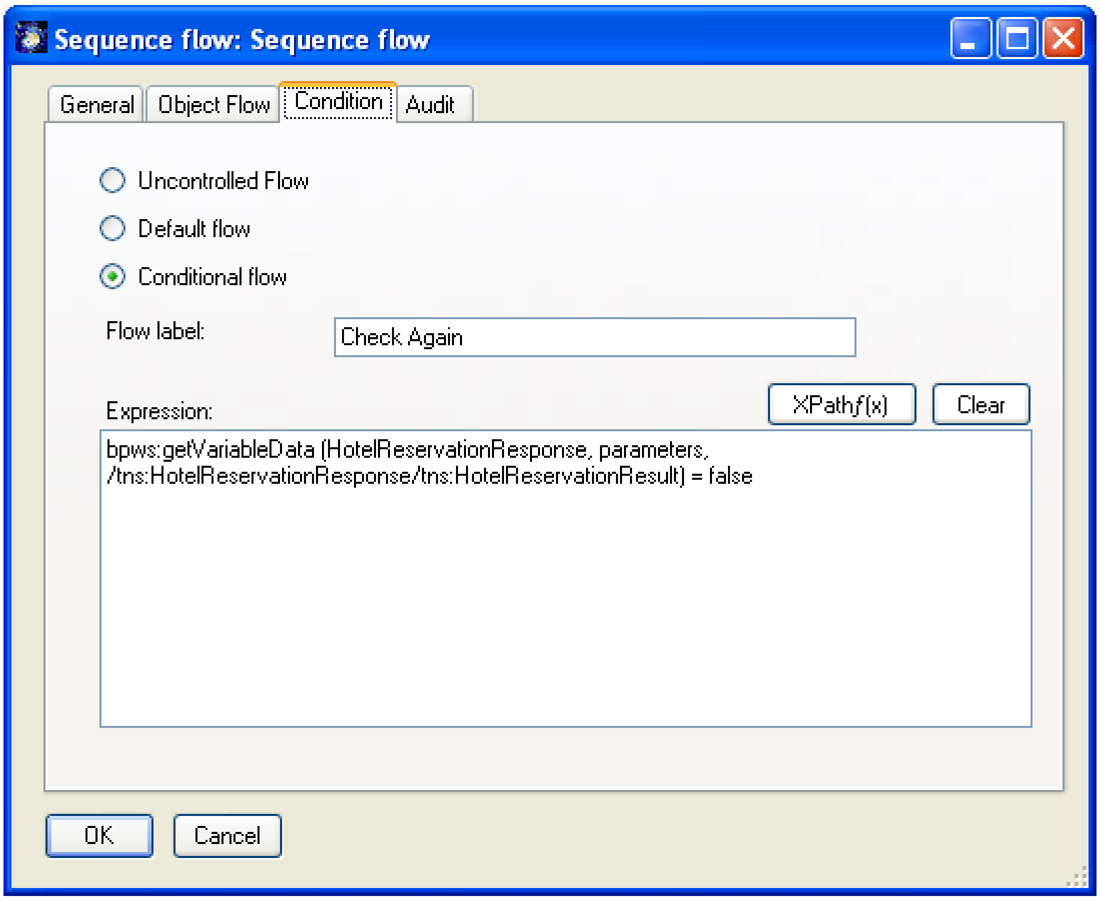


Links (hyperlinks, attachments, documents), can be added with the right click menu “New” or the “+” button and removed by the right click menu “Remove” or the “-“ button on the background of the links list. Double clicking the link in the listbox does a shell execute to open that document with the (for windows) associated application. Functions having at least one link will show up with a little marker in the upper left corner.

In order to display a comment / definition of an object drag & drop the “Annotation” shape in the drawing area and point with the “definition of” connector to one or more objects.

6.3.2 Sequence Flow “Condition” Tab

In this tab, you can define different aspects of a Sequence Flow. For example, you can decide whether a Sequence Flow represent an uncontrolled flow, a conditional flow or a default flow. The Sequence Flow “Condition” Tab looks like this:



Conditional flow [✓]

Flow label: Check Again

Expression:

```
bpws:getVariableData (HotelReservationResponse, parameters,
/tns:HotelReservationResponse/tns:HotelReservationResult) = false
```

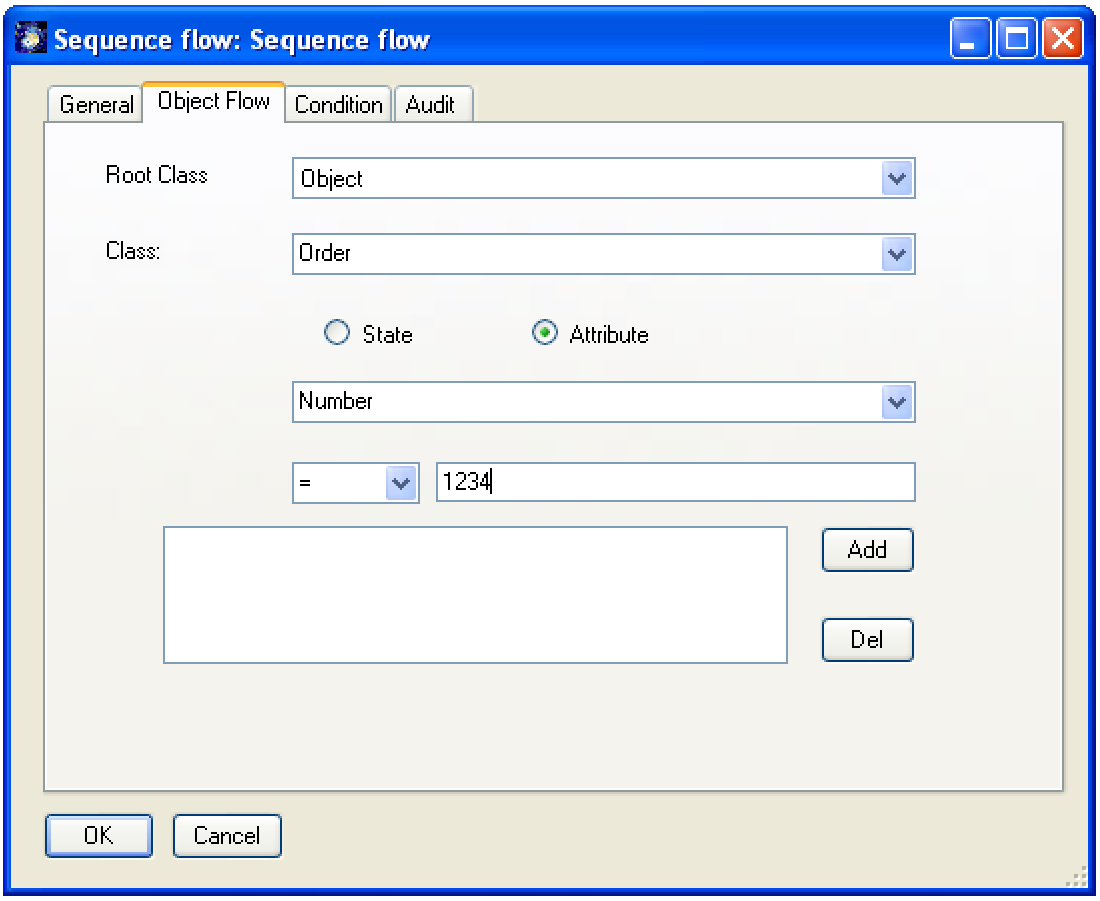
With the options **Uncontrolled Flow** or **Default Flow** you can define an equivalent behavior to the Sequence Flow and you will not be able to add more information to the Flow. With the two options grouped under **Condition Flow** you will define, first of all, the behavior of the Sequence Flow as conditional (it contains a condition expression to be evaluated before a token passes) and you will decided if the label of the Sequence Flow should be the expression itself or another label.

Flow label: In this textbox you will enter the label you want to see on the Sequence Flow in your model.

Expression: This textbox will show the expression formula that you previously defined for an expression in this Sequence Flow.

6.3.3 Sequence Flow "Object Flow" Tab

A Data Object is considered an Artifact because they do not have any direct affect on the Sequence Flow of the Process, but they do provide information about what the Process does. That is, how documents, data, and other objects are used and updated during the Process. While the name "Data Object" may imply an electronic document, they can be used to represent many different types of objects, both electronic and physical. The “Data Object Properties” Tab looks like this:



Root Class: Object
Class: Order
( ) State (•) Attribute
Number
= 1234

The base class for a Data Object is simply “Object”. You can select existing objects from the drop-down lists. If you select an object type or enter a new one, you must push the “**Add**” button in order to add it to the list of information that flows.

Object composition is done similar to the “Compose” in the add activity dialog:

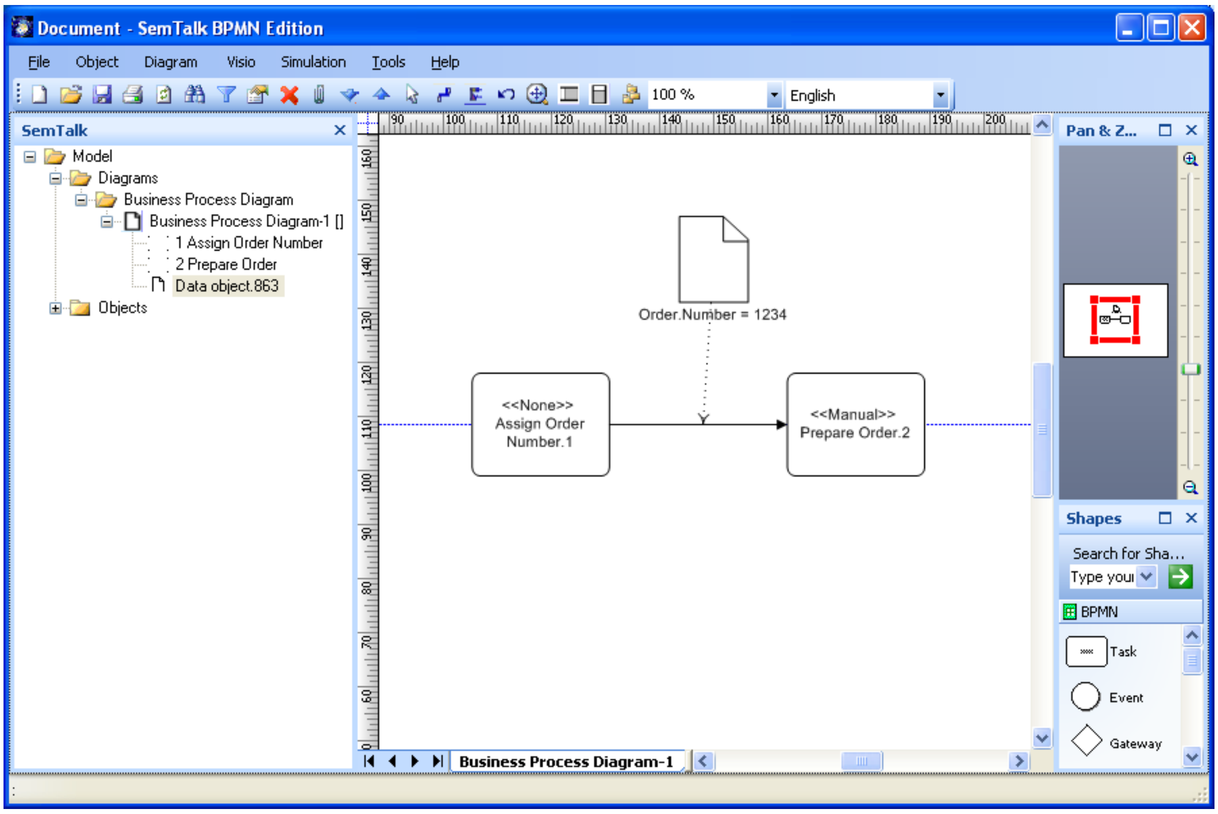
RootClass is the base object type: Object.

Class is the object. You can reuse existing classes or create new classes in a bottom up style.

State (and Attribute) is the state of the object which is being send e.g. “Order entered”.

You can remove an object from the list by pressing the button “**Del**”.

In order to visualize the Data Objects you have defined, please drag the shape “Data Object” from the BPMN stencil and connect its “Association” connector to the Sequence Flow.

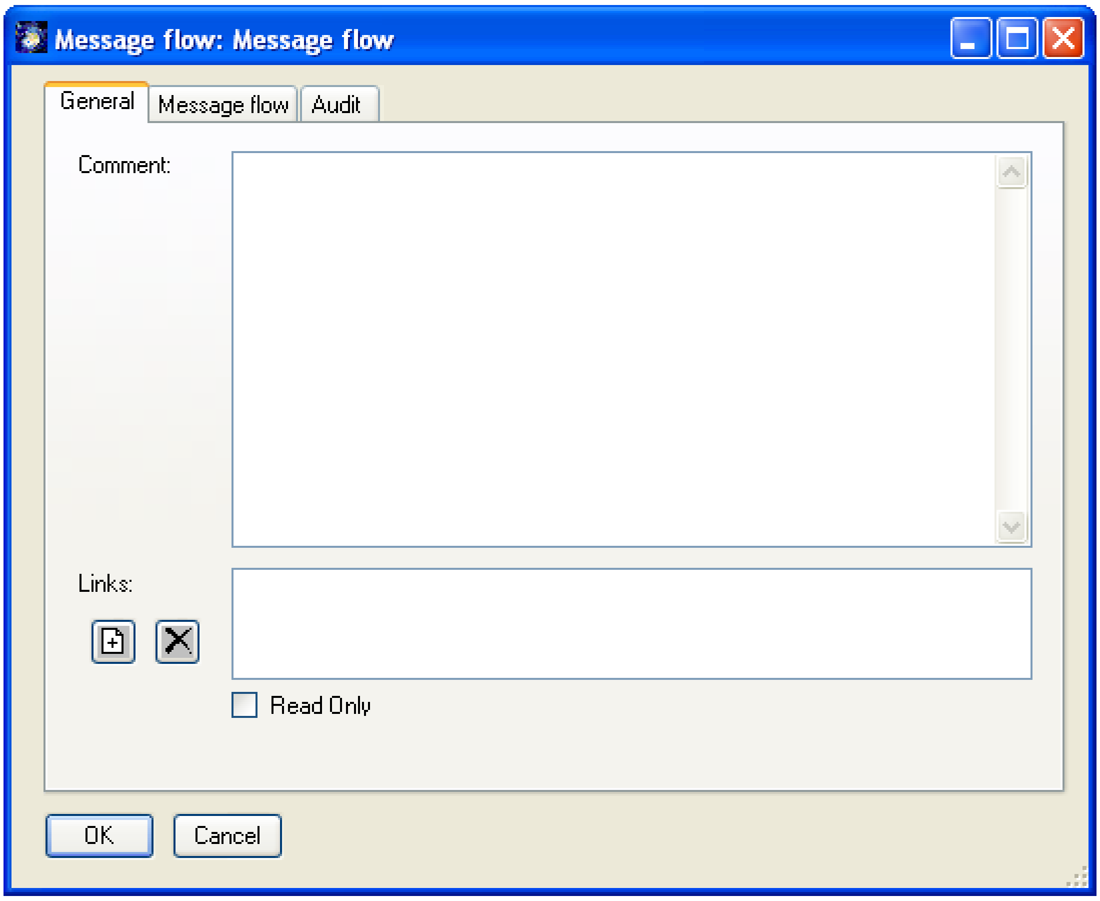


6.4 Message Flow

A Message Flow is used to show the flow of messages between two particpants that are prepared to send and receive them. In BPMN, two separate Lanes in the Diagram will represent the two entities. Thus, Message Flow MUST connect two Lanes.

6.4.1 Message Flow Dialog

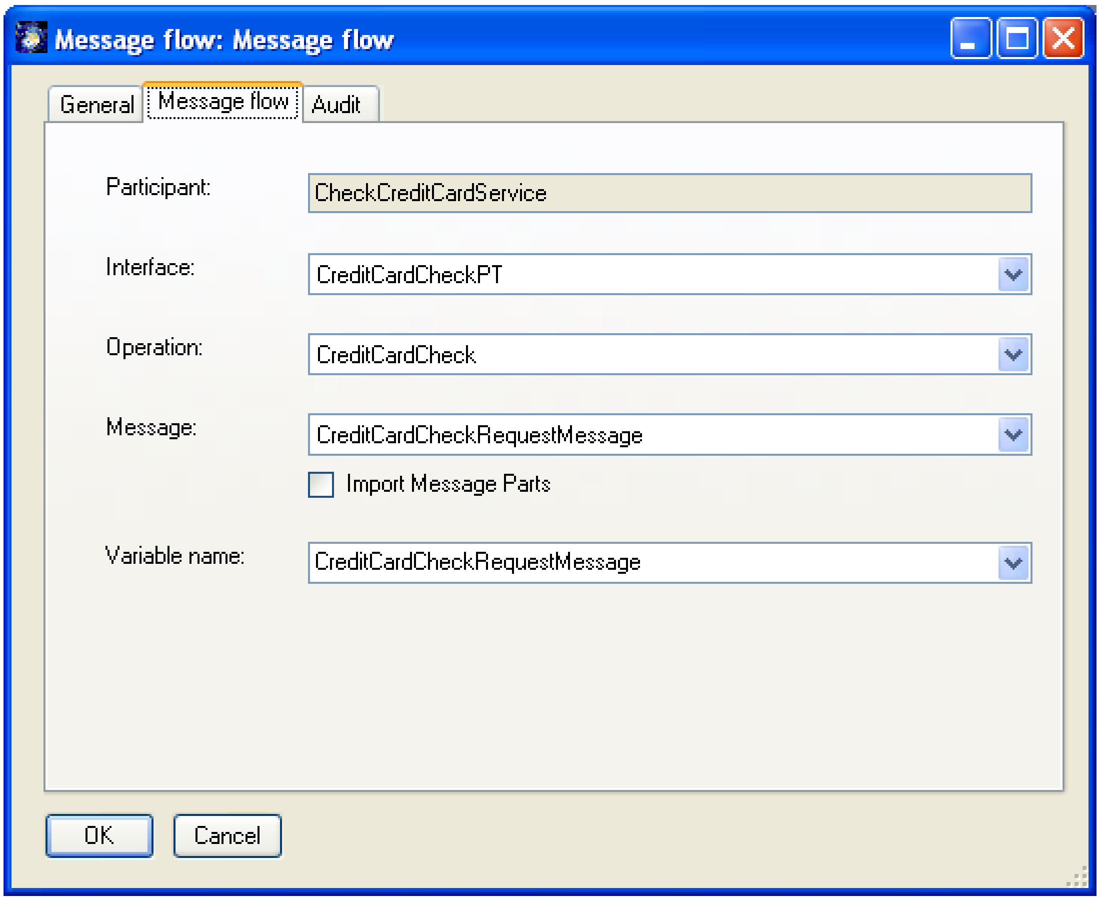
The Message Flow dialog looks like this:



Its handling is the same as for a Sequence Flow (see section 6.3).

6.4.2 Message flow Properties Tab

In this tab, you can define assign a Message to a Message Flow. The Message Flow "Connecting Object Properties" Tab looks like this:



Participant: CheckCreditCardService

Interface: CreditCardCheckPT

Operation: CreditCardCheck

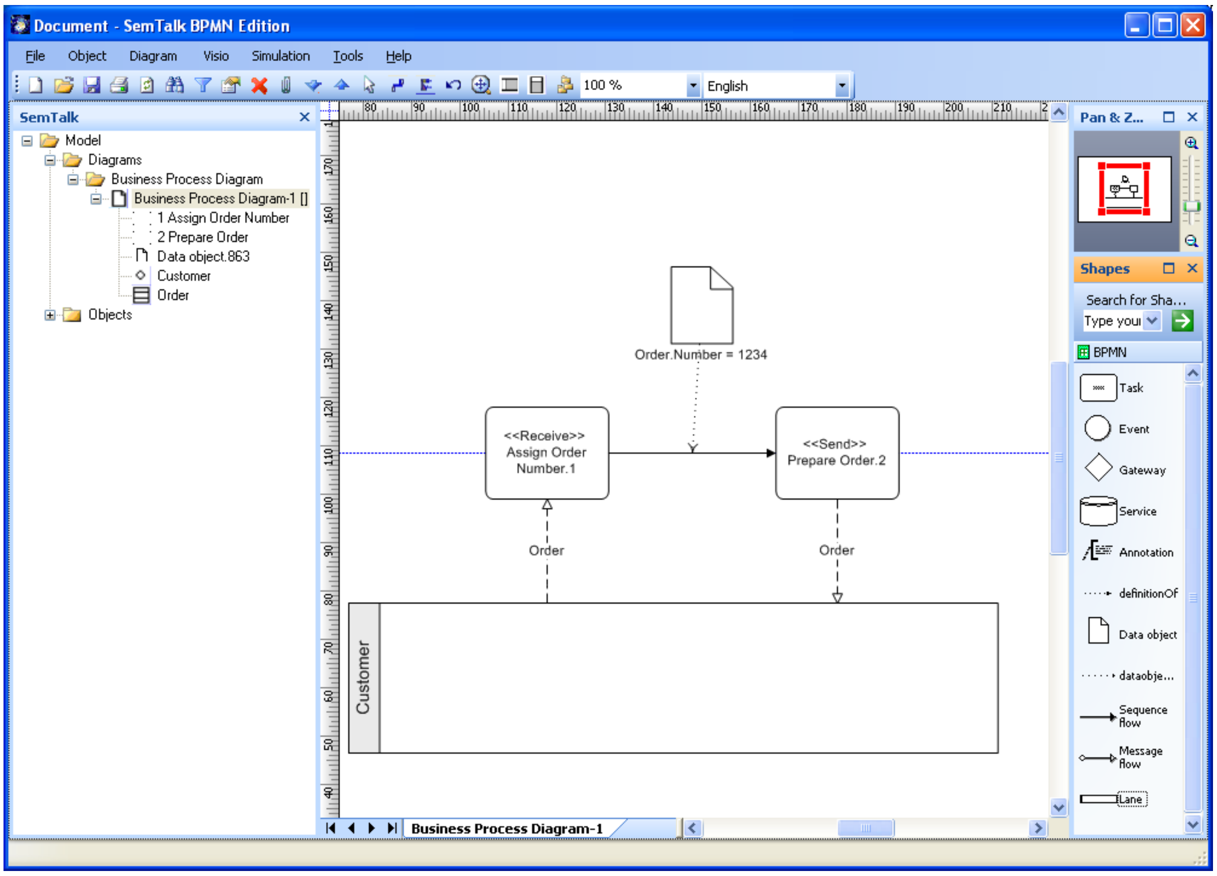
Message: CreditCardCheckRequestMessage

☐ Import Message Parts

Variable name: CreditCardCheckRequestMessage

In the drop-down list under the title “Enter new message name” you will define a the new message new or choose from the drop-down list list an information object as basis for the message. After choosing one, press “Add” to create the message and a Message dialog will open so you can define further properties of a message.

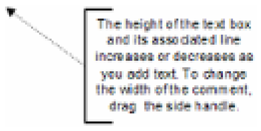
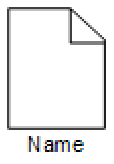
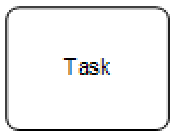
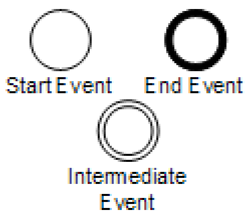
The following is an example:


Document - SemTalk BPMN Edition
File Object Diagram Visio Simulation Tools Help
100 %
English
SemTalk
Model
Diagrams
Business Process Diagram
Business Process Diagram-1 []
1 Assign Order Number
2 Prepare Order
Data object.863
Customer
Order
Objects
Order.Number = 1234
<<Receive>> Assign Order Number.1
<<Send>> Prepare Order.2
Order
Order
Customer
Business Process Diagram-1
Pan & Z...
Shapes
Search for Sha...
Type your
BPMN
Task
Event
Gateway
Service
Annotation
definitionOf
Data object
dataobje...
Sequence flow
Message flow
Lane

7 BPMN Modeling Elements

The following table shows the graphical elements and their corresponding markers:

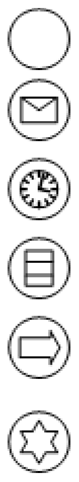
Table 3: BPMN Graphical Elements

Graphical Elements	Marker
Artifact: Used to show additional information about a process that is not directly related to the Sequence Flow or Message Flow of the Process.	
Annotation: Mechanism for a modeler to provide additional information for the reader of a BPMN Diagram. It is connected to a specific object on the Diagram with an "definitionOf" connector, but do not affect the flow of the Process. Text associated with the Annotation is be placed within the bounds of the open rectangle.	 The height of the text box and its associated line increases or decreases as you add text. To change the width of the comment, drag the side handle.
Data object: They do not have any direct affect on the Sequence Flow or Message Flow of the process, but they do provide information about what the Process does. That is, how documents, data, and other objects are used and updated during the Process. While the name "Data Object" may imply an electronic document, they can be used to represent many different types of objects, both electronic and physical.	
Flow Object: A Flow Object is one of the set of following graphical objects: Events, Tasks, and Gateways.	
Task: It is an atomic activity that is included within a Process. A Task is used when the work in the Process is not broken down to a finer level of Process Model detail. Generally, an end-user and/or an application are used to perform the Task when it is executed. Task types: • Service Task: Task that provides some sort of service, which could be a Web service or an automated application. • Receive Task: Task that is designed to wait for a message to arrive from an external participant. Once the message has been received, the Task is completed. • Send Task: Task that is designed to send a message to an external participant. Once the message has been sent, the Task is completed. • User Task: Typical "workflow" task where a human performer performs the Task with the assistance of a software application and is scheduled through a task list manager of some sort. • Script Task: Tasks executed by a business process engine. The modeller or implementer defines a script in a language that the engine can interpret. When the Task is ready to start, the engine will execute the script. When the script is completed, the Task will also be completed. • A Manual Task is a Task that is expected to be performed without the aid of any business process execution engine or any application.	
Event: Is something that "happens" during the course of a business process. These Events affect the flow of the Process and usually have a cause or an impact. BPMN has restricted the use of events to include only those types of events that will affect the sequence or timing of activities of a process. Start and most Intermediate Events have "Triggers" that define the cause for the event. End Events may define a "Result" that is a consequence of a Sequence Flow ending. A Start Event generates a Token that must eventually be consumed at an End Event. • A Start Event is OPTIONAL.	

- If a Process is complex and/or the starting conditions are not obvious, then it is RECOMMENDED that a Start Event be used.
- If there is an End Event, then there MUST be at least one Start Event.
- If the Start Event is used, then there MUST NOT be other flow elements that do not have incoming Sequence Flow—all other Flow Objects MUST be a target of at least one Sequence Flow.
- If the Start Event is not used, then all Flow Objects that do not have an incoming Sequence Flow (i.e., are not a target of a Sequence Flow) SHALL be instantiated when the Process is instantiated.
- There MAY be multiple Start Events for a given Process level.

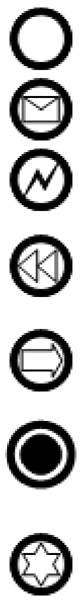
Start Event Triggers: There are many ways that can business process can be started (instantiated).

- None: The modeler does not display the type of Event.
- Message: A message arrives from a participant and triggers the start of the Process.
- Timer: A specific time-date or a specific cycle can be set that will trigger the start of the Process.
- Rule: This type of event is triggered when the conditions for a rule such as “Temperature above 300C” become true.
- Link: A Link is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another.
- Multiple: There are multiple ways of triggering the Process. Only one of them will be required to start the Process. The attributes of the Start Event will define which of the other types of Triggers apply.



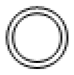
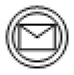
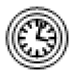
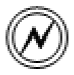
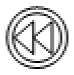
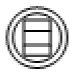
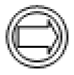
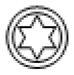
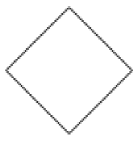
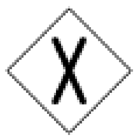
End Event Triggers: A BPMN modeller can define the consequence of reaching an End Event. This will be referred to as the End Event Result.

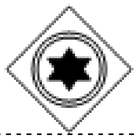
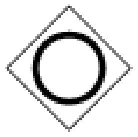
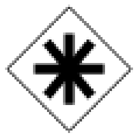
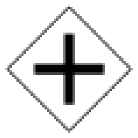
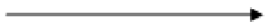
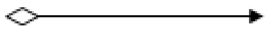
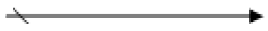
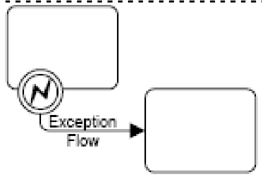
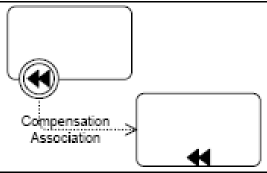
- None: The modeller does not display the type of Event.
- Message: This type of End indicates that a message is sent to a participant at the conclusion of the Process
- Error: Indicates that a named Error should be generated.
- Compensation: Indicates that a Compensation is necessary. The Compensation identifier will trigger an Intermediate Event when the Process is rolling back.
- Link: A Link is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another.
- Terminate: Indicates that all activities in the Process should be immediately ended. The Process is ended without compensation or event handling.
- This means that there are multiple consequences of ending the Process. All of them will occur. The attributes of the End Event will define which of the other types of Results apply.

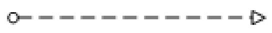
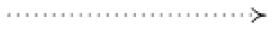


Intermediate Event Triggers: Intermediate Events occur between a Start Event and an End Event. They indicate the different ways that a Process may be interrupted or delayed after it has started. They will affect the flow of the process, but will not start or (directly) terminate the process. Intermediate Events can be used to:

1. Show where messages are expected or sent within the Process,

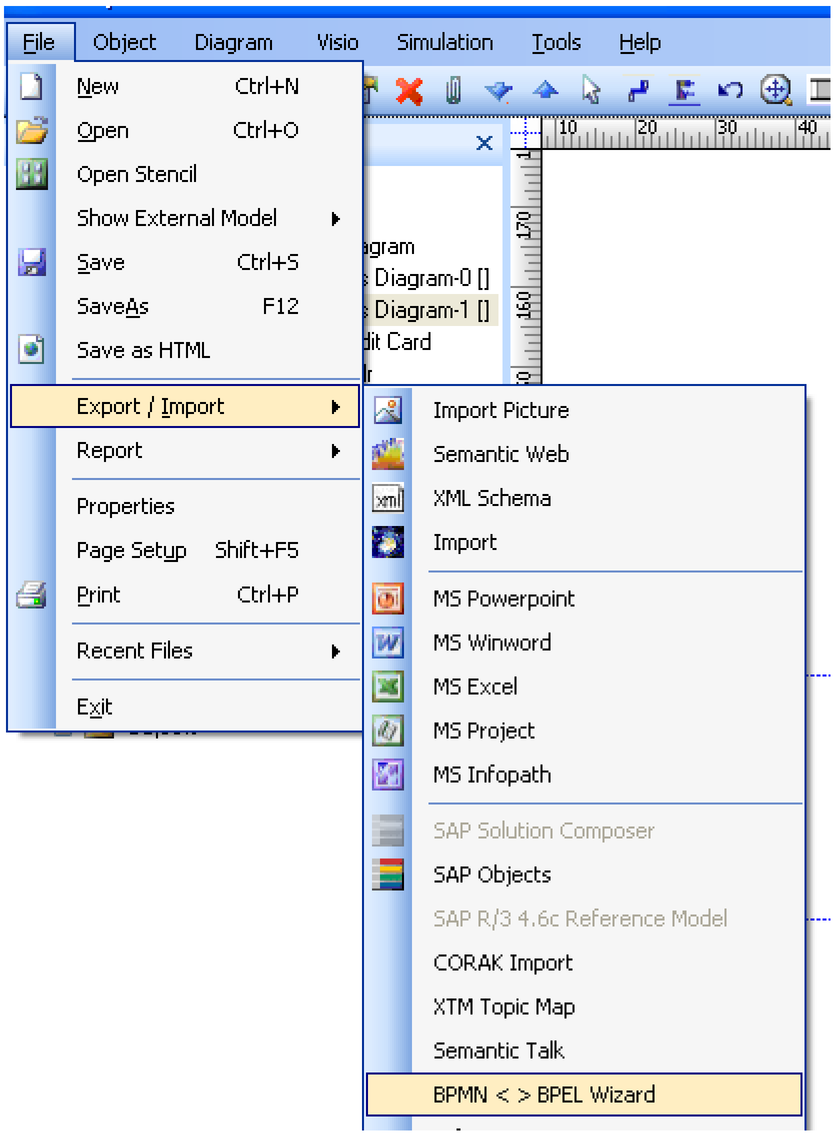
2. Show delays are expected within the Process, 3. Disrupt the Normal Flow through exception handling, or 4. Show the extra work required for compensation.	
• None: The modeler does not display the type of Event. It is used for modeling methodologies that use Events to indicate some change of state in the Process.	
• Message: A message arrives from a participant and triggers the Event. This causes the Process to continue if it was waiting for the message, or changes the flow for exception handling. In Normal Flow, Message Intermediate Events can be used for sending messages to a participant. If used for exception handling it will change the Normal Flow into an Exception Flow.	
• Timer: A specific time-date or a specific cycle can be set that will trigger the Event. If used within the main flow it acts as a delay mechanism. If used for exception handling it will change the Normal Flow into an Exception Flow.	
• Error: This is used for error handling. It sets (throws) an error if the Event is part of a Normal Flow. It reacts to (catches) a named error, or to any error if a name is not specified, when attached to the boundary of an activity.	
• Compensation: This is used for compensation handling. It call for compensation if the Event is part of a Normal Flow. It reacts to a named compensation call when attached to the boundary of an activity.	
• Rule: This is only used for exception handling. This type of event is triggered when a Rule becomes true. A Rule is an expression that evaluates some Process data.	
• Link: A Link is a mechanism for connecting an End Event (Result) of one Process to an Intermediate Event (Trigger) in another Process. Paired Intermediate Events can also be used as "GoTo" objects within a Process.	
• Multiple: This means that there are multiple ways of triggering the Event. Only one of them will be required. The attributes of the Intermediate Event will define which of the other types of Triggers apply.	
Gateway: Gateways are modeling elements that are used to control how Sequence Flow interact as they converge and diverge within a Process. It implies that there is a gating mechanism that either allows or disallows passage through the Gateway--that is, as Tokens arrive at a Gateway, they can be merged together on input and/or split apart on output as the Gateway mechanisms are invoked. Gateways can define all the types of Sequence Flow behavior: Decisions/branching (OR-Split; exclusive – XOR, inclusive – OR, and complex), merging (OR-Join), forking (AND-Split), and joining (AND-Join).	
Exclusive Gateways (XOR): The Exclusive Decision has two or more outgoing Sequence Flow, but only one of them may be taken during the performance of the Process. Thus, the Exclusive Decision defines a set of alternative paths for the Token to take as it traverses the Flow. There are two types of Exclusive Decisions: Data-Based and Event-Based.	
• **Data-Based**: The Process Flow will continue if one of the boolean (conditional) expressions of the outgoing Sequence Flow of the Gateway is True. These expressions use the values of process data to determine which path should be taken (hence the name Data-Based)	

• **Event-Based**: The event-based XOR gateway is used to model decisions based on events rather than process data. The process flow will continue if one of the specified events becomes true.	
Inclusive Gateways (OR): This Decision represents a branching point where Alternatives are based on conditional expressions contained within outgoing Sequence Flow. However, in this case, the True evaluation of one condition expression does not exclude the evaluation of other condition expressions. All Sequence Flow with a True evaluation will be traversed by a Token.	
Complex Gateways: BPMN includes a Complex Gateway to handle situations that are not easily handled through the other types of Gateways. Complex Gateways can also be used to combine a set of linked simple Gateways into a single, more compact situation. Modellers can provide complex expressions that determine the merging and/or splitting behaviour of the Gateway.	
Parallel Gateways (AND): Parallel Gateways provide a mechanism to synchronize parallel flow and to create parallel flow. These Gateways are not required to create parallel flow, but they can be used to clarify the behaviour of complex situations where a string of Gateways are used and parallel flow is required.	
Connecting Objects: There are three ways of connecting the Flow Objects to each other or other information. There are three Connecting Objects:	
Sequence flow: Sequence flow is used to show the order that activities will be performed in a Process. Normal Sequence Flow refers to the flow that originates from a Start Event and continues through activities via alternative and parallel paths until it ends at an End Event.	
• **Uncontrolled flow**: Uncontrolled flow refers to flow that is not affected by any conditions or does not pass through a Gateway. The simplest example of this is a single Sequence flow connecting two activities. This can also apply to multiple Sequence flow that converge on or diverge from a task. For each uncontrolled Sequence flow a "Token" will flow from the source object to the target object.	
• **Conditional flow:** Sequence flow can have condition expressions that are evaluated at runtime to determine whether or not the flow will be used. If the conditional flow is outgoing from a task, then the Sequence Flow will have a mini-diamond at the beginning of the line. If the conditional flow is outgoing from a Gateway, then the line will not have a mini-diamond	
• **Default flow**: For Data-Based Exclusive Decisions or Inclusive Decisions, one type of flow is the Default condition flow. This flow will be used only if all the other outgoing conditional flows are not true at runtime. These Sequence flow will have a diagonal slash will be added to the beginning of the line.	
Exception flow: An Exception flow occurs outside the Normal Flow of the Process and is based upon an Intermediate Event that occurs during the performance of the Process.	
Compensation Association: Compensation Association occurs outside the Normal Flow of the Process and is based upon an event (a Cancel Intermediate Event) that is triggered through the failure of a Transaction or a Compensate Event. The target of the Association must be marked as a Compensation Activity.	

Message flow: A Message flow is used to show the flow of messages between two entities that are prepared to send and receive them. In BPMN, two separate Lanes in the Diagram will represent the two entities (Participants).	
Association: An Association is used to associate information with Flow Objects. (e.g. Data Object)	
Lane: It is Participant, i.e. a entity (e.g. a company, institution, or a customer) or a role (e.g. an expert, a buyer or a seller) , which controls or is responsible for a business process. In SemTalk we have extended this concept to systems. i.e. in case that activities are carried out automatically a system or if the communication between participants take place between systems a participant can represent an information system (e.g. Web Services)	

8 Generating BPEL for BizTalk by Example:

The BPEL4WS export is at File->Export / Import -> BPEL4WS Export.



Please be aware that not any arbitrary process graph can be converted to a structured program. The following examples show some cases or “design patterns”, which are recognized.

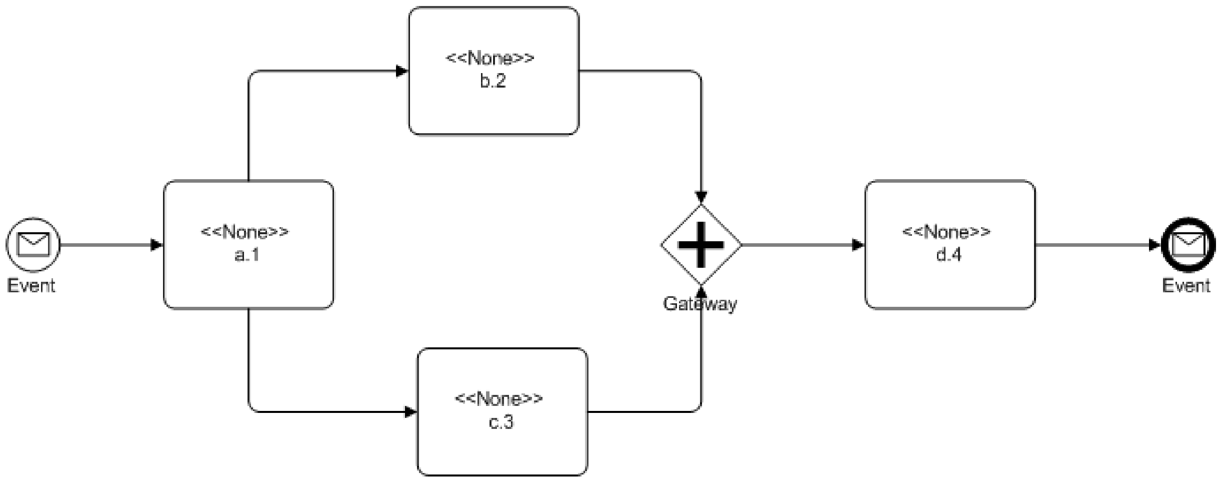
It is highly recommend reviewing the generated BPEL carefully. Export and test your BPMN process to BizTalk frequently because it may be hard to find out why BizTalk does not import it.

All BPEL code shown in these examples has been tested with BizTalk. BizTalk's BPEL Import may stop/break without giving a hint on arbitrary BPEL files.

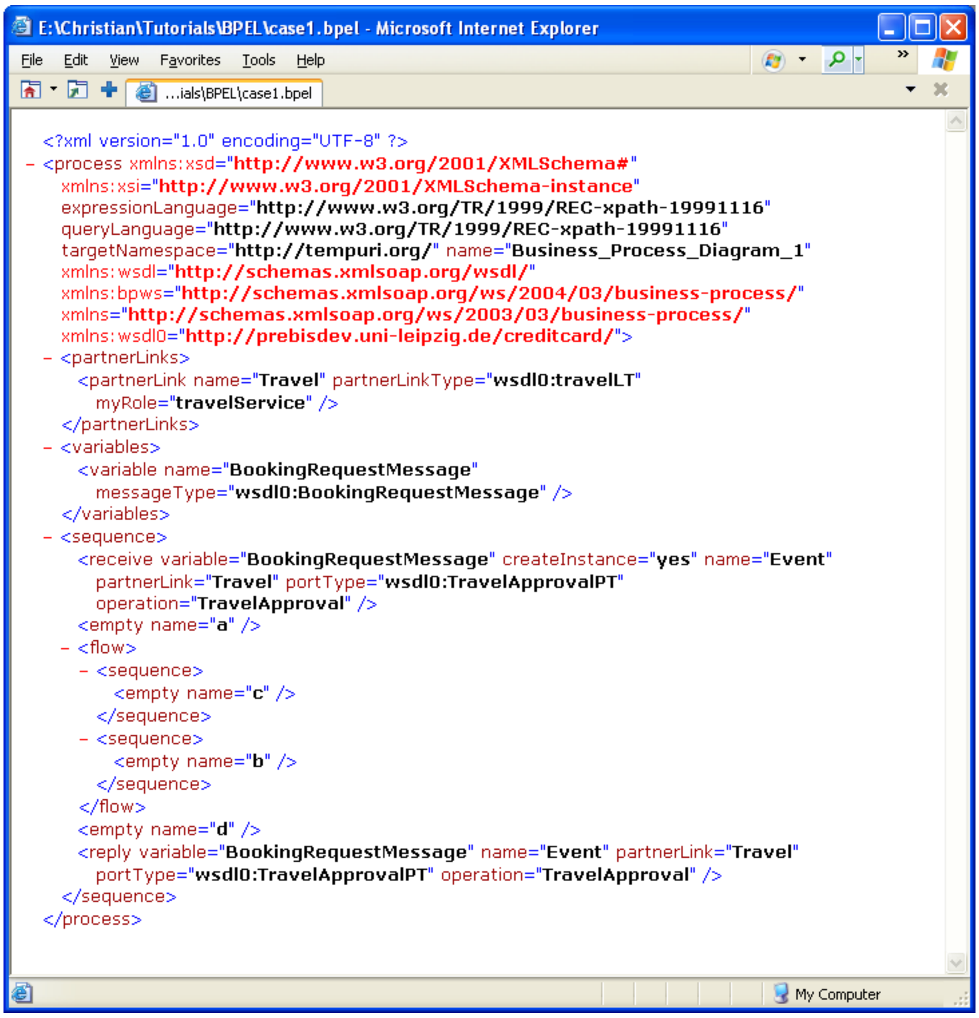
WSDL files generated with Visual Studio usually can be used with BizTalk. Others may need some additional adjustments.

8.1 Parallel process with synchronization

A simple parallel process with synchronization. We have used travel service wsdl for the input and output events.

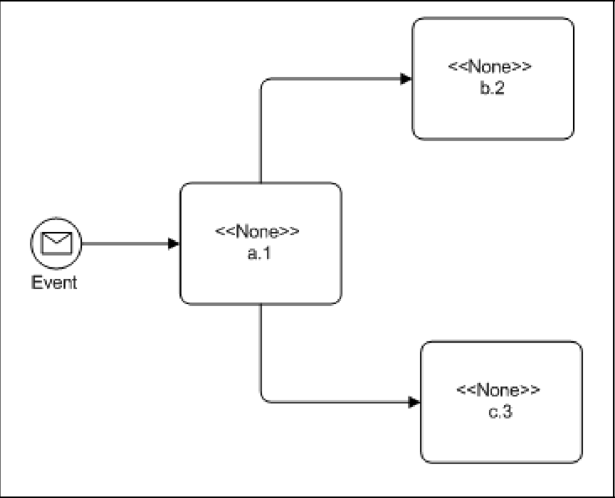


Resulting BPEL:


E:\Christian\Tutorials\BPEL\case1.bpel - Microsoft Internet Explorer
File Edit View Favorites Tools Help
...ials\BPEL\case1.bpel
My Computer

```
<?xml version="1.0" encoding="UTF-8" ?>
- <process xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
    queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
    targetNamespace="http://tempuri.org/" name="Business_Process_Diagram_1"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:bpws="http://schemas.xmlsoap.org/ws/2004/03/business-process/"
    xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
    xmlns:wsdl0="http://prebisdev.uni-leipzig.de/creditcard/">
  - <partnerLinks>
      <partnerLink name="Travel" partnerLinkType="wsdl0:travelLT"
        myRole="travelService" />
    </partnerLinks>
  - <variables>
      <variable name="BookingRequestMessage"
        messageType="wsdl0:BookingRequestMessage" />
    </variables>
  - <sequence>
      <receive variable="BookingRequestMessage" createInstance="yes" name="Event"
        partnerLink="Travel" portType="wsdl0:TravelApprovalPT"
        operation="TravelApproval" />
      <empty name="a" />
    - <flow>
      - <sequence>
          <empty name="c" />
        </sequence>
      - <sequence>
          <empty name="b" />
        </sequence>
      </flow>
      <empty name="d" />
      <reply variable="BookingRequestMessage" name="Event" partnerLink="Travel"
        portType="wsdl0:TravelApprovalPT" operation="TravelApproval" />
    </sequence>
</process>
```

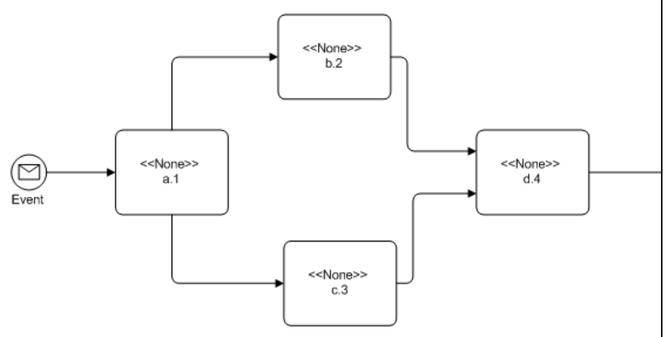
8.2 Open Parallelization without synchronization



```
- <sequence>
    <receive variable="BookingRe
      partnerLink="Travel" portTy
      operation="TravelApproval
    <empty name="a" />
  - <flow>
    - <sequence>
        <empty name="c" />
      </sequence>
    - <sequence>
        <empty name="b" />
      </sequence>
    </flow>
  </sequence>
```

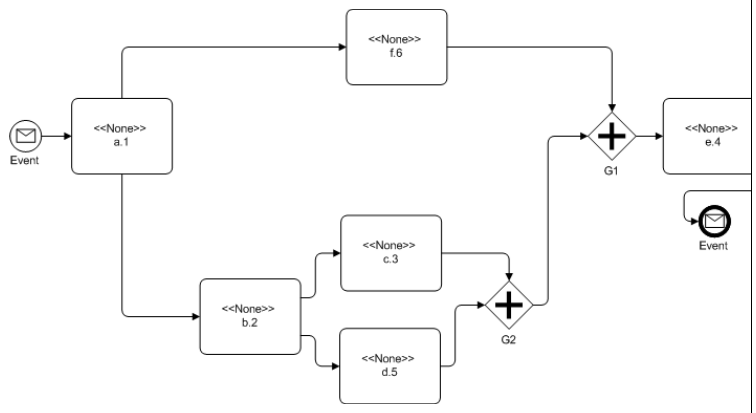
8.3 Closed Parallelization without synchronization

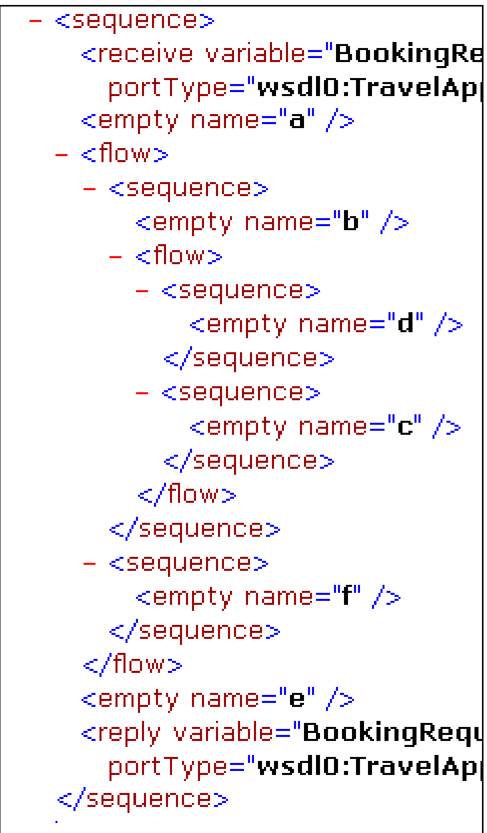
Parallelization without synchronization. “d” is executed after “b” and not after “c”. This usually not, what the user has intended.



```
- <sequence>
    <receive variable="BookingRequest
      partnerLink="Travel" portType="w
      operation="TravelApproval" />
    <empty name="a" />
  - <flow>
    - <sequence>
        <empty name="c" />
      </sequence>
    - <sequence>
        <empty name="b" />
        <empty name="d" />
        <reply variable="BookingReque
          partnerLink="Travel" portType
          operation="TravelApproval" /
      </sequence>
    </flow>
  </sequence>
```

8.4 Nested Split / Join



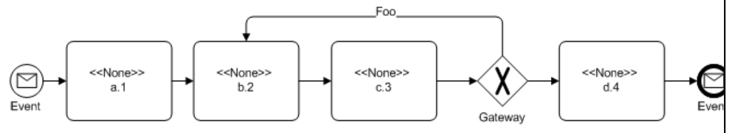


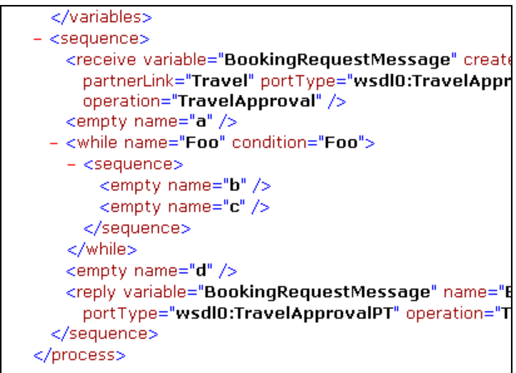
```
- <sequence>
    <receive variable="BookingRe
      portType="wsdl0:TravelAp
    <empty name="a" />
  - <flow>
    - <sequence>
        <empty name="b" />
      - <flow>
        - <sequence>
            <empty name="d" />
          </sequence>
        - <sequence>
            <empty name="c" />
          </sequence>
        </flow>
      </sequence>
    - <sequence>
        <empty name="f" />
      </sequence>
    </flow>
    <empty name="e" />
    <reply variable="BookingRequ
      portType="wsdl0:TravelAp
  </sequence>
```

8.5 While

A simple “While” - loop

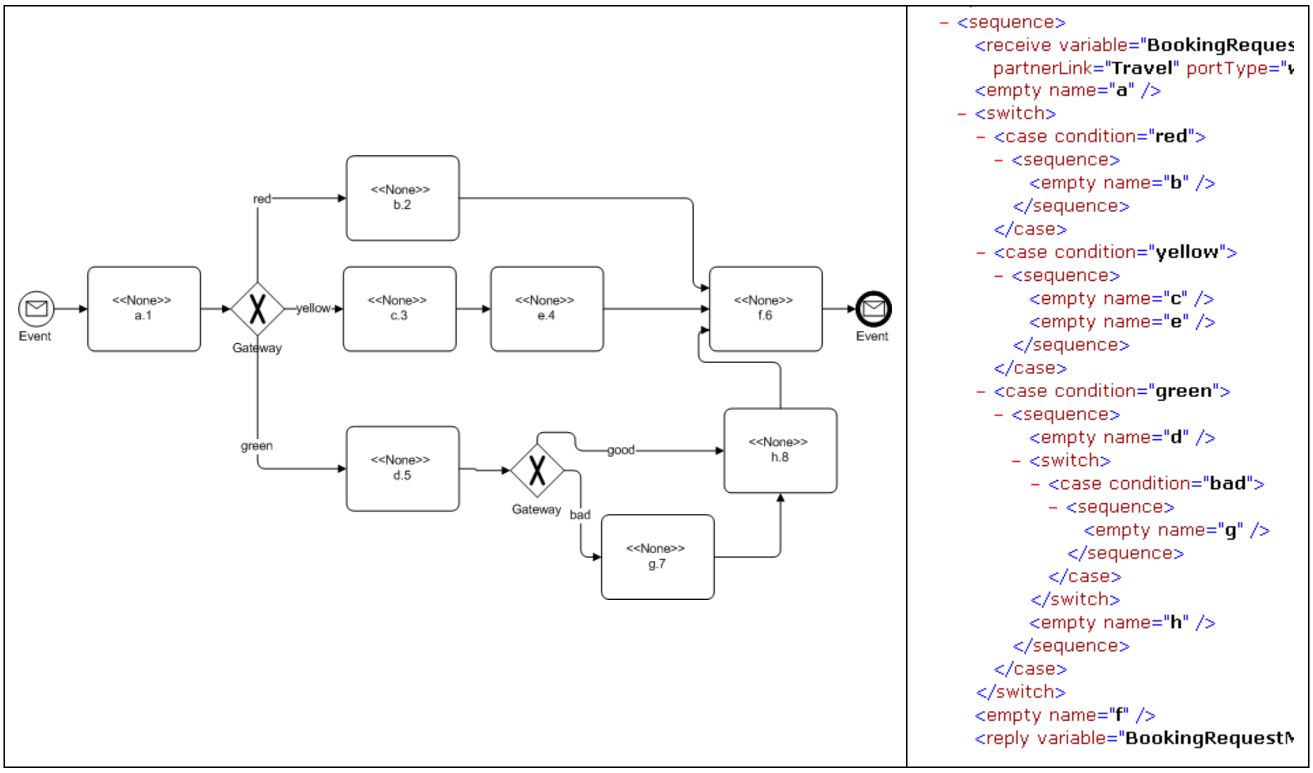
Do not add any activities on the “loop” back sequence.





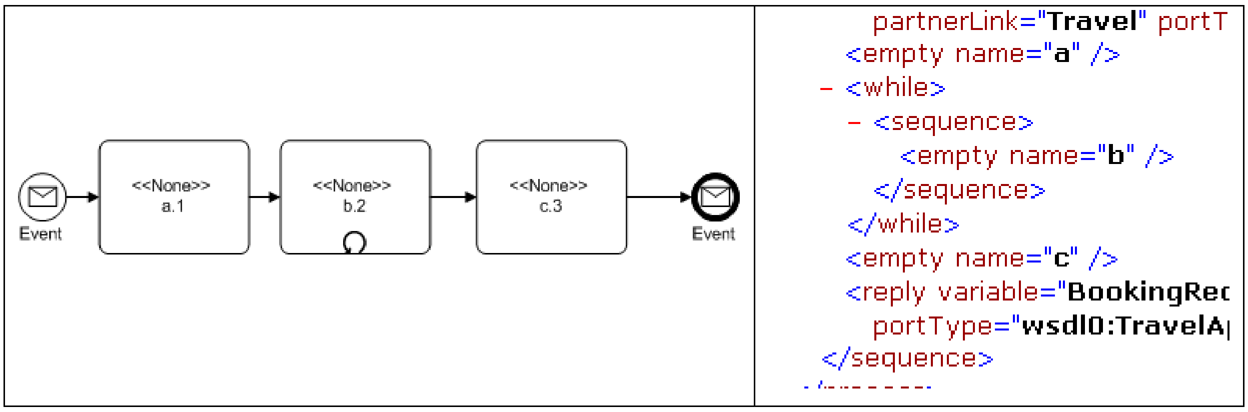
```
  </variables>
- <sequence>
    <receive variable="BookingRequestMessage" create
      partnerLink="Travel" portType="wsdl0:TravelAppr
      operation="TravelApproval" />
    <empty name="a" />
  - <while name="Foo" condition="Foo">
    - <sequence>
        <empty name="b" />
        <empty name="c" />
      </sequence>
    </while>
    <empty name="d" />
    <reply variable="BookingRequestMessage" name="E
      portType="wsdl0:TravelApprovalPT" operation="T
  </sequence>
</process>
```

8.6 Nested (closed) Switches



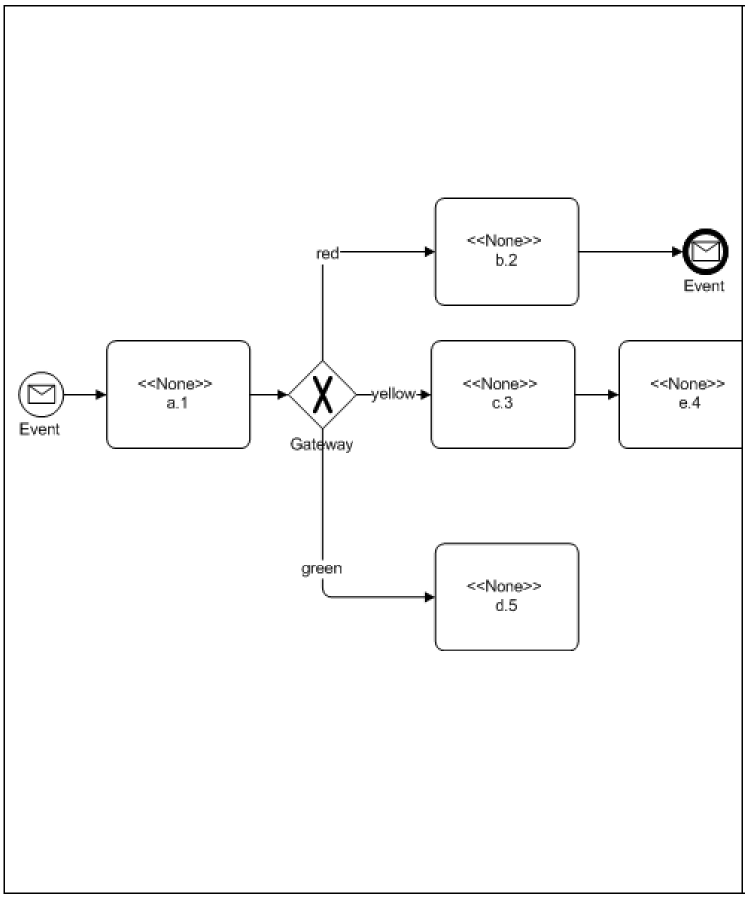
```
– <sequence>
    <receive variable="BookingReques
      partnerLink="Travel" portType="v
    <empty name="a" />
  – <switch>
    – <case condition="red">
      – <sequence>
          <empty name="b" />
        </sequence>
      </case>
    – <case condition="yellow">
      – <sequence>
          <empty name="c" />
          <empty name="e" />
        </sequence>
      </case>
    – <case condition="green">
      – <sequence>
          <empty name="d" />
        – <switch>
          – <case condition="bad">
            – <sequence>
                <empty name="g" />
              </sequence>
            </case>
          </switch>
          <empty name="h" />
        </sequence>
      </case>
    </switch>
    <empty name="f" />
    <reply variable="BookingRequestN
```

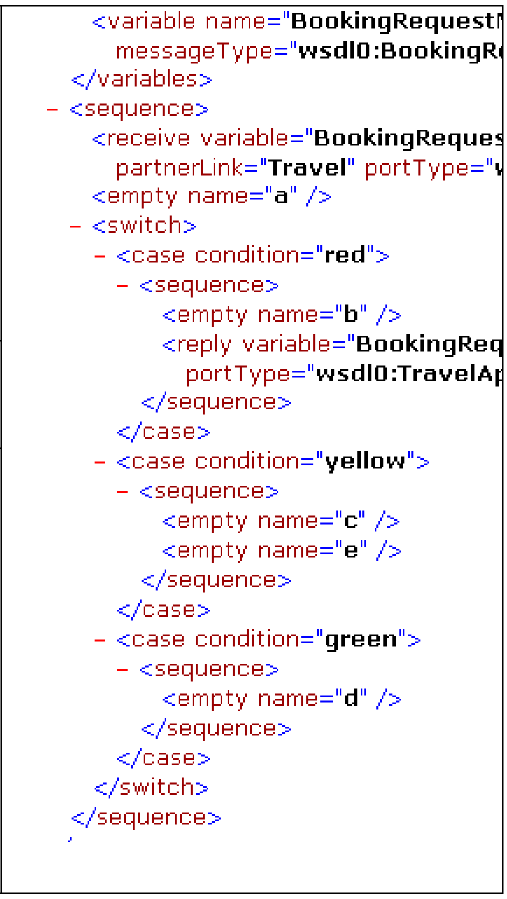
8.7 Simple Loop



```
    partnerLink="Travel" portT
  <empty name="a" />
– <while>
  – <sequence>
      <empty name="b" />
    </sequence>
  </while>
  <empty name="c" />
  <reply variable="BookingRec
    portType="wsdl0:TravelA
</sequence>
```

8.8 Open (not closed) Switches

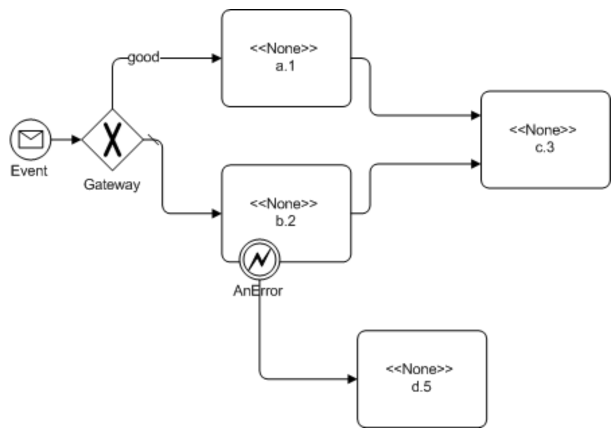




```
    <variable name="BookingRequest
      messageType="wsdl0:BookingR
  </variables>
- <sequence>
    <receive variable="BookingReques
      partnerLink="Travel" portType="
    <empty name="a" />
  - <switch>
    - <case condition="red">
      - <sequence>
          <empty name="b" />
          <reply variable="BookingReq
            portType="wsdl0:TravelA
        </sequence>
      </case>
    - <case condition="yellow">
      - <sequence>
          <empty name="c" />
          <empty name="e" />
        </sequence>
      </case>
    - <case condition="green">
      - <sequence>
          <empty name="d" />
        </sequence>
      </case>
    </switch>
  </sequence>
```

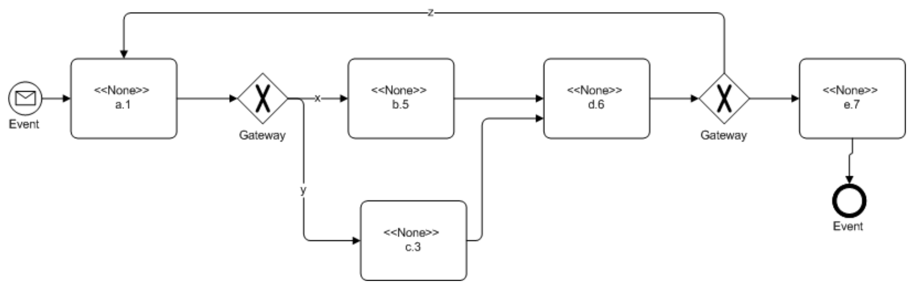
8.9 Switch with Default condition

A Switch with Default condition and a process error.



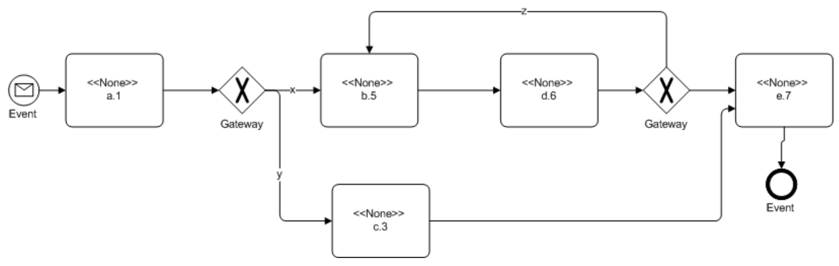
```
- <sequence>
    <receive variable="BookingRequestMess
      partnerLink="Travel" portType="wsdl0:T
  - <switch>
    - <case condition="good">
      - <sequence>
          <empty name="a" />
        </sequence>
      </case>
    - <otherwise>
      - <sequence>
        - <scope name="b">
          - <faultHandlers>
            - <catch faultName="AnError">
              - <sequence>
                  <empty name="d" />
                </sequence>
              </catch>
            </faultHandlers>
          - <sequence>
              <empty name="b" />
            </sequence>
          </scope>
        </sequence>
      </otherwise>
    </switch>
    <empty name="c" />
  </sequence>
```

8.10 Switch nested in a While



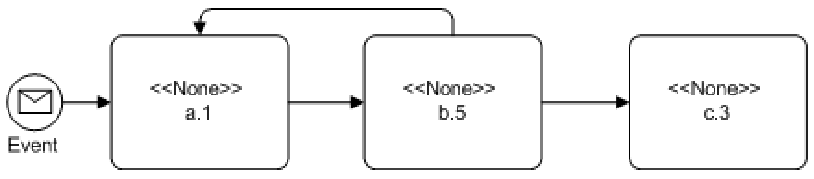
```
- <sequence>
    <receive variable="BookingReq
      portType="wsdl0:TravelAppl
  - <while name="z" condition="z">
    - <sequence>
        <empty name="a" />
      - <switch>
        - <case condition="x">
          - <sequence>
              <empty name="b" />
            </sequence>
          </case>
        - <case condition="y">
          - <sequence>
              <empty name="c" />
            </sequence>
          </case>
        </switch>
        <empty name="d" />
      </sequence>
    </while>
    <empty name="e" />
    <empty name="Event" />
  </sequence>
</process>
```

8.11 While nested in a Switch



```
- <sequence>
    <receive variable="BookingRequestN
      portType="wsdl0:TravelApprovalP
    <empty name="a" />
  - <switch>
    - <case condition="x">
      - <sequence>
        - <while name="z" condition="z">
          - <sequence>
              <empty name="b" />
              <empty name="d" />
            </sequence>
          </while>
        </sequence>
      </case>
    - <case condition="y">
      - <sequence>
          <empty name="c" />
        </sequence>
      </case>
    </switch>
    <empty name="e" />
    <empty name="Event" />
  </sequence>
</process>
```

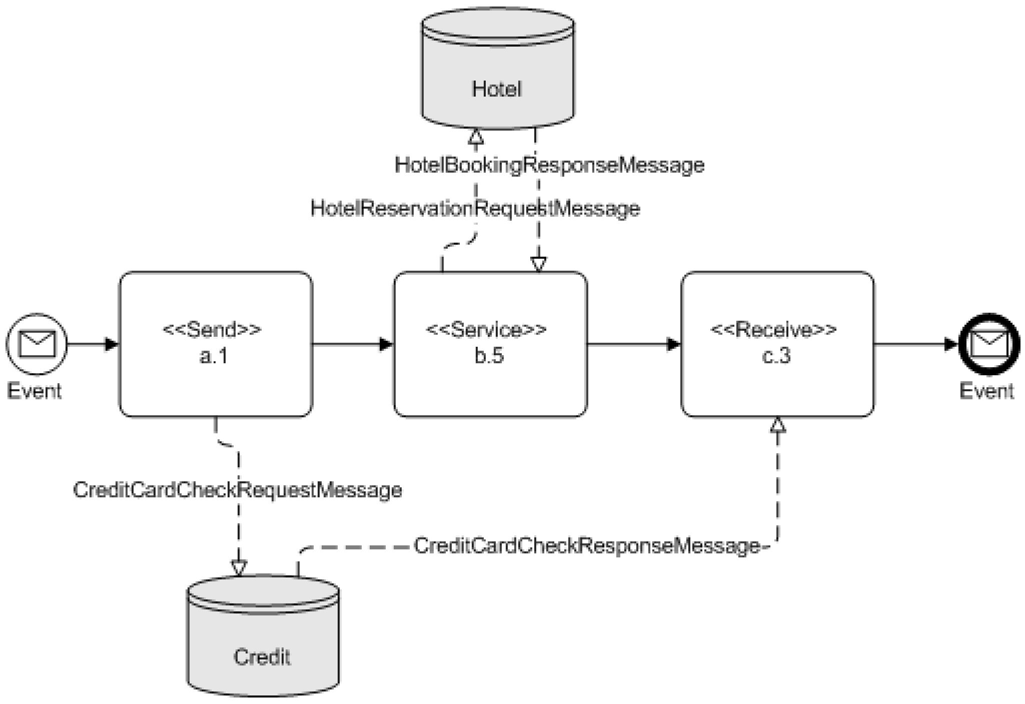
8.12 Loops without Switch

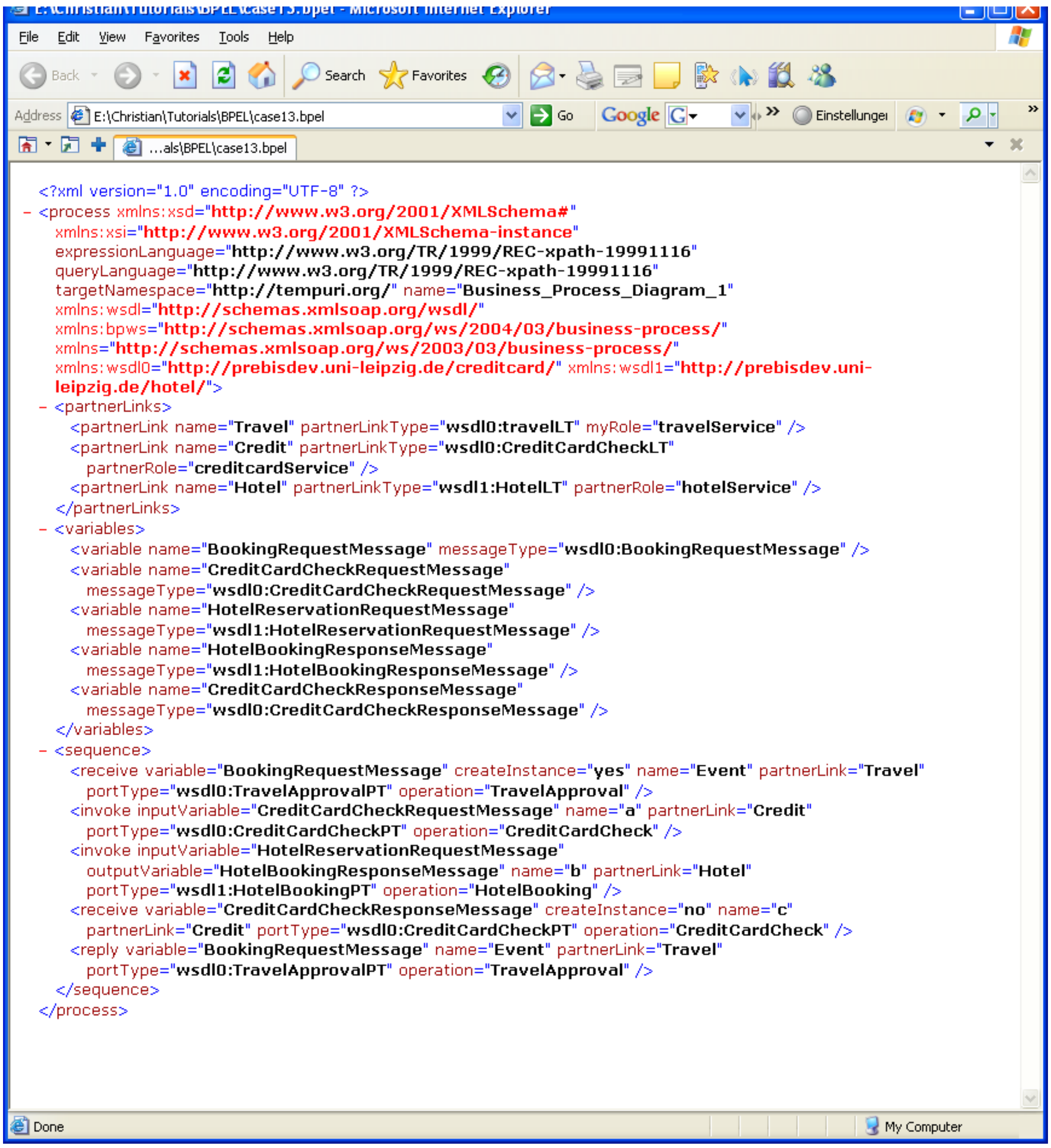


```
- <sequence>
    <receive variable="Bookir
      portType="wsdl0:Trave
    <empty name="a" />
    <empty name="b" />
  - <flow>
    - <sequence>
        <empty name="c" />
      </sequence>
    </flow>
  </sequence>
</process>
```

Loops without Switch can not be compiled to BPEL!

8.13 Calling Web Services


Hotel
HotelBookingResponseMessage
HotelReservationRequestMessage
Event
<<Send>>
a.1
<<Service>>
b.5
<<Receive>>
c.3
Event
CreditCardCheckRequestMessage
CreditCardCheckResponseMessage
Credit


File Edit View Favorites Tools Help
Back Search Favorites
Address E:\Christian\Tutorials\BPEL\case13.bpel Go
...als\BPEL\case13.bpel
Done
My Computer

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <process xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
    queryLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
    targetNamespace="http://tempuri.org/" name="Business_Process_Diagram_1"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:bpws="http://schemas.xmlsoap.org/ws/2004/03/business-process/"
    xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
    xmlns:wsdl0="http://prebisdev.uni-leipzig.de/creditcard/" xmlns:wsdl1="http://prebisdev.uni-
    leipzig.de/hotel/">
  - <partnerLinks>
      <partnerLink name="Travel" partnerLinkType="wsdl0:travelLT" myRole="travelService" />
      <partnerLink name="Credit" partnerLinkType="wsdl0:CreditCardCheckLT"
        partnerRole="creditcardService" />
      <partnerLink name="Hotel" partnerLinkType="wsdl1:HotelLT" partnerRole="hotelService" />
    </partnerLinks>
  - <variables>
      <variable name="BookingRequestMessage" messageType="wsdl0:BookingRequestMessage" />
      <variable name="CreditCardCheckRequestMessage"
        messageType="wsdl0:CreditCardCheckRequestMessage" />
      <variable name="HotelReservationRequestMessage"
        messageType="wsdl1:HotelReservationRequestMessage" />
      <variable name="HotelBookingResponseMessage"
        messageType="wsdl1:HotelBookingResponseMessage" />
      <variable name="CreditCardCheckResponseMessage"
        messageType="wsdl0:CreditCardCheckResponseMessage" />
    </variables>
  - <sequence>
      <receive variable="BookingRequestMessage" createInstance="yes" name="Event" partnerLink="Travel"
        portType="wsdl0:TravelApprovalPT" operation="TravelApproval" />
      <invoke inputVariable="CreditCardCheckRequestMessage" name="a" partnerLink="Credit"
        portType="wsdl0:CreditCardCheckPT" operation="CreditCardCheck" />
      <invoke inputVariable="HotelReservationRequestMessage"
        outputVariable="HotelBookingResponseMessage" name="b" partnerLink="Hotel"
        portType="wsdl1:HotelBookingPT" operation="HotelBooking" />
      <receive variable="CreditCardCheckResponseMessage" createInstance="no" name="c"
        partnerLink="Credit" portType="wsdl0:CreditCardCheckPT" operation="CreditCardCheck" />
      <reply variable="BookingRequestMessage" name="Event" partnerLink="Travel"
        portType="wsdl0:TravelApprovalPT" operation="TravelApproval" />
    </sequence>
  </process>
```