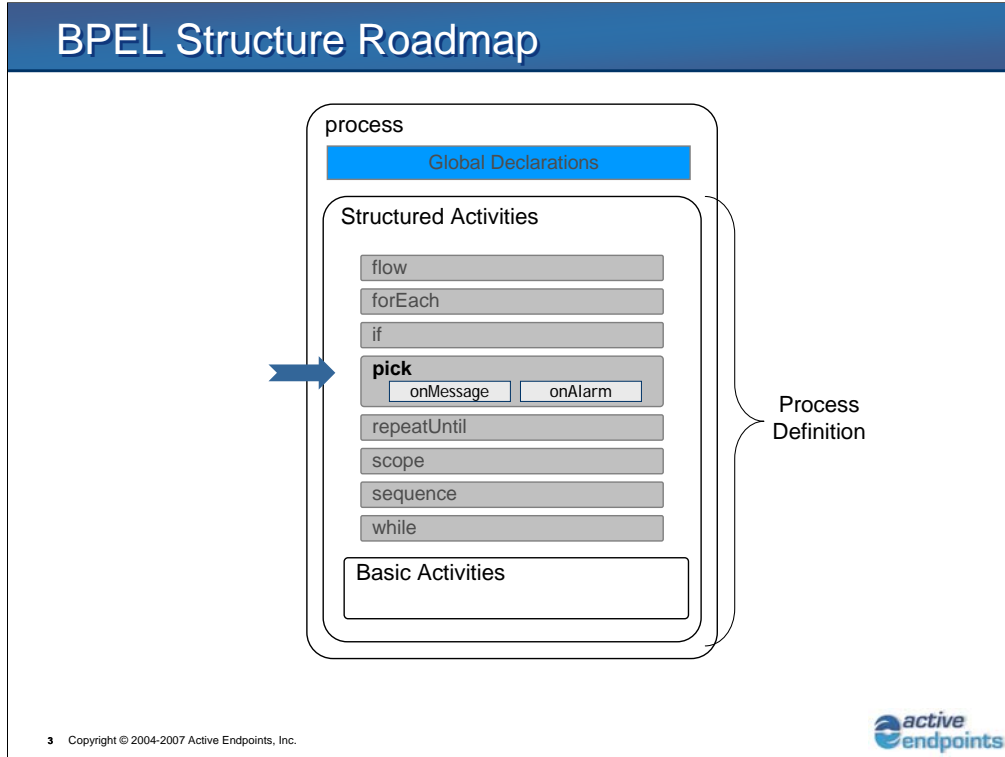


This is Unit #18 of the BPEL Fundamentals course. In past Units we've looked at ActiveBPEL Designer, Workspaces and Projects, created the Process itself and then declared our Imports, PartnerLinks and Variables and then we created Interaction Activities in various ways. Next, we looked at the Sequence activity, Assignments and Copies and after that we studied Correlation, Scopes and Fault Handling. Then, we examined Compensation, Event Handling, Termination Handlers and the If activity, which allows us to do conditional processing and finally at the rest of the BPEL Basic activities. In the last Unit we looked at BPEL's Flow activity, and in this Unit we'll look at the Pick Activity.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - pick activity



A Pick is a structured BPEL activity and is part of our process definition, and a Pick activity can contain one or more OnMessage and OnAlarm activities.

pick Overview

- Used to have the process wait until one of a set of events is triggered
 - Message events via `onMessage` element
 - Alarm events via `onAlarm` element
- Plays a role in the lifecycle of a business process
 - If `createInstance="yes"`
 - Instructs the BPEL engine to create a new process instance
 - As a result of receiving one of a set of possible messages
 - Each `onMessage` is equivalent to a `receive` activity with the `createInstance="yes"`
 - No alarms are permitted in this case

4 Copyright © 2004-2007 Active Endpoints, Inc.



The Pick activity forces the process to wait until one of a set of events is triggered. All of these events are either `onMessage` elements or `onAlarm` elements. You can have as many `onMessage` and `onAlarm` activities as you want, but exactly one of them will be executed. Once one event is executed, all others are disabled. A Pick Activity can create an instance in response to an `onEvent`, making it equivalent to a `Receive` activity, but no `onAlarm` events are permitted in the Pick if doing so. Note that it is the Pick that is creating the instance, not the `onMessage`.

pick Activity Syntax

```

<pick createInstance="yes|no"? standard-attributes>
  standard-elements
  <onMessage partnerLink="ncname" portType="qname"
    operation="ncname" variable="ncname"?
    messageExchange="ncname"?>+
    <correlations?>
      <correlation set="ncname" initiate="yes|no"?>+
    </correlations>
    <fromParts?>
      <fromPart part="ncname"
        toVariable="BPELVariableName" />+
    </fromParts>
    activity
  </onMessage>
  <onAlarm>*
  (
    <for expressionLanguage="anyURI">duration-expr</for>
    |
    <until expressionLanguage="anyURI">deadline-expr</until>
  )
  activity
  </onAlarm>
</pick>

```

5 Copyright © 2004-2007 Active Endpoints, Inc.



Here is the syntax of the Pick activity. First, the attribute for “createInstance” is set to either yes or no, and then we have all the standard attributes and elements. Then we have the optional onMessage definitions, which require the PartnerLink, PortType, Operation and the optional variable and optional messageExchange. Following that, we have the Correlation Sets, with the initiate attribute choices of yes/no and join, because the Pick is part of our partner conversations, so they have to be correlated. Then we have the From parts with their variable, followed by the primary activity of the onMessage element. Finally, we have our onAlarm activity, which has the expression language and the expression itself for our deadline or duration elements, followed by the onAlarm’s primary activity.

onMessage Overview and Syntax

- Used to receive exactly one of several messages into a process
 - Uses many of the same attributes as the receive activity

```
<onMessage partnerLink="ncname" portType="qname"
  operation="ncname" variable="ncname"?
  messageExchange="ncname" ?>
  <correlations?>
    <correlation set="ncname" initiate="yes|no|join"?>+
  </correlations>
  <fromParts?>
    <fromPart part="ncname"
      toVariable="BPELVariableName" />+
  </fromParts>
  activity
</onMessage>
```

6 Copyright © 2004-2007 Active Endpoints, Inc.



Here we have the syntax for the onMessage element used in a Pick activity, and you'll notice that it is very similar to a Receive activity, in that its job is to wait for a message to arrive.

onMessage Semantics

- Represents an event that waits for a message to arrive
 - When the message arrives, the primary activity specified in the corresponding handler is performed
- The attributes and semantics are the same as the attributes and semantics of the **receive** activity except
 - An `onMessage` can not specify the `createInstance` attribute

7 Copyright © 2004-2007 Active Endpoints, Inc.



The `onMessage` element of the *Pick* activity waits for the arrival of a specific message and then fires the appropriate activity. The attributes and semantics are the same as for the *Receive* activity, except it cannot create an instance. Note that the attribute for “`createInstance=yes`” is on the *Pick* activity, not on the *onMessage* *element* of that activity.

onAlarm Overview and Syntax

- Used to make the process time-aware
 - Equivalent to the behavior of a `wait` activity
- An alarm event can either be
 - *For* a certain period of time
 - Duration-valued expression
 - *Until* a certain deadline is reached
 - Deadline-valued expression

```
<onAlarm>*  
  (  
    <for expressionLanguage="anyURI">duration-expr</for>  
    |  
    <until expressionLanguage="anyURI">deadline-expr</until>  
  )  
  activity  
</onAlarm>
```

© Copyright © 2004-2007 Active Endpoints, Inc.



Here is the syntax for the `onAlarm` element of the `Pick` activity. The `onAlarm` element (much like the `onAlarm` element of the `OnEvent` activity) makes a process time aware by linking execution to a specific deadline or duration, either of which is in the form of an expression.

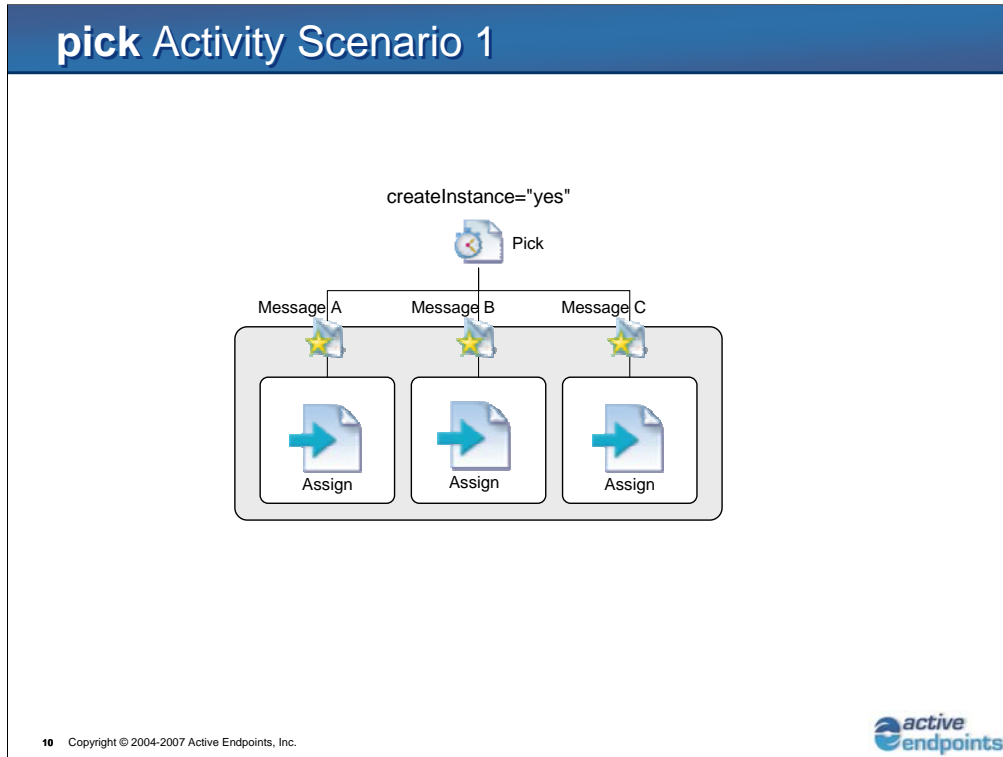
onAlarm Semantics

- For a duration-based **onAlarm** event
 - Counting of time starts at the point in time when the `pick` activity starts
 - An alarm event goes off when the specified time or duration has been reached

© Copyright © 2004-2007 Active Endpoints, Inc.



The firing of the Pick activity starts the “clock,” with the onAlarm element waiting on a specific time or until a certain amount of time has passed (a duration.)



Now, let's take a look at an example that uses the Pick activity. Here we have a Pick Activity with the "createInstance" attribute set to "yes." Note that we have multiple onMessage choices, but no "onAlarm" because we are using "createInstance." Once one of the three onMessages is received, we'll perform one of the three Assign activities, as appropriate.

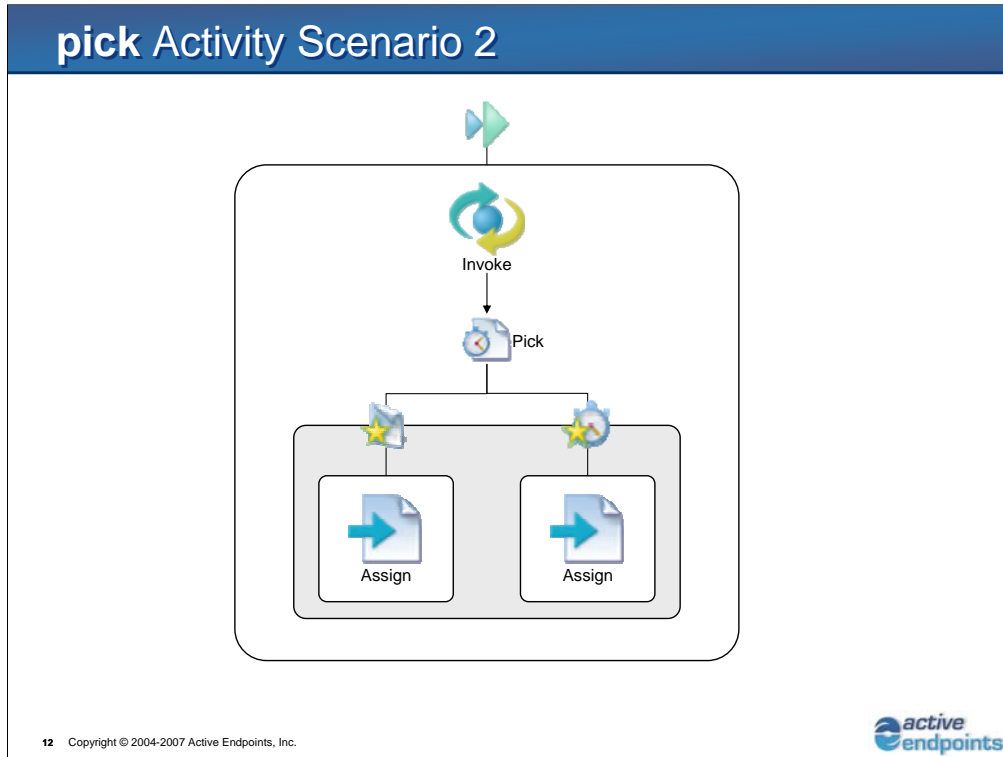
pick Activity Example 1

```
<pick createInstance="yes">
  <onMessage partnerLink="CustMsgA" ... >
    <assign .../>
  </onMessage>
  <onMessage partnerLink="CustMsgB" ... >
    <assign .../>
  </onMessage>
  <onMessage partnerLink="CustMsgC" ... >
    <assign .../>
  </onMessage>
</pick>
```

11 Copyright © 2004-2007 Active Endpoints, Inc.



Here is the syntax for the previous example. Note that the attribute for “createInstance” is set on the Pick activity itself, not on the onMessage element of that activity.



Here is another scenario where we have a Pick activity inside a Sequence that contains both an onMessage (L) and an onAlarm (R). This construction essentially puts a timer on the receipt of a message. The process will wait for the *specific* incoming message while the alarm clock is ticking. If the onAlarm goes off before the onMessage arrives, the onAlarm will do its Assign, but if the onMessage arrives before the alarm goes off, then it will execute its own Assign. In either case, only one of the two will be executed, while the other will be disabled and will not execute.

pick Activity Example 2

```
<sequence>
  <invoke partnerLink="Customer" portType="AskPT"
    operation="AskForResponse" ... />
  <pick>
    <onMessage partnerLink="Customer" portType="ResponsePT"
      operation="ReceiveResponse" ...>
      <assign .../>
    </onMessage>
    <onAlarm>
      <for>'PT5H'</for>
      <!-- Did not receive response within 5 hours -->
      <assign .../>
    </onAlarm>
  </pick>
</sequence>
```

13 Copyright © 2004-2007 Active Endpoints, Inc.



Here is a second example of the Pick activity's syntax. We have a Pick that is designed to wait for a response, so if we get the Message "ReceiveResponse" in < 5 hrs. we'll execute the onMessage element. If we don't get the message within 5 hrs, we will execute the onAlarm element of the activity. Only one element of a Pick activity will execute, no matter when the message arrives.

pick Semantics

- Only one of the events defined will be executed
 - The first event to occur
- Must have at least one or more **onMessage** events
 - Optionally have one or more `onAlarm` events
- Can be used to create a process instance
 - The create instance attribute is defined on a pick element
 - No `onAlarms` are permitted in this case

14 Copyright © 2004-2007 Active Endpoints, Inc.



Now a quick review of the Pick activity's semantics. Only one of the activity's event elements will execute. The Pick must have at least one, and can have more than one, `onMessage` events. The Pick can have one or more `onAlarm` events. If your configuration of the Pick has "createInstance" set to Yes, then no `onAlarm` elements allowed.

Lab 13 – pick Activity

- Overview of Lab Exercises
 - Add an `invoke` for AskCustomer service
 - Use a `pick` activity to either
 - Wait for a response from customer or
 - Wait until a certain amount of time has elapsed

15 Copyright © 2004-2007 Active Endpoints, Inc.



The next Lab in the BPEL Fundamentals class is Lab #13. (Note: This is lab #5 if you are taking BPEL Fundamentals II.) In this lab we will add an Invoke activity that will implement the "AskCustomer" service. This service will ask the customer whether or not they would like to receive a partial order. Note that this Pick activity will only apply to those customers who had previously indicated (in their initial order) that they'd like to be asked whether or not they'd accept such an order. So we'll create a Pick that executes the `onMessage` or the `onAlarm`, based upon their response – or non-response, in the case of the `onAlarm` - to a request.

Unit Objectives

- Now you are familiar with:
 - pick activity