



This is Unit #6 of the BPEL Fundamentals course. In past Units we've looked at ActiveVOS Designer, Workspaces and Projects and at the Process itself. Then we looked at the Global Declarations used by a process. So now, let's take a look at the Interaction Activities that define the process.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - Overview of interaction activities
 - receive activity
 - reply activity
 - Creating interaction activities using the Operation wizard
 - invoke activity
 - Creating interaction activities using the Interfaces View and the Activity Palette

2 Copyright © 2004-2008 Active Endpoints, Inc.



In this unit we'll start by looking at the BPEL language's Receive and Reply activities and at the Designer's Operation Wizard. This will give us what we need to complete the next set of labs. Then, after the labs are completed, we'll move on to the Invoke activity, and learn more ways to create activities.

Interaction Activities Overview

- Primary way for a process to
 - Invoke operations on partners
 - Receive and optionally reply to invocation requests from partners
- Comprised of the following activities
 - Basic
 - receive
 - reply
 - invoke
 - Structured
 - onMessage (pick)
 - onEvent (eventHandler)
- Uses the operations defined for a **portType** associated through a particular **partnerLink**
 - Variables that are defined using `messageTypes` are used as input or output targets

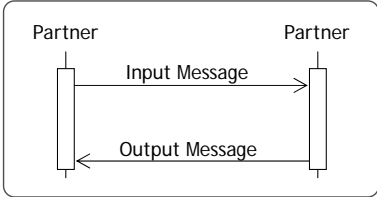
3 Copyright © 2004-2008 Active Endpoints, Inc.



Interaction activities are the means by which we invoke partner services and by which we allow our partners to invoke the services our process offers to them. These activities encompass both the basic and structured interaction activities defined by BPEL - as shown here - and these activities use the Operations defined for their particular PortType.

Supported Message Exchange Patterns

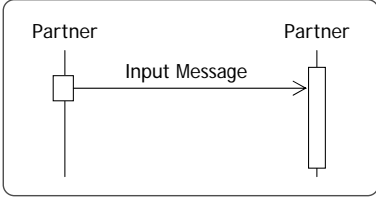
■ request-response



WSDL Fragment

```
<portType name="...">
  <operation name="...">
    <input message="..." />
    <output message="..." />
  </operation>
</portType>
```


■ one-way



WSDL Fragment

```
<portType name="...">
  <operation name="...">
    <input message="..." />
  </operation>
</portType>
```

Note: Follows WS-I Basic Profile 1.1 guidelines

4 Copyright © 2004-2008 Active Endpoints, Inc. 

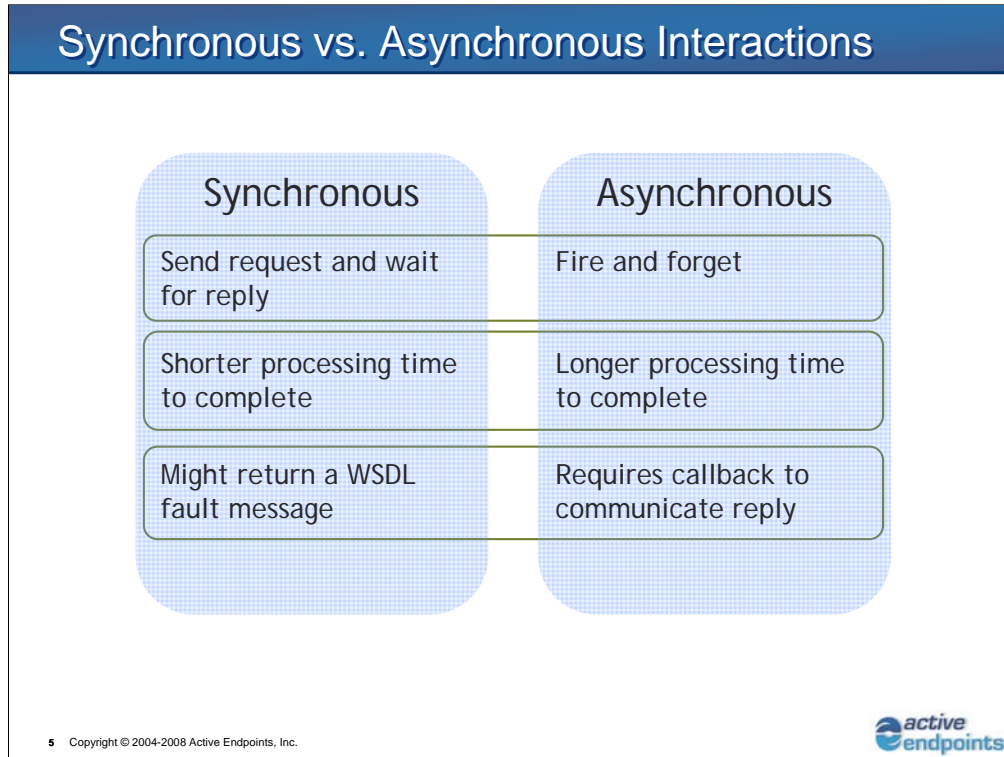
BPEL supports these two WSDL message exchange patterns. The process can be on either end of these patterns, it doesn't matter which.

The portType defines which of the two it is:

- Does the PortType have input and output? Then it is a request-response pattern
- Does the PortType have just input? Then it is a one-way pattern

Note that there are four types of message exchange patterns, and these are two of them. We don't need the other two because BPEL does not support them.

- 1 – Request/Response
- 2 – One-Way
- 3 – Notification // rarely used, not supported
- 4 – Solicit/Response // rarely used, not supported



Let's quickly review our terms, as they relate to interaction activities.

Synchronous: We send message to a partner, and then we wait for the partner to reply. This usually takes place over a short window because we are waiting for the response, and it might return a fault message as the reply.

Asynchronous: We send a message to a partner, and then we forget about it. There is no reply expected. This interaction is not usually over a short window because we are not waiting for our partner's reply. Note also that there is no mechanism to allow a reply, including one that is in the form of a fault. A reply will require a callback by the partner.

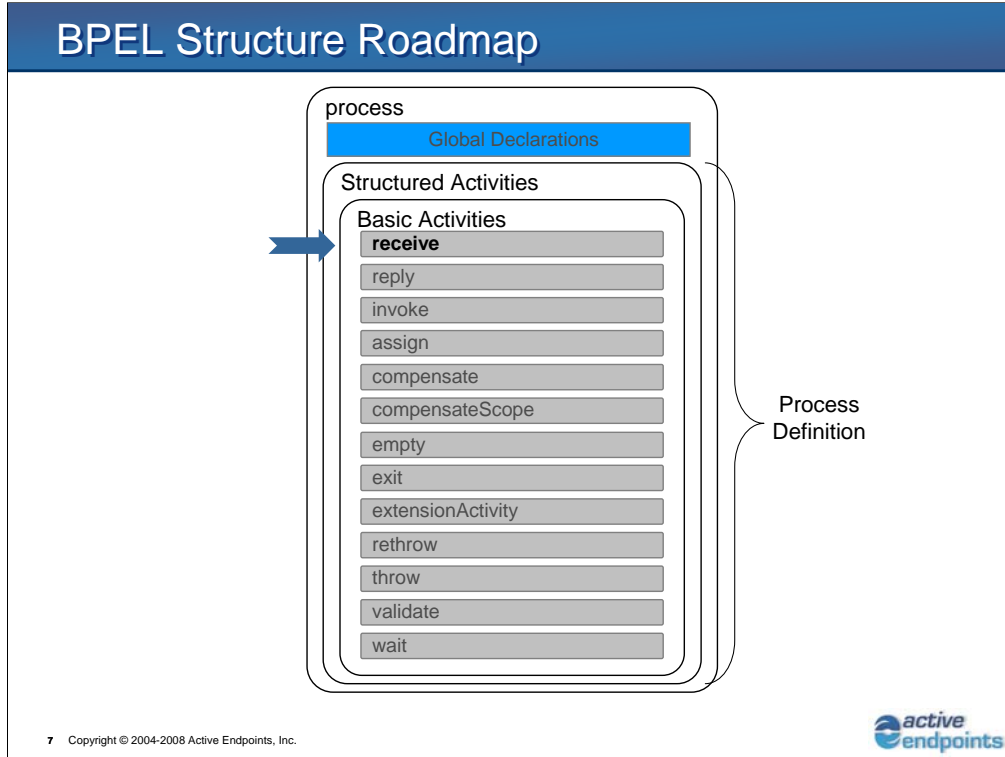
Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Overview of interaction activities
 - receive activity
 - reply activity
 - Creating interaction activities using the Operation wizard
 - invoke activity
 - Creating interaction activities using the Interfaces view and the Activity Palette

6 Copyright © 2004-2008 Active Endpoints, Inc.



Now that we have an idea how these interaction activities work, let's look at each of them in turn.



First, the Receive activity, which is a Basic BPEL activity in our Process Roadmap.

receive Activity Overview

- Used to accept requests from partners
 - Waits for an incoming message to arrive
 - Initial message to begin a new process
 - Subsequent callback messages from partners
 - Specifies a WSDL `operation` the partner is expected to invoke, and optional `portType`
 - via the named `partnerLink`
 - The message being received is optionally stored in an appropriately defined variable
- Plays a role in the business process lifecycle
 - If `createInstance="yes"` then it is considered an initial receive
 - Instructs the BPEL engine to create a new process instance

8 Copyright © 2004-2008 Active Endpoints, Inc.



The job of the Receive activity is to wait for an incoming message from a partner. The Receive can be in the form of a brand new communication or it can be a callback from a partner. Once it arrives, it kicks off some specified Operation(s), as described by the portType, and that portType is referenced by the partnerLinkType that defines the relationship. If the inbound message is to be stored in a variable, this is done through a Copy operation in an Assign activity. If the Receive is an “initial receive” then when the inbound message comes into the Receive activity, the server will attempt to match it to an existing process. If it cannot match the message to any one of the currently running processes, it starts a new process instance. This is the purpose of the Receive activity's “createInstance=yes” setting.

receive Activity Syntax

```

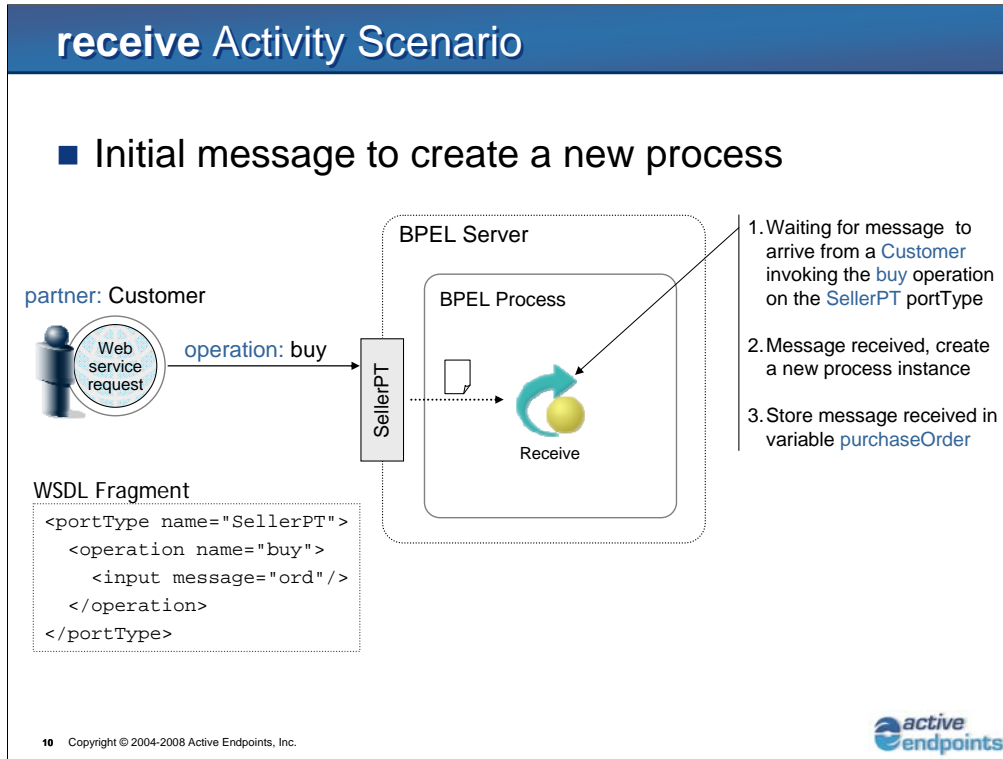
<receive partnerLink="NCName"
  portType="QName"?
  operation="NCName"
  variable="BPELVariableName"?
  createInstance="yes|no"?
  messageExchange="NCName"?
  standard-attributes>
  standard-elements
  <correlations>?
    <correlation set="NCName" initiate="yes|join|no"? />+
  </correlations>
  <fromParts>?
    <fromPart part="NCName" toVariable="BPELVariableName" />+
  </fromParts>
</receive>

```

© Copyright © 2004-2008 Active Endpoints, Inc.



Here we see the syntax for a Receive activity. For a complete declaration the partnerLink is required, the PortType is optional (because the PartnerLinkType implies it), the Operation is required, and the Variable is optional. The CreateInstance value is set to “yes” if this is an initial activity and “no” if it is not. The messageExchange is an optional attribute that disambiguates between messages and replies. (Message Exchanges are an advanced topic, and are not covered in this course.) And, of course, our activity has all of the standard attributes and elements, and optional correlations and FromParts.



Let's take a look at how a Receive creates a new process instance. On the left is our partner, on the right is our process. The Server, waiting for a message from the Customer, gets a message that invokes the process' "buy" operation, part of the SellerPT portType. The waiting Receive activity processes the inbound message and tries to match it – that is, *correlate it* - with an existing process instance. If it can't it creates a new instance of the process. The Server stores the inbound message in the variable "purchaseOrder" (shown on next slide) and from there the Process proceeds as defined.

receive Activity Example

```
<receive
  partnerLink="Customer "
  portType="SellerPT"
  operation="buy"
  variable="purchaseOrder"
  createInstance="yes" />
```

11 Copyright © 2004-2008 Active Endpoints, Inc.



Here we see the syntax for the previous example. The partnerLink – which is an instance of the partnerLinkType – is called “Customer”, the portType is called “SellerPT” and the operation defined for the portType is “buy.” The variable for the receive is “purchaseOrder” and, finally, we have the createInstance attribute, which tells the server whether to create an instance upon receipt of this message or not. Here, the createInstance attribute is set to “yes.”

receive Activity Semantics

- **receive** is a blocking activity
 - It does not complete until a matching message is received
 - Must have at least one `receive` activity
- Messages are routed to **receive** activities with matching **partnerLink** / **portType** / **operation**
 - Optionally matching correlation data
- New process instance is created if the **receive**'s **createInstance** attribute is set to “yes”
 - No other basic activity can come before an initial `receive` activity

12 Copyright © 2004-2008 Active Endpoints, Inc.



All processes must have at least one Receive activity, and the Receive is a blocking activity, meaning that nothing else happens until it gets the message it is waiting for. The partnerLink-operation definition is used to route the messages, optionally using a Correlation set to match the inbound message with a specific process instance. Since the Receive creates the process instance, obviously nothing can be done before this initial receive activity, because no process exists yet.

conflictingReceive Fault

- This fault is thrown when
 - More than one `receive` activity is simultaneously enabled for the same partner link, port type, operation and correlation sets

13 Copyright © 2004-2008 Active Endpoints, Inc.



Earlier we mentioned Standard BPEL Faults, and this is an example of a situation where one would be generated. If more than one Receive activity is enabled for the same partnerLink, operation and correlation set, then the system doesn't know what to do with an inbound message because there is no way for it to disambiguate the Receives and determine who is supposed to get the specific incoming message. Note that this situation can not occur in standard "in-line" execution of a BPEL process due to the fact that Receives are blockers. This situation can only arise when a Process has Receives that are active in parallel. (These kinds of activities will be covered in later units.)

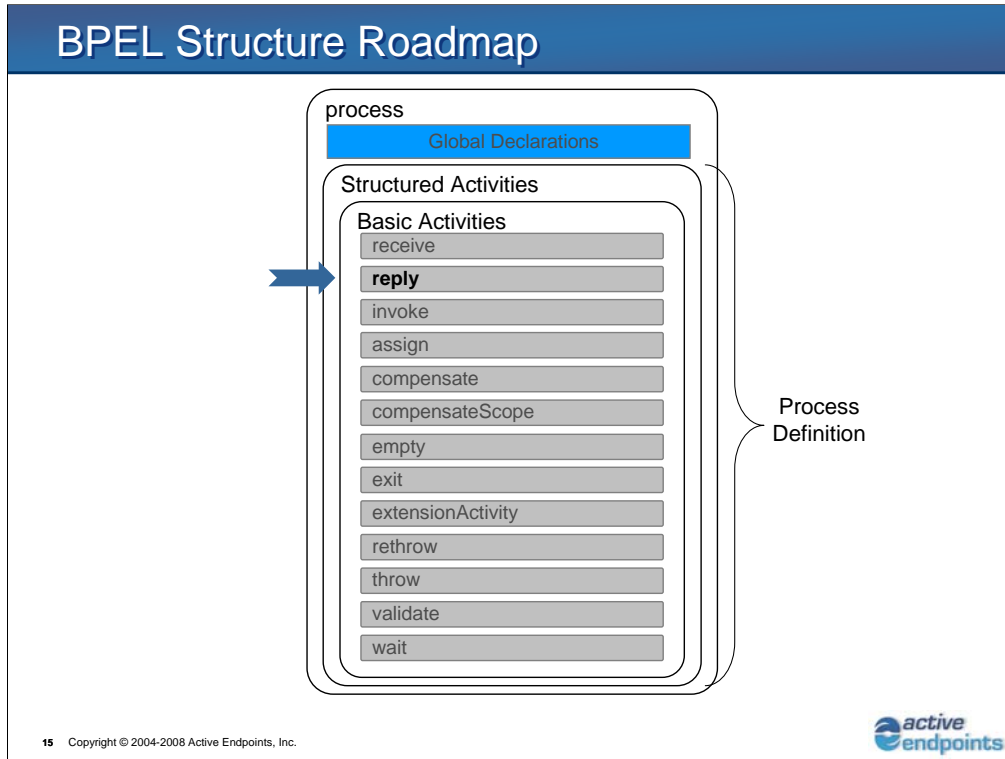
Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Overview of interaction activities
 - ✓ receive activity
 - reply activity
 - Creating interaction activities using the Operation wizard
 - invoke activity
 - Creating interaction activities using the Interfaces view and the Activity Palette

14 Copyright © 2004-2008 Active Endpoints, Inc.



We've seen the Receive activity, so now we'll look at the Reply activity.



According to our BPEL Roadmap, the Reply is a basic activity that is part of the process definition.

reply Activity Overview and Syntax

- Used to send a response to a request previously accepted through a **receive** activity
 - Combination of a *receive* and a *reply* activity forms a request-response message exchange

```

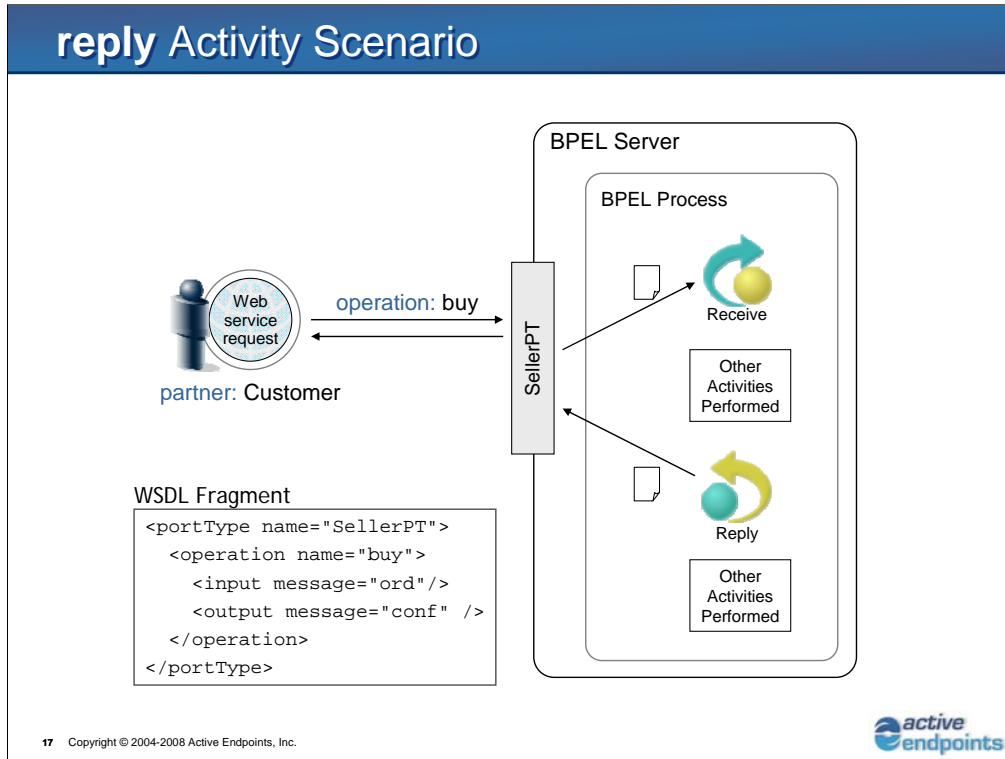
<reply partnerLink="NCName"
  portType="QName"?
  operation="NCName"
  variable="BPELVariableName"?
  faultName="QName"?
  messageExchange="NCName"?
  standard-attributes>
  standard-elements
  <correlations>?
    <correlation set="NCName" initiate="yes|join|no"? />+
  </correlations>
  <toParts>?
    <toPart part="NCName" fromVariable="BPELVariableName" />+
  </toParts>
</reply>

```

16 Copyright © 2004-2008 Active Endpoints, Inc.



Reply is only used to send a response to a message that was previously accepted through a Receive activity. The syntax for the Reply activity is almost exactly the same as it is for the Receive activity, except there is no option for “createInstance” because it can only be sent by an existing instance.



Here we have a diagram of a typical Request-Response conversation. Our "Customer" partner invokes our web service with the "buy" operation that is exposed by the SellerPT portType. The Seller portType receives the input message defined as "ord" and then it replies with an output message "conf." Other activities can be performed *between* the Receive and its Reply, as needed, and once the Reply is sent other activities can be performed.

reply Activity Example

```
<reply  
  partnerLink="Customer "  
  portType="SellerPT"  
  operation="buy"  
  variable="orderConfirmation" />
```

18 Copyright © 2004-2008 Active Endpoints, Inc.



Here is the Reply syntax for the previous example. As we saw, the partnerLink is “Customer”, the portType is “SellerPT” and the operation is “buy.” The variable "orderConfirmation" is used by the Reply.

Replying With A Fault

- The response message can indicate a fault rather than a normal response
 - `faultName` attribute indicates which fault to return
 - `variable` attribute to indicate a `messageType` variable for the corresponding fault data

WSDL Fragment For The Process

```
<wsdl:portType name="SellerPT">
  <wsdl:operation name="buy">
    <wsdl:input message="tns:buyInput" />
    <wsdl:output message="tns:buyOutput" />
    <wsdl:fault name="BadPOFault" message="tns:BadPO" />
  </wsdl:operation>
</wsdl:portType>
```

19 Copyright © 2004-2008 Active Endpoints, Inc.



We can reply with the Operation's defined output message or with a user-defined fault, as appropriate. In this example, we can respond with either the output message "tns:buyOutput" or with the fault message called "BadPOFault", which uses the message defined as "tns:BadPO." Note that this is a user-defined fault, not a system fault.

Replying With A Fault Example

```
<reply
  partnerLink="Customer "
  portType="SellerPT"
  operation="buy"
  faultName="ord:BadPOFault "
  variable="poError" />
```

20 Copyright © 2004-2008 Active Endpoints, Inc.



Here we see the Reply syntax for responding with a fault message. If the process calls for it, then we will respond with a fault named "ord:BadPOFault." The Variable has been called "poError," which probably means that it is going to be used in some kind of error response or handling.

reply Activity Semantics

- Meaningful only for synchronous process interactions
- A **reply** MUST always be preceded by a **receive** activity for the same **partnerLink** and **operation**

21 Copyright © 2004-2008 Active Endpoints, Inc.



Of course, a Reply activity only makes sense if it is preceded by a Receive activity. The Reply must match up to a previous Receive with the same partnerLink-operation pair. It is the combination of the two that defines the process's synchronous interactions.

Standard BPEL Faults Which May Occur

- Replying to a matching **receive** more than once
 - Results in a BPEL `invalidReply` fault
- Not replying to a matching **receive**
 - Results in a `missingReply` fault
- When there are multiple simultaneous open inbound synchronous requests
 - Results in a `conflictingRequest`

22 Copyright © 2004-2008 Active Endpoints, Inc.



Here are some of the standard BPEL errors which may be generated when dealing with Receives and Replies. Too many Replies to a single Receive results in an `invalidReply` fault. This is obvious, when you think about. If you get three replies, which one is the correct one to respond to? No reply at all results in a `missingReply` fault. This is also obvious. If you never get an answer something's wrong. Unable to reply due to multiple inbound synchronous requests results in a "conflictingRequest" fault. (This fault was discussed earlier in this unit.) This fault occurs when you receive duplicate (or more) inbound Request messages before you've replied to the first one (too many Requests.) These requests must have the same `partnerLink`, operation and `Correlation Set` to produce this fault.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Overview of interaction activities
 - ✓ `receive` activity
 - ✓ `reply` activity
 - Creating interaction activities using the Interface View
 - `invoke` activity
 - Creating interaction activities using Interfaces View and the Activity Palette

23 Copyright © 2004-2008 Active Endpoints, Inc.



Now that we've looked at the Receive and Reply activities, let's look at how we can create activities using ActiveVOS Designer's Interface View.

Creating Interaction Activities

- ActiveVOS Designer provides three ways to create interaction activities

- 1. Drag & Drop from the Interfaces View
- 2. Drag & Drop from the Process Editor Palette
- 3. Using the New Interface Wizard

24 Copyright © 2004-2008 Active Endpoints, Inc.



There are three options available to you when you need to create Interaction Activities:

- 1.) Using Drag & Drop from Interfaces view, which uses the available artifact definitions to give you a fully or partially completed activity.
- 2.) Using Drag & Drop from the Process Editor's Palette – which requires that your new activity be defined completely from scratch.
- 3.) The New Interface Wizard, which guides you through the creation of a new Operation from the ground up.

Creating Interaction Activities – Interface View

- Assists in creating an interaction activity for an existing WSDL defined operation
 - Interfaces View allows you to drag and drop an operation onto the canvas
- Interface View will either:
 - Create from a defined PartnerLinkType
 - Uses the defined portType, operation, messages and partnerlinkType necessary to satisfy the activity
 - Create from a defined PortType
 - Uses the defined portType, operation, messages and assists you in creating the necessary PartnerLinkType to satisfy the activity

25 Copyright © 2004-2008 Active Endpoints, Inc.

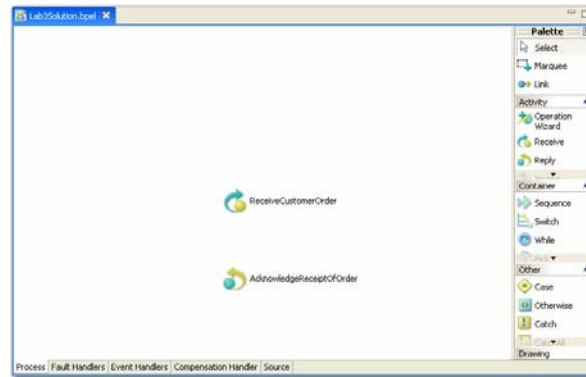


If we have an existing operation (i.e., our Interfaces view has an existing definition for this operation) with a PartnerLinkType, the Interfaces view will allow you to drag and drop an Operation onto the Process Editor's canvas using the full definition. Or, if you have a defined PortType, but need to define the PartnerLinkType and PartnerLink, it will guide you through the process of adding what it needs to complete the definition.

Lab 3 – Create a New Process

■ Overview of Lab Exercises

- Create new BPEL Process
- Use the Interface View to create the initial `receive / reply` pair



26 Copyright © 2004-2008 Active Endpoints, Inc.



This is the third lab in the BPEL Fundamentals course. Its purpose is to continue with the development of the project we created during Labs #1 & 2 by creating the process that will be used by the labs that follow. During the course of this lab you will create a new BPEL Process file (OrderProcess.bpel) and then use the Interface view to create the process's initial Receive/Reply pair.

Lab #3 has four tasks:

- 1.) Create a new BPEL Process
- 2.) Add a Receive/Reply pair of interaction activities
- 3.) Rename the Activities
- 4.) Update the WSDL Namespace prefixes in the workspace Preferences

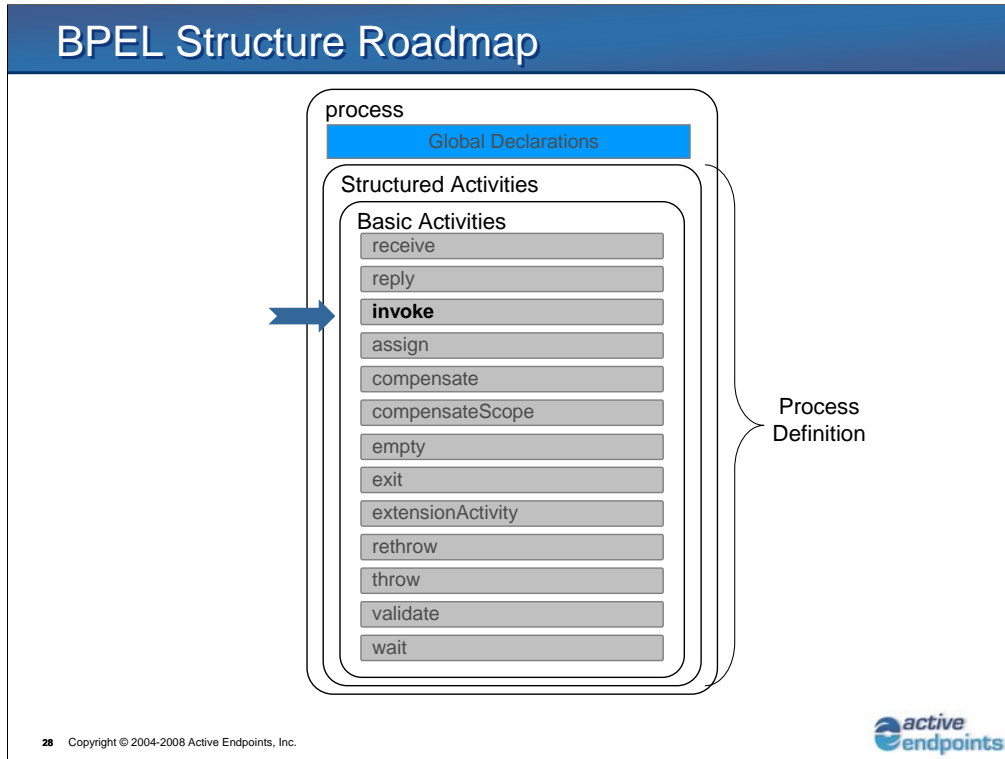
Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Overview of interaction activities
 - ✓ receive activity
 - ✓ reply activity
 - ✓ Creating interaction activities using the New Interface wizard
 - invoke activity
 - Creating interaction activities using the Activity Palette

27 Copyright © 2004-2008 Active Endpoints, Inc.



Now that we are familiar with the Receive and a Reply activities, we'll continue our studies with the Invoke activity.



As we can see on the BPEL Structure Roadmap, the Invoke is a basic BPEL activity.

invoke Activity Overview and Syntax

- Allows a business process to invoke a **portType operation** offered by a partner-provided service
 - via the named `partnerLink`
 - Can be a one-way or request-response message exchange

Minimal Syntax

```
<invoke partnerLink="NCName"
        portType="QName"?
        operation="NCName"
        inputVariable="BPELVariableName"?
        outputVariable="BPELVariableName"?
        standard-attributes>
  standard-elements
</invoke>
```

29 Copyright © 2004-2008 Active Endpoints, Inc.



Here we see the minimal syntax needed for our Process to Invoke a partner's web service. An Invoke can be either one-way or a request-response exchange. Here, we know that is a Request-Response exchange because we have both input and output messages. When declaring the Invoke, the `PartnerLink` is required, as is the `Operation`, but the `PortType` is optional because the `PartnerLink` implicitly specifies the `portType`. We can also specify the activity's input and output message variables, which are both optional... because we can use a `FromPart` or a `ToPart`. And, of course, the Invoke can also have all the standard attributes and elements.

Compared to the `Receive` and `Reply` activities, `Invoke` has by far the most complex semantics:

A `<toPart>` and an `inputVariable` cannot be used at the same time.

A `<fromPart>` and an `outputVariable` cannot be used at same time.

Use of `<toPart>` and `<fromPart>` constitute a virtual assign, meaning that the value is copied into the target w/o an actual `Assign` activity.

invoke Activity Full Syntax

```

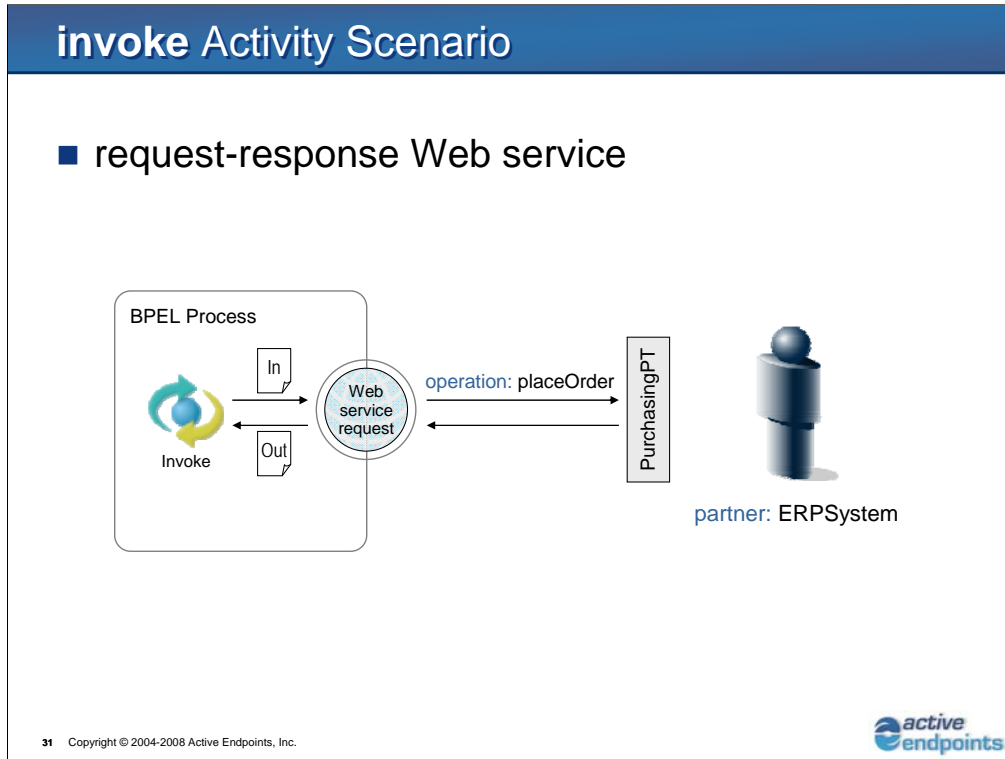
<invoke partnerLink="NCName" portType="QName"? operation="NCName"
  inputVariable="BPELVariableName"? outputVariable="BPELVariableName"?
  standard-attributes>
  standard-elements
  <correlations>?
    <correlation set="NCName" initiate="yes|join|no"?
      pattern="request|response|request-response"? />+
  </correlations>
  <catch faultName="QName"? faultVariable="BPELVariableName"?
    faultMessageType="QName"? faultElement="QName"?>*
    activity
  </catch>
  <catchAll>?
    activity
  </catchAll>
  <compensationHandler>?
    activity
  </compensationHandler>
  <toParts>?
    <toPart part="NCName" fromVariable="BPELVariableName" />+
  </toParts>
  <fromParts>?
    <fromPart part="NCName" toVariable="BPELVariableName" />+
  </fromParts>
</invoke>

```

30 Copyright © 2004-2008 Active Endpoints, Inc.



Here we see the full syntax, where the white area is what we just looked at and the pale yellow area defines the optional sections of the activity definition. The Optional definitions for the Invoke activity include Correlations, one or more Catches, a catchAll and any Compensation handlers, all of which are topics we'll discuss later on in the course. The "From" and "To" parts are used to populate inbound and outbound messages, respectively, as an alternate way to communicate with a web service, beyond the actual Input and Output variables declared for the activity. Note also that multiple parts can be declared in the From and To sections.



Here our BPEL process – which is on the left – Invokes a web service, which is on the right. The Web Service being Invoked is part of an ERP System (Enterprise Resource Planning) that uses the “PurchasingPT” portType, and it is from that portType that we call one of its defined operations, in this case the operation “placeOrder.” Because this is a Request-Response service, we use both Input and Output variables (however they are defined) to send and receive communication to and from the service.

invoke Activity Example

- request-response Web service

```
<invoke partnerLink="ERPSystem"  
        portType="PurchasingPT"  
        operation="placeOrder"  
        inputVariable="purchaseOrder"  
        outputVariable="poResult" />
```

32 Copyright © 2004-2008 Active Endpoints, Inc.



Here is the syntax for the previous scenario. We have the PartnerLink called “ERPSystem” and a portType associated with it, called “PurchasingPT”, and that portType has an operation called “placeOrder.” Then we declare the inputVariable “purchaseOrder” and the outputVariable “poResult.” These are messageType variables and they must match the schema that is defined in the WSDL for that particular Operation. Note that an Invoke with a Request-Response has two-way communication, and therefore requires either an outputVariable or a fromPart and an inputVariable or a toPart. A one-way exchange only needs one variable, either the inputVariable or a ToPart.

invoke Activity Semantics

- Request-response message exchange requires both an input variable and an output variable
 - Might return a WSDL fault message
- One-way message exchange requires only the input variable of the operation
 - As it does not expect a response as part of the operation

33 Copyright © 2004-2008 Active Endpoints, Inc.



A Request-Response type of Invoke must have both input and output messages (or their to-from substitutions), and can optionally have one or more fault messages. Those fault messages are defined by the User and are over and above the standard faults generated by BPEL itself. A one-way Invoke activity has only an input message and cannot have fault messages, as there is no response expected, and therefore no means to communicate the fault back to the process.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Overview of interaction activities
 - ✓ receive activity
 - ✓ reply activity
 - ✓ Creating interaction activities using the New Interface Wizard
 - ✓ invoke activity
 - Creating interaction activities using the Activity Palette

34 Copyright © 2004-2008 Active Endpoints, Inc.



Now that we know how it works, how do we create an Invoke activity? As we discussed previously, we can use the Interface view to create new interaction activities, as was done in the previous lab. Now we will revisit the Interfaces view and then see how to create an Invoke from scratch using the Activity Palette.

Creating Interaction Activities

- ActiveVOS Designer provides three ways to create interaction activities
 1. Using the New Interfaces Wizard
 2. Drag & Drop from the Interfaces View
 - ➔ 3. Drag & Drop from the Process Editor Palette

35 Copyright © 2004-2008 Active Endpoints, Inc.

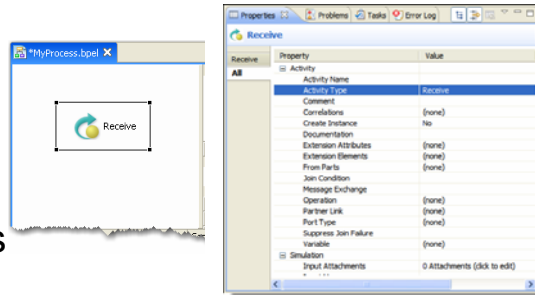


Dragging & Dropping from the Process Editor's Palette creates the Operation from scratch, so you have to fill in all of the required definition information.

Creating Interaction Activities – Palette

- Use the palette to add Receive, Reply, or Invoke activities
- Will need to manually specify the properties for the interaction activity
 - Including partner link, port type, operation, and messages

Note: Also need to manually import WSDL, create the partner links, and all required variables



36 Copyright © 2004-2008 Active Endpoints, Inc.



When we use the palette, we get an empty shell for the activity. We have no defined Operation, portType or partnerLinkType to work with. We need to complete all of the data – in this order – from the Properties View:

- First, we select the partnerLink. This is an instance of a PartnerLinkType, which defines a relationship with a partner.
- Second, we choose from among the Operations defined for the PortType associated with the PartnerLink.

Note that we also have to import any WSDLs that are needed (i.e., the ones that hold the Operation, portType or partnerLinkType definitions chosen previously) if they are not already part of the project.

Unit Summary

- Now you are familiar with:
 - Overview of interaction activities
 - `receive` activity
 - `reply` activity
 - Creating interaction activities using the New Interfaces wizard
 - `invoke` activity
 - Creating interaction activities using the Activity Palette

37 Copyright © 2004-2008 Active Endpoints, Inc.



We've talked about the Receive, Reply and Invoke activities and we've gone over four ways to create these activities:

- from the Interfaces View (dragging from the portType)
- from the Interfaces View (dragging from the partnerLinkType)
- from the Process Editor's Palette
- Using the New Interfaces Wizard