

This is Unit #9 of the BPEL Fundamentals course. In past Units we've looked at ActiveVOS Designer, Workspaces and Projects and then we created the Process itself. Next, we looked at global declarations and interaction activities. Then, we looked at our first container activity, the Sequence and at Assignments and their Copy Operations. In this Unit we will take a look at a new subject, Correlation.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - What is Correlation
 - Message Properties
 - Correlation Sets
 - Working with Correlation Sets
 - Using Correlations Sets

2 Copyright © 2004-2008 Active Endpoints, Inc.



So, what is Correlation in a BPEL process?

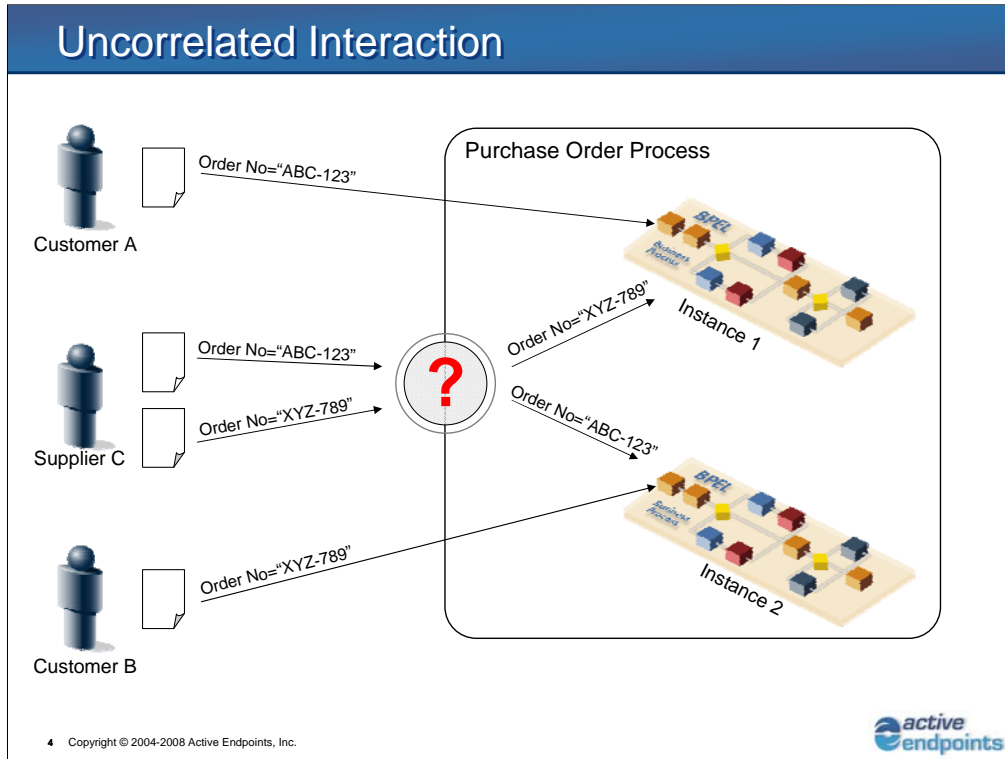
Correlation Overview

- Many if not all business processes require the ability to have multiple instances executing simultaneously
 - Therefore messages sent to the business process need to be delivered to the right instance
- Messages being exchanged typically contain data that can be used to uniquely identify which instance to deliver the message to
 - The exact name or location of this data often varies from message to message
- BPEL provides a way to use this business data to maintain a conversational reference to a specific process instance

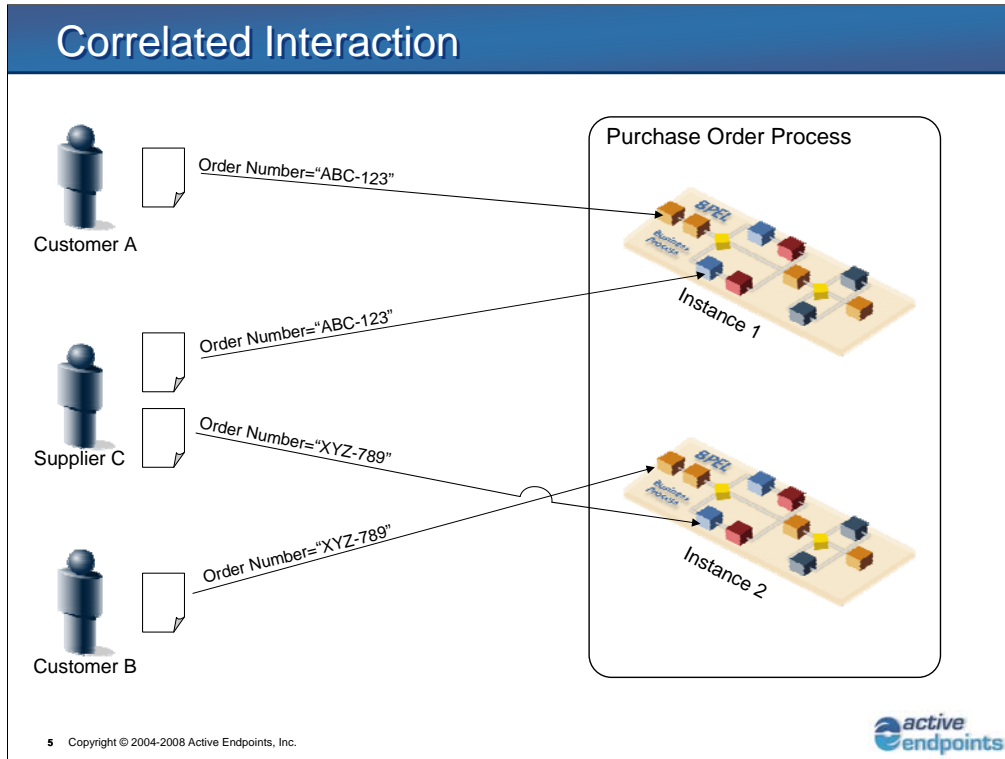
3 Copyright © 2004-2008 Active Endpoints, Inc.



Correlation makes it practical to have more than one process instance active at a time. Why is this so? Let's say you have three process instances executing simultaneously, and all might be doing exactly the same thing at exactly the same time. In this situation you will have three inbound messages, so how can you tell which message matches up to which process instance? The answer is that we use the data contained in the messages themselves to perform the Correlation, data such as a Purchase Order Number and/or a Customer Number. The correlated data must, as a whole, create a unique data set. As we'll see later, it is possible that this unique data can be found in different locations for different message definitions. So, using business data to correlate the messages is what we want to accomplish, in order to maintain *conversational integrity*.



Here we have a purchase order process with two Customers “A” and “B”. Each Customer has their own Order Number, of course, and there are two Process instances, each associated with one of the customers. We can have others involved, too, such as supplier “C” who is filling the two Customer’s orders. If these customers send order data - such as pricing or confirmation numbers - to the instance... on the two orders... how do we tell... which message goes to which instance? We can see in the slide that unless we correlate the messages correctly, it is possible that the an incoming message could be matched to the wrong instance.



To solve this problem we use the data in the messages themselves to direct them to the correct process instance. In this example we are using the order number as the correlation key. Note that we are not using any kind of extra “tag” or “label” for this purpose. That would require that all the services we invoke comply with our tagging requirements, which is a deal-killer.

Steps for Adding Correlation to a Process

1. Create Message Properties and Property Aliases
2. Create a Correlation Set
3. Initiate Correlation in an Activity
4. Correlate Messages for a Process Instance

6 Copyright © 2004-2008 Active Endpoints, Inc.



Here are the steps to add correlation functionality to a process.

- 1.) First, messages used in the conversation need a defined *Property* and *Property Alias* for each correlation element.
- 2.) Next, we use these to create a Correlation Set.
- 3.) Then, we initiate correlation for an activity.
- 4.) Finally, we correlate the message traffic using the internal message data for each process instance.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ What is Correlation
 - Message Properties
 - Correlation Sets
 - Working with Correlation Sets
 - Using Correlations Sets

Message Properties Overview

- Used to name and represent distinguished data elements within a message
 - Properties
 - Defines a globally unique name that is associated with an XML Schema simple type
 - Property Aliases
 - maps a named property to a field in a message part
 - Identified by a query whose type must evaluate to the same type as defined by the property
- BPEL defines the **property** and **propertyAlias** constructs within WSDL
 - Extensibility mechanism of WSDL 1.1 is used to define these new definition types
 - like `partnerLinkTypes`

8 Copyright © 2004-2008 Active Endpoints, Inc.



All variables in WS-BPEL can have Properties defined for them. A *Variable Property* defines a globally unique name for a data item that is associated with an XML schema type or element. A Variable Property is a direct reflection of the underlying variable and its value is therefore always the same. *Message Properties* are a specific type of the more generic “<variable> properties” mechanism, applying it to variables of Message Type. A *Property Alias* maps a property to a specific field in a message Part. Each Property Alias uses a query that tells it how to find the data under scrutiny in a particular type of message. It logically follows that the query’s return type must be the same as the Property Alias’ type. A Property Alias, therefore, says “Here’s how to find *this* property in *this* type of message...” Both Properties and Property Aliases are extensions of the WSDL specification, like `partnerLinkTypes`.

Note: (from the BPEL 2.0 specification) "Properties are useful on non-message variables as a way to isolate the WS-BPEL process’s logic from the details of a particular variable’s definition. Using properties, a WS-BPEL process can isolate its variable initialization logic in one place and then set and get properties on that <variable> in order to manipulate it. If the <variable>’s definition is later changed the rest of the WS-BPEL process definition that manipulates that variable can remain unchanged."

Message Properties Syntax

```

<wsdl:definitions
  xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop">
  ...
  <vprop:property name="ncname" type="qname"/>

  <vprop:propertyAlias propertyName="qname" messageType="qname"
    part="ncname">
    <vprop:query>query</vprop:query>
  </vprop:propertyAlias>
  ...
</wsdl:definitions>

```

© Copyright © 2004-2008 Active Endpoints, Inc.



So, just as a WSDL document can be extended with partnerLinkType definitions, our WSDL documents can also be extended to include Property and Property Alias definitions. Here we have a fragment from a WSDL document's definitions section, starting with the namespace that sets the prefix using the convention "vprop."

```
<vprop:property name="NCName" type="QName" element="QName" />
```

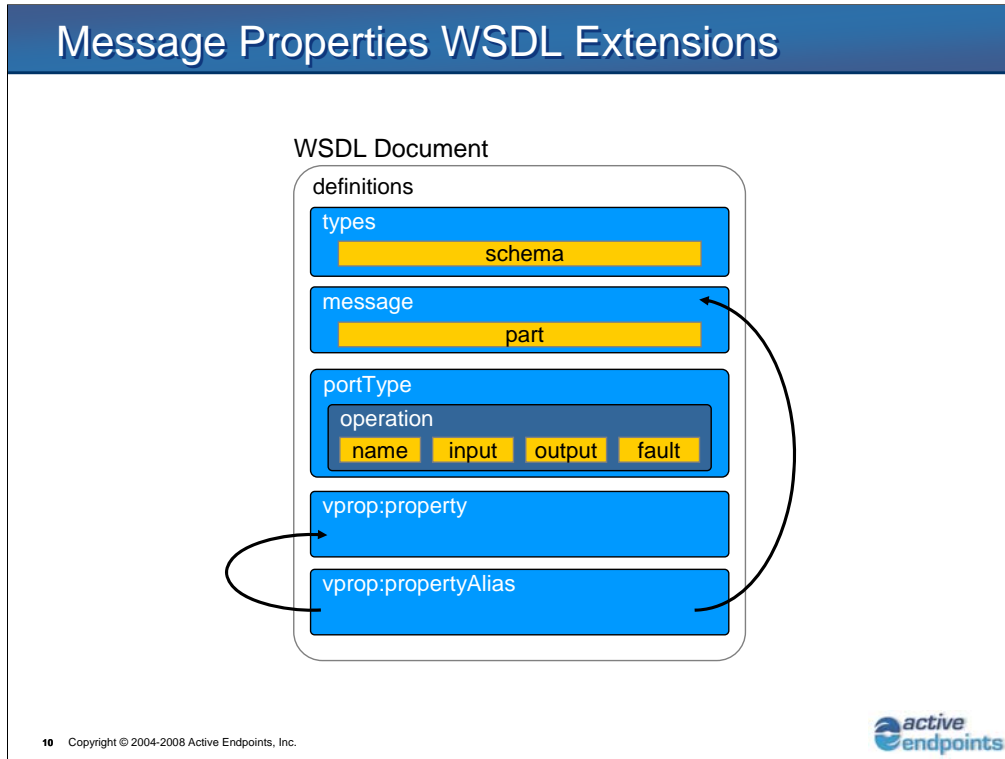
"vprop" is a convention for the prefix, and the property must have a name and it must have *either* an element or a type attribute.

```
<vprop:propertyAlias propertyName="QName" messageType="QName"
part="NCName">
```

The Property Alias is defined in the next line, with the name of the property it is an alias for, the MessageType involved and the Message part that has the data.

```
<vprop:query>query</vprop:query>
```

It uses a query of some kind that tells the Process how to get the variable data it needs out of this particular MessageType.



Therefore, a Property Alias has relationships with both the Property itself and with the Message part it refers to. Note that there can be multiple Properties defined for a single message, and multiple Aliases defined for each Property.

Message Properties Example

WSDL Fragment

```

<wsdl:definitions
  ...
  <message name="POInput">
    <part element="ord:purchaseOrder" name="Input" />
  </message>

  <vprop:property name="PONumber" type="xs:string" />
  <vprop:propertyAlias propertyName="tns:PONumber"
    messageType="ord:POInput" part="Input">
    <vprop:query>/po/orderNo</vprop:query>
  </vprop:propertyAlias>
  ...
</wsdl:definitions>

```

11 Copyright © 2004-2008 Active Endpoints, Inc.



In this example we have a message called “POInput” that has a part that is an Element. That element is of the “ord:purchaseOrder” type, and is named “Input.”

And below that we have a *property* defined this way:

vprop:property the name = “PONumber” the type = “xs:string”

And we have a *property alias* for that property, defined this way:

vprop:propertyalias propertyName=“tns:PONumber” which is what ties this alias to the “PONumber” property.

for messageType: “ord:POInput” which has a message part called “Input”...

We use this query: “/po/orderNo” to extract the information.

getVariableProperty Function

- `getVariableProperty` is used to extract property values from variables
 - Increases flexibility in extracting data from a message
 - Do not have to specify the exact location of the data
 - One of the BPEL extensions to XPath functions

```
bpel:getVariableProperty ('variableName',  
                           'propertyName')
```

12 Copyright © 2004-2008 Active Endpoints, Inc.



We can use the *getVariableProperty* function to extract values from variables, even if you don't know exactly where the data is in the specific variable under scrutiny. Looking at the syntax, the first argument is the source variable for the data, and the second argument is the QName of the variable property to select. This argument must be a string literal and this is enforced by Static Analysis. The return value of this function is calculated by running the property alias query defined for the type of message involved. Both arguments must be enclosed in single quotes, also enforced by Static Analysis.

Message Properties Semantics

- Properties are typed using XML Schema simple types
 - Any one of the built-in primitive or derived data types
 - Any user derived data type whose base is one of the built-in primitive or derived data types
- Properties are bound ("aliased") via XPath expressions to locations in message parts

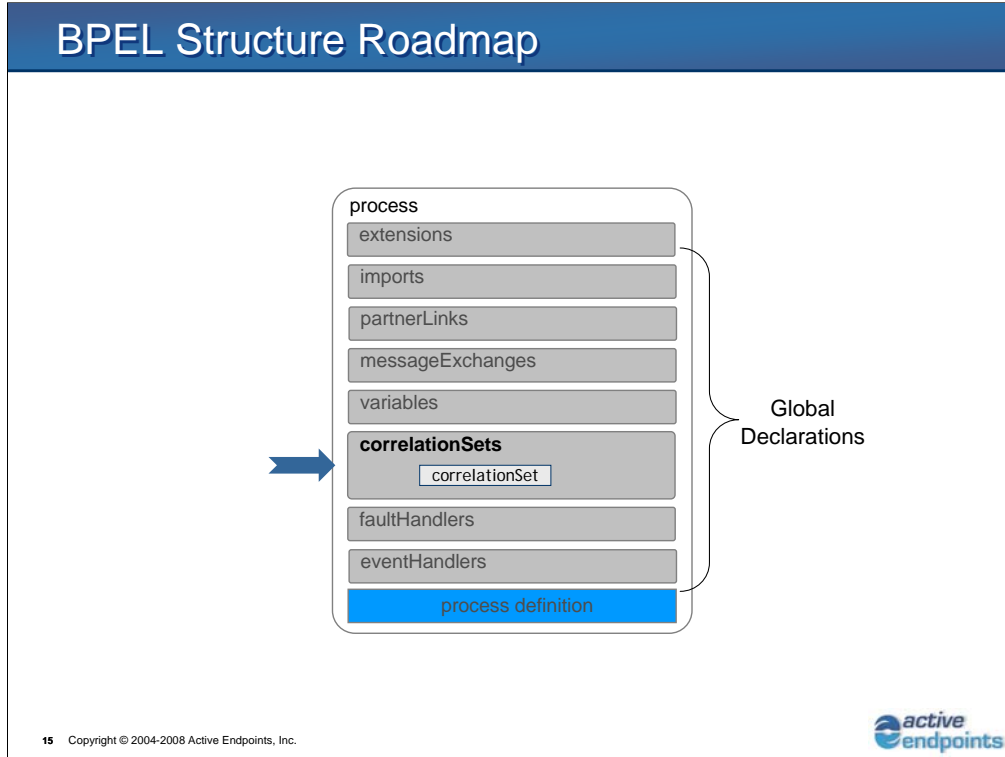
13 Copyright © 2004-2008 Active Endpoints, Inc.



Properties are typed using XML Schema simple types. They can be a built in simple type or a derived simple type. A built-in simple type could be a string, for example, whereas a derived type could be a string with a pattern restriction (called a "facet") such as the "sku" type we saw earlier. Properties are aliased to a specific location in each message type.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓What is Correlation
 - ✓Message Properties
 - Correlation Sets
 - Working with Correlation Sets
 - Using Correlations Sets



Looking at our BPEL Structure Roadmap, we see that the Correlation Set definitions are part of our Global Declarations.

Correlation Set Overview and Syntax

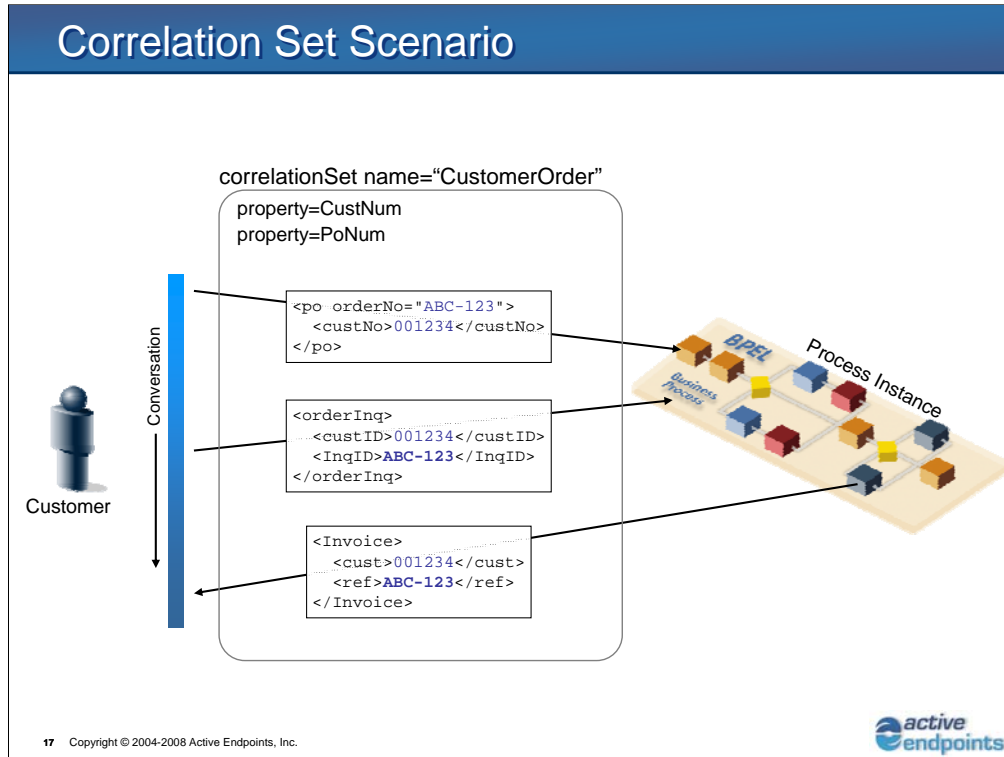
- Used to associate messages with process instances
 - Comprised of a set of properties shared by messages
 - Defines a way to identify an application-level conversation
 - Can be declared globally for the whole process or locally within scopes

```
<correlationSets>?  
  <correlationSet name="NCName"  
                  properties="QName-list" />+  
</correlationSets>
```

16 Copyright © 2004-2008 Active Endpoints, Inc.



As we discussed earlier, the purpose of a Correlation Set is to associate an inbound message with a specific process instance. They are needed to maintain conversational integrity for long running process instances, and they are defined at either the process or scope level.



Our Correlation Set name is CustomerOrder, and it contains two properties: CustNum and PoNum.

The CustNum Property has an alias that references the “custNo” element in any message of that type, which, in this case, has the value “001234.”

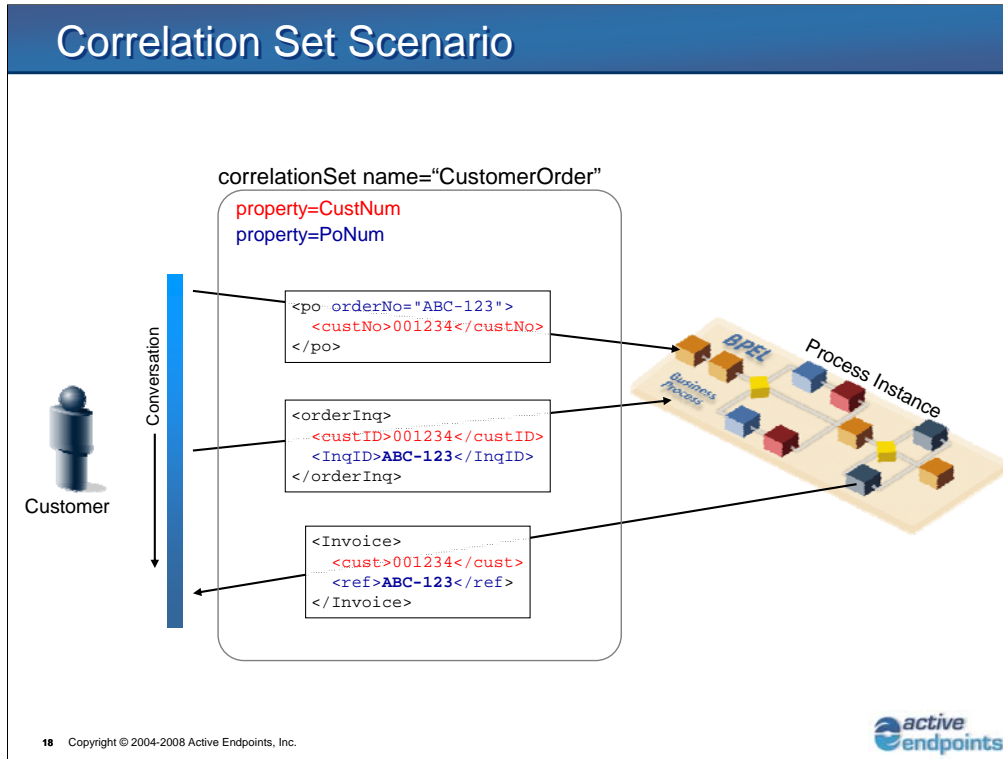
The PoNum property has an alias called “orderNo” which, in this case, has the value “ABC-123.”

As the conversation continues, the Correlation set is still using the properties CustNum and PoNum, but now we use additional *aliases* for these properties.

The CustNum property is using a second alias called “custID” which has the value “001234,” and the PoNum property uses the alias “InqID” which has the value “ABC-123.”

Finally, we have a later part of the conversation with another set of aliases for our properties, “cust” for the CustNum property and “ref” for the PoNum property.

Notice that we are using different property aliases at different stages of the conversation, but the *values* involved remain the same. It is this *consistent data set* that keeps things correlated.



Here we see the same slide we just looked at, but in this version we've highlighted in red the property "CustNum" and its three property aliases – "custNo", "custID" and "cust" - that are used to keep track of the customer number to correlate the conversation. The "PoNum" Property and its aliases – "orderNo", "InqID" and "ref" - are shown in blue. Each of the aliases finds the correlation data (whose values remain consistent) in a different type of message.

Correlation Set Example

```
<process>
  ...
  <correlationSets>
    <correlationSet name="CustomerOrder"
      properties="cor:CustNum cor:PoNum" />
  </correlationSets>
  ...
</process>
```

19 Copyright © 2004-2008 Active Endpoints, Inc.



Here we see the declaration for a Correlation Set at the Process level. In this case the name of the Correlation Set is “CustomerOrder”, and it has two properties – “CustNum” and “PoNum.”

Correlation Set Semantics

- A correlationSet specifies a named list of properties
 - Multiple properties can be combined into a composite correlation key
- BPEL correlates messages based on properties referenced in a correlationSet

20 Copyright © 2004-2008 Active Endpoints, Inc.



Recall that a Correlation Set uses a message's own data to correlate it. Correlation sets examine specific business data inside the message and then use that data to direct the message to the proper instance. Note that there are no "correlation tokens" being created. We are using pre-existing business process data to perform the correlation.

Because a single property's value might be duplicated in another instance's conversation, we can use *multiple* properties to define a single Correlation Set, which can help ensure the uniqueness of the Correlation Set. For example, a CustomerID might be used in more than one conversation, so we add the Purchase Order property. The combination of these two is likely to be unique, although each business situation will be different. Note also that *all* data in a Correlation Set must *always* match in order for the correlation of the conversation to succeed.

As we've said, we can combine multiple properties into a single Correlation Set to ensure data uniqueness. We can also have more than one Correlation Set with a single partner if the different interactions with that partner use completely different data sets (one for each of a portType's different operations, for example.)

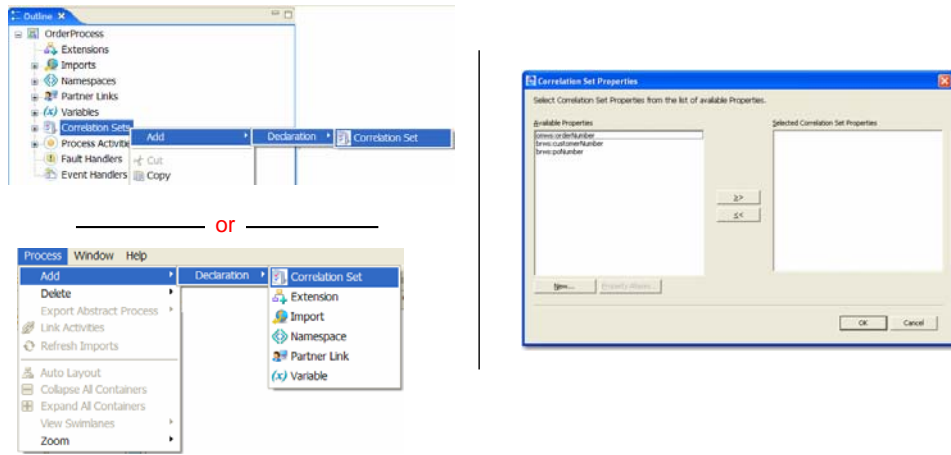
We can also have multiple Correlation Sets in order to maintain the integrity of conversations we are having with each of our different partners. So, for example, if we are talking to three different partners during the process, we will have three different Correlation Sets, one for each of these different conversations.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓What is Correlation
 - ✓Message Properties
 - ✓Correlation Sets
 - Working with Correlation Sets
 - Using Correlations Sets

Working with Correlation Sets - Adding

- Correlation Sets are added to a process definition via the Outline view or Process menu



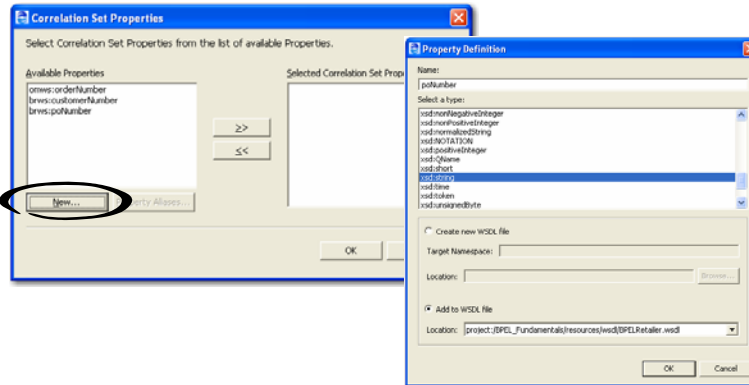
22 Copyright © 2004-2008 Active Endpoints, Inc.



Now, let's take a look at how to work with Correlation Sets in ActiveVOS Designer. Create a new Correlation Set using either the Outline View or the Process menu, and then use the Right Mouse menu to select Add->Declaration->Correlation Set to open up the Correlation Set properties dialog. All of the available Properties are displayed in the left pane, and any Properties that have been selected for the current Correlation Set are displayed in the right pane.

Working with Correlation Sets – Creating Properties

- You can create message Property definitions and add them to:
 - New WSDL file
 - Existing WSDL file



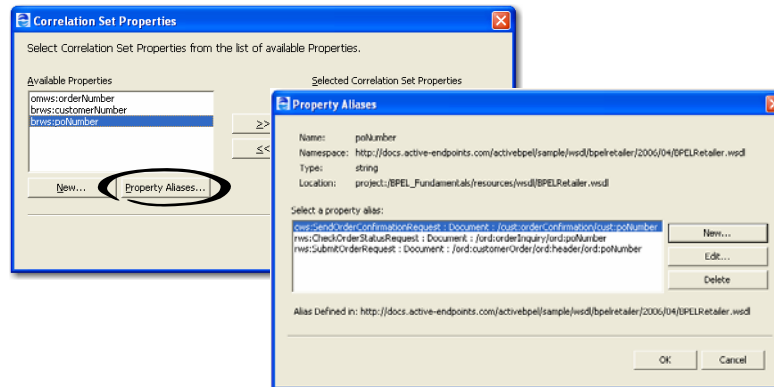
23 Copyright © 2004-2008 Active Endpoints, Inc.



Before you can associate an activity with a Correlation Set, you need to have the property's definition in a WSDL file. There are two necessary pieces of information: the property name and the schema type or element. In this example, the User selected New to create a new property definition. In the Property Definition dialog, the "Select A Type" field shows the various Schema available and the types they define. In this example, the schema type "xsd:string" is selected and the User adding a Property to an existing WSDL document. Once you have a Property defined you can add the Property Aliases as needed. As we've said before one property can have multiple aliases assigned to it. And, of course, you can add or delete correlation sets. We are only using data types defined in the XML schema (`xmlns:xs="http://www.w3.org/2001/XMLSchema"`), which is why they all carry "xsd:" prefixes. If we declared our own data types, when they appeared in this window they would have the different prefixes we defined, each pointing to a different namespace.

Working with Correlation Sets – Creating Property Aliases

- Select a message Property from the Available Properties list, then select Property Aliases...



24 Copyright © 2004-2008 Active Endpoints, Inc.

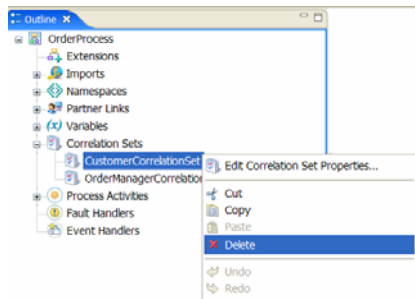


Once you have your properties defined in the WSDL file, you can create a new Alias for it. Here the User has selected the “brws:poNumber” property. You can either use existing aliases – here there are three - or create a new alias for it. The Alias definition will have the Property and the Query that you need to get that property’s value.

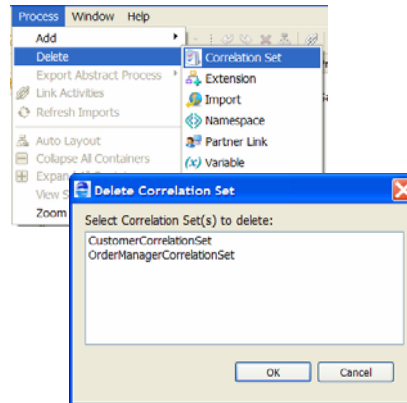
Working with Correlation Sets - Deleting

- Correlation Sets are deleted from a process definition via the Outline view or Process menu

- Outline view



- Process menu



25 Copyright © 2004-2008 Active Endpoints, Inc.



You can Delete a Correlation Set by using the Right Mouse context menu from the Outline view or by choosing Delete->Correlation Set from the Process Menu.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓What is Correlation
 - ✓Message Properties
 - ✓Correlation Sets
 - ✓Working with Correlation Sets
 - Using Correlations Sets

Using Correlation Sets Overview

- Can be used whenever messages are exchanged with partners
 - Core interaction activities
 - `invoke`, `receive`, and `reply` activities
 - `onMessage` variant of `pick` activity or `onEvent` variant of `eventHandler`
- **correlations** element nested within the above activities
 - Used to specify which correlation sets occur in the messages being exchanged

27 Copyright © 2004-2008 Active Endpoints, Inc.



Correlation Sets can be used whenever you need to match inbound messages to a specific process instance. They can be used with `Invoke`, `Receive`, or `Reply` activities or with an `onMessage` variant of a `Pick` or an `onEvent` variant of an `EventHandler`. Recall that `Pick` and `onEvent` are simply specialized types of `Receive` activities. The “`correlations`” element in those activities will contain the `Correlation Set` (or sets) that the activities are part of.

Using Correlation Sets Syntax

```
<correlations>?  
  <correlation set="NCName" initiate="yes|join|no"?  
    <!-- Following attribute used only  
         for invoke activity -->  
    pattern="request|response|request-response"? />+  
</correlations>
```

28 Copyright © 2004-2008 Active Endpoints, Inc.



Here is the syntax for a Correlation Set. We have two attributes – initiate and pattern.

Using Correlation Sets

- **initiate** attribute

- Each usage of a correlation set on an activity requires the setting of the `initiate` attribute

- May contain one of the following values:

- “yes” – correlation set values are populated with the message data being transmitted or received in this activity

- “join” - the activity must attempt to initiate the correlation set if it is not yet initiated

- “no” – the activity does not initiate correlation

29 Copyright © 2004-2008 Active Endpoints, Inc.



The "Initiate" attribute determines whether or not correlation will be initiated by the activity under scrutiny. The available options are: yes, no and join.

a.) Yes: When the initiate attribute is set to "yes", the related activity **MUST** attempt to initiate the correlation set. If the correlation set is already initiated, the standard fault `bpel:correlationViolation` **MUST** be thrown.

b.) Join: When the initiate attribute is set to "join", the related activity **MUST** attempt to initiate the correlation set, if the correlation set is not yet initiated. If the correlation set is already initiated and the correlation consistency constraint is violated, the standard fault `bpel:correlationViolation` **MUST** be thrown. This is useful for multi-start activities or any scenario where more than one activity (multiple receives) may initiate the correlation set.

c.) No: When the initiate attribute is set to "no" or is not explicitly set, the related activity **MUST NOT** attempt to initiate the correlation set. If the correlation set has not been previously initiated, the standard fault `bpel:correlationViolation` **MUST** be thrown. If the correlation set is already initiated and the correlation consistency constraint is violated, the standard fault `bpel:correlationViolation` **MUST** be thrown.

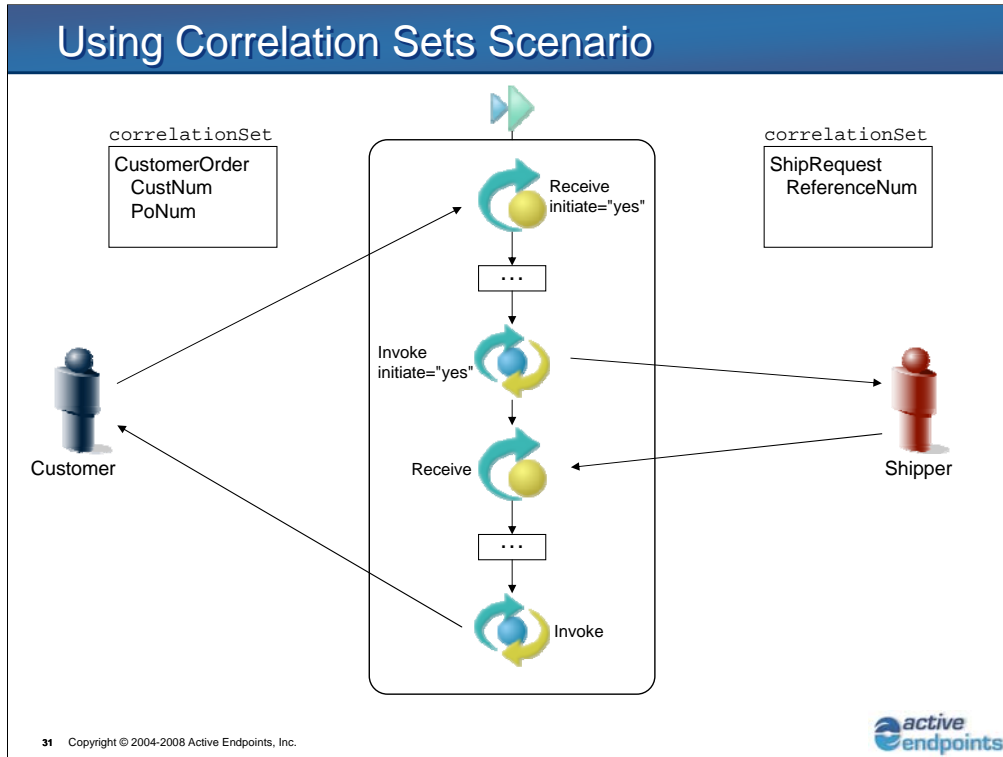
Using Correlation Sets

- **pattern** attribute
 - Additional attribute used for `invoke` activities
 - Indicates when a correlation set should be used
- May contain one of the following values:
 - “request” – correlation is applied to the outbound message
 - “response” – correlation is applied to the inbound message
 - “request-response” – correlation is applied to both the outbound and inbound messages

30 Copyright © 2004-2008 Active Endpoints, Inc.



The Invoke activity has an additional attribute called “pattern” that recognizes the activity's dual nature. Invokes can receive incoming messages, transmit outgoing messages, or do both. The *pattern* attribute describes at what times during the process the correlation set should be used. If it is set to *request* then it applies only to the outbound message. In this case the input variable going to the invoked service is validated against the correlation set to ensure that the correct data is being transmitted. In the case of *response*, it ensures that the output variable coming from the invoked service matches the data in the correlation set. In the case of *request-response* the data is validated both on the way out *and* on the way in.



Now, let's take a look at a typical Correlation Set example. There are three actors in this scenario: the Customer, a Process and the Shipper. The Customer is ordering from the Process and the Process is then sending the Customer's order out to the Shipper who will then send it to the Customer.

To keep the communications synchronized, there are two Correlation Sets:

CustomerOrder is the Correlation Set for the Customer/Process conversation, using "CustNum" and "PoNum" data

ShipRequest is the Correlation Set for the Process/Shipper conversation, using "ReferenceNum" data

The Receive activity at the top has the initiate value set to "yes" because it is the first Receive and will be starting the correlation for the Customer-Process conversation when the first Customer message arrives. In between this initial Receive and the following Invoke there may be other activities that are executed, as needed. As we move down the Sequence, we see that the first Invoke activity has the initiate value set to "yes" because it will be starting the Correlation for the Process-Shipper conversation when it sends the first message to the Shipper. The second Receive activity does not have the initiate value set to "yes" because it is coming into a conversation that is already in progress, the Customer/Process conversation. After the second Receive activity, there may be other activities that are executed, as needed. Finally, the second Invoke activity does not have the initiate value set to "yes" because it, too, is coming into a conversation that is already in progress, the Process/Shipper conversation.

Using Correlation Sets Example

```

<sequence>
  <receive ...>
    <correlations>
      <correlation set="CustomerOrder" initiate="yes" />
    </correlations>
  </receive>
  <invoke ...>
    <correlations>
      <correlation set="ShipRequest" initiate="yes"
        pattern="request" />
    </correlations>
  </invoke>
  <receive ...>
    <correlations>
      <correlation set="ShipRequest" />
    </correlations>
  </receive>
  <invoke ...>
    <correlations>
      <correlation set="CustomerOrder" pattern="request" />
    </correlations>
  </invoke>
</sequence>

```

32 Copyright © 2004-2008 Active Endpoints, Inc.



Here is the Correlation Set syntax for the example we just examined.

Two Correlation Sets are defined:

Two activities (one Invoke and one Receive) are in the ShipRequest set

Two activities (one Invoke and one Receive) are in the CustomerOrder set

Starting at the top, we have the first Receive activity, which begins the conversation between the Customer and the Process. We see that it is part of the “CustomerOrder” correlation set and that the initiate attribute value is set to “yes”, which means that it will start the correlation for the Customer/Process conversation. Next is the first Invoke activity, which begins the conversation between the Process and the Shipper. We see that it is part of the “ShipRequest” correlation set and that the initiate attribute value is also set to “yes”, which means that it will start the correlation for the Process/Shipper conversation. Following the first Invoke is the second Receive, which is part of the “ShipRequest” correlation set, and that there is no “initiate” reference, which means it is using the default value, which is “no”. This activity is part of a conversation that is already in progress, started by the first Invoke. Following the second Receive is the second Invoke, which is part of the “CustomerOrder” correlation set, and that there is no “initiate” reference, which means it is using the default value, which, again, is “no”. This activity is also part of a conversation that is already in progress, in this case started by the first Receive.

Using Correlation Sets Semantics

- After a correlation set is initiated the values become constant
 - A correlation set can be initiated only once during the lifetime of the scope it belongs to
 - Subsequent related messages must have property values identical to the initial correlation set

33 Copyright © 2004-2008 Active Endpoints, Inc.



A Correlation Set can only be initiated once during the lifetime of the scope it is in, and after a Correlation Set is initiated, the values become constant. Each subsequent message must contain the exact same data for the message to be correlated correctly. You should think of the correlation data more as a late-bound constant than as a variable.

Adding Correlation to an Invoke Activity

- On the Process Editor canvas, select an invoke activity

| Property | Value |
|----------------|--|
| Activity | |
| Activity Name | CreateSalesOrder |
| Activity Type | Invoke |
| Comment | |
| Correlations | (none) ... |
| Input Variable | SalesOrderRequest |
| Operation | CreateSalesOrder |

| Correlation Set | Initiate | Pattern |
|----------------------------|----------|----------|
| OrderManagerCorrelationSet | Yes | Response |
| | | |
| | | |

34 Copyright © 2004-2008 Active Endpoints, Inc.

So, once we have created a Correlation Set, how do we associate it with an activity? Select the Activity, go to Properties View, place your cursor in the Correlations field and then click the ellipses (...) to open the dialog box. (Note that the ellipsis only appears when you click on the field.)

In the Correlation Set dialog box...

set the initiate value to yes/no/join by selecting from the droplist

set the pattern value to receive/reply/receive-reply by selecting from the second droplist

Note: As was discussed previously, the pattern attribute droplist is available ONLY when setting the Correlation Set for Invoke activities.

Adding Correlation to Other Interaction Activities

- On the Process Editor canvas, select a **receive, reply, onMessage, or onEvent**

WaitForNotificationFromOrderManager

AccessDataForOrderManagerAcknowledgment

| Property | Value |
|------------------|--|
| Activity | |
| Activity Name | WaitForNotificationFromOrderManager |
| Activity Type | Receive |
| Comment | |
| Correlations | (none) ... |
| Create Instance | No |
| Message Exchange | |

Correlation Sets

Enter all Correlation Set information for the activity.

| Correlation Set | Initiate |
|----------------------------|----------|
| OrderManagerCorrelationSet | No |
| | |
| | |
| | |

35 Copyright © 2004-2008 Active Endpoints, Inc.

How do we add a Correlation Set to another type of activity (i.e., a “non-Invoke”)? We select the Activity, go to its Properties View, put our cursor in the Correlations value field and then click the ellipses (...) to open the dialog box. In the Correlation Set dialog, set the initiate value to yes/no/join by selecting from the droplist. (Recall that there will be no pattern attribute droplist when working with non-Invoke activities.)

Lab 6 – Correlation

- Overview of Lab Exercises
 - Define Correlation Sets
 - Customer / Process conversation
 - Process / Order Manager conversation
 - Use the Correlation Sets in the appropriate activities

36 Copyright © 2004-2008 Active Endpoints, Inc.



The next Lab in the BPEL Fundamentals class is Lab #6. In this lab we will create two Correlation Sets:

- one handles Customer-to-Process traffic
- one handles Process-to-OrderManager traffic

Once we have defined these two correlation sets, then we will associate the proper activities with each.

Unit Summary

- Now you are familiar with:
 - What is Correlation
 - Message Properties
 - Correlation Sets
 - Working with Correlation Sets
 - Using Correlations Sets