



This is Unit #5 of the BPEL Fundamentals course.

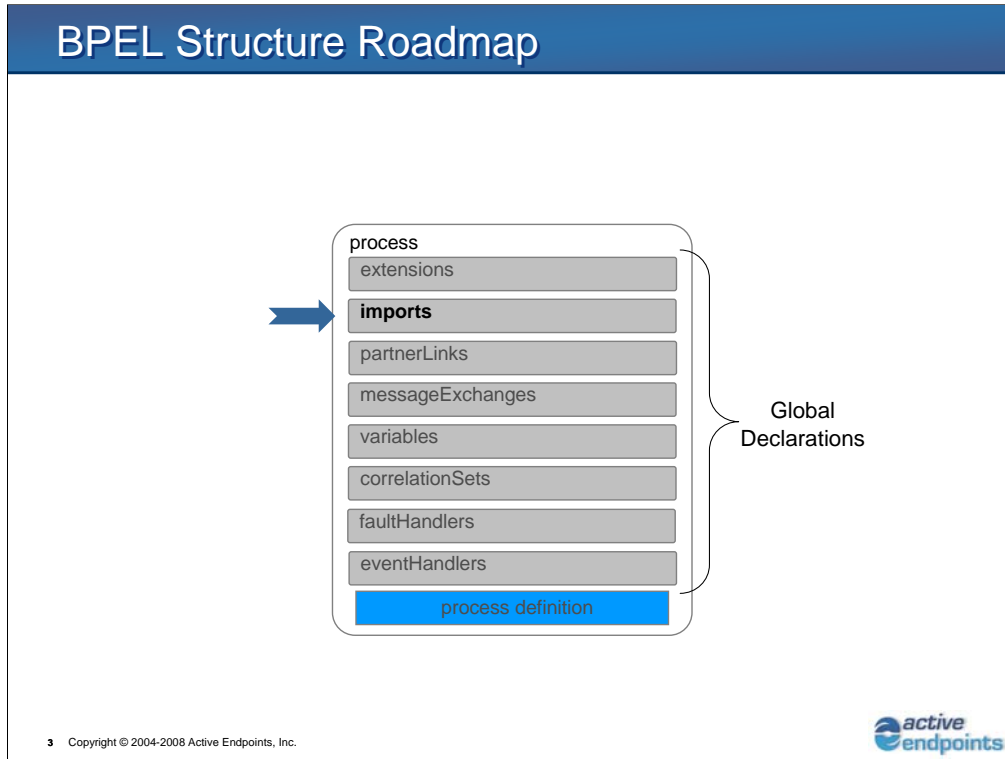
Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - Imports
 - Working with Imports
 - Partner Links
 - Working with Partner Links
 - Variables
 - Working with Variables

2 Copyright © 2004-2008 Active Endpoints, Inc.



In past units we looked at ActiveVOS Designer, Workspaces and Projects and then at our BPEL Processes. Now, let's take a look at the a process' Essential Global Declarations.



Imports are part of the Global Declarations for our process.

import Overview and Syntax

- Used to declare a dependency on external XML Schema or WSDL definitions
 - Any number of `import` elements may appear as children of the `process` element, before any other child element.
 - Each `import` element contains one mandatory and two optional attributes.

```
<import namespace="anyURI"?  
         location="anyURI"?  
         importType="anyURI" />*
```

4 Copyright © 2004-2008 Active Endpoints, Inc.



Imports are used to declare a dependency based on external definitions contained in WSDL and/or Schema files located elsewhere. You can have as many Import elements as you want, and they are considered children of the process level, which is why they are located at the top of the file. Both the Namespace and Location attributes are optional, but the Import type is required (and you can have more than one.) We'll cover Import types in later slides.

import Example

BPEL Fragment

```
<process>
  ...
  <import importType="http://schemas.xmlsoap.org/wsdl/"
    location="../wsdl/Retailer.wsdl"
    namespace="http://docs.active-endpoints.com/..." />
  ...
</process>
```

5 Copyright © 2004-2008 Active Endpoints, Inc.



Here we see an import type reference for a WSDL file. Note the relative location of the "Retailer.wsdl" file, followed by the Import's Namespace.

import Semantics

- Must have an **import** for all referenced schema and WSDL files
- Specific type URIs must be used for WSDL and Schema files
 - For WSDL files the `importType` must be `http://schemas.xmlsoap.org/wsdl/`
 - For Schema files the `importType` must be `http://www.w3.org/2001/XMLSchema`

6 Copyright © 2004-2008 Active Endpoints, Inc.



We need to have an import statement for all WSDL and Schema file references that are not part of the project. Unlike namespaces (whose URLs are need not be resolved), these locations must be resolved in order for the system to be able to find the file's definitions. Note also that WSDL and Schema files each have their own import type.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Imports
 - Working with Imports
 - Partner Links
 - Working with Partner Links
 - Variables
 - Working with Variables

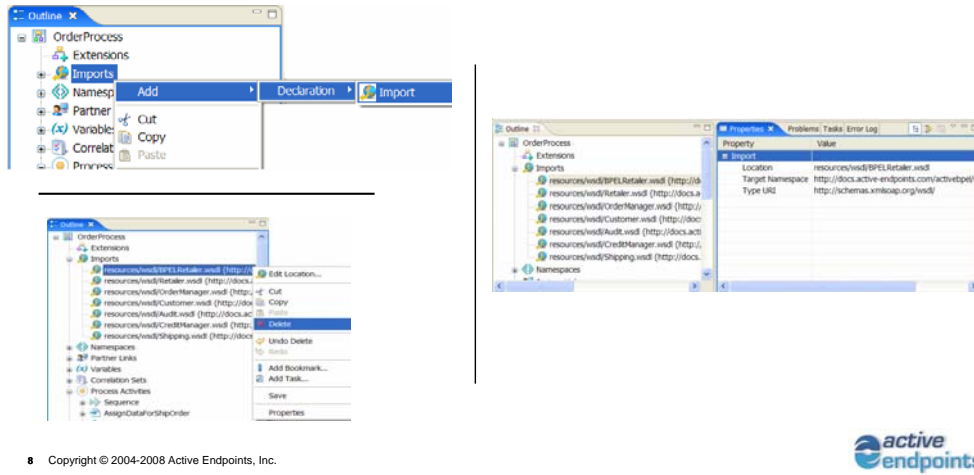
7 Copyright © 2004-2008 Active Endpoints, Inc.



So now that we know we have to import WSDL and Schema files in order to access their contents, how do we work with them in the Designer?

Working with Imports – Adding and Deleting

- Imports are added to a process definition either
 - Automatically via completion of a wizard
 - Manually via the Outline view or Process menu



ActiveVOS Designer's Wizards will add import references automatically as you move through them. You can also add imports manually through either the process menu or by using the Right Mouse menu on the imports node of the Outline View. However they are created, Imports can always be seen in the Outline View and their details can be examined in the Properties View. Imports can be deleted by selecting the import and using the RM menu's Delete item. Note that for the latest definitions you may have to refresh the Outline's Imports section after changes have been made to the process or to the underlying WSDL and Schemas.

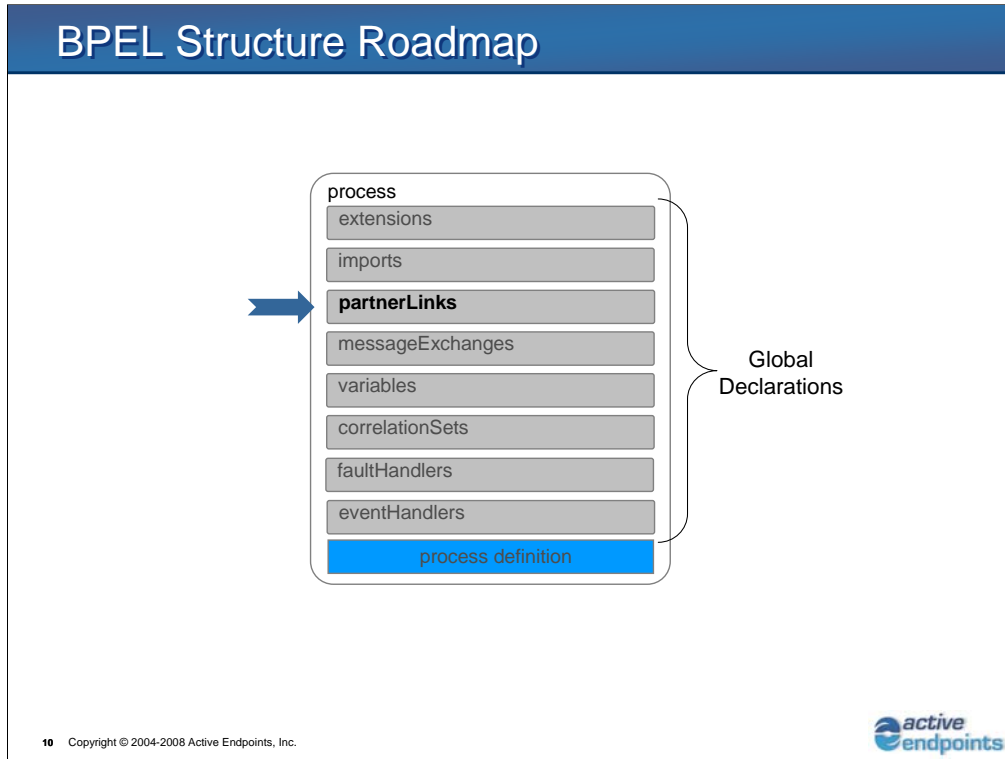
Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Imports
 - ✓ Working with Imports
 - Partner Links
 - Working with Partner Links
 - Variables
 - Working with Variables

© Copyright © 2004-2008 Active Endpoints, Inc.



Now that we've looked at imports, let's examine Partner Link declarations.



Partner Links and the relationships they define are part of the Global Declarations section of the Process.

Partner Link Overview and Syntax

- Represents a logical connection between the business process and a partner provided service
 - From the viewpoint of the business process
 - Binds to a specific `portType` via its named `partnerLinkType`
 - States which role the process is playing and which role the partner is playing in the interaction

```
<partnerLinks>?
  <!-- Note: At least one role must be specified. -->
  <partnerLink name="NCName" partnerLinkType="QName"
    myRole="NCName"?
    partnerRole="NCName"?
    initializePartnerRole="yes|no"?>+
  </partnerLink>
</partnerLinks>
```

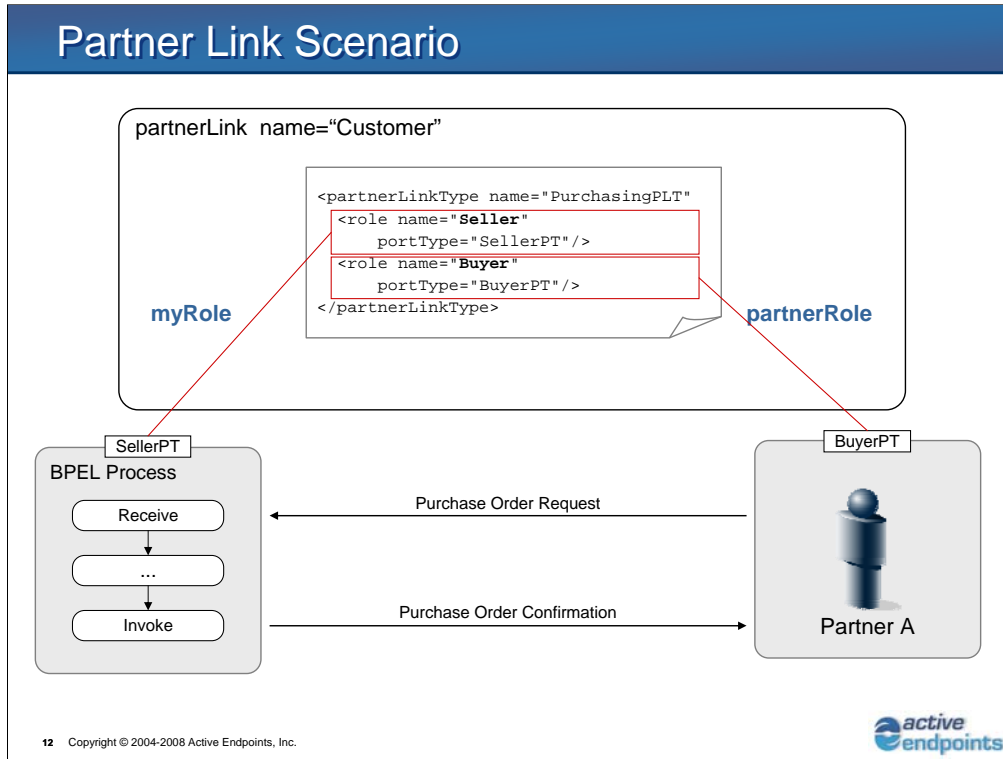
11 Copyright © 2004-2008 Active Endpoints, Inc.



A Partner Link represents a logical connection between your business process and the service(s) you are invoking. A PartnerLink is an instance of a partnerLinkType, and is based on the relationship between the partner and process, involving exactly one or exactly two PortTypes.

A PartnerLink includes:

- A Name for the PartnerLink (required)
- A partnerLinkType (required)
- A Role is optional
- A partnerRole is optional
- initializePartnerRole attribute has either of the values yes or no



Now we are looking at a PartnerLinkType from a third-party view. We have a PartnerLink named “Customer” as defined in the code snippet. We see the definition for the “Seller” role is based on the SellerPT PortType and that in this case the “Seller” is the BPEL Process. The Definition of the “Buyer” role is based on the BuyerPT PortType and that in this case the “Buyer” is Partner A.

Partner Link Example

BPEL Fragment

```
<process>
  ...
  <partnerLinks>
    <partnerLink name="Customer"
      partnerLinkType="PurchasingPLT"
      myRole="Seller"
      partnerRole="Buyer" />
  </partnerLinks>
  ...
</process>
```

On this slide we see the BPEL code implementing the PartnerLink seen on the previous slide.

Partner Link Semantics

- Must specify either one or both of the following attributes
 - myRole
 - partnerRole
- There can be multiple **partnerLinks** referencing the same **partnerLinkType**
 - Used to represent different instances of the same partnerLinkType

14 Copyright © 2004-2008 Active Endpoints, Inc.



A partnerLinkType must specify either a “myRole” or a “partnerRole” attribute, or it can specify both. You can have multiple partnerLinks, each one being an instance of the “partnerLinkType.”

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Imports
 - ✓ Working with Imports
 - ✓ Partner Links
 - Working with Partner Links
 - Variables
 - Working with Variables

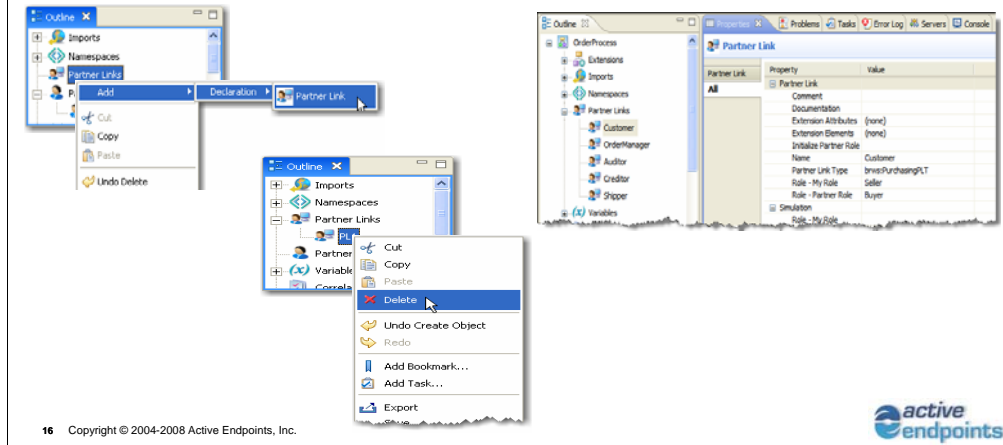
15 Copyright © 2004-2008 Active Endpoints, Inc.



Now that we've looked at PartnerLinks, let's see how we work with them in the Designer.

Working with Partner Links - Adding

- Partner links are added to a process definition either
 - Automatically via completion of a wizard
 - Manually via the Outline view or Process menu



A Process' partnerLinkTypes are added to the BPEL code just like any other XML elements. ActiveVOS Designer will add them automatically if you use a Wizard or you can use the Right Mouse functionality to add or delete them using the Outline View. Once they are part of the process, their details can be examined and edited in the Properties View.

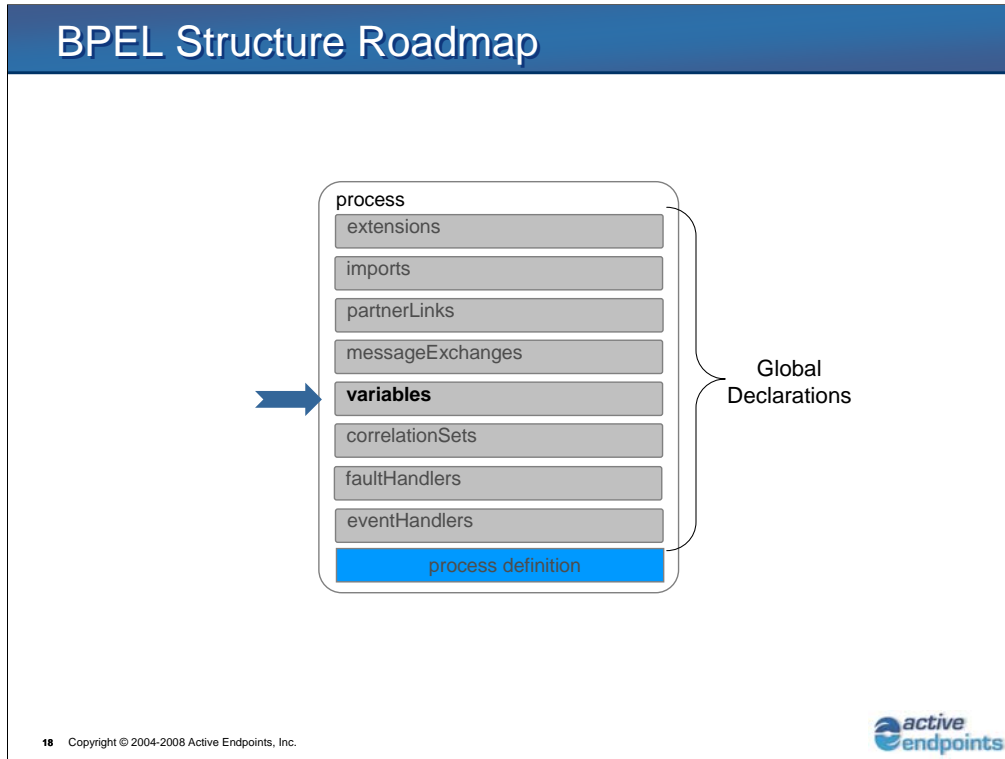
Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Imports
 - ✓ Working with Imports
 - ✓ Partner Links
 - ✓ Working with Partner Links
 - Variables
 - Working with Variables

17 Copyright © 2004-2008 Active Endpoints, Inc.



Now that we've looked at Imports and Partner Links, let's look at working with Variables in the Designer.



Here in the Global Declarations section of the process we can see process-level variables.

Variables Overview and Syntax

- Used to store and maintain the data of a business process
 - Messages exchanged to and from partner services
 - Arbitrary data used within the process
 - Never exchanged with partners
- Variables declared at the process level are global in scope

```
<variables>
  <variable name="BPELVariableName"
    messageType="QName" ?
    type="QName" ?
    element="QName" ?>+
    from-spec?
  </variable>
</variables>
```

19 Copyright © 2004-2008 Active Endpoints, Inc.



Variables store and maintain process data that is either exchanged with partners or used within the process itself. If we invoke a web service we will probably have to send it a message, so we need to store that message data before we send it and so we can respond to it later, if necessary. Note that Variables declared at the process level are considered to be global in scope.

Variable Types

- Variable can be one of the following three types:
 - messageType
 - element
 - type

20 Copyright © 2004-2008 Active Endpoints, Inc.



There are three Variable types: messageType, schema element, schema type.

messageType Variable Type

- Variable that can hold a WSDL message type definition

```
<variables>
  <variable name="newOrder" messageType="ord:orderRequest" />
</variables>
```

WSDL Fragment

```
<definitions>
  ...
  <message name="orderRequest">
    <part name="Input" element="po:purchaseOrder" />
  </message>
  ...
</definitions>
```

21 Copyright © 2004-2008 Active Endpoints, Inc.



Here we see a BPEL Process fragment for a *messageType* variable.

The variable name is "newOrder," the variable type is "messageType" and it is defined in the "ord" namespace as "orderRequest."

Below that we see a WSDL code fragment from the "ord" namespace that defines this variable's structure.

The Message's name is "orderRequest," and the Message has one part: the part name is "Input" and it is a simple element of the "purchaseOrder" type, as that type is defined in the "po" namespace.

Remember that all *messageType* variables should be initialized before using them in your process. Uninitialized variables will throw a fault during process execution.

element Variable Type

- Variable that can hold an XML Schema global element definition

```
<variable name="orderItems" element="po:orderItem" />
```

Schema Fragment

```
<schema>  
  <element name="orderItem">  
    <complexType>  
      <sequence>  
        <element name="partNumber" type="po:SKU" />  
        <element name="quantity" type="xs:positiveInteger" />  
        <element name="price" type="amount" />  
      </sequence>  
    </complexType>  
  </element>  
</schema>
```

22 Copyright © 2004-2008 Active Endpoints, Inc.



Here the Variable is "orderItems", and its type is the *Schema Element* called "orderItem", as that element is defined in the "po" namespace. Below is a Schema fragment from the "po" namespace that defines the "orderItem" complex type. This complex variable type that has a sequence with three ordered elements in it: partNumber, quantity, price.

type Variable Type

- Variable that can hold an XML Schema simple type
 - 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

```
<variables>
  <variable name="orderTotal" type="xs:decimal" />
</variables>
```

or

```
<variables>
  <variable name="barcodeNum"
    type="po:sku" />
</variables>
```

Schema Fragment

```
<simpleType name="sku">
  <restriction base="string">
    <pattern value="\d{3}-[A-Z]{2}" />
  </restriction>
</simpleType>
```

23 Copyright © 2004-2008 Active Endpoints, Inc.



Here we examine a variable called “orderTotal”, which is a schema type. The specific type is a *decimal*, one of the standard schema simple types. In the first code snippet, we see that "orderTotal" is a built-in simple type of decimal, as "decimal" is defined in the standard schema namespace "xs".

In the second code snippet, starting on the left, we see a Variable that is a derived type called “sku”, as "sku" is defined in the "po" namespace. In the schema fragment on the right we drill into the "sku" definition. This definition shows that "sku" is *derived* from a simple schema type, in this case a *string*. The “sku” type is based on a simple string, but with a pattern restriction that defines it as having: exactly 3 digits, followed by a dash, and then exactly two uppercase characters.

Variable Initialization

- A variable can be initialized by
 - Value in the `from-spec`
 - Receiving a message
 - Assignment
- Partial initialization may occur
 - When some, but not all, message parts of a `messageType` variable are assigned

24 Copyright © 2004-2008 Active Endpoints, Inc.



In a BPEL process, all variables *must* be initialized before being used, in any one of the three ways shown here.

- Using a value in the From-Spec
- Using a value received via an incoming message
- By Assignment of a value from within the process

Note that if a `messageType` has more than one part, it is also possible that the variable will not be *fully* initialized, and this can also cause a fault when those parts of the variable are accessed. ActiveVOS Designer allows you to initialize the variable from the process variables or outline views. Select it and go to the properties view and then select “initial value” to set this.

Variable Semantics

- Name of a variable must be unique within the process
- Variables must be declared before they can be used
- Variable parts must be initialized before being referenced in a query
- All parts of **messageType** variables must be initialized before being used in an outbound interaction

25 Copyright © 2004-2008 Active Endpoints, Inc.



Variable names must be unique within the process, and they must be declared and initialized before they can be used. Note that a variable in a lower level scope can be declared with the same name and type as a process level variable, and that this variable will be the one that is "seen" by scopes nested within that lower scope. Note also that messageType variables must be *fully* initialized before they can be used in an outbound interaction (because we don't want to send incomplete data to a partner.)

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Imports
 - ✓ Working with Imports
 - ✓ Partner Links
 - ✓ Working with Partner Links
 - ✓ Variables
 - Working with Variables

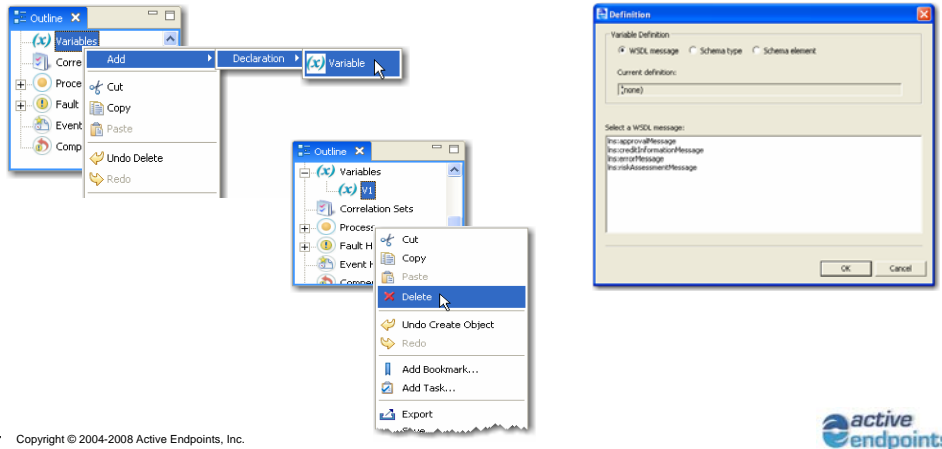
26 Copyright © 2004-2008 Active Endpoints, Inc.



Now that we've seen how variables work in a BPEL Process, let's see how to work with variables in the Designer.

Working with Variables - Adding

- Variables are added to a process definition either
 - Automatically via completion of a wizard
 - Manually via the Outline view or Process menu



Process-level variables can be added manually through either the Process menu or the Outline View, or you can add them automatically by using one of the Designer's wizards. They can be deleted from the Outline View or the Process menu, and their details can be examined and edited in the Properties View.

Unit Summary

- Now you are familiar with:
 - Imports
 - Working with Imports
 - Partner Links
 - Working with Partner Links
 - Variables
 - Working with Variables