

This is Unit #8 of the BPEL Fundamentals I course. In past Units we've looked at ActiveBPEL Designer, Workspaces and Projects and then we created the Process itself. Next, we looked at using Imports, PartnerLinks and Variables and how to create Interaction Activities in various ways. Finally, we looked at our first container activity, the Sequence. In this Unit we will take a look at a new activity, Assignments.

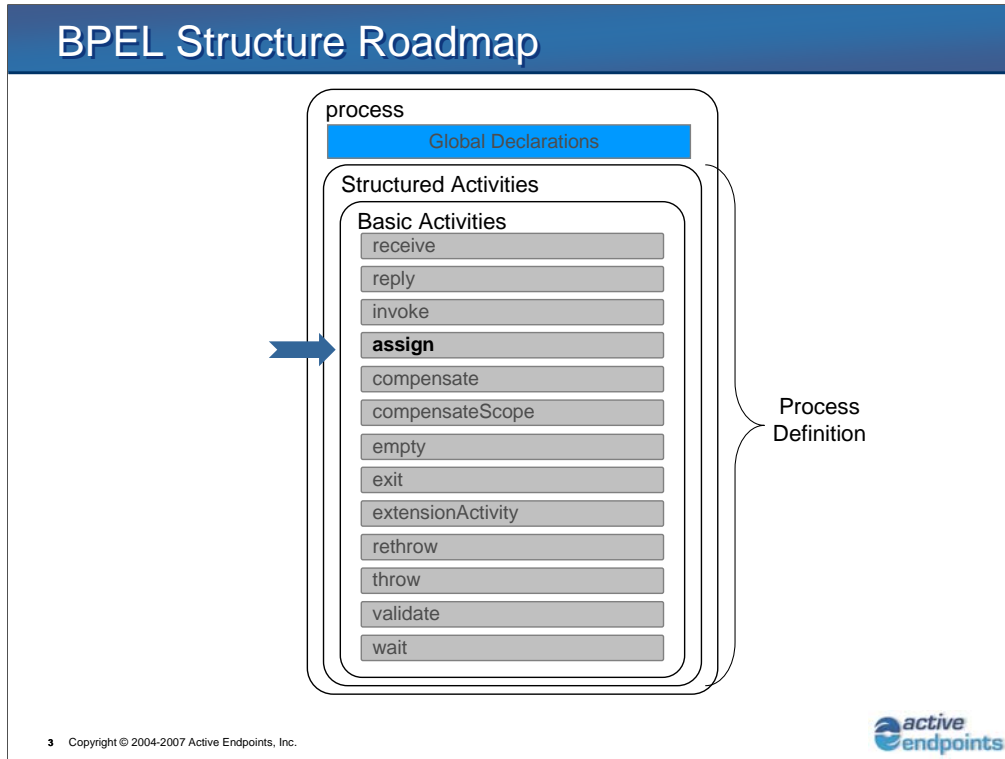
## Unit Objectives

- At the conclusion of this unit, you will be familiar with:
  - assign activity
  - Working with Copy Operations in an assign

2 Copyright © 2004-2007 Active Endpoints, Inc.



In this Unit we'll look at the Assignment activity and at the Copy Operations it uses.



Here is our BPEL Structure Roadmap, and we see that the Assign is considered to be a Basic activity in BPEL and is part of the process definition.

## assign Activity Overview

- Prescribes the movement of data within a BPEL process
  - Between variables, expressions, and partner links
- Includes one or more
  - copy commands
    - From a source to a destination
  - Extension assignment operations

4 Copyright © 2004-2007 Active Endpoints, Inc.



Assign Activities, using their internal Copy operations, control the movement of data in a BPEL process. One Assign activity can have many internal Copy commands, each of which copies a type compatible value from a source to a destination. So, the purpose of an Assign activity is to copy data from one place to another, or from many places to many other places. Assign activities can copy data to and from Variables, Expressions and PartnerLinks, as needed.

An Alternate way to move data from one place to another is to create an **extension assign** activity which gives Vendors (i.e., *not* developers) the option of customizing the Assign activity to accept more than just the standard copy operations. Vendors can create their own Copy (or other operations) that are then used as extension Assignment operations. Extension Assign activities are not standard BPEL and will not be covered in this course.

## assign Activity Syntax

```

<assign validate="yes|no"?
  standard-attributes>
  standard-elements
  (<copy keepSrcElementName="yes|no"?
    ignoreMissingFromData="yes|no">
    from-spec
    to-spec
  </copy>
  |
  <extensionAssignOperation>
    assign-element-of-other-namespace
  </extensionAssignOperation>)+
</assign>

```

5 Copyright © 2004-2007 Active Endpoints, Inc.



Here is the Assign activity syntax. Assign Activities can have all of the standard Attributes and Elements.

The purpose of the Validate attribute is to check all of the variables being modified in the Assign against their definitions. If an error is encountered the “bpel:invalidvariables” fault will be thrown. (Note that including this attribute can be resource intensive.) The Validate attribute is optional and takes a “yes” or “no” value. If this attribute is not used, the default value is “No.”

Within each Assign activity are the individual Copy operations. Remember that one assign can have many Copy Operations and optionally one or more extension assign Operations. The From and To specifications of the Copy Operation tell us what we are copying from – the source - and what we are copying to – the target. Each one of these operations has two optional attributes. The first of these “keepSrcElementName”, which means the copy will copy both the data *and* the variable’s name to the target. The second is “ignoreMissingFromData” which means that if some of the source’s “FromParts” are not set, that’s still OK, and the “bpel:SelectionFailurefault” will not be thrown.

## Valid forms of the `from-spec`

- `<from variable="BPELVariableName" part="NCName"?>`  
`<query queryLanguage="anyURI"?>?`  
`queryContent`  
`</query>`  
`</from>`
- `<from partnerLink="NCName"`  
`endpointReference="myRole|partnerRole"/>`
- `<from variable="BPELVariableName" property="QName"/>`
- `<from expressionLanguage="anyURI"?>`  
`expression`  
`</from>`
- `<from><literal>literal value</literal></from>`
- `<from/>`

6 Copyright © 2004-2007 Active Endpoints, Inc.



Now, let's look at the Copy Operations and the various forms of the Source and Destination. We'll start with the Source end.

There are different ways we can copy from a **Variable**, determined by the query used (first example)

- messageType: specify the variable to copy the entire variable
- specify a QUERY and a PART to copy only a part, for either the From or the To. Note that Part is optional, only used if you are copying a part.
- type (schema): use the variable name – *cannot use a part!*
- element (schema): use query to find data – *cannot use a part!*

We can Copy from a **Partner Link** (second example) where the name is a partnerLink that is in scope. Note that in the case of From specs the Role must be specified, where “myRole” is the process and “partnerRole” is the partner’s Endpoint reference. By inference, if this is used, the partnerLink must define the role that is referred to.

We can Copy from a Variable’s **Property** (third example). Since a single variable can have multiple properties, you must specify which one you are copying From.

We can Copy from an **Expression** (fourth example) essentially returns a value, and uses the default language – XPath 1.0 – if no language is specified.

We can Copy from a **Literal** (fifth example.)

And, finally, we can copy using an **ExtensionAssign** activity – shown by a blank (fifth example) - which is used to show that Copy can be extended by vendors, as we saw earlier when we discussed the Assign activity itself.

## Valid forms of the to-spec

- `<to variable="BPELVariableName" part="NCName"?>`  
    `<query queryLanguage="anyURI"?>`  
        `queryContent`  
    `</query>`  
    `</to>`
- `<to partnerLink="NCName" />`
- `<to variable="BPELVariableName" property="QName" />`
- `<to expressionLanguage="anyURI"?>`  
    `expression`  
    `</to>`
- `<to/>`

7 Copyright © 2004-2007 Active Endpoints, Inc.



Now, let's look at the Copy Operation's destination, via the TO specification.

We can Copy TO a **Variable** (first example)

- for messageType: specify the variable to copy the entire variable, or specify a PART AND A query to retrieve just a part
- type (schema): use the variable name – *cannot use a part!*
- element (schema): use query to find data – *cannot use a part!*

We can copy to a **PartnerLink** (second example) where the name is the partnerLink that is in scope. Note that in the case of the To spec it must be the partner's Endpoint reference because assignment is only possible to the partnerRole. Note also that, by inference, the partnerRole must be defined in the partnerLink. Note: The fundamental use of endpoint references is to serve as the mechanism for dynamic communication of port-specific data for services. An endpoint reference makes it possible in WS-BPEL to dynamically select a provider for a particular type of service and to invoke their operations.

We can Copy to a Variable's **Property** (third example)

We can Copy to an **Expression** (fourth example) essentially is used to select a value that will be the target.

And, finally, we can Copy to an **ExtensionAssign** activity (fifth example), which is used to show that the Assign activity can be extended by vendors, as we saw earlier.

Note that there is no "Literal" included in the To spec, as Copying to a Literal does not make sense.

## Copying Variables

- Copy values from one variable to another variable
  - Specify the variable attribute in the `from-spec` and `to-spec`
  - Valid only if both variables are of the same type

```
...  
<variables>  
  <variable name="newOrder" messageType="ord:orderRequest" />  
  <variable name="orderConf" messageType="ord:orderRequest" />  
</variables>  
  
<assign>  
  <copy>  
    <from variable="newOrder" />  
    <to variable="orderConf" />  
  </copy>  
</assign>  
...
```

8 Copyright © 2004-2007 Active Endpoints, Inc.



Here is an example showing how we Copy a value from one Variable to another. First, the Variables involved have to be defined. In this example we have two – “newOrder” and “orderConf” – both of which are messageType variables. In our Assign activity, we have a single Copy Operation that copies the value from “newOrder” to “orderConf.” Note that both variables used in this example are of the same type: “orderRequest”

## Copying Variables - messageType

- Can further specify which message part to copy
  - Specify the `part` attribute in the `from-spec` and `to-spec`

```

<variables>
  <variable name="newOrder" messageType="ord:orderRequest" />
  <variable name="orderConf" messageType="ordmgr:orderConfirmation" />
</variables>
...
<assign>
  <copy>
    <from variable="newOrder" part="PartsOrder" />
    <to variable="orderConf" part="PartsConfirmed" />
  </copy>
</assign>

```

© Copyright © 2004-2007 Active Endpoints, Inc.



Here is a second example, where we are Copying part of one variable to part of another variable. Note that we are not Copying the whole variable. To start with, we have two variables that each have parts. Note that the variables themselves are both messageType variables, but their definitions are different types and come from different Namespaces. Inside the Assign activity, we use a Copy from the variable “newOrder” part called “PartsOrder” to the “orderConf” variable’s “PartsConfirmed” part. Notice that the two parts, while they may come from variables that are different types, are **themselves** of the same type. So, we see that if the variables are different types we can’t copy from one to the other directly. But, here they both have *parts* that **are** the same type, so that we can do. For an analogy, it is like copying a field from a struct or record type into the field of another struct or record type.

## Copying Variables - Selecting XML Data

- Can also select XML data from **messageType** and **element** variables
- Specify the **query** attribute in the **from-spec** and **to-spec**
  - Uses the specified query language
    - Default language is XPath 1.0

10 Copyright © 2004-2007 Active Endpoints, Inc.



We can select XML data from Variables defined as MessageType or as Schema Elements and copy it to a valid destination. Queries use a specific query language, and the default language is XPath 1.0. If you use a different language, like JavaScript 1.5, you must specify it, and it must be supported by the engine you are using.

## Copying Variables - Selecting XML Data

```

...
<variables>
  <variable name="newOrder" messageType="ord:orderRequest" />
  <variable name="itemCount" type="xsd:positiveInteger" />
</variables>

<assign>
  <copy>
    <from variable="newOrder" part="Input">
      <query>count(/po/items/item)</query>
    </from>
    <to variable="itemCount" />
  </copy>
</assign>
...

```

11 Copyright © 2004-2007 Active Endpoints, Inc.



Here we are using a query to pull data from a complex variable as part of a Copy Operation.

In this example we are Copying From the variable “newOrder’s” part called “Input” to a variable called “itemCount.” Note that they are of different types, i.e., “itemCount” is a positiveInteger and “newOrder” is a messageType. In this case we cannot do a Part-to-Part copy because the variable “itemCount” is not a messageType variable, so it has no parts. What we can do is Copy from the “newOrder” variable’s “Input” part, and then massage the query result to change the type. In this example we are using a Count function, which takes the query result as an argument and returns a positiveInteger type. This positiveInteger type is the same type as the “itemCount” variable, so now the Copy is valid.

Note: if the return value of the Count function is “0” it is NOT a positiveInteger type and executing this line of code would throw a fault.

## Copying Variables using XPath expression

```
...  
<variables>  
  <variable name="newOrder" messageType="ord:orderRequest" />  
  <variable name="itemCount" type="xsd:positiveInteger" />  
</variables>  
  
<assign>  
  <copy>  
    <from>count($newOrder.Input/po/items/item)</from>  
    <to variable="itemCount" />  
  </copy>  
</assign>
```

12 Copyright © 2004-2007 Active Endpoints, Inc.



Here we see the same Copy operation as in the previous example, but with a different syntax. Using Dollar Sign syntax, we see that the messageType variable “newOrder” has a part called “Input” that can be referred to this way: “\$newOrder.Input/po/items/item”

Where “newOrder” is the message, “Input” is the message part, and the result of the “/po/items/item” query is fed to the Count function, which returns a positiveInteger. Otherwise, the Copy operation remains the same as the one we saw previously.

## Copying Expressions

- Allows processes to perform simple computations on variables
  - Specify the `expression` attribute in the `from-spec`
  - Uses the specified expression language
    - Default language is XPath 1.0
      - Any valid XPath expression can be used

```
<assign>
  <copy>
    <from>substring-after('2003/09/25','/')</from>
    <to variable="shipMonthDay" />
  </copy>
</assign>
```

13 Copyright © 2004-2007 Active Endpoints, Inc.



In this example we are copying from an **expression**, which allows us to do some simple manipulation of a variable before we copy it. Here we copy from an expression using the “substring-after” function to get a specific part of the target string... where ‘/’ (forward slash) is the delimiter. Here the expression language is assumed to be XPath1.0 by default. If it were a different language we would have to specify that. Note that there can be a difference between the *Query* language and the *Expression* language.

FROM THE USER GUIDE:

***substring-after(string, string)***

Returns that part of the first string which occurs after the first occurrence of the second string

## Copying Literal Values

- Allows any literal value to be specified as the source to assign to a destination
  - Useful to initialize variables
  - Data type of the literal value MUST match the data type of the destination

```

<assign>
  <copy>
    <from>
      <literal>
        <PartsOrder xmlns="urn:Parts">
          <Item>
            <productNumber />
            <quantity />
            <price />
          </Item>
        </PartsOrder>
      </literal>
    </from>
    <to variable="OrderItems" part="PartsOrder" />
  </copy>
</assign>

```

14 Copyright © 2004-2007 Active Endpoints, Inc.



Another option is copying using a Literal value. BPEL allows any Literal to be specified as a source, always in the From spec. As we stated before, copying to a Literal does not make any sense. Here we use the Copy to initialize a messageType variable's part with the text enclosed by the Literal's opening and closing tags. After the closing of the From tag (line #13), we see that the destination - which in this case is the To variable - is called "OrderItems" specifically its part called "PartsOrder."

## Copying Partner Links

- Allows dynamic manipulation of the endpoint references associated with partner links
  - Specify the `partnerLink` attribute in the `from-spec` and `to-spec`
    - Can obtain both the `myRole` and `partnerRole` endpoint references in the `from-spec`
    - Sets only the `partnerRole` endpoint reference in `to-spec`
  - Endpoint reference is defined using WS-Addressing

```
<assign>
  <copy>
    <from variable="Configuration"
          part="cfg"
          query="/ns:Configuration/ns2:endPointRef" />
    <to partnerLink="RetailerPL" />
  </copy>
</assign>
```

15 Copyright © 2004-2007 Active Endpoints, Inc.



We can use Copy Operations on Partner Links, which allows us to copy one link's attributes and/or values to another. We can also use the Copy Operation to copy `myRole` or `partnerRole` values to a `PartnerLink`. Performing this type of Copy prior to invoking a web service makes sense. In that situation, we can Copy endpoint reference information to a partner link, for example, to dynamically set endpoint references at runtime.

A brief excerpt from the ActiveBPEL Designer's User Guide, Page 212:

*During the deployment of a process, each role defined in a partner link is assigned an endpoint reference. The fundamental use of endpoint references is to serve as the mechanism for dynamic communication of port-specific data for services. An endpoint reference makes it possible in WS-BPEL to dynamically select a provider for a particular type of service and to invoke their operations. An endpoint indicates a specific location for accessing a Web service using a specific protocol and data format. An endpoint reference conveys the information needed to access a Web service endpoint. Using endpoint references in deployment makes possible the dynamic selection of a service partner.*

## Extending the `assign` activity

- Two ways for a BPEL Vendor to provide additional variable manipulation capabilities
  - Extend the `assign` activity:
    - using an `<extensionAssignOperation>`
    - elements defined here must belong to a namespace different from the WS-BPEL namespace
  - Extend the `copy` operation:
    - Used to show that the `from-spec` and `to-spec` are extensible

16 Copyright © 2004-2007 Active Endpoints, Inc.



We previously discussed Extending the Assign Activity. If we Extend the Assign activity, we must specify `<extensionAssignOperation>` to make it our own. Note that the definition must be in a Namespace other than the default WS-BPEL namespace. We can also extend the Copy operation, which means in essence that the From and To specs are extensible, since they define the Copy operation itself.

## Complex XML Manipulation

- Complex business processes often require that messages be transformed from one format to another for use by the services in the process
- As described previously, `copy` operations in BPEL allow for the basic manipulation of XML but lack the ability to perform this type of complex data transformations
- To solve this issue BPEL provides a function to execute standard XSL Transformations

### –`bpel:doXslTransform()`

- XPath 1.0 extension function
- Used to perform complex XML manipulation

```
object bpel:doXslTransform(string, node-set, (string, object)*)
```

- Parameters:
  - string: URI naming the stylesheet to be used
  - node-set: source document for the transformation to be performed
  - (string,object): optional parameters providing a name-value pair for XSLT parameters
- Returns either a text or element node

17 Copyright © 2004-2007 Active Endpoints, Inc.



Up to this point we've seen simple ways to Copy data from one place to another. But, a common pattern in a BPEL process involves receiving an XML document from one service, converting it to form a new request message, and then sending the request to another service. To accomplish this, BPEL 2.0 provides a extension function for complex manipulation of XML called "`doXslTransform()`." The purpose of this function is to copy content from one XML document, Transform the content, and then Copy the content into another xml document.

Syntax: `object BPEL:doXslTransform( string, node-set, (string, object)* )` // see ActiveBPEL Designer's Help or the User's Guide for more details.

Note: eXtensible Style Sheet Language Transform, comes from a W3C recommendation.

## XSL Transformation example

```

<variables>
  <variable name="A" element="foo:AElement" />
  <variable name="B" element="bar:BElement" />
</variables>
...
<sequence>
  <invoke ... inputVariable="..." outputVariable="A" />
  <assign>
    <from>
      bpel:doXsltTransform("urn:stylesheets:A2B.xsl", $A)
    </from>
    <to variable="B" />
  </assign>
  <invoke ... inputVariable="B" ... />
</sequence>

```

18 Copyright © 2004-2007 Active Endpoints, Inc.



We have a variable “A” which is an element of the type “AElement” with the namespace prefix “foo.” Then, we have a variable “B” which is an element of the type “BElement” in the namespace with prefix “bar.” Our process uses a Sequence, which calls an Invoke. The Invoke has an inputVariable “...” and an outputVariable “A.”

We perform an Assign in which we use **doXsltTransform(“urn:stylesheets:A2B.xsl”, \$A )** to perform a complex manipulation of “A” before we write it into “B”. Variable “A” is the source, the doXsltTranform is executed by applying the stylesheet, and the result is sent to the target variable “B”. The first argument “urn:stylesheets:A2B.xsl” names the URI of the style sheet to be used. This will be verified by static analysis and if the stylesheet document is invalid or missing it will throw the “bpel:xsltStyleSheetNotFound” fault. The second argument is the source document for the transformation. It must contain a single element node or it will throw a “bpel:xsltInvalidSource” fault. Finally, we have an Invoke activity with “B” as its inputVariable.

## assign Activity Optional Attributes

- **validate** - used to validate all variables being modified by the activity
- **keepSrcElementName** - used to specify whether the element name of the destination will be replaced with the element name of the source
- **ignoreMissingFromData** - used to specify whether a `bpel:selectionFailure` is suppressed

19 Copyright © 2004-2007 Active Endpoints, Inc.



Here are the optional attributes for the Assign activity. The Validate attribute, if it has a value of "Yes", validates all variables modified against their schema type definitions.

**keepSrcElementName** is used to specify whether the target element's name is replaced with the source element's name.

FROM THE USER GUIDE PAGE 246:

*Use Keep Source Element Name to replace the element at the destination with a copy of the entire element at the source, including children and attribute properties. This attribute supports XSD substitution groups and choice.*

**ignoreMissingFromData** is evaluated when there is a BPEL selection failure because the From data is not complete or missing. If set to yes, then we suppress any faults that would otherwise be raised. This attribute is commonly used for optional data.

FROM USER GUIDE PAGE 248:

*Use **Ignore Missing From Data** to avoid a selection failure when copying data that may not exist in the source variable. If enabled for a copy operation, then any from-spec that selects zero elements, attributes, or text items causes the copy operation to have no effect. The to-spec for the copy operation is not evaluated and nothing changes in the target variable. If this attribute is not enabled, then the regular `bpel:selectionFailure` fault is raised if the from-spec selects zero items.*

## Selection rules

- Selection result of **from-spec** and **to-spec** must result in an element, attribute or text node
- **ignoreMissingFromData** Logic Table

returned nodes		ignoreMissingFromData	
from	to	"no"	"yes"
0	0	selectionFailure	no-op
0	1	selectionFailure	no-op
0	N	selectionFailure	no-op
1	0	selectionFailure	selectionFailure
1	1	copy	copy
1	N	selectionFailure	selectionFailure
N	0	selectionFailure	selectionFailure
N	1	selectionFailure	selectionFailure
N	N	selectionFailure	selectionFailure

20 Copyright © 2004-2007 Active Endpoints, Inc.



Here are the “ignoreMissingFromData” selection rules (i.e., whether to raise a fault.) This table shows what happens in each circumstance, based on the setting for ignoreMissingFromData.

The first two columns specify the number of Nodes returned in the From and the To spec. The second two columns specify the behaviour in each circumstance. That’s why the only one that works is 1 and 1 in each of the first two columns. It only works when a single node is copied to a single node. If the From returns no nodes, then it results in a “no-op”. In all other circumstances, it results in a selectionFailure fault.

## Message Variable Replacement logic

- **from-spec** copied exactly “as-is” to **to-spec**
  - Uninitialized parts in *from-spec* result in uninitialized parts in the *to-spec*
  - Original message parts in *to-spec* no longer available
- When **from-spec** and **to-spec** both evaluate to element nodes
  - Entire destination element is replaced including attributes and text nodes
  - Otherwise only the destination’s content is replaced

21 Copyright © 2004-2007 Active Endpoints, Inc.



The BPEL Standard rules for replacing data in a copy are defined in Appendix D of the BPEL 2.0 standard.

Just a quick summary here... If a fully initialized From is copied to the To, everything is copied as is. If there are any uninitialized parts in the From spec, this copy operation results in the exact same parts being uninitialized in the To spec. If From and To are both element nodes, the entire destination (To) node – e.g., its attributes and text nodes - is replaced. Otherwise only the destination element’s *content* is replaced.

From appendix D of the BPEL 2.0 spec... this topic was not clear in the original BPEL 1.0 spec, and was clarified in BPEL 2.0

Q. What is the difference between the node and the content being replaced?

A. The entire node means the content plus the attributes and elements of the node.

## assign Activity Semantics

- Values copied from the source (**from-spec**) to the destination (**to-spec**) MUST be of compatible data types

22 Copyright © 2004-2007 Active Endpoints, Inc.



Values copied from a From to a To must be of compatible types. If any incompatible types are detected during the assignment, the “`bpel:mismatchedAssignmentFailure`” fault is thrown. Now let’s look at some typical errors that result from problems in Assign Activities and Copy operations...

The following situations are considered type **incompatible**:

- 1.) the selection results of both the from-spec and the to-spec are variables of a WSDL message type, and the two variables are not of the same WSDL message type (two WSDL message types are the same if their QNames are equal).
- 2.) the selection result of the from-spec is a variable of a WSDL message type and that of the to-spec is not, or vice versa (parts of variables, selections of variable parts, or endpoint references cannot be assigned to/from variables of WSDL message types directly).
- 3.) the selection result of the from-spec is an EII, that of the to-spec is a Document EII of an element-based variable or an element-based part of a WSDL message-type-based variable, the `keepSrcElementName` attribute is set to “yes” and the name of the source element

## mismatchedAssignmentFailure Fault

- This fault is thrown when
  - Incompatible types are encountered in an assign activity

`bpel:mismatchedAssignmentFailure` 

23 Copyright © 2004-2007 Active Endpoints, Inc.



This fault results from mismatched types in a Copy operation. EX: trying to copy a date type to string type.

## selectionFailure Fault

- This fault is thrown when
  - A selection operation encounters an error in an Assignment's copy operation unless
    - ignoreMissingFromData attribute = "yes"
- If a given query selects a node set of a size other than one

`bpel:selectionFailure`



The selectionFailure fault can result from a Copy operation where the source is missing some data. Note that this fault can be suppressed by setting the "ignoreMissingFromData" attribute to YES, as we've seen previously. This fault can also result from a Copy where a Node size other than one causes the error. The Node might be zero or it might be repeating, i.e., greater than one. We saw this situation when we looked at the Copy Selection table a few slides ago.

## uninitializedVariable Fault

- This fault is thrown when
  - There is an attempt to access the value of an uninitialized variable

`bpel:uninitializedVariable`



25 Copyright © 2004-2007 Active Endpoints, Inc.




Remember that we have to initialize all variables, or a fault will be thrown when we attempt to access it. The fault above is thrown when there is an attempt to access the value of an uninitialized variable, or in the case of a message type variable, one of its uninitialized parts, unless the “ignoreMissingFromData” attribute is set to YES.

FROM ActiveBPEL Designer's USER GUIDE PAGE 524:


*Thrown when there is an attempt to access the value of an uninitialized variable, or in the case of a messageType variable, one of its uninitialized parts.*

## uninitializedPartnerRole Fault

- This fault is thrown when
  - There is an attempt to access the value of an uninitialized Partner Role

`bpel:uninitializedPartnerRole` 

26 Copyright © 2004-2007 Active Endpoints, Inc.



Just as we must initialize our variables before we use them, we must also initialize partner roles before we use them in a copy operation. If we don't, we get this fault.

FROM USER GUIDE PAGE 524:

*Thrown when an <invoke> or <assign> activity references a partner link whose partnerRole endpoint reference is not initialized*

FROM USER GUIDE PAGE 210:

*The **Initialize Partner Role** property is provided as a hint to the deployer of a process. This property does not affect the runtime behavior of the process. If the process contains logic to initialize its partner links then it should set this property to **No**. This type of initialization would occur through the use of an Assign activity.*

*If the process requires its deployment environment to initialize its PartnerLinks, either through a static endpoint at deployment or perhaps through an endpoint reference scheme like WS-Addressing, then it should set this property to **Yes**. If the property is missing, then there is no information conveyed to the deployer as to how the partner links are initialized.*

## invalidVariable Fault

- This fault is thrown when
  - “validate” attribute set to “yes” and assignment fails due to validation error

`bpel:invalidVariable`



27 Copyright © 2004-2007 Active Endpoints, Inc.



We can get the “invalidVariable” fault if we set the “validate” attribute to “yes” and one of the variables fails the validation.

FROM USER GUIDE PAGE 523:

*Thrown when an XML Schema validation (implicit or explicit) of a variable value fails...*

Q. If we set the “Validate” attribute to “No”, when is this error thrown? Maybe never, it depends on what you’re doing.

## xsltStylesheetNotFound Fault

- This fault is thrown when
  - The first parameter of the `bpel:doXsltTransform()` function, consisting of a URI to the location of the XSL stylesheet, cannot be found

`bpel:xsltStylesheetNotFound`



This fault results when you use the `doXsltTransform` function and it references a style sheet that can't be found.

## xsltInvalidSource Fault

- This fault is thrown when
  - The second parameter of the `bpel:doXsltTransform()` function, consisting of an XPath nodeset, does not resolve to a single element node

`bpel:xsltInvalidSource`



29 Copyright © 2004-2007 Active Endpoints, Inc.



If the second parameter of the `doXsltTransform` function, which is the document (i.e., the “node-set”) resolves to anything other than a single element, you get this fault.

FROM the ActiveBPEL Designer's USER GUIDE PAGE 524:

*xsltInvalidSource* fault is thrown when the transformation source provided in a `bpel:doXsltTransform` function call was not legal.

## subLanguageExecution Fault

- This fault is thrown when
  - Any fault occurs during XSL transformation, query execution or expression evaluation

`bpel:subLanguageExecution`



30 Copyright © 2004-2007 Active Endpoints, Inc.



This is what you get when the system throws any transform fault, or faults during execution of a query, or during the evaluation of an expression that was invalid. This won't be much help because BPEL can only verify itself (i.e., the BPEL language), and can't drill into the individual issues of expression and query languages.

FROM the ActiveBPEL Designer's USER GUIDE PAGE 524:

*subLanguageExecutionFault is thrown when the execution of an expression results in an unhandled fault in an expression language or query language.*

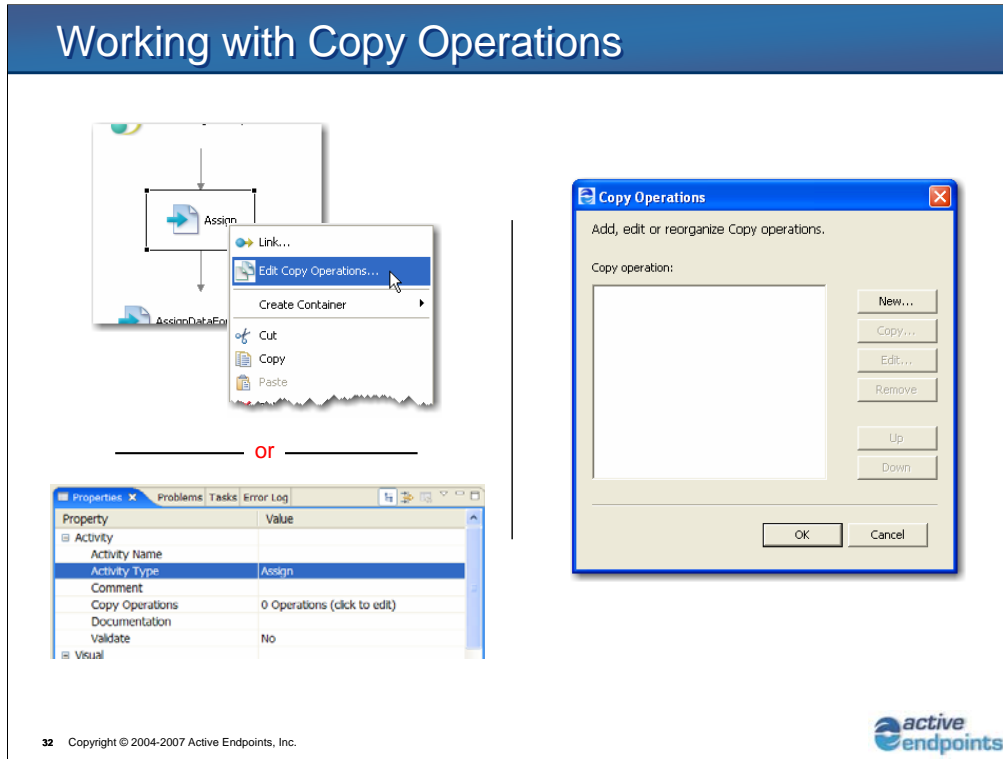
## Unit Objectives

- At the conclusion of this unit, you will be familiar with:
  - ✓ `assign` activity
  - Working with Copy Operations in an `assign`

31 Copyright © 2004-2007 Active Endpoints, Inc.



Now that we've looked at the Assign Activity and its Copy Operations, let's take a look at how we work with Assigns and Copies in the tool.



Assign is considered to be a Basic activity in BPEL, but you can also consider it a container activity because it has one or more Copy operations inside it. We can Right-Mouse on an Assign activity to see the Copy operations inside it, or we can look in the Properties view under Copy operations. Note that you will have to click in the Copy Operation's value field to get the ellipsis icon, and then you can click on that to get to the Copy Operations dialog. Once you are in the Copy Operations dialog, the buttons for copy, edit, remove, etc. will become enabled. We can also drill down to the specific Assign activity in the Outline View and see its Copy operations listed there. We can also copy existing Copy Operations from one Assign activity and use them in another Assign activity. We can also change the *order* of the Copy Operations in an Assign by using the Up and Down buttons in the Copy Operations dialog. Execution order can be important because they will execute in the order displayed.

## New Copy Operation – Variable

- Specify the Variable and optionally the Part and/or Query

The screenshot shows a dialog box titled "Copy Operation" with the instruction "Define the From and To parts of the Copy Operation." It features two main sections: "From" and "To". Each section includes a "Type" dropdown menu (currently set to "Variable"), a "Variable" text field, a "Part" dropdown menu, and a "Query" text field with an ellipsis button. At the bottom of the dialog, there are two checkboxes: "Keep source element name" and "Ignore missing 'From' data". "OK" and "Cancel" buttons are located at the bottom right.

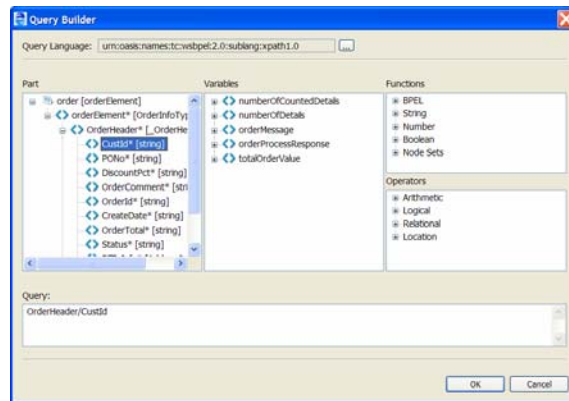
33 Copyright © 2004-2007 Active Endpoints, Inc.



This is the Copy Operations dialog. From a high level, in this dialog we are specifying the From and the To, i.e., the target and the destination for the Copy. In both From and To we need to first select the type – variable, literal, expression, etc., - choosing the From and To specs we saw earlier. Subsequent Drop lists are then populated based on your selections in the earlier droplists. To create a query, you Click on the elipsis, which opens query builder and allows you to create the queries based on selected language. And also note the two checkboxes, which are available for each copy operation.

## Query Builder

- Assists with creating queries used to extract particular values from a variable of message type within a copy operation
  - Uses XPath 1.0 as the query language



34 Copyright © 2004-2007 Active Endpoints, Inc.



If you click on the ellipsis you get to the Query Builder dialog. The query builder's panes are populated based on your process's definitions. Dbl-click on a part, variable, function or operator to insert it into the query. We recommend that you use the Query Builder because there are no typos when using this Wizard. Having said that, be careful when you do this because if you dble-click on the Document, it will insert a "." into the query and you might not notice that you did it.

## New Copy Operation – Expression

- Specify the Expression to use

Copy Operation

Define the From and To parts of the Copy Operation.

From

Type: Expression

Expression: ...

To

Type: Variable Property

Variable:

Property:

Keep source element name

Ignore missing "From" data

OK Cancel

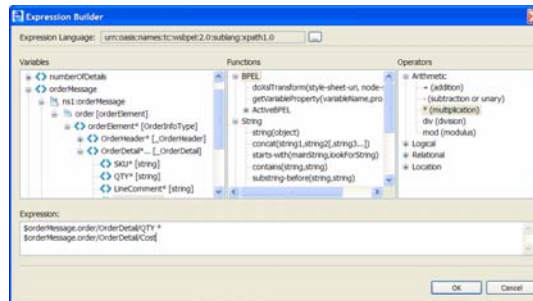
35 Copyright © 2004-2007 Active Endpoints, Inc.



To Copy from an expression Click on the ellipsis icon ( “...” ) in the Copy Operation dialog to open the expression builder.

## Expression Builder

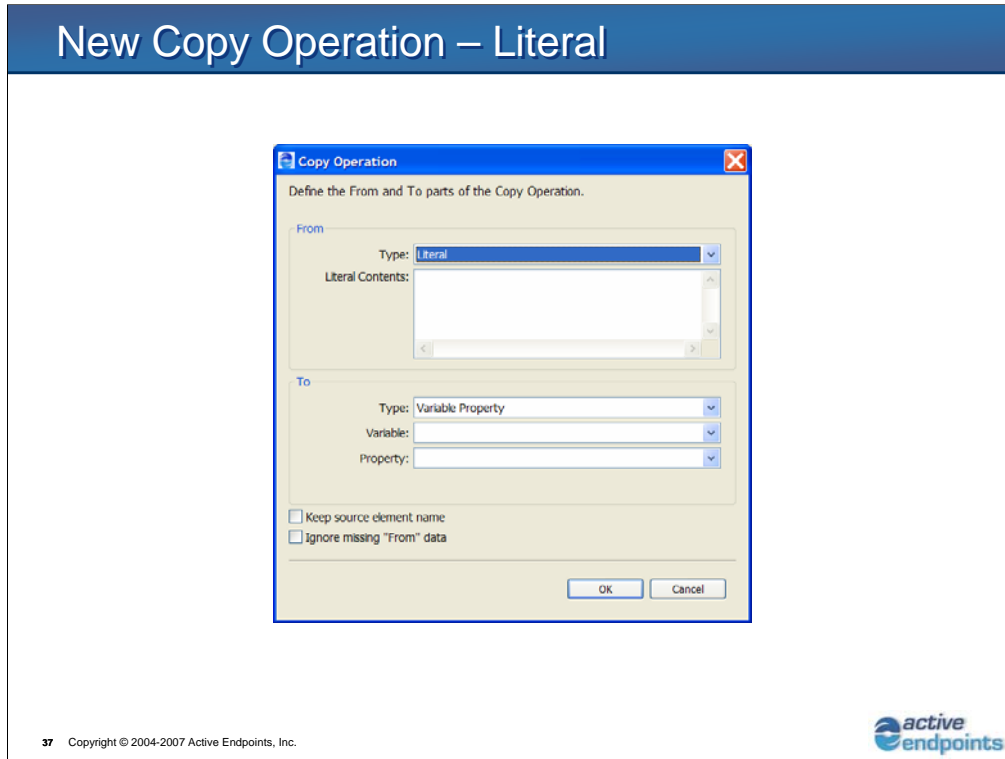
- Assists with creating expressions for use with certain BPEL constructs
  - Copy operations, Link transitions, If conditions
  - Default expression language is XPath 1.0
    - Extension support for XQuery 1.0 and JavaScript 1.5
- Select the variables, functions and operators to form a valid expression



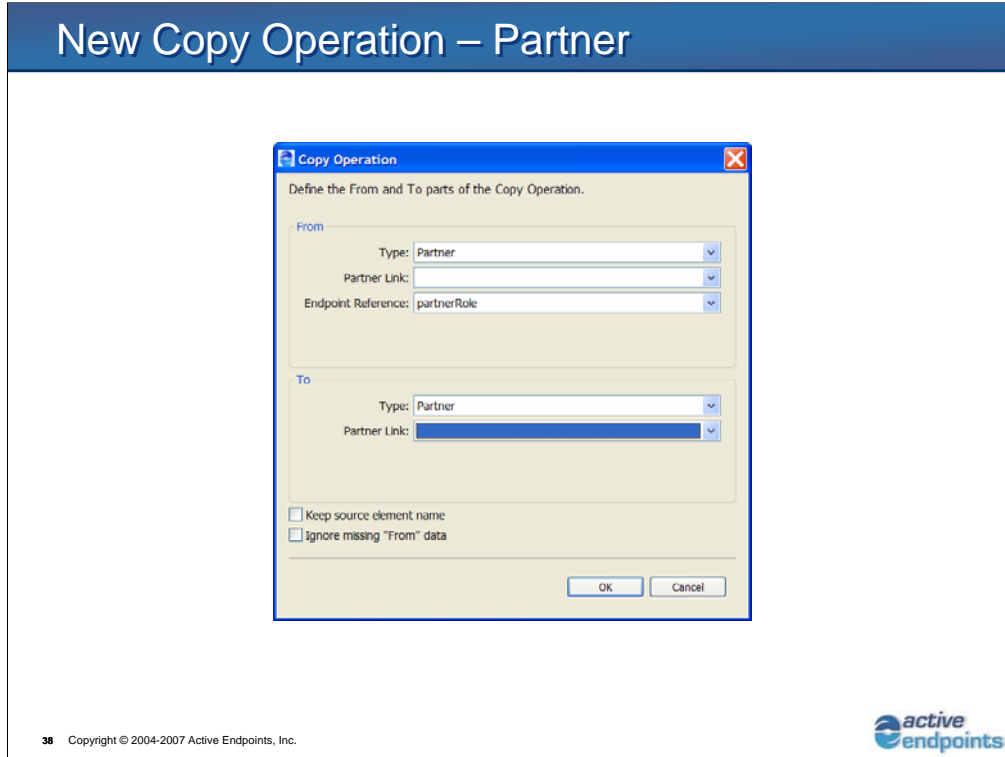
36 Copyright © 2004-2007 Active Endpoints, Inc.



The Expression Builder is used to set the expressions for Copy, Link and If statement conditionals. Note the Expression Language field at the top of the dialog, which allows you to designate which language the expression will use. In ActiveBPEL Designer, XPath1.0 language is default, but can use others like XQuery 1.0 and JS1.5. You can Dble-click on variables, functions and operations to add them to the expression.



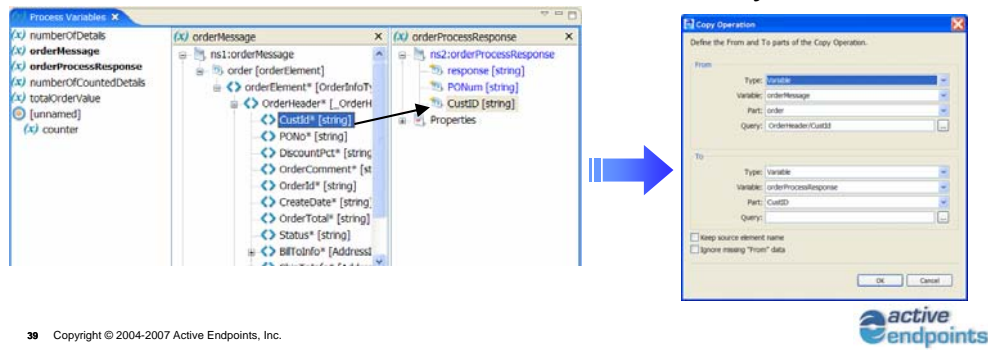
To Copy from a literal, to whatever target, simply paste the literal's text into the "Literal contents" field and then select the target information from the drop-lists below.



To copy from a Partner to a Partner, specify the Partner as the type for the Copy Operation. The PartnerLink Droplist will show the available partner links. Once you select a partnerLink, then the Endpoint reference field will be populated. The Endpoint Reference droplist lets you set the endpoint references available for the source (but not for target, obviously.) This is to copy PartnerLink info at runtime. For example, if a customer is given a choice of three providers, and then chooses one, you need to set the info dynamically, when the choice is actually made, before you can invoke the specific service desired. You might also be copying info from one partnerLink to another, including info such as the endpoint reference (i.e., the URL) of the service. It can also have “WS:addressing” information, such as policy info, password and login info, etc.

## Creating Copy Operation via Drag and Drop

- In the Process Variables view you can create **copy** operations by dragging and dropping
  - From Variable [Part] To Variable [Part]
  - From Variable Property To Variable Property
- If an **assign** was selected a new **copy** operation is added otherwise a new **assign** activity is added



In the Designer, you can use the Process Variables view to perform Drag & Drop copy operations. Simply double-click on the “From” and “To” variables to open them up, then drill into the variable definitions themselves. Now, Drag one process variable part to another process variable's part, staying inside the Process Variables View. As you do, watch the status bar of the application (in the Designer's lower left hand corner) and it will show either the Copy Operation that it will add, or one of the applicable error messages that will tell you why it can't create the Copy for you. Note that ActiveBPEL Designer won't let you assign incompatible types as the From and To when building a Copy operation, and will display feedback in the form of a red error message in the status bar, as necessary. Once you release the mouse button, the Copy operation goes into the activity that is selected in the process diagram, which must be an Assign activity. If there is no Assign activity selected or if a non-Assign activity is selected, *BPEL will automatically add a new Assign activity, placed in the upper left hand corner of the diagram.*

One helpful hint is that if you hover over the various artifacts in the Process Variables View, ActiveBPEL Designer will echo back in a tool-tip what the item is, such as a Message or a Part.

MESSAGE-Variables are shown as a piece of paper type of icon

Properties are shown as a piece of paper with a red check mark

A Message Part is shown with an <> icon

A Message's element is shown as a <>

## Suppressing selectionFailure Faults

- ActiveBPEL provides an extension mechanism to suppress `bpel:selectionFailure` faults during copy operations
  - Disable Selection Failure
    - Assignment's copy operation
    - Allows a given query to return an empty node set
    - Useful when source XML Schema declares optional elements which are not present in the instance
  - Create XPath
    - Useful when the target
      - XML instance does not contain the element specified in the location path
        - e.g., the variable is not initialized
      - XML Schema declares repeating elements where the number of occurrences vary
    - Location path is first constructed and then the value is copied

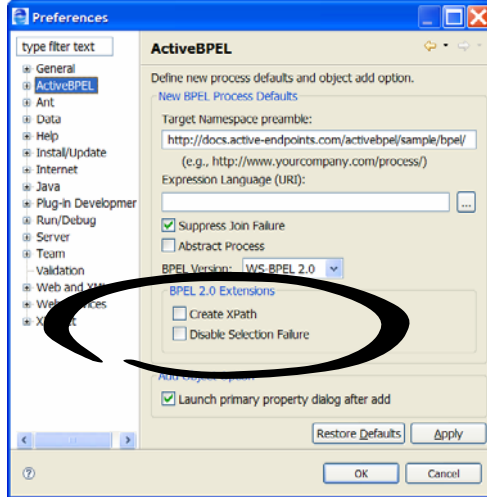
40 Copyright © 2004-2007 Active Endpoints, Inc.



We'll use both of these settings during the course of our labs. Recall that we saw these earlier in the Copy Operations dialog. **Suppress selection failure faults** means that if a BPEL fault is thrown when a Copy operation in an Assign activity generates an error we can override the fault using the extension mechanism "Disable Selection Failure" if the From element (i.e., the source) that generated the failure is an optional element (which you can tell by the ? icon). **Create XPath** is useful when you have a good 'Source' but a bad 'Target' because the TO variable does not have the element specified in the copy, or because there are Repeating elements where the number of occurrences varies. Also, if the target lacks the element targeted by the copy, it will add the element, i.e., the location path is constructed first and then the values are copied to it. We can also initialize complex variables using this technique. Both of these Preference choices can be set individually for each Copy operation and can also be set at the process level.

## Suppressing selectionFailure Faults Preferences

- By default these two preferences are disabled



Note: Lab solution assumes that the Create XPath option is enabled

41 Copyright © 2004-2007 Active Endpoints, Inc.



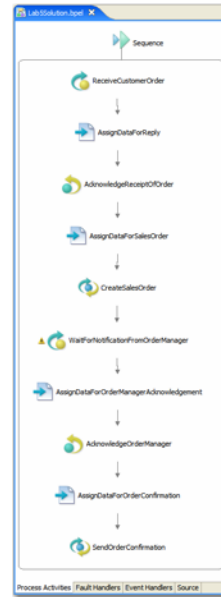
Go to the Windows->Preferences menu item. Note the “Create Xpath” and “Disable Selection Failure” checkboxes. By default both are *disabled*. Any process we create will have these settings set to the default, which is disabled. Note that the solution for our Process assumes that “create xpath” selection is *enabled*, meaning that we let the engine check this for us.

## Lab 5 – Assignment Activity

### ■ Overview of Lab Exercises

–Add the required assign activities

- Expression copy using the Expression Builder
- Variable copy using the Query Builder
- Variable copy using Drag and Drop in the Process Variables view



42 Copyright © 2004-2007 Active Endpoints, Inc.



The next Lab in the BPEL Fundamentals class is Lab #5. At this point, we have a Workspace, a Project, a Process and several interaction activities on our canvas – receives, invokes and replies - but there is no data yet. In this Lab we'll add Assign activities and Copy Operations in each of the three available ways:

Using the Expression Builder

Using the Expression Builder

Using the Drag & Drop functionality in the Process Variables View

## Unit Summary

- Now you are familiar with:
  - assign activity
  - Working with Copy Operations in an assign