This is Unit #4 of the BPEL Fundamentals I course. In past Units we've looked at ActiveBPEL Designer, Workspaces and Projects... so now let’s take a look at the actual BPEL Processes...
Unit Objectives

- At the conclusion of this unit, you will be familiar with:
  - BPEL process definitions
  - Standard BPEL Faults
  - Working with BPEL processes
  - Considerations before getting started

In this Unit we'll look at how BPEL Processes are defined and the standard faults supported by the system. Then we'll examine BPEL Processes and go over some things that should be considered before we get started.
Abstract and Executable BPEL Processes

Abstract Processes
- Models the public behavior imposed on two interacting partner services

Executable Processes
- Models the private behavior of a partner service in a business interaction

On the left we see two partners, each with a role as a buyer or seller, and the traffic pattern of their interaction. This is an Abstract Process and it defines the process' public-facing behavior and is the conceptual definition of what will end up as the completed process. Think of it as the process from the view of a third party. This is not used much, mostly in the preliminary review of a design. The Executable Process on the right shows the private definition of the process and defines the actual interactions of the completed process. Think of it as the process when viewed from behind the developer's curtain. So, on the right we see the actual implementation of the process that will be deployed. From now on, we are only concerned with executable processes.
A BPEL Process is one that defines a business process, is displayed in XML format, that conforms to the BPEL 2.0 schema, and is deployed as a web service. The Process file contains the definitions listed on the slide, including:

- Process Variables
- Handlers (Event, Fault, Termination, Compensation)
- Partner Links (i.e., instances of the WSDL-defined PartnerLinkTypes)
- A single primary activity (Start Activity) that will instantiate the process.
Here we see the structure of a process. At the top we name of the process and then follow with the process' Global Declarations and the Process Definition. The Global Declarations section includes the extensions, imports, partnerLinks, etc. that are used in the Process. And finally, the Process Definition has the actual activities that make up the process' functionality.
We assemble the Process graphically in ActiveBPEL Designer, but this is the actual BPEL code that’s created in the background. It follows the file structure we saw on the previous slide:

It names the process and designates the target namespace.

It lists the Global declarations, which define items such as PartnerLinks and Process Variables.

It holds the Process Definition, which is the process' individual activities and their containers that make up the process’ actions.

In this example the Process boils down to: An initial Receive, an Assign with a single Copy operation, and a Reply.
BPEL Activities

- Activities represent the actions to be carried out within a business process
- BPEL defines two types of activities
  - Basic Activities
    - invoke
    - receive
    - reply
    - assign
    - compensate
    - compensateScope
    - empty
    - exit
    - throw
    - rethrow
    - validate
    - wait
  - Structured Activities
    - flow
    - foreach
    - if
    - pick
    - repeatUntil
    - scope
    - sequence
    - while

BPEL Activities are the actions the process actually performs. Basic activities are those that are considered to be atomic, meaning that they cannot be subdivided. Structured activities are those activities which can contain other activities.
There are two major ways to define a business process – using a program structure and using a graph structure. A Program structure is defined by its content, is controlled by loops, logic, branches, sequences and executes in a linear fashion. A Graph structure is in a format defined by a network of links and it can execute its activities in parallel. These two approaches can be mixed and matched as needed. Why have both? Recall that BPEL originated as the merging of two different specifications… the program structure is from Microsoft and the Graph structure is from IBM.

Note that BPEL's Flow activity allows for the parallel processing of Process activities (the only such activity, other than the ForEach.)
A BPEL process definition contains a single primary activity
   - The main entry point for the process definition

Primary activity can be
   - Simple basic activity
   - Complex structured activity
     - With many nested activities contained to arbitrary depth

Meaningful BPEL processes use either the sequence or flow activity as the primary activity

Recall that a BPEL process must receive a message in order to instantiate the process. All BPEL processes must have a single primary activity, which receives the message and is the main entry point for the process. This primary activity can be a simple activity or a structured activity. If it is a structured activity it can be nested to any depth. Most BPEL processes will have a Sequence or a Flow as their primary activity.
Common Activity Syntax

- **Common Attributes (Optional)**
  
  ```
  name="ncname"?
  suppressJoinFailure="yes|no"?
  ```

- **Common Elements (Optional)**
  
  ```
  <targets>?
    <joinCondition expressionLanguage="anyURI">?
      bool-expr
    </joinCondition>
    <target linkName="NCName" />+
  </targets>

  <sources>?
    <source linkName="NCName">+</n
    <transitionCondition expressionLanguage="anyURI">?
      bool-expr
    </transitionCondition>
  </source>
  </sources>
  ```

Every activity also can have a set of common attributes and elements. Many of these elements will only apply to certain activities. For example, Flow activities will have links, joinConditions and source/target designations.
Now let’s take a look at the Life Cycle of a BPEL process.

1. When an incoming message is received, the Server will try and match it to a currently running instance.

2. If it cannot do so, it will create a new Process instance, and the message will be received by this new instance.

Remember that every executable process MUST have at least one “start” activity.
Termination of a process instance occurs
- **Normally**, when primary activity of process completes
- **Abnormally**, when
  - Unhandled fault reaches the primary activity
  - Explicitly terminated using an **exit** activity

Now that we know how a process is instantiated, let's look at how a process terminates. There are two ways a running process can terminate – normally or abnormally.

*Normal termination* is when the process activities are completed successfully.

*Abnormal termination* is when the process activities are not completed successfully:
Usually because of a fault being thrown that reaches the Process level without being handled.

Also through the use of an “Exit”, which stops the running process instantly.

Note that the Exit activity is new to the BPEL 2.0 specification and was previously known as the Termination activity in BPEL v1.1
Unit Objectives

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

- BPEL process definitions
  - Standard BPEL Faults
  - Working with BPEL processes
  - Considerations before getting started
Standard BPEL Faults

- BPEL defines a set of built-in system-centric faults
  - BPEL runtime is responsible for signaling these faults
  - Defined in the BPEL namespace (bpel:)
    - http://docs.oasis-open.org/wsbpel/2.0/process/executable

The ActiveBPEL Engine generates and manages system faults. It signals the process that a fault has been encountered. These are the faults that are defined in the BPEL executable namespace shown here.
### Standard BPEL Faults

<table>
<thead>
<tr>
<th>Fault Name</th>
<th>Fault is thrown when:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ambiguousReceive</td>
<td>more than one IMA for the same partnerLink, portType, operation but different correlationSets, matches an incoming request message</td>
</tr>
<tr>
<td>completionConditionFailure</td>
<td>if it is determined that the completion condition of a forEach activity can never be true.</td>
</tr>
<tr>
<td>conflictingReceive</td>
<td>more than one receive activity or equivalent are enabled simultaneously for the same partner link, port type, operation and correlation set(s)</td>
</tr>
<tr>
<td>conflictingRequest</td>
<td>more than one synchronous inbound request on the same partner link for a particular port type, operation and correlation set(s) are active</td>
</tr>
</tbody>
</table>

On the following four slides are the Standard BPEL faults. These are faults that are generated by the system.
<table>
<thead>
<tr>
<th>Fault Name</th>
<th>Fault is thrown when:</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlationViolation</td>
<td>the contents of the messages that are processed in an invoke, receive, or reply activity do not match specified correlation information</td>
</tr>
<tr>
<td>invalidBranchCondition</td>
<td>the integer value used in the &lt;branches&gt; completion condition of &lt;forEach&gt; is larger than the number of directly enclosed &lt;scope&gt; activities</td>
</tr>
<tr>
<td>invalidExpressionValue</td>
<td>an invalid value is returned from the execution of an expression</td>
</tr>
<tr>
<td>invalidVariables</td>
<td>an XML Schema validation (implicit or explicit) of a variable value fails</td>
</tr>
<tr>
<td>joinFailure</td>
<td>join condition of an activity evaluates to false</td>
</tr>
<tr>
<td>mismatchedAssignmentFailure</td>
<td>incompatible types are encountered in an assign activity</td>
</tr>
</tbody>
</table>

.. And some more faults.
### Standard BPEL Faults (cont.)

<table>
<thead>
<tr>
<th>Fault Name</th>
<th>Fault is thrown when:</th>
</tr>
</thead>
<tbody>
<tr>
<td>missingReply</td>
<td>a scope (or process) completes without replying to a receive</td>
</tr>
<tr>
<td>missingRequest</td>
<td>a reply activity is executed with no open receive</td>
</tr>
<tr>
<td>scopeInitializationFailure</td>
<td>there is any problem creating any of the objects (e.g., variables or partnerLinks) defined as part of scope initialization</td>
</tr>
<tr>
<td>selectionFailure</td>
<td>a selection operation performed either in a function or in an assignment, encounters an error</td>
</tr>
<tr>
<td>subLanguageExecutionFault</td>
<td>an unhandled exception is encountered during the execution of an expression or query</td>
</tr>
<tr>
<td>uninitializedPartnerRole</td>
<td>an invoke or assign activity references a partner link whose partnerRole endpoint reference is not initialized</td>
</tr>
</tbody>
</table>

… and some more!
<table>
<thead>
<tr>
<th>Fault Name</th>
<th>Fault is thrown when:</th>
</tr>
</thead>
<tbody>
<tr>
<td>uninitializedVariable</td>
<td>there is an attempt to access the value of an uninitialized part in a message variable</td>
</tr>
<tr>
<td>unsupportedReference</td>
<td>unable to interpret the combination of the reference-scheme attribute and the content element OR just the content element alone</td>
</tr>
<tr>
<td>xsltInvalidSource</td>
<td>if an invalid XML element is provided as the source document to the <code>bpe1:doXslTransform</code> function call</td>
</tr>
<tr>
<td>xsltStylesheetNotFound</td>
<td>when the named style sheet in a <code>bpe1:doXslTransform</code> function call was not found</td>
</tr>
</tbody>
</table>

… and just a few more. That’s all of the Standard BPEL Faults… but of course, there are also the faults that are defined in the process, too. There are an unlimited number of these User-defined faults.
Unit Objectives

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

- BPEL process definitions
- Standard BPEL Faults
  - Working with BPEL processes
  - Considerations before getting started

Now that we've looked at BPEL process definitions and the standard system faults, let's see how to work with BPEL processes.
Create a BPEL Process

1. Select File>New>BPEL Process
   - New BPEL Process Wizard opens
2. Specify which parent folder to add the process to
3. Specify a name for the new process
4. Select Finish
   - Process editor opens

Alternatively either:
- Right click on a project folder, then New>BPEL Process
- Select New toolbar dropdown, then BPEL Process

In ActiveBPEL Designer, we have already created a new project, so now we will create the actual process. Go to File->New->BPEL Process and when the Wizard opens, specify what folder to store the new file in, what you want to name it, and then click finish.
When you have the New BPEL Process dialog open, you can click on Advanced to change the preamble for the target namespace. When we change the target namespace preamble the new text will be amended to the namespaces created by ActiveBPEL Designer. If we do nothing, it will use what’s in the preferences by default.
ActiveBPEL Designer Preferences

- Default preferences used for creating new processes
  - Can override preferences on a per process basis

While we are here, we can change the system’s default preferences, as needed. Note that these settings are at the Workspace level, but we can override these defaults on a per-process basis. The namespace preamble is pre-pended to the namespace for processes that you create.
Importing an Existing BPEL Process

- You can import existing BPEL process into your project
  - e.g., you have created a BPEL process outside of ActiveBPEL Designer

- Steps
  - In the Navigator view, right-click the parent folder and then select Import...
  - Select File System and then select Next
  - Browse to the appropriate location on your file system and then select the resources to import

Note: Be sure to add any additional WSDL and Schema files referenced in your BPEL process to Web References

You can import individual resource files or entire processes (and then the process’ supporting files) into your project. Open up the Navigator View, use the Right Mouse menu on the parent folder, select “Import”, select File System, click Next and then browse to the location. Note that if you import a process you must also import the WSDL and Schema files referenced by that process and bring them into the Web References View.
Here is the ActiveBPEL Designer’s Process Editor.

1.) The Process Editor’s **Canvas** is the default Editor for the tool.

2.) The default that selects the Process Editor's canvas is the Process Activities tab from the **Section** tabs.

3.) The **Palette** has all of the process activities, as outlined in the BPEL 2.0 specification, and its presentation can be customized.
For every `.bpel` file there is a corresponding `.vbpel` file to accompany it. The `.BPEL` file contains the process description in XML format and contains everything needed to deploy and execute the process. The `.vbpel` (pronounced VEE – BEEP-UHL) is an extra file that represents the visual layout of the process' diagram. This file is in XGMML format and contains everything needed to display the process in a graphical format. This file is not needed for deployment because a deployed process does not need to be displayed anywhere.
Showing .vbpel Files in Navigator

- By default, .vbpel files are excluded from display in the Navigator view
  - To display .vbpel files in the Navigator view
    - select Filters... from the Navigator view menu, then uncheck the ".vbpel" entry

- You may need to use .vbpel files for the one of the following reasons
  - When copying BPEL files
    - .vbpel files do not get copied
  - When using Replace with Local History
    - Need to separately replace the .vbpel file
  - When editing the BPEL file outside ActiveBPEL Designer
    - .vbpel file will need to be re-generated

When you copy the project files, the .vbpel is not copied automatically. You have to do that yourself. Even after you copy it, it is filtered out of the Navigator View by default. Use Navigator->Filters and select the "* .vbpel" type to see them in the heirarchy.
Clicking on the process editor and opening the Right Mouse menu gives you this set of choices. It supports all of the common functionality of the menus and other Right Mouse actions. As well as display choices, such as Collapsing or Expanding containers and whether to show Swimlanes.
Toolbar options provide much of the same functionality…
Working with Activities

- To add an activity to the process
  - First choose the appropriate activity from the palette and then click within the process editor canvas
  - Alternatively, drag and drop an activity from the palette to the desired location in the process editor canvas

- You can also
  - Copy and paste activities
  - Select multiple activities and update their properties as a group
  - Align multiple activities with one of the align options from the process toolbar

Select the activity in the palette which “loads” the activity, and then click on the process where you’d like the activity to be inserted. Alternatively, you can use Drag & Drop to insert an activity from the palette onto the canvas. You can also use Copy and Paste with activities that are already in the diagram, and you can select multiple activities and then edit their properties as a group. Finally, you can Align activities in the graphic as needed, using “AutoLayout” or “Align Left”, for example.
Activity Properties

- With the activity in focus on the process editor canvas, you can set property values in the Properties view.

- Select the Advanced button to display all the properties for an activity.

Select an Activity on the drawing canvas and then select the Properties View. Note that you may have to use the Advanced Properties button to see all of the activity’s properties.
Static Analysis

- When you save your process, static analysis is performed on the BPEL file
- Validates the BPEL definition, XPath syntax, and WSDL references
  - Adds items for any issues found to the Problems view

Static Analysis is a syntax check and a requirements check, which verifies that the BPEL process is consonant with the definitions required by the BPEL Specification. This check is done whenever you save your files. The check's results show up on the Problems tab and you can filter these items for errors/warnings/info as needed. For more details on the definition of Static Analysis you can go to Appendix B of the BPEL specification, which covers all 95 checkpoints for a BPEL process.
Unit Objectives

- At the conclusion of this unit, you will be familiar with:
  - BPEL process definitions
  - Standard BPEL Faults
  - Working with BPEL processes
  - Considerations before getting started

Now, let’s take a look at some things to consider before we start working.
There are two different approaches to designing a BPEL process - the top-down approach and the bottom-up approach.

If using the top down approach:
- sketch out your high level intent with BPEL constructs
- derive your implementation requirements – operations and messages
- fill in the details of the information flow
- build your process implementation constructs

If using the bottom up approach:
- create your activities and process definition
- Add message definitions
- tie them together to create your information flows and process steps based on your business rules
### Process Design Planning

- **Identify**
  - Business process to automate
    - e.g., Purchase Order process, Claims process, Service Provisioning process
  - Discrete business functions that are described as Web services
    - e.g., Inventory, credit rating, shipping and logistics
  - Information flow between the process steps
    - e.g., Receive PO request as input to process, pass the list of order items to the check inventory levels, and so on
  - Business rules required to orchestrate the process
    - e.g., After checking inventory levels, depending on the result, either place the order or send an out-of-stock notification

Before you begin creating your process, though, you need to step back and plan the process design.

1.) What is the Process and what is its purpose? What does it actually do?
2.) What Business Functions (i.e., Web Services) will the process will use? Do they all exist now or do they have to be created also?
3.) What is the Information flow (i.e., the variables used) both inside and between the process steps? //this can also involve mapping data between partner invocations.
4.) What are the Business rules (logic/flow/execution routes) that must be applied to the process? How do they affect the way you will orchestrate the process?
Unit Summary

- Now you are familiar with:
  - BPEL process definitions
  - Standard BPEL Faults
  - Working with BPEL processes
  - Considerations before getting started