



This is Unit #2 of the BPEL Fundamentals course. In this Unit we will look at Partner Links and at the Partner Interactions that occur during a BPEL process.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - Partner Interactions
 - Using the `partnerLinkType` construct
 - Where to define `partnerLinkTypes`

2 Copyright © 2004-2008 Active Endpoints, Inc.



The topics in this unit are Partner Interactions, the PartnerLinkType construct and where PartnerLinkTypes are defined. (You should already know about portTypes from your previous experience with WSDL.) In a BPEL process we will extend the WSDL specification by introducing the concept of a partnerLinkType. A partnerLinkType must include definitions for exactly one portType or exactly two portTypes. The partnerLinkType is in the WSDL file and it establishes a structured relationship between the process and the service that will facilitate communication.

Note: portTypes are from the WSDL 1.1 standard and the partnerLinkType is from the BPEL 2.0 standard.

Partner Interaction

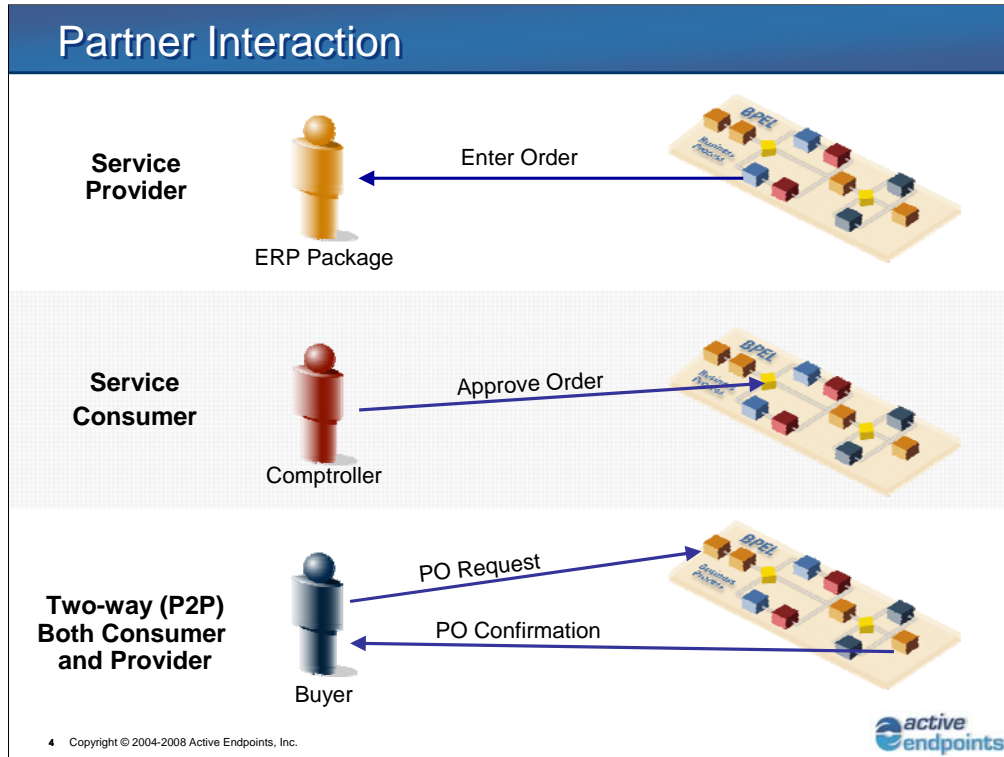
- At the highest level, a BPEL process defines the interactions among one or more partner services
- Partners may
 - Provide Web services to the process
 - Consume Web services from the process
 - Interact in a two-way asynchronous conversation with the process
 - Both provide Web services to the process and consume Web services from the process

3 Copyright © 2004-2008 Active Endpoints, Inc.



Looked at from a high level, a BPEL process defines how one or more web services interact with our process.

A partner may provide a web service to the process, it may consume a web service from the process, or it may do both.



In this diagram our process partners are shown on the left, and the BPEL process is shown on the right.

1. The ERP Package is a Service Provider to the Process (note the direction of the arrow) so the process is reaching out to a partner site get a web service to use.
2. The Comptroller is a Consumer of the Process's Services (and, again, note the direction of the arrow) so the partner is reaching out to the process to get a web service.
3. The Buyer is both a Consumer of - and a Provider to - the process's web services.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Partner Interactions
 - Using the `partnerLinkType` construct
 - Where to define `partnerLinkTypes`

5 Copyright © 2004-2008 Active Endpoints, Inc.



Now that we understand partner interactions, let's take a look at BPEL's `partnerLinkType` construct.

partnerLinkType WSDL Extension

- BPEL defines the **partnerLinkType** construct to model service interactions
 - Extensibility mechanism of WSDL 1.1
 - Defined in the namespace
<http://docs.oasis-open.org/wsbpel/2.0/plnktype>
- Represents a third party viewpoint of the conversational relationship between two partner services
 - Describes the WSDL `portTypes` required to be implemented for the relationship to succeed

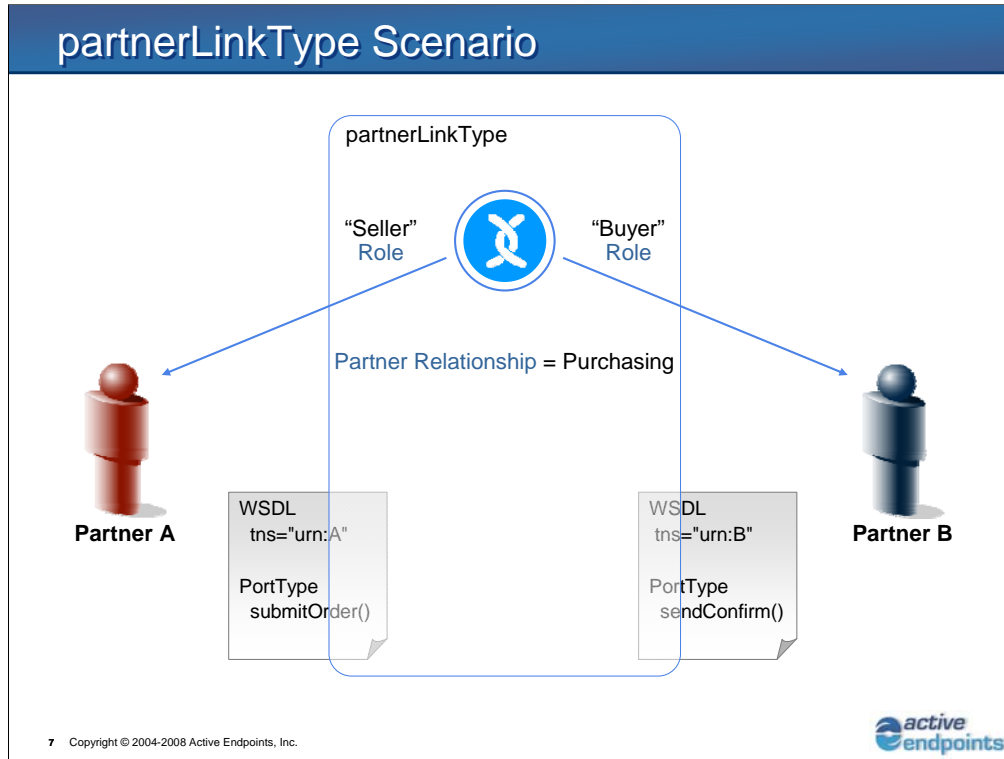
6 Copyright © 2004-2008 Active Endpoints, Inc.



WSDL-defined extensions called *partnerLinkTypes* can be used to represent dependencies between web services, whether or not those services are used by a BPEL process. This is the reason that *partnerLinkTypes* are defined in a WSDL file and not in the BPEL process itself.

PartnerLinks, which are simply instances of a *PartnerLinkType*, describe the conversational relationship between the partners, as seen from a third-party perspective. They describe how the *portTypes* involved must be implemented in order for the relationship to succeed.

One thing that should be noted is that the *PartnerLinkType* is a new construct. As it says in the specification itself... “It is important to emphasize that the notions of partner link and endpoint reference used here are preliminary. The specification for these concepts as they relate to Web services is still evolving, and we expect normative definitions for them to emerge in future.”



Here we have two partners, where one is the partner service and one is the BPEL process.

The two of them have a purchasing relationship which is described by the PartnerLinkType.

Here Partner A is in the "Seller" role.

Partner B is in the role of "Buyer."

Partner A and its PortType contains the operation "submitOrder," which defines its Service.

Partner B and its PortType contains the operation "sendConfirm," which defines its Service.

The relationship between these two partners is what is defined by the partnerLinkType.

partnerLinkType Code Sample

WSDL Fragment

```
<definitions targetnamespace="urn:C"
  xmlns:sell="urn:Sell"  xmlns:buy="urn:Buy"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:plnk="http://docs.oasis-open.org/
    wsbpel/2.0/plnktype">

  <plnk:partnerLinkType name="PurchasingPLT">
    <plnk:role name="Seller" portType="sell:SellerPT" />
    <plnk:role name="Buyer" portType="buy:BuyerPT" />
  </plnk:partnerLinkType>

</definitions>
```

8 Copyright © 2004-2008 Active Endpoints, Inc.



Here is a code sample for a partnerLinkType. (We should note that this kind of WSDL fragment could very possibly be needed during one of the labs... you never know!) Note first the nameSpace declaration, which uses the convention "plnk" as the prefix. Then we declare the partnerLinkType and name it "PurchasingPLT." The PurchasingPLT partnerLinkType has two roles defined, "Seller" and "Buyer," each of which is associated with exactly one portType, "SellerPT" and "BuyerPT," respectively.

partnerLinkType Semantics

- Each **partnerLinkType** must have at least one role and at most two roles
 - Two role case implies asynchronous conversation
 - Single role case when there is no need to place any requirements on the other partner service
- Each role specifies exactly one WSDL **portType**

© Copyright © 2004-2008 Active Endpoints, Inc.



Now let's take a look at the semantics of PartnerLinkTypes. Each partnerLinkType has exactly one or exactly two roles, depending upon what kind of conversation it supports. If it has two roles, it is defining an asynchronous conversation. If it has one role it is defining a synchronous conversation. Recall that each role specifies exactly one portType.

Unit Objectives

- At the conclusion of this unit, you will be familiar with:
 - ✓ Partner Interactions
 - ✓ Using the `partnerLinkType` construct
 - Where to define `partnerLinkTypes`

Now that we know what `partnerLinkTypes` are and how they are used, where do we define them?

Where To Define Partner Link Types

- Placed into a separate WSDL document
 - Used when the referenced `portTypes` are defined in two different services
 - Must use the WSDL `import` element to import the two different WSDL documents
 - With its own `targetNamespace`
- Placed directly into the originating WSDL document
 - Used when the referenced `portTypes` exist in the same `targetNamespace`
- Can be performed manually or automated through the use of a tool

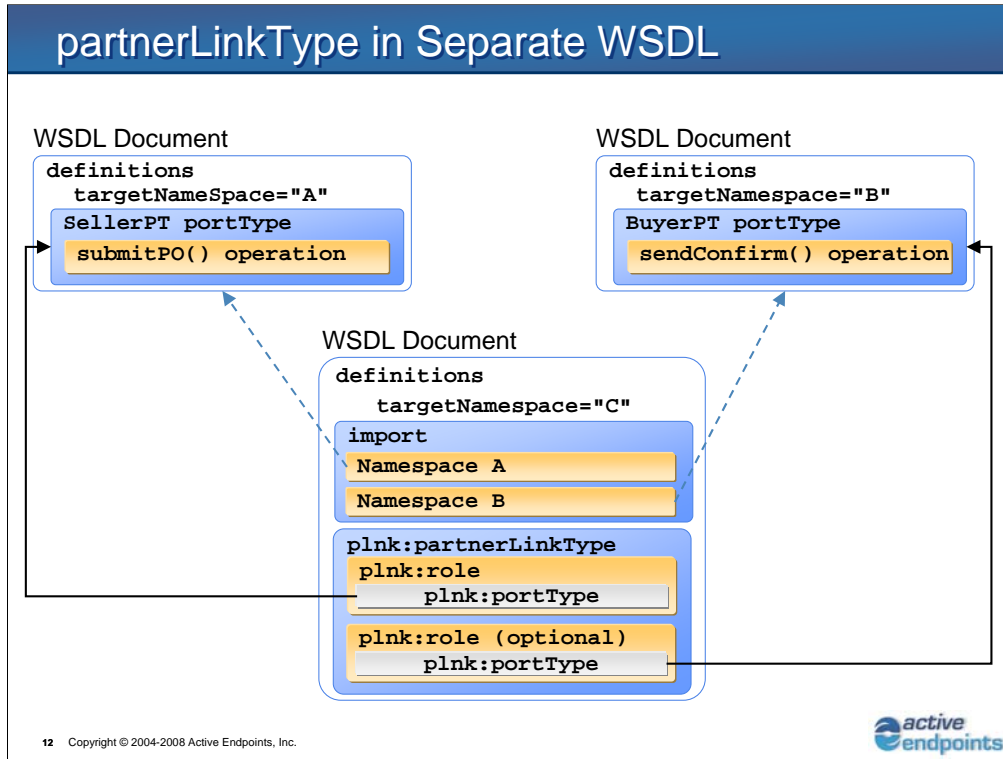
11 Copyright © 2004-2008 Active Endpoints, Inc.



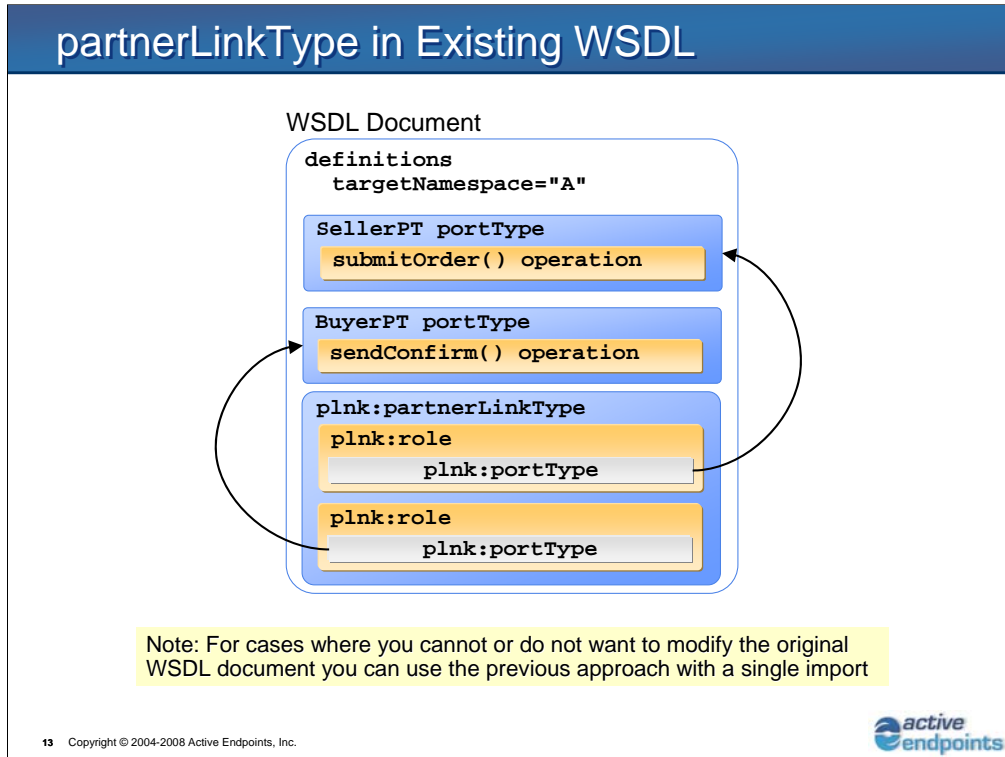
There are two different scenarios possible when defining a `PartnerLinkType`.

The first is where we are in a situation where we do not have access to the WSDLs that define the `portTypes` and therefore we cannot edit them. For example, it could be a WSDL that belongs to one of your business partners, and they've exposed the `portTypes` and messages, but you can't edit the original document. In this case, we can add one or more `Import` statements to our WSDL that will pull in the `portType` and message definitions. We can then create the `partnerLinkTypes` in our own WSDL using those imported definitions. (Note that in this situation our WSDL doc will have its own `targetNameSpace`.)

The second is where we are in a situation where we do have access to the WSDL documents where the `portTypes` are defined and we can edit those files. In this case we can insert the `partnerLinkTypes` directly into that WSDL document. (Note that in this case the `portTypes` will exist in the same `targetNamespace`.) This editing of the WSDL to create the `partnerLinkType` can be done through a manual edit of the file, by using a third party tool or through the use of ActiveVOS Designer's GUI.



This slide shows how we go about creating a PartnerLinkType in a separate WSDL document. We start out with two existing WSDL documents, A and B, shown at the top. WSDL A defines the SellerPT portType and WSDL B defines the BuyerPT portType. In this scenario we would create a third WSDL document called “C” and import the two other WSDLs, “A” and “B.” Now we have the components – i.e., the portTypes – we need to define our partnerLinkType in WSDL “C.”



Of course, it is much easier to add our partnerLinkType to an existing WSDL, because in that case no imports are necessary. Here we simply declare the partnerLinkType in WSDL “A”, using the existing portType definitions, which are also declared in WSDL “A.” We'd like to add some more information about the mechanics of imports and includes to this subject. WSDL files have Import capabilities only. Schema, however, have both Include and Import capabilities. The difference is that Include refers to artifacts in the *same* namespace, but an Import refers to artifacts in a *different* namespace.

Everything is a WSDL-defined Web Service

- WSDL defined Web services are used to represent all partner services required by the process
- Every BPEL process is itself a Web service described using WSDL

14 Copyright © 2004-2008 Active Endpoints, Inc.

Everything is a Web service to BPEL.

Once deployed, a BPEL process is also a WSDL-defined Web service.

With this idea as a foundation, we can create more and more complex processes based on these lower level Web service relationships.

Lab 1 – Getting Started

- Overview of Lab Exercises
 - Lab Files and Setup
 - Manually create a `partnerLinkType` in the `OrderManager.wsdl`

15 Copyright © 2004-2008 Active Endpoints, Inc.

This is the first lab in the BPEL Fundamentals course. Its purpose is to get you set up and ready for to do the other labs that follow. During the course of this lab you will download and install the lab instruction files, the lab solution files and the other resource files (i.e., .wsdl, .xsd and .xml) that you will need to start with, including the file "OrderManager.wsdl" You will then finish by adding a PartnerLinkType to this WSDL file manually.

When doing this (or any other) lab follow the instructions exactly as they guide you through the unzip and the install.

Do *not* ad lib the instructions.

Install the files *exactly* as instructed.

Do *not* change the path names or file names.

Do *not* improvise! IF YOU DO YOU WILL HAVE TROUBLE LATER ON.

One thing to take note of is that the solutions folder contains solution files for each of the labs, but these should only be used to *verify* your work against the solution.

Unit Summary

- Now you are familiar with:
 - Partner Interactions
 - Modeling partner interactions using the `partnerLinkType` construct
 - Where to define `partnerLinkTypes`