

Developing BPEL processes

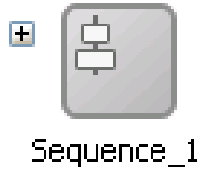
Third part: advanced BPEL concepts and examples



Table of contents

- BPEL: Sequence
- BPEL: Terminate
- BPEL: Empty
- BPEL: Scope
- BPEL: Wait
- BPEL “Correlation Sets”
- Example: Buyer and Seller
- BPEL: Flow
- BPEL: FlowN
- Exercise: Anagrams





BPEL: Sequence

- Resembles Java code blocks { }
- Defines a sequence of operation that have to be executed sequentially
- Sequences can be nested to generate more complex structures
- But:
 - Unlike Java code blocks, a sequence doesn't define a scope



exitProcess

BPEL: Terminate

- This activity halts process execution
- It works like Java “return” statement
 - Not like “System.exit()”
- Works everywhere inside a process: during normal execution flow, inside structured activities or an handler



emptyActivity

BPEL:Empty

- Like the name suggests, doesn't do anything
- Useful
 - for acting as a placeholder, TODO stubs
 - when inserting an activity is mandatory but no action has to be taken
 - e. g., catch and suppress a fault
 - for helping synchronization of concurrent execution branches inside a Flow activity



BPEL: Scope

- Works like Java scopes
 - Name must be unique inside the same scope
 - Nested visibility
 - Variable defined in an inner scope are not visible from outside the scope
 - Variable defined in an outer scope are visible in inner scopes
 - Inner scope variables hide outer scope ones
 - Variables defined at Process root are global
 - Variables inside a scope are local
 - The same is true for Fault handlers, Event handlers, Correlation sets



BPEL:Wait

- Stops process execution...
 - ...for a certain amount of time
 - Wait for n seconds, minutes, hours, days, ...
 - ...until a certain moment comes
 - Wait until 3.15 pm, April 13th, 2010
- Useful when waiting for another process/branch/activity to complete
- Similar to Java Object.wait(n) method
 - But this activity cannot be interrupted

BPEL “Correlation Sets”

- Until now, we supposed implicitly each BPEL process has just one instance running at the same time
 - May be true in a development environment
 - Typical BPEL production environment aims at offering services to many customers simultaneously
 - BPEL processes expect nothing less from web services they query
- Many instances of the same process executed at the same time
- Message routing problem!

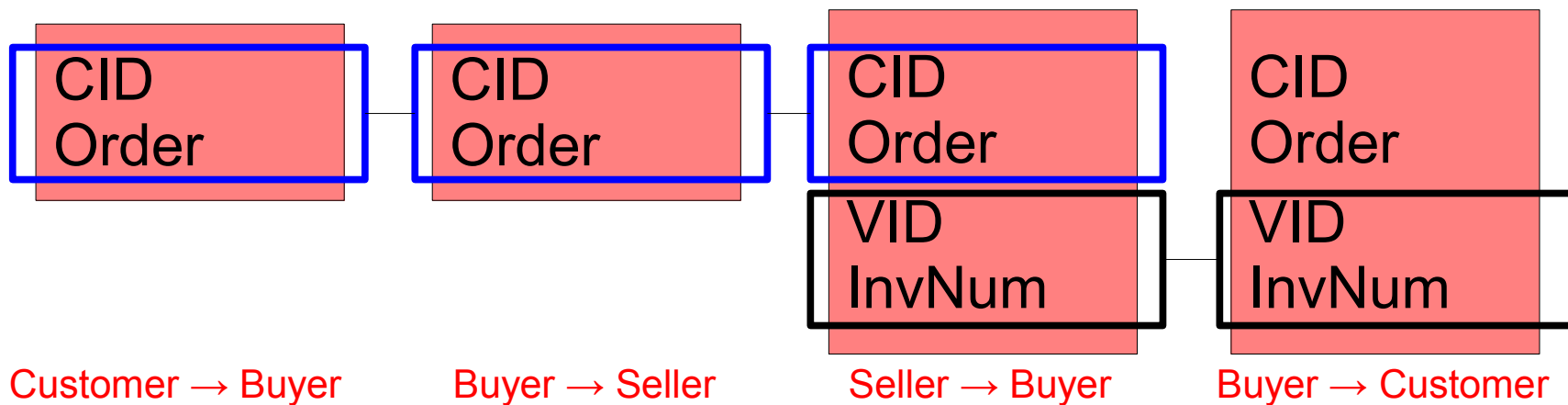


BPEL “Correlation Sets” (2)

- An example:
 - A Buyer BPEL process
 - Receives orders from several customers
 - Forward orders to a Seller process
- Customers don't wait for previous requests to be completed
 - Many process instances running at the same time
- Seller answers asynchronously
 - Reply messages from Seller may arrive in random order

BPEL “Correlation Sets” (3)

- How to manage this?
- **Putting information that helps BPEL engine choose the right instance inside exchanged messages**
- In our example:



BPEL “Correlation Sets” (4)

- In BPEL they are called Correlation Sets
- To use them, modify
 - WSDLs
 - Define **properties** (those parts of exchanged messages that don't change)
 - Assign them a **property alias** (so properties may have different actual names)
 - BPEL
 - Define **correlation sets** (group of one or more aliases)
 - Use them inside Receive / Reply / Invoke activities



BPEL “Correlation Sets” (5)

- Inside “correlated activity”, specify
 - initiate
 - “yes”: extract data from the message and create a new correlation set
 - “no”: use existing correlation set
 - pattern
 - “out”: correlation applies to outbound message content
 - “in”: correlation applies to inbound message content
 - “out-in”: on both messages (synchronous invoke)
- Warning: BPEL 2.0 specification differs slightly from Oracle BPEL process manager implementation at this point

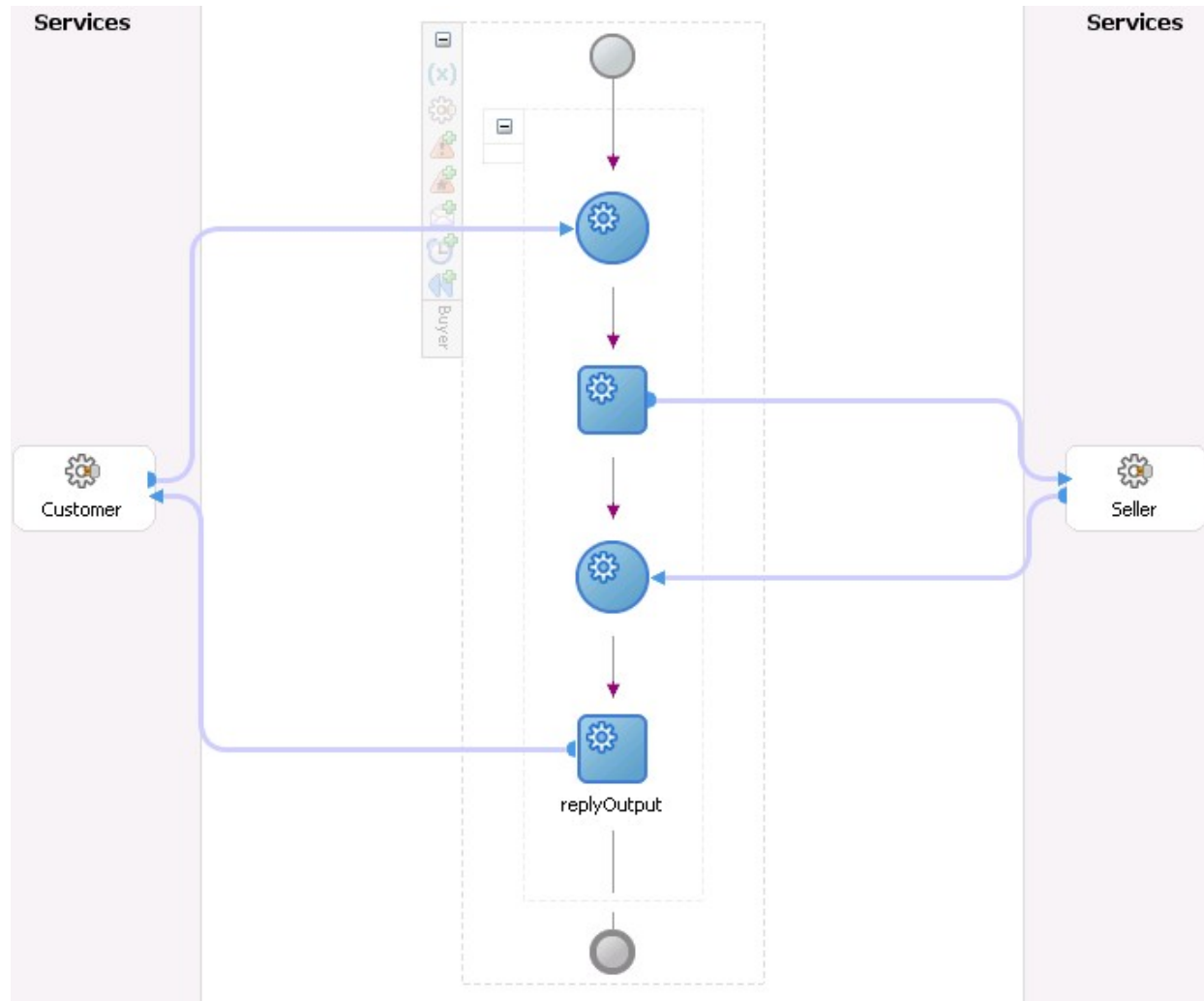


Example: Buyer and Seller

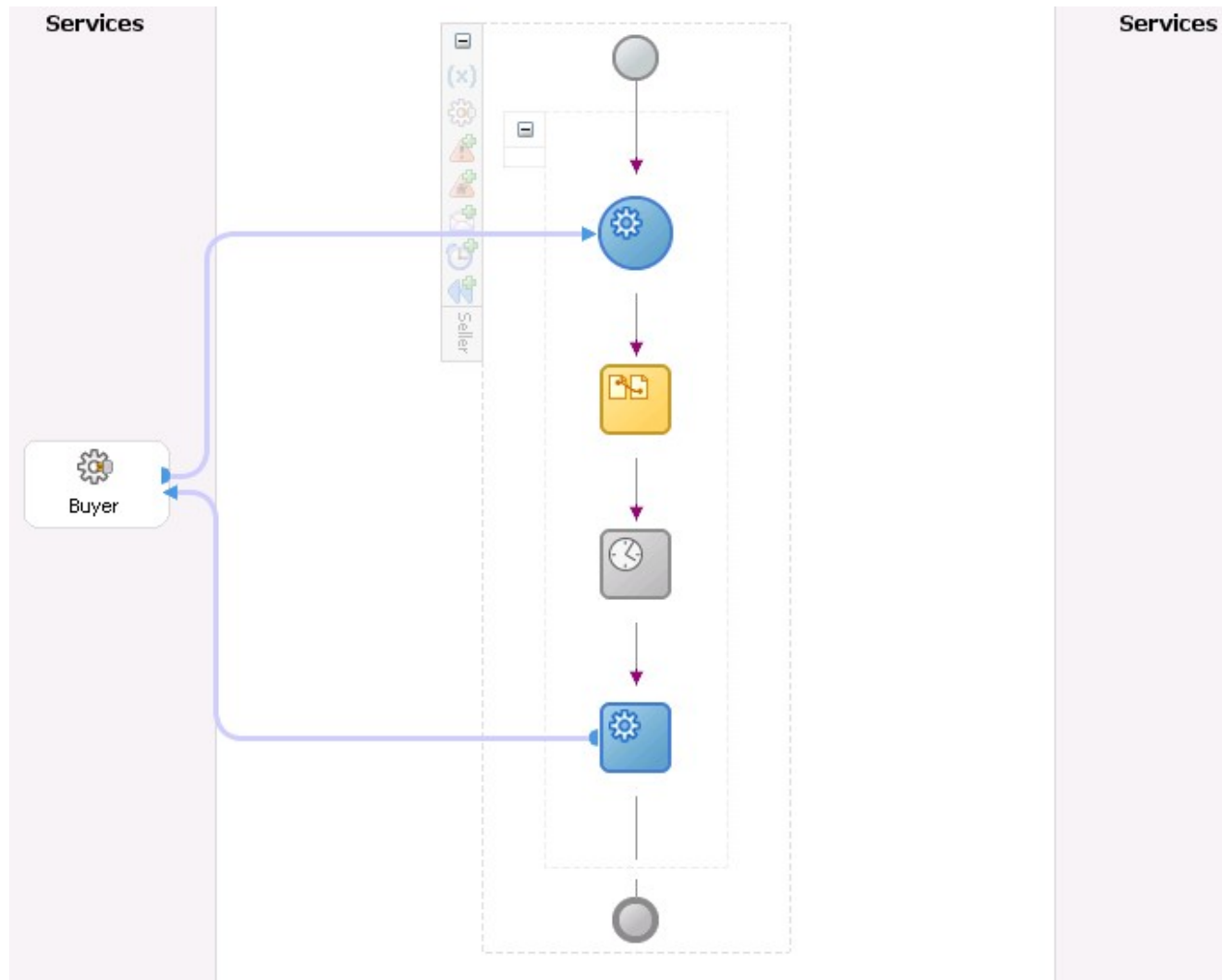
- This example is introduced in the BPEL specification for explaining correlation sets
- Oracle BPEL process manager offers a working version of it
- Two partners, a buyer and a seller
 - Shared content
 - properties definition (in a WSDL file)
 - properties aliases (in a WSDL file)
 - correlation sets definition (in each process' BPEL file)
- Two correlation sets
 - Purchase Order
 - Invoice



Example: Buyer and Seller (2)



Example: Buyer and Seller (3)



BPEL:Flow

- Introducing parallel processing in BPEL
- Allows (groups of) activities to be executed concurrently
 - Execution order is **non deterministic**
 - The groups are called **branches** or **paths**
- Flow is a structured activity
 - Each execution path can contain any number of activities
- Flow activity terminates when all enclosed execution paths are terminated



BPEL:Flow (2)

- Flow activity allows concurrent (parallel) programming
 - Single thread vs. multi threaded code
- Something similar, written in Java:

```
Thread[] threads = new Thread[n];  
...  
threads[n] = new Thread(new Runnable() {  
public void run() { ... } });  
...  
for(Thread toStart : threads) toStart.start();  
for(Thread toEnd : threads)  
try {  
toEnd.join();  
} catch (InterruptedException e) {}
```



BPEL:FlowN

- Oracle proprietary extension of BPEL 1.1 specification
 - Works like BPEL 2.0 'ForEach'
 - But doesn't support sequential executing
- A special case of Flow
 - When each branch does the same job
- Has two parameters
- Does not define a scope!

Exercise: Anagrams

- The idea: take advantage of FlowN activity to anagram a sentence
- Set FlowN 'N' parameters to the length of the input string
- Each FlowN branch add the character at position = index of the input string to the output string
 - Suggestion: declare a temporary variable
- You will notice that the result sometime isn't “random” apparently
 - In fact, it looks like output string = input string
 - Why?
 - FlowN is too fast: instantiation and execution of new branches are almost instantaneously
 - We'll see a way to overcome this problem in the next lesson