BPEL: Building Standards-Based Business Processes with Web Services

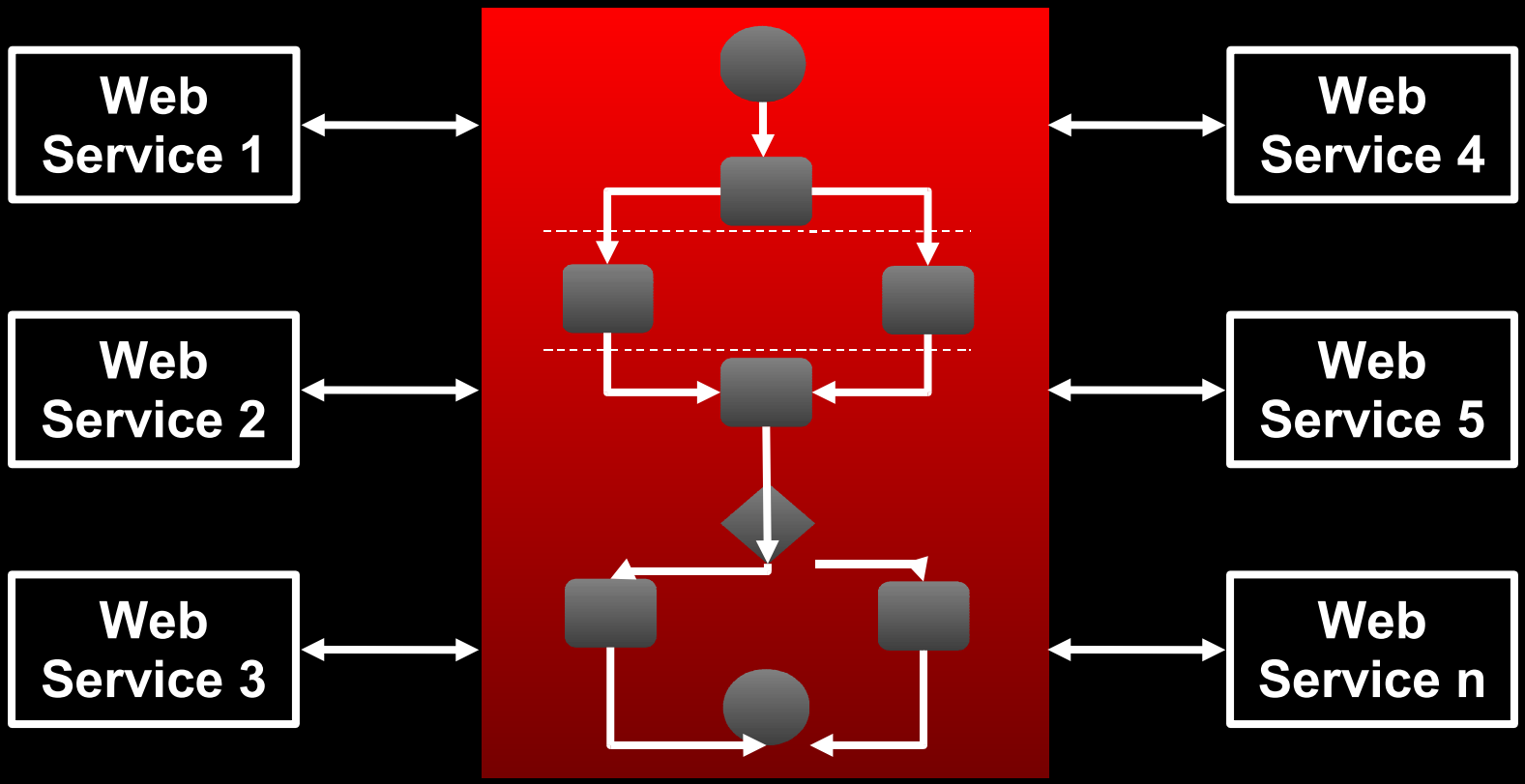
Acknowledgement: This is a slightly modified version of Oracle's BPEL presentation.

Agenda

- ☒ What is and Why BPEL?
- ☒ Orchestration vs. Choreography
- ☒ BPEL vocabulary
  - Activities
  - Partners
  - Variables
  - Flow, Link, Correlation
- ☒ Scopes of BPEL
- ☒ BPEL and Java technology
- ☒ Open issues of BPEL

Why BPEL?

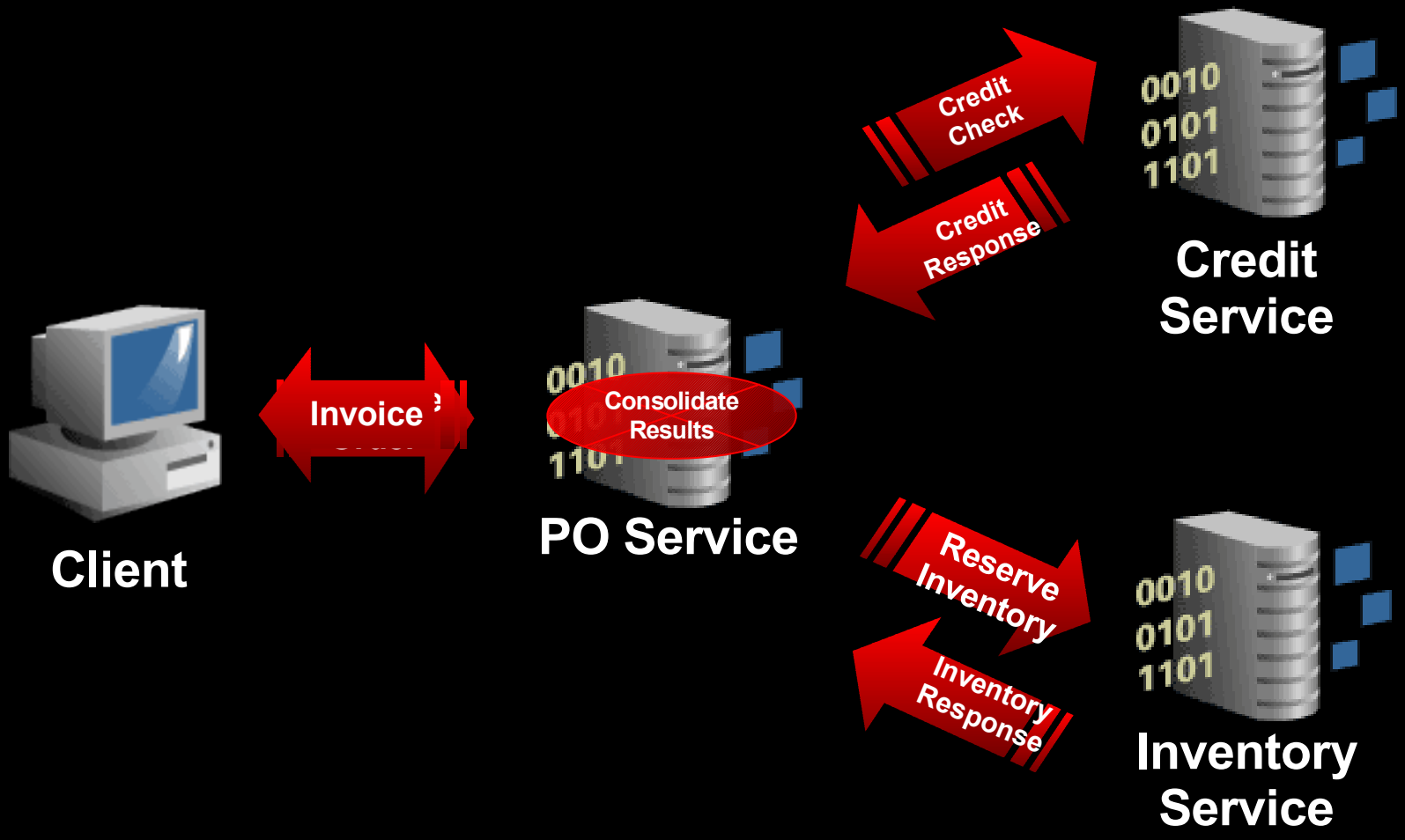
Web Services Meet Business Processes



Need for Composition of Web Services

- Business applications have to interoperate and integrate
- The most recent answer to the integration challenge is the Service Oriented Architecture (SOA) and the web services technologies
  - Different business applications exposing their functionalities through web services
- Developing the web services and exposing the functionalities is not sufficient. We also need a way to compose these functionalities in the right order (Business process)

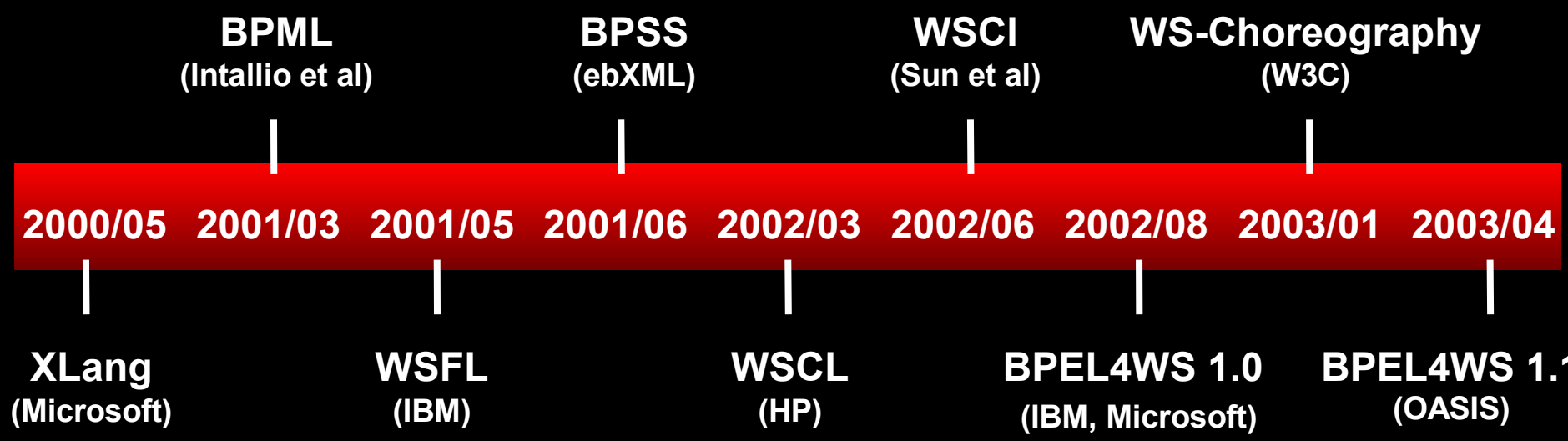
Example Problem Space



Business Process Challenges

- Coordinate asynchronous communication between services
- Correlate message exchanges between parties
- Implement parallel processing of activities
- Implement compensation logic (Undo operations)
- . . .
- Manipulate/transform data between partner interactions
- Support for long running business transactions and activities
- Provide consistent exception handling
- Need for universal data model for message exchange
- . . .

Recent History of Business Process Standards



What is BPEL?

Web Services Business Process Execution Language

- Version 1.0 released by IBM, Microsoft and BEA in August 2002
  - Accompanied by WS-Coordination, WS-Transaction which remain unsubmitted to standards bodies
- Version 1.1 submitted to OASIS April 2003
- Version 1.2 draft spec available (May 2005)
- XML language for describing business processes based on Web services
  - Convergence of XLANG (Microsoft) and WSFL (IBM)
- Unprecendented industry consensus
  - IBM, Microsoft, Oracle, Sun, BEA, SAP, Siebel …

BPEL “Fixes” WSDL

- ☒ WSDL: unordered set of operations
  - Operations are message exchanges
- ☒ Need rules for ordering

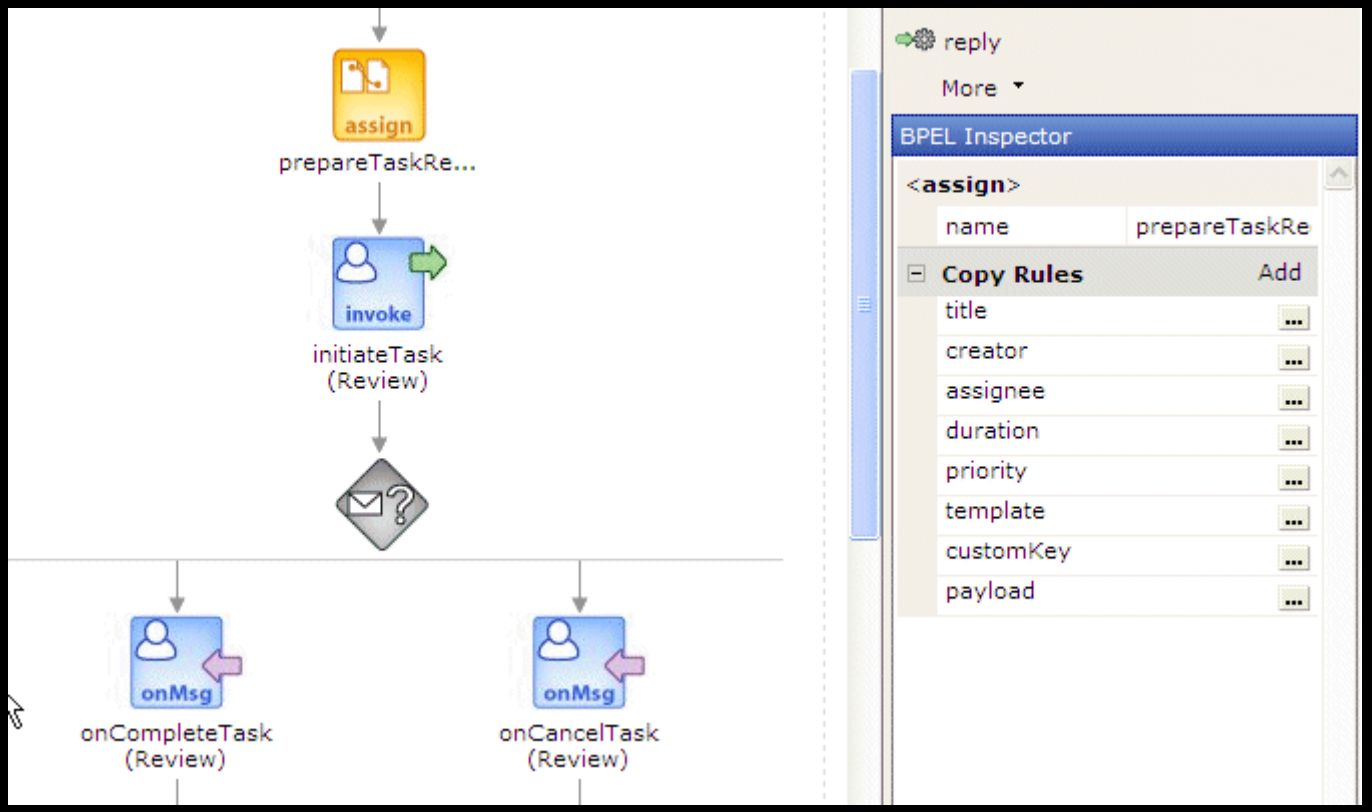
BPEL Is a Web Service Sequencing Language

- Process defines “conversation” flow chart
  - Conversation consists of only WSDL-described message exchanges.
- Process instance is a particular conversation following the chart
  - Execution systems can support multiple concurrent conversations.

BPEL Is a Graph-oriented Language

- The XML-based nature of BPEL processes enables the editing of process logic using visual design and modeling tools
- Activity Nodes
  - Represent message exchanges, internal operations, decision points
- Arcs
  - Order of execution
  - Constraints on concurrent processing
- Directed acyclic graph
  - Much like Java
- Separate error & compensation handling

Example



Source: Collaxa *BPEL Designer*

BPEL is a Web Service Composition Language

- ☒ Consumes services (invoke)
- ☒ Creates services (receive/reply)
- ☒ Aggregates fine-grained services
- ☒ Creates coarser-grained service

Value Proposition of WS-BPEL

- ☒ Portable business processes
  - Built on top of an interoperable infrastructure of Web services
- ☒ Industry wide language for business processes
  - Common skill set and language for developers
- ☒ Choice of process engines
  - Standards lead to competitive offerings

Usage of BPEL

- Within an enterprise (Intranet)
  - To standardize enterprise application integration and extend the integration to previously isolated systems
- Between enterprises (Internet)
  - Enable easier and more effective integration with business partners

BPEL Extends Web Services

- BPEL is the key technology in environments where functionalities already are or will be exposed via web services
- With increases in the use of web service technology the importance of BPEL will rise further

BPEL Process as a Web Service

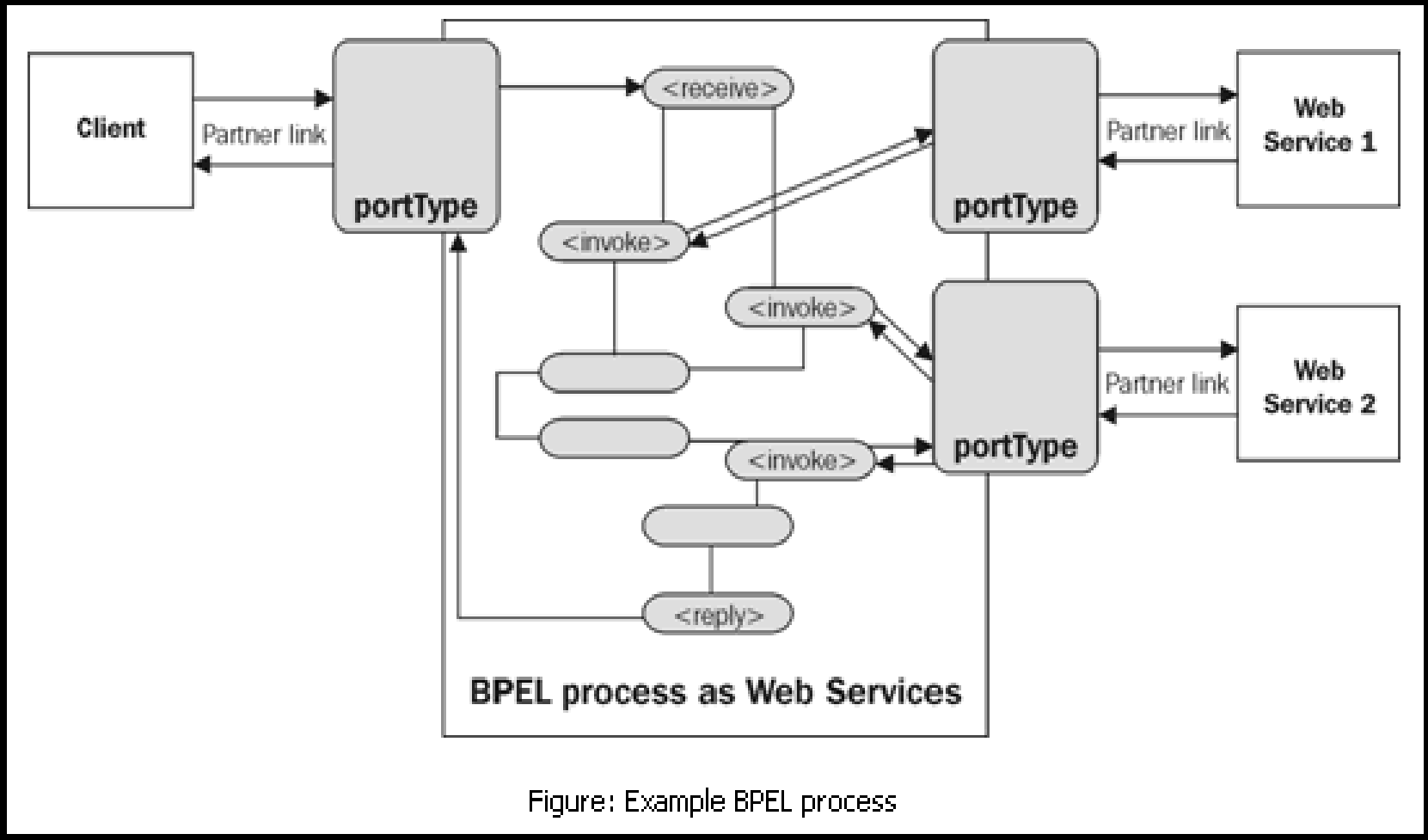


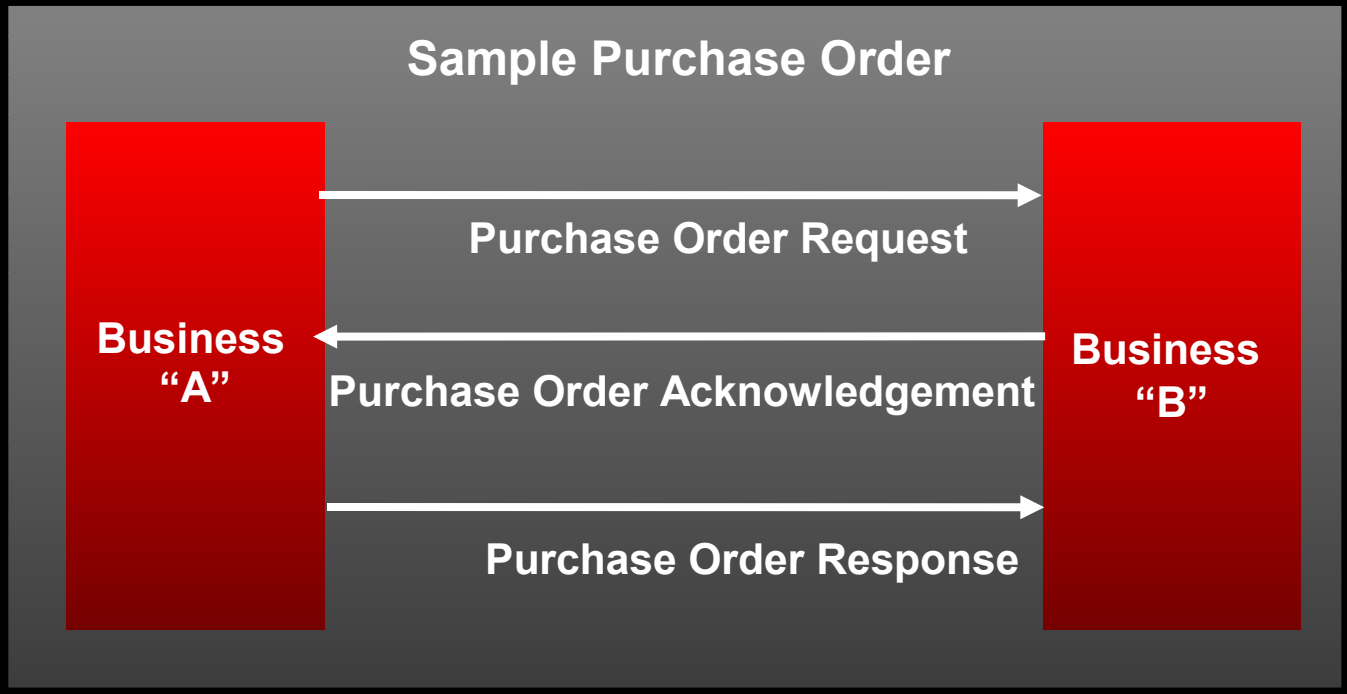
Figure: Example BPEL process

Orchestration vs. Choreography

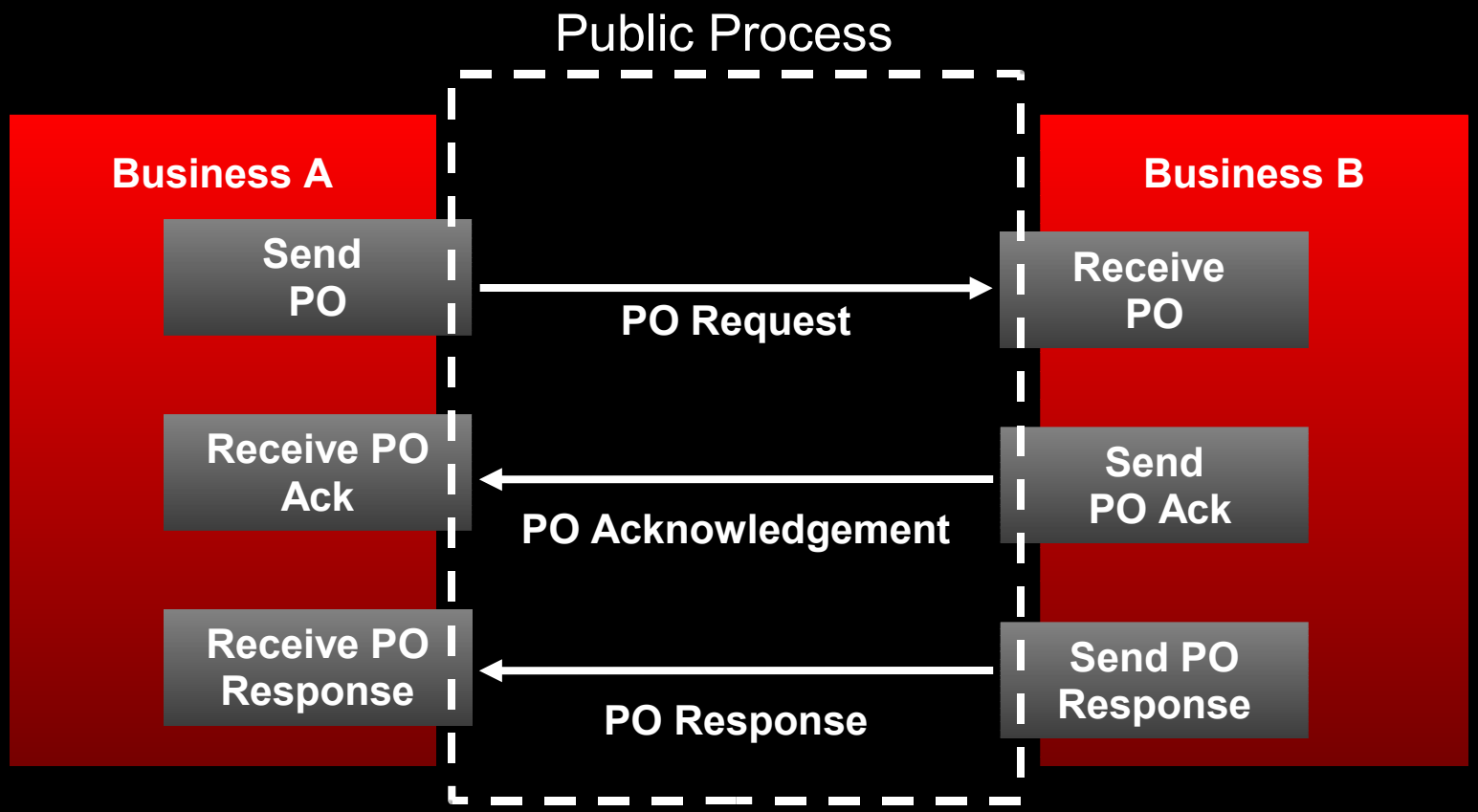
Orchestration vs. Choreography

- Orchestration
  - An executable business process describing a flow from the perspective and under control of a single endpoint (commonly: Workflow)
  - BPEL handles Orchestration
- Choreography (WSDL)
  - The observable public exchange of messages, rules of interaction and agreements between two or more business process endpoints
  - WSDL handles Choreography
  - CDL4WS

Sample Business Process: Purchase Order

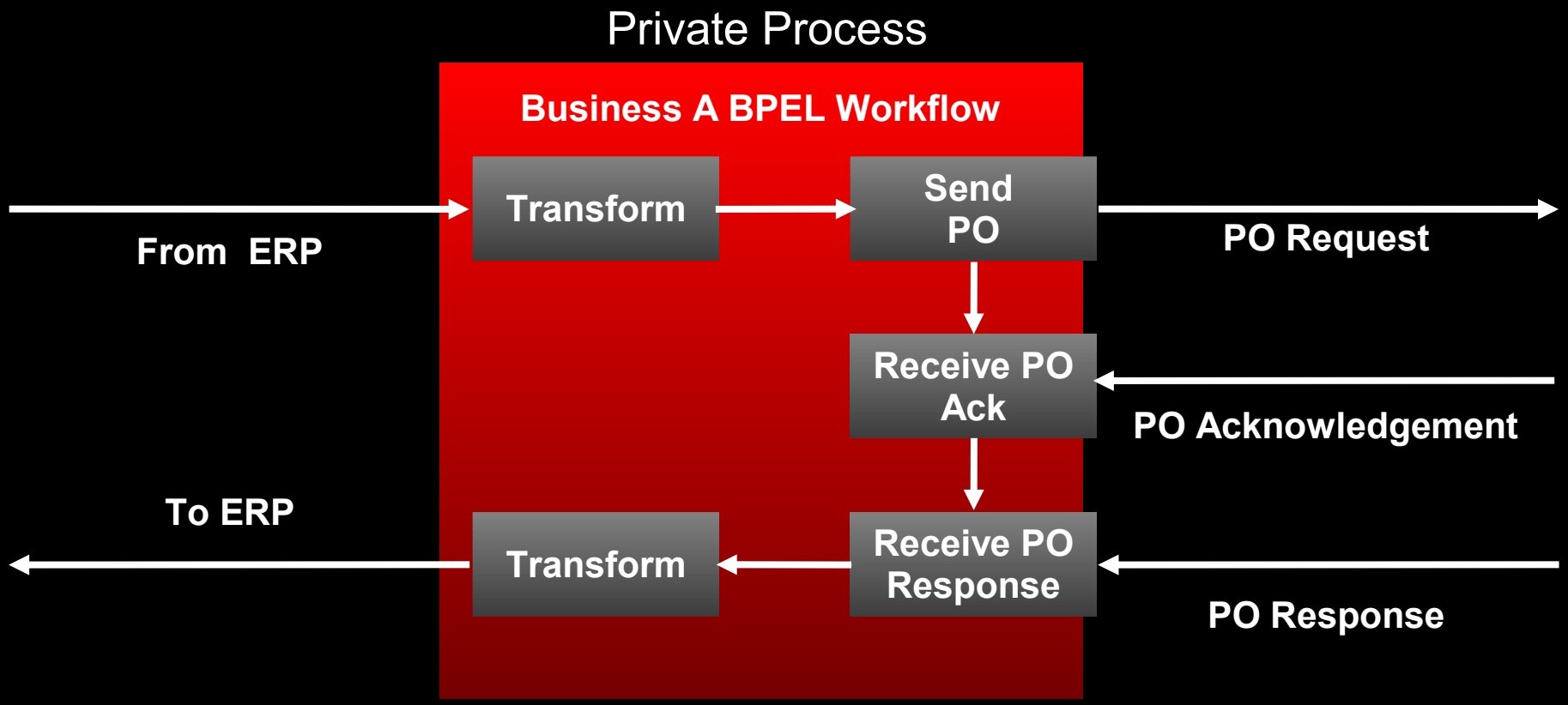


From a Choreography Perspective



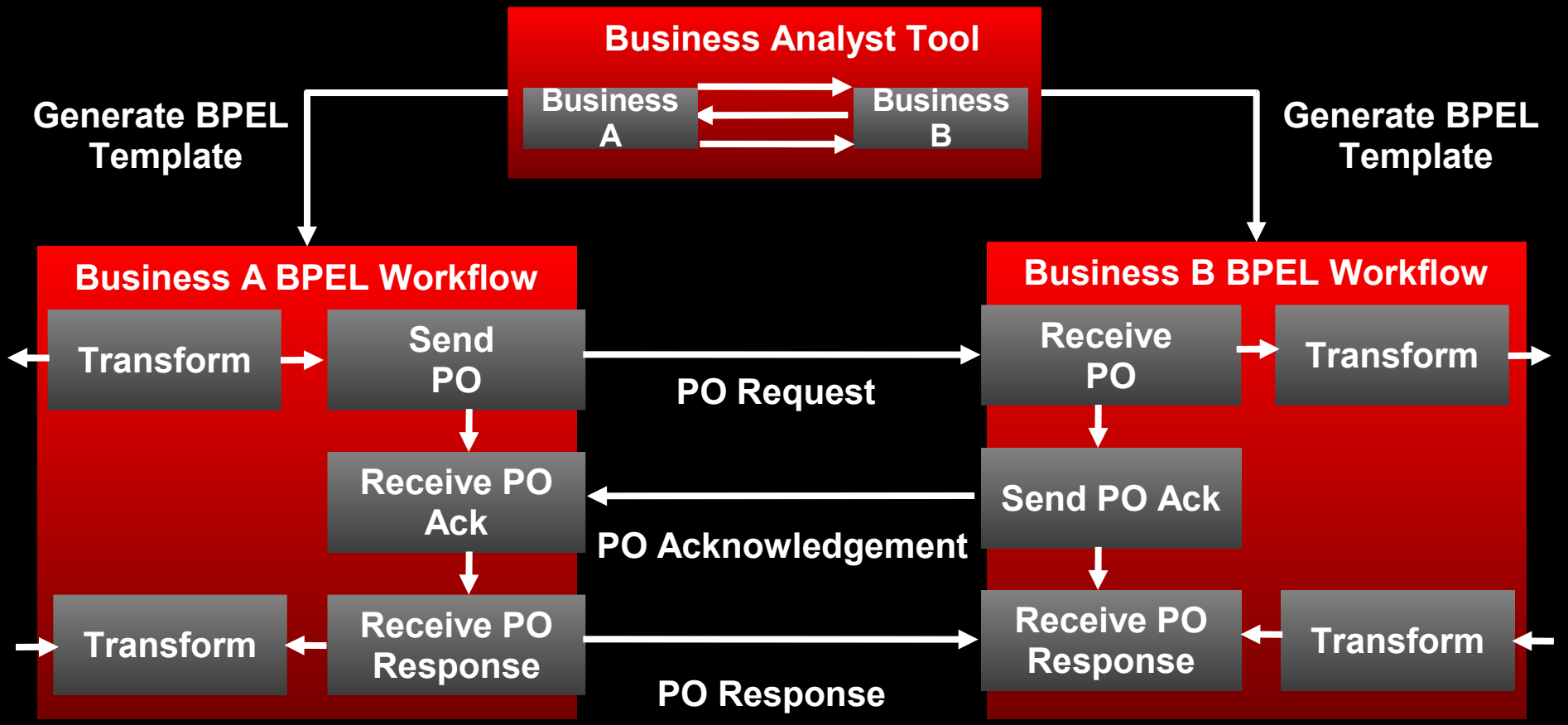
Choreography – The observable public exchange of messages

From an Orchestration Perspective



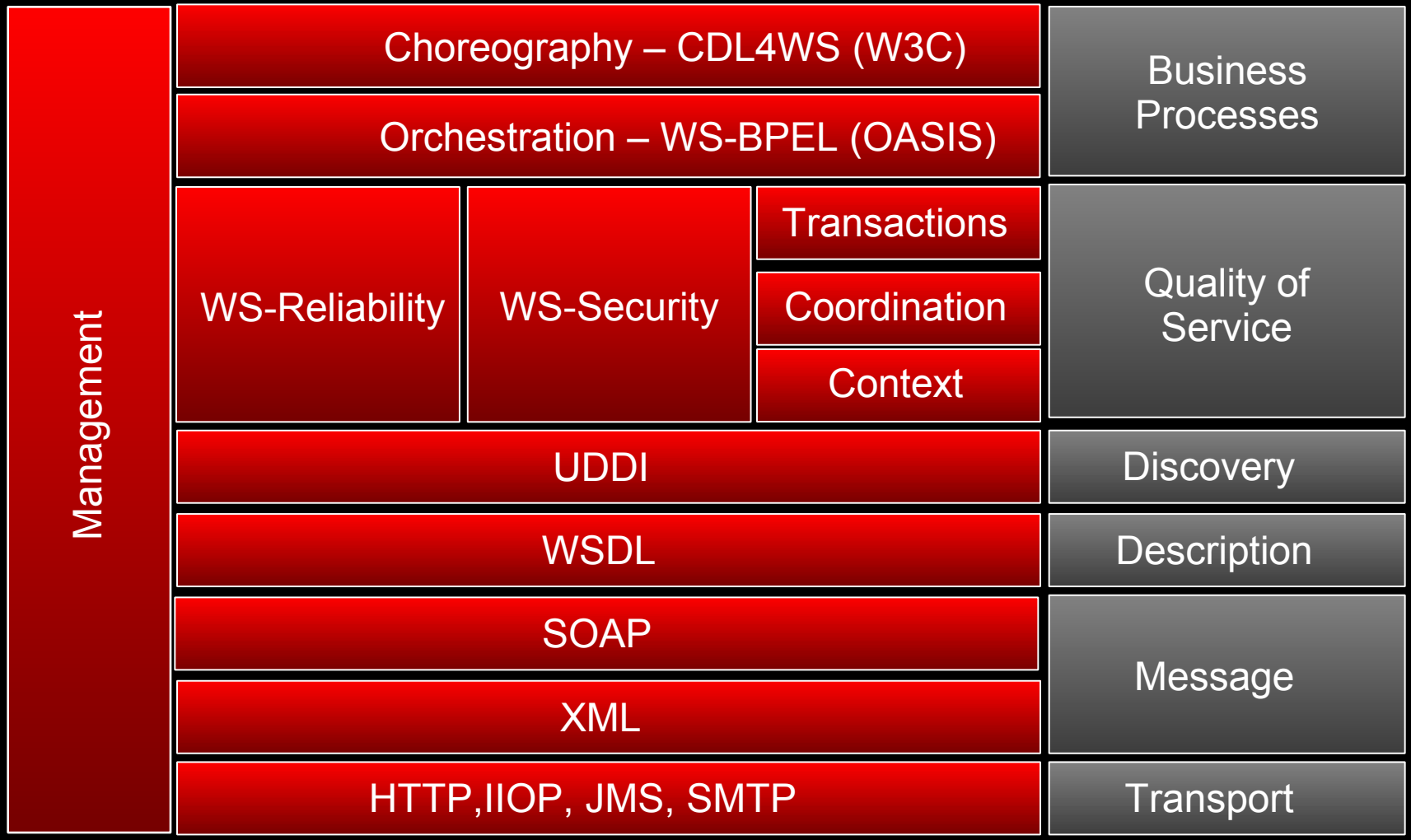
Orchestration – A private executable business process

Orchestration and Choreography Together

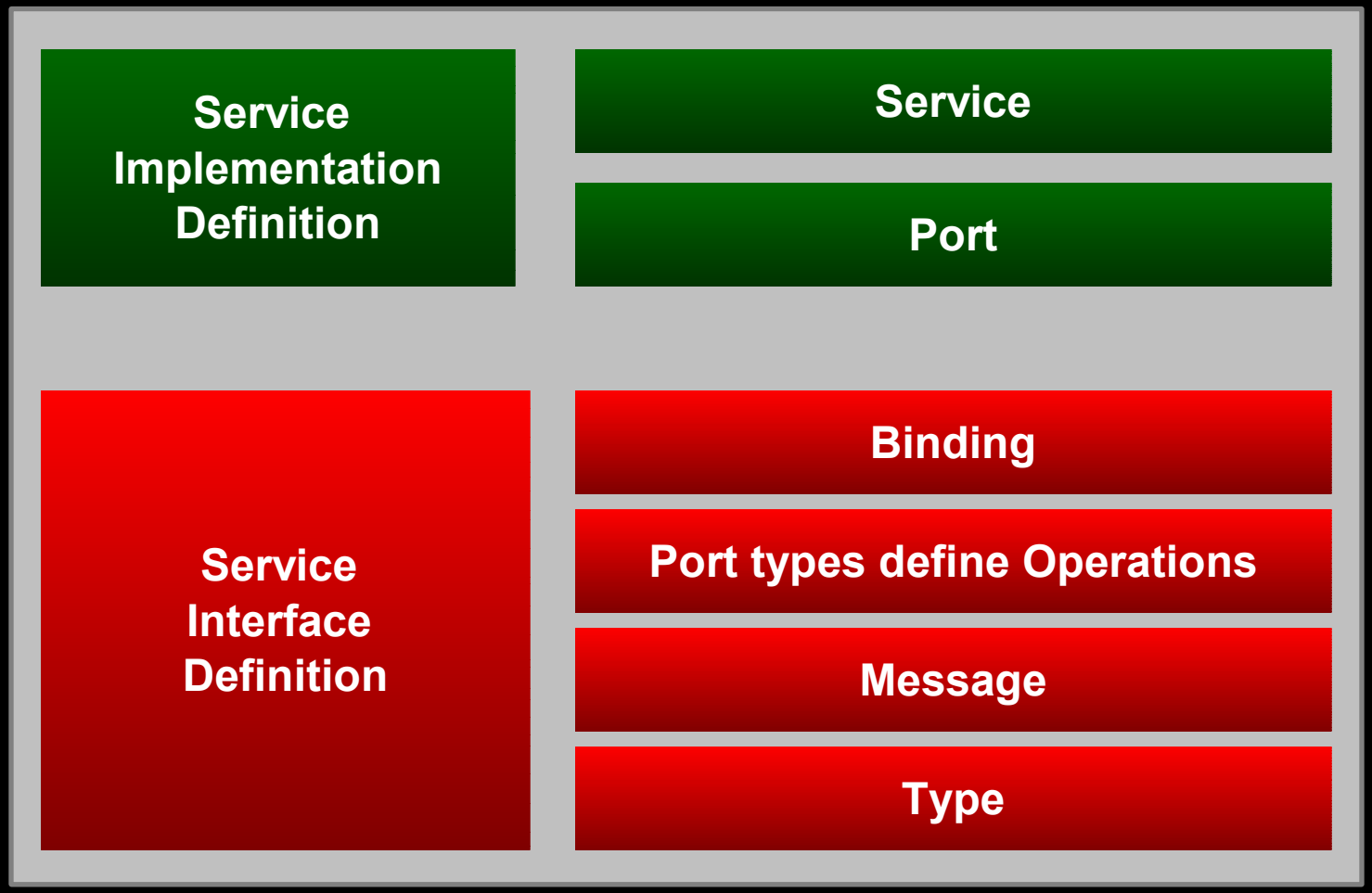


Two BPEL workflow templates reflecting a business agreement

Standards Building Blocks of BPEL



BPEL Depends on WSDL and WSDL Extensions



BPEL Vocabulary:
Activities

BPEL Document Structure

```
<process>
   <!- Definition and roles of process participants -->
   <partnerLinks> ... </partnerLinks>
   <!- Data/state used within the process -->
   <variables> ... </variables>
   <!- Properties that enable conversations -->
   <correlationSets> ... </correlationSets>
   <!- Exception handling -->
   <faultHandlers> ... </faultHandlers>
   <!- Error recovery – undoing actions  -->
   <compensationHandlers> ... </compensationHandlers>
   <!- Concurrent events with process itself  -->
   <eventHandlers> ... </eventHandlers>
   <!- Business process flow  -->
   (activities)*
</process>
```

BPEL Activities

- A BPEL process consists of steps
- Each step is called an activity
- Two types of BPEL activities
  - basic activities
  - structured activities.

BPEL Activities

Basic Activities

- ☒ <invoke>
- ☒ <receive>
- ☒ <reply>
- ☒ <assign>
- ☒ <throw>
- ☒ <wait>
- ☒ <empty>

Structured Activities

- ☒ <sequence>
- ☒ <while>
- ☒ <pick>
- ☒ <flow>
- ☒ <scope>
- ☒ <compensate>
- ☒ <switch>
- ☒ <link>

Basic Activities

- Invoke
  - Allows the business process to invoke a one-way or request/response operation on a portType offered by a partner
- Receive
  - Allows the business process to do a blocking wait for a matching message to arrive
  - Can be the instantiator of the business process
- Reply
  - Allows the business process to send a message in reply to a message that was received through a <receive>
  - The combination of a <receive> and a <reply> forms a request-response operation on the WSDL portType for the process.

Basic Activities

- Assign
  - Can be used to update the values of variables with new data
- Throw
  - Generates a fault from inside the business process
- Wait
  - Allows you to wait for a given time period or until a certain time has passed
- Empty
  - Allows you to insert a "no-op" instruction into a business process
  - This is useful for synchronization of concurrent activities, for instance

Structured Activities

- Sequence
  - Allows you to define a collection of activities to be performed sequentially in lexical order
- While
  - Allows you to indicate that an activity is to be repeated until a certain success criteria has been met
- Pick
  - Allows you to block and wait for a suitable message to arrive or for a time-out alarm to go off
  - When one of these triggers occurs, the associated activity is performed and the pick completes.

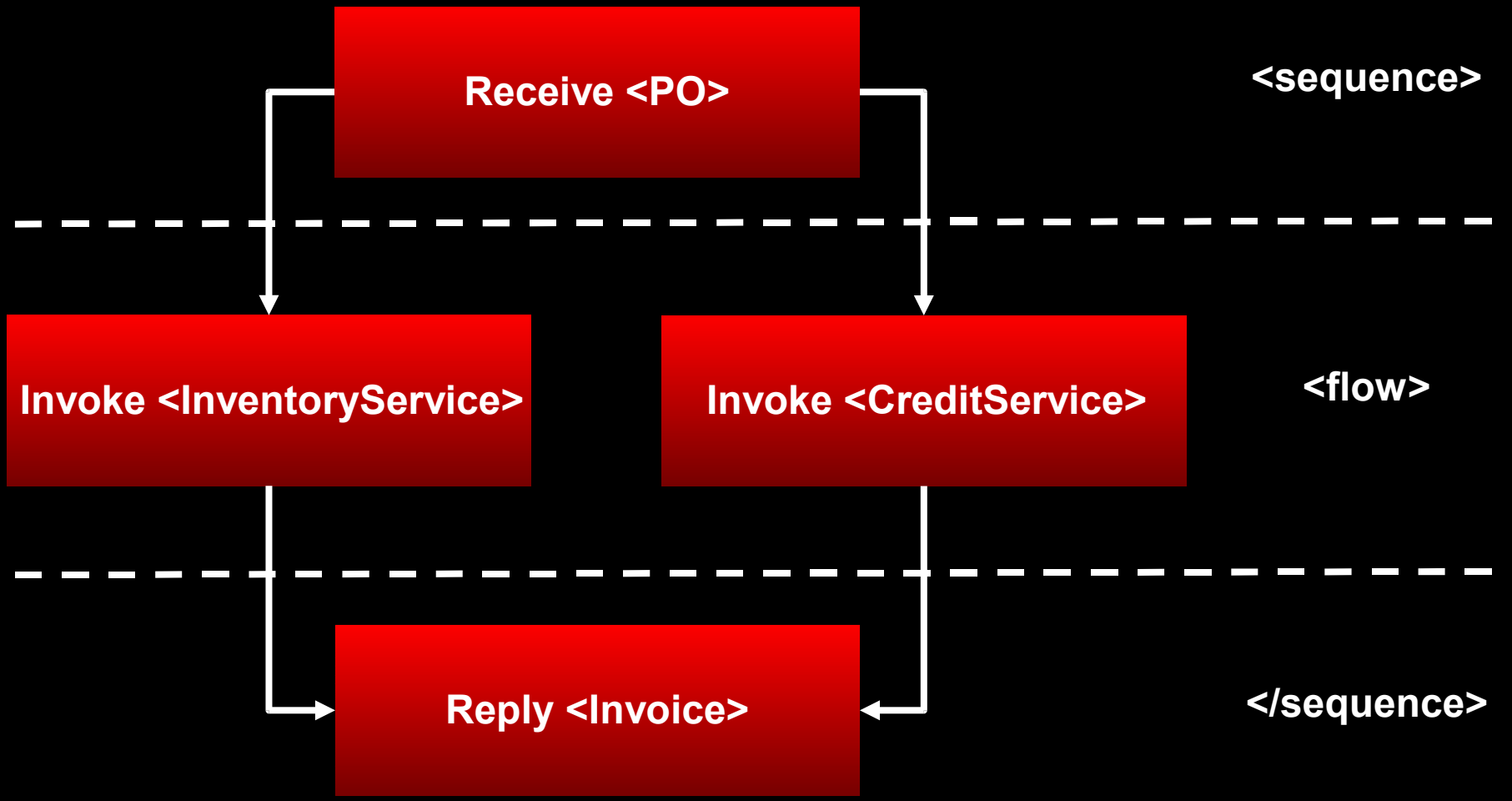
Structured Activities

- Flow
  - Allows you to specify one or more activities to be performed concurrently. Links can be used within concurrent activities to define arbitrary control structures.
- Scope
  - Allows you to define a nested activity with its own associated variables, fault handlers, and compensation handler
- Compensate
  - Used to invoke compensation on an inner scope that has already completed normally
  - This construct can be invoked only from within a fault handler or another compensation handler

Structured Activities

- Switch
  - Supports conditional behavior in a pattern that occurs quite often
  - The activity consists of an ordered list of one or more conditional branches defined by case elements, followed optionally by an otherwise branch
- While
  - Supports repeated performance of a specified iterative activityAllows you to define a nested activity with its own associated variables, fault handlers, and compensation handler

Basic Activities Combined with Structured Activities



Sample Activities in BPEL

```
<sequence>
 <receive partnerLink="customer" portType="lns:purchaseOrderPT"
          operation="sendPurchaseOrder" variable="PO"
          createInstance="yes" />
 <flow>
  <invoke partnerLink="inventoryChecker" portType="lns:inventoryPT"
          operation="checkINV" inputVariable="inventoryRequest"
          outputVariable="inventoryResponse" />

  <invoke partnerLink="creditChecker" portType="lns:creditPT"
          operation="checkCRED" inputVariable="creditRequest"
          outputVariable="creditResponse" />
 </flow>
 ...
 <reply partnerLink="customer" portType="lns:purchaseOrderPT"
        operation="sendPurchaseOrder" variable="invoice"/>
</sequence>
```

Wait Example

```
<wait standard-attributes>

    standard-elements

      ( <for expressionLanguage="anyURI"?>duration-expr</for> |

        <until expressionLanguage="anyURI"?>deadline-expr</until> )

</wait>
```

Switch Example

```
<switch standard-attributes>
    standard-elements
    <case>+
      <condition expressionLanguage="anyURI"?>
        ... bool-expr ...
      </condition>
        activity
    </case>
   <otherwise>?
        activity
   </otherwise>
</switch>
```

While Example

```
<while standard-attributes>
   standard-elements

   <condition expressionLanguage="anyURI"?>
     ... bool-expr ...
   </condition>

     activity
</while>
```

Switch Example

```
<switch standard-attributes>
    standard-elements
    <case>+
      <condition expressionLanguage="anyURI"?>
        ... bool-expr ...
      </condition>
        activity
    </case>
   <otherwise>?
        activity
   </otherwise>
</switch>
```

BPEL Vocabulary:
Partner Links

Two Ways BPEL Process Interact with External Web Services

- The BPEL process invokes operations on other web services
  - represented by invoked partnerLink
- The BPEL process receives invocations from clients
  - represented by client partnerLink
- For its clients a BPEL process looks like any other web service. When we define a BPEL process, we actually define a new web service that is a composition of existing services

BPEL as a Web Service

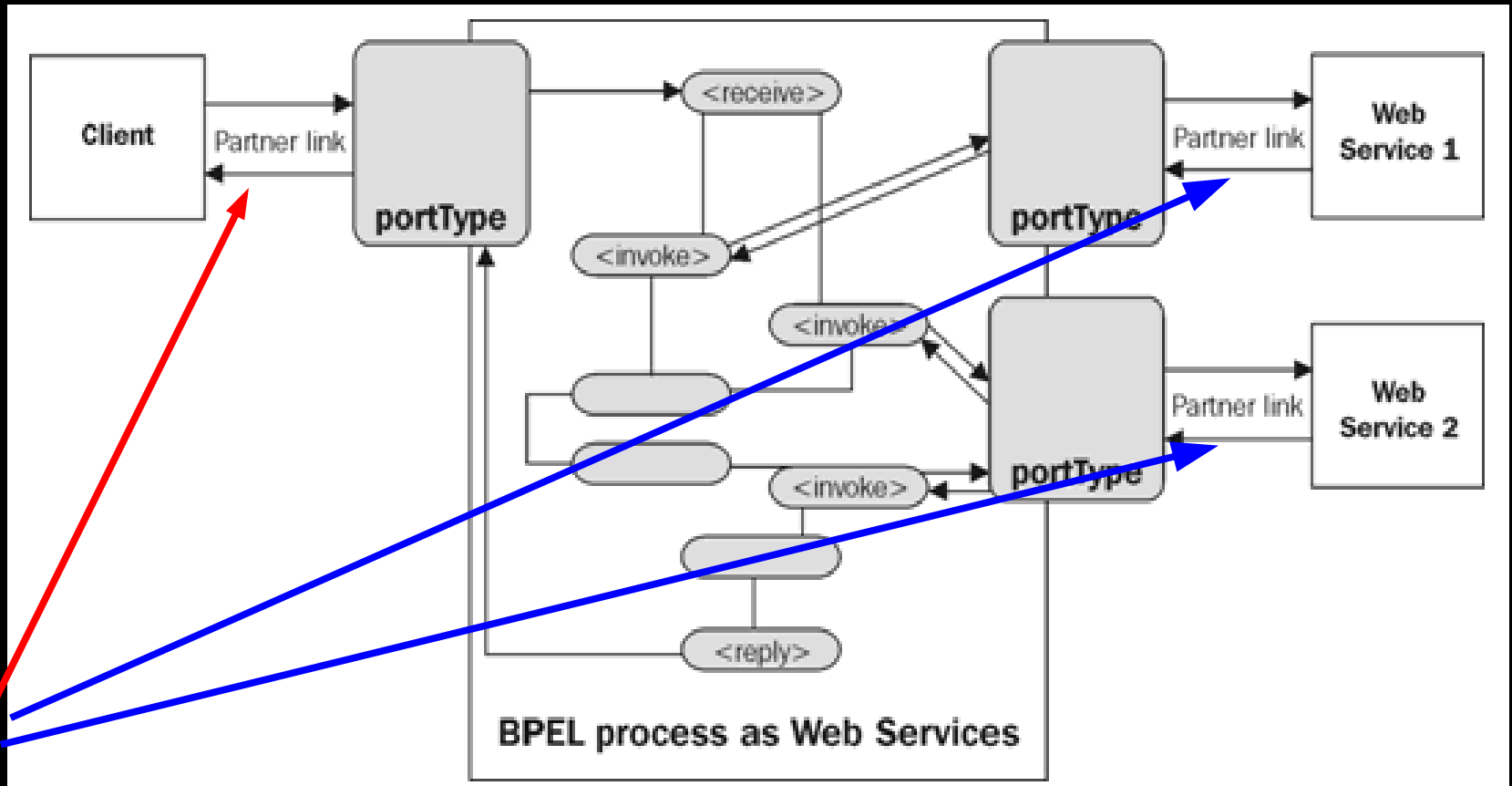
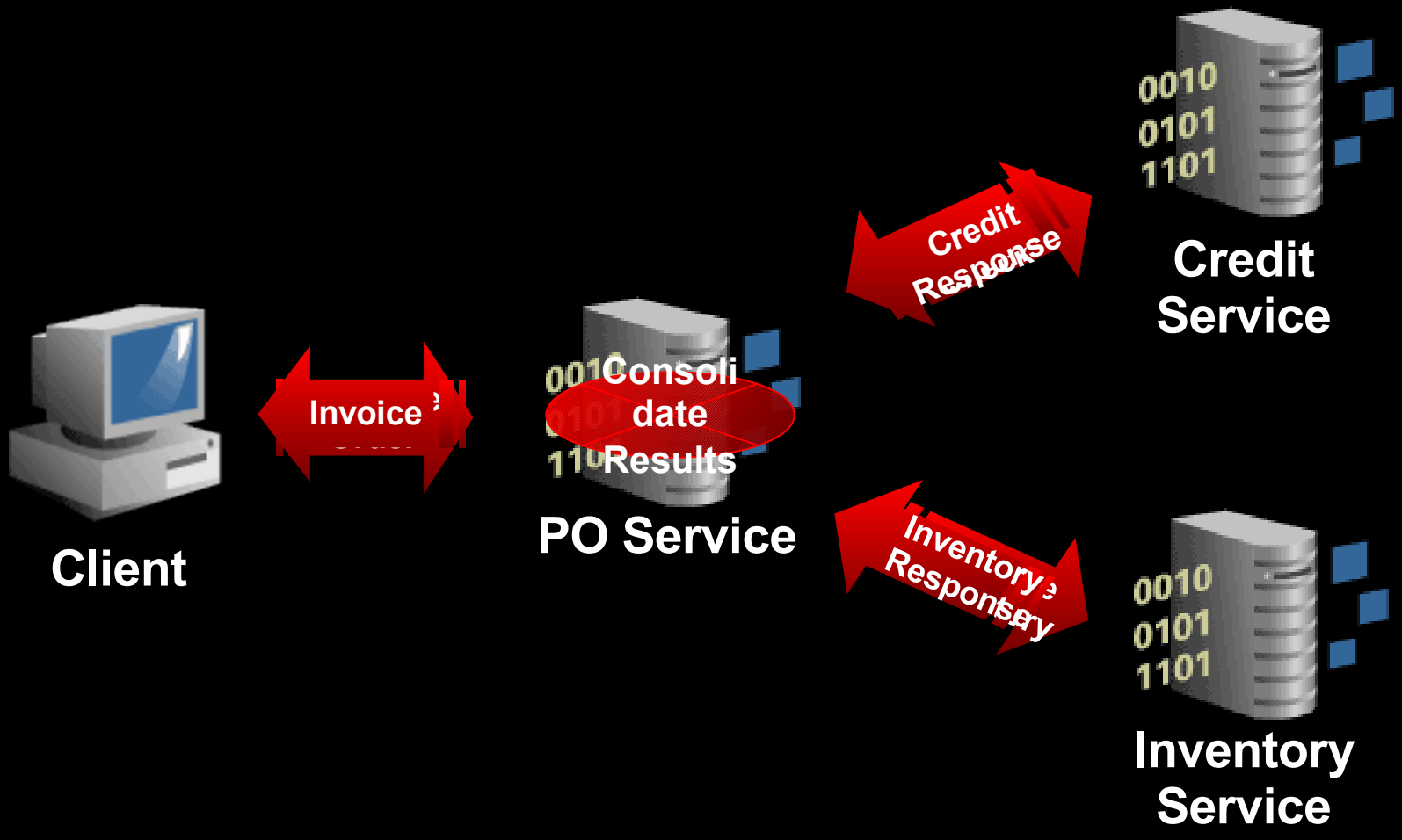


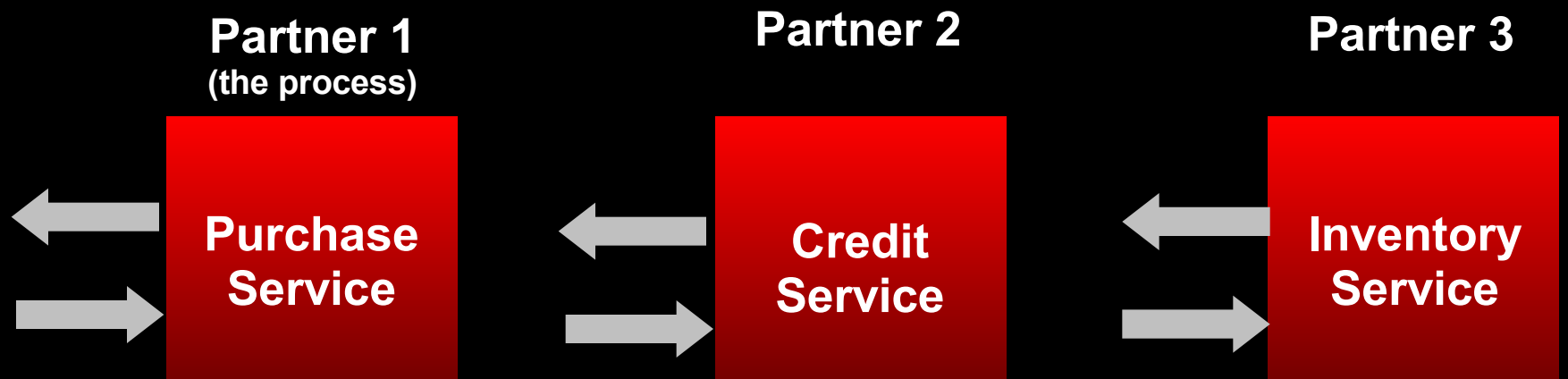
Figure: Example BPEL process

Example Problem Space



Partner Links

- The services with which a business process interacts are modeled as partner links
- Each partner link is characterized by a partnerLinkType
- The role of the business process itself is indicated by the attribute myRole and the role of the partner is indicated by the attribute partnerRole.



Partner Link

```
<partnerLinks>
    <partnerLink name="ncname"
                 partnerLinkType="qname"
                 myRole="ncname"?
                 partnerRole="ncname"?>+
    </partnerLink>
</partnerLinks>
```

Partner Link Example

```
<?xml version="1.0" encoding="utf-8"?>
<process name="insuranceSelectionProcess"
        targetNamespace="http://packtpub.com/bpel/example/"
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:ins="http://packtpub.com/bpel/insurance/"
        xmlns:com="http://packtpub.com/bpel/company/" >

   <partnerLinks>
     <partnerLink name="client"
                partnerLinkType="com:selectionLT"
                myRole="insuranceSelectionService"/>

     <partnerLink name="insuranceA"
                partnerLinkType="ins:insuranceLT"
                myRole="insuranceRequester"
                partnerRole="insuranceService"/>

     <partnerLink name="insuranceB"
                partnerLinkType="ins:insuranceLT"
                myRole="insuranceRequester"
                partnerRole="insuranceService"/>

   </partnerLinks>
...
```

Partner Link Type

- Characterizes the conversational relationship between two services by defining the "roles" played by each of the services in the conversation and specifying the portType provided by each service to receive messages within the context of the conversation
- Each role specifies exactly one WSDL portType.

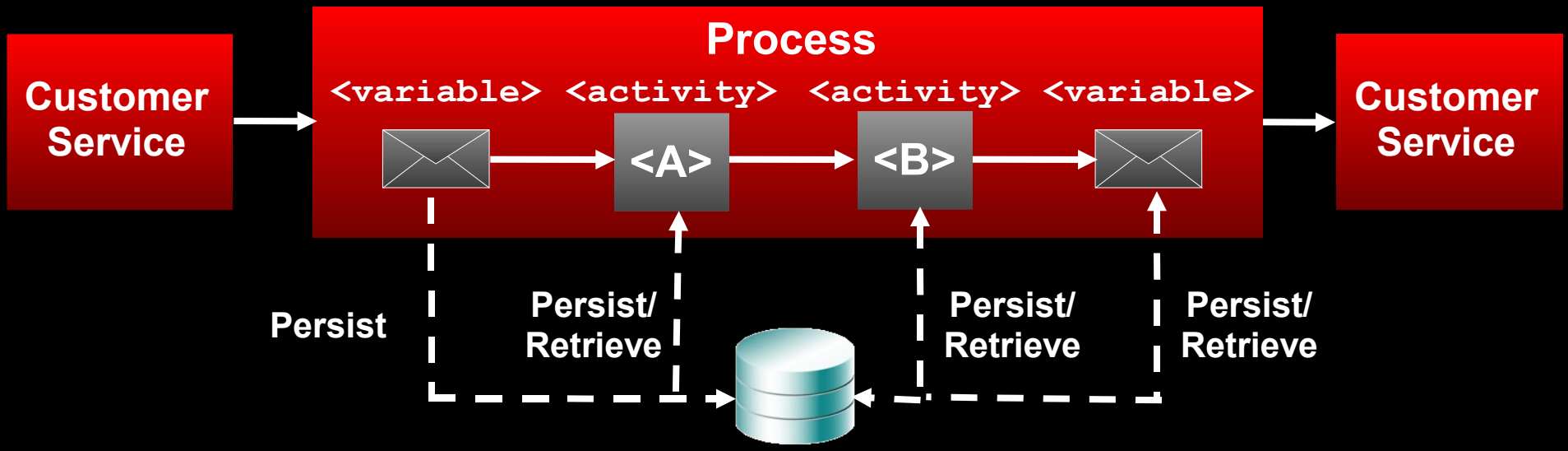
Partner Link Type

```
<partnerLinkType name="BuyerSellerLink"
    xmlns="http://schemas.xmlsoap.org/ws/2004/03/partner-link/">
    <role name="Buyer" portType="buy:BuyerPortType"/>
    <role name="Seller" portType="sell:SellerPortType"/>
</partnerLinkType>
```

BPEL Vocabulary:
Variables

Variables

- Messages sent and received from partners
  - Persisted for long running interactions
  - Defined in WSDL types and messages



Variables in BPEL

BPEL:

```
<variables>
 <variable name="PO" messageType="lns:POMessage"/>
 <variable name="Invoice" messageType="lns:InvMessage"/>
 <variable name="POFault" messageType="lns:orderFaultType"/>
</variables>
```

Purchase Process WSDL:

```
<message name="POMessage">
  <part name="customerInfo" type="sns:customerInfo"/>
  <part name="purchaseOrder" type="sns:purchaseOrder"/>
</message>
<message name="InvMessage">
  <part name="IVC" type="sns:Invoice"/>
</message>
<message name="orderFaultType">
 <part name="problemInfo" type="xsd:string"/>
</message>
```

How is Data Manipulation Done?

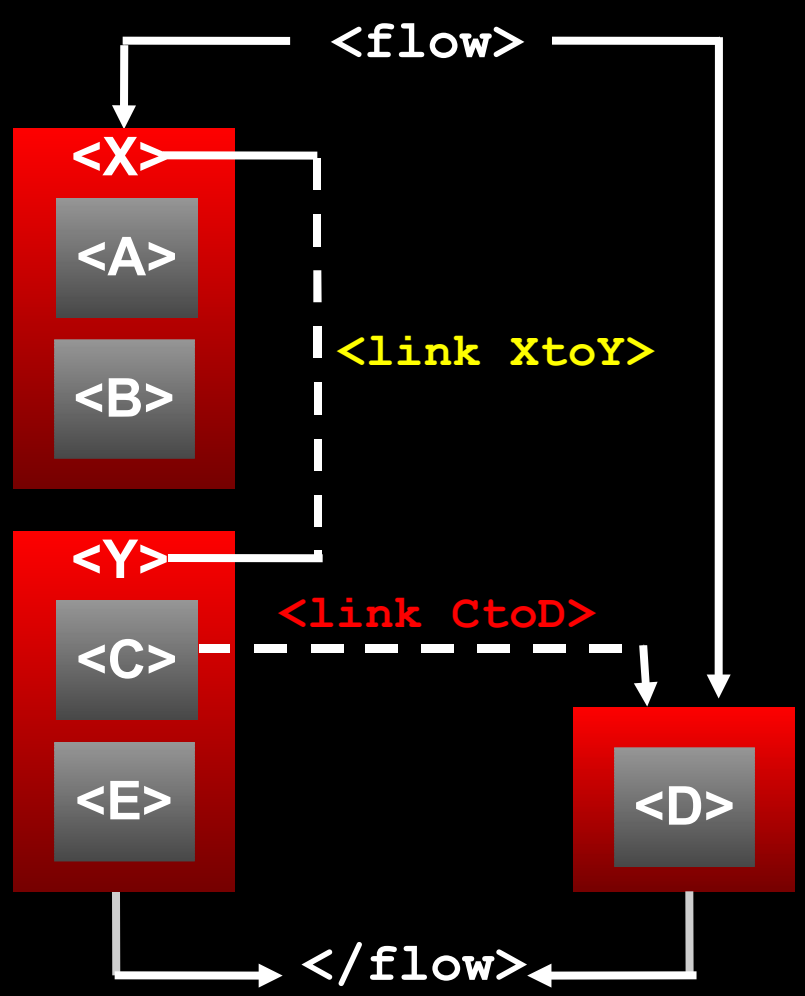
- Using `<assign>` and `<copy>`, data can be copied and manipulated between variables
- `<copy>` supports XPath queries to sub-select data

```
<assign>
 <copy>
  <from variable="PO" part="customerInfo"/>
  <to variable="creditRequest" part="customerInfo"/>
 </copy>
</assign>
```

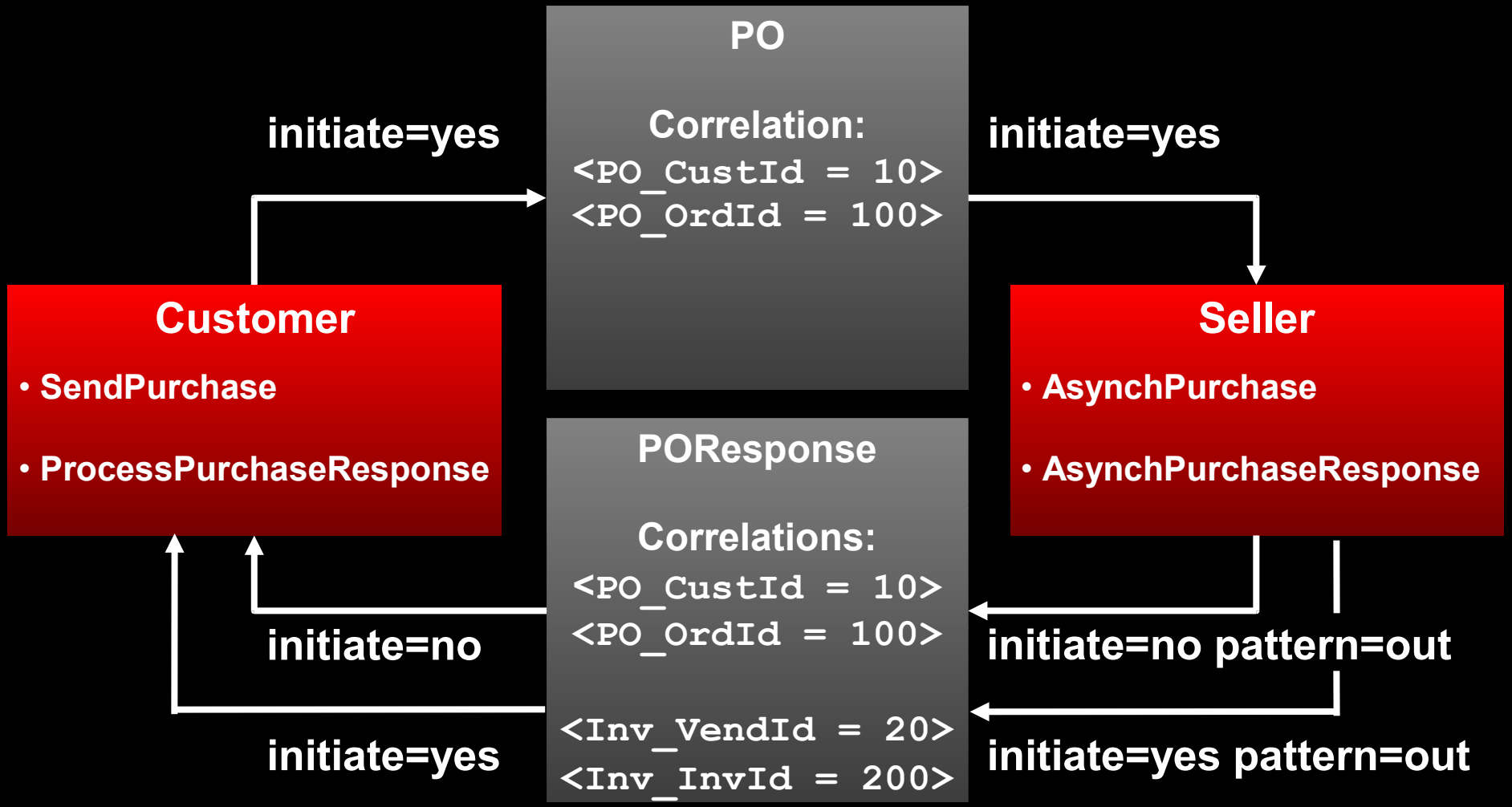
BPEL Vocabulary:
Flow, Links, Corelation

Links – Control Flow

```
<flow>
 <links>
  <link name="XtoY"/>
  <link name="CtoD"/>
 </links>
 <sequence name="X">
   <source linkName="XtoY"/>
   <invoke name="A" .../>
   <invoke name="B" .../>
 </sequence>
 <sequence name"Y">
   <target linkName="XtoY"/>
   <receive name="C"/>
     <source linkName="CtoD"/>
   </receive>
   <invoke name="E" .../>
 </sequence>
 <invoke partnerLink="D">
 <target linkName="CtoD"/>
 </invoke>
```



Correlation



Correlations in BPEL

```
<correlationSets>
 <correlationSet name="POCorr" properties="cor:custId cor:ordId"/>
 <correlationSet name="InvoiceCorr" properties="cor:vendId cor:invId"/>
</correlationSets> ...

<receive partnerLink="Customer" portType="SP:PurchaseOrderPT"
    operation="AsynchPurchase" variable="PO">
  <correlations>
    <correlation set="POCorr" initiate="yes">
  </correlations>
</receive> ...

<invoke partnerLink="Customer" portType="SP:CustomerPT"
  operation="ProcessPurchaseResponse" inputVariable="POResponse">
 <correlations>
  <correlation set="POCorr" initiate="no" pattern="out">
  <correlation set="InvoiceCorr" initiate="yes" pattern="out">
 </correlations>
</invoke>
```

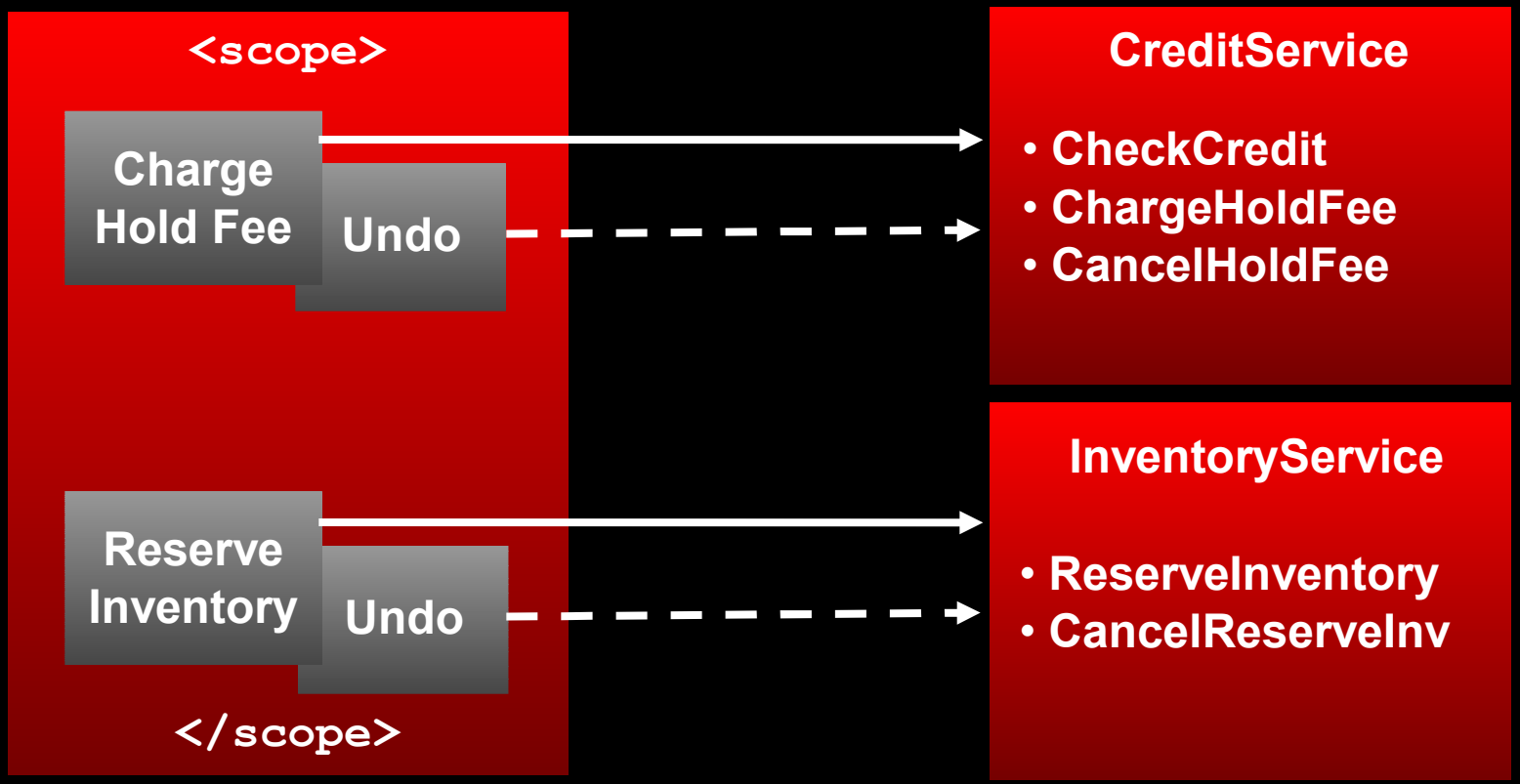
Scopes of BPEL

Scopes in BPEL

- Provide a shared context for subset of activities
- Can contain
  - fault handlers
  - event handlers,
  - compensation handler variables
  - correlation sets
- Can serialize concurrent access to variables

```
<scope
variableAccessSerializable="yes|no"
 ...>

<variables>
</variables>

<correlationSets>? ...
</correlationSets>

<faultHandlers>
</faultHandlers>

<compensationHandler>? ...
</compensationHandler>

<eventHandlers>
</eventHandlers>
(activities)*

</scope>
```

Long Running Transactions and Compensation (Undo)



Compensation Handlers in BPEL

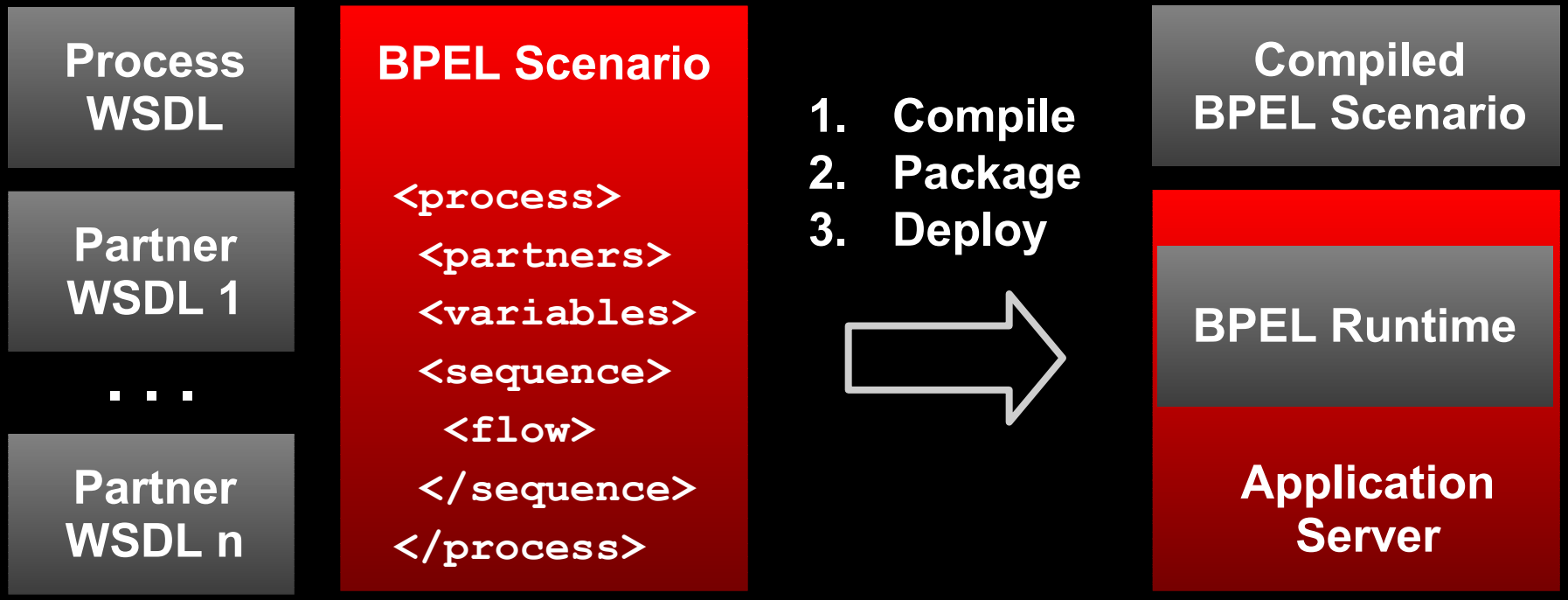
```
<scope>
 <compensationHandler>
  <invoke partnerLink="Seller" portType="SP:Purchasing"
          operation="CancelPurchase"
          inputVariable="getResponse"
          outputVariable="getConfirmation">
  <correlations>
    <correlation set="PurchaseOrder" pattern="out"/>
  </correlations>
  </invoke>
 </compensationHandler>
 <invoke partnerLink="Seller" portType="SP:Purchasing"
         operation="SyncPurchase"
         inputVariable="sendPO"
         outputVariable="getResponse">
 <correlations>
  <correlation set="PurchaseOrder" initiate="yes" pattern="out"/>
 </correlations>
 </invoke>
</scope>
```

Exception Handling in BPEL

- `<faultHandlers>` catch exception
  - Based on WSDL port defining fault
- `<faultHandlers>` can perform activities upon invocation

```
<faultHandlers>
    <catch faultName="lns:cannotCompleteOrder"
        faultVariable="POFault">
  <reply partnerLink="customer"
         portType="lns:purchaseOrderPT"
         operation="sendPurchaseOrder"
         variable="POFault"
         faultName="cannotCompleteOrder"/>
 </catch>
```

Just Show Me How to Do it!



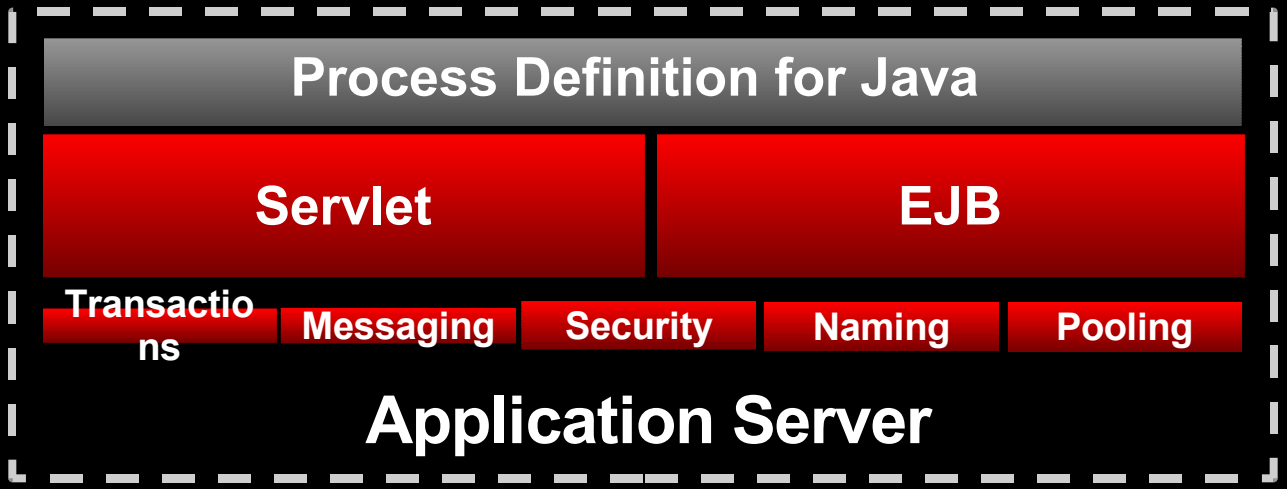
BPEL and Java Technology

BPEL and Java

- BPEL is thus comparable to general purpose programming language such as Java but it is not as powerful as Java
  - Exact order in which participating web services should be invoked
  - Conditional behavior, Loop, Variables, ...
- It is simpler and better suited for business process definition.
- Therefore BPEL is not a replacement but rather a supplement to modern languages such as Java.

What Happened to Java?

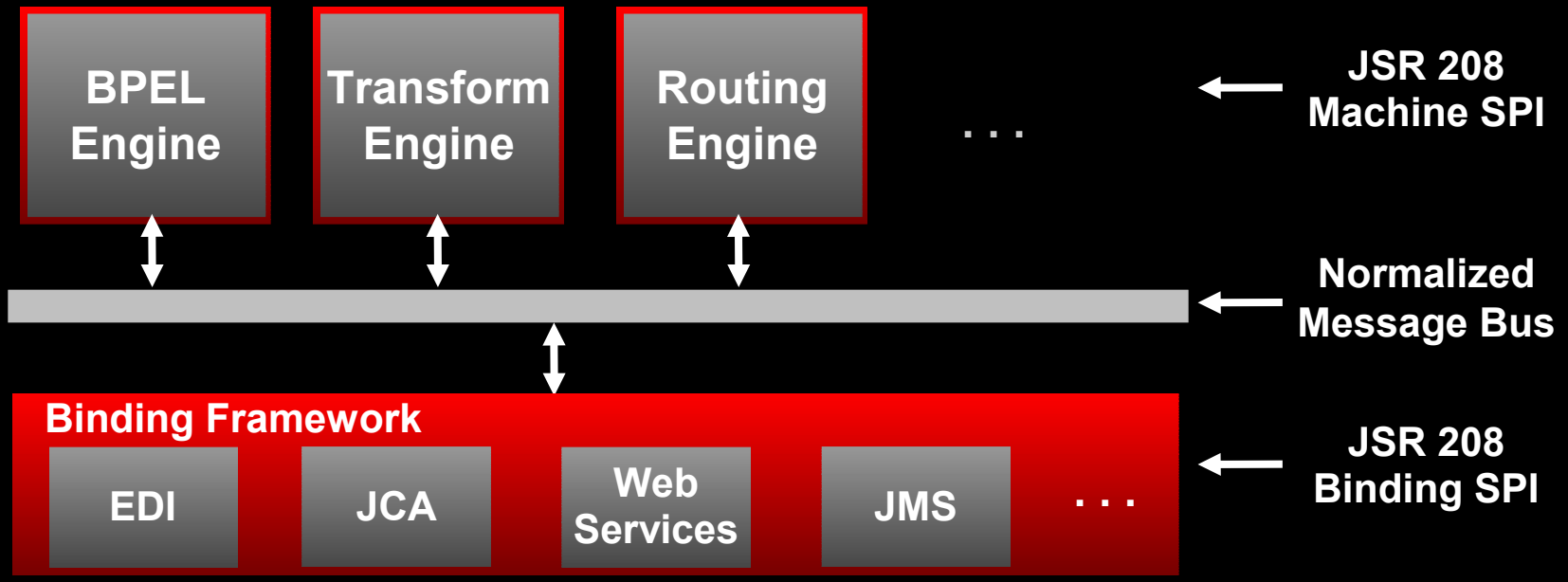
- JSR 207 - Process Definition Language for Java
- Make business processes natural for Java programmers



Based on JSR 207 Session at JavaOne 2003

What Happened to J2EE?

- JSR 208 – Java Business Integration
- Make business processes a first class citizen in J2EE containers



Based on JSR 208 Session at JavaOne 2003

Tooling Requirements

- ☒ IDE – build your Web services
- ☒ WSDL authoring – model your interfaces
- ☒ Schema authoring – model your messages
- ☒ Process modeling – model your orchestration
- ☒ Packaging and deployment
- ☒ Debugging
- ☒ Monitoring
- ☒ Analyzing

Issues with BPEL

Remember BPEL Does Not Solve "World Hunger"

- ☒ No data transformation
- ☒ No data translation (EDI, binary formats …)
- ☒ No human workflow
- ☒ No trading partner agreements
- ☒ Silent on existing business protocols (ebXML, RosettaNet …)
- ☒ Silent on non Web service interactions (e.g. java to java)
- ☒ . . .

But Remember: People Are Trying to Solve “World Hunger”

- ☒ W3C: WS-Choreography
- ☒ Spec: WS-Transaction
- ☒ Spec: WS-Coordination
- ☒ Spec: WS-Composite Application Framework
- ☒ OASIS: WS-Reliability
- ☒ Spec: ReliableMessaging
- ☒ Spec: WS-Addressing
- ☒ OASIS: WS-Security
- ☒ …

Parting Thoughts

- Business process portability?
  - Java/J2EE is portable across application servers
  - BPEL is portable independent of Java
- Programming language in XML?
  - Vendors, big and small, are busy building design times and modelers…
- Is BPEL in 2003/2004 J2EE in 1998?
  - Much missing but compelling foundation