SOA-BPEL

tutorialspoint

SIMPLY EASY LEARNING

About the Tutorial

SOA or the Service Oriented Architecture is an architectural approach, which makes use of technology to present business processes as reusable services.

Business Process Engineering Language is a technology used to build programs in SOA architecture.

Audience

This tutorial is designed for users who are keen on learning the basics of the BPEL Process.

Prerequisites

We assume that you have the Oracle BPEL Service Manager installed in your system. This will help in better understanding of the tutorial.

Copyright & Disclaimer

Table of Contents

1. BPEL – Introduction

SOA or the Service Oriented Architecture is an architectural approach, which makes use of technology to present business processes as reusable services.

- It is focused on the business and enables process transformation to new levels of integration, visualization, monitoring, and optimization.
- It is not a technology, it is a concept and a strategy for using technologies to build business automation solutions.

We will now see what BPEL is and how it helps in the SOA.

What is BPEL?

Business Process Engineering Language is a technology used to build programs in SOA architecture.

Adding a BPEL Process Service Component

Follow these steps to add a BPEL Process Service Component:

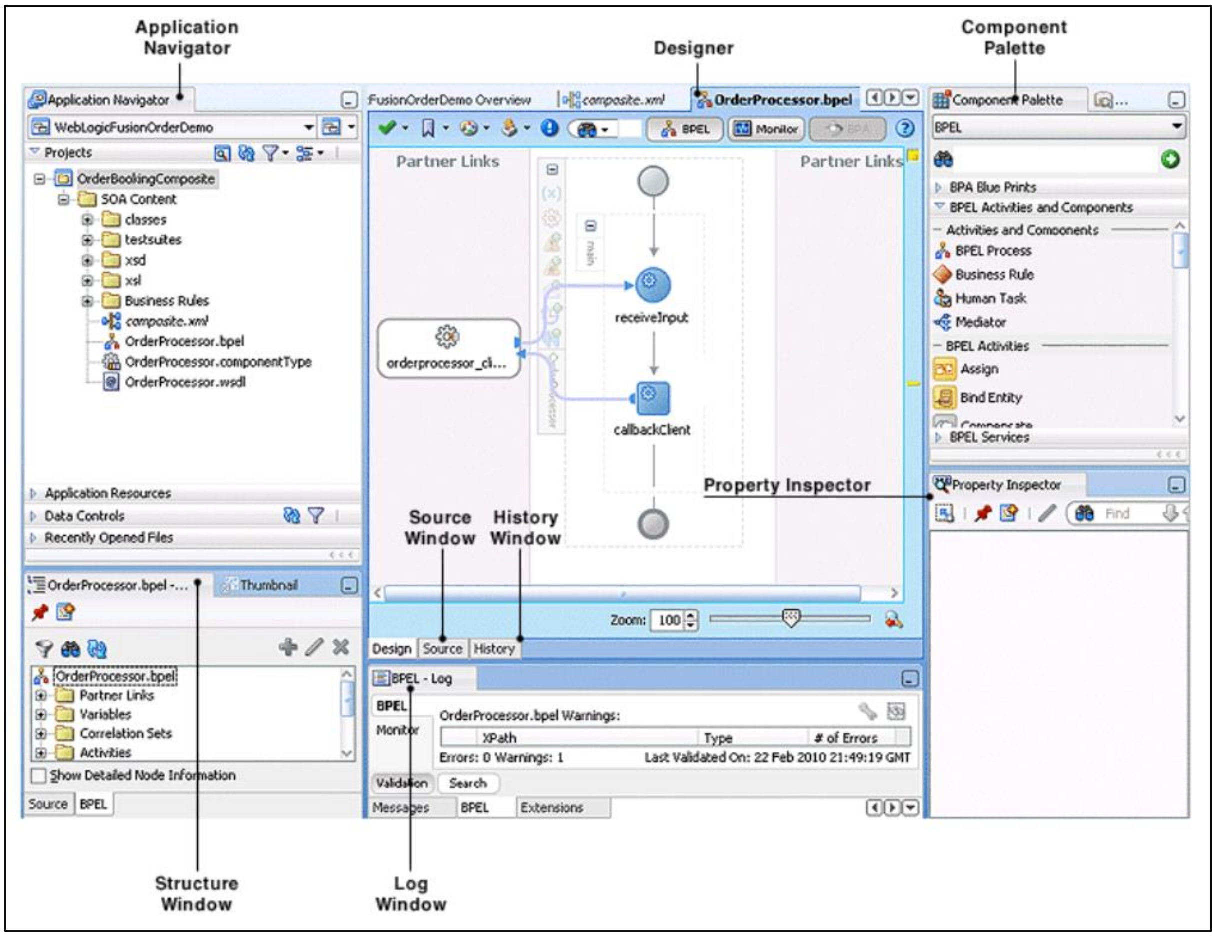
- From the Application Navigator, select File > New > Applications > SOA Application.
- This starts the Create SOA Application wizard.
- In the Application Name dialog, enter an application name in the Application Name field.
- In the Directory field, enter a directory path in which to create the SOA composite application and project.
- Click Next.
- In the Project Name dialog, enter a name in the Project Name field.
- Click Next.
- In the Project SOA Settings dialog, select Composite with the BPEL Process.
- Click Finish.

Files in the BPEL Composite

The BPEL composite contains the following files:

- **composite.xml** - This file describes the entire composite assembly of services, service components, references, and wires.

- **.bpel** – This file contains the set of activities added to the process.
- **.componentType** – This file describes the services and references for the BPEL process service component.
- **.wsdl** – This file defines the input and output messages for this BPEL process flow, the supported client interface and operations, and other features.



OrderProcessor.bpel Warnings:

Errors: 0 Warnings: 1 Last Validated On: 22 Feb 2010 21:49:19 GMT

Concepts used in the BPL process

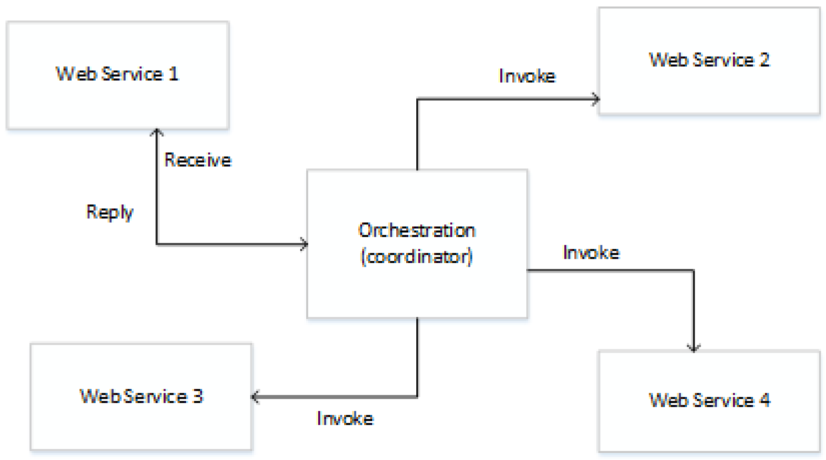
In this section, we will learn the different concepts involved in the BPL process.

Orchestration

- Usually used in private business processes.
- A central process (which can be another Web service) takes control of the involved Web services.
- Coordinates the execution of different operations on the web services involved in the operation.

- The involved Web services do not "know" (and do not need to know) that they are involved in a composition process and that they are taking part in a higher-level business process.

- Only the central coordinator of the orchestration is aware of this goal, so the orchestration is centralized with explicit definitions of operations and the order of invocation of Web services.



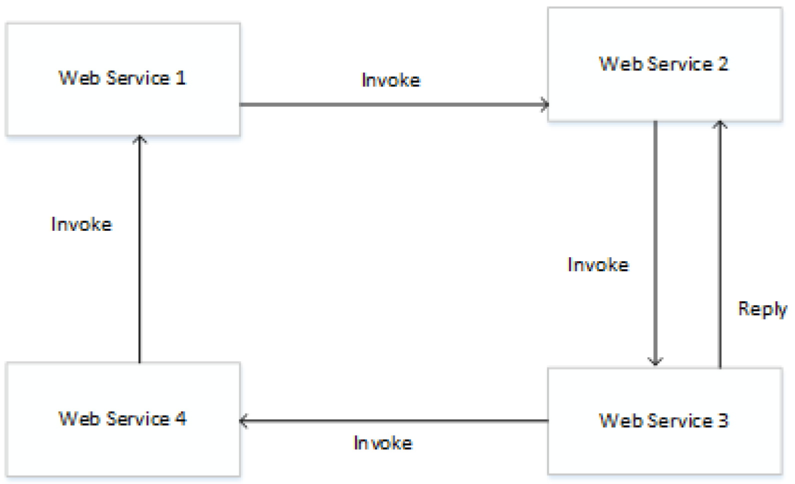
Choreography

- Does not rely on a central coordinator.

- Each Web service involved in the choreography knows exactly when to execute its operations and with whom to interact.

- Collaborative effort focusing on the exchange of messages in public business processes.

- All participants in the choreography need to be aware of the business process, operations to execute, messages to exchange, and the timing of message exchanges.



2. BPEL – Activities

In this chapter, we will learn about the different activities that make up the building blocks The building blocks of a BPEL process service component.

Oracle BPEL Designer includes a set of activities that you drag into a BPEL process service component and double-click an activity to define its attributes and property values.

Assign Activity

An assign activity enables to manipulate data, such as copying the contents of one variable to another.

Invoke Activity

An invoke activity enables you to invoke a service (identified by its partner link) and specify an operation for this service to perform.

Receive Activity

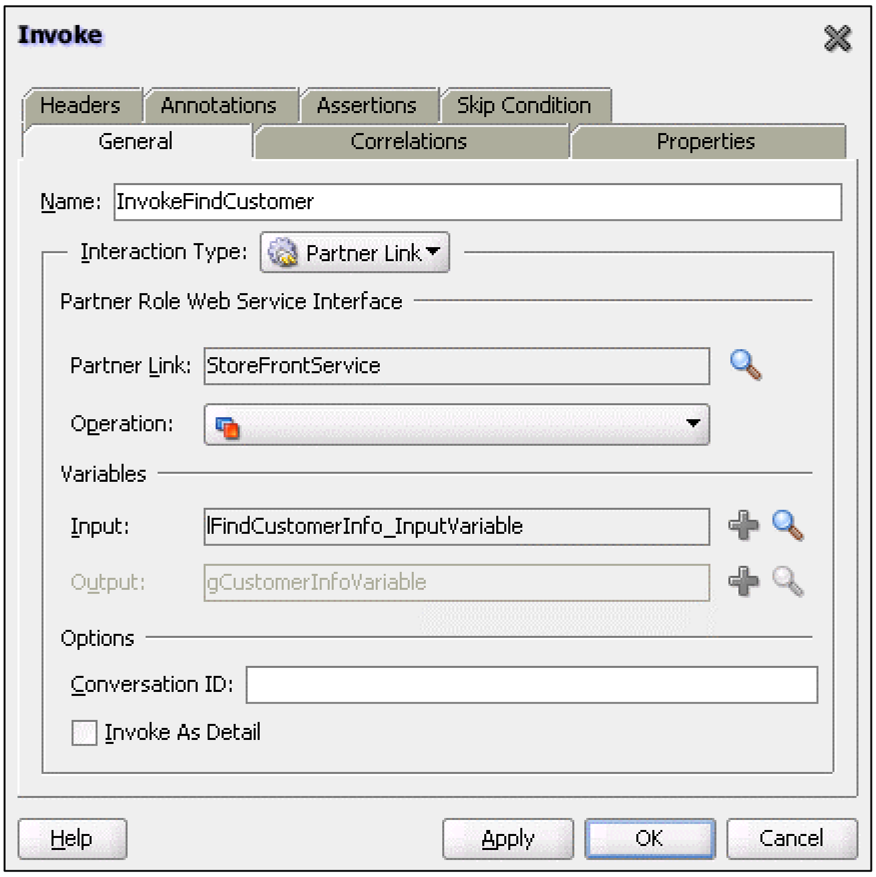
A receive activity waits for an asynchronous callback response message from a service.

Let us learn more about the Invoke activity in our subsequent section.

Invoke Activity

The invoke activity enables to specify an operation that is to be invoked for the service (identified by its partner link). The operation can be one-way or request-response on a port provided by the service. Variables can be automatically created in an invoke activity. An invoke activity invokes a synchronous service or initiates an asynchronous web service.

The invoke activity opens a port in the process to send and receive data. This port can be further used to submit required data and receive a response. For synchronous callbacks, only one port is needed for both the send and the receive functions.



Name: InvokeFindCustomer

Interaction Type: Partner Link

Partner Role Web Service Interface

Partner Link: StoreFrontService

Operation:

Variables

Input: lFindCustomerInfo_InputVariable

Output: gCustomerInfoVariable

Options

Conversation ID:

☐ Invoke As Detail

3. BPEL – Partner Link in BPEL Process

Partner Links are defined as communication exchanges between all the parties with which the BPEL Process interacts.

They are the references to the actual implementations, through which the BPEL process interacts with the external world.

Invoked Partner Links

These are links to services that are invoked by the BPEL process.

Client Partner Links

These are links to services that can invoke a BPEL process.

Partner Link Properties

The Partner Link Property Editor allows you to establish partner links for your BPEL processes. With the Partner Link Property Editor, you can specify the following:

- **Name**: Specifies the name of the Invoke element.
- **WSDL File**: Indicates the WSDL file associated with the Partner Link.
- **Partner Link Type**: Indicates the Partner Link type defined in the WSDL.
- **My Role**: Indicates the role of the business process itself.
- **Partner Pole**: Indicates the role of the partner.
- **Documentation**: Accessed on the Properties window.

Partner Links are defined in the .bpel file.

A BPEL can interact with the services in the following three ways:

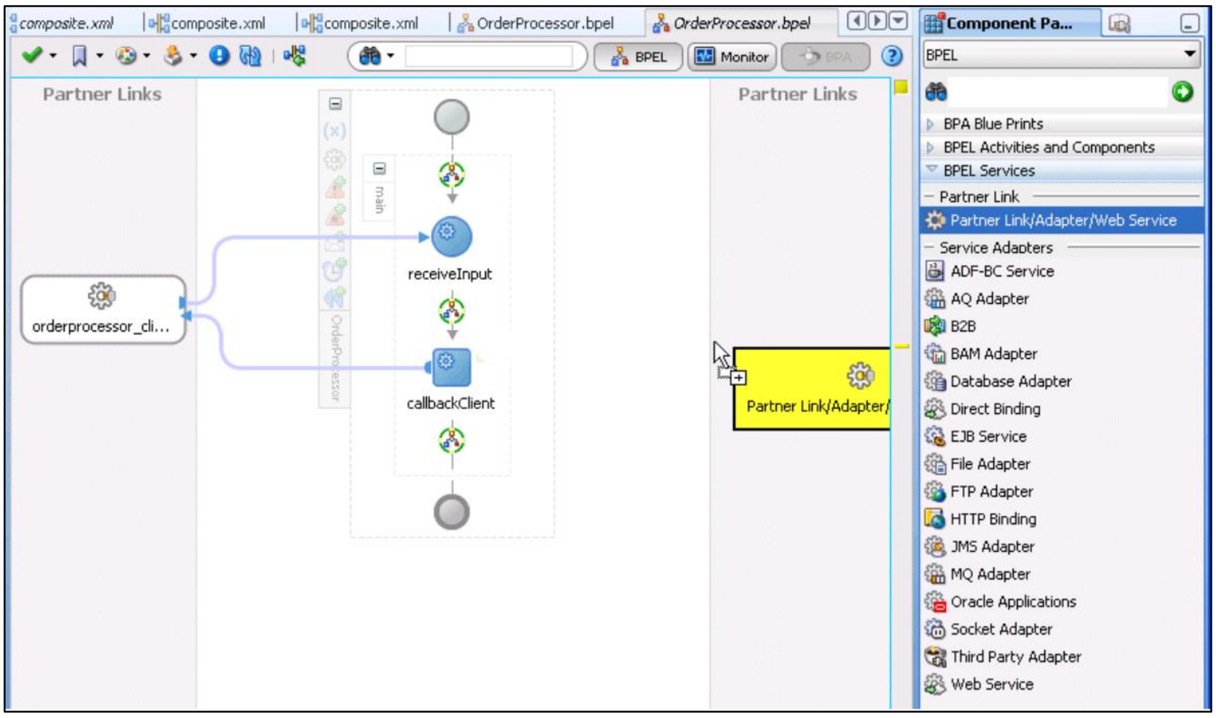
- Services that invoke a BPEL process
- Services that are invoked by the BPEL process
- Services that act both ways

4. BPEL – Creating a Partner Link

In this chapter, we will learn how to create a partner link.

Follow the steps shown below to create a partner link:

- In the SOA Composite Editor, double-click the BPEL process service component.
- Upon clicking the service component, the Oracle BPEL Designer is displayed.
- In the Component Palette, expand BPEL Services.



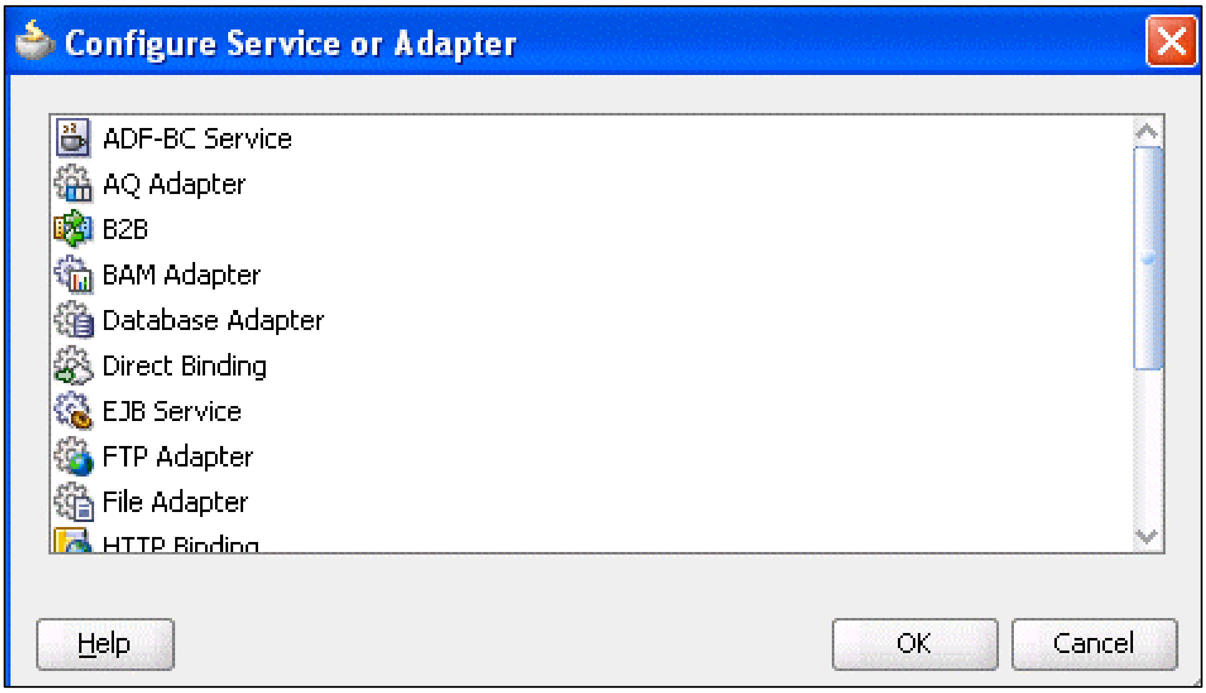
- Drag a Partner Link into the appropriate Partner Links swimlane.
- Complete the fields for this dialog as mentioned above in the Partner Link Properties.

5. BPEL – Adapters

Adapters enable to integrate the BPEL process service component with access to file systems, FTP servers, database tables, database queues, sockets, Java Message Services (JMS), MQ, and Oracle E-Business Suite. This wizard enables to configure the types of adapters shown in below figure for use with the BPEL process service component:

Adapter Types

The following image shows the different adapter types:



ADF-BC Service
AQ Adapter
B2B
BAM Adapter
Database Adapter
Direct Binding
EJB Service
FTP Adapter
File Adapter
HTTP Binding

Advanced Queuing (AQ)

For interaction with a queue. AQ provides a flexible mechanism for bidirectional, asynchronous communication between participating applications.

Oracle Business Activity Monitoring (BAM)

For publishing data to data objects in an Oracle BAM Server.

Database

For interaction with Oracle and non-Oracle databases through JDBC and Oracle Business Intelligence (which is a special data source type).

FTP and File

For file exchange (read and write) on local file systems and remote file systems (through use of the file transfer protocol (FTP)).

Java Messaging Service (JMS)

For interaction with JMS. The JMS architecture uses a one client interface to many messaging servers architecture.

Message Queue (MQ)

For message exchange with WebSphere MQ queuing systems.

Oracle Applications

For interaction with Oracle Application's set of integrated business applications.

Oracle B2B

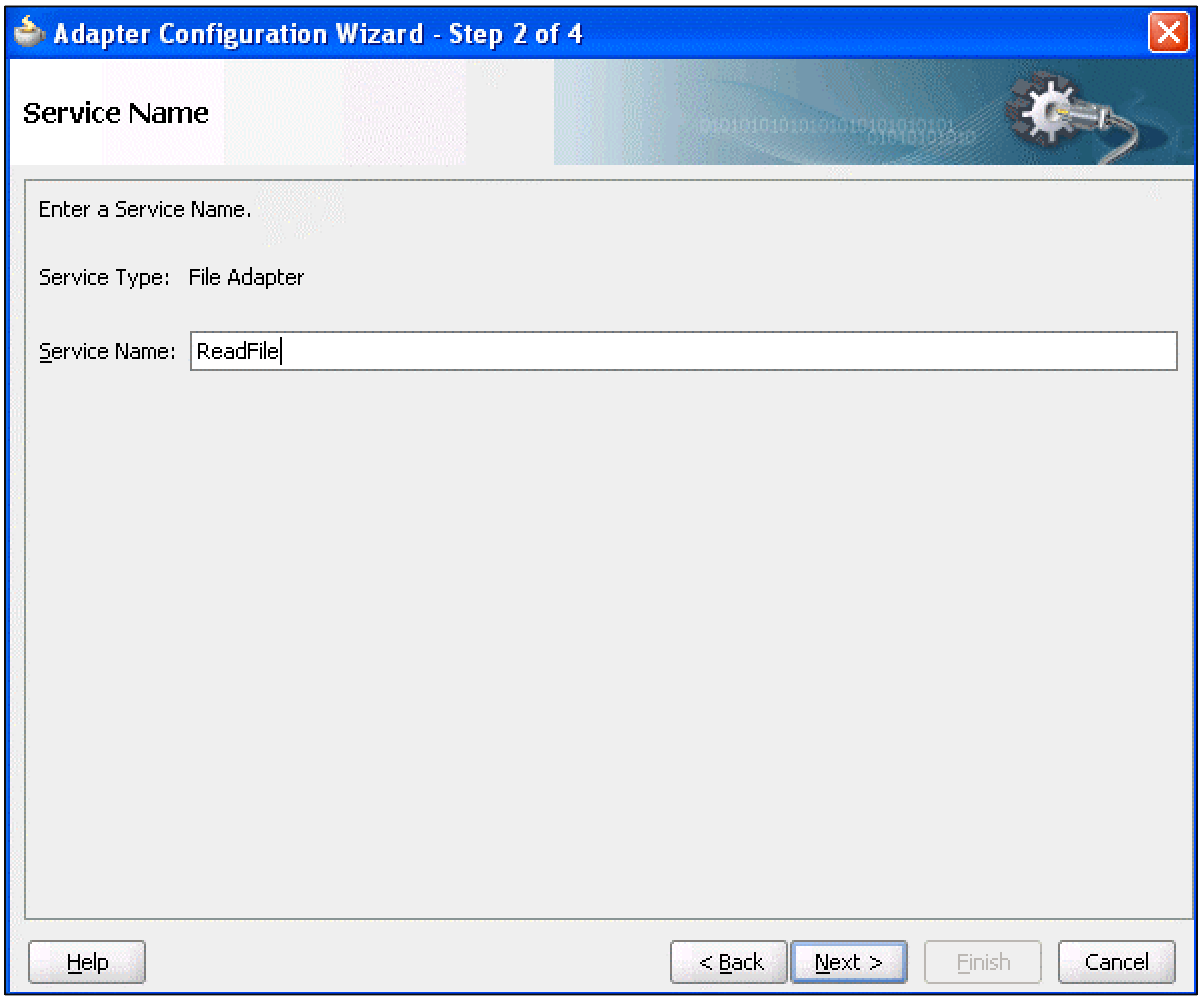
For browsing B2B metadata in the metadata service (MDS) repository and selecting document definitions.

Sockets

For modeling standard or nonstandard protocols for communication over TCP/IP sockets.

Adapter Service Name

The Service Name window prompts to enter a name, when the adapter type is selected from the pallet. For this example, **File Adapter** was selected. When the wizard completes, a WSDL file by this service name appears in the Application Navigator for the BPEL process service component (for this example, named **ReadFile.wsdl**). The service name must be unique within the project. This configuration file includes the adapter configuration settings specified with this wizard. Other configuration files (such as header files and files specific to the adapter) are also created. These files are displayed in the Application Navigator.



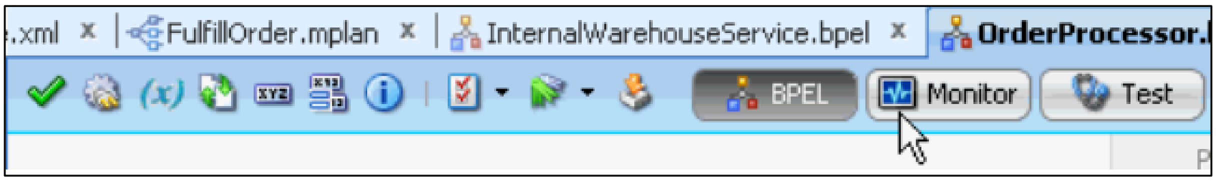
Enter a Service Name.

Service Type: File Adapter

Service Name: ReadFile

6. BPEL – Process Monitors

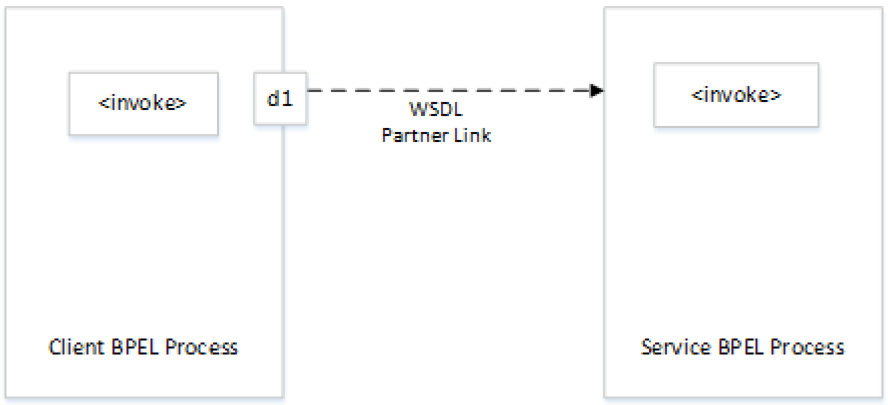
BPEL process monitors in Oracle BPEL Designer can be configured by selecting Monitor at the top of Oracle BPEL Designer.



At this stage, the Oracle BAM Adapter needs to be configured.

7. BPEL – One-Way Messages

The Client BPEL Process sends a message to Service BPEL Process and the Service BPEL Process is not required to reply as shown in the figure below:

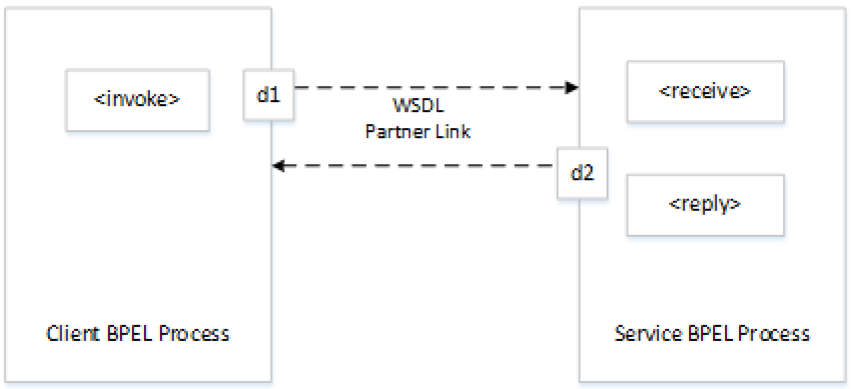


- The Client BPEL Process needs a valid partner link and an invoke activity.
- The Service BPEL Process needs a receive activity.
- As with all partner activities, the Web Services Description Language (WSDL) file defines the interaction. The WSDL file is as shown below.

```
<wsdl:portType name="BPELProcess">
      <wsdl:operation name="process">
         <wsdl:input  message="client:BPELProcessRequestMessage" />
         <wsdl:output message="client:BPELProcessResponseMessage"/>
      </wsdl:operation>
</wsdl:portType>
```

8. BPEL – Synchronous Interactions

The Client BPEL Process sends a request to the Service BPEL Process (d1 in the below figure), and receives an immediate reply (d2 in the below figure). For example, a user requests a subscription to an online application form for admission to a college and immediately receives email confirmation that their request has been accepted.



- The Client BPEL Process needs an invoke activity. The port on the client side sends the request and receives the reply.

- The Service BPEL Process needs a receive activity to accept the incoming request, and a reply activity to return either the requested information or an error message (a fault; f1 in the below figure) defined in the WSDL.

- As with all partner activities, the Web Services Description Language (WSDL) file defines the interaction. The WSDL file is as shown below.

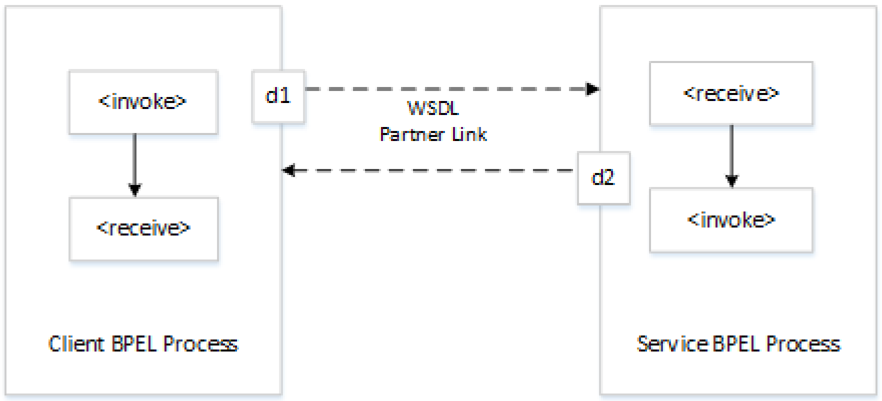
  WSDL File:

```
<wsdl:portType name="BPELProcess">

      <wsdl:operation name="process">

         <wsdl:input  message="client:BPELProcessRequestMessage" />

         <wsdl:output message="client:BPELProcessResponseMessage"/>

      </wsdl:operation>

</wsdl:portType>
```

9. BPEL – Asynchronous Interactions

The Client BPEL Process sends a request to the Service BPEL Process (d1 in the figure given below), and waits until the service replies (d2 in the figure given below).

For example, a user requests a subscription to an online application form for admission to a college and the request cannot be confirmed unless it is accepted at the admission office.



- The Client BPEL Process needs an invoke activity to send the request and a receive activity to receive the reply.

- The Service BPEL Process needs a receive activity to accept the incoming request and an invoke activity to return either the requested information or a fault.

  Note: The difference between responding from a synchronous and asynchronous BPEL process is that the synchronous service uses a reply activity to respond to the client and an asynchronous service uses an invoke activity.

- As with all partner activities, the Web Services Description Language (WSDL) file defines the interaction. The WSDL file is as shown below.

  WSDL File:

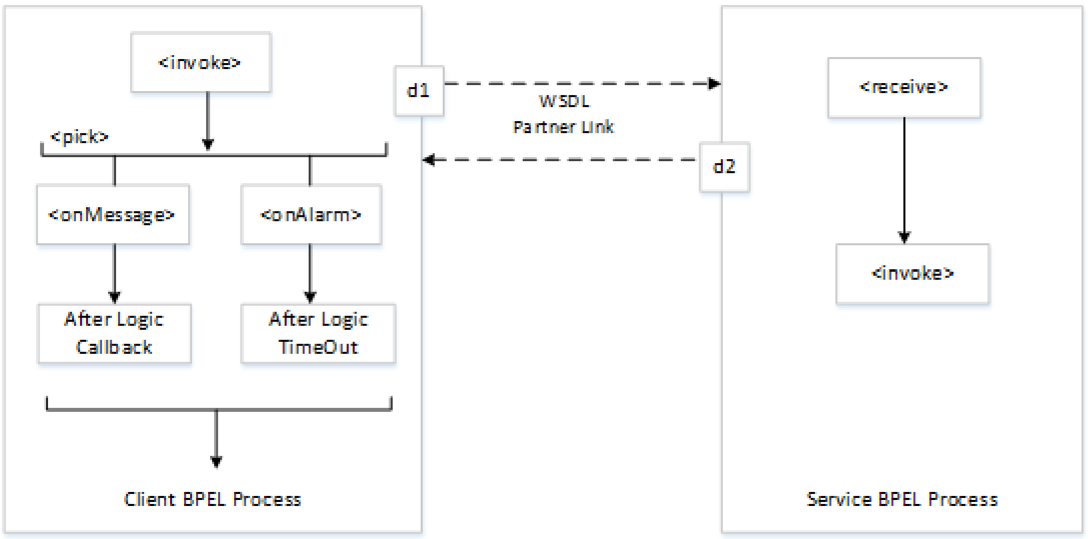
```
<wsdl:portType name="BPELProcess">
      <wsdl:operation name="process">
         <wsdl:input message="client:BPELProcessRequestMessage"/>
      </wsdl:operation>
</wsdl:portType>


<wsdl:portType name="BPELProcessCallback">
      <wsdl:operation name="processResponse">
         <wsdl:input message="client:BPELProcessResponseMessage"/>
      </wsdl:operation>
```

```
</wsdl:portType>
```

10. BPEL – Asynchronous Interactions with a Timeout

The Client BPEL Process sends a request to the Service BPEL Process (d1 in the below figure), and waits until the service replies or until a certain time limit is reached, whichever comes first. (d2 in the below figure).

For example, a user requests a subscription to an online application form for admission to a college and the request is cancelled if the user does not receive a confirmation reply within a specified amount of time.



The Client BPEL Process needs an invoke activity to send the request and a pick activity with two branches - an **onMessage** branch and an **onAlarm** branch. If the reply comes after the time limit has expired, the message goes to the dead letter queue.

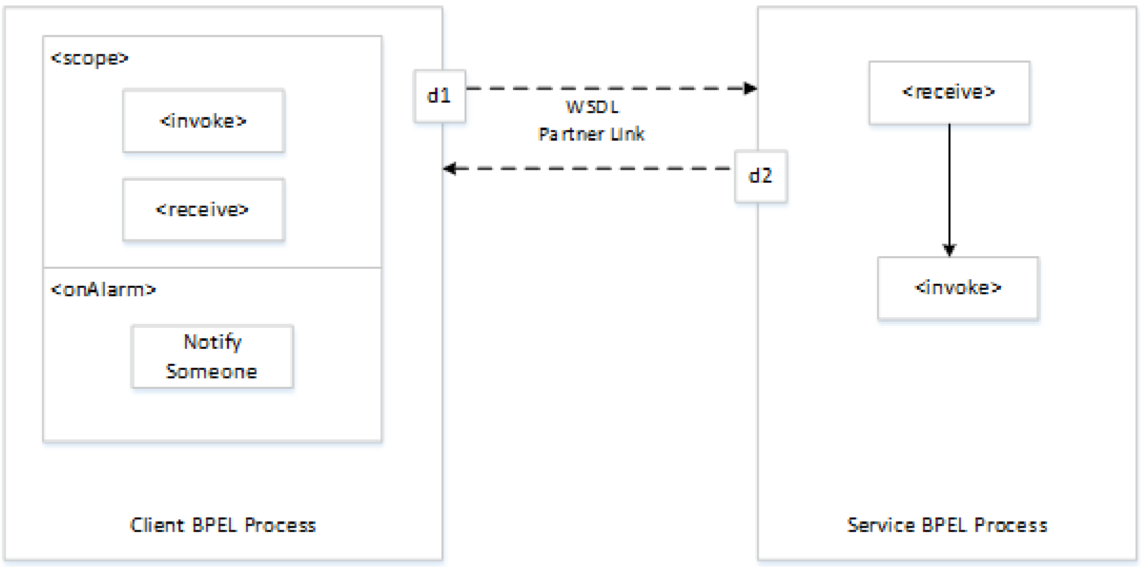
The Service BPEL Process needs a receive activity to accept the incoming request and an invoke activity to return either the requested information or a fault.

As with all partner activities, the Web Services Description Language (WSDL) file defines the interaction.

11. BPEL – Asynchronous Interactions with a Notification Timer

In this chapter, we will learn about asynchronous interactions with a notification timer. Consider the following points related to the asynchronous interactions:

- The Client BPEL Process sends a request to the Service BPEL Process and waits for a reply, although a notification is sent after a timer expires.
- The Client BPEL Process continues to wait for the reply from the Service BPEL Process even after the timer has expired.
- The Client BPEL Process needs a scope activity containing an invoke activity to send the request, and a receive activity to accept the reply. The **onAlarm** handler of the scope activity has a time limit and instructions on what to do when the timer expires.
- For example, wait 60 seconds, then send a warning indicating that the process is taking longer than expected.
- The Service BPEL Process needs a receive activity to accept the incoming request and an invoke activity to return either the requested information or a fault.
- As with all partner activities, the Web Services Description Language (WSDL) file defines the interaction.



12. BPEL – One Request, Multiple Responses

In this chapter, we will learn about the concept of One Request and Multiple Responses.

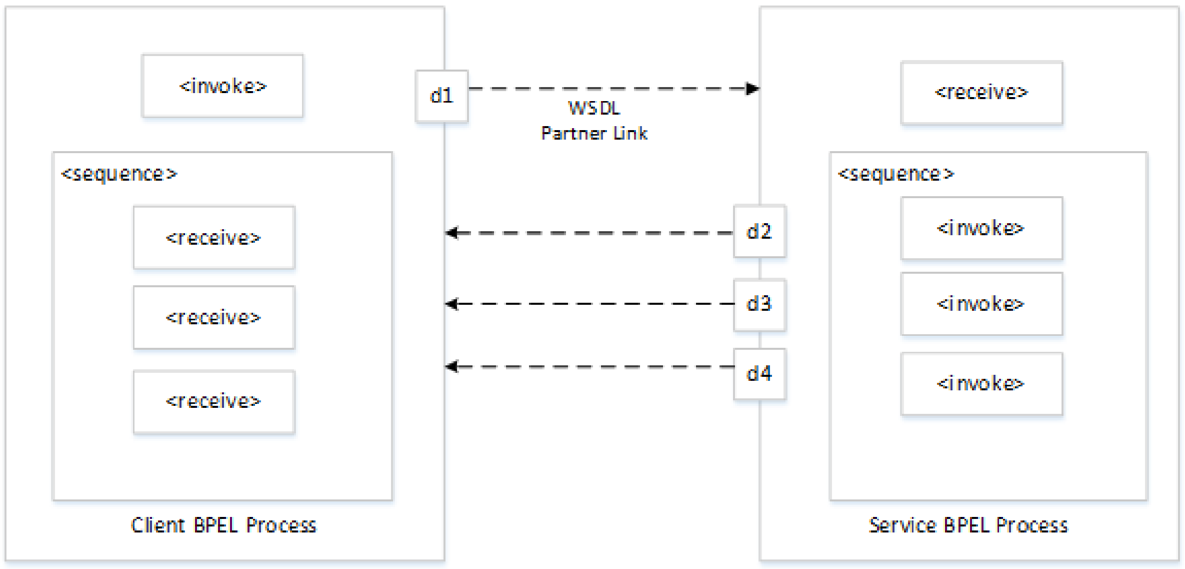
- The Client BPEL Process sends a single request to the Service BPEL Process and receives multiple responses in return.

  For example, the request can be to order a product online, and the first response can be the estimated delivery time, the second response a payment confirmation, and the third response a notification that the product has shipped. In this example, the number and types of responses are expected.

- The Client BPEL Process needs an invoke activity to send the request, and a sequence activity with three receive activities.

- The Service BPEL Process needs a receive activity to accept the message from the client, and a sequence attribute with three invoke activities, one for each reply.

- As with all partner activities, the Web Services Description Language (WSDL) file defines the interaction.



13. BPEL – One Request, One of Two Possible Responses

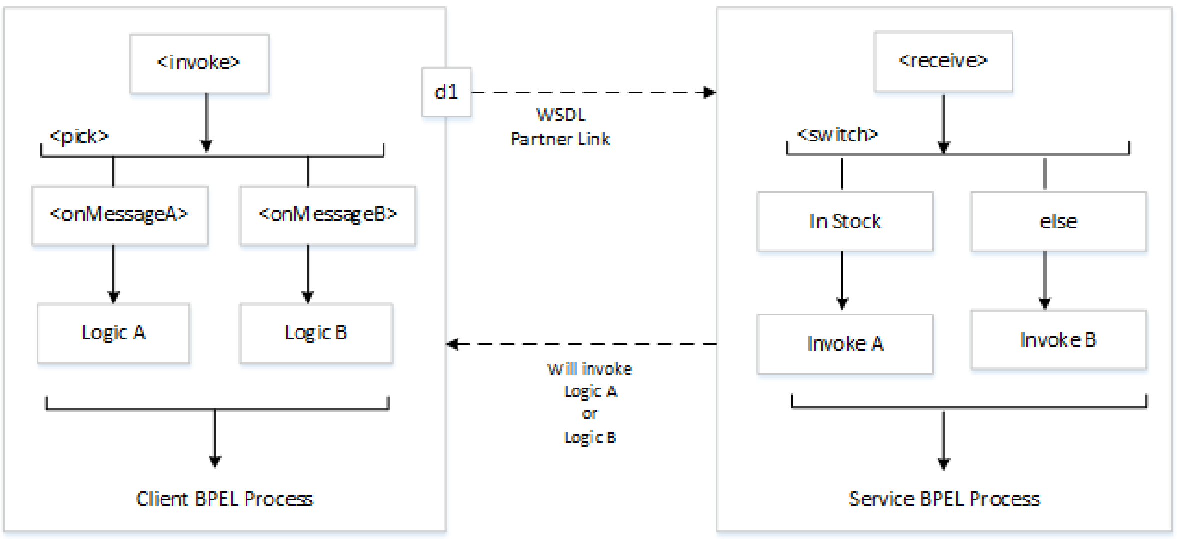
In this chapter, we will learn about the concept of one request and one of two possible responses.

- The Client BPEL Process sends a single request to the Service BPEL Process and receives one of two possible responses.

  For example, the request can be to order a product online, and the first response can be either an in-stock message, or an out-of-stock message.

- The Client BPEL Process needs the following:
  - An invoke activity to send the request.
  - A pick activity with two branches: one onMessage for the in-stock response and instructions on what to do if an in-stock message is received.
  - A second onMessage for the out-of-stock response and instructions on what to do if an out-of-stock message is received.
- The Service BPEL Process needs a receive activity to accept the message from the client, and a switch activity with two branches, one with an invoke activity sending the in-stock message if the item is available, and a second branch with an invoke activity sending the out-of-stock message if the item is not available.

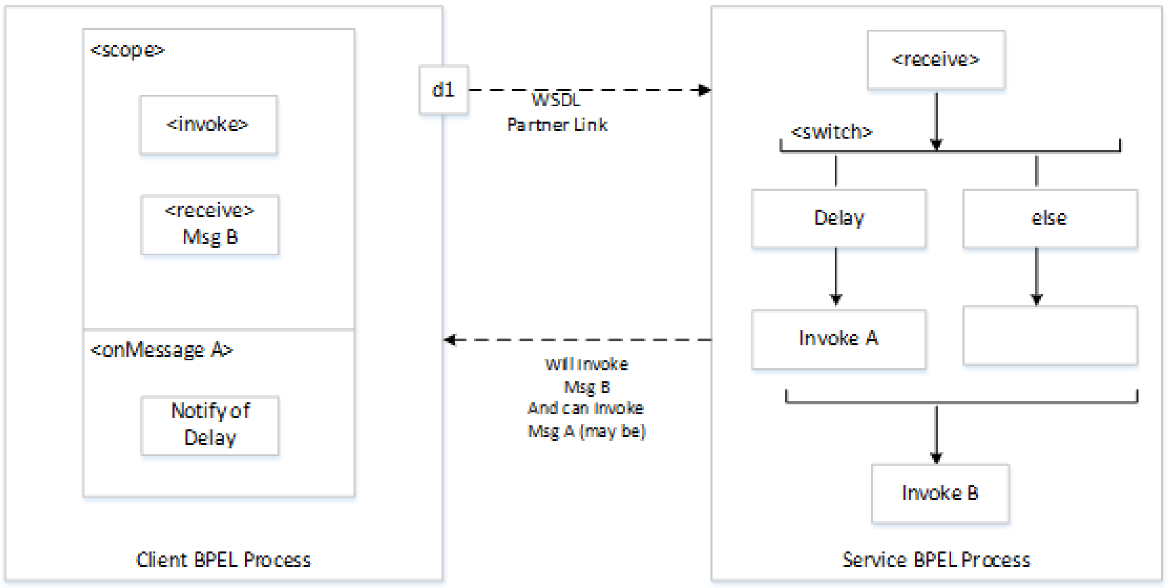
As with all partner activities, the Web Services Description Language (WSDL) file defines the interaction.



14. BPEL – One Request, a Mandatory Response, and an Optional Response

In this chapter, we will understand the concept of one request, a mandatory response, and an optional response.

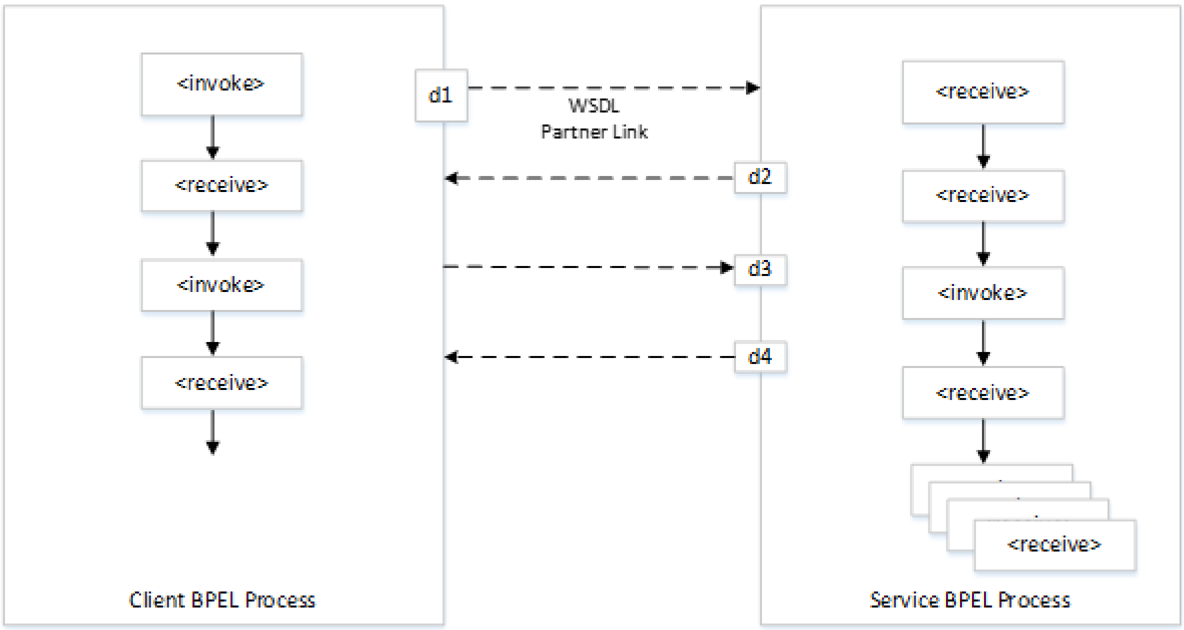
- The Client BPEL Service sends a single request to the Service BPEL Process and receives one or two responses.
- Here, the request is to order a product online. If the product is delayed, the service sends a message letting the customer know. In any case, the service always sends a notification when the item ships.
- The Client BPEL Service needs a scope activity containing the invoke activity to send the request, and a receive activity to accept the mandatory reply. For the optional message, the **onMessage** handler of the scope activity is set along with the instructions on what to do if the optional message is received (for example, notify you that the product has been delayed). The Client BPEL Process waits to receive the mandatory reply. If the mandatory reply is received first, the BPEL Process continues without waiting for the optional reply.
- The Service BPEL Process needs a scope activity containing the receive activity and an invoke activity to send the mandatory shipping message, and the scope's **onAlarm** handler to send the optional delayed message if a timer expires (for example, send the delayed message if the item is not shipped in 24 hours).
- As with all partner activities, the Web Services Description Language (WSDL) file defines the interaction.



15. BPEL – Partial Processing

Now, we will learn the concept of partial processing in BPEL.

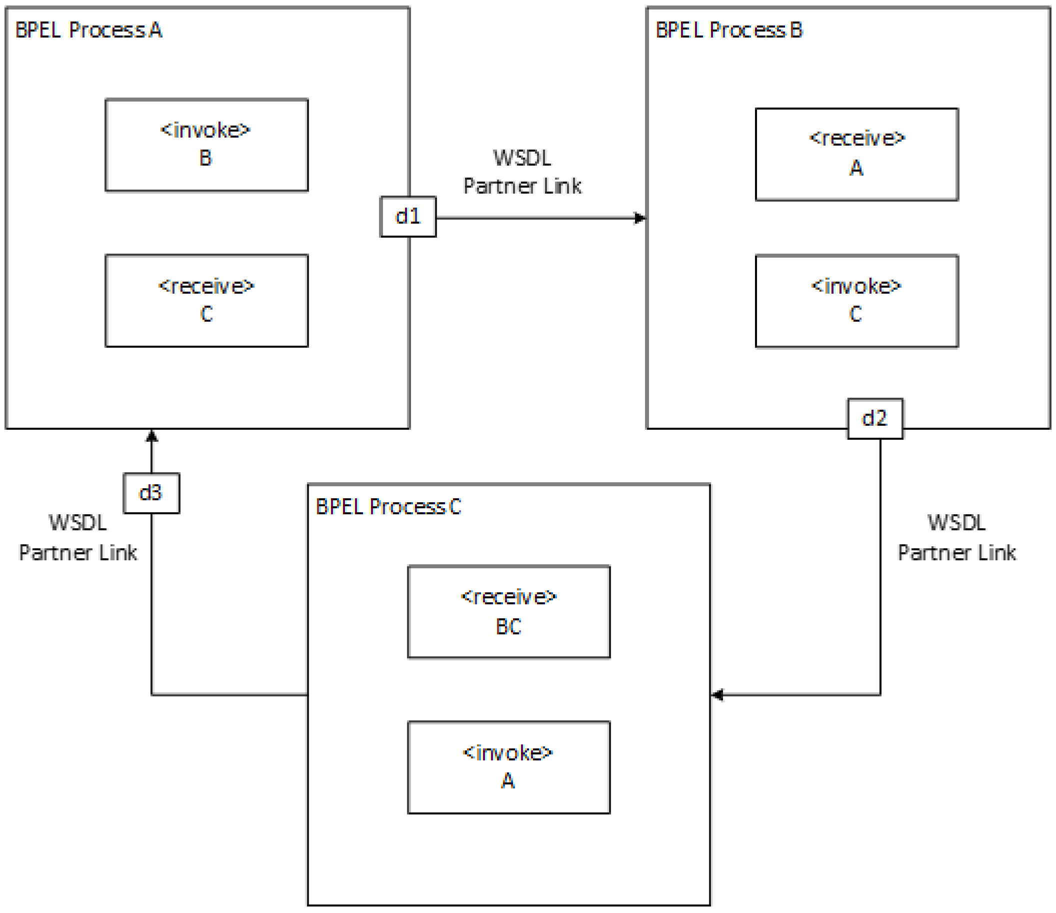
- The Client BPEL Process sends a request to the Service BPEL Process and receives an immediate response, but processing continues on the service side.
- This pattern can also include multiple shot callbacks, followed by longer-term processing.
- For example, the client sends a request to purchase a vacation package, and the service sends an immediate reply confirming the purchase, then continues to book the hotel, the flight, the rental car, and so on.
- The Client BPEL Process needs an invoke activity for each request and a receive activity for each reply for asynchronous transactions, or just an invoke activity for each synchronous transaction.
- The Service BPEL Process needs a receive activity for each request from the client, and an invoke activity for each response. Once the responses are finished, the Service BPEL Process as the service can continue with its processing, using the information gathered in the transaction to perform the necessary tasks without any further input from the client.
- As with all partner activities, the Web Services Description Language (WSDL) file defines the interaction.



16. BPEL – Multiple Application Interactions

We will learn about the multiple application interactions with BPEL in this chapter.

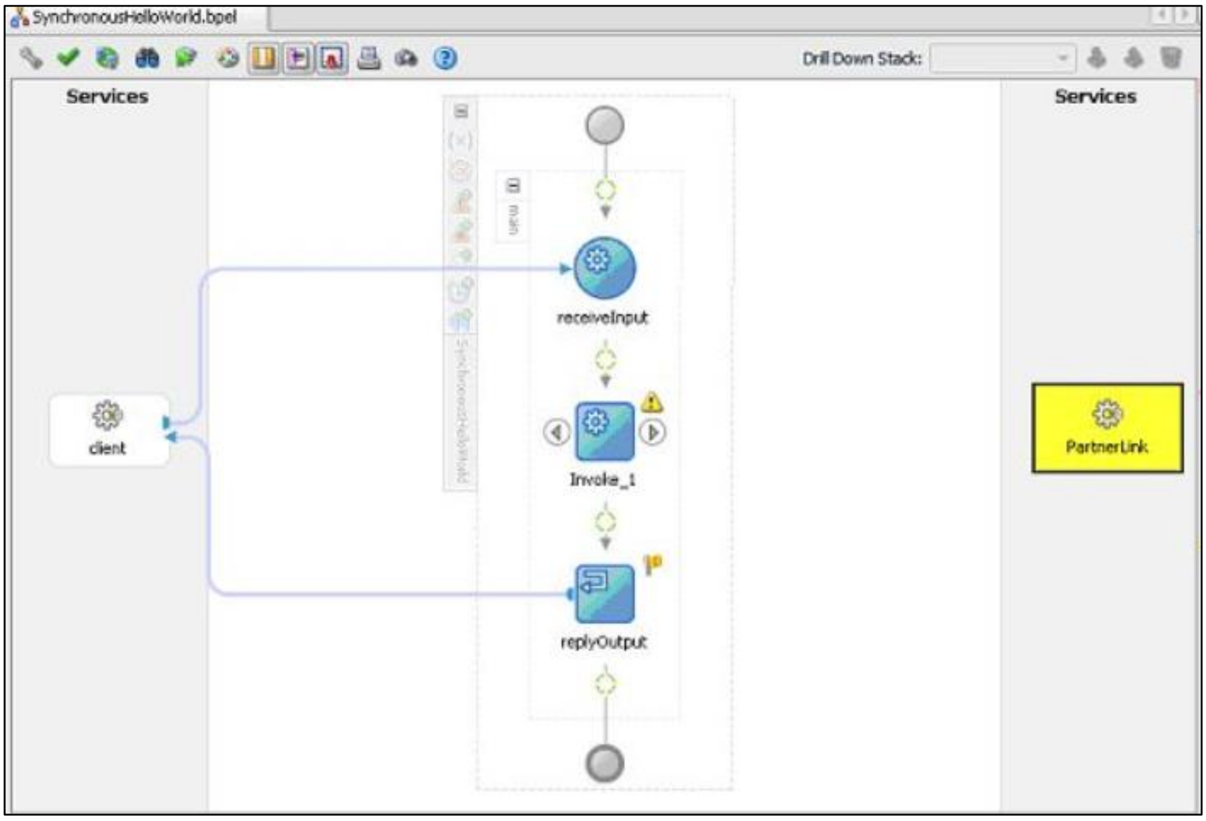
- When there are more than two applications involved in a transaction.
- This A-to-B-to-C-to-A transaction pattern can handle many transactions at the same time. Therefore, a mechanism is required for keeping track of which message goes where.
- This can be handled using WS-Addressing or correlation sets.



17. BPEL – Invoking a Synchronous Web Service

We have discussed in one of the previous chapters that Synchronous Web Service is one, which provides an immediate response to an invocation.

In the screenshot shown below, we have created a Synchronous BPEL Process which has a receive activity to accept the request from the user. The reply activity simultaneously sends the response back.



18. BPEL – Invoking an Asynchronous Web Service

As discussed before Asynchronous Web Service is one which sends a request to other web service and waits for the response.

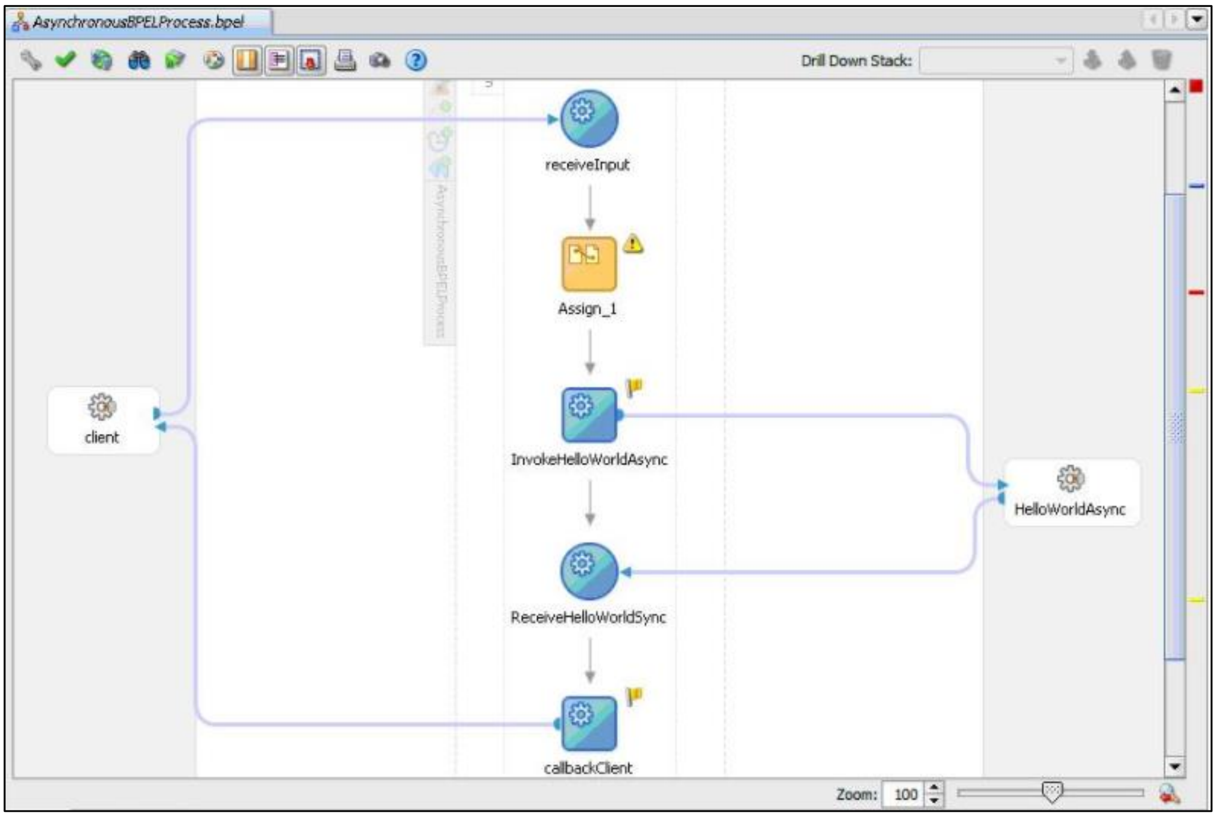
In the screenshot shown below, we have created the Asynchronous BPEL Process which has a receive activity to accept the request from the user. The assign activity further assigns values to the different elements in the request.

Next, the invoke activity invokes the HelloWorld Application which sends the response simultaneously and that is captured in receive activity.

Further, we have the callback activity which finally generates output and sends response asynchronously.

If you double-click the **receiveInput** or **callbackClient**, you will see each of them has only one variable.

```
receiveInput → inputVariable
callbackClient → outputVariable
```



19. BPEL – Using Parallel Flow

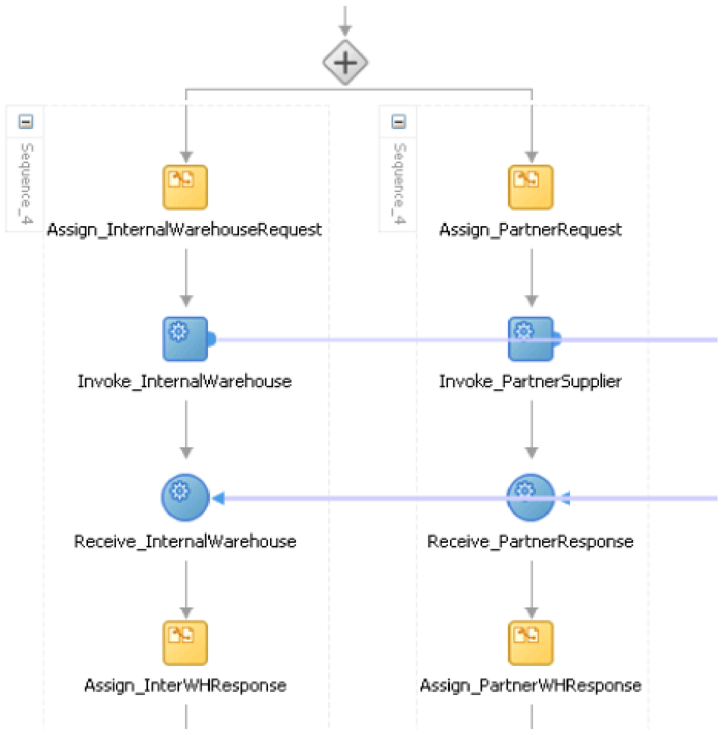
In this chapter, we will understand how parallel flow works in BPEL.

What is Flow Activity?

A flow activity typically contains many sequence activities, and each sequence is performed in parallel. A flow activity can also contain other activities.

For example, two asynchronous callbacks execute in parallel, so that one callback does not have to wait for the other to complete first. Each response is stored in a different global variable.

In the flow activity, the BPEL code determines the number of parallel branches. However, often the number of branches required is different depending on the available information.

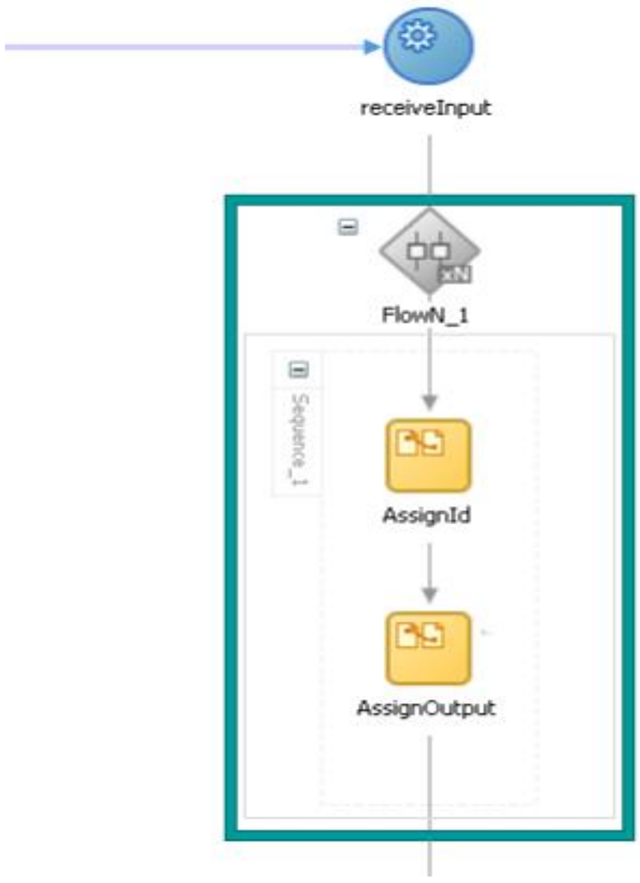


What is FlowN Activity?

The flowN activity creates multiple flows equal to the value of N, which is defined at the run time based on the data available and logic within the process. There is an Index variable increment each time a new branch is created, until the index variable reaches the value of N.

The flowN activity performs activities on an arbitrary number of data elements. As the number of elements changes, the BPEL process adjusts accordingly.

The branches created by flowN perform the same activities, but use different data. Each branch uses the index variable to look up input variables. The index variable can be used in the XPath expression to acquire the data specific for that branch.



20. BPEL – Using Conditional Branching

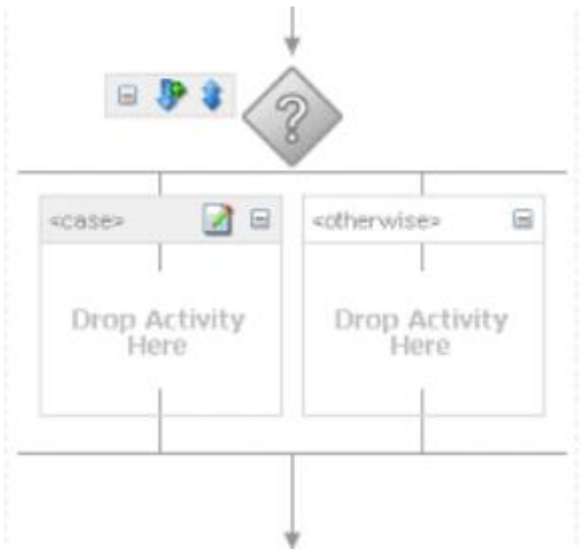
BPEL applies logic to make choices through conditional branching. The two different actions based on conditional branching are shown below:

Switch Activity

In this method, you set up two or more branches, with each branch in the form of an XPath expression. If the expression is true, then the branch is executed. If the expression is false, then the BPEL process moves to the next branch condition, until it either finds a valid branch condition, encounters an otherwise branch, or runs out of branches. If more than one branch condition is true, then BPEL executes the first true branch.

While Activity

You can use a while activity to create a while loop to select between two actions.



21. BPEL – Using Fault Handling

To understand how to use fault handling, we need to learn the basic architecture of a Service Composite in Oracle SOA Suite.

- **Service components** – BPEL Processes, Business Rule, Human Task, Mediator. These are used to construct a SOA composite application.

- **Binding components** – Establish connection between a SOA composite and external world.

- **Services** – Provides an entry point to SOA composite application.

- **Binding** – Defines the protocols that communicate with the service like SOAP/HTTP, JCA adapter, etc.

- **WSDL** – Defines the service definition of a web service.

- **References** – Enables a SOA composite application to send messages to external services

- **Wires** – Enables connection between service components.

Types of Faults

Let us now see the different types of faults.

Business faults

Occurs when application executes THROW activity or an INVOKE receives fault as response. Fault name is specified by the BPEL process service component. The fault handler using Fault name and Fault variable catches this fault.

Runtime faults

This is thrown by the system. These faults are associated with **RunTimeFaultMessage** and are included in http://schemas.oracle.com/bpel/extensionnamespace.

Ways of Fault Handling

In this section, we will learn about the different ways of fault handling.

Throw Activity

Throw activity explicitly throws the fault. The catch block catches this fault and the corresponding actions get executed thereby.

- Using throw activity, you can throw business faults & within the created scope, you can catch this fault and redirect to the caller (consumer) to take action.

- Instead of the above approach, you throw the same fault caught in catch activity of the created scope. In the main scope, you can catch this fault using the catchall activity.

Error Handler Framework (EHF)

The 2 main files used in EHF are:

- Fault-Policy.xml
- Fault-Bindings.xml

Whenever the BPEL process throws an error, the EHF will check whether the error exists in Fault-Bindings.xml files. If so, the action in the Fault-Policy.xml file will be taken. If the action is not found, the fault will the thrown and it will be handled in the catch block.

Fault management framework (Fault-Policy.xml and Fault-Bindings.xml) is kept inside a SOA Composite.

Fault-handlers like catch and catchall are inside a BPEL to catch all faults, but **fault policies will only be executed when an invoke activity fails**.

22. BPEL – Resubmitting a Faulted Process

In this chapter, we will see different scenarios related to the resubmitting of a faulted process.

Scenario A

The BPEL code uses a fault-policy and a fault is handled using the "ora-human-intervention" activity. The fault is then marked as Recoverable and the instance state is set to "Running".

Scenario B

The BPEL code uses a fault-policy and a fault is caught and re-thrown using the "ora-rethrow-fault" action. The fault is then marked as Recoverable and the instance state is set to "Faulted"; provided the fault is a recoverable one (like URL was not available).

23. BPEL – Incorporating Java and Java EE Code

There are several methods for incorporating Java and Java EE code in BPEL processes. Following are a few important methods:

- Wrap as a Simple Object Access Protocol (SOAP) service
- Embed Java code snippets into a BPEL process with the bpelx: exec tag
- Use an XML facade to simplify DOM manipulation
- Use bpelx: exec built-in methods
- Use Java code wrapped in a service interface

The Java Embedding activity allows us to add activities in a BPEL process. We can write a Java snippet using standard JDK libraries, the BPEL APIs, custom and 3rd party Java Classes included in JAR files in deployed SCA composites (in SCA-INF/lib directory) and Java Classes and libraries available on the Classpath for the SOA Suite run time.

Java Embedding means functionality hidden inside, in a not very decoupled way. The Java code is hard to maintain. By embedding Java in BPEL (XML driven), we start mixing technology, that require different skills as well as expensive XML to Java Object marshalling and unmarshalling.

The best use cases for Java Embedding seems to be for advanced logging/tracing or for special validations/transformations. However, not to replace built in capabilities of the BPEL engine as well as the other components in SOA Suite 11g and the adapters that come with it.

24. BPEL – Manipulating XML Data

XPath is mainly used to manipulate XMLs in the BPEL process. There are some valuable Xpath functions that can be used for manipulating XML. Let us see the functions below.

bpel:getVaribleData(varName, partName, xpathStr)

This can be used to extract a set of elements from a variable, using a XPath expression.

```
<bpel:copy>
     <bpel:from>

     <![CDATA[count(bpel:getVariableData('$Variable','$partName')/ns:return)]]>
     </bpel:from>
     <bpel:to variable="itemNumber"></bpel:to>
</bpel:copy>
```

bpel:getLinkStatus()

This can be used to evaluate and return a Boolean whether a particular link is active or inactive.

bpel:getVariableProperty(string, string)

This is helpful in extracting properties in Variables.

bpel:doXSLTTransform()

This performs the XSLT transformations.

string ()

This can be used to extract text content out of elements rather using /text ().

string-length()

This function is used to calculate the length of the string. But the **!=** operator seems not to work with the output from this function. So you can use **>** or **<** rather using **! =** .

Boolean Values

You can assign boolean values with the XPath boolean function.

```
     <assign>
        <!-- copy from boolean expression function to the variable -->
```

```
           <copy>
                   <from expression="true()"/>
                   <to variable="output" part="payload"
query="/result/approved"/>
           </copy>
       </assign>
```

Assigning a Date or Time

You can assign the current value of a date or time field by using the Oracle BPEL XPath function getCurrentDate, getCurrentTime, or getCurrentDateTime, respectively.

```
<!-- execute the XPath extension function getCurrentDate() -->
<assign>
   <copy>
      <from expression="xpath20:getCurrentDate()"/>
      <to variable="output" part="payload"
         query="/invoice/invoiceDate"/>
   </copy>
</assign>
```

Concatenating Strings

Rather than copying the value of one string variable (or variable part or field) to another, you can first perform string manipulation, such as concatenating several strings.

```
<assign>
   <!-- copy from XPath expression to the variable -->
   <copy>
      <from expression="concat('Hello ',
         bpws:getVariableData('input', 'payload', '/p:name'))"/>
      <to variable="output" part="payload" query="/p:result/p:message"/>
   </copy>
</assign>
```

Assigning String Literals

You can assign string literals to a variable in BPEL.

```
<assign>
   <!-- copy from string expression to the variable -->
   <copy>
```

```
      <from expression="'GE'"/>
      <to variable="output" part="payload" query="/p:result/p:symbol"/>
   </copy>
</assign>
```

Assigning Numeric Values

You can assign numeric values in XPath expressions.

```
<assign>
   <!-- copy from integer expression to the variable -->
   <copy>
      <from expression="100"/>
      <to variable="output" part="payload" query="/p:result/p:quantity"/>
   </copy>
</assign>
```

Note: A few XSLT functions were used to transform an XML document.

25. BPEL – Using Correlation Sets and Message Aggregation

BPEL correlation matches inbound messages with a specific process instance. When you need to associate specific data to a specific instance of a business process, you use correlation.

For example, while creating a process that verifies an account number and checks the account's credit limit. When verified, the process makes a call to another system to check inventory and, if the item is in stock, generates a purchase order. How does the purchase order know which account is to be debited? The answer to this question is correlation.

Correlation Sets

Correlation sets are used to uniquely identify process instances. You provide each correlation set with a unique name and then define it by one or more properties. Each property has a name and a type (for example, string or integer).

Property Alias

The property alias for each property in the correlation set needs to be defined. A property alias is a mapping that binds the property with the input or output values.

Important Points

Consider the following important points related to the **Correlation Sets and Message Aggregation**:

- A process that contains more than one receive or pick activity must have a correlation set.
- Correlation sets are initialized with values from process inbound or outbound messages.
- If you have groups of messages that are associated together with one specific process, you can set up one or more correlation sets to handle.

26. BPEL – Using Events and Timeouts in BPEL Processes

Asynchronous web services usually take a long time to return a response and as such, a BPEL process service component must be able to time out, or give up waiting, and continue with the rest of the flow after a certain amount of time. You can use the pick activity to configure a BPEL flow either to wait over a specified amount of time or to continue performing its duties. To set an expiration period for the time, you can use the wait activity. To manage message, events can be used particularly when the business process is waiting for callbacks from partner Web services.

Events

BPEL supports two types of events:

Message Events

These events are triggered by incoming messages through operation invocation on port types.

Alarm Events

These events are time-related and are triggered either after a certain duration or at a specific time.

- Often, however, it is more useful to wait for more than one message, of which only one will occur.
- Alarm events are useful when you want the process to wait for a callback for a certain period of time, such as 15 minutes.
    - If no callback is received, the process flow continues as designed.
    - Useful in loosely coupled service-oriented architectures, where you cannot rely on Web services being available all the time.

Pick Activity

The pick activity has 2 branches:

- onMessage: the code on this branch is equal to the code for receiving a response before a timeout was added.
- onAlarm: this condition has code for a timeout of one minute.

Wait Activity

The wait activity allows a process to wait for a given time period or until a time limit has been reached. Exactly one of the expiration criteria must be specified.

27. BPEL – Using the Notification Service

The BPEL process can be made use of for notification service. The process can be designed to send the following:

- email
- voice message
- instant messaging (IM), or
- short message service (SMS) notifications

For the services mentioned above, you can configure the channel for the incoming and outgoing message.

28. BPEL – Using Oracle BPEL Process Manager Sensors

Composite sensors within a SOA application provides the ability to define trackable fields on messages and enables you to find a specific composite instance by searching for a field or fields within a message. For example, a sensor could be defined for an order number within a message, thus allowing us to find the instance where the order number in question is found.

Composite sensors can be defined within a SOA application in several components:

- Service component (exposed service)
- Reference component (external reference)
- Mediator or BPEL component that have subscribed to a business event (publishing an event cannot have a sensor)

Different Ways to Define Composite Sensor

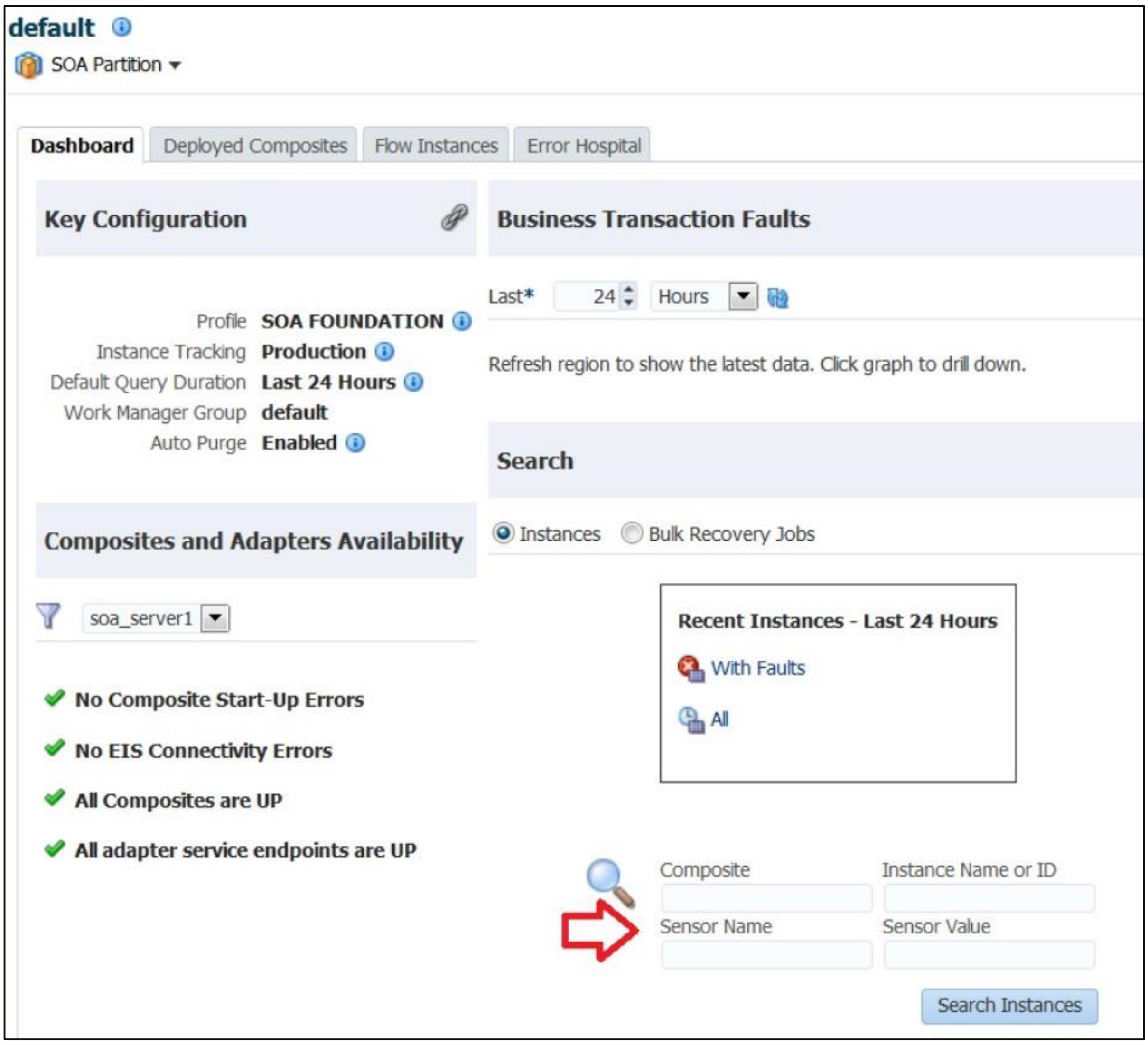
There are different ways to define a composite sensor:

- By specifying an existing variable as the sensor.
- By an expression with the help of the expression builder.
- By using properties (e.g. message header properties).

Sensors in Enterprise Manager

Defining a sensor allows for a quick search for data within a composite instance in the EM Console.

In the EM Console dashboard, a user can search for instances by sensor name and value.


default

SOA Partition

Dashboard | Deployed Composites | Flow Instances | Error Hospital

Key Configuration

Profile	SOA FOUNDATION
Instance Tracking	Production
Default Query Duration	Last 24 Hours
Work Manager Group	default
Auto Purge	Enabled

Composites and Adapters Availability

soa_server1

✓ No Composite Start-Up Errors

✓ No EIS Connectivity Errors

✓ All Composites are UP

✓ All adapter service endpoints are UP

Business Transaction Faults

Last* 24 Hours

Refresh region to show the latest data. Click graph to drill down.

Search

[✓ Instances] Bulk Recovery Jobs

Recent Instances - Last 24 Hours

With Faults

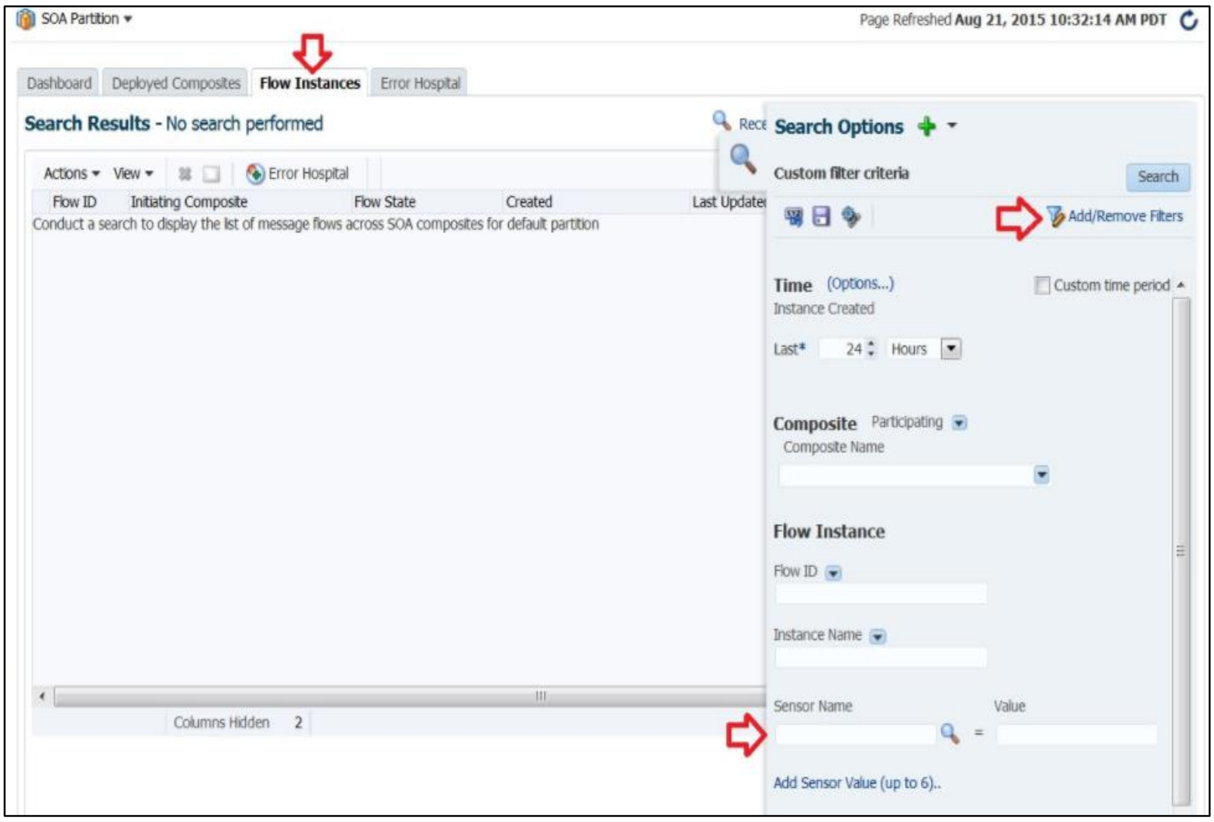
All

Composite | Instance Name or ID

Sensor Name | Sensor Value

Search Instances

In the “Flow Instances” tab, you can select sensors from the dropdowns and can use wildcard-like values for the sensor value.



29. BPEL – Difference between BPEL 1.1 and BPEL 2.0

New Activities have been added in 2.0 which have replaced the ones in 1.1.

<forEach>

This activity helps repeat the set of activities. The activity replaces the FlowN activity in BPEL 1.1 version.

<repeatUntil>

This activity comes of use if the body of an activity must be performed at least once. The XPath expression condition in the repeatUntil activity is evaluated after the body of the activity completes.

<if>-<elseif>-<else>

This activity replaces the switch activity in BPEL 2.0. The activity enables you to define conditional behavior for specific activities to decide between two or more branches. Only one activity is selected for execution from a set of branches.

<compensateScope>

This activity helps compensate the specified child scope.

<rethrow>

This activity has been added to fault handlers. It enables you to rethrow a fault originally captured by the immediately enclosing fault handler.