Chapter No. 17
"Event Delivery Network"
In this package, you will find:
A Biography of the authors of the book
A preview chapter from the book, Chapter NO.17 "Event Delivery Network"
A synopsis of the book’s content
Information on where to buy this book

About the Authors

Heidi Buelow is a product manager with Oracle and is responsible for Oracle SOA Suite programs such as beta and technical previews. Heidi joined Oracle in 2006, after having spent the previous 10 years as Chief Application Architect with a startup developing a Business Process Management engine, developer toolset, and application framework. Heidi started her career as a software developer at Xerox, working on the Xerox Network Services and Star Workstation products where she first learned to appreciate object-oriented and services-oriented technologies. She holds a Bachelor of Science degree in Computer Science from the University of Southern California.
Manas Deb is a senior director in the Fusion Middleware/SOA, BPM, Governance Suites Product Group at Oracle HQ. He currently leads outbound product management and many strategic engagement initiatives for Oracle's SOA, BPM, and Governance solutions, worldwide. He is also responsible for Oracle/HQ-based SOA Methodology initiatives. Manas has worked in the software industry for over twenty years, most of which was spent in software product management/marketing and on architecting; he has also led a wide variety of enterprise-level application development and business integration projects in a wide variety of industries. A graduate of the Indian Institute of Technology (KGP), Manas attended post-graduate studies at the University of Texas at Austin. He received his PhD in an inter-disciplinary program comprising Computer Science, Applied Mathematics, and Engineering. Manas also has an MBA with specialization in international business.

Jayaram Kasi is a product manager with Oracle, and focuses on SOA technologies. Before that, he had been a software architect for 20 years working on relational database kernels at HP, OLTP monitors based on DCE at HP, High Availability at HP, ECommerce Infrastructure at Commerce One, and Enterprise Service Bus at BEA. Jayaram has a Bachelor of Science degree in electrical engineering from the Indian Institute of Technology, and a Master of Science in electrical engineering from the University of Hawaii.

Demed L'Her is Director of Product Management at Oracle, where he is responsible for the Oracle SOA Suite. He has been with Oracle since 2006, focusing on ESB, JMS, and next-generation SOA platforms. Before joining Oracle, Demed spent eight years with TIBCO Software, a pioneer in electronic trading, message-oriented middleware, and enterprise integration. He has been involved in some of the largest messaging and integration projects around the world, from trading floors in Tokyo to semiconductor manufacturing fabs in Arizona. Demed holds an Engineering Degree from the Ecole Nationale Supérieure des Télécommunications de Bretagne, a Bachelor of Science in Electrical Engineering from Université de Bretagne Occidentale and a Master of Science in Computer Science from Université de Rennes I.

For More Information:  
**Prasen Palvankar** is a Director of Product Management at Oracle and is responsible for providing strategic support to Oracle's SOA Suite current and prospective customers. He is also responsible for outbound SOA Suite product-related activities, including field and partner enablement and training. Prasen has over 20 years of experience in software development and has been working for Oracle since 1998. He was a Technical Director in the Advanced Technology Solutions group at Oracle Consulting, delivering large-scale integration projects before taking on his current role four years ago. Before joining Oracle, Prasen worked as a Principal Software Engineer at Digital Equipment Corporation.
Getting Started with Oracle SOA Suite 11g R1
A Hands-On Tutorial

As the concept of Service-Oriented Architecture has matured, it has triggered the emergence of new, sophisticated, and specialized tools: Enterprise Service Buses (ESB) for service virtualization, BPEL for orchestration, Human Workflow, Business Rules for externalizing key pieces of logic, and so on. As a result, developers now have a rich set of tools to work with. However, this can itself present a challenge: how can one keep up with all these various tools and their capabilities? One of the key goals of Oracle SOA Suite 11g is to assemble these tools in a cohesive, simple-to-use, and highly-integrated development environment. This book, organized around a tutorial that is built in an iterative fashion, will guide you through all the components of the suite and how they relate to each other.

The authors are part of the Oracle SOA Suite product management team, and the idea of the book came as we were delivering an earlier version of this material, as an accelerated internal training at Oracle—before the product was even released. These training sessions were very well received and we decided it was worth sharing this material with a larger audience.

This book is not meant to be used as reference material—it is an accelerated learning path to the Oracle SOA Suite. The focus is on breadth rather than on depth. More specifically, we wanted to highlight the key capabilities and role of each product in the Oracle SOA Suite and explain how they can be put to work together to deliver highly capable and flexible applications. Too often we, as developers, tend to stretch the limits of (not to say abuse!) a few technologies, simply to stay within our comfort zone—and because there is always so little time to learn new things. With its streamlined format, we hope this book will give you the confidence to further explore some of these technologies you had never looked at before.

What This Book Covers
The principal aim of this book is to get you operational with Oracle SOA Suite 11gR1 quickly and easily. In this spirit, the largest part of this book is dedicated towards a set of hands-on step-by-step tutorials that build a non-trivial SOA composite that you can deploy, test, run, monitor, and manage.
Chapter 1 starts the book off with a quick refresher on some of the useful concepts regarding SOA and services and concludes with an introduction to Service Component Architecture (SCA).

Chapter 2 discusses the key challenges in the technical implementation of SOA-based applications and how Oracle SOA Suite 11g leverages SCA principles to address these challenges.

Chapter 3 describes the business and technical requirements for a purchase order (PO) processing composite and gives you an overview of how the complete solution will be built up in a set of discrete steps using a series of tutorials using Oracle SOA Suite 11gR1.

Chapter 4 gives you the necessary instructions for download, installation, and configuration of Oracle SOA Suite 11gR1.

The core functionalities of the PO processing composite that is described in Chapter 3 are built in Chapters 5 through 10. This series of chapters will teach you the basics of working with Oracle SOA Suite 11g and the IDE (JDeveloper).

You start building the composite using a mediator, as well as web services and database adapters. You then add a file adapter and a BPEL (Business Process Execution Language) component to create a process that orchestrates the overall flow, adding human interaction, creating conditional process execution using business rules, and accessing external services via a JMS (Java Message Service) adapter. At the end of each and every chapter, you will have a composite that can be deployed, run, and tested. You are advised to go through these tutorial chapters, 5-10 in a sequential manner.

The tutorials in Chapters 11 through 19 let you add more functionality to the composite and explore some of the operational features of Oracle SOA Suite 11gR1. You will learn service re-use and virtualization using Oracle Service Bus (OSB), explore some of the composite life cycle management features, test the composite using the unit testing framework, incorporate exception handling, add security policies to a service, set up a business activity-level tracking of the composite transactions using Oracle Business Activity Monitoring (BAM), work with events using the unified services and events platform of Oracle SOA Suite 11g, handle data using Service Data Object (SDO) specification, and connect the composite to a Business-to-Business (B2B) gateway using Oracle B2B.

By the end of Chapter 19, you should have a good grasp of all components in Oracle SOA Suite 11gR1, and be able to create modular, full-featured service composites. The concluding remarks in Chapter 20 will briefly discuss some of the ways you could use such composites to provide business benefits.
Event Delivery Network

Creating truly decoupled composite SOA applications requires a complete separation of the service consumer and the service provider. This is typically achieved through the use of asynchronous messaging. In an asynchronous messaging pattern, applications can perform in a "fire and forget" mode. This removes the need of an application to know details of the application on the other side. Additionally, it also improves resource utilization as applications are not holding onto resources until the interaction is complete. On the other hand, this introduces complexities of creating and managing message queues and topics. It requires that both the publisher of the message and the consumer use the same messaging technology. Each messaging system also has its own constraints on the types of programming languages and environments that can use the service.

In a service-oriented world, this tight coupling to the implementation of the underlying messaging system is at odds with the fundamental requirement of implementation independence. What's needed is a level of abstraction that allows applications to generate an event using business terms and associate a business object in an implementation-independent form.

Oracle SOA Suite 11g addresses this with the introduction of a new feature in the form of the Event Delivery Network.
Introducing events

The Event Delivery Network (EDN) in Oracle SOA Suite 11g provides a declarative way to use a publish/subscribe model to generate and consume business events without worrying about the underlying message infrastructure.

Developers only need to produce or consume events without having to deal with any particular messaging API like JMS, AQ, and MQ, and so on. Consuming an event means expressing an interest in the occurrence of a specific situation, while producing an event means advertising this occurrence.

Using the same concepts that are used in Web Service Definition Language (WSDL), EDN uses an XML-based Event Definition Language, which allows you to define the event and its associated, strongly typed data. This definition is then registered with the SOA Infrastructure and is available to all composites to publish or subscribe.

<table>
<thead>
<tr>
<th>SERVICES</th>
<th>MESSAGING</th>
<th>EDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSDL:</td>
<td>JMS API:</td>
<td>EDL:</td>
</tr>
<tr>
<td>Standard service interface model</td>
<td>Application Programming Interface</td>
<td>Event Definition Language</td>
</tr>
<tr>
<td>XSD:</td>
<td>Handful of raw types</td>
<td>XSD</td>
</tr>
<tr>
<td>Strong typing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business-oriented</td>
<td>Developer-oriented</td>
<td>Business-oriented</td>
</tr>
<tr>
<td>Wealth of tools</td>
<td>Mostly coding tools</td>
<td>Fully declarative</td>
</tr>
</tbody>
</table>

This allows for a fully-declarative, business-oriented, rather than developer-oriented approach for using events in a SOA application.

Another feature of EDN is the ability to publish and subscribe to these events from a variety of programming environments such as Java, PL/SQL, SOA Composites, and ADF-BC applications.

With EDN, Oracle has fostered the concept of events in the context of SCA, and this has given birth to an additional SCA specification: the "Assembly Model Specification Extensions for Event Processing and Pub/Sub". Since EDN in Oracle SOA Suite 11g predates the specification, you will find differences in the details, but the concepts are the same and EDN will eventually be aligned with the standard that emerges from the specification.
Creating and managing event definitions

Events are defined using an Event Definition Language (EDL), an XML schema used to build business event definitions. An EDL consists of the following:

- A global name.
- Custom headers: These can be used for content-based routing without having to parse the XML payload and apply XPath queries to get at the value to be used for the routing decisions. For instance, one can put the purchase order type in the custom header. This easily accessible custom header could then be used to efficiently decide how to process delivered events.
- Payload definition: Typically this is an XML schema for the business data that needs to accompany the event. For example, a "NewPO" event's payload definition will be a schema for a purchase order.

Event definitions can be created declaratively in JDeveloper in a couple of different ways depending on triggering conditions.

To publish events from a SOA composite, one would create new event definitions and register them with the SOA Infrastructure. It is this option that you will be able to try out later in this chapter during the hands-on exercise.

If you want to raise events on one or more database operations such as insertion of a new row or update of an existing one, you can use ADF-BC to define these events. Application Development Framework (ADF), a model-view-controller pattern based UI development framework for creating Rich Internet Applications (RIAs) and Business Components (BC), an object-relational mapping tool, provide an easy way to define events. ADF-BC has built-in support for associating these events with database actions like insert, delete, and modify. For example, an event called "NewCustomerAdded" could be generated every time a new customer record is inserted into the database.

Registered events, their subscribers, and any faulted deliveries can all be tracked and monitored using the Oracle Enterprise Manager, in the same fashion that you would be managing and monitoring other aspects of your SOA infrastructure.
Consuming events from the Event Delivery Network

The first step to consume an event is to discover its definition. Event definitions can be stored locally in a composite or globally on the server—in either case, they can be discovered through the JDeveloper resource catalogue.

To subscribe to an event within an SOA composite application, you start by defining a Mediator component, selecting the event of interest from a list of registered events in the network.

To further refine a subscription, and narrow down the information you will receive, you can define an XPath-based filter on your subscriptions. For example, for an event named "NewPOCreated", you could either subscribe to all "NewPOCreated" events (default when no filter is defined) or only those where the order amount is more than a million dollars (by defining an XPath-based filter on the order amount field within the payload).

Tutorial: Use of Event Delivery Network (EDN) and POProcessing

Let's try out a simple EDN use case. You extend the POProcessing example that you have been building in previous chapters.

Chapters 11-19 can be done in any order after completing Chapter 10. In order to follow the tutorial in this chapter you must have either completed the tutorial from Chapter 10 or higher, or set up the environment as described in Chapter 4 and deployed the solution for Chapter 10, which can be found in c:\po\solutions\ch10.

To illustrate the usage of events, you modify the existing POProcessing composite to accept new orders from two sources: the Web Service interface, which you have already implemented, and another application that will publish events to indicate new orders.

Modifying the composite to consume events

In this step, you modify the POProcessing composite to use EDN. You add a new mediator component called receivePO, which subscribes to an event called NewPO and sends the received PO to the routePO service.
Defining the event

1. Open the POProcessing application in JDeveloper and open composite.xml, and click on the Event Definition Creation icon:

2. In the Event Definition Creation window, enter POEvents as the name of the event definition. Accept the namespace value generated.

Event names are fully qualified names, which means the combination of a namespace and the event name together identify a unique event. For instance, the event called NewCustomerAdded with name space http://schemas.oracle.com/events/edl/POEvents is different than the event with the same name but under a different namespace, for example, event NewCustomerAdded with namespace http://schemas.oracle.com/event/edl/CRMEvents.

3. Add a new event by clicking on the + icon.
4. Select the PurchaseOrder element from the po.xsd file using the chooser.
5. Enter NewPO as the name of the event and click on OK to close the Add an Event window.
6. Click on OK to complete the event definition.
7. Close the POEvents.edl panel.
8. Save all.

You have just created a new event called POEvent. It is now available to this and other composites to subscribe to or publish. Whenever this event is delivered, it will also carry with it a document for the new purchase order.

**Subscribing to the NewPO event**

1. Drag-and-drop a **Mediator component** onto the composite.
2. Name the mediator **receiveNewPO**.
3. Select **Subscribe to Events** for the **Template**.
4. Click on the + to add an event. Select the **NewPO** event from the **Event Chooser** window.
5. Click on OK to accept and create the mediator.
6. Save all.
7. Connect the **receiveNewPO** mediator to the **routePO** mediator:

8. Double-click on the **receiveNewPO** mediator and define a new transformation map. In the transformation-map, map all the fields from the source to the target.
9. Close the mediator and save all.

**Deploying and testing**
1. Deploy the **POProcessing** composite to the server.
2. Browse to the EM console at **http://localhost:7001/em**.
3. Right-click on folder `soa-infra (soa_server1)` under folder SOA:

4. Click on item **Business Events**.

5. You should see the Business Events management page with the **NewPO** event listed in the **Events** tab.

6. Select the event and click on the **Test...** button to publish the event.

---

For More Information:

7. In the pop-up window, enter the following XML and click on **Publish**:

   ```xml
   <PurchaseOrder xmlns="http://xmlns.oracle.com/ns/order">
     <CustID>1111</CustID>
     <ID>33412</ID>
     <productName>Sony Bluray DVD Player</productName>
     <itemType>Electronics</itemType>
     <price>350</price>
     <quantity>5</quantity>
     <status>Initial</status>
     <ccType>Mastercard</ccType>
     <ccNumber>1234-1234-1234-1234</ccNumber>
   </PurchaseOrder>
   ```

8. You should get a "The Event published successfully" message.

9. Click on **soa-infra** in the navigation panel on the left to get the soa-infra dashboard.

10. Click on the **POProcessing** composite to view new instances. You should see an instance created for processing the event you just published. Click on the instance ID to view the flow trace.

This was just one way of publishing an event and will typically be used as a way to test the composite. In most cases, the events will be published from a number of sources—from a Mediator, from PL/SQL by calling the EDN_PUBLISH_EVENT stored function or from a Java class using the EDN API. An ADF-BC application can also publish events based on database insert, update, and delete operations.
Summary

You cannot really build comprehensive SOA applications without business events. Traditionally, this requirement has been fulfilled by message-oriented-middleware (MOM). However, MOM-based solutions don't necessarily fit very well within a service-oriented architecture. They are low-level technical solutions that provide no business semantics, whereas one of the main objectives of creating services is to provide business functions using semantics that are better understood by business analysts.

With EDN, Oracle SOA Suite 11g fills this gap by providing an event-handling solution that allows creation and use of events using business semantics, without the publisher or subscriber of the event ever having to worry about the mechanics of messaging.

As you have seen in this short lab, events are created in a way that directly maps to actual business events, in this case creation of a new purchase order. The process of subscribing to this event was done declaratively without having to configure any messaging queues or topics.

Quick-build Instructions

This section gives you all of the operations and names for objects created in this chapter. Experienced users can use this to bypass the step-by-step instructions and complete this chapter more quickly. The complete details for any particular operation listed here can be found in the preceding sections. The information is organized in the same manner that it is introduced in the chapter:

- Define an event:
  - Event Definition Name: POEvents
  - Namespace: http://schemas.oracle.com/events/edl/POEvents

- Subscribe to the NewPO event:
  - Mediator name: receiveNewPO
  - Mediator template: Subscribe to Events
  - Choose event: NewPO

- Connect receiveNewPO to routePO:
  - Create a new transformation map, to map data received from the event to data to be sent to the routePO mediator

The application is completed. Deploy and test.
Where to buy this book

You can buy Getting Started with Oracle SOA Suite 11g R1 from the Packt Publishing website: http://www.packtpub.com/getting-started-with-oracle-soa-suite-11g-r1/book

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.