Warranty Disclaimer

THIS DOCUMENT AND ALL INFORMATION PROVIDED HEREIN (THE "INFORMATION") IS PROVIDED ON AN "AS IS" BASIS AND FOR GENERAL INFORMATION PURPOSES ONLY. ORACLE EXPRESSLY DISCLAIMS ALL WARRANTIES OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. ORACLE MAKES NO WARRANTY THAT THE INFORMATION IS ERROR-FREE, ACCURATE OR RELIABLE. ORACLE RESERVES THE RIGHT TO MAKE CHANGES OR UPDATES AT ANY TIME WITHOUT NOTICE.

As individual requirements are dependent upon a number of factors and may vary significantly, you should perform your own tests and evaluations when making technology infrastructure decisions. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle Corporation or its affiliates. If you find any errors, please report them to us in writing.

Third Party Content, Products, and Services Disclaimer

This document may provide information on content, products, and Services from third parties. Oracle is not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and Services. Oracle will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or Services.

Limitation of Liability

IN NO EVENT SHALL ORACLE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY YOU OR ANY THIRD PARTY, WHETHER IN AN ACTION IN CONTRACT OR TORT, ARISING FROM YOUR ACCESS TO, OR USE OF, THIS DOCUMENT OR THE INFORMATION.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
# Contents

## Preface
- ORA EDA Perspective ................................................................. vii
- IT Strategies from Oracle (ITSO) ................................................ viii
- Audience ................................................................. ix
- Document Structure ............................................................. ix
- How to Use This Document ................................................... ix
- Conventions ................................................................. ix

## 1 Introduction
1.1 Enterprise Foundation for EDA Infrastructure .................. 1-1
1.1.1 Messaging Infrastructure ................................................. 1-2
1.1.2 Middleware Infrastructure ............................................... 1-2
1.1.3 Service Infrastructure .................................................... 1-2
1.1.4 Data Infrastructure ....................................................... 1-2

## 2 EDA Logical View
2.1 Engineering ........................................................................ 2-3
2.1.1 Integrated Development Environment (IDE) ............... 2-3
2.1.2 Event Repository .......................................................... 2-4
2.1.3 Configuration Tools ...................................................... 2-6
2.2 Event Producers .................................................................. 2-6
2.2.1 Internal Producers ........................................................ 2-6
2.2.2 External Producers ....................................................... 2-7
2.3 Inbound Adapters .............................................................. 2-7
2.4 Inbound Channels .............................................................. 2-8
2.5 Processors .......................................................................... 2-8
2.5.1 Query Processor .......................................................... 2-9
2.5.2 Pattern Recognizer ....................................................... 2-10
2.5.3 Direct Handlers ........................................................... 2-10
2.5.4 Extension Cartridge ...................................................... 2-10
2.5.5 Analytics Engine ........................................................ 2-10
2.5.6 Splitter .......................................................................... 2-10
2.5.7 Enricher ......................................................................... 2-11
2.5.8 Aggregator ................................................................. 2-11
2.5.9 Recorder ................................................................. 2-11
3 Product Mapping View

3.1 Oracle Complex Event Processing (CEP) ................................................................. 3-3
  3.1.1 Oracle CEP Development Environment ............................................................... 3-3
  3.1.2 Oracle CEP Architecture ....................................................................................... 3-4
  3.1.3 Oracle CEP High Availability Architecture .......................................................... 3-5
3.2 Oracle Business Activity Monitoring (Oracle BAM) ................................................... 3-6
3.3 Oracle Service Bus (OSB) ........................................................................................... 3-7
3.4 Oracle JRockit Real Time (JRRT) ................................................................................ 3-8
  3.4.1 Deterministic Garbage Collection (GC) ................................................................. 3-9
3.5 Oracle Business Rules .................................................................................................. 3-10
3.6 Oracle BPEL Process Manager .................................................................................... 3-11
3.7 Oracle Enterprise Repository (OER) .......................................................................... 3-11
3.8 Oracle Coherence ......................................................................................................... 3-12
3.9 Oracle Business Intelligence Suite ............................................................................... 3-12
3.10 Oracle database and Oracle Real Application Clusters (RAC) .................................... 3-13
3.11 Security Products ....................................................................................................... 3-13

4 EDA Deployment View

4.1 EDA Deployment ......................................................................................................... 4-1
4.2 Scalability and High Availability .................................................................................. 4-2
4.3 Scalable Deployment ..................................................................................................... 4-3
4.4 Highly Available EDA Deployments ............................................................................ 4-4
  4.4.1 HA Deployment: Active-Active .............................................................................. 4-5
  4.4.2 HA Deployment: Active-Passive ............................................................................ 4-6
4.5 Scalable and HA Deployment ....................................................................................... 4-6
5 Summary

A EDA Detailed Logical View

B Further Reading and References
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–1</td>
<td>Enterprise Foundation for EDA</td>
<td>1-1</td>
</tr>
<tr>
<td>2–1</td>
<td>EDA Conceptual View</td>
<td>2-1</td>
</tr>
<tr>
<td>2–2</td>
<td>EDA High Level Logical View</td>
<td>2-2</td>
</tr>
<tr>
<td>2–3</td>
<td>EDA Engineering</td>
<td>2-3</td>
</tr>
<tr>
<td>2–4</td>
<td>EDA Integrated Development Environment Capabilities</td>
<td>2-4</td>
</tr>
<tr>
<td>2–5</td>
<td>Event Repository</td>
<td>2-5</td>
</tr>
<tr>
<td>2–6</td>
<td>Event Producers</td>
<td>2-6</td>
</tr>
<tr>
<td>2–7</td>
<td>Adapters</td>
<td>2-8</td>
</tr>
<tr>
<td>2–8</td>
<td>Inbound Channels</td>
<td>2-8</td>
</tr>
<tr>
<td>2–9</td>
<td>Event Processor Inputs in an EPN</td>
<td>2-9</td>
</tr>
<tr>
<td>2–10</td>
<td>Event Processor</td>
<td>2-9</td>
</tr>
<tr>
<td>2–11</td>
<td>Outbound Channels</td>
<td>2-11</td>
</tr>
<tr>
<td>2–12</td>
<td>Outbound Channels and Cache</td>
<td>2-12</td>
</tr>
<tr>
<td>2–13</td>
<td>Event Cache in an EPN</td>
<td>2-13</td>
</tr>
<tr>
<td>2–14</td>
<td>Event Listeners</td>
<td>2-13</td>
</tr>
<tr>
<td>2–15</td>
<td>Event Consumers</td>
<td>2-13</td>
</tr>
<tr>
<td>2–16</td>
<td>Event Mediation/Bus</td>
<td>2-15</td>
</tr>
<tr>
<td>2–17</td>
<td>Event Monitoring and Management</td>
<td>2-16</td>
</tr>
<tr>
<td>2–18</td>
<td>EDA Security</td>
<td>2-17</td>
</tr>
<tr>
<td>2–19</td>
<td>Event Data Management</td>
<td>2-18</td>
</tr>
<tr>
<td>3–1</td>
<td>EDA Oracle Product Mapping</td>
<td>3-2</td>
</tr>
<tr>
<td>3–2</td>
<td>Oracle CEP High Level Architecture</td>
<td>3-4</td>
</tr>
<tr>
<td>3–3</td>
<td>Oracle CEP Components</td>
<td>3-5</td>
</tr>
<tr>
<td>3–4</td>
<td>Oracle BAM Logical View</td>
<td>3-6</td>
</tr>
<tr>
<td>3–5</td>
<td>JRockit Architecture</td>
<td>3-9</td>
</tr>
<tr>
<td>3–6</td>
<td>Oracle Business Rules</td>
<td>3-10</td>
</tr>
<tr>
<td>3–7</td>
<td>BPEL Process Manager</td>
<td>3-11</td>
</tr>
<tr>
<td>4–1</td>
<td>Typical EDA Deployment</td>
<td>4-1</td>
</tr>
<tr>
<td>4–2</td>
<td>EDA Scalability Options</td>
<td>4-3</td>
</tr>
<tr>
<td>4–3</td>
<td>Scaling EDA deployments</td>
<td>4-4</td>
</tr>
<tr>
<td>4–4</td>
<td>Active-Active HA Deployment</td>
<td>4-5</td>
</tr>
<tr>
<td>4–5</td>
<td>HA Deployment Active-Passive</td>
<td>4-6</td>
</tr>
<tr>
<td>4–6</td>
<td>Scalable HA Deployment</td>
<td>4-7</td>
</tr>
</tbody>
</table>
Preface

The business landscape in today’s environment is very dynamic and demanding. Businesses can no longer be operating in reactive mode. If they want to maintain and increase their competitive advantage, they must become very agile and proactive. Business events must be acted upon in a timely manner and business decisions must be made in real time. Situation awareness provides businesses competitive advantage that others may lack.

Traditional “pull” based paradigms are not best suited to handle this need alone. They need to be complemented with “push” based technologies to create this niche. Event Driven Architecture (EDA) is one such approach that nicely complements the existing enterprise technologies to provide the infrastructure needed to support the real-time, event-based push architectures.

The other important driver for EDA is the need to analyze complex relationships between business events and identify business opportunities, threats, and anomalies that need to be addressed. Business users need this insight in time to improve the quality of the decisions they make and to stay nimble in business.

Today, probes and sensors are deployed in everything from IT networks to enterprise software systems and physical world devices (through RFID readers, barcode scanners, manufacturing equipment sensors, and others). As these systems continue to proliferate, they generate events at a growing rate. Significant improvements in operational business decisions await those organizations that can capture and process these events into meaningful business insight. Analyzing and interpreting millions of minor events is beyond human capability. EDA is the solution for this problem of handling a high volume of events.

ORA EDA Perspective

Oracle Reference Architecture (ORA) defines a detailed and consistent architecture for developing and integrating solutions based on Oracle technologies. The ORA EDA documents present the ORA architectural concepts from the perspective of EDA, highlighting the specific details of EDA as an elaboration of the ORA core concepts with respect to this technological approach. This ORA EDA perspective comprises two documents:

- EDA Foundation: primarily a reference architecture for EDA, including principles, standards, definition of EDA, EDA concepts, and its relationship to ORA
- EDA Infrastructure: relates the EDA capabilities, as defined by the reference architecture, to the Oracle infrastructure and provides a number of architecture views to help the architects and developers focusing on EDA.
The ORA EDA perspective is incremental to the ORA core relying on all underlying descriptions and architectural principles. The underlying ORA core concepts are not reproduced in this perspective document except where it is necessary to show EDA-specific views.

The EDA infrastructure includes a number of views of the Event Driven Architecture including logical, deployment and product mapping. This document also explores the high availability and scalability aspects of EDA deployments.

This document does not attempt to offer a methodology for the implementation of an EDA strategy, although it is expected that it will be used in conjunction with such a methodology as the foundation for the architectural approach.

**IT Strategies from Oracle (ITSO)**

IT Strategies from Oracle (ITSO) is a series of documentation and supporting collateral designed to enable organizations to develop an architecture-centric approach to enterprise-class IT initiatives. ITSO presents successful technology strategies and solution designs by defining universally adopted architecture concepts, principles, guidelines, standards, and patterns.

ITSO is made up of three primary elements:

- **Oracle Reference Architecture (ORA)** defines a detailed and consistent architecture for developing and integrating solutions based on Oracle technologies. The reference architecture offers architecture principles and guidance based on recommendations from technical experts across Oracle. It covers a broad spectrum of concerns pertaining to technology architecture, including middleware, database, hardware, processes, and services.

- **Enterprise Technology Strategies (ETS)** offer valuable guidance on the adoption of horizontal technologies for the enterprise. They explain how to successfully execute on a strategy by addressing concerns pertaining to architecture, technology, engineering, strategy, and governance. An organization can use this material to measure their maturity, develop their strategy, and achieve greater levels of success and adoption. In addition, each ETS extends the Oracle Reference Architecture by adding the unique capabilities and components provided by that particular technology. It offers a horizontal technology-based perspective of ORA.

- **Enterprise Solution Designs (ESD)** provide extensive architecture perspectives that are tailored and enhanced for industry verticals. Each solution design is built
on a customization of ORA that includes specific capabilities, processes, services, and applications for that industry. Oracle products are mapped to the solution designs in order to illustrate where they fit within the overall industry architecture.

**ORA EDA Foundation, along with ORA EDA Infrastructure**, extend the Oracle Reference Architecture. They are part of a series of documents that comprise the EDA Enterprise Technology Strategy, which is included in the IT Strategies from Oracle collection.

Please consult the [ITSO web site](https://www.oracle.com/) for a complete listing of SOA and ORA documents as well as other materials in the ITSO series.

### Audience

This document is intended for enterprise architects, application architects, project managers and developers. The material is designed for a technical audience that is interested in learning about the intricacies of Event Driven Architecture and how to design and build enterprise class EDA infrastructure.

### Document Structure

This document is organized into chapters that introduce EDA infrastructure and various views of the EDA infrastructure including logical, product mapping, and deployment views.

**Chapter 1, "Introduction"** - provides an introduction to Event Driven Architecture (EDA) infrastructure and outlines the enterprise foundation for EDA.

**Chapter 2, "EDA Logical View"** - provides various views of the EDA logical architecture and describes the logical components required for EDA infrastructure.

**Chapter 3, "Product Mapping View"** - describes how to implement the EDA logical architecture using Oracle products.

**Chapter 4, "EDA Deployment View"** - describes the deployment architecture including scalability and high availability configurations.

**Chapter 5, "Summary"** - summarizes the ORA EDA Infrastructure document.

**Appendix A, "EDA Detailed Logical View"** - provides a detailed view of the EDA logical architecture.

**Appendix B, "Further Reading and References"** - provides a list of documents and reading resources for further information and reference.

### How to Use This Document

This document is designed to be read from beginning to end. Those that are already familiar with EDA may wish to skip the introduction chapters and proceed with the conceptual view.

### Conventions

The following typeface conventions are used in this document:
<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface text</strong></td>
<td>Boldface type in text indicates a term defined in the text, the <em>ITSO Master Glossary</em>, or in both locations.</td>
</tr>
<tr>
<td><em>italic text</em></td>
<td>Italics type in text indicates the name of a document or external reference.</td>
</tr>
<tr>
<td><strong>underline text</strong></td>
<td>Underline text indicates a hypertext link.</td>
</tr>
</tbody>
</table>
1 Introduction

The concept of Event Driven Architecture (EDA) has been around for a while now. Messaging infrastructure that support point-to-point messaging and publish-subscribe messaging have been used to implement event-driven systems. Evolution of technology has paved way for capturing and processing complex events. The current state of the economy requires businesses to be nimble. This drives the need for architectures that support real time decision making. This means that business events must be acted upon as soon as they occur in real life. EDA enables the build-out of infrastructure that supports agile architectures, eventually leading to business agility.

1.1 Enterprise Foundation for EDA Infrastructure

EDA infrastructure is almost never implemented in isolation. It is always built to support and integrate with existing technologies such as SOA and Messaging. Consequently, EDA infrastructure builds on top of existing middleware and messaging infrastructure of the enterprise. So it is important to understand the basic enterprise foundation for EDA. This includes

- Messaging Infrastructure
- Middleware Infrastructure
- Service Infrastructure
- Data Infrastructure

Figure 1–1 Enterprise Foundation for EDA
1.1.1 Messaging Infrastructure

Messaging backbone is a key foundation for building EDA. Messaging is the primary way of implementing asynchronous, loosely coupled architectures. In EDA, events are transmitted from the producer to the consumer through a number of intermediaries. The sender and receiver are usually decoupled through a messaging channel. In addition to providing the loose coupling required by EDA, messaging backbone also provides reliable messaging and store and forward (SaF) capabilities as well. Business critical events may be transported over reliable messaging channels to improve reliability.

Messaging infrastructures support several styles including one-to-one (simple queuing) and one-to-many (publish/subscribe and broadcast). These styles may be used appropriately in an EDA to fulfill specific business requirements.

1.1.2 Middleware Infrastructure

EDA itself is part of enterprise middleware and it can benefit from the rest of the middleware infrastructure. This includes application servers, web servers, transaction monitors, and Enterprise Application Integration (EAI) infrastructure. Middleware provides a distributed architecture platform on which EDA infrastructure is built.

1.1.3 Service Infrastructure

The true potential of EDA is realized only when combined with Services. EDA can greatly leverage the components of Service infrastructure.

The ORA EDA Foundation document discussed an event repository where the event specifications are maintained in order to promote search and discovery. The event repository can leverage the same enterprise repository used by Services. This allows the inter-relationships between the events and Services to be captured in an event-driven SOA.

The Enterprise Service Bus (ESB) can be used as the Event Bus for event mediation. ESBs have been discussed in the ORA SOA Infrastructure document. In a composite architecture that involves both EDA and SOA, the ESB serves as an integral central component that seamlessly integrates these technologies.

Similarly there are other Service infrastructure components such as Service management infrastructure that are leveraged by EDA in a symbiotic relationship between SOA and EDA.

1.1.4 Data Infrastructure

Data infrastructure includes databases, storage, and hardware. EDA requires high performance, distributed data center infrastructure for providing fast response times and a high degree of decoupling. The volume of events may vary significantly based on several factors like the time of the day or market conditions. EDA can benefit from the on-demand capabilities of the grid data infrastructure. EDA leverages shared infrastructure and grid platforms that the enterprise may already have built. Like any other technology, EDA has its own scalability and availability requirements and can benefit from scalable, highly available hardware infrastructure that the enterprise runs.
The ORA EDA Foundation document describes the conceptual view of the Event Driven Architecture. The conceptual view covers the core capabilities of the EDA design-time and runtime infrastructure. Figure 2–1 recaps the conceptual view.

Figure 2–1  EDA Conceptual View

The purpose of this section is to present a logical view of EDA that describes the logical layers and components that provide the capabilities shown in the conceptual view above. Building enterprise class EDA infrastructure requires a good understanding of the architectural capabilities required to fulfill the Real-Time Enterprise (RTE) requirements and what logical components make up the architecture. Chapter 3 maps Oracle products to the logical view described in this section to illustrate how an enterprise EDA infrastructure can be implemented using Oracle technologies.
Figure 2–2 shows the logical layers and components of the EDA infrastructure. Please refer to Appendix A for a complete and detailed EDA logical view. These logical components provide the capabilities listed in Figure 2–1.

The key groupings as shown in the diagram are:

- **Engineering**: covers the design and development tools for implementing EDA infrastructure.
- **Event Producers**: Internal and external producers generate business events.
- **Inbound Adapters**: Inbound Adapters convert the events from the source format to the format understood by the event processors.
- **Streams**: Event Streams are linearly ordered sequence of events. Streams are usually ordered by time.
- **Event Processors**: Event processors process the event based on the predefined rules.
- **Event Cache**: Events may be stored in a temporary cache for availability and performance reasons.
- **Inbound and Outbound Channels**: Channels are inbound or outbound conduits through which the events are transported.
- **Outbound Adapters**: Outbound Adapters deliver the outbound events and provide subscription and filtering capabilities.
- **Consumers**: Consumers subscribe to the interested events and react to them appropriately.
- **Event Bus**: Event Bus or mediation layer provides the components to route and transform the events.
- **Event Monitoring and Management**: The components in this layer deal with monitoring and management of events.
These components are described in detail in this section.

## 2.1 Engineering

EDA Engineering covers the design-time activities of both the producer and the consumer.

From a producer's perspective, the essential tasks involved in EDA engineering are:

- Search and discover the event if it already exists.
- Model and design the events, event flow and event processing networks.
- Develop the event objects.
- Implement the processing logic by means of a query language (e.g. Continuous Query Language - CQL).
- Configure the inbound and outbound channels.
- Test the event processing logic and the event objects.
- Deploy the components for staging and production.
- Publish the event specifications in an enterprise repository for discovery and consumption.

From a consumer's perspective, the essential tasks involved are

- Identify the event to be used. The event may already exist and be operational. Hence, search and discover the event if it already exists.
- Identify the modes of subscribing and receiving the event. This includes the end points, channels and other pieces of necessary information.
- Design and develop the subscription and response code.
- Configure and test the consumer-side application.
- Deploy the consumer-side application.

*Figure 2–3  EDA Engineering*

These engineering tasks require the following basic components as shown in *Figure 2–3*:

- Integrated Development Environment (IDE)
- Event Repository
- Configuration Tools

### 2.1.1 Integrated Development Environment (IDE)

IDEs for EDA provide a number of design-time capabilities to build and deploy the event-driven application components.
**2.1.2 Event Repository**

Event Repository or Event Type Repository is a metadata store that houses the event specifications and other related artifacts. The Event Repository is not a standalone repository dedicated just for managing event artifacts. Rather, it is an enterprise-scoped repository that also manages all other enterprise artifacts. This allows the event types to be associated with the other enterprise solutions such as SOA services, business processes, functional models, and data models.
Figure 2–5  Event Repository

Figure 2–5 shows the Event Repository in the context of the larger enterprise repository. Event related artifacts managed in the repository may include the event interfaces, release plans, event models, test plans, producers & consumers, and operational metrics. Associations and dependencies are also captured in the repository to perform impact analysis or just for documentation purposes. The repository may capture relationships such as the business processes and SOA services triggered by the events or the events produced by the business processes and SOA Services.

The Event Repository is primarily a human interface for event-based asset capture and presentment. The primary objective of the Event Repository is to capture the metadata and not the actual payload. It integrates with other enterprise asset sources like Source Code Management (SCM) tools and file servers for linking to the actual asset payload.

Core capabilities of the Event Repository include the following:

- **Event Lifecycle Management**: Manages the development lifecycle states of the event.
- **Usage Tracking**: Keeps track of the usage of the events.
- **Search and Discovery**: Provides an interface to search the event types and identify the ones that can be used for a given scenario.
- **Version Management**: Maintains the history and versions of the events.
- **Taxonomy**: Classifies the events using business ontology and other categorizations.
- **Dependency Analysis**: Provides an interface to maintain the relationships between event types and other assets. This enables dependency analysis.
- **Event Portfolio Management**: Manages the catalog of events for portfolio planning and release purposes.
2.1.3 Configuration Tools

Configuration tools are used to perform a wide variety of tasks related to EDA development and deployment. The list below captures some of the typical tasks accomplished through the configuration tools.

- Creation of the event processing domain and event servers
- Management of the event servers
- Configuration of the event channels
- Security configuration
- Adapter configuration
- Event Processing Network (EPN) configuration
- Tools for assembly and deployment

2.2 Event Producers

Producers are the systems or components that define the event, and generate the event as the business change happens. Producers are also known by several other names including Event Source, Event Publisher, and Event Provider. This document uses the term "Producer" or "Event Producer" to refer to any of these. Please refer to the ORA EDA Foundation document for a discussion on various types of the Event Producers.

Figure 2–6 Event Producers

From a logical view standpoint, Event Producers are categorized into internal and external producers, as shown in Figure 2–6.

2.2.1 Internal Producers

Events may be produced by a number of internal sources listed below.

- **Applications**: Enterprise applications such as ERP and CRM are one of the primary producers of events. These applications handle several business support transactions every day and it is critical to provide them an interface for event integration.

- **Business processes**: A business process describes a sequence or flow of activities in an organization with the objective of carrying out work. The flow of a process is controlled by events, decisions, triggers, and exceptions, any of which may be influenced by rules or policies. Business processes are designed modularly using sub-processes to support asynchrony. The interaction between sub-processes and the business activities is a key concern in the design of business processes. EDA provides a loosely coupled mechanism for achieving this interaction.

- **SOA Services**: In an Event Driven SOA (ED-SOA), events and SOA services integrate seamlessly to provide capabilities that are otherwise not possible to be offered by them individually. The ORA SOA Foundation document defines various
types of SOA services in an enterprise that interact directly through composition or through a mediation layer. Services in any of these layers may need to publish business events of importance.

- **Portals**: Enterprise Portals are a key producer of business events as they are the primary interface to the business users. User actions are sometimes translated into business events that can be acted upon by the IT systems.

- **Infrastructure Services**: Infrastructure services such as email services, alerts, monitoring and management services may produce important events that may need to be analyzed and correlated to other business events within the enterprise.

- **Dashboards**: Business dashboards consolidate and present business activity monitoring information to the business users. Agile enterprises provide ways for the business users to respond promptly to the business opportunities and threats. The reaction of the business users must be received and transformed into business events as appropriate.

- **Custom Solutions**: Custom solutions are home-grown applications that are built internally by organizations to automate business functionality. Custom solutions are also a primary producer of business events.

- **Event Generators**: Event generators produce events when certain conditions are met with respect to the domain they support. For example, a file event generator produces events based on the changes to the status of resources in the file system. The event generator creates events with metadata useful for processing those events.

- **Sensor Devices**: Sensor devices produce events based on physical changes such as goods movement or flight take-off.

### 2.2.2 External Producers

Events may be produced by entities outside the organization. External market feeds and B2B trading partners are some of the examples of external producers.

- **Market Feeds**: External market feeds such as market interest rates, stock prices, holiday calendar updates, and competitive intelligence are key input to effectively run the business day to day and to stay agile. Market feeds are received by dedicated components in the secure zone and distributed internally for the consumption within the trusted network.

- **Trading Partners**: B2B trading partners may send events to exchange trade information such as order, inventory, and fulfillment information. EDA is a great fit for typical B2B trading partner interactions due to the need for loosely-coupled information exchange and inter-enterprise integration.

### 2.3 Inbound Adapters

Producers may publish an event in their own proprietary format and protocol of choice. However the event processor may not be designed for handling a wide variety of protocols and formats. This problem is solved by introducing the adapter layer that can mediate the differences.
Adapters understand the inbound and outbound protocol, and are responsible for converting the event data into a normalized form that is compatible with the processor. Adapters forward the normalized event data as streams.

Adapters should be built-in for most common protocols such as HTTP and JMS. The HTTP publish-subscribe adapters allow producers and consumers to easily publish and subscribe to an HTTP publish-subscribe server channel either locally or remotely. The JMS adapters allow sending and receiving of messages to and from JMS queues, respectively, without the need for custom development.

### 2.4 Inbound Channels

Channels are conduits that provide a way of sending and receiving events. Among other things, they are responsible for queuing event data until the event processor can act upon it. The event processor removes the event data from the Channel to process it.

Although it is useful to have typed Channels that transmit only a specific type of event, Channels may also carry events of different types. On the inbound side, a Channel may handle one or more event Streams.

The inbound events are sometimes correlated to contextual information from internal sources such as databases. The Stream that models relational data from sources such as RDBMS or a cache is called a "Relation". A Stream has no fixed size and data can only be inserted to it, while a Relation has a fixed size, and data can be inserted, deleted, and updated in it. The rest of the document uses "Streams" to refer to both Streams and Relations.

Although a Cache is technically not a Channel, sometimes event streams may be cached and consumed, which is represented by the Cache box in the diagram. Event Caching is explained in detail later in the next section.

### 2.5 Processors

Event Processors, also called Event Processing Agents, consume normalized event data from the inbound channels and process it using the rules predefined in the form of queries. The processors may generate new events to an output channel or simply execute the response action.
**Figure 2–9  Event Processor Inputs in an EPN**

Figure 2–9 shows the Event Processor and the input channels in the context of an EPN. The Event Processor may draw input from an Inbound Stream or an Event Cache. Event Cache is discussed in the "Event Cache" section later in this document.

There are multiple ways in which a cache can be connected to the processor. Some examples are:

- Cache is connected through a cache adapter.
- Cache is configured for direct access from the Event Processor using event processing queries.

**Figure 2–10  Event Processor**

Figure 2–10 shows the logical components of an Event Processor. These logical components are explained below.

### 2.5.1 Query Processor

Event-driven applications are rule-driven. In EDA, rules are expressed as queries using an event processing language (EPL) such as the Continuous Query Language (CQL). These queries are persisted to a data store and are used for processing the inbound stream of events and generating the outbound stream of events. Queries typically perform filtering and aggregation functions to discover and extract notable events from the inbound event streams. As a result, the number of outbound events is generally much lower than that of the inbound events.
Query processors may provide the following capabilities in addition to simple query processing:

- **Parameterized Queries**: Parameterized queries allow placeholders inside of an EPL query in the form of a question mark. At runtime these placeholders are bound with values and they are then compiled into regular statements.

- **Subqueries**: Query Processor supports both simple subqueries as well as correlated subqueries. In a simple subquery, the inner query is not correlated to the outer query. In correlated subqueries, the inner queries and outer queries are joined using common parameters.

- **Dynamic Event Properties**: Dynamic properties are event properties that need not be known at the statement compilation time. Such properties are resolved during runtime. The idea behind dynamic properties is that for a given underlying event representation, the properties are not always known in advance.

### 2.5.2 Pattern Recognizer

A key capability of the event processor is pattern recognition. The pattern recognizer can look for complex patterns and generate events when they are detected. In order to do it, the following capabilities need to be supported by the pattern recognizer.

- Specification of the pattern to look for
- Identification of the pattern in a stream of events
- Generation of composite event that signifies the occurrence of the pattern

For a more detailed discussion on pattern recognition, please refer to the ORA EDA Foundation document.

### 2.5.3 Direct Handlers

Sometimes an immediate response might be required when a pattern is detected. Direct Handlers allow the response to be implemented and executed within the event processor without the need to publish the event over the outbound channels. An example of a Direct Handler would be a POJO (Plain Old Java Object) that implements certain business logic and configured to be invoked when an event occurs.

### 2.5.4 Extension Cartridge

Extension Cartridge allows the capabilities of the processor to be extended with new components and capabilities. For example, capability to support new languages or language extensions can be added to the processor using the Extension Cartridge mechanism.

### 2.5.5 Analytics Engine

Analytics Engine enables predictive analytics to be done at real-time by combining the event intelligence with the historical perspective from data mining.

### 2.5.6 Splitter

Splitter allows a single event to be split into multiple events for further processing. This is particularly useful when an event is overloaded with logically unrelated information that may not be useful together. Also the potential consumers for these different segments of information may be different.
In such cases, the event content may be split into multiple events so that the consumers can subscribe to the appropriate event type.

2.5.7 Enricher

Events are sometimes enriched with additional details that are needed for processing them. Enricher augments the event with the additional data from various other sources.

2.5.8 Aggregator

Aggregator combines events to create a composite event or an event batch. A composite event aggregates specific fields in the event or combines event fields. Batching allows events to be grouped based on time or count.

2.5.9 Recorder

Recorder records the events flowing through the EPN and persists for future playback. Events are recorded at event sources. Recorder should provide the ability to schedule event recording by specifying start and end times.

2.5.10 Player

Recorded events are played back by the Player in the sequence they were recorded in. The events are played back to the event sink. Event playback allows testing and troubleshooting.

2.6 Outbound Channels

Outbound Channels are event processing endpoints that receive the output from the processors. Outbound channels may handle one or more Event Streams as shown in Figure 2–11. As with the Inbound Channels, although Cache is technically not an Outbound Channel, outbound events may be stored in a cache for availability and performance reasons.

Figure 2–11  Outbound Channels

Figure 2–12 illustrates the flow of events between Processor and Consumers through the outbound channels and Cache.
2.6.1 Streams

Streams are the primary channels that receive and distribute outbound events. Among other things, streams are responsible for queuing event data until the events are delivered.

2.6.2 Event Cache

Applications can optionally publish or consume events to and from a cache to increase the availability of the events and increase the performance of their applications.

A cache can be considered a stage in the event processing network in which an external element (the cache) consumes or produces events. A cache, however, does not have to be an actual stage in the network. Another component can access a cache programmatically using the caching APIs.

The processor may connect to the cache using a direct handler (e.g. a POJO) or directly through the use of the event processing language.

Following list describes some of the factors that should be considered when designing the event cache:

- **Cache size**: How big should the cache be? This really depends on the size and frequency of the events cached.
- **Eviction policy**: The policy to determine which data should be removed from the cache next.
- **Time-to-live**: Answers the question how long the cache data is valid for?
- **Write policy**: Determines how the cache write is managed. (For example, write-through or write-behind)

Figure 2–13 shows the role of an Event Cache in an Event Processing Network (EPN). This is really an aggregated view of the inbound and outbound sides discussed previously. The Event Cache may receive events from other processors and cache them. It may also receive events from a Stream through an Event Listener. Another Processor in the EPN may access the cache data to process the events further in the network. Consumers and other third party applications may access the cache directly for any information of interest.
2.7 Outbound Adapters

The role of outbound adapters is the reverse of the role of inbound adapters. Outbound adapters push out the outbound events using the appropriate protocol. For example, a JMS adapter can be used to publish events to a JMS queue or topic. This is especially useful when consumers want to receive offline events as they can use a JMS queue or JMS topic durable subscription.

Most adapters use a notification based approach to delivering events. Some adapters also support subscription based services where the consumer can subscribe to the events of interest. For example, the HTTP adapters may support subscription services based on Bayeux protocol, which is described in the ORA EDA Foundation document.

2.8 Event Consumers

Consumers or Event Consumers receive and react to the event. Event Consumers are also known as Event Sinks.
Figure 2–15 shows some of the categories and examples of Event Consumers. Consumers may be internal or external to the organization.

2.8.1 Internal Consumers

Most of the internal producers listed in the Event Producer section may also consume events.

- **Applications**: Enterprise applications such as ERP and CRM consume events to react to the business opportunities and threats.

- **Business processes**: Business processes may consume events in multiple ways. Business processes, primary or sub, may be started by significant business events. Business processes may also wait in a quiescent state to be woken up by particular events of interest.

- **SOA Services**: Similar to the Business processes, SOA Services may be invoked by the occurrence of an event. The ORA EDA Foundation document provides more information on such Event-SOA Service integration.

- **Portals**: Enterprise Portals consume events for displaying to the portal dashboards or to react appropriately.

- **Infrastructure Services**: Infrastructure services such as email services, alerts, monitoring and management services may consume events that may need to be analyzed and correlated to other business events within the enterprise.

- **Dashboards**: Business dashboards consolidate and present business activity monitoring information to the business users.

- **Custom Solutions**: Custom solutions are home-grown applications that are built internally by organizations to automate business functionality. Custom solutions consume business events for real-time decision making.

- **BI Analytics**: Business Intelligence (BI) Analytics systems subscribe to the events to receive business insight and provide real-time analytics to the business users.

- **Business Activity Monitoring (BAM)**: BAM consume events for providing insight into business operations. Business Activity Monitoring deserves special attention with respect to EDA due to its critical role in providing insight and allowing the user to respond instantaneously. BAM is explained in detail below.

2.8.1.1 Business Activity Monitoring (BAM)

The objectives of Business Activity Monitoring are to:

- Monitor Events, Business Processes, and SOA Services in real-time. This requires Key Performance Indicators (KPIs) and Service-Level Agreements (SLAs) to be defined, measured and monitored.

- Analyze Events as they occur by correlating Events & KPIs. Identify trends as they emerge and alert users on bottlenecks.

- Act on current conditions to reduce the gap between insight and action through real-time dashboards, process integration and Service integration.

BAM requires the infrastructure to get real-time as well as historical information. The primary source of data is real-time events that help BAM update reports and generate alerts very quickly.

BAM consolidates real-time business insight from various enterprise sources to provide a unified view of the business activities as they are happening. In order to do it, the BAM server should provide open interfaces to the enterprise solutions so that it
can receive the real-time events. BAM accepts a huge volume of updates per second into a memory-based persistent cache that is the center of the BAM architecture.

Any application can send Events using any standard protocol such as Web services or JMS. In addition to integrating real-time information coming out of the Enterprise Solutions, it should be possible to integrate historical data or information coming out of any operational database or data warehouse.

BAM assembles and formats data to generate reports and present them to thin and thick clients. Reports are available in a variety and combination of view types including charts, columnar, cross tab, spreadsheets, Key Performance Indicators (KPIs), lists and more.

2.8.2 External Consumers

External Consumers are Event Consumers that are outside the enterprise boundaries.

- **External Subscribers**: External Subscribers subscribe to the public or restricted events of the enterprise. These events may be market feeds or other information of interest.
- **Trading Partners**: Business to Business (B2B) trading partners may consume trading information such as purchase order submission or fulfillment requests and take appropriate actions required to satisfy the trading partner agreements.

2.9 Event Mediation/Bus

Event mediation is typically performed by the Event Bus. An Enterprise Service Bus (ESB) can be used as the Event Bus.

*Figure 2–16 Event Mediation/Bus*

The key logical components in the Event Mediation stage are:

- **Event Router**: The Router routes the events based on the routing rules. The events are routed based on the header values or content values.
- **Transformation Engine**: Events are transformed into another form or type by the transformation engine.
- **Adapters**: Adapters allow various formats and protocols to be mediated at the bus layer.

2.10 Event Monitoring and Management

As explained in the ORA EDA Foundation document, there are two types of monitoring in an EDA environment. The components of the EDA infrastructure need to be monitored for healthy and efficient operation. But more importantly the business activities must be monitored for identifying opportunities, threats and anomalies. This type of Business Activity Monitoring (BAM) plays a key role in Event Driven Architecture. The focus of this section is the first type of monitoring, which is the system monitoring of the EDA infrastructure. BAM is discussed in the Event
Conference section. Figure 2–17 shows the primary components of Event Monitoring and Management.

**Figure 2–17  Event Monitoring and Management**

2.10.1 Management Console

As EDA infrastructure gets complex, management of it becomes more challenging. This warrants the need for infrastructure components to configure, administer and manage the EDA infrastructure. The Management Console consumes data from the Event Processor, displays it in a useful and intuitive way to the system administrators and operators, and, for specified tasks accepts data that is then passed back to the Event Processor in order to change its configuration. The Management Console also helps to visualize and manage the event infrastructure including security, rules, Event Processing Network configuration, event recording, playback, and the event infrastructure’s operational parameters. The following list summarizes the primary tasks performed by the Management Console.

- Define and view the structure of an EDA Deployment
- Configure EDA resources such as servers
- Manage EDA security information
- Manage the lifecycle of the EDA infrastructure components (start, stop, resume etc.)
- Define and view the Event Processing Networks

2.10.2 Dashboard

The monitoring dashboard provides insight into the EDA infrastructure and applications for monitoring and troubleshooting purposes. It uses text and visual representations to display key monitoring information to identify bottlenecks and to check the performance. For example, the latency and throughput graphs display the amount of time it takes an event to pass through the specified stage or path in the EPN or the number of events passing through, respectively.

2.10.3 Logging and Notification

An important aspect of monitoring the EDA infrastructure is to raise alerts and to log key infrastructure events. The information logged or notified may include the following:

- Policy violations
- SLA exceptions
- Access records
- User actions
- Event volumes
2.11 Security

The ORAEDA Foundation document outlines the general security aspects of Event Driven Architecture and the ORA Security document describes Security architecture in detail. Security, in the context of EDA, deals with securing and protecting the EDA resources and event data. Figure 2–18 shows the primary logical components required for implementing a secure EDA infrastructure.

Figure 2–18  EDA Security

2.11.1 Authentication Service

Authentication is the process of verifying that a user or resource consumer is who he/she claims to be. This is generally accomplished by providing an identifier along with a password, token, or signature that is unique to the user and trusted to be secret.

Authenticating users is a common capability of any business solution. The way in which a user is authenticated varies based on the implementation of security architecture and the degree of stringency one places on proving identity.

Events carry business information that may be very sensitive in nature. EDA requires authentication to ensure that enterprise information is protected. EDA infrastructure should seamlessly integrate with the enterprise security stores and services to authenticate the users when accessing the EDA resources. The degree of sensitivity of the event data dictates how strong the authentication should be (e.g. Weak Vs Strong Vs Multi-Factor).

2.11.2 Authorization Service

Authorization is the process of granting or denying requests by a party (e.g. client or user) to perform actions on a resource. It is generally performed by comparing rights granted to the user with actions the user is attempting to perform. Access decisions have been divided into three topics including coarse-grained, fine-grained, and data field level access control.

EDA resources must be appropriately secured to allow authorized access to the users. Some resources require administrative privileges and some require monitoring privileges. Similarly some of the event information might be accessible only to privileged users. Authorization allows role based access control to the EDA resources and event data.

2.11.3 Audit Service

Auditing provides a record of activities or transactions that have occurred in the system. It may be used for many purposes such as retracing steps of a transaction, quantifying activities in the system, identifying erroneous behavior, or attempting to determine if a specific action did or did not occur.

Auditing is an important concept for security, although is often used outside of the context of security. Without security auditing some attacks can go undetected, such as...
route-force attacks, password hacking, and inappropriate activity perpetrated by insiders.

Auditing Service allows the activities affecting the security of the EDA infrastructure to be captured. Auditing may capture information such as, successful or failed authentication attempts, authorization requests, EDA resource access information, and security information updates.

2.11.4 Role/Credential Mapping

As a result of silo-ed security data, users often have multiple identities. Along with each identity is a password, or proof, that is required in order to authenticate. As user requests traverse boundaries between applications, the user must somehow authenticate in order to proceed. Often the authentication process happens automatically on behalf of the user. The source system locates the appropriate credentials (id and password) for the user and performs authentication. The process of obtaining the appropriate credentials for the target application based on credentials used for the source application is called credential mapping.

Role Mapping is the ability to determine which roles a user is associated with. It is a necessary step for role-based access control. Computing platforms do this by obtaining a list of roles the user is associated with, either via a locally managed store or via a remote (common) store. The use of a common role store is recommended in order to promote information sharing, centralized management and modeling, auditing, etc.

Complex EDA deployments and Event Processing Networks may integrate with a number of internal systems and "plumbing" components such as the Event Bus. In order to seamlessly integrate with the enterprise systems, they need to provide credential mapping capabilities to authenticate the users. Role mapping allows access control to the EDA resources based on a universal identity and role mapping.

2.12 Event Data Management

A variety of data needs to be managed in an Event Driven Architecture. Figure 2–19 illustrates the types of data managed in EDA implementations.

*Figure 2–19  Event Data Management*

The list below briefly describes each of the data types listed in Figure 2–19.

- **Event Metadata**: Event specifications, event metadata, and dependencies are stored by the Event repository.
- **Configurations**: EDA infrastructure configurations and personalization settings are some of the examples of this type of data.
- **Persistent Cache**: Caching improves the availability and performance in EDA. Some applications may require higher level of availability and zero tolerance for information loss. They may benefit from Persistent Cache that saves data to the disks.
- **Event Assets**: Event assets such as EPL/CQL queries, EDN definitions, and EPN specifications need to be managed as well.
- **Event Store**: Recorded events are stored on the disk for future playback.

- **Data Warehouse**: Predictive analytics uses historical information from the data warehouse. The event data is also sent to the data warehouse for historical and data mining purposes.

- **Reports**: Event reports and business activity reports may be stored to be presented to the business users.

- **Monitoring Cache**: The monitoring system may cache information related to the analysis and display of business activities.

- **Security Store**: EDA infrastructure is typically integrated with the enterprise security infrastructure. However, on certain occasions, security information may need to be managed locally for various reasons.

The sheer volume and performance requirements of EDA solutions demand a robust, highly available, and fast data management solution that can scale easily. EDA can benefit from database and storage grids that provide capacity on demand.
Today’s leading organizations are extending their applications to incorporate business events to achieve real-time analysis and response. However, hand-coding event processing solutions requires complex development skills and often fails to achieve business agility. Oracle’s Event-Driven Architecture Solution provides companies across a range of industries, including financial services, telecommunications, retail, government, and manufacturing, the ability to become a real-time enterprise, enabling the build, deployment and management of event-driven architectures through a pre-integrated set of event processing technologies.

Oracle’s Event-Driven Architecture Solution is comprised of best-in-class Oracle Fusion Middleware components and complements the service-oriented model of SOA, providing infrastructure to manage both simple event-based interactions and more complex event analysis for pattern matching in real-time. Oracle’s EDA infrastructure solution consists of the following:

- Complex Event Processing (CEP) engine to handle a high throughput of events with extremely low latency and evaluate patterns in event data
- Business Activity Monitoring (BAM) solution to define and monitor events and event patterns that occur throughout an organization
- Business Rules engine to capture, automate, and flexibly change business policies
- Enterprise Service Bus to connect applications and route messages
- Enterprise Messaging to reliably deliver event messages with configurable Qualities-of-Service
- Integrated caching and data grid technology to scale to enterprise-class event processing use cases
- High-performance Java Virtual Machine to ensure deterministic garbage collection for high-throughput, low latency event processing on a java platform

The Oracle Event-Driven Architecture Solution is a comprehensive, enterprise-class solution for creating, processing, and monitoring events and provides a flexible, declarative environment to rapidly build and adapt event-driven applications.

Oracle EDA Suite improves an organization’s ability to predict change by improving its visibility to happenings in the physical world and business environment in real time. It leverages existing investments by being modular, open, extensible and interoperable; this eliminates the need to remove or replace existing systems as well as supporting incremental deployment and ROI. Finally it extends system and application performance with a lightweight, Java approach to event-driven application deployment.
As markets become more competitive, companies must adapt their applications and IT systems to be more responsive to events in real time. Oracle Event-Driven Architecture Suite provides a flexible, declarative environment for rapidly building and adapting event-driven applications. Oracle EDA Suite is a key part of today’s intelligent enterprise, which expands beyond the service-interaction model of SOA to manage event-based interactions and complex event analysis in real time. With personalized dashboards and custom reporting and alerting capabilities, Oracle EDA Suite empowers both IT and business users to stay on top of their businesses.

The solutions that produce or consume events are implemented using one or more of the Oracle middleware products such as Oracle SOA suite or Oracle BPM suite. Oracle Applications may also produce or consume events.

Oracle’s complete SOA offering, Oracle SOA Suite, is an integrated, best-of-breed suite of products that helps rapidly design and assemble, deploy and manage, highly agile and adaptable business solutions. This standards-based, hot-pluggable infrastructure interoperates with the existing IT investments lowering the upfront costs.

This section maps the Oracle technologies to the EDA logical model. Figure 3–1 shows the mapping of Oracle products to the logical model.

**Figure 3–1  EDA Oracle Product Mapping**

The products required to implement an EDA infrastructure are part of the Oracle EDA, SOA, and BPM Suites. These suites include the following products shown in the product mapping view:

- Oracle Complex Event Processing (CEP)
- Oracle Business Activity Monitoring (BAM)
- Oracle Service Bus (OSB)
- Oracle JRockit
- Oracle Business Rules
Oracle Complex Event Processing (CEP)

In addition, the following products play a key role in the EDA solutions.

- Oracle BPEL Process Manager
- Oracle Enterprise Repository and Oracle Metadata Repository (MDS - part of Oracle SOA Suite) manage event metadata.
- Oracle Coherence provides highly distributed cache capability
- Oracle BI suite provides the data warehousing and analytics capabilities.
- Oracle security products, Oracle Access Manager (OAM), Oracle Entitlement Server (OES), and Oracle Platform Security Services (OPSS) provide security capabilities.

3.1 Oracle Complex Event Processing (CEP)

Oracle CEP is a Java server for the development and deployment of high-performance event driven applications. It is a complete solution for building applications to filter, correlate and process events in real-time so that downstream applications, service oriented architectures and event-driven architectures are driven by true, real-time intelligence.

Oracle CEP uses a lightweight Java application container based on Equinox OSGi, with shared services, including the Oracle CEP Service Engine, which provides a rich, declarative environment based on Oracle Continuous Query Language (Oracle CQL), a query language based on SQL with added constructs that support streaming data, to improve the efficiency and effectiveness of managing business operations.

Oracle CEP supports ultra-high throughput and microsecond latency using JRockit Real Time and provides Oracle CEP Visualizer and Oracle CEP IDE for developer tooling for a complete real time end-to-end Java Event-Driven Architecture (EDA) development platform.

Oracle CEP has the capability of deploying user Java code (POJOs) which contain the business logic. Running the business logic within Oracle CEP provides a highly tuned framework for time and event driven applications.

Oracle CEP provides organizations with a complete "top-down" solution for designing, defining, developing, and implementing complex event processing applications that not only meet business requirements but perform to the highest levels of enterprise expectation. Built on industry-standards including ANSI SQL, Java, Spring DM and OSGi, Oracle CEP provides an open architecture for sourcing, processing, and publishing complex events throughout the enterprise. With both a visual development environment as well as standard Java-based tooling, Oracle CEP ensures that the IT team can be developing event-driven applications without the hurdle of specialized training or unique skill-set investment.

3.1.1 Oracle CEP Development Environment

Oracle CEP Development Environment is a set of plugins for the Eclipse IDE designed to help develop, deploy, and debug applications for Oracle CEP.

- Project creation wizards and templates to quickly get started building event driven applications
- Advanced editors for source files including Java and XML files common to Oracle CEP applications
- Integrated server management to seamlessly start, stop, and deploy to Oracle CEP instances all within the IDE
· Integrated debugging
· Event Processing Network (EPN) visual design views for orienting and navigating in event processing applications

3.1.2 Oracle CEP Architecture

Internally, Oracle CEP is a lightweight, modular application server that leverages a number of cutting-edge technologies. Figure 3–2 is a high-level diagram of the Oracle CEP software stack. It shows Oracle CEP running on Oracle’s JRockit Real Time JVM which offers Deterministic Garbage Collection (DGC). This is appropriate for applications with demanding requirements for deterministic behavior and low latency. Oracle CEP does not require DGC, however, and can also be run using regular JRockit.

*Figure 3–2  Oracle CEP High Level Architecture*

![Figure 3–2  Oracle CEP High Level Architecture](image)

*Figure 3–2* illustrates that Oracle CEP uses OSGi to provide the core of its modular architecture. The important thing to note is that server subsystems, such as the logging and configuration subsystems, as well as developer-written applications are packaged as OSGi modules or OSGi bundles.
Figure 3–3 shows the architecture layers and components of Oracle Complex Event Processing. At the bottom of the stack is the OSGi Service platform that includes the OSGi framework, configuration admin, JMX, and Blueprint Service (Dependency Injection and Inversion of Control that allows external configuration of bundle dependencies) modules. The next layer, the Oracle Core Engine, includes Work Manager, Data Sources, and Spring Dynamic Module. The Oracle CEP Work Manager is optimized for EDA applications. It is latency driven as opposed to throughput driven and avoids thread context switches to improve performance. The top layer is the event processing layer with components such as the Channel, Adapter SDK, and CQL Engine.

3.1.3 Oracle CEP High Availability Architecture

Oracle CEP supports an active-active HA architecture. The active-active approach has the advantages of high performance, simplicity, and short failover time relative to other approaches. An Oracle CEP application that needs to be highly available is deployed to a group composed of two or more Oracle CEP server instances running in an Oracle CEP cluster. Oracle CEP will choose one server in the group to be the active primary. The remaining servers become active secondaries. It is not possible to specify the server that will be the initial primary as it is chosen automatically.

The number of active secondaries depends, of course, on the number of servers in the group hosting the application. If the group contains n server instances then there will be n-1 secondary instances running the application. The number of secondaries in the group determines the number of concurrent server failures that the application can handle safely. A server failure may be due to either a software or hardware failure which effectively causes termination of the server process. Note that most applications require just one or possibly two secondaries to ensure the required level of availability.

During normal operation, prior to a failure occurring, all server instances hosting the application process the same stream of input events. The active primary instance is
responsible for sending output events to the downstream clients of the application. The active secondary instances, on the other hand, typically insert the output events that they generate into an in-memory queue. Events are buffered in the queue in the event that they are needed to recover from a failure of the active primary instance. Queued events are discarded, or “trimmed”, when it is determined that they are no longer needed for recovery.

Failure of the active primary instance, results in failover to an active secondary instance. The secondary instance becomes the new active primary and takes over the responsibility of sending output events to downstream clients. The new active primary will begin by sending the output events that are currently contained in its output queue(s) before sending any new output events that are generated following failover.

Developers make the Oracle CEP applications that they write HA-capable by adding additional components to the application’s event processing network (EPN). The EPN components that enable HA functionality are termed “HA adapters” because there is a 1-1 correspondence between them and the regular output adapters that send the application’s output events. An HA adapter can be thought of as a proxy stage in the EPN which implements HA behavior, such as queuing output events, and delegates to the regular output adapter for sending events to downstream clients.

### 3.2 Oracle Business Activity Monitoring (Oracle BAM)

Oracle Business Activity Monitoring (Oracle BAM) gives business executives the ability to monitor their business services and processes in the enterprise, to correlate KPIs down to the actual business process themselves, and most important, to change business processes quickly or to take corrective action if the business environment changes.

![Oracle BAM Logical View](image)

**Figure 3–4 Oracle BAM Logical View**

Figure 3–4 shows the logical view of the Oracle BAM architecture. BAM requires the infrastructure to get real-time as well as historical information. The primary source of data is real-time events that help BAM update reports and generate alerts very quickly.

BAM consolidates real-time business insight from various enterprise sources to provide a unified view of the business activities as they are happening. In order to do it, the BAM server should provide open interfaces to the enterprise solutions so that it can receive the real-time events. The Enterprise Integration Framework, or simply the
Integration Framework, provides the abstract layer to interface with the Enterprise Solutions such as Business Process Management systems, Complex Event Processing applications, Enterprise Service Bus, and Message Queues. The Integration Framework lets BAM connect to enterprise information sources such as database server, flat files, and XML sources.

The Event Engine monitors complex changing conditions in the data and the system in real-time and based upon user-defined rules. It takes a variety of actions in response to those changes, including notifying the appropriate user with an alert or reports. This allows users to effectively monitor their business for key conditions and sends the right information to the right person at the right time. The Event Engine may also raise outbound events or perform an action in certain cases. The BAM Event Engine requires the same kind of capabilities offered by the Event Processor discussed earlier and it is logical to use the Event Processor for the implementation of the Event Engine.

BAM accepts a huge volume of updates per second into a memory-based persistent cache that is the center of the BAM architecture. The Cache Manager manages the persistent cache. Cache Manager should support configuring various third party cache implementations.

Any application can send Events using any standard protocol such as Web services or JMS. In addition to integrating real-time information coming out of the Enterprise Solutions, it should be possible to integrate historical data or information coming out of any operational database or data warehouse. The Data Manager logical component indicated in Figure 3–4 provides the interface to the data sources and metadata.

The Reports Manager running on the Reports Server assembles and formats data to generate reports and present them to thin and thick clients. Reports are available in a variety and combination of view types including charts, columnar, cross tab, spreadsheets, Key Performance Indicators (KPIs), lists and more.

The Reports Manager obtains a "snapshot" of the most current data and establishes a change stream. Using the snapshot, it creates an initial display and sends it to the client. Once the client has rendered the initial display, it continually processes data as it changes, and integrates those changes into the live display-allowing for up-to-the-second information delivery. The Reports Manager may also cache the report data to provide faster access to reports and to offload the burden on the BAM server.

User interaction is typically supported through several channels including email, mobile, dashboards, and application interfaces. The reports and alerts are formatted appropriately for the given channel.

### 3.3 Oracle Service Bus (OSB)

Oracle Service Bus transforms complex and brittle architectures into agile integration networks by connecting, mediating, and managing interactions between services and applications. Oracle Service Bus delivers low-cost, standards-based integration for mission critical SOA environments where extreme performance and scalability are requirements.

The category of infrastructure identified as the Service Bus represents many capabilities including:

- **Mediation** - the ability to connect consumers and producers that may use different transports, protocols, message formats, error codes, etc.
Oracle JRockit Real Time (JRRT)

Message mediation and Transformation - the ability to transform messages from one format to another, aggregate data elements, and enhance message content based on business rules

Routing - the ability to route requests to different endpoints at runtime based on mechanisms such as message content, business rules, security constraints, etc.

Composition - the ability to execute multiple steps, possibly including multiple back end requests, based on configuration, to fulfill an inbound service request

Oracle Service Bus (OSB) is designed to meet the needs of an enterprise-class service bus. In addition to the Service Bus features, OSB also provides useful monitoring, management, and security capabilities. This makes it applicable to these SOA Infrastructure components as well.

The OSB acts primarily as an intermediary that takes in messages, processes them to determine where to route them, and transforms them as specified. It receives messages through a transport protocol such as HTTP(S), JMS, File, FTP, and so on, and sends messages through the same or a different transport protocol. Message response follows the inverse path. The message processing by OSB is driven by metadata specified as the message flow definition, which is created at design time, for a proxy service in the OSB Configuration.

OSB is policy driven. It enables you to establish loose coupling between service clients and Services while maintaining a centralized point of control and monitoring.

The following features provided by OSB provide value to the SOA environment:

- Routing
- Transformations
- Mediation
- Message Flow Modeling
- Error Handling
- Quality of Service

3.4 Oracle JRockit Real Time (JRRT)

Oracle JRockit Real Time (JRRT) provides lightweight, front-office infrastructure for low latency, event-driven applications. For companies in highly-competitive environments where performance is key and every millisecond counts, JRRT provides the first Java-based real-time computing infrastructure.
Figure 3–5  JRockit Architecture

Figure 3–5 shows the architecture of JRockit. The components shown in the figure are described below.

- I/O handles communication with files, databases and network.
- Memory management is concerned with things like garbage collection, when the JVM reclaims unused memory, and finding the optimal heap size for an application.
- Threads management schedules threads, handles synchronization and locks.
- The Java model takes care of very java specific areas like reflection and class loading.
- In Code generation the JVM translates the java code to assembler code that runs directly on the target operating system. JRockit JVM will also detect and perform possible optimizations of the application code.
- The external interfaces and monitoring/management are used to get information directly from inside the JVM, and to control some of it’s runtime features, used by, among others, JRockit Mission Control.
- JRockit Mission Control (JRMC) is a multi functional tool suite that allows users to manage, monitor and profile their applications. Designed for low overhead it can be used in production environments. Oracle JRockit Mission Control consists of three different tools, the management console, the runtime analyzer and the memory leak detector.

### 3.4.1 Deterministic Garbage Collection (GC)

Most EDA solutions require very high throughput, faster response times and predictable performance. With JRRT one can request a pause time SLA as low as 1 ms to achieve predictable latency. The key objective of JRRT is deterministic GC pause time. Finer control of GC behavior adds workload to the VM. Optimizing for low-latency with determinism typically reduces overall throughput. General JRockit optimizations benefit both latency and throughput while the impact on throughput is highly dependent on the application. It is important to understand that what is offered is determinism in pause times, not maximum throughput.
3.5 Oracle Business Rules

Oracle Business Rules is a high performance lightweight business rules product that addresses the requirements for agility, business control, and transparency. It is part of the Fusion Middleware stack and integrates seamlessly across the entire Oracle SOA Suite and BPM Suite stack. It is also a core component for present and future Oracle Fusion Middleware and Fusion Applications products.

Figure 3–6 Oracle Business Rules

The components of the Oracle Business Rules product include:

- **Rule Author**: Rule Author is a web based graphical authoring environment that enables creation of business rules using a click and select user experience.

- **Rules Engine**: Rules engine is an inference capable Rete rules engine. The Rete algorithm is an efficient pattern matching algorithm for implementing production rule systems.

- **Rules Repository**: The Rules repository enables rules to be organized in rulesets and rulesets organized in dictionaries. It also supports versioning of dictionaries. In current releases, the repository may be file based or webDAV based. In addition, APIs are available to plug in any desired repository. Oracle MDS is used as the repository consistent with all Oracle middleware and applications products.

- **Rules SDK**: The Rules SDK provides complete access to the Rule Repository and is designed to facilitate creation of rule authoring environments. The Rule Author itself uses the SDK. It is also used by the Workflow Application in Oracle BPEL PM to provide workflow specific authoring of rules. It may be used to build any custom authoring environments.

- **Rules Language (RL)**: Oracle uses a Java based language called Rules Language, or commonly just RL. Typically, RL would be used directly by users only for writing supporting functions. The Rule Author abstracts away rest of RL from users.

- **Decision Service**: Decision Service enables Oracle business rules to be invoked as a web service from BPEL or other Web Service clients. The Decision Service tooling in Oracle BPEL PM provides seamless integration between BPEL and Business Rules.
3.6 Oracle BPEL Process Manager

BPEL is the standard for assembling a set of discrete services into an end-to-end process flow, radically reducing the cost and complexity of process integration initiatives. Oracle BPEL Process Manager offers a comprehensive and easy-to-use infrastructure for creating, deploying and managing BPEL business processes.

As shown in Figure 3–7, BPEL provides a platform and capabilities to build versatile Services with partner, services, system and people integration. BPEL PM’s built-in integration services enable developers to easily leverage advanced workflow, connectivity, and transformation capabilities from standard BPEL processes. These capabilities include support for XSLT and XQuery transformation as well as bindings to hundreds of legacy systems through JCA adapters and native protocols. Human workflow services such as task management, notification management, and identity management are provided as built-in BPEL services to enable the integration of people and manual tasks into BPEL flows. The extensible WSDL binding framework enables connectivity to protocols and message formats other than SOAP. Bindings are available for JMS, email, JCA, HTTP GET, POST, and many other protocols enabling simple connectivity to hundreds of back-end systems.

BPEL PM supports several open standards in addition to BPEL4WS. These standards include WSIF, WSDL, SOAP, UDDI, JMS, WS-ReliableMessaging, JCA, XQuery, XPath, XSLT and WSIL.

3.7 Oracle Enterprise Repository (OER)

Oracle Enterprise Repository serves as the core element to the Oracle SOA Governance and metadata management solution. Oracle Enterprise Repository provides a solid foundation for delivering governance throughout the development lifecycle by acting as the single source of truth for information surrounding assets and their
dependencies. Oracle Enterprise Repository provides a common communication channel for the automated exchange of metadata and asset information between consumers, providers, policy decision points, and additional governance tooling. It provides the visibility, feedback, controls, and analytics to deliver business value.

3.8 Oracle Coherence

Coherence is an essential ingredient for building reliable, high-scale EDA applications. Coherence provides all of the necessary capabilities for applications to achieve the maximum possible availability, reliability, scalability, and performance.

Coherence provides replicated and distributed (partitioned) data management and caching services on top of a reliable, highly scalable peer-to-peer clustering protocol. Coherence has no single points of failure; it automatically and transparently fails over and redistributes its clustered data management services when a server becomes inoperative or is disconnected from the network. When a new server is added, or when a failed server is restarted, it automatically joins the cluster and Coherence fails back services to it, transparently redistributing the cluster load. Coherence includes network-level fault tolerance features and transparent soft re-start capability to enable servers to self-heal.

By clustering the application’s objects and data, Coherence solves many of the difficult problems related to achieving availability, reliability, scalability, performance, serviceability and manageability of EDA solutions.

3.9 Oracle Business Intelligence Suite

Oracle Business Intelligence Enterprise Edition (OBIEE) is a comprehensive Business Intelligence platform that delivers a full range of analytic and reporting capabilities. Designed for scalability, reliability, and performance, Oracle Business Intelligence Enterprise Edition delivers contextual, relevant and actionable insight to everyone in an organization, resulting in improved decision-making, better-informed actions, and more efficient business processes. Oracle also provides the industry’s only multi-sourced BI applications, as well as market-leading performance management applications that are powered by this BI platform. OBIEE provides the following capabilities:

- Comprehensive BI functionality built on a unified infrastructure: Reduce cost and increase productivity with a common infrastructure for producing and delivering enterprise reports, scorecards, dashboards, ad hoc analysis, and OLAP analysis
- Powerful user experience: Increase user adoption with a powerful, task-oriented navigation framework. Features such as rich visualization, interactive dashboards, a vast range of animated charting options, OLAP style interactions and innovative search and actionable collaboration capabilities deliver an unrivaled end-user experience
- Unified business model: Ensure that all metrics, calculations and definitions are uniformly consistent to provide better alignment and visibility across the organization
- Hot-Pluggable: Integrate with all popular extraction, transformation and load tools, databases, business applications, application servers, security tools, enterprise portals, and desktop tools
- Best-in-class scalability, reliability and performance: Optimizes performance while simplifying systems configuration and lifecycle management. An integrated
systems management console provides superior scalability, high availability, and security benefits, while making upgrades and systems management effortless.

3.10 Oracle database and Oracle Real Application Clusters (RAC)

Persisting and managing the data is a fundamental requirement of any infrastructure. EDA infrastructure requires high performance databases with very high availability requirements as data access cannot become a bottleneck. Oracle database is a proven and market leading database that offers sophisticated clustering and management features.

Oracle Real Application Cluster (RAC) provides scalability and high availability at the database tier. The database tier can be horizontally scaled by adding database server instances that access the storage grid. The RAC instances can be configured for load balancing or failover based on the specific needs of the application.

Fast Application Notification (FAN), is a feature of Oracle Real Application Clusters (RAC) that further differentiates it for high availability and scalability. FAN enables the automated recovery of applications when cluster components fail. The RAC HA framework provides notifications of any change in the cluster configuration. Applications can subscribe to events and react quickly so their users can immediately take advantage of additional resources and are unaffected (or minimally affected) by a reduction in available resources.

3.11 Security Products

The following security products provide security-related capabilities required for implementing EDA solutions.

- **Oracle Access Manager (OAM)** - OAM provides an identity management and access control system that is shared by all applications. It offers a centralized and automated single sign-on (SSO) solution for managing who has access to what information across IT infrastructure.

- **Oracle Identity Manager (OIM)** - OIM is a user provisioning and administration solution that automates the process of adding, updating, and deleting user accounts from applications and directories; and improves regulatory compliance by providing granular reports that attest to who has access to what resources.

- **Oracle Entitlements Server (OES)** - OES externalizes and centralizes fine-grained authorization policies for enterprise applications and Web Services.

- **Oracle Platform Security Services (OPSS)** - Oracle WebLogic Server is the de facto application server for Oracle Fusion Middleware 11g. With the release of Oracle Fusion Middleware 11gR1, Oracle combined BEA’s internal security framework used in products like WebLogic Server and Oracle Entitlements Server (OES), with Oracle Fusion Middleware’s security platform, Java Platform Security (JPS), into one complete security framework known as Oracle Platform Security Services (OPSS). OPSS builds upon Java SE and Java EE security and provides an abstraction layer in the form of standard Java APIs, based on the Java Community Process, that insulate developers from security and identity management implementation details.
This section provides an example of how the EDA products mapped in the previous chapter might be deployed to physical hardware. A network topology format is used to illustrate where products are most likely to be deployed in terms of network tiers.

A number of factors influence the way products are deployed in an enterprise. For instance, load and high availability requirements will influence decisions about the number of physical machines to use for each product. Federation and disaster recovery concerns will influence the number of deployments and failover strategy to use. In addition, deployment configurations and options may vary depending on product versions. Given these and other variables it is not feasible to provide a single, definitive deployment for the products, or to guarantee the deployment scenarios presented in this chapter will work for a particular version. The scenarios presented here are meant to be used as an example. Please consult product documentation for further deployment information.

4.1 EDA Deployment

Figure 4–1 shows a typical EDA deployment using the Oracle products.
Typically RFID readers and sensor devices use some kind of Edge Server to capture and rationalize the edge events and pass them on to the event infrastructure using a messaging platform such as JMS. The example shows a clustered JMS Server that provides events to the Complex Event Processing.

Central to the architecture is the Event Bus, which is implemented using an Enterprise Service Bus (ESB) such as Oracle Service Bus. The Event Bus provides several capabilities including routing, transformation, and protocol mediation. The ESB connects various infrastructure components including the Event infrastructure, SOA Service infrastructure, and BAM infrastructure. Hence, the ESB should be designed to be Highly Available to handle high volume of requests and to avoid single point of failure (SPOF). In a distributed EPN events may flow back and forth through the Event Bus. It is important to decide when the Events should be routed through the Event Bus. Using the Event Bus as a flow through component for all types of events would flood it with a large number of fine grained ordinary events. Most CEP applications have low latency requirements that cannot be met with this sort of a design. Hence, only coarse grained notable and transactional events should generally be routed through the Event Bus.

The Enterprise Repository manages the metadata for the Event and Service assets. It provides web based interface for adding and managing Event assets. Since the repository is not a runtime component, it may not be necessary to make it highly available by clustering the server.

The Event Processing nodes are clustered for high availability and scalability. The event processing is implemented using Oracle Complex Event Processing (OCEP). The event processing node may receive the inbound events from a number of sources including the JMS queues, ESB, and the business solutions such as Business Processes, SOA Services, or custom applications. There are adapters available to connect various protocols to the event processing nodes. The outbound events are received by the consumers, event bus, or BAM servers. The outbound events may also be cached for availability and performance.

The Cache shown in the diagram uses Oracle Coherence. The cache is typically distributed for high availability and the data is synchronized across the nodes. Although it is shown deployed on separate servers, some enterprises may not require dedicated servers for the cache. The cache may be co-deployed with the event processing component.

The Business Activity Monitoring (BAM) server is also deployed in a highly available, multi-server configuration. It is connected to the Event Processing nodes and the ESB to receive key business events to track and display the KPIs. The BAM report server is shown deployed separately. The business users may access the reports through the reports server. Although not shown in the diagram, the BAM server may also receive event inputs directly from other sources such as the business processes or SOA Services.

Almost all of the components shown in the figure require highly available database for persisting data. Oracle database and Oracle RAC are shown as the database solution for the components. These components require separate physical or logical database schemas in the database.

### 4.2 Scalability and High Availability

As event-driven architecture and complex event processing have become prominent features of the enterprise computing landscape, more and more enterprises have begun to build mission-critical applications using EDA. Today, mission-critical EDA solutions can be found in many different industries. For example, EDA is being used...
in the utilities industry to make consumption more efficient by allowing them to react instantaneously to changes in demand for energy. EDA is being used in the financial industry to detect potentially fraudulent transactions as they occur in real time. The list of mission-critical CEP applications continues to grow.

The use of EDA to build mission-critical applications has led to a need for Oracle CEP applications to be made scalable, highly available, and fault-tolerant. The next section discusses scalable and highly available deployment of EDA infrastructure.

4.3 Scalable Deployment

In a typical IT deployment scalability involves distributing load by creating additional processing units that process part of the load. Most systems can be scaled by balancing the load based on a certain criteria. For example, web-based requests may be distributed across multiple servers based on the end-user making the request. As long as all the requests from the same user go to the same processing unit, the integrity of the system is protected and the solution can function normally.

However, the nature of EDA poses some challenges that make scalability more complex. The primary goal of EDA is to correlate inbound events to produce high-value, notable events. This means that all the related events must be processed by the same server or servers that act as the same unit.

Figure 4–2  EDA Scalability Options

Figure 4–2 shows couple of options to scale the EDA deployments. The first option uses a shared database that stores the stream. Both servers use the persisted stream for processing the events. Most EDA solutions also demand very fast response times; hence, most EDA implementations process the events in memory. Persisting the event stream in a database for sharing across multiple servers would significantly slow down the performance of the solution.

The second option shows an in-memory distributed cache that stores the event stream and replicates the state in real-time. This is obviously a better choice than the previous
The third, and most logical option shown in Figure 4–3, partitions an input event stream and processes events in parallel at the point of event ingress, within the Event Processing Network (EPN), or both. The inbound event stream is split into multiple event streams that are logically separated for processing. These inbound streams (IS) are processed individually by different processors and the outbound streams are consolidated for further processing.

The inbound event stream must be divided in such a way that all events required to process the queries in each processor are grouped in the same stream. If there are situations that require correlation across the streams, then the streams should either be consolidated or that correlation must be processed in a subsequent EPN step.

One should plan to use scalability and parallel processing as early in the event processing sequence as possible. For example, parallel process high-volume, low-value events to extract the typically low-volume, high-value events the application depends on.

Because scalability often involves deploying an application to multiple servers, it is advantageous to also consider high availability options when designing the EDA solution for scalability. Input stream partitioning and parallel processing impose important restrictions on solution design, such as preserving event order and carefully managing the use of multi-threading.

4.4 Highly Available EDA Deployments

High Availability (HA) is as important as, if not more so than scalability. HA measures the uptime of the EDA infrastructure. An important issue with EDA is the time it takes to failover and continuing to process the events from the point it failed. It is important to define the QoS requirements in order to design HA properly for a given architecture. It is quite possible to run into one of the following issues with HA deployments.

- Missing events: Due to latency, some of the events may be lost during failover.
Duplicate events: Sometimes the same events may be processed more than once causing duplicate events.

Miscalculated events: Due to changes in order or missing inbound events, the outbound events may not be derived properly.

When choosing an HA design, an architect should consider what is more important to the enterprise. If missing events is not acceptable, a configuration that does not allow events to be lost should be chosen. Similarly if an enterprise prefers duplicate events over performance, then the right HA configuration must be chosen accordingly.

EDA HA deployments may use an active-active or active-passive configuration. Choosing the right configuration is a tradeoff between performance, cost and QoS.

### 4.4.1 HA Deployment: Active-Active

As the name implies, active-active systems employ primary and secondary servers that are active. The secondary servers are also known as "hot" standbys. In this mode, each server is processing an identical stream of events, regardless of whether the results of that processing are actually used or not. Figure 4–4 contains a high-level view of the active-active architecture where each server produces an identical stream(s) of output events.

Since all servers are processing identical streams, only one output should be directed to the outbound channel at any given point of time. This is achieved through a set of coordinating HA adapters that control the outbound events. These HA adapters are aware of which server is the primary and when it fails. The adapters associated with the secondary servers queue up the outbound events until a point when the primary fails. When the primary server fails, one of the HA adapters in the secondary servers takes over and starts sending the events to the outbound channel.

**Figure 4–4  Active-Active HA Deployment**

The two main advantages of an active-active setup are performance and simplicity. The system has good performance during normal operation and at failover time because a hot standby has little processing to do in order to take over from a failing primary. The state of the standby should reflect that of the old primary since it has processed the same set of events and the only impact at failover is actually detecting that the old primary has failed and synchronizing the new primary with the output.
state of the old. The system is also simple because failover does not require any state replication between servers.

In EDA, fast failover is essential since it is often expected to perform near real-time processing of high volume event streams. The downsides of active-active systems are that it can be difficult to build state for newly started servers, and the hardware resource requirements are high because of the redundant processing involved.

### 4.4.2 HA Deployment: Active-Passive

In active-passive systems, backups are not processing the incoming event stream; instead they are expected to take over from a failed primary through some kind of state replication or backup.

Upstream backup is an active-passive HA configuration. The incoming event stream is saved so that it can be replayed to secondary servers at failover. Figure 4–5 presents a high-level view of the upstream backup HA architecture.

*Figure 4–5  HA Deployment Active-Passive*

Saving the event stream has the advantage of not putting significant performance burden on the primary and is relatively simple to implement since an understanding of processor’s state is not required. The state can be thought of as being implicitly held by the saved event stream.

However, saving the incoming event stream can be costly at high event rates; and the overhead and complexity for the primary is not zero since it needs to keep a record of both where it has processed to in the event stream and which events need to be processed in order to affect the output.

### 4.5 Scalable and HA Deployment

*Figure 4–6* shows how to combine the concepts discussed in scalable deployment and HA deployment to design a configuration that provides best of both worlds.
In this configuration, the inbound stream is divided into multiple disjoint streams that are processed by different scalable server groups (SSG). The servers within the SSG process the inbound events in parallel, producing the outbound events. The outbound events are controlled by coordinating HA adapters that queue the output in the secondary until the primary functions normally. The outputs from all the SSGs are aggregated to create a combined stream that is published through the outbound channel.
As traditional enterprises transform to Real-Time Enterprises (RTE), they require IT infrastructure that is capable of supporting the agility and business insight requirements. The addition of EDA to the existing IT infrastructure adds precisely the capabilities that enable this transformation. EDA also complements and enhances the existing technologies.

Architecting an EDA infrastructure should not be an isolated exercise; rather, it should be done in the context of the overall IT infrastructure. Other technology strategies like SOA and BPM benefit from and contribute to EDA. Complex Event Processing (CEP) and Business Activity Monitoring (BAM) play a key role in the implementation of EDA infrastructure. Together, they create an IT ecosystem that can support the needs of a Real-Time Enterprise that not only detects opportunities and threats to business in a timely manner but can also close the insight to action gap by responding in time.

ORA EDA ETS Foundation and Infrastructure documents describe the EDA Reference Architecture to design and deploy EDA infrastructure and solutions in an architecturally consistent fashion. This document discussed a number of views including logical, deployment, and product mapping views. These views illustrated the logical components involved in EDA infrastructure and how they can be implemented using Oracle technologies.

Mission-critical applications also demand scalability and high availability. A number of options are available to create an architecture that meets the QoS, cost, and performance requirements of the customer. The right architecture should be chosen based on a tradeoff analysis between these factors. Large organizations may need to think creatively to design a solution that would fit their organizational and IT structures. Centralized architecture might work better for some organizations while federated architecture might work better for the others. The key is to design an EDA infrastructure that caters to the business units in the most effective way.
ORA EDA Infrastructure document is part of the EDA perspective of the Oracle Reference Architecture. This document refers to and builds upon a number of other documents in the IT Strategies from Oracle (ITSO) series. These documents include but not limited to the following.

- Oracle Reference Architecture Security Document
- Oracle Reference Architecture Management and Monitoring document

**ORA Security** - The ORA Security document describes important aspects of security including identify, role, and entitlement management, authentication, authorization, and auditing (AAA), and transport, message, and data security.

**ORA Monitoring and Management** - This document provides a reference architecture for designing a management and monitoring framework to address the needs for the modern IT environment.

Oracle product documentation and whitepapers contain a wealth of information on this topic. Please consult the Oracle SOA Suite in the Oracle documentation library. It includes Oracle Complex Event Processing (CEP) and Oracle Business Activity Monitoring (BAM) that are the primary implementation technologies of Event Driven Architecture.

- [http://download.oracle.com/docs/cd/E12839_01/soa.htm](http://download.oracle.com/docs/cd/E12839_01/soa.htm) - Oracle SOA Suite documentation
- [http://download.oracle.com/docs/cd/E13157_01/wlevs/docs30/index.html](http://download.oracle.com/docs/cd/E13157_01/wlevs/docs30/index.html) - Oracle Complex Event Processing (CEP) documentation

A number of other external resources are available on EDA. EDA is a topic that is being researched by academies as well as industry analysts. The following list provides some academic and analyst links that were referred in this document.

- Michelson, Brenda, Patricia Seybold Group - Event Driven Architecture Overview
- Chandy, Mani and Schulte, Roy - What is Event Driven Architecture (EDA) and Why does it matter?
http://complexevents.com/ - Complex Event Processing - Applications, products, research, and developments in event processing


