Oracle® Application Server

Concepts

10*g* Release 2 (10.1.2)

Part No. B13994-01

November 2004

ORACLE®

Oracle Application Server Concepts, 10*g* Release 2 (10.1.2)

Part No. B13994-01

Primary Author: Theresa Robertson

Contents

3 Portal Applications

4 Wireless Applications

5 Oracle Business Intelligence

6 Oracle Application Server Integration

7 Oracle Application Server Infrastructure

List of Figures

List of Tables

Send Us Your Comments

Oracle Application Server Concepts, 10*g* Release 2 (10.1.2)

Part No. B13994-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appserverdocs_us@oracle.com
- FAX: 650-506-7375 Attn: Oracle Application Server Documentation Manager
- Postal service:

  Oracle Corporation
  Oracle Application Server Documentation
  500 Oracle Parkway, M/S 1op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This preface contains these topics:

Intended Audience

Oracle Application Server Concepts is intended for anyone with an interest in Oracle Application Server or the latest Internet technologies.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Organization

This document contains:

Chapter 1, "Introduction to Oracle Application Server"

This chapter provides an overview of application servers and Oracle Application Server.

Part I, "Development"

Chapters in this part discuss application development topics.

Chapter 2, "J2EE, Web Services, and Internet Applications"

This chapter describes the Java 2 Platform, Enterprise Edition (J2EE) and Internet application development and deployment environment in Oracle Application Server.

Chapter 3, "Portal Applications"

This chapter provides an overview of Oracle Application Server Portal features and benefits.

Chapter 4, "Wireless Applications"

This chapter provides an overview of Oracle Application Server Wireless features and benefits.

Chapter 5, "Oracle Business Intelligence"

This chapter provides an overview of the Oracle Business Intelligence Discoverer features and benefits.

Chapter 6, "Oracle Application Server Integration"

This chapter describes Oracle Application Server Integration features and benefits.

Chapter 7, "Oracle Application Server Infrastructure"

This chapter provides an overview of Oracle Application Server Infrastructure features and benefits.

Part II, "Deployment"

Chapters in this part discuss deployment topics.

Chapter 8, "Scalability and High Availability"

This chapter provides an overview of Oracle Application Server high availability and scalability features and benefits.

Chapter 9, "Performance and Caching"

This chapter provides an overview of Oracle Application Server performance and caching features and benefits.

Chapter 10, "System Management"

This chapter provides an overview of Oracle Application Server system management features and benefits.

Chapter 11, "Security and Identity Management"

This chapter provides an overview of Oracle Application Server security features and benefits.

Chapter 12, "Enterprise Deployments"

This chapter provides an overview of Oracle Application Server recommended enterprise deployment configurations.

Glossary

The glossary defines terminology used throughout this guide and the Oracle Application Server documentation set.

Related Documentation

For more information, see these Oracle resources:

- Oracle Application Server Documentation Library
- Oracle Application Server Platform-Specific Documentation on Oracle Application Server Disk 1

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technology/membership/`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technology/documentation/`

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

Introduction to Oracle Application Server

This chapter provides an overview of Oracle Application Server. The topics include:

- The Challenges of Creating and Maintaining an E-Business
- Building an E-Business Solution with the Oracle Platform
- Overview of Oracle Application Server

The Challenges of Creating and Maintaining an E-Business

The Internet introduces significant opportunities for companies to reach new markets and to streamline their internal business processes. At the same time, these opportunities generate new challenges because applications must be delivered quickly and must serve vast numbers of users.

Quickly delivering highly scalable applications often requires integrating diverse products that were not designed to work together. When this is the only way to get a consolidated set of business applications, companies must spend large amounts of capital, time, and effort, both initially and continuously, to support the complicated technologies or systems that they have built.

Companies that create **e-business** Web sites experience common problems and challenges associated with developing and deploying applications.

Development Challenges

There are numerous challenges associated with application development. Developing applications in a complex Internet environment involves dozens of interfaces, programming languages, and platforms. Assembling the infrastructure to support such applications requires comprehensive knowledge on the part of Internet application developers. In addition, developing, testing, and deploying applications is very time consuming.

There is a common set of requirements that today's Internet applications need to meet. The challenge for e-businesses is to create applications that integrate all of these requirements.

J2EE Applications

Most e-business application developers face the fact that development cycles have become shorter, while demand for an increasing number of programming languages and multiple platforms keeps growing. In recent years, developers began using **Java**, a platform-independent programming language, in application development. However, their applications are still vendor-dependent because of proprietary Application Programming Interfaces (**API**s). Complying with industry standards to create a level

of cross-vendor standardization has become an increasing demand and challenge. The J2EE platform and component specifications define a standard platform for developing multi-tier, Web-based enterprise applications. Using J2EE as a development requirement addresses the issue of conflicting industry standards.

Web Services

As e-business evolves, companies rely increasingly on their Web sites to conduct business. Web services provide a standards-based infrastructure through which any business can both offer internal business process services and dynamically link its business processes with those of its partners. Web services support direct interactions with other software applications using XML-based messages and Internet-based products.

Portals

Over the past several years, the portal has emerged as the new Internet desktop, enabling users to access information through Web browsers and to combine information from different sources into a single entry point. Portals also support personalized views, so that users or user groups can customize the content and the appearance of the portal to suit individual preferences and requirements. Secure, easy-to-use, and dynamic portals that provide access to all applications and Web services are essential infrastructure elements for e-business.

Wireless

An increasing number of people conduct business out of the office. Wireless applications allow users rapid access through mobile devices, such as Web-enabled phones, personal digital assistants, or pagers. However, wireless applications are limited by small screen size, limited data entry capacity, and heterogeneity in standards supporting wireless devices.

Enterprise Integration

Most companies need to integrate their existing applications and data sources with new business processes. As companies grow, it is essential to integrate existing applications and data sources with those of their partners, customers, and suppliers. These companies must perform a complete enterprise **application integration** without rewriting the applications or building extensive customized program logic.

Deployment Challenges

The key to the success of any Web site is how quickly and reliably the **server** can deliver the appropriate content to users. If a server takes too long to respond or if it fails, then users will take their business elsewhere. If content and applications are not secure, then sensitive information or information technology assets are vulnerable. If deployed applications cannot use hardware efficiently, then information technology budgets are depleted quickly on equipment purchases.

Concerns associated with application deployment include **availability**, **scalability**, performance, **caching**, systems management, and user and security management.

Availability

The **availability** of an overall system is measured by the percentage of time that it works normally. A successful e-business requires a 100% available operation without downtime. To improve availability, companies may use **redundant** components, but

only when the back-up component can take over for a failed primary component immediately.

Scalability

The **scalability** of a system refers to how well it responds as user demands increase. The number of Web users has increased tremendously in the past few years. It is crucial to maintain high quality Web sites that scale when you increase your hardware resources.

Performance

Increasing the performance of your Web site and increasing the speed of your applications without redesigning or rebuilding the Web site are common goals. Typical performance factors include the ability to handle multiple requests simultaneously, competition for resources, latency, response time, service time, throughput, wait time, and scalability.

Caching

Many high-volume Web sites serve thousands of users concurrently and need to provide them with accurate data in a dynamic environment where content changes frequently. Caching is one of the key technologies that promises to alleviate the computational and economic burdens resulting from complicated Web site infrastructure and technologies. However, it is often difficult to cache the dynamic, personalized content that the e-business model requires.

Systems Management

Application server administrators require sophisticated tools to monitor and manage the application platform, deployed applications, and the **privilege**s and access rights of individual application users. Administrators benefit from a central administration console that simplifies the management and monitoring of the application server and its components, and also allows them to manage application deployments, application configuration, and security.

Security and Identity Management

With the Internet continually growing, the threat to information traveling over the network increases exponentially. Maintaining a secure Web site is crucial. Once your Web site is running, you must protect the databases and the servers on which they reside. You must administer and protect the rights of internal database users, and you must guarantee the confidentiality of e-business customers as they access your database.

Building an E-Business Solution with the Oracle Platform

Oracle Application Server is a part of the Oracle platform, a complete and integrated e-business platform. The Oracle platform consists of:

- **Oracle Developer Suite** for developing applications
- **Oracle Application Server** for deploying Internet applications
- **Oracle Database Server** for storing content

To deliver scalable and high performance e-business solutions successfully, you must be able to leverage an integrated, comprehensive, flexible, and open platform. Oracle provides integrated development, deployment, and management tools, as well as a

runtime platform. The Oracle platform simplifies the process of creating and deploying the applications that run your business on the Internet.

> **See Also:** The product pages on the Oracle Technology Network at `http://www.oracle.com/technology` for more information about the Oracle Database and Oracle Developer Suite

Overview of Oracle Application Server

Oracle Application Server is a completely standards-based application server that provides a comprehensive and fully integrated platform for running Web sites, **J2EE** applications, and Web services. It addresses all the challenges that you face as you refine your business processes to become an e-business.

Oracle Application Server provides full support for the J2EE platform, **XML**, emerging Web services, and grid standards. With Oracle Application Server you can simplify information access for your customers and trading partners by delivering enterprise **portal**s, which can be customized and accessed from a network browser or from wireless devices. It allows you to redefine your business processes, and integrate your applications and data sources with those from your customers or partners.

Oracle Application Server allows you to save on Web site infrastructure by deploying your fast, scalable Internet applications through built-in Web **caching**, load balancing and clustering capabilities.

Oracle Application Server Solutions

The following sections describe solutions that Oracle Application Server provides to address the development and deployment challenges common to e-business Web sites. These solutions include J2EE and Internet applications, portals, wireless, business intelligence, e-business integration, caching, management, and security, built on a scalable and highly available infrastructure.

J2EE and Internet Applications

Oracle Application Server is built entirely on a J2EE framework that supports the latest industry standard technologies and programming languages, including J2EE **API** specifications, XML, and Web services. This comprehensive and flexible framework allows you to design, develop, and deploy dynamic Web sites, portals, and transactional applications using familiar programming languages and technologies such as servlets, JavaServer pages, XML, PL/SQL Server Pages, and SOAP. Oracle Application Server also provides comprehensive Web services to expose business functions to authorized parties over the Internet from any Web device.

Portals

Oracle Application Server provides an out-of-the-box portal that does not require endless programming and maintenance. You can use Oracle Application Server Portal to build, deploy, and maintain self-service and integrated enterprise portals. Oracle Application Server Portal allows for self-service content management and publishing, wizard-based development, and deploying, publishing, and consuming Web services on an extensible framework.

Wireless

Oracle Application Server Wireless simplifies wireless development and deployment by providing the ability to deliver content to any device, to use any protocol, and to work across any wireless network. In addition, Oracle Application Server Wireless

includes wireless services, such as e-mail and location-based services, that simplify wireless-enabling applications and portals. Oracle Application Server provides application developers independence from the underlying wireless infrastructure. Oracle Application Server Wireless is built on Oracle Application Server Containers for J2EE (OC4J), leveraging open standards support in XML and J2EE to deliver a high-performance and scalable wireless infrastructure.

Business Intelligence

Oracle Business Intelligence Discoverer provides the foundation for the Oracle Application Server Business Intelligence solution. It allows users to readily access, view, and analyze business data in a variety of ways, providing users with enhanced decision-making capabilities through up to the minute information.

E-Businesses Integration

Oracle Application Server has a powerful set of features that provide communications and integration capabilities for e-business applications. Using Oracle Application Server, you can integrate enterprise applications, trading partners, and Web services, emphasizing **scalability** and manageability, and provide seamless query and **transaction** access to many non-Oracle data sources. You can implement both human and process workflow automation, and connect out-of-the-box to a variety of packaged applications and legacy systems.

Availability and Scalability

Oracle Application Server provides a flexible deployment model that allows you to architect your system for high availability and scalability. Oracle Application Server provides a variety of options for improving availability and scalability, and provides features for implementing fault tolerance, death detection, and failover. Additionally, Oracle Application Server supports such high availability options as cold failover clusters and active failover clusters.

Caching

Oracle Application Server provides a Web caching solution with the unique capability of caching both static and dynamically-generated Web content. **Oracle Application Server Web Cache** significantly improves the performance and **scalability** of heavily loaded Web sites. In addition, the Web cache provides a number of features to ensure consistent and predictable responses. These features include page fragment caching, Edge Side Includes (ESI) and Edge Side Includes for Java (JESI) support, compression, dynamic content assembly, Web server load balancing, Web cache clustering, and **failover**.

Systems Management

Oracle Application Server provides a set of management facilities to simplify all aspects of Web site and application server administration. Using the management capabilities of Oracle Application Server, you can:

- Configure and monitor Oracle Application Server **instance**s to optimize them for performance and scalability from a centralized console
- Monitor and manage your entire suite with Oracle Enterprise Manager
- View real-time, graphical information about your application server processes with Topology Viewer
- Have your HTTP port management automated

- Create and manage application server clusters
- Deploy and configure applications
- Start and stop the application server and its components
- Respond to problem conditions from a centralized console
- Use encrypted secure sockets layer (**SSL**) connections, user and client certificate-based **authentication**, and **single sign-on** across all applications
- Implement Oracle Internet Directory, an **LDAP**-compliant directory that provides a single repository and administration environment for user accounts

Identity Management

The Oracle Application Server identity management infrastructure allows you to manage user identity throughout the application security life cycle. Oracle Application Server provides components for handling authentication, security services, authorization, and user provisioning to ensure the security of your Internet applications.

Oracle Application Server Components

Table 1–1 presents the Oracle Application Server components that are associated with these solutions.

Table 1–1 Oracle Application Server Components

Solution	Oracle Application Server Components
J2EE and Internet Applications	Oracle HTTP Server Oracle Application Server Containers for J2EE Oracle Application Server TopLink Oracle Application Development Framework Oracle Application Server Web Services Oracle JDeveloper Oracle XML Developer's Kit Oracle PL/SQL Oracle Content Management SDK Oracle Application Server MapViewer
Portals	Oracle Application Server Portal Oracle Application Server Portal Developer Kit Oracle Ultra Search
Wireless	Oracle Application Server Wireless Oracle Application Server Wireless Developer Kit Oracle Sensor Edge Server
Business Intelligence	Oracle Business Intelligence Discoverer
E-Business Integration	Oracle Application Server Integration InterConnect Oracle Application Server Integration B2B Oracle BPEL Process Manager Oracle Application Server Integration Adapters

Table 1–1 (Cont.) Oracle Application Server Components

Solution	Oracle Application Server Components
Caching	Oracle Application Server Web Cache
System Management	Oracle Enterprise Manager 10*g*
Identity Management and Security	Oracle Application Server Single Sign-On Oracle Application Server Certificate Authority Java Authentication and Authorization Service Oracle Internet Directory
High Availability	Oracle Application Server Guard Oracle Application Server Backup and Recovery Tool Oracle Application Server Disaster Recovery Tool
Installation and Upgrade	Oracle Application Server Metadata Repository Creation Assistant Oracle Application Server Upgrade Assistant Oracle Application Server Metadata Repository Upgrade Assistant

Part I

Development

Part I discusses topics related to application development, and contains the following chapters:

2

J2EE, Web Services, and Internet Applications

This chapter describes the Java 2 Platform, Enterprise Edition (**J2EE**), Web Services, and Internet application development and deployment environment in Oracle Application Server, explaining its features and concepts. The topics include:

- Application Development and Deployment in Oracle Application Server
- Oracle HTTP Server
- Oracle Application Server Containers for J2EE
- Oracle Application Server TopLink
- Oracle Application Development Framework
- Oracle Application Server Web Services
- Oracle XML Developer's Kit
- Oracle Application Server PL/SQL Platform
- Oracle Content Management Software Development Kit
- Oracle Application Server MapViewer

Application Development and Deployment in Oracle Application Server

Oracle Application Server provides an integrated, standards-based environment that provides for both developing applications and then deploying them, reliably delivering the applications to users across an enterprise. The following sections describe how this environment aids application development and deployment.

What Types of Applications Can Be Developed for Deployment in Oracle Application Server?

Oracle Application Server allows Web application developers to develop their sites in a variety of languages and technologies:

- **Java** and **J2EE**
- Web services
- **XML**
- **PL/SQL**

Table 2–1 lists the different technologies and programming languages that you can use to build applications for deployment with Oracle Application Server.

Table 2–1 Supported Technologies and Programming Languages

Java and J2EE	XML	PL/SQL	Web Services
■ JavaServer Pages (JSP) v. 1.2 ■ Java Servlets v. 2.3 ■ Enterprise JavaBeans (EJB) v. 2.0 ■ Java Database Connectivity (JDBC) v. 2.0 Extensions ■ Java Transaction API (JTA) v. 1.0 ■ Java Naming and Directory Interface (JNDI) v. 1.2 ■ Java Message Service (JMS) v. 1.0.2b ■ Java Authentication and Authorization Service (JAAS) v. 1.0 ■ J2EE Connector Architecture v. 1.0 ■ Java API for XML Parsing (JAXP) v. 1.1 ■ Java Mail v. 1.0	■ XML v. 1.0 ■ XML Namespaces v. 1.0 ■ Document Object Model (DOM) v. 1.0/2.0 ■ Extensible Stylesheet Language Transformations (XSLT) v. 1.0 ■ XML Schemas v. 1.0 ■ Simple API for XML (SAX) v. 1.0/2.0 + Extensions ■ XML Path Language (XPath) v. 1.0 ■ XSQL ■ Internet Data Access Presentation (IDAP)	■ PL/SQL Server Pages v. 10.1.2 ■ PL/SQL Web Toolkit v. 10.1.2	■ Web Services Description Language (WSDL) v. 1.1 ■ Universal Description, Discovery, and Integration (UDDI) v. 2.0 ■ Simple Object Access Protocol (SOAP) v. 1.1

Table 2–2 lists application development and deployment recommendations and requirements for various application types.

Table 2–2 Application Development and Deployment Recommendations and Requirements

Application Type	Programming Language/Methodology	Build Time Recommendation	Deploy Time Installation Requirement
Java	JSP, Swing, Applets	JDeveloper	OC4J Standalone or J2EE and Web Cache
J2EE	JMS, Servlet, EJB/Persistence Management, with JAZN-XML, with JAZN-LDAP, with JAZN-SSO	JDeveloper, TopLink Workbench	J2EE and Web Cache, J2EE and Web Cache associated with Infrastructure
J2EE	ADF/SOA	JDeveloper	J2EE and Web Cache, Portal and Wireless if you need UDDI

Table 2–2 (Cont.) Application Development and Deployment Recommendations and Requirements

Application Type	Programming Language/Methodology	Build Time Recommendation	Deploy Time Installation Requirement
non-J2EE	PL/SQL, Perl, PHP, C/C++	PL/SQL Web Toolkit for PL/SQL; otherwise use development tool of choice	J2EE and Web Cache, if using PSP and mod_plsql also need OWA packages in the database (can get through OracleAS Metadata Repository Creation Assistant)
Portal	Portal, portlets, omniportlet	Portal, JPDK	Portal and Wireless
Wireless	Wireless applications	Wireless SDK (part of OracleAS Developer Kits)	Portal and Wireless
Business Intelligence	Discoverer or BI Beans with Java	Discoverer Plus OLAP/Relational; JDeveloper with BI Beans (included in Oracle Developer Suite); Portal with Discoverer Portlets	Oracle Business Intelligence; OracleAS Portal (optional)
Integration	Data Integration Modeler, Business Process Integration Modeler, BPEL Designer, XSLT Mapping Tool, JCA Adapter Wizards	JDeveloper, OracleAS InterConnect iStudio	Oracle Application Server middle tier plus Integration installation (with appropriate install types)

The following sections explain how Oracle Application Server supports these technologies and programming languages.

Oracle HTTP Server

Oracle HTTP Server is the underlying deployment platform for all programming languages and technologies that Oracle Application Server supports. It provides a Web **listener** for Oracle Application Server Containers for J2EE (**OC4J**) and the framework for hosting static and dynamic pages and applications over the Web. Based on the proven technology of the **Apache** HTTP Server, Oracle HTTP Server includes significant enhancements that facilitate load balancing, administration, and configuration. It also includes a number of enhanced **module**s, or **mods**, which are extensions to the **HTTP server** that extend its functionality for other enterprise applications and services.

Oracle HTTP Server allows developers to program their sites in a variety of languages and technologies, such as **Java**, Perl, C, C++, PHP, and PL/SQL. Additionally, it can serve as either a forward or reverse **proxy server**. The following sections describe how Oracle HTTP Server provides a robust deployment platform for dynamic Web sites and applications.

Note: The version of Oracle HTTP Server that is installed as part of the standard Oracle Application Server installation types is built on Apache 1.3. Additionally, Oracle HTTP Server is available as a standalone installation in two versions: one based on Apache 1.3, and one based on Apache 2.0.

Oracle HTTP Server Components

Oracle HTTP Server consists of several components that run within the same process. These components provide the extensive list of features that Oracle HTTP Server offers when handling **client** requests. Major components include the following:

- **HTTP listener:** Oracle HTTP Server is based on an Apache **HTTP listener** to serve client requests.
- **Modules (mods):** Many of the standard Apache mods are included with Oracle HTTP Server. Oracle also includes several internal modules that are specific to Oracle HTTP Server components. Table 2–3 lists some of the Oracle HTTP Server modules.
- **Perl Interpreter:** A persistent Perl runtime environment embedded in Oracle HTTP Server through mod_perl.

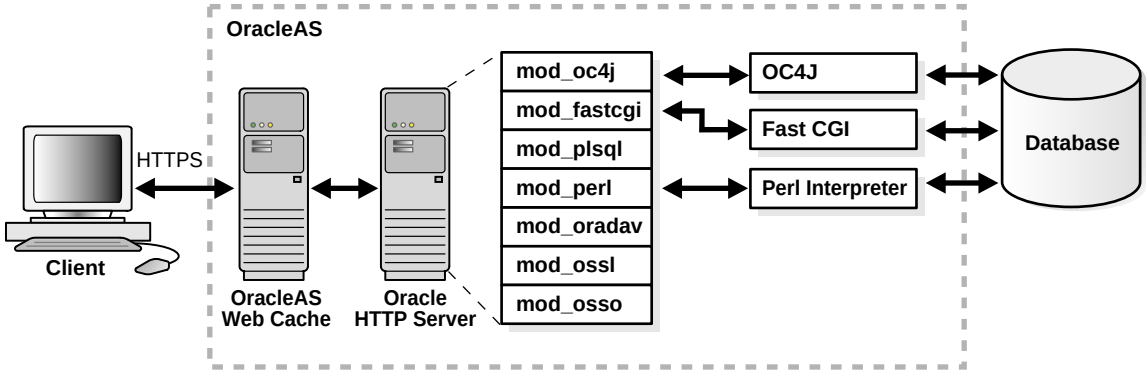
Table 2–3 Prominent Oracle HTTP Server Modules

Module	Description
mod_php	Supports PHP version 4.3.9, an open source client-side scripting language that is embedded in standard HTML.
mod_ security	Increases Web application security by protecting Web applications from known and unknown attacks.
mod_fastcgi	Supports FastCGI, which allows C, C++, and Java CGI programs to run in a performant environment
mod_perl	Routes requests to the Perl Interpreter
mod_plsql	Routes requests for stored procedures to the database server
mod_oc4j	Supports communication with Oracle Application Server Containers for J2EE and also performs some load balancing tasks
mod_oradav	Supports file as well as database distributed authoring and versioning
mod_ossl	Supports Secure Sockets Layer (SSL) and certificate sharing
mod_osso	Routes requests to Oracle Application Server Single Sign-On server

See Also: *Oracle HTTP Server Administrator's Guide* for a complete list of modules

Figure 2–1 shows the path of various requests through Oracle HTTP Server components.

Figure 2–1 Oracle HTTP Server Request Flow



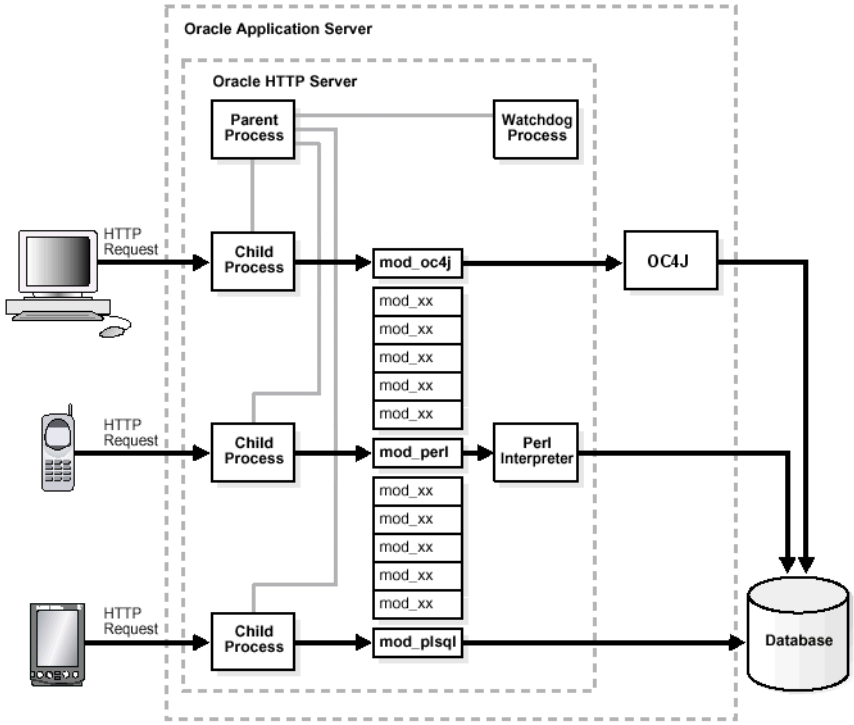
Oracle HTTP Server Architecture

At startup, the Web server parent process loads the entire configuration and the associated mods, and spawns a preconfigured number of child processes.

> **Note:** On Windows systems, the Web server main process is a child process that spawns multiple threads.

Figure 2–2 shows the process architecture of Oracle HTTP Server in a UNIX environment.

Figure 2–2 Oracle HTTP Server Process Architecture



The parent process does not listen to **HTTP** requests. Its sole job is to ensure that the child processes are running or that new ones are started when the load requires it.

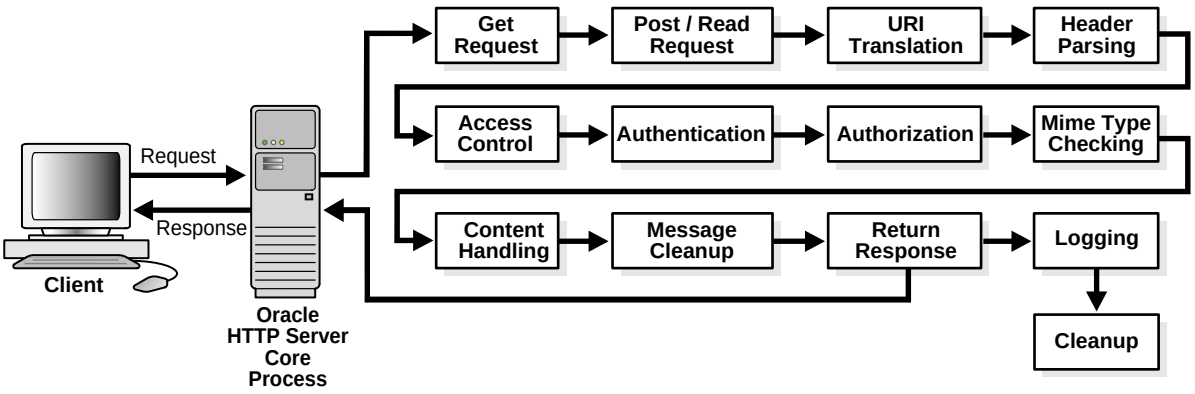
On UNIX platforms, each child process handles a single HTTP request. The child processes determine who should take the next request based on a mutex mechanism that you can configure.

Note: On Windows platforms, threads handle HTTP requests instead of child processes.

Modular Architecture

The architecture of Oracle HTTP Server is modular. The core HTTP listener is very small, and all capabilities are implemented as modules that plug in and are invoked at the appropriate place during the HTTP request lifecycle. Figure 2–3 shows the lifecycle of an HTTP request in Oracle HTTP Server.

Figure 2–3 Oracle HTTP Server HTTP Request–Response Cycle



A child process guides the request through this entire lifecycle. The modules register their application programming interfaces (**API**s), which are then either invoked automatically when the request reaches a certain stage in its lifecycle, or can be configured to be invoked only in certain situations.

Oracle HTTP Server Features

In addition to the standard Web **server** functionality of serving client requests to other Oracle Application Server components, Oracle HTTP Server provides the necessary features for both creating dynamic applications and implementing enterprise support. It also contains security enhancements that help protect important business resources. Key features of Oracle HTTP Server include the following:

- **PHP support through mod_php:** Oracle HTTP Server now supports PHP (recursive acronym for PHP: Hypertext Preprocessor), an open source, client-side scripting language used to generate dynamic HTML pages. PHP support is provided through `mod_php`.
- **Increased application security with mod_security:** `mod_security` is an open source intrusion detection and prevention engine for Web applications, which protects against known and unknown attacks.

- **Security through Secure Sockets Layer:** Secure Sockets Layer (SSL) is required to run any Web site securely. Oracle HTTP Server supports SSL encryption based on patented, industry standard algorithms. SSL works with both Internet Explorer and Netscape browsers. Oracle HTTP Server also provides support for dedicated SSL hardware, session renegotiation, and communication with OC4J using the AJP protocol over SSL.
- **Single Sign On Support:** Oracle HTTP Server supports the standard basic authentication features of Web servers. In addition, the mod_osso module is included to support single sign on across sites and across applications, improving the end-user experience.
- **Virtual Host Support:** A virtual host is a Web site that may share its Web server with other similar sites. Oracle HTTP Server provides a "container" environment for a virtual host, providing the virtual host with its own set of security and configuration directives. It also provides the location from which files are served. This allows ISPs to save on hardware and administrative costs by enabling multiple sites to be served from a single runtime instance of Oracle HTTP Server. However, due to technology limitations, only one virtual host can enable SSL.
- **Dynamic monitoring services (DMS):** These services automatically measure runtime performance statistics for both Oracle HTTP Server and **Oracle Application Server Containers for J2EE (OC4J)** processes. As applications run, DMS collects detailed performance statistics. This data allows you to monitor the duration of important request processing phases and status information. With this information, you can locate performance bottlenecks and tune the application server to maximize throughput and minimize response time.
- **Request ID:** To enhance request tracking through various components, a request ID is now attached to each request. This provides more detailed tracking information, allowing you to see how much time a particular request spends in any component or layer.
- **External API for performance monitoring:** This API allows you to use external, third-party performance monitoring tools to monitor Oracle Application Server-based J2EE components, such as servlets and JSPs, as well as J2EE containers.
- **Proxy Plug-In:** To accommodate requirements for non-Oracle HTTP listeners, Oracle HTTP Server provides a proxy plug-in that can be plugged directly into Sun ONE or Microsoft **Internet Information Server (IIS)**. This proxy plug-in is used to forward requests for Oracle Application Server component services to Oracle HTTP Server, which is placed behind the non-Oracle listener.
- **OC4J Plug-In:** The OC4J Plug-In provides a way for you to use Apache as well as the IIS and Sun ONE third-party listeners to access servlets running in OC4J without having to use the Oracle HTTP Server as a proxy. The OC4J Plug-In routes requests directly from the third-party HTTP listener to OC4J.
- **Oracle Application Server Single Sign-On Plug-in:** This plug-in is the Oracle single sign-on solution for third-party listeners such as Sun ONE and IIS. The plug-in is designed to protect native third-party listener applications using the single sign-on infrastructure. Using this plug-in, you can be authenticated to different third-party listener applications using only one password. You can integrate these protected third-party listener applications with other single sign-on enabled applications as long as they are all protected by the same single sign-on server.

Common Gateway Interface (CGI) Support

Requests that are sent to a common gateway interface (**CGI**) program may invoke two new processes—the child process that handles the HTTP request and the CGI program itself. It is possible to avoid this overhead by configuring Oracle HTTP Server to pre-start child processes and keep them running, leveraging the Perl module to run the CGI programs in memory, or using the FastCGI mechanism.

The following Oracle HTTP Server features support CGI:

- **FastCGI:** FastCGI supports applications written with Perl, C, C++, and Java. Each CGI application runs in a single child process of the Web server. This improves server performance because it eliminates the need to start a new process for every application request.
- **Perl Interpreter:** The Perl Interpreter supports Perl applications and runs them inside of the Web server process. This allows for greater performance because running applications does not start new processes. Since the interpreter runs inside of the Web server process, it has access to Web server services such as log files.

Oracle Application Server Containers for J2EE

Oracle Application Server Containers for J2EE (OC4J) is a fast, lightweight, and scalable **J2EE** 1.3 certified server implementation that is written in **Java** and runs on a standard Java Virtual Machine (**JVM**). It has been designed to support the standard **API**s in Table 2–4.

Table 2–4 Oracle Application Server Containers for J2EE Supported APIs

API	Version
JavaServer Pages (JSP)	1.2
Java Servlet	2.3
Enterprise JavaBeans (EJB)	2.0
Java Database Connectivity (JDBC)	2.0 Extensions
Java Transaction API (JTA)	1.0
Java Message Service (JMS)	1.0.2b
JavaMail	1.2
JavaBeans Activation Framework	1.0
Java API for XML (JAXP)	1.1
J2EE Connector Architecture	1.0
Java Authentication and Authorization Service (JAAS)	1.0

Introduction to J2EE Application Development

The following sections provide introductory definitions and summaries of application programming technologies that Oracle Application Server supports. For detailed information on the technologies introduced in this section, refer to the following information sources.

See Also:

- The official J2EE Web site at `http://java.sun.com/j2ee`
- *The J2EE Tutorial* at `http://java.sun.com/j2ee/tutorial`
- The Oracle Application Server Documentation Library

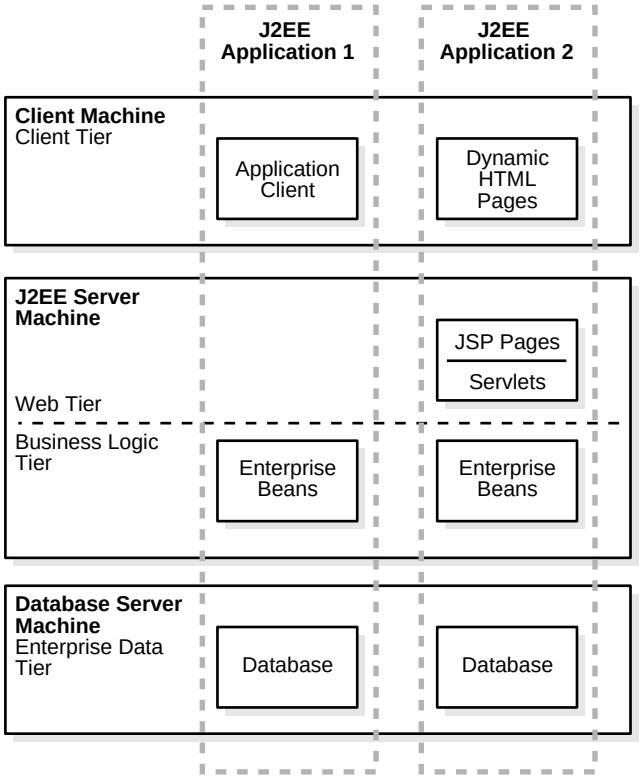
What Is a J2EE Application?

A J2EE application is an application that is written in Java using the J2EE APIs. It can be deployed, managed, and executed on a J2EE-compatible server. The J2EE application itself is composed of a set of components, such as Web presentation modules, business logic modules, and data access modules. Each component is assembled into the overall application with all of its related classes and **XML** deployment descriptors.

J2EE Distributed Multi-tiered Application Model

The J2EE platform provided in Oracle Application Server uses a multi-tiered distributed application model. A multi-tiered distributed application model divides application logic into components according to function, and the various application components that make up J2EE applications can be installed on different machines, depending on which tier in the multi-tiered J2EE environment the application component belongs. Figure 2–4 shows two multi-tiered J2EE applications divided into the **client**, Web, business logic, and enterprise data tiers.

Figure 2–4 J2EE Distributed Multi-tiered Application Architecture



You can distribute J2EE applications across the four tiers shown in this figure, but generally they are considered to be three-tier applications because they are usually distributed over the following machine locations:

- client machines
- **J2EE Server** machines hosting presentation services, like JSPs and **servlet**s, and business logic components, like EJBs
- database **server**s or legacy machines at the back end

Three-tiered applications that run in this way extend the standard two-tiered client and server model by placing an application server between the client and the back-end storage.

Types of J2EE Clients

J2EE applications support the following clients:

- **Application Clients:** Applications running on a client machine that directly access **enterprise beans** that are running in the business logic tier. Application clients can also open an **HTTP** connection to establish communication with a servlet running on the Web tier if a J2EE application requires it.
- **Dynamic HTML and XML Pages:** In the context of J2EE applications, dynamic HTML and XML pages are either generated by servlets or created with JavaServer Pages technology running in the Web tier. These pages can be extensions to traditional static HTML pages, allowing application developers to offer customized and personalized pages to the client.

Types of J2EE Application Components

You can use the following components in J2EE applications:

- **Servlets:** A servlet is a Java class that executes behind a Web server and can extend the capability of the Web server to provide services for dynamic page creation or application logic. The servlet works in the standard HTTP request-response model.
- **JavaServer Pages:** JavaServer Pages (JSPs) are text files that contain two types of information: static template data, which can be expressed in any text-based format, such as HTML, XML, or WML (Wireless Markup Language); and JSP elements, which construct dynamic content.
- **Enterprise Beans:** Enterprise beans are server-side components that encapsulate the business logic of an application.

Types of J2EE Containers

A **container** provides the runtime support for J2EE application components. Containers provide a federated view of the underlying J2EE APIs to the application components. J2EE application components never interact directly with other J2EE application components. They use the protocols and methods of the container for interacting with each other and with platform services. Interposing a container between the application components and the J2EE services allows the container to transparently inject the services defined by the components' deployment descriptors, such as declarative transaction management, security checks, resource pooling, and state management.

Before a Web or enterprise bean component can run, it must be assembled into a J2EE application and deployed into a J2EE container. The assembly process involves specifying container settings for each component, which customize the underlying

support provided by the J2EE server. These settings can be standard J2EE settings or container-specific settings, depending on your application requirements. Some of the container settings that you can specify include security services, **transaction** model, naming and directory lookup, and remote connectivity model.

There are two primary types of J2EE container:

- **Enterprise JavaBeans Container:** The Enterprise JavaBeans container (**EJB container**) manages the execution of all enterprise beans for J2EE applications. Enterprise beans and their container run on the J2EE server.
- **Web Container:** Web components such as JSP pages and servlets are managed and executed in the **servlet container**. The container provides services such as request dispatching, security, concurrency, and life cycle management. The Web container also gives **Web component**s, typically JSPs and servlets, access to the J2EE APIs such as naming, transactions, and JDBC.

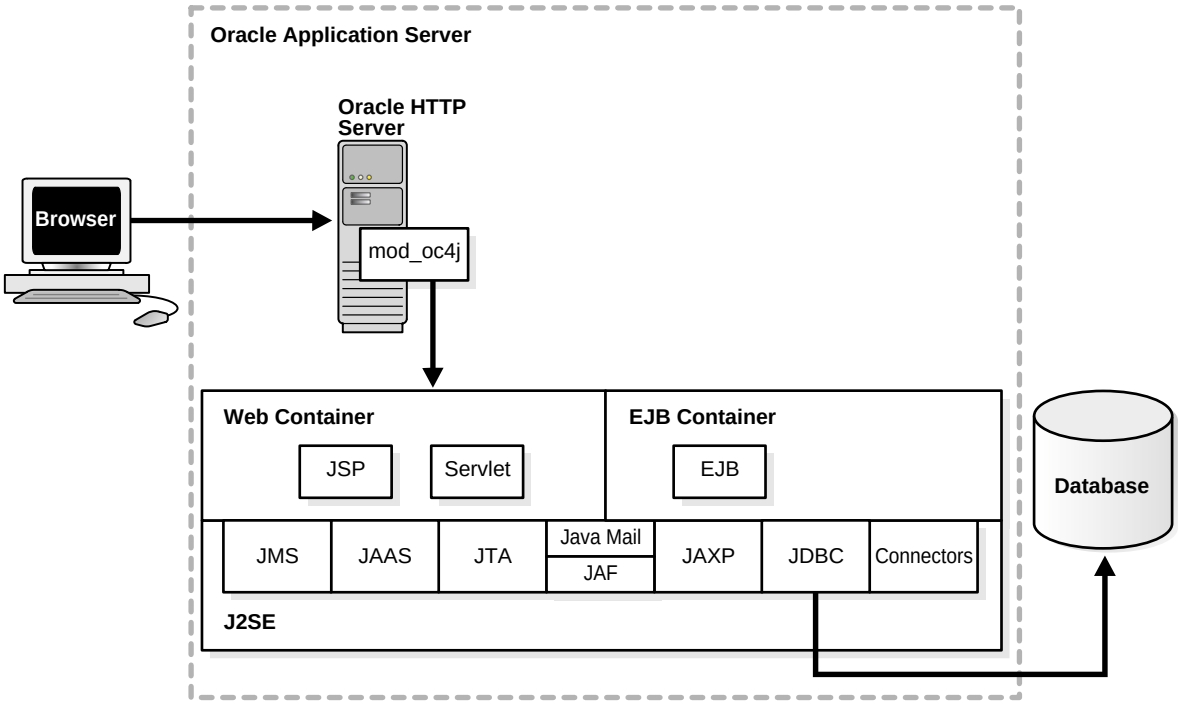
J2EE Application Packaging Concepts

J2EE components are packaged separately and bundled into a J2EE application. Each component, with its related files such as GIF and HTML files or server-side utility classes, are packaged together with a deployment descriptor (DD), and are assembled into a **module** that is added to the J2EE application. Typically, a J2EE application is composed of one or more enterprise beans and Web or application client component modules.

Oracle Application Server Containers for J2EE Architecture

Figure 2–5 shows the architecture of OC4J within Oracle Application Server.

Figure 2–5 Oracle Application Server Containers for J2EE Architecture



OC4J is supported by the Java 2 Platform, Standard Edition (**J2SE**) infrastructure as shown in Figure 2–5. This means that the OC4J Web container and OC4J **EJB container** use the J2SE **virtual machine**. J2EE applications are modularized for reuse of application components, such as:

- User interfaces, which can be composed of JavaServer Pages (**JSP**s), dynamic **HTML**, and so on
- Business logic, which is usually contained in an Enterprise JavaBean (**EJB**) or normal Java classes

The J2EE containers also perform services for applications, such as providing access to the APIs and lifecycle management.

Oracle Application Server Containers for J2EE Features

Oracle Application Server Containers for J2EE (OC4J) has the following features.

Oracle Application Server Containers for J2EE Containers

OC4J supplies the following J2EE containers:

- A servlet container that complies with the servlet 2.3 specification
- A JSP container that complies with the Sun JSP 1.2 specification
- An EJB container that complies with the EJB 2.0 specification

Oracle Application Server Containers for J2EE Servlet Container A servlet is a Java program that runs on a J2EE server, such as OC4J. A servlet is one of the application component types of a J2EE application. It must execute under the control of a servlet container, which is part of the OC4J Web container. The servlet container calls the servlet's methods and provides services that the servlet needs when running.

See Also: "Types of J2EE Application Components" on page 2-10

About the Servlet Container

The servlet container executes and manages servlets. It provides the servlet with access to properties of the HTTP request, such as headers and parameters. Also, the container provides the servlet with access to other Java APIs, such as **JDBC** to access a database, remote method invocation (**RMI**) to call remote **object**s, or JMS to perform asynchronous messaging.

How the Servlet Container Works

When a request is mapped to a servlet, the servlet container performs the following steps:

1. If an **instance** of the servlet does not exist, the container does the following:
   a. Loads the servlet class
   b. Instantiates an instance of the servlet class
   c. Initializes the servlet instance
2. The container then invokes the servlet, passing request and response objects. The request object contains information about the client, request parameters, and **HTTP header**s. The response object returns the servlet's output to the client.

The servlet extracts information from the client request, accesses external resources, and then populates the response based on that information.

Oracle Application Server Containers for J2EE JavaServer Pages Container JavaServer Pages provide a convenient way to generate dynamic content in Web pages. JSP technology, which is closely coupled with servlet technology, allows you to include Java code fragments and make calls to external Java components (in the form of tags and directives) from within your Web pages. Typically, the markup code used to compose your Web pages is HTML or XML. JSPs work well as a front-end for business logic and dynamic functionality in **JavaBeans** or Enterprise JavaBeans (EJBs).

A JSP is translated into a Java servlet before being run, and it processes HTTP requests and generates responses like any servlet. However, JSP technology provides a more convenient way to code a servlet. Translation occurs the first time the application is run. A **JSP translator** is triggered by the `.jsp` file name extension in a **URL**.

JSPs are fully interoperable with servlets. You can include output from a servlet or forward the output to a servlet, and a servlet can include output from a JSP or forward output to a JSP.

About the JSP Translator

The JSP translator has a translator and a compiler. The JSP translator translates a JSP into a Java source file. The container compiles the source file into a Java bytecode (`.class`) file, which executes as a servlet in the servlet container using the JSP runtime library. The servlet container provides access to Java APIs and other services.

How the JSP Translator Works

When a user requests a URL that maps to a JSP file, such as `http://host/Hello.jsp`, the following steps occur:

1. The Web server invokes the JSP translator, which translates `Hello.jsp` and produces the file `Hello.java`.
2. The Java compiler is invoked, creating a `Hello.class` servlet.
3. `Hello.class` runs, using the JSP runtime library, which contains the supporting files to interpret the tags and directives from the JSP.
4. If the `Hello` class requires information from a database, then the servlet container provides JDBC access to the class so it can retrieve the information and return its output to the client browser.

Oracle Application Server Containers for J2EE Enterprise JavaBeans Container The OC4J **EJB container** manages the execution of enterprise beans for J2EE applications. Like the OC4J Web container, the EJB container uses the J2SE **virtual machine**. The following sections describe what services the EJB container provides to J2EE applications and how it works.

Enterprise beans are the J2EE components that implement Enterprise JavaBeans technology. Enterprise beans run in the EJB container. An enterprise bean is a portable server-side component that encapsulates the business logic of an application. There are three types of EJBs: **session beans**, **entity beans**, and **message-driven beans**.

About the Oracle Application Server Containers for J2EE EJB Container

The OC4J EJB container provides system-level services to EJBs similar to the services that the Web container provides to servlets and JSPs. The container has configurable settings that customize the underlying support provided by OC4J, the J2EE server. The configurable settings include security, transaction management, Java Naming and Directory Interface (**JNDI**) lookups, and remote connectivity. In addition to the configurable settings, the container also manages EJB life cycles, database connection resource pooling, data persistence, and access to the J2EE APIs.

How the EJB Container Works

How the EJB container works depends on what type of enterprise bean you are using. The container manages the execution of the enterprise bean for one J2EE application.

For **session bean**s, the EJB container provides all of the services that the Web container provides to Web components, such as access to APIs and the **virtual machine**, **transaction service**s like Container Managed Transactions (CMTs), and secure and authorized EJB method invocation.

For entity beans, which represent business objects in a persistent storage mechanism, there are models for how the persistence is performed. You can have either **bean-managed persistence (BMP) bean**s or **container-managed persistence (CMP) bean**s. With bean-managed persistence, the entity bean code contains the calls that access the database and the EJB container triggers callback methods on your code. Entity beans with bean-managed persistence execute in the EJB container with the typical container support and services. However, if you are using container-managed persistence, then the EJB container automatically generates the necessary database access calls. The EJB methods do not require any JDBC code to manage EJB data persistence.

J2EE Services

Java 2 Platform Enterprise Edition (J2EE) provides core services for writing J2EE components. The J2EE containers manage access to these services for the application components. The services are as follows:

- **Java Database Connectivity (JDBC):** This service lets you invoke **SQL** commands from Java programming methods. You use the JDBC API in an enterprise bean when you override the default container-managed persistence or have a session bean access the database. You can also use the JDBC API from a servlet or a JSP to access a database directly without going through an enterprise bean.

  Oracle Application Server includes the following drivers to provide highly scalable and reliable connectivity to both Oracle and non-Oracle data sources:

  - **Oracle JDBC drivers:** The Oracle JDBC drivers, in addition to providing standard JDBC API support, have extensions to support Oracle-specific datatypes and to enhance their performance. They are meant to be used with the Oracle database.
  - **J2EE Connectors:** The J2EE Connector architecture, part of the J2EE platform, provides a Java-based solution for connecting various application servers and enterprise information systems that are already in place.
  - **DataDirect Connect Type 4 JDBC drivers:** DataDirect JDBC drivers are meant specifically for connecting to non-Oracle databases, such as Microsoft SQL Server and Sybase.

- **Java Message Service (JMS):** This service is a messaging standard that allows J2EE application components to create, send, receive, and read **message**s. It enables distributed communication that is loosely coupled, reliable, and asynchronous.

  For this release, JMS support has been enhanced by the addition of a lightweight JMS provider in addition to the Oracle JMS offered in previous versions. The new lightweight JMS is fully JMS 1.0 compatible and can support durable messaging through a file-based persistence mechanism, which provides improved stability and performance. Support for Message Driven Beans is also now available for both Oracle JMS (AQ) and the lightweight OC4J JMS.

- **Java Transaction API (JTA):** This service provides a standard demarcation interface for demarcating transactions. Typically, it is used in J2EE applications that use two or more separate database access operations that depend on each other to demarcate where the entire transaction begins, rolls back, and commits.

  OC4J provides additional support for two-phase commits for applications that require commit coordination across machines and containers. The two-phase commit engine is responsible for ensuring that when a distributed transaction ends, changes to all participating databases are all either committed or rolled back.

- **Java Naming and Directory Interface (JNDI):** This service provides a standard interface to naming and directory services. J2EE applications use JNDI to find other distributed objects. The JNDI Interface has two parts: an application-level interface used by application programs to access naming and directory services, and a service provider interface to attach a provider of naming and directory services.

- **JavaMail Technology:** This service provides J2EE applications with a JavaMail API and a JavaMail service provider with which to send e-mail notifications.

- **Java API for XML (JAXP):** This service provides support for the industry standard SAX and DOM APIs for parsing XML documents, as well as support for XSLT transform engines. It enables applications running in a J2EE container to make use of XML.

- **J2EE Connector Architecture:** This service provides an interface for resource adapters that allow J2EE applications to access and interact with databases and other enterprise information systems (**EIS**). Oracle Application Server provides several out-of-the box adapters, and also allows you to build your own.

- **Java Authentication and Authorization Service (JAAS):** This service provides a way for a J2EE application to authenticate users against different security provider systems and authorize a specific user or group of users to run J2EE applications under **role** permission and control enforcement.

  Oracle Application Server provides an implementation of Java Authentication and Authorization Service (JAAS) that integrates with the Oracle Application Server J2EE security infrastructure to enforce security constraints for Web components (servlets and JSPs) and EJB components. The Oracle Oracle Application Server Java Authentication and Authorization Service (JAAS) Provider implementation does the following:

  - Integrates Java-based applications with Oracle Application Server Single Sign-On, including **authentication**, thereby giving you extensible security for Java-based applications
  - Manages access control policies centrally in Oracle Internet Directory, controls access by role, and partitions security **policy** by subscriber
  - Supports impersonation of a specific user, allowing an enterprise bean, servlet, or JSP to run with the permissions associated with the current client or a specified user

  See Also: The Sun Microsystems, Inc. published specifications for these services, and *The J2EE Tutorial* for instructions on how to use them in J2EE applications, at:

  `http://java.sun.com`

Oracle J2EE Services

In addition to the standard J2EE Services, Oracle also provides the following services to Java developers:

- **Java Object Cache** stores frequently accessed or resource-intensive objects in memory or on disk. It is a low-level object caching API, supporting generic object types such as memory objects, disk objects, pooled objects, and StreamAccess objects. Java Object Cache uses distributed object management to coordinate updates and invalidations of Java objects.

  Objects are loaded using a user-defined CacheLoader object and accessed through the easy-to-use API. This eliminates the need to repeatedly create and load information within a Java application. The Java Object Cache retrieves content quickly and greatly reduces the load on Oracle Application Server.

  Java Object Cache provides caching for expensive or frequently used Java objects when the application servers use a Java program to supply their content. Cached Java objects may contain generated pages, or they may provide support objects within the program to assist in creating new content. Java Object Cache automatically loads and updates objects as specified by the Java applications and includes APIs to manage the cached Java objects. The generic nature of Java Object Cache makes it an ideal cache repository for higher level caches such as Web Object Cache, easing the development effort and reducing complexity.

Oracle Application Server TopLink

Oracle Application Server TopLink is an advanced object-to-relational persistence framework, suitable for a wide range of Java 2 Enterprise Edition (J2EE) and Java application architectures. OracleAS TopLink development tools and runtime capabilities reduce development and maintenance efforts, and increase enterprise application functionality. Use OracleAS TopLink to build high-performance applications that store persistent data in a relational database.

The following sections introduce OracleAS TopLink and include discussions on these topics:

- Advantages of OracleAS TopLink
- OracleAS TopLink Components
- Application Development with OracleAS TopLink
- OracleAS TopLink Architectures Overview

Advantages of OracleAS TopLink

Enterprise applications rely on Java-to-database integration to implement objects and logic. OracleAS TopLink enables developers to efficiently develop and refine enterprise applications. To fully understand OracleAS TopLink, you must understand the problems that enterprise application developers face and how OracleAS TopLink resolves them.

The OracleAS TopLink Problem Space

Java-to-database integration is a widely underestimated problem in enterprise Java applications. This complex problem involves more than reading from and writing to a database. The database world includes elements such as tables, rows, columns, and primary and foreign keys; the Java and J2EE world contains entity classes (regular Java classes or Enterprise JavaBeans (EJB) entity beans), business rules, complex

relationships, and inheritance. Bridging these two fundamentally different technologies is a challenging and resource-intensive problem.

The process of translating object-oriented data into relational data is referred to as *object-relational* (O-R) *mapping*. To enable an O-R solution, developers must resolve the following O-R bridging issues:

- Fundamentally different technologies
- Different skill sets
- Different staff and ownership for each of the technologies
- Different modeling and design principles

Application developers need a product that enables them to integrate Java applications and relational databases, without compromising ideal application design or database integrity. In addition, Java developers need the ability to store (or *persist*) and retrieve business domain objects using a relational database as a repository.

The OracleAS TopLink solution is a persistence framework that manages O-R mapping in a seamless manner and enables developers to rapidly build applications that combine the best aspects of object technology and relational databases.

The OracleAS TopLink Solution

OracleAS TopLink provides a mature and powerful solution that addresses the disparity between Java objects and relational databases. OracleAS TopLink enables developers to:

- Persist Java objects in virtually any relational database supported by a JDBC 2.0 compliant driver
- Map any object model to any relational schema, using the Oracle Application Server TopLink Mapping Workbench graphical mapping tool
- Use OracleAS TopLink successfully, even if they are unfamiliar with SQL or JDBC, because OracleAS TopLink provides a clean, object-oriented view of relational databases

Other OracleAS TopLink Advantages In addition to providing industry-leading O-R mapping capabilities, OracleAS TopLink provides flexibility, increases performance and maximizes the productivity of your applications. OracleAS TopLink provides the following features:

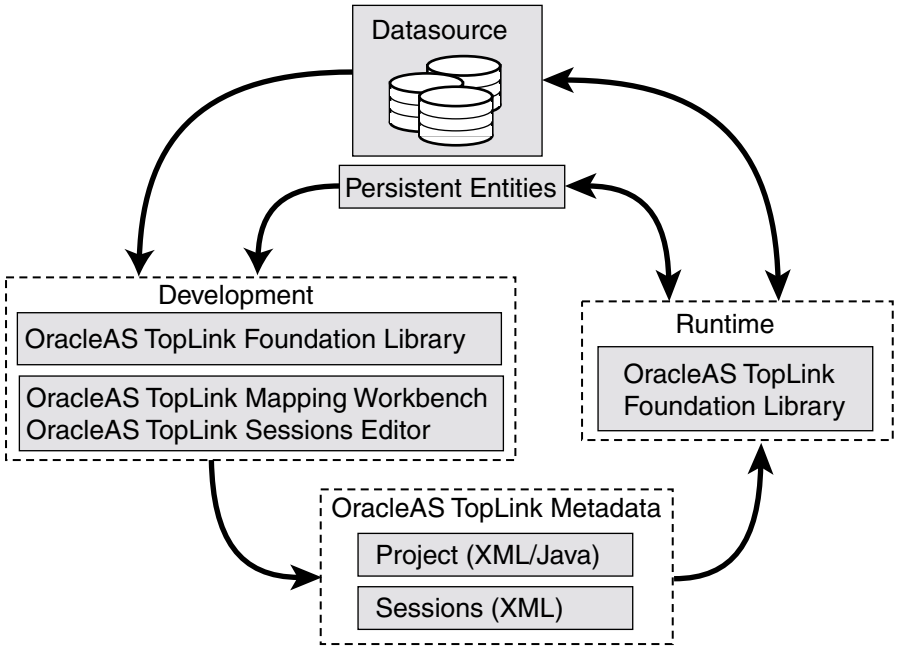
- Advanced object caching that improves performance by minimizing database access.
- Rich query support that provides easy access to sophisticated, dynamic query languages and tools such as *query by example*, Java expression-based queries, EJB QL, and SQL.
- A transactional framework that enables developers to easily create and modify mapped objects. This framework integrates the complexities of a shared memory space and caches, and provides scalability that supports multiple server instances (clustering). Although the mechanisms involved are complex, OracleAS TopLink makes it easy to leverage this functionality by simplifying the task of writing transactional code that complies with database referential integrity and optimal access patterns.

OracleAS TopLink Components

At its core, OracleAS TopLink is a runtime engine that provides Java or J2EE applications with access to persistent entities stored in a relational database. In addition to runtime capabilities, the Oracle Application Server TopLink Foundation Library includes the OracleAS TopLink Application Programming Interface (API). This API enables applications to access OracleAS TopLink runtime features, as well as development tools that simplify application development. The tools capture mapping and runtime configuration information in metadata files that TopLink passes to the runtime.

Figure 2–6 shows how the OracleAS TopLink components relate to one another throughout the development cycle.

Figure 2–6 TopLink Components in the Development Cycle



OracleAS TopLink Development Components

OracleAS TopLink application development comprises three elements: the development environment, the OracleAS TopLink runtime, and the metadata that ties them together.

Development To create an OracleAS TopLink application, map the object and relational models using the OracleAS TopLink Mapping Workbench, and capture the resulting mappings and additional runtime configurations in the OracleAS TopLink project file (the `project.xml` file). Then build a session configuration file (the `sessions.xml` file) in the OracleAS TopLink Sessions Editor. These files together represent your entire OracleAS TopLink project.

During development, developers leverage the OracleAS TopLink API to define query and transaction logic. When developers use EJB entity beans, there is generally little or no direct use of the OracleAS TopLink API.

Runtime The OracleAS TopLink Foundation Library provides the OracleAS TopLink runtime component. Access the runtime component either directly through the OracleAS TopLink API, or indirectly through a J2EE container when using EJB entity

beans. The runtime engine is not a separate or external process; instead, it is embedded within the application. Application calls invoke OracleAS TopLink to provide persistence behavior. This function allows for transactional and thread-safe access to shared database connections and cached objects.

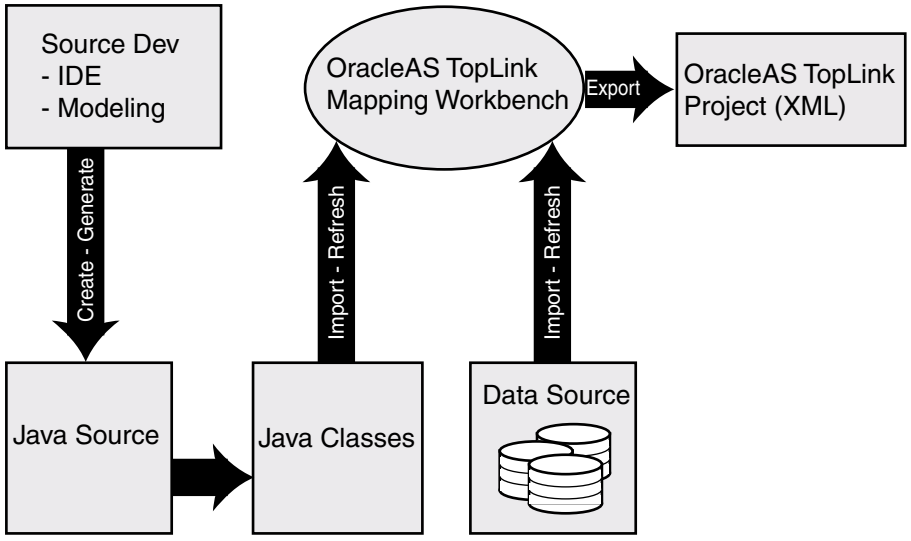
Metadata OracleAS TopLink metadata is the bridge between the development of an application and its deployed runtime. Capture the metadata using the OracleAS TopLink Mapping Workbench and OracleAS TopLink Sessions Editor, and pass the metadata to the runtime using deployment `project.xml` and `sessions.xml` files. It is also possible to hand-code these files using Java and the OracleAS TopLink API, but this approach is more labor-intensive.

The metadata, encapsulated in the `project.xml` file and the `sessions.xml` file, allows developers to pass configuration information into the runtime environment. The runtime uses the information in conjunction with the persistent entities (Java objects or EJB entity beans), and the code written with the OracleAS TopLink API, to complete the application.

Oracle Application Server TopLink Mapping Workbench

The OracleAS TopLink Mapping Workbench is a graphical development tool that enables developers to map between the object and relational models, and configure many of the OracleAS TopLink Foundation Library features. The OracleAS TopLink Mapping Workbench creates an OracleAS TopLink project, the primary object in the OracleAS TopLink metamodel. Export the project as a single deployment XML file (the `project.xml` file), which OracleAS TopLink uses in conjunction with the OracleAS TopLink runtime to provide the application-specific persistence capabilities. Figure 2–7 shows how the OracleAS TopLink Mapping Workbench fits into the OracleAS TopLink environment.

Figure 2–7 The OracleAS TopLink Mapping Workbench in a TopLink Environment



The OracleAS TopLink Mapping Workbench can import compiled entity classes (Java objects or EJB entity beans), as well as relational schema through a JDBC driver that the developer configures. Because OracleAS TopLink imports the object and relational models for mapping, developers can develop the two models relatively independently from the O-R mapping phase of project development.

Oracle Application Server TopLink Sessions Editor Most OracleAS TopLink applications include a session configuration file, the sessions.xml file, to simplify the application deployment process. The OracleAS TopLink Sessions Editor provides a graphical environment in which to configure the `sessions.xml` file.

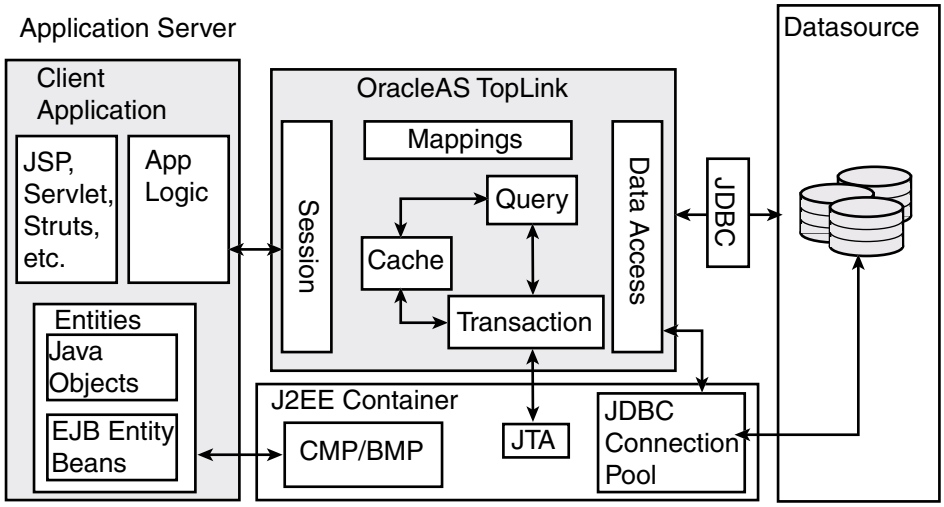
Use the `sessions.xml` file to configure one or more sessions for the OracleAS TopLink project, and associate the sessions with the project. This approach allows developers to specify individual configurations for each session and to add or modify the following:

- Database (JDBC) login information different from the login information used during development (for example, external data sources for the host application server's connection pools)
- JTA / JTS transaction usage
- Cache synchronization
- Session broker (enables client applications to view multiple databases and projects as a single OracleAS TopLink session)

Oracle Application Server TopLink Foundation Library

The Oracle Application Server TopLink Foundation Library includes a Java library that forms the runtime component of the product. It provides support and the API for the components that make up an OracleAS TopLink application. The API enables developers to interact with OracleAS TopLink to retrieve and modify their application persistent entities. Figure 2–8 shows the interaction of the parts of an OracleAS TopLink application.

Figure 2–8 OracleAS TopLink Application Components



Note: Although this section describes how these components fit into J2EE architectures, note that OracleAS TopLink also supports non-J2EE solutions. The *Oracle Application Server TopLink Application Developer's Guide* describes these solutions in more detail.

Sessions A session is the primary interface between the client application and OracleAS TopLink, and represents the connection to the underlying relational

database. OracleAS TopLink offers several different session types, each optimized for different design requirements and architectures. The session manager configures and manages the session as a singleton within the application.

The most commonly-used session is the server session, a singleton session that clients access on the server through a client session. The server session provides a shared cache and shared JDBC connection resources. OracleAS TopLink also supports sessions for two-tier architectures, distributed applications, and multiple databases.

Data Access The OracleAS TopLink data access component provides access to JDBC connections through connection pooling, provided either by OracleAS TopLink or a host application server. This component manages the SQL generation required by the various query operations and reconciles any differences between JDBC drivers and SQL dialects. OracleAS TopLink offers many performance tuning options that optimize its data access capabilities.

Caching OracleAS TopLink supplies an object level cache that guarantees object identity and provides performance enhancement. Developers can configure the OracleAS TopLink cache and maximize the application efficiency by reducing the number of times the application accesses the database. In a clustered environment, developers can configure OracleAS TopLink to synchronize changes with other instances of the deployed application.

Queries The OracleAS TopLink query framework provides developers with the flexibility necessary to manage the complex persistence requirements of enterprise applications. The key features of this query framework include:

- A rich set of query types to allow object retrieval, summary results, and raw data retrieval
- The ability to specify the search criteria using OracleAS TopLink Expressions (for object model based queries), EJB QL, SQL, stored procedures, or query by example
- Configuration options that enable developers to specify how the query is executed, and customize many of its performance optimizing features

Developers can define OracleAS TopLink queries using the OracleAS TopLink Mapping Workbench, in Java code using the OracleAS TopLink API, or, in the case of EJB entity beans, through EJB Finders.

Transactions OracleAS TopLink provides the ability to write transactional code isolated from the underlying database and schema. OracleAS TopLink achieves this functionality through the Unit of Work.

The Unit of Work isolates changes in a transaction from other threads until it successfully commits the changes to the database. Unlike other transaction mechanisms, the Unit of Work automatically manages changes to the objects in the transaction, the order of the changes, and changes that might invalidate other OracleAS TopLink caches. The Unit of Work manages these issues by calculating a minimal change set, ordering the database calls to comply with referential integrity rules and deadlock avoidance, and merging changed objects into the shared cache. In a clustered environment, the Unit of Work also synchronizes changes with the other servers in the cluster.

If an application uses EJB entity beans, developers do not access the Unit of Work API directly, but they still benefit from its features: the integration between the OracleAS TopLink runtime and the J2EE container leverages the Unit of Work automatically.

JTA/JTS Integration: By default, OracleAS TopLink allows the application to create transaction boundaries for all object-level changes. OracleAS TopLink explicitly manages the database transaction, and if it encounters problems, safely rolls back both the database changes and the object-level changes.

In the case of a J2EE application, developers can configure OracleAS TopLink to synchronize with the JTA / JTS subsystem of the host application server. This feature allows an application to use container-managed transactions, rather than the default user-managed transactions.

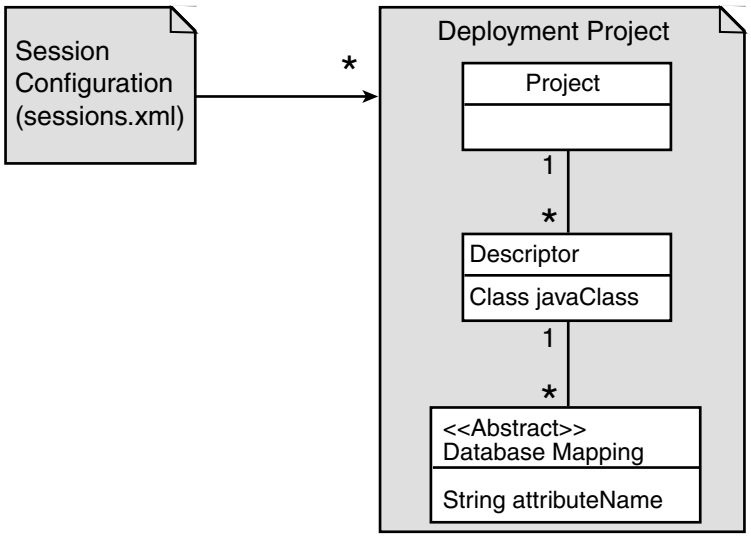
Note that this functionality is not limited to EJB architectures. Developers can configure any OracleAS TopLink architecture to use container-managed transactions.

OracleAS TopLink Metadata

The OracleAS TopLink approach to persistence is based on metadata that defines the class structure (objects) and relational schema, along with other configuration information used by OracleAS TopLink at runtime. Developers can use the OracleAS TopLink Mapping Workbench to define this metadata, and the OracleAS TopLink runtime component uses the metadata to provide the necessary persistence capabilities, using Java's reflective and introspective capabilities.

The TopLink application metadata model is based around the OracleAS TopLink project. The project includes descriptors, mappings, and various policies that customize the runtime capabilities. Figure 2–9 shows this metadata model.

Figure 2–9 OracleAS TopLink Metadata



Sessions XML Use the `sessions.xml` file to configure sessions for the project. Developers can build and edit these files with the OracleAS TopLink Sessions Editor. The session manager uses the `sessions.xml` configuration file during application initialization.

Project The OracleAS TopLink deployment project is the primary container for the metadata. A project generally represents an application and contains the mapping information for all persistent classes and their relationships. Each session (excluding the session broker) in the deployed application references a single project. Although developers can build a project by coding it using the OracleAS TopLink API, we recommend that developers create and manage the project in the OracleAS TopLink

Mapping Workbench, and use the OracleAS TopLink Mapping Workbench to generate either an XML or Java source version of the project for use at runtime.

Descriptor A descriptor represents the association between a persistent Java class and a relational table or tables. The descriptor contains configuration information for the class level within a project, as well as a set of mappings for each of its persistent attributes. Many of the more advanced configuration options are set at the descriptor level. The OracleAS TopLink Mapping Workbench supports most of these options, but there are a few that developers must set using the OracleAS TopLink API.

Mappings Mappings describe how the attributes of a mapped class are associated with columns in the database. OracleAS TopLink provides a sophisticated set of flexible and customizable mappings that allow for complex mapping scenarios between the object and relational models.

There are two types of mappings: direct mappings and relationship mappings.

- **Direct Mappings:** Direct mappings relate an attribute or attributes to a column or columns in the relational schema. OracleAS TopLink provides several direct mappings that allow for conversions between the types from the database and the object model's attribute types. Here are the direct mappings and their functions:
  - **Direct-to-field mappings** map a Java attribute directly to a value database column.
  - **Type conversion mappings** explicitly map a database type to a Java type.
  - **Object type mappings** match a fixed number of database values to Java objects.
  - **Serialized object mappings** store large data objects, such as multimedia files and BLOBs, in the database.
  - **Transformation mappings** offer specialized translations between how a value is represented in Java and in the database, such as when developers map multiple fields into a single attribute.
- **Relationship Mappings:** OracleAS TopLink offers sophisticated relationship mapping, which enables developers to represent object relationships based on the database table columns and foreign keys. Here are the relationship mappings and their functions:
  - **One-to-one mappings** represent simple pointer references between two Java objects. The references use any of foreign keys, target foreign keys, or variable classes to define the pointer.
  - **Aggregate object mappings** represent the relationship between a given object and a target object. The objects have a strict one-to-one relationship, and all the attributes of the second object are retrievable from the same table as the owning object.
  - **Aggregate collection mappings** represent the relationship between a single-source object and a collection of target objects. Unlike one-to-many mappings, in which there must be a one-to-one back reference mapping from the target objects to the source object, there is no back reference required for the aggregate collection mappings, because the foreign key relationship is resolved by the aggregation (object and collection).
  - **One-to-many mappings** represent the relationship between a single source object and a collection of target objects.

- **Many-to-many mappings** represent the relationships between a collection of source objects and a collection of target objects. They require an intermediate table for managing the associations between the source and target records.
- **Object-relational mappings** are mappings that leverage databases that support object-relational entity storage within tables.

Application Development with OracleAS TopLink

Using OracleAS TopLink to build an application does not affect the choice of development tools or the creative process. However, OracleAS TopLink does influence how developers approach development. This section highlights some of the key areas in which using OracleAS TopLink affects application development. These areas exist, regardless of whether developers are building an application to support Java objects, EJB entity beans, or both.

Mapping

OracleAS TopLink maps the application's persistent entities to the database, using the descriptors and mappings developers build with the OracleAS TopLink Mapping Workbench. The OracleAS TopLink Mapping Workbench supports several approaches to project development, including:

- Importing classes and tables for mapping
- Importing classes and generating tables and mappings
- Importing tables and generating classes and mappings
- Creating both class and table definitions with mapping creation and model generation

The OracleAS TopLink Mapping Workbench supports all these options; however, the most common solution is to develop the persistent entities using a development tool, such as an integrated development environment (IDE) or modeling tool, and to develop the relational model through appropriate relational design tools. Developers then use the OracleAS TopLink Mapping Workbench to construct mappings that relate these two models.

The OracleAS TopLink Mapping Workbench does offer some facilities for generating persistent entities or the relational model components for an application; however, these utilities are intended only to assist in rapid initial development strategies, rather than complete round-trip application development.

> **See Also:** *Oracle Application Server TopLink Mapping Workbench User's Guide*

Session Management

Sessions are the primary interface between the application and OracleAS TopLink persistence capabilities. When developing an OracleAS TopLink application, developers must ensure that they properly initialize and manage the sessions.

When using EJB entity beans with container-managed persistence (CMP) or bean-managed persistence (BMP), the client code that modifies the entity beans does not access the OracleAS TopLink session directly. Instead, changes occur transparently, through integration with the container or through EJB callbacks.

Well-designed applications that employ Java objects as persistent entities use the session manager provided in the OracleAS TopLink API. This class initializes and manages the singleton session. Developers configure the session manager in the

`sessions.xml` file, which allows for easy configuration and customization of the deployed application.

Querying

OracleAS TopLink furnishes several object and data query types, and offers flexible options for query selection criteria, including:

- OracleAS TopLink expressions
- EJB QL
- SQL
- Stored procedures
- Query by example

With these options, developers can build any type of query. We recommend that developers use predefined queries to define application queries. Predefined queries are held in the project metadata and referenced by name. This simplifies application development and encapsulates the queries to reduce maintenance costs.

The OracleAS TopLink Mapping Workbench provides the simplest way to define queries. Developers can also build queries in code, using the OracleAS TopLink API.

If the application includes EJB entity beans, developers can code finders completely using EJB QL, which enables the application to comply with the J2EE specification. Alternatively, developers can use any of the other OracleAS TopLink query options. All querying options are available, regardless of the architecture or persistent entity type.

Transactions

In an OracleAS TopLink application, the Unit of Work ensures that OracleAS TopLink transactions comply with the transactional requirements of the application.

The Unit of Work is one of the most sophisticated and powerful components of the OracleAS TopLink Foundation Library. Although developers that use CMP or BMP entity beans do not use the OracleAS TopLink API to apply transactional changes to their persistent entities, the Unit of Work is used behind the scenes. Understanding how the Unit of Work behaves, and developing simple coding patterns to use it, are the keys to building efficient, maintainable applications.

Packaging and Deployment

Application packaging (for deployment in the host Java or J2EE environment) influences OracleAS TopLink use and configuration. For example, developers package a J2EE enterprise application in an Enterprise Archive (EAR) file. Within the EAR file, there are several ways to package persistent entities within Web Application (WAR) and Java libraries (JAR). How developers configure OracleAS TopLink depends, in part, on how they package the application and how they use the host application server class loader.

Monitoring and Performance Tuning

OracleAS TopLink enables developers to monitor functionality and performance throughout application development, testing, and quality assurance cycles. OracleAS TopLink offers many textual logging features, as well as the API required to implement custom logging strategies. Developers can use these features to ensure that the application behaves and performs as they expect.

OracleAS TopLink includes a performance profiler feature, available through the OracleAS TopLink Foundation Library API. This runtime feature tracks query execution time, which developers can use for performance analysis. This tool provides the information necessary to identify bottlenecks that hinder application performance.

OracleAS TopLink also offers a rich set of performance enhancement features. Understanding how to configure these features can have a strong influence on application performance, especially in the later phases of application development.

OracleAS TopLink Architectures Overview

OracleAS TopLink is designed to work in both Java and J2EE applications. Since it was first introduced, the flexibility OracleAS TopLink provides has led to its use in many architectural styles. This section introduces the five most common architectures associated with OracleAS TopLink. Although this section describes the architectures in relation to J2EE, OracleAS TopLink continues to fully support non-J2EE and Java applications as well.

Three-Tier

The three-tier (or J2EE Web) application is one of the most common OracleAS TopLink architectures. This architecture is characterized by a server-hosted environment in which the business logic, persistent entities, and the OracleAS TopLink Foundation Library all exist in a single Java virtual machine (JVM).

The most common example of this architecture is a simple three-tier application in which the client browser accesses the application through servlets, Java Server Pages (JSPs), and HTML. The presentation layer communicates with OracleAS TopLink through other Java classes in the same JVM, to provide the necessary persistence logic. This architecture supports multiple servers in a clustered environment, but there is no separation across JVMs from the presentation layer and the code that invokes the persistence logic against the persistent entities using OracleAS TopLink.

EJB Session Bean Facade

A popular variation on the three-tier application involves wrapping the business logic, including the OracleAS TopLink access, in EJB session beans. This architecture provides a scalable deployment and includes integration with transaction services from the host application server. Communication from the presentation layer occurs through calls to the EJB session beans. This architecture separates the application into different tiers for the deployment.

The session bean architecture can persist either Java objects or EJB entity beans.

EJB Entity Beans with CMP

OracleAS TopLink provides CMP support for applications that require the use of EJB entity beans. This support is available on the leading application servers. OracleAS TopLink CMP support provides the developer with an EJB 1.1 and 2.1 CMP solution transparent to the application code, but still offers all the OracleAS TopLink runtime benefits.

Applications can access OracleAS TopLink-enabled EJB entity beans using CMP directly from the client, or from within a session bean layer. OracleAS TopLink also offers the ability to use regular Java objects in relationships with EJB entity beans.

EJB Entity Beans with BMP

Another option for using EJB entity beans is to leverage OracleAS TopLink BMP in the application. This architecture enables developers to access the persistent data through the EJB API, but is platform-independent.

The BMP approach is portable—that is, after a developer creates an application, you can move it from one application server platform to another.

Two-Tier

A two-tier (or client-server) application is one in which the OracleAS TopLink application accesses the database directly. Although less common than the other architectures discussed here, OracleAS TopLink supports this architecture for smaller or embedded data processing applications.

Oracle JDeveloper

Oracle JDeveloper is a J2EE and XML development environment with end-to-end support for developing, debugging, and deploying business applications and Web services. To maximize developer productivity, JDeveloper provides a comprehensive set of integrated tools to support the complete development life cycle, including the industry's fastest Java debugger and innovative profiler, and CodeCoach tools for code performance analysis and improvement. JDeveloper simplifies J2EE development by providing wizards, editors, visual design tools, and deployment tools to create high-quality, standard J2EE components, including applets, JavaBeans, JavaServer pages, servlets, and Enterprise JavaBeans.

Oracle JDeveloper Features

Oracle JDeveloper is written entirely in Java and is certified for use with all major platforms, including Windows, Solaris, and Unix. The following some key features of Oracle JDeveloper.

- **CodeCoach Utility and Profiling Tools:** To assist developers with optimizing Java code, JDeveloper provides a code coaching utility and three integrated profiling tools. With CodeCoach, you can increase the robustness of your code by optimizing system resources. The three profiling modes enable you to create a statistical analysis of the performance of your application with respect to it functionality, both at compile time and runtime. You can also analyze your application's memory use in the Java heap, and the occurrence and duration of various events.
- **Extensible and Pluggable Integrated Development Environment:** JDeveloper provides a public Extension SDK, enabling its development environment to be extended and customized. Using this SDK, customers and partners can develop extensions and integrate them directly into the JDeveloper IDE. Developers can also share the technologies written with the Extension SDK at the JDeveloper Extensions Exchange at:

  `http://www.oracle.com/technology/products/jdev/htdocs/partners/addins/exchange/index.html`
- **Support for SQL, PL/SQL, and XML:** In addition to Java, JDeveloper provides native support for SQL, PL/SQL, and XML. This support includes syntax highlighting and code insight, as well as PL/SQL development, PL/SQL debugging, and SQL tuning. Additionally, JDeveloper provides direct access to the database, allowing you to view, create, modify, and delete tables, views, triggers, indexes, sequences, and more.

- **Debugging Support:** JDeveloper offers robust debugging support for both Java and PL/SQL. Debugging in these two environments is seamlessly integrated when using an Oracle database, providing the ability to step from Java code directly into PL/SQL code within the same debugging session. Other features include remote debugging, debugging multiple processes on multiple servers, tracking relevant data members in a smart data window, and a count of instantiations per class.
- **Business Intelligence Beans:** To allow the development of highly analytical business intelligence applications, JDeveloper has integrated Oracle Business Intelligence Beans (BI Beans) into its environment. BI Beans enable developers to build applications that take advantage of the new analytic capabilities available in the Oracle database. Using modular components and intuitive wizards, developers can build sophisticated BI applications quickly and easily.

Oracle Application Development Framework

Oracle Application Development Framework (Oracle ADF) is a runtime framework included with Oracle JDeveloper. Together with the design time tools JDeveloper provides, Oracle ADF allows J2EE application developers to partition application layers.

While developing and deploying J2EE applications involves many technologies and a diverse infrastructure, many of the design patters are common to distributed applications, and can be abstracted into a runtime framework. Without such a framework, the various J2EE and Web services technologies must be defined at the interface level.

One of the roles of Oracle ADF is to support deployed applications that fit into the robust, scalable, and flexible architecture of the J2EE platform. When developers work with Oracle ADF and develop applications in JDeveloper, they can work with these familiar technologies to write code while benefiting from the J2EE platform infrastructure:

- Presentation tier: JSP pages for Web applications and Swing UI components for the standalone Java client
- Business tier: JavaBeans, EJB Session Beans, and Web services
- Integration tier: EJB Entity Beans

Partitioning Application Development Using the Model-View-Controller Design Pattern

The Model-View Controller (MVC) pattern is a J2EE architectural design pattern useful for developing applications that combine distributed application logic and a complex user interface. The goal when using MVC is to enforce partitioning between the application logic and the user interface.

Generally, the model is the underlying logical representation, the view is the visual representation, and the controller specifies how to handle user input. When the data model changes, it notifies all of the views that depend on it. This separation of state and presentation means that multiple views can be based on the same model, and views can be created and modified without affecting the underlying model.

Together with JDeveloper design-time tools, Oracle ADF allows J2EE application developers to partition application layers based on desired technologies. Developers are able to do the following:

- Partition applications based on J2EE-compliant objects using the Oracle ADF technology stack, including Oracle ADF Business Components, Oracle ADF Model, Struts Web application controller, and Oracle ADF view technologies.
- Partition applications based on the components of J2EE BluePrints (such as JSP, Jakarta Struts, and EJBs) while using only the data binding services of the Oracle ADF Model layer to simplify access to your existing business services from both the view and controller.
- Partition applications that combine your existing business services with Oracle ADF views, Struts Web applications controller, and the Oracle ADF Model data-binding layer.

Developers can use as much or as little of Oracle ADF as they choose when building components using the MVC architectural design pattern for J2EE applications. Oracle ADF does not prevent an application development team from mixing any layer of Oracle ADF with any J2EE component or any legacy business service.

Oracle Application Server Web Services

To enable **e-business**es to work effectively, the Internet needs to support a standards-based infrastructure that enables companies and their enterprise applications to communicate with other companies and their applications more efficiently. These standards should allow discrete business processes to expose and describe themselves on the Internet, allow other services to locate and invoke them, and provide a predictable response.

Web services drive this transformation by promising a fundamental change in the way businesses function and enterprise applications are developed and deployed. A Web service is a discrete business process that does the following:

- **Exposes and describes itself:** A Web service defines its functionality and attributes so that other applications can understand it. A Web service makes this functionality available to other applications.
- **Allows other services to locate it on the Web:** A Web service can be registered in an electronic Yellow Pages, so that applications can easily locate it.
- **Can be invoked:** Once a Web service has been located and examined, the remote application can invoke the service using an Internet standard protocol.
- **Returns a response:** When a Web service is invoked, the results are passed back to the requesting application over the same Internet standard protocol used to invoke the service.

Web services provide a standards-based infrastructure through which any business can do the following:

- Offer appropriate internal business processes as value-added services that can be used by other organizations
- Integrate its internal business processes and dynamically link them with those of its business partners

Oracle Application Server Web Services Architecture

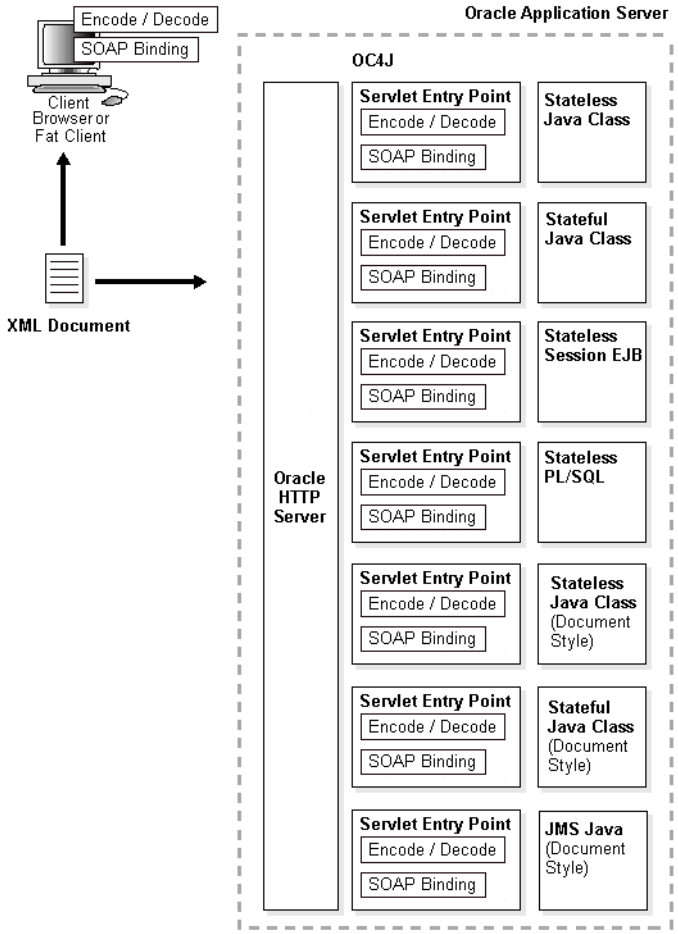
Oracle Application Server Web Services run as **servlet**s in the **OC4J servlet container**. This gives Web services the same **scalability**, **availability**, and load balancing facilities that all **J2EE** applications have in Oracle Application Server.

See Also: "Oracle Application Server Containers for J2EE Containers" on page 2-12

Oracle Application Server Web Services support both Remote Procedure Call (RPC) style exchange and message oriented, or Document Style, exchange. Supported RPC Web services include Java classes, stateless session EJBs, and stateless PL/SQL stored procedures. Supported Document Style Web services include Java Class Document Style Web Services and JMS Document Style Web Services.

Oracle Application Server Web Services use a different J2EE standards-compliant servlet for each implementation type to provide an entry point into the Web services implementation. Figure 2–10 illustrates the Oracle Application Server Web Services runtime architecture, including the servlet entry points.

Figure 2–10 Oracle Application Server Web Services Architecture



For a detailed discussion of the Oracle Application Server Web Services architecture, see the *Oracle Application Server Web Services Developer's Guide*.

Oracle Application Server Web Services Features

Oracle Application Server Web Services provides advanced runtime features and comprehensive support for developing and deploying Web services.

Oracle Application Server Web Services Development Features

Key Oracle Application Server Web Services development features include:

- **Development Environment:** Oracle Application Server Web Services allows application developers to implement Web services using J2EE components. In addition, you can use Java classes or PL/SQL stored procedures to implement Web services. Web services inherit all the runtime and lifecycle management elements of J2EE applications.
- **Development Tools and Wizards:** Oracle Application Server Web Services developers can use the same set of command line utilities to create, package, and deploy Web services that they use for other Oracle Application Server Containers for J2EE (OC4J) applications. In addition, Oracle Application Server Web Services provides the Web Service HTML/XML Streams Processing Wizard that assists developers in creating an EJB whose methods access and process XML or HTML streams.
- **Automatically Generates WSDL:** Oracle Application Server Web Services can generate Web Services Description Language (WSDL) and client-side proxy stubs. This generation occurs when the Web service is assembled using the Web Services Assembly tool or, for a deployed Web service, the first time the WSDL or the client-side stubs are requested. After the first request, the previously generated WSDL or client-side proxy stubs are sent when requested.
- **Registration, Publishing, and Discovery:** Oracle Application Server Web Services provides a standards-compliant UDDI registry where Web services can be published and discovered. The Oracle UDDI registry supports both a private and public UDDI registry and can also synchronize information with other UDDI nodes.
- **Developer Simplicity:** Using Oracle Application Server Web Services, developers do not need to learn a completely new set of concepts. Web services are developed, deployed, and managed using the same programming concepts and tools as J2EE applications.
- **Business Logic Reuse:** Application developers can transparently publish their J2EE applications to new Web services clients with no change in the application itself. The existing business logic developed in J2EE can be transparently accessed from existing J2EE/EJB clients.
- **Common Runtime Services:** Oracle Application Server has a common runtime and brokering environment for J2EE applications and Web services. As a result, Web services transparently inherit various services available with the J2EE container, including transaction management, messaging, naming, logging, and security services.

Oracle Application Server Web Services Deployment and Management Features

Key Oracle Application Server Web Services deployment and management features include:

- **Packaging and Assembly:** The Web Services Assembly Tool assists with assembling Web services and producing a J2EE `.ear` file.

- **Deployment:** Oracle Enterprise Manager provides a comprehensive set of facilities for deploying Web services to Oracle Application Server through Oracle Enterprise Manager Application Server Control. Application Server Control provides a single, consistent *Deploy Application* page for deploying Web services to Oracle Application Server. It accepts a J2EE `.ear` file, and takes you through the steps to specify information about the application you are deploying, and then deploys the application.
- **Register Web Service:** The *Deploy Application* page also provides access to facilities for registering Web services in the UDDI registry.
- **Browse the UDDI Registry:** The Oracle UDDI registry provides the UDDI standards-compliant, pre-defined, hierarchical categorization schemes. Oracle Enterprise Manager can drill down through these categories and look up specific Web services registered in any category.
- **Monitoring and Administration:** Once deployed, Oracle Enterprise Manager provides facilities to deinstall a Web service, and also to monitor Web service performance as measured by response time and throughput, and to monitor status as measured by up time, CPU consumption, and memory consumption. Oracle Enterprise Manager also provides facilities to identify and list all of the Web services deployed to a specific Oracle Application Server instance.

Oracle XML Developer's Kit

XML makes data portable and interoperable across heterogeneous systems. XML is a metamarkup language that supports markup tags that have been defined by the user to encapsulate and describe data in a Web page. Because it is a self-defined language, it is extremely flexible.

Oracle Application Server includes the **Oracle XML Developer Kit** (XDK) to support development of any applications that use XML. The XDK provides basic XML infrastructure components for manipulating and transforming XML documents. The XDK contains component libraries and utilities that are based on World Wide Web Consortium (W3C) specifications. You can use these components to generate, manipulate, render, and store XML-formatted data in an Oracle database or to share data between applications written in diverse programming languages.

Oracle XML Developer Kit Tools

The following sections describe the tools included in the Oracle XML Developer's Kit.

XML and XSLT Parsers

The XML parser **API**s are defined by the W3C specifications so that application developers can use standard programming interfaces. However, if some functionality is not specified, then Oracle Corporation may implement its own enhancements to the specifications.

The Oracle XML and XSLT parsers provide international **character set** and multithreaded support, allow optional validation, and cache document type definitions (**DTD**s) and stylesheets for performance.

Oracle XML Developer's Kit provides the following APIs:

- **Document Object Model (DOM) APIs:** Because an XML document is structured, in the sense that "start" tags have corresponding "end" tags and these tags are nested in an ordered fashion, an XML document can be viewed as a tree whose hosts consist of these tags and information between and corresponding to the tags.

When the DOM APIs are used to navigate an XML document, the XML parser parses the document and forms a tree representation of it in memory.

- **Simple APIs for XML (SAX) APIs:** SAX APIs are event-based, meaning that notification of certain events and data encountered during the parsing of an XML document can be reported with callback functions to the application program. One advantage over DOM is that an in-memory representation of the parse tree does not have to be built, thus saving memory and improving performance. When application programs are notified of these events, then they must handle them.
- **Namespace APIs:** These interfaces resolve namespace prefixes that are used to qualify element and attribute names in XML documents. XML document namespaces are identified by uniform resource identifier (**URI**) references that qualify element or attribute names and locate resources that may be on different machines or in different XML documents. Namespaces make it possible to have identical names for elements and attributes by qualifying them with URIs that differentiate the names.
- **Parser APIs:** Application programmers invoke the parser API to read an XML document and to provide access to its content and structure through DOM or SAX APIs. Typically, initialization and termination functions must also be invoked in association with the parse function. Various flags, such as to discard whitespace and turn on validation, can be set with initialization functions before the application program invokes the parse function.
- **XSLT APIs:** These interfaces read the input stylesheet file and transform the input XML document according to the stylesheet. A command-line interface to the integrated XSLT processor is provided. This allows application programmers to specify the number of threads to use to parse and transform XML documents and other useful options. To enhance performance, the XSLT engines allow **caching** of both stylesheets and Document Type Definitions (**DTD**s) so they can be reused for multiple XML documents.

XML Schema Processors

Oracle XML schema processors comply with the Structures and Datatypes sections of the W3C XML Schema Working Drafts, except for certain features (such as unique, key and keyref constraints) when the SAX parser is used. XML schemas are a superset of DTDs, except for the support of primitive and complex datatypes in XML schemas. This allows you to validate XML documents, with embedded datatype information, against XML schemas.

XML Class Generators

The XML class generators create a set of **Java** or C++ classes that create XML documents corresponding to a DTD or to XML schema definitions. This is useful when an application wants to send an XML **message** to another application based on an agreed-upon DTD, or as the back end of a Web form to construct XML documents. You can use the generated classes to programmatically construct XML documents that comply with the DTD or schema definition.

XSQL Servlet

XSQL Servlet is a tool that processes **SQL** queries and outputs the result set as an XML document. This processor uses an XML file with embedded SQL queries as its input. You can use XSQL Servlet to perform the following tasks:

- Build dynamic XML datapages from the results of one or more SQL queries and serve the results over the Web as XML datagrams or **HTML** pages using server-side XSLT transformations
- Receive XML posted to your Web **server** and insert it into your database

XML Transviewer Beans

XML transviewer beans are a set of XML components that constitute XML for Java beans. These are used by Java applications or applets to view and transform XML documents.

These beans are visual and non-visual Java components that are integrated into Oracle JDeveloper to enable the fast creation and deployment of XML-based database applications.

Oracle Application Server PL/SQL Platform

Oracle Application Server provides support for building **PL/SQL**-based applications on the Web. PL/SQL **stored procedure**s retrieve data from a database and generate **HTTP** responses containing data and code to display in a Web browser. There are many techniques for coding dynamic pages, but PL/SQL is particularly suited for producing dynamic pages based on database processing. Its support for DML statements, dynamic SQL, cursors, and tight server integration provide both power and flexibility for Web applications. The dynamic pages can contain links and HTML forms that call more stored procedures, for example to drill down on the displayed data. The set of interlinked HTML pages forms the user interface of the Web application. Producing dynamic content using PL/SQL stored procedures gives you the flexibility and interactive behavior of **CGI** programs, without the memory overhead of starting a new CGI process each time.

Oracle Application Server PL/SQL Tools

Oracle Application Server PL/SQL developers can create applications using either Oracle PL/SQL Server Pages or Oracle PL/SQL Web Toolkit.

mod_plsql

Mod_plsql is an Apache plug-in that provides support for building PL/SQL-based applications on the Web. It maps browser requests into PL/SQL stored procedures, which retrieve data from a database and generate HTTP responses containing data and code to display in a Web browser.

Oracle PL/SQL Server Pages

To include dynamic content inside Web pages, including the results of SQL queries, you can use server-side scripting through PL/SQL Server Pages (PSPs). You can author the Web pages in a script-friendly HTML authoring tool and drop the pieces of PL/SQL code into place. You may find this technique more convenient than using the HTP and HTF **package**s to write out HTML content line by line.

Because the processing is done on the database server rather than the Web server, the browser receives a plain HTML page with no special script tags, and you can support all browsers and browser levels equally. It also makes network traffic efficient by minimizing the number of server roundtrips.

Embedding the PL/SQL code in the HTML page that you create lets you write content quickly and follow a rapid, iterative development process. You maintain central control of the software, with only a Web browser required on the **client** machine.

Oracle PL/SQL Web Toolkit

The PL/SQL Web Toolkit contains PL/SQL packages that allow you to create applications that generate dynamic HTML. Using the toolkit, you can access data and insert it into Web pages. The packages remove the need for you to know the specifics of HTML syntax and allow you to focus on writing applications.

Oracle Application Server PL/SQL Architecture

Oracle Application Server PL/SQL leverages the mod_plsql module to deliver Web content, as follows:

- Visiting a Web page, following a hypertext link, or submitting an **HTML** form causes the database server to run a stored procedure.
- Any choices that a user makes on an HTML form are passed as parameters to the stored procedure. Parameters can also be hardcoded in the **URL** used to invoke the stored procedure.
- The results of the stored procedure are printed as tagged HTML text and are displayed in the browser as a Web page. Web pages generated this way are dynamic: code runs inside the database server, producing HTML that varies depending on the database contents and the input parameters.

  This kind of dynamic content is different from dynamic HTML (DHTML). With DHTML, the code is downloaded as JavaScript or some other scripting language, and processed by the browser along with the HTML. A PL/SQL Web application can print JavaScript or other script code in its output to produce complex DHTML that would be tedious to produce manually.

Oracle Content Management Software Development Kit

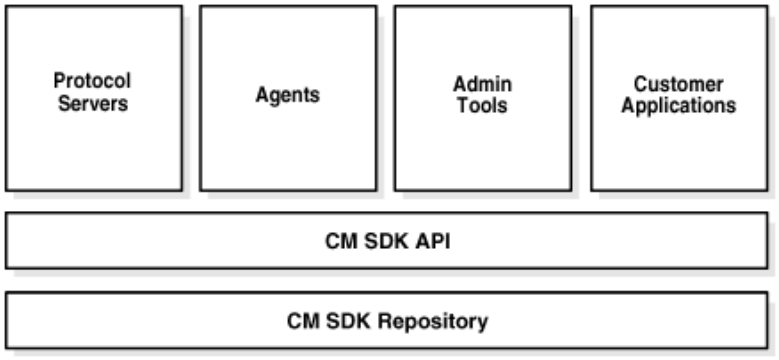
Oracle Application Server includes the Oracle Content Management Software Development Kit (CM SDK), a robust document lifecycle system that is built in Java and integrated with other Oracle content management API products. Oracle customers and partners have built successful solutions on this Java-based platform that leverages the features and capabilities of the Oracle platform.

Oracle Content Management SDK Architecture

Oracle CM SDK provides, out of the box, the content and metadata repository, protocol servers, and class libraries that allow developers to quickly create content management solutions.

Figure 2–11 shows the basic architecture of Oracle CM SDK. Once installed, developers can immediately start storing content into the repository via standard file system protocols such as FTP or WebDAV. They can then provide the business logic for their specific requirements and create a Web-based user interface. Also included with the Oracle CM SDK installation is a Web starter application that demonstrates how to create Web-based content management applications.

Figure 2–11 Oracle Content Management SDK Architecture



Oracle Content Management SDK Features

Important features and capabilities of Oracle CM SDK include the following:

- Integration with Oracle Text, Oracle Workflow, and Oracle *inter*Media
- Complete document lifecycle API with a variety of options for security, check-in / check-out, versioning, searching, extensible metadata, and other standard content management operations
- Easy extensibility to help you develop and deploy your content management solutions faster
- Managed through Oracle Enterprise Manager Application Server Control
- Synchronous and asynchronous trigger mechanisms to enforce business rules
- Built-in support for standard protocols such as HTTP / WebDAV, SMB / NTFS, FTP, NFS, and IMAP4 / SMTP
- Support for standard Oracle performance, scalability, and high availability features such as RAC

Oracle Application Server MapViewer

Oracle Application Server MapViewer is a programmable tool for rendering maps using spatial data managed by Oracle Spatial or Oracle Locator. MapViewer provides tools that hide the complexity of spatial data queries and cartographic rendering, while providing customizable options for more advanced users. These tools can be deployed in a platform-independent manner and are designed to integrate with map-rendering applications.

The following concepts are fundamental to understanding MapViewer:

- **Style:** A style defines the rendering properties for features that are associated with the style. For example, a text style determines how a feature is labeled on a map, while a line style determines the rendition of a linear feature such as a road.
- **Theme:** A theme is a collection of features (entities with spatial and nonspatial attributes) that are associated with styles through the use of styling rules.
- **Base map:** A base map consists of one or more themes.
- **Mapping metadata:** Mapping metadata consists of a repository of styles, themes, and base maps stored in a database.

- **Map:** A map is one of the components that MapViewer creates in response to a map request. The map can be an image file, the object representation of an image file, or a URL referring to an image file.

When an application uses MapViewer, it applies specific styles, such as colors and patterns, to specific themes, such as cities, rivers, and highways, to render a map. For example, the application might display a map in which state parks appear in green and restaurants are marked by red stars. A map typically has several themes representing political or physical entities, or both. For example, a map might show national and state boundaries, cities, mountain ranges, rivers, and historic sites. When the map is rendered, each theme represents a layer in the complete image.

MapViewer lets you defines styles, themes, and base maps, including the rules for applying one or more styles to each theme. These styles, themes, base maps, and associated rules are stored in the database in map definition tables under the MDSYS schema, and they are visible through metadata views. All styles in a database instance are shared by all users. The mapping metadata that you can access is determined by MapViewer metadata views. The set of map definition objects that a given user can access is sometimes called that user's mapping profile.

Oracle Application Server MapViewer Architecture

With MapViewer, the basic flow of action involves two steps, regardless of whether the client requests a map or a MapViewer administrative action.

For a map request, the following occurs:

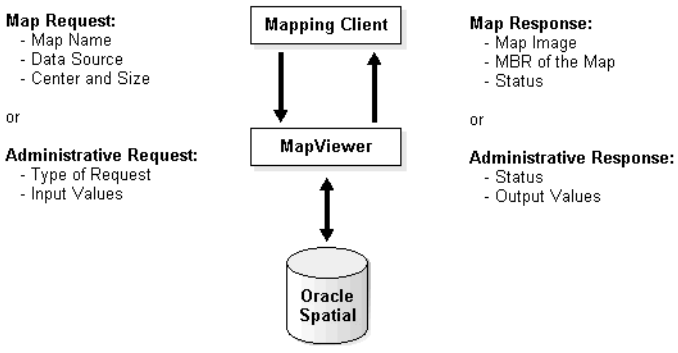
1. The client requests a map, passing in the map name, data source, center location, map size, and optional other data to be plotted on top of a map.
2. The server returns the map image (or a URL for the image), the minimum bounding rectangle (MBR) of the map, and the status of the request.

For a MapViewer administrative request, the following occurs:

1. The client requests a MapViewer administrative action, passing in the specific type of request and appropriate input values.
2. The server returns the status of the request and the requested information.

Figure 2–12 shows the MapViewer architecture, which supports this flow of action.

Figure 2–12 MapViewer Architecture



As shown in Figure 2–12:

- MapViewer is part of the Oracle Application Server middle tier.

- MapViewer includes a rendering engine.
- MapViewer can communicate with a client Web browser or application using the HTTP protocol.
- MapViewer performs spatial data access (reading and writing Oracle Spatial and Locator data) through JDBC calls to the database.
- The database includes Oracle Spatial or Locator, as well as mapping metadata.

Oracle Application Server MapViewer Features

MapViewer includes the following main components:

- A rendering engine (Java class library) that provides cartographic capabilities (a map renderer)
- An XML API that provides a programmable interface to MapViewer

The rendering engine connects to the Oracle database through Java Database Connectivity (JDBC). It also loads the map metadata from the database, and applies it to the retrieved spatial data.

The XML API provides high-level application developers with a convenient interface for submitting a map request to the middle-tier MapViewer and handling its map response.

3

Portal Applications

This chapter provides an overview of **Oracle Application Server Portal** features and benefits. The topics include:

- Introduction to Oracle Application Server Portal
- Oracle Application Server Portal Architecture
- Oracle Application Server Portal Features
- Application Access and Integration

Introduction to Oracle Application Server Portal

Portals allow **client**s to access information through a Web browser. This information usually comes from different data sources that the portal makes available through a single entry point. That entry point is known as a page.

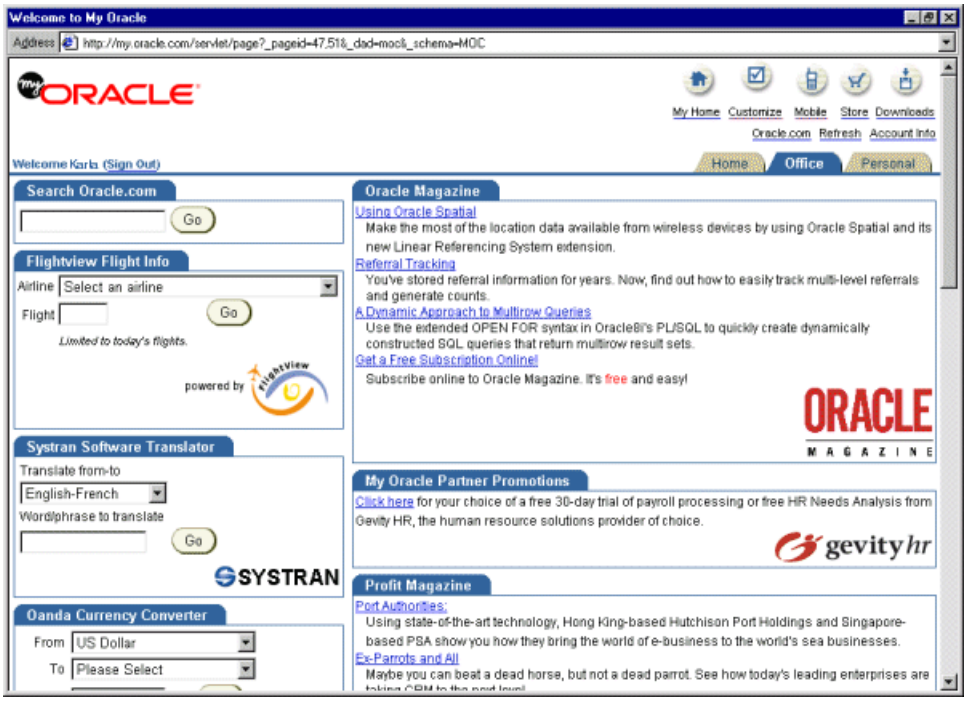
Portals also support personalized views, so that each user or user group can customize both the content and the appearance of the portal to suit individual preferences and requirements.

For example, a financial analyst's page would likely include information from real-time Internet-based stock quotes, financial reports from an online repository, and access to legacy financial accounting and banking systems. The data from these systems are independent of each other, but the portal allows them to exist within a single page.

What is Oracle Application Server Portal?

Oracle Application Server Portal is a Web-based tool for building and deploying **e-business portal**s. It provides a secure, manageable environment for accessing and interacting with enterprise software services and information resources. A portal page makes data from multiple sources accessible from a single location. Figure 3–1 shows a sample portal page from `http://my.oracle.com`. Each one of the tabbed areas within the Office tab contains information from a different data source.

Figure 3–1 Sample Portal Page



E-Business Support with Oracle Application Server Portal

Many organizations are turning to portals as key components of their e-business strategy. Portals are emerging as essential problem-solving mechanisms that provide a single source of interaction with all corporate information and the focal point for conducting day-to-day business. Companies are using portals and supporting applications in many ways, including:

- To provide access to instantly available, personalized, job-based information
- In forums for groups to exchange, analyze, and discuss ideas
- To automate business processes that integrate into daily activities

OracleAS Portal supports e-businesses by doing the following:

- **Providing secure access to existing information, no matter where it is:** Portal developers can organize and structure content in a consistent and logical way. This reduces or eliminates the need for users to hunt for information through a variety of sources.
- **Supporting personalized views:** Users and user communities can organize the information they access in ways that complement their work habits or interactions.
- **Providing self-service to users:** Employees, partners, and suppliers who have important information to share with the community can do so without specific technical skills or help from a technology expert.
- **Enabling single sign-on:** Users can log into the portal once per session and access all internal and external applications without logging into any of them.

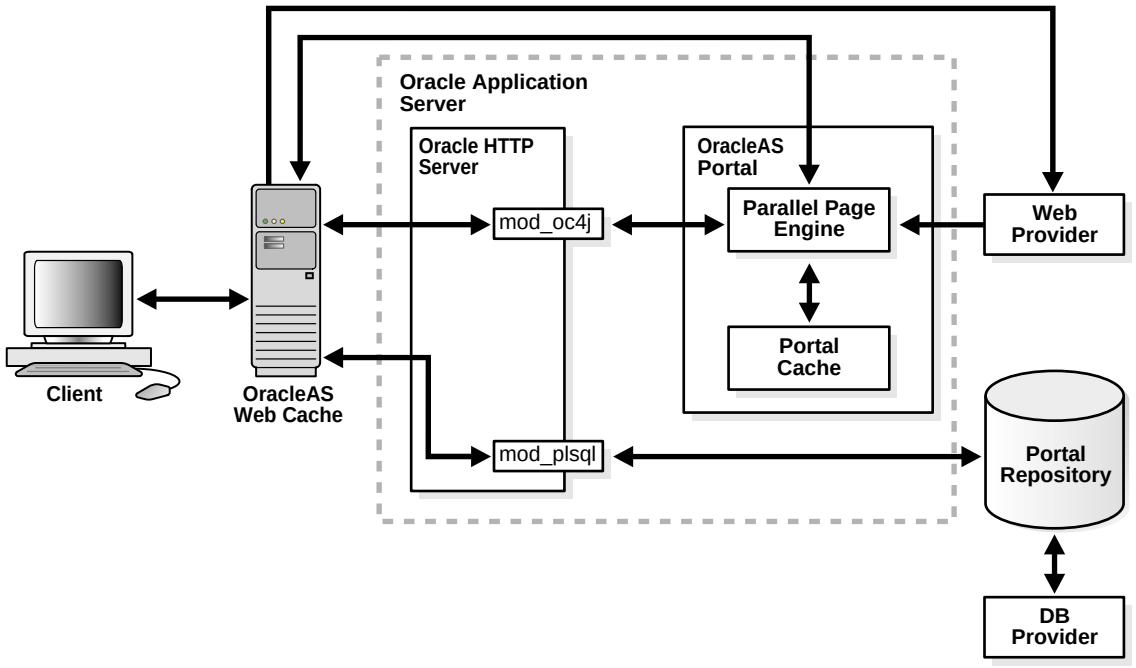
Oracle Application Server Portal Architecture

When a user requests an **Oracle Application Server Portal** page, many Oracle Application Server components service parts of the request. Requests have the following flow:

1. The browser requests a **portal** page. **Oracle Application Server Web Cache** receives this request.
2. OracleAS Web Cache forwards the request to the OracleAS Portal Parallel Page Engine (PPE) through Oracle HTTP Server.
3. The PPE retrieves the portal page definition, and coordinates with all of the portlet providers to see if there is a cached copy available or if fresh information is needed.
4. The PPE aggregates all of the portlet content into a single page. This page is sent to OracleAS Web Cache.
5. OracleAS Web Cache returns the final page to the browser.

Figure 3–2 illustrates this flow in terms of the OracleAS Portal architecture.

Figure 3–2 OracleAS Portal Request Flow



Oracle Application Server Portal Features

Key features of **Oracle Application Server Portal** include the following:

- **An extensible framework:** OracleAS Portal has an extensible framework that integrates Web-based resources such as Web pages, applications, business intelligence reports, and syndicated content feeds within standardized, reusable information components called **portlet**s. Within a portlet, these resources are personalized and managed as a service of OracleAS Portal. Companies can create their own portlets for their existing Web resources and can select additional portlets from the growing catalog of third-party portlet providers. Additionally,

Oracle Application Server provides many ready-made portlets out-of-the-box that are available for immediate use. The **portal** framework provides additional services including single sign-on, content classification, enterprise search, directory integration, and access control.

- **An easy-to-use, personalized interface:** The OracleAS Portal interface provides an organized, consistent view of the business information, Web content, and applications that each user needs. Portal administrators and designers use a browser-based control panel to selectively grant access to applications and information by making portlets available only to specific users or user groups.
- **Self-service publishing:** OracleAS Portal self-service publishing features allow authorized users to post and share any kind of document or Web content with other users anywhere in the world. Content contributors use controls for uploading documents, implementing version control, customizing page formatting and display, limiting access, and managing content without any requirement for technical expertise or **HTML** knowledge.
- **A scalable deployment architecture:** The OracleAS Portal architecture is easily configured for departmental, regional, and enterprise-wide deployment. The deployment model supports a variety of configurations, including single **host** and multi-tier, on a broad set of hardware platforms and operating systems.
- **Integration with Oracle Application Server Web Cache:** OracleAS Web Cache provides caching, compression, and assembly features to accelerate the delivery of both static and dynamically-generated Portal content.

Portal Page Creation, Management, and Customization

OracleAS Portal incorporates a portal creation and deployment framework. The framework defines Web information sources as information components, and assembles these components within a portal page. It also supports customization of the Web page to one or more user communities.

Each portal page is divided into either item regions or portlet regions. Item regions allow you to add text, images, and files to a portal page. Portlet regions provide an area where you can place one or more portlets.

A portlet is an HTML or **XML** area that summarizes, promotes, or provides basic access to an information resource. The information resources can take on many forms and can serve many purposes.

Items and portlets are the fundamental building blocks of an OracleAS Portal page. Page owners create their own portal pages, and page owners can either maintain their own pages or delegate maintenance responsibilities to other users. Each portal page consists of content presented through one or more items, portlets, and links that allow the **client** to navigate to another page or take some action.

Portal Content Publishing and Management

OracleAS Portal provides an integrated set of features for self-service document publishing, file upload, page formatting, and access control. Collaborators and content publishers no longer need specific technical skills or a Webmaster to publish their content. Instead, they can use an item region on a portal page to publish their content and format content appearance using simple controls.

An item region includes built-in features for publishing, organizing, classifying, cross-referencing, and displaying the content it manages. Key components that make up an item region include:

- **Items that are the pieces of content themselves:** Items are defined by the base content that makes up the item and one or more attributes that describe the item. Item publishers create items by completing a series of steps in a wizard. Additional features, such as item version control, check-in and check-out, expiration, and automatic indexing support collaborative document creation and ease content management tasks.
- **Styles that define how items are displayed**: Item region style properties govern the colors, font properties, size, background images, banners, and other graphical elements for the items and navigation bars in an item region.

Item regions also include components that assist users in navigating or locating content of interest and allow for classifying content with categories and perspectives.

OracleAS Portal provides categories and perspectives as a means of applying a classification to the content (items and portlets) you add to your portal. Categories and perspectives can be used to locate content during searches. Categories are used to describe the type of content you are adding. For example, if you are creating a Human Resources page, you might have categories such as Benefits, Policies, and Payroll. Perspectives are used to describe the type of audience who might be interested in the content, such as managers, supervisors, and non-exempt employees.

Content Searching

OracleAS Portal searches can quickly and easily locate information managed within a portal page. OracleAS Portal supports the following search methods:

- **Basic searches** only match attributes of items in the current portal page against the search criteria. A basic search compares the search criteria to the name, author, description, and keywords of all items within the selected page group.
- **Advanced searches** allow the user to define or restrict the search by specific attributes. Users can specify that their searches do the following:
  - Return matches to any or all search terms
  - Search within a specific portal page or across all portal pages
  - Restrict searches to a particular category, perspective, item type, or attribute
- **Oracle Ultra Search searches** perform comprehensive searches against portal-managed information. The Ultra Search engine searches through all content that has been indexed using Ultra Search, not just portal metadata. These searches can also return matches based on near matches (terms that appear close together), soundex matches (terms that sound like the search term), and fuzzy matches (terms with similar spellings to the search term).

A search result portlet displays the matches of any search. You can add this portlet to any portal page and customize which attributes it displays.

Portals and Wireless Devices

In addition to standard Web browsers, wireless users can also access OracleAS Portal pages. Working with **Oracle Application Server Wireless**, the portal automatically transforms the portal page structure to a format appropriate for the smaller screens of most wireless devices. Only portlets generating Oracle Application Server Wireless XML content display on the wireless device.

OracleAS Portal developers also have access to a set of page design tools that help in creating portal pages that optimize the wireless experience. With these tools, developers can build a distinct portal structure for their wireless users. The wireless

pages and portal pages can share portlet instances. This allows portal pages to reuse portlets on browsers and wireless devices without reconfiguring each portlet.

Portal Integration with Oracle Application Server Single Sign-On

Oracle Application Server Portal leverages Oracle Application Server Single Sign-On to provide single sign-on capabilities for secure access to portal content and applications. OracleAS Single Sign-On works as a single, unified authentication service to all Oracle Application Server components, applications, and Web pages, storing user information and authenticating users against Oracle Internet Directory.

See Also: Chapter 11, "Security and Identity Management"

Application Access and Integration

Portal **client**s access **Oracle Application Server Portal** applications through **portlet**s. Users can select the portlets that appear on their page from a list of providers registered with OracleAS Portal. Additionally, developers can use the Oracle Application Server Portal Developer Kit to create their own portlets.

Integrating with Portlet Providers

Applications and information sources, represented as portlets, communicate with the **portal** through a provider. Each portlet only has one provider, and a provider can have one or more portlets that expose an underlying application or information source.

All portlets from portlet providers make use of Oracle Application Server Single Sign-On, regardless of their location. It is not necessary for portlets to be deployed within Oracle Application Server or even on the same hardware. This ensures that only authorized users are able to subscribe to a particular portlet and that authorized users can access all registered portlets by logging into their main portal page.

Oracle Application Server Portlets

Some Oracle Application Server components act as portlet providers to OracleAS Portal. This allows you to easily integrate information from various Oracle Application Server components into a single portal page.

Oracle Business Intelligence Discoverer As a portlet provider, **Oracle Business Intelligence Discoverer** offers worksheet portlets and list of workbooks portlets to OracleAS Portal users. A worksheet portlet contains information from a single Discoverer worksheet. The portlet displays this information in either a table, a graph, or both. The list of workbooks portlet presents a list of available workbooks.

See Also: Chapter 5, "Oracle Business Intelligence"

Partner Portlets

In addition to the list of Oracle Application Server components, a growing community of independent software vendors (ISVs) and Internet content providers are creating standard, supported portlets that access their applications and services. For these partners, customers can access the partner's application or service through one or more pre-integrated portlets.

The current catalog of portlets includes the following services:

- Business intelligence and reporting (including Axis Technology and Quest Software)

- Collaboration (including Cubika Internet Technology and SiteScape)
- Document, content, and knowledge management (including Interwoven and STI AS)
- E-business applications, such as **customer relationship management** and enterprise resource management (including Billboard and Droplets)
- News and information sources (including NT-Exchange.com)
- Portal tools (including Curl Corporation and Oracle Application Server Portal Developer Kit)
- Internet searches (including Business Objects and Quest Software)

> **See Also:**
> `http://www.oracle.com/technology/products/ias/portal/` for a complete list of Partner portlets

Custom Portlets

The Oracle Application Server Portal Developer Kit (PDK) allows developers to either reuse existing applications as portlets or create new portlets. Developers can write portlets using familiar languages and technologies. These technologies can be as follows:

- **Java** applications
- Oracle **PL/SQL package**s
- Web pages built with any standard technology like Active Server Pages (ASPs) or Perl
- Web services

> **See Also:**
> `http://www.oracle.com/technology/products/ias/portal/` for more information on the Oracle Application Server Portal Developer Kit

Additionally, PDK developers can take advantage of the Knowledge Exchange, available through Portal Studio, to leverage portlets created by other Oracle Application Server Portal users.

Oracle Application Server Portal Integration with Oracle Application Server Web Cache

Oracle Application Server Portal is closely integrated with Oracle Application Server Web Cache to improve the overall availability, scalability, and performance of OracleAS Portal. OracleAS Web Cache combines caching, compression, and assembly technologies to accelerate the delivery of both static and dynamically generated Portal content.

OracleAS Portal functions as a web cache origin server to take advantage of OracleAS Web Cache features such as fine-grained cache control and load balancing.

> **See Also:** Chapter 9, "Performance and Caching" for more information about Oracle Application Server Web Cache

Oracle Application Server Web Cache Deployment with Oracle Application Server Portal

When you install Oracle Application Server Portal, an OracleAS Web Cache instance is automatically created with pre-defined cache configuration settings. Portal sites can choose from the following deployment options:

- **Co-located:** OracleAS Web Cache runs on the same physical server as the Portal middle tier. This configuration is appropriate for smaller, low-volume sites where the scalability of the middle tier is not a concern.
- **Dedicated:** OracleAS Web Cache is deployed on a dedicated server that sits in front of one or more Portal middle-tier servers. Dedicated deployments are usually preferable to co-located deployments, as there is no risk of resource contention with other server processes. OracleAS Web Cache performs well on commodity hardware, so a dedicated deployment does not have to be costly in terms of hardware expenditure.

For very high-volume sites and to avoid a single point of failure, two or more nodes running OracleAS Web Cache may be deployed behind a third-party network load balancer. If you have multiple deployments of OracleAS Portal, each Portal site can have its own OracleAS Web Cache server, or one or more sites can share a single Web Cache. Similarly, a Web Provider can share a Web Cache with a Portal site, or a dedicated Web Cache can be deployed in front of the Web server that hosts the Web Provider.

A browser-based console, Oracle Application Server Web Cache Manager, is used to administer all aspects of OracleAS Web Cache, including configuration of caching and load balancing rules, security, manual and automated invalidation, monitoring, and logging.

4

Wireless Applications

This chapter provides an overview of **Oracle Application Server Wireless** features and benefits. The topics include:

- Introduction to Oracle Application Server Wireless
- Oracle Application Server Wireless Architecture
- Oracle Application Server Wireless Features

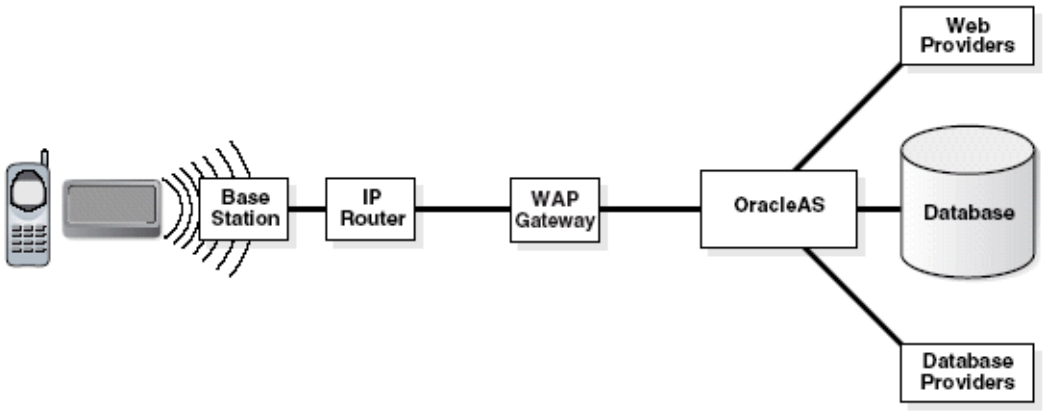
Introduction to Oracle Application Server Wireless

Wireless computing extends the e-business infrastructure to new classes of devices, delivering on-demand information wherever it is needed, using any device. Oracle Application Server Wireless enables access to information, applications, and services by delivering information that users want, when they want it, on the mobile device of their choice, thereby making them more productive and saving the enterprise money. It provides a solution to the critical enterprise requirement to provide mobile workers with access to their email, calendars, and key enterprise information.

A component of Oracle Application Server, Oracle Application Server Wireless enables enterprises and service providers to efficiently build, manage, and maintain wireless and voice applications. OracleAS Wireless makes Web and database applications (such as e-mail, news, and directory services) accessible to mobile device users by enabling optimal device experience through a common application development framework and environment.

It renders the same application to multiple target markup languages, including HTML, WML, HDML, cHTML, VoiceXML, SMS, and XMPP. Figure 4–1 shows the flow of information between a wireless client and enterprise information systems.

Figure 4–1 Oracle Application Server Wireless



Oracle Application Server Wireless Architecture

In order to set up an Oracle Application Server Wireless environment, you must have the following:

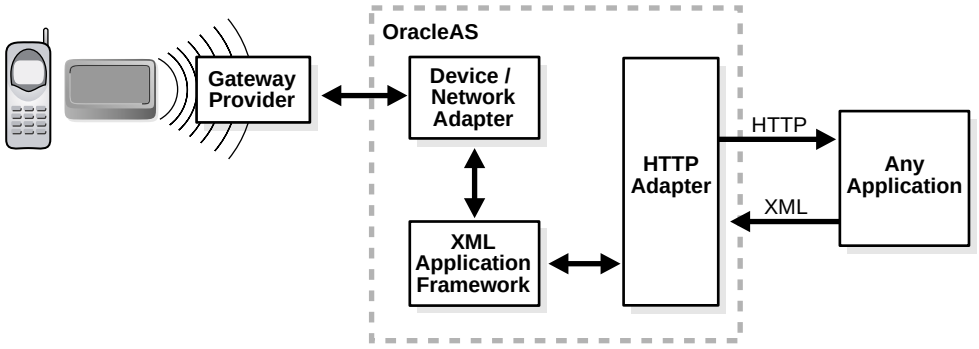
- Content or service provider application
- Oracle Application Server Wireless
- Gateway (such as WAP, SMS)
- Wireless network provider
- Wireless devices

When users request wireless service, the following occurs:

1. The wireless device connects to the gateway, passing the URL of the requested service.
2. The gateway collects the device profiles, such as subscriber ID, device ID, user agent profile from the wireless network.
3. The gateway submits a URL request to OracleAS Wireless, passing the device profiles.
4. OracleAS Wireless normalizes the request and forwards the request to the target URL.
5. OracleAS Wireless transforms the response from the target URL to the requesting device markup language.
6. The gateway sends the response from OracleAS Wireless to the device.

Figure 4–2 illustrates this flow in terms of the **Oracle Application Server Wireless** architecture.

Figure 4–2 Oracle Application Server Wireless Architecture



Oracle Application Server Wireless Server

OracleAS Wireless normalizes the request from any channel and forwards the request to the target URL using the HTTP or HTTPS protocols. It then retrieves the content (which can be in MobileXML, XHTML / MP or XForms). The server renders the device-independent content to any target device.

If the content is not accessible through HTTP or HTTPS, the wireless application container provides the API for you to plug in your own protocol adapters, which can retrieve content from non-HTTP sources (such as PL / SQL applications, SQL queries, LDAP, and IMAP). OracleAS Wireless expects the response from the adapters in MobileXML, XHTML / MP, or XForms.

Oracle Application Server Wireless Transformers

Oracle Application Server Wireless transformers transform device-independent markup into device-specific markup. The transformers render the same content into a functional, actionable page on any device. The transformers use the device knowledge base to activate the device-specific capability as it renders the page.

The device and network adapters automatically transform and optimize application content for any wireless device and network. The adapters support the following mobile technologies:

- 2-way pagers for asynchronous services (SMTP / SMS)
- WAP devices
- Voice access through regular phone lines
- PDA devices

OracleAS Wireless provides these types of transformers:

- **Generic service transformers:** Support languages such as WML. Generic service transformers convert Mobile XML to a generic WML format that works on any AP-compliant wireless phone.
- **Device-specific transformers:** Optimized for specific devices. For example, instead of using the generic WML transformer, you can use a device-specific transformer that exploits the device characteristics of a specific phone. The OracleAS Wireless initial repository includes transformers for several target formats, including CHTML, HDML, MML, and VoiceXML.
- **Custom transformers:** Create custom transformers to target new device platforms and optimize content presentation for specific devices. OracleAS Wireless

publishes device transformation rules so that anyone can create support for any type of device and markup language.

Oracle Application Server Wireless Features

Oracle Application Server Wireless 10*g* (10.1.2) includes many new features and enhancements, improving the way enterprises and service providers conduct business.

Oracle Application Server Wireless can be split into four component groups:

- **Multi-Channel Server:** Detects devices and transforms content and applications to the device.
- **Sensor Edge Server:** Integrates sensors and other types of command and response indication equipment with applications.
- **Foundation Services:** Services application developers can use to enhance applications and speed development. These are in the form of Java APIs or Web services.
- **Development Tools:** Enable developers to code, test, and debug wireless and voice applications.
- **Mobile Portal:** End-user wireless portal to access the developed applications and content.

Multi-Channel Server

At the core of Oracle Application Server Wireless is the Multi-Channel Server, which enables application access through multiple delivery methods, such as SMS, voice access, WAP, and Pocket PCs. The Multi-Channel Server greatly simplifies and reduces the cost of development by acting as an intelligent wireless proxy for mobile applications. The magnitude of mobile devices and networks are relieved from developers' concerns. Developers can now focus on creating mobile applications for any channel in one, future-proof open standards language. The new Multi-Channel Server extends the existing multi-channel capabilities of previous Oracle Application Server Wireless releases.

Applications written in XHTML are passed through the Multi-Channel Server and translated for any device and network. For example, an XHTML application passed through the Multi-Channel Server is translated to VoiceXML if a phone is accessing the application, and is translated to WML if a WAP phone is accessing the application. The stylesheets used to transcode are maintained and regularly updated by Oracle.

Also new to the Multi-Channel server are Multimedia Adaptation Services. Oracle Application Server Wireless Multimedia Adaptation Services provide device-specific adaptation of images, ringtones, voice grammars, and audio/video streams. Devices support different image formats and have different screen sizes and color depths. As part of the content adaptation performed by OracleAS Wireless in responding to a request, images are dynamically adapted to suit the device. Ringtone adaptation allows for conversion of ringtone data to formats supported by the most popular phones, such as RTTTL, iMelody, and MIDI. The flexible framework for ringtone adaptation allows developers to easily add support for new ringtone formats.

Oracle Sensor Edge Server

Oracle Sensor Edge Server is a middle tier component that integrates sensors and other types of command and response indication equipment and applications. *Sensors* are hardware or software end points that make observations of certain changes of

state. Usually this is a physical state change, for example when a laser diode detects that something has blocked its beam. Radio Frequency Identification (RFID) is this type of sensor. Observations can also be made of software states of change or defect, such as when a monitor daemon that is running on an edge controller exits.

J2ME Support

Java 2 Micro Edition (J2ME) provides a lightweight operating system for mobile devices, enabling client-side development using open standards. With the large number of J2ME-enabled phones on the market, vendors need a method for efficiently building, managing, and delivering J2ME applications to the right mobile users. Oracle Application Server Wireless includes complete, end-to-end support for building J2ME applications and delivering them to mobile devices. OracleAS Wireless J2ME support includes the J2ME Developer's Kit and the J2ME Provisioning System.

There is a restriction on the complexity of J2ME applications because of the limited computing power of mobile devices. The more complicated the J2ME application is, the less usable the application will be on a mobile device. One way to create compelling J2ME applications is to use Web services. Applications are able to push some of the CPU-intensive logic to the Web services residing on the server side. However, even the call to Web services from J2ME devices is too CPU intensive. Oracle J2ME Developer's Kit offers the ability to extend Web services to J2ME devices in an optimized manner for mobile devices. Using the J2ME Developer's Kit, J2ME application (MIDlet) developers can make Web services calls through the Oracle Application Server J2ME proxy server using a client stub. Additionally, MIDlet developers can utilize built-in features optimizing communication, such as request and response caching, if the network is unavailable. The calls can automatically resume when the network connectivity resumes.

OracleAS Wireless streamlines the deployment, management, and delivery of J2ME applications with a provisioning system. The application management Web-based tool allows users to upload J2ME applications for management and secure storage. A byte-code inspector verifies the application for any malicious content. The OracleAS Wireless support for over-the-air (OTA) efficiently delivers applications to target users or devices. Digital rights management adds a digital layer around J2ME applications to support business logic that provides full control over the application. The digital wrapper supports billing strategies and application lifespan control.

Notifications and Multimedia Messaging

Oracle Application Server Wireless further enhances intelligent messaging with functionality for actionable alerts, message adaptation, and failover delivery control. Also new are multi-media messaging (MMS) features that allow for richer messaging experiences. Existing messaging capabilities have been improved to include more flexible message templates, security to prevent message spoofing, support for message prioritization, and more flexibility in handling volume alerts.

OracleAS Wireless supports multi-media messaging (MMS) for rich mobile messages, including graphics, videos, and audio. MMS messages can be authored natively in SMIL or in open standards XHTML. Messages that are authored in XHTML are automatically adapted for wireless devices by OracleAS Wireless. The power of adaptation allows a message to be written once and automatically optimized for any target device.

Notifications are improved by allowing messages to be sent, and responded to, using the new actionable alerting capabilities, enabling further action from a sent alert. For example, a stock alert can prompt a user to take an action and sell when a target price

is hit. Location can now also play a role in alerting. Location-based alerts generate and deliver alert messages based on a mobile user's current location. For example, a field service coordinator receives an alert when a service engineer is within two miles of a customer with an urgent service request.

Asynchronous applications enable messaging devices, such as SMS and email devices, to access applications. A mobile user can maintain a session with an application via SMS or email. To invoke an application, the user sends a message with the message body containing the name of the application and any inputs. A separate message is sent back, by the application, with the results.

Wireless Development Kit

The Oracle Application Server Wireless Development Kit is a small footprint Oracle Application Server Wireless development environment for developing wireless and voice applications. This speeds the development process by giving extra flexibility to fit any development process using any IDE, development tool, Web service, and device simulator. The Wireless Development Kit can be used on any PC or laptop, connected or disconnected, to build and test wireless and voice applications. It is no longer necessary to have a full installation of Oracle Application Server to build and test wireless applications. The Wireless Development Kit supports development for voice, mobile browser, J2ME, and messaging applications.

Oracle offers a version of the Wireless Development Kit specifically for JDeveloper called the JDeveloper Wireless Extension. JDeveloper users can utilize the JDeveloper Wireless Extension for complete wireless development with code templates, wizards, code insight, and automatic deployment to Oracle Application Server.

Web Clipping

The Wireless Web Clipping Server allows clipping and scraping of existing Web content to create wireless applications that reuse your existing PC browser-based applications. The Wireless Web Clipping Server is used to create many applications, each of which represents Web content that has been clipped and scraped from one or more Web sites scattered throughout a large organization.

To create a Wireless Web Clipping application, the user simply uses a Web browser to navigate to the Web page containing the desired content, then selects the portion of the page to clip and scrape. The user then sets some attributes, exposes input parameters if the Web clipping uses form-based submission, saves the application, and tests the application. The following are some of the features that the Wireless Web Clipping Server supports:

- Navigation through various styles of login mechanisms, including form-based and JavaScript-based submission and HTTP Basic and Digest Authentication with cookie-based session management.
- Fuzzy matching of clippings. If a Web clipping gets recorded within the source page or if its character font, size, or style changes, it will still be identified correctly by the Wireless Web Clipping Server and be delivered as the Wireless Web Clipping application content.
- Reuse of a wide range of Web content, including basic support of pages written in HTML 4.0.1, JavaScript, applets, and plug-in enabled content, retrieved through HTTP `GET` and `POST` functions (form submission).

All Wireless Web Clipping application definitions are stored persistently in the Oracle Application Server infrastructure database. Any secure information, such as

passwords, is stored in encrypted form, according to the Data Encryption Standard (DES), using Oracle encryption technology.

Location Services

Oracle Application Server Wireless Location Services give access to the full Location Based Service (LBS) functionality, such as positioning, geocoding, mapping, driving directions, and business directory lookup in an open standards manner. Any application or generic client can use the included WSDLs to invoke the LBS Web services. In addition, OracleAS Wireless instances can use LBS features more conveniently by using the "Web service" provider proxy. This allows you to switch LBS providers without having to make modifications to the applications using LBS features.

The LBS features are available through the Web-based tools in addition to being available through APIs. The LBS features allow mobile positioning, to provide the user's current location, and privacy management, to control when and to whom a mobile user's location is available. Both mobile positioning and the caching of the location information can be enabled or disabled by the system or by individual users. Users can grant mobile positioning access to other users or groups of users (communities) for a certain date range and for specified time windows.

This release also allows a mobile user / device to send the current location, which is usually provided by a GPS receiver, to OracleAS Wireless. The current location can be subsequently queried through the existing mobile positioning and privacy management framework. Users can also choose to position themselves manually using the location mark feature. A location mark can be either a point location specified by an address or a region specified by a city, state, or even a country.

In the previous release, users could configure multiple content providers for geocoding, mapping, driving directions, and business directory services. A provider was selected based on static ordering or its availability region. This release adds the ability to monitor the performance and reliability of providers and dynamically adjust the selection criteria. It also logs performance statistics that help administrators in managing their systems.

Mobile Office Applications

Enterprises can get up and running on wireless or voice applications quickly by deploying the mobile applications that are shipped with Oracle Application Server Wireless. The Mobile Office suite of applications provides wireless browser and voice access to email, calendars, address books, tasks, directories, and files. These applications are fully integrated, enabling the most convenient user experience through such features as directory-based or address book-based recipient selection while composing email messages.

All Mobile Office applications are based on standard protocols, allowing a simple integration into existing environments. Mobile Email gives access from any mobile device to any IMAP or POP3 server, including Microsoft Exchange and Lotus Domino. Mobile Directory connects to any LDAP directory server. And finally, Mobile Calendar integrates with Microsoft Exchange and Lotus Domino servers, and through published interfaces Oracle enables easy customization to support any calendar server.

5

Oracle Business Intelligence

Oracle Business Intelligence Discoverer is an integrated, intuitive, and interactive business intelligence solution that provides comprehensive report creation and delivery capabilities, from data preparation to final presentation, against multidimensional OLAP or relational data sources.

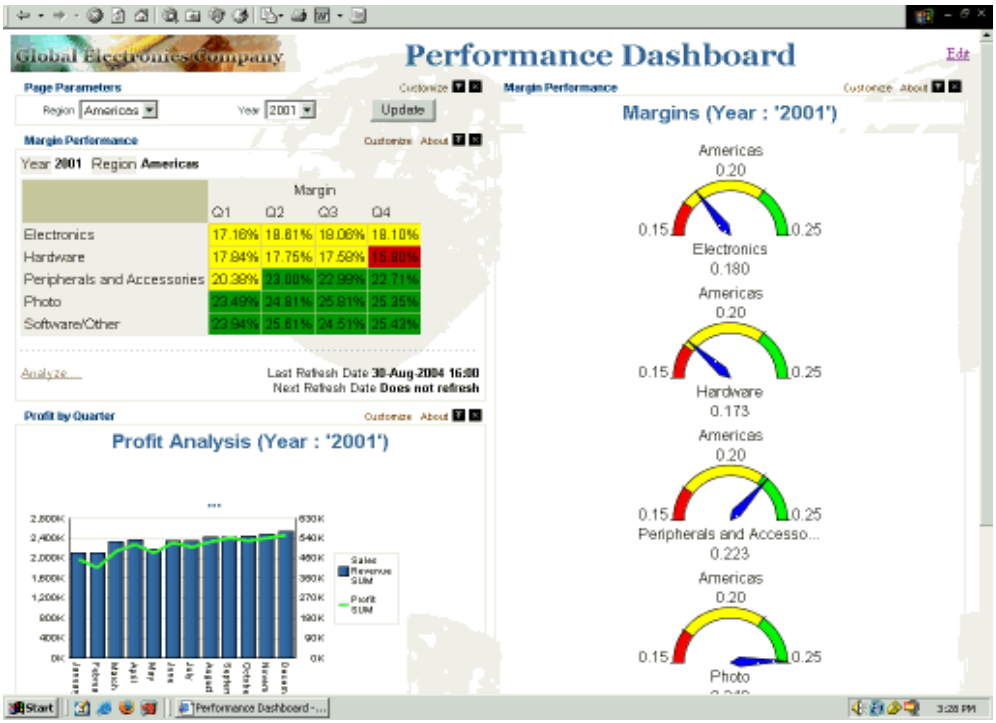
This chapter provides an overview of Oracle Business Intelligence Discoverer features and benefits. The topics include:

Introduction to Oracle Business Intelligence Discoverer

OracleBI Discoverer is a business intelligence tool for analyzing data. It provides an integrated business intelligence solution that includes intuitive ad-hoc query, reporting, analysis, and Web-publishing functionality. OracleBI Discoverer allows non-technical users to gain immediate access to information from multidimensional OLAP data sources, data marts, data warehouses, or online transaction processing systems.

Figure 5–1 shows an example of a business intelligence dashboard, using OracleBI Discoverer reports through Oracle Application Server Portal.

Figure 5–1 Oracle Business Intelligence Discoverer Dashboard Example



Using the various OracleBI Discoverer components, you can do the following:

- Create new worksheets and analyze data from both relational and multidimensional data sources across the Web
- Analyze data in existing worksheets and save personalized customizations of those worksheets
- Display Discoverer worksheets as gauges, tables, crosstabs, or graphs in a dashboard-style portal

With OracleBI Discoverer, business users at all levels of the organization have secure and immediate access to data from a variety of data sources, all through a standard Web browser.

Oracle Business Intelligence Discoverer Components

Oracle Business Intelligence Discoverer includes the following components:

- Oracle Business Intelligence Discoverer Plus OLAP
- Oracle Business Intelligence Discoverer Plus Relational
- Oracle Business Intelligence Discoverer Viewer
- Oracle Business Intelligence Discoverer Portlet Provider

Oracle Business Intelligence Discoverer Plus OLAP

Oracle Business Intelligence Discoverer Plus OLAP allows non-technical users to create reports that leverage the powerful OLAP analysis capabilities of the Oracle database.

The OLAP query model works with users' own common business terms and definitions. This allows users to build complex queries in a series of steps. The flexibility of the OLAP model allows users to add new analytic calculations to their reporting environment that look, feel, and operate like stored measures and execute directly in the database.

After creating a report, Oracle Business Intelligence Discoverer Plus OLAP lets you save your report definitions and calculations independent of the report, saving you time in creating future reports.

Oracle Business Intelligence Discoverer Plus Relational

Oracle Business Intelligence Discoverer Plus Relational is a Web-based report authoring tool designed for non-technical users. It provides efficient, interactive report layout and formatting capabilities, as well as data analysis tools and built-in calculation wizards.

Discoverer Plus Relational works against any relational data source, from data warehouses and datamarts to online transaction processing systems.

After creating reports, you can share them directly or export them as Excel, PDF, or HTML files.

Oracle Business Intelligence Discoverer Viewer

Oracle Business Intelligence Discoverer Viewer allows business users to access reports and analyze their data from a standard Web browser using a pure HTML interface, without needing to install or download additional software. With Discoverer Viewer you can open reports created with Discoverer Plus OLAP or Discoverer Plus Relational.

After analyzing the data, you can save your changes for future viewing, or you can export the report to a variety of file formats, such as Microsoft Excel, HTML, or PDF. You can also send exported files as e-mail attachments directly from Discoverer Viewer.

Oracle Business Intelligence Discoverer Portlet Provider

Integration between Oracle Business Intelligence Discoverer and **Oracle Application Server Portal** allows you to create secure and convenient dashboards to track performance measures critical to your business.

The OracleBI Discoverer Portlet Provider lets you publish existing Discoverer reports to an OracleAS Portal page through a wizard-based interface, without having to write any code. Report data can be presented as a gauge, graph, table, or crosstab, depending on the report.

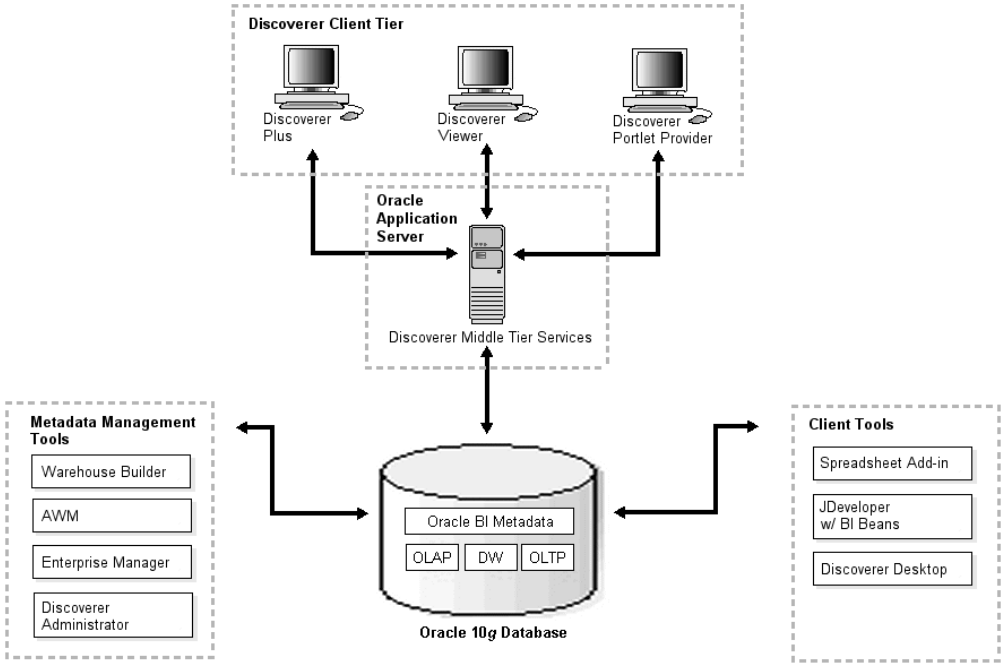
You can then allow other users to view and access their data through these reports, taking advantage of Oracle Identity Management in Oracle Application Server and the fine-grained Access Control in the Oracle Database. Users can click the **Analyze** link to access the reports in Discoverer Viewer. The Discoverer Viewer interactive analysis environment lets users make personalized customizations to the report, tailoring information to their own needs.

See Also: Chapter 3, "Portal Applications"

Oracle Business Intelligence Discoverer Architecture

Figure 5–2 shows the relationship among the components of Oracle Business Intelligence.

Figure 5–2 Oracle Business Intelligence Discoverer Architecture



Oracle Business Intelligence Discoverer has a multi-tier architecture, which takes advantage of the distributed nature of the Web environment. OracleBI Discoverer components are installed across the following tiers:

- **The Discoverer client tier** is the Web browser accessing Discoverer Plus (OLAP or Relational), Discoverer Viewer, or Discoverer Portlets included on an OracleAS Portal page.

  For Discoverer Plus, the only requirement for the client machine is that it runs a Java-enabled Web browser. The first time a machine is used to connect to Discoverer, the Discoverer Plus applet is downloaded from the Discoverer services tier and cached on the client machine. The applet provides the user interface and functionality for creating workbooks and analyzing data.

  For Discoverer Viewer and Discoverer portlets on a portal page, the only requirement for the client machine is that it runs a JavaScript-enabled Web browser.

- **The Discoverer services tier** (also called the **Discoverer middle tier**) consists of Discoverer J2EE and CORBA components. The Discoverer middle tier manager controls these components using Oracle Enterprise Manager 10*g* Application Server Control. The services tier also stores the Discoverer Plus applet.

- **The Discoverer database tier** contains both data and metadata that includes:

  – The Discoverer workbooks used to store reports and charts

  - The Discoverer End User Layer (EUL) that provides an easy-to-understand view of the relational data sources
  - The Discoverer Catalog for access to multidimensional data sources
  - The data users want to analyze, such as an OLAP analytical workspace, data warehouse, or online transaction processing system

  The database tier may include more than one database.

  Note: Metadata management tools and Client tools are not part of Oracle Application Server. They are available with Oracle Developer Suite or Oracle Database.

  See Also: *Oracle Business Intelligence Installation Guide* for information on preparing your data or downloading and installing the Oracle Business Intelligence samples

  See Also: *Oracle Business Intelligence Discoverer Plus User's Guide* and *Oracle Business Intelligence Discoverer Configuration Guide*

- **Metadata management tools:** Before using Oracle Business Intelligence Discoverer, you need either an OLAP data source or a Discoverer End User Layer (EUL) in your database. If you have the connection information for a properly prepared OLAP data source or Discoverer EUL, then you can launch Discoverer Plus to create and edit workbooks and launch Discoverer Viewer to view and customize workbooks.

  Depending on your datasource, you may also need one or several of the following in order to create and maintain your Oracle Business Intelligence metadata:

  - Oracle Warehouse Builder
  - Analytic Workspace Manager
  - Oracle Enterprise Manager
  - Discoverer Administrator
  - **Client tools:** In addition to the Discoverer client tier, users may directly access their Oracle Business Intelligence data from Microsoft Excel using the Spreadsheet Add-in. Developers can use Oracle BI Beans with Oracle JDeveloper to create highly interactive custom applications.

Integrating Oracle Business Intelligence

Tight integration between Oracle Application Server, Oracle Developer Suite, and the Oracle Database delivers performance and scalability. Oracle Business Intelligence Discoverer provides the technology that makes data visible across all parts of the business.

The following sections discuss OracleBI Discoverer in relation to other Oracle Application Server components.

- Leveraging Single Sign-on Functionality
- Using Oracle Enterprise Manager for Management
- Improving Performance with Oracle Application Server Web Cache

Leveraging Single Sign-on Functionality

A single sign-on service provides a single authentication mechanism that allows users to identify themselves securely to multiple applications through a single authentication step. Web-based e-businesses can use **single sign-on** functionality for deployment of business intelligence applications to employees, customers, and partners.

Oracle Application Server Single Sign-On enables users to log in to multiple Web-based applications, such as expense reports, e-mail, and benefits information, using a single user name and password. OracleAS Single Sign-On can serve as the security gateway for all OracleBI Discoverer features.

With OracleAS Single Sign-On, each user maintains only one identity and password for all applications they access.

OracleBI Discoverer leverages OracleAS Single Sign-On functionality to provide a secure point of access to all of your Discoverer users.

See Also: "Oracle Application Server Single Sign-On" on page 11-4

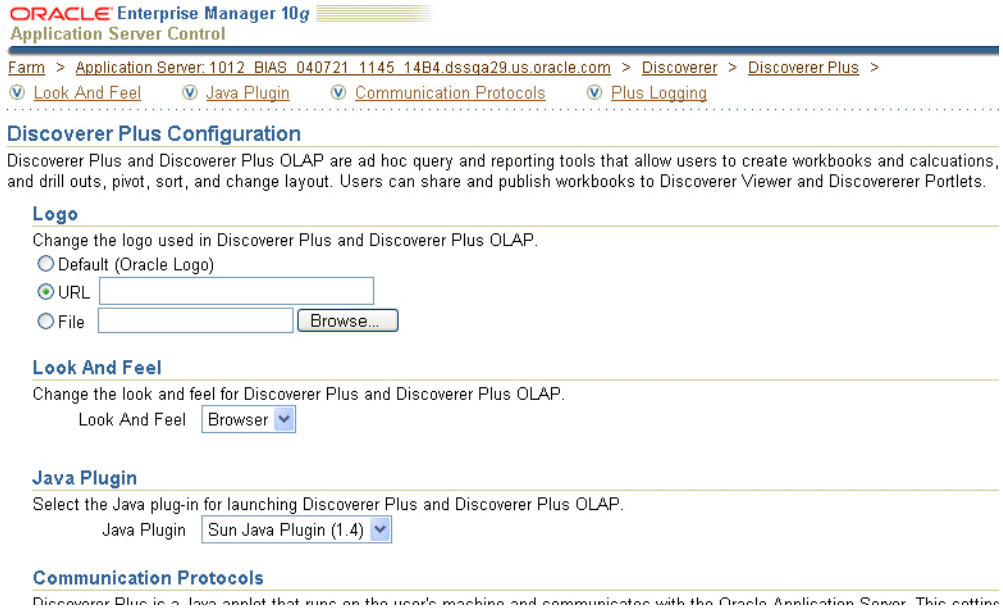
Using Oracle Enterprise Manager for Management

Oracle Application Server Discoverer integration with Oracle Enterprise Manager provides a centralized configuration management tool, enabling system administrators to view and configure Oracle Application Server services in the middle tier. The Application Server Control Console provides easy-to-use graphical interfaces for managing Oracle Business Intelligence services. From Application Server Control Console you can:

- Monitor CPU and memory consumption for OracleBI Discoverer Plus, OracleBI Discoverer Viewer, and OracleBI Discoverer Portlet Provider
- Customize the look and feel for both OracleBI Discoverer Plus and OracleBI Discoverer Viewer
- Install and manage the Discoverer catalog
- Customize the functionality available to users in OracleBI Discoverer Viewer
- Administer OracleBI Discoverer services
- View and search log files
- Switch user-defined connections on or off
- Set default locale for connections
- Set connections to Oracle Application Server Single Sign-On
- Link directly to the Oracle Enterprise Manager Central Console

Figure 5–3 shows some of the OracleBI Discoverer options you can customize through Enterprise Manager.

Figure 5–3 Configuring Discoverer in Enterprise Manager



From the Application Server Control Console you can view alerts and analyze historical data for OracleBI Discoverer services. In addition you can use the Application Server Control Console to manage other related components such as the OC4J and the database supporting these services.

Improving Performance with Oracle Application Server Web Cache

To boost performance over the Internet or extranet, OracleBI Discoverer leverages Oracle Application Server Web Cache. Oracle Business Intelligence Discoverer uses OracleAS Web Cache to speed up the response time for the most common requests. Additionally, you can use OracleAS Web Cache as a router to balance the load across all available application servers.

> **See Also:** *Oracle Application Server Web Cache Administrator's Guide* and *Oracle Business Intelligence Discoverer Configuration Guide*

6

Oracle Application Server Integration

This chapter provides an overview of the two Oracle Application Server Integration components: Oracle Application Server Integration InterConnect and Oracle Application Server Integration B2B.

This chapter contains these topics:

> **See Also:** *Oracle Application Server Integration B2B User's Guide* and *Oracle Application Server Integration InterConnect User's Guide* for detailed information on Oracle Application Server Integration

Introduction to Oracle Application Server Integration InterConnect

Oracle Application Server Integration InterConnect integrates Oracle applications with third-party applications or third-party messaging middleware. This integration can be deployed within an enterprise or across enterprise boundaries through the Internet. OracleAS Integration InterConnect is designed as a hub and spoke system, where OracleAS Integration InterConnect Adapters function as the spokes that access other applications and systems, and the InterConnect Repository Server serves as the hub. The InterConnect Repository Server is a standalone Java application, and the repository is a database that stores the design-time metadata definition.

Oracle Application Server Integration InterConnect Architecture

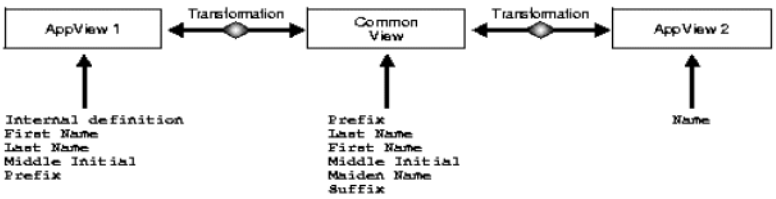
OracleAS Integration InterConnect uses adapters to perform transformations between your application and the application with which it is communicating. OracleAS Integration InterConnect supports four messaging paradigms:

- Implemented procedures (request-response service)
- Publish (event service)
- Subscribe (one-way request service)

- Invoked procedure (InterConnect is the server and the EIS is the client making the request through the adapter)

Figure 6–1 illustrates the relationship between your application and partner applications through adapter transformations.

Figure 6–1 Oracle Application Server Integration InterConnect Architecture



See Also: *Oracle Application Server Adapter Concepts*

Adapter Types

There are four primary adapter types available with Oracle Application Server:

- **Technology adapters:** OracleAS Integration technology adapters are available out of the box with every Oracle Application Server installation. They provide connectivity with data stores, messaging middleware, and various transport protocols. These adapters include OracleAS Integration Adapter for HTTP, OracleAS Integration Adapter for FTP, OracleAS Integration Adapter for Databases, and OracleAS Integration Adapter for Files.
- **B2B adapters:** OracleAS Integration B2B adapters integrate with business applications used between trading partners, over the Internet in real time. These adapters include Oracle Adapters for Rosettanet, EDI, and ebXML.
- **Application adapters:** OracleAS Integration application adapters integrate with packaged applications such as SAP, Siebel, PeopleSoft, and J.D. Edwards.
- **Legacy adapters:** OracleAS Integration legacy adapters integrate with legacy and mainframe systems such as CICS, IMS/TM, and Tuxedo.

Oracle Application Server Integration InterConnect Features

When enterprises implement an enterprise information system (EIS) by integrating diverse applications that run on different platforms, they rely on a set of adapters to provide connectivity between these applications.

The various applications that enterprises use in their daily operations may include database, mainframe, and Web interface applications, all of which run on different operating systems and are developed by different vendors. Oracle provides a suite of integration adapters that implement bi-directional connectivity between applications and various back-end systems to enable fast, flexible, and efficient integration.

Key features of Oracle Application Server Integration Adapters include:

- OracleAS Integration Adapters are written in Java and integrate with several back-end APIs.
- OracleAS Integration Adapters perform translation between Oracle data, usually in XML format, and native back-end data formats.

- GUI-based design-time tools available with OracleAS Integration Adapters provide a standard methodology for downloading, configuring, and storing the schemas that represent the EIS business objects and operations. These tools facilitate rapid change without additional coding effort.
- Through its use of standards like Web services, JCA, and J2EE, OracleAS Integration Adapters provide high availability and scalability to the enterprise. OracleAS Integration Adapters can be deployed as JCA 1.0 adapters or as Web services servlets within the OC4J container.
- OracleAS Integration Adapters can be deployed in Oracle Application Server clusters.
- By supporting bi-directional connectivity, the OracleAS Integration Adapters enable request-response services to back-end applications, and can act as a client in event notifications.

Using OracleAS Integration Adapters to connect Oracle Application Server and other systems and applications ensures that the distributed system has the advantage of inheriting many other Oracle Application Server features. These features include system security, scalability, persistence, transaction support, portability, and NLS support.

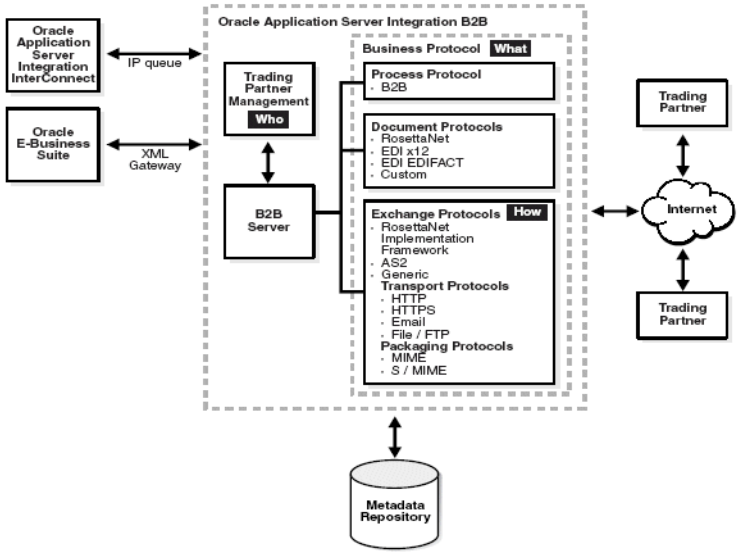
Introduction to Oracle Application Server Integration B2B

Oracle Application Server Integration B2B (OracleAS Integration B2B) is an e-business integration product for the business-to-business (B2B) exchange of services, information, and products. If you know who you want to trade with (for example, a specific supplier), what you want to do (for example, send a purchase order), and how you want to do it (for example, over the Internet using a secure channel), then you have defined a basic B2B transaction.

Oracle Application Server Integration B2B Architecture

Figure 6–2 illustrates the OracleAS Integration B2B architecture. Who you want to trade with is defined in the trading partner management component. What you do is defined in the business protocol (shown within the dotted lines). How you do the B2B transaction is defined in the transport and packaging components. OracleAS Integration B2B provides extensive connectivity to external trading partners through standard B2B protocols. Within the enterprise, OracleAS Integration B2B can interface with OracleAS Integration InterConnect, Oracle BPEL Process Manager, the E-Business Suite XML Gateway, and third party software applications through Oracle AQ and JMS. This figure also shows that the B2B core supports a number of standard protocols.

Figure 6–2 Oracle Application Server Integration B2B Architecture



Oracle Application Server Integration B2B Features

OracleAS Integration B2B provides the features you need to design, deploy, monitor, and manage your integrations. These features include the following:

- **Comprehensive trading partner management:** As you identify trading partners, you are also defining the basic B2B transaction. This task flow means that you associate business protocols and the communication details as you set up each partner.
- **Standard B2B protocols:** You can work with trading partners who use various protocols, such as RosettaNet, EDI EDIFACT, EDI X12, AS2, or you can create custom document protocols.
- **Predefined business protocols:** By defining combinations of a document protocol, an exchange protocol, and a process protocol, OracleAS Integration B2B provides protocols based on your purpose. Predefined business protocols also make it easy to set up multiple protocols for the same trading partner.
- **Integrated user experience:** The user interface tool and wizards guide all aspects of your B2B transaction, no matter which business protocol you choose.
- **Reports:** With the integrated reports feature, you can create business messages, wire messages, collaboration, and error status reports. You can also save report definitions for reuse.
- **Security:** The security features of OracleAS Integration B2B include automatic encryption of the host trading partner's parameters using an obfuscated encryption key created during installation. OracleAS Integration B2B also leverages Oracle Application Server security features.
- **Integration with other Oracle integration products:** OracleAS Integration B2B can interface with OracleAS Integration InterConnect, Oracle BPEL Process Manager, the Oracle E-Business Suite XML Gateway, and Oracle AQ and JMS.

Oracle BPEL Process Manager

Business Process Execution Language (BPEL) provides enterprises with an industry standard for business process orchestration and execution. BPEL offers a standard language for defining how to send XML messages to remote services, manipulate XML data structures, receive XML messages asynchronously from remote services, manage events and descriptions, define parallel sequences of execution, and undo parts of processes when exceptions occur.

Oracle BPEL Process Manager enables organizations to model and deploy business processes based on the BPEL standard. Key features of Oracle BPEL Process Manager include:

- The BPEL Designer provides a graphical interface for building BPEL processes.
- The core BPEL engine provides a scalable and robust BPEL server that executes standard BPEL processes. It also uses a database to maintain the state of long-running processes, enabling clustering for failover and scalability.
- The built-in integration services enable developers to leverage advanced connectivity and transformation capabilities from standard BPEL processes. These capabilities include support for XSLT and XQuery transformations, as well as bindings to legacy systems through JCA adapters and native protocols.
- The extensive WSDL binding framework enables connectivity to protocols and message formats such as JMS, email, JCA, HTTP GET and POST, and many others.
- The BPEL Console provides a Web-based interface for management, administration, and debugging of processes deployed to the BPEL server. Audit trails and process history / reporting information are automatically maintained and available both through the BPEL Console and through a Java API.

7

Oracle Application Server Infrastructure

This chapter provides an overview of the Oracle Application Server Infrastructure. The topics include:

Introduction to Oracle Application Server Infrastructure

Oracle Application Server provides an industry standards-based application deployment platform and security and management facilities to simplify all aspects of application deployment. Oracle Application Server Infrastructure is a comprehensive deployment platform designed to streamline application deployment by leveraging a single security, directory, and product metadata framework for all applications.

Oracle Application Server Infrastructure includes the Oracle Identity Management infrastructure, which provides security life cycle management for network entities such as users, devices, processes, and applications. It provides a comprehensive, integrated security framework to support all Oracle Application Server components, as well as third party and custom applications deployed on Oracle Application Server. The framework is based on Oracle Application Server Single Sign-On for authentication, Oracle Internet Directory for authorization, user provisioning, password policy and delegated administration services, and directory integration and synchronization, and Oracle Application Server Certificate Authority to manage X.509v3 certificates supporting PKI-based (strong) authentication.

By providing an integrated infrastructure, Oracle Application Server reduces the time required to develop Internet applications, makes these applications more reliable when deployed, and lowers the total deployment cost.

What is Oracle Application Server Infrastructure?

Oracle Application Server Infrastructure is an installation type that provides centralized product metadata and security services, configuration information, and data repositories for middle tier installations. The middle tier instances typically use the Infrastructure for three main services:

- **Product Metadata Service:** All of the product metadata required by the Oracle Application Server middle tier instances is bundled as part of the infrastructure. Product metadata is not accessed directly by customer applications. The Product Metadata Service is provided by Oracle Application Server Metadata Repository, which middle tier instances can use as a centralized component repository and leverage for product metadata lookups.

- **Identity Management Services:** Identity Management Services provide a consistent security and identity management model for all Oracle Application Server applications. It also provides a single source of security metadata containing all administration and user privileges. Middle tier components use the Identity Management Services to increase security, centralize the authentication services, and manage passwords. The Identity Management Services are provided by the Oracle Identity Management infrastructure and its components.
- **Management Service:** The Management Service in the Oracle Application Server Infrastructure is used to support the Distributed Configuration Management (DCM) tool. DCM stores information in the metadata repository.

Oracle Application Server Infrastructure Components

Oracle Application Server Infrastructure contains the following components:

- Oracle Application Server Metadata Repository
- Oracle Identity Management
  - Oracle Application Server Single Sign-On
  - Oracle Internet Directory and its components
  - Oracle Application Server Certificate Authority

In addition to these components, dedicated Oracle HTTP Server and Oracle Application Server Containers for J2EE (OC4J) instances are also installed with Oracle Application Server Infrastructure. These instances are used by the various components of the Infrastructure to service requests.

Oracle Application Server Metadata Repository

Oracle Application Server Metadata Repository is an information store that enables both infrastructure and middle tier instances to manage and configure their components in an optimal way. Oracle Application Server Metadata Repository can be installed into either a new or an existing database. When you install the Metadata Repository into a new database, infrastructure installs an Oracle Enterprise Edition database server that contains the demo data, schemas, and metadata required by most of the Oracle Application Server middle tier instances. You can also choose to install the Metadata Repository and its associated data, schemas, and metadata into an existing database, using the Oracle Application Server Metadata Repository Creation Assistant tool.

There are two general types of data that can be stored in a database: customer or application data, and metadata. Customer or application data is user data created by a client application. It is accessed directly by the client application.

Metadata, by comparison, includes component-specific information that is accessed by the Oracle Application Server middle tier or Infrastructure components as part of their application deployment. The end user or the client application does not access this data directly. For example, a Portal application on the middle tier accesses the Portal metadata as part of the Portal page assembly aggregation. Metadata also includes demo data for many Oracle Application Server components.

The Oracle Application Server Metadata Repository stores three main types of metadata:

- Management metadata
- Identity Management metadata

- Product metadata

Table 7–1 shows the Oracle Application Server components that store and use these types of metadata during application deployment.

Table 7–1 Metadata and Infrastructure Components

Type of Metadata	Infrastructure Components Involved
Management metadata	Distributed Configuration Management (DCM)
Identity Management metadata	Oracle Application Server Single Sign-On, Oracle Internet Directory, Oracle Application Server Certificate Authority
Product metadata (includes demo data)	Oracle Application Server Metadata Repository

Using Oracle Application Server Infrastructure with Middle Tier Installations

Oracle Application Server provides three middle tier install options. Oracle Application Server Metadata Repository is required for all installation types except for J2EE and Web Cache.

- **J2EE and Web Cache:** Installs Oracle HTTP Server, Oracle Application Server Containers for J2EE, Web Cache, Web Services, UDDI, and Oracle Enterprise Manager Application Server Control. Installing Oracle Application Server Metadata Repository is optional, but recommended. Installing Oracle Application Server Infrastructure allows you to create database-managed OC4J clusters, and enables you to use Single Sign-On and other identity management protections available through the Infrastructure.
- **Portal and Wireless:** Installs all components of J2EE and Web Cache, plus Portal, Ultra Search, and Wireless. Installing Oracle Application Server Metadata Repository is required.

Oracle Application Server Integration components, such as Oracle Application Server Integration B2B, Oracle Application Server Integration InterConnect, and Oracle Workflow are installed on top of any of these middle tier installation options.

Oracle Business Intelligence Discoverer and Oracle Content Management Software Developer Kit (CM SDK) are also installed separately in addition to one of the standard middle tier installations.

The Distributed Configuration Management (DCM) component enables you to manage middle tiers and the Identity Management Infrastructure, and stores its metadata in the Metadata Repository for the Portal and Wireless install options. For the J2EE and Web Cache install type, by default DCM uses a file-based repository. If you choose to associate the J2EE and Web Cache install type with an Infrastructure, the file-based repository is moved into the Metadata Repository, enabling database-managed Oracle Application Server clustering.

Oracle Application Server Metadata Repository Contents

Within Oracle Application Server Metadata Repository, there is metadata for many Oracle Application Server components. Oracle Application Server Metadata Repository contains metadata for the following components:

- Distributed Configuration Management (DCM)
- Oracle Internet Directory
- Oracle Application Server UDDI Registry (for Web Services)
- Oracle Application Server Portal

- Oracle Ultra Search
- Oracle Application Server Single Sign-On
- Oracle Application Server Wireless
- Oracle Business Intelligence Discoverer
- Oracle Application Server Integration
- Oracle Workflow
- Oracle Application Server Web Services
- Online Analytical Processing (OLAP)
- Oracle Application Server Certificate Authority

For information related to the metadata for each component, please see the *Oracle Application Server Administrator's Guide*.

Oracle Identity Management

Identity management is the process of managing the security life cycle for network entities in an organization, and most commonly refers to the management of an organization's application users. Oracle Identity Management is an integrated infrastructure that Oracle products use for centralized security in a complex multi-application or distributed processing environment. The Oracle Identity Management infrastructure includes the following components:

- Oracle Application Server Single Sign-On
- Oracle Internet Directory and its components
- Oracle Application Server Certificate Authority

The following sections contain brief overviews of each Identity Management component. For more information on the Identity Management infrastructure, please see Chapter 11, "Security and Identity Management".

Oracle Application Server Single Sign-On

Oracle Application Server Single Sign-On enables users to access multiple Oracle Application Server applications with a single password. Using Single Sign-On, users can log in to Oracle Application Server and gain access to all applications for which they are authorized, without requiring them to re-enter a user name and password for each application. Oracle Application Server Single Sign-On retrieves user information from Oracle Internet Directory, and LDAP v3 compliant directory.

Oracle Internet Directory

Oracle Internet Directory is the Oracle implementation of Lightweight Directory Access Protocol (LDAP), version 3. Application server instances, components, and infrastructures store security and management information in the directory. Oracle Internet Directory serves the Oracle Application Server environment by providing authentication and a centralized user provisioning model whereby you can create and manage users on an enterprise scale. It provides a single source of access to security administration information such as Oracle Application Server instance objects, Oracle Application Server instance configuration, Oracle Application Server component schema mappings, and application group information by components (such as Portal). When users log in, they are authenticated once by Oracle Application Server Single Sign-On against their OID credentials, and afterwards can access multiple applications seamlessly.

Directory Integration and Provisioning Directory Integration and Provisioning is a component of Oracle Internet Directory. It permits synchronization between Oracle Internet Directory and other directories; user repositories and automatic provisioning services for Oracle components; applications; and third-party applications through standard interfaces. Typically, provisioning an application means creating and managing separate user accounts and their privileges.

Delegated Administration Services Delegated Administration Services is a component of Oracle Internet Directory. It allows users and application administrators to perform trusted proxy-based administration of directory information. You can assign administrative responsibilities according to business requirements, and control security policies for different components of the enterprise.

Oracle Application Server Certificate Authority

The Oracle Application Server Certificate authority manages and publishes X.509v3 certificates to support PKI-based (strong) authentication methods. OracleAS Certificate Authority also serves as an assertion services, since the certificates it generates are assertions about a network's identity and its entitlements.

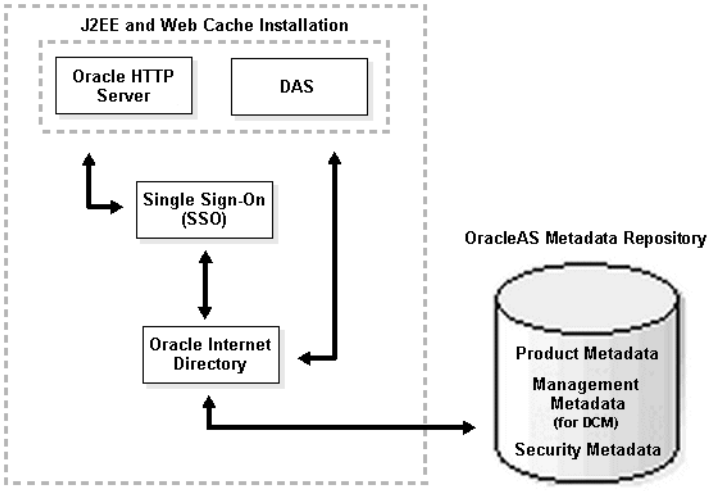
See Also: *Oracle Identity Management Concepts and Deployment Planning Guide*

Oracle Application Server Infrastructure Architecture

Oracle Application Server Infrastructure provides centralized product metadata and security, configuration information, and data repositories for middle tier installations.

Figure 7–1 shows how the Oracle Application Server Infrastructure components work closely together to provide these services to the middle tier Oracle Application Server instances.

Figure 7–1 Oracle Application Server Infrastructure Components



- Oracle Application Server Metadata Repository contains three types of metadata:
  - Product Metadata for middle tier instances
  - Management Metadata for Distributed Configuration Management

    - Security Metadata for Oracle Internet Directory and Oracle Application Server Single Sign-On
- Oracle Application Server Single Sign-On uses Oracle HTTP Server, and the Delegated Administration Service uses the Infrastructure OC4J instance, which are part of the J2EE and Web Cache instance that is embedded in Oracle Application Server Infrastructure.
- Oracle Application Server Single Sign-On stores user information in Oracle Internet Directory.

Oracle Application Server Infrastructure has several deployment architectures, which makes it easy to fit it into an existing enterprise deployment methodology. Some of the most frequently used deployment topologies for both Oracle Application Server Infrastructure itself and for applications which use Oracle Application Server Infrastructure are discussed in Chapter 12, "Recommended Topologies".

Part II

Deployment

Part II discusses topics related to application deployment, and contains the following chapters:

8

Scalability and High Availability

This chapter provides an overview of high availability and scalability solutions provided by Oracle Application Server. The topics include:

- Scalability
- High Availability

Scalability

Scalability is the ability of a system to provide throughput in proportion to, and limited only by, available hardware resources. A scalable system is one that can handle increasing numbers of requests without adversely affecting response time and throughput.

The growth of computational power within one operating environment is called vertical scaling. Horizontal scaling is leveraging multiple systems to work together on a common problem in parallel.

Oracle Application Server scales both vertically and horizontally. Horizontally, Oracle Application Server can increase its throughput with Oracle Application Server Clusters, where several application server instances are grouped together to share a workload. Also, Oracle Application Server provides great vertical scalbility, allowing you to start several virtual machines from the same configuration files inside a single operating enviroment (automatically configuring ports, applications and routing). This provides the advantage of vertical scaling based on multiple processes, but eliminates the overhead of administering several separate application server instances.

High Availability

The **availability** of a system or any component in that system is defined by the percentage of time that it works normally. The formula for determining the availability for a system is:

Availability = average time to failure (ATTF) / [average time to failure (ATTF) + average time to recover (ATTR)]

For example, a system that works normally for twelve hours per day is 50% available. A system that has 99% availability is down 3.65 days per year on average. Critical systems may need to meet exceptionally high availability standards, and experience as little as four to five minutes of downtime per year.

Oracle Application Server is designed to provide a wide variety of high availability solutions, ranging from load balancing and basic clustering to providing maximum system availability during catastrophic hardware and software failures.

High availability solutions can be divided into two basic categories: local high availability and disaster recover.

Local High Availability Solutions

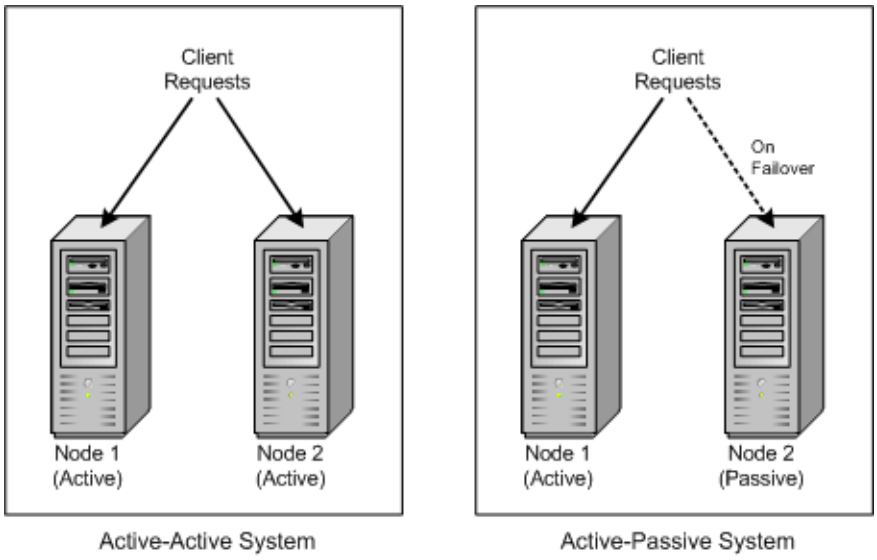
Local high availability solutions ensure availability in a single data center deployment. These solutions guard against process, node, and media failures, as well as human errors. Local high availability solutions can be further divided into two types: active-passive and active-active.

Active-passive solutions deploy an active instance that handles requests and a passive instance that is on standby. When the active instance fails, the active instance is shut down and the passive instance is brought online, and resumes application services. At this point the active-passive roles are switched. This process can be done manually or it can be handled through vendor-specific clusterware. Active-passive solutions are generally referred to as cold failover clusters.

Active-active solutions deploy two or more active system instances at all times. All instances handle requests concurrently.

Figure 8–1 illustrates active-active and active-passive system deployments.

Figure 8–1 Active-active and active-passive high availability solutions



In addition to architectural redundancies, other local high availability features include:

- **Process death detection and automatic restart:** Processes may die unexpectedly due to configuration or software problems. A proper process monitoring and restart system should monitor all system process constantly and restart them if there are problems.
- **Clustering:** Clustering components of a system together allows the components to be viewed functionally as a single entity from the client perspective. A cluster is a set of processes running on a single or multiple computers that share the same workload. A cluster contains one or more runtime instances of an Oracle Application Server that have identical instance configuration but have different application configurations. A cluster provides redundancy for one or more applications.

- **Configuration management:** A clustered group of similar components often need to share common configurations. Proper configuration management enables the components to synchronize their configurations and also provides highly available configuration management for less administrative downtime.
- **State replications and routing:** For stateful client requests, client state can be replicated to enable stateful failover of requests in the event that processes serving these requests fail.
- **Server load balancing and failover:** When multiple instances of identical server components are available, client requests to these components can be load balanced to ensure that the instances have roughly the same workload. With a load balancing mechanism in place, the instances are redundant. If any of the instances fail, requests to the failed instance can be sent to the surviving instances.
- **Connection failure management:** Clients often connect to services on the server and reuse these connections. When a process implementing one of these services on the server is restarted, the connection may need to be re-established. Correct re-connection management ensures that clients have uninterrupted service.

Backup and Recovery Solutions

Backup and recovery refers to the various strategiew and procedures involved in guarding against hardware failures and data loss, and reconstruction data should a loss occur. There are failure scenarios that do not involve the catastrophic loss of an entire production environment. But regardless of the type of failure, once a failure has occurred in your system it is important to restore the failed component or process as quickly as possible.

User errors may cause a system to malfunction. In certain circumstances, a component or system failure may not be repairable. A backup and recovery facility should be available to back up the system at certain intervals and restore a backup when an irreparable failure occurs.

Disaster Recovery Solutions

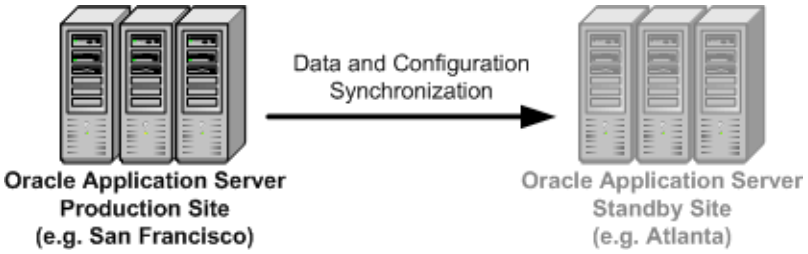
Disaster recovery solutions are usually geographically distributed deployments that protect your applications from disasters such as floods or regional network outages. Disaster recovery solutions typically set up two homogeneous sites, once active and one passive. Each site is a self-contained system. The active site is generally called the production site, and the passive site is called the standby site.

During normal operation, the production site services requests. In the event of a site failover or switchover, the standby site takes over the production role, and all requests are routed to that site.

To maintain the standby site for failover, not only must the standby site contain homogeneous installations and applications, but data and configurations must also be synchronized constantly from the production site to the standby site.

Figure 8–2 illustrates a geographically distributed disaster recovery solution.

Figure 8–2 Geographically distributed disaster recovery



Oracle Application Server High Availability

Oracle Application Server provides local high availability, backup and restore, and disaster recovery solutions for maximum protection against any kind of failure with flexible installation, deployment, and security options.

Oracle Application Server Local High Availability Solutions

Oracle Application Server local high availability is achieved by several active-active and active-passive solutions for the Oracle Application Server middle tier and the Oracle Application Server Infrastructure. With both active-active and active-passive high availability solutions, there are options that differ in ease of installation, cost, scalability, and security.

Oracle Application Server Backup and Recovery Solutions

Some failures require more involved recovery scenarios than simply restarting processes. In some cases, you will have to perform restoration operations based on backup procedures you have previously implemented.

Complete Backup and Restore A complete Oracle Application Server environment backup includes:

- A full backup of all files in the middle-tier Oracle homes (including Oracle software files and configuration files)
- A full backup of all files in the Infrastructure Oracle home (including Oracle software files and configuration files)
- A complete cold backup of the Metadata Repository
- A full backup of the Oracle system files on each host in your environment

Failures that require the complete backup and restore solution for recovery include node failure where the node needs to be completely replaced, and the deletion or corruption of Oracle software or binary files. Failures that require this type of recovery solution also then require the manual restart of all processes. For details about specific failure types and how to recover, see the *Oracle Application Server Administrator's Guide*.

Online Backup and Restore Depending on the type of failure your system is experiencing, you may need to restore your system from an online backup. An online backup includes:

- An incremental backup of the configuration files in the middle tier Oracle homes
- An incremental backup of the configuration files in the Infrastructure Oracle home
- An online backup of the Metadata Repository

Failures that require online backup and restore solutions for recovery include data failure in the metadata repository and deletion or corruption of Oracle Application Server component runtime configuration files. Failures that require this type of solution also then require one or more processes to be restarted. For details about specific failure types and how to recover, see the *Oracle Application Server Administrator's Guide*.

Oracle Application Server Disaster Recovery Solutions

Built on top of the local high availability solutions is the Oracle Application Server disaster recovery solution. This solution requires homogeneous production and standby sites that mirror each other in Oracle Application Server and platform configurations. These configurations must be synchronized regularly to maintain the homogeneity.

Details about each of the high availability solutions you can implement with Oracle Application Server are described in detail in the *Oracle Application Server High Availability Guide*, along with instructions on how to configure, operate, and manage each one.

9

Performance and Caching

This chapter provides an overview of Oracle Application Server performance and **caching** features and benefits. The topics include:

- Introduction to Performance
- Overview of Caching Solutions
- Introduction to Oracle Application Server Web Cache
- Oracle Application Server Web Cache Deployment Architecture
- Oracle Application Server Web Cache Features
- Additional Caching Components

Introduction to Performance

Increasing the performance of your Web site and increasing the speed of your applications without redesigning or rebuilding the Web site are common goals. To maximize Oracle Application Server performance, all components need to be monitored, analyzed, and tuned. Performance must be built in to an application deployment; you must anticipate performance requirements during application analysis and design, and balance the costs and benefits of optimal performance.

The overall performance of an application is determined by these factors:

- How many resources are available?
- How many clients need the resource?
- How long must they wait for the resource?
- How long do they hold the resource?

The following concepts are fundamental to understanding performance:

- **Response time:** The **response time** is equal to the **service time** plus the **wait time** for a task to complete. You can increase response time performance by reducing the service time, the wait time, or both. For example, you can decrease wait time by implementing parallel processing with multiple resources, such that more resources are available to the incoming tasks. Oracle HTTP Server processes requests in this way, allocating client requests to available `httpd` processes.
- **System throughput:** System **throughput** is the amount of work accomplished in a given amount of time. You can increase throughput with a combination of reducing service time and reducing the overall response time by increasing the amount of scarce resources that are available. For example, if the system CPU is bound, then adding CPU resources should improve performance.

- **Wait time:** While the service time for a task may stay the same, the wait time will lengthen with increased **contention**. If many users are waiting for a service that takes one second, the tenth user must wait nine seconds. Reducing contention should improve performance.
- **Critical resources:** Resources such as CPU, memory, I/O capacity, and network bandwidth are key to reducing service time. Adding resources increases throughput and reduces response time.

  As the number of requests rises, the time to service completion increases if the number of resources stays the same. To improve performance, you can either limit the demand rate to maintain acceptable response times, or you can add resources.

Performance Methodology

Achieving optimal effectiveness in your system requires planning, monitoring, and periodic adjustment. The first step in performance tuning is to determine the goals you need to achieve, and then design effective usage of available technology into your applications. After implementing your system, you must periodically monitor and adjust your system.

Performance Targets

Whether you are designing or maintaining a system, you should set specific performance goals so that you know how and what to optimize. If you alter parameters without a specific goal in mind, you can waste time tuning your system without significant gain.

An example of a specific performance goal is an order entry response time under three seconds. If the application does not meet that goal, identify the cause and take corrective action. During development, test the application to determine if it meets the desired performance goals.

User Expectations

Application developers, database administrators, and system administrators must be careful to set appropriate performance expectations for users. When the system carries out particularly complicated operations, response time may be slower than when it is performing a simple operation. Users should be made aware of which operations might take longer.

Performance Evaluation

With clearly defined performance goals, you can readily determine when performance tuning has been successful. Success depends on the functional objectives you have established with the user community, your ability to measure whether or not criteria are being met, and your ability to take corrective action to overcome any performance issues.

Ongoing performance monitoring enables you to maintain a well-tuned system. Keeping a history of the application's performance over time enables you to make useful comparisons. With data about actual resource consumption for a range of loads, you can conduct objective scalability studies and from these predict the resource requirements for anticipated load volumes.

Improving Performance

In order to improve the performance of your applications, you have to consider the various factors that influence performance and make changes to your system as necessary.

Factors in Improving Performance

Performance spans several areas:

- **Sizing and configuration:** Determining the type of hardware needed to support your performance goals
- **Parameter tuning:** Setting configurable parameters to achieve the best performance for your application
- **Performance monitoring:** Determining what hardware resources are being used by your application and what response time your users are experiencing
- **Troubleshooting:** Diagnosing why an application is using excessive hardware resources, or why the response time exceeds the desired limit

Countering the Effects of Excessive Demand

Excessive demand increases response time and reduces throughput. If the demand rate exceeds the achievable throughput, then you must determine through monitoring which resource is exhausted, and if possible increase that resource.

Making Adjustments to Relieve Performance Problems

Performance problems can be relieved by making adjustments to the following:

- **Unit consumption:** Reducing the resource consumption of each request can improve performance. This might be achieved by pooling and caching.
- **Functional demand**: Rescheduling or redistributing the work can improve performance in some cases.
- **Capacity:** Increasing or reallocating resources can improve performance in some cases.

Tuning usually involves a series of trade-offs. After you have determined the bottlenecks, you may have to modify performance in some other areas to achieve the desired results. For example, if I/O is a problem, you may need to purchase more memory or more disks. If a purchase is not possible, you may have to limit the **concurrency** of the system to achieve the desired performance. However, if you have clearly defined goals for performance, the decision on what to trade for higher performance is easier because you have identified the most important areas.

Overview of Caching Solutions

Caching is one of the key technologies that promises to alleviate the computational and economic burdens faced by today's overstrained **e-business** infrastructures. Nearly all applications benefit from having Web content cached on **host**s between the consumers searching for content and the content source itself. When applied to Web applications, caching is essentially a technique for storing partial or complete Web pages, both static and dynamic, in memory closer to the browser to address the problem of slow access to Web sites.

A practical caching solution must do the following tasks:

- serve dynamic content, ensuring freshness

- handle thousands of concurrent users at high sustained rates of throughput
- provide fast response times
- support local and global deployments
- integrate with other caching techniques
- post-process cached content
- provide high gains with low-cost infrastructure

Caching solutions can be employed in different tiers. Each solution targets a specific tier and presents certain capabilities. It is important to note that response time is the cumulation of the times to access different tiers in your architecture. Most often a complete solution is a combination of one or more caching solutions. Caching solutions include browser caching, proxy caching, content delivery network services, and server accelerators.

The following sections describe Oracle Application Server Web Cache, the middle-tier server acceleration and load balancing component of Oracle Application Server.

Introduction to Server Accelerators

A server accelerator is a cache and compression engine that stands in for one or more specific Web servers, rather than working on behalf of a group of browser users. A server accelerator cache, or "reverse proxy" cache, intercepts all requests to the Web servers, caches a copy of the objects served, and then serves those objects when it next receives requests for them. As the server accelerator's cache becomes populated, it is able to serve more of the requested content itself, freeing up processing resources in the application server and the database for other tasks. Server accelerators also help cut costs, as they are implemented on inexpensive platforms and take some of the load off of more expensive back-end content generation systems.

Introduction to Oracle Application Server Web Cache

Oracle Application Server Web Cache is a powerful, state-of-the-art server acceleration and load balancing solution. OracleAS Web Cache offers intelligent caching, page assembly, and compression features that distinguish it from all other Web caching solutions on the market. Unlike legacy **proxy server**s, which cache only static **object**s, OracleAS Web Cache accelerates the delivery of both static and dynamic Web content, improving response time for feature-rich pages.

OracleAS Web Cache also supports Edge Side Includes (ESI) for performing page assembly in at the network edge. OracleAS Web Cache leverages this technology to enable partial-page caching and dynamic page assembly using both cacheable and non-cacheable page fragments. In this way, OracleAS Web Cache optimizes the delivery of rich, personalized content.

Deployed before a farm of application servers or globally at the network edge, OracleAS Web Cache provides load balancing, **failover**, clustering, and surge protection features for application servers.

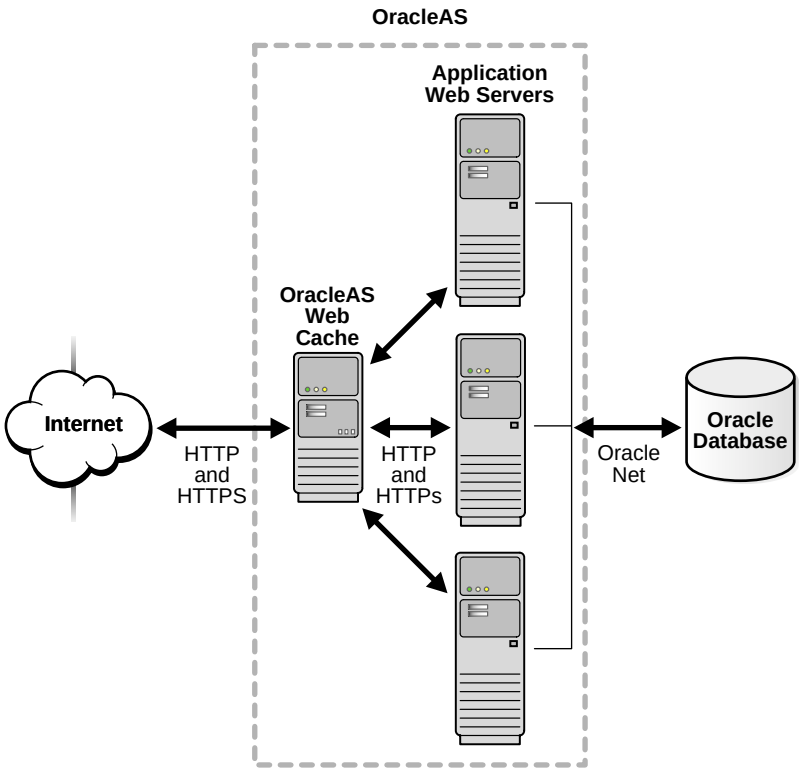
Oracle Application Server Web Cache Deployment Architecture

In the simplest of deployment scenarios, OracleAS Web Cache is positioned in front of one or more Web servers to cache and compress content generated by those servers. OracleAS Web Cache then delivers that content to Web browsers. When Web browsers access the Web site, they send HTTP or HTTPS requests to OracleAS Web Cache,

which acts as a virtual server for the Web site, masking the existence of the application server farm and the database. If the requested content has changed, OracleAS Web Cache retrieves the new content from the application servers according to the relative load on each server.

OracleAS Web Cache can be deployed on the same **host** as the origin application server (co-located) or on a separate **node** of its own (dedicated). Figure 9–1 shows a dedicated OracleAS Web Cache deployment.

Figure 9–1 Caching Architecture



Because OracleAS Web Cache consumes memory, co-location is only viable if the cache and the application servers do not contend for resources.

Dedicated deployment is often preferable to co-located deployment. In a dedicated scenario, there is no risk of resource contention with other server processes. OracleAS Web Cache also performs well on commodity hardware, so a dedicated deployment need not be a costly one in terms of hardware expenditure. For very high-volume Web sites, and to avoid a single point of failure, two or more hosts running OracleAS Web Cache may be deployed behind a third-party network load balancing device.

Cache hierarchies: OracleAS Web Cache offers hierarchical caching features that enable customers to easily create Content Delivery Networks (CDNs). Many Web-based applications mirror their Web sites in strategic geographical locations. Caching provides a low-cost alternative to mirroring, and can also be used to serve local markets to shorten response times to these markets and to reduce bandwidth and rack space costs for the content provider. Additionally, in a distributed cache hierarchy, the central cache is aware of the local caches. As a result, any content invalidation messages sent to the central cache automatically propagate to these remote caches. This invalidation propagation ensures content consistency across the CDN and simplifies the management of cache hierarchies.

Using OracleAS Web Cache in heterogeneous environments: While integrated with Oracle Application Server, OracleAS Web Cache is also compatible with third-party application servers, databases, and content management systems.

See Also: *Oracle Application Server Web Cache Administrator's Guide*

Oracle Application Server Web Cache Features

Oracle Application Server Web Cache is a powerful solution for accelerating Web-based applications. The key features of Oracle Application Server Web Cache are divided into three categories:

- Compression and Caching
- Workload Management
- End-User Experience Management

See Also: *Oracle Application Server Web Cache Administrator's Guide* for detailed discussions of each of these features

Compression and Caching

Oracle Application Server Web Cache uses compression, caching, page assembly, and invalidation technologies to speed the delivery of dynamically-generated content and make more efficient use of low-cost hardware.

Automatic Content Compression

You can select to have OracleAS Web Cache compress both cacheable and non-cacheable documents. Because compressed documents are smaller in size, they are delivered to browsers faster with fewer round-trips, reducing overall latency. OracleAS Web Cache is able to compress text files by up to a factor of 10.

Full Page Static and Dynamic Content Caching

OracleAS Web Cache uses cacheablilty rules to store documents. There are rules for storing static content, and also rules for storing dynamically-generated content created using technologies such as JavaServer Pages (**JSP**). Supporting dynamic content caching allows OracleAS Web Cache to recognize multiple versions of documents with the same URL, cache session-aware pages, and cache pages that contain personalized information. There are also rules for pages that require personalized content assembly of dynamic Edge Side Includes (ESI) fragments.

Partial-Page Caching and Personalized Page Assembly

OracleAS Web Cache provides dynamic assembly of pages with both cacheable and non-cacheable page fragments. It does this by enabling pages to be broken down into fragments of differing cacheablilty profiles. With partial-page caching, more HTML content can be cached, then assembled and delivered by OracleAS Web Cache when requested.

The basic structure used to create dynamic content is a template page containing HTML fragments. The template consists of common elements, such as the "look and feel" elements of the page. The HTML fragments represent dynamic subsections of the page. The template page is associated with the **URL** that end users request. The template page uses the Edge Side Includes (ESI) markup language to tell OracleAS Web Cache to fetch and include the HTML fragments. Each individual fragment is a separate object with its own caching policy. ESI can be used with HTML, **XML**, and

any Web publishing technology. ESI is an open standard. For more information, see `http://www.esi.org`.

For JSP applications, OC4J supports JESI. JESI is a specification and custom **JSP tag** library that developers can use to automatically generate ESI code using JSP syntax. Even though JSP developers can always use ESI, JESI provides an even easier way for JSP developers to express the modularity of pages and the cacheablilty of those modules, without requiring developers to learn a new syntax. In addition, Oracle JDeveloper provides the ESI Servlet filter extension, which enables developers to create JSPs with ESI or JESI tags, and test them within the development environment.

Workload Management

Oracle Application Server Web Cache includes workload management features like surge protection, load balancing, failover, session binding, cache consistency options, and clustering to enhance application availability and ensure quality of service.

Surge Protection

OracleAS Web Cache passes requests for non-cacheable, stale, or missing objects to application servers. To prevent an overload of requests on the application servers, OracleAS Web Cache has a surge protection feature that enables you to set a limit on the number of concurrent requests that the application servers can handle. When the limit is reached, subsequent requests are queued. If the queue is full, then OracleAS Web Cache rejects the request and serves a site busy error page to the Web browser that initiated the request.

Web Server Load Balancing and Failover

Load balancing and failover allows Web sites to be built with a collection of **server**s for better **scalability** and **reliability**. OracleAS Web Cache sends requests to the application server with the most available load using its *load balancing* feature. When an application server becomes unavailable, OracleAS Web Cache automatically performs backend **failover**. OracleAS Web Cache distributes the load over the remaining application servers and polls the failed application server for its status until it is back online. When the failed server returns to operation, OracleAS Web Cache includes it in the load distribution.

Session Binding

OracleAS Web Cache enables you to bind user sessions to a given application server in order to maintain state for a period of time. An application binds user sessions by including session data in the HTTP header or body it sends to Web browsers in such a way that the browser is forced to include it with its next request. This data is transferred between the application server and the browser through OracleAS Web Cache.

Cache Invalidation and Expiration

OracleAS Web Cache supports invalidation as a way to ensure that its cache stays valid with respect to the content being served. Administrators and developers can invalidate cache content by either sending an invalidation message to the computer running OracleAS Web Cache or by assigning an expiration time limit to the cached documents.

Cache Consistency and Performance Assurance

OracleAS Web Cache provides several features for ensuring consistency between the cache and application servers, including:

- **Invalidation and Expiration:** OracleAS Web Cache supports invalidation as a way to ensure that its cache stays valid with respect to the content being served. Administrators and developers can invalidate cache content by either sending an invalidation message to the computer running OracleAS Web Cache or by assigning an expiration time limit to the cached documents. Expiration is useful for documents where the frequency of content changes is predictable and regular, such as standard images or templates.
- **HTTP Cache Validation:** OracleAS Web Cache uses HTTP/1.1 validation models to determine how to best serve a response to browsers. Validation works by the comparing two validators, one in the request header and the other in the cached object's response header, to determine if they represent the same or different entities.
- **Performance Assurance Heuristics:** To handle performance issues while maintaining cache consistency, OracleAS Web Cache uses built-in performance assurance heuristics that enable it to assign a queue order to documents. These heuristics determine which documents can be served stale and which documents must be refreshed immediately.

Oracle Application Server Cluster (Web Cache)

An OracleAS Cluster (Web Cache) is a loosely related set of Web cache instances working together to provide a single logical cache. You can configure multiple instances of OracleAS Web Cache to run as members of an OracleAS Cluster (Web Cache). This increases the availability and scalability of your caching solution.

End-User Experience Management

Oracle Application Server Web Cache also includes performance monitoring functionality that provides valuable insight into end-user service levels.

End-User Performance Monitoring

Oracle Application Server Web Cache includes instrumentation for end-user performance monitoring. Administrators can configure OracleAS Web Cache to measure end-user response times for individual URLs, sets of URLs, or entire Web-based applications, regardless of whether the URLs are cached. For each instrumented request, the complete user experience is recorded. The raw measurements are collected in the OracleAS Web Cache access logs.

Additional Oracle Application Server Web Cache Features

The following sections describe other important features of Oracle Application Server Web Cache.

Support for SSL

As part of Oracle Application Server, OracleAS Web Cache can be configured to use SSL and to work with SSL acceleration cards or third-party SSL acceleration appliances. OracleAS Web Cache supports applications that require client-side SSL certificates for PKI-based authentication. For HTTPS requests that require client-side certificates, the client browser sends its certificate to OracleAS Web Cache during the SSL handshake. The cache forwards the request to Oracle HTTP Server along with the

client's certificate information inserted in special HTTP request headers. Oracle HTTP Server recognizes the headers and is able to pass user credentials to Oracle Application Server Single Sign-On for authentication. OracleAS Web Cache can also perform SSL termination, and provide caching for applications that use `mod_osso`.

Flexible Multi-version Caching Rules

Administrators now have better control over the granularity of multi-version caching rules based on user-agent request headers, or browser types. Previously, you could either cache one version of a page for all browsers, or you could cache one version for each browser type and version. Now, you can customize the caching rules to define groups of browsers that will share a cached version of a page. For example, you could cache one page for all versions the Internet Explorer, one for all versions of Netscape, and one for all other browsers.

Integration with Oracle Process Manager and Notification Server (OPMN)

In addition to managing Oracle HTTP Server and OC4J processes, OPMN now manages the cache and administration server processes for OracleAS Web Cache. These include the start, stop, and auto-restart operations. However, standalone OracleAS Web Cache deployments will continue to use the OracleAS Web Cache Control and "watchdog" process management utilities.

Inline Invalidation and Search Key Invalidation

OracleAS Web Cache provides an inline invalidation mechanism as an additional way to manage content freshness. The inline invalidation model is implemented as part of the OracleAS Web Cache ESI support, and provides a useful way for origin servers to include invalidation messages along with transactional responses sent to Web Cache. The ability to send invalidation messages inline reduces the connection overhead associated with sending invalidations separately.

Another new invalidation feature for this release is support for search key invalidation. Previously, a cached document was identified by a URL-based cache key. Invalidation requests needed to specify either exact URLs or a set of URLs and headers matching a regular expression in order to invalidate cached objects. In this release, OracleAS Web Cache invalidation has been extended to support search keys. Cached objects can now be associated with multiple application-specified search keys, with the URL-based key being the primary key. Invalidation can be based on the search keys instead of the primary URL-based key, making invalidation easier for administrators and application developers to use.

Additional Caching Components

Web application developers may also encounter situations when application objects are not HTML or XML fragments; they may have to deal with XML DOM objects or Java serializable objects. There may also be requirements to reuse or post-process cached content, or maintain intermediate results. Oracle Application Server provides two components for dealing with application level caching:

- **Java Object Cache**
- **Web Object Cache**

These two cache offerings can be used independently as well as together to provide enhanced caching capabilities.

Java Object Cache

Java Object Cache is a set of Java classes designed to manage Java objects within a process, across processes, and on local disks. Java Object Cache provides a powerful and flexible service that improves server performance by managing local copies of objects that are expensive to retrieve or create. There are no restrictions on the type of object that can be cached or the original source of the object. The management of each object in the cache is easily customized. Each object has a set of attributes associated with it to control such things as how the object is loaded into the cache, where the object is stored, how it is invalidated, and who should be notified when the object is invalidated. Objects can be invalidated as a group or individually.

Web Object Cache

Web Object Cache is a Web-application-level caching facility that is embedded and maintained within a Java Web application. The Web Object Cache is a hybrid cache, both Web-based and object-based. Using the Web Object Cache, applications can cache programmatically using API calls (for servlets) or custom tag libraries (for JSPs). The Web Object Cache is generally used as a complement to the Web cache. By default, the Web Object Cache uses the Java Object Cache as its repository.

10

System Management

This chapter provides an overview of Oracle Application Server system management features and benefits. The topics include:

Introduction to System Management

Oracle Application Server provides a set of industry standards-based management facilities to simplify all aspects of Web site administration. It does so by leveraging a single security and directory framework for all applications, and by providing administrators with a management tool to manage, monitor, tune, and troubleshoot across Oracle Application Server **instance**s.

In the context of Oracle Application Server 10*g*, Oracle Enterprise Manager has two main components: the Oracle Enterprise Manager 10*g* Grid Control framework and the Oracle Enterprise Manager 10*g* Application Server Control framework.

Oracle Enterprise Manager 10*g* Grid Control Framework

The Oracle Enterprise Manager 10*g* Grid Control framework provides a unified view of your entire Oracle environment, including Oracle databases, Oracle Application Server, and Oracle Collaboration Suite. You can use Grid Control to monitor and manage your applications, hosts, and grids.

By deploying the Grid Control framework, you gain additional management capabilities such as the following:

- Automatic monitoring for all targets on the host system, with defaults set out-of-the-box to Oracle recommended settings
- Historical data collection for trend analysis
- Application Service Level Management for monitoring business transactions and understanding your Web application's end-user experience
- Configuration management for tracking hardware and software configurations and implementing changes throughout the enterprise

> **See Also:** *Oracle Enterprise Manager Concepts* for more information on Oracle Enterprise Manager 10*g* Grid Control

Oracle Enterprise Manager 10*g* Application Server Control Framework

The Oracle Enterprise Manager 10*g* Application Server Control framework provides you with the management tools you need to monitor and administer Oracle Application Server instances. Application Server Control is installed with every instance of Oracle Application Server.

Oracle Application Server uses Oracle Enterprise Manager 10*g* Application Server Control to enable Web site administrators to configure and administer Oracle Application Server instances, to monitor and optimize them for performance and **scalability**, and to help diagnose problems occurring with the application server.

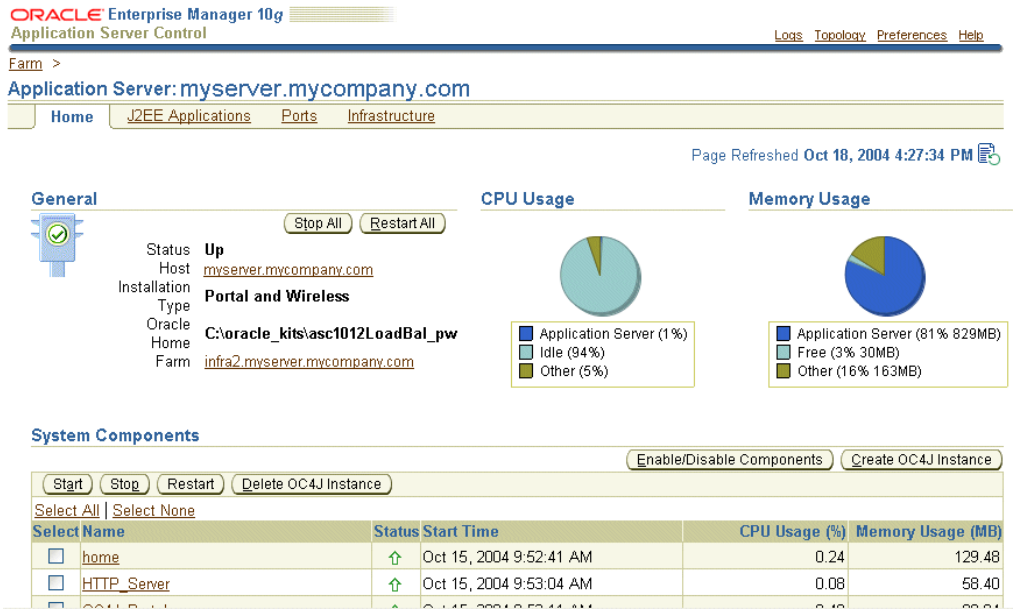
The remaining sections in this chapter provide an overview of Oracle Enterprise Manager 10*g* Application Server Control and describe its architecture and features.

Introduction to Oracle Enterprise Manager 10*g* Application Server Control

Oracle Enterprise Manager 10*g* Application Server Control provides Web-based management tools designed specifically for Oracle Application Server. Using Application Server Control, you can monitor and configure components of your application server. You can deploy applications, manage security, and create and manage Oracle Application Server clusters.

Figure 10–1 shows an example of the Oracle Enterprise Manager 10*g* Application Server Control Console.

Figure 10–1 Oracle Enterprise Manager 10g Application Server Control Console



Application Server Control consists of the following:

- **Oracle Enterprise Manager 10*g* Application Server Control Console:** The Enterprise Manager Web-based user interface for managing Oracle Application Server 10*g*. The Application Server Control Console is installed and available with every Oracle Application Server 10*g* installation.

  From the Application Server Control Console, you can monitor and administer a single Oracle Application Server instance, a farm of Oracle Application Server instances, or Oracle Application Server Clusters.

- **Oracle Enterprise Manager 10*g* Application Server Control underlying technologies**: Application Server Control relies on various underlying technologies to discover, monitor, and administer the Oracle Application Server environment. These technologies include:
  - Oracle Dynamic Monitoring Service (DMS)
  - Oracle Process Manager and Notification Server (OPMN)
  - Distributed Configuration Management (DCM)
  - A local version of the Oracle Management Agent specifically designed to gather monitoring data for the Application Server Control Console.

    See Also: "Introduction to Administration Tools" in the *Oracle Application Server Administrator's Guide*

Oracle Enterprise Manager 10*g* Application Server Control Architecture

Oracle Enterprise Manager 10*g* Application Server Control provides immediate, out-of-the-box management value with each Oracle Application Server instance you install. Each Oracle Application Server installation includes an Application Server Control for managing that installation. Application Server Control is based on several underlying pieces that comprise the application server management stack, including Distributed Configuration Management (DCM), Oracle Process Manager and Notification Server (OPMN), and Dynamic Monitoring Service (DMS).

Application Server Control relies on various technologies to discover, monitor, and administer the Oracle Application Server environment. These technologies include:

- **Distributed Configuration Management (DCM):** DCM manages configurations among application server instances that are associated with a common Metadata Repository. It enables Oracle Application Server cluster-wide deployment so you can deploy an application to one instance and have it automatically propagated to the entire cluster. You can also make a single host or instance configuration change to one instance and have it propagated across all instances in the cluster. Application Server Control uses DCM to make configuration changes and to propagate configuration changes and deployed applications across the cluster.
- **Oracle Process Manager and Notification Server (OPMN):** OPMN provides process control and monitoring for application server instances and their components. It gathers component status information, and distributes the status information to components that are interested in it. Application Server Control uses OPMN for such tasks as starting and stopping the components of your application server instance.
- **Oracle Management Agent:** The Oracle Management Agent is a component of Oracle Enterprise Manager that gathers monitoring data for Application Server Control.
- **Oracle Dynamic Monitoring Service (DMS):** The Management Agent leverages another underlying service, the Dynamic Monitoring Service (DMS), to collect performance data. Oracle Application Server components are instrumented with DMS to provide a comprehensive set of built-in performance metrics to automatically measure runtime performance statistics. As a result, Application Server Control uses this data to monitor the duration of important phases of request processing, as well as status information, such as the number of requests being handled at any given time.

See Also:

- *Distributed Configuration Management Administrator's Guide*
- *Oracle Process Manager and Notification Server Administrator's Guide*

Oracle Enterprise Manager 10*g* Application Server Control Features

Oracle Enterprise Manager 10*g* includes the following features that enable you to manage your Oracle Application Server framework:

- Complete Oracle Application Server Administration
- Monitoring Oracle Application Server

Complete Oracle Application Server Administration

Application Server Control provides a full set of features for performing Oracle Application Server administration, with Web-based interfaces for performing operations such as:

- Starting and stopping services
- Modifying server configuration parameters
- Creating new Oracle Application Server Containers for J2EE (OC4J) instances and adding Java Virtual Machines (JVMs)
- Configuring J2EE resources such as Java Database Connectivity (JDBC) data sources and Java Authentication and Authorization Service (JAAS) providers for J2EE application security
- Deploying J2EE and Web Services Applications
- Managing additional application server components such as Oracle Application Server Business Intelligence
- Creating and managing clusters that speed up the configuration and deployment of your Web applications
- Viewing a graphical topology of your application server environment
- Managing application server port values from a single, central location
- Locating and reviewing application server log files to quickly diagnose problems

Monitoring Oracle Application Server

After you have installed and configured Oracle Application Server, one of your primary tasks as an administrator is to monitor the application server for both performance and availability. Oracle Enterprise Manager 10*g* Application Server Control Console allows you to quickly determine the state of the application server and its components, and to efficiently gauge the performance of components and the applications deployed on them.

Application Server Control allows you to take a top-down approach to your monitoring activities. For example, you can start by reviewing the basic characteristics of your application server on the Application Server Home page, and then drill down to examine the performance of individual components.

Using this approach, Application Server Control can help you identify high-level performance issues, such as a high CPU load on your system. You can then drill down

to individual components or applications to isolate the root cause of the high-level problem.

See Also:

- *Oracle Application Server Administrator's Guide*
- *Oracle Application Server Performance Guide*

11

Security and Identity Management

This chapter provides an overview of Oracle Application Server security solutions. The topics include:

Introduction to Security

Oracle Application Server provides a comprehensive integrated security framework supporting all of its components, as well as third party and custom applications deployed on Oracle Application Server. The framework is based on Oracle Identity Management for single sign-on, user administration, group management, and provisioning.

In addition to the components involved in Identity Management, other Oracle Application Server components are also involved in providing security for your online applications. The main components involved are Oracle Application Server Web Cache, Oracle HTTP Server, Oracle Application Server Portal, Oracle Application Server Single Sign-On, Oracle Internet Directory, Oracle Application Server Certificate Authority, and Oracle Application Server Metadata Repository.

See Also: *Oracle Application Server Security Guide*

Introduction to Identity Management

Identity management is the process by which the complete security life cycle for network entities is managed for an organization.

Identity management most commonly refers to the management of an organization's application users, where steps in the security life cycle include account creation, suspension, privilege modification, and account deletion. The network entities managed may also include devices, processes, applications, or anything else that needs to interact in a networked environment. Entities managed by an identity management process may also include users outside of the organization, such as customers and trading partners.

Identity management is important to IT deployments because it can reduce administrative costs while at the same time improving security. Identity Management benefits include:

- Identity management saves money. For most enterprises, application user administration is a very expensive, laborious, and error-prone process. Identity

management centralizes and automates many user administration tasks, reducing costs while improving accuracy and security.

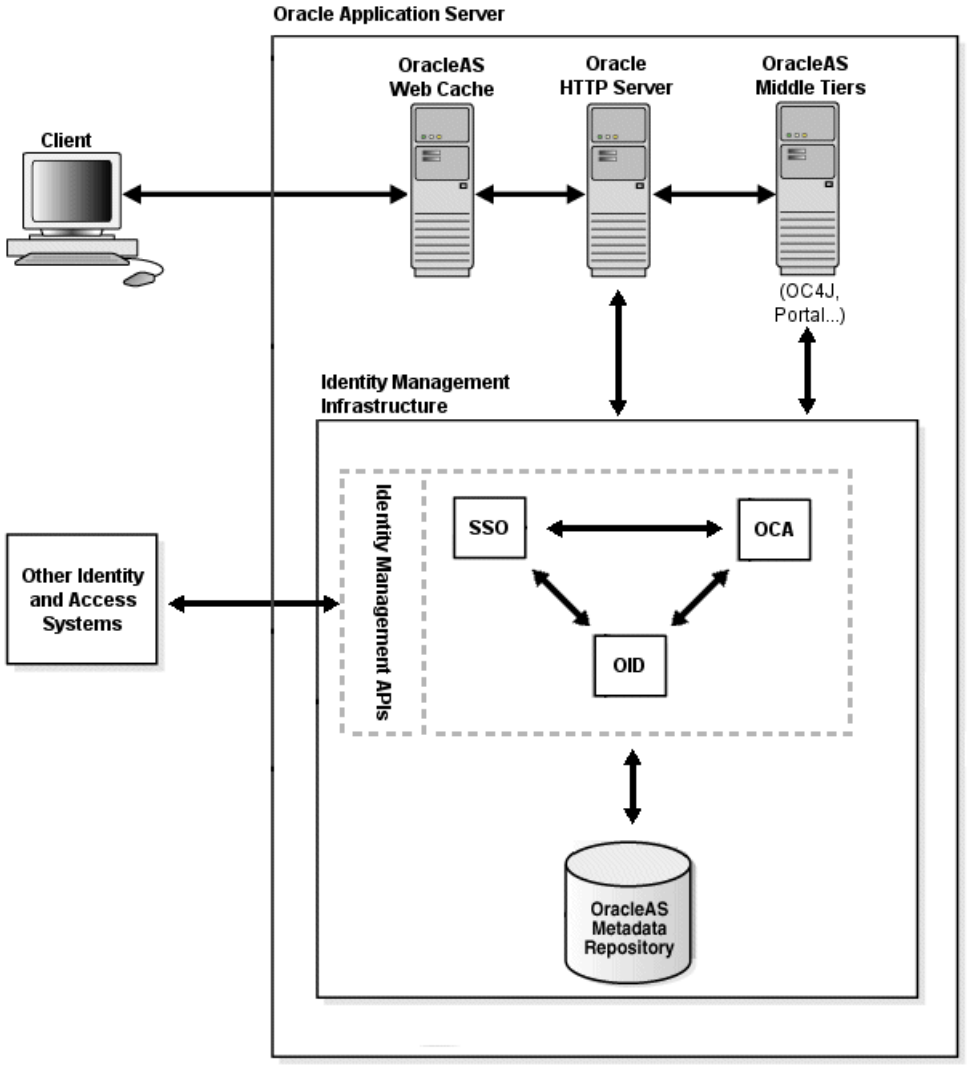
- Identity management enables faster deployments. Typically, provisioning of a new application means creating and managing separate user accounts and their privileges. Identity management enables the new applications to leverage the existing infrastructure for its user management, and thus reduces the time it takes to deploy and manage new applications.
- Identity management improves the end-user experience. An identity management strategy allows new users to gain access to their applications quickly, eliminating wasted employee time. By providing single sign-on for all applications, users no longer have to keep track of different login and password information for different purposes. Further, identity management enables a customized application experience, enhancing usability.
- Identity management improves application security. An identity management strategy allows users to have their passwords and security credentials managed centrally. This reduces the temptation for users to write down security information, raising the risk of unauthorized access.

Security Architecture

Figure 11–1, illustrates how the elements of Oracle Application Server function together. Following is a list of the functionality of the various components:

- **OracleAS Web Cache** is positioned closest to the client, where it provides efficiency and performance.
- **Oracle HTTP Server** is the front-end Web server for Oracle Application Server. Through **Apache**-based modules as well as modules developed by Oracle, users can access a variety of Oracle Application Server services.
- **OracleAS Portal** provides the infrastructure, including the ability to create and manage Web pages. It lets you display multiple portlets on each Web page, with links to content through **Java** applications.
- The **Java** engine lies underneath **Oracle Application Server Web Cache** and Oracle HTTP Server, supporting their ability to link efficiently.
- **OracleAS Single Sign-On** enables users to log in to Oracle Application Server and gain access to those applications for which they are authorized, without requiring them to re-enter a user name and password for each application.
- **Oracle Internet Directory** is a general purpose directory service that enables fast retrieval and centralized management of information about dispersed users and network resources.
- **Oracle Application Server Certificate Authority** generates and publishes X.509v3 certificates to support PKI-based (strong) authentication methods.
- **Oracle Application Server Metadata Repository** is an Oracle database used to hold metadata, including identity information.

Figure 11–1 Oracle Application Server Security Architecture



Security Components and Features

The Oracle Application Server security framework involves many components, each of which contributes features that enable you to secure your Oracle Application Server deployment:

- Oracle Identity Management
- Oracle HTTP Server Security
- OracleAS Web Cache Security
- Portal Security

Oracle Identity Management

Oracle Identity Management is an integrated infrastructure that Oracle products rely on for distributed security. The Oracle Identity Management infrastructure includes the following components:

- Oracle Internet Directory, a scalable, robust LDAP V3-compliant directory service implemented on the Oracle Database.
- Oracle Directory Integration and Provisioning Platform, a component of Oracle Internet Directory, which consists of two parts:
  - Directory Provisioning Integration Service, which sends notifications to target applications to reflect changes including creating and deleting users, as well as changes to a user's status or information
  - Directory Integration, which allows you to synchronize data between Oracle Internet Directory and other connected directories, and develop and deploy custom connectors
- Oracle Delegated Administration Services, which provides self-customizing administration of directory information by users and application administrators.
- Oracle Application Server Single Sign-On, which provides single sign-on access to Oracle and third-party Web applications.
- Oracle Application Server Certificate Authority, which generates and publishes X.509v3 certificates to support PKI-based (strong) authentication methods.

While Oracle Identity Management is designed to provide an enterprise infrastructure for Oracle products, it may also serve as a general-purpose, robust, and scalable enterprise-wide identity management platform for user-written and third-party applications, hardware, and network operating systems. Custom applications may leverage Oracle Identity Management through a set of documented and supported services and APIs, for example:

- LDAP APIs for C, Java, and PL/SQL, each compatible with other LDAP SDKs.
- Oracle Delegated Administration Services Web-based services for building customized administration interfaces that manipulate directory data.
- Oracle Directory Integration Services to facilitate the development and deployment of custom solutions for synchronizing Oracle Internet Directory with third-party directories and other user repositories that are not configurable using existing Oracle connectors.
- Oracle Provisioning Integration Services for provisioning Oracle and third-party applications, as well as a means of integrating the Oracle environment with other provisioning systems.
- Oracle Application Server Single Sign-On provides APIs for developing and deploying partner applications that share a single sign-on session with other Oracle Web applications.
- Oracle Application Server Java Authentication and Authorization Service (JAAS) Provider, the Oracle implementation of the JAAS standard, allows applications developed for the Web using the Oracle J2EE environment to leverage the identity management infrastructure for authentication and authorization.

In addition, Oracle works with third-party application vendors to ensure their applications can leverage Oracle Identity Management out of the box.

See Also: *Oracle Identity Management Integration Guide*

Oracle Application Server Single Sign-On

An important security feature of Oracle Application Server is support of single sign-on to Web-based applications. Oracle Application Server Single Sign-On addresses the problem of "too many passwords." With the rapid growth of the Internet, this problem

has become increasingly prevalent, causing users inconvenience that typically results in poor security practices and increased administrative costs.

Oracle Application Server Single Sign-On resolves this problem by enabling users to log in to Oracle Application Server and gain access to those applications for which they are authorized, without requiring them to re-enter a user name and password for each application.

It is fully integrated with Oracle Internet Directory, which stores user information. It supports LDAP-based user and password management through Oracle Internet Directory.

Oracle Application Server Single Sign-On provides the following functionality:

- It authenticates users and passes their identities securely to partner applications, such as **Oracle Application Server Portal**. It prompts users for a username and password when they access the system for the first time in a given time period.
- It uses **cookie**s, which are formatted pieces of information stored on a **browser client** by a Web **server**. Cookies allow Web servers to store and retrieve information about the client user, effectively maintaining client state information in the otherwise stateless Web environment.
- It supports Public Key Infrastructure (PKI) client authentication, which enables PKI authentication to a wide range of Web applications. By means of an **API**, Oracle Application Server Single Sign-On can integrate with third-party **authentication** mechanisms such as Netegrity SiteMinder.

With Oracle Application Server Single Sign-On, users typically sign on to a centrally administered Single Sign-On Server through a designated Web **portal**. Once it authenticates the user, Single Sign-On Server displays links to all the applications for that user.

Using a centrally administered Single Sign-On Server has these advantages:

- **Convenience:** The user enters the user name and password only once, at a central corporate Web portal, to access all the needed applications. From the user's perspective, authentication to each application happens transparently.
- **Increased Security:** Fewer user name and password combinations lower the risk of unauthorized access to a user's restricted information.
- **Ease of Administration:** Oracle Application Server Single Sign-On provides centralized provisioning of user accounts, so that administrators can easily create new user accounts. Centralizing the authentication process also makes it possible to support additional authentication mechanisms in a localized manner. For example, you can implement password-based authentication, using Single Sign-On and Oracle Internet Directory, then switch to digital certificate-based authentication using OracleAS Certificate Authority, Single Sign-On, and Oracle Internet Directory, and the change would be localized to the Single Sign-on Server.

Partner and External Applications There are two kinds of applications to which Oracle Application Server Single Sign-On provides access:

- Partner Applications
- External Applications

Partner applications are integrated with the Single Sign-On Server. They are built upon an Oracle Application Server Single Sign-On API that enables them to delegate authentication to the Single Sign-On Server.

External applications are Web-based applications that retain their authentication logic. They do not delegate authentication to the Single Sign-On Server and, as such, require a user name and password to provide access. Currently, these applications are limited to those which employ an **HTML** form for accepting the user name and password. The user name may be different from the single sign-on user name, and the Single Sign-On Server provides the necessary mapping.

The single sign-on offering in Oracle Application Server is a critical differentiator for users seeking a robust, fully integrated single sign-on architecture. Oracle Application Server leverages **JAAS**, as well as Oracle Internet Directory, to deliver a comprehensive end-to-end security infrastructure across the entire Oracle Application Server product.

Oracle Internet Directory

Oracle Internet Directory is a critical component of Oracle Application Server management and security infrastructure. It ensures that user accounts and groups are managed centrally through the LDAP Version 3 standard. Oracle Application Server enables users to be created centrally in Oracle Internet Directory and shared across all components in Oracle Application Server. When users log in, they are authenticated once by Oracle Application Server Single Sign-On against their Oracle Internet Directory credentials, and can thereby access multiple applications seamlessly.

Oracle Delegated Administration Services Oracle Internet Directory includes Oracle Delegated Administration Services, which provides trusted proxy-based administration of directory information by users and application administrators. Oracle Delegated Administration Services includes a Self-Service Console, an easy-to-use, Web-based interface which allows end-users and application administrators to search for and manage data in the directory. Through this console, Oracle Delegated Administration Services provides Oracle Application Server with a means of provisioning end-users in the Oracle Application Server environment. Oracle Internet Directory also enables components of Oracle Application Server to broadcast data about users and group events, so that those components can update any user information stored in their local application instances.

Oracle Directory Integration and Provisioning Oracle Directory Integration and Provisioning enables customers to synchronize data between various directories and Oracle Internet Directory. The Oracle Directory Integration Platform is a set of services and interfaces that make it possible to develop synchronization solutions with third party metadirectories and other enterprise repositories, such as SunONE/iPlanet Directory Server. With Oracle Application Server, Oracle Internet Directory includes connectors for out-of-the-box synchronization with Oracle Human Resources, Active Directory, and SunONE/iPlanet Directory Server 4.2 and 5.0.

Oracle Internet Directory also provides a plug-in framework for applications that require customized functionality, such as referential integrity of data. The plug-in framework is delivered as a highly-flexible **PL/SQL** interface, allowing user-defined operations to be invoked by the **directory server** before, after, or in place of standard LDAP commands.

User and Directory Searches Oracle Internet Directory provides users with directory searches capabilities with sophisticated server-side **caching** capabilities. Oracle Internet Directory also provides two key features that ensure administrators can deliver seamless directory services to all users:

- **Alias De-reference:** When a user or an application searches on an alias, Oracle Internet Directory automatically de-references the alias and returns the entry to

which it refers. This feature enables administrators to change the names of **object**s in ways that are transparent to users and applications.

- **Enhanced Proxy Capabilities:** Administrators can safely establish performant, auditable middle-tier application access to the directory on behalf of end user communities.

Key Directory Features Oracle Internet Directory provides the following key directory features:

- Native LDAP server complying with all LDAP v2 and v3 **RFC**s, along with the Open Group's "LDAP CERTIFIED" test specifications (VSLDAP test suite 2.2)
- Supports the **X.500** information, naming, and storage model
- Extensible directory schema for online modifications with no downtime
- LDAP APIs in Java, C, and PL/SQL to assist with application development

Using Oracle Internet Directory with Middle Tier Components The middle tier components use Oracle Internet Directory in the following ways:

- Application server instances and infrastructures store security and management information in Oracle Internet Directory. Oracle Internet Directory stores users' information, such as user names and privileges, required for internal operation of the application server.
- Oracle Application Server Java Authentication and Authorization Service (JAAS) Provider stores **realm** and JAAS **policy** in Oracle Internet Directory.
- Oracle Application Server Single Sign-On validates the user name and password against user and group profiles stored in Oracle Internet Directory.

See Also: *Oracle Internet Directory Administrator's Guide*

Oracle Application Server Certificate Authority

Oracle Application Server Certificate Authority (OCA) is a component of the Oracle public key infrastructure (PKI) offering that allows you to create and manage X.509v3 digital certificates for use in Oracle or third-party software. The Certificate Authority is fully standards-compliant, and is fully integrated with Oracle Application Server Single Sign-On and Oracle Internet Directory. Oracle Application Server Certificate Authority provides Web-based certificate management and administration, as well as XML-based configuration. It leverages the identity management infrastructure, high availability, and scalability of the Oracle platform.

Java Authentication and Authorization Service (JAAS)

Oracle Application Server provides an implementation of Java Authentication and Authorization Service (**JAAS**) that integrates with the Oracle Application Server J2EE security infrastructure to enforce security constraints for Web (**servlet**s and **JSP**s) and **EJB** components.

Oracle Application Server Java Authentication and Authorization Service (JAAS) Provider support provides the following benefits:

- Integrates **Java**-based applications with Oracle Application Server Single Sign-On, includes authentication, thereby giving you extensible security for Java-based applications
- Manages access control policies centrally in Oracle Internet Directory, controls access by role, and partitions security **policy** by subscriber

- Supports impersonation of a specific user, allows an enterprise bean, servlet, or JSP to run with the permissions associated with the current client or a specified user

OracleAS Web Cache Security

Oracle Application Server Web Cache provides the following security-related features:

- Restricted Administration
- Secure Sockets Layer (SSL) Support

Restricted Administration

OracleAS Web Cache restricts administration with the following features:

- Password **authentication** for administration and invalidation operations
- Control over ports from which administration and invalidation operations can be requested
- IP and subnet administration **restriction**s

Secure Sockets Layer (SSL) Support

The secure sockets layer (**SSL**) protocol, developed by Netscape Corporation, is an industry standard for network transport layer security. SSL provides authentication, **encryption**, and data integrity in a public-key infrastructure (PKI). By supporting SSL, OracleAS Web Cache is able to cache pages from HTTPS requests.

Note that HTTPS traffic can be process intensive. If OracleAS Web Cache needs to have traffic travel over the open Internet, then configure OracleAS Web Cache to send HTTPS requests to the application server. If traffic only travels through a LAN in a data center, then the traffic can be sent with **HTTP** so as to reduce the load on the application servers.

SSL interacts with the following entities:

- **Certificate Authority:** A certificate authority (**CA**) is a trusted third party that certifies the identity of third parties and other entities, such as users, databases, administrators, **client**s, and **server**s. The CA verifies the party identity and grants a certificate, signing it with its **private key**. The OracleAS Web Cache certificate must be signed by a CA. Oracle Application Server provides Oracle Application Server Certificate Authority (OCA), which allows you to create and manage X.509v3 digital certificates for use in Oracle or third-party software. For more information on OCA see "Oracle Application Server Certificate Authority" in this chapter.
- **Certificate:** A **certificate** is created when a party's **public key** is signed by a trusted CA. A certificate ensures that a party's identification information is correct, and that the public key actually belongs to that party. A certificate contains the party's name, public key, and an expiration date, as well as a serial number and certificate chain information. It can also contain information about the privileges associated with the certificate. When a network entity receives a certificate, it verifies that it is a trusted certificate, one issued and signed by a trusted CA.
- **Wallet:** A **wallet** is a transparent database used to manage authentication data such as keys, certificates, and trusted certificates needed by SSL. A wallet has an X.509 version 3 certificate, private key, and list of trusted certificates.

Security administrators use the **Oracle Wallet Manager** to manage security credentials on the OracleAS Web Cache server. Wallet owners use it to manage security credentials on clients. Specifically, Oracle Wallet Manager is used to do the following:

- Generate a public-private **key pair** and create a certificate request for submission to a CA
- Install a certificate for the identity
- Configure trusted certificates for the identity

To support HTTPS between browsers and OracleAS Web Cache, configure a wallet on the OracleAS Web Cache server for each supported site. To support HTTPS between OracleAS Web Cache and the application servers, configure a wallet on the application server.

See Also: *Oracle Application Server Security Guide*

Oracle HTTP Server Security

Oracle HTTP Server provides the following security-related features:

- Session Renegotiation Support
- SSL Hardware Acceleration Support
- Port Tunnelling
- OHS to OC4J SSL Support

Session Renegotiation Support

With session renegotiation support, individual directories can be protected by different levels of encryption. Some directories may only need a minimum level of encryption, while others require stronger encryption.

SSL Hardware Acceleration Support

Software-based SSL encryption can sometimes be slow. Oracle HTTP Server supports the option of having dedicated SSL hardware through nCipher. nCipher is a third-party math accelerator that improves the performance of the PKI cryptography that SSL uses.

Port Tunnelling

Port tunneling lets all communication between Oracle HTTP Server and OC4J happen on a single port or a small number of ports. Previously, the firewall configuration had to include port information for several ports to handle communication between Oracle HTTP Server and multiple OC4J instances. With the port tunnel, a daemon routes requests to the appropriate OC4J instances. Using this method, only one port has to be opened through the firewall regardless of the number of OC4J instances involved.

OHS to OC4J SSL Support

Oracle HTTP Server and OC4J can communicate using the AJP protocol over the Secure Sockets Layer (SSL). Previously, OHS and OC4J used the AJP 13 protocol unencrypted, without support for authentication. Now, OHS has been modified to extend support to the AJP13 protocol over SSL.

Portal Security

This section discusses the following security topics:

- User Authentication in OracleAS Portal
- Access Control in OracleAS Portal

User Authentication in OracleAS Portal

Oracle Application Server Single Sign-On provides a single point of validation for **portal** user credentials and governs user access to intranet resources based on employee profiles. When a user logs into a portal page, the single sign-on **server** validates the user name and credentials against user profiles stored in Oracle Internet Directory or another user credential repository.

Oracle Application Server Single Sign-On also supports external, partner applications. For these applications, the single sign-on server logs in to the application automatically for the user.

See Also: Chapter 10, "System Management"

Access Control in OracleAS Portal

Most portal elements have an access control list (**ACL**). This list controls which users and groups may access the element, and to what extent. For example, if you wanted all of the users in a group to be able to see the items on a portal page, in the page's ACL you would grant that group View privileges.

Besides ACLs, you can also use global privileges to grant access to all **object**s of a given type in **Oracle Application Server Portal**. For example, granting the Create privilege for All Pages to a group enables all members of that group to create pages.

12

Enterprise Deployments

This chapter provides an overview of the Oracle Application Server recommended enterprise deployments. The topics include:

Introduction to Oracle Application Server Enterprise Deployments

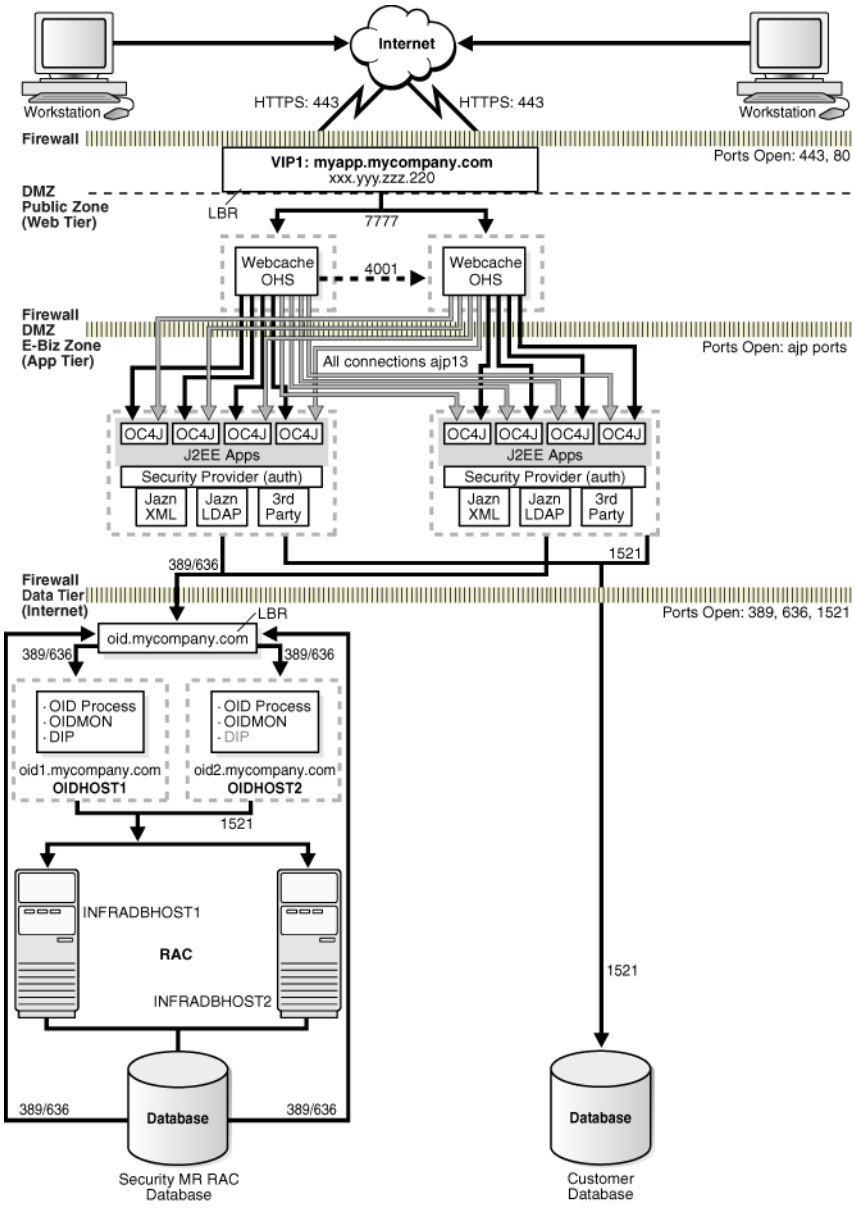
The Oracle Application Server enterprise deployment is a configuration that supports large-scale, mission-critical business software applications. The hardware and software in an enterprise deployment delivers the following:

- High quality service:
  - The system workload is managed and balanced effectively.
  - Applications continue to operate when hardware and other resources are added or removed.
  - System maintenance and unexpected failures cause zero downtime.
- Built-in security:
  - All incoming network traffic is received by the load balancing router on a single, secure port and directed to internal IP addresses within the firewall; inside the firewall, functional components are grouped within DMZs.
  - User accounts are provisioned and managed centrally
  - Delegation of administration is performed consistently
  - Security systems are integrated
- Efficient software provisioning and management:
  - Application distribution is simple.
  - Systems are managed and monitored as one logical unit in a central console.
  - Death detection and restart mechanisms ensure availability.

Standard Enterprise Deployment for J2EE Applications

Figure 12–1 illustrate the enterprise deployment architecture for J2EE applications.

Figure 12–1 Enterprise Deployment Architecture for myJ2EECompany.com



In the J2EE application enterprise deployment architecture, users access the application through an HTTPS connection to the load balancer for myapp.mycompany.com in the **demilitarized zone** (DMZ), behind the first firewall. The load balancer accesses OracleAS Web Cache and Oracle HTTP Server instances in the Web tier to route requests through the second firewall to available OracleAS Containers for J2EE (OC4J) instances in the application tier.

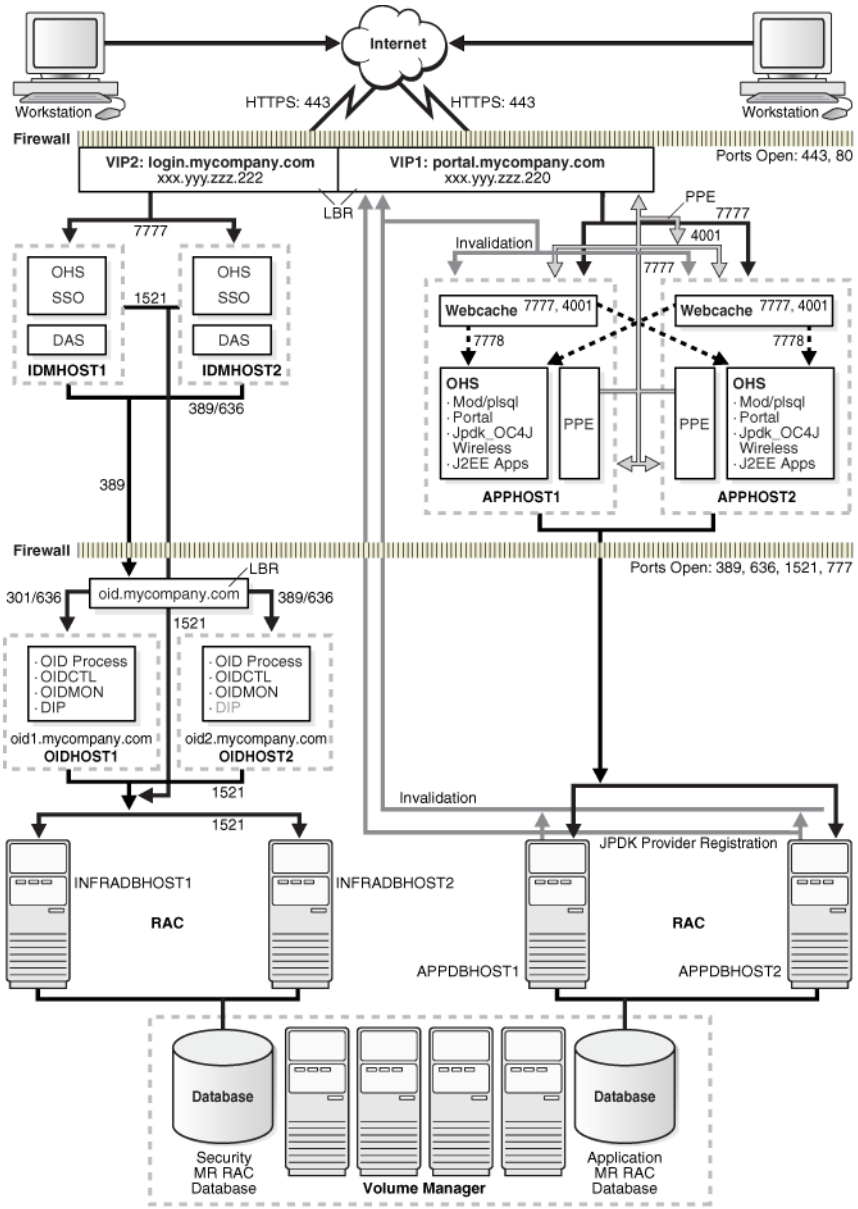
As needed, the application tier servers access the data tier through a third firewall. The data tier contains the Oracle Application Server Infrastructure as well as any the customer database.

> **See Also:** *Oracle Application Server Enterprise Deployment Guide* for detailed information about the enterprise deployment architecture for J2EE applications

Standard Enterprise Deployment for Portal Applications

Figure 12–1 illustrate the enterprise deployment architecture for OracleAS Portal applications.

Figure 12–2 Enterprise Deployment Architecture for myPortalCompany.com



In the OracleAS Portal enterprise deployment architecture, users access the application through an HTTPS connection to the load balancers in the **demilitarized zone** (DMZ), behind the first firewall. If the user needs to log in, the request will go to login.mycompany.com. If not, the request will go to portal.mycompany.com.

The portal.mycompany.com load balancer routes requests to one of two OracleAS Web Cache servers, which are part of the two application middle tiers deployed in the DMZ.

The login.mycompany.com load balancer routes requests to one of the two Oracle HTTP Servers, which are part of the two Identity Management middle tiers, also deployed in the DMZ.

The middle tier deployments in the DMZ access the data tier through a firewall. The data tier contains a two-node RAC database installation, one for the security infrastructure metadata repository and one for the application metadata repository. The data tier also contains two Oracle Internet Directory standalone servers, which also access the security infrastructure metadata repository.

> **See Also:** *Oracle Application Server Enterprise Deployment Guide* for detailed information about the enterprise deployment architecture for Portal applications

Glossary

A2A

Application to application (A2A) integration (also known as enterprise **application integration**) is the integration of applications and business processes within the same company (also known as an enterprise).

access control list

See ACL.

access control list model

The access control list model is a method for organizing **authorization** information. The access control list model is resource-centered or **object**-centered. This means that authorization information is associated with objects.

ACID characteristics

A **transaction** has ACID characteristics (Atomic, Consistent results, Isolated, and Durable), if it demonstrates the following:

- If interrupted by failure, all effects are undone or rolled back (Atomic).
- The effects of a transaction preserve invariant properties (Consistent results).
- Its intermediate states are not visible to other transactions. Transactions appear to execute serially, even if they are performed concurrently (Isolated).
- The effects of a completed transaction are persistent; they are never lost (except in a catastrophic failure) (Durable).

ACL

An access control list (ACL) is a list of groups and users authorized for specific access to an **object**. It can also be a list of entities, together with their access rights, which are authorized to have access to a specified resource.

Apache

Apache is a public domain **HTTP server** derived from the National Center for Supercomputing Applications (NCSA).

Apache module

An Apache module is an add-on to either **Oracle HTTP Server** or **Apache**. Modules extend the basic functionality of the Web **server** and support integration between the **Oracle HTTP Server** and other Oracle Application Server components.

API

An application program interface (API) is a set of exposed data structures and functions that an application can use to invoke services on a component.

applet

An applet is a **Java** program that runs from an applet viewer or a Web page.

application integration

Application integration is the ability to link different types of enterprise applications and business processes together so that they can smoothly and effectively communicate to conduct **e-business**. These applications can reside within a company's enterprise boundaries or across multiple company boundaries connected over the Internet.

application-to-application

See A2A.

authentication

Authentication ensures that access to static pages, **CGI** scripts, and applications is limited to authorized users. When access to an application or document is protected by an authentication scheme, the **client** sends identification information to the **server**, which checks if the client is authorized to access the **object**.

See Also: **security scheme**, **restriction**

authentication broker

An authentication broker is the portion of the **authentication server** that responds to and evaluates **authorization** requests.

authentication provider

An authentication provider is an **object** that specifies all of the **realm**s used to implement a particular **security scheme**. It is a code **module** that runs within the **authentication server** and implements a particular security scheme.

authentication server

An authentication server is an **object** that encapsulates the **authentication** performed against applications. It consists of one **authentication broker** object and several **authentication provider** objects.

authorization

Authorization is the evaluation of security constraints to send a **message** or make a request. Authorization uses specific criteria to determine whether the request should be permitted. The criteria are **authentication** and **restriction**.

See Also: **authentication**, **restriction**

automatic deployment

Automatic deployment is a method for re-deploying applications, **servlet**s, or **JSP**s after changes have been made to the application code that does not require updating the **server** configuration files or restarting the application server. With automatic deployment, the application server detects the changes in code and automatically re-deploys the applications.

See Also: **hot deployment** for comparison

availability

Availability is the percentage or amount of scheduled time that a computing system provides application service.

B2B

Business-to-business (B2B) integration is the integration of a company's applications and business processes with external business systems operated by customers, suppliers, trading partners, exchanges, and marketplaces. B2B integration extends **A2A** integration to the Internet.

basic authentication

Basic authentication is a username-and-password-based **authentication** scheme that does not encrypt passwords when sending them over the Internet.

See Also: **digest authentication**, **authentication**

bean-managed persistence (BMP) bean

A bean-managed persistence (BMP) bean is a **Java** bean that stores all state information within itself.

browser client

A browser client is a client that can access static pages, **CGI** scripts, and applications via a **URL** over **HTTP**, **HTTPS (secure HTTP)**, or **IIOP**.

business intelligence

Business intelligence is information describing your business, data, and Web site traffic.

business-to-business

See B2B.

CA

A certificate authority (CA) is a trusted third party that vouches for the identity of an individual, company, or **server** and signs a **certificate**.

See Also: **trustpoint**

caching

Caching is the act of storing information that is frequently accessed in a location where it can be accessed quickly. For example, **Oracle Application Server Web Cache** stores dynamically generated Web pages locally and serves them in response to incoming requests. This reduces the total time spent handling the request by avoiding costly connections to the back-end database and other Web site bottlenecks.

canonical

Authoritative or officially approved. The term is generally used to describe whether or not a programming interface follows an accepted standard.

capability model

A capability model is a method for organizing **authorization** information. The capability model is **principal**-centered or **subject**--centered. This means that

authorization information is associated with subjects.The **Java 2 Security Model** is an example of a capability-based system.

cascading style sheets

See CSS.

certificate

A certificate is a specially formatted data item signed by a trusted third party to attest to the validity of the item's information. Public-key certificates use a **CA**'s signature to attest that the enclosed **public key** belongs to the **principal** identified by the enclosed name.

certificate authentication

Certificate authentication is an **authentication** method in which **client**s identify themselves using X.509 v3 **certificate**s.

certificate authority

See CA.

certificate revocation list

See CRL.

CGI

Common Gateway Interface (CGI) is the industry-standard technique for transferring information between a Web **server** and any program designed to accept and return data that conforms to the CGI specifications.

character set

A character set is a set of symbols used to write one or more human languages, as defined by **RFC** 1521.

clickstream

A clickstream is a virtual trail that a user leaves behind while moving from place to place on the Internet. A clickstream is a record of a user's activity on the Internet, including every Web site and every page of every Web site that the user visits, how long the user was on a page or site, in what order the pages were visited, any newsgroups that the user participated in, and even the e-mail addresses of mail that the user sends and receives.

client

A client is a user, software application (such as a browser), or computer that requests the services, data, or processing of another application or computer (the **server**).

cluster

A cluster is a collection of application server **instance**s with identical configuration and application deployment. Clusters enforce homogeneity between member **instance**s so that a cluster of application server instances can appear and function as a single instance. With appropriate front-end load balancing, any instance in an application server cluster can serve **client** requests. This simplifies configuration and deployment across multiple instances and enables **fault tolerance** among clustered instances.

clustering

Clustering is the process of collecting Oracle Application Server **instance**s into **cluster**s for load balancing and **fault tolerance**. With clustering, the collection of application server **instance**s are treated as a pool to service incoming requests. If one instance does not respond, then the request is forwarded to another instance in the cluster. The load balancer also maintains session context, so that when a **client** reconnects, its request is sent to the application server instance that was previously serving it.

collaboration protocol agreements

See CPA.

collaboration protocol profile

See CPP.

common gateway interface

See CGI.

common object request broker architecture

See CORBA.

concurrency

Concurrency is the ability to handle multiple requests simultaneously. Threads and processes are examples of concurrency mechanisms.

connect string

A connect string is the set of parameters, including a protocol, that is used to connect to a specific database instance on the network. Other names for a connect string include: SQL*Net V2 service name and Net8 connect string.

container

A container is a component that contains other components, such as a **servlet**. A container executes and manages a servlet. A container is either part of or associated with and used by a Web **server**. When a **client HTTP** request calls a servlet, the Web server passes the HTTP request to the container. The container translates the HTTP request into a **Java** method invocation and then passes the request to the servlet.

See Also: **JSP translator**, **servlet container**

container-managed persistence (CMP) bean

A container-managed persistence (CMP) bean is a **Java** bean that stores all state information in the **container**.

contention

Contention is a measure of competition for resources.

cookie

A cookie is a text string that is stored on the **client** browser by the **server** to maintain state between **HTTP** calls. Cookies enable applications to store and retrieve information about a client, such as the domain, path, lifetime, and other variables. Cookies can either expire when the user exits the browser or at a date specified by the creator of the cookie.

CORBA

Common Object Request Broker Architecture (CORBA) is an industry standard for allowing code modules called "**object**s" to communicate with one another regardless of the programming language in which they are written or the operating system on which they are running.

CORBA object

CORBA object is a generic term for a **server**-side program that conforms to the **OMG**'s **CORBA** specification. Objects can be written in any language, deployed on any machine, and can exist locally or over a wide-area network.

CPA

A collaboration protocol agreement (CPA) documents the technical agreement between two or more trading partners to engage in electronic business collaboration.

CPP

A collaboration protocol profile (CPP) defines one business partner's technical capabilities to engage in electronic business collaborations with other partners to exchange electronic messages.

CRL

A Certificate Revocation List (CRL) is a list of **certificate**s that have been revoked before their scheduled expiration date. CRLs only list revoked certificates. When a revoked certificate is past its original expiration date, it is removed from the CRL.

CRM

Customer relationship management (CRM) consists of the methodologies, software, and usually Internet capabilities that help an enterprise manage customer relationships in an organized way.

CSS

Cascading style sheets (CSS) provide a simple mechanism for adding style, such as fonts, colors, and spacing, to **XML** documents.

customer database

A customer database is the original and primary database for storing your data. The customer database is commonly located on the database **server** tier in a **three-tier architecture**.

customer relationship management

See CRM.

DAD

A database access descriptor (DAD) is a set of values that specify how an application connects to an Oracle database to fulfill an **HTTP** request. The information in the DAD includes the username (which also specifies the **schema** and the **privilege**s), password, connect-string, error log file, standard error message, and national language support (**NLS**) parameters such as NLS language, NLS date format, NLS date language, and NLS currency.

database access descriptor

See DAD.

default DAD

The default DAD is the database access descriptor that a **PL/SQL** application uses when the configuration information in the application does not specify a DAD.

default MIME type

The default MIME type is the **MIME type** the **HTTP listener** uses to interpret requested files of an unsupported MIME type.

See Also: **MIME type**

demilitarized zone

See DMZ.

deployment descriptor

A deployment descriptor (DD) is an **XML** text-based file with an `.xml` extension that describes the deployment settings for a component. A **J2EE** application and each of its modules has its own deployment descriptor (DD). For example, an enterprise bean module DD declares **transaction** attributes and security **authorization**s for an enterprise bean. DD information is declarative so it can be changed without modifying the bean source code. At runtime, the **J2EE Server** reads the DD and acts on the component accordingly.

digest authentication

Digest authentication is an **authentication** scheme that does not send passwords over the Internet. Digest authentication is safer than **basic authentication**, but is not supported by most browsers.

See Also: **authentication**, **basic authentication**

digital signature

A digital signature is a code attached to an electronic document that reliably identifies the author or sender, and verifies that the document has not been tampered with.

directory server

A directory server defines a hierarchical view of an organization's employees, units, and other resources.You can protect applications using directory servers by limiting access to the virtual paths of the applications to particular branches in the directory server.

distinguished name

A distinguished name is the unique name of an **LDAP**-based directory entry. A distinguished name comprises all of the individual names of the parent entries back to the root.

distributed transaction processing

See DTP.

DLL

A dynamic link library (DLL) is an archive of executable functions or data that can be used by a Microsoft Windows application. Typically, a DLL provides one or more particular functions and a program accesses the functions by creating either a static or dynamic link to the DLL. A DLL can be used by several applications at the same time.

DMZ

The demilitarized zone (DMZ) is the area between outer and inner **firewall**s. It is normally used to protect the internal application servers from being attacked by those attempting to gain unauthorized access to a network or intranet.

DNS

The domain name system (DNS) is the mechanism that divides the Internet into separate, hierarchical groups called domains, identified by unique alphanumeric names, such as `us.oracle.com`. DNS identifies each computer within a domain by a unique **hostname**. For example, a computer named hal in the `us.oracle.com` domain would be uniquely identified on the Internet as `hal.us.oracle.com`.

document type definition

See DTD.

domain-based restriction

Domain-based restriction is a **restriction** scheme that allows or denies access to files based on the **client** machine's domain name.

See Also: **restriction**, **IP-based restriction**

domain name system

See DNS.

DTD

A document type definition (DTD) is a set of rules that define the allowable structure of an **XML** document. DTDs are text files that derive their format from SGML and are either embedded within an XML document or referenced by an XML document.

DTP

Distributed transaction processing (DTP) is the protocol that guarantees a two-phase commit when multiple databases are involved in a **transaction**.

dynamic link library

See DLL.

e-business

E-business is the conduct of business over the Internet, such as buying and selling products, servicing customers, and collaborating with business partners.

EAI

Enterprise application integration (EAI) is the sharing and transferring of information and applications between systems so that they appear as a unified application.

EAR file

An Enterprise Archive (EAR) file is a standard JAR file with an `.ear` extension. A **J2EE** application with all of its modules is delivered in an EAR file. An EAR file contains the JAR and **WAR files** that comprise a J2EE application.

ebXML

Electronic Business Extensible Markup Language (ebXML) is a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business on the Internet. By using ebXML, companies have a standard

method for exchanging business messages, conducting trading relationships, and communicating data in common terms.

EDI

Electronic data interchange (EDI) is the computer-to-computer exchange of business data in standard formats. In EDI, information is organized according to a specified format that is set by both parties, allowing a computer transaction that requires no human intervention. All information contained in an EDI transaction set is, for the most part, the same as on a conventionally printed document. It defines the data formats and encoding rules that are required for a number of business transactions, including order placement and processing, shipping and receiving, invoicing, and payment systems.

EIS

Enterprise information systems (EIS) are systems that provide the information infrastructure for an enterprise. Enterprises run their businesses using the information stored in these systems. Examples of enterprise information systems include **enterprise resource planning** systems, mainframe **transaction** processing systems, relational database management systems, and other legacy information systems. Enterprise applications require access to applications running on enterprise information systems.

EJB

Enterprise JavaBeans (EJB) are the component-based application model for **Java** defined by JavaSoft. This model provides most of the system-level services, such as multi-threading, to ease application programming. EJB relies on various standardized enterprise services, such as **JNDI**, **JTS**, and **JDBC**, to facilitate application programming and enable EJB **object**s to be interoperable across various EJB servers. It fulfills the write once, run anywhere paradigm.

EJB application

An EJB application is a framework of deploying **CORBA object**s written in **Java**, which adhere to the **EJB** specification.

EJB container

An EJB container is the component coordinator in an **EJB application** and one of the key **EJB** runtime components.

See Also: **JNDI**, **EJB application**

EJB deployment descriptor

An EJB deployment descriptor is a serialized **object** that provides information, such as **transaction** and security policies, about how an **EJB application** or object should be deployed.

Electronic Business Extensible Markup Language

See ebXML.

electronic data interchange

See EDI.

encapsulation

Encapsulation is the mechanism that binds code together with the data that it manipulates. Encapsulation provides a wrapper that keeps both the code and the data safe from outside intervention.

encoding

Encoding is an algorithm used to alter a file's format, such as compression.

encryption

Encryption is the practice of **encoding** (encrypting) data in such a way that only an intended recipient can decode (decrypt) and read the data.

See Also: **public-key encryption**, **shared secret-key encryption**

enterprise application integration

See EAI.

enterprise beans

Enterprise beans are **server**-side components that encapsulate the business logic of an application, which is the code that fulfills the purpose of the application.

enterprise information systems

See EIS.

Enterprise JavaBeans

See EJB.

enterprise resource planning

See ERP.

entity bean

An entity bean is a complex business entity. An entity bean models a business entity or models multiple actions within a business process. Entity beans are often used to facilitate business services that involve data and computations on that data. For example, an application developer might implement an entity bean to retrieve and perform computation on items within a purchase order. Your entity bean can manage multiple, dependent, persistent **object**s in performing its necessary tasks.

ERP

The set of activities supported by multi-module application software that helps a manufacturer or other business manage important parts of their business, including product planning, parts purchasing, inventory management, supplier interaction, customer service, and order tracking.

extract, transform, and load (ETL) capabilities

The three database functions of extract, transform, and load are combined into one function and used to retrieve data from one database and write it to another.

- **Extract:** the retrieving of data from a data source
- **Transform:** the converting of the extracted data from it existing format into the format of the target database, using pre-defined rules
- **Load:** the writing of the transformed data into the target database

failover

Failover is the ability to reconfigure a computing system to utilize an alternate active component when a similar component fails.

failure recovery

Failure recovery is a system of failure detection and recovery. Components monitor each other continuously. When a component fails, Oracle Application Server detects the failure and restarts the failed component, restoring any preserved state information when possible.

farm

A farm is a collection of **cluster**s and **instance**s that share the same **Oracle Application Server Infrastructure**. A farm can be file based or database based. The repository for a file-based farm exists within the middle-tier instance Oracle Home, The repository for a database based farm exists within the Metadata Repository.

fault tolerance

Fault tolerance is the ability of a computing system to withstand errors while continuing to provide the required services.

file protection

File protection is the practice of assigning an **authentication** or **restriction** scheme by controlling access to a specific file or group of files.

filename extension

A filename extension is a short alphanumeric suffix attached to a filename following a dot "." that represents the file's format. Oracle Application Server uses filename extensions to identify several kinds of file formats, including **MIME type**s and **encoding**s.

firewall

A firewall is a machine that acts as an intermediary to protect a set of computers or networks from outside attack. It regulates access to computers on a local area network from outside, and regulates access to outside computers from within the local area network. A firewall can work either by acting as a **proxy server** that forwards requests so that the requests behave as though they were issued by the firewall machine, or by examining requests and attempting to eliminate suspect calls.

graphical user interface

See GUI.

GUI

A graphical user interface (GUI), sometimes referred to as the UI (user interface), is the graphical, as opposed to purely textual, user interface to a computer.Elements of the GUI include pull-down menus, buttons, icons, windows, and graphics.

Health Level Seven

See HL7.

HL7

Health Level Seven (HL7) is a standard for electronic data exchange in health care environments. It focuses on health care in the clinical and administrative data domain. "Level Seven" refers to the highest level of the International Standard Organization's

(ISO) communications model for Open Systems Interconnection (OSI), the application level.

host

A host is a computer with a unique domain name.

hosted applications

Hosted applications are typically applications developed by independent software vendors and administered by application service providers. These applications are accessed on an external Web site that enables multiple companies to utilize the applications.

hosted environment

A hosted environment is an application deployment environment in which multiple customers and companies subscribe to shared services.

hosted services

Hosted services are services are offered by application service providers for multiple **subscriber**s.

hostname

A hostname is a character string that uniquely identifies a computer within a **DNS** domain.

hot deployment

Hot deployment is a method for deploying new applications that does not require restarting the application server. With hot deployment, changes made to the **server** configuration files are automatically detected, and you can deploy new applications without restarting the server.

See Also: **automatic deployment** for comparison

HTML

Hypertext markup language (HTML) is a format for **encoding** hypertext documents that may contain text, graphics, and references to programs and other hypertext documents.

HTTP

Hypertext Transfer Protocol (HTTP) is the underlying format used by the Web to format and transmit **message**s and determine what actions Web **server**s and browsers should take in response to various commands. HTTP is the protocol used between Oracle Application Server and **client**s.

HTTP header

An HTTP header is a body of information that a browser sends along with a **URL** when requesting a Web page. It includes such information as the browser type and **MIME type**s.

HTTP listener

See listener.

HTTP request information

HTTP request information is the information requested by a **client** in the form of an **HTTP header**.

HTTP response information

HTTP response information is the information supplied by an **HTTP listener** or an application.

HTTP server

An HTTP server is a **server** that receives HTTP requests from remote browsers, converts the requested **URL** to a filename, and returns the file to the requester.

HTTPS (secure HTTP)

HTTPS is a version of **HTTP** with provisions for secure data transmission.

See Also: **HTTP**

hypertext markup language

See HTML.

hypertext transfer protocol

See HTTP.

IBAC

Identity-based access control (IBAC) is the use of digital IDs to control access to a resource.

IDE

An integrated development environment is a visual tool containing editors, debuggers, screen painters, **object** browsers etc.

identity-based access control

See IBAC.

IDL

Interface definition language (IDL) is a standard language for interface specification primarily used for **CORBA object** interface definition. IDL is declarative and does not reveal the implementation of a CORBA object. **CORBA** defines standard mappings from IDL to various programming languages.

IIOP

Internet inter-ORB protocol (IIOP) is an Internet transport protocol used by **CORBA object**s to communicate with each other. In the context of Oracle Application Server, IIOP is used by ECO / **Java** and **EJB object**s. IIOP is also used between Oracle Application Server components.

inheritance

Inheritance is the process by which one class acquires the methods and properties of another in **object-oriented programming** methodology.

instance

An application server instance is the set of processes required to run the configured components within an application server installation. There can be only one

application server instance per application server installation. The terms *installation* and *instance* are sometimes used interchangeably; however, it is important to remember that an installation is the set of files installed into an Oracle home and an instance is a set of processes associated with those files.

integrated development environment

See IDE.

interface definition language

See IDL.

Internet Information Server (IIS)

Internet Information Server (IIS) is Microsoft's Web **server** that runs on Windows NT/2000 platforms.

Internet inter-ORB protocol

See IIOP.

interoperable object reference

See IOR.

IOR

Interoperable object reference (IOR) is a unique string for each **CORBA object** and is created when an **object reference** is passed among different **ORB**s. You can pass this string to a method to determine the actual object reference.

See Also: **object reference**

IP address

An IP address is a four-part number separated by periods that uniquely identifies a computer on the Internet; the number format is defined by the Internet Protocol (IP).

IP-based restriction

IP-based restriction is a **restriction** scheme that allows or denies access to files based on the **client** machine's **IP address**.

See Also: **restriction**

island

An island is a logical grouping of **OC4J** processes that allows you to determine which OC4J processes will replication state.

J2EE

Java 2 Platform, Enterprise Edition (J2EE) is a platform that enables application developers to develop, deploy, and manage multi-tier, **server**-centric, enterprise level applications.

J2EE Server

The J2EE Server is the runtime portion of a **J2EE** product, which provides **EJB container**s, Web containers, or both. J2EE servers are usually located in the middle tier of a **three-tier architecture**.

J2SE

Java 2 Platform, Standard Edition (J2SE) is a platform that enables application developers to develop, deploy, and manage **Java applet**s and applications on a desktop **client** platform such as a personal computer or workstation.

JAAS

Java Authentication and Authorization Service (JAAS) is a **Java package** that enables services to authenticate and enforce access control upon users. JAAS implements a Java version of the standard Pluggable Authentication Module (PAM) framework, and extends the access control architecture of the **Java 2 Security Model** to support user-based **authorization**.

JAR files

There are different types of Java Archive (JAR) files. An **EJB** JAR file contains its **deployment descriptor** (DD), related files, and the `.class` files for the enterprise bean. An application **client** JAR file contains its DD, related files, and the `.class` files for the application client. JAR files are packaged together with **WAR files** into **EAR file**s.

Java

Java is a programming language developed by Sun Microsystems. This language is fully **object**-oriented, extremely portable, and optimized for creating distributed applications on the Internet or other computer networks.

Java 2 Platform, Enterprise Edition

See J2EE.

Java 2 Platform, Standard Edition

See J2SE.

Java 2 Security Model

The Java 2 Security Model provides developers and administrators with increased control over many aspects of enterprise **applet**, component, **servlet**, and application security. The Java 2 Security Model is capability-based and enables you to establish **protection domain**s, and set security policies for these domains.

Permissions are the basis of the Java 2 Security Model. All **Java** classes (whether run locally or downloaded remotely) are subject to a configured security **policy** that defines the set of permissions available for those classes. Each permission represents a specific access to a particular resource.

Java Authentication and Authorization Service

See JAAS.

JavaBeans

JavaBeans is a portable, platform-independent component model that enables developers to write reusable components once and run them anywhere.

See Also: **Enterprise JavaBeans**

Java Connector Architecture

See JCA.

Javadoc

Javadoc is a tool for generating **API** documentation in **HTML** format from documentation comments in **Java** source code. These HTML pages describe the classes, inner classes, interfaces, constructors, methods, and fields.

Java interpreter

A Java interpreter is a program that interprets and executes **Java** bytecode independently of a Web browser.

Java Message Service

See JMS.

Java Naming and Directory Interface

See JNDI.

JavaServer Pages

See JSP.

Java Transaction API

See JTA.

Java Transaction Service

See JTS.

Java Virtual Machine

See JVM.

JDBC

Java DataBase Connectivity (JDBC) is a **Java package** that provides connectivity to databases from within Java.

JCA

The Java Connector Architecture (JCA) provides a standard architecture for integrating heterogeneous Enterprise Information Systems (**EIS**).

JIT compilation

Just-in-time (JIT) compilation is the process by which the Java Virtual Machine (**JVM**) keeps a copy of native code that it generates from bytecode the first time a method is encountered. Subsequently, when the method is run, the JIT uses the native code without having to interpret the method, resulting in a boost in performance.

JMS

The Java Message Service (JMS) API provides a reliable, flexible service for the asynchronous exchange of critical business data and events throughout an enterprise.

JNDI

Java Naming and Directory Interface (JNDI) consists of a standard set of **API**s that provide directory and naming services. Oracle Application Server has a JNDI naming server that **client**s can use to obtain **object reference**s to ECO / **Java object**s or Enterprise JavaBean objects.

JSP

JavaServer Pages (JSPs) are an extension to the **servlet** functionality that enables a simple programmatic interface to Web pages. JSPs are **HTML** pages with special tags and embedded **Java** code that is executed on the Web or application server, providing dynamic functionality to HTML pages. JSPs are actually compiled into servlets when first requested and run in the **servlet container**.

JSP engine

See JSP translator.

JSP tag

JSP tags are tags that are used in JavaServer Pages (**JSP**s). These tags use the `<jsp:` syntax and enclose action elements in the JSP with "begin" and "end" tags similar to **XML** statements.

JSP translator

A JSP translator is an entity that translates, executes, and processes **JSP** pages and delivers requests to them. The exact architecture of a JSP translator varies from implementation to implementation, but it consists of a **servlet** or a collection of servlets. The JSP translator, therefore, is executed by a **servlet container**.

JTA

Java Transaction API (JTA) enables your applications to participate in distributed transactions and to access transaction services from other components.

JTS

Java Transaction Service (JTS) provides the services necessary for applications and databases to become part of a **transaction**. JTS is the **Java** version of **OTS**.

just-in-time compilation

See JIT compilation.

JVM

A Java Virtual Machine (JVM) is an abstract specification for a computing device that can be implemented in different ways, in software or hardware. You compile to the instruction set of a **virtual machine** much like you would compile to the instruction set of a microprocessor.

The Java Virtual Machine is part of the **Java** runtime environment responsible for interpreting Java bytecode. It consists of a bytecode instruction set, a set of registers, a stack, a garbage-collected heap, and an area for storing methods. Java bytecode is executable by any JVM running on any machine.

key pair

A key pair is a pair of mathematically related keys (a **public key** and a **private key**) associated with a user and used in **public-key encryption**.

language identifier

A language identifier is a two-character alphanumeric string that identifies a human language, as defined by **RFC** 1766.

LDAP

Lightweight Directory Access Protocol (LDAP) is a protocol that allows **client**s to access information from a **directory server**. This protocol enables corporate directory

entries to be arranged in a hierarchical structure that reflects geographic and organizational boundaries.

light weight directory access protocol

See LDAP.

listener

A listener is an **HTTP server** that handles incoming requests and routes them to the dispatcher.

Login Module

A Login Module is the **authentication** module configured for a particular application. The `LoginContext` class decouples the application code from the authentication services, and different login modules can be plugged in under an application without affecting the application code.

Login Server

The Login Server authenticates the username and password of a **client** user attempting to access an application. Once authenticated, the Login Server passes the client's identity to various applications. An encrypted login **cookie** identifies the client as being authenticated. The Login Server provides single sign-on **authentication**. This enables a user to access multiple accounts and applications with a single username and password.

manifest file

A manifest file is a text file that describes the contents of a JAR file. For **EJB** JAR files, the manifest file identifies it as an "ejb-jar" file. The manifest file also indicates which elements in the JAR file are the EJB **deployment descriptor**s for the components to be deployed.

message

A message is the smallest unit of information inserted into and retrieved from a queue. A message consists of the following:

- Control information (metadata)
- Payload (data)

The control information represents message properties used by the queue to manage messages. The payload data is the information stored in the queue. A message can reside in only one queue.

message-driven beans

Message-driven beans (MDB) provide an easier method for implementing asynchronous communication than using straight JMS. MDBs were created to receive asynchronous JMS **message**s. The **container** handles much of the setup required for JMS queues and topics. It sends all messages to the interested MDB.

metric

A metric is a performance statistic, such as uptime or queue.

MIME type

The Multipurpose Internet Mail Extension (MIME) type is a **message** format used on the Internet to describe the contents of a message and defined by the MIME standard.

MIME is used by **HTTP server**s to describe the type of content being delivered. Several **RFC**s define MIME.

See Also:

`http://www.mhonarc.org/~ehood/MIME/MIME.html`

mods

See Apache module.

mod_access

The `mod_access` **module**, or plug-in, of Oracle HTTP Server provides access control based on **client** host name or **IP address**.

mod_osso

The `mod_osso` **module** provides communication between the Single Sign-On enabled **Login Server** and the Oracle HTTP Server **listener**.

module

See Apache module.

multiport

A multiport is a single **listener** that can respond to requests directed at more than one address/**port** combination. A multiport can be configured to respond to requests differently if they are made to a different address/port combination on the same listener.

Multipurpose Internet Mail Extensions

See MIME type.

national language support

See NLS.

NLS

National language support (NLS) is the set of mechanisms used to translate data between various languages and **character set**s.

node

See host.

noun

In the scope of performance monitoring tools, a noun is a component of an Oracle Application Server site. Examples of a noun are component, **object**, class, bean, process, piece of code, computer, and process.

OAGI

The Open Applications Group, Inc. (OAGI) is a nonprofit consortium focusing on the best practices and processes that are based on XML content for e-business and application integration. OAGI was set up to create common standards for the integration of enterprise business applications. Member companies are building specifications to standardize integration between enterprise business applications. The OAGI Business Object Document (BOD) is the architecture that is used to communication message or business documents between software applications or

components. Each BOD includes supporting details to enable the destination business application to accomplish the action.

object

An object is any item that can be individually selected or manipulated. This includes buttons and pull-down menus. In **object-oriented programming**, an object is a self-contained entity that consists of both data and procedures to manipulate the data.

Object Management Group

See OMG.

object-oriented programming

Object-oriented programming is a method of programming that organizes a program around its data (**object**s) and a set of well-defined interfaces to that data. It is a revolutionary new way of looking at computer programming. A generalization of the data object along with its possible data variables and methods (what to do with variables) in a class of data objects.

object reference

An object reference is a unique identifier that is used to represent an **object** instance in a distributed system.

See Also: IOR

Object Transaction Service

See OTS.

OC4J

See Oracle Application Server Containers for J2EE (OC4J).

OCI

Oracle Call Interface (OCI) is an **API** or low-level tool for accessing Oracle databases and executing **SQL** and **PL/SQL** statements.

OLTP

Online **transaction** processing (OLTP) refers to the immediate processing of a transaction. The opposite of transaction processing is batch processing, where transactions are stored (batched) and then all of the stored transactions are executed at one time.

OMG

Object Management Group (OMG) is a consortium with a membership of more than 700 companies. The organization's goal is to provide a common framework for developing applications using **object-oriented programming** techniques. OMG is responsible for the **CORBA** specification.

online transaction processing

See OLTP.

Open Applications Group, Inc.

See OAGI.

ORACLE_HOME

ORACLE_HOME is an environment variable (UNIX) or registry key (Windows) that indicates the directory where Oracle software is installed.

Oracle platform

The Oracle platform is a **server** deployment platform consisting of Oracle Application Server, Oracle Database Server, and Oracle Developer Suite.

Oracle Application Server Cold Failover Cluster

Oracle Application Server Cold Failover Clusters are a high availability solution where the Oracle Application Server Infrastructure is typically deployed on a two-node hardware cluster with a shared storage device. Once node is active or "hot," meaning it is running the Infrastructure, while the other node is "cold" and is not running the Infrastructure. When the active node fails, the clusterware switches Infrastructure operations to the previously "cold" node and the Infrastructure is started on that node.

Oracle Application Server Containers for J2EE (OC4J)

Oracle Application Server Containers for J2EE (OC4J) is a complete set of **J2EE containers** written entirely in **Java** that execute on the Java Virtual Machine (**JVM**) of the standard Java Development Kit (JDK).

Oracle Business Intelligence Discoverer

Oracle Application Server Discoverer is a **business intelligence** tool for analyzing data. Using Oracle Application Server Discoverer's award-winning user interface, users can access and analyze database data. There are two Oracle Application Server Discoverer products:

- Oracle Application Server Discoverer Plus is the Internet version of the award-winning Windows version of Discoverer. With Discoverer Plus, business professionals can get and analyze data in a company's database without having to understand complex database concepts. Using Wizard dialogs and menus, Discoverer Plus guides users through the steps to get and analyze data to support their business decisions.
- Oracle Application Server Discoverer Viewer is a tool for viewing workbooks created by Discoverer Plus. Oracle Application Server Discoverer Viewer can also be used to integrate database output into a Web site and **portal**. It is easy to customize Discoverer Viewer both to conform to a particular Web site look and feel and to build custom Discoverer applications for the Web. Discoverer Viewer is optimized for performance and designed to minimize network traffic.

Oracle Application Server Infrastructure

Oracle Application Server Infrastructure is a single, consistent combination of the **Oracle Application Server Metadata Repository**, a **directory server**, management server, and **single sign-on** server. Most Oracle Application Server components use the infrastructure to provide a more integrated environment for application developers and system managers.

Oracle Application Server InterConnect

Oracle Application Server InterConnect is the integration hub that coordinates the communication and transformation of **message**s between two or more heterogeneous applications. Oracle Application Server InterConnect defines business events, their associated data, and any transformations required to map one application's view of a business **object** to another's view.

Oracle Application Server InterConnect Adapters

Oracle Application Server InterConnect Adapters are a set of connectivity adapters that enable third-party applications and technology environments to participate in integration.

Oracle Application Server Metadata Repository

Oracle Application Server Metadata Repository is a pre-seeded database containing metadata required by Oracle Application Server **instance**s.

Oracle Application Server Portal

Oracle Application Server Portal is a complete solution for building, deploying and monitoring Web database applications and content-driven Web sites. Oracle Application Server Portal enables you to create and view database **object**s through an easy-to-use **HTML**-based interface, and provides tools for creating HTML-based interfaces. It also allows you to resolve performance problems using performance tracking facilities, and enables you to manage database security through its interface.

See Also: **portal**, **portlet**

Oracle Application Server Portal Login Server

See Login Server.

Oracle Application Server Web Cache

Oracle Application Server Web Cache is a **server** accelerator **caching** service that improves the performance, **scalability**, and **availability** of frequently used **e-business** Web sites that run on the **Oracle platform**. By storing frequently accessed **URL**s in virtual memory, Oracle Application Server Web Cache eliminates the need to repeatedly process requests for those URLs on the Web server, and it caches both static and dynamically-generated **HTTP** content from one or more applications Web servers.

Oracle Application Server Wireless

Oracle Application Server Wireless is a **portal** service for delivering information and applications to mobile devices. Using Oracle Application Server Wireless, you can create custom portal sites that use different kinds of content, including Web pages, custom **Java** applications, and **XML**-based applications. Oracle Application Server Wireless sites make this diverse information accessible to mobile devices without you having to rewrite the content for each target device platform.

Oracle Advanced Queuing

Oracle Advanced Queuing provides database-resident messaging capabilities that enable reliable, asynchronous communication between applications. Oracle Application Server integration uses Oracle Advanced Queuing as the communication mechanism between adapters, participating applications, and **Oracle Workflow**.

Oracle Advanced Security

Oracle Advanced Security provides a comprehensive suite of security features to protect enterprise networks and securely extend corporate networks to the Internet. It provides a single source of integration with network **encryption** and **authentication** solutions, **single sign-on** services, and security protocols. By integrating industry standards, it delivers unparalleled security to the Oracle network and beyond.

Oracle Call Interface

See OCI.

Oracle Database Client Developer Kit

The Oracle Database Client Developer Kit contains the following **client** libraries:

- Oracle Java Database Connectivity (**JDBC**) Drivers
- Oracle Java Messaging Service (JMS) Toolkit
- Oracle SQLJ Translator

Oracle HTTP Server

Oracle HTTP Server is the Web **server** that Oracle Application Server uses, which is built on **Apache** Web server technology. Oracle HTTP Server offers **scalability**, stability, speed, and extensibility. It also supports Java **servlet**s, JavaServer Pages (**JSP**s), Perl, **PL/SQL**, and **CGI** applications.

Oracle Internet Directory

Oracle Internet Directory is a general purpose directory service that enables retrieval of information about dispersed users and network resources. Oracle Internet Directory combines **LDAP** version 3 with the high performance, **scalability**, robustness, and **availability** of the Oracle database.

Oracle Internet Directory runs as an application in the Oracle database. It communicates with the database, which may be on the same or a different operating system.

Oracle Content Management SDK

Oracle Content Management SDK (CMSDK) is a file system that stores its contents in an Oracle database. The contents can range from files to metadata that defines the file system (including foldering, security, and users). Oracle CMSDK indexes textual content in files as they are added or updated. This enables you to search the contents of more than 150 file formats.

Oracle LDAP Developer Kit

Oracle LDAP Developer Kit supports **client** interaction with any **LDAP**-compliant **directory server**; for example, **Oracle Internet Directory**. LDAP (Lightweight Directory Access Protocol) is the emerging Internet standard for directory services. The toolkit provides tools and development libraries to support client calls to directory services, encrypted connections, and enables you to manage your directory data.

Oracle Wallet Manager

Oracle Wallet Manager is a **Java**-based application that security administrators use to manage public-key security credentials on **client**s and **server**s. Security credentials consist of a public/private **key pair**, a **certificate**, and a **trustpoint**.

See Also: **public-key cryptography**, **wallet**

Oracle Workflow

Oracle Workflow lets you automate and continuously improve business processes, **routing** information of any type according to easily-changeable business rules to users both inside and outside a company's enterprise.

Oracle XML Developer Kit

Oracle XML Developer Kit (XDK) contains the necessary **XML** component libraries and utilities to give developers the ability to easily XML-enable applications and Web sites. Oracle XDK supports development in **Java**, C, C++, and **PL/SQL** with a collection of libraries, command-line utilities, and tools.

ORB

An Object Request Broker (ORB), in **CORBA**, acts as a "broker" between a **client** request for a service from a distributed **object** or component and completion of that request. Having ORB support in a network means that a client program can request a service without having to understand where the **server** is in the distributed network or in which language the server object is written.

OTS

Object Transaction Service (OTS) provides the services necessary for applications and databases to become part of a **transaction**.

See Also: **JTS**

overloading

Overloading is the ability to use the same name for more than one variable or procedure, requiring the compiler to differentiate them based on context. Overloaded procedures and functions (in **PL/SQL**) or methods (in **Java**) have the same name but take different parameters and do similar but not identical things.

package

A package is a group of **PL/SQL** or **Java** functions and procedures.

partner interface processes

See PIP.

PIP

Partner interface processes (PIPs) are specialized XML-based dialogs that determine how two trading partners must interact to conduct a business collaboration. Some examples are: a buyer submitting a purchase order to a seller, a buyer notifying a seller of a change in a purchase order, or a customer submitting a forecast to a supplier. PIPs are grouped into clusters and these clusters are further grouped into segments. These segments are divided into business functions. The PEP name is coded to reflect the cluster, segment, and business functions of the PIP.

PKCS

The Public-Key Cryptography Standards (PKCS) are specifications produced by RSA Laboratories in cooperation with secure systems developers worldwide for the purpose of accelerating the development of **public-key cryptography**.

PL/SQL

PL/SQL is Oracle's proprietary extension to the **SQL** language. PL/SQL adds procedural and other constructs to SQL that make it suitable for writing applications.

policy

A policy is an **authorization** rules repository for **JAAS**. A policy contains the rules with which a user is authorized to use resources, such as a database. The primary function of a policy implementation is to provide the following information:

Given a grantee, return the granted permissions for the grantee.

policy file

A policy file grants permissions to the trusted codes or applications that you run. These codes or applications are not allowed to access resources unless they are

explicitly granted permission to do so by the security **policy** in effect. In **Java** environments, the permission must be granted by an entry in a policy file.

policy store

A policy store provides secure, centralized storage, retrieval, and administration of **JAAS** policies (the **role**s and **privilege**s of a user). Oracle Internet Directory is the policy store used for storing this information.

polymorphism

In **object-oriented programming**, polymorphism (from the Greek meaning "having multiple forms") is the characteristic of being able to assign a different meaning to a particular symbol or “operator” in different contexts.

port

A port is a number that TCP uses to route transmitted data to and from a particular program.

portal

Portal sites give users a single, centralized, personalized view of relevant applications and data.

See Also: **Oracle Application Server Portal**, **portlet**

portlet

A portlet is a reusable information component that summarizes or provides access to an information source. Portlets are the fundament building blocks of an **Oracle Application Server Portal** page.

See Also: **portal**

principal

A principal is a specific authenticated identity, such as a user named `frank`. A principal is essentially an identity associated with a **subject**. A principal can be associated with a subject (such as a user) upon successful **authentication** to a computing service.

private key

A private key is a key used by a limited number of communicating parties to decrypt data encrypted with a **public key**.

See Also: **public-key encryption**

privilege

A privilege is the right to perform an action on the database. Privileges can either be general (system privileges) or specific to particular database **object**s (object privileges). They can also be grouped into **role**s.

See Also: **role**

processor sets

Processor sets, which are specific to Solaris systems and were introduced in Solaris 2.6, allow a group of processors to be allocated for the exclusive use of one or more

applications. This gives a system administrator control over the creation, management, and binding of processes into processor sets.

protection domain

A protection domain is a fundamental building block of system security. The protection domain concept serves as a convenient mechanism for grouping and isolation between units of protection. For example, it is possible to separate protection domains from interacting with each other so that any permitted interaction must be either through trusted system code or explicitly allowed by the domains concerned.

proxy server

A proxy server typically sits on a network **firewall** and allows **client**s behind the firewall to access Web resources. All requests from clients go to the proxy server rather than directly to the destination **server**. The proxy server forwards the request to the destination server and passes the received information back to the client. The proxy server channels all Web traffic at a site through a single, secure **port**; this allows an organization to create a secure firewall by preventing Internet access to internal machines, while allowing Web access.

public key

A public key is a key known to all users, used to encrypt data in such a way that only a specific user can decrypt it.

See Also: **private key**, **public-key encryption**

public-key cryptography

Public-key cryptography is a set of well-established techniques and standards that ensure that data passes securely over a network, from the sender to the receiver, without any interference from a third party. Public-key cryptography facilitates **encryption** and decryption, tamper detection, **authentication**, and nonrepudiation.

Public-Key Cryptography Standards

See PKCS.

public-key encryption

Public-key encryption is a form of **encryption** that uses a **key pair** (a **public key** and a **private key**) to encrypt and decrypt data.

query string

A query string is the optional portion of a **URL** that specifies parameters to be passed to an application.

RAC

Real Application Clusters (RAC) is a parallel database clustering technology from Oracle. RAC is an active-active cluster with shared storage, whereby multiple servers can work in parallel on the same set of data.

Real Application Clusters

See RAC.

realm

A realm provides access to a **policy store** of users and **role**s (groups) and optionally provides administrative functionality. Oracle provides a proprietary Realm **API** that

implements the notion of a user community. Essentially a user community instance is a realm that is maintained internally by the **authorization** system.

RC4, RC5

RC4 and RC5 are **RSA encryption** algorithms.

redundant

Redundant components are duplicate or extra computing components that protect a computing system.

reliability

Reliability is the ability of a computing system to operate without failing. Reliability is measured by mean-time-between-failures (MTBF).

remote method invocation

See RMI.

remote method invocation

See RPC.

request latency

Request latency is the time required to process a request.

request throughput

Request throughput is the number of requests processed per unit of time.

response time

Response time is the amount of time between the submission of a request and the receipt of the response.

restriction

Restriction is a **security scheme** that restricts access to files provided by the **server** to **client** machines within certain groups of **IP address**es or **DNS** domains.

See Also: **domain-based restriction**, **IP-based restriction**

reverse mapper

A reverse mapper is a utility that converts a **CORBA** server **object**'s interfaces to **IDL** (for example, **Java** to IDL). These IDL interfaces can then be converted to CORBA stub and skeleton code by an IDL compiler (for example, IDL to Java) for use by **client**s.

RFC

Request for Comments (RFC) is a collection of Internet-related articles.

See Also: `http://www.rfc-editor.org`

RMI

Remote method invocation (RMI) is an interaction scheme for distributed **object**s written in **Java**. It enables a Java program running on one computer to access the methods of another Java program running on another computer.

RNIF

The RosettaNet Implementation Framework (RNIF) defines implementation guidelines for creating software applications that provide for the secure and reliable transport of **PIP**s in **XML**-formatted business documents between trading partners.

role

A role is a group of database **privilege**s that can be granted and revoked as a unit. Specific privileges can be granted and revoked from the role dynamically, and the role can be enabled or disabled dynamically for specific users.

role activation

Role activation is the process by which a user can selectively enable the required **role**s to accomplish a specific task in a user session. A user is granted multiple roles, but not all roles are enabled by default. Enabling only those roles the user needs, the user ensures the principle of least **privilege**. That is, the user is not enabling permissions or privileges unnecessary for the task. This limits the damage that can result from an accident or error.

role-based access control

Role-based access control is a process that simplifies the administrative process of **authorization** management based on direct assignment of permissions to users. Instead of directly assigning permissions to users, permissions are assigned to **role**s, and users are granted their permissions by being made members of appropriate roles. Multiple roles can be granted to a user, and a role can also be granted other roles, thus forming a role hierarchy that provides security administrators with a tool to model enterprise security policies.

RosettaNet

RosettaNet is a business-to-business (**B2B**) protocol standard with the following components: **PIP**s, **RNIF**, and **RosettaNet Dictionaries**.

RosettaNet Dictionaries

RosettaNet dictionaries define and describe a common set of properties for use in **PIP** messages. These properties define a common vocabulary for conduction business so that the information that is exchanged between trading partners is consistent.

RosettaNet Implementation Framework

See RNIF.

routing

Routing is the process of directing data from one machine on the Internet to another by way of intermediate machines.

RPC

Remote procedure call (RPC) is a type of protocol that allows a program on one computer to execute a program on a **server**. Using RPC, a system developer need not develop specific procedures for the server. The client program sends a message to the server with appropriate arguments and the server returns a message containing the results of the program executed.

RSA

RSA is an Oracle partner supplying **encryption** algorithms for the **HTTP server** and Web Request Broker.

scalability

Scalability is the ability to handle increasing numbers of hardware requests without adversely affecting latency and throughput.

schema

A schema is a collection of database **object**s, including logical structures such as tables, views, sequences, **stored procedure**s, synonyms, indexes, **cluster**s, and database links.

A schema has the name of the user who controls it.

secure sockets layer

See SSL.

security scheme

A security scheme prevents unauthorized users from accessing protected files or applications. The **Oracle platform** supports both **authentication** schemes and **restriction** schemes.

The Oracle platform supports the following types of **authentication**:

- **basic authentication**
- **digest authentication**
- **certificate authentication**

The following **restriction** schemes are supported:

- **IP-based restriction**
- **domain-based restriction**

serializable

An interface that a **Java** class implements to enable itself to be persistent through the Java Serialization service. This service allows a bean's state to be stored and retrieved.

server

There are two types of servers relevant to this product. The first is Oracle Database Server, which is a relational database server dedicated to performing data management duties on behalf of **client**s using any number of possible interfaces. The other is Oracle Application Server, which is a collection of middleware services and tools that provide a scalable, robust, secure, and extensible platform for distributed, **object**-oriented applications. Oracle Application Server supports access to applications from both Web clients (browsers) using **HTTP** and Common Object Request Broker Architecture (**CORBA**) clients, which use the CORBA and the Internet Inter-ORB (**IIOP**) protocols.

service time

Service time is the time between the receipt of a request and the completion of the response to the request.

servlet

A servlet is a small **Java** program that runs on a Web **server**, as opposed to an **applet**, which is a Java program that runs in a **client** browser. Servlets take client **HTTP** requests from a browser, generate dynamic content (such as through querying a database), and provide an HTTP response back to the browser.

servlet container

A servlet container is the runtime environment for a Java **servlet**. The servlet container calls the methods of a servlet and provides services to the servlet while it is executing.

servlet zones

Servlet zones are analogous to virtual directories in a Web **server**. Each zone can have its own set of **servlet**s and can use a dedicated Java Virtual Machine (**JVM**). All of the servlets in one servlet zone share the same servlet repository, which is a location where **Java** class files are loaded.

session bean

A session bean implements one or more business tasks. A session bean might contain methods that query and update data in a relational table. Session beans are often used to implement services. For example, an application developer might implement one or several session beans that retrieve and update inventory data in a database.

session key

A session key is a secret key used by **SSL** to encrypt data transmitted over a secure connection. The **client** generates the session key after Oracle Application Server authenticates itself and sends it to Oracle Application Server using **public-key encryption**.

shadow process

A shadow process is a simple process on every **host** that detects if an Oracle HTTP Server **listener** process on that host is alive. If the listener process dies, then the shadow process brings a new instance up to handle requests.

shared secret-key encryption

Shared secret-key encryption is a form of **encryption** that uses a single key both to encrypt and to decrypt a document. Shared secret-key encryption is much faster than **public-key encryption**, but is more vulnerable to attack.

short message service

See SMS.

Simple Network Management Protocol

See SNMP.

Simple Object Access Protocol

See SOAP.

single sign-on

Single sign-on is the ability for a user to authenticate once, combined with strong **authentication** occurring transparently in subsequent connections to other databases or applications. Single sign-on lets a user access multiple accounts and applications with a single password.

skeletons

In **CORBA**, skeletons are classes that are generated as part of the **IDL** compilation process. They receive calls from **client** stubs, unmarshal parameters, and call the method implementation.

See Also: **IDL**, **skeletons**

SMS

Short message service (SMS) is the transmission of short text messages (under 160 alpha-numeric characters and no images) to a mobile device.

SNMP

Simple Network Management Protocol (SNMP) is a set of protocols for managing complex networks. The versions of SNMP were developed in the early 1980s. SNMP works by sending **message**s, called protocol data units (PDUs), to different parts of a network. SNMP-compliant devices, called agents, store data about themselves in Management Information Bases (MIBs) and return this data to the SNMP requesters.

SOAP

Simple Object Access Protocol (SOAP) is a lightweight, **XML**-based protocol for exchanging information in a decentralized, distributed environment. SOAP supports different styles of information exchange, including Remote Procedure Call style (RPC) and Message-oriented exchange. RPC style information exchange allows for request-response processing, where an endpoint receives a procedure-oriented **message** and replies with a correlated response message. Message-oriented information exchange supports organizations and applications that need to exchange business or other types of documents where a message is sent but the sender may not expect or wait for an immediate response.

SQL

Structured query language (SQL) is the internationally accepted standard for relational systems, covering not only query but also data definition, manipulation, security, and some aspects of referential integrity.

See Also: **PL/SQL**

SSL

Secure Sockets Layer (SSL) is a standard for the secure transmission of documents over the Internet using **HTTPS (secure HTTP)**. SSL uses **digital signature**s to ensure that transmitted data is not tampered with.

stateful application

A stateful application is an application that maintains its own state information.

stateful session bean

A stateful session bean is a bean in which instance variables represent the state of a unique **client**-bean session. Because the client interacts with its bean, this state is often called the conversational state. This conversation state is retained for the duration of the client-bean session. If the client removes the bean or terminates, the session ends and the state disappears.

stateless application

A stateless application is an application that does not store state information. Requests to a stateless application must include all information that the application needs to run properly.

stateless session bean

A stateless session bean is a bean which does not maintain a conversational state for a particular **client**. When a client invokes the method of a stateless bean, the bean instance variables may contain a state, but only for the duration of the invocation.

When the method is finished, the state is no longer retained. Except during method invocation, all instances of a stateless bean are equivalent, allowing the **EJB container** to assign an instance to any client. Because stateless session beans can support multiple clients, they can offer better **scalability** for applications that require large numbers of clients.

stored procedure

A stored procedure is a set of **PL/SQL** instructions that are stored in a database.

structured query language

See SQL.

stub

In **CORBA**, a stub is a class that is generated as part of the **IDL** compilation process (e.g. using an IDL-to-**Java** compiler). The compiler creates the stub from the IDL interfaces of a server **object**. The stub contains code that handles operations needed to access remote objects. A **client** application uses the generated stubs to access methods in the remote server object.

See Also: **IDL** and **skeletons**

subagent

A subagent is a process that receives queries for a particular managed element from the master agent and sends back the appropriate answers to the master agent. The **SNMP** subagent detects unexpected events and forwards them to the master agent.

subject

A subject refers to any user of a computing service. Since users, **role**s, and computing services use computing services, all are subjects. A subject is the source of a service request. To identify the subjects with which it interacts, a computing service typically relies on names. Subjects can have different names for each individual service with which they interact. These individual names are known as **principal**s. For example, a user can have a name principal (user `frank`), a role principal (role `hr`), and a social security number principal (`123-45-6789`).

subscriber

A subscriber is customer (such as a company) that uses (subscribes to) an application deployed in a **hosted environment**.

subscription

In the scope of the performance monitoring tools, a subscription is a request by a user to track a particular **metric**.

SWIFT

SWIFT is the industry-owned cooperative that supplies a standard for secure messaging services and interface software to financial institutions throughout the world. SWIFT focuses on messaging services to banks, brokers or dealers, investment managers, as well as to market infrastructures in payments, treasury, securities, and trade.

three-tier architecture

Three-tier architecture is a **client-server** computing architecture where systems are separated into three layers: the interface layer, the application logic layer, and the database layer. Clients in this model are thin clients, which do not contain as many

resources as clients in the **two-tier architecture**. The application logic layer, which is also known as the middle-tier, provides the computing power and resources for the client. Oracle Application Server fulfills this role ideally as a development and deployment platform for database-enabled applications.

throughput

Throughput is the number of requests processed per unit of time.

TNS string

A Transparent Network Substrate (TNS) string is a foundation technology built into Net8, Oracle Connection Manager and Oracle Names, that works with any standard network transport protocol. The TNS string contains destination service and network route information. Users initiate a connect request by passing a user name and password along with a connect identifier in a TNS String for the service to which they want to connect.

topology management request

A topology management request is an **HTTP** request that is handled by the Oracle HTTP Server **module**, mod_oprocmgr. It is used for **server** process management and administration.

transaction

A transaction is a collection of operations that work toward a goal of satisfying a single user's need. A transaction guarantees that all operations within it either succeed or fail.

As an example, a transfer from a savings account to a checking account has a transaction that debits the savings account and credits the checking account. If both operations succeed, the transaction is committed, otherwise the transaction is rolled back.

transaction processing

See OLTP.

transaction service

A transaction service is an **OTS** service that enables you to perform **transaction**s that span several **HTTP** requests. The transaction service is based on the XA open model transactions defined by the X/Open Company.

transactional DAD

A transactional DAD is a database that has been configured to be involved within a **transaction**. Once configured, more than one database can be managed by Oracle Application Server in committing resources on all databases involved in the transaction. A database cannot be involved in a transaction unless the **DAD** is configured to be transactional.

trustpoint

Also referred to as a trusted certificate authority (**CA**), a trustpoint is an issuer of **certificate**s whom you can trust. You can use **Oracle Wallet Manager** to manage trustpoint information associated with your application.

two-tier architecture

Two-tier architecture is the traditional **client**-**server** computing architecture with application front-end and logic executing on a client that accesses a database.

UDDI

Universal Description, Discover, and Integration (UDDI) is an online electronic registry that serves as electronic Yellow Pages, providing an information structure where various business entities register themselves and the services they offer through their **WSDL** definitions. There are two types of UDDI registries, public UDDI registries that serve as aggregation points for a variety of businesses to publish their services, and private UDDI registries that serve a similar role within organizations.

UNICODE

UNICODE is a 16-bit character **encoding** scheme supporting a large subset of characters found in both Western and Asian languages.

uniform resource identifier

See URI.

uniform resource locator

See URL.

Universal Description, Discover, and Integration

See UDDI.

URI

Uniform Resource Identifier (URI) is the generic term for all types of names and addresses that refer to **object**s on the World Wide Web. A **URL**is one kind of URI.

URL

Uniform Resource Locator (URL), a form of **URI**, is a compact string representation of the location for a resource that is available via the Internet. It is also the text-string format Web **client**s use to encode requests to Oracle Application Server.

virtual host

See multiport.

virtual machine

See JVM.

virtual pathname

In a virtual file system, a virtual pathname is a synonym that the virtual file system maps to a file stored in the file system maintained by the **host** machine's operating system.

VM

See JVM.

wait time

Wait time is the time between the submission of a request and initiation of the request.

wallet

A wallet is an abstraction used to store and manage security credentials for an individual entity. It implements the storage and retrieval of credentials for use with various cryptographic services.

WAR files

Web Archive (WAR) files contain the **Web component module**s of **J2EE** applications. A WAR file contains its **deployment descriptor** (DD), related files, and the `.class` files for the **servlet** or the `.jsp` files for a **JSP** page.

watcher

A watcher is a thread in an Oracle HTTP Server parent process. It is spawned by the parent `init` process of mod_oprocmgr. Its purpose is to periodically check the status of each process running on the **server**. If a process is dead or if it fails to start up, then the watcher thread will issue an **HTTP** request to start a new process.

Web component

J2EE Web components can be either JavaServer Pages (**JSP**s) or **servlet**s. Static **HTML** pages, **applet**s, or **server**-side utilities can be bundled with Web components during application assembly, but are not considered Web components by the **J2EE** specification.

Web services

Web services are Web applications that are transmitted over the Internet using the **XML**, **SOAP**, **WSDL**, and **UDDI** open standards. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the available services, and UDDI is used for listing the available services.

Web Services Description Language

See WSDL.

wireless portal

A wireless portal is accessible from wireless devices such as cellular telephones.

See Also: **Oracle Application Server Wireless**, **portal**

WSDL

Web Services Description Language (WSDL) is an **XML** format for describing network services containing RPC-oriented and **message**-oriented information. Programmers or automated development tools can create WSDL files to describe a service and can make the description available over the Internet. Client-side programmers and development tools can use published WSDL descriptions to obtain information about available Web Services and to build and create proxies or program templates that access available services.

X.500

X.500 is a standard for developing an electronic directory of users in an organization so that it can be part of a global directory available to anyone in the world with Internet access. Such a directory is sometimes called a global White Pages directory. The idea is to be able to look up people in a user-friendly way by name, department, or organization.

XDK

See Oracle XML Developer Kit.

XML

eXtensible Markup Language (XML) is a set of rules for defining data markup in a plain text format.

XSL

eXtensible Stylesheet Language (XSL) is used within style sheets to transform or render XML documents.

XSLT

The eXtensible Stylesheet Language Transformation (XSLT) processor transforms XML into other text-based formats, such as **HTML** and Wireless Markup Language (WML).

XSQL servlet

The XSQL servlet combines XML, **SQL**, and XSLT in the server to deliver dynamic Web content.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X