Oracle9*i* Application Server

Overview Guide

Release 1.0.2

November 2000

Part No. A87353-01

Oracle® Internet Application Server 8*i* Overview Guide, Release 1.0.2

Part No. A87353-01

Primary Author: Laurel Hale

Contributing Authors: Janice Nygard, Sanjay Singh, Priya Darshane, Francisco Abedrabbo, Jeremy Lizt, Matthieu Devin, Sheryl Maring

Contributors: Jesse Anton, Christopher Barron, Mike Boros, Juliana Button, Steve Button, Florence Chatzigianis, Jay Daughterty, Rakesh Dhoopar, Moe Fardoost, Marie Goodell, Helen Grembowicz, Pat Hinkley, Scott Howley, Pushkar Kapasi, Nick Kritikos, Vasant Kumar, Thomas Kurian, William Lankenau, Kai Li, Karen Masterson, Andrew Mendelsohn, Eduardo Mendez, Paul Narth, Paul Nock, Gilbert Pascal, Andy Page, Julia Pond, Hanne Rasmussen, Beth Roeser, Marcus Ruehmann, John Russell, David M. J. Saslav, Mark Scardina, Tejas Shah, Kasturi Shekhar, Jimmy Shi, Janelle Simmons, Ena Singh, Ingrid Snedecor, Deborah Steiner, Rao Vasireddy, Yaqing Wang, Alan Wiersba, Brian Wright

Contents

Send Us Your Comments vii

Note About Product Naming ix

Preface xi

1 Introduction

Oracle Internet Suite 1-2
Oracle9*i* Application Server Overview 1-3
Oracle9*i* Application Server Architecture 1-7
Two-Tier and Three-Tier Computing Models 1-7
Two-Tier Computing Model 1-7
Three-Tier Computing Model 1-8
Oracle9*i* Application Server 1-9
Available Versions 1-11
Supported Technologies and Programming Languages 1-13

2 Oracle9*i* Application Server Services

Communication Services 2-2
Oracle HTTP Server *powered by Apache* 2-3
Oracle HTTP Server Modules (mods) 2-4
mod_ssl 2-4
mod_plsql 2-4
mod_perl 2-4

mod_jserv 2-5
mod_ose 2-5
Oracle Plug-in for Microsoft IIS 2-5
PL/SQL Web Component Access 2-6
Java Web Component Access 2-6
Content Management Services 2-7
Oracle Internet File System 2-8
User's View 2-8
File System Management 2-9
Content Management 2-10
Development Tools 2-11
Supported Protocols 2-12
Component Architecture Overview 2-13
Business Logic Services 2-14
Oracle Business Components for Java 2-15
Oracle8*i* JVM 2-16
Oracle8*i* PLSQL 2-16
Oracle Forms Services 2-17
Presentation Services 2-18
Apache JServ 2-19
OracleJSP (JavaServer Pages) 2-19
Oracle PL/SQL Server Pages (PSP) 2-20
Perl Interpreter 2-21
Developer's Kits 2-22
Oracle Database Developer's Kit 2-23
Oracle Java Messaging Service (JMS) Toolkit 2-23
Oracle SQLJ Translator 2-23
Oracle Java Database Connectivity (JDBC) Drivers 2-25
Oracle Call Interface (OCI) 2-26
Oracle XML Developer's Kit 2-27
Oracle LDAP Developer's Kit 2-29
Portal Services 2-31
Oracle Portal 2-32
Create and Administer a Single Point of Interaction with Enterprise Information... 2-33
Self-Service Web Publishing 2-33

Implement Custom Portlets 2-34
Oracle Portal Architecture 2-34
Common Portal Services 2-35
Oracle Portal-to-Go 2-36
Personalized User View 2-36
Retrieve Content from Various Sources 2-37
Supported Target Device Formats 2-37
Oracle Portal-to-Go Architecture 2-38
Caching Services 2-40
Oracle Database Cache (formerly Oracle8*i* Cache) 2-41
Who Should Use Oracle Database Cache? 2-41
Performance and Scalability Benefits 2-41
Oracle Database Cache Environment 2-42
How Oracle Database Cache Differs from an Origin Database 2-42
How Do Applications Use Oracle Database Cache? 2-43
Oracle Web Cache 2-44
Oracle Web Cache Main Features 2-44
Oracle Web Cache Architecture Overview 2-45
System Services 2-46
Oracle Enterprise Manager 2-47
Oracle Advanced Security 2-49
Network Security 2-49
Enterprise User Security 2-50
Business Intelligence Services 2-52
Oracle Reports Services 2-53
Oracle Discoverer 3*i* Viewer 2-53
Architecture Overview 2-54
Workbook Accessibility 2-55

3 Developing Applications for Oracle9*i* Application Server

Content Publishing 3-2
Static Content 3-2
Dynamic Content 3-2
Common Gateway Interface (CGI) Applications 3-3
Perl Scripts 3-3

XML 3-4
Servlets 3-5
JavaServer Pages 3-7
PL/SQL Server Pages 3-9
Oracle Reports Services 3-12
Business Logic 3-15
Java Servlets, Applications, and Enterprise JavaBeans 3-15
Running Java on the JDK JVM 3-15
Running Java on Oracle8i JVM 3-18
Deploying Enterprise JavaBeans in the Middle Tier 3-19
Global Transactions Including Oracle Database Cache and the Origin Database 3-20
Oracle Forms Services 3-22
Processing Forms Requests 3-23

Index

Send Us Your Comments

Oracle9*i* Application Server Release 1.0.2, Overview Guide

Part No. A87353-01

Oracle Corporation 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 chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail - iasdocs_us@oracle.com
- Fax - (650) 654-6206 Attn: Oracle9*i* Application Server Documentation Manager
- Postal service:

  Oracle Corporation
  Oracle9*i* Application Server Documentation Manager
  500 Oracle Parkway, M/S 6op4
  Redwood Shores, CA 94065 USA

If you would like a reply, please give your name, address, and telephone number below.

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

Note About Product Naming

Oracle Database Cache is also referred to as Oracle8*i* Cache in the software, documentation, and collateral for Oracle9*i* Application Server. Both product names refer to the same middle-tier database caching service.

Preface

This book introduces the features and functionality of Oracle9*i* Application Server. It describes the services that are available to develop, deploy, and manage all of your content and applications on company Web sites and intranets.

This preface contains these topics:

- Audience
- Organization
- Related Documentation
- Conventions

Audience

Oracle9*i* Application Server Overview Guide is intended for a general audience who perform the following tasks:

- manage information technology (IT) departments
- administer company Web sites and intranets
- develop transactional Web applications

To use this document, you need general knowledge of application servers, Web servers, and database-driven Web sites. It also assumes that readers are familiar with the technologies and programming languages used in middle-tier environments.

Organization

This document contains:

Chapter 1, "Introduction"

Provides an introduction to Oracle9*i* Application Server, explaining how it supports your e-business. This chapter includes an architectural overview, a description of how you can use this product, and a list of the major technologies and programming languages it supports.

Chapter 2, "Oracle9i Application Server Services"

Gives an overview of each service contained in Oracle9*i* Application Server and lists additional information sources for each service.

Chapter 3, "Developing Applications for Oracle9i Application Server"

Demonstrates how you can use Oracle9*i* Application Server to build your applications.

Related Documentation

For more information, see the following documentation in the Oracle9*i* Application Server Release 1.0.2 documentation set:

The following documentation is included on your product CD-ROM:

- *Oracle9i Application Server Installation Guide*
- *Migrating from Oracle Internet Application Server 1.0.1*
- *Oracle HTTP Server Performance Guide*
- *Oracle9i Application Server Release Notes*

Oracle9*i* Application Server Documentation Library

The following documentation is included in the Oracle9*i* Application Server Documentation Library. All documents are provided in HTML and most are also provided in Adobe Portable Document Format (PDF).

- *Oracle9i Application Server Quick Tour*
- *Migrating from Oracle Application Server*

Communication Services

- *Apache 1.3.12 User's Guide*
- *Apache JServ Documentation*
- *Apache mod_perl Documentation*
- *mod_ssl Documentation*
- *OpenSSL Documentation*
- *Using the PL/SQL Gateway*
- *Oracle8i Oracle Servlet Engine User's Guide* (includes mod_ose)
- *Oracle Plug-in for Microsoft IIS Configuration and User's Guide*

Content Management Services

Oracle Internet File System

- *Quick Tour*
- *Setup and Administration Guide*
- *User's Gu*ide
- *Developer's Guide*
- *Class Reference*
- *Java Reference API*
- *XML Reference*

Business Logic Services

Oracle Business Components for Java (Business Components for Java)

- *Developing Business Components*
- *Tutorial - Building Business Components for Java*
- *Reference API*

Oracle8*i* JVM

- *Oracle8i CORBA Developer's Guide and Reference*
- *Oracle8i Enterprise JavaBeans Developer's Guide and Reference*
- *Oracle8i Java Developer's Guide*
- *Oracle8i Java Stored Procedures Developer's Guide*
- *Oracle8i Java Tools Reference*
- *Oracle8i JDBC Developer's Guide and Reference*
- *Oracle8i JPublisher User's Guide*
- *Oracle8i Oracle Servlet Engine User's Guide*
- *Oracle8i JavaServer Pages Developer's Guide and Reference*
- *Oracle8i SQLJ Developer's Guide and Reference*
- *Oracle8i Supplied Java Packages Reference*

Oracle Forms Services

- *Forms Developer Quick Tour*
- *Deploying Forms Applications to the Web*
- *Form Builder Reference Manual*
- *Guidelines for Building Applications*
- *Graphics Builder Reference Manual*
- *Procedure Builder Reference Manual*
- *Common Built-in Packages*

Presentation Services

Apache JServ

- *Apache JServ Documentation*

OracleJSP

- *Developer's Guide and Reference*
- *Developer's Toolkit*

Developer's Kits

Oracle XML Developer's Kit

- *Oracle8i Application Developer's Guide - XML*
- *Oracle8i XML Reference Guide*

Oracle Database Client Developer's Kits

- *Oracle8i SQLJ Developer's Guide and Reference*
- *Oracle8i JDBC Developer's Guide and Reference*

Oracle LDAP Developer's Kit

- *Oracle Internet Directory Application Developer's Guide*

Portal Services

Oracle Portal

- *Quick Tour*
- *Tutorial*
- *Configuration Guide*
- *Building Advanced Portals*
- *Oracle Single Signon Application Developer's Guide*

Oracle Portal-to-Go

- *Configuration Guide*
- *Implementation Guide*

Caching Services

Oracle Web Cache

This documentation is available at

`http://otn.oracle.com/products/ias`

Oracle Database Cache

- *Quick Tour*
- *Concepts and Administration Guide*

System Services

Oracle Enterprise Manager

- *Enterprise Manager Console Quick Tour*
- *Standard Management Pack Quick Tour*
- *Concepts Guide*
- *Configuration Guide*
- *Administrator's Guide*
- *Messages Manual*
- *Oracle Intelligent Agent User's Guide*

- *Oracle SNMP Support Reference Guide*

Business Intelligence Services

Oracle Discoverer 3*i*

- *Discoverer 3i Viewer Configuration Guide*

Oracle Reports Services

- *Reports Developer Quick Tour*
- *Publishing Reports to the Web*
- *Building Reports*
- *Reports Developer Reference Manual*
- *Guidelines for Building Applications*
- *Graphics Builder Reference Manual*
- *Procedure Builder Reference Manual*
- *Common Built-in Packages*

In North America, printed documentation is available for sale in the Oracle Store at

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

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

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://technet.oracle.com/membership/index.htm`

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://technet.oracle.com/docs/index.htm`

Conventions

This section describes the conventions used in the text and code examples of the this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	The C datatypes such as **ub4**, **sword**, or **OCINumber** are valid. When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles, emphasis, syntax clauses, or placeholders.	*Oracle8i Concepts* You can specify the *parallel_clause*. Run `U`*`old_release`*`.SQL` where *old_release* refers to the release you installed prior to upgrading.
UPPERCASE monospace (fixed-width font)	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, user names, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Specify the `ROLLBACK_SEGMENTS` parameter. Use the `DBMS_STATS.GENERATE_STATS` procedure.
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates executables and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, user names and roles, program units, and parameter values.	Enter `sqlplus` to open SQL*Plus. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (digits [ , precision ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`

|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}`
		`[COMPRESS | NOCOMPRESS]`

. . .	Horizontal ellipsis points indicate either:
	■ That we have omitted parts of the code that are not directly related to the example
	■ That you can repeat a portion of the code
		`CREATE TABLE ... AS subquery;`
		`SELECT col1, col2, ... , coln FROM employees;`

. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	

Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as it is shown.	`acctbal NUMBER(11,2);`
		`acct    CONSTANT NUMBER(4) := 3;`

Italics	Italicized text indicates variables for which you must supply particular values.	CONNECT SYSTEM/*system_password*

Convention	Meaning	Example

`UPPERCASE`	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.	`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`

1 Introduction

This chapter describes Oracle9*i* Application Server, how you can use it, and what technologies it supports.

This chapter contains the following topics.

- Oracle Internet Suite
- Oracle9i Application Server Overview
- Oracle9i Application Server Architecture
- Available Versions
- Supported Technologies and Programming Languages

Oracle Internet Suite

Oracle9*i* Application Server is part of the Oracle Internet Suite, which is a complete and integrated e-business platform solution. It consists of:

■ **Oracle8*i***:	Manages all of your content
■ **Oracle Internet Developer Suite**:	Builds all of your applications
■ **Oracle9*i* Application Server**:	Runs all of your applications

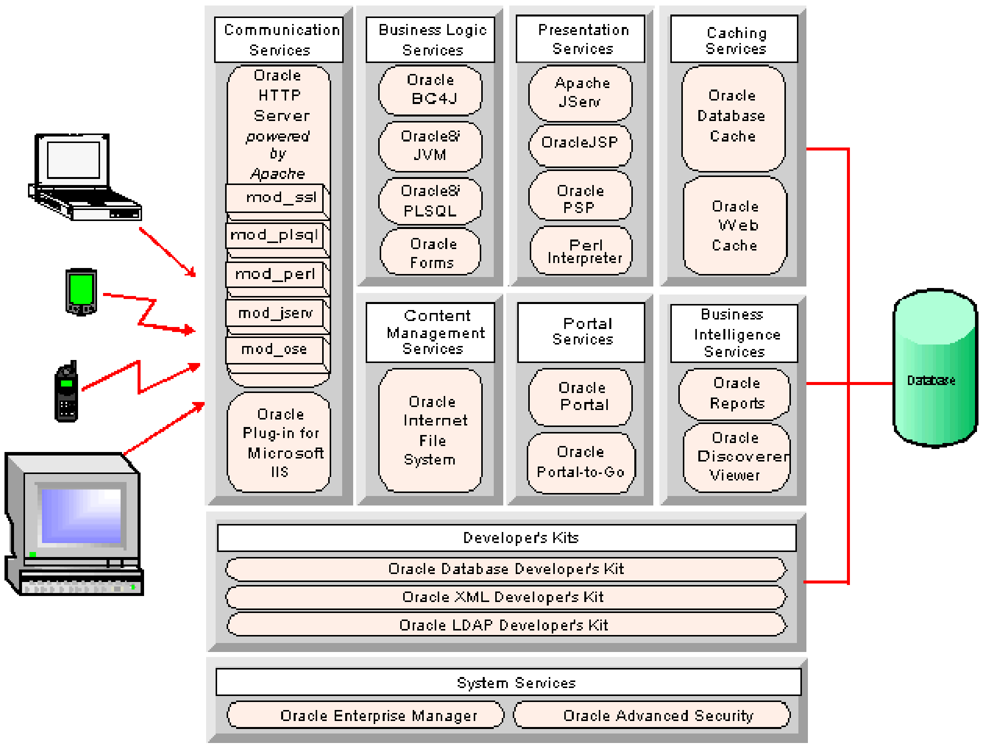
To successfully deliver scalable and high performance e-business solutions, you must be able to leverage an integrated, comprehensive, flexible, and open platform. Oracle Internet Suite provides integrated development, deployment, and management tools that simplify creating and deploying applications that you need to run your business on the Internet.

> **See Also:** The product pages at Oracle Technology Network at `http://otn.oracle.com` for more information about Oracle8*i* and Oracle Internet Developer Suite.

Oracle9*i* Application Server Overview

Oracle9*i* Application Server is a reliable, scalable, secure, middle-tier application server that is designed to support your evolution into an e-business. It provides a set of services so the technological complexity of assembling a complete middle-tier Internet infrastructure is managed for you. Oracle9*i* Application Server provides an infrastructure that can grow with your business—one that can start small and support growing numbers of users and sophisticated functionality on your Web sites. Figure 1–1 shows these services.

Figure 1–1 Oracle9i Application Server Services



Oracle9*i* Application Server provides the tools and infrastructure to start your e-business and support its growth. The following sections explain how Oracle9*i* Application Server services support each step in building your e-business.

1. **Launch your content on the Web**

   Oracle HTTP Server *powered by Apache* provides fast, reliable Web listening services so you can launch your content on the Web and make it accessible to your users.

2. **Manage your content**

   Once you have your content launched on the Web, you need integrated, flexible access to it and a powerful tool to manage it. Oracle Internet File System provides file system services that store multiple file types together in one file system hierarchy on Oracle8*i*. This heterogeneous file system hierarchy is accessible through Web browsers, Microsoft Windows networking, FTP, or e-mail clients.

3. **Build transactional Web applications**

   After you launch your content to the Web and manage it, you need to build applications that can access and manipulate your content. Using Business Logic Services, you can build and run transactional Web applications in:

   - Java
   - PL/SQL
   - Oracle Forms

4. **Build a presentation layer**

   To make your transactional Web applications more re-usable and to provide a dynamic interface to users, you must separate your business logic from the presentation layer of your applications. Oracle9*i* Application Server Presentation Services provide the tools to build a presentation layer in:

   - JavaServer Pages or servlets
   - PL/SQL Server Pages
   - Perl
   - XML with XSL (Extensible Markup Language with Extensible Stylesheet Language)

5. **Access data in your database**

   Because a database-driven Web site requires database-driven Web applications, Oracle9*i* Application Server includes Developer's Kit services that you can use to access your database using:

   - OCI (Oracle Call Interface)
   - JDBC (Oracle JDBC Drivers)
   - SQLJ (Oracle SQLJ Translator)

6. **Build portal sites**

   When you have many different components to offer users, you can integrate them all into portal sites. Portal sites provide one consistent user interface so your users have fast access to relevant and current information without having to navigate dissimilar user interfaces and enter multiple passwords. Use these Portal Services:

   - Oracle Portal for traditional desktop clients, or
   - Oracle Portal-to-Go in Oracle9*i* Application Server Wireless Edition for mobile clients.

7. **Scale and deploy your content**

   When your e-business Web site generates an increasing number of hits, you need to scale it so your users have fast, reliable access to applications and data. Then you can use these Caching Services provided in Oracle9*i* Application Server:

   - Oracle Database Cache to cache frequently requested data from your database, or
   - Oracle Web Cache to cache and load balance HTTP requests across a Web farm or cluster of Oracle9*i* Application Server machines.

8. **Manage your deployment environment**

   Now that you have built a complex deployment environment, you need centralized tools to manage your systems and your security. Oracle9*i* Application Server System Services include:

   - Oracle Enterprise Manager for systems management, and
   - Oracle Advanced Security for security management.

9. **Understand your business and your Web site activity**

   You can analyze your Web site activity and your business by using these Business Intelligence Services:

   - Oracle Reports Service to define and publish reports to the Web that pull data directly from Oracle8*i*, or
   - Oracle Discoverer 3*i* Viewer to view Discoverer workbooks, which provide ad hoc querying, what-if analysis, and advanced analysis capabilities.

Oracle9*i* Application Server Architecture

Oracle9*i* Application Server consists of a set of services that can be implemented in a distributed environment for scalability and reliability. The following sections provide an architectural overview of Oracle9*i* Application Server.

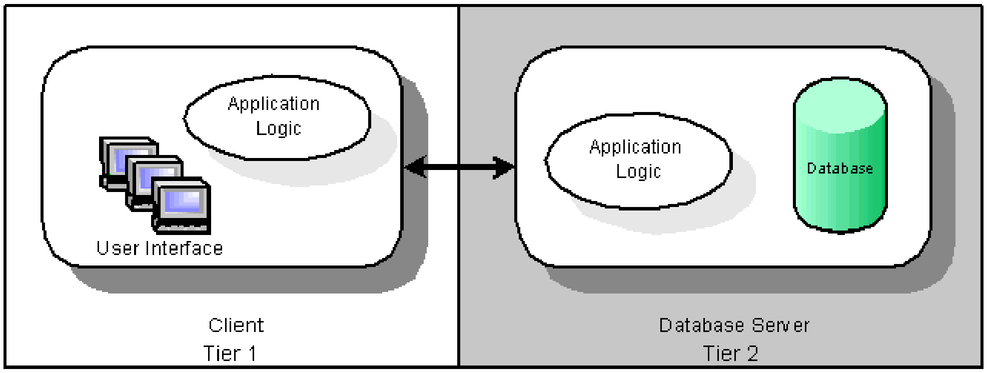
Two-Tier and Three-Tier Computing Models

Client/server computing architectures are commonly described as having two or more tiers according to how application logic is distributed between client and server. Minimally, a client/server architecture must have a client tier and a server tier. Oracle's internet computing model is based on a multitiered computing model in which Oracle9*i* Application Server functions as a middle tier, or application server tier.

Two-Tier Computing Model

Traditional database client/server architecture is based on a two-tier computing model. This model consists of a client tier and a database server tier (see Figure 1–2). Processing tasks and application logic are shared between the database server and the client.

Figure 1–2 Two-tier Computing Model



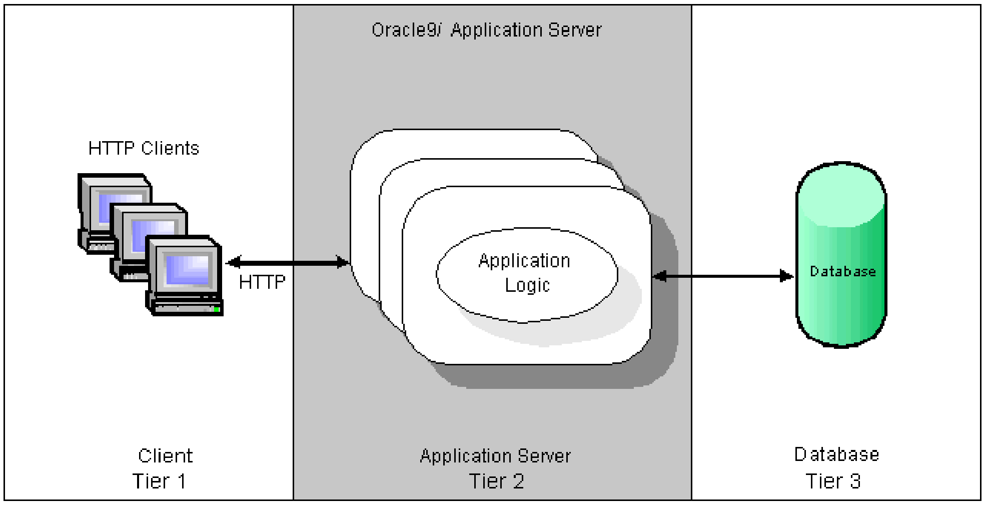
Several disadvantages exist for this model. The clients in a two-tier computing model are fat clients, where much of the processing power and application logic reside. This makes the clients costly to maintain. Furthermore, clients can be operating on different platforms, necessitating the deployment of platform-specific versions of applications.

Three-Tier Computing Model

The three-tier computing model evolved to address the problems of the two-tier model. In a three-tier model, a middle tier exists between clients and the database server. This middle tier consists of an application server that contains the bulk of the application logic. Clients in the model are thin clients. With this architecture, application logic resides in a single tier and can be maintained easily at one location. The architectural design of the middle tier is optimized for server functions including access to a database.

Oracle9*i* Application Server serves as the middle tier of the three-tier model as shown in Figure 1–3.

Figure 1–3 Three-tier Oracle9i Application Server Architecture



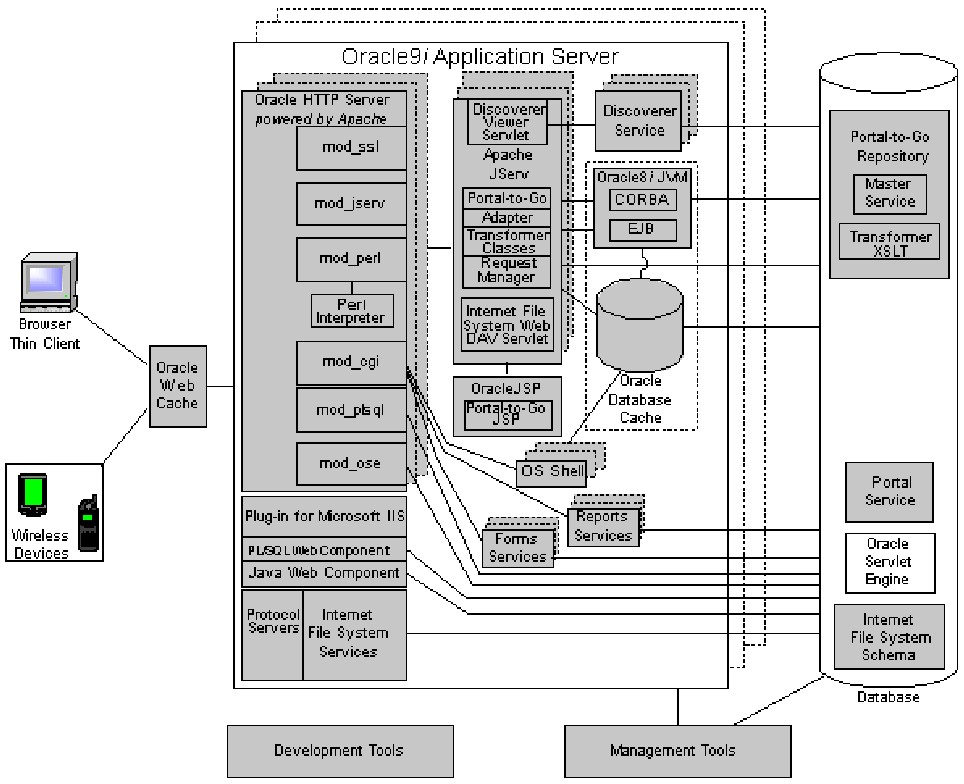
In this three-tier architecture, the client software (the client tier) is lightweight enough to be downloaded on demand, and does little but present the user interface for a server-side application. The bulk of the application logic is implemented either in the application server or in the database.

Oracle9*i* Application Server

Oracle9*i* Application Server enables users to deploy applications within its multitiered architecture. The middle-tier server centrally manages application logic sending request responses back to thin clients, typically Web browsers. A third tier houses your database, so transaction processing on the database may be optimized. This multitiered model offers great savings in administration and maintenance costs when deploying applications.

Oracle9*i* Application Server components reside on the middle-tier in a three-tier architecture as shown in Figure 1–4.

Figure 1–4 Oracle9i Application Server Architecture



Alternatively, you may also run the Oracle9*i* Application Server in a multitiered architecture of four or more tiers, depending on your application needs. For example, it is possible to divide the Report Services across multiple machines. In a four-tier configuration, you should run the Reports Web CGI or the Reports Servlets on the same machine as the HTTP Server and run the Report Server on a separate machine. In this example, the client browser resides in the first tier and your database resides on a fourth-tier.

> **See Also:** *Publishing Reports to the Web with Oracle9i* Application Server in the Oracle9*i* Application Server Documentation Library for more information on this example.

Available Versions

Currently, Oracle9*i* Application Server is available in three versions:

- The **Standard Edition** is suitable if you need a lightweight Web server with minimal application support.
- The **Enterprise Edition** is recommended if you have a medium-sized to large-sized Web site that handles a high volume of transactions.
- The **Wireless Edition** gives you full enterprise support that includes Oracle Portal-to-Go if you need to deliver content to mobile devices.

Table 1–1 lists what services each edition contains.

Table 1–1 Available Versions of Oracle9i Application Server

	Standard Edition	Enterprise Edition	Wireless Edition
Communication Services			
Oracle HTTP Server	x	x	x
mod_ssl	x	x	x
mod_plsql	x	x	x
mod_perl	x	x	x
mod_jserv	x	x	x
mod_ose	x	x	x
Oracle Plug-in for Microsoft IIS	x	x	x
Content Management Services			
Oracle Internet File System	x	x	x
Business Logic Services			
Oracle Business Components for Java	x	x	x
Oracle8*i* JVM (Java Virtual Machine)	x	x	x
Oracle8*i* PLSQL		x	x
Oracle Forms Services		x	x
Presentation Services			
Apache JServ	x	x	x
OracleJSP (JavaServer Pages)	x	x	x
Oracle PL/SQL Server Pages	x	x	x
Perl Interpreter	x	x	x

Table 1–1 Available Versions of Oracle9i Application Server (Cont.)

	Standard Edition	Enterprise Edition	Wireless Edition
Developer's Kits			
Oracle Database Client Developer's Kit	x	x	x
Oracle XML Developer's Kit	x	x	x
Oracle LDAP Developer's Kit	x	x	x
Portal Services			
Oracle Portal	x	x	x
Oracle Portal-to-Go			x
Caching Services			
Oracle Database Cache		x	x
Oracle Web Cache		x	x
System Services			
Oracle Enterprise Manager[1]	x	x	x
Oracle Advanced Security	x	x	x
Business Intelligence Services			
Oracle Reports Services		x	x
Oracle Discoverer 3*i* Viewer		x	x

[1] Standard edition contains the Enterprise Manager console only; Enterprise edition contains both Enterprise Manager console and Management Server. For more information, see "Oracle Enterprise Manager" on page 2-47.

Supported Technologies and Programming Languages

For publishing content, transaction processing, and program development and deployment, Oracle9*i* Application Server supports these technologies and programming languages:

- Oracle Business Components for Java Runtime
- EJB (Enterprise JavaBeans)
- Java 2
- Java Stored Procedures
- JDBC (Java Database Connectivity)
- JMS (Java Messaging Service)
- JNDI (Java Naming and Directory Interface)
- Servlets
- JSP (JavaServer Pages)
- SQLJ
- CORBA (Common Object Request Broker Architecture)
- PL/SQL
- SQL
- HTML (Hypertext Markup Language)
- XML (Extensible Markup Language)
- Perl
- Web DAV (Distributed Authoring and Versioning)
- OCI (Oracle Call Interface)
- ODBC (Open Database Connectivity)
- OLE (Object Linking and Embedding)
- IIOP (Internet Inter-ORB Protocol)
- RMI (Remote Method Invocation)/IIOP
- LDAP (Lightweight Directory Access Protocol)
- SMTP (Simple Mail Transfer Protocol)
- IMAP (Internet Message Access Protocol)
- HTTP (Hypertext Transfer Protocol)
- FTP (File Transfer Protocol)
- SMB (Server Message Block)
- SSL (Secure Sockets Layer)
- WML (Wireless Markup Language)
- Tiny HTML (HTML for handheld devices)
- VoxML (Voice Markup Language)
- VoiceXML (Voice Extensible Markup Language)
- HDML (Handheld Devices Markup Language)

See Also:

- Chapter 2, "Oracle9i Application Server Services" for a description of each Oracle9*i* Application Server service.
- Chapter 3, "Developing Applications for Oracle9i Application Server" for information about developing and deploying applications with Oracle9*i* Application Server.

2

Oracle9*i* Application Server Services

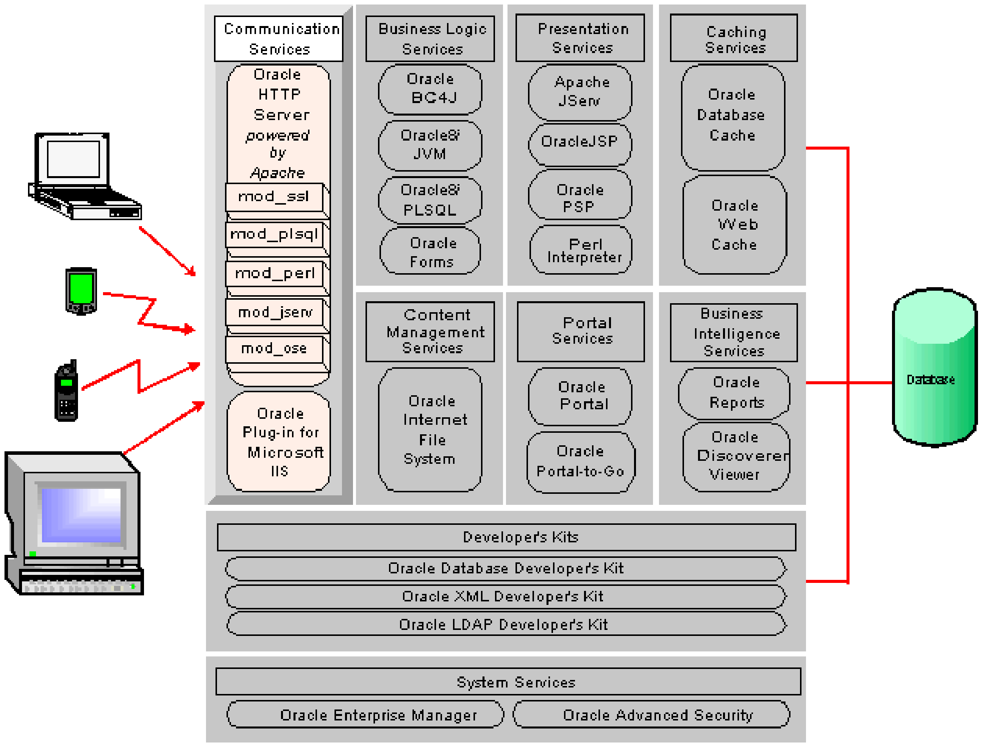
Oracle9*i* Application Server services provide an integrated Internet infrastructure that supports open Internet technologies. This chapter describes all of the Oracle9*i* Application Server services:

- Communication Services
- Content Management Services
- Business Logic Services
- Presentation Services
- Developer's Kits
- Portal Services
- Caching Services
- System Services
- Business Intelligence Services

Communication Services

These services handle all incoming requests received by Oracle9*i* Application Server, by providing Web listening services. Some of these requests are processed by Oracle HTTP Server and some requests are routed to other areas of Oracle9*i* Application Server for processing. The major elements that support these services are highlighted in Figure 2–1.

Figure 2–1 Communication Services in Oracle9i Application Server



Oracle HTTP Server *powered by Apache*

Oracle9*i* Application Server uses Oracle HTTP Server to provide Web listening services, which is powered by Apache Web server technology. Using this technology offers the following:

- **Scalability**: Because the Apache Web server is designed to be very scalable, it can be replicated across many middle-tier machines.
- **Stability**: The Apache Web server is the defacto standard for Web listeners. Apache, an open source technology, has been continuously improved by a large community of developers since its first release. Consequently, Apache Web server technology is very stable.
- **Speed**: The compact design of the Apache Web server makes it very fast when it responds to requests, and it is possible to rapidly reconfigure Apache using *graceful restart*, which reloads the configuration files in seconds.
- **Extensibility**: The Apache Web server delegates the handling of HTTP requests to its modules (mods), which add functionality not included in the server by default. Using the Apache APIs, it is easy to extend Apache functionality. A large number of mods have already been created and are included on your product CD-ROM. Although the default Apache Web server supports only stateless transactions, you can configure it to support stateful transactions by leveraging the functionality supplied by Apache JServ.

See Also:

- "Apache JServ" on page 2-19 for information about Apache JServ.
- Oracle HTTP Server documentation in the Oracle9*i* Application Server Documentation Library for detailed information about Oracle HTTP Server.

Oracle HTTP Server Modules (mods)

In addition to the compiled Apache mods provided with Oracle HTTP server, Oracle has enhanced several of the standard mods and has added Oracle-specific mods, which are described in the following sections.

mod_ssl

This module provides standard S-HTTP that is fully supported by Oracle. It enables secure listener connections with an encryption mechanism that is provided by Oracle through Secure Sockets Layer (SSL).

See Also: *mod_ssl Documentation* in the Oracle9*i* Application Server Documentation Library for detailed information.

mod_plsql

This module routes PL/SQL requests to the Oracle8*i* PLSQL service, which then delegates requests to PL/SQL programs.

See Also: *Using the PL/SQL Gateway* in the Oracle9*i* Application Server Documentation Library for detailed information.

mod_perl

This module forwards Perl application requests to the Perl Interpreter that is embedded in Oracle HTTP Server. The primary advantages of using mod_perl are power and speed. The embedded Perl Interpreter saves the overhead of starting an external interpreter, and the code caching feature, where modules and scripts are loaded and compiled only once, allows the server to run code that is already loaded and compiled.

See Also: *Apache mod_perl Documentation* in the Oracle9*i* Application Server Documentation Library for detailed information.

mod_jserv

This module routes all servlet requests to the Apache JServ servlet engine, which is embedded in Oracle HTTP Server. To support stateful transactions, it can share servlets across multiple zones and ensures that requests get routed to the same servlet engine.

See Also: *Apache JServ Documentation* in the Oracle9*i* Application Server Documentation Library for detailed information.

mod_ose

This module forwards servlet requests to stateful Java and PL/SQL servlets in Oracle Servlet Engine (OSE) contained in Oracle8*i*. This module:

- keeps session IDs in cookies or redirected URLs
- routes requests to the appropriate OSE sessions
- communicates with OSE over NS (SQL*Net) which gives
  - firewall support between Oracle HTTP Server and OSE, and
  - connection pooling.

See Also: *Oracle8i Oracle Servlet Engine User's Guide* in the Oracle9*i* Application Server Documentation Library for detailed information.

Oracle Plug-in for Microsoft IIS

With Oracle Plug-in for Microsoft IIS, you can use Microsoft Internet Information Server (IIS) to directly access PL/SQL and Java Web components stored in an Oracle database.

Oracle Plug-in for Microsoft IIS provides functionality in a Microsoft IIS environment that is similar to the Oracle HTTP Server Modules, mod_plsql and mod_ose. Using it, you can access Web components in one of two ways:

- passing a preconfigured virtual directory prefix (PL/SQL access)
- passing a predefined file extension and virtual directory prefixes which are stored in the Java configuration file (Java access)

PL/SQL Web Component Access

Oracle Plug-in for Microsoft IIS supports accessing PL/SQL server pages and PL/SQL stored procedures that have been written with Oracle PL/SQL Web Toolkit. PL/SQL requests are filtered using a predefined prefix and then are executed using pooled database connections. Users have the ability to configure PL/SQL component access from multiple databases.

Java Web Component Access

Oracle Plug-in for Microsoft IIS supports accessing JavaServer Pages (JSPs) and servlets. Java requests are filtered using a Java configuration file.

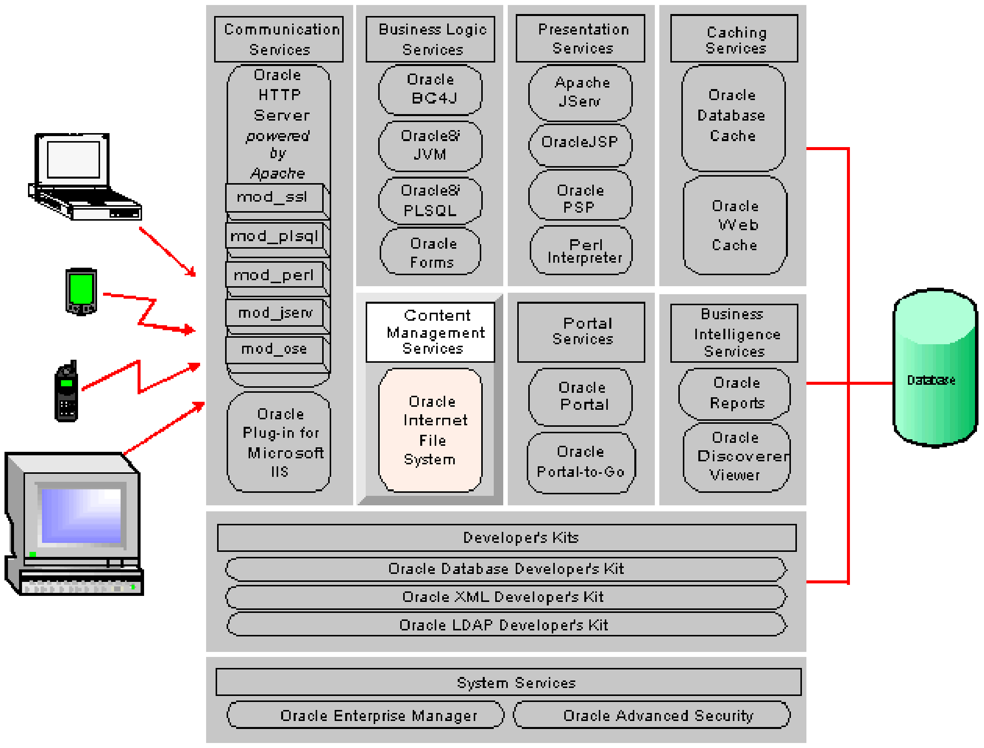
Note: Currently users can configure Java component access from only one database in a configuration.

See Also: *Oracle Plug-in for Microsoft IIS Configuration and User's Guide* in the Oracle9*i* Application Server Documentation Library for detailed information.

Content Management Services

These services make all of your content, regardless of the file type, accessible in one heterogeneous file hierarchy through Web browsers, Microsoft Windows networking, File Transfer Protocol (FTP), or an e-mail client. In addition, you can use these services to configure sophisticated file searching capabilities, event alerts, and check-in-check-out functionality to support collaborative projects. These services are highlighted in Figure 2–2.

Figure 2–2 Content Management Services in Oracle9i Application Server



Oracle Internet File System

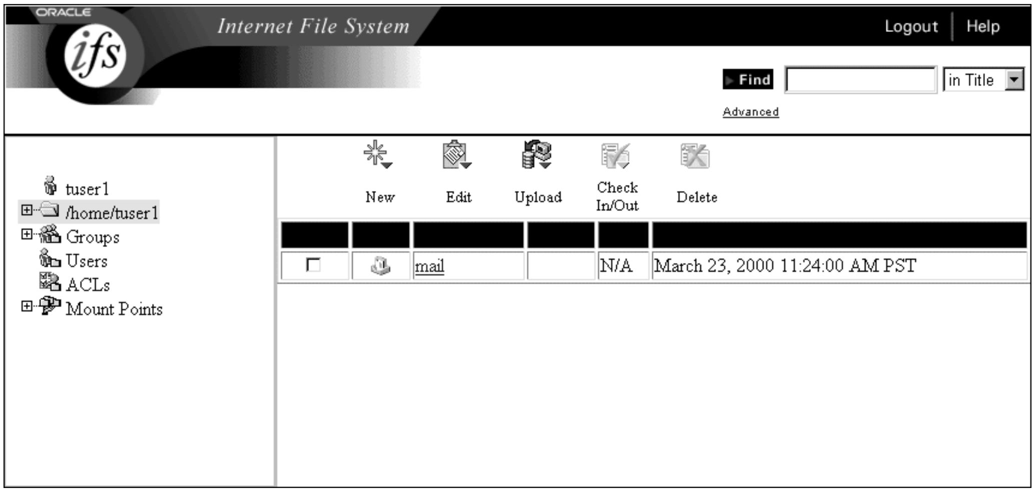
Oracle Internet File System is a service that stores files in an Oracle8*i* database. From the user's viewpoint, Oracle Internet File System appears as any other file system accessible through Web browsers, Microsoft Windows networking, FTP, or an e-mail client. The fact that files are stored in the database is transparent to users because users do not directly interact with the database.

Unlike other file systems, Oracle Internet File System stores all of your files—from Web pages to e-mail, from spreadsheets to XML files—in the same file system. For example, you can display e-mail message files, Web files, and word processing files within a single file hierarchy, or you can perform a single search to locate all references to a topic.

User's View

You can view your file hierarchies through either a Web or a Windows interface. Although each interface has a unique look and feel, the Web and the Windows interfaces perform the same functions. Figure 2–3 shows the Web interface as it appears in a Web browser.

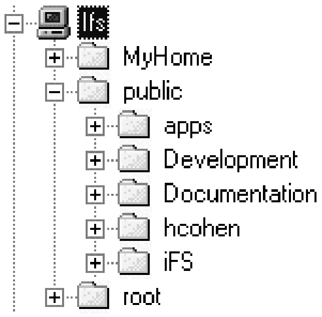
Figure 2–3 Oracle Internet File System Web Interface



☐		mail		N/A	March 23, 2000 11:24:00 AM PST

Figure 2–4 shows the Windows interface as it appears using Windows Explorer in the Microsoft Windows operating system.

Figure 2–4 Oracle Internet File System Windows Interface



File System Management

Users can manage the files that they own or those for which they have the correct permissions. The main file management features for users include

- **renaming files**: Users can rename files that they own.
- **deleting files**: Users can delete files for which they have the correct permissions.
- **uploading files**: Users can upload files using FTP, Windows Explorer (by using Server Message Block and Web Folders), and the Web.
- **setting automatic expiration on files**: Users can set an expiration date for each file that they upload to Oracle Internet File System. The file is automatically deleted on that date.
- **modifying file attributes**: Users can view and modify file attributes such as description, filename, and the access control permissions.

Content Management

Oracle Internet File System content management features support collaborative projects and sophisticated searching across different types of files. The main content management features include

- **check-in, check-out (CICO)**: For collaborative projects, users can check out files so that others cannot overwrite their work. Files remain locked until either users check them back in or an administrator releases the lock.
- **versioning**: Users can decide to make a file versioned. Each time the versioned file is checked in, a new version is created and stored. File versions can be kept or purged as needed.
- **searching**: Oracle Internet File System includes a Find utility that uses *inter*Media Text to search across multiple file types. This advanced search utility allows users to search quickly without having to know database syntax.
- **multiple folders per file**: To avoid making multiple copies of files when the same document fits into a number of folder categories, multiple folders can be assigned to files. This feature reduces maintenance when files are updated, and it also saves storage space.
- **extensible file attributes**: Application developers can extend Oracle Internet File System to manage new types for custom information. For each new type, the developer can define custom attributes about the information that are then used to track and search for the information.

Development Tools

The purpose of hosting files on any type of file service is to make them available to applications. While all of your files are still available to word processors, Web authoring tools, and other types of client applications, Oracle Internet File System includes a Software Development Kit (SDK) that you can use to customize the server for specific purposes, such as making file contents available to reporting tools, sending alerts when files are updated, or formatting files for Web-based applications.

Oracle Internet File System SDK includes the following components described in Table 2–1.

Table 2–1 Oracle Internet File System SDK Components

SDK Component	Description	Examples
Default XML Parser	A Java application that extracts file contents and stores them separately from the file itself. You can add your own parsers to Oracle Internet File System by registering them through XML.	■ Extract information from files and make it available to data mining tools ■ Organize your files and folders into subclasses so you can set alerts or add additional attributes to files
Default XML Renderer	Like the parser, a Java application that you register with Oracle Internet File System by using XML	■ Reconstruct a parsed file or portions of a parsed file into other formats, such as HTML
Agents	A Java application that executes on a schedule that you define	■ Send e-mail alerts when insert, delete, or update operations occur ■ Set events, such as file deletions or file moves, that execute on a schedule
Overrides	A method that allows an application program to intervene with the standard schema operations (Insert, Update, and Free)	■ Add an attribute with a specified value to every new instance of a custom document class ■ Provide special server-side validation on Update operations

Table 2–1 Oracle Internet File System SDK Components (Cont.)

SDK Component	Description	Examples
Java API	A collection of published Java classes	■ Create and manage agents ■ Develop applications ■ Create custom parsers ■ Manage database objects ■ Create custom renderers ■ Manage search functionality

Supported Protocols

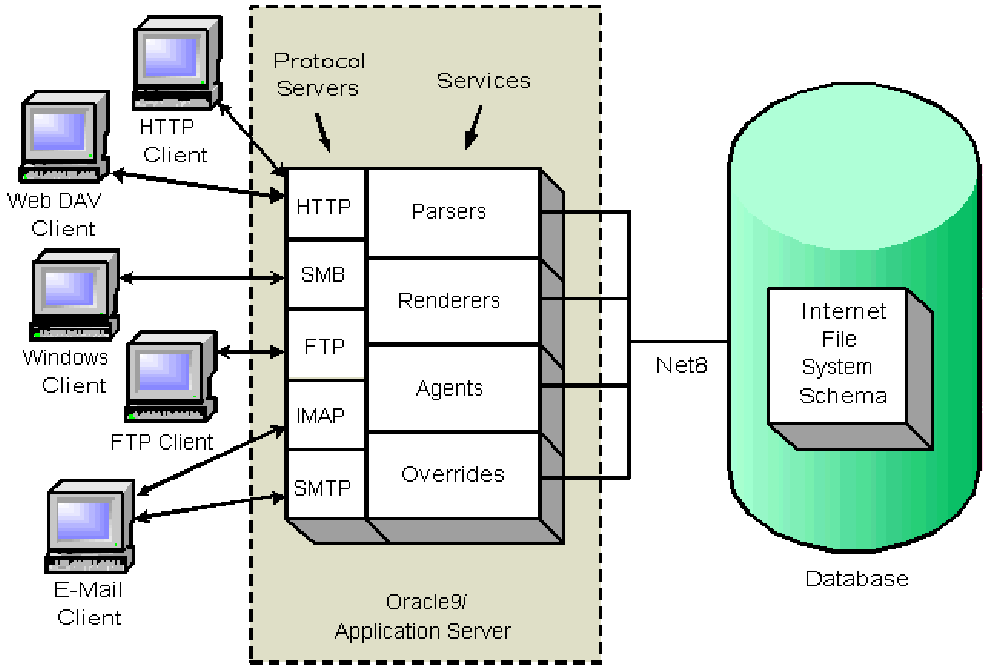
Oracle Internet File System includes a set of standard protocol servers. Each protocol server accepts commands from a standard client and maps those commands to schema operations. The protocol servers that are included are:

- **SMTP (Simple Mail Transfer Protocol)**: Verifies and delivers e-mail, and runs parsers.
- **IMAP (Internet Message Access Protocol)**: Delivers e-mail messages to your e-mail applications.
- **HTTP**: Delivers Web pages to client browsers and provides Web DAV (Distributed Authoring and Versioning) features for Web Folders.
- **FTP**: Transfers files.
- **SMB (Server Message Block)**: Provides access to Oracle Internet File System from Windows applications, including Windows Explorer.

Component Architecture Overview

Figure 2–5 shows the main components of Oracle Internet Files system, which includes the protocol servers, the services, and the schema.

Figure 2–5 Oracle Internet File System Component Architecture

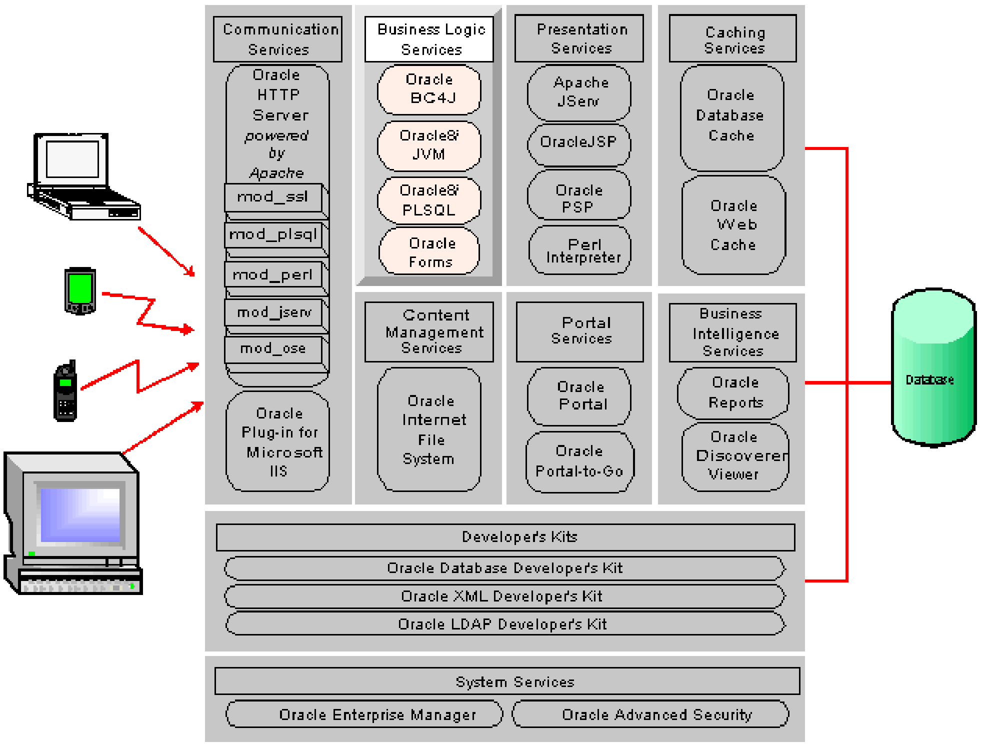


See Also: Oracle Internet File System documentation in the Oracle9*i* Application Server Documentation Library.

Business Logic Services

These services support your application logic. Using Business Logic Services, you can build and run your Web applications. The following sections describe the major elements that provide business logic services in Oracle9*i* Application Server. These services are highlighted in Figure 2–6.

Figure 2–6 Business Logic Services in Oracle9i Application Server



Oracle Business Components for Java

Oracle Business Components for Java is a 100% pure Java and XML framework that enables productive development, portable deployment, and flexible customization of multitiered, database applications from reusable business components. Application developers can use this framework to

- author and test business logic in components that automatically integrate with databases.
- reuse business logic through multiple SQL views of data that support different application tasks
- access and update the views from servlets, JavaServer Pages (JSPs), and Thin-Java Swing clients
- customize application functionality in layers without modifying the delivered application

Once developed, these application services can then be deployed as either EJB Session Beans or CORBA Server Objects on Oracle9*i* Application Server.

Note: Oracle JDeveloper, an application development tool, is required to create, manage, debug, and deploy Oracle Business Components for Java applications. You can download Oracle JDeveloper from Oracle Technology Network at `http://otn.oracle.com/products/jdev/`

See Also: Oracle Business Components for Java documentation in the Oracle9*i* Application Server Documentation Library.

Oracle8*i* JVM

Designed as a highly scalable, server-side Java platform, Oracle8*i* JVM is an enterprise-level server environment that uses 100% pure Java and supports Enterprise JavaBeans, CORBA, and database stored procedures. Oracle8*i* JVM achieves scalability through its unique architectural design, which aids memory management when the number of users increases.

Oracle8*i* JVM provides a number of advantages, such as

- enhanced garbage collection algorithms that facilitate more efficient memory allocation
- support for concurrent, stateful, conversational clients
- shared resources across multiple user sessions so only one Java Virtual Machine needs to run to provide user session support
- automatic load balancing so that no multi-threaded applications are necessary

Oracle8*i* JVM is the common foundation for running Java and Java services in Oracle9*i* Application Server and Oracle8*i*. Consequently, components can be seamlessly moved across tiers without having to change any code.

See Also: Oracle8*i* JVM documentation in the Oracle9*i* Application Server Documentation Library.

Oracle8*i* PLSQL

Oracle8*i* PLSQL is a scalable engine for running business logic against data in Oracle Database Cache and Oracle8*i* database. It enables users to invoke PL/SQL procedures stored in Oracle databases through their Web browsers. The stored procedures retrieve data from tables in the database and generate HTML pages that include the data which are then returned to the client browser.

See Also: *PL/SQL User's Guide and Reference* in the Oracle Database Documentation Library.

Oracle Forms Services

You can run applications based on Oracle Forms technology either over the Internet or over your corporate intranet. On the application server tier, Oracle Forms Services consists of a listener and a runtime engine, where the application logic is stored. On the client tier, Oracle Forms Services consists of a Java applet, which provides the user interface for the runtime engine, and Oracle JInitiator (a Java plug-in) that provides the ability to specify the use of a specific Java virtual machine on the client.

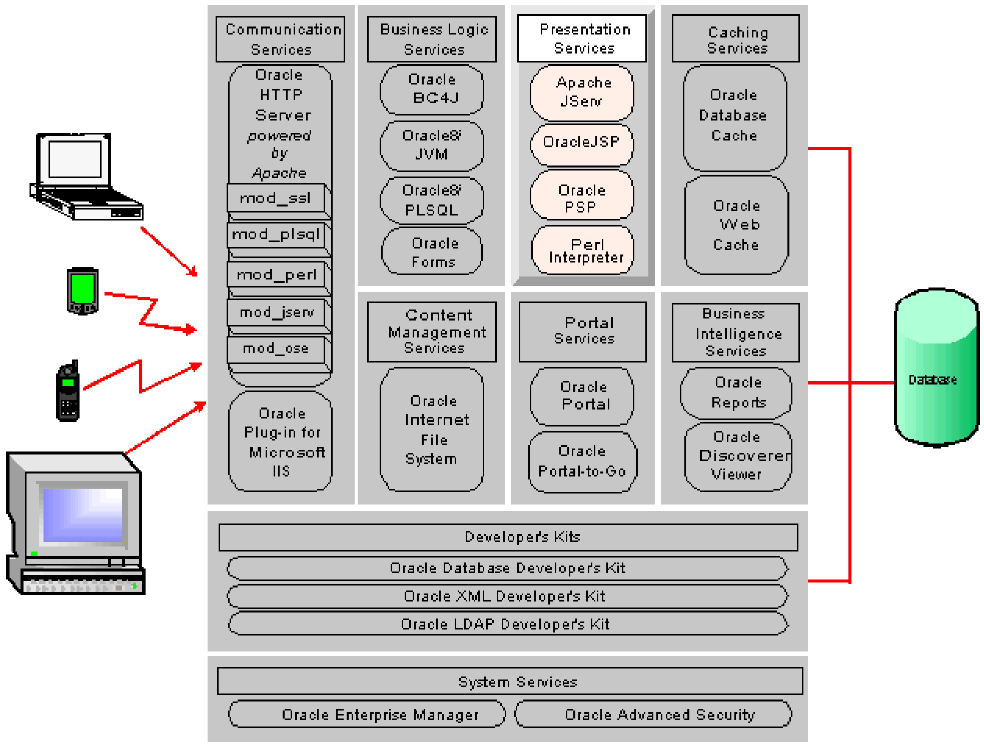
In Oracle9*i* Application Server, when a user submits a URL to launch an Oracle Forms-based application, the Web listener accepts the request and downloads the Oracle Forms applet to the user's browser. Then the Oracle Forms applet establishes a persistent connection to an Oracle Forms runtime engine. All processing takes place between the Oracle Forms applet and the Oracle Forms Services runtime engine, which seamlessly handles any queries or commits to the database.

See Also: Oracle Forms Services documentation in the Oracle9*i* Application Server Documentation Library.

Presentation Services

These services deliver dynamic content to client browsers, supporting servlets, JavaServer Pages, Perl/CGI scripts, PL/SQL Pages, forms, and business intelligence. By using Presentation Services, you can build your presentation layer for your Web applications. These services are highlighted in Figure 2–7.

Figure 2–7 Presentation Services in Oracle9i Application Server



Apache JServ

Apache JServ is a 100% pure Java servlet engine that is fully compliant with the following Sun Microsystems Java specifications:

- Java Servlet APIs version 2.0
- Java Runtime Environment (JRE) version 1.1

Apache JServ works on any Java Virtual Machine that is compliant with this JRE and will execute any Java servlet that is compliant with this version of the Java Servlet APIs specification.

When the HTTP server receives a servlet request, it is routed to mod_jserv, which forwards the request to the Apache JServ servlet engine.

> **See Also:** *Apache JServ Documentation* in the Oracle9*i* Application Server Documentation Library.

OracleJSP (JavaServer Pages)

As Sun Microsystems explains, JavaServer Pages technology extends Java Servlet technology, and supports the use of Java calls and scriptlets within HTML and XML pages. By using JSP pages, you can combine static template data with dynamic content to create user interfaces. JSP pages support component-based development, separating business logic (usually in JavaBeans) from the presentation, thus allowing developers to focus on their areas of expertise. Consequently, JSP developers (who may not know Java) can focus on presentation logic, while Java developers can focus on business logic.

For general information about JavaServer Pages, refer to the JavaServer Pages Specification (available from http://java.sun.com).

OracleJSP is an implementation of JavaServer Pages version 1.1 as specified by Sun Microsystems and extends this specification to provide these benefits:

- **portability between servlet environments**: OracleJSP pages are supported on all Web servers that support Java servlets built to the version 2.0 or higher specification. Consequently, you can migrate your existing JSP pages that are compliant with the version 1.0 specifications to Oracle9*i* Application Server without needing to rewrite them.
- **support for SQLJ:** SQLJ is a standard syntax for embedding SQL commands directly into Java code. OracleJSP supports SQLJ programming in JSP scriptlets. This includes support for an additional filename extension, `.sqljsp`, which causes the OracleJSP translator to invoke the Oracle SQLJ translator.

- **OracleJSP Markup Language (JML)**: Oracle provides the JML tag set as a sample tag library, which allows developers to use loop, conditional, and other high-level programming logic without having to know Java syntax.
- **extended National Language Support (NLS)**: OracleJSP provides extended NLS support for servlet environments that cannot encode multibyte request parameters and bean property settings. For such environments, OracleJSP offers the `translate_params` configuration parameter, which can be enabled to direct OracleJSP to override the servlet container and do the encoding itself.
- **extended datatypes**: OracleJSP provides the `JmlBoolean`, `JmlNumber`, `JmlFPNumber`, and `JmlString` JavaBean classes in the `oracle.jsp.jml` package to wrap the most common Java datatypes. These extended datatypes provide a way to work around the limitations of Java primitive types and wrapper classes in the standard `java.lang` package.
- **custom JavaBeans:** OracleJSP includes a set of custom JavaBeans for accessing an Oracle database.

See Also:

- JavaServer Pages Specification at `http://java.sun.com` for general information about JavaServer Pages.
- OracleJSP documentation in the Oracle9*i* Application Server Documentation Library.

Oracle PL/SQL Server Pages (PSP)

Oracle PSPs are analogous to JavaServer Pages, but they use PL/SQL rather than Java for the server-side scripting. Oracle PSP includes the PSP Compiler and the PL/SQL Web Toolkit. By using this service when developing applications, you can separate page format from application logic.

Starting with either an existing Web page or stored procedure, you can create dynamic Web pages. These dynamic Web pages can perform database operations and display the results as HTML, XML, plain text, or some other document type that your browser has been configured to recognize. Typically, a PL/SQL server page is intended to be displayed in a Web browser. However, it can also be retrieved and interpreted by a program that can make HTTP requests, such as a Java or Perl application.

A PSP file can contain whatever content you like, with text and tags interspersed with PSP directives, declarations, and scriptlets:

- In the simplest case, it is nothing more than an HTML file or an XML file. Compiling it as a PSP produces a stored procedure that outputs the exact same HTML or XML file.
- In the most complex case, it is a PL/SQL procedure that generates all the content of the Web page, including the tags for title, body, and headings.
- In the typical case, it is a mix of HTML or XML (which provide the static parts of the page) and PL/SQL (filling in the dynamic content).

You can author the Web pages in a script-friendly HTML authoring tool and drop the pieces of PL/SQL code into place. Embedding the PL/SQL code in the HTML page that you create lets you write content quickly and follow a rapid, iterative development process.

See Also: *Oracle8i Application Developer's Guide - Fundamentals* in the Oracle Database Documentation Library.

Perl Interpreter

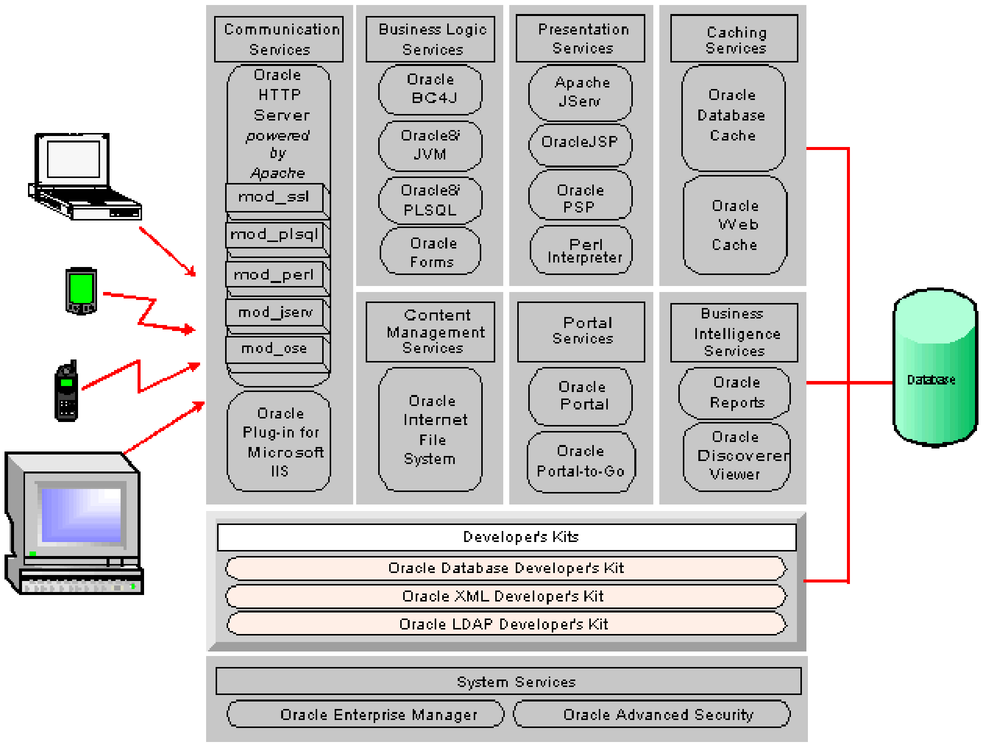
The Perl Interpreter is a persistent Perl runtime environment that is embedded in Oracle HTTP Server, thus saving the overhead of starting an external interpreter. When Oracle HTTP Server receives a Perl request, it is routed through mod_perl to the Perl Interpreter for processing.

See Also: *Apache mod_perl Documentation* in the Oracle9*i* Application Server Documentation Library.

Developer's Kits

To support application development and deployment, several toolkits containing libraries and tools are included in Oracle9*i* Application Server. Using the Developer's Kits, you can build Web applications that can access your database. These services are highlighted in Figure 2–8.

Figure 2–8 Developer's Kits in Oracle9i Application Server



Oracle Database Developer's Kit

Oracle Database Client Developer's Kit contains client libraries for Oracle8*i* and the Java client libraries (JMS, SQLJ, JDBC, and OCI).

Oracle Java Messaging Service (JMS) Toolkit

Oracle JMS extends the standard Sun Microsystems Java Message Service version 1.02 specification. In addition to the standard JMS features, Oracle JMS provides a Java API for Oracle Advanced Queuing (AQ). This API supports the AQ administrative operations and other AQ features including

- **an administrative API** to create queue tables, queues, and topics
- **point-to-multipoint communication**, which extends the standard point-to-point by using recipient lists for topics
- **message propagation between destinations**, which allows applications to define remote subscribers
- **transacted sessions** so you can perform JMS as well as SQL operations in one atomic transaction
- **message retention** after messages have been dequeued
- **message delay** so you can make messages visible after a specified time
- **exception handling** so messages can be moved to exception queues if they cannot be processed successfully
- **Oracle8*i* `AdtMessages`** message types (These are stored in the database as Oracle objects. Consequently, the payload of the message can be queried after it is enqueued. Subscriptions can be defined on the contents of these messages in addition to the message properties.)

See Also: *Oracle8i Application Developer's Guide - Advanced Queuing* in the Oracle Database Documentation Library.

Oracle SQLJ Translator

Oracle SQLJ is a preprocessor that developers can use to embed *static* SQL operations in Java code. A SQLJ program is a Java program that contains embedded static SQL statements which comply with the ANSI-standard SQLJ Language Reference syntax. Static SQL operations are predefined—the operations themselves do not change in real-time as a user runs the application, although the data values transmitted can change dynamically.

Oracle SQLJ consists of a *translator* and a *runtime component*. The translator replaces embedded SQL with calls to the SQLJ runtime component, which implements the SQL operations. In standard SQLJ, this is typically done through calls to a JDBC driver. In the case of Oracle9*i* Application Server, you use an Oracle JDBC driver. When users run SQLJ applications, the runtime component is invoked to handle the SQL operations.

Oracle SQLJ Translator The translator is a precompiler, which developers run after creating SQLJ source code. The translator checks

- syntax of the embedded SQL
- SQL constructs, which are compared to a specified database schema to ensure consistency within a particular set of SQL entities. For example, it verifies table names and column names. This feature is optional.
- datatypes, to ensure that the data exchanged between Java and SQL have compatible types and proper type conversions

The translator, written in 100% pure Java, supports a programming syntax that allows you to embed SQL operations inside SQLJ executable statements. SQLJ executable statements and SQLJ declarations are preceded by the `# sql` token and can be interspersed with Java statements in a SQLJ source code file.

The translator produces a Java source file and one or more SQLJ *profiles*, which are serialized Java resources that contain details about the embedded SQL operations in your SQLJ source code. After the translator produces the Java source file and the profiles, SQLJ automatically invokes a Java compiler to produce class files from the Java source file.

Oracle SQLJ Runtime This component is a thin layer of pure Java code that runs above the JDBC driver. When Oracle SQLJ translates your SQLJ source code, embedded SQL commands in your Java application are replaced by calls to the SQLJ runtime component. Runtime classes act as wrappers for equivalent JDBC classes, providing SQLJ functionality. When the user runs the application, the SQLJ runtime component acts as an intermediary, reading information about your SQL operations from the profile and passing instructions along to the JDBC driver.

A SQLJ runtime component can be implemented to use any JDBC driver or proprietary means of accessing the database cache or origin database. Oracle SQLJ runtime requires a JDBC driver but can use any standard JDBC driver. To use Oracle-specific database types and features, however, you must use an Oracle JDBC driver.

See Also: *Oracle8i SQLJ Developer's Guide and Reference* in the Oracle9*i* Application Server Documentation Library.

Oracle Java Database Connectivity (JDBC) Drivers

JDBC is a database access API that enables you to connect to a database and then prepare and execute SQL statements against the database. Core Java class libraries provide the interfaces and Oracle JDBC drivers implement those interfaces to access an Oracle database.

Oracle JDBC drivers are described in Table 2–2.

Table 2–2 Oracle JDBC Drivers

Driver	Description
JDBC Thin Driver	You can use the JDBC thin driver to write 100% pure Java applications that access Oracle SQL data. You can use it for EJBs in the database. The JDBC thin driver is especially well-suited to Web browser-based applications because you can dynamically download it from a Web page.
JDBC Oracle Call Interface Driver	The JDBC Oracle Call Interface (OCI) driver accesses Oracle-specific native code (that is, non-Java) libraries on the client or in the middle tier, providing a richer set of functionality and some performance boost compared to the JDBC thin driver, at the cost of significantly larger size.
JDBC Server-side Internal Driver	Oracle8*i* JVM uses the JDBC server-side internal driver when Java code runs on Oracle9*i* Application Server. It allows Java applications running in Oracle8*i* JVM to access locally defined data (that is, on the same machine and in the same process) with JDBC. It provides a further performance boost because of its ability to use underlying Oracle RDBMS libraries directly, without the overhead of an intervening network connection between your Java code and SQL data. This driver can also be used for EJBs in Oracle8*i* JVM.

See Also: *Oracle8i JDBC Developer's Guide and Reference* in the Oracle9*i* Application Server Documentation Library for detailed information on the Oracle JDBC drivers and Oracle extensions to the standard JDBC API.

Oracle Call Interface (OCI)

Oracle Call Interface is an application programming interface (API) that allows you to create applications that use the native procedures or function calls of a third-generation language to access an Oracle database server and control all phases of SQL statement execution. OCI supports the datatypes, calling conventions, syntax, and semantics of a number of third-generation languages including C, C++, COBOL, and FORTRAN.

OCI provides

- improved performance and scalability through the efficient use of system memory and network connectivity
- consistent interfaces for dynamic session and transaction management in a two-tier client/server or multitiered environment
- database authentication through the middle tier
- comprehensive support for application development using Oracle objects
- access to external databases
- applications that can service an increasing number of users and requests without additional hardware investments

OCI allows you to manipulate data and schemas in an Oracle database using a host programming language, such as C. It provides a library of standard database access and retrieval functions in the form of a dynamic runtime library (OCI library) that can be linked in an application at runtime.

OCI Components OCI components encompass these main sets of functionality:

- APIs to design a scalable, multithreaded application that can support large numbers of users securely
- SQL access functions, for managing database access, processing SQL statements, and manipulating objects retrieved from an Oracle database server
- Datatype mapping and manipulation functions, for manipulating data attributes of Oracle types

- Data-loading functions, for loading data directly into the database without using SQL statements
- External procedure functions, for writing C callbacks from PL/SQL

See Also: *Oracle Call Interface Programmer's Guide* in the Oracle Database Documentation Library.

Oracle XML Developer's Kit

Oracle XML Developer's Kit (XDK) contains XML (Extensible Markup Language) component libraries and utilities that you can use to enable applications and Web sites with XML. The XDK in the Oracle9*i* Application Server contains the components described in Table 2–3.

Table 2–3 XDK Components in Oracle9i Application Server

XDK Component	Description
XML Parser for Java	Parses XML documents or stand-alone Document Type Definitions (DTDs) so Java applications can process them. This parser uses industry standard Document Object Model (DOM) and Simple API for XML (SAX) interfaces. ▪ DOM APIs permit applications to access and manipulate an XML document as a tree structure in memory. This interface is used by such applications as editors. ▪ SAX APIs permit an application to process XML documents using an event-driven model.
XML Class Generator for Java	Automatically generates Java class source files from XML DTDs. Using these classes, Java applications can construct, validate, and print XML documents that comply with the input DTD. The class generator works in conjunction with the parser, which parses the DTD and passes the parsed document to the class generator.

Table 2–3 XDK Components in Oracle9i Application Server (Cont.)

XDK Component	**Description**
XML Transviewer JavaBeans	Displays and transforms XML documents and data through Java components to add graphical or visual interfaces to XML applications. The included beans are: ■ DOM Builder Bean: Encapsulates the Java XML Parser with a bean interface and extends its functionality to permit asynchronous parsing. By registering a listener, Java applications can parse large or successive documents having control return immediately to the caller. ■ TreeViewer Bean: Displays XML formatted files graphically as a tree. ■ SourceViewer Bean: Displays XML files with color syntax highlighting when modifying an XML document with an editing application. ■ XSL Transformer Bean: Transforms an XML document to other text-based formats, including HTML and DDL, by applying an XSL (Extensible Stylesheet Language) stylesheet.
XSQL Servlet	Processes SQL queries and outputs the result set as XML. This processor is implemented as a Java servlet and takes as its input an XML file containing embedded SQL queries.

See Also: Oracle XML Developer's Kit documentation in the Oracle9*i* Application Server Documentation Library.

Oracle LDAP Developer's Kit

This service supports client interaction with Oracle Internet Directory, the Oracle LDAP (Lightweight Directory Access Protocol) directory server. Oracle Internet Directory combines a native implementation of the Internet Engineering Task Force's (IETF) LDAPv3 standard with an Oracle8*i* back-end data store.

Specifically, you can use Oracle LDAP Developer's Kit to develop and monitor LDAP-enabled applications. It supports client calls to directory services, encrypted connections, and you can use it to manage your directory data. Oracle LDAP Developer's Kit contains the following components:

- **Oracle Internet Directory C API**

  This is an LDAP C API that is based on the Internet Engineering Task Force (IETF) RFC 1823. This component supports client calls to directory services from a standard C programming environment.

- **JNDI 1.2 (Java Naming and Directory Interface)**

  This programming interface from Sun Microsystems enables Java-based programs to communicate directly with LDAP servers such as Oracle Internet Directory.

- **SSL Toolkit**

  These extensions to the LDAP API enable one-way, two-way, and simple encrypted connections to directory services.

- **Oracle Internet Directory Command-Line Tools**

  This set of LDAP-compliant command line tools is used for querying and returning results from Oracle Internet Directory.

- **Oracle Directory Manager**

  This is a Java-based application used for managing data stored in Oracle Internet Directory.

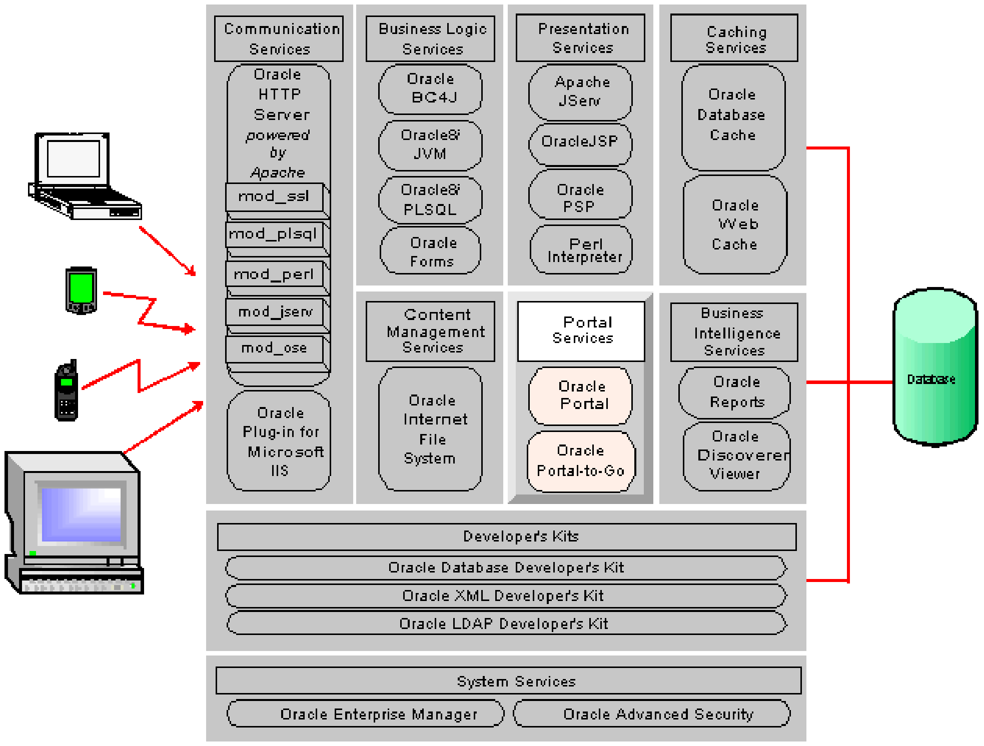
See Also:

- JNDI documentation at `http://java.sun.com/products/jndi`
- *Oracle Internet Directory Application Developer's Guide* in the Oracle9*i* Application Server Documentation Library for information about Oracle Internet Directory C API, SSL Toolkit, and Oracle Internet Directory Command-Line Tools.
- *Oracle Internet Directory Administrator's Guide* in the Oracle Database Documentation Library for information about Oracle Directory Manager.

Portal Services

You can use these services to build portal sites that integrate all of your content on a single Web page. Portal sites give your users a single, centralized, personalized view of relevant applications and data. By using Oracle9*i* Application Server Portal Services, you can make your portal sites accessible to both fixed and mobile clients. The major elements that support these services are highlighted in Figure 2–9.

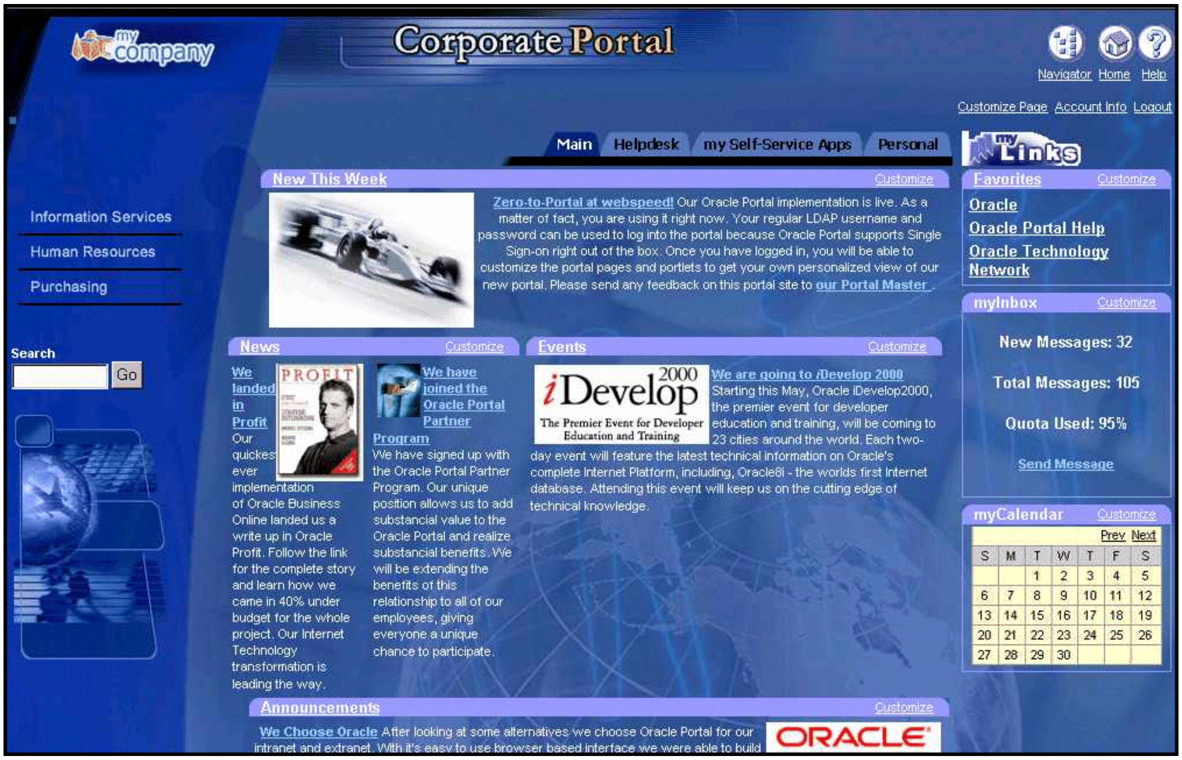
Figure 2–9 Portal Services in Oracle9i Application Server



Oracle Portal

Oracle Portal provides portal services for users connecting from a traditional desktop. An enterprise *portal* is a Web-based application that provides a common, integrated entry point for accessing dissimilar data types on a single Web page. For example, you can create portals that give users access to Web applications, business documents, business intelligence reports, graphics, and URLs which reside both inside and outside your corporate intranet. Figure 2–10 shows a sample portal page.

Figure 2–10 Sample Portal Page



Zero-to-Portal at webspeed! Our Oracle Portal implementation is live. As a matter of fact, you are using it right now. Your regular LDAP username and password can be used to log into the portal because Oracle Portal supports Single Sign-on right out of the box. Once you have logged in, you will be able to customize the portal pages and portlets to get your own personalized view of our new portal. Please send any feedback on this portal site to our Portal Master .

New Messages: 32

Total Messages: 105

Quota Used: 95%

Send Message

We landed in Profit Our quickest ever implementation of Oracle Business Online landed us a write up in Oracle Profit. Follow the link for the complete story and learn how we came in 40% under budget for the whole project. Our Internet Technology transformation is leading the way.

We have joined the Oracle Portal Partner Program We have signed up with the Oracle Portal Partner Program. Our unique position allows us to add substancial value to the Oracle Portal and realize substancial benefits. We will be extending the benefits of this relationship to all of our employees, giving everyone a unique chance to participate.

We are going to iDevelop 2000 Starting this May, Oracle iDevelop2000, the premier event for developer education and training, will be coming to 23 cities around the world. Each two-day event will feature the latest technical information on Oracle's complete Internet Platform, including, Oracle8i - the worlds first Internet database. Attending this event will keep us on the cutting edge of technical knowledge.

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

We Choose Oracle After looking at some alternatives we choose Oracle Portal for our intranet and extranet. With it's easy to use browser based interface we were able to build

Oracle Portal provides an organized, personalized view of business information, Web content, and applications. Its extensible framework integrates disparate sources of applications and content into dynamically assembled Web pages. Self-service publishing features allow users to post and share business documents and Web content. The flexible deployment architecture allows you to centrally administer all portal services through a browser.

Create and Administer a Single Point of Interaction with Enterprise Information

You can manage the user experience, creating and administering automatically generated portal pages that contain *portlets*. To users, portlets on a page appear as live areas of HTML that present text or graphics as links to information or applications. Portlets provide access to Web resources, such as applications, Web pages, or syndicated content feed. They are owned by *portlet providers*, which provide the communication link between the portal page and the portlets.

Each of the Web resources that the portlets provide access to can be personalized and managed as an Oracle Portal service. Portal administrators can manage portlets in the following ways:

- **unify and streamline access to information**: Portal administrators grant access to applications and information by adding portlets to a portal page. Then when users request the page, Oracle Portal calls the portlet provider, accepts the output, applies user page preferences, and assembles the completed Web page.
- **customize pages and page views**: Using page personalization features, portal administrators can create and manage community or departmental pages while maintaining a standardized corporate style. Group and individual privileges can be extended so that users can personalize a page and create unique page views.

Self-Service Web Publishing

Oracle Portal incorporates self-service features so that users and administrators can directly publish and manage their own information, such as

- **static content**: To publish and manage their own content, users can upload their files, use version control, format page display, and set access controls.
- **dynamic content**: Users with minimal development experience can build application components that display and interact with database data, such as Web forms, charts, or reports. When complete, these components can be published as a portlet.

Implement Custom Portlets

Custom portlets are used to access information that is both specific to an enterprise and accessible through the Web. To accommodate the diversity of these information sources and their underlying architectures, Oracle Portal offers two implementation strategies for creating custom portlets:

- **leverage built-in services**: The Oracle Portal self-service content management features provide an easy way to make content available within the portal. No coding is required to use the built-in services.
- **develop for public APIs**: Oracle Portal can be extended programmatically through the development, registration, and execution of custom portlets that are developed for the Oracle Portal public APIs. These portlets can be implemented as stored procedures in an Oracle database or as server-side logic written in any Web language, such as Java, Perl, C, or ASP.

Oracle Portal Architecture

Oracle Portal is designed around a three-tier architecture that scales to fit departmental or enterprise requirements. The three tiers are

- **database server tier**: The stored procedures that comprise Oracle Portal are executed within the Oracle8*i* database. All of the user content that is uploaded and accessed through Oracle Portal is stored on the database. Oracle Portal uses Oracle Login Server for all user authentication, user management, and single signon services.
- **application server tier**: Oracle Portal uses Oracle HTTP Server for middle-tier Web communication services. Oracle HTTP Server mod_plsql translates HTTP requests from client browsers into calls to the database. Oracle HTTP Server mod_jserv and mod_ssl are used to execute Java servlets and to support S-HTTP, respectively.
- **client tier**: A standard Web browser is required to develop, deploy, administer, and configure Oracle Portal.

Common Portal Services

Developers who choose to implement stored procedure portlets can use a set of API services in their portlet code, including:

- **context**: Access general session information about the user
- **session store**: Store and retrieve portlet-specific values and preferences for the current login session
- **preference store**: Store and retrieve portlet-specific preferences that are defined by the user
- **parameter passing**: Access parameters from the Web page URL
- **National Language Support**: Store multiple language strings to use as portlet text; retrieve strings based on browser settings
- **logging**: Record and retrieve actions performed on portlet objects
- **error handling**: Detect and react programmatically to errors during the execution of portlet objects
- **security and access control list**: Access security and authorization information about the user

See Also: Oracle Portal documentation in the Oracle9*i* Application Server Documentation Library.

Oracle Portal-to-Go

Oracle Portal-to-Go is a portal service for delivering information and applications to mobile devices. Using Portal-to-Go, you can create custom portal sites that use different kinds of content, including Web pages, custom Java applications, and XML applications. Portal sites make this diverse information accessible to mobile devices without you having to rewrite the content for each target device platform.

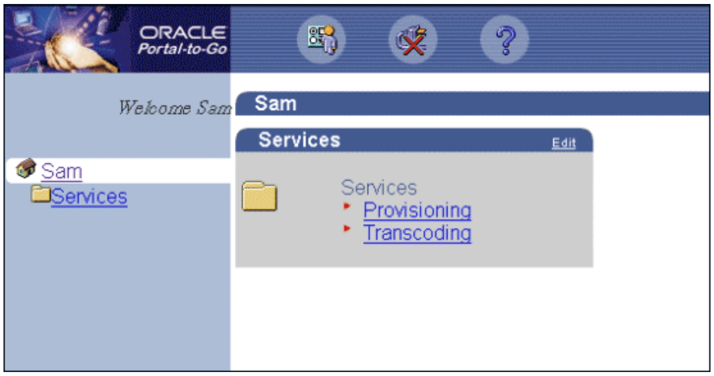
Portal-to-Go works by isolating content acquisition from content delivery. It provides an intermediary format layer, Portal-to-Go XML, between the source format and the target format. Portal-to-Go XML is a set of DTDs (Document Type Definitions) and XML document conventions used to define content and internal objects in Oracle Portal-to-Go.

Personalized User View

Users can configure their device portals from the Portal-to-Go Personalization Portal. The Personalization Portal is a Web-based interface that users access from their desktop computers. Using the Personalization Portal, users can store frequently accessed e-mail addresses, passwords, PINs, and account numbers.

Portal-to-Go uses JavaServer Pages, graphics, and stylesheets to generate the Personalization Portal user interface. By modifying the elements that Portal-to-Go uses to generate the Personalization Portal, users can add, rename, or move services and jobs. Then by using the customized interface, users can configure what services are available to access on their mobile devices. A sample of the Personalization Portal is shown in Figure 2–11.

Figure 2–11   Personalization Portal



Retrieve Content from Various Sources

Oracle Portal-to-Go uses adapters, which are dynamically loaded Java class files, to retrieve content from external sources. You can build your own adapters with the development tools that are included in Oracle Portal-to-Go, or you can use the following adapters that are provided:

- **provisioning adapter**: You can use this adapter to add, remove, or modify user accounts.
- **SQL adapter**: You can use this adapter to create services that query databases, invoke PL/SQL procedures, or call stored procedures in the database.
- **servlet adapter**: You can use this adapter to integrate applications that use servlets so they can be called as Portal-to-Go services, making them available to mobile devices.
- **Web integration adapter**: You can use this adapter to retrieve and adapt Web content. The Web Integration adapter works with Web Interface Definition Language (WIDL) files to map source content to Portal-to-Go XML.
- **stripper adapter**: You can use this adapter to dynamically retrieve and convert the content of a URL target. Unlike the Web Integration adapter, the stripper adapter dynamically processes the markup tags in the content, either removing them or leaving them intact. However, you can extend this adapter to process the tags in other useful ways that fit your implementation.
- **URL adapter**: You can use this adapter to access any content in Portal-to-Go XML through a URL, making it easy to integrate different types of applications.

Supported Target Device Formats

Portal-to-Go transformers convert content, which has been retrieved by adapters, into the appropriate format for the target device. Transformers can be either a Java class file or an XSLT (Extensible Stylesheet Language Tranformations) stylesheet. Like adapters, you can create your own transformers with the development tools that are included in Oracle Portal-to-Go, or you can use the transformers that are provided to convert content into the following target formats:

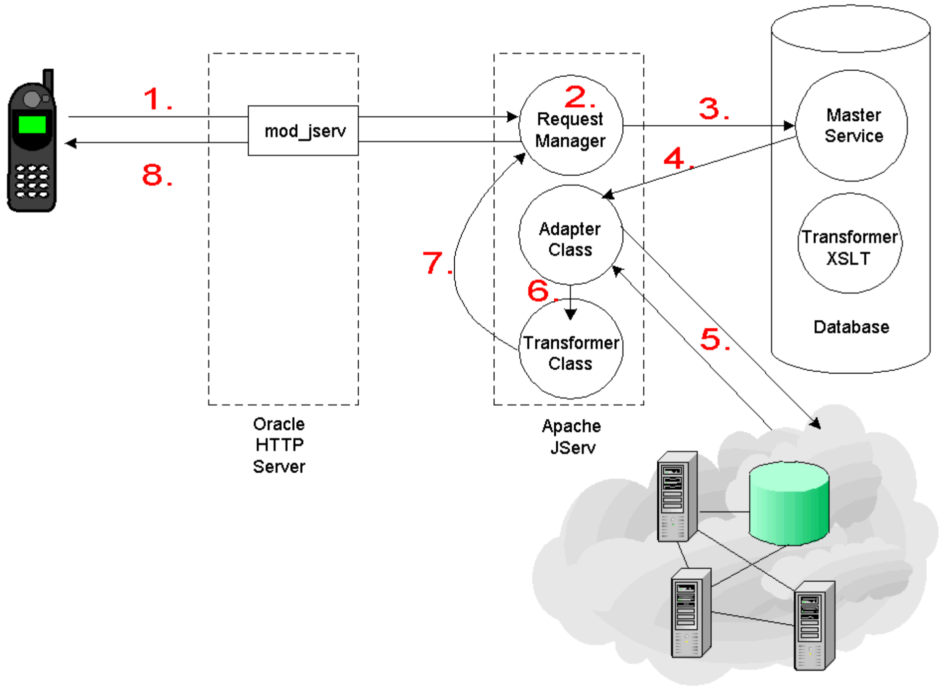
- **WML 1.1**: The Wireless Markup Language that is defined by the WAP (Wireless Access Protocol) Forum. Portal-to-Go provides transformers for a general WML implementation, as well as for device-specific WML implementations.
- **Tiny HTML**: A minimal HTML implementation suitable for handheld devices, such as Palm Computing Platform and Windows CE devices.

- **VoxML/VoiceXML**: The Motorola markup language that enables voice interaction with applications.
- **HDML**: The Handheld Devices Markup Language is a simplified version of HTML designed specifically for handheld devices.
- **Plain Text**: The plain text transformer converts content for Short Message Service devices and e-mail applications.

Oracle Portal-to-Go Architecture

In Oracle9*i* Application Server, Portal-to-Go uses the Oracle HTTP Server listener, Apache JServ, and OracleJSP to deliver content in the appropriate format to target mobile devices. The Portal-to-Go request and response sequence in Oracle9*i* Application Server is shown in Figure 2–12.

Figure 2–12 Portal-to-Go Request and Response Sequence



1. When users request information with their mobile devices, the request is received by Oracle HTTP Server and routed through mod_jserv to the Portal-to-Go request manager in Apache JServ.
2. The request manager performs initial preprocessing, including user authentication.
3. If the user is authenticated, then the request manager creates a request object, which it forwards to the master service associated with the request.

   A master service is an XML file that is stored in the Portal-to-Go repository, located in the database. It describes which adapter class file to invoke to service the request.
4. The master service invokes the appropriate adapter class file in Apache JServ.
5. The adapter retrieves the content from the appropriate source, which can be a database, Web site, application, e-mail server, or other mobile device.
6. The adapter returns the content in Portal-to-Go XML and forwards it to the transformer.
7. The transformer converts the content into the format appropriate for the target device and forwards it to the request manager.

   Note: If the transformer is a Java class file, then it resides in Apache JServ. If it is an XSLT stylesheet, then it resides in the Portal-to-Go repository in the database.

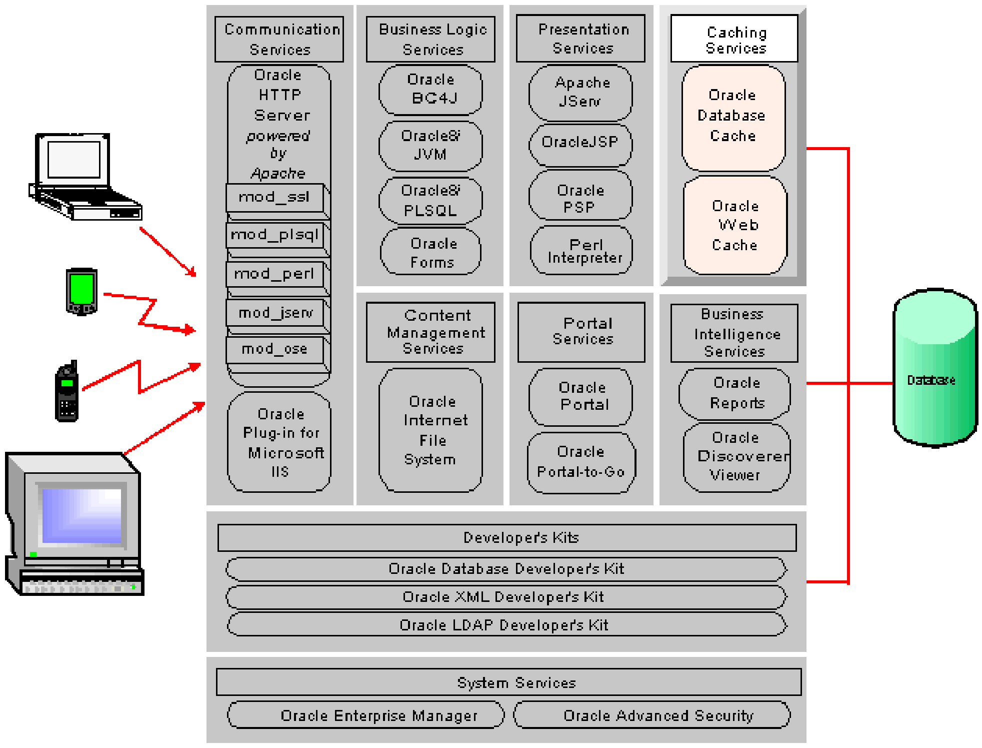
8. The request manager returns the information to the user.

   See Also: Oracle Portal-to-Go documentation in the Oracle9*i* Application Server Documentation Library.

Caching Services

To scale your e-business Web site, Oracle9*i* Application Server provides caching services. Caching services include Oracle Web Cache and Oracle Database Cache. Oracle Web Cache is a content-aware server accelerator that improves the performance, scalability, and availability of Web sites by caching both static and dynamic pages. Oracle Database Cache is a middle-tier database cache that reduces the load on the back-end database instance by caching frequently requested data, avoiding unnecessary network roundtrips for read-only data. These services are highlighted in Figure 2–13.

Figure 2–13 Caching Services in Oracle9i Application Server



Oracle Database Cache (formerly Oracle8*i* Cache)

Oracle Database Cache is a database cache that resides on the middle tier as a service of Oracle9*i* Application Server. It improves the performance and scalability of applications that access Oracle databases by caching frequently used data on the middle-tier machine. With Oracle Database Cache, your applications can process several times as many requests as their original capacity because roundtrips to the back-end database are greatly reduced. Oracle Database Cache service supports running stateful servlets, JavaServer Pages, Enterprise JavaBeans, and CORBA objects in Oracle8*i* JVM.

Who Should Use Oracle Database Cache?

If your applications meet the following criteria, then you can use Oracle Database Cache to boost the scalability of your Web sites and the performance of your applications:

- Your applications access an Oracle database.
- You use a multitiered environment, where the clients, Oracle9*i* Application Server, and Oracle database servers are located on separate systems.
- Your applications communicate with an Oracle database through Oracle Call Interface (OCI), or an access layer built on OCI, like JDBC-OCI, ODBC, or OLE. The applications can be written using scripting or programming languages.
- Your Web or application users generate mostly read-only queries.
- Your applications can tolerate some synchronization delay with data in the origin database. That is, the cache does not need to be as up-to-date as the data in the origin database. (You decide how often to refresh the data.)

Performance and Scalability Benefits

Using Oracle Database Cache in the middle tier provides a number of performance and scalability benefits. The most significant benefits are listed below.

- Processing database queries on the middle tier reduces time spent sending and receiving data over the network so hardware resources are used more efficiently.
- Reducing the load on the back-end database means that it can support more users.

Oracle Database Cache Environment

In the Oracle Database Cache environment, the cache software consists of a middle-tier database cache for caching frequently accessed data and running intelligent software that routes queries. The origin database is on the back-end. It is the original and primary storage for the data. Currently, Oracle Database Cache can cache data from only one origin database.

Example of How Requests are Routed to Oracle Database Cache When users request frequently accessed data, the requests pass from the client to Oracle Database Cache, through Oracle HTTP Server, and the database cache returns the data. For example, assume that you cache information about books, such as the book title, author, description, and ranking in a best-seller list, in the middle-tier database cache. If User A and User B both request the list of the top ten mystery books, then each user's request is routed to the middle-tier database cache. Because the data is stored on the same tier, the data is returned quickly and the request does not need to pass to the origin database server for retrieval of the data. If each user clicks on one of the titles to get more detailed information about the book, then the query is again routed to the middle-tier database cache, which returns the data quickly.

How Oracle Database Cache Differs from an Origin Database

Although Oracle Database Cache looks like an origin database placed on the middle-tier, there are important differences:

- Oracle Database Cache can cache data, but it is not a persistent store for data. You cannot perform backup and recovery on it. If you need to back up your data, then you must store it on the origin database.
- Oracle Database Cache cannot be used to create tables. The only way you can put data into this environment is by caching data from the origin database to the middle-tier system.
- Oracle Database Cache does not provide transparent application failover (TAF) guarantees if it fails because Oracle Database Cache is not a true persistent store for data.
- Oracle Database Cache should not be treated as an enlisted database in global transactions to avoid an unnecessary and expensive two-phase commit. For more information about transaction handling with Oracle Database Cache, see "Global Transactions Including Oracle Database Cache and the Origin Database" on page 3-20.

How Do Applications Use Oracle Database Cache?

To take advantage of the benefits of Oracle Database Cache, you only need to configure the environment of your applications. You do not need to make any modifications to your applications if they

- use SQL statements to access the database, or
- are linked with OCI using dynamic libraries and are layered directly on OCI.

If your applications satisfy either criteria, then queries are routed to the middle-tier database cache automatically.

Note: Applications must use OCI in threaded mode. If they do not, then refer to the Oracle Database Cache documentation in the Oracle9*i* Application Server Documentation Library.

Applications that are linked with OCI using static libraries and layered directly on OCI, require that you link your application with the OCI library that ships with Oracle Database Cache.

See Also: Oracle Database Cache documentation in the Oracle9*i* Application Server Documentation Library.

Oracle Web Cache

Oracle Web Cache is a server accelerator caching service that improves the performance, scalability, and availability of busy e-business Web sites that run on Oracle9*i* Application Server and Oracle8*i*. By storing frequently accessed URLs in virtual memory, Oracle Web Cache eliminates the need to repeatedly process requests for those URLs on the Web server. Unlike legacy proxy servers that only handle static images and text, Oracle Web Cache caches both static and dynamically generated HTTP content from one or more application Web servers. Using Oracle Web Cache, Web clients experience faster content retrieval and the load on Oracle9*i* Application Server is greatly reduced.

Oracle Web Cache Main Features

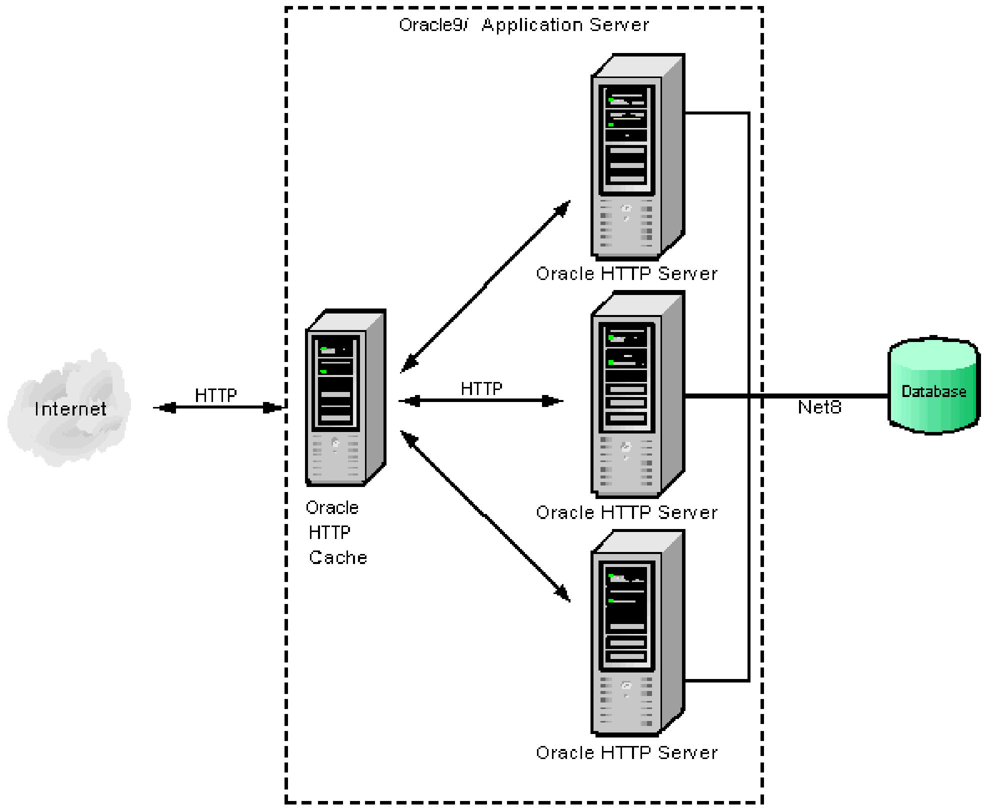
The main features of Oracle Web Cache make it a perfect caching solution for e-business Web sites that host online catalogs, news services, and portals:

- **static and dynamically generated content caching**: Caches documents according to rules that you specify
- **cache invalidation**: Supports invalidation as a mechanism to keep Oracle Web Cache contents consistent with the content on Oracle9*i* Application Server
- **performance assurance**: Built-in performance assurance heuristics handle performance issues while maintaining cache consistency. These heuristics assign a queue order to documents and determine which documents can be served stale and which must be retrieved immediately.
- **surge protection**: Allows you to set a limit on the number of concurrent requests that are passed to Oracle Web Cache to avoid overloading it.
- **load balancing**: Evenly distributes requests over many Oracle HTTP Servers that Oracle Web Cache cannot serve. Oracle Web Cache is designed to manage HTTP requests for up to 100 Oracle9*i* Application Server machines.
- **backend failover**: Automatically redistributes the load over the remaining Web servers when one server fails. When the failed server returns to operation, Oracle Web Cache automatically includes it in the load mix.
- **session tracking**: Supports Web sites that use session tracking to track users.
- **security**: Provides password authentication for administration and invalidation operations, control over which ports administration and invalidation operations can be requested from, and timeout for inactive connections.

Oracle Web Cache Architecture Overview

Oracle Web Cache is positioned in front of Oracle HTTP Servers to cache their content and to provide that content to Web browsers that request it. When Web browsers access your Web site, they send HTTP requests to Oracle Web Cache, which acts as a virtual server to the Oracle HTTP Servers. If the requested content has changed, then Oracle Web Cache retrieves the new content from an Oracle HTTP Server. Figure 2–14 shows the basic architecture of Oracle Web Cache.

Figure 2–14 Oracle Web Cache Architecture

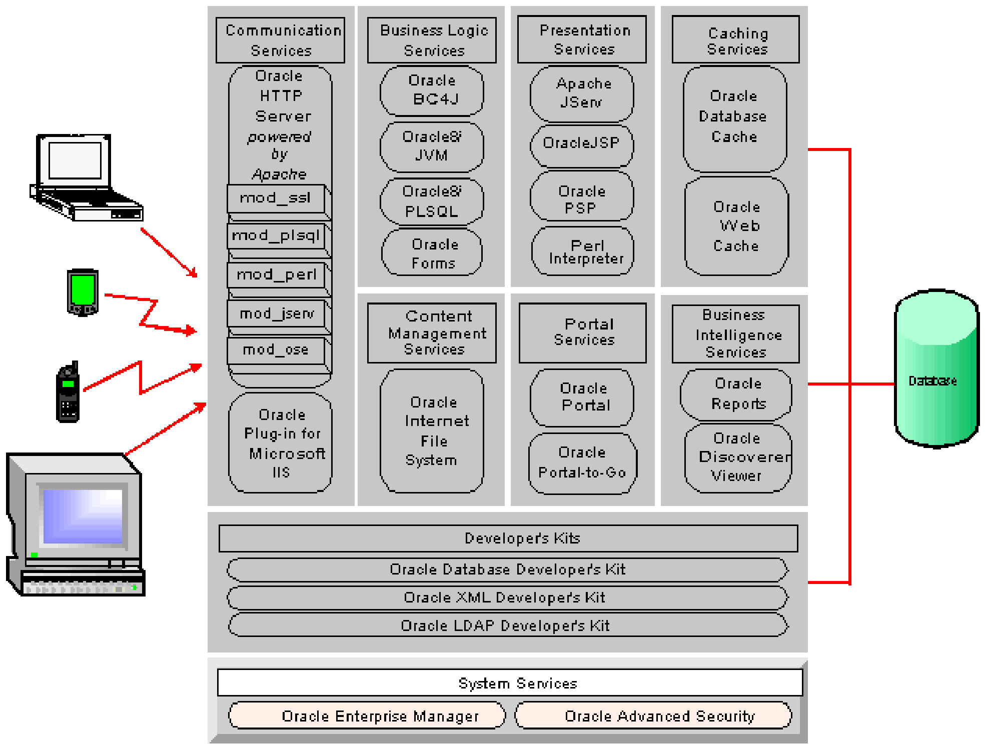


See Also: Oracle Web Cache documentation at `http://otn.oracle.com/product/ias`

System Services

To provide system management and security services, Oracle9*i* Application Server includes Oracle Enterprise Manager and Oracle Advanced Security. These services manage your Oracle environment and network security through encryption and authentication that uses SSL (Secure Sockets Layer). The major elements that support these services are highlighted in Figure 2–15.

Figure 2–15 System Services in Oracle9i Application Server



Oracle Enterprise Manager

Oracle Enterprise Manager is a system management tool that provides an integrated solution for centrally managing your Oracle platform. Combining a graphical console, Oracle Management Servers, Oracle Intelligent Agents, common services, and administrative tools, Oracle Enterprise Manager provides a comprehensive systems management platform for managing your Oracle products.

In Oracle9*i* Application Server, you use Oracle Enterprise Manager console (a graphical interface) to manage Oracle Database Cache, Oracle Forms Services, and the host operating system.

From Oracle Enterprise Manager console, you can perform the following tasks:

- centrally manage, administer, and diagnose Oracle Database Cache and Oracle Forms Services
- effectively monitor and respond to the status of your Oracle products and third-party services
- schedule activities on multiple machines at varying time intervals
- monitor networked services for events
- customize your display by organizing your server components and services into logical administrative groups

Three-Tier Architecture Oracle Enterprise Manager architecture consists of a three-tier framework, which is described in Table 2–4.

Table 2–4 Oracle Enterprise Manager Three-Tier Architecture

Tier	Description
First-tier Console	Provides a graphical interface for administrators. This tier includes Cache Manager console for managing Oracle Database Cache. For information about Cache Manager console, see the *Oracle8i Concepts and Administration Guide* on your Documentation Library CD-ROM.
Second-tier Management Servers[1]	Provides a scalable middle tier for processing all system management tasks.
Third-tier Intelligent Agents[2]	Monitors databases and services on each machine, and executes tasks received from the Management Server.

[1] Only available with Oracle9*i* Application Server Enterprise edition.

[2] Intelligent Agents support Simple Network Management Protocol (SNMP), enabling third-party applications to communicate with the agent and be managed along with Oracle services.

Note: For this release, Oracle Database Cache and Oracle Forms are the only services in Oracle9*i* Application Server that you can manage using Oracle Enterprise Manager. If you do not plan to use Oracle Database Cache or Oracle Forms Services, then you are not required to deploy Oracle Enterprise Manager.

Benefits of Oracle Enterprise Manager This system management tool enables administrators to maintain the highest level of performance and availability while controlling system management costs. The major benefits of Oracle Enterprise Manager are

- single point of management
- multi-administrator system support
- scalability for growing, distributed environments
- extensible architecture
- automated lights-out administration
- autonomous Intelligent Agents
- ease of use

See Also:

- Oracle Enterprise Manager documentation in the Oracle9*i* Application Server Documentation Library for information about Oracle Enterprise Manager.
- Oracle Database Cache documentation in the Oracle9*i* Application Server Documentation Library for information about Cache Manager.

Oracle Advanced Security

Oracle Advanced Security provides a comprehensive suite of security services for Oracle Database Cache, Oracle8*i* JVM, and Oracle8*i* PLSQL. Its functionality is twofold. First, network security features protect enterprise networks and securely extend corporate networks to the Internet. Second, it integrates security and directory services, combining to provide enterprise user management and single signon.

Network Security

- **Data Privacy**

  Oracle Advanced Security ensures that data is not disclosed during transmission using the following encryption types:

  – RSA Encryption: Oracle Advanced Security provides RSA RC4 with 128-bit, 56-bit, and 40-bit keys.

  – DES Encryption: Oracle Advanced Security offers a standard, optimized 56-bit key DES (Data Encryption Standard) encryption algorithm and supports DES40 for backwards compatibility.

- **Data Integrity**

  Oracle Advanced Security makes it virtually impossible for an intruder to modify, delete, or replay packets without detection. During transmission, Oracle Advanced Security generates a cryptographically secure message digest through cryptographic checksums using the MD5 algorithm. Then, the secure message digest is included with each packet sent across the network.

- **Authentication**

  Oracle Advanced Security provides strong authentication of Oracle users support for third-party authentication services. The following authentication methods are supported:

  - SSL with X.509v3 certificates
  - Kerberos and CyberSafe
  - Token Cards (SecurID- or RADIUS-compliant)
  - DCE (Distributed Computing Environment)
  - RADIUS
  - Smart Cards (RADIUS-compliant)
  - Biometrics (Identix- or RADIUS-compliant)

- **Single Signon**

  Oracle Advanced Security provides single signon where the user authenticates once, then strong authentication occurs transparently in subsequent connections to other databases or services. Using single signon, users can access multiple accounts and applications with a single password. Oracle Advanced Security supports many forms of single signon, including Kerberos and CyberSafe, as well as SSL-based single signon.

- **Authorization**

  Once users are authenticated, they are then authorized to access only those services permitted by a corporate or business policy as found in a policy repository. Authorizations are provided with some of the third-party authentication solutions, such as DCE (Distributed Computing Environment), as well as with the enterprise user security functionality in Oracle Advanced Security.

Enterprise User Security

Oracle Advanced Security integrates with LDAP v3-compliant (Lightweight Directory Access Protocol) directory services, such as Oracle Internet Directory, for enterprise user management, enterprise role management, and single signon.

- **Single Signon**

  Oracle Advanced Security provides SSL-based (Secure Sockets Layer) single signon for Oracle users by virtue of integration with LDAP v3-compliant directory services. Integrated security and directory services and Oracle PKI (Public Key Infrastructure) implementation in Oracle Advanced Security enable SSL-based single signon to Oracle8*i* databases. Single signon enables users to authenticate once at the initial connection and subsequent connections authenticate the user transparently based on his or her X.509 certificate. This brings ease of use to the users and single station administration for the administrator with centralized management of users and authorizations.

- **Enterprise User Management**

  Oracle Advanced Security provides enterprise user management, allowing administrators to centrally manage users on a central directory service, rather than repeatedly managing the same users on individual databases. Using Oracle Enterprise Security Manager, a tool accessible through Oracle Enterprise Manager, enterprise users and their authorizations are managed in Oracle Internet Directory or other LDAP v3-compliant directory services. Enterprise users can be assigned enterprise roles that determine their access privileges in a database, and enterprise roles can be granted to one or more enterprise users.

- **Schema-Independent Users**

  Oracle Advanced Security allows the separation of users from schemas so that many enterprise users can access a single, shared application schema. Instead of creating a user account in each database that a user needs to access, administrators only need to create an enterprise user in the directory and point the user at a shared schema, which many other enterprise users can also access. This allows administrators to create an enterprise user once in the directory. Then that enterprise user can access multiple databases using only the privileges he or she needs, thus lowering the overhead of managing users in an enterprise.

- **PKI Credential Management**

  Oracle Wallet Manager provides secure management of PKI user credentials. It issues certificate requests to Certificate Authorities (CA), manages the X.509 certificates and trusted certificates, and creates a private and public key pair for users. In most cases, a user never needs to access a wallet once it has been configured, but can easily access a wallet using Oracle Enterprise Login Assistant, a login tool that hides the complexity of a private key and certificates from users. Users can then connect to multiple services over SSL without providing additional passwords. This provides the benefit of strong, certificate-based authentication as well as single signon.

- **Directory Integration**

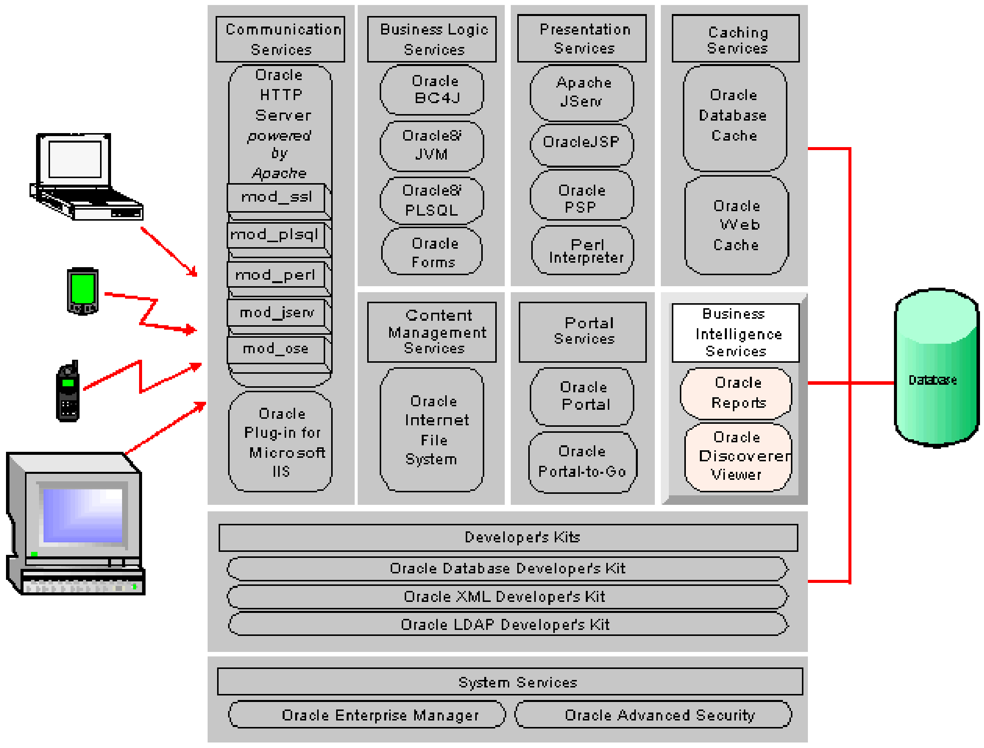
  An Oracle Advanced Security license provides the use of Oracle Internet Directory to store and manage users and their authorizations. It supports enterprise user management with Oracle Internet Directory, which is fully integrated with Oracle8*i*. Additionally, Oracle Advanced Security supports other leading LDAP-compliant directories.

  See Also: *Oracle Advanced Security Administrator's Guide* in the Oracle Database Documentation Library.

Business Intelligence Services

To understand what is happening in your business and on your Web site every day, you can use these services to deploy and share business intelligence over the Web or over your corporate intranet. These services are highlighted in Figure 2–16.

Figure 2–16 Business Intelligence Services in Oracle9i Application Server



Oracle Reports Services

By using Oracle Reports Services and its Reports Servlet services, you can run new and existing Oracle Reports Developer reports on an internal company intranet, an external company extranet, or the Internet. Oracle Reports Services is optimized to deploy Oracle Reports Developer applications (Reports and Graphics) in a multitiered environment. It consists of the server component, runtime engines, and the servlet runner.

In Oracle9*i* Application Server, when a client submits a request for a report, the Oracle HTTP Server Web listener routes that request to the Oracle Reports Services server component. The server routes the request to the Oracle Reports Services runtime engine, which runs the report. Then the report output is sent back to the client through the Oracle HTTP Server Web listener.

See Also: Oracle Reports Services documentation in the Oracle9*i* Application Server Documentation Library.

Oracle Discoverer 3*i* Viewer

Oracle Discoverer 3*i* Viewer is an environment for running and viewing Oracle Discoverer workbooks (reports) over the Web that have been created with Oracle Discoverer 3*i* Plus. By using Discoverer Viewer, Web authors can access database information and embed it in their sites without being database experts. They can publish live reports to Web sites by creating a URL that indicates to Discoverer Viewer which workbooks to open. Clicking the URL invokes the workbook query to the database and returns live results to the browser. Users interact with the query results to show more or less detailed information, to enter values into parameters, or to follow links to other applications.

Note: Currently, Oracle Discoverer 3*i* Viewer is available in the Windows NT and SPARC Solaris versions of Oracle9*i* Application Server.

By using Discoverer Viewer, you can

- run workbooks that have been saved dynamically in the database
- print workbooks
- export workbooks to various file formats
- perform drill-down analysis
- drill out to data held in other applications, such as Web pages or business documents
- leverage the security features of Oracle HTTP Server and the database (supports SSL, X.509 certificates, and other standard Web security protocols)
- embed workbooks into portals, such as Oracle Portal

Architecture Overview

Discoverer Viewer supports an HTML client and consists of two pieces:

- Discoverer Viewer servlet
- Discoverer Service

When users request Oracle Discoverer workbooks, the Viewer servlet, which runs on Oracle HTTP Server (Apache JServ servlet engine), interprets the HTTP request and makes the necessary calls to the Discoverer Service. The Discoverer Service response is represented in XML (Extensible Markup Language), which the servlet generates. Then the XML response is sent to an XML/XSL (Extensible Stylesheet Language) processor. This combines the XML with XSL configuration files that define the user interface. The processor then generates HTML to send back to the client browser. The Discoverer Viewer user interface can be customized for individual sites by defining or editing the XSL configuration files.

The Discoverer Viewer servlet and Discover Service can both run on the same system (with Oracle HTTP Server), if necessary. However, they are designed to be deployed on separate or multiple computers so that Oracle HTTP Server can be replicated using standard Web farm techniques. If you deploy the Discoverer Viewer servlet and Discoverer Service on separate computers, then Discoverer Service can be replicated and Discoverer Service sessions can be started on alternate systems to balance the load. This allows your implementation to support large numbers of Discoverer Viewer users.

Workbook Accessibility

Because Discoverer Viewer is an application built with standard Web technologies, all the usual features of your browser are available. Favorite workbooks can be bookmarked or embedded in other Web pages. Font sizes and link styles can be changed simply by changing your browser options. Discoverer Viewer uses no Java, no JavaScript, and no frames, so any browser can be used.

See Also:

- Oracle Discoverer 3i Viewer documentation in the Oracle9*i* Application Server Documentation Library.
- Chapter 3, "Developing Applications for Oracle9i Application Server" for information about using Oracle9*i* Application Server services.

3 Developing Applications for Oracle9*i* Application Server

Oracle9*i* Application Server provides several options for developing and deploying applications by using various programming languages and communication protocols.

This chapter demonstrates how you can use Oracle9*i* Application Server to build your applications through the following topics:

- Content Publishing
- Business Logic

Content Publishing

Static Content

The simplest Web sites consist of static pages where a client request returns HTML content that does not change after its initial display. Typically, a simple Web site identifies an HTTP request, responds by sending the requested content to the client, and returns a resulting response to the Oracle HTTP Server *powered by Apache*.

See Also: Oracle HTTP Server documentation in the Oracle9*i* Application Server Documentation Library.

Dynamic Content

Usually, the most interesting content requires more complex functionality, which typically consists of dynamic content delivered from the server. Server-side components run on the server and generate output which is then sent to the client browser. Oracle9*i* Application Server supports a variety of technologies for developing and deploying dynamic content, including:

- Common Gateway Interface (CGI) Applications
- Perl Scripts
- XML
- Servlets
- JavaServer Pages
- PL/SQL Server Pages
- Oracle Reports Services

Common Gateway Interface (CGI) Applications

Web content developers can write CGI programs that fetch data and produce entire Web pages within the same application. These applications typically contain scripts that mix application logic with presentation logic. Application logic manipulates the data while presentation logic formats the content. The separation of HTML content from application logic makes script based applications, such as CGI applications, easier to develop, debug, and maintain.

There are two categories of scripting applications: client-side and server-side scripting. Client-side scripts run in the client's Web browser. Server-side scripts run on the server, fetching and manipulating data. The resulting data is embedded in the HTML page, which is then sent to the client's Web browser.

In Oracle9*i* Application Server, the Oracle HTTP Server uses mod_cgi to receive requests for a Common Gateway Interface (CGI) application, the server invokes an operating system shell that runs the application and uses the CGI to deliver any accompanying data to the application. Every time the server receives a request for a CGI application, it starts a new process to run the application.

Oracle9*i* Application Server fully supports CGI applications, providing the middle-tier environment to meet the demands for performance and scalability when running Web applications. However, CGI applications are most useful for processing simple forms on a small scale.

Perl Scripts

Perl is an interpreted language with powerful text processing capabilities, which makes it ideal for parsing requests from clients and generating dynamic HTML. Perl scripts contain the logic to produce the dynamic portions of Web pages that run as requested by a client Web browser.

The Perl Interpreter embedded in the Oracle HTTP Server provides a performant, internal interpreter for running Perl scripts. The Perl Interpreter receives and runs Perl scripts through mod_perl, an Oracle HTTP Server module that delegates the handling of HTTP requests to run Perl programs. Using mod_perl, the interpreter links directly with the Oracle HTTP Server daemons, eliminating outside network communication and reducing access time.

Upon receiving a request, the interpreter loads, compiles, and caches the Perl script. The script remains in the cache as long as the server remains running, so subsequent requests for the same script do not have to recompile. If the script has been modified in any way, then the cache purges the original script and compiles and stores a fresh copy of the modified script.

See Also:

- Apache Software Foundation documentation at `http://www.apache.org` for general information about Perl and the Perl module in Oracle HTTP Server.
- Apache mod_perl documentation in the Oracle9*i* Application Server Documentation Library.

XML

XML (Extensible Markup Language) is a flexible and standard way to create common document formats for building and deploying Web content. As with CGI scripts, XML data separates the content and structure from the presentation, making it easy to present the data in a variety of layouts and applications.

Oracle9*i* Application Server allows you to generate XML on the middle tier, invoke server-side components to access the database, run applications, process the information, and deliver the content to the client Web browser. The Oracle XML Developer's Kit (XDK) provides support for reading, manipulating, transforming, and viewing XML documents for Java applications.

The Oracle XDK includes a set of XML parsers to process XML documents for Java. The parser's job is to verify that the XML is well-formed and to validate the XML by comparing it to an existing Document Type Definition (DTD). After this verification and validation phase, the parser manipulates the XML documents into a useful format for applications. Oracle also provides several products that support automatic generation of classes from DTD elements and ways to programmatically use class methods to construct XML documents.

XML also gives you a common format for transferring data between databases. You can generate XML from multiple databases to a common, extensible XML document type. You can use XML to give context to words and values, identifying elements as data. For example, you can identify elements for data such as names, addresses, and contact information. When you collect data from several sources using these common elements, you can easily integrate the data into a common format for a specific application.

See Also: Oracle XML Developer's Kit documentation in the Oracle9*i* Application Server Documentation Library.

Servlets

Servlets are Java applications that run in a server-side environment and service HTTP requests from client browsers. While servicing client requests, servlets can use common Java technologies to increase their functionality. For example, a servlet using JDBC can connect to a database and execute a query. The servlet can then format the result of the query in an HTML table and return it to the client.

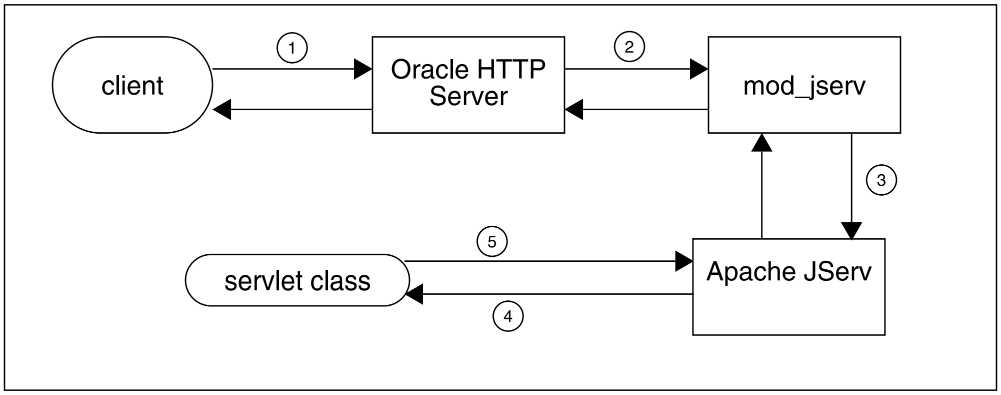
Advantages of Using Servlets The ability to use existing Java code and standard APIs makes servlets a powerful tool for servicing requests. Other advantages of using servlets are

- servlets, as a Java-based technology, are platform and Oracle HTTP Server independent. A servlet that is compliant with Sun's servlet specification can run on any Oracle9*i* Application Server node without any modifications.
- servlets often perform better than CGI applications because requests can reuse an existing Java Virtual Machine (JVM). This saves system resources that can otherwise be used to start up and shut down the JVM for each request.
- database connections can remain open for a session reducing the overhead of reconnecting to the database with each request.
- free servlet instances are reused to service new requests instead of creating new instances.

Developing Servlets Developing servlets requires a compiler, a debugger, and access to the standard and servlet Java libraries. These libraries are installed with Oracle9*i* Application Server. You can also use integrated development environments, such as Oracle JDeveloper, which provide a set of tools that help in developing and deploying servlets.

Processing Servlet Requests In Oracle9*i* Application Server, servlets run in the Oracle HTTP Server component. The mod_jserv module forwards requests from the Oracle HTTP Server to Apache JServ. Apache JServ is the Java servlet engine designed to service servlet requests from the Oracle HTTP Server. The JVM processes the servlet and returns the output to mod_jserv. The mod_jserv module then returns the response to the client. This process is illustrated in Figure 3–1.

Figure 3–1 Control Flow for a Servlet in the Oracle9i Application Server



1. The Oracle HTTP Server receives a request for a servlet from a client.
2. The Oracle HTTP Server dispatches the request for a servlet class to the mod_ jserv module.
3. The mod_jserv module forwards the request to the servlet engine. If the engine is not already running, then mod_jserv will spawn one and initialize the servlet.
4. The servlet engine translates the incoming request into a request object. The engine then creates a response object and passes both objects to the servlet class.
5. The servlet class executes and generates response data. This data is passed to the servlet engine in the servlet response object. The response then travels back to the client through the mod_jserv module and Oracle HTTP Server.

See Also:

- Java Servlet API Specification version 2.0 at `http://java.sun.com`
- Apache JServ documentation in the Oracle9*i* Application Server Documentation Library.

JavaServer Pages

JavaServer Pages (JSP), as specified by Sun Microsystems, offer an easy to use and convenient method for developers to add dynamic content to an HTML file by using Java code and some special tags. The ability to use existing Java code and standard APIs make JSP a powerful tool for generating dynamic HTML pages. This allows Java developers to simplify their applications because presentation logic can be handled by the JSP. To use a new presentation method, such as a new HTML tag, you only change your JSP. The underlying Java code will not need any modifications since it only delivers the data to the JSP.

Advantages of Using JSP Since JavaServer pages are platform independent, any JSP that is compliant with the Sun Microsystems JSP specification can run on any Oracle9*i* Application Server node without modification.

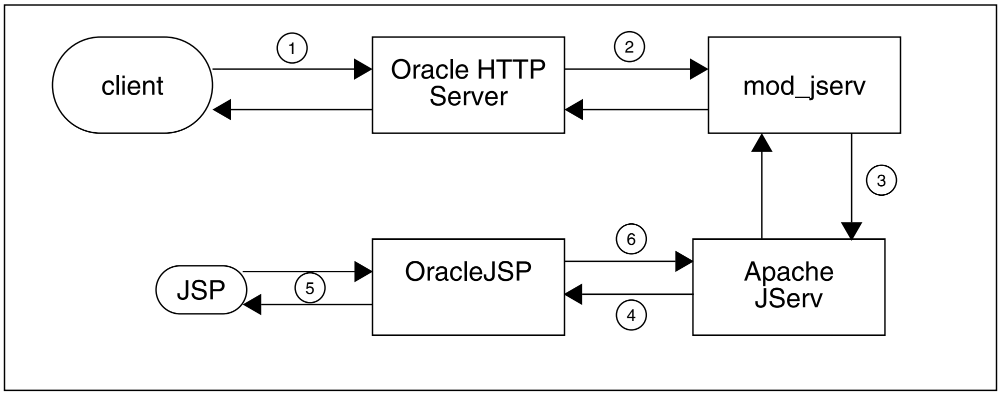
OracleJSP, the Oracle translator and runtime engine for JavaServer Pages, supports the following enhancements available to JSP developers:

- More data types to increase the functionality of embedded logic.
- Two options, primitive types (PT) or java.lang objects. Only objects can be stored in the scope object, therefore, values of PT cannot have a valid scope and cannot be stored in a JSP scope object. Wrapper classes in the java.lang package are objects that cannot be declared in a JSP usebean statement because the wrapper classes do not follow the JavaBean model and do not provide a null constructor.
- JSP Markup Language (JML) offers an alternative to the JSP scripting syntax. This supports non-Java developers who want to create dynamic Web pages.
- XML and XSL support developers who want to create XML pages instead of HTML pages.

Deploying JSP Applications Since the servlet engine compiles pages on request, you must test your pages by requesting them through the Oracle HTTP Server in Oracle9*i* Application Server. Some integrated development environments, such as Oracle JDeveloper, provide built-in debugging environments for JSP.

Processing JSP Requests In Oracle9*i* Application Server, JavaServer Pages run in the Oracle HTTP Server component. The Oracle HTTP Server forwards the request through the servlet environment (mod_jserv and the Apache JServ servlet engine) to OracleJSP. The translator processes the JavaServer Pages and compiles any embedded code. The output returns along the originating path to the client. This process is illustrated in Figure 3–2.

Figure 3–2 Control Flow for a JavaServer Page in the Application Server



1. The Oracle HTTP Server receives a request for a JSP from a client.
2. The Oracle HTTP Server dispatches the request for a servlet class to the mod_ jserv module.
3. The mod_jserv module forwards the request to the Apache JServ servlet engine.
4. The servlet engine translates the incoming request into a request object. The servlet engine then creates a response object and passes both objects to OracleJSP.
5. The OracleJSP parses the JSP file and executes the embedded application logic.
6. The result is passed from the OracleJSP to the Apache JServ servlet engine in the servlet response object. The response then travels back to the client through the mod_jserv module and Oracle HTTP Server.

See Also:

- JavaServer Pages Specification version 1.0 at `http://java.sun.com`
- OracleJSP documentation in the Oracle9*i* Application Server Documentation Library.

PL/SQL Server Pages

Oracle PL/SQL Server Pages (PSP) is Oracle's PL/SQL dynamic server-side scripting solution for Web application development. Oracle PSP includes the PL/SQL Server Pages Compiler and the PL/SQL Web Toolkit. Oracle PSP enables PL/SQL users to develop Web pages with dynamic content by embedding PL/SQL scripts in HTML. PSPs separate application logic (embedded PL/SQL scripts) from the layout logic (HTML) making the development and maintenance of PL/SQL Server Pages easy. By using this method, content developers design the static portions of Web pages in HTML, then add scripts that generate the dynamic portions of the pages. In addition, PSP tightly integrates with the database so it is easy to retrieve, manipulate, and present data.

Advantages of Using PSP PL/SQL Server Pages provide server-side scripting with traditional database operations and programming logic for developing Web-based applications. The advantages of using PL/SQL Server Pages include the following:

- Oracle PSP supports HTML and HTML authoring tools with added dynamic content or applications. You can start with an existing Web page or with an existing stored procedure to create dynamic Web pages that perform database operations and display the results.
- Typically, PL/SQL Server Pages display in a Web browser. By default, Oracle8*i* PL/SQL transmits files as HTML documents, so that the browser formats them according to the HTML tags. Files can also be retrieved and interpreted by a program that can make HTTP requests, such as a Java or Perl application.
- The PL/SQL Web Toolkit allows you to build PL/SQL code that produces formatted output. You can specify the output format as XML, a MIME type, or plain text, depending on the technology supported in the user's browser.
- You can run insert, update, and delete operations within PL/SQL Server Pages. As with any program that is expected to return results as HTML, such pages should include some output to confirm that the operation is successful or to show the updated state.
- You can share procedures, constants, and types across different PL/SQL server pages, and compile them into a separate package in the database.

- To handle database errors that occur when the script runs, you can include PL/SQL exception-handling code within a PSP file, which may launch another PSP to handle or to report the error.

Developing PSP Applications Developers use the Oracle PL/SQL Web Toolkit to develop their PSP applications. The PL/SQL Web Toolkit contains a set of packages you can use in stored procedures to retrieve request information, construct HTML tags, and return header information to the client.

PL/SQL Server Pages compile to PL/SQL stored procedures. Compiling an HTML file as a PL/SQL Server Page produces a stored procedure that outputs the exact same HTML file. The PSP is an HTML file mixed with PL/SQL procedures combining all the content and formatting of your Web page. The HTML file contains text and tags interspersed with PSP directives, declarations, and scripts.

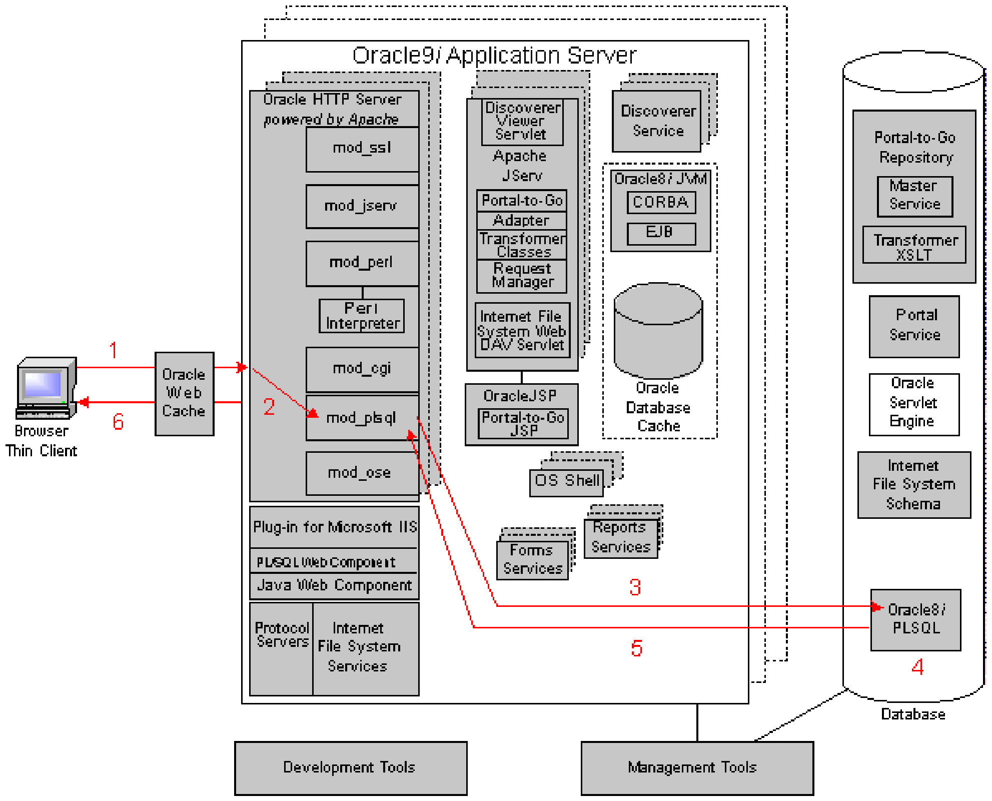
Deploying PSP Applications Oracle8*i* PL/SQL and mod_plsql in Oracle9*i* Application Server provide support for deployment and performance of your PL/SQL Server Pages. By deploying PSPs on the middle-tier, the server centrally manages the application logic saving in administration and maintenance costs and optimizing performance of your PL/SQL applications. Oracle HTTP Server forwards PL/SQL request(s) to the PL/SQL engine with the plug-in mod_plsql. Applications invoke PL/SQL scripts and stored procedures to retrieve data from a database, then generate HTML pages that return the data to the client browser.

Once the PSP has been compiled into a stored procedure, you can run it by retrieving an HTTP URL through a Web browser. The virtual path in the URL depends on the way that mod_plsql is configured.

The `POST` and `GET` methods in the HTTP protocol tell browsers how to pass parameter data to the applications. The `POST` method passes the parameters directly from an HTML form and are not visible in the URL. The `GET` method passes the parameters in the query string of the URL. You can use the `GET` method to call a PSP from an HTML form, or you can use an HTML link to call the stored procedure with a given set of parameters.

Processing PSP Requests The process of handling a PL/SQL Server Page is illustrated in Figure 3–3.

Figure 3–3 Control Flow for a PSP in the Oracle9i Application Server



1. The Oracle HTTP Server receives a PL/SQL Server Page request, through Oracle Web Cache, from a client browser.
2. The Oracle HTTP Server routes the request to mod_plsql.
3. The request is forwarded by mod_plsql to Oracle8*i* PLSQL. By using the configuration information stored in your Database Access Descriptor (DAD), mod_plsql connects to the database, prepares the call parameters, and invokes the PL/SQL procedure in the database.
4. The PL/SQL procedure generates an HTML page using data and stored procedures accessed from the database.

5. The response is returned to mod_plsql.
6. The Oracle HTTP Server sends the response, through Oracle Web Cache, to the client browser.

> **See Also:** *Oracle8i Application Developer's Guide - Fundamentals* in the Oracle Database Documentation Library.

Oracle Reports Services

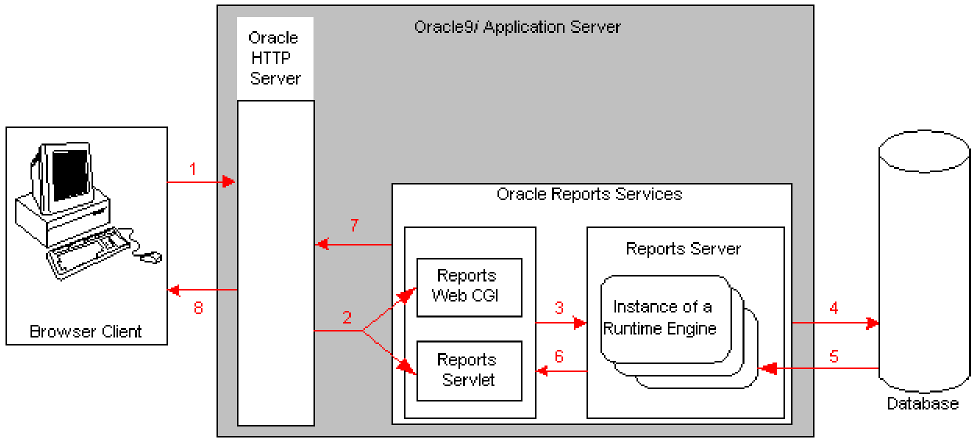
Oracle Reports Services enables you to deploy new and existing reports within the Oracle9*i* Application Server multitiered architecture. All report processing, administration, and maintenance occurs on the server, which dramatically simplifies the client configuration and reduces the client storage and processing requirements. Oracle Reports Services supports all of the major industry standard Web output formats, making it easy for users to view reports in their favorite Web browser.

Oracle Reports Services supports output in the following industry standard formats:

PDF	Adobe Portable Document Format provides high quality output that users can view on the Web and easily print.
HTML and HTMLCSS	HTML without cascading style sheets provides output that can be viewed in any HTML 3.0 compliant browser. HTML with cascading style sheets (HTMLCSS) provides higher fidelity output, but it requires an HTML 3.0 compliant browser that supports style sheets.
XML	XML (Extensible Markup Language) stores report definitions. If you apply different XML report definitions at runtime, then you can customize report formats for different users or purposes.

Processing Report Requests Oracle Reports Services uses the Oracle9*i* Application Server multitiered architecture to publish and run report requests on the Web. Figure 3–4 shows Oracle Reports Services residing on the same machine as Oracle HTTP Server. Alternatively, you may distribute the Oracle Reports Services across multiple machines by running the Reports Server on a separate machine. If you choose to use this distributed architecture, then the Oracle HTTP Server and the Reports Web CGI or Reports Servlet components must always reside on the same machine.

Figure 3–4 Control Flow for a Report Request in the Oracle9i Application Server



Note: This diagram depicts Oracle Reports Services in a three-tiered architecture. You may also run Oracle Reports Services in a distributed four-tiered architecture.

---

1. When a client requests a report from their Web browser, the browser passes the request to Oracle HTTP Server on Oracle9*i* Application Server.
2. Oracle HTTP Server invokes either the Reports Web CGI or the Reports Servlet, depending on the configuration.
3. The Reports Web CGI or the Reports Servlet parses the request, handles the login transaction, converts the report to command-line format, and submits the request for execution by the Reports Server.
4. The Reports Server checks its output cache for an existing response to the request. If the cache has an acceptable output, then the Reports Server immediately returns that output. If the cache does not have the acceptable output, then the Reports Server processes the request with the database.
5. The Reports Server receives and queues the job request. When one of the runtime engines becomes available, the Reports Server sends the command line to that runtime engine for execution. The runtime engine runs the report.
6. The Reports Web CGI or Reports Servlet receives the report output from the Reports Server.

7. The Reports Web CGI or Reports Servlet sends the output to Oracle HTTP Server.
8. Oracle HTTP Server sends the report output to the client's Web browser.

Deploying Reports When deploying reports in a Web environment, you should consider the following:

- **choose the Report Web CGI or Reports Servlet:** You must install and configure either the Reports Web CGI or Reports Servlet to handle the transmission of job requests and output between your Oracle HTTP Server and Oracle Reports Services. Oracle9*i* Application Server supports both CGI and Java applications.
- **choose the location of the Reports Server:** You can deploy Oracle Reports Services in Oracle9*i* Application Server using a three-tiered or four-tiered architecture. When using a three-tiered architecture, the Reports Server resides on the same machine as Oracle HTTP Server. This requires more system resources, since all processing occurs on a single machine. In a four-tiered architecture, the Reports Server resides on a machine separate from Oracle HTTP Server, spreading the resource requirements across multiple machines. However, if you choose to have the Reports Server and Oracle HTTP Server on different machines, then the transmissions to the Reports Web CGI and the Reports Servlet must travel across a network resulting in greater network traffic.
- **configure request handling for optimal performance:** Oracle Reports Services handles report requests using dynamic management of runtime engines. You can specify a maximum number of runtime engines to handle your requests with optimal performance for your server. When Oracle Reports Services receives a request to run a report, the server launches a runtime engine to handle the request. Each runtime engine resides in memory to run additional report requests over a period of time. A runtime engine automatically removes itself from memory when it is either idle for a specified amount of time to free up system resources. After removal, the Reports Server replaces it with a new engine.

  See Also: Oracle Reports Services documentation in the Oracle9*i* Application Server Documentation Library.

Business Logic

The Oracle9*i* Application Server provides support for running applications with advanced functionality such as business logic, scalability, and performance. The following topics are covered in this section:

- Java Servlets, Applications, and Enterprise JavaBeans
- Global Transactions Including Oracle Database Cache and the Origin Database
- Oracle Forms Services

Java Servlets, Applications, and Enterprise JavaBeans

Java code can be deployed for execution on either one of two Java runtime environments in Oracle9*i* Application Server:

- Apache JServ running on the Java Development Kit Java Virtual Machine (JDK JVM), or
- Oracle8*i* JVM.

Note: Java applications running in either one of these environments on Oracle HTTP Server can communicate with Java applications in Oracle8*i* JVM.

Running Java on the JDK JVM

Apache JServ supports running servlets and JSPs. When deploying Java applications on the JDK JVM, you should consider the following:

- Conversational State
- JNI Access
- Stateful vs. Stateless Applications

Conversational State Applications that are stateless or hold on to minimal conversational state will benefit from the responsiveness of the JDK JVM.

JNI Access If your Java application requires access to the Java Native Interface, then you must use the JDK JVM. JNI access is not supported in Oracle8*i* JVM.

Stateful vs. Stateless Applications When comparing the JDK JVM with Oracle8*i* JVM, the terms stateful and stateless are useful.

A *stateful* application maintains session state information within its runtime environment between successive client calls.

A *stateless* application maintains no state information within its environment. It may, however, refer to state information in a common store such as a database or a browser.

The JDK JVM scales by giving quick performance to many clients, since stateless Java applications do not need to maintain state information. This works well for stateless Java applications because the JVM does not get weighted down by holding onto a lot of state. However, stateful applications force the JDK JVM to perform a lot of concurrent memory management when multiple users access the system. Managing state may inhibit the scalability of the JDK JVM. Figure 3–5 and Figure 3–6 illustrate the effects of adding clients to a JDK JVM for both stateful and stateless Java applications.

Figure 3–5 Adding Clients to JDK JVM for Stateful Java Applications

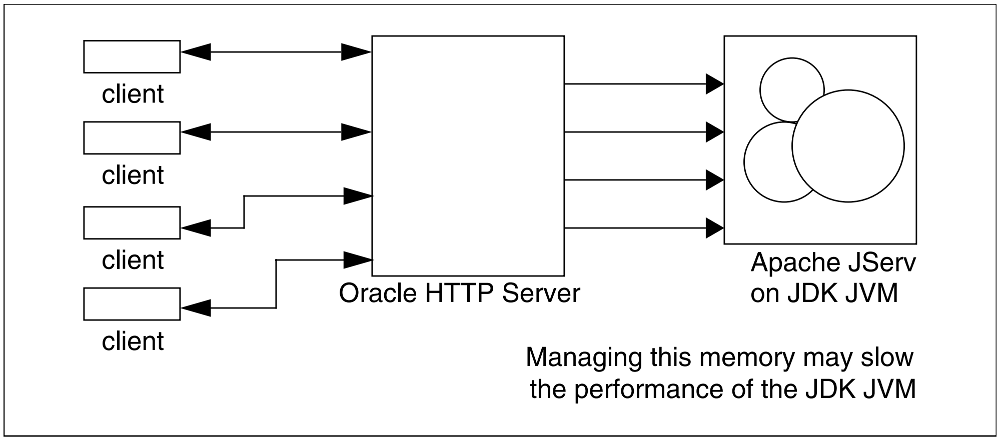
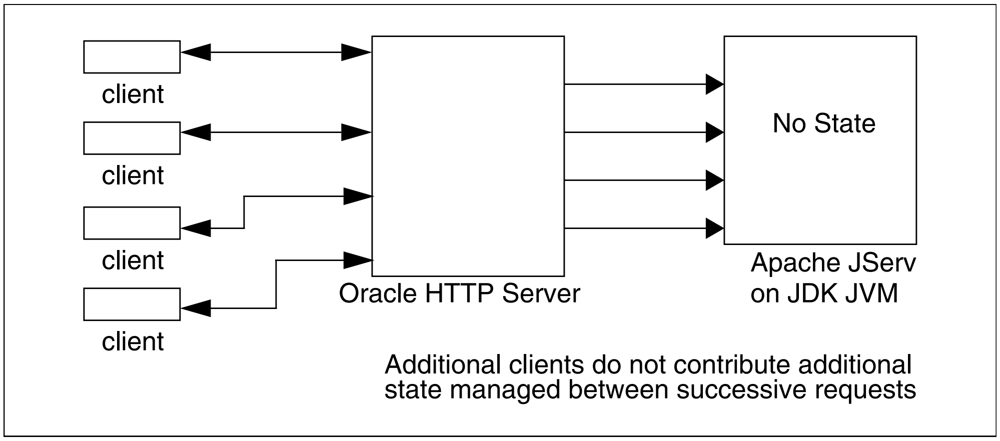
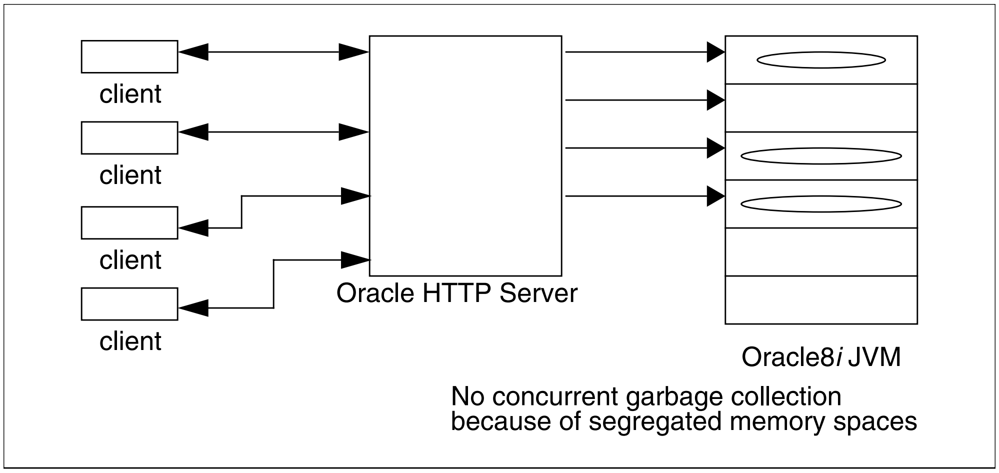


Figure 3–6 Adding Clients to JDK JVM for Stateless Java Applications


client
client
client
client
Oracle HTTP Server
No State
Apache JServ on JDK JVM
Additional clients do not contribute additional state managed between successive requests

Oracle8*i* JVM is a session-based JVM that handles stateful applications with good performance, depending on the hardware capacity. As Oracle8*i* JVM segregates clients' memory spaces, the JVM can garbage collect each user's memory space independently. This architecture avoids concurrent garbage collection, which often constitutes the major scalability bottleneck when running heavily stateful applications on a typical JVM.

Figure 3–7 Adding Clients to Oracle8i JVM with Stateful Java Applications


client
client
client
client
Oracle HTTP Server
Oracle8i JVM
No concurrent garbage collection because of segregated memory spaces

Running Java on Oracle8*i* JVM

Oracle8*i* JVM supports Enterprise JavaBeans. When deploying Java applications on Oracle8*i* JVM, you should consider the following:

- Conversational State
- Benefit of Fast SQL Access
- Security, Reliability, and Availability Requirements

Conversational State Applications that hold on to substantial amounts of conversational state will scale better in Oracle8*i* JVM.

Benefit of Fast SQL Access Oracle8*i* JVM runs in the same process space as the Oracle8*i* SQL engine. In the Oracle8*i* database, Oracle8*i* JVM benefits from fast data access when reading and writing to the database. In Oracle9*i* Application Server, Java applications running in Oracle8*i* JVM benefit from very fast access when reading cached data in Oracle Database Cache.

Security, Reliability, and Availability Requirements Oracle8*i* JVM inherits many of the security, reliability, and availability features of the Oracle8*i* database, creating a very stable Java environment. For example, Oracle8*i* JVM isolates client sessions, so that failure in one session is not propagated to other concurrent user sessions. In many JVMs, one user session has the potential to bring down the entire JVM, affecting all concurrent JVM users. The session isolation in Oracle8*i* JVM protects sessions from one another. Even if one user's session goes down in Oracle8*i* JVM, other users' sessions are unaffected.

Deploying Enterprise JavaBeans in the Middle Tier

In Oracle9*i* Application Server, you can deploy Enterprise JavaBeans (EJBs) either in the middle tier or in the database. The deployment process is similar: You must direct the deployment tools (`deployejb` and `loadjava`) to the correct server.

Deploying EJBs in the Database To deploy EJBs in the database, you direct the `deployejb` and `loadjava` commands to the origin, or back-end, database, and then pass the `host`, `port`, and `sid` of the database to the command-line tools.

Deploying EJBs in the Middle Tier To deploy EJBs in the middle tier you must perform the following steps:

1. Before you can deploy EJBs to the middle tier, you must install Oracle8*i* JVM.
2. Direct the `deployejb` and `loadjava` commands to Oracle8*i* JVM.
3. Pass the `host`, `port`, and `sid` of the JVM to the command-line tools.

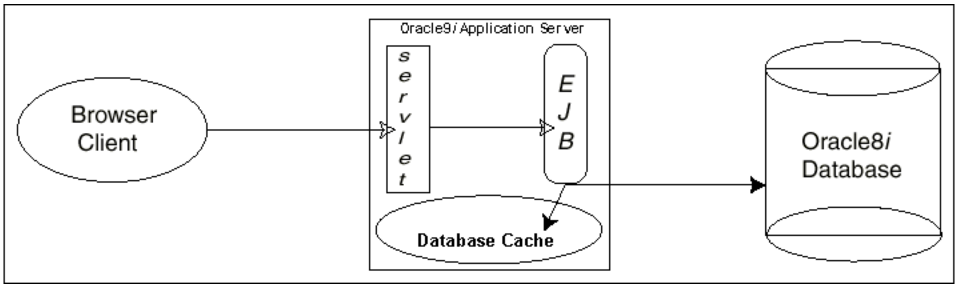
Note: There are restrictions on using EJBs in the middle tier. These restrictions are documented in the *Oracle9i* Application Server *Release Notes*.

Global Transactions Including Oracle Database Cache and the Origin Database

In a normal global transaction, you open connections to each database that you want included in the transaction. After the transaction completes, the transaction manager commits all changes to all databases involved in the transaction.

By using Oracle9*i* Application Server, Oracle Database Cache may incorrectly be treated by the transaction manager as one of the databases in the global transaction. A typical application is shown in Figure 3–8. The Enterprise JavaBean (EJB) that is active in the middle-tier retrieves a connection to both Oracle Database Cache and to the origin database. However, the EJB must only update the origin database. The transaction manager must not treat Oracle Database Cache as another database involved in the transaction or it will perform a two-phase commit when the transaction ends. The two-phase commit process is expensive and unnecessary. Only the origin database should be enlisted in the global transaction so the transaction manager will perform a single-phase commit.

Figure 3–8 Global Transaction That Includes Oracle Database Cache and the Origin Database



All database resources are enlisted only if you perform one of the following:

- You retrieve the connection to a remote database through the `getConnection` method of the `DataSource` class. However, the `DataSource` object must be previously bound through `bindds` with the `-type jta` option.
- You retrieve the connection to the local database using one of the methods described in Table 3–1.

Table 3–1 Local Resource Enlistment

Retrieval Method	Description
`defaultConnection` method of the `OracleDriver` class	Pre-JDBC 2.0 method for retrieving the local connection.
`getConnection` method of the `DriverManager` class	Pre-JDBC 2.0 method for retrieving connections. For retrieving the local connection, the input parameter must be the string `"jdbc:oracle:kprb:"`.
`getConnection` method of the `DataSource` class	JDBC 2.0 method for retrieving connections. For retrieving the local connection, the input parameter must be the string `"jdbc:oracle:kprb:"`.

To ensure that Oracle Database Cache is not treated as an enlisted database in the global transaction, perform the following steps:

1. Ensure that the `<default-enlist>` element in the Oracle-specific deployment descriptor is either not set (and it defaults to FALSE) or is set to FALSE.
2. Bind the `DataSource` for Oracle Database Cache with any non-Java Transaction API (JTA) type. Only a JTA `DataSource` object can be automatically enlisted in a global transaction.

   The behavior for the methods indicated in Table 3–1 are as follows:

   - With the pre-JDBC methods (the `OracleDriver defaultConnection` and `DriverManager getConnection` methods) Oracle9*i* Application Server understands that Oracle Database Cache is not a database, and the cache is not enlisted in the global transaction.

- With the JDBC 2.0 method (the `DataSource getConnection` method) the `DataSource` object bound for the cache must be bound through the `bindds` command with any `-type` other than `jta`. If bound with `bindds -type jta`, then the cache will be considered part of the global transaction and the transaction manager will complete the global transaction with a two-phase commit.

  See Also: *Oracle8i Enterprise JavaBeans Developer's Guide and Reference* in the Oracle9*i* Application Server Documentation Library.

Oracle Forms Services

Oracle Forms Services deploys Forms applications with database access to Java clients in a Web environment. Together with Oracle Forms Developer, Oracle Forms Services provides

- application infrastructure and the event model for scalability and performance over a network
- support for integrating technologies such as PL/SQL, Java Stored Procedures, Enterprise JavaBeans, XML, and CORBA
- extensible user interface through native Java with Pluggable Java Components
- services for building and optimizing Oracle8*i* transactional database applications

When combined with Oracle Forms Developer and Oracle Designer, you can design, develop, and deploy a complete application framework for Oracle Forms applications on the Internet. Oracle Forms Developer allows you to quickly build complex Java applications for accessing a database, without writing any Java code. The development environment provides a full set of tools, such as wizards and utilities, for building custom, extensible applications. Oracle Forms Developer is specifically designed and optimized to build Oracle8*i* transactional database applications, with built-in database connectivity, query, management, and transactional services. Oracle Forms Services and Oracle Forms Developer also integrate with the Oracle Designer modeling tool to provide services for full life cycle application development and deployment.

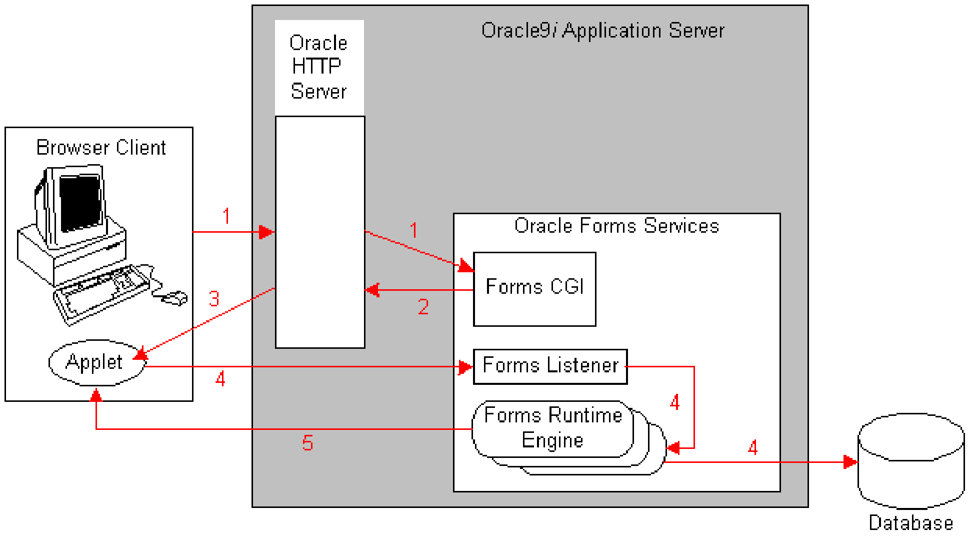
With the integration of Forms Developer and Oracle8*i*, your Internet applications can share resources and improve application performance and scalability. Oracle Forms Services supports this integration with

- transaction management
- advanced queuing
- record caching
- record locking
- exception handling
- load balancing
- security management features

Processing Forms Requests

Oracle Forms Services resides as a component in Oracle9*i* Application Server. The process of handling Forms applications is illustrated in Figure 3–9.

Figure 3–9 Control Flow for a Forms Request in Oracle9i Application Server



1. Oracle HTTP Server receives an HTTP request from the browser client and contacts Forms CGI.
2. Forms CGI dynamically creates an HTML page containing all information to start the Forms session.
3. Oracle HTTP Server downloads a generic Java applet to the client. The client caches the applet so the applet does not need to be downloaded again. The applet runs all future Forms applications for as long as it resides in the client's cache.
4. The client applet contacts the Forms Listener to start the session. The Forms Listener starts an instance of the Forms Runtime engine on the Forms Server. The Forms Runtime engine handles the user's context, executing the business logic and conducting necessary transactions with the Oracle8*i* database instance.
5. The client applet and Forms Runtime engine establish network communication using socket, HTTP, or S-HTTP protocols.

The Forms Java client has been optimized for high performance on many platforms allowing efficient display of widgets and minimizing the time and frequency for client page refreshes. Another key element of the optimized Java client is the use of Java Archive (JAR) file caching. Oracle Forms Services use Oracle JInitiator to preform persistent client-side caching of the applet after the initial download. Any subsequent access to the application pulls the JAR file directly from the persistent client-side cache, significantly minimizing startup time. JAR file caching is an essential performance feature for any application with remote users who dial in to access the application over a wide area network.

Oracle Forms Services also optimizes network traffic by using several methods for communicating to the Java client, including:

- reducing the amount of network traffic by using metadata messages as a collection of name/value pairs. These messages tell the Java client which object to act upon and how to act
- comparing messages as collections of name/value pairs and sending only the differences between the messages
- sending an initial string once and referencing the string in subsequent messages
- minimizing the transfer of data types by using the lowest number of bytes required for their value
- bundling events triggered between two objects for single packet processing

See Also: Oracle Forms Services documentation in the Oracle9*i* Application Server Documentation Library.

Index

A

ad hoc query, 2-53
Apache JServ, 2-19, 3-8
application logic, 1-8
architecture overview diagram, 1-9
authentication, 2-49

C

Cache Manager console. *See* Oracle Enterprise Manager
cache. *See* Oracle Database Cache
Calypso. *See* Oracle Web Cache
clients, thin, 1-8
Common Gateway Interface (CGI), 3-3

D

database access descriptor (DAD), 3-11
database server, 1-7
Discoverer Service, 2-54
Discoverer Viewer servlet, 2-54
Document Object Model (DOM). *See* Oracle XML Developer's Kit
Document Type Definitions (DTDs). *See* Oracle XML Developer's Kit

E

Enterprise JavaBeans (EJBs), 3-15
  deploying in the middle tier, 3-19
enterprise portal, 2-32

F

Forms, 3-22
Forms Developer, 3-22
Forms Services, 3-22

H

HDML (Handheld Devices Markup Language), 2-38
HTML
  Oracle Reports Services, 3-12
HTMLCSS
  Oracle Reports Services, 3-12

I

IMAP (Internet Message Access Protocol), 2-12
*inter*Media Text, 2-10

J

Java
  Java Messaging Service (JMS) client, 2-23
  JavaServer Pages (JSP), 3-7
  Oracle SQLJ Translator, 2-25
Java Virtual Machine (JVM), 3-5
JDBC drivers
  JDBC Oracle Call Interface (JDBC-OCI) driver, 2-25
  Oracle JDBC server-side internal driver, 2-25
  Oracle JDBC thin driver, 2-25
JInitiator, 3-24
  Oracle Forms Services, 3-24

JNDI (Java Naming and Directory Interface). *See* Oracle LDAP Developer's Kit
jserv module, 2-5
JSP Markup Language (JML), 3-7

L

LDAP. *See* Oracle LDAP Develper's Kit

M

Microsoft Internet Information Server (IIS), 2-5
middle-tier database cache, 2-42
mod_8*i*. *See* mod_ose
mod_jserv, 2-5, 3-5
  JavaServer Pages (JSP), 3-8
  Oracle HTTP Server powered by Apache, 3-5
  servlets, 3-5
mod_ose, 2-5
mod_perl, 2-4
mod_plsql, 2-4
mod_ssl, 2-4
Motorola, 2-38

O

Oracle Advanced Security, 2-49
  authentication methods supported, 2-49
  DES encryption, 2-49
  MD5 algorithm, 2-49
  PKI credential management, 2-51
  RSA encryption, 2-49
  single signon, 2-50
Oracle Call Interface (OCI), 2-26
Oracle Database Cache, 2-41
  application criteria, 2-41
  benefits, 2-41
  cannot create tables, 2-42
  cannot perform backup/recovery, 2-42
  environment, 2-42
  global transactions, 2-42, 3-20
  TAF (transparent application failover), 2-42
Oracle Designer, 3-22
Oracle Discoverer 3*i* Viewer, 2-53
  workbooks, 2-53
Oracle Enterprise Manager, 2-47
Oracle Forms Services, 2-17, 3-22
  applets, 2-17
Oracle Internet Developer Suite, 1-2
Oracle Internet Directory (O*i*D). *See* Oracle LDAP Developer's Kit
Oracle Internet File System
  schema, 2-13
  SDK, 2-11
Oracle Internet Suite, 1-2
Oracle JDeveloper
  Java, 3-5
  servlets, 3-5
Oracle LDAP Developer's Kit, 2-29
Oracle Plug-in for Microsoft IIS, 2-5
  Java Web component access, 2-5
  PL/SQL Web component access, 2-5
Oracle Portal, 2-32
  portal, 2-32
  portlet providers, 2-33
  portlets, 2-33
  preference store, 2-35
  session store, 2-35
Oracle Portal-to-Go
  adapters, 2-37
  JavaServer Pages (JSPs), 2-36
  master service, 2-39
  Personalization Portal, 2-36
  Portal-to-Go XML, 2-36
  repository, 2-39
  request and response sequence, 2-38
  request manager, 2-39
  request object, 2-39
  transformers, 2-37
Oracle Reports Developer applications, 2-53
Oracle Reports Services, 2-53, 3-12
  Reports, 3-12
Oracle Servlet Engine (OSE), 2-5
Oracle SQLJ
  runtime, 2-24
  SQLJ profiles, 2-24
  translator, 2-23
Oracle SQLJ Translator, 2-25
Oracle Web Cache, 2-44
  architecture diagram, 2-45

Oracle XML Developer's Kit (XDK), 2-27, 3-4
  XML Class Generator for Java, 2-27
  XML Parser for Java, 2-27
  XML Transviewer JavaBeans, 2-28
  XSQL Servlet, 2-28
Oracle8*i* Cache, ix
Oracle8*i* Cache. *See* Oracle Database Cache
Oracle8*i* PL/SQL, 3-9
Oracle8*i* PLSQL, 2-16
OracleJSP, 3-7, 3-8
origin database, 2-42

P

Palm Computing Platform, 2-37
PDF
  Oracle Report Services, 3-12
Perl, 3-3
  Perl Interpreter, 2-21, 3-3
  Perl module, 2-4
Personalization Portal, 2-36
PL/SQL module, 2-4
PL/SQL Server Pages (PSP), 3-9
PL/SQL Web Toolkit, 3-9
  PL/SQL Server Pages (PSP), 3-9
portal, 2-32
portal service, 2-32
portlet providers, 2-33
portlets, 2-33
  custom, 2-34
presentation layer, 2-18
presentation services, 2-18
programming languages supported, 1-13
provisioning adapter, 2-37

R

Reports and Graphics, 2-53
Reports Servlet, 2-53
runtime engines
  Apache JServ, 3-8
  Oracle8*i* PL/SQL, 3-9

S

security
  Oracle Advanced Security, 2-49
services diagram, 1-3
servlet adapter, 2-37
servlets, 3-5, 3-15
  Apache JServ, 3-8
Short Message Service, 2-38
single signon, 2-50
SMB (Server Message Block Protocol), 2-12
SMTP (Simple Mail Transfer Protocol), 2-12
SPARC Solaris, 2-53
SQL adapter, 2-37
SQLJ. *See* Oracle SQLJ
SSL module, 2-4
SSL Toolkit. *See* Oracle LDAP Developer's Kit
stored procedures, 2-16
stripper adapter, 2-37

T

TAF (transparent application failover), 2-42
technologies supported, 1-13
thin clients, 1-8
three-tier computing model, 1-8
Tiny HTML, 2-37
transparent application failover (TAF), 2-42
two-tier computing model, 1-7

U

URL adapter, 2-37

V

versions available
  enterprise edition, 1-11
  standard edition, 1-11
  wireless edition, 1-11
VoxML/VoiceXML, 2-38

W

WAP (Wireless Access Protocol), 2-37
Web DAV (Distributed Authoring and Versioning), 2-12
Web Folders, 2-9
Web integration adapter, 2-37
Web Interface Definition Language (WIDL), 2-37
WIDL (Web Interface Definition Language), 2-37
Windows CE devices, 2-37
Windows NT, 2-53
Wireless Access Protocol (WAP), 2-37
WML 1.1 (Wireless Markup Language), 2-37
workbooks, 2-53

X

XDK. *See* Oracle XML Developer's Kit
XML, 3-4, 3-7, 3-12
  Oracle Reports Services, 3-12
  XML Class Generator for Java, 2-27
  XML Parser for Java, 2-27
  XML Transviewer JavaBeans, 2-28
  XSQL Servlet, 2-28
XSLT (Extensible Stylesheet Language Transformations), 2-37