Oracle Application Server 4.0 White Paper: Product Overview

An Oracle White Paper
September 1998

Product Overview

INTRODUCTION

Customers in every industry today are web-enabling their applications and using intranets and the Internet to perform business functions critical to the success of their enterprises. This trend has created many new challenges in the area of application development and deployment.

Oracle's 1996 introduction of the network computing initiative (also known as Internet computing) created a company-wide focus on building the scalable, reliable, manageable and secure server-side technology needed for these web-enabled solutions. At the core of the network computing initiative is a new breed of software technology called an application server.

In 1997, Oracle pioneered the market for application servers with Web Application Server 3.0. The new version of this product, now known as Oracle Application Server (OAS) 4.0, continues to lead the market by offering new features and functions unmatched in the industry. OAS 4.0 supports multiple application programming models (Web, CORBA, Enterprise Java Beans) and languages (Java, Perl, C, PL/SQL, COBOL and LiveHTML), allowing it to unify the capabilities of multiple middleware products into a single offering that can be installed and managed from a common user-friendly interface.

This paper provides an overview of the OAS 4.0 product, beginning with a description of how the growth of the Internet and the limits of simple web servers have led to the demand for application servers. The capabilities of OAS 4.0 are described, along with the types of applications that will benefit from the use of this product. An overview of how applications developed and deployed on OAS 4.0 achieve enterprise-level scalability, reliability, manageability and security is also provided. Development tools that programmers can use to write OAS 4.0 applications are listed in the appendix to this product overview.

Oracle tools, applications and Enterprise Manager now use OAS as their middle-tier platform. A closer look at how Oracle product divisions will use OAS 4.0 is available in the paper "*Integration Platform for Oracle Products with the Internet.*"

IMPACT OF THE INTERNET ON MIDDLEWARE CONVERGENCE

In recent years, the widespread adoption and use of several Internet technologies has pushed client/server applications from the two-tier to the three-tier model, and created challenges not solved by standard web servers.

Internet Technologies and Associated Challenges

THE WORLD WIDE WEB

The power of the web derives first and foremost from the core capability of URLs and easy-to-use browsers. With a simple click of the mouse, users can access content stored in HTML pages in the

form of text, video, graphics and sound. The HTTP protocol running over TCP/IP allows users to access HTML pages stored on any server connected to the web, anywhere in the world.

BROWSERS AS UNIVERSAL CLIENTS

There is a high cost to build different applications to run on multiple, heterogeneous client platforms (e.g., Windows, Macintosh, UNIX, browser). It is also labor-intensive to deploy, and update applications across thousands of client PCs. When the Internet first exploded on the scene in the early 90s, IT managers and developers immediately seized on the idea of using the universal browser interface as a way around the development and deployment expense of PC-based clients. These benefits are driving many customers to web-enable applications that will never be deployed on the Internet.

JAVA AS THE LANGUAGE FOR NETWORK COMPUTING

The arrival of Java marked a watershed for Internet computing. The "write once, run anywhere" slogan for Java applications is analogous to the "write once, view anywhere" HTML slogan for information stored on the Web. Being inherently OS independent, Java applications can be written once and run anywhere that has a Java Virtual Machine.

Realizing the folly of recreating the deployment issues of fat clients, developers have resisted coding business logic on the client. Instead, client-side Java is used sparingly to enhance the "presentation" side of solutions. Forms for data input include applets (or Java "Beans") which improve the user interface experience. Browsers can download Java code from servers and run it, allowing new applications and data to be distributed to clients on an as-needed basis. These applications are platform independent and can be downloaded and managed from a central location.

The result is that users running browsers—on any type of client platform—are now just a point-and-click away from the rich content available on HTML pages, as well as the new features and capabilities implemented in Java Beans and applets (see below for details). As an added bonus, many customers have found that enterprise applications can be extended to a wider audience of distributed users as browser-based clients operate over wide-area networks instead of the local area networks required in traditional two-tier client-server applications.

ASSOCIATED CHALLENGES

This Internet technology has had a tremendous impact on the number of users and the number of requests that can arrive at a server at any given instant in time. Internet servers can be accessed 24 hours a day by millions of users anywhere in the world. At the same time, the number of end users accessing internal, web-enabled corporate applications often skyrockets as well. For example, an Oracle internal human resources application that was once accessed by no more than 100 people can now be accessed by 30,000 people—the number of Oracle employees. This kind of demand presents several challenges to standard web servers because:

- Scalability is critical when the number of end users and the number of requests being made to a server are unpredictable and subject to fluctuations.
- Reliability must be addressed when access to an application is available 24 hours a day.
- Transactions involving business critical information between browser-based clients and web servers must be guaranteed to complete fully, or not at all.
- Proprietary security schemes used in two-tier client server applications to protect sensitive data must be replaced by new, public key-based encryption mechanisms, such as SSL.
- IT managers must have a mechanism that can intercede between incoming requests and requested resources to enforce corporate access policies. For interoperability with other web servers, the solution must be standards-based and support the use of digital certificates.

Distributed Component Technologies and Associated Challenges

Two additional middleware technologies are often included in truly mission-critical web-based solutions. Both provide valuable capabilities but also have serious limitations. The first is CORBA, an industry standard for distributed component solutions, defined by a the Object Management Group (OMG), a standards body consisting of more than 800 member companies (vendors and enterprise customers).

While dramatically different from the operation of PCs as clients in two-tier client-server applications, the use of browsers and web servers retains the semblance of clients as "requesters" that depend on a central server in order to perform meaningful work. The introduction of CORBA completely obliterates this paradigm.

CORBA AS THE INFRASTRUCTURE FOR NETWORK COMPUTING

For many years the goal of MIS departments has been to speed software development by assembling complete solutions from reusable software components. The architects of IT solutions dream of being able to host these components, which may be built in-house or purchased from vendors, on distributed servers and clients. This flexibility of partitioning would maximize performance and reliability.

The OMG has dedicated the last eight years to making this dream a reality. The OMG's CORBA standards specify how software components written in various languages can be "wrapped" in a uniform way. The wrapper defines a set of methods (roughly equivalent to function calls in traditional programs) that are visible to other components. These components, which may be on the same server or distributed across a network, are hosted on object request brokers (ORBs) that communicate using the standard Internet Inter-ORB Protocol (IIOP).

CORBA 2.0-compliant ORBs, and the remote procedure call (RPC) mechanisms they are based on, mask the underlying communications technologies between clients and servers, allowing components to make calls to other components located anywhere on the network as if they were local. This is key for network computing, where clients are broken up from servers and both exist as objects and components[1] that can freely interact with each other.

JAVA BEANS AS THE COMPONENT MODEL FOR NETWORK COMPUTING

A surprising truth is that, despite years of effort, CORBA has not defined a standard component model accepted by the market, and without this, a true market for reusable components cannot reach critical mass. In a less-than-amazing coincidence, the recent "Enterprise Java Bean" (EJB) component model has been warmly embraced by vendors as diverse as Oracle, IBM, Sun and Netscape.

What is an EJB? Similar to a Java Bean—a Java program that has been packaged as a component for use within a browser—an EJB is a Java program that has been packaged for use on a server. But instead of accessing Java classes that enable a user interface (the primary use of Java Beans), an EJB accesses Java classes that provide network services such as security, directory and transactions.

---

[1] In this document, the terms "objects" and "components" are used somewhat interchangeably. More accurately, it is generally agreed that a component consists of one or more objects. Objects tend to be very small-grained (perform one small task) whereas components are much larger grained and hence more useful.

[2] See Paul Allen and Stuart Frost, *Component-Based Development for Enterprise Systems*, Cambridge UK: Press Syndicate of the University of Cambridge, 1998

While many perceive CORBA and EJBs as competing standards, they are actually synergistic. CORBA provides the infrastructure for EJBs, and EJBs are the component model for CORBA. Clients as network objects require support for Java, a CORBA-compliant ORB and IIOP. Server components as network objects will require support for EJBs. Standard web servers do not begin to meet these requirements.

ASSOCIATED CHALLENGES

While the value of ORB and EJB functionality has been clear, the challenges of using this technology have been equally apparent:

- ORBs lack the scalability or reliability demanded by enterprise systems.
- Traditional ORB vendors are small, with extremely limited service and support.
- ORBs pre-date the web and therefore must be paired with HTTP listeners for web-based applications. The result is two platforms with two installations and two management consoles.

Transaction Processing Technologies and Associated Challenges

The transaction processing monitor (TP monitor) is the second middleware technology often found in business-critical solutions. Like ORBs, TP monitors provide valuable capabilities mitigated by glaring weaknesses.

TRANSACTION PROCESSING MONITORS AND SCALABLE SOLUTIONS

Scalability, reliability and manageability are the hallmarks of TP monitors. Before TP monitors existed, an operating system had to set up a unique and separate execution environment for incoming requests for an application, and the application itself. Execution environments typically consist of memory, CPU cycles, and communication and database connections. These constitute the system resources required for request and application processing.

As online transaction-processing applications began to emerge, the number of incoming requests soared. The system resources available to process these requests were quickly exhausted. TP monitors were created to allow the same system resources allocated to one request for an application to be shared concurrently by multiple requests. At the same time, the application logic executed in response to a request is shared as well. This enables hundreds of requests to be processed simultaneously using the same amount of system resource previously used to execute a single request. Resource sharing is at the heart of application scalability.

TP monitors work with databases to provide transaction integrity. Users performing a logical transaction are guaranteed that either all the steps complete successfully, or the transaction does not complete at all. Another aspect of reliability is fault tolerance, which guarantees that if a failure occurs in the execution environment, the TP monitor restarts what failed in a way that is transparent to the end user and does not affect transaction integrity.

TP monitors have a management interface that allows system mangers to configure, monitor and control the overall processing environment. System managers use the interface to specify things like the maximum number of requests a particular application process can handle, the location of the database, the priority associated with certain requests, etc. Advanced TP monitors allow maintenance operations, such as software updates, to be performed while the application is online. This increases overall application reliability.

ASSOCIATED CHALLENGES

While TP monitors are proven elements of enterprise solutions and are well recognized for their scalability and reliability, they also have these serious shortcomings:

- Because of the lack of open standards in this area, solutions built to a particular platform are locked into a single platform vendor forever.
- Like ORBs, TP monitors pre-date the web and therefore must be paired with HTTP listeners for web-based applications. Once again, the result is two platforms with two installations and two management consoles.

OAS 4.0 Responds to These Challenges

The challenges resulting from widespread use of Internet technologies are not new. Scalability, reliability, manageability, security, interoperability, and support for distributed transactions are usually addressed through the use of middleware products such as TP monitors, ORBs, message queuing products, and security and directory services. Oracle's strategy is to provide an application server that offers all of the required middleware functionality needed to respond to these challenges in a single, distributed product.

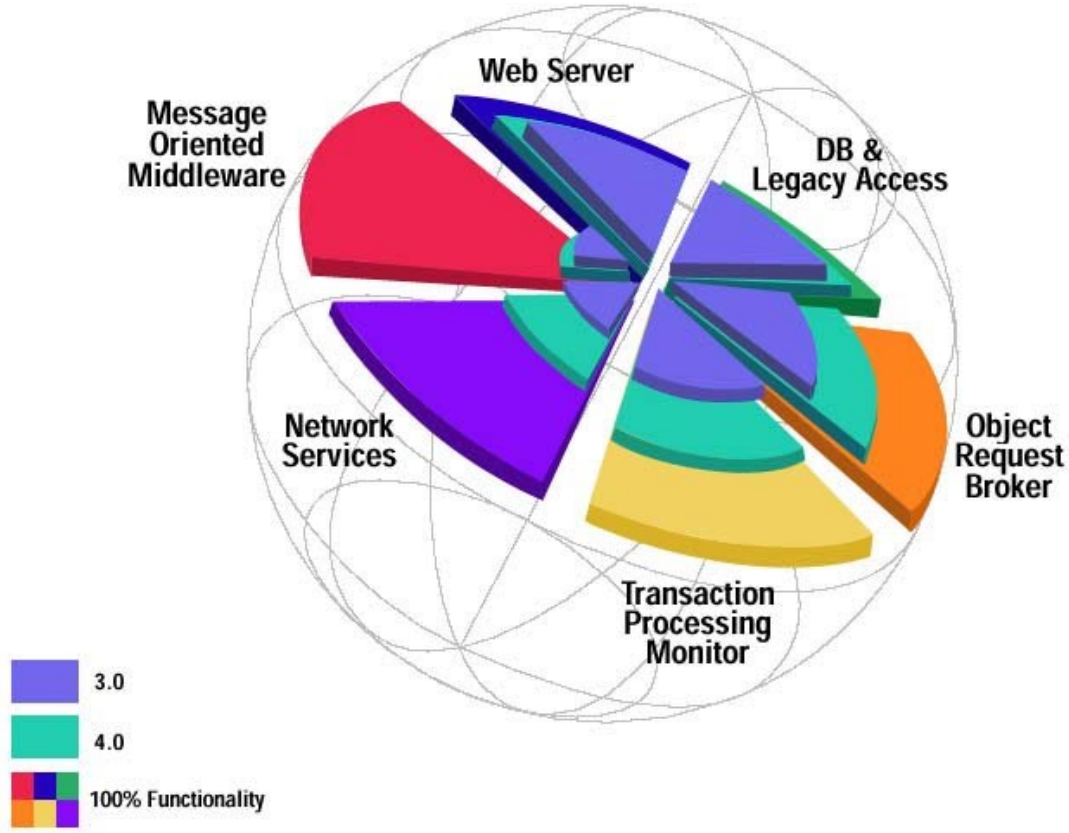

Figure 1: Unification of middleware capabilities in Oracle Application Server versions.

The following table provides a very brief summary of the capabilities provided by six different middleware technologies. OAS 4.0 features that are equivalent to these middleware capabilities are listed. Many of these features are detailed in a subsequent section of this product overview called "Scalable, Reliable, Manageable and Secure."

Technology	Metaphor	What It Does	OAS 4.0 Equivalent Features
Web Server	Fast Food Chef	Responds to HTTP requests by serving up HTML pages.	• Includes an HTTP listener (the web server core) • Supports multiple other common web servers • Configurable and programmatic web sessions • Declarative and programmatic web transactions • LiveHTML transactions • SSL for privacy of messages
ORB (CORBA 2.0 Compliant)	Translator	Allows components written in different languages to transparently inter-operate across a network.	• Includes a CORBA 2.0-compliant ORB • Supports standard IIOP protocol • JTS transactions (compliant with OMG's OTS) • Interoperable with Visibroker 3.2 ORB

TP Monitors	Illusionist	Using sophisticated resource management techniques, hundreds of clients appear to each have 1-to-1 connections with a data server. Also guarantees transactions and offers scalability, reliability.	• Scales to support large pools of end users
			• Multi-threaded cartridges support increase in processing demand
			• Supports two-phase commit transactions
			• Leverages OLTP features of Oracle DBMS when using OCI via PL/SQL and C:
			  • Multiple users share single connection
			  • Pooling drops idle connections

Security and Directory	Card Key Reader	Intercedes between incoming requests and requested resources to enforce access policies.	• Basic authentication (username/password) • Domain-based authentication (enable/disable users from particular domain) • IP based authentication (enable/disable users from a particular IP address/node) • Support for LDAP access to directories, used to authenticate users with X.509 certificates • Integration with Oracle database authorization • Supports 'virtual paths' which map URLs to non-static pages (CGI scripts, components, transactions) • Access to virtual paths can be secured via access control lists

Database Access Middleware	Gateway	Facilitates access to data stored in different databases.	• Access to Oracle databases via:
			  • OCI (Oracle Call Interface) in PL/SQL, C
			  • Sharing of pooled connections
			• Access to Oracle and non-Oracle databases via ODBC, JDBC, JSQL, X/A

Messaging	Postal Service	Instead of sending requests directly between clients and servers, the requests can be dropped into a mailbox queue, and delivered as posted (e.g., certified, priority, bulk mail)	• A recent, though growing, trend is to integrate applications using point to point or publish and subscribe communications. The ability to integrate applications in this manner will be supported in future versions of OAS.

ORACLE APPLICATION SERVER (OAS) 4.0

Overview

In a single sentence, the goal of OAS 4.0 is to provide a powerful infrastructure for deploying applications so that developers can focus on solving business problems rather than dealing with the low-level systems plumbing. As the middle tier of Oracle's network computing architecture, it is the core of any network computing solution.

OAS is situated between any client device that communicates over HTTP or IIOP, and any database that supports the ODBC or JDBC gateways. In addition to the middleware capabilities described in the previous section, OAS 4.0 supports multiple programming models (Web, CORBA/EJB) and languages (Java, Perl, C, PL/SQL, LiveHTML and Cobol). Open standards are supported across the board. When deployed on OAS, applications are automatically scalable, reliable, manageable and secure. Customers can begin writing simple web-enabled applications and migrate to component-based service architectures using the same OAS 4.0 infrastructure.

OAS Architectural Overview

Let's begin with a short overview of OAS's internal architecture before describing the features and benefits provided. An OAS solution has three distinct parts (see figure 2):

- HTTP listeners
- OAS
- Application cartridges

For maximum flexibility, scalability and reliability, these parts can be distributed over multiple hardware platforms.

HTTP LISTENERS

HTTP listeners/servers are excellent at handling requests for static pages and CGI. OAS 4.0, which bundles its own listener, can also work with non-Oracle listeners such as Apache, Netscape and Microsoft. All requests, other than those for static pages or CGI, are passed to OAS for handling.

OAS

The application server itself provides resource management when handling requests for applications deployed as cartridges on the server. It provides a common set of services for managing these applications, including load balancing, automatic recovery of failed processes, security, directory, and transaction.

APPLICATION CARTRIDGES

Cartridges are managed objects or components deployed on OAS. Each application cartridge runs within its own cartridge server process.

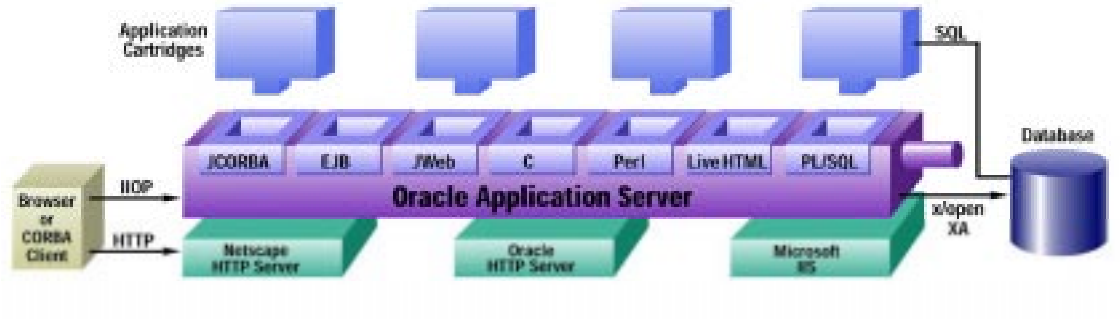

Figure 2: Oracle Application Server 4.0 Architecture.

OAS is a platform for a variety of environments that play host to business components built using various programming models. For example, the Perl and LiveHTML environments include an interpreter for Perl and dynamic HTML scripts, respectively. There is also an environment for executing business logic written in C.

In keeping with Oracle's "300% Java" initiative and goal of allowing a single language to serve across all tiers of a solution (client, application server and database server), OAS provides rich support for Java in the middle tier. Java environments include the JWeb environment, which furnishes a JVM for running Java components (JWeb components) accessible from web/HTTP clients. The JCORBA environment provides a JVM that runs Java components (JCORBA components) accessible from CORBA/IIOP clients These JCORBA components interface with the underlying CORBA technologies within OAS. It is important to note that the complexity of CORBA is hidden from the Java developer, who simply invokes Java classes. Finally, Oracle is enhancing the EJB environment to fully support the EJB 1.0 specification for running industry-standard EJBs in the next release of OAS.

One special-purpose environment is that provided for PL/SQL. Rather than including an engine for executing PL/SQL stored procedures, this environment links to PL/SQL stored procedures running on the database server.

A "cartridge" is Oracle's historical term for business components, i.e., a package of coded business logic. This term was invented before the advent of industry-standard component models such as EJBs. To reduce confusion, this product overview uses the term "component" rather than "cartridge," unless absolutely necessary. In figure 2, the cartridges shown are simply the business logic of the application hosted on OAS, often in the form of components.

Features and Benefits

OAS 4.0 is a comprehensive, open platform for web-based application development and deployment. As such, it offers the least-cost path to web and component-based applications. This improves time-to-market, increasing the buyer's ability to gain a competitive advantage in their market. In addition, applications deployed on OAS 4.0 have built-in enterprise quality-of-service (scalability, reliability, manageability and security), which allows greater focus of resources on building and deploying applications. Other features and benefits include:

Feature:	Client-independent application platform (any hardware device running browser over HTTP, or CORBA object over IIOP).
Benefits:	Reduce costs through use of inexpensive hardware, flexibility to choose non-PC clients without re-writing code, total cost of ownership (TCO) savings by using browser-based thin clients.

Feature: Unified alternative to multiple, point-solution, middleware products.
Benefit: Complete middleware solution, installed and managed as a unified whole. More cost-effective and less complicated than integrating a collection of piece-parts.

Feature: Single platform for leading development languages/models (Java, Perl, C, PL/SQL, COBOL, LiveHTML, CORBA objects and EJBs).
Benefits: Leverage existing skills in languages and tools (avoid developer re-training), language independence avoids lock-in, easy access to benefits of components while complexity hidden from developers.

Feature: Single platform for data and application integration (OCI, ODBC/JDBC/XA, PL/SQL Java class wrappers, COBOL cartridge and 3270 terminal emulators).
Benefits: Leverage existing database and applications resources, improving return on investment in existing technology; a compelling growth path for current users of two-tier systems.

Feature: Single platform to install and manage.
Benefits: Improve system administrator productivity through reduced complexity and better ease-of-use. Increase application availability by reducing time to detect, reconfigure and repair potential problems.

Feature: Choice of hardware, operating system, HTTP listener, toolset, international language.
Benefits: Reduce costs by leveraging existing technology, minimize lock-in by maximizing choice.

Feature: Based on open and de facto standards (HTTP, IIOP, Java, OTS, JTS, X/A, ODBC, JDBC, JSQL, SSL, LDAP).
Benefits: Maximize return on investment in technology, synergy with standards supported by the software vendors such as IBM, Sun and Netscape.

Common Application Types

OAS is well suited to three main application types: web-based applications, component-based applications and enterprise application integration. Customers can use OAS to write web applications that are static, dynamic or interactive in nature. At the same time, they can use OAS to write component-based applications in Java. Existing enteprise applications can be integrated through OAS to new applications. All of these application-types can co-exist on a single deployment of OAS.

This is good news for customers who wish to maintain flexibility in choosing development and deployment platforms for their business logic, without the inconvenience and expense of integrating a collection of middleware products.

WEB-BASED APPLICATIONS: OVERVIEW

Customers are choosing to deploy applications on the web for a variety of reasons. Most begin by publishing corporate information on an external web site as a way to advertise and promote their products. Next, or in parallel, they reduce the manageability costs of fat clients used to access internal applications by replacing them with browser-based interfaces. After that they may deploy new web-based applications as a way to provide customer service and support, or as a way to connect with vendors and suppliers. Finally, some customers are using web-based applications as new ways to conduct business. Travel planning, online banking, news delivery and e-commerce are some examples.

Web servers do a great job of providing navigation for HTTP requests and serving up HTML pages. Scripting languages can be used to enhance static HTML pages with dynamically executed content. Customers can choose from dozens of different tools to develop web applications.

However, web servers and their associated tools do not meet the requirements for enterprise applications. They are limited in the kinds of services they provide to applications. They do not scale well, are not robust and do not support end-to-end business transactions.

Once a customer has moved past the early stages of web application deployment, problems with response time, down time, performance and throughput begin to emerge. Some respond to these problems by simply replicating web servers and their static content, lock stock and barrel, over multiple machines. Others rely on loading up in-memory processing as a way of improving performance. These kinds of techniques provide some relief to the problems described earlier but, in the end, they are temporary fixes. If used carelessly these workarounds can actually exacerbate the very problems they are meant to solve.

Problems with response time, down time, performance and throughput are best addressed early in the development process through the use of a solid infrastructure that supports the requirements of enterprise. Customers can struggle to build such an infrastructure from multiple middleware products in a piece-part fashion, or they can avoid becoming system integrators by using OAS.

OAS FOR WEB-BASED APPLICATIONS

OAS offers the least cost path to web-enabled, enterprise-strength applications. Clients can be any hardware device running a browser that communicates with the application server via HTTP. Dynamic and static content, as well as business logic can be developed using HTML, scripting, or procedural languages such as Java. The interactions between the client and server can be simple request-reply, session or transaction oriented (see the later section entitled “Scalable, Reliable, Manageable and Secure” for more details). Access to databases can be via Oracle’s OCI or the ODBC, JDBC and X/A industry standards. These capabilities, along with the enterprise services described in a subsequent section, provide a solid foundation for web-enabled business-critical applications.

COMPONENT-BASED APPLICATIONS: OVERVIEW

The Internet has been described as the “killer application” for objects, and is accelerating the move to component-based computing. Gartner Group states that by 2001, more than three-quarters of new applications will be built, in part, using pre-existing components, without direct exposure to low-level procedural middleware APIs (0.7 probability).[3]

Components are software modules that implement a set of related functions, encapsulate only the required logic and data for those functions, and publish an interface that allows access to the functions by other components and programs. The major benefits of component-based applications include:

- Reuse -- Components are written once and used over and over again to assemble new applications. This improves programmer productivity and results in better quality.

- Flexibility -- Components can be modified without changing the interface or affecting other components. Instead of re-writing a monolithic application to make incremental changes, only the

[3] “Object Transaction Monitors: The Foundation for a Component-Based Enterprise,” Gartner Group Strategic Analysis Report, 8/97

appropriate components need to be touched. This allows applications to be quickly and easily adapted to changing business environments.

- Extensibility -- Components are easy to extend by adding new logic and data and extending their interfaces, while offering backward compatibility and smooth migration. New capabilities can be added to an application while existing ones remain continuously available.

Because components allow software applications to be separated from underlying heterogeneous platforms, it is possible to think of them as business services that can be deployed and re-used throughout the enterprise. Business processes can be modeled using components and built into what are called service-based architectures. These architectures provide an overall design philosophy for reusable software that addresses the needs of the business, not the technology.[4]

OAS FOR COMPONENT-BASED APPLICATIONS

Customers moving to components will find OAS 4.0 advantageous in several ways. Programmers can use the JDeveloper tool (available with the product) to write in Java and package the code as a business component. That logic is deployed in JWeb, JCORBA or EJB environments. Client objects communicate with OAS over IIOP and transactions are enabled through OTS (the CORBA object transaction service) and JTS (the simplified Java classes allowing access to OTS. Components and web-enabled applications can coexist simultaneously on OAS, making it an ideal platform for customers investigating component technology while building browser-based interfaces for existing applications.

ENTERPRISE APPLICATION INTEGRATION

In addition to choosing to deploy web applications, customers are choosing internet and CORBA technologies for application integration. The demand to integrate applications developed by different IT organizations is skyrocketing in the US, as a result of the largest wave of mergers and acquisitions since the turn of the century. In fact, merger activity will be close to $1.75 trillion in 1998, equal to one-fifth of the US GNP.[5] Application integration can also result from business process re-engineering or the desire to offer end users more functions from new web enabled interfaces.

OAS 4.0 offers a variety of means to integrate existing applications. PL2Java and PL2Bean are tools that allow existing PL/SQL stored procedures to be wrapped as Java classes, making them accessible to Java developers using OAS. Also, a COBOL cartridge and 3270 terminal emulators are available for legacy application integration. In conjunction with the Oracle8 database, transparent and procedural gateways enable access to legacy data. Transparent gateways provide access to data stored in flat files and non-Oracle relational databases, including DB2®, Informix® and Sybase/Microsoft SQL Server®. Procedural gateways provide access to transaction technologies such as IBM's CICS® and MQ Series®.

One of the most popular methods of integrating legacy applications with new solutions is through the use of CORBA-based technologies. As discussed earlier, the CORBA standards allow software written in various languages to be "wrapped." The wrapper defines a set of methods (roughly equivalent to function calls in traditional programs) that are visible to other components. These components, which may be on the same server or distributed across a network, are hosted on ORBs that communicate using IIOP.

---

[4] For more information, see *Component-Based Development for Enterprise Systems*

[5] "Greenspan Questions Antitrust Efforts," *Wall Street Journal*, June 10, 1998

The software being wrapped is often a legacy application that is left unchanged; the developer simply defines interfaces to the long-lived code. Since OAS 4.0 includes a CORBA 2.0-compliant ORB, it acts as an integration point between wrapped legacy applications and new component solutions. In addition, by embracing CORBA standards, OAS is able to interoperate with other compliant platforms. For example, Oracle has extensively tested interoperability with the Visibroker 3.2 ORB. Oracle is also dedicating resources to test OAS with IBM and other third-party vendor solutions.

SCALABLE, RELIABLE, MANAGEABLE AND SECURE

OAS 4.0 offers built-in enterprise-level quality of service. This allows customers to put greater focus on actually building and deploying applications, instead of dealing with infrastructure plumbing. The level of enterprise services in OAS distinguishes it from other products that claim to be platforms for business-critical application deployment. It will continue to be an area of ongoing differentiation for OAS.

Distributed Architecture

As discussed earlier, from an architectural perspective, an OAS solution has three distinct parts: HTTP listeners, the application server itself, and application cartridges (also called business components). For maximum flexibility, scalability and reliability, these parts can be distributed over multiple hardware platforms. For more information see the previous section entitled “OAS Architectural Overview.”

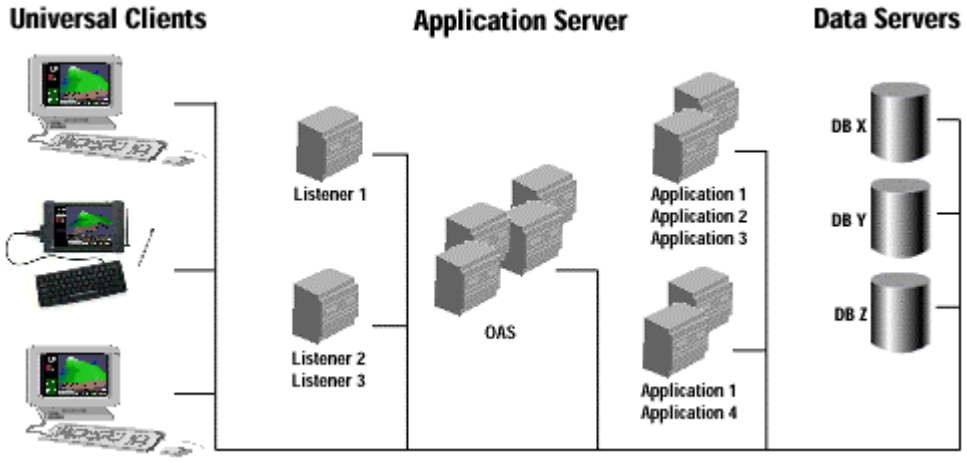

Figure 3: OAS 4.0 sections distributed over multiple systems, with different numbers of CPUs.

DISTRIBUTED ARCHITECTURE FOR SCALABILITY AND RELIABILITY

The OAS architecture allows it to be distributed over a cluster of machines. This distributed configuration gives system administrators the flexibility to add machines as application demands increase. Machines can be added to accommodate large numbers of users and high transaction rates. Applications can be configured to fail over to backup machines for improved reliability.

It is important to note that multiple heterogeneous HTTP listeners can be used in any given configuration. For instance, a configuration could include one Netscape listener, one Microsoft listener, and one Oracle listener on different nodes.

Scalability

An application's ability to scale dynamically can be critical for customers. Multi-threaded cartridges and the new load balancing features in OAS 4.0 allow applications to provide better response time to end users during peak loads, and better throughput and performance during peak processing times.

MULTI-THREADED CARTRIDGES

One of the new features in OAS 4.0 is the ability to support multi-threaded cartridges. Cartridge servers can be configured to have multiple instances of a single cartridge. Instances belonging to the same cartridge server can have multiple threads associated with them to serve multiple requests from different clients. These threads are shared across instances.

If all cartridge instances are busy servicing current requests, a new instance of that cartridge is automatically started—provided the maximum number of instances configured by the administrator has not been exceeded. When the request load drops, OAS shuts down cartridge instances to conserve system resources, e.g., memory. These features provide dynamic scalability for better throughput and performance.

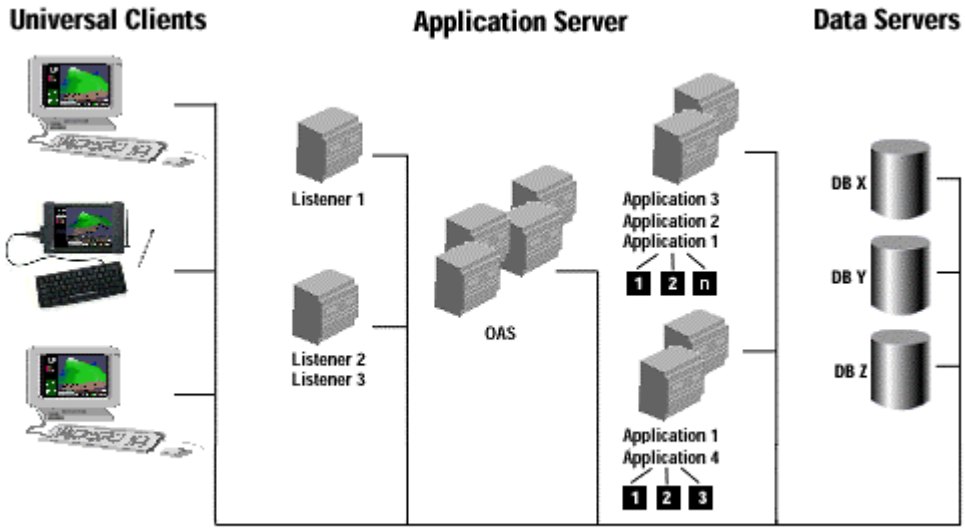

Figure 4: Application 1 and 4 in this scenario are configured to have multiple instances.

LOAD BALANCING

Web Application Server 3.0 was one of the first products in the industry to introduce load balancing in the form of round-robin scheduling. OAS 4.0 uses several new algorithms and techniques to provide more sophisticated load balancing. Mathematical formulas have been constructed to determine which cartridge instance should handle each request, making efficient use of all resources available for running cartridges. There are also two new mechanisms available in OAS 4.0 for load balancing:

- Configurable, weighted load balancing by application

  This is the ability to specify the percentage of instances of the application on a given node, relative to the other nodes in a cluster. This is especially useful when running the application server in a distributed configuration on nodes with different capacities. For instance, if the application server is configured to run on three nodes, the administrator can direct a certain

percentage of requests to each cartridge, e.g. 30% to node 1, 50% to node 2, and 20% to node 3.

- Dynamic load balancing by system metrics

  Here, system load information on each node in the distributed application server installation is gauged before OAS decides where the next request should be routed. In particular, the swap space, memory, and CPU utilization on each node in the system is factored into the decision. The load balancing protocols are designed to allow additional nodes to be added to an OAS installation without disrupting existing load balancing. The load is appropriately directed to the new host, based on the system configuration.

Reliability

As more and more business-critical applications are deployed on the web, reliability increases in importance. The fault-tolerant architecture of OAS 4.0 protects against any single point of failure, minimizing costly downtime. In addition, OAS 4.0 supports transactions, which are used to guarantee the reliability and integrity of client interactions involving business-critical information.

PROCESS ISOLATION

Applications written to HTTP server extensions are linked into a single, monolithic process. If one application in that stack fails, it can end up crashing all the applications that may be running on that HTTP server. The OAS process architecture is designed to make the system stable and robust. If one of many applications being serviced by OAS crashes, it does not affect any of the other applications. This feature was originally introduced in Web Application Server 3.0, and has recently been adopted by competitive products.

FAULT-TOLERANT ARCHITECTURE

The OAS architecture ensures that there is no single point of failure. Failover and recovery are possible whether a listener, the application server itself, or the application cartridge fails. An associated process detects the failure and initiates recovery. In-flight transactions are either fully completed (committed) or are backed out (rollback). Failure detection is implemented through heartbeat messages. The frequency of the heartbeats is configurable.

RELIABILITY AND CLIENT-SERVER INTERACTIONS

The architecture of OAS 4.0 lends itself to three basic approaches to application design: request-response (HTTP/IIOP), sessions (HTTP only) or transactions (HTTP/IIOP). For maximum reliability, interactions involving business-critical information should be implemented as transactions. Non-critical and repeatable interactions can be either simple request-replies or sessions.

The Request-Response Model: HTTP

Clients send HTTP requests to HTTP listeners. Requests for CGI scripts and static pages are handled directly by the HTTP listener, and not OAS. Other requests are passed on to OAS, which manages the routing of the request to the appropriate application cartridge. Requests for applications are handled by an available instance of that application cartridge and the result is passed back to the browser. After the result is returned, the connection between the browser and the web server is dropped, and the context for the interaction between the browser and server is lost. This is called a “stateless” request.

The Request-Response Model: IIOP

JCORBA components, written in Java and hosted on the underlying CORBA infrastructure of OAS, are accessible to clients and other JCORBA objects through IIOP. IIOP requests may come from

browsers that include an ORB, for instance Netscape Communicator®, which includes Visigenic VisiBroker for Java.

If a client does not have a built-in ORB, an IIOP connection can still be established. The client's initial HTTP request solicits the download into the browser of a Java applet. This applet then establishes an IIOP connection to OAS, which routes the request to the appropriate JCORBA object. The connection is persistent until the results are processed and sent back to the client.

HTTP: The Session Model

Web applications can use a session mechanism in OAS 4.0 to maintain a persistent association between browsers and a particular application cartridge instance. This is useful in particular situations where a logical interaction between an end user and an application consists of multiple, interrelated HTTP requests (and responses). For example, if the action taken upon the second request is different depending upon the results of the first request, context (i.e., state) must be maintained. Alternative solutions require the state to be passed back to the client, over the network, in the form of a (potentially large) cookie.

The much better solution offered by OAS is to maintain context on the server. But on a distributed OAS system consisting of multiple nodes and multiple instances of application cartridges, which server/cartridge should maintain the context? The solution is to enable the first and subsequent requests from the same browser to be handled by the same cartridge instance. This means the context is saved by the cartridge instance and is available locally to the application upon each request from the client. In this model, the session persists until the application has been idle for a configurable timeout period or until the client or server breaks the connection.

HTTP and IIOP: The Transaction Model

A transaction is similar to a session in that it usually consists of multiple HTTP requests. However, unlike sessions, transactions are used to ensure that interactions involving business-critical information either complete entirely and permanently, or not at all.

The classic transaction example is the transference of funds between a checking account and savings account in a bank. The transfer actually consists of two discrete steps: withdrawing the money from checking and then depositing the same amount in savings. If one step succeeds and the other fails, either the bank or their customer lose money. The desired result is that either both steps succeed and the transaction is "committed" (the money is transferred), or that the transaction is rolled back (the transfer fails but the money is not lost). In mainframe and client-server solutions, this capability is old news. Unfortunately, it has never been news, old or recent, for traditional web servers.

OAS supports both declarative and programmatic transactions. Declarative transactions are "declared" at runtime through configuration options for cartridge environments. They do not require explicit programming. These "no code transactions" are simple and easy to use. Programmatic transactions are written explicitly into the application logic using transactional APIs. These APIs include key words that describe when a transaction begins and ends, and when it should commit or rollback. The actual behavior of a transaction can be tailored to meet specific circumstantial criteria when the application is programmed.

The way transactions are implemented in OAS 4.0 depends on the language used to write the application.

C applications	Programmatic	Transactions are written using the transaction service API.
	Declarative	Virtual paths are associated with transaction demarcation operations (begin, commit, rollback). When a client requests the virtual path, the transaction automatically executes.
CORBA and EBJ objects	Programmatic	Transactions are written using the Java Transaction Service API.
	Declarative	Transaction demarcation is done by setting object attributes on or off.
LiveHTML	Tagged pages	Perl scripts embedded in tagged pages are automatically transactional.
PL/SQL	Declarative	Using the transaction service for the PL/SQL cartridge, URLs are associated with transaction operations (begin, commit, rollback). When a user invokes the URL, the corresponding operation automatically executes.

Manageability

OAS and applications built for OAS are easy to manage because OAS is a single platform that unifies multiple single-purpose middleware technologies, allowing them to be managed from a single console. OAS 4.0 has a new, easy-to-use management interface that improves administrator productivity. Application availability can be monitored from Oracle's Enterprise Manager product, as well as from SNMP-based management applications.

ONE MANAGEMENT ENVIRONMENT

The OAS 4.0 management console has a new user interface and provides all management capabilities through one comprehensive tool. The interface is Java/HTML-based, easy to use and intuitive, and enables application developers and system administrators to configure, start/stop, and monitor listeners, application servers and cartridges. The console also provides mechanisms for configuring the system for optimal load balancing, performance tuning and enabling/disabling security in applications.

ENTERPRISE MANAGER INTEGRATION WITH OAS 4.0

The OAS 4.0 console can be launched from Oracle's Enterprise Manager, allowing operators the ability to configure, monitor and manage OAS applications from the same console used to monitor Oracle database products.

SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

Oracle SNMP support enables OAS 4.0 applications to be discovered, identified and monitored by any SNMP-based management application. OAS 4.0 ships with an SNMP MIB that includes configuration, status and statistical information about OAS components. SNMP traps are generated when OAS processes unexpected events.

Security

OAS 4.0 naturally supports the multiple authentication mechanisms available earlier with Web Application Server 3.0. These include:

- ***Basic Authentication*** – user name and password
- ***Domain-based Authentication*** - enable/disable users from a particular domain
- ***IP-based Authentication*** - enable/disable users from a particular IP address/node
- ***Database Authentication*** - applications can be authenticated based on the database user id and password

Application and system security available in OAS 4.0 has been significantly enhanced to secure not only web clients but communications between CORBA objects. Support for X.509 certificates and storage of certificates in a standard LDAP directory have also been added. Instead of entering and managing all the certificates of all the potential users for the applications running on OAS, it is now possible to configure OAS to refer to a shared Lightweight Directory Access Protocol (LDAP) directory.

New security features include:[6]

- Secure Sockets Layer v.3 (SSLv3) protection of IIOP traffic from client objects to server objects or to the embedded OAS 4.0 ORB.
- Support for industry-standard X.509v3 digital certificates.
- A "Wallet Manager" for certificate and trust-point management.
- Identity-based access control using X.509 based certificates stored in an industry-standard LDAP directory.

SSL AND X.509 CERTIFICATES

To protect client/server communications, OAS 4.0 utilizes Secure Sockets Layer version 3 (SSLv3), the most widely used Internet security protocol. SSL encrypts data on the network, preventing attackers from viewing or tampering with HTTP or IIOP data exchanged over the Internet. Moreover, Oracle's SSL uses industry-standard X.509v3 certificates for secure connection setup and mutual client and server authentication. This prevents attackers from masquerading as legitimate users and gaining unauthorized access to the system.

WALLET MANAGER

To facilitate installation and use of X.509 certificates and protect key material on the server, OAS includes a Wallet Manager tool. The Wallet Manager allows the server administrator to store certificates and associated private keys in an encrypted wallet file, decrypting and activating them for SSL use as needed. Since certificates may be issued by multiple certificate authorities (CAs), the Wallet Manager allows the administrator to install and manage specific CA trustpoints, specifying which CAs are trusted to issue client certificates for authentication to OAS.

ACCESS CONTROL AND LDAP

Although authenticating users is critical for security, a system must also control which resources users can access on the server. Access to OAS 4.0 cartridges is controlled through the use of an LDAP directory and Access Control Lists (ACLs).

---

[6] The RSA home page has background information on Internet-related security topics

Cartridges are associated with virtual paths—similar to URLs—that identify a cartridge instead of a static HTML page. Each virtual path is associated with a certificate realm. The realm points to an LDAP directory server that contains certificates of prospective users, as well as an ACL that lists the groups or individuals entitled to use the cartridge associated with that virtual path.

The following example may help clarify this methodology. When a user requests access to a cartridge through a virtual path, the user's certificate is extracted and authenticated using the LDAP directory specified in the virtual path's certificate realm. Then the realm's ACL is checked in order to authorize or reject the request.

OAS 4.0 has been certified for use with Netscape Directory Server. Certifications for other LDAP-compliant directory servers are planned.

SUMMARY

The widespread use of Internet technologies by applications running on both public and private networks has created requirements not met by traditional web servers. These requirements include scalability, reliability, manageability, security, interoperability, directory and transaction services and database access. In the past, these requirements were met by cobbling together various single-purpose middleware products. Today, these requirements are met by a new class of product called an application server.

OAS 4.0, the first entrant and current market leader, provides the middleware features required by enterprise applications in a single product that is easy to install and manage. This single platform supports the industry's leading development models and languages, which can be used to write applications for any kind of client, either web browsers or true object-oriented clients. Applications built on OAS can access both Oracle and non-Oracle databases.

Offering "built-in" enterprise-level quality of service, OAS 4.0 is the least-cost path to both web-enabled and component-based applications. It is the right choice for Oracle's tools, applications and Enterprise Manager products. And it is the right choice for customers who wish to maintain flexibility in choosing development and deployment platforms for their business logic, without the inconvenience and expense of integrating a collection of middleware products.

WHY CHOOSE ORACLE?

Oracle is the second largest software company in the world, and the industry's leading database vendor. It has proven experience and expertise in industry applications and development tools. The company's stature in the industry is enhanced by its financial health and strong third-party relationships. Oracle works hard to ensure the success of its customers through broad educational offerings, a large, worldwide consulting organization and 24-hour global support. It is also doing its part to ensure that customers have viable alternatives to Microsoft by supporting open standards such as EJBs, and by actively participating in standards bodies such as the OMG.

OAS 4.0 already has thousands of licensed customers around the world. InfoWorld named OAS its Server Software Product of the Year for 1997. VARBusiness selected OAS as the Product Report's #1 Internet Server Software for 1997. OAS also received InfoWorld's 1997 "Best of the Test Center" #1 Analyst's Choice Award. This industry recognition highlights the wide support for Oracle's compelling vision of web and component-based enterprise computing.

Appendix: Language Cartridges and Tools

Oracle offers a range of tools to assist programmers in building applications for OAS 4.0. JDeveloper is used to write Java for Web, CORBA or EJB cartridges. Oracle Designer can be used to generate applications for the PL/SQL cartridge. Oracle Developer can be used to generate web-enabled object code. Other tools mentioned in the following tables are available from third-party partners of Oracle.

Table 1: Developing Content HTML Applications on OAS 4.0

Language	Tools Available	Benefits
HTML	HTML Editor: Symantec VisualPage (included with OAS 4.0)	Easy HTML application development Out-of-box solution
Perl	Perl: Standard Perl Support	Large installed base, easy to use
Java	Java IDE: Oracle JDeveloper Suite, Oracle JDeveloper	More customized clients Easy application development Wizard technology for common tasks Presentation and application logic can be more interactive and sophisticated
LiveHTML (Server Side Includes)	HTML Editor: Symantec VisualPage	Ease of embedding application logic in HTML Support the scripting paradigm through server side scripts for including dynamic content within HTML pages, effectively separating designer from developer. Access to other applications in Oracle Application Server using InterCartridge eXchange (ICX), e.g., inclusion of client information from a payment cartridge

Table 2: Languages and Database Application Models

***Database*: Oracle Database Server**	
***Language*:**	**PL/SQL Application Cartridge: Native access to Oracle Database Server**
***Benefits*:**	Leverage existing skills to web publish from Oracle database servers
	Embed application logic and database access within an application
	Scaleable and secure performance solution
	Business logic stored in database as stored procedures
	Easy toolset for generating HTML
	Re-use existing stored procedures with minor changes for HTML display
	Supports configurable enterprise transaction service for Web clients
Language:	**PL2Java: Java access to Oracle Database Server**
Benefits:	Application logic in both PL/SQL and Java
	Reuse existing PL/SQL packages without any modification
	Wizard support in AppBuilder for Java
Language:	**C Application Cartridge - Pro*C and OCI: Access to Oracle Database Server**
Benefits:	Developer controls application logic and database access
	Supports transactional applications through configurable transactions
	Supports transactions programmatically through the X/Open TX API
	Highest performance mechanism to access database through native Oracle interface

***Database*: Oracle Database Server and any ODBC-Compliant Database**

Language:	**JWeb Application Cartridge: Java JDBC: SQL Access to Databases**
Benefits:	100% pure Java applications supported
	JDBC access to databases
	PL2Java supported for access to PL/SQL stored procedures through Java class wrappers
	Application logic in Java
	Output HTML using Java web toolkit
	Integrated with Oracle JDeveloper
	Integrated development and deployment wizards
	Support for Java Transaction Service (JTS)

Language:	**EJB Cartridge, JCORBA, JDBC: SQL Access to Databases**
Benefits:	Industry-standard distributed component model
	CORBA development and deployment platform of choice
	Enables CORBA developers access to Oracle
	Access to CORBA through Java
	Database access using JDBC
	Access to transaction services such as JTS/ OTS

Language:	**Perl Cartridge: Perl Access to Databases**
Benefits:	Easy access through scripts
	Perl DBI access to databases
	CGI developers can leverage existing code and add new functionality

Language:	**ODBC Application Cartridge: SQL Access to Databases**
Benefits:	Industry standard
	Easy to use

***Database*: Rdb Database Server**

Language:	**Rdb Application Cartridge: Access to Rdb Databases**
Benefits:	Fastest and easiest way to access Rdb databases

Oracle Application Server 4.0 White Paper:
Product Overview
September 1998

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
+1.650.506.7000
Fax +1.650.506.7200