

Oracle® Application Server

Overview

Release 3.0.1 for Windows NT

Overview for Windows NT

Oracle Web Application Server 3.0.1

Copyright © 1997, 1998, Oracle Corporation. All rights reserved.

The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be licensee's responsibility to take all appropriate fail-safe, back up, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the Programs.

This Program contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright, patent and other intellectual property law. Reverse engineering of the software is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation

If this Program is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication and disclosure of the Programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-14, Rights in Data -- General, including Alternate III (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

Oracle is a registered trademark, and the Oracle logo, NLS*WorkBench, Pro*COBOL, Pro*FORTRAN, Pro*Pascal, SQL*Loader, SQL*Module, SQL*Net, SQL*Plus, Oracle7, Oracle8, Oracle Enterprise Manager, Oracle Call Interface, Oracle7 Enterprise Backup Utility, Oracle TRACE, Oracle WebServer, Oracle Web Application Server, Oracle Application Server, Oracle Network Manager, Secure Network Services, Oracle Parallel Server, Advanced Replication Option, Oracle Data Query, Cooperative Server Technology, Oracle Toolkit, Oracle MultiProtocol Interchange, Oracle Names, Oracle Book, Pro*C, and PL/SQL are trademarks or registered trademarks of Oracle Corporation. All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.



Overview

As the World-Wide Web (WWW) evolves from a platform that serves static documents to one that runs distributed, cross-platform, dynamic applications, Hypertext Transfer Protocol (HTTP) web servers play an increasing role.

The Oracle Web Application Server is an HTTP server that enables you to create and deploy such applications. It provides a framework that encompasses a modular distributed architecture, an open API that enables you to create portable applications, and different application models or paradigms.

Contents

- [Features of the Web Application Server](#)
- [Architecture](#)
- [Handling Requests](#)
- [Authentication Service](#)
- [Security Summary](#)
- [Differences Between the Advanced and Standard Versions](#)
- [File Format Negotiation](#)

Features of the Web Application Server

- **Support for a distributed environment** - You can run the components of the Web Application Server on different machines on the network to improve performance and scalability. For example, if you have many requests coming in and they are getting queued up at the Listener, you can increase the number of Listeners without increasing the number of applications.
- **Extensibility through a portable and open API** - The Web Application Server enables you to customize or extend the capabilities of the web server by building

your own applications using an open API. The API has been ported to platforms supported by the web server, so to run your applications on multiple platforms, you just need to re-compile them for each platform.

- **Fault-tolerant applications** - Since each application is its own process, it is protected from other applications. If an application misbehaves, Listeners and other applications are not affected.
- **Transactions** - The Web Application Server supports the XA open model transactions as defined by the X/Open Company. This model allows you to perform transactions on a database that span requests. In earlier versions of the web server, each request begins and completes a transaction.
- **Intercartridge exchange (ICX)** - This feature allows applications to communicate with each other. You can build modular applications that take advantage of this feature: common components of your applications can be factored out and implemented as a separate application, which can be used by other applications.

Architecture

The Web Application Server consists of the following components:

- [HTTP Daemons \(UNIX\), or Listeners](#)
- [Web Request Broker \(WRB\)](#)
- [Cartridges, or Server-Side Applications](#)

HTTP Daemons (UNIX), or Listeners

The HTTP daemon (UNIX) or Listener component of the Web Application Server listens for requests from clients. The Listener uses HTTP to communicate with clients, and it can accept connections on one or more IP address/port combinations. See [File Format Negotiation](#) for a list of file negotiation features in the Listener that is shipped with the Web Application Server.

You do not have to use the Listener that is shipped with the Web Application Server. You can use web servers from Netscape or Microsoft as the Listener component. See the Installation Guide for details on how to configure other Listeners to use with the Web Application Server.

Web Request Broker (WRB)

The Web Request Broker (WRB) keeps track of the execution of the entire system. It handles load balancing tasks, tracks global resources such as available cartridges, and obtains the address (the object reference) of the required cartridge resource.

The WRB also provides system services that all cartridges can use. Some examples of these services are:

- The Virtual Path Manager, which maps virtual paths to cartridge types
- The Authentication Server, which authenticates clients. See “[Authentication Service](#)” for details.

- The Logger Service, which logs requests and messages in a file or database. The Logger Service can log in Common Log Format (CLF) or Extended Log Format (XLF).
- The Configuration Provider, which reads and stores information from the configuration file.
- The Intercartridge Exchange (ICX) Service, which enables applications to communicate with each other
- The Transaction Service, which enables you to perform transactions that span requests on a database
- The Content Service, which enables you to save documents in repositories

To access these services, cartridges use the WRB API. See the Cartridge Development section for a list of APIs.

The WRB is implemented as a CORBA-compliant object request broker. It can send requests to and get responses from applications running on machines on the network.

Cartridges, or Server-Side Applications

Cartridges are server-side applications that handle cartridge-specific requests from clients. The Web Application Server comes bundled with some cartridges. Some examples of these cartridges are:

- The PL/SQL Cartridge, which handles requests that need to run procedures and functions in an Oracle7 database
- The Java Cartridge, which runs Java applications (not applets) on the server-side
- The LiveHTML Cartridge, which parses HTML documents that contain “server-side includes” (SSI)
- The Oracle Worlds Cartridge, which enables you to build dynamic three-dimensional worlds using VRML (virtual reality modeling language)
- The Perl Cartridge, which runs Perl scripts
- The ODBC Cartridge, which enable you to connect to relational databases that support the ODBC (Open Database Connectivity) interface

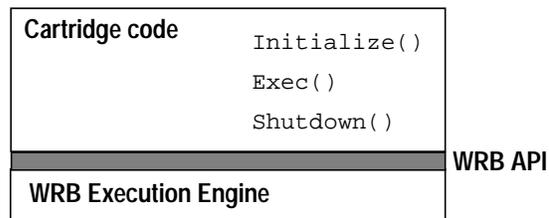
Each cartridge is independent of one another. If a cartridge fails, other cartridges and components in the Web Application Server are not affected.

A cartridge instance is a process for a cartridge. You usually have multiple cartridge instances for each cartridge type. For example, you can have 25 instances of the PL/SQL Cartridge and 5 instances of the Java Cartridge. Instances can run on different machines on the network.

Cartridge Architecture

Cartridges are built using the WRB API, which allows them to access WRB system services, and contain a copy of the WRB execution engine. Graphically, a cartridge looks like the following:

Figure 1: Cartridge architecture



`Initialize`, `Exec`, and `Shutdown` are callback functions defined in a cartridge. These functions are called when the cartridge is started up, handles a response, or shuts down. See [Callback Functions in a Web Cartridge](#) for a list of callback functions. The WRB execution engine implements the WRB API.

Using the WRB API, you can build your own cartridges as well. To run them with the Web Application Server, you need to register your cartridge with the Web Request Broker. Currently, you develop cartridges using the C language. These cartridges are portable: they are supported on all platforms that the WRB runs on. See [Designing Your Application](#) for details on cartridge development.

Handling Requests

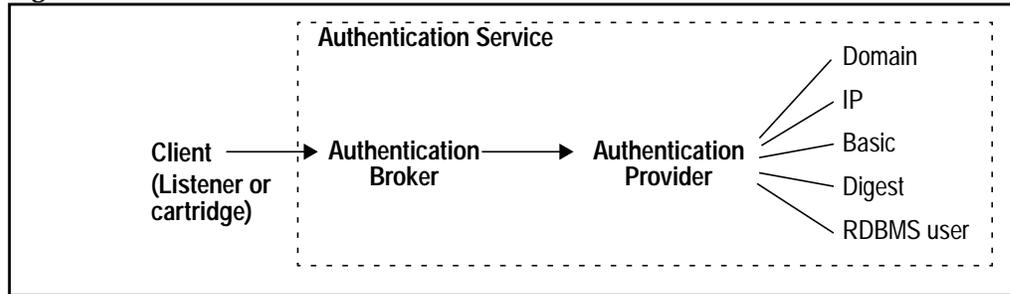
When the Web Application Server receives a request from a client, the Listener component processes the request first. It reads the request and determines how it should handle the request:

- If the request is for a static file, the Listener fetches the file from the operating system and sends it to the client.
- If the request is to execute a CGI file, the Listener executes the file and sends the results back to the client.
- If the request is for a cartridge, it performs the following:
 - a. Looks up the virtual path to determine the cartridge type that this request is for.
 - b. Calls the Authentication Service to authorize the client.
 - c. Sends the request to an instance of the appropriate cartridge. The cartridge instance processes the request, and returns the results to the client.

Authentication Service

The Authentication Service is one of the security features of the Web Application Server. It enables you to validate clients in several ways. The following figure shows the components in the Authentication Service:

Figure 2: Authentication Service



When a client wants to use the Authentication Service, it sends a request to the Authentication Broker specifying the authentication provider to use. The Authentication Broker then forwards the request to the appropriate provider, and the provider validates the request.

The client can be a cartridge or a Listener. For example, using the Web Application Server Manager, you can specify that cartridges are protected by one or more providers.

The Authentication Broker coordinates authentication requests and returns the results to the clients.

The Authentication Provider implements the actual authentication. Currently there are five authentication providers, as shown in the figure above.

- IP and Domain are restriction schemes, which allow access to files only if the client is from a list of registered IP addresses or domain names.
- Basic and Digest are authentication schemes, which prompt the user to enter a user name and password before the user's request is processed. The list of user names and passwords are stored in a configuration file on the primary node.
- RDBMS user, which is also called Basic_Oracle, specifies that the user name and password are verified by an Oracle database.

The broker and providers can be in memory or accessible via CORBA. If in memory, the broker and the providers are part of the client's process. The benefit of this is that it allows the client to access the broker and the provider very quickly since they are in the same process space. The trade-off is that the client's process becomes larger (and uses more memory), and if you have multiple Listeners or cartridges, data such as user names and passwords are replicated for all these processes. This might not be practical if you have hundreds of users and passwords.

Accessing the providers via CORBA means that the broker and the providers run as separate processes, and the client makes network calls to access them. This results in smaller Listeners and cartridges processes, and no replication of data, but the trade-off is that the client takes longer to access the broker (since network calls are required). However, this set-up allows you to run the Authentication Server on a separate machine on the network.

See the "[Authentication Server](#)" section in the "[Security](#)" chapter for more details.

Security Summary

This section summarizes the security features provided by the Web Application Server. See “[Security](#)” for details.

To ensure security, the Web Application Server supports the following:

- Secure Sockets Layer (SSL) Version 3, which enables the server to transmit encrypted messages. Clients use a secure version of HTTP called HTTPS to establish secure connections. To establish a secure connection with a Listener, a request must use “https” instead of “http” and must specify a port on which the Listener has enabled SSL. For example:

```
https://www.acme.com:443/
```

- Basic and digest authentication schemes, which prompt the user to enter a user name and password before processing the user’s request.
- IP-based and domain-based restriction schemes, which allow access to files only if the client is from a list of IP addresses or domain names.

Differences Between the Advanced and Standard Versions

The Web Application Server comes in two version: the Advanced version and the Standard version. The Advanced version is a superset of the Standard version; it contains the following features that are not available in the Standard version:

- The Transaction Service
- The Content Service
- The ODBC Cartridge
- Distributed cartridges - You can run cartridges on remote nodes only if you have the Advanced version. If you have the Standard version, the cartridges all run on the same node.

In the documentation, there is a note at the top of each section that describes the Advanced features to indicate if the feature is not available in the Standard version.

File Format Negotiation

The Listener that is shipped with the Web Application Server can maintain several versions of a document in different formats, and provide them to clients in the formats they prefer. If you are using another type of Listener (for example, Netscape’s server), the features listed here may not apply, and you should refer to your Listener’s documentation.

Language Formats

For example, if a client requests a document in French, the Listener checks to see if it has a French version of the document, and, if so, returns that version to the client.

Otherwise, the Listener returns the version of the document in its own default language (usually English).

MIME Formats

Similarly, a client can request a document of a particular Multipurpose Internet Mail Extensions (MIME) type. If the Listener can find a version of the requested file in the requested MIME format, it returns that version to the client. Otherwise, it returns the version of the file that has the smallest size.

Encodings

The Listener can also maintain versions of a document that have been encoded by programs such as a compression utilities. For example, if a client can uncompress a document that has been compressed by the gzip program, the Listener can return to the client the compressed version of document instead of the uncompressed document, saving transfer time.

Filename Extensions

The Listener uses filename extensions to identify a file's format, which can represent language, MIME type, or encoding. For ease of maintenance, it is best to advertise a file to clients only by its base name, allowing clients and server to negotiate formats transparently.

File Handling

The Listener uses a *virtual file system* to keep track of the files it makes available to clients. The virtual file system maps the pathnames used in request URLs to the file system maintained by the host machine's operating system.

File Memory Mapping

The Listener uses the host operating system's memory mapping capability when opening a file requested by a client so that the Listener's virtual address space refers directly to the file's contents. This speeds access, and makes it possible for several clients to share the same copy of the file, which conserves the Listener's memory resources.

File Caching

Ordinarily, when the Listener opens a requested file, the file remains open and mapped into memory until all clients using the file are finished with it, at which point the Listener closes the file and frees the memory mapping associated with it. The Listener allows you to specify files to be *cached*. Cached files are opened when a client requests them, and remain open for the life of the Listener process. This optimizes access times for files, such as your home page, that are requested often.

Index

A

advanced vs. standard versions, 6
architecture of Web Application Server, 2
Authentication Broker, 4
Authentication Server, 4

C

caching
files, 7
cartridges, 3
architecture, 3
compression, 7

E

extensions
filename, 7

F

features of Web Application Server, 1
file sharing, 7
files
caching, 7
compression of, 7
filename extensions, 7
language of, 6
MIME types, 7

H

HTTP daemons, 2

L

languages
file formats and, 6
Listeners, 2

M

memory
management of, 7

N

NLS
file formats and, 6

R

requests, handling, 4

S

standard vs. advanced versions, 6

U

URLs
specifying secure connections with, 6
use virtual file names, 7

V

virtual file systems
defined, 7

W

Web Application Server

advanced vs. standard versions, 6

architecture, 2

features, 1

overview, 1

Web Listeners

file type negotiation, 7

memory management, 7

Web Request Broker, 2