

Contents

- [Security Overview](#)
- [Authentication Server](#)
- [Security Schemes](#)
- [Listener Security](#)
- [Dispatcher Security](#)
- [Cartridge Security](#)
- [Database Security](#)
- [PL/SQL Cartridge Security](#)
- [Using the Web Application Server with Firewalls](#)
- [Setting Up Your Web Application Server to Use SSL](#)
- [Migration from Previous Versions](#)

Security Overview

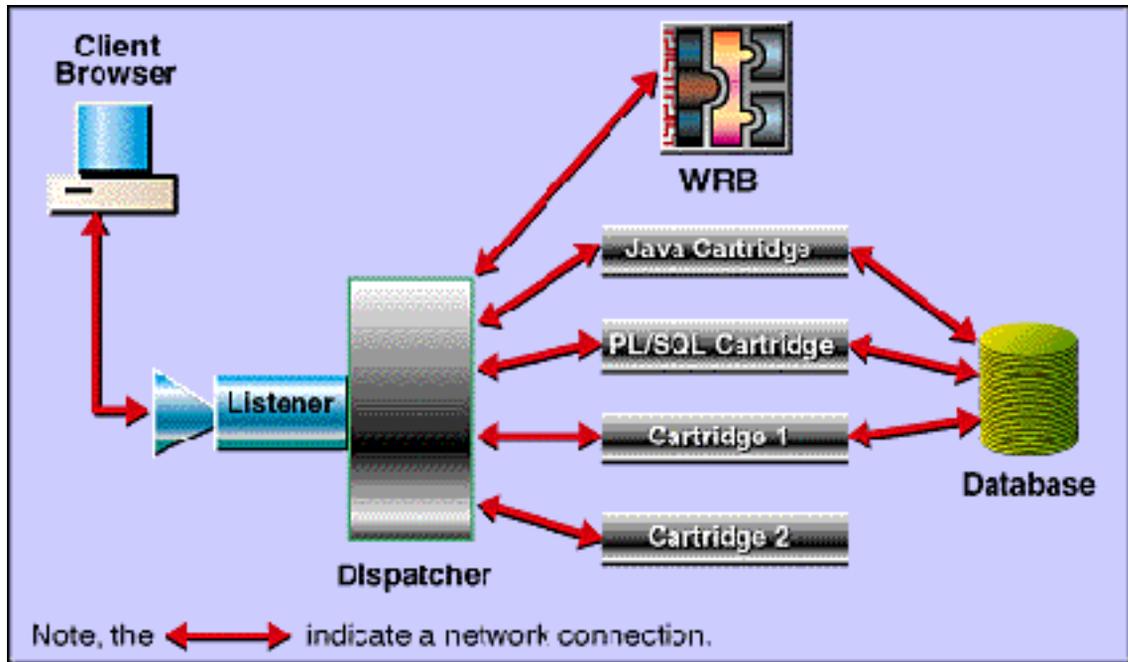
- [Security Architecture](#)
- [Security Features](#)
- [New Feature Summary](#)

Security Architecture

Oracle Web Application Server encompasses all the components shown in the following figure except the browser and the database (the Oracle7 Server). The

Oracle7 Server can be on a different machine than the Web Application Server, and the various modules of the Web Application Server itself can be distributed across a network as shown.

Figure 1-1 Web Application Server architecture



When the modules are distributed, they communicate over an Object Request Broker (ORB) called Oracle Media Exchange (OMX). An ORB is an object management technology based on the Common Object Request Broker Architecture (CORBA), an industry standard for cross-platform inter-operability between executable objects. If the modules are not distributed, they communicate using UDP.

The components:

- **Browser** - This can be a typical browser like Netscape Navigator, a Web spider, a robot, or another Web-based application.
- **Listener** - The HTTP daemon (httpd) that is listening for requests for service. For more information, see "[Listener Security](#)".

- **Dispatcher** - The dispatcher handles HTTP requests that require cartridges. There is one dispatcher per listener. The dispatcher is composed of various components including the Authentication Server, the Configuration Provider, the Logger, and so on. For more information, see “[Dispatcher Security](#)”.
- **WRB** - The Web Request Broker (WRB) contains information about and manages all the cartridges on the network. There is just one WRB on the network and it is responsible for creating and destroying cartridges. There can be many dispatchers connected to a single WRB.
- **Cartridges** - The cartridges are responsible for executing applications on the server and returning HTML content back to the browser via the dispatcher and listener. There is one set of cartridges, used by all listeners and dispatchers in the systems. For each cartridge, however, execution instances (WRBXs) are created and destroyed by the WRB as needed to handle the load. The result is a many-to-many relationship between cartridge instances and listeners. For more information, see “[Security Features](#)”.
- **Database** - This refers to the Oracle7 Server. Cartridges can interface with an Oracle database via OCI (Oracle Call Interface) or other protocols. If the database is on a different machine than the cartridge, then they communicate through SQL*Net. For more information, see “[Database Security](#)” and “[PL/SQL Cartridge Security](#)”.

Security Features

- It allows authorization of users accessing Web pages. This feature has the following characteristics:
 - If the request is for a WRB cartridge, the authorization will be performed by the dispatcher (see “[Dispatcher Security](#)”) or by the cartridge itself (see “[Cartridge Security](#)”). Either of these can employ the [Authentication Server](#).
 - The passwords passed over the Net can be encrypted, either through the Secure Sockets Layer protocol (see “[SSL Security](#)”) or through Digest Authentication (see “[Security Schemes](#)”).
 - If the HTTP request is for a static page or a CGI script, the authorization is performed by the listener (httpd) receiving the request. If it is for a WRB cartridge, the authorization performed by the dispatcher or by the cartridge itself. If the authorization is done by the dispatcher, the dispatcher invokes the Authentication Server and uses its configuration to authorize the request. Cartridge authorization can either use the Authentication Server or rely on an internal protection scheme. The main difference

between these approaches is that, if the dispatcher is used, the authorization is configured through the Web Application Server Manager. Since there is a separate dispatcher for each listener, this allows the administrator to set different authorization mechanisms for each cartridge based on the listener handling the request.

If the authorization is handled by the cartridge itself, on the other hand, the call is hard-coded into the cartridge, and the same protection scheme therefore applies to the cartridge regardless of which listener handles the request. You can combine the two approaches, for example, using cartridge authorization to define a minimum level of protection and the dispatcher for optional additional protection. See [“Dispatcher Security”](#) and [“Cartridge Authentication”](#) for details on these techniques.

- For requests that access Oracle7 Server through a WRB cartridge, the authentication can be based on usernames recognized by the Oracle7 Server without these actually being exposed to the client. This is achieved by having the URL specify a Database Access Descriptor (DAD). This is a configuration file on the server, which in turn specifies the database username and password. If a URL accessing the database specifies a DAD that does not include a username and password, the user is prompted for these by the listener. Roles can also be used to further control user actions. For more information, see [Database Access Descriptors \(DADs\)](#).
- The authorization that the listener can perform depends upon which listener you use. This section covers the listener (under [“Listener Security”](#)) that is bundled with the Web Application Server, but you can use others.
- If authentication is done through the Authentication Server, the usernames and passwords can (but need not) be taken from a database (Oracle7 Server), rather than a configuration file. This is called *Database Authentication*. For more on this feature, see [“Security Schemes”](#).
- Cartridges are managed by the WRB, and there is one WRB on the network, although there may be any number of listeners. Therefore, authorization done through the cartridge can apply simultaneously to any number of listeners and dispatchers. See [“Implications of Multiple Listeners”](#).
- For listener, dispatcher, or cartridge-based security, access can be restricted by IP address or domain. This can be combined with username/password authentication.
- HTTP traffic can be encrypted under the Secure Sockets Layer protocol (see [“SSL Security”](#)).

- Cartridges and listeners have a many-to-many relationship. There is a group containing any number of cartridges, and this group can be accessed by any number of listeners. Since access can be controlled at both the listener and the cartridge level, some thought needs to be given to this, as explained under [Implications of Multiple Listeners](#).
- The PL/SQL Cartridge can be configured to specify that certain packages cannot be directly invoked by a URL. Such packages can only be invoked by other procedures or functions or by users connecting to the database through some means other than the PL/SQL Cartridge. See [“Stored Procedure Access \(Protecting Packages\)”](#).

These features will mostly be used by Web Application Server Administrators, although application developers will have to use or at least be aware of some of them as well. Of particular interest to PL/SQL developers are the issues discussed under [“PL/SQL Cartridge Security”](#), and those developing cartridges will want to read [“Cartridge Security”](#).

Terminology Note: In this section, authorization refers to the security validation of an HTTP request. This can be achieved in two ways: authentication, which validates the identity of the user based on prompting for a username and password, or on checking for the presence of a digital certificate, and restriction, which only accepts HTTP requests originating from certain IP addresses or domain names.

New Feature Summary

Here is a summary of the security features that have changed between Oracle WebServer 2.x and Web Application Server 3.0. For information on migrating your existing security configuration to Web Application Server 3.0, see [“Migration from Previous Versions”](#).

- The listener for WebServer 2.x supported SSL 2.0. Web Application Server 3.0 now supports SSL 2.0 and 3.0. For more on SSL, see [“SSL Security”](#) and [“Setting Up Your Web Application Server to Use SSL”](#).
- In WebServer 2.x, all authorization was performed by the listener itself. The listener used authorization mechanisms specified for a cartridge’s virtual path. In Web Application Server 3.0, the listener still authenticates static pages and CGI scripts. Cartridges, however, are authenticated either by the dispatcher or by themselves. These mechanisms are explained under [“Cartridge Authentication”](#) and [“Dispatcher Security”](#).

- Usernames and passwords used for Basic or Digest Authentication (see “[Security Schemes](#)”), when used to protect WRB cartridges, are now stored in encrypted format.
- Web Application Server 3.0 now supports [Database Authentication](#), enabling database-driven authentication for cartridges.
- In Web Application Server 3.0, the PL/SQL Cartridge was modified to support querying of the username and password from the browser if they did not exist in the DAD (previously, the DCD). Also, a new field, *pkg_protect*, was added so that you can control which PL/SQL procedures the PL/SQL Cartridge can directly execute from a URL. See “[PL/SQL Cartridge Security](#)”.
- In Web Application Server 3.0, cartridges that you write can invoke the [Authentication Server](#). This means they can use any pre-existing dispatcher authorization schemes to authenticate a client. See “[Cartridge Authentication](#)”.
- Because of the preceding, you can now place a firewall between the dispatcher and the WRB or between the WRB and various cartridges, as explained under “[Using the Web Application Server with Firewalls](#)”.

Authentication Server

The Authentication Server component of the Web Application Server encapsulates the authorization to be performed against cartridges. When the dispatcher receives an HTTP request requiring access to a cartridge, it contacts the Authentication Server and waits for its response to determine whether the request can be filled.

If the cartridge itself is handling authorization, it can also invoke the Authentication Server, provided that it specifies the “Existing Methods” authorization. Note that you specify this only for cartridges that you created yourself using the WRB API. The authorization code is part of the cartridge, and therefore existing cartridges already have their authorization schemes defined. None of the cartridges currently bundled with the Web Application Server invokes the Authentication Server. For more information on this, see “[Cartridge Security](#)”.

The Authentication Server itself functions the same whether it is invoked by a dispatcher or a cartridge.

The purpose of the Authentication Server is to enable you to manage security at a high level across all cartridges. As cartridges are added by Oracle, third parties, or you, you have the option of protecting them by simply extending your existing security schemes in highly flexible ways. The potential uses of the Authentication Server go beyond the Web Application Server to encompass Oracle’s entire Network Computer Architecture (NCA).

The remainder of this section is divided into the following sections:

- [Elements of the Authentication Server](#), which explains how the Authentication Server works
- [Modes of the Authentication Server](#), which explains the difference between the ORB and inmemory modes of the Authentication Server

Elements of the Authentication Server

- **Protect strings** - These associate a set of protections with the virtual path that indicates a particular cartridge. The protections consist of groups of schemes and realms (explained below) that are combined with standard logical AND ('&') and OR ('|') operators. There is no more than one protect string for each cartridge. To set the protect string for use by the dispatcher, use the Web Request Broker Configuration form in the Web Application Server Manager.

Although the Authentication Server enforces protect strings, the protect strings themselves are stored by the object that invokes the Authentication Server, which is either the dispatcher or a cartridge. Any invocation of the Authentication Server includes the protect string to be evaluated.

- **Schemes** - A scheme specifies how the authorization is to be performed. The Authentication Server supports the following schemes:

- [Basic Authentication](#)
- [Digest Authentication](#)
- [IP Address Restriction](#)
- [Domain Name Restriction](#)
- [Database Authentication](#)

Schemes are implemented using the mechanism of providers (described below).

- **Realms** - A realm specifies a set of criteria within a scheme that must be met. For example, Basic Authentication is a scheme. This scheme specifies that some username/password combination must be supplied for the request to be authorized. The *specific* username/password combination, for example Scott/Tiger, that must be supplied is determined by the realm. A protect string specifies both the scheme and the realm within it that must be satisfied for the request to be authorized. For example:

```
Authorization: Basic realm="The_Usual"
```

This is a simple protect string that specifies one protection: a realm, located in the Basic Authentication scheme, called *The_Usual*. *The_Usual*, of course, is simply the name, not the content, of the realm. The provider (explained below) implementing Basic Authentication will have a realm called *The_Usual* that specifies the username and password to be used for authorization (let us say Scott/Tiger). The name of a realm must be unique within its scheme. Among other things, encapsulating the username and password in this way supplies you a means to reference the username/password combination liberally without it having to be stored in more than one place.

- **Brokers** - The Authorization Broker is the portion of the Authentication Server that receives the authorization requests from the dispatchers or cartridges, sends back the message specifying whether authorization was successful, and optionally provides a specific failure message if it was not. Each request to be authorized is transmitted with a protect string that specifies one or more realms and their associated schemes. For each scheme to be used (generally, no more than one realm from a given scheme is used), the broker invokes the Authorization Provider associated with that scheme, and lets it determine whether the protection for that scheme is satisfied. When it has received responses from all providers, the broker applies the AND/OR logic specified in the protect string, and sends a message back to the dispatcher or cartridge that invoked it, saying whether authorization is granted.
- **Providers** - Authorization providers are objects called by the broker to implement authorization schemes. For each scheme the broker recognizes, there is one and only one provider. The difference between a scheme and a provider is that a scheme is an abstraction, a type of protection such as IP restriction. A provider is a code module that implements a specific set of realms in a scheme.

Modes of the Authentication Server

The Authentication Server, like most components of the Web Request Broker, can run in either of two modes: *inmemory* or *ORB*. The functionality is the same for either mode; the differences are in performance and opportunities for distribution. Basically, the differences are these:

- In *inmemory* mode, there is a separate instance of the Authentication Server for each dispatcher and for each cartridge that invokes it. This greatly increases the memory footprint of the Authentication Server, but it also speeds performance considerably. In this mode, the Authentication Server must be on the same machines as all modules that invoke it.

- In *ORB* mode, there is but one instance of the Authentication Server for all objects that will call it. This is slightly slower than the in-memory approach, but also requires less memory. Naturally, distributing the system in this way also has performance consequences.

You set the mode of the Authentication Server using the Authorization Server Configuration page of the Web Application Server Manager.

Security Schemes

This section describes the security schemes available for use by the listener or by the Authentication Server. For more information on the ways these are invoked, see “[Authentication Server](#)”, “[Listener Security](#)”, “[Dispatcher Security](#)”, and “[Cartridge Security](#)”.

Basic Authentication

If this is set, the browser prompts the user for a username and password when an attempt is made to access the protected page or application. Note that, in the case of a non-SSL connection, the username and password are sent over unencrypted, and anyone capable of intercepting the HTTP traffic can obtain the username and password. For this reason, Oracle recommends using [SSL Security](#) in conjunction with Basic Authentication. Under the Authentication Server, both the username and password are stored locally in an encrypted format.

Digest Authentication

This protocol is part of HTTP 1.1. It is the same as Basic Authentication, except that it encrypts the password. Currently, only a few browsers support Digest Authentication (Spyglass Mosaic does, but not Netscape Navigator 3.0 or Microsoft Internet Explorer 3.0). If Digest Authentication is specified for a page, and a user with a browser that does not support Digest authentication tries to access that page, the listener automatically reverts to Basic Authentication. For this reason, you should treat Digest as effectively equivalent to Basic for the majority of users and take precautions accordingly. Under the Authentication Server, both the username and password are stored locally in an encrypted format.

Because Digest Authentication is not supported by all browsers and accepts passwords in the clear from a browser that does not support Digest Authentication, Oracle recommends that you use [SSL Security](#) in addition to or instead of Digest Authentication when password encryption is a concern.

IP Address Restriction

The administrator can either allow or deny access to specified IP addresses. IP addresses are organized hierarchically, this mechanism allows entire subtrees to be specifically included or excluded (class A, B, or C), as in the example "+130.*". Note that it is possible to defeat this method through IP spoofing.

Domain Name Restriction

Instead of allowing or restricting access via IP addresses, the administrator can allow or restrict access via domain names. Portions of the domain name can be wildcards, so that restricted strings like *.xyz.com deny access to all clients coming from the xyz.com domain. Domain names are basically a more convenient expression of IP addresses, and, as with IP address restrictions, this mechanism can be defeated through spoofing.

Database Authentication

Note that Database Authentication is only available through the Authentication Server, not the listener. In Database Authentication (also called BASIC_ORACLE authentication), usernames and passwords are stored in an Oracle database instead of an OS file as in the Basic and Digest Authentication schemes. This simplifies the work of the administrator, since only one set of usernames and passwords needs to be managed for both Web and database access. It also enhances security, since the Oracle7 Server provides more granular access control than operating systems do. The administrator configures a *realm*, which consists of either a database connect string (for a remote database) or an Oracle SID (for a local database). In addition, the realm may contain a database role, which allows only those database users that have the privilege to assume that role to be authenticated. For more on roles, see the *Oracle7 Server SQL Reference*. For more on SQL*Net, see “[SQL*Net Security](#)” or *Understanding SQL*Net*.

Note: Database Authentication is different from Database Access Descriptors, which are OS configuration files used by cartridges to establish database connections. For more on DADs, see “[Database Access Descriptors \(DADs\)](#)”.

Listener Security

- [SSL Security](#)
- [Listener Authentication](#), which is how the listener controls access to static HTML pages and to CGI programs
- [Implications of Multiple Listeners](#)
- [The Listener and OS Security](#)

SSL Security

This explains how to impose public key cryptography on the communication between the listener and the browser.

The SSL protocol works at the transport level, which is one level below the application level. This means that it can encrypt messages before they are handled by application-level protocols such as Telnet, FTP, and HTTP. Oracle Web Application Server can encrypt and decrypt messages using the SSL protocol.

URLs that require SSL use “https” in place of “http”. The listener must be configured to accept an SSL connection, and the system administrator must have a certificate for the listener (usually obtained from Verisign or generated by the Oracle Security Server).

In addition to providing for encryption, SSL also provides for authentication. When the browser connects to the server, the server presents its certificate. The browser then either accepts that certificate (if it is included in the “site certificates” in the browser) or prompts the user to decide whether to accept the certificate. The certificate allows the user to be sure that the server is who it is claiming to be. However, the simple fact that a server has a certificate does not mean the user must or should permit access.

In addition to server being authenticated, the server can request that the client be authenticated. The client must have obtained its own certificate and present that certificate to the server when prompted. Note, client authentication is not required as part of the SSL protocol and both the Web Application Server and Netscape listeners allow this to be configured.

SSL encryption and authentication can be used for static HTML pages, CGI programs, and cartridges. If a request for a cartridge uses SSL, then the resulting HTML page generated by the cartridge is encrypted.

See also [Setting Up Your Web Application Server to Use SSL](#).

Listener Authentication

This section explains how the listener defines authorization mechanisms to protect static Web pages and CGI scripts. WRB cartridges are protected by the dispatcher or by themselves.

In addition to or instead of SSL certificate authentication, the listener can be configured for authentication. In this case, the listener prompts the user for a username and password. You can configure this listener using the “Listener Configuration” section of the Web Application Server Manager. *Note that the usernames and passwords are stored in the OS file system in the clear.*

Note: In previous releases of this product, authorization for WRB cartridges was handled by the listener by protecting the virtual paths that specified WRB calls. Now you use dispatcher or cartridge security to protect cartridges. See “[Migration from Previous Versions](#)”, “[Dispatcher Security](#)”, and “[Cartridge Security](#)” for details.

Authorization mechanisms supported by the Web Application Server listener:

- [Basic Authentication](#)
- [Digest Authentication](#)
- [IP Address Restriction](#)
- [Domain Name Restriction](#)

You set up these authorization mechanisms in the “Listener Configuration” section of the Web Application Server Manager.

Implications of Multiple Listeners

The Web Application Server now provides support for multiple listeners on a network, each having equal access to the same set of cartridges. Previously, multiple listeners were on the same machine, and each listener had its own set of cartridges. To the extent that you handle your authorization at the listener level, your Web Application Server installation is only as secure as your least secure listener.

- Any application protected at the dispatcher level is only protected from users entering the listener associated with that dispatcher. All other listeners must have similar security in order for the protection to be meaningful.

- An SSL secured site (see “[SSL Security](#)”) may only be meaningful if other listeners which are part of that installation are similarly secured. If you are aware of this situation, you can use it deliberately. For example, you can use SSL listeners when users will be prompted for database passwords, but not when the passwords will be taken from the DAD. See “[PL/SQL Cartridge Security](#)”.
- Any new listeners added to the system must be immediately secured with the same (or better) security than is available at the other listeners.
- The **wladmin** utility allows you to configure more than one listener on a single port. Make sure only one listener is running on a given port.

The Listener and OS Security

As run by the OS, the listener consists of a group of processes called “*oraweb*” processes.

- The listener process does most of the listener’s work. For any given listener, there is one listener process. In general, this process can be run by any user.
- The companion processes respond to HTTP input on a port and perform domain name resolution. There is one companion process for each port to which a given listener responds. If they are responding to port numbers below 1024, these processes must be run by *root*. Otherwise, they can be run by any user.

One exception to the first point above is the Admin Listener. Since it has to create other listeners and perform other special tasks, the listener process for the Admin Listener must be run by *root* (the Admin Listener is the listener used to administer the Web Application Server itself). Therefore, your normal procedure for starting the Web Application Server is as follows:

```
su wrbusername (UNIX)
owsctl start wrb
su root (UNIX)
owsctl start admin
```

Where *wrbusername* is the OS user that you wish to own the Web Request Broker. In other words, you start the WRB under the user who will own it and then you become root to start the Admin Listener. After this, you invoke your web browser, so that you can do the rest of the administration using the Web Application Server Manager to interface to the Admin Listener you have just started. From there, you create other listeners and set the users that will own the listener processes for them. Their companion processes are all automatically owned by root. For more informa-

tion on how to create listeners using the Web Application Server Manager, see [Web Application Server Manager](#).

- Any listener that runs a CGI program runs it as the user that is running the listener process. Therefore, it is safer not to run such listener processes as root or another highly privileged user. Since CGI programs are executed directly by the OS, it is hard to have the same control over them as you do over WRB cartridges.
- If you have an Oracle7 Server installed, you can do limited administration of it from the Admin Listener. Since the Admin Listener runs as *root*, however, the Oracle7 Server will not allow a login using OS authentication. You must specify a username and password to connect to any Oracle database. For more information on logging in to Oracle, look under “CONNECT” in the current version of the *Oracle7 Server SQL Reference*.
- It is advisable to create a specific user to own the Web Request Broker and the listeners other than the Admin Listener. Although this user may in practice be equivalent to the default user *nobody*, creating a separate user simplifies security management. This way you can, if necessary, add a privilege to the WRB or Web Application Server owner without making it part of the default set of privileges assigned to nobody.

Note: The Admin Listener is password protected, and you may want to password protect many of your applications or pages as well. Although you can achieve some security in this by specifying Digest rather than Basic Authentication (see “[Security Schemes](#)”), it is important to remember that Digest Authentication only works with certain browsers, and you generally cannot control which browser a user will choose. Therefore, Oracle recommends that you also use [SSL Security](#) on connections where passwords will be transmitted.

Dispatcher Security

This section covers the mechanisms the dispatcher can use to restrict access to WRB cartridges.

If the virtual path specified in a URL indicates a WRB cartridge, that request is handled by the dispatcher. The dispatcher distributes requests among cartridge instances (WRBXs), but is itself part of the same OS process as the listener. The dis-

patcher contacts the [Authentication Server](#) to determine whether the request can be authorized.

The dispatcher can use all the same protection schemes as the listener. In addition, the dispatcher can use a scheme called *Database Authentication*, provided that there is an Oracle7 Server installed and accessible to the Web Application Server.

Authorization mechanisms supported by the Web Application Server dispatcher:

- [Basic Authentication](#)
- [Digest Authentication](#)
- [IP Address Restriction](#)
- [Domain Name Restriction](#)
- [Database Authentication](#)

The configuration for dispatcher-based security is stored in the WRB configuration file *wrb.app*.

Cartridge Security

This section explains how developers who write their own cartridges using the WRB API can create cartridge-specific security mechanisms. In addition, cartridges that access the Oracle7 Server can use DADs to authenticate for database access, as described under “[Database Access Descriptors \(DADs\)](#)”.

Cartridges play an important part in security. Cartridges can perform their own authorization, as described under “[Cartridge Authentication](#)” below. In addition, cartridges can invoke one another through Inter-Cartridge Exchange (ICX). In this case, the destination cartridge has to determine whether to accept the request from the source cartridge. This is described under “[ICX Security](#)”.

Cartridge Authentication

Cartridges have a number of event handlers (callbacks), each of which has a specific purpose. These event handlers include:

- **Initialization** - This event handler is called once at cartridge startup.
- **Authorization** - This event handler is called for every request and allows the cartridge to grant or deny access to the request.
- **Execution** - This event handler is only called if the authorization event handler accepts the request. It invokes the code the cartridge uses to fulfil the request.

As a cartridge developer, you control the authorization you wish to use in the code called by the authorization handler. For example, you can keep a local database of usernames and passwords for purposes of authentication. This can be stored either in a relational database, such as the Oracle7 Server, or in OS files, and can either be encrypted or not. If this authorization fails, the Web Application Server returns the correct HTTP error message.

The different kinds of authorization the cartridge has the option of performing in the authorization event handler include BASIC_NEW, DIGEST_NEW, and EXISTING_METHODS.

BASIC_NEW

In this scenario, the cartridge is performing its own authentication against the username and password that the browser sends. This is functionally equivalent to [Basic Authentication](#), except that the cartridge itself, rather than the [Authentication Server](#), maintains the valid set of usernames and passwords and performs the authorization. Whether these are stored in an OS file or a database, whether they are stored in the clear or encrypted, and whether they are stored locally remotely, are all in the hands of the cartridge developer.

Note: The PL/SQL and Java Cartridges that come with the Web Application Server use BASIC_NEW authentication in conjunction with DADs. You can do this for cartridges that you write as well. For details on DADs, see “[Database Access Descriptors \(DADs\)](#)”.

DIGEST_NEW

In this scenario, the cartridge is performing its own authentication against the encrypted username and password that the browser sends (provided that the browser supports Digest Authentication). This is functionally equivalent to [Digest Authentication](#), except that the cartridge itself, rather than the [Authentication Server](#), maintains the valid set of usernames and passwords and performs the authorization. Whether these are stored in an OS file or a database, whether they are stored in the clear or encrypted, and whether they are stored locally or remotely, are all in the hands of the cartridge developer.

EXISTING_METHODS

In this scenario, the cartridge invokes the [Authentication Server](#), just as the dispatcher would. The options under this method are therefore Basic, Digest, IP, Domain, Database (see “[Security Schemes](#)”), and custom-written (see “[Cartridge](#)

[Security](#)”), just as for the dispatcher. The reason for providing this option is so that the application developer can programmatically enforce an Authentication Server-driven authorization scheme. Whether the Authentication Server is called by the dispatcher is determined by the Web Server Administrator, who can change that configuration parameter at any time.

If you specify existing methods, you can optionally place the WRB API authorization call in the code associated with the initialization handler, rather than that associated with the authorization handler. This has the effect of setting up the call to the Authentication Server once for the life of the cartridge, rather than for each request. This is a short-cut for having the cartridge specify the method during the authorization event handler. *Note, however, that authorization specified in the initialization handler is circumvented by any ICX calls that your cartridge receives (see “[ICX Security](#)”).* If you use either BASIC_NEW or DIGEST_NEW, placing the authorization call in the code associated with the initialization handler is not an option.

ICX Security

ICX (Inter-Cartridge Exchange) is a feature of the Web Application Server that allows a cartridge (the *calling cartridge*) to send a request to another cartridge (the *called cartridge*). The calling cartridge may need the results from a called cartridge, and when those results are received, the calling cartridge sends back a combined response to the client. For example, a cartridge that implements a virtual bookstore can use ICX to talk to another cartridge that processes a payment request to debit the user’s account. Since ICX sends a request from one cartridge to another, the calling cartridge must provide the authorization information needed in order to talk to the called cartridge.

- For security mechanisms based on prompting for a username and password (Basic, Digest, BASIC_NEW, DIGEST_NEW, and Database. See “[Security Schemes](#)” and “[Cartridge Authentication](#)”), the cartridge making the call must set the username and password using the ICX calls provided for this purpose in the WRB API. Once this is done, the authorization is handled transparently, according to the principles outlined below.
- For security mechanisms based on restriction, which means only accepting calls from specified IP addresses or domains (see “[Security Schemes](#)”), the IP address or domain name used is that of the machine containing the calling cartridge.

The calling cartridge uses one of the following methods:

- If the called cartridge is in a different ORB system than the calling cartridge, the request is formatted as a standard HTTP call with the appropriate authori-

zation information in the HTTP header. In this case, it is transparent to the called cartridge whether this call comes from another cartridge or directly from a user. Therefore, the authorization performed is identical to what would be performed for the same request coming directly from a user.

- If the two cartridges are in the same ORB system, however, the calling cartridge circumvents this process by directly invoking the code of the called cartridge, beginning with the *authorize* event handler. This greatly enhances performance.
 - Since execution passes directly to the called cartridge, the dispatcher is not being directly invoked. Nonetheless, the ICX mechanism emulates the dispatcher security (see “[Dispatcher Security](#)”) that would have been applied had the called cartridge been invoked by an outside request. It performs this verification before it actually issues the ICX call, so, if this authorization fails, there will be no logging of the action for the called cartridge, only for the calling cartridge.
 - Since execution passes directly to the authorize event handler of the called cartridge, *any protection for that cartridge specified in the initialization handler (see “[Cartridge Authentication](#)”) is circumvented*. Since specifying authorization in the initialization handler is just an optimization, you may want to forego it for cartridges that are expected to receive ICX calls.

Internal Cartridges

You can create *internal cartridges*. These are cartridges that can be invoked only through ICX, not through HTTP requests from outside the ORB. To do this, protect the internal cartridge using either [IP Address Restriction](#) or [Domain Name Restriction](#) to only accept calls coming from the machines that contain the calling cartridges.

- You can only do this if the internal cartridge and all calling cartridges are in the same ORB system.
- A direct call from a browser will have the correct IP address (or domain name), and therefore still will work, if the browser is located on the same machine as one of the calling cartridges.

Database Security

Communication between cartridges and the Oracle7 Server is over SQL*Net. Therefore, security between the cartridge and the Oracle7 Server is maintained by SQL*Net, not by the Web Application Server. How SQL*Net achieves this is

addressed briefly under [SQL*Net Security](#) below. For more information, see *Understanding SQL*Net*.

An HTTP request that requires access to the Oracle7 Server must use a username and password in order to log in. The mechanism to achieve this is described under [Database Access Descriptors \(DADs\)](#) below.

SQL*Net Security

SQL*Net is a messaging layer that operates transparently across most common networking protocols or within the memory of a single machine. It enables the Oracle7 Server to communicate with various other Oracle components, including the Web Application Server, without having to be concerned with underlying network protocols or system architecture. SQL*Net has a set of its own encryption methods that can be set up to protect the message traffic.

Oracle's Advanced Networking Option (ANO) provides an additional layer of security on top of the SQL*Net protocol for secure client to server communication. ANO integrates the Oracle7 Server with a series of enterprise network services, including enterprise directory services, network encryption, single sign-on services, and token authorization services. Installation of these services makes the difference between basic client/server connectivity and enterprise client/server systems that are secure, manageable, configurable, and suitable for mid and high end deployment.

PL/SQL Cartridge Security

The PL/SQL Cartridge is a WRB cartridge provided with the Web Application Server for the purpose of running PL/SQL applications in the Oracle7 Server. The PL/SQL Cartridge provides additional security for its clients that is beyond the security provided by the WRB API. A URL that invokes the PL/SQL Cartridge specifies the PL/SQL Agent to be used for that request, as follows:

```
http://www.xxx.com/<agent name>/owa/<procedure name>.
```

The Agent names a configuration file on the server that sets various parameters relevant to the PL/SQL Cartridge including the following:

- the DAD to be used for this request
- the package protect value for this request

Note: Even though PL/SQL procedures are case-insensitive, the WRB protection mechanism is case-sensitive. To protect all packages and procedures you need to protect the virtual path for the PL/SQL Cartridge, for example: `/owa_dba/plsql`. Notice that this does not specify a particular package or procedure. To specify a single procedure or package, you must use custom authentication, which is described in the section “Custom Authentication” in “Using Web Application Server Cartridges”.

Database Access Descriptors (DADs)

Database Access Descriptors are used to control database access from the PL/SQL and Java cartridges. They can also, but need not, be used by any cartridge you write that access the Oracle7 Server. The discussion assumes the PL/SQL Agent is being used.

The DAD is a configuration section in the *wrb.app* file. The DAD may contain an Oracle username and password, which are used to connect to the Oracle7 Server to fill the request. Note that this method prevents Oracle usernames and passwords from being transmitted through HTTP.

If the DAD does not contain a username and password, or if the attempt to log on using the username and password in the DAD fails, the PL/SQL Cartridge performs [BASIC_NEW](#) authentication. This means it will instruct the Web Application Server to prompt for a username and password to log on to the Oracle7 Server. The difference between this and [Database Authentication](#) is that, in this case, the [Authentication Server](#) is not involved.

Note: In this situation, the username and password are sent from the browser to the listener in the clear. If encryption is important, [SSL Security](#) should be enabled for all listeners invoking the PL/SQL Cartridge.

Stored Procedure Access (Protecting Packages)

As shown above, the PL/SQL Cartridge enables a client to enter a URL that directly specifies a PL/SQL procedure to be executed by the Oracle7 Server. This is desirable, but should be controlled. The only procedures that should be made available in this fashion are those intended to be entry points into your application. These procedures themselves should be able to call other procedures, without exposing the latter procedures to the public.

The PL/SQL Cartridge lets you enforce this by protecting packages. A protected package can only be executed by another stored procedure, not directly by a URL. By default, all the OWA utilities provided with the Web Application Server are protected. Other packages are by default unprotected. To override the default, use the PL/SQL Cartridge configuration section of the Web Application Server Manager to change the `package_protect` parameter. A value of `TRUE` means the package can be executed from a browser.

Using the Web Application Server with Firewalls

A firewall is used to secure the Web Application Server across an Internet/ Intranet junction. Typically, a firewall may be placed:

- **in front of the listener** - In this case, the firewall provides access for HTTP and SSL for browser to listener communication
- **behind the Web Application Server, but before the Oracle7 Server** - In this way, the data is kept secure and a firewall is used that allows secure Oracle SQL*Net protocols to pass through. Currently, the firewalls that permit SQL*Net traffic to pass through are as follows:
 - Trusted Information System's Gauntlet
 - Raptor
- **between the components of the Web Application Server itself** - Since the Web Application Server has a distributed architecture (see "[Security Architecture](#)"), you can place a firewall between any two components that are configured to run over the ORB. It is best to contact your firewall vendor before attempting this, however. The underlying network protocol used by the Web Application Server components to communicate is encrypted UDP.

Setting Up Your Web Application Server to Use SSL

The Web Application Server can accept SSL connections on a particular TCP/IP [port](#). SSL is an emerging standard for encrypted data transmission. To set up your Web Application Server to use SSL, you must do the following:

1. Generate a [certificate](#) request and send the request to VeriSign, Inc.®, a Certifying Authority (CA).
2. Physically secure and prepare your Web Application Server host machine.
3. Install the certificate granted you by the CA.
4. Activate SSL on at least one Web Application Server port.

Generating and Requesting a Certificate Using `genreq` and Verisign

To generate a certificate request, run the interactive utility `genreq` and enter the information for which it prompts you. When the prompt specifies a default value, you can just press Return to enter that value, or enter a different value if you prefer. To run `genreq`:

1. Type `genreq` to start the utility.
2. Type `G` to begin creating a certificate request.
3. When prompted, type a password used in generating the [private key](#). Just choose a random string of characters—you need only remember this string long enough to repeat it in the next step.
4. Retype the password for confirmation.
5. Choose the public exponent you want to use in generating the [key pair](#). The only two recognized exponents are 3 and 65537, commonly called Fermat 4 or F4.
6. Enter the size in bits of the modulus you want to use in generating the key pair. For the version of `genreq` sold in the United States of America, the size may be from 1 to 1024. The default size is 768 bits and the maximum is 1024 bits. A modulus size between these two values is recommended.

For versions of `genreq` sold outside the USA, the maximum (and default) modulus size is 512 bits.

7. Choose one of three methods for generating a random seed to use in generating the key pair:
 - `F` - `genreq` prompts you to enter the full pathname of a file in your local file system. This can be any file that is at least 256 bytes in size, does not contain any secret information, and has contents that cannot easily be guessed.
 - `K` - `genreq` prompts you to enter random keystrokes. `genreq` uses the variation in time between keystrokes to generate the seed. Don't use the keyboard's autorepeat capability, and don't wait longer than two seconds between keystrokes. `genreq` prompts you when you have typed enough keystrokes. You must delete any unused characters typed after this prompt.
 - `B` - `genreq` prompts you to enter both a file name and random keystrokes. This option is recommended.
8. Enter the name of a file in which to store your server's distinguished name. You can choose the default, or enter any filename with a `.der` extension. `gen-`

`req` creates this file in the current directory, though you may later move it to any convenient location.

9. Enter the name of a file in which to store your server's private key. You can choose the default, or enter any filename with a `.der` extension. `genreq` creates this file in the current directory, though you may later move it to any convenient location. Do not specify a location that is accessible via NFS or other network protocol.
10. Enter the name of a file in which to store the certificate request. You can choose the default, or enter any filename with a `.pkc` extension.
11. Enter the requested identification information for your organization:
 - **Common Name** - the fully qualified host name of your organization's Internet point of presence as defined by the Domain Name Service (DNS), for example, `www.oracle.com`.
 - **Email Address** - the email address where the CA can contact you.
 - **Organization** - the official, legal name of your company or organization. Most CAs require you to verify this name by providing official documents, such as a business license.
 - **Organizational Unit** - (optional) the name of the group, division, or other unit of your organization responsible for your Internet presence, or an informal or shortened name for your organization.
 - **Locality** - (optional) the city, principality, or country where your organization is located.
 - **State or Province** - the full name of the state or province where your organization is located. VeriSign does not accept abbreviations.
 - **Country** - the two-character ISO-format abbreviation for the country where your organization is located. The country code for the United States is "US."
 - **Email Address** - the email address where the CA can contact you.
 - The name and version number of the application for which you are getting the certificate (you should accept the default value). When you have entered all the requested information, `genreq` responds with `Thank you`, and processes the data you have entered. When it is finished, it outputs `done` and returns you to the main menu.
12. Type `Q` to quit the program.

To request a certificate, email the request to VeriSign, Inc.® a subsidiary of RSA Data Security, Inc.

The certification process can take from a few days to several weeks. The more organized and complete your paperwork, the better your chances are for quick certification.

Preparing Your Web Application Server Host Machine

For your Web Application Server to be secure as advertised to clients, you must make sure that no unauthorized person has access to your Web Application Server's host machine. Here are some suggestions:

- Place the machine in a locked server room.
- Limit distribution of keys or combinations to the server room to a few trusted individuals.
- Set up a secure area of the machine's file system that can be accessed only by the root user. This is where you will store your private key and your certificate when you receive it. Do not use a location that is accessible via NFS or other network protocol.
- Set a secure root password on the machine, using at least six characters and mixing numbers, legal punctuation marks, and mixed-case letters. Try not to use a character string that is a proper name or a word in any language. Change passwords frequently, and never write a password on paper.
- Strictly limit the programs that are installed and allowed to run on the machine.
- Limit TCP/IP connections to the machine to port 443, the default port for secure connections. Disable all other ports.

To learn how to implement these suggestions on your Web Application Server machine, see your machine's operating system reference manual.

Installing Your Certificate

When you receive your certificate, you must use the Web Application Server Manager to install it.

1. Use your email reader to save the message from the CA containing the certificate to a file with a `.der` extension, such as `cert.der`.
2. Use a text editor to remove the header information before the `BEGIN CERTIFICATE` line and the footer information after the `END CERTIFICATE` line. *Do not*

delete or alter the BEGIN CERTIFICATE and END CERTIFICATE lines themselves.

3. Using your web browser, go to the *Web Application Server Administration Home* page.
4. Follow the link to the *Oracle Web Listener Administration* form.
5. Select a Web Listener process from the list and click its Configure button to go to the *Oracle Web Listener Advanced Configuration* form.
6. Follow the Security: Access Control and Encryption link to go to the *Oracle Web Listener Security* form.
7. Select the Secure Sockets Layer section of the form and follow the instructions.
8. Go back to the *Oracle Web Listener Advanced Configuration* form.
9. Follow the Network Parameters link and go to the Addresses and Ports section.
10. Set the Security pull-down menu to SSL in the entry for at least one port.

Note: For increased security, it is best to activate SSL only on port 443.

11. Go back to the *Oracle Web Listener Administration Form* and repeat the process for all Web Listener processes that you want to make secure.

Migration from Previous Versions

This section provides a guide for migrating your authorization configuration from Oracle WebServer version 2.0 and later to Web Application Server, version 3.0.

- Previously, WRB cartridges were protected just as static files and CGI programs, in the listener. This is no longer the case. Now, cartridges either are protected by the dispatcher or protect themselves. Therefore, you must migrate the virtual path mappings previously used to protect the cartridges to use one or both of these other schemes. For details on these schemes, see “[Dispatcher Security](#)” or “[Cartridge Security](#)”.
- Previously, each listener was associated with its own set of cartridges. Now all the listeners share the same set of cartridges. The security implications of this are discussed under “[Implications of Multiple Listeners](#)”.
- Since the authorization mechanisms supported by the [Authentication Server](#) are a superset of those supported by the listener, you can seamlessly transfer

your existing listener-based protection for cartridges to the dispatcher (see “[Dispatcher Security](#)”). Since there is an instance of the dispatcher associated with each listener, you can easily retain configurations that may have varied between listeners.

- In release 2.0, the Database Connection Descriptor (DCD) had to specify a username and password for database access by the PL/SQL Cartridge or CGI program. In 3.0, the function of the DCD has been given to the DAD (see “[Database Access Descriptors \(DADs\)](#)”), but the username and password are optional. If they are omitted, the user will be prompted for them.
- Previously, any PL/SQL procedure accessible to the PL/SQL Agent could be executed directly from a URL. This is no longer the case. Now you can specify that certain packages are entry points and certain packages are to be called only by other program units, not directly by users. This mechanism is explained under “[Stored Procedure Access \(Protecting Packages\)](#)”.