Oracle9*i* Application Server

Security Guide

Release 2 (9.0.2)

January 2002

Part No. A90146-01

Oracle9*i* Application Server Security Guide, Release 2 (9.0.2)

Part No. A90146-01

Primary Author: Laurel P. Hale

Contributors: Henry Abrecht, Mike Cowan, Janis Greenberg, John Heimann, Pushkar Kapasi, Mark Kennedy, Jeffrey Levinger, Bruce Lowenthal, Rita Moran, Julia Pond, Frank Rovitto, Deborah Steiner

Contents

6 Configuring Oracle9*i*AS Portal Security

Send Us Your Comments

Oracle9*i* Application Server Security Guide, Release 2 (9.0.2)

Part No. A90146-01

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

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

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

- Electronic mail: iasdocs_us@oracle.com
- FAX: (650) 506-7409 Attn: Oracle9*i* Application Server Documentation Manager
- Postal service:
  Oracle Corporation
  Oracle9*i* Application Server Documentation Manager
  500 Oracle Parkway, M/S 4op11
  Redwood Shores, CA 94065
  USA

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

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

Preface

This document presents basic Web security concepts and describes the Oracle9*i* Application Server security framework and how to use it. First, it provides a survey of security issues and requirements that arise when operating private business systems in the public Internet environment. Then it introduces the security features of Oracle9*i* Application Server and provides configuration information for setting up a secure middle tier.

This preface contains these topics:

- Audience
- Organization
- Related Documentation
- Conventions
- Documentation Accessibility

Audience

The *Oracle9i* Application Server Security Guide is intended for security administrators, application developers, database administrators, system operators, and other Oracle users who perform the following tasks:

- Configure middle-tier system security
- Analyze application security requirements
- Implement security technologies
- Administer middle-tier system security

To use this document, you need to have general knowledge of Web server administration, Internet concepts, and networking concepts.

Organization

This document contains:

Chapter 1, "Security Fundamentals in a Web Environment"

This chapter introduces the fundamental concepts of communication and data security in an Internet environment, and outlines the threats against which data and systems must be defended.

Chapter 2, "Oracle9i Application Server Security Architecture and Features"

This chapter describes the Oracle9*i* Application Server security framework, including its architecture. It describes each element and how they work together.

Chapter 3, "Configuring Oracle9iAS Single Sign-On"

This chapter describes the security features of Oracle9*i*AS Single Sign-On and provides basic configuration information for setting up single sign-on in the middle tier. It includes information about enabling other elements of Oracle9*i* Application Server to use Oracle SSO technology.

Chapter 4, "Configuring HTTP Server Security"

This chapter describes the security features of Oracle HTTP Server and provides basic configuration information for setting up a secure HTTP server, including how to configure it for basic authentication and to use Secure Sockets Layer (SSL).

Chapter 5, "Using Oracle Wallet Manager"

This chapter describes how to use Oracle Wallet Manager, a software program for requesting, storing, and managing digital certificates in Oracle wallets.

Chapter 6, "Configuring Oracle9iAS Portal Security"

This chapter describes the security features of Oracle9*i*AS Portal and provides basic configuration information for setting up a secure corporate portal.

Chapter 7, "Configuring JAAS Support"

This chapter describes how to configure Java Authentication and Authorization Services (JAAS) for Java 2 Standard Edition (J2SE) and Java 2 Enterprise Edition (J2EE) environments.

Chapter 8, "Configuring Security for Oracle9iAS Web Cache"

This chapter describes the security features of Oracle9*i*AS Web Cache and provides basic security configuration information.

Chapter 9, "Configuring Secure Database Access by Oracle9i Application Server"

This chapter outlines the steps for configuring secure access to an Oracle database from Oracle9*i* Application Server.

Glossary

This glossary contains terms that are pertinent to Web security and Oracle environments.

Related Documentation

For more information, see these Oracle resources.

Descriptions of documents have been added to some listings to guide you to where specific security information can be found. Where document titles are self-explanatory, no description is provided.

Oracle9*i* Application Server Documentation Library contains the following documents unless otherwise specified:

- *Oracle9i Application Server Quick Tour*

  A brief graphical overview of the application server.

- *Oracle9i Application Server Concepts Guide*

  An overview of the application server features.

- *Oracle Internet Directory Administrator's Guide*

  Detailed description of Oracle Internet Directory, including Delegated Administration Service and Directory Integration Service, and how to use them.

- *Oracle Internet Directory Application Developer's Guide*

  Detailed description of how to enable applications to access Oracle Internet Directory by using the C API and the PL/SQL API.

- "Oracle Internet Directory Administration and Delegation Model in Oracle9*i* Application Server, Release 2"

  White paper that provides a complete description of the starter Oracle Context that is set up in Oracle Internet Directory when you install Oracle9*i*AS Infrastructure. It is available on Oracle Technology Network (OTN) at

  `http://otn.oracle.com/docs/index.htm`

  > **See Also:** For information about what OTN is and how to use it, please refer to the end of this section.

- *Oracle9iAS Single Sign-On Administrator's Guide*

  Detailed description of how to enable single sign-on for Oracle9*i* Application Server.

- *Oracle9iAS Single Sign-On Application Developer's Guide*

  Detailed description of how to enable applications to use Oracle9*i*AS Single Sign-On.

- *Oracle HTTP Server Administration Guide*
- *Oracle9iAS Portal Configuration Guide*
- *Oracle9iAS Containers for J2EE Services Guide*

  Detailed description of all J2EE services that are supported by Oracle9*i* Application Server, including JAAS support.
- *Oracle9iAS JAAS Provider API Reference*
- *Oracle9iAS Containers for J2EE User's Guide*
- *Oracle9iAS Web Cache Administration and Deployment Guide*
- *Oracle9i Application Server mod_plsql User's Guide*

  Detailed description of how to configure and use Oracle HTTP Server plug-in, mod_plsql, which enables communication between the middle tier and an Oracle database.

Oracle9*i* Application Server Platform-specific Documentation contains the following documents:

- *Oracle9i Application Server Installation Guide*

  Detailed description of what you need to install to get the security functionality that you require.
- *Oracle9i Application Server Release Notes*
- *Oracle9i Application Server: Migrating from Oracle9i Application Server 1.x*

  Detailed description of what you need to do if you are migrating from a previous version of Oracle9*i* Application Server, such as migrating digital certificates.
- *Oracle9i Application Server Performance Guide*
- *Oracle9i Application Server Best Practices*

  Detailed description of Oracle9*i* Application Server best practices, including security best practices.

Oracle Database Documentation Library contains the following documents:

- *Oracle Advanced Security Administrator's Guide*

  Detailed description of how to configure and use Oracle Advanced Security, the Oracle database option that provides encryption, integrity protection, and advanced authentication to Oracle database clients and servers.

- *Oracle9i Database Administrator's Guide*

  Description of the Oracle9*i* Database Server feature, proxy authentication, which allows Oracle9*i* Application Server to establish an authenticated session with the database.

- *Oracle9i Application Developer's Guide - Fundamentals*

  Detailed description of how to enable Oracle9*i* Application Server to use database proxy authentication.

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

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

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

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

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

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

`http://otn.oracle.com/admin/account/membership.html`

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

`http://otn.oracle.com/docs/index.htm`

To access the database documentation search engine directly, please visit

`http://tahiti.oracle.com`

Conventions

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

- Conventions in Text
- Conventions in Code Examples
- Conventions for Microsoft Windows Operating Systems

Conventions in Text

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

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.

Convention	Meaning	Example
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true.` Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

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

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

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

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

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

Convention	Meaning	Example

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

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

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

`Italics`	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`*
		`DB_NAME =` *`database_name`*

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

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Microsoft Windows Operating Systems

The following table describes conventions for Microsoft Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Database Configuration Assistant, choose Start > Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\\\, then Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32` is the same as `C:\WINNT\SYSTEM32`
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`
	The backslash (\\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\"` `C:\>imp SYSTEM/`*`password`* `FROMUSER=scott TABLES=(emp, dept)`
`HOME_NAME`	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`

Convention	Meaning	Example
`ORACLE_HOME` and *`ORACLE_BASE`*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *`ORACLE_HOME`* directory that by default used one of the following names: ■ `C:\orant` for Windows NT ■ `C:\orawin95` for Windows 95 ■ `C:\orawin98` for Windows 98 This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *`ORACLE_HOME`* directory. There is a top level directory called *`ORACLE_BASE`* that by default is `C:\oracle`. If you install Oracle9*i* release 1 (9.0.1) on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\ora90`. The Oracle home directory is located directly under *`ORACLE_BASE`*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle9i Database Getting Starting for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\rdbms\admin` directory.

Documentation Accessibility

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

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

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

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

1
Security Fundamentals in a Web Environment

This chapter presents an overview of security requirements in a Web environment. For security professionals, the issues and solutions will be familiar. For readers whose experience is less extensive, this chapter can provide a framework for understanding the problems that must be confronted and the methods currently in use.

This chapter contains the following topics:

Promises and Problems of the Internet

This section provides an overview of Internet security issues.

- Introduction to Security Issues
- Security Vulnerabilities
- Attributes Needed for Successful Security
- Trade-offs Between Security and Other Business Needs

Introduction to Security Issues

Security for computer systems means protection for data, processes, and transmissions against unauthorized, accidental, malicious, or otherwise inappropriate access, use, corruption, or misrepresentation. This chapter refers to all such undesired effects as "inappropriate."

In today’s Internet-connected world, security concerns have multiplied beyond every simple solution because the number and complexity of such inappropriate possibilities have multiplied with every new Internet user and business connection.

While databases, applications, operating systems, and communication methods have grown ever more complicated and interwoven, so too has the number and sophistication of attackers. Some are teenagers; some are competitors; some are foreign governments; and some are criminals. Whatever the source of the inappropriate action, the ideal scene is for it to fail. The design goals of security systems and methods are that, to the greatest extent possible with the resources available, the following objectives be achieved:

- Inappropriate system access is denied, so that someone requesting access who cannot prove legitimacy cannot act as a user.
- If inappropriate system access occurs, inappropriate access to databases and system resources is denied, so that even validated users can access only the resources for which they are known to have permission.
- If inappropriate access is gained to databases, changes to critical or sensitive (such as national security and payroll) data can be made only by specific users, by users with permissions specific to the sensitive data, or by using additional knowledge not contained in the database and not derivable from the same validation that granted access, that is, the original password.
- If inappropriate access to data occurs, the following protections are active:
  - That data is unusable without special knowledge.

    - Operations on the data are constrained by links to other data regulated by rules or relationships within the database.
    - Inappropriate communication of the data is difficult or impossible.
    - Inappropriate communication of the data will not be believed, used, or relied upon.
- If inappropriate access to system resources occurs, limitations are in place that restrict their inappropriate use, such as erasing files not owned, communicating over restricted lines, or avoiding firewall-type barriers.

These objectives represent layers of defense against inappropriate access and use. The intention is to make each layer as impenetrable and incorruptible as possible, within constraints of time, money, staff, and "user convenience." But this layered defense also requires each layer to contain defenses against inappropriate actions by users who, though validated for entry to this level, might accidentally or intentionally act inappropriately.

This chapter presents today's concepts and methods that address the vulnerabilities inherent in e-business and Internet interactions, using the terminology defined in Tables 1–1, 1–2, and 1–3. Reading these definitions will help you understand the approaches taken by security experts and will clarify the connections among the security measures discussed in this book.

Table 1–1 Intrusion Terminology

Intrusion Terms	Meaning
Unauthorized	Access achieved or actions taken that were not predicted or explicitly authorized or allowed by the responsible security authority, executive, or administrator
Accidental	Access achieved or actions taken that were not intended by the user to occur or to cause the actual consequences
Malicious	Access achieved or actions taken that were done intentionally to circumvent, subvert, or damage security protections, sometimes with the additional intentions either to harm the systems, data, or other users, or to gain benefits that were not paid for or were disallowed by rules, conventions, or laws
Inappropriate	Unauthorized, accidental, malicious, or other actions with the purpose or effect of obtaining information or privileges to which one is not entitled by law, custom, or administrative choice

Table 1–2 Protection Terminology

Protection Terms	Meaning
Authentication	The process or result of validating an entity's identity, typically based on what the entity is, what it knows, or what it has. For example, someone might be authenticated by proving she is a voter, or by knowing her social security number, or by having a driver's license.
Authorization	The process or result of establishing what objects and actions an entity can access or perform, such as view, add, change, delete, etc. One authenticated person might be authorized to view but not change his profile in the company records; another might be authorized to change a title but not a salary; and so forth.
Accountability	A verifiable, undeniable association between actions and the person or entity who performed those actions.
Certificates	Digital identity records, issued by trusted third parties, that identify users and machines.
Electronic Signature	Also known as digital signature. An encrypted compression of a message that can provide reliable proof that the person signing the electronic transmission really is that person, since no one else can create the unique digital signature supplied.
Encryption	Transforming plain text into a less readable or understandable form. Using a mathematical process to do so. If a specific word, phrase, or number is central to the encryption, it is called the key.
Decryption	Transforming the less readable form back into the original plain text. Decrypting means undoing the encrypting. If a key was used, decrypting requires knowing and using the key.
Integrity	The original form, relationships, and rules of the data. If these are changed without permission or notice, integrity is said to be lost or corrupted.
Permissions	Attributes of an entity that allow it to view data or take actions that would otherwise be restricted or prohibited.

Table 1–2 Protection Terminology

Protection Terms	Meaning
PKI, public and private key	Public Key Infrastructure: a system of digital certificates issued by Certificate Authorities (CA) that verify and authenticate that the parties to an Internet transaction are who they say they are. Each party has a public key available to anyone, and a private key known only to itself. These are used for encryption and decryption of messages or transactions. A PKI is also called a trust hierarchy, since a CA with a certificate from a known-good CA can also issue certificates. See Certificates and Certificate Authorities on page 1-20.
Privileges	Permissions. See above.
Non-repudiation	Making it impossible to deny that a message or transaction you sent was sent by you.

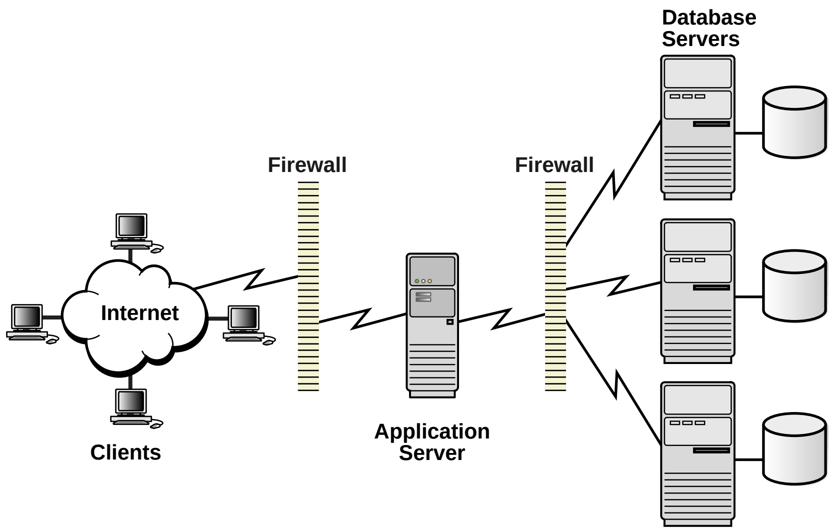
Table 1–3 **Vulnerability** ***Terminology***

Vulnerability Terms	Meaning
Rerouting	Causing messages or transactions to arrive at a destination different from the one intended by the original source of the message or transaction. The intent can be to disrupt the original processing or to falsify the results and present them as if created by the original destination.
Observing	Viewing messages, transactions, or packets by an entity not a party to the desired communication. The intent of observing can be to decode protected data or to view, expose, or use private communications.
Repudiation	Denying that a message or transaction you sent was sent by you.
Corruption	Unauthorized or hidden changes to the content or relationships in a database or a communication.

Security Vulnerabilities

As noted in the Introduction, the exponential growth of business and personal connections over the Internet has put valuable and sensitive data at greater risk than ever before. Figure 1–1 illustrates the complex computing environment that your system security plan must encompass.

Figure 1–1 Scope of System Security Needs



You must protect databases and the servers on which they reside; you must administer and protect the rights of internal database users; and you must guarantee the confidentiality of e-business customers and their data as they access your database.

The Internet enables businesses to use information more effectively by allowing customers, suppliers, employees, and partners to get access to the business information they need, when they need it. The greatest promise of e-business is more timely information accessible to more people, at reduced cost of information access.

Changed Processes

These benefits are challenged by the security vulnerabilities associated with replacing trusted and accountable human processes with easy access over the

Internet. "Cutting out the middleman" too often cuts out the information security that the middleman provides. Many brick-and-mortar business processes typically performed by employees, such as typing in an order received by telephone, mail, or fax, are now done directly by outsiders using an Internet connection. While employees are not invariably reliable, at least they are known, and their access to sensitive data is limited by their job function. Physical and procedural controls can be more readily enforced, and there is disciplinary or legal recourse against employees who pass sensitive information outside the company contrary to policy. The threat of punishment thus helps prevent unauthorized access.

Higher Volumes

But in the Internet-enabled world, users now include persons outside the traditional corporate boundary, such as prospects, customers, suppliers, partners, and ex-employees. The potential user community expands from a small group of known, vetted users accessing data on an intranet to thousands of users accessing data over the Internet.

All these users can have direct and immediate online access to business information. Only some of it pertains to each legitimate user; the rest needs protection even during legitimate access. And all of it needs protection from illegitimate access.

More Valuable Data

In addition, the data available for access has changed. Online data has grown more diverse, more timely, more integrated, and more valuable. It is more tempting than ever before. The reasons arise from the efficiencies offered by Internet-enabled business practices. A great variety of costs can be reduced or eliminated while reaching ever more prospects and serving ever more customers. Inventories can be reduced by streamlined operations that give suppliers direct access to consolidated order information and allow just-in-time purchasing. Online competitive bidding can help companies pay lower costs and offer consumers lower prices. Costly errors and delays from manual data handling can be reduced or eliminated by enabling other businesses and consumers to submit and receive business information directly through the Internet.

These Internet processes can often replace even electronic data interchange mechanisms, which are typically proprietary and difficult to integrate with multiple companies' internal data infrastructures.

Linking or consolidating formerly compartmentalized departmental databases allows users to obtain better information, and to get more benefit from it. The integration of formerly physically separate and incompatible databases and applications — often called silos or islands of information — enables faster and better use of sales, manufacturing, distribution, and financial information.

But the better you make the timeliness, accuracy, and scope of data available to legitimate users, the greater its value to intruders as well. As the rewards rise for unauthorized access, the potential also rises for damage to the image and effectiveness of the corporation whose confidentiality can be breached and whose data can be corrupted or misused.

Attributes Needed for Successful Security

Protecting against such misuse is made more complex by the diversity and sheer size of the user communities that can access business systems over the Internet. Business and security systems designed to cope with this level of risk and complexity need to be

- Scalable, to handle far more users and transactions than non-Internet systems, that is, millions rather than thousands,
- Manageable, to automate, reliably and securely, the administrative tasks such as assigning each user an account and password, and handling all associated information the user may supply or want organized, and
- Interoperable, to communicate or even integrate with the proprietary systems of customers, suppliers, partners, and others, enabling outsourcing to acquire supplies and collaboration to provide services.

These requirements demand designs based on widely-accepted standards such as Java, C, and XML. Only then can security mechanisms deployed in e-business systems have the flexibility and interoperability to work easily with multiple systems, thin clients, and multitier architectures.

Hosted Systems and Exchanges

Secure hosting and data exchange can enable economical, secure partitioning of data access by customer or by user, while supporting secure data sharing among communities of interest. Oracle9*i* Application Server makes this possible through support for a public key infrastructure and enterprise user security.

The principal security challenge of hosting is keeping data from different hosted user communities separate. Providing separate systems for each hosted community has the disadvantage of requiring separate installation, configuration, and management for each hosted user community. This solution provides little in the way of economies of scale to a hosting company.

Using Oracle 9*i*AS provides several factors that can greatly reduce costs to hosting service providers. These factors include mechanisms that allow multiple user communities to share a single hardware and software instance, that securely separate data for different user communities, and that provide a single administrative interface to service all the hosted communities.

Similar considerations support the requirements that exchanges have for both data separation and data sharing. For example, an exchange may ensure that a supplier’s bid remains unviewable by other suppliers, yet allow all bids to be evaluated by the entity requesting the bid. Furthermore, exchanges may also support "communities of interest" in which groups of organizations can share data selectively, or work together to provide such things as joint bids.

Trade-offs Between Security and Other Business Needs

No system can be 100 percent secure and still allow user access: there is a trade-off between security and ease of access. A similar trade-off must also be expected in terms of cost and performance.

As you apply security mechanisms to protect data, the cost to break those defenses increases. No single solution can provide total security for a system. If you identify eight different ways to break the security of a system, you can begin to improve security by making a particular way more expensive to break. You can then move on to making the next way more expensive, such as incorporating 128-bit cryptography, which is extremely difficult for hackers to break. Faced with this and other obstacles, malefactors might try to bribe the CEO, rather than try to break in and decrypt the data. Here is a rule of thumb: when the cost to break security is greater than the value of the data protected, then you can quit making the system more secure.

Furthermore, trying to protect *all* the data with every possible defense might degrade performance. For this reason, you must decide just what data needs to be protected. You may want to apply security protection to certain classes of data, and not to others. Regional sales data, for example, may require protection, while promotional photographs may not.

Security Needs in an Internet Environment

This section outlines the security needs of systems within a Web environment using the following headings:

- Confidentiality
- Authentication
- Authorization
- Non-Repudiation
- Network Attacks
- Fault Containment
- Complex User Management Requirements

Confidentiality

Confidentiality refers to not revealing or exposing critical or sensitive information. Data must be stored and transmitted securely, so that information such as credit card numbers cannot be stolen.

Over the Internet and in Wide Area Network (WAN) environments, both public carriers and private network owners often route portions of their network through insecure land lines, extremely vulnerable microwave and satellite links, or a number of servers. This situation leaves valuable data open to view by any interested party. However, communications known to be sensitive, such as credit card numbers, are routinely encrypted, so that even if observed, they cannot be read or used.

In Local Area Network (LAN) environments within a building or campus, insiders with access to the physical wiring can potentially view data not intended for them. Network sniffers can easily be installed to eavesdrop on network traffic. Packet sniffers can be designed to find and steal user names and passwords. Frequent password changes can lessen the risk of misuse, since the stolen data would only be usable until the next change.

Authentication

Authentication ensures that users are who they claim to be. Some authentication methods require the user to be known in advance, by name and password, but other methods dispense with this requirement by using unforgeable certificates.

Authentication can be applied in several ways at various points of vulnerability to guard against unauthorized access and actions.

The idea remains the same even though authentication mechanisms vary for different contexts. In database authentication, the database performs both identification and authentication of users. In external authentication, the operating system or network service performs the authentication. When database user identities are verified by SSL (Secure Sockets Layer), they are called global users, and their access to the database through global roles is authenticated by means of an enterprise directory. When users are allowed to connect through a proxy server, the verification is called multitier authentication and authorization.

Requiring passwords at several points can act as a layered defense against unauthorized access and actions in that a stolen password at one level would not unlock all lower level services. Some companies safeguarding sensitive or valuable data, such as credit card firms, require several items of identification, such as social security number, mother's maiden name, and mailing zipcode. These items are usually not used as multiple passwords but rather as a combined authentication mechanism when passwords have not yet been established. They are advantageous in being specific to the individual while remaining generally not easy to access or to guess, criteria often suggested for selecting passwords. On the other hand, passwords generally should be selectable and easy to change, advantages that these items lack.

Password-Related Threats

However, there are problems inherent in requiring users to have multiple passwords. In large systems, users must remember several passwords for the different applications and services that they use. For example, a developer can have access to a development application on a workstation, a PC for sending e-mail, and several computers or intranet sites for testing, reporting bugs, and managing configurations. Security vulnerabilities arise from the typical user responses to the problem of managing multiple passwords:

- Users often select easily guessed passwords—such as a name, fictional character, or a word found in a dictionary. All of these passwords are vulnerable to attacks that simply try every word in a list or dictionary of commonly-used passwords, such as "guest," "welcome," or "admin".
- Users also often choose to standardize passwords, using the same one on all machines or Web sites, with the potential that a compromised password allows an impostor to use any of them. Similar risks apply to those who use passwords with only slight variations that are easily derived from known passwords.

- Users with complex passwords may write them down where an attacker can easily find them, or they may just forget them—requiring costly administration and support efforts.

All of these strategies compromise password secrecy and service availability. From the user's point of view, remembering multiple passwords is a hassle. From an administrator's viewpoint, maintaining multiple user accounts and passwords is complex, time-consuming, and expensive. Password queries from legitimate users account for a high percentage of help-desk time. And user reactions to the complexity often compromise the intended security benefits.

Authorization

Authorization guards against misuse of systems, applications, or data after access has already been granted; it controls what objects and actions can be used. Authorization generally refers to the process that determines what a user can access or maintains a record thereof. The enforcement of that authorization is called access control, which can require an additional password or validate a request for resources against lists of approved users or permissible activities. For example, a directory that lists your privileges performs an authorization function, whereas database software using that information to limit which data you can see is doing access control.

Unauthorized Access to Data

As an example, the data and services provided by an Oracle database can be protected by such authorization actions. Using Oracle9*i* Application Server can mediate access to services provided by the back-end database. For each transaction, Oracle9*i* Application Server reports a user identity to the database. At that point, the database's native access controls take over, providing resource restrictions based on the identity of the requestor as established by the authentication function. The user's database privileges determine the specific data (that is, the tables, columns, and rows) that are accessible to him.

Intrusions

Authorization is also a defense against hackers who may try to corrupt your Web site. They also try to redirect users to a different site, fooling a client or server into believing that the site is something it is not.

To prevent corruption, you can control access to the administrative functions that govern the content of the site. To help protect against stolen Web connections, you can employ user authorization and encryption.

Non-Repudiation

The intent of non-repudiation is to preserve accountability and prevent misrepresentation. Non-repudiation means that when someone actually sends a message, the sender cannot later disclaim responsibility for sending it.

To ensure against false claims, there must be a digital "signature," usable only by the true sender, that any recipient can verify. A digital signature also solves the parallel problem of someone else sending a message that falsely claims to be from a third party.

How can such a signature be created, and how is it to be protected? If hackers were to steal someone's private key (the person's digital signing capability), that person could be held responsible for any actions the hacker performed using it.

> **See Also:** "Security Considerations in an Internet Environment" on page 1-17 where public and private keys, which are used for encryption and key distribution as well as digital signatures, are discussed.

Network Attacks

This section describes various types of network attack:

- Data Corruption
- Loss or Display of Confidential Information
- Denial of Service

Data Corruption

Distributed environments bring with them the possibility that a malicious third party can perpetrate a computer crime by tampering with data. The damage can be done to messages as they move between sites on the network or, more seriously, to the sites themselves.

In a data modification attack, an unauthorized party on the network intercepts data in transit and changes parts of that data before retransmitting it. An example of this is changing the dollar amount of a banking transaction from $100 to $10,000.

In a replay attack, an entire set of valid data is interjected again onto the network. An example would be to repeat, one thousand times or even once, an initially-valid $100 bank account transfer transaction.

This type of injury to messages is relatively rare, given the usual protections. The more dangerous attack puts in undetected changes to the site itself, either to the data in its presentation or to an underlying database. There is high potential for damage to subsequent visitors or users, and certainly to the responsible company.

Levels of authentication and authorization are your primary defenses in preventing a hacker’s access and use of administrative or database functions to corrupt, falsify, or otherwise misuse site data.

Auditing mechanisms can ensure that data tampering is detected. Non-repudiation mechanisms can help to identify perpetrators.

Loss or Display of Confidential Information

Data in transit must not be modified or viewed, and database data must not be accessible for unauthorized copying or sharing. No unauthorized party should be able to intercept, display, or otherwise misuse confidential information while it is being transmitted over the network or available online for legitimate users.

Denial of Service

Data and Web security also involve the accessibility of information to authorized users, as needed. While system security by itself does not ensure availability, availability may not be possible without security.

Most denial-of-service attacks prevent legitimate users from getting serviced by overwhelming a site with an extremely high volume of repetitive requests. The results can include much slower service for legitimate users or actual site shutdown due to resource overload. Security of the attacked site has little to do with such an attack.

On the other hand, many such attacks begin by exploiting security flaws in other systems, which are not targets of denial-of-service. The intent is to acquire rights and resources that can later be used to generate some of those repetitive phony requests that do build a denial-of-service attack.

In addition to protecting the site itself, appropriate security measures prevent exposing any vulnerabilities that could be exploited by a malicious intruder to mount an attack elsewhere. Individual user profiles that limit the system resources available to each user can help. Examples include limiting usable disk space, the allowable number of concurrent sessions, the permissible CPU processing time, and the amount of logical I/O available to the user.

Fault Containment

If there is a security breach, how do you limit the damage it can cause?

Among the best ways to lessen security risk on the Internet is to provide multiple layers of security mechanisms. Each layer's independent security measures prevent a single security failure from compromising all critical information. This concept is referred to as deep data protection. Deep data protection ensures well-formed, comprehensive security from client to application server to data server, as well as throughout the layers of an application.

Oracle9*i* Application Server provides fault containment through access control, data encryption, and extensive auditing. Access to the database is limited by controls intended to permit only those with established identities to gain entry. Nevertheless, if an unauthorized entity does gain access, the fact that the data is encrypted makes it difficult or impossible for that illegitimate entity to view or use it. Furthermore, Oracle9*i* Application Server's auditing can quickly reveal this breach and provide valuable information in tracking that entity.

As an example at a different level, Oracle9*i* Application Server can ensure that applications do not use root privileges and that cross-site scripting is disallowed. In other words, even a person using root privilege would not be able to insert into a normal message a Java script to send the entire machine configuration to a third party.

Complex User Management Requirements

Security mechanisms must remain effective and easy to administer even when the number of transactions and the size of the databases become huge. Yet with corporate mergers and acquisitions, or even simple unexpected Web site success, the number of users can grow from 6,000 to six million within a single month. A company's system must be able to handle these enormous numbers of simultaneous users without compromising the confidentiality and integrity of its data and transactions with each one. Dissatisfied customers or prospects may turn to competitors (or lawyers).

In such large-scale environments, the burden of managing user accounts and passwords can make a system vulnerable to error and attack. To have reliable security, you need to know who the user really is, across all tiers.

Oracle9*i* Application Server provides a number of security features that support development of Internet-scale applications. These features include proxy authentication, support for Internet and relevant public key infrastructure (PKI) standards, and enterprise user security features such as directory-based privilege management.

Multitier Systems

The user management problem becomes particularly complex in multitier systems. Here, as in most packaged applications, the typical security model is that of One Big Application User. The user connects to the application, and the application (or application server) logs on and provides complete access for everyone, with unlimited privileges and no auditing. This model places your data at risk — especially on the Internet, where your Web server or application server depends on the security imposed by a firewall. Firewalls are commonly subject to continual attack from outside, and a single breach can spawn a multitude of problems.

Scaling the Security Administration of Multiple Systems

Security mechanisms that can handle hundreds of users may not be adequate to handle enormous communities of users. Security administration must thus be scalable. The administration of hundreds of thousands of users is difficult enough on a single system. When security must be administered on multiple such systems, there must be some way to divide and conquer the complexity. Only an intelligent combination of sharing, automated self-service, and delegation can make it manageable.

Creating and maintaining separate databases for multiple application subscribers is not a cost-efficient model for an application service provider. While technically possible, the separate database model would itself quickly become unmanageable. To be successful, a single application installation should be able to host multiple companies—and be administered centrally.

If user identities and privileges can be maintained securely in a single, central repository, other systems and applications can rely on that as a secure starting point. Downstream, other systems and applications can require additional security validations appropriate to their functionality or content. But the burden is then split among collaborating entities, each with a narrower focus more closely related to its specific service.

Thus challenges of scale in administration — particularly for security — can be met through a combination of central management cooperating with multiple applications and databases. Applying appropriate layers of security using a directory based on industry standards can reduce system management costs and increase business efficiency.

Security Considerations in an Internet Environment

This section presents some important system security considerations applicable to a multitier network environment.

- Considerations for Use of Public Key Infrastructure (PKI)
- Authentication Considerations
- Authorization Considerations
- Encryption Considerations
- Data Integrity Considerations

Considerations for Use of Public Key Infrastructure (PKI)

As described in earlier sections, effective Internet security requires secure information exchange mechanisms that are scalable and that support the security of distributed systems. Public Key Infrastructure (PKI) is a technology that meets these requirements with minimal inconvenience.

Oracle9*i* Application Server can use elements of PKI to provide a secure, resilient environment for deploying electronic commerce. This reliable environment supports building systems to handle virtually any type of electronic interaction, from corporate intranets to e-business applications designed for deployment on the Internet.

Strong system security starts with the physical security of systems and the trustworthiness of personnel. With these in place, PKI enhances secure electronic commerce and Internet communications by supporting the following processes:

Authentication	Verifying the identity of users and machines becomes crucial when an organization opens its doors to the Internet. Strong authentication mechanisms, of which PKI is one, verify identities without allowing transmission or storage of reusable passwords. They ensure that persons and machines are the entities they claim to be. This is typically done by a trusted third-party authentication or certification service using conventional cryptography. Proper use of PKI makes impersonation virtually impossible and supports mechanisms enabling systems and applications to trust each other's connections and transmissions.

Encryption	Encryption and integrity algorithms are used to secure communications and ensure the privacy of data sent from one computer to another. They ensure that data remains confidential, that it cannot be modified, and that lost packets can be detected.
Non-repudiation	Non-repudiation means that senders of digitally signed transactions or email cannot claim they did not do so. Digital signatures using PKI can provide reliable proof that the person signing the electronic transmission really is that person, since no one else can create their unique digital signature. This fact also prevents impersonation, because the impostor cannot create that person's digital signature. A PKI digital signature proves that a specific user performed certain operations.

For public-key cryptography, entities that want to communicate in a secure manner must possess certain security credentials. This collection of security credentials is stored in a wallet. Security credentials consist of:

Public and private keys	This form of cryptography uses a secret private key and a mathematically-related public key. Only the public key can be used to encrypt information, and only the corresponding private key can be used to decrypt that information. Only the owner of the key pair knows the private key; the public key can be distributed widely and remains associated with its owner. A message encrypted with the public key can only be decrypted by the owner who knows the associated private key. Such keys are also used in digital signatures to prevent Internet impersonation and repudiation of valid messages.
Digital certificates	Certificates are digital identities, issued by trusted third parties, that identify users and machines. Certificates are issued when that third party receives trusted information proving to its satisfaction the validity of those identities. The certificates can then be securely stored in wallets or in directories and used to prove the claimed identity to anyone on the Internet who trusts that third party.
Certificate Authority (CA)	A CA is a third party that acts as a trusted, independent provider of digital certificates.

Use of a cryptographic key pair to set up a secure, encrypted channel ensures the privacy of a message and can validate the authenticity of the sender of the message. Wide distribution of the public key on a server, or in a central directory, does not jeopardize security because the private key is never shared. The public key for an entity is published by a certificate authority in a user certificate. Entities that want to send secure information can encrypt the information with the recipient entity's public key. An entity that receives a communication encrypted by this method can use its own private key to decrypt the message. (In some cases, the sender might need to reassure the recipient regarding who sent the message. Encrypting the coded message again using its own public key would do the trick. The recipient could decrypt the doubly-encoded message using his private key, and then decrypt the resulting coded message using the sender's public key. If the original message was not encoded using both public keys, the result of decrypting will be unreadable.)

Authentication Considerations

Without effective authentication, authorization policies have little value. Authentication mechanisms rely on the user supplying something uniquely associated with her: something she is, something she knows, or something she has.

This section describes the following aspects of authentication services:

- Passwords
- Certificates and Certificate Authorities
- Secure Sockets Layer (SSL) Authentication and X.509v3 Digital Certificates
- Storing Secure Credentials in an LDAP-Compliant Directory
- Single Sign-on

Passwords

Passwords are one of the basic forms of authentication. A user must provide the correct password when establishing a connection to prevent unauthorized access to systems, applications, or data. Security systems that are dependent on passwords require that passwords be kept secret at all times. However, passwords are vulnerable to theft and misuse.

A number of steps can strengthen the basic password feature and provide greater control over Web site, application, or database security. For example, password management policy can be controlled by administrators and security officers through user profiles. The administrator can establish standards for password complexity, and deny reuse of passwords. Passwords can be timed out, expiring after a certain amount of time; they should not be stored, or sent over the network, in unprotected form.

Certificates and Certificate Authorities

Having a trusted authority available to authenticate all members of the network (clients to servers, servers to servers, users to both clients and servers) is an effective way to address the threat of nodes on a network falsifying their identities. In the PKI model, this method involves certificates and certificate authorities.

A certificate authority (CA) is a trusted third party able to certify that other entities—users, databases, administrators, clients, servers—are who they say they are. A certificate authority might be an external company that offers certificate services, or an internal organization such as a corporate MIS department. The certificate authority has its own certificate and public key, which it publishes, as well as a private key, which is securely maintained. When certifying a user, the certificate authority verifies the user's identity and grants a certificate, signing it with the CA's private key. Servers and clients use the CA's root certificate to verify signatures that the certificate authority has made. The primary services of a CA are

- Issuing certificates,
- Revoking certificates,
- Validating that a human or other entity is represented by the certificate, and
- Providing liability to damages caused by fraudulent use.

A certificate is like an electronic passport that proves the identity of a user or device seeking to access the network. Used with security policies and other infrastructure, the certificate ensures that the entity's information is correct and that the public key it offers as its "passport number" actually belongs to that entity. The primary contents of a certificate are

- The certificate user's name,
- The user's public key, and
- Information about the rights and uses associated with the certificate.

Other validating information in a certificate includes the name of the CA that issued the certificate, an expiration date, a unique serial number assigned to the certificate by the CA, the CA's signature, and an algorithm identifier that indicates the particular algorithm that was used to sign the certificate.

A certificate is created when an entity's public key is signed by a CA. Because a certificate is signed by a trusted authority and is obtained in a secure manner, it does not need to be validated for authenticity each time it is accessed. A client or a server can validate that an entity is who it claims to be by verifying that the entity's certificate was issued by a known and trusted certificate authority. A server need be consulted only to find out if the certificate has been revoked. Clients and servers can use these credentials to access secure services, such as SSL, using public key cryptography.

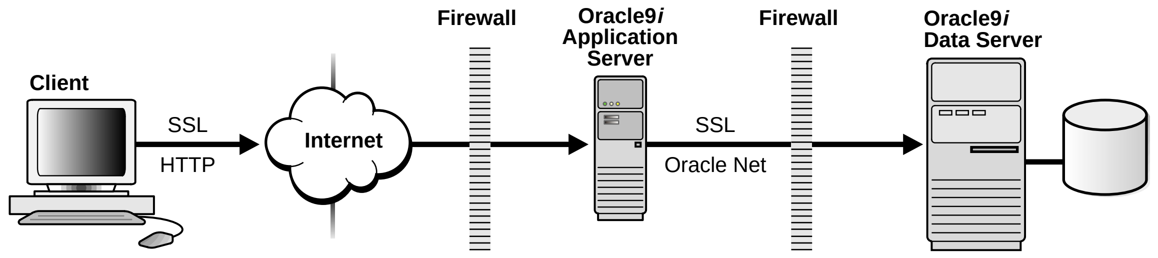
Although every certificate is signed by some known and trusted certificate authority, the particular CA named by a certificate might not be known to the verifying system. To enable that system to trust this certificate, the certificate also names the CA who validated its CA, and the CA before that who validated the earlier CA. This becomes a list of the trusted CAs whose prior verification processes culminated in validating the received certificate. The earliest of these is called the root certificate. Oracle9*i* Application Server provides several default trusted root certificates, so that users do not have to install their own.

Secure Sockets Layer (SSL) Authentication and X.509v3 Digital Certificates

The Secure Sockets Layer (SSL) protocol, developed by Netscape Corporation, is a widely accepted standard for network security. It provides authentication, data encryption, and data integrity, in a public-key infrastructure. SSL is widely employed over the Internet to give users established digital identities and to prevent eavesdropping, tampering with, or forging messages. SSL uses digital certificates (X.509 v3), and a public/private key pair to authenticate users and systems. SSL is supported by all currently available Web servers and Web browsers. It is also gaining acceptance for other protocols, including LDAP and IMAP.

SSL addresses the problem of protecting user data exchanged between tiers in a multitier system. By providing strong, standards-based encryption and integrity algorithms, SSL provides system developers and users with confidence that data will not be compromised on the Internet. Unlike password-based authentication, which authenticates client to server only, SSL can authenticate server to client as well as client to server. This feature is useful in a multitier system that is exposed to the Web, because users want the server authenticated before providing sensitive information, such as credit card numbers. Figure 1–2 illustrates SSL-protected communication links to the Oracle server from a remote client through the Internet and an Oracle Application Server.

Figure 1–2 SSL Secures Internet and Oracle Communications



See Also: "Secure Sockets Layer (SSL) and PKI with Oracle HTTP Server" on page 2-30

Storing Secure Credentials in an LDAP-Compliant Directory

Many organizations manage users and authorizations separately in an LDAP-compliant directory. They can also store credentials securely in the directory, enhancing their ability to manage users centrally. Doing so also supports user mobility.

With PKI, secure credentials such as digital certificates can be stored in containers called "wallets." A wallet is used to manage authentication data such as keys and the trusted certificates needed by the Secure Sockets Layer. Wallets can be stored in an LDAP-compliant directory. Security administrators use a tool such as Oracle Wallet Manager to manage security credentials on the server. Wallet owners also use this tool, to manage security credentials on clients.

Public Key Certificate Standards are a set of security standards laid out by the PKI vendor RSA. In particular, Oracle9*i* Application Server supports PKCS#12, the standard for secure credential storage.

See Also: "Oracle Internet Directory Overview" on page 2-12

Single Sign-on

Single sign-on (SSO) to Web-based applications enables users to log on only once to access multiple databases and services. The user need not remember multiple separate identities and passwords, because SSO handles all that after the initial user log-in to each registered application or resource. SSO can be used with security features such as SSL, certificates, and IP checking, as explained later in the SSO chapter of this book.

This section explains why the single sign-on feature saves time and improves security.

It starts with the fact that more and more companies are deploying diverse Web-based e-business applications for use by employees, customers, and partners. Each application typically requires a user or account ID and a password.

From the user’s point of view, keeping track of so many account and password pairs is tedious and annoying. The temptation to use shortcuts, as described below, reduces security not only for the individual user but also for the entire community of users. Each breach potentially exposes some or all of the others to damage or loss of confidentiality, whether malicious or unintentional.

From the administrator’s point of view, maintaining multiple accounts and passwords for each user is expensive and potentially insecure.

Single sign-on therefore benefits users while easing the burdens of coping with Internet-related volumes and of maintaining security in rapidly changing circumstances.

This section discusses the following points:

- Multiple Accounts and Passwords Are Insecure
- Multiple Passwords Are Expensive

Multiple Accounts and Passwords Are Insecure

Most users cannot remember more than a few passwords. Users who maintain more than one login account often choose passwords that are easy to remember, choose identical passwords for different accounts, reuse passwords when asked to change them, or write passwords down. All these practices reduce password security.

Writing passwords down or choosing easily remembered (and thus easily guessed) passwords increases the risk of passwords being compromised. When a password is compromised, the potential damage increases if users have reused passwords when asked to change them, or have used the same password on multiple systems.

Many systems implement password management mechanisms that force users to choose complex passwords, or prevent them from reusing a password. However, these mechanisms often backfire because users figure out ways to defeat them that may reduce security even more. For example, forcing users to use random passwords almost guarantees that they will be written down.

Changes in a user’s membership or functional roles in an organization should cause appropriate corresponding changes in the user’s privileges when using the organization’s applications. Multiple independent accounts per user make it even more likely that associated user privileges will not correspond with organizational changes. User accounts and access privileges, for example, may remain unchanged in the system long after the user has left the organization or changed roles. This circumstance leaves the system vulnerable to potential attack by disgruntled former employees.

Single sign-on's single point of entry empowers users while they are active, and also gives administrators a single point of control for changing conditions.

Multiple Passwords Are Expensive

Managing multiple accounts and passwords per user is expensive. In many enterprise deployments, a substantial fraction of the system administrator's time is spent on account- and password-related problems. This time includes initial creation of users' accounts when they join the organization and deletion of accounts when they leave. It includes changing settings when they change roles, and resetting passwords that have been forgotten. Administrators must sometimes access multiple systems to add or remove user accounts on each, using multiple administrative interfaces that can have different requirements. All this is expensive, and prone to various types of errors. Single sign-on reduces both types of cost.

See Also:

- "Centralized User Provisioning and Single Sign-On in Oracle9iAS" on page 2-10
- Chapter 3, "Configuring Oracle9iAS Single Sign-On"

Authorization Considerations

Authorization is the process of controlling access to data, resources, or services based on the identity of the user, host, or client. Proper authorization ensures that a user, program, or process receives the appropriate privileges. Once the user is identified through the process of authentication, his appropriate authorization can be determined.

In enterprise systems, it is often desirable to centralize authorization management and control of privileges in the directory. This approach has two important aspects:

- Single station administration (SSA). This feature addresses the problem of system administrators having to access multiple administrative tools to configure accounts for the different applications that each user accesses. SSA allows system administrators to set up, administer, and delete accounts for multiple applications from a single administrative interface.

- Single source of control (SSC). This feature addresses the problem that user privileges are often separately maintained by each application. SSC provides a central source of information on user roles and privileges for multiple applications, so that changes in user roles or privileges take effect immediately and coherently in all applications managed by the single source.

Consider, for example, a user who needs access to 40 servers within an enterprise. With single station administration, the administrator can set up access for the user from one place: she need not log in to 40 separate machines. To change the user's privileges on the 40 servers, the system administrator can utilize a single source of control, performing the change at a single server.

Nevertheless, in some circumstances single sign-on is a better solution than is central authority for the organization's needs. With Oracle9*i* Application Server, both choices are available for appropriate use.

See Also: "Delegated Administration Service (DAS)" on page 2-18

Oracle9iAS Single Sign-On Developer's Guide

Encryption Considerations

Sensitive information that travels over an intranet or the Internet can be protected by encryption. Encryption is the transformation of information into a pattern readable only with a decryption key. This security mechanism is powerful because decryption can be practically infeasible if you do not possess the decryption key.

Consider, for example, an Internet buyer who wishes to purchase a company's product using a credit card in a secure fashion. The buyer's credit card number is encrypted with an encryption key. The encrypted credit card number is sent across the network to the database. Encryption can be used to scramble the message, rendering it unreadable to anyone but the recipient, although this encryption is not mandated since the number itself is encrypted. The server decrypts the message with a decryption key and reads the credit card number.

Encryption must address all communications with the database, including transmissions from clients and transmissions from middle tiers. It must also secure all protocols into the database. Table 1–4 lists encryption algorithms that have become industry standards for the encryption and decryption of data.

Table 1–4 Encryption Algorithms

Algorithm	Characteristics
RSA Data Security RC4	Allows high-speed encryption for data privacy. Using a secret, randomly-generated key unique to each session, all network traffic is fully safeguarded—including all data values, SQL statements, and stored procedure calls and results. The client, server, or both, can request or require the use of the encryption module to guarantee that data is protected. This is a stream cipher, that is, an encryption method that works on its message one bit at a time. It is optimized for data sent over the network, but is not appropriate for data which is to be stored or reused (cookies, for example).
Data Encryption Standard (DES)	Safeguards network communications with symmetric key cryptography used by U.S. Data Encryption Standard algorithm (DES). DES is required for financial institutions and many other institutions. This is a block cipher, that is, a symmetric method that encrypts a message by breaking it down into blocks and then encrypts each block. This type of encryption is best for encrypting stored data such as cookies and tokens. It is not optimized for data to be sent over the network.
Triple DES (3DES)	Encrypts message data with three passes of the DES algorithm. 3DES provides a high degree of message security, but with a performance penalty—the magnitude of which depends on the speed of the processor performing the encryption; 3DES typically takes three times as long to encrypt a data block as the standard DES algorithm takes. Like DES, 3DES is a block cipher appropriate for stored data.

Note that the secrecy of encrypted data depends on the existence of a secret key shared between the communicating parties. Providing and maintaining such secret keys is known as key management. In a multiuser environment, distributing keys securely can be difficult, since it's hard to be certain that secrecy is maintained by the parties and that the transmission remains secure. Public-key cryptography was invented to solve this problem.

Data Integrity Considerations

In addition to encryption, there are integrity algorithms that can ensure data has not been tampered with and that packets have not been replayed. In a replay attack, an observer inserts a copy of something you entered earlier for a different purpose, disrupting what's going on and possibly having destructive effects. These integrity algorithms can be used to detect corruption in data blocks.

Table 1–5 Integrity Algorithms

Algorithm	Characteristics
MD5 Checksum	Provides data integrity through hashing to ensure that data is not altered or stolen as it is transmitted over a network. It enables the system administrator to detect alterations to data. The MD5 algorithm transforms a message of any size into a 128-bit message digest. Mathematicians believe each digest is for all practical purposes unique, in that it is computationally infeasible (practically impossible in any reasonable time) to create two different messages having the same message digest, or to generate a message that will have a prespecified target message digest. The MD5 algorithm is intended for digital signatures, where a large file must be compressed reliably and securely before being encrypted with a private key in a PKI system. In essence, MD5 verifies data integrity more reliably than checksum or other commonly-used methods.
Secure Hash Algorithm (SHA)	Similar to MD5, this method produces a larger message digest for greater security. SHA is a U.S. government standard.

Web Browser Security

A Web browser enables people to use the Internet conveniently by searching, accessing Web sites, sending and receiving transactions, e-mail, and instant messages. A secure browser is essential, so that sensitive data sent or accessed over the Internet is not corrupted or copied by an intruder. Examples of such data include customers' names, e-mail IDs, addresses, and credit card numbers.

In the overall system security picture, the Web browser may be the component over which e-business sites have least control. When running a Web storefront, for example, you may not be able to control the browser that customers use. The customer's browser nonetheless impacts the security of your system, and must be taken into consideration.

Most commercially available Web browsers support a number of security-related features. However, users must configure the browser properly, to take advantage of its security capabilities.

Full consideration of browser security issues is beyond the scope of this document.

Note: Oracle Corporation does not support Global Server ID, also known as "Server-gated cryptography." It does support strong encryption (128-bit encryption or Triple DES) if the user's browser also supports strong encryption.

See Also: Documentation provided with your Web browser

Security for Database Access

Oracle9*i* Application Server (Oracle9*i*AS) can be used as a client to the database. In this configuration, all the fundamental database security concepts apply. When designing such a system, you can employ a number of powerful security features of the Oracle9*i* database. This section provides an overview of the major server security issues, and describes how they are addressed in Oracle9*i* Application Server:

- Enterprise User Security
- Authentication and Digital Certificates
- Connecting From the Middle Tier to the Database
- Proxy Authentication

Enterprise User Security

In Oracle9*i* Application Server, enterprise user security is provided by the Oracle9*i*AS Portal component. This section introduces users and groups, and the relationship of users to database schemas, as reflected in the architecture of Oracle9*i*AS Portal.

In Internet computing, where millions of users may access a portal, user representation must be as lightweight as possible. It is important to avoid a situation in which each user must have a corresponding database schema and a distinct database login.

With Oracle9*i*AS Portal, a user account does not require its own database schema. Oracle9*i*AS Portal is primarily implemented in PL/SQL, so a database account is still needed for execution of the PL/SQL code. Users are by default mapped onto a single schema for executing procedures. When Oracle9*i*AS Portal is installed, a default set of user accounts is created: for the database administrator, for the portal administrator, and for the public.

With Oracle9*i*AS Single Sign-On, the administrator specifies a user name and password and the Oracle9*i*AS Single Sign-On privilege level to be given to the user. This privilege level is either End User or Full Administrator. The Full Administrator privilege is required for creating users and other Oracle9*i*AS Single Sign-On administration tasks.

Oracle9*i*AS Portal supports groups, which provide a convenient means of granting privileges to a collection of users in one action. In addition, certain attributes in the Oracle9*i*AS Portal system can be associated with a group, and if a user has a default group specified in his preferences, then those attributes can be applied to his session. Examples of this include a default home page for a group, or a default style.

Management of users, groups, and permissions is increasingly done by a directory. Oracle Enterprise Security Manager is the graphical user interface used to centrally administer enterprise users and enterprise roles in an LDAP directory, such as Oracle Internet Directory. System administrators can use this tool to perform a variety of tasks, such as creating new enterprise domains, assigning enrolled users and published databases to an enterprise domain, and authorizing enterprise roles on systems in the enterprise domain.

Oracle Enterprise Security Manager, which launches out of Oracle Enterprise Manager, scales to tens of thousands of users, and enables you to manage thousands of users and systems in various domains.

See Also: "Oracle Internet Directory Overview" on page 2-12

Authentication and Digital Certificates

Authentication is the process of verifying the identity of a user being presented to the system, usually in a login screen. Typically, the user enters a user name for identification and password for verification.

Authorization depends on authenticating the person or entity requesting the processing of a particular HTTP request. A number of facilities in Oracle HTTP Server can be invoked to establish various forms of user authentication. These include:

- X.509 certificates
- Basic authentication

Although basic authentication is a standard Web authentication mechanism, it is relatively insecure because user code and password pairs are essentially transmitted in the clear as a base 64-encoded string. Where security is very important, SSL facilities should be invoked because they protect the confidentiality of the challenge and response dialog used when Oracle HTTP Server requests user code and password information.

After the user's identity has been verified (authenticated), rules can be applied to restrict or allow further processing of URL requests. Also, user authentication can be combined with IP and hostname requirements so that processing is allowed only when either or both directives are satisfied.

See Also:

- "Centralized User Provisioning and Single Sign-On in Oracle9iAS" on page 2-10
- "Authorization, Authentication, and SSL in Oracle9iAS" on page 2-28

Connecting From the Middle Tier to the Database

When you use Oracle9*i* Application Server as a secure connection to an Oracle9*i* database, the application server must authenticate itself to the database. These authentication procedures are different from the authentication procedure used by a Web browser.

See Also: "Middle-Tier Connection Management" on page 9-5

Proxy Authentication

Proxy authentication is a feature of the Oracle9*i* database that enables the administrator to allow middle tier connections only on behalf of a particular set of users. The middle tier can authenticate itself, and then establish a lightweight session for its users without the need to authenticate each user separately.

Further, the Oracle9*i* database can be configured to empower a specific middle tier to assume a specific set of database roles on behalf of a specific user. In other words, the database uses both middle-tier and client user identity to determine the middle tier's privileges when acting for a user through a lightweight session.

See Also: "Proxy Authentication with Oracle9iAS" on page 9-2

JAAS

Because Java development is very important to the Web, Java security is a vital feature of Oracle9*i* Application Server. JAAS is part of a larger set of Oracle9*i* Application Server security services that includes Web single sign-on, network encryption, and other features. JAAS provides core security services for developing Java-based applications. Oracle Corporation designs all of these products to the emerging industry standards for Java security.

JAAS is the Oracle implementation of Java Authentication and Authorization Service (JAAS), a Java package that supports user authentication and access control. While the JAAS specification is not yet integrated with the J2EE security model, JAAS does integrate JAAS with J2EE for developers of Java applications in an Oracle environment. This enables developers to add core security functionality to their Java applications, which J2EE alone cannot provide.

JAAS provides key security services in the following areas:

Authentication	Identifying users
Authorization	Limiting what users can do
Delegation	Enabling code to run securely, with privileges of other users

JAAS provides benefits to any customer developing Java applications. JAAS is also used by a number of Java components within Oracle9*i* Application Server to provide authentication and authorization services within those components.

See Also:

- "JAAS Security" on page 2-38
- *Oracle9iAS Containers for J2EE Services Guide* in the Oracle9iAS Documentation Library where JAAS support is discussed.

Firewalls

To eliminate potential weak points in the network infrastructure, you may opt to pass data from protocol to protocol without the complexity of decryption and re-encryption. To do so securely, you must have some way to transfer data securely across network protocol boundaries.

The Internet enables you to connect your corporate intranet to a broad public network. Although this capability provides enormous business advantages, it also entails risk to your data and your computer system. One way of protecting the privacy and integrity of your system is to place a firewall between the public network and your intranet.

A firewall is a single point of control on a network, used to prevent unauthorized clients from reaching the server. It acts as a filter, screening out unauthorized network users from using the intranet. It does this by enforcing access controls based on the contents of the packets of data being transmitted, and can thus protect against attacks on individual protocols or applications.

Firewalls are rule-based. They have a list of rules that define which clients can connect, and which cannot. They can compare the client's hostname or IP name with the rules, and either grant the client access, or not.

Summary

The Internet provides enormous and unparalleled opportunities for both expansion of and efficiencies in communication, commerce, and business practices. Dangers accompany almost every one of these opportunities. These include threats to our privacy, our savings, and our confidence in the reliability of our daily interactions in the personal, professional, and commercial arenas.

These threats can be reduced or eliminated by selecting mechanisms, policies, and practices that enforce both accountability and security limitations in each area of vulnerability. Secure communications can make eavesdropping useless.

Authentication can limit inappropriate system access, and authorization policies can restrict the possibilities for inappropriate data visibility or manipulation.

Each of the applications described in the later sections of this book has built-in security features that protect users and data from the vulnerabilities discussed in this chapter. These features also combine to work seamlessly with the other security software that Oracle supplies and supports.

Once you have understood and mastered the principles and techniques presented in this book, you will be prepared to take the steps necessary to providing effective security for your sites and systems.

See Also:

- Chapter 2, "Oracle9i Application Server Security Architecture and Features" for information about how Oracle9*i*AS addresses these security threats.

2 Oracle9*i* Application Server Security Architecture and Features

Oracle9*i* Application Server (Oracle9*i*AS) provides a comprehensive security framework supporting all Oracle9*i*AS components, as well as third-party and custom applications deployed on the application server. The framework is based on Oracle9*i*AS Single Sign-On for authentication, Oracle Internet Directory for authorization and centralized user provisioning, and the Oracle Java Authentication and Authorization Service (JAAS) provider for security in Java2 Enterprise Edition (J2EE) applications.

This chapter provides an overview of the security architecture and features of Oracle9*i*AS in the following topics:

Introduction to Oracle9*i*AS

Oracle9*i*AS is a reliable, scalable, secure, middle-tier application server designed to support a company's evolution into an e-business. With this product, the technological complexity of assembling a complete middle-tier Internet foundation is managed for you. The technological foundation that Oracle9*i*AS provides can grow with your business. It can start small and support growing numbers of users and sophisticated functionality on all of your Web sites.

Oracle9*i*AS includes components which provide a general framework for development and deployment of applications, as well as components that provide specific application services or functionality. This chapter focuses on the security services provided by the Oracle9*i*AS Infrastructure, which includes Oracle9*i*AS Single Sign-On Server and Oracle Internet Directory, an LDAP, version 3-compliant directory service. It also provides an overview of the security services provided by Oracle HTTP Server, Oracle9*i*AS Web Cache, Oracle9*i*AS Portal, and JAAS (Java Authentication and Authorization Service), which provide support for a broad range of application development and deployment strategies.

Security Architecture of Oracle9*i*AS

Oracle9*i*AS provides a solid framework for building and deploying Web applications using the Apache-based Oracle HTTP Server, Oracle9*i*AS Containers for J2EE, and Oracle9*i*AS Portal, which use the advanced security functionality provided by Oracle9*i*AS Infrastructure. Oracle9*i*AS Infrastructure consists of Oracle9*i*AS Metadata Repository, Oracle Internet Directory, Oracle9*i*AS Single Sign-On, and Oracle Management Server. Oracle9*i*AS security starts from the well-tested and highly configurable Web security services provided by Oracle HTTP Server, adds a comprehensive set of Web single sign-on services, and extends them further with centralized user provisioning that is available in Oracle Internet Directory, an LDAP, version 3-compliant directory service. In addition, Oracle9*i*AS provides the Oracle implementation of Java Authorization and Authentication Services (JAAS) for J2EE application security, and extensive portal authorization and application integration mechanisms. Oracle9*i*AS also supports secure access to Oracle database systems using Oracle Advanced Security.

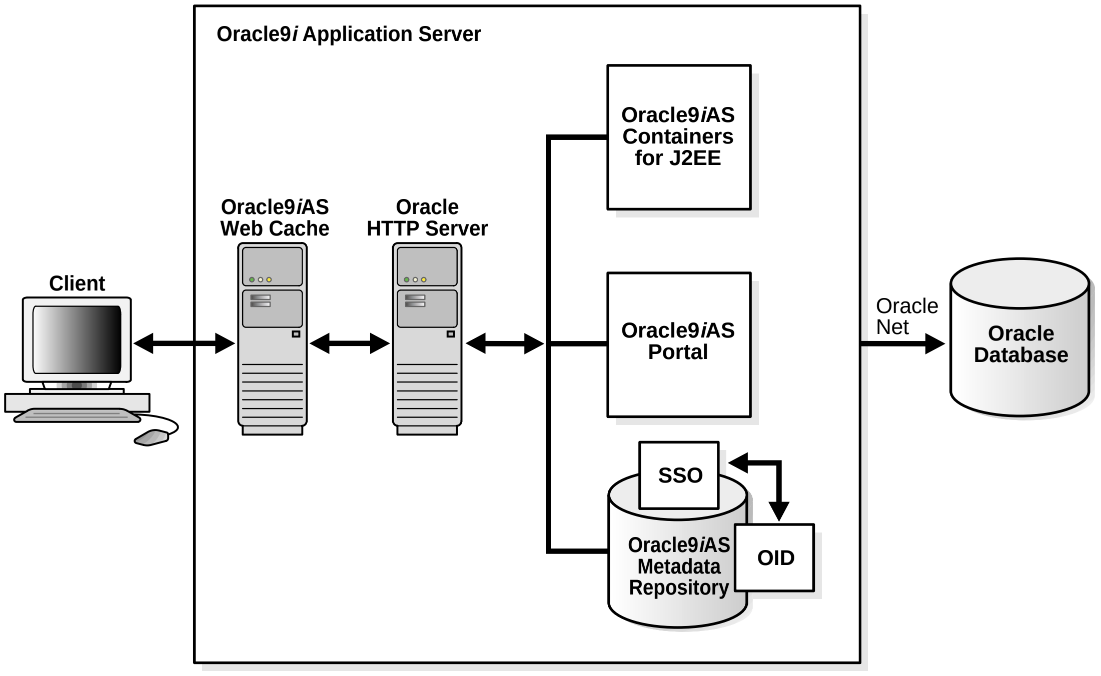
This section introduces the application server security architecture, and shows how the different elements provide the security features required in an Internet environment.

> **See Also:** "What to Install to Get the Security You Require" on page 2-7 for a description of Oracle9*i*AS Infrastructure and a summary of software requirements for security.

Elements of Oracle9*i*AS Security Architecture

Figure 2–1, described below, illustrates how the elements of Oracle9*i* Application Server function together.

Figure 2–1 Security Architecture of Oracle9i Application Server



Oracle9*i*AS Security Features	Description
Oracle9*i*AS Web Cache	The Web cache, which can be configured to support HTTPS, is positioned in front, where it caches frequently accessed Web pages or partial pages.
Oracle HTTP Server	The Web server supplies Web listener services for both HTTP and HTTPS and, through plug-ins, routes requests for authentication and authorization.

Oracle9*i*AS Security Features (Cont.)	**Description (Cont.)**
Oracle9*i*AS Containers for J2EE and the JAAS	Oracle9*i*AS Containers for J2EE provides the Java runtime environment for Oracle9*i*AS components. The JAAS ensures secure access to and execution of Java applications along with integration of Java-based applications with Oracle9*i*AS Single Sign-On.
Oracle9*i*AS Portal	The portal provides the infrastructure to create and manage Web pages. It lets you display multiple Web pages on each portal page, with links to content through Java applications. The portal uses Oracle9*i*AS Single Sign-On to provide single sign-on capabilities for secure access to content and applications.
Oracle9*i*AS Single Sign-On	The single sign-on (SSO) feature provides a single, unified authentication service to Oracle9*i*AS components, applications, and Web pages. The Single Sign-On server stores and authenticates users against Oracle Internet Directory.
Oracle Internet Directory	This LDAP, version 3-compliant directory serves the middle tier by providing authentication and a centralized user model whereby users can be created, managed, and stored.
Oracle9*i*AS Metadata Repository	This is an Oracle9*i* Enterprise Edition database, which is pre-seeded with the schemas used by Oracle9*i*AS components.

Note: If SSO capabilities are required, then Oracle9*i*AS Infrastructure, which installs the Oracle9*i*AS Single Sign-On server and Oracle Internet Directory must be installed first before installing all other components.

See Also: *Oracle9i Application Server Installation Guide* in the Oracle9*i*AS Platform-specific Documentation for complete installation information.

Oracle9*i* Application Server Implementation of Public Key Infrastructure (PKI)

The Oracle9*i* Application Server PKI implementation provides a variety of security services, in compliance with industry-standard specifications. It incorporates a whole suite of products and features, including the following:

Secure Sockets Layer

The Secure Sockets Layer (SSL) is an application layer protocol that can be employed for certificate-based authentication. All of the major components of Oracle9*i*AS support SSL.

> **See Also:** "Authorization, Authentication, and SSL in Oracle9iAS" on page 2-28 for information about SSL support in Oracle9*i*AS.

Oracle Wallets

An Oracle wallet is a container in which certificates and trusted certificates are stored and managed. These data structures securely store a user private key, a user certificate, and a set of trusted certificates (the list of root certificates which the user trusts).

Oracle Wallet Manager

This is a Java-based application that security administrators use to manage public-key security credentials on both Oracle clients and servers. It creates an Oracle wallet. Oracle Wallet Manager creates a public-private key pair and manages credentials for a user. It issues PKCS#10 certificate requests to the certificate authority, and installs the certificate in the wallet. It ships with trusted certificates from VeriSign, RSA, and Baltimore CyberTrust, and can use a site’s own in-house certificate authority.

> **See Also:** Chapter 5, "Using Oracle Wallet Manager" for information about Oracle Wallets and Oracle Wallet Manager.

Oracle Internet Directory

Oracle Internet Directory, an LDAP V3-compliant directory built on the Oracle9*i* database, helps to enable PKI-based single sign-on. It enables you to securely manage the user and system configuration environment, including security attributes and privileges, for users authenticated using X.509 certificates. Oracle Internet Directory enforces attribute-level access control, enabling the directory to restrict read, write, or update privileges on specific attributes to specific named users (for example, a security administrator). It also supports protection and authentication of directory queries and responses through SSL encryption.

> **See Also:** "Centralized User Provisioning and Single Sign-On in Oracle9iAS" on page 2-10 for information about Oracle Internet Directory and centralized user provisioning in Oracle9*i*AS.

What to Install to Get the Security You Require

Oracle9*i*AS provides several installation options so you can install only the functionality that you require. The two primary software installations that involve security functionality are

- Oracle9iAS Infrastructure Installation
- Oracle9i Application Server Installation

Oracle9*i*AS Infrastructure Installation This installation provides single sign-on (SSO), LDAP directory, and centralized management features. It installs Oracle9*i*AS Metadata Repository, Oracle Internet Directory, Oracle9*i*AS Single Sign-On, and Oracle Management Server, and it configures all except Oracle Management Server. You must first install and configure the infrastructure before installing Oracle9*i*AS components with the Oracle9*i* Application Server Installation.

Oracle9*i* Application Server Installation This installation provides several installation types that allow you to choose which components of Oracle9*i*AS you want to install. For example, you can select the basic installation type, "J2EE and Web Cache," which includes a Web server, a Java runtime environment, and a Web cache. Or you can select to install all components of Oracle9*i*AS, which includes the basic set of software plus the functionality to enable portals, wireless applications, business intelligence, and messaging.

The components in this type of installation can be enabled to support PKI over SSL without installing Oracle9*i*AS Infrastructure. However, the infrastructure supports functionality that is not directly related to security for many of the components in the Oracle9*i* Application Server Installation.

Table 2–1 lists what you need to install and configure to get the security features that you require for your Web sites and applications.

Table 2–1 Installation Requirements for Security Services in Oracle9iAS

If you need...	**Then install and configure...**
SSL and PKI across the middle tier	Oracle9*i* Application Server Installation
JAAS to develop and deploy secure Java applications	Oracle9*i* Application Server Installation
Single sign-on and centralized user provisioning with an LDAP directory	Oracle9*i*AS Infrastructure Installation first, then install and configure Oracle9*i* Application Server Installation

See Also: *Oracle9i Application Server Installation Guide* in the Oracle9*i*AS Platform-specific Documentation for complete installation information.

Default User Password Policy in Oracle9*i*AS

When you install Oracle9*i*AS Infrastructure and use Oracle Internet Directory for security management, a default security implementation is provided. These default security settings include a default user password policy. This password policy applies to all user entries that are managed in the user container that is identified by the subscriber's user searchbase (orclcommonusersearchbase) in the directory. Table 2–2 lists the default user password policy in Oracle9*i*AS when Oracle Internet Directory is used for security management.

Table 2–2 Default User Password Policy in Oracle9iAS

Password Policy	Value
Password expiration	60 days
Number of failed attempts before account lockout	10
Minimum number of characters required in the password	5
Minimum number of numeric characters required in the password	1

See Also:

- *Oracle9i Application Server Release Notes* in the Oracle9*i* Application Server Platform-specific Documentation for any changes in the password policy.
- "Centralized User Provisioning and Single Sign-On in Oracle9iAS" on page 2-10 for information about Oracle Internet Directory.
- "Oracle Internet Directory Administration and Delegation Model in Oracle9*i* Application Server, Release 2," a white paper that describes the DIT (directory information tree) that is configured by default when you install Oracle Internet Directory. This white paper is available on Oracle Technology Network at:

  `http://otn.oracle.com/docs/index.htm`

Centralized User Provisioning and Single Sign-On in Oracle9*i*AS

When you install the Oracle9*i*AS Infrastructure, you enable single sign-on (SSO) and centralized user provisioning. SSO and centralized user provisioning are provided by Oracle9*i*AS Single Sign-On and Oracle Internet Directory. When these services are installed, the installation program automatically performs the following tasks that are related to security:

- Installs and configures Oracle Internet Directory for your environment according to the information that you supply during installation
- Sets up a default administration and delegation structure for Oracle products in the **directory information tree (DIT)** of Oracle Internet Directory
- Installs and configures Oracle9*i*AS Single Sign-On server for your environment according to the information you supply during installation
- Configures Oracle HTTP Server module, mod_osso, so the HTTP server can route requests to the Single Sign-On server for applications that use SSO

The security services that are provided by Oracle9*i*AS Infrastructure are described in the following topics:

- Overview of Centralized User Provisioning and Single Sign-On Model
- Benefits of Centralized User Provisioning and Single Sign-On
- Overview of Security in Oracle Internet Directory
- The Oracle Context: A Directory Administration and Delegation Model
- How Oracle Internet Directory is Implemented
- Delegated Administration Service (DAS)
- Oracle9iAS Single Sign-On Overview
- Oracle9iAS Single Sign-On Components
- Functional Overview of Oracle9iAS Single Sign-On

Overview of Centralized User Provisioning and Single Sign-On Model

When you install Oracle9*i*AS Infrastructure, all users and groups who access Web applications are created, stored, and maintained in Oracle Internet Directory and authenticated with Oracle9*i*AS Single Sign-On.

In this model, the following steps outline how users are created and authenticated so they can access Web applications:

1. Initially, users' identity entries are created in the directory using a Web-based GUI tool, Delegated Administration Service, which is a component of Oracle Internet Directory. Depending on your internal security policies, users may create their own entries, or central administrators may create user identity entries in the directory. Typically, if user entries are created by administrators, then users can modify certain information, such as phone numbers and addresses.
2. Each time users need to access Oracle9*i*AS, they go to a central corporate server where they log in to Oracle9*i*AS Single Sign-On. During this step, users supply their user name and password.
3. The Single Sign-On server authenticates users against the directory and sets cookies in the users' browsers.
4. With SSO cookies placed in their browsers, then users can access all available Web applications listed in their personalized portal views, such as expense reporting, e-mail, or calendars.

Alternatively, users can also enter a URL for a specific Web application in their browsers, which causes the Oracle9*i*AS Single Sign-On login dialog box to appear so they can be authenticated before accessing the application.

Benefits of Centralized User Provisioning and Single Sign-On

Centralized user provisioning and single sign-on are important features of the comprehensive Oracle9*i*AS security framework. Together, these features reduce security risks and increase the efficiency of security administrators and employees.

The centralized user provisioning that is provided by Oracle Internet Directory, an LDAP version 3-compliant directory, makes maintaining separate stores of user identity and security information in various components of the application server unnecessary. Because all user identity and security information is stored in a central repository, administrators can manage users in one place, thereby reducing the time it takes to manage this information and reducing the likelihood that user identities can be compromised.

Oracle9*i*AS also supports single sign-on (SSO) to Web-based applications. There are a number of reasons why businesses are considering SSO. These include the increasing use of Web-based e-business applications which companies are deploying for use by employees, customers, and partners. Without SSO, each user must maintain a separate identity and password for each application accessed. When users must maintain multiple user names and passwords, it is easier for them to forget or lose them. When users must maintain multiple user names and passwords for various applications and information, administrators must keep track of them. In brief, maintaining multiple accounts and passwords for each user is insecure and expensive.

Using centralized user provisioning and single sign-on improves employee efficiency and makes your Internet infrastructure secure.

Oracle Internet Directory Overview

Oracle Internet Directory consolidates the management of users and groups in Oracle9*i*AS. It retrieves and stores information about dispersed users, groups, and network resources.

The directory implements version 3 of the Lightweight Directory Access Protocol (LDAP), which is the Internet standard for directory services. LDAP is based on the earlier ISO X.500 Directory Access Protocol (DAP) standard, but simplifies that standard considerably, enabling LDAP to be more efficient, straightforward, and easier to implement. LDAP is especially suited for deployment with "thin-client" applications that are developed for an Internet environment.

Oracle Internet Directory is not a security product, but rather a technology for managing enterprise data, including security data such as user names and passwords for Oracle9*i*AS Single Sign-On.

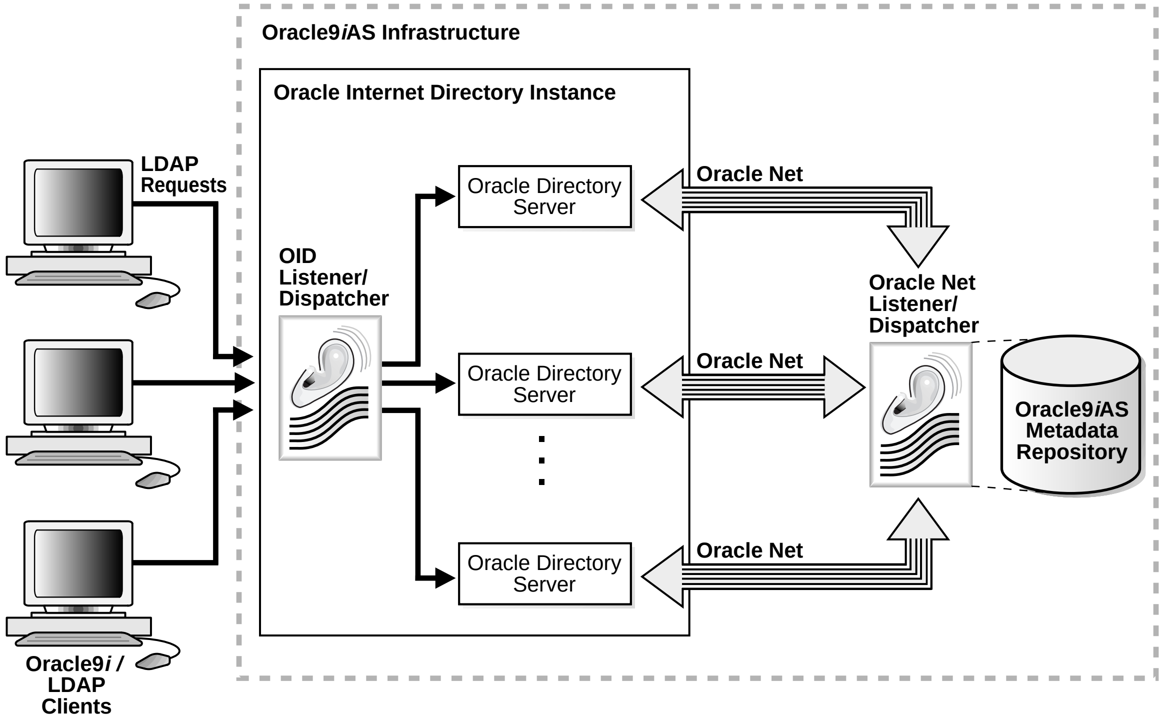
Each LDAP directory server instance looks like the configuration in Figure 2–2.

See Also: For complete information about Oracle Internet Directory, see the following documents, which are available in the Oracle9*i*AS Documentation Library unless otherwise specified here:

- *Oracle Internet Directory Administrator's Guide*
- *Oracle Internet Directory Application Developer's Guide*
- *Oracle Directory Service Integration and Deployment Guide*
- "Oracle Internet Directory Administration and Delegation Model in Oracle9*i* Application Server, Release 2," a white paper that is available on Oracle Technology Network at:

  `http://otn.oracle.com/docs/index.htm`

Figure 2–2 LDAP Server Instance Architecture



Overview of Security in Oracle Internet Directory

Oracle Internet Directory offers comprehensive and flexible support for directory access control. This includes **entry** level, attribute level, and prescriptive access control to provide varying levels of security to meet the specific needs of enterprise and service providers. An administrator can grant or control access to a specific directory object or to an entire directory subtree. The directory implements three levels of user authentication: anonymous, password-based, and certificate-based using Secure Sockets Layer (SSL) Version 3 for authenticated access and data privacy.

In addition, the directory provides many powerful features you can use in an enterprise or hosted environment to control access to application metadata—the information governing how applications behave and who can access them. To do this, you deploy the directory for administrative delegation. Using this deployment, a global administrator can give department administrators access to the metadata of applications in their departments. These department administrators can then control access to their department applications.

Oracle Internet Directory offers the following important security benefits:

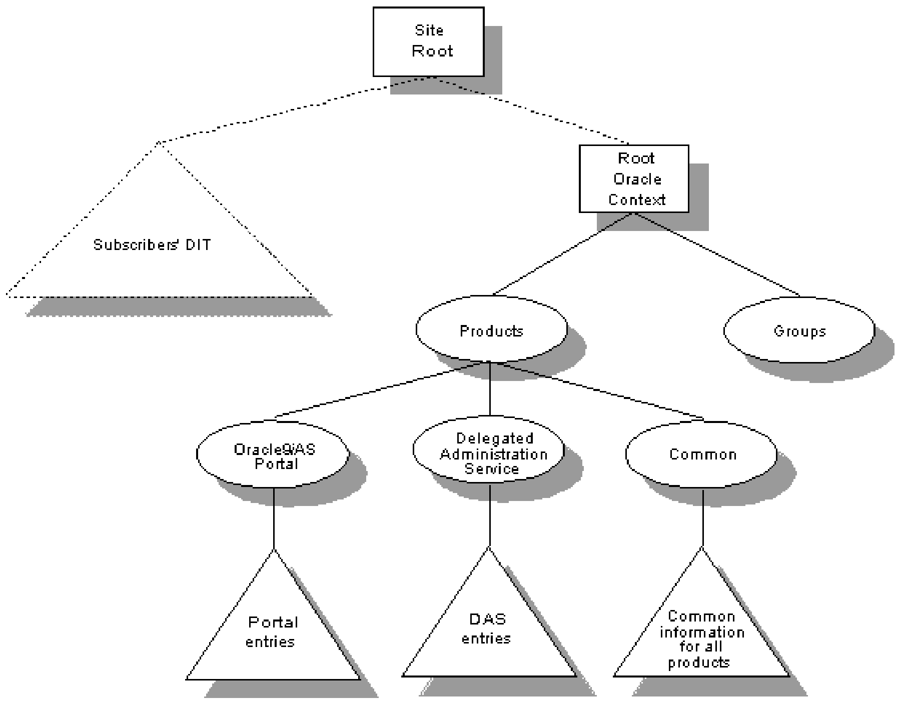
Security Benefit	Description
Data Integrity	Oracle Internet Directory uses SSL to ensure that data has not been modified, deleted, or replayed during transmission. SSL can generate a cryptographically secure message digest, through cryptographic checksums using either the MD5 algorithm or the Secure Hash Algorithm (SHA), and include it with each packet sent across the network.
Data Confidentiality	Oracle Internet Directory ensures that data is protected against undesired disclosure during transmission by using encryption available with SSL.
Password Protection	To protect passwords, Oracle Internet Directory uses the MD4 algorithm as the default. MD4 is a one-way hash function that produces a 128-bit hash, or message digest.
Data Access Control	Oracle Internet Directory supports access control down to the attribute level for read, write, or update of attributes.

The Oracle Context: A Directory Administration and Delegation Model

A directory stores all information pertaining to Oracle software in a root container called the Oracle Context. A starter Oracle Context is automatically created for you when you install the Oracle9*i*AS Infrastructure, but you can create an Oracle Context under any entry in the DIT (directory information tree). Oracle Net Configuration Assistant is a tool you can use to configure directory access. It displays a list of published directory entries as suggested locations from which you can build an Oracle Context.

Figure 2–3 shows a simplified view of the starter Oracle Context that is set up when you install Oracle9*i*AS Infrastructure. The starter Oracle Context looks very similar to a directory subtree with Products and Groups containers subsumed under the root Oracle Context. In this figure, there are containers for two products under the Products container, plus a container that holds all of the entries which are common to all of the Oracle9*i*AS products represented in the DIT.

Figure 2–3 Simplified View of DIT Structure with a Starter Oracle Context



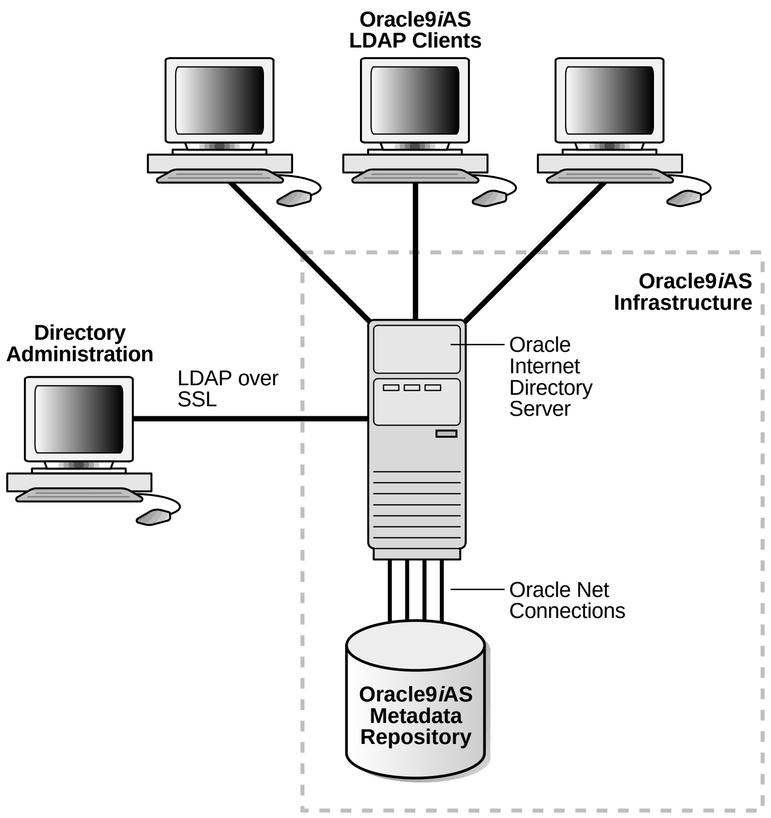
See Also: For a complete description of the starter Oracle Context that is set up when you install Oracle9*i*AS Infrastructure, refer to the white paper, "Oracle Internet Directory Administration and Delegation Model in Oracle9*i* Application Server, Release 2," on Oracle Technology Network at:

`http://otn.oracle.com/docs/index.htm`

How Oracle Internet Directory is Implemented

An Oracle Internet Directory node is implemented as an application running on the Oracle9*i*AS Metadata Repository. To communicate with the repository, which may be on the same system or on a different one, Oracle Internet Directory uses Oracle Net Services, the Oracle platform-independent database connectivity solution. This relationship is illustrated in Figure 2–4.

Figure 2–4 Oracle Internet Directory Architecture


Oracle9*i*AS LDAP Clients
Oracle9*i*AS Infrastructure
Directory Administration
LDAP over SSL
Oracle Internet Directory Server
Oracle Net Connections
Oracle9*i*AS Metadata Repository

Delegated Administration Service (DAS)

DAS is a Web-based GUI tool that directory administrators can use to create users, and users can use to modify their own personal data (such as addresses, phone numbers, and photos), without an administrator's intervention. Using DAS, users can also search other parts of the directory to which they have access, so administrators are free to perform other tasks.

This tool relies on small Java programs, called servlets. Servlets receive requests from clients, process those requests (by either retrieving or updating data in the directory), then generate results, which they send back to clients.

Oracle9*i*AS Single Sign-On Overview

Oracle9*i*AS Single Sign-On (SSO) technology provides single sign-on for Web users. It is designed to work in an environment such as that provided by Oracle9*i* Application Server, where multiple Web-based applications are accessible through a portal. The Oracle strategy for SSO encompasses a variety of technologies. For the growing field of Web-based applications, Oracle has developed an SSO framework, Oracle9*i*AS Single Sign-On, which is specifically designed to provide Web SSO.

The Oracle9*i*AS Single Sign-On approach has a number of benefits. It provides a framework for secure SSO from browser clients to Web-based applications, including Oracle applications and tools, through standard protocols. It supports both partner applications, which take full advantage of the SSO framework, as well as external applications for support of legacy and third-party products. Partner applications work within the SSO framework and rely on the SSO service for authentication of users; external applications continue to use their own user names and passwords. The Oracle9*i*AS Single Sign-On approach is based on cookies, which are created both by partner applications and by a centralized Oracle9*i*AS Single Sign-On server.

This section includes these topics:

- Oracle9iAS Single Sign-On Components
- Functional Overview of Oracle9iAS Single Sign-On

See Also:

- *Oracle9iAS Single Sign-On Administrator's Guide*
- *Oracle9iAS Single Sign-On Application Developer's Guide*

This documentation is available in the Oracle9*i*AS Documentation Library.

Oracle9*i*AS Single Sign-On Components

Oracle9*i*AS Single Sign-On is composed of the following components, which are described in the following sections:

- Oracle9iAS Single Sign-On Server
- mod_osso

Oracle9*i*AS Single Sign-On Server

Oracle9*i*AS Single Sign-On server is the core of Oracle SSO technology. It is well integrated with Oracle HTTP Server through mod_osso, which is an HTTP server plug-in that routes HTTP requests that require SSO to the Single Sign-On server for initial authentication. In addition, it allows management of user information in Oracle Internet Directory. Oracle9*i*AS Single Sign-On also permits integration with SSO technologies for other, non-Oracle applications. It supports PKI client authentication, which enables PKI authentication to a wide range of Web applications. By means of an API, Oracle9*i*AS Single Sign-On can integrate with third-party authentication mechanisms such as Netegrity Site Minder.

Two different types of applications can use Oracle9*i*AS Single Sign-On:

- **Partner Applications** Partner applications are those that are designed or modified to delegate authentication to the Single Sign-On server. Because partner applications take advantage of the authentication services of Oracle9*i*AS Single Sign-On, they do not need to implement their own authentication modules.
- **External Applications** External applications are those which retain their own user names and passwords, and do not delegate responsibility for authenticating users to Oracle9*i*AS Single Sign-On. These applications have not been developed or modified to work within the SSO framework. A typical external application might be one developed or deployed by a third party, such as a portal Web site which requires user name and password for access to custom services like e-mail.

mod_osso

The mod_osso component of Oracle9*i*AS Single Sign-On is a module that plugs into Oracle HTTP Server. It enables the HTTP listener as a partner application that can use the Single Sign-On server to authenticate users. Once mod_osso is installed and configured, Web applications can register URLs that require SSO authentication with the module. Then when URL requests are received by the HTTP server, mod_osso detects which requests require SSO authentication and redirects them to the

Single Sign-On server. After the Single Sign-On server authenticates the user, it passes the user's authenticated identity back to mod_osso in a secure token, or cookie. The module retrieves the user's identity from the cookie and propagates the user's identity information to applications running in the HTTP server instance. The module can propagate the user's identity information to applications running in the CGI, those running in Oracle9*i*AS Containers for J2EE, and it can also authenticate users for access to static files.

Note: For most applications, mod_osso is the preferred way to use Oracle9*i*AS Single Sign-On in Oracle9*i*AS.

Functional Overview of Oracle9*i*AS Single Sign-On

This section includes these topics:

- Initial Authentication
- Authentication to Partner Applications
- Authentication to External Applications
- LDAP Integration
- PKI Support
- Multitier Integration

Initial Authentication

When a user attempts to access a partner application, which includes mod_osso, for the first time, he is redirected to Oracle9*i*AS Single Sign-On. Oracle9*i*AS Single Sign-On checks to determine whether the user has a valid SSO cookie set; if not, it requests that the user authenticate by submitting a user name and password. Once the user has done so, the Oracle9*i*AS Single Sign-On verifies the password and then sets an SSO cookie in the user's browser. Henceforth, the client sends this cookie along with all HTTP interactions with Oracle9*i*AS Single Sign-On, authenticating itself and avoiding the need to reauthenticate for as long as the cookie is valid. Figure 2–5 on page 2-23 shows how partner applications are authenticated with Oracle9*i*AS Single Sign-On.

The SSO cookie is encrypted by Oracle9*i*AS Single Sign-On, so that it cannot be set or read by a third party. Cookies expire after a certain period of time, as set by the administrator (typically 8 hours); or when users shut down their browser. SSO cookies are not persistent, and are deleted when a browser is shut down.

Note that interactions between the Oracle9*i*AS Single Sign-On Server, browser clients, and applications are all by means of standard HTTP. No special requirements are placed on clients, other than that they support cookies. It is recommended that SSL be enabled between Oracle9*i*AS Single Sign-On Server and the client to prevent user names, passwords, and SSO cookies from being intercepted by a third party, who could possibly use them to spoof the Oracle9*i*AS Single Sign-On Server.

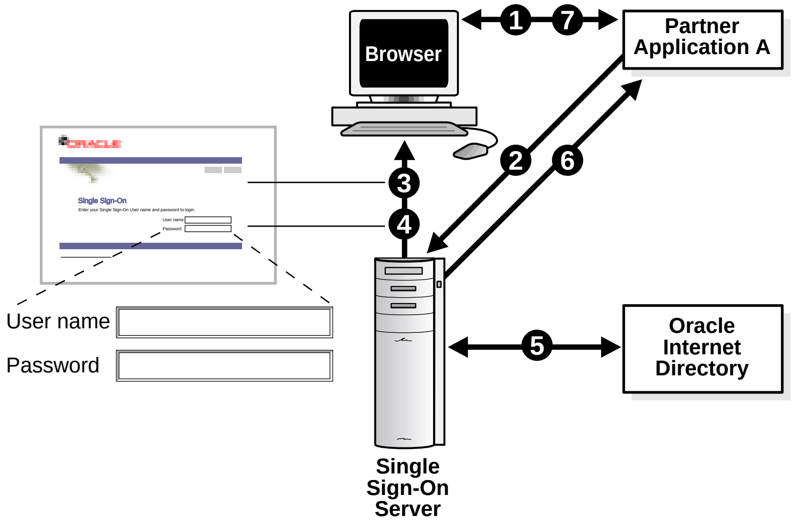
Authentication to Partner Applications

Once a user has been authenticated and an SSO cookie has been set, Oracle9*i*AS Single Sign-On directs the user back to the partner application, and includes an encrypted token in the partner application URL. This token contains the user's identity and session information. The token is encrypted in a key which is shared only by Oracle9*i*AS Single Sign-On and the partner application. This assures the partner application that the token is authentic, and was created by Oracle9*i*AS Single Sign-On.

When the partner application receives and decrypts the URL token, it can determine whether to grant the authenticated user access to the application. To grant access, it sets a partner application cookie in the user's browser. The client sends the partner application cookie along with subsequent HTTP requests to the partner application. This allows the application to identify and grant access to the user, without having to redirect the user to Oracle9*i*AS Single Sign-On for authentication.   Partner application cookies, like SSO cookies, expire after a certain period of time. Unlike SSO cookies, partner cookies may be persistent or non-persistent (that is, they may or may not survive the shutdown of a browser). The expiration period for a partner application cookie is determined by the application, and may be different than the SSO cookie expiration time. Figure 2–5 on page 2-23 shows how authentication with partner applications works.

As with SSO cookies, it is recommended that SSL encryption be used to protect the cookie exchange between browser and partner application.

Figure 2–5 Oracle9iAS Single Sign-On Authentication to Partner Applications



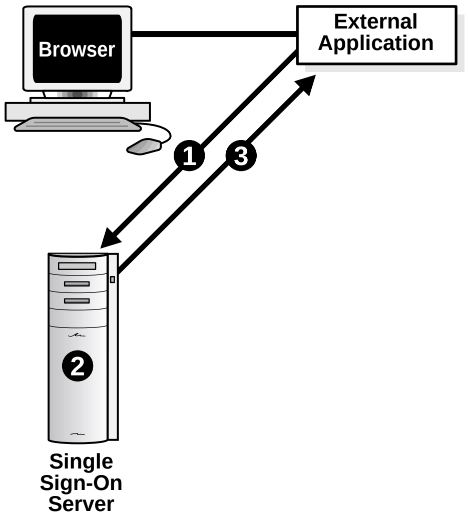
1. A user accesses Partner Application A. The application determines that the user is not authenticated because there is no Application A cookie.
2. Partner Application A redirects the user to Single Sign-On server.
3. Single Sign-On server displays a user name and password page that prompts the user to supply this information.
4. Single Sign-On server verifies the password and sets an SSO cookie in the user’s browser for authentication to Single Sign-On server.
5. SSO credentials are stored in Oracle Internet Directory.
6. Single Sign-On server redirects the user to Partner Application A with an encrypted token to authenticate the user to the application.
7. Partner Application A sets an application cookie in the user’s browser.

Authentication to External Applications

External applications cannot accept an authenticated identity directly from Oracle9*i*AS Single Sign-On. Oracle9*i*AS Single Sign-On provides SSO to external applications which support it by means of Web forms. Oracle9*i*AS Single Sign-On provides SSO by means of a password store, which maintains application-specific user names and passwords in a table within Oracle9*i*AS Single Sign-On. Access to this table is restricted by Oracle9*i*AS Single Sign-On, and passwords are further protected by encryption. When a user who has been authenticated by Oracle9*i*AS Single Sign-On needs to access an external application, Oracle9*i*AS Single Sign-On retrieves the user's user name and password for that specific application from the password store, formats them in the appropriate Web form, and submits them to the application. This is done transparently to the user. Figure 2–6 on page 2-25 shows how authentication to external applications works.

Note that SSL encryption between Oracle9*i*AS Single Sign-On and external applications can be used to prevent exposure of application passwords in the network.

Figure 2–6 Oracle9iAS Single Sign-On Authentication to External Applications



1. A user requests access to an external application. The application redirects the user to Single Sign-On server.
2. Single Sign-On server looks up the external user name and password.
3. Single Sign-On server sends the external user name and password to the external application.

LDAP Integration

LDAP directories are increasingly used as a single source of enterprise-wide information about users. These directories provide a convenient mechanism for provisioning (creating and configuring) and managing users who use multiple applications or servers in an enterprise. This is because LDAP is a widely supported, Internet-standard protocol, and because an LDAP directory can be used as a convenient, single source of information about users, accessible throughout the enterprise. Oracle Internet Directory is particularly well suited for this type of application, because it provides a secure, fast, scalable, and available directory service.

Oracle9*i*AS Single Sign-On allows SSO user names and passwords to be verified using Oracle Internet Directory. This is the default, unless you specify otherwise. When a user submits an SSO user name and password as part of the initial authentication, Oracle9*i*AS Single Sign-On performs an LDAP compare against Oracle Internet Directory using this user name and password. If the LDAP compare succeeds, the SSO user name and password are considered to be verified.

PKI Support

PKI authentication is beginning to replace passwords in many applications. In Web-based applications, PKI authentication is typically performed through an exchange of X.509 certificates, as part of a Secure Sockets Layer (SSL) session establishment. PKI by itself can be used to provide SSO, because a user with a certificate can authenticate to multiple applications without entering a password.

Users can authenticate to Oracle9*i*AS Single Sign-On by means of PKI. This provides SSO both to Web-based applications supported by Oracle9*i*AS Single Sign-On, and to other PKI-enabled applications. Instead of providing an SSO user name and password, users authenticate to Oracle9*i*AS Single Sign-On by means of SSL with client and server X.509 certificate exchange. Authentication to partner and external applications can then be performed using the cookie-based approach described previously. The directory translates the certificate identity to the SSO user name.

Note: Before the directory can translate the certificate identity to the SSO user name, that certificate must be added to the user's entry in the directory as a user attribute.

Benefits of this approach are that applications which work within Oracle9*i*AS Single Sign-On framework are PKI-enabled automatically when Oracle9*i*AS Single

Sign-On is PKI enabled. Oracle Internet Directory assumes responsibility for name mapping. Moreover, because getting and checking a cookie requires fewer system resources than performing an SSL exchange, using PKI for initial authentication to the SSO framework and cookies for authentication to partner applications has better performance than a PKI-only authentication approach. For Web applications which are characterized by many short-lived sessions, this leads to significant improvement in server performance and throughput.

Finally, enabling Oracle9*i*AS Single Sign-On for PKI allows users to authenticate to Oracle Applications using PKI. In this way, Oracle Applications can participate in the SSO framework as partner applications.

Multitier Integration

Oracle9*i*AS Single Sign-On provides SSO for Web client access to Web servers. Web servers are increasingly deployed as the middle tier in a multitier architecture, where they provide access to a back-end tier database. Users of Web applications that require access to the database should not have to supply a database user name and password for access to data stored there. Although Oracle9*i*AS Single Sign-On does not support non-Web based applications, the Oracle database includes features specifically designed to support secure access to databases through multitier architectures.

Oracle9*i*AS provides a number of mechanisms for accessing the database and invoking applications on it. The most commonly used of these fat-client JDBC and mod_plsql, a plug-in to Oracle HTTP Server that allows Oracle9*i*AS to invoke database applications written in the Oracle database programming language, PL/SQL. Because fat-client JDBC and mod_plsql access an Oracle database with the Oracle client-server networking protocol, Oracle Net, developers who use fat-client JDBC or mod_plsql can use Oracle Advanced Security, an option of Oracle9*i* Database Server. Oracle Advanced Security, which provides encryption, integrity protection, and advanced authentication, can be used to protect data exchanged between Oracle9*i*AS and an Oracle database.

See Also:

- "Securing Application Database Access Through mod_plsql" on page 9-7
- *mod_plsql User's Guide* in the Oracle9*i*AS Documentation Library
- *Oracle Advanced Security Administrator's Guide* in the Oracle Database Documentation Library

Authorization, Authentication, and SSL in Oracle9*i*AS

Each major component of the application server provides a set of basic security services, which authorize and authenticate users with or without SSL, that can be used individually or in conjunction with other components. This section describes the basic security services that are available in the following Oracle9*i*AS components:

- Oracle HTTP Server Security
- Oracle Wallet Manager Overview
- Oracle9iAS Portal Security Overview
- JAAS Security
- Oracle9iAS Web Cache Security

Oracle HTTP Server Security

Oracle HTTP Server is the Web server component of Oracle9*i*AS. It is based on the Apache HTTP Server. The Apache open source Web server is among the most widely-adopted Web server products; it supports a rich set of existing applications, and provides a flexible and well-understood security model. Apache is a very well-tested platform on which to deploy secure applications. Customers familiar with Apache should find it easy to build and deploy secure Web applications using Oracle HTTP Server.

This section describes the security features of Oracle HTTP Server in the following topics:

- Oracle HTTP Server Security Services Overview
- Access Control, User Authentication, and Authorization with Oracle HTTP Server
- User Authentication and Authorization
- Secure Sockets Layer (SSL) and PKI with Oracle HTTP Server
- Secure Access to an Oracle Database with Oracle HTTP Server

Oracle HTTP Server Security Services Overview

Oracle HTTP Server extends Apache with a variety of standard enhancements called "mods," which is a shortened form of module. It also includes mods that have been developed by Oracle Corporation. It allows users with Web browsers to access

Oracle9*i*AS using standard Web protocols. It provides an HTTP listener, which supports HTTP and secure HTTP, or HTTPS, and serves up information to users in standard HTML format. It provides access to both static Web pages and dynamic content.

Oracle HTTP Server security services include the ability to restrict or allow access to files and services based on the identity of users established by means of basic authentication, by client- supplied X.509 certificates, and by IP or hostname addresses.

Another important feature of Oracle HTTP Server security is protection of data exchanged between clients and the server. This is provided by means of the SSL protocol, which also provides data integrity and strong authentication of both users and HTTP servers.

In addition, Oracle HTTP Server provides logging and other facilities needed to detect and resolve intrusion attempts. It provides integration with the other Oracle9*i* Application Server components, such as mod_osso, which enables the HTTP server to receive and route requests for single sign-on services to Oracle9*i*AS Single Sign-On server. Oracle HTTP Server is also well integrated with other Oracle products such as Oracle applications and the database. In this way, the Oracle HTTP Server provides a comprehensive set of security services for building and deploying Web applications.

See Also:

- Chapter 4, "Configuring HTTP Server Security" for information about configuring the HTTP server for SSL
- Chapter 3, "Configuring Oracle9iAS Single Sign-On" for information about mod_osso and single sign-on services
- Chapter 9, "Configuring Secure Database Access by Oracle9i Application Server" for information about secure communication between the HTTP server and an Oracle database
- *Oracle HTTP Server Administration Guide* in the Oracle9*i*AS Documentation Library for detailed information about configuring and using the HTTP server

Access Control, User Authentication, and Authorization with Oracle HTTP Server

When URL requests arrive at Oracle HTTP Server they are processed in a number of steps which are implemented by means of a module or plug-in architecture common to many Web servers.

Access Control Oracle HTTP Server access control is based on Apache access control mechanisms which allow the server administrator to restrict access to particular files, directories, or URLs on the server. For each restricted object on the server, the administrator can use a directive in the main configuration file, `httpd.conf`, to specify that access to the object is denied or allowed, based on the value of one or more attributes associated with the requester. The administrator can configure directives such as `deny`, `allow`, and `order` to inhibit further processing, based on user attributes such as hostname, IP address, or browser type. Restrictions can be applied to particular files, directories, or URL formats.

Note: Although the Oracle HTTP Server is based on the open source Apache server, it contains some access control enhancements which improve security. For example, the Apache server provides for access restrictions per directory or folder by means of files that have the suffix `.htaccess`. The processing of these files is disabled in Oracle HTTP Server, because `.htaccess` processing can produce security problems and degrade performance.

User Authentication and Authorization In many applications it is desirable to control access to resources on the Web server based on user identity. Oracle HTTP Server provides several mechanisms for user authentication, including client authentication over the Single Sockets Layer (SSL) using X.509 certificates, user name/password pairs (as in basic authentication), and other forms. A server administrator can use a configuration directive in the `httpd.conf` file to specify that access to certain URLs is restricted to particular users.

Secure Sockets Layer (SSL) and PKI with Oracle HTTP Server

The Secure Sockets Layer provides point-to-point security between Oracle HTTP Server and client browsers. Security-related services provided by SSL include authentication, authorization, confidentiality, and data integrity. The HTTP server supports SSL with a module developed by Oracle called mod_ossl. If you also need to implement PKI, Oracle Wallet Manager can be used to request, store, and manage certificates. These features are described in the following sections.

mod_ossl This Oracle HTTP Server module is a plug-in to the HTTP server that enables the server to use SSL. It is very similar to the OpenSSL module, mod_ssl. However, in contrast to the OpenSSL module, mod_ossl is based on the Oracle implementation of SSL, which supports SSL, version 3 and is based on Certicom and RSA Security technology.

See also:

- "Using Secure Sockets Layer (SSL) to Authenticate Users" on page 4-12 for more information about mod_ossl and how to use it.
- "Oracle Wallet Manager Overview" on page 2-32 for information about using certificates in Oracle9*i*AS.

Secure Access to an Oracle Database with Oracle HTTP Server

Oracle9*i*AS makes it easy to build multitier systems using an Oracle database back-end repository. Oracle9*i*AS provides a number of mechanisms for accessing the database and invoking applications on the database. The most commonly used of these are JDBC and mod_plsql, a plug-in to the Oracle HTTP Server that allows the application server to invoke database applications written in the Oracle database programming language, PL/SQL. Because mod_plsql accesses the Oracle database using the Oracle client-server networking protocol, developers using mod_plsql can take advantage of Oracle Advanced Security to protect data exchanged between Oracle9*i*AS and an Oracle database. Oracle Advanced Security provides encryption, integrity protection, and advanced authentication services to Oracle database clients and servers. It supports industry standard encryption protocols such as SSL, and standard encryption algorithms including RSA RC4, DES, and 3DES.

It is also possible to use Oracle9*i* Database Server proxy authentication, which is designed to address a performance problem associated with three-tier application design. Specifically, it allows Oracle9*i*AS to access an Oracle9*i* database and get specific database user privileges without having to log out and log in again each time that the application server switches user contexts. Proxy authentication allows the application server to establish a single authenticated session using fat-client JDBC or mod_plsql with an Oracle9i database server, and act on behalf of multiple database users without having to submit separate authentication credentials for each user within the session. In addition, the database can use both the authentication identity of the application server and the identity of the user on whose behalf the application server performs proxy authentication. Thus, proxy authentication allows the database to delegate limited trust to a middle-tier application server without having to grant it superuser privilege on the database, or store multiple database user passwords in the application server. Proxy authentication is available with Oracle9*i* Database Server.

See Also:

- "Securing Application Database Access Through mod_plsql" on page 9-7
- *mod_plsql User's Guide* in the Oracle9*i*AS Documentation Library
- *Oracle Advanced Security Administrator's Guide* in the Oracle Database Documentation Library
- *Oracle9i Database Administrator's Guide* in the Oracle Database Documentation Library for information about proxy authentication.
- *Oracle9i Application Developer's Guide - Fundamentals* in the Oracle Database Documentation Library for information about how to design a middle-tier server to proxy users.

Oracle Wallet Manager Overview

This is a Java-based application that security administrators use to manage public-key security credentials on both Oracle clients and servers. Oracle Wallet Manager creates a public-private key pair and manages credentials for a user. It issues PKCS#10 certificate requests to the certificate authority, and installs the certificate in the wallet. It ships with trusted certificates from VeriSign, RSA, and Baltimore CyberTrust, and can use a site's own in-house certificate authority.

See Also: Chapter 5, "Using Oracle Wallet Manager" for information about Oracle Wallet Manager and how to use it.

Oracle9*i*AS Portal Security Overview

This section describes several aspects of Oracle9*i*AS Portal security:

- Introduction to Oracle9iAS Portal
- Portal Users
- Portal Groups
- Portal Authentication
- Portal Authorization
- Application Integration with Oracle9iAS Portal
- Oracle9iAS Portal Support for HTTPS
- Auditing in Oracle9iAS Portal

See Also:

- Chapter 6, "Configuring Oracle9iAS Portal Security"
- *Oracle9iAS Portal Configuration Guide* in the Oracle9*iAS* Documentation Library

Introduction to Oracle9*i*AS Portal

Oracle9*i*AS Portal is an "enterprise portal" program, which provides a gateway to business information on corporate intranets. Although it is targeted at the corporate portal market, Oracle9*i*AS Portal can be scaled to provide access to much larger Internet communities.

Oracle9*i*AS Portal allows Oracle9*i*AS customers to organize their Web content and applications, and provide this to users in a logical, consistent Web portal format. It also provides a set of tools for creating and managing users and their access to Oracle9*i*AS Portal content. Enterprise portals, as both a consolidation and extension of existing market spaces, can utilize three powerful components:

- The strong information management technology inherent in the Oracle database
- A wide range of Oracle applications and applications from other vendors which manage critical business data, including enterprise resource planning (ERP), customer relationship management (CRM), and business intelligence (BI) offerings

- A portal framework which leverages this technology to bring the applications together with other datastores on an intranet.

This section provides an overview of the security features and architecture of Oracle9*i*AS Portal. Oracle9*i*AS Portal provides a comprehensive and extensible authorization model, based on user privileges and groups, for user access to content and applications on the portal. It also supports auditing of security events through its event logging service.

Portal Users

In the Internet computing model, where millions of users may potentially be accessing a portal, it is important to keep the representation of users as lightweight as possible. To manage large numbers of users and large amounts of data in a secure, scalable, and fault-tolerant way, Oracle9*i*AS Portal leverages the security and data management technology of the Oracle database.

Oracle9*i*AS Portal defines its own user accounts, which are referred to as "lightweight" because they do not each have a unique database schema associated with them in the Oracle database. By contrast, each Oracle9*i*AS Portal user account does correspond uniquely to an Oracle9*i*AS Single Sign-On user account. Oracle9*i*AS Portal provides a convenient mechanism to manage users of applications configured to use Oracle9*i*AS Single Sign-On.

Installed Users When Oracle9*i*AS Portal is installed, a default set of user accounts is created. These include accounts for the portal administrator and a public account. The public account allows certain portal content to be publicly accessible; that is, it allows the content to be accessed by users who do not have their own Oracle9*i*AS Portal user accounts, or by users who do have accounts but who have not yet logged in.

> **See Also:** Table 6–10, "Default Oracle9iAS Portal Users" on page 6-30 for a list of the users that are created by default when Oracle9*i*AS Portal is installed.

Portal Groups

Oracle9*i*AS Portal supports groups, which are used for two main purposes. Groups provide a convenient means of granting privileges to a collection of users in one action. In addition, certain attributes in the Portal system can be associated with a group, and if a user has a default group specified in his preferences, then those attributes can be applied to his session. Examples of these attributes include a default home page for a group, or a default style. Because a user can belong to multiple groups, he must specify a default group in his own profile so he can inherit group preferences. He should not specify any personal preferences that would override the group preferences.

Groups can be comprised of users, as well as other groups. This allows for construction of hierarchical groups. A user belonging to a subgroup of another group is a member of the parent group, by virtue of group membership. Privileges granted to the parent group are thus assumed by the subgroup user.

> **See Also:** Table 6–11, "Default Oracle9iAS Portal Groups" on page 6-31 for a list of the groups that are created by default when Oracle9*i*AS Portal is installed.

Portal Authentication

Oracle9*i*AS Portal is implemented as an Oracle9iAS SSO partner application for the purpose of user authentication. Note that users who are not authenticated may be allowed access to certain content on Oracle9*i*AS Portal by means of the public user account.

Portal Authorization

Authorization is the process of controlling access to various areas of Oracle9*i*AS Portal, based on the identity of the user. Once the user is identified, after going through the process of authentication, Oracle9*i*AS Portal determines the appropriate authorization of the user based on his identity.

Oracle9*i*AS Portal allows portal objects to be defined as public or private, depending on whether they are accessible to general Web users or only to trusted portal administrators. Objects which are public can still implement access control logic that is based on identity, in which case they must first obtain the user's identity from Oracle9*i*AS Single Sign-On. Once this has been done, the user's privilege to access the object can be determined. Oracle9*i*AS Portal provides an extensible set of privileges which are used to define the access control lists for each object in the portal.

Application Integration with Oracle9*i*AS Portal

One of the key features of Oracle9*i*AS Portal is the ability to integrate various applications into its framework to provide a seamless experience for those using it to access their business applications. Oracle9*i*AS Portal depends on Oracle9*i*AS Single Sign-On for this purpose, and supports varying degrees of integration between portal security and application security.

The tightest integration occurs when an application is implemented on Oracle9*i*AS Portal itself and obtains user identity directly from Oracle9*i*AS Portal. Such applications are called portal applications. Oracle9*i*AS Portal is a partner application for Oracle9*i*AS Single Sign-On, allowing it to obtain a user's identity from the Single Sign-On server. Portal applications obtain a user's identity directly from the once the user has authenticated by means of Oracle9*i*AS Single Sign-On.

Other applications accessible through Oracle9*i*AS Portal may be Oracle9*i*AS Single Sign-On partner applications or external applications, as described in the section on Oracle9*i*AS Single Sign-On. Partner applications need not be implemented on the portal itself, but do participate in the Oracle9*i*AS Single Sign-On authentication framework which Oracle9*i*AS Portal also uses. External applications maintain their own authentication mechanisms, and do not share directly in the authentication framework provided by Oracle9*i*AS Single Sign-On.

Oracle9*i*AS Portal Support for HTTPS

For increased security, Oracle9*i*AS Portal can be run in HTTPS mode over SSL. However, running Oracle9*i*AS Portal in HTTPS mode can significantly degrade this program's performance.

Auditing in Oracle9*i*AS Portal

Besides monitoring activity which could indicate unauthorized system use, auditing of security related events is often the most effective means to ensure that authorized users adhere to system usage policies, and thus "keep honest users honest."

Oracle9*i*AS Portal has a logging service which logs certain security events and which can be invoked to log arbitrary events defined by portal applications. The reporting features of the Oracle9*i*AS Portal application building capabilities can then be leveraged to view the logged data. Event logging can be used to audit security events, to detect possible attempts to undermine system security, or to use the system in a way that is contrary to security policy.

Oracle9*i*AS Portal runs on the Oracle9i database. Many Oracle9*i*AS Portal events generate database events that can be audited.

> **See Also:** Chapter 6, "Configuring Oracle9iAS Portal Security" for more information about portal authorization, authentication, and application integration.

JAAS Security

This section introduces the primary security services provided by JAAS:

- Authentication Features of JAAS
- Authorization Features of JAAS
- Delegation Features of JAAS

See Also:

- Chapter 7, "Configuring JAAS Support"
- *Oracle9iAS Containers for J2EE Services Guide* in the Oracle9*i*AS Documentation Library
- *JAAS Provider API Reference* in the Oracle9*i*AS Documentation Library

Authentication Features of JAAS

JAAS includes two major features for authentication:

- Security servlets that integrate Oracle9*i*AS Single Sign-On with Oracle9*i*AS Containers for J2EE
- LoginModules that enable customers to develop strong authentication for Java applications

One of the biggest benefits of using JAAS is the ability to integrate applications with Oracle9*i*AS Single Sign-On. Most Web applications need single sign-on to enhance the user experience whether or not they are developed in Java.

This integration enables any Java application to participate in Web single sign-on, and increases the ability of Oracle9*i*AS Single Sign-On to support applications running within Oracle9*i*AS, whether they are designed to use Oracle HTTP Server, Oracle9*i*AS Portal, or Oracle9*i*AS Containers for J2EE.

LoginModules provide extensible authentication for Java applications. Developers can create custom authentication modules as required by their application and organization. For example, a Java banking application might require stronger authentication (such as a challenge-response mechanism) than a Web site offering only static content. A developer can create a `LoginModule` that incorporates a challenge-response mechanism which would integrate the application with other JAAS services.

Authorization Features of JAAS

Once users have been authenticated, it is important to enforce the principle of least privilege—that is, that users should have the fewest privileges necessary to perform their jobs. JAAS includes the following authorization features:

- The ability to centrally manage authorizations in Oracle Internet Directory
- The ability to manage authorizations in a flat file in XML format
- The ability to partition security policy by subscriber
- Support for hierarchical, role-based access control
- Support for both user-based and code-based policies

The ability to manage and retrieve authorizations in Oracle Internet Directory allows Java applications to leverage the centralized user provisioning capabilities of Oracle Internet Directory. This lowers the cost of application deployment, because Java applications do not have to create their own user repository, but can use existing user and authorization definitions in Oracle Internet Directory. Java applications can retrieve roles for users from Oracle Internet Directory. Optionally, roles can be hierarchical, which means that a role can be assigned to other roles.

The ability to partition security policy by subscriber supports application hosting. For example, each realm (subscriber) can have its own private, partitioned policy, administered by its own administrator. Then each organization subscribing to a hosted application can create its own users and provide its own authorizations to users. In this example, subscribers cannot create users for other subscribers.

User- and code-based policies allow applications to require that a user only assume particular privileges when accessing particular Java applications from particular locations. This enables Java developers to finely tune their security policies. For example, a banking application may give a user access rights to update bank account information. Rather than give every user authorization to update bank account information—which each user does not need—the application need only enforce that users can only update banking information when accessing a particular digitally signed Java banking application from the banking application directory.

For example, the following code specifies that `DBPrincipal SCOTT` is granted `AllPermission` only when `SCOTT` is running application code originating from `http://www.oracle.com/HRApp`, that is signed by `LARRY`.

```
grant Principal oracle.security.DBPrincipal "SCOTT"
               Codebase "http://www.oracle.com/HRApp" signed "LARRY" {
           permission java.security.AllPermission;
       }
```

Delegation Features of JAAS

Delegation features support impersonation of a specified user (both `RunAsClient` and `RunAsID`) within enterprise JavaBeans, JSPs, and servlets.

`RunAsClient` means an enterprise bean can be configured to run with the permissions associated with the current client. `RunAsID` means an enterprise bean can be configured to run with the permissions associated with the specified user (for example, run as "DBAdmin"). This allows developers to enforce "least privilege" in their applications, allowing users only those privileges needed to perform a function. In this way, users can only exercise privileges associated with a well-formed business transaction (an enterprise bean).

Oracle9*i*AS Web Cache Security

Oracle9*i*AS Web Cache is an innovative content delivery solution designed to accelerate dynamic Web-based applications and reduce hardware costs. Deployed before a farm of Oracle HTTP Servers or globally at the network edge, Oracle9*i*AS Web Cache uses caching, compression, and assembly technologies to speed the delivery of static and dynamic Web pages. Oracle9*i*AS Web Cache also provides surge protection, load balancing and failover for Web servers. Combined, these features ensure application performance, scalability and availability, and help to lower the cost of conducting business online.

Using Oracle9*i*AS Web Cache, e-businesses benefit simultaneously from improved response times and lower infrastructure costs. With Oracle9iAS Web Cache, e-businesses can now serve rich content faster, to more customers, using fewer computing resources than ever before.

As part of the security architecture of Oracle9*i*AS, Oracle9*i*AS Web Cache can be configured to accept HTTPS requests from clients and forward them to Oracle HTTP Server.

See Also:

- Chapter 8, "Configuring Security for Oracle9iAS Web Cache"
- *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i*AS Documentation Library

How to Proceed from Here

The following chapters in this document contain conceptual and configuration information about the security features of primary Oracle9*i*AS elements. If you have installed the necessary software and need to configure the security features of the application server, please refer to the remaining chapters in this document.

If you need information about administering the application server, then please refer to the Oracle9*i*AS Documentation Library for the appropriate administrator’s guide or development guide.

See Also: *Oracle9i Application Server Release Notes* in the Oracle9*i* Application Server Platform-specific Documentation for any issues that may pertain to your security implementation which are not included here.

3 Configuring Oracle9*i*AS Single Sign-On

Oracle9*i*AS Single Sign-On is a component of Oracle9*i* Application Server that enables users to log in to multiple Web-based applications, such as expense reports, e-mail, and benefits information, using a single user name and password. As such, Oracle9*i*AS Single Sign-On serves as the security gateway for all Oracle9*i*AS features.

This chapter explains how to configure security features for Oracle9*i*AS Single Sign-On. It also provides a conceptual overview of the product.

The chapter covers the following topics:

Single Sign-On Overview

This section takes a quick look at the salient features of Oracle9*i*AS Single Sign-On. Each component is described in the following sections.

> **See Also:** For more detailed information about Oracle9*i*AS Single Sign-On, refer to the *Oracle9iAS Single Sign-On Administrator's Guide* in the Oracle9*i*AS Documentation Library.

- Oracle9*i*AS Single Sign-On Server (Single Sign-On server)

  The server consists of program logic in the Oracle9*i*AS database that enables users to log in securely to single sign-on enabled applications.

- Single Sign-On Applications

  Single sign-on applications take two forms: partner and external. Partner applications are those that delegate authentication to the Single Sign-On server. Examples of such applications are Oracle9*i*AS Portal, Oracle9*i*AS Discoverer and Oracle9*i*AS Reports Services. External applications require reauthentication, but the Single Sign-On server performs this function for the user. Examples include Yahoo! Mail and Oracle Mobile.

  Partner applications use either an Oracle HTTP Server authentication module called mod_osso or the Oracle9*i*AS Single Sign-On Software Development Kit (SDK) to redirect first-time user requests to the Single Sign-On server and to validate additional application requests once a user is logged in. Applications integrated with mod_osso are actually not partner applications at all, because only mod_osso is registered with the Single Sign-On server. SDK-integrated applications, on the other hand, must be registered individually.

- Mod_osso

  The Oracle HTTP Server authentication module mod_osso is an alternative to the Single Sign-On SDK, used in earlier releases of Oracle9*i*AS Single Sign-On to integrate partner applications. Mod_osso simplifies the authentication process by serving as the sole partner application to the Single Sign-On server, rendering authentication transparent for Oracle9*i*AS applications.

- Single Sign-On SDK

  The Single Sign-On Software Developer's Kit consists of application programming interfaces for PL/SQL and Java. It also contains sample code that explains how to implement these interfaces. The SDK ships separately with

Oracle9*i*AS. As such, its APIs must be incorporated into partner applications. Mod_osso spares application developers the burden of integrating these APIs.

Configuring Security Features for Oracle9*i*AS Single Sign-On

Oracle9*i*AS Single Sign-On is installed automatically as part of an Oracle9*i*AS infrastructure installation. At install time, users are offered the option of registering mod_osso with the Single Sign-On server. They may elect to use the SDK instead.

Beyond simple authentication, Oracle9*i*AS Single Sign-On has a few security features that must be configured after installation if an enterprise elects to use them. These features are as follows:

- Secure Sockets Layer (SSL)
- Digital Certificates
- Global User Inactivity Timeout
- IP Checking

Password policies, another Oracle9*i*AS Single Sign-On security feature, are managed in Oracle Internet Directory.

The rest of this chapter describes how to configure the security features just enumerated.

See Also:

- For a more detailed treatment of this topic, see *Oracle9iAS Single Sign-On Administrator's Guide* in the Oracle9*i*AS Documentation Library.
- If you specifically require SSL connections between the Single Sign-On server and Oracle Internet Directory, refer to Chapter 3 in the *Oracle9iAS Single Sign-On Administrator's Guide*.

Enabling the Single Sign-On Server for SSL

The Single Sign-On server can be enabled for Secure Sockets Layer (SSL) at install time. If the administrator does not select this option, SSL must be configured manually.

To configure the Single Sign-On server for SSL:

1. Configure the Oracle HTTP server to use SSL.

   See Also: "Using Secure Sockets Layer (SSL) to Authenticate Users" on page 4-12 for information about configuring Oracle HTTP Server to use SSL.

2. Change all references of HTTP in Single Sign-On URLs to HTTPS. The script `ssocfg.sh` is provided for this purpose. It can be found at the following location:

   ```
   IAS_HOME/sso/bin
   ```

   Enter the command, using the following syntax:

   ```
   ssocfg.sh protocol host port [sso_schema_name]
   ```

   In this case, *protocol* is `https`. (To change back to HTTP, use `http`.) The parameter *new_host* is the host name of the Web listener for the Single Sign-On server. You can either assign a new host name or use an existing one. The parameter *new_port* is the port number of the listener, and *sso_schema_name* is the name of the Single Sign-On schema. The default schema name is orasso. Note from the syntax that this last parameter is optional.

   Here is an example:

   ```
   ssocfg.sh https login.acme.com 443
   ```

   Port 443 is the default port number for Single Sign-On over SSL.

3. Protect Single Sign-On URLs to use SSL. In the dads.conf file, use the following HTTP directive to protect the Single Sign-On DAD with SSL.

   ```
   <IfDefine SSL>
      <Location /pls/orasso>
         SSLRequireSSL
      </Location>
   </IfDefine>
   ```

   The dads.conf file can be found at the following location:

```
IAS_HOME/Apache/modplsql/conf/dads.conf
```

Configuring Oracle9*i*AS Single Sign-On for Digital Certificates

Oracle9*i*AS Single Sign-On users have the option of using digital certificates instead of the SSO user name and password to authenticate. This form of authentication involves an exchange of X.509 certificates between client and server over Secure Sockets Layer (SSL).

Oracle9*i*AS Single Sign-On can be configured for SSL both with and without client certificates. The first option, server-side authentication, offers a strong degree of security. Still, the user's password is vulnerable to attack—either by guesswork or by brute force. Certificate-based authentication on both client and server sides, on the other hand, makes it difficult to sniff or modify data or to impersonate the client or server.

This section covers the following topics:

- System Requirements
- Configuration Tasks

System Requirements

The following criteria must be met before certificate-enabled single sign-on can proceed:

- The Single Sign-On server and Oracle Internet Directory, Oracle9*i*AS, Release 2, must be installed.
- The Oracle HTTP Server must have a valid server certificate installed.
- The distinguished name (DN) of the client certificate must be chosen in such a way that it contains some information about the SSO user name, or nickname, and, optionally, the subscriber name for this user name. The `cn` attribute within the DN, for instance, might be the user's nickname (`cn=jsmith`). An `o` attribute might be the user's subscriber name (`o=acme`).
- The certificate of the client certificate issuer must be installed as a trusted certificate in the Single Sign-On server.
- The certificate of the server certificate issuer must be installed as a trusted certificate in the user's browser.

Configuration Tasks

Certificate-enabled single sign-on is not a default option in Oracle9*i*AS, and it must be configured manually. The follow components may require configuration:

- Oracle HTTP Server (SSL)
- Single Sign-On DAD (mod_plsql)
- User Name Mapping Module
- Oracle Internet Directory
- Single Sign-On Server

Oracle HTTP Server (SSL)

To configure the Oracle HTTP server, navigate to the server configuration file, using the following path:

```
IAS_HOME/Apache/Apache/conf/httpd.conf
```

In the SSL Virtual Host Context section of the `httpd.conf` file, add the parameters listed in Table 3–1:

Table 3–1 Oracle HTTP Server Parameters for Certificate-Enabled Single Sign-On

Parameter	**Description**
`ServerName`	The name of the server to be enabled for SSL
`SSLEngine [on	off]`	Setting the `SSLEngine` parameter to `on` enables the server for SSL
`SSLWallet file:`	The location, or path, of the server wallet
`SSLVerifyClient`	The verification type for client certificates. The options are as follows: ▪ `none`—SSL without certificates ▪ `optional`—server certificate only ▪ `require`—server and client certificates

When configured properly, the SSL Virtual Host Context section of the httpd.conf file looks similar to the example that follows.

```
## SSL Virtual Host Context
##
#
# file otherwise your virtual host will not respond to SSL requests.
#
<VirtualHost _default_:443>
#  General setup for the virtual host
DocumentRoot "/private/oracle/work/Apache/Apache/htdocs"
ServerName db_host:db_port:db_sid
ServerAdmin you@your.address
ErrorLog /pivate/oracle/work/Apache/Apache/logs/error_log
TransferLog /private/oracle/work/Apache/Apache/logs/access_log

#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   Server Wallet:
#   The server wallet contains the server’s certificate, private key
#   and trusted certificates. Set SSLWallet at the wallet directory
#   using the syntax:  file:<path-to-wallet-directory>

SSLWallet file:/private/iAS/wallet

#   Certificate Revocation Lists (CRL):
#   Set the CA revocation path where to find CA CRLs for client
#   authentication or alternatively one huge file containing all
#   of them (file must be PEM encoded)
#   Note: Inside SSLCARevocationPath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCARevocationPath /private/oracle/Apache/Apache/conf/ssl.crl
#SSLCARevocationFile /private/oracle/Apache/Apache/conf/ssl.crl/ca-
 bundle.crl
#   Client Authentication (Type):
#   Client certificate verification type and depth.  Types are
#   none, optional, require and optional_no_ca.  Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.
SSLVerifyClient optional

</VirtualHost>
```

Single Sign-On DAD (mod_plsql)

Configuring the Oracle HTTP Server PL/SQL module for certificates entails adding environment variables to the database access descriptor (DAD) for the Single Sign-On server. To add these variables, navigate to the DAD configuration file, using the following path:

```
IAS_HOME/Apache/modplsql/conf/dads.conf
```

In the `dads.conf` file, add the `PlsqlCGIEnvironmentList` parameter and the variables in Table 3–2.

Table 3–2 dads.conf Environment Variables

Variable	Description
SSL_CLIENT_S_DN	The distinguished name of the user
SSL_CLIENT_CERT	The client certificate in base 64 format

mod_plsql must pass these variables to the user name mapping module.

When configured properly, the relevant section of the dads.conf file looks something like this:

```
<IfModule mod_plsql.c>
<Location /pls/orasso>
  SetHandler pls_handler
  Order deny,allow
  PlsqlDatabaseConnectString    db_host:db_port:db_sid
  PlsqlDatabasePassword         password
  PlsqlDatabaseUsername         orasso
  PlsqlDefaultPage              orasso.home
  PlsqlDocumentTablename        orasso.wwdoc_document
  PlsqlDocumentPath             docs
  PlsqlDocumentProcedure        orasso.wwdoc_process.process_download
  PlsqlEnableConnectionPooling  On
  PlsqlAuthenticationMode       SingleSignOn
  PlsqlPathAlias                url
  PlsqlPathAliasProcedure       orasso.wwpth_api_alias.process_download
  PlsqlSessionCookieName        orasso
  PlsqlCGIEnvironmentList       SSL_CLIENT_S_DN, SSL_CLIENT_CERT
</Location>

<IfDefine SSL>
<Location /pls>
  SSLOptions +ExportCertData +StdEnvVars
</Location>
</IfDefine>
```

User Name Mapping Module

The module that maps a user DN to a user name is actually the package `ssodnmap.pks`, which is located in the following directory:

```
IAS_HOME/sso/admin/plsql/sso
```

If the user accepts the default implementation for the package, no file configuration is required. The default implementation assumes that the user’s DN in the directory is the same as the certificate DN.

> **See Also:** Those who want to customize the module can view the file `ssodnmap.pks` in Chapter 4 of *Oracle9iAS Single Sign-On Administrator’s Guide*, "Configuring Single Sign-On for Certificates." This document is located in the Oracle9*i*AS Documentation Library.

Oracle Internet Directory

For certificate-based authentication to be successful, the user certificate must be present in Oracle Internet Directory. If the certificate is issued by an in-house certificate authority (CA) or by Oracle's CA, it might be possible to publish the certificate in the directory automatically. If the certificate issuer is a third-party CA, a self-service application can fulfill this function.

> **See Also:** To determine whether a self-service application is feasible, see the procedures in "Configuring Single Sign-On for Certificates" in Chapter 4 of *Oracle9iAS Single Sign-On Administrator's Guide* in the Oracle9*i*AS Documentation Library.

Single Sign-On Server

To enable the Single Sign-On server for SSL, all references to HTTP in SSO URLs must be changed to HTTPS. The script `ssocfg.sh` is provided for this purpose.

To run `ssocfg.sh`:

1. Go to the directory that contains the script. The path is as follows:

   ```
   IAS_HOME/sso/bin
   ```

2. Enter the command, using the following syntax:

   ```
   ssocfg.sh protocol host port [sso_schema_name]
   ```

   In this case, *protocol* is `https`. (To change back to HTTP, use `http`.) The parameter *new_host* is the host name of the Oracle HTTP Server listener for the Single Sign-On server. You can either assign a new host name or use an existing one. The parameter *new_port* is the port number of the listener, and *sso_schema_name* is the name of the SSO schema. The default schema name is `orasso`. This last parameter is optional.

   Here is an example:

   ```
   ssocfg.sh https login.acme.com 443
   ```

   Port 443 is the default port number for single sign-on over SSL.

Enabling Timeouts

Oracle9*i*AS Single Sign-On has two timeout features: the single sign-on session timeout and the global user inactivity timeout. The first can be configured through the SSO user interface. The second must be configured from the command line.

This section covers the following topics:

- Configuring the SSO Session Timeout
- Configuring the Global User Inactivity Timeout

Configuring the SSO Session Timeout

By default, an SSO session lasts eight hours. The administrator can specify a shorter or longer period on the Edit SSO Server page.

To change the SSO session duration:

1. Enter a URL of the following form:

   `http://`*`host`*`:`*`port`*`/pls/`*`Single_Sign_On_DAD`*

   where *host* is the name of computer on which the Single Sign-On server is located, *port* is the port number of the server, and *Single_Sign_On_DAD* is the database access descriptor for the SSO schema. The default DAD is `orasso`.

   The Access Partner Applications page appears.

2. Select Login in the upper right corner of the Access Partner Applications page.

   The Single Sign-On Login page appears.

3. Enter your administrative user name and password, and then click the Login button.

4. The Single Sign-On home page appears.

5. From the Single Sign-On home page navigate, in succession, to the following pages:

   —SSO Server Administration

   —Edit SSO Server Configuration

   —Edit SSO Server

6. On the Edit SSO Server page, under the heading SSO Session Policy, enter the number of hours that a user can be logged in without timing out.

Configuring the Global User Inactivity Timeout

The Single Sign-On server uses the Web cookie `SSO_TIMEOUT_ID` to track user inactivity across mod_osso-protected applications and to enable these applications to force users to reauthenticate if they have been idle for a preconfigured amount of time. The global user inactivity timeout is a useful feature for sensitive applications that may require a much shorter user inactivity timeout than the SSO session timeout.

The global user inactivity timeout is not configured by default. You must enable it by running the script `ssogito.sql` and by modifying the file `mod_osso.conf`.

To configure the global user inactivity timeout:

1. Log in to SQL*Plus using the SSO schema name and password. The default is `orasso/orasso`.
2. Run the script `ssogito.sql` by entering the following command:

   ```
   SQL> @ssogito.sql
   ```

3. Running the script brings up a list of fields.
4. In the field "Enter value for timeout_cookie_domain:" enter a domain name that is common to all of the applications enabled by the Single Sign-On server.

   Note: If this field is left blank, the domain name defaults to the host name for the Single Sign-On server.

5. In the field "Enter value for inactivity period:" enter the length of the desired inactivity period. For example, you can enter 15 minutes.
6. To enable the new settings, select the return key. To cancel the transaction, select the return key twice.
7. Once you have completed a transaction, the script furnishes you a summary of the new timeout settings.
8. In the file `mod_osso.conf`, make sure that the parameter `ossoIdleTimeout` exists and that it is set to `on`.

Enabling IP Checking

IP checking can be enabled for both the Single Sign-On server and for mod_osso. On the Single Sign-On server, IP checking is activated on the Edit SSO Server page of the SSO user interface. An IP check verifies that the IP address of the browser is the same as the IP address of the authentication request. On the mod_osso side, the directive `OssoIPCheck` verifies that the user who authenticates to the Single Sign-On server is the same user who is accessing a mod_osso-protected application.

This section covers the following topics:

- Enabling IP Checking for the Oracle9iAS Single Sign-On Server
- Enabling IP Checking for Mod_osso

Note: IP checking should be enabled only if a browser can communicate with both the Single Sign-On server and the HTTP server without using proxy servers. This is because proxies may not use static IP addresses.

Enabling IP Checking for the Oracle9*i*AS Single Sign-On Server

To enable IP checking, use the following steps:

1. Navigate to the Edit SSO Server page using the procedures presented in "Configuring the SSO Session Timeout" on page 3-11.
2. On the Edit SSO Server page, under the heading SSO Session Policy, select the check box Verify IP Addresses for Requests Made to the Single Sign-On Server.

Enabling IP Checking for Mod_osso

To enable mod_osso for IP checks, the directive `OssoIPCheck` in the `mod_osso.conf` file must be set to `on`. If `OssoIPCheck` is enabled when proxy servers are used, then an error message might be displayed.

Managing Password Policies

The SSO user password is stored in Oracle Internet Directory as an attribute of the user’s entry. Users can change their passwords either in the SSO user interface or through Delegated Administration Service (DAS). Oracle Directory Manager, a GUI tool, enables the directory administrator to adjust password rules, password expiry, and account lockout to suit enterprise needs.

See Also: To learn how to configure password policies, see "Managing Password Policies by Using Oracle Directory Manager" and "Setting Password Policies by Using Command-Line Tools." Both topics can be found in Chapter 17 of *Oracle Internet Directory Administrator’s Guide* in the Oracle9*i*AS Documentation Library.

4 Configuring HTTP Server Security

As described in the introductory chapter of this document, security can be organized into the three categories of authentication, authorization, and encryption. Oracle HTTP Server provides support for all three of these categories. It is based on the Apache Web server and its security infrastructure is primarily provided by the Apache module, mod_auth, and the Oracle modules, mod_ossl and mod_osso. The module mod_auth provides authentication based on user name and password pairs, mod_ossl provides confidentiality and authentication with X.509 client certificates over SSL, and mod_osso enables single sign-on for Web applications.

This chapter provides an overview of Oracle HTTP Server security features and configuration information for setting up a secure Web site using them. It contains these topics:

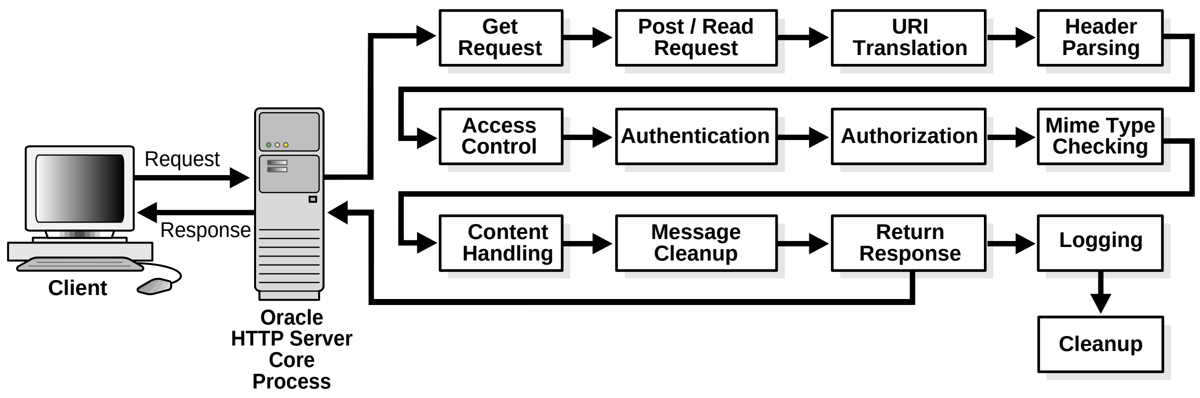
- Overview of Oracle HTTP Server Security
- Understanding Host-Based Access Control
- Overview of User Authentication
- Using Basic Authentication and Authorization with mod_auth
- Using Secure Sockets Layer (SSL) to Authenticate Users
- Using the iasobf Utility to Encrypt Wallet Passwords

> **See Also:** Chapter 3, "Configuring Oracle9iAS Single Sign-On" for information about configuring the Oracle module, mod_osso, that enables single sign-on for Web applications.

Overview of Oracle HTTP Server Security

Based on the Apache model, Oracle HTTP Server provides access control, authentication, and authorization methods that can be configured with access control directives that are used in the `httpd.conf` file. When URL requests arrive at Oracle HTTP Server, they are processed in a number of steps determined by the default design of the server and by what configuration parameters you set in the configuration files. The steps for handling URL requests are implemented through a module or plug-in architecture that is common to many Web listeners. Figure 4–1 shows how URL requests are handled by the server. Each step in this process is handled by a different server module depending on how the server is configured. For example, if only basic authentication is used, then the step labeled "Authentication" in Figure 4–1 represents the module, mod_auth.

Figure 4–1 Steps for Handling URL Requests in Oracle HTTP Server



Oracle HTTP Server provides user authentication and authorization at two stages:

- **First stage (host-based access control)**: based on the details of the incoming HTTP request and its headers, such as IP addresses or host names.
- **Second stage (user authentication)**: based on different criteria depending on the HTTP server configuration. The server can be configured to authenticate users with user name and password pairs that are checked against a list of known users and passwords. You can also configure the server to use single sign-on authentication for Web applications or X.509 client certificates over SSL.

Specifying Configuration Parameters in httpd.conf

Parameters for all Oracle HTTP Server configuration directives that are described in this chapter must be entered in the main HTTP server configuration file called `httpd.conf`, which is located in the following directory:

- (UNIX) `ORACLE_HOME/Apache/Apache/conf/httpd.conf`
- (Windows) `ORACLE_HOME\Apache\Apache\conf\httpd.conf`

See Also: (All of the following documents are located in the Oracle9*i*AS Documentation Library unless otherwise specified.)

- *Oracle HTTP Server Administration Guide* for information about all of the modules that Oracle Corporation supports.
- *Oracle9i Application Server Administrator's Guide* for information about specifying configuration parameters in the `httpd.conf` file in the Oracle HTTP Server home page of the Oracle9*i* Application Server Administration Service. Use this means of setting the configuration parameters for a clustered environment.

The following sections explain how host-based access control and user authentication is handled in the HTTP server and how to configure them.

Understanding Host-Based Access Control

Early in the request processing cycle access control is applied, which can inhibit further processing based on the host name, IP address, or other characteristics such as browser type. You use the `deny`, `allow`, and `order` directives to set this type of access control. These restrictions are configured with Oracle HTTP Server configuration directives and can be based on particular files, directories, or URL formats using the <files>, <directory>, and <location> configuration directives as shown in the sample below:

```
<Directory /internalonly/>
    order deny, allow
    deny from all
    allow from 192.168.1 us.oracle.com
</Directory>
```

In this example, requests originating from any IP address in the 192.168.1.* range or with the host name us.oracle.com are allowed access to files in the directory `/internalonly/`.

If you want to match objects at the file system level, then you must use <Directory> or <Files>. If you want to match objects at the URL level, then you must use <Location>. However, if you want to direct proxy control, then you must use <Directory>.

Note: Allowing or restricting access based on a host name for Internet access is not considered a very good method of providing security, because host names are easy to spoof. While the same is true of IP addresses, sabotage is more difficult. However, setting access control with IP addresses and host names for intranet use is reasonable because the same risks do not apply. This assumes that your firewalls have been properly configured.

Access Control for Virtual Hosts

To set up access control for virtual hosts, place the `AccessConfig` directive inside a virtual host container in the server configuration file, `httpd.conf`. When used in a virtual host container, the `AccessConfig` directive specifies an access control policy contained in a file. The following example shows an excerpt from an `httpd.conf` file which provides the syntax for using `AccessConfig` this way:

```
...
<VirtualHost ip.address.of.host.some_domain.com>
    ... virtual host directives ...
    AccessConfig conf/access.conf
</VirtualHost>
```

Overview of Host-Based Access Control Schemes

Using host-based access control schemes you can control access to restricted areas based on where HTTP requests originate. Oracle HTTP Server uses mod_access and mod_setenvif to perform host-based access control. When you enter configuration directives into the `httpd.conf` that use these modules, the server fulfills or denies requests based on the address or name of the host, or based on the HTTP request header contents.

You can use host-based access control to protect static HTML pages, CGI applications, or components.

Oracle HTTP Server supports four host-based access control schemes:

- Controlling Access by IP Address
- Controlling Access by Domain Name
- Controlling Access by Network or Netmask
- Controlling Access with Environment Variables

All of these allow you to specify the machines from which access to protected areas is granted or denied. Your decision to choose one or more of the host-based access control schemes is determined by which scheme most efficiently protects your restricted content and applications, or which scheme is easiest to maintain.

Note: Although the Oracle HTTP Server is based on the open source Apache server, it contains some access control enhancements which improve security. For example, the Apache server provides for access restrictions per directory or folder by means of files that have the suffix `.htaccess`. The processing of these files is disabled in Oracle HTTP Server, because `.htaccess` processing can produce security problems and degrade performance.

Controlling Access by IP Address

Controlling access with IP addresses is a preferred method of host-based access control. It does not require DNS lookups that consume time, system resources, and make your server vulnerable to DNS spoofing attacks. However, if IP addresses change without warning, your server is left unprotected.

To configure IP address-based access control, use the syntax shown in the following example:

```
<Directory /secure_only/>
     order deny,allow
     deny from all
     allow from 207.175.42.154 192.220.208.9
</Directory>
```

In this example, requests originating from all IP addresses except 207.175.42.154 and 192.220.208.9 are denied access to the `/secure_only/` directory.

Controlling Access by Domain Name

Domain name-based access control can be used with IP address-based access control to solve the problem of IP addresses changing without warning. When you combine these methods, if an IP address changes, then the secure areas of your site are still protected because the domain names you want to keep out will still be denied access.

To combine domain name-based with IP address-based access control, use the syntax shown in the following example:

```
<Directory /co_backgr/>
     order allow,deny
     allow from all
     # 141.217.24.179 is the IP for malicious.cracker.com
     deny from malicious.cracker.com 141.217.24.179
</Directory>
```

In this example all requests for directory `/co_backgr/` are accepted except those that originate from the domain name malicious.cracker.com or the IP address 141.217.24.179. Although this is not a foolproof precaution against domain name or IP address spoofing, it protects your site from malicious.cracker.com even if they change their IP address.

Controlling Access by Network or Netmask

You can control access based on subsets of networks, specified by IP address. The syntax is shown in the following example:

```
<Directory /payroll/>
     order deny,allow
     deny from all
     allow from 10.1.0.0/255.255.0.0
</Directory>
```

In this example, access is allowed from a network/netmask pair. A netmask shows how an IP address is to be divided into network, subnet, and host identifiers. Netmasks enable you to refer to only the host ID portion of an IP address.

The netmask in the example above, 255.255.0.0, is the default netmask setting for a Class B address. The binary ones (decimal 255) mask the network ID and the binary zeroes (decimal 0) retain the host ID of a given IP address.

Controlling Access with Environment Variables

You can use arbitrary environment variables for access control, instead of using IP addresses or domain names. Two directives, `BrowserMatch` and `SetEnvIf`, can be used for this type of access control.

Note: Typically, `BrowserMatch` and `SetEnvIf` are not used to implement security policies. Instead they are used to provide different handling of requests based on browser types and versions.

Use `BrowserMatch` when you want to base access on the type of browser used to send a request. For example, if you want to allow access only to requests that come from a Netscape browser, then use the syntax shown in the following example:

```
BrowserMatch ^Mozilla netscape_browser
<Directory /mozilla-area/>
     order deny,allow
     deny from all
     allow from env=netscape_browser
</Directory>
```

Use `SetEnvIf` when you want to base access on header information contained in the HTTP request. For example, if you want to deny access from any browsers using HTTP version 1.0 or earlier, then use the syntax shown in the following example:

```
SetEnvIf Request_Protocol ^HTTP/1.1 http_11_ok
<Directory /http1.1only/>
     order deny,allow
     deny from all
     allow from env=http_11_ok
</Directory>
```

Overview of User Authentication

User authentication is a term for a process used to ensure that users are who they claim to be. Basic authentication uses a user name and password. SSL uses an exchange of keys and client certificates. Oracle9*i*AS Single Sign-On initially uses a user name and password that are stored on Oracle Internet Directory, but after initial authentication, it places a cookie in the user's browser. During subsequent user logins to applications that are registered to use single sign-on, the browser cookie authenticates users transparently until the cookie expires.

The type of user authentication that you choose to use depends on the type of content you want to protect. Typically, basic authentication that is based on user name and password pairs is adequate for static content that only requires minimal security within an intranet. For Internet communications, SSL, which encrypts data and supports X.509 client certificates, is usually used for transmitting sensitive information such as passwords and authenticating users to Web applications and databases. To provide additional security and convenience, Oracle HTTP Server also supports single sign-on, which allows users to log in to multiple Web applications using a single user name and password.

The following sections describe how to use basic authentication and SSL with Oracle HTTP Server:

- Using Basic Authentication and Authorization with mod_auth
- Using Secure Sockets Layer (SSL) to Authenticate Users

See Also: Chapter 3, "Configuring Oracle9iAS Single Sign-On" for information about how single sign-on works and how to register Oracle HTTP Server to use the Oracle module mod_osso that supports communication with the Oracle9*i*AS Single Sign-On server.

Using Basic Authentication and Authorization with mod_auth

Basic authentication prompts for a user name and password before serving an HTTP request. When a browser requests a page from a protected area, Oracle HTTP Server responds with an Unauthorized message (status code 401) containing a `WWW-Authenticate:` header and the name of the realm configured by the configuration directive, `AuthName`. When the browser receives this response, it prompts for a user name and password. After the user enters a user name and password combination, the browser sends this information back to the server in an Authorization header. In the Authorization header message, the user name and password are encoded as a base 64 encoded string.

Frequently, basic user authentication and authorization are implemented together. User authentication, the second stage of security provided by Oracle HTTP Server, is based on user names and passwords that are checked against a list of known users and passwords. These user name and password pairs may be stored in a variety of forms, such as a text file, database, or directory service. Then configuration directives are used in `httpd.conf` to configure this type of user authentication on the server. For example, mod_auth uses the `AuthUserFile` directive to set up basic authentication.

What Directives to Use for Basic Authentication Configuration with mod_auth

Any authentication scheme that you devise requires that you use a combination of the configuration directives listed in Table 4–1.

Table 4–1 Authentication Configuration Directives

Directive Name	Description
AuthName	Defines the name of the realm in which the user names and passwords are valid. Use quotation marks if the name includes spaces
AuthType	Specifies the authentication type. Most authentication modules use basic authentication, which transmits user names and passwords in clear text. This is not recommended.
AuthUserFile	Specifies the path to a file that contains user names and passwords.
AuthGroupFile	Specifies the path to a file that contains group names and their members.

User Authorization

User authorization involves checking the authenticated user against an access control list that is associated with a specific server resource such as a file or directory. To configure user authorization, place the `require` directive in the `httpd.conf` file, usually within a virtual host container. User authorization is commonly used in combination with user authentication. After the server has authenticated a user’s user name and password, then the server compares the user to an access control list associated with the requested server resource. If Oracle HTTP Server finds the user or the user’s group on the list, then the resource is made available to that user.

Using Secure Sockets Layer (SSL) to Authenticate Users

SSL is an encrypted communication protocol that is designed to securely send messages across the Internet. It resides between Oracle HTTP Server on the application layer and the TCP/IP layer, transparently handling encryption and decryption when a secure connection is made by a client.

SSL provides secure communication between client and server by allowing mutual **authentication**, the use of digital **certificate**s for integrity, and **encryption** for confidentiality. The protocol is designed to support a range of choices for specific algorithms used for **cryptography**, **message digest**s, and **certificate**s. This allows algorithm selection for specific servers to be made based on legal, export, or other concerns, and also enables the protocol to take advantage of new algorithms. Choices are negotiated between client and server at the start of establishing a protocol session.

About Securing HTTP Communication with mod_ossl

One common use of SSL is to secure Web HTTP communication between a browser and a Web server. This case does not preclude the use of non-secured HTTP. The secure version is simply HTTP over SSL (named HTTPS). The differences are that HTTPS uses the URL scheme `https://` rather than `http://`, and its default communication port is 443.

mod_ossl is the Oracle Secure Sockets Layer (SSL) implementation in use with the Oracle database. mod_ossl replaces mod_ssl in the Oracle HTTP Server distribution. Oracle no longer supports mod_ssl. A tool is provided to enable you to migrate from mod_ssl to mod_ossl, and convert your text certificates to Oracle wallets.

mod_ossl supports SSL v. 3.0, and provides:

- Encrypted communication between client and server, using **RSA** or **DES** encryption standards.
- Integrity checking of client-server communication using **MD5** or **SHA** checksum algorithms.
- Certificate management with Oracle **wallet**s.
- Authorization of clients with multiple access checks, exactly as performed in mod_ssl.

Table 4–2 identifies the differences between the Oracle module, mod_ossl, and mod_ssl.

Table 4–2 Differences between mod_ossl and mod_ssl

Feature	mod_ossl	mod_ssl
SSL versions supported	3.0	2.0, 3.0, TLS 1.0
Certificate management	Oracle Wallet[1, 2]	Text file
Per-directory SSL renegotiation	no	yes

[1] Oracle Wallet Manager is a tool that manages certificates for mod_ossl.
[2] Supports obfuscated passwords.

The mod_ssl directives listed below are not supported by mod_ossl.

- SSLRandomSeed
- SSLCertificateFile
- SSLCertificateKeyFile
- SSLCertificateChainFile
- SSLCACertificateFile
- SSLCACertificatePath
- SSLVerifyDepth

Caution: The server will not start if these directives are used.

See Also: Chapter 5 for information about Oracle Wallet Manager and how to use it to request, store, and manage certificates.

Understanding Classes of Directives Used for Configuring SSL

Directives are classified according to the context in which they can be used: global, per-server, and per-directory.

Table 4–3 Classes and Directives

Class	Context	Where Used
global	server configuration	Inside server configuration files, but only outside of container directives (directives such as `VirtualHost` that have a start and end directive).
per-server	server configuration, virtual host	Inside server configuration files, both outside (for the main server) and inside `VirtualHost` directives.
per-directory	server configuration, virtual host, directory	Everywhere; particularly inside the server configuration files.

Note that each class is a subset of the class above it. For example, directives from the per-directory class can also be used in the per-server and global contexts, and directives from the per-server class can be used in the global context.

> **See Also:** *Oracle HTTP Server Administration Guide* in the Oracle9*i*AS Documentation Library for information about using all of the configuration directives available with Oracle HTTP Server.

Using mod_ossl Directives

To configure SSL for your HTTP server, you enter the mod_ossl directives you want to use in the `httpd.conf` file. Each directive is described below.

See Also: "Specifying Configuration Parameters in httpd.conf" on page 4-3 for general information about how to use these parameters in the main HTTP server configuration file.

SSLWallet

Description:	Specifies the location of the wallet with its **WRL**. Oracle Corporation recommends that you use wallets created with the Auto Login feature of Oracle Wallet Manager. Wallets that are created with the Auto Login feature do not require a password, so when they are used, the `SSLWalletPassword` directive does not need to be set.
Syntax:	`SSLWallet` *`wrl`* The format of `wrl` is: *file:path to wallet*
Example:	`SSLWallet` *`file:/etc/ORACLE/WALLETS/server`* Other values of wrl may be used as permitted by the Oracle SSL product.
Default:	None
Context:	server configuration, virtual host

Caution: When Auto Login is enabled for a wallet, that wallet is only available to the operating system user who created it.

See Also: "Using Auto Login" on page 5-15 for information about using the Auto Login feature of Oracle Wallet Manager.

SSLWalletPassword

Description:	Wallet password needed to access the wallet specified within the same context. You can choose either a **cleartext** wallet password or an obfuscated password. The obfuscated password is created with the command line tool `iasobf` described below. If you must use a regular wallet, Oracle Corporation recommends that you use the obfuscated password instead of a cleartext password.
Syntax:	`SSLWalletPassword` *`password`* If no password is required do not set this directive. Note: If a wallet created with the Auto Login feature of Oracle Wallet Manager is used, then do not set this directive because these wallets do not require passwords.
Default:	None
Context:	server configuration, virtual host

See Also: "Using the iasobf Utility to Encrypt Wallet Passwords" on page 4-32

SSLPassPhraseDialog

Description:	Type of pass phrase dialog for wallet access. mod_ossl asks the administrator for a pass phrase in order to access the wallet.
Valid Values	■ `builtin` - when the server is started, mod_ossl prompts for a password for each wallet. ■ `exec:`*`path/to/program`* - when the server is started, mod_ossl calls an external program configured for each wallet. This program is invoked with two arguments: *`servername:portnumber`* and RSA or DSA.
Syntax:	`SSLPassPhraseDialog` *`type`*
Example:	`SSLPassPhraseDialog exec:`*`/usr/local/apache/sbin/pfilter`*
Default:	`SSLPassPhraseDialog` *`builtin`*
Context:	server configuration

SSLCARevocationPath

Description:	Specifies the directory where **PEM**-encoded Certificate Revocation Lists (CRLs) are stored. These CRLs come from the **CA**s (Certificate Authorities) that you accept certificates from. If a client attempts to authenticate itself with a certificate that is on one of these CRLs, then the certificate is revoked and the client cannot authenticate itself with your server.
Syntax:	`SSLCARevocationPath` *`path/to/CRL_directory/`*
Example:	`SSLCARevocationPath /ias2/Apache/conf/ssl.crl/`
Default:	None
Context:	server config, virtual host

SSLCARevocationFile

Description:	Specifies the file where you can assemble the Certificate Revocation Lists (CRLs) from **CA**s (Certificate Authorities) that you accept certificates from. These are used for client authentication. Such a file is the concatenation of various **PEM**-encoded CRL files in order of preference. This directive can be used alternatively or additionally to SSLCARevocationPath.
Syntax:	`SSLCARevocationFile` *`file_name`*
Example:	`SSLCARevocationFile /ias2/Apache/conf/ssl.crl/ca_bundle.crl`
Default:	None
Context:	server config, virtual host

SSLMutex

Description: Type of semaphore (lock) for SSL engine's mutual exclusion of operations that have to be synchronized between HTTP Server processes.

Valid Values:

- `none` - Uses no mutex at all

  Not recommended, because the mutex synchronizes the write access to the SSL session cache. If you don't configure a mutex, the session cache can become garbled.

- `file:`*`path/to/mutex`* - Uses a file for locking. The process ID (PID) of the HTTP Server parent process is appended to the filename to ensure uniqueness. If the filename does not begin with a slash ('/'), it is assumed to be relative to ServerRoot.

  This setting is not available on Windows.

- `sem` - Uses an operating system semaphore to synchronize writes. On UNIX, it would be a Sys V IPC semaphore; on Windows, it is a Windows Mutex.

  This is the best choice, if the operating system supports it.

Example: `SSLMutex file:`*`/usr/local/apache/logs/ssl_mutex`*

Syntax: `SSLMutex` *`type`*

Default: `SSLMutex` *`none`*

Context: server configuration

SSLSessionCache

Description:	Specifies the global/interprocess session cache storage type. The cache provides an optional way to speed up parallel request processing.
Valid Values:	■ `none` - disables the global/interprocess session cache Produces no impact on functionality, but produces a noticeable performance penalty. ■ `shmht:`*`/path/to/datafile[bytes]`* - Uses a high-performance hash table (`bytes` specifies approximate size) inside a shared memory segment in RAM, which is established by the `/path/to/datafile`. This hash table synchronizes the local SSL memory caches of the server processes. ■ `shmcb:`*`/path/to/datafile[bytes]`* - Uses a high-performance Shared Memory Cyclic Buffer (SHMCB) session cache to synchronize the local SSL memory caches of the server processes. The performance of `shmcb` is more uniform in all environments when compared to `shmht`.
Syntax:	`SSLSessionCache` *`type`*

Examples:	`SSLSessionCache shmht:`*`/iasv2/Apache/Apache/logs/ssl_scache(512000)`*
	`SSLSessionCache shmcb:`*`/iasv2/Apache/Apache/logs/ssl_scache(512000)`*

Default:	`SSLSessionCache` *`none`*
Context:	server configuration

SSLSessionCacheTimeout

Description:	Number of seconds before an SSL session in the session cache expires.
Syntax:	`SSLSessionCacheTimeout` *`seconds`*
Default:	300
Context:	server configuration

SSLProtocol

Description:	SSL protocol(s) for mod_ossl to use when establishing the server environment. Clients can only connect with one of the specified protocols.
Valid Values:	■ `SSLv3` SSL version 3.0 ■ `All` SSL version 3.0 and any other version supported by Oracle products
Example:	To specify only SSL version 3.0, set this directive to the following: `SSLProtocol` *`+SSLv3`*
Syntax:	`SSLProtocol` *`[+-]`* *protocol*
Default:	`SSLProtocol +SSLv3`
Context	server configuration, virtual host

SSLCipherSuite

Description: Specifies the SSL **cipher suite** that the client can use during the SSL handshake. This directive uses a colon-separated cipher specification string to identify the cipher suite. Table 4–4 shows the tags you can use in the string to describe the cipher suite you want.

The tags are joined together with prefixes to form the cipher specification string.

Valid Values:

- `none` Adds the cipher to the list
- `+:` Adds the cipher to the list and place them in the correct location in the list
- `-:` Remove the cipher from the list (can be added later)
- `!:` Remove the cipher from the list permanently

Example: `SSLCipherSuite` *`ALL:!LOW:!DH`*

In this example, all ciphers are specified except low strength ciphers and those using the **Diffie-Hellman key negotiation algorithm**.

Syntax: `SSLCipherSuite` *`cipher-spec`*

Default: None

Context server configuration, virtual host, directory

Table 4–4 SSLCipher Suite Tags

Function	Tag	Meaning
Key exchange	kRSA	RSA key exchange
Key exchange	kDHr	Diffie-Hellman key exchange with RSA key
Authentication	aNULL	No authentication
Authentication	aRSA	RSA authentication
Authentication	aDH	Diffie-Hellman authentication
Encryption	eNULL	No encryption
Encryption	DES	DES encoding
Encryption	3DES	Triple DES encoding
Encryption	RC4	RC4 encoding
Data Integrity	MD5	MD5 hash function

Table 4–4 SSLCipher Suite Tags (Cont.)

Function	Tag	Meaning
Data Integrity	SHA	SHA hash function
Aliases	SSLv3	All SSL version 3.0 ciphers
Aliases	EXP	All export ciphers
Aliases	EXP40	ALl 40-bit export ciphers only
Aliases	EXP56	All 56-bit export ciphers only
Aliases	LOW	All low strength ciphers (export and single DES)
Aliases	MEDIUM	All ciphers with 128-bit encryption
Aliases	HIGH	All ciphers using triple DES
Aliases	RSA	All ciphers using RSA key exchange
Aliases	DH	All ciphers using Diffie-Hellman key exchange

Note:

- Not all of the ciphers shown in the tags listed in Table 4–4 are supported by Oracle Advanced Security. Table 4–5 lists those supported as of version 9*i*.
- There are restrictions if export versions of browsers are used. Oracle module, mod_ossl, supports RC4-40 encryption only when the server uses 512 bit key size wallets.

Table 4–5 Cipher Suites Supported in Oracle Advanced Security 9i

Cipher Suite	Authentication	Encryption	Data Integrity
SSL_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES EDE CBC	SHA
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4 128	SHA
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4 128	MD5
SSL_RSA_WITH_DES_CBC_SHA	RSA	DES CBC	SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH anon	3DES EDE CBC	SHA
SSL_DH_anon_WITH_RC4_128_MD5	DH anon	RC4 128	MD5
SSL_DH_anon_WITH_DES_CBC_SHA	DH anon	DES CBC	SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5	RSA	RC4 40	MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA	DES40 CBC	SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5	DH anon	RC4 40	MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA	DH anon	DES40 CBC	SHA

SSLVerifyClient

Description:	Specifies whether or not a client must present a certificate when connecting.
Valid Values:	▪ `none` - No client certificate is required ▪ `optional` - Client may present a valid certificate ▪ `require` - Client must present a valid certificate
Syntax:	`SSLVerifyClient` *`level`*
Default:	None
Context	server configuration, virtual host

Note: The level `optional_no_ca` included with mod_ssl (in which the client can present a valid certificate, but it need not be verifiable) is not supported in mod_ossl.

SSLLog

Description:	Specifies where the SSL engine log file will be written. (Error messages will also be duplicated to the standard HTTP server log file specified by the `ErrorLog` directive.) Place this file at a location where only root can write, so that it cannot be used for symlink attacks. If the filename does not begin with a slash ('/'), it is assumed to be relative to the `ServerRoot`. If the filename begins with a bar ('|'), then the string following the bar is expected to be a path to an executable program to which a reliable pipe can be established. This directive should occur only once per virtual server configuration.
Syntax:	`SSLVerifyClient` *`path/to/filename`*
Default:	None
Context	server configuration, virtual host

SSLLogLevel

Description:	Specifies the verbosity degree of the SSL engine log file.
Valid Values:	The levels are (in ascending order, where each level is included in the levels above it):

- `none` - No dedicated SSL logging is done. Messages of type 'error' are duplicated to the standard HTTP server log file specified by the `ErrorLog` directive.
- `error` - Only messages of the type 'error' (conditions that stop processing) are logged.
- `warn` - Messages that notify of non-fatal problems (conditions that do not stop processing) are logged.
- `info` - Messages that summarize major processing actions are logged.
- `trace` - Messages that summarize minor processing actions are logged.
- `debug` - Messages that summarize development and low-level I/O operations are logged.

Syntax:	`SSLLogLevel` *`level`*
Default:	None
Context	server configuration, virtual host

SSLOptions

Description: Controls various runtime options on a per-directory basis. In general, if multiple options apply to a directory, the most comprehensive option is applied (options are not merged). However, if all of the options in an `SSLOptions` directive are preceded by a plus ('+') or minus ('-') symbol, then the options are merged. Options preceded by a plus are added to the options currently in force, and options preceded by a minus are removed from the options currently in force.

Valid Values:

- `StdEnvVars` - Creates the standard set of CGI/SSI environment variables that are related to SSL. This is disabled by default because the extraction operation uses a lot of CPU time and usually has no application when serving static content. Typically, you only enable this for CGI/SSI requests.
- `ExportCertData` - Enables the following additional CGI/SSI variables:

  `SSL_SERVER_CERT`

  `SSL_CLIENT_CERT`

  `SSL_CLIENT_CERT_CHAIN_n` (where n= 0, 1, 2...)

  These variables contain the Privacy Enhanced Mail (**PEM**)-encoded X.509 certificates for the server and the client for the current HTTPS connection, and can be used by CGI scripts for deeper certificate checking. All other certificates of the client certificate chain are provided. This option is off by default because there is a performance cost associated with using it.
- `FakeBasicAuth` - Translates the subject Distinguished Name (DN) of the client X.509 certificate into an HTTP basic authorization user name. This means that the standard HTTP server authentication methods can be used for access control. Note that no password is obtained from the user; the string 'password' is substituted.

Valid Values: (for SSLOptions continued)	■ `StrictRequire` - Denies access when, according to `SSLRequireSSL` or `SSLRequire` directives, access should be forbidden. Without `StrictRequire`, it is possible for a '`Satisfy any`' directive setting to override the `SSLRequire` or `SSLRequireSSL` directive, allowing access if the client passes the host restriction or supplies a valid user name and password. Thus, the combination of `SSLRequireSSL` or `SSLRequire` with `SSLOptions +StrictRequire` gives mod_ossl the ability to override a `'Satisfy any'` directive in all cases.
	■ `CompatEnvVars` - Exports obsolete environment variables for backward compatibility to Apache SSL 1.x, mod_ssl 2.0.x, Sioux 1.0, and Stronghold 2.x. Use this to provide compatibility to existing CGI scripts.
	■ `OptRenegotiate` - This enables optimized SSL connection renegotiation handling when SSL directives are used in a per-directory context.
Syntax:	`SSLOptions` *`[+-] option`*
Default:	None
Context	server configuration, virtual host, directory

SSLRequireSSL

Description:	Denies access to clients not using SSL. This is a useful directive for absolute protection of an SSL-enabled virtual host or directories in which configuration errors could create security vulnerabilities.
Syntax:	`SSLRequireSSL`
Default:	None
Context	directory

SSLRequire

Description: Denies access unless an arbitrarily complex boolean expression is true. The expression must match the syntax below (given as a BNF grammar notation):

```
expr ::= "true" | "false"
"!" expr
expr "&&" expr
expr "||" expr
"(" expr ")"

comp ::=word "==" word | word "eq" word
word "!=" word |word "ne" word
word "<" word |word "lt" word
word "<=" word |word "le" word
word ">" word |word "gt" word
word ">=" word |word "ge" word
word "=~" regex
word "!~" regex
wordlist ::= word
wordlist "," word

word ::= digit
cstring
variable
function

digit ::= [0-9]+

cstring ::= "..."

variable ::= "%{varname}"
```

Table 4–6 and Table 4–7 list standard and SSL variables. These are valid values for varname.

```
function ::= funcname "(" funcargs ")"
```

For funcname, the following function is available:

`file(`*`filename`*`)`
The file function takes one string argument, the filename, and expands to the contents of the file. This is useful for evaluating the file’s contents against a regular expression.

Syntax: `SSLRequire` *`expression`*

Default: None

Context directory

Table 4–6 Standard variables for SSLRequire varname

Standard Variables		
HTTP_USER_AGENT	PATH_INFO	AUTH_TYPE
HTTP_REFERER	QUERY_STRING	SERVER_SOFTWARE
HTTP_COOKIE	REMOTE_HOST	API_VERSION
HTTP_FORWARDED	REMOTE_IDENT	TIME_YEAR
HTTP_HOST	IS_SUBREQ	TIME_MON
HTTP_PROXY_ CONNECTION	DOCUMENT_ROOT	TIME_DAY
HTTP_ACCEPT	SERVER_ADMIN	TIME_HOUR
HTTP:headername	SERVER_NAME	TIME_MIN
THE_REQUEST	SERVER_PORT	TIME_SEC
REQUEST_METHOD	SERVER_PROTOCOL	TIME_WDAY
REQUEST_SCHEME	REMOTE_ADDR	TIME
REQUEST_URI	REMOTE_USER	ENV:variablename
REQUEST_FILENAME		

Table 4–7 SSL variables for SSLRequire varname

SSL Variables		
HTTPS	SSL_PROTOCOL	SSL_CIPHER_ALGKEYSIZE
SSL_CIPHER	SSL_CIPHER_EXPORT	SSL_VERSION_INTERFACE
SSL_CIPHER_USEKEYSIZE	SSL_VERSION_LIBRARY	SSL_SESSION_ID
SSL_CLIENT_V_END	SSL_CLIENT_M_SERIAL	SSL_CLIENT_V_START
SSL_CLIENT_S_DN_ST	SSL_CLIENT_S_DN	SSL_CLIENT_S_DN_C
SSL_CLIENT_S_DN_CN	SSL_CLIENT_S_DN_O	SSL_CLIENT_S_DN_OU
SSL_CLIENT_S_DN_G	SSL_CLIENT_S_DN_T	SSL_CLIENT_S_DN_I
SSL_CLIENT_S_DN_UID	SSL_CLIENT_S_DN_S	SSL_CLIENT_S_DN_D
SSL_CLIENT_I_DN_C	SSL_CLIENT_S_DN_Email	SSL_CLIENT_I_DN
SSL_CLIENT_I_DN_O	SSL_CLIENT_I_DN_ST	SSL_CLIENT_I_DN_L

Table 4–7 SSL variables for SSLRequire varname (Cont.)

SSL Variables		
SSL_CLIENT_I_DN_T	SSL_CLIENT_I_DN_OU	SSL_CLIENT_I_DN_CN
SSL_CLIENT_I_DN_S	SSL_CLIENT_I_DN_I	SSL_CLIENT_I_DN_G
SSL_CLIENT_I_DN_Email	SSL_CLIENT_I_DN_D	SSL_CLIENT_I_DN_UID
SSL_CLIENT_CERT	SSL_CLIENT_CERT_ CHAINn	SSL_CLIENT_VERIFY
SSL_CLIENT_M_VERSION	SSL_SERVER_M_VERSION	SSL_SERVER_V_START
SSL_SERVER_V_END	SSL_SERVER_M_SERIAL	SSL_SERVER_S_DN_C
SSL_SERVERT_S_DN_ST	SSL_SERVER_S_DN	SSL_SERVER_S_DN_OU
SSL_SERVER_S_DN_CN	SSL_SERVER_S_DN_O	SSL_SERVER_S_DN_I
SSL_SERVER_S_DN_G	SSL_SERVER_S_DN_T	SSL_SERVER_S_DN_D
SSL_SERVER_S_DN_UID	SSL_SERVER_S_DN_S	SSL_SERVER_I_DN
SSL_SERVER_I_DN_C	SSL_SERVER_S_DN_Email	SSL_SERVER_I_DN_L
SSL_SERVER_I_DN_O	SSL_SERVER_I_DN_ST	SSL_SERVER_I_DN_CN
SSSL_SERVER_I_DN_T	SSL_SERVER_I_DN_OU	SSL_SERVER_I_DN_G
SSL_SERVER_I_DN_I		

Using the iasobf Utility to Encrypt Wallet Passwords

If you are using an Oracle Wallet that has been created with Auto Login enabled (an SSO wallet), then you do not need to use this utility. However, if you must use a regular wallet with a password, then Oracle Corporation recommends that you use the password obfuscation tool `iasobf`, which is located in the HTTP server bin directory to generate an obfuscated wallet password from a **cleartext** password.

To generate an obfuscated wallet password, the command syntax is:

```
iasobf -p password
```

The obfuscated password is printed to the terminal. The arguments are optional. If you do not type them, the tool will prompt you for the password.

For Windows systems: The corresponding tool for Windows environments is called `osslpassword`, which can be used in the same way as `iasobf`.

See Also: All of the following documents can be found in the Oracle9*i*AS Documentation Library unless otherwise specified.

- *Oracle HTTP Server Administration Guide* for complete information about administering Oracle HTTP Server.
- *Oracle9i Application Server Performance Guide* for server performance tuning information.
- *Oracle9i Application Server Administrator's Guide* for information about using the management console GUI tool for Oracle HTTP Server.
- Chapter 5, "Using Oracle Wallet Manager" for information about creating wallets and using them.

5 Using Oracle Wallet Manager

Security administrators use Oracle Wallet Manager to manage public-key security credentials on Oracle9*i* Application Server and other Oracle clients and servers. It can be used to request, store, and manage certificates. The wallets it creates can be opened with Oracle Wallet Manager.

This chapter describes the Oracle Wallet Manager, in the following sections:

- About Public Key Infrastructure (PKI)
- PKCS #12 Support
- Multiple Certificate Support
- LDAP Directory Support
- Managing Wallets
- Managing Certificates

About Public Key Infrastructure (PKI)

Traditional private-key or symmetric-key cryptography requires a single, secret key that is shared by two or more parties to a secure communication. This key is used to both encrypt and decrypt secure messages sent between the parties, requiring prior, secure distribution of the key to each party. *The problem with this method is that it is difficult to securely transmit and store the key.*

Public-key cryptography provides a solution to this problem, by employing **public/private key pair**s and a secure method for key distribution. The freely available **public key** is used to encrypt messages that can *only* be decrypted by the holder of the associated **private key**. The private key is securely stored, together with other security credentials, in an encrypted container—called a **wallet**.

Public-key algorithms can guarantee the secrecy of a message, but they don't necessarily guarantee *secure communications* because they don't verify the identities of the communicating parties. In order to establish secure communications, it is important to verify that the public key used to encrypt a message does in fact belong to the target recipient. Otherwise, a third party can potentially eavesdrop on the communication and intercept public key requests, substituting its own public key for a legitimate key (the **man-in-the-middle** attack).

In order to avoid such an attack, it is necessary to verify the owner of the public key, a process called **authentication**. Authentication can be accomplished through a **certificate authority** (CA)—a third party that is trusted by both of the communicating parties.

The CA issues public key certificates that contain an entity's name, public key, and certain other security credentials. Such credentials typically include the CA name, the CA signature, and the certificate effective dates (From Date, To Date).

The CA uses its private key to encrypt a message, while the public key is used to decrypt it, thus verifying that the message was encrypted by the CA. The CA public key is well known, and does not have to be authenticated each time it is accessed. Such CA public keys are stored in an Oracle wallet.

Wallet Password Management

Oracle Wallet Manager includes an enhanced wallet password management module that enforces Password Management Policy guidelines, including the following:

- Minimum password length (8 characters)
- Maximum password length unlimited
- Alphanumeric character mix required

Strong Wallet Encryption

Oracle Wallet Manager stores private keys associated with X.509 certificates, requiring strong encryption. Accordingly, 3-key Triple-DES, which is a substantially stronger encryption algorithm is supported.

Microsoft Windows Registry

Oracle Wallet Manager lets you optionally store multiple Oracle wallets in the user profile area of the Microsoft Windows System Registry (for Windows 95 / 98 / ME / NT 4.0 / 2000), or in a Windows file management system. Storing your wallets in the registry provides the following benefits:

- **Better Access Control.** Wallets stored in the user profile area of the registry are only accessible by the associated user. User access controls for the system thus become, by extension, access controls for the wallets. In addition, when a user logs out of a system, access to that user's wallets is effectively precluded.
- **Easier Administration.** Since wallets are associated with specific user profiles, no permissions need to be managed, and the wallets stored in the profile are automatically deleted when the user profile is deleted. Oracle Wallet Manager can be used to create and manage the wallets in the registry.
- **Improved Security.** Because the wallets are imbedded in the registry, the wallets associated with a particular user profile are transparent to all other users. Viewed in combination with *better access control* and *easier administration*, this amounts to an additional security layer for Oracle wallets.

Options Supported:

- Open wallet from the Registry
- Save wallet to the Registry
- Save As to a different Registry location
- Delete wallet from the Registry
- Open wallet from the file system and save it to the Registry
- Open wallet from the Registry and save it to the file system

See Also:

- *Oracle9i Administrator's Guide for Windows* in the Oracle Database Documentation Library
- *Oracle9i Network, Directory and Security Guide for Windows* in the Oracle Database Documentation Library

Oracle Wallet Functions

Oracle Wallet Manager is a stand-alone Java application that wallet owners use to manage and edit the security credentials in their Oracle wallets. These tasks include the following:

- Generating a public/private key pair and creating a certificate request for submission to a CA.
- Installing a certificate for the entity.
- Configuring **trusted certificate**s for the entity.
- Opening a wallet to enable access to PKI-based services.
- Creating a wallet that can be accessed by Oracle Wallet Manager.
- Uploading a wallet to an LDAP directory such as Oracle Internet Directory.
- Downloading a wallet from an LDAP directory such as Oracle Internet Directory.
- Importing wallets.
- Exporting wallets.

See Also: *Oracle Directory Service Integration and Deployment Guide* in the Oracle9*i*AS Documentation Library, if you need to synchronize your existing LDAP directory with Oracle Internet Directory.

Backward Compatibility

Oracle Wallet Manager is backward-compatible to Oracle Data Server, Release 8.1.5.

PKCS #12 Support

Oracle Wallet Manager stores X.509 certificates and **private keys** in industry-standard, PKCS #12 format. This makes the Oracle wallet structure interoperable with supported third party PKI applications, and provides wallet portability across operating systems.

Note: Although Oracle Advanced Security and Oracle Wallet Manager fully comply with PKCS #12, there may be some compatibility issues using third-party products—such as Netscape Communicator and Microsoft Internet Explorer.

Importing Third-Party Wallets

Oracle Wallet Manager can import and support the following PKCS #12-format wallets, subject to product-specific procedures and limitations:

- Netscape Communicator 4.x
- Microsoft Internet Explorer 5.x
- OpenSSL

To import a third-party wallet:

1. Follow the product-specific procedure to export the wallet.
2. Save the exported wallet to a platform-specific file name in a directory expected by your application.

   For UNIX and Windows NT, the file name is `ewallet.p12`.

   For other platforms, see the platform-specific documentation.
3. Open the wallet by specifying the directory that contains the wallet in the Open dialog box.

   See Also: Importing a Trusted Certificate on page 5-21.

Notes:

- You must copy the third-party PKCS #12 wallet file name to a directory expected by Oracle Wallet Manager and change the name; the UNIX/NT wallet file name is `ewallet.p12`.
- Since browsers typically do not export **trusted certificate**s under PKCS #12 (other than the signer's own certificate), you may need to add trust points to authenticate the other party in the SSL connection. You can use Oracle Wallet Manager to do this.

Exporting Oracle Wallets

Oracle Wallet Manager can export its own wallets to third party environments. To export a wallet:

1. Use Oracle Wallet Manager to save the wallet file.
2. Follow the third-party product-specific import procedure to import a platform-specific PKCS #12 wallet file created by Oracle Wallet Manager (called `ewallet.p12` on UNIX and NT platforms).

Note: Current browsers typically support import of only one user certificate per wallet. Accordingly, you must export an Oracle **single key-pair wallet**, even though Oracle Wallet Manager supports multiple user certificates per wallet.

Multiple Certificate Support

Oracle wallet tools support multiple **certificate**s for each wallet, supporting the following **Oracle PKI certificate usages**:

- SSL
- S/MIME signature
- S/MIME encryption
- Code-Signing
- CA Certificate Signing

Oracle Wallet Manager supports multiple certificates for a *single digital entity*, where each certificate can be used for a set of Oracle PKI certificate usages—but the same certificate cannot be used for all such usages (See: Tables 5–2 and 5–3 for legal usage combinations). There must be a one-to-one mapping between certificate requests and certificates. The same certificate request cannot be used to obtain multiple certificates, installed in the same wallet.

Oracle Wallet Manager uses X.509 V3 extension `KeyUsage` to define Oracle PKI certificate usages (Table 5–1):

Table 5–1 KeyUsage Values

Value	Usage
0	digitalSignature
1	nonRepudiation
2	keyEncipherment
3	dataEncipherment
4	keyAgreement
5	keyCertSign
6	cRLSign
7	encipherOnly
8	decipherOnly

When installing a certificate (user certificate, **trusted certificate**), Oracle Wallet Manager uses Tables 5–2 and 5–3 to map the KeyUsage extension values to Oracle PKI certificate usages:

Table 5–2 OWM Import of User Certificate to an Oracle Wallet

KeyUsage Value	Critical?[1]	Usage
none	na	Certificate is importable for SSL or S/MIME encryption use.
0 alone, or any combination including 0 but excluding 5 and 2	na	Accept certificate for S/MIME signature or code-signing use.
1 alone	Yes	Not importable.
	No	Accept certificate for S/MIME signature or code-signing use.
2 alone, or 2 + any combination excluding 5	na	Accept certificate for SSL or S/MIME encryption use.
5 alone, or any combination including 5	na	Accept certificate for CA certificate signing use.
Any settings not listed above	Yes	Not importable.
	No	Certificate is importable for SSL or S/MIME encryption use.

[1] If the KeyUsage extension is *critical*, the certificate cannot be used for other purposes.

Table 5–3 OWM Import of Trusted Certificates to an Oracle Wallet

KeyUsage Value	Critical?[1]	Usage
none	na	Importable.
Any combination excluding 5	Yes	Not importable.
	No	Importable.
5 alone, or any combination including 5	na	Importable.

[1] If the KeyUsage extension is *critical*, the certificate cannot be used for other purposes.

You should obtain certificates from the certificate authority with the correct `KeyUsage` value for the required Oracle PKI certificate usage. A single wallet can contain multiple **key pair**s for the same usage. Each certificate can support multiple

Oracle PKI certificate usages, as indicated by Tables 5–2 and 5–3. Oracle PKI applications use the first certificate containing the required PKI certificate usage.

For example: For SSL usage, the first certificate containing the SSL Oracle PKI certificate usage is used.

Note: SSL Oracle PKI Certificate Usage is the only usage supported by Oracle PKI applications.

LDAP Directory Support

Oracle Wallet Manager can upload wallets to—and retrieve them from—an LDAP-compliant directory, such as Oracle Internet Directory.

Storing wallets in a centralized LDAP-compliant directory lets users access them from multiple locations or devices, ensuring consistent and reliable user authentication—while providing centralized wallet management throughout the wallet life cycle. To prevent accidental over-write of functional wallets, only wallets containing a functional certificate can be uploaded.

Oracle Wallet Manager requires that enterprise users are already defined and configured in the LDAP directory, to be able to upload or download wallets. If a directory contains Oracle8*i* (or prior) users, they are automatically upgraded to use the wallet upload / download feature—upon first use.

> **See Also:** *Oracle Advanced Security Administrator's Guide* in the Oracle Database Documentation Library for information about creating enterprise users.

Oracle Wallet Manager downloads a user wallet using a simple password-based connection to the LDAP directory. However, for uploads it uses an SSL connection if the open wallet contains a certificate with SSL Oracle PKI certificate usage.

> **See Also:** Multiple Certificate Support on page 5-7, for more information about Oracle PKI certificate user.

If an SSL certificate is not present in the wallet, password-based authentication is used.

> **Note:** The directory password and the wallet password are independent, and can be different. Oracle recommends that these passwords are maintained to be consistently different, where neither one can logically be derived from the other.

Managing Wallets

This section describes how to create a new wallet and perform associated wallet management tasks, such as generating certificate requests, exporting certificate requests, and importing certificates into wallets, in the following subsections:

Starting Oracle Wallet Manager

To start Oracle Wallet Manager:

- Microsoft NT:

  Select `Start—>Programs—>Oracle-`*`ORACLE_HOME`*`—>Network Administration—>Wallet Manager`

- UNIX: Enter `owm` at the command line.

Creating a New Wallet

Create a new wallet as follows:

1. Choose `Wallet > New` from the menu bar; the New Wallet dialog box appears.
2. Follow the required guidelines for creating a password and enter a password in the Wallet Password field.

Because an Oracle wallet contains user credentials that can be used to authenticate the user to multiple databases, it is especially important to choose a strong wallet password. A malicious user who guesses the wallet password can access all the databases to which the wallet owner has access.

Caution: It is strongly recommended that users avoid choosing easily guessed passwords based on user names, phone numbers, or government identification numbers, such as "admin0," "oracle1," or "2135551212A." This prevents a potential attacker from using personal information to deduce the users' passwords. It is also a prudent security practice for users to change their passwords periodically, such as once per month or once per quarter.

See Also: Wallet Password Management on page 5-2.

3. Re-enter that password in the Confirm Password field.
4. Choose `OK` to continue.
5. An Alert is displayed, and informs you that a new empty wallet has been created. It prompts you to decide whether you want to create a certificate request.

   See also: Adding a Certificate Request on page 5-17

   If you choose Cancel, you are returned to the Oracle Wallet Manager main window. The new wallet you just created appears in the left window pane. The certificate has a status of `Empty`, and the wallet displays its default trusted certificates.
6. Select `Wallet > Save In System Default` to save the new wallet.

   If you do not have permission to save the wallet in the system default, you can save it to another location.

   A message at the bottom of the window informs you that the wallet was successfully saved.

Opening an Existing Wallet

Open a wallet that already exists in the file system directory as follows:

1. Choose `Wallet > Open` from the menu bar; the Select Directory dialog box appears.
2. Navigate to the directory location in which the wallet is located, and select the directory.
3. Choose OK; the Open Wallet dialog box appears.
4. Enter the wallet password in the Wallet Password field.
5. Choose OK.
6. The message `Wallet opened successfully` appears at the bottom of the window, and you are returned to the Oracle Wallet Manager main window. The wallet's certificate and its trusted certificates are displayed in the left window pane.

Closing a Wallet

To close an open wallet in the currently selected directory:

- Choose `Wallet > Close`.
- The message `Wallet closed successfully` appears at the bottom of the window, to confirm that the wallet is closed.

Saving Changes

To save your changes to the current open wallet:

- Choose `Wallet > Save`.
- A message at the bottom of the window confirms that the wallet changes were successfully saved to the wallet in the selected directory location.

Saving the Open Wallet to a New Location

Use the `Save As` option to save the current open wallet to a new directory location:

1. Choose `Wallet > Save As;` the select directory dialog box appears.
2. Select a directory location to save the wallet.

3. Choose `OK`.

   The following message appears if a wallet already exists in the selected directory:

   ```
   A wallet already exists in the selected path. Do you
   want to overwrite it?.
   ```

   Choose `Yes` to overwrite the existing wallet, or `No` to save the wallet to another directory.

   A message at the bottom of the window confirms that the wallet was successfully saved to the selected directory location.

Saving in System Default

Use the `Save in System Default` menu option to save the current open wallet to the system default directory location. This makes the current open wallet the wallet that is used by SSL:

- Choose `Wallet > Save in System Default`.
- A message at the bottom of the window confirms that the wallet was successfully saved in the system default wallet location.

Deleting the Wallet

To delete the current open wallet:

1. Choose `Wallet > Delete`; the `Delete Wallet` dialog box appears.
2. Review the displayed wallet location to verify you are deleting the correct wallet.
3. Enter the wallet password.
4. Choose `OK`; a dialog panel appears to inform you that the wallet was successfully deleted.

Note: Any open wallet in application memory will remain in memory until the application exits. Therefore, deleting a wallet that is currently in use does not immediately affect system operation.

Changing the Password

A password change is effective immediately. The wallet is saved to the currently selected directory, with the new encrypted password.To change the password for the current open wallet:

1. Choose `Wallet > Change Password;` the `Change Wallet Password` dialog box appears.
2. Enter the existing wallet password.
3. Enter the new password.

   See Also: Wallet Password Management on page 5-2, for password policy restrictions.

4. Re-enter the new password.
5. Choose `OK`.

A message at the bottom of the window confirms that the password was successfully changed.

Using Auto Login

The Oracle Wallet Manager Auto Login feature creates an obfuscated copy of the wallet and enables PKI-based access to services without a password until the Auto Login feature is disabled for the wallet. When Auto Login is enabled for a wallet, it is only available to the operating system user who created that wallet.

You must enable Auto Login if you want single sign-on access to multiple Oracle databases (disabled by default).

Enabling Auto Login

To enable Auto Login:

1. Choose `Wallet` from the menu bar.
2. Choose the check box next to the Auto Login menu item; a message at the bottom of the window displays `Autologin enabled`.

Disabling Auto Login

To disable Auto Login:

1. Choose `Wallet` from the menu bar.

2. Choose the check box next to the Auto Login menu item; a message at the bottom of the window displays `Autologin disabled`.

Managing Certificates

Oracle Wallet Manager uses two kinds of certificates: user certificates and trusted certificates. This section describes how to manage both certificate types, in the following subsections:

- Managing User Certificates
- Managing Trusted Certificates

Note: You must first install a trusted certificate from the certificate authority before you can install a user certificate issued by that authority. Several trusted certificates are installed by default when you create a new wallet.

Managing User Certificates

Managing user certificates involves the following tasks:

- Adding a Certificate Request
- Importing the User Certificate into the Wallet
- Removing a User Certificate from a Wallet
- Removing a Certificate Request
- Exporting a User Certificate
- Exporting a User Certificate Request

Adding a Certificate Request

You can use this task to add multiple certificate requests. Note that when creating multiple requests, Oracle Wallet Manager automatically populates each subsequent request dialog box with the content of the initial request—which you can then edit.

The actual certificate request becomes part of the wallet. You can reuse any certificate request to obtain a new certificate. However, you cannot edit an existing certificate request; store only a correctly filled out certificate request in a wallet.

To create a PKCS #10 certificate request:

1. Choose `Operations > Create Certificate Request`; the `Create Certificate Request` dialog box appears.
2. Enter the following information (Table 5–4):

Table 5–4 Certificate Request: Fields and Descriptions

Field Name	Description
Common Name	Mandatory. Enter the name of the user's or service's identity. Enter a user's name in first name /last name format.
Organizational Unit	Optional. Enter the name of the identity's organizational unit. Example: Finance.
Organization	Optional.Enter the name of the identity's organization. Example: XYZ Corp.
Locality/City	Optional. Enter the name of the locality or city in which the identity resides.
State/Province	Optional. Enter the full name of the state or province in which the identity resides. Enter the full state name, because some certificate authorities do not accept two–letter abbreviations.
Country	Mandatory. Choose the drop-down list to view a list of country abbreviations. Select the country in which the organization is located.
Key Size	Mandatory. Choose the drop-down box to view a list of key sizes to use when creating the public/private key pair.
Advanced	Optional. Choose Advanced to view the Advanced Certificate Request dialog panel. Use this field to edit or customize the identity's distinguished name (DN). For example, you can edit the full state name and locality.

3. Choose OK. An Oracle Wallet Manager dialog box informs you that a certificate request was successfully created. You can either copy the certificate request text from the body of this dialog panel and paste it into an e-mail message to send to a certificate authority, or you can export the certificate request to a file.
4. Choose OK. You are returned to the Oracle Wallet Manager main window; the status of the certificate is changed to `Requested`.

Importing the User Certificate into the Wallet

You will receive an e-mail notification from the certificate authority informing you that your certificate request has been fulfilled. Import the certificate into a wallet in either of two ways: copy and paste the certificate from the e-mail you receive from the certificate authority, or import the user certificate from a file.

Pasting the Certificate

To paste the certificate:

1. Copy the certificate text from the e-mail message or file you receive from the certificate authority. Include the lines `Begin Certificate` and `End Certificate.`
2. Choose `Operations > Import User Certificate` from the menu bar; the Import Certificate dialog box appears.
3. Choose the `Paste the Certificate` button, and choose `OK`; an Import Certificate dialog box appears with the following message:

   `Please provide a base64 format certificate and paste it below.`
4. Paste the certificate into the dialog box, and choose `OK`. A message at the bottom of the window confirms that the certificate was successfully installed. You are returned to the Oracle Wallet Manager main panel, and the wallet status changes to `Ready`.

Selecting a File that Contains the Certificate

To select the file:

1. Choose `Operations > Import User Certificate` from the menu bar.
2. Choose the `Select a file...` certificate button, and choose `OK`; the Import Certificate dialog box appears.
3. Enter the path or folder name of the certificate location.
4. Select the name of the certificate file (for example, `cert.txt`).
5. Choose `OK`. A message at the bottom of the window appears, to inform you that the certificate was successfully installed. You are returned to the Oracle Wallet Manager main panel, and the wallet status is changes to `Ready`.

Removing a User Certificate from a Wallet

1. Choose `Operations > Remove User Certificate`; a dialog panel appears and prompts you to verify that you want to remove the user certificate from the wallet.
2. Choose `Yes`; you are returned to the Oracle Wallet Manager main panel, and the certificate displays a status of `Requested`.

Removing a Certificate Request

To remove a certificate request:

1. Choose `Certificate Request.`
2. Select menu item `Remove Certificate Request.`

Note: You must remove a certificate before removing its associated request.

Exporting a User Certificate

Save the certificate in a file system directory when you elect to export a certificate:

1. Choose `Operations > Export User Certificate` from the menu bar; the Export Certificate dialog box appears.
2. Enter the file system directory to save your certificate in, or navigate to the directory structure under Folders.
3. Enter a file name to save your certificate, in the Enter File Name field.
4. Choose `OK`. A message at the bottom of the window confirms that the certificate was successfully exported to the file. You are returned to the Oracle Wallet Manager main window.

Exporting a User Certificate Request

Save the certificate request in a file system directory when you elect to export a certificate request:

1. Choose `Operations > Export Certificate Request` from the menu bar; the Export Certificate Request dialog box appears.
2. Enter the file system directory in which you want o save your certificate request, or navigate to the directory structure under Folders.
3. Enter a file name to save your certificate request, in the Enter File Name field.
4. Choose `OK`. A message at the bottom of the window confirms that the certificate request was successfully exported to the file. You are returned to the Oracle Wallet Manager main window.

Managing Trusted Certificates

Managing trusted certificates includes the following tasks:

- Importing a Trusted Certificate
- Removing a Trusted Certificate
- Exporting a Trusted Certificate
- Exporting All Trusted Certificates
- Exporting a Wallet

Importing a Trusted Certificate

You can import a trusted certificate into a wallet in either of two ways: paste the trusted certificate from an e-mail that you receive from the certificate authority, or import the trusted certificate from a file.

Oracle Wallet Manager automatically installs trusted certificates from VeriSign, RSA, Entrust, and GTE CyberTrust when you create a new wallet.

Pasting the Trusted Certificate To paste the trusted certificate:

1. Choose `Operations > Import Trusted Certificate` from the menu bar; the Import Trusted Certificate dialog panel appears.
2. Choose the `Paste the Certificate` button, and choose `OK`. An Import Trusted Certificate dialog panel appears with the following message:

   ```
   Please provide a base64 format certificate and paste it below.
   ```
3. Copy the trusted certificate from the body of the e-mail message you received that contained the user certificate. Include the lines `Begin Certificate` and `End Certificate.`
4. Paste the certificate into the window, and Choose `OK`. A message at the bottom of the window informs you that the trusted certificate was successfully installed.
5. Choose `OK`; you are returned to the Oracle Wallet Manager main panel, and the trusted certificate appears at the bottom of the Trusted Certificates tree.

Selecting a File that Contains the Trusted Certificate

To select the file:

1. Choose `Operations > Import Trusted Certificate` from the menu bar. The Import Trusted Certificate dialog panel appears.
2. Enter the path or folder name of the trusted certificate location.
3. Select the name of the trusted certificate file (for example, `cert.txt`).
4. Choose `OK`. A message at the bottom of the window informs you that the trusted certificate was successfully imported into the wallet.
5. Choose `OK` to exit the dialog panel; you are returned to the Oracle Wallet Manager main panel, and the trusted certificate appears at the bottom of the Trusted Certificates tree.

Removing a Trusted Certificate

To remove a trusted certificate from a wallet:

1. Select the trusted certificate listed in the Trusted Certificates tree.
2. Choose `Operations > Remove Trusted Certificate` from the menu bar.

   A dialog panel warns you that your user certificate will no longer be verifiable by its recipients if you remove the trusted certificate that was used to sign it.
3. Choose `Yes`; the selected trusted certificate is removed from the Trusted Certificates tree.

Note: A certificate that is signed by a trusted certificate is no longer verifiable when you remove it from your wallet.

Also, you cannot remove a trusted certificate if it has been used to sign a user certificate that is still present in the wallet. To remove such a trusted certificate, you must first remove the certificates that it has signed.

Exporting a Trusted Certificate

To export a trusted certificate to another file system location:

1. Select `Operations > Export Trusted Certificate`; the Export Trusted Certificate dialog box appears.
2. Select a file system directory to save your trusted certificate, or choose `Browse` to display the directory structure.
3. Enter a file name to save your trusted certificate.
4. Choose `OK`; you are returned to the Oracle Wallet Manager main window.

Exporting All Trusted Certificates

To export all of your trusted certificates to another file system location:

1. Choose `Operations > Export All Trusted Certificates`. The Export Trusted Certificate dialog box appears.
2. Select the file system directory to save your trusted certificates, or choose `Browse` to display the directory structure.
3. Enter a file name to save your trusted certificates.
4. Choose `OK`; you are returned to the Oracle Wallet Manager main window.

Exporting a Wallet

You can export a wallet to text-based PKI formats. Individual components are formatted according to the following standards (Table 5–5):

Table 5–5 PKI Wallet Encoding Standards

Component	**Encoding Standard**
Certificate chains	X509v3
Trusted certificates	X509v3
Private keys	PKCS #8

6 Configuring Oracle9*i*AS Portal Security

One of the most important aspects of any portal solution is security. The ability to control user access to Web content and to protect your site against people breaking into your system is critical. This chapter describes the architecture of Oracle9*i*AS Portal security in the following topics:

Portal Security Model

When you make content available on the Web, it is very likely that you need to restrict access to at least some parts of it. For example, it is unlikely that you want every user to be able to see every document on your site. It is even less likely that you want every user to be able to change every document on your site. Oracle9*i*AS Portal provides a comprehensive security model that enables you to completely control what users can see and change on your Web site.

Before a user logs on to Oracle9*i*AS Portal, they can only view the content that the content contributors designate as public. Public content can be viewed by any user who knows the URL of a portal object (for example, a page) and can connect to the machine where it is stored. The user sees only those aspects of the object that are designated as public, such as the public portlets. If the object has no public contents, then the user is denied access to it.

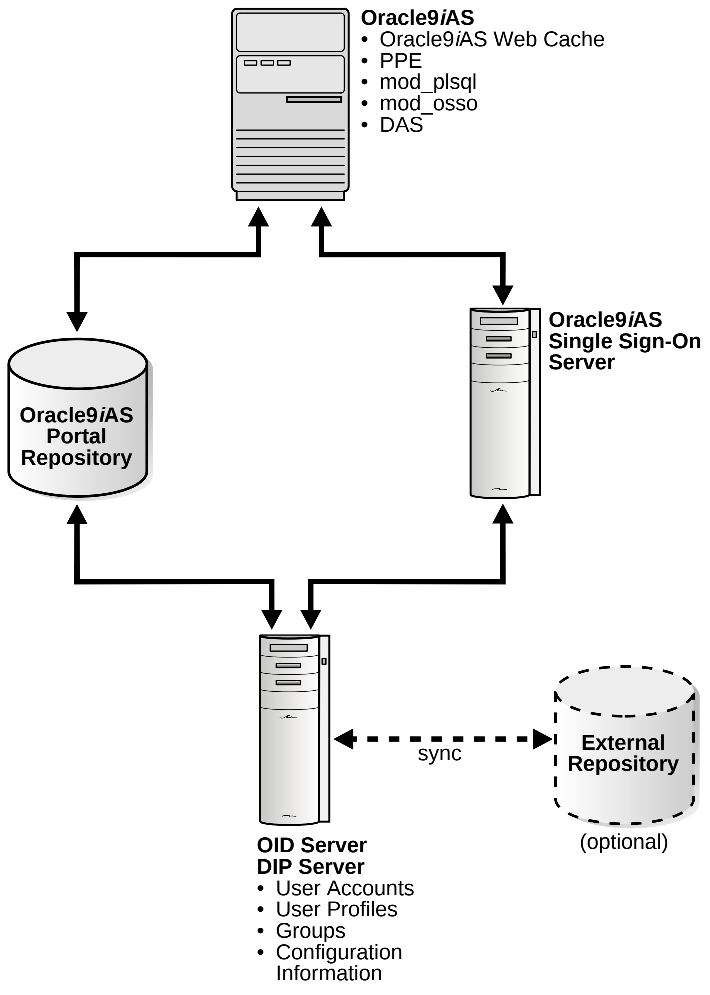
Once the user logs in to the portal, they may or may not be able to see and change content depending upon their access privileges. Typically, an authenticated user can see and do more in the portal than a public user. For example, an authenticated user might see items or portlets on the page that the public user cannot view. An authenticated user might also be able to add and edit content, and change properties, privileges that would typically be denied to a public user. In the portal, you can control access to objects (pages, items, or portlets) by user and group. That is, you might grant access privileges for a page to specific named users, user groups, or a combination of both.

To support this flexible approach to controlling access to Web content, Oracle9*i*AS Portal leverages the other components of Oracle9*i*AS and Oracle9*i* to provide strong protection for your portal. Oracle9*i*AS Portal interacts with all of the following components to implement its security model:

- Oracle9*i*AS Single Sign-On (Single Sign-On server) authenticates users, who are attempting to gain access to non-public areas of your portal.
- mod_osso is an Oracle HTTP Server module that redirects authentication requests to Single Sign-On server.
- Oracle Internet Directory, Oracle's native LDAP version 3 service, acts as the repository for user credentials and group memberships.
- Delegated Administration Service adds or updates the information stored inside the directory (users and groups).
- Oracle Directory Integration Server notifies Oracle9*i*AS Portal upon the occurrence of any directory events (for example, user deletions) to which

Oracle9*i*AS Portal subscribes. In essence, the directory integration server informs Oracle9*i*AS Portal when a change occurs in the directory that requires a change in Oracle9*i*AS Portal.

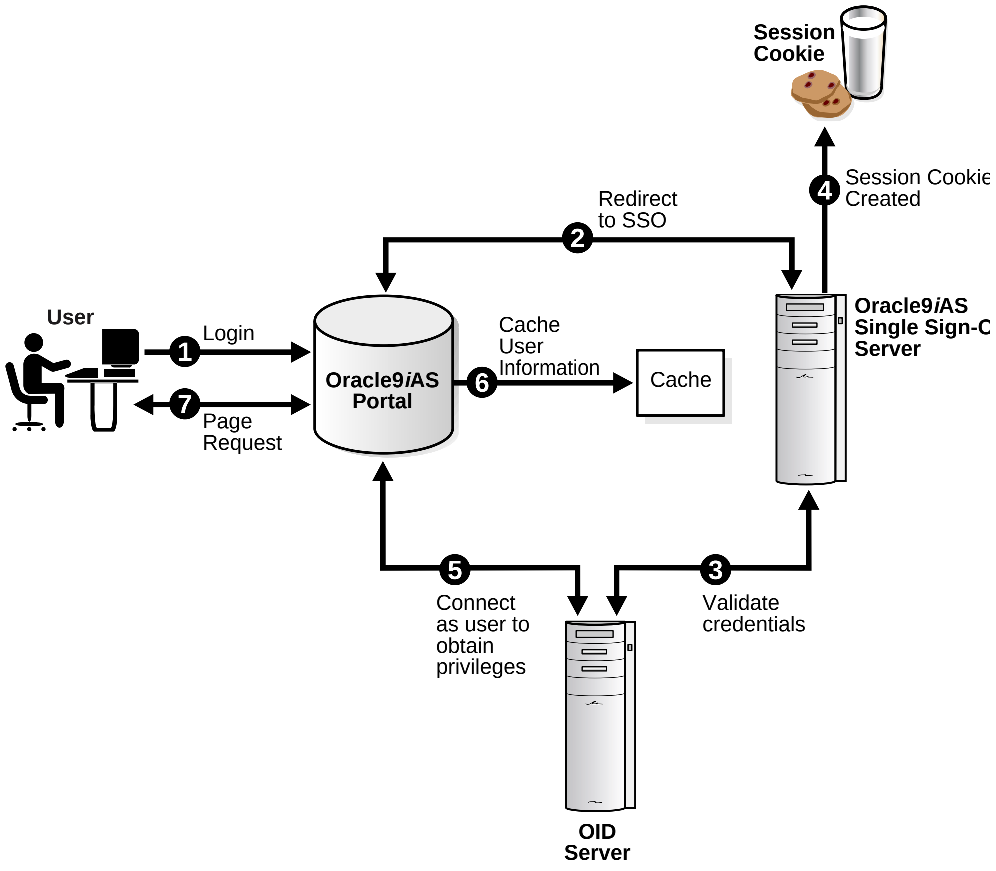
Figure 6–1 Oracle9iAS Portal Security Architecture



User Authentication and Privilege Model

When a user attempts to log in to Oracle9*i*AS Portal, their credentials must first be verified by Single Sign-On server against the directory. Once their identity has been verified, Oracle9*i*AS Portal checks their access privileges in the directory to determine which objects they may see and use within the portal. The figure and text below describe this model in more detail.

Figure 6–2 Oracle9iAS Portal Authentication Model



1. From Oracle9*i*AS Portal, the user requests to log in by clicking the Login link.
2. The login request is forwarded to Single Sign-On server for authentication.

3. Single Sign-On server verifies the user credentials against the information stored in the directory.
4. If authentication is successful, Single Sign-On server creates an SSO cookie for the user. If authentication is not successful, the user is denied access and returned to the login page to re-enter their user name and password.
5. Once the user’s identity has been verified, control is returned to Oracle9*i*AS Portal, which creates a portal session cookie. Oracle9iAS Portal then connects to the directory and determines the user's group memberships and privileges.
6. Oracle9*i*AS Portal caches the user's membership and privilege information locally for the duration of their session.
7. When the user attempts to access a page, Oracle9*i*AS Portal performs the following checks:
   - Checks whether the page is public. If so, the user can view it.
   - If the page is not public, Oracle9*i*AS Portal checks the local privilege table to determine whether the current user has privileges to view the page. If the user has viewing privileges, the user can view it.
   - If the current user does not have direct viewing privileges on the page, Oracle9*i*AS Portal checks the cached membership information and privilege table to determine whether any of the groups to which the user belongs has privileges to view the page. If one of the groups to which the user belongs has viewing privileges on the page, the user can view it.

   Note: If changes are made to Oracle Internet Directory that effect the user’s privileges, a notification is raised and the cached information about the user is invalidated. Thus, Oracle9*i*AS Portal starts enforcing the user’s updated privileges as soon as it receives the notification.

Portal Security Architecture

As you might expect, based upon the model described in the previous sections, Oracle9*i*AS Portal administrators need to understand all of the following:

- Relationship between Oracle9iAS Portal and Oracle9iAS Single Sign-On
- Relationship between Oracle9iAS Portal and Oracle Internet Directory
- Relationship between Oracle9iAS Portal and Delegated Administration Service
- Relationship between Oracle9iAS Portal and the Oracle Directory Integration Server

Relationship between Oracle9*i*AS Portal and Oracle9*i*AS Single Sign-On

Portal uses Oracle9*i*AS Single Sign-On for user authentication, as discussed in "User Authentication and Privilege Model" on page 6-4.

Note: Oracle9*i*AS Portal can only be associated with Oracle9*i*AS Single Sign-On Release 9.0. It cannot be used with older versions. Similarly, Oracle9*i*AS Single Sign-On Release 9.0 will not work with earlier versions of Oracle9*i*AS Portal.

The Oracle9*i*AS Single Sign-On manages the Single Sign-On sessions of users. In order for Single Sign-On security to function properly, Oracle9*i*AS Portal requires the following for integration with Oracle9*i*AS Single Sign-On:

- Oracle9*i*AS Portal must be added as a partner application for Oracle9*i*AS Single Sign-On.
- Oracle9*i*AS Portal entries must be added to the partner application enabler configuration table.

These two configuration steps are performed for you upon installation by the Oracle Universal Installer. If you need to make changes to your configuration after installation, you can do so by invoking `ptlasst.csh` (Unix) or `ptlasst.bat` (MS Windows). These scripts and their documentation are located in ORACLE_HOME/assistants, where ORACLE_HOME is the home directory for Oracle9*i* Application Server. To change the Oracle9*i*AS Single Sign-On settings for Oracle9*i*AS Portal, you must invoke these scripts with `-mode` set to one of the following:

- SSOCONFIG

- SSOPARTNERCONFIG
- MIDTIER

Relationship between Oracle9*i*AS Portal and Oracle Internet Directory

Oracle Internet Directory is Oracle's highly scalable, native LDAP version 3 service and hosts the Oracle common user identity. As stated in the previous section, Oracle9*i*AS Portal queries the directory to determine a user's privileges and what they are entitled to see and do in the portal. In particular, Oracle9*i*AS Portal retrieves the group memberships of the user from the directory to determine what they may access and change.

Given this model, Oracle9*i*AS Portal requires the following interactions with Oracle Internet Directory:

- Oracle9*i*AS Portal specific entries stored in the directory
- Group attributes stored in the directory
- User attributes stored in the directory
- Caching of user and group information from the directory
- Populating user and group lists of values from the directory through the Delegated Administration Service

Directory entries in Oracle Internet Directory for Oracle9*i*AS Portal In order for security to function properly, Oracle9*i*AS Portal requires the following entries in the directory's DIT structure:

- **Oracle9*i*AS Portal default user accounts** (cn=PUBLIC, cn=PORTAL, cn=PORTAL_ADMIN) are created in the subscriber's user base (cn=Users,o=MyCompany,dc=com[1]). The PORTAL and PORTAL_ADMIN users are added to the DBA and PORTAL_ADMINISTRATORS groups, respectively. The PUBLIC user is created for unauthenticated users. Typically, the PUBLIC user entry is for granting viewing privileges on portal content that is accessible to any user, unrestricted.
- **Oracle9*i*AS Portal group container** (cn=PORTAL_GROUPS) is created within the subscriber's group base (cn=Groups,o=MyCompany,dc=com[1]). Oracle9*i*AS Portal can leverage any group in the directory, but groups are more easily

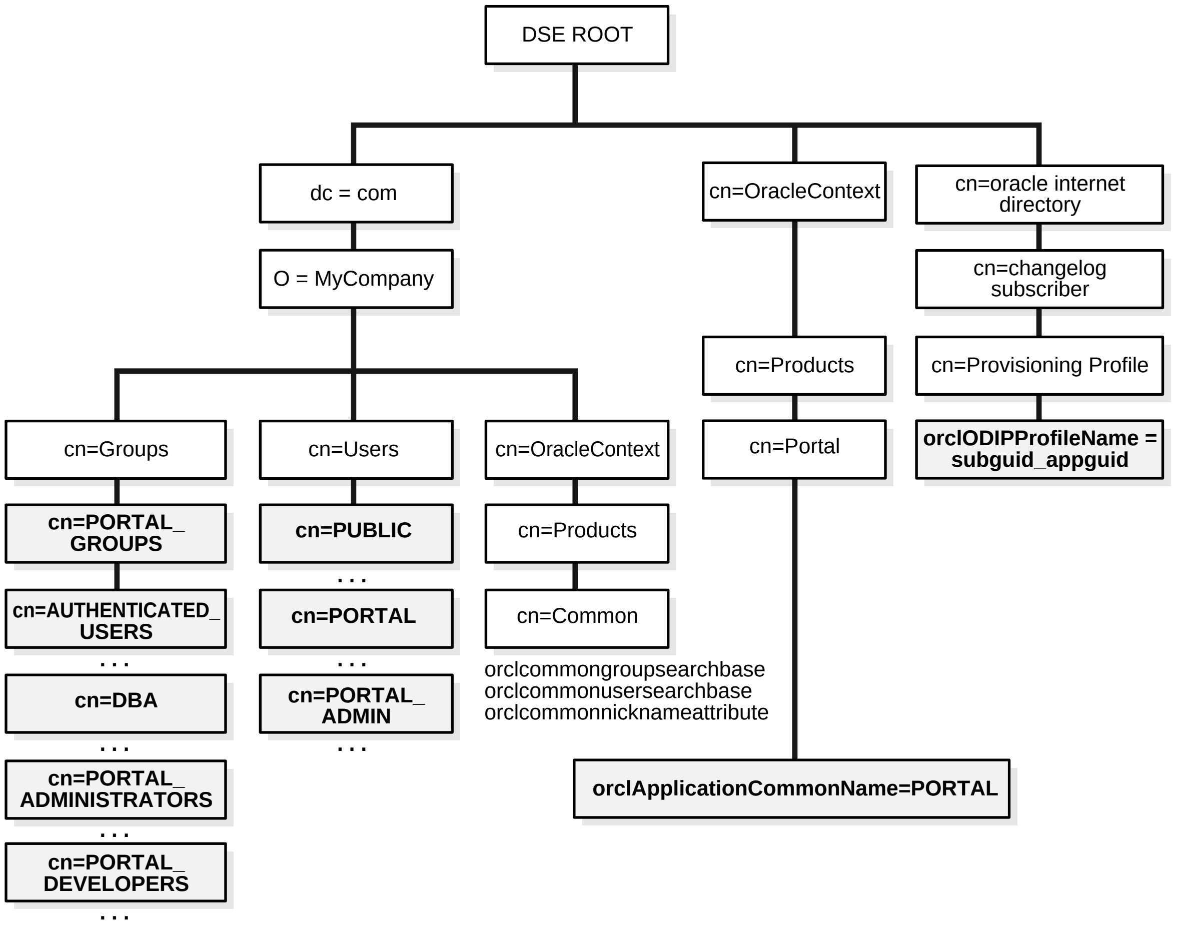
[1] The default subscriber name is determined by the domain name of the server on which the system is installed. For example, if the domain name server was oracle, the default subscriber name would be o=oracle,dc=com. If the domain name server cannot be determined, the default name assigned by the directory is o=Default Company,dc=com

accessed for display in a list of values if they are located within the Oracle9*i*AS Portal group container.

- **Oracle9*i*AS Portal groups** are created within the Oracle9*i*AS Portal group container in the directory:
  - cn=AUTHENTICATED_USERS
  - cn=DBA
  - cn=PORTAL_ADMINISTRATORS
  - cn=PORTAL__DEVELOPERS
  - cn=PORTLET_PUBLISHERS
  - cn=RW_ADMINISTRATOR
  - cn=RW_DEVELOPER
  - cn=RW_POWER_USER
  - cn=RW_BASIC_USER
- **Oracle9*i*AS Portal application entity** (orclApplicationCommonName=PORTAL) is created in Oracle Context (cn=Portal,cn=Products,cn=Oracle Context). Oracle9*i*AS Portal uses this entity to bind to the directory when it needs to query it or perform actions against it (for example, adding a user) on behalf of the user. When Oracle9*i*AS Portal binds to the directory for a user, it uses a proxy connection to connect as the user. This method ensures that the user's authorization restrictions are properly enforced by the directory. The Oracle9*i*AS Portal application entity obtains the privileges to initiate proxy connections by its membership in the user proxy privileges group (cn=UserProxyPrivilege,cn=Groups,cn=OracleContext).
- **Oracle9*i*AS Portal directory synchronization subscription** A provisioning profile entry is created in the provisioning profile of the directory (cn=Provisioning Profile,cn=changelog subscriber,cn=oracle internet directory). This entry indicates that the directory must notify Oracle9*i*AS Portal when user or group privilege information has changed. It enables Oracle9*i*AS Portal to keep its authorizations synchronized with the information stored in the directory. This registration is performed from the directory provisioning subscription tool.

The figure below shows where the Oracle9*i*AS Portal information is located in the directory's DIT structure.

Figure 6–3 DIT Structure for Oracle9iAS Portal



User attributes stored in Oracle Internet Directory Oracle9*i*AS Portal, like all other components of Oracle9*i* Application Server, relies upon the directory to store user information. All users in the directory are defined using the following object classes:

- The inetOrgPerson object class contains all of the user attributes defined by IETF (RFC 2798).
- The orclUser and orclUserV2 object classes contain a set of standard, additional attributes for Oracle products.

The tables below show the various user attributes stored in Oracle Internet Directory.

Table 6–1 inetOrgPerson Attributes

inetOrgPerson (IETF) attributes	Comment
Cn	The common name and default nickname of the user. This is the name the user logs in to Oracle9*i*AS Portal via Single Sign-On server. This attribute is mandatory.
EmployeeNumber	Number used to identify employees
Sn	Last name. This attribute is mandatory. If nothing is explicitly specified for this attribute, the user’s nickname (Cn by default) is used.
GivenName	First name
MiddleName	
DisplayName	Nickname
Mail	e-mail address
TelephoneNumber	
HomePhone	
Mobile	
Pager	
FacsimileTelephoneNumber	
Street	
L	City of office
St	State of office
PostalCode	Postal code of office
C	Country of office
HomePostalAddress	Home address
JpegPhoto	Person’s picture
O	Organization
Title	

Table 6–1 inetOrgPerson Attributes (Cont.)

inetOrgPerson (IETF) attributes	Comment
Manager	Employee’s supervisor
UserPassword	
PreferredLanguage	

Table 6–2 orclUserV2 Attributes

orclUserv2 attributes	Comments
orclIsVisible	A flag to indicate whether the user should be hidden from all but administrators.
orclDisplayPersonalInfo	A flag to indicate whether a user’s personal information should be hidden from all but administrators.
orclMaidenName	
orclDateOfBirth	
orclHireDate	
orcleDefaultProfileGroup	Default user group for the person
orclActiveStartDate	When account was activated
orclActiveEndDate	when account was (or will be) terminated
orclTimeZone	
orclIsEnabled	A flag to indicate whether the user account is active. If not active, the user will not be allowed to log in.

For users who are familiar with the user properties from previous versions of Oracle9*i*AS Portal, the following table maps the old user properties to the new Oracle Internet Directory attributes.

Table 6–3 Mapping of Oracle9iAS Portal User Properties to Oracle Internet Directory

Previous Oracle9*i*AS Portal user property	inetOrgPerson or orclUserv2 attributes
ID	Not applicable because ID remains a local Oracle9*i*AS Portal attribute that is linked to the corresponding directory entry by means of a globally unique identifier.
EMPNO	EmployeeNumber
LAST_NAME	Sn
FIRST_NAME	GivenName
MIDDLE_NAME	MiddleName
KNOWN_AS	DisplayName
EMAIL	Mail
WORK_PHONE	TelephoneNumber
HOME_PHONE	HomePhone
MOBILE_PHONE	Mobile
PAGER	Pager
FAX	FacsimileTelephoneNumber
OFFICE_ADDR(1,2,3)	Street
OFFICE_CITY	L
OFFICE_STATE	St
OFFICE_ZIP	PostalCode
OFFICE_COUNTRY	C
HOME_ADDR[1,2,3],CITY, STATE,ZIP,COUNTRY	HomePostalAddress
IMAGE	JpegPhoto
ORGANIZATION	O
TITLE	Title
MANAGER	Manager
PASSWORD	UserPassword

Table 6–3 Mapping of Oracle9iAS Portal User Properties to Oracle Internet Directory

Previous Oracle9*i*AS Portal user property	inetOrgPerson or orclUserv2 attributes
DISPLAY	orclIsVisible
DISPLAY_PERSONAL_INFO	orclDisplayPersonalInfo
NOTIFICATION_PREFERENCE	orclWorkflowNotificationPref
USER_NAME	orclCommonNickNameAttribute, which is the nickname used in place of the user's full Dn. The full Dn attribute is quite long (cn=name,o=domain,dc=com), hence it is simpler for users to log in with this nickname. For more information, refer to the documentation on Oracle Internet Directory.
MAIDEN_NAME	orclMaidenName
DATE_OF_BIRTH	orclDateOfBirth
HIREDATE	orclHireDate
SUBSCRIBER_ID	Not applicable because the subscriber identifier is obtained from the user's subscriber node.
DEFAULT_GROUP	orcleDefaultProfileGroup

Group attributes stored in Oracle Internet Directory Oracle9*i*AS Portal, like all other components of Oracle9*i* Application Server, relies upon the directory to store group information. All groups in the directory are defined using the following object classes:

- The groupOfUniqueNames object class contains all of the group attributes defined by IETF (RFC 2798).
- The orclGroup object class contains a set of standard, additional attributes for Oracle9*i*AS Portal.

Note: Unlike Oracle9*i*AS Portal Release 3.x, you cannot scope groups in Release 9.0 to a specific page group.

The tables below show the various group attributes stored in Oracle Internet Directory:

Table 6–4 groupOfUniqueNames Attributes

groupOfUniqueNames (IETF) attributes	Comment
Cn	The common name of the group, which can be typed into places like the Edit Group field in the Group portlet to locate the group.
Description	The text description of the group, which is displayed in lists of values where the group appears.
uniqueMember	A list of the distinguished names (DNs) of all of the members of the group. The member DNs can represent a user or another group.
Owner	A list of the DNs of all of the users and groups that have the privilege of administering this group.

Table 6–5 orclGroup Attributes

orclGroup attributes	Comment
orclGUID	The globally unique identifier (GUID) for this group.
orclIsVisible	A flag to indicate whether the group is public or private. Private groups only appear in lists of values for their owners. Other users cannot see them.

For users who are familiar with the group properties from previous versions of Oracle9*i*AS Portal, the following table maps the old user properties to the new Oracle Internet Directory attributes:

Table 6–6 Mapping of Oracle9iAS Portal Group Properties to Oracle Internet Directory

Previous Oracle9*i*AS Portal group property	groupOfUniqueNames or orclGroup attribute
ID	local ID for the group, which can be matched to the orclGUID in the directory by a new locally stored orclGUID.
HIDDEN_GROUP	orclIsVisible
SUBSCRIBER_ID	Subscriber id is no longer needed because the location of the group entry under a subscriber base indicates the subscriber.
NAME	Cn
DESCRIPTION	Description
group membership	uniqueMember
OWNER	Owner

Oracle Internet Directory Cache in Oracle9*i*AS Portal To improve performance, Oracle9*i*AS Portal caches some directory information locally. In particular, Oracle9*i*AS Portal caches the following:

- Directory connection information for Oracle9*i*AS Portal
- URLs for Delegated Administration Service
- orclGUIDs of certain privilege groups for authorization checks on directory portlets (for example, the User and Group portlets)
- some Oracle Context information
- the locally selected group search and creation bases
- group memberships and default group for each user

The majority of the information cached by Oracle9*i*AS Portal is fairly static (for example, directory connection information). For those items that are more dynamic, such as group memberships and default group, Oracle9*i*AS Portal relies upon the Directory Synchronized Provisioning agent for updates. Oracle9*i*AS Portal maintains a directory synchronization subscription in the directory that flags the agent to notify it of any change events that effect Oracle9*i*AS Portal security (for example, adding or deleting a user from a group).

User and Group Lists of Values in Oracle9*i*AS Portal The User, Group, Portal User Profile, and Portal Group Profile portlets include lists of values for users or groups. These lists of values must be populated with information stored in the directory.

If you have your directory and Oracle9*i*AS Portal servers residing in different domains, you must explicitly set the JavaScript domain for Oracle9*i*AS Portal such that it can resolve user and group lists of values. For example, suppose that your installation has Oracle9*i*AS Portal configured to use a different Oracle HTTP Server than the Delegated Administration Service. In this situation, you need to have a common domain so that the values can be transferred from the list of values displayed by the Delegated Administration Service to the page displayed by Oracle9*i*AS Portal.

To create a single domain in this case, do the following:

1. Login to SQL*Plus as PORTAL.
2. Run the following SQL script:

   ```
   secjsdom.sql <domain_name>
   ```

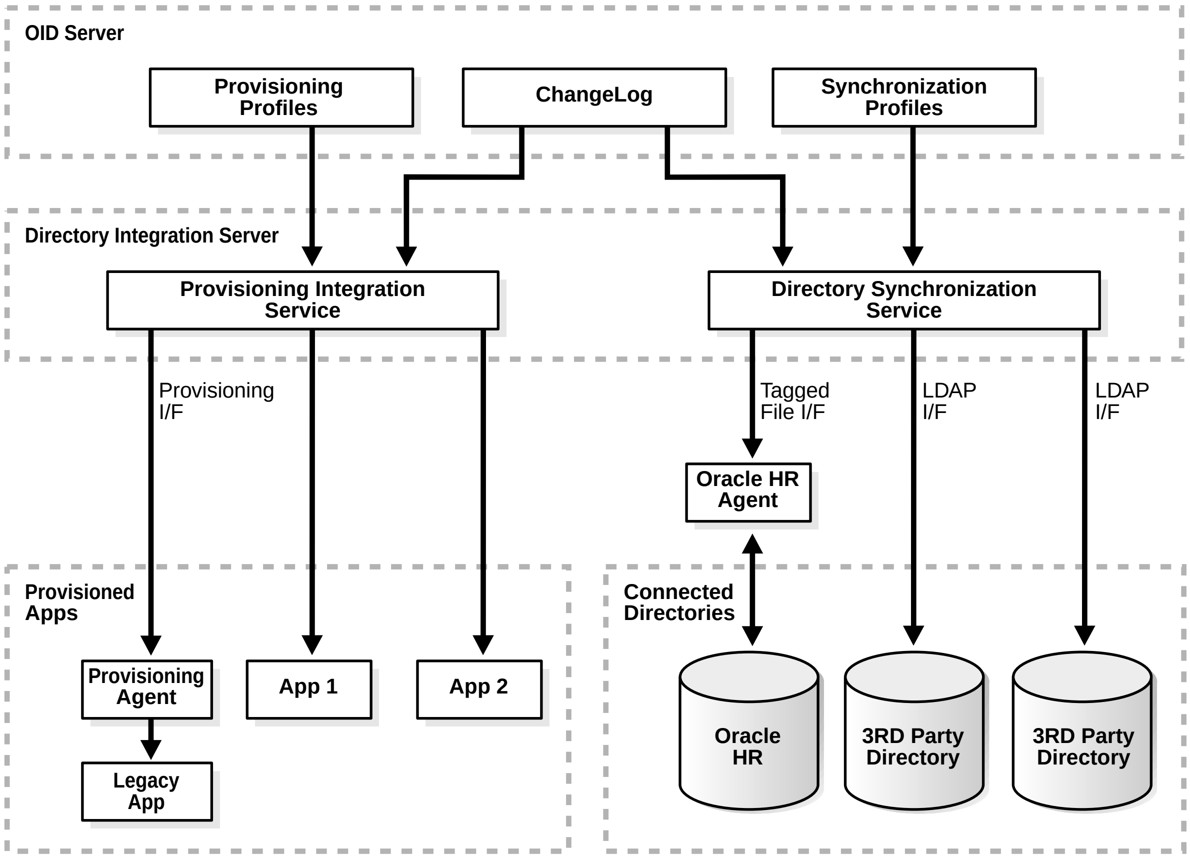
Performing this procedure enables you to run directory lists of values from Oracle9*i*AS Portal in either Netscape or MicroSoft Internet Explorer. When using lists of values, a transit window is displayed in addition to the list of values itself. The transit window is required to pass values to Oracle9*i*AS Portal without forcing pages to reset their domain.

> **See Also:** "Group Search Base Distinguished Name (DN)" on page 6-40 for information about choosing where Oracle9*i*AS Portal searches for groups.

Relationship between Oracle9*i*AS Portal and the Oracle Directory Integration Server

Directory synchronized provisioning is a service provided by Oracle Directory Integration Server to notify components of user and group change events in the directory. The figure below illustrates how the directory integration server keeps components synchronized with the latest information in the directory.

Figure 6–4 Oracle Directory Integration Server Synchronization Model



Components, such as Oracle9*i*AS Portal, subscribe to provisioning events (for example, deletion of a group) in order to keep their local caches of user and group information synchronized with the central user and group repository in the directory. When a change event occurs, all of the components that are subscribed to that change event are notified by the Directory Synchronized Provisioning agent of the directory integration server. Oracle9*i*AS Portal sets the Portal directory synchronization subscription flag in the directory to indicate that it should be notified whenever a subscribed change event takes place. The table below shows the events to which Oracle9*i*AS Portal subscribes and the actions it takes when those events occur:

Table 6–7 DIrectory Synchronized Events Handled by Oracle9iAS Portal

Subscribed event	Oracle9*i*AS Portal action
USER DELETE	The local user profile entry is deleted, resulting in the deletion of the user's privileges. Pages associated with this user are invalidated in Oracle9*i*AS Web Cache.
USER MODIFY (orclDefaultProfileGroup)	The default group of the user is changed in the local user profile.
GROUP DELETE	The local group profile is deleted, resulting in the deletion of the privileges assigned to this group. The WWSEC_FLAT$ table is updated accordingly.
GROUP MODIFY (uniqueMember)	The WWSEC_FLAT$ table is updated to reflect membership changes that effect Oracle9*i*AS Portal. If the membership changes involve a group being added or deleted from the modified group, the pages associated with the users of the added or deleted group are invalidated in Oracle9*i*AS Web Cache. The reason for this action is that the security changes might effect what is visible on the page or the access privileges of the page itself.

Note: Oracle9*i*AS Portal does not need to subscribe to user and group creation events. The local user profile is created automatically when a new user first logs on or is assigned some privilege that causes the user to be referenced in an access control list of Oracle9*i*AS Portal. Similarly, a local group profile is created automatically when a new group is first referenced in an access control list.

Relationship between Oracle9*i*AS Portal and Delegated Administration Service

In addition to querying the directory for user and group information, Oracle9*i*AS Portal must provide users with the means to add and modify user and group information. To change information in the directory, use the Delegated Administration Service. Oracle9*i*AS Portal provides links to the delegated administration server for users with the privileges to add and change users and groups.

Creating and updating information stored in Oracle Internet Directory The Delegated Administration Service provides a comprehensive interface for making updates to the directory. Authenticated users who have the appropriate privileges can access the delegated administration server through the User and Group portlets on the Administration tab in Oracle9*i*AS Portal. To access these portlets, a user must be a member of the OracleDASCreateUser and OracleDASCreateGroup groups, respectively. The PORTAL and PORTAL_ADMIN users are members of both of these groups by default. AUTHENTICATED_USERS may also create groups by default.

Relationship between Delegated Administration Service, mod_osso, and the Oracle9*i*AS Single Sign-On Server When users access the delegated administration server, they do so through mod_osso for authentication. If they are successfully authenticated and have the appropriate privileges, they can access the delegated administration server.

Creating Users and Groups

As stated in the previous section, the most common way to create users and groups for your portal is through the User and Group portlets on the Administration page of Oracle9*i*AS Portal. Furthermore, you can set global privileges and preferences for portal users and groups via the Portal User Profile and Portal Group Profile portlets.

You must be a member of one of the following groups to access the User, Group, Portal User Profile, and Portal Group Profile portlets:

- PORTAL_ADMINISTRATORS
- DBA

If you are not a member of one of these groups, then you must be a member of the following privilege groups:

- OracleDASCreateUser
- OracleDASCreateGroup

The directory access control policy on the directory information tree provides members of these privilege groups with access to the User, Group, Portal User Profile, and Portal Group Profile portlets.

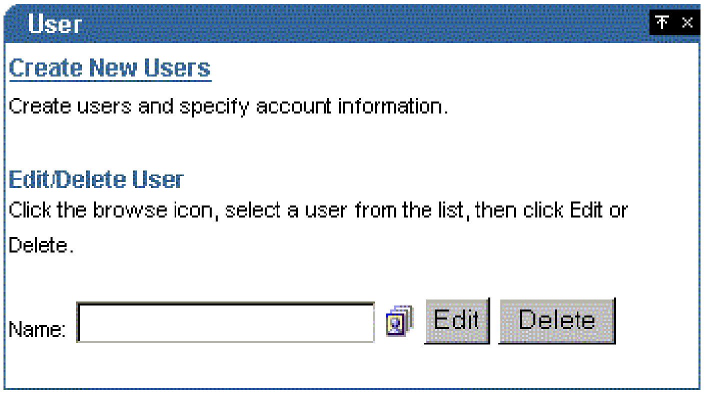
User Portlet

The User portlet on the Administration tab enables you to create and update users through Delegated Administration Service. To create a new user, click the Create User link in the User portlet. To update information for an existing user, type their user name in the Name field or choose it from the list of values and click Edit. To delete a user, type their user name in the Name field or choose it from the list of values and click Delete.

See Also: The following documentation for detailed information on the settings that are available through the User portlet:

- *Oracle9iAS Portal Online Help*
- *Oracle Internet Directory Administrator's Guide* in the Oracle9*i*AS Documentation Library for information about Delegated Administration Service

Figure 6–5 User Portlet



User

Create New Users

Create users and specify account information.

Edit/Delete User

Click the browse icon, select a user from the list, then click Edit or Delete.

Name:

Portal User Profile Portlet

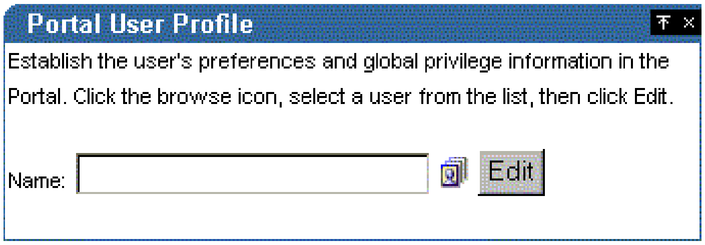
To set global user privileges and preferences that pertain specifically to the portal, use the Portal User Profile portlet. To update a user's portal preferences and privileges, type their user name in the Name field or choose it from the list of values. You can set all of the following for the user's profile:

- Preferences
  - whether the user can access the portal

  - database schema name for the user
  - whether the user has a personal page
  - default user group for the user
  - default home page for the user
  - whether to clear the Oracle9*i*AS Web Cache for the user
- Global Privileges
  - page group privileges
  - application privileges
  - administration privileges

See Also: *Oracle9i*AS Portal *Online Help* for the Edit User Profile pages for more information on these settings.

Figure 6–6 Portal User Profile Portlet


Portal User Profile

Establish the user's preferences and global privilege information in the Portal. Click the browse icon, select a user from the list, then click Edit.

Name:

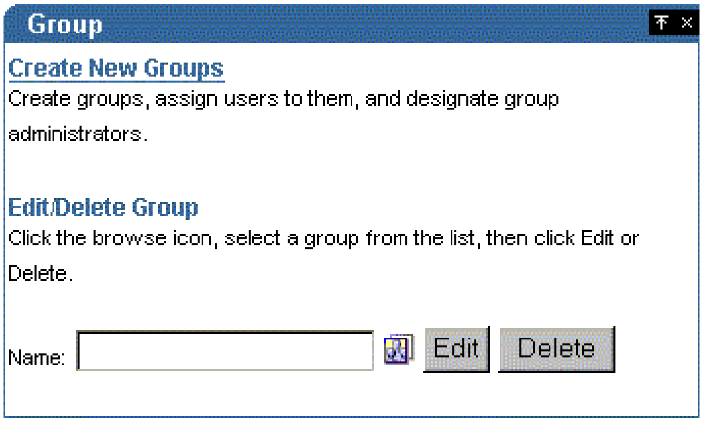
Group Portlet

The Group portlet on the Administration tab enables you to create and update user groups through Delegated Administration Service. To create a new group, click the Create Group link in the Group portlet. To update information for an existing group, type its name in the Name field or choose it from the list of values and click Edit. To delete a group, type the group name in the Name field or choose it from the list of values and click Delete.

See Also: Detailed information on the Group portlet settings in the following documentation:

- *Oracle9iAS Portal Online Help*
- *Oracle Internet Directory Administrator's Guide* in the Oracle9*i*AS Documentation Library for information about Delegated Administration Service

Figure 6–7 Group Portlet



Create New Groups

Create groups, assign users to them, and designate group administrators.

Edit/Delete Group

Click the browse icon, select a group from the list, then click Edit or Delete.

Name:

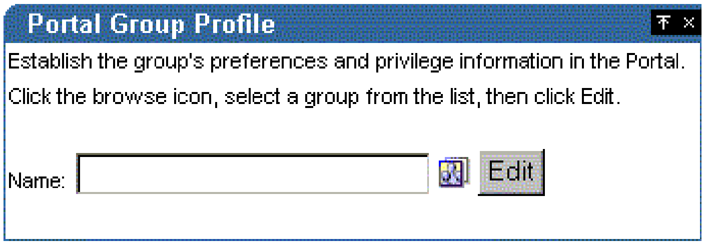
Portal Group Profile Portlet

To set global group preferences and privileges that pertain specifically to the portal, you need to use the Portal Group Profile portlet. To update a group's portal preferences and privileges, type the group name in the Name field or choose it from the list of values. You can set all of the following for the group's profile:

- Preferences
  - default home page for the group
- Global Privileges
  - page group privileges
  - application privileges
  - administration privileges

See Also: *Oracle9i*AS Portal *Online Help* for the Edit Group Profiles pages for more information about these settings.

Figure 6–8 Portal Group Profile Portlet



Portal Group Profile

Establish the group's preferences and privilege information in the Portal.

Click the browse icon, select a group from the list, then click Edit.

Name:

Granting Access Privileges

Within Oracle9*i*AS Portal, you decide at what level of granularity you want to control access. You can assign privileges to any object on a per user or per group basis. For example, you can assign access privileges on a per user basis for each and every item in your portal, but this approach creates considerable overhead for your content contributors.

If you want to lessen the burden on contributors, then you can assign privileges on a per group basis at the page level and simply ensure that all of the items that you place on any given page have similar security requirements. With this approach, the security that items receive through the page that contains them is usually sufficient and content contributors only need to assign privileges for items that require higher security than the page.

Access Tab

You can assign access privileges to users or groups for all of the following objects within Oracle9*i*AS Portal through the Access tab of the object's Edit Page:

Table 6–8 Oracle9iAS Portal Objects with Privilege Control

Type of Object	Available Privileges	Inherited Privileges
Calendar	■ Manage ■ View ■ Customize ■ Execute	From Database Provider
Chart (based on SQL query)	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
Chart (based on wizard)	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
Data Component	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
Data Component Cell	■ Edit ■ View	From Data Component
Database Provider	■ Manage ■ Edit ■ View Source ■ Customize ■ Execute	Not applicable

Table 6–8 Oracle9iAS Portal Objects with Privilege Control (Cont.)

Type of Object	Available Privileges	Inherited Privileges
Document	■ Own ■ Manage ■ View Only	From page or item
Dynamic Page Component	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
Form[1]	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
Frame Driver	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
Hierarchy	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
Image Chart	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider

Table 6–8 Oracle9iAS Portal Objects with Privilege Control (Cont.)

Type of Object	Available Privileges	Inherited Privileges
Link	▪ Manage ▪ Edit ▪ View ▪ Customize ▪ Execute	From Database Provider
List of Values	▪ Manage ▪ Edit ▪ View ▪ Customize ▪ Execute	From Database Provider
Menu	▪ Manage ▪ Edit ▪ View ▪ Customize ▪ Execute	From Database Provider
Oracle9*i* Reports printer	▪ Manage ▪ Edit ▪ View ▪ Execute	From Database Provider
Oracle9*i* Reports report	▪ Manage ▪ Edit ▪ View ▪ Customize ▪ Execute	From Database Provider
Oracle9*i* Reports Server	▪ Manage ▪ Edit ▪ View ▪ Execute	From Database Provider

Table 6–8 Oracle9iAS Portal Objects with Privilege Control (Cont.)

Type of Object	Available Privileges	Inherited Privileges
Page	■ Manage ■ Manage Content ■ Manage Items With Approval ■ Manage Style ■ Customize Portlets (Full) ■ Customize Portlets (Add-Only) ■ Customize Portlets (Hide-Show) ■ Customization(Style) ■ View	Not applicable
Page group	■ Manage All ■ Manage Classifications ■ Manage Templates ■ Manage Styles ■ View	Not applicable
Page Item	■ Own ■ Manage ■ View Only	From page
Portlet	■ Manage ■ Edit ■ Execute ■ Access ■ Publish	Not applicable
Provider	■ Manage ■ Edit ■ Publish ■ Execute	Not applicable

Table 6–8 Oracle9iAS Portal Objects with Privilege Control (Cont.)

Type of Object	Available Privileges	Inherited Privileges
Query by example form	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
Report[2]	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
Schema	■ Manage ■ Modify ■ Insert ■ View	Not applicable
URL	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider
XML	■ Manage ■ Edit ■ View ■ Customize ■ Execute	From Database Provider

[1] You can have many different types of forms (stored procedure or table based, version 2 or version 3 based, and master-detail), but all of these types have the same available privileges and privilege inheritance.

[2] You can have two different types of reports (SQL and table based), but all of these types have the same available privileges and privilege inheritance.

See Also: *Oracle9i*AS Portal *Online Help* for more information on the available privileges for each of these objects.

Administering Portal Security

To effectively administer Oracle9*i*AS Portal security, you must decide where you will install the portal components, understand the default security settings, complete the security checklist, and understand how to change the Oracle Internet Directory settings. Each of these tasks is described in the following sections:

- Security Settings Upon Installation
- Post-Installation Security Checklist
- Changing LDAP Settings on the Global Settings Page

Security Settings Upon Installation

Before you can begin to administer Oracle9*i*AS Portal, you must first understand the default settings that are created during installation.

Oracle9*i*AS Portal Default Schemas and Accounts

The tables that follow describe the schemas, users, and groups that are created by default when Oracle9*i*AS Portal is installed.

Table 6–9 Default Oracle9iAS Portal Schemas

Schema	Description
`PORTAL`	Contains the Portal product database objects and code. This schema also represents the proxy user account that mod_plsql uses to connect to the database through the credentials provided in the corresponding DAD. To execute Web requested procedures, mod_plsql then uses N-Tier authentication to connect to the schema to which the lightweight user accounts are assigned (by default, PORTAL_PUBLIC). The default name for this schema in a standard Oracle9*i*AS Portal installation is PORTAL. If you want to give it another name, you must perform a custom installation.

Table 6–9 Default Oracle9iAS Portal Schemas (Cont.)

Schema	Description
`PORTAL_PUBLIC`	Is the schema that all lightweight users are mapped to by default. All procedures publicly accessible through the Web are granted execute to PUBLIC, which makes them accessible through this schema. In a standard Oracle9*i*AS Portal installation, this schema is named PORTAL_PUBLIC. If you want to give it another name, you must perform a custom installation.
`PORTAL_DEMO`	Is created to hold some demonstration code. The installation of this schema is optional.

See Also:

- "Securing Application Database Access Through mod_plsql" on page 9-7
- *mod_plsql User's Guide* in the Oracle9*i*AS Documentation Library

For more information about mod_plsql and how to use it.

Table 6–10 Default Oracle9iAS Portal Users

User	Description
`PUBLIC`	Is the user account that identifies unauthenticated access to the Oracle9*i*AS Portal. Once a user logs in, the user name changes from PUBLIC to the user name by which the user authenticated herself. When granting Portal privileges on individual objects which do not have an explicit checkbox for granting the object to Public, this user can be identified as the grantee of the privilege to grant access to it for unauthenticated users.
`PORTAL`	Is the super-user for the portal. In a standard installation, the user name is PORTAL. This user account has the highest privileges because it is granted all the global privileges available in the portal.

Table 6–10 Default Oracle9iAS Portal Users (Cont.)

User	Description
`PORTAL_ADMIN`	Is a privileged Oracle9*i*AS Portal user account with administrative privileges excluding those that would give the user the ability to obtain higher privileges or perform any database operations. This user cannot edit any group or manage privileges on any schema or shared object. This account is typically intended for an administrator who manages pages and provisions user accounts.

Table 6–11 Default Oracle9iAS Portal Groups

Group[1]	Description
AUTHENTICATED_USERS	Is the group that includes any authenticated, or logged in, user. The purpose of this group is to provide a means to assign the default privileges you want every logged in user to have in the portal. Hence, this group is initialized with all of the privileges that you want to grant to the least privileged user who logs in to the portal.
DBA	Is a highly privileged group established for Oracle9*i* Application Server administrators. Components that are part of Oracle9*i* Application Server grant full component-specific privileges to members of this group. The DBA group is a member of the PORTAL_ADMINISTRATORS group.
PORTAL_ADMINISTRATORS	Is a highly privileged group established for Oracle9*i*AS Portal. The PORTAL_ADMINISTRATORS group is a member of the IASADMINS group. Thus, members of PORTAL_ADMINISTRATORS can, by default, administer Oracle9*i*AS Single Sign-On (just as they could in earlier versions of Oracle9*i*AS Portal).
PORTLET_PUBLISHERS	Is a privileged group established for users who need to publish portlets to other users of the portal.
PORTAL_DEVELOPERS	Is a privileged group established for users who are building portlets.
RW_ADMINISTRATOR	Is the group of users who administer Oracle9*i* Reports reports, printers, and servers
RW_DEVELOPER	Is the group of users who develop Oracle9*i* Reports reports

Table 6–11 Default Oracle9iAS Portal Groups (Cont.)

Group[1]	Description
RW_POWER_USER	Is the group of users who can modify Oracle9*i* Reports reports
RW_BASIC_USER	Is the group of users who use Oracle9*i* Reports reports

[1] All groups shown in this table are located in cn=PORTAL_GROUPS,cn=Groups,o=MyCompany,dc=com. Note that subscriber name is determined by the domain name of the server on which the system is installed. For example, if the domain name server was oracle, the default subscriber name would be o=oracle,dc=com. If the domain name server cannot be determined, Oracle Internet Directory defaults to the domain specified during installation by the administrator.

Post-Installation Security Checklist

After Oracle9*i*AS Portal is installed, you should consider performing the following steps to complete the security configuration:

- Configure mod_plsql Settings
- Safeguard Passwords for Lightweight Oracle9iAS Portal Users
- Remove Unnecessary Objects
- Revoke Public Access to Provider Components
- Control Access to Administration Pages
- Protect Oracle9iAS Portal Monitoring Packages
- Consider SSL and the Login Portlet
- Consider LDAP over SSL for Oracle Internet Directory Connections
- Change the Application Entity Password

Configure mod_plsql Settings

The mod_plsql settings are configured in Oracle Enterprise Manager, which can be accessed from Oracle9*i*AS Portal as follows:

1. From the Oracle9*i*AS Portal Design-Time Pages page, click the **Administer** tab.
2. In the Services portlet, click **Portal Service Monitoring**.
3. Go to the mod_plsql page.
4. Change the password of the corresponding database user for the `PORTAL` DAD.

5. Delete all DADs that you do not need. For example `SAMPLE_DAD` is unnecessary.
6. Add a new DAD for the portal you are building, and set the default home page for this DAD. For example:

   ```
   <hostname>.<some_domain>.com/<home_page>/<page_name>.htm
   ```

7. Make the new DAD the default DAD. This redirects the browser to the DAD when the following URL is entered:

   ```
   http://<hostname>:<port>/pls
   ```

> **See Also:** Oracle9*i* Application Server mod_plsql User's Guide to find more information about configuring mod_plsql settings.

Safeguard Passwords for Lightweight Oracle9*i*AS Portal Users

Unscrupulous users might try to learn the passwords of your default users, which could result in an account lock. This lock can be released from the server, but it is far better that you not depend on the default user accounts for administrative purposes. To safeguard the passwords for these accounts do the following:

1. Immediately change the default passwords for all of the following default users:
   - `PORTAL`
   - `PORTAL_ADMIN`
   - `PUBLIC`
2. Create new lightweight administrator accounts with the same access rights as the default users, and set the account termination date in Single Sign-On server for the default users. Alternatively, you can also uncheck the Allow User To Log In setting in the Edit User page for the default users.
3. Once you have disabled login or changed the passwords for the default users, try logging in to the portal as the default users with the default passwords to ensure that your changes have been successful.

Remove Unnecessary Objects

In order to prevent users from entering your portal through obsolete or unnecessary pages, you should remove any unused objects from your Oracle9*i*AS Portal and database environment. For example:

- Delete page groups that are no longer in use.
- Delete Oracle9*i*AS Portal providers that are no longer in use.

Revoke Public Access to Provider Components

In some cases, Oracle9*i*AS Portal provider components may give users the option to view or modify records in application tables. To tighten security, you should revoke public access from such components if it is unnecessary. You can also use a menu component with specific access rights on the menu options to more tightly control application access.

Control Access to Administration Pages

In order to prevent users who should not have access to administration interfaces from entering administration pages, you should ensure that you control access rights for the following page groups and the pages they contain:

- Corporate Pages is the page group that contains the Oracle9*i*AS Portal Home Page.
- Portal Design-Time Pages is the page group that contains the Builder and Navigator pages.
- Portlet Repository
- Documentation is the page group that contains the Oracle9*i*AS Portal online help.

To control access to the above page groups, perform the following steps:

1. In the Navigator, click **Page Groups**.
2. Click **Edit Properties** next to the page group for which you want to change the access settings.
3. Click the **Access** tab.
4. Grant `MANAGE ALL` to specific users or to certain groups. For example, `DBA`, `PORTAL_ADMINISTRATORS`, `PORTAL_DEVELOPERS`, and your own groups.
5. When you are done, click **OK**.

To control access to individual administration pages in these page groups, perform the following steps:

1. In the Navigator, click **Page Groups**.

2. Click **Contents** next to the page group that contains the pages on which you want to change the access settings.
3. Click **Pages**.
4. Click **Properties** next to the page for which you to change the access settings.
5. Click the **Access** tab.
6. Grant `MANAGE ALL` to specific users or to certain groups. For example, `DBA`, `PORTAL_ADMINISTRATORS`, `PORTAL_DEVELOPERS`, and your own groups.
7. When you are done, click **OK**.

Note: The Builder page is the root page of the Portal Design-Time Pages page group. In order to alter its access settings, you must click **Edit Root Page** next to the Portal Design-Time page group.

Protect Oracle9*i*AS Portal Monitoring Packages

You must protect the execution of PL/SQL procedures granted to `PUBLIC` in the database. These procedures pose a security hole when they are executed through a Web browser. For example, some procedures in the `dbms_%` packages allow access to sensitive information. You can specify which packages to protect with the `PlsqlExclusionList` directive in the mod_plsql configuration file called `dads.conf`.

See Also: "Protecting the PL/SQL Procedures Granted to PUBLIC" on page 9-11 for information about how to use the `PlsqlExclusionList` directive in `dads.conf`.

In relation to Oracle9*i*AS Portal, `PUBLIC` access to the monitoring packages can expose pages that contain sensitive information or degrade system performance because of heavy queries to the portal database. To resolve this, specify some or all of the following packages with the `PlsqlExclusionList` directive in the `dads.conf` file:

- `WWMON_CHART_BY_ACTION.show`
- `WWMON_CHART_BY_BROWSER.show`
- `WWMON_CHART_BY_DATE.show`
- `WWMON_CHART_BY_IPADDRESS.show`
- `WWMON_CHART_BY_LANGUAGE.show`

- `WWMON_CHART_BY_OBJECT.show`
- `WWMON_CHART_BY_ROWS.show`
- `WWMON_CHART_BY_TIME.show`
- `WWMON_CHART_BY_USER.show`
- `WWMON_CHART_SEARCHES.show`

To ensure the best security, specify the following system default settings with the `PlsqlExclusionList` directive for each DAD:

```
PlsqlExclusionList sys.*
PlsqlExclusionList dbms_*
PlsqlExclusionList utl_*
PlsqlExclusionList owa_util.*
PlsqlExclusionList portal.wwmon_*
```

To test your changes, try to access the following URL without logging in first:

```
http://<hostname>:<port>/pls/<dad>/portal30.WWMON_CHART_BY_USER.show
```

This attempt should result in an HTTP 404 error: “It is forbidden to call this procedure from the browser!”

Note: To run the Oracle9*i*AS Portal monitor charts, create an extra secretly named DAD with a less restrictive `PlsqlExclusionList`.

Consider SSL and the Login Portlet

To secure passwords going across the Internet you can use Secure Sockets Layer (SSL) communications by configuring Oracle9*i*AS Portal to run in HTTPS. However, to enable or disable SSL, you must have portal administrator privileges.

Login Portlet Versus Login Page Portal has the option to place a login portlet on a page (typically the home page). This portlet shows user name and password fields and a login button when the user is not logged in. If the user is logged in, it shows a logout link. This portlet provides an easy way to log in without having to navigate to a dedicated login page. It also displays in the Oracle9*i*AS Portal page layout style.

However, if you use this portlet, you must ensure that the pages on which it appears are SSL-encrypted. Bear in mind, that SSL encryption for your complete site could adversely affect performance because it requires more resources.

Furthermore, the login portlet presents a security risk because you cannot prevent showing the login screen since it is shown when the user is not logged in. Hence, in situations where you want SSL encryption on passwords, you should not use the login portlet when you want SSL encryption. To enforce this restriction, you must remove access rights for the login portlet in the Portlet Repository.

> **See Also:** *Oracle9iAS Portal Configuration Guide* for a detailed description on how to enable SSL within Oracle9*i*AS Portal.

Consider LDAP over SSL for Oracle Internet Directory Connections

By default, Oracle9*i*AS Portal connects to the directory using LDAP without SSL. If the directory server is configured for an SSL port, though, Oracle9*i*AS Portal can be configured to use LDAP over SSL, also known as LDAPS.

> **See Also:** *Oracle Internet Directory Administrator's Guide* in the Oracle9*i*AS Documentation Library for a detailed description on how to configure the directory for an LDAP port over SSL.

To configure Oracle9*i*AS Portal to use SSL to connect to the directory, you must run the `secupoid.sql` script. This script allows you to change the following Oracle9*i*AS Portal configuration parameters related to the directory:

- Directory host name
- Directory port
- Application directory password
- SSL setting

When you install Oracle9*i*AS Portal, it is automatically associated with a directory server. However, you may want to change some settings, such as whether to use SSL, after installation. To change to an SSL connection for the directory, simply run the `secupoid.sql` script in the PORTAL schema to specify the LDAPS port instead of the LDAP port, and indicate that you want to use SSL.

Running the secupoid.sql script The section that follows illustrates a sample execution of `secupoid.sql` from SQL*Plus.

In the example, the directory was initially configured to run LDAP on port 389. Later, an LDAPS port was activated on 636. Since the server name does not change, we retain the old value, update the port, and indicate that we want to use SSL by setting the Use SSL? value to Y. When you run the script, it displays the current configuration and lets you replace any of the configurable settings. The script also allows you to update Oracle9*i*AS Portal’s directory cache after running it. Since activating SSL does not change any of the directory information cached by Oracle9*i*AS Portal, it is not usually necessary to refresh the cache in this case.

```
SQL> @secupoid
Current Configuration
--------------------
OID Host: oid.domain.com
OID Port: 389
Application DN:
orclApplicationCommonName=PORTAL,cn=Portal,cn=Products,cn=OracleContext
Application Password: 3E8C2D1B87CB61011757239C5AA9B390
Use SSL? N

PL/SQL procedure successfully completed.

Updating OID Configuration Entries
Press [Enter] to retain the current value for each parameter
For SSL Connection to LDAP, specify "Y"es or "N"o
------------------------------------------------
Enter value for oid_host:
Enter value for oid_port: 636
Enter value for app_password:
Enter value for use_ssl_to_connect_to_ldap: Y
Enter value for refresh_with_new_settings: N

PL/SQL procedure successfully completed.

No errors.
```

After executing the script, Oracle9*i*AS Portal is configured for LDAPS access of the directory.

Change the Application Entity Password

Oracle9*i*AS Portal never passes a user's password to the directory. Only Oracle9*i*AS Single Sign-On performs that task. However, Oracle9*i*AS Portal authenticates itself to the directory through its application entity and password. This account can proxy as any user because it has proxy privileges and thus it is also an account that ought to be protected.

If you want to change the application entity's password, you need to first change its entry in the directory, using command line utilities or the Oracle Directory Manager. To locate the application entry in the directory, you need its DN, which is reported by the `secupoid.sql` script. By default, Oracle9*i*AS Portal's application entry is:

```
orclApplicationCommonName=PORTAL,cn=Portal,cn=Products,cn=OracleContext
```

To change the password, you set the userPassword attribute for the application entry to the new password.

After you have changed the password in the directory, you run `upsecoid.sql` script in the PORTAL schema and specify the new password there, too. Running the script enables Oracle9*i*AS Portal to encrypt the password and store it for retrieval when it needs to connect to the directory.

See Also: Directory entries in Oracle Internet Directory for Oracle9iAS Portal on page 6-7 for more information about the application entity.

Changing LDAP Settings on the Global Settings Page

Once you have installed Oracle9*i*AS Portal and performed the appropriate tasks from "Post-Installation Security Checklist" on page 6-32, your Oracle9*i*AS Portal configuration is secure. From the Global Settings page of Oracle9*i*AS Portal, you can now change all of the following settings that pertain to security:

- Cache for Oracle Internet Directory Parameters
- Oracle Directory Integration Server Synchronization
- Group Search Base Distinguished Name (DN)
- Group creation base DN

Cache for Oracle Internet Directory Parameters

As pointed out in "Portal Security Architecture" on page 6-6, Oracle9*i*AS Portal maintains a cache of information from the directory. From the Global Settings Page, you can refresh this cache with the updated information from the directory. Refresh Cache for the directory parameters immediately updates the cache with the latest parameters values from the directory. The cached information is relatively static information, hence you do not need to refresh the cache frequently.

Oracle Directory Integration Server Synchronization

Because Oracle9*i*AS Portal caches group membership information, it requires a mechanism for updating the cache when the information is changed in the directory. The directory integration server notifies Oracle9*i*AS Portal whenever a change is made in the directory that must be reflected in Oracle9*i*AS Portal. In Global Settings, you can set:

- **Enable directory synchronization** defines whether the directory integration server notifies Oracle9*i*AS Portal when a relevant change is made in the directory. If this setting is not checked, then Oracle9*i*AS Portal will not be notified of any directory integration server subscribed events.
- **Send event notifications every n seconds** defines the interval of time between event notifications sent by the directory integration server to notify Oracle9*i*AS Portal of relevant changes. This setting is available only when Enable directory synchronization is checked.

Group creation base DN

Oracle9*i*AS Portal maintains its user group information in the directory. When groups are created through the Groups portlet, they are created under a node of the LDAP Directory Information Tree (DIT). A node is identified by its distinguished name (DN). Therefore, in Oracle9*i*AS Portal, you need to specify in which node you wish to create groups:

Group Creation Base DN is the DN of the node in which you want Oracle9*i*AS Portal to maintain its user groups. For example:

```
cn=PORTAL_GROUPS,cn=Groups,o=MyCompany,dc=com
```

This setting is particularly useful if you adapt Oracle9*i*AS Portal to interact with an existing DIT.

Group Search Base Distinguished Name (DN)

Just as you need to define the node in which you want to create groups, you must also define the node in which you want Oracle9*i*AS Portal to search for existing groups. For example, you need to specify where Oracle9*i*AS Portal searches when it displays the groups list of values in the Groups portlet.

Local Group Search Base DN is the DN of the node in which you want Oracle9*i*AS Portal to maintain its user groups. For example:

```
cn=PORTAL_GROUPS,cn=Groups,o=MyCompany,dc=com
```

This setting is particularly useful if you adapt Oracle9*i*AS Portal to interact with an existing DIT.

7

Configuring JAAS Support

This chapter describes the configuration tasks you must perform to use JAAS support in a Java2 Platform, Standard Edition (J2SE) or Java2 Platform, Enterprise Edition (J2EE) environment.

This chapter contains these topics:

Note: This chapter does not describe how to configure Oracle9*i*AS Containers for J2EE (OC4J) with your application. See the *Oracle9iAS Containers for J2EE User's Guide* in the Oracle9*i*AS Documentation Libraryfor those instructions.

See Also: Part 2 of *OC4J Services Guide* in the Oracle9*i*AS Documentation Library for information on using JAAS support after completing these configuration tasks.

What JAAS Components Do You Need to Configure?

You must configure the JAAS components after installation and before using your JAAS-based application. The JAAS components that must be configured depend on the environment in which the application runs. Table 7–1 identifies the necessary configuration tasks.

Table 7–1 JAAS Component Configuration Tasks

For Applications in...	Follow These Configuration Tasks...
J2SE Environments	■ Performing Configuration Tasks Common to J2SE and J2EE Environments on page 7-4 ■ Performing Configuration Tasks Unique to J2SE Environments on page 7-13
J2EE Environments	■ Performing Configuration Tasks Common to J2SE and J2EE Environments on page 7-4 ■ Performing Configuration Tasks Unique to J2EE Environments on page 7-16

Sample Files

Configuration tasks in this chapter require that you either create or edit certain configuration or deployment descriptor files. To make this task easier, sample configuration files are provided. Copy and edit these files as described in this chapter with values appropriate to your development or runtime environment.

These files are located in the `$ORACLE_HOME/j2ee/home/config` directory.

- `jazn.xml`

  JAAS property file
- `jazn-data.xml`

  Default JAAS datafile for using an XML-based provider
- `java2.policy`

  The Java2 policy file

  The following are application-specific deployment descriptor files you may need to modify:
- `orion-application.xml`
- `orion-web.xml`
- `web.xml`

Performing Configuration Tasks Common to J2SE and J2EE Environments

J2SE and J2EE environments require several similar configuration tasks. Complete these tasks to configure JAAS components for applications developed in J2SE and J2EE environments.

- Task 1: Ensure That You Installed the Correct Components
- Task 2: Load the JAZN Schema and Default Entries into Oracle Internet Directory (Optional)
- Task 3: Specify JAAS as the Policy Provider (optional)
- Task 4: Configure a Java2 Policy File (optional)
- Task 5: Create a LoginModule Configuration File (optional)

  If your application runs in the J2EE environment, then perform this task only if you do not need your application to be SSO-enabled.
- Task 6: Perform Configuration Tasks Unique to Your Java Environment

Task 1: Ensure That You Installed the Correct Components

Table 7–2 identifies required components for J2SE and J2EE environments. Ensure that you have installed the correct components for your environment.

Table 7–2 Components Required for J2SE and J2EE Environments

	Component	Required For J2SE?	Required For J2EE?
1.	Java Authentication and Authorization Service (JAAS)	Yes	Yes
2.	JDK 1.3	Yes	Yes
3.	JAAS support, which includes `JAZNUserManager`[1] and a provider for storing realms, users, roles, and policy. Depending on your 9*i*AS installation type, you have either the default provider type or a choice of two types:	Yes	Yes
■	XML-based provider type, the default, available with both installation types, Oracle9*i*AS Infrastructure and Oracle9*i* Application Server		
■	LDAP-based provider type, Oracle Internet Directory, available with Oracle9*i*AS Infrastructure installation type only. If Oracle Internet Directory is used, this guide assumes that you have already: Installed Oracle Internet Directory; Run Oracle Internet Directory Configuration Assistant to load Oracle Internet Directory schema into the directory; Created users and roles with the Delegated Administration Service (DAS)		
4.	Oracle9*i*AS Containers for J2EE (OC4J)	No	Yes
5.	Oracle9*i*AS Single Sign-On (for SSO-enabled applications). This guide assumes that you have already: Installed Oracle9*i*AS Single Sign-On; Loaded required database objects into a repository	No	Yes[2]
6.	`RealmLoginModule` or other login module. Authenticates the user name and password of a client user attempting to access an application. The `RealmLoginModule` is part of Oracle's proprietary Realm API package (`oracle.security.jazn.realm`). `RealmLoginModule` can be used in non-SSO environments.	No	No
7.	Oracle HTTP Server, which includes:	No	Yes
■	mod_oc4j	No	Yes
■	mod_osso (for SSO-enabled applications)	No	Yes
■	mod_ossl (for SSL-enabled applications)	No	Yes

[1] `JAZNUserManager` functionality is provided for J2EE environments only.
[2] If you want your applications to be SSO-enabled in J2EE environments, use Oracle9*i*AS Single Sign-On.

See Also: *Oracle9i Application Server Installation Guide* in the Oracle9*i*AS Platform-specific Documentation for installation and postinstallation tasks

Task 2: Load the JAZN Schema and Default Entries into Oracle Internet Directory (Optional)

These configuration steps pertain only to LDAP-based environments. If you decide to use the LDAP-based Oracle Internet Directory as your provider type, then you must run the scripts described in this section. These scripts load the JAZN schema and default entries.

Note: Use JAAS support with the version of Oracle Internet Directory available through the Oracle9*i*AS Infrastructure installation type of Oracle9*i* Application Server release 2 (9.0.2). Older versions of Oracle Internet Directory must first be upgraded to this version in order to work with JAAS.

If you have installed Oracle9*i*AS Infrastructure, then the following steps 1 through 4 are unnecessary because these configuration steps are performed when the infrastructure is installed. However, step 5 to load the demo data only is still required.

If you have not installed the infrastructure and want to use Oracle Internet Directory as your provider type, then you must perform steps 1 through 5. In this case, use the script in step 5 that loads both the JAZN schema and the demo data.

To load the JAZN schema and default entries into Oracle Internet Directory:

1. Obtain the following information about Oracle Internet Directory:

Element	Description
`ldaphost`	Host name of the computer on which Oracle Internet Directory is installed
`ldapport`	Port number for Oracle Internet Directory (default value is `389`)

Element	Description
`binddn`	Bind distinguished name; the default value is `cn=orcladmin`
`passwd`	Bind password; the default password is `welcome`
`OracleContextDN`	Distinguished name of the Oracle site context

This information is used to configure JAAS support with Oracle Internet Directory in steps 4 through 5.

2. Go to the computer on which JAAS support is installed.
3. Go to the `$ORACLE_HOME/jazn/install` directory.
4. Run the `generateldif.sh` script to generate the `.ldif` files that are required by JAAS. These LDIF (LDAP Data Interchange Format) files are used to add the JAZN LDAP schema and demo data in the directory. Use the following syntax:

   ```
   generateldif.sh -b cn=OracleContextDN
   ```

   where *`OracleContextDN`* is the distinguished name of the Oracle site context.
5. If you are running a version of Oracle Internet Directory that is not available with Oracle9*i*AS Infrastructure release 2 (9.0.2), you must first upgrade Oracle Internet Directory to this version. After that, run the following scripts to load the JAZN schema and demo data using the `.ldif` files generated in step 4:

   To load the demo data only, run the script with the following arguments:

   ```
   postinstall.sh -h ldaphost -p ldapport -D binddn -w password
   ```

   To load everything (the JAZN schema and the demo data), run the script with the following arguments:

   ```
   postinstall.sh -h ldaphost -p ldapport -D binddn -w password -load All
   ```

Task 3: Specify JAAS as the Policy Provider (optional)

You must specify JAAS as the policy provider for JAAS if it has not been configured automatically. Configuration is automatic if you use the JVM shipped with Oracle9*i* Application Server.

To specify JAAS as the policy provider:

1. Add the following information to the end of the `$JAVA_HOME/jre/lib/security/java.security` file:

   ```
   auth.policy.provider=oracle.security.jazn.spi.PolicyProvider
   login.configuration.provider=oracle.security.jazn.spi.LoginConfigProvider
   ```

Task 4: Configure a Java2 Policy File (optional)

Note: This task is required only when the Java Security Manager is enabled.

The Java2 policy file grants permissions to the trusted codes or applications that you run. This enables these codes or applications to access Oracle support for JAAS, JAAS, or JDK APIs requiring specific access privileges.

A preconfigured Java2 policy (`java2.policy`) has been provided in `$ORACLE_HOME/j2ee/home/config` with grants necessary to launch OC4J with `SecurityManager` enabled.

You need to modify the Java2 policy file to grant permissions to trusted codes or applications.

For example, the following section of a `java2.policy` file grants `java.security.AllPermission` to the trusted codes `jazn.jar` and an application named `appdemo_runtime.jar` running in the `$ORACLE_HOME/appdemo` directory:

```
/* grant the JAZN library AllPermission */
grant codebase "file:/Oracle/OraHome/lib/jazn.jar" {
    permission java.security.AllPermission;
};

/* Assuming you are running your application demo in $ORACLE_HOME/appdemo/,
*/
/* Grant JAZN permissions to the demo to run JAZN APIs*/
grant codebase "file:/${oracle.ons.oraclehome}/appdemo/-" {
   permission oracle.security.jazn.JAZNPermission "getPolicy";
   permission oracle.security.jazn.JAZNPermission "getRealmManager";
   permission oracle.security.jazn.policy.AdminPermission
"oracle.security.jazn.realm.RealmPermission$*$createRealm,dropRealm,
        createRole, dropRole,modifyRealmMetaData";
```

Task 5: Create a LoginModule Configuration File (optional)

Oracle support for JAAS fully complies with the J2EE JAAS specification so users can plug in any LoginModule implementation, if desired. RealmLoginModule is just one implementation that is included with JAAS for Oracle9*i*AS.

The `jazn-data.xml` file enables the `RealmLoginModule` class. The `RealmLoginModule` class authenticates user login credentials before the user can access:

- J2SE applications
- J2EE applications that do not use the SSO feature

  If you want to use SSO authentication with your application, do not perform this task. See "Performing Configuration Tasks Unique to J2EE Environments" on page 7-16 for SSO configuration tasks.

To enable the `RealmLoginModule` class, perform the following step:

Use a text editor to modify the login configuration file `jazn-data.xml` where needed.

The default configuration for the `RealmLoginModule` class setting in the `jazn-data.xml` file is as follows:

```
<!DOCTYPE jazn-data (View Source for full doctype...)>
- <jazn-data>
    .
    .
    .
<!--  Login Module Data -->
- <jazn-loginconfig>
 - <application>
     <name>JAZNUserManager</name>
   - <login-modules>
     - <login-module>
         <class>oracle.security.jazn.realm.RealmLoginModule</class>
         <control-flag>required</control-flag>
       - <options>
         - <option>
             <name>addRoles</name>
             <value>true</value>
           </option>
         </options>
       </login-module>
     </login-modules>
```

```
  </application>
 </jazn-loginconfig>
</jazn-data>
```

See Also: The JAAS Provider API Reference (**Javadoc**) is located in the Oracle9*i*AS Documentation Library

Task 6: Perform Configuration Tasks Unique to Your Java Environment

The configuration tasks that must be performed next depend on the environment in which the application runs, as indicated in Table 7–3.

Table 7–3 Configuration Tasks Unique to Your Java Environment

For...	Go To...
J2SE Environments	"Performing Configuration Tasks Unique to J2SE Environments" on page 7-13
J2EE Environments	"Performing Configuration Tasks Unique to J2EE Environments" on page 7-16

Performing Configuration Tasks Unique to J2SE Environments

Perform these configuration tasks after the configuration tasks described in "Performing Configuration Tasks Common to J2SE and J2EE Environments" on page 7-4.

Task 1: Configure the JAAS Property File

Configure the JAAS property file, `jazn.xml`, in the `$ORACLE_HOME/j2ee/home/config` directory, according to the provider environment type being used:

- Task 1a: Configure the LDAP-Based Provider Type for J2SE
- Task 1b: Configure the XML-Based Provider Type for J2SE

Task 1a: Configure the LDAP-Based Provider Type for J2SE

If you enable Oracle9*i*AS Single Sign-On by installing the Oracle9*i*AS Infrastructure, which installs and automatically configures the Single Sign-On server, Oracle Internet Directory, and Oracle Enterprise Manager, then you do not need to perform this task. The following steps for manually configuring the LDAP-based provider type are included in the following section for your reference only. These steps can be performed if you need to configure another LDAP-based provider.

1. Modify the preconfigured `jazn.xml` file using the following example:

```
<jazn provider="LDAP" location="ldap://orclcomp-sun.us.oracle.com:389">
</jazn>
```

   For this example, `orclcomp-sun.us.oracle.com` is the LDAP-based URL being used.

   Additional attributes and property names can also be set. `jazn.xml` permits the following attribute settings:

Attribute	Status	Value
`provider`	Optional	`LDAP` (default is XML)
`location`	Required	An LDAP server. For example: `ldap://orclcomp-sun.us.oracle.com:389`

`jazn.xml` permits the following property name settings. If you want to permit anonymous, read-only logins to the application, do not set and assign values to these property names.

Property Name	Status	Value
`ldap.user`	Optional	A valid LDAP user name or DN. For example: `orcladmin` or `cn=orcladmin`
`ldap.password`	Optional	An obfuscated password for the LDAP user name. For example: `QJ+w7NJUlm=`
`ldap.cache.enable`	Optional	Setting this property name to `"true"` enables an LDAP cache feature that is implemented based on a simple TTL (time-to-live) expiration algorithm. When it is enabled, calling `Policy.grant/revoke` results in an `UnsupportedOperationException`. If you need to manage the JAZN policy, then you must disable caching (the default setting).

The following example shows a `jazn.xml` file with all attributes and property names specified.

```
<jazn   provider="LDAP"
        location="ldap://orclcomp-sun.us.oracle.com:389" >
        <property name="ldap.user" value="orcladmin" />
        <property name="ldap.password" value="QJ+w7NJUlm=" />
        <property name="ldap.cache.enable" value="true" />
</jazn>
```

Note: If you do not want to obfuscate the password entry in `jazn.xml`, place an exclamation point in front of the password value (`!`). For example:

```
<property name="ldap.password" value="!welcome" />
```

Task 1b: Configure the XML-Based Provider Type for J2SE

The `jazn.xml` file is preconfigured as follows:

```
<jazn provider="XML" location="./jazn-data.xml" />
```

Additional attributes and property names can also be set. `jazn.xml` permits the following attribute settings:

Attribute	Status	Values
`provider`	Optional	■ `XML` (Default) ■ `LDAP`
`location`	Required	Path to file. For example: `./jazn-data.xml` This can be an absolute path, or a path relative to the `jazn.xml` file, where the JAAS provider first looks for the `jazn-data.xml` in the directory containing the `jazn.xml` file.
`persistence`	Optional	■ `NONE` Changes do not persist ■ `ALL` Changes persist after every modification ■ `VM_EXIT` (Default) Changes persist when VM exits
`xml.credentials.auto.obfuscate`	Optional	■ `ON` ■ OFF

The following example shows a `jazn.xml` file with all attributes specified.

```
<jazn   provider="XML"
        location="./jazn-data.xml"
        persistence="ALL"
        xml.credentials.auto.obfuscate="ON">
</jazn>
```

Performing Configuration Tasks Unique to J2EE Environments

Perform these configuration tasks after the configuration tasks described in "Performing Configuration Tasks Common to J2SE and J2EE Environments" on page 7-4.

- Task 1: Configure the JAAS Provider and Enable the JAZNUserManager
- Task 2: Configure an Authentication Method and Filter Modes
- Task 3: Configure Your Application for SSL Environments
- Task 4: Configure mod_oc4j to Delegate HTTP Requests to OC4J
- Task 5: Configure the Security Role (run-as)

Task 1: Configure the JAAS Provider and Enable the JAZNUserManager

Configure the JAAS and enable the `JAZNUserManager` through the OC4J-specific configuration file, `orion-application.xml`. Indicate the JAAS environment type and related information:

- Task 1a: Configure the LDAP-Based Provider Type for J2EE (Optional)
- Task 1b: Configure the XML-Based Provider Type for J2EE

Task 1a: Configure the LDAP-Based Provider Type for J2EE (Optional)

If you enable Oracle9*i*AS Single Sign-On by installing the Oracle9*i*AS Infrastructure, which installs and automatically configures the Single Sign-On server, Oracle Internet Directory, and Oracle Enterprise Manager, then you do not need to perform this task. The following steps for manually configuring the LDAP-based provider type are included in the following section for your reference only. These steps can be performed if you need to configure another LDAP-based provider.

Specifying the `default-realm` in `orion-application.xml` is necessary if there is more than one realm registered. In a hosted environment, where Oracle9*i*AS Single Sign-On and Oracle Internet Directory are enabled, specifying the default realm is optional. In this situation, JAAS reads the subscriber information from the HTTP header (the attributes that are set by mod_osso). These header attributes provide JAAS with the realm information, which represents the subscriber.

Configure the JAAS to use LDAP-based Oracle Internet Directory by adding an entry to the `orion-application.xml` file similar to the following example:

```
<jazn provider="LDAP"
       default-realm="sample_subrealm"
```

```
        location="ldap://orclcomp-sun.us.oracle.com:389">
</jazn>
```

This information identifies the LDAP-based Oracle Internet Directory URL (for this example, `orclcomp-sun.us.oracle.com`), the default realm (sample_subrealm.

Additional attributes and property names can also be set. The following attributes can be set in `orion-application.xml`:

Attribute	Status	Value
`provider`	Required	`LDAP`
`location`	Optional if `jazn.xml` file configured, otherwise Required	An LDAP server. For example: `ldap://orclcomp-sun.us.oracle.com:389`
`default-realm`	Optional (unnecessary if only one realm is configured)	A realm name. For example: `sample_subrealm`

The following property names can be set in `orion-application.xml`. If you want to permit anonymous, read-only logins to the application, do not set and assign values to these property names.

Property Name	Status	Value
`ldap.user`	Optional	A valid LDAP user name or DN. For example: `orcladmin` or cn=`orcladmin`
`ldap.password`	Optional	An obfuscated password for the LDAP user name. For example: `QJ+w7NJUlm=`
`ldap.cache.enable`	Optional	Setting this property name to `"true"` enables an LDAP cache feature that is implemented based on a simple TTL (time-to-live) expiration algorithm. When it is enabled, calling `Policy.grant/revoke` results in an `UnsupportedOperationException`. If you need to manage the JAZN policy, then you must disable caching (the default setting).

A sample `orion-application.xml` file with all attributes and property names specified is provided in "orion-application.xml file" on page 7-21.

Note: If you do not want to obfuscate the password entry, place an exclamation point (`!`) in front of the password value:

```
!welcome
```

See Also: "Task 2: Configure an Authentication Method and Filter Modes" on page 7-20 for information on the benefits of setting `auth-method`, `runas-mode`, and `doasprivileged-mode`:

Task 1b: Configure the XML-Based Provider Type for J2EE

Configure the JAAS to use the XML-based provider type by adding the following entry to the `orion-application.xml` file:

```
<jazn provider="XML" location="./jazn-data.xml" />
```

Additional attributes and property names can also be set. `orion-application.xml` permits the following attribute settings:

Attribute	Status	Value
`provider`	Optional	`XML` (Default)
`location`	Optional if `jazn.xml` file configured, otherwise Required	Path to file. For example: `./jazn-data.xml` This can be an absolute path, or a path relative to the `jazn.xml` file, where the JAAS Provider first looks for the `jazn-data.xml` in the directory containing the `jazn.xml` file.
`persistence`	Optional	■ `NONE` Do not persist changes ■ `ALL` Persist changes after every modification ■ `VM_EXIT` (Default) Persist changes when VM exits
`default-realm`	Optional (if only one realm is configured)	A realm name. For example: `sample_subrealm`

The following property names can be set in `orion-application.xml`:

Property Name	Status	Value
`xml.permclsmgr.enable`	Optional	■ `true` Enables the JAAS Permission Class Management feature ■ `false` (Default) Disables the feature
`xml.princlsmgr.enable`	Optional	■ `true` Enables the JAAS Principal Class Management feature ■ `false` (Default) Disables the feature

A sample `orion-application.xml` file with all attributes and property names specified is provided in "orion-application.xml file" on page 7-21.

See Also:

- Part 2, Chapter 3 of the *OC4J Services Guide* in the Oracle9*i*AS Documentation Library for information on authentication environments
- "Task 2: Configure an Authentication Method and Filter Modes" on page 7-20 for information on the benefits of setting `auth-method`, `runas-mode`, and `doasprivileged-mode`

Task 2: Configure an Authentication Method and Filter Modes

Integrate the JAAS with the type of authentication method you want to use:

- Single sign-on (SSO), with either single or multiple JAAS Web applications
- Secure Socket Layer (SSL)
- Basic Authentication

In addition to the authentication method, you can also use the filter element of `JAZNUserManager` and configure the optional `runas-mode` and `doasprivileged-mode` features. The filter is configured by the `<jazn-web-app>` element.

`runas-mode` and `doasprivileged-mode` include the following range of values for the `orion-web.xml` and `orion-application.xml` files:

Table 7–4 runas-mode and doasprivileged-mode Settings

If runas-mode is Set To...	If doasprivileged-mode Is Set To...	Then...
`true`	`true` (default)	`Subject.doAsPrivileged` in a `privilegedExceptionAction` block that calls `chain.doFilter` `(`*`myrequest`*`,response)`
`true`	`false`	`Subject.doAs` in a `privilegedExceptionAction` block that calls `chain.doFilter` `(`*`myrequest`*`,response)`
`false` (default)	`true`	`chain.doFilter` `(`*`myrequest`*`,response)`
`false`	`false`	`chain.doFilter` `(`*`myrequest`*`,response)`

See Also: Part 2, Chapter 3 of the *OC4J Services Guide* in the Oracle9*i*AS Documentation Library for information on the following:

- Authentication environments
- The filter element of `JAZNUserManager`
- Using the JAAS after completing these configuration tasks

The authentication method, as well as the `runas-mode`, and `doasprivileged-mode` features are specified in a configuration file. Since it is possible to specify the information in several files, the precedence indicated in Table 7–5 prevails. Specification in the first file overrides specification in the second and so on.

Table 7–5 Precedence of Configuration Files for Specifying Authentication Methods

Precedence	Configuration File
1	`web.xml`
2	`orion-application.xml`
3	`orion-web.xml`

orion-web.xml file

Specify your authentication method within the `<jazn-web-app>` element which enables the filter. In the following example, all three settings are optional:

```
<jazn-web-app
    auth-method="SSO"
    runas-mode="false"
    doasprivileged-mode="true"
/>
```

Set `auth-method` to `SSO` (single sign-on). If you do not set this parameter, it defaults to `null`. See Table 7–4 on page 7-20 for information on the impact of setting `runas-mode` and `doasprivileged-mode`.

orion-application.xml file

Specify your authentication method within the `<jazn-web-app>` element of the `<jazn>` element. The `<jazn-web-app>` element enables the filter. For example:

```
<jazn provider="XML"
        location="jazn-data.xml"
        default-realm="JAZN.com"
        persistence="ALL">

        <!-- default values for this application -->
        <jazn-web-app
                auth-method="SSO"
                runas-mode="true"
                doasprivileged-mode="true"
        />
        <property name="xml.princlsmgr.enable" value="true" />
        <property name="xml.permclsmgr.enable" value="true" />
</jazn>
```

All three settings in bold are optional. Set `auth-method` to `SSO`. If you do not set this parameter, it defaults to `null`. See Table 7–4 on page 7-20 for information on the impact of setting `runas-mode` and `doasprivileged-mode`.

web.xml

Specify your authentication method within the `<login-config>` element. For example:

```
<login-config>
        <auth-method>BASIC</auth-method>
</login-config>
```

Enter the following:

- `BASIC` or `FORM` to use basic authentication
- `DIGEST` to use digest authentication
- `CLIENT-CERT` to use SSL authentication

You cannot set `runas-mode` or `doasprivileged-mode` in this file. If they are required, configure `runas-mode` or `doasprivileged-mode` in the appropriate file: `orion-application.xml` or `orion-web.xml`. Be careful to maintain the desired precedence.

Task 3: Configure Your Application for SSL Environments

Note:

- If you are using basic authentication, then do not perform this configuration task.
- If you require Oracle9*i*AS Single Sign-On, then install the Oracle9*i*AS Infrastructure to automatically perform the necessary configuration tasks to enable single sign-on.

If you require SSL, then perform the following steps:

When you configure an application to run in an SSL environment, you must

- Set the port in the `httpd.conf` file
- Specify the server wallet directory and password. The server wallet contains the server's certificate, private key, and trusted certificates.
- Set SSL to on in the `mod_oc4j.conf` file

To configure an application for the SSL environment:

1. Uncomment the following lines in the `httpd.conf` file. If these lines are not in the `httpd.conf` file, add them.

```
LoadModule ossl_module        libexec/mod_ossl.so

<IfDefine SSL>
  Port 80
  Listen 80
  Listen 443
</IfDefine>

<VirtualHost _default_:443>
```

Note: SSL typically uses port 443. If this port is currently being used by a different process, change this port number in both places to one currently not being used.

2. Specify the server wallet file directory:

```
SSLWallet file:/wallet_file_directory
```

3. Add `$ORACLE_HOME/lib` to the `LD_LIBRARY_PATH` environment parameter.
4. Either use a clear text wallet password or use the `iasobf` utility to create an encrypted server wallet password. The `iasobf` utility is located in the `Apache/Apache/bin` directory.
5. Uncomment the following line and specify the server wallet password created in step 4.

   ```
   #SSLWalletPassword password
   ```

6. Uncomment the following line:

   ```
   #SSLOptions +FakeBasicAuth +ExportCertData +CompatEnvVars +StrictRequire
   ```

7. If you also want to use client certificate authentication, uncomment the following line and change `SSLVerifyClient` from `none` to `require` or `optional`.

   ```
   #SSLVerifyClient require
   ```

   Note: You must also configure your Web browser by adding the client certificate.

8. Go to the `mod_oc4j.conf` file.
9. Uncomment the following line and set `Oc4JExtractSSL` to `On`. If this line is not in `mod_oc4j.conf`, then add it.

   ```
   #Oc4jExtractSSL On
   ```

   See Also: "Using Secure Sockets Layer (SSL) to Authenticate Users" on page 4-12 for information about using the Oracle HTTP Server configuration directives for enabling SSL.

Task 4: Configure mod_oc4j to Delegate HTTP Requests to OC4J

To configure mod_oc4j to delegate HTTP requests to OC4J, you must add mount directives to the `mod_oc4j.conf` file. Use the following steps:

1. Go to `$ORACLE_HOME/Apache/Apache/conf/mod_oc4j.conf`.
2. Add the following lines. For this example, the application starts with the URI "reports."

   ```
   Oc4jMount /reports
   Oc4jMount /reports/*
   ```

3. Save your changes and exit `mod_oc4j.conf`.

Task 5: Configure the Security Role (run-as)

You can map J2EE security roles to JAAS roles by way of OC4J groups. This enables your application to run with the privileges of the security role or specific `RealmPrincipal` class. The following tasks pertain to both kinds of privileges; additional information appears on "RealmPrincipal Class" on page 7-27.

If the `run-as` element is specified, the `<role-name>` maps to a security role already defined for the Web application.

The following steps assume that `sr_manager` has already been defined as a security role in `web.xml` as follows:

```
<security-role>
   <role-name>sr_manager</role-name>
</security-role>
```

To map J2EE security roles to JAAS roles:

1. Specify the `run-as` element within the `<servlet>` tag to run as the specific J2EE security role or specific `RealmPrincipal` class in the `web.xml` file

   For example, to run as the security role `sr_manager`:

   ```
   <servlet>
     <servlet-name>DevGroup</servlet-name>
     <servlet-class>DevGroupServlet</servlet-class>
    <!--  run as security role "sr_manager" -->
       <run-as>
         <role-name>sr_manager</role-name>
       </run-as>
   </servlet>
   ```

2. Define a JAAS `role` element in the `jazn-data.xml` file:

   For example, `developer` is defined a role:

```
<roles>
        <role>
               <name>developer</name>
               <members>
                      <member>
                               <type>user<type>
                               <name>john<name>
                       </member>
                </members>
        </role>
 </roles>
```

   The `jazn-data.xml` file is discussed in Part 2, Chapter 4 of the *OC4J Services Guide*.

3. Integrate the definitions created in Step 1 and Step 2 using OC4J groups in the `orion-application.xml` file as follows:

   - Map the `role-name` defined in the `web.xml` file as a security role (`sr_manager`)
   - Map the `role` defined in `jazn-data.xml` as a OC4J group name (`developer`)

   For example, the `sr_manager` security role is mapped to the group named `developer` in the JAAS Provider:

   ```
   <security-role-mapping name="sr_manager">
      <group name="developer" />
   </security-role-mapping>
   ```

Because the `developer` group is mapped to the J2EE security role `sr_manager`, the user (`john` in this example) has access to the application resources defined by the `sr_manager` role.

RealmPrincipal Class

When the `<role-name>` element is set to a `RealmPrincipal` class name, the `<description>` element can also be set. For example:

```
<role-name>jazn.com/john</role-name>
<description>oracle.security.jazn.spi.xml.XMLRealmUser </description>
```

where `jazn.com` is the realm and the `RealmPrincipal` class name is `john`.

The filter attempts to look up the `RealmPrincipal` class object mapping to the security role (defined in the `<description>` element) and adds it to the subject.

If no mapping is found, the filter gets the `RealmPrincipal` class object based on the `<role-name>` element and optional `<description>` (`RealmPrincipal` class name) element, and adds it and its granted roles to the subject.

See Also:

- *Java Servlet Specification Version 2.3*
- Part 2, Chapter 6, "Developing Secure J2EE Applications" of the *OC4J Services Guide* in the Oracle9*i*AS Documentation Library
- Part 2, Chapter 3, "Integrating Oracle9*i*AS JAAS with Java2 Applications" of the *OC4J Services Guide* in the Oracle9*i*AS Documentation Library

Differences between <jazn> Tags and the <user-manager> Property

The <jazn> tags are very similar to the <user-manager> property and currently either can be used to configure Oracle support for JAAS. However, the following important reasons make using the <jazn> tags the preferred way to configure JAAS:

- <jazn> tags, which include <jazn>, <jazn-web-app>, can specify a greater range of functionality than that specified with the <user-manager> property.
- <jazn> tags are more secure because they support password obfuscation.
- <jazn> tags are easier to use.
- The <user-manager> property will not be supported beyond this current release of Oracle9*i* Application Server.

Consequently, Oracle Corporation recommends that <jazn> tags be used whenever possible.

Note: Currently, the <user-manager> property is only used when applications are deployed with the Oracle9*i*AS Containers for J2EE (OC4J) Deploy Application wizard, a GUI tool for deploying J2EE applications.

As previously described, the <jazn> tag is very similar to the <user-manager> property. When you transform the attributes of <jazn> tag into the properties of <user-manager>, that results in an almost equivalent configuration. For example, the attributes of the <jazn> tag compare to the properties of <user-manager> as follows:

<jazn> Tag Attribute Names	<user-manager> Properties
provider	provider.type
location	location
default-realm	realm-default
persistence	persistence
config	config

Similary, the attributes of the <jazn-web-app> tag compare to the properties of <user-manager> as follows:

<jazn-web-app> Tag Attribute Names	<user-manager> Properties
runas-mode	runas.mode
doasprivileged-mode	doasprivileged.mode
auth-method	authentication.method

See Also: *Oracle9iAS Containers for J2EE Services Guide* in the Oracle9*i*AS Documentation Library for more information about <jazn> tags and how to use them.

8

Configuring Security for Oracle9*i*AS Web Cache

This chapter explains how to configure security settings for Oracle9*i*AS Web Cache, including configuration for passwords and executable ownership. In addition, this chapter describes how to configure Oracle9*i*AS Web Cache for HTTPS support of secure pages.

This chapter contains these topics:

Modifying Default Security Settings

When Oracle9*i*AS Web Cache is installed, it is set up with default passwords for administration and invalidation requests. In addition, the computer on which you installed Oracle9*i*AS Web Cache is the default trusted host.

To change the security settings:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i*AS Documentation Library.

2. Change the password for the administrator.

   Configuration and operational tasks can be performed with the Oracle9*i*AS Web Cache `administrator` user. The `administrator` user has a default password of `administrator` set up during installation. Before you begin configuration, change the default password to a secure password.

   a. In the navigator pane, select **Administering Oracle Web Cache** > **Security**.

      The Security page appears in the right pane.

   b. In the Security page, click **Change Admin Password** under **Administration User**.

      The Change Administration User Password dialog box appears.

   c. Enter `administrator` in the **Old Password** field and a new password between four and 20 characters in the **New Password** and **Confirm New Password** fields.

   d. Click **Submit**.

3. Optionally, change the password for the invalidation administrator.

   The invalidation administrator has a user ID of `invalidator`, with default password of `invalidator`.

   a. In the Security page, click **Change Invalidation Password** under the **Invalidation User**.

      The Change Invalidation User Password dialog box appears.

b. Enter `invalidator` in the **Old Password** field, and a new password between four and 20 characters long in the **New Password** and **Confirm New Password** fields.

c. Click **Submit**.

4. Optionally, change the trusted subnet or trusted host from which Oracle9*i*AS Web Cache and invalidation administration can take place.

   By default, the computer on which you installed Oracle9*i*AS Web Cache is the trusted host.

   a. In the Security page, click **Change Trusted Subnets** under the **Currently trusted subnets**.

      The Change Trusted Subnets dialog box appears.

   b. Select one of the following options:

      All subnets

      Select to allow administration requests from all computers in all the subnets in the network.

      This machine only

      Select to allow administration and invalidation requests from only this computer.

      Enter list of IPs

      Select to allow requests from all IP addresses you enter in a comma-separated list. You can enter IP addresses in one of the following formats:

      – Complete IP address in dot notation, including the network number, subnet address, and unique host number

        Example: `10.1.2.3`

      – Network/netmask pair for subnet restriction through masking

        Example: `10.1.0.0/255.255.0.0` allows all the hosts in the `10.1` subnet access.

– Network / *nnn* Classless Inter-Domain Routing (CIDR) specification to require *nnn* bits from high end to match

  Example: `10.1.0.0/16` allows all the hosts in the `10.1` subnet access. This example is similar to the network / netmask example, except the netmask consists of *nnn* high-order 1 bits.

Note: Sometimes requests come through a proxy server. If the proxy server is not covered by the trusted subnet settings, then requests will fail.

---

c. Click **Submit**.

5. Optionally, change the user ID and group ID for the Oracle9*i*AS Web Cache executables on UNIX.

   By default, the user that performed the installation is the owner of Oracle9*i*AS Web Cache executables. This can user can execute `webcachectl` commands. Users that belong to the same group ID of the user that performed installation can also execute `webcachectl` commands.

   a. In the navigator pane, select **Administering Oracle Web Cache** > **Process Identity**.

      The Process Identity page appears in the right pane.

   b. In the Process Identity page, click **Change IDs**.

      The Change Process Identity dialog box appears.

   c. Enter the new user in the **New User ID** field and the group ID of the user in the **New Group ID** field.

   d. Click **Submit**.

6. In the Oracle9*i*AS Web Cache Manager main window, click **Apply Changes**.

Note: If you changed the password for the `administrator` user in Step 2, you must restart the `admin` server process with the `webcachectl restart` command rather than with the Restart option in the Operations page (**Administration** > **Operations**).

Configuring HTTPS Protocol Support

You can configure Oracle9*i*AS Web Cache to receive HTTPS browser requests and send HTTPS requests to the origin server. HTTPS uses the Secure Sockets Layer (SSL) to encrypt and decrypt user page requests as well as the pages that are returned by the origin server.

To describe the how SSL works in an HTTPS connection, the word client is used to describe either a browser or Oracle9*i*AS Web Cache, and the word server is used to describe either Oracle9*i*AS Web Cache or an origin server.

The authentication process between the client and server consists of the steps that follow:

1. The client initiates a connection to the server by using HTTPS.
2. SSL performs the handshake between the client and the server.

At the commencement of an HTTPS network connection between the client and server, an SSL handshake is performed. An SSL handshake includes the following actions:

- The client and server establish which cipher suites to use.
- The server sends its certificate to the client, and the client verifies that the server's certificate was signed by a trusted CA.
- The client and server exchange key information using public key cryptography; based on this information, each generates a session key. All subsequent communications between the client and the server is encrypted and decrypted by using this set of session keys and the negotiated cipher suite.

To configure HTTPS support, perform these tasks:

- Task 1: Create Wallets
- Task 2: Configure HTTPS Listening Ports and Wallet Location
- Task 3: Permit Only HTTPS Requests for a Site

Task 1: Create Wallets

Wallets are needed to support the following HTTPS requests:

- Browser requests for sites hosted by Oracle9*i*AS Web Cache
- Administration, invalidation, and statistics monitoring requests to Oracle9*i*AS Web Cache
- Oracle9*i*AS Web Cache requests to origin servers

Each site requires at least one wallet. One wallet can be shared among all the Oracle9*i*AS Web Cache listening ports, or a separate wallet can be created for each Oracle9*i*AS Web Cache listening port.

To create a wallet, use Oracle Wallet Manager. Create the wallet as the following user:

- The user and group ID configured in the Process Identity page (**Cache-Specific Configuration** > **Process Identity**) on UNIX
- The owner of `Oracle`*`HOME_NAME`*`WebCache` service on Windows

When the `webcachectl` or `Oracle`*`HOME_NAME`*`WebCache` service starts the `cache` server process, Oracle9*i*AS Web Cache opens the wallet as the `webcachectl` or the `Oracle`*`HOME_NAME`*`WebCache` service owner.

By default, wallets are stored in the following locations:

- `/etc/ORACLE/WALLETS/`*`user_name`* on UNIX
- `%USERPROFILE%\ORACLE\WALLETS` on Windows NT and Windows 2000

See Also: Chapter 5, "Using Oracle Wallet Manager" for information about using Oracle Wallet Manager to create and manage Oracle Wallets.

Enabling Wallets to Open on Windows

Oracle9*i*AS Web Cache attempts to open wallets at startup on Windows. On Windows, wallets are protected so that only the user that created them can open and use them. By default, Oracle9*i* Application Server services are associated with the local system account, which does not have permission to open wallets.

To enable Oracle9*i*AS Web Cache to open wallets at startup:

1. Create a wallet with an administrator account.
2. Change the system account information for the Oracle9*i*AS Web Cache services:

Windows NT	Windows 2000
8. Choose the **Services** icon from the Control Panel window. The Services window appears.	**1.** Choose **Administrative Tools** > **Services** from the Control Panel window. The Services window appears.
9. Select the `Oracle`*`HOME_NAME`*`WebCache` service. The Service dialog appears.	**2.** Select the `Oracle`*`HOME_NAME`*`WebCache` service. The Oracle*HOME_NAME*WebCache Properties dialog appears.
10. Click **This Account**. By default the LocalSystem user account is associated with the service.	**3.** Click the **Log On** tab. **4.** In the Log On tab, click **This account**. By default the LocalSystem user account is associated with the service.
11. Click the ellipse (**...**) next to **This Account**. The Add User dialog box appears.	**5.** Click **Browse** next to **This Account**. The Select User dialog box appears.
12. Select the user that created the wallet from the Names list, and then click **Add**.	**6.** Select the user that created the wallet from the list, and then click **OK**.
13. Click **OK** to close the Add User dialog box.	**7.** Click **OK** to close the Add User dialog box.
14. In the Service dialog box, provide the password for the wallet administrator in the **Password** field, and then confirm the password in the **Confirm Password** field.	**8.** In the Oracle*HOME_NAME*WebCache Properties dialog box, provide the password for the wallet administrator in the **Password** field, and then confirm the password in the **Confirm Password** field.
15. In the Services dialog box, click **OK**.	**9.** In the Services dialog box, click **OK**.
16. Repeat Steps 3 - 9 for the `Oracle`*`HOME_NAME`*`WebCacheAdmin` and `Oracle`*`HOME_NAME`*`WebCacheMon` services.	**10.** Repeat Steps 3 - 9 for the `Oracle`*`HOME_NAME`*`WebCacheAdmin` and `Oracle`*`HOME_NAME`*`WebCacheMon` services.
17. In the Services window, click **Close**.	

On Windows NT, additionally grant the wallet administrator the right to run Oracle9*i*AS Web Cache as a service:

1. Choose **Start** > **Programs** > **Administrative Tools** > **User Manager**.

   The User Manager window appears.

2. Select the wallet administration, and then choose **Policies** > **User Rights**.

   The User Rights Policy dialog box appears.

3. Click the **Show Advanced User Rights** check box, and then select **Log on as a service** from the **Right** list.

4. Select **Users** from the **Grant To** list.

   If **Users** does not exist, create it:

   a. Click **Add**.

      The Add Users and Groups dialog box appears.

   b. Select the name of the local host computer from the **List Names From** list.

   c. Select **Users** from the **Names** list, and then choose **Add**.

   d. Click **OK**.

      Users appears in the **Grant To** list.

5. Click **OK** in the User Rights Policy dialog box.

   The User Manager window reappears.

6. Choose **User** > **Exit**.

Task 2: Configure HTTPS Listening Ports and Wallet Location

To configure HTTPS protocol support between browsers and Oracle9*i*AS Web Cache:

1. Select **Cache-Specific Configuration** > **Listening Ports** in Oracle9*i*AS Web Cache Manager to configure Oracle9*i*AS Web Cache with an HTTPS listening port and the location of the wallet for each supported site.
2. Select **Cache-Specific Configuration** > **Operations Ports** in Oracle9*i*AS Web Cache Manager to configure administration, invalidation, and statistics monitoring requests with HTTPS listening ports and the location of the site’s wallet.

   The ports for these requests can share the same wallet as established for the Oracle9*i*AS Web Cache listening port in Step 1.

To configure HTTPS protocol support between Oracle9*i*AS Web Cache and origin servers:

1. Select **General Configuration** > **Application Web Servers** or **Proxy Servers** in Oracle9*i*AS Web Cache Manager to configure an application Web server or proxy server with an HTTPS communication port.
2. Select **Cache-Specific Configuration** > **Origin Server Wallet** in Oracle9*i*AS Web Cache Manager to specify the location of the wallet used for communication from Oracle9*i*AS Web Cache to an application Web server or proxy server.

   The ports for these requests can share the same wallet as established for the Oracle9*i*AS Web Cache listening ports.

See Also: Chapter 6, "Initial Setup and Configuration," of *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i*AS Documentation Library for further information about configuring HTTPS listening ports and specifying the location of the wallet.

Task 3: Permit Only HTTPS Requests for a Site

You can restrict a URL or set of URLs for a site to permit only HTTPS requests.

To allow only HTTPS traffic for a URL or a set of URLs:

1. Select **General Configuration** > **Sites** in Oracle9*i*AS Web Cache Manager to configure to specify a site definition.
2. In the Site Definitions page, choose **Add Site** to add a site definition or Edit Site to modify an existing site definition.

   The Add Site or Edit Site dialog box appears.
3. In the **HTTPS Only Prefix** field, enter the URL prefix for which only HTTPS requests will be served.

   If all traffic must be restricted to HTTPS, enter "`/` " for the entire site.
4. Enter appropriate information in the other fields, as described in the online help and Chapter 6, "Initial Setup and Configuration," of *Oracle9iAS Web Cache Administration and Deployment Guide* in the Oracle9*i*AS Documentation Library.

9 Configuring Secure Database Access by Oracle9*i* Application Server

This chapter introduces features and practices which enable clients to securely access the database in an Internet environment. By using these approaches to set up the database and PL/SQL, you can avoid a number of known security problems.

Providing for Secure Access to the Oracle9*i* Database Server

This section describes Oracle9*i* Database Server tools which Oracle9*i* Application Server (Oracle9*i*AS) can use to facilitate secure deployment of the system.

- Proxy Authentication with Oracle9iAS
- Auditing for Multitier Applications
- Secure Application Roles
- Application Query Rewrite: Virtual Private Database (VPD)
- Selective Encryption of Stored Data
- Middle-Tier Connection Management
- Java Security Implementation in the Database

Proxy Authentication with Oracle9*i*AS

The proxy authentication feature is designed so that a specific middle tier can be restricted to acting on behalf of a specified set of users. Once the middle tier has authenticated itself to the database, it can establish a lightweight session on behalf of those users without submitting user-specific authentication information such as passwords. Moreover, Oracle9*i* can be configured so that a specific middle tier can assume a specific set of database roles when acting at the database on behalf of a specific user. In other words, the database uses both middle-tier identity and client user identity when determining what privileges to grant a middle tier acting for a user through a lightweight session.

In Oracle9*i* Application Server, this feature supports Java Database Connectivity (JDBC) access to the database. A middle-tier server can now access the Oracle9*i* database on behalf of a client user by establishing a lightweight session for that user through JDBC-OCI.

Oracle9*i* extends proxy authentication to include additional credential proxy of either the Distinguished Name (DN) or full X.509 certificate to the database. This provides strong, multitier security by enabling an SSL credential—an X.509 certificate or DN—to be passed to the database for purpose of identifying (but not authenticating) the user. (SSL cannot be used to authenticate a user through multiple tiers, because it is a point-to-point protocol rather than an end-to-end protocol.) For example, a user can authenticate to a middle tier using SSL, the middle tier can extract the DN from the certificate and pass it (or the full certificate) to the database. As an additional benefit, the DN or certificate is available in the lightweight session and the elements contained therein can be used with Virtual

Private Database to limit access. For example, an organization could restrict data access based on the Organizational Unit (OU) element in a user certificate presented to the database. Application user proxy authentication is thus particularly valuable in e-business applications with thousands of users, as it supports data access control by user while meeting user scalability requirements.

The database can use the DN or certificate to look up a user in Oracle Internet Directory or other LDAP-based directory certified for enterprise user security (an Oracle Advanced Security feature). Integration of proxy authentication with enterprise user security enables the user identity to be maintained throughout all tiers of an application, yet the user need only be created once, in the directory. This also enables enterprise user security to be used in multitier applications, instead of merely client/server.

Auditing for Multitier Applications

Many multitier applications authenticate users to the middle tier, and then the transaction processing monitor or application server connects as super-privileged user, and performs all activity on behalf of all users. But it is important to be able to preserve the identity of the real client over the middle tier and enforce "least privilege" through a middle tier. In addition, you may want to audit actions taken on behalf of the user by the middle tier. Auditing user activity, whether users are connected through a middle tier or directly to the data server, enhances user accountability, and thus the overall security of multitier systems.

With Oracle9*i* Application Server, you can do all of these things. Oracle9*i* audit records capture both the logged-in user (that is, the middle tier) who initiated the connection, and the user on whose behalf an action is taken.

Secure Application Roles

A long-standing security problem has been that of limiting how users access data, to prevent users from bypassing application logic to access data directly. For example, in Web-based applications, even if users are known to the database, it may not be desirable to allow them to have direct access to data. To date, this has been a very difficult security problem to solve, because there has been no secure way to validate which application is used to access data. For example, a malicious user could write a program that *appears* to be a valid human resources application.

One way to address this challenge is through a secure application role: a role implemented by a package. The package can perform any desired validation to ensure that the appropriate conditions are met before the user can exercise privileges granted to the role in the database. The database ensures that it is only the trusted package implementing the role that determines the correct access conditions.

This feature, unique to Oracle9*i*, enables you to base use of roles on user-defined criteria. A secure application role is a role which is implemented by a package. For example, you could write a package permitting use of a role by a user connecting only from a particular IP address, or accessing the database only through a particular middle tier. A secure application role is used by an application, can only be enabled by the application, and does not need a password.

In multitier systems using proxy authentication, the package can validate that the user session was created by a middle tier, and thus that the user is accessing the database through the correct application. The secure application role can also ensure that a user connecting directly to the database is not able to access any data. A secure application role can enforce other security conditions, as well; for example, the user may not be permitted to access especially sensitive human resources data from the Internet.

A secure application role enhances the native strong authentication and fine-grained access control of the database to prevent users from assuming any privileges unless the correct access conditions are met. Secure application role thus solves a very difficult security issue and supports secure Web-based application data access.

Application Query Rewrite: Virtual Private Database (VPD)

Virtual Private Database is the ability to perform query modification based on a security policy you have defined in a package, and associated with a table or view. Virtual private database provides fine-grained access control which is data-driven, context-dependent, and row-based. It is a key enabling technology in building multitier systems which expose mission-critical resources to customers and partners.

Selective Encryption of Stored Data

For certain applications, you may decide to encrypt data as an additional measure of security. Most issues of data security can be handled by appropriate authentication and access control, ensuring that only properly identified and authorized users can access data. Data in the database, however, cannot normally be secured against the database administrator's access, because a DBA has all privileges. Likewise, organizations may have concerns about securing sensitive data stored off-line, such as backup files stored with a third party. They may want to guard against intruders accessing the data where it is physically stored on the database.

Although encryption is not a substitute for effective access control, you can obtain an additional measure of security by selectively encrypting sensitive data before it is stored in the database. Information which may be especially sensitive and warrant encryption could include credit card numbers, national identity numbers in countries with strict privacy laws, or trade secrets, such as industrial formulas. Applications for which a user is authenticated to the application, rather than to the database, may also use encryption to protect the application user password or cookie.

Middle-Tier Connection Management

You can configure a middle tier that manages the connections of very large user populations. To support a large number of users, you can configure multiple instances of Oracle Connection Manager. This product multiplexes multiple client network sessions through a single network connection to the database, increasing the total number of connections.

It is also possible to filter on source, destination, and host name. Thus you can ensure that connections only come from a physically secure terminal, or from an application Web server with a known IP address. (IP address alone is not enough for authentication, because it can be faked.) You thus could allow connections from IP address FOO, connecting to host BAR for PAYROLL.

Java Security Implementation in the Database

Oracle9*i* contains a Java security implementation in the server. The Java virtual machine (JVM) is the Java interpreter that converts the compiled Java bytecode into the machine language of the platform and runs it. JVMs can run on a client, in a browser, in a middle tier, on a Web, on an application server such as Oracle9*i* Application Server, or in a database server such as Oracle9*i*.

Class Execution

In the Oracle9*i* JVM implementation, the right to execute code in classes is controlled by execute privileges on the classes themselves. This is the same database privilege as execute privilege on a PL/SQL package, and is managed in the same way.

SecurityManager Class

The Oracle9*i* JVM starts with the class java.lang.SecurityManager installed. The Oracle9*i* database is based on the Java Developer's Kit 1.2 release from Sun Microsystems, and implements the security features of that release. In this implementation, permissions are controlled by the contents of a database table. The table is normally managed by PL/SQL procedures (and Java methods). The table can be used to grant permissions to either users or roles, and the "code source" of a class is identified with the user in whose schema the class has been loaded. Specific Oracle permissions control the right to update the table and perform other security sensitive operations.

See Also: *Oracle9i Java Developer's Guide* in the Oracle Database Documentation Library.

Securing Application Database Access Through mod_plsql

This section describes how to set up the database and PL/SQL to avoid known security problems. It covers the following topics:

- Introduction to mod_plsql
- Authenticating Users Through mod_plsql
- Protecting the PL/SQL Procedures Granted to PUBLIC
- Providing Secure Database Connections with JDBC

> **See Also:** For more information about mod_plsql, refer to the *mod_plsql User's Guide* and the *PL/SQL Web Toolkit Reference* in the Oracle9*i*AS Documentation Library.

Introduction to mod_plsql

There are essentially two types of mod_plsql security. One is authentication, which establishes who the user is. The second is protection, which determines the privileges of the user.

In the context of Oracle HTTP Server, the Web listener authenticates its users and controls database access by means of the mod_plsql module. The front end of mod_plsql appears as an HTTP server module, and the back end appears as a client which obtains the correct level of access to the database on behalf of the user. The mod_plsql client connects to the database, gets results, and transmits the results back to the HTTP server.

Authenticating Users Through mod_plsql

mod_plsql provides different levels of authentication in addition to those provided by the Oracle HTTP Server. The Oracle HTTP Server protects documents, virtual paths and so forth, while mod_plsql protects users logging into the database or running a PL/SQL Web application.

You can enable different authentication modes, as described in Table 9–1.

Table 9–1 Authentication Modes Used with mod_plsql

Authentication Mode	Approach
Basic	Authentication is performed using basic HTTP authentication. Most applications use basic authentication.
Global Owa	Authentication is performed using the `owa_custom.authorize` procedure in the schema containing the PL/SQL Web Toolkit packages.
Custom Owa	Authentication is performed using packages and procedures in the user's schema (`owa_customize.authorize`), or if not found, in the schema containing the PL/SQL Web Toolkit packages.
PerPackage	Authentication is performed using packages and procedures in the user's schema (`packageName.authorize`).
Single Signon	Authentication is performed using Oracle9*i*AS Single Sign-On. Use this mode only if your application works with Oracle9*i*AS Single Sign-On.

Basic (Database Controlled Authentication)

The module, mod_plsql, supports authentication at the database level. It uses HTTP basic authentication but authenticates credentials by using them to attempt to log on to the database. Authentication is verified against a user database account, using user names and passwords that are either

- stored in the DAD. The end user is not required to log in. This method is useful for Web pages that provide public information.
- provided by the users by means of the browser's Basic HTTP Authentication dialog box. The user must provide a user name and password in the dialog box.

Oracle9*i* Application Server Basic Authentication Mode

Oracle9*i* Application Server has a different mechanism for the basic authentication mode. The user name and password must be stored in the DAD. Oracle9*i* Application Server uses HTTP basic authentication where the credentials are stored in a password file on the file system. Authentication is verified against the users listed in that file.

Oracle9*i*AS Basic Authentication Mode mod_plsql supports Oracle9*i* Application Server basic authentication. Oracle HTTP Server authenticates users' credentials against a password file on the file system. This functionality is provided by a module called `mod_auth`.

> **See Also:** "Using Basic Authentication and Authorization with mod_auth" on page 4-10 for more information about mod_auth.

Deauthentication

mod_plsql allows users to log off (clear HTTP authentication information) programmatically through a PL/SQL procedure without having to exit all browser instances. This feature is supported on Netscape 3.0 or higher and on Microsoft Internet Explorer. On other browsers, the user may have to exit the browser to deauthenticate.

Another method of deauthentication is to add `/logmeoff` after the DAD in the URL. For example:

```
http://myhost:2000/pls/myDAD/logmeoff
```

Global OWA, Custom OWA, and Per Package (Custom Authentication)

Custom authentication enables applications to authenticate users within the application itself, not at the database level. Authorization is performed by invoking a user-written authorization function. Custom authentication uses a static user name and password that is stored in the DAD. It cannot be combined with dynamic user name and password authentication.

You can place the authentication function in different locations, depending on when it is to be invoked:

- Global OWA enables you to invoke the same authentication function for all users and procedures.
- Custom OWA enables you to invoke a different authentication function for each user and for all procedures.

- Per Package authentication enables you to invoke the authentication function for all users, but only for procedures in a specific package or for anonymous procedures.

Table 9–2 summarizes the parameter values.

Table 9–2 Custom Authentication Modes and Callback Functions

Mode	Access control scope	Callback function
Global OWA	All packages	`owa_custom.authorize` in the OWA package schema
Custom OWA	All packages	`owa_custom.authorize` in the user's schema, or, if not found, in the OWA package schema
Per package	Specified package	`packageName.authorize` in the user's schema, or `anonymous.authorize` is called

Protecting the PL/SQL Procedures Granted to PUBLIC

Every database package granted to public can be directly executed using the following URL:

```
http://hostname:port/pls/dad/schema.package.procedure
```

With the different levels of authentication, you must protect the execution of the PL/SQL procedures granted to PUBLIC in the database. These procedures (in the `dbms_%` packages, `utl_%` packages, and all packages under the `SYS` schema) pose a security vulnerability when they are executed through a Web browser. Such packages are intended only for the PL/SQL application developer.

Using the PlsqlExclusionList Directive in mod_plsql mod_plsql provides a DAD parameter directive called `PlsqlExclusionList` to protect the execution of these PL/SQL packages and other packages that are specific to applications. The `PlsqlExclusionList` directive specifies a pattern for procedure, package, and schema names that are forbidden to be directly executed from a browser. This is a multiline directive in which each pattern is specified on one line. The pattern is not case sensitive and it accepts simple wildcards such as `*`, `?`, and `[a-z]`. The default patterns that are not accessible from a direct URL are `sys.*`, `dbms_*`, `utl_*`, and `owa_util.*`.

> **Caution:** Setting the `PlsqlExclusionList` directive to `#NONE#` will disable all protection. It is not recommended for a live Web site. Only use this setting for debugging purposes.

If the `PlsqlExclusionList` directive is overridden, then the default settings do not apply. In this case, you must add the default list to the list of excluded patterns.

Accessing the PlsqlExclusionList Directive You can set the `PlsqlExclusionList` directive in the mod_plsql configuration file called `dads.conf`. This configuration file is located in the following directories:

- (UNIX) `ORACLE_HOME/Apache/modplsql/conf/`
- (Windows) `ORACLE_HOME\Apache\modplsql\conf`

Where ORACLE_HOME is the location of your Oracle9*i*AS Portal installation.

To ensure the best security for PL/SQL procedures that are granted to PUBLIC, specify the system default settings with the `PlsqlExclusionList` directive in the `dads.conf` file as shown in Example 9–1.

Example 9–1 System Default Settings Specified with the PlsqlExclusionList Directive

```
PlsqlExclusionList sys.*
PlsqlExclusionList dbms_*
PlsqlExclusionList utl_*
PlsqlExclusionList owa_util.*
PlsqlExclusionList portal.wwmon*
```

In addition to the patterns that are specified with this directive, mod_plsql also disallows any URLs that contain special characters such as tabs, newlines, carriage returns, single quotation marks ('), or reverse slashes (\). You cannot change this.

Providing Secure Database Connections with JDBC

This section describes use of Java Database Connectivity (JDBC) in the context of Oracle9*i* Application Server to ensure that access to the database is secure. It contains the following topics:

- Introduction to Java Database Connectivity (JDBC)
- Java Encryption Features of Oracle Advanced Security

Introduction to Java Database Connectivity (JDBC)

You can use Java to transmit data securely in a multitier environment. JDBC (Java Database Connectivity) is an industry-standard API (Applications Programming Interface) which allows Java programs to send SQL statements to an object-relational database such as Oracle. Java Database Connectivity enables a middle-tier server to access a database on behalf of a client user by establishing a lightweight session for the user.

Java applets can thus transmit data over secure channels. You can have secure connections from middle tier servers with Java Server Pages (JSPs) to the database. This enhances security because:

- Every protocol can be secured.
- JDBC-OCI and thin clients can be supported.
- Multitier architectures can be supported.

There are basically two ways to implement Java security and negotiate algorithms:

- Hard code it into your JDBC client application
- Configure it like native network cryptography

Java Encryption Features of Oracle Advanced Security

Sun Microsystems, Inc. defined the Java Database Connectivity (JDBC) standard, and Oracle Corporation, as an individual provider, implements and extends the standard with its own JDBC drivers. Oracle offers 4 kinds of JDBC driver:

- JDBC-OCI built on top of OCI libraries and Oracle Net Services client
- JDBC thin for both applets and applications (that do not have client installation and want maximum portability)
- JDBC server-side thin for connecting to remote Oracle databases from inside the target database
- Server-side internal driver for code such as stored procedures and functions executing inside the database.

JDBC-Oracle Call Interface Driver

The JDBC-Oracle Call Interface (JDBC-OCI) driver can be used for client-side use with an Oracle client installation. Because the JDBC-OCI driver uses the full Oracle Net Services communications stack on both client and server, it can take advantage of existing Oracle Advanced Security encryption and authentication mechanisms. In Oracle9*i*, proxy authentication has been extended to Java Database Connectivity (JDBC-OCI), which enables a middle-tier server to access the Oracle9*i* database on behalf of a client user by establishing a lightweight session for the user.

JDBC Thin Driver for Applets and Application

Because the thin JDBC driver is designed to be used with applets that are downloaded over the Internet, Oracle9*i* includes a 100% Java implementation of Oracle Advanced Security encryption and integrity algorithms for use with thin clients. Several benefits enable e-businesses deploying Oracle and other components to securely transmit a variety of information over a variety of channels:

- Data encryption for privacy of communications
- Data integrity checking to safeguard against data modification, replay, and eavesdropping
- Secure connections from thin JDBC clients to the Oracle9*i* database
- Ability for developers to build applets that transmit data over a secure communication channel
- Secure connections from Oracle9*i* databases to older versions of Oracle databases that are configured to use Oracle Advanced Security

- Secure connections from middle-tier servers with Java Server Pages (JSP) to the Oracle RDBMS

The Oracle JDBC Thin driver implements the Oracle password protocol for authentication. It does not support Oracle Advanced Security SSL implementation, nor does it support third-party authentication features such as RADIUS or Kerberos. The Oracle JDBC-OCI driver supports all Oracle Advanced Security features.

Oracle Advanced Security continues to encrypt and provide integrity checking of Oracle Net Services traffic between Oracle Net Services clients and Oracle servers using algorithms written in C. The Oracle Advanced Security Java implementation for Thin JDBC provides Java versions of the following encryption algorithms:

- RC4_256
- RC4_128
- RC4_56
- RC4_40
- DES56
- DES40

JDBC Server-Side Thin Driver

The JDBC Thin driver is a Type 4 (100% pure Java) driver that uses Java sockets to connect directly to a database server. It has a lightweight Java implementation of Oracle Net called Java Net.

The Thin driver does not require Oracle software on the client side. It does need a TCP/IP listener on the server side. Use this driver in regular Oracle Net listener Java applets that are downloaded into a Web browser. The Thin driver is self-contained, but it opens a Java socket, and thus can only run in a browser that supports sockets.

Oracle Java SSL

Oracle Java SSL is an implementation of Java Secure Socket Extension (JSSE) that is suitable for commercial use. In order to create a secure, fast implementation of SSL, Oracle Java SSL uses native code to improve the performance of critical components. In addition to the functionality included in the JSSE specifications, Oracle Java SSL supports the following:

- Multiple cryptographic algorithms
- Certificate and key management using Oracle Wallet Manager
- SSL-specific session capabilities, including authentication, that can be used by applications built on top of Oracle Java SSL

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

Secure Connections for Virtually Any Client

On the server, the negotiation of algorithms and the generation of keys function exactly the same as Oracle Advanced Security Oracle Net Services encryption, thus enabling backward and forward compatibility of clients and servers. On the clients, the algorithm negotiation and key generation occur in exactly the same manner as C-based Oracle Advanced Security encryption. The client and server negotiate encryption algorithms, generate random numbers, use Diffie-Hellman to exchange session keys, and use the Oracle Password Protocol, in the same manner as traditional Oracle Net Services clients. Thin JDBC contains a complete implementation of an Oracle Net Services client in pure Java. Consistent with other encryption implementations, the Java implementation of Oracle Advanced Security prevents access to the cryptographic algorithms, makes it impossible to double encrypt data, and encrypts data as it passes through the network. Users cannot alter the keyspace nor alter the encryption algorithms themselves.

Glossary

authentication

The process of verifying the identity of a user, device, or other entity in a computer system, often as a prerequisite to granting access to resources in a system. A recipient of an authenticated message can be certain of the message's origin (its sender). Authentication is presumed to preclude the possibility that another party has impersonated the sender.

availability

The percentage or amount of scheduled time that a computing system provides application service.

CA

See **certificate authority**.

certificate

Also called a digital certificate. An ITU x.509 v3 standard data structure that securely binds an identity to a public key.

A certificate is created when an entity's public key is signed by a trusted identity, a certificate authority. The certificate ensures that the entity's information is correct and that the public key actually belongs to that entity.

A certificate contains the entity's name, identifying information, and public key. It is also likely to contain a serial number, expiration date, and information about the rights, uses, and privileges associated with the certificate. Finally, it contains information about the certificate authority that issued it.

certificate authority

A trusted third party that certifies that other entities—users, databases, administrators, clients, servers—are who they say they are. When it certifies a user, the certificate authority first seeks verification that the user is not on the certificate revocation list (CRL), then verifies the user's identity and grants a certificate, signing it with the certificate authority's private key. The certificate authority has its own certificate and public key which it publishes. Servers and clients use these to verify signatures the certificate authority has made. A certificate authority might be an external company that offers certificate services, or an internal organization such as a corporate MIS department.

ciphertext

Data that has been encrypted. Cipher text is unreadable until it has been converted to plain text (decrypted) with a key. See **decryption**.

cipher suite

A set of authentication, encryption, and data integrity algorithms used for exchanging messages between network nodes. During an SSL handshake, for example, the two nodes negotiate to see which cipher suite they will use when transmitting messages back and forth.

cleartext

See **plaintext**.

cryptography

The art of protecting information by transforming it (encrypting) into an unreadable format (**ciphertext**). See **encryption**.

decryption

The process of converting the contents of an encrypted message (**ciphertext**) back into its original readable format (**plaintext**).

DES

Data Encryption Standard. A commonly used symmetric key **encryption** method that uses a 56-bit key.

Diffie-Hellman key negotiation algorithm

This is a method that lets two parties communicating over an insecure channel to agree upon a random number known only to them. Though the parties exchange information over the insecure channel during execution of the Diffie-Hellman key negotiation algorithm, it is computationally infeasible for an attacker to deduce the random number they agree upon by analyzing their network communications. Oracle Advanced Security uses the Diffie-Hellman key negotiation algorithm to generate session keys.

digital certificate

See **certificate**.

digital wallet

See **wallet**.

directory information tree (DIT)

A hierarchical tree-like stucture consisting of the DNs of the directory entries. See **distinguished name (DN)**.

distinguished name (DN)

The unique name of a directory entry. It comprises all of the individual names of the parent entries back to the root.

encryption

The process of disguising a message thereby rendering it unreadable to any but the intended recipient. Encryption is performed by translating data into secret code. There are two main types of encryption: **public-key encryption** (or asymmetric-key encryption) and symmetric-key encryption. See **symmetric-key cryptography**.

entry

In the context of a directory service, entries are the building blocks of a directory. An entry is a collection of information about an object in the directory. Each entry is composed of a set of attributes that describe one particular trait of the object. For example, if a directory entry describes a person, that entry can have attributes such as first name, last name, telephone number, or e-mail address.

failover

The ability to reconfigure a computing system to utilize an alternate active component when a similar component fails.

fault tolerance

The ability of a computing system to withstand faults and errors while continuing to provide the required services.

hot standby

A second running computing system that is ready to pick up application processing in the event that the primary computing system fails. That is, the secondary system takes over the processing at the point where the original computing system stopped and the secondary system continues the processing.

key

A password or a table needed to decipher encoded data.

key pair

A public key and its associated private key.

LDAP

See **Lightweight Directory Access Protocol (LDAP)**

LDAP Data Interchange Format (LDIF)

The set of standards for formatting an input file for any of the LDAP command-line utilities.

LDIF

See **LDAP Data Interchange Format (LDIF)**

Lightweight Directory Access Protocol (LDAP)

A standard, extensible directory access protocol. It is a common language that LDAP clients and servers use to communicate. The framework of design conventions supporting industry-standard directory products, such as the Oracle Internet Directory.

man-in-the-middle

A security attack characterized by the third-party, surreptitious interception of a message, wherein the third-party, the *man-in-the-middle*, decrypts the message, re-encrypts it (with or without alteration of the original message), and re-transmits it to the originally-intended recipient—all without the knowledge of the legitimate sender and receiver. This type of security attack works only in the absence of **authentication**.

MD5

A hashing algorithm intended for use on 32-bit machines to create digital signatures. MD5 is a **one-way hash function**, meaning that it converts a message into a fixed string of digits that form a **message digest**.

message digest

Representation of text as a string of single digits. It is created using a formula called a **one-way hash function**.

mission critical

See fault tolerance.

one-way hash function

An algorithm that turns a message into a single string of digits. "One way" means that it is almost impossible to derive the original message from the string of digits. The calculated **message digest** can be compared with the message digest that is decrypted with a **public key** to verify that the message has not been tampered with.

Oracle Net

An Oracle product that enables two or more computers that run an Oracle database server or Oracle tools, such as Designer/2000 to exchange data through a third-party network. Oracle Net supports distributed processing and distributed databases. Oracle Net is an open system because it is independent of the communication protocol, and users can interface Oracle Net to many network environments.

Oracle PKI certificate usages

Defines Oracle application types that a **certificate** supports.

PEM

Privacy-Enhanced Electronic Mail. An **encryption** technique that provides encryption, authentication, message integrity, and **key** management.

PGP

Pretty Good Privacy. An **encryption** technique that is based on **public key** cryptography. The PGP encryption package is free.

PKCS #12

A **public-key encryption** standard (PKCS). RSA Data Security, Inc., PKCS #12 is an industry standard for storing and transferring personal authentication credentials—typically in a format called a **wallet**.

PKI

Public Key Infrastructure. The basis for managing **public key**s used to provide **encryption**.

plaintext

Also called cleartext. Unencrypted data in ASCII format.

private key

In **public-key cryptography**, this key is the secret key. It is primarily used for decryption but is also used for encryption with digital signatures. See **public/private key pair**.

public key

In **public-key cryptography**, this key is made public to all. It is primarily used for encryption but can be used for verifying signatures. See **public/private key pair**.

public-key cryptography

Encryption method that uses two different random numbers (**key**s). See **public key** and **public-key encryption**.

public-key encryption

The process where the sender of a message encrypts the message with the public key of the recipient. Upon delivery, the message is decrypted by the recipient using its private key.

public/private key pair

A set of two numbers used for **encryption** and **decryption**, where one is called the **private key** and the other is called the **public key**. Public keys are typically made widely available, while private keys are held by their respective owners. Though mathematically related, it is generally viewed as computationally infeasible to derive the private key from the public key. Public and private keys are used only with asymmetric encryption algorithms, also called public-key encryption algorithms, or public-key cryptosystems. Data encrypted with either a public key or a private key from a **key pair** can be decrypted with its associated key from the key-pair. However, data encrypted with a public key cannot be decrypted with the same public key, and data encrypted with a private key cannot be decrypted with the same private key.

relative distinguished name (RDN)

The leftmost component in a directory entry's distinguished name (DN). See **distinguished name (DN)**.

reliability

The ability of a computing system to operate without failing. Reliability is measured by mean-time-between-failures (MTBF).

redundant

Duplicate or extra computing components that protect a computing system.

RSA

A **public-key encryption** technology developed by RSA Data Security. The RSA algorithm is based on the fact that it is laborious to factor very large numbers. This makes it mathematically unfeasible, because of the computing power and time required to decode an RSA key.

scalability

A measure of how well the software or hardware product is able to adapt to future business needs.

SHA

See **Secure Hash Algorithm**.

Secure Hash Algorithm

An algorithm that assures data integrity by generating a 160-bit cryptographic message digest value from given data. If as little as a single bit in the data is modified, the Secure Hash Algorithm checksum for the data changes. Forgery of a given data set in a way that will cause the Secure Hash Algorithm to generate the same result as that for the original data is considered computationally infeasible.

An algorithm that takes a message of less than 264 bits in length and produces a 160-bit message digest. The algorithm is slightly slower than MD5, but the larger message digest makes it more secure against brute-force collision and inversion attacks.

single key-pair wallet

A **PKCS #12**-format **wallet** that contains a single user **certificate** and its associated **private key**. The **public key** is embedded in the certificate.

single sign-on

The ability of a user to authenticate once, combined with strong authentication occurring transparently in subsequent connections to other databases or applications. Single sign-on lets a user access multiple accounts and applications with a single password, entered during a single connection. Single password, single authentication.

symmetric-key cryptography

Encryption method that uses the same random number (**key**) in conjunction with a mathematical formula to encode and decode data.

trusted certificate

A trusted certificate, sometimes called a root key certificate, is a third party identity that is qualified with a level of trust. The trusted certificate is used when an identity is being validated as the entity it claims to be. Typically, the certificate authorities you trust are called trusted certificates. If there are several levels of trusted certificates, a trusted certificate at a lower level in the certificate chain does not need to have all of its higher level certificates verified again.

wallet

Also called a digital wallet. A wallet is a data structure used to store and manage security credentials for an individual entity. It implements the storage and retrieval of credentials for use with various cryptographic services. A **wallet resource locator** (WRL) provides all the necessary information to locate the wallet.

wallet resource locator

A wallet resource locator (WRL) provides all necessary information to locate a wallet. It is a path to an operating system directory that contains a wallet.

WRL

See **wallet resource locator**.

X.509

Public keys can be formed in various data formats. The X.509 v3 format is one such popular format.

Index