Oracle9*i*AS Single Sign-On

Administrator's Guide

Release 2 (9.0.2)

January 2002

Part No. A96115-01

Oracle9*i*AS Single Sign-On Administrator’s Guide, Release 2 (9.0.2)

Part No. A96115-01

Primary Author: Henry Abrecht

Graphic Artist: Valarie Moore

Contributors: Gaurav Bhatia, Kamalendu Biswas, Margaret Chou, John Heimann, Mike Hwa, Pei-fung Lam, Harish Singh Rawat, Saurabh Shrivastava, Arun Swaminathan

Contents

List of Figures

List of Tables

Send Us Your Comments

Oracle9*i*AS Single Sign-On Administrator's Guide, Release 2 (9.0.2)

Part No. A96115-01

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: infodev_us@oracle.com
- FAX: (650) 506-7227 Attn: Server Technologies Documentation Manager
- Postal service:
  Oracle Corporation
  Server Technologies Documentation
  500 Oracle Parkway, Mailstop 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

Oracle9iAS Single Sign-On Administrator's Guide provides administrators concepts and procedures for managing user authentication to Oracle9*i* Application Server (*i*AS).

This preface covers the following topics:

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

Audience

Oracle9*i*AS Single Sign-On Administrator's Guide is intended for the following users:

- administrators charged with configuring and managing authentication to Oracle9*i*AS
- developers of products enabled for Oracle Single Sign-On, particularly those who want to integrate these products with mod_osso
- anyone who wants to understand how to protect access to a Web application server, using Oracle Single Sign-On

This document assumes that the reader is familiar with Oracle9*i*AS and has successfully installed, or is able to successfully install, Oracle9*i*AS, Release 2.

Organization

This document contains:

Chapter 1, "Single Sign-On Basics"

This chapter provides a conceptual overview of Oracle9*i*AS Single Sign-On. It examines Single Sign-On components and processes.

Chapter 2, "Administering Oracle Single Sign-On"

This chapter examines essential administration tasks such as enabling applications for Oracle Single Sign-On, assigning administrative privileges, and configuring session timeouts.

Chapter 3, "Directory-Enabled Single Sign-On"

This chapter examines the role that Oracle Internet Directory plays in Single Sign-On. The directory is the repository for the Single Sign-On user name and password. As such, it plays a key role in user management.

Chapter 4, "Single Sign-On Using Digital Certificates"

This chapter explains how to make Oracle Single Sign-On even more secure, by using X.509 certificates over Secure Sockets Layer.

Chapter 5, "Third-Party Single Sign-On"

This chapter explains how to integrate Oracle Single Sign-On with a third-party Single Sign-On system. By working in tandem with Oracle Single Sign-On, the third-party system gains access to the Oracle9*i*AS product complement.

Chapter 6, "Mobile Single Sign-On"

This chapter provides a conceptual overview of wireless Single Sign-On. This option may be chosen when Oracle9*i*AS is installed.

Chapter 7, "Monitoring the Single Sign-On Server"

This chapter explains how to use the GUI tool Oracle Enterprise Manager to monitor server load and user activity.

Chapter 8, "Customizing the Single Sign-On Interface"

This chapter explains how Single Sign-On pages are invoked; then it explains how to rework these pages to suit enterprise needs.

Related Documentation

For more information, see these Oracle resources:

- *Oracle9iAS Single Sign-On Application Developer's Guide*
- *Oracle Internet Directory Administrator's Guide*
- *Oracle9iAS Wireless Getting Started and System Guide*
- *Oracle9iAS Wireless Developer's Guide*

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 username 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.
`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]`

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

Convention	Meaning	Example

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

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	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 Oracle 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`

Convention	Meaning	Example
`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.

1 Single Sign-On Basics

Oracle9*i*AS Single Sign-On is a component of Oracle9*i* Application Server (*i*AS) that enables users to log in to all features of the *i*AS product complement, as well as to other Web applications, using a single user name and password.

Oracle9*i*AS Single Sign-On provides the following benefits:

- Reduced administrative costs

  The Single Sign-On server eliminates the need to support multiple accounts and passwords

- Convenient login

  Users do not have to maintain a separate user name and password for each application that they access

- Increased security

  When a password is required only once, users are less likely to use simple, easily exposed passwords or to write these passwords down

This chapter covers the following topics:

- Oracle9iAS Single Sign-On Components
- Oracle9iAS Single Sign-On Processes
- Single Sign-On Authentication Repository

Oracle9*i*AS Single Sign-On Components

Oracle9*i*AS Single Sign-On consists of the following components:

- Single Sign-On Server
- Partner Applications
- External Applications
- Mod_osso
- Single Sign-On Software Development Kit

Single Sign-On Server

The Single Sign-On server consists of program logic in the Oracle9*i*AS database that enables users to log in securely to Single Sign-On applications such as expense reports, mail, and benefits. These applications take two forms: partner applications and external applications. In both cases, the user gains access to several applications by authenticating only once.

Partner Applications

Oracle9*i*AS applications delegate the authentication function to the Single Sign-On server. For this reason, they are called partner applications. Either an HTTP authentication module called mod_osso or the Single Sign-On Software Development Kit (SDK) enables these applications to accept HTTP headers in lieu of a user name and password once the user has logged into the Single Sign-On server.

A partner application is responsible for determining whether a user authenticated by Oracle9*i*AS Single Sign-On has the requisite privileges for using the partner application. It also controls user access within the application.

Examples of partner applications include Oracle9*i*AS Portal, Oracle9*i*AS Discoverer, and the Single Sign-On server itself.

External Applications

External applications do not delegate authentication to the Single Sign-On server. Instead, they display HTML login forms that ask for application user names and passwords.

Examples of external applications that use HTML login forms include Oracle Mobile and Yahoo! Mail. A unique username and password may be required to access each external application.

The Single Sign-On server can be configured to provide user names and passwords to external applications on behalf of users once they are logged into the Single Sign-On server. Users have the option of storing credentials for external applications in the Single Sign-On database. The Single Sign-On server uses the Single Sign-On user name to transparently locate and retrieve application names and passwords and log the user in to the application. To save an application user name and password, the user selects the Remember My Login Information For This Application check box when first logging in to the application.

Mod_osso

Mod_osso is an HTTP module that provides authentication to *i*AS applications. It is an alternative to the Single Sign-On SDK, used in earlier releases of Oracle 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 *i*AS applications. Thus, the administrator for these applications is spared the burden of integrating them with the SDK.

After authenticating the user, mod_osso transmits the simple header values that *i*AS applications require to validate the user. These include the following:

- user name
- user DN
- user GUID
- language and territory

For details about the attributes that the Single Sign-On server passes to mod_osso in the URLC token, see Table 3–1, "User Attributes Passed to Partner Applications" in Chapter 3, "Directory-Enabled Single Sign-On." To learn how to develop applications using mod_osso, see Chapter 2 in *Oracle9iAS Single Sign-On Application Developer's Guide.*

Single Sign-On Software Development Kit

The Single Sign-On SDK can be used to create partner applications. As such, the SDK is an alternative to mod_osso. Note though that mod_osso is easier to integrate with the Single Sign-On server because developers need not incorporate Single Sign-On application programming interfaces (APIs) into applications. The SDK consists of PL/SQL and Java APIs as well as sample code that demonstrates how these APIs might be implemented.

> **See Also:** Chapter 3, "Single Sign-On Software Development Kit", in *Oracle9iAS Single Sign-On Application Developer's Guide*

Oracle9*i*AS Single Sign-On Processes

This section describes the following Oracle9*i*AS Single Sign-On processes:

- Accessing the Single Sign-On Server
- Authentication Flow for Oracle Single Sign-On
- Accessing an External Application
- Single Sign-Off
- Global User Inactivity Timeout

Accessing the Single Sign-On Server

Nonadministrative users first gain access to the Single Sign-On server by entering the URL of a partner application such as Oracle9*i*AS Portal or Oracle9*i*AS Discoverer. Entering such a URL invokes the Single Sign-On login screen. Once they have entered the correct user name and password, users can gain access to other partner applications and to external applications without having to provide credentials again.

Administrative users can access the adminstrative home page for Single Sign-On by typing a URL of the following form:

```
http://host:port/pls/Single_Sign-On_DAD
```

where *host* is the computer where 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 Single Sign-On schema.

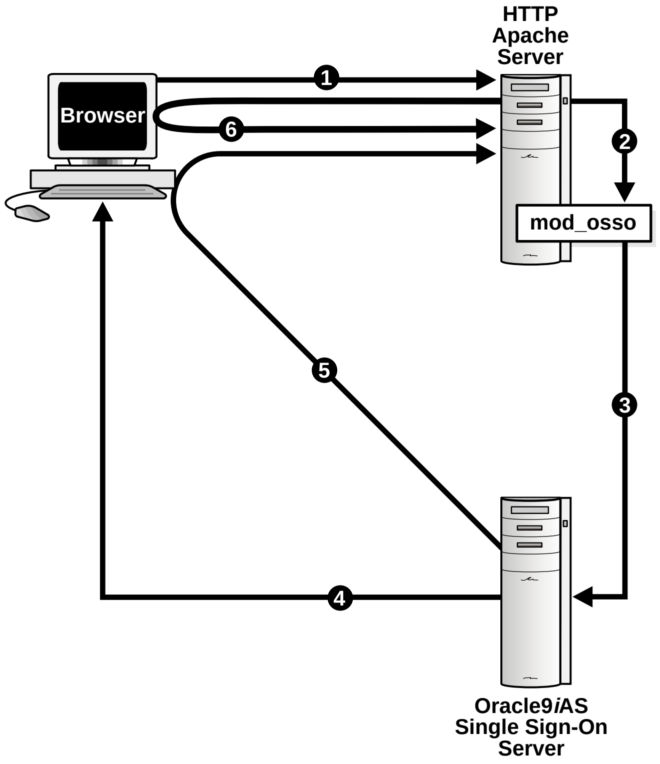
The default DAD is `orasso`.

See Also: "Accessing the Single Sign-On Administration Pages" in Chapter 2, "Administering Oracle Single Sign-On"

Authentication Flow for Oracle Single Sign-On

Figure 1–1 shows what happens when a user requests the URL for a partner application using the Oracle HTTP authentication module mod_osso.

Figure 1–1 Single Sign-On With mod_osso



1. The user requests a URL through a Web browser
2. The Web server looks for a mod_osso cookie for the user. If the cookie exists, the Web server extracts the user's identity and credentials and uses this information to log the user in to the requested application.
3. If the cookie does not exist, the Web server redirects the user to the Single Sign-On server.
4. The Single Sign-On server looks for its own cookie in the browser. If it finds none, it tries to authenticate the user. If authentication is successful, the Single Sign-On server creates a cookie in the browser as a reminder that the user has been authenticated.
5. The Single Sign-On server returns the user's encrypted identity and credentials to the Web server.
6. The Web server creates its own cookie for the user in the browser and redirects the user to the requested URL.
7. The second authentication call to mod_osso is approved.

If, during the same session, the user again seeks access to the same or to a different partner application, the user is not prompted for a username and password, because the application can obtain this information from the mod_osso session cookie.

Mod_osso redirects the user to the Single Sign-On server only if the requested URL is protected. Some protected applications may have public URLs. These require no redirection to the Single Sign-On server.

Accessing an External Application

External applications are accessible only through Oracle9*i*AS Portal, a Single Sign-On partner application.

This section contains these topics:

- Accessing the External Applications Portlet in Oracle9iAS Portal
- Authenticating to an External Application for the First Time
- Authenticating to an External Application After the First Time

Accessing the External Applications Portlet in Oracle9*i*AS Portal

In the lower left corner of the Portal home page, select External Applications Portlet; then, from the list of external applications that appears, select an application.

Authenticating to an External Application for the First Time

Selecting an external application in the External Applications portlet begins the external application login procedure. Oracle Single Sign-On uses the following process if the user is accessing an external application for the first time.

1. The external application login procedure checks the Single Sign-On server password store for the user's credentials for the requested external application. If it finds that the user has no such credentials, the Single Sign-On server prompts the user for them.
2. The user enters the user name and password. The user can save these credentials in the Single Sign-On server password store by selecting the Remember My Login Information check box on the application login screen.
3. If the user elects to save the credentials in the Single Sign-On server password store, the server uses these credentials to construct a login form to submit to the login processing routine for the external application. This routine has been preconfigured by the Single Sign-On server administrator and is associated with the requested application.
4. The Single Sign-On server sends the form to the client browser, with a directive to post it immediately to the external application.

5. The client posts the form to the external application and logs the user in.

If the user declines to save her credentials in the Single Sign-On password store, she must enter a user name and password each time she logs in to the application.

Authenticating to an External Application After the First Time

If the user saved her credentials when accessing an external application for the first time, The Single Sign-On server retrieves her credentials for her during subsequent logins. The process works like this:

1. The user selects one of the links in the External Applications portlet of Oracle Portal.
2. The external application login procedure checks the password store for the user's credentials.
3. The Single Sign-On server finds the user's credentials and uses them to construct a login form to submit to the login processing routine for the external application. This routine has been preconfigured by the Single Sign-On server administrator and is associated with the requested application.
4. The Single Sign-On server sends the form to the client browser, with a directive to post it immediately to the external application.
5. The client posts the form to the external application and logs in.

Single Sign-Off

Single Sign-On users can terminate a Single Sign-On session and log out of all active partner applications simultaneously by logging out of whatever application they are working in. Selecting Logout in a partner application takes them to the Single Sign-Off page, where logout occurs.

If Single Sign-Off has been successful, each of the applications listed on the Single Sign-Off page will have a check mark next to the application name. A broken image next to an application name denotes an unsuccessful logout.

Once all of the application names activated in a session have a check mark, users can select Return to go to the application from which they initiated logout.

Changing the Single Sign-On Password

The Single Sign-On password screen appears only when a password has expired, or is about to expire, and the user tries to log in. If the password is still valid, the user has the option of selecting the Cancel button on this screen and proceeding with the login.

To change or reset a password under other circumstances, the nonadministrative user must go to Delegated Administration Service (DAS), a service of Oracle Internet Directory that performs user and group management functions. The Single Sign-On Administrator can change a password at any time by selecting the Change Password link on the Single Sign-On home page.

The DAS home page can be accessed at a URL of the following form:

```
http://host:port/oiddas/
```

where *host* is the name of the computer where the DAS server is located, and *port* is the port number of this server. The DAS host computer and Single Sign-On host computer are generally the same.

Note: Unlike Single Sign-On user names, Single Sign-On passwords are case sensitive and must conform with rules for Oracle Internet Directory.

Global User Inactivity Timeout

The Oracle 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 Single Sign-Out session timeout.

If the Single Sign-On server is enabled for the global user inactivity timeout, it sets the domain cookie and the user authentication time after authenticating the user. The cookie is encrypted with a single shared key and the Single Sign-On server and mod_osso know the value of the key. Because mod_osso updates the timestamp of the cookie, applications can determine whether a user has timed out because of inactivity.

When the user exceeds the global user inactivity timeout limit and tries to access the application, the application sends the Single Sign-On server an authentication request as usual. The Single Sign-On server, ascertaining that the user has exceeded the global user inactivity timeout limit, prompts the user to log in. If the user has not exceeded the limit, the Single Sign-On server authenticates him using the existing session cookie.

Note: The user might have a valid Single Sign-On session, but if he has exceeded the global timeout limit, he will be prompted for credentials.

See Also: "Configuring the Global User Inactivity Timeout" in Chapter 2, "Administering Oracle Single Sign-On"

Single Sign-On Authentication Repository

In Oracle9*i*AS, Release 2, Single Sign-On authentication is directory based, and user names and passwords are managed not in the Single Sign-On database schema, but in Oracle Internet Directory.

Formerly, local authentication meant using a lookup table in the Single Sign-On schema, located in the database associated with Oracle9*i*AS Portal. In Oracle9*i*AS, Release 2, local authentication is synonymous with authentication using Oracle Internet Directory.

Both Single Sign-On and Oracle Internet Directory are included in an *i*AS Infrastructure installation. To learn how to install the Oracle9*i*AS infrastructure, see "Oracle9*i*AS Infrastructure Installation," in Chapter 4 of *Oracle9i Application Server Installation Guide.*

2 Administering Oracle Single Sign-On

This chapter describes GUI-based and command-line methods for administering and configuring the Single Sign-On Server and applications enabled by the server. It also describes how to grant administrative privileges.

The chapter covers the following topics:

Default Single Sign-On Schemas

When Oracle9*i*AS is installed, five Single Sign-On schemas are created by default. Table 2–1 lists and describes these schemas.

Table 2–1 Single Sign-On Schemas Created by Default

Schema	Description
`ORASSO`	The schema in which the Single Sign-On server is installed.
`ORASSO_PUBLIC`	The schema used to execute the procedures that display Single Sign-On server pages.
`ORASSO_DS`	The schema for the Oracle9*i*AS Discoverer password store. This product has its own store because it accesses different databases.
`ORASSO_PA`	The schema used to generate the configuration table for partner applications when these applications are registered.
`ORASSO_PS`	The schema for the Single Sign-On password store.

The Single Sign-On Administrator’s Role

When the Single Sign-On server is accessed for the first time, only one Single Sign-On administrator exists: orcladmin, the *i*AS superuser. The person installing *i*AS selects the password for this user at install time. The orcladmin account is used to create other accounts, including accounts for iASAdmins, the group that administers Single Sign-On.

Single Sign-On administrators have full privileges for the Single Sign-On server. Using the Single Sign-On administration pages, they can do the following:

- Configure Single Sign-On server settings.
- Administer partner applications. These are Web applications that delegate authentication to the Single Sign-On server.
- Administer external applications. These are Web applications that perform their own authentication using HTML forms.

Granting Privileges to Single Sign-On Administrators

To exercise their privileges, Single Sign-On administrators must be members of the administrative group iASAdmins. This means that an existing member of this group must add a new administrator to it. The Single Sign-On server becomes a member of the group iASAdmins when the server is installed. Use the GUI tool Oracle Directory Manager (ODM) to assign administrative privileges.

To grant administrative privileges to an existing user:

1. Start Oracle Directory Manager. To learn how to start this tool, see "Using Oracle Directory Manager" in Chapter 4 of *Oracle Internet Directory Administrator's Guide.*
2. Log in to ODM as orcladmin, the directory superuser. You must use the password that was assigned to this user when Oracle Internet Directory was installed.

   Note: The directory superuser orcladmin is not the same as the *i*AS superuser orcladmin. These are separate, hierarchically unequal accounts.

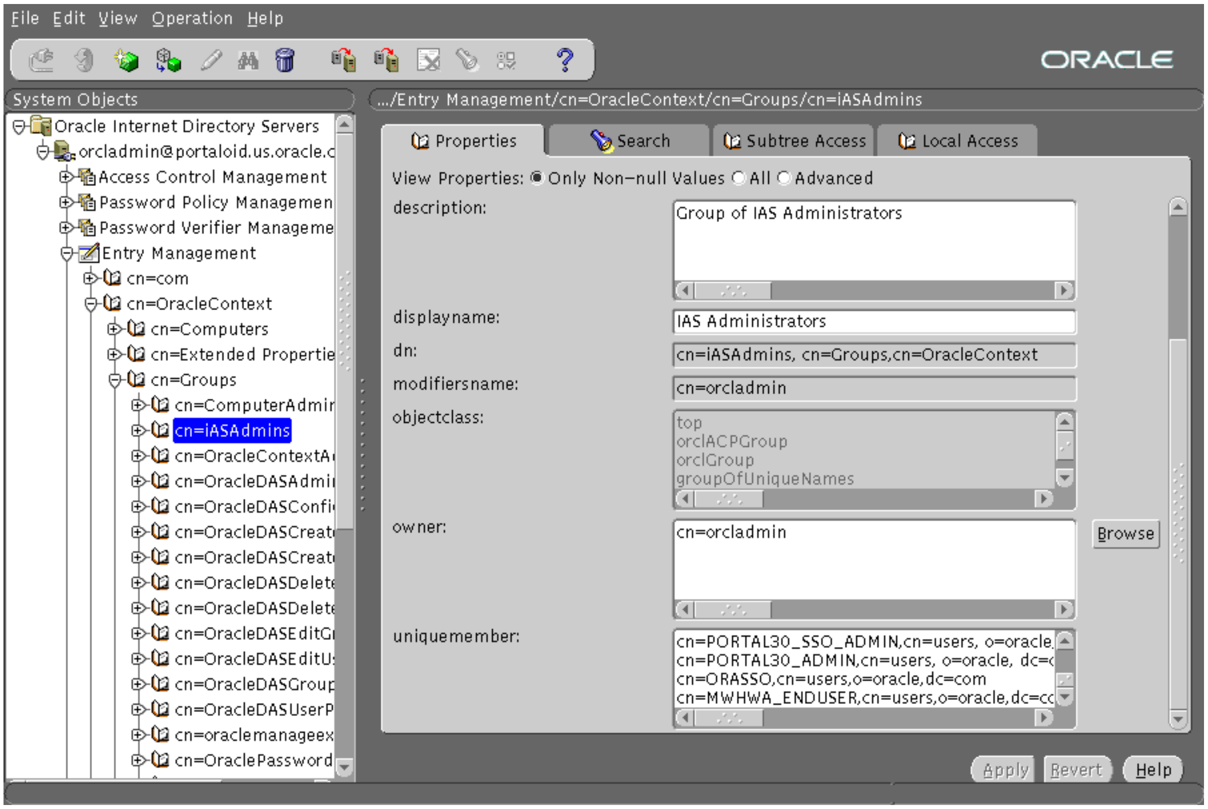
3. In the **System Objects** frame of ODM, select in succession the following entries:
   - Entry Management
   - cn=OracleContext
   - cn=Groups
   - cn=iASAdmins
4. In the **uniquemembers** text box of the **iASAdmins** tab, add an entry for the user. Use the following format:

   ```
   cn=user,cn=users,o=subscriber,dc=com
   ```

   uniquemembers is an attribute of the entry iASAdmins. As such it defines members of the group iASAdmins.
5. Select Apply

Figure 2–1 on page 2-4 reproduces the interface for granting administrative privileges.

Figure 2–1 iASAdmins Tab of Oracle Directory Manager



.../Entry Management/cn=OracleContext/cn=Groups/cn=iASAdmins

- Oracle Internet Directory Servers
  - orcladmin@portaloid.us.oracle.c
    - Access Control Management
    - Password Policy Managemen
    - Password Verifier Manageme
    - Entry Management
      - cn=com
      - cn=OracleContext
        - cn=Computers
        - cn=Extended Propertie
        - cn=Groups
          - cn=ComputerAdmir
          - cn=iASAdmins
          - cn=OracleContextA
          - cn=OracleDASAdmi
          - cn=OracleDASConfi
          - cn=OracleDASCreat
          - cn=OracleDASCreat
          - cn=OracleDASDelete
          - cn=OracleDASDelete
          - cn=OracleDASEditGı
          - cn=OracleDASEditU:
          - cn=OracleDASGroup
          - cn=OracleDASUserP
          - cn=oraclemanageex
          - cn=OraclePassword

View Properties: [✓ Only Non-null Values] All Advanced

description:	Group of IAS Administrators
displayname:	IAS Administrators
dn:	cn=iASAdmins, cn=Groups,cn=OracleContext
modifiersname:	cn=orcladmin
objectclass:	top orclACPGroup orclGroup groupOfUniqueNames
owner:	cn=orcladmin

uniquemember:	cn=PORTAL30_SSO_ADMIN,cn=users, o=oracle,
	cn=PORTAL30_ADMIN,cn=users, o=oracle, dc=
	cn=ORASSO,cn=users,o=oracle,dc=com
	cn=MWHWA_ENDUSER,cn=users,o=oracle,dc=cc

New users are created with a Web tool called Delegated Administration Service (DAS). This tool is accessed at a URL of the following form:

`http://`*`host`*`:`*`port`*`/oiddas/`

Accessing the Single Sign-On Administration Pages

Single Sign-On administrative functions are performed through the Single Sign-On home page.

To access the Single Sign-On home page:

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 Single Sign-On schema. The default DAD is `orasso`.

   The **Access Partner Applications** page appears.

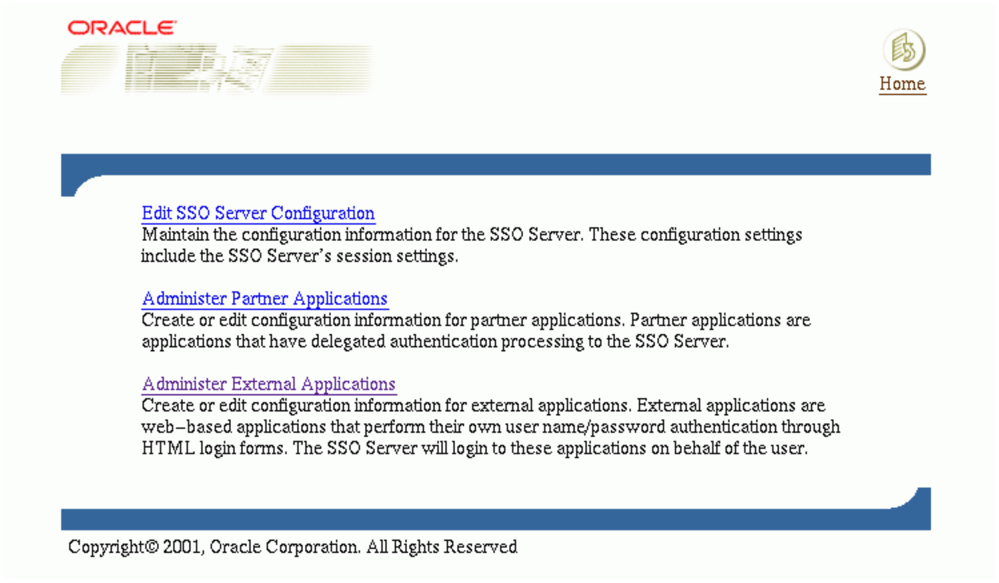
2. Select Login in the upper right corner of the **Access Partner Applications** page.

   The Single Sign-On **Login** page appears.

3. Enter your user name and password and then select the **Login** button.
4. The Single Sign-On home page appears. To perform administrative functions, select the **SSO Server Administration** link.

Figure 2–2 on page 2-6 reproduces the **SSO Server Administration** page.

Figure 2–2 SSO Server Administration Page



Edit SSO Server Configuration
Maintain the configuration information for the SSO Server. These configuration settings include the SSO Server's session settings.

Administer Partner Applications
Create or edit configuration information for partner applications. Partner applications are applications that have delegated authentication processing to the SSO Server.

Administer External Applications
Create or edit configuration information for external applications. External applications are web–based applications that perform their own user name/password authentication through HTML login forms. The SSO Server will login to these applications on behalf of the user.

Copyright© 2001, Oracle Corporation. All Rights Reserved

Edit SSO Server Page

The Edit SSO server page is used to fix the length of Single Sign-On sessions, to verify IP addresses, to enable users to choose territory and language, and to retrieve information about the authentication repository.

To access the Edit SSO Server page, select the link Edit SSO Server Configuration on the SSO Server Administration page.

The Edit SSO Server page contains the following headings and fields:

Table 2–2 SSO Session Policy

Field	Description
Single Sign-On session duration	Enter the number of hours a user can be logged in to the Single Sign-On server without having to time out and log in again.
Verify IP addresses for request made to the Single Sign-On server	Select to verify that the IP address of the browser is the same as the IP address in the authentication request to the Single Sign-On server.

Note: The fields under the heading Authentication Mechanism cannot be modified through the user interface. Instead, see "Changing Single Sign-On Server Settings in the Directory" in Chapter 3, "Directory-Enabled Single Sign-On."

Table 2–3 Territory Selection

Field	Description
Enable users to choose territory	Select to enable users to choose their geographical location and language when they log in. Localization settings such as date, currency, and decimal formats are determined by territory settings.

Administering Partner Applications

The Administer Partner Applications page, accessible as a link on the SSO Server Administration page, is used to add, edit, or delete a partner application. Within the Administer Partner Applications page are links for adding and editing applications.

This section covers the following topics:

- Adding a Partner Application
- Editing a Partner Application

Adding a Partner Application

Selecting the Add Partner Applications link takes you to the Create Partner Applications page. Use the fields on this page, described in the tables immediately following, to register an application with the Single Sign-On server.

Note: If mod_osso authentication is used, mod_osso is the only application that must be registered. If SDK authentication is used, each Single Sign-On application must be registered.

Table 2–4 Partner Application Login

Field	Description
Name	Enter a unique name for the partner application.
Home URL	Enter the URL of the home page for the application.

Table 2–4 Partner Application Login

Field	Description
Success URL	Enter the URL to the routine responsible for establishing the partner application's session and session cookies. This routine should redirect the browser to the URL that the user originally requested. The URL must point to a procedure that processes the user identification information from the Single Sign-On server. Include the `http://` prefix in the URL. The following example shows a success URL for Oracle9*i*AS Portal: `http://server.domain.com:5000/pls/DAD/portal.wwsec_app_priv.process_signon`
Logout URL	Enter the URL for the logout routine of the application. The Single Sign-Off page invokes this URL in parallel with others, enabling users to simultaneously log out of the server and active partner applications.

Table 2–5 Valid Login Timeframes

Field	Description
Start Date	Enter the date when users will first be able to access the partner application through the Single Sign-On server. Use the format shown next to the field label.
End Date	Enter the end date when users will last be able to access the partner application through the Single Sign-On server. Use the format shown next to the field label. **Note:** If you leave this field blank, users can log in to the partner application using the Single Sign-On server indefinitely.

Table 2–6 Application Administrator

Field	Description
Administrator E-mail	Enter the e-mail address of the partner application administrator.
Administrator Information	Enter any additional information that you want to include about the partner application administrator.

Use the following steps to add a partner application:

1. From the **Administer Partner Applications** page, select **Add Partner Application**.

   The **Create Partner Application** page appears.

2. In the **Partner Application Login** section, enter the name and URL of the partner application and a success URL. The success URL points to a Web page where the browser should be redirected after a successful login. It must correspond to the procedure that processes the user identification information provided by the Single Sign-On server. Enter the logout URL for the application.

3. In the **Valid Login Timeframe** section, enter the dates when users can log in to the application through the Single Sign-On server. If you leave the **End Date** field blank, users can log in to the application indefinitely.

4. In the **Application Administrator** section, enter the e-mail address and other information for the application contact person or administrator.

5. Select **OK**. The new partner application appears in the **Edit/Delete Partner Application** list on the Partner Application page.

Editing a Partner Application

The Edit Partner Application page is used to edit configuration information for partner applications.

The Edit Partner Application page contains all of the fields that are in the Create Partner Application page, plus five additional fields in the Partner Application Login section. Table 2–7 on page 2-10 describes the additional fields.

Table 2–7 Fields on the Edit Partner Application Page

Field	Description
ID	The ID value is automatically set when a partner application is added. It is used by the Single Sign-On server to identify the partner application.
Token	The token is automatically set when a partner application is added. It is used by the Single Sign-On server to identify the partner application. The partner application must use the application token to identify itself to the Single Sign-On server when requesting authentication.
Encryption Key	The encryption key is automatically set when a partner application is added. When a user tries to log in using Oracle9*i*AS Single Sign-On, the Single Sign-On server generates a token that indicates a user's identity and whether the user has been authenticated. This key is used to encrypt the login cookie.
Login URL	This is the same as the success URL, which sets the application session and cookies.
Single Sign-Off URL	This is the same as the application logout URL.

Use the following steps to edit a partner application:

1. From the **Administer Partner Applications** page, select an application from the list that appears under the heading **Edit/Delete Partner Application**.
2. Select the **Edit** link for that application. This link appears as a pencil icon.
3. On the **Edit Partner Application** page, edit the appropriate field values, as described in Table 2–4 through Table 2-9.
4. Select **Apply** to store changes for the current screen and to display the updated screen. Select **OK** to store all changes and to return to the previous screen.

See Also: "Partner Applications" in Chapter 1, "Single Sign-On Basics"

Administering External Applications

The Administer External Applications page, accessible as a link on the SSO Server Administration page, is used to add, edit, or delete an external application. Within the Administer External Applications page are links for adding and editing external applications.

This section covers the following topics:

- Adding an External Application
- Editing an External Application
- Storing External Application Credentials in the Single Sign-On Database

Adding an External Application

Selecting the Add External Application link takes you to the Create External Application page. This page contains the following headings and fields:

Table 2–8 External Application Login

Field	Description
Application Name	Enter a name that identifies the external application. This is the default name for the external application.
Login URL	Enter the URL to which the HTML login page for the external application is submitted for authentication. For example, the login URL for Yahoo! Mail is `http://login.yahoo.com/config/login?6p4f5s403j3h0`
Username/ID Field Name	Enter the term that identifies the user name or user ID field of the HTML login form for the application. You find this term by viewing the HTML source for the login form. (See the example after the steps immediately following). This field is not applicable if you are using Basic authentication.
Password Field Name	Enter the term that identifies the password field of the HTML login form for the external application. You find this term by the viewing the HTML source for the login form. (See the example after the steps immediately following). This field is not applicable if you are using Basic authentication.

Table 2–9 Authentication Method

Field	Description
Type of Authentication Use	Use the pulldown menu to select the form submission method for the application. This method specifies how message data is sent by the browser. You find this term by viewing the HTML source for the login form. Select one of the following three methods: POST: Posts data to the Single Sign-On server and submits login credentials within the body of the form. GET: Presents a page request to a server, submitting the login credentials as part of the login URL. BASIC AUTHENTICATION: Submits the login credentials in the application URL, which is protected by HTTP Basic authentication

Warning: The Basic authentication method poses a security risk because the user name and password may be stored in clear text in the browser cache and browser URL history.

Table 2–10 Additional Fields

Field	Description
Field Name	Enter the name of any additional fields on the HTML login form that may require user input to log in to the application. This field is not applicable if you are using Basic authentication.
Field Value	Enter a default value for a corresponding field name value, if applicable. This field is not applicable if you are using Basic Authentication.

Use the following steps to add an external application:

1. From the **Administer External Applications** page, select **Add External Application**.

   The **Create External Application** page appears.

2. In the **External Application Login** field, enter the name of the external application and the URL to which the application's HTML login form is submitted or the protected URL to access if you are using Basic authentication.

3. If the application uses HTTP POST or HTTP GET authentication, in the **User Name/ID Field Name** field, enter the term that identifies the user name or user ID field of the external application's HTML login form. You can find the name by viewing the HTML source for the login form.

   If the application uses the Basic authentication method, the **User Name/ID Field** Name should be empty.

4. If the application uses HTTP POST or HTTP GET authentication, in the **Password Field Name** field, enter the term that identifies the password field of the external application. See the HTML source for the login form.

   If the application uses the Basic authentication method, the **Password Field Name** field should be empty.

5. In the **Additional Fields** field, enter the name and default values for any additional fields on HTML login form that may require user input.

   If the application uses the Basic authentication method, these fields should be empty.

6. Select the **Display to User** check box to allow the default value of an additional field to be changed by the user on the HTML login form.

7. Select **OK**. The new external application appears under the heading **Edit/Delete External Application** on the **Administer External Applications** page, along with the other external applications.

8. Select the link for the application to test the login.

The following example shows the source of the values that are used for Yahoo! Mail.

```
<form method=post action="http://login.yahoo.com/config/login?6p4f5s403j3h0"
autocomplete=off name=a>
...
<td><input name=login size=20 maxlength=32></td>
....
<td><input name=passwd type=password size=20 maxlength=32></td>
...
<input type=checkbox name=".persistent" value="Y" >Remember my ID & password
...
</form>
```

The source provides values for the following:

- Login URL: `http://login.yahoo.com/config/login?6p4f5s403j3h0`
- Username/ID Field Name: `login`
- Password Field Name: `passwd`
- Type of Authentication Used: `POST`
- Field Name: `.persistent Y`
- Field Value: `[off]`

Editing an External Application

Selecting the pencil icon next to an application takes you to the Edit External Application page, where you can edit the values that you entered when you added the application. When you are finished editing, select Apply to enter the changes and to redisplay the page with the updated values.

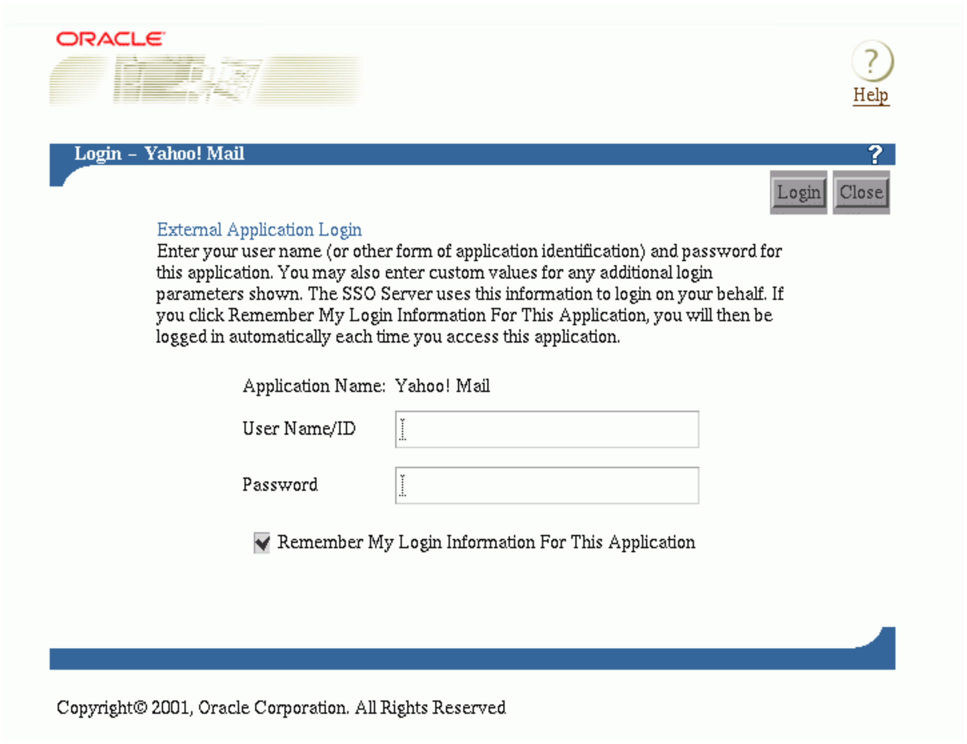
Storing External Application Credentials in the Single Sign-On Database

Each external application expects to receive a user name and password each time the user logs in to the application. To enable single sign-on to these applications, users are given the option of storing their credentials in the Single Sign-On database when they log in to the application.

If Single Sign-On users are logging in to an external application for the first time, they are presented with the External Application Login page. After entering credentials they can select the check box Remember My Login Information for This Application. If they choose this option, the next time they access the application, the Single Sign-On server logs in on their behalf.

Figure 2–3 reproduces the External Application Login page

Figure 2–3 External Application Login Page



External Application Login

Enter your user name (or other form of application identification) and password for this application. You may also enter custom values for any additional login parameters shown. The SSO Server uses this information to login on your behalf. If you click Remember My Login Information For This Application, you will then be logged in automatically each time you access this application.

Application Name: Yahoo! Mail

User Name/ID

Password

✓ Remember My Login Information For This Application

Copyright© 2001, Oracle Corporation. All Rights Reserved

Note: If you change your password for an external application, you must also update your password on the External Application Login page. If you neglect to do so, this page returns an error message when you attempt to log in.

See Also: "External Applications" in Chapter 1, "Single Sign-On Basics"

Changing Passwords

Single Sign-On users and administrators can change their passwords at any time by selecting the Change Password link on the SSO Administration page.

To access the SSO Administration page, user and administrator alike must enter the the URL for the Single Sign-On home page, as explained in "Accessing the Single Sign-On Administration Pages".

The administrator can alter the Change Password page to suit his tastes by following the procedures in Chapter 8, "Customizing the Single Sign-On Interface". For rules governing passwords, see "Password Policies" in Chapter 3, "Directory-Enabled Single Sign-On".

Configuring National Language Support

The Single Sign-On administrator can enable users to select from a number languages, including Chinese and other languages encoded in multibyte character sets. These languages can be selected when *i*AS is installed. If the user chooses a language, territories associated with that language appear at login. This feature enables the user to choose localization settings such as date, currency, and decimal formats.

Users select a particular language at login time. When they select a language, The server sets a persistent cookie in the user's browser. This cookie retains the user's chosen language between sessions. The default language is English.

At the same time, the Single Sign-On server passes the user's language preference as an attribute (`accept_language`) to partner applications. This attribute enables these applications to set up sessions in the user's chosen language.

If the Single Sign-On administrator has selected the Enable Users to Choose Territory check box on the Edit SSO Server page, links for territories associated with a chosen language appear in the Single Sign-On interface. These links can be used to specify localization settings such as date, currency, and decimal formats.

Configuring the Global User Inactivity Timeout

The global user inactivity timeout is not configured by default. You must enable it by running the script ssogito.sql.

To configure the global user activity timeout:

1. Log in to SQL*Plus using the Single Sign-On schema name and password. The default in both cases is `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—say, 15 minutes.
6. To enable the new settings, select the return key. To cancel the transaction, select the return key twice.

   Once you have completed a transaction, the script furnishes you a summary of the new timeout settings.
7. In the file mod_osso.conf, make sure that the parameter `ossoIdleTimeout` exists and that it is set to `on`. The path to this file is as follows:

   ```
   IAS_HOME/Apache/mod_osso/conf/mod_osso.conf
   ```

See Also: "Global User Inactivity Timeout" in Chapter 1, "Single Sign-On Basics"

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 "Using Secure Sockets Layer (SSL) to Authenticate Users" in Chapter 3 of *Oracle9i Application Server Security Guide*.
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 Oracle HTTP 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
   ```

Protecting URLs Accessed by Oracle9*i*AS Portal and Mod_osso

Some Single Sign-On URLs must be configured so that only browsers on Oracle9*i*AS Portal and mod_osso hosts can access them. Oracle Portal must be able to show the authenticated user a list of links to external applications. A mod_osso host configured for extended Basic authentication must be able to access these applications. In both cases, the dads.conf file must be modified.

For Oracle Portal, enter the following lines:

```
<Location /pls/orasso/orasso.wwsso_app_admin.external_apps_list*>
  Order deny,allow
  Deny from all
  Allow from <oracle_portal_host>
</Location>

<Location /pls/orasso/orasso.wwsso_app_admin.print_fapp_username*>
  Order deny,allow
  Deny from all
  Allow from <oracle_portal_host>
</Location>
```

For mod_osso, enter the following lines:

```
<Location /pls/orasso/orasso.wwsso_app_admin.get_ext_app*>
  Order deny,allow
  Deny from all
  Allow from <mod_osso_host>
</Location>
```

3 Directory-Enabled Single Sign-On

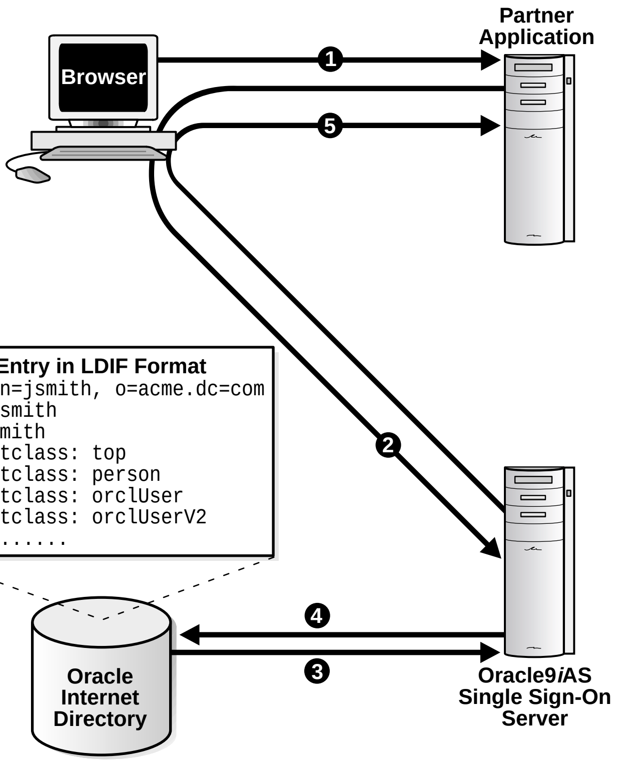
This chapter examines those aspects of Oracle9*i*AS Single Sign-On that are dependent upon Oracle Internet Directory. The directory is the repository for all Single Sign-On user accounts and passwords—administrative and nonadministrative. In Oracle 9*i*AS, Release 2, all user and group management functions for Single Sign-On are handled by the directory.

The chapter covers the following topics:

Authentication Flow for Directory-Enabled Single Sign-On

Figure 3–1 illustrates how Single Sign-On authentication works when Oracle Internet Directory is the authentication repository.

Figure 3–1 Directory-Enabled Single Sign-On


Partner Application
Browser
1
5
2
User Entry in LDIF Format
4
3
Oracle Internet Directory
Oracle9*i*AS Single Sign-On Server

```
dn: cn=jsmith, o=acme.dc=com
cn: jsmith
sn: Smith
objectclass: top
objectclass: person
objectclass: orclUser
objectclass: orclUserV2
...........
```

1. The user attempts to access a protected URL.
2. The user is redirected to the Single Sign-On server.
3. The user's credentials are verified against his user entry in the directory. Specifically, the user's nickname—the name he enters on the Single Sign-On login page—is mapped to his distinguished name (DN) in the directory; then his password, an attribute of his entry, is validated. Once his password is validated, he is authenticated.

4. The Single Sign-On server fetches the user's attributes, specifically his globally unique user ID (GUID) and distinguished name (DN).
5. The Single Sign-On server passes the fetched attributes, as well as the user's nickname and language preference to the partner application.
6. The partner application may fetch other attributes using attributes that have already been fetched—the user's GUID, for instance.

Managing Users in Oracle Internet Directory

Management functions for Single Sign-On users are performed with the following tools:

- Delegated Administration Service (DAS)

  DAS is a self-service application that enables administrators to manage users and groups. For example, you can create and delete Single Sign-On users and change passwords.

  You can access DAS with a URL of the following form:

  ```
  http://host:port/oiddas/
  ```

  where *host* is the name of the computer on which the DAS server is located, and *port* is the port number of the server. In a typical *i*AS infrastructure installation, the host for the Single Sign-On Server and the DAS server are the same.

  > **See Also:** "Managing Users, Groups, and Subscribers by Using the Delegated Administration Service" in Chapter 9 of *Oracle Internet Directory Administrator's Guide*

- Oracle Directory Manager (ODM)

  ODM is a Java-based GUI tool for managing most functions in Oracle Internet Directory. Use it to configure password policies.

- LDAP Command-Line Tools

  You can use command-line tools like `ldapmodify` in place of DAS and Oracle Directory Manager. These tools operate on text files. They take arguments that use the Lightweight Directory Interchange (LDIF) format.

See Also:

- Chapter 4, "Directory Administration Tools" in *Oracle Internet Directory Administrator's Guide*
- Appendix A, "Syntax for LDIF and Command-Line Tools" in *Oracle Internet Directory Administrator's Guide*

Single Sign-On User Accounts

In Oracle9*i*AS, user accounts are stored and managed in Oracle Internet Directory. This means that Single Sign-On authentication is performed in the directory, against the user's entry. When the client requests an application and is redirected to the Single Sign-On server, the server validates the user's credentials against Oracle Internet Directory. This validation involves verifying the user's password and, also, his account status—for instance, whether the user is locked out or whether his password is about to expire.

If the user is successfully authenticated, the Single Sign-On server passes her nickname— which is typically her user name—her globally unique user ID (GUID), her distinguished name (DN), and her language preference to the partner application.

Table 3–1 lists all of the user attributes that the Single Sign-On server sends to partner applications using the URLC token. Note that some of these attributes are stored in the Single Sign-On database.

Table 3–1 User Attributes Passed to Partner Applications

Attribute	Description	Source
`ssousername`	User nickname as entered by user on Single Sign-On login page	Single Sign-On login page
`user_dn`	Single Sign-On user's distinguished name	User entry in Oracle Internet Directory
`user_guid`	Single Sign-On user's globally unique user ID (GUID)	Single Sign-On user's globally unique user ID (GUID)
`accept_language`	Language and territory. User selects these on the login page	Single Sign-On server

Table 3–2 on page 3-5 lists the Java functions that partner applications can use to retrieve attributes from the HTTP headers set by mod_osso. Non-Java applications can also make function calls to retrieve attributes.

Table 3–2 Functions Used to Retrieve Attributes from HTTP Headers

Attribute	Function
`ssousername`	HTTPServletRequest.getRemoteUser()
`nls_info`	HTTPServletRequest.getHeader("Accept-Language")
`user_dn`	HTTPServletRequest.getHeader("Osso-User-Dn")
`user_guid`	HTTPServletRequest.getHeader("Osso-User-Guid")

Password Policies

The Single Sign-On user password is stored in Oracle Internet Directory as an attribute of the user's entry. Users can change their passwords either in the Single Sign-On interface or by going to DAS. Oracle Directory Manager enables the directory administrator to adjust password expiry behavior to suit enterprise needs.

Password Rules

Oracle Directory Manager has fields that enable the administrator to do the following when configuring password behavior:

- specify the minimum number of characters a password requires
- specify the minimum number of numeric characters a password requires
- determine whether an old password can be used as a new one

Password Expiry

Using either Oracle Directory Manager or LDAP command-line tools, administrators can configure password life and can specify when users are prompted to change their passwords. Administrators can also configure a grace login period for users. This is a period after which the user's password has expired. If the user neglects to change his password within this period, he must have an administrator reset it for him.

Account Lockout

An account lockout occurs when users are unable to access the Single Sign-On server from any number of workstations because they have submitted the incorrect user name and password combination more times than is permitted by Oracle

Internet Directory. Once the limit has been reached, even a valid user name and password combination fails to log the user in.

Because Single Sign-On user accounts are managed in the directory, the directory administrator determines account lockout policies. Oracle Directory Manager has fields for enabling and disabling lockout and for specifying lockout duration.

Configuring Password Policies

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*.

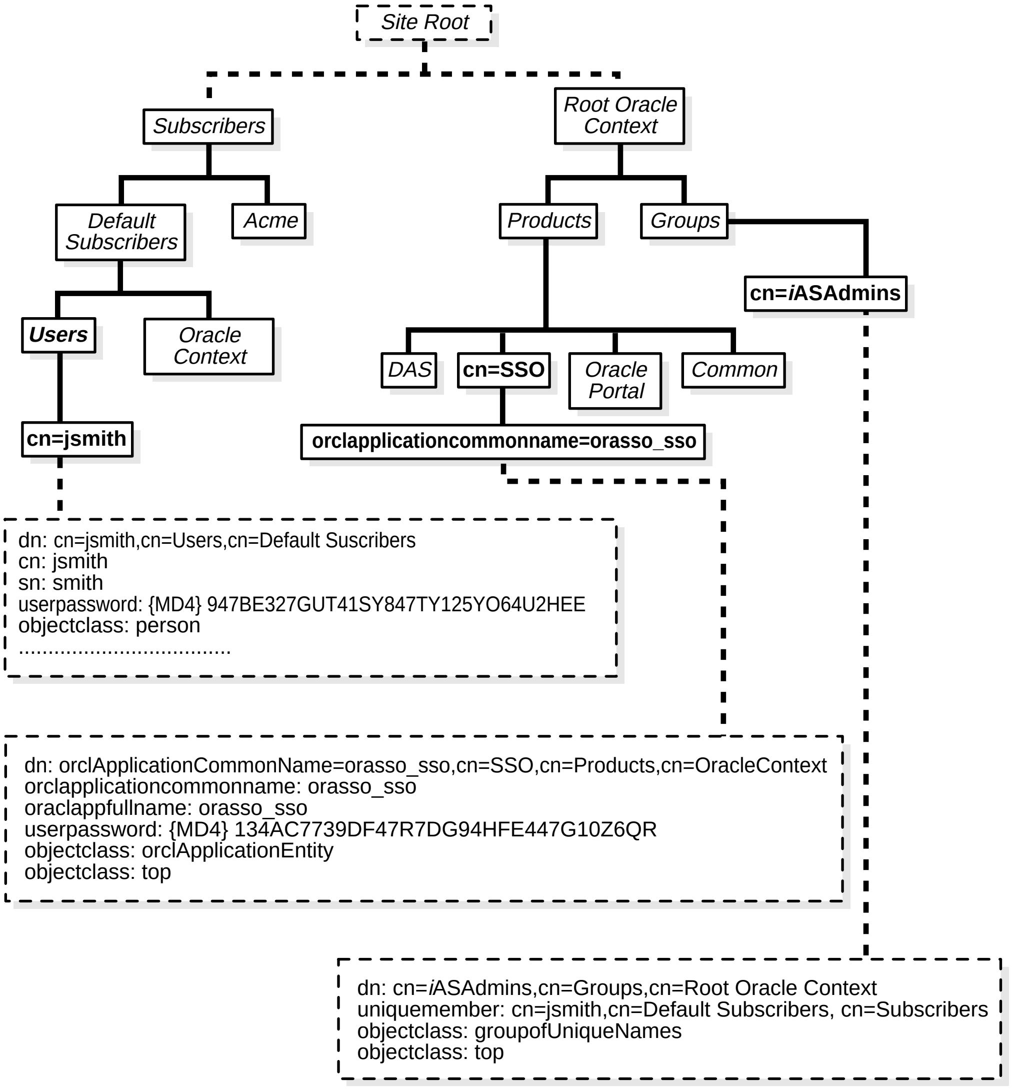
Directory Tree for Oracle Single Sign-On

Oracle Single Sign-On, like other components in the *i*AS complement, has its own "container" within the directory information tree (DIT). This container is found within the Oracle Context, an entry that serves as the root for all Oracle-specific data. In the simplified DIT shown in Figure 1-2, only the root Oracle Context is expanded. The root Oracle Context is the repository for sitewide information—that is, information that applies to all subscribers and products. Structurally, subscriber-specific Oracle Contexts are mirror images of the root context, but the information they contain pertains only to a particular subscriber.

In Figure 3–2 on page 3-7 , the Single Sign-On container is identified by the entry `cn=SSO`. It contains a single entry, `orclApplicationCommonName=orasso_sso`, which is the entry for the Single Sign-On server. In the illustration, this entry has been expanded to show the object classes and attributes that define the entry. For example, the attribute `orclapplicationcommonname` gives the default name for the Single Sign-On server, `orasso`. Note, too, that the Single Sign-On server has its own password, which, along with `orclapplicationcommonname`, the directory server uses to authenticate the Single Sign-On server when the latter performs user searches.

The container `Common` is a repository for information common to all *i*AS products. For instance, it houses attributes that enable products to identify the subscriber search base, or node, and the subscriber nickname. Subscriber-specific `Common` containers—not shown here—contain attributes that enable products to locate users within a subscriber subtree. In addition to expanding the SSO container, the illustration expands entries for an *i*AS user who is also an administrator.

Figure 3–2 Directory Information Tree for Oracle Single Sign-On



dn: cn=jsmith,cn=Users,cn=Default Suscribers
cn: jsmith
sn: smith
userpassword: {MD4} 947BE327GUT41SY847TY125YO64U2HEE
objectclass: person
....

dn: orclApplicationCommonName=orasso_sso,cn=SSO,cn=Products,cn=OracleContext
orclapplicationcommonname: orasso_sso
oraclappfullname: orasso_sso
userpassword: {MD4} 134AC7739DF47R7DG94HFE447G10Z6QR
objectclass: orclApplicationEntity
objectclass: top

dn: cn=*i*ASAdmins,cn=Groups,cn=Root Oracle Context
uniquemember: cn=jsmith,cn=Default Subscribers, cn=Subscribers
objectclass: groupofUniqueNames
objectclass: top

See Also:

- Chapter 14, "Oracle Components and Oracle Internet Directory" in *Oracle Internet Directory Administrator’s Guide*
- Appendix C, "Schema Elements" in *Oracle Internet Directory Administrator’s Guide*

Changing Single Sign-On Server Settings in the Directory

The script ssooconf.sql enables the Single Sign-On administrator to change the following settings in the directory:

- directory host name
- directory port
- DN for Single Sign-On server
- password for Single Sign-On Server
- SSL connections to the directory

Note: A new instance of Oracle Internet Directory must be a replicated instance.

To change directory settings for the Single Sign-On server:

1. Log in to SQL*Plus as the Single Sign-On schema. The default user name and password is `orasso`.
2. Run the script ssooconf.sql by issuing the following command:

   SQL> @ssooconf.sql
3. In the fields prefaced by the words **Enter value for**, make the desired changes.
4. To update the file, select **Return**.

   The script displays updated settings for the Single Sign-On server.

If you run the script and then decide not to make changes, select **Return** to retain existing values.

4 Single Sign-On Using Digital Certificates

Single Sign-On using simple authentication poses a serious security risk. If hackers can access a Single Sign-On account, they can access numerous applications and do untold damage. Fortunately, Single Sign-On users have the option of using digital certificates instead of the Single Sign-On 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).

This chapter covers the following topics:

- Benefits of Certificate-Enabled Single Sign-On
- Authentication Options
- Authentication Flow in Certificate-Enabled Single Sign-On
- System Requirements
- Configuring Single Sign-On for Certificates
- Troubleshooting Certificate-Enabled Single Sign-On
- Maintaining a Certificate Revocation List

Benefits of Certificate-Enabled Single Sign-On

Certificate authentication offers the benefit that partner applications are, by default, PKI enabled when the Single Sign-On Server is PKI enabled. As in Single Sign-On without certificates, these applications rely on cookies for authentication. Retrieving and checking a cookie requires less computing overhead than performing an SSL handshake. For Web applications characterized by short-lived sessions, the result is a significant improvement in server performance and throughput.

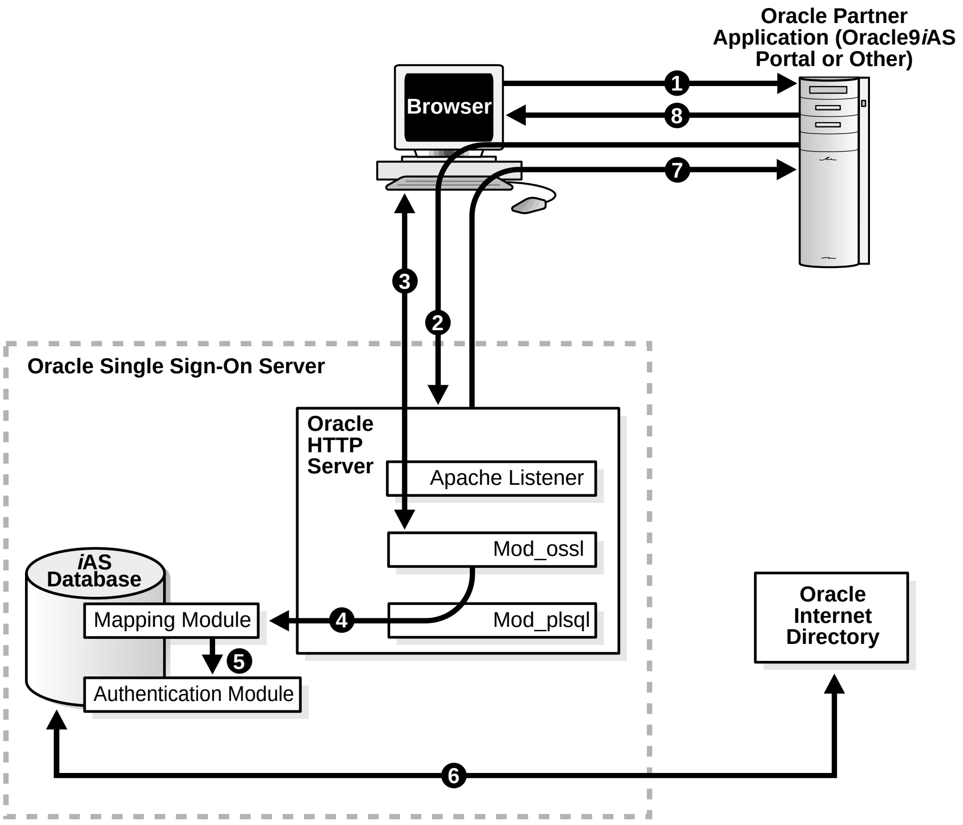
Authentication Options

Oracle 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.

Authentication Flow in Certificate-Enabled Single Sign-On

Figure 4–1 on page 4-3 depicts the authentication flow for certificate-enabled Single Sign-On.

Figure 4–1 Certificate-Enabled Single Sign-On



1. The Single Sign-On user tries to access a partner application.
2. The partner application redirects the user to the Single Sign-On server for authentication. The server must be configured so that, at the very least, it asks for optional client certification.
3. If the server has been configured for SSL, the SSL handshake occurs. The browser prompts the user for a password that opens the browser certificate and key database. The browser then sends the user’s certificate to the login URL of the Single Sign-On server.
4. Mod_ossl, the SSL module on the Single Sign-On server, passes the user’s certificate information to a PL/SQL module that maps the user’s nickname and, optionally, his subscriber name.

5. The mapping module passes the mapped user name and optional subscriber name to an authentication module.
6. The authentication module uses the mapped user name to retrieve the user certificate stored in the directory; then it compares this certificate with the browser certificate received from mod_ossl.
7. If the two certificates match, the Single Sign-On server signifies that authentication has been successful by passing a URLC token that contains user information to the application.
8. The application redirects the user to the requested URL, which then delivers content.

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 client certificate DN must be chosen in such a way that it contains some information about the Single Sign-On 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 on the Single Sign-On server.
- The certificate of the server certificate issuer must be installed as a trusted certificate in the user's browser

Configuring Single Sign-On for Certificates

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)
- 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 4–1:

Table 4–1 HTTP 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 like this:

```
## 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 "/ade/lkethana_iasdemo/oracle/work/Apache/Apache/htdocs"
ServerName eastsun5.us.oracle.com
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>
```

Mod_plsql

Configuring the plsql module for certificate-based authentication 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 4–2.

Table 4–2 dads.conf Environment Variables

Variable	Description
`SSL_CLIENT_S_DN_CN`	The user's nickname, represented by the attribute `cn` (common name)
`SSL_CLIENT_S_DN_O`	The subscriber name associated with the user, if any. The subscriber name is represented by the attribute `O` (organization)
`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    host:port:database_sid
  PlsqlDatabasePassword         orasso
  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_CN,SSL_CLIENT_S_DN_O,SSL_CLIENT_
  S_DN, SSL_CLIENT_CERT
</Location>

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

User Name Mapping Module

The user name mapping module, which consists of the files ssodnmap.pks and ssodnmap.pkb, 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. The files ssodnmap.pkb and ssodnmap.pks are reproduced here for those who want to customize the module.

```
/*  Copyright (c) Oracle Corporation 2001. All Rights Reserved. */

create or replace package body wwsso_map_dn
as

/* Global Variable */
g_user_certificate_dn  varchar2(2000) default null;

/**
 * This function reverses order of a given DN
 */
function reverse_dn(p_dn in varchar2 )
 return varchar2
is
 l_rdns   dbms_ldap.string_collection;
 l_rdn    varchar2(500);
 l_rev_dn varchar2(2000);
 l_index  pls_integer;
begin
 if(p_dn is null) then
     return null;
 end if;
 l_rdns  :=  dbms_ldap.explode_dn(p_dn, 0);
 l_index := l_rdns.first;
 loop
     exit when l_index is null;
     l_rdn    := l_rdns(l_index);
     l_rev_dn := l_rdn || ',' ||l_rev_dn;
     l_index  := l_rdns.next(l_index);
  end loop;
  l_rev_dn := substr(l_rev_dn, 0, length(l_rev_dn) -1);
  return l_rev_dn;
end reverse_dn;


/**
 * Get header list
 */
procedure get_headers
(
  p_header_list out header_collection
)
as
begin
  /* Set header name for certificate DN */
  p_header_list.delete;
  p_header_list(0) := 'SSL_CLIENT_S_DN';
end get_headers;

/**
 * Set value for given headers
 */
procedure set_header_values
(
   p_header_value_list in header_value_collection
)
as
begin
   /* Set certificate DN value */
   g_user_certificate_dn := p_header_value_list(0);
end set_header_values;

/**
 * Map SSO user from user certificate information
 *
 */
 procedure map_name
 (
    p_sso_user        out varchar2
  , p_subscriber      out varchar2
 )
 as
 begin
   /* You may change this default implementation according to your
    * requirements.
    *
    * In this simple mapping example, we will assume that
    * user certificate DN is same as directory DN
    */

   -- SSO USER INFORMATION --
   -- Set SSO User DN  from certificate attribute
   -- Note: certificate DN value send b yApache/mod_ossl is just
   -- reverse order of directory DN
   p_sso_user := reverse_dn(g_user_certificate_dn);

   -- SUBSCRIBER INFORMATION --
   -- Set subscriber name to null
   -- Note: subscriber information for this user can be retrieved
   --       from directory
   p_subscriber := null;

 end map_name;

end wwsso_map_dn;
/

show errors package body wwsso_map_dn;



/*  Copyright (c) Oracle Corporation 2001. All Rights Reserved. */
create or replace package wwsso_map_dn
as
/**
 * ssodnmap.pks
 *
 * Description:
 *    This package maps SSO username and subscriber name
 *    from user certificate information
 *
 * Configuration setup:
 *
 *  1. A copy of user certificate for this user should be
 *     stored  the userCertificate directory attribute for successful
 *     authentication
 *
 *  2. SSO DAD configuration(dads.conf) should have following entry:
 *      PlsqlCGIEnvironmentList = SSL_CLIENT_S_DN_CN, SSL_CLIENT_S_DN_O,
        SSL_CLIENT_S_DN, SSL_CLIENT_CERT
 *
 *  3. SSO DAD should be able to get SSL variables with following
 *     Apache configuration setting(dads.conf file):
 *
 *     <IfDefine SSL>
 *      <Location /pls/<sso_dad_name>>
 *        SSLOptions +ExportCertData +StdEnvVars
 *      </Location>
 *     </IfDefine>
 *
 *  Please refer to the SSO Server administration guide for detailed
 *  information
 */
```

```
/**
 *  Exception list
 */

 CERT_DN_MAPPING_EXCEPTION    exception;
 INVALID_HEADER_EXCEPTION     exception;

/**
 * These data structures are used to hold a list of values
 */

  type header_collection is table of varchar2(100)
    index by binary_integer;

  type header_value_collection is table of varchar2(5000)
      index by binary_integer;


/**
 *  Get header list
 *  This procedure returns required CGI headers for name mapping
 *
 * <template>
 *   wwsso_map_dn.get_headers(
 *      p_header_list   ==> -- out header_collection
 *      );
 * </template>
 *
 * <code>
 *   wwsso_map_dn.get_headers
 *   (
 *      p_header_list   ==> -- out header_collection
 *   );
 * </code>
 *
 *  @return p_header_list => list of headers
 *
 *  This procedure is used to specify required CGI variable names
 *  necessary for mapping module to map SSO user and subscriber name
 */
procedure get_headers
(
  p_header_list out header_collection
);
```

```
/**
 *  Set value for given headers
 *  This procedure sets required CGI headers  necessary
 *  for name mapping
 *
 * <template>
 *   wwsso_map_dn.set_header_values(
 *      p_header_value_list  ==> -- out header_value_collection
 *      );
 * </template>
 *
 * <code>
 *   wwsso_map_dn.set_header_values
 *   (
 *      p_header_value_list   ==> -- out header_value_collection
 *   );
 * </code>
 *
 *  @return p_header_value_list  => list of header values
 *
 *  This procedure is used set required CGI header values
 *  necessary for mapping module to map SSO user and subscriber name
 */

procedure set_header_values
(
   p_header_value_list in header_value_collection
);

 /**
 * Map user and subscriber name
 *
 * This procedure will map SSO username and subscriber name
 * from user certificate information, received from Apache/mod_ossl.
 *
 * <template>
 *   wwsso_map_dn.map_name(
 *      p_sso_user   ==> -- out varchar2
 *      p_subscriber ==> -- out varchar2
 *      );
 * </template>
 *
 * <code>
 *   wwsso_map_dn.map_name
 *   (
 *      p_sso_user    ==> Mapped SSO user name
 *    , p_subscriber  ==> Mapped subscriber name
 *   );
 * </code>
 *
 * @return     p_sso_user   => Mapped sso username
 * @return     p_subscriber => Mapped subscriber name
 *
 * @exception  CERT_DN_MAPPING_EXCEPTION
 *             This exception is raised if user information
 *             can not be mapped correctly
 *
 * If browser certificate DN is same as directory DN then
 * p_sso_user should be set to the directory DN and
 * subscriber name should be set to null since subscriber
 * information will be retrieved from directory.
 * Please note that the certificate DN received from
 * Apache/mod_ossl may be in the reverse order of directory DN
 *
 * If SSO username is mapped from certificate DN and mapped
 * subscriber name is set to null then the default
 * subscriber will be used to authenticate the user
 *
 * User certificate received from Apache/mod_ossl
 * will be checked against certificate stored in the
 * userCertificate directory attribute of the mapped
 * SSO user.
*/

 procedure map_name
 (
    p_sso_user        out varchar2
  , p_subscriber      out varchar2

 );
end wwsso_map_dn;
/

show errors package wwsso_map_dn;
```

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 or by the Oracle 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.

To determine whether a self-service application is feasible, the directory administrator can try to add the certificate to the directory as an LDIF file, using the command-line tool `ldapmodify`.

The syntax for this command is as follows:

```
ldap -h host -p 389 -D "cn=directory_administrator" -w password -f file_
name.ldif
```

In the example LDIF file that follows, the certificate of user jsmith is represented as an attribute of his entry in the directory. The attribute type is `usercertificate`. The attribute value is the long string that follows.

```
dn: cn=jsmith,o=oracle,dc=com
changetype: modify
replace: usercertificate
usercertificate::MIIC3TCCAkYCAgP3MA0GCSqGSIb3DQEBBAUAMIG8MQswCQ
    YDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEXMBUGA1UEBxMOUmVkd29vZCBTaG9yZXMxGz
    AZBgNVBAoTEk9yYWNsZSBDb3Jwb3JhdGlvbjEfMB0GA1UECxMWV2ViIFNpbmdsZSBTaWduLU9uLC
    BTVDEeMBwGA1UEAxMVQ2VydGlmaWNhYoEHmF4gomtc4mxSKh/zAgMBAAEwDQYJKoZIhvcNAQEEBQ
    ADgYEAKwXoCLDRqmK1Y9LQtIjLnCaIJKUZmS1Qj+bhu/IHeZLGHg4TJg3O2XVA5u/VxwjLeGBqLX
    y2z7o3RujNKx2CVx6p/0Hkjnw4w6KVau2hcBgC9m4kzUGhHJ9b65v/zx7dIUKyJr4RF+lJhJg4/o
    YXxLrYHp5NAkHP4htT0gqCXiI=
```

Because it is a non-ASCII value, the certificate must be encoded in base 64 format, as shown here. Unlike other attributes, a base 64 attribute requires a double colon (::) as a delimiter. Note, too, that the use of a tab enables a base 64 attribute to be folded.

Single Sign-On Server

To enable the SSO 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 HTTP 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.

Troubleshooting Certificate-Enabled Single Sign-On

Use Table 4–3 on page 4-17 to identify and solve problems that you are likely to face when configuring Oracle Single Sign-On for certificates.

Table 4–3 Troubleshooting Checklist for Certificate-Enabled Single Sign-On

Problem	Cause	Solution
The user tries to access a partner application over SSL and is presented with the error message "Network Error: Connection Refused"	The `SSLEngine on` parameter is missing from the httpd.conf file or has not been entered correctly	Add the missing parameter as specified in "Oracle HTTP Server (SSL)". If the parameter is present and is entered correctly, see if the Apache error log file identifies the problem. This file can be found at `IAS_HOME/Apache/Apache/logs/error_log`
The Single Sign-On login page fails to prompt the user for a certificate	The SSLVerifyClient optional parameter is missing from the httpd.conf file or has not been entered correctly	Add the missing parameter as specified in "Oracle HTTP Server (SSL)". If the parameter is present and is entered correctly, see if the Apache error log file identifies the problem. This file can be found at `IAS_HOME/Apache/Apache/logs/error_log`
Certificate authentication fails to work, and the user is presented the login page.	This problem has three possible causes: **1.** PlsqlCGIEnvironmentList variables are missing from the dads.conf file **2.** PlsqlCGIEnvironmentList variables are being used improperly in the user name mapping module **3.** The user's certificate is missing from the directory or has been entered incorrectly	**1.** Add the missing variables to the dads.conf file as specified in the section "Mod_plsql". **2.** Check the formatting and spelling of variables in the file ssodnmap.pkb. This file is reproduced in the section "User Name Mapping Module". **3.** Reenter the user's certificate into the directory, as specified in the section "Oracle Internet Directory"

Maintaining a Certificate Revocation List

To ensure that users are unable to log in using invalid or expired certificates, the administrator must maintain an up-to-date certificate revocation list (CRL) on the Oracle HTTP server, and the CA that issued the certificate must provide this list. The file ca-bundle.crl can be used to maintain the list. At any rate, the path for the CRL file used must be as follows:

```
/private/oracle/Apache/Apache/conf/ssl.crl
```

For details about implementing and maintaining a CRL, see comments in the httpd.conf file. The file is reproduced in the section "Oracle HTTP Server (SSL)".

5 Third-Party Single Sign-On

This chapter explains how to integrate Oracle Single Sign-on with third-party single sign-on products. It describes how third-party integration works; then it presents the application programming interfaces (APIs) for integration. Finally, it presents sample code that integrates Oracle9*i*AS Single Sign-On with SiteMinder®, a single sign-on product from Netegrity, Inc.

An enterprise that has a third-party system in place can gain access to the Oracle 9*i*AS suite by using APIs that enable the Oracle Single Sign-On server to act as an authentication gateway between the third-party system and Oracle applications.

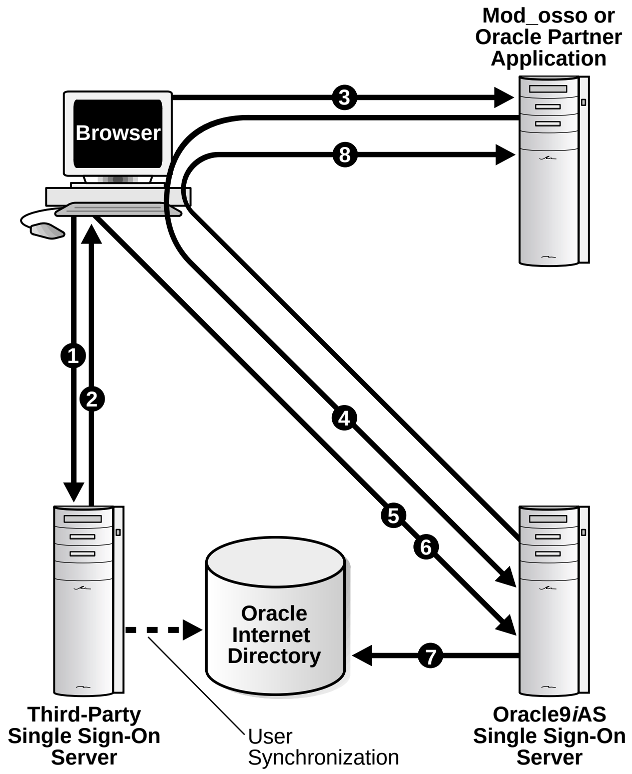
The chapter covers the following topics:

How Third-Party Single Sign-On Works

In third-party single sign-on, the Oracle Single Sign-On server, the third-party single sign-on server, and the partner application form a chain of trust. The Oracle Single Sign-On server delegates authentication to the third-party single sign-on server, becoming essentially a partner application to it. Oracle applications continue to work only with the Oracle Single Sign-On server and are unaware of the third-party single sign-on server. Implicitly, however, they trust the third-party server.

For Oracle Single Sign-On to issue users an authentication token under this arrangement, the third party single sign-on server must pass the former the user's identity by setting HTTP headers. Once it obtains the user's identity, the Oracle Single Sign-On server functions as before, managing user accounts, checking account policies, auditing, generating tokens, and redirecting users to its partner applications. Figure 5–1 on page 5-3 illustrates the process.

Figure 5–1 Authentication Flow in Third-Party Single Sign-On



1. The user logs in to the third-party single sign-on server.
2. If login is successful, the third-party server sets a token in the user's browser.
3. The user attempts to access an Oracle partner application.
4. The partner application, ignorant of the third-party server, redirects the user to the Oracle Single Sign-On server. At the same time, the user passes the third-party single sign-on token to the Oracle Single Sign-On server.
5. The Single Sign-On server looks for its own cookie.
6. Failing to find its cookie, the Oracle Single Sign-On server looks for a token from the third-party single sign-on server.
7. Finding the token, the Oracle Single Sign-On server retrieves user attributes from Oracle Internet Directory.

8. The Oracle Single Sign-On server sets its own cookie and redirects the user back to the Oracle partner application, passing a URL token that contains the user's attributes.

Notes:

- In the case of users who try to access a partner application before logging in to the third-party single sign-on server, the Oracle Single Sign-On server can be configured to redirect users to the third-party login page. For more information, see "Installing Customized Login, Change Password, and Single Sign-Off Pages", in Chapter 8, "Customizing the Single Sign-On Interface.
- If the single sign-on systems are to be accessible to all authorized users, the user repository must be centralized in one place. This means that, before deployment, users may have to be migrated from the third-party single sign-on repository to the Oracle Single Sign-On repository or the reverse.

Synchronizing the Third-Party Repository with Oracle Internet Directory

The authentication scenario presented in the preceding steps assumes either that the user repository is Oracle Internet Directory or that the repository is a third-party directory or database. If the repository is the latter, the user name must be synchronized with the user entry in Oracle Internet Directory. This synchronization enables the Single Sign-On server to fetch the user attributes required by Single Sign-On-enabled applications.

Note: Third-party single sign-on cannot proceed if the synchronization mechanism is not in place.

To synchronize the third-party repository with Oracle Internet Directory, use either Oracle Directory Integration Platform (DIP) or bulk load tools.

See Also:

- Chapter 28, "Directory Synchronization: Profiles and Connectors" in *Oracle Internet Directory Administrator's Guide*
- Chapter 27, "Oracle Directory Integration Platform Concepts and Components" in *Oracle Internet Directory Administrator's Guide*

Third-Party Integration Modules

To achieve third-party integration, the developer must implement the package body of `wwsso_auth_external`. The package specification is located in the file `ssoauthx.pks`. The required interfaces perform the following functions:

- Authentication Using a Token
- Set External Cookies

Authentication Using a Token

This function is called before a login screen is displayed to the user. If authentication using a token is to be supported, the implementer of this function must return the user name to the Oracle Single Sign-On server by retrieving the user identity in a secure fashion—by looking at a securely set HTTP header, for instance, or at a secure cookie.

```
FUNCTION authenticate_user
(
 p_user OUT VARCHAR2
)
RETURN PLS_INTEGER;

/*The function throws the following exceptions:
EXT_AUTH_FAILURE_EXCEPTION,EXT_AUTH_UNKNOWN_EXCEPTION
EXT_AUTH_SETUP_EXCEPTION
*/
```

Set External Cookies

If authentication is successful, the Oracle Single Sign-On server sets all the cookies provided in the p_cookie_list parameter on behalf of the external authentication server.

```
PROCEDURE set_external_cookies
(
  p_username IN VARCHAR2
 ,p_password IN VARCHAR2
 ,p_cookie_list OUT wwsso_ls_private.cookie_list
);
```

Integration Case Study: Netegrity SiteMinder

SiteMinder by Netegrity, Inc., is a product, which, like Oracle9*i*AS Single Sign-On, offers single sign-on authentication to protected resources. SiteMinder consists of two components: the SiteMinder policy server and the SiteMinder agent. The first provides users with a variety of services including user and session management, authentication, and authorization. The second is located on Web servers and Web application servers. It screens requests for resources and determines whether a resource is protected by SiteMinder.

Customers who have SiteMinder already installed may want to use it to gain access to Oracle9*i*AS applications. They can achieve this access by using APIs that enable SiteMinder to talk to Oracle applications by way of Oracle9*i*AS Single Sign-On.

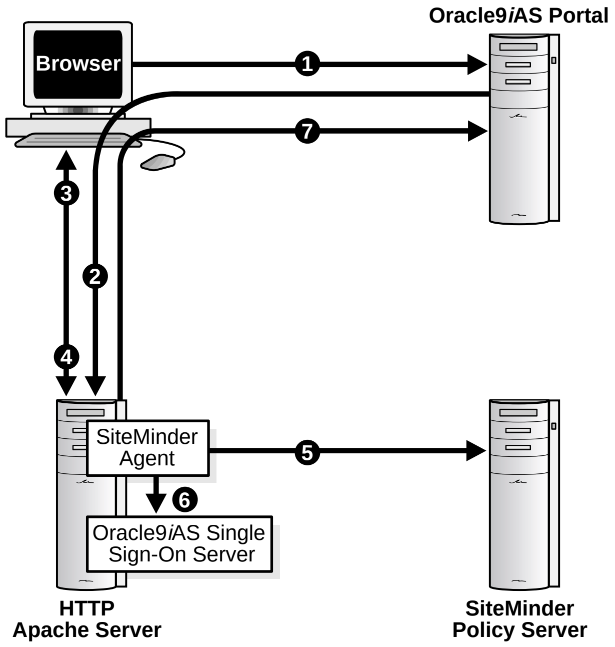
This section covers the following topics:

- Authentication Flow for the SiteMinder Solution
- Logging Out of the Integrated System
- Sample Integration Package
- Installing and Deploying the SiteMinder Solution

Authentication Flow for the SiteMinder Solution

Figure 5–2 on page 5-7 depicts the authentication flow for an integrated Single Sign-On / SiteMinder system. It shows what happens when the user tries to access a partner application—in this case, Oracle9*i*AS Portal—without logging in to SiteMinder first.

Figure 5–2 Authentication Flow for the SiteMinder Solution



1. The user tries to access a protected resource within Oracle Portal.
2. Oracle Portal redirects the user to the Oracle Single Sign-On server.
3. The SiteMinder agent prompts the user for credentials.
4. The user presents his or her credentials to the SiteMinder agent.
5. The SiteMinder agent checks the user's credentials in the SiteMinder policy server. The SiteMinder policy server in turn tries to authenticate the user. If configured to do so, the policy server checks the credentials against Oracle Internet Directory.
6. If authentication is successful, the SiteMinder agent passes the user's identity to the Oracle Single Sign-On server in the form of HTTP headers.
7. The Oracle Single Sign-On server generates a URLC token and uses it to transfer the user's identity to Oracle Portal.

8. If the user in this scenario is already logged in to SiteMinder, steps 3, 4, and 5 are skipped, and the SiteMinder agent sends the user's identity in the form of HTTP headers.

Note: In Oracle9*i*AS v1.0.2.2, the Oracle Single Sign-On user repository need not be Oracle Internet Directory. In Oracle9*i*AS, Release 2, integration with this directory is required.

Logging Out of the Integrated System

The integrated Oracle Single Sign-On / SiteMinder system requires that, when a user logs out of the Oracle Single Sign-On server, he or she is also logged out of SiteMinder. For concurrent logout to occur, the Single Sign-On logout procedure must be registered as a URI with the SiteMinder agent. See Installing and Deploying the SiteMinder Solution for details.

When the Oracle Single Sign-On logout procedure is invoked from Oracle Portal, the SiteMinder agent intercepts the request and ends the SiteMinder session. It then transfers control to the Oracle Single Sign-On logout procedure, which ends the Oracle Single Sign-On session.

Selecting Logout in Oracle Portal initiates the following sequence:

1. Oracle Portal ends the Portal session and redirects the user to the Single Sign-On server with the done_URL for the application home page.
2. The SiteMinder agent intercepts the logout request, contacts the SiteMinder policy server, and ends the SiteMinder session.
3. The SiteMinder agent transfers control to the Single Sign-On logout procedure.
4. The Single Sign-On server ends the Single Sign-On session and redirects the user to the application home URL sent in Step 1.

Before concurrent logout can begin, customer applications must redirect users to the Portal logout link at

http://*host*:*port*/pls/*Portal_DAD*/*Portal_schema*.wwsec_app_priv.logout?p_done_url=*url_encoded_apps_URL*

The done_url of the application might be the following:

```
http%3A%2F%2Fmysite.com/home
```

In this example, users are redirected back to the home page of mysite.com.

Sample Integration Package

The package `ssoxnete.pkb`, presented here, can be used to integrate an existing SiteMinder implementation with Oracle9*i*AS Single Sign-On.

```
Rem ssoxnete.pkb
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      ssoxnete.pkb - Single Sign-On Netegriry SiteMinder Integration
Rem
Rem    DESCRIPTION
Rem      This package body is used to achieve integration with Netegrity
Rem      SiteMinder. It may be customized as required. This is just a default
Rem      implementation and changes might be required based on customer’s
Rem      specific deployment scenario.


CREATE OR replace PACKAGE BODY wwsso_auth_external AS

    GLOBAL_SEPARATOR CONSTANT varchar2(1)    := ’~’;

/* This function needs to be implemented to provide a DN
 * to UID mapping. One way to do this mapping is to lookup
 *  the UID for a given DN in the directory
 */

FUNCTION map_dn_to_uid(p_user_dn IN VARCHAR2)
  return VARCHAR2
IS
BEGIN

  -- NULL implementation by default

  raise EXT_AUTH_FAILURE_EXCEPTION;

  return p_user_dn;

END map_dn_to_uid;
```

```
FUNCTION authenticate_user
  (
   p_user OUT VARCHAR2
  )
  return PLS_INTEGER
IS
 l_http_header varchar(1000);
 l_ssouser wwsec_person.user_name%type := NULL;
BEGIN

   l_http_header := owa_util.get_cgi_env('HTTP_SM_USER');
   debug_print('SiteMinder ID : ' || l_http_header);

  /*
   if l_http_header IS NULL then user may be authenticated by PKI
   in SiteMinder so check the DN header
   */

   IF (l_http_header is NULL) THEN
   BEGIN
       debug_print('check if user authenticated using PKI');
       l_http_header := owa_util.get_cgi_env('HTTP_SM_USERDN');
       l_ssouser := map_dn_to_uid(l_http_header);
   END;
   ELSE
       l_ssouser := l_http_header;
   END IF;

   IF ( (l_ssouser IS NULL) or
       ( INSTR(l_ssouser, GLOBAL_SEPARATOR) != 0) ) THEN
       debug_print('malformed user id: '
               || l_ssouser
               || ' returned by wwsso_auth_external.authenticate_user');
       RAISE EXT_AUTH_FAILURE_EXCEPTION;
   ELSE
     p_user := NLS_UPPER(l_ssouser);
     return 0;
   END IF;


EXCEPTION
   WHEN OTHERS THEN
     debug_print('unknown exception in authenticate_user(p_user)'
                 || sqlerrm);
     RAISE EXT_AUTH_FAILURE_EXCEPTION;
```

```
END authenticate_user;


FUNCTION get_authentication_name
 RETURN VARCHAR2
AS
BEGIN
    RETURN 'Netegrity SiteMinder';
END get_authentication_name;



PROCEDURE set_external_cookies
  (
    p_username IN VARCHAR2
  , p_password IN VARCHAR2
  , p_cookie_list OUT wwsso_ls_private.cookie_list
  )
AS
BEGIN
    null;

END set_external_cookies;



END;
/
show errors;
```

Installing and Deploying the SiteMinder Solution

Perform the following steps to install and configure the Oracle Single Sign-On server with SiteMinder:

1. Install the Oracle Single Sign-On server.
2. Run `ssonete.sql`. This script configures the Oracle Single Sign-On server to operate in external mode and loads the default implementation found in `ssoxnete.pkb`.
3. Install the SiteMinder agent in front of the Oracle Single Sign-On server. This task involves installing mod_sm, a SiteMinder module, on the same instance as the Oracle Single Sign-On server. For more details, see *SiteMinder Agent Operations Guide.*
4. Configure the SiteMinder policy server to protect access to the Oracle Single Sign-On server URLs and to associate a user population with the Oracle Single Sign-On server. To accomplish this task, create policy domains and realms for the Oracle Single Sign-On server on the SiteMinder policy server. The Oracle Single Sign-On server is accessed at URLs of the form:

   ```
   http://hostname:port/pls/Single_Sign-On_server_DAD
   ```

5. If you are using PKI authentication, customize the function `map_dn_to_uid(p_user_dn IN VARCHAR2)`. Currently, this function has a default implementation of NULL, as indicated in `ssoxnete.pkb`.
6. Edit your modplsql DAD file, typically named wbdbsvr.app, to include the following SiteMinder headers:

```
[DAD_Single_Sign-On_server_schema]
connect_string = Single_Sign-On_server_schema_DB_connect_string


..
cgi_env_list = HTTP_SM_USER,HTTP_SM_USERDN
```

Register the Single Sign-Out logout procedure as a URI with the SiteMinder agent. To do this, add the following line to the WebAgent.conf file:

```
logoffuri="/pls/Single_Sign-On_DAD/Single_Sign-On_schema.wwsso_app_admin.ls_
logout"
```

After these steps have been completed, the user can log in to a partner application. Because credentials are stored in a repository managed by SiteMinder, the Change Password page in the Oracle Single Sign-On server can be customized to point to the SiteMinder Change Password screen.

See Also: "Installing Customized Login, Change Password, and Single Sign-Off Pages" in Chapter 8, "Customizing the Single Sign-On Interface

6

Mobile Single Sign-On

Oracle *i*AS users have the option of using mobile, or wireless, devices—specifically personal digital assistants (PDAs) and cellular phones—to gain access to the *i*AS server. As in PC-based systems, the authentication mechanism is Oracle Single Sign-On. The *i*AS-wireless option can be selected when Oracle9*i*AS is installed. If *i*AS-wireless is selected, Portal-to-Go (PTG), the gateway for mobile devices, is registered with the Single Sign-On server automatically.

This chapter covers the following topics:

- iAS-Wireless Concepts and Architecture
- Wireless Single Sign-On
- Wireless Single Sign-Off
- Change Password Page for iAS-Wireless

*i*AS-Wireless Concepts and Architecture

Wireless products communicate with *i*AS using either wireless markup language (WML) or HTML. Cellular phones use the first option, PDAs the second. Because these devices request URLs using Wireless Access Protocol (WAP) and other non-HTTP protocols, hardware gateways must be used to convert messages to HTTP and back again.

The heart of *iAS*-wireless is PTG. It serves as a browser for interactions between the wireless device and the Single Sign-On server and for interactions between the wireless device and Oracle applications. Actually, the PTG server has a fourfold function:

- It authenticates the user directly to the Single Sign-On server
- It serves up private pages of its own
- It serves as a proxy browser for "external," Single Sign-On-protected applications, passing requests to these applications, which then perform single sign-on authentication
- It converts mobile XML to the appropriate device markup language—either WML or HTML

In the PTG framework, external applications are partner applications that are integrated with the Single Sign-On Software Development Kit (SDK). PTG treats these applications as public applications even if they are not. A PTG instance uses an HTTP adapter to serve as a proxy browser for such applications.

Wireless Single Sign-On

The wireless user has two single sign-on authentication options: she can authenticate directly from the PTG home page, or she can request a partner application, which then performs the authentication.

This section covers the following topics:

- Authenticating Through PTG
- Authenticating by Requesting a Partner Application

Authenticating Through PTG

The wireless user can authenticate from the PTG public page either by requesting a private application or by logging in explicitly to the Single Sign-On server.

The sequence is as follows:

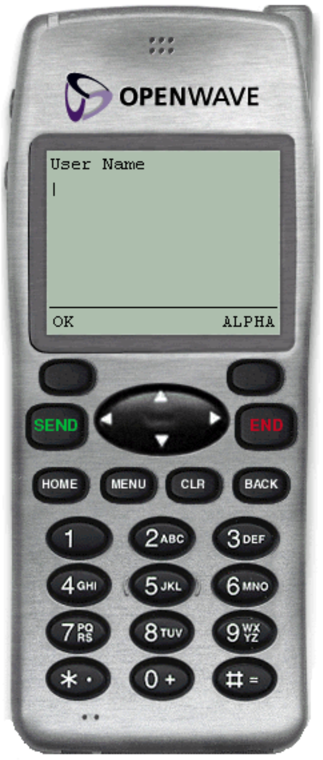
1. The wireless user accesses PTG by entering a URL of the following form:

   `http://host:port/ptg/rm`

   The PTG public page appears. It displays links for public and private PTG applications.

2. The user requests a private application or selects the key icon that invokes the Single Sign-On Login page. The portion of this page where the user enters her name is reproduced in Figure 6–1 on page 6-4.
3. The Single Sign-On server looks for the encrypted Single Sign-On cookie. If the cookie is present, the server uses it to identify the user. The server then sends the Single Sign-On redirect form (Step 7). If the cookie is not present, the server sends the mobile XML login form to PTG.
4. PTG transforms the mobile XML login form to the appropriate markup language and sends the converted form to the device browser.
5. The user submits the login form with her user name and password.
6. PTG forwards the login form to the Single Sign-On server.
7. The Single Sign-On server authenticates the user. If authentication succeeds, the server sends PTG the Single Sign-On redirect form.
8. PTG sends the user her home page or the requested URL.

Figure 6–1 Wireless Single Sign-On Page: User Name Field



User Name

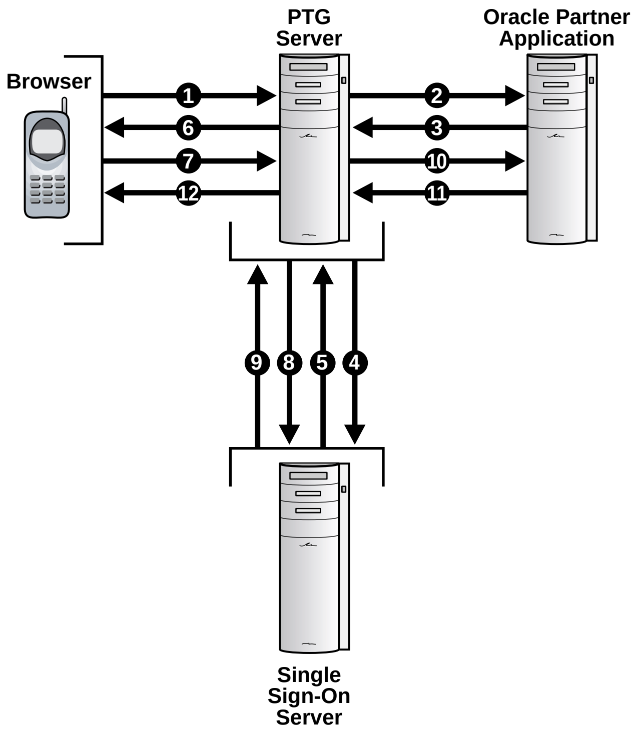
OK ALPHA

Authenticating by Requesting a Partner Application

Using the mobile device, the user may also authenticate to the Single Sign-On server by requesting URLs for other partner applications. In this case, the authentication redirection agent is not PTG, but an application integrated with the Single Sign-On SDK.

Figure 6–2 on page 6-5 illustrates the authentication sequence:

Figure 6–2 Authenticating by Requesting a Partner Application



1. An unauthenticated user requests a partner application.
2. PTG sends the request to the partner application, using an HTTP adapter situated on its back end.
3. If the URL requested is protected, the partner application issues an HTTP redirect to the Single Sign-On server.
4. PTG follows the redirected URL.
5. The Single Sign-On server looks for the encrypted Single Sign-On cookie, which is set in the PTG "browser." If the cookie is present, the server uses it to identify the user. The server then sends the Single Sign-On redirect form (Step 9). If the cookie is not present, the server sends the mobile XML login form to PTG.
6. PTG converts the mobile XML login form to the appropriate markup language and delivers the converted form to the device browser.

7. The user submits the login form with her user name and password.
8. PTG passes the login request to the Single Sign-On server.
9. Upon successful authentication, the Single Sign-On server sends a redirect form that points to the partner application.
10. PTG follows the redirect form. At this point, PTG, knowing that authentication has been successful, updates the user's session.
11. The partner application serves up content in mobile XML.
12. PTG converts the mobile XML content to the appropriate markup language and delivers the converted content to the device browser.

Wireless Single Sign-Off

The Single Sign-off process in wireless is as follows:

1. The user selects the application logout link.
2. PTG forwards the request to the Single Sign-On server.
3. The Single Sign-On server delivers the Single Sign-Off page in mobile XML. Like its PC counterpart, this page contains an image for each of the applications that are active. Unlike its PC counterpart, this page is never seen by the wireless user.
4. PTG sends an HTTP request to each image link to clean up the user's session in each of the applications.
5. PTG terminates the user session.
6. PTG returns the user's home page if the user logged in through PTG. It returns the done_url of the global logout page if the user logged in by requesting a partner application URL.

Change Password Page for *i*AS-Wireless

The wireless user sees only two Single Sign-On pages: the Login page and the Change Password page. Unlike its PC counterpart, the *i*AS-wireless Change Password page appears only when users try to log in to the Single Sign-On server and their password has already expired. Wireless users have no access to the Change Password link on the SSO Administration page.

See Also:

- *Oracle9iAS Wireless Getting Started and System Guide*
- *Oracle9iAS Wireless Developer's Guide*

7 Monitoring the Single Sign-On Server

This chapter explains how to access the GUI tool Oracle Enterprise Manager (OEM) and use it to monitor the Single Sign-On server.

The chapter covers the following topics:

Accessing the Single Sign-On Monitoring Pages

The Single Sign-On Monitoring interface consists of two pages: The home page and the Details of Login Failures page. The first provides general information about server load and user activity. The second provides a login failure profile for a particular user.

To access the home page for Single Sign-On monitoring:

1. Go to the Oracle*i*AS home page by entering a URL of the following form:

   ```
   http://host:port/emd/console/targets
   ```

   where *host* is the name of the computer on which the *i*AS instance resides, and *port* is the port number of the *i*AS server.

2. From the **Application Servers** list on the Oracle9*i*AS home page, choose the appropriate *i*AS instance.

3. From the **System Components** list for the instance that you chose in Step 2, select the Single Sign-On server for that instance.

Interpreting and Using the Home Page

The Home page displays the following metrics under the General heading:

- Total logins
- Number of successful logins
- Number of failed logins
- Number of login failures per user

The statistics displayed are for the previous 24 hours.

In addition, the home page provides information about the location and status of the Single Sign-On server and the database where the Single Sign-On schema resides. This information can be found under the General heading. Table 7–1 on page 7-3 lists and defines the fields that appear under this heading.

Table 7–1 Single Sign-On Server: Location and Status

Field	Description
Database	The connect string for the *i*AS database
Http Server	The name of the HTTP server where the Single Sign-On server resides.
Current Status	How long the Single Sign-On server has been up or down. A green check mark means that the server is up, a read arrow that the server is down.

To view IP addresses and associated failure times for a particular user, the administrator selects a name from the Login Failures During the Last 24 Hours table. He then selects the associated link under the Failures heading. Table 7–2 lists and defines the fields that appear in the Login Failures During the Last 24 Hours table.

Table 7–2 Login Failures During the Last 24 Hours Table

Field	Description
Username	The user trying to log in
Failures	The number of login failures a user has logged during the previous 24 hours. Select this link to view details about each login failure.
Status	The number of login failures in the last 24 hours has reached or is approaching a critical level. This field shows one of three values: ■ Green check mark—below the warning threshold ■ Warning—At or above the warning threshold, but below the critical threshold ■ Critical—Above the critical threshold

Interpreting and Using the Details of Login Failures Page

The Details of Login Failures table displays login failure times and associated IP addresses for a particular user. The statistics displayed are for the previous 24 hours.

See Also: For more information about Oracle Enterprise Manager, see *Oracle9i Application Server Administrator's Guide*

8

Customizing the Single Sign-On Interface

This chapter explains how to incorporate Single Sign-On Login, Change Password, and Single Sign-Off pages customized to match your portal or product. At the end of the chapter are examples of how the three pages might be reworked as JavaServer pages (JSPs).

Customized pages can be any type of Web page: a PL/SQL procedure, a CGI script, or a JSP. With each of these options, the pages must support certain parameters to function properly.

The chapter covers the following topics:

Installing Customized Login, Change Password, and Single Sign-Off Pages

The `WWSSO_LS_CONFIGURATION_INFO$` table in the Single Sign-On schema contains the `LOGIN_URL` column, which is used to enable customized Login, Change Password, and Single Sign-Off pages.

The `LOGIN_URL` column contains three values separated by a space. The first value specifies the URL for the Login page, the second the URL for the Change Password page, and the third the value for the Single Sign-Off page.

By default, the `LOGIN_URL` column contains the values `UNUSED UNUSED UNUSED`, which specify that the Login, Change Password, and Single Sign-Off pages use the standard Single Sign-On pages.

Perform the following steps to install customized Login, Change Password, and Single Sign-Off pages.

1. On the database where the Single Sign-On server is installed, log in to the Single Sign-On schema using SQL*Plus, as in the following example:

   ```
   sqlplus orasso/orasso
   ```

2. Update the `LOGIN_URL` column.

   To replace just the Login page with the customized page, update the first value in the `LOGIN_URL` column, as in the following example:

   ```
   UPDATE WWSSO_LS_CONFIGURATION_INFO$
   SET LOGIN_URL='http://server.domain[:port]/login.jsp UNUSED';
   ```

   To replace just the Change Password page with a customized page, update the second value in the LOGIN_URL column, as in the following example:

   ```
   UPDATE WWSSO_LS_CONFIGURATION_INFO$
   SET LOGIN_URL='UNUSED http://server.domain[:port]/change_password.jsp';
   ```

   To replace just the Single Sign-Off page with a customized page, update the third value in the LOGIN_URL column, as in the following example:

   ```
   UPDATE WWSSO_LS_CONFIGURATION_INFO$
   SET LOGIN_URL='UNUSED UNUSED http://server.domain[:port]/single_sign_
   off.jsp';
   ```

   To replace all three pages, update all three values in the LOGIN_URL column, as in the following example:

```
UPDATE WWSSO_LS_CONFIGURATION_INFO$
SET LOGIN_URL='http://server.domain[:port]/login.jsp
http://server.domain[:port]/change_password.jsp
http://server.domain[:port]/single_sign_off.jsp';
```

3. To revert to using the standard pages, restore the original values, as in the following example:

```
UPDATE WWSSO_LS_CONFIGURATION_INFO$
SET LOGIN_URL='UNUSED UNUSED UNUSED';
```

How Customized Single Sign-On Pages Are Enabled

This section contains the following sections:

- Enabling the Customized Login Page
- Enabling the Customized Change Password Page
- Enabling the Single Sign-Off Page

Enabling the Customized Login Page

When a partner application redirects a user to the Single Sign-On server, the server calls a procedure that creates the Login page.

The process is as follows:

1. The application calls `WWSSO_APP_ADMIN.LS_LOGIN` to authenticate the user.
2. If the user does not yet have an Oracle9*i*AS Single Sign-On session, `LS_LOGIN` calls `WWSSO_LOGIN.DRAW_LOGIN_PAGE` to display the standard login page.
3. `DRAW_LOGIN_PAGE` submits a form to `WWSSO_APP_ADMIN.LS_LOGIN` to process the credentials.
4. If the user is authenticated, `LS_LOGIN` redirects to the application's success URL, which then redirects to the requested application page.

The customized solution provides the option of redirecting to a separate URL to create the login page, instead of making a PL/SQL call to `WWSSO_LOGIN.DRAW_LOGIN_PAGE`. The URL can point to a Java Server Page, a CGI script, or other type of page. The page should process the name of the routine to which the login form is submitted—such as `WWSSO_APP_ADMIN.LS_LOGIN`—and submit the form appropriately.

The flow of logic is as follows:

1. The user selects the Login button on the Login page. Selecting this button invokes the `WWSSO_APP_ADMIN.LS_LOGIN` routine to authenticate the user.
2. If a URL is specified in the `LOGIN_URL` column for displaying the login page, `LS_LOGIN` redirects to that URL.

   If a URL is not specified in the `LOGIN_URL` column, `LS_LOGIN` calls `WWSSO_LOGIN.DRAW_LOGIN_PAGE` to draw the standard login page.
3. The login page submits a form to `WWSSO_APP_ADMIN.LS_LOGIN` to process the credentials.
4. If the user is authenticated, `LS_LOGIN` redirects to the requested application page. The Single Sign-On server uses the `LOGIN_URL` column of the `WWSSO_LS_CONFIGURATION_INFO$` table to store the URL for the customized login page.

Enabling the Customized Change Password Page

The Change Password page is created by the PL/SQL routine `WWSSO_APP_USER_MGR.CHANGE_PASSWORD`. This routine renders the screen and commits the form through an API to the database.

The process is as follows:

1. The user selects the Change Password link on the SSO Server Administration page. Selecting this link invokes the `WWSSO_APP_USER_MGR.CHANGE_PASSWORD` routine.
2. `CHANGE_PASSWORD` displays the Change Password page, which displays the username and has fields for the old password, the new password, and the password confirmation. It also has OK and Cancel buttons.
3. `CHANGE_PASSWORD` processes the new password.
4. `CHANGE_PASSWORD` saves the new password and redirects to the appropriate application page.

To accommodate a customized Change Password page, the logic for the Change Password page has been modified as follows:

1. The WWSSO_APP_USER_MGR.CHANGE_PASSWORD routine is invoked to display the Change Password page.
2. If a separate URL is to display the Change Password page, CHANGE_PASSWORD redirects to that URL.

   If no separate URL is specified, CHANGE_PASSWORD calls WWSSO_APP_USER_MGR.DRAW_CHANGE_PASSWORD_PAGE to display the standard Change Password page.
3. The Change Password page submits a form to WWSSO_APP_USER_MGR.SAVE_NEW_PASSWORD to process and save the new password.
4. If there are no errors, SAVE_NEW_PASSWORD saves the new password and redirects to the Single Sign-On home page.

The LOGIN_URL column of the WWSSO_LS_CONFIGURATION_INFO$ table stores the URL for the customized Change Password page. The CHANGE_PASSWORD routine queries the value of the LOGIN_URL column to determine how to proceed. This column contains URLs for the Login and Change Password pages, separated by a space.

The Change Password page is also displayed immediately following a user login if the user's password has expired or will be expiring soon. If the password has expired, the Change Password page appears with the appropriate message and the following process occurs:

1. WWSSO_APP_ADMIN.LS_LOGIN calls WWSSO_APP_USER_MGR.CHANGE_PASSWORD to display the Change Password page.
2. If a separate URL is to display the Change Password page, CHANGE_PASSWORD redirects to that URL.

   If a separate URL is not specified, CHANGE_PASSWORD calls WWSSO_APP_USER_MGR.DRAW_CHANGE_PASSWORD_PAGE and displays the standard Change Password page.
3. The Change Password page submits a form to WWSSO_APP_USER_MGR.SAVE_NEW_PASSWORD to process and save the new password.
4. If there are no errors and the user selects OK, SAVE_NEW_PASSWORD saves the new password and returns control to WWSSO_APP_ADMIN.LS_LOGIN to perform the necessary login steps.

5. If there are errors or if the user selects Cancel, `SAVE_NEW_PASSWORD` calls `CHANGE_PASSWORD` and redisplays the Change Password page. This process repeats until the user changes the password successfully.

> **Note:** Selecting Cancel should not allow a user to continue if the password has expired. However, if the password is set to expire after a certain number of days and the user selects Cancel, the login resumes and the user's password remains unchanged.

If the user's password is about to expire, the Change Password page appears with the appropriate message and the following process occurs:

1. `WWSSO_APP_ADMIN.LS_LOGIN` calls `WWSSO_APP_USER_MGR.CHANGE_PASSWORD` to display the Change Password page.
2. If a separate URL is to display the Change Password page, `CHANGE_PASSWORD` redirects to the separate URL.

   If no separate URL is specified, `CHANGE_PASSWORD` calls `WWSSO_APP_USER_MGR.DRAW_CHANGE_PASSWORD_PAGE` to display the standard Change Password page.
3. The Change Password page submits a form to `WWSSO_APP_USER_MGR.SAVE_NEW_PASSWORD` to process and save the new password.
4. If there are no errors and the user selects OK, `SAVE_NEW_PASSWORD` saves the new password and returns control to `WWSSO_APP_ADMIN.LS_LOGIN` to perform the necessary login steps.
5. If there are errors, `SAVE_NEW_PASSWORD` calls `CHANGE_PASSWORD` and redisplays the Change Password page.
6. If the user selects Cancel, `SAVE_NEW_PASSWORD` does not save the new password but returns control to `WWSSO_APP_ADMIN.LS_LOGIN` to perform the login steps using the current password.

Enabling the Single Sign-Off Page

The Single Sign-Off page logs off users from all active partner applications simultaneously. When users select an application logout link, they are redirected to the Single Sign-Off URL of the Single Sign-On server. First this URL deletes the server cookies; then it calls application logout URLs in parallel, using the HTML IMG tag. The application logout URLs delete application session cookies and then stream a small image indicating a successful application logout.

The process is as follows:

1. The User selects the application logout link.
2. The user is redirected to the Single Sign-Off URL, `WWSSSO_APP_ADMIN.LS_LOGOUT` with a return URL, p_done_url parameter.
3. The Single Sign-Off URL deletes Single Sign-On server cookies and then checks the login_url column of the wwsso_ls_configuration_info$ table to determine whether a customized page should be used.
4. If the login_url column of the wwsso_ls_configuration_info$ table indicates that the standard page should be used, the Single Sign-On server calls the wwsso_login.draw_logout_page procedure. This procedure renders the standard Single Sign-Off page.
5. If the login_url column indicates that a customized page should be used, the Single Sign-Off URL redirects the user to the customized page, passing the parameters required for Single Sign-Off.
6. The Single Sign-Off page calls application logout URLs in parallel and the user is logged out from all applications.

Parameters for Login, Change Password, and Single Sign-Off Pages

The URLs for Login, Change Password, and Single Sign-Off pages must accept the parameters described in the tables that follow if these pages are to function properly.

This section covers the following topics:

- Login Page Parameters
- Change Password Page Parameters
- Single Sign-Off Page Parameters

Login Page Parameters

The URL for the Login page must accept the parameters listed in Table 8–1.

Table 8–1 Login Page Parameters

Parameter	Description
`site2pstoretoken`	Contains the authentication request token for login processing.
`ssousername`	Contains the username.
`p_error_code`	Contains the error code, in the form of a VARCHAR2, if an error occurred during authentication.
`p_cancel_url`	Contains the URL to redirect to if the user selects Cancel, if such a button exists on the login page.
`p_submit_url`	Contains the URL that the login page must submit the form to, `WWSSO_APP_ADMIN.LS_LOGIN`.
`subscribername`	Reserved for future use. **Note**: This field is required on the login page.

The customized login page must conform to the `wwsso_app_admin.ls_login` procedure in the same manner as the standard login page; passing the parameters listed in Table 8–2 to the `p_submit_url` routine:

Table 8–2 Customized Login Page Parameters

Parameter	Description
`site2pstoretoken`	Contains the redirect URL information for login processing.
`ssousername`	Contains the username.
`p_error_code`	Contains the error code, in the form of a VARCHAR2, if an error occurred during authentication.
`password`	Contains the password entered by the user.
`subscribername`	Reserved for future use. **Note**: This field is required on the login page.

The customized login page must have at least two fields: a text field with the parameter name `ssousername` and a password field with the parameter name `password`. The values are submitted to the `p_submit_url` routine. The login page must also submit the `site2pstoretoken` value as a hidden parameter.

In addition to submitting these parameters, the login page is responsible for displaying appropriate error messages, as specified by the `p_error_code` parameter, redirecting to `p_cancel` if the user selects Cancel and populating the `ssousername` text field with the given parameter value in the case of a login error.

If the customized login page requires additional fields, you can include them. Ensure that additional fields are appropriately wrapped to conform to the above convention for integration with the Single Sign-On server.

Change Password Page Parameters

The URL for the Change Password page must accept the parameters listed in Table 8–3.

Table 8–3 Change Password Page Parameters

Parameter	Description
`p_username`	Contains the username to be displayed somewhere on the page.
`p_password`	Contains the user's original password (if password is or is about to expire).
`p_error_code`	Contains the error code, in the form of a VARCHAR2, if an error occurred in the prior attempt to change password.
`p_submit_url`	Contains the URL that the Change Password form must submit to.
`p_done_url`	Contains the URL of the appropriate Oracle9*i*AS Portal page to return to after the password is saved.
`p_pwd_is_exp`	Contains the flag value indicating whether the password has expired or is about to expire.
`site2pstoretoken`	Contains the `site2pstoretoken` that is required by the `LS_LOGIN` routine if the password has expired or is about to expire.

The customized Change Password page must pass the parameters listed in Table 8–4 to the `p_submit_url` routine.

Table 8–4 Customized Change Password Page Parameters

Parameter	Description
`p_username`	Contains the username to be displayed somewhere on the page. Should be posted as a hidden field by the custom Change Password page.
`p_old_password`	Contains the user's old password.
`p_new_password`	Contains the user's new password.
`p_new_password_confirm`	Contains the confirmation of the user's new password.
`p_done_url`	Contains the URL of the appropriate Oracle9*i*AS Portal page to return to after the password is saved.
`p_pwd_is_exp`	Contains the flag value indicating whether the password has expired or is about to expire.
`site2pstoretoken`	Contains the redirect URL information for login processing.
`p_password`	Contains the password entered by the user.
`p_action`	Commits changes. The values must be either `OK` (commit) or `CANCEL` (ignore); otherwise, p_action defaults to null and does not commit changes.

The Change Password page must have at least three password fields with the following parameter names:

- `p_old_password`
- `p_new_password`
- `p_new_password_confirm`.

The Change Password page should submit these fields to the `p_submit_url` parameter.

The Change Password page should also submit the `p_done_url` parameter, as a hidden parameter, to the `p_submit_url` parameter, and should appropriately display any error messages according to the value of `p_error_code`.

For external applications, the Change Password page must submit the following parameters, as hidden parameters, to the standard HTML login form.

- `p_pwd_is_exp`
- `site2pstoretoken`
- `p_password`

Single Sign-Off Page Parameters

The URL for the Single Sign-Off page must accept the parameters listed in Table 8–5.

Table 8–5 Single Sign-Off Page Parameters

Parameter	Description
`p_app_name`	Contains the application name to be displayed on the page.
`p_app_logout_url`	Contains the application logout URL.
`p_done_url`	Contains the return URL. This URL returns users to the application from which they initiated logout.

Error Codes for Login and Change Password Pages

URLs for Login and Change Password pages must accept the parameters described in the tables that follow if these pages are to function properly.

This section covers the following topics:

- Login Page Error Codes
- Change Password Page Error Codes

Login Page Error Codes

The customized login page must process the error codes listed in Table 8–6.

Table 8–6 Customized Login Page Error Codes

Value of p_error_code	Corresponding error
`acct_ip_lock_err`	The user has committed too many login failures from this IP address and has been locked out.
`acct_lock_err`	The user has committed too many login failures from any IP address and has been globally locked out.
`null_uname_pwd_err`	The user did not type in a username.
`no_papp_err`	The partner application configuration is missing or expired.
`ssl_not_used_err`	SSL is not being used.
`ls_config_not_found_err`	The Single Sign-On server configuration is missing.
`cookies_disabled_err`	The user's browser is not accepting cookies.
`auth_fail_exception`	Authentication has failed.
`account_deactivated_err`	The user's account has been terminated.
`value_error_exception`	An invalid value was specified in `site2pstoretoken`.
`null_password_err`	The user did not type in a password.
`ext_auth_unknown_err`	There was an unknown error in accessing the external authentication mechanism.
`ext_auth_setup_err`	There was an error in the setup of the external authentication mechanism.
`sso_cookie_expired_err`	The login cookie has expired. The user needs to log in again.
`unexpected_exception`	An unexpected error occurred during authentication.

Change Password Page Error Codes

The customized Change Password page must process the error codes listed in Table 8–7.

Table 8–7 Change Password Page Error Codes

Value of p_error_code	Corresponding Error
`null_old_pwd_err`	The user did not type in an old password.
`null_new_pwd_err`	The user did not type in a new password.
`confirm_pwd_fail_txt`	The user typed in a new password confirmation that did not match the new password.
`auth_fail_err`	The user typed in an invalid old password.
`pwd_rule_err`	The user typed in a new password that does not meet the password requirements of the Single Sign-On server.
`invalid_auth_mode_err`	The change password operation is not supported by the current authentication mechanism.
`ext_not_supported_err`	The external repository is not supported.
`ext_change_pwd_err`	The change password operation was unsuccessful on the external repository.
`pwd_expired_err`	The password has expired.
`pwd_needs_change_err`	The password is about to expire. The user is allowed to log in.

Sample Customized Pages

This section gives examples of how standard Single Sign-On Login, Change Password, and Single Sign-Off pages might be modified as JSPs. As such, the examples provide a basis for customizing these pages to suit enterprise needs.

Three examples are provided:

- Sample Login Page
- Sample Change Password Page
- Sample Single Sign-Off Page

Sample Login Page

```
<html>
<body bgcolor="white">

<%@ page buffer="5" autoFlush="true" %>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 2000 17:04:19 GMT");

try
{
   String  str_token  = request.getParameterValues("site2pstoretoken")[0];
   String  str_user   = request.getParameterValues("ssousername")[0];
   String  str_err    = request.getParameterValues("p_error_code")[0];
   String  str_cancel = request.getParameterValues("p_cancel_url")[0];
   String  str_submit = request.getParameterValues("p_submit_url")[0];

   out.println("<center><h1>Single Sign-On Login</h1><p>");
   out.println("<form method='post' action='"+str_submit+"'>");
   out.println("<INPUT TYPE='hidden' NAME='site2pstoretoken'");
   out.println(" value='"+str_token+"'>");

   out.println("<table border=0>");
   if((str_err != null) && (str_err.length() > 1))
   {
       out.println("<tr>");
       out.println("<td>");
       out.println("<font color='red'>ERROR:</font>");
       out.println("</td>");
       out.println("<td>");
       out.println(str_err);
       out.println("</td>");
       out.println("</tr>");
   }

   out.println("<tr>");
   out.println("<td>");
   out.println("User Name:");
   out.println("</td>");
   out.println("<td>");
   out.println("<INPUT TYPE='text' NAME='ssousername'>");
   out.println("</td>");
   out.println("<tr>");
   out.println("<tr>");
   out.println("<td>");
   out.println("Password");
   out.println("</td>");
   out.println("<td>");
   out.println("<INPUT TYPE='password' NAME='password'>");
   out.println("</td>");
   out.println("<tr>");

   out.println("<tr>");
   out.println("<td>");
   out.println("<INPUT TYPE='submit' VALUE='Login'>");
   out.println("<INPUT TYPE='button' NAME='p_request' VALUE='Cancel' ");
   out.println("  onClick='javascript:document.location.href = ’"+str_
cancel+"’;'>");
   out.println("<td>");
   out.println("</tr>");
   out.println("</table>");

   out.println("</form>");
}
catch(Exception e)
{
    out.println("<h2><center><font color='red'>ERROR:</font>");
    out.println("This page can not be accessed directly!</center></h2>");
}

%>
</body>
</html>
```

Sample Change Password Page

```
<html>
<body bgcolor="white">

<%@ page buffer="5" autoFlush="true" %>

<%
    response.setHeader("Pragma", "no-cache");
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Expires", "Thu, 29 Oct 2000 17:04:19 GMT");
%>

<script language="JavaScript">
<!--
function button1submit()
{
   document.ChangePassword.p_action.value = "OK";
   document.ChangePassword.submit();
}

function button2submit()
{
   document.ChangePassword.p_action.value = "CANCEL";
   document.ChangePassword.submit();
}
//-->"
</script>

<%
try
{
   String str_token   = request.getParameterValues("site2pstoretoken")[0];
   String str_user    = request.getParameterValues("p_username")[0];
   String str_pwd     = request.getParameterValues("p_password")[0];
   String str_err     = request.getParameterValues("p_error_code")[0];
   String str_done    = request.getParameterValues("p_done_url")[0];
   String str_submit  = request.getParameterValues("p_submit_url")[0];
   String str_pwd_exp = request.getParameterValues("p_pwd_is_exp")[0];
   out.println("<center><h1>Single Sign-On Change Password</h1><p>");

   out.println("<form method='post'  name='ChangePassword' action='"+str_
submit+"'>");

   out.println("<INPUT TYPE='hidden' NAME='p_username' value='"+ str_user
+"'>");
   out.println("<INPUT TYPE='hidden' NAME='site2pstoretoken' value='"+ str_token
+"'>");
   out.println("<INPUT TYPE='hidden' NAME='p_done_url' value='"+ str_done
+"'>");
   out.println("<INPUT TYPE='hidden' NAME='p_pwd_is_exp' value='"+ str_pwd_exp
+"'>");
   out.println("<INPUT TYPE='hidden' NAME='p_password' value='"+ str_pwd +"'>");
   out.println("<INPUT TYPE='hidden' NAME='p_request' value=''>");
   out.println("<INPUT TYPE='hidden' NAME='p_action' value=''>");

   out.println("<table border=0>");

   if((str_err != null) && (str_err.length() > 1))
   {
       out.println("<tr>");
       out.println("<td>");
       out.println("<font color='red'>ERROR:</font>");
       out.println("</td>");
       out.println("<td>");
       out.println(str_err);
       out.println("</td>");
       out.println("</tr>");
   }
   out.println("<tr>");
   out.println("<td>");
   out.println("User Name:");
   out.println("</td>");
   out.println("<td>");
   out.println(str_user);
   out.println("</td>");
   out.println("<tr>");


   out.println("<tr>");
   out.println("<td>");
   out.println("Old password:");
   out.println("</td>");
   out.println("<td>");
   out.println("<INPUT TYPE='password' NAME='p_old_password'>");
   out.println("</td>");
   out.println("<tr>");

   out.println("<tr>");
   out.println("<td>");
   out.println("New password:");
   out.println("</td>");
   out.println("<td>");
   out.println("<INPUT TYPE='password' NAME='p_new_password'>");
   out.println("</td>");
   out.println("<tr>");

   out.println("<tr>");
   out.println("<td>");
   out.println("Confirm new password:");
   out.println("</td>");
   out.println("<td>");
   out.println("<INPUT TYPE='password' NAME='p_new_password_confirm'>");
   out.println("</td>");
   out.println("<tr>");

   out.println("<tr>");
   out.println("<td>");
   out.println("<INPUT TYPE='button' NAME='p_request' VALUE='OK' "
       +" onClick='javascript:button1submit();'>");
   out.println("<INPUT TYPE='button' NAME='p_request' VALUE='Cancel' "
       +"onClick='javascript:button2submit();'>");
   out.println("</td>");
   out.println("<tr>");

   out.println("</table");

   out.println("</form>");
}
catch(Exception e)
{
    out.println("<h2><center><font color='red'>ERROR:</font>");
    out.println("This page can not be accessed directly!</center></h2>");
}

%>
</body>
</html>
```

Sample Single Sign-Off Page

```
<html>
<body bgcolor="white">

<%@ page buffer="5" autoFlush="true" %>
<%
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setHeader("Expires", "Thu, 29 Oct 1970 17:04:19 GMT");

String done_url = null;
int i = 0;
try
{
   done_url =  request.getParameterValues("p_done_url")[0];

   out.println("<center><h1>Single Sign-Off</h1><p>");
   out.println("<table border=0>");
   out.println("<tr>");
   out.println("<th>Appliction Name</th>");
   out.println("<th>Logout Status</th>");
   out.println("</tr>");

   for(;;)
   {
      i++;

      String app_name = request.getParameterValues("p_app_name"+i)[0];
      String url_name = request.getParameterValues("p_app_logout_url"+i)[0];
      out.println("<tr>");
      out.println("<td>"+app_name+"</td>");
      out.println("<td><img src='" +url_name +"'></td>");
      out.println("</tr>");
   }
}
catch(Exception e)
{
    if(i>1)
    {
      out.println("</table>");
      out.println("<br>");
      out.println("<form><INPUT TYPE='button' "
          +" NAME='p_request' VALUE='Return' ");
      out.println(" onClick='javascript:document.location.href = ’"
            + done_url +"';'><form></center>");
    }
    else
    {
       out.println("<h2><center><font color='red'>ERROR:</font>");
       out.println("This page can not be accessed directly!</center></h2>");
    }
}

%>
</body>
</html>
```

Index