Oracle9*i*AS Single Sign-On

Integration with Third-Party Single Sign-On Products

Version 3.0.9

August 2001

Part No. A95114-01

Oracle9*i*AS Single Sign-On provides single sign-on to all features in the Oracle9*i*AS product complement. Because these applications delegate authentication to the Oracle Single Sign-On server, users need authenticate only once to gain access to Oracle9*i*AS applications. Customers who have third-party single sign-on products in place can also gain access to the 9*i*AS suite by using APIs that enable the Oracle Single Sign-On server to act as an authentication gateway between third-party single sign-on systems and Oracle applications.

This presentation explains how the Oracle single sign-on integration solution works; then it presents the integration APIs. Finally, it presents sample code that integrates Oracle9*i*AS Single Sign-On with SiteMinder®, a single sign-on product from Netegrity, Inc.

The presentation contains the following topics:

How Oracle9*i*AS Single Sign-On Works

The key component in most single sign-on systems is the authentication token. The first time a successful login to a Web application occurs, the Oracle Single Sign-On server issues the user a token that other Web sites use subsequently to establish his or her identity. The user provides a user name and password only once. Key to this arrangement are the notions of delegation and trust. The applications involved trust the single sign-on server to delegate the authentication function to it. For this reason, they are called partner applications.

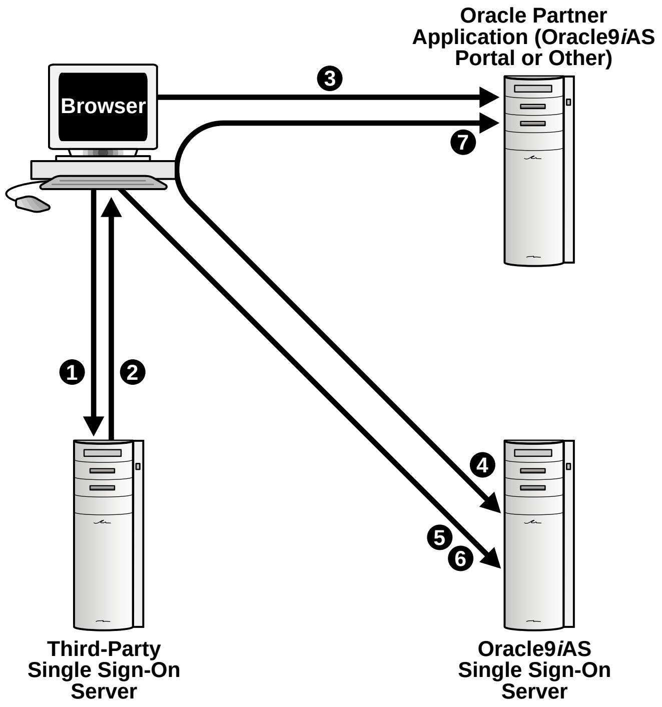
Integration with Third-Party Single Sign-On Products

In cases where Oracle9*i*AS Single Sign-On integrates with other single sign-on products, the principle is the same. The only difference is that 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 Oracle9*i*AS Single Sign-On to issue users an authentication token under this arrangement, the third party single sign-on server must pass it 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 1 on page 3 illustrates the process.

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


Oracle Partner Application (Oracle9*i*AS Portal or Other)
Browser
3
7
1
2
4
5
6
Third-Party Single Sign-On Server
Oracle9*i*AS Single Sign-On Server

1. The user logs in to the third-party single sign-on server.
2. If login is successful, the third-party single sign-on 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. 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 identity.

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 "Customizing the Single Sign-On Login Page," in Chapter 5 of *Oracle9iAS Single Sign-On Administrator's Guide*
- 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.

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 User Name and Password
- Authentication Using a Token
- Account Policy Enforcement
- Change Password
- Reset Password
- Set External Cookies

Authentication Using a User Name and Password

The following function must be implemented if the Oracle Single Sign-On server is to authenticate using an external user repository. The server uses information entered in the Login form to call this function.

```
FUNCTION authenticate_user
(
 p_user IN VARCHAR2
,p_password IN VARCHAR2
)
RETURN PLS_INTEGER;

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

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
*/
```

Account Policy Enforcement

The Oracle Single Sign-On server provides account and password policies. Where an external repository provides similar features, these policies can be switched off and on by modifying the following function to return 0 (false) or 1 (true).

```
FUNCTION enforce_account_policies
RETURN BOOLEAN;

/*Returns 0 or 1 for now
Policies that are enforced or not are:
1. User termination
2. Lockout
*/
```

Change Password

The Oracle Single Sign-On server prompts the user for the old and the new password when the user requests a change password. The change password procedure below uses these parameters to implement a password change in the external repository.

```
PROCEDURE change_passwd
(
 p_user IN VARCHAR2
,p_oldpwd IN VARCHAR2
,p_newpwd IN VARCHAR2
);

/*The procedure throws the following exceptions:
EXT_NOT_SUPPORTED_EXCEPTION,EXT_CHANGE_PASSWORD_FAILED,
EXT_CHANGE_PASSWD_EXCEPTION
*/
```

Reset Password

In the following procedure, the Oracle Single Sign-On server resets the user's password using a randomly generated value. The call specification `wwsso_ls_private.ls_automated_reset_password` obtains this value by calling the procedure.

```
PROCEDURE reset_passwd
(
 p_user IN VARCHAR2
,p_passwd IN VARCHAR2
);

/*The procedure throws the following exceptions:
EXT_NOT_SUPPORTED_EXCEPTION,EXT_RESET_PASSWORD_EXCEPTION,
EXT_AUTH_SETUP_EXCEPTION
*/
```

Get Authentication Name

The following function returns the name of the external authentication agent and then displays this name on the Oracle Single Sign-On server configuration screen.

```
FUNCTION get_authentication_name
RETURN VARCHAR2;

/*The function throws the following exceptions:
EXT_AUTH_SETUP_EXCEPTION,EXT_NOT_SUPPORTED_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.

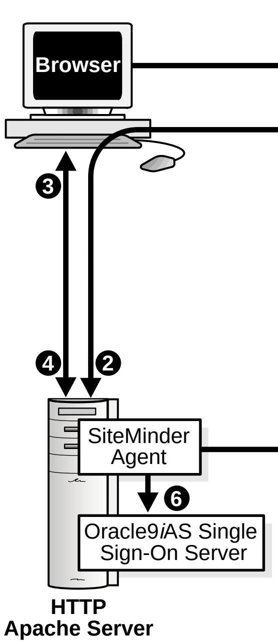
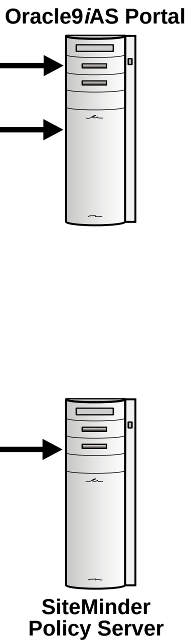
This section covers the following topics:

- 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 2 on page 9 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 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.

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 v2 (Release 9.0.1), 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.

Clicking 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 Netegrity 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 a
Rem      default implementation and changes might be required based on a
Rem      customer's specific deployment scenario.
Rem    NOTES
Rem
Rem


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 look up
 * the UID for a given DN in the directory. Note that the function must
 * be modified only if users are authenticated using PKI.
 */


FUNCTION map_dn_to_uid(p_user_dn IN VARCHAR2)
  return VARCHAR2
IS
BEGIN

  -- NULL implementation by default

  raise EXT_AUTH_FAILURE_EXCEPTION;

  --In actual implementation, map DN to UID and return UID.
  --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 authenticate_user
  (
    p_user IN VARCHAR2,
    p_password IN VARCHAR2
  )
   RETURN PLS_integer
IS
BEGIN

  raise EXT_AUTH_FAILURE_EXCEPTION;

END authenticate_user;


PROCEDURE get_configuration
  (
    p_config OUT ext_config
  )
AS
BEGIN

  null;
  -- p_config := NULL;

END get_configuration;


PROCEDURE change_passwd
  (
    p_user IN VARCHAR2,
    p_oldpwd IN VARCHAR2,
    p_newpwd IN VARCHAR2
  )
AS
BEGIN

 raise EXT_NOT_SUPPORTED_EXCEPTION;

EXCEPTION
    WHEN OTHERS THEN
      RAISE EXT_CHANGE_PASSWORD_EXCEPTION;
END change_passwd;
```

```
FUNCTION enforce_account_policies.
    RETURN BOOLEAN
IS
BEGIN
    return FALSE;

END enforce_account_policies;


PROCEDURE reset_passwd
  (
    p_user IN VARCHAR2
  , p_passwd IN VARCHAR2
  )
IS
BEGIN

   raise EXT_NOT_SUPPORTED_EXCEPTION;

END reset_passwd;


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, the SiteMinder Apache 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
   ```

7. 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: "Customizing the Change Password Page," in Chapter 5 of *Oracle9iAS Single Sign-On Administrator's Guide*