Oracle® Access Manager

Integration Guide

10*g* (10.1.4.3)

E12492-01

May 2009

Explains how to set up Oracle Access Manager to run with other Oracle products, for example, OracleAS Web Cache, and also third-party products.

ORACLE®

Oracle Access Manager Integration Guide 10*g* (10.1.4.3)

E12492-01

Primary Author: Gail Flanegin

Contributing Author: Nina Wishbow

Contents

5 Integrating with Oracle Application Server 10*g*: OC4J

Index

Preface

This Integration Guide provides information about integrating Oracle Access Manager with other Oracle and third-party applications servers and portals.

> **Note:** Oracle Access Manager was previously known as Oblix *NetPoint*.

This Preface covers the following topics:

Audience

This guide is intended for administrators who are responsible for integrating their product with Oracle Access Manager.

This guide assumes that you are familiar with your LDAP directory and Web servers, Oracle Access Manager, and the product that you are integrating.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. 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. Accessibility standards will continue to evolve over time, and Oracle 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 more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers 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, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

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

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

Related Documents

For more information, see the following documents in the Oracle Access Manager Release 10*g* (10.1.4.3) documentation set:

- *Oracle Access Manager Introduction*—Provides an introduction to Oracle Access Manager, a road map to the manuals, and a glossary of terms.
- *Oracle Access Manager Release Notes*—Read these for the latest Oracle Access Manager information.
- *Oracle Access Manager Patch Set Notes Release 10.1.4 Patch Set 2 (10.1.4.3.0) For All Supported Operating Systems*—Read this document if you want to apply the 10*g* (10.1.4.3) patch set to an existing 10*g* (10.1.4.2.0) deployment. It includes a list of enhancements, bug fixes, and known issues related to the patch set.
- *Oracle Access Manager Installation Guide*—Explains how to prepare for, install, and set up each Oracle Access Manager component.
- *Oracle Access Manager Upgrade Guide*—Explains how to upgrade earlier releases to the latest major Oracle Access Manager release using either the in-place component upgrade method or the zero downtime method.
- *Oracle Access Manager Identity and Common Administration Guide*—Explains how to configure Identity System applications to display information about users, groups, and organizations; how to assign permissions to users to view and modify the data that is displayed in the Identity System applications; and how to configure workflows that link together Identity application functions, for example, adding basic information about a user, providing additional information about the user, and approving the new user entry, into a chain of automatically performed steps. This book also describes administration functions that are common to the Identity and Access Systems, for example, directory profile configuration, password policy configuration, logging, and auditing.
- *Oracle Access Manager Access Administration Guide*—Describes how to protect resources by defining policy domains, authentication schemes, and authorization schemes; how to allow users to access multiple resources with a single login by configuring single- and multi-domain single sign-on; and how to design custom login forms. This book also describes how to set up and administer the Access System.
- *Oracle Access Manager Deployment Guide*—Provides information for people who plan and manage the environment in which Oracle Access Manager runs. This guide covers capacity planning, system tuning, failover, load balancing, caching, and migration planning.

- *Oracle Access Manager Customization Guide*—Explains how to change the appearance of Oracle Access Manager applications and how to control operation by making changes to operating systems, Web servers, directory servers, directory content, or by connecting CGI files or JavaScripts to Oracle Access Manager screens. This guide also describes the Access Manager API and the authorization and authentication plug-in APIs.
- *Oracle Access Manager Developer Guide*—Explains how to access Identity System functionality programmatically using IdentityXML and WSDL, how to create custom WebGates (known as AccessGates), and how to develop plug-ins. This guide also provides information to be aware of when creating CGI files or JavaScripts for Oracle Access Manager.
- *Oracle Access Manager Integration Guide*—Explains how to set up Oracle Access Manager to run with other Oracle and third-party products.
- *Oracle Access Manager Schema Description*—Provides details about the schema.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in Oracle Access Manager?

This section describes changes for Oracle Access Manager 10*g* (10.1.4.3). The following sections are included:

- Product and Component Name Changes
- Supported Integrations

> **Note:** For a comprehensive list of all new features and functions in Oracle Access Manager 10.1.4, and a description of where each is documented, see the chapter on what's new in the *Oracle Access Manager Introduction*.

Product and Component Name Changes

The original product name, Oblix NetPoint, has changed to Oracle Access Manager. Most component names remain the same. However, there are several important changes that you should know about, as shown in the following table:

Item	Was	Is
Product Name	Oblix NetPoint Oracle COREid	Oracle Access Manager
Product Name	Oblix SHAREid NetPoint SAML Services	Oracle Identity Federation
Product Name	OctetString Virtual Directory Engine (VDE)	Oracle Virtual Directory
Product Name	BEA WebLogic Application Server BEA WebLogic Portal Server	Oracle WebLogic Server Oracle WebLogic Portal
Product Release	Oracle COREid 7.0.4	Also available as part of Oracle Application Server 10g Release 2 (10.1.2).
Directory Name	COREid Data Anywhere	Data Anywhere
Component Name	COREid Server	Identity Server
Component Name	Access Manager	Policy Manager
Console Name	COREid System Console	Identity System Console
Identity System Transport Security Protocol	NetPoint Identity Protocol	Oracle Identity Protocol

Item	Was	Is
Access System Transport Protocol	NetPoint Access Protocol	Oracle Access Protocol
Administrator	NetPoint Administrator COREid Administrator	Master Administrator
Directory Tree	Oblix tree	Configuration tree
Data	Oblix data	Configuration data
Software Developer Kit	Access Server SDK ASDK	Access Manager SDK
API	Access Server API Access API	Access Manager API
API	Access Management API Access Manager API	Policy Manager API
Default Policy Domains	NetPoint Identity Domain COREid Identity Domain	Identity Domain
Default Policy Domains	NetPoint Access Manager COREid Access Manager	Access Domain
Default Authentication Schemes	NetPoint None Authentication COREid None Authentication	Anonymous Authentication
Default Authentication Schemes	NetPoint Basic Over LDAP COREid Basic Over LDAP	Oracle Access and Identity Basic Over LDAP
Default Authentication Schemes	NetPoint Basic Over LDAP for AD Forest COREid Basic Over LDAP for AD Forest	Oracle Access and Identity for AD Forest
Access System Service	AM Service State Policy Manager API Support Mode	Access Management Service **Note**: Policy Manager API Support Mode and Access Management Service are used interchangeably.

All legacy references in the product or documentation should be understood to connote the new names.

Supported Integrations

The introduction describes supported integrations for this release. An overview of supported integrations is provided for quick reference. All other chapters in this guide describe implementation details for a specific integration.

See Also: Chapter 1, "Introduction".

The following integrations are supported with Oracle Access Manager 10*g* (10.1.4.3) initially. Oracle performs certification on an on-going basis and this book is updated accordingly.

- This guide introduces integrating with Oracle HTTP Server.

See Also: Chapter 2, "Integrating Oracle HTTP Server".

- This guide introduces integrating with Oracle Virtual Directory.

  The Oracle Virtual Directory combines user data from multiple data sources to create an aggregated virtual directory.

  See Also: Chapter 3, "Integrating the Oracle Virtual Directory".

- This guide provides information on integrating with OracleAS Web Cache to provide POST data restoration and cookieless support in Oracle Access Manager.

  See Also: Chapter 4, "Integrating OracleAS Web Cache".

- This guide provides information on integrating with OracleAS Web Cache to provide POST data restoration and cookieless support in Oracle Access Manager.

  See Also: Chapter 5, "Integrating with Oracle Application Server 10g: OC4J"

1 Introduction

This chapter provides an overview of the Oracle Access Manager 10.1.4 integrations described in this guide. For an introduction to Oracle Access Manager, see the *Oracle Access Manager Introduction*.

> **Note:** Oracle Access Manager was previously known as Oblix *NetPoint*. However, you may see the name NetPoint in manuals and within the product itself when references are made to specific functions, paths, file names, and so on.

About Oracle Access Manager Integrations

Integrating Oracle Access Manager 10.1.4 with other applications and portals requires some knowledge of both products. This guide provides the details you need to successfully set up Oracle Access Manager for specific applications and portals you can integrate with Oracle Access Manager.

Integrations With Other Oracle Products

The following integrations with other Oracle products are described in this guide:

- **Oracle HTTP Server:** Oracle HTTP Server provides key infrastructure for serving the Internet's HTTP protocol. Oracle HTTP Server is used to return responses for both process-to-process and human-generated requests from browsers. Oracle HTTP Server serves both static and dynamic content and integrates with both Oracle and non-Oracle products. See "Integrating Oracle HTTP Server" on page 2-1 for details.
- **Oracle Virtual Directory (OVD):** This product combines user data from multiple data sources to create an aggregated virtual directory. The virtual directory looks and behaves like any other LDAP directory, and the user does not know that the data has come from heterogeneous sources. See "Integrating the Oracle Virtual Directory" on page 3-1 for details.
- **OracleAS Web Cache:** Oracle Access Manager 10*g* (10.1.4.3) provides support for integration with OracleAS Web Cache. OracleAS Web Cache is a reverse proxy cache and compression engine that is deployed between the browser and the Oracle Access Manager WebGate Web server. This configuration provides the following Oracle Access Manager functionality: POST data restoration and cookieless support. See "Integrating OracleAS Web Cache" on page 4-1.
- **OracleAS Single Sign-On 10g, OC4J:** Oracle Application Server Single Sign-On (also referred to as OracleAS Single Sign-On) enables you to use a single user name, password, and optionally a realm ID to log in to all features of the Oracle

Application Server and also to other Web applications. You can enable single sign-on between resources protected by Oracle Access Manager and OracleAS Single Sign-On. See "Integrating with Oracle Application Server 10g: OC4J" on page 5-1 for details.

See Also: Patch number 5957301 on My Oracle Support (formerly MetaLink) for details about other integrations, available for Oracle Access Manager 10*g* (10.1.4.2.0): `http://metalink.oracle.com`

2

Integrating Oracle HTTP Server

Oracle HTTP Server provides key infrastructure for serving the Internet's HTTP protocol. Oracle HTTP Server is used to return responses for both process-to-process and human-generated requests from browsers. Oracle HTTP Server serves both static and dynamic content and integrates with both Oracle and non-Oracle products. This chapter includes the following sections:

About Oracle Access Manager Packages for Oracle HTTP Server

Oracle Access Manager provides Web components (WebPass, Policy Manager, and WebGate) for various releases of Oracle HTTP Server. Oracle HTTP Server is based on Apache open source.

Oracle Access Manager Web component package names indicate which Oracle HTTP Server. release is supported. For example:

- Oracle HTTP Server 11g based on Apache 2.2:

  Oracle_Access_Manager10_1_4_3_0_ *platform*_OHS11g_WebGate

- Oracle HTTP Server releases based on Apache v2.0 (10*g* R2 (10.1.2) and 10g (10.1.3.1.0)):

  Oracle_Access_Manager10_1_4_3_0_*platform*_OHS2_WebPass

- Oracle HTTP Server based on Apache v1.3 (10*g* R2 (10.1.2) and 10g (10.1.3.1.0)):

  Oracle_Access_Manager10_1_4_3_0_ *platform*_OHS_WebGate

About Using Oracle HTTP Server with Oracle Access Manager

Oracle Access Manager Web components can be installed on a stand alone Oracle HTTP Server on supported platforms. Oracle HTTP Server 11g is also an enabling technology for deploying Oracle Access Manager single sign-on with Oracle Fusion Middleware products:

- To use Oracle HTTP Server as a host for Oracle Access Manager Web components, see the following *Oracle Access Manager Installation Guide* topics:
  - Meeting Web Server Requirements

  - Configuring Apache v1.3-based Web Servers for Oracle Access Manager
  - Configuring Oracle HTTP Server, Apache v2, and IHS Web Servers for Oracle Access Manager

- To enable enterprise-level single sign-on with Oracle Fusion Middleware products, see the *Oracle Fusion Middleware Security Guide* chapter on "Configuring Single Sign-on for Oracle Fusion Middleware".

  See Also: General information on Orache HTTP Server 11g in

  - *Oracle Fusion Middleware Administrator's Guide for HTTP Server 11g.*
  - *Oracle Fusion Middleware Installation Guide for Web Tier* (E14260-01)
  - *Oracle HTTP Server Administrator's Guide (E10144-01)*

Supported Integrations

For the latest Oracle Access Manager certification information, see Oracle Technology Network.

To locate the latest certification details

1. Go to Oracle Technology Network:

   http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

2. Locate Oracle Identity and Access Management, and then click the link for the latest Oracle Access Manager certification spreadsheet. For example:

   System Requirements and Supported Platforms for Oracle Access Manager 10gR3 (xls)

   http://www.oracle.com/technology/products/id_mgmt/coreid_acc/pdf/oracle_access_manager_certification_10.1.4_r3_matrix.xls

3

Integrating the Oracle Virtual Directory

Oracle Virtual Directory combines user data from multiple data sources to create an aggregated virtual directory.

From the point of view of Oracle Access Manager applications, the virtual directory looks and behaves just like any other LDAP directory, and the Oracle Access Manager user usually does not receive any obvious indications that the data retrieved by Oracle Access Manager has come from heterogeneous sources.

From the perspective of the target data store owners, the impact of Oracle Virtual Directory is minimal. The data store owners do not relinquish ownership of their data, Oracle Virtual Directory does not reformat the native data structures, and no permanent copies of the original data are maintained by Oracle Virtual Directory.

From the perspective of the administrator, Oracle Virtual Directory enables you to use multiple data sources for Oracle Access Manager.

Much of this integration is accomplished during product installation and setup. For this reason, the details of integrating Oracle Access Manager and Oracle Virtual Directory are provided in the *Oracle Access Manager Installation Guide.*

4

Integrating OracleAS Web Cache

This chapter describes integrating OracleAS Web Cache and Oracle Access Manager to provide post-data restoration and cookieless support. in Oracle Access Manager.

This chapter covers the following topics:

About OracleAS Web Cache

OracleAS Web Cache is a reverse proxy cache and compression engine that optimizes application performance and more efficiently utilizes low-cost existing hardware resources. OracleAS Web Cache provides whole-page caching for static and dynamic content based on caching rules. OracleAS Web Cache supports invalidation as a mechanism to keep the cache consistent with the content on the application Web servers (in this case in Oracle Access Manager). For more information about OracleAS Web Cache, see the *Oracle Application Server Web Cache Administrator's Guide*.

When integrated with Oracle Access Manager, OracleAS Web Cache also enables the following functions:

- **POST Data Restoration**: WebGate uses Web Cache to provide POST data restoration after the POST request is interrupted for re-authentication due to timeout. performance. For more information, see "POST Data Restoration Scenario" on page 4-3.
- **Cookieless Session Support**: Oracle Access Manager uses Web Cache for cookieless session management. For more information, see "Cookieless Session Support Scenario" on page 4-5.

For general information about this integration, see "Integration Scenarios and Architecture" on page 4-2.

Integration Scenarios and Architecture

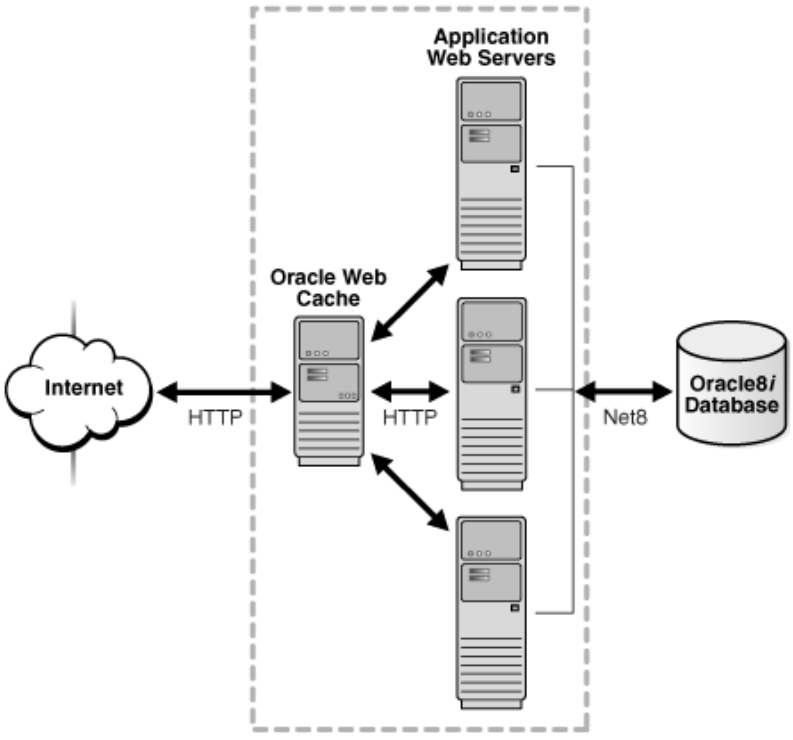
This section describes the integration architecture and flow of information after integrating Oracle Access Manager with OracleAS Web Cache for the following scenarios.

- Supported OracleAS Web Cache Topology for Integration with Oracle Access Manager
- POST Data Restoration Scenario
- Cookieless Session Support Scenario
- Supported Integrations

Supported OracleAS Web Cache Topology for Integration with Oracle Access Manager

OracleAS Web Cache topology support for integration with Oracle Access Manager provides for a single Web Cache configured with Web Servers hosting resources in a 1:N mapping. The supported topology is shown in Figure 4–1.

Figure 4–1 Supported OracleAS Web Cache Topology



The following conditions apply to integration between Oracle Access Manager and OracleAS Web Cache:

- OracleAS Web Cache cache clustering, failover, and load balancing are not supported with this integration scenario.
- OracleAS Web Cache Authorization and Access Enforcement requires the mod_access module through the Oracle HTTP Server available in the OracleAS Web Cache Suite.

- Oracle Application Server Single Sign-On Partner Applications (mod_osso) is not in part of the integration with Oracle Access Manager, which provides only Post Data Restoration and cookieless session support.
- OracleAS Web Cache can be managed using the Oracle Enterprise Manager 10g Application Server Control Console: Port numbers 18100 and 9400 host the OracleAS Administration Console and OracleAS Web Cache Administration console, respectively.

  Note: All OracleAS Web Cache administration tasks are handled through the OracleAS Web Cache Administration console. However, you can use Oracle Process Manager and Notification (OPMN) as described in the *Oracle Application Server Web Cache Administrator's Guide*.

- Integration with OracleAS Web Cache adds a layer between the end-user browser and Oracle Access Manager components. Diagnosing connectivity issues requires investigation of both Web Cache logs and Oracle Access Manager component logs. Audit events can be captured between the browser and OracleAS Web Cache. These events can be correlated with the audit events captured between OracleAS Web Cache and WebGate. For more information about auditing Oracle Access Manager events, see the *Oracle Access Manager Identity and Common Administration Guide*.

  See Also:

  - POST Data Restoration Scenario
  - Cookieless Session Support Scenario

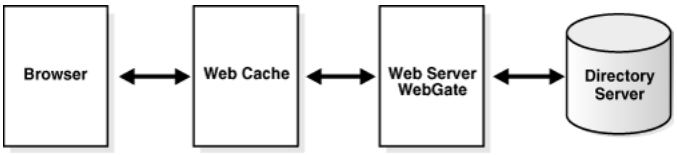
POST Data Restoration Scenario

With the integration of OracleAS Web Cache and Oracle Access Manager, POST data is retained and WebGate can retrieve it after the user re-authenticates. WebGate processes a POST data request after re-authentication only if WebGate gets the credential of the user who initiated the POST request.

Note: Without this integration with OracleAS Web Cache, WebGate does not store POST data if the POST request is interrupted by an authentication event (for example, a secure session timeout).

As a reverse proxy, OracleAS Web Cache acts a gateway to the back-end servers. Figure 4–2 illustrates the components and flow of information involved in POST-data restoration when Oracle Access Manager is integrated with OracleAS Web Cache.

Figure 4–2 POST Data Restoration Between OracleAS Web Cache and Oracle Access Manager



Process overview: POST data restoration with OracleAS Web Cache and Oracle Access Manager

1. User requests an Oracle Access Manager protected-resource from OracleAS Web Cache, which takes the request and generates a:
   - Private key (a unique identifier generated by OracleAS Web Cache for each request).
   - Cache ID (the id number for the owner cache). The cache ID is 0 by default because there is no cluster in OracleAS Web Cache.
   - Oracle Web Cache public cookie ORA_WX_SESSION. This is generated by the session binding mechanism, mostly when there are more than two Origin Servers. Therefore, it might be absent. It is not barred and sent to the browser during the cookieless transaction.
2. OracleAS Web Cache passes the request to WebGate, which forwards the request to Access Server, which challenges the user with form-based authentication.
3. After successful authentication and authorization, the requested page is sent to OracleAS Web Cache.
4. When the requested content is in the cache, OracleAS Web Cache sends the content to the user.
5. The user posts some data to the target resource received in Step 3, which is to be posted ahead to the Web server. To post the data, the user clicks the Submit button.

   This Submit Button is made available by the Web Application Developer which triggers another HTML page that consumes the posted data. If the session expires or the idle session timeout is reached, the user is challenged for re-authentication with a Form Login Page.
6. When a POST request is interrupted by re-authentication, WebGate returns a unique identifier (UID) to OracleAS Web Cache to be used as a key for cached post data:

   Protected Resource: If the resource was protected and the POST request interrupted by a timeout, the UID includes the user DN and OracleAS Web Cache private key.

   Web Cache Not Available: WebGate passes the request to Oracle Access Manager.
7. After successful re-authentication by the requesting user, the halted POST data request is processed by WebGate and the requested item is sent by OracleAS Web Cache.

Note: When restored POST data comes to WebGate, WebGate verifies that this request comes from the original user. If it is not from the original user, WebGate rejects posting data.

Cookieless Session Support Scenario

Without this integration, Oracle Access Manager uses cookies to store user identity information for protected resources that require the same level (or a lower level) of authentication. Within Oracle Access Manager, the Access Server creates a session-based encrypted single sign-on cookie (the ObSSOCookie) and sends it to WebGate after successful authentication. WebGate sends the ObSSOCookie to the Web browser.

Note: The Oracle Access Manager ObSSOCookie is a host cookie, accessible only by the Web site that issued it. The ObSSOCookie cannot be used to track user actions across different Web sites.

With the ObSSOCookie, Oracle Access Manager supports following types of single sign-on (SSO):

- **Single Domain SSO**: For the set of URLs within a single domain (for example, *company*.com).
- **Multi Domain SSO**: For the set of URLs within more than one domain (for example, *mycompany*.com and *yourcompany*.com).

Without this integration, the user session expires when the ObSSOCookie expires or goes away for any reason.

Cookieless Sessions: With this integration, cookieless session support is provided for SSO with OracleAS Web Cache in front of:

- One WebGate in a single domain
- Multiple WebGates in a single domain
- Multiple WebGates in multiple domains

Cookieless sessions are supported in several situations. For example:

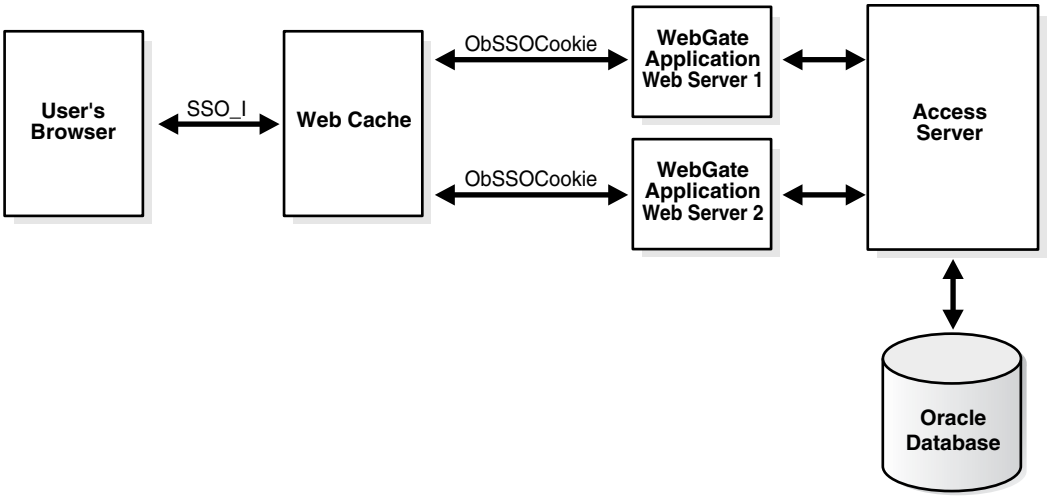
- **Disabled Cookies on a Web Browser**: The end user can explicitly turn off cookie support. A browser with cookies disabled must still be able to access a resource that is protected by Oracle Access Manager. In this case, the ObSSOCookie generated by Oracle Access Manager is stored in the OracleAS Web Cache server cache and can be retrieved by WebGate.
- **Mobile Device (or Devices that Cannot Process a Large Cookie)**: Electronic devices like mobile phones, PDAs, and other wireless devices have limited bandwidth. These devices might not be able to handle the ObSSOCookie due to the large amount of data in the cookie. This is similar to a cookie-disabled browser. However, in this case, the devices are mobile.
- **Privacy**: A user can disable or disallow all cookies to protect their own privacy. Certain government agencies mandate that cookies on certain government Web sites must not contain any personal identity data, even if the cookie is encrypted. The cookies are only allowed to contain session identifiers.

  Cookies can be used to track user movement among sites, which can be a sensitive issue in some countries and government agencies. The Oracle Access Manager

ObSSOCookie is a host cookie, accessible only by the Web site that issued it. The ObSSOCookie cannot be used to track user actions across different Web sites.

Figure 4–3 illustrates processing during a cookieless session. Additional details follow the figure.

Figure 4–3 Cookieless Support When Oracle Access Manager is Integrated with OracleAS Web Cache



The Oracle Access Manager ObSSOCookie is passed by WebGate to OracleAS Web Cache

Process overview: In a cookieless session

1. User requests an Oracle Access Manager protected-resource from OracleAS Web Cache.
2. OracleAS Web Cache intercepts the request and:
   - If resource is not cached, OracleAS Web Cache sends a request to the Application Web server.
   - If the resource is protected by an Oracle Access Manager policy, the user is authenticated.
3. On successful authentication, an encrypted single sign-on cookie (ObSSOCookie) is generated and sent to the WebGate hosted by the Application Web server.

   The ObSSOCookie is generated by the Access Server when a user authenticates successfully. This session-based cookie stores user identity information. For more information, see the*Oracle Access Manager Access Administration Guide*.
4. OracleAS Web Cache:
   - Intercepts the ObSSOCookie sent by WebGate to the browser, along with the requested resource
   - Stores the ObSSOCookie and responds to the client by the sending the requested page
   - Utilizes the stored cookie for authenticating the user on subsequent requests until the cookie expires
   - Sends its own public cookie ORA_WX_SESSION to the user's Web browser.

Note: OracleAS Web Cache cookie (ORA_WX_SESSION) expiration is not governed by Oracle Access Manager. For details about the OracleAS Web Cache cookie, see the *Oracle Application Server Web Cache Administrator's Guide*. However, ObSSOCookie expiration is governed by Oracle Access Manager as described in the *Oracle Access Manager Access Administration Guide*.

Supported Integrations

For the latest Oracle Access Manager certification information, see the following procedure.

See Also: "Integration Requirements"

To locate the latest certification details

1. Go to Oracle Technology Network:

   `http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html`

2. Locate Oracle Identity and Access Management, and then click the link for the latest Oracle Access Manager certification spreadsheet. For example:

   System Requirements and Supported Platforms for Oracle Access Manager 10gR3 (xls)

   `http://www.oracle.com/technology/products/id_mgmt/coreid_acc/pdf/oracle_access_manager_certification_10.1.4_r3_matrix.xls`

Integration Requirements

You must install and configure OracleAS Web Cache release 10.1.2.2.1 (or greater) as a reverse proxy in-front of a WebGate Web server. This routes all requests to the Web server through OracleAS Web Cache. This integration requires the OracleAS 10.1.2.3 patch.

The following task overview outlines the tasks that you must perform to implement the integration scenarios in this chapter.

Task overview: Install and integrate components

1. Install and configure OracleAS Web Cache based on your intended usage, as described in "Installing Components for the Integration" on page 4-8.
   - Configuration without SSL enabled can be used only for POST data restoration.
   - Configuration with SSL enabled can be used for cookieless session support (or POST data restoration).
2. Install and set up Oracle Access Manager, as described in "Installing Components for the Integration" on page 4-8.
3. **POST-Data Restoration**: Perform tasks in "Configuring OracleAS Web Cache for Post Data Restoration" on page 4-15.
4. **Cookieless Sessions**: Perform tasks in "Configuring OracleAS Web Cache for Cookieless Session Support" on page 4-15.

5. Protect resources using Oracle Access Manager, as described in "Protecting Resources with Oracle Access Manager" on page 4-16.
6. **Testing**: Perform tasks in "Validating and Testing the Integration" on page 4-19.

Installing Components for the Integration

This section introduces the tasks that must be performed to install components for the integration scenarios in this chapter.

You must install and configure OracleAS Web Cache release 10.1.2.2.1 (or greater) as a reverse proxy in-front of a WebGate Web server. This routes all requests to the Web server through OracleAS Web Cache.

Based on your intended usage the following conditions apply for installation and setup:

- Without SSL Enabled: Use this for POST data restoration only.

  Note: Cookieless session support requires that you configure Web Cache and Oracle Access Manager with SSL communication enabled.

- SSL-Enabled: Use this for cookieless session support (or POST data restoration).

  Note: For Post Data Restoration, you can integrate Web Cache and Oracle Access Manager in either an SSL-enabled or non-SSL enabled deployments.

The following task overview points to general steps provided in other books, and also to specific procedures that are included in this section. In the following procedure, a Windows example is featured.

Note: Oracle Universal Installer lays down ORACLE_HOME in which OracleAS Web Cache is installed.

Task overview: Install integration components

1. Install and set up Oracle Access Manager for the desired integration scenario, using information in the *Oracle Access Manager Installation Guide*.

   See Also: "Protecting Resources with Oracle Access Manager" on page 4-16 for details about the policy base, policy domain root, and creating a Master Access Administrator.

2. Install and set up OracleAS Web Cache as follows using instructions in the *Oracle Application Server Web Cache Administrator's Guide*:

   - Locate OracleAS Web Cache as either "J2EE and Web Cache"or "Web Cache Standalone" on the Oracle Technology Network at:

     `http://www.oracle.com/technology/software/products/ias/htdocs/101202.html`

   - Install OracleAS Web Cache in-front of an Oracle Access Manager WebGate Web server. For example: as_windows_x86_j2ee_webcache_101202.zip.

- Apply OracleAS patch 10.1.2.3 patch, as follows:
  - Apply patch 5983622 to install OracleAS Web Cache 10.1.2.3. For example: patch p5983622_10123_WINNT_webcache.zip.
  - Apply bundle patch 7438911, which includes Oracle Access Manager cookieless support. For example: p7438911_10123_WINNT.zip.
  - Apply patch 8255470.

  See Also: The My Oracle Support (formerly MetaLink) Web site at this URL for patches: `https://metalink.oracle.com`.

- Configure OracleAS Web Cache as a reverse proxy.

3. Proceed as follows to complete the initial configuration:
   - Without SSL Enabled: Use this for POST data restoration only. See "Configuring OracleAS Web Cache Without SSL Enabled" on page 4-9.
   - SSL-Enabled: Use this for cookieless session support (or POST data restoration). See "Configuring OracleAS Web Cache With SSL Enabled" on page 4-11.
4. Following these steps for initial installation and setup, proceed as follows to meet your requirements:
   - "Configuring OracleAS Web Cache for Post Data Restoration" on page 4-15.
   - "Configuring OracleAS Web Cache for Cookieless Session Support" on page 4-15.
   - "Protecting Resources with Oracle Access Manager" on page 4-16
   - "Validating and Testing the Integration" on page 4-19
   - "Auditing and Logging" on page 4-21

Configuring OracleAS Web Cache Without SSL Enabled

This topic describes how to configure Web Cache when SSL is not enabled in the deployment.

For Post Data Restoration, you can integrate Web Cache and Oracle Access Manager in either an SSL-enabled or non-SSL enabled deployments. However, cookieless session support requires that you configure Web Cache and Oracle Access Manager with SSL communication enabled.

When you finish the following steps, resources hosted on the Web server configured as the origin server can be accessed using the Web Cache site *computer:port* URL. Any caching rules specified in the Site definition are applied to the accessed resources.

To configure OracleAS Web Cache for integration without SSL

1. Install OracleAS Web Cache and apply 10.1.2.3.0 patch set., as described in "Installing Components for the Integration" on page 4-8.
2. Install Oracle Access Manager in Open mode on a non-SSL-enabled Web server, as explained in the *Oracle Access Manager Installation Guide*.
3. Access a OracleAS Web Cache administrative console. For example:

   http://*webcache_host.company.com:18100*

In this example, *webcache_host.company.com:18100* refers to the Oracle Application Server Administration site where the Oracle Application Server components, including OracleAS Web Cache, can be managed.

OR

http://*webcache_host.company.com:9400*

In this example, *webcache_host.company.com:9400* refers to a dedicated OracleAS Web Cache Administration site where Web Cache processes and configurations can be managed (as in the Oracle Application Server Administration site). Port 9400 can be configured in the Oracle Application Server Administration site.

4. From the System Components of installed Oracle Application Server on the Administrative console, click Web Cache.

   The following tabs are available: Home, Performance, Administration.

5. Click the Administration tab to view links for Web Cache configuration.
6. Under Properties, click the Ports link.
7. In the list of Listening ports, add a row to create a port for a Web Cache instance that can be used to access resources hosted on the configured back-end origin servers. For example:
   - IP Address: Enter * or the specific IP address of the computer hosting the Web Cache instance.
   - Port: Enter the unique port number for the Web Cache site used to access the resources hosted on the configured back-end origin server.
   - Protocol: Enter HTTP for a non-SSL site or HTTPS for an SSL-enabled site.
   - Client-side Certificate for HTTPS: Not Required.
   - Wallet for HTTPS: Leave blank.
   - Click OK and restart Web Cache.
8. Click the Administration tab.
9. Under Properties, click the Origin Server link.
10. Create a new entry for the origin server to be mapped with the Web Cache site.

    Origin Server holds the *computer:port* of the Web server on which protected resources are hosted. This origin server entry when mapped with a Web Cache site makes us available with the resources hosted on the Web server using the Web Cache Site URL created in Step 7.

11. Click Create and add an entry for Origin Server,. For example:
    - Host: Name of the computer hosting the resource Web Server
    - Port: Number of the Web server listening port
    - Protocol: HTTP
    - Retain default values for other details on the Origin Server configuration page.
    - Click OK and restart Web Cache.
12. Click the Administration tab.
13. Under Properties, click Sites.
14. In the Named Sites Definition, click Create, and enter the following details under the General tab:

- Host: Enter the name of the computer hosting Web Cache
- Port: Enter the unique port number for the Web Cache site used to access the resources hosted on the configured back-end origin server (as specified in Step 7).
- Prefix: Leave blank (optionally, enter as /*).
- In the Origin Server details for the Site, move the *computer:port* entry of the origin server created in Step 11, as follows:

  From: Available Origin Server List

  To: Selected Origin Server List

  Note: More than one origin server entry can be moved to a selected origin server list. The moved entries are then used based on the load balancing concept.

- Click OK and restart Web Cache.

15. Proceed as needed:
    - Configuring OracleAS Web Cache for Post Data Restoration
    - Configuring OracleAS Web Cache for Cookieless Session Support

Configuring OracleAS Web Cache With SSL Enabled

This topic describes how to configure OracleAS Web Cache when SSL is enabled. Cookieless session support requires that you configure Web Cache and Oracle Access Manager with SSL communication enabled.

Note: All normal listen ports (new and existing ones) must have SSL enabled because the cookie cache setting is global and cannot be turned on or off based on sites or ports. The cache does not work if some ports are SSL enabled while others are not. The only exceptions are the administration, invalidation, and statistics ports, which are not required to be SSL-enabled.

For Post Data Restoration, you can integrate Web Cache and Oracle Access Manager in either an SSL-enabled or non-SSL enabled deployment.

Following topics provide the steps you must perform:

- Creating a Wallet for OracleAS Web Cache SSL Communication
- Configuring OracleAS Web Cache for SSL Communication

Creating a Wallet for OracleAS Web Cache SSL Communication

This topic provides the steps to create a wallet for OracleAS Web Cache SSL communication.

To create a wallet for OracleAS Web Cache SSL-enabled communication

1. Install OracleAS Web Cache and apply 10.1.2.3.0 patch set.

2. Install Oracle Access Manager in Simple or Cert mode using SSL enabled Web servers. For details, see the *Oracle Access Manager Installation Guide*.
3. Open Wallet Manager by clicking Start, Programs, Orahome, Integrated Management Tools, Wallet Manager.
4. From the Wallet Console:
   - Create a new wallet: Under the Wallet menu, click New.
   - Enter and confirm the wallet password.
   - Wallet Type: Standard.
5. Request to create a certificate request. Click Yes to create the certificate request for Web Cache and perform the following steps using information for your environment:
   a. Fill in details for the certificate request, including:

      Common Name: Enter the fully-qualified domain name of the computer hosting the Web Cache instance.

      Organization Unit:

      Organization:

      Locality:

      State:

      Country:

      Key Size: The default value can be retained.

   b. Click OK.
   c. Submit the certificate request.

   Note: The certificate request can be saved to a file with a .csr extension. To save the request for a .csr file: Right-click the Certificate request node in Wallet Manager, click Export Certificate Request, and save to any desired location with .csr extension. Also: OpenSSL can be used for any non-Oracle HTTP Server environment. OCA can be used for an Oracle HTTP Server-based environment.

   ---

6. Get a signed from the Certificate Authority (CA).

   Note: If a trusted CA chain containing the OracleAS Web Cache certificate is imported, the following Step 7 is not required. Step 7 refers to the certificate generated on the creation of a new wallet, which is now signed by the Certificate Authority.

   ---

7. Import the CA signed Web Cache certificate into the Wallet.

   Note: If a common CA is used the following Step 8 is not required because it refers to the certificate of the Certificate Authority.

   ---

8. Import the certificate of the CA used to sign the certificate of the back-end Web server configured in SSL.
9. In the Wallet Manager, Wallet menu, check the Auto-Login box so it creates the .sso file of the wallet used during the SSL handshake activity.
10. Save the wallet to any desired location.
11. Verify that .sso and .p12 files are created from the above procedure
12. Proceed with "Configuring OracleAS Web Cache for SSL Communication"

Configuring OracleAS Web Cache for SSL Communication

Before you configure OracleAS Web Cache for SSL-enabled communication, you must create a wallet, as described in the previous topic.

When you finish the following steps, resources hosted on the Web server configured as the origin server can be accessed using the SSL-enabled Web Cache site *computer:port* URL. Any caching rules specified in the Site definition are applied to the accessed resources.

To configure OracleAS Web Cache for SSL-enabled communication

1. Install OracleAS Web Cache and apply 10.1.2.3.0 patch set.
2. Install Oracle Access Manager in Simple or Cert mode using SSL enabled Web servers. For details, see the *Oracle Access Manager Installation Guide*.
3. Access a Web Cache administrative console. For example:

   http://*webcache_host.company.com:18100*

   In this example, *webcache_host.company.com:18100* refers to the Oracle Application Server Administration site where the Oracle Application Server components, including OracleAS Web Cache, can be managed.

   OR

   http://*webcache_host.company.com:9400*

   In this example, *webcache_host.company.com:9400* refers to a dedicated OracleAS Web Cache Administration site where Web Cache processes and configurations can be managed (as in the Oracle Application Server Administration site). Port 9400 can be configured in the Oracle Application Server Administration site.
4. From the System Components of installed Oracle Application Server on the Administrative console, click Web Cache.

   The following tabs are available: Home, Performance, Administration.
5. Click the Administration tab to view links for Web Cache configuration.
6. Under Properties, click the Ports link.
7. In the list of Listening ports, add a row to create a port for a Web Cache instance that can be used to access resources hosted on the configured back-end origin servers. For example:
   - IP Address: Enter * or the specific IP address of the computer hosting the Web Cache instance.
   - Port: Enter the unique port number for the Web Cache site used to access the resources hosted on the configured back-end origin server.
   - Protocol: Enter HTTP for a non-SSL site or HTTPS for an SSL-enabled site.

- Client-side Certificate for HTTPS: Not Required.
- Wallet for HTTPS: Provide the exact location of the Wallet created in "Creating a Wallet for OracleAS Web Cache SSL Communication" on page 4-11.
- Click OK and restart Web Cache.

8. Click the Administration tab.
9. Under Properties, click the Origin Server link.
10. Create a new entry for the origin server to be mapped with the Web Cache site.

    Origin Server holds the *computer:port* of the Web server on which protected resources are hosted. This origin server entry when mapped with a Web Cache site makes us available with the resources hosted on the Web server using the Web Cache Site URL created in Step 7.
11. Click Create and add an entry for Origin Server,. For example:
    - Host: Name of the computer hosting the resource Web Server
    - Port: Number of the Web server listening port
    - Protocol: HTTPS
    - Retain default values for other details on the Origin Server configuration page.
    - Click OK and restart Web Cache.
12. Click the Administration tab.
13. Under Properties, click Sites.
14. In the Named Sites Definition, click Create, and enter the following details under the General tab:
    - Host: Enter the name of the computer hosting Web Cache.
    - Port: Enter the unique port number for the Web Cache site used to access the resources hosted on the configured back-end origin server (as specified in Step 7).
    - Prefix: Leave blank (optionally, enter as /*).
    - In the Origin Server details for the Site, move the *computer:port* entry of the origin server created in Step 11, as follows:

      From: Available Origin Server List

      To: Selected Origin Server List

      Note: More than one origin server entry can be moved to a selected origin server list. The moved entries are then used based on the load balancing concept.

    - Click OK and restart Web Cache.
15. Proceed as needed:
    - Configuring OracleAS Web Cache for Post Data Restoration
    - Configuring OracleAS Web Cache for Cookieless Session Support

Configuring OracleAS Web Cache for Post Data Restoration

To configure OracleAS Web Cache for POST Data restoration, you must modify the webcache.xml file.

For POST Data restoration, you must add the POSTBODYCACHE element under the <GENERAL> node, after the <ACCESSLOG> nodes (and the SITES node, if any), and before the <SITE> nodes.

Other attributes for the POSTBODYCACHE elements include:

- MAXAGE: The time, in seconds, that POST content can stay in the cache. The default is 120 seconds.
- MAXCACHEABLESIZE: The maximum size, in bytes, of POST request content that is considered cacheable. The default is 8152 bytes.

> **Note:** Take care when editing webcache.xml. Improper editing can result in errors.

To configure OracleAS Web Cache for POST data restoration

1. Locate webcache.xml in the following path:

   ```
   $ORACLE_HOME/webcache/config/webcache.xml
   ```

2. Add the POSTBODYCACHE element under the <GENERAL> node and before the <SITE> nodes, as shown in the following example:

   ```
   <GENERAL> ...
   <ACCESSLOG> ...
   <POSTBODYCACHE> ...
   <POSTBODYCACHE ENABLED="YES" MAXAGE="120" MAXCACHEABLESIZE="8152" />
   <SITE> ...
   ```

3. Restart the Web Cache server.
4. Proceed to "Protecting Resources with Oracle Access Manager".

Configuring OracleAS Web Cache for Cookieless Session Support

To configure OracleAS Web Cache for cookieless session support, you modify webcache.xml to include the COOKIECACHE element under the <GENERAL> node, after the <ACCESSLOG> nodes and before the <SITE> element.

Other attributes for the COOKIECACHE elements include

- COOKIENAME: 0 or more sub-elements can be defined as follows:

  ```
  <COOKIENAME="cookiename"/>
  ```

  Here *cookiename* represents the actual name of the cookie that Web Cache is to cache.

  > **Note:** If no COOKIE elements are specified, then all cookies are cached.

- REDIRECTURL: The SSO redirect URL for sign on that is used for domain cookie association.

- PASSNOCACHECOOKIES: For cookies that do not match the list of names specified in COOKIE elements you can determine if Web Cache passes them on or strip them of all responses. Values are:
  - YES: The default specifies that Web Cache passes cookies on to browser clients.
  - NO: Cookies are stripped from all responses.

The following procedure describes how to configure Web Cache for cookieless session support.

> **Note:** Take care when editing webcache.xml. Improper editing can result in errors. Also, cookieless session support requires that you configure Web Cache and Oracle Access Manager with SSL communication enabled. For details, see "Configuring OracleAS Web Cache With SSL Enabled" on page 4-11

To configure OracleAS Web Cache for cookieless session support

1. Locate webcache.xml in the following path:

   ```
   $ORACLE_HOME/webcache/config/webcache.xml
   ```

2. Add the COOKIECACHE element under the <GENERAL> node and before the <SITE> nodes, as shown in the following example:

   ```
   <GENERAL> ...
   <ACCESSLOG> ...
   <POSTBODYCACHE> ...
   <SITE> ...
   <COOKIECACHE ENABLED="YES" COOKIENAME="OBSSOCOOKIE"/>
   REDIRECTURL="http://www.oracle.com" />
   PASSNOCACHECOOKIES=No/>
   ```

3. Restart the Web Cache server.
4. Proceed to "Protecting Resources with Oracle Access Manager".

Protecting Resources with Oracle Access Manager

For the integration scenarios in this chapter to operate properly, Web server resources must be protected by an Oracle Access Manager policy domain and access policies. Before you protect resources, you must perform the following prerequisite tasks.

Prerequisites for the Master Administrator

1. Define the policy base during Policy Manager setup.

   To review the policy base from the Access System Console, click System Configuration, Server settings. On the View Server Settings page, locate the Policy Data Configuration section to obtain the computer name, port, root DN, directory server security, and policy base, as described in the *Oracle Access Manager Access Administration Guide*.

2. Define the policy domain root during Policy Manager setup.

   During Policy Manager setup, the Master Administrator specifies a policy domain root. This is a URL prefix under which all resources are protected. The default policy domain root must be broad to encompass all of your resources. The default

root is /. For more information, see the *Oracle Access Manager Access Administration Guide*.

3. Create a Master Access Administrator to create policy domains, resource types, and access control schemes, and to assign the role of Delegated Administrator of a policy domain to other people.

   Only a Master Administrator can create Master Access Administrators. A Master Access Administrator can perform any function in the Access System (except creating other Master Access Administrators), and can delegate administrative functions. Topics on configuring Master Access Administrators and delegating policy domain administration in the *Oracle Access Manager Access Administration Guide*.

A Master Access Administrator (or a Master Administrator) must create the first policy domain after the policy domain root is defined. He or she can then create policy domains for URLs beneath the first one and delegate administration of those policy domains to other administrators. Tasks in the following overview outline activities needed to create a policy domain the first policy domain to protect Web resources.

> **See Also:** The *Oracle Access Manager Access Administration Guide* for more information about each of the following tasks.

Task overview: Creating a policy domain

1. First Domain: If this is the first policy domain, the following tasks must be performed before the policy domain is created.

   a. Configure the resource types for any resources to be included in the domain (if the types are not already defined by default). You need these in task 2c.

      From the Access System Console, click the Access System Configuration tab, click the Common Information Configuration link in the left pane, then click the Resource Type Definitions sub-tab.

   b. Create the Master Audit Rule. This is needed in task 2f. The Access System does not log any audit information to the audit log file until the Master Administrator or Master Access Administrator creates a Master Audit Rule.

      The Master Audit Rule can be configured by a Master Access Administrator. Delegated Access Administrators can use the Master Audit Rule to create their own audit rules for policy domains and policies.

      From the Access System Console, click the Access System Configuration tab, click Common Information Configuration in the left pane, then click the Master Audit Rule sub-tab.

   c. Create an authentication scheme to use in the policy domain. A policy domain must have at least one authentication rule and therefore one authentication scheme. You need this scheme during task 2e.

      An authentication scheme includes the method used to challenge the user for credentials. It also includes one or more steps, each of which can include one or more plug-ins that perform part of the authentication process. A single-step scheme, or a chained scheme, requires an authentication flow.

      From the Access System Console, click the Access System Configuration tab, then click the Authentication Management link in the left pane.

   d. Create the authorization scheme to use in the policy domain. This is needed if if custom authorization schemes are to be utilized using authorization

plug-ins. Otherwise, the policy domain enables you to create specific authorization rules that make up an authorization expression.

You need an authorization scheme (also known as an authorization plug-in) for every authorization rule that you define. You can use the default authorization scheme, or provide a custom one. A policy domain requires an authorization expression containing at least one authorization rule. Once a scheme is defined, Delegated Access Administrators for different policy domains can use the same scheme in rules for their domains or in rules for policies within their domains.

From the Access System Console, click Access System Configuration, click Authorization Management in the left pane. You may want to view the contents and definition of existing authorization schemes before you create new ones.

2. Create the policy domain, as outlined here and described in the *Oracle Access Manager Access Administration Guide*:

   a. Create Policy Domain and specify general information:

      From the Policy Manager, click Create Policy Domain in the left pane.

   b. General: Click this tab and then enter general information about this policy domain (Name and optional Description).

   c. Resources: Click the Resources tab, click Add, select the Resource type (see Step 1, above), Host Identifier, URL prefix, enter a description, and click Save.

      The URL prefix is the starting point for resources in a policy domain. A URL prefix defines the beginning boundary of a policy domain (its first resource). A URL prefix maps to a directory on the file system of one of your application servers or Web servers.

   d. Authorization Rules: Specify general information, timing conditions, actions, and access (Allow or Deny) conditions.These fields are optional and if not specified use default values. However, most policy domains specify the access (Allow or Deny) conditions.

      Click the Authorization Rules tab, click Add, and enter general information about the rule, then Save.

      Timing Conditions: Click Timing Conditions. If none are defined, the rule is valid at all times. Add timing conditions as needed.

      Actions: Click Actions and specify an associated set of actions to occur in response to authorization success or failure when a user requests access to a protected resource.

      Allow Access: If no conditions are specified, no one is allowed access. Click Allow Access and add appropriate information.

      Deny Access: If no conditions are specified, no one is denied access. Click Deny Access and add appropriate details.

      Enable the Rule: If you do not enable this rule, it is not be available when you create an authorization expression. Click the Authorization Rules tab, click the box beside the rule, and click Enable.

   e. Default Rules: This is where you add the authentication rule and authorization expression.

      Authentication Rule: Click the Default Rules tab to display the General and Actions tabs for this rule. Click Add, and enter details for the rule, including

an authentication scheme (see task 3 above) that specifies how to verify a user's identity. Click Actions, click Add, and specify the optional actions for authentication failure, authentication success, or both.

Note: A policy domain must have one—and only one—authorization expression. The Authorization Expression is created from the available and defined authorization rules. If a custom authorization scheme is defined (Step 1d), you can select it from the list provided for the Authorization rule (Step 2d) in the policy domain. The other option in the list is the Oracle authorization scheme.

Authorization Expression: Click the Authorization Expressions tab, click Add, and create the expression. Duplicate Actions: If no policy is defined, the Access System level default policy for duplicate action headers is used. From the Duplicate Actions sub-tab, you can change this. Actions: From this sub-tab you can specify an associated set of actions in response to authorization success or failure or inconclusive results.

Audit Rule: If there is no Master Audit Rule defined, you are instructed to contact your Access System Administrator.

f. Policies: Before setting up a policy, decide the level of access control that is needed for the URL that you want to protect. For each policy you create, you can assign a specific authentication rule, authorization expression, and auditing rule. If no rules are defined, the default rules for the policy domain remain in effect (defined in task 2e).

Click the Policies tab, and fill in general information for this policy.

Click the name of the policy to display the sub-tabs for this policy.

Authentication Rule: Click this sub-tab and then click Add (or click View Default). Create (or edit) as authentication rule for this policy. The guidelines for this rule are the same as for the policy domain

Authorization Expressions: Click this sub-tab and then click Add (or click View Default). Create (or edit) an authorization expression for this policy. The guidelines for this expression are the same as for the policy domain.

Audit Rule: Click this sub-tab. If there is no Master Audit Rule defined and you want to add an auditing rule to this policy, you are instructed to contact your Access System Administrator.

g. Delegated Admins: Only Delegated Access Administrators who have rights to a specific domain—or the Master Access Administrator—can view a policy domain. Click the Delegated Admins tab.

h. When you are ready, enable the policy domain.

3. Test the policy domain, as described in the *Oracle Access Manager Access Administration Guide*.

For more information about setting up an Oracle Access Manager policy domain, authentication scheme, and access policies, see the *Oracle Access Manager Access Administration Guide*.

Validating and Testing the Integration

Here are some details for testing the integrations:

Validating POST Data Restoration

You can perform the following functions to test your POST data restoration configuration.

POST data restoration provides a function that prevents the WebGate from losing posted data if the POST request is interrupted by an authentication event. To test this, you can use any simple form-type resource containing fields where a user can enter data to be posted (for example, username and password fields to authenticate the user). The form resource used to post data, directs the posted data to a target resource page that collects the posted data. The authentication scheme used to protect this resource must use the Form challenge method.

During this test, you attempt to access the form resource and provide valid credentials. When the form appears, enter the data to be posted in the text fields provided and then wait for a period greater than idle session timeout to force re-authentication. This tests whether posted data can be restored after a re-authentication event. After the session timeout, submit the data and then provide user credentials when asked. You should be re-authenticated. Once re-authenticated, if the posted data is restored in the target resource page, POST data restoration is verified.

To test POST data restoration functionality

1. In a browser window, enter the Web Cache host and port as you access an Oracle Access Manager protected resource.
2. Login with appropriate user credentials to obtain the resource protected by Oracle Access Manager.
3. Enter values in the text fields: (i.e. user name and department name). For example:

   `user name` and *`department name`*

   Here "user name" and "department name" are not used to authenticate the user, but are used to illustrate the data that might be posted by the user.
4. Remain idle; do not make any modifications for a period longer than the idle session timeout.

   In Step 4 you are on the page containing text fields where the user enters the data to be posted. Remain idle for a period longer than the idle session timeout to force a re-authentication. During this period, do not perform any operation or modification on this page. This allows idle session time to expire.
5. Enter the remaining fields and click "Submit" or "Next" which ever applies.

   Entering data into remaining text fields is an optional step. The required action to be performed here is submitting the data to be posted after the idle session timeout occurs.

   After the idle session timeout occurs, you are asked to re-authenticate.
6. Provide user credentials when asked to obtain the resource protected by Oracle Access Manager.

   After successful re-authentication, you are directed to the requested page. The page should be able to populate the Posted data from Web Cache.

Validating Cookieless Session Support

You can perform the following functions with Oracle Access Manager to test cookieless session support.

To accomplish this task you must have added the XML tag COOKIECACHE into the webcache.xml file. For details, see "Configuring OracleAS Web Cache for Cookieless Session Support" on page 4-15.

To test cookieless session support

1. In a browser window, enter the Web Cache host and port as you access an Oracle Access Manager protected resource.
2. Login with appropriate user credentials to obtain the resource protected by Oracle Access Manager.

   Access to the resource should be granted.
3. Check for headers received by the Web browser using tools like HTTPheaders.

   Oracle does not provide tools to check for headers received by the Web browser.
4. An OracleAS Web Cache-generated public cookie (ORA_WX_SESSION) should be seen as received by the Web browser. The Oracle Access Manager ObSSOCookie should not be sent to the Web browser.
5. Restart the Web Cache server.

Auditing and Logging

The Oracle Access Manager auditing feature collects and presents data pertaining to policy and profile settings, system events, and usage patterns. You can record all dynamic audit reports and some static audit reports to disk file, to a relational database, or both. Some static reports can also be displayed in limited form through the graphical user interface.

In addition to auditing, an Oracle Access Manager component instance can write information about its processes and states to a log file. The logs can be configured to provide information at various levels of granularity. For example, you can record errors, errors plus state information, or errors, states, and other information to the level of a debug trace. You can also eliminate sensitive information from the logs.

For more information about auditing and logging with Oracle Access Manager, see the *Oracle Access Manager Identity and Common Administration Guide*.

With Oracle Enterprise Manager 10g Application Server Control Console or OracleAS Web Cache Manager, you can view a list of the most popular requests and a list of the contents of the cache.

OracleAS Web Cache events and errors are stored in an event log. The event log can help you determine which objects have been inserted into the cache. It can also identify listening port conflicts or startup and shutdown issues. OracleAS Web Cache generates an access log that contains information about the HTTP requests sent to OracleAS Web Cache.

For more information about using diagnostic tools with Oracle Web Cache, see the *Oracle Application Server Web Cache Administrator's Guide*.

Tips and Troubleshooting

This section provides tips to help you troubleshoot any issues.

Double Authentication on Login

Problem

When Post Data Restoration is enabled, double authentication might occur on login with a user other than the one who logged in to the original session. The target resource is made available to the end user based on the authorization set for the protected resource. If a user attempts access with the credentials of a user who cannot be authenticated, the login form appears until the maximum login tries limit is exceeded.

Cause

This is expected behavior. On a re-authentication event, the user can log in with the same credentials or those of another known user. However, in the second case, the login form appears twice.

Solution

A user must authenticate twice if they are using the same browser on which another user already tried to log in.

Redirection or Session Restoration Issues

Problem

Whenever Web Cache is configured in front of Oracle Access Manager for post-data restoration and cookieless support, enabling a challenge redirect for authentication to a separate WebGate does not function properly if the WebGate host port are directly provided in the authentication scheme definition. The redirection occurs but the features may not serve their purpose. In the case of SSL transactions, session restoration issues might occur.

Cause

If the host port of the authenticating WebGate is directly provided in the Oracle Access Manager authentication scheme definition, redirections to the authenticating WebGate during authentication do not take Web Cache into account.

Solution

When enabling a challenge redirect for Oracle Access Manager wherein a separate authenticating WebGate is to be used for authentication, then you must configure a Web Cache site that corresponds to the authenticating WebGate host port.

To solve the redirection or session restoration issue

1. Ensure that a Web Cache site has been defined for the corresponding authenticating WebGate.

2. Confirm that the Web Cache site's host port is mentioned in the challenge redirect field of the authentication scheme to redirect authentication to the specified back-end WebGate.

Non-SSL Oracle Access Manager with SSL OracleAS Web Cache and Cookieless Session

Problem

The https session is converted into a http session, which leads to an inaccessible site. The issue is mainly observed on accessing folder structure-based resources.

Cause

Cookieless sessions demand SSL communication between the OracleAS Web Cache site and the Oracle Access Manager originating server. As a result, a non-SSL-enabled Oracle Access Manager originating server. can lead to an error.

Solution

You must add information in the Oracle HTTP Server (OHS or OHS2) httpd.conf file to configure a non- SSL hosted Oracle Access Manager with an SSL-enabled OracleAS Web Cache site and a cookieless session.

> **Note:** This solution is not yet supported for other Web server types.

To support a cookieless session with a non- SSL hosted Oracle Access Manager and an SSL-enabled OracleAS Web Cache site

1. Locate and open the OHS or OHS2 http.conf file in a text editor. For example:

   ```
   OHS_install_dir/conf/httpd.conf
   ```

2. OHS: Add the following information to the end of the httpd.conf file.

   ```
   LoadModule certheaders_module libexec/mod_certheaders.so
   AddCertHeader HTTPS
   SimulateHttps On
   ```

3. **OHS2**:

   ```
   LoadModule certheaders_module modules/mod_certheaders.so
   AddCertHeader HTTPS
   SimulateHttps On
   ```

4. Save the file.

Target HTML Files Available with No Authentication on Subsequent Access Attempts

Problem

Accessing Oracle Access Manager-protected HTML files using Web Cache site URLs might cause the sessions to get copied if the HTML caching rule is enabled for Web Cache. This results in authentication only in the first access attempt to access the HTML file. On subsequent access attempts, the file might be available to the user without authentication. Also, the HTML caching rule might cause the cookies generated on the first successful access attempt to be utilized for further access to the same HTML file.

Cause

The Web Cache HTML caching rule causes the entire HTML file and its contents to be cached for future rapid access.

Solution

If a HTML caching rule is enabled for the Web Cache, then it may cause session copies for protected HTML files. It may cause to have the target HTML files available without authentication.

Disable the HTML caching rule from the Web Cache Administration Console.

5

Integrating with Oracle Application Server 10*g*: OC4J

This chapter describes integrating Oracle Access Manager with OracleAS Single Sign-On 10g (which for authentication and authorization. Either Oracle Access Manager or OracleAS Single Sign-On can act as the authentication engine.

This integration enables you to provide identity management functionality across Web-based applications that run on Oracle Application Servers, for example, Oracle E-Business Suite, Oracle Forms, Portals, and other Access System-protected resources.

This chapter covers the following topics:

- Integration Overview and Environment Preparation
- Single Sign-On with OracleAS 10g
- Authorization Support for Applications Protected by OracleAS Single Sign-On
- Testing the Integration with OracleAS
- OracleAS 10g Files
- Troubleshooting the OracleAS 10g Integration

> **Note:** This chapter does not describe configuring single sign-on for Oracle Fusion Middleware. For those details, see the 11g *Oracle Fusion Middleware Security Guide*.

Integration Overview and Environment Preparation

This section discusses the following topics:

- Supported Authentication Schemes for the Oracle Application Servers
- OracleAS 10g Infrastructure
- Integration Architecture
- Supported Integrations
- Preparing Your Environment

Supported Authentication Schemes for the Oracle Application Servers

Oracle Access Manager provides authentication and single sign-on for OracleAS 10g. This enables you to use a single user name and password (and optionally a realm ID),

to log in to all features of the Oracle Application Servers and other Web applications. The integration uses the following authentication schemes:

- Form based
- Basic
- Custom
- Integrated Windows Authentication

OracleAS 10g Infrastructure

OracleAS 10g applications provide a similar infrastructure and a security framework for single sign-on for Oracle and other partner applications. The integration of Oracle Access Manager single sign-on with OracleAS 10g involves the following components.

OracleAS Single Sign-On Server: This enables Oracle applications to accept authentication from other applications. You can enable single sign-on between Access System-protected applications and applications protected within the OracleAS 10g single sign-on framework. You can use a single user name and password and optionally a realm ID to log in to all features of the Oracle Application server and other Web applications.

Oracle HTTP Server: This is the Web server interface for OracleAS 10g. Oracle HTTP Server is the integration point between Oracle Access Manager and OracleAS 10g.

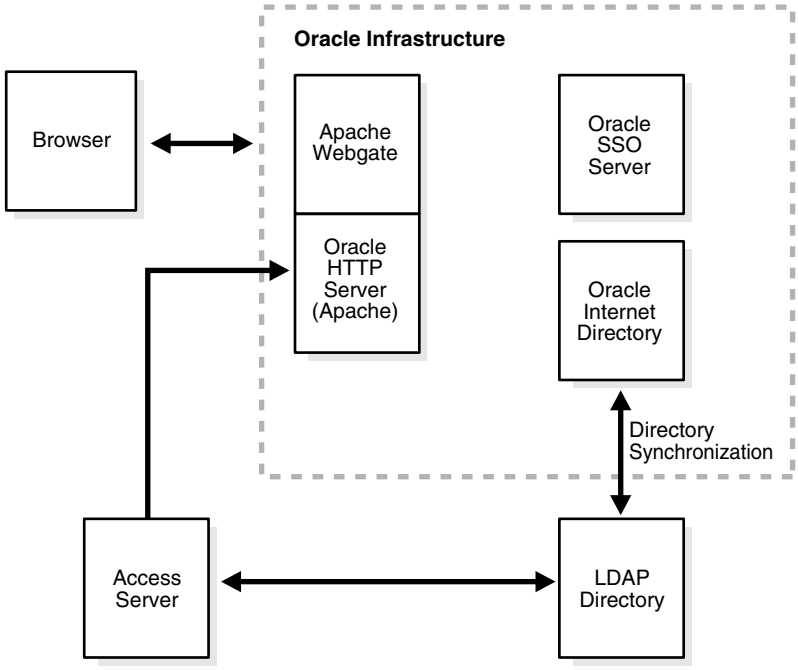
Note: You must use the 10*g* (10.1.4.2) WebGate for Oracle HTTP Server.

Oracle Internet Directory: The LDAP directory that serves as a user repository for OracleAS 10g applications. Oracle Internet Directory can be synchronized with other connected directories.

Integration Architecture

Figure 5–1 illustrates the integration between Oracle Access Manager and Oracle Application Servers.

Figure 5–1 Oracle Access Manager and Oracle Application Server Integration Architecture



Process overview: Integration of Oracle Access Manager with Oracle Application Server

1. When a user attempts to access an Oracle Access Manager-protected application or Web resource, a WebGate intercepts the request.
2. WebGate requests the security policy from the Access Server to determine if the resource is protected.
3. When the resource is protected, WebGate prompts the user to authenticate.
4. The credentials entered by the user are validated against the directory for authentication.
5. When authentication is successful, an encrypted Oracle Access Manager single sign-on cookie is set on the user's browser.
6. After successful authentication, the Access System determines if the user is authorized by applying policies that have been configured for the resource.
7. Upon successful authorization, the Access System executes the actions that have been defined in the security policy and sets an HTTP header variable that maps to the OracleAS 10g user ID.
8. The OracleAS Single Sign-On Server recognizes the Oracle Access Manager HeaderVar, authenticates the user, and sets the Oracle single sign-on Cookie.

Note: Oracle Internet Directory must be synchronized with the Oracle Access Manager directory to ensure that user data is up-to-date. Oracle Internet Directory performs the synchronization.

Supported Integrations

For the latest support information, see the Oracle Technology Network (OTN).

To locate the latest certification details

1. Go to Oracle Technology Network:

   http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

2. Locate Oracle Identity and Access Management, and then click the link for the latest Oracle Access Manager certification spreadsheet. For example:

   System Requirements and Supported Platforms for Oracle Access Manager 10gR3 (xls)

   http://www.oracle.com/technology/products/id_mgmt/coreid_acc/pdf/oracle_access_manager_certification_10.1.4_r3_matrix.xls

Preparing Your Environment

The following task overview lists the requirements for preparing for configuring single sign-on.

Task overview: Preparing your Environment

1. Install OracleAS 10g.
2. Install the OracleAS Infrastructure 10g, which includes.
   - Oracle Application Server Metadata Repository
   - OracleAS Single Sign-On Server
   - Oracle Internet Directory (a lightweight directory access protocol (LDAP))

   Note: The servers where the Oracle infrastructure and Oracle Access Manager are installed must have fully qualified domain names, for example, *hostname.domain*.net.

3. Install and set up Oracle Access Manager components.

   See the *Oracle Access Manager Installation Guide* for details. Install the following:
   - Identity Server
   - WebPass
   - Policy Manager
   - Access Server
4. On the computer hosting the Oracle HTTP Server, install a WebGate for use with OracleAS 10g as described in the *Oracle Access Manager Installation Guide*, and choose one of the following options to update the Web server configuration file:
   - **Automatic Web Server Updates**: Click Yes to automatically update your Web server configuration file (Oracle HTTP Server httpd.conf) during WebGate installation, as described in the *Oracle Access Manager Installation Guide.*
   - **Manual Web Server Updates**: Use one of the following methods:

Either: Locate the Oracle HTTP Server httpd.conf file after WebGate installation, add the WebGate entry at the end of the file, then run the following commands on an infrastructure terminal:

```
Opmnctl restartproc process-type=HTTP_Server
```

Or: Use the Oracle Enterprise Manager Console to:

a. Launch the Oracle Enterprise Manager.

b. Select the Oracle Application Server hosting the Oracle Infrastructure.

c. Select the HTTP Server hosting the WebGate.

d. Navigate to Advanced Server Properties.

e. From the list of configured files, select httpd.conf for update.

f. Include the WebGate entry at the end of the file.

5. Restart the Oracle HTTP Server after the Web Server configuration file update.
6. Configure OracleAS Single Sign-On for integration with third-party access management systems.

   See the related chapter in the *Oracle Application Server Single Sign-On Administrator's Guide 10g (10.1.4.0.1)* on Oracle Technology Network at: `http://www.oracle.com/technology/documentation/oim1014.html`.
7. Configure the Web browser to allow cookies.
8. Proceed to "Single Sign-On with OracleAS 10g" on page 5-5.

Single Sign-On with OracleAS 10g

When integrating Oracle Access Manager with OracleAS 10g Application Server, each OracleAS application's configuration is provided separately. This integration requires configuring OracleAS 10g to integrate with third-party access management systems and configuring Oracle Access Manager logout.

You complete the following procedures to set up OracleAS 10g for the integration:

- Enabling Single-Sign On
- Integrating the Delegated Administration Service
- Integrating the Portal
- Enabling Single-Sign On for Forms
- Integrating Reports Services
- Synchronizing the Oracle Internet Directory and Oracle Access Manager LDAP Directory
- Implementing Global Logout from OracleAS Single Sign-On and Access Server

Task overview: Integrating Oracle Access Manager with OracleAS 10g

1. Set up your computers, as described in "Preparing Your Environment" on page 5-4.
2. Set up the OracleAS.
3. Set up Oracle Access Manager, as described in "Configuring Oracle Access Manager 10g (10.1.4.3) for Integration with OracleAS 10g" on page 5-10.

4. Test the integration, as described in "Testing the Integration with OracleAS" on page 5-17.

Enabling Single-Sign On

Enabling single-sign on for the integration between Oracle Access Manager and OracleAS 10g includes creating a java class and editing the policy.properties file, as discussed in the following paragraphs.

Creating the Java Class for Integration

The first step in enabling single sign-on for the integration involves coding a Java class, which looks for the Header variable from Oracle Access Manager.

> **Note:** This example assumes you have installed and set up the Identity System and Access System, created a policy domain in the Access System, defined an authorization action that sets a Header Variable with the ID of the user, and configured global logout. See "Protecting the Single-Sign On Login URL" on page 5-13 and "Implementing Global Logout from OracleAS Single Sign-On and Access Server" on page 5-9 for details.

To code a JAVA class to look for a Oracle Access Manager HeaderVar

1. In the Access System, create rules to protect the following URIs:

   `/sso/auth/`

   `/pls/orasso/orasso.wwsso_app_admin.ls_login`

   See "Protecting the Single-Sign On Login URL" on page 5-13 for details.

2. Create a Java file for your package.

   For help, copy the source code from "SSOOblixAuth.java" on page 5-18, or the Sample Files section #SSOOblixAuth.java in the following location:

   ORCLE_HOME`/sso/lib`

   Save the file as SSOOblixAuth.java. Before it is compiled, this package directive must be added to it:

   ```
   package oblix.security.ssoplugin;
   ```

3. Compile the file, including *ORACLE_HOME*`/sso/lib/ipastoolkit.jar` in the class path. The sample file SSOOblixAuth.java is compiled this way:

   ```
   ORACLE_HOME/jdk/bin/javac -classpath
   ORACLE_HOME/sso/lib/ipastoolkit.jar:ORACLE_HOME/lib/servlet.j
   ar -d ORACLE_HOME/sso/plugin SSOOblixAuth.java
   ```

   Note that the colon separator (":") is appropriate for Linux. On Windows, use a semicolon (";") as the separator.

   This command creates SSOOblixAuth.class and places it in the directory *ORACLE_HOME*`/sso/plugin/oblix/security/ssoplugin`.

4. Next you must register the Java class for integration by editing the policy.properties file in the following location:

   OracleAS_install_dir`/sso/conf`

Where *OracleAS_install_dir* is the directory where OracleAS Single Sign-On infrastructure is installed.

5. In the OracleAS Single Sign-On policy.properties file, replace the simple authentication plug-in with the plug-in that you created in the previous steps. In this class, navigate to the line MediumSecurity_AuthPlugin:

   ```
   MediumSecurity_AuthPlugin =
   oracle.security.sso.server.auth.SSOServerAuth
   ```

   Comment out the existing line and add a new line to register your Java class, as follows:

   ```
   MediumSecurity_AuthPlugin =
   oblix.security.ssoplugin.SSOOblixAuth
   ```

   When editing policy.properties, take care not to insert blank space at the end of a line.

6. Save the file.

7. Restart the single sign-on middle tier, and restart the OC4J instance OC4J_SECURITY to have your changes to take effect:

   ```
   ORACLE_HOME/opmn/bin/opmnctl restartproc
   process-type=HTTP_Server
   ```

   ```
   ORACLE_HOME/opmn/bin/opmnctl restartproc
   process-type=OC4J_SECURITY
   ```

8. Test the integrated system

Integrating the Delegated Administration Service

The Delegated Administration Service (DAS) is part of Oracle Identity Management 10g, an integrated infrastructure that includes the following components:

- Oracle Internet Directory—An LDAP V3-compliant directory service
- Delegated Administration Service (DAS) 10g—The Oracle Internet Directory component that provides trusted proxy-based administration of directory information by users and application administrators.
- Oracle Directory Integration Service—A component of the Oracle Internet Directory that permits synchronization between the Oracle Internet Directory and other directories and user repositories.
- Provisioning Integration Service—The Oracle Internet Directory component that provides automatic provisioning of services, as described in Oracle documentation.

The Delegated Administration Service is installed by default when you install the OracleAS 10g Infrastructure, and should integrate automatically. No additional steps are needed for a user to access Delegated Administration Service when Oracle Access Manager is integrated with single sign-on.

The Delegated Administration Service link is:

`http://infra-computer-name:port/oiddas`

Note: If you experience errors using Create/Edit user and Create/Edit groups portlets, move the Delegated Administration Service to the middle tier from the Infrastructure. For details, see "Integrating the Portal" on page 5-8.

Integrating the Portal

The Oracle Application Server Portal enables you to build, deploy, and maintain self-service, integrated Enterprise Information Portals (EIPs). A customized portal page can present information from different providers and can include both enterprise search and directory lookup fields.

A portal page consists of multiple portlets. Each portlet is a region of the portal page that provides dynamic access to a Web-based resource.

When Oracle Access Manager single sign-on is integrated with OracleAS 10g, users should be able to access the portal as follows:

`http://`*`midtier_home`*`:`*`port`*`/pls/portal`

Note: The Create/Edit user and Create/Edit groups portlets call the DAS from the portal. If you experience errors using Create/Edit user and Create/Edit groups portlets, you must move the DAS to the middle tier from the Infrastructure.

Enabling Single-Sign On for Forms

The Oracle Application Server Forms Services is a middle-tier application framework that you use to deploy complex transactional forms applications to the internet.

When you integrate Oracle Access Manager with OracleAS 10g, you must enable single sign-on for forms. Once single sign-on is enabled for forms, Oracle Access Manager handles authentication and you should not be challenged to enter the schema user ID and password either by the single sign-on login page or by the forms.

To enable single sign-on for forms

1. Locate the forms90.conf file located in the following directory:

   `midtier_home``/forms90/server`

2. At the end of the forms90.conf file add the following lines.

```
<IfModule mod_osso.c>
   <Location /forms90/f90servlet>
      require valid-user
      AuthType Basic
   </Location>
</IfModule>
```

3. Restart OC4J_BI_FORMS and the forms server to have you changes take affect.

   Next you create a Resource Access Descriptor (RAD) for the Oracle Internet Directory users. A RAD can be created at a global level so all users can use the same RAD to access the resource. Alternatively, the RAD can be created for each user.

4. Create a Resource Access Descriptor (RAD) for the Oracle Internet Directory users to map the LDAP user to the Database schema.

The next step can be done at the global level in the formsweb.cfg file (the default configuration), or at the application level to make individual applications single sign-on enabled.

5. Set the ssoMode to true to make the application single sign-on enabled using the Enterprise Manager to update the formsweb.cfg file.

   For example, to make an individual application single sign-on enabled:

   ```
   [myApp]
   form=myFmxs
      ssoMode=true
   ```

   For more information, see chapter 6 in the *Oracle Application Server Forms Services Deployment Guide 10g (9.0.4) for Windows and UNIX*, Part No. B10470-02.

6. Test this implementation by navigating to the following URL:

   `http://midtier_home:port/forms90/f90servlet?config=default`

Integrating Reports Services

The Oracle Application Server Reports Services allow you to deploy reports to the OracleAS 10g, as described in your Oracle documentation.

Reports are single sign-on-enabled out of the box and should work without further steps when you integrate Oracle Access Manager with OracleAS 10g.

To access the protected reports page

1. Point your browser to the following URL:

   `http://computer:port/reports/rwservlet/showenv`

2. Log in when challenged by WebGate.

3. Confirm that once authenticated you can view the Environment settings for Oracle Reports (an single sign-on-protected page).

For more information, see chapter 10 of the *Oracle Application Server Reports Services Publishing Reports to the Web 10g (9.0.4),* Part No B13673-01.

Synchronizing the Oracle Internet Directory and Oracle Access Manager LDAP Directory

The next step in the configuration of OracleAS 10g for integration with Oracle Access Manager is to use the Oracle synchronization tool to synchronize user information between the Oracle Internet Directory and the LDAP directory server used by Oracle Access Manager.

For details about this synchronization tool and process, see the Oracle Internet Directory documentation.

> **Note:** To test the integration without synchronizing the directories, you must create an Oracle administrator (oracladmin) within Oracle Access Manager for login purposes.

Implementing Global Logout from OracleAS Single Sign-On and Access Server

By default, the WebGate logs a user out when it receives a URL containing "logout." See the section on logout from a single domain single sign-on session in the *Oracle*

Access Manager Access Administration Guide for details. As a result, the default single sign-on logout page does not work with OracleAS Single Sign-On. The discussion "Logout.jsp" on page 5-19 provides a sample file you that need to configure logout.

To implement global logout from OracleAS Single Sign-On

1. Edit the following parameters in *ORACLE_HOME*/sso/conf/policy.properties. Substitute the paths to your logout page for the value shown in the following example:

   ```
   #Deployment login page link
   loginPageUrl = /sso/pages/login.jsp
   logoutPageUrl = /sso/pages/logout.jsp
   ```

2. Restart the single sign-on server:

   ```
   ORACLE_HOME/opmn/bin/opmnctl restartproc process-type=HTTP_Server
   ORACLE_HOME/opmn/bin/opmnctl restartproc process-type=OC4J_SECURITY
   ```

3. In the Access System, go to the page where you configure the single sign-on logout URL.

   From the Access System Console, click System Configuration, then click Server Settings, then click Configure SSO Logout URL.

4. On this page, configure the single sign-on logout URL to invoke the OracleAS Single Sign-On logout URL.

   Add a logout URL similar to the following:

   http://*host*:*port*/sso/logout

   Where *host* is the computer where the OracleAS Single Sign-On server is installed and *port* is the listen port for the server. When the user clicks the Logout link in Oracle Access Manager, the logout URL removes session cookies and redirects users to a logout page. See the appendix on configuring logout in the Oracle Access Manager Access Administration Guide for details.

5. Go to the page where you configure the WebGate logout URL from the Access System Console by clicking Access System Configuration, then click AccessGate Configuration, then select a WebGate.

6. On the page that shows the WebGate details, click Modify, then provide a new logout URL similar to the following:

   /access/oblix/lang/en-us/style2/oblixlogo.gif

   The URL can be any gif file or Web page. This page is embedded in logout.jsp. See "Logout.jsp" on page 5-19 for details.

7. Repeat the previous two steps for every WebGate-protected cookie domain.

8. Add a page that you want to display after the user is logged out.

9. Confirm that you can perform a global logout both from Oracle AS Single Sign-On Server and from the Access Server.

Configuring Oracle Access Manager 10*g* (10.1.4.3) for Integration with OracleAS 10g

After installing Oracle Access Manager and installing a WebGate on the OracleAS HTTP Server, you must create Oracle Access Manager access control policies to protect OracleAS resources.

Task overview: Setting up Oracle Access Manager for integration with OracleAS 10g includes

1. Install and set up the Identity System and Access System, as outlined in "Preparing Your Environment" on page 5-4.
2. Navigate to the Identity System Console and create an Oracle Administrator (orcladmin) user to match the orcladmin user who already exists in the Oracle Internet Directory, as described in the *Oracle Access Manager Identity and Common Administration Guide*.
3. Complete "Protecting the Single-Sign On Login URL" on page 5-13.

Configuring the Access System for OracleAS Single Sign-On 10.1.2.0.2

In addition to following the other information in this chapter, you must also complete the following procedure to integrate the Access System with OracleAS Single Sign-On 10.1.2.0.2.

To configure the integration with OracleAS Single Sign-On 10.1.2.0.2

1. Follow the steps in this chapter on configuring the integration.
2. In the Access System Console, click **System Configuration**, then click **Server Settings**, and configure the following logout URL:

   ```
   http://[host.domain]:[port]/pls/orasso/ORASSO.wwsso_app_admin.ls_logout?p_done_
   url=http%3A%2F%2F[host.domain]%3A[port]
   ```

   URL-encode the `p_done_url` value.

   See the *Oracle Application Server Single Sign-On Administrator's Guide* for release 10.1.2.0.2 for details on configuring the logout link for single sign-on. A sample JSP that can be used for this purpose is included at the end of this release note.
3. If you use the following sample JSP, go to the Access System Console, click **Access System Configuration**, then click **AccessGate Configuration**, and include the following in the **LogOutURLs** parameter for every WebGate in your environment:

   ```
   /access/oblix/lang/en-us/style2/oblixlogo.gif
   ```

The following is a sample `logout.jsp` file:

```
<!-- Copyright (c) 1999, 2003, Oracle. All rights reserved. -->
<%@page autoFlush="true" session="false"%>
<%
// Declare English Message Strings
String msg1 = "Single Sign-Off";
String msg2 = "Application Name";
String msg3 = "Logout Status";
String msg4 = "ERROR: The return URL value not found.";
String msg5 = "ERROR: Logout URL for partner applications not found.";
// Get the user language preference
String userLocaleParam = null;
java.util.Locale myLocale = null;
// Get the user locale preference sent by the SSO server
try
{
userLocaleParam = request.getParameterValues("locale")[0];
}
catch(Exception e)
{
userLocaleParam = null;
```

```
}
if( (userLocaleParam == null) || userLocaleParam.equals("") )
{
myLocale = request.getLocale();
}
else
{
if(userLocaleParam.indexOf("-") > 0 )
{
// SSO server sent the language and territory value (e.g. en-us)
myLocale = new java.util.Locale(userLocaleParam.substring(0, 2),
userLocaleParam.substring(3, 5));
}
else
{
// SSO server sent only the language value (e.g. en)
myLocale = new java.util.Locale(userLocaleParam, "");
}
}
// The following two lines will be used only for the Multilingual support
with
// proper resource bundle class supplied
// java.util.ResourceBundle myMsgBundle
// = java.util.ResourceBundle.getBundle("MyMsgBundleClassName", myLocale);
// Get the message string in the appropriate language using the message key.
// Use this string to display the message in this page.
// String mesg = myMsgBundle.getString("mesg_key");
%>
<html>
<body bgcolor="#FFFFFF">
<h1><%=msg1%></h1>
<%
String done_url = null;
int i = 0;
// Get the return URL value
try
{
done_url = request.getParameterValues("p_done_url")[0];
}
catch(Exception e)
{
done_url = "";
}
// Get the application name and logout URL for each partner application
try
{
%>
<b> <%=msg2%>   <%=msg3%> </b>
<br>
// Substitute an actual host, domain, and port for
myhost.us.mydomain.com:7777
// that points to the WebGate.
<img
src="http://myhost.us.mydomain.com:7777/access/oblix/lang/en-us/style2/oblixlo
go.gif">
<%
for(;;)
{
i++;
String app_name = request.getParameterValues("p_app_name"+i)[0];
```

```
String url_name = request.getParameterValues("p_app_logout_url"+i)[0];
%>
<%=app_name%>

<img src="<%=url_name%>">
<br>
<%
}
}
catch(Exception e)
{
if(done_url == null)
{
%>
<%=msg4%> <br>
<%
}
if(i>1)
{
%>
<br> <a href="<%=done_url%>">Return</a>
<%
}
else
{
%>
<%=msg5%><br>
<%
}
}
%>
</body>
</html>
```

Protecting the Single-Sign On Login URL

You must protect the following single sign-on login URL so that the WebGate challenges the user whenever the OracleAS Single Sign-On 10g is accessed:

```
/sso/auth/
```

The following activities are required to protect the single sign-on login URLs, or any other resources, using the Access System.

Each step in the following task list is a full procedure. For complete details, see the related chapters in this guide.

Task overview: Protecting resources with Oracle Access Manager

1. Define an authentication scheme using the Access System Console.

   For example:

   ```
   Access System Console, Access System Configuration, Authentication Management,
   Add
   ```

2. Create a policy domain using the Policy Manager.

   For example:

   ```
   Policy Manager, Create Policy Domain
   ```

3. Add a resource to your policy domain using the Policy Manager.

   For example:

   ```
   Policy Manager, Create Policy Domain, Resources
   ```

4. Define rules for your policy domain using the Policy Manager.

   For example:

   ```
   Policy Manager, Create Policy Domain, Default Rules
   ```

5. Define an Authorization action that sets a Header Variable with the ID of the user.

   For example:

   ```
   Policy Manager, Create Policy Domain, Default Rules, Authorization Expressions,
   Actions
   ```

   Authorization Success

   Return

   Type: HeaderVar

   Name: *XXX*_REMOTE_USER

   Return Attribute: *loginAttribute*

   where *XXX* is any prefix (used because "REMOTE_USER" is often an internal header for HTTP servers) and where *loginAttribute* is the attribute configured as the Login semantic type in the Identity System. This name must map to the login name of the user stored in the OracleAS single sign-on repository. Some people have used the "EMPLID" attribute, which passes the Employee ID of logged in user.

   Upon successful authorization, the value of *loginAttribute* is passed on to the OracleAS 10g server.

   > **Note:** To use a HeaderVar that is different from *XXX*_REMOTE_USER, you must replace *XXX*_REMOTE_USER with the desired variable in two locations: Access System Console, Authorization Rule, Actions, and in the OracleAS Java class. See "Creating the Java Class for Integration" on page 5-6 for details.

6. In the Authorization rule, allow access to Anyone.

   For example:

   ```
   Policy Manager, Create Policy Domain, Authorization Rules, Name, Allow Access,
   Any one
   ```

7. Enable the Authorization rule.

   For example:

   ```
   Policy Manager, Create Policy Domain, Authorization Rules, Name,
   ```

8. Enable the Policy Domain.

   For example:

   ```
   Policy Manager, My Policy Domains, Name, Modify, Enabled
   ```

The single sign-on configuration is now complete.

9. Test your policy domain, as described in the section on using Access Tester in the *Oracle Access Manager Access Administration Guide*.

Authorization Support for Applications Protected by OracleAS Single Sign-On

By default, the WebGate component of Oracle Access Manager intercepts all URLs, and the Access System authenticates the users who invoked the URLs. However, if you want to use OracleAS Single Sign-On to provide the authentication functionality for application login, you can configure the Oracle HTTP Server to pass authentication requests to mod_osso. This enables OracleAS Single Sign-On to continue to authenticate the user. Additionally, you can configure OracleAS Single Sign-On to pass the user's information to Oracle Access Manager for authorization.

This section describes how to implement Access System-based authorization for OracleAS Single Sign-On-protected HTTP resources.

The rest of this section discusses the following topics:

- About Authorization of OracleAS Single Sign-On-Protected Applications
- Configuring Authorization Support for OracleAS Single Sign-On-Protected Resources

About Authorization of OracleAS Single Sign-On-Protected Applications

In this type of integration, it is assumed that you have configured user authentication for various applications using OracleAS Single Sign-On. See the *Oracle Application Server Single Sign-On Administrator's Guide* for details.

After OracleAS Single Sign-On authenticates a user, Oracle Access Manager applies an external authentication scheme that looks for a REMOTE_USER header variable and maps it to an Oracle Access Manager user. If Oracle Access Manager can authenticate the user, the Access System performs user authorization. During authorization, the WebGate checks for the REMOTE_USER header variable. If it is set, the WebGate performs authorization according to polices that are defined in the Access System.

Configuring Authorization Support for OracleAS Single Sign-On-Protected Resources

This section assumes that you have installed OracleAS Single Sign-On, configured the middle tier applications to use OracleAS Single Sign-On authentication, and installed the WebGate on the middle-tier Oracle HTTP Server. See the information on configuring the middle tier in the *Oracle Application Server Single Sign-On Administrator's Guide* and the section on "Preparing Your Environment" on page 5-4 in this chapter for details.

The following procedure describes configuring OracleAS Single Sign-On authentication with Oracle Access Manager authorization.

To configure authentication using OracleAS Single Sign-On and authorization using Oracle Access Manager

1. On the computer that hosts the Oracle HTTP Server, comment following lines in the WebGate section in the file *ORACLE_HOME*/Apache/Apache/conf/httpd.conf:

```
<LocationMatch "/*">
```

```
AuthType Oblix
require valid-user
</LocationMatch>
```

2. On Linux, locate the WebGate-specific section in the httpd.conf file.

   This section is enclosed by the following lines:

   ```
   #*** BEGIN WebGate Specific ****
   #*** END WebGate Specific ****
   ```

   Move this section before the line that contains the include statement for mod_osso.conf.

3. Restart the Web server for this WebGate.

4. Protect your resources on the middle-tier Oracle HTTP Server with OracleAS Single Sign-On using static pattern rules.

   See the *Oracle Identity Management Application Developer's Guide* for details. This is required to use OracleAS Single Sign-On authentication features, for example, Windows Native Authentication.

To define an external authentication scheme in Oracle Access Manager

1. From the Oracle Access Manager landing page, click Access System Console, click Authentication Management, and click Add.

2. Define an authentication scheme similar to the following on the General tab for the authentication scheme:

   Name: External auth scheme

   Challenge Method: Ext

   Challenge Parameter: creds:REMOTE_USER

3. On the Plug-ins tab for the authentication scheme, add a credential mapping plug-in that uses the REMOTE_USER header variable, for example:

   ```
   obMappingBase="dc=us,dc=mycompany,dc=com",obMappingFilter="(&(&
   (objectclass=inetorgperson)(uid=%REMOTE_USER%))(|(!(obuseraccountcontrol=*))
   (obuseraccountcontrol=ACTIVATED)))"
   ```

   When implementing this plug-in, substitute values for obMappingBase and the person object class that are appropriate for your environment.

To define the policies to protect the middle-tier application URLs

1. From the landing page for Oracle Access Manager, click Policy Manager.

2. Click Create Policy Domain.

3. Define policies to protect any middle-tier application URL.

   Configure the polices using the external authentication scheme that you configured in the previous procedure. See the *Oracle Access Manager Access Administration Guide* for details.

4. If a WebPass and Policy Manager are installed on the same Web server as the WebGate, configure OracleAS Single Sign-On to authenticate users who try to access the Identity and Access Systems.

   Add two static URL patterns to the OracleAS Single Sign-On http.conf file:

   ```
   <LocationMatch "/identity/oblix">
   ```

```
  AuthType Basic
  require valid-user
</LocationMatch>

<LocationMatch "/access/oblix">
  AuthType Basic
  require valid-user
</LocationMatch>
```

These rules enable OracleAS Single Sign-On to perform authentication for the Identity System and Policy Manager.

5. Also, if a WebPass and Policy Manager are installed on the same Web server as the WebGate, ensure that the external authentication scheme that you configured in the previous procedure is protecting the Identity and Access domains.

   See the *Oracle Access Manager Access Administration Guide* for details.

To configure logout for the integration

1. See"Implementing Global Logout from OracleAS Single Sign-On and Access Server" on page 5-9 for details.

Testing the Integration with OracleAS

After you set up OracleAS and Oracle Access Manager for integration, test to ensure that the integration is successful.

To test Oracle Access Manager single sign-on for OracleAS

1. For OracleAS 10.1.4.0.1 and higher, enter the following URL in the browser:

   `http://computername:port/sso/`

   Where *computername* is the computer where the OracleAS Server is installed and *port* is the port number of the computer.

   For OracleAS 10.1.2, enter the following URL in the browser:

   `http://computername:port/pls/orasso/`

   You should be presented with a login page. After you have successfully authenticated, the OracleAS Web resource page appears.

2. You can try to access various applications as the same user.

   If Oracle Access Manager single sign-on is successful, you are allowed access to the page without being challenged for authentication.

3. You an also try to test different authorization rules in the Access System.

   For example, if there are time conditions set for login, you may try logging in at different times.

4. When you are ready to log out, click the Logout link.

   If Oracle Access Manager single sign-on is successful, you are logged out of all Oracle Access Manager-protected resources.

OracleAS 10g Files

The following two sample files can be customized to meet your requirements:

- SSOOblixAuth.java

- Logout.jsp

SSOOblixAuth.java

```
package oblix.security.ssoplugin;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import oracle.security.sso.ias904.toolkit.IPASAuthInterface;
import oracle.security.sso.ias904.toolkit.IPASAuthException;
import oracle.security.sso.ias904.toolkit.IPASUserInfo;
import oracle.security.sso.ias904.toolkit.IPASInsufficientCredException;
import java.net.URL;
import java.util.*;

public class SSOOblixAuth implements IPASAuthInterface
{
    private static String OBLIX_USER_HEADER = "XXX_REMOTE_USER";
    private static String CLASS_NAME = "SSOOblixAuth";

    public SSOOblixAuth()
    {
      System.out.println("Inside SSOOblixAuth constructor.....");
    }
    public IPASUserInfo authenticate(HttpServletRequest request)
      throws IPASAuthException, IPASInsufficientCredException {


      String OblixUserName = null;

      try
       {
         System.out.println(".............Getting Header Variable.............");
         OblixUserName = request.getHeader(OBLIX_USER_HEADER);

         System.out.println("The Header name............."+OblixUserName);
       }
       catch (Exception e)
       {
         throw new IPASInsufficientCredException("No Oblix Header");
       }

       if (OblixUserName == null)
           throw new IPASInsufficientCredException("No Oblix Header");

       IPASUserInfo authUser = new IPASUserInfo(OblixUserName);
       System.out.println("The IPASUserInfo Class............."+authUser);
       return authUser;

    }


    public URL getUserCredentialPage(HttpServletRequest request,String msg) {

System.out.println("Inside Get User Crediential Page ..........Should not come
here>......");

        URL errorURL=null;
        try
        {
```

```
            errorURL=new URL(new String(request.getRequestURL()));
        }
        catch(Exception ee){};
        return  errorURL;
    }


}
```

Logout.jsp

You can use the following sample file as discussed in "Implementing Global Logout from OracleAS Single Sign-On and Access Server" on page 5-9.

```
<!-- Copyright (c) 1999, 2003, Oracle. All rights reserved. -->
<%@page autoFlush="true" session="false"%>
<%
// Declare English Message Strings
String msg1 = "Single Sign-Off";
String msg2 = "Application Name";
String msg3 = "Logout Status";
String msg4 = "ERROR: The return URL value not found.";
String msg5 = "ERROR: Logout URL for partner applications not found.";
// Get the user language preference
String userLocaleParam = null;
java.util.Locale myLocale = null;
// Get the user locale preference sent by the SSO server
try
{
userLocaleParam = request.getParameterValues("locale")[0];
}
catch(Exception e)
{
userLocaleParam = null;
}
if( (userLocaleParam == null) || userLocaleParam.equals("") )
{
myLocale = request.getLocale();
}
else
{
if(userLocaleParam.indexOf("-") > 0 )
{
// SSO server sent the language and territory value (e.g. en-us)
myLocale = new java.util.Locale(userLocaleParam.substring(0, 2),
userLocaleParam.substring(3, 5));
}
else
{
// SSO server sent only the language value (e.g. en)
myLocale = new java.util.Locale(userLocaleParam, "");
}
}
// The following two lines will be used only for the Multilingual support with
// proper resource bundle class supplied
// java.util.ResourceBundle myMsgBundle
// = java.util.ResourceBundle.getBundle("MyMsgBundleClassName", myLocale);
// Get the message string in the appropriate language using the message key.
// Use this string to display the message in this page.
// String mesg = myMsgBundle.getString("mesg_key");
```

```
%>
<html>
<body bgcolor="#FFFFFF">
<h1><%=msg1%></h1>
<%
String done_url = null;
int i = 0;
// Get the return URL value
try
{
done_url = request.getParameterValues("p_done_url")[0];
}
catch(Exception e)
{
done_url = "";
}
// Get the application name and logout URL for each partner application
try
{
%>
<b> <%=msg2%>   <%=msg3%> </b>
<br>
// Substitute an actual host, domain, and port for myhost.us.mydomain.com:7777
// that points to the WebGate.
<img
src="http://myhost.us.mydomain.com:7777/access/oblix/lang/en-us/style2/oblixlogo.g
if">
<%
for(;;)
{
i++;
String app_name = request.getParameterValues("p_app_name"+i)[0];
String url_name = request.getParameterValues("p_app_logout_url"+i)[0];
%>
<%=app_name%>
 
<img src="<%=url_name%>">
<br>
<%
}
}
catch(Exception e)
{
if(done_url == null)
{
%>
<%=msg4%> <br>
<%
}
if(i>1)
{
%>
<br> <a href="<%=done_url%>">Return</a>
<%
}
else
{
%>
<%=msg5%><br>
<%
```

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

Troubleshooting the OracleAS 10g Integration

The following are troubleshooting tips for the Oracle 10g integration.

Problem: With a form-based authentication scheme, while accessing OIDDAS/Form application externally deployed J2EE applications, the OracleAS single sign-on login page is displayed after the Oracle Access Manager Form login page.

Solution: This happens if mod_osso uses a POST based redirection method instead of GET to call the single sign-on server. The redirection method used is based on value of OssoRedirectByForm directive. To use GET method, this directive must be set to false. In Oracle 10g Application Server, this value is set to false by default.

To verify that this directive is set to false

1. Verify the value of OssoRedirectByForm directive.
2. Launch the Oracle Enterprise Manager.
3. Select the Oracle Application Server instance where the Oracle Infrastructure is installed.
4. Select the HTTP Server where WebGate is installed and navigate to Advanced Server Properties.
5. From the list of configured files, select the mod_osso.conf file.
6. Check if OssoRedirectByForm is set to true.

   By default the values is false.
7. If the default directive value is not used, set it to false as shown in the following example:

```
<IfModule mod_osso.c>
OssoIpCheck off
OssoIdleTimeout off
OssoConfigFile
/private1/iasinst/install_set1/904infra/Apache/Apache/conf/osso/osso.conf
OssoRedirectByForm off
</IfModule>
```

8. Click Apply.
9. Restart the OracleAS HTTP Server.

Problem: How do I find ORASSO and Portal schema passwords?

Solution: Complete the following procedure.

To find these database schema passwords

1. Login to Oracle Directory Manager as the super user orcladmin.
2. Expand the tree on the left hand side, as follows:

   Cn= OracleContext

   Cn=Products

Cn=IAS

Cn=IAS Infrastructure Databases

OrclReferenceName=<global database name>

OrclResourceName=ORASSO

3. Click the ORASSO entry and look for the value for attribute orclpasswordattribute (the Password for ORASSO schema).

> **Note:** Similarly you can click the OrclResourceName=PORTAL for the portal schema password.

Problem: How do I check the single sign-on logs?

Solution: You can view the single sign-on logs from Enterprise Manager.

1. Log in to Enterprise Manager.
2. Click the Logs link at the bottom of the page.

   A search screen appears.
3. From the Available Components list select Single Sign-on:orasso and move it to the Selected Components.
4. Perform the search to view the single sign-on logs.

Problem: How do I create a default RAD?

Solution: Complete the following steps to create a default RAD:

To create a default RAD

1. Access Oracle Internet Directory Delegated Administration Services Console, Configuration, Preference, as usual.
2. Scroll to the bottom of the page to display Resource Access Information.
3. Click Create to create a new resource file.
4. Enter a Resource Name:

   For example, for a default configuration you can use:

   ```
   default
   ```

> **Note:** Resource name created over here should be the same as the configuration present in formsweb.cfg file.

5. Click Next, fill in the user ID and password and the connect string for the database, and click Submit.

   The user ID is a valid DB user. Database refers to the DB used. For example, if a schema named "Scott" is used and a Database "asdb"', the test entries are:

   Username: *scott*

   Password: *tiger*

   Database: *asdb*

Problem: How do I create a user-specific RAD?

Solution: Complete the following steps to create a user-specific RAD:

To create a user-specific RAD

1. Access the Oracle Internet Directory Delegated Administration Services console, as usual.
2. Select the Directory tab found at the top right hand corner of the page.
3. Click Create to create a new user.
4. Select a user name, for example, ssotest with a password of ssotest1.

   You can choose to add all other details.
5. Scroll to the bottom of the page to Resource Access Information.
6. Click Create to create a new resource file.
7. Enter a Resource Name, for example, ssotest_db.
8. Click Next, fill in the user ID, password, and connect string for the database, then click Submit.

The user ID here is a valid DB user. For testing purposes, the default Scott schema can be used. Database is the DB used, with a default value of asdb. For example, the test entries could be:

Username: scott

Password: tiger

Database: asdb

Index

S

T

U

V