Oracle OID, OHS, WebGate, and OAM

How Oracle Access Management Delivers Single Sign-On

Component Roles, Integration, and the End-to-End SSO Login Flow

https://digitalksystems.com, digitalk.fmw@gmail.com

IDM Administration
DT246055-04

Aug, 2026

Oracle Fusion Middleware: Oracle Identity Management Administering
DT246055-04

Primary Author: DigiTalk Systems

This document is being provided by DigiTalk Systems as part of its effort to assist users in understanding and working with Oracle Identity Management (Oracle IDM) technologies, including products and components such as Oracle Internet Directory (OID), Oracle Unified Directory (OUD), Oracle Access Manager (OAM), Oracle Identity Governance (OIG), and related Oracle Identity Management components. DigiTalk Systems wishes to emphasize that this document is not affiliated with, endorsed by, certified by, or sponsored by Oracle Corporation ("Oracle") in any way. The content contained herein is based primarily on publicly available Oracle product documentation, technical resources, product behavior, and generally accepted administration and configuration practices.

While every effort has been made to ensure the accuracy and reliability of the information presented in this document, there is a possibility of typographical errors, configuration differences, product-version differences, or other inaccuracies. DigiTalk Systems does not guarantee the correctness, completeness, or suitability of the content provided in this document.

Users of this document are encouraged to cross-reference the information presented here with Oracle's official Oracle Identity Management documentation and other authoritative Oracle resources. Product behavior, commands, configuration procedures, supported features, and administrative interfaces may vary depending on the Oracle Identity Management product, version, operating system, Java version, database version, and overall environment configuration.

We would also like to clarify that any code snippets, configuration examples, commands, scripts, sample applications, or other examples included in this document are intended primarily for educational and learning purposes. Such materials may be created by DigiTalk Systems or adapted from publicly available Oracle documentation, examples, demonstrations, or other publicly available technical resources, subject to their respective copyright and licensing terms. Any Oracle-provided sample code, examples, product documentation excerpts, trademarks, product names, logos, copyrights, and other intellectual property remain the property of Oracle Corporation and/or its affiliates. DigiTalk Systems does not claim ownership of Oracle's copyrighted or proprietary materials. Our charges are solely for the time, effort, expertise, practical exercises, lab environments, explanations, documentation preparation, and other resources invested in creating these learning materials. DigiTalk Systems does not charge for or claim ownership of Oracle's copyrighted material or intellectual property.

By using or accessing this document, you acknowledge and agree that:

- This document is not officially endorsed, verified, certified, or supported by Oracle Corporation.
- Oracle Identity Management and the associated Oracle products referenced in this document are Oracle products and remain subject to Oracle's applicable licensing and intellectual-property terms.
- DigiTalk Systems makes no claims or guarantees regarding the accuracy, completeness, or suitability of the information contained in this document.
- Users are responsible for independently verifying and validating any information presented here before applying it to their specific environment or use case.
- Configuration procedures, commands, interfaces, supported features, and product behavior may differ across Oracle Identity Management product versions and deployment architectures.
- The examples and lab exercises provided in this document are intended for learning, demonstration, and training purposes and should be appropriately reviewed and tested before being implemented in production environments.
- DigiTalk Systems disclaims any liability for errors, omissions, configuration issues, or any direct, indirect, incidental, or consequential loss or damage that may result from the use of this document.

If you discover any inaccuracies or errors in this document, please report them to digitalk.fmw@gmail.com, and DigiTalk Systems will endeavor to review and correct them as appropriate in future revisions.

This disclaimer is provided to ensure transparency regarding the source, purpose, ownership, and limitations of the information contained in this document.

By using or accessing this document, you acknowledge that you have read, understood, and agreed to the terms and conditions outlined above.

Table of Contents

1. Introduction

1.1 The Problem: One Login Per Application Doesn't Scale

A typical enterprise runs dozens or hundreds of internal web applications - HR portals, procurement systems, custom line-of-business apps, dashboards. Without a shared authentication layer, each application would need to implement its own login screen, manage its own password store, and independently enforce who is allowed to see what. This is both a poor user experience (a different login for every application) and a security liability (credentials and access logic duplicated and re-implemented, inconsistently, across every application team).

Oracle's Access Management stack solves this by centralizing authentication and authorization into a single, shared layer that sits in front of every protected application, rather than inside each one. This document explains the four components that make this possible - OID, OHS, WebGate, and OAM - how each one specifically contributes, and walks through the complete login sequence step by step.

1.2 The Four Components at a Glance

Component	Category	One-Line Role
OID (Oracle Internet Directory)	Identity store (LDAP)	Holds users, groups, and credentials that authentication is checked against
OHS (Oracle HTTP Server)	Web tier / reverse proxy	The web server that receives browser traffic and proxies it to backend applications
WebGate	Policy enforcement point (PEP)	A plug-in inside OHS that intercepts every request and asks OAM whether it should be allowed
OAM (Oracle Access Manager)	Policy decision point (PDP)	The central server that evaluates authentication and authorization policies and manages SSO sessions

Note: A common point of confusion: OHS and WebGate are not the same thing. OHS is the web server itself (based on Apache HTTP Server); WebGate is a plug-in module installed into OHS. OHS could run perfectly well without WebGate - it just wouldn't enforce any OAM policy if it did.

2. How the Components Fit Together

The clearest way to understand this stack is to trace a single request through it, end to end. Figure 1 shows the full architecture: a browser request enters through OHS/WebGate, policy decisions are delegated to the OAM Server, identity data is looked up in OID, and once approved the request is proxied on to the actual protected application.

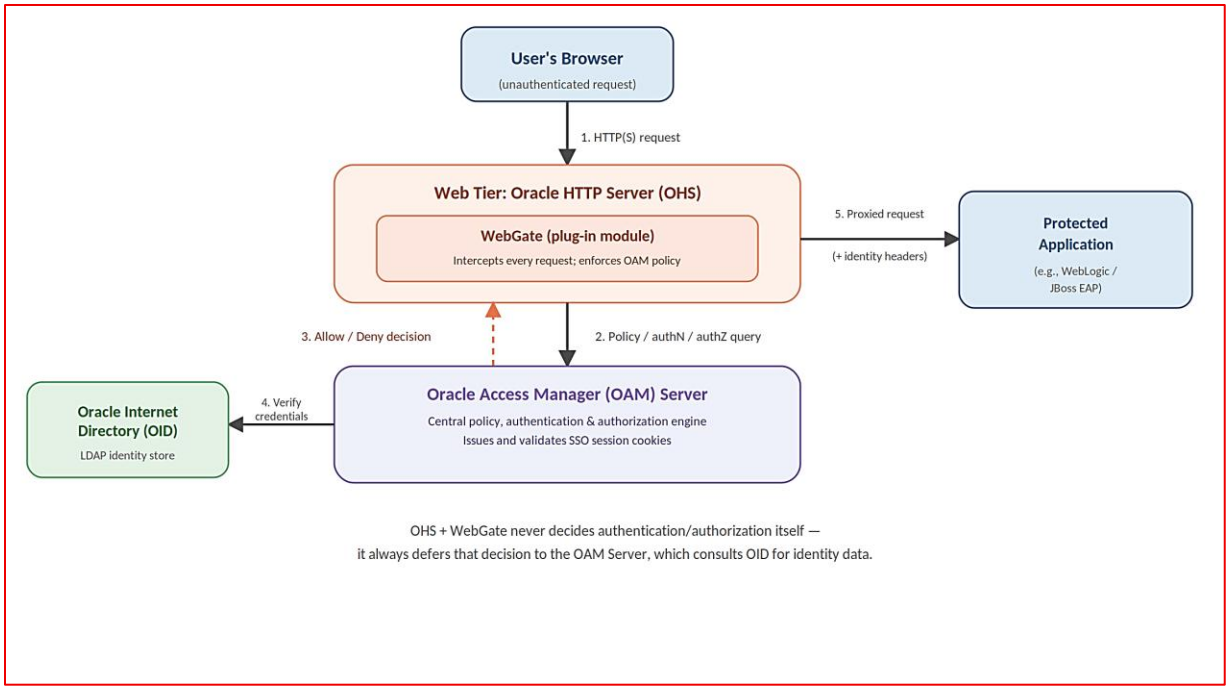

Figure 1 — End-to-end architecture: browser → OHS/WebGate → OAM Server → OID, with the approved request finally reaching the protected application.

2.1 The Policy Enforcement Point vs. the Policy Decision Point

This architecture follows a standard access-management pattern worth naming explicitly, since it clarifies exactly why four separate components are involved rather than one: the split between where a decision is enforced and where it is made.

Concept	Who Plays This Role	What It Means
PEP - Policy Enforcement Point	WebGate (inside OHS)	The component sitting in the traffic path that actually blocks or allows a request — but never decides on its own
PDP — Policy Decision Point	OAM Server	The component that evaluates policy and identity, and tells the PEP what decision to enforce
Identity Store	OID (or OUD/AD)	Where the PDP looks up who a user is and what groups/roles they belong to

This separation is deliberate and valuable: WebGate instances can be deployed at every entry point across the enterprise (in front of dozens of different OHS servers, each fronting different applications), while all of them share and enforce the exact same centrally managed policies from one OAM Server change a policy once, and every enforcement point picks it up immediately, with no per-application redeployment needed.

2.2 Why OHS Sits in Front of the Application, Not OAM Itself

As visible in Figure 1: the browser never talks to the OAM Server directly for normal application traffic (it does redirect there specifically during login, covered in Section 3). Ordinary application requests flow browser → OHS → application, with WebGate (inside OHS) making a side-channel call to OAM to ask "should this be allowed?" OHS also commonly serves a second, unrelated purpose here acting as a reverse proxy that forwards approved requests on to the real application server (WebLogic, JBoss EAP, etc.) using a module such as mod_wl_ohs, entirely independent of the WebGate/OAM policy check.

2.3 Why OID Specifically (and What Can Replace It)

OID is Oracle's LDAP-compliant directory server, and in many Oracle Access Management deployments it serves as the identity store OAM consults for authentication and group/role lookups. It's worth noting that OID is not the only option: OAM can just as validly be configured against Oracle Unified Directory (OUD), Microsoft Active Directory, or another LDAP-compliant directory. OID is simply the traditional, commonly paired choice in classic Oracle Fusion Middleware deployments. Regardless of which directory is used, its role in this architecture is the same: it is where user records, passwords (or password verifiers), and group memberships actually live.

3. The End-to-End SSO Login Flow

This section walks through exactly what happens, in order, from the moment a user requests a protected resource to the moment they see the application based on the standard OAM Agent (WebGate) flow with the Embedded Credential Collector, Oracle's default login-handling configuration.

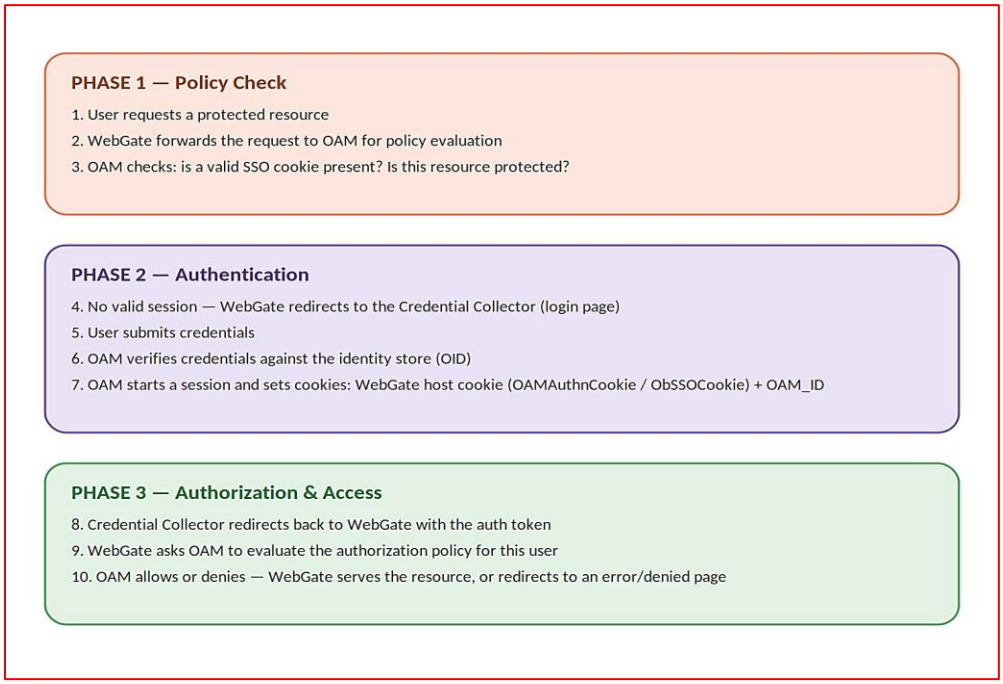

Figure 2 — The complete SSO login sequence, grouped into three phases: policy check, authentication, and authorization/access.

3.1 Phase 1 - Policy Check

When the user's browser requests a resource, WebGate does not decide anything on its own, it immediately forwards the request to the OAM Server for a policy evaluation. OAM checks two things: whether the incoming request already carries a valid SSO session cookie, and whether the requested resource is actually configured as protected in the first place. Unprotected resources are simply served immediately with no further involvement from OAM.

3.2 Phase 2 - Authentication

If the resource is protected and no valid session cookie is present, WebGate redirects the browser to the Credential Collector, the component that actually serves the login form, based on whatever authentication policy applies to this resource (a simple username/password form, for instance, though OAM supports many other authentication schemes and can require stronger methods for more sensitive resources). Once the user submits credentials, OAM verifies them against the configured identity store (OID, per Section 2.3), and on success starts a session, issuing two categories of cookie: an agent-specific cookie tied to this WebGate (OAMAuthnCookie 11g onwards, ObSSOCookie in 10g), and a server-side session identifier, OAM_ID, that represents the user's overall authenticated session with OAM itself not tied to any one application.

3.3 Phase 3 - Authorization and Access

With authentication complete, the Credential Collector redirects the browser back to WebGate, which now asks OAM a second, distinct question: not “who is this user” but “is this specific, now-identified user allowed to access this specific resource?” OAM evaluates the authorization policy which can reference user attributes, group membership, or other conditions and returns an allow or deny decision. WebGate enforces whatever OAM decides: serving the resource on an allow, or redirecting to an administrator-defined error/access-denied page on a deny.

> ***Note: Authentication answers “who are you,” authorization answers “what are you allowed to do.” OAM evaluates these as two separate, sequential policy checks - which is exactly why it's possible for a fully authenticated, legitimate user to still be correctly denied access to a specific resource they simply aren't authorized for.***

3.4 Session Cookies Used Throughout the Flow

Cookie	Set By	Purpose
OAMAuthnCookie_<host:port> (11g onwards WebGate) / ObSSOCookie (10g WebGate)	OAM, via the WebGate	The per-agent SSO token proving this browser has an authenticated session with this WebGate
OAM_ID	OAM Server	The server-side session identifier correlating the user's overall OAM session across every WebGate/application in the domain
OAM_REQ	WebGate (before authentication)	Temporarily tracks the originally requested URL, so the user lands back on the right page after login

Practical tip: If SSO appears to be failing intermittently only for certain users or browsers, checking whether these cookies are actually being set and sent (via browser dev tools) is usually the fastest way to narrow down whether the problem is on the WebGate/OHS side, the OAM Server side, or a browser-level cookie policy (e.g., third-party cookie blocking) issue.

4. Single Sign-On Across Multiple Applications

The real payoff of this architecture becomes visible once a second, separately protected application enters the picture. Because the OAM_ID cookie represents the user's session with OAM itself not with any one application a user who has already authenticated once can access a second protected application, fronted by a completely different WebGate, without being prompted to log in again.

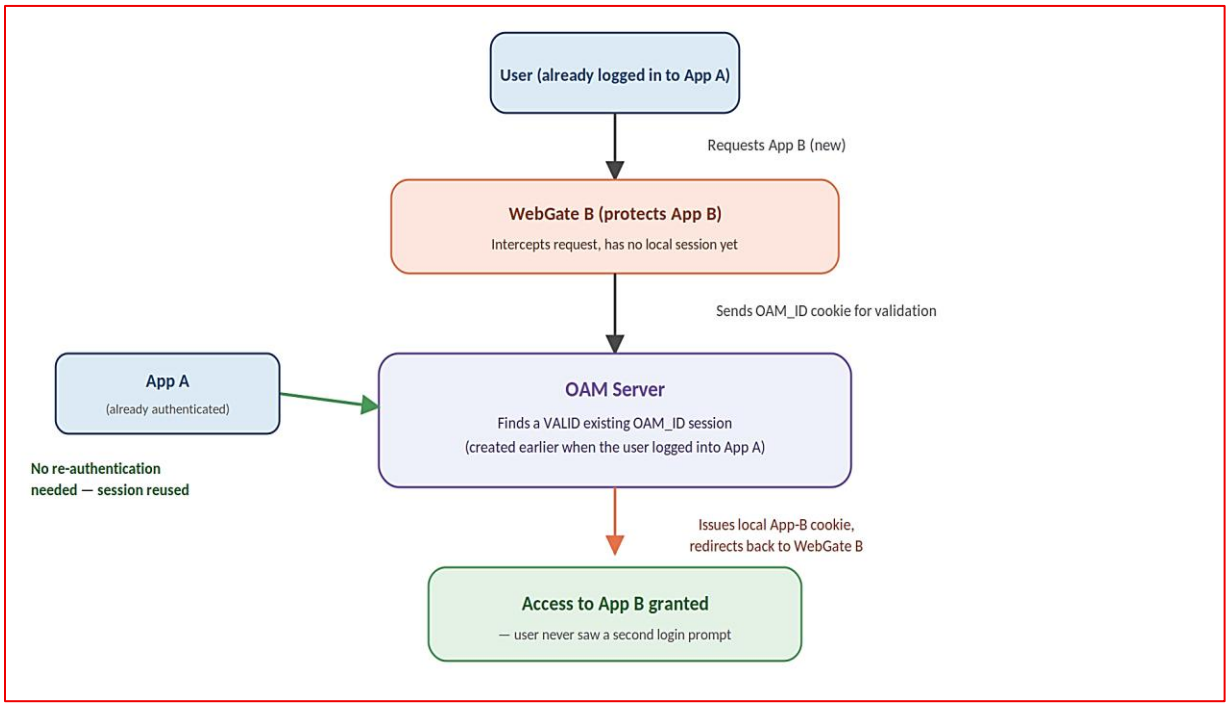

Figure 3 — A second application's WebGate validates the user's existing OAM_ID session instead of forcing a fresh login.

This is the concrete mechanism behind “single sign-on” as a user-facing experience: WebGate B, protecting a different application entirely, still performs the same Phase 1 policy check from Section 3.1 — but this time, OAM finds a valid existing OAM_ID session already associated with this browser, so Phase 2 (Section 3.2) is skipped entirely. OAM simply issues a new, local agent cookie for WebGate B and redirects straight back, landing the user on Application B having never seen a second login form.

Note: This is also precisely why logging out should always be done through OAM's central logout endpoint rather than by clearing one application's local cookie - clearing only the WebGate-B-specific cookie would leave the underlying OAM_ID session (and therefore access to every other protected application) still fully valid.

5. How an Application Gets Protected in OAM

Sections 1–4 described how the runtime login flow works once everything is configured. This section covers the administrative side: what actually has to be set up in OAM before any of that flow can happen for a given application.

5.1 Application Domains

In OAM, resources belonging to a given application are organized under an Application Domain a named container holding that application's resource definitions, authentication policies, and authorization policies together. Grouping by Application Domain is what allows an administrator to manage one application's access rules independently of every other application registered in the same OAM instance.

5.2 Resources, Authentication Policies, and Authorization Policies

- **Resources** - URL patterns that identify which parts of an application are covered (e.g.,
- /myapp/** to cover an entire application, or a narrower pattern to protect only specific paths).
- **Authentication Policy** - defines which authentication scheme applies to a set of resources (form-based login, basic authentication, certificate-based, multi-factor, etc.) and which Application Domain resources it covers.
- **Authorization Policy** - defines who is actually allowed in, once authenticated — referencing specific users, groups (looked up from OID), or additional conditions such as time-of-day or client IP range.

5.3 WebGate Registration

Before any of the above takes effect for real traffic, the OHS instance sitting in front of the application must have a registered, configured WebGate agent, this is the piece that physically connects a specific web server to the OAM Server and Application Domain policies. Registration can be done manually through the OAM Console's Application Security → SSO Agent Registration screens, or via the Quick Start Wizard for a faster, guided setup; either produces a configuration package that is then deployed onto the actual OHS host.

5.4 Step-by-Step: Protecting a New Application

Step	Task	Where It's Done
1	Register/associate the application's URL prefix with an Application Domain in OAM	OAM Console - Application Security
2	Define Resources (URL patterns) that belong to this application	OAM Console - Application Domain → Resources
3	Create an Authentication Policy - which authentication scheme (form login, basic auth, etc.) applies to these resources	OAM Console - Authentication Policies

4	Create an Authorization Policy - which users/groups are allowed, and any conditions (time, IP, risk)	OAM Console - Authorization Policies
5	Register a WebGate agent for the OHS instance fronting the application (if not already registered)	OAM Console - SSO Agent Registration, or the Quick Start Wizard
6	Deploy the WebGate agent's generated configuration to the OHS server and restart OHS	OHS host - webgate/config directory
7	Verify: access the application's URL and confirm redirection to the OAM login page	Browser test

Practical tip: A very common real-world mistake is completing Steps 1–4 (all Application Domain/policy configuration, done centrally in the OAM Console) but forgetting Step 6 (deploying the WebGate configuration to the actual OHS host and restarting it) - the policies exist correctly in OAM, but the specific OHS server fronting the application has no way to know about them until its local WebGate configuration is actually updated and the server restarted.

6. Summary

Concept	Key Takeaway
OID	The identity store - holds users, groups, and credentials that OAM authenticates against
OHS	The web server / reverse proxy that sits in front of applications and receives browser traffic
WebGate	A plug-in inside OHS that intercepts every request and enforces - but never independently decides - OAM's policy
OAM	The central policy engine: makes every authentication and authorization decision, and manages SSO sessions
PEP vs. PDP	WebGate enforces (PEP); OAM Server decides (PDP) - this split is what lets one policy set cover many enforcement points
OAM_ID cookie	Represents the user's session with OAM itself, not any one application - the actual mechanism behind cross-application SSO
Application Domain	Where an application's resources, authentication policy, and authorization policy are grouped and managed together
Protecting an app	Requires both OAM-side policy configuration AND WebGate deployment/restart on the actual OHS host - both sides are necessary

Together, these four components implement a clean separation of concerns: OID answers “who exists and what do they belong to,” OAM answers “should this be allowed,” and OHS/WebGate is the enforcement point standing in the traffic path that carries out whatever OAM decides giving every protected application in the enterprise consistent, centrally managed authentication and authorization, and giving every user a single login that follows them across every application behind this shared layer.