

# Oracle OAuth Service

ORACLE WHITE PAPER | MARCH 2015



## Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Introduction	4
OAuth 2.0 Overview	4
OAM OAuth 2.0 Service – Features	5
Securing Mobile Clients	8
Additional Differentiations	11
Sample Use Case	12
Conclusion	15

## Introduction

The OAuth 2.0 Service in Oracle Access Management 11g R2PS2 provides organizations with a standards-based solution that allows their users to securely share or access resources with partners / SaaS providers.

The OAuth protocol was originally designed and has achieved wide acceptance in that regard as the de-facto standard that enables users to grant third-party access to their resources without sharing their passwords and while also providing a means to grant limited/scoped access to those resources. Since this initial capability, OAuth has also quickly become the preferred method to provide tokenized authentication and access control between any type of client (including mobile devices and other services) to a service.

The Oracle Access Manager (OAM) OAuth 2.0 service is a standards compliant OAuth 2.0 authorization service implementation that supports the following roles defined by OAuth:

- **Resource Server:** The server hosting the protected resources, capable of accepting and responding to resource requests using access tokens.
- **Client:** An application making protected resource requests on behalf of the resource owner and with its authorization. The term client is not specific to a particular entity, for example the client could be an application that executes on a server or mobile device.
- **Authorization Server:** The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

Besides providing new or existing Oracle IAM on-premise customers with a standards-based solution to securely share or access resources with partners/SaaS providers, following are some other compelling drivers addressed by the OAM OAuth 2.0 service:

- Provide a standards compliant token based authentication service that can be leveraged by third-party services including Oracle Public Cloud hosted applications and services.
- Support identity propagation use cases where application clients are required to impersonate end users and standards based tokens are the natural choice.
- Provide standards compliant authentication and authorization for RESTful Identity Services for the OAM Mobile & Social Service that offers a rich set of access management features as a service model to different types of client applications such as mobile applications, SaaS providers and Rich Internet Applications.

## OAuth 2.0 Overview

In the traditional client-server authentication model, the client accesses a protected resource on the server by authenticating with the server using the resource owner's credentials. In order to provide third-party applications access to protected resources, the resource owner shares their credentials with the third-party. This creates several problems and limitations:

- Third-party applications are required to store the resource-owner's credentials for future use typically a password in clear-text.
- Servers are required to support password authentication despite the security weaknesses created by passwords.
- Third-party applications gain overly broad access to the resource-owner's protected resources leaving resource owners without any ability to restrict duration or access to a limited subset of resources.
- Resource owners cannot revoke access to an individual third-party without revoking access to all third parties and must do so by changing their password.

OAuth 2.0 addresses these issues by introducing an authorization layer and separating the role of the client from that of the resource owner. In OAuth 2.0, the client requests access to resources controlled by the resource owner and hosted by the resource server and is issued a different set of credentials than those of the resource owner. Instead of using the resource owner's credentials to access protected resources, the client obtains an access token - a string denoting a specific scope, duration, and other access attributes. An authorization server with the approval of the resource owner issues access tokens to third-party clients. The client uses the access token to access the protected resources hosted by the resource server.

The scenario sketched out above is commonly referred to as a 3-legged OAuth flow as it involves a resource owner approving a request for resources however the 2-legged OAuth flow i.e. a scenario that does not directly involve a resource owner and includes an interaction only between a client and a resource server is also gaining traction in the marketplace for service-to-service communication. This use case involves an OAuth Client that has essentially been preapproved to access resources. This model fits the service-to-service model, especially when the requesting and resource services are connected by organization or close partnership and therefore resource owner approval is either assumed or not required.

The OAM OAuth 2.0 service is built to support both 3-legged and 2-legged OAuth 2.0 flows as an OAuth 2.0 authorization server and also offers several compelling differentiations to enable the OAuth 2.0 Client and the OAuth 2.0 Resource Server roles.

## OAM OAuth 2.0 Service – Features

### Authorization Service

The OAM OAuth 2.0 authorization service endpoint is used to interact with the resource owner and obtain an authorization grant. The OAM OAuth 2.0 authorization service endpoint needs to verify the identity of the resource owner before issuing an authorization grant. Since resource owner identity verification is outside the scope of the OAuth 2.0 specification, OAM OAuth 2.0 uniquely leverages its built-in integration with OAM and provides the ability to use any of the OAM

authentication schemes for user authentication as well as uses OAM for user session and cookie management.

#### Token Service and Supported Grant types

When an OAuth Client makes a POST request to the OAM OAuth 2.0 token service endpoint, the service validates the client credentials if the client type is confidential or was issued client credentials, validates the requested scope based on configuration as well as user consent and returns an access token for the following grant types:

- Authorization Code
- Resource Owner Credentials
- Client Credentials
- Extension Grant type to support JWT tokens

Support for these Grant types by the token service allows the OAM OAuth 2.0 service to fully support the following OAuth 2.0 flows for *confidential* clients:

1. **Authorization Code Flow:** The Authorization Code Flow is suitable for OAuth clients that can interact with the resource owner's user-agent (typically a browser), and that can receive incoming requests from the OAuth 2.0 authorization server i.e. this flow is suitable for web server based OAuth clients. In this flow after the resource owner approves access, the Web server OAuth client receives a callback with an authorization code and passes back the authorization code to the OAuth 2.0 authorization server to obtain an access token response.
2. **Resource Owner Password Credentials Flow:** This flow also called the username-password authentication flow as the client token request to obtain an access token is sent in an HTTP POST to the OAM OAuth token service endpoint and contains the resource owner's username and password. It uses a grant type that is suitable for clients capable of obtaining the Resource Owner's credentials (username and password, typically using an interactive form) and where the Resource Owner has a trust relationship with the client (for example, the device operating system or a highly privileged application)
3. **Client Credentials Grant Flow:** This flow is also called the 2-legged OAuth flow as the client requests an access token using only its client credentials (or other supported means of authentication). The client can also request access to resources of another Resource Owner that has been previously arranged with the Authorization Server (the method of which is beyond the scope of the specification).

#### Client Registration

The OAM OAuth 2.0 Service allows an administrator to use the OAM console to register OAuth Clients with redirection URIs. The OAuth Service issues a client identifier for each client – a unique string that represents registration information provided for the OAuth client.

#### Scope Management

The OAM OAuth 2.0 Service provides the ability to specify specific scope definitions for a resource server and enforces scope checking when handling both authorization service endpoint requests and token service endpoint requests. It also uniquely allows a seamless mapping of OAM protected resources to OAuth 2.0 resource scopes.

#### User Authentication and Consent Management

For user authentication, the OAM OAuth 2.0 service uniquely supports all authentication schemes provided by OAM and also leverages OAM for user session/cookie management and protection of the consent page.

#### Refresh Access Token

The OAM OAuth 2.0 service returns a refresh token together with an access token in the token response where applicable. When an OAuth Client makes a refresh request to the token endpoint with a valid refresh token, the OAM OAuth 2.0 service performs client authentication for confidential clients or for any client that was issued client credentials, validates the refresh token and issues the requested token to the client.

#### Administration

The OAM OAuth 2.0 Service provides administrators with comprehensive and easy configuration options to configure several aspects of the service using the OAM Console including:

- OAuth Authorization/Token Service Profiles
- OAuth Access Service Providers
- OAuth Client Profiles
- Resource Servers configuration

#### Mobile OAuth

Since mobile clients (i.e. native apps on mobile devices) are public/non-confidential clients, the OAM OAuth2.0 service requires that mobile applications need to be first registered with OAM to use the service. The OAM OAuth 2.0 Mobile OAuth flow uniquely couples mobile application registration and device identification with a traditional 3-legged OAuth flow to provide trusted access for mobile clients.

## Securing Mobile Clients

Several OAuth Clients are consumer applications that cannot keep the client secret confidential (application password or private key). These OAuth Clients are called public clients or non-confidential clients. Mobile Client applications i.e. native applications on mobile devices are also categorized as public clients because when a native application is first downloaded from an app store to a device it has the client credentials that uniquely identify the client application baked into the application. Since all users that download the native application have access to the binary, a malicious user could easily decompile the client credentials out of the binary and insert their own credentials. During an OAuth flow when the access code gets exchanged for the access token this leads to a major vulnerability as there is now no secure means of really identifying who is actually receiving and using the access token. Hence, providing a mechanism to secure the mobile application on the device in order to ensure trusted access is a key requirement specifically for enterprise mobile applications that routinely require access to sensitive data. Following sections describe key innovations in the OAM OAuth 2.0 Service's Mobile OAuth flow to facilitate secure access for mobile OAuth Clients:

### Mobile Application Registration and Mobile Device Identification

The OAM OAuth 2.0 Service provides built-in support that provides a mechanism for mobile applications to be first registered with OAM to use OAuth Access Services. Security is enforced for mobile applications by creating a client profile and issuing client tokens specific to the application on a device. The user provides explicit authorization to register each mobile OAuth application. In the OAM OAuth 2.0 Service, mobile application registration is modeled uses the same paradigm as the OAuth protocol flow where mobile application registration is modeled as a scope for which the user needs to provide explicit consent. As a result of the mobile application registration the application is issued a client token. The mobile application always submits this client token as an input parameter for accessing OAM OAuth 2.0 Service end points. Furthermore, the OAM OAuth 2.0 service also allows easy coupling of device identification with mobile application registration where mobile devices and applications can be checked against fraud and security using a built-in integration with Oracle Adaptive Access Manager (OAAM). The mobile application passes device claims along with the client token for accessing OAM OAuth 2.0 Service end points and the OAM OAuth2.0 Service provides a configuration to make certain claims as mandatory and certain claims as optional.

### Secure token delivery

Modern mobile operating systems provide a notification service mechanism that is meant to notify the application with some data and the receipt of the data confirms that the application is a valid application. The native mobile OS application notification mechanisms provide an extra level of assurance for confirming application identity. Unfortunately, this level of assurance varies on different mobile OS platforms. Apple Push Notification Service (APNS) currently provides better security characteristics than Google Cloud Messaging (GCM) as it can ensure that the receiver of the message is the proper client application authenticated with proper private certificates to decrypt



the message as well as that the application runs on a real device (instead of an emulator). However, while GCM can guarantee that the messages are only routed to the same application on the same device which initiates the OAuth protocol request, it cannot ensure the message receiver runs on a real physical device and it cannot properly ensure the client application is the authentic one i.e. in GCM a malicious application can fake in the real app's sender ID and server registration interface –allowing the malicious application to receive notifications meant for the real application.

The OAM OAuth 2.0 Service provides support to leverage native mobile OS notification service mechanisms (APNS/GCM) as a level of assurance to confirm mobile application identity and then augments these mobile OS notification services with configuration options to use standard, hybrid and advanced modes that allow varying levels of security to deliver tokens/codes based on a combination of HTTP(S) and push mechanisms during client registration and token/code requests in the Mobile OAuth flow. It also provides a splitting logic mechanism to the authorization codes/client tokens sent to the client application such that the code is sent partially in an HTTP request response and partially over the OS specific notification mechanism and the client must combine the data sent over notification mechanism and HTTP response to get the authorization code/token.

#### Using a Pre-Authorization code

In a classic OAuth scenario, the Authorization endpoint is not protected by any security credentials, during or before authorization code creation. The operation on the endpoint where the authorization code or the client token is produced can be compromised in a Mobile OAuth scenario if a fake device profile was submitted during a Mobile OAuth request (during client registration or afterwards while requesting for an Authz Code) or if the user password that is used during mobile application registration was compromised.

The OAM OAuth 2.0 Service introduces a Pre-Authz Code to provide an extra layer of protection for the Authorization endpoint in a Mobile OAuth Scenario. A Mobile Client needs to acquire the Pre-Authz Code first as the pre-requisite before interacting with OAuth Authorization endpoint. This Pre-Authz Code can be used exactly once similar to the Authz Code. Furthermore, in order to ensure that only the original client application (where authorization requests originated from) can obtain the Pre-Authz code, splitting logic is applied and only the original client application requests the second portion of those security codes/tokens through the Mobile OS specific communication channel (APNS/GCM).

#### Server Side Device Store and Server Side SSO for Mobile Applications

As mentioned earlier, the OAM OAuth 2.0 Service performs mobile application registration and mobile device identification to first secure the mobile application on the device during the Mobile OAuth flow to ensure trusted access from the Mobile OAuth Client. However, this core functionality comes with a price i.e. that the data handles that correspond to the registered device and per application session information must be stored and updated.

In order to efficiently solve the problem mentioned above, the OAM OAuth 2.0 Service introduces a server side device store feature. Examples of security material stored in the server side device store include User Tokens (JWT or OAM) and OAM specific handles for the mobile application such as

a “oaam.device” handle and a “oaam.session” handle. The built-in use of a server side store provides the following benefits:

- Secure token Sharing: Some of the tokens and security material (e.g. user tokens and the “oaam.device” handle) need to be shared among multiple client applications. If those applications are from different application vendors / publishers, there is no effective and secure way to share those tokens and keep them in sync within those mobile applications
- Higher security: Even when a device or a client application is compromised the token values never get leaked as these tokens are never sent back to the client.
- Significantly decreased burden on the Mobile Application: Instead of asking a mobile client application to manage all these security materials and keep their lifecycle in sync, this logic is handled securely on the server side in a centralized fashion.

A key innovation provided by the use of a server side device store in the OAM OAuth 2.0 Service is built-in Single Sign On (SSO) for Mobile OAuth Client applications. The client token obtained during mobile application registration contains a device hardware ID that serves as an “Index Key” to a server side device store entry and the client token itself serves as a “Security Key” to access security-sensitive data inside that particular server side device store entry. After identifying a server side device store entry with a hardware ID in the device profile, the server side device store logic retrieves security data from the server side store and can detect the presence of a valid user session in the context of an OAuth request allowing secure SSO across multiple client applications for the user.

## Additional Differentiations

Following are some additional key features provided by the OAM OAuth 2.0 Service that leverage its built-in integration with OAM and provide higher levels of security and flexibility for enterprise usage:

1. Multi-Tenancy support for Cloud deployments
2. Supports multiple target applications through a single OAuth Service end-point
3. Token Expiry time customization based on each Target Application requirements
4. Static and Dynamic user profile attributes in OAuth Access Tokens
5. Extensions Support
6. Supports OAuth Assertion specifications for SAML bearer & JWT tokens
7. Built-in integration with OAM during resource owner authentication and consent allowing:
8. Leveraging any supported OAM authentication scheme(s)
9. Fraud Detection & Strong Authentication
10. Single Sign On and Session Management
11. OAM Resource Protection
12. Allow protecting OAM webgate resources with OAuth tokens.
13. Common OAM configuration, deployment and infrastructure

## Sample Use Case

### Scenario Description

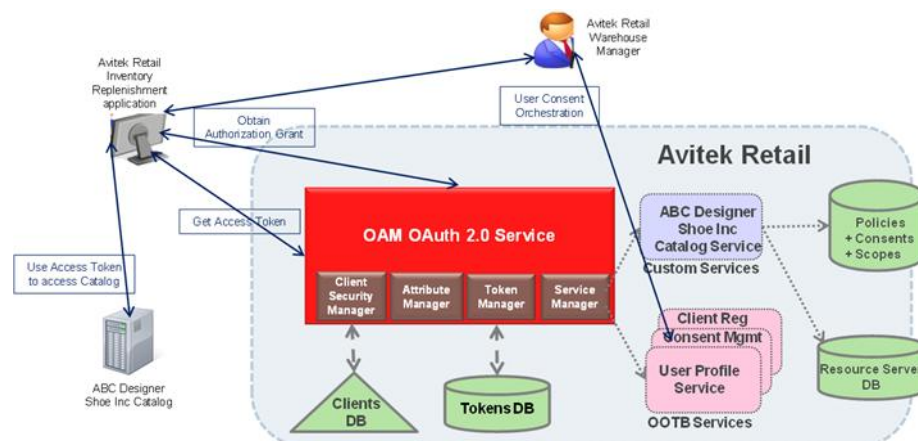
Avitek Retail is a large multi brand retail enterprise. In addition to several other products, Avitek Retail also sells shoes and accessories from different competing brands in their retail outlets. Avitek Retail's warehouse managers must ensure that adequate inventory levels are maintained to meet customer demand and they typically need access to supplier catalogs to replenish inventory.

Avitek Retail- an Oracle Access Manager customer has built its own inventory replenishment application that is used by its warehouse managers. Avitek Retail originally built its inventory replenishment application as a web based application but has also recently made it available as a mobile native application for both the iOS and Android platforms that can be downloaded from iTunes or the Google Play app stores.

In our scenario, ABC Designer Shoe Inc provides a catalog of designer shoes as a REST service that requires a valid OAuth 2.0 access token. This catalog needs to be accessed by Avitek Retail's Inventory Replenishment application. From an OAuth perspective then we have the following scenario:

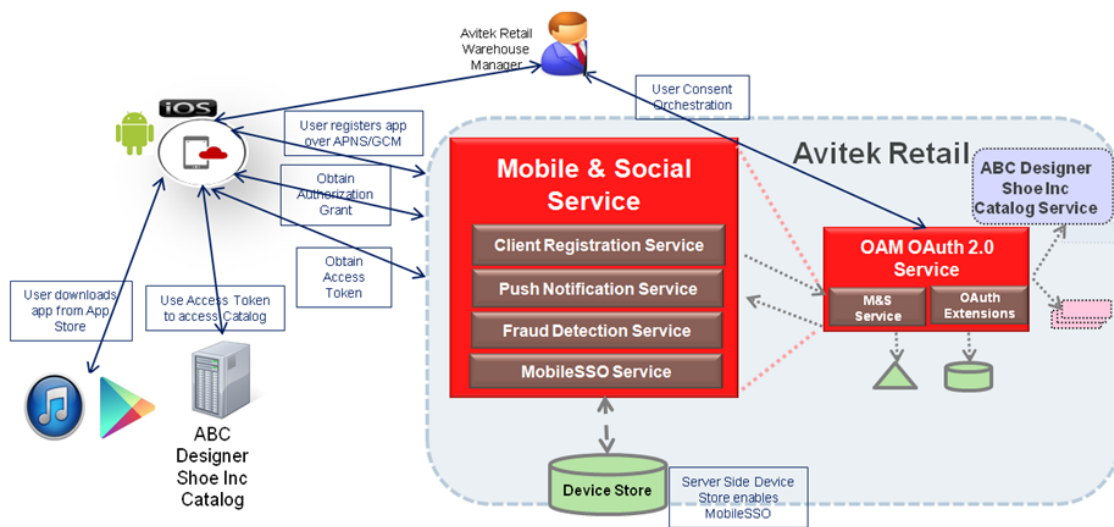
- OAuth 2.0 Server: Oracle Access Manager 11gR2PS2 deployed at Avitek Retail
- OAuth 2.0 Client: Avitek Retail's Inventory Replenishment application
- OAuth Resource Server: ABC Designer Shoe Catalog
- Resource Owner: Tom Dole- Warehouse Manager at Avitek Retail who owns the shoe catalog provided by ABC Designer Shoe Inc to Avitek Retail.

### 3-legged OAuth Flow with Authorization Code Grant Type



1. Tom Dole (ABC Shoe Inc Catalog Resource Owner and Avitek Retail Warehouse Manager) attempts to access the Inventory Replenishment application( OAuth Client) and requests that it access resources at a different site (ABC Designer Shoe Inc Catalog Resource Server)
2. The Inventory Replenishment application (OAuth client) invokes the authorization server endpoint on the OAM OAuth 2.0 authorization server at Avitek Retail.
3. The OAM OAuth 2.0 Authorization server at Avitek Retail redirects the Inventory Replenishment application (OAuth Client) via user-agent redirection for user authentication and presents Tom Dole with a consent page on successful authentication.
4. On receiving Tom Dole's consent, the OAM OAuth 2.0 Authorization server at Avitek Retail sends an authorization code to the Inventory Replenishment application (OAuth Client)
5. The Inventory Replenishment application (OAuth Client) uses the authorization code to retrieve an OAuth Access Token from the OAM OAuth 2.0 Authorization Server by POSTing to the OAM OAuth 2.0 Authorization server token endpoint at Avitek Retail
6. The Inventory Replenishment application (OAuth Client) presents the Access Token to the ABC Designer Inc Shoe Catalog (OAuth Resource Server)
7. The ABC Designer Shoe Inc Catalog (Resource Server) validates the token with the OAM OAuth 2.0 Authorization server at Avitek Retail
8. The ABC Designer Shoe Inc Catalog (Resource Server) provides the requested content to the Inventory Replenishment application (OAuth Client)

### Mobile OAuth Flow



1. Tom Dole (ABC Shoe Inc Catalog Resource Owner and Avitek Retail Warehouse Manager) attempts to access the Inventory Replenishment application (Mobile OAuth Client).
2. The Inventory Replenishment application (Mobile OAuth client) requests a pre-authorization code with a device token from the OAM OAuth 2.0 Server.
3. The OAM OAuth 2.0 Server returns pre-authorization code over APNS(iOS)/GCM (Android).
4. The Inventory Replenishment application (Mobile OAuth Client) sends a registration request with device claims and pre-authorization code to the OAM OAuth 2.0 Server.
5. The OAM OAuth 2.0 server uses user-agent redirection, authenticates the user, checks for device security, gets user consent for Mobile OAuth Client registration ( - if Tom Dole is accessing the app for the first time on the mobile device) and returns a client token over APNS/GCM to the Mobile OAuth Client.
6. The traditional OAuth flow starts now. The OAuth server checks for device security and depending on the configuration, presents the user with a consent page for accessing the ABC Shoe Designer Inc Catalog
7. The OAM OAuth 2.0 authorization server sends an authorization code to the Inventory Replenishment application (Mobile OAuth client)
8. The Inventory Replenishment application (Mobile OAuth client) requests for an access token from the OAuth authorization server by sending the authorization code, client token and device claims.
9. The OAM OAuth 2.0 server returns access token with optional refresh token to the Inventory Replenishment application (Mobile OAuth Client)
10. The Inventory Replenishment application (Mobile OAuth Client) presents the Access Token to the ABC Shoe Designer Inc Catalog (OAuth 2.0 Resource Server)
11. The ABC Shoe Designer Inc Catalog (OAuth 2.0 Resource Server) validates the token with the OAM OAuth 2.0 Authorization Server
12. The ABC Shoe Designer Inc Catalog (OAuth 2.0 Resource Server) provides the requested content to the Inventory Replenishment application (Mobile OAuth Client)

## Conclusion

The Oracle Access Manager OAuth 2.0 Service provides a fully standards compliant OAuth 2.0 authorization Server with support for both 3-legged and 2-legged OAuth flows and enables the OAuth 2.0 Client and the OAuth 2.0 Resource Server roles. It uniquely provides several compelling differentiations and innovations for mobile OAuth 2.0 clients (such as native applications on mobile devices) specifically for extranet access in enterprise scenarios. These include built-in support for mobile application registration and device identification during the OAM OAuth 2.0 mobile flow ensuring trusted access from mobile devices and built-in server side single sign on for mobile OAuth clients. It is ideally suited for enterprise scenarios that may require higher levels of security during an OAuth flow and would benefit from built-in OAM integrations provided by the OAM OAuth 2.0 service

**ORACLE®**

---

**FUSION MIDDLEWARE**





**Oracle OAuth Service**

Author: Kanishk Mahajan

March 2015




**Oracle Corporation, World Headquarters**


500 Oracle Parkway  
Redwood Shores, CA 94065, USA

**Worldwide Inquiries**


Phone: +1.650.506.7000  
Fax: +1.650.506.7200

**CONNECT WITH US**

 [blogs.oracle.com/oracle](http://blogs.oracle.com/oracle)

 [facebook.com/oracle](http://facebook.com/oracle)

 [twitter.com/oracle](http://twitter.com/oracle)

 [oracle.com](http://oracle.com)

**Hardware and Software, Engineered to Work Together**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.0115





