

Oracle® Fusion Middleware

Developer's Guide for Oracle Adaptive Access Manager

Release 11g (11.1.1)

E15480-02

August 2010

Oracle Fusion Middleware Developer's Guide for Oracle Adaptive Access Manager, Release 11g (11.1.1)

E15480-02

Copyright © 2009, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Priscilla Lee

Contributors: Mandar Bhatkhande, Roopang Chauhan, Sree Chitturi, Josh Davis, Jordan Douglas, Bosco Durai, Philomina Dorai, Arunkumar Jayaraman, Daniel Joyce, Mark Karlstrand, Wei Jie Lee, Derick Leo, Srinivas Nagandla, Madhan Neethiraj, Paresh Raote, Uday Sambhara, Kamal Singh, Nandini Subramani, Elangovan Subramanian, Vidhya Subramanian, Dawn Tyler, and Sachchidanand Vanungare

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

This documentation is in prerelease status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. 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 in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

Contents

Audience.....	xi
Documentation Accessibility.....	xi
Related Documents.....	xii
Conventions.....	xii

1 Introduction to the Developer's Guide

1.1 Native Integration.....	1-1
1.2 Universal Installation Option Integrations.....	1-2
1.3 Features Integration.....	1-2
1.4 Customizations and Extension.....	1-3
1.5 Authentication and Password Management Integration.....	1-3
1.6 Lifecycle Management.....	1-3
1.7 Troubleshooting/FAQ.....	1-3

Part I Native Integration

2 Natively Integrating with Oracle Adaptive Access Manager

2.1 Overview.....	2-1
2.1.1 Using Web Services and SOAP API.....	2-2
2.1.2 Using Static Linking.....	2-2
2.2 Integration Options.....	2-3
2.2.1 Integrating with Virtual Authentication Devices and Knowledge-Based Authentication 2-3	
2.2.1.1 User Name Page (S1).....	2-5
2.2.1.2 Device Fingerprint Flow (F2).....	2-5
2.2.1.3 Running Pre-Authentication Rules (R1).....	2-6
2.2.1.4 Running Virtual Authentication Device Rules (R2).....	2-6
2.2.1.5 Generating a Generic TextPad (P2).....	2-7
2.2.1.6 Generating a Personalized TextPad or Keypad (P3).....	2-8
2.2.1.7 Displaying TextPad and Keypad (S4 and S5).....	2-9
2.2.1.8 Decoding Virtual Authentication Device Input (P4).....	2-9
2.2.1.9 Validate User and Password (CP1).....	2-10
2.2.1.10 Update Authentication Status (P5).....	2-10
2.2.1.11 Password Status (C1).....	2-11
2.2.1.12 Post-Authentication Rules (R3).....	2-11

2.2.1.13	Check Question Registration for User (C2)	2-11
2.2.1.14	Registration Required Rules (R4)	2-11
2.2.1.15	Challenging the User with QuestionPad (S6).....	2-12
2.2.1.16	Checking Answers to Challenge Questions (C3)	2-13
2.2.1.17	Run Challenge Rules (R5).....	2-13
2.2.1.18	Lock Out Page (S2)	2-13
2.2.1.19	Landing or Splash Page (S3)	2-13
2.2.2	Integrating with Knowledge-Based Authentication.....	2-14

3 Integrating Native .NET Applications

3.1	Overview	3-1
3.2	Installing Oracle Adaptive Access Manager .NET SDK	3-2
3.3	Application Configuration	3-2
3.4	Properties	3-2
3.5	User-Defined Enumerations.....	3-3
3.6	User Details.....	3-4
3.7	User Logins and Transactions.....	3-5
3.8	Rules Engine	3-6
3.8.1	Device ID.....	3-7
3.8.2	Creating and Updating Bulk Transactions	3-7
3.9	Validating a User with Challenge Questions.....	3-7
3.10	Resetting Challenge Failure Counters	3-8
3.11	Virtual Authentication Devices.....	3-8
3.11.1	Creating a Virtual Authentication Device.....	3-8
3.11.2	Embedding a Virtual Authentication Device in a Web Page	3-9
3.11.3	Validating User Input with a Virtual Authentication Device	3-9
3.12	Specifying Credentials to the Oracle Adaptive Access Manager SOAP Server	3-9
3.13	Encrypting Property Values	3-10
3.14	Tracing Messages	3-10
3.15	ASP.NET Sample Applications	3-11
3.15.1	SampleWebApp	3-12
3.15.2	SampleWebAppWithTracker	3-12
3.15.3	SampleWebAppWithAuthTracker.....	3-13
3.15.4	SampleWebAppWithKBATracker	3-14

4 Integrating Native Java Applications

4.1	About the Oracle Adaptive Access Manager Shared Library	4-1
4.1.1	Using Oracle Adaptive Access Manager Shared Library in Applications	4-1
4.1.2	Customizing/Extending/Overriding Oracle Adaptive Access Manager Properties	4-1
4.2	About VCryptResponse	4-2
4.3	Oracle Adaptive Access Manager APIs.....	4-2
4.3.1	handleTrackerRequest	4-2
4.3.2	createTransaction	4-3
4.3.3	updateTransaction	4-4
4.3.4	handleTransactionLog	4-4
4.3.5	updateTransactionStatus	4-5
4.3.6	updateLog.....	4-6

4.3.7	getUserByLoginId.....	4-7
4.3.8	updateAuthStatus.....	4-7
4.3.9	processPatternAnalysis.....	4-9
4.3.10	markDeviceSafe	4-10
4.3.11	IsDeviceMarkedSafe.....	4-10
4.3.12	clearSafeDeviceList.....	4-10
4.4	Rules Engine	4-10
4.4.1	processRules	4-11
4.5	Customer Care.....	4-12
4.5.1	getFinalAuthStatus.....	4-12
4.5.2	setTemporaryAllow.....	4-12
4.5.3	cancelAllTemporaryAllows	4-12
4.5.4	resetUser.....	4-13
4.5.5	getRulesData.....	4-13
4.5.6	getActionCount.....	4-13

Part II Universal Installation Option and Related Integrations

5 Oracle Adaptive Access Manager Proxy

5.1	Introduction	5-1
5.1.1	Important Terms	5-2
5.1.2	Architecture	5-2
5.1.3	References	5-3
5.2	Installing Oracle Adaptive Access Manager Proxy for Microsoft ISA.....	5-4
5.2.1	Proxy Web Publishing Configuration	5-4
5.2.1.1	Web Listener Creation	5-4
5.2.1.2	Web Publishing Rule Creation	5-4
5.2.1.2.1	5-5
5.2.1.2.2	Web Publishing Rule Creation for Protected Web Applications	5-5
5.2.2	Registering the Oracle Adaptive Access Manager Proxy for Microsoft ISA DLL	5-6
5.2.3	Settings to Control the Proxy	5-6
5.2.3.1	Configuration files.....	5-6
5.2.3.2	Configuration Reload.....	5-7
5.2.3.3	Session ID Cookie	5-7
5.2.3.4	Configuring Session Id Cookie attributes via Global Variables	5-7
5.2.3.5	Session Inactive Interval	5-7
5.2.3.6	Settings for Troubleshooting.....	5-8
5.3	Installing Oracle Adaptive Access Manager Proxy for Apache.....	5-8
5.3.1	Proxy Files for Windows and Linux	5-9
5.3.1.1	Windows.....	5-9
5.3.1.2	Linux.....	5-10
5.3.2	Apache httpd Requirements	5-10
5.3.2.1	Windows.....	5-10
5.3.2.2	Linux.....	5-11
5.3.3	Copying the Oracle Adaptive Access Manager Proxy for Apache and Supported Files to Apache 5-11	
5.3.3.1	Windows.....	5-11

5.3.3.2	Linux.....	5-12
5.3.4	Configuring Memcache (for Linux only)	5-13
5.3.5	Configuring httpd.conf	5-14
5.3.5.1	Basic Configuration without SSL	5-14
5.3.5.2	Configuration with SSL	5-15
5.3.6	Modifying the Oracle Adaptive Access Manager Proxy for Apache Settings.....	5-15
5.3.6.1	UIO_Settings.xml.....	5-15
5.3.6.2	UIO_log4j.xml	5-18
5.3.6.3	Application configuration XMLs	5-18
5.4	Setting Up Rules and User Groups	5-18
5.5	Setting Up Policies	5-18
5.6	Configuring the Oracle Adaptive Access Manager Proxy.....	5-18
5.6.1	Elements of the Proxy Configuration File	5-18
5.6.1.1	Components of Interceptors.....	5-19
5.6.1.2	Conditions	5-19
5.6.1.3	Filters	5-22
5.6.1.4	Filter Examples - ProcessString	5-25
5.6.1.5	Filter Examples - FormatString.....	5-25
5.6.1.6	Actions.....	5-25
5.6.1.7	Variables	5-26
5.6.1.8	Application	5-28
5.6.2	Interception Process	5-28
5.6.3	Configuring Redirection to the Oracle Adaptive Access Manager Server Interface	5-29
5.7	Application Discovery.....	5-31
5.7.1	Application Information	5-31
5.7.2	Setting Up the Oracle Adaptive Access Manager Proxy for Microsoft ISA.....	5-32
5.7.3	Setting Up the Oracle Adaptive Access Manager Proxy for Apache.....	5-32
5.7.4	Scenarios.....	5-33
5.8	Samples.....	5-34

6 Configuring OAAM Server

6.1	Architecture	6-1
6.2	OAAM Server Settings.....	6-2
6.3	Determining Application ID and User Group.....	6-3
6.3.1	Determining the Application ID.....	6-3
6.3.2	Determining Default User Groups	6-3
6.4	Customizing User Interface Branding	6-4
6.4.1	Custom Header / Footer	6-4
6.4.2	Custom CSS	6-4
6.4.3	Custom Content and Messaging	6-5
6.5	Configuring Application Properties.....	6-5
6.5.1	Property Extension	6-6
6.5.2	User-Defined Enums	6-6
6.5.3	Overriding Existing User-Defined Enums.....	6-6
6.5.4	Disabling Elements.....	6-7

7 Virtual Authentication Device Properties

7.1	Property Files.....	7-1
7.2	Authentication Devices and Background Images.....	7-2
7.3	Display and Security Feature Properties.....	7-2
7.3.1	TextPad.....	7-2
7.3.1.1	TextPad Visual Elements.....	7-3
7.3.1.2	TextPad Authenticator Properties.....	7-4
7.3.2	QuestionPad.....	7-4
7.3.2.1	QuestionPad Visual Elements.....	7-5
7.3.2.2	QuestionPad Authenticator Properties.....	7-6
7.3.3	Keypad.....	7-6
7.3.3.1	KeyPad Visual Elements.....	7-7
7.3.3.2	KeyPad Authenticator Properties.....	7-8
7.3.4	PinPad.....	7-8
7.3.4.1	PinPad Visual Elements.....	7-9
7.3.4.2	PinPad Authenticator Properties.....	7-10
7.4	Accessibility.....	7-10
7.5	KeysSets.....	7-11
7.5.1	User Defined Enums Overview.....	7-11
7.5.2	KeySet Definition.....	7-11
7.6	Localization.....	7-13
7.6.1	Enabling Localization.....	7-13
7.6.2	Configuring Words Used in the Authenticator Caption.....	7-13
7.6.3	Localizing the KeyPad.....	7-14
7.6.4	Configuring Enter on the Authenticator Forgot Password Page.....	7-14
7.6.5	Configuring Tooltip for TextPad's Enter Button.....	7-14

Part III Features Integrations

8 Configurable Actions

8.1	Integration.....	8-1
8.2	Executing Configurable Actions in a Particular Order and Data Sharing.....	8-2
8.3	How to Test Configurable Actions Triggering.....	8-3
8.4	Sample JUnit Code.....	8-3

9 OTP Anywhere

9.1	OTP Integration.....	9-1
9.2	Implementing Challenge Processors.....	9-1
9.2.1	Create a Challenge Processor.....	9-2
9.2.2	Configure Custom Challenge Processor as Challenge Type with Required Profile Data 9-5	
9.3	Configuring the Challenge Pad Used for Challenge.....	9-6
9.4	Configuring User Information Properties.....	9-6
9.4.1	Configuration Settings for Information Registration and Preferences and PIN Generation 9-7	
9.4.2	Set Contact Information Inputs.....	9-7

9.4.3	SMS Configuration to Receive OTP via SMS.....	9-8
9.4.3.1	Enable Registration and Preference Setting.....	9-8
9.4.3.2	Set Input Information.....	9-8

10 Flash Fingerprinting

10.1	Device Fingerprinting	10-1
10.2	Definitions of Variables and Parameters	10-1
10.3	Option 1	10-2
10.3.1	Option 1 Flow.....	10-2
10.3.2	Option 1 Code Example.....	10-3
10.4	Option 2.....	10-3
10.4.1	Option 2 Flow.....	10-3
10.4.2	Option 2 Code Example.....	10-4
10.5	Option 3.....	10-5
10.5.1	Option 3 Flow.....	10-5
10.5.2	Option 3 Code Example.....	10-6
10.6	Common Update.....	10-7

11 Device Registration

Part IV Customizing Oracle Adaptive Access Manager

12 Customizing Oracle Adaptive Access Manager

12.1	Overview	12-1
12.2	Add Customizations Using Oracle Adaptive Access Manager Extensions Shared Library ...	12-1

Part V Authentication and Password Management Integration

13 Access and Password Management Integration

13.1	Benefits and Features of the Integration.....	13-1
13.2	Secure Password Collection and Management Scenarios.....	13-2

Part VI Lifecycle Management

14 Handling Lifecycle Management Changes

14.1	Oracle Virtual Directory (OVD) Host, Port, and SSL Enablement Changes.....	14-1
14.2	Oracle Identity Manager (OIM) URL Changes.....	14-2
14.3	Oracle Access Manager (OAM) Host and Port Changes	14-3
14.4	Oracle Internet Directory (OID) Host and Port Changes and SSL Enablement	14-3
14.5	Database Host and Port Changes	14-4
14.6	Moving Oracle Adaptive Access Manager to a New Production Environment	14-4
14.7	Moving Oracle Adaptive Access Manager to an Existing Production Environment ...	14-5

Part VII Troubleshooting

15 FAQ/Troubleshooting

15.1	Universal Installation Option Proxy	15-1
15.2	Virtual Authentication Devices.....	15-3
15.3	Configurable Actions.....	15-5
15.4	One-Time Password	15-5
15.5	Localization.....	15-6
15.6	Man-in-the-Middle/Man-in-the-Browser	15-7

Index

Preface

The *Oracle® Fusion Middleware Developer's Guide for Oracle Adaptive Access Manager* provides information about Oracle Adaptive Access Manager integrations.

The Preface covers the following topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Related Documents](#)
- [Conventions](#)

Audience

This guide is intended for administrators and developers who are responsible for integrating Oracle Adaptive Access Manager.

This guide assumes that you are familiar with your Web servers, Oracle Adaptive Access Manager, .NET and Java, 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 Fusion Middleware 11g Release 1 (11.1.1) documentation set:

- *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager*
- *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*
- *Oracle Fusion Middleware High Availability Guide*
- *Oracle Fusion Middleware Upgrade Planning Guide*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Reference for Oracle Identity Management*

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.
<i>italic</i>	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.

Introduction to the Developer's Guide

The *Oracle Fusion Middleware Developer's Guide for Oracle Adaptive Access Manager* provides information to help developers integrate and customize Oracle Adaptive Access Manager and manage configuration changes in integrated deployments of Oracle Adaptive Access Manager.

Information in this book is grouped into the following main parts to help developers quickly locate information:

- Part I - Native integration
- Part II - Universal Installation Option and related integrations
- Part III - Features integration
- Part IV - Oracle Adaptive Access Manager customizations
- Part V - Oracle Adaptive Access Manager, Oracle Access Manager, and Oracle Identity Manager integration
- Part VI - Lifecycle Management
- Part VII - Troubleshooting tips/FAQ

Detailed information about Oracle Adaptive Access Manager integration with Oracle Identity Manager and Oracle Access Manager is not covered in this guide. Refer to the *Oracle Fusion Middleware Integration Guide for Oracle Access Manager* for in-depth conceptual and procedural information.

1.1 Native Integration

Oracle Adaptive Access Manager provides APIs to fingerprint devices, collect authentication and transaction logs, run security and business rules, challenge the user to provide correct answers to pre-registered questions, and generate authentication pads such as KeyPad, TextPad, or QuestionPad.

Part 1 contains information about APIs used to integrate Oracle Adaptive Access Manager.

Native Integration Guidelines

An introduction to integrating a client application with Oracle Adaptive Access Manager is presented in [Chapter 2, "Natively Integrating with Oracle Adaptive Access Manager."](#) In native integration, the application invokes Oracle Adaptive Access Manager directly and the application itself manages the authentication and challenge flows.

Native and Web Services Integration

A Web application can communicate with Oracle Adaptive Access Manager using the OAAM Native Client API or through Web Services.

For information on these integrations, see [Chapter 3, "Integrating Native .NET Applications,"](#) and [Chapter 4, "Integrating Native Java Applications."](#)

Static Linked Integration

The native integrations include APIs that are wrappers of the SOAP API published by OAAM and written in the client's native application language.

The static linked integration is an option available for integrations using just the Java language. In this integration, there are no SOAP calls to OAAM, and, instead, the API implementation runs within the client application itself.

For information on the static linked integration, see [Chapter 4, "Integrating Native Java Applications."](#)

1.2 Universal Installation Option Integrations

Oracle Adaptive Access Manager's Universal Installation Option (UIO) reverse proxy deployment option offers login risk-based multifactor authentication to Web applications without requiring any change to the application code.

Part II contains configuration instructions and guidelines for the reverse proxy deployment option in the following chapters:

- [Chapter 5, "Oracle Adaptive Access Manager Proxy"](#)
- [Chapter 6, "Configuring OAAM Server"](#)
- [Chapter 7, "Virtual Authentication Device Properties"](#)

1.3 Features Integration

Part III provides instructions and reference material for the following feature integrations:

Configurable Actions

Oracle Adaptive Access Manager provides Configurable Actions, a feature which allows users to create new supplementary actions that are triggered based on the result action and/or based on the risk scoring after a checkpoint execution.

[Chapter 8, "Configurable Actions"](#) describes how to integrate a Configurable Action with the Oracle Adaptive Access Manager software.

One-Time Password

Oracle Adaptive Access Manager 11g provides the framework to support the One Time Password authentication method.

One Time Password (OTP) is used to authenticate an individual based on a single-use alphanumeric credential.

[Chapter 9, "OTP Anywhere"](#) provides an example of how to integrate OTP into the system.

Device Registration

Device registration is a feature that allows a user to flag the computer he is using as a safe device. Instructions to enable the feature is provided in [Chapter 11, "Device Registration."](#)

Flash Fingerprinting

Oracle Adaptive Access Manager uses device fingerprinting along with many other types of data to determine the risk associated with a specific access request. Outlines of calls needed to perform the flash fingerprinting are presented in [Chapter 10, "Flash Fingerprinting."](#)

1.4 Customizations and Extension

Oracle Adaptive Access Manager can be customized by adding custom jars and files to the Oracle Adaptive Access Manager Extensions Shared Library.

Part IV provides instructions for customizations in [Chapter 12, "Customizing Oracle Adaptive Access Manager."](#)

1.5 Authentication and Password Management Integration

Benefits of the Oracle Access Manager-Oracle Adaptive Access Manager-Oracle Identity Manager integration is presented in [Chapter 13, "Access and Password Management Integration."](#)

1.6 Lifecycle Management

Because of integrated deployment of Oracle Adaptive Access Manager with other applications, configuration changes in those applications might be required in Oracle Adaptive Access Manager.

Part VI contains examples for handling these configuration changes in [Chapter 14, "Handling Lifecycle Management Changes."](#)

1.7 Troubleshooting/FAQ

[Chapter 15, "FAQ/Troubleshooting"](#) provides troubleshooting tips and answers to frequently asked questions.

Part I

Native Integration

Oracle Adaptive Access Manager provides APIs to fingerprint devices, collect authentication and transaction logs, run security and business rules, challenge the user to answer correctly pre-registered questions and answers, and to generate authentication pads such as KeyPad, TextPad, or QuestionPad.

Part 1 contains information about APIs used to integrate Oracle Adaptive Access Manager in the following chapters:

- [Chapter 2, "Natively Integrating with Oracle Adaptive Access Manager"](#)
- [Chapter 3, "Integrating Native .NET Applications"](#)
- [Chapter 4, "Integrating Native Java Applications"](#)

Natively Integrating with Oracle Adaptive Access Manager

Oracle Adaptive Access Manager provides APIs to fingerprint devices, collect authentication and transaction logs, run security rules, challenge the user to answer pre-registered questions correctly, and generate virtual authentication devices such as KeyPad, TextPad, or QuestionPad.

In native integration, the application invokes Oracle Adaptive Access Manager directly and the application itself manages the authentication and challenge flows.

Using the Oracle Adaptive Access Manager APIs, you can:

- Change the default user registration flow.
- Control and manage the authentication process flow.

This chapter contains guidelines to integrate a client application with Oracle Adaptive Access Manager using the APIs the server exposes. The typical process flows for the authentication and challenge scenarios are presented in this chapter. Within these flow sections, there are details about which API should be called at each stage.

The integration options are presented in the following sections:

- [Using Web Services and SOAP API](#)
- [Using Static Linking](#)
- [Integrating with Virtual Authentication Devices and Knowledge-Based Authentication](#)
- [Integrating with Knowledge-Based Authentication](#)

2.1 Overview

To integrate with Oracle Adaptive Access Manager, the application uses the native API.

You can choose one of the following native API options:

- Use the SOAP service wrapper API for Java or .NET applications.
Refer to "[Using Web Services and SOAP API](#)".
- Link libraries statically for Java applications only
Refer to "[Using Static Linking](#)".

2.1.1 Using Web Services and SOAP API

In this scenario, the application communicates with Oracle Adaptive Access Manager using the Oracle Adaptive Access Manager native client API (SOAP service wrapper API) or via Web services.

The SOAP service wrapper API enables you to create SOAP objects and invoke SOAP calls and abstracts the SOAP WSDL and other Web services details from the application code. Libraries for this API are available for the following languages: Java, .NET, and C++.

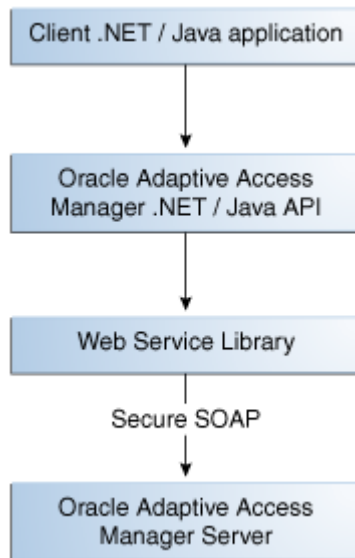
Using this API is recommended over making direct SOAP calls. The reasons are as follows:

- The client library constructs the SOAP objects, and the details involved in SOAP calls is abstracted from the client application.
- A SOAP API signature change does not require any change in the client code.
- The API provides higher-level utility methods to extract parameters directly from the HTTP request and HTTP session objects.
- It provides methods to encode and decode fingerprint data.

This integration requires adding lightweight client libraries (JARs or DLLs) to the client library.

Figure 2–1 illustrates how an application communicates with Oracle Adaptive Access Manager using Web services and the server API.

Figure 2–1 Client Application Using Web Services and Server API



2.1.2 Using Static Linking

Java applications can be static-linked. This scenario only involves local API calls and therefore no remote server risk engine calls (SOAP calls). The integration imbeds the processing engine for Oracle Adaptive Access Manager with the application and enables it to leverage the underlying database directly for processing. In this scenario, the application must include the server JARs and configured properties, as appropriate.

Even though static linking may provide slightly better performance, it is not suitable for all Java clients.

Static linking is recommended for clients developing their own applications with Oracle Adaptive Access Manager built in their J2EE or application.

Static-linking an application has several advantages:

- The application makes no SOAP calls, thus eliminating the need to create and delete TCP/IP connections.
- It experiences no network latencies.
- It does not require a load balancer.

2.2 Integration Options

This section describes the following integration options:

- [Integrating with Virtual Authentication Devices and Knowledge-Based Authentication](#)
- [Integrating with Knowledge-Based Authentication](#)

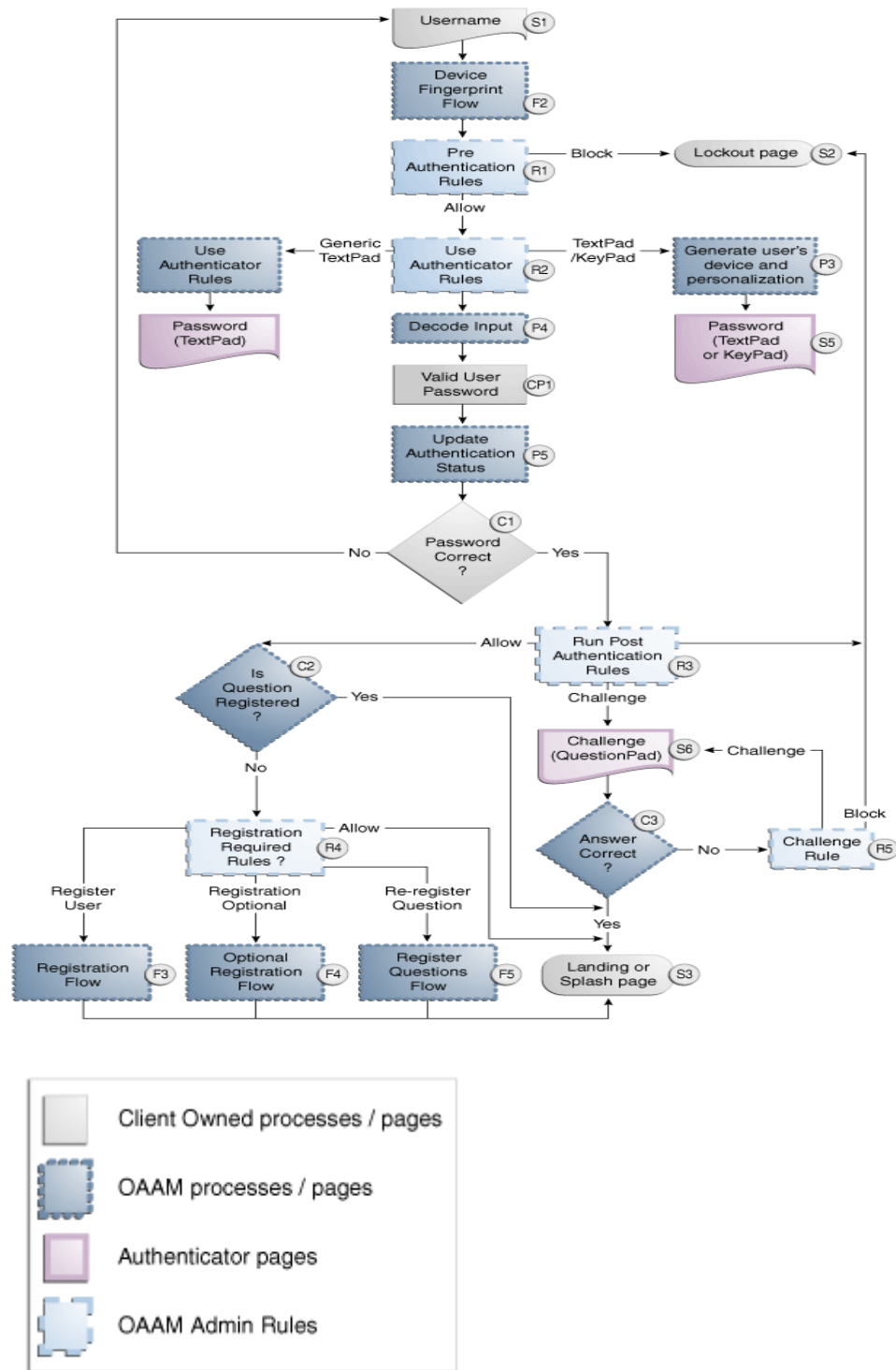
2.2.1 Integrating with Virtual Authentication Devices and Knowledge-Based Authentication

This integration consolidates virtual authentication devices and knowledge-based authentication.

Globalized virtual authentication device image files including registration flows must be developed by the deployment team.

[Figure 2–2](#) illustrates an authentication flow example that uses these two solutions (virtual authentication devices and knowledge-based authentication). Note that the flow illustrated is an example and that other authentication flows are possible.

Figure 2–2 Virtual Authentication Devices and Knowledge-Based Authentication Scenario



The details of the stages in the [Figure 2–2](#) are explained in the following sections:

- [User Name Page \(S1\)](#)
- [Device Fingerprint Flow \(F2\)](#)
- [Running Pre-Authentication Rules \(R1\)](#)

- Running Virtual Authentication Device Rules (R2)
- Generating a Generic TextPad (P2)
- Generating a Personalized TextPad or KeyPad (P3)
- Displaying TextPad and KeyPad (S4 and S5)
- Decoding Virtual Authentication Device Input (P4)
- Validate User and Password (CP1)
- Update Authentication Status (P5)
- Password Status (C1)
- Post-Authentication Rules (R3)
- Check Question Registration for User (C2)
- Registration Required Rules (R4)
- Challenging the User with QuestionPad (S6)
- Checking Answers to Challenge Questions (C3)
- Run Challenge Rules (R5)
- Lock Out Page (S2)
- Landing or Splash Page (S3)

2.2.1.1 User Name Page (S1)

When the application uses a custom login page, the login page must be split into two pages. The user inputs the login ID (user name) in the first page, and this data is stored in the HTTP session. The second login page is a transient page to capture the flash and secure cookies and for fingerprinting the user device. [Figure 2-3](#) shows a sample of the first page.

Figure 2-3 User Name Page



ORACLE

Sign In:
Please enter your username.

Username:

[Where do I enter my password?](#)

2.2.1.2 Device Fingerprint Flow (F2)

The device fingerprint stage involves fingerprinting the user device. The APIs used for this purpose are detailed in [Table 2-1](#).

Table 2–1 Device Fingerprinting APIs

Module	APIs	Description
Server	VCryptTracker::updateLog()	For method details, see Section 4.3.6, "updateLog."
Oracle Adaptive Access Manager Sample	handleJump.jsp	Sets the client's time zone Sets a secure cookie Sets the browser fingerprint Sets the status to pending Calls the pre-authentication rules; expects "allow" to allow the user to proceed or "block" or "error" to stop the user from continuing Stores bharosaSession Forwards the user to the password.jsp page
Oracle Adaptive Access Manager Sample	handleFlash.jsp	Sets the flashCookie if the browser is flash-enabled

2.2.1.3 Running Pre-Authentication Rules (R1)

Pre-authentication rules are run before the user is authenticated. Common values returned by the pre-authentication checkpoint include:

- **Allow** to allow the user to proceed forward.
- **Block** to block the user from proceeding forward.

The APIs used for pre-authentication are listed in [Table 2–2](#).

Table 2–2 Pre-Authentication Rules Reference APIs

Module	APIs	Description
Server	VCryptRulesEngine::processRules()	For method details, see Section 4.4.1, "processRules."
Oracle Adaptive Access Manager Sample	handleJump.jsp	Invokes the pre-authentication rules; returns "allow" to proceed forward to password.jsp or "block" or "error" to signal an error Stores bharosaSession
BharosaHelper	BharosaHelper::runPreAuthRules()	

2.2.1.4 Running Virtual Authentication Device Rules (R2)

This stage determines the virtual authentication device to use. If the user has not registered an image and a phrase, the rule returns the Generic TextPad; otherwise, if the user has registered, the rule returns either the personalized TextPad or KeyPad. Common values returned by virtual authentication devices include:

- **Generic TextPad**, to use the default generic TextPad.
- **TextPad**, to use a personalized TextPad.
- **KeyPad**, to use a personalized KeyPad.

The APIs used to run virtual authentication device rules are listed in [Table 2–3](#).

Table 2–3 Virtual Authentication Device Rules APIs

Module	APIs	Description
Server	VCryptRulesEngine::processRules()	For method details, see Section 4.4.1, "processRules."
Oracle Adaptive Access Manager Sample	password.jsp	<p>Invokes rules to identify the user's virtual authentication device type</p> <p>Creates the virtual authentication device, names it, and sets all initial background frames</p> <p>Invokes kbimage.jsp as configured</p> <p>Forwards to page handlePassword.jsp</p>
BharosaHelper	BharosaHelper::getAuthentiPad()	

2.2.1.5 Generating a Generic TextPad (P2)

A generic, non-personalized TextPad is used for users who have not yet registered with Oracle Adaptive Access Manager. [Figure 2–4](#) illustrates a generic TextPad.

Figure 2–4 Generic, Non-Personalized TextPad

[Table 2–4](#) lists the APIs used to generate a generic TextPad.

Table 2–4 Generation of a GenericTextPad APIs

Module	APIs	Description
Server	VCryptAuth::getUserByLoginId()	For method details, see Section 4.3.7, "getUserByLoginId."
Oracle Adaptive Access Manager Sample	Password.jsp	<p>Invokes rules to identify the virtual authentication device type to use; the default is KeyPad</p> <p>Creates the virtual authentication device, names it, and sets all initial background frames</p> <p>Invokes kbimage.jsp as configured</p> <p>Forwards to page handlePassword.jsp</p>
BharosaHelper	BharosaHelper::createPersonalizedAuthentiPad () BharosaHelper::createAuthentiPad()	
Client	AuthentiPad::getHTML()	

2.2.1.6 Generating a Personalized TextPad or Keypad (P3)

A personalized TextPad is used for users who have registered with Oracle Adaptive Access Manager. [Figure 2-5](#) and [Figure 2-6](#) illustrate personalized text and key virtual authentication devices.

Figure 2-5 Personalized TextPad

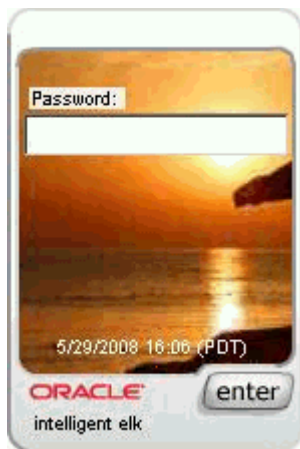
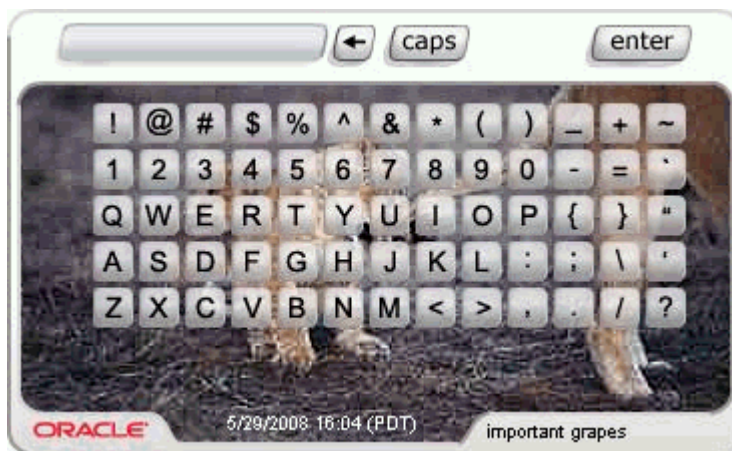


Figure 2-6 Personalized Keypad



[Table 2-5](#) lists the APIs used to generate a personalized TextPad or Keypad.

Table 2–5 *Generating a Personalized TextPad or KeyPad APIs*

Module	APIs	Description
Server	VCryptAuth::getUserByLoginId()	For method details, see Section 4.3.7 , "getUserByLoginId."
Oracle Adaptive Access Manager Sample	password.jsp	<p>Invokes rules to identify the virtual authentication device type to use; the default is KeyPad</p> <p>Creates the virtual authentication device, names it, and sets all initial background frames</p> <p>Forwards to page handlePassword.jsp</p> <p>Invokes kbimage.jsp as configured</p>
BharosaHelper	BharosaHelper:: createPersonalizedAuthentiPad () BharosaHelper::createAuthentiPad()	
Client	AuthentiPad::getHTML()	

2.2.1.7 Displaying TextPad and KeyPad (S4 and S5)

The HTML code example to display TextPad and KeyPad should be embedded in the password page. This HTML renders the TextPad or KeyPad using Javascript, and it includes an `` tag, which makes a HTTP request to the server to get the TextPad or KeyPad image.

[Table 2–6](#) lists the APIs used to display TextPad and KeyPad.

Table 2–6 *Displaying TextPad and KeyPad APIs*

Module	APIs	Description
Server	VCryptAuth::getUserByLoginId()	
Oracle Adaptive Access Manager Sample	password.jsp	<p>Invokes rules to identify the virtual authentication device type to use; the default is KeyPad</p> <p>Creates the virtual authentication device, names it, and sets all initial background frames</p> <p>Invokes kbimage.jsp as configured</p> <p>Forwards to page handlePassword.jsp</p>
Oracle Adaptive Access Manager Sample	kbimage.jsp	Outputs the virtual authentication device(s)
BharosaHelper	BharosaHelper:: createPersonalizedAuthentiPad () BharosaHelper::createAuthentiPad() BharosaHelper::imageToStream()	
Client	AuthentiPad::getHTML() KeyPadUtil::encryptImageToStream()	

2.2.1.8 Decoding Virtual Authentication Device Input (P4)

In this stage, the chosen virtual authentication device decodes the data the user supplies to it; the decoded value is in raw text format, and it is recommended that it be

saved in the HTTP Session. The Virtual Authentication Device object is can be serialized and stored in the database or the file system.

[Table 2–7](#) lists the APIs used to decode user input.

Table 2–7 Decoding Virtual Authentication Device Input APIs

Module	APIs	Description
Oracle Adaptive Access Manager Sample	handlePassword.jsp	Retrieves the password Decodes the password Validates the user
BharosaHelper	BharosaHelper::decodePadInput()	Removes the Virtual Authentication Device object from the HTTP Session
Client	KeyPadUtil::decodeKeyPadCode	

2.2.1.9 Validate User and Password (CP1)

This stage represents the client's existing process in which the client invokes the local API to authenticate the user, and the authentication result is passed on to Oracle Adaptive Access Manager Server. The API used is detailed in [Table 2–8](#).

Table 2–8 Validating User and Password API

Module	API	Description
Oracle Adaptive Access Manager Sample	handlePassword.jsp	Retrieves the password Decodes the password Updates the status to "success" (if user is valid), or to "invalid," "error," or "bad password" (if the user is invalid) Runs post-authentication rules and returns one of the following values: REGISTER_USER_OPTIONAL REGISTER QUESTIONS REGISTER_USER CHALLENGE

2.2.1.10 Update Authentication Status (P5)

After validating the user password, the status is updated with the APIs detailed in [Table 2–9](#).

Table 2–9 Updating Authentication Status APIs

Module	APIs	Description
Server	VCryptTracker::updateAuthStatus()	For method details, see Section 4.3.8, "updateAuthStatus."
Oracle Adaptive Access Manager Sample	handlePassword.jsp	Retrieves the password Decodes the password Validates the user Forwards to registerImageandPhrase, or challenges a registered user
BharosaHelper	BharosaHelper::updateStatus()	

2.2.1.11 Password Status (C1)

Depending on the password authentication status, the user is directed to the retry page or to post-authentication.

2.2.1.12 Post-Authentication Rules (R3)

These rules are run after the user password has been authenticated. Common actions returned by post-authentication include:

- **Allow**, to allow the user to proceed forward.
- **Block**, to block the user from proceeding forward.
- **Challenge**, to challenge the user, if he has registered questions.

The APIs used for post-authentication is listed in [Table 2–10](#).

Table 2–10 Post-Authentication Rules Reference APIs

Module	APIs	Description
Server	VCryptRulesEngine::processRules()	For method details, see Section 4.4.1, "processRules."
Oracle Adaptive Access Manager Sample	handlePassword.jsp	Runs post-authentication rules and returns one of the following values: REGISTER_USER_OPTIONAL REGISTER_QUESTIONS REGISTER_USER CHALLENGE BLOCK ALLOW SYSTEM_ERROR Forwards to registerImageandPhrase, or challenges a registered user
BharosaHelper	BharosaHelper::runPostAuthRules()	

2.2.1.13 Check Question Registration for User (C2)

This stage checks whether the user is registered; if he is not registered, the user is directed to do so.

2.2.1.14 Registration Required Rules (R4)

The registration is required depending on business and security requirements, which specify whether the registration is mandatory or optional. Values returned by registration rules include the following:

- **Register**, to require user registration.
- **Registration Optional**, to make user registration optional.
- **Skip Registration**, to skip registration for this session.

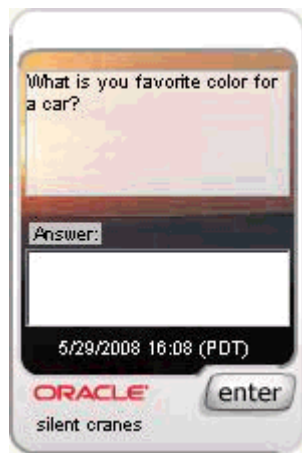
[Table 2–11](#) lists the APIs used to run registration rules.

Table 2–11 Registration Required Rules Reference APIs

Module	APIs	Description
Server	VCryptRulesEngine::processRules()	For method details, see Section 4.4.1 , "processRules."
Oracle Adaptive Access Manager Sample	password.jsp	<p>Invokes rules to identify the virtual authentication device type to use; the default is KeyPad</p> <p>Creates the virtual authentication device, names it, and sets all initial background frames</p> <p>Invokes kbimage.jsp as configured</p> <p>Forwards to page handlePassword.jsp</p>
BharosaHelper	BharosaHelper::getAuthentiPad()	

2.2.1.15 Challenging the User with QuestionPad (S6)

If appropriate, the user is challenged to answer a registered question. [Table 2–7](#) illustrates a QuestionPad where a challenge question is displayed.

Figure 2–7 Question Pad

[Table 2–12](#) lists the APIs to challenge the user with registered questions.

Table 2–12 Challenge User APIs

Module	APIs	Description
Server	VCryptAuth::getSecretQuestions()	
Oracle Adaptive Access Manager Sample	Challenge.jsp	<p>Gets the question</p> <p>Gets the QuestionPad</p> <p>Displays the question in the virtual authentication device</p> <p>Submits the answer to handleChallenge.jsp</p>
BharosaHelper	BharosaHelper::createPersonalizedAuthentiPad () BharosaHelper::createAuthentiPad()	
Client	AuthentiPad::getHTML()	

2.2.1.16 Checking Answers to Challenge Questions (C3)

This stage involves calling Oracle Adaptive Access Manager Server to determine whether the answer the user has supplied matches the registered reply.

Table 2–13 lists the APIs used to validate answers to a challenged question.

Table 2–13 Validate Answer to a Challenge Question APIs

Module	APIs	Description
Server	VCryptAuth::authenticateQuestion() VCryptRulesEngine::processRules() VCryptTracker::updateAuthStatus()	For method details, see Section 4.4.1 , "processRules," and Section 4.3.8 , "updateAuthStatus."
Oracle Adaptive Access Manager Sample	handleChallenge.jsp	Validates the user If valid, updates the status to "success" and allows the user to move forward; otherwise, if invalid, runs Challenge Rules to determine what should be done
BharosaHelper	BharosaHelper::validateAnswer()	

2.2.1.17 Run Challenge Rules (R5)

If the user fails to answer a challenge question correctly, the challenge rules is invoked to determine whether the user should be given another chance to answer the question or to block him from proceeding forward. Values returned by the challenge rules include the following:

- **Challenge**, to challenge the user one more time.
- **Block**, to block the user.

Table 2–14 lists the APIs used to run the challenge rules.

Table 2–14 Run Challenge Rules APIs

Module	APIs	Description
Server	VCryptRulesEngine::processRules()	For method details, see Section 4.4.1 , "processRules."
Oracle Adaptive Access Manager Sample	handleChallenge.jsp	Validates the user If valid, updates the status to "success" and allows the user to move forward; otherwise, if invalid, runs Challenge Rules to determine what should be done
BharosaHelper	BharosaHelper::validateAnswer()	

2.2.1.18 Lock Out Page (S2)

The Lock Out page is the page to which the user is redirected when the post-authorization rules return Block.

2.2.1.19 Landing or Splash Page (S3)

This page is the page to which the user is redirected after a successful login, that is, when the post-authorization rules return Allow.

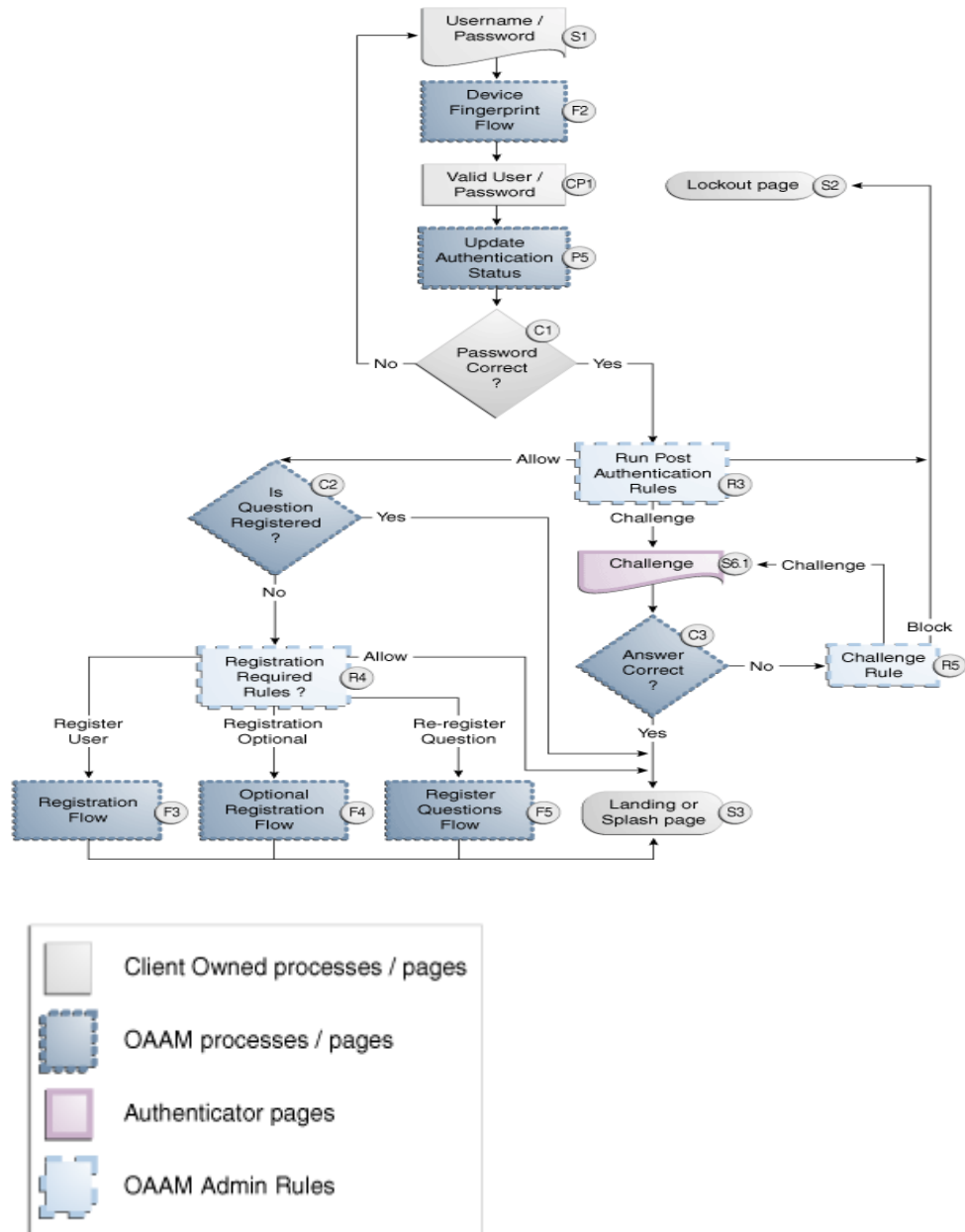
2.2.2 Integrating with Knowledge-Based Authentication

This scenario is a subset of the scenario described in [Section 2.2.1, "Integrating with Virtual Authentication Devices and Knowledge-Based Authentication."](#) This scenario does not have a split login flow and does not include personalizations or virtual authentication devices.

[Figure 2–8](#) illustrates a flow of authentication that uses this solution. For details about the stages of this flow, see the following sections:

- [Section 2.2.1.1, "User Name Page \(S1\)"](#)
- [Section 2.2.1.2, "Device Fingerprint Flow \(F2\)"](#)
- [Section 2.2.1.9, "Validate User and Password \(CP1\)"](#)
- [Section 2.2.1.10, "Update Authentication Status \(P5\)"](#)
- [Section 2.2.1.11, "Password Status \(C1\)"](#)
- [Section 2.2.1.12, "Post-Authentication Rules \(R3\)"](#)
- [Section 2.2.1.17, "Run Challenge Rules \(R5\)"](#)
- [Section 2.2.1.13, "Check Question Registration for User \(C2\)"](#)
- [Section 2.2.1.14, "Registration Required Rules \(R4\)"](#)
- [Section 2.2.1.15, "Challenging the User with QuestionPad \(S6\)"](#)
- [Section 2.2.1.18, "Lock Out Page \(S2\)"](#)
- [Section 2.2.1.19, "Landing or Splash Page \(S3\)"](#)

Figure 2-8 Knowledge-Based Authentication Scenario



Integrating Native .NET Applications

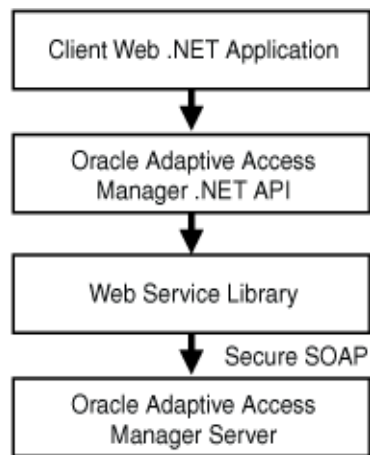
This chapter explains the integration of ASP.NET applications with Oracle Adaptive Access Manager using the .NET API provided by Oracle Adaptive Access Manager. It also contains a description of the four sample applications that illustrates the integration of different features with a basic Web application.

This chapter contains the following sections:

- [Overview](#)
- [Installing Oracle Adaptive Access Manager .NET SDK](#)
- [Application Configuration](#)
- [Properties](#)
- [User-Defined Enumerations](#)
- [User Details](#)
- [User Logins and Transactions](#)
- [Rules Engine](#)
- [Validating a User with Challenge Questions](#)
- [Resetting Challenge Failure Counters](#)
- [Virtual Authentication Devices](#)
- [Specifying Credentials to the Oracle Adaptive Access Manager SOAP Server](#)
- [Encrypting Property Values](#)
- [Tracing Messages](#)
- [ASP.NET Sample Applications](#)

3.1 Overview

ASP.NET applications, written in any ASP.NET language, use the OAAM .NET API to access Oracle Adaptive Access Manager. This API communicates with the server using SOAP, as illustrated in [Figure 3-1](#).

Figure 3–1 .NET Application

3.2 Installing Oracle Adaptive Access Manager .NET SDK

The Oracle Adaptive Access Manager .NET development kit (SDK) is packaged in the ZIP file, `Bharosa_SDK_DotNet2.0.zip`. The contents of this ZIP file should be extracted to the virtual directory of the Web application to be integrated with Oracle Adaptive Access Manager, and the Web application project files must be updated with references to the unzipped development kit DLLs.

3.3 Application Configuration

To configure the application, define the `BharosaSOAPURL` key in the `<appSettings>` section of the configuration file for the Web application, `web.config`.

The value of this key should be set to the URL that the application will use to access the Oracle Adaptive Access Manager Server SOAP services. The following section of a `web.config` file illustrates this setting:

```

<appSettings>
  <add key="BharosaSOAPURL" value="http://localhost:9090/oaam_server/services"/>
</appSettings>
  
```

3.4 Properties

The Oracle Adaptive Access Manager .NET SDK includes properties files that specify values for configuration used by the Oracle Adaptive Access Manager APIs. A developer can modify these properties to specify application-specific values or add new ones.

The Oracle Adaptive Access Manager .NET API uses these properties to read configurable values at runtime, such as the location of images for virtual authentication devices. Properties are read and cached from a list of files at startup and updated whenever one of the properties files is updated.

The sequence in which the properties files are loaded by Oracle Adaptive Access Manager .NET API is as follows:

1. The `lookup.properties` file, if present, is loaded first.

2. If the `properties.filelist` property is defined in `lookup.properties`, then all the files listed in that property are added to the queue (in the listed order).
3. The `bharosa_lookup.properties` file, if present, is loaded.
4. If the `properties.filelist` property is defined in `bharosa_lookup.properties`, then all the files listed in that property are added to the queue (in the listed order)
5. All files in the queue are loaded.
6. When any of the loaded properties files is changed, the properties are reloaded.

The properties files, including `lookup.properties`, are searched in the following directories in the order stated in [Table 3-1](#); the search for a given file stops when the file is first found or when no file is found.

Table 3-1 Directories Searched for Properties Files

Directory	Example
<ApplicationDirectory>/	c:/Inetpub/wwwroot/MyApp/
<CallingAssemblyDirectory>/	c:/Windows/System32/
<CurrentAssemblyDirectory>/	c:/Inetpub/wwwroot/MyApp/bin/
<CurrentAssemblyDirectory>/../	c:/Inetpub/wwwroot/MyApp/
<CurrentDirectory>/	c:/Windows/System32/
<ApplicationDirectory>/bharosa_properties/	c:/Inetpub/wwwroot/MyApp/bharosa_properties/
<CallingAssemblyDirectory>/bharosa_properties/	c:/Windows/System32/bharosa_properties/
<CurrentAssemblyDirectory>/bharosa_properties/	c:/Inetpub/wwwroot/MyApp/bin/bharosa_properties/
<CurrentAssemblyDirectory>/../bharosa_properties/	c:/Inetpub/wwwroot/MyApp/bharosa_properties/
<CurrentDirectory>/bharosa_properties/	c:/Windows/System32/bharosa_properties/

3.5 User-Defined Enumerations

A user-defined enumeration is a collection of items; each item is assigned an integer and may contain several attributes. A user-defined enumeration is specified in a properties file, and its name, the names of its items, and the name of the item attributes must conform to the following rules:

- The name of the enumeration has the suffix `.enum`
- The name of an item has a prefix equals to the name of the enumeration
- The name of an attribute of an item has a prefix equals to the name of the item

Here is an example of a user-defined enumeration:

```
#Example of a user-defined enumeration
auth.status.enum=Enumeration to describe authentication status

#first item and its attributes
auth.status.enum.success=0
auth.status.enum.success.name=Success
auth.status.enum.success.description=Success
auth.status.enum.success.success=true

#second item and its attributes
```

```
auth.status.enum.invalid_user=1
auth.status.enum.invalid_user.name=Invalid user
auth.status.enum.invalid_user.description=Invalid User

#third item and its attributes
auth.status.enum.wrong_password=2
auth.status.enum.wrong_password.name=Wrong password
auth.status.enum.wrong_password.description=Wrong password

#fourth item and its attributes
auth.status.enum.wrong_pin=3
auth.status.enum.wrong_pin.name=Wrong pin
auth.status.enum.wrong_pin.description=Wrong Pin

#fifth item and its attributes
auth.status.enum.session_expired=4
auth.status.enum.session_expired.name=Session expired
auth.status.enum.session_expired.description=Session expired
```

Here is an example of the use of the previous user-defined enumeration in application code:

```
UserDefEnumFactory factory = UserDefEnumFactory.getInstance();
UserDefEnum statusEnum = factory.getEnum("auth.status.enum");
int statusSuccess = statusEnum.getElementValue("success");
int statusWrongPassword = statusEnum.getElementValue("wrong_password");
```

3.6 User Details

Oracle Adaptive Access Manager stores user details in its database and uses this information to perform the following tasks:

- Determine the risk rules to run for a user
- Find user-specific virtual authentication device attributes
- Propose challenge questions
- Validate answers to challenge questions

The client application is responsible for populating the Oracle Adaptive Access Manager database with user details at runtime.

For example, when a user logs in, the client application should first determine whether the user record exists. If the record is not found, then the application should call the appropriate APIs to create a user record and set the user status.

The following sample illustrates the calls to create a user record:

```
string loginId = "testuser"; // loginId of the user logging in

// set the proxy to access the SOAP server that communicates with the
// OAAM SOAP Server
IBharosaProxy proxy = BharosaClientFactory.getProxyInstance();

// find the user record in OAAM
VCryptAuthUser user = proxy.getUserByLoginId(loginId);

// if user record does not exist, create one
if(user == null || StringUtil.IsEmpty(user.LoginId))
{
    string customerId = loginId;
```

```

string userGroupId = "PremiumCustomer";
string password    = "_"; // this value is not used for now

user = new VCryptAuthUser(loginId, customerId,
                          userGroupId, password);
user = proxy.createUser(user);

// set the status of the new user to Invalid; once the user is
// authenticated, set the status to PendingActivation; after the
// user successfully completes registration, set the status to Valid
proxy.setUserStatus(user.CustomerId, (int)UserStatus.Invalid);
}

// save the user record in the session for later reference
AppSessionData sessionData = AppSessionData.GetInstance(Session);

sessionData.CurrentUser = user;

```

For further details, see the sample applications in [Section 3.15, "ASP.NET Sample Applications."](#)

3.7 User Logins and Transactions

Oracle Adaptive Access Manager provides APIs to capture user login information, user login status, and other user session attributes to determine device and location information. Oracle Adaptive Access Manager also provides APIs to collect transaction details.

The following code sample illustrates the use of this API:

```

// record a user login attempt in OAAM
string  requestId      = sessionData.RequestId;
string  remoteIPAddr   = Request.UserHostAddress;
string  remoteHost     = Request.UserHostName;
bool    isFlashRequest = Request.Params["client"].Equals("vfc");
string  secureCookie   = (Request.Cookies["vsc"] != null)
                        ? Request.Cookies["vsc"].Value : null;
string  digitalCookie  = isFlashRequest
                        ? Request.Params["v"] : null;
object[] browserFpInfo = HttpUtil.GetBrowserFingerprint();
object[] flashFpInfo   = HttpUtil.GetFlashFingerprint();

int browserFingerprintType =
    browserFpInfo == null ? 0 : (int) browserFpInfo [0];
string browserFingerprint =
    browserFpInfo == null ? "" : (string) browserFpInfo [1];
int flashFingerprintType =
    flashFpInfo == null ? 0 : (int) flashFpInfo[0];
string flashFingerprint =
    flashFpInfo == null ? "" : (string) flashFpInfo[1];

// if user name and password have been validated by now, set the status
// to the appropriate value, such as success, wrong_password, or invalid_user
int status = statusEnum.getElementValue("success");

// if user name and password have not yet been validated, set the status to
// pending; after validation is done call updateLog to update status
int status = statusEnum.getElementValue("pending");

// Call updateLog to record the user login attempt

```

```
CookieSet cs = proxy.updateLog(requestId, remoteIPAddr, remoteHost,
    secureCookie, digitalCookie, user.CustomerGroupId,
    user.CustomerId, user.LoginId, false,
    status, ClientTypeEnum.Normal,
    "1.0", browserFingerPrintType, browserFingerPrint,
    flashFingerPrintType, flashFingerPrint);

// Update secure cookie in the browser with the new value from OAAM
if (cs != null)
{
    HttpUtil.UpdateSecureCookie(Response, cs);
}
```

3.8 Rules Engine

The Rules Engine is the component of Oracle Adaptive Access Manager used to enforce policies. Based on a calling context, the Rules Engine evaluates policies and provides the results of those evaluations. Policies are configured by the administrator; for details on policy configuration, see "Creating Policies" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

The following code sample illustrates the use of APIs to invoke the Rules Engine after a user has been authorized and to process the rule evaluation result:

```
AppSessionData sessionData = AppSessionData.GetInstance(Session);
IBharosaProxy proxy = BharosaClientFactory.getProxyInstance();
UserDefEnumFactory factory = UserDefEnumFactory.getInstance();
UserDefEnum profileTypeEnum = factory.getEnum("profile.type.enum");

string requestId = sessionData.RequestId;
BharosaStringList profileTypes = new BharosaStringList();
BharosaStringTable contextList = new BharosaStringTable();

int postAuthType = profileTypeEnum.getElementValue("postauth");

profileTypes.Add(postAuthType.ToString());

// Run postauth rules
VCryptRulesResult res = proxy.processRules(requestId,
    profileTypes, contextList);

// process the rule result
if (StringUtil.EqualsIgnoreCase(res.Result, "Allow"))
{
    // Allow the user login
}
else if (StringUtil.EqualsIgnoreCase(res.Result, "Block"))
{
    // Block the user login
}
else if (res.Result.StartsWith("Challenge"))
{
    // Take the user through challenge question flow
}
else if (res.Result.StartsWith("RegisterUser"))
{
    // Take the user through registration flow
}
```


3.8.1 Device ID

In addition to delivering the rules result, the Rules Engine can return a device ID, an internal Oracle Adaptive Access Manager identifier for the device used for this login session.

The following sample code illustrates how to get the device ID:

```
VCryptRulesResult rulesResult = proxy.processRules ...);

If (!rulesResult.Response.IsSuccess) {
    BharosaTrace.Error("Error running rules " + rulesResult.Response.ErrorMessage);
}
Long deviceId = rulesResult.DeviceId;
```

Important: The code shown assumes that:

- You are using Oracle Adaptive Access Manager 10.1.4.5 or above
- You have set the property `bharosa.tracker.send.deviceId` to true in Oracle Adaptive Access Manager:

```
bharosa.tracker.send.deviceId=true
```

3.8.2 Creating and Updating Bulk Transactions

The `IBharosaProxy.createTransactions()` method can be used to create bulk transactions, as illustrated in the following call:

```
VCrypResponse[] createTransactions(TransactionCreateRequestData[]
transactionCreateRequestData);
```

The `IBharosaProxy.updateTransactions()` method can be used to update bulk transactions, as illustrated in the following call:

```
VCrypResponse[] updateTransactions(TransactionUpdateRequestData[]
transactionUpdateRequestData);
```

3.9 Validating a User with Challenge Questions

Oracle Adaptive Access Manager can challenge a user with pre-registered questions and match user answers with pre-registered answers during high-risk or suspicious scenarios.

Typically, a user is asked to choose questions from a given set and provide answers for them, all of which are then registered. When the user is challenged with one of these questions, he must supply the correct answer, that is, one that matches the answer he registered.

The following sample code illustrates the calls to register questions and answers and challenge the user:

```
// Retrieve a question-pickset, containing groups of questions from
// which the user would pick one question from each group for
// registration
VCryptQuestionList[] groups = proxy.getSignOnQuestions(
    user.CustomerId);

// See the sample application at the end of this chapter
// for details on displaying the questions in the UI and processing the user input
// Here, we assume that the q's and a's are in the question object
```

```
// Register the questions and answers with OAAM
VCryptResponse response = proxy.addQuestions(
    user.CustomerId, questions);

// Retrieve the question to challenge the user
VCryptQuestion secretQuestion = proxy.getSecretQuestion(
    user.CustomerId);

// Create QuestionPad authenticator to display the question text.
// See the sample application at the end of this chapter for details;
// Here, we assume that the user entered an answer stored in the string answer

// Validate the user entered answer
VCryptAuthResult res = proxy.authenticateQuestion(customerId, answer);

bool isValid = (res != null && res.ResultCode == 0);
```

For further details, see the sample applications in [Section 3.15, "ASP.NET Sample Applications."](#)

3.10 Resetting Challenge Failure Counters

Oracle Adaptive Access Manager records the number of wrong answers to the questions posed to the user in the failure counters. Failure counters are used to enforce a lock. The API includes a method, `resetChallengeFailureCounters()`, to reset the failure counters for a given user or user and question combination.

If a Question ID is specified (i.e. `questionId != BharosaGlobals.LongNull`), in the call, only the failure counters associated with that question are reset; if no Question ID is specified, the failure counters for all registered questions of the user are reset.

The following sample code illustrates a call to reset failure counters:

```
VCryptResponse resetChallengeFailureCounters(String requestId,
    String customerId, long questionId);
```

3.11 Virtual Authentication Devices

This section describes the creation and use of virtual authentication devices in ASP.NET applications in the following subsections:

- [Creating a Virtual Authentication Device](#)
- [Embedding a Virtual Authentication Device in a Web Page](#)
- [Validating User Input with a Virtual Authentication Device](#)

3.11.1 Creating a Virtual Authentication Device

To create a virtual authentication device, use the method, `BharosaClient.getAuthentiPad()`, as illustrated in the following sample code:

```
IBharosaClient client = BharosaClientFactory.getClientInstance();

String padName = "passwordPad";

if (! IsPostBack)
{
    AuthentiPadType padType = AuthentiPadType.TYPE_ALPHANUMERICPAD;
```

```

String          bgFile      = proxy.getImage(user.CustomerId);
String          captionText = proxy.getCaption(user.CustomerId);
String          frameFile   = BharosaConfig.get(
"bharosa.authentipad.alphanumeric.frame.file",
"alphanumpad_bg/kp_v2_frame_nologo.png");

AuthentiPad authPad = client.getAuthentiPad(padType, padName,
                                           frameFile, bgFile,
                                           captionText, false,
                                           true, true);

// save the authenticator object in sessData: it will be needed
// in GetImage.aspx.cs to generate the authenticator image, and
// while decoding the user input
sessionData[padName] = authPad;
}

```

3.11.2 Embedding a Virtual Authentication Device in a Web Page

To display a virtual authentication device properly, such as the one created in the previous section, both the .ASPX file and the code-behind file need to be updated.

To update these files, proceed as follows:

1. Include the JavaScript `bharosa_web/js/bharosa_pad.js` in the ASPX file.
2. Create a label in the ASPX file where the virtual authentication device is to be displayed:

```
<asp:Label ID="authenticator" runat="server"></asp:Label>
```

3. Generate the HTML in the code-behind file from the virtual authentication device object and assign it to the label:

```
this.authenticator.Text = client.getAuthentiPadHTML(authPad, false, false);
```

3.11.3 Validating User Input with a Virtual Authentication Device

The input that a user supplies to a virtual authentication device is posted to the application in the HTTP parameter named `padName + "DataField"`. This input should be decoded using the virtual authentication device as illustrated in the following sample code:

```

if (IsPostBack)
{
    AuthentiPad authPad      = sessionData[padName];
    String        encodedPasswd = Request.Params[padName + "DataField"];
    String        passwd       = authPad.decodeInput(encodedPasswd);

    // continue to validate the password
}

```

3.12 Specifying Credentials to the Oracle Adaptive Access Manager SOAP Server

The credentials to access the Oracle Adaptive Access Manager SOAP Server can be specified in one of the following ways:

- By adding the following settings to application `web.config` file:

```
<appSettings>
  <add key="BharosaSOAPUser" value="soapUser" />
  <add key="BharosaSOAPPASSWORD" value="soapUserPassword" />
  <add key="BharosaSOAPDomain" value="soapUserDomain" />
</appSettings>
```

- By adding the following properties to one of the application properties files:

```
BharosaSOAPUser=soapUser
BharosaSOAPPASSWORD=soapUserPassword
BharosaSOAPDomain=soapUserDomain
```

Note: When specifying SOAP credentials in this way, you can use either clear text or an encrypted string for a value (typically, for the value of a password)

3.13 Encrypting Property Values

A property value specified in a properties file can be encrypted using the command-line utility `BharosaUtils.exe` included in the Oracle Adaptive Access Manager .NET SDK.

An encryption key (arbitrarily selected by the user) is required to encrypt and decrypt values. This key is available to Oracle Adaptive Access Manager .NET API through the property `bharosa.cipher.client.key`, which must be set in one of the application properties files.

`BharosaUtil.exe` prompts the user to enter the encryption key and a value, and the encrypted value is output to the console. The following run of the utility illustrates how to encrypt a string:

```
C:\> BharosaUtil.exe -enc
Enter key (min 14 characters len): <your key>
Enter key again: <your key>
Enter text to be encrypted: <string to encryp>
Enter text to be encrypted again: <string to encryp>
vCCKC19d14a39hQSKSirXSiWfgbaVG5SKIg==
```

3.14 Tracing Messages

The Oracle Adaptive Access Manager .NET API allows to print trace messages of various levels using diagnostics switches in `web.config`. The trace messages can be saved to a file by configuring the appropriate listeners.

The following `web.config` file sample shows the configuration of switches and a listener that writes trace messages to a file:

```
<system.diagnostics>
  <switches>
    <add name="debug" value="0" />
    <add name="info" value="0" />
    <add name="soap" value="0" />
    <add name="perf" value="0" />
    <add name="warning" value="1" />
    <add name="error" value="1" />
    <add name="traceTimestamp" value="1" />
    <add name="traceThreadId" value="1" />
  </switches>
  <trace autoflush="true" indentsize="2">
```

```

<listeners>
  <add name="BharosaTraceListener"
        type="System.Diagnostics.TextWriterTraceListener, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"
        initializeData="BharosaTrace.log" />
</listeners>
</trace>
</system.diagnostics>

```

3.15 ASP.NET Sample Applications

The SDK includes ASP.NET applications that illustrate the integration of several Oracle Adaptive Access Manager features with a simple ASP.NET Web application.

The names and brief description of these applications is shown in [Table 3–2](#).

Table 3–2 *ASP.NET Applications Samples*

Application name	Description
SampleWebApp	Plain ASP.NET application with no Oracle Adaptive Access Manager integration. It is the application to be integrated and provided so that the developer can identify the incremental changes to integrate it with various Oracle Adaptive Access Manager features such as the virtual authentication devices and KBA.
SampleWebAppWithTracker	This application illustrates the integration of the Oracle Adaptive Access Manager Risk Engine with SampleWebApp.
SampleWebAppWithAuthTracker	This application illustrates the integration of the Oracle Adaptive Access Manager Risk Engine and virtual authentication device with SampleWebApp.
SampleWebAppWithKBATracker	This application illustrates the integration of the Oracle Adaptive Access Manager Risk Engine and KBA with SampleWebApp.

The source code for each of these applications is packaged in a separate directory, and the Visual Studio Solution files for each of them are located in the root directory. The solutions file `SampleWebApps` can be loaded to view all applications together in a development environment.

Note: The three sample integrations require that the archive `Bharosa_SDK_DotNet2.0.zip` be extracted to the root directory of the corresponding Web application sample.

When setting up an environment to run the sample applications, make sure that:

- The URL of the service that communicates via SOAP with Oracle Adaptive Access Manager is correctly set in the application's `web.config` file.

A sample setting is shown:

```

<appSettings>
  <add key="BharosaSOAPURL"
        value="http://localhost:9090/mySOAPservices/services" />
</appSettings>

```

- Oracle Adaptive Access Manager used with these applications is loaded with the policies contained in the file `models.zip`.

- Both the Oracle Adaptive Access Manager SOAP Server and the Web applications are configured to look for virtual authentication device images in the same directory paths. To this purpose, set the following properties:

```
bharosa.authentipad.full.background.dirlist
bharosa.authentipad.alphanumeric.background.dirlist
bharosa.authentipad.numeric.background.dirlist
bharosa.authentipad.textpad.background.dirlist
bharosa.authentipad.textpadreset.background.dirlist
bharosa.authentipad.questionpad.background.dirlist
```

3.15.1 SampleWebApp

This application contains the following pages:

- LoginPage.aspx
 - Collects the user name and password using a simple HTML form.
 - Validates the user login information.
 - Depending upon the result of the validation, redirects the user to either `Success.aspx` or to `LoginPage.aspx`.
- Success.aspx
 - Displays the message "Successfully logged in" and allows the user to log out.
- LogoutPage.aspx
 - Logs out the session and redirects the user to `LoginPage.aspx`.

3.15.2 SampleWebAppWithTracker

This application integrates `SampleWebApp` with Oracle Adaptive Access Manager Risk Engine and contains the following pages:

- LoginPage.aspx
 - Collects the user name and password using a simple HTML form.
 - Saves the login and password in the session.
 - Redirects the user to `LoginJumpPage.aspx` to collect the flash fingerprint of the user device.
- LoginJumpPage.aspx
 - Loads the user from Oracle Adaptive Access Manager by calling `AppUtil.InitUser`. If the user record is not found, it creates a new user record.
 - Returns the HTML to load the flash object `bharosa_web/flash/bharosa.swf` in the browser. The flash object calls `CookieManager.aspx` (included in the SDK package) with the flash fingerprint details. `CookieManager.aspx` records the fingerprint in Oracle Adaptive Access Manager and sets a flash cookie on the user's device.
 - After getting the flash cookie from Oracle Adaptive Access Manager, it redirects the user to `LoginHandlerPage.aspx`.
- LoginHandlerPage.aspx

- Records the user login attempt with Oracle Adaptive Access Manager by calling `AppUtil.InitTracker`.
 - Validates the user login information.
 - Updates Oracle Adaptive Access Manager with the password validation status (such as success, wrong user, wrong password, or disabled user) by calling `AppUtil.UpdateAuthStatus`.
 - If password validation succeeds, it runs the post-authentication rules by calling `AppUtil.RunPostAuthRules`.
 - If the post-authentication rules return `block`, it updates Oracle Adaptive Access Manager and blocks the user login.
 - Depending on the validation and the rules results, it redirects the user to either `Success.aspx` or to `LoginPage.aspx`.
- `Success.aspx`
 - Displays the message "Successfully logged in" and allows the user to log out.
 - `LogoutPage.aspx`
 - Logs out the session and redirects the user to `LoginPage.aspx`.

3.15.3 SampleWebAppWithAuthTracker

This application integrates `SampleWebApp` with the Oracle Adaptive Access Manager Risk Engine and a virtual authentication device, and it contains the following pages:

- `LoginPage.aspx`
 - Collects the user name and password using a simple HTML form.
 - Saves the login and password in the session.
 - Redirects the user to `LoginJumpPage.aspx` to collect the flash fingerprint of the user device.
- `LoginJumpPage.aspx`
 - Loads the user from Oracle Adaptive Access Manager by calling `AppUtil.InitUser`. If the user record is not found, it creates a new user record.
 - Returns the HTML to load the flash object `bharosa_web/flash/bharosa.swf` in the browser. The flash object calls `CookieManager.aspx` (included in the SDK package) with the flash fingerprint details. `CookieManager.aspx` records the fingerprint in Oracle Adaptive Access Manager and sets a flash cookie on the user's device.
 - After getting the flash cookie from Oracle Adaptive Access Manager, it redirects the user to `LoginHandlerPage.aspx`.
- `LoginHandlerPage.aspx`
 - Records the user login attempt with Oracle Adaptive Access Manager by calling `AppUtil.InitTracker`.
 - Redirects the user to `PasswordPage.aspx` to collect the user password using a virtual authentication device.
- `PasswordPage.aspx`

On Load:

1. Sets the session authentication status to "Pending".
2. Runs pre-authentication rules by calling `AppUtil.RunPreAuthRules`.
3. If the pre-authentication rules return `block`, it updates Oracle Adaptive Access Manager and blocks the user login.
4. If the pre-authentication rules return `allow`, it runs another set of rules to determine the virtual authentication device to use for this user, by calling `AppUtil.RunAuthentiPadRules`.
5. Creates the appropriate virtual authentication device by calling `AppUtil.CreateAuthentiPad` and renders the virtual authentication device into HTML by using `AppUtil.getAuthentiPadHTML`. The virtual authentication device HTML would fetch the virtual authentication device image by calling `GetImage.aspx`.
6. Stores the virtual authentication device in the session for later use during image generation and password decoding.

OnPostBack

1. Decodes the user password using the stored virtual authentication device.
 2. Validates the user login and password information.
 3. Updates Oracle Adaptive Access Manager with the password validation status (such as success, wrong user, wrong password, or disabled user) by calling `AppUtil.UpdateAuthStatus`.
 4. If the password validation succeeds, it runs post-authentication rules by calling `AppUtil.RunPostAuthRules`.
 5. If the post-authentication rules return `block`, it updates Oracle Adaptive Access Manager and blocks the user login.
 6. Depending on the validation and the rules results, it redirects the user to either `Success.aspx` or to `LoginPage.aspx`.
- `Success.aspx`
 - Displays the message "Successfully logged in" and allows the user to log out.
 - `LogoutPage.aspx`
 - Logs out the session and redirects the user to `LoginPage.aspx`.

3.15.4 SampleWebAppWithKBATracker

This application illustrates the integration of `SampleWebApp` with the Oracle Adaptive Access Manager Risk Engine and KBA, and it contains the following pages:

- `LoginPage.aspx`
 - Collects the user name and password using a simple HTML form.
 - Saves the login and password in the session.
 - Redirects the user to `LoginJumpPage.aspx` to collect the flash fingerprint of the user device.
- `LoginJumpPage.aspx`
 - Loads the user from Oracle Adaptive Access Manager by calling `AppUtil.InitUser`. If the user record is not found, it creates a new user record.

- Returns the HTML to load the flash object `bharosa_web/flash/bharosa.swf` in the browser. The flash object calls `CookieManager.aspx` (included in the SDK package) with the flash fingerprint details. `CookieManager.aspx` records the fingerprint in Oracle Adaptive Access Manager and sets a flash cookie on the user's device.
- After getting the flash cookie from Oracle Adaptive Access Manager, it redirects the user to `LoginHandlerPage.aspx`.
- **LoginHandlerPage.aspx**
 - Records the user login attempt with Oracle Adaptive Access Manager by calling `AppUtil.InitTracker`.
 - Redirects the user to `PasswordPage.aspx` to collect the user password using a virtual authentication device.
- **PasswordPage.aspx**

On Load:

 1. Sets the session authentication status to "Pending".
 2. Runs pre-authentication rules by calling `AppUtil.RunPreAuthRules`.
 3. If the pre-authentication rules return `block`, it updates Oracle Adaptive Access Manager and blocks the user login.
 4. If the pre-authentication rules return `allow`, it runs another set of rules to determine the virtual authentication device to use for this user, by calling `AppUtil.RunAuthentiPadRules`.
 5. Creates the appropriate virtual authentication device by calling `AppUtil.CreateAuthentiPad` and renders the virtual authentication device into HTML by using `AppUtil.getAuthentiPadHTML`. The virtual authentication device HTML would fetch the virtual authentication device image by calling `GetImage.aspx`.
 6. Stores the virtual authentication device in the session for later use during image generation and password decoding.

On PostBack:

 1. Decodes the user password using the stored virtual authentication device.
 2. Validates the user login and password information.
 3. Updates Oracle Adaptive Access Manager with the password validation status (such as success, wrong user, wrong password, or disabled user) by calling `AppUtil.UpdateAuthStatus`.
 4. If the password validation fails, it redirects the user to `LoginPage.aspx`.
 5. If password validation succeeds, it runs post-authentication rules by calling `AppUtil.RunPostAuthRules`.
 6. The user is directed through different flows, as shown in the table, depending on the action from post-authenticator rules result:

Post-Authentication Action	Target URL
Block	LoginPage.aspx
Allow	Success.aspx
ChallengeUser	ChallengeUser.aspx

Post-Authentication Action	Target URL
RegisterQuestions	RegisterQuestionsPage.aspx
RegisterUser	PersonalizationPage.aspx
RegisterUserOptional	PersonalizationPage.aspx

- PersonalizationPage.aspx
 - Introduces the user to device personalization by explaining the steps to create a new user security profile.
 - If the post authentication rule returns `RegistrationOptional`, the user is allowed to skip the registration process by clicking **Skip** and to proceed to the `Success.aspx` page directly.
 - If registration is not optional, the user must register by clicking **Continue** to proceed to `RegisterImagePhrase.aspx`.
- RegisterImagePhrase.aspx
 - Allows the user to customize the randomly generated background image, a caption, and the type of security device to be used during authentication.
 - Assigns a new background image and caption by calling `AppUtil.AssignNewImageAndCaption`.
 - Assigns the type of security device by calling `AppUtil.SetAuthMode`.
- RegisterQuestionsPage.aspx
 - Displays a set of questions by calling `proxy.getSignOnQuestions`.
 - Registers the selected questions and answers the user enters.
- ChallengeUser.aspx
 - Challenges the user by displaying `QuestionPad` with one of the registered questions.
 - The answer is validated by calling `proxy.authenticateQuestion` and the result is updated by calling `AppUtil.UpdateAuthStatus`.
 - If the answer does not match the registered answer, it calls `AppUtil.RunChallengeUserRules` and, based on its result, it allows the user to reenter the answer or redirects the user to the block page after updating the block status. The number of attempts that a user gets to answer a question correctly is set by the security administrator.
 - If the answer matches the registered answer, it redirects the user to `Success.aspx`.
- Success.aspx
 - Displays the message "Successfully logged in" and allows the user to log out.
- LogoutPage.aspx
 - Logs out the session and redirects the user to `LoginPage.aspx`.

Integrating Native Java Applications

This chapter explains how to integrate Java applications with Oracle Adaptive Access Manager Server using the Oracle Adaptive Access Manager Java API. This integration is supported for applications written in Java 1.4 or higher.

This section contains the following sections:

- [About the Oracle Adaptive Access Manager Shared Library](#)
- [About VCryptResponse](#)
- [Oracle Adaptive Access Manager APIs](#)
- [Rules Engine](#)
- [Customer Care](#)

4.1 About the Oracle Adaptive Access Manager Shared Library

The Oracle Adaptive Access Manager Shared Library is the Java SDK for integrating with Oracle Adaptive Access Manager. This is deployed in OAAM Server when you install the Oracle Adaptive Access Manager 11g applications. In OAAM Server contains this shared library, `oracle.oaam.libs`.

4.1.1 Using Oracle Adaptive Access Manager Shared Library in Applications

To use the Oracle Adaptive Access Manager Shared Library, you must refer to the shared library by adding the following entry to your WebLogic deployment descriptor file, `weblogic.xml`:

```
<library-ref>
  <library-name>oracle.oaam.libs</library-name>
</library-ref>
```

4.1.2 Customizing/Extending/Overriding Oracle Adaptive Access Manager Properties

To override any Oracle Adaptive Access Manager properties or extend Oracle Adaptive Access Manager enumerations, add those properties and enumerations to `bharosa_server.properties` and place that file in `WEB-INF\classes` or `WEB-INF\classes\bharosa_properties` file of the application.

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, refer to [Chapter 12, "Customizing Oracle Adaptive Access Manager."](#)

4.2 About VCryptResponse

VCryptResponse contains information about the status of the processing. It contains useful information if the status of the processing was "Success" (`isSuccess`). If there were an error, it also contains error codes. It can also contain other payload information in the form of extended data maps. You can use these features of VCryptResponse depending on your requirements for integration.

4.3 Oracle Adaptive Access Manager APIs

Oracle Adaptive Access Manager provides APIs to:

- Collect and track information from the client application
- Capture user login information, user login status, and various attributes of the user session to determine device and location information
- Collect transaction details

For descriptions of all authentication scenarios and typical flows, see [Chapter 2, "Natively Integrating with Oracle Adaptive Access Manager."](#)

The following sections provide details of the following important methods:

- [handleTrackerRequest](#)
- [createTransaction](#)
- [updateTransaction](#)
- [handleTransactionLog](#)
- [updateTransactionStatus](#)
- [updateLog](#)
- [updateAuthStatus](#)
- [processPatternAnalysis](#)
- [markDeviceSafe](#)
- [IsDeviceMarkedSafe](#)
- [clearSafeDeviceList](#)

4.3.1 handleTrackerRequest

Captures fingerprint details and identify the device; it may also capture fingerprint details for a given request time, which can be in the past.

```
public CookieSet handleTrackerRequest(String requestId,
                                     String remoteIPAddr,
                                     String remoteHost,
                                     String secureCookie,
                                     int secureClientType,
                                     String secureClientVersion,
                                     String digitalCookie,
                                     int digitalClientType,
                                     String digitalClientVersion,
                                     int fingerPrintType,
                                     String fingerPrint,
                                     int fingerPrintType2,
                                     String fingerPrint2);
```

```

public CookieSet handleTrackerRequest(String requestId,
                                     Date requestTime,
                                     String remoteIPAddr,
                                     String remoteHost,
                                     String secureCookie,
                                     int secureClientType,
                                     String secureClientVersion,
                                     String digitalSigCookie,
                                     int digitalClientType,
                                     String digitalClientVersion,
                                     int fingerprintType,
                                     String fingerprint,
                                     int fingerprintType2,
                                     String fingerprint2);

```

The returned object has functions to access its contents

They are:

```

public String getFlashCookie()
public String getSecureCookie()
public String getRequestId()
public VCryptResponse getVCryptResponse()

```

Table 4–1 *handleTrackerRequest Parameters*

Parameter	Description
requestId	The login session ID; this is the ID that should be used in all API calls for the login session
remoteIPAddr	The IP from where the request came; extracted from the HTTP request
remoteHost	The host name from where the request came; optional
secureCookie	The secure cookie; passed only if it is received from a browser
secureClientType	An enumeration value that identifies the type of client used for authentication
secureClientVersion	The version of the client; optional
digitalCookie	The digital signature cookie; can be the flash cookie; passed only if it is sent by a browser
digitalClientType	The digital client type that specifies the type of flash client used; if not available, use the value 0
digitalClientVersion	The version of the digital client; can be the version of the flash client
fingerprintType	The type of fingerprinting; defined in the properties file
fingerprint	The fingerprint; if it describes browser characteristics, then the header is parsed into this string; it represents the browser header information
fingerprintType2	Used in case the same request has multiple fingerprints; defined in the properties file; optional
fingerprint2	The second fingerprint value; optional
requestTime	The time at which the request was made

4.3.2 createTransaction

Creates a new transaction.

```

public VCryptResponse createTransaction(
    TransactionCreateRequestData transactionCreateRequestData);

```

Table 4–2 createTransaction Parameter and Returned Value

Parameter	Description
TransactionCreateRequestData	<p>The object to create a new transaction; it throws the exception BharosaException if it fails validation.</p> <p>The structure of this object is as follows:</p> <ul style="list-style-type: none"> ▪ requestId, identifies the user session; required ▪ requestTime, the time of the request; can be null; if null, the server uses the current time ▪ transactionKey, the key to the transaction definition; used to create a transaction definition; required ▪ externalTransactionId, handle to the transaction; optional if there is external transaction ID ▪ status, the transaction status; can be null
VCryptResponse	The response object; make sure to check isSuccess() before obtaining the transaction ID with the method getTransactionResponse()

4.3.3 updateTransaction

Updates a previously created transaction.

```
public VCryptResponse updateTransaction(
    Transaction UpdateRequestData transactionUpdateRequest Data);
```

Table 4–3 updateTransaction Parameter and Returned Value

Parameter	Description
TransactionUpdateRequestData	<p>The object to update a transaction; a handle to the transaction to be updated is either the transaction ID returned by the method createTransaction, or the external transaction ID passed to the method createTrasnaction. it throws the exception BharosaException if it fails validation.</p> <p>The structure of this object is as follows:</p> <ul style="list-style-type: none"> ▪ requestId, identifies the user session; required ▪ requestTime, the time of the request; can be null; if null, the server uses the current time ▪ transactionId ID, the ID returned by a previous call to createTransaction ▪ status, the transaction status ▪ analyzePatterns, Boolean to indicate if pattern processing should be performed. When the value is passed in as "true," the pattern processing is performed for the transaction if the "resultStatus" value is "success." ▪ externalTransactionId, the external transaction ID that was passed to createTransaction when the transaction was created
VCryptResponse	The response object; make sure to check isSuccess() before obtaining the transaction ID with the method getTransactionResponse()

4.3.4 handleTransactionLog

Captures transaction details.

Note: Deprecated as of 10.1.4.5.1; instead, use the method [createTransaction](#).

```
public VCryptResponse handleTransactionLog(String requestId, Map[] contextMap);
```

```
public VCryptResponse handleTransactionLog(String requestId, Date requestTime,
Map[] contextMap);
```

```
public VCryptResponse handleTransactionLog(String requestId, Date
requestTime,Integer status, Map[] contextMap);
```

Table 4–4 *handleTransactionLog Parameters*

Parameter	Description
requestId	The login session ID; this is the ID that should be used in all API calls for the login session
requestTime	The time at which the request was made
contextMap	An array of contextMaps; multiple transactions can be created with a single call; it expects to find a transactionType key in each context map of the array
status	The transaction status

4.3.5 updateTransactionStatus

Updates a transaction status and, if appropriate, triggers the data pattern processing.

Note: Deprecated as of 10.1.4.5.1; instead, use the method [updateTransaction](#).

```
public VCryptResponse updateTransactionStatus(String requestId, long
transactionId, int status);
```

```
public VCryptResponse updateTransactionStatus(String requestId, Date requestTime,
long transactionId, int status);
```

```
public VCryptResponse updateTransactionStatus(String requestId, long
transactionId, int status, Map[] contextMap);
```

```
public VCryptResponse updateTransactionStatus(String requestId, Date requestTime,
long transactionId, int status, Map[] contextMap);
```

```
public VCryptResponse updateTransactionStatus(String requestId, long
transactionId, int status, boolean analyzePatterns);
```

```
public VCryptResponse updateTransactionStatus(String requestId, Date requestTime,
long transactionId, int status, Map[] contextMap, boolean analyzePatterns);
```

Table 4–5 *updateTransactionStatus Parameters*

Parameter	Description
requestId	The login session ID; this is the ID that should be used in all API calls for the login session
requestTime	The time at which the request was made
contextMap	An array of contextMaps; multiple transactions can be created with a single call; it expects to find a transactionType key in each context map of the array

Table 4–5 (Cont.) updateTransactionStatus Parameters

Parameter	Description
Status	The transaction status
transactionId	The ID of the transaction whose status to update; if null, it uses the last transaction in the given session
analyzePatterns	Boolean to indicate if pattern processing should be performed. When the value is passed in as "true," the pattern processing is performed for the transaction if the "resultStatus" value is "success."

4.3.6 updateLog

Updates the user log and, if required, creates a CookieSet.

```
public CookieSet updateLog(String requestId,
                          String remoteIPAddr,
                          String remoteHost,
                          String secureCookie,
                          String digitalCookie,
                          String groupId,
                          String userId,
                          String loginId,
                          boolean isSecure,
                          int result,
                          int clientType,
                          String clientVersion,
                          int fingerPrintType,
                          String fingerPrint,
                          int digFingerPrintType,
                          String digFingerPrint);
```

```
public CookieSet updateLog(String requestId,
                          Date requestTime,
                          String remoteIPAddr,
                          String remoteHost,
                          String secureCookie,
                          String digitalCookie,
                          String groupId,
                          String userId,
                          String loginId,
                          boolean isSecure,
                          int result,
                          int clientType,
                          String clientVersion,
                          int fingerPrintType,
                          String fingerPrint,
                          int fingerPrintType2,
                          String fingerPrint2);
```

Table 4–6 updateLog Parameters

Parameter	Description
requestId	The login session ID; this is the ID that should be used in all API calls for the login session
remoteIPAddr	The IP from where the request came; extracted from the HTTP request
remoteHost	The host name from where the request came; optional

Table 4–6 (Cont.) updateLog Parameters

Parameter	Description
secureCookie	The secure cookie; passed only if it is received from a browser
digitalCookie	The digital signature cookie; can be the flash cookie; passed only if it is sent by a browser
groupId	The ID of the group this user belongs to
userId	The user ID; this is the primary ID key for the user; for invalid users, it is null
loginId	The ID used by the user to login in; required
isSecure	A Boolean indicating whether this node is secure and can be registered; it also indicates that the login is from a secure or registered device; if there is no concept of device, then set to false
result	A value of the user-defined enumeration <code>auth.status.enum</code>
clientType	An enumeration value indicating the client type used for authentication
clientVersion	The version of the client; optional
fingerprintType	The type of fingerprinting; defined in the properties file
fingerprint	The fingerprint; if it describes browser characteristics, then the header is parsed into this string; it represents the browser header information
digFingerprintType	The type of the digital fingerprinting
digFingerprint	The digital fingerprint
requestTime	The time at which the request was made
fingerprintType2	Used in case the same request has multiple fingerprints; defined in the properties file; optional
fingerprint2	The second fingerprint value; optional

4.3.7 getUserByLoginId

Return the user details without the password and pin for the given customer and group.

```
public VCryptAuthUser getUserByLoginId(String loginId, String groupName);
```

Table 4–7 getUserByLoginId

Parameter	Description
loginId	The ID used by the user to login in
groupName	The group name

4.3.8 updateAuthStatus

Updates the user authentication status and, if appropriate, it triggers pattern data processing. This method must be called when there is a change in the user authentication status; make sure that, before calling `updateAuthStatus`, the application calls [updateLog](#).

The list of authentication status values are specified in the user-defined enumeration `auth.status.enum`; you can add or remove items to this enumeration, as appropriate to your application, but only values of this enumeration can be used to identify an authentication status.

The following scenarios describe alternative ways to handle updating a user login (authentication) status:

- Pass the login status in the `updateLog` call; this scenario avoids calling `updateAuthStatus` altogether.
- Allow the user to login before setting the login status; in this scenario, first pass status "pending" in the `updateLog` call, then process the login data, and then pass the appropriate status in the `updateAuthStatus` call.
- If your application flow includes challenging the user, then first set the status to "pending", then pose the challenge questions, and then, depending on the answers, reset the status to "success" or "wrong_answer".
- Typically, there is no need to call `updateAuthStatus` after invoking the rules engine, since this engine includes setting the authentication status as part of running the rules.

```
public VCryptResponse updateAuthStatus(String requestID,
                                       int resultStatus,
                                       int clientType,
                                       String clientVersion);

public VCryptResponse updateAuthStatus(String requestID,
                                       Date requestTime,
                                       int resultStatus,
                                       int clientType,
                                       String clientVersion);

public boolean isSuccess() {
    return this.success;
}

public String getResponseCode() {
    return this.responseCode;
}

public String getErrorMessage() {
    return this.errorMessage == null ? "" : this.errorMessage;
}

public String getErrorMessageRBKey() {
    return this.errorMessageRBKey;
}

public String[] getErrorMessageParams() {
    return this.errorMessageParams;
}

public Date getTimeStamp() {
    return this.timeStamp;
}

public String getServer() {
    return this.server;
}

public Map getExtendedDataMap() {
    return (extendedDataMap == null) ?
        Collections.EMPTY_MAP :
```

```

Collections.unmodifiableMap(extendedDataMap);
}

public String getExtendedMap(String key) {
Map map = getExtendedDataMap();
if (map != null) {
return (String) map.get(key);
}
return null;
}

public String getSessionId() {
return sessionId;
}

public TransactionResponse getTransactionResponse() {
return transactionResponse;
}

public VCryptResponse updateAuthStatus(String requestID,
int resultStatus,
int clientType,
String clientVersion,
boolean analyzePatterns);

public VCryptResponse updateAuthStatus(String requestID,
Date requestTime,
int resultStatus,
int clientType,
String clientVersion,
boolean analyzePatterns);

```

Table 4–8 *updateAuthStatus Parameters*

Parameter	Description
requestId	The login session ID; this is the ID that should be used in all API calls for the login session
requestTime	The time at which the request was made
resultStatus	A value of the user-defined enumeration <code>auth.status.enum</code>
clientType	An enumeration value indicating the client type used for authentication
clientVersion	The version of the client; optional
analyzePatterns	Boolean to indicate if pattern processing should be performed. When the value is passed in as "true," the pattern processing is performed for the transaction if the "resultStatus" value is "success."

4.3.9 processPatternAnalysis

Triggers the data pattern processing.

```

public VCryptResponse processPatternAnalysis(String requestId,
long transactionId,
int status,
String transactionType);

```

Table 4–9 processPatternAnalysis Parameters

Parameter	Description
requestId	The login session ID; this is the ID that should be used in all API calls for the login session
transactionId	The identifier of the transaction. For authentication type of data this is ignored. (It can be passed in as "null"). For pattern processing of transaction data this param is required.
status	A value of the user-defined enumeration <code>auth.status.enum</code> . If the value of the status is the value corresponding to a Success value in the enum, pattern analysis will be performed; otherwise, it will not be performed.
transactionType	Indicates the type of the transaction; must be "auth" for authentication transactions; other transaction type values, such as "bill_payment", can be customized.

4.3.10 markDeviceSafe

Marks the user device as safe.

```
public boolean markDeviceSafe(String requestId, boolean isSafe);
```

Table 4–10 markDeviceSafe Parameters

Parameter	Description
requestId	The login session ID; this is the ID that should be used in all API calls for the login session
isSafe	Indicates whether this user device is safe

4.3.11 IsDeviceMarkedSafe

Returns a value indicating whether the user device associated with a request is safe.

```
public VCryptBooleanResponse IsDeviceMarkedSafe(String requestId);
```

Table 4–11 IsDeviceMarkedSafe Parameters

Parameter	Description
requestId	The login session ID; this is the ID that should be used in all API calls for the login session

4.3.12 clearSafeDeviceList

Clears the user safe device list of the user associated with a request.

```
public VCryptBooleanResponse clearSafeDeviceList(String requestId);
```

Table 4–12 clearSafeDeviceList Parameters

Parameter	Description
requestId	The ID for the login session. The same ID should be used for all the calls to Bharosa API for the login session.

4.4 Rules Engine

The Rules Engine is the part of the Adaptive Risk Manager that enforces policies at checkpoint. Adaptive Risk Manager includes APIs to evaluate policies that return results depending on the calling context.

The following section provides details of the method `processRules` and on how to get the device ID.

4.4.1 processRules

Processes policy sets for the passed checkpoints.

```
public VCryptRulesResult processRules(String requestId, List runtimeTypes, Map
contextMap);
public VCryptRulesResult processRules(String requestId, Date requestTime, List
runtimeTypes, Map contextMap);
```

Table 4–13 processRules Parameters

Parameter	Description
requestId	The login session ID; this is the ID that should be used in all API calls for the login session
runtimeTypes	The list of checkpoints to be evaluated; each checkpoint in this list is evaluated
requestTime	The time at which the request was made
contextMap	A list of key-value pairs identifying the context data; rules in policies can make decisions based on this data

In addition to rule results, the Rules Engine can return a device ID, an internal identifier identical to the user session.

The following code sample illustrates how to get a device ID:

```
VCryptRulesResult rulesResult = new
VCryptRulesEngineImpl().processRules(<params...>);

If (!rulesResult.getVCryptResponse().isSuccess()) {

    Logger.error("Error running rules " +
rulesResult.getVCryptResponse().getErrorMessage());

}

Long deviceId = VCryptRulesResult#getDeviceId();
```

When getting a device ID, make sure that:

- The Oracle Adaptive Access Manager version is 10.1.4.5 or above
- The property `bharosa.tracker.send.deviceId` is set to true, so the device ID can be captured:

```
bharosa.tracker.send.deviceId=true
```

With property `bharosa.tracker.sendLocationData=true` set, location and device data is returned when `processRules` API is called.

```
VCryptRulesResult rulesResult = processRules(/*params*/);
VCryptResponse response = rulesResult.getVCryptResponse();
If (response.isSuccess()) {

    String ipAddress = response.getExtendedMap(VCryptResponse.DATA_REMOTE_IP_
ADDRESS) ;
    String deviceId= response.getExtendedMap(VCryptResponse.DATA_DEVICE_ID) ;
```

```

// if interested in city, state, country
String city = response.getExtendedMap(VCryptResponse.DATA_CITY_NAME) ;
String state = response.getExtendedMap(VCryptResponse.DATA_STATE_NAME) ;
String country = response.getExtendedMap(VCryptResponse.DATA_COUNTRY_NAME) ;
}

```

4.5 Customer Care

Customer Care provides APIs typically used in customer care portals; these APIs do not use audit or access control.

The following sections provide details of the following important methods of this interface:

- [getFinalAuthStatus](#)
- [setTemporaryAllow](#)
- [cancelAllTemporaryAllows](#)
- [resetUser](#)
- [getRulesData](#)
- [getActionCount](#)

4.5.1 getFinalAuthStatus

Returns the final authentication status of a user. The status can be no more than 30-day old.

```
public VCryptIntResponse getFinalAuthStatus(String requestId, String userId);
```

Table 4–14 *getFinalAuthStatus Parameters*

Parameter	Description
requestId	The request ID (used in logging and tracing client requests in case of error)
userId	The ID uniquely identifying the user; cannot be null

4.5.2 setTemporaryAllow

Sets a temporary allow for a user. A temporary allow can override the final rule action.

```
public VCryptResponse setTemporaryAllow(String customerId, int tempAllowType, Date expirationDate);
```

Table 4–15 *setTemporaryAllow Parameters*

Parameter	Description
customerId	The customer ID
tempAllowType	The type of the temporary allow; the user-defined enumeration for this type is <code>customercare.case.tempallow.level.enum</code>
expirationDate	The expiration date, if the <code>tempAllowType</code> is "userset"; otherwise null or empty

4.5.3 cancelAllTemporaryAllows

Cancels all temporary allows that have been set for a customer ID.

```
public VCryptResponse cancelAllTemporatyAllows(String customerId);
```

Table 4–16 *cancelAllTemporaryAllows Parameters*

Parameter	Description
customerId	The customer ID

4.5.4 resetUser

Resets all the profiles that have been set for a customer, including registration, questions, images, and phrases.

```
public VCryptResponse resetUser(String customerId);
```

Table 4–17 *resetUser Parameters*

Parameter	Description
customerId	The customer ID

4.5.5 getRulesData

Returns all rules executed for the given session ID, and provides some information about what rules were triggered.

```
public VCryptSessionRuleData getRulesData(String requestId);
```

Table 4–18 *getRulesData Parameters*

Parameter	Description
requestId	The request ID (used in logging and tracing client requests in case of error)

4.5.6 getActionCount

Gets the number of actions for a given `actionEnumId` from the configured action enumerations.

```
public VCryptIntResponse getActionCount(String requestId, Sting customerId, Integer actionEnumId);
```

Table 4–19 *getActionCount Parameters*

Parameter	Description
requestId	The request ID (used in logging and tracing client requests in case of error)
customerId	The customer ID
actionEnumId	An integer identifying an <code>actionEnum</code> ; required

Part II

Universal Installation Option and Related Integrations

Part II contains the following chapters:

- [Chapter 5, "Oracle Adaptive Access Manager Proxy"](#)
- [Chapter 6, "Configuring OAAM Server"](#)
- [Chapter 7, "Virtual Authentication Device Properties"](#)

Oracle Adaptive Access Manager Proxy

Oracle Adaptive Access Manager's Universal Installation Option (UIO) reverse proxy deployment option offers login risk-based multifactor authentication to Web applications without requiring any change to the application code.

The proxy's main function is to redirect user traffic from the application login flow to the Oracle Adaptive Access Manager login flow.

The Oracle Adaptive Access Manager Proxy is available for the Apache Web server and Microsoft Internet Security and Acceleration (ISA) Server.

This chapter:

- Explains the use and configuration of the Oracle Adaptive Access Manager Proxy.
- Provides instructions for both Microsoft ISA and Apache implementations.

The intended audience is for integrators who configure the Oracle Adaptive Access Manager Proxy to add multifactor authentication to Web applications. An understanding of HTTP request/response paradigm is required to understand the material presented in this document.

The chapter contains the following sections:

- [Introduction](#)
- [Installing Oracle Adaptive Access Manager Proxy for Microsoft ISA](#)
- [Installing Oracle Adaptive Access Manager Proxy for Apache](#)
- [Setting Up Rules and User Groups](#)
- [Setting Up Policies](#)
- [Configuring the Oracle Adaptive Access Manager Proxy](#)
- [Interception Process](#)
- [Configuring Redirection to the Oracle Adaptive Access Manager Server Interface](#)
- [Application Discovery](#)
- [Samples](#)

For information on configuring OAAM Server, the client-facing multifactor authentication Web application specific to the Universal Installation Option deployment, refer to [Chapter 6, "Configuring OAAM Server."](#)

5.1 Introduction

The Introduction section of this chapter contains the following topics:

- [Important Terms](#)
- [Architecture](#)
- [References](#)

5.1.1 Important Terms

For your reference, important terms are defined in this section.

Microsoft ISA

From the Microsoft Web site: "the Internet Security and Acceleration (ISA) Server is the integrated edge security gateway that helps protect IT environments from Internet-based threats while providing users with fast and secure remote access to applications and data."

Universal Installation Option

The Universal Installation Option is the Oracle Adaptive Access Manager integration strategy that does not require any code modification to the protected Web applications. The Universal Installation Option involves placing the Oracle Adaptive Access Manager Proxy in front of the protected Web applications

Proxy

A proxy is a server that services the requests of its clients by forwarding requests to other servers. This chapter is concerned with the Web proxy, where the proxy handles Web Protocols, mainly HTTP.

Forward Proxy

A forward proxy is an intermediate server that sits between the client and the origin server. To get content from the origin server, the client sends a request to the proxy naming the origin server as the target, and the proxy then requests the content from the origin server and returns it to the client. The client must be specially configured to use the forward proxy to access other sites.

Reverse Proxy

A reverse proxy appears to the client just like an ordinary Web server. No special configuration on the client is necessary. The client makes ordinary requests for content in the name-space of the reverse proxy. The reverse proxy then decides where to send those requests and returns the content as if it were itself the origin. The Oracle Adaptive Access Manager Proxy running in the Microsoft Internet Security and Acceleration (ISA) Server is an example of a reverse proxy.

OAAM Server

OAAM Server is the Web application component of Oracle Adaptive Access Manager. The Oracle Adaptive Access Manager Proxy redirects the client browser to OAAM Server for tracking and authentication purposes as defined by the Oracle Adaptive Access Manager Universal Installation Option Proxy XML configuration.

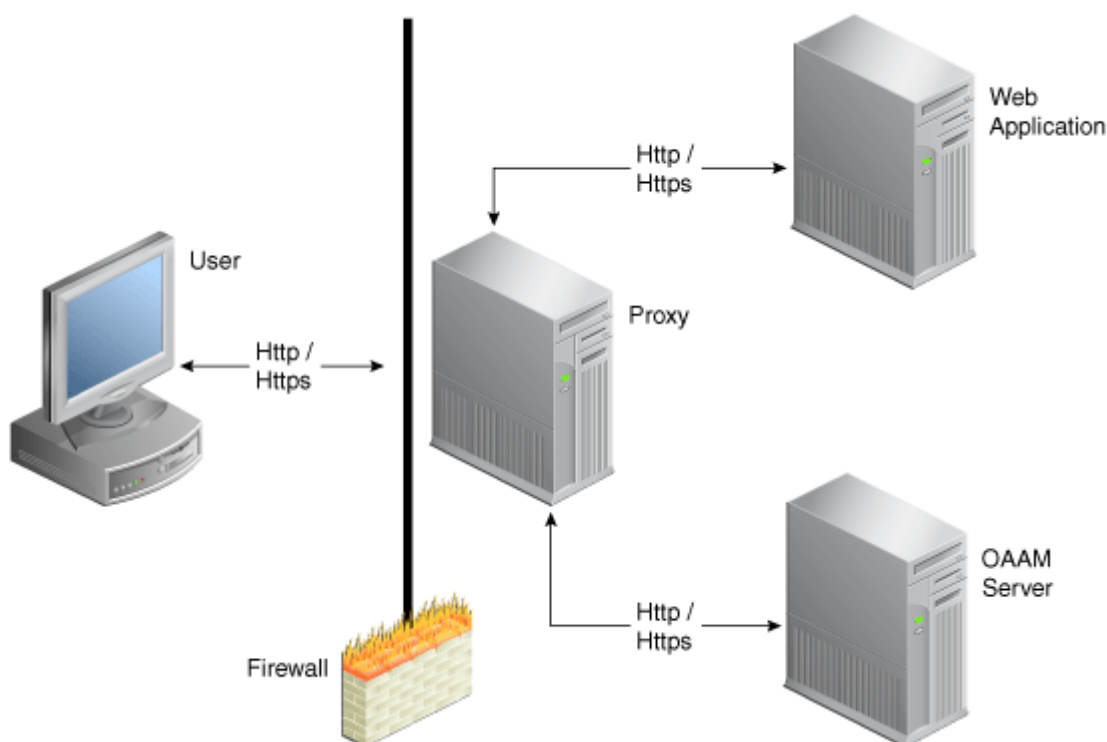
5.1.2 Architecture

The following diagrams show a typical Oracle Adaptive Access Universal Installation Option deployment.

The first diagram shows a Web application before the Oracle Adaptive Access Universal Installation Option is deployed to provide multifactor authentication.

Figure 5-1 Before the Oracle Adaptive Access Universal Installation Option

The second diagram shows various components added after the Oracle Adaptive Access Universal Installation Option deployment.

Figure 5-2 After Universal Installation Option Deployment

The Oracle Adaptive Access Manager Proxy intercepts the HTTP traffic between the client (browser) and the server (Web application) and performs the appropriate actions, such as redirecting the traffic to OAAM Server, to provide multifactor authentication and authorization. OAAM Server, in turn, communicates with OAAM Admin to assess the risk, and then takes the appropriate actions, such as permitting the login, challenging the user, blocking the user, and other actions.

The Oracle Adaptive Access Manager Proxy is available for the Apache Web server and Microsoft Internet Security and Acceleration (ISA) Server.

5.1.3 References

For detailed information on installing and configuring the Microsoft ISA server, refer to the Microsoft ISA Server setup documentation. Web publishing rule creation and listener creation are explained further in this document.

For more information about the Apache HTTP Server, refer to the Apache HTTP Server 2.2 documentation at:

<http://httpd.apache.org/docs/2.2>

5.2 Installing Oracle Adaptive Access Manager Proxy for Microsoft ISA

The Oracle Adaptive Access Manager Proxy for Microsoft ISA uses the API provided by Microsoft ISA Server to monitor the HTTP traffic and perform various actions. Refer to the Microsoft ISA Server setup documentation for the details on installing and configuring the ISA server. For a successful installation of the proxy, a .NET framework 2.0 or better should be installed. Install all the recommended updates from Microsoft on the machine.

Microsoft ISA Server 2006 Standard Edition should be installed and Web publishing rules for Web applications should be created before installing the Oracle Adaptive Access Manager Proxy.

This section provides:

- Information on creating Web publishing rules and listeners so that Web applications and OAAM Server can be accessible from the Internet.
 - [Section 5.2.1, "Proxy Web Publishing Configuration."](#)
- Instructions on installation and programming information for the Oracle Adaptive Access Manager Proxy for Microsoft ISA.
 - [Section 5.2.2, "Registering the Oracle Adaptive Access Manager Proxy for Microsoft ISA DLL."](#)
 - [Section 5.2.3, "Settings to Control the Proxy."](#)

5.2.1 Proxy Web Publishing Configuration

The purpose of this section is to explain the creation of Web publishing rules and listeners in Microsoft ISA for Adaptive Access Manager applications. It is intended for integrators who install and configure Microsoft ISA to support multiple Web applications.

5.2.1.1 Web Listener Creation

For details on creating a Web listener, refer the Microsoft Web site.

1. For the Web Listener Name, enter **Bharosa Proxy Listener**.
2. Select **SSL** secure connection as the type of connection the Web listener will establish with clients.
3. For the Web Listener IP Addresses, choose external, internal, and local host.
4. Choose to use a single certificate for the Web Listener and select the certificate.
5. Select no authentication for how clients will validate their credentials.

5.2.1.2 Web Publishing Rule Creation

In a typical deployment, Web applications and OAAM Server run on machines in an internal network and are not directly accessible from the Internet.

In the case of the Oracle Adaptive Access Manager Proxy for Microsoft ISA, only the Oracle Adaptive Access Manager Proxy machine, which runs Microsoft ISA Server, will be accessible from the Internet.

The following should be published via Web publishing rules in the Microsoft ISA Server.

- OAAM Server
- Web applications

Therefore, you need to set two (at least) rules, one for the main application and another for OAAM Server.

Refer to the Microsoft documentation for detailed instructions. The following tips are provided for your reference.

5.2.1.2.1 Web Publishing Rule Creation for OAAM Server

To create a new Web publishing rule for OAAM Server you will need to access Microsoft ISA Server's Web publishing rule wizard and follow the on-screen instructions.

1. For the name of the rule, enter a name such as `Bharosa OAAM Server`.
2. Choose to allow incoming requests matching the rule conditions.
3. Choose to publish a single Web site or a load balancer in front of several servers.
4. Choose **SSL** as a connection option if the Web application is listening on SSL; otherwise, choose the non-secured connection option.
5. For the internal site name, provide the machine name where the Web server runs. Translate the public name into the internal name.
6. If the IP address or the machine name of the Web application to be published is known, select the option to use the computer name or IP address and provide that information.
7. If you want to include all files and subfolders within a folder, enter `/*` for the name of the file or folder you want to publish. If you need to publish more than one file or folder, enter only the first file/folder instead. The remaining files can be entered later by editing the rule. Later you will enter the path you entered here for your public details.
8. For your Web listener, select **Bharosa Proxy Listener**.
9. For authentication delegation, select no delegation and that client cannot authenticate directly.
10. Make sure you are able to apply the rule to requests from all users.

Check the properties for your newly created rule by accessing the rule properties.

1. If more than one file or folders need to be published, add all paths.
2. If you have more than one domain name to access the application, add all the domain names.

5.2.1.2.2 Web Publishing Rule Creation for Protected Web Applications

To create a new Web publishing rule for Web applications, you will need to access Microsoft ISA Server's Web publishing rule wizard and follow the onscreen instructions.

1. For the name of the rule, enter a name such as `Online Banking Application`.
2. Choose to allow incoming requests matching the rule conditions.
3. Choose to publish a single Web site or a load balancer in front of several servers.

4. Choose **SSL** as a connection option if the Web application is listening on SSL; otherwise, choose the non-secured connection option.
5. For the internal site name, provide the machine name where the Web server runs.
6. If the IP address or the machine name of the Web application to be published is known, select the option to use the computer name or IP address and provide that information.
7. If you want to include all files and subfolders within a folder, enter `/*` for the name of the file or folder you want to publish. If you need to publish more than one file or folder, enter only the first file/folder instead. The remaining files can be entered later by editing the rule. Later you will enter the path you entered here for your public details.
8. For your Web listener, select **Bharosa Proxy Listener**.
9. For authentication delegation, select no delegation and that client cannot authenticate directly.
10. Make sure you are able to apply the rule to requests from all users.

Check the properties for your newly created rule by accessing the rule properties.

1. If more than one file or folders need to be published, add all paths.
2. If you have more than one domain name to access the application, add all the domain names.

5.2.2 Registering the Oracle Adaptive Access Manager Proxy for Microsoft ISA DLL

The Oracle Adaptive Access Manager Proxy for Microsoft ISA is installed as a Web filter in Microsoft ISA Server. To install the Oracle Adaptive Access Manager Proxy for Microsoft ISA, follow these steps:

1. Copy the `BharosaProxy.dll` to the Microsoft ISA Server installation directory, which is by default, `%ProgramFiles%\Microsoft ISA Server`
2. Open the command prompt and navigate to the Microsoft ISA Server installation directory
3. Register the `BharosaProxy.dll` with the following command:

```
regsvr32 .\BharosaProxy.dll
```

5.2.3 Settings to Control the Proxy

Various aspects of the Oracle Adaptive Access Manager Proxy for Microsoft ISA can be controlled using the registry values. All Oracle Adaptive Access Manager Proxy for Microsoft ISA settings are stored under `HKLM\SOFTWARE\Bharosa\Proxy` key. Changes to most of the registry values are picked up by the proxy immediately without requiring a proxy restart.

5.2.3.1 Configuration files

During startup (and during config reload), the proxy loads the configuration from the files listed under the `HKLM\SOFTWARE\Bharosa\Proxy\ConfigFiles` key.

- The type of each value under this key should be `REG_DWORD`.
- The name of each value under this key should be the filename containing the proxy configuration.

- The filename can either be fully qualified or relative to the location of the `BharosaProxy.dll`.
- The proxy will load a configuration file only if the data has a nonzero value. This can be used to dynamically load and unload proxy configuration files.
- The files will be loaded in the lexicographic order of the filenames in the registry.
- Changes under the `ConfigFiles` key will not be effective until either the proxy is restarted or `HKLM\SOFTWARE\Bharosa\Proxy\ReloadConfig` is set to 1.

5.2.3.2 Configuration Reload

The proxy configuration can dynamically be changed while the proxy is running; new configuration files can be added and currently loaded files can be updated or removed. These changes will not be applied until the `ReloadConfig` registry value is set to a nonzero value. When setting `ReloadConfig` to a nonzero value, the proxy will load configuration files. After loading the files, the proxy will reset the `ReloadConfig` value to 0.

Note that the new configuration will be used only for new client (browser) connections. Clients already connected will continue to use the previous configuration.

5.2.3.3 Session ID Cookie

The Oracle Adaptive Access Manager Proxy for Microsoft ISA uses a cookie to associate multiple requests from a client. The name of this cookie can be configured in the registry value, `SessionIdCookieName` (of type `REG_SZ`). If this value is not present or empty, the Oracle Adaptive Access Manager Proxy for Microsoft ISA will use the cookie name, `BharosaProxy_SessionId`.

5.2.3.4 Configuring Session Id Cookie attributes via Global Variables

The attributes of the `Session Id Cookie` can be configured using global variables listed in [Table 5-1](#).

Table 5-1 Session Id Cookie Attributes via Global Variables

Cookie Attribute	Global Variable Name	Description
expires	<code>SessionCookie_ExpiryInMinutes</code>	'expires' attribute will be added to the session cookie if this global variable is set to value greater than 0. This variable specifies the number of minutes the session cookie should be persisted by the client browser.
HttpOnly	<code>SessionCookie_IsHttpOnly</code>	'HttpOnly' attribute will be added to the session cookie if this global variable is set to value greater than 0
secure	<code>SessionCookie_IsSecure</code>	'secure' attribute will be added to the session cookie if this global variable is set to value greater than 0.
domain	<code>SessionCookie_DomainLevelCount</code>	'domain' attribute will be added to the session cookie if this global variable is set. For example, to set the cookie domain as ".mydomain.com" for an application at "test.myserver.mydomain.com", set this global variable to "2". The value should be greater than 1 - if a lower value is specified, proxy will use 2 as the value.

5.2.3.5 Session Inactive Interval

Sessions in the Oracle Adaptive Access Manager Proxy for Microsoft ISA will be removed after a certain period of inactivity. This period, in seconds, is specified in the

`MaxSessionInactiveInterval` registry value. If this value is not specified, the Oracle Adaptive Access Manager Proxy for Microsoft ISA will remove a session after 1200 seconds (20 minutes) of inactivity. This value should be set to at least a few seconds higher than the Web application session timeout value.

5.2.3.6 Settings for Troubleshooting

Trace messages from the Oracle Adaptive Access Manager Proxy for Microsoft ISA can be used for troubleshooting any issues with the proxy configuration and operation. Trace settings, like trace level and destinations, can be controlled using the registry values under `HKLM\SOFTWARE\Bharosa\Proxy`. Registry values are shown in [Table 5–2](#).

Table 5–2 Settings for Troubleshooting

Name	Type	Description
TraceFilename	REG_SZ	Full path to the file in which the trace messages should be written to
TraceFileMaxLength	REG_DWORD	Maximum length of the trace file in bytes. Once the trace file reaches this size, the proxy will rename the file by adding the timestamp to the filename and create a new trace file to write subsequent trace messages.
TraceToFile	REG_DWORD	Trace messages will be written to file only if this value is nonzero.
TraceToDebugTerminal	REG_DWORD	Trace messages will be written to debug the terminal only if this value is nonzero. Tools like DbgView can be used to view these trace messages in real time.
TraceLevel	REG_DWORD	Each trace level (debug, info, warning, error) has an integer value associated. The registry value should be set to the sum of desired the trace level values. FATAL 0x1, ERROR 0x2, WARN 0x4 INFO 0x8, DEBUG 0x10, HTML 0x80, FLOW 0x80000
IgnoreUrlMappings	REG_DWORD	If this value is nonzero, the proxy will ignore all the interceptors defined in the proxy configuration. Essentially this will put the Oracle Adaptive Access Manager Proxy for Microsoft ISA in "pass-through" mode.
CaptureTraffic	REG_DWORD	The proxy does not handle (save, inspect) the HTTP traffic for URLs that don't have interceptors defined in the configuration. But during application discovery process, it will be necessary to get a dump of all the HTTP traffic thorough the proxy. On such occasion, this registry value should be set to nonzero.

5.3 Installing Oracle Adaptive Access Manager Proxy for Apache

To install the Oracle Adaptive Access Manager Proxy for Apache, a new Apache `httpd` has to be installed into which the Oracle Adaptive Access Manager Proxy for Apache will be installed. This Apache `httpd` will use `mod_proxy` to `reverse-proxy` to the backend application that has to be protected.

The Installation section contains information for installing the Oracle Adaptive Access Manager Proxy for Apache for Windows and Linux platforms.

The installation procedure involves:

- Ensuring that the Apache `httpd` requirements are met
See [Section 5.3.2, "Apache httpd Requirements."](#)
- Copying the proxy `dlls` and supported `dlls` to specific directories in Apache
See [Section 5.3.3, "Copying the Oracle Adaptive Access Manager Proxy for Apache and Supported Files to Apache."](#)
- Configuring `memcache` (for Linux only)
See [Section 5.3.4, "Configuring Memcache \(for Linux only\)."](#)
- Editing the `httpd.conf` to activate the proxy
See [Section 5.3.5, "Configuring httpd.conf."](#)

As part of this section, information is also provide on optionally installing the `mod_proxy_html`, which is needed to rewrite the HTML links in a proxy situation, to ensure that links work for the users outside the proxy

- Modifying the settings of the proxy using application configuration XML files
[Section 5.3.6, "Modifying the Oracle Adaptive Access Manager Proxy for Apache Settings."](#)

The post-installation procedures involve:

- [Section 5.4, "Setting Up Rules and User Groups."](#)
Creating a new user to run the Universal Installation Option Proxy Apache process (on Linux only)
- [Section 5.5, "Setting Up Policies."](#)

5.3.1 Proxy Files for Windows and Linux

The Oracle Adaptive Access Manager Proxy for Apache binaries for Windows and Linux are different. Since the proxy is in C/C++, the same binary will not work on different platforms (unlike Java).

The files are located under `$ORACLE_HOME/oaam/oaam_proxyplatform_specific_file`.

5.3.1.1 Windows

For Windows, the binary files are listed in [Table 5-3](#).

Table 5-3 Windows Binary Files

Name	Description
<code>mod_uio.so</code>	Oracle Adaptive Access Manager Proxy for Apache module
<code>log4cxx.dll</code>	Apache Log4cxx library
<code>libxml2.dll</code>	XML/HTML Parser
<code>apr_memcache.dll</code>	APR Memcache client library.

The data files are listed in [Table 5-4](#).

Table 5-4 Windows Data files

Name	Description
<code>UIO_Settings.xml</code>	Oracle Adaptive Access Manager Proxy for Apache Settings XML file

Table 5–4 (Cont.) Windows Data files

Name	Description
UIO_log4j.xml	Oracle Adaptive Access Manager Proxy for Apache Log4j (log4cxx) configuration XML file
TestConfig.xml	Oracle Adaptive Access Manager Proxy for Apache Sample application configuration file
UIO_Settings.rng	Relax NG grammar for UIO_Settings.xml
UIO_Config.rng	Relax NG grammar for application configuration XML files

5.3.1.2 Linux

For Linux, the binary files are listed in [Table 5–5](#).

Table 5–5 Linux Binary Files

Name	Description
mod_uio.so	Oracle Adaptive Access Manager Proxy for Apache module
liblog4cxx.so.0.10.0.0	Apache Log4cxx library
libxml2.so.2.6.32	XML/HTML parser
libapr_memcache.so.0.0.1	APR Memcache client library.

The data files are listed in [Table 5–6](#).

Table 5–6 Linux Data Files

Name	Description
UIO_Settings.xml	Oracle Adaptive Access Manager Proxy for Apache Settings XML file
UIO_log4j.xml	Oracle Adaptive Access Manager Proxy for Apache Sample Log4j (log4cxx) configuration XML file
TestConfig.xml	Oracle Adaptive Access Manager Proxy for Apache Sample application configuration files
UIO_Settings.rng	Relax NG grammar for UIO_Settings.xml
UIO_Config.rng	Relax NG grammar for application configuration XML files

5.3.2 Apache httpd Requirements

The pre-installation steps involved for downloading or building the Apache httpd, depend on the platform, Windows or Linux, and on whether certain requirements are met.

5.3.2.1 Windows

You can download the latest Apache httpd (2.2.8) build for Windows from the Apache Web site.

Ensure that:

- the Apache httpd (2.2.8) build is version 2.2.8
- the mod_proxy support is enabled (the standard installation contains the mod_proxy)
- the mod_ssl support is enabled

5.3.2.2 Linux

Instructions to build the Apache `httpd` are available on the Apache Web site. When you build Apache, ensure that

- the Apache `httpd` (2.2.8) build is version 2.2.8
- the `mod_so` is enabled (for dynamically loading modules)
- the `mod_proxy` is enabled
- the `mod_ssl` support is enabled

5.3.3 Copying the Oracle Adaptive Access Manager Proxy for Apache and Supported Files to Apache

Instructions are provided in this section for copying the Oracle Adaptive Access Manager Proxy for Apache and support files to specific directories in Apache for both Windows and Linux platforms.

5.3.3.1 Windows

Table 5–7 summarizes:

- The directories you have to copy the Oracle Adaptive Access Manager Proxy for Apache files to after installation
- The tree structure of the Oracle Adaptive Access Manager Proxy for Apache libraries and configuration files, assuming that you installed the files in `C:\Apache2.2`
- The directories the Oracle Adaptive Access Manager Proxy for Apache binary files go into are listed in Table 5–7.

Table 5–7 Directories for Windows UIO Binary Files

Directories	File Descriptions
<code>C:\Apache2.2\modules\mod_uio.so</code>	Oracle Adaptive Access Manager Proxy for Apache module
<code>C:\Apache2.2\bin\log4cxx.dll</code>	Apache Log4cxx library
<code>C:\Apache2.2\bin\libxml2.dll</code>	XML/HTML Parser
<code>C:\Apache2.2\bin\apr_memcache.dll</code>	APR Memcache library.

The data files will go in the directories summarized in Table 5–8.

Table 5–8 Directories for Windows UIO Data Files

Directories	File Descriptions
<code>C:\OAAMUIO\UIO_Settings.xml</code>	Oracle Adaptive Access Manager Proxy for Apache settings XML file
<code>C:\OAAMUIO\UIO_log4j.xml</code>	Oracle Adaptive Access Manager Proxy for Apache Log4j (log4cxx) configuration XML file
<code>C:\OAAMUIO\TestConfig.xml</code>	Oracle Adaptive Access Manager Proxy for Apache application configuration files (any number)
<code>C:\OAAMUIO\UIO_Settings.rng</code>	Relax NG grammar for <code>UIO_Settings.xml</code>
<code>C:\OAAMUIO\UIO_Config.rng</code>	Relax NG grammar for application configuration XML files

Table 5–8 (Cont.) Directories for Windows UIO Data Files

Directories	File Descriptions
C:\OAAMUIO\logs\uiolog	Oracle Adaptive Access Manager Proxy for Apache log

If you want to change the location of the various configuration files, refer to the "Configuring `httpd.conf`" section.

5.3.3.2 Linux

After the installation of the Apache `httpd`, you will have to copy the Oracle Adaptive Access Manager Proxy for Apache binary files into (assuming Apache `httpd` is installed in `/usr/local/apache2`) the directories shown in [Table 5–9](#).

Table 5–9 Directories for Linux UIO Binary Files

Directories	Description
<code>/usr/local/apache2/modules/mod_uio.so</code>	Oracle Adaptive Access Manager Proxy for Apache Module
<code>/usr/local/apache2/lib/liblog4cxx.so.0.10.0.0</code>	Apache Log4cxx Library
<code>/usr/local/apache2/lib/libxml2.so.2.6.32</code>	XML/HTML Parser
<code>/usr/local/apache2/lib/libapr_memcache.so.0.0.1</code>	APR Memcache client library.

Then, create soft links to the libraries as follows:

```
cd /usr/local/apache2/lib
ln -s liblog4cxx.so.0.10.0.0 liblog4cxx.so.10
ln -s libxml2.so.2.6.32 libxml2.so.2
ln -s libapr_memcache.so.0.0.1 libapr_memcache.so.0
```

Also, ensure that the binary files have executable permission.

Apache `httpd` is typically run as `root` so that it creates a parent process that listens on port 80, and it spawns handler processes that run as the user given in the `User` directive in `httpd.conf`.

For this reason, create a user called `oaamuiio` that will be the checkpoint user for the Oracle Adaptive Access Manager Universal Installation Option for Apache. The Oracle Adaptive Access Manager Universal Installation Option configuration and log files will be accessible by this user. Ensure that only this user can access the log files. Assuming `/home/oaamuiio` is the home directory for this user, the directory structure will look like the one presented in [Table 5–10](#).

The Oracle Adaptive Access Manager Universal Installation Option for Apache data files should follow the directory structure shown in [Table 5–10](#).

Table 5–10 Directories for Linux UIO Data Files

Directories	Description
<code>/home/oaamuiio/uio/UIO_Settings.xml</code>	Oracle Adaptive Access Manager Proxy for Apache settings XML file
<code>/home/oaamuiio/uio/UIO_log4j.xml</code>	Oracle Adaptive Access Manager Proxy for Apache Log4j (log4cxx) configuration XML file

Table 5–10 (Cont.) Directories for Linux UIO Data Files

Directories	Description
/home/oaamuio/uio/TestConfig.xml	Oracle Adaptive Access Manager Proxy for Apache application configuration files (any number)
/home/oaamuio/uio/UIO_Settings.rng	Relax NG grammar for UIO_Settings.xml
/home/oaamuio/uio/UIO_Config.rng	Relax NG grammar for application configuration XML files
/home/oaamuio/uio/logs/uio.log	Oracle Adaptive Access Manager Proxy for Apache log

If you want to change the location of the various configuration files, refer to the "Configuring httpd.conf" section.

The run-time user of httpd should have the appropriate permissions to access all these files.

5.3.4 Configuring Memcache (for Linux only)

Apache httpd ships with a selection of Multi-Processing Modules (MPMs) which are responsible for binding to network ports on the machine, accepting requests, and dispatching children to handle the requests. On Linux, httpd can run with two different MPMs (the httpd kernel): httpd with prefork MPM (httpd kernel) or with worker MPM. The MPM is built into the httpd and is not a run-time option. Usually, the binary distribution of Apache httpd is with prefork MPM. If you need to use worker MPM, you will have to build Apache httpd using the instructions from the Apache Web site.

With prefork MPM, httpd maintains a pool of single-threaded processes, where each request is handled by a single process. With worker MPM, httpd maintains a pool of multithreaded processes, where every process could be handling multiple requests at a time. (On Windows, httpd MPM is always in multi-threading mode with a single process.)

On Linux, in the case where the httpd runs multiple process (irrespective of single or multithreaded), the Oracle Adaptive Access Manager Proxy for Apache session data must be maintained in a common store (database or cache) so that multiple processes can access the session data. The Oracle Adaptive Access Manager Proxy uses memcache (a memory based very fast cache) to store the session data.

At startup, the Oracle Adaptive Access Manager Proxy autodetects whether httpd is running with a single process or multiple processes. If httpd is running with multiple processes (which is the case with prefork or worker mpm on Linux), it will try to connect to the memcache daemon using default connection parameters (that are defined in Section 5.3.6.1, "UIO_Settings.xml"). On Windows, by default, the Oracle Adaptive Access Manager Proxy will use local sessions. It does not connect to the memcache daemon; however it can be configured to maintain session data in the memcache daemon (explained in Section 5.3.6.1, "UIO_Settings.xml").

For the scenarios where the Oracle Adaptive Access Manager Proxy for Apache will be connecting to memcache daemon, you will have to install memcache on your system using the instructions from the memcache Web site and run the memcache daemon(s) before running the Apache httpd.

Install memcache using instructions at:

<http://www.danga.com/memcached>

You may already have a binary installation available from your Linux distribution. The Oracle Adaptive Access Manager Proxy for Apache has been tested with version 1.2.5 of memcache.

In the simple configuration, you can run a single memcache daemon on the machine that is running your Apache httpd.

You can choose to have a highly scalable installation, where you run more than one memcache daemon-- all of which are accessed by multiple machines running Apache httpds.

5.3.5 Configuring httpd.conf

This section provides information on how to edit the httpd.conf file to activate the Oracle Adaptive Access Manager Proxy for Apache.

5.3.5.1 Basic Configuration without SSL

In the sample installation, the Apache httpd has been installed in `c:\ProgramFiles\Apache2.2` or `/usr/local/apache2`.

Also, in the sample installation, BigBank40 and BharosaUIO40 are running on `test.dummy.com`.

To ensure that http.conf is correctly set up in your environment, follow these steps:

1. Ensure that the following lines are uncommented to enable `mod_proxy`.

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

2. Add the following line to the end of the `LoadModule` group of lines to activate the Oracle Adaptive Access Manager Proxy for Apache.

```
LoadModule uio_module modules/mod_uio.so
```

3. Add a line to point to the `UIO_Settings.xml` file that has the settings for the Oracle Adaptive Access Manager Proxy for Apache.

Note: This should be an absolute path to the `UIO_Settings.xml` file.

On Windows (all paths should be with forward slashes),

```
UioProxySettingsFile c:/OAMUIO/UIO_Settings.xml
```

On Linux,

```
UioProxySettingsFile /home/oaamuio/uio/UIO_Settings.xml
```

4. Disable `mod_proxy`'s forward-proxy-ing capability since it is not needed.

```
ProxyRequests Off
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>
```

5. Enable the `mod_proxy` configuration to reverse-proxy to the protected applications (BigBank40 and BharosaUIO40 in our sample installation).

```
ProxyPass / http://uio-dev.oracle.com:9090/
```



```
ProxyPassReverse / http://uio-dev.oracle.com:9090/
```

6. Set the user/group of `httpd` using `User` and `Group` directives to `oaamuio`.

The actual settings for #4 and #5 are installation-specific. They are only examples of the settings you must set. For information on setting details, refer to the Apache Web site.

With the changes described and by properly setting up `UIO_Settings.xml`, you should be able to access BigBank40 and run Phase One scenarios. The URL for BigBank40 is

```
http://<apache-host>:<apache-port>/bigbank40
```

So far in this chapter, we have performed the configuration to the proxy without using SSL.

5.3.5.2 Configuration with SSL

To enable SSL, refer to the Apache Web site for Tomcat and for Apache procedures.

Note that the Oracle Adaptive Access Manager Proxy for Apache requires `mod_ssl` to be part of `httpd`. This ensures that the OpenSSL library is linked in and is properly configured for the Oracle Adaptive Access Manager Proxy for Apache to generate session ids. You need to ensure that `mod_ssl` is loaded and you do not need to do any configuration if you are not using SSL.

mod_proxy_html module (optional)

Optionally, you may need to install the `mod_proxy_html` (http://apache.webthing.com/mod_proxy_html/) Apache module. This module is needed only if the protected application has Web pages that have hard-coded URL links to itself. If the application has relative URLs, you do not need this module.

From their Web site, the executive summary of this module is as follows:

`mod_proxy_html` is an output filter to rewrite HTML links in a proxy situation, to ensure that links work for users outside the proxy. It serves the same purpose as Apache's `ProxyPassReverse` directive does for HTTP headers, and is an essential component of a reverse proxy.

For example, if a company has an application server at `appserver.example.com` that is only visible from within the company's internal network, and a public webserver `www.example.com`, they may wish to provide a gateway to the application server at `http://www.example.com/appserver/`. When the application server links to itself, those links need to be rewritten to work through the gateway. `mod_proxy_html` serves to rewrite `foobar` to `foobar` making it accessible from outside."

5.3.6 Modifying the Oracle Adaptive Access Manager Proxy for Apache Settings

5.3.6.1 UIO_Settings.xml

```
<UIO_ProxySettings xmlns="http://bharosa.com/">
```

```
<Log4jProperties location="C:/OAAMUIO/UIO_log4j.xml" />
```

Or

```
<Log4jProperties location="/home/oaamuio/uio/UIO_log4j.xml" />

<GlobalVariable name="@one" value="something" />

<ConfigFile location="/home/oaamuio/uio/TestConfig1.xml" enabled="false" />
<ConfigFile location="/home/oaamuio/uio/TestConfig.xml" enabled="false" />

<ConfigFile location="C:/OAAMUIO/TestConfig1.xml" enabled="false" />
<ConfigFile location="C:/OAAMUIO/TestConfig.xml" enabled="true" />

<Setting name="GarbageCollectorInterval_ms" value="5" />
<Setting name="MaxSessionInactiveInterval_ms" value="5" />
<Setting name="SessionIdCookieName_str" value="UIOSessionId" />

<Setting name="IgnoreUrlMappings" value="0" />
<Setting name="CaptureTraffic" value="0" />

</UIO_ProxySettings>
```

Log4jProperties

Set the location of log4j.xml file that defines the logging configuration for the Oracle Adaptive Access Manager Proxy for Apache. The location should be an absolute path; it cannot be ServerRoot relative. On Linux, you have to ensure that the httpd process can access the directory.

When using httpd in a multiprocessing mode, do not use FileAppender; use SocketAppender instead to log the logs from the different processes. Refer to the log4j documentation on the Internet for more information.

GlobalVariable

GlobalVariable is a global variable that is used in the application configuration. You can have any number of such name-value pairs.

ConfigFile

ConfigFile is the absolute path to an application configuration. You can have any number of such configurations. Again, you need to make sure, on Linux, that the httpd process has the permissions to access these files. Refer to ["Configuring the Oracle Adaptive Access Manager Proxy"](#) to understand how to perform a configuration for an application.

Memcache

Memcache has the IP address and port of a memcache server. You can have multiple Memcache elements in the settings file if you have multiple memcache servers running. If you have a single local memcache running, you do not need to have this element at all. By default, the Oracle Adaptive Access Manager Proxy for Apache will try to connect to memcache on IP address 127.0.0.1 and port 11211.

Settings

These are flags to control the behavior of the Oracle Adaptive Access Manager Proxy for Apache. Various settings are listed in [Table 5-11](#).

Table 5–11 UIO Proxy Flags.

Flags	Description
MaxSessionInactiveInterval_sec	Session expiry time in sec (default = 30 minutes) For example, <Setting name="MaxSessionInactiveInterval_sec" value="1800"/>
GarbageCollectorInterval_ms	Interval for running session expiry thread (default = 5 minutes) For example, <Setting name="GarbageCollectorInterval_ms" value="300000"/>
FileWatcherInterval_ms	Interval for checking if the settings or any config file has changed (default = 1minute) For example, <Setting name="FileWatcherInterval_ms" value="60000"/> (After modifying the configuration XML file, even though the proxy will update the configuration on the fly, it is advisable to restart the httpd server.)
SessionIdCookieName_str	Name of the cookie used by Universal Installation Option to maintain its session (default = OAAM_UIOProxy_SessionId) For example, <Setting name="SessionIdCookieName_str" value="SessionId"/>
SessionCookie_DomainLevelCount	Domain level for the Sessions cookie For example, <Setting name="SessionCookie_DomainLevelCount" value="2"/>
SessionCookie_ExpiryInMinutes	The value of this setting is used to compute the expiry time that is put in the expires attribute of the Set-Cookie header of the Apache UIO Proxy session cookie. Default is zero which means the expires attribute is not added.
SessionCookie_IsHttpOnly	If set to 1, the cookie is marked as HTTP only in the Set-Cookie header. Default is not to mark the cookie as HTTP only.
SessionCookie_IsSecure	If set to 1, the cookie is marked as secure in the Set-Cookie header. Default is not to mark the cookie as secure.
IgnoreUrlMappings	Ignore the application configuration XML files; the proxy behaves as a flow-through proxy For example, <Setting name="IgnoreUrlMappings" value="0"/>. The value of 0 disables this mode and the value of 1 enables capture traffic mode. The value of 1 will make the proxy act as flow-through and the value of 0 will enable the configuration XML interceptors.
CaptureTraffic	Capture the HTTP traffic - headers and content in the log files. This mode is for debugging purpose. Note that it captures the headers and contents as is and could contain customer's personal data. Use this mode with caution and only for debugging/test. For example, <Setting name="CaptureTraffic" value="0"/>. Value of 1 enables capture traffic and 0 disables it.
MaxReqBodyBytes	Maximum request body size to cache while processing requests. This is necessary when the application has POSTs with big files getting uploaded. For example, <Setting name="MaxReqBodyBytes" value="10240"/>

Table 5–11 (Cont.) UIO Proxy Flags.

Flags	Description
UseMemcache	Force the use of memcache even when httpd is running in single process mode. Has no effect when running in multiple process mode. Applies at startup and requires restarting httpd for change to apply. For example, <Setting name="UseMemcache" value="1"/>. Value of 1 enables use of memcache for a single process httpd. Value of 0 is ignored.
CachedConfigExpiry_sec	Expiry time for unused config XML data in memory, if multiple config XML configurations have been loaded into memory. This happens when config XML files are automatically loaded when they are modified. (Default = 60 minutes). For example, <Setting name="CachedConfigExpiry_sec" value="3600"/>
AutoLoadConfig	Set to 1 to enable auto-loading of config XML files when they are modified by user. Set to 0 to turn this feature off. It is OK to enable this feature when using single-process mode of httpd. Do not enable this feature for multiple process mode of httpd for production use, since individual processes could have different versions of the config XML files. For example, <Setting name="AutoLoadConfig" value="1"/>. Value of 1 enables auto-load and 0 disables it.

5.3.6.2 UIO_log4j.xml

For actual log4j format details, refer to log4j manual available on the Internet. Apache: :log4cxx is a C++ implementation of the log4j framework and the XML file format is common to log4cxx and log4j.

5.3.6.3 Application configuration XMLs

These XML files are the application configuration files that are defined in the ConfigFile element of UIO_Settings.xml file.

5.4 Setting Up Rules and User Groups

For information on setting up rules and user groups, refer to the *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*.

5.5 Setting Up Policies

To set up policies for Universal Installation Option, import the out-of-the-box policies. Information about importing policies is available in the *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

5.6 Configuring the Oracle Adaptive Access Manager Proxy

The Oracle Adaptive Access Manager Proxy intercepts all HTTP traffic between the client browser and the Web application and performs actions specified in the configuration files. The configuration files are in XML format that comply with the Oracle Adaptive Access Manager Proxy configuration XML schema 2.

5.6.1 Elements of the Proxy Configuration File

The following sections describe various elements of the Oracle Adaptive Access Manager Proxy configuration file.

5.6.1.1 Components of Interceptors

Interceptors are the most important elements in the Oracle Adaptive Access Manager Proxy configuration. You will see that authoring the Oracle Adaptive Access Manager Proxy configuration file is all about defining interceptors.

There are two types of interceptors: request interceptors and response interceptors. As the names suggest, request interceptors are used when the Oracle Adaptive Access Manager Proxy receives HTTP requests from the client browser and response interceptors are used when the Oracle Adaptive Access Manager Proxy receives HTTP response from the server i.e. Web application or OAAM Server.

There are four components to an interceptor and all of them are optional.

1. List of URLs - the interceptor will be evaluated if the interceptor URL list contains the current request URL or if the URL list is empty.
2. List of conditions - conditions can inspect the request/response contents, such as checking for the presence of an HTTP header/parameter/cookie, and so on, or testing whether a header/parameter/cookie has a specific value or not. Filters and action defined in the interceptor will be executed only if all the conditions specified are met or if no condition is specified.
3. List of filters - filters perform an action that might modify the request/response contents or modify some state information in the Oracle Adaptive Access Manager Proxy. For example, a filter can add/remove HTTP headers, save HTTP header/parameter/cookie value in a proxy variable, and so on.
4. Action - after executing the filters the interceptor will perform the action, if one is specified. Actions can be one of:
 - a. redirect the client to a different URL
 - b. send a saved response to the client
 - c. perform a HTTP get on server
 - d. perform a HTTP post on server
 - e. send a saved request to the server

5.6.1.2 Conditions

Conditions are used in the Oracle Adaptive Access Manager Proxy to inspect HTTP request/response or the state information saved in the proxy (variables). Each condition evaluates to either true or false. Conditions are evaluated in the order they are listed in the configuration file until a condition evaluates to false or all conditions are evaluated. [Table 5–12](#) lists conditions that can be defined in an interceptor.

Table 5–12 Conditions Defined in an Interceptor

Condition name	Attributes	Description
HeaderPresent	id, enabled, name	Checks the presence of the specified header in request/response. The header name should be terminated by a colon (":"). Example: <HeaderPresent name="userid:"/>
ParamPresent	id, enabled, name	Checks the presence of the specified parameter in request. Example: <ParamPresent name="loginID"/>
QueryParamPresent	id, enabled, name	Checks the presence of the specified query parameter in the URL. Example: <QueryParamPresent name="TraceID"/>
VariablePresent	id, enabled, name	Checks whether the specified proxy variable has been set. Example: <VariablePresent name="\$userid"/>
RequestCookiePresent	id, enabled, name	Checks the presence of the specified cookie in request Example: <RequestCookiePresent name="SESSIONID"/>
ResponseCookiePresent	id, enabled, name	Checks the presence of the specified cookie in response Example: <ResponseCookiePresent name="MCWUSER"/>
HeaderValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified request/response header value matches the given value. The header name should be terminated by a colon (":"). Example: <HeaderValue name="Rules-Result:" value="allow"/>
ParamValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified request parameter value matches the given value. Example: <ParamValue name="cancel" value="Cancel"/>
QueryParamValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified URL query parameter value matches the given value. Example: <QueryParamValue name="requestID" value="Logout"/>
VariableValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified proxy variable value matches the given value. Example: <VariableValue name="%REQUEST_METHOD" value="post"/>

Table 5–12 (Cont.) Conditions Defined in an Interceptor

Condition name	Attributes	Description
RequestCookieValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified request cookie value matches the given value. Example: <RequestCookieValue name="CurrentPage" value="/onlineserv/" mode="begins-with" ignore-case="true"/>
ResponseCookieValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified response cookie value matches the given value. Example: <ResponseCookieValue name="CurrentPage" value="/onlineserv/" mode="begins-with" ignore-case="true"/>
HttpStatus	id, enabled, status	Checks whether the status code of the response matches the given value. Example: <HttpStatus status="302"/>
HtmlElementPresent	id, enabled, name, attrib-name, attrib-value, attrib-name1, attrib-value1, ... attrib-name9, attrib-value9,	Checks presence of a html element to match the specified conditions: <name attrib-name="attrib-value" attrib-name1="attrib-value1" .../> Example: <HtmlElementPresent name="form" attrib-name="name" attrib-value="signon"/>
PageContainsText	id, enabled, text	Checks whether the response contains the given text. Example: <PageContainsText text="You have entered an invalid Login Id"/>
NotVariableValue	id, enabled, name, value, mode, ignore-case	Checks whether the specified proxy variable value does not match the given value. Example: <NotVariableValue name="\$Login-Status" value="In-Session"/>

Table 5–12 (Cont.) Conditions Defined in an Interceptor

Condition name	Attributes	Description
And	id, enabled	Evaluates to true only if all the child conditions evaluate to true. Example: <And> <PageContainsText text="Your password must be"/> <PageContainsText text="Please re-enter your password"/> </And>
Or	id, enabled	Evaluates to true if one of the child conditions evaluates to true. Example: <Or> <ParamValue name="register" value="Continue"/> <ParamValue name="cancel" value="Cancel"/> </Or>
Not	id, enabled	Reverses the result of the child condition(s). Example: <Not> <HttpStatus status="200"/> </Not>

Attribute "id" is optional and is used only in trace messages. If no value is specified, the condition name (like `HeaderPresent`) will be used.

Attribute "enabled" is optional and the default value is "true". This attribute can be used to enable/disable a condition. The value of this attribute can be set to the name of a global variable; in such case, the condition will be enabled or disabled according to the value of the global variable.

Attribute "value" can be set to the name of a proxy variable. In such a case, the proxy will evaluate the variable at checkpoint and use that value in the condition.

Attribute "mode" can be set to one of the following: `begins-with`, `ends-with`, `contains`.

Attribute "ignore-case" can be set to one of the following: `true`, `false`.

5.6.1.3 Filters

Filters are used in the Oracle Adaptive Access Manager Proxy to modify HTTP request/response contents or modify the state information saved in the proxy (variables). Filters are executed in the order they are listed in the configuration file. [Table 5–13](#) lists filters that can be defined in an interceptor.

Table 5–13 *Filters Defined in an Interceptor*

Filter name	Attributes	Description
AddHeader	id, enabled, name, value	Adds the specified header with a given value to request/response. The header name should be terminated by a colon (":"). Example: <code><AddHeader name="userid:" value="\$userid"/></code>
SaveHeader	id, enabled, name, variable	Saves the specified request/response header value in the given proxy variable. The header name should be terminated by a colon (":"). Example: <code><SaveHeader name="userid:" variable="\$userid"/></code>
RemoveHeader	id, enabled, name	Removes the specified header from request/response. The header name should be terminated by a colon (":"). Example: <code><RemoveHeader name="InternalHeader:"/></code>
AddParam	id, enabled, name, value	Adds a request parameter with a specified name and value. Example: <code><AddParam name="loginID" value="\$userid"/></code>
SaveParam	id, enabled, name, variable	Saves the specified request parameter value in to the given proxy variable. Example: <code><SaveParam name="loginID" variable="\$userid"/></code>
AddRequestCookie	id, enabled, name, value	Adds the specified cookie with a given value to request Example: <code><AddRequestCookie name="JSESSIONID" value="\$JSESSIONID"/></code>
SaveRequestCookie	id, enabled, name	Saves the specified request cookie value in the given proxy variable
AddResponseCookie	id, enabled, name	Adds the specified cookie with a given value to response Example: <code><AddResponseCookie name="JSESSIONID" value="\$JSESSIONID"/></code>
SaveResponseCookie	id, enabled, name	Saves the specified response cookie value in the given proxy variable. Example: <code><SaveResponseCookie name="JSESSIONID" variable="\$JSESSIONID"/></code>
SaveHiddenFields	id, enabled, form, variable, save-submit-fields	Saves all the hidden, submit fields value, in the given form if form name is specified to the given proxy variable. To not save submit fields, set save-submit-fields attribute to false. Example: <code><SaveHiddenFields form="pageForm" variable="%lg_HiddenParams"/></code>

5.6.1.4 Filter Examples - ProcessString

Find the sub-string between the given start-tag and end-tag in the source string, extract the sub-string found and save extracted sub-string in the given variable.

```
<ProcessString source="%RESPONSE_CONTENT"
  find="sub-string"
  start-tag="var traceID = '" end-tag="';"
  action="extract"
  variable="$TRACE_ID"/>
```

Find the given search-string in the source string, replace it with the replace string and save the updated string in the given variable.

```
<ProcessString
  source="/bfb/accounts/accounts.asp?TraceID=$TRACE_ID"
  find="string" search-str="$TRACE_ID"
  action="replace"
  replace="$TRACE_ID"
  variable="%POST_URL"/>
```

Find the sub-string between the given start-tag and end-tag in the source string, replace it (including the start and end tags) with the evaluated value of the sub string found and save the updated string in the given variable.

```
<ProcessString
  source="/cgi-bin/mcw055.cgi?TRANEXIT[$UrlSuffix]"
  find="sub-string" start-tag="[" end-tag="]"
  action="eval"
  variable="%LogoffUrl"/>
```

5.6.1.5 Filter Examples - FormatString

Here is an example to create a HTTP Basic Authentication response header in variable `$AuthHeaderValue`, using the username/password in variables `%userid` and `%password`:

```
<FormatString variable="%UsernamePassword"
  format-str="{0}:{1}"
  param-0="%userid"
  param-1="%password"
  encoder="Base64"/>
```

```
<FormatString variable="$AuthHeaderValue"
  format-str="Basic {0}"
  param-0="%UsernamePassword"/>
```

5.6.1.6 Actions

An interceptor can optionally perform one of the following actions after executing all the filters. No further interceptors will be attempted after executing an action.

redirect-client

Often the proxy would need to redirect the client to load another URL; `redirect-client` is the action to use in such cases. The proxy will send a 302 HTTP response to request the client to load the specified URL.

If the `display-url` attribute is specified in the interceptor, the proxy will send a HTTP 302 response to the browser to load the URL specified in `display-url`

attribute. When the proxy receives this request, it will do a HTTP-GET on the server to get the URL specified in "url" attribute.

send-to-client

Often a response from the server would have to be saved in the proxy and sent to the client later after performing a few other HTTP requests; `send-to-client` is the action to use in such cases. The proxy will send the client the contents of specified variable.

If the `display-url` attribute is specified in the interceptor, the proxy will send a HTTP 302 response to the browser to load the URL specified in `display-url` attribute. When the proxy receives this request, it will send the response specified in the interceptor.

get-server

Sometimes the proxy would need to get a URL from the server; `get-server` is the action to use in such cases. The proxy will send a HTTP-GET request for the specified URL to the server.

If the `display-url` attribute is specified in the interceptor or if this action is specified in a response interceptor, the proxy will send a HTTP 302 response to the browser. When the proxy receives this request it will do a HTTP-GET on the server to get the URL specified in "url" attribute.

post-server

Sometimes the proxy would need to post to a URL in the server; `post-server` is the action to use in such cases. The proxy will send a HTTP-POST request for the specified URL to the server.

If `display-url` attribute is specified in the interceptor or if this action is specified in a response interceptor, the proxy will send a HTTP 302 response to the browser. When the proxy receives this request it will do a HTTP-POST to the server to the URL specified in "url" attribute.

send-to-server

In certain situations the request from client needs to be saved in the proxy and sent to the server later after performing a few other HTTP requests; `send-to-server` is the action to use in such cases. The proxy will send the contents of the specified variable to the server.

If `display-url` attribute is specified in the interceptor or if this action is specified in a response interceptor, the proxy will send a HTTP 302 response to the browser. When the proxy receives this request it will send the request specified in the interceptor to the server.

5.6.1.7 Variables

The proxy variables can store string data in the proxy memory. Variables can be used in conditions, filters and actions. For example, `SaveHeader` filter can be used to save the value a specific header in the given proxy variable. This variable value could later be used, for example, to add a parameter to the request. Variables can also be used in conditions to determine whether to execute an interceptor or not.

The proxy variables are of 3 types, depending upon the lifespan of the variable. The type of variable is determined by the first letter of the variable name, which can be one of: %, \$, @.

All types of variables can be set using filters like `SetVariable`, `SaveHeader`, `SaveParam`, `SaveResponse`, etc.

All types of variables can be unset/deleted by `UnsetVariable` filter. `ClearSession` filter can be used to remove all session variables.

Request variables

Request variables - these variable names start with `%`. These variables are associated with the current request and are deleted at the completion of the current request. Request variables are used where the value is not needed across requests.

Session variables

Session variables - these variable names start with `$`. These variables are associated with the current proxy session and are deleted when the proxy session is cleaned up. Session variables are used where the value should be preserved across requests from a client.

Global variables

Global variables - these variable names start with `@`. These variables associated with the current proxy configuration and are deleted when the proxy configuration is unloaded. Global variables are used where the value needs to be preserved across requests and across clients.

Global variables can be set at the proxy configuration load time using `SetGlobal` in the configuration file. Global variables can also be set by adding registry values under key `HKLM\Software\Bharosa\Proxy\Globals`. Name of each entry under this key should be the variable name, starting with `@`. And the data of the entry should be the value of the variable. The registry-type of the value can be `REG_DWORD`, `REG_SZ` or `REG_EXPAND_SZ`.

Configuring Session ID cookie attributes via Global Variables

The attributes of the Session ID Cookie can be configured using global variables listed in [Table 5–14](#).

Table 5–14 *Global variables to configure session ID cookie*

Cookie Attribute	Global Variable Name	Description
expires	@SessionCookie_ExpiryInMinutes	'expires' attribute will be added to the session cookie if this global variable is set to value greater than 0. This variable specifies the number of minutes the session cookie should be persisted by the client browser.

Table 5–14 (Cont.) Global variables to configure session ID cookie

Cookie Attribute	Global Variable Name	Description
HttpOnly	@SessionCookie_IsHttpOnly	'HttpOnly' attribute will be added to the session cookie if this global variable is set to value greater than 0.
secure	@SessionCookie_IsSecure	'secure' attribute will be added to the session cookie if this global variable is set to value greater than 0.
domain	@SessionCookie_DomainLevelCount	'domain' attribute will be added to the session cookie if this global variable is set. For example, to set the cookie domain as ".mydomain.com" for an application at "test.myserver.mydomain.com", set this global variable to "2". The value should be greater than 1 - if a lower value is specified, proxy will use 2 as the value.

Pre-defined variables

The Oracle Adaptive Access Manager Proxy supports the following pre-defined request variables:

Table 5–15 Pre-defined Variables Supported by the Proxy

Variable name	Description
%RESPONSE_CONTENT	This variable contains the contents of the entire response from the Web server for the current request.
%REQUEST_CONTENT	This variable contains the contents of the entire request from the client.
%QUERY_STRING	This variable contains the query string, starting with ?, for the current request URL.
%REQUEST_METHOD	HTTP method verb for the request: GET, POST, etc
%REMOTE_HOST	Hostname of the client or agent of the client
%REMOTE_ADDR	IP address of the client or agent of the client
%HTTP_HOST	The content of HTTP Host header
%URL	URL for the current request

5.6.1.8 Application

A single Oracle Adaptive Access Manager Proxy installation can be used to provide multifactor authentication for multiple Web application that run in one or more Web servers. In the Oracle Adaptive Access Manager Proxy configuration, an application is a grouping of interceptors defined for a single Web application.

Request and response interceptors can be defined outside of an application in the Oracle Adaptive Access Manager Proxy configuration file. These interceptors are called "global" interceptors and will be evaluated and executed prior to interceptors defined in applications.

5.6.2 Interception Process

When a request arrives, the Oracle Adaptive Access Manager Proxy evaluates request interceptors defined for the URL in the order they are defined in the configuration file. Similarly when on receiving response from the Web server, the Oracle Adaptive Access

Manager Proxy evaluates response interceptors defined for the URL in the order defined in the configuration file.

If the conditions in an interceptor evaluate to true, the Oracle Adaptive Access Manager Proxy will execute that interceptor i.e. execute the filters and action. After executing an interceptor, the Oracle Adaptive Access Manager Proxy will continue with the next interceptor only if the following conditions are met:

- no action is specified for the current interceptor
- post-exec-action attribute for the current interceptor is continue

Even if one of the conditions is not met the Oracle Adaptive Access Manager Proxy will stop evaluating subsequent interceptors.

It is highly recommended that "post-exec-action" attribute is specified for interceptors that don't define an action. For global interceptors (for example, the interceptors defined outside of any application), the default value of "post-exec-action" attribute is continue. For non-global interceptors, the default value is stop-intercept.

As mentioned earlier the Oracle Adaptive Access Manager Proxy configuration can contain multiple applications. While finding the list of interceptors to evaluate for a URL, only the following interceptors are considered:

- global interceptors that are defined outside of any application
- interceptors defined in the application associated with the current session

Each session will be associated with at most one application. If no application is associated with the current session (yet) when the proxy finds an interceptor in an application for the URL, it will associate the application with the current session.

If the current session already has an application associated, and if no interceptor is found in that application for the URL, the proxy will then look for intercepts in other applications. If an interceptor is found in another application for the URL, a new session will be created and the request will be associated with the new session.

5.6.3 Configuring Redirection to the Oracle Adaptive Access Manager Server Interface

The Oracle Adaptive Access Manager Proxy redirects the user to OAAM Server pages at appropriate times, for example to collect the password using OAAM Server, to run risk rules, etc. HTTP headers are used to exchange data between Oracle Adaptive Access Manager Proxy and OAAM Server. The following table lists OAAM Server pages referenced in the proxy configuration along with the details of HTTP headers used to pass data. It also lists the expected action to be taken by the proxy on the given conditions.

Table 5–16 OAAM Server Interface

URL	Condition	Action
Any request to OAAM Server page	On receiving request	Set header "BharosaAppId". OAAM Server will use this header value to select appropriate customizations (UI, rules, etc.).
loginPage.jsp or login.do	On receiving request to application login page	Redirect to this URL to use the Oracle Adaptive Access Manager login page instead of the application's login page.
password.do	Response contains headers userid, password (could be more depending upon the application)	Save the credentials from the response headers and post to the application

Table 5–16 (Cont.) OAAM Server Interface

URL	Condition	Action
login.do	Phase-1 only: After validating the credentials entered by the user.	Redirect to this URL to update the status in Oracle Adaptive Access Manager and run appropriate risk rules.
login.do	Phase-1 only: On receiving the request.	Set "userid" header to the userid entered by the user. Set "Login-Status" header to one of the following: success, wrong_password, invalid_user, user_disabled, system_error. Set "OAAM ServerPhase" header to "one".
updateLoginStatus.do	Phase-2 only: After validating the credentials entered by the user.	Redirect to this URL to update the status in Oracle Adaptive Access Manager and run appropriate risk rules
updateLoginStatus.do	Phase-2 only: On receiving request	Set "Login-Status" header to one of the following: success, wrong_password, invalid_user, user_disabled, system_error
updateLoginStatus.do challengeUser.do registerQuestions.do userPreferencesDone.do	Response header "Rules-Result" has value "allow"	The Oracle Adaptive Access Manager rules evaluated to permit the login. The proxy can permit access to the protected application URLs after this point.
updateLoginStatus.do challengeUser.do registerQuestions.do userPreferencesDone.do	Response header "Rules-Result" has value "block"	Either the application did not accept the login credentials or the Oracle Adaptive Access Manager rules evaluated to block the login. The proxy should logoff the session in the application, if login was successful. Then a login blocked message should be sent to the browser.
changePassword.do	Response contains headers "password", "newpassword" and "confirmpassword"	Save the passwords from the response headers and post to the application
loginFail.do	To display error message in OAAM Server page, like to display login blocked message	Redirect to this URL with appropriate "action" query parameter, like loginFail.do?action=block
logout.do	On completion of application session logout	Redirect to this URL to logout OAAM Server session
logout.do	On receiving response	Redirect to application logout URL to logoff application session, if it is not done already
resetPassword.do	Response contains headers "newpassword" and "confirmpassword"	Save the passwords from the response headers and post to the application
getUserInput.do	Response contains headers "BH_UserInput"	Save the user input and take appropriate action (like post to application, etc)
changeUserId.do	On receiving request	Add "newUserId" header

Table 5–16 (Cont.) OAAM Server Interface

URL	Condition	Action
changeUserId.do	On receiving response	Redirect to appropriate application page or send back saved application response
updateForgotPasswordStatus.do	Phase-2 only: After validating the forgot-password-credentials entered by the user.	Redirect to this URL to update the status in Oracle Adaptive Access Manager and run appropriate risk rules.
updateForgotPasswordStatus.do	Phase-2 only: On receiving request	Set "BH_FP-Status" header to one of the following: success, wrong_password, invalid_user, user_disabled, system_error.
updateForgotPasswordStatus.do challengeForgotPasswordUser.do	Response header "BH_FP-Rules-Result" has value "allow"	The Oracle Adaptive Access Manager rules evaluated to permit the forgot-password flow. The proxy can permit continuation of to forgot-password flow, perhaps to reset the password or allow the user login, depending on the application.
updateForgotPasswordStatus.do challengeForgotPasswordUser.do	Response header "BH_FP-Rules-Result" has value "block"	Either the application did not accept the forgot-password credentials or the Oracle Adaptive Access Manager rules evaluated to block the forgot-password flow. A login blocked message should be sent to the browser.
Any request to OAAM Server page	If the proxy needs to get a property value from OAAM Server. On receiving request	"BH_PropKeys" request header should be set to list of property names (separated by comma). OAAM Server will return the values in multiple response headers, one for each property. The return response header names will be of format: "BH_Property-<name>"

5.7 Application Discovery

Application discovery is the process of studying an existing Web application to author the proxy configuration to add multifactor authentication using the Oracle Adaptive Access Manager Universal Installation Option. Few logins attempts to the application would be made via the proxy to capture the HTTP traffic in each attempt. The captured HTTP traffic would be then be analyzed to author the proxy configuration. The Oracle Adaptive Access Manager Proxy should be set up to dump all the HTTP traffic through it to a file. Then a few logins/login attempts to the application should be made via the proxy. The captured HTTP traffic should be then be analyzed to author the proxy configuration.

5.7.1 Application Information

For application discovery process it is preferable to work with the Web application in customer's test environment, rather than the live application being used by users. If the test environment is not available for some reason, the live application can be used.

The following information is needed from the client for the application discovery process:

1. URL to login to the application.
2. Test user account credentials, including the data required in forgot password scenario. It will be best to get as many test accounts as possible, preferably at least 5 accounts, for uninterrupted discovery and testing. Note that during discovery process some accounts could become disabled, perhaps due to multiple invalid login attempts.
3. Contact (phone, email) to enable/reset test accounts

5.7.2 Setting Up the Oracle Adaptive Access Manager Proxy for Microsoft ISA

The Microsoft ISA server should be set up to publish the Web application under discovery i.e. creating a Web site publishing rule with appropriate parameters. During the application discovery process, the application will be accessed via Microsoft ISA, which hosts the Oracle Adaptive Access Manager Proxy for Microsoft ISA. Refer to the Microsoft ISA configuration document for details of setting up Microsoft ISA.

The Oracle Adaptive Access Manager Proxy for Microsoft ISA settings (registry values under HKLM\SOFTWARE\Bharosa\Proxy key) should be set as given in [Table 5–17](#) for the proxy to capture the HTTP traffic to the specified file. This HTTP traffic captured will later be used for analysis to author the proxy configuration.

Table 5–17 *Setting up the proxy*

Setting	Value
IgnoreUrlMappings	1
CaptureTraffic	1
TraceFilename	<filename>
TraceLevel	0x87
TraceToFile	1

It might be useful to capture the HTTP traffic for each scenario (like successful login attempt, wrong password, wrong username, disabled user, etc.) in separate files. `TraceFilename` setting should be updated to the desired filename before start of the scenario.

After application discovery is done, the proxy settings should be set as given in [Table 5–18](#) to restore the default Oracle Adaptive Access Manager Proxy for Microsoft ISA behavior.

Table 5–18 *Proxy settings after application discovery*

Setting	Value
IgnoreUrlMappings	0
CaptureTraffic	0
TraceFilename	<filename>
TraceLevel	0x7
TraceToFile	1

5.7.3 Setting Up the Oracle Adaptive Access Manager Proxy for Apache

For application discovery, the HTTP traffic needs to be captured through the proxy.

Table 5–19 shows the settings (in UIO_Settings.xml) to enable this mode of operation.

Table 5–19 Settings for Capturing HTTP

Settings	Value
IgnoreUrlMappings	1
CaptureTraffic	1

The `IgnoreUrlMappings` setting is used to disable URL interception of the HTTP traffic through the proxy.

The `CaptureTraffic` setting captures the HTTP traffic through the logger name `http` set to log level of `info`.

It might be useful to capture the HTTP traffic for each scenario (like successful login attempt, wrong password, wrong username, disabled user, and so on) in separate files. The log file name setting should be updated to the desired filename before the start of the scenario.

After application discovery is performed, the proxy settings should be set, as shown in Table 5–20, to restore the default Oracle Adaptive Access Manager Proxy for Apache behavior.

Table 5–20 Settings to restore default proxy behavior

Settings	Value
IgnoreUrlMappings	0
CaptureTraffic	0

5.7.4 Scenarios

Information should be collected for the following scenarios during the discovery process:

Login

1. URL that starts the login process
2. URL that contains the login form
3. Names of the input fields like username, password used to submit the credentials
4. URL to which the login form submits the credentials
5. Identifying successful login. The HTTP traffic dump needs to be studied carefully to derive this information. Here are few ways applications respond on successful login:
 - a. by setting a specific cookie in the credential submit response
 - b. by redirecting to a specific URL (like account summary, welcome page)
 - c. by responding with specific text
6. Identifying failure login with the reason for failure. This would often be derived by looking for certain text in the response to credential submit request.

Logout

1. URL that starts the logout process

2. URL that completes the logout process. In most cases the logout completes on receiving response to the logout start URL.

Change password

1. URL that starts the change password process
2. URL that contains the change password form
3. Names of the input fields like password, new-password, confirm-password used to submit the change password request
4. URL to which the change password form submits the passwords
5. Identifying the status (success/failure) of the change password request. This would often be derived by looking for certain text in the response.

Reset password

Follow the same process as Change password.

Change LoginId

1. URL to which the login-id change is posted to the application
2. Names of the input fields like new-login used to submit the change password request.
3. Identifying the status (success/failure) of the change login-id request. On successful change login-id request, changeUserId.do page in OAAM Server should be called to update the login-id in the Oracle Adaptive Access Manager database.

Forgot password

Forgot-password options provided by the application should first be understood. Most applications ask for alternate ways to identity the user (account number/PIN, SSN/PIN, question/answer, etc.); some applications provide more than one option. Some applications let the user reset the password on successfully entering alternate credentials; others send a new password to the user by mail/email; and some other applications would require the user to call customer care. For each of the supported scenarios, the following data should be captured:

1. URL that starts the forgot-password process
2. URL that contains the forgot-password form
3. Names of the input fields and URLs to submit the forgot-password request
4. Identifying the status (success/failure) of the forgot-password request.

5.8 Samples

The Oracle Adaptive Access Manager Proxy configuration to add multifactor authentication to BigBank Web application is listed as follows:

For ISA proxy use:

```
<?xml version="1.0" encoding="utf-8"?>
<BharosaProxyConfig xmlns="http://bharosa.com/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://bharosa.com/ BharosaProxy.xsd ">
```

For Apache proxy use:

```

<?xml version="1.0" encoding="utf-8"?>
<BharosaProxyConfig xmlns="http://bharosa.com/">

  <Application id="BigBank">

    <RequestInterceptor id="AddAppIdToBharosauioRequests"
      desc="Add BharosaAppId header to each request to bharosauio"
      post-exec-action="continue">
      <Conditions>
        <VariableValue name="%URL"
          value="/bharosauio/"
          mode="begins-with"
          ignore-case="true"/>
      </Conditions>

      <Filters>
        <AddHeader name="BharosaAppId:" value="BigBank"/>
      </Filters>
    </RequestInterceptor>

    <!-- Phase-1: Use BigBank login form to collect credentials -->
    <!-- Phase-2: Use BharosaUIO login forms to collect credentials -->

    <!-- Disable this interceptor after phase one is retired -->
    <RequestInterceptor id="Phase1BigBankLoginPostRequest"
      desc="get the loginid from the post parameters"
      post-exec-action="continue" enabled="true">
      <RequestUrl url="/bigbank/login.do"/>

      <Conditions>
        <VariableValue name="%REQUEST_METHOD" value="post"/>
      </Conditions>

      <Filters>
        <ClearSession/>
        <SetVariable name="$WebUIOPhase" value="one"/>
        <SaveParam name="userid" variable="$userid"/>
      </Filters>
    </RequestInterceptor>

    <!-- Enable this interceptor after phase one is retired -->
    <RequestInterceptor id="Phase2RedirectBigBankLoginPageRequest"
      desc="Redirect BigBank login page requests to UIO login page"
      enabled="false">
      <RequestUrl url="/bigbank"/>
      <RequestUrl url="/bigbank/" />
      <RequestUrl url="/bigbank/loginPage.jsp"/>

      <Target action="redirect-client" url="/bharosauio/login.do"/>
    </RequestInterceptor>

    <RequestInterceptor id="Phase2BharosaLoginPageRequest"
      desc="Phase-2 loginid post request"
      post-exec-action="continue">
      <RequestUrl url="/bharosauio/login.do"/>

      <Conditions>
        <VariableValue name="%REQUEST_METHOD" value="post"/>
        <ParamPresent name="userid"/>
      </Conditions>
    </RequestInterceptor>
  </Application>
</BharosaProxyConfig>

```

```
<Not>
  <ParamPresent name="password"/>
</Not>
</Conditions>

<Filters>
  <ClearSession/>
  <SetVariable name="$WebUIOPhase" value="two"/>
</Filters>
</RequestInterceptor>

<ResponseInterceptor id="Phase2PassowrdPageResponse"
  desc="Save userid, decoded password from Bharosa response">
  <ResponseUrl url="/bharosauio/password.do"/>

  <Conditions>
    <HeaderPresent name="userid:" />
    <HeaderPresent name="password:" />
  </Conditions>

  <Filters>
    <SaveHeader name="userid:" variable="$userid"/>
    <SaveHeader name="password:" variable="$password"/>
  </Filters>

  <Target action="redirect-client"
    url="/bigbank/login.do"
    display-url="/bigbank/GetLoginPage"/>
</ResponseInterceptor>

<ResponseInterceptor id="GetBigBankLoginPageResponse"
  desc="Save hidden fields; then post login crdentials">
  <ResponseUrl url="/bigbank/GetLoginPage"/>

  <Filters>
    <SaveHiddenFields variable="%LoginPageHiddenParams"/>

    <AddHiddenFieldsParams variable="%LoginPageHiddenParams"/>
    <AddParam name="userid" value="$userid"/>
    <AddParam name="password" value="$password"/>

    <UnsetVariable name="$password"/>
  </Filters>

  <Target action="post-server" url="/bigbank/login.do"/>
</ResponseInterceptor>

<ResponseInterceptor id="InvalidLoginResponse"
  desc="Invalid login response from BigBank">
  <ResponseUrl url="/bigbank/login.do"/>

  <Conditions>
    <PageContainsText text="You have entered an invalid Login Id"/>
  </Conditions>

  <Filters>
    <SetVariable name="$Login-Credentials-Status"
      value="invalid_user"/>
    <SetVariable name="$Login-Continue-Url"
      value="%URL"/>
  </Filters>
</ResponseInterceptor>
```

```

        <SaveResponse variable="$Submit-Credentials-Response"/>
    </Filters>

    <Target action="redirect-client"
            url="/bharosauio/UpdateLoginStatusPage"/>
</ResponseInterceptor>

<ResponseInterceptor id="WrongPasswordResponse"
                    desc="Invalid login response from BigBank">
    <ResponseUrl url="/bigbank/login.do"/>

    <Conditions>
        <PageContainsText text="We do not recognize your password"/>
    </Conditions>

    <Filters>
        <SetVariable name="$Login-Credentials-Status"
                    value="wrong_password"/>
        <SetVariable name="$Login-Continue-Url"
                    value="%URL"/>
        <SaveResponse variable="$Submit-Credentials-Response"/>
    </Filters>

    <Target action="redirect-client"
            url="/bharosauio/UpdateLoginStatusPage"/>
</ResponseInterceptor>

<ResponseInterceptor id="LoginSuccessResponse"
                    desc="Login success response from BigBank">
    <ResponseUrl url="/bigbank/activity.do"/>
    <ResponseUrl url="/bigbank/login.do"/>

    <Conditions>
        <NotVariableValue name="$Login-Status" value="In-Session"/>
        <PageContainsText text="/bigbank/images/success.gif"/>
    </Conditions>

    <Filters>
        <SetVariable name="$Login-Credentials-Status" value="success"/>
        <SetVariable name="$Login-Continue-Url" value="%URL"/>
        <SaveResponse variable="$Submit-Credentials-Response"/>
    </Filters>

    <Target action="redirect-client"
            url="/bharosauio/UpdateLoginStatusPage"/>
</ResponseInterceptor>

<RequestInterceptor id="Phase1UpdateLoginStatusPageRequest"
                    desc="Update Bharosa Tracker with the login status">
    <RequestUrl url="/bharosauio/UpdateLoginStatusPage"/>

    <Conditions>
        <VariableValue name="$WebUIOPhase" value="one"/>
    </Conditions>

    <Filters>
        <AddHeader name="WebUIOPhase:" value="$WebUIOPhase"/>
        <AddHeader name="userid:" value="$userid"/>
        <AddHeader name="Login-Status:"
                    value="$Login-Credentials-Status"/>
    </Filters>

```

```

</Filters>

<!-- Any interceptors for /bigbank/login.do will not run because we are
doing get-server. -->
<Target action="get-server" url="/bharosauio/login.do"/>
</RequestInterceptor>

<RequestInterceptor id="Phase2UpdateLoginStatusPageRequest"
    desc="Update Bharosa Tracker with the login status">
    <RequestUrl url="/bharosauio/UpdateLoginStatusPage"/>

    <Filters>
        <AddHeader name="Login-Status:"
            value="{$Login-Credentials-Status}"/>
    </Filters>

    <Target action="get-server"
        url="/bharosauio/updateLoginStatus.do"/>
</RequestInterceptor>

<ResponseInterceptor id="AllowLoginResponse"
    desc="Tracker returned 'allow' - continue with login">
    <ResponseUrl url="/bharosauio/UpdateLoginStatusPage"/>
    <ResponseUrl url="/bharosauio/updateLoginStatus.do"/>
    <ResponseUrl url="/bharosauio/challengeUser.do"/>
    <ResponseUrl url="/bharosauio/registerQuestions.do"/>
    <ResponseUrl url="/bharosauio/userPreferencesDone.do"/>

    <Conditions>
        <HeaderValue name="Rules-Result:" value="allow"/>
    </Conditions>

    <Filters>
        <SetVariable name="{$Login-Status}" value="In-Session"/>
    </Filters>

    <Target action="send-to-client"
        html="{$Submit-Credentials-Response}"
        display-url="{$Login-Continue-Url}"/>
</ResponseInterceptor>

<ResponseInterceptor id="Phase1FailLoginResponse"
    desc="BigBank failed the login">
    <ResponseUrl url="/bharosauio/UpdateLoginStatusPage"/>
    <ResponseUrl url="/bharosauio/updateLoginStatus.do"/>
    <ResponseUrl url="/bharosauio/challengeUser.do"/>
    <ResponseUrl url="/bharosauio/registerQuestions.do"/>
    <ResponseUrl url="/bharosauio/userPreferencesDone.do"/>

    <Conditions>
        <VariableValue name="{$WebUIOPhase}" value="one"/>
        <NotVariableValue name="{$Login-Credentials-Status}"
            value="success"/>
        <HeaderValue name="Rules-Result:" value="block"/>
    </Conditions>

    <Filters>
        <UnsetVariable name="{$Login-Status}"/>
    </Filters>

```



```

    <Target action="send-to-client"
        html="$Submit-Credentials-Response"
        display-url="$Login-Continue-Url"/>
</ResponseInterceptor>

<ResponseInterceptor id="FailLoginResponse"
    desc="BigBank failed the login">
    <ResponseUrl url="/bharosauio/UpdateLoginStatusPage"/>
    <ResponseUrl url="/bharosauio/updateLoginStatus.do"/>
    <ResponseUrl url="/bharosauio/challengeUser.do"/>
    <ResponseUrl url="/bharosauio/registerQuestions.do"/>
    <ResponseUrl url="/bharosauio/userPreferencesDone.do"/>

    <Conditions>
        <HeaderValue name="Rules-Result:" value="block"/>
        <NotVariableValue name="$Login-Credentials-Status"
            value="success"/>
    </Conditions>

    <Filters>
        <UnsetVariable name="$Login-Status"/>
    </Filters>

    <Target action="redirect-client"
        url="/bharosauio/loginPage.jsp?action=invalid_user"/>
</ResponseInterceptor>

<ResponseInterceptor id="BlockLoginResponse"
    desc="BigBank passed login but tracker returned 'block'">
    <ResponseUrl url="/bharosauio/UpdateLoginStatusPage"/>
    <ResponseUrl url="/bharosauio/updateLoginStatus.do"/>
    <ResponseUrl url="/bharosauio/challengeUser.do"/>
    <ResponseUrl url="/bharosauio/registerQuestions.do"/>
    <ResponseUrl url="/bharosauio/userPreferencesDone.do"/>

    <Conditions>
        <HeaderValue name="Rules-Result:" value="block"/>
    </Conditions>

    <Filters>
        <UnsetVariable name="$Login-Status"/>
    </Filters>

    <!-- /bigbank/LoginBlockedPage isn't a real page. The request will be
intercepted and redirected. -->
    <Target action="redirect-client" url="/bigbank/LoginBlockedPage"/>
</ResponseInterceptor>

<RequestInterceptor id="LoginBlockedPageRequest"
    desc="logoff the session in BigBank">
    <RequestUrl url="/bigbank/LoginBlockedPage"/>

    <Target action="get-server" url="/bigbank/logout.do"/>
</RequestInterceptor>

<ResponseInterceptor id="Phase1LoginBlockedPageResponse"
    desc="BigBank approved; but Bharosa blocked the login"
    post-exec-action="stop-intercept">
    <ResponseUrl url="/bigbank/LoginBlockedPage"/>

```

```
<Conditions>
  <VariableValue name="$WebUIOPhase" value="one"/>
</Conditions>

<Filters>
  <ClearSession/>
</Filters>

  <Target action="redirect-client"
    url="/bharosauio/loginFail.do?action=block"/>
</ResponseInterceptor>

<ResponseInterceptor id="Phase2LoginBlockedPageResponse"
  desc="BigBank approved; but Bharosa blocked the login">
  <ResponseUrl url="/bigbank/LoginBlockedPage"/>

  <Filters>
    <ClearSession/>
  </Filters>

  <Target action="redirect-client"
    url="/bharosauio/loginPage.jsp?action=block"/>
</ResponseInterceptor>

<ResponseInterceptor id="LogoutPageResponse"
  desc="Bharosa logout selected; logoff BigBank session ">
  <ResponseUrl url="/bharosauio/logout.do"/>

  <Target action="redirect-client" url="/bigbank/logout.do"/>
</ResponseInterceptor>

<ResponseInterceptor id="Phase1LogoffPageResponse"
  desc="Logoff - clear Bharosa proxy session"
  post-exec-action="stop-intercept">
  <ResponseUrl url="/bigbank/logout.do"/>

  <Conditions>
    <VariableValue name="$WebUIOPhase" value="one"/>
  </Conditions>

  <Filters>
    <ClearSession/>
  </Filters>
</ResponseInterceptor>

<ResponseInterceptor id="Phase2LogoffPageResponse"
  desc="Logoff - clear Bharosa proxy session">
  <ResponseUrl url="/bigbank/logout.do"/>

  <Filters>
    <ClearSession/>
  </Filters>

  <Target action="redirect-client"
    url="/bharosauio/loginPage.jsp"/>
</ResponseInterceptor>
</Application>
</BharosaProxyConfig>
```

Configuring OAAM Server

This chapter provides information on customizing the client-facing OAAM Server Web application. The Oracle Adaptive Access Manager Universal Installation Option offers multifactor authentication to Web applications without requiring any change to the application code. The OAAM Server configuration is specific to the Universal Installation Option deployment. Refer to the architectural diagram ([Figure 6-1](#)) for the components involved.

The user interface provided by the OAAM Server Web application can be easily customized to achieve the look-n-feel of the customer applications. This chapter is intended for integrators who install and configure OAAM Server to support one or more Web application authentication and user registration flows.

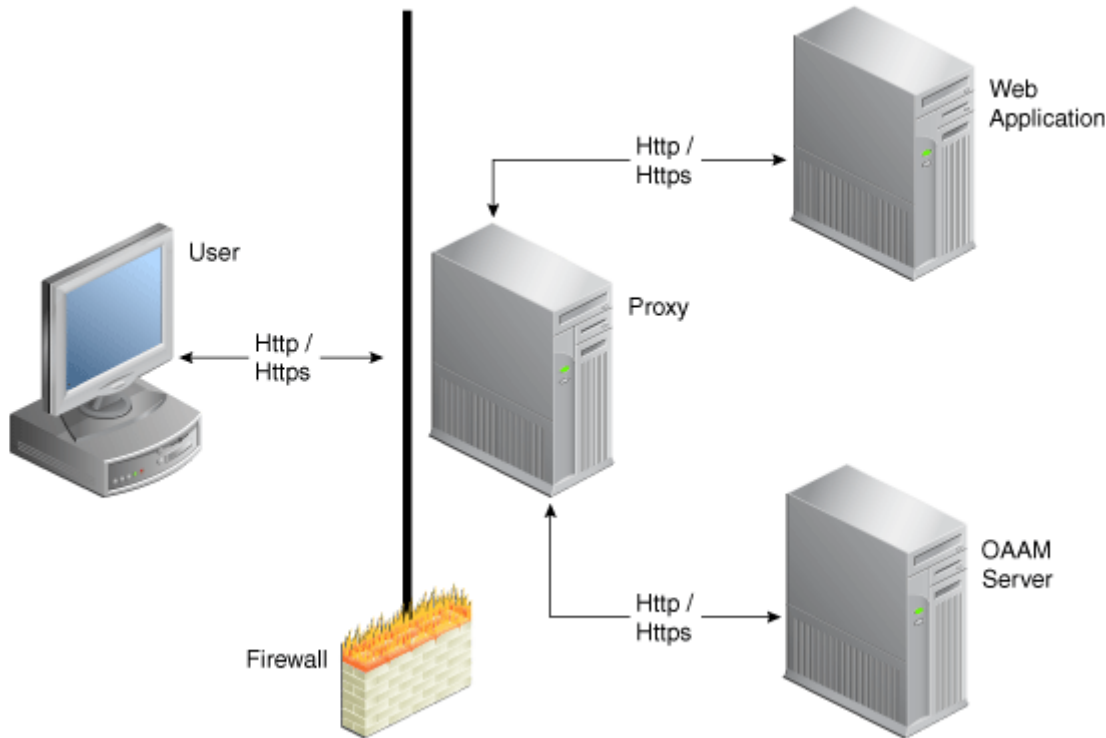
This chapter contains the following sections:

- [Architecture](#)
- [OAAM Server Settings](#)
- [Determining Application ID and User Group](#)
- [Customizing User Interface Branding](#)
- [Configuring Application Properties](#)

6.1 Architecture

[Figure 6-1](#) shows an Oracle Adaptive Access Manager Universal Installation Option deployment.

Figure 6–1 Universal Installation Deployment



The OAAM Server proxy intercepts the HTTP traffic between the client (browser) and the server (Web application) and performs appropriate actions, such as redirecting to OAAM Server, to provide multifactor authentication and authorization. OAAM Server in turn communicates with OAAM Admin to assess the risk and takes the appropriate actions, such as permitting the login, challenging the user, blocking the user, and other actions.

6.2 OAAM Server Settings

OAAM Server configuration is controlled through property files.

Configuration Files

Use the following property files to configure OAAM Server:

- `bharosa_server.properties` or `bharosauio_client.properties`
 - for client-configured properties (any properties that have been customized for a specific deployment)
 - for Universal Installation Option system /device configurations. These properties deal with the structural changes in the overall application. It is where the header, footer, and CSS properties are located.
- `client_resource_<locale>.properties` where `<locale>` is the locale string for which you wish to use the custom values (`en`, `es`, and others)
 - for client-configured properties that are configurable for each locale being supported. `<locale>` is the locale string for which you wish to use the custom values (`en`, `es`, and others).

- for Universal Installation Option messaging and page content configuration. For example, page titles, links at the bottom of the pages, page messages, error message, and confirmation messages.

In the deployed application, the `bharosa_server.properties` and `bharosauio_client.properties` files are located in the `web-inf/classes` directory.

The `client_resource_<locale>.properties` is created by the administrator customizing the application to contain locale-specific properties.

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, refer to [Chapter 12, "Customizing Oracle Adaptive Access Manager."](#)

6.3 Determining Application ID and User Group

The initial steps to configure and customize OAAM Server are:

1. Determine the application ID of each application being secured.
2. Assign default user groups for each application being secured.

6.3.1 Determining the Application ID

The Universal Installation Option can be placed in front of multiple applications, and customized to work with each one as required. Determine how many applications are to be configured, assign each application an Application ID. This Application ID is the same one used to configure the Proxy (see [Chapter 5, "Oracle Adaptive Access Manager Proxy"](#)). In many cases applications are referred to internally by some name or abbreviation, so an integrator configuring OAAM Server might want to use that name. For an example, if the client has two applications, one wholesale banking application and one retail banking application, the integrator might choose to use `wholesale` and `retail` as the Application IDs for the two applications.

The Proxy will send the `AppId` to OAAM Server as needed via an HTTP header. This `AppId` is then used to determine which configuration is used when displaying pages to the client. OAAM Server is configured by a set of properties which we will discuss in more detail later. An example of how `AppId` is used in a property definition is shown as follows:

```
bharosa.uio.appId1.default.user.group=applGroup
```

The bold "**appId1**" is the location in the property where the `AppId` is used to configure application specific values.

6.3.2 Determining Default User Groups

Each application can be configured to have a unique default user group. This is the group that a user of that application will be associated with as their Organization ID when first created in the Oracle Adaptive Access Manager database. Similarly, it will be the Organization ID used to attempt to load user information from the database when a user attempts to log in to the application.

As used in the previous example the property for Organization ID looks as follows:

```
bharosa.uio.appId1.default.user.group=app1Group
bharosa.uio.appId2.default.user.group=app2Group
```

In the example, two Organization IDs are defined to two different applications. The application with an `AppId` of "appId 1" has been assigned the Organization ID of

"app1Group" and the application with an AppId of "appId2" has been assigned the Organization ID of "app2Group".

6.4 Customizing User Interface Branding

The OAAM Server user interface branding is customized in several ways.

- Custom header / footer files
- Custom CSS file
- Custom properties for page content and messaging

6.4.1 Custom Header / Footer

OAAM Server provides the ability to create custom header and footer files for applications being secured. The header and footer files are JSP and can contain any HTML or JSP code required to replicate the look of the application being secured. All the customer resources (JSP files, image files, HTML, and others) should be copied into the deployed application directories along with the OAAM Server Web application.

The header (header.jsp) and footer (footer.jsp) files should contain only content html, all page related tags (<html>, <head>, <body>, etc) are already provided by OAAM Server. As a simple example we will create a header and footer that contain a single image each, to be used as the header and footer of an application called "appId1".

Copy the following code into a file called header.jsp for the header.

```
/client/app1/header.jsp
  
```

Copy the following code into a file called footer.jsp for the footer.

```
/client/app1/footer.jsp
  
```

These files will be housed in the "/client/app1/" directory within the Web application.

To associate these files with the application we would add the following properties to `client_resource_<locale>.properties`:

```
bharosa.uio.appId1.header = /client/app1/header.jsp
bharosa.uio.appId1.footer = /client/app1/footer.jsp
```

6.4.2 Custom CSS

OAAM Server styles are controlled through a single CSS file, `bharosa_uio.css`, located in the `css` directory. These styles can be overridden by including a custom CSS file. Much like the header and footer example show previously, you can create your own file and include that file on an application or global level through properties. Refer to [Section 6.5, "Configuring Application Properties."](#)

In this example we will override the font-family of the default body style definition.

The body style in `bharosa_uio.css` is defined as follows:

```
body{
  background-color:#ffffff;
  font-size:12px;
  color:#000000;
  font-family:arial, helvetica, sans-serif;
```

```
margin:0px 0px 0px 0px
}
```

Now to use our newly created file, we will add the following property:

```
bharosa.uio.appId1.custom.css=/client/app1/css/app1.css
```

In this case, all we did was change helvetica to the primary font-family in our "appId1" application. Any style defined in bharosa_uio.css can be overridden in this manner if required.

6.4.3 Custom Content and Messaging

OAAM Server pages have a variety of content and messaging sections. These sections can be customized by properties in `client_resource_<locale>.properties`. Some customizable items, like page title and message, are applicable for each page. While other items, like login blocked message, are specific to a particular page.

To change the page title on the login page in our example "appId1" application, we would add the following line to `client_resource_<locale>.properties`.

To change the page title on the login page in our example "appId1" application, we would add the following line to `client_resource_<locale>.properties`.

```
bharosa.uio.appId1.signon.page.title=Welcome to App1, please sign in.
```

The contents of error messages are also controlled in the same way. In the following example we will customize the error message displayed when a user has been blocked by security rules.

```
bharosa.uio.appId1.login.user.blocked = You are not authorized to login. Please
contact customer service at 1-888-555-1234.
```

6.5 Configuring Application Properties

An application in OAAM Server is made up of a grouping or set of properties. You can configure OAAM Server properties on a global or application specific level.

OAAM Server property names are prefixed with bharosa.uio. They are followed by the Application ID or "default" if the setting is global.

An "application-level" property is one that only effects a single application when there are more than one application defined in the properties.

For example,

```
# Global or Default header and footer definitions
# - Apply to all applications that don't specifically define their own
bharosa.uio.default.header = /globalHeader.jsp
bharosa.uio.default.footer = /globalFooter.jsp
# Application specific definitions
# - These values override the default settings
bharosa.uio.app1.header = /app1Header.jsp
bharosa.uio.app1.footer = /app1Footer.jsp
bharosa.uio.app2.footer = /app2Footer.jsp
```

In this example, app1 uses an "application-level" defined header and footer file, but app2 uses an "application-level" defined footer but a "global" or "default" defined header file.

The `bharosa.uio.default.header` property, shown as follows, defines the location of the header file.

```
bharosa.uio.default.header = /globalHeader.jsp
```

The property is used across all applications of the OAAM Server installation unless the specific application has another location specified.

In the case shown, "default" is used instead of the Application ID to designate the property as a global default. If the same property is not defined for an application; then, this value will be used.

6.5.1 Property Extension

In addition to configuring properties for each application, you can configure a set of properties that several applications have in common. You can then extend that set to customize the parameters that differ between the set of applications.

If you were to configure three applications that all use a single footer, but each has a unique header, you can include the following properties:

```
bharosa.uio.myAppGroup.footer = /myAppGroup/footer.jsp
```

```
bharosa.uio.appId1.extends=myAppGroup  
bharosa.uio.appId1.header=/client/app1/header.jsp
```

```
bharosa.uio.appId2.extends=myAppGroup  
bharosa.uio.appId2.header=/client/app2/header.jsp
```

```
bharosa.uio.appId3.extends=myAppGroup  
bharosa.uio.appId3.header=/client/app3/header.jsp
```

6.5.2 User-Defined Enums

The following is an example of an enum defining credentials displayed on the login screen of an OAAM Server implementation:

```
bharosa.uio.default.credentials.enum = Enum for Login Credentials  
bharosa.uio.default.credentials.enum.companyid=0  
bharosa.uio.default.credentials.enum.companyid.name=CompanyID  
bharosa.uio.default.credentials.enum.companyid.description=Company ID  
bharosa.uio.default.credentials.enum.companyid.inputname=companyid  
bharosa.uio.default.credentials.enum.companyid.maxlength=24  
bharosa.uio.default.credentials.enum.companyid.order=0  
bharosa.uio.default.credentials.enum.username=1  
bharosa.uio.default.credentials.enum.username.name=Username  
bharosa.uio.default.credentials.enum.username.description=Username  
bharosa.uio.default.credentials.enum.username.inputname=userid  
bharosa.uio.default.credentials.enum.username.maxlength=18  
bharosa.uio.default.credentials.enum.username.order=1
```

This set of properties defines one user-defined enum that contains two elements, each of which with five attributes. The "name" and "description" attributes are required to define any user-defined enum, other attributes are defined and used as needed by each individual use of a user-defined enum.

6.5.3 Overriding Existing User-Defined Enums

Overriding existing user-defined enums has some special cases. You may override any existing enum element's attribute value of the default application ID just as you would

any other property, but to change the value of an element's attribute in a single application using an `appId`, you must create the entire enum in that application using the appropriate `appId`.

For example, using the User Defined Enum defined in [Section 6.5.2, "User-Defined Enums,"](#) if we wanted to change "Company ID" to "Profile ID" for only one application (`appId1`), we would need to do the following:

For example, using the User Defined Enum, if we wanted to change "Company ID" to "Profile ID" for only one application (`appId1`), we would need to do the following:

```
bharosa.uio.appId1.credentials.enum = Enum for Login Credentials
bharosa.uio.appId1.credentials.enum.profileid=0
bharosa.uio.appId1.credentials.enum.profileid.name=ProfileID
bharosa.uio.appId1.credentials.enum.profileid.description=Profile ID
bharosa.uio.appId1.credentials.enum.profileid.inputname=profileid
bharosa.uio.appId1.credentials.enum.profileid.maxlength=20
bharosa.uio.appId1.credentials.enum.profileid.order=0
bharosa.uio.appId1.credentials.enum.username=1
bharosa.uio.appId1.credentials.enum.username.name=Username
bharosa.uio.appId1.credentials.enum.username.description=Username
bharosa.uio.appId1.credentials.enum.username.inputname=userid
bharosa.uio.appId1.credentials.enum.username.maxlength=18
bharosa.uio.appId1.credentials.enum.username.order=1
```

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties or enums, refer to [Chapter 12, "Customizing Oracle Adaptive Access Manager."](#)

6.5.4 Disabling Elements

To disable any already defined element in a user-defined enum, simply add an "enabled" attribute with a value of "false". Using the `appId1` credentials enum from [Section 6.5.3, "Overriding Existing User-Defined Enums,"](#) we would add the following line to remove "Profile ID" from the elements used by the application:

```
bharosa.uio.appId1.credentials.enum.profileid.enabled=false
```

Virtual Authentication Device Properties

OAAM Server provides end users a secure method to enter sensitive credentials online. OAAM Server is comprised of multiple secure interfaces. There are many security technologies employed in the OAAM Server user interfaces.

Each OAAM Server interface is a virtual authentication device (VAD). Each VAD has its own unique set of security features that make it much more than a mere image on a web page.

Details on the virtual authentication device properties are provided in this chapter for your reference.

- [Property Files](#)
- [Authentication Devices and Background Images](#)
- [Display and Security Feature Properties](#)
- [Accessibility](#)
- [KeySets](#)
- [Localization](#)

7.1 Property Files

Virtual authentication devices uses the following files:

- **bharosa_server.properties** - file where custom properties would be added for virtual authentication devices, KeySet definitions used in the KeyPad and PinPad devices, and configuration properties that are not localized (translated).
- **client_resource_<locale>.properties** - files to be created by the administrator customizing the application to contain locale-specific properties such as translated displayed messages. The locale identifier consists of at least a language identifier, and a region identifier (if required). For example, the custom properties file for US English is `client_resource_en_US.properties`.

Note: Many of the properties related to the virtual authentication devices are in resource bundles so that they are capable of being localized. If the default value is in a "resource" file, then the override value should be placed in the client override file for resource bundle values (`client_resource.properties`).

7.2 Authentication Devices and Background Images

Virtual authentication devices are provided with Oracle Adaptive Access Manager as samples to use if you choose to. These samples are provided in English only.

A set of sample background images are also shipped with Oracle Adaptive Access Manager. For the images to be displayed, set the following properties:

```
vcrypt.user.image.dirlist.property.name=bharosa.image.dirlist  
bharosa.image.dirlist=<imagePath>
```

If any of the images are to be edited, make sure not to increase the physical dimensions or change the aspect ratio of the sample images because distortions will occur.

7.3 Display and Security Feature Properties

Virtual authentication devices are provided with Oracle Adaptive Access Manager as samples to use if you choose to. These samples are provided in English only. Source art and information in this chapter are provided to allow you to develop your own custom virtual authentication device frames, keys, personalization images and phrases.

Alteration of these samples is considered custom development.

The following sections outline the visual elements that are within the virtual authentication device visual display for each device and the unique security features of each authentication device.

Each virtual authentication device has its own unique security features. Some of these features can be enabled and disabled by editing the configuration properties in the `bharosa_server.properties`.

For visual display, important terms are:

- Enter Key Hotspot - Link area allowing user to submit data entered in the authentication device.
- Phrase - Personalized phrase assigned to the user at the time of registration. The phrase allows the user to ensure they are on their intended web site.
- Timestamp - Timestamp of when the image was generated, allowing the user to ensure the authentication device is current.

7.3.1 TextPad

TextPad is a personalized device for entering passwords or PIN using a regular keyboard. Like other virtual authentication devices, the TextPad helps in solving phishing problems. An example TextPad is shown in [Figure 7-1](#).

Figure 7–1 TextPad

7.3.1.1 TextPad Visual Elements

This section provides information on the visual elements of TextPad.

Phrase (Caption)

```
bharosa.authentipad.textpad.caption.personalize = true
bharosa.authentipad.textpad.caption.x = 14
bharosa.authentipad.textpad.caption.y = 203
bharosa.authentipad.textpad.caption.frame = false
bharosa.authentipad.textpad.caption.wrap = false
bharosa.authentipad.textpad.caption.width = 130
bharosa.authentipad.textpad.caption.height = 16
bharosa.authentipad.textpad.caption.font.name = Arial
bharosa.authentipad.textpad.caption.font.color = 000000
bharosa.authentipad.textpad.caption.font.type= 0
bharosa.authentipad.textpad.caption.font.size = 9
```

Timestamp

```
bharosa.authentipad.textpad.timestamp.x = 25
bharosa.authentipad.textpad.timestamp.y = 165
bharosa.authentipad.textpad.timestamp.width = 132
bharosa.authentipad.textpad.timestamp.height = 16
bharosa.authentipad.textpad.timestamp.frame = false
bharosa.authentipad.textpad.timestamp.wrap = false
bharosa.authentipad.textpad.timestamp.font.name = Arial
bharosa.authentipad.textpad.timestamp.font.color = ffffff
bharosa.authentipad.textpad.timestamp.font.type= 0
bharosa.authentipad.textpad.timestamp.font.size = 9
```

Enter Key Hotspot

```
bharosa.authentipad.textpad.enterkey.x=98
bharosa.authentipad.textpad.enterkey.y=181
```

```
bharosa.authentipad.textpad.enterkey.width=45
bharosa.authentipad.textpad.enterkey.height=19
bharosa.authentipad.textpad.enterkey.label=enter
bharosa.authentipad.textpad.enterkey.enable=true
```

7.3.1.2 TextPad Authenticator Properties

Table 7–1 lists the TextPad Authenticator Properties

Table 7–1 TextPad Authenticator Properties

Feature	Property
Default BG (Can be application specific)	bharosa.uio.<appId>.DeviceTextPad.default.image = textpad_bg/UIO_BG.jpg
Password Frame File (Can be application specific)	bharosa.uio.<appId>.password.DeviceTextPad.frame =
Challenge Frame File (Can be application specific)	bharosa.uio.<appId>.<challengeType>.DeviceTextPad.frame = Note: Challenge type can be any configured challenge type (ChallengeQuestion, ChallengeEmail, and others)
Registration Frame File (Can be application specific)	bharosa.uio.<appId>.register.DeviceTextPad.frame = textpad_bg/TP_O_ preview.png
User Preferences Frame File (Can be application specific)	bharosa.uio.<appId>.userpreferences.DeviceTextPad.frame = textpad_bg/TP_ O_preview.png

7.3.2 QuestionPad

QuestionPad is a personalized device for entering answers to challenge questions using a regular keyboard. The QuestionPad is capable of incorporating the challenge question into the Question image. Like other Adaptive Strong Authentication devices, QuestionPad also helps in solving the phishing problem. An example QuestionPad is shown in [Figure 7–2](#).

Figure 7-2 QuestionPad

7.3.2.1 QuestionPad Visual Elements

This section provides information on the visual elements of QuestionPad.

Note: In 10.1.4.5 and above, the QuestionPad is a single line field.

Phrase (Caption)

```
bharosa.authentipad.questionpad.caption.personalize = true
bharosa.authentipad.questionpad.caption.x = 14
bharosa.authentipad.questionpad.caption.y = 203
bharosa.authentipad.questionpad.caption.frame = false
bharosa.authentipad.questionpad.caption.wrap = false
bharosa.authentipad.questionpad.caption.width = 130
bharosa.authentipad.questionpad.caption.height = 16
bharosa.authentipad.questionpad.caption.font.name = Arial
bharosa.authentipad.questionpad.caption.font.color = 000000
bharosa.authentipad.questionpad.caption.font.type= 0
bharosa.authentipad.questionpad.caption.font.size = 9
```

Timestamp

```
bharosa.authentipad.questionpad.timestamp.x = 25
bharosa.authentipad.questionpad.timestamp.y = 165
bharosa.authentipad.questionpad.timestamp.width = 132
bharosa.authentipad.questionpad.timestamp.height = 16
bharosa.authentipad.questionpad.timestamp.frame = false
bharosa.authentipad.questionpad.timestamp.wrap = false
bharosa.authentipad.questionpad.timestamp.font.name = Arial
bharosa.authentipad.questionpad.timestamp.font.color = ffffff
bharosa.authentipad.questionpad.timestamp.font.type= 0
bharosa.authentipad.questionpad.timestamp.font.size = 9
```

Question Text

```
bharosa.authentipad.questionpad.question.x = 9
bharosa.authentipad.questionpad.question.y = 32
bharosa.authentipad.questionpad.question.width = 132
bharosa.authentipad.questionpad.question.height = 62
bharosa.authentipad.questionpad.question.frame = false
bharosa.authentipad.questionpad.question.wrap = true
bharosa.authentipad.questionpad.question.font.name = Arial
bharosa.authentipad.questionpad.question.font.color = 000000
bharosa.authentipad.questionpad.question.font.type= 0
bharosa.authentipad.questionpad.question.font.size = 9
```

Enter Key Hotspot

```
bharosa.authentipad.questionpad.enterkey.x=98
bharosa.authentipad.questionpad.enterkey.y=181
bharosa.authentipad.questionpad.enterkey.width=45
bharosa.authentipad.questionpad.enterkey.height=19
bharosa.authentipad.questionpad.enterkey.label=enter
bharosa.authentipad.questionpad.enterkey.enable=true
```

Visible Text Input or Password (Non-Visible) Input Setting

The following property in `client_resource_<locale>.properties` determines whether the QuestionPad is set for visible text input or password (non-visible) input.

```
bharosa.authentipad.questionpad.datafield.input.type
```

Valid values are text and password.

7.3.2.2 QuestionPad Authenticator Properties

[Table 7–2](#) lists the QuestionPad Authenticator Properties

Table 7–2 QuestionPad Authenticator Properties

Feature	Property
Default BG (Can be application specific)	<code>bharosa.uio.<appId>.DeviceQuestionPad.default.image = textpad_bg/UIO_BG.jpg</code>
Challenge Frame File (Can be application specific)	<code>bharosa.uio.<appId>.<challengeType>.DeviceQuestionPad.frame =</code> Note: Challenge type can be any configured challenge type (ChallengeQuestion, ChallengeEmail, and others)

7.3.3 Keypad

KeyPad is a personalized graphics keyboard, which can be used to enter alphanumeric and special character that can be enter using a traditional keyboard. KeyPad is ideal for entering passwords and other sensitive data. For example, credit card numbers can be entered. An example KeyPad is shown in [Figure 7–3](#).

Figure 7-3 KeyPad

7.3.3.1 KeyPad Visual Elements

This section provides information on the visual elements of KeyPad.

Phrase (Caption)

```
bharosa.authentipad.keypad.caption.personalize = true
bharosa.authentipad.keypad.caption.x = 240
bharosa.authentipad.keypad.caption.y = 206
bharosa.authentipad.keypad.caption.frame = false
bharosa.authentipad.keypad.caption.wrap = false
bharosa.authentipad.keypad.caption.width = 130
bharosa.authentipad.keypad.caption.height = 16
bharosa.authentipad.keypad.caption.font.name = Arial
bharosa.authentipad.keypad.caption.font.color = 000000
bharosa.authentipad.keypad.caption.font.type= 0
bharosa.authentipad.keypad.caption.font.size = 9
```

Timestamp

```
bharosa.authentipad.keypad.timestamp.x = 110
bharosa.authentipad.keypad.timestamp.y = 202
bharosa.authentipad.keypad.timestamp.width = 132
bharosa.authentipad.keypad.timestamp.height = 16
bharosa.authentipad.keypad.timestamp.frame = false
bharosa.authentipad.keypad.timestamp.wrap = false
bharosa.authentipad.keypad.timestamp.font.name = Arial
bharosa.authentipad.keypad.timestamp.font.color = ffffffff
bharosa.authentipad.keypad.timestamp.font.type= 0
bharosa.authentipad.keypad.timestamp.font.size = 9
```

Enter Key Hotspot

```
bharosa.authentipad.keypad.enterkey.x=292
bharosa.authentipad.keypad.enterkey.y=8
bharosa.authentipad.keypad.enterkey.width=50
bharosa.authentipad.keypad.enterkey.height=20
bharosa.authentipad.keypad.enterkey.label=enter
bharosa.authentipad.keypad.enterkey.enable=true
```

Backspace Key Hotspot

```
bharosa.authentipad.keypad.backspace.x=164
bharosa.authentipad.keypad.backspace.y=8
bharosa.authentipad.keypad.backspace.width=20
bharosa.authentipad.keypad.backspace.height=20
bharosa.authentipad.keypad.backspace.enable=true
```

Caps States

```
bharosa.authentipad.keypad.capslock.x=188
bharosa.authentipad.keypad.capslock.y=0
bharosa.authentipad.keypad.capslock.width=43
bharosa.authentipad.keypad.capslock.height=29
bharosa.authentipad.keypad.capslock.capsonimg=kp_v2_all_caps.jpg
bharosa.authentipad.keypad.capslock.capsshifting=kp_v2_first_caps.jpg
```

7.3.3.2 KeyPad Authenticator Properties

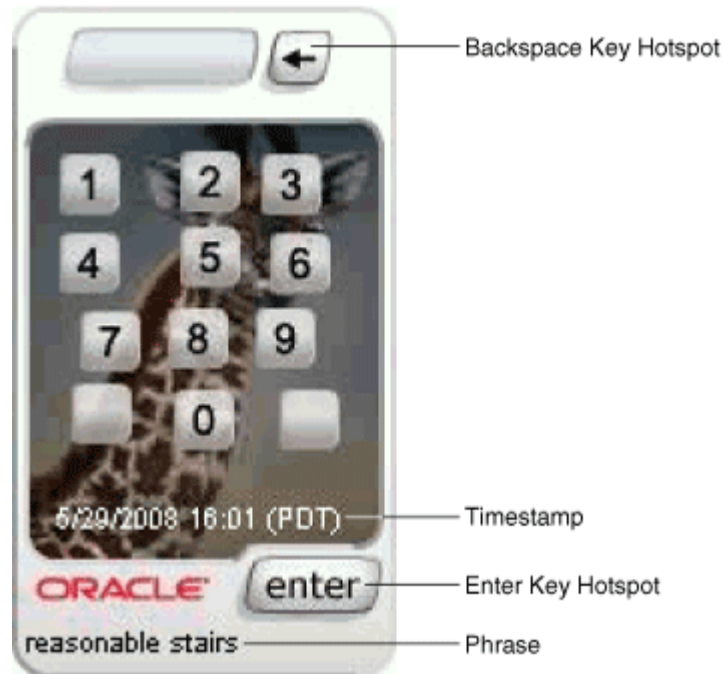
Table 7–3 lists the KeyPad Authenticator Properties

Table 7–3 KeyPad Authenticator Properties

Feature	Property
Default BG (Can be application specific)	bharosa.uio.<appId>.DeviceKeyPadFull.default.image = keypad_bg/UIO_BG.jpg
Password Frame File (Can be application specific)	bharosa.uio.<appId>.password.DeviceKeyPadFull.frame =
Challenge Frame File (Can be application specific)	bharosa.uio.<appId>.<challengeType>.DeviceKeyPadFull.frame = Note: Challenge type can be any configured challenge type (ChallengeQuestion, ChallengeEmail, and others)
Registration Frame File (Can be application specific)	bharosa.uio.<appId>.register.DeviceKeyPadFull.frame = alphapad_bg/kp_O_preview.png
User Preferences Frame File (Can be application specific)	bharosa.uio.<appId>.userpreferences.DeviceKeyPadFull.frame = alphapad_bg/kp_O_preview.png

7.3.4 PinPad

PinPad is a lightweight authentication device for entering a numeric PIN. An example PinPad is shown in [Figure 7–4](#).

Figure 7-4 PinPad

7.3.4.1 PinPad Visual Elements

This section provides information on the visual elements of PinPad.

Phrase (Caption)

```
bharosa.authentipad.pinpad.caption.personalize = true
bharosa.authentipad.pinpad.caption.x = 5
bharosa.authentipad.pinpad.caption.y = 206
bharosa.authentipad.pinpad.caption.frame = false
bharosa.authentipad.pinpad.caption.wrap = false
bharosa.authentipad.pinpad.caption.width = 130
bharosa.authentipad.pinpad.caption.height = 16
bharosa.authentipad.pinpad.caption.font.name = Arial
bharosa.authentipad.pinpad.caption.font.color = 000000
bharosa.authentipad.pinpad.caption.font.type= 0
bharosa.authentipad.pinpad.caption.font.size = 9
```

Timestamp

```
bharosa.authentipad.pinpad.timestamp.x = 15
bharosa.authentipad.pinpad.timestamp.y = 165
bharosa.authentipad.pinpad.timestamp.width = 132
bharosa.authentipad.pinpad.timestamp.height = 16
bharosa.authentipad.pinpad.timestamp.frame = false
bharosa.authentipad.pinpad.timestamp.wrap = false
bharosa.authentipad.pinpad.timestamp.font.name = Arial
bharosa.authentipad.pinpad.timestamp.font.color = ffffff
bharosa.authentipad.pinpad.timestamp.font.type= 0
bharosa.authentipad.pinpad.timestamp.font.size = 9
```

Enter Key Hotspot

```
bharosa.authentipad.pinpad.enterkey.x=78
bharosa.authentipad.pinpad.enterkey.y=182
```

```
bharosa.authentipad.pinpad.enterkey.width=49
bharosa.authentipad.pinpad.enterkey.height=20
bharosa.authentipad.pinpad.enterkey.label=enter
bharosa.authentipad.pinpad.enterkey.enable=true
```

Backspace Key Hotspot

```
bharosa.authentipad.pinpad.backspace.x=86
bharosa.authentipad.pinpad.backspace.y=8
bharosa.authentipad.pinpad.backspace.width=20
bharosa.authentipad.pinpad.backspace.height=20
bharosa.authentipad.pinpad.backspace.label=&lt;
bharosa.authentipad.pinpad.backspace.enable=true
```

7.3.4.2 PinPad Authenticator Properties

Table 7–4 lists the PinPad Authenticator Properties

Table 7–4 PinPad Authenticator Properties

Feature	Property
Default BG (Can be application specific)	bharosa.uio.default.DevicePinPad.default.image = pinpad_bg/UIO_BG.jpg
Password Frame File (Can be application specific)	bharosa.uio.<appId>.password.DevicePinPad.frame =
Challenge Frame File (Can be application specific)	bharosa.uio.<appId>.<challengeType>.DevicePinPad.frame = Note: Challenge type can be any configured challenge type (ChallengeQuestion, ChallengeEmail, and others)
Registration Frame File (Can be application specific)	bharosa.uio.<appId>.register.DevicePinPad.frame = pinpad_bg/PP_v02_frame_preview.png
User Preferences Frame File (Can be application specific)	bharosa.uio.<appId>.userpreferences.DevicePinPad.frame = pinpad_bg/PP_v02_frame_preview.png

7.4 Accessibility

Users who access using assistive techniques will need to use the accessible versions of the virtual authentication devices. Accessible versions of the TextPad, QuestionPad, KeyPad and PinPad are not enabled by default. If accessible versions are needed in a deployment, they can be enabled via properties.

The accessible versions of the pads contain tabbing, directions and ALT text necessary for navigation via screen reader and other assistive technologies.

To enable these versions, set the `is ADA compliant` flag to true.

For native integration the property to control the pads is

```
desertref.authentipad.isADACompliant
```

For UIO, the property to control the pads is

```
bharosa.uio.default.authentipad.is_ada_compliant
```

7.5 KeysSets

A KeySet is the configuration that defines what character keys are present on the virtual authentication device. KeySets are used by the Keypad and PinPad virtual authentication devices.

7.5.1 User Defined Enums Overview

KeySets are defined by a series user defined enums.

User-defined enums are a collection of properties that represent a list of items. Each element in the list may contain several different attributes. The definition of a user-defined enum begins with a property ending in the keyword ".enum" and has a value describing the use of the user-defined enum. Each element definition then starts with the same property name as the enum, and adds on an element name and has a value of a unique integer as an ID. The attributes of the element follow the same pattern, beginning with the property name of the element, followed by the attribute name, with the appropriate value for that attribute.

The following is an example of an enum defining credentials displayed on the login screen of an OAAM Server implementation:

```
bharosa.uio.default.credentials.enum = Enum for Login Credentials
bharosa.uio.default.credentials.enum.companyid=0
bharosa.uio.default.credentials.enum.companyid.name=CompanyID
bharosa.uio.default.credentials.enum.companyid.description=Company ID
bharosa.uio.default.credentials.enum.companyid.inputname=comapanyid
bharosa.uio.default.credentials.enum.companyid.maxlength=24
bharosa.uio.default.credentials.enum.companyid.order=0
bharosa.uio.default.credentials.enum.username=1
bharosa.uio.default.credentials.enum.username.name=Username
bharosa.uio.default.credentials.enum.username.description=Username
bharosa.uio.default.credentials.enum.username.inputname=userid
bharosa.uio.default.credentials.enum.username.maxlength=18
bharosa.uio.default.credentials.enum.username.order=1
```

This set of properties defines one user-defined enum that contains two elements, each of which with five attributes. The "name" and "description" attributes are required to define any user-defined enum, other attributes are defined and used as needed by each individual use of a user-defined enum.

7.5.2 KeySet Definition

The first enum defines the rows of the KeySet and points to an another enum describing the keys present in that row.

For example, the following enum defines the rows of keys in a PinPad:

```
bharosa.authentipad.pinpad.default.keyset.enum=Default PinPad Keyset Enum
bharosa.authentipad.pinpad.default.keyset.enum.row1=0
bharosa.authentipad.pinpad.default.keyset.enum.row1.name=Default PinPad Keyset Row
1
bharosa.authentipad.pinpad.default.keyset.enum.row1.description=Default PinPad
Keyset Row 1
bharosa.authentipad.pinpad.default.keyset.enum.row1.keys=bharosa.authentipad.pinpa
d.default.keyset.row1.enum
bharosa.authentipad.pinpad.default.keyset.enum.row1.order=1

bharosa.authentipad.pinpad.default.keyset.enum.row2=1
bharosa.authentipad.pinpad.default.keyset.enum.row2.name=Default PinPad Keyset Row
```

```

2
bharosa.authentipad.pinpad.default.keyset.enum.row2.description=Default PinPad
Keyset Row 2
bharosa.authentipad.pinpad.default.keyset.enum.row2.keys=bharosa.authentipad.pinpa
d.default.keyset.row2.enum
bharosa.authentipad.pinpad.default.keyset.enum.row2.order=2

bharosa.authentipad.pinpad.default.keyset.enum.row3=2
bharosa.authentipad.pinpad.default.keyset.enum.row3.name=Default PinPad Keyset Row
3
bharosa.authentipad.pinpad.default.keyset.enum.row3.description=Default PinPad
Keyset Row 3
bharosa.authentipad.pinpad.default.keyset.enum.row3.keys=bharosa.authentipad.pinpa
d.default.keyset.row3.enum
bharosa.authentipad.pinpad.default.keyset.enum.row3.order=3

bharosa.authentipad.pinpad.default.keyset.enum.row4=3
bharosa.authentipad.pinpad.default.keyset.enum.row4.name=Default PinPad Keyset Row
4
bharosa.authentipad.pinpad.default.keyset.enum.row4.description=Default PinPad
Keyset Row 4
bharosa.authentipad.pinpad.default.keyset.enum.row4.keys=bharosa.authentipad.pinpa
d.default.keyset.row4.enum
bharosa.authentipad.pinpad.default.keyset.enum.row4.order=4

```

Each row is made of the following properties:

Table 7–5 Properties of Rows

Property	Description
name	Name of the row.
description	Description of the row.
keys	Enum identifier of the enum that defines the keys in the row.
order	The order the key resides in the row of keys.

In this case, the row1 enum is defined as follows:

```

bharosa.authentipad.pinpad.default.keyset.row1.enum=Default Pinpad Keyset Row 1
bharosa.authentipad.pinpad.default.keyset.row1.enum.key1=0
bharosa.authentipad.pinpad.default.keyset.row1.enum.key1.name=1
bharosa.authentipad.pinpad.default.keyset.row1.enum.key1.description=1
bharosa.authentipad.pinpad.default.keyset.row1.enum.key1.value=1
bharosa.authentipad.pinpad.default.keyset.row1.enum.key1.shiftvalue=1
bharosa.authentipad.pinpad.default.keyset.row1.enum.key1.image=kp_v2_1.png
bharosa.authentipad.pinpad.default.keyset.row1.enum.key1.order=1

bharosa.authentipad.pinpad.default.keyset.row1.enum.key2=1
bharosa.authentipad.pinpad.default.keyset.row1.enum.key2.name=2
bharosa.authentipad.pinpad.default.keyset.row1.enum.key2.description=2
bharosa.authentipad.pinpad.default.keyset.row1.enum.key2.value=2
bharosa.authentipad.pinpad.default.keyset.row1.enum.key2.shiftvalue=2
bharosa.authentipad.pinpad.default.keyset.row1.enum.key2.image=kp_v2_2.png
bharosa.authentipad.pinpad.default.keyset.row1.enum.key2.order=2

bharosa.authentipad.pinpad.default.keyset.row1.enum.key3=2
bharosa.authentipad.pinpad.default.keyset.row1.enum.key3.name=3
bharosa.authentipad.pinpad.default.keyset.row1.enum.key3.description=3
bharosa.authentipad.pinpad.default.keyset.row1.enum.key3.value=3

```

```
bharosa.authentipad.pinpad.default.keyset.row1.enum.key3.shiftvalue=3
bharosa.authentipad.pinpad.default.keyset.row1.enum.key3.image=kp_v2_3.png
bharosa.authentipad.pinpad.default.keyset.row1.enum.key3.order=3
```

Each key is made of the following properties:

Table 7–6 Properties of Each Key

Property	Description
name	Name of the key.
description	Description of the key.
value	The character value the key represents when clicked.
shiftvalue	The character value the key represents when in caps mode.
image	The image file name that will be used to display the visual representation of the key.
order	The order the key resides in the row of keys.

7.6 Localization

This section contains information on customizing the application/virtual devices to contain locale-specific properties.

7.6.1 Enabling Localization

To enable locale-specific customizations, you must perform the following steps:

1. Create a client resource override file, `client_resource_<locale>.properties` file. `<locale>` is the locale for which you wish to use the custom values (en, es, and others)
2. Using the Properties Editor, set the value of `bharosa.config.resourcebundle.clientoverride` to `client_resource_<locale>.properties`.

The default value of this property is `client_resource`.

The `client_resource_<locale>.properties` file should contain:

- Client-configured properties that are configurable for each locale being supported.
- Messaging and page content configuration for the UIO system. For example, page titles, links at the bottom of the pages, page messages, error message, and confirmation messages.

7.6.2 Configuring Words Used in the Authenticator Caption

During initial registration a user is assigned a `word:word` pair for his Keypad that is generated randomly from word list properties. In English the `word:word` pairs are in the form, `adjective:noun`.

In the English version of Oracle Adaptive Access Manager, there are several hundred values in the word lists. In all other languages it is necessary for the installer to enhance the brief word lists provided.

To add words to the word lists, in `client_resource_fr.properties`, modify the `bharosa.user.caption.word1.list` and `bharosa.user.caption.word2.list` properties.

7.6.3 Localizing the KeyPad

Localization of the KeyPad may have issues since not all languages have the same number of characters. Portuguese for example has special characters not found in English. The key layout may be a bit different when these character keys are added. When adding keys to the layout it is vital that there is still enough free space around the keys to allow the "jitter" to function. General best practice is a space at least as large as a single key all the way around the bank of keys when they are positioned in the center of the jitter area. The source art contains notes with the pixel sizes for this area.

7.6.4 Configuring Enter on the Authenticator Forgot Password Page

To configure **Enter** to be in a local-specific language, modify the property, `bharosa.uio.default.forgotpassword.primary.page.message=property`.

7.6.5 Configuring Tooltip for TextPad's Enter Button

To configure **Enter** in a locale-specific language for TextPad's tooltip, modify the property, `bharosa.authentipad.textpad.enterkey.label=enterproperty`.

Part III

Features Integrations

Part III provides instructions and reference material for feature integrations.

It contains the following chapters:

- [Chapter 8, "Configurable Actions"](#)
- [Chapter 9, "OTP Anywhere"](#)
- [Chapter 10, "Flash Fingerprinting"](#)
- [Chapter 11, "Device Registration"](#)

Configurable Actions

Oracle Adaptive Access Manager provides Configurable Actions, a feature which allows users to create new supplementary actions that are triggered based on the result action and/or based on the risk scoring after a checkpoint execution. This section describes how to integrate a Configurable Action with the Oracle Adaptive Access Manager software.

8.1 Integration

To add a new Configurable Action, perform the following tasks:

1. Develop the Configurable Action by implementing the `com.bharosa.vcrypt.tracker.dynamicactions.intf.DynamicAction` java interface.

Note: In this step, implementing means writing java code based on the contract specified by the Java interface `com.bharosa.vcrypt.tracker.dynamicactions.intf.DynamicAction`.

While implementing the `com.bharosa.vcrypt.tracker.dynamicactions.intf.DynamicAction` java interface, the following two methods have to be coded:

- `getParameters()` - In this method, the code has to be written that returns the parameters used by the Configurable Action. Make sure that the size of the parameters array returned is the same as the number of parameters. Look at the sample configurable actions java code in Oracle Adaptive Access Manager Sample application.
 - `execute()` - In this method, code has to be written that performs the logic required by the Configurable Action. Configurable Action parameter values are passed in `actionParamValueMap` where the parameter name is the key and the `RuntimActionParamValue` object is the value. Use the appropriate `getXXXValue()` method to get the parameter value.
2. Compile your custom java classes that extend or implement Oracle Adaptive Access Manager classes by adding the jars from `$ORACLE_IDM_HOME\oaam\cli\lib` folder to the build classpath
 3. Test the implementation of the Configurable Action thoroughly.

Since Configurable Actions are standalone java classes, they can be tested with Unit Testing Methodology using JUnit framework.

For sample JUnit code for testing configurable actions, refer to the "[Sample JUnit Code](#)" section.

4. Compile the java class and create a jar file of the compiled class files.
5. Extend/customize Oracle Adaptive Access Manager to add the custom jar. Refer to [Section 4.1.2, "Customizing/Extending/Overriding Oracle Adaptive Access Manager Properties"](#) for steps for adding the custom jar to Oracle Adaptive Access Manager.
6. Restart OAAM Server and the OAAM Admin Server.
7. Log in to OAAM Admin and create an action definition entry for the newly deployed Configurable Action.
8. Make sure all the parameters required for the Configurable Action are displayed in the user interface.
9. Use the newly available Configurable Action by adding it to the required checkpoints. For more information on configuring Configurable Actions, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

8.2 Executing Configurable Actions in a Particular Order and Data Sharing

Configurable Actions can be used to implement chaining in such a way that

- they execute in a particular order
- data can be shared across these actions

Note: Sharing data across Configurable Actions involves writing java code and requires more effort than just a configuration task.

To be able to execute Configurable Actions in a particular order and share data:

1. Configure Configurable Actions as synchronous actions with the required order of execution in ascending order.

Note: A Configurable Action is executed only if the trigger criteria is met; therefore, make sure the trigger criteria is correct.

2. To share data, insert the data into the `actionContextMap` parameter of the Configurable Action's `execute()` method. Since the `actionContextMap` is a `Map`, it requires a key and value pair that represents the data to be shared.

Note: it is the implementer's responsibility to ensure that

- the duplicate keys are not used while inserting data
 - the same key is used when trying to access this shared data from another Configurable Action.
-
-

3. Ensure that the code can handle the case where the key is not present in the `actionContextMap`. This step must be performed to avoid errors or `NullPointerException` when the other action do not insert the value into the `actionContextMap`.

8.3 How to Test Configurable Actions Triggering

To test if configurable actions triggering:

1. Make sure there is a way to identify if the code in the Configurable Action is executed. This could be as simple as an entry in log file or an entry in database etc.
2. Enable debug level logging for `oracle.oaam` logger in OAAM Server.
3. Create an action template for the given Configurable Action.
4. Add the action to a Pre-Authentication checkpoint with trigger criteria as score between 0 and 1000.
5. Try logging in to OAAM Server as a user.
6. Check OAAM Server logs for the entry `Enter: executeAction(): Executing Action Instance.`
7. If there is no error then you will see a related log statement like `Exit: executeAction(): Action Instance.`
8. If there is an error, you will see a log statement like `Error: executeAction().`
9. Apart from these, check for a log entry or a database entry that is created by the Configurable Action itself

8.4 Sample JUnit Code

The following is an sample JUnit code for testing dynamic action:

```
public class TestDynamicActionsExecution extends TestCase {
    static Logger logger =
Logger.getLogger(TestDynamicActionsExecution.class);
    private DynamicAction caseCreationAction = null;

    public void setUp()throws Exception {
        caseCreationAction = new CaseCreationAction();
    }

    public void testDynamicAction() {

        //RequestId
        String requestId = "testRequest";

        //Request Time
        Date requestTime = new Date();

        //Map that contains values passed to the rule/model execution
        Map ruleContextMap = new HashMap();

        //Result from rule execution
        VCryptRulesResultImpl rulesResult = new VCryptRulesResultImpl();
        rulesResult.setResult("Allow");
        rulesResult.setRuntimeType(new Integer(1));

        //Configurable action's parameter values
        Map actionParamValueMap = new HashMap();
        RuntimeActionParamValue caseTypeParamValue = new
RuntimeActionParamValue();
        caseTypeParamValue.setIntValue(CaseConstants.CASE_AGENT_TYPE);
```

```
        RuntimeActionParamValue caseSeverityParamValue = new
RuntimeActionParamValue();
        caseSeverityParamValue.setIntValue(1);

        RuntimeActionParamValue caseDescriptionParamValue = new
RuntimeActionParamValue();
        caseDescriptionParamValue.setStringValue("Testing CaseCreation
Action");

        //ActionContext Map for passing data to/from the dynamic action
execution
        Map actionContextMap = new HashMap();

        //Execute the action
        try {
            caseCreationAction.execute(requestId, requestTime,
ruleContextMap, rulesResult, actionParamValueMap, actionContextMap);
        }catch(Exception e) {
            Assert.fail("Exception occurred while executing dynamic
action");
            logger.error("Exception occurred while executing dynamic
action", e);
        }

        //Write appropriate asserts to check if the configurable action
has executed properly
    }

    public void tearDown() throws Exception {

    }
}
```

OTP Anywhere

Oracle Adaptive Access Manager 11g provides the framework to support the One Time Password authentication method.

One Time Password (OTP) is used to authenticate an individual based on a single-use alphanumeric credential.

The OTP is delivered to the user's configured delivery method. The user then provides the OTP credential as the response to proceed with the operation.

The following are major benefits of using out-of-band OTP:

- If the end user's browser/internet is compromised, the authentication can safely take place in another band of communication separate from the browser
- The user does not require any proprietary hardware or client software of any kind.

This chapter provides an example of how to integrate OTP into the system. It contains the following sections:

- [OTP Integration](#)
- [Implementing Challenge Processors](#)
- [Configuring the Challenge Pad Used for Challenge](#)
- [Configuring User Information Properties](#)

9.1 OTP Integration

To integrate OTP into your system, the following tasks must be performed:

1. Implement a challenge processor.
2. Configure the authentication device
3. Configure the user information properties

9.2 Implementing Challenge Processors

Challenge Processors are a plug-in framework that allows integrators the ability to write custom processor classes that can be triggered when an associated rule action is returned by the challenge checkpoint.

Challenge processors perform the following tasks:

- Generate challenge secret (PIN) to send to the user.
- Validate the user answer

- Control delivery wait page (if needed)
- Check if delivery service is available (if needed)

Processor classes are:

- Customizable challenge method processors
- Customizable rules result (rule action) processors (including legacy processors for existing policies)

To use SMS, you must implement a method for generating the secret PIN and checking the status of the send and the class that is called for a challenge type. You must also configure policies to use SMS.

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, refer to [Chapter 12, "Customizing Oracle Adaptive Access Manager."](#)

9.2.1 Create a Challenge Processor

To implement a challenge processor, you will need to extend the following class:

```
com.bharosa.uio.processor.challenge.AbstractChallengeProcessor
```

Later, you will compile the code by adding `oaam.jar` from `$ORACLE_IDM_HOME\oaam\cli\lib` folder to the build classpath.

Methods to Implement

A few methods to implement a challenge processor is shown as follows:

```
protected boolean generateSecret(UIOSessionData sessionData, boolean isRetry) to  
generate code to send to client
```

```
protected boolean validateAnswer(UIOSessionData sessionData, String answer) to  
validate the user answer
```

```
public String checkDeliveryStatus(UIOSessionData sessionData, boolean userWaiting,  
boolean isRetry) if you want to provide wait until message is sent
```

```
public boolean isServiceAvailable(UIOSessionData sessionData) to check if external  
service is available
```

Example

An example of implementing a challenge processor is shown as follows:

```
package oracle.oaam.challenge.processor.challenge;  
  
import com.bharosa.common.util.*;  
import com.bharosa.uio.util.UIOUtil;  
import com.bharosa.uio.util.UIOSessionData;  
  
import com.bharosa.common.logger.Logger;  
  
import java.io.Serializable;  
  
/**  
 * Email Challenge Processor - provides OTP Code generation, delivery and  
 validation  
 */
```



```

public class EmailChallengeProcessor extends
com.bharosa.uio.processor.challenge.AbstractOTPChallengeProcessor implements
Serializable{

    static Logger logger = Logger.getLogger(EmailChallengeProcessor.class);

    public EmailChallengeProcessor( ) {
    }

    /**
     * Generates OTP Code and stores it in sessionData
     *
     * @param sessionData data object available for the session
     * @param isRetry boolean value if method was called as a result of a failed
answer attempt
     * @return
     */
    protected boolean generateSecret(UIOSessionData sessionData, boolean isRetry) {
        String otpCode = sessionData.getOTPCode();

// If no secret code is present in session, generate one.
        if (StringUtil.isEmpty(otpCode)) {
            if (logger.isDebugEnabled())
                logger.debug("ChallengeEmail generating security code for user: " +
                    sessionData.getCustomerId());
            otpCode = generateCode(sessionData);

            // save the code for later reference - validate / resend
            sessionData.setOTPCode(otpCode);
        }

        if (logger.isDebugEnabled())
            logger.debug("OTP code for user " + sessionData.getCustomerId() + " : " +
                otpCode);

        if (StringUtil.isEmpty(otpCode)) {
            logger.error("Email Challenge pin generation returned null.");
            return false;
        }
        // isRetry flag is turned on if user fails to answer the question
        if (!isRetry) {
            return sendCode(sessionData);
        }

        return true;
    }

    /**
     * Validate user entered answer against value in sessionData
     *
     * @param sessionData validate code and return result.
     * @param answer answer provided by the user
     * @return
     */
    protected boolean validateAnswer(UIOSessionData sessionData, String answer){
        //need to authenticate OTP Code
        String otpCode = sessionData.getOTPCode();

        if (otpCode != null && otpCode.equals(answer)) {
            // Expire OTP Code

```

```
        sessionData.setOTPCode(null);
        return true;
    }

    return false;
}

/**
 * Private methods to send secret code to client
 *
 * @param sessionData
 * @return
 */
private boolean sendCode(UIOSessionData sessionData){
    String otpCode = sessionData.getOTPCode();

    try {
        // UIOUtil.getOTPContactInfo fetches the information registered by the user.
        Refer to ChallengeEmail.requiredInfo in configuration.
        String toAddr = UIOUtil.getOTPContactInfo(sessionData, "email");
        if (StringUtil.isEmpty(toAddr)) {
            logger.error("No user email in profile.");
            return false;
        }

        // Send secret code to customer using your email provider

    } catch (Exception ex) {
        logger.error("ChallengeEmail Error sending code.", ex);
        return false;
    }

    return true;
}

public String checkStatus(UIOSessionData sessionData, boolean userWaiting,
boolean isRetry) {
    String target = ChallengeProcessorIntf.TARGET_WAIT;
    // user already has code, trying again - send to challenge page
    if (isRetry){
        return ChallengeProcessorIntf.TARGET_CHALLENGE;
    }

    boolean sendComplete = false;
    if (userWaiting){
        // if secret code is sent set target to
        target = ChallengeProcessorIntf.TARGET_CHALLENGE;
        // failed to send
        target = ChallengeProcessorIntf.TARGET_ERROR;
        // still processing
        target = ChallengeProcessorIntf.TARGET_WAIT;
    }
    return target;
}
}
```

9.2.2 Configure Custom Challenge Processor as Challenge Type with Required Profile Data

Challenge types are configured by the enum, "challenge.type.enum". The actual enum value is shown as follows:

```
bharosa.uio.<application>. challenge.type.enum.<challenge type>
```

Example for Defining an OTP Type

Configure the bharosa.uio.default.challenge.type.enum property to edit out of the box OTP challenge types or add a new challenge type.

Here is an example for defining an OTP type:

```
bharosa.uio.default.challenge.type.enum.MyChallenge
```

In the example, the "default" is the UIO application name, and "MyChallenge" is challenge Type being newly added

To enable/disable a challenge type, the available flag should be set (not the enable flag, that would remove it from list)

Table 9–1 Challenge type Properties

Property	Description
available	if the challenge type is available for use (service ready and configured). To enable/disable an OTP challenge type, the available flag should be set.
processor	java class for handling challenges of this type.
requiredInfo	comma separated list of inputs from the registration input enum

Attributes of the enum with example values is shown as follows:

```
bharosa.uio.default.challenge.type.enum.MyChallenge = 1 // unique value to
identify Challenge Email in bharosa.uio.default.challenge.type.enum
```

```
bharosa.uio.default.challenge.type.enum.MyChallenge.name = MyChallenge // unique
string to identify Challenge Email in bharosa.uio.default.challenge.type.enum, no
spaces
```

```
bharosa.uio.default.challenge.type.enum.MyChallenge.description = Email Challenge
// descriptive name
```

```
bharosa.uio.default.challenge.type.enum.MyChallenge.processor =
oracle.oaam.challenge.processor.challenge.EmailChallengeProcessor // Processor
used for sending emails instance of
com.bharosa.uio.processor.challenge.ChallengeProcessorIntf
```

```
bharosa.uio.default.challenge.type.enum.MyChallenge.requiredInfo = email // comma
separated field names, User registration flow captures these data fields, check
Contact information Inputs section to define this enum
```

```
bharosa.uio.default.challenge.type.enum.MyChallenge.available = false // to turn
off this service
```

```
bharosa.uio.default.challenge.type.enum.MyChallenge.otp = true // indicates this
challenge is used for OTP, set it to true
```

Email Example

```

bharosa.uio.default.challenge.type.enum.ChallengeEmail = 1
bharosa.uio.default.challenge.type.enum.ChallengeEmail.name = Email Challenge
bharosa.uio.default.challenge.type.enum.ChallengeEmail.description = Email
Challenge
bharosa.uio.default.challenge.type.enum.ChallengeEmail.processor =
com.bharosa.uio.processor.challenge.EmailChallengeProcessor
bharosa.uio.default.challenge.type.enum.ChallengeEmail.requiredInfo = mobile
bharosa.uio.default.challenge.type.enum.ChallengeEmail.available = true
bharosa.uio.default.challenge.type.enum.ChallengeEmail.enabled = true

```

SMS Example

```

bharosa.uio.default.challenge.type.enum.ChallengeSMS = 2
bharosa.uio.default.challenge.type.enum.ChallengeSMS.name = SMS Challenge
bharosa.uio.default.challenge.type.enum.ChallengeSMS.description = SMS Challenge
bharosa.uio.default.challenge.type.enum.ChallengeSMS.processor =
com.bharosa.uio.processor.challenge.SmsChallengeProcessor
bharosa.uio.default.challenge.type.enum.ChallengeSMS.requiredInfo = mobile
bharosa.uio.default.challenge.type.enum.ChallengeSMS.available = true
bharosa.uio.default.challenge.type.enum.ChallengeSMS.enabled = true

```

9.3 Configuring the Challenge Pad Used for Challenge

By default, challenge devices are configured through rules. The rules are under the AuthentiPad checkpoint and determine the type of device to use based on the purpose of the device (ChallengeEmail, ChallengeSMS, ChallengeQuestion, and so on).

To create/update policies to use the challenge type:

1. Add a new rule action, MyChallenge, with the enum, rule.action.enum.
2. Create policy to return newly created action, MyChallenge, to use the challenge method.

Alternatively, if you want to configure challenge devices using properties, you can bypass the AuthentiPad checkpoint by setting

```
bharosa.uio.default.use.authentipad.checkpoint to false.
```

Devices to use for the challenge type can be added.

```
bharosa.uio.<application>.<challengeType>.authenticator.device=<value>
```

The examples shown use the challenge type key, ChallengeEmail and ChallengeSMS to construct the property name.

```

bharosa.uio.default.ChallengeSMS.authenticator.device=DevicePinPad
bharosa.uio.default.ChallengeEmail.authenticator.device=DevicePinPad

```

Available challenge device values are DeviceKeyPadFull, DeviceKeyPadAlpha, DeviceTextPad, DeviceQuestionPad, DevicePinPad, and DeviceHTMLControl.

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, refer to [Chapter 12, "Customizing Oracle Adaptive Access Manager."](#)

9.4 Configuring User Information Properties

Instructions to configure user information properties are in the following sections:

- [Configuration Settings for Information Registration and Preferences and PIN Generation](#)
- [Set Contact Information Inputs](#)
- [SMS Configuration to Receive OTP via SMS](#)

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, refer to [Chapter 12, "Customizing Oracle Adaptive Access Manager."](#)

9.4.1 Configuration Settings for Information Registration and Preferences and PIN Generation

Default configurations for settings for registration of information, preference, and PIN generation are listed as follows:

Contact information registration

```
bharosa.uio.default.register.userinfo.enabled=false
```

Contact information preferences

```
bharosa.uio.default.userpreferences.userinfo.enabled=false
```

OTP PIN Generation

```
# Length of the Pin
bharosa.uio.otp.generate.code.length = 5
# Characters to use when generating the Pin
bharosa.uio.otp.generate.code.characters = 1234567890
```

The `AbstractOTPChallengeProcessor` class has a default pin generation method, `generateCode`, that you can override to provide your pin generation logic.

9.4.2 Set Contact Information Inputs

Contact information inputs are defined in `userinfo.inputs.enum`. The enum element is:

```
bharosa.uio.<application>.userinfo.inputs.enum.<inputname>
```

Example

```
bharosa.uio.default.userinfo.inputs.enum.email=1
bharosa.uio.default.userinfo.inputs.enum.email.name=Email Address
bharosa.uio.default.userinfo.inputs.enum.email.description=Email Address
bharosa.uio.default.userinfo.inputs.enum.email.inputname=email
bharosa.uio.default.userinfo.inputs.enum.email.inputtype=text
bharosa.uio.default.userinfo.inputs.enum.email.maxlength=40
bharosa.uio.default.userinfo.inputs.enum.email.required=true
bharosa.uio.default.userinfo.inputs.enum.email.order=2
bharosa.uio.default.userinfo.inputs.enum.email.enabled=true
bharosa.uio.default.userinfo.inputs.enum.email.regex=.\+@[a-zA-Z_
]+?\.\.[a-zA-Z]{2,3}
bharosa.uio.default.userinfo.inputs.enum.email.errorCode=otp.invalid.email
bharosa.uio.default.userinfo.inputs.enum.email.managerClass=com.bharosa.uio.manage
r.user.DefaultContactInfoManager
```

9.4.3 SMS Configuration to Receive OTP via SMS

For OTP to be receive via SMS, you must:

- [Enable Registration and Preference Setting](#)
- [Set Input Information](#)

9.4.3.1 Enable Registration and Preference Setting

To enable OTP profile registration and preference setting, set the following properties to true:

- `bharosa.uio.default.register.userinfo.enabled`
Setting the property to true enables the profile registration pages if the OTP channel is enabled and requires registration.
- `bharosa.uio.default.userpreferences.userinfo.enabled`
Setting the property to true enables the user to set preferences if the OTP channel is enabled and allows preference setting.

9.4.3.2 Set Input Information

If user information registration or user preferences is true, configure input information via enum:

```
bharosa.uio.default.userinfo.inputs.enum
```

Table 9–2 *OTP Properties for Contact Input*

Property	Description
<code>inputname</code>	Name used for the input field in the HTML form
<code>inputtype</code>	Set for text or password input
<code>maxlength</code>	Maximum length of user input
<code>required</code>	Set if the field is required on the registration page
<code>order</code>	The order displayed in the user interface
<code>regex</code>	Regular expression used to validate user input for this field
<code>errorCode</code>	Error code used to look up validation error message (bharosa.uio.<application ID>.error.<errorCode>)
<code>managerClass</code>	java class that implements <code>com.bharosa.uio.manager.user.UserDataManagerIntf</code> (if data is to be stored in Oracle Adaptive Access Manager database this property should be set to <code>com.bharosa.uio.manager.user.DefaultContactInfoManager</code>)

The following is an example of an enum defining cell phone registration on the OTP registration page of a virtual authentication device:

```
bharosa.uio.default.userinfo.inputs.enum.mobile=0
bharosa.uio.default.userinfo.inputs.enum.mobile.name=Mobile Phone
bharosa.uio.default.userinfo.inputs.enum.mobile.description=Mobile Phone
bharosa.uio.default.userinfo.inputs.enum.mobile.inputname=cellnumber
bharosa.uio.default.userinfo.inputs.enum.mobile.inputtype=text
bharosa.uio.default.userinfo.inputs.enum.mobile.maxlength=15
bharosa.uio.default.userinfo.inputs.enum.mobile.required=true
bharosa.uio.default.userinfo.inputs.enum.mobile.order=1
```

```
bharosa.uio.default.userinfo.inputs.enum.mobile.enabled=true
```

Flash Fingerprinting

This chapter focuses on the specifics of Flash Fingerprinting within an Oracle Adaptive Access Manager native integration.

All code examples included in the chapter are outlines of calls needed to perform the tasks. They should not be considered complete implementations.

Note: This chapter assumes that the reader is familiar with Oracle Adaptive Access Manager native integrations and APIs.

10.1 Device Fingerprinting

Oracle Adaptive Access Manager captures information about the devices that a user utilizes when accessing protected applications. This information consists of many different datapoints gathered through a variety of means. The data collected is encoded into a unique fingerprint for the device.

When a device is used for an access request, Oracle Adaptive Access Manager interrogates the device for the fingerprint and uses it along with many other types of data to determine the risk associated with the specific access request. Some of the technology used to gather fingerprint data include HTTP header, secure cookie, shared flash object and behavior profiling.

10.2 Definitions of Variables and Parameters

[Table 10–1](#) lists the parameter and response variable in the interaction between the flash movie and the application.

Table 10–1 *Flash movie Parameters and Response Variables*

Parameter/Response Variable	Usage
v	Used as an HTTP request parameter sent from the flash movie to the application. It contains the generated "cookie" string that is used a single time by the user. This value is also returned in the HTTP response to the flash movie as "&v=<new value>".
client	Used as an HTTP request parameter sent from the flash movie to the application. This indicates the type of client performing the fingerprinting (in this case, flash). The expected value from the flash movie is "vfc".
fp	Used as an HTTP request parameter sent from the flash movie to the application. It contains information about the client computer accessible to the flash player.

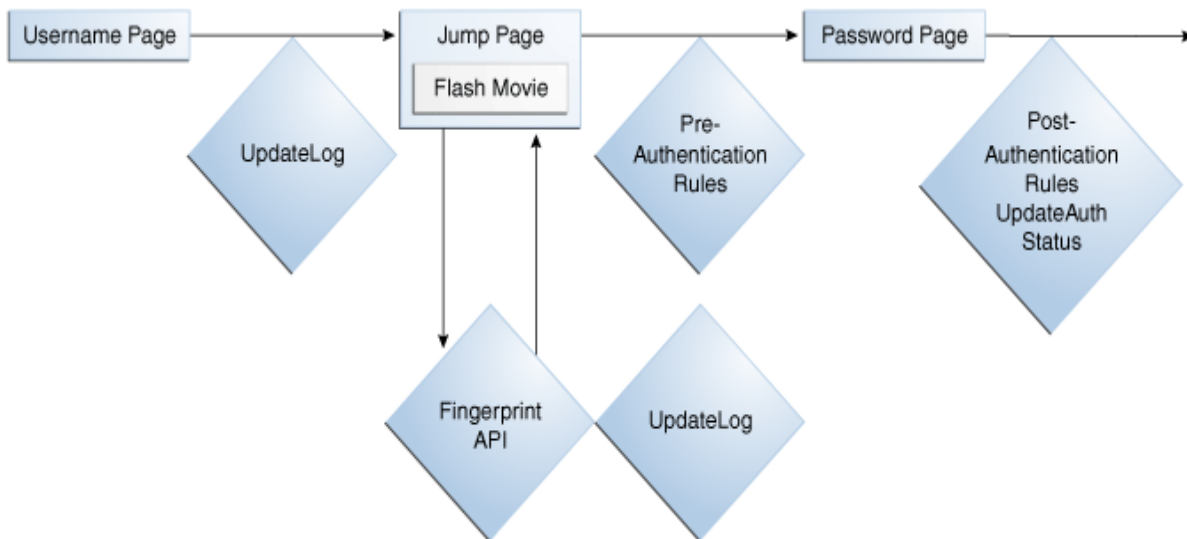
10.3 Option 1

Option 1 is the traditional implementation using a "Jump Page" to include the flash movie that is used for fingerprinting. In Option 1, the flash movie sends the user's current flash cookie value to the server and the server responds with a new value in a single transaction.

10.3.1 Option 1 Flow

Figure 10–1 shows the flow of Option 1.

Figure 10–1 Option 1



1. The user is presented with the username page
2. The user submits the username
 - a. The application loads the user
 - b. The application calls `VCryptTracker.updateLog` with the User and HTTP Cookie information
3. The user is taken to the jump page containing the embedded flash movie
 - a. The flash movie makes an HTTP request triggering flash fingerprint handling
 - i. The server retrieves the HTTP request parameter "v" and stores it in session
 - ii. The server retrieves the HTTP request parameter "client"
 - iii. The server retrieves the HTTP request parameter "fp"
 - iv. Parse fp with `VCryptServletUtil.getFlashFingerprint(client, fp)`
 - v. Calls `VCryptTracker.updateLog` with the User, HTTP Cookie, and Flash information
 - vi. The new flash cookie returned in `CookieSet` from `updateLog` is returned to the flash movie in the HTTP response ("`&v=" + cookieSet.getFlashCookie()`")
4. The user is taken to password page after jump page wait period
 - a. Run the Pre-Authentication Rules
5. The user submits the password

- a. The application verifies the password
- b. Run Post-Authentication Rules
- c. Calls `VCryptTracker.updateAuthStatus` with authentication result

10.3.2 Option 1 Code Example

This section provides a code example for Option 1.

```
public String flashFingerPrint(HttpServletRequest request) {
    HttpSession session = request.getSession(true);
    try {
        String digitalCookie = request.getParameter("v");
        String fpStr = request.getParameter("fp");
        String client = request.getParameter("client");
        String flashFingerprint =
VCryptServletUtil.getFlashFingerPrint(client, fpStr);
        session.setAttribute("v", digitalCookie);
        session.setAttribute("fp", flashFingerprint);

        VCryptAuthUser clientUser = (VCryptAuthUser)
session.getAttribute("clientUser");

        if (clientUser == null) {
            // User not found in session
            return "";
        }

        String loginId = clientUser.getLoginId();
        String customerId = clientUser.getCustomerId();
        String groupId = clientUser.getCustomerGroupId();
        int clientType = UserDefEnum.getElementValue(IBharosaConstants.ENUM_
CLIENT_TYPE_ID, FLASH_CLIENT_ENUM);

        cookieSet = updateLog(request, loginId, customerId, groupId,
clientType, authResult);

        session.setAttribute("cookieSet");
return cookieSet.getFlashCookie();
    } catch (Exception e) {
        // Handle fingerprinting error
    }
    return "";
} // flashFingerPrint
```

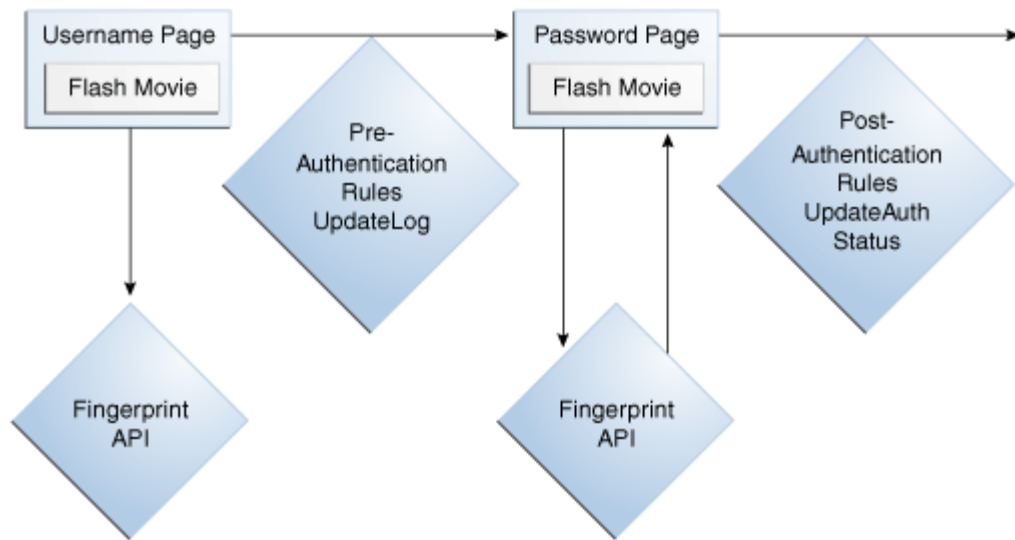
10.4 Option 2

Option 2 is a newer, more streamlined user experience that eliminates the "Jump Page" from the user experience. To do this, the flash movie is included in both the username page and the password page.

10.4.1 Option 2 Flow

Figure 10–2 shows the flow of Option 2.

Figure 10–2 Option 2



1. The user is presented with the username page with the embedded flash movie
 - a. The flash movie makes an HTTP request triggering the flash fingerprint handling
 - i. The server retrieves the HTTP request parameter "v" and stores it in session
 - ii. The server retrieves HTTP request parameter "client"
 - iii. The server retrieves HTTP request parameter "fp"
 - iv. Parse fp with `VCryptServletUtil.getFlashFingerprint(client, fp)` and store result in user session.
 - v. The value of "v" received is returned to the flash movie in the HTTP response ("`&v=" + cookieSet.getFlashCookie()`")
2. The user submits the username
 - a. The application loads the user
 - b. Run Pre-Authentication Rules
 - c. Calls `VCryptTracker.updateLog` with the User, HTTP Cookie and Flash value
3. The user is taken to the password page with the embedded flash movie
 - a. The flash movie makes an HTTP request triggering the flash fingerprint handling
 - i. The server already has the value from the previous flash request
 - ii. The new value generated by `UpdateLog` call is returned to flash movie
4. The user submits the password
 - a. The application verifies the password
 - b. Run the Post-Authentication Rules
 - c. Calls `VCryptTracker.updateAuthStatus` with the authentication result

10.4.2 Option 2 Code Example

This section provides a code example for Option 2.

```

public String flashFingerPrint(HttpServletRequest request) {
    HttpSession session = request.getSession(true);
    try {
        CookieSet cookieSet = (CookieSet)session.getAttribute("cookieSet");
        if (cookieSet == null) {
            String digitalCookie = request.getParameter("v");
            String fpStr = request.getParameter("fp");
            String client = request.getParameter("client");
            String flashFingerprint =
VCryptServletUtil.getFlashFingerPrint(client, fpStr);
            session.setAttribute("v", digitalCookie);
            session.setAttribute("fp", flashFingerprint);
        } else {
            // finger printing already happened, using previously
generated cookie set
        }
        return cookieSet.getFlashCookie();
    } catch (Exception e) {
        // Handle fingerprinting error
    }
    return "";
} // flashFingerPrint

```

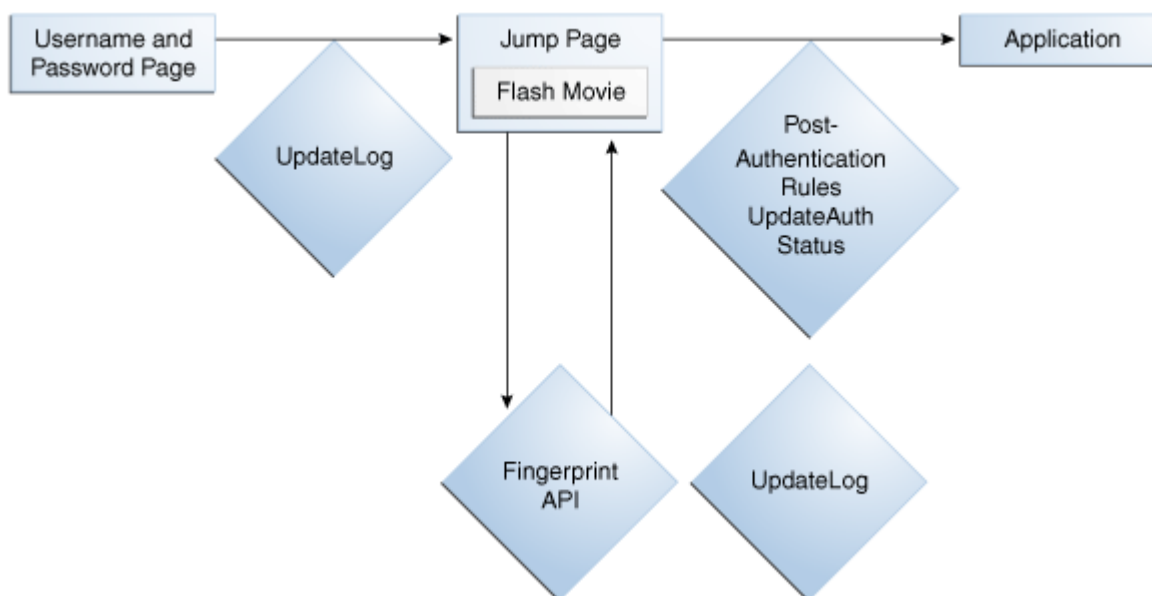
10.5 Option 3

Option 3 is an implementation using a single page for username and password (not using virtual authentication devices), and uses a "Jump Page" to include the flash movie used for fingerprinting. In this case, the flash movie will send the server the user's current flash cookie value and the server will respond with a new value in a single transaction.

10.5.1 Option 3 Flow

Figure 10-3 shows the flow of Option 3.

Figure 10-3 Option 3 Flow



1. The user is presented with a single username and password page
2. The user submits the username and password
 - a. The application loads user
 - b. The application verifies password
 - c. Calls `VCryptTracker.updateLog` with User, authentication result and HTTP Cookie information
3. The user is taken to the jump page containing the embedded flash movie
 - a. The flash movie makes an HTTP request triggering the flash fingerprint handling
 - i. The server retrieves the HTTP request parameter "v" and stores it in session
 - ii. The server retrieves the HTTP request parameter "client"
 - iii. The server retrieves HTTP request parameter "fp"
 - iv. Parse fp with `VCryptServletUtil.getFlashFingerprint(client, fp)`.
 - v. Calls `VCryptTracker.updateLog` with User, HTTP Cookie, and Flash information
 - vi. The new flash cookie returned in `CookieSet` from `updateLog` is returned to the flash movie in the HTTP response ("`&v="` + `cookieSet.getFlashCookie()`)
4. The user continues on to the application after the jump page wait period
 - a. Run Post-Authentication Rules
 - b. Calls `VCryptTracker.updateAuthStatus` with authentication result

10.5.2 Option 3 Code Example

This section provides a code example for Option 3.

```
public String flashFingerPrint(HttpServletRequest request) {
    HttpSession session = request.getSession(true);
    try {
        String digitalCookie = request.getParameter("v");
        String fpStr = request.getParameter("fp");
        String client = request.getParameter("client");
        String flashFingerprint =
VCryptServletUtil.getFlashFingerprint(client, fpStr);
        session.setAttribute("v", digitalCookie);
        session.setAttribute("fp", flashFingerprint);

        VCryptAuthUser clientUser = (VCryptAuthUser)
session.getAttribute("clientUser");

        if (clientUser == null) {
            // User not found in session
            return "";
        }

        String loginId = clientUser.getLoginId();
        String customerId = clientUser.getCustomerId();
        String groupId = clientUser.getCustomerGroupId();
        int clientType = UserDefEnum.getElementValue(
IBharosaConstants.ENUM_
CLIENT_TYPE_ID, FLASH_CLIENT_ENUM);
    }
}
```

```

        cookieSet = updateLog(request, loginId, customerId, groupId,
clientType, authResult);

        session.setAttribute("cookieSet");
        return cookieSet.getFlashCookie();
    } catch (Exception e) {
// Handle fingerprinting error
    }
    return "";
} // flashFingerPrint

```

10.6 Common Update

The implementations would use a method similar to the following for making updateLog calls:

```

protected CookieSet updateLog(HttpServletRequest request,
                                String loginId, String userId, String
groupId,
                                int clientType, int authStatus) throws
BharosaProxyException {
    HttpSession session = request.getSession(true);

    String requestId = (String) session.getAttribute("requestId");
    String remoteIPAddr = request.getRemoteAddress();
    String remoteHost = request.getRemoteHost();

    String secureCookie =
VCryptServletTrackerUtil.getSecureCookie(request);
    String secureClientVersion = "1.0";

    Object[] fingerPrintInfo =
VCryptServletUtil.getBrowserFingerPrint(request);
    int fingerPrintType = fingerPrintInfo == null ? 0 :
((Integer)fingerPrintInfo[0]).intValue();
    String fingerPrint = fingerPrintInfo == null ? "" :
(String)fingerPrintInfo[1];

    int fingerPrintType2 = VCryptServletUtil.flashFPTType.intValue();
    String fingerPrint2 = (String) session.getAttribute("fp");
    String digitalCookie = (String) session.getAttribute("v");

    CookieSet cookieSet = (CookieSet)
session.getAttribute("cookieSet");

    if (secureCookie == null && cookieSet != null) {
        secureCookie = cookieSet.getSecureCookie();
    }

    if (digitalCookie == null && cookieSet != null) {
        digitalCookie = cookieSet.getFlashCookie();
    }

    boolean isSecure = false;

    VCryptTracker vTracker =
VCryptTrackerUtil.getVCryptTrackerInstance();
    cookieSet = vTracker.updateLog(requestId, remoteIPAddr, remoteHost,
secureCookie,
        digitalCookie, groupId, userId, loginId,

```

```
        isSecure, authStatus, clientType,  
        secureClientVersion, fingerprintType,  
        fingerprint, fingerprintType2,  
        fingerprint2);  
  
        return cookieSet;  
    }  
}
```

Device Registration

Device registration allows a user to flag the computer, PDA, mobile phone, or other devices he is logging in with as a safe device.

The device is added to the user's profile as a registered device.

Enabling Device Registration in Native Integration

In native integration, to enable device registration:

1. Set `bharosa.tracker.send.deviceId` to true, so the device ID can be captured.
2. Call these APIs directly:
 - `handleTrackerRequest`
 - `updateLog`
 - `markDeviceSafe`
 - `IsDeviceMarkedSafe`
 - `clearSafeDeviceList`
 - `processRules`

Enabling Device Registration Out-of-the-Box

In Oracle Adaptive Access Manager out-of-the-box, to enable device registration for all applications:

1. Add the following properties to `bharosa_server.properties`:

```
# Adds device registration to the challenge question registration page
bharosa.uio.default.register.questions.registerdevice.enabled=true

# Adds device registration to the Contact Information registration page
bharosa.uio.default.register.userinfo.registerdevice.enabled=true

# Enables device registration
bharosa.uio.default.registerdevice.enabled=true

# Enables user to be able to unregister current device in user preferences
bharosa.uio.default.userpreferences.unregister.this.enabled=true

# Enables user to be able to unregister all devices in user preferences
bharosa.uio.default.userpreferences.unregister.all.enabled=true
```

To enable the features on an application-specific bases, "default" can be replaced with the appropriate `appId` in each of the prior property names.

-
2. Follow the instructions in [Chapter 12, "Customizing Oracle Adaptive Access Manager"](#) to add the customizations to Oracle Adaptive Access Manager.

Create Policies to Use Device Information

Once the feature is enabled, information about the device is collected for that user. If you want to make use of the information you are collecting, you must create policies and configure them properly. For example, you can create a policy with rules to challenge a user that is not logging in from one of the registered devices.

Resetting Registration

A customer reset action to unregister all devices for a user is available in CSR type cases. The "Unregister Devices" action will delete all registered devices from the user's profile.

Part IV

Customizing Oracle Adaptive Access Manager

Part IV contains the following chapter:

- [Chapter 12, "Customizing Oracle Adaptive Access Manager"](#)

Customizing Oracle Adaptive Access Manager

The chapter provides information on how to customize Oracle Adaptive Access Manager by using the Oracle Adaptive Access Manager Extensions Shared Library.

It contains the following sections:

- [Overview](#)
- [Add Customizations Using Oracle Adaptive Access Manager Extensions Shared Library](#)

12.1 Overview

Shared libraries are collections of programming and data that can be used by multiple applications. They can permit applications to use memory efficiently by sharing common programming and resources.

You can customize Oracle Adaptive Access Manager by adding custom jars and files to the Oracle Adaptive Access Manager Extensions Shared Library.

This shared library, a war file named `oracle.oaam.extensions`, is deployed in both OAAM Server and OAAM Admin Server.

By default `oracle.oaam.extensions.war` is empty and does not contain any files except the `MANIFEST.MF` that has the definition of the shared library.

12.2 Add Customizations Using Oracle Adaptive Access Manager Extensions Shared Library

Follow these steps to add customizations to Oracle Adaptive Access Manager:

1. Create a work folder called `oaam-extensions`.
The folder can be created anywhere as long as it is outside the installation folder.
2. In the `oaam-extensions` folder, create the following subfolders:
 - `META-INF`
 - `WEB-INF`
 - `WEB-INF\lib`
 - `WEB-INF\classes`
3. In the `META-INF` folder, create a file named `MANIFEST.MF` and add the following lines:

Extension-Name: oracle.oaam.extensions
Specification-Version:99.9.9.9.9
Implementation-Version:99.9.9.9.9

4. Compile custom java classes that extend or implement Oracle Adaptive Access Manager classes, adding the jars from `$ORACLE_IDM_HOME\oaam\cli\lib` folder to the build class path.
5. Add custom jars and files as described:
 - a. Add the custom jars to the `oaam-extensions\WEB-INF\lib` folder.
 - b. Add custom properties to a file named `bharosa_server.properties` and save it in the `oaam-extensions\WEB-INF\classes` folder.
 - c. Add custom JSPs to `oaam-extensions` folder.
6. Rejar `oracle.oaam.extensions.war` using the command from the parent folder of `oaam-extensions`:

```
jar -cvfm oracle.oaam.extensions.war oaam-extensions\META-INF\MANIFEST.MF -C oaam-extensions/ .
```
7. Start the WebLogic Server where Oracle Adaptive Access Manager is deployed and log into the WebLogic Administration Console.
8. Deploy the `oracle.oaam.extensions.war` file created in Step 6 as a Shared Library with `oaam_server` and `oaam_admin` as target applications.
9. Test the custom functionality and make sure files added to `oracle.oaam.extensions.war` are used by Oracle Adaptive Access Manager applications.

Part V

Authentication and Password Management Integration

Part V contains a chapter on Oracle Adaptive Access Manager, Oracle Access Manager, and Oracle Identity Manager integration.

Access and Password Management Integration

This chapter provides an overview of the benefits and a list of scenarios of Oracle Access Manager with Oracle Identity Manager and Oracle Adaptive Access Manager.

Detailed conceptual and procedural information is provided in the *Oracle Fusion Middleware Integration Guide for Oracle Access Manager*.

13.1 Benefits and Features of the Integration

Integrating Oracle Access Manager, Oracle Adaptive Access Manager, and Oracle Identity Manager provides these features:

- Password entry protection through personalized virtual authentication devices
- KBA challenge questions for secondary login authentication based on risk
- OTP challenge for secondary login authentication based on risk
- Registration flows to support password protection and KBA and OTP challenge functionality
- User preferences flows to support password protection and KBA and OTP challenge functionality
- Password management flows

Oracle Adaptive Access Manager

Oracle Adaptive Access Manager is responsible for:

- Running fraud rules before and after authentication
- Navigating the user through Oracle Adaptive Access Manager flows based on the outcome of fraud rules

Oracle Identity Manager

Oracle Identity Manager is responsible for:

- Provisioning users (add/modify, delete users)
- Managing passwords (reset/change password)

Oracle Access Manager

Oracle Access Manager is responsible for:

- Authenticating and authorizing users

- Providing statuses such as Reset Password, Password Expired, User Locked, and others

13.2 Secure Password Collection and Management Scenarios

In this integration, Oracle Access Manager redirects users to Oracle Adaptive Access Manager when a trigger condition for password management is in effect. The "trigger condition" is the authentication scheme used in Oracle Access Manager.

Oracle Adaptive Access Manager interacts with the user based on lifecycle policies retrieved from Oracle Access Manager, and when the condition is resolved, notifies Oracle Access Manager so that the user is redirected to the protected resource. In this integration, Oracle Identity Manager serves to provide password policy enforcement.

Challenge Registration Flow

The Challenge Registration flow allows the user to register challenge questions and answers.

The user is successfully authenticated but is required to register challenge questions. He cannot skip the registration. The user is not authorized to access protected resources until the challenges questions have been registered.

Note: When adding Oracle Adaptive Access Manager to existing Oracle Identity Manager deployments, you will need to forego all the existing questions and answers that are registered in Oracle Identity Manager. Instead, users are asked to register the challenge questions again in Oracle Adaptive Access Manager on the next login.

Forgot Password Flow

The Forgot Password flow allows the user to reset the password after successfully answering all challenge questions.

A "Forgot Your Password" link is made available from the Oracle Adaptive Access Manager password page for the user.

Reset Password Flow

The Reset Password flow allows the user to reset the password.

The user is successfully authenticated. The "Change your password" link is available to the user at the Oracle Adaptive Access Manager password page.

Challenge Reset Flow

The Challenge Reset flow allows the user to reset challenge registration.

The user is successfully authenticated. The "Reset your challenge questions" link is available in the Oracle Adaptive Access Manager password page.

Part VI

Lifecycle Management

Part VI contains a chapter on Oracle Adaptive Access Manager lifecycle management use cases.

Handling Lifecycle Management Changes

Because of integrated deployment of Oracle Adaptive Access Manager with other applications, Oracle Virtual Directory, Oracle Identity Manager, Oracle Access Manager, Oracle Internet Directory, and configuration changes in those applications, various configuration changes might be required in Oracle Adaptive Access Manager. Instructions for handling such types of configuration changes are described in this chapter:

- [Oracle Virtual Directory \(OVD\) Host, Port, and SSL Enablement Changes](#)
- [Oracle Identity Manager \(OIM\) URL Changes](#)
- [Oracle Access Manager \(OAM\) Host and Port Changes](#)
- [Oracle Internet Directory \(OID\) Host and Port Changes and SSL Enablement](#)
- [Database Host and Port Changes](#)

References are also provided for moving Oracle Adaptive Access Manager from a test environment to a production environment:

- [Moving Oracle Adaptive Access Manager to a New Production Environment](#)
- [Moving Oracle Adaptive Access Manager to an Existing Production Environment](#)

14.1 Oracle Virtual Directory (OVD) Host, Port, and SSL Enablement Changes

To change the Oracle Virtual Directory host, port, and SSL enablement:

1. Start the Oracle Adaptive Access Manager server-related managed server.
2. Go to OAAM Admin at `http://<OAAM Managed Server Host>:<OAAM Admin Managed Server Port>/oaam_admin`.
3. Log in as a user with access to the Properties Editor.
4. Open the Oracle Adaptive Access Manager Property Editor to modify parameters to:
 - Change the password authentication provider to LDAP
 - Rewire existing Oracle Adaptive Access Manager for Oracle Virtual Directory hostname
 - Rewire existing Oracle Adaptive Access Manager for Oracle Virtual Directory port changes

- Rewire existing Oracle Adaptive Access Manager for SSL Enablement of Oracle Virtual Directory (Change Plain Text Communication to SSL for wiring between Oracle Adaptive Access Manager and Oracle Virtual Directory)

Table 14–1 Configuring Oracle Directory Manager Property Values

Property Name	Property Values
bharosa.uio.default.password.auth.provider.class name	com.bharosa.vcrypt.services.LDAPOAAMAuthProvider
oaam.uio.ldap.host	<OVD host> For example, host.oracle.com
oaam.uio.ldap.port	<OVD port>
oaam.uio.ldap.userdn.template	<User Search DN> For example, uid= {USER_ID}, cn=user,dc=us,dc=oracle,dc=com.
oaam.uio.ldap.isSSL	false

For information on setting properties in Oracle Adaptive Access Manager, see "Using the Property Editor" in *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

5. Restart the Oracle Adaptive Access Manager server-related managed server.

14.2 Oracle Identity Manager (OIM) URL Changes

Follow these steps to rewire an existing deployment of Oracle Adaptive Access Manager with Oracle Identity Manager:

1. Start the Oracle Adaptive Access Manager server-related managed server.
2. Go to OAAM Admin at `http://<OAAM Managed Server Host>:<OAAM Admin Managed Server Port>/oaam_admin`.
3. Log in as a user with access to the Properties Editor.
4. Open the Oracle Adaptive Access Manager Property Editor to modify parameters to:
 - Rewire existing Oracle Adaptive Access Manager for password flow
 - Rewire existing Oracle Adaptive Access Manager for other redirection

Table 14–2 Configuring Oracle Identity Manager Property Values

Property Name	Property Values
oaam.oid.url	t3://<OIM Managed Server>:<OIM Managed Port> For example, t3://host.oracle.com:14000
bharosa.uio.default.signon.links.enum.selfregistration.url	http://<OIM Managed Server>:<OIM Managed Port>/oim/faces/pages/USelf.jspx?E_TYPE=USELF&OP_TYPE=SELF_REGISTRATION&backUrl=<OAAM Login URL for OIM> where <OAAM Login URL for OIM> is http://<OHS host>:<OHS port>/oim/faces/pages/Self.jspx or (in case of IDMDOMAINAgent) is http://<OIM host>:<OIMport>/oim/faces/pages/Self.jspx OHS setup was performed during the integration between Oracle Access Manager and Oracle Identity Manager.
bharosa.uio.default.signon.links.enum.trackregistration.url	http://<OIM Managed Server>:<OIM Managed Port>/oim/faces/pages/USelf.jspx?E_TYPE=USELF&OP_TYPE=UNAUTH_TRACK_REQUEST&backUrl=<OAAM Login URL for OIM> where <OAAM Login URL for OIM> is http://<OHS host>:<OHS port>/oim/faces/pages/Self.jspx or (in case of IDMDOMAINAgent) is http://<OIM host>:<OIMport>/oim/faces/pages/Self.jspx. OHS setup was performed during the integration between Oracle Access Manager and Oracle Identity Manager.

For information on setting properties in Oracle Adaptive Access Manager, see "Using the Property Editor" in *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

- Restart the Oracle Adaptive Access Manager server-related managed server.

14.3 Oracle Access Manager (OAM) Host and Port Changes

For information on rewiring Oracle Access Manager for Oracle Adaptive Access Manager hostname and port changes, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager*.

14.4 Oracle Internet Directory (OID) Host and Port Changes and SSL Enablement

Follow these steps to change the Oracle Internet Directory Host, Port and SSL enablement in an existing deployment of Oracle Adaptive Access Manager:

- Start the Oracle Adaptive Access Manager server-related managed server.
- Go to OAAM Admin at `http://<OAAM Managed Server Host>:<OAAM Admin Managed Server Port>/oaam_admin`.
- Log in as a user with access to the Properties Editor.
- Open the Oracle Adaptive Access Manager Property Editor to modify parameters to:
 - Change the password authentication provider to LDAP

- Rewire existing Oracle Adaptive Access Manager for Oracle Internet Directory hostname
- Rewire existing Oracle Adaptive Access Manager for Oracle Internet Directory port changes
- Rewire existing Oracle Adaptive Access Manager for SSL Enablement of Oracle Internet Directory (Change Plain Text Communication to SSL for wiring between Oracle Adaptive Access Manager and Oracle Internet Directory)

Table 14–3 Configuring Oracle Directory Manager Property Values

Property Name	Property Values
bharosa.uio.default.password.auth.provider.class name	com.bharosa.vcrypt.services.LDAPOAAMAuthProvider
oaam.uio.ldap.host	<OID host> For example, host.oracle.com
oaam.uio.ldap.port	<OID port>
oaam.uio.ldap.userdn.template	<User Search DN> For example, uid= {USER_ID}, cn=user,dc=us,dc=oracle,dc=com.
oaam.uio.ldap.isSSL	false

For information on setting properties in Oracle Adaptive Access Manager, see "Using the Property Editor" in *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

5. Restart the Oracle Adaptive Access Manager server-related managed server.

14.5 Database Host and Port Changes

After installing Oracle Adaptive Access Manager, if there are any changes in the database host or port number, follow these instructions:

1. Go to the `ORACLE_HOME` of the database.
2. Change the port number in `ORACLE_HOME /network/admin/listener.ora`.
3. Stop and then restart the Oracle listener.
4. Change the database pointer in the data sources screen in the Weblogic Administration Console

To changes the data source:

1. In the WebLogic Administrative Console, navigate to **Services**, select **JDBC**, select **Data Sources**, and then **oaamDS**.
2. Click **oaamDS** and edit it for hostname/port or username/password.

14.6 Moving Oracle Adaptive Access Manager to a New Production Environment

For information on moving Oracle Adaptive Access Manager to a new production environment, see "Moving Identity Management to a New Production Environment" in *Oracle Fusion Middleware Administrator's Guide*.

14.7 Moving Oracle Adaptive Access Manager to an Existing Production Environment

For information on moving Oracle Adaptive Access Manager to an existing production environment, see "Moving Identity Management to an Existing Production Environment" in *Oracle Fusion Middleware Administrator's Guide*.

Part VII

Troubleshooting

Part VII contains the following chapter:

- [Chapter 15, "FAQ/Troubleshooting"](#)

FAQ/Troubleshooting

This chapter provides troubleshooting tips and answers to frequently asked questions.

It contains the following sections:

- [Universal Installation Option Proxy](#)
- [Virtual Authentication Devices](#)
- [Configurable Actions](#)
- [One-Time Password](#)
- [Man-in-the-Middle/Man-in-the-Browser](#)

15.1 Universal Installation Option Proxy

Oracle Adaptive Access Manager Proxy for Microsoft ISA

To troubleshoot Proxy Web publishing issues:

- Ensure that the .NET2.0 framework is installed and enabled to successfully register the Bharosa Proxy DLL.
- Ensure the database access credentials are correct when the firewall logging properties in Microsoft ISA use SQL Database as the Log Storage Format.
- IP Exceptions are defined for Trusted IPs (like Router IP) when Flood Mitigation settings are enabled to mitigate flood attacks and worm propagation.
- Ensure that the default inbound and outbound rules allow HTTP/HTTPS traffic to be forwarded to/from OAAM server.
- Check the order (precedence) of the rules to ensure that the default rule, "deny," is not at a higher order; otherwise, it blocks all rules. If the rule is last in precedence, all rules are executed.
- In OAAM server rule you must ensure
 - The external IP/name is mapped to the internal IP/name
 - The external port is mapped to the internal port where OAAM server is listening
 - The /OAAM server path is published

To troubleshoot problems experienced while configuring the Oracle Adaptive Access Manager Proxy, enable tracing to file and set the trace level to 0x8008f. This will print detailed interceptor evaluation and execution information to the log file.

Oracle Adaptive Access Manager Proxy for Apache

Tips to troubleshoot problems with the Oracle Adaptive Access Manager Proxy for Apache are listed in this section.

- On launching httpd, an error for loading mod_uio.so occurs. Ensure that mod_uio.so and all the libraries are placed in the proper directories. On Linux, use the 'ldd' command to confirm that mod_uio.so can load all the dynamic libraries that it depends upon. On Windows, use Dependency Walker to find out any missing DLLs and in some cases, you may have to install the "Microsoft Visual C++ 2005 Redistributable Package" from the Microsoft Web site, if your server does not have these libraries pre-installed.
- If nothing is working- no logs and so on, ensure that the user of httpd has permissions to read the uio directory. Typically httpd is run as a daemon user. Ensure the daemon user has write permissions for the logs directory.
- In case of a parsing error in UIO_Settings.xml or any configuration XML, an error log will be created in httpd's logs directory with the name UIO_Settings.xml.log.
- For errors, look in uio.log. Use log level of error for production use; info for more details; debug for debugging issues and trace for verbose logs.
- Ensure that the config XML and settings XML are conforming to the RNG schema. You can use the UIO_Settings.rng and UIO_Config.rng in any XML editor to edit the UIO_Settings.xml and application configuration XML files.
- You can change the Apache httpd log level to debug for testing, or keep it at info to reduce log file size. The Apache httpd log is separate from Oracle Adaptive Access Manager Proxy for Apache log.
- When migrating ISA config XML to be used with the Apache Universal Installation Option Proxy, you need to do the following:

1. Change the header of the XML file to use

```
<?xml version="1.0" encoding="utf-8"?>
<BharosaProxyConfig xmlns="http://bharosa.com/">
```

2. Run your config XML file through libxml2's xmllint utility.

For Windows, download the latest libxml2-2.x.x.win32.zip file from

<http://www.zlatkovic.com/pub/libxml>

and unzip it.

For Linux, if you have libxml2 installed then xmllint command should be available, or check with your Linux System Administrator.

Copy the UIO_Config.rng file from the Apache Universal Installation Option distribution and run following command:

```
xmllint --noout --relaxng UIO_Config.rng <your config xml file>
```

And fix any errors that are reported.

- The Oracle Adaptive Access Manager Proxy for Apache is not working or intercepting request.

Problem: The following error appears:

```
Failed to create session in memcached, err = 70015(Could not find specified
socket in poll list.) proxy - Failed to create session, cannot process this
request distsessions - memcache server localhost create failed 111
```

Possible Solutions:

- Make sure "memcache" is installed and configured.
- Make sure "memcache" process is up and running before creating the session.

Oracle Adaptive Access Manager Debug Mode

In debug mode, the value of any variable--username, password, and any other information--is not displayed. In capture mode, the HTTP traffic is shown. Therefore, capture mode is not recommended in production.

In-Session/Transaction Analysis

The Oracle Adaptive Access Manager proxy is a solution for login security only. It does not support in-session capabilities. Options are provided below based on possible requirements:

- If you are using a packaged application you do not have access to alter/integrate with, the Oracle Adaptive Access Manager Proxy or Oracle Access Manager are options for real-time/in-line use cases like anti-malware, anti-phishing, risk-based authentication in the login flow.
- If you have the ability to integrate with the application and require in-session/transactional use cases, then consider native integration. This is the most flexible option for this case.
- If you want in-session/transactional use cases but do not have the ability to integrate with the application, a custom option could potentially be possible using either Oracle Adaptive Access Manager offline 10g or Oracle Adaptive Access Manager with a listener.

15.2 Virtual Authentication Devices

Accessible Versions of the Virtual Authentication Devices

Question/Problem: Users who access using assistive techniques need to use the accessible versions of the virtual authentication devices. How do I enable these versions?

Answer/Solution: Accessible versions of the TextPad, QuestionPad, KeyPad and PinPad are not enabled by default. If accessible versions are needed in a deployment, they can be enabled using the Properties Editor in OAAM Admin or using the Oracle Adaptive Access Manager extensions shared library.

The accessible versions of the virtual authentication devices contain tabbing, directions and ALT text necessary for navigation via the screen reader and other assistive technologies.

To enable these versions, set the "is ADA compliant" flag to true.

For native integration the property to control the virtual authentication device is

```
desertref.authentipad.isADACompliant
```

For Oracle Adaptive Access Manager out-of-the-box, the property to control the virtual authentication device is

```
bharosa.uio.default.authentipad.is_ada_compliant
```

Visible Text Input or Password (Non-Visible) Input Setting

Question/Problem: How can I configure QuestionPad so that challenge answers can be enter as non-visible text?

Answer/Solution: Add the following property to `client_resource_<locale>.properties`. This property determines whether the QuestionPad is set for visible text input or password (non-visible) input.

```
bharosa.authentipad.questionpad.datafield.input.type
```

Valid values are text and password.

KeyPad or PinPad for KBA challenges?

Question/Problem: Can I use KeyPad or PinPad for KBA challenges?

Answer/Solution: KBA is designed for use with QuestionPad or plain HTML. Using KeyPad or PinPad is not recommended because KBA questions are not presented in that scenario.

How can the virtual authentication devices protect users from screen capture malware?

Question/Problem: How can virtual authentication devices protect users from screen capture malware?

Answer/Solution: These attacks currently require a manual process. An individual must look at the video or images captured to figure out the PIN or password. The virtual devices are primarily aimed at preventing automated attacks that affect large numbers of customers. If the Trojan did include OCR technology, finding the characters clicked on KeyPad and PinPad would be more difficult to read than other types of onscreen keyboards since Oracle Adaptive Access Manager keys are translucent so that background image can be seen and the font and key shapes can be randomized each session.

Also, the jitter would complicate the task. The virtual authentication devices are a good mix of security and usability for large scale deployments that want to keep the authentication already used and layer more security on top of it. Even if there were malware developed that is capable of deciphering the password, it does not necessarily cause fraud to occur. The virtual authentication devices are only one component of the full solution. Even if a fraudster has the PIN or password, he will have to pass the real-time behavioral/event/transactional analysis and secondary authentication. Oracle Adaptive Access Manager tracks, profiles and evaluates users/devices/locations activity in real-time regardless of authentication. Oracle Adaptive Access Manager takes proactive action to prevent fraud when it detects high risk situations. In this way, fraud could be prevented even if the standard form of authentication (password/PIN/etc.) is removed from the applications

KeyPad Troubleshooting

Question/Problem: I am having trouble with KeyPad. How should I troubleshoot the problem?

Answer/Solution: Refer to the following list:

KeyPad does not display.

- Check the property:

```
bharosa.authentipad.image.url=kbimage?action=kbimage&
```


- Make certain that the client application is pointing to the correct server application.

Buttons stop jittering.

- Someone has changed the KeyPad settings. Check with your server personnel regarding property modifications they may have made.

Same image displayed to all users.

- Check the properties file to make sure that the backgrounds directory setting is correct.

No image displayed in pad background.

- User may have images disabled in the browser.
- Users image may have been deleted from the backgrounds directory.
- Check the properties file to make sure that the backgrounds directory setting is correct.
- Check that the system is configured to assign images for personalization.

15.3 Configurable Actions

Moving Configurable Action from testing environment to a production environment

Question/Problem: I defined a custom configurable action in the test environment and now I want to move the custom action template from test and to production.

Answer/Solution: To do this:

1. Use the Oracle Adaptive Access Manager extensions shared library to package the jar.
2. Add the jar to "oaam-extensions\WEB-INF\lib" folder.
3. Rejar oracle.oaam.extensions.war.
4. Deploy the jar.

Refer to [Chapter 12, "Customizing Oracle Adaptive Access Manager."](#)

15.4 One-Time Password

Are numeric/alphanumeric and pluggable random algorithms supported?

Question/Problem: Are numeric/alphanumeric and pluggable random algorithms supported in OTP?

Answer/Solution: OTP is configurable with a set of two properties:

```
# Length of the Pin
bharosa.uio.otp.generate.code.length = 5
# Characters to use when generating the Pin
bharosa.uio.otp.generate.code.characters = 1234567890
```

The pin generation method is in the base class (AbstractOTPChallengeProcessor), allowing integrators to override the generateCode method.

15.5 Localization

Customize and localize the virtual devices

Question/Problem: Can I make customizations and localize the virtual authentication devices?

Answer/Solution: The virtual authentication devices are provided as "samples" to use if you choose to. These samples are provided in English only. Source art and documentation are provided to allow you to develop your own custom virtual authentication device frames, keys, personalization images and phrases. Localization is included in these customizations. Custom development is not supported. Localization of the KeyPad may have issues since not all languages have the same number of characters. Portuguese for example has special characters not found in English. The key layout may be a bit different when these character keys are added. When adding keys to the layout it is vital that there is still enough free space around the keys to allow the "jitter" to function. General best practice is a space at least as large as a single key all the way around the bank of keys when they are positioned in the center of the jitter area. The source art contains notes with the pixel sizes for this area.

Alteration of these samples is considered custom development.

The "Pad" frame and key images

The frame and key samples are provided in English only. Master files for the virtual authentication device frames and keys along with descriptions of the parts are provided on request. You may create your own custom frame and key images and deploy them using product documentation. Any and all alterations to these images or the properties that correspond to them are considered custom development. Some issues to be careful of here are text, hot spot, key sizes. It is not recommended that these be made smaller than the provided samples.

Background images and phrase text

A set of sample images are shipped with Oracle Adaptive Access Manager. These images are for use in the virtual authentication devices only. For security reasons they should never be available to end users outside the context of the virtual authentication devices. The content, file sizes, and other attributes were optimized for a broad range of user populations and fast download speed. The sample phrase text for each supported language is provided with the package. Any and all alterations to these images or text is considered custom development. If the images are to be edited, make sure not to increase the physical dimensions or change the aspect ratio of the sample images because distortions will occur. Also, there must be an identically named version of each image for each virtual authentication device used in your deployment.

Images displayed during registration

Question/Problem: The images displayed in the page before user registration appear in English instead of the locale language.


ORACLE

Váš nový bezpečnostní profil
 Nastavením nového bezpečnostního profilu zlepšíte svou online ochranu. Přidá k vašemu účtu nové vrstvy zabezpečení, které nám pomohou vás identifikovat a vám zase identifikovat naše webové stránky.

Krok 1: Bezpečnostní obrázek a fráze

Vylepšené zabezpečení dat
 Vaše nová přizpůsobená bezpečnostní zařízení vás pomohou chránit při používání online bankovníctví. Zadané informace jsou chráněny před většinou dnešních hrozeb zabezpečení. Současně jsou obrázek, fráze a datum dokladem, že jste na našich oficiálních stránkách.

This is an example of a personalized TextPad



Krok 2: Bezpečnostní otázky a odpovědi

Další vrstva zabezpečení
 Registrací tří bezpečnostních otázek přidáte další vrstvu zabezpečení. V budoucnu vám položíme jednu z těchto otázek pomocí vašeho přizpůsobeného zabezpečení, pokud se situace zdá riziková. Tyto otázky a odpovědi by měly být stejně tajné jako vaše heslo.


Questions (Choose a question from each set):

1. [What year was your significant other born?]

2. [What was your first name?]

3. [What was the year of your favorite sports?]

Answers



Zaregistrujte svůj profil zabezpečení ještě teď >> Pokračovat

Copyright (c) 2010, Oracle a její přidružené společnosti. Všechna práva vyhrazena.

Answer/Solution: Globalized virtual authentication device image files including the authentication registration flows are not provided. The deployment team develop these.

15.6 Man-in-the-Middle/Man-in-the-Browser

Question/Problem: I use mobile transaction authentication number to sign each transaction using an OTP via SMS. SMS costs are high. How can Oracle Adaptive Access Manager help? In addition, I want a solution that protects against Man-in-the-Middle (MiTM)/Man-in-the-Browser (MiTB) attacks.

Answer/Solution:

1. Use Oracle Adaptive Access Manager to assess risk and base the use of secondary authentication such as mTAN on risk. Then, SMS can be sent for transactions that are medium to high risk instead of all transactions.
2. One of the best ways to protect against MiTM and MiTB is to perform transactional risk analysis. For example, check to see if the target account has ever been used by this user before or if the user has ever performed a transfer over set dollar amount thresholds. To perform transactional analysis in real-time today requires native integration with the Web application.
3. Use PinPad to input the target account number. This ensures that the account number entered by the user cannot be easily changed in a session hijacking situation. The account number is not sent over the wire and cannot be easily altered by a MiTM/MiTB.
4. It is recommended that KeyPad and PinPad virtual authentication devices always be used over HTTPS. The virtual authentication devices send the one time random data generated on the end-user's machine (mouse click coordinates) to the server

to be decoded and HTTPS provides the traditional encryption in addition. No client software or logic resides on the end-user's machine to be compromised.

5. With Oracle Adaptive Access Manager extremely high risk transfers can be blocked all together. Blocking high risk transfers reduces the fraud regardless of the authentication methods used.

Index

A

access and password management Integration, 13-1
AppUtil.AssignNewImageAndCaption, 3-16
AppUtil.CreateAuthentiPad, 3-14, 3-15
AppUtil.getAuthentiPadHTML, 3-14, 3-15
AppUtil.InitTracker, 3-13, 3-15
AppUtil.InitUser, 3-12, 3-13, 3-14
AppUtil.RunAuthentiPadRules, 3-14, 3-15
AppUtil.RunChallengeUserRules, 3-16
AppUtil.RunPostAuthRules, 3-13, 3-14, 3-15
AppUtil.RunPreAuthRules, 3-14, 3-15
AppUtil.SetAuthMode, 3-16
AppUtil.UpdateAuthStatus, 3-13, 3-14, 3-15, 3-16
ASP.NET applications integration, 3-1, 3-11
ASP.NET sample applications, 3-11
authenticateQuestion, 2-13
authenticator caption, configuring words used
in, 7-13

B

Backspace Key Hotspot
 KeyPad, 7-8
 PinPad, 7-10
bharosa_pad.js, 3-9
Bharosa_SDK_DotNet2.0.zip, 3-2
bharosa_server.properties, 7-1
bharosa.cipher.client.key, 3-10
BharosaClient.getAuthentiPad(), 3-8
BharosaSOAPURL key, 3-2
bharosa.swf, 3-12, 3-13, 3-15
BharosaUtils.exe, 3-10
bulk transactions, creating and updating, 3-7

C

cancelAllTemporaryAllows, 4-12
Caps States
 KeyPad, 7-8
challenge failure counters, reset, 3-8
Challenge Pad used for challenge, configuring, 9-6
Challenge Processor, creating, 9-2
Challenge Processors, 9-1
challenge questions, validating user, 3-7
Challenge with QuestionPad flow (S6), 2-12

Challenge.jsp, 2-12
Check Challenge Question Answer flow (C3), 2-13
Check Question Registration for User flow (C2), 2-11
clearSafeDeviceList, 4-10
client_resource.properties, 7-1
com.bharosa.vcrypt.tracker.dynamicactions.intf.Dyna
 micAction java interface, 8-1
Configurable Actions
 executing in order and data sharing, 8-2
 integration, 8-1
 JUnit code example, 8-3
CookieManager.aspx, 3-12, 3-13, 3-15
createAuthentiPad, 2-7, 2-9, 2-12
createPersonalizedAuthentiPad, 2-7, 2-9, 2-12
createTransaction, 4-3
Custom Challenge Processor as Challenge Type, 9-5
custom login page, 2-5
Customizing Oracle Adaptive Access Manager, 12-1

D

Decode Virtual Authentication Device Input flow
 (P4), 2-9
decodeKeyPadCode, 2-10
decodePadInput, 2-10
deployment, UIO, 5-2
Developer's Guide, introduction, 1-1
Device Fingerprint flow (F2), 2-5
Device Fingerprinting, 10-1
device ID, Rules Engine return, 3-7
device registration, enable, 11-1
device registration, enabling, 1-3
Display TextPad or KeyPad flows (S4 and S5), 2-9

E

encryptImageToStream, 2-9
Enter Key Hotspot
 KeyPad, 7-7
 PinPad, 7-9
 QuestionPad, 7-6
 TextPad, 7-3
enumeration definition, 3-3
Extensions Shared Library, 12-1

F

fingerprinting device, 2-5
forward proxy, 5-2

G

Generate Non-Personalized TextPad flow (P2), 2-7
Generate Personalized TextPad or KeyPad flow (P3), 2-8
Generic TextPad, 2-7
getActionCount, 4-13
getAuthentiPad, 2-7, 2-12
getFinalAuthStatus, 4-12
getHTML, 2-7, 2-9, 2-12
GetImage.aspx, 3-14, 3-15
getRulesData, 4-13
getSecretQuestions, 2-12
getUserByLoginId, 2-7, 2-9, 4-7

H

handleChallenge.jsp, 2-13
handleFlash.jsp, 2-6
handleJump.jsp, 2-6
handlePassword.jsp, 2-10, 2-11
handleTrackerRequest, 4-2
handleTransactionLog, 4-4

I

IBharosaProxy.createTransactions(), 3-7
IBharosaProxy.updateTransactions(), 3-7
imageToStream, 2-9
integration
 native, 2-1
 native and web services, 1-2
 static linked, 1-2
 static-linked, 2-2
integration options
 Adaptive Risk Manager and KBA Scenario, 2-14
 virtual authentication devices and KBA scenario, 2-3
IsDeviceMarkedSafe, 4-10

K

kbimage.jsp, 2-7, 2-9
KeyPad, 2-8, 7-6

L

Landing or Splash Page, 2-13
Lifecycle Management Changes, 14-1
Lock Out page, 2-13
LoginHandlerPage.aspx, 3-13, 3-15
LoginJumpPage.aspx, 3-12, 3-13, 3-14
LoginPage.aspx, 3-12, 3-13, 3-14, 3-15, 3-16

M

markDeviceSafe, 4-10
memcache, configuring, 5-13
multi-factor authenticator, adding, 5-34

N

native and web services integration, 1-2
native API options, 2-1
native integration, 2-1
 Java, 4-1
 .NET, 4-1
 .NET applications, 3-1
 SOAP service wrapper API, 2-2
Native integration Java
 Customer Care, 4-12
 Rules Engine, 4-10
native integration .NET
 application configuration, 3-2
 architecture, 3-1
 configuration property files, 3-2
 encrypting property values, 3-10
 installing SDK, 3-2
 Rules Engine, 3-6
 troubleshooting, 3-10
 virtual authentication devices, 3-8
native integration options, 2-3
 .NET API, 4-1
 .NET API, tracing messages, 3-10
 .NET API, using, 3-1

O

OAAM Server, 5-2
 customizing user interface branding, 6-4
 determining default user groups, 6-3
 KeyPad, 7-6
 PinPad, 7-8
 properties, 6-5
 QuestionPad, 7-4
 TextPad, 7-2
OAAM Server interface, proxy, 5-29
OAAM Server Web application, 6-1
One Time Password (OTP), 9-1
Oracle Access Manager, 13-1
Oracle Adaptive Access Manager APIs, 4-2
Oracle Adaptive Access Manager properties, override, 4-1
Oracle Adaptive Access Manager's Universal Installation Option, 5-1
Oracle Identity Manager, 13-1
OTP Integration, 9-1
OTP User Information Properties, 9-6

P

Password Status flow (C1), 2-11
password.jsp, 2-7, 2-9, 2-12
PasswordPage.aspx, 3-13, 3-15
personalized KeyPad, 2-8

- personalized TextPad, 2-8
- Phrase (Caption)
 - KeyPad, 7-7
 - PinPad, 7-9
 - QuestionPad, 7-5
 - TextPad, 7-3
- PinPad, 7-8
- Post-Authentication rules, 2-11
- Pre-Authentication rules, 2-6
- Pre-Authentication Rules flow (R1), 2-6
- processPatternAnalysis, 4-9
- processRules, 2-6, 2-7, 2-11, 2-12, 2-13, 4-11
- proxy
 - application discovery, 5-31
 - get-server action, 5-26
 - global variables, 5-27
 - interception process, 5-28
 - OAAM Server interface, 5-29
 - post-server action, 5-26
 - pre-defined request variables, 5-28
 - redirect-client action, 5-25
 - request variables, 5-27
 - scenarios, 5-33
 - send-to-client action, 5-26
 - send-to-server action, 5-26
 - session variables, 5-27
- proxy conditions, 5-19
- proxy configuration, 5-18
- proxy filters, 5-22
- Proxy for Apache, 5-8
 - ConfigFile, 5-16
 - configuring httpd.conf, 5-14
 - GlobalVariable, 5-16
 - httpd requirements, 5-10
 - log4j.xml, 5-16
 - Memcache, 5-16
 - UIO_log4j.xml, 5-18
 - UIO_Settings.xml, 5-15
- Proxy for Apache settings, 5-16
- Proxy for Microsoft ISA
 - configuration files settings, 5-6
 - configuration reload settings, 5-7
 - proxy Web publishing configuration, 5-4
 - registering for Microsoft ISA DLL, 5-6
 - session ID cookie settings, 5-7
 - session inactive interval settings, 5-7
 - troubleshooting settings, 5-8
- Proxy for Microsoft ISA installation, 5-4
- proxy interceptors, 5-19
- proxy variables, 5-26
- proxy.authenticateQuestion, 3-16
- proxy.getSignOnQuestions, 3-16

Q

- Question Text
 - QuestionPad, 7-6
- QuestionPad, 7-4

R

- RegisterImagePhrase.aspx, 3-16
- RegistrationOptional, 3-16
- resetChallengeFailureCounters(), 3-8
- resetUser, 4-13
- reverse proxy, 5-2
- Run Challenge Rules flow (R5), 2-13
- Run Virtual Authentication Rules flow (R2), 2-6
- runPostAuthRules, 2-11
- runPreAuthRules, 2-6

S

- SampleKBATracker application, 3-14
- SampleWebApp application, 3-12
- SampleWebAppAuthTracker application, 3-13
- SampleWebAppWithTracker application, 3-12
- setTemporaryAllow, 4-12
- SOAP service wrapper API (for Java or .NET applications), 2-1
- static linked integration, 1-2
- static-linked integration, 2-2
- static-linked library for Java applications, 2-1
- Success.aspx, 3-12, 3-13, 3-14, 3-16

T

- TextPad, 2-8, 7-2
- Timestamp
 - KeyPad, 7-7
 - PinPad, 7-9
 - QuestionPad, 7-5
 - TextPad, 7-3
- transaction details collection API, 3-5
- transient page, 2-5

U

- UIO deployment, 5-2
- Universal Installation Option, 5-2
- Update Authentication Status flow (P5), 2-10
- updateAuthStatus, 2-10, 2-13, 4-7
- updateLog, 2-6, 4-6
- updateStatus, 2-10
- updateTransaction, 4-4
- updateTransactionStatus, 4-5
- user details in its database, storing, 3-4
- user login information, capturing, 3-5
- user login status, capturing, 3-5
- User Name Page (S1) flow, 2-5
- user session attributes, capturing, 3-5

V

- Validate User and Password flow (CP1), 2-10
- validateAnswer, 2-13
- VCryptResponse, 4-2
- Virtual Authentication Device display
 - configuration, 7-2
- Virtual Authentication Device KeySet, 7-11

- Virtual Authentication Device properties, 7-1
- Virtual Authentication Device property files, 7-1
- virtual authentication device, embedding, 3-9
- virtual authentication device, validating user, 3-9
- virtual authentication devices
 - creating, 3-8
 - embedding in a Web page, 3-9
- virtual authentication devices, in ASP.NET applications, 3-8

W

- Web Listener creation, 5-4
- Web publishing rule creation
 - for OAAM Server, 5-5
 - for protected Web applications, 5-5
- Web publishing rules and listeners, 5-4
- Web publishing rules creation, 5-5